Universiteit
Leiden
The Netherlands

Multi-dimensional feature and data mining

Georgiou, T.

Citation
Georgiou, T. (2021, September 29). *Multi-dimensional feature and data mining*. Retrieved from https://hdl.handle.net/1887/3214119

Version:	Publisher's Version
License:	Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden
Downloaded from:	https://hdl.handle.net/1887/3214119

Note: To cite this publication please use the final published version (if applicable).

Multi-dimensional feature and data mining

Theodoros Georgiou

Cover Design: Andreas Noussas

Copyright ©by 2021 Theodoros Georgiou. All rights reserved.

ISBN: 978-90-9035132-2

The research leading to this thesis was funded by the the research program DAMIOSO (Data Mining on High Volume Simulation Output) with project number 628.006.002, which is partly financed by the Netherlands Organization for Scientific Research (NWO) and partly by Honda Research Institute-Europe (GmbH).

Multi-dimensional Feature and Data Mining

Proefschrift

ter verkrijging van
de graad van doctor aan de Universiteit Leiden,
op gezag van rector magnificus prof. dr. ir. H. Bijl,
volgens besluit van het college voor promoties
te verdedigen op woensdag 29 september 2021
klokke 13.45

door

Theodoros Georgiou

geboren te Athene, Griekenland
in 1989

Promotiecommissie

Promotors:	Prof. dr. T.H.W. Bäck	
	Prof. dr. M.S.K. Lew	
Overige leden:	Prof. dr. B. Sendhoff	Technical University Darmstadt
	Prof. dr. Z. Zhang	Binghamton University
	Prof. dr. A. Plaat	
	Prof. dr. K.J. Batenburg	
	Prof. dr. ir. N. Mentens	
	Dr. M. Baratchi	

Acknowledgements

I started my doctoral journey in 2016 at Leiden University. During that time I have been blessed with wonderful colleagues and supervisors that have helped me both professionally and personally.

I want to start by thanking my promotors. They have helped me from the beginning until the end, by giving me the opportunity to pursue my PhD and guided me in the steps of scientific research.

I want to thank my fellow students in the Natural Computing group, the Media Lab as well as other colleagues from the university with which we had many constructive discussions both regarding our work as well as personal matters. I hope we will continue to collaborate and support each other in the future.

I want to thank all the people, at the Honda Research institute - Europe for their support and advice. I want to give special thanks to Sebastian Schmitt and Markus Olhofer fore their time and valuable discussions.

Special thanks to Andreas Noussas for his great work with my cover and Jan van Rijn for his help with the Dutch translation of the summary.

I want to thank all of my friends for their support during this time without whom I might have not gone through this journey.

I also want to give special thanks to Orla Rodgers for her help and support during the last stretch, essential for the completion of this thesis.

Finally, I want to give my gratitude to my family. Their support, not only these years but throughout my life, put me in the position to be able and both start and finish this chapter of my life. Without you I wouldn't have come this far.

Contents

1 Introduction

1.1 Background

Computer vision is a very broad field with numerous applications in scientific, medical and ”everyday life” domains. Human understanding of the world through visual queues extends to three as well as four dimensions, e.g. understanding an action such as falling, hugging etc. In recent years people have been generating and consuming a plethora of multimedia content which has higher dimensionality than the classic two dimensional image, i.e. three and four dimensions. This usage has given rise to many exciting and important applications, such as autonomous vehicles and automated medical 3D & 4D image analysis. These applications both have potentially high impact in our society, but also introduce complex challenges, making the process of such high dimensional data the leading edge of modern computer vision.

In the meantime, the increasing computational capacity of modern hardware, has led to an unprecedented capability of simulating complex fluid dynamics systems, i.e. computational fluid dynamics (CFD) simulations, such as the air around objects, the mix and tumble of air and gas in an internal combustion engine et cetera. These simulations can produce an enormous amount of data, in multiple dimensions, e.g. 4, and many modalities, i.e. a plethora of information for every physical location. The increasing availability of such data has given rise to new applications. For example, is it possible to retrieve engineering designs from a large database based on the flow similarities? Is it possible to utilize these large flow fields to automatically optimize the designs? Such applications might help engineers and scientists understand better the correlation of design principles to the flow and the flow patterns to performance.

For the purpose of machine learning and more specifically computer vision, CFD simulation output shares many similarities to visual data. They both represent the real

world in a similar manner. The physical (2-4D) space is separated by a grid, and each individual cell is assigned values representing the real world. In the case of images, these values are the RGB colors, for depth images they are (usually) the distance to the sensor and in the case of CFD simulation output properties of the flow, such as the velocity vector, pressure etc. This similarity between the representation of the data deems possible the adaptation of computer vision techniques to processing CFD simulation output, as it is a much more mature research field.

1.1.1 Feature extraction of CFD simulation output

There exist a plethora of feature extraction methods for flow fields, the vast majority of which are focused on visualization. Good overviews of the flow field feature visualization are [83, 229, 293, 360, 273, 390]. The steps towards feature visualization can be divided into feature definition, decomposition, extraction and visualization. All of these steps can be categorized into two main categories, steady and unsteady (i.e. time dependent) flow field. According to our research, most of unsteady flow feature visualization focuses on tracking steady flow features in time. There are a few exceptions to this rule, such as the path lines, which are specific unsteady flow features.

There are two main categories of steady flow features, local and global features. Local features have specific local behavior and are mathematically defined. Although this is the case, there are many algorithms that try to extract them, all with their limitations and advantages. These features are defined around points in the flow where the flow "vanishes", i.e. the magnitude of the vectors of the vector field becomes zero. These features can be categorized according to the behavior of the flow around the "vanishing" point. The categorization was introduced by Helman and Hesselink in 1989 for 2D [155] and 1991 for 3D [156]. The field of extracting and visualizing these features was created in the same work and called Vector Field Topology (VFT). VFT is now one of the most established ways of visualizing and analyzing flow field behavior.

Global features, unlike local usually have vague definitions and their detection depends on the specific implementation and application. Some examples are the vortex, flow separation and shock waves. For example, a vortex refers to the swirling motion of a fluid around a specific point [229]. An example definition of the vortex is given by Robinson [294]:

"A vortex exists when instantaneous streamlines mapped onto a plane normal to the vortex core exhibit a roughly circular or spiral pattern, when viewed from a reference frame moving with the center of the vortex core."

This swirling motion is understood through visual observations and is hard to be mathematically defined. For example how much ”swirliness” is enough to categorize a vortex? How are the boundaries of the vortex defined? These issues make the detection of such features difficult and the implementation usually varies according to the needs of the application.

The limitations described above make the use of flow visualization defined flow field features very hard to use for the purpose of machine learning, where the known algorithms need specific definitions to operate.

1.1.2 Computer vision

Computer vision has been pushing the limits of automated image understanding for decades. The main focus of it is to extract high-level information from visual content such as images, with applications varying from image classification [184] and object detection [100], to scene understanding [334], localization and mapping [351, 352] and many more. Throughout the years, the approaches followed by computer vision researchers have changed significantly. Popular approaches have been global features and description of images such as textures, and color histograms [357, 120]. These features were very computationally efficient which was very attractive for the computational capacity of hardware at that time. As years progressed though, these approaches showed their limits as they are very sensitive to occlusion and clutter. Moreover, local information such as object shapes are disregarded making the distinction between similar objects (e.g. red car vs red motorbike) infeasible.

To deal with such issues, local features are introduced. These approaches follow several steps. First detection of more informative points and regions in an image, then description of such areas and finally feature matching, or aggregation for global description. Some popular examples of local features are the SIFT [225, 226], SURF [19], FREAK [7] and the ORB [299]. These features encode local information, such as histograms of image gradients in a neighborhood, or pixel differences for different point patterns in a neighborhood. These features have enabled applications such as object or scene matching, using algorithms such as the RANSAC. Meanwhile, using feature aggregation to create image global descriptions has enabled applications such as image classification, and content based image retrieval [341].

In recent years the focus has moved to artificial neural networks (ANN) and more specifically deep learning and convolutional neural networks (CNN). Deep learning models boosted the performance of computer vision systems by a large margin [184, 368]. The modern availability of large scale datasets as well as the high computational capabilities of modern GPUs has rendered the training of huge models

feasible. Deep learning models are hierarchical models that, in contrast to the more traditional local features description, learn representations solely from the data that they are trained on. Very important stepping stones, that made the success of these models possible is early research done on optimization algorithms, such as the gradient descent and back-propagation [301, 202]. The back-propagation algorithm enables the propagation of error from one layer to the next (or previous, hence "back") in hierarchical models and thus updating the parameters of these layers. Later, the introduction of convolutional layers made possible the reusability of parameters, minimizing the total number of free parameters and thus making that training of huge models possible [199, 201]. Today, deep neural networks are the most popular approach for high performance tasks, varying from object classification [445], image retrieval [74], scene semantic segmentation [111] and even generating new content like style transfer approaches [221] and swapping faces [181].

With the increase of available higher dimensional content, such as video [340], RGB-D images [344] and CAD models [419], the need to extract high-semantical information from them became prevalent. As we will see in Chapter 2, the same trends with the approaches applied on two dimensional images, are followed on the higher dimensions.

The vast amount of research done in extracting high level information from a data source so similar to CFD simulation output motivates us to explore the adaptation of the core ideas to the high dimensionality of CFD simulation output.

1.2 Research questions

In this thesis we focus on high dimensional computer vision (i.e. higher than the two dimensional image), and more specifically on extracting meaningful features from CFD simulations output. To achieve our goal, we focus on the following research questions:

RQ1: How are computer vision approaches being generalized to deal with higher dimensionality problems?

Computer vision methodologies are not applied only on the traditional two dimensional image. A lot of research areas and applications concentrate on higher than two dimensional data, such as video processing or RGB-D images. These areas can may seem to be disconnected from areas focusing on two dimensional data. We want to investigate, to what extent methodologies are extended from the two dimensional case to the higher dimensions, as well as common practices and pitfalls these higher

dimensionality methodologies share.

RQ2: Can deep learning techniques represent flow fields, in a meaningful manner?

Deep learning techniques are the focus of modern computer vision. As the dimensionality of the datasets increases, the complexity of the patterns needed to be identified is increasing as well. From our previous research question we already got a glimpse of how deep learning approaches are able to handle the increase of the number of dimensions. Nonetheless, the output of CFD simulations has one of the highest dimensionalities we have seen so far. Our question then becomes, how can we best apply deep learning approaches in such complicated data towards maximizing performance?

RQ3: Can local feature based approaches represent flow fields in a meaningful way?

Deep learning approaches, although very powerful, require a vast amount of data to be trained on. Most hand crafted local feature based approaches though were proposed and evolved before the modern huge datasets were available. One of the main differences to deep learning is that the low level features, or encodings are not learned from the data as with deep learning, but are predefined by scientists. Thus, in theory, one only requires enough data to learn high level correlations. Meanwhile, CFD simulation complexity can vary. As the simulation complexity increases, the amount of time required to produce the simulation output increases as well. In some applications, like the flow in a cylinder of a internal combustion engine, it can even take a month to compute on high core count clusters. Thus, acquiring many examples of such high complexity simulations to train deep neural networks is infeasible. Therefore, we want to investigate whether the, more traditional, hand crafted features are capable of representing the flow fields in a meaningful manner and how they compare to deep learning approaches, especially when the number of training data is limited.

RQ4: Can we take advantage of the vector field representation to construct a more efficient convolutional operator?

A large part of the CFD simulation output is the velocity vector field. Convolutional neural networks perform scalar convolutions regardless of the input. Can we take advantage of the fact that the input to the convolution is a vector field? What extra information can be extracted? Can we utilize any extra information in a deep learning

framework?

RQ5: How can we better regularize deep neural networks, to reduce overfitting and increase the convergence speed?

There is a variety of limiting factors on the performance of artificial neural networks. Many of them are related to optimization inefficiencies. Some examples are the covariant shift, the exploding and vanishing gradients as well as the scaling-based weight space symmetry. All existing approaches have their limitations. Usually, while trying to solve an issue we are introducing another. For example, by applying orthogonalization, the learning capacity of each layer is limited. Therefore we are interested in investigating whether we can efficiently regularize the weight learning such that performance is maximized.

1.3 Dissertation outline

This dissertation is structured according to the research questions defined in Section 1.2. In Chapter 2 a wide literature study is conducted on how high dimensional data is used in computer vision. Moreover, the approaches are clustered according to (i) the data they are applied to, (ii) whether they are deep learning approaches or traditional hand crafted feature based approaches and (iii) what kind of increase of dimensionality they are tackling, i.e. increase of physical dimensions or increase of amount of information per physical point. Finally, we identify the most popular datasets and benchmarks concentrating on higher than two dimensional data and describe the most studied research areas and discuss the respective state of the art approaches.

In Chapter 3 we construct a large scale 3D CFD simulation dataset, which focuses on the air flow around a passenger car. Using this dataset as a benchmark, a number of deep learning approaches, tailored to the specific high dimensional data are proposed and evaluated, tackling RQ2. Chapter 4 then tackles RQ3, evaluating hand crafted based approaches and comparing them to deep learning approaches. Since in computer vision there is a much larger variety and more generic two dimensional feature detectors and descriptors, whilst two dimensional data require less computational time, we decided to first evaluate using two dimensional data and potentially move to three. Thus, we constructed another dataset which consists of 2D flows of air around an airfoil.

In Chapter 5 we investigate whether its possible to take advantage of vector field representation to gain more information than the response of scalar convolution. We define a new operator that takes advantage of the vector field representation

and show that its applicable to more standard computer vision problems as well. In Chapter 6 we proposed a new weight regularization method that tackles some of the issues mentioned in RQ5 and test it on popular benchmarks and architectures. Finally, in Chapter 7 the conclusions of this dissertation are presented and potential future directions discussed.

1.4 Dissertation contributions

The main contributions of the author of this dissertation are the following:

Theodoros Georgiou, Yu Liu, Wei Chen, and Michael Lew. A survey of traditional and deep learning-based feature descriptors for high dimensional data in computer vision. *International Journal of Multimedia Information Retrieval (IJMIR)*, pages 1–36, 2019

Theodoros Georgiou, Sebastian Schmitt, Markus Olhofer, Yu Liu, Thomas Bäck, and Michael Lew. Learning fluid flows. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018

Theodoros Georgiou, Sebastian Schmitt, Nan Pu, Wei Chen, Thomas Bäck, and Michael Lew. Comparison of deep learning and hand crafted features for mining simulation data. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*. IEEE

Theodoros Georgiou, Sebastian Schmitt, Thomas Bäck, and Michael Lew. Orientational equivariant neural networks using clifford convolutions
(Submitted for publication at Neurocomputing, Elsevier)

Theodoros Georgiou, Sebastian Schmitt, Wei Chen, Thomas Bäck, and Michael Lew. Norm loss: An efficient yet effective regularization method for deep neural networks. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*. IEEE

1.5 Other work by the author

Yu Liu, Yanming Guo, **Theodoros Georgiou**, and Michael Lew. Fusion that matters: convolutional fusion networks for visual recognition. *Multimedia Tools and Applications*, 77:1–28, 2018

Umut Özaydın, **Theodoros Georgiou**, and Michael Lew. A comparison of cnn and classic features for image retrieval. In *International Conference on Content-Based Multimedia Indexing (CBMI)*, pages 1–4, 2019

Yanming Guo, Yu Liu, **Theodoros Georgiou**, and Michael Lew. A review of semantic segmentation using deep neural networks. *International Journal of Multimedia Information Retrieval (IJMIR)*, 7:87–93, 2018

Nan Pu, **Theodoros Georgiou**, Erwin M Bakker, and Michael Lew. Learning a domain-invariant embedding for unsupervised person re-identification. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019

2

Deep learning and traditional approaches for high dimensional data

Higher dimensional data such as video and 3D are the leading edge of multimedia retrieval and computer vision research. In this chapter, a comprehensive overview of the state of the art of higher dimensional features from deep learning and also traditional approaches is given. Moreover, key insights into current research areas and challenges that arise are discussed. Current approaches are frequently using 3D information from the sensor or are using 3D in modeling and understanding the 3D world. With the growth of prevalent application areas such as 3D games, self-driving automobiles, health monitoring and sports activity training, a wide variety of new sensors have allowed researchers to develop feature description models beyond 2D. Although higher dimensional data enhance the performance of methods on numerous tasks, they can also introduce new challenges and problems. The higher dimensionality of the data often leads to more complicated structures which present additional problems in both extracting meaningful content and in adapting it for current machine learning algorithms. Due to the major importance of the evaluation process, we also present an overview of the current datasets and benchmarks. Finally, we make remarks on potential future directions, some of which are tackled in this thesis.

2.1 Introduction

With the current growth of computing systems and technologies, three and four dimensional data, such as 3D images and videos, are becoming a commodity in multimedia systems. Understanding and utilizing this data is the leading edge of modern computer vision. In this chapter we present a comprehensive study (including a categorization) of these high dimensional data types, as well as the methods developed to process them, accompanied with their strengths and weaknesses. Finally, we collect and give an overview of the main areas that utilize such representations.

One of the first steps towards developing, testing and applying methods on high dimensional data is the acquisition of complicated datasets, for instance datasets consisting of 3D models [419, 45], three dimensional medical images and videos (MRI, Ultrasound etc.) [54, 143], large 2D and 3D video datasets for action recognition [235, 324] and more. Different datasets are used for different data mining tasks. For example, object retrieval, movie retrieval and action classification tasks are performed on video data such as movies, YouTube clips et cetera. Clustering and classification tasks are performed on medical images for computer aided diagnostics and surgery. Object classification and detection, as well as scene semantic segmentation are usually applied on RGB-D images and videos retrieved by sensors such as the Microsoft Kinect [444].

We perform two types of categorization. The first is dataset and application driven and the second is method driven. Although these datasets find applications in different fields there are some similarities between the methods used. For example, deep learning techniques are used for 2.5D and 3D object classification (either retrieved from depth maps or designed models), action classification, video retrieval as well as medical applications, for instance landmark detection and tracing in ultrasound video. Histograms of different metrics (e.g. gradients, optical flow or surface normals) are used as features that describe the content of the data.

One of the recent breakthroughs has been the development of new deep learning architectures which could overcome (to some extent) the well known vanishing gradient problem in training. In the case of neural networks, they changed the landscape from typically using a few layers to using hundreds of layers. These methods typically learn the features based on large datasets directly from the raw data and have the least supervision. The other main approach from the literature is the continuation of advances in traditional or ”hand crafted” and ”shallow learning” based features. Features extracted from 2D optical information, e.g. natural images, in computer vision have had a major impact in computer vision and human-computer interaction across many applications [247, 225, 19, 373, 322, 299, 7, 447] and many of the higher dimensional methods were inspired or adapted from the 2D versions. These

approaches usually require significantly more supervision but also can be effective when large training datasets are not accessible.

High dimensional computer vision, with the definition given in this thesis (i.e. higher than 2D), is a very broad field that contains many different research areas, data types and methods. There have been surveys on specific areas within high dimensional computer vision. For example, when it comes to the static world, some of surveys focus on specific research areas such as 3D object detection [112, 306], semantic segmentation [111, 432, 96], object retrieval [79, 362] or Human Action Recognition [274, 168, 132]. Others focus on methodologies such as interest point detectors and descriptors [36, 378, 195], spatio-temporal salient point detectors and descriptors [210] or deep learning [152]. Finally some surveys focus on datasets and benchmarks of a specific research area, such as human action recognition [124]. This chapter differs from these since the focus is on the generalization of methodologies with the increase of dimensionality, regardless of the research area or the type of data. The most relevant work to this was done by Ioannidou et al. [152] where they focus on computer vision on static 3D data. There are two main differences with this work, (i) they focus only on deep learning methods and (ii) they focus only on 3D representation of the static world which means that they neglect the temporal dimension, which is a significant focus of this chapter.

2.2 Deep learning

Deep learning techniques refer to a cluster of machine learning methods that construct a multi-layered representation of the input data. The transformation of the data in each layer is typically trained through algorithms similar to back-propagation. There are several deep learning methods. In this section we will give a summary of the methods that have been used with high dimensional data. The main examples are the Convolutional Neural Networks (CNNs), the Recurrent Neural Networks (RNNs), Auto-Encoders (AE) and Restricted Boltzmann Machines (RBMs). For a detailed overview of deep learning in computer vision the reader is referred to [110] and for a general deep learning overview to [102].

Deep learning approaches can be split into two main categories, supervised and unsupervised methods. Supervised methods define an error function which depends on the task the method needs to solve and change the model parameters according to that error function. These kind of methods provide an end-to-end learning scheme, meaning that the model is learning to perform the task from the raw data. Unsupervised methods usually define an error function to be minimized which depends on the reconstruction ability of the model. Together with the reconstruction error, depend-

ing on the method, an auxiliary error function might be defined which forces some characteristics to the learned representation. For example sparse auto-encoders try to force the learned representation to be sparse, which helps the overall learning procedure and provides a more discriminative representation. The most commonly used deep learning method is Convolutional Neural Networks (CNNs). In the rest of this section we give a brief introduction to the basic deep learning methods and provide an in depth analysis on their generalization from the image domain to the higher dimensional domains.

2.2.1 Basic deep learning methods

2.2.1.1 Convolutional neural networks (CNN)

Convolutional Neural Networks (CNNs) consist of multiple layers of convolutions, pooling layers and activation functions. Usually each layer will have a number of different convolutional kernels, a non-linear activation function and, maybe, a pooling mechanism to lower the dimensionality of the output data. An example of such a layer is shown in Figure 2.1. These networks were initially applied on handwritten digit recognition [199] but got the attention they have today after the introduction of LeNet [200] and more so after Krizhevsky et al.'s [184] work in 2012, where they won the ImageNet 2012 image classification competition with a deep-CNN. This recent success of CNNs highly depends on the increased processing power of modern GPUs as well as the availability of large scale and diverse datasets which made training models with millions of trainable parameters possible.

One of the main drawbacks of deep convolutional neural networks is that they tend to overfit the data. Moreover, they suffer from vanishing and exploding gradients. Resolving these issues have motivated a lot of research in various directions. More specifically, different elements of CNNs are studied and proposed, e.g. activation functions or normalization layers, training strategies and the generic network architecture, for example the inception networks [358]. Most of this research is based on image recognition as the established benchmark due to the availability of large scale annotated datasets such as the ImageNet [302] and the Microsoft COCO [217]. Nonetheless, many of these methods have been generalized and adapted to be applicable to 2.5D and 3D data, such as videos, and RGB-D images.

2.2.1.1.1 Activation functions. One of the main components of the successful AlexNet [184] on the ImageNet 2012 challenge is the Rectified Linear Unit [159, 253] activation function. The output of the function is $\max(0, y)$, where y is the output of a node in the network. The main advantages of this layer is the sparsity it provides to

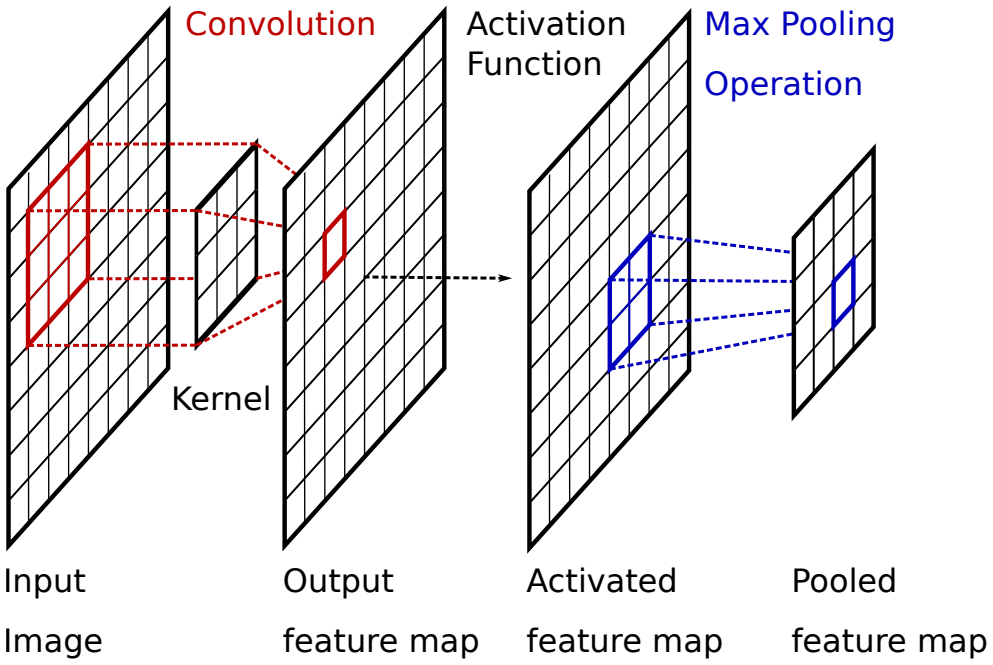


Figure 2.1: Basic CNN block. A single layer is shown which applies a kernel on an input filter followed by an activation function and a max-pooling operation.

the output as well as reduction of the vanishing gradients problem, compared to the more traditional hyperbolic tan and the sigmoid functions [101].

In the past years, many researchers have proposed new activation functions in order to improve the quality of neural networks. Some examples are the leaky ReLU (LReLU) [230], which instead of having always zero as output of negative inputs has a small response proportional to the input, i.e. $\alpha * y$, the Parametric Rectified Linear Unit (PReLU) [126], which learns the parameter α of LReLU, the Exponential Linear Unit (ELU) [53] and its trainable counterpart Parametric ELU (PELU) [381], and many more [5, 103, 163, 176]. For a more detailed overview of activation functions the reader is referred to [381].

2.2.1.1.2 Normalization. Experimental results suggest that when networks have normalized inputs, with zero mean and standard deviation of one, they tend to converge much faster [184]. In order to take advantage of this finding it is a common practice to rescale and normalize the input images [184, 338, 146]. Besides the input normalization, many researchers try to also normalize the input of individual layers,

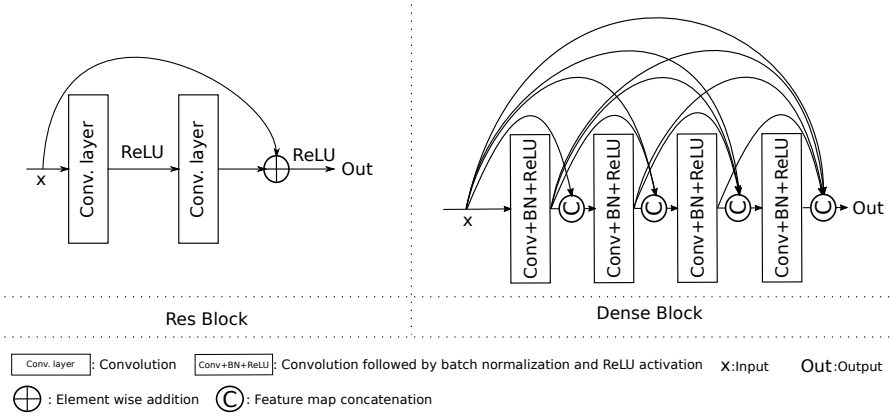


Figure 2.2: On the left is the ResBlock, the building block of ResNet [127]. After two convolution operations the input is added to the output in order to produce the residual learning function $H(x) = F(x) + x$. On the right is the building block of Dense Net [146]. The layer l gets as an input the output of all layers $[l-4, l-1]$.

in order to alleviate the covariate shift affect [331]. The traditional method of activation normalization is the Local Response Normalization [159, 184]. The most established work though is the later batch normalization technique [153]. In this work the output of each layer is rescaled and centered according to the batch-statistics of activations. The success of this method gave rise to more research in this direction like [12, 382, 311, 418, 394, 150]. For a detailed overview and comparison of these methods the reader is referred to [287, 418, 394].

2.2.1.1.3 Network structure. In an attempt to increase their performance, a large group of works have also explored different architectures of the internal structure of CNNs. After the work of Krizhevsky et al. [184], researchers tried to understand how different parameters effected the quality of the networks. Here we will give a small overview of the main milestone works since then.

One of the first important works was the one of Simonyan and Zisserman [338] who proposed the VGG nets. In their work they showed that with small convolutional kernels (3x3), deeper networks could be trained. They introduced 11, 13, 16 and 19 weighted layered networks. One main constraint on the possible depth of neural networks is the vanishing gradients problem. In an attempt to alleviate this issue, HighWay networks [349] and residual networks (ResNet) [127] make use of "skip" or "shortcut" connections in order to pass information from one layer to one or several layers ahead (Figure 2.2). Huang et al. [146] generalized this idea even further, with

their DenseNet, by giving as input to the l-th layer all previous $l-1$ layers. The building blocks of ResNet, Res Block, and DenseNet, Dense Block, are shown in Figure 2.2.

Besides skip connections, which helped deeper networks to be trained, different methods to increase the quality of networks have also been studied. Lin et al. [216] proposed the Network in Network (NiN) architecture. In their work they substituted the linear convolutional nodes with small Multi-Layer Perceptrons (MLP), giving to the network the ability to learn non linear mappings in a layer. Lee et al. [203] proposed the Deeply Supervised Nets (DSN) which use secondary supervision signals directly to hidden layers of the network. Liu et al. [222] explore a different approach, where the final decision, either classification or any other task, is made not only by the information in the last layer but also from deeper layers. They do so with their Convolutional Fusion Network (CFN), in which Locally Connected (LC) layers are used to fuse lower level information from deeper layers with the high level information of the top layer and make a more informative decision.

2.2.1.2 Recurrent neural networks (RNN)

Recurrent neural networks are a special class of artificial neural networks. A basic RNN module is composed by a feed forward node computing a "hidden state", a recurrent connection, which connects the hidden unit to the next time step input, and an output unit, as seen in Figure 2.3 . This recurrent connection gives the network the ability to make predictions not only according to the current input but also historic inputs that comprise a sequence of data.

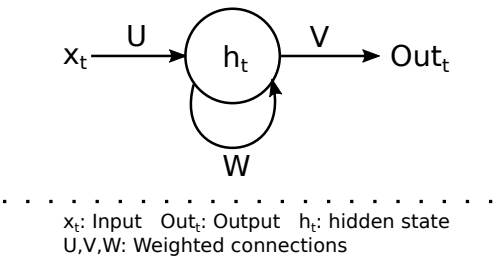


Figure 2.3: Basic module of an RNN processing time step t.

Although this architecture was successful, in problems with a large number of time steps it could no longer maintain high performance. That happens due to the vanishing gradient problem in back propagation through time (BPTT), a most widely used training procedure of RNN. In order to counter these limitations a new architecture, the long short-term memory node (LSTM), was proposed by Hochreiter and Schmidhuber [141]. It contains several gates that control the flow of information and allow the network to store long term information, if needed. Such an architecture has been used for many tasks that deal with sequential data, such as language modeling [439]

and translation [228], action classification in videos [70], speech synthesis [84] and more.

Inspired from the success of the LSTM method, researchers proposed many variations. Some are generic, and can be applied to any problem that simple LSTM is applied while others are application specific.

To the best of our knowledge, the first generic extension of LSTM was proposed in the work of Gers et al. [99]. They noticed that none of the gates have direct connections to the memory cell they are supposed to control. In order to alleviate that limitation they proposed "peephole" connections from the memory cell to the input of each gate. Cho et al. [51] proposed an extension, the Gated Recurrent Unit (GRU), that simplified the architecture and reduced the number of trainable parameters by combining the forget and input gates. Laurent et al. [198] and Cooijmans et al. [56] proposed batch normalized LSTM. Although [198] batch normalized only the input of the node, Cooijmans et al. [56] did so also in the hidden unit. Zhao et al. [446] proposed a combination of several of the above extensions. Specifically, they proposed a bidirectional [319] GRU unit, combined with batch normalization. They fed the network with human joints and action labels for 3D video action recognition. For a more thorough review regarding LSTM and its variants, the reader is referred to [106].

As mentioned above, some extensions of the LSTM are application specific. For example, Shahroudy et al. [324] proposed the Part-Aware LSTM (PA-LSTM), an architecture tailored for skeleton based data. Instead of having one memory cell for the whole skeleton, as is a common approach, they introduced one memory cell per joint of the skeleton, each with its own input, forget and output gates. Liu et al. [219] proposed the spatio-temporal LSTM unit with trust gates (ST-LSTM) for 3D human action recognition. This unit extends the recurrent learning with memory to the spatial domain as well.

2.2.1.3 Restricted Boltzmann machine (RBM)

The Restricted Boltzmann Machine (RBM) was first introduced by Hinton [140] in 1986. It is a two-layer, undirected, bipartite and undirected model (Figure 2.4). It comprises of a set of visible units, which are either binary or real valued, and a set of binary hidden nodes. A configuration with visible vector **v** and hidden vector **h** is assigned an energy given by:

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i \in visible} \alpha_i v_i - \sum_{j \in hidden} b_j h_j - \sum_{ij} v_i h_i w_{ij}, \tag{2.1}$$

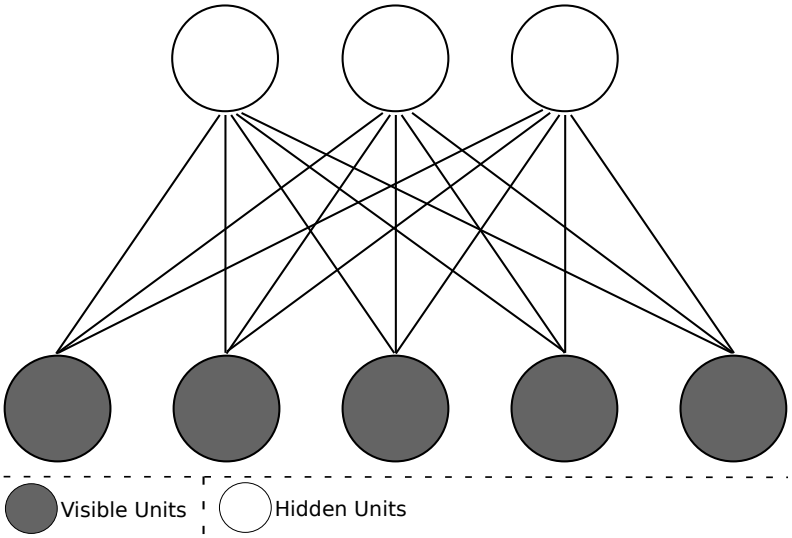


Figure 2.4: RBM architecture. Notice that the connections are undirected.

where α_i, b_j, w_{ij} are the network parameters. Given this energy the network assigns to every pair ($\mathbf{v}$, $\mathbf{h}$) a probability:

$$P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v},\mathbf{h})} \tag{2.2}$$

where Z is the partition function and is given by summing over all possible pairs of visible and hidden vectors. Since there are no direct connections between the hidden or visible units, we can easily obtain an unbiased pair ($\mathbf{v}$, $\mathbf{h}$). Given the visible vector $\mathbf{v}$ the hidden unit h_j is assigned to one with probability:

$$P(h_j = 1|\mathbf{v}) = \sigma(b_j + \sum_i v_i w_{ij}), \tag{2.3}$$

where $\sigma(\cdot)$ is the logistic sigmoid function. Similarly given a hidden vector $\mathbf{h}$ the probability of a visible unit v_i to be assigned to one is given by:

$$P(v_i = 1|\mathbf{h}) = \sigma(\alpha_i + \sum_j h_j w_{ij}), \tag{2.4}$$

Starting from the training data, the network parameters are tuned in order to maximize the likelihood of the visible and hidden vectors pair $(\mathbf{v}, \mathbf{h})$.

RBMs are only two layer deep models and thus are restricted in the complexity of the data they can represent. In order to alleviate this issue a number of deeper models

built on RBMs are designed. The most well known models derived from RBMs are the Deep Belief Networks (DBN) [138], Deep Boltzmann Machines (DBM) [310] and the Deep Energy Models (DEM) [255]. They are all multi-layer probabilistic models that perform non-linear transformation to the data.

DBNs are trained in a greedy layer wise manner, where each layer is trained as an RBM. After the training is done the weights are fixed and the next layer is trained. The final model keeps only the top-down connections of the layers except the top two that remain undirected. Finally, in order to avoid the poor local minimum of the greedy learning, the weights are fine tuned with the up-down algorithm. In case of classification the labels are given to the last layer as binary input. Unlike DBNs, DBMs have undirected weights in all layers. Initially the weights are also trained in a greedy fashion, like a DBN. Since it is very computationally expensive to estimate and maximize the likelihood directly, Salakhutdinov and Larochelle [310] proposed an approximative algorithm which maximizes the lower bound of the log-likelihood [308, 309]. Finally, DEM, the most recent deep model based on RBMs, is a fully connected feed forward network with an RBM on top [255]. The non-stochastic nature of the hidden layers renders it possible to have an efficient training of the whole model simultaneously. For a more comprehensive review of these models the reader is referred to [110].

2.2.1.4 Auto-encoders (AE)

Auto-encoders are a collection of neural network methods based on unsupervised learning. They were first introduced by Bourlard and Kamp [31] in 1988, as auto-association networks. The main idea is to reduce the dimensionality of the data with a fully connected layer and then try to recover the input from the reduced representation. In the case where the network is able to reconstruct the input, the intermediate low-dimensional representation should contain most of the information of the original data (Figure 2.5). Since a single layer network is able to perform only linear transformations, it is not sufficient for performing high dimensionality reduction of complicated data. Thus Hinton and Salakhutdinov [139] proposed a multiple layer version, called auto-encoder (AE). It utilizes several layers to transform or "encode" the data. In some cases, if there is large error in the first layers, these models only learn the average of the training data. In order to alleviate this issue, [139] proposed to pre-train the network so the initial parameters are already close to a good solution. Since then many variants of AEs have been proposed.

One of the first variations of AEs is the Sparse auto-encoder. The basic idea behind it is to transform the data on an over-complete representation of higher dimensionality than the original. The benefits of such a transformation is that (i) there is a high

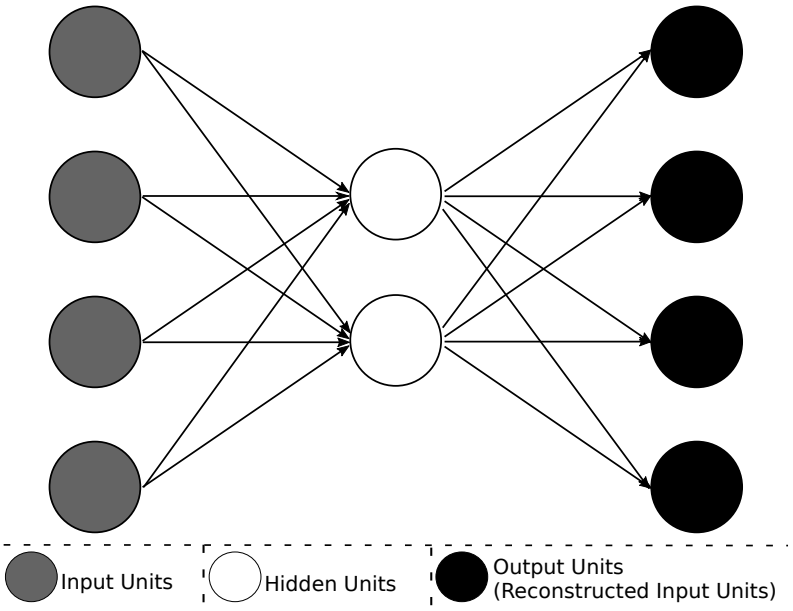


Figure 2.5: Auto-association network. Notice that the output units are reconstructed input units.

probability that in the new representation the data will be linearly separable and (ii) it can provide a simple interpretation of the input data in terms of a small number of parts by extracting the structure hidden in the data [276].

Vincent et al. [386, 387] suggested that a good transformation should provide similar representation for two similar data points. In an effort to force the model to be more robust in small variations of the data they proposed the Denoising AE (DAE), which tried to reconstruct the original data given slightly modified data as input. Rifai et al. [291] proposed a different method to achieve robustness to small input variations, the Contractive AE. They do so by penalizing the sensitivity of encoded representation with respect to the input data point.

Masci et al. [236], inspired by the success of CNNs, proposed a combination of AE with CNNs the Convolutional AE (CAE), and applied it on image datasets, MNITST and CIFAR10. The architecture comprises of several stacked convolutional layers. The model is used as a pre-train mechanism for a CNN which is then trained in a supervised manner for object classification.

2.2.2 Deep learning for high dimensional data

In this section we describe the main deep learning approaches applied on high dimensional data and provide a categorization of them. Specifically, we cluster the methods according to the type of generalization performed.

Most of the deep learning methods applied on higher than two dimensional data are generalized from lower dimensional counterparts, e.g. CNNs, CAEs, etc.. The methods can be divided into two categories, namely increase of physical dimensions and increase of modalities. There are also several models that are developed for high dimensional data and were not generalized from lower dimensions, such as the PointNet [279]. It is important to note that all of the deep learning methods developed for 2D (images) and the generalization to 3D as well are either CNNs or a variation of them, like CAE.

2.2.2.1 Increase of physical dimensions

In this section we describe the methods that were based on generalizing an existing approach to higher dimensions. Although this seems straightforward, due to the curse of dimensionality, as well as the large demand of memory and computational power of deep learning approaches, the extension from two to three dimensional data is not trivial. When considering the static world, i.e., time is not involved in some way, two main concepts exist: the straightforward extension to three dimensional kernels and the projection of data to fewer dimensions coupled with the use of an assembly of lower dimensional models, usually pre-trained on a large dataset, like the ImageNet 2012 [302].

The first approach to extend the 2D convolutional deep learning techniques to the 3D case is the work of Chang et al. [45] on ShapeNets. They implemented a convolutional DBN with three dimensional kernels with which they learned a 3D shape representation from CAD models. The three dimensional convolutional kernels (and pooling) have also been combined with other models, such as the feed forward CNNs [323], CAEs [35] and GANs [417]. Moreover, they have been utilized in many fields such as 3D medical Images [69], Computational Fluid Dynamics (CFD) Simulations [371], 3D objects [240] and Videos [160]. The main drawback of these approaches is the high computational and memory demand of the resulting models, which limit both their size and the input resolution they can support. Although this is the case they are able to exploit relationships in all three dimensions, unlike the 2D methods.

The second cluster is the reduction of the data dimensionality to two, in order to be able to construct complicated models as well as take advantage of pre-trained ones. The reduction from three to two dimensions depends on the type of data in question. For example, when CAD models or 3D objects are concerned, the projection

to two dimensions is done from an outside perspective, i.e. “taking photos” of the object from different angles [353]. Shi et al. [327] proposed an alternative representation of the 3D models. Specifically they proposed a projection of the 3D shape on a cylinder around the object. The height of the cylinder is equal to the height of the object, making their representation invariant to scaling. Three dimensional medical images contain information in three dimensional space, in which case the outside perspective misses all information relevant to most applications. Thus, the data are not projected but rather processed in a slice-by-slice manner [69]. In the case of videos three strategies for lowering the dimensionality have been proposed. In the first one each frame can be considered separately [70, 380]. The second considers frames as extra channels [87, 337, 169, 403]. This is usually done when passing to the networks the optical flow for several frames. Another approach is to try and compress the information of several frames into one. The work of Bilen et al. [23] is in that direction. They propose the Dynamic Image. More specifically, they adapt the method of Fernando et al. [88] that combines features from multiple frames to the pixel level. The result is an image which contains movement information, similar to a blurred one.

Due to the lower dimensionality of the transformed input data, it is possible to construct very complicated and large models. Moreover, a common approach is to use and fine-tune pre-trained models on very large and diverse datasets such as the ImageNet 2012 [302]. Although this is the case, as mentioned in the previous section, these methods lose the ability to explore the correlations in the data in all available dimensions.

2.2.2.2 Increase of modalities

The second type of generalization refers to the increase of the available modalities of the data. To be more precise, although the physical dimensions of the data remain the same, for example from 2D image to 2D image or 2D+time to 2D+time, the information given per point increases. Some examples are the RGB-D data, optical flow added to the videos and more. Depending on the nature of the extra information, the resulting representation might result in a partial space-dimensionality increase. For example, the RGB-D data do not increase the dimensions to three. Nonetheless, the extra information is the distance to the sensor, which provides some information about the extra third physical dimension.

When dealing with this type of dimensionality increase, researchers proposed various strategies to incorporate the extra information. In this work we identified four different strategies.

The most simple and naive approach is to consider the extra information as an

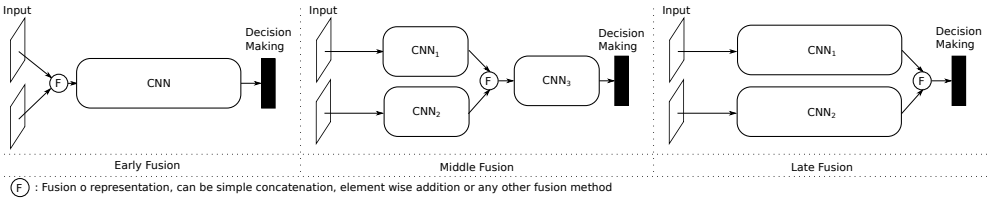


Figure 2.6: Three naive approaches of fusing information from different modalities. Left shows the early fusion, which fuses before any processing. Middle shows mid-fusion and on the right is the late fusion approach.

extra channel and process with the same data dimensionality as before. This is very common when dealing with RGB-D data [58, 391].

In the second category belong approaches that process the different types of information separately and fuse the extracted features by concatenating the feature maps [224, 117]. The extreme case that the fusion happens before any processing layers is the aforementioned first category. Some methods fuse the representations in a mid-stage [371, 43, 169] and some in a late stage [337, 87, 403], as shown in Figure 2.6.

In the third category belong methods that do not apply a naive fusion of the different representations, such as concatenation. Many works propose more sophisticated strategies for fusing the different modalities. For example, Wang et al. [401] try to specifically learn modality specific and common features during training. As a result the total complexity of the model reduces. Moreover, one modality might be missing some of the common features due to noise, such as occlusion, clutter or illumination. In such a case the quality of the representation will not drop since the other modality will provide the necessary information. The sharing of the features happen between the convolutional and deconvolutional layers with fully connected layers. In order to force the networks to learn common and modality unique features, the similarity and dissimilarity of the features, respectively, is added to the global loss function. The similarity and dissimilarity of the features is measured using multiple kernel maximum mean discrepancy (MK-MMD) [401]. Another example is the work of Hazirbas et al. [125], where they make the assumption that one of the modalities is the main source of information and the rest are complementary. They assign one CNN to each modality and then, at several levels of the CNN's hierarchy they insert information from the complementary branches to the main one. Deng et al. [64] followed a different approach. Instead of having two streams, they introduced a third stream, the interaction stream, which is comprised be their newly defined GFU unit. By using this interaction stream, the feature maps of all streams are updated at the interaction points. Park

et al. [271] proposed the multi-modal feature fusion module in order to combine information from different modality-specific branches. Valada et al. [383] proposed a fusion module (SSMA) that emphasizes areas and modality specific feature maps according to the feature map contents, and thus leveraging common and modality specific features.

Finally, some researchers defined data specific solutions. For example, the work of Georgiou et al. [371] evaluates three different modality-processing strategies specific for CFD simulation output, which consist of 4 different modalities over 6 channels of information. Gupta et al. [117] propose a data transformation for the depth channel in RGB-D data, called HHA. Mainly, they introduced two more channels. Although the values of those channels are computed from the depth map itself, they are transformations that are easily learnable, by convolutional kernels, namely height from ground and surface angle to gravity vector.

The benefits of using this transformation are two-fold. First, the network gets more relative information to its input, and second with the depth information transformed to a three channel representation it is possible to use pre-trained networks on ImageNet for this modality as well. Eitel et al. [78] proposed three more encodings that transfer the depth data to a three channel representation and compared them to each other and HHA. Their intuition was that since in object classification, all objects have similar elevation not all channels of HHA are interesting. The projections they proposed are (i) copy the depth values to all channels, (ii) transform to the surface normal vector field and (iii) apply jet colormap of depth values to RGB, ranging from red (near), through green to blue (far). They argue that since the networks are pre-trained on RGB data, transforming depth to RGB might result in a more stable fine tuning of the networks. The last method showed the best results on object classification. Nonetheless, they do not perform a comparison in the case where the elevation makes a difference and thus there is no objective comparison between their method and HHA. For a visual comparison of the four different schemes, the reader is referred to [78].

2.3 Traditional methods

Traditional methods vary a lot depending on the application and the type of data they are applied on. For example, when dealing with semantic segmentation the most common, non-deep, approach is to apply a graph model like a Conditional Random Field (CRF) [334, 174, 65, 352]. On the other hand a large group of works utilize template matching approaches [135, 137, 113, 292] in order to tackle object detection. Although there is a large diversity on the applied methods, there are some

common practices between most of them. The data are not processed in their raw format, but they are transfered in a feature space in which they are represented and then processed by any machine learning pipeline.

Building from the very successful work of feature representation of images in many applications of computer vision, a lot of methods are developed that generalize them to be applicable to higher dimensional data as well. The main idea is to describe the content of an image using a number of points or neighborhoods instead of the whole image. The type of description can vary, from raw values to histograms of gradients and point wise comparisons. In order to get a good content description and not background description, researchers develop specialized detectors which detect points according to several characteristics. This very well known pipeline is extended and applied to higher dimensional data.

The most common types of higher dimensional data that people are dealing with are objects represented by surfaces and/or color, volumetric representation of the world, videos or sequences of images, or in the extreme scenario four dimensional data, a three dimensional representation evolving in time. A large group of works try to generalize the interest point detectors and descriptors of images to the data available. Because of the different nature of different data types the definition and development of features change accordingly. The main categories of such features are surface features, volumetric features and spatio-temporal features.

2.3.1 Object surface features

Many people have tried to derive heuristics and encodings of 3D shapes and objects that help to process them in an efficient way. The first approaches date back to 1984 with the work of B. Horn, Extended Gaussian Images [144]. Since then numerous approaches and features have been developed. The main common objective is to have a low dimensional yet discriminative description of three dimensional objects and shapes. There are many ways one can separate these methods according to their characteristics. A common distinction is global and local features. Global features describe the whole object whilst local ones describe a small neighborhood around a point on the object. The final description of the object is comprised by a collection of such local descriptions.

2.3.1.1 Global features

Global features usually try to aggregate low-level structural and geometric statistics of the complete objects like point pair distances, surface normals and curvature. Their advantage is the very low dimensional representation they offer in comparison with local descriptors that make object retrieval much faster. Unfortunately they require

the whole object to be available and fully separated from the environment [114]. Thus they are very limited in real world scenarios where objects are partially occluded and usually blended in their environment. Some examples of global methods are the Extended Gaussian Images [144], shape distributions [266], the light field descriptor (LFD) [46], the spatial structure circular descriptor (SSCD) [93] and the elevation descriptor (ED) [329]. For a more comprehensive review of global features the reader is referred to [93, 329, 114].

2.3.1.2 Local features

Local features describe some properties of the local neighborhood of an object's surface points. In order to describe a complete object, a set of these local descriptors have to be used. Depending on the needs of an application a different scheme of accumulating these local features is used. For example, for object recognition the local features of an object in the repository are added to a feature library. These features are searched for candidate correspondences with the features of a scene, which vote for specific objects and poses [112]. Bronstein et al. [37] incorporated the well established "Bag of Features" model of computer vision to 3D shape retrieval, in which the local features are translated to "visual words", or in this case "shape words", in order to obtain a global compact description of the full object. When tackling the scene semantic segmentation task, these features are considered as the data primitives in order to construct geometric unary potentials that are considered in an CRF pipeline [334, 335].

As mentioned above, local descriptors encode information of a neighborhood around a point. In order to exclude points that do not carry enough information, feature detectors are introduced. These detectors usually find points whose neighborhoods exhibit large variance of some property, e.g. fast and multiple changes of the surface normals. Given a detector, a set of "highly informative" points is detected. Then, one can extract local descriptors only for those points and describe an object or scene only using these points neighborhoods. Since most real world applications deal with varying scales of objects, as well as a variety of occlusions and deformations, feature detectors and descriptors must be invariant to scaling, rigid and non-rigid deformations as well as illumination changes. Moreover, they need to be repeatable, and unique. A very comprehensive study on surface detectors and descriptors has been published in [112]. In this work we will give a brief overview of the available detectors and descriptors.

2.3.1.2.1 Detectors. Interest point, salient or *keypoint* detectors are a classic first step to object description, since they define which points of the surface are the most

important for describing the object. A generic and popular division of detectors depends on whether they are scale invariant or not [378, 112]. Although scale invariance is an important feature, not all detectors have that ability. Some of them take the scale or neighborhood size, in which they will detect keypoints, as an input. Consequently, detectors are classified as fixed-scale or adaptive-scale keypoint detectors.

Most fixed-scale keypoint detectors have two common steps [378]. They first compute a quality measurement across all points. Then, the points are checked for saliency by checking whether they are local maxima of the quality measurement. As an example we describe the detector defined by Mokhtarian et al. [249]. A point is declared as interest point if its curvature is larger than the curvature of every 1-ring neighbor, where the k-ring neighbors are defined as the neighbors that have k edges distance. On the other hand, adaptive-scale detectors, inspired by the works of image detectors, first construct a scale-space and then search for local maxima of a defined function along the scale-space [378]. For example, Zaharescu et al. [438] build a scale-space by applying Gaussian filters directly on the 3D mesh and detect points as the extrema of the DoG space. For an extensive review of keypoint detectors the reader is referred to [378, 112].

2.3.1.2.2 Descriptors. Local surface descriptors can be subdivided according to different factors. For example, they can be subdivided according to the invariance properties, i.e., invariant to rigid or non-rigid transformations, invariant to scaling etc. The most common division for surface features is according to their encoding, i.e., histograms, point signatures and transformations [112, 376], which we will follow in this work as well.

Histograms are a broadly used type of feature description, not only in describing 3D surface features but also in image and video analysis. Histograms accumulate different measurements of the neighborhood of a point and use that as a feature. Histograms have been very popular due to their simplicity combined with high descriptive capabilities. Three dimensional surface histogram descriptors can be subdivided to spatial distribution histograms (SDH), geometric attribute histograms (GAH) and oriented gradient histograms (OGH) [112].

SDH accumulate in histograms the spatial relationship, e.g. pair point distances, of points in a neighborhood. One of the first examples of SDH descriptors is the spin images (SI) [166, 164]. The spin image is a two dimensional histogram. First, all the neighboring points are transfered to a cylindrical coordinate system starting from the interest point. The points are expressed with the radial distance α and the elevation distance β. The 2D histogram accumulates the number of points existing in a square of the $\alpha - \beta$ plane. Other examples include the extensions of the SI, Scale Invariant SI

(SISI) [63] and Tri-SI [114, 109], the generalization of shape context (SC) [22], 3DSC [91], and the Rotational Projection Statistics (RoPS) [113]. More recent examples are the TOLDI [426], RSM [283], BRoPH [450] and the MVD [107].

GAH accumulate geometric properties of the neighborhood of a point, e.g. angle between surface normals. Some examples are the Local Surface Patch (LSP) [47], the THRIFT [90], the point feature histogram (PFH) [305], its fast counterpart the fast point feature histogram (FPFH) [304] and the Signature of Histograms of Orientation (SHOT) [376].

OGH accumulate gradients of various metrics of the surface. This kind of descriptors are closely related and inspired from image descriptors like SURF [19] and SIFT [225, 226]. Some examples are the 2.5D SIFT [223], the meshSIFT [231], the meshHOG [438], 3DLBP [238], 3DBRIEF [238] and 3DORB [238].

Yang et al. [425] proposed a descriptor (LFSH) which combines SDH and GAH. Specificaly, they use histograms of a depth map, point distribution and deviation angle between normals.

Signatures describe the local neighborhood of a point by encoding one or more geometric measures computed individually at each point of a subset of the neighborhood [376, 112]. Some examples of signature descriptors are the Exponential Map [259] and the Binary Robust Appearance and Normal Descriptor (BRAND) [67], a binary descriptor that encodes geometrical and intensity information from a local patch. This is achieved by fusing intensity variations with surface normal displacement.

Transforms. These descriptors perform a transformation of the surface to a different domain and describe the neighborhood according to the characteristics of the surface on that domain. For example, Rustamov [303] performed a Laplace-Beltrami transform whilst Knopp et al. [178] performed a Hough transform on a voxelized representation of the surface. Other examples of transform descriptors are the Heat Kernel Signature (HKS) [355], its scale invariant variation (SI-HKS) [38] as well as the more recent Wave Kernel Signature (WKS) [11].

A collection of the most important, according to this study, surface features is shown in Table 2.1. The features are shown together with what, in our opinion, is their most important contribution to the field.

2.3.1.2.3 Rotation Invariance. A common goal for most descriptors is to achieve rotational invariance. In order to achieve that they try to find a repeatable and unique Reference Angle (RA) or local Reference Frame (LRF) to which the local patch or neighborhood is rotated before they describe it [164]. The first approaches used the surface normal as a reference vector in order to achieve rotation invariance. Although

Method	Year	Comments
SI [166, 164]	1998	Most sited surface descriptor
PFH [305]	2008	captures multiple characteristics
FPFH [304]	2009	improved computational efficiency of PFH
2.5D SIFT [223]	2009	SIFT for depth images
HKS [355]	2009	invariant to non-rigid transformations
mesh-HOG [438]	2009	extension of HOG [62] descriptor for triangular meshes
3D-SURF [178]	2010	extension of SURF [19] descriptor for triangular meshes
SI-HKS [38]	2010	scale invariant extension of HKS
SHOT [376]	2010	signatures of histograms, balance between descriptiveness and robustness
CSHOT [377]	2011	extension of SHOT descriptor to incorporate texture information
WKS [11]	2011	invariant to non-rigid transformations, scale invariant, outperforms HKS
TriSI [109]	2013	rotation, scale invariant and robust extension of SI descriptor
RoPS [113]	2013	unique and repeatable LRF, robust to noise and mesh resolution
3DLBP [238]	2015	Generalization of LBP to 3 dimensions
3DBRIEF [238]	2015	Generalization of BRIEF to 3 dimensions
3DORB [238]	2015	Generalization of ORB to 3 dimensions
LFSH [425]	2016	combines depth map, point distribution and deviation angle between normals.
TOLDI [426]	2017	robust to noise, resolution, clutter and occlusion LRF. Multi-view depth map descriptor
RSM [283]	2018	uses multi-view silhouette instead of depth map. Outperforms RoPS
BRoPH [450]	2018	binary descriptor, combines depth map and spatial distribution.
MVD [107]	2019	Extremely low dimensional. Performs similar to SotA descriptors in Object Recognition.

Table 2.1: A collection of surface descriptors with the most influence on the field, according to our study. The table shows the most important contribution of the work to the field. For a more comprehensive study of surface descriptors the reader is referred to [112].

the surface normal is easy and fast to compute, it is very sensitive to noise. Other methods use the Singular Value Decomposition (SVD) or Eigenvalue Decomposition (EVD) [259, 448, 34]. Unfortunately these methods do not produce a unique LRF and in order to tackle that multiple descriptors are extracted per point. A good overview and comparison of these methods is given in [376]. Moreover, they propose their own method which is more robust to noise and tackles the limitations mentioned above. To do that it computes the EVD of a weighted N-nearest neighbor covariance matrix, in combination with the sign swapping of [34].

2.3.2 Volume features

In some applications, the data of interest are not represented by surfaces, but by volumes. Some examples include voxelized representation of the objects, as well as 3D images, mainly medical images, like 3D ultrasound, CT scans and MRI scans [50, 256]. In some cases, videos are considered as three dimensional data where the time dimension is considered equivalent to the two spatial ones [321]. In order to describe the content of these kind of data, scientists generalized one of the known Interest Point detector and descriptor of 2D images to 3D, namely Lowe's SIFT detector and descriptor [225, 226].

Scovanner et al. [321] were one of the first that tried to generalize the SIFT descriptor to the three dimensional case. Although they did extend the SIFT descriptor they did not generalize the detector as well. The method picks random points in the volume as salient points and then describes them in a similar fashion to the SIFT. Orientation invariance is achieved by computing the dominant solid angle of the gradient and rotating the neighborhood around the point so that the solid angle is equal to zero. Finally, the neighborhood is split into eight sub-regions and a gradient orientation histogram is computed per region. The final descriptor is the concatenation of these histograms, which results in a 2048-D vector. They tested their descriptor on action recognition and showed that their method performs better than the regular 2D-SIFT.

At the same time, Cheung and Hamarneh [50] developed independently their own generalization. In contrast to Scovanner et al.'s work [321], they generalized both the descriptor and the detector. Moreover, instead of generalizing to the 3D case, they generalized to the nD case making their method applicable to many more datasets and applications. They use $n-1$ directions, with β bins for each, resulting in β^{n-1} bins in total. The gradients are computed using hyperspherical coordinates. They tested their method on 3D MRI of the brain and 4D CT scans of a beating heart.

Allaire et al. [9] focused on the 3D case. They observed that the aforementioned methods failed to account for the tilt that a neighborhood can have, resulting in the

Method	Data Type	Comments
Scovanner et al. [321]	Video	first 3D SIFT
Cheung and Hamarneh [50]	3D MRI & 4D CT	detector & nD
Allaire et al. [9]	3D CT, MRI, CBCT	detector & account for tilt
Ni et al. [256]	3D Ultrasound	US noise filter and smoothing

Table 2.2: Extensions of the SIFT descriptor to 3D volumetric data.

need for an extra angle in order to have full orientation invariance. For detecting points they extended Lowe's method by computing the Difference of Gaussians (DoG) in a manner similar to Lowe. The local minima/maxima of the DoG in the scale-space are picked as interest points. After detection in the scale-space, feature points are filtered and localized. The remaining points are described as follows. First, they find the dominant solid angle and for each angle with magnitude above 80% of the maximum they calculate the tilt. As with the solid angle, every angle that has a magnitude more than 80% of the maximum is considered as a different interest point. They evaluated their method on 3D registration and segmentation of clinical datasets such as CT, MR and CBCT images.

Ni et al. [256] used a similar method to the one developed by Allaire et al. [9] and adapted it for optimal description of ultrasound content, which is very noisy. They used the same filtering techniques at the detection stage with different thresholds, necessary due to the increased noise of ultrasound images. Besides the extension of Lowe's detector, they also applied the Rohr3D detector developed by [296]. It first defines the cornerness as the determinant of the matrix C, given by equation 2.5.

$$C = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix} \tag{2.5}$$

where I_{ij} are the second order intensity gradients of a voxel. The local maxima of the cornerness are then detected as interest points. For description they do not use all three angles defined by [9] but only the two constituting the solid angle, like in [321]. They evaluate their method on 3D ultrasound (US) registration and compare it to the original 3D SIFT of Scovanner et al. [321].

An overview of the aforementioned methods, together with the milestone of each work, is given in Table 2.2.

2.3.3 Spatio-temporal features

As with images and three dimensional representation of objects, traditional approaches that deal with videos follow the same regime. First, a number of points is defined as interest points. These points are either detected through some saliency measurement, which means that their neighborhood is considered as very informative, or they are densely sampled, as in [171]. These points are then used to describe the whole sequence of frames (either 2D or 3D). There are many methods that try to detect and describe these kind of interest points.

First traditional approaches dealing with time dependent data, like video, either used a collection of 2D features, i.e., image features, to describe the clip or consider time as an extra dimension equivalent to the spatial ones and thus represent the clip as a 3D volume. As such, simple extensions of the image features to the 3D case are used to describe the volume [321]. Although this method produced good results at the time, the different nature of the time dimension as well as the large variance in sampling frequencies by different sensors, i.e., frame rate, motivated scientists to develop methods that describe spatio-temporal volumes whilst regarding time separately. These features are called spatio-temporal features. The new interest points are known as space-time Interest Points (STIPs).

2.3.3.1 STIP detectors

The first STIP detector was proposed by Laptev [190]. It's an extension of the Harris corner [123], called Harris3D. The Harris3D operator considers different scales in the space and time dimensions. To achieve that it convolves the video sequence f with a Gaussian kernel g given by equation 2.6.

$$L(\cdot;\sigma_l^2,\tau_l^2) = g(\cdot;\sigma_l^2,\tau_l^2) * f(\cdot) \tag{2.6}$$

where the spatio-temporal Gaussian kernel is given by:

$$g(\cdot;\sigma_l^2,\tau_l^2) = \frac{1}{\sqrt{(2\pi)^3\sigma_l^4\tau_l^2}} \cdot \exp\left(\frac{-(x^2+y^2)}{2\sigma_l^2} - \frac{t^2}{2\tau_l^2}\right) \tag{2.7}$$

where σ_l^2, τ_l^2 are the spatial and temporal variances respectively and x, y are the spatial coordinates whilst t is the temporal one. Given a space and a temporal scale, a corner or interest point is found by finding the local maxima of the corner function given by equation 2.8.

$$H = det(\mu) - k\ trace^3(\mu) \tag{2.8}$$

where μ is the 3 by 3 second-moment matrix weighted by a Gaussian function, given by equation 2.9. In a later work, Laptev and Lindeberg [192] extended the detector

in order to be velocity adaptable, which provides invariance to camera motion. In order to achieve that they considered the transformation caused by camera motion as a Galilean transformation, which is computed iteratively. This approach was later used by [191] for motion recognition. Schuldt et al. [318] combined the feature size adaptation of [190] and the velocity adaptation [192] in a single framework.

$$\mu = g(\cdot; \sigma_i^2, \tau_i^2) * \begin{bmatrix} L_x^2 & L_x L_y & L_x L_z \\ L_x L_y & L_y^2 & L_y L_z \\ L_x L_z & L_y L_z & L_z^2 \end{bmatrix} \tag{2.9}$$

Another very popular spatio-temporal detector is the one developed by Dollár et al. [68], known as cuboids. The motivation behind their detector lies in the observations that i) corners are very sparse in images and even sparser in videos and ii) there are movements, like opening and closing of a jaw, that do not include corners and thus if only corners are chosen to represent a video clip, many actions will not be recognizable. STIP are detected at the local maxima of the response function given in equation 2.10.

$$R = (I * g * h_{ev})^2 + (I * g * h_{od})^2 \tag{2.10}$$

where $g(x, y; \sigma)$ is a 2D Gaussian smoothing function applied only on the spatial dimensions and h_{ev} and h_{od} are a quadrature pair of 1D Gabor filters, given by equations 2.11, applied temporally. The scale of the feature in the spatial dimensions is defined by the Gaussian (σ) whilst in the temporal dimension by the quadrature pair ($\tau, \omega = \frac{4}{\tau}$).

$$\begin{aligned} h_{ev}(t; \tau, \omega) &= -\cos(2\pi t\omega) e^{-\frac{t^2}{\tau^2}} \\ h_{od}(t; \tau, \omega) &= -\sin(2\pi t\omega) e^{-\frac{t^2}{\tau^2}} \end{aligned} \tag{2.11}$$

Bregonzio et al. [33] observed that the aforementioned detector has some drawbacks. The Gabor filters applied in the temporal dimension are very sensitive to noise and produce many false detections in textured scenes. Moreover, it fails to recognize slow movements. In order to deal with these drawbacks they propose their own STIP detector which works in two steps. The first step is simple differencing between consecutive frames in order to produce regions of interest in which there is motion. The second step is to apply, spatially, a 2D Gabor filter.

Oikonomopoulos et al. [261] followed a different approach. They extended to the spatio-temporal case the approach of Kadir and Brady [167]. They first defined a measure of saliency based on the amount of information change in a neighborhood, which they expressed by the entropy of the signal in the neighborhood. The extension

to the spatio-temporal case is done by considering a cylindrical neighborhood instead of a two dimensional circle.

Wong and Cipolla [415] argued that all the above methods detect interest points using only local information, which produces a lot of false positives in the presence of noise. In order to counter this drawback they proposed an alternative approach which uses global information in order to detect interest points in a video sequence. In order to do so they applied non-negative decomposition of the sequence, which is represented by a two dimensional matrix, in which each column is a frame of the video. The result of the decomposition is a number of subspaces ϕ and transitions χ. By applying Difference of Gaussians (DoG) on the subspaces and the transitions, they detect spatio-temporal interest points. They compared their method with the aforementioned approaches on gesture recognition using the same description for all detectors, and showed that their method outperforms the rest.

Inspired by the work of Laptev [190], Willems et al. [413] proposed an new detector which instead of utilizing the second moment matrix μ (given by equation 2.9) they utilized the Hessian matrix H given by equation 2.12. The points are detected at the local maxima of the saliency measurement S given by equation 2.13. Unlike the 2D case [20], maxima of S do not ensure positive eigenvalues of H which means that saddle points will also be detected.

$$H = \begin{bmatrix} L_{xx} & L_{xy} & L_{xz} \\ L_{xy} & L_{yy} & L_{yz} \\ L_{xz} & L_{yz} & L_{zz} \end{bmatrix} \tag{2.12}$$

$$S = |\det(H)| \tag{2.13}$$

Yu et al. [433] developed a generalization of the FAST [298] detector to the spatio-temporal case, which they call V-FAST. For each candidate point they considered three 2D planes, the XY, XT and YT planes. They applied the FAST detector in each plane. If the point is detected as interest point in the spatial domain (XY plane) and at least one of the time comprising planes (XT or YT) then the point is considered as a STIP.

Cao et al. [42] observed that from all STIPs detected by Laptev's [190] detector, only 18% belong to a specific action whilst the rest belong to the background. Inspired by this phenomenon, Chakraborty et al. [44] proposed an new pipeline for STIP detection. They initially detect spatial interest points (SIPs) using the Harris detector [123] and then apply background suppression and other temporal and spatial constraints in order to keep only features relative to the motion in the sequence.

Finally, Li et al. [211] proposed a new detector, the UMAM-detector. The video is transfered to a Clifford algebra based representation. There, a vector is extracted for each pixel which contains both motion and appearance information. In this new space

Method	comments	Year
Harris3D [190]	first STIP detector	2003
Harris3D + velocity adaptation [192, 318]	limit camera motion detections	2004
Cuboids [68]	more dense point detection	2005
Bregonzio et al. [33]	limit false detections & detect slow movements	2009
Oikonomopoulos et al. [261]	information based saliency	2005
Wong et al. [415]	use of local and global information	2007
V-FAST [433]	efficient computation	2010
Chakraborty et al. [44]	limit background detections	2012
Li et al. [211]	unified motion and appearance	2018

Table 2.3: Existing spatio-temporal detectors. The left column shows the name of the descriptor together with the paper that proposes it, the middle column the contribution of the method to the field and the right column the year the method was published.

they apply a Harris corner detector to detect STIPs. According to their experiments the UMAM-detector outperform all the aforementioned detectors and some deep learning methods, in classification performance.

All the above detectors are summarized in Table 2.3, together with their contribution to the field.

2.3.3.2 STIP descriptors

In order for the STIPs to be represented in an optimal form for machine learning pipelines, special descriptors are defined that try to capture important information for the neighborhood of the STIP. Most proposed descriptors can be categorized depending on the type of measurements they contain or the way they quantize that information. More specifically, the most typical measurements taken to describe a STIP are the N-jets [179], Gaussian gradient field (similar to HoG and SIFT [225, 62]) or optical flow field [24]. These measurements are usually quantized or vectorized by histogramming or Principal Component Analysis (PCA) [193, 191].

The N-Jets represent a collection of point derivatives (up to Nth order) at a specific scale of the scale-space representation L, given by equation 2.14.

$$J(g(\cdot;\sigma_0,\tau_0) * f) = \{\sigma L_x, \sigma L_y, \tau L_t, \sigma^2 L_{xx}, ..., \sigma\tau^{N-1} L_{yt..tt}, \tau^N L_{tt..tt}\} \qquad (2.14)$$

The Gaussian first order gradient field is also computed on the scale-space represent-

ation L, in order to make the descriptors invariant to scaling and noise. The optical flow field represents the movement in a clip at each pixel by a velocity vector field. There are a lot of methods that try to efficiently and accurately extract that vector field. For a good overview of the optical flow estimation field the reader is referred to [354].

As mentioned above there are many ways to accumulate information over the spatio-temporal neighborhood. The most common ones are histogramming and applying PCA. Histogramming is either applied globally, i.e., one histogram over the STIP neighborhood, or on several small neighborhoods around the STIP. In the later case the separate histograms are concatenated in order to constitute a single descriptor. PCA is usually applied on a number of IP of a train set in order to obtain D most significant dimensions defined by the eigenvectors.

Laptev et al. [193, 191] tested a number of different descriptors both in terms of measurements accumulated and in the type of accumulation. Their study showed that, on average, local histograms on adaptive scales perform better than the rest of the approaches. Moreover, methods based on the first order gradient field outperform both optical flow and the N-Jets.

In a parallel work, Dollár et al. [68] performed a similar comparison. They tested normalized pixel values, first order intensity gradients and optical flow values. They tried all the above measurements by flattening the cuboid and within global or local histograms. Finally, on all descriptors, they applied PCA to reduce the dimensionality. According to their experiments histogramming did not benefit performance and thus concluded to the flattened values with PCA. As with Laptev et al.'s experiments, the gradient based descriptors showed higher overall performance than the rest.

Niebles et al. [257] extended the aforementioned descriptor. They first smoothen the image at a specific scale and then extract the intensity gradients. The apply this function for several scales and then apply PCA to get the final descriptor. Their method indeed outperforms Dollár et al.'s [68] method but it is still outperformed by Laptev et al.'s [191] histogram of gradients, with velocity adaptation.

Laptev et al. [194] proposed a combined histogram of gradients with a histogram of optical flow. Their descriptor together with the non linear SVMs managed to outperform all previous methods on the KTH dataset [318]. Willems et al. [413] extended the known SURF descriptor [19] to the spatio-temporal case. Their implementation differentiates between the spatial and temporal dimensions by setting a different number of bins, as well as different scales (σ and τ). They evaluated their method on the mouse behavior dataset as well as the KTH and they achieve comparable to the state of the art results.

Klaser et al. [177] designed a new 3D HoG descriptor. They introduced a generalization of the orientation binning of the known SIFT descriptor by introducing a

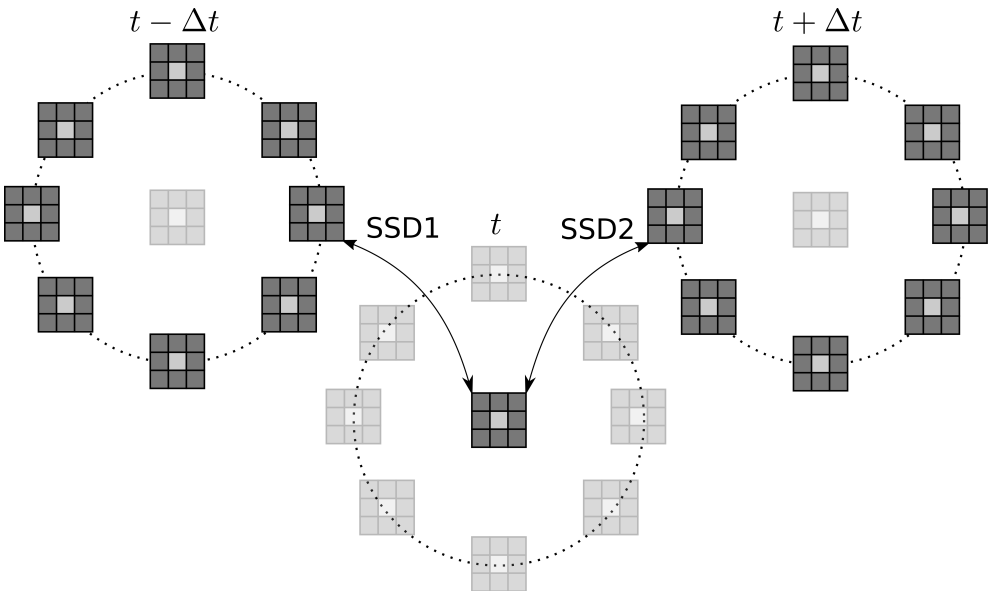


Figure 2.7: An illustration of the encoding process of LTP. For each of 8 different locations at time $t - \delta t$ and the same locations at $t + \delta t$ SSD distances of 3x3 patches to a central patch at time t are computed [429].

normal polyhedron, dodecahedron or icosahedron, and considering each face of the polyhedron as a bin. The angle of the gradient vector to the surface normals of the faces is computed and if its smaller than a threshold, the projection of the gradient vector to the surface normal contributes to the respective face's bin. Moreover, they generalized the integral image method of [388] to the integral video method. The integral video is a representation of the video volume that helps the fast computation of average gradients. Given a video volume $\nu(x, y, t)$ and its three first order partial derivatives $\nu_{\partial x}, \nu_{\partial y}, \nu_{\partial t}$, the integral video, $iv(\cdot)$, of direction j is given by:

$$i\nu_j(x, y, t) = \sum_{x'<x, y'<y, t'<t} \nu_{\partial j}(x', y', t') \tag{2.15}$$

A block of video **b** is first divided into SxSxS sub-blocks. For each sub-block the average gradient and its contribution to the histogram bins are calculated. The final descriptor is a concatenation of several such histograms computed on MxMxN blocks around the STIP. Willems et al. [412], inspired by the quantization of Klaser et al. [177] extended the method of [413] to quantize the gradient orientations in the same way.

Yeffet and Wolf [429], inspired by the Local Binary Pattern descriptor [262], pro-

posed the Local Trinary Pattern (LTP) a spatio-temporal motion descriptor. The main idea of the descriptor is to compare patches between frames instead of pixels within an image. Eight patches neighboring the pixel in question in the previous and next frames are defined, as well as a "central" patch which includes the pixel in question, as seen in Figure 2.7. A so-called trit is calculated for each spatial location (i, j) according to the following rule:

$$\begin{aligned} -1 \; if \; SSD1 < SSD2 \\ 0 \; if \; SSD1 = SSD2 \\ +1 \; if \; SSD1 > SSD2 \end{aligned} \tag{2.16}$$

where SSD is the sum of squared differences between the patches (Figure 2.7). A global descriptor is calculated by combining the trinary patters for all available pixels in histograms. First spatial histograms are created by splitting each frame in $(m \times n)$ patches. The resulting histograms are then merged temporally to create one global spatio-temporal descriptor.

2.3.3.3 3D space

Due to the inexpensive available sensors, scientists extended the STIPs to the 3.5 and 4 dimensional cases as well. To the best of our knowledge, the first to define detectors and descriptors for higher than 2 + Time dimensional data are Xia and Aggarwal [420]. Their detector is similar to Dollár et al. [68]'s Cuboids. The motivation behind their method is that due to the nature of depth images, detectors developed for color based STIP detection tend to find many points in the background and thus introducing a lot of noise in the description of a clip. In order to avoid that they introduced a correction function that smoothens out depth map specific type of noise. After the detection of the Depth-STIPs (DSTIPs) the information of the spatio-temporal neighborhood is described by a occupancy histogram.

In later work Oreifej and Liu [265] generalized the Histogram of surface Normals (HON) [361] to four dimensional surfaces (HON4D) and applied it on 3D Action Recognition. Finally, Rahmaniet al. [285] proposed the Histogram of Oriented Principal Component (HOPC). Their descriptor calculates the principal components of the scatter matrix of spatio-temporal points around an interest point and create a histogram of principal components for all points in a neighborhood. In a later work, they also proposed a detector in order to filter out points that are irrelevant [284]. Their method first computes the ratio of sequential eigenvalues. If the surface is symmetric then at least one of these ratios is going to be one. Thus they define a threshold, and if a ratio is below that the point is excluded. Otherwise the neighborhood of that point is considered informative enough to be of interest.

2.3.3.4 Trajectories

Driven by the poor generalization performance of the aforementioned approaches, researchers proposed a new strategy for handling the time dimension [237, 244, 356]. Instead of describing the change in the temporal dimension in a local manner as with the spatial ones, researchers tried to describe motion using trajectories of spatial interest points and their spatial description.

More specifically, Matikainen et al. [237] track features in a video using the standard KLT method [227, 328]. For every tracked feature they keep a vector of frame-by-frame position derivatives. The resulting vector is the trajectory-feature. These features are then clustered and the Bag of Words (BoW) model is implemented. The final action classification happens using an SVM. In parallel work, Messing et al. [244] proposed a very similar feature which they call velocity history. The difference with the aforementioned method is that they quantize the velocities in eight directions and five magnitudes. Moreover, the classification is done by a generative mixture model instead of the BoW approach. Sun et al. [356] proposed a different approach, but in the same direction. Instead of the KLT method, they find trajectories by applying frame-by-frame SIFT feature matching. According to their results, this is a more robust approach for feature tracking. Then, the visual characteristics of each trajectory is described by the average SIFT descriptor tracked. In order to describe the temporal dynamics of the trajectory, a Hidden Markov Chain (HMC) is employed that is trained on the spatial development of features. Finally, the inter-trajectory context is encoded with their proximity descriptor.

Wang et al. [395, 396], inspired by the success of the aforementioned methods as well as the dense sampling of features in images [260], proposed a combination, the dense trajectories. The trajectories are sampled on multiple scales on a spatial grid via dense optical flow. Finally, the area around the trajectories is described by the HOG-HOF spatio-temporal descriptor. Their method achieved state of the art results at the time, on many benchmarks. In later work, Wang and Schmid [397] proposed an improvement on the dense trajectories. They tracked camera movement and used it to reject trajectories caused by it. Moreover, they applied the estimated camera movement as a correction to the optical flow, in order to extract camera motion invariant trajectories.

2.4 Datasets and benchmarks

One of the main motives behind the research on higher than two dimensional data is the large availability of datasets comprised by such representations. Depending on the application and the type of data different datasets and benchmarks are proposed,

both small and large scale. In this section we will give an overview of the well known and current benchmarks and large datasets for the domain of computer vision in higher dimensions and we categorize them according to their intended application. To be more precise, numerous small scale datasets and benchmarks exist that are meant for very specific applications. Nonetheless, for each type of data, i.e., 3D scene, action in video, objects etc., there are some large scale datasets that help evaluate the data representation methods that can be applied on many different tasks. These are the datasets that are presented here and are categorized according to the type of data they deal with, namely object understanding, scene understanding and video understanding. More specific concepts can be added, like video retrieval, but due to the small number of datasets, they are grouped together in a category called "other datasets".

2.4.1 Object understanding

There is a large collection of datasets with various 3D models of objects used for object understanding tasks, like detection and classification, shape understanding and more. These datasets either contain 3D images or scans of real objects, e.g. [313, 330] or they might contain designed objects like CAD models [419]. Moreover, different datasets are used for different tasks. For example, the LINEMOD dataset [136] is used for object detection, classification and pose estimation, whilst the Princeton shape benchmark (PSB) [330] focuses on different classification themes. Besides these state of the art datasets, there are also smaller but well known datasets. Some of these are Lai et al.'s [188] dataset, the big bird [339] and the SHREC [206]. For a good overview of all these benchmarks and datasets the reader is referred to [89]. Table 2.4 gives a comparison of the state of the art datasets.

The largest datasets available, to date, are datasets that contain designed models and objects instead of real scans, largely due to the longstanding graphics communities. Some of the well known datasets are the Princeton shape benchmark [330], which consists of 161 object classes and a total of 1814 models. The ModelNet [419], a dataset which consists of 151,128 3D CAD models in 660 categories. ShapeNet [45] is also a recent database, which tries to make even more detailed annotations than just object labels. The raw dataset consists of roughly 3 million models, from which 220,000 have been classified into 3,135 categories. Besides the raw dataset the authors also made two subsets. The first, called shapeNetCore, consists of 51,300 models in 55 common categories, with extra alignment annotations and the second, shapeNetSem, consists of 12,000 models from 270 categories. In addition to manually verified category labels and consistent alignments, they are also annotated with real-world dimensions, estimates of their material composition at the category level,

Dataset	Data Type	#Images	#Objects	#Object Cat	6DoF pose
PSB [330]	PSG	-	1 814/6 670	161/1 271	-
ModelNet [419]	CAD	-	151 128	660	-
ShapeNet [45]	CAD	-	3M/220K	- /3 135	-
shapeNetCore [45]	CAD	-	51 300	55	-
shapeNetSem [45]	CAD	-	12K	270	-
YCB [40]	RGB-D	600	75	-	No
Rutgers APC [289]	RGB-D	10K	24	24	Yes
Redwood [52]	RGB-D	23M	> 10K	44	No

Table 2.4: Large scale datasets and benchmarks for object understanding. PSG stands for polygonal surface geometry. DoF stands for degrees of freedom.

and estimates of their total volume and weight [45, 314].

As mentioned above there are also datasets with scanned real life objects instead of designed models. One example is the YCB object and model set [40]. It consists of every-day object scans from 75 object categories. For each object the dataset includes 600 RGB-D images coupled with 600 high resolution RGB images, segmentation masks as well as calibration information and texture-mapped 3D mesh models. The Rutgers APC RGB-D dataset [289] consists of more than 10 thousand RGB-D images. In total it contains 25 objects along with their 6DoF pose. Choi et al. [52] created a dataset of scanned 3D objects with an RGB-D camera. The dataset provides a variety of different objects, from bottles of shampoo to sculptures and even an howitzer. They grouped these objects in 44 categories. Besides the raw RGB-D videos they also provide 3D reconstruction for some of the objects. Some example 3D reconstructions can be seen in Figure 2.8. For more information about the reconstruction technique and the number of objects reconstructed we refer the reader to the original paper [52]. All the above datasets are summarized in Table 2.4.

2.4.2 Scene understanding

Scene understanding is a domain that refers to machine learning pipelines that are able to perform several tasks given a scene, such as object detection and localization, scene semantic segmentation, scene classification and more. In general it includes all methods that increase the understanding of a scene through visual means. Due to the significant qualitative difference in terms of applied sensors and the structure of indoor and outdoor scenes, they are considered as separate problems.

One of the first “bigger” datasets is Berkley’s B3DO dataset introduced by Janoch

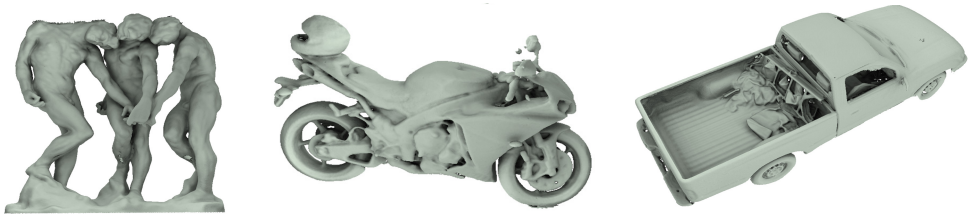


Figure 2.8: Example scans of real objects from Choi et al.'s [52] dataset. Original Figure from [52].

et al. [158]. It is comprised by 849 images from 75 scenes captured by an RGB-D camera. Overall it includes more than 50 object classes. One of the most known datasets and most used benchmarks for indoor scene understanding is the NYUv2, created by Silberman et al. [335] in 2012. It is comprised by a set of indoor videos taken with RGB-D camera, resulting in 795 labeled images with 894 object classes. Xiao et al. [421] tried to provide a richer dataset, in the sense that the segmentation is not pixel wise but there is a better 3D representation of the objects. The result is the SUN 3D dataset [421] which also provides point cloud segmentation produced by Structure from Motion (SfM). Song et al. [344] realized that existing datasets were limited in (i) the number of scenes and sequences they include and (ii) they have sequences from a single RGB-D camera type. They created a more large scale and generic dataset, the SUN-RGBD dataset. They achieved that by taking images from existing datasets and also introducing their own. The result was a dataset with 10,335 RGB-D images of a total of 47 scene categories and 800 object classes. Hua et al. [145] created sceneNN, a dataset that contains 100 scenes with per-pixel annotation of objects. The scenes are 3D reconstructed on triangular meshes.

Most of the scene understanding datasets suffer from small variation of well annotated scenes and limited number of objects. Handa et al. [121] created a method for dataset creation in order to tackle these problems. They claimed that their system is able to create a virtually infinite number of scenes with various objects in them and perfect per-pixel annotation. They accomplish that by using computer graphics to artificially create scenes. They also acquired a large number of 3D CAD models, from some of the datasets mentioned in Section 2.4.1, and randomly placed them in the scenes. The resulting dataset can be used in order to properly pre-train a CNN which can be then fine tuned on a real world dataset. McCormac et al. [241] continued this work with the goal to create a dataset, called SceneNet RGB-D, with annotation not only for semantic segmentation, object detection and instance segmentation but also scene trajectories and optical flow. For comparison, example real scenes from the

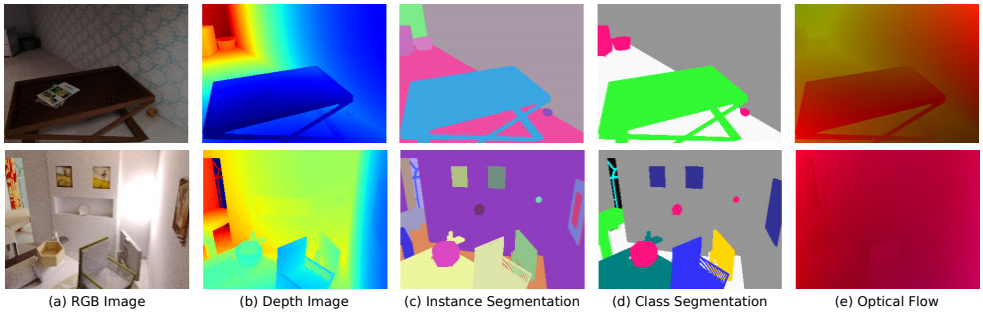


Figure 2.9: Example Images from the SceneNet RGB-D dataset [241]. (a) RGB Image, (b) Depth Image, (c) Ground Truth Instance Segmentation, (d) Ground Truth Class Segmentation, (e) Optical Flow

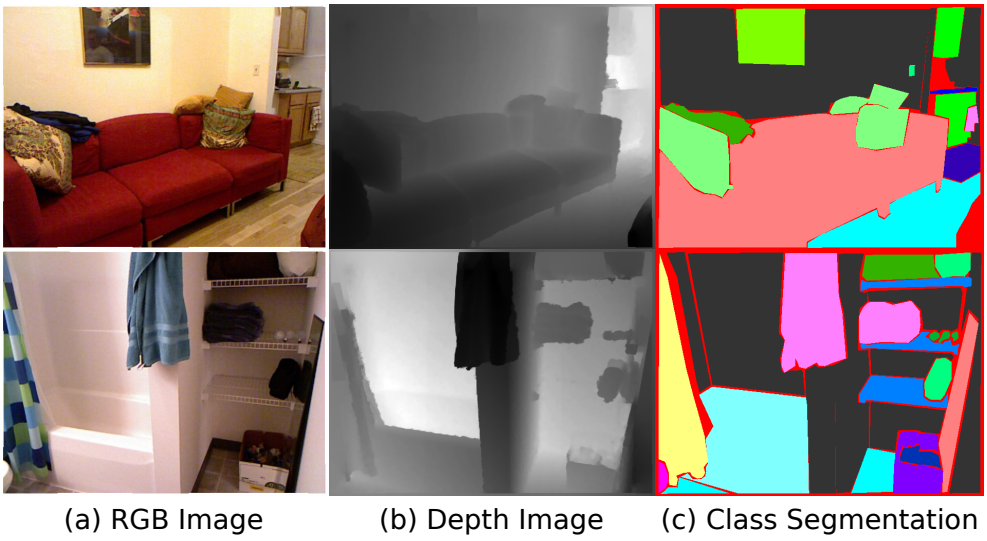


Figure 2.10: Example Images from the NYUv2 dataset [335]. (a) RGB Image, (b) Depth Image, (c) Ground Truth Segmentation

NYUv2 are shown in Figure 2.10 and some artificial scenes from the SceneNet RGB-D in Figure 2.9. Similar to their work, Song et al. [346] created a synthetic 3D scene dataset called SUN-CG, which contains 45,622 synthetic scene layouts created using Planner5D [346]. Dai et al. [61] introduced a much bigger dataset with real world scenes than all the aforementioned. It consists of 1,513 scenes with overall 2.5M RGB-D frames and more than 36K object instances. All scenes have been reconstructed and labeled manually.

For a good comparison the datasets, together with their features and details, are shown in Table 2.5. As with the object datasets of the previous section, we can see that the artificial datasets are orders of magnitude larger than the datasets that contain images and videos of real scenes.

The aforementioned datasets focus only on indoor scenes and objects. When considering outdoor scenes, the availability of datasets decreases significantly. One of the reasons is the low quality of the RGB-D sensors in open space. Most of the existing datasets are limited to 2D RGB images, for example Richter et al.'s [290] dataset and the SYNTHIA dataset [297]. Nonetheless the KITTI dataset [98], although built for pedestrian, car and cyclist detection on images, it also includes Velodyne 64E range scan data with 2D and 3D bounding boxes for 7500+ frames. Moreover, the Sydney Urban Objects dataset [282] contains labeled Velodyne LiDAR scans of 631 urban objects in 26 categories.

2.4.3 Video understanding

Video understanding is a broad field. There are several research areas that can be considered subfields of video understanding, for example action recognition, video retrieval as well as object detection and tracking. Object detection and tracking is highly related to object and scene understanding which are covered in the previous sections. The most active areas in video understanding are action recognition and video retrieval. Most of video understanding related research focuses on action recognition and more specifically human action recognition. Action recognition is the main research area for which new representation approaches and video understanding methods are developed and tested on. There is a large collection of datasets and benchmarks whose content relates a lot to the evolution of the "action recognition" research. Good overviews of these benchmarks and their historic value are given by Hassner [124] and Idrees et al. [151]. In this section we will give an overview of the state of the art datasets and benchmarks.

One of the well known and used benchmarks today is the Human Motion Data Base (HMDB51) [186]. It consists of 6,766 video clips, each representing one out of 51 "every day" actions collected from various sources on the Internet. The annotation

Dataset (Reference)	RGB-D video	Per-pixel annotation	traj. GT	RGB Texture	# scenes	# layouts	# object classes	3D Models avail.
B3DO [158]	No	Key frames	No	Real	75	-	> 50	No
NYUv2 [335]	Yes	Key frames	No	Real	464	464	894	No
SUN 3D [421]	Yes	3D point cloud + Video	No	Real	254	415	-	Yes
SUN RGB-D [344]	No	Key frames	No	Real	-	-	~ 800	No
sceneNN [145]	Yes	Video	Yes	Real	100	100	≥ 63	Yes
SceneNet [121]	No	Key frames	No	non-pr	57	1 000	-	Yes
SceneNet RGB-D [241]	Yes	Video	Yes	pr	57	16 895	255	Yes
SUN-CG [346]	Yes	Video	Yes	non-pr	45 622	45 622	84	Yes
ScanNet [61]	Yes	3D + Video	?	Real	1 513	?	≥ 20	Yes

Table 2.5: Big scale datasets and benchmarks for indoor scene understanding. The first column shows the name of the dataset, the second column shows whether the dataset provides RGB-D video of the scenes, the third one the level of the annotation, the forth one whether trajectory ground truth is included and the fifth whether the data are real, or synthetic. “pr” means photorealistic whilst “non-pr” means non-photorealistic. Sixth, seventh and eighth columns show the number of scenes, layouts and object classes respectively and the ninth, last, column shows whether the dataset provides 3D models of the objects present in the dataset.

is done in a redundant way (each label is verified by at least two humans) in order to ensure its quality. Moreover, every video has some extra meta-data such as camera view-point and motion. Although, for todays standards, this is a small to medium scale dataset, it is still widely used due to its very accurate ground truth. A similarly popular dataset is the UCF101 [348] dataset. It consists of 13,320 clips which belong to one of the 101 action classes of the dataset. These classes are single person actions as well as person to person interactions. Caba Heilbron et al. [39] proposed the ActivityNet, a dataset of human activities. It contains about 20 thousand videos from 203 different human activities. Most videos are between 5 and 10 minutes long with a maximum of 20 minutes. In these videos the classes are manually annotated and specified in time. This results in about 30 thousand human-annotated clips of a specific human action. Recently, Kay et al. [170] proposed the Kinetics dataset, the largest human action dataset to date. It consists of 306,245 trimmed clips from YouTube that include human-object and human-human interactions. The clips are classified to one of the 400 possible classes and were annotated using Amazons Mechanical Turk (AMT) [170].

One of the largest datasets at the time of this paper is the Sports 1M dataset [169]. It consists of 1 million YouTube videos assigned to one of 487 classes. These classes are sport actions such as road bicycle training, track cycling, monster truck etc. These videos have been automatically annotated according to the video tags. Moreover, these are five minutes videos so the class might be a small proportion of the whole video. Due to the above reasons, the labeling of the data is very weak and thus hard to properly evaluate different algorithms. Jiang et al. [161] released the Fudan-Columbia Video Dataset (FCVID), a dataset that contains over 90 thousand videos from 239 categories. Most of these categories are actions like “making cake” whilst there are some object and scene categories as well. The videos are collected from YouTube and are manually labeled. Abu-El-Haija et al. [4] released the largest to date video dataset, the YouTube-8M. It consists of about 8 million videos with 4 thousand labels in total. Each label is supposed to shortly explain the content of the video. For example, a video of biking on dirt roads and cliffs would have a central topic/theme of Mountain Biking, not Dirt, Road, Person, Sky [4]. Possible labels are also filtered out according to some characteristics. For example a label must be visually recognizable and should not require specialized knowledge.

Barekatain et al. [18] introduced an aerial view video dataset for human action recognition it consists of 43 videos with varying camera position and motion. The videos are staged and include multiple actors that perform several actions out of the 12 defined classes. Goyal et al. [104] introduced the ”something - something” dataset. It is an action recognition dataset where the labels are of the form ”something” action ”something”, for example Dropping [something] into [something]. The dataset

Dataset	#Videos	#Clips	#Classes	m-l	trimmed	m-ann
HMDB51 [186]	3 312	6 766	51	No	Yes	Yes
UCF101 [348]	2 500	13 320	101	No	Yes	Yes
Sports 1M [169]	1M	-	487	No	No	No
ActivityNet [39]	19 994	28 108	203	No	Both	Yes
FCVID [161]	91 223	91 223	239	No	No	Yes
YFCC100M [374]	0.8M	-	-	-	No	-
YouTube-8M [4]	$\sim$ 8M	-	4 800	Yes	No	No
Kinetics [170]	306 245	306 245	400	No	Yes	Yes
Okutama - Action [18]	43	43	12	Yes	Yes	Yes
Something-Something [104]	108 499	108 499	174	No	Yes	Yes
Moments in Time [250]	1M	1M	339	No	Yes	Yes

Table 2.6: Big scale datasets and benchmarks for video understanding. First column depicts the name of the dataset, the second third and forth the number of videos, clips and classes respectively. The fifth column (m-l) specifies whether each video can have multiple labels. The sixth column depicts whether the videos are trimmed and the final column (m-ann) shows whether the videos/clips are manually annotated.

is manually annotated and consists of about 108K short videos ($\sim$ 4 seconds) with 174 action classes and more than 23K object names. Monfort et al. [250] introduced the "Moments in Time" dataset, a big dataset of one million 3-second clips with 339 classes of verbs, which are picked from the VerbNet.

A summary of all the above datasets can be found in Table 2.6. For a more comprehensive review on Human Action Recognition datasets the reader is referred to [340].

2.4.4 Other datasets

Besides the scene understanding, object and action classification datasets mentioned in the previous sections there are also datasets for a big variety of applications. For example the Cornell dataset [162] is a dataset built with the goal of training robotic grasp detection on various objects. It contains 1035 RGB-D images with 280 graspable objects annotated with several positive and negative graspable rectangles. For the goal of shape deformation, Yumer et al. [436] created a dataset, containing objects from various categories and their deformations scales, that was later also used for other research purposes, for example [435]. Garcia and Vogiatzis [95] proposed the MovieDB, a dataset for different image-to-video retrieval tasks [94]. The TACoS

dataset [286], with action labels on videos as well as natural language descriptions with temporal locations, and the Charades-STA [92] have been used for text-to-clip video retrieval. The DiDeMo dataset [10] has been introduced for temporal localization given natural language, but has also been used for the purpose of text-to-clip video retrieval [423]. Recently, the Hollywood 3D dataset was proposed [118] which contains 650 stereo clips with 14 action classes, together with stereo calibration and depth reconstruction.

2.5 Research areas

2.5.1 Object classification and recognition

A very well researched topic that includes three dimensional representation of the world is 3D object classification and recognition. Given an object with a 3D representation, a system has to classify the category or the instance of the object. Although, conceptually, a straight forward task, it constitutes a very complex problem because it requires efficient and complicated representation methods that are able to capture the high level content from the raw representation. Moreover, it is a fundamental step in understanding the three dimensional world. As a result, it is considered a very good benchmark for 3D world representation methods. Two large clusters of object classification and recognition methods are identified, depending on the data they process. These are methods that try to classify full 3D objects, usually available as CAD models, and methods that classify RGB-D images of objects.

2.5.1.1 RGB-D object recognition

The first methods applied for this task are inspired by the imaging community. Researchers were trying to develop hand-crafted descriptors that were then used to discriminate between different objects. One of the first examples of such methods is the work of Lai et al. [187], which extracts spin images from the depth map and SIFT features from the RGB values. They create two different vocabularies using the Efficient Match Kernel (EMK) method. The resulted representation is fed into a linear SVM (linSVM), a Gaussian kernel SVM (kSVM) and a random forest (RF) and compare their performance on their RGB-D object dataset [187, 188]. Other works apply the well know Kernel descriptors (KDE) [27] on several characteristics of an RGB-D image whilst other use the Hierarchical Kernel descriptor (HKDE) [25], which applies the kernel descriptor also on the kernel representation instead of only on the pixel level, creating a hierarchy of kernel descriptors.

With the recent success of deep convolutional neural networks (Deep CNN) in image analysis tasks, researchers try to extend these methods to the three dimensional representations as well. One of the first approaches towards training features from data from more than two dimensional representations was done by Bo et al.[28] who learned features in an unsupervised manner from RGB-D data and Socher et al. [342] who trained a Convolutional-Recursive neural network. Alexandre [8] proposed a transfer learning method where different networks are used for each channel (three color channels and depth map). Instead of training each network from scratch they take as initialization method the weights of the best performing network trained so far. Since their experiments aim to test the increase of performance using the transfer learning method, they do not compare to other methods. Unfortunately, they also use a subset of the original dataset which makes the comparison to other methods impractical. Eitel et al. [78] propose a fusion architecture, in which two networks are trained, one on the RGB data, pre-trained on ImageNet [302] and an other on the depth map. The two networks are combined with a late fusion to produce the final result.

Method	Category	Instance
linSVM [187]	81.9 ± 2.8	73.9
kSVM [187]	83.8 ± 3.5	74.8
RF [187]	79.6 ± 4	73.1
KDE [27]	86.2 ± 2.1	84.5
HKDE [25]	84.1 ± 2.2	82.4
Upgraded HMP [28]	87.5 ± 2.9	92.8
CNN-RNN [342]	86.8 ± 3.3	-
Fus-CNN [78]	91.3 ± 1.4	-

Table 2.7: Performance of object recognition methods on the RGB-D Object recognition dataset [187]. The performance is measured by classification accuracy. Left column describes the method, the middle column presents the results on the category level classification benchmark and the right the Instance level classification performance.

We summarize the performance of all the above methods, on the RGB-D Object recognition benchmark [187, 188] in Table 2.7. The benchmark used for this comparison provides two different tasks. One is the category level classification, where a classifier is supposed to label the type of object. The second is instance level classification, where the classifier is supposed to identify the specific object from different views and in different environments.

2.5.1.2 3D object classification

As mentioned in Section 2.2.2.1, early deep learning approaches on learning from a three dimensional representation define two design concepts. The first approach is to

train CNNs straight from a 3 dimensional representation of voxel grids [419], while the second one applies 2D projections. In the context of 3D object classification the projection is done via a multi-view approach [353]. In order to compensate for the increased computational complexity of three dimensional convolutional kernels, the networks are trained on very low resolution inputs (usually do not exceed 30^3 voxels) and they are comprised by a small number of layers, restricting the complexity of the features learned. Most of the proposed methods for 3D object classification belong to one of these two categories.

Both strategies have received a lot of attention. The 3D kernel approach was first applied in this research area by Wu et al. [419]. They utilize a 3D Convolutional DBN, which is trained on their newly proposed ModelNet. The idea of 3D convolutional kernels is further explored with the works of Maturana and Scherer [240], who introduced a 3D CNN as well as a new representation approach. They argued that the voxel representation usually used disregards a lot of information. The reason is than the value of a voxel can only be "occupied" or "empty". Thus, they propose a representation in which the voxel value represents the posterior probability of the cell being occupied. Later, Qi et al. [280] tried to improve the 3D CNN approach in three stages: 1) new network structure, 2) data augmentation, 3) feature pooling. They proposed two new architectures. The first improves the performance by training on an extra auxiliary task, i.e., object classification from sub-volume, and the second mimics the behavior of the multi-view CNN. A specialized convolutional layer performs a 2D projection of the 3D volume and then an image CNN performs the classification. The object is presented to the network in various orientations. The network performs orientation pooling to obtain a orientation invariant representation. Sedaghat et al. [323] argued that the network should, at some point, be able to estimate the pose of an object in order to be able to classify it. Thus, they added an auxiliary task, namely pose estimation and increased the performance of the tested networks by a significant margin on a variety of benchmarks. Hegde and Zadeh [128] fused multi-view and 3D CNNs, whilst Brock et al. [35] defined blocks of layers based on the inception [358] and ResNet [127] architectures, namely Voxception, Voxception-downsample and Voxception-ResNet.

The projection to lower dimensions has also received a lot of attention. As mentioned above, Su et al. [353] proposed a multi-view approach, where pictures of the object are taken from 20 different views and processed by a pre-trained, on ImageNet, network. Shi et al. [327] proposed the projection of the shape on a cylinder, described in Section 2.2.2.1 and Qi et al. [280] improved the multi-view approach by introducing a multi-resolution extension of data augmentation. Wang et al. [392] argued that the view pooling approach of the multi-view strategies fails to take into account important information from different views since only one survives the pooling. In order

Method	Type	ModelNet10	ModelNet40
shapeNet [419]	3D	83.54	77.32
MV-CNN [353]	2D proj.	-	90.1
VoxNet [240]	3D	92.0	83.0
DeepPano [327]	2D proj.	88.66	82.54
MVCNN-MultiRes [280]	2D proj.	-	91.4
MO-AniProbing [280]	3D	-	89.9
ORION [323]	3D	93.9	89.4
FusionNet [128]	Both	93.11	90.8
VRN [35]	3D	93.61	91.33
VRN-Ensemble [35]	3D	**97.14**	**95.54**
Wang et al. [392]	2D proj.	-	93.8

Table 2.8: Performance of object classification methods on the ModelNet 10 and 40 benchmarks [419]. The performance is measured by classification accuracy. Left column describes the method, the middle column presents the results on the ModelNet10 classification benchmark and the right one the performance on the ModelNet40 classification benchmark.

to alleviate this issue they introduced a recurrent clustering and pooling layer based on graph theory. With their approach they achieved state of the art performance on the ModelNet 40 dataset.

The performance of the above methods is summarized on Table 2.8. Although, for the most part multi-view approaches were outperforming the voxel-based approaches, the work of Brock et al. [35] with the Voxception-ResNet approach managed to outperform all multi-view approaches. Nonetheless, their strategy needs to train multiple big networks from scratch whilst the work of Wang et al. [392] only needs to fine tune the networks lowering the training time by multiple orders of magnitude while still having competitive performance.

2.5.2 Semantic segmentation

An important research area using such three dimensional datasets is semantic segmentation. Semantic segmentation or scene labeling is the procedure of labeling every pixel, or voxel, in an image, as seen in Figures 2.9 and 2.10. Most methods tackle this problem by utilizing only RGB images. Since depth sensors became widely accessible, people started to use this extra information in order to make better predictions. The

methods that utilize these features are heavily influenced by their RGB-only counterpart. In this work we will only focus on the methods that utilize the depth information since we are interested in applications and methods that deal with higher than two dimensional data. Most traditional methods tackle this problem by utilizing hand-crafted features, introduced in Section 2.3, in a Conditional Random Field (CRF) or Markov Random Field (MRF) model. The usual pipeline is to over segment the image in super pixels, extract features from the superpixels and then use them to construct unary and pair wise potentials for the CRF or MRF model. With the success of deep learning in image classification, researchers try to adapt these methods for three dimensional semantic segmentation as well.

2.5.2.1 Traditional approaches

The first to tackle this problem in the higher than two dimensional representations are Silberman and Fergus [334]. In their work they use a CRF-based approach and define unary potentials encoding spatial location and pairwise potentials encoding relative depth. The unary potentials are learned from a neural network using local descriptors. They evaluate their approach on their NYUv1 dataset, which they construct for the purpose of their project. Moreover, they test different descriptors, both image and depth descriptors, and compare their performance. They extended their work [335], by introducing a new extended version of NYU, NYUv2, which is still one of the most used datasets for benchmarking scene segmentation algorithms. Couprie [57] explored other CRF-like approaches in order to improve the computational complexity of the algorithm. Ren et al. [288] improved the segmentation performance by using kernel descriptors [26, 27] and by combining superpixel MRF with segmentation trees for contextual modeling.

Koppula et al. [180] oversegmented a 3D pointclound [80] based on smoothness and surface continuity. The segments are then labeled using an MRF. Gupta et al. [116, 115] introduced gravity direction prediction and combine it with appearance and depth contours coupled with either RDFs or SVMs. Hermans et al. [133] proposed an RDF classification which is refined using a Dense CRF.

Deng et al. [65] argue that although hand-crafted features have achieved much progress, the geometric relationship of local and global object spatial configurations have not been studied, leaving room for improvement. They propose a method that jointly considers local and global spatial configurations by adding mutex constraints on Gupta et al.'s method [116].

[351, 352] proposed a method for real time semantic segmentation of RGB-D videos. Their method estimates camera pose by using simultaneous localization and mapping (SLAM) and recognize and segments object classes in the image, using a

GPU implementation of random forests. Müller and Behnke [251] used the output of this method as a feature for unary node potentials on a CRF model. For pairwise depth-sensitive features they use a number of features such as contrast, vertical alignment, depth difference and more. The unary and pairwise functions are learned using structural support vector machines (SSVM).

[174] propose a new CRF model that combines appearance and geometric information in various levels of the models hierarchy. They introduce a new region growing algorithm to extract fundamental geometric planes, and extract appearance and geometric unary potentials from these planes. More information is included by adding pairwise potentials between these planes and extra higher order potentials defined on cliques, which encompass planar patches.

2.5.2.2 Deep learning

As mentioned above, a lot of methods that utilize deep learning have been also developed. Within this category we can identify two clusters of methods. The first represents a transition from the aforementioned traditional methods to the pure deep learning ones. In these the networks are used in order to extract features that are then used to classify segments or superpixels either using graph models like CRF and MRF or some other classifiers. Some examples are the works of Couprie et al. [58] who adopted a multi-scale approach by adapting previous work in semantic segmentation [85, 86], Höft et al. [142] and Wang et al. [391] who proposed a multi-modal unsupervised method that would automatically learn rich high and low-level features from an auto-encoder.

The second cluster is initiated by the work of Long et al.[224], who introduced the Fully Convolutional Networks (FCN) in order to produce per pixel, dense, classifications. These networks are end-to-end trainable and do not rely on other methods. Eigen and Fergus [77] trained a multi-scale convolutional neural network to predict the depth-map, surface normals and provide semantic segmentation. Wang et al. [401] designed two convolutional and deconvolutional networks, one trained on depth values and one on RGB values. These networks explicitly try to learn common features between different modalities (see Section 2.2.2.2). Li et al. [213, 212] proposed an LSTM-CNN approach called LSTM-CF. They first transform the depth data to the HHA form [117]. They use three LSTM networks, namely one for each modality, that explore vertical contexts. The results are concatenated and fed into the last LSTM network which performs bi-directional propagation along the horizontal direction. Hazirbas et al. [125] argue that the aforementioned method is very complicated, resulting in inefficient training. Moreover, they argue that the HHA representation of the depth information is very computationally expensive to compute whilst it does

not provide any new information over the depth channel. In order to gain as much information possible from the depth channel without increasing the preprocessing computational complexity or the complexity of the model they propose the FuseNet. They extended the work of Noh et al. and Badrinarayanan et al. [258, 14] to also utilize depth information, by having two encoder parts, one processing the RGB data and one the depth data. Every few layers of the encoders, the feature maps of the depth data are inserted to the RGB processing network.

Park et al. [271] adapted the very successful work of Lin et al. [215], RefineNet, to use RGB-D data. They do that by introducing the multi-modal feature fusion (MMF) block which fuses feature maps from an RGB specific and a depth specific network. These fused representations are used as input to the refine blocks of RefineNet [215]. Valada et al. [383] used the SSMA (Section 2.2.2.2) module to fuse geometric and color features, while Deng et al. [64] used the interaction stream that they introduced, described in Section 2.2.2.2 as encoders. The outputs of the streams are fused together and sent to a decoder to predict the class labels. Qi et al. [281] introduced a method which combines the two methodologies. They do that by utilizing graph neural networks (GNN) instead of a CRF or MRF. They experiment with unary potentials extracted from a pretrained VGG as well as a ResNet. Moreover, as an update function for the GNN they try both MLP and an LSTM.

The performance of the aforementioned methods on the NYU benchmarks [334, 335] can be seen in Table 2.9. For all benchmarks, the highest performance is reported by deep learning methods, and more specifically the second cluster of the deep learning methods. Nonetheless, the best performing traditional approaches still outperform the first cluster of the deep learning approaches. Table 2.10a shows the performance evaluation of the methods on the SUN-RGBD dataset and 2.10b on the ScanNet. From both tables, it can be seen that the RDF-Net of Park et al. [271] outperforms all other methods by a large margin, on every benchmark tested.

2.5.3 Object detection

Object detection is a conceptually similar task to scene semantic segmentation. The main difference between the two fields is that object detection does not classify all data points in a scene. Instead, the scene is parsed in order to detect the position of specific objects. Traditional methods that tackle these problems mainly depend on template matching [112, 366, 367].

2.5.3.1 Traditional methods

Template matching methods are very popular due to the absence of need for a large annotated training set and training times. On the other hand, a system that is able

Method	Year	Shallow/ Deep	NYUv1	NYUv2					
				4 Classes		40 Classes			
			pixacc	pixacc	clacc	fwavacc	avacc	pixacc	clacc
SIFT+MRF [334]	2011	Shallow	56.6 ± 2.9	-	-	-	-	-	-
Silberman et al. [335]	2012	Shallow	-	58.6	-	-	-	-	-
KDES [288]	2012	Shallow	*76.1 ± 0.9	-	-	-	-	-	-
Gupta et al. [116]	2013	Shallow	-	-	-	45.1	26.1	57.9	*28.4
Hermans et al. [133]	2014	Shallow	59.5	69.0	-	-	-	-	-
RF + SP + CRF [251]	2014	Shallow	-	*72.3	*71.9	-	-	-	-
Khan et al. [174]	2014	Shallow	-	69.2	65.6	-	-	-	-
Gupta et al. [115]	2015	Shallow	-	-	-	45.9	26.8	58.3	-
Deng et al. [65]	2015	Shallow	-	-	-	*48.5	*31.5	*63.8	-
Stückler et al. [352]	2015	Shallow	-	70.9	67.0	-	-	-	-
Couprie et al. [58]	2013	Deep	-	64.5	63.5	-	-	-	-
R-CNN [117]	2014	Deep	-	-	-	47.0	28.6	60.3	35.1
FCN [224]	2015	Deep	-	-	-	49.5	34.0	65.4	46.1
Eigen and Fergus [77]	2015	Deep	-	**83.2**	-	**51.4**	34.1	65.6	45.1
Wang et al. [401]	2016	Deep	**78.8**	-	**74.7**	-	-	-	47.3
RDF-152 [271]	2017	Deep	-	-	-	-	**50.1**	**76.0**	**62.8**
3DGNN [281]	2017	Deep	-	-	-	-	43.1	-	59.5

Table 2.9: Performance evaluation of different methods on the NYU datasets (v1 & v2). First column refers to the methods and the papers that present them. The second column is the year that the methods where published. The third column shows whether the method is follows a traditional approach, or shallow learning, or a deep learning approach. Fourth column shows the per pixel average accuracy on the NYUv1 dataset using all 13 classes. The rest of the columns show performance results on the NYUv2 dataset. The fifth and sixth column refer to the 4-class segmentation task whilst the rest on the 40-class segmentation task [335]. pixacc refers to the average per pixel accuracy, clacc refers to the average per class accuracy, fwavacc is the frequency weighted average accuracy and avacc refers to the meanIU, or the mean Intersection over Union [125]. We highlight the per category (shallow or deep) best performance with a * and the over-all best with **bold** (in which case the asterisk is omitted).

Table 2.10: Semantic segmentation performance evaluation of different methods. The first column refers to the method and the second shows the year the method was published. The rest of the columns show the performance results on the respective benchmark. pixacc refers to the average per pixel accuracy, clacc refers to the average per class accuracy and avacc refers to the meanIU, or the mean Intersection over Union [125]. We highlight the best performance with **bold**. It should be noted that all methods shown in this table are deep learning methods.

Method	Year	SUN-RGBD		
		clacc	avacc	pixacc
*FCN [224]	2015	41.13	30.46	68.35
LSTM-CF [212]	2016	48.1	-	-
FuseNet-SF5 [125]	2016	48.3	37.29	76.27
RDF-152 [271]	2017	**60.1**	**47.7**	**81.5**
SSMA [383]	2018	-	38.4	-

(a) Performance evaluation of different methods on the SUN-RGBD 37 class benchmark [344]. *FCN refers to the work of [224] but the performance on the SUN-RGBD is reported by [383].

Method	Year	avacc
SSMA [383]	2018	57.7
RFB-Net [64]	2019	**59.2**

(b) Performance evaluation of different methods on the ScanNet dataset [61] as reported by the benchmark website.

to detect objects from many classes has to try and fit many templates which makes the algorithms slow for real time applications such as robotics [135]. The pipeline of most of template matching method can be divided into four steps [112]: **1) feature extraction**, where local features are extracted from template object points, **2) feature matching** where features from the new scene are corresponded to the library (template) features. There are several matching strategies, such as threshold based, nearest neighbor approach (NN) based and nearest neighbor distance ratio (NNDR) based [247, 112]. An important step of the matching step is the strategy with which the feature library is searched, such as the naive brute force search, kd-trees [113, 109], hash tables [91] and more. Then, each match is used for a **3) hypothesis generation** where the match is voting for an object position and pose. Popular methods include pose clustering [245, 72, 113, 109, 448], the RANSAC algorithm [269, 270], hough transform [178, 375] and more. Finally, all hypotheses are leveraged in the final **4) hypothesis verification** step. For a more comprehensive study on traditional methods that perform object detection and recognition the reader is referred to [112].

In recent years, learning based methods started getting attention for the problem of 3D object detection and 6 DoF pose estimation. These methods rely on the size of the dataset to provide both positive and negative examples for objects, contrary to template methods that only require positive examples, and thus are more difficult to generalize to new types of scenes. For example, Rios-Cabrera and Tuytelaars [292] extended Hinterstoisser et al.'s [135, 134] method and learn the templates in a discriminative fashion. Since the pipeline of this method is mostly a template method we consider it a "learning boosted" template method. More representative works of learning based method are Song et al.'s [345] sliding shapes method, where a sliding window is fed into an Exemplar-SVM for each object and Bonde et al.'s [30] work who followed a "sliding cuboid" approach coupled with a random forest classifier. Tejani et al. [366, 367] proposed a novel method using latent-class hough forests to detect objects in a 3D scene. The descriptions of objects are used in order to train a multi class hough forest. Recently, Hinterstoisser et al. [137] extended the point pair feature (PPF) [72], by introducing novel sampling and voting schemes that are not influenced as much by occlusion.

2.5.3.2 Deep learning

Besides the traditional methods, people have tried to adapt the deep learning methods to perform object detection and 6 DoF pose estimation as well. One of the first approaches was introduced by Wohlhart et al. [414]. They adapted the template approach to take advantage of the discriminative power of CNNs. In their approach they

trained a CNN to produce a patch descriptor. The training is done by their triplet error function which leverages between similar and dissimilar objects. This is accomplished by having two terms on the error function. One term punishes descriptor difference of the same object with different background noise and clutter whilst the other punishes small distance of descriptors of different objects. This descriptor is then used to describe both the templates and image patches. The detection and pose estimation is done by the k Nearest Neighbors approach. In order to introduce scale invariance, they adjust the size of the patch according to the distance of the center of the patch to the camera so that the "real world" size of the patch is always the same. Balntas et al. [16] extended the aforementioned method by adding one more term to the error function that takes into account the pose of the object as well, punishing similar/dissimilar descriptors for different/same pose respectively. With this method they achieved state of the art results on a modification [414] of the LINEMOD dataset [135].

Krull et al. [185] proposed an analysis-by-synthesis method for 6DoF pose estimation. Their method uses a CNN to calculate an energy function which compares an scene object estimate with a rendered version of the same object from the estimated view point. Doumanoglou et al. [71] followed a similar approach to that of [366, 367], but instead of the LINEMOD [135] descriptor, they trained a sparse AE to learn a discriminative representation. Kehl et al. [172], inspired by [414], introduced a new method, in which they applied a similar approach but instead of training a CNN on positive and negative objects, they used unsupervised learning with a CAE. They argue that by using unsupervised learning, negative examples are not needed any more and the method becomes less dataset-dependent.

Although there are some datasets that are commonly used, such as [135], a lot of different variations of it as well as different datasets are proposed and used for evaluation, making a global comparative analysis very difficult.

2.5.4 Human action classification

To the best of our knowledge, human action classification is the most researched area concerning image sequences, or videos. Given a short video clip that contains humans performing an action, an automated system has to be able and classify the given action. Depending on the dataset these actions might be single human actions, like standing up or opening door, single human actions in a sport environment, or person to person actions, like hugging or kissing. Like with many fields that deal with visual data, early approaches include template matching while a bulk of traditional approaches define interest points in order to describe small clips and using these interest point and special descriptors try to classify the actions. More recent approaches

try to apply deep learning methods to this field as well.

2.5.4.1 Traditional methods

As stated above the very early approaches are based on templates [29, 326, 325]. Unfortunately these methods can not define single templates for each activity which renders them insufficient [295]. Thus, researchers turned their attention to other models, like the Hidden Markov Model (HMM), Hidden Semi-Markov Model (HSMM), Conditional Random Fields (CRF) and support vector machines (SVMs). Another group of methods extract a representation that is derived using the STIP detectors and descriptors introduced in Section 2.3.3. Finally, a group of works exploit trajectories of points in order to describe and classify actions [237, 244, 356, 395, 396, 397], as described in Section 2.3.3.4.

Yamato et al. [424] were the first to apply HMM on the action classification problem. The method first extracts the mesh feature vector [424] for every frame I. Then all features are quantized to codewords. Each frame codeword is then used as input of the HMM. Oliver et al. [263] follow a different approach. They first extract the human positions and their trajectories and utilize a Coupled HMM (CHMM) in order describe pairwise human interactions. Wang and Mori [406] utilized the hidden CRF (HCRF) in order to classify actions. The optical flow based descriptor of [76] is used to describe each frame. Song et al. [347] proposed a hierarchical recursive sequence representation coupled with a CRF model for sequence learning. In order to retrieve the summarized representation for the next level in the hierarchy, the samples are grouped adaptively, i.e., observations are grouped together when they have similar semantic labeling by the CRF model in the previous layer. Fernando et al. [88] tried to model the evolution of the actions in an video. In order to do that he used the "learning to rank" framework on the Fisher Vector representation of each frame.

As mentioned above, many methods followed the classical approach for image classification, utilizing interest points. Schuldt et al. [318] proposed a local SVM approach combined with the BoF representation in order to classify single human actions in videos. Later Laptev et al. [194] test both HoG and HoF to describe the STIPs, as well as a number of different grids for accumulating histograms, and created a BoF representation of the clips. Finally, the clips are classified with a local-SVM [441]. From the combinations that they tested the best performing one was the HoF features in combination with a three way vertical split on the spatial dimensions and no temporal split of the movie clip.

Sun et al. [356] were one of the first to explore trajectories. They extract SIFT trajectories from the clips and measure the average SIFT descriptor along those trajectories. Wang and Schmid [397] used dense trajectories with corrected camera mo-

tion, encodes them using Fisher Vectors and finally classify them using a linear SVM. Kovashka and Grauman [182] proposed a hierarchical feature approach. They created different vocabularies for a BoF representation for multiple scales. From all the aforementioned methods, the only approach that still stands out today and can be compared to the state of the art deep learning methods is the trajectory based improved Dense Trajectories (IDT) of Wang and Schmid [397] and thus it is the only for which we report results.

2.5.4.2 Deep learning

Many deep learning approaches have been proposed for tackling the HAR task. The main bulk of works can be divided in three schemes, namely full 3D CNNs, two-stream networks and CNN-LSTM approaches. Regardless of the class of the method, besides a small number of works, the input to the networks is a small part of the video, usually referred to as clip. The length of these clips can vary from five to sixteen frames. A more detailed overview of the methods is given bellow.

To the best of our knowledge the first to apply deep learning on HAR were Taylor et al. [363]. In their work they proposed a special RBM, the convolutional gated RBM (convGRBM), which is a generalization of the gated RBM (GRBM) [242]. Their method alleviates a limitation of GRBM, the fact that it can not scale up to large inputs. Their method shares weights in all locations of an image and thus can scale to large inputs. As an old approach, this work does not fit with our classification scheme.

Ji et al. [160] proposed the first 3D CNN for action recognition. Their network has five 3D convolutional layers, one 2D convolutional layer and the output, classification layer. Since their network takes as an input only seven frames, it can not take into account actions that span for a longer period. Thus, they use a feature vector from a long span of frames, i.e., the BoW representation using SIFT features of the frames, as auxiliary input through a hidden layer. In a later work, Tran et al. [379] delved into optimizing the architecture of 3D convNets for spatio-temporal learning. Their experiments indicated that uniform kernels (3x3x3) give the best overall performance. Karpathy et al. [169] did a detailed research on what architecture can exploit the time-dimension better. They tested four different strategies, namely single frame network, early, late and slow fusion networks. Interestingly enough, the single frame network has similar performance to the rest, which means that these first approaches towards spatio-temporal understanding using deep CNNs are not able to exploit the temporal dimension as well.

Baccouche et al. [13] also proposed a 3D convolutional neural network. They deal with the long-term actions by building an RNN-LSTM network which takes as input the output of the 3D CNN network. Donahue et al. [70] proposed a very similar

architecture, stacking an LSTM on top of a CNN network and calling the complete architecture Long-term Recurrent Convolutional Neural network (LRCN). The two main differences with the model of [13] are that they train their network end-to-end and that the CNN is pre-trained on ImageNet.

Simonyan and Zisserman [337] proposed a new strategy, the two-stream networks. In this architecture one network processes the RGB values of a single frame whilst an other processes 10 stacked frames of optical flow fields. The spatial network is first pre-trained on ImageNet and thus increasing the performance of the approach. The final decision on the class of a clip is done by averaging the classification results of the separate networks. Wang et al. [403] identified as drawbacks of deep learning approaches on HAR the lack of large data and the limitation of the complexity and depth of the networks applied. In order to alleviate these issues they proposed some "good practices" for training very deep two-stream networks. The first important step is that the temporal network is also pre-trained on images and thus able to be much deeper. Second they utilized state of the art very deep networks, (VGG19 [338] and GoogleNet [359]) for both streams. Furthermore they proposed more data augmentation techniques for the videos and applied smaller learning rates. Feichtenhofer et al. [87] identified two drawbacks with the two-stream strategy as applied until then. (i) It was not able to learn correlations between spatial and temporal features since the fusion happened after the classification and (ii) the temporal scale was limited since the temporal network only considered 10 frames. Also inspired by the work of [254] they proposed a temporal fusion two-stream network. They applied feature map fusion before the last convolutional layer. They fused the two streams and activations from several frames with a 3D convolutional layer followed by a 3D pooling layer. Carreira and Zisserman [43] proposed to inflate existing architectures from images to three dimensions. They do that not only in terms of architecture but also inflate the trained parameters. Given this starting point they trained two networks, one on RGB values and one on optical flow. Finally, they averaged the outputs in order to provide a unified prediction.

Ng et al. [254] followed a different approach, where they make predictions while processing the whole video sequence rather than short clips. They tested several architectures including two-stream networks, LSTM and other temporal feature pooling mechanisms. Applying max-pooling over the temporal dimension in the last convolutional layer (i.e., convPooling) and the LSTM are the two best performing strategies for temporal handling. Their convPooling network takes as input 120 frames whilst the LSTM 30 and both give similar results. In similar work, Varol et al. [384] proposed a Long-Temporal Convolutional network (LTC). Their network is processing 60 frames per video clip. They defined a number of 3D convolutional networks, each processing different resolutions and modality, i.e., RGB and optical flow. The classification scores

Method	Year	+IDT	RGB	Flow	UCF-101	HMDB-51
IDT [397]	2013	-	-	-	86.4	61.7
Two-Stream [337]	2014	No	Yes	Yes	88.0	59.4
Karpathy et al. [169], Sport 1M pre-train	2014	No	Yes	No	65.2	-
TDD [402]	2015	No	Yes	Yes	90.3	63.2
C3D ensemble [379], Sport 1M pre-train	2015	No	Yes	No	85.2	-
Very deep two-stream [403]	2015	No	Yes	Yes	91.4	-
Two-stream fusion [87]	2016	No	Yes	Yes	92.5	65.4
LTC [384], Kinetics pre-train	2017	No	Yes	Yes	91.7	64.8
Two-stream I3D [43], Kinetics pre-train	2017	No	Yes	Yes	**97.9**	**80.2**
(2+1)D [380], Kinetics+Sports 1M pre-train	2018	No	Yes	Yes	97.3	78.7
TDD + IDT [402]	2015	Yes	Yes	Yes	91.5	65.9
C3D ensemble + IDT [379], Sport 1M pre-train	2015	Yes	Yes	No	90.1	-
Dynamic Image Networks + IDT [23]	2016	Yes	Yes	No	89.1	65.2
Two-stream fusion + IDT [87]	2016	Yes	Yes	Yes	93.5	69.2
LTC+IDT [384], Kinetics pre-train	2017	Yes	Yes	Yes	92.7	67.2

Table 2.11: Performance evaluation of different methods on the UCF-101 [348] and HMDB-51 [186] datasets. The first column refers to the method and the second shows the year the method was published. The third column specifies whether IDT is used in combination with the networks. The forth and fifth column show whether the method is utilizing RGB and optical flow inputs respectively. The sixth and seventh columns show classification accuracies of the methods on the UCF-101 and HMDB-51 datasets, respectively.

of all networks are averaged in order to produce the final prediction.

Wang et al. [402] proposed the trajectory-pooled CNNs (TDDs). Inspired by the work of [397] and the lack of CNNs in exploiting long term temporal relationships, they proposed the Trajectory pooled Deep-Convolutional Descriptors (TDDs), where they compute descriptors by computing trajectories of CNN features maps using the method of [397] and encoding them using Fisher Vectors.

Tran et al. [380] proposed to decompose the spatial to the temporal convolution, and thus creating the (2+1)D convolution which is a 2D spatial convolution followed by a 1D convolution exploiting the temporal dimension. Their top performing network is a (2+1)D, two stream network which has a much lower complexity than the top performing 3D networks, while keeping the performance competitive.

We summarize the results of some of the above methods on Table 2.11. There are several conclusions we can derive from these results. Simple 3D networks seem to be outperformed by CNN-LSTM as well as two stream networks, but the combination of them outperform the 'single solution' networks. Moreover, pretraining on large datasets with not very accurate annotation, such as Sports 1M [169], benefit the quality of the networks. Last but not least, as with many applications, the best performing traditional approach, IDT [397], is outperformed by most deep recent deep learning approaches. Nonetheless, the combination of IDT and networks produces better results, by a constantly large margin, driving us to the conclusion that the high-level hand crafted features seem to capture information that is not learned by the networks, rendering them complementary.

2.5.5 Other areas

There are numerous more research areas and applications that deal with high dimensional data. Some examples are:

2.5.5.1 Outdoor object detection

Outdoor object detection is a very well studied research topic with many real life applications, like autonomous vehicles and security. Some more specific examples of object detections are pedestrian detection, vehicle detection, like cars motorcycles and bicycles. Traditional methods first segmented the input point cloud and then classified the segments with various methods [365, 364, 393, 21]. For example, Behley et al. [21] used the BoW model to describe each segment and used it to classify it. State of the art methods take advantage of deep neural networks. Some examples are [278, 82]. Qi et al. [278] uses the pointnet++ as a base, whilst [82] utilizes 3D convolutional kernels and [207] utilizes a 2D FCN with the depth data as an extra

modality. To the best of our knowledge [278] achieves the state of the art performance on the KITTI benchmark [98].

2.5.5.2 Landing zone detection.

This is a very important task for autonomous rotorcrafts. It provides the ability to localize possible landing locations. Methods that perform such a task usually depend on LiDAR [320] data. Traditional approaches depend simple geometric properties such as terrain roughness and slope [410, 315, 165, 407]. The performance of such methods lowers significantly once the terrain is covered by low vegetation. Maturane and Scherer [239] propose a deep learning approach which utilizes a CNN with three dimensional convolutional kernels which predicts which parts of the seen terrain are safe for landing and which are not.

2.5.5.3 Structure from motion (SfM) and Simultaneous localization and mapping (SLAM)

These are very challenging tasks. SLAM is the process where the algorithm is trying to identify the position of the camera or sensor in the environment while constructing a map of the environment. SLAM is a very challenging while very interesting and important task in the field of robotics as well as augmented reality. Traditionally people were trying to match new environment parts to the constructed map by matching features (usually hand crafted) and RANSAC like algorithms. Some representative work can be found in [350, 80, 173, 81, 411, 252]. SfM is the process of building a 3D representation of a scene/environment of a camera by using multiple views, and more specifically views from the same camera as it moves in the space. It usually is part of SLAM since it tries to built a 3D representation of the local environment of the camera. A comprehensive survey on SLAM and SfM was recently published by Saputra et al. [312].

2.5.5.4 Action Recognition in 3D videos

This is a relatively new research field. As with video action recognition the target of the task is the classification of human actions in different kind of categories. The methods applied in this field can be divided into two categories depending on the type of data they process. More precisely, they process skeleton data or depth data [284]. Also methods that process color-data have been proposed but since these are much closer to the 2D Video action recognition, described in Section 2.5.4, than the rest of these methods we do not consider it as part of this section. Skeleton-based approaches first extract the joints positions, usually using the OpenNI tracking framework [332],

and then either use them [427], or information from the area around them [400, 399], to describe the motion. Depth-based approaches use either silhouettes [208, 385] or 4D histogram descriptors [284, 265, 428] in a BoW framework to describe each action and then try to classify them. In recent years plenty of DL approaches have been proposed as well. They usually utilize an RNN-LSTM on joints and skeletons [324, 73, 219] or process directly the depth data in time [404]. For a good overview of deep learning approaches the user is referred to [446].

2.6 Discussion

Although this field has come a long way, there are still a lot of challenges that the researchers face. Since most of these methods are generalized from successful methods developed for two dimensional images, all limitations and problems that arise when dealing with two dimensional images exist here as well. For example, when it comes to deep learning, the models are typically not understood and treated as black boxes [110]. Although researchers know how these models update their parameters and learn from the data, retrieving the information that they have learned is still an open research area. More specifically, although there has been done research on feature visualization [440, 336, 434], it is still unknown how to discover or understand what the networks learn and how they behave. Another inherent limitation is the typical lack of rotation invariance of the models, although some methods try to work around it. For example, Cheng et al. [49] train a specific layer to be orientation invariant. They do that by adding a penalty term to the loss function to force the layer to become rotation invariant. Although the result of the specific layer is rotation invariant the rest of the network is not. In cases where information from multiple layers is needed, such as semantic segmentation, this solution does not suffice. Another example is the work of Marcos et al. [234]. They rotate the kernels and convolve with the rotated kernels and thus obtain responses from all possible orientations. The rotation invariance of this strategy is also limited since the information of the orientation is getting lost during the orientation pooling operation.

Besides the inherited difficulties from the two dimensional case, other problems arise when trying to extrapolate to more dimensions, either when the increase is an increase of physical dimensions or if it is an increase of available modalities. A common limitation to all state of the art methods that deal with higher than two dimensional data is the high demand of resources. This limits the possible size of the deep learning methods. Moreover, as shown from the two dimensional case, these methods highly depend on the complexity and size of the resulted models [127, 358, 110, 146], which combined with the increased complexity of the data as well as the

increase of demand renders it very difficult to efficiently apply them.

According to the results shown in this chapter, the state of the art performance on volumetric data is achieved using deep learning models. As described above, these methods have many drawbacks, both inherited from the drawbacks of deep learning in general as well as drawbacks regarding computational complexity. Moreover, it is still unclear which strategy for dealing with the higher dimensionality of the data is better. To be more precise it is still unclear whether reducing the dimensionality to two is better than using three dimensional kernels. In the later case it is still unclear which representation of the data works best. All these questions are left unanswered whilst the computational complexity of the models together with the lack of very large scale, high dimensional, diverse and well annotated datasets make the unbiased comparison between approaches very hard.

Difficulties arise when processing spatio-temporal data as well. Although current results show that methods that utilize optical flow outperform methods that do not, it is still unclear how to optimally include this information. Moreover, the difference of space and time is still a challenging concept. It is still not clear how to process them in order to acquire as much information as possible from both spatial contexts as well as their temporal interactions. Furthermore, most approaches process only short term interactions and only a few process more that 16 frames long clips, and thus encoding long term interactions [384]. Processing many frames though becomes very computationally expensive and thus the question of how to optimally perform temporal and spatial pooling arises. Although there has been significant development in the field the long term impact and directions for continued advances are still unclear. Some of the limiting factors being the fundamental theory for understanding the strengths and limitations of the networks, approaches for learning with small training sets and/or the availability of accurately annotated, diverse and large scale real life datasets.

2.6.1 Major challenges

In summary, the major challenges as described by the research community are:

- Deep learning in high dimensional data is very computationally and memory expensive, limiting the capabilities of the applied approaches.
- Deep learning approaches lack invariance in many transformations, such as scale and rotation, which are usually tackled by very computationally expensive approaches.
- There exist many competing strategies for handling high dimensional data and it is still not clear which approaches are suited better for which type of data and more importantly why.

- For many applications there are not enough labeled data to properly train and test methods. Nonetheless, the past few years, in some research areas this issue has been slowly tackled by introducing large scale datasets such as the ScanNet [61] and the Moments in Time [250].

2.6.2 Future work

According to this study, there is significant room for improvement in all research areas covered in this chapter. Nonetheless, we can identify some common issues to most of them. In most cases deep learning approaches are too computationally expensive for many real world applications, whilst the traditional counterparts have much lower performance. It is important to get as high performing approaches while minimizing computational complexity and memory demands. Moreover, being able to leverage information from different modalities without performing unnecessary computations for common features whilst not missing modality specific information is very important to the whole field. Although there are similarities in the type of dimensionality increase in different research areas, the solutions applied are usually unique to the research area. It would be interesting to acquire knowledge from multiple approaches and create unified solutions.

2.7 Conclusions

This chapter presents a comprehensive review of methodologies, data types, datasets, benchmarks and applications of computer vision on high dimensional data (higher than 2D). Based on the recent research literature we identify four main data sources, namely image videos, RGB-D images and videos, and 3D object models, such as CAD models. Moreover, we identify common practices between methods that are applied on all data types despite their qualitative difference. For example, deep learning approaches and hand crafted features, such as histograms, are developed and applied on all data types and research areas considered. Most of the methods are inspired by previous work in computer vision on 2D data.

Regarding deep learning methods, we discuss the interrelationships and give a categorization of generalization of methods to higher dimensions, namely generalization in case of increase of physical dimensions and generalization in case of increase of modalities, or information per physical position. Finally, we review and discuss the state of the art methods on the most researched areas using these data, such as 3D object recognition, classification and detection, 3D scene semantic segmentation, human action recognition and more.

According to this study, some conclusions can be drawn regarding the top performing approaches. Deep learning approaches seem to outperform hand crafted feature based approaches when it comes to recognition performance in all tested settings (i.e., object classification, recognition and detection, semantic segmentation and human action classification). Nonetheless, hand-crafted feature based approaches have much lower time complexity. In some cases they can produce similar performance to the state of the art deep learning method, as shown in object detection by Tejani et al. [366, 367]. As shown in Human action Recognition, with the IDT approach [396], the hand crafted features can provide complementary information to the deep learning features increasing the overall performance of a system by a significant margin. When the number of physical dimensions is increasing, although early experiments showed that projecting information to lower dimensions and taking advantage of large available systems outperformed the raw processing of the high dimensional data, nowadays, we see an opposite trend. For example, the work by Brock et al. [35] on object detection as well as Carreira and Zisserman [43] on HAR, outperform 2D projection methods. Finally, late fusion seems to be the best performing naive strategy across the board for combining different modalities, whilst fusion in multiple levels and fusion on multiple stages of the process seem to outperform all other methods, e.g. Wang et al. [401] and Park et al. [271].

Understanding the world around us is a difficult task [222]. Although there is a lot of progress in this area, there is still a lot of room for improvement. For most data types there is no clear solution or approach that properly handles the extra dimensions. For example, even in the well studied area of video understanding, there is not a definitive way to handle the difference between space and time. Similarly, in the three dimensional static world even the optimal raw format of the data, e.g. point cloud, 3D mesh or voxelized, is unknown.

3

Deep learning for computational fluid dynamics simulation output

Computational Fluid Dynamics (CFD) simulations are able to produce complex and large outputs that accurately describe the physical properties of fluids and gases in various domains, such as air flow around a car, or the multi-phase flow inside an internal combustion engine. The simulation results, i.e. the flow fields, are often too complex to be analyzed directly. With the increasing number of simulations as well as their complexity, there is a need of automated processes that can analyze these complex outputs. In this chapter, inspired by the success of convolutional neural networks (CNNs) in Computer Vision, CNNs are applied for the first time on CFD output. We show their capabilities in capturing and processing flow patterns. Furthermore, a novel CNN architecture is designed tailored to the data produced by CFD simulations, as well as two conventional architectures. A new dataset of turbulent flow is proposed and constructed, within the application domain of steady flow around passenger cars. The approaches developed are evaluated and compared on the aforementioned dataset, on different tasks that depend on flow patterns. Finally, the CNN approaches are compared to a baseline k-nearest neighbor approach, tuned to be comparable to the state of the art.

3.1 Introduction

3.1.1 Computational fluid dynamics simulations

Compared to physical experiments, CFD simulations provide a cheap way to test, analyze, and optimize complex engineering designs, such as minimizing the drag force applied by the air on a moving object such as a car or an airplane. Such simulations are also used in medical applications, for example simulating the flow in the arteries and calculating the sheer stress can help discover potential threats to our health [442]. These benefits and issues motivate methods that can explore all the information given by the simulation and can help the automated optimization of engineering systems, as well as help us understand complex phenomena around us.

Computational Fluid Dynamics (CFD) simulations produce very complex and information rich outputs. They usually produce results on 3D or 4D (3D + time) space with many physical properties per point (pressure, velocity, turbulent kinetic energy, etc.). Such outputs are difficult to analyze directly in detail and to interpret and derive conclusions from [105]. In order to analyze these results, data reduction and feature extraction methods that extract specific properties of the flow and present them in a visually understandable way need to be employed [97]. With these methods an engineer can look only at specific features and properties at a time making it difficult to analyze the information as a whole. Moreover, this method of analyzing simulation outputs requires a lot of manual labor per simulation. Thus there is a need for an automated way to extract information from, and analyze, CFD simulation outputs that is capable of exploiting all the information available.

This chapter focuses on investigating the potential of deep learning, and more specifically convolutional neural networks (CNNs), on learning patterns of the flow fields by taking into account all given information (the velocity field, pressure field, turbulent kinetic energy and turbulent viscosity). Moreover, the network's ability to provide a lower dimensional yet discriminative representation is evaluated. The state of the art representation of flow fields is the Vector Field Topology (VFT) introduced by Helman and Hesselink [130]. This method suffers from interpolation errors and is not always able to capture all information [405]. More specifically, VFT, as the name suggests, focuses on vector fields and as a result only processes the velocity field while it completely neglects other information available. Moreover, the features used for this representation are hand crafted. This is especially problematic since it is a very hard task to define general features which can be applied to many different application domains without limiting the detectable physical features or loosing discriminative ability.

3.1.2 Convolutional neural networks

In recent years, deep learning approaches have outperformed the hand-crafted approaches by a large margin in a variety of computer vision tasks like image classification [184], semantic segmentation [224], 3D object detection [240] and many more (see Section 2.5). They have demonstrated to successfully capture semantic information and exploit complex patterns, without being limited by the imagination of the scientist that designs them. This constitutes a major advantage of deep learning over hand-crafted features, which are also exploited in the domain of fluid flows in Chapter 4. However, they are limited by the complexity and diversity of the examples they have been trained on [280].

A key component of the success of these methods is the availability of large scale, fully annotated and diverse image and video labeled datasets, like ImageNet [302]. The application of deep learning techniques to fluid flows is a little more problematic as compared to image or video datasets. Realistic CFD simulations which try to simulate viscous effects and turbulence need hours or even days to compute on high performance compute clusters. Additionally, there exists a multitude of configuration options such as design parameters specifying the geometry, the spacial and temporal grid on which the flow is going to be solved, the number of solver iterations to converge the flow field and many more. Thus producing a large and diverse dataset on which a deep learning approach might be trained is a very long process that requires a lot of resources, both computing infrastructure and human labor. Such complex CFD simulations also have the tendency to sometimes not converge at all. This introduces either the need to supervise each simulation during dataset creation in order to detect failed simulations, or if they are not detected, the dataset will contain a considerable amount of noise, i.e., unconverged and thus unrealistic flow fields. Moreover, such CFD simulations produce large and high dimensional results usually composed of millions of grid points, making each data example very big in size and thus difficult to use in a deep learning approach, which relies on GPU memory for their computations.

Such problems seem to make the direct application of deep learning approaches to CFD output data almost impossible. In this chapter we try to tackle these problems by using a partially automated way to create and simulate new designs, which enforces convergence with as little supervision as possible. Tasks for the CNN are designed, that should force the CNN to identify patterns of the flow field while learning to complete multiple tasks to a given accuracy. Each data sample to be processed is build up from many points in space (3D), where the velocity vector and three scalar fields (pressure, turbulent kinetic energy, and turbulent viscosity) are associated with each point. In order to avoid even larger resource demand, we are simulating viscous but incompressible flow where the density is assumed to be constant. The extension

to compressible flow or more general flows, like multi-phase flow, is conceptually straight forward, as the additional fields just need to be added as input variables. A number of CNN architectures are designed, tailored and optimized for this kind of data and we evaluate their strengths and weaknesses. Since it is not trivial to find tasks that will depend on any possible patterns that might appear, a supervised CNN trained on them will learn to recognize only the patterns specific to the given task. In order to force the CNNs to learn a more general representation they are forced to learn more than one task at the same time while also trying to reconstruct the input. We compare our results with a k-nearest neighbor approach, using optimum conditions, in order to make it competitive.

The rest of this chapter is structured as following: In Section 3.2 the related work in the field of flow field feature extraction and convolutional neural networks is outlined, Section 3.3 describes the dataset constructed as well as the tasks defined with it. In Section 3.4 the proposed methods are described and in Section 3.5 the experiments are shown. Finally, the conclusions from the experiments are drawn in Section 3.6.

3.2 Related work

3.2.1 Flow field pattern recognition

The analysis of steady flows has been researched for many years. Many interesting and useful features of steady flow fields exist, even though they are not always very well defined. There are two categories of features for steady flow fields, local and global features. Local features are features that have specific local behavior of the flow and they mostly can be mathematically defined precisely. Some examples of local features are the critical points of a vector field [130]. These features are used by Vector Field Topology (VFT) in order to produce a visually comprehensible representation of the flow, as introduced by Helman and Hesselink [130]. There are many algorithms that try to extract them, all having their limitations and advantages. Global features usually do not have a single definition and the algorithms extracting them need a lot of manual processing and fine tuning in order to produce a desired result [275]. For example, such features are vortices, shock waves, and flow separation. A good overview of flow field feature extraction and visualization can be found in [275, 196, 405].

Although the above mentioned methods deal with the same data as we do they have some core differences. VFT only takes into account the vector fields, which in fluid flows means the velocity and completely neglects the rest of the data. The majority of global flow features can not be automatically extracted in a reliable way, since

they require manual tuning for each application and data example. As such they are inefficient for a large scale automated machine learning approach, which is the target of this work.

3.2.2 Convolutional neural networks for CFD simulation output

Previous approaches have applied CNNs to the CFD simulation domain. In those studies, the target was to either speed up the simulation itself or predict the simulation output from the input design. This is in contrast to the current approach, where we try to learn from the simulation output. For example, [108] tries to predict the output of Lattice Boltzmann Method Simulation of laminar flow, which is much faster to compute and much simpler than the currently used turbulent viscous flow. [218] use neural networks to predict the Reynolds stress anisotropy tensor in order to get a more accurate approximation in less time, leading to more accurate and faster simulations.

To the best of our knowledge this is the first work that tries to apply CNNs on the output of simulations.

3.3 Dataset collection

In order for a network to be able to learn flow patterns, a big and diverse dataset is needed. Moreover there is a need of tasks that depend on flow patterns so a network can be forced to learn general flow features in order to solve the task.

3.3.1 Example creation

One of the contributions of this work is the creation of a large dataset of turbulent flow fields within the application domain of steady flow around passenger cars. In order for CNNs to be trained a diverse and big dataset is needed. Given the complexity and time needed to calculate such simulations this process is not trivial. Starting from a set of given car shapes, an automatized shape deformation setup is employed, which utilized free form deformations [243, 333] to generate variations of the starting shapes. The computational CFD grid is then created for each deformed shape in a way which adjusts itself to the specific geometry, in order to have a sufficient resolution where necessary while keeping the grid coarse where complexity is not needed. This is achieved by using the `snappyHexMesh` meshing tool from the `OpenFOAM` package [264]. The flow field is obtained by running the `simpleFoam` solver where the boundary conditions are specified by a constant inflow velocity magnitude $44.5\frac{m}{s}$ coming from the front with a very small angle.

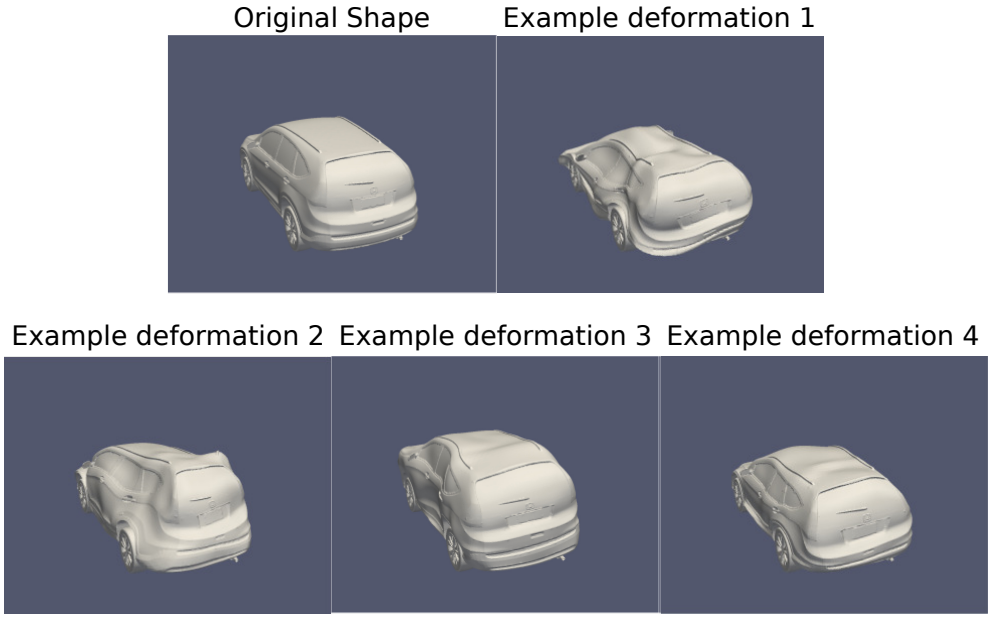


Figure 3.1: Example deformations of a passenger car, for four different deformation scales.

For this first study two base models for passenger cars are used. The deformed shapes are obtained by defining a number of control points on the car surface which are moved randomly, stretching the shape as they move. In order to avoid very sharp edges on the deformed shape the deformations are smoothed out where neighboring control points can not have arbitrary large distance. Example deformations for four different deformation scales are shown in Figure 3.1.

Each simulation provides many different attributes of the flow in every location of the mesh used to calculate it. These are the velocity (3 components) ($\vec{U}$), turbulent viscosity (ν_t), turbulent kinetic energy (k) and pressure (p). For every simulation some metrics are also calculated from the flow, more specifically the forces and torque applied to the passengers car due to pressure difference as well as due to friction.

Each base car is deformed with eight different deformation scales. For each scale 1,024 random deformations are performed, resulting in 16,384 examples. Even though our pipeline is designed to increase the chances of the simulation to converge, some shapes are still deformed in a way that the simulation could not converge. These examples are discarded, resulting in 14,238 usable examples. These are split into training and test sets. The split is done randomly, by picking 498 examples for the

test set. The remaining ones constitute the training set.

Most of the simulation domain contains almost uniform air flow, which is not interesting for this application. Thus a box behind the car is cropped where most of the flow patterns appear. The cropping procedure is done in a randomized way and each crop is used as a separate example enlarging the dataset even further. The flow field is calculated on a mesh of irregular tetrahedrals. In order to make them optimal for being input to CNNs, the flow is interpolated on a fixed uniform grid. We tried three different grid scales, namely 96x64x32, 192x128x64 and 384x256x128 voxels. Due to the process and memory limitations of currently available GPUs, the only feasible resolution is 96x64x32. Regarding the flow physics, this constitutes a very coarse representation of the flow field where most of the small scale details have been averaged out. However, the large scale structures are still preserved and due to the nature of fluid flow, where strong correlations exist between large and small scale structures, we hope to be able to capture relevant flow features. Additionally, in the future with increasing computational power much finer resolutions might be possible. The velocity field is mapped on three channels, one for each direction (x, y, z), and the scalar fields take one channel each, resulting in six channels overall.

3.3.2 Training tasks

Convolutional neural networks have demonstrated high capacity of learning semantically meaningful representations in many computer vision tasks. One of the key components needed to achieve that is the availability of large and diverse datasets, accompanied with tasks that depend on these semantics, which steer the training of the networks. In order to exploit the capabilities of CNNs in identifying meaningful patterns in flow fields, without knowing in advance which these are, tasks that depend on the properties of the flow are needed for having an efficient training procedure. For the networks described in this chapter we considered three tasks, namely force regression, flow prediction and flow reconstruction.

3.3.2.1 Force regression

Calculating the forces which act on a geometry due to the flow is a pretty straight forward task and easily computable from the flow around that geometry. Nonetheless, when considering a flow of a specific direction, the patterns that appear after the geometry in the direction of the flow are an indication of the forces that are applied on the geometry. For example, at the back of a very tall car there would be big vortices which increase the drag force on the car. On the contrary, if the car is short these vortices would be much smaller and possibly not even there. Additionally, areas of reversed flow behind the car also crucially depend on the exact car shape and create

more patterns that also reflect the forces acting on the car. This relationship of downstream patterns of the flow and the forces on the shape is exploited in order to force the network to identify relevant flow features. The network is only presented the flow behind the car and it should learn to predict the forces as well as the torque acting on the car.

3.3.2.2 Flow prediction

As it is well known from fluid dynamics, the patterns of a flow are interdependent. The flow field at a specific point is in causal relation to a large part of the flow field at distant locations, possibly even the whole flow volume (details of this depend on the general flow conditions, see for example [60]). In an attempt to also exploit this dependence of patterns of the flow we introduce the flow prediction task. Namely, given a part of the flow, the network is asked to predict the downstream flow.

3.3.2.3 Reconstruction

Encoding and decoding data in order to extract features is a well known practice in the computer vision community. Usual methods include deep auto-encoders or Boltzmann machines [110]. In this work we also try to exploit the power of these methods. In order to force the representation to be discriminative and meaningful, the reconstruction is done in parallel with the other two tasks.

3.4 Network architecture and training details

3.4.1 General network architecture

Most of the existing work with applying CNNs on three dimensional data can be divided into two main groups [280]: (i) Applying 3D convolutions [419, 240] and (ii) projecting the input to one or multiple 2D representations and using of-the-shelf state of the art networks pre-trained on ImageNet [353, 327]. To the best of our knowledge (see Chapter 2), taking 2D projections outperforms the 3D convolution in object classification and recognition tasks [280]. There are three main reasons for this finding. First, the 3D representation of the objects does not take full advantage of the extra dimension. Since the objects in 3D are usually represented by a occupancy grid the only information given is whether a voxel belongs to the object or not. Secondly, the 2D projections can take advantage of very deep networks trained on the very big ImageNet dataset. A comparably large dataset is not available for 3D data. Thirdly, the 2D projections of objects are very closely related to images of objects which is the content of ImageNet and making use of networks trained on it is ideal for 3D objects.

The examples in this case are very rich in 3D information, since all values change in all three directions with various gradients. Taking 2D projections or slices would largely decrease the information content of the network input. On top of that, although we would be able to use state of the art very deep architectures, the data would have very different structures and statistics as compared to images from objects. This would render most of a 2D network layers irrelevant since they are trained to the task of recognizing objects. For the above reasons a 3D approach is followed rather than a 2D projections. Nonetheless, this is just an assumption and thus it needs to be verified with experiments.

As mentioned in Section 3.3.1, the input data has six channels: three from the vector field ($\vec{U}$) and the remaining three from the pressure field (p), turbulent kinetic energy (k) and turbulent viscosity (ν_t). These constitute different modalities for the data. The goal of this work is to design a system which is able to analyze a flow field without disregarding any information. Thus, all channels are used as an input. For images it is common practice for a network to process all channels with the same feature maps. Due to the curse of dimensionality, when scaling to three dimensional problems, it becomes very restrictive in the size of networks that can be trained, both in terms of memory and computational complexity. A solution to this problem applied in many methods that solve tasks with high dimensional input is to split the input and feed each channel to a separate network and finally fuse the models to make the final prediction. This strategy is followed in order to be able to construct deeper networks. We consider three different schemes of splitting the input.

One option for organizing the inputs is to consider each of the six channels separately. However, this results in a very high number of free parameters for the network to be trained. In order to reduce the number of parameters, the following approach seems reasonable. Since three of the channels contain scalar fields (pressure, turbulent viscosity, turbulent kinetic energy) we expect similar low level features for them such as edges and ridges. In an attempt to take advantage of that, one option is to processes all three scalar fields with the same network which has only one input channel, i.e., the networks processing the scalar fields share the same weights. In contrast to that, the three channels of the velocity field are expected to show a different type of interdependency than the scalar fields since they constitute a coherent vector field. We envisage that processing these channels together can be beneficial and thus we consider another option which processes the velocity with one network which has three input channels.

The above described options facilitate four different schemes for organizing the input from which the following three are considered for further experiments: The baseline network is defined with sharing weights networks for processing the scalar fields and one network with three input channels for processing the velocity field. We

Table 3.1: Number of feature maps per layer for different schemes, as well as the "Common layers" of the five and six layer networks. $(/2)$ denotes a max-pooling layer after the denoted layer. "Common layers" refers to the part of the network after the fusion of the feature maps (see Figure 3.3).

	Scalar Layer 1	Scalar Layer 2 $(/2)$	Vector Layer 1	Vector Layer 2 $(/2)$
FMS	16*3	32*3	64	128
DS	16	32	21*3	42*3
VC	16	32	64	128
VC^*	16	32	21	42
Common Layers	Layer 3	Layer 4 $(/2)$	Layer 5	Layer 6
All above	128	128	64	-
VC^{**}	128	128	64 $(/2)$	64

refer to this network as Velocity Coherent (VC). The second network differs from the baseline VC network by using three independent networks for processing the scalar fields, one for each scalar field, while it still retains one network with three input channels for processing the velocity field. We refer to this network as Full Modality Specific network (FMS). Finally, the third network differs from the VC on how it processes the velocity field, for which it utilizes three separate networks, one for each direction of the velocity (but still has sharing weights networks for processing the scalar fields). This network is referred to as Direction Specific (DS) network. These three networks are visualized in Figure 3.2.

In all convolutional layers, the kernel is of size 3x3x3. Each layer is followed by batch normalization [154] and the ReLU activation function.

Independent of which scheme is used, the rest of the architecture follows the same principles. After a few convolutional and pooling layers, the resulting feature maps are fused together by concatenating them. The fused representation is further processed by more convolutional and pooling layers. The depth on which the fusion happens is a parameter to be experimented with. Overall, two different network depths are tested in the encoding stage which have either five or six layers. Finally, the resulting feature representation is passed to different output networks which perform one of the tasks defined above.

As an example, the work flow for the VC network is shown in Figure 3.3. The number of feature maps of the five and six layer networks as well as each different scheme are shown in Table 3.1.

Figure 3.2: The three input schemes proposed and evaluated.

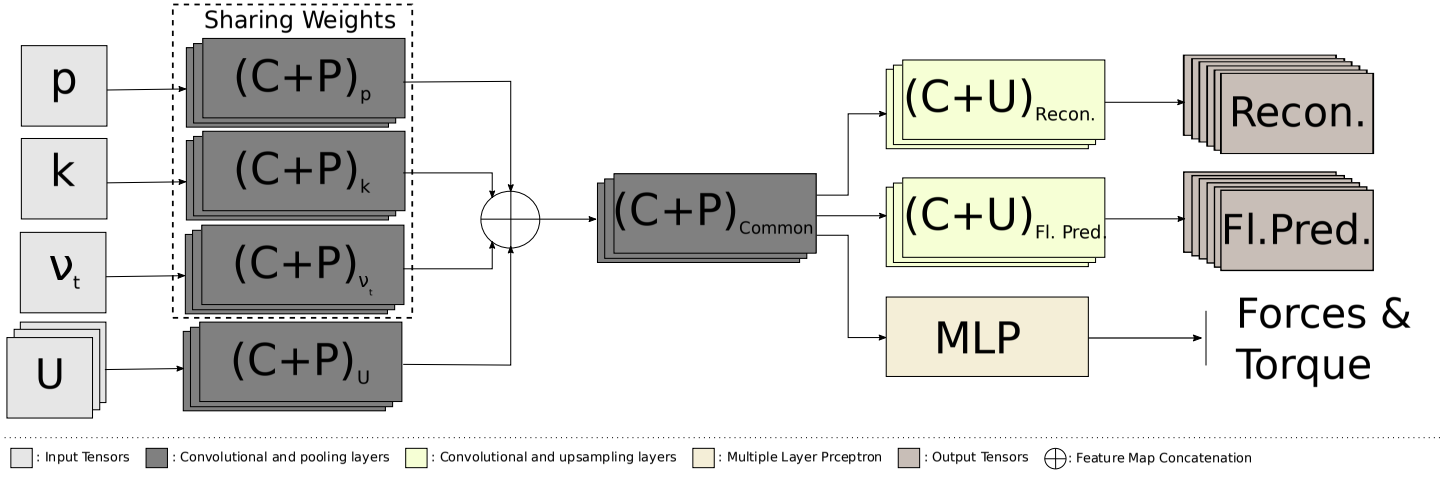


Figure 3.3: General Architecture of the velocity coherent (VC) Network. The network takes as an input the Velocity field ($\vec{U}$), turbulent kinetic energy (k), pressure (p) and turbulent viscosity (ν_t). It performs three tasks simultaneously namely, input reconstruction, flow prediction and force regression.

3.4.2 Prediction networks

The output networks take as input the encoded representation (flow features) and are trained to perform one specific task. The force regression task is performed by a three layer multi-layer perceptron (MLP) network that has six outputs, three force components and three torque components. The last layer is not passed through an activation function, but the output is forced to be the actual prediction. The flow prediction and reconstruction networks consist of convolutional and up-sampling layers. The reconstruction network is mirroring the encoding network in terms of number of feature maps per layer, but it does not split into multiple networks as happens in the input. The flow prediction network consists of five convolutional layers. The up-sampling layers are setup to give the output the desired shape. Since the values that the flow prediction and reconstruction networks are predicting are in the range [-1, 1], the activation function of their last layer is set to the hyperbolic tangent.

3.4.3 Activation functions

As activation functions several different options are considered: ReLU, ReLU with batch normalization [154], ELU [53] and PReLU [126] activation functions. In total five different setups are tried: (i) all layers have ReLU activations, (ii) half the layers have ReLU and the other half ReLU with batch normalization, (iii) all layers use ReLU with batch normalization, (iv) all layers use ELU activation functions and (v) all layers use PReLU activation functions.

Unfortunately our implementation of the PReLU activation function was too memory intensive which, in combination with the high dimensionality of the data, was impossible to train. For the rest of the activation functions we used tensorflow's native implementations. The networks using the ELU activation function where not able to converge. In most cases the activations diverged resulting in NaN loss values after approximately half the training steps. Thus the only activation schemes for which training finished successfully are the ReLU, referred to as No Batch Normalization (NBN), ReLU with batch normalization, referred to as Batch Normalization (BN) and half the layers batch normalized (HBN).

3.4.4 Training details

As mentioned in Section 3.3.1 each separate data sample is a 3D volume of 96x64x32 voxels with six channels in each voxel. In order to introduce translation invariance, the common practice is followed where random crops are extracted from the volume and considered as the separate example. When training on the force regression task, the size of each random crop is 80x56x32. When training on the flow prediction the

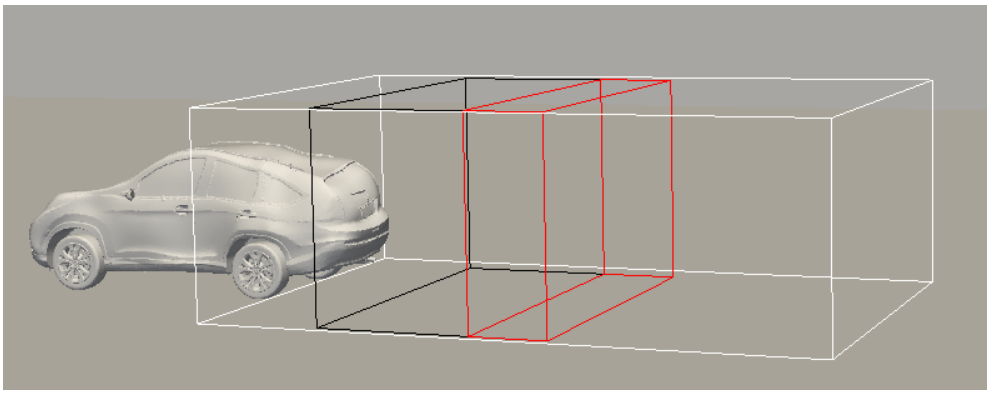


Figure 3.4: Input and ground truth crops for the flow prediction task. The **white** box is the initial example, the **black** box is the input and the **red** is the ground truth.

dimensions of the input crop are 24x56x32, while adjacent 12x56x32 downstream voxels are taken as the output ground truth, as seen in Figure 3.4. An example slice of the input as well as the prediction and the ground truth can be seen in Figure 3.5. The input of the reconstruction task depends on the auxiliary task. In case that both flow prediction and force regression tasks are used, both possible crops (24x56x32 and 80x56x32) are used.

When considering the larger crops used for force regression training or reconstruction, the batch size is set to 4. When the small crop is used as input, the batch size is set to 16. The weight decay of the layers is set to 10^{-4}. We are training using the Adam optimizer [175], with learning rate 10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. In all cases we train for a total of $5 \cdot 10^5$ iterations.

For all training tasks the L_2 distance between the predictions and the ground truth is used as the error function with the addition of the weight decay. In the case of the force regression that is calculated over the forces and torque that the network is predicting while on the flow prediction and reconstruction tasks it is calculated over all voxels and channels.

3.4.5 Multi task training

In the current approach of learning 3D flow field features we are faced with the problem, that the size of the dataset is not sufficient for training a deep convolutional network from scratch. When dealing with such tasks for which there are not enough data to properly train a neural network, it is common practice to use networks pre-trained on larger and more diverse data of the same type (i.e., for an image task

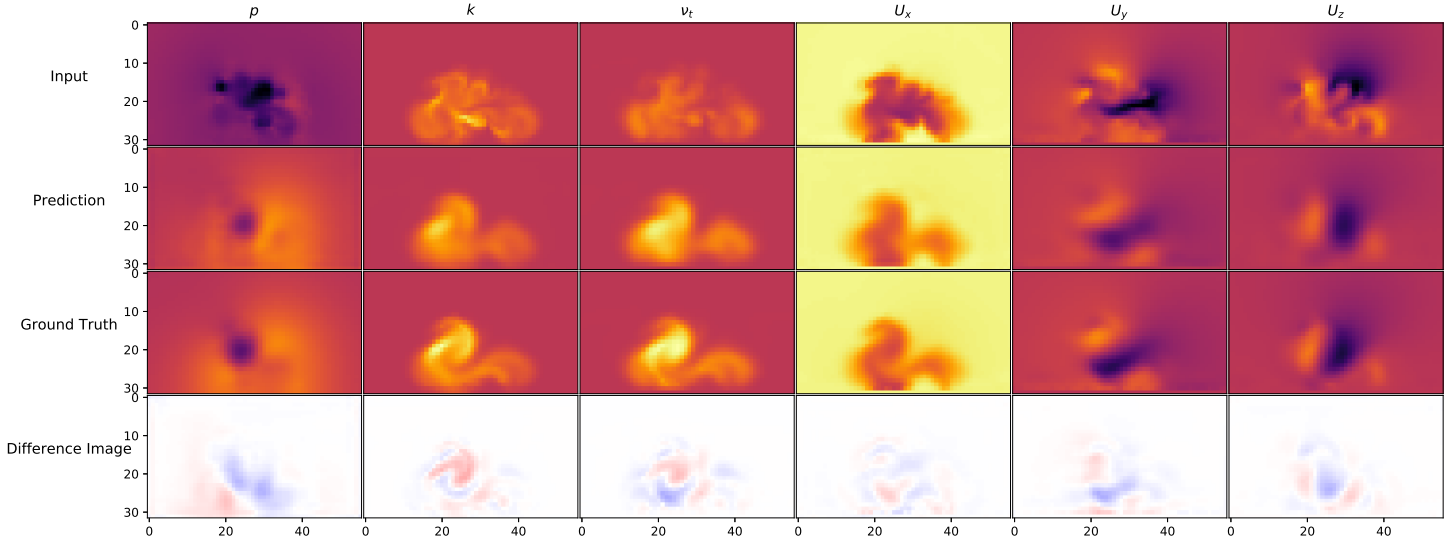


Figure 3.5: Example slice of flow prediction task. The top row shows an example slice from the network input, whilst the second and third an example slice from the network output and ground truth respectively. The last row shows the difference between the flow prediction and the output ground truth slices (Ground Truth minus Prediction). White is zero difference, red is positive and blue is negative.

one would use a network pre-trained on the Imagenet dataset). When a network is trained on a small dataset and on one task, the features learned during its training are tailored for this specific task and more importantly, there is a big chance of overfitting the data. In such a case, most of the layers, if not all, will learn only the patterns important for the specific task which might not capture all interesting information of the input and are not so easily transferable to new tasks and examples. In an attempt to force the networks to learn a more general, yet discriminative, representation we try to train the core CNN on multiple tasks at the same time.

Networks are trained simultaneously on all three tasks we have defined. In one step, the network processes two examples, or mini batches, one which is passed to the flow prediction part and one which is passed to the force prediction part. Both examples are also passed to the reconstruction part. The individual errors, together with the weight decay factor are added to the final global error, as shown in equation 3.1,

$$E_{global} = L_{2,force\,regr.} + L_{2,flow\,pred.} + L_{2,recon._1} + L_{2,recon._2} + wd\ , \tag{3.1}$$

where L_2 denotes the Euclidean distance between the network predictions and the ground truth.

During training, the gradients are computed on the global error and we maintain all parameters of the individual task training, i.e., the batch size for the flow prediction examples is 16, for force regression the batch size is four, etc.

3.5 Experiments

We conducted several numerical experiments and evaluated the results based on the mean absolute error (MAE) between predictions and ground truth. In case of flow prediction and reconstruction the numbers presented are the average over all possible examples (all possible crops of all test set examples), all voxels and all channels. In the case of force prediction, our evaluation measurement is calculated as an averaged relative MAE,

$$Error_{performance} = \frac{1}{6\,N} \sum_j \frac{\sum_i \left| prediction_{i,j} - GT_{i,j} \right|}{max(GT_j) - min(GT_j)} \tag{3.2}$$

where i denotes a test example and j a predicted value. $prediction_{i,j}$ and $GT_{i,j}$ are the prediction value and ground truth of the network for example i and predicted value j, respectively. $max(GT_j)$ and $min(GT_j)$ are the maximum and minimum possible values of the jth target value over all test examples. The above formula first calculates the MAE for all test examples for each target value j. It is then transformed

Table 3.2: Comparison of different activation function schemes in flow prediction task and force regression tasks. The left column shows the performance error on force regression using equation 3.2 whilst the right column shows the results on the flow prediction task using the MAE.

	Force Regression Error	Flow Prediction Error
VC_NBN	**0.02689**	0.00661
VC_HBN	0.03642	0.00446
VC_BN	0.03667	**0.00347**

to relative MAE by dividing with the range of possible values and finally averaged over all six predicted values.

All our experiments are done on modern Nvidia GPUs, namely the *Geforce GTX 980 Ti* and *Geforce Titan X (Maxwell)*. Our network implementations are done using Google's Tensorflow framework [1], with the Python interface.

3.5.1 Activation functions

As mentioned in Section 3.4.3, several activation functions were tested of which only ReLU, referred to as No Batch Normalization (NBN), ReLU with batch normalization, referred to as Batch Normalization (BN) and half the layers batch normalized (HBN) produced useful results.

In order to compare them properly, all other parameters of the networks remain the same for all schemes. In particular, the encoder part of the network follows the VC scheme and has five layers overall, two of which are before fusion and the last three after fusion. The flow prediction and force regression networks are the ones defined in Section 3.4.2. The HBN network uses batch normalization layers during the encoding part whilst simple ReLU in their prediction parts. The results are summarized in Table 3.2.

The results show different trends as to which architecture is beneficial for flow prediction as compared to force regression. Batch normalization seems to hurt the force prediction performance of the networks. This is to be expected, since the forces that are predicted are not normalized. Batch normalization is normalizing the activations to [-1,1]. Thus when predicting the forces the scaling of the results is solely handled by the last few layers in cases of HBN and BN. The performance variation from force regression to flow prediction between the VC_NBN and VC_HBN architectures is much larger than for VC_BN architecture. This suggests that the VC_BN architecture is more

Table 3.3: Comparison of different schemes for handling the input on flow prediction and force regression tasks. The left column shows the performance on force regression using equation 3.2 whilst the right shows the results of flow prediction task using the MAE.

	Force Regression	Flow Prediction
VC	**0.03667**	0.00347
DS	0.03959	**0.00342**
VC^*	0.03859	0.00348
FMS	0.03715	0.00359
VC^{**}	0.03863	0.00495

robust with respect to the specific task and we therefore conclude that quality of the network using batch normalization layers is higher than the simple ReLUs. Thus, for the rest of the experiments only BN networks are used.

3.5.2 Input handling schemes

In Section 3.4.1 three schemes were defined (FMS, DS and VC) for handling the peculiar input data. We trained three networks which only differed by input layers prior to the fusion of feature maps. In all other respects the networks are the same. Two kinds of comparison are performed, one that keeps the number of feature maps similar and one that keeps the number of trainable variables similar.

In our experiments the input networks to be analyzed are two layers deep. The number of feature maps per layer in those networks are shown in Table 3.1.

The DS and VC networks have approximately the same number of feature maps. The VC^* network is defined using the same principles with the VC but with fewer feature maps (Table 3.1), in order to keep the number of trainable parameters similar to the DS network. We also tried to increase the number of layers with the VC^{**} network. It is the same as the VC network, in all respects, except that it has an extra max-pooling and a convolutional layer with 64 feature maps before the prediction networks. Table 3.3 summarizes the performances of the various networks.

The FMS scheme has rather low performance on flow prediction and force regression, strengthening the assumption that the scalar fields have similar low level features. Thus, using different networks for each field does not provide much more information while it increases the overall complexity as well as the number of trainable variables. On the other hand, the DS scheme shows better performance than

Table 3.4: Comparison of different training processes with reconstruction. The first two columns show the results for the force regression and flow prediction tasks. The two right most columns show the results of reconstruction using the MAE, for small (24x56x32) and big (80x56x32) crops. For comparative purposes the performance of the VC network trained on a single task is also shown.

Evaluation Task / Input resolution / Training tasks	Force Regression	Flow Prediction	Reconstruction	
	80x56x32	24x56x32	24x56x32	80x56x32
Flow - Reconstruction	-	0.005799	**0.00288**	**0.00428**
Force - Reconstruction	0.15595	-	0.10367	0.06845
All tasks	0.043197	0.01624	0.0090997	0.01079
Single task	0.03642	0.00347	-	-

the VC network at the flow prediction task, whilst the performance is lower for the force prediction task. The same behavior is seen when DS is compared to the VC^* network.

The VC^{**} has the lowest performance on both tasks, which leads us to the conclusion that increasing the number of layers hurts the performance of the networks (VC^{**} compared to VC). More experimentation is needed in order to properly evaluate the performance of the networks with increasing depth which is left for future work.

3.5.3 Reconstruction

Three training processes are applied for input reconstruction. In the first two, the network is trained on two tasks at the same time, namely reconstruction and flow prediction or force regression, respectively. In the third the network is trained on all three tasks at the same time. For all experiments the VC network is used. When training on two tasks (flow and reconstruction or force and reconstruction) only one input size is used, defined by the prediction task. For better comparison the trained networks are evaluated on both input sizes. When training on all tasks, the network is trained on both input sizes at the same time. During training the total error is computed by adding the errors of the individual tasks. In the case where the network is trained on all three tasks, the overall error is given by equation 3.1. The performance of the trained networks is shown in Table 3.4.

In terms of reconstruction, the best results are achieved when training on reconstruction and flow prediction. Since the network is only trained on the small input

Table 3.5: Number of feature maps per layer for the late fusion network (VC_{late}).

	Scalar Layers	Vector Layers	Common Layers
Layer 1	16	64	-
Layer 2 ($/2$)	32	128	-
Layer 3	32	128	-
Layer 4 ($/2$)	16	64	-
Common Layer ($\#5$)	-	-	64

crops, when reconstructing the big input crops the performance drops significantly (the error is doubled). Still it is much better than any other tested training process. Training on force regression and reconstruction produced the worst results. Training on all tasks also performs worse than the flow - reconstruction training. An interesting observation is that it still produces better results on force regression than the force - reconstruction training.

3.5.4 Fusion stage

The next experiment evaluates how the performance of the network is effected by the stage of the fusion. With that goal in mind a new network is defined, VC_{late} with the same principles as the VC network. The new network fuses the activation maps of the separate input handling networks after 4 convolutional layers and 2 pooling layers, and only has one convolutional layer after the fusion, as shown in Table 3.5.

The performance of the network on the force regression and the flow prediction tasks is summarized in Table 3.6. In both cases the VC_{late} network performs worse than the VC network. As with the total depth of the networks, more experimentation is needed to properly evaluate the effect of the fusion stage on the quality of the networks and is left for future work.

3.5.5 Comparison to a k-NN regressor

As mentioned in Section 3.2.1, the state of the art in flow field feature extraction and representation is VFT. Since VFT is meant for flow field visualization, it is not trivial to directly compare the methods for machine learning applications, since a method which uses VFT of performing the tasks still has to be developed. In order to have a good comparison we perform the force regression task using simple k-NN regression. The motivation behind it lies in the distance measurement. In literature

Table 3.6: Comparison of VC and VC_{late} networks on flow prediction and force regression tasks. The left column shows the performance on force regression using equation 3.2 whilst the right shows the results of flow prediction task using the MAE.

	Force Regression	Flow Prediction
VC	**0.03667**	**0.00347**
VC_{late}	0.04035	0.00366

there are a couple of ways to define flow field similarity [105, 66]. A naive way to measure the similarity between flow fields is the L_2 distance. In the general case this will produce very low quality comparison between flow fields since it is sensitive to any translation and rotation. Nonetheless, given that the flow fields are aligned, the L_2 distance will not diverge much from other similarity, or distance, measurements. In this case aligning the flow fields is a very easy task given the pipeline used to create them. By always using a crop in the same position relative to the car, we ensure that the flow fields are aligned. Moreover, for each comparison we move one of the fields over the other and measure their distance for every position. The smallest number is taken as the final distance. Given all these restrictions we consider the L_2 distance a good approximation of the actual distance of the flows. Using this distance we perform a k-NN regression with three different values of k.

From Table 3.7 it can be seen that the VC method produces significantly better results than the k-NN method, with less than half total error. It should be noted that the networks are trained over crops in random positions, relative to the car, whilst the k-NN method only considers a crop at the same position, resulting in no flexibility.

Table 3.7: Comparison of k-NN method with VC network on force regression task.

Method	Force Regression
$1-NN$	0.08655
$2-NN$	0.080499
$4-NN$	0.082705
VC	**0.03667**

3.6 Conclusion

Motivated by the large amount of data produced by CFD simulations, the need for a machine learning pipeline capable of processing these amounts of data, and the success of CNN in computer vision, we proposed several CNN strategies designed

to handle the output of CFD simulations. We performed a qualitative comparison between the networks and compared them to a k-NN approach. The experiments showed the capabilities of the approach proposed, which outperformed the aforementioned method and generally produced very promising results.

The networks were successfully trained to solve prediction tasks, such as predicting the forces and torque applied on a car, or the prediction of the flow field downstream of the training volume. Additionally, the networks were capable of reconstructing the input flow field in the training volume. A key aspect of the proposed approach is that a network is trained simultaneously on different tasks thereby learning general features of flow fields which are independent of a specific task.

Moreover, a novel dataset was established accompanied with three tasks, as a benchmarking platform for machine learning on steady flow fluids.

This chapter constitutes a first important step in the direction of large scale machine learning on CFD simulation output, which can be beneficial for several engineering applications, meteorology or even for the medical domain.

4 Comparing deep learning and hand crafted features for simulation data

Computational Fluid Dynamics (CFD) simulations are a very important tool for many industrial applications, such as aerodynamic optimization of engineering designs like cars shapes, airplanes parts etc. The output of such simulations is usually very complex and hard to interpret for realistic three-dimensional real-world applications. Automated data analysis methods are warranted but a non-trivial obstacle is given by the very large dimensionality of the data. Deep learning techniques usually require very large datasets to be properly trained on. As we saw in the previous chapter, creating a big scale dataset of CFD simulation requires a lot of time, even for a toy example as the steady flow simulation of the air around a passenger car. On the other hand hand-crafted feature based approaches from computer vision have already defined the low level features and thus might require less data. In this chapter we propose an adaptation of the classical hand crafted features known from computer vision to address the same problem and compare a large variety of descriptors and detectors. Moreover, we adjust the deep learning approaches of the previous chapter to the data used in this chapter and propose a new one. Finally, we compile a large dataset of 2D simulations of the flow field around airfoils with which we tested and compared approaches. Our results show that both deep learning-based methods and hand crafted feature based approaches, are well-capable to accurately describe the content of the CFD simulation output.

4.1 Introduction

Computational Fluid Dynamics (CFD) simulations provide a relatively fast way to evaluate and optimize the performance of different engineering designs. For example, estimations for drag and lift forces of moving objects such as cars or airplanes, as well as tumble motion patterns in internal combustions engines can readily be extracted. The availability of such simulations as well as their high complexity, which renders them difficult to analyze, motivate the development of methods that can analyze them automatically. For example, in the previous chapter we developed and applied deep learning techniques in order to analyze the simulation while taking into account all the information produced by the simulation. Most existing feature extraction techniques, developed for CFD simulations focus on visualization and not machine learning [273, 405]. Deep learning techniques require exhaustive datasets with a very large number of data samples to produce reliable and generalizable performance. On the other hand, CFD simulations are computationally very expensive and large datasets are therefore hard to produce. Additionally, each data sample is substantially bigger, compared to typical input of deep learning pipelines, such as images and videos, due to the high dimensionality and the high spatio-temporal resolutions typical for engineering applications. These contradicting issues set the stage for the challenging research field considered in this work where deep learning methods are applied to CFD simulation data.

Hand crafted features are a well known topic in computer vision. A big variety of features has been proposed along with methods that utilize them for numerous applications. One of the most well known is the SIFT detector and descriptor [226]. After its success, a number of different descriptors and detectors were proposed, in order to improve performance, to be more efficient, or both. A very well known one is the SURF [19] descriptor and detector as well as a number of different binary descriptors, such as ORB [299], BRIEF[41], BRISK [205] and FREAK [7]. In this chapter we utilize those features in the context of CFD simulation output. To the best of our knowledge we are the first to do so.

In recent years, deep learning approaches are outperforming the more traditional approach of feature extraction and description in most applications. Nonetheless, for some applications hand crafted features are better suited, for example when hardware availability is limited. Since running complex CFD simulations takes a very long time, e.g. the simulation of a combustion process in an engine can take more than a month to compute, handcrafted features, which do not rely on a data-heavy training procedure, might still be a viable option. In order to validate that hypothesis we test a number of different detectors and descriptors on their ability to represent different flow fields and to discriminate between them. Moreover, we implement and

test several deep learning approaches, adjusted from Chapter 3, and compare them to the aforementioned hand crafted features. Finally, a new dataset consisting of 16K 2D flow fields of the air around airfoils is compiled and utilized as the benchmark platform.

The rest of the chapter is organized as following. Section 4.2 discusses the relevant work to this paper, Section 4.3 describes the hand crafted features tested, as well as the implementation details. Section 4.4 discusses the deep learning techniques used, Section 4.5 describes the dataset and benchmark developed for the purpose of this comparison. In Section 4.6 we report and comment on our experimental results and finally, in Section 4.7 the conclusions of this study are drawn.

4.2 Related work

Most existing approaches for feature extraction on CFD simulations focus on visualization [273, 405]. Moreover, they focus specifically on the vector field and neglect other information, such as pressure and turbulent viscosity. The only work we are aware of that applied hand crafted features for machine learning on CFD simulation output, collects a number of streamlines, i.e., theoretical particle path in a vector field, with which they described each example [105].

Hand crafted features from computer vision, such as SIFT [226], SURF [19] and ORB [299], have been studied in much detail and applied to a plethora of applications [317, 220]. Nonetheless, there is no work that applies them on CFD simulation output data. There have been many comparisons between the detectors and descriptors [317, 220, 268, 307, 209], but since there have been no applications of them on CFD simulation output, there is also no performance comparison.

In recent years, deep learning has become a mainstream approach for processing a number of different data types [368, 111] and especially data types that show some spatial or temporal relationship within each example, making the convolution a very effective operator. Since each example on the CFD simulation output can show both spatial and temporal relationships, deep learning and more specifically deep convolutional neural networks are an obvious choice for applying machine learning. As such there have been several applications of D-CNNs on CFD data [108, 218, 371, 416]. Most of these, though, focus on either predicting the flow itself [108, 416], and thus substituting the simulation or substituting parts of the simulation with a deep learning predictor [218]. The work in [371] (Chapter 3) is the first that applies deep learning on the output of CFD simulations in order to extract features from it, and thus the most relevant to this chapter. In this work some of the methods introduced in Chapter 3 are adapted to the 2D case. The work of [233] proposed a deep learning architec-

ture for processing vector fields. A large part of the output of the CFD simulation is the velocity vector field. Thus a combination of it with the approaches in Chapter 3 is also tested. Finally, a comparison between the deep learning and traditional approaches, based on hand crafted features, is performed.

4.3 Hand crafted features

Hand crafted feature detection and description is a very well studied subject in computer vision [317, 220, 268, 307, 209]. These features have been utilized for many applications, such as image classification [260], image retrieval [341], object detection [100] and scene semantic segmentation. Depending on the application the utilization strategy may vary. For example, when used for object detection separate descriptors are matched and aligned [226]. In applications such as image classification or image retrieval, it is more useful to create a global description of the image. A common approach towards a global image description is to create the so called Bag of Words (BoW) model [341]. For the purpose of this comparison a global description of the flow is created aiming to predict the drag and lift forces applied on an airfoil.

In the literature a multitude of approaches exists, both for detecting and describing visual features. Moreover there have been numerous studies that compare them for many applications. In order to limit the search the most popular and best performing detectors and descriptors are chosen, according to the studies found [317, 220, 268, 307, 209]. CFD simulation output is very peculiar in comparison to natural imagery. Each example consists of multiple data modalities and the values of each modality usually change smoothly. Thus, detectors created for natural images with many corners and abrupt changes do not detect many points. This raises some issues since some detector-descriptor combinations completely fail to find any features for some examples. In these cases it would be completely impossible to create a global description. Thus, these detectors are neglected from this study. Moreover, extracting features from a regular grid instead of detected keypoints is considered, as this strategy has proven to be superior in several applications that require global image description [260, 398]. The combinations tested, as well as whether they were successful in producing a global description for all examples are given in Tab. 4.1. Besides these detectors and descriptors the CenSure [6], Harris-Laplace [246] detectors as well as the BRIEF [41] and FREAK [7] descriptors were tested, but we did not manage to get comparable results with any combination and thus they are not included this comparison. The focus of this chapter has moved to the 2D case since there exist big libraries with many implementations of hand crafted features in comparison to the 3D case, making this an easier benchmark to see if these methods are suited for

Table 4.1: Combination of detectors and descriptors tested. ”SD” signifies that the combination is used with the single dictionary approach, ”MD” signifies that the combination is used with the multiple dictionaries approach, ”x” signifies that the combination didn’t manage to produce results and ”-” signifies that the combination was not tested.

	SIFT [226]	SURF [19]	ORB [299]	AGAST [232]
SIFT [226]	x,MD	-	-	SD,MD
SURF [19]	-	SD,MD	-	SD,MD
ORB [299]	-	-	SD,MD	SD,MD
BRISK [205]	-	-	-	SD,x

describing CFD simulation output.

One of the peculiarities of CFD data, compared with traditional imagery, is the number of modalities. Overall there are five modalities, i.e. two for the velocity vector field and three for pressure, turbulent viscosity and a viscosity related field from the turbulence modeling, i.e. $\tilde{\nu}$ from SpalartAllmaras RAS model [264]. According to this study [187, 334, 368] (see Chapter 2), the most common approach for combining information from multiple modalities is to process each modality separately, e.g. create a dictionary for each modality and concatenate the per-modality representation to construct the final representation [187] or perform classification based on each modality and then average the results [334]. In this study a slightly different approach is also explored. Features from all modalities are extracted and the common features are filtered out by discarding the ones that have intersection over union ratio above 0.9. Then, for each detected feature, a description for each modality is created and concatenated to produce the final description. Finally, one common dictionary is constructed from the concatenated features. In order to differentiate between the different strategies, the common approach is called ”Multiple Dictionaries” (MD) and the second one ”Single Dictionary” (SD). Dense feature extraction is also tested, which is denoted as ”DE”.

After acquiring the global description of each example, a Random Forest (RF) regressor is trained to predict the drag and lift forces. We utilize the OpenCV [32] implementations of the detectors and descriptors from the Python API. The dictionaries for real valued descriptors are built using an approximate K-Means clustering, whilst for binary descriptors the K-Majority algorithm is used with Hamming distance as the distance metric. The approximation of K-Means, pre-computes all the pairwise distances of data points and sets as cluster center the closest point to the actual cluster center. Thus, the distances of all points to all centers do not need to be computed in

each step since they are already available. We utilize the scikit-learn package's [272] implementation of the RF with default parameters, while the clustering algorithms are implemented from scratch, using the Python-numpy library.

4.4 Deep learning approaches

4.4.1 Methodology

As mentioned in Section 4.2, deep learning has already been applied to the same subject. In Chapter 3 three different strategies were proposed and tested for processing the multi-modal data produced by the simulations. Based on the results, the Velocity Coherent (VC) and Direction Specific (DS) approaches were picked and adjusted, for the 2D case, as the baseline models. Specifically, both approaches use one network with one input channel to process the scalar fields (applied to each one separately). The VC approach utilizes a network with two input channels to process the velocity vector field whilst the DS uses two networks with one input channel, one for each direction of the velocity. These input processing networks are comprised by four convolutional layers. Their output is concatenated and then passed to another CNN for further processing. The structure ends with three fully connected layers the last of which is performing the drag and lift regression. Besides reducing the dimensionality to two, skip connections are also added every two layers, since it proved to increase the performance of the networks.

Moreover, as mentioned in Section 4.2, a third approach is defined based on the RotEqNet, proposed by Marcos et. al.[233]. In their work, they propose an architecture tailored for vectorized data. This approach is utilized by using the RotEqNet as a substitute of the velocity processing network of the VC and DS approaches. The output of each layer is a vector field for each filter, resulting in double number of channels compared to a plain CNN with the same number of filters. Moreover, since the input consists of vector field feature maps, the number of trainable parameters is given by: $F_h \cdot F_w \cdot 2 \cdot c_i \cdot c_o$, where F_h, F_w are the filter height and width respectively and c_i, c_o the input and output number of channels. The three different input pipelines are shown in Fig. 4.1, top. After the processing of each modality, the feature maps are concatenated and further processed by a common CNN of five layers. Finally, three fully connected layers perform the regression. Notice that in the case of RotEqNet the number of filters is halved. This is a consequence of the vectorization of the activations where for each filter there are two output channels, the magnitude and the angle of the vector.

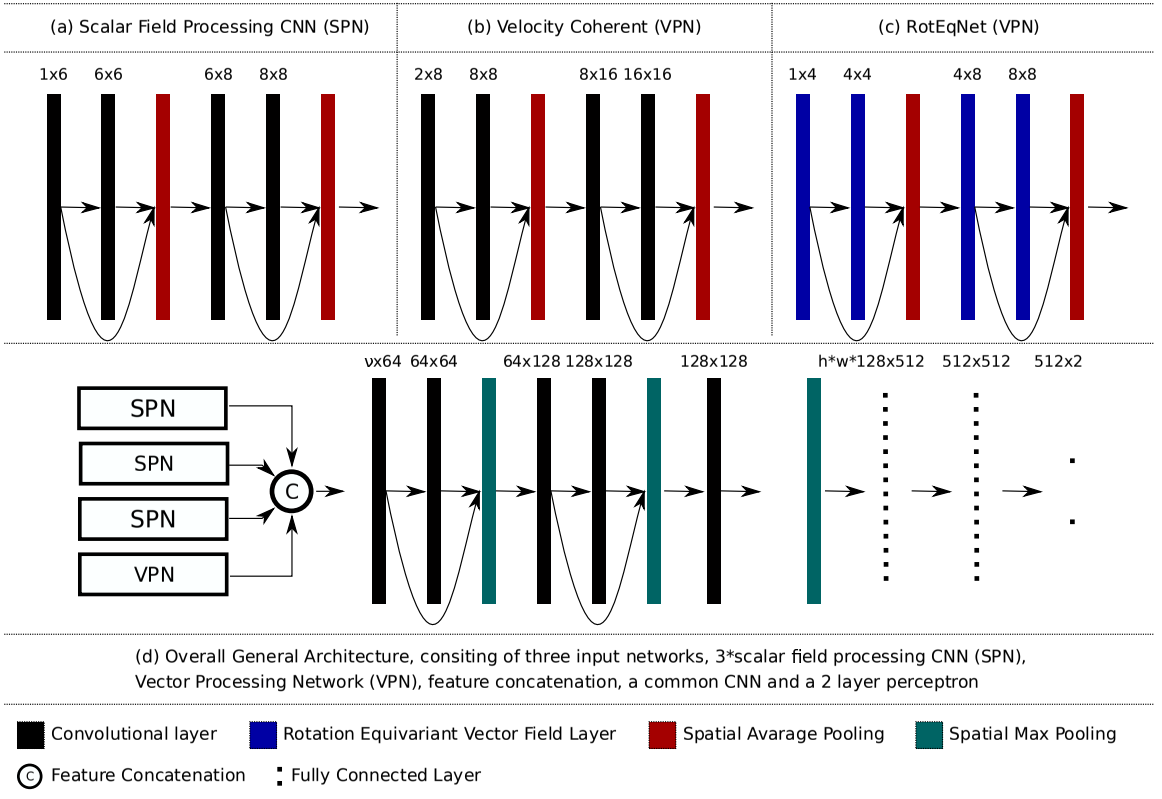


Figure 4.1: CNN architecture. (a) Is the network processing the scalar fields (SPN), shared for all scalar fields. (b) The Vector Processing Network (VPN) for the VC strategy. (c) The VPN network for the RotEqNet strategy. (d) The complete architecture. The VPN is either (b), (c) or in the case of DS 2*SPN. The numbers above each layer are the input and output filters respectively. h,w are the height and width of the input feature maps.

4.4.2 Implementation and training details

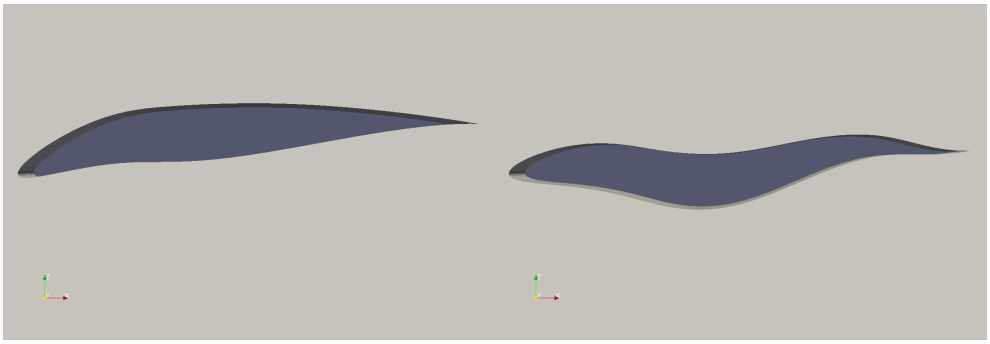
For all strategies, the input processing networks are four layers deep and the network applied after the feature concatenation consists of five convolutional layers. Spatial average pooling operations are performed every two convolutional layers in the input processing networks, whilst spatial max-pooling is performed every two convolutional layers in the network after the feature concatenation. All convolutional kernels have spatial dimensions 5×5 and are followed by batch normalization [154] and a leaky ReLU activation function [230] where $\alpha = 0.1$. The number of nodes per layer is given in Fig. 4.1. The prediction network consists of a max-pooling operation, followed by three fully connected layers, with 512, 512 and 2 nodes respectively. The third fully connected layer is tasked to predict the drag and lift forces. All networks are trained with Adam optimizer [175], with the default parameters and a batch size of 200, for 36K iterations. The implementation is done using Tensorflow 1.13.1 [1] and all experiments ran on NVIDIA GTX 1080Ti graphics cards.

4.5 Dataset

The aim of this chapter is to compare the performance of deep learning based and more traditional hand crafted feature based approaches, for mining CFD simulation output. Due to the much larger variety of hand crafted features for 2D imagery, as well as the high computational demand of deep learning methods on high dimensional data, the 2D simulation domain is picked as the setting for this benchmark, since it exhibits many of the characteristics that exist in the 3D domain, such as a large number of modalities and a velocity vector field, that satisfy, to a certain extent, the Navier-stokes equations.

To the best of our knowledge there is no 2D dataset that can be utilized as the benchmark. Consequently, a new dataset is proposed. The focus is on the standard airfoil example. In order to create a large dataset with as big variety as possible, a similar approach to [371] (Chapter 3) is followed. First, a baseline airfoil shape (Fig. 4.2) is defined and random deformations are applied to it. Then, given intake air from the left of the simulation domain, the air around the airfoil is simulated using Reynolds-Average Simulator (RAS) implemented in OpenFOAM-v5 [264]. The output of the simulation consists of the velocity vector field, the pressure field, turbulent viscosity, and the drag and lift forces applied on the airfoil. Overall 2K shapes are created and simulations are done for 8 different angles of attack per shape, resulting in 16K simulations. 15K are chosen for training at random and the remaining 1K are using as a test set.

The aim of this work is to perform data mining and pattern recognition on the flow



(a) (b)

Figure 4.2: Example airfoil shapes. (a) Baseline model. (b) Randomly deformed shape.

fields. Thus, we want to discard any information that relates to the shape of the airfoil. Consequently, a window behind the airfoil is cropped (see Fig. 4.3) and the flow field in this region is extracted for further processing. The simulation is performed on an unstructured mesh. In order to bring the data to a format that hand crafted features and CNNs can be easily applied to, the values are interpolated on a regular grid with resolution 192×128. Finally, to evaluate whether the defined methods are able to extract meaningful information, they aim is to predict the drag and lift forces applied on the airfoil. The dataset is publicly available on Zenodo [1].

4.6 Experiments

In order to asses whether the features are capable of encoding relevant information, they are used to predict drag and lift force coefficients of each airfoil. We then compare the predicted values with the actual simulation results and quantify the performance using the root-mean squared error (RMSE) as metric.

The first experiment aims at identifying the optimal number of clusters which are used to construct the dictionaries. Dictionary sizes of 512, 1024 and 2048 are tested for the SD approaches. The results are summarized in Table 4.2.

For the MD approach, the number of clusters for each modality needs to be defined. Most of the detectors detected extremely low number of points for some of the modalities, e.g. pressure, and thus large dictionary sizes are infeasible. As a result the dictionary sizes for the two velocity directions as well as pressure are set to 32 and

[1] https://zenodo.org/record/4077323#.X4QeI3VfhhE

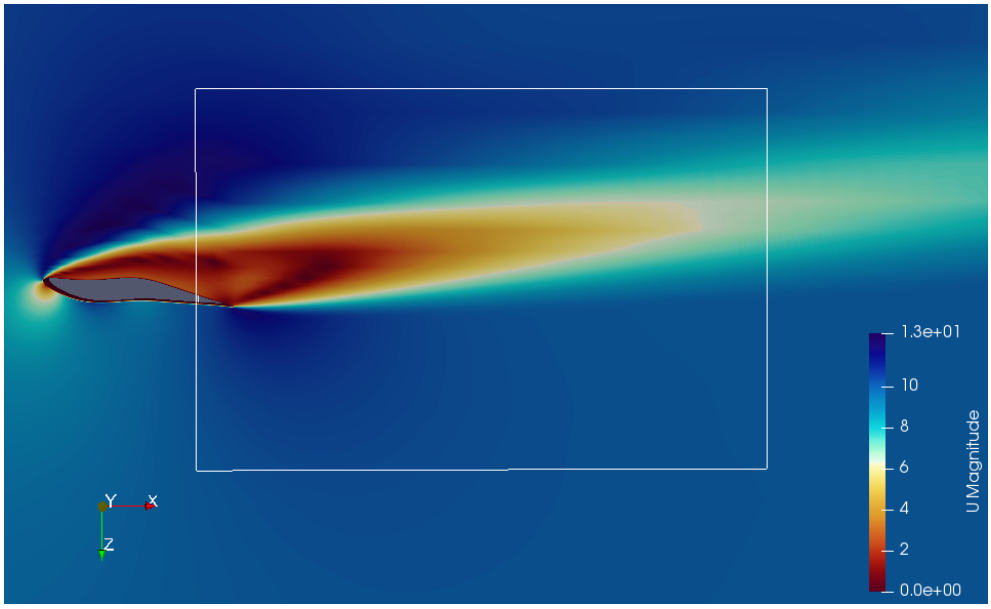


Figure 4.3: Example simulation. The square behind the airfoil is the window used as an example for our pipeline.

Table 4.2: Regression performance of the SD approach with varying dictionary sizes, measured by RMSE. The best performing method per column is highlighted with italics, and the overall best for each evaluation measure (Drag or Lift) is highlighted with bold.

Detector	Descriptor	Drag (*1e-3)			Lift (*1e-2)		
		512	1024	2048	512	1024	2048
SURF	SURF	10.17	9.37	9.57	5.6	5.59	6.02
ORB	ORB	*7.88*	8.22	7.77	*4.74*	*4.1*	**3.87**
AGAST	SIFT	8	*7.83*	**7.51**	5.25	4.72	4.88
AGAST	SURF	10.06	9.89	10.49	7.73	7.39	8.64
AGAST	ORB	7.93	7.86	7.84	5.8	5.56	5.69
AGAST	BRISK	8.67	8.42	8.95	7.95	7.42	7.32

Table 4.3: Regression performance of the MD approach with varying dictionary sizes, measured by RMSE. The best performing method per column is highlighted with italics, and the overall best for each evaluation measure (Drag or Lift) is highlighted with bold.

Detector	Descriptor	Drag (*1e-3)		Lift (*1e-2)	
		32-512	64-1024	32-512	64-1024
SIFT	SIFT	19.55	16.59	15.16	14.11
SURF	SURF	11.67	9.52	7.97	9.56
ORB	ORB	8.52	8.84	*4.6*	**4.52**
AGAST	SIFT	**7.22**	*7.68*	6.68	7.3
AGAST	SURF	9.99	9.92	11.98	10.24
AGAST	ORB	9.76	9.26	7.15	6.93

64. For the rest of the modalities dictionary sizes of 512 and 1024 are tested. The results are given in Table 4.3.

Looking at Tables 4.2 and 4.3 one can identify a few trends. Overall, the ORB-ORB combination produces the best performance for predicting the lift force while being a close second on the lift prediction performance. The AGAST-SIFT combination similarly has the highest performance in predicting drag forces and a close second on predicting lift forces. Regarding the modality aggregation strategy, for most detector-descriptor combinations, the SD approach outperformed the MD approach, with the only exceptions being the AGAST-SIFT on drag prediction and the AGAST-ORB on lift prediction.

For dense sampling we extract features in 4 different scales, i.e., {12, 16, 24, 32} pixels. The step size for each scale is the same number of pixels as the size of the scale. Regarding modality aggregation both approaches are evaluated, i.e., SD and MD. Due to time limitations only the, according to our previous experiments, best performing descriptors were used, namely ORB and SIFT. For the MD approach dictionary sizes of {256, 512} were tested, for each modality, resulting in a 1280 and 2560 global description sizes, respectively. For the SD approach the same dictionary sizes with the detector approach were used. The results can be seen in Tables 4.4 and 4.5.

The results show that dense sampling outperforms the detection mechanism in terms of global description in most cases, similar to what is found for other computer vision tasks. This is not the case only for the ORB descriptor in the context of drag prediction. Moreover, in contrast to the use of detectors, the MD approach performs better than the SD approach. The dense description approach increased the quality of the results for the SIFT by a significant margin, rendering it the highest performing

Table 4.4: Regression performance, measured by RMSE, of the DE-SD approach with varying dictionary sizes and modality aggregation strategies.

Descriptor	Drag ($\cdot 10^{-3}$)			Lift ($\cdot 10^{-2}$)		
	512	1024	2048	512	1024	2048
ORB	10.46	9.22	9.26	3.34	3.58	3.94
SIFT	*7.03*	*8.08*	**6.59**	*2.68*	*3.11*	**2.65**

Table 4.5: Regression performance, measured by RMSE, of the DE-MD approach with varying dictionary sizes and modality aggregation strategies.

Descriptor	Drag ($\cdot 10^{-3}$)		Lift ($\cdot 10^{-2}$)	
	256	512	256	512
ORB	21.14	38.78	10.74	16.63
SIFT	**6.28**	*9.09*	**2.33**	*2.87*

method for both tasks. In contrast, the performance increase is not found with the ORB descriptor, where the performance even dropped by a significant margin.

Table 4.6 shows the performance achieved by the deep learning approaches. Comparing Tables 4.5 and 4.6, deep learning approaches outperform hand crafted feature approaches in all benchmarks. Particularly, all deep learning approaches perform better than the DE-SIFT-MD, i.e., the best hand crafted feature based approach, for both drag and lift prediction. For a more thorough comparison relevant in practice, it needs to be stated that hand crafted feature approaches have less computational complexity. The DE-SIFT-MD approach with dictionary sizes of 256 per modality takes 7.5k seconds to extract features from the training set, cluster them and train the random

Table 4.6: Regression performance, measured by RMSE, of the deep learning approaches.

Approach	Drag ($\cdot 10^{-3}$)	Lift ($\cdot 10^{-2}$)
VC	5.32	2.18
DS	5.53	2.25
RotEqNet	5.22	2.2
VC-RF	**2.83**	**0.92**

Table 4.7: Regression performance, measured by R^2, of the three best performing approaches. The best performing method is highlighted with bold.

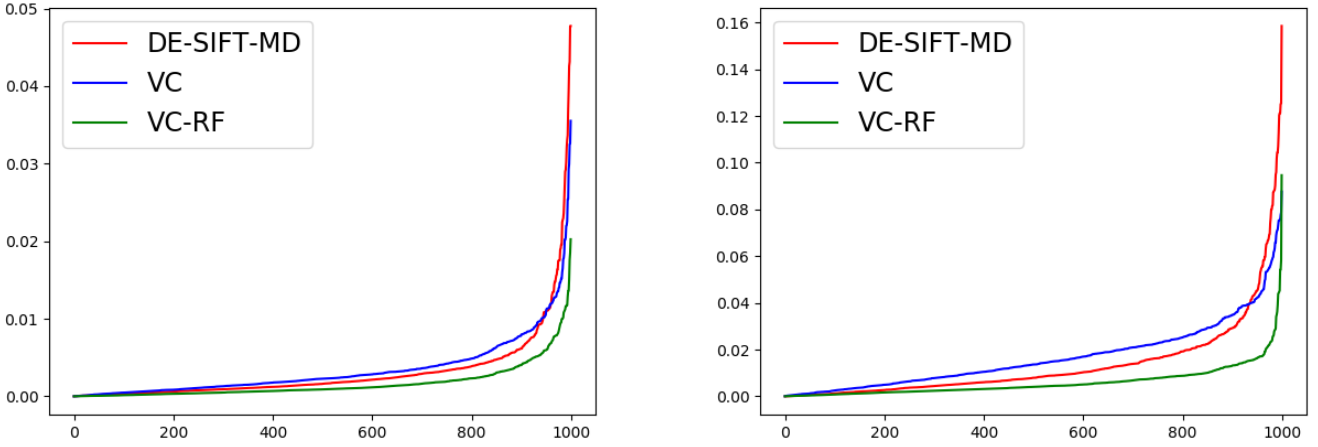
Approach	Drag	Lift
DE-SIFT-MD	0.965	0.987
VC	0.917	0.949
VC-RF	**0.981**	**0.994**

forest on two Intel(R) Xeon(R) CPU E5-2699. At the same time the VC approach takes around 11.1K seconds to be trained on the same machine running on an NVIDIA GTX 1080Ti. Applying these approaches to large scale CFD simulations, with 4 physical dimensions and multiple modalities, where the data complexity is much larger, the high computational demand of deep learning approaches might render them infeasible, making the hand crafted feature based approaches an appealing alternative.

For a further comparison between the features produced by deep learning and the hand crafted features, we extract the output of the last fully connected layer of our best network (VC) and train a RF regressor. The result is given in the last row of Table 4.6, depicted as VC-RF. It is apparent that the use of CNNs as feature extractor and a random forest regressor to perform the given task achieves much higher performance than the equivalent neural network solution, or the use of hand crafted feature based description with an RF regressor.

In order to get a more informative image of the performance of the evaluated methods, the R^2 values for the three top performing methods, i.e., the DE-SIFT-MD, the VC and the VC-RF are also calculated and shown in Table 4.7. Moreover, the sorted absolute errors, per test example, are plotted for both lift and drag forces in Figure 4.4. There is a qualitative difference between the performance of the VC and the DE-SIFT-MD approaches. The VC approach has much lower R^2 values whilst it achieves lower RMSE. This can be explained by the two figures. Although overall the DE-SIFT-MD approach has lower absolute errors, the error of the extreme cases become much more severe than in the case of the VC approach. Depending on the behavior one needs from a system, a different approach would be preferable. Finally, in both drag and lift prediction, the VC-RF approach manages to produce much better results in all measures tested, RMSE, R^2 as well as overall better absolute error curves.

One of the motivations of this work was to assess the usability of hand-crafted features for CFD simulation output and compare them to deep learning solutions for situations with small training datasets. In order to evaluate this, the best performing configurations, from both deep learning and hand-crafted features are trained with



(a) (b)

Figure 4.4: Sorted absolute drag (a) and lift (b) regression error per test example. The Y-axis is the absolute error, whilst the X-axis is the index of the test examples (after sorting based on the absolute error).

Table 4.8: Drag regression performance, measured by RMSE, of the DE-SIFT-MD and VC with varying training set sizes.

	1K	2K	4K	8K	14K
VC	**11.7**	**8.39**	**6.81**	**6.05**	**5.32**
DE-SIFT-MD	21.26	19.26	14.56	12.19	6.27

Table 4.9: Lift regression performance, measured by RMSE, of the DE-SIFT-MD and VC with varying training set sizes.

	1K	2K	4K	8K	14K
VC	**4**	**3.05**	**2.71**	**2.3**	**2.18**
DE-SIFT-MD	10.3	7.36	6.16	5.19	2.33

varying training set sizes and the same validation and test sets. The training set sizes are {1K, 2K, 4K, 8K, 14K}. The performance for drag and lift forces are shown in Tables 4.8 and 4.9 respectively. It is apparent, that even with small training set sizes the deep learning approaches outperform the hand-crafted features. Surprisingly, the difference between the performances becomes larger as the training set size reduces, showing that deep learning approaches are more capable of maintaining relatively good performance even with small training set sizes.

4.7 Conclusion

In this chapter the performance of hand crafted features such as SIFT [226], SURF [19] and ORB [299] is evaluated on their ability to efficiently describe CFD simulation output. Furthermore, they are compared to a number of deep learning approaches. CFD simulation output can be very complex and large, compared to standard computer vision application examples, such as 2D and 3D imagery. Moreover, creating these examples can even take months making the generation of enough examples for deep learning approaches infeasible. On the other hand, a complete working pipeline based on hand crafted features might not require as many examples, since the basic features are predefined and not learned from the data, an assumption that was, surprisingly, falsified by the experimental results. Due to the large variety of detectors and descriptors that exist in 2D, as well as the lower computational complexity of 2D CFD simulations compared to 3D & 4D, a dataset of 2D CFD simulations is created

and used as our benchmark platform.

Overall 4 detectors and 6 descriptors are tested as well as dense sampling. Moreover, two different approaches for combining different data modalities are evaluated, namely a multiple dictionary (MD) approach and a single dictionary (SD) approach. Their difference lies in the concatenation step. Specifically, in the SD approach we concatenate the low level features, before the dictionary construction, whilst in the MD approach we concatenate the information after we create the dictionaries. According to our experiments, for the drag and lift prediction tasks, dense sampling combined with SIFT descriptor produces the best results of all hand crafted feature based approaches by a large margin. Moreover, we identify a contradiction. In most cases, the SD approach outperforms the MD approach, but with dense sampling and SIFT features, which is the best combination tested, we see the opposite behavior.

We also implemented and tested a number of deep learning approaches. They are adapted approaches from data mining on 3D simulation data [371] (Chapter 3). We also combine them with the work of Marcos et al. [233], which is designed for vector fields. Deep learning methods outperformed the hand crafted feature based approaches in the benchmarks tested. Moreover, our experiments showed that using the neural networks as feature extractors whilst a random forest regressor to perform the task produces higher performance than the pure deep CNN counterpart. Comparing different regression characteristics, like R^2 and the absolute error curves, we see a qualitative difference between the DE-SIFT-MD and the VC approaches. The DE-SIFT-MD produces more accurate predictions on most test instances but also gets much higher maximum error than the VC approach. Moreover, the R^2 performance of the DE-SIFT-MD is higher than that of the VC, even thought the the RMSE of VC is lower than that of DE-SIFT-MD. Thus, depending on the requirements of an application, different method would be preferable.

We compared the deep learning methods to the hand crafted features in terms of efficiency. As expected, the hand crafted feature approach is more efficient as it managed to construct the global description of all examples as well as train the RF regressor in 67% of the time it took to train the best performing CNN.

The 2D airfoil example is considered a very important benchmark for CFD simulations. Nonetheless, it is much simpler than many industrial application CFD simulations. As such, we are able to get a much higher number of examples for this dataset. In order to get an intuition on how these methods would generalize to the more realistic case with much smaller datasets, we perform an experiment and train our models on much smaller training set sizes while testing on the same test set. Our results show that deep learning approaches are much more capable of getting a high performance with the drop of the training set size, than the hand-crafted feature based approaches. We speculate that the reason behind this is the necessity of a large

number of descriptors required to build generalizable enough dictionaries.

5

Clifford convolution inspired orientation equivariant CNNs

Convolutional Neural Networks (CNNs) such as VGG [338], ResNet [127] and DenseNet [146], are very powerful models for processing image and sensor data. However, a notable drawback for their application in many domains is their sensitivity to rotations of the input data. In order to alleviate this issue several methods have been proposed. Most of these implement orientation pooling through max-pooling as a core aspect to achieve rotational invariance or equivariance. However, such operations are very computationally intensive since all the activations have to be computed for many different orientations. In this chapter, we take advantage of vector field representation to repeatably calculate the angle of the kernel with the input pattern. This is achieved by generalizing the basic convolutional kernels to realize Clifford convolutions. With this architecture, rotation angles can be calculated without computing activations for all orientations. Since the rotation angle is computed from the Clifford kernels and the input alone, this information can be utilized in the learning process through backpropagation. The proposed method is evaluated on the rotated MNIST [200, 197] on classification performance (rotation invariance) and orientation prediction (rotation equivariance). We show that this method improves the equivariance property of networks over max-pooling, preserving or even improving classification performance while having lower computational cost.

5.1 Introduction

Convolutional Neural Networks (CNNs) are very powerful models that can extract high level information from their input, such as the contents of an image [302]. State of the art networks, such as AlexNet [184], NiN [216], VGG nets [338], ResNet [127], and DenseNet [146], achieve this by combining a hierarchy of feature extractors, i.e., convolutional layers, to complex and deep architectures. A limitation of the convolution operation is that it is not rotation equivariant, and thus the output signal changes if input signal is rotated. As a result, if a network needs to identify the same pattern at different orientations it would require multiple kernels.

There are multiple potential benefits in a rotational invariant or equivariant convolution operator as proposed in this work. The number of trainable parameters might be reduced [409, 408] since no additional kernels are required for detecting rotated patterns. Due to the resulting models being simpler, they will also be less prone to overfitting and have lower memory footprints. Additionally, in some cases it is essential to extract the orientation of the detected patterns, such as for aerial imagery [233] and robotic grasp detection [204].

From the research literature on rotational equivariance, four main groups can be identified. One is to rotate the input in multiple orientations and then pool the top most activations from the different orientations [189]. The second rotates the kernels from all layers and forwards the activations of different orientations throughout the network, and then pools the activations from different orientations at the latest layer [449]. The third group of methods is similar to the second group, with the difference that orientation pooling happens at all layers and all positions of the feature maps [233]. Thus the orientation information is still propagated, whilst the number of trainable parameters and the information flow does not explode. The last method is inspired by the group convolutional networks [55] and convolves in the orientation dimension as well. This combined with an initial input of convolutions through all rotations and orientation pooling in the last layer results in one of the highest performing methods to date. Nonetheless, since all equivariant layers are convolving in the orientation dimension as well, it is a very computationally intensive approach. All four methods increase the memory footprint of the networks and incur additional computational cost.

In this work, we take advantage of vector field representation to calculate the angle between two vector fields, i.e., the kernel and input signal. This is achieved by using convolutional kernels which utilize operations from Clifford Algebra [316, 75]. This enables the detection of rotated patterns without significant memory and computational overhead, but also allows for the extraction of the rotation angles of the detected patterns. The orientation angles are calculated from the kernels and the

layer input alone, and thus can contribute to the learning process through back-propagation. Even though the extractable rotation angles are in principle continuous, we employ rotation quantization in order to reduce complexity. However, we reconstruct (approximate) continuous angles by employing a correction mechanism. Our experiments show that our method increases the rotation equivariance of the models over max-pooling based networks. At the same time it manages to maintain and in some cases improve the classification performance while being less memory and computationally intensive.

5.2 Related work

An early approach to rotation equivariant CNNs is the Spatial Transform Networks (STN) [157]. A specific module is proposed which transforms the network input before it is passed to the prediction network. Although this method does not suffer from the extra memory footprint and computational demand of computing feature maps for all orientations, the invariance is not intrinsic but rather learned by the network, and thus restricted by the training data. TI-pooling [189] rotates the input multiple times and processes it with Siamese networks. After the first fully connected (FC) layer, they perform orientation pooling and thus the next FC layer takes as input rotation equivariant features.

Another group of works rotate the feature maps in every layer and propagate all the information to the next layers. At a final stage the orientation and spatial information are pooled together. The Orientation Response Networks (ORN) [449] additionally define the orientation alignment layer, a SIFT like alignment applied before the orientation pooling. The same principle is also applied in the RI-LBC [443].

The Rotation Equivariant Networks (RotEqNet) [233] calculate the response at each layer and position for a number of rotated kernels, which represent vector fields. Then, the most dominant orientation per location and output channel is computed via max-pooling, and the result is represented by a new vector field with the response of the convolution as the magnitude at the respective angle. The approach presented here is similar in the sense that the activations are also vector fields, and that different orientations in all positions and channels throughout the network are accounted for.

Another approach to rotation equivariance is the group convolution [55], which define operation groups, e.g. rotation, and perform convolution on all the operations in the group. The result are filters that have equivariant response to all operations in the group. The main difference with the above is that all rotations are being processed and the equivalent information propagated in the network instead of performing orientation pooling and propagating only a fraction of the information. This approach

is utilized by the steerable filter CNNs defined by [409]. These networks make use of the group convolutions in combination with steerable filters to produce rotation equivariant nets. At the last convolution layer they perform orientation pooling and finally three fully connected layers perform the classification. The work of Weiler et al. [408] created a more general formulation regarding and re-implement existing architectures, such as the steerable CNNs [409] and propose new architectures based on their finding and existing operators. With a combination of cyclic equivariant group with a cyclic and flip group (only in the first few layers) they achieve state of the art performance in the MNIST-rot dataset. It should be noted that most of the networks in this cluster of methods, i.e., utilizing group convolutions, only perform orientation pooling before the fully connected layers and thus process much more information than most methods.

The aforementioned methods perform orientation pooling in various variants, where to the best of our knowledge the best performing methods incorporate max-pooling for orientation pooling. In this paper we propose a method which specifically tackles the orientation estimation and therefore we compare to max-pooling as the state of the art orientation pooling mechanism in our experiments. Additional techniques which would influence the performance, such as the steerable filters [409], could be combined with our approach. In order to evaluate our claim we re-implement the RotEqNet [233], as well as group convolutional CNNs with steerable filters [409, 408] using the proposed Clifford convolution inspired layer for orientation pooling.

Clifford convolution has been previously used for pattern recognition on 3D vector fields [75]. Patterns, such as vortices, were identified using predefined kernels, that represent those patterns, i.e., hand crafted feature extraction was done. As far as the authors know, this is the first time Clifford algebra has been used in the area of deep learning where multiple layers of filters (so called features) are learned.

5.3 Clifford convolutions and calculation of rotation angles

One central aspect of Clifford Algebras is the definition of general products between vectors and their geometric interpretation. Most relevant for this work is the ability to estimate the angle between two vector fields by performing two convolutions [75],

$$\tan \phi_{i,j} = \frac{conv_2(i,j)}{conv_0(i,j)} \;\Rightarrow\; \phi_{i,j} = \arctan \frac{conv_2(i,j)}{conv_0(i,j)}, \tag{5.1}$$

where

$$conv_0(i',j') = \sum_{c_i}^{c_{in}} \sum_{i}^{F_h} \sum_{j}^{F_w} \vec{W}_{i,j,c_i} \cdot \vec{O}_{i'-\frac{F_h}{2}+i,j'-\frac{F_w}{2}+j,c_i}$$

$$conv_2(i',j') = \sum_{c_i}^{c_{in}} \sum_{i}^{F_h} \sum_{j}^{F_w} (W_{0,i,j,c_i} \cdot O_{1,i'-\frac{F_h}{2}+i,j'-\frac{F_w}{2}+j,c_i} - W_{1,i',j',c_i} \cdot O_{0,i'-\frac{F_h}{2}+i,j'-\frac{F_w}{2}+j,c_i}) \tag{5.2}$$

where $\vec{O}_{i',j',c_i}, \vec{W}_{i,j,c_i} \in R^2$ are multi-channel 2D signal and kernel vector fields with discrete coordinates $(i',j') \in [(0,0),(O_h,O_w)]$ and $(i,j) \in [(0,0),(F_h,F_w)]$, where O_h, O_w, F_h, F_w are the image (O) and filter (F) height and width respectively:

$$\vec{O}_{i',j',c_i} = (O_{0,i',j',c_i} \ , \ O_{1,i',j',c_i}) \tag{5.3}$$

The original formulation considers continuous vector fields and integrations instead of summations [75]. Depending on the instantiation of $\vec{O}$ and $\vec{W}$, as well as the angle between them, the computation might be inaccurate. In order minimize this error, we calculate the angles for four orientations of the kernels for successive angles of $\frac{\pi}{2}$. The kernel orientation that produced the smallest angle is considered the most accurate, and thus we calculate the final angle from this choice. Experimentally we found that choosing the orientation that produced the highest value of $conv_0$ results in more accurate networks and thus it is chosen for all our experiments.

To illustrate the angle measurement, we construct a random 50-channel 2D vector field $\vec{O}$ on a 3×3 grid. Each magnitude is randomly drawn from a uniform distribution in $[0,1)$ and each angle in $[0,2\pi)$. To construct example filters, we rotate $\vec{O}$ randomly and add Gaussian noise, $\vec{W}_{ij} = rotate_{i'j' \to ij}(\vec{O}_{i'j'},\phi) + \vec{G}_{ij}$. Both the grid positions and the 2D vectors are rotated by a random angle ϕ. We denote the rotated kernel $\vec{W}$ by angle ϕ as $\vec{W}_\phi$.

We generate $10,000$ such random vector fields and filters and calculate the angle of the original fields ($\vec{O}$) to the rotated fields ($\vec{W}$) with and without the addition of noise. Figure 5.1 shows the distribution of the difference between the calculated angle and the real angle used for the rotation. Even with the addition of noise, the average absolute difference is less than 10^{-2} radians.

5.4 Layer construction

5.4.1 Forward pass computations

The rotation equivariant processing layer proposed in this work can be decomposed into five steps $[C_1, C_5]$ which is schematically shown in Figure 5.2. For the sake of

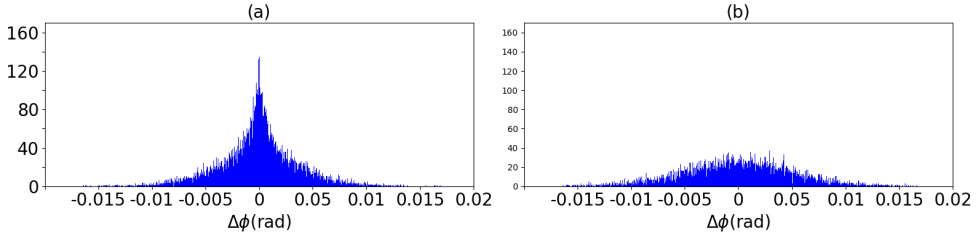


Figure 5.1: Distribution of the difference, in radians, between the rotation angle and the calculated one. (a) The difference without Gaussian noise added to the rotated signal. (b) The difference with Gaussian noise where $\mu = 0$ and σ is 10 percent of the maximum possible value of the signal.

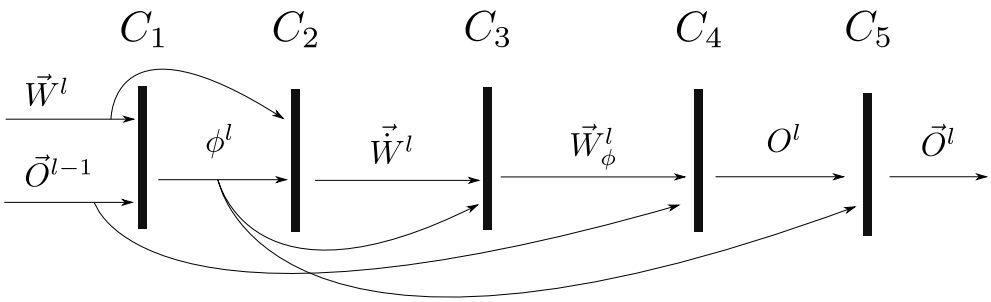


Figure 5.2: Computational pipeline of a filter $\vec{W}^l$ in layer l. C_{1-5} define the different operations. Input to the pipeline is a kernel of dimensions (F_h, F_w, c_{in}) and a window of the input with the same size with the kernel. For that combination, an angle ϕ^l is computed which is used to rotate the kernel. The rotated kernel is used to compute the scalar convolution which is then transformed to a vector.

simplicity we define the following shorthand indices:

$$\begin{aligned} p_w &: \text{weight position, } (i, j, c_{in}, c_{out}) \\ p_o &: \text{output position, } (i', j', c_{out}) \\ p_{in} &: \text{input position, } (i'', j'', c_{in}) \end{aligned} \tag{5.4}$$

For clarity, we note that if certain indices of a Tensor are not displayed, then the complete dimensionality is considered.

The first step (C_1) calculates the angle $\phi^l_{p_o}$ between the input to this layer $\vec{O}^{l-1}$ and a kernel $\vec{W}^l_{c_{out}}$ for all output positions and all output channels c_{out}. In the second and third step the rotated kernels $\vec{W}^l_{\phi\ p_o}$ are calculated ($C_{2,3}$), for all output positions and channels, where the first step C_2 only rotates in the 2D vector space for each channel and grid location (indicated by $\vec{W}$), and in the third step C_3 the rotation is

also performed with respect to the grid indices (i, j) using bi-linear interpolation, i.e., plane rotation. Without loss of generality we can consider both steps as one operation. The step C_4 performs a scalar convolution of the rotated kernels and the input. The scalar convolution result $O^l_{p_o}$ and the angle $\phi^l_{p_o}$ used to compute it are considered as vectors in the polar coordinate system which are then transformed to the Cartesian representation at step C_5.

In large networks, calculating rotated kernels, $\vec{W}^l_{\phi\ p_o}$, for all positions and channels is very inefficient. Thus, we precompute the rotation of the kernels for B angles, $\phi_b = b\frac{2\pi}{B}$ $(b = 0, ..., B-1)$. At each output point we calculate an angle $\phi^l_{p_o}$ using formulas 5.1 and 5.2 and then select the precomputed kernel with the closest angle to $\phi^l_{p_o}$. If the input pattern is significantly different from the kernels, the computed angle might still be very large. We set a maximum threshold for the angle, in which case the output $O^l_{p_o}$ is set to zero. The final pipeline can be seen in Figure 5.3. In order to reconstruct continuous angles ϕ and the respective scalar convolution result O_ϕ from the quantized angles ϕ_b, we make the assumption that for a small angle ϵ and the output of the scalar convolution at the most appropriate angle ϕ, O_ϕ, the convolution can be approximated as $O_{\phi+\epsilon} = O_\phi \cos\epsilon$. In our case this becomes:

$$O_{\phi_b} = O_\phi \cos\epsilon_b \Rightarrow O_\phi = \frac{O_{\phi_b}}{\cos\epsilon_b}, \tag{5.5}$$

where ϕ_b and $\epsilon_b = \phi - \phi_b$ are the closest quantized angle and the difference to the continuous angle. We also considered calculating the Taylor expansion to calculate the response in the real angle ϕ given the response at angle ϕ_b. For every order of the Taylor expansion an extra convolution operation has to be computed, which adds a lot to the final computational complexity. Thus we only implement a first order Taylor expansion:

$$O_\phi = O_{\phi_b} + \frac{\frac{\partial O_\phi}{\partial \phi}(\phi_b)}{1!} \cdot (\phi - \phi_b) \tag{5.6}$$

In our experiments, the correction in Equation 5.5 produced significantly better results than the first order Taylor expansion approximation 5.6. Given that it is also less computationally intensive, we use this correction for all our experiments.

5.4.2 Back propagation

Implementing the back propagation algorithm is straight forward for the operations C_4 and C_5, whereas some explanation is needed for operations C_1 and $C_{2,3}$ (for simplicity, we group C_2 and C_3, without effecting the derivations.) At operations $C_{2,3}$ the output is:

$$\vec{W}^l_{\phi\ p_o} = vector\_field\_rotation(\vec{W}^l_{c_{out}}, \phi^l_{p_o}), \tag{5.7}$$

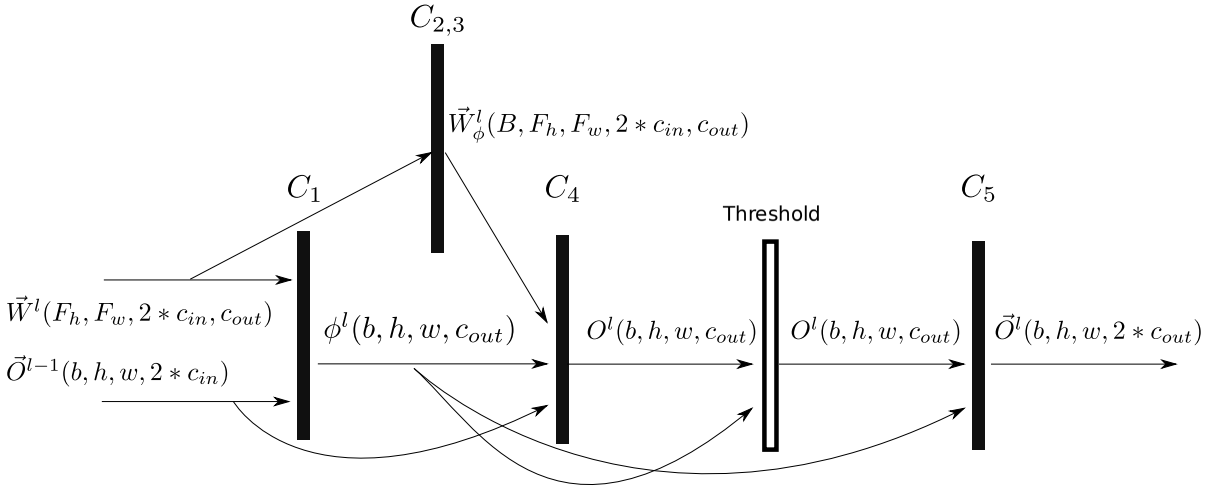


Figure 5.3: Implementation Workflow. In parentheses are the shapes of the respective tensors, where b is the batch size, h, w, c_{in} the size of the input, c_o the number of channels and F_h, F_w the kernel height and width respectively. Finally, B is the number of bins used for the precomputed rotated kernels.

where $\vec{W}^l_{c_{out}}$ is the complete vector field kernel of shape $(F_h, F_w, 2 \times c_i)$. For the back propagation two sets of gradients need to be computed, $\frac{\partial E}{\partial \vec{W}^l_{c_{out}}}$ and $\frac{\partial E}{\partial \phi^l_{p_o}}$. Since $\vec{W}^l_{\phi\ p_o}$ is the rotated $\vec{W}^l_{c_{out}}$, we get:

$$\begin{aligned}\frac{\partial E}{\partial \vec{W}^l_{c_{out}}} &= vector\_field\_rotation(\frac{\partial E}{\partial \vec{W}^l_{\phi\ p_o}}, -\phi^l_{p_o}),\ \frac{\partial E}{\partial \phi^l_{p_o}} \\ &= \sum_i \sum_j \sum_{c_{in}} \frac{\partial E}{\partial \vec{W}^l_{\phi\ p_o p_w}} \cdot \frac{\partial \vec{W}^l_{\phi\ p_o p_w}}{\partial \phi^l_{p_o}}\end{aligned} \tag{5.8}$$

To avoid confusion we note that $\vec{W}^l_{\phi\ p_o p_w}$ refers to the rotated weights for the output position p_o indexed by p_w. The last term in Equations 5.8 above can be estimated with the precomputed kernel weight sets for the quantized angle ϕ_b as:

$$\frac{\partial \vec{W}^l_{\phi_b}}{\partial \phi^l} = \frac{\vec{W}^l_{\phi_{b+1}} - \vec{W}^l_{\phi_{b-1}}}{2\frac{2\pi}{B}} \tag{5.9}$$

To back propagate the gradient through step C_1, where $\phi^l_{p_o} = \arctan(\frac{conv_2}{conv_0})$, we need to calculate two sets of gradients:

$$\frac{\partial E}{\partial conv_0} = -\frac{\partial E}{\partial \phi^l_{p_o}} \frac{conv_2}{conv_0^2 + conv_2^2},\ \frac{\partial E}{\partial conv_2} = \frac{\partial E}{\partial \phi^l_{p_o}} \frac{conv_0}{conv_0^2 + conv_2^2} \tag{5.10}$$

Equations 5.10 introduce numeric instabilities when both $conv_0$ and $conv_2$ are very close to zero, and even undefined when exactly zero. As a remedy, we set a small threshold for $conv_0$ and $conv_2$, below which we turn off the feature map at the specific location. The derivatives of $conv_0$ and $conv_2$ can easily be calculated from their definitions in Eq. 5.2. The analytical derivations can be found in the Appendix A.

5.5 Experiments

5.5.1 Datasets and ground truth

In order to evaluate our method, we perform experiments on rotation invariance and rotation equivariance. With that goal in mind we utilize the MNIST-rot [200, 197] dataset and a car orientation dataset [131]. Moreover, we construct our own version of the MNIST-rot, as explained in Section 5.5.4, to help us evaluate the rotation equivariance property of the methods tested.

For the RotEqNet [233] networks and their Clifford convolution counterparts we extract the gradients of the color from a 3×3 window and utilize them as input, in order to have a vector field as input.

5.5.2 Networks

For our experiments we utilize two network architectures, i.e. the $SFCNN$ [409, 408] and the RotEqNet [233]. The $SFCNN$ represent the state of the art in rotation equivariance and equivariance, utilizing group convolutions whilst the RotEqNet remains one of highest performing approaches and its much less computationally intensive than the group convolution approach.

Each layer of the RotEqNet performs convolutions in a number of predefined orientations of the kernels and then forwards the maximum response. Furthermore, it utilizes the orientation information to create a vector field as an output, i.e., the magnitude of each vector is the maximum response and the angle is the rotation angle of the orientation that produced that maximum response. Every convolutional layer is followed by batch normalization and a ReLU activation function (both applied on the magnitudes of the vectors). We also experimented with batch normalization with only rescaling, as proposed by the authors [233], but in our experiments the full batch normalization proved to produce better results. After a number of layers global pooling is applied and the resulted information is processed by a number of fully connected layers which perform classification.

The Clifford convolution (cc) inspired counterparts of the RotEqNet follow the same architecture with a few differences. Instead of full batch normalization we apply only rescaling (as with the cc networks it proved to result in higher performance) and the addition of a second normalization which makes the average norm of a square window, with size equal to the next layer's kernel size, to be one (Equation 5.11). This is needed in the cc networks due to the threshold introduced at the end of Section 5.4.2. With varying input norm the effect of the threshold changes. For fair comparison we did also try to add them at the original RotEqNet and found that it marginally improves the performance and thus utilized it in all our experiments.

$$norm(\vec{O}^{l-1}) = \frac{\vec{O}^{l-1}}{\lambda_O},\ \lambda_O = \sqrt{\frac{F_w^l F_h^l}{O_w^{l-1} O_h^{l-1}}} ||O^{l-1}||_2 \tag{5.11}$$

For the $SFCNN$ we follow the proposed hyperparameter settings proposed in [408] and reproduced their results. The first layer outputs the response of the networks for the predefined orientations (sixteen orientations). The rest of the convolutional layers besides the standard spatial 2D convolution, convolve in the angular dimension as well and output responses for all orientations. Before their three fully connected layers they perform both spatial and orientation pooling by applying max-pooling.

The cc counterpart utilizes the input layer and group convolutions as with the max-pooling version. The difference lies in the orientation pooling layer in which we

Table 5.1: Number of channels per layer. CL denotes Convolutional Layer, FC are Fully Connected and Out is the classification layer. With (/2) we denote layers that are followed by 2x2 spatial pooling. (-) denotes layers for which the kernel size is the same as the input and do not use padding, producing output size 1x1.

	CL1	CL2 (/2)	CL3	CL4 (/2)	CL5	CL6(-)	FC	FC	Out
kernel size	9	7	7	7	7	5	-	-	-
$rotEqNet$ [233]	16	32	36	36	64	96	96	96	10
$SFCNN$ [409]	24	32	36	36	64	96	96	96	10
$D_{16\|5}C_{16}{}^{*}$ [408]	24	32	36	36	64	96	96	96	10

utilize the cc approach.

All the above networks have fully connected layers performing the classification and thus rendering the whole network as rotation invariant and not rotation equivariant. In order to test rotation equivariant networks, we consider the fully connected layers as convolutional layers with kernel size 1×1 and thus we are able to utilize the orientation equivariant operators above. The output of the network outputs classification probabilities (after a soft max activation) as well as a prediction angle, either produced by the argmax for max-pooling or by the cc operator. In the case of the cc operator, the produced angle is a function of the weights and the input and thus can be used for extra supervision, by utilizing the ground truth angle of the examples. The resulted networks approximate end-to-end orientation equivariant behavior.

For our experiments we implemented our forward pass and back propagation steps for using the "new op" module of Tensorflow [1], both for CPU and GPU computations. Our source code has been tested with Tensorflow versions 1.12 and 1.13, with CUDA version 10.0. All experiments were done on nVidia Geforce GTX 1080Ti and RTX 2080Ti GPUs.

5.5.3 MNIST-rot

For all our experiments, we use the number of layers as well as number and sizes of kernels as in [409, 408, 233]. All the relevant parameters can be seen in Table 5.1. For reference we also show the parameters and performance of the $D_{16|5}C_{16}$ [408] network since, to the best of our knowledge, it has the highest performance on the rotated MNIST dataset. It should be noted that it is very similar to the $SFCNN$, utilizing group convolutions. The main difference is that it performs convolution over the flip operation as well for the first few layers.

Table 5.2: Classification accuracy of the networks on the rotated MNIST dataset. * denotes results reported in the respective papers. Values in parentheses denote average performance over 5 runs and outside the parenthesis is the maximum performance achieved. In case there is no value in parenthesis the presented value is the performance of the only run.

	op	Acc. our lr scheduler	Acc. [408] lr scheduler
$RotEqNet$ [233]	mp	98.954 (98.918)	99.064 (99.031)
$RotEqNet$	cc	99.088 (99.071)	99.004 (98.974)
$SFCNN$ [409]	mp	99.358 (**99.308**)	99.320 (99.294)
$SFCNN$ [409]	cc	99.318	99.238 (99.166)
$RotEqNet^*$ [233]	mp	-	- (98.99)
$SFCNN^*$ [409]	mp	-	- (99.286)
$D_{16\|5}C_{16}{}^*$ [408]	mp	-	- (**99.318**)

For all networks we trained using the Adam optimizer [175] with learning rate 10^{-3} and a mini-batch size of 128. We drop the learning rate to 10^{-5} at 30 epochs and train for 320 epochs overall. We also train with the scheduler defined in [409] for fairness. The angle threshold defined in Section 5.4.1 is set to $\frac{\pi}{2}$, as it proved to increase the performance. The threshold due to gradient instabilities, described in Section 5.4.2, is set to 10^{-3}.

Following the practice in literature [233, 409, 408] we set the number of orientations to $B = 16$. For both Clifford and max-pooling based networks. We found out that training with 16 bins and testing with 32 slightly increases the performance, while keeping the training times lower. For the $SFCNN$ this is not possible since increasing the number of bins B changes the shape of the weight matrices as well. The results can be seen in Table 5.2.

We should note that as it is apparent that the learning rate scheduler of [408] is not favorable to the Clifford convolution (cc) inspired operator, since using our scheduler increases the performance of the networks. When replacing the max-pooling operation with the cc operator, the accuracy of the $RotEqNet$ model increases by a small margin. When applied on the $SFCNN$ the result remains approximately the same (with our learning rate scheduler). As a benefit, our method has much less computational intensity discussed in Section 5.5.6.

5.5.4 Enriched MNIST-rot

As a second test, we wanted to also predict the angles of the digits. Unfortunately the original MNIST-rot dataset does not provide the angles with which each image is rotated and thus it is not possible to evaluate the performance of the networks. Thus, we create our own MNIST-rot in the same way that is described in the original work [200, 197], with the only difference that we use cubic spline interpolation instead of bilinear interpolation for rotation. Moreover, we save the angle used to rotate each digit.

The networks in the previous section can not predict any angles and thus we adjust them. We change the fully connected layers (even the output) to convolutional, with kernel size 1x1. Then we perform the same orientation prediction as with the rest convolutional layers, i.e., max-pooling with argmax or the one proposed here (Equations 5.1, 5.2). Similarly, for the $SFCNN$ networks we remove the orientation pooling layer after the last convolution, transform the two fully connected layers to group convolutional layers with filter size 1x1 and make the output layer orientation equivariant with either argmax or Clifford convolution to predict the angles (like the RotEqNet).

Our method is the only one that can utilize angle information as a supervisory signal (without big adjustments on the network). Thus we implement two setups. In the first setup the networks are trained only with class labels and the angle is left unsupervised. For the second setup the angle information is also utilized during training. The networks that utilize the max-pooling with argmax can not utilize angle information. Thus we include a second output branch (in parallel to the first) which is tasked to predict the sinus and cosine of the angle. This second branch consists of two layers. We follow the approach of [233], where a hand crafted fully connected layer of the form $[[\cos(0), \cos(1 \cdot \frac{2\pi}{B}), \cos(2 \cdot \frac{2\pi}{B}), \cdots, \cos((B-1) \cdot \frac{2\pi}{B})], [\sin(0), \sin(1 \cdot \frac{2\pi}{B}), \sin(2 \cdot \frac{2\pi}{B}), \cdots, \sin((B-1) \cdot \frac{2\pi}{B})]]$ takes as input the responses of all orientations of the previous layer. A final fully connected layer predicts the sinus and cosine of the angle. An interesting remark is that this method utilizes a secondary network that predicts the angle, increasing the complexity of the network and its trainable parameters. Meanwhile, the Clifford convolution approach calculates the angle from the existing weight vectors and their respective input. It is also important to note that the unsupervised setup does not utilize this new branch and the predicted angle is the result of the argmax function.

During test, we expect that the networks that utilized the angle information while training will be able and predict the correct orientation of the input. In the case of unsupervised angles, we expect the output of the network to be rotation equivariant. Thus, there should be a steady offset between the angles predicted by the network

and the ground truth angles. In order to test the angle prediction performance, we compute the average angle difference between predicted and ground truth angles on the training set. We use that offset to "correct" the predicted test angles and measure the difference to the ground truth.

To measure the performance on angle prediction, the absolute value ($\in [0, \pi)$) of the difference between the predicted angles and the ground truth angles is averaged out. The results of the unsupervised angle experiment can be seen in Table 5.3a and those of the supervised angle experiment in Table 5.3b. For a fair comparison we also train the networks from the previous section, i.e., not equivariant fully connected layers with this version of the MNIST-rot (four first rows of Table 5.3a).

From the first four rows of Table 5.3a, we see that the methods have similar performance on the proposed MNIST-rot with the original one in Table 5.2. It is important to note that all experiments here use our learning rate scheduler instead of the one used in [408]. Regarding classification accuracy, the fully convolutional and end-to-end rotation equivariant networks seem to have similar performance to the non-fully convolutional counterparts. Regarding the angle prediction (unsupervised in this case), the inclusion of the cc operator increases the performance of the $RotEqNet$ by a significant margin. This is not the case though with the $SFCNN$. There the unsupervised angle prediction becomes worse than the performance of the max-pooling. In a closer inspection we found out that this is a result of the vector field construction we used for the $SFCNN$, where most of the vectors end up being parallel due to the fact that the group convolutional layers approximate very well rotation equivariant behavior.

When the angle information is used to supervise the networks, it is apparent that the cc operator improves the angle prediction performance over the introduced branch used for the networks utilizing the max-pooling operator. Regarding classification performance we see a similar trend to our previous two experiments.

As an extra experiment, we train the networks on only straight digits (from the original MNIST) and measure their performance on the test set of our MNIST-rot. For training we pick randomly 12K digits from the original MNIST (the same for all networks) and train with the learning rate scheduler proposed in this work. The results can be seen in Table 5.4. It is obvious that since the extra branch introduced for the previous experiment explicitly learns to predict the angles and is not inherently rotation equivariant, it does not generalize to unseen angles and its performance is very poor. On the other hand the cc operator manages to maintain high angle prediction performance.

Table 5.3: Classification accuracy and angle prediction performance of the networks on the enriched rotated MNIST dataset. "(Full)" donates networks that are equivariant end-to-end, i.e., they output angle of prediction. The first column denotes the method, the second the orientation pooling (op) method, while the third and forth columns the classification accuracy and average absolute angle difference (in radians) respectively. Values in parentheses denote average performance over 5 runs and outside the parenthesis is the maximum performance achieved. In case there is no value in parentheses the presented value is the performance of the only run. In case of two values separated by a '/', the first value is the model that achieved maximum classification accuracy while the second is the model that achieved minimum average absolute angle difference.

	op	Accuracy	Angle Prediction
$RotEqNet$ [233]	mp	99.06 (98.99)	-
$RotEqNet$	cc	99.14 (99.12)	-
$SFCNN$ [409]	mp	99.36 (99.32)	-
$SFCNN$ [409]	cc	99.28	-
$RotEqNet$ (Full) [233]	mp	99.05/98.92 (98.99)	0.4441/0.3711 (0.4301)
$RotEqNet$ (Full)	cc	99.15/99.08 (99.09)	0.2970/0.2756 (0.3059)
$RotEqNet$ (0.5 dropout, Full)	cc	99.14/99.11 (99.10)	0.2720/0.2579 (0.2696)
$SFCNN$ (Full)	mp	99.37 (99.36)	0.1798 (0.1843)
$SFCNN$ (Full)	cc	99.29 (99.27)	0.1985 (0.2227)

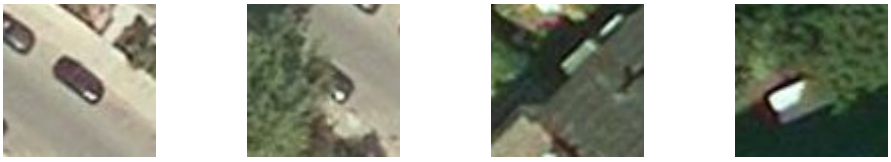
(a) Unsupervised angles.

	op	Accuracy	Angle Prediction
$RotEqNet$ (Full) [233]	mp	98.93/98.88 (98.84)	0.1156/0.1140 (0.1200)
$RotEqNet$ (Full)	cc	99.15 (99.10)	0.0985 (0.1005)
$RotEqNet$ (0.5 dropout, Full)	cc	99.29 (99.24)	0.1095 (0.1110)
$SFCNN$ (Full)	mp	99.38/99.35 (99.34)	0.0688/0.0659 (0.0694)
$SFCNN$ (Full)	cc	99.33/99.28 (99.26)	0.0608/0.0580 (0.0620)

(b) Supervised angles.

Table 5.4: Classification accuracy and angle prediction performance of the networks on the our rotated MNIST dataset, with supervised angles trained on 12K digits of the original MNIST dataset. "(Full)" donates networks that are equivariant end-to-end, i.e., they output angle of prediction. The first column denotes the method, the second the orientation pooling (op) method, while the third and forth columns the classification accuracy and average absolute angle difference (in radians) respectively.

	op	Accuracy	Angle Prediction
$RotEqNet$ (Full) [233]	mp	98.33	1.5693
$RotEqNet$ (0.5 dropout, Full)	cc	98.41	0.1892
$SFCNN$ (Full)	mp	99.12	0.3183
$SFCNN$ (Full)	cc	98.99	0.1727



(a) (b) (c) (d)

Figure 5.4: Example images from the vehicle orientation dataset. (a) and (b) are positive examples (include vehicles), (c) and (d) are negative examples.

5.5.5 Vehicle Orientation

As mentioned above we also utilize the vehicle orientation dataset from [131]. This dataset consists of 15 aerial images with labeled vehicles and their orientations. In order to include non trivial negative examples, i.e., crops with no vehicle in them, we add specific crops from these images that do not include vehicles. We automatically extract negative patches that are challenging by utilizing the results of the detector of [129]. We pick the patches that have high scores, i.e., high confidence for vehicle detection, but have very small or no overlap to positive crops. We use the same number of negative examples are there are positive in the train set and no negative examples in the test set. To further enrich the dataset, we extract patches of resolution 76×76 randomly rotate them and crop the center 48×48 pixels to be used as input to the networks. With this dataset we train on both vehicle detection by patch classification and orientation estimation. Some pictures from our dataset can be seen in Figure 5.4. The network structures can be seen in Table 5.5 and the results in Table 5.6.

Our results show that in the case of the car vehicle detection and orientation

Table 5.5: Number of channels per layer. CL denotes Convolutional Layer, FC are Fully Connected and Out is the classification layer. With (/2) we denote layers that are followed by 2x2 spatial pooling. (-) denotes layers for which the kernel size is the same as the input and do not use padding, producing output size 1x1.

	CL1	CL2 (/2)	CL3	CL4	CL5(-)	Out
kernel size	9	7	7	5	7	-
$rotEqNet$ [233]	4	8	8	8	16	2
$SFCNN$ [409]	4	8	8	8	16	2

Table 5.6: Classification accuracy and angle prediction performance of the networks on the vehicle orientation dataset, with supervised angles. The first column denotes the method, the second the orientation pooling (op) method, while the third and forth columns the classification accuracy and average absolute angle difference (in radians) respectively. Values in parentheses denote average performance over 5 runs and outside the parenthesis is the maximum performance achieved. In case of two values separated by a '/', the first value is the model that achieved maximum classification accuracy while the second the model that achieved minimum average absolute angle difference.

	op	Accuracy	Angle Prediction
$RotEqNet$ [233]	mp	87.08 (83.88)	0.5355 (0.5894)
$RotEqNet$	cc	93.06/91.39 (91.39)	0.3694/0.3614 (0.4433)
$SFCNN$	mp	94.02 (93.83)	0.4348 (0.4796)
$SFCNN$	cc	96.41 (95.50)	0.3325 (0.4093)

Table 5.7: Time required to process a mini-batch of size 200. "(Full)" donates networks that are equivariant end-to-end, i.e., they output angle of prediction. The first column denotes the method, the second the orientation pooling (op) method, while the third and forth columns the average time in seconds on a GPU and a CPU respectively.

	op	GPU Time (s)	CPU time (s)
$RotEqNet$ (Full) [233]	mp	**0.1248**	3.1202
$RotEqNet$ (Full)	cc	0.4880	**1.8781**
$SFCNN$ (Full)	mp	0.2180	8.3354

prediction dataset the cc networks, both $RotEqNet$ and $SFCNN$ outperform their max-pooling counterparts both in classification accuracy and angle estimation.

5.5.6 Computational complexity

The last experiment is designed to compare the time each network requires to process data. For this test, we process with each network 100 batches of batch size 200 and average the the time required to precess each batch. The results can be seen in Table 5.7. For each layer, the $RotEqNet$ with max-pooling requires B convolutions to be performed, a $reduce\_max$ and $argmax$ function. In the current implementation, the equivalent cc layer requires 9 convolutions and an $arctan$ function, which overall is much lower than the $RotEqNet$. The $SFCNN$ on the other hand, not only requires B convolutions, but the dimensionality of the filters and input is also B times larger than the equivalent $RotEqNet$. Thus, we expect the cc implementation to be much faster on average than the other two approaches. Unfortunately, this is not always the case. In particular, for GPU computations the cc operator is much slower than the max-pooling counterpart. We believe that there are two main reasons for this finding. The first is that the efficiency of our code is not on par with the native code of Tensorflow. Moreover, when it comes to GPU computations, the architecture is such that the constant change of the filter being used is penalized a lot due to intense input/output overhead. This is not the case with CPU computations, where we can observe the benefit of the cc approach. Nonetheless, the difference we observe is still smaller than the one expected, leading us to the belief that with higher optimization the cc method can have a big benefit in computational cost and processing time.

5.6 Conclusion and future work

In this chapter, vector field representation is utilized and a new operator, based on Clifford convolution, is proposed for measuring feature angles in images and construct rotation equivariant CNNs. The angles are computed solely based on the kernels and the input signal, enabling the use of this information during training through back-propagation, which increases the accuracy of the resulting networks while being memory and computationally more efficient. We compare our networks to the state of the art rotation equivariant method, using orientation pooling through max-pooling.

The new operator is evaluated with two network architectures representative of the state of the art on the rotated MNIST and a vehicle orientation dataset [131] in rotation invariance, equivariance and time complexity. Our experiments show that our method is competitive to the state of the art in classification accuracy, while gaining higher accuracy in angle prediction and having a lower computational and memory resource demand.

Our work clearly demonstrates the potential of the proposed method in achieving orientation equivariant networks. However, more detailed insights are necessary. For example, the influence of the filter size on the performance, or the robustness and performance with increasing depth of networks has to be investigated. Applying our method to other datasets will give valuable insights as well. A very promising route is provided by the possibility to combine our Clifford convolution approach with other state of the art architectures, such as the res-block [127] and dense-block [146].

6

Norm Loss: Regularizing artificial neural networks

Convolutional neural network training can suffer from diverse issues like exploding or vanishing gradients, scaling-based weight space symmetry and covariant-shift. In order to address these issues, researchers develop weight regularization methods and activation normalization methods. In this chapter, a weight soft-regularization method is proposed based on the Oblique manifold. The proposed method uses a loss function which pushes each weight vector to have a norm close to one, i.e., the weight matrix is smoothly steered toward the so-called Oblique manifold. It is evaluated on the very popular CIFAR-10, CIFAR-100 and ImageNet 2012 datasets using two state of the art architectures, namely the ResNet and wide-ResNet. This regularization approach introduces negligible computational overhead and the results show that it is competitive to the state of the art and in some cases superior to it. Additionally, the results are less sensitive to hyperparameter settings such as batch size and regularization factor.

6.1 Introduction

Convolutional neural networks, and deep learning in general, have received a lot of attention in the past few years [184, 127, 437, 111, 368] and have been applied in many research areas, such as image understanding [111, 368], natural language processing [431, 300] and game solving [248]. The research in these models has been motivated by the success of deep learning in image classification with AlexNet [184] and later by much deeper models such as VGG [338], ResNet [127] and wide-ResNet [437]. In order to understand these models and enhance their performance, a lot of research has been carried out and in many directions [111, 368], including data preprocessing strategies [59, 214], activation normalization [394, 418], weight regularization [17, 147], activation functions [381], and overall network architectures [127, 437, 119]. This chapter focuses on weight regularization methods and a new soft regularization loss function, the norm loss, is proposed.

Weight regularization has been a wide research topic. The main issues of deep learning training procedures are the exploding/vanishing gradients, as well as the scaling-based weight space symmetry and covariant-shift [153, 148, 17]. In order to alleviate one or more of the aforementioned issues, researchers are developing methods that restrict the search space of possible weight vectors.

One of the first approaches, and most popular, is the weight decay [184], which adds a penalty to the Euclidean norm of the weight vector. In the past few years a number of approaches and theories have emerged, regarding weight regularization, such as weight normalization [311, 148] and weight orthogonalization [149, 17, 147]. The most popular goal for regularization methods has been weight orthogonalization. The reason is that orthogonal weights have very nice properties in relation to deep network training [17, 48, 147]. For example, orthogonal weights can obtain dynamical symmetry [147] which accelerates convergence. Nonetheless, it is not always possible to have orthogonal weights, due to the fact that in many DNN architectures, the weight matrices are over-complete. Moreover, hard-orthogonalization can restrict the learning capacity of a network [17]. As an oversimplified, intuitive example of learning capacity restriction we show the schematic visualization of the weights of two 3×3, one-channel kernels in Figure 6.1. A combination of (a) with a ReLU activation function would result in upper edge detection, while the combination of (b) with a ReLU activation function would result in lower edge detection. The inner product of these filters is -1 and thus it would not be possible to have both kernels in a layer restricted to orthogonal weight matrices. Thus, it is possible that in some cases weight orthogonalization can be too restrictive, hurting the final performance. In order to overcome this limitation some authors, like [17], reduce the effect of regularization after some point during training, or drop it completely.

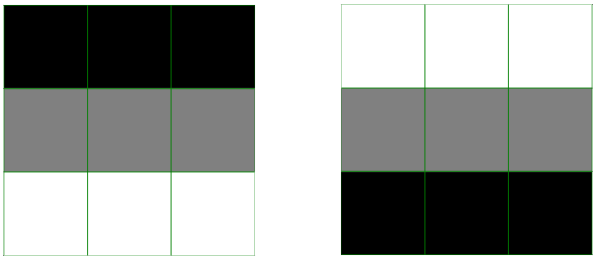


Figure 6.1: Schematic visualization of the weights of two example 3×3 one-channel kernels. White (dark) values indicate high (low) numbers. The weight matrix in example (b) is a 180°rotated version of (a).

In this chapter, weight regularization is addressed by imposing a normalization constraint and propose a new soft-regularization method pushing the weight matrix into the Oblique manifold [2, 148]. To that end a new loss function is proposed, the norm loss. Its performance is evaluated on state of the art networks on standard research benchmarks, i.e., CIFAR-10, CIFAR-100 [183] and ImageNet 2012 [302]. The experimental results show that networks trained with the norm loss achieve comparable to the state of the art performance and in some cases superior to it, while having negligible computational overhead. To the best of our knowledge we are the first to propose this regularization loss function.

The rest of this chapter is organized as follows. In Section 6.2 the related research to norm loss is presented, in Section 6.3 describes the theory on which the proposed approach is based and in Section 6.4 the proposed method is formulated. In Section 6.5 the experimental procedure as well as the experimental results are presented and finally the conclusions are discussed in Section 6.6.

6.2 Related work

There are many methods that try and alleviate the scaling-based weight space symmetry and the exploding gradients problems. Most of them can be classified into one of two groups, namely hard-regularization and soft-regularization [147]. A very well known approach is the inclusion of the weight decay loss to the total loss which imposes a penalty to the magnitude of the weight vectors [184]. It can be considered as a soft-regularization method, since it applies a small change to the weights or the gradients (depending on the implementation) in every step. Although this method does address the exploding gradients issue, it does not address the scaling-based

weight space symmetry problem. Most methods try to normalize the weights, i.e. force them to have unit norm, make them orthogonal, i.e., $\mathbf{W}^T\mathbf{W} = \mathbf{I} * \lambda$, where λ is a small scaling factor, or both, i.e., $\mathbf{W}^T\mathbf{W} = \mathbf{I}$. Another work proposed to use the norm of the CNN Jacobian as a training to regularize the models [343], while [430] proposed spectral norm regularization, which penalizes the high spectral norm of weight matrices.

Hard-regularization methods specify hard limits for the weights. For example, [311] divide the weight vector with its norm ($\mathbf{W}^* = \frac{\mathbf{W}}{\|\mathbf{W}\|}$), before applying them to the input, ensuring that the applied weights are normalized. This method is computationally expensive since the normalization happens on the fly. A later work [148] ensures that the weights are normalized by normalizing the weight vector every T iterations, i.e., instead of normalizing on the fly they change the original weight vector. This method is more computationally efficient than the previous one and experiments show that it also produces higher performing networks. Moreover, it is the most similar to the norm loss approach as it is targeting the same goal but with a soft constraint. Instead of abruptly changing the weights to force unit norm, a term is added to the loss function (instead of the weight decay) to smoothly guide the weights towards unit norm.

Other regularization methods try to force the weights to be orthogonal to each other, i.e., $\mathbf{W}^T\mathbf{W} = \mathbf{I}$ [149, 147]. Some works [389, 17, 422, 15] apply soft regularization methods by utilizing the standard Frobenius norm and define a term in the training loss function which requires the Gram matrix of the weight matrix to be close to identity, i.e.:

$$L = \lambda\|\mathbf{W}^T\mathbf{W} - \mathbf{I}\| \tag{6.1}$$

Although this is an efficient method, when considering over-complete weight vectors it can only be a rough approximation [17]. In order to overcome this issue, [17] proposed and tested a number of different regularization terms, all focused around weight orthogonalization. The norm loss is similar to the aforementioned, in the sense that a loose regularization term is applied on the loss function. Norm loss is a bit softer regularization since it only loosely constrains the norm of the weight vectors and does not force them to be orthogonal. Although orthogonal weights have many nice properties, they also restrict the learning capacity of the weights [147].

One of the first works that implemented weight orthogonalization with hard regularization, did so for only fully connected layers [122]. They defined the generalized back propagation algorithm and compute gradients on the Stiefel manifold. Though this procedure requires the form of Riemannian gradient and a retraction mechanism, which are computationally intensive. In a later work [267] extended this approach to convolutional layers as well. [149], [147] propose re-parameterization techniques in order to overcome the limitations of a retraction mechanism. [149] uses eigen

decomposition to calculate the transformation, whilst [147] used the Newtonian iteration (ONI), which is shown to be more stable and more computationally efficient. The last method also allows for weights to be orthogonal but not normalized, i.e. $\mathbf{W}^T\mathbf{W} = \mathbf{I} * \lambda$. The approach proposed in this chapter provides a softer regularization than all aforementioned, with the exception of weight decay, whilst having negligible computational overhead over the "vanilla" weight decay.

6.3 Preliminaries

Given a set of corresponding sets $\{\mathbf{X}, \mathbf{Y}\}$, where every $x_i \in \mathbf{X}$ has a unique corresponding value, or ground-truth label, $y_i \in \mathbf{Y}$, a neural network f should predict the value y_i for the corresponding input x_i with a set of model parameters $\mathbf{W}$. For a given set of model parameters, the neural network output, i.e., the predicted values $\tilde{y_i} = f(x_i, \mathbf{W})$, does not match the desired output y_i. If $\mathbf{E}$ is the discrepancy between the $\mathbf{Y}$ and the predicted values $\tilde{\mathbf{Y}}$, the aim of the network training process is to find a set of parameters $\mathbf{W}$ that minimize $E(\mathbf{Y}, \tilde{\mathbf{Y}})$. This is done with the help of a differentiable loss function $L(\mathbf{Y}, \tilde{\mathbf{Y}})$. The training process is carried out by changing by a small factor the set of parameters $\mathbf{W}$ in the direction that minimizes $L(\cdot)$. For an example x_i, y_i or a set of examples $\{\mathbf{X}, \mathbf{Y}\}$ the direction is given by the gradients of L, and the weights $\mathbf{W}$ are updated as follows:

$$\mathbf{W}_{new} = \mathbf{W} - \eta \frac{\partial L}{\partial \mathbf{W}} \tag{6.2}$$

where $\eta \ll 1$ denotes the learning rate.

Remark: Since the Oblique manifold defines matrices with normalized rows, the weight matrix is constructed as a row-matrix of the individual weight vectors of each filter, $\mathbf{W} \in \mathbb{R}^{n \times p}$ where p is the dimensionality of each weight vector of a filter, while n is the number of filters. This corresponds to the transpose of the weight matrix typically used in the orthogonality related literature [17].

As shown by [148], the Hessian matrix can be ill-conditioned due to the scaling-based weight space symmetry. In an effort to avoid this issue they propose to optimize the network parameters using the Riemannian optimization [3] over the Oblique manifold. The Oblique manifold $\mathcal{BO}(n, p)$ defines a subset of $\mathbb{R}^{n \times p}$, where for every $\mathbf{W} \in \mathcal{BO}(n, p)$:

$$ddiag(\mathbf{W}\mathbf{W}^T) = I \tag{6.3}$$

where $ddiag(\cdot)$ is a function that sets all elements of an input array except the diagonal to zero. The above formulation restricts all rows of the matrix $\mathbf{W}$ to have a norm of

one. Notice that imposing the requirement of equation 6.3 is less restricting than the usual orthogonalization requirement $\mathbf{W}\mathbf{W}^T = I$, since it only enforces a unit norm but does not enforce the weight vectors to be orthogonal to each other.

Given a set of weights $\mathbf{w}$ of a single neuron, i.e., $\mathbf{w} \in \mathbb{R}^{1\times p}$, where $\mathbf{w}\mathbf{w}^T = 1$, the Riemannian gradients are given by the following equation [148]:

$$\widehat{\frac{\partial L}{\partial \mathbf{W}}} = \frac{\partial L}{\partial \mathbf{W}} - \left(\mathbf{W}^T \frac{\partial L}{\partial \mathbf{W}}\right)\mathbf{w} \tag{6.4}$$

where the norm of the gradients is bound [148]:

$$\left\|\frac{\partial L}{\partial \mathbf{W}}\right\| \leq \left\|\left(\mathbf{W}^T \frac{\partial L}{\partial \mathbf{W}}\right)\mathbf{w}\right\| \tag{6.5}$$

From equation 6.5 and experimental evidence they conclude that $\frac{\partial L}{\partial \mathbf{W}}$ is the dominant factor of the derivative in equation 6.4 and thus propose to apply the Euclidean gradient (equation 6.2) and then project the weights back to the Oblique manifold by normalizing them:

$$\mathbf{w}^{new} = \frac{\mathbf{w}}{\|\mathbf{w}\|} \tag{6.6}$$

6.4 Proposed method

Inspired by the insights of using the Riemannian gradients described in the previous section, we propose to use a similar normalization for neural network training. However, the hard change in the weights after normalization can cause disturbance in the training process since they abruptly shift the weights from the direction of the original gradients. This can be a problem with large learning rates or large projection period T, where T is the number of training steps between each projection operation [148]. In order to overcome this issue we propose a soft regularization method that instead of abruptly changing the weights, slowly guides them towards the Oblique manifold, i.e., to have unit norm. We implement that by introducing a normalization loss, or norm loss (nl):

$$L_{nl} = \sum_{c_o=1}^{C_o}\left(1 - \sqrt{\sum_{c_i=1}^{C_i}\sum_{i=1}^{F_h}\sum_{j=1}^{F_w} w_{ijc_ic_o}^2}\right)^2 \tag{6.7}$$

where F_w, F_h, C_i, C_o are the filter (or weight vector) width, height, number of input and number of output channels respectively. The loss is penalizing the weight vector of each neuron if its Euclidean norm is different from one. The final loss function then becomes:

$$L_{total} = L_{target} + \lambda_{nl} \cdot L_{nl} \tag{6.8}$$

where λ_{nl} a small factor that determines how strong the regularization will be and L_{target} is the loss function defined by the task to be solved, e.g., triplet loss function, cross entropy loss etc.

6.4.1 Connection to weight decay

The weight decay is similar to norm loss since it introduces a small penalty on the magnitude of the weights. The loss function for the weight decay is given by:

$$L_{wd} = \sum_{c_o=1}^{C_o} \sum_{c_i=1}^{C_i} \sum_{i=1}^{F_h} \sum_{j=1}^{F_w} w_{ijc_ic_o}^2 \tag{6.9}$$

There are two main differences. Firstly, weight decay penalizes the absolute magnitude of the weight vector while norm loss penalizes the deviation to having unit norm. This means that in the case where the norm is smaller than one, our method will try to increase it, whilst the weight decay will continue pushing to decrease it. This makes a difference in situations where some components are rarely being utilized (e.g., due to nonlinearities such as the ReLU) in which case the main loss changing these components is the weight decay loss, resulting in very small weights that might never recover. The second difference is that it applies to all components of a layer uniformly, whilst the norm loss differentiates between the vectors of each output channel. This can be seen by the derivatives of each method. The derivatives for each component of the weight matrix are given by:

$$\frac{\partial L_{wd}}{\partial w_{ijc_ic_o}} = 2w_{ijc_ic_o} \tag{6.10}$$

From Equation 6.7 it is easy to derive the gradients of the norm loss:

$$\begin{aligned} L_{nl} &= \sum_{c_o=1}^{C_o} \left(1 - \sqrt{\sum_{c_i=1}^{C_i} \sum_{i=1}^{F_h} \sum_{j=1}^{F_w} w_{ijc_ic_o}^2}\right)^2 \\ \Rightarrow \frac{\partial L_{nl}}{\partial w_{ijc_ic_o}} &= \frac{\partial}{\partial w_{ijc_ic_o}} \left(\sum_{c_o=1}^{C_o} \left(1 - \sqrt{\sum_{c_i=1}^{C_i} \sum_{i=1}^{F_h} \sum_{j=1}^{F_w} w_{ijc_ic_o}^2}\right)^2\right) = \\ &\frac{\partial}{\partial w_{ijc_ic_o}} \left(1 - \sqrt{\sum_{c_i=1}^{C_i} \sum_{i=1}^{F_h} \sum_{j=1}^{F_w} w_{ijc_ic_o}^2}\right)^2 = \\ &2\left(1 - \sqrt{\sum_{c_i=1}^{C_i} \sum_{i=1}^{F_h} \sum_{j=1}^{F_w} w_{ijc_ic_o}^2}\right) \frac{\partial}{\partial w_{ijc_ic_o}} \left(1 - \sqrt{\sum_{c_i=1}^{C_i} \sum_{i=1}^{F_h} \sum_{j=1}^{F_w} w_{ijc_ic_o}^2}\right) \end{aligned} \tag{6.11}$$

The norm of a weight matrix of a kernel with index c_o is:

$$\|w_{c_o}\| = \sqrt{\sum_{c_i=1}^{C_i}\sum_{i=1}^{F_h}\sum_{j=1}^{F_w} w_{ijc_ic_o}^2} \tag{6.12}$$

From equations 6.11, 6.12:

$$\frac{\partial L_{nl}}{\partial w_{ijc_ic_o}} = 2\left(1 - \|w_{c_o}\|\right)\left(-\frac{1}{2\|w_{c_o}\|}\right)\frac{\partial}{\partial w_{ijc_ic_o}}\left(\sum_{c_i=1}^{C_i}\sum_{i=1}^{F_h}\sum_{j=1}^{F_w} w_{ijc_ic_o}^2\right) \Rightarrow$$
$$\frac{\partial L_{nl}}{\partial w_{ijc_ic_o}} = 2\left(1 - \|w_{c_o}\|\right)\left(-\frac{1}{2\|w_{c_o}\|}\right)2w_{ijc_ic_o} \Rightarrow \tag{6.13}$$

$$\frac{\partial L_{nl}}{\partial w_{ijc_ic_o}} = 2w_{ijc_ic_o}\left(1 - \frac{1}{\|w_{c_o}\|}\right) \tag{6.14}$$

Comparing equations 6.10 and 6.14 we can see that effectively norm loss can be seen as an extension of the weight decay where the weight decay factor and its sign are regulated during training by the norm of the weight vector. This is explicitly visible from the overall update rule (combining equations 6.2, 6.8, 6.14):

$$\mathbf{w}^{new} = \mathbf{w} - \eta\lambda_{nl}2\left(1 - \frac{1}{\|w_{c_o}\|}\right)\mathbf{w} - \eta\frac{\partial L_{target}}{\partial \mathbf{w}} \tag{6.15}$$

6.4.2 Computational cost

For a convolutional layer with C_o filters of shape $F_h \times F_w \times C_i$, the computational cost of weight decay is two operations per weight, thus $2 \cdot C_o \cdot F_h \cdot F_w \cdot C_i$. For the norm loss it is $3 \cdot C_o \cdot (F_h \cdot F_w \cdot C_i) + C_o + 2 \cdot C_o \cdot F_h \cdot F_w \cdot C_i$. The overhead then is $3 \cdot C_o \cdot (F_h \cdot F_w \cdot C_i) + C_o$. The computational cost of a convolutional layer is $6 \cdot m \cdot C_o \cdot C_i \cdot F_h \cdot F_w \cdot I_h \cdot I_w$, where m, I_h, I_w are the number of images in a mini batch, and the input (to the layer) image height and width respectively. The computational overhead of the norm loss is orders of magnitude smaller than the computational cost of a convolutional layer with weight decay, for usual network and image input sizes.

6.5 Experiments

We evaluate our method on three well known benchmarks, i.e., CIFAR-10, CIFAR-100 and ImageNet2012. CIFAR-10 consists of 50K training and 10K test 32×32 natural images, divided into 10 classes. CIFAR-100 also consists of 50K training and 10K test natural images with the same resolution, but in this dataset they are divided into

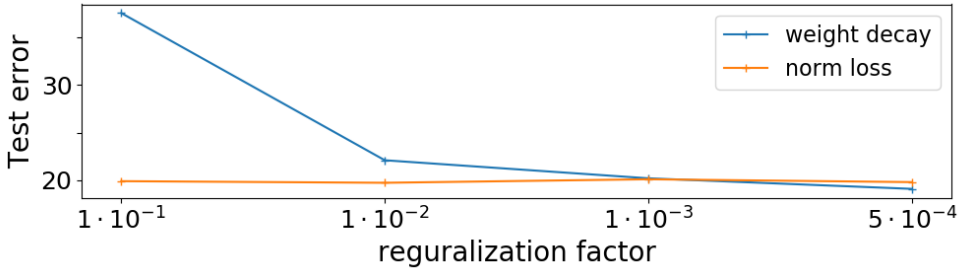


Figure 6.2: Test error of WRN-28-10 on CIFAR-100 with weight decay and norm loss for different regularization factors.

100 classes. For both CIFAR-10 and CIFAR-100 we evaluate using classification error on the designated test sets. The ImageNet 2012 is a large scale image recognition dataset. The train set consists of 1.281 million natural images of arbitrary resolution and aspect ratio. The images are divided into 1000 classes. The validation set consists of 50K images, i.e., 50 images per class. We evaluate our method on the validation set top 1 and top 5 error rates, as is common practice in the field.

We utilize two different state of the art architectures, namely the ResNet [127] and the wide ResNet (WRN) [437], since they are usual test cases for weight regularization methods [148, 149, 147, 17]. Both ResNet and WRN have been defined for many different sizes with different learning capacity. Unfortunately, training such big networks is very computationally expensive and thus we choose only one architecture per model. For all our three benchmarks we utilize the cross-entropy loss function as our target loss function (L_{target}).

The norm loss approach is compared with state of the art approaches, like weight decay (wd), weight normalization (WN)[311], projection based weight normalization (PBWN)[148], orthogonalization with Newton iteration (ONI)[147], orthogonal linear module (OLM)[149].

6.5.1 Regularization factor

With the first experiment we evaluate the effect of the regularization factor λ_{nl} on the training. As a benchmark we use the CIFAR-100 dataset. We train for four different values of λ_{nl} and plot our results in Figure 6.2. It is apparent that the effect of λ_{nl} on the training is much smaller than that of λ_{wd} in the case of weight decay. We believe that this results from the regularization of the λ_{nl} factor discussed before (equation 6.15).

Figure 6.3 shows the evolution of the train cross entropy (per batch) during train-

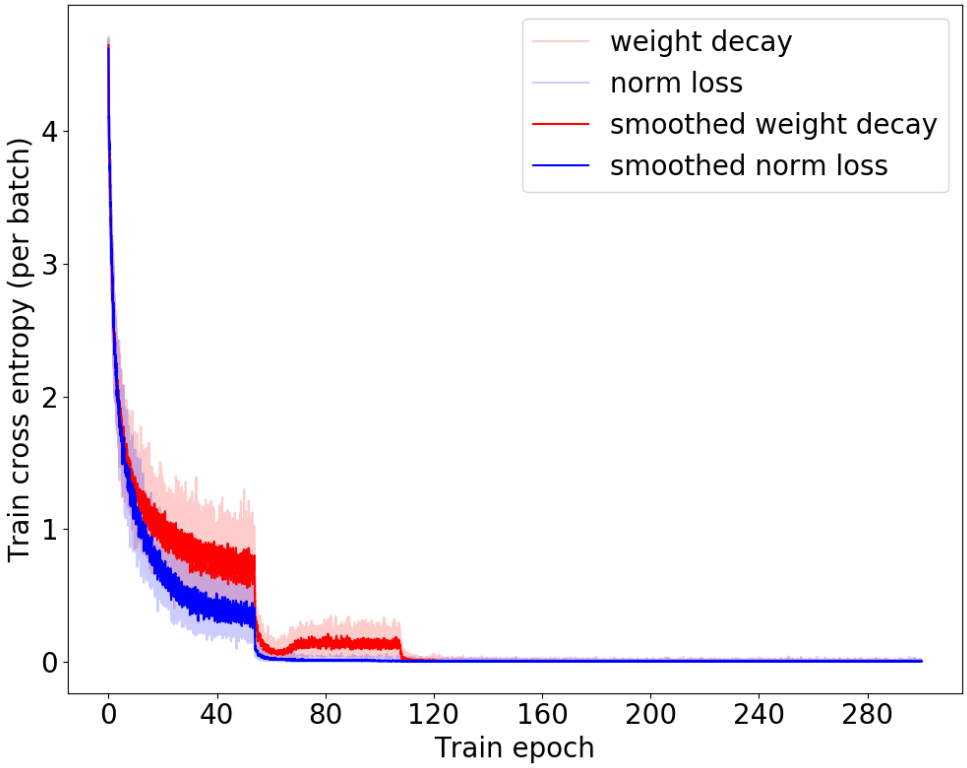


Figure 6.3: Evolution of cross entropy during training on CIFAR-100 for weight decay and norm loss. The regularization factor for both runs is 5e-4. The shaded lines are the true lines, whilst the non-shaded are smoothed version of the original (averaged over 19 steps) for more comprehensive visualization.

ing, for the case of $\lambda = 5 \cdot 10^{-4}$. We can see that with the norm loss, the networks are being trained faster than with weight decay, even in the case where the weight decay training has marginally higher accuracy in the end of training (see Figure 6.2).

6.5.2 Batch size

The next experiment is to test the training behavior for different batch sizes. We train networks with both weight decay and our method for batch sizes $\{8, 16, 32, 64, 128\}$ on CIFAR-100. The performance of the trained networks on the test set can be seen in Figure 6.4. We can see that the norm loss has more steady behavior than the weight decay. We can still see some dependence on the batch size, which is expected since our networks utilize batch normalization. Moreover, although for most batch sizes

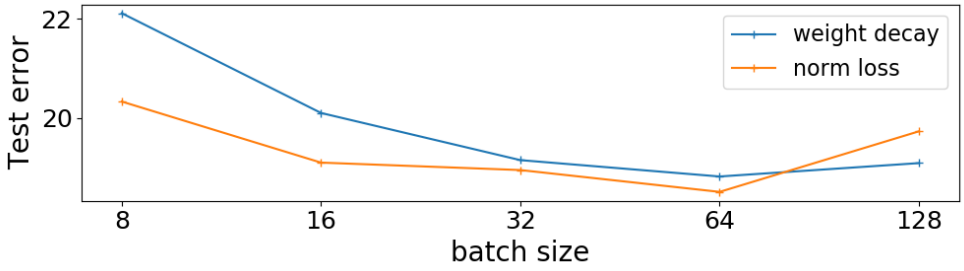


Figure 6.4: Test error of WRN-28-10 on CIFAR-100 with weight decay and norm loss for different batch sizes.

tested the networks trained with norm loss show better performance, only for batch size 128 the opposite is true. Finally, we can see that the highest overall performance for both methods is achieved by the networks trained with batch size 64.

6.5.3 CIFAR-10

On this dataset we train on the ResNet110 and WRN-28-10. As is common practice [127, 437] we train the networks using SGD with momentum of 0.9 and a batch size of 128. The initial learning rate is set to 0.1. We follow a learning schedule close to the one used in [437]. When training the WRN, we drop the learning rate by a factor of 5 at epochs 53, 107, 230 and we train for a total of 300 epochs. For the ResNet we train for 164 epochs and reduce the learning rate at epochs 82 and 123. We follow the standard preprocessing for training as in [127, 437, 17], i.e., padding each train example by four pixels and getting a random 32×32 crop. Both train and test sets are mean and std normalized [437]. We train five times and report the mean as well as the best run. For both networks, the regularization factor λ_{nl} is set to 0.01 while the weight decay factor λ_{wd} is set to 10^{-4} for the ResNet and $5 \cdot 10^{-4}$ for the WRN, as in the original papers [127, 437]. The results can be seen in Table 6.1.

When training the ResNet110 training with norm loss outperforms all other regularization methods we are aware of, that tested on the same network [147, 311, 148]. Norm loss also outperforms the reported accuracy of the SRIP regularization method of [17]. Due to large difference with our and their baseline performance, we do not consider it a valid comparison and thus their result is omitted from Table 6.1.

When training the WRN-28-10 we observe a big decrease in performance over our benchmark. Figures 6.5, 6.6 show the training process of the WRN-28-10 on the CIFAR-10 with weight decay and norm loss. We can see that even in this case, where the final performance of weight decay is better, the network that utilizes the norm loss

Table 6.1: Performance of different methods on CIFAR-10 test set for the ResNet110 and WRN-28-10. In parentheses are the mean or median over some runs given by the authors of the respective papers. Outside parentheses the best accuracy (if shown by the authors). For methods denoted by *, the performance is given in the respective papers.

model	reg. method	error
ResNet110	wd	6.32 (6.568)
ResNet110 [127]*	wd	6.43 (6.61)
ResNet110 [311]*	WN	- (7.56)
ResNet110 [148]*	PBWN	- (6.27)
ResNet110 [147]*	ONI	- (6.56)
ResNet110 (Ours)	nl	5.9 (5.996)
WRN-28-10	wd	3.9 (3.966)
WRN-28-10 [437]*	wd	- (3.89)
WRN-28-10 [149]*	OLM	- (3.73)
WRN-28-10 [149]*	OLM-L1	- (3.82)
WRN-28-10 (Ours)	nl	4.47 (4.662)

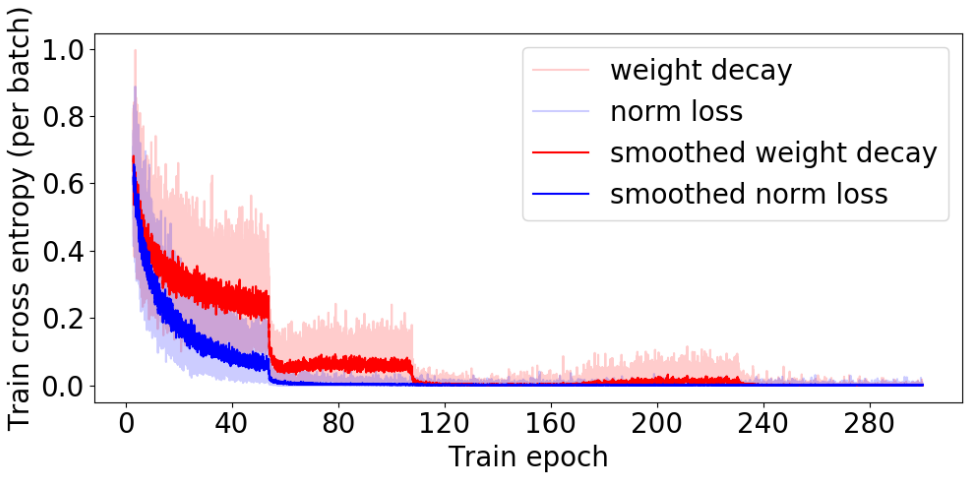


Figure 6.5: Evolution of cross entropy during training on CIFAR-10 for weight decay and norm loss for the WRN-28-10. The shaded lines are the true lines, whilst the non-shaded are smoothed version of the original (averaged over 19 steps) for more comprehensive visualization.

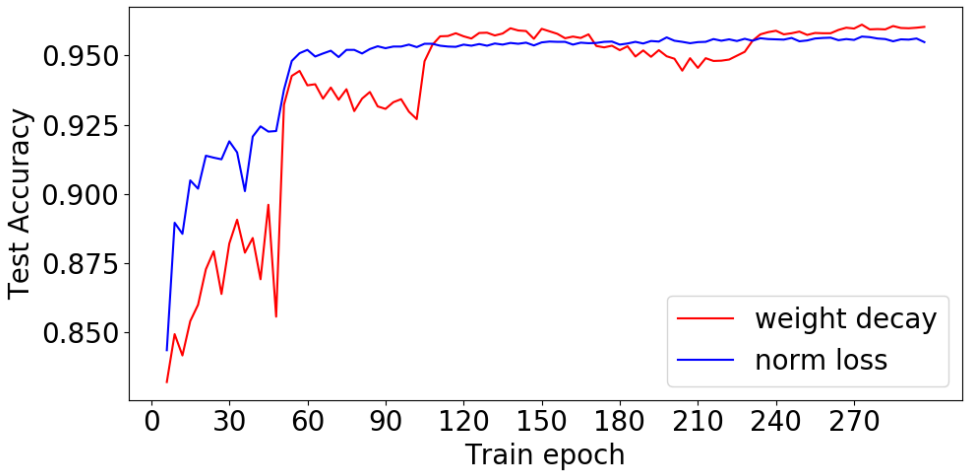


Figure 6.6: Evolution of test accuracy during training on CIFAR-10 for weight decay and norm loss for the WRN-28-10.

is converging much faster and shows more stable behavior. More experimentation is needed to understand why in this specific scenario the final performance is worse than the baseline. For example, a schema like the one used in [17] could be used, where from a certain epoch on the regularization is minimized or even dropped completely.

In order to evaluate the computational overhead of our method, we report that training the ResNet110 on CIFAR-10 with weight decay takes, on average, 4.56 hours while training the same network with norm loss takes, on average, 4.79 hours. All aforementioned experiments ran on an Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz and an NVIDIA GTX 1080Ti graphics card.

6.5.4 CIFAR-100

As with CIFAR-10, for this section we use the same hyper parameters as in [437], with a slightly different learning rate schedule. We train the WRN using a batch size of 64 instead of 128, because it results in better performance even for the baseline method. The regularization factor λ_{nl} was set to 10^{-3} and the λ_{wd} to $5 \cdot 10^{-4}$. We train for a total of 448 epochs and reduce the learning rate by a factor of 5 in epochs 77, 153 and 307. We train the ResNet for 300 epochs and drop the learning rate at epochs 53, 107, 230. We train 5 times and report the accuracy of the mean and best run. The results are shown in Table 6.2. We also trained with the learning rate scheduler used for ResNet110 on CIFAR-10, i.e., 164 epochs and reduce the learning rate at epochs 82 and 123, but the aforementioned scheduler produced results for the baseline that

Table 6.2: Performance of different methods on CIFAR-100 test set for the ResNet110 and WRN-28-10. In parentheses are the mean or median over some runs given by the authors of the respective papers. Outside parentheses the best accuracy (if shown by the authors). For methods denoted by *, the performance is given in the respective papers.

model	reg. method	error
ResNet110	wd	27.9 (28.398)
ResNet110 [311]*	WN	- (28.38)
ResNet110 [148]*	PBWN	- (27.03)
ResNet110 (Ours)	nl	26.24 (26.526)
WRN-28-10	wd	18.85 (19.138)
WRN-28-10 [437]*	wd	- (18.85)
WRN-28-10 [149]*	OLM	- (18.76)
WRN-28-10 [149]*	OLM-L1	- (18.61)
WRN-28-10 (Ours)	nl	18.57 (18.648)

match the literature and thus is reported on Table 6.2. For clarity we also report the results using the CIFAR-10 scheduler: Average error for the weight decay is 28.094 with a best run of 27.56 whilst for the norm loss the average error is 25.878 with a best run of 25.2. For all ResNet experiments, the regularization factor λ_{nl} was set to 10^{-2} and the λ_{wd} to 10^{-4}.

The norm loss manages to produce better performance than most methods for both ResNet110 and WRN-28-10. Although in the case of CIFAR-10 we managed to outperform the method developed in [17] (results were omitted from the tables due to large difference of their and our baseline performance), in the case of CIFAR-100 their methods outperform our own. For the same reason as before their results are omitted (25.42% vs 27.49% accuracy for the baseline ResNet110). It should be noted that they used a different optimizer for their experiments, the Adam optimizer [175].

6.5.5 ImageNet

For the ImageNet dataset, we utilize the ResNet50 architecture to evaluate our method. We use the same data augmentation and hyper parameters (for batch normalization, dropout rates, etc.) as the implementation of [148] on GitHub[1]. We train using SGD with Nesterov momentum of 0.9. The learning rate is initialized at 0.1 and divided by 10 every 30 epochs. We train with a batch size of 128, whilst due to GPU memory

[1]https://github.com/huangleiBuaa/NormProjection

Table 6.3: Top-1 and Top-5 error rates of different methods on ImageNet validation set for the ResNet50. For methods denoted by *, the performance is given in the respective papers.

model	reg. method	Top-1 error	Top-5 error
ResNet50	wd	25.29	7.86
ResNet50 [147]*	wd	23.85	-
ResNet50 [147]*	ONI	23.30	-
ResNet50 (Ours)	nl	24.34	7.44

limitations, the batch normalization parameters are trained on half, i.e., 64 examples. The regularization factor for both methods, i.e., $\lambda_{nl}, \lambda_{wd}$ is set to 10^{-4}. The top-1 and top-5 error rates[2] of the networks on the ImageNet 2012 validation set are shown in Table 6.3.

The first two rows show the performance of the baseline method as reported in literature [147] and our attempt at reproducing it. As it can be seen, there is a large difference between the results as our error is much larger than the one reported in literature. The origin of this is currently not clear to us and therefore a direct comparison of performance numbers is not warranted. Nonetheless, we can observe that for our implementations the norm loss approach improves both TOP-1 and TOP-5 performances substantially over weight decay.

6.6 Conclusions

In this chapter, a new soft-regularization method is proposed, that tries to guide the weight vector towards the Oblique manifold. It accomplishes that by utilizing the proposed norm-loss function as regularization during the neural network training.

We evaluate the norm loss on standard benchmarks, i.e., CIFAR-10, CIFAR-100 and ImageNet 2012 and compare the performance to the state of the art regularization methods. The Norm loss approach accelerates the convergence speed of networks and leads to a more robust, i.e., less sensitive training process with regard to the hyperparameters settings for the regularization factor and the batch size. While having a negligible computational overhead over weight decay during training, and no extra computational cost during inference, our method has comparable performance to the state of the art and some cases even higher, while only for one setup norm loss is

[2]Top-1 error is 100 - classification accuracy. Top-5 error counts a correct classification if the ground truth label is in the top-5 predictions of the network.

under performing, i.e., WRN-28-10 on CIFAR-10.

7 Conclusions

7.1 Conclusions

In this thesis, we explore the application of computer vision methods on high dimensional data. With high dimensional data we define data that have more than two dimensions. More specifically, we explore whether computer vision inspired approaches are capable of representing computational fluid dynamics simulation output.

In Chapter 2 we presented an extensive overview of the methodologies, data sources and applications whose subject is high dimensional vision data. The aim of this chapter is to find common practices between methodologies that, at a first glance, seem disconnected but have in common the high dimensionality of the data they process and thus answer **research question 1**: Overall we identified four common data sources, namely 2D videos, RGB-D images and videos, and 3D models such as CAD models or point clouds. Moreover, we identified common characteristics of methodologies regardless the data they are applied on and we identified two main categories of high dimensionality, high physical dimensions and large number of information per physical location. Finally, we concluded that although hand crafted feature based approaches are outperformed by deep learning based approaches, they can provide complementary information and increase the overall performance of an approach.

In Chapter 3, we constructed a large scale dataset and proposed several approaches for dealing with the high dimensionality of the data. This dataset is used as a benchmark to help us answer **research question 2**. The experiments conducted showed that deep learning approaches are well capable of representing CFD simulation output and moreover, how to better utilize our model resources, i.e., number of trainable parameters, hardware computational capabilities and available memory. Our best models are able to accurately, i.e., with about 3% accuracy, predict forces applied on

a shape not visible to them, by leveraging flow data. Meanwhile, the encoded representation is enough to reconstruct the input in a large extent leading us to believe that it still encodes most of the information of the original flow field.

In Chapter 4 we answered **research question 3**. We implemented hand crafted feature based approaches and evaluated whether they are capable of representing flow field data and how they compare to deep learning approaches. Overall we made three tests, (i) which hand-crafted feature produces the most accurate results and with what kind of sampling (i.e., dense or which detector), (ii) how does the performance compare to deep learning approaches and (iii) which approach maintains its performance better as the training set size reduces. The experimental results showed interesting behavior. The dense sampling of SIFT [225, 226] features produced the best performance out of all setups based on hand-crafted features tested. Interestingly, although in terms of $RMSE$ the SIFT approach was outperformed by the deep learning approaches, it was able to outperform them in terms of R^2. As we showed, deep learning approaches and hand crafted feature based approaches show different behavior. For most test examples, the SIFT based approach had smaller error than the deep learning one, but it also produced much larger errors than the deep learning approaches at the extreme (bringing the $RMSE$ high). This finding is strengthening the observation that hand-crafted feature based approaches can provide complementary information to the deep learning approaches.

In Chapter 5 we answered **research question 4**. By taking advantage of the vector field representation we are able to extract angle information from the input and the kernel of a convolutional layer and subsequently build rotation invariant and equivariant convolutional layers and networks. Our experiments showed that this approach produces similar classification performance to the state of the art, while having better angle prediction with a much lower computational cost.

Finally, in Chapter 6 the norm loss was proposed, answering **research question 5**. The norm loss is a soft weight regularization method for deep neural networks and it aims to keep the norm of the kernels of a convolutional layer close to one. The experimental results suggest that networks that utilize the norm Loss during training converge much faster than the equivalent that utilized weight decay. Moreover, in most cases, networks trained with the norm loss exhibit state of the art performance.

7.2 Limitations

Although the research presented here has revealed interesting information, it has also shown many limitations of the approaches proposed and tested. As the main data source for this thesis has been CFD simulation output we deem necessary to start with

this application in mind. The major limiting factor in extracting meaningful information from CFD simulation data, is its availability. CFD simulations can even produce terabytes of data. Nonetheless, most information usually comes from the same example. As a result, we have huge amounts of data, split into very few examples. This is the opposite from the ideal situation, where many examples exist and thus correlations can be found. The main reason for this limitation is the time required for CFD simulations to converge. A complicated simulation might even take a month to converge on a cluster with thousands cores. Moreover, there are many hyper parameters as well as different algorithms for obtaining CFD simulation output. With many of these parameters, such as the grid on which the flow field is computed, the result can vary significantly. Having a model capable of representing simulations produced in a different manner increases the complexity and amount of data required even further.

Regarding deep learning models, we can identify some limitations as well. A common approach to produce general high performing models is to augment the training data. By applying slight augmentations on the data during training, effectively increases the train data size by introducing new variations of the same subject, i.e., the content of the image. The resulting models become more robust to certain types and amount of noise. If a flow field is augmented using the popular approaches such as affine transformations, it will probably not be an approximation of a solution to Navier-Stokes equations anymore and more importantly it will not maintain the same label. Thus, one must be careful in how data is augmented. Moreover, many patterns appear in small scales in the flow. For the models to fit in GPU memory the input resolution has to be relatively low which may result in the disappearance of many important characteristics. Nonetheless, large scale patterns, like vortices and reverse flow still exist in the simulations examples that we utilized which made the training of models possible.

In the last two chapters, new deep learning approaches are proposed. In Chapter 5 a new operator is proposed, that utilizes the vector field representation to measure the angle between a convolutional kernel and an input signal. Utilizing this operator in deep neural networks, although shows great potential, it has several limitations. The most important one is that for most applications, the input is scalar, e.g. RGB images. Moreover, due to the $arctan$ function, during back propagation, numeric instabilities can occur which forces the usage of thresholding and masking the output. In our experience, depending on the application, the network size, the learning rates and other parameters, the most effective value of this threshold changes. This introduces another hyper parameter to the many existing in deep learning approaches. Moreover, as we saw with the current implementation, utilizing this operator on GPUs increases the computation time by a large margin, which may significantly reduce one of the main advantages of the method - lower computational requirements.

7.3 Future work

The Clifford convolution operator was implemented for two dimensional fields, as an initial case study. Most CFD simulation outputs though, have three spatial dimensions. Extending the operator to calculate solid angles might be very beneficial for minimizing the required number of free parameters in a deep learning model. Interestingly, the potential benefit of calculating solid angles might be even higher than that of the two dimensions, since the amount of convolutions needed for a max-pooling operation over solid angle increases exponentially with the number of dimensions.

This research is one of the early works on deep learning applied on CFD simulation output. One of the main limitations is the large amount of time required to produce simulations. Thus, a feasible approach in applying deep learning on more complicated simulations is transferring models. To be more precise, a model could be trained on a large scale CFD dataset that requires feasible amount of time to be created. Would this model then be applicable, with minor adjustments i.e., fine tuning, on a small dataset of complex and time consuming CFD simulations?

From a deep learning perspective, there are still many open questions. For example, one question is how to better utilize the extra dimensions of higher than two dimensional images. When a temporal dimension is included, it is not certain yet which approach is more effective. Is there a certain approach to handling the temporal dimension that suits better, or will it always depend on the application? For the static world there are similar questions that are important to research if greater performing or efficiency is required. For example, the projection to lower dimensionality, seems to be performing better than having three dimensional kernels for low computational models. Increasing the capacity of networks with three dimensional kernels, coupled with large and diverse datasets seems to perform better than the 2D projections. In many applications the computational and memory budgets are limited. It is still unclear at which point does a more complex network with high dimensional kernels is a more appropriate choice. These are questions that are becoming more relevant as computational demand of deep learning approaches seem to grow and mobile applications seem to have increasing demand.

APPENDICIES

A Clifford Convolution gradients calculations

Bellow are all the calculations of all the equations used for the back-propagation algorithm. For all formulas, for the sake of simplicity we use the following notation:

$$\begin{aligned} &p_w : \text{weight position, } (i, j, c_{in}, c_{out}) \\ &p_o : \text{output position, } (i', j', c_{out}) \\ &p_{in} : \text{input position, } (i'', j'', c_{in}) \end{aligned} \tag{1}$$

During the backward pass, every arrow in Figure 5.2 should return the respective derivatives. We start the computations from step C_5 and work our way to step C_1.

C_5:
The final output $\vec{O}^l_{p_o}$ is given by:

$$\vec{O}^l_{p_o} = (O^l_{0,p_o}, O^l_{1,p_o}) = (O^l_{p_o} \cdot \cos(\phi_{p_o}), O^l_{p_o} \cdot \sin(\phi_{p_o})) \tag{2}$$

Following the chain rule:

$$\begin{aligned} &\frac{\partial E}{\partial O^l_{p_o}} = \frac{\partial E}{\partial O^l_{0,p_o}} \frac{\partial O^l_{0,p_o}}{\partial O^l_{p_o}} + \frac{\partial E}{\partial O^l_{1,p_o}} \frac{\partial O^l_{1,p_o}}{\partial O^l_{p_o}} \Rightarrow \\ &\frac{\partial E}{\partial O^l_{p_o}} = \delta^{l,C_5}_{0,p_o} \cdot \cos(\phi_{p_o}) + \delta^{l,C_5}_{1,p_o} \cdot \sin(\phi_{p_o}) \end{aligned} \tag{3}$$

$$\begin{aligned}\frac{\partial E}{\partial \phi^l_{p_o}} &= \frac{\partial E}{\partial O^l_{0,p_o}}\frac{\partial O^l_{0,p_o}}{\partial \phi^l_{p_o}} + \frac{\partial E}{\partial O^l_{1,p_o}}\frac{\partial O^l_{1,p_o}}{\partial \phi^l_{p_o}} \Rightarrow \\ \frac{\partial E}{\partial \phi^l_{p_o}} &= -\delta^{l,C_5}_{0,p_o} \cdot O^l_{0,p_o} \cdot \sin(\phi_{p_o}) + \delta^{l,C_5}_{1,p_o} \cdot O^l_{1,p_o} \cdot \cos(\phi_{p_o})\end{aligned} \tag{4}$$

C_4:

$$O^l_{p_o} = \sum_i \sum_j \sum_{c_{in}} \vec{W}^l_{\phi\ p_o p_w} \cdot \vec{O}^{l-1}_{p_{in}} = \sum_i \sum_j \sum_{c_{in}} \sum_k W^l_{\phi\ p_o p_w,k} \cdot O^{l-1}_{p_{in},k} \tag{5}$$

where k indexes the x and y coordinates of the vectors.

$$\frac{\partial E}{\partial W^l_{\phi\ p_o p_w,k}} = \frac{\partial E}{\partial O^l_{p_o}}\frac{\partial O^l_{p_o}}{\partial W^l_{\phi\ p_o p_w,k}} \tag{6}$$

Since for every output pixel we calculate a different angle ($\phi^l_{p_o}$), there is a different set of weights associated with every output pixel, $\vec{W}^l_{\phi\ p_o}$.

$$\frac{\partial E}{\partial W^l_{\phi\ p_o p_w,k}} = \frac{\partial E}{\partial O^l_{p_o}}\frac{\partial O^l_{p_o}}{\partial W^l_{\phi\ p_o p_w,k}} = \sum_{p_o \in P_\phi} \delta^{l,C_4}_{p_o} \cdot O^{l-1}_{p_{in},k} \tag{7}$$

$$\frac{\partial E}{\partial O^{l-1}_{p_{in},k}} = \sum_i \sum_j \sum_{c_{out}} \frac{\partial E}{\partial O^l_{p_o}}\frac{\partial O^l_{p_o}}{\partial O^{l-1}_{p_{in},k}} = \sum_i \sum_j \sum_{c_{out}} \delta^{l4}_{p_o} \cdot W^l_{\phi\ p_o p_w,k} \tag{8}$$

where the ϕ in $W^l_{\phi\ p_o p_w,k}$ refers to the predefined angles ($\in [0, 2\pi)$) used for the specific output position.

C_3:

$$\vec{W}^l_{\phi\ p_o} = rotation(\vec{W}^l_{p_o}, \phi^l_{p_o}) \tag{9}$$

As mentioned above, there is a seperate set of weights for every output pixel. The gradients from all sets of weights are calculated and then added to the orginal weight vector. For simplicity the index of p_o on the weight vectors is omited. From equation 9 we see that there are two sets of gradients need to be computed, $\frac{\partial E}{\partial \vec{W}^l}$ and $\frac{\partial E}{\partial \phi^l_{p_o}}$.
Since $\vec{W}^l_\phi$ are the rotated $\vec{W}^l$, we set:

$$\frac{\partial E}{\partial \vec{W}^l} = rotation(\frac{\partial E}{\partial \vec{W}^l_\phi}, -\phi^l_{p_o}) \tag{10}$$

For the second set we have:

$$\frac{\partial E}{\partial \phi^l_{p_o}} = \sum_i \sum_j \sum_{c_{in}} \frac{\partial E}{\partial \vec{W}^l_{\phi\ p_w}} \frac{\partial \vec{W}^l_{\phi\ p_w}}{\partial \phi^l_{p_o}} = \sum_i \sum_j \sum_{c_{in}} \delta^{l,C_3}_{\vec{W}_{\phi\ p_w}} \cdot \frac{\partial \vec{W}^l_{\phi\ p_w}}{\partial \phi^l_{p_o}} \tag{11}$$

We have two options for calculating $\frac{\partial \vec{W}^l_{\phi\ p_w}}{\partial \phi^l_{p_o}}$. The first is to differentiate the bilinear interpolation (at least at the points that it is differentiable), or use the precalculated rotated weights. Let θ be the quantized calculated angle ϕ. Then:

$$\frac{\partial \vec{W}^l_{\theta\ p_w}}{\partial \phi^l_{p_o}} = \frac{\vec{W}^l_{\theta+1\ p_w} - \vec{W}^l_{\theta-1\ p_w}}{2\frac{2\pi}{B}} \tag{12}$$

where B is the number of predefined orientations. Have in mind that the rotation above ($\vec{W}^l_{\theta+1\ p_w}$) considers only plane rotation after acquiring the in-place vector rotation $\vec{W}_\phi$.

C_2:
On all equations related to C_2, like C_3, the indexes p_o on weight vectors are omitted.

$$\vec{W}^l_{\phi\ p_w} = \vec{W}^l_{p_w} \cdot \begin{pmatrix} \cos\phi^l_{p_o} & \sin\phi^l_{p_o} \\ -\sin\phi^l_{p_o} & \cos\phi^l_{p_o} \end{pmatrix} \Leftrightarrow \begin{cases} \vec{W}^l_{\phi\ 0,p_w} = \vec{W}^l_{0,p_w}\cos\phi^l_{p_o} - \vec{W}^l_{1,p_w}\sin\phi^l_{p_o} \\ \vec{W}^l_{\phi\ 1,p_w} = \vec{W}^l_{0,p_w}\sin\phi^l_{p_o} + \vec{W}^l_{1,p_w}\cos\phi^l_{p_o} \end{cases} \tag{13}$$

As with C_3, there are two set of gradients to be calculated, specifically $\frac{\partial E}{\partial \vec{W}^l_{p_w}}$ and $\frac{\partial E}{\partial \phi^l_{p_o}}$ where the first represents two sets, one for each direction of the vectors in $\vec{W}^l$.

$$\frac{\partial E}{\partial \vec{W}^l_{p_w}} = (\frac{\partial E}{\partial W^l_{0,p_w}}, \frac{\partial E}{\partial W^l_{1,p_w}}) \tag{14}$$

For the two components we get:

$$\begin{aligned} \frac{\partial E}{\partial W^l_{0,p_w}} &= (\frac{\partial E}{\partial \dot{W}^l_{\phi\ 0,p_w}} \frac{\partial \dot{W}^l_{\phi\ 0,p_w}}{\partial W^l_{0,p_w}} + \frac{\partial E}{\partial \dot{W}^l_{\phi\ 1,p_w}} \frac{\partial \dot{W}^l_{\phi\ 1,p_w}}{\partial W^l_{0,p_w}}) \\ &= (\frac{\partial E}{\partial \dot{W}^l_{\phi\ 0,p_w}} \cos\phi^l_{p_o} + \frac{\partial E}{\partial \dot{W}^l_{\phi\ 1,p_w}} \sin\phi^l_{p_o}) \\ \frac{\partial E}{\partial W^l_{1,p_w}} &= (\frac{\partial E}{\partial \dot{W}^l_{\phi\ 0,p_w}} \frac{\partial \dot{W}^l_{\phi\ 0,p_w}}{\partial W^l_{1,p_w}} + \frac{\partial E}{\partial \dot{W}^l_{\phi\ 1,p_w}} \frac{\partial \dot{W}^l_{\phi\ 1,p_w}}{\partial W^l_{1,p_w}}) \\ &= (-\frac{\partial E}{\partial \dot{W}^l_{\phi\ 0,p_w}} \sin\phi^l_{p_o} + \frac{\partial E}{\partial \dot{W}^l_{\phi\ 1,p_w}} \cos\phi^l_{p_o}) \end{aligned} \tag{15}$$

$$
\begin{aligned}
(14),(15) \rightarrow \frac{\partial E}{\partial \vec{W}^l_{p_w}} &= \frac{\partial E}{\partial \vec{\dot{W}}^l_{\phi\ p_w}} \cdot \begin{pmatrix} \cos(-\phi^l_{p_o}) & \sin(-\phi^l_{p_o}) \\ -\sin(-\phi^l_{p_o}) & \cos(-\phi^l_{p_o}) \end{pmatrix} \\
&= \vec{\delta}^{l,C_2}_{\phi\ p_w} \cdot \begin{pmatrix} \cos(-\phi^l_{p_o}) & \sin(-\phi^l_{p_o}) \\ -\sin(-\phi^l_{p_o}) & \cos(-\phi^l_{p_o}) \end{pmatrix}
\end{aligned} \tag{16}
$$

$$
\frac{\partial E}{\partial \phi^l_{p_o}} = \sum_i \sum_j \sum_{c_{in}} \left(\frac{\partial E}{\partial \dot{W}^l_{0,p_w}} \frac{\partial \dot{W}^l_{0,p_w}}{\partial \phi^l_{p_o}} + \frac{\partial E}{\partial \dot{W}^l_{1,p_w}} \frac{\partial \dot{W}^l_{1,p_w}}{\partial \phi^l_{p_o}} \right) \tag{17}
$$

$$
\begin{aligned}
\frac{\partial \dot{W}^l_{0,p_w}}{\partial \phi^l_{p_o}} &= -W^l_{0,p_w} \sin \phi^l_{p_o} - W^l_{1,p_w} \cos \phi^l_{p_o} = -\dot{W}^l_{1,p_w} \\
\frac{\partial \dot{W}^l_{1,p_w}}{\partial \phi^l_{p_o}} &= W^l_{0,p_w} \cos \phi^l_{p_o} - W^l_{1,p_w} \sin \phi^l_{p_o} = \dot{W}^l_{0,p_w}
\end{aligned} \tag{18}
$$

$$
\begin{aligned}
(17),(18) \rightarrow \frac{\partial E}{\partial \phi^l_{p_o}} &= \sum_i \sum_j \sum_{c_{in}} \left(\frac{\partial E}{\partial \dot{W}^l_{0,p_w}} (-\dot{W}^l_{1,p_w}) + \frac{\partial E}{\partial \dot{W}^l_{1,p_w}} \dot{W}^l_{0,p_w} \right) \\
&= \sum_i \sum_j \sum_{c_{in}} (\delta^{l,C_2}_{0,p_w} (-\dot{W}^l_{1,p_w}) + \delta^{l,C_2}_{1,p_w} \dot{W}^l_{0,p_w})
\end{aligned} \tag{19}
$$

In our implementation C_2 and C_3 are considered as one operation. Moreover, we keep in memory the rotated weights $\vec{W}_\phi$ and not $\vec{\dot{W}}_\phi$. Fortunately, we can approximate the gradients as of the separate operations as following:

$$
\vec{W}^l_\phi = vector\_field\_rotation(\vec{W}^l, \phi^l_{p_o}) \tag{20}
$$

Similarly with C_3:

$$
\frac{\partial E}{\partial \vec{W}^l} = vector\_field\_rotation(\frac{\partial E}{\partial \vec{W}^l_\phi}, -\phi^l_{p_o}) \tag{21}
$$

$$
\frac{\partial \vec{W}^l_{\theta\ p_w}}{\partial \phi^l_{p_o}} = \frac{\vec{W}^l_{\theta+1\ p_w} - \vec{W}^l_{\theta-1\ p_w}}{2\frac{2\pi}{B}} \tag{22}
$$

Unlike C_3, here the rotated $\vec{W}^l_{\theta+1\ p_w}$ are the complete vector field rotation with angle $\theta + 1$ from the original $\vec{W}$.

C_1:
Let:

$$tan^l_{p_o} = \frac{conv_0^{l_{p_o}}}{conv_2^{l_{p_o}}} \tag{23}$$

then:

$$(5.2), (5.1) \rightarrow \phi^l_{p_o} = \arctan\left(\frac{conv_2}{conv_0}\right) = \arctan\left(tan^l_{p_o}\right) \tag{24}$$

$$\frac{\partial E}{\partial tan^l_{p_o}} = \frac{\partial E}{\partial \phi^l_{p_o}} \frac{\partial \phi^l_{p_o}}{\partial tan^l_{p_o}} = \frac{\partial E}{\partial \phi^l_{p_o}} \frac{1}{1 + (tan^l_{p_o})^2} \tag{25}$$

$$\begin{aligned} \frac{\partial E}{\partial conv_0} &= \frac{\partial E}{\partial tan^l_{p_o}} \frac{\partial tan^l_{p_o}}{\partial conv_0} = \frac{\partial E}{\partial tan^l_{p_o}} (-\frac{conv_2}{conv_0^2}) \overset{25}{\Rightarrow} \\ \frac{\partial E}{\partial conv_0} &= \frac{\partial E}{\partial \phi^l_{p_o}} \frac{1}{1 + (tan^l_{p_o})^2} (-\frac{conv_2}{conv_0^2}) = -\frac{\partial E}{\partial \phi^l_{p_o}} \frac{conv_2}{conv_0^2 + conv_2^2} \end{aligned} \tag{26}$$

$$\begin{aligned} \frac{\partial E}{\partial conv_2} &= \frac{\partial E}{\partial tan^l_{p_o}} \frac{\partial tan^l_{p_o}}{\partial conv_2} = \frac{\partial E}{\partial tan^l_{p_o}} \frac{1}{conv_0} \overset{25}{\Rightarrow} \\ \frac{\partial E}{\partial conv_2} &= \frac{\partial E}{\partial \phi^l_{p_o}} \frac{1}{1 + (tan^l_{p_o})^2} \frac{1}{conv_0} = \frac{\partial E}{\partial \phi^l_{p_o}} \frac{conv_0}{conv_0^2 + conv_2^2} \end{aligned} \tag{27}$$

From Equation 5.2 we see that $conv_0$ is the conventional convolutional operation, meaning that the derivatives are the standard derivatives used in all CNN works. For $conv_2$ we have:

$$\begin{aligned} \frac{\partial E}{\partial W^l_{0,p_w}} &= \sum_{i'} \sum_{j'} \frac{\partial E}{\partial conv_2} \frac{\partial conv_2}{\partial W^l_{0,p_w}} = \sum_{i'} \sum_{j'} \frac{\partial E}{\partial conv_2} O^{l-1}_{1,p_{in}} \\ \frac{\partial E}{\partial W^l_{1,p_w}} &= \sum_{i'} \sum_{j'} \frac{\partial E}{\partial conv_2} \frac{\partial conv_2}{\partial W^l_{1,p_w}} = \sum_{i'} \sum_{j'} \frac{\partial E}{\partial conv_2} (-O^{l-1}_{0,p_{in}}) \end{aligned} \tag{28}$$

Similarly:

$$\begin{aligned} \frac{\partial E}{\partial O^{l-1}_{0,p_{in}}} &= \sum_{i} \sum_{j} \frac{\partial E}{\partial conv_2} \frac{\partial conv_2}{\partial O^{l-1}_{0,p_{in}}} = \sum_{i} \sum_{j} \frac{\partial E}{\partial conv_2} (-W^l_{1,p_w}) \\ \frac{\partial E}{\partial O^{l-1}_{1,p_{in}}} &= \sum_{i} \sum_{j} \frac{\partial E}{\partial conv_2} \frac{\partial conv_2}{\partial O^{l-1}_{1,p_{in}}} = \sum_{i} \sum_{j} \frac{\partial E}{\partial conv_2} W^l_{0,p_w} \end{aligned} \tag{29}$$

For each output pixel a separate weight vector was calculated and thus different gradients as well, i.e., $\left(\frac{\partial E}{\partial \vec{W}}\right)_{p_o}$. The final result is given by adding the $\left(\frac{\partial E}{\partial \vec{W}}\right)_{p_o}$ for all p_o.

B Table of abbreviations

Abbreviation	Explanation
2D	two dimensions/dimensional
3D	three dimensions/dimensional
3DBRIEF	3D BRIEF
3DLBP	3D LBP
3DORB	3D ORB
3DSC	3D SC
4D	four dimensions/dimensional
Adam	adaptive moment estimation
AE	auto-encoder
AGAST	adaptive and generic accelerated segment test
AlexNet	Alex Network
AMT	Amazon mechanical turk
ANN	artificial neural network
APC	Amazon picking challenge
API	application programming interface
avacc	meanIU
B3DO	Berkley 3D Objects
BN	batch normalization
BoF	bag of features
BoW	bag of words
BPTT	back propagation through time
BRAND	binary robust appearance and normal descriptor
BRIEF	binary robust independent elementary features
BRISK	binary robust invariant scale keypoint
BRoPH	binary rotational projection histogram
C3D	convolutional 3D
CAD	computer-aided design
CAE	convolutional AE

Abbreviation	Explanation
CBCT	cone beam computed tomography
cc	Clifford convolution
CFD	computational fluid dynamics
CFN	convolutional fusion network
Charades-STA	Charades sentence temporal annotations
CHMM	coupled HMM
CIFAR	Canadian institute for advanced research
CL	convolutional layer
clacc	classification accuracy
CNN	convolutional neural network
COCO	common objects in context
convGRBM	convolutional GRBM
CPU	central processing unit
CRF	conditional random field
CT	computerized tomography
DAE	denoising AE
DB	database
DBM	deep Boltzmann machines
DBN	deep belief network
D-CNN	deep CNN
DE	dense sampling
DEM	deep energy model
DenseNet	dense network
DiDeMo	distinct describable moments
DL	deep learning
DNN	deep neural network
DoG	difference of Gaussians
DoF	degrees of freedom
DS	direction specific
DSN	deeply supervised nets
DSTIP	depth STIP
ED	elevation descriptor
ELU	exponential linear unit
EMK	efficient match kernel
EVD	eigenvalue decomposition
FAST	features from accelerated segment test
FC	fully connected

Abbreviation	Explanation
FCN	fully convolutional networks
FCVID	Fudan-Columbia video dataset
FMS	full modality specific
FPFH	fast PFH
FREAK	fast retina keypoint
fus-CNN	fusion CNN
fwavacc	frequency weighted average accuracy
GAH	geometric attribute histograms
GAN	generative adversarial network
GFU	gated fusion unit
GNN	graph neural network
GPU	graphics processing unit
GRBM	gated RBM
GRU	gated recurrent unit
GT	ground truth
HAR	human action recognition
Harris3D	Harris 3D
HBN	half layers batch normalized
HCRF	hidden CRF
HHA	horizontal disparity, height above ground, angle the pixels local surface normal makes with the inferred gravity direction
HKDE	hierarchical KDE
HKS	heat kernel signature
HMC	hidden Markov chain
HMDB51	human motion database
HMM	hidden Markov model
HMP	hierarchical matching pursuit
HOF	histogram of flow
HOG	histogram of oriented gradients
HON	histogram of surface normals
HON4D	HON 4D
HOPC	histogram of principal components
HSMM	hidden semi-Markov model
I3D	inflated 3D CNN
IDT	improved dense trajectories
IP	interest point
KDE	kernel descriptor

Abbreviation	Explanation
kd-tree	k dimensional tree
KITTI	? (not mentioned in the work that proposes it [98])
KLT	Kanade Lucas-Tomasi
k-NN	k nearest neighbors
kSVM	kernel SVM
KTH	Royal institute of technology, Stockholm
LBP	local binary pattern
LeNet	LeCun network
LFD	light field descriptor
LFSH	local feature statistics histogram
LiDAR	light detection and ranging
LINE	linearizing the memory
LINEMOD	multimodal LINE
linSVM	linear SVM
LN	locally connected
LRCN	long-term recurrent CNN
LReLU	leaky ReLU
LRF	local reference frame
LSP	local surface patch
LSTM	long short-term memory node
LSTM-CF	LSTM context fusion
LTC	long temporal convolutional network
LTP	local trinary pattern
MAE	mean absolute error
MD	multiple dictionary
meanIU	mean intersection over union
MK-MMD	multiple kernel maximum mean discrepancy
MLP	multi-layer perceptron
MMF	multi modal feature fusion
MNIST	modified national institute of standards and technology
MO-AniProbing	multi orientation anisotropic probing
mp	max pooling
MR	magnetic resonance
MRF	Markov random field
MRI	magnetic resonance imaging
MVCNN	multi view CNN
MVD	multi-view depth

Abbreviation	Explanation
NaN	not a number
NBN	no BN
NiN	network in network
nl	norm loss
NN	nearest neighbor
NNDR	nearest neighbor distance ratio
NYU	New York University
NYUv2	NYU version 2
OGH	oriented gradient histograms
OLM	orthogonal linear module
ONI	orthogonalization using Newton's iteration
op	orientation pooling
ORB	oriented FAST and rotated BRIEF
ORION	orientation boosted voxel net
ORN	orientation response network
PA-LSTM	part-aware LSTM
PBWN	projection based weight normalization
PCA	principal component analysis
PELU	parametric ELU
PFH	point feature histogram
pixacc	pixel accuracy
PPF	point pair feature
PReLU	parametric ReLU
PSB	Princeton shape benchmark
PSG	polygonal surface geometry
RA	reference angle
RANSAC	random sample consensus
RAS	Reynolds-averaged simulation
RBM	restricted Boltzmann machine
R-CNN	regions with CNN features
RDF	randomized decision forest
RDF-Net	RGB-D fusion network
ReLU	rectified linear unit
ResBlock	residual block
ResNet	residual network
RF	random forest
RFB	residual fusion block

Abbreviation	Explanation
RGB	Red-Green-Blue
RGB-D	Red-Green-Blue-Depth
RI-LBC	rotation invariant local binary convolution
RMSE	root mean square error
RNN	recurrent neural network
Rohr3D	Karl Rohr 3D
RoSP	rotational projection statistics
RotEqNet	rotation equivariant vector field network
RQ	research question
RSM	rotational silhouette map
SC	shape context
SD	single dictionary
SDH	spatial distribution histograms
SF	sparse fusion
SFCNN	steerable filter CNN
SfM	structure from motion
SGD	stochastic gradient descent
SHOT	signature of histograms of orientation
SHREC	shape retrieval contest
SI	spin image
SIFT	scale invariant feature transform
SI-HKS	SI HKS
SISI	scale invariant SI
SLAM	simultaneous localization and mapping
SP	superpixel
SPN	scalar field processing network
SRIP	spectral restricted isometry property
SSCD	spatial structure circular descriptor
SSD	sum of squared differences
SSMA	self-supervised model adaptation
SSVM	structural SVM
std	standard deviation
STIP	spatio-temporal interest point
ST-LSTM	spatio-temporal LSTM
STN	spatial transform networks
SUN	scene understanding
SUN-CG	? (not mentioned in the work that proposes it [346])

Abbreviation	Explanation
SURF	speeded up robust features
SVD	singular value decomposition
SVM	support vector machine
SYNTHIA	synthetic collection of imagery and annotations
TACoS	textually annotated cooking scenes
TDD	trajectory pooled deep convolutional descriptors
THRIFT	? (not mentioned in the work that proposes it [90])
TI	transformation invariant
TOLDI	triple orthogonal local depth images
Tri-SI	Tri-Spin-Image
UCF	university of central Florida
UMAM	unified model of appearance and motion
US	ultrasound
VC	velocity coherent
V-FAST	video FAST
VFT	vector field topology
VGG	? (not mentioned in the work that proposes it [338])
VPN	vector processing network
VRN	Voxeption ResNet
wd	weight decay
WKS	wave kernel signature
WN	weight normalization
WRN	wide ResNet
YCB	Yale-CMU-Berkeley
YFCC100M	Yahoo Flickr creative commons 100 million

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th symposium on operating systems design and implementation (OSDI)*, pages 265–283, 2016.

[2] P-A Absil and Kyle A Gallivan. Joint diagonalization on the oblique manifold for independent component analysis. In *IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 5, pages V–V, 2006.

[3] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[4] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016.

[5] Forest Agostinelli, Matthew Hoffman, Peter Sadowski, and Pierre Baldi. Learning activation functions to improve deep neural networks. *arXiv preprint arXiv:1412.6830*, 2014.

[6] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. Censure: Center surround extremas for realtime feature detection and matching. In *Proceedings of the European conference on computer vision (ECCV)*, pages 102–115. Springer, 2008.

[7] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. Freak: Fast retina keypoint. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 510–517. IEEE, 2012.

[8] Luís A Alexandre. 3d object recognition using convolutional neural networks with transfer learning between input channels. In *Intelligent Autonomous Systems 13*, pages 889–898. Springer, 2016.

[9] Stéphane Allaire, John J Kim, Stephen L Breen, David A Jaffray, and Vladimir Pekar. Full orientation invariance and improved feature selectivity of 3d sift with application to medical image analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops (CVPRW)*, pages 1–8. IEEE, 2008.

[10] Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. Localizing moments in video with natural language. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 5803–5812. IEEE, 2017.

[11] Mathieu Aubry, Ulrich Schlickewei, and Daniel Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. In *Proceedings of the IEEE international conference on computer vision workshops (ICCVW)*, pages 1626–1633. IEEE, 2011.

[12] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[13] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Sequential deep learning for human action recognition. In *International workshop on human behavior understanding*, pages 29–39. Springer, 2011.

[14] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 39:2481–2495, 2017.

[15] Randall Balestriero et al. A spline theory of deep learning. In *International Conference on Machine Learning (ICML)*, pages 374–383, 2018.

[16] Vassileios Balntas, Andreas Doumanoglou, Caner Sahin, Juil Sock, Rigas Kouskouridas, and Tae-Kyun Kim. Pose guided rgbd feature learning for 3d object pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3856–3864. IEEE, 2017.

[17] Nitin Bansal, Xiaohan Chen, and Zhangyang Wang. Can we gain more from orthogonality regularizations in training deep networks? In *Advances in Neural Information Processing Systems*, pages 4261–4271, 2018.

[18] Mohammadamin Barekatain, Miquel Martí, Hsueh-Fu Shih, Samuel Murray, Kotaro Nakayama, Yutaka Matsuo, and Helmut Prendinger. Okutama-action: An aerial view video dataset for concurrent human action detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops (CVPRW)*, pages 28–35. IEEE, 2017.

[19] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 404–417. Springer, 2006.

[20] Paul R Beaudet. Rotationally invariant image operators. In *Proceedings of the 4th international joint conference on pattern recognition*, 1978.

[21] Jens Behley, Volker Steinhage, and Armin B Cremers. Laser-based segment classification using a mixture of bag-of-words. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 4195–4200. IEEE, 2013.

[22] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 24:509–522, 2002.

[23] Hakan Bilen, Basura Fernando, Efstratios Gavves, Andrea Vedaldi, and Stephen Gould. Dynamic image networks for action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3034–3042. IEEE, 2016.

[24] Michael J Black and Allan D Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International journal of computer vision (IJCV)*, 26:63–84, 1998.

[25] Liefeng Bo, Kevin Lai, Xiaofeng Ren, and Dieter Fox. Object recognition with hierarchical kernel descriptors. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1729–1736. IEEE, 2011.

[26] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Kernel descriptors for visual recognition. In *Advances in neural information processing systems 23*, pages 244–252. Curran Associates, Inc., 2010.

[27] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Depth kernel descriptors for object recognition. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 821–826. IEEE, 2011.

[28] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Unsupervised feature learning for rgb-d based object recognition. In Jaydev P. Desai, Gregory Dudek, Oussama Khatib, and Vijay Kumar, editors, *Experimental Robotics*, pages 387–402. Springer, 2013.

[29] Aaron F. Bobick and James W. Davis. The recognition of human movement using temporal templates. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 23:257–267, 2001.

[30] Ujwal Bonde, Vijay Badrinarayanan, and Roberto Cipolla. Robust instance recognition in presence of occlusion and clutter. In *Proceedings of the European conference on computer vision (ECCV)*, pages 520–535. Springer, 2014.

[31] Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59:291–294, 1988.

[32] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[33] Matteo Bregonzio, Shaogang Gong, and Tao Xiang. Recognising action as clouds of space-time interest points. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1948–1955. IEEE, 2009.

[34] Rasmus Bro, Evrim Acar, and Tamara G Kolda. Resolving the sign ambiguity in the singular value decomposition. *Journal of Chemometrics*, 22:135–140, 2008.

[35] Andrew Brock, Theodore Lim, JM Ritchie, and Nick Weston. Generative and discriminative voxel modeling with convolutional neural networks. *arXiv preprint arXiv:1608.04236*, 2016.

[36] Alexander Bronstein, Michael Bronstein, and Maks Ovsjanikov. 3d features, surface descriptors, and object descriptors. *3D Imaging, Analysis, and Applications*, pages 1–27, 2010.

[37] Alexander M Bronstein, Michael M Bronstein, Leonidas J Guibas, and Maks Ovsjanikov. Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Transactions on Graphics (TOG)*, 30:1, 2011.

[38] Michael M Bronstein and Iasonas Kokkinos. Scale-invariant heat kernel signatures for non-rigid shape recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1704–1711. IEEE, 2010.

[39] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 961–970. IEEE, 2015.

[40] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *International conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015.

[41] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 778–792, 2010.

[42] Liangliang Cao, Zicheng Liu, and Thomas S Huang. Cross-dataset action detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1998–2005. IEEE, 2010.

[43] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 4724–4733. IEEE, 2017.

[44] Bhaskar Chakraborty, Michael B Holte, Thomas B Moeslund, and Jordi Gonzàlez. Selective spatio-temporal interest points. *Computer vision and image understanding (CVIU)*, 116:396–410, 2012.

[45] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[46] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. On visual similarity based 3d model retrieval. In *Computer graphics forum*, pages 223–232. Wiley Online Library, 2003.

[47] Hui Chen and Bir Bhanu. 3d free-form object recognition in range images using local surface patches. *Pattern Recognition Letters*, pages 1252–1262, 2007.

[48] Minmin Chen, Jeffrey Pennington, and Samuel Schoenholz. Dynamical isometry and a mean field theory of RNNs: Gating enables signal propagation in recurrent neural networks. In *Proceedings of the International conference on machine learning (ICML)*, volume 80, pages 873–882. PMLR, Jul 2018.

[49] Gong Cheng, Peicheng Zhou, and Junwei Han. Rifd-cnn: Rotation-invariant and fisher discriminative convolutional neural networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 2884–2893. IEEE, 2016.

[50] Warren Cheung and Ghassan Hamarneh. N-sift: N-dimensional scale invariant feature transform for matching medical images. In *4th IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 720–723. IEEE, 2007.

[51] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[52] Sungjoon Choi, Qian-Yi Zhou, Stephen Miller, and Vladlen Koltun. A large dataset of object scans. *arXiv:1602.02481*, 2016.

[53] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[54] Chris A Cocosco, Vasken Kollokian, Remi K-S Kwan, G Bruce Pike, and Alan C Evans. Brainweb: Online interface to a 3d mri simulated brain database. In *NeuroImage*. Citeseer, 1997.

[55] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *Proceedings of the International conference on machine learning (ICML)*, pages 2990–2999, 2016.

[56] Tim Cooijmans, Nicolas Ballas, César Laurent, Çağlar Gülçehre, and Aaron Courville. Recurrent batch normalization. *arXiv preprint arXiv:1603.09025*, 2016.

[57] Camille Couprie. Multi-label energy minimization for object class segmentation. In *Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pages 2233–2237. IEEE, 2012.

[58] Camille Couprie, Clément Farabet, Laurent Najman, and Yann LeCun. Indoor semantic segmentation using depth information. *arXiv preprint arXiv:1301.3572*, 2013.

[59] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of*

the IEEE conference on computer vision and pattern recognition (CVPR), pages 113–123, 2019.

[60] John D. and Anderson Jr. *Computational Fluid Dynamics*. McGraw-Hill, 1995.

[61] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 5828–5839. IEEE, 2017.

[62] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 886–893. IEEE, 2005.

[63] Tal Darom and Yosi Keller. Scale-invariant features for 3-d mesh models. *IEEE Transactions on Image Processing*, 21:2758–2769, 2012.

[64] Liuyuan Deng, Ming Yang, Tianyi Li, Yuesheng He, and Chunxiang Wang. Rfbnet: Deep multimodal networks with residual fusion blocks for rgb-d semantic segmentation. *arXiv preprint arXiv:1907.00135*, 2019.

[65] Zhuo Deng, Sinisa Todorovic, and Longin Jan Latecki. Semantic segmentation of rgbd images with mutex constraints. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 1733–1741. IEEE, 2015.

[66] H Quynh Dinh and Liefei Xu. Measuring the similarity of vector fields using global distributions. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 187–196. Springer, 2008.

[67] Erickson R do Nascimento, Gabriel L Oliveira, Antônio W Vieira, and Mario FM Campos. On the development of a robust, fast and lightweight keypoint descriptor. *Neurocomputing*, 120:141–155, 2013.

[68] Piotr Dollár, Vincent Rabaud, Garrison Cottrell, and Serge Belongie. Behavior recognition via sparse spatio-temporal features. In *Workshop on visual surveillance and performance evaluation of tracking and surveillance.*, pages 65–72. IEEE, 2005.

[69] Jose Dolz, Christian Desrosiers, and Ismail Ben Ayed. 3d fully convolutional networks for subcortical segmentation in mri: A large-scale study. *NeuroImage*, 170:456–470, 2017.

[70] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 2625–2634. IEEE, 2015.

[71] Andreas Doumanoglou, Rigas Kouskouridas, Sotiris Malassiotis, and Tae-Kyun Kim. Recovering 6d object pose and predicting next-best-view in the crowd. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3583–3592. IEEE, 2016.

[72] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE, 2010.

[73] Yong Du, Wei Wang, and Liang Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1110–1118. IEEE, 2015.

[74] Aman Dureja and Payal Pahwa. Image retrieval techniques: a survey. *International Journal of Engineering & Technology*, 7(1.2):215–219, 2018.

[75] Julia Ebling and Gerik Scheuermann. Clifford convolution and pattern matching on vector fields. In *Proceedings of the 14th IEEE visualization*, page 26. IEEE Computer Society, 2003.

[76] Alexei A Efros, Alexander C Berg, Greg Mori, and Jitendra Malik. Recognizing action at a distance. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, page 726. IEEE, 2003.

[77] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 2650–2658. IEEE, 2015.

[78] Andreas Eitel, Jost Tobias Springenberg, Luciano Spinello, Martin Riedmiller, and Wolfram Burgard. Multimodal deep learning for robust rgb-d object recognition. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 681–687. IEEE, 2015.

[79] Hanan ElNaghy, Safwat Hamad, and M Essam Khalifa. Taxonomy for 3d content-based object retrieval methods. *International Journal of Recent Research and Applied Studies (IJRRAS)*, 14:412–446, 2013.

[80] Felix Endres, Jürgen Hess, Nikolas Engelhard, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. An evaluation of the rgb-d slam system. In *IEEE international conference on robotics and automation (ICRA)*, pages 1691–1696. IEEE, 2012.

[81] Felix Endres, Jürgen Hess, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. 3-d mapping with an rgb-d camera. *Transactions on Robotics*, 30:177–187, 2014.

[82] Martin Engelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, and Ingmar Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *IEEE international conference on robotics and automation (ICRA)*, pages 1355–1361. IEEE, 2017.

[83] POST F., V. ROLIJK B., H AUSER H., L ARAMEE R., and D OLEISCH H. The state of the art in flow visualisation: Feature extraction and tracking. *Computer Graphics Forum*, 22(4):775–792, 2003.

[84] Yuchen Fan, Yao Qian, Feng-Long Xie, and Frank K Soong. Tts synthesis with bidirectional lstm based recurrent neural networks. In *Fifteenth annual conference of the international speech communication association*, 2014.

[85] Clément Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Scene parsing with multiscale feature learning, purity trees, and optimal covers. In *Proceedings of the International conference on machine learning (ICML)*, pages 1857–1864. Omnipress, 2012.

[86] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 35:1915–1929, 2013.

[87] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1933–1941. IEEE, 2016.

[88] Basura Fernando, Stratis Gavves, Oramas Mogrovejo, José Antonio, Amir Ghodrati, and Tinne Tuytelaars. Modeling video evolution for action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 5378–5387. IEEE, 2015.

[89] Michael Firman. Rgbd datasets: Past, present and future. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops (CVPRW)*, pages 19–31. IEEE, 2016.

[90] Alex Flint, Anthony Dick, and Anton Van Den Hengel. Thrift: Local 3d structure recognition. In *9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications (DICTA)*, pages 182–188. IEEE, 2007.

[91] Andrea Frome, Daniel Huber, Ravi Kolluri, Thomas Bülow, and Jitendra Malik. Recognizing objects in range data using regional point descriptors. In *Proceedings of the European conference on computer vision (ECCV)*, pages 224–237. Springer, 2004.

[92] Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. Tall: Temporal activity localization via language query. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 5267–5275. IEEE, 2017.

[93] Yue Gao, Qionghai Dai, and Nai-Yao Zhang. 3d model comparison using spatial structure circular descriptor. *Pattern Recognition*, 43:1142–1151, 2010.

[94] Noa Garcia. Temporal aggregation of visual features for large-scale image-to-video retrieval. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, pages 489–492. ACM, 2018.

[95] Noa Garcia and George Vogiatzis. Dress like a star: Retrieving fashion products from videos. In *Proceedings of the IEEE international conference on computer vision workshops (ICCVW)*, pages 2293–2299. IEEE, 2017.

[96] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, and Jose Garcia-Rodriguez. A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*, 2017.

[97] Christoph Garth, Robert S Laramee, Xavier Tricoche, Jürgen Schneider, and Hans Hagen. Extraction and visualization of swirl and tumble motion from engine simulation data. In *Topology-based Methods in Visualization*, pages 121–135. Springer, 2007.

[98] Andreas Geiger. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3354–3361. IEEE, 2012.

[99] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of machine learning research (JMLR)*, 3:115–143, 2002.

[100] R. B Girshick, P. F Felzenszwalb, and D. A Mcallester. Object detection with grammar models. In *Advances in Neural Information Processing Systems*, pages 442–450, 2011.

[101] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. PMLR, 2011.

[102] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[103] Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *Proceedings of the International conference on machine learning (ICML)*, pages III–1319–III–1327. Omnipress, 2013.

[104] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, page 3. IEEE, 2017.

[105] Lars Graening and Thomas Ramsay. Flow field data mining based on a compact streamline representation. Technical report, SAE Technical Paper, 2015.

[106] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *Transactions on neural networks and learning systems*, 28:2222–2232, 2017.

[107] Wulong Guo, Weiduo Hu, Chang Liu, and Tingting Lu. 3d object recognition from cluttered and occluded scenes with a compact local feature. *Machine vision and applications*, 30:763–783, 2019.

[108] Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 481–490. ACM, 2016.

[109] Y Guo, Ferdous Sohel, Mohammed Bennamoun, M Lu, and J Wan. Trisi: A distinctive local surface descriptor for 3d modeling and object recognition. In *8th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, pages 86–93. Scitepress, 2013.

[110] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48, 2016.

[111] Yanming Guo, Yu Liu, **Theodoros Georgiou**, and Michael Lew. A review of semantic segmentation using deep neural networks. *International Journal of Multimedia Information Retrieval (IJMIR)*, 7:87–93, 2018.

[112] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, and Jianwei Wan. 3d object recognition in cluttered scenes with local surface features: a survey. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, pages 2270–2287, 2014.

[113] Yulan Guo, Ferdous Sohel, Mohammed Bennamoun, Min Lu, and Jianwei Wan. Rotational projection statistics for 3d local surface description and object recognition. *International journal of computer vision (IJCV)*, 105:63–86, 2013.

[114] Yulan Guo, Ferdous Sohel, Mohammed Bennamoun, Jianwei Wan, and Min Lu. A novel local surface feature for 3d object recognition under clutter and occlusion. *Information Sciences*, pages 196–213, 2015.

[115] Saurabh Gupta, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Indoor scene understanding with rgb-d images: Bottom-up segmentation, object detection and semantic segmentation. *International journal of computer vision (IJCV)*, 112:133–149, 2015.

[116] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 564–571. IEEE, 2013.

[117] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 345–360. Springer, 2014.

[118] Simon Hadfield, Karel Lebeda, and Richard Bowden. Hollywood 3d: What are the best 3d features for action recognition? *International journal of computer vision (IJCV)*, 121:95–110, 2017.

[119] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 5927–5935, 2017.

[120] Ju Han and Kai-Kuang Ma. Fuzzy color histogram and its use in color image retrieval. *IEEE Transactions on image Processing*, 11(8):944–952, 2002.

[121] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. Understanding real world indoor scenes with synthetic data. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 4077–4085. IEEE, 2016.

[122] Mehrtash Harandi and Basura Fernando. Generalized backpropagation,\'{E} tude de cas: Orthogonality. *arXiv preprint arXiv:1611.05927*, 2016.

[123] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, pages 10–5244. Citeseer, 1988.

[124] Tal Hassner. A critical review of action recognition benchmarks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops (CVPRW)*, pages 245–250. IEEE, 2013.

[125] Caner Hazirbas, Lingni Ma, Csaba Domokos, and Daniel Cremers. Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture. In *Asian conference on computer vision (ACCV)*, pages 213–228. Springer, 2016.

[126] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 1026–1034. IEEE, 2015.

[127] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 770–778. IEEE, 2016.

[128] Vishakh Hegde and Reza Zadeh. Fusionnet: 3d object classification using multiple data representations. *arXiv preprint arXiv:1607.05695*, 2016.

[129] Geremy Heitz and Daphne Koller. Learning spatial context: Using stuff to find things. In *Proceedings of the European conference on computer vision (ECCV)*, pages 30–43. Springer, 2008.

[130] James Helman and Lambertus Hesselink. Representation and display of vector field topology in fluid flow data sets. *IEEE computer*, 22(8):27–36, 1989.

[131] Joao F Henriques and Andrea Vedaldi. Warped convolutions: Efficient invariance to spatial transformations. In *Proceedings of the International conference on machine learning (ICML)*, pages 1461–1469. PMLR, 2017.

[132] Samitha Herath, Mehrtash Harandi, and Fatih Porikli. Going deeper into action recognition: A survey. *Image and vision computing*, 60:4–21, 2017.

[133] Alexander Hermans, Georgios Floros, and Bastian Leibe. Dense 3d semantic mapping of indoor scenes from rgb-d images. In *IEEE international conference on robotics and automation (ICRA)*, pages 2631–2638. IEEE, 2014.

[134] Stefan Hinterstoisser, Cedric Cagniart, Slobodan Ilic, Peter Sturm, Nassir Navab, Pascal Fua, and Vincent Lepetit. Gradient response maps for real-time detection of textureless objects. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, pages 876–888, 2012.

[135] Stefan Hinterstoisser, Stefan Holzer, Cedric Cagniart, Slobodan Ilic, Kurt Konolige, Nassir Navab, and Vincent Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 858–865. IEEE, 2011.

[136] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian conference on computer vision (ACCV)*, pages 548–562. Springer, 2012.

[137] Stefan Hinterstoisser, Vincent Lepetit, Naresh Rajkumar, and Kurt Konolige. Going further with point pair features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 834–848. Springer, 2016.

[138] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18:1527–1554, 2006.

[139] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, pages 504–507, 2006.

[140] Geoffrey E Hinton and Terrence J Sejnowski. Learning and releaming in boltzmann machines. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1:2, 1986.

[141] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, pages 1735–1780, 1997.

[142] Nico Höft, Hannes Schulz, and Sven Behnke. Fast semantic segmentation of rgb-d scenes with gpu-accelerated deep neural networks. In *Joint German/Austrian Conference on Artificial Intelligence*, pages 80–85. Springer, 2014.

[143] David R Holmes, Ellis L Workman, and Richard A Robb. The nlm-mayo image collection: Common access to uncommon data. In *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI) Workshop*, 2005.

[144] Berthold Klaus Paul Horn. Extended gaussian images. *Proceedings of the IEEE*, pages 1671–1686, 1984.

[145] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. Scenenn: A scene meshes dataset with annotations. In *Fourth International Conference on 3D Vision (3DV)*, pages 92–101. IEEE, 2016.

[146] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 2261–2269. IEEE, 2017.

[147] Lei Huang, Li Liu, Fan Zhu, Diwen Wan, Zehuan Yuan, Bo Li, and Ling Shao. Controllable orthogonalization in training dnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6429–6438, 2020.

[148] Lei Huang, Xianglong Liu, Bo Lang, and Bo Li. Projection based weight normalization for deep neural networks. *arXiv preprint arXiv:1710.02338*, 2017.

[149] Lei Huang, Xianglong Liu, Bo Lang, Adams Wei Yu, Yongliang Wang, and Bo Li. Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[150] Lei Huang, Dawei Yang, Bo Lang, and Jia Deng. Decorrelated batch normalization. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 791–800. IEEE, 2018.

[151] Haroon Idrees, Amir R Zamir, Yu-Gang Jiang, Alex Gorban, Ivan Laptev, Rahul Sukthankar, and Mubarak Shah. The thumos challenge on action recognition for videos in the wild. *Computer vision and image understanding (CVIU)*, 155:1–23, 2017.

[152] Anastasia Ioannidou, Elisavet Chatzilari, Spiros Nikolopoulos, and Ioannis Kompatsiaris. Deep learning advances in computer vision with 3d data: A survey. *ACM Computing Surveys (CSUR)*, page 20, 2017.

[153] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, volume 37, pages 448–456. PMLR, 2015.

[154] Sergey Ioffe and Christian Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 448–456. Omnipress, 2015.

[155] L. Helman J. and Hesselink L. Representation and display of vector field topology in fluid flow data sets. *IEEE Computer*, 22(8):27–36, 1989.

[156] L. Helman J. and Hesselink L. Visualizing vector field topology in fluid flows. *IEEE Computer Graphics and Applications*, 11(3):36–46, 1991.

[157] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.

[158] Allison Janoch, Sergey Karayev, Yangqing Jia, Jonathan T Barron, Mario Fritz, Kate Saenko, and Trevor Darrell. A category-level 3d object dataset: Putting the kinect to work. In Andrea Fossati, Juergen Gall, Helmut Grabner, Xiaofeng Ren, and Kurt Konolige, editors, *Consumer Depth Cameras for Computer Vision*, pages 141–165. Springer, 2013.

[159] Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, et al. What is the best multi-stage architecture for object recognition? In *IEEE 12th international conference on computer vision (ICCV)*, pages 2146–2153. IEEE, 2009.

[160] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, pages 221–231, 2013.

[161] Yu-Gang Jiang, Zuxuan Wu, Jun Wang, Xiangyang Xue, and Shih-Fu Chang. Exploiting feature and class relationships in video categorization with regularized deep neural networks. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 40:352–364, 2018.

[162] Yun Jiang, Stephen Moseson, and Ashutosh Saxena. Efficient grasping from rgbd images: Learning using a new rectangle representation. In *IEEE international conference on robotics and automation (ICRA)*, pages 3304–3311. IEEE, 2011.

[163] Xiaojie Jin, Chunyan Xu, Jiashi Feng, Yunchao Wei, Junjun Xiong, and Shuicheng Yan. Deep learning with s-shaped rectified linear activation units. In *AAAI Conference on Artificial Intelligence*, pages 1737–1743, 2016.

[164] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 21:433–449, 1999.

[165] Andrew E Johnson, Allan R Klumpp, James B Collier, and Aron A Wolf. Lidar-based hazard avoidance for safe landing on mars. *Journal of guidance, control, and dynamics*, pages 1091–1099, 2002.

[166] Andrew Edie Johnson and Martial Hebert. Surface matching for object recognition in complex three-dimensional scenes. *Image and vision computing*, 16:635–651, 1998.

[167] T. Kadir and M. Brady. Scale saliency: a novel approach to salient feature and scale selection. In *International conference on visual information engineering (VIE)*, pages 25–28. IET, 2003.

[168] Soo Min Kang and Richard P Wildes. Review of action recognition and detection methods. *arXiv preprint arXiv:1610.06906*, 2016.

[169] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1725–1732. IEEE, 2014.

[170] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[171] Yan Ke, Rahul Sukthankar, and Martial Hebert. Efficient visual event detection using volumetric features. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 166–173. IEEE, 2005.

[172] Wadim Kehl, Fausto Milletari, Federico Tombari, Slobodan Ilic, and Nassir Navab. Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 205–220. Springer, 2016.

[173] Christian Kerl, Jurgen Sturm, and Daniel Cremers. Dense visual slam for rgb-d cameras. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 2100–2106. IEEE, 2013.

[174] Salman Hameed Khan, Mohammed Bennamoun, Ferdous Sohel, and Roberto Togneri. Geometry driven semantic labeling of indoor scenes. In *Proceedings of the European conference on computer vision (ECCV)*, pages 679–694. Springer, 2014.

[175] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[176] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in neural information processing systems 30*, pages 971–980. Curran Associates, Inc., 2017.

[177] Alexander Klaser, Marcin Marszałek, and Cordelia Schmid. A spatio-temporal descriptor based on 3d-gradients. In *Proceedings of the British machine vision conference (BMVC)*, pages 995–1004. BMVA Press, 2008.

[178] Jan Knopp, Mukta Prasad, Geert Willems, Radu Timofte, and Luc Van Gool. Hough transform and 3d surf for robust three dimensional classification. In *Proceedings of the European conference on computer vision (ECCV)*, pages 589–602. Springer, 2010.

[179] Jan J Koenderink and Andrea J van Doorn. Representation of local geometry in the visual system. *Biological cybernetics*, 55:367–375, 1987.

[180] Hema S Koppula, Abhishek Anand, Thorsten Joachims, and Ashutosh Saxena. Semantic labeling of 3d point clouds for indoor scenes. In *Advances in neural information processing systems 24*, pages 244–252. Curran Associates, Inc., 2011.

[181] Iryna Korshunova, Wenzhe Shi, Joni Dambre, and Lucas Theis. Fast face-swap using convolutional neural networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[182] Adriana Kovashka and Kristen Grauman. Learning a hierarchy of discriminative space-time neighborhood features for human action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 2046–2053. IEEE, 2010.

[183] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[184] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[185] Alexander Krull, Eric Brachmann, Frank Michel, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Learning analysis-by-synthesis for 6d pose estimation in rgb-d images. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 954–962. IEEE, 2015.

[186] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 2556–2563. IEEE, 2011.

[187] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *IEEE international conference on robotics and automation (ICRA)*, pages 1817–1824. IEEE, 2011.

[188] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Rgb-d object recognition: Features, algorithms, and a large scale benchmark. In *Consumer Depth Cameras for Computer Vision*, pages 167–192. Springer, 2013.

[189] Dmitry Laptev, Nikolay Savinov, Joachim M Buhmann, and Marc Pollefeys. Ti-pooling: transformation-invariant pooling for feature learning in convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 289–297, 2016.

[190] Ivan Laptev. On space-time interest points. *International journal of computer vision (IJCV)*, pages 107–123, 2005.

[191] Ivan Laptev, Barbara Caputo, Christian Schüldt, and Tony Lindeberg. Local velocity-adapted motion events for spatio-temporal recognition. *Computer vision and image understanding (CVIU)*, 108:207–229, 2007.

[192] Ivan Laptev and Tony Lindeberg. Velocity adaptation of space-time interest points. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pages 52–56. IEEE, 2004.

[193] Ivan Laptev and Tony Lindeberg. Local descriptors for spatio-temporal recognition. In *Spatial Coherence for Visual Motion Analysis*, pages 91–103. Springer, 2006.

[194] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1–8. IEEE, 2008.

[195] Graciela Lara López, Adriana Pena Pérez Negrón, Angélica De Antonio Jiménez, Jaime Ramírez Rodríguez, and Ricardo Imbert Paredes. Comparative analysis of shape descriptors for 3d objects. *Multimedia Tools and Applications*, 76:6993–7040, 2017.

[196] Robert S Laramee, Helwig Hauser, Lingxiao Zhao, and Frits H Post. Topology-based flow visualization, the state of the art. In *Topology-based methods in visualization*, pages 1–19. Springer, 2007.

[197] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the International conference on machine learning (ICML)*, pages 473–480. ACM, 2007.

[198] César Laurent, Gabriel Pereyra, Philémon Brakel, Ying Zhang, and Yoshua Bengio. Batch normalized recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2657–2661. IEEE, 2016.

[199] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, R Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2:396–404, 1989.

[200] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[201] Yann LeCun, Lawrence D Jackel, Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, Urs A Muller, Eduard Sackinger, Patrice Simard, et al. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, 261:276, 1995.

[202] Yann LeCun, D Touresky, G Hinton, and T Sejnowski. A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school*, volume 1, pages 21–28. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988.

[203] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Artificial intelligence and statistics*, pages 562–570. PMLR, 2015.

[204] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015.

[205] S. Leutenegger, M. Chli, and R. Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 2548–2555, 2011.

[206] Bo Li, Yijuan Lu, Chunyuan Li, Afzal Godil, Tobias Schreck, Masaki Aono, Martin Burtscher, Hongbo Fu, Takahiko Furuya, Henry Johan, et al. Shrec14 track: extended large scale sketch-based 3d shape retrieval. In *Eurographics workshop on 3D object retrieval (3DOR)*, pages 121–130, 2014.

[207] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle detection from 3d lidar using fully convolutional network. *arXiv preprint arXiv:1608.07916*, 2016.

[208] Wanqing Li, Zhengyou Zhang, and Zicheng Liu. Action recognition based on a bag of 3d points. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops (CVPRW)*, pages 9–14. IEEE, 2010.

[209] Y. Li, S. Wang, Q. Tian, and X. Ding. A survey of recent advances in visual feature detection. *Neurocomputing*, 149:736–751, 2015.

[210] Yanshan Li, Rongjie Xia, Qinghua Huang, Weixin Xie, and Xuelong Li. Survey of spatio-temporal interest point detection algorithms in video. *IEEE Access*, 5:10323–10331, 2017.

[211] Yanshan Li, Rongjie Xia, and Weixin Xie. A unified model of appearance and motion of video and its application in stip detection. *Signal, Image and Video Processing*, pages 403–410, 2018.

[212] Zhen Li, Yukang Gan, Xiaodan Liang, Yizhou Yu, Hui Cheng, and Liang Lin. Lstm-cf: Unifying context modeling and fusion with lstms for rgb-d scene labeling. In *Proceedings of the European conference on computer vision (ECCV)*, pages 541–557. Springer, 2016.

[213] Zhen Li, Yukang Gan, Xiaodan Liang, Yizhou Yu, Hui Cheng, and Liang Lin. Rgb-d scene labeling with long short-term memorized fusion model. *arXiv preprint arXiv:1604.05000*, 2016.

[214] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. In *Advances in Neural Information Processing Systems*, pages 6665–6675, 2019.

[215] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE, 2017.

[216] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[217] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proceedings of the European conference on computer vision (ECCV)*, pages 740–755. Springer, 2014.

[218] Julia Ling, Andrew Kurzawski, and Jeremy Templeton. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166, 2016.

[219] Jun Liu, Amir Shahroudy, Dong Xu, and Gang Wang. Spatio-temporal lstm with trust gates for 3d human action recognition. In *Proceedings of the European conference on computer vision (ECCV)*, pages 816–833. Springer, 2016.

[220] L. Liu, P. Fieguth, Y. Guo, X. Wang, and M. Pietikäinen. Local binary features for texture classification: Taxonomy and experimental study. *Pattern Recognition*, 62:135–160, 2017.

[221] Long Liu, Zhixuan Xi, RuiRui Ji, and Weigang Ma. Advanced deep learning techniques for image style transfer: A survey. *Signal Processing: Image Communication*, 78:465–470, 2019.

[222] Yu Liu, Yanming Guo, **Theodoros Georgiou**, and Michael Lew. Fusion that matters: convolutional fusion networks for visual recognition. *Multimedia Tools and Applications*, 77:1–28, 2018.

[223] Tsz-Wai Rachel Lo and J Paul Siebert. Local feature extraction and matching on range images: 2.5 d sift. *Computer vision and image understanding (CVIU)*, pages 1235–1250, 2009.

[224] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3431–3440. IEEE, 2015.

[225] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 1150–1157. IEEE, 1999.

[226] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision (IJCV)*, 60:91–110, 2004.

[227] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 1981 DARPA imaging understanding Workshop (IJCAI)*. Vancouver, BC, Canada, 1981.

[228] Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*, 2014.

[229] Jiang M, Machiraju R, and Thompson D. Detection and visualization of vortices. *The visualization handbook*, page 295, 2005.

[230] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the International Conference on Machine Learning (ICML)*, page 3. Omnipress, 2013.

[231] Chris Maes, Thomas Fabry, Johannes Keustermans, Dirk Smeets, Paul Suetens, and Dirk Vandermeulen. Feature detection on 3d face surfaces for pose normalisation and recognition. In *Fourth IEEE International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, pages 1–6. IEEE, 2010.

[232] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In *Proceedings of the European conference on computer vision (ECCV)*, pages 183–196, 2010.

[233] D. Marcos, M. Volpi, N. Komodakis, and D. Tuia. Rotation equivariant vector field networks. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 5048–5057, 2017.

[234] Diego Marcos, Michele Volpi, and Devis Tuia. Learning rotation invariant convolutional filters for texture classification. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pages 2012–2017. IEEE, 2016.

[235] Marcin Marszalek, Ivan Laptev, and Cordelia Schmid. Actions in context. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 2929–2936. IEEE, 2009.

[236] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International conference on artificial neural networks*, pages 52–59. Springer, 2011.

[237] Pyry Matikainen, Martial Hebert, and Rahul Sukthankar. Trajectons: Action recognition through the motion analysis of tracked features. In *Proceedings of the IEEE international conference on computer vision workshops (ICCVW)*, pages 514–521. IEEE, 2009.

[238] Takahiro Matsuda, Takahiko Furuya, and Ryutarou Ohbuchi. Lightweight binary voxel shape features for 3d data matching and retrieval. In *IEEE International Conference on Multimedia Big Data*, pages 100–107. IEEE, 2015.

[239] Daniel Maturana and Sebastian Scherer. 3d convolutional neural networks for landing zone detection from lidar. In *IEEE international conference on robotics and automation (ICRA)*, pages 3471–3478. IEEE, 2015.

[240] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 922–928. IEEE, 2015.

[241] John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J Davison. Scenenet rgb-d: 5m photorealistic images of synthetic indoor trajectories with ground truth. *arXiv preprint arXiv:1612.05079*, 2016.

[242] Roland Memisevic and Geoffrey Hinton. Unsupervised learning of image transformations. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1–8. IEEE, 2007.

[243] Stefan Menzel and Bernhard Sendhoff. Representing the change - free form deformation for evolutionary design optimisation. In Tina Yu, David Davis, Cem Baydar, and Rajkumar Roy, editors, *Evolutionary Computation in Practice*, chapter 4, pages 63–86. Springer, 2008.

[244] Ross Messing, Chris Pal, and Henry Kautz. Activity recognition using the velocity histories of tracked keypoints. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 104–111. IEEE, 2009.

[245] Ajmal Mian, Mohammed Bennamoun, and Robyn Owens. On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes. *International journal of computer vision (IJCV)*, pages 348–361, 2010.

[246] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *Proceedings of the IEEE international conference on computer vision (ICCV)*, 60(1):63–86, 2004.

[247] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 27:1615–1630, 2005.

[248] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[249] Farzin Mokhtarian, Nasser Khalili, and Peter Yuen. Multi-scale free-form 3d object recognition using 3d models. *Image and Vision Computing*, 19:271–281, 2001.

[250] Mathew Monfort, Alex Andonian, Bolei Zhou, Kandan Ramakrishnan, Sarah Adel Bargal, Tom Yan, Lisa Brown, Quanfu Fan, Dan Gutfreund, Carl Vondrick, et al. Moments in time dataset: one million videos for event understanding. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 42(2):502–508, 2019.

[251] Andreas C Müller and Sven Behnke. Learning depth-sensitive conditional random fields for semantic segmentation of rgb-d images. In *IEEE international conference on robotics and automation (ICRA)*, pages 6232–6237. IEEE, 2014.

[252] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *Transactions on Robotics*, 33:1255–1262, 2017.

[253] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the International conference on machine learning (ICML)*, pages 807–814. Omnipress, 2010.

[254] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 4694–4702. IEEE, 2015.

[255] Jiquan Ngiam, Zhenghao Chen, Pang W Koh, and Andrew Y Ng. Learning deep energy models. In *Proceedings of the International conference on machine learning (ICML)*, pages 1105–1112. Omnipress, 2011.

[256] Dong Ni, Yim Pan Chui, Yingge Qu, Xuan Yang, Jing Qin, Tien-Tsin Wong, Simon SH Ho, and Pheng Ann Heng. Reconstruction of volumetric ultrasound panorama based on improved 3d sift. *Computerized Medical Imaging and Graphics*, 33:559–566, 2009.

[257] Juan Carlos Niebles, Hongcheng Wang, and Li Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. *International journal of computer vision (IJCV)*, 79:299–318, 2008.

[258] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 1520–1528. IEEE, 2015.

[259] John Novatnack and Ko Nishino. Scale-dependent/invariant local 3d shape descriptors for fully automatic registration of multiple sets of range images. In *Proceedings of the European conference on computer vision (ECCV)*, pages 440–453. Springer, 2008.

[260] Eric Nowak, Frédéric Jurie, and Bill Triggs. Sampling strategies for bag-of-features image classification. In *Proceedings of the European conference on computer vision (ECCV)*, pages 490–503. Springer, 2006.

[261] Antonios Oikonomopoulos, Ioannis Patras, and Maja Pantic. Spatiotemporal salient points for visual recognition of human actions. *Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36:710–719, 2005.

[262] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 24:971–987, 2002.

[263] Nuria M Oliver, Barbara Rosario, and Alex P Pentland. A bayesian computer vision system for modeling human interactions. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 22:831–843, 2000.

[264] openFOAM and the openFOAM foundation. openfoam, 2011-2017.

[265] Omar Oreifej and Zicheng Liu. Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 716–723. IEEE, 2013.

[266] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Shape distributions. *ACM Transactions on Graphics (TOG)*, 21:807–832, 2002.

[267] Mete Ozay and Takayuki Okatani. Optimization on submanifolds of convolution kernels in cnns. *arXiv preprint arXiv:1610.07008*, 2016.

[268] Umut Özaydın, **Theodoros Georgiou**, and Michael Lew. A comparison of cnn and classic features for image retrieval. In *International Conference on Content-Based Multimedia Indexing (CBMI)*, pages 1–4, 2019.

[269] Chavdar Papazov and Darius Burschka. An efficient ransac for 3d object recognition in noisy and occluded scenes. In *Asian conference on computer vision (ACCV)*, pages 135–148. Springer, 2010.

[270] Chavdar Papazov, Sami Haddadin, Sven Parusel, Kai Krieger, and Darius Burschka. Rigid 3d geometry matching for grasping of known objects in cluttered scenes. *The International Journal of Robotics Research (IJRR)*, pages 538–553, 2012.

[271] Seong-Jin Park, Ki-Sang Hong, and Seungyong Lee. Rdfnet: Rgb-d multi-level residual feature fusion for indoor semantic segmentation. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 4990–4999. IEEE, 2017.

[272] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[273] A. Pobitzer, R. Peikert, R. Fuchs, B. Schindler, A. Kuhn, H. Theisel, K. Matkovi, and H. Hauser. The state of the art in topology-based visualization of unsteady flow. *Computer Graphics forum*, 30:1789–1811, 2011.

[274] Ronald Poppe. A survey on vision-based human action recognition. *Image and vision computing*, 28:976–990, 2010.

[275] Frits H Post, Benjamin Vrolijk, Helwig Hauser, Robert S Laramee, and Helmut Doleisch. The state of the art in flow visualisation: Feature extraction and tracking. In *Computer Graphics Forum*, volume 22, pages 775–792. Wiley Online Library, 2003.

[276] Christopher Poultney, Sumit Chopra, Yann L Cun, et al. Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems*, pages 1137–1144, 2007.

[277] Nan Pu, **Theodoros Georgiou**, Erwin M Bakker, and Michael Lew. Learning a domain-invariant embedding for unsupervised person re-identification. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[278] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. *arXiv preprint arXiv:1711.08488*, 2017.

[279] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE, 2017.

[280] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 5648–5656. IEEE, 2016.

[281] Xiaojuan Qi, Renjie Liao, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. 3d graph neural networks for rgbd semantic segmentation. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 5199–5208. IEEE, 2017.

[282] Alastair Quadros, James Patrick Underwood, and Bertrand Douillard. Sydney urban objects dataset. `http://www.acfr.usyd.\protect\discretionary{\char\hyphenchar\font}{}{}edu.au/papers/SydneyUrbanObjectsDataset.shtml.`, 2013.

[283] Siwen Quan, Jie Ma, Tao Ma, Fangyu Hu, and Bin Fang. Representing local shape geometry from multi-view silhouette perspective: A distinctive and robust binary 3d feature. *Signal Processing: Image Communication*, 65:67–80, 2018.

[284] Hossein Rahmani, Arif Mahmood, Du Huynh, and Ajmal Mian. Histogram of oriented principal components for cross-view action recognition. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 38:2430–2443, 2016.

[285] Hossein Rahmani, Arif Mahmood, Du Q Huynh, and Ajmal Mian. Hopc: Histogram of oriented principal components of 3d pointclouds for action recognition. In *Proceedings of the European conference on computer vision (ECCV)*, pages 742–757. Springer, 2014.

[286] Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. Grounding action descriptions in videos. *Transactions of the Association for Computational Linguistics*, 1:25–36, 2013.

[287] Mengye Ren, Renjie Liao, Raquel Urtasun, Fabian H Sinz, and Richard S Zemel. Normalizing the normalizers: Comparing and extending network normalization schemes. *arXiv preprint arXiv:1611.04520*, 2016.

[288] Xiaofeng Ren, Liefeng Bo, and Dieter Fox. Rgb-(d) scene labeling: Features and algorithms. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 2759–2766. IEEE, 2012.

[289] Colin Rennie, Rahul Shome, Kostas E Bekris, and Alberto F De Souza. A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place. *Robotics and Automation Letters*, 1:1179–1185, 2016.

[290] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *Proceedings of the European conference on computer vision (ECCV)*, pages 102–118. Springer, 2016.

[291] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the International conference on machine learning (ICML)*, pages 833–840. Omnipress, 2011.

[292] Reyes Rios-Cabrera and Tinne Tuytelaars. Discriminatively trained templates for 3d object detection: A real time scalable approach. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 2048–2055. IEEE, 2013.

[293] S. Laramee Robert, Hauser Helwig, xiao Ling, Frits Zhao, and Post H. Topology-based flow visualization, the state of the art. *Hauser H., Hagen H., Theisel H. (eds) Topology-based Methods in Visualization. Mathematics and Visualization.*, pages 1–19, 2007.

[294] Stephen K Robinson. Coherent motions in the turbulent boundary layer. *Annual Review of Fluid Mechanics*, 23(1):601–639, 1991.

[295] Mikel D Rodriguez, Javed Ahmed, and Mubarak Shah. Action mach a spatio-temporal maximum average correlation height filter for action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1–8. IEEE, 2008.

[296] Karl Rohr. On 3d differential operators for detecting point landmarks. *Image and Vision Computing*, 15:219–233, 1997.

[297] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3234–3243. IEEE, 2016.

[298] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Proceedings of the European conference on computer vision (ECCV)*, pages 430–443. Springer, 2006.

[299] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R Bradski. Orb: An efficient alternative to sift or surf. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, volume 11, page 2. Citeseer, 2011.

[300] Sebastian Ruder, Matthew E Peters, Swabha Swayamdipta, and Thomas Wolf. Transfer learning in natural language processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 15–18, 2019.

[301] DE Rumelhart. Hinton and williams, rj (1986):learning internal representations by error propagation. *parallel distributed processing*, 1, 1986.

[302] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision (IJCV)*, 115:211–252, 2015.

[303] Raif M Rustamov. Laplace-beltrami eigenfunctions for deformation invariant shape representation. In *Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 225–233. Eurographics Association, 2007.

[304] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *IEEE international conference on robotics and automation (ICRA)*, pages 3212–3217. IEEE, 2009.

[305] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Aligning point cloud views using persistent feature histograms. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 3384–3391. IEEE, 2008.

[306] Ajmal Saeed Mian, Mohammed Bennamoun, and Robyn Owens. Automated 3d model-based free-form object recognition. *Sensor Review*, pages 206–215, 2004.

[307] E. Salahat and M. Qasaimeh. Recent advances in features extraction and description algorithms: A comprehensive survey. In *IEEE international conference on industrial technology (ICIT)*, pages 1059–1063, 2017.

[308] Ruslan Salakhutdinov. Learning and evaluating boltzmann machines. *Tech. Rep., Technical Report UTML TR 2008-002, Department of Computer Science, University of Toronto*, 2008.

[309] Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In *Artificial intelligence and statistics*, pages 448–455. PMLR, 2009.

[310] Ruslan Salakhutdinov and Hugo Larochelle. Efficient learning of deep boltzmann machines. In *Artificial intelligence and statistics*, pages 693–700. PMLR, 2010.

[311] Tim Salimans and Diederik P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems 29*, pages 901–909. Curran Associates, Inc., 2016.

[312] Muhamad Risqi U Saputra, Andrew Markham, and Niki Trigoni. Visual slam and structure from motion in dynamic environments: A survey. *ACM Computing Surveys (CSUR)*, page 37, 2018.

[313] Silvio Savarese and Li Fei-Fei. 3d generic object categorization, localization and pose estimation. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 1–8. IEEE, 2007.

[314] Manolis Savva, Angel X. Chang, and Pat Hanrahan. Semantically-enriched 3d models for common-sense knowledge. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops (CVPRW)*, pages 24–31. IEEE, 2015.

[315] Sebastian Scherer, Lyle Chamberlain, and Sanjiv Singh. Autonomous landing at unprepared sites by a full-scale helicopter. *Robotics and Autonomous Systems*, pages 1545–1562, 2012.

[316] Gerik Scheuermann. Topological vector field visualization with clifford algebra. In *Ausgezeichnete Informatikdissertationen*, pages 213–222. Springer, 2000.

[317] J. L. Schonberger, H. Hardmeier, T. Sattler, and M. Pollefeys. Comparative evaluation of hand-crafted and learned local features. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1482–1491, 2017.

[318] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pages 32–36. IEEE, 2004.

[319] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *Transactions on Signal Processing*, 45:2673–2681, 1997.

[320] Brent Schwarz. Lidar: Mapping the world in 3d. *Nature Photonics*, page 429, 2010.

[321] Paul Scovanner, Saad Ali, and Mubarak Shah. A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th ACM international conference on Multimedia (ICM)*, pages 357–360. ACM, 2007.

[322] Nicu Sebe, Michael Lew, and Thomas S Huang. The state-of-the-art in human-computer interaction. In *International workshop on computer vision in human-computer interaction*, pages 1–6. Springer, 2004.

[323] Nima Sedaghat, Mohammadreza Zolfaghari, Ehsan Amiri, and Thomas Brox. Orientation-boosted voxel nets for 3d object recognition. *arXiv preprint arXiv:1604.03351*, 2016.

[324] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+ d: A large scale dataset for 3d human activity analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1010–1019. IEEE, 2016.

[325] Eli Shechtman and Michal Irani. Space-time behavior based correlation. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 405–412. IEEE, 2005.

[326] Eli Shechtman and Michal Irani. Space-time behavior-based correlation-or-how to tell if two underlying motion fields are similar without computing them? *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 29:2045–2056, 2007.

[327] Baoguang Shi, Song Bai, Zhichao Zhou, and Xiang Bai. Deeppano: Deep panoramic representation for 3-d shape recognition. *Signal Processing Letters*, 22:2339–2343, 2015.

[328] Jianbo Shi et al. Good features to track. In *1994 Proceedings of IEEE conference on computer vision and pattern recognition*, pages 593–600. IEEE, 1994.

[329] Jau-Ling Shih, Chang-Hsing Lee, and Jian Tang Wang. A new 3d model retrieval approach based on the elevation descriptor. *Pattern Recognition*, 40:283–295, 2007.

[330] Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser. The princeton shape benchmark. In *Shape modeling applications, Proceedings*, pages 167–178. IEEE, 2004.

[331] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

[332] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1297–1304. IEEE, 2011.

[333] Daniel Sieger, Sergius Gaulik, Jascha Achenbach, Stefan Menzel, and Mario Botsch. Constrained space deformation techniques for design optimization. *Computer-Aided Design*, 72:40–51, March 2016.

[334] Nathan Silberman and Rob Fergus. Indoor scene segmentation using a structured light sensor. In *Proceedings of the IEEE international conference on computer vision workshops (ICCVW)*, pages 601–608. IEEE, 2011.

[335] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Proceedings of the European conference on computer vision (ECCV)*, pages 746–760. Springer, 2012.

[336] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[337] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems 27*, pages 568–576. Curran Associates, Inc., 2014.

[338] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning*

Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.

[339] Arjun Singh, James Sha, Karthik S Narayan, Tudor Achim, and Pieter Abbeel. Bigbird: A large-scale 3d database of object instances. In *IEEE international conference on robotics and automation (ICRA)*, pages 509–516. IEEE, 2014.

[340] Tej Singh and Dinesh Kumar Vishwakarma. Video benchmarks of human action datasets: a review. *Artificial Intelligence Review*, 52:1107–1154, 2019.

[341] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 1470–1478, 2003.

[342] Richard Socher, Brody Huval, Bharath Putta Bath, Christopher D Manning, and Andrew Y Ng. Convolutional-recursive deep learning for 3d object classification. In *Advances in neural information processing systems 25*, page 8. Curran Associates Inc., 2012.

[343] Jure Sokolić, Raja Giryes, Guillermo Sapiro, and Miguel RD Rodrigues. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, 65(16):4265–4280, 2017.

[344] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 567–576. IEEE, 2015.

[345] Shuran Song and Jianxiong Xiao. Sliding shapes for 3d object detection in depth images. In *Proceedings of the European conference on computer vision (ECCV)*, pages 634–651. Springer, 2014.

[346] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1746–1754. IEEE, 2017.

[347] Yale Song, Louis-Philippe Morency, and Randall Davis. Action recognition by hierarchical sequence summarization. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3562–3569. IEEE, 2013.

[348] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[349] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[350] Hauke Strasdat, Andrew J Davison, JM Martìnez Montiel, and Kurt Konolige. Double window optimisation for constant time visual slam. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 2352–2359. IEEE, 2011.

[351] Jörg Stückler, Nenad Biresev, and Sven Behnke. Semantic mapping using object-class segmentation of rgb-d images. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 3005–3010. IEEE, 2012.

[352] Jörg Stückler, Benedikt Waldvogel, Hannes Schulz, and Sven Behnke. Dense real-time mapping of object-class semantics from rgb-d video. *Journal of Real-Time Image Processing*, 10:599–609, 2015.

[353] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 945–953. IEEE, 2015.

[354] Deqing Sun, Stefan Roth, and Michael J Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International journal of computer vision (IJCV)*, 106:115–137, 2014.

[355] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer graphics forum*, pages 1383–1392. Wiley Online Library, 2009.

[356] Ju Sun, Xiao Wu, Shuicheng Yan, Loong-Fah Cheong, Tat-Seng Chua, and Jintao Li. Hierarchical spatio-temporal context modeling for action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 2004–2011. IEEE, 2009.

[357] Michael J Swain and Dana H Ballard. Color indexing. *International journal of computer vision*, 7(1):11–32, 1991.

[358] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

[359] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich,

et al. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1–9. IEEE, 2015.

[360] Salzbrunn T., Janicke H., Wischgoll T., and Scheuermann G. The state of the art in flow visualization: Partition-based techniques. *Proceedings of the 2008 Simulation and Visualization Conference*, pages 75–92, 2008.

[361] Shuai Tang, Xiaoyu Wang, Xutao Lv, Tony X Han, James Keller, Zhihai He, Marjorie Skubic, and Shihong Lao. Histogram of oriented normal vectors for object recognition with a depth sensor. In *Asian conference on computer vision (ACCV)*, pages 525–538. Springer, 2012.

[362] Johan WH Tangelder and Remco C Veltkamp. A survey of content based 3d shape retrieval methods. *Multimedia tools and applications*, 39(3):441–471, 2008.

[363] Graham W Taylor, Rob Fergus, Yann LeCun, and Christoph Bregler. Convolutional learning of spatio-temporal features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 140–153. Springer, 2010.

[364] Alex Teichman, Jesse Levinson, and Sebastian Thrun. Towards 3d object recognition via classification of arbitrary object tracks. In *IEEE international conference on robotics and automation (ICRA)*, pages 4034–4041. IEEE, 2011.

[365] Alex Teichman and Sebastian Thrun. Tracking-based semi-supervised learning. *The International Journal of Robotics Research (IJRR)*, 31:804–818, 2012.

[366] Alykhan Tejani, Rigas Kouskouridas, Andreas Doumanoglou, Danhang Tang, and Tae-Kyun Kim. Latent-class hough forests for 6 dof object pose estimation. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 40:119–132, 2017.

[367] Alykhan Tejani, Rigas Kouskouridas, Andreas Doumanoglou, Danhang Tang, and Tae-Kyun Kim. Latent-class hough forests for 6 dof object pose estimation. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 40:119–132, 2018.

[368] **Theodoros Georgiou**, Yu Liu, Wei Chen, and Michael Lew. A survey of traditional and deep learning-based feature descriptors for high dimensional data in computer vision. *International Journal of Multimedia Information Retrieval (IJMIR)*, pages 1–36, 2019.

[369] **Theodoros Georgiou**, Sebastian Schmitt, Thomas Bäck, and Michael Lew. Orientational equivariant neural networks using clifford convolutions.

[370] **Theodoros Georgiou**, Sebastian Schmitt, Wei Chen, Thomas Bäck, and Michael Lew. Norm loss: An efficient yet effective regularization method for deep neural networks. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*. IEEE.

[371] **Theodoros Georgiou**, Sebastian Schmitt, Markus Olhofer, Yu Liu, Thomas Bäck, and Michael Lew. Learning fluid flows. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.

[372] **Theodoros Georgiou**, Sebastian Schmitt, Nan Pu, Wei Chen, Thomas Bäck, and Michael Lew. Comparison of deep learning and hand crafted features for mining simulation data. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*. IEEE.

[373] Bart Thomee, Mark J Huiskes, Erwin Bakker, and Michael Lew. Large scale image copy detection evaluation. In *Proceedings of the 1st ACM international conference on Multimedia information retrieval (ICMIR)*, pages 59–66. ACM, 2008.

[374] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. The new data and new challenges in multimedia research. *arXiv preprint arXiv:1503.01817*, 2015.

[375] Federico Tombari and Luigi Di Stefano. Hough voting for 3d object recognition under occlusion and clutter. *IPSJ Transactions on Computer Vision and Applications*, pages 20–29, 2012.

[376] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *Proceedings of the European conference on computer vision (ECCV)*, pages 356–369. Springer, 2010.

[377] Federico Tombari, Samuele Salti, and Luigi Di Stefano. A combined texture-shape descriptor for enhanced 3d feature matching. In *18th IEEE international conference on image processing (ICIP)*, pages 809–812. IEEE, 2011.

[378] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Performance evaluation of 3d keypoint detectors. *International journal of computer vision (IJCV)*, 102:198–220, 2013.

[379] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 4489–4497. IEEE, 2015.

[380] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 6450–6459. IEEE, 2018.

[381] Ludovic Trottier, Philippe Gigu, Brahim Chaib-draa, et al. Parametric exponential linear unit for deep convolutional neural networks. In *16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 207–214. IEEE, 2017.

[382] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[383] Abhinav Valada, Rohit Mohan, and Wolfram Burgard. Self-supervised model adaptation for multimodal semantic segmentation. *International journal of computer vision (IJCV)*, 2019.

[384] Gul Varol, Ivan Laptev, and Cordelia Schmid. Long-term temporal convolutions for action recognition. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 40:1510–1517, 2017.

[385] Antonio W Vieira, Erickson R Nascimento, Gabriel L Oliveira, Zicheng Liu, and Mario FM Campos. Stop: Space-time occupancy patterns for 3d action recognition from depth map sequences. In *Iberoamerican congress on pattern recognition*, pages 252–259. Springer, 2012.

[386] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the International conference on machine learning (ICML)*, pages 1096–1103. ACM, 2008.

[387] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408, 2010.

[388] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, page 3. IEEE, 2001.

[389] Eugene Vorontsov, Chiheb Trabelsi, Samuel Kadoury, and Chris Pal. On orthogonality and learning recurrent networks with long term dependencies. *arXiv preprint arXiv:1702.00071*, 2017.

[390] Wang W., Wang W., and Li S. From numerics to combinatorics: a survey of topological methods for vector field visualization. *Journal of Visualization*, 19:727–752, 2016.

[391] Anran Wang, Jiwen Lu, Gang Wang, Jianfei Cai, and Tat-Jen Cham. Multi-modal unsupervised feature learning for rgb-d scene labeling. In *Proceedings of the European conference on computer vision (ECCV)*, pages 453–467. Springer, 2014.

[392] Chu Wang, Marcello Pelillo, and Kaleem Siddiqi. Dominant set clustering and pooling for multi-view 3d object recognition. *arXiv preprint arXiv:1906.01592*, 2019.

[393] Dominic Zeng Wang, Ingmar Posner, and Paul Newman. What could move? finding cars, pedestrians and bicyclists in 3d laser data. In *IEEE international conference on robotics and automation (ICRA)*, pages 4038–4044. IEEE, 2012.

[394] Guangrun Wang, Ping Luo, Xinjiang Wang, Liang Lin, et al. Kalman normalization: Normalizing internal representations across network layers. In *Advances in Neural Information Processing Systems 31*, pages 21–31. Curran Associates, Inc., 2018.

[395] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3169–3176. IEEE, 2011.

[396] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Dense trajectories and motion boundary descriptors for action recognition. *International journal of computer vision (IJCV)*, 103:60–79, 2013.

[397] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 3551–3558. IEEE, 2013.

[398] Heng Wang, Muhammad Muneeb Ullah, Alexander Klaser, Ivan Laptev, and Cordelia Schmid. Evaluation of local spatio-temporal features for action recognition. In *Proceedings of the British machine vision conference (BMVC)*, pages 1–11, 2009.

[399] J. Wang, Z. Liu, Y. Wu, and J. Yuan. Learning actionlet ensemble for 3d human action recognition. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 36:914–927, 2014.

[400] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan. Mining actionlet ensemble for action recognition with depth cameras. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1290–1297. IEEE, 2012.

[401] Jinghua Wang, Zhenhua Wang, Dacheng Tao, Simon See, and Gang Wang. Learning common and specific features for rgb-d semantic segmentation with deconvolutional networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 664–679. Springer, 2016.

[402] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 4305–4314. IEEE, 2015.

[403] Limin Wang, Yuanjun Xiong, Zhe Wang, and Yu Qiao. Towards good practices for very deep two-stream convnets. *arXiv preprint arXiv:1507.02159*, 2015.

[404] Pichao Wang, Wanqing Li, Zhimin Gao, Jing Zhang, Chang Tang, and Philip O Ogunbona. Action recognition from depth maps using deep convolutional neural networks. *Transactions on Human-Machine Systems*, 46:498–509, 2016.

[405] Wentao Wang, Wenke Wang, and Sikun Li. From numerics to combinatorics: a survey of topological methods for vector field visualization. *Journal of Visualization*, 19(4):727–752, 2016.

[406] Yang Wang and Greg Mori. Hidden part models for human action recognition: Probabilistic versus max margin. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 33:1310–1323, 2011.

[407] Michael Warren, Luis Mejias, Xilin Yang, Bilal Arain, Felipe Gonzalez, and Ben Upcroft. Enabling aircraft emergency landings using active visual site detection. In *Field and Service Robotics*, pages 167–181. Springer, 2015.

[408] Maurice Weiler and Gabriele Cesa. General e (2)-equivariant steerable cnns. In *Advances in Neural Information Processing Systems*, pages 14334–14345, 2019.

[409] Maurice Weiler, Fred A Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 849–858, 2018.

[410] Matt Whalley, Marc Takahashi, P Tsenkov, G Schulein, and C Goerzen. Field-testing of a helicopter uav obstacle field navigation and landing system. In *65th Annual Forum of the American Helicopter Society, Grapevine, TX*, 2009.

[411] Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. Elasticfusion: Real-time dense slam and light source estimation. *The International Journal of Robotics Research (IJRR)*, 35:1697–1716, 2016.

[412] Geert Willems, Jan Hendrik Becker, Tinne Tuytelaars, and Luc J Van Gool. Exemplar-based action recognition in video. In *Proceedings of the British machine vision conference (BMVC)*, page 3. BMVA Press, 2009.

[413] Geert Willems, Tinne Tuytelaars, and Luc Van Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *Proceedings of the European conference on computer vision (ECCV)*, pages 650–663. Springer, 2008.

[414] Paul Wohlhart and Vincent Lepetit. Learning descriptors for object recognition and 3d pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3109–3118. IEEE, 2015.

[415] Shu-Fai Wong and Roberto Cipolla. Extracting spatiotemporal interest points using global information. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 1–8. IEEE, 2007.

[416] H. Wu, X. Liu, W. An, S. Chen, and H. Lyu. A deep learning approach for efficiently and accurately evaluating the flow field of supercritical airfoils. *Computers & Fluids*, 198:104393, 2020.

[417] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in neural information processing systems 29*, pages 82–90. Curran Associates, Inc., 2016.

[418] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19. Springer, 2018.

[419] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1912–1920. IEEE, 2015.

[420] Lu Xia and JK Aggarwal. Spatio-temporal depth cuboid similarity feature for activity recognition using depth camera. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 2834–2841. IEEE, 2013.

[421] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 1625–1632. IEEE, 2013.

[422] Di Xie, Jiang Xiong, and Shiliang Pu. All you need is beyond a good init: Exploring better solution for training extremely deep convolutional neural networks with orthonormality and modulation. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 6176–6185, 2017.

[423] Huijuan Xu, Kun He, Leonid Sigal, Stan Sclaroff, and Kate Saenko. Text-to-clip video retrieval with early fusion and re-captioning. *arXiv preprint arXiv:1804.05113*, 2018.

[424] Junji Yamato, Jun Ohya, and Kenichiro Ishii. Recognizing human action in time-sequential images using hidden markov model. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 379–385. IEEE, 1992.

[425] Jiaqi Yang, Zhiguo Cao, and Qian Zhang. A fast and robust local descriptor for 3d point cloud registration. *Information Sciences*, 346:163–179, 2016.

[426] Jiaqi Yang, Qian Zhang, Yang Xiao, and Zhiguo Cao. Toldi: An effective and robust approach for 3d local shape description. *Pattern Recognition*, 65:175–187, 2017.

[427] Xiaodong Yang and Ying Li Tian. Eigenjoints-based action recognition using naive-bayes-nearest-neighbor. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 14–19. IEEE, 2012.

[428] Xiaodong Yang and YingLi Tian. Super normal vector for activity recognition using depth sequences. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 804–811. IEEE, 2014.

[429] Lahav Yeffet and Lior Wolf. Local trinary patterns for human action recognition. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 492–497. IEEE, 2009.

[430] Yuichi Yoshida and Takeru Miyato. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017.

[431] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *IEEE Computational intelligence magazine*, 13(3):55–75, 2018.

[432] Hongshan Yu, Zhengeng Yang, Lei Tan, Yaonan Wang, Wei Sun, Mingui Sun, and Yandong Tang. Methods and datasets on semantic segmentation: A review. *Neurocomputing*, 304:82–103, 2018.

[433] Tsz-Ho Yu, Tae-Kyun Kim, and Roberto Cipolla. Real-time action recognition by spatiotemporal semantic and structural forests. In *Proceedings of the British machine vision conference (BMVC)*, pages 1–7. BMVA Press, 2010.

[434] Wei Yu, Kuiyuan Yang, Yalong Bai, Hongxun Yao, and Yong Rui. Visualizing and comparing convolutional neural networks. *arXiv preprint arXiv:1412.6631*, 2014.

[435] M Ersin Yumer and Niloy J Mitra. Learning semantic deformation flows with 3d convolutional networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 294–311. Springer, 2016.

[436] Mehmet Ersin Yumer, Siddhartha Chaudhuri, Jessica K Hodgins, and Levent Burak Kara. Semantic shape editing using deformation handles. *ACM Transactions on Graphics (TOG)*, 34(4):1–12, 2015.

[437] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12, 2016.

[438] Andrei Zaharescu, Edmond Boyer, Kiran Varanasi, and Radu Horaud. Surface feature detection and description with applications to mesh matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 373–380. IEEE, 2009.

[439] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[440] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 818–833. Springer, 2014.

[441] Jianguo Zhang, Marcin Marszałek, Svetlana Lazebnik, and Cordelia Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *International journal of computer vision (IJCV)*, pages 213–238, 2007.

[442] Jun-Mei Zhang, Liang Zhong, Boyang Su, Min Wan, Jinq Shya Yap, Jasmine PL Tham, Leok Poh Chua, Dhanjoo N Ghista, and Ru San Tan. Perspective on cfd

studies of coronary artery disease lesions and hemodynamics: A review. *International journal for numerical methods in biomedical engineering*, 30(6):659–680, 2014.

[443] Xin Zhang, Li Liu, Yuxiang Xie, Jie Chen, Lingda Wu, and Matti Pietikainen. Rotation invariant local binary convolution neural networks. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 1210–1219, 2017.

[444] Zhengyou Zhang. Microsoft kinect sensor and its effect. *multimedia*, 19:4–10, 2012.

[445] Bo Zhao, Jiashi Feng, Xiao Wu, and Shuicheng Yan. A survey on deep learning-based fine-grained object classification and semantic segmentation. *International Journal of Automation and Computing*, 14(2):119–135, 2017.

[446] Rui Zhao, Haider Ali, and Patrick Van der Smagt. Two-stream rnn/cnn for action recognition in 3d videos. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4260–4267. IEEE, 2017.

[447] Liang Zheng, Yi Yang, and Qi Tian. Sift meets cnn: A decade survey of instance retrieval. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 40(5):1224–1244, 2017.

[448] Yu Zhong. Intrinsic shape signatures: A shape descriptor for 3d object recognition. In *Proceedings of the IEEE international conference on computer vision workshops (ICCVW)*, pages 689–696. IEEE, 2009.

[449] Yanzhao Zhou, Qixiang Ye, Qiang Qiu, and Jianbin Jiao. Oriented response networks. In *Proceedings of the IEEE Conference on computer vision and pattern recognition (CVPR)*, pages 519–528, 2017.

[450] Yu Zou, Xueqian Wang, Tao Zhang, Bin Liang, Jingyan Song, and Houde Liu. Broph: An efficient and compact binary descriptor for 3d point clouds. *Pattern Recognition*, 76:522–536, 2018.

Summary

In this thesis we explore machine and deep learning approaches that address key challenges in high dimensional problem areas and also in improving accuracy in well known problems. In high dimensional contexts, we have focused on computational fluid dynamics (CFD) simulations. CFD simulations are able to produce complex and large outputs that accurately describe the physical properties of fluids and gases in various domains and they are frequently used for studying the effects of flow patterns and design choices on many engineering designs, such as wing, car and engine shapes. Due to the high dimensional aspect of the data, it is difficult to model toward achieving critical goals such as optimizing lift and drag forces. The key research question addressed in this thesis is whether we develop automated approaches that accurately abstract this information? We tackle these issues by studying a closely related field, 3D computer vision, and adapt approaches to the particular data type. Moreover, inspired by this data type we propose new, deep learning, approaches that are also applied to traditional computer vision.

The first part of this thesis focuses on understanding how computer vision deals with higher dimensional data than the traditional 2D image. We identify several categories of approaches as well as a generalization of methods from 2D to higher dimensions. We identify two main types of generalization, i.e. generalization to higher physical dimensions and generalization to more information per physical point, i.e. increasing the number of modalities. As the benchmarks and datasets are key components that drive the research questions and proposed approaches we also include a categorization of the available big scale dataset and benchmarks.

The second part of this thesis focuses on adapting computer vision approaches to

CFD simulation output. More specifically, combinations of CNNs and auto-encoders are used to learn to represent as well as perform model based prediction conditioned to CFD simulation output. Moreover, the more traditional feature engineering approach is tested as well and compared to the aforementioned deep learning approaches. We propose two different large scale datasets of CFD simulation output, i.e. a 3D simulation domain of the air around passenger vehicles in a virtual wind tunnel and a 2D simulation domain of the air around airfoils, which are used for training models and benchmarking their performance. With extensive experimentation, we conclude that deep learning and traditional approaches have different strengths and weaknesses and thus, according to the application in mind, a different approach might be favorable. Moreover, we concluded that, for generalization purposes, deep learning approaches outperform the hand crafted feature based ones. Finally, a common trend in most computer vision applications is that hand crafted features can provide complementary information to the deep learning approaches and a combination of the two produces higher performance models than any of the individual parts. A similar approach is considered very promising and is left for future work.

In the third, and final, part of the thesis, inspired by a large proportion of the CFD simulation output, i.e. the velocity vector fields, a new approach is proposed which focuses on vector fields and it is generalized back to traditional computer vision to create rotation invariant and equivariant deep learning models. These approaches are tested on standard benchmarks in the field, i.e. the MNIST-rot and a vehicle orientation benchmark. Finally, a weight regularization approach is defined and tested on the standard computer vision large scale image classification benchmarks and models, i.e. CIFAR-10, CIFAR-100 and ImageNet.

Samenvatting

In dit proefschrift onderzoeken we machine- en deep learning-technieken, in het bijzonder technieken die opereren in hoog dimensionale probleemgebieden of die de prestaties op bekende problemen verbeteren. In hoog dimensionale probleemgebieden hebben we ons gericht op computationele vloeistofdynamica-simulaties. Deze simulaties kunnen complexe en omvangrijke resultaten produceren die de fysieke eigenschappen van vloeistoffen en gassen in verschillende domeinen nauwkeurig beschrijven. Daardoor worden ze vaak gebruikt voor het bestuderen van de effecten van stromingspatronen en ontwerpkeuzes op veel technische ontwerpen, zoals de vorm van vleugels, auto's en motoren. Vanwege het hoog dimensionale aspect van de data is het moeilijk om belangrijke doelen te modelleren, zoals het optimaliseren van lift- en sleepkrachten. De hoofdvraag van deze thesis luidt: Kunnen we geautomatiseerde benaderingen ontwikkelen die deze informatie nauwkeurig abstraheren? We pakken deze problemen aan door een nauw verwant veld, 3D-computer vision te bestuderen, en de benaderingen aan te passen naar het specifieke probleemdomein. Geïnspireerd door dit probleemtype stellen we bovendien nieuwe, deep learning-technieken voor die ook worden toegepast op traditionele computer vision.

Het eerste deel van dit proefschrift richt zich op het begrijpen hoe computer vision omgaat met data van een hogere dimensionaliteit dan het traditionele 2D-beeld. We identificeren verschillende categorieën benaderingen en generalisatie van methoden van 2D naar hogere dimensies. We onderscheiden twee hoofdtypen generalisatie, (i) generalisatie naar hogere fysieke dimensies en (ii) generalisatie naar meer informatie per fysiek punt (i.e., toenemend aantal modaliteiten). Omdat de benchmarks en datasets de belangrijkste componenten zijn die de onderzoeksvragen

en voorgestelde benaderingen aansturen, nemen we ook een categorisering van de beschikbare grootschalige dataset en benchmarks op. Het tweede deel van dit proefschrift richt zich op het aanpassen van computer vision benaderingen aan de uitvoer van de computationele vloeistofdynamica-simulatie. We combineren CNN's en autoencoders om dit te representeren en om model-gebaseerde voorspellingen te maken, geconditioneerd op de uitvoer van de simulatie. Bovendien wordt de meer traditionele 'feature engineering' methode ook getest en vergeleken met de eerder genoemde deep learning-technieken. We stellen twee verschillende grootschalige datasets van computationele vloeistofdynamica-simulaties uitvoer voor, namelijk een 3D-simulatie domein van de lucht rond passagiersvoertuigen in een virtuele windtunnel en een 2D-simulatie domein van de lucht rond draagvlakken. Die worden gebruikt voor het trainen van modellen en het benchmarken van hun prestaties. Onderbouwd door uitgebreide experimenten concluderen we dat deep learning en traditionele benaderingen verschillende sterke en zwakke punten hebben en dat dus, afhankelijk van de toepassing, een andere benadering gunstig kan zijn. Bovendien concludeerden we dat, voor generalisatie doeleinden, deep learning-technieken beter presteren dan de handgemaakte, op functies gebaseerde benaderingen. Ten slotte, een algemene trend in de meeste computer vision-applicaties is dat handgemaakte functies aanvullende informatie kunnen bieden aan de deep learning-technieken. Daardoor zou een combinatie van beide modellen betere prestaties opleveren dan elk van de afzonderlijke onderdelen. Een vergelijkbare aanpak wordt als veelbelovend beschouwd en wordt overgelaten aan toekomstig werk.

In het derde en laatste deel van het proefschrift, dat is geïnspireerd op snelheidsvector van de computationele vloeistofdynamica-simulatie, wordt een nieuwe benadering voorgesteld die zich richt op vectorvelden en wordt gegeneraliseerd naar traditionele computer vision om rotatie-invariante en equivariante deep learning-modellen. Deze benaderingen worden standaard getest in de veld benchmarks, d.w.z. de MNIST-rot en een voertuig oriëntatie benchmark. Ten slotte wordt een benadering voor gewichtsregularisatie gedefinieerd en getest op de standaard computer vision grootschalige benchmarks en modellen voor beeldclassificatie, d.w.z. CIFAR-10, CIFAR-100 en ImageNet.

About the author

Theodoros Georgiou was born in Athens, Greece on the 11th of February 1989. He studied Physics at the National and Kapodistrian University of Athens where he graduated as a B.Sc. in 2013. He then moved to the Netherlands and obtained his M.Sc. in Computer Science from Leiden University in 2016.

In September 2016, he started his PhD supported by the Dutch Research Council (Nederlandse Organisatie voor Wetenschappelijk Onderzoek, NWO) and the Honda Research Institute - Europe GmbH (HRI-EU) in Offenbach, Germany. He worked at the Media Lab in the Leiden Institute of Advanced Computer Science (LIACS), Leiden University, the Netherlands, in the Natural Computing group of the same institute and HRI-EU, under the supervision of Prof.Dr. T.H.W. Bäck and Prof.Dr. M.S.K. Lew. During his PhD he worked in the project ”Data Mining on High Volume Simulation Output (DAMIOSO)” during which he collaborated with and was supervised by Dr. S. Schmitt and Dr. M. Olhofer. His research interests include computer vision, high dimensional data mining, deep learning and artificial intelligence applications. Specifically he is focusing on description methods for various data types, such as RGB, RGB-D and volumetric images. Moreover, he developed deep learning approaches for similar fields as well as core deep learning methods such as rotation invariant operators, regularization and normalization techniques with application CFD simulation output. He has published papers in international conferences and journal such as WCCI, IJMIR, MTAP, CBMI and ICPR.