Multi-dimensional feature and data mining

Georgiou, T.

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.

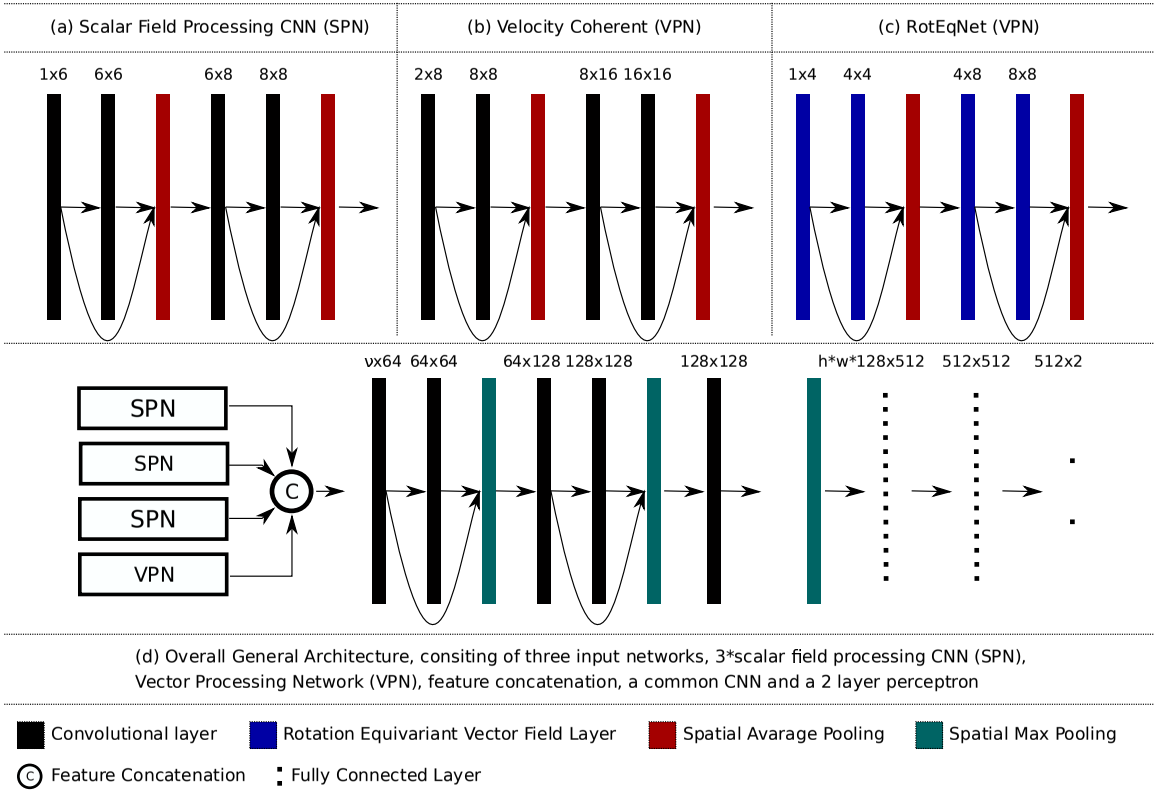


Figure 4.1: CNN architecture. (a) Is the network processing the scalar fields (SPN), shared for all scalar fields. (b) The Vector Processing Network (VPN) for the VC strategy. (c) The VPN network for the RotEqNet strategy. (d) The complete architecture. The VPN is either (b), (c) or in the case of DS 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

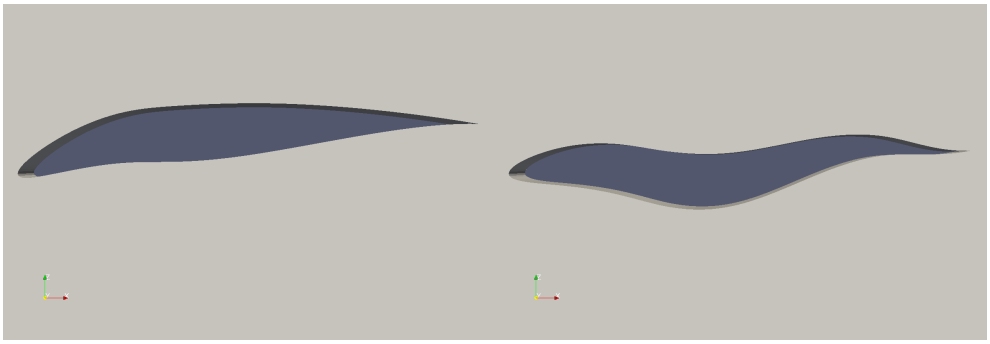
For all strategies, the input processing networks are four layers deep and the network applied after the feature concatenation consists of five convolutional layers. Spatial average pooling operations are performed every two convolutional layers in the input processing networks, whilst spatial max-pooling 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

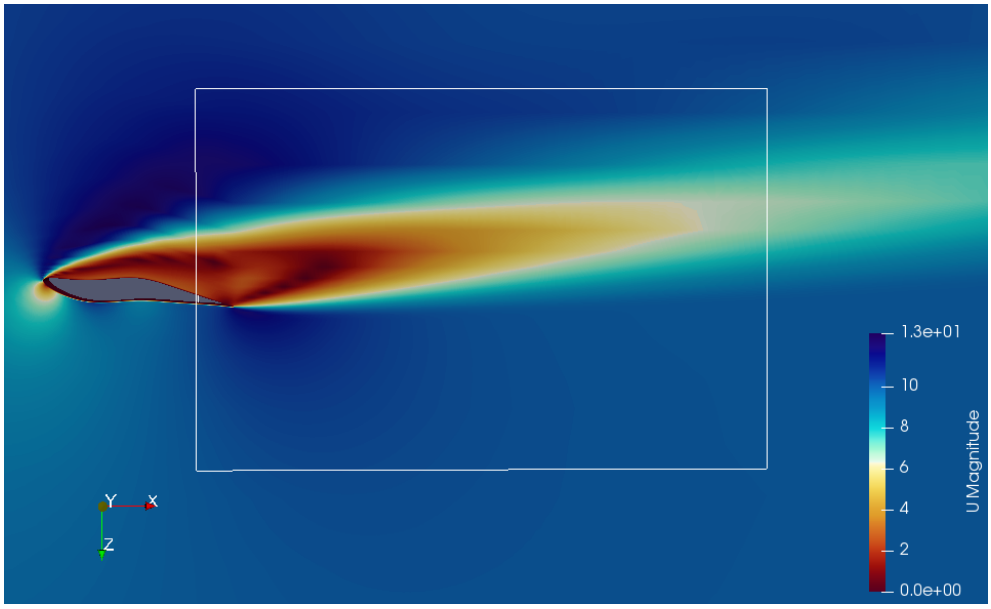


Figure 4.3: Example simulation. The square behind the airfoil is the window used as an example for our pipeline.

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.

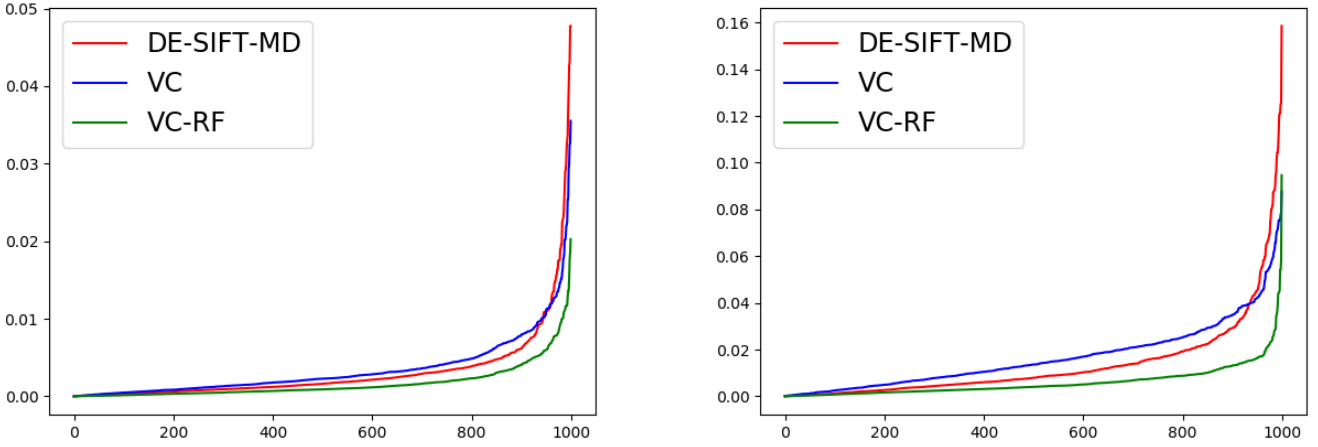
Approach	Drag	Lift
DE-SIFT-MD	0.965	0.987
VC	0.917	0.949
VC-RF	**0.981**	**0.994**

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.