Universiteit Leiden
The Netherlands

Learning from small samples

Kocaman, V.

Chapter 5

The Role of Self-Supervised Learning

5.1 Self-Supervised Learning

5.1.1 Motivation

CV models have typically been trained using supervised learning, a process in which humans provide labels for images in order to help the model recognize patterns. This can be time-consuming, as it requires human annotators to assign class labels or draw bounding boxes around objects in the images. An alternative approach is self-supervised learning (SSL), in which the model is able to learn from the data itself, without the need for human-provided labels.

This is often achieved by applying different image transformations or crops to the same image, and then using the model to learn that these modified images still contain the same visual information. During the SSL training process (Figure 5.1), the augmented version of the original sample is considered as a positive sample, and the rest of the samples in the batch/dataset (depends on the method being used) are considered negative samples. Next, the model is trained in a way that it

learns to differentiate positive samples from the negative ones [187]. While doing so, the model learns quality representations of the samples and is used later for transferring knowledge to downstream tasks.

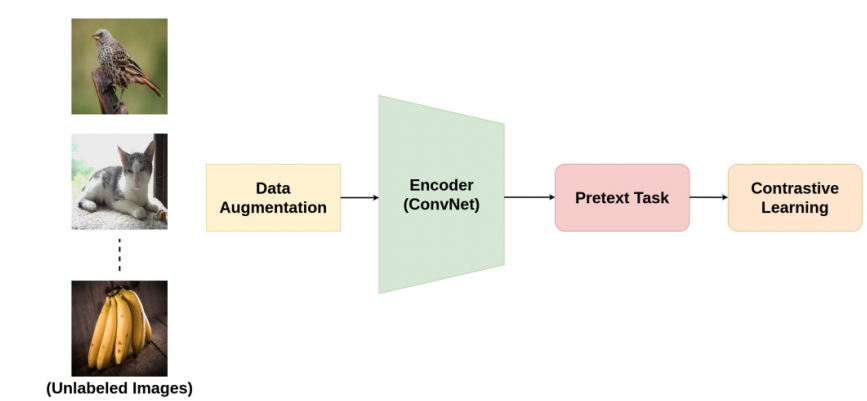

Figure 5.1: Contrastive learning pipeline for self-supervised training (source: [187])

This idea is advocated by an interesting experiment conducted by Epstein in 2016 [188], where he asked his students to draw a dollar bill with and without looking at the bill (from memory) (Figure 5.2). Evidently, the drawing made in the absence of the dollar bill is quite primitive compared with the drawing made from an exemplar, even though the students have seen a dollar bill thousands of times. The results from the experiment show that the brain does not require complete information of a visual piece to differentiate one object from the other. Instead, only a rough representation of an image is enough to do so [187].

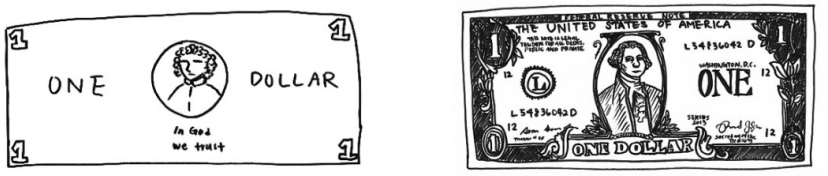

Figure 5.2: Epstein dollar bill experiment: Drawing a dollar bill with and without looking at the bill (source: [188])

This training process allows the model to learn a latent representation (a vector output) for the same objects, which can be useful for downstream tasks such as object detection and semantic segmentation. Transfer learning can then be applied to this pre-trained model, allowing it to be fine-tuned on a smaller labeled dataset to perform specific tasks. As we'll be illustrating clearly in the following sections, models trained with self-supervised learning (SSL) tend to generalize better than their supervised counterparts for transfer learning (see Figure 5.3)

5.1.2 Background

The capacity to discover structure in the world without someone explicitly providing supervision/teaching is considered a human-level capability. Likewise, learning about the structure of data leads to useful representations about the world. Ideally, we'd want neural networks to learn initial structures which have wide applicability and to do so on their own without human annotated labels. On the other hand, supervised learning requires vast amounts of carefully annotated and curated data that actually increases the amount of tedious and laborious jobs in many ways. It's almost impossible to label everything in the world with every possible useful label if we want to achieve a decent generalization power that would lead to zero-shot learning capabilities. We need orders of magnitude more data, to which labeling is going to be very expensive.

The proposed solution to this issue is usually applying a transfer learning, a well-known technique in CV that involves using pre-trained deep convolutional neural networks (DCNNs) as a starting point to learn a new task. These large models are trained on massive supervised datasets, such as ImageNet, and their features have been found to adapt well to new problems. Transfer learning is often used when annotated training data is scarce and involves taking the pre-trained weights of a model, adding a new classifier layer on top, and retraining the network. It has been shown that using pre-trained weights to initialize a model for a new task tends to result in faster learning and higher accuracy compared to training from scratch with random initialization.

However, transfer learning relies on models trained on supervised datasets, which can be costly to create due to the need for annotation. In contrast, unsupervised

data is abundant and can be used to learn representations that can be used to improve supervised models. This makes it a potentially valuable approach for improving the efficiency and effectiveness of transfer learning. Also known as unsupervised learning, unsupervised representation learning is concerned with addressing the following issue: learning good representations from unlabeled data.

Unsupervised representation learning, which involves training a model to learn useful representations of data without the need for labeled examples, has the potential to unlock a number of applications that current transfer learning has not been able to address. However, it has historically been a more difficult problem than supervised representation learning. One example of this is in the task of breast cancer detection, where the best solutions currently rely on ImageNet pre-trained models as a starting point. Despite the significant differences between breast cancer images and regular ImageNet samples, transfer learning still proves effective to some extent in this context. This is because most supervised datasets for breast cancer detection are smaller and have less variability than common CV supervised datasets. However, there is a large number of non-annotated slide images of breast cancer available, and if good representations could be learned from this unsupervised data, it could help to improve specific downstream tasks that have limited annotated data. Fortunately, visual unsupervised representation learning has shown promise, particularly with the use of contrastive-based techniques. These techniques are now able to learn visual representations that are similar to those learned using supervised methods in some self-supervised benchmarks.

There is often confusion surrounding the concepts of unsupervised learning and self-supervised learning. While self-supervised learning can be considered a subcategory of unsupervised learning because it does not involve manually-provided labels, unsupervised learning is more specifically focused on identifying specific patterns in data, such as clustering, community discovery, or anomaly detection. On the other hand, self-supervised learning aims to reconstruct or predict missing information, which is more similar to the goals of supervised learning. In this sense, self-supervised learning can be seen as a hybrid approach that combines elements of both supervised and unsupervised learning [31].

The self-supervised training strategy in CV has so far been inspired by the approach

in natural language processing (NLP) in a sense of pre-training on a vast amount of training data and then fine-tuning on a target data set [189]. Owing to the latest breakthrough in 'attention' based neural network architectures [190], namely 'transformers', no one would argue that the key ingredient of success in the NLP world recently has been the self-supervised pre-training. The pre-training tasks use vast amounts of data available online to create a learning signal and this scale is much better than supervised learning and it doesn't deflate away any of the rich semantic information in the images by projecting onto a single label selected from only a handful of possible categories. Given different representations of the same image we want to learn a representation space where representations from the same image are close together and at the same time far apart from representations of other images. This simple idea is remarkably based on nothing other than similarity, which eventually leads to powerful representations.

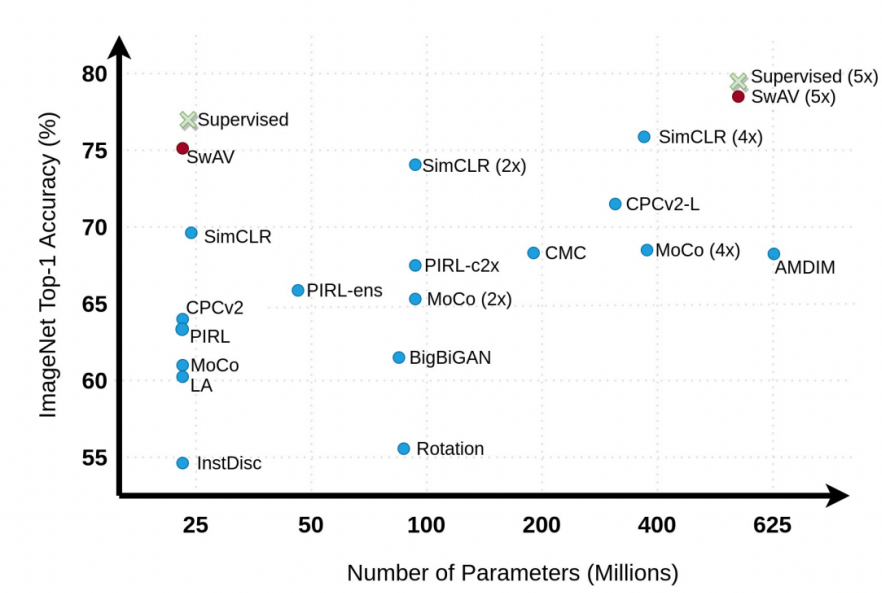

Figure 5.3: Self-supervised representation learning performance on ImageNet top-1 accuracy under linear classification protocol. The self-supervised learning's ability on feature extraction is rapidly approaching the supervised method. (source: [187])

SSL models can be grouped under four categories:

- Generative (autoregressive models (predicting the next token/pixel), non-autoregressive models (masking a token/pixel and predicting the masked token/pixel) flow-based models, auto-encoding (AE) models, and hybrid generative models)
- Predictive tasks (predicting the relative location of the image patches, predicting whether the next fragment is the next sentence, solving the puzzles created from shuffled patches of an image, predicting the cluster id of each sample, predicting the image rotation angle, etc.)
- Contrastive (MoCo, SimCLR, SimSiam etc.)
- Generative-Contrastive (adversarial) (GAN, VQ-VAE2, BiGAN etc.)

5.1.3 Contrastive Self-Supervised Learning

In this dissertation, we will focus more on contrastive SSL approaches. Contrastive learning involves the creation of positive and negative training sample pairs based on the characteristics of the data. The goal is for the model to learn a function that assigns high similarity scores to positive samples and low similarity scores to negative samples. Therefore, it is crucial to properly generate the samples in order to ensure that the model can effectively learn the underlying features and structures of the data.

Forcing the model to generate the supervisory signals out of the data itself, the machine supervision is the process of creating a label but the human supervision is reduced to the process of selecting suitable data sets and data augmentations. We're still adding prototypical human knowledge into the model via the back door but with significantly less onus on the humans. In other words, we're effectively letting the machine itself discover structures and categories by predicting unobserved or hidden relationships either in time or in space. We see similar examples around the same idea in the literature. Recent advances in large language models such as ChatGPT and GPT-4 have demonstrated the potential of machines to discover patterns and relationships in data on their own, paving the way for new approaches to machine learning and artificial intelligence that rely less on human supervision. As we continue to explore these exciting new possibilities, it's clear that the role of

humans in machine learning will continue to evolve and transform in the years to come.

Self-supervised learning in NLP, particularly with BERT [191] and GPT [192], has demonstrated significant success, prompting researchers to explore the application of similar generative approaches in other domains such as image and speech recognition. However, generating masked inputs for image data presents more challenges compared to text data, as it is more difficult to selectively mask a large number of image pixels (it is easier to choose a limited amount of text tokens).

With the help of SSL, in CV we're trying to learn the fundamental property of an image that you should be able to recognize irrespective of many common distortions of images. The idea of using data augmentations to vary individual data points and then differentiating all of those variations from other data points is not new by any means. Exemplar networks for example cast the representation learning problem as a classification problem with every image in the data set representing its own class. Later the idea is refined and the problem is converted from a parametric to a non-parametric classification by using the data points themselves as classification weights.

The idea of Noise Contrastive Estimation (NCE) from the discipline of learning word vectors and NLP is highly used in SSL based models. NCE [193] is a way of learning a data distribution by comparing it against a noise distribution. This allows us to cast an unsupervised problem as a supervised problem. It is similar to Negative Sampling, which is an approximation mechanism that was invented to reduce the computational cost of normalizing network outputs by summing over the entire vocabulary. Negative Sampling [194] is a backbone of word embeddings concept in NLP (learning vector embeddings of words to help with Natural Language tasks) and it aims at maximizing the similarity of the words in the same context and minimizing it when they occur in different contexts. However, instead of doing the minimization for all the words in the dictionary except for the context words, it randomly selects a handful of words depending on the training size and uses them to optimize the objective.

One of the most popular self-supervised frameworks with this simple concept

of NCE, SimCLR [195] delivered another push in performance by dramatically simplifying the architecture and the data augmentations used. Now in the early days of contrastive SSL such as FaceNet and SimCLR, we had explicit positives and negatives. The models would try to distinguish different augmented images by classifying which image it was.

Contrastive learning in the field of CV involves using data augmentation to generate positive sample pairs from a single original image, and using two distinct images as negative sample pairs. Two key factors that can affect the effectiveness of this approach are the strength of the data augmentation and the selection of negative sample pairs. If the data augmentation is too strong, such that the two augmented samples bear no relationship to one another, the model will not be able to learn useful information. Similarly, if the data augmentation is too weak, the model will also not be able to learn effectively. In terms of selecting negative sample pairs, it is important to avoid randomly assigning two images as negative pairs if they are likely to belong to the same class, as this can introduce conflicting noise to the model. At the same time, the negative pairs should not be too easy to distinguish, as this can prevent the model from learning the underlying features and structures of the data.

This approach of contrastive learning, i.e., learning representations by enforcing similar elements to be equal and dissimilar elements to be different, made the strong assumption that everything else that isn't within this particular class is dissimilar. At the same time, this approach was also quite flawed because training a classifier to distinguish between all the possible pairs of images didn't scale very well with the number of images. SimCLR also produced this really clever idea of a projection head where the representations we compare in the objective are not the ones we actually use downstream. This means that we can effectively learn a broad similarity but maintain fine grained representations. The projectors may have as many parameters as the backbones themselves.

It's a really interesting question if we just want to turn an image into a vector that we could then train a linear classifier on top of what is the best way of doing that. The "bootstrap your own latent", or BYOL method [196], moved away from comparing different images and instead focused on metric learning.

Features are trained by comparing them to a momentum encoder and using a much smaller batch size. It turned out that the momentum encoder wasn't even that useful. What's clear from looking at the self-attention masks is that the self-supervised representations are far richer than the supervised ones even though the supervised ones typically have slightly better accuracy. Supervised classification is like cheating on an exam by knowing what the questions will be in advance and all the model needs to know is how to pass that particular exam. This is the so-called shortcut rule where we optimize on a particular metric at the cost of everything else. We already know from the No Free Lunch theorem [197] that such highly specific accuracy comes at the cost of generalization. Self-supervised learning can bypass such specific and narrow objectives that perform well at specific downstream tasks and instead build much more broadly applicable and general purpose representations.

Data augmentation is the secret and the most important ingredient in making self-supervised learning work so well. The data augmentation that we need for self-supervised learning is different from the data augmentation that we need for supervised learning. When we have contrastive learning, we have this problem which is that if there's only one dominant feature in your data that can be used to perform the contrastive task. That would be the only feature that the network would ever learn. So with augmentation we're somewhat making the task harder so that the network actually has to learn many different kinds of features.

In a nutshell, the data augmentation techniques that we do are different for SSL compared to supervised learning. For instance, aggressive color distortion is quite important in self-supervised learning. This is because in the self-supervised setting where we're comparing different augmented representations from the same image it would be easy for the model to overfit on the color histogram instead of learning the richer information. If we think about it, not all trees in nature have the same color histogram but in SSL all of the individual images will have the same color histogram. Data augmentation also serves as a proxy for what might happen. In real life we're an agent interacting with the environment going around all of these different places and we see objects in a variety of different circumstances and we know different transformations being applied to our viewpoint.

In this section, we will briefly explore the popular SSL algorithms to give a rough idea about how each one of them tackles the problem of extracting the signals from unlabelled data. You can find a detailed survey of SSL methods in [187].

- SimCLR [195]
- MoCo [198]
- SimSiam [199]
- BYOL [196]
- SwAV [200]
- Barlow Twins [201]
- DINO: [202]

BYOL and Barlow Twins in this list are considered as non-contrastive SSL methods.

5.1.3.1 SimCLR: A Simple Framework for Contrastive Learning of Visual Representations

There has been a proliferation of SSL approaches for learning image representations in recent years, with each method showing incremental improvements over previous ones. However, the performance of these methods has generally remained inferior to that of supervised learning methods. This changed with the publication of the SimCLR paper [195], which demonstrated not only superior performance over previous state-of-the-art self-supervised learning methods, but also outperformed supervised learning methods on ImageNet classification when using a larger model architecture.

The SimCLR framework is based on a straightforward concept: a single image is transformed through various random modifications to generate a pair of augmented images. These images are then processed through an encoder, resulting in representations that are further transformed by a non-linear fully connected layer. The objective of the framework is to maximize the similarity between the two resulting representations for a given image (see Figure 5.4).

The training flow can be described as follows:

1. Obtain an input image
2. Apply two random augmentations to the image, including transformations such as rotations, changes in hue/saturation/brightness, zooming, and cropping, etc. (see [195] for the entire list and ranges).
3. Use a deep neural network (preferably a convolutional one, such as ResNet50) to generate image representations (embeddings) of the augmented images.
4. Utilize a small, fully connected linear neural network to project the embeddings into another vector space.
5. Compute the contrastive loss and apply backpropagation through both networks. The contrastive loss decreases when the projections derived from the same image are similar (e.g. have a high cosine similarity).

The contrastive loss decreases when the projections of augmented images that were derived from the same input image are similar. For two augmented images (i) and (j) that were generated from the same input image, the contrastive loss for (i) tries to identify (j) among the other images in the same batch.

5.1.3.2 MoCo: Momentum Contrast for Unsupervised Visual Representation

MoCo was first presented in [198] in 2020 and further improved to MoCo v2 in [203]. There is another iteration called MoCo v3 in [204] which introduces a fundamental adaptation of the training process and the model architecture. In the following paragraphs, We will first discuss the original ideas presented in the original implementation [198], followed by a review of the improvements made in subsequent versions as the method has been continually refined.

In the MoCo paper, the unsupervised learning process is framed as a dictionary look-up: Each view or image is assigned a key, just like in a dictionary. This key is generated by encoding each image using a convolutional neural network, with the output being a vector representation of the image. Now, if a query is presented to

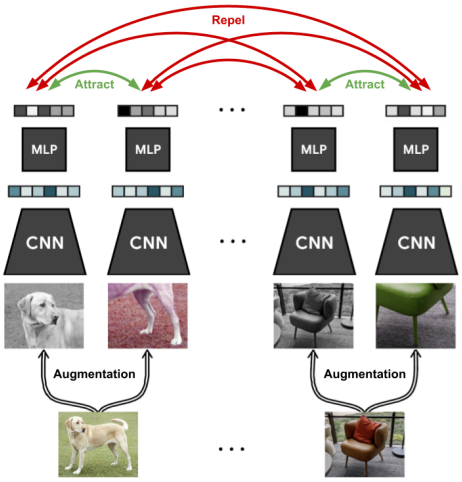

Figure 5.4: An illustration of the SimCLR training procedure. SimCLR employs contrastive learning to optimize the consistency between two augmented versions of the same image. The aim is to increase the similarity between these two augmented representations of the same image. (source: https://simclr.github.io/)

this dictionary in the form of another image, this query image is also encoded into a vector representation and will belong to one of the keys in the dictionary, the one with the lowest distance.

In the original study, the unsupervised learning process is described as a dictionary look-up: Each view or image is assigned a key, similarly to the way words are assigned definitions in a dictionary. The key is generated by encoding each image using a convolutional neural network, which produces a vector representation of the image. If a query, in the form of another image, is presented to this dictionary, it is also encoded into a vector representation and will be associated with the key in the dictionary that has the lowest distance from it.

The training process works as follows (see Figure 5.5): A query image is selected and processed by the encoder network to compute the encoded query image (q). The goal of the model is to learn to differentiate between a large number of different images, so the encoding of the query image is compared to not just one mini-batch of encoded key images, but to multiple mini-batches. To do this, the MoCo algorithm maintains a queue of mini-batches that are encoded by the momentum encoder

network. When a new mini-batch is selected, its encodings are added to the queue and the oldest encodings are removed. This allows the model to query from a much larger dictionary, decoupling the dictionary size (represented by the queue) from the batch size. If the encoding of the query image matches a key in the dictionary, it is determined that the two views are from the same image (e.g. different crops of the same image). If the encoding of the query image is found to be similar to a key in the dictionary, it can be inferred that the two views are from the same image (e.g. different crops of the same image).

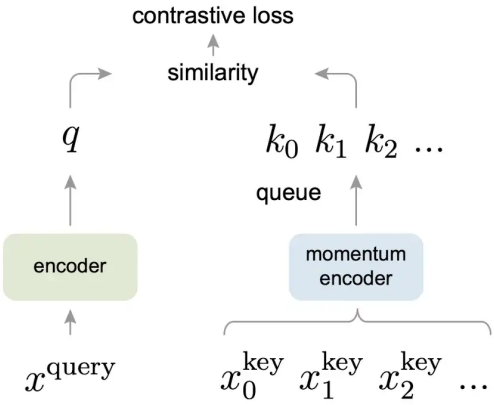

Figure 5.5: The encoder to compute the encoded query image is depicted on the left, and the queue of mini-batches of keys and the momentum encoder is on the right [198].

This loss function enables the model to learn to assign smaller distances to views from the same image and larger distances to views from different images. The model is trained using backpropagation through the encoder network. The authors aim to obtain a consistent encoding for each image using the momentum encoder. To prevent the weights from becoming frozen and not reflecting the progress of the model's learning, they propose using a momentum update instead of copying the encoder's weights. This is achieved by setting the momentum encoder network to be updated using a momentum-based moving average of the query encoder for each training iteration.

In MoCo v2 [203], the authors incorporate some of the ideas from the SimCLR paper into their model. Previously, for training a linear classifier after pre-training

MoCo, they had added a single linear layer to the model. In MoCo v2, this layer has been replaced with an MLP projection head, resulting in improved classification performance. Additionally, stronger data augmentations similar to those used in SimCLR have been introduced, including the addition of blur and stronger color distortion to the v1 augmentations. In the latest version, MoCo v3, a different training process is adopted to improve the architecture.

5.1.3.3 SimSiam: Simple Siamese Representation Learning

In previous approaches, the use of large numbers of negative sample pairs, which required large batch sizes, resulted in high computational and memory demands. Some prior approaches addressed this issue by employing momentum encoders as a workaround to avoid the need for large batch sizes. SimSiam aims to eliminate these requirements and create a simple SSL framework.

SimSiam is a proposed simple Siamese network that is able to learn meaningful representations without using negative sample pairs, large batches, or momentum encoders. The Siamese structure is a key factor in the overall success of the considered baselines and a stop-gradient operation plays a crucial role in preventing collapse. It does not require a large batch size in order to operate, unlike SimCLR, and performs better than SimCLR in most of the cases.

[205] says that the performance actually degrades as the number of negatives is increased for a fixed batch size and using fewer negatives can lead to a better signal-to-noise ratio for the model gradients. This could explain the SimSiam's improved performance with respect to SimCLR since large negative samples or large batch sizes to ensure large negative samples are not needed in SimSiam.

SimSiam follows a standard contrastive learning setup in which two networks with shared weights are presented with two different augmented views of the same image, and an MLP projection head transforms one view to match the other. Notably, the contrastive loss is symmetrical: the output of one of the networks is treated as constant (using stop-gradient), and the output of the second network is projected to match it. The roles of the two networks are then reversed, with the output of the first network projected and matched to the "constant" output of the second network. The two losses are averaged, and each of the networks receives gradient

for the loss term, with stop-gradient not applied to its output.

In a nutshell, given an image, we create two augmented views and process these two versions with the same encoder network (a backbone plus a projection MLP). Then, we maximize the similarity at the end by minimizing the negative cosine similarity to optimize the parameters.

5.1.3.4 BYOL: Bootstrap Your Own Latent

Contrastive methods are less expensive as described by the authors of BYOL [196]: "Contrastive approaches avoid a costly generation step in pixel space by bringing representation of different views of the same image closer ('positive pairs'), and spreading representations of views from different images ('negative pairs') apart."

Having to compare each sample to many other negative samples is problematic, because it introduces instabilities into the training, and reinforces systematic biases from the dataset. Here is how this phenomenon es clearly explained in [196]: "Contrastive methods are sensitive to the choice of image augmentations. For instance, SimCLR does not work well when removing color distortion from its image augmentations. As an explanation, SimCLR shows that crops of the same image mostly share their color histograms. At the same time, color histograms vary across images. Therefore, when a contrastive task only relies on random crops as image augmentations, it can be mostly solved by focusing on color histograms alone. As a result the representation is not incentivized to retain information beyond color histograms."

Unlike other contrastive learning methods, BYOL achieves state-of-the-art performance without requiring negative samples. Like a siamese network, BYOL uses two identical encoder networks, referred to as the online and target networks, to obtain representations and minimizes the contrastive loss between the two representations.

The goal of BYOL is to ensure that similar samples have similar representations, while contrastive learning aims to ensure that dissimilar samples have dissimilar representations. Training with BYOL is more efficient, as it does not require negative sampling and only samples each training example once per epoch. This can lead to better generalization of the model, as it is less sensitive to systematic biases

in the training dataset. BYOL minimizes the distance between the representation of each sample and a transformed version of that sample, such as through translation, rotation, or blurring.

Since BYOL does not require negative sampling, the training process is more efficient as we only sample each training example once per epoch. The negative counterparts can be ignored altogether. In return, the model is less sensitive to systematic biases in the training dataset, leading to better generalizations on unseen examples.

5.1.3.5 SwAV: Swapping Assignments between multiple Views of the same image

As explained before, contrastive learning methods involve the direct comparison of features from different transformed versions of the same images. In contrast, SwAV does not directly compare image features. Instead, it involves the creation of intermediate targets by assigning image features to prototype vectors and solving a "swapped" prediction problem, in which the targets are altered for two different views of the same image. Figure 5.6 illustrates the difference between contrastive instance learning and swapping assignments between the views.

SwAV is more efficient because it does not require a large memory bank or an auxiliary momentum network. Instead of directly comparing features, it clusters the data and enforces consistency between cluster assignments for different augmentations of the same image simultaneously. In other words, it uses a "swapped" prediction mechanism where the cluster assignment of a view is predicted from the representation of another view. This method can be trained with large and small batches and can handle an unlimited amount of data.

One of the important contributions of the original SwAV paper is a new data augmentation strategy called 'multi-crop', that uses a mix of views with different resolutions in place of two full-resolution views, without increasing the memory or compute requirements. Hence, not only two larger crops are created, but also up to 4 smaller crops are taken. This is one of the keys for why SwAV can boost its performance in comparison to other approaches [200].

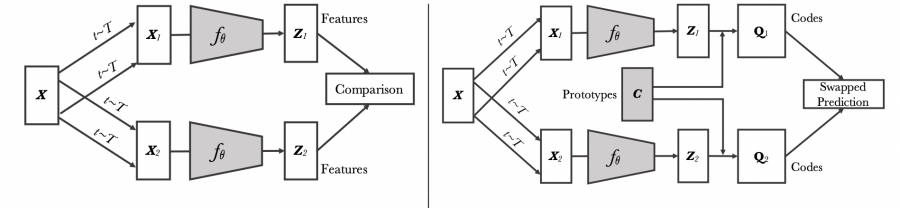

Figure 5.6: In contrastive learning methods applied to instance classification, the features from different transformations of the same images are compared directly to each other. In SwAV, the codes obtained from one data augmented view are predicted using the other view. Thus, SwAV does not directly compare image features [200].

5.1.3.6 Barlow Twins: Self-Supervised Learning via Redundancy Reduction

Recent research in SSL aims to learn embeddings that are invariant to distortions of the input sample. This is typically accomplished by applying augmentations to an input sample and attempting to make their representations as similar as possible. However, a common issue is that there are often trivial constant solutions that the network can rely on, resulting in no meaningful learning.

Similarly to other SSL methods, Barlow twins, aims at learning representations (embeddings) that are invariant to input distortions. An important detail is that the encoder is actually made up of an encoder network followed by a "projector" network. The encoder executes feature extraction while the projector designs the embedding's high dimensional space. Representations learned are used to train classifiers on downstream tasks, while embeddings are fed into the loss function to train the model.

The objective function in the Barlow Twins method is designed to prevent collapse by measuring the cross-correlation matrix between the outputs of two identical networks fed with distorted versions of a sample. This causes the embedding vectors of distorted versions of a sample to be similar while minimizing redundancy between the components of these vectors. This method does not require large batches or asymmetry between the network twins, such as a predictor network, gradient stopping, or a moving average on weight updates. Interestingly, it benefits

from very high-dimensional output vectors. It outperforms previous methods on ImageNet for semi-supervised classification in the low-data regime and performs comparably to the current state-of-the-art for ImageNet classification with a linear classifier head and for transfer tasks of classification and object detection. The Barlow twins method either outperforms or performs comparably to most other self-supervised learning architectures on a variety of computer vision downstream tasks [201].

5.1.3.7 DINO: Emerging Properties in Self-Supervised Vision Transformers

DINO works by interpreting self-supervision as a special case of self-distillation, where no labels are used at all. The idea is to train a student network by simply matching the output of a teacher network over different views of the same image. The authors of DINO [202] identified two components from previous self-supervised approaches that are particularly important for strong performance on the Vision Transformer (ViT) – the momentum teacher (MoCo) and multi-crop training (SwAV) – and integrated them into DINO framework. The resulting model achieves state-of-the-art performance above all previously proposed self-supervised systems, clearly depicting the potential of ViTs for self-supervised learning.

In DINO, there is a teacher and student network, both having the same architecture, a Vision Transformer (ViT). The teacher is a momentum teacher, which means that its weights are an exponentially weighted average of the student's. During training, different crops of one image are taken. Small crops are called Local views and large crops are called Global views. All crops are passed through the student, while only the global views are passed through the teacher. Additionally, random augmentations are applied on the views to make the network more robust. The Cross entropy loss is applied to make the student's distribution match the teacher's, allowing the student network to have the same proportions of features as the teacher. The teacher, having a larger context, predicts more high-level features, which the student must also match.

To ensure the teacher's knowledge is not lost, the stop-gradient operator is applied on the teacher to propagate gradients only through the student. The teacher param-

eters are updated with an exponential moving average of the student parameters. This process is aimed at making the network address a classification problem such that it can learn meaningful global representations from local views. To prevent mode collapse two techniques are used: Centering and Sharpening. In centering, the teacher's raw activations have their exponentially moving average subtracted from them. In sharpening, temperature is applied to the softmax to artificially make the distribution more peaked. DINO learns a significant amount about the visual world by identifying object parts and shared characteristics across images, resulting in a feature space with a structured and interpretable organization of similar categories. This allows for efficient k-NN classification without the need for fine tuning or learning classifiers. When compared to state-of-the-art SSL techniques on the task of ImageNet classification, DINO consistently demonstrates superior performance across different throughput regimes. The original DINO paper [202] also shares some interesting findings and observations regarding the zero shot object segmentation capacity of DINO. DINO has shown to have understood object semantics so well that its attention maps look like segmentation masks. Figure 5.7 shows such an example in which a ViT network trained with DINO with no supervision automatically learns class-specific features and able to segment the objects better than a network that is trained with full supervision.

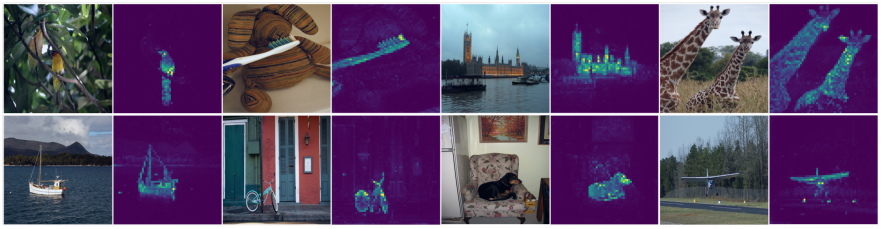

Figure 5.7: Self-attention from a ViT with 8×8 patches trained with no supervision automatically learns class-specific features leading to unsupervised object segmentations [202].

Other than the ones covered above, there are many other SSL algorithms proposed so far. Even though the fundamental idea behind each of these methods looks similar, there might be some nuances in their implementation due to the tasks they're trying to solve. Figure 5.8 shows the fundamental differences between siamese based SSL algorithms in one visual.

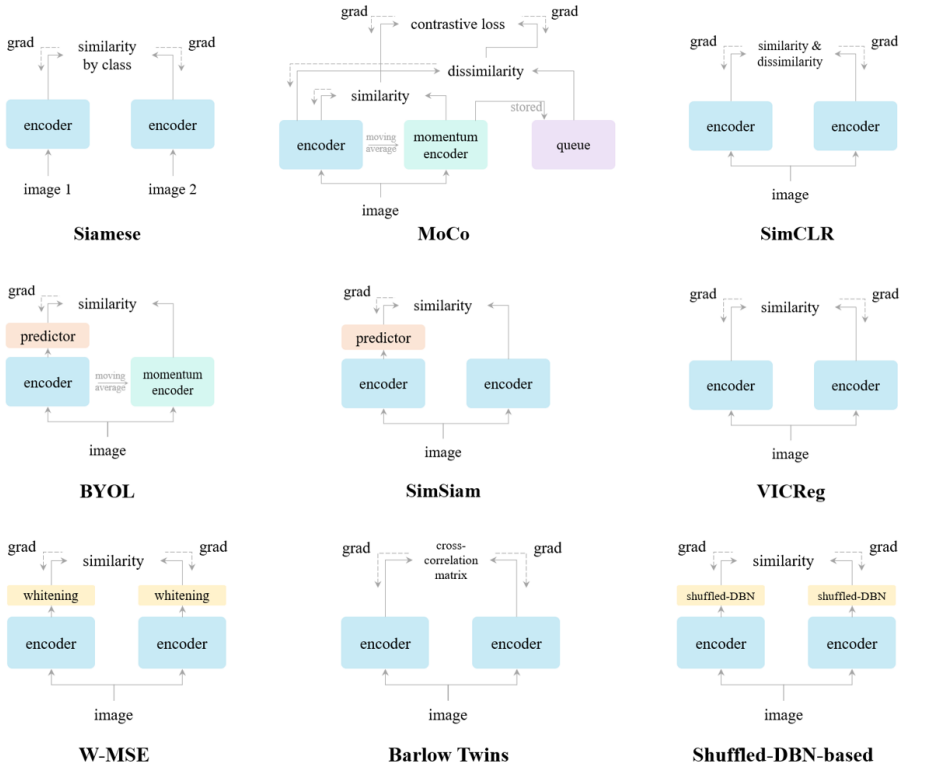

Figure 5.8: Comparison on Siamese architecture-based SSL frameworks. The encoder includes all layers that can be shared between both branches. The dash lines indicate the gradient propagation flow. Therefore, the lack of a dash line denotes stop-gradient [206].

5.2 Preliminary: SSL applied to small subsets of Plant Village data

In order to assess the effectiveness SSL on small data, we run some experiments with SimCLR on Plant Village (PV) dataset that we explored in detail in chapter 4. Here we applied the following steps:

1. Train a supervised image classifier with ResNet34 on the entire PV dataset **using the labels**, drop the fully connected (FC) classifier (head) layers, use the backbone network as a feature extractor, collect feature embeddings of a small sample of PV dataset (Apple class, 10 healthy, 10 unhealthy), train a

simple classifier model (backbone network plus a couple of FC dense layers as a head classifier) on this small dataset to classify healthy and unhealthy classes, and then test on a comparatively larger dataset (Apple class, 150 healthy, 150 unhealthy).

2. Train a self-supervised model with SimCLR (backbone ResNet34) model on the entire PV dataset **without using the labels**, use the backbone network as a feature extractor and do the rest as described above.

3. Repeating the same with G-SimCLR (backbone ResNet34) model with and without multi-crop augmentation on a small sample of PV dataset (Apple class, 10 healthy, 10 unhealthy).

4. Repeating the same with SwAV over various set of small samples (Apple class 20 healthy vs 10 unhealthy, 5 healthy vs 5 unhealthy etc.).

Before delving into the results of these experiments, it is worth to explain the underlying concept behind G-SimCLR, Self-Supervised Contrastive Learning with Guided Projection via Pseudo Labelling [207]. As mentioned before, SimCLR aims to maximize the similarity between the two resulting representations for a given image while also minimize the similarity between the very same image and then resulting representations from the other images. It's quite probable that there might be near-duplicate or similar images already appearing in the same dataset and SimCLR would still try to treat even the near-duplicate images as negative pairs of one another, causing high loss that slows down the convergence. Even if the negative impact of this phenomenon is slightly prevented by large batches and large amount of unlabelled images, it would still be undesirable when we have small amount of images.

In order to attack this, G-SimCLR proposes a method for generating information about the label space without using explicit labels, and then using this information to improve the quality of representations learned through SimCLR. This approach involves training a denoising autoencoder in a self-supervised manner by minimizing the reconstruction loss between the input and noisy images, and using the latent space representations from the trained autoencoder to cluster the input space of images and assign each image to a corresponding cluster, referred to as pseudo

labels. In simple terms, the image representations from encoder are clustered and the batches are created by picking images from different (dissimilar) clusters. These pseudo labels are then incorporated into the determination of training batches for SimCLR to reduce the risk of images from the same label space being included in the same batch. The steps applied in G-SImCLR can be seen at Figure 5.9 and the cluster visualizations (with tSNE) can be seen at Figure 5.10.

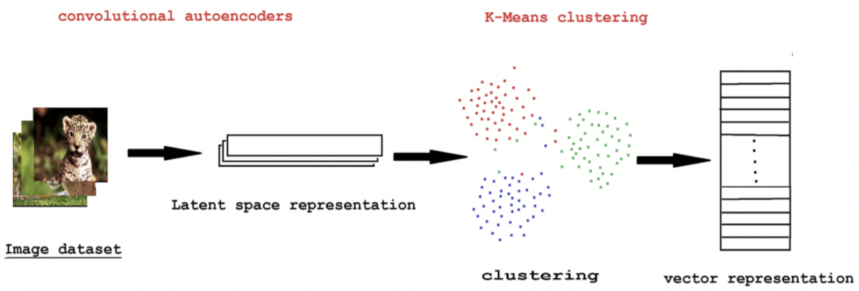

Figure 5.9: In G-SimCLR, the latent space representations of all the images in the dataset is clustered and then the batches to feed in SimCLR training process are generated via these clusters to reduce the risk of images from the same label space being included in the same batch.

The results from these experiments are depicted in Figure 5.11. These results demonstrate that the G-SimCLR method with multi-crop augmentation achieves the highest accuracy (0.9113) among all the experiments; even higher than what we get from the approach that we collect the feature embeddings through the backbone of fully supervised trained network (0.8767). This indicates that SSL would be a strong player for dealing with small datasets where the labelled samples are scarce or expensive to gather.

5.3 Salient Image Segmentation as a Surprising Player in SSL

5.3.1 Introduction: Salient Image Segmentation

Salient image segmentation is a CV task that involves identifying the most prominent or eye-catching objects or regions in an image. It plays a crucial role in

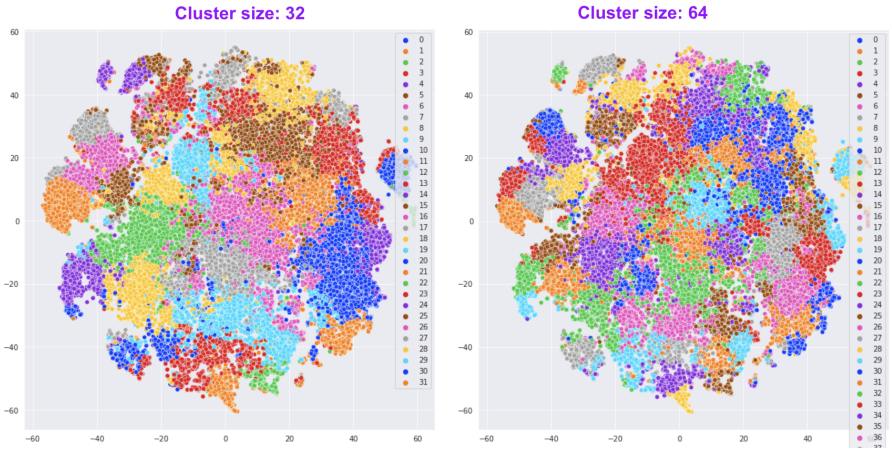

Figure 5.10: The visualizations of clusters that is generated by the latent space representations (vectors) of all the images in the PV dataset show that there are several distinct clusters that we can leverage to form the batches accordingly.

various image and video analysis applications such as object recognition, scene understanding, and image retrieval. The goal of salient image segmentation is to automatically highlight the most important regions in an image, which may include objects, backgrounds, or even textures. These regions are typically characterized by their visual distinctiveness, such as their color, shape, size, or texture.

There are several approaches to salient image segmentation, including bottom-up approaches that rely on low-level image features and top-down approaches that leverage high-level context and prior knowledge. In recent years, there has been a significant amount of research on developing deep learning-based approaches for salient image segmentation, which have shown promising results on various benchmark datasets [208]. Despite the progress made in the field, salient image segmentation remains a challenging task due to the variability and complexity of visual scenes. It is also a highly dynamic task, as the saliency of an image can change depending on the context and the viewer's focus of attention.

We propose an augmentation policy for Contrastive SSL in the form of an already established Salient Image Segmentation technique entitled Global Contrast based Salient Region Detection. This detection technique, which had been devised for

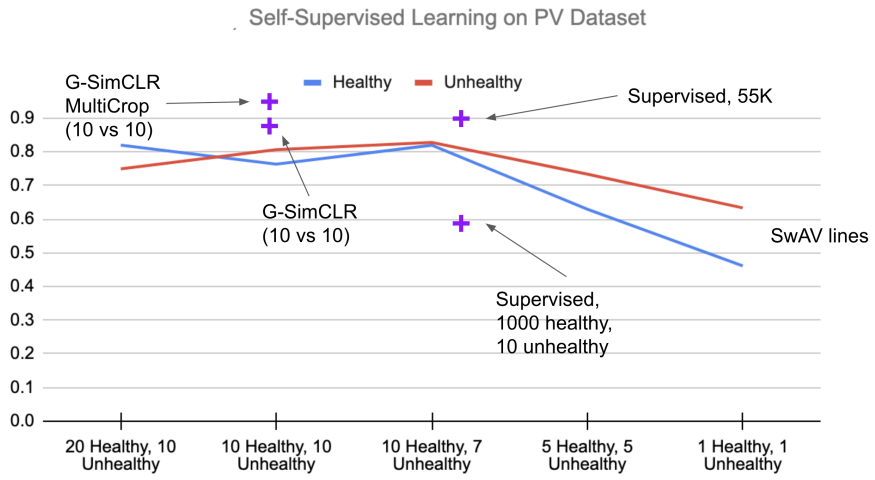

Figure 5.11: The chart above shows the performance of various approaches for classifying healthy and unhealthy plant leaves using different numbers of labeled and unlabeled samples. The x-axis represents the number of samples from both healthy and unhealthy leaves in the test set, while the y-axis indicates the classification accuracy achieved by each method. The performance of each approach was evaluated using a 10-epoch SSL training process on the entire Plant Village (PV) dataset (size 55K) without any labels, followed by a logistic regression classification for 50-epochs using reduced samples. The results show that G-SimCLR with multicrop augmentation and only 10 labeled samples from each class performed the best, outperforming both a fully supervised version using the entire 55K labeled dataset and a supervised training approach with 1000 healthy samples and 10 unhealthy samples. In addition, SwAV was also found to perform better than the supervised approach, but not better than G-SimCLR.

unrelated Computer Vision tasks, was empirically observed to play the role of an augmentation facilitator within the SSL protocol. This observation is rooted in our practical attempts to learn, by SSL-fashion, aerial imagery of solar panels, which exhibit challenging boundary patterns. Upon the successful integration of this technique on our problem domain, we formulated a generalized procedure and conducted a comprehensive, systematic performance assessment with various Contrastive SSL algorithms subject to standard augmentation techniques. This evaluation, which was conducted across multiple datasets, indicated that the proposed technique indeed contributes to SSL. We hypothesize whether salient image segmentation may suffice as the only augmentation policy in Contrastive SSL when treating downstream segmentation tasks.

Despite recent advances in CV, the effectiveness of visual recognition and learning still very much depends on manual annotations and on how well they represent the images at hand. Given the substantial amount of available unlabeled data and the costly manual annotation, new methods for online and unsupervised learning are crucial for achieving robust and efficient visual learning. Current unsupervised learning methods do not perform learning under a genuine unsupervised state – they rather focus on transfer learning, or unsupervised tasks subject to strong features that were pre-trained in a supervised way. Therefore, there is a need to investigate algorithms for unsupervised learning of completely new categories, such as SSL to learn in a continuous, long-term fashion. Indeed, SSL needs also to generalize the classical unsupervised learning scenario of physical objects to scenarios of higher-level actions and more complex activities, and at the same time develop capabilities to detect abnormalities in visual data [209], [210].

The requirement for very large datasets of manually labeled instances may seem counter-intuitive since this is not how humans learn to recognize new objects. Humans are constantly fed with images through their eyes, and are able to learn an object's appearance and to distinguish it from other objects without knowing what the object exactly is. Moreover, collecting large-scale datasets is time-consuming and expensive, while the supervised approach to learn features from labeled data has almost reached its saturation due to the intense labor required in manually annotating millions of data instances. This is because most of the modern CV systems (that are supervised) try to learn some form of image representations by finding a pattern that links the data points to their respective annotations in large datasets [187]. Moreover, the data annotation efforts vary from task to task, and it is estimated that the time spent on image segmentation and object detection (i.e., carefully drawing boundaries) is four times longer than the image classification itself [211]. The annotation efforts become significantly higher when it comes to highly regulated and specialized domains like medicine and finance in which the expertise level of the human annotator matters more than in any other domain. Moreover, supervised learning not only depends on expensive annotations but also suffers from other drawbacks such as generalization errors, spurious correlations, and being prone to adversarial attacks [31].

In this section, we explore the viability of using an unsupervised image segmentation

technique called 'Global Contrast based Salient Region Detection (SGD)' [100] as an augmentation policy (rather than a proxy task) in contrastive SSL methods and study its impact on downstream supervised image segmentation tasks in low data regimes. To the best of our knowledge, this is the first study that explores salient object segmentation techniques as an augmentation policy in contrastive SSL.

We will show that using this SGD algorithm as an image augmentation policy in SSL produces better representations for downstream image segmentation tasks when compared to default augmentation policies commonly utilized in SSL methods. In order to make the integration of SGD into SSL pretraining routines feasible, we also devise a simple manipulation called offline augmentation with hashing that enables running comprehensive experiments with various parameters and configurations. We will also provide evidence that SGD-based augmentation policy in SSL performs better with low resolution images.

The current section targets the following *research questions*:

> Would SGD produce better representation when used as an augmentation policy in Contrastive SSL? Moreover, would salient image segmentation suffice as the only augmentation policy in Contrastive SSL when treating downstream segmentation tasks?

The concrete contributions of this section are the following:

- Comparing a relatively old unsupervised image segmentation technique, Global Contrast based Salient Region Detection (SGD), with recent deep learning (DL)-based image segmentation algorithms.
- Evaluating the viability of a generative model (Pix2Pix) as a proxy for computationally expensive image augmentation methods.
- Devising an SGD-based efficient offline augmentation technique to incorporate any expensive augmentation policy in DL training routines.
- Employing SGD as an offline augmentation policy (rather than a proxy task) in contrastive SSL methods and study its impact on downstream supervised image segmentation and object detection tasks.

- Illustrating that fine-grained details in high resolution images would negatively impact the performance of SSL when compared to the coarse-grained details in low resolution images.

- Formulating a recommendation for choosing the most appropriate SSL method (accounting for the the augmentation technique, downstream task and even the imagery resolution).

The remainder of the section is organized as follows: Section 5.3.2 describes the concrete motivation that ignited this research, and then summarizes related work as well as various SSL approaches, including the role of data augmentation therein. It concludes with presenting the SGD method in detail. Section 5.3.4 outlines the experimental setup regarding SGD, including the datasets the various preliminary efforts to make SGD-based augmentation policies viable in DL training routines. It then presents the attained results. Section 5.3.5 discusses the findings and proposes possible *mechanistic* explanations, and finally Section 5.3.6 concludes by pointing out the key points and future directions.

5.3.2 Background

Explicitly, SSL is a machine learning process where the model trains itself to learn one part of the input from another part of the input. In other words, the model learns from labels that are presumably already intrinsic in the data itself. This process is also known as predictive or pretext learning and the unsupervised problem is transformed into a supervised problem by auto-generating the labels. To effectively exploit the huge quantity of unlabeled data, it is crucial to set the right learning objectives, in order to obtain the appropriate supervision from the data itself.

Transfer learning (TL) has been presented as an effective solution for constructing robust feature representations when the training set for a given problem is small. As its name suggests, TL aims to transfer knowledge and learned features from one task (the source task) to another related target task, just as a person can utilize the same knowledge across different projects. To do this, TL trains the model on a large labeled dataset and then treats this model as a starting point in the target task's training, without learning from scratch. This dataset creates the

target task's representation model, using the same architecture as the source task. The initialized representation network in the target task is then further trained on the target dataset. So we can say that the workflows of SSL and TL are similar, with only slight differences. The key difference between TL and SSL is that TL pre-trains on labeled data, whereas SSL utilizes unlabeled data to learn features and need only a small number of labelled example [212].

Within the context of utilizing unlabeled data to learn the underlying representations, SSL can be organized as (i) handcrafted pretext tasks-based, (ii) contrastive learning-based and (iii) clustering learning-based approaches. In handcrafted pretext tasks-based method, a popular approach has been to propose various pretext tasks that help in learning features using pseudo-labels while the networks can be trained by learning objective functions of the pretext tasks and the features are learned through this process [187]. Tasks such as image-inpainting [213], colorizing gray-scale images [214], solving jigsaw puzzles [215], image super-resolution [216], video frame prediction [217], audio-visual correspondence [218], to mention the most prominent, have proven to be effective for learning good representations. In doing so, the model learns quality representations of the samples and is used later for transferring knowledge to downstream tasks. The selection of an appropriate proxy task (pretext) is critical to the effectiveness of self-supervised learning. It requires careful design, and indeed, numerous researchers investigated various approaches for given downstream tasks. For example, [219] proposed a proxy task to classify whether a given pair of kidneys belong to the same side; with the assumption that the network needs to develop an understanding of the structure and sizes of the kidneys.

On the other hand, contrastive learning (CL) is a training method wherein a classifier distinguishes between "similar" (positive) and "dissimilar" (negative) input pairs. It is essentially the task of grouping similar samples closer to each other, unlike setting diverse samples far from each other. During training, the augmented version of the original sample is considered as a positive sample, and the rest of the samples in the batch/dataset (depends on the method being used) are considered negative samples. Next, the model is trained in a way that it learns to differentiate positive samples from the negative ones. Constructing positive and negative pairs via data augmentation allows the model to learn from inter-class variance

(uniformity) by pushing negative pairs far away, and from intra-class similarity (alignment) by pulling positive pairs together. The core concept is to maximize the dot product between the feature vectors which are similar and minimize the dot product between those of which are not similar. CL methods use contrastive loss that is evaluated based on the feature representations of the images extracted from an encoder network. As briefly explained in section 5.1.3, the popular methods that recently started to produce results comparable to the state-of-the-art supervised learning methods, even with less labelled images, can be regarded as DINO [202], SwAV [200], MoCo [198], [203] , BYOL [196], SimSiam [199] and SimCLR, [220], [195]. The representation extraction strategies differ from one method to another (e.g. BYOL does not even need negative pairs) but the changes are very subtle and without rigorous ablations, it is hard to tell which one works better on a case at hand. The widely used approach to evaluate the learned representations through the SSL's pre-training process is the linear evaluation protocol [221], where a linear classifier (e.g., SVM, Logistic Regression, etc.) is trained on top of the frozen backbone network (image representations are derived from the final or penultimate layers of the backbone network as a feature vector). Finally, the test accuracy is used as a proxy for representation quality (Figure 5.12).

Data augmentation is critical for effective learning in contrastive SSL methods, and the composition of multiple data augmentation operations is key to defining effective contrastive prediction tasks that yield high-quality representations. Stronger data augmentation is particularly beneficial for contrastive SSL when compared to supervised learning, with techniques such as color distortion and cropping playing a key role in producing effective views for the considered dataset [195]. These findings underscore the importance of carefully selecting and combining data augmentation techniques when developing contrastive SSL models for a wide range of applications.

Augmentations can be regarded as an indirect way to pass human prior knowledge into the model, and an effective augmentation should discard the unimportant features for the downstream task (i.e., removing the "noise" to classify an image). The nature of the selected augmentations should fit the downstream task, maintain the image semantics (meaning), introduce the model with challenging assignments/tests. Their selection depends upon the dataset's underlying distribution and cardinality, whereas a recent study [222] further claims that augmentations

are problem-class-dependent.

Let us add more clarity to crafting an augmentation policy that should give the model a hard time and why its selection is dependent on the dataset's diversity and size. In contrastive SSL, we feed pairs/different views of the same image to the network and these views should be visually dissimilar in a way that the model should have a hard time while bringing them together if the views are coming from the same image or pushing them apart if they are coming from different images. If a weak augmentation is applied, it will be an easy job for the network to puss/pull; hence the learning will not be strong enough. Dataset's diversity and size play a role here as the more diverse the dataset, the easier it gets for the model to push dissimilar views (in low diversity datasets in which there many similar images, it is harder to push dissimilar views).

The most popular data augmentation techniques in contrastive SSL include rotation, crop, cut, flip, color jitter, blur, Gaussian noise and gray scale. Almost all of the contrastive SSL methods use these (or similar) augmentation techniques at certain degrees no matter what the downstream task is. Selection of the most suitable augmentation policy is a crucial step in contrastive SSL. [195] systematically studied the impact of data augmentation and observed that no single transformation suffices to learn good representations, even though the model can almost perfectly identify the positive pairs in the contrastive task. When composing augmentations, the contrastive prediction task becomes harder, but the quality of representation improves dramatically. Compared to handcrafted pretext tasks-based methods, contrastive SSL methods are easier to implement but the pretraining process is computationally expensive as they require large batches and large amount of unlabelled datasets. Nevertheless, the pretrained backbone ConvNets through contrastive SSL methods generally produce better latent representations and perform well on downstream tasks.

Importantly, even though classical image segmentation methods can be regarded as one of the pretext tasks in SSL, the downstream segmentation and object detection tasks usually try to detect/segment certain salient objects and areas in an image, rather than entire textures, which unsupervised segmentation algorithms generate. While essentially solving a segmentation problem, salient object detection and

segmentation approaches segment only the salient foreground object from the background, rather than partitioning an image into regions of coherent properties as in general segmentation algorithms. That is, using a salient image segmentation in SSL is likely to generate better representations when the downstream task is defined as either image segmentation or object detection.

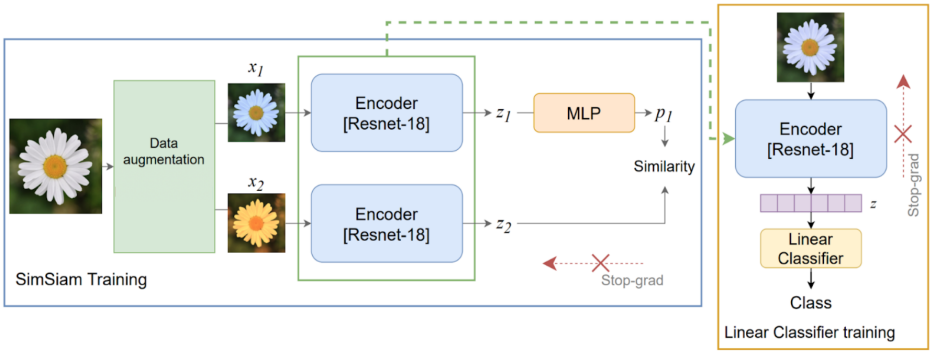

Figure 5.12: Linear evaluation protocol to evaluate the learned representations of a pretraining process of SimSiam [199].

5.3.2.1 Global Contrast based Salient Region Detection (SGD)

SGD, also known as SaliencyCut [100], is an automated unsupervised salient region extraction method, an improved iterative version of GrabCut [223], which introduced a contrast analysis method to integrate spatial relationships into region-level contrast computation. In GrabCut, the user initially draws a rectangle around the foreground region in an image, and then the algorithm iteratively segments it to get the best result. The executed steps are (i) estimating the color distribution of the foreground and background via a Gaussian Mixture Model (GMM), (ii) constructing a Markov random field over the pixels labels (i.e., foreground vs. background), and (iii) applying a graph cut optimization to arrive at the final segmentation. Instead of manually selecting this rectangular region to initialize the process, SaliencyCut is using histogram-based contrast (HC) and region-based contrast (RC) techniques (that are also suggested in the same paper) to create saliency maps and then binarizes this map using a fixed threshold (e.g., value of 70).

In RC, the input image is first segmented into regions, then the color contrast at the region level is computed, and saliency for each region is finally defined as the weighted sum of the region's contrasts to all other regions in the image. The weights are set according to the spatial distances with farther regions being assigned smaller weights.

In HC, the number of colors needed to consider is reduced to 1728 by quantizing each color channel to have 12 different values (12 color values per channel in R-G-B). Considering that color in a natural image typically covers only a small portion of the full color space, the number of colors is further reduced by ignoring less frequently occurring colors. By choosing more frequently occurring colors, and by ensuring that they cover the colors used *de facto* by more than 95% of the image pixels, we are typically left with around n = 85 colors. The colors of the remaining pixels, which comprise fewer than 5% of the image pixels, are replaced by the closest colors in the histogram.

Once the saliency map is initialized with RC and HC, GrabCut is iteratively run (i.e., iterative refinements) to improve the SaliencyCut result. After each iteration, dilation and erosion operations are used on the current segmentation result to get a new trimap for the next GrabCut iteration. During this iterative refinement, adaptive fitting is used (regions closer to an initial salient object region are more likely to be part of that salient object than far-away regions). Thus, the new initialization enables GrabCut to include nearby salient regions, and exclude non-salient regions according to color dissimilarity. See [100] for more details regarding SGD and Figure 5.13.

Finally, Figure 5.14 presents the outcomes' comparison of SGD-based segmentation versus latest unsupervised segmentation on PASCAL VOC 2012 [225].

5.3.3 Related Work

Semantic segmentation is the task of assigning each pixel to a specific class label. The class labels can be the same as for object detection, but unlike the object detection task, which labels each instance of an object as separate objects, semantic segmentation only assigns a pixel a specific class label and does not differentiate instances of objects.

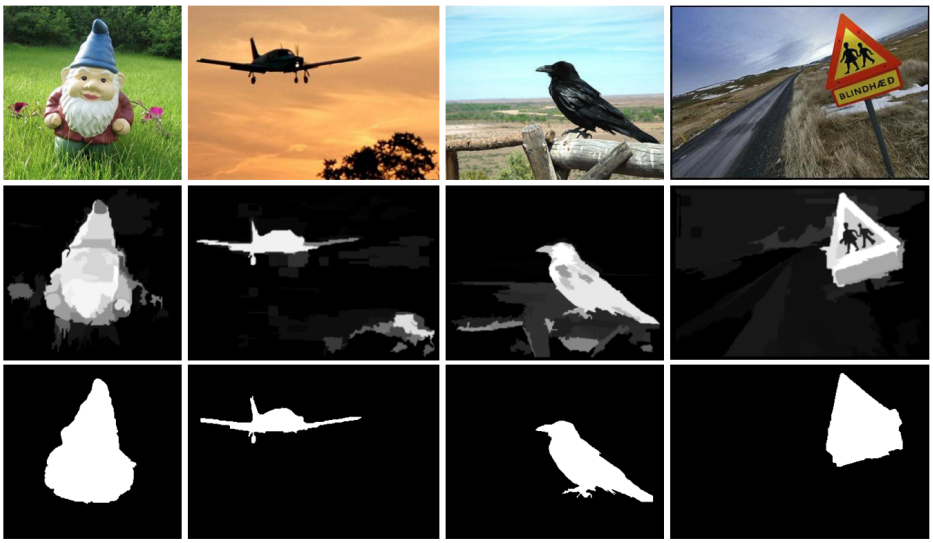

Figure 5.13: Given an input image (top), a global contrast analysis is used to compute a saliency map (middle), which can be used to produce an unsupervised segmentation mask (bottom) for an object of interest [224].

As the augmentations in any SSL method should be tailored *a priori* and fit in terms of the downstream tasks, devising an augmentation policy for downstream segmentation tasks depends on harnessing the intrinsic features that help model learn how to segment the objects in an image. This is usually achieved through auxilary tasks such as rotating, cropping, colorization etc. One of the most useful auxilary tasks can be regarded as image colorization that is introduced in [214] as a process of estimating RGB colors for grayscale images. The backbone network within the pretrained model performed well for downstream tasks like object classification, detection, and segmentation compared to other methods. The study in [229] also suggested a similar technique to automatically colorize grayscale images by merging local information dependent on small image patches with global priors computed using the entire image. Since these two techniques employ a strategy of finding suitable reference images and transferring their color onto a target grayscale image, the semantic information plays a little role and has the potential to actually hamper the SSL. Probably one of the most useful colorization tasks is suggested by [230], in which a system must interpret the

semantic composition of the scene (what is in the image) as well as to localize objects (where things are) to incorporate semantic parsing and localization into a colorization system. In a generative manner, [213] proposed image inpainting, a context-based pixel prediction where the network understands the context of the entire image as well as the hypothesis for missing parts, hence assisting in extracting the semantic information.

There are many other auxiliary tasks proposed in SSL but most of these methods focus on basic inherent visual features, like image patches and rotation, which are very simple tasks that are not likely to completely learn the semantics and the spatial features of the image. Actually, the default augmentation policies in contrastive SSL methods are the derivation of these auxiliary tasks. To the best of our knowledge, salient image segmentation has not been used as an auxiliary task in any SSL method before, let alone contrastive SSL.

5.3.4 Implementation, Setup & Results

Given the aforementioned research question, we derive concrete experimentation tasks:

1. Preliminary: Applying SGD to the PVs' datasets
2. Obtaining a solid and an efficient implementation
3. Testing the primary hypothesis: SGD as an augmentation policy

5.3.4.1 Setup and Datasets

We used the following datasets in the current study:

- **Aerial Drone Images for Solar Photovoltaic (PV) Panels Defects (NORCE-PV):** This dataset is provided by NORCE (Norwegian Research Centre) and has 790 high resolution aerial drone images. It is originally gathered for PV panels defect detection classification and annotated by NORCE internally. In this study, we will not be dealing with defect classes but the images themselves to run the experiments. The dataset is not publicly available.

- **Multi-resolution PV Dataset From Satellite and Aerial Imagery (MultiRes-PV):** This dataset includes three groups of PV samples collected at the spatial resolution of 0.8m, 0.3m and 0.1m, namely PV08 from Gaofen-2 and Beijing-2 imagery, PV03 from aerial photography, and PV01 from UAV orthophotos. In this study, we only used PV01 rooftop images (645 in total) that comes with segmentation masks that come within 256x256 size [231].

- **CIFAR10:** The CIFAR-10 dataset consists of 60,000 32x32 colour images in 10 classes, with 6000 images per class. There are 50,000 training images and 10,000 test images [232].

- **STL10:** The STL-10 dataset is a subset of ImageNet, consists of 1300 labelled, 100,000 96x96 unlabelled colour images in 10 classes. In particular, each class has fewer labeled training examples than in CIFAR-10, but a very large set of unlabeled examples is provided to learn image models prior to supervised training [233].

5.3.4.2 Preliminary: Running SGD on NORCE-PV and MultiRes-PV Datasets

SGD is a computationally expensive process and it produces segmentations on various level of details given the size of an image. For instance, it takes $\sim$ 2min to generate a saliency map (segmentation) of an 600x450 RGB image, while it takes around $\sim$ 1sec to accomplish the same process on 60x45 (10% of the original image). In the cases wherein a highly detailed segmentation map is needed, SGD implementation would be practically impossible to implement on real time. In some other cases wherein a rough segmentation map would suffice, applying SGD on a reduced size of an image would still produce useful segmentation. On the other hand, when we apply SGD on low resolution images like MultiRes-PV images (256x256), we may need to upscale the image to get a better segmentation. The outcome of applying SGD on various sizes of NORCE-PV (high resolution), NORCE-PV (high resolution, zoomed-in) and MultiRes-PV (low resolution) images is presented in Figures 5.16, Figures 5.17 and 5.18, respectively.

We also run STEGO [234] and DETIC [102], two recent popular zero-shot DL-based image segmentation algorithms, on our datasets and compare results with SGD. As

evident in Figure 5.15, DETIC obtains impressive results as it is already capable of detecting solar arrays from natural images (it is trained to detect 20 thousand different objects out of the box). Compared to the heavy STEGO algorithm, developed just a few months ago, SGD performs comparatively better.

5.3.4.3 An Efficient Implementation

Employing SGD on any DL training routine as an augmentation or transformation policy is highly inefficient in terms of computational costs. Notably, using GPU is not an option due to a layered (a mixture of OpenCV and Python functions) image preprocessing techniques along with RC and HC methods. The fact that thousands of images are to pass through this process in each epoch renders the integration of SGD in the DL training process practically infeasible. Our tests indicate that training a Resnet-18 ConvNet with SGD augmentation to classify PV defects using the NORCE-PV dataset (790 images, shape 32x32), for one epoch, takes $\sim$ 50 min on Tesla K80 (11.5GB) GPU.

Image to Segmentation Map Translation via Pix2Pix To achieve feasiblity, we firstly thought about training a generative DL model to learn SGD segmentations via image translation and then replace the SGD process with this generative model. We chose the Pix2Pix architecture [235] for this purpose and trained several models. We basically fed the original images and SGD-segmented version to the network and trained further to get the model learn translating one image to another. Using this Pix2Pix model within the Resnet-18 training routine to replace SGD reduced the duration from $\sim$ 50 min to $\sim$ 30 min per epoch. Even though the results look promising (Figure 5.19), there are several drawbacks, e.g., information loss during the translation process, being outperformed on image instances that were under-represented, and the limited speed-improvement gains – which altogether render this process unviable in practice.

Offline Augmentation with Hashing SGD is a computationally expensive method that results in a speed bottleneck when used in DL training routines and it might be one of the reasons that such a strong segmentation method could not establish itself despite all the latest advancements in similar fields. Since SGD is basically a deterministic approach, the segmentation map is always the same except colors; hence the contours and overall shapes are always identical for an

image. In other words, even if the pixel colors change at each iteration of SGD, the overall shapes and lines in an image are always the same, unless resizing is applied. Experimenting around this attribute, we devised an unprecedented solution by running SGD once for all the images in the dataset and creating a segmentation mask for each image and then reusing that every epoch during training without running SGD again and again. In order to simulate the random colors and make the model invariant to colors (so it can focus more on other features, e.g., contrast, shapes, lines etc.), we also applied random color jittering and swapping from the same color palette utilized by SGD. Next, we describe the explicit steps:

(Step-1) Read every image in the dataset as a `numpy` array and hash it to store the entire image as a single hash code (string),

(Step-2) Run SGD over every image in the dataset and save the segmentation map as an image to the disk,

(Step-3) Create a dictionary (key-value pairs) with the hash strings (of Step-1) as keys and the file path of every segmentation map as the associated value (if reading from disk becomes a bottleneck, all of the segmentation maps can be read at once and stored in the memory as an array),

(Step-4) In model training with original images, get the hashmap of every image array and find the file path of corresponding segmentation map from the dictionary above at each iteration, and read that image from the disk or from memory if it is stored in memory,

(Step-5) Apply color jittering to randomly swap the colors of the augmented image and use that as a proxy augmented image during the rest of the process.

This technique allowed SGD process to run instantly (practically below 1sec) as the segmentation maps are read from disk/memory at real-time at no cost.

5.3.4.4 Using SGD as an Augmentation Policy in Contrastive SSL Algorithms

In the Contrastive SSL pre-training routine, the augmented version of the original sample is considered as a positive sample, and the rest of the samples in the

batch/dataset (depends on the method being used) are considered negative samples. Then, the model is trained in a way that it learns to differentiate positive samples from the negative ones. In doing so, the model learns quality representations of the samples and is used later for transferring knowledge to downstream tasks.

A predefined CV architecture (e.g., ResNet50, VGG19, EfficientNet etc.) is typically used as a ConvNet backbone, and the weights (model parameters) are updated during this contrastive SSL process. Then, the backbone is saved and used in another architecture as a starting point or just as a facilitator of feature extraction (representations from one of the last fully connected (FC) layers).

In this study, in order to quickly iterate across various heavy augmentation combinations under limited computational resources, we used a lightweight ResNet18 architecture and then pretrained several backbones with the combination of SGD versus standard image augmentation techniques (crop, grayscaling, jittering etc.) using the following SSL algorithms: SimSiam [199], BYOL [196], SimCLR [195], MoCo [198], SwAV [200] and Barlow Twins [201]. Then we tested the effectiveness of these backbones in downstream image clustering, image segmentation, object detection and classification problems.

Image Clustering Using SSL Backbones Using the 80% of the entire NORCE-PV dataset with no label, we trained SvAW, SimSiam and SimCLR models for 100 epochs with the combinations of default SSL augmentations and SGD (with offline augmentation through hashing as well as Pix2Pix versions), used their backbone networks to extract the image representations (vectors) of 20% of the dataset, and then used KMeans and t-SNE methods to cluster and visualize the clusters. The outcome is presented in Figure 5.20. Evidently, SGD, when pretrained on unlabelled images, enables better image representations (vectors, embeddings) in all of the tested cases. The more distant the clusters, the better the representations generated by the SSL backbone network. In fact, observing this clear delineation between the clusters on a 2D space had originally given us an idea of applying SGD as an augmentation on larger scales with other combinations on a downstream image segmentation task.

Image Segmentation and Object Detection Using SSL Backbones As a downstream image segmentation architecture, we used Detectron2 [236] framework

on MultiRes-PV dataset to detect and segment solar panels from rooftop aerial images. In order to pretrain the SSL algorithms, the following augmentation policies are applied (p within the parenthesis corresponds to the probability of applying the respective technique, and sample augmentations are shown in Figure 5.21). The clear difference between random augmentation versus SGD augmentation can also be seen in Figure 5.22).

- Default SSL augmentations (*Random Resized Crop, Random Rotate, Random Horizontal Flip, Random Vertical Flip, Random Color Jittering, Random Grayscale, Gaussian Blur at various probabilities, p*)
- SGD ($p = 1.0$),
- SGD ($p = 1.0$) with default SSL augmentations
- SGD ($p = 0.5$) with default SSL augmentations
- SGD ($p = 1.0$) with jittering ($p = 1.0$)
- SGD ($p = 0.5$) with jittering ($p = 1.0$)
- SGD ($p = 0.8$) with jittering ($p = 0.8$)

The batch size is known to be one of the most important parameters in SSL and the larger the batch size, the better the representations as most of SSL algorithms greatly benefit from large batches (in order to have larger number of negatives available in mini batches) but we could not investigate performance for more than 128 images in a batch due to limited computational resources and the high number of experiments that we need to run (more than 500 runs, each takes at least a few hours). In order to test the impact of image resizing and batch sizing on SGD in image segmentation tasks, we picked three parameter combinations with five SSL methods (SimSiam, SimCLR, BYOL, MoCo, Barlow Twins): (i) Offline SGD applied on the original size ($100p$) with batch size 16 ($bs16$), (ii) Offline SGD applied on the original size ($100p$) with batch size 128, (iii) Offline SGD applied on the upscaled size ($200p$) with batch size 16.

Upscaling (x2) is tested to see the impact of detailed SGD segmentations, and it is found that the performance reaches its peak when SGD is applied on the original image-size with a batch size of 128. We also tested with doubling the resolution of each image and then applying SGD but the impact of SGD on SSL performance was not as good as in the original image-size. This can be explained by the fact

that the color variation in SGD with low resolution (original size) images is more or less similar to what we can expect to see in a solar panel segmentation. Once we increase the resolution by up-scaling, the details on a solar panel become more evident and the SGD may assign different colors for such areas, which we do not want to see for a simple panel segmentation task (e.g., every object is represented by a single color in image segmentation).

Then, using 500 images as a training set (out of 645 images from MultiRes-PV dataset), we pretrained all of the five SSL methods in a training loop with image segmentation and object detection tasks and measured the test accuracy on the test set (145 images). We observed that performances of SSL methods vary given the augmentation policy, as shown in Figure 5.23. For instance, while MoCo performs better compared to other SSL methods, with default SSL augmentations yet no SGD, SimSiam and BYOL perform better compared to other SSL methods with SGD plus default SSL augmentations as well as with only default augmentations. In another case, SimCLR performs the best when SGD is applied at $p = 0.5$ along with default augmentations. This result tells us that each SSL method performs differently, as a function of the augmentation policy, while the impact of SGD also varies under different settings.

Then, we experimented with the same SSL methods, having additionally no-SSL (with default ImageNet checkpoints) at various levels of labeled images (subsets from 10% to 100%) and measured the test accuracy on the same 145 images. For instance, we picked at first 50 labeled images from the training set, then using a pretrained SSL backbone from the previous step, we trained an image segmentation and object detection model with Detectron2, and tested the models on the test set. Then we did the same for 20%, 30% and so on. The AP metrics for the selected SSL methods for selected augmentation are shown in Figure 5.24. As seen from this figure, in every level of the reduced training set, an augmentation policy with an SGD component usually performs better than an SSL with default augmentations as well as a vanilla (no-SSL) model training. In some cases, using only SGD augmentation would even suffice without further application of augmentation.

Image Classification Using SSL Backbones As a downstream image classification model, using CIFAR10 and STL10 datasets, we used Logistic Regression

to classify the image representations taken from the penultimate layers of SimCLR backbones that are pretrained on CIFAR10 training set (5000 unlabelled images) with the augmentation policies mentioned before (with hashing applied to SGD augmentations). During these experiments, various levels of labeled images from CIFAR10 (from 10% to 100%) are picked at each iteration and only one SSL method (SimCLR) is picked for the sake of brevity. The results from this experiment can be seen in Figure 5.25). Even though there is a slim performance gain (around 5% improvement) with SGD plus default augmentations applied, the default pretrained ImageNet backbone still performs better without SSL. This can be explained by the smaller number of utilized epochs (100) while pretraining with SimCLR and the large number of labelled images being used in the ImageNet pretraining. Considering that SimCLR is pretrained only by 5000 unlabelled images from CIFAR10 while the default ResNet18 model is pretrained with the entire ImageNet dataset with ground truth labels, getting similar metrics with SimCLR is still worth mentioning herein.

We run a similar experiment on the STL10 dataset (using 100 thousand unlabeled images), but SGD did not contribute in that case and performed worse. Since STL is already a subset of the ImageNet dataset, the default pretrained ImageNet backbone did quite well and exceed all the metrics achieved with SSL methods. The results are shown in Figure 5.26. Given the fact that the images in the STL10 dataset have higher resolution (96x96) than the images in CIFAR10 (32x32), getting worse performance with SSL using SGD can be explained by the same phenomenon we observed in SGD of upscaled images doing worse compared to original size images.

5.3.5 Discussion

Our experiments with clustering the image representations via SGD-augmented backbone networks indicate that SGD enables obtaining better image representations in all of the cases that we tested. Even if we may not know the number of clusters within a dataset, the clear delineation between the clusters indicates a better representation to separate one image from another using visual clues. Usually, the most salient details and objects play the role of separators between one image from another, but using classical image augmentation policies (e.g., cropping,

jittering, etc.) totally fails on this end. At the same time, SGD-based salient object segmentation does a better job on catching those critical details, thus assisting more in downstream clustering tasks.

We argue that using a salient image segmentation in SSL generates better representations when the downstream task is image segmentation or object detection due to the fact that salient object detection and segmentation approaches segment only the salient foreground object from the background, rather than partition an image into regions of coherent properties as in general segmentation algorithms. Our experiments with various augmentation combinations with and without SGD show that the augmentation policy having an SGD component usually performs better than an SSL with default augmentations as well as a vanilla (no SSL) model training. In some cases, using only SGD augmentation would even suffice, with no further application of augmentation.

Another important observation is that each SSL method performs differently as a function of the underlying augmentation policy, whereas the impact of SGD also varies under different settings. For instance, while MoCo performs better compared to other SSL methods, with default SSL augmentations yet no SGD, SimSiam and BYOL perform better compared to other SSL methods with SGD plus default SSL augmentations as well as with only default augmentations, as was elaborated in the previous section. Notably, this in fact the common question in the age of DL in which there are a myriad of alternatives: How to know which algorithm does better on a certain use case and how to pick the right one? Not only the SSL method but also the augmentation technique, downstream task and even the resolution of your images matter.

One of the most unexpected observation of our tests can be regarded as having worse results with SGD applied on high resolution images. We have observed this effect on various occasions during our experiments as reported in the previous section. This can be explained by the fact that the color variation in SGD with low resolution (original size) images is lower and more similar to what we can expect to see in a coarse grained segmentation tasks. Once we increase the resolution by up-scaling, the details on an image becomes more evident and SGD may assign different colors for such areas, which we do not want to see for a simple image

segmentation task (e.g., every object is represented by a single color in image segmentation). However, SSL performing well on low resolution images can also be attributed to the low number of epochs (i.e., 100) as the model is not able to catch the low level details in high resolution images in such a limited number of epochs.

Nevertheless, a recent SSL study also supports our claim and sheds some light on this issue as follows: [237] claims that current self-supervised methods learn representations that can easily disambiguate coarse-grained visual concepts like those in ImageNet. However, as the granularity of the concepts becomes finer, self-supervised performance lags further behind supervised baselines.

5.3.6 Conclusions

By empirically addressing the posed research questions, our investigation confirmed the capacity of SGD to play a role in modern SSL. Overall, this study successfully leveraged SGD, a 10-years-old salient object detection and segmentation algorithm, as a potent image augmentation technique for downstream image segmentation tasks. To achieve an effective integration of SGD into SSL pretraining routines, we devised a simple manipulation called offline augmentation with hashing. This fine implementation detail enabled us to run hundreds of SSL experiments with various parameters and configurations. We then demonstrated that using a salient image segmentation in SSL generates better representations when the downstream task is image segmentation. Our experiments with clustering the image representations via SGD-augmented backbone networks indicate that SGD helps better image representations in all of the cases tested. Our experiments with various augmentation policies including SGD show that the augmentation policy having a SGD component usually does better than a SSL with default augmentations; in some cases, using only SGD augmentation alone would even be be better.

Our experiments with MultiRes-PV, CIFAR10 and STL10 also showed that SSL with SGD-based augmentation policy performs well with low resolution images. This still remains to be verified whether fine-grained features and/or the low number of epochs played a role in this observation. We contend that salient object segmentation algorithms produce coarse grained segmentation due to saliency, thus perform well on segmenting coarse grained objects like PV solar panels and it may

not hold true if our goal were to extract fine grained details in an image.

We observed that each SSL method performs differently given the augmentation policy, whereas the impact of SGD also varies under different settings. In our opinion, the most unexpected observation of our tests can be regarded as having worse results with SGD applied to high resolution images compared to low resolution ones. We conclude that the augmentation technique, type of a downstream task and image resolution are the most important elements that have a high impact on the success of a SSL method picked.

Figure 5.14: Comparing the outcomes of unsupervised versus SGD-based segmentation on PASCAL VOC 2012. Invariant Information Clustering (IIC) [226], Superpixels [227], Continuity Loss [228]. Different segments are shown in different colors (the images in the first 5 rows are taken from [228]). Notably, SGD was developed at least 8 years prior to IIC and Superpixels, and can still perform comparatively better on some cases.

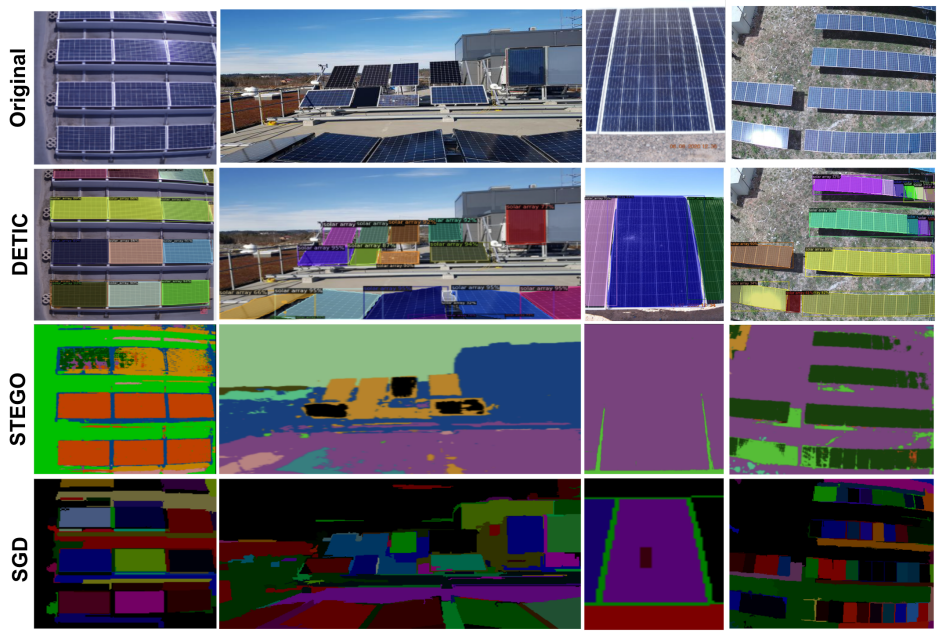

Figure 5.15: Comparison of transformer-based zero-shot segmentation methods versus SGD on NORCE PV dataset.

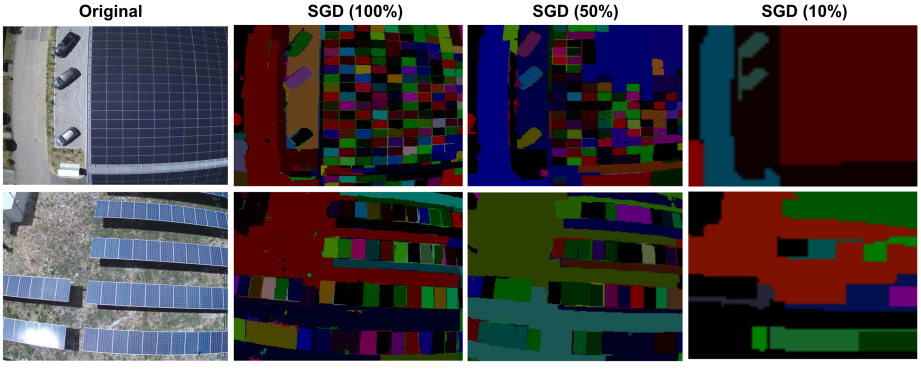

Figure 5.16: Segmentation produced by SGD from NORCE-PV images with various sizes.

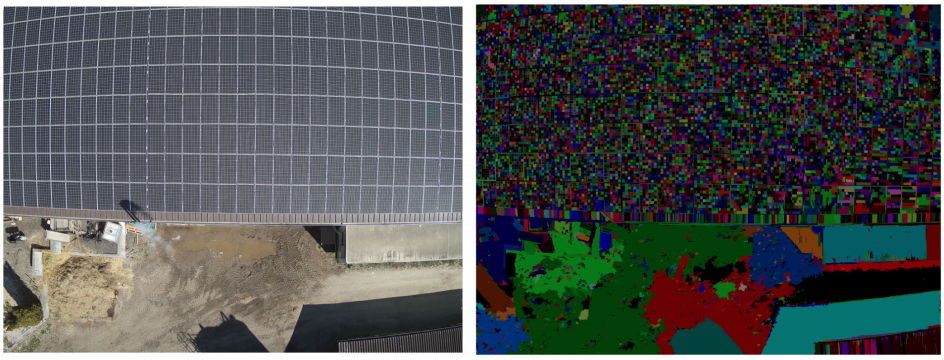

Figure 5.17: Segmentation produced by SGD from a NORCE-PV image with original resolution (dimension 4000x3000, resolution 72x72). The higher the resolution, the better the segmentation at fine grained detailed.

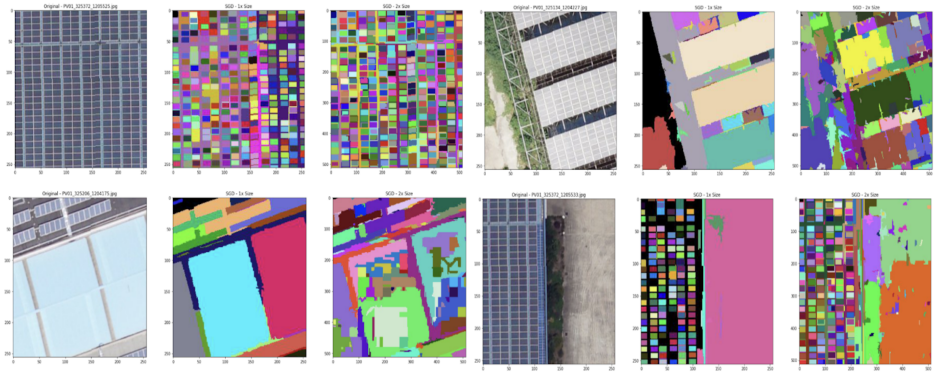

Figure 5.18: Segmentation produced by SGD from low resolution MultiRes-PV images with original versus 200% rescaled.

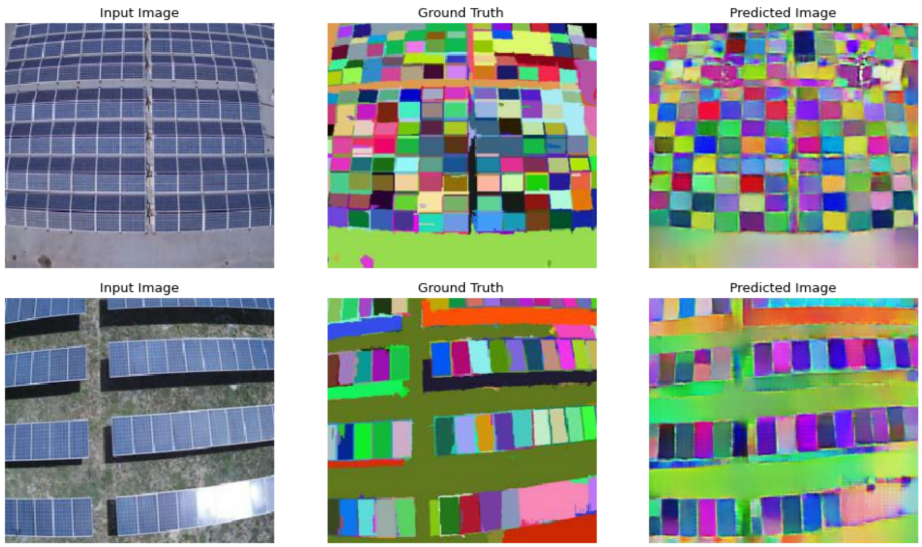

Figure 5.19: The second column of this figure shows the SGD segmentation of the original images and the images generated with Pix2Pix image translation model are shown at the third column.

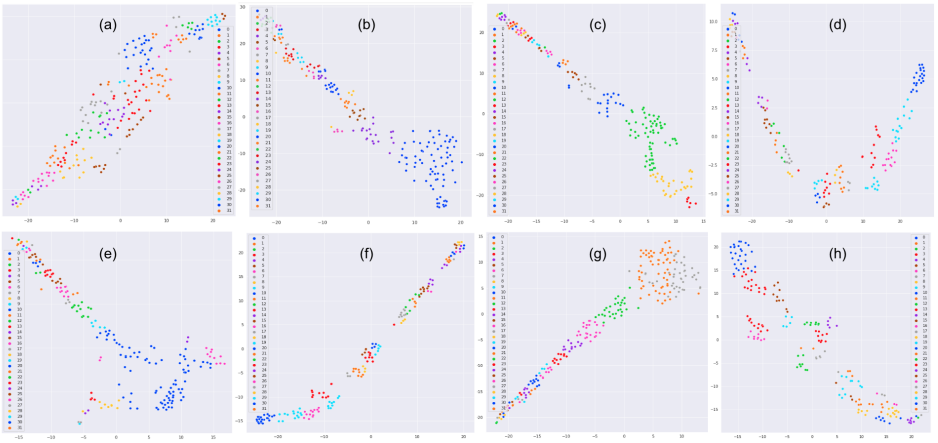

Figure 5.20: Clustering partial NORCE-PV dataset image representations produced by (a) ResNet18 ImageNet weights (b) SimSiam random aug. (c) SimSiam random + PixPix based SGD aug. (d) SimCLR random + SGD aug. (e) SimSiam random + SGD aug. f) SimSiam with PixPix based SGD aug. (g) SwAV with SGD aug. (h) SimSiam with SGD aug.

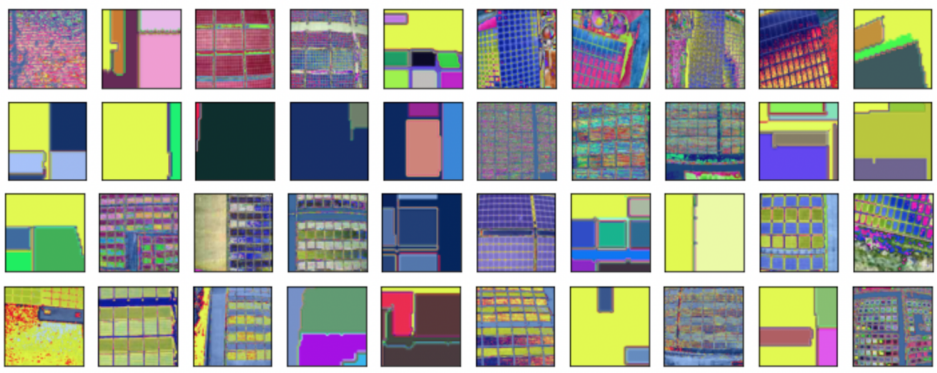

Figure 5.21: This figure shows a batch of 40 NORCE-PV images. Every image in this batch is segmented by SGD at first and then the other default random augmentations are applied. Basically, this is what the model sees at each iteration.

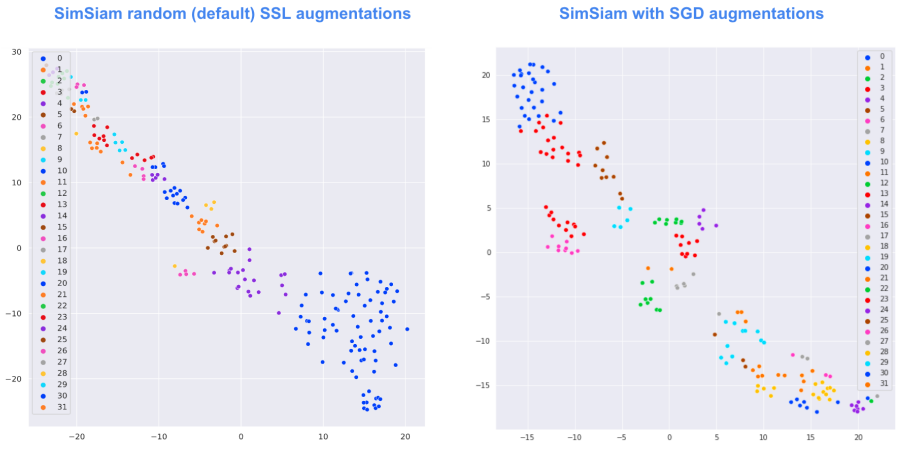

Figure 5.22: Clustering partial NORCE-PV dataset image representations produced by ResNet18 ImageNet weights finetuned with SimSiam using random augmentation versus SGD augmentation that returns much better delineation between the clusters representing different classes.

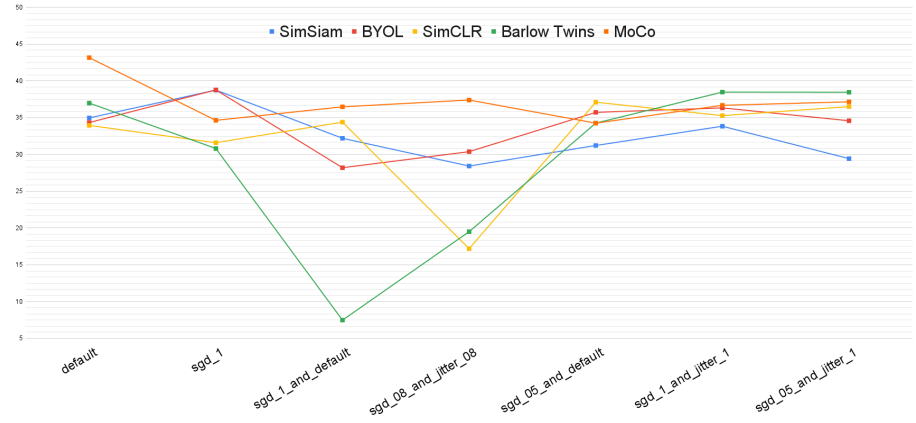

Figure 5.23: Selecting the best combination of SGD and default augmentation strategies for various SSL methods using original size images with batch size 128. The performance of augmentation strategies varies by the SSL methods accompanied. SGD alone without any other augmentation strategies produced the best result in BYOL and SimSiam compared to other methods and augmentation policies.

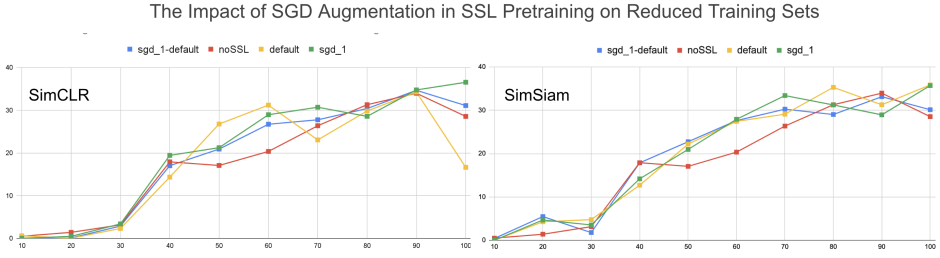

Figure 5.24: Segmentation metrics for SGD with SimSiam and SimCLR experiments with reduced number of images from the training set at various levels. x-axis denotes the percentage of the samples with respect to entire training set (from 10% to 100%) and y-axis denotes test set average precision (AP) of segmentation model trained with the partial training set.

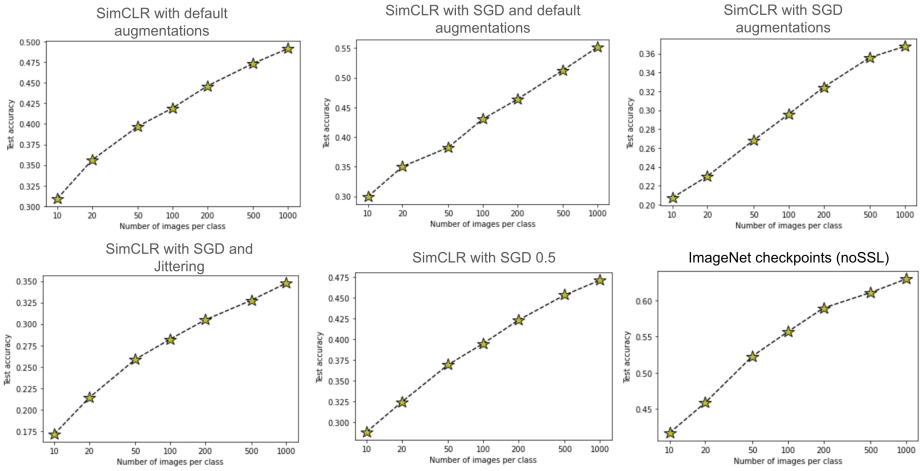

Figure 5.25: Classification of images in CIFAR10 dataset using partial samples from training set and SSL method with SGD applied. SGD plus default augmentations outperforms default SSL augmentations by 5% and a little worse than ImageNet checkpoints.

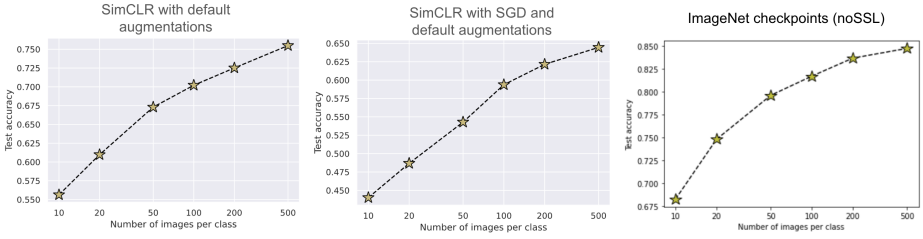

Figure 5.26: Classification of images in STL10 dataset using partial samples from training set and SSL method with SGD applied. SGD did worse than default SSL augmentations and the default pretrained ImageNet backbone outperforms all the metrics achieved with SSL methods (due to the fact that STL is already a subset of ImageNet dataset).