Dimensional Collapse in Video Representation Learning

Master's Thesis
to obtain the academic degree
Master of Science

submitted by
Paul Kapust

Computer Vision and Remote Sensing
Technische Universität Berlin

November 2023

1. Reviewer : Prof. Dr.-Ing. Olaf Hellwich
2. Reviewer : Prof. Dr. Marc Alexa

Abstract

Video representation learning aims to reduce videos into a more compact vector form of maximum information content to enable efficient and effective machine learning. Many representation learning approaches rely on a labelled dataset to learn meaningful representation vectors. Such, the potential of supervised representation learning is restricted by the size of the labelled dataset. Thus, the technique of self-supervised learning has risen in popularity, which does not require labelled data. Yet, a downside of self-supervised methods is the possibility of encountering complete collapse. An event in which all data is represented as a constant vector. Presently, self-supervised approaches are designed to deal with complete collapse. However, recently, a subsequent collapse pattern has been detected. Dimensional collapse limits the utilization of the representation space and restricts representation vectors to a lower-dimensional subspace. In this thesis, we will extend the research for dimensional collapse in image representation learning and start determining dimensional collapse for common video representation learning methods. To do so, we will create an extensive literature review of the most relevant representation learning methods and identify how effectively these different approaches use their representation space.

Kurzfassung

Das Lernen von Videorepräsentationen zielt darauf ab, Videos in eine kompaktere Vektorform mit maximalem Informationsgehalt zu reduzieren, um effizientes und effektives maschinelles Lernen zu ermöglichen. Viele Ansätze zum Repräsentationslernen benötigen einen annotierten Datensatz, um sinnvolle Repräsentationsvektoren zu lernen. So wird das Potenzial des überwachten Repräsentationslernens durch die Größe des annotierten Datensatzes eingeschränkt. Daher hat die Technik des selbstüberwachten Lernens an Popularität zugenommen, die keine gekennzeichneten Daten benötigt. Jedoch ist ein Nachteil der selbstüberwachten Methoden die Möglichkeit eines vollständigen Zusammenbruchs. Ein Ereignis, bei dem alle Daten als ein konstanter Vektor dargestellt werden. Gegenwärtig sind die selbstüberwachten Verfahren so entworfen, dass sie einen vollständigen Zusammenbruch verhindern können. Vor Kurzem wurde jedoch ein weiteres Muster des Zusammenbruchs entdeckt. Der dimensionale Zusammenbruch schränkt die Nutzung des Repräsentationsraums ein und beschränkt die Repräsentationsvektoren auf einen niedrigdimensionalen Unterraum. In dieser Arbeit werden wir die Studie zum dimensionalen Zusammenbruch beim Lernen von Bildrepräsentationen erweitern und damit beginnen, den dimensionalen Zusammenbruch für weitverbreitete Methoden zum Lernen von Videorepräsentationen zu bestimmen. Um dies zu verwirklichen, werden wir eine umfassende Literaturübersicht über die wichtigsten Methoden des Repräsentationslernens erstellen und ermitteln, wie effektiv diese verschiedenen Ansätze ihren Repräsentationsraum nutzen.

Contents

List of Figures

List of Tables

1 Introduction

As the popular saying goes: *A picture is worth a thousand words.* Just imagine how many words a video would be worth. For us humans, images and videos are quite naturally processed and understood. They reflect how we perceive life. Subconsciously, inferences are made and responses are prepared. Such to represent images or videos, which for us are so easily to appreciate in their high-dimensional form, for a computer, they have to be broken down into a more compact numeric form to facilitate effective visual processing. Conventionally, both, images and videos, are reduced to representation vectors. For representations, to aid computation and prediction, recognisable features of the data have to be separated from distracting noise. Representation learning guides the search of representation vectors with many meaningful and expressive features of the former high-dimensional data. As displayed in the middle of Figure 1.1, the goal is for the representation vectors to span and utilize the whole of the representation space. *The performance of machine learning methods is heavily dependent on the choice of data representation* [21].

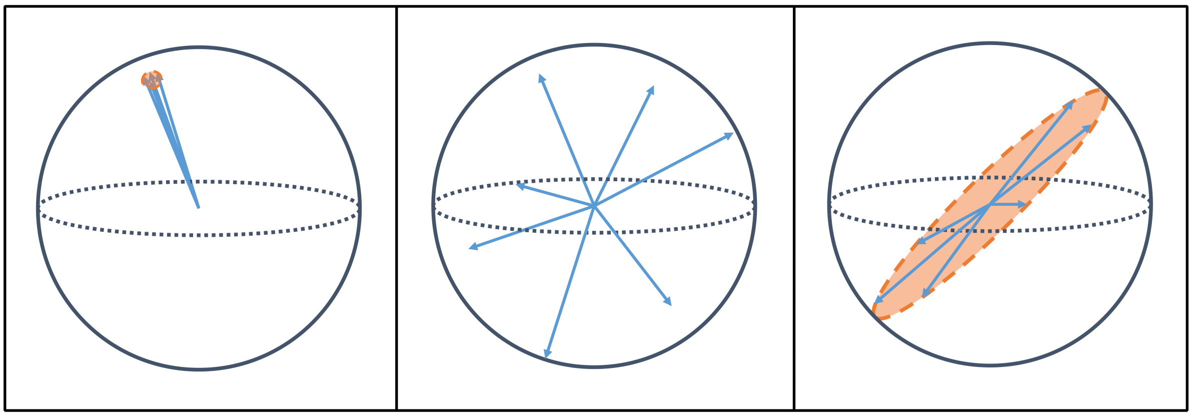


Figure 1.1: Left: Complete Collapse. **Middle**: No Collapse. **Right**: Dimensional Collapse. [1]

A common choice for representation learning is supervised training of Artificial Neural Networks. Supervised learning is enabled through the use of large manually-labelled datasets. The network breaks down the data into a lower-dimensional representation to learn its relevant components for predicting the provided target. For example, such learned representations of large informational depth, which have retained important visual characteristics, can be used to help further disentangle and compress images or videos of smaller datasets that previously had to rely on less context. However, a bottleneck exists. Major leaps in network model accuracy for videos have stemmed from utilizing larger and larger labelled datasets. Additional performance improvements would require expanding existing human-annotated datasets by several orders of magnitude [18]. The demand and cost becomes harder and harder to meet. Consequently, unsupervised representation learning methods, that do not rely on labelled data, come to be more relevant.

Self-supervised learning, a form of unsupervised learning, has recently risen in popularity. The general idea behind self-supervised approaches is to learn signals of supervision directly from the data itself instead of depending on or trusting human annotation. Thus, learning more pure visual representations. For instance, a common optimization procedure in self-supervised representation learning for images and videos is to build representations that are invariant to random data augmentations. The intention is for differently augmented views of the same data sample to produce a similar representation vector. Therefore, the neural network encoder has to focus on the semantic content of the visual data. Although, these kind of optimization problems have a big hurdle to overcome which is the complete collapse of the representation space, as depicted on the left side of Figure 1.1. Such problems have an optimal solution that cheats the original intention of learning meaningful representations. By converting all high-dimensional data into a constant representation, nothing is learned but maximization of similarity is secured.

Through adaption of their approaches, many representation learning methods have found ways to prevent complete collapse. However, complete collapse is not the only collapse pattern. Hua et. al (2021) [19] encountered the possibility of a dimensional collapse pattern, where the representation space falls into a lower subspace, as shown on the right side of Figure 1.1. Thus, the assumption is that this phenomenon of dimensional collapse limits the full potential of its representation learning approach by preventing the utilization of the whole representation space. Testing for dimensional collapse is not yet standard procedure and research into image representation learning methods regarding dimensional collapse has only just started. Therefore, we will extend the examination of image representation learning approaches for dimensional collapse and begin to determine dimensional collapse in video representation learning approaches. Dimensional collapse gives reason to inspect the representation space to determine how effectively many approaches use their representation space available.

In this thesis, we will first identify the most commonly used approaches in image and video representation learning, present a comprehensive literature review, discuss how to evaluate dimensional collapse, test these approaches for dimensional collapse, and finally, put the results into perspective.

2 Literature Review

This literature review will summarize the representation learning approaches which will later be used to determine dimensional collapse. More so, throughout the main sections, tables will be presented that list the specific pretrained models which will be analyzed. In general, the literature review is divided twofold. First, we will go over supervised approaches to learn representations, and secondly, we will go over self-supervised approaches to learn representations. A layer underneath that, image and video methods will be differentiated.

2.1 Supervised Image Representation Learning

Now, let us start with supervised approaches. For supervised training, the setup is more simplistic because of the convenience of labelled data. The main focus will be put on the neural network architectures that encode the images or videos from datasets into representation vectors.

2.1.1 AlexNet

In the year 2012, Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton of the University of Toronto entered the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [22] and blew their competition out of the water. They presented a deep Convolutional Neural Network architecture (CNN), one of the largest at that time, to be able to distinguish successfully between a total of 1000 different image classes. While the whole ImageNet dataset contains more than 20,000 labelled classes, the challenge made use of a popular subset containing more than 1.4 million images [22]. Previous in demand datasets like the CIFAR-100 dataset only contained up to 100 labelled classes with a total of 60,000 images [23]. Therefore, the challenge was quite complex and to make training of such a deep CNN feasible, the authors utilized an efficient GPU implementation of the convolution operation to speed up training. The results spoke for themselves. The challenge was won with a winning top-5 test error rate of 15.3% which was an over 10% improvement compared to the runner-up [2].

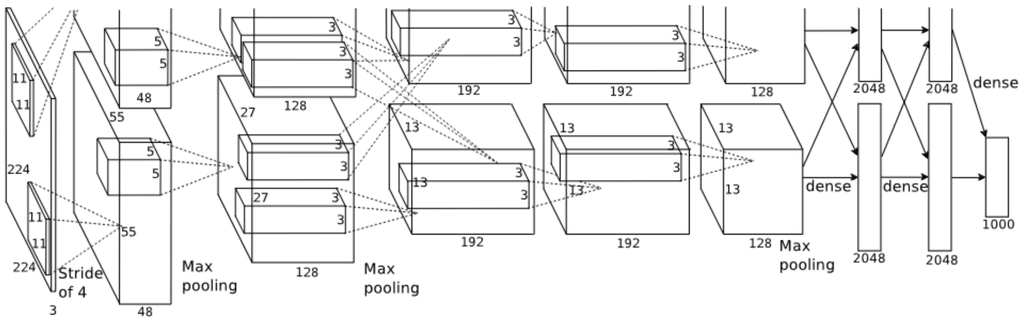


Figure 2.1: Illustration of the AlexNet architecture [2]

The winning neural network was dubbed AlexNet and is displayed in Figure 2.1. The architecture is in total comprised of five convolutional layers [2]. In each layer, multiple filters are trained/learned to extract important features from the high-dimensional input image. Such features are then important to construct low-dimensional and semantically meaningful representations of the data in the following hidden layers of the neural network. Non-linearities are introduced between the layers and help the representations to learn complex structures in the data. Furthermore, pooling layers are intersected between the convolutional layers to further reduce the dimensionality of the feature representations as the image traverses the neural encoder. After having passed all the convolutional layers, three fully connected linear layer remain [2]. The last fully connected layer contains as many output neurons as are required for the specific classification task. It uses the feature representation of the penultimate layer and learns to translate features to class memberships.

Figure 2.2 displays a selection of fully trained filters of the first convolutional layer of AlexNet. Here, many of these first level filters bear some resemblance with frequency and orientation filters [2]. The image is convolved with the early filters and such broken down in more fundamental low-level parts. One of the key novelties of the neural network framework is that these type of filter weights do not have to be handcrafted anymore but are learned automatically during training. Orientation filters could be detected, when visualized, but additionally, CNNs are also able to learn filters for which we might not even have a handcrafted correspondent. This does have a substantial effect on the resulting quality and more so potential of representations learned.

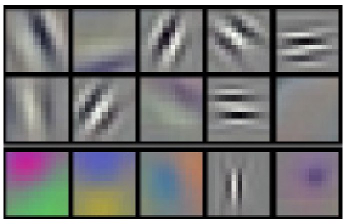


Figure 2.2: Filters of first convolutional layer in AlexNet [2]

Like in a standard Artificial Neural Network, a CNN learns task appropriate filter weights by applying the gradients of its weights iteratively. The gradients of a loss function are being computed using the famed backpropagation algorithm [24] which goes backwards through the network to determine the impact each weight has on the output of the loss function. A loss function measures how well the currently trained network performs a given task like the task of classifying an object in an image. Moreover, it is dependent on the weights and network structure. In the standard supervised case, a dataset is labelled. It contains the data and the associated labels. In that case, the loss function would be a measurement on how well the network is able to correctly predict the corresponding class. Concluding, the gradient $\frac{\partial L}{\partial w}$ computed from the loss function L related to a certain filter weight w tells us how changing this weight in any direction will impact the loss function and therefore, the performance of the network.

In supervised learning, a common metric to evaluate representations and its encoder is test accuracy. How many classifications does the neural network predict correctly based on data from the same distribution which it has not seen before. Inspecting accuracy offers an important but limited view into the quality of representations. Especially, if the goal is to later transfer the trained base encoder to other tasks of related distributions. The semantic quality of learned representations is not always straight forward to quantify. In [2], further approaches have been made to determine the semantic quality of representations. The authors compared the Euclidean distance between the 4096-dimensional representation vectors of the last hidden layer to determine which images the network believes to be closely related. This also tests the visual strength of the network. In its first column, Figure 2.3 shows four images from the test set of ImageNet.

To the right of it, images from the training set are displayed who share most similarity with each test image based on the Euclidean distance of their encoded representations.

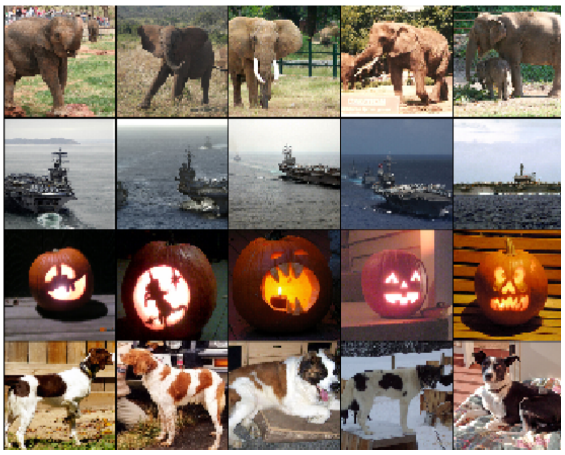


Figure 2.3: AlexNet representations with closest Euclidean similarity [2]

First, it can be observed that images of closer measured similarity share the same object, for example an elephant or a dog. Secondly, what is more important, similar images are being recognized regardless of the positioning or noise around the object. A direct pixel-wise Euclidean distance of these RGB images might not even deem them similar considering all other images in the dataset. Furthermore, it would also be computationally more expensive. In view of their results, AlexNet retains some deeper visual knowledge of the data which is present in the produced representations [2].

AlexNet, an architecture from the year 2012, is not the state-of-the-art anymore but it started a series of breakthroughs for image classification [3] and inspired a lot of research in the field of deep convolutional neural networks. Neural encoders like AlexNet can be trained with supervision using a large labelled dataset, but as an encoder, AlexNet can also be used in a self-supervised setting. In such setting, AlexNet still finds use every so often as a baseline to compare to other more timely neural networks. In a similar vain, we will compare, in a later chapter, dimensional collapse in a supervised trained AlexNet with more recent encoders for image representation learning.

2.1.2 ResNet

In Table 2.1, AlexNet is being compared to a more contemporary convolutional architecture, the Residual Neural Network, short ResNet [3]. While identifying most commonly used approaches in representation learning for images, be it supervised or self-supervised, the ResNet architecture has been by far the most regularly used base encoder to learn representations. It is ubiquitous and a central part in almost all self-supervised approaches. This will become even more evident in the following sections regarding self-supervised approaches in image representation learning. The general idea behind residual networks will also play an important role in video representation learning. Here as a baseline, the ResNet-50 architecture is in particularly a very common and popular one. The number 50 refers hereby to the layer depth of the architecture. As many other papers have done, ResNet builds on the initial ideas presented by AlexNet. It even rivals AlexNet in terms of demand. According to Google Scholar, the ResNet paper Deep residual learning for image recognition [3] has been cited over 184,300 times as of October 2023. This

Network Architecture	Top-1 Accuracy	Top-5 Accuracy	Release	Origin
AlexNet	56.52%	79.06%	2012	[25, 26]
ResNet-50	76.13%	92.86%	2015	[3, 27]

Table 2.1: Comparison of deep Convolutional Neural Networks on ImageNet-1k validation set

makes it one of the most cited papers in the field of machine learning [28].

Increasing the depth (i.e. number of convolutional layers) of a neural network is beneficial for the performance of a model [29]. After AlexNet won the ILSVRC classification challenge using a deep convolutional network, the obvious route for others to take was to go even deeper and to add more layers to their network. As simple as it sounds, evidence showed that network and representation depth is of central importance. The subsequent victors of the ILSVRC challenge all exploited even deeper models. Furthermore, it has also been shown that extremely deep representations have excellent generalization performance on other task involving smaller datasets [3,29].

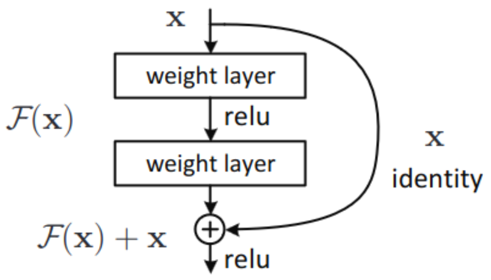

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.4: Residual learning building block [3]

Is learning better networks as easy as stacking more layers [3] and how are such deep models still trainable? Those are some of the central questions, the authors of the ResNet paper [3] tackled. Needless to say, they did so with quite some success. In 2015, the ResNet architecture won the ILSVRC classification challenge. To be able to build even deeper models, a degradation problem needed to be addressed. By simply adding more and more layers, the networks hits eventually a road block. With increasing depth, the accuracy gets saturated and then degrades rapidly [3]. This also can not be attributed to overfitting since the training error and test error degrade equally. At first, it might not seem intuitive. Should a deeper model not be able to learn the same as a shallower model? One might assume that the deeper model could learn a copy of the shallower model and for the remaining layers just learn the identity function. In reality, experiments show that current solvers are unable to find such solutions (or are unable to do so in feasible time) [3]. To address this situation, residual connections, or shortcut connections, are introduced. Shortcut connections attempt to make the identity mapping the default function which will be deviated from during training. Figure 2.4 displays such a residual block including its shortcut connection. These residual blocks are the building blocks of a ResNet model. The shortcut connection carries the identity over, while weight layers are learning a difference to the identity. The authors hypothesize that it is easier to optimize the residual mapping. To the extreme, if an identity mapping were optimal, it would be easier to push the residual to zero than to fit an identity mapping using a stack of non-linear layers [3]. Additionally, shortcut connections do not come with extra trainable parameters which would otherwise increase the complexity further. Using these residual blocks, models were trainable with a number of up to 152 layers to solve the ImageNet classification task. Previously, models were already considered *very deep* with a convolutional layer size of 16 up to 30 layers [3]. Lastly, overfitting was not an obstacle, thanks to these residual blocks, even when optimizing a 152 layer deep ResNet. Overfitting only seemed to become noticeable again when the authors stretched the limits and trained a ResNet model on the smaller CIFAR-10 dataset [23] with an incredible amount of 1202 layers [3].

Lastly, as mentioned before, residual functions are not only helpful to train deep networks on image data but do also play a role for video network encoders to learn deep spatiotemporal features. Subsequently, since a sizeable part of this thesis will be about video representation learning and contrasting both mediums, image and video, ResNet models will play a central role.

2.2 Supervised Video Representation Learning

Video as a medium introduces an additional dimension besides the two spatial dimensions, height and width. Where as an image remains on its own, a video is a composite of multiple images. The temporal signal accounts for the motion in the video and its passing of time. It almost feels like a kind of magical component. The human visual system can process 10 to 12 images per second and perceive them individually, while at higher rates it will create a sensation of visual continuity [30]. Imagine a pedestrian standing at a train station testing out her new camera. She is trying out the new burst mode feature for the very first time. Burst mode takes many pictures in quick sequence. As a new train appears at the station, she points her camera at the arriving train and shoots many pictures in high frequency. Later, she looks at the pictures on her camera, which she took that day. She uses the scroll wheel to transition from picture to picture. As she scrolls very quickly through the pictures, she notices how the many pictures of the arriving train seem to melt together to form a cohesive video. Doing this action more slowly, would allow her to discern the still images individually and not perceive them as a video in motion. This swift change between time-related images elevates a collection of frames to a video.

As videos are an extension of images, what big of a role does the additional temporal dimension play in learning meaningful video representations? A central task in computer vision for images is classification. A similar task for videos would be video classification or action recognition. To determine an action in a video, a human user would often not even have to consider the whole video. The spatial dimensions of a single frame already provide a lot of context and allude to what actions might occur. Depending on the video, spatial and temporal dimensions might differ in importance regarding certain vision tasks. To recognise the action of playing guitar, the hand motion in the video might be not as relevant as detecting the guitar spatially. On the other hand, to recognise someone playing air guitar, it is highly important to analyse the temporal dimension since single frames might only show a person in the midst of a movement.

Figure 2.5 considers different convolutional operations on how to process videos. This includes consideration and disregard of the temporal dimension. First in (a), a 2D convolution is visualized on a single channel image, as touched upon earlier. A filter steps over local regions of the frame and convolves filter and frame to create a 2D output frame. This output frame describes the overlap between input frame and filter. In (b), a 2D convolution is described on multiple channels. Considering an input to a CNN, these could be the three red, green and blue channels of a typical RGB image. In this exemplary convolutional layer, the CNN would learn filter for each individual channel and aggregate the results. An image of size $3 \times H \times W$, where H and W stand for the image height and width, would be convolved with filters of dimensions $k \times k$ for every channel. A video with the additional temporal dimension would be of shape $3 \times L \times H \times W$, where L is the number of frames in the video clip. While 2D convolution does not match a straight volume like a video, a concession is made by neglecting the temporal signal to make it work. Restructuring the frames as channels and reformulating the video as $3L \times H \times W$ would allow for a 2D convolution operations to process the video but at the cost of collapsing the temporal signal. Filters work independently on every frame and aggregate all information into a single output feature map. The temporal context between frames would be lost.

To maintain temporal and spatial information, 3D convolution filters can be used to also convolve along the temporal dimension. In (c) of Figure 2.5, such a 3D convolution is visualized. The filter size is $d \times k \times k$, where d stands for the temporal depth of the filter. During convolution,

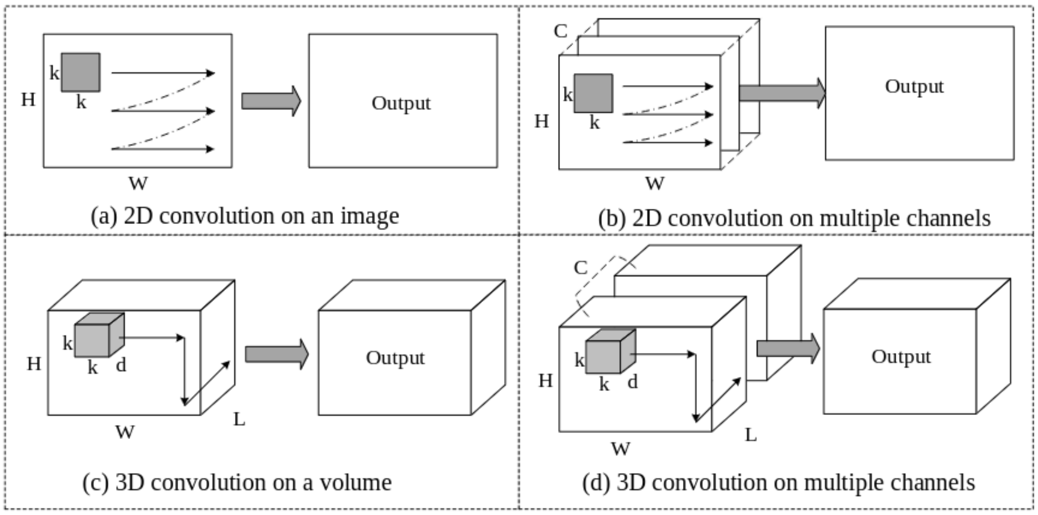

H
k
k
W
Output
(a) 2D convolution on an image
C
H
k
k
W
Output
(b) 2D convolution on multiple channels
H
k
k d
W
L
Output
(c) 3D convolution on a volume
C
H
k
k d
W
L
Output
(d) 3D convolution on multiple channels

Figure 2.5: Transition of 2D to 3D convolution operations [4]

the filter can be imagined as a box that moves through the volume, symbolizing the video, and locally computes its results. The feature map output would be a volume containing spatial and also temporal information. As in (a), (c) only considers one input channel and one output channel. Illustration (b) and (d) examine multiple input channels for 2D and 3D convolutions respectively. The use of multiple output channels is not depicted in Figure 2.5 but basically the number of output channels denotes how many different convolutional filters are applied on each frame or volume, and how many feature maps are being created as a result.

In 2018, a very deep image-based ResNet-152 model achieved close to state-of-the-art on the Sports-1M video dataset [31] by simply operating on the individual frames of the video [5]. Without considering the temporal component, this network was able to compete with shallower neural networks which employed 3D convolutions [32]. An image-based solution for video action recognition which performs close to state-of-the-art puts in question how advantageous 3D convolutions can be to produce meaningful video representations.

2.2.1 3D ResNet and ResNet(2+1)D

In [5], authors confront the importance of the temporal dimension by exploring the possible advantages of different 3D convolutions over image-based 2D convolutions. The advantage of performing convolutions over the whole spatiotemporal volume instead of only performing it over the spatial dimensions. Furthermore, a 3D ResNet architecture is presented through combining 3D convolutions and the residual learning paradigm, previously introduced with the ResNet architecture [3]. To establish a baseline, a typical 2D ResNet is used, named R2D. All the networks used in [5] are residual neural networks with 18 convolutional layers. R2D processes video clips like described in Figure 2.5 (b), whereas all frames are treated

Figure 2.6: Different convolution operations on the Kinetics-400 validation set [5]

Net	# params	Clip@1	Video@1	Clip@1	Video@1
Input		8×112×112		16×112×112	
R2D	11.4M	46.7	59.5	47.0	58.9
R3D	33.4M	49.4	61.8	52.5	64.2
R(2+1)D	33.3M	**52.8**	**64.8**	**56.8**	**68.0**

analogously to channels. After the first convolutional layer, the temporal information is collapsed. Its performance is recorded in Figure 2.6. Accuracy is recorded on networks trained with either 8-frame or 16-frame video clips. R3D is a ResNet-18 architecture which relies on 3D convolutions in its residual blocks. The individual filter size is $3 \times 3 \times 3$ for temporal and spatial depth of the filter. Such filter would symbolize the grey box in (c) and (d) of Figure 2.5. An additional 3D convolutional operation, which the authors present and performs strongest, is the termed (2+1)D convolution [5].

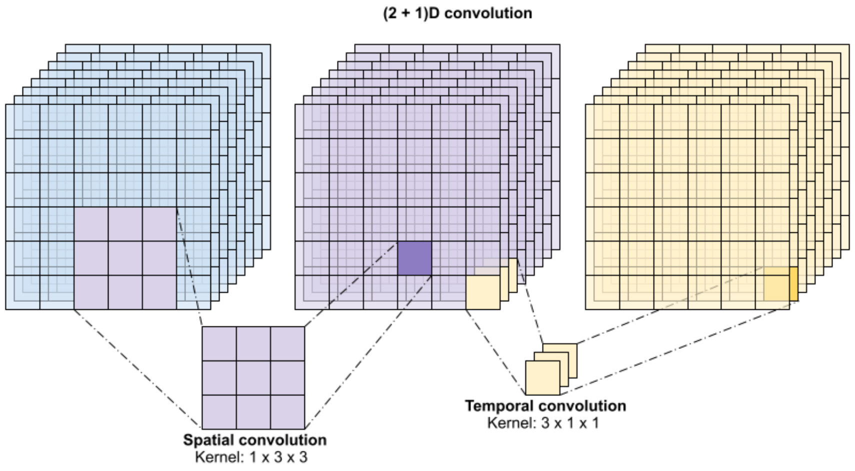


Figure 2.7: Visualization of the (2+1)D convolution operation [6]

Figure 2.7 pictures this type of convolution in detail. (2+1)D convolution factorises the operation into a spatial convolution followed by a temporal convolution. This allows us to view the operation in two steps. First step, the spatial convolution utilizes a $1 \times 3 \times 3$ filter which creates spatial feature maps. Furthermore, it maintains temporal depth opposite to the R2D architecture that collapsed the temporal information. Second step, the temporal convolution convolves alongside the frames with a $3 \times 1 \times 1$ filter to learn features of motion. Lastly, this decomposition method carries further advantages in comparison to the full 3D convolution. Because of the separation, non-linear functions can now also be placed between the spatial and temporal convolution. This effectively doubles the number of non-linearities in the network.

To sum up, Figure 2.6 compares all of these three architectures trained on the Kinetics-400 [33] video action recognition dataset. The R(2+1)D architecture achieves the best accuracy on the validation set. R3D with full 3D convolutions follows behind with 3%-3.8% less percentage points of accuracy. The network, which only considers the spatial dimensions and ignores the temporal signal, performs worst. This suggests that modeling motion does play an important factor in building the best performing deep learning architecture for videos and that spatiotemporal representations are worth learning [5].

Table 2.2 compares three backbone architectures for video representation learning which will be inspected for dimensional collapse in this thesis. Here, their supervised performance on the Kinetics-400 validation set is recorded. Furthermore, these networks find common use as backbones in many self-supervised approaches, which will be discussed later. So far, the R(2+1)D architecture has already been examined.

Network Architecture	$T \times \tau$	Top-1 Accuracy	Top-5 Accuracy	Release	Origin
3D ResNet-50 (Slow-only)	8×8	74.58%	91.63%	2019	[7,34]
ResNet(2+1)D-50	16×4	76.01%	92.23%	2018	[5,35]
SlowFast-50	8×8	76.94%	92.69%	2019	[7,36]

Table 2.2: Comparison of deep 3D Convolutional Neural Networks on Kinetics-400

2.2.2 Slow-only and SlowFast

For our brain to perceive motion in a video, we have to be shown more than 10 to 12 frames per seconds to not be able to distinguish between individual frames. For example, the frame rate of many films is 24 frames per second [30]. Having so many images displayed in such a short amount of time is mainly done for our visual system to perceive motion and not necessarily for the introduction of additional informational content. Imagine a 10 second video running at 24 fps, with a frame resolution of 360×640 pixels and each frame containing all three RGB channels. A video requires a lot more storage space than a simple RGB image. Additionally, the spatial content in a video might not change drastically during its run. Analysing all frames of a video does often not add much more insight and is computationally impractical. Instead, what frequently is done, is grabbing frames from a video clip following a certain selection pattern. In Table 2.2, the trained ResNet(2+1)D-50 network is described as having a 16×4 configuration. Here, 16 stands for the number of frames or frame length , and 4 stands for the sampling rate. This particular network model was trained with video clips that contain 16 frames from the original video and which were taken in an interval of 4 frames. Every fourth frame was sampled. Following this procedure, the temporal dimension of a video can be estimated while being more moderate in the number of frames which must be analysed per video clip. On the other hand, this introduces yet another hyperparameter for which must be adjusted for.

Motion and visual content of a video usually do not contribute equally. The R(2+1)D network [5] already started a trend of factorizing 3D convolutions in its spatial and temporal segments to analyze them separately. The SlowFast architecture [7] expands on the idea of spatial and temporal separation. Figure 2.8 pictures this SlowFast construction. The CNN consists of two pathways, a Slow pathway and a Fast pathway. The Slow pathway examines a video clip that has been sparsely sampled while the Fast pathway inspects the same clip of high frame rate,

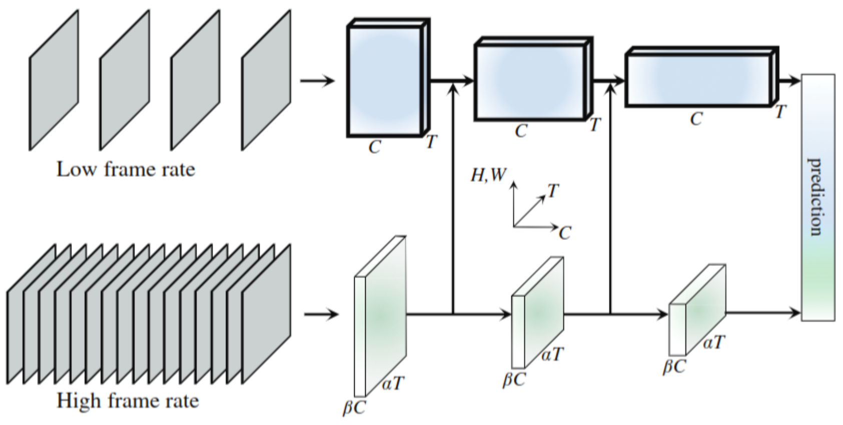


Figure 2.8: Illustration of two pathway SlowFast architecture [7]

hence the name SlowFast. The pathways analyse the same video set on two different temporal speeds. Furthermore, the channel capacity differentiates both pathways [7].

There exists two additional hyperparameters to this network, α and β, which are also pictured in Figure 2.8. The trained SlowFast network in Table 2.2 is noted as having a 8×8 frame rate and sampling rate. This references the settings of the Slow pathway. α and β are parameters to relate both pathways. α dictates the speed ratio and is always strictly larger than one. If the Slow pathway has a frame rate of T, the Fast pathway will have a higher frame rate of αT. Inversely, β dictates the channel ratio and is always strictly smaller than one. If the Slow pathway has channel size of C at a particular depth, the Fast pathway will hold βC channels at the same network depth. As such, more feature maps are being created for the Slow pathway to thoroughly analyse those few frames semantically that are given as input to this pathway. On the other hand, the Fast pathway has a lower channel capacity but this makes it also computationally more lightweight. Additionally, its role is focused on the temporal signal. As the high frame rate clip passes through the Fast pathway, the network obtains high temporal resolution features since temporal pooling or striding is not used throughout the pathway [7].

As many architectures before, for both pathways, the SlowFast network utilizes the residual blocks of the ResNet [3] architecture. During the majority of stages, information is also passed from the Fast pathway into the Slow pathway to merge features. So, one pathway is not oblivious to the representation learned by the other pathway [7].

Additionally, the convolution operations in both pathways are not symmetrical. The Fast pathway utilizes spatial and temporal convolutions during all five stages of the ResNet while the Slow pathway only makes use of temporal convolutions in its last two. Experiments unveil that using temporal convolutions in earlier layers degrades accuracy [7]. The authors argue that when sampling frames far from each other and objects move fast, there is little correlation within a temporal receptive field unless the spatial field is large enough like in later layers [7].

Table 2.2 shows that the SlowFast network outperforms the R(2+1)D network slightly with an architecture of the same layer depth. 3D ResNet (Slow-only) is a CNN which only uses the Slow pathway of the SlowFast network, therefore the fitting name Slow-only. It performs about 2.3% worse than the SlowFast network. The reason for including the Slow-only network in this comparison table is that it is used frequently as a baseline backbone encoder for self-supervised methods to learn video representations.

2.3 Self-Supervised Image Representation Learning

Now, let us go over the self-supervised learning methods that allow to learn helpful representations of images or videos without relying them to be manually labelled. Previously, the introduced backbone architectures have all been trained through supervised classification. In self-supervision, the labelling is drawn from more variety, the data itself. Multiple different and new approaches have been investigated in the recent years to create a proxy supervision signal to learn semantic content of media. Table 2.3 shows a list of self-curated popular self-supervised representation learning methods. These named approaches will be summarized shortly and later explored further for dimensional collapse. All of the pretrained network models have achieved their respective listed linear evaluation accuracy using a common ResNet-50 backbone. The ResNet architecture has been introduced in section 2.1. Linear evaluation entails to judge the representation quality of a pretrained model using a single linear classifier. Furthermore, the methods presented in Table 2.3 are all image-based self-supervised learning (SSL) methods. There are several reasons to start with inspecting image-based self-supervised methods. First, images are what make up videos, and therefore, offer an interesting baseline and comparison for video-based SSL methods. Secondly, new ideas and approaches are typically first researched for the image domain. For example, BarlowTwins [14] and VICReg [15] are two new promising approaches in information maximization that have not yet been produced for video representation learning. Video-based methods often follow suit and make use of the advances from the image field like in the work by Feichtenhofer et al. [16], which will play a bigger part later on in the thesis. Concluding, video-based self-supervised learning methods will succeed the image-based approaches. Many of those video-based methods will appear familiar to the image-based ones introduced in this section.

SSL Method	Epochs	Top-1 Accuracy	Top-5 Accuracy	Release	Origin
RotNet	105	48.2%	-	2018	[9,37]
Jigsaw	105	48.57%	-	2017	[8,37]
SimCLR-v1	800	69.68%	-	2020	[10,37]
MoCo-v2	800	71.1%	-	2020	[11,38,39]
VICReg	1000	73.2%	91.1%	2022	[15,40]
BarlowTwins	1000	73.5%	91%	2021	[14,41]
BYOL	1000	74.4%	-	2020	[13,42]
SwAV	800	75.3%	-	2021	[12,43]
Supervised	90	76.13%	92.86%	2015	[3,27]

Table 2.3: Image-based SSL methods with backbone ResNet-50 on ImageNet-1k

2.3.1 Pretext Tasks

From here on out, we are going to traverse the listed image-based SSL methods in Table 2.3 chronologically, starting with the pretext tasks Jigsaw [8] and RotNet [9]. Pretext tasks are handcrafted tasks which try to outwit the network to learn image features. The idea behind pretext tasks is, that in order to solve the tasks, the network needs to be able to understand the semantics behind the image. In short, the network solves a task under the *pretext* of learning features of images, hence the name. An example of such pretext task would be the colorization of single channel images [44]. A network would not be able solve a task like this without deeper semantic knowledge of the image. Jigsaw [8] and RotNet [9] are further examples. These pretext tasks are not state-of-the-art anymore, but based on their different approach, offer an interesting comparison to more current methods.

As some might expect, the name Jigsaw is in reference to the famous jigsaw puzzles. An image of a large city skyline or an image of a cute animal, to name a few, has been divided into many small tiles of differently shaped pieces. The goal is it stitch those pieces back together to form the whole image. Since their inception, Jigsaw puzzles have been associated with learning [8]. To successfully complete such a puzzle, a player needs to be able to relate local image tiles to the full image. Colours need to be matched, the context of the surroundings must be taken into account and common objects need to be recognized. Additionally, in the actual game, a player often makes use of the outer shape of pieces to match them together correctly, but this is not a cue used in the neural network version of Jigsaw [8].

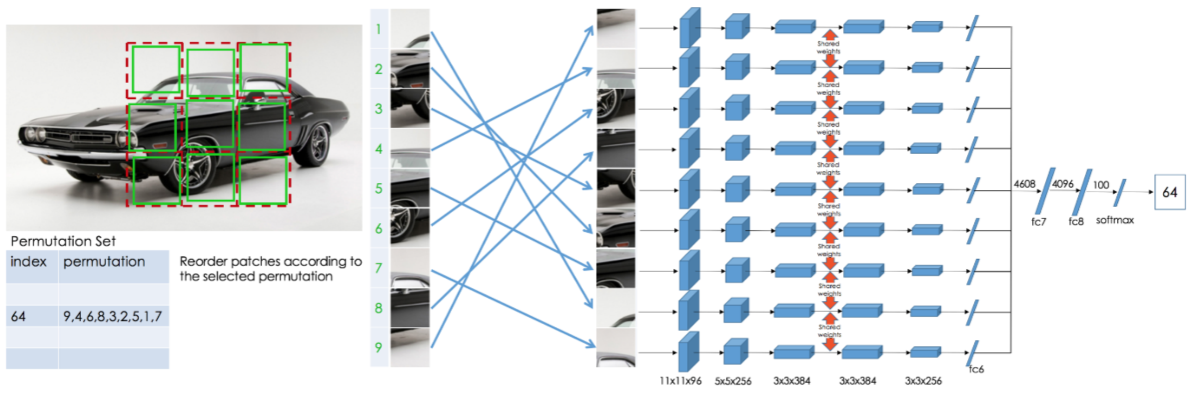


Figure 2.9: Reshuffling and architecture overview of the Jigsaw pretext task [8]

The architecture, which is supposed to solve the Jigsaw pretext task, is titled context-free network (CFN). Its structure is displayed in Figure 2.9. Before an input image is entered into the CFN, it must be prepared. A 225×225 crop of the input image is taken, and subsequently, the image is broken up into 9 tiles like shown in Figure 2.9. Each of the 75×75 tiles is then again randomly cropped into 64×64 pixel tiles. This follows an arbitrary permutation of the 9 tiles. Since there exist 9 tiles, $9! = 362{,}880$ permutations are possible. Because differentiating between 362,880 possible permutations is quite the overkill, a subset of permutations is used in the actual task. For instance, the hyperparameter for the number of permutations of the trained network model in Table 2.3 is 100. Each individual tile is uniquely indexed, and therefore, each permutation has its own special list of indices. This list is then again assigned a unique identifier. The task of the CFN is to predict the scalar identifier of the chosen permutation [8].

In the original paper, the CFN architecture is based on the AlexNet architecture. AlexNet has previously been discussed in section 2.1. Like displayed in Figure 2.9, each image tile is being individually processed by an AlexNet architecture with shared weights before their intermediate representations are concatenated and given as input to a fully connected layer. This architecture is called context-free because of the separated data flow of each tile. Context is only being handled by the fully connected linear layers in the last section of the network [8]. Using AlexNet, Jigsaw records an accuracy of 34.82% on ImageNet-1k [37]. The variant with a ResNet-50 backbone, which we will inspect later, achieves an accuracy of 48.57% as listed in Table 2.3. An improvement based on the backbone encoder of 13.75%.

Lastly, for pretext tasks, it is important to make sure that the network solves the task without cheating the true purpose of the task. It is crucial to avoid that the CFN accomplishes the task of solving a Jigsaw puzzle without learning significant image features in return. This could be the case when only one Jigsaw puzzle is being created per image. The network would tend to learn an absolute position rather than semantic meaning of the images. To avoid it multiple Jigsaw puzzle configurations are used on the same image during training [8].

RotNet is another example of a popular pretext task for image-based self-supervised representation learning. The crafted task behind RotNet is even more straightforward than for Jigsaw. The goal is for an convolutional network to predict the rotation applied to an input image. To accurately predict the rotation of the input image, the model has to learn to localize salient objects within the image. Furthermore, it must be able to understand their orientation and relate it to the prominent, mostly upright, orientation of that specific object type [9]. In the original paper, variants of the AlexNet architecture were used. In this thesis, we will inspect the ResNet architecture as a possible backbone. Accuracy with ResNet-50 on ImageNet-1k is recorded in Table 2.3.

The log-loss function for RotNet, which judges how well the network recognizes those geometric transformations, is described as follows [9]:

$$loss(X_i, \theta) = -\frac{1}{K}\sum_{y=1}^{K} \log(F^y(g(X_i|y)|\theta)).$$

K stands for the number of chosen rotation transformations. Gidaris et al. [9] discovered, using a non-linear evaluation protocol, that distinguishing between four discrete rotations scores the highest accuracy. Two transformations seem too few to learn and eight transformations might not be distinguishable enough. So, the chosen rotations are in the steps of 90°(0°, 90°, 180°and 270°). Furthermore, $g(X_i|y)$ applies the rotation of label y to image X_i, and $F(.)$ describes the convolutional model. In particular, $F^y(g(X_i|y)|\theta)$ stands for the predicted probability of rotation transformation y given the learnable parameters θ of the network [9]. The total loss is then computed over a certain mini-batch, often called batch as an abbreviation. This setup as a transformation classification task attempts to force the model to learn semantic representations useful for visual perception tasks [9].

First indications of RotNet's validity are shown through its generated attention maps. Attention maps are computed based on magnitude of feature activations at each spatial position of a convolutional layer and roughly approximate the focus of a network [9]. So, it can be observed that focus is placed on high level object parts to achieve the rotation prediction task. In comparison, attention of the same network trained in a supervised way following a object recognition task, shows that attention is spent on around the same high level regions as in the self-supervised rotation task [9].

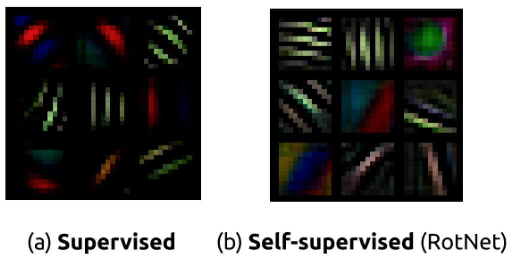


Figure 2.10: First layer filters of AlexNet [9]

Lastly, an important insight is also won by inspecting the filters of the first convolutional layer of RotNet. Figure 2.10 displays a selection of visualized filters. In (a) the filters of a supervised trained AlexNet are shown, and in (b) filters of AlexNet trained as RotNet are shown. Surprisingly, the presented self-supervised filters are more expressive and show a bit more of variety than the supervised ones. Noticeably, the self-supervised filters contain many oriented edge filters which is most likely related to the pretext task of rotation classification [9]. The style of pretext task is engraved in its filters.

Pretext tasks are handcrafted tasks based on the heuristics of its designer [10]. Such a self-supervised learning system might lead to representations that are suitable to solve the pretext task, but might not be as good in solving certain downstream tasks [8]. The adaption to specialized scenarios like, for example, the rotation transformation classification in RotNet or puzzle solving in Jigsaw might limit the generality of learned representations.

2.3.2 SimCLR

A big next step in the field of self-supervised learning was achieved by SimCLR, a simple framework for contrastive learning of visual representations [10]. SimCLR made use of many different advancements throughout the field of self-supervised learning and combined them effectively in a new framework. The approach differs from standard supervised learning only in the choice of data augmentation, loss function and the additional use of a non-linear projection head [10]. The backbone encoder remains unaltered. As choice for network architecture, the commonly used ResNet is adopted for its simplicity [10]. This architecture was previously presented in section 2.1.

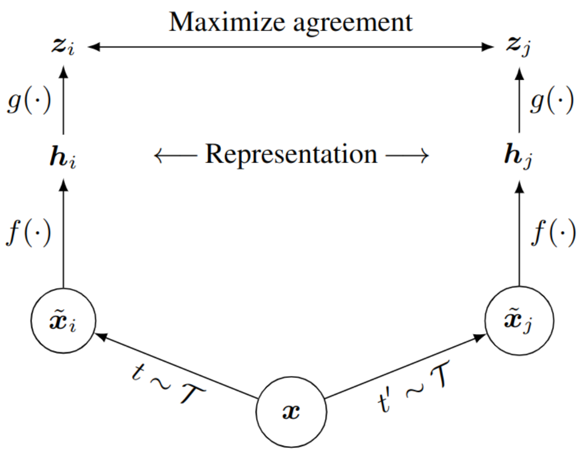


Figure 2.11: Basic overview of SimCLR structure [10]

Figure 2.11 displays a basic joint layout of SimCLR's progression for a single input image x. $t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$ are two sets for randomly drawn image augmentations which are applied to said image. These correlated views, $t(x) = \tilde{x}_i$ and $t'(x) = \tilde{x}_j$, are then fed to the backbone neural network, $f(.)$. In the original paper and also in our future experiments, a ResNet model is used for encoder f. The following output of the encoder f are the image representations h_i and h_j. The siamese-style sturcture in Figure 2.11 is more so for visualization purposes. Both representations, h_i and h_j, are part of the same mini-batch and are computed using the identical network f with learnable parameters θ. The backbone follows a two-layer multilayer perceptron (MLP) projection head, $g(.)$, a new addition to SimCLR. This added projection head is non-linear thanks to a ReLU non-linearity between both hidden layers. Its functionality is to translate the image representations into a lower-dimensional embedding space reducing their dimensions further. Most commonly in SimCLR, this lower embedding space spans 128 dimensions while the previous output vector of a ResNet-50 model after a last global averaging pooling layer spans 2048 dimensions. In a last step, these low-dimensional embeddings are sent to a contrastive loss function. The goal is to train a network which maximizes agreement between embeddings originating from the same image. In its simplest form, the network learns visual representations by learning to be invariant towards data augmentation. Lastly, for downstream tasks, the projection head g is being discarded and the representations won through the backbone f are used [10]. Contrastive loss, non-linear projection head and strong data augmentations are the crux of SimCLR. More so, these topics will stay relevant throughout this thesis for different methods in video and image learning.

SimCLR follows a contrastive prediction task. Provided a set of images $\{\tilde{x}_k\}$ including a positive pair $\tilde{x}_i$ and $\tilde{x}_j$, the contrastive prediction task aims to identify image $\tilde{x}_j$ in $\{\tilde{x}_k\}_{k \neq i}$ for a given image $\tilde{x}_i$ [10]. Like in the example above, the positive pair, $\tilde{x}_i$ and $\tilde{x}_j$, stems from the same image. The remaining images in set $\{\tilde{x}_k\}_{k \neq i \neq j}$ are considered negative examples towards the positive pair. If the N examples in a mini-batch are accounted for, there will exist a total number of $2N$ images, because of the augmented pairs of each example. Furthermore, $2(N-1)$ augmented images can be treated as negative examples for each pair in the contrastive prediction task. Batch size indicates the number of negative examples and plays in contrastive learning an even more elevated role. For short training periods (e.g. 100 epochs), large batch sizes have a significant advantage over small ones (e.g. 4096 vs 256) with an accuracy difference of roughly 7%. With longer training periods, this difference subsides [10].

The contrastive loss for SimCLR of positive image pair (i,j) is stated as follows [10]:

$$\ell_{i,j} = -\log \frac{exp(sim(z_i,z_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq i]} exp(sim(z_i,z_k)/\tau)}.$$

The loss is named NT-Xent for normalized temperature-scaled cross entropy loss. In the equation, sim stands for the cosine similarity which denotes the dot product between l_2 normalized embedding vectors $\mathbf{u}$ and $\mathbf{v}$ (i.e. $sim(\mathbf{u},\mathbf{v}) = \mathbf{u}^T\mathbf{v}/\|\mathbf{u}\|\|\mathbf{v}\|$) [10]. τ hereby is a temperature value that scales the similarity scores. It is also a hyperparameter to the loss. The altered scores can be viewed as logits that are passed to a Softmax-like function. The score of the positive pair is placed in the numerator while the remaining scores are placed in the denominator. With the difference that the score between i itself is not considered. Finally, the loss is averaged between all positive pairs. Symmetries (i,j) and (j,i) are included [10]. The goal of the contrastive loss is to evaluate the network's progress and makes sure that positive pairs of embedding vectors expand their similarities while negative pairs are pushed away. Importantly, this dual push and pull effect helps to prevent the collapse pattern of complete collapse. Treating positive and negative pairs differently in the loss function, ensures that the model does not learn trivial solutions where all the input images are mapped to the same constant output vector [1]. As a side note, many previous contrastive learning methods related their success to maximization of mutual information between latent representations [10]. However, it has been shown that strictly maximizing lower bounds on mutual information can lead to bad representations, while the success of methods like SimCLR strongly depends on the inductive bias in both the choice of neural network architecture and specific form of the contrastive loss [10, 45].

Without question, data augmentation is vital for SimCLR. It defines the contrastive prediction task since learning invariance is part of the objective. Furthermore, no singular data augmentation is enough to learn good representations. Compositions of transformations are required. The prediction task becomes harder to solve but representations of higher quality are learned in return [10]. The combination of random cropping and random colour distortion are shown to perform best under linear evaluation. Colour distortion is an important transformation since it also alters the colour histograms of images. Using only cropping, a network may exploit a shortcut through the histogram to solve the prediction task. Lastly, self-supervised contrastive learning also benefits from stronger data augmentations than supervised learning does. Data augmentations, which do not improve performance in supervised learning, might still yet help in the context of contrastive learning [10].

The added non-linear projection head, which SimCLR introduced, is the last pivotal component to consider. Especially, since it will be a talking point throughout the thesis. The projector extends the backbone architecture and consists in SimCLR of two linear weight layers with one ReLU non-linearity in between. Furthermore, the projector is only used during training. After training, the projector is discarded and the representations from the backbone are used for linear evaluation or other downstream tasks. The improvement, which the non-linear projection head accounts for, is substantial. In linear evaluation, the model with non-linear projector performs better than a model with linear projector (+3%) and much better than a model with no projector (10%). As for the reason why, the authors hypothesize that, in particular, because the embedding vectors are trained to be invariant to data augmentations, the projector can remove color and object orientation information that may be useful for downstream tasks. While this type of information would be removed from the embeddings, the information can still be retained in the representation vectors [10]. We will learn moving forward in this thesis that the usefulness of non-linear projection heads may be present in even more areas.

2.3.3 MoCo

MoCo, Momentum Contrast for Unsupervised Visual Representation Learning [11], is another contrastive learning framework. MoCo uses the concept of a dictionary to contrast between representations. The structure of MoCo is shown in Figure 2.12. On first view, MoCo is more convoluted than SimCLR yet it is an effective contrastive method. It was able to outperform supervised pretrained counterparts in 7 detection and segmentation task, manifesting the relevance of self-supervised representation learning further [11].

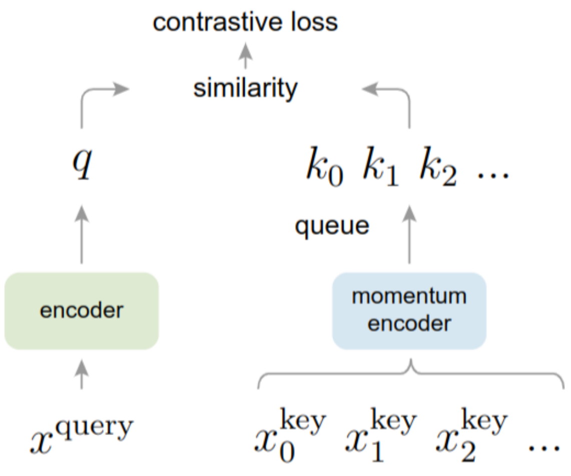


Figure 2.12: Basic overview of MoCo structure [11]

A contrastive loss is a function whose value is low when the element is similar to its positively related element and dissimilar to all other unrelated elements [11]. This is alike to a classification of a key using its related query. In SimCLR, those positive and negative pairs were constructed via the mini-batch. MoCo, on the other hand, explains contrastive loss in terms of dictionary look-up where pairs are query and key pairs. A query q matches to only one key, noted k_+. This sets up the positive pair. Otherwise, the dictionary is filled with keys of set $\{k_0, k_1, k_2, ...\}$ including the positive key and K negative keys. In implementation, the set is viewed as a queue. Different to SimCLR, this queue is not limited by the size of one mini-batch. The dictionary can include keys of multiple mini-batches which allows the loss to contrast the positive query against a great variety of keys. This contrastive loss used in MoCo is slightly different to SimCLR's loss. It is called InfoNCE [46] and is formulated as follows [11]:

$$\mathcal{L}_q = -\log \frac{exp(q \cdot k_+/\tau)}{\sum_{i=0}^{K} exp(q \cdot k_i/\tau)}.$$

Whereas NT-Xent used cosine similarity as measure, InfoNCE uses a simple dot product. Additionally, all keys are considered in the denominator of the loss.

Queries and keys are the encoded images depicted through their representations. As seen in Figure 2.12, queries and keys are encoded through two different models, $q = f_q(x^q)$ and $k = f_k(x^k)$. In comparison to SimCLR, where both views were computed using the same network. The queue in MoCo is able to hold samples much larger than one mini-batch can hold. The dictionary is supposed to always represent a sampled subset of all data [11]. To manage this subset, the oldest mini-batch is removed from the queue while the newest mini-batch gets added. Since, through training, the representations can change rapidly, this can also be advantageous because the oldest mini-batch is the most outdated and inconsistent one. Consistency is especially important for the momentum encoder, shown in Figure 2.12, which produces the keys for a large dictionary queue spanning more than one mini-batch. Using the same network for encoder and momentum encoder fails. The hypothesis is that such failure is caused by the rapidly changing encoder which reduces the key representations' consistency [11]. Proposed is a momentum update, $\theta_k \leftarrow m\theta_k + (1 - m)\theta_q$ [11]. θ_q denotes the parameters of the encoder and θ_k denotes the parameters of the momentum encoder. This is also the reason for the name momentum encoder. Only the parameters of the encoder are updated by back-propagation while the parameters of the momentum encoder evolve more smoothly alongside to represent the dictionary. Regarding the coefficient m, large momentum values like $m = 0.999$ have shown to produce the best results [11].

Like in SimCLR, the encoder, theoretically, could stand for any kind of machine learning model, and like in SimCLR, the ResNet architecture is chosen. This will remain true for the remaining approaches in this chapter about self-supervised image representation learning. After training, the encoder can be extracted and used for downstream tasks. Remarkably, MoCo is able to outperform supervised pretrained counterparts on ImageNet in seven detection and segmentation downstream task. Namely, object detection on VOC [47]/COCO [48], instance segmentation on COCO/LVIS [49], keypoint detection on COCO, dense pose on COCO, and semantic segmentation on Cityscapes [11, 50]. Furthermore, in all these tasks, MoCo pretrained on the Instagram-1B dataset [51] is consistently better than MoCo pretrained on the ImageNet-1M dataset. This shows that MoCo can perform well on large-scale, relatively uncurated datasets like the Instagram-1B dataset [11]. In most papers, ImageNet is used to train self-supervised methods. However, this does not challenge the need for self-supervised learning quite yet since ImageNet is a labelled dataset and could always also be used in such a way. The results of MoCo reveal more of the potential behind self-supervised learning on unlabelled data.

As presented in Table 2.3, we will later inspect MoCo-v2, an updated version of the MoCo framework described above. MoCo-v2 integrates two advancements from SimCLR, which establishes a stronger baseline that even outperforms SimCLR [39]. Namely, those advancements are the non-linear projection head and additional data augmentations. Such, SimCLR and MoCo-v2 have much in common. As a difference, this leaves MoCo-v2 still with the ability to hold batches worth of negative samples thanks to its dictionary. SimCLR, on the other hand, needs to make use of very large mini-batches, which is memory-wise expensive and limiting, to carry negative samples. This discrepancy is plausibly the reason for the performance difference of both contrastive approaches.

2.3.4 SwAV

SwAV, Swapping Assignments between multiple Views [12], is a framework which utilizes clustering for self-supervised learning. The last two approaches have shown that increasing the number of negative samples is going to increase the performance of the model. Positive samples maximize agreement while negative samples, belonging to a foreign image, enforce their dissimilarity. SimCLR is restricted of only using negative samples from a mini-batch to approximate the negative samples in a dataset, MoCo extends a dictionary to go past the limitations of a mini-batch and SwAV relies on cluster formations and assignments which are build of the dataset throughout the training. Previously, to contrast between views, many direct comparisons between image embedding vectors were necessary. On one hand, it is important in a contrastive loss to keep the number of negatives high while, on the other hand, it becomes computationally more and more unpractical to compute all these pairwise feature comparisons [12]. Cluster-based methods alleviate this problem by not contrasting a view with every single negative sample but by contrasting with its cluster affiliation. In SimCLR and MoCo, augmentations of the same image were treated as a collective pseudo class during the contrastive loss. SwAV applies clusters, group of related images, as their pseudo classes. Cluster-based methods have formerly been trained in two steps where, in an offline-fashion, cluster assignments of features are determined to create targets, and during the training step, those assignment codes are then predicted for different image views. This does not scale well considering the pass over the full dataset to compute the cluster assignments. SwAV presents an online methodology of cluster assignment computation to address this situation [12].

Besides, SwAV, like MoCo before, makes use of the data augmentation compositions and projection head introduced in SimCLR [10] with the difference of the projection head containing a batch normalization layer following its first linear layer. Additionally, SwAV does not rely on a momentum encoder like MoCo.

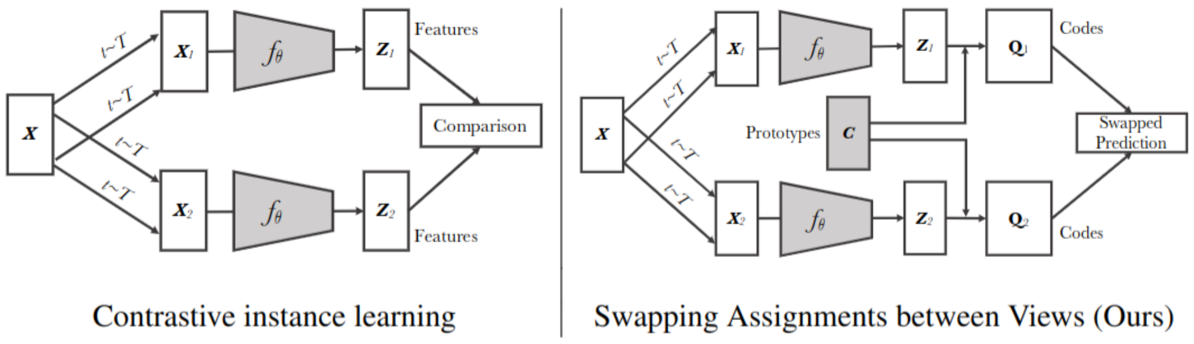

X
t~T
X1
X2
fθ
Z1
Z2
Features
Comparison
Contrastive instance learning
Prototypes
C
Q1
Q2
Codes
Swapped Prediction
Swapping Assignments between Views (Ours)

Figure 2.13: Comparison of Contrastive and SwAV approach [12]

Figure 2.13 compares between a typical contrastive structure (e.g. SimCLR) and the setup structure of SwAV. Similar to both, the backbone encoder f_θ computes embedding vectors for differently augmented views of the same image. Furthermore, SwAV projects the embeddings to the unit sphere (i.e. $z = f_\theta(t(x))/\|f_\theta(t(x))\|_2$ for image x and embedding z). How the network learns for representations to be invariant to those transformations is where the methods differ. SwAV contrasts clusters instead of individual images. Those cluster prototypes are held as columns in matrix $C = [c_1,...,c_K]$. A typical value for K, the number of clusters, is 3000. The codes which determine the cluster assignment for each embedding are held in matrix Q [12]. The loss is defined by a "swapped" prediction problem, such the name [12]:

$$L(z_t, z_s) = \ell(z_t, q_s) + \ell(z_s, q_t).$$

z_t and z_s are embeddings of different augmentations of the same image and q_t and q_s are cluster assignment codes of the same image. The idea is that if both feature embeddings share the same underlying information, the cluster assignment of one view should be able to be predicted by the features other view and vice versa. This swapped prediction $\ell(z_t, q_s)$ is further expressed as follows [12]:

$$\ell(z_t, q_s) = -\sum_k q_s^{(k)} \log p_t^{(k)}, \quad \text{where} \quad p_t^{(k)} = \frac{exp(z_t^T c_k/\tau)}{\sum_{k'} exp(z_t^T c_{k'})}.$$

The probability function $p_t^{(k)}$ contains the softmax function which is often used to weigh the similarity of a positive pair in relation to the similarity with all the negative samples (e.g. InfoNCE and NT-Xent). Here, it contrasts the similarity of embedding z_t with cluster prototype c_k in relation to all clusters, instead of over all negative samples. Lastly, the cross entropy loss between cluster assignment and probability is build [12].

What remains, is to determine how the clusters are assigned during training and how matrix Q is build to maximize the similarity between features and prototypes. The cluster prototypes themselves alongside the encoder are being trained via the backpropagation algorithm. An important criteria to ensure is that the assignment codes to clusters are equally partitioned [12]. SwAV wants to prevent that different images in a mini-batch receive the same code and are therefore not distinct, which can lead to trivial solutions and collapse. The mapping from the feature embedding matrix Z of a mini-batch to the prototype matrix C is described by matrix Q which holds the assignment codes. To optimize Q for equipartition and to maximize similarity, the following optimization problem is posed [12]:

$$\max_{Q \in \mathcal{Q}} \operatorname{Tr}(Q^T C^T Z) + \epsilon H(Q).$$

Set $\mathcal{Q}$ enforces equipartition of Q in the current mini-batch while H adds an entropy regularization term. Furthermore, the Sinkhorn-Knopp algorithm [52] is used to solve the optimization problem and to obtain the cluster assignment codes [12].

An additional feature introduced alongside SwAV is their new data augmentation strategy, multi-crop. The idea is to sample a number of additional low resolution crops that cover only small parts of the image besides the two standard resolution crops. Furthermore, cluster assignment codes are still just computed for the full resolution crops. This ensures that the increase in computing cost is minimal but the accuracy gain of the additional crops is notable. SwAV improves by 4% on linear evaluation with ImageNet. Also, SimCLR can profit from multi-crop with an increase of around 2% on performance [12].

Lastly, SwAV, similar to MoCo, is able to outperform supervised pretraining on ImageNet for several downstream tasks but does so for the first time without needing to finetune its neural encoder. Only a linear classifier is trained on top of the frozen network [12]. Combining all these advances, SwAV is also the best performing image-based self-supervised method which we will inspect, as seen in Table 2.3.

2.3.5 BYOL

For the past approaches, how to handle negative samples, how to increase the amount of negative samples and how to manage the increasing pairwise comparison cost played an important role. Bootstrap Your Own Latent (BYOL) [13], on the other hand, is a method that can do without any negative samples. It belongs to the class of non-contrastive methods. BYOL is sometimes also referred to as a distillation method. Previously, a central reason for using contrastive methods, contrasting between positive and negative examples, was to avoid trivial solutions and complete collapse. The state where maximizing similarity between augmented views of the same image leads to a constant representation vector, irregardless of input. BYOL successfully avoids complete collapse while not relying on the additional negative samples [13]. Whether this is also true for dimensional collapse, remains to be seen.

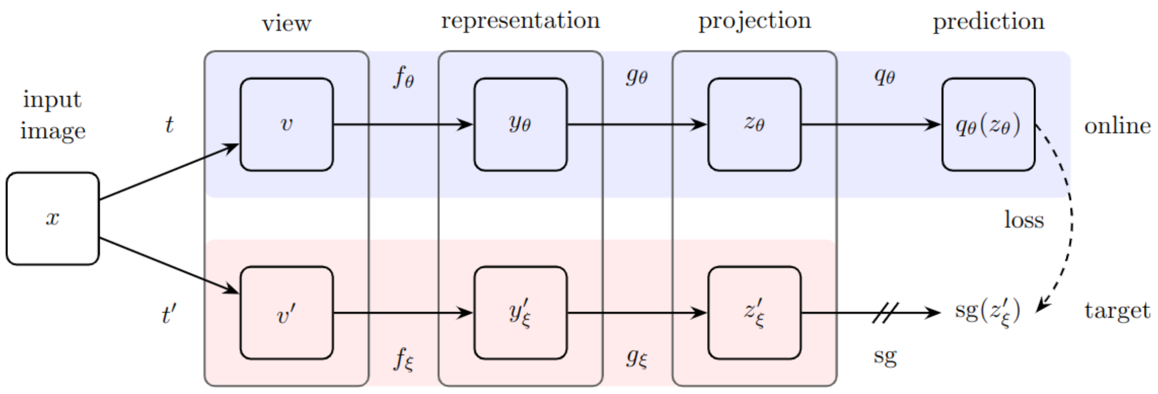


Figure 2.14: Overview of BYOL's approach [13]

Figure 2.14 shows BYOL's setup. Again two differently augmented views of the same image x are created. Two separate neural networks receive these views. The networks are called online and target network and are supposed to interact and learn from each other. In short, the goal in BYOL is to train the online network to predict the target network's representation [13]. The online network computes its representation of one view (i.e. $y_\theta = f_\theta(v)$) while the target network computes its representation of the other view of image x (i.e. $y'_\xi = f_\xi(v')$). To be more precise, the goal of the online network is to predict the target's embedding. Therefore, embeddings, $z_\theta = g_\theta(y_\theta)$ and $z'_\xi = g_\xi(y'_\xi)$, for both networks are computed by evaluating their representations over the projection head of each individual network [13].

Considering a ResNet-50 encoder for both f_θ and f_ξ, the output dimensionality for the representations, y_θ and y'_ξ, will be 2048. What is different to projection heads seen already in other methods, is that g_θ and g_ξ project the dimensions up to 4096 dimensions before decreasing them to the desired embedding output dimensionality of 256. Previously, the first linear layer of a projection head maintained the number of dimensions. Additionally, a batch normalization layer follows this first linear layer in BYOL [13].

After the embedding vectors for both sides are computed, the online network takes the extra step to predict the embedding of the target network. This causes for asymmetry. The prediction head q_θ follows the same architecture as prior projection head g_θ. Before comparing online prediction and target embedding, both are normalized (i.e. $\bar{q}_\theta(z_\theta) = q_\theta(z_\theta)/\|q_\theta(z_\theta)\|_2$ and $\bar{z}_\xi{}' = z'_\xi/\|z'_\xi\|_2$). What follows, is a mean squared loss that judges the accuracy of the prediction [13]:

$$\mathcal{L}_{\theta,\xi} = \|\bar{q}_\theta(z_\theta) - \bar{z}_\xi{}'\|_2^2.$$

This loss is symmetric such that the embedding of the first view has to predict the second but also vice versa. The total loss, $\mathcal{L}_{\theta,\xi}^{BYOL} = \mathcal{L}_{\theta,\xi} + \tilde{\mathcal{L}}_{\theta,\xi}$, is used to optimize the online network via backpropagation. Meanwhile, the target network utilizes a stop gradient (i.e. sg) operation and is not optimized. The target's weights ξ are updated through a slow-moving average of the trained online network: $\xi \leftarrow \tau\xi + (1-\tau)\theta$ [13]. Therefore, the target network can also be considered a momentum encoder since it uses the same update pattern as MoCo.

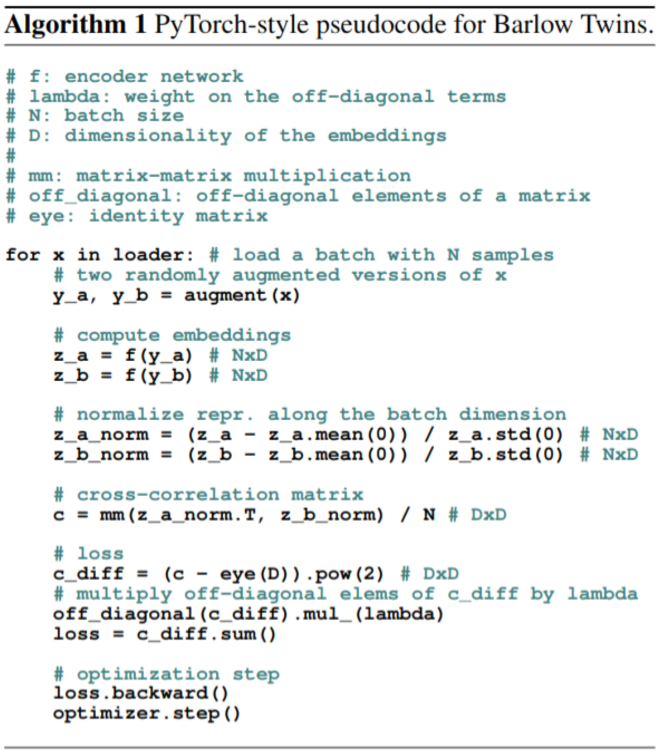
Finally, after training, the online encoder f_θ can be extracted, to be used for further downstream tasks. Even without any contrastive samples, BYOL is able to outperform both SimCLR and MoCo-v2 in linear evaluation of ImageNet, as seen in Table 2.3. Additionally, BYOL is shown to be more stable to the removal of certain augmentations than SimCLR [13].

A question that still remains is how BYOL avoids the complete collapse of representations to constant vectors. Some mention the addition of the predictor or the slow-moving average [13], while others postulate the importance of batch-wise or feature-wise normalization [15].

2.3.6 BarlowTwins

This literature review inspected so far approaches that used negative sample pairs, enabled transition from contrastive to non-contrastive, applied momentum-encoders and created asymmetric models to avoid trivial constant representations. BarlowTwins does things a little bit different to avoid complete collapse and to learn meaningful representations. BarlowTwins was named after British neuroscientist Horace Barlow. He hypothesized that the goal of sensory, visual processing is to recode highly redundant sensory inputs into a factorial code of statistically independent components [14]. The authors of BarlowTwins were influenced by this redundancy-reduction principle. They propose a loss function that reduces redundancy and, in hand, avoids collapse. Therefore, BarlowTwins is also often regarded as an information maximization approach [14].

The objective of BarlowTwins is to compute the sample cross-correlation matrix of batched outputs of differently augmented views and to drive this matrix as close as possible towards an identity matrix. What is supposed to follow, is the reduction of redundancy and correlation between dimensions of the embeddings, and furthermore, an enforcement to learn non-trivial embeddings. BarlowTwins' objective allows to maintain the overall problem in a simple form. Two augmented views of the same image are encoded and the agreement between those two embeddings is to be maximized to learn representations which are invariant to distortions. The setup is simple while the loss is constructed to optimize the invariance and redundancy [14].



Algorithm 1 PyTorch-style pseudocode for Barlow Twins.

```
# f: encoder network
# lambda: weight on the off-diagonal terms
# N: batch size
# D: dimensionality of the embeddings
#
# mm: matrix-matrix multiplication
# off_diagonal: off-diagonal elements of a matrix
# eye: identity matrix

for x in loader: # load a batch with N samples
    # two randomly augmented versions of x
    y_a, y_b = augment(x)

    # compute embeddings
    z_a = f(y_a) # NxD
    z_b = f(y_b) # NxD

    # normalize repr. along the batch dimension
    z_a_norm = (z_a - z_a.mean(0)) / z_a.std(0) # NxD
    z_b_norm = (z_b - z_b.mean(0)) / z_b.std(0) # NxD

    # cross-correlation matrix
    c = mm(z_a_norm.T, z_b_norm) / N # DxD

    # loss
    c_diff = (c - eye(D)).pow(2) # DxD
    # multiply off-diagonal elems of c_diff by lambda
    off_diagonal(c_diff).mul_(lambda)
    loss = c_diff.sum()

    # optimization step
    loss.backward()
    optimizer.step()
```

Figure 2.15: Pseudo-code for BarlowTwins [14]

Like in previous summaries, a sketch of the method could be presented. Instead, this time, the pseudo-code of the methodology will be inspected to examine its simplicity. The pseudo-code is presented in Figure 2.15. Like in more recent approaches, two augmented views of the same image are created and fed each through a surrogate backbone encoder f. This encoder also includes a projection head, which outputs are the batched embeddings z^A and z^B. As indicated in Figure 2.15, these embeddings are being normalized along the batch dimension before any further computations. What follows is the construction of the sample cross-correlation matrix C. This matrix has dimensions $D \times D$, while D is standing for the dimensionality of the embedding vector. A singular entry of C is measured as [14]:

$$C_{ij} = \frac{\sum_b z^A_{b,i} z^B_{b,j}}{\sqrt{\sum_b (z^A_{b,i})^2} \sqrt{\sum_b (z^B_{b,j})^2}}.$$

C_{ij} measures the correlation between the i-th dimension of batch A and the j-th dimension of batch B. Value of 1 means perfect correlation while a value of -1 means perfect anti-correlation [14].

Finally, the loss is constructed from two key terms that both make use of the created sample cross-correlation matrix C [14]:

$$\mathcal{L}_{BT} = \sum_i (1 - C_{ii})^2 + \lambda \quad \sum_i \sum_{j \neq i} C_{ij}^2.$$

The first term of the loss is the invariance term. It nudges the diagonal elements of C towards a value of 1. This enforces strong correlation between opposite dimensions in the embeddings of the two views, resulting in invariance to the applied transformations. The second term is named redundancy reduction term. It is concerned with the off-diagonal elements of C. Those elements describe the pairwise correlation or anti-correlation between non-identical dimensions of the embeddings. Forcing these entries towards 0 is supposed to decorrelate them and lead to non-redundant information in those dimensions which would also rule out trivial solutions like constant vectors [14].

An further interesting and important property of BarlowTwins, that has not occurred before in other examined methods, is related to the dimensionality of its projection head. Figure 2.16 shows that linear evaluation accuracy on ImageNet rises almost linearly to the dimensionality of the projector output. BYOL and SimCLR do not seem to be too influenced by the choice of projector dimensionality while BarlowTwins sees major performance improvements with larger embeddings. Furthermore, the common choice for backbone encoder, ResNet-50, only has 2048 dimensions in its output vector and still a projection into a space much larger than the representation space is beneficial for BarlowTwins. The embedding dimensionality for SimCLR and BYOL are 128 and 256, respectively. BarlowTwins' embeddings used for its loss function $\mathcal{L}_{BT}$ are 8192 dimensions large, much larger than its representations. More so, BarlowTwins' projection head contains three linear layers, instead of two, with batch normalizations and ReLU activations between each layer. The final output also has to pass one last batch normalization [14].

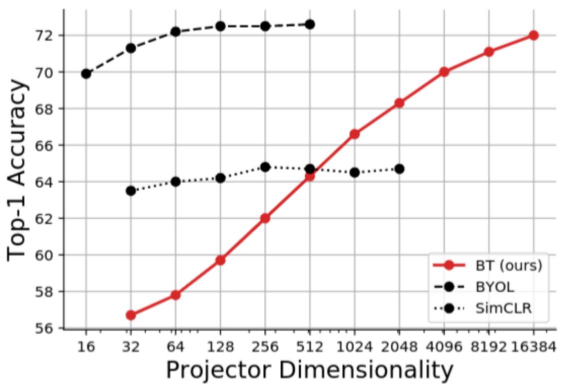
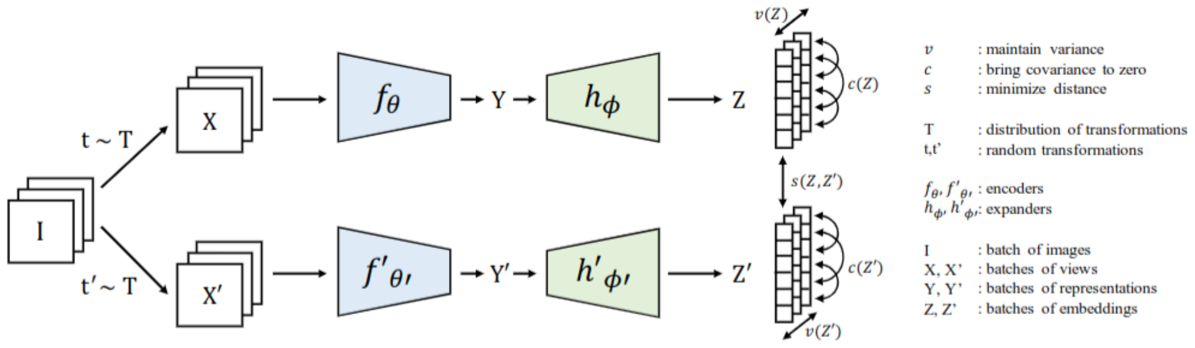


Figure 2.16: Accuracy over Projector Dimensionality of BarlowTwins [14]

While BarlowTwins does not quite outperform BYOL or SwAV, as seen in Table 2.3, it does a lot with seemingly little. This new direction of information maximization confronts collapse head on and encourages the exploration of dimensional collapse.

2.3.7 VICReg

VICReg, Variance-Invariance-Covariance Regularization [15], is a subsequent method in the area of information maximization. It sets itself apart from BarlowTwins by utilizing a loss which is even more modular. This loss function is composed of three objectives: (1) learning invariance to differently augmented views with an invariance term, (2) avoiding collapse with a variance preservation term, and (3) maximizing the information content of the representation with a covariance regularization term [15].



t ~ T
t′ ~ T
I
X
X′
f_θ
$f'_{\theta'}$
Y
Y′
h_ϕ
$h'_{\phi'}$
Z
Z′
$v(Z)$
$c(Z)$
$s(Z,Z')$
$c(Z')$
$v(Z')$
v : maintain variance
c : bring covariance to zero
s : minimize distance
T : distribution of transformations
t,t' : random transformations
$f_\theta, f'_{\theta'}$: encoders
$h_\phi, h'_{\phi'}$: expanders
I : batch of images
X, X' : batches of views
Y, Y' : batches of representations
Z, Z' : batches of embeddings

Figure 2.17: Overview of the VICReg's approach [15]

Figure 2.17 displays VICReg's structure. Another difference between VICReg and BarlowTwins is that the mentioned variance and covariance terms are applied to each branch's embeddings separately. Previously, in BarlowTwins, a cross-correlation matrix was computed over both differently augmented image batches. In VICReg, preserving information content and preventing collapse are handled independently for the two transformed batches [15].

The invariance loss term is what maximizes agreement and communicates between both batches. It is defined as the mean-squared euclidean distance between embedding matrices Z and Z', as seen in Figure 2.17. Also, n defines the batch size [15]:

$$s(Z, Z') = \frac{1}{n} \sum_i \|z_i - z'_i\|_2^2.$$

The variance preservation term is a hinge loss to maintain standard deviation over each embedding dimension in a batch [15]:

$$v(Z) = \frac{1}{d} \sum_{j=1}^{d} \max(0, \gamma - S(z^j, \epsilon)) \quad \text{where} \qquad S(x, \epsilon) = \sqrt{\mathrm{Var}(x) + \epsilon}.$$

For the variance term, S defines the regularized standard deviation, for which ϵ is a small scalar preventing numerical instabilities. Furthermore, γ is a constant target value which is fixed to 1 in most experiments, d is the dimensionality of the embeddings and z^j denotes a vector composed of each value at dimension j in all vectors in batch Z [15].

Lastly, the covariance regularization term drives the off-diagonal elements of a covariance matrix towards 0. The covariance matrix C is constructed separately for each batch. This particular loss term is inspired by the BarlowTwins' loss [15]:

$$c(Z) = \frac{1}{d}\sum_{i \neq j}[C(Z)]^2_{i,j} \quad \text{where} \qquad C(Z) = \frac{1}{n-1}\sum_{i=1}^{n}(z_i - \bar{z})(z_i - \bar{z})^T \quad \text{with} \quad \bar{z} = \frac{1}{n}\sum_{i=1}^{n} z_i.$$

A covariance matrix with off-diagonal elements close to zero infers pairwise decorrelated dimensions, same as in BarlowTwins. Highly correlated dimensions, on the other hand, infer informational collapse [15]. Furthermore, VICReg suggests that decorrelation of the dimensions at the embedding level (i.e. projector output) has also a decorrelation effect at the representation level (i.e. backbone encoder output used for transfer tasks), which is a non trivial phenomenon [15]. While the expression dimensional collapse is not explicitly mentioned during this paper, a method that has a prevention term against highly correlated dimensions is of interest to be explored later. Therefore, the reason of inclusion in this thesis.

Aside from that, as teased before, the overall loss is a weighted average of the invariance, variance and covariance terms of the two batches of embeddings, Z and Z' [15]:

$$\ell(Z,Z') = \lambda s(Z,Z') + \mu[v(Z) + v(Z')] + \nu[c(Z) + c(Z')].$$

Similar to BarlowTwins, VICReg benefits from projectors of large dimensionality (e.g. 8192). Additionally, VICReg is also a very stable algorithm. Only 0.1% difference in accuracy has been noticed between the best and worst run [15]. A further feature of VICReg is that the two branches, displayed in Figure 2.17, are not required to share the same parameters, architecture or input modality [15]. Conceptually, VICReg would also allow for the processing of videos. Unfortunately, no such network model have been trained or papers been published yet. For now, this concludes the reviews of image-based self-supervised representation learning approaches.

2.4 Self-Supervised Video Representation Learning

Now, it is time to expand the literature review to video-based self-supervised representation learning methods. With the addition of the temporal dimension, new challenges and opportunities arise. More data needs to be analysed and compressed, while novel and creative approaches can be explored. Besides already mentioned ideas, like contrastive and non-contrastive learning, additional approaches will be examined to learn spatiotemporal representations. For instance, generative approaches, which try to predict the proceeding frames of a video clip, or cross-modal approaches, that make use of the dual modalities of audio and video. Finally, Table 2.4 presents the self-curated list of self-supervised video representation learning frameworks that will be reviewed in this upcoming section. Most importantly, we will explore these methods later on for dimensional collapse.

2.4.1 SimCLR, SwAV, MoCo and BYOL

Image-based self-supervised learning approaches have experienced huge success in the recent years with some outperforming their supervised counterparts in many common image transfer tasks [12]. With videos comprising images, the idea is not too far fetched to try to make use of such image-based methods in the video domain as well. What is missing, is the incorporation of the temporal dimension into the training objective to learn effective and consistent spatiotemporal representations.

SSL Method	Backbone	Epochs	Pretrain	UCF-101	Kinetics-400	Release	Origin
DPC	3D ResNet-34	-	Kinetics-400	75.7%	-	2019	[17,53]
SimCLR	Slow-only	200	Kinetics-400	88.3%	61.5%	2021	[16,54]
SwAV	Slow-only	200	Kinetics-400	90.2%	62.6%	2021	[16,54]
MoCo	Slow-only	200	Kinetics-400	91.3%	66.6%	2021	[16,54]
BYOL	Slow-only	200	Kinetics-400	94%	67.4%	2021	[16,54]
XDC	R(2+1)D-18	-	IG-Curated	95.5%	-	2020	[18,55]
Supervised	Slow-only	-	Kinetics-400	-	74.58%	2019	[7,34]

Table 2.4: Video-based SSL methods

Feichtenhofer et al. [16] performed such a study to analyze the performance of image-based frameworks in spatiotemporal learning of video representations. They proposed a simple objective to generalize four popular image-based approaches to space-time. These approaches are, previously discussed methods, SimCLR, SwAV, MoCo and BYOL. All four of these methods share a common objective: learn augmentation-invariant representations through maximizing similarity of different augmented views of the same image. Feichtenhofer et al. [16] extend this objective to consider the temporal dimension: learn temporally-persistent representations through maximizing similarity of different augmented clips at various points in time of the same video. The hypothesis behind this idea is that visual content is often persistent and interconnected over time. If two non-identical clips of the same video are in no way related (e.g. through surroundings, objects or actions), the video will likely have changed scenery, which can be considered as a separate video in its own right. More so, the domain transition to temporally-persistent representations can be seen as an abstraction of crops in images to clips in videos [16]. Similar to crops, clips are puzzle pieces of the whole picture. This duality allows for the transition from image domain to video domain for the named frameworks [16].

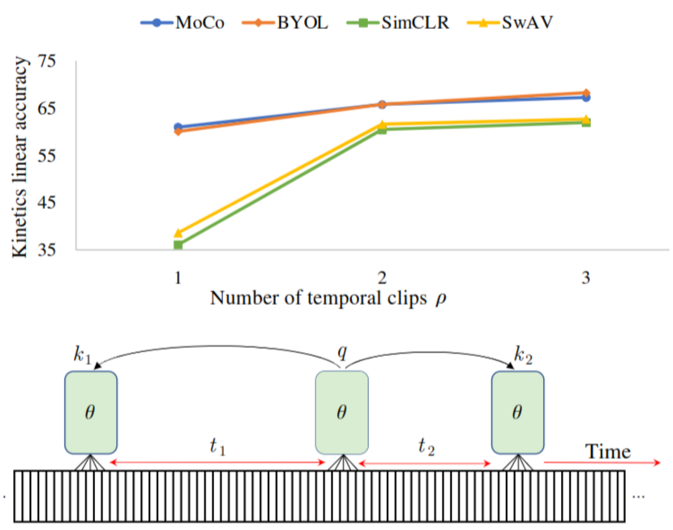
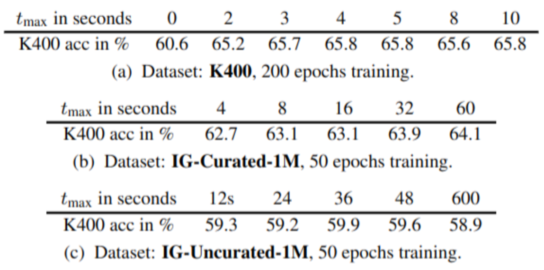


Figure 2.18: Maximizing similarity between different temporal clips of the same video [16]

BYOL, MoCo and SimCLR, with the exception of SwAV, only maximized agreement between two views. For the persistency in time, maximizing agreement between multiple clips is considered. Figure 2.18 shows such setup. The lower graphic relates three temporally separated clips with each other. The clips are sampled at unequal intervals of time. The upper graph compares performance of all four frameworks depending for the number of clips accounted for in the representation learning. For $p = 1$ (a single clip), two spatial crops are taken at the same temporal location [16]. Temporal information is only considered within the clip. Especially, SimCLR and SwAV seem to struggle under such conditions with MoCo and BYOL achieving considerable performances. For $p = 2$, temporal persistence is obtained in the video representation learning. SimCLR and SwAV recover substantially. This demonstrates the importance of temporally-persistent video representations. What strikes out, is that SimCLR and SwAV still underperform in comparison to MoCo and BYOL. The big difference between those two groups of methods is the utilization of momentum encoders in MoCo and BYOL. Momentum encoder that evolve as a moving average of the principal encoder seem to be crucial in the domain of video learning [16]. Previously, in image representation learning, this trend was not evident. More so, there is no visible performance difference of contrastive vs non-contrastive methods. This suggests that space-time persistence within a video is most important [16]. While the increase in accuracy is considerable with additional clips sampled from the same video, the difference between $p = 1$ and $p = 2$ is the starkest one [16]. In the exploration chapter for dimensional collapse, we will consider models pretrained using two clips from the same video (i.e. $p = 2$). Each individual clip will consist of 8 frames with a sample rate of 8.

Moreover, Figure 2.18 illustrates the timespan between sampled clips. In all previous experiments, Feichtenhofer et al. [16] used global temporal sampling to obtain different clips of a video. Clips were sampled unconstrained to temporal location. In a following experiment, the impact of sampling at different timespans is tested. As a metric, linear evaluation on Kinetics-400 is considered. For pretraining, three different datasets are considered (i.e. Kinetics-400 , IG-Curated-1M [56] and IG-Uncurated-1M [56]). IG-Curated-1M consists of Instagram videos that are loosely classified through hashtags that are based on Kinetics' class types. IG-Uncurated-1M contains random Instagram videos. The average duration differs between datasets. Kinetics' videos last on average 10 seconds while IG-Curated last 26 seconds and IG-Uncurated last 35 seconds. Feichtenhofer et al. [16] discovered that restricting the time window of sampled clips is not beneficial. Figure 2.19 displays the results of increasing timespans. The model pretrained on these three datasets observed two clips per video ($p = 2$) and utilized BYOL. Based on the performance of the model for each dataset, the training is steady and only the model pretrained on IG-Uncurated-1M decreases slightly in accuracy when considering long timespans of 36+ seconds. Intuitively, some might think that an increasing timespan hinders learning the proposed objective based on possible semantic context changes, but far distanced clips seem to be no detriment to representation learning. Rather, the opposite is the case [16].



$t_{\max}$ in seconds	0	2	3	4	5	8	10
K400 acc in %	60.6	65.2	65.7	65.8	65.8	65.6	65.8

(a) Dataset: **K400**, 200 epochs training.

$t_{\max}$ in seconds	4	8	16	32	60
K400 acc in %	62.7	63.1	63.1	63.9	64.1

(b) Dataset: **IG-Curated-1M**, 50 epochs training.

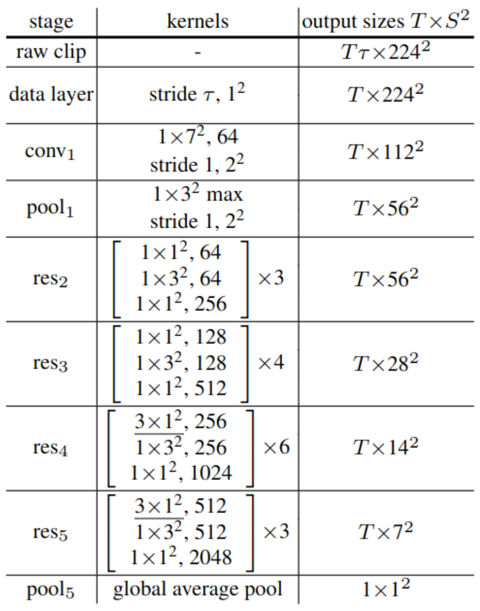
$t_{\max}$ in seconds	12s	24	36	48	600
K400 acc in %	59.3	59.2	59.9	59.6	58.9

(c) Dataset: **IG-Uncurated-1M**, 50 epochs training.

Figure 2.19: Timespan between clips [16]

In their imaged-based framework and our upcoming experiments, BYOL, SimCLR, SwAV and MoCo all defaulted to using the ResNet architecture as baseline neural encoder, particularly the ResNet-50 variant. After global average pooling, the representation vector size of a ResNet-50 model is 2048 dimensions. In these video-based iterations of BYOL, SimCLR, SwAV and MoCo, a backbone encoder also has to handle the additional temporal dimension. Feichtenhofer et al. [16] utilize the Slow-only architecture with a depth of 50 layers as baseline encoder for the new video objective. In Figure 2.20, the architecture is summarized. The Slow-only network has this specific name because the architecture consists of the Slow pathway from the SlowFast network described in section 2.2. Essentially, the Slow-only architecture could be described as a 3D ResNet model. As mentioned back in that section, temporal convolutions are only starting to be introduced in the fourth and fifth stage of the residual network structure. After global average pooling, the resulting vector has a size of 2048 dimensions. The output shape for the

Slow-only network and image-based ResNet-50 network is identical. Even though a video is normally described by a lot more scalars than an image, the visual representation of the video has the same number of dimensions as the one of an image, considering this setup. What this allows, is a simple transition with minimal modifications between the objectives since the representation shape is the same. Furthermore, all these methods back in the image domain use a projection head on top of the backbone encoder to project the representations to a lower embedding space. Also here, a projection head is placed on top of the Slow-only backbone for each different framework [16].



stage	kernels	output sizes $T\times S^2$
raw clip	-	$T\tau\times 224^2$
data layer	stride τ, 1^2	$T\times 224^2$
conv$_1$	1×7^2, 64 stride 1, 2^2	$T\times 112^2$
pool$_1$	1×3^2 max stride 1, 2^2	$T\times 56^2$
res$_2$	$\begin{bmatrix} 1\times 1^2, 64 \\ 1\times 3^2, 64 \\ 1\times 1^2, 256 \end{bmatrix}\times 3$	$T\times 56^2$
res$_3$	$\begin{bmatrix} 1\times 1^2, 128 \\ 1\times 3^2, 128 \\ 1\times 1^2, 512 \end{bmatrix}\times 4$	$T\times 28^2$
res$_4$	$\begin{bmatrix} 3\times 1^2, 256 \\ 1\times 3^2, 256 \\ 1\times 1^2, 1024 \end{bmatrix}\times 6$	$T\times 14^2$
res$_5$	$\begin{bmatrix} 3\times 1^2, 512 \\ 1\times 3^2, 512 \\ 1\times 1^2, 2048 \end{bmatrix}\times 3$	$T\times 7^2$
pool$_5$	global average pool	1×1^2

Figure 2.20: Summary of Slow-only architecture [16]

All in all, the adaption of popular image-based self-supervised representation learning frameworks to the spatiotemporal domain by using an objective to learn temporally-persistent representations has worked remarkably, as seen in Table 2.4. On three benchmarks, a version of video-based SimCLR, SwAV, BYOL or MoCo is able to outperform their supervised counterpart [16]. On the AVA dataset [57] for short-term action detection BYOL is able to outperform, on the Charades dataset [58] for long-term action classification MoCo is able to outperform and on the Something-Something v2 dataset [59] all four methods outperform the supervised counterpart [16]. Seemingly, the four methods can perform quite differently depending on the downstream task which shows that every methods has its own strength and weaknesses.

To have similar frameworks in image and video domain, will lend itself to compelling experiments. It enables us to compare the phenomenon of dimensional collapse in both mediums. More so, it allows to inspect, what impact the framework, medium or encoder architecture has on dimensional collapse.

2.4.2 DPC

DPC, Dense Predictive Coding [17], is a self-supervised generative approach for learning video representations. Predicting the next frames in a sequence of past frames based on a shared history allows for an intuitive supervision signal without the use of labelling. Nonetheless, pixel-level generation of future frames is computationally expensive but might also not be quite necessary [10]. Here, the objective is only to learn a representation of the video for downstream tasks like, for example, action recognition and not pixel-wise visualizations. Generating and contrasting pixel-level generated frame appearances in all their detail (e.g. shadows, illumination etc.) is going past the essential scope. More so, it is important to resolve uncertainty in future predictions [17]. The future is not deterministic, and therefore, a effective representation needs to learn the shared semantics of possible future states to most accurately represent the inspected video and its future [17].

Figure 2.21 shows the structure of Dense Predictive Coding in great detail. To initially train a framework like this, a video x is split into eight blocks. The first five blocks manage the

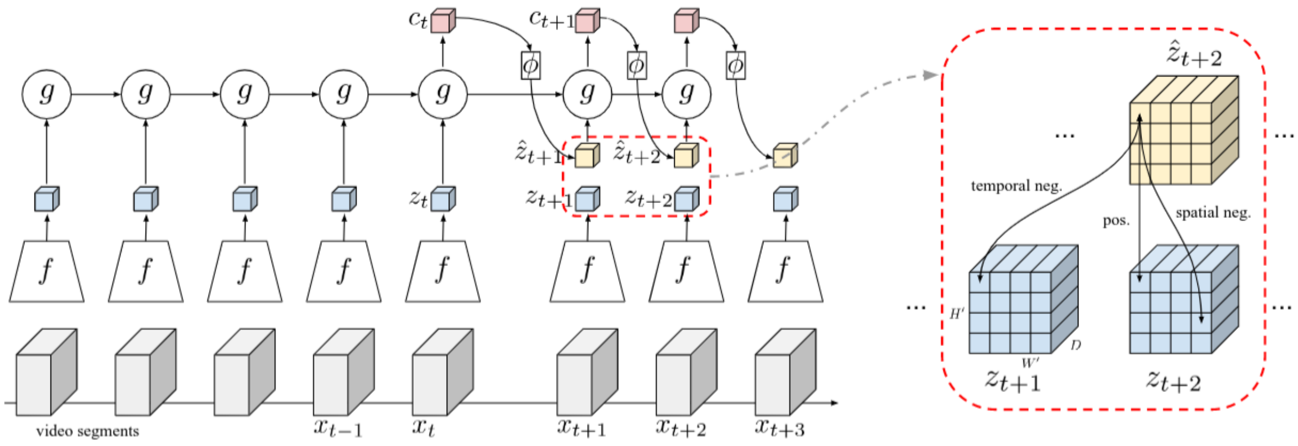


Figure 2.21: Left: Framework of DPC. Right: Illustration of comparisons for contrastive loss [17].

context, while the last three blocks compare against the predictions. All blocks are sampled from the video sequentially. Every block contains five frames, which were sampled every third frame. As well, the video blocks are augmented (crop and flip consistently, random grey and color jitter frame-wise) to prevent the network from learning trivial solutions from low-level flow information [17].

In the following setup, illustrated in Figure 2.21, the local representations, $z_t = f(x_t)$, of every block are computed. In a later experiment, the encoder $f(.)$ is a 3D ResNet-34 network similar as described in [60]. Dense Predictive Coding mostly owes his name to how the representations are processed. In many self-supervised frameworks, the output of an encoder is pooled into a one dimensional vector form before the loss is computed. DPC, on the other hand, works with the volumetric representations pre-pooling. The output z_t of the 3D ResNet-34 encoder f after processing video block x_t has a dimensional shape of $4 \times 4 \times 256$. For our consideration, input x_t, describing a singular block, had a shape of $5 \times 224 \times 224 \times 3$. After computing the local representations for every of the five input segments, their views are being aggregated to create one principal context representation $c_t = g(z_1, z_2, ..., z_t)$. As weak aggregation function $g(.)$, a one-layer Convolutional Gated Recurrent Unit is used to propagate features along the temporal axis [17]. After aggregation over the temporal axis, the output of function g shares the same output shape as a single local representation (i.e. $4 \times 4 \times 256$). Furthermore, c_t is also the representation of video x later used for downstream tasks, and therefore, the goal of the optimization [17].

The context representation sets up the predictions into the future. $\phi(.)$ is the prediction encoder which takes as input c_t and predicts the possible next latent representation at the following time step $t+1$. The prediction function ϕ is a two-layer perceptron that consists of two convolutional layers with one ReLU activation in between. But the predicted latent representation of clip x_{t+1} is not the only representation, which is necessary to foresee. The newly predicted representation $\hat{z}_{t+1}$ is fed into the aggregation function alongside the preceding representations $z_1, ..., z_t$, to create an updated context representation and to project the latent representation of a more distant time step $t+2$ (i.e. $\hat{z}_{t+2} = \phi(c_{t+1}) = \phi(g(z_1, z_2, ..., z_t, \hat{z}_{t+1}))$) [17]. This procedure repeats one last time for clip x_{t+3}. Lastly, predicting the latent representation of such a clip, that is not in the immediate neighbourhood of x_t, urges the network to learn signals of the data which encompass the whole video. *With increasing difficulty and uncertainty of the task, a more abstract and semantic understanding is required* [17].

What remains, is the optimization of the framework, and therefore, the optimization of the predictions. So far, spatiotemporal feature maps acting as latent representations have been predicted for time steps $t + 1$ to $t + 3$, as seen in Figure 2.21. This responds to $\hat{z}_{t+1}$, $\hat{z}_{t+2}$ and $\hat{z}_{t+3}$. The yet withheld video clips x_{t+1}, x_{t+2} and x_{t+3} are responsible for producing the ground-truth representations z_{t+1}, z_{t+2} and z_{t+3} using the shared encoder f. These predicted and ground-truth pairs share the same dimensionality of $4^2 \times 256$. So, the representations have not been pooled to reduce dimensionality. Densely encoded spatial and temporal features are not pooled and remain in their encoded structure.

Similar to SimCLR and MoCo, DPC decided on using a Noise Contrastive Estimation (NCE) [61] as loss function. Figure 2.21 shows how such contrasting can look like and how it makes use of the unpooled 3D feature map structure. As previously mentioned, each latent representation retains a shape of $4^2 \times 256$. This allows in contrast to differentiate between temporal and spatial samples. Furthermore, the contrastive loss considers feature vectors of shape $1^2 \times 256$. Therefore, every feature map contains 16 feature vectors, which can be compared to other feature vectors of different feature maps. The feature vector $z_{i,k}$ describes a feature vector at the i-th time step at spatial position k with $k \in \{(1,1),(1,2),...,(4,4)\}$. This approach never requires to predict an exact future but to choose the best one among all these negative distractions [17]. SimCLR and MoCo already taught us the importance of choosing as many negative samples as possible but also the right ones. Figure 2.21 illustrates that when comparing predicted latent representation $\hat{z}_{t+2}$ with its ground-truths, temporal negatives are feature vectors at the right spatial location but wrong time step while spatial negatives consider the right temporal feature map but wrong spatial feature vector. More so, during training the whole mini-batch could be considered, this would allow for more negative samples. These negative samples would be feature vectors from a whole different video. All in all, the contrastive loss for DPC is be constructed as follows [17]:

$$\mathcal{L} = -\sum_{i,k} \left[\log \frac{exp(\hat{z}_{i,k}^T \cdot z_{i,k})}{\sum_{j,m} exp(\hat{z}_{i,k}^T \cdot z_{j,m})} \right].$$

In the numerator, the similarity between the positive pairs is computed, that means pairs at the same temporal and spatial location. The denominator is additionally made up of all the negative pairs. This cross-entropy loss encourages the positive pair to have higher similarity than any negative pair [17]. Similar to the image-based contrastive methods, positive future states attract while impossible future states repulse.

In Table 2.4, the performance of Dense Predictive Coding is shown on linear evaluation. It is being outperformed by the other listed methods, even though the smaller backbone has to be considered (e.g. Slow-only contains 50 layers). Still, DPC is a very fascinating method which utilizes the intrinsic nature of videos. A generative method uses a slightly different approach and idea than previous methods like SimCLR, SwAV, MoCo or BYOL. But that should also be the reason, why it is appealing to explore such method for dimensional collapse and compare.

2.4.3 XDC

In this thesis, when transitioning from image approaches to video approaches, the addition of the temporal dimension has often been highlighted and its relevance for video representation learning. Time certainly is the obvious new addition when shifting the medium from images to videos but it is also not the only one. Videos are almost always accompanied by sound. Audio offers an additional modality alongside to the visual information of the video. This modality has not yet been mentioned or utilized in the previous summarized methods. When watching a film, for example, video and audio track of the film tell a shared story and are therefore highly correlated. Still, the information in a scene is expressed differently by audio and video. Sometimes, the provided information is presented complementary, simultaneously or exclusively by the modalities. There exists a intrinsic difference between both modalities, besides their strong correlation, that lends itself to be an interesting supervision signal. Hearing someone cough does not require a visual clue for us to image what action took place. Reversely, seeing someone cough gives a good guess as to what it must have sounded like [18]. Alwassel et al. [18] made themselves this connection to use, to develop a self-supervised representation learning method based on the modalities of video and audio.

XDC, Cross-Modal Deep Clustering [18], is a cross-modal self-supervised approach which leverages the correlation of audio and video for cross-prediction. The main idea behind XDC is to use individual clustering of each modality to obtain their cluster assignment as pseudo-label for classification of the other modality. The general clustering approach is inspired by the work on DeepCluster by Caron et al. [62]. DeepCluster is a single-modality image clustering self-supervised framework that works in an iterative two step training process. First, images of a dataset are encoded and those features are clustered. In a second step, the cluster assignments are used as feature pseudo-labels to refine the image representations [18]. Interestingly enough, DeepCluster is the predecessor to the SwAV framework. While DeepCluster utilizes an offline step to compute the feature clusters, SwAV does this online. Alwassel et al. [18] adapt the DeepCluster methodology to video learning. Before arriving at Cross-Modal Deep Clustering (XDC), multiple different approaches are attempted to learn from clustering two modalities.

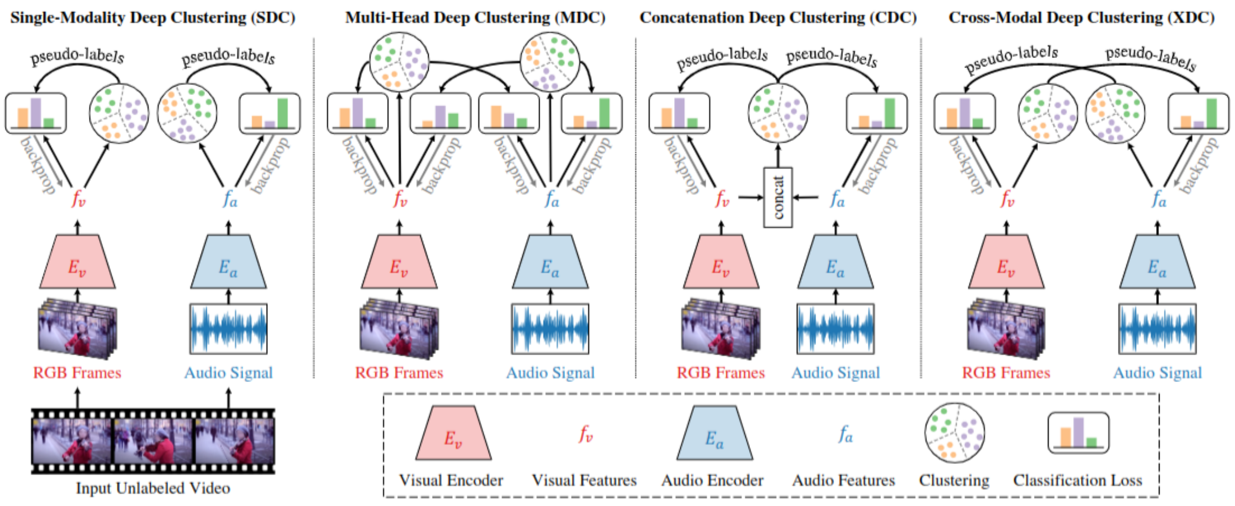


Figure 2.22: Overview of the four clustering frameworks [18]

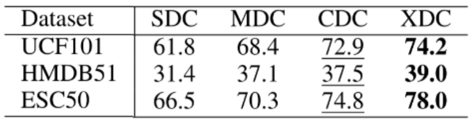
Figure 2.22 displays four various self-supervised clustering frameworks that utilize audio and video signals for training. Going from left to right, the first framework, Single-Modality Deep Clustering (SDC), simply exemplifies the DeepCluster structure transitioned to the video and audio domain. Each modality is trained separately. To obtain the feature representations, f_v for video and f_a for audio, that are used for clustering and classifying, two backbone encoders, E_v and E_a, are operated to encode video and audio signal, respectively. Both encoder architectures employed in their experiments are known to us. To encode videos, a ResNet(2+1)D-18 model is presented, as described in section 2.2.1. To encode audio signals, a ResNet-18 model, previously used on images, is presented, as described in section 2.1.2. To reiterate, each model is trained by its own modality. In two separate steps, features are clustered and cluster assignments are used as pseudo-labels for training [18].

The second framework, Multi-Head Deep Clustering (MDC), is the first cross-modal framework. Each encoder is extended by a second classification head. This additional head is supervised by the other modality. More so, this means that modality-specific pseudo-labels are used for both networks to train [18].

The third framework, Concatenation Deep Clustering (CDC), combines the representation of both modalities. Before clustering, representations f_v and f_a are l_2-normalized separately and afterwards concatenated. After clustering, these mixed cluster assignments shared between both modalities are used to update the weights of both encoders E_v and E_a [18].

Finally, the main attraction, Cross-Modal Deep Clustering (XDC), is pictured on the far right side of Figure 2.22. The difference to the other cluster frameworks is simple. In XDC, every encoder is exclusively trained on the pseudo-labels of the other modality. Cluster assignments formed by the audio features f_a are used for the video encoder E_v and vice versa. The supervision signal is entirely based on the other modality and its characteristics [18].

Figure 2.23 compares all the just described representation learning frameworks. Models are pretrained using Kinetics-400 as dataset and finetuned on either UCF-101 [63], HMDB51 [64] or ESC50 [65]. UCF-101 and HMDB51 are both action recognition datasets while ESC50 is a sound classification dataset. The top-1 accuracy on split-1 is reported for each dataset [18]. The trend is similar for all three evaluated datasets. XDC performs best, CDC performs second-best, MDC performs third-best and SDC performs the worst. Thus, it can be stated that all three cross-modal approaches (i.e. XDC, CDC and MDC) exceeded the single-modality approach (i.e. SDC). A sign for the potential of complementing audio and video signals for self-supervised video representation learning. Furthermore, XDC's learned representations generalized the best out of the four approaches. Supervision purely based on the other modality seems to behave superior in comparison to a mix of both. Alwassel et al. [18] hypothesize that methods, which cluster based on both modalities, group samples only together if they are similar in accordance to both of the two modalities. On the other hand, XDC groups samples together, when they are similar to one of the two modalities [18].



Dataset	SDC	MDC	CDC	XDC
UCF101	61.8	68.4	72.9	**74.2**
HMDB51	31.4	37.1	37.5	**39.0**
ESC50	66.5	70.3	74.8	**78.0**

Figure 2.23: Performance comparison of the four clustering frameworks [18]

Besides the choice of framework, the choice regarding the number of clusters is also important. For clustering, XDC uses the classic k-means clustering algorithm. Alwassel et al. [18] found that the number of cluster centers k is not sensitive to the number of semantic labels in the downstream dataset (e.g. $k = 101$ for UCF-101). More so, the best value for k tends to get larger as the size of the pretraining dataset increases. In general, XDC's performance improves across all three downstream datasets (i.e. UCF-101, HMDB51 and ESC50) as the size of the pretraining dataset increases [18].

In Table 2.4, XDC is shown to achieve a 95.5% accuracy on the UCF-101 action recognition dataset, when pretrained on the IG-Curated dataset with 65 million data points. This accuracy considers finetuning, re-training the video encoder on the last clustering assignment and using 32-frame clips [18]. However, this pretraining and evaluation outperforms the supervised counterpart pretrained on Kinetics-400 (94.2%). Even though both models are pretrained on differently large datasets, according to the authors of the paper, this is the first time a self-supervised framework has outperformed a large-scale full-supervision pretraining for action recognition, considering the same architecture [18].

Lastly, many self-supervised approaches suffer from collapsing to trivial solutions. DeepCluster, for example, has to utilize workarounds to avoid empty clusters or few-class predictions [62]. XDC, on the other hand, has never encountered such issues in training. Cross-modal clustering methods seem to be more robust against trivial solutions because they learn to classify on one modality and receive the pseudo-labels from the other modality [18]. In the exploration chapter, we will explore if this also holds true for dimensional collapse.

2.5 Dimensional Collapse

2.5.1 First Discovery

The phenomenon of dimensional collapse in self-supervised representation learning was first illustrated by Hua et al. (2021) [19]. Previously, only the problem of complete collapse was considered. The general idea behind many state-of-the-art self-supervised methods is to learn representations that are robust to augmentations. A framework which many of these methods (e.g. SimCLR, SwAV, MoCo, BYOL etc.) are built on is shown in Figure 2.24. An image x is randomly augmented into two different views, x_1 and x_2. Now, the encoder is supposed to learn meaningful representations of these views that maximize their common semantic content. These representations have to minimize a distance metric, and to do so, have to learn to be invariant to the applied augmentations. Unfortunately, the global minimum of such a objective does not constitute meaningful representations. To reach the global minimum, the encoder can simply learn to treat all dimensions of the representation vector as constants. Regardless of the inputs, this would every time account for a distance of zero. Therefore, the collapse over all dimensions is named complete collapse. How those self-supervised approaches (e.g. SimCLR, SwAV, MoCo, BYOL etc.) differ from the base framework, shown in 2.24, is often dependent on how they address and avoid this complete collapse. SimCLR and MoCo use negative samples, SwAV uses clustering and BYOL uses prediction plus momentum encoder. Since all those listed methods have been successful in avoiding complete collapse and are able to create effective representations, the study on further collapse patterns has been ignored [19].

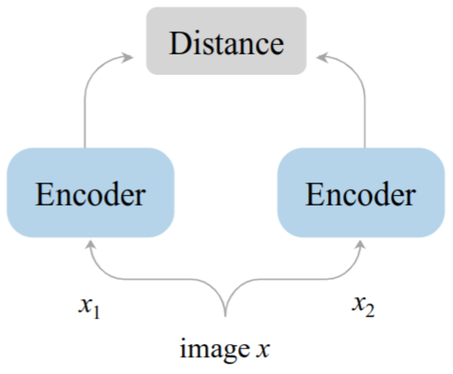


Figure 2.24: Base framework for SSL [19]

Hua et al. [19] revisit the collapse issue, and first, verify the existence of complete collapse. This leads in turn as a result to their discovery of dimensional collapse. The experiment setup is minimal and based on the framework shown in Figure 2.24. The encoder is a ResNet-18 model with an additional MLP as projection head. Therefore, the collapse issues addressed here concern the embedding space (i.e. output of the projector) and not the representations directly (i.e. outputs of backbone encoder used for transfer tasks). More so, the MLP consists of two hidden layers of 64 neurons each. In between each hidden layer, a ReLU activation and Batch Normalization (BN) is inserted. The final output of the projector is of two dimensions for visualization purposes. The dataset used for training and validation is the CIFAR-10 [23] image classification dataset. For data augmentations, several common augmentations are adopted like random scaling and cropping, random horizontal flipping, color distortion, color dropping and random gaussian blur [19]. Lastly, mean squared error (i.e. $l(z_1,z_2) = \|z_1 - z_2\|_2^2$ for embedding vectors z_1 and z_2) is used as distance function [19].

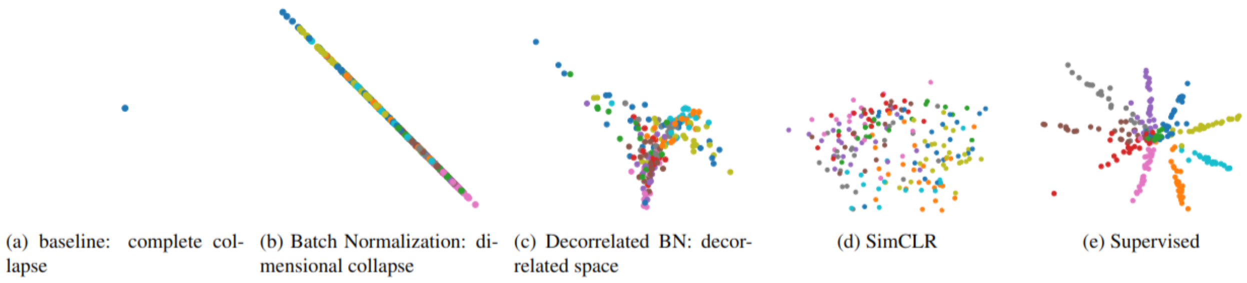

(a) baseline: complete collapse (b) Batch Normalization: dimensional collapse (c) Decorrelated BN: decorrelated space (d) SimCLR (e) Supervised

Figure 2.25: Visualizations on the different experiments about collapse [19]

Figure 2.25 displays the visualizations of different experiments related to collapse, starting with the verification of complete collapse. Simply training the described baseline results in complete collapse, as visualized in (a) of Figure 2.25. In the embedding space, every single embedding holds the same value for both dimensions. The space collapses to a single point with no variance (i.e. complete collapse). Furthermore, liner evaluation with CIFAR-10 on the learned representations only achieves 28.56% [19].

To combat these vanishing variances, a BN layer is appended to projection output layer. BN in the embedding space can be a way to mitigate complete collapse by standardizing variance [19]. Furthermore, this specific BN layer has no trainable parameters and the numerical stability value ϵ is set to zero. Visualization (b) of Figure 2.25 shows the resulting two-dimensional space. Here, Hua et al. [19] detect the additional collapse pattern of dimensional collapse. The two-dimensional embedding space is collapsed into a one-dimensional subspace (i.e. a line). Both dimensions are heavily correlated. Thus, Hua et al. [19] associate strong correlation between axes with dimensional collapse. More so, this offers strong motivation to utilize feature decorrelation (i.e. standardizing the covariance matrix of the embeddings) to mitigate dimensional collapse [19]. Self-supervised approaches which we have already seen utilizing feature decorrelation are BarlowTwins and VICReg. Lastly, the baseline plus final BN layer achieves a linear evaluation of 69.52% [19].

Due to dimensional collapse, the potential of the whole embedding space can not be taken advantage of. To attempt to standardize the covariance matrix, Hua et al. [19] alter the last BN layer into a Decorrelated Batch Normalization layer (DBN) [66]. In a DBN layer, a batch of embedding vectors is split into multiple groups of a selected size and ZCA whitening [67] is applied individually to each group [19]. After whitening, the covariance matrix of each group corresponds to the identity matrix. The resulting visualization is pictured in (c) of Figure 2.25. This shows that feature decorrelation can alleviate dimensional collapse and that dimensional collapse is associated with strong correlations between axes [19]. Also, the accuracy on linear evaluation increased from 69.52% to 72.45% [19].

Additionally, the two-dimensional embedding space of SimCLR and the representation space of supervised training are displayed in (d) and (e) of Figure 2.25, respectively. Both do not seem to strongly suffer from any collapse pattern.

	acc. (%)	std.	corr.	loss
vanilla	35.44	**0.00**	0.13	0.00
BN	70.85	1.00	**0.99**	7.01
DBN	84.41	1.00	0.00	39.04

Figure 2.26: Comparison of higher dimensional embedding space [19]

For visualization purposes, the embedding space was restricted to only two dimensions. However, Hua et al. [19] experience that the findings also stay true for high dimensional embedding spaces. In Figure 2.26, the baseline framework, BN altered framework and DBN altered framework are compared when equipped with a wider projection head. Now, the output dimensionality of the embeddings is 128. Furthermore, the two hidden layers both contain 128 neurons. Regarding accuracy on linear evaluation, while still large, the gap between BN and baseline framework shrinks a bit. On the other hand, mitigating dimensional collapse seems to bear more fruit. The gap between BN framework and DBN framework grows by more than 10%. But what is more expressive, are the standard deviation (std.) and correlation strength (corr.) values for all three models. The average standard deviation is computed over the 128 dimensions of the embeddings, while the average correlation strength measures the average of the absolute values of non-diagonal entries of the correlation matrix of the embeddings [19]. For the baseline framework, the standard variance remains zero and for the BN framework, the correlation strength almost equals one. This reaffirms the assumption that vanishing variances is a sign of complete collapse and that strong correlations of the dimensions is a sign of dimensional collapse [19].

2.5.2 Contrastive Learning and Projectors

While Hua et al. [19] were the first to study dimensional collapse, they are not the last. Jing et al. (2022) [1] analyse dimensional collapse in the setting of contrastive learning, and furthermore, investigate the important role projection heads play in relation to dimensional collapse.

In contrastive methods (e.g. SimCLR, MoCo etc.), complete collapse is prevented by using negative samples, additional to the positive pair, in a contrastive loss (e.g. InfoNCE). This loss maximizes similarity between the positive pair and pushes embeddings away from the negative samples. Previously, Hua et al. [19] showed that dimensional collapse can occur in a relatively unconstrained framework. Here, it is shown that even contrastive methods, which introduced the additional load of negative samples to avoid complete collapse, can suffer from dimensional collapse [1].

To measure and illustrate the occurrence of dimensional collapse in contrastive learning, Jing et al. [1] train a SimCLR model with a two-layer projector on ImageNet for 100 epochs. After training, the model is evaluated on the validation set and the computed embedding vectors z_1,z_2,...,z_N are stored. The dimensionality of the embeddings is 128. To determine the dimensional collapse, first the covariance matrix of those embeddings is computed [1]:

$$C = \frac{1}{N}\sum_{i=1}^{N}(z_i - \bar{z})(z_i - \bar{z})^T \quad \text{where} \qquad \bar{z} = \sum_{i=1}^{N}\frac{z_i}{N}$$

Secondly, the singular value decomposition is applied on this covariance matrix C to obtain their singular values. The amount of non-zero singular values indicate the rank of the matrix [68]. Also, for further inspection, the logarithm is taken of the singular values. This methodology is only briefly touched upon here since it will be discussed more in the upcoming chapter.

The left plot of Figure 2.27 presents the singular values of the covariance matrix in logarithmic scale and in sorted order. Inspecting the logarithmic scale, it is visible how after roughly 95 dimensions, a number of singular values suddenly collapse. These singular values represent collapsed dimensions with a singular value of zero [1]. The collapsed dimensions add no additional information content to the embeddings. The embedding vectors occupy a lower-dimensional subspace instead of making use of the available 128 dimensions [1].

After establishing the possibility of rank deficient embedding spaces in contrastive approaches, Jing et al. [1] determine two possible origins for dimensional collapse in contrastive learning. Their considered theoretical setup is a single linear layer (i.e. $z = Wx$ with weight matrix W and input vector x) and an InfoNCE loss. The first resulting corollary of their experiments, which they assume, is that *with strong augmentation, the embedding space covariance matrix becomes low-rank*, leading to dimensional collapse [1]. Previously, when talking about the SimCLR approach, it was said that self-supervised contrastive methods benefit from stronger augmentations than their supervised counterpart and that they are preferable for good training [10]. According to Jing et al. [1] and considering this setup, strong augmentations will become a detriment, if the information content provided by the data augmentations overshadows the information content provided by the data distribution. For their second theoretical experiment, two weight matrices are used (i.e. $z = W_2W_1x$). Their second corollary follows that *with small augmentation and over-parametrized linear networks, the embedding space covariance matrix becomes low-rank* [1]. Therefore, their overall findings are that either strong augmentations or over-parametrized linear networks can lead to dimensional collapse in a linear contrastive setting. Furthermore, Jing et al. [1] show empirically that their theory can also be extended to multilayer networks and non-linear settings [1].

So far, dimensional collapse has only been considered in the embedding space (i.e. the vectors used for loss) but not in the representation space (i.e. the vectors used for further downstream tasks). In many self-supervised approaches, projection heads have shown to significantly boost

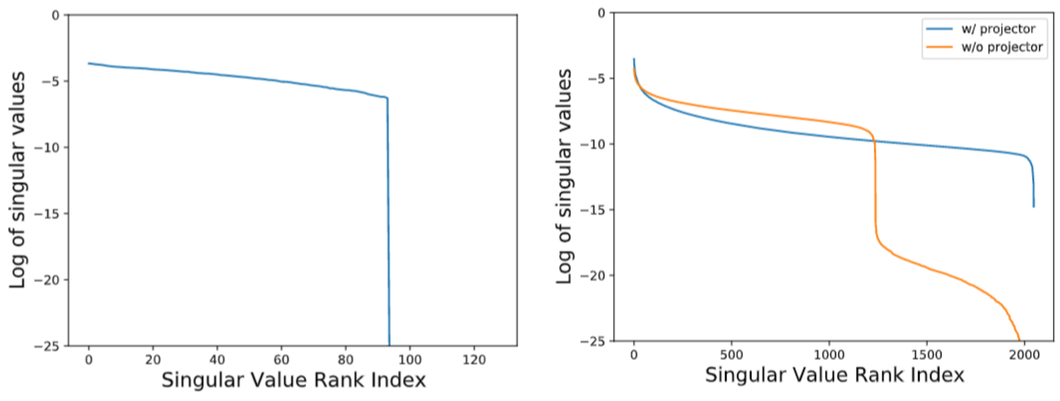


Figure 2.27: Left: Logarithmic singular value spectrum of SimCLR embeddings. **Right**: Spectrum of SimCLR representations trained with and without projector. [1]

performance [10]. In a following experiment, SimCLR is trained once with and once without a projection head. Afterwards, the singular value spectrum is plotted for both. The right plot of Figure 2.27 pictures those spectra. Dimensional collapse in the representation space only occurs for the model trained without projection head. Consequently, the added projector prevents the dimensional collapse in the other case [1]. So, a projector does not only improve training performance but also seems to prevent collapse in the obtained representations. More so, Jing et al. [1] claim, based on their findings, that the weight matrix of a linear projector in contrastive learning only needs to be diagonal and low-rank to prevent dimensional collapse. But additionally, they also state that their *theory is not able to fully explain why a nonlinear projector is able to prevent dimensional collapse* [1].

2.5.3 Non-Contrastive Learning and Predictors

Contrastive methods can suffer from dimensional collapse. What about non-contrastive methods like BYOL and SimSiam [69]? SimSiam, simple Siamese network, is an approach similar to BYOL that does not use a momentum encoder. The online and target encoder share the same weights and the gradient is only being propagated through the online network. Just like BYOL complete collapse is avoided but the reason as to why is not clear [69].

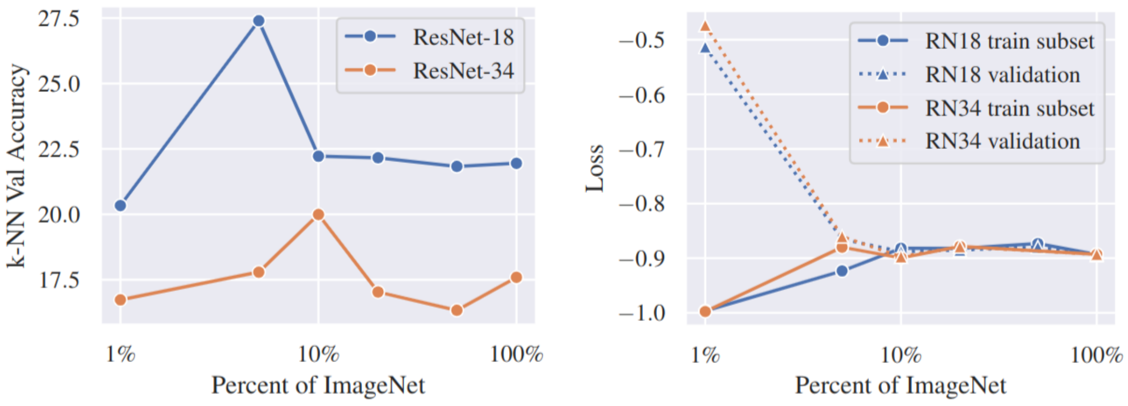


Figure 2.28: SimSiam performance on ResNet-18 and 34 on different subsets of ImageNet [20]

Li et al. [20] study collapse patterns in SimSiam and find that SimSiam is very sensitive to model capacity and dataset size. In an experimental setup, SimSiam is trained both on a ResNet-18 backbone and a larger ResNet-34 backbone. To evaluate performance over datasets of different sizes, increasingly large subsets of ImageNet-1k are used for training. As performance metric, the K-nearest neighbors algorithm is used for evaluation. Figure 2.28 plots the results for both encoders. The SimSiam with the ResNet-18 and ResNet-34 backbone experiences peaks at different subsets of ImageNet. The model using ResNet-18 peaks in accuracy around 5%, and the model using ResNet-34 peaks at around 10%. This follows the hypothesis of Li et al. [20], regarding sensitivity of SimSiam, since more data only helps performance up to a certain threshold. Afterwards, both models' performance drops significantly. More so, those performance peaks are not visible in the loss metric of the models, be it training or validation loss [20].

In a next step, Li et al. [20] analyze all the various models trained on the ResNet-18 backbone for dimensional collapse. Their methodology for presenting dimensional collapse is different to the one of Jing et al. [1], but also similar at the same time, since both ultimately describe the rank of the spanned representation space. Here, Principal Component Analysis (PCA) is performed on the representation matrix. This matrix consists of the d-dimensional representations for each considered image. Regarding ResNet-18, the dimensionality of the representations is 512. PCA is used to obtain their singular values $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_d$. For presentation and to measure the degree of collapse, the cumulative explained variance fo the singular values is computed [20]:

$$(\text{Cumulative explained variance})_j = \frac{\sum_{i=1}^{j} \sigma_i}{\sum_{k=1}^{d} \sigma_k}.$$

Figure 2.29 plots these cumulative explained variance values for models trained on 1%, 5%, 10%, 20%, 50% and 100% of ImageNet. Previously, the performance peak for k-NN evaluation on ResNet-18 was measured at 5%. In Figure 2.29, the quicker a curve rises from 0 to 1, the more it has collapsed. Specifically, the more variance can already be explained before having utilized all dimensions. For instance, the model using 1% of ImageNet shows no collapse. The influence of each dimension is similar. On the other hand, the 10%, 20%, 50% and 100% model exhibit a larger degree of collapse while the 5% model collapses only to a small degree [20]. This kind of visualization allows to better differentiate between degrees of collapse. Speaking only of dimensional collapse might sometimes not be specific enough in terms of the extent. *Collapse exists on a spectrum* [20] and such plots emphasize that. For example, the 5% ResNet-18 model has the highest k-NN accuracy but collapses slightly more than the 1% model which exhibits no collapse. Li et al. [20] speculate that the better performance of the 5% model has to do with the significantly lower SimSiam loss compared to the 1% model, as seen in Figure 2.28. A tradeoff seems to exist between the SimSiam loss and dimensional collapse of a model.

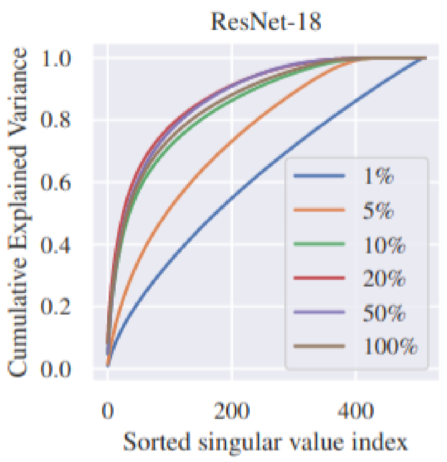


Figure 2.29: Cumulative explained variance of singular values [20]

In a third experiment, this tradeoff between loss and collapse is utilized to test whether it is expressive enough to solely predict model performance on a downstream task. For prediction, a simple linear model is fitted to predict the validation linear probing accuracy of a trained ResNet model based on its SimSiam validation loss and an additional dimensional collapse metric. This specific collapse metric is the area under the curve of the cumulative explained variance of the singular values [20]:

$$\text{AUC} = \frac{\frac{1}{d}\sum_{i=1}^{d}\sum_{j=1}^{i} \sigma_j}{\sum_{k=1}^{d} \sigma_k}$$

The values for AUC range from 0.5 to 1 with 0.5 meaning no collapse and larger values signifying more collapse. Furthermore, the linear prediction model is fitted with 22 trained

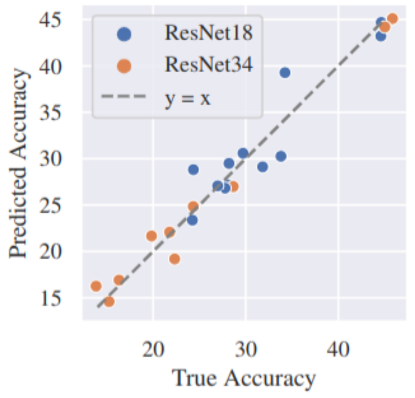


Figure 2.30: Validation accuracy predicted with loss and collapse [20]

SimSiam models, which either consist of a ResNet-18 or ResNet-34 backbone encoder. Figure 2.30 displays how well the linear model is able to predict the validation accuracy. The R^2 score of the linear model using both loss and collapse for prediction equals 0.95. A prediction only based on either AUC or validation loss seems to not be sufficient [20]. Nonetheless, this allows us to judge representation quality without relying on labels. From the self-supervised pretraining til the evaluation of representation quality, no labels would be required [20].

In a fourth experiment, the performance of SimSiam on smaller models like ResNet-18 is improved drastically by considering their dimensional collapse. On larger models like ResNet-50, SimSiam is able to compete with other state-of-the-art approaches like MoCo or BYOL, but when switching to smaller encoders SimSiam falls short. Li et al. [20] record a linear probing accuracy of 53.8% and 47.6% for MoCo-v3 and BYOL, respectively, using a ResNet-18 encoder on ImageNet, while SimSiam only achieves a 30% accuracy. An important difference between those three methods is that SimSiam experiences a larger degree of dimensional collapse, when trained on the whole of ImageNet with a ResNet-18 backbone.

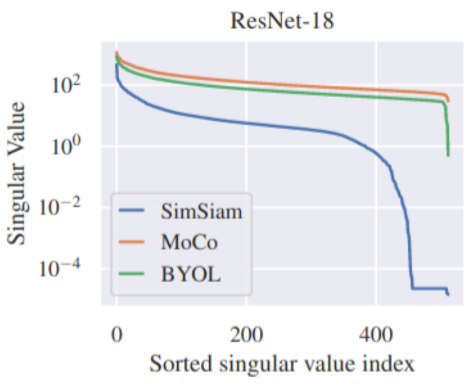


Figure 2.31: Singular value spectrum of MoCo-v3 and BYOL [20]

Figure 2.31 displays the singular value spectrum of the representations, like described in [1]. MoCo and BYOL do not suffer from collapse while SimSiam does to a degree. Since the main difference between BYOL and SimSiam is a momentum encoder, its relevance can not be ignored, especially when considering smaller encoders [20]. More so, there was a point at which SimSiam also did not suffer strongly from dimensional collapse. This was the case when only a subset of ImageNet was considered for training (e.g. 1%-5%). Therefore, Li et al. [20] examine continual learning to be able to efficiently train SimSiam on a smaller ResNet-18 model with ImageNet. Continual learning uses single passes. In a single pass, the dataset is divided into many smaller subsets. The model is trained on one subset, this subset is discarded and the next subset arrives [20]. Since SimSiam is sensitive to the model capacity and dataset size, this type of training allows to avoid dimensional collapse by matching the model capacity with the dataset size [20]. Using single pass, the linear probing accuracy on ImageNet increases by 14.5% to a total of 44.5%. Additionally, using a hybrid form of single pass til epoch 40 and standard training with the full dataset afterwards, increases the accuracy further to 48.3%, which is even a higher accuracy than the 47.3% for BYOL [20].

These experiments with SimSiam about dataset size and model capacity reveal what relevance the study of dimensional collapse has. It ranges from the prediction of representation quality to the investigation of more fitting training procedures. So far, all these studies have focused on self-supervised image representation learning. We will go a step further, and besides image representations, also explore, if common video representation learning methods suffer from dimensional collapse.

3 Methodology

3.1 Model Origins and Limitations

This thesis attempts to analyze many common representation learning methods in computer vision for dimensional collapse. In order to explore, we require the means to produce representation vectors of such methods. The most naive way would be to train each model on our own. Follow the directions offered in the research paper, construct the model, train the model, and afterwards, compute its representations. However, this is unfeasible for multiple reasons. First, not every research paper might present all their implementation details. This can range from optimization details to structural details. Having said that many research papers provide good documentation of their procedure. Secondly, to be able to successfully train substantial machine learning models like a SlowFast network [7] on large video datasets like Kinetics-400 [33], an abundance of Graphical Processing Units (GPU) or Tensor Processing Units (TPU) are required. Thirdly, training models of such size, takes time. For example, listed in Table 2.4, the DPC model on Kinetics-400 with a 224x224 resolution utilizing four NVIDIA P40 GPUs took the researchers about six weeks to train [53]. Time and resources which are out of scope for this thesis. Therefore, all models inspected in this thesis are obtained pretrained and are not self-trained.

Throughout the thesis, we rely on pretrained models provided by the research team behind each approach. Tables 2.1, 2.2, 2.3 and 2.4 reference the origin for those pretrained models. Furthermore, this thesis is a pure evaluation thesis. Pretrained models are evaluated for their representations, and in turn, those representations are used to examine the degree of dimensional collapse for their respective learning method. For evaluation, a NVIDIA GeForce GTX 1070 was used.

Not having to train, and instead, use pretrained models comes with a lot of benefits but also some limitations. First of all, using pretrained models is a huge time saver. As previously mentioned, training these large models requires a lot of time and resources. Secondly, we can be assured that models are correctly implemented. On the other hand, it comes with restrictions. Naturally, we can only use those pretrained models which are provided. Some research papers provide a repository with a wide array of pretrained model configurations like SwAV in [43], other research papers only provide a single configuration like Barlow Twins in [41] and some, especially older papers, do not provide any pretrained models or only provide outdated models for evaluation. Therefore, only research papers can be considered for which pretrained models have been made available. More so, most of these models have been highly optimized to rival the state-of-the-art. Thus, dimensional collapse will only be analyzed on well performing models. Additionally, since the models have already been trained, further ablation studies are not possible.

Lastly, evaluation does not work straight out of the box after obtaining the file of a pretrained model. Research teams use different machine learning frameworks to train their models. The most popular and most utilized one is PyTorch [70]. PyTorch is an open source machine learning framework by Meta AI. Furthermore, most state-of-the-art research papers in self-supervised learning involve researchers from Meta AI in some capacity. Therefore, PyTorch is employed the most for training and evaluation. At this point, it is easier to list the official implementations of self-supervised approaches that do not use PyTorch than to list those that do. The methods SimCLR [10] and CVRL [71] both use TensorFlow [72] to train their models. TensorFlow is an open source machine learning framework created by Google. Lastly, BYOL [13] uses JAX [73] which is a more recent machine learning framework also developed by Google. To be able to evaluate all these various models, many different frameworks need to be learned and utilized.

3.2 Datasets

Representations require a trained network model to extract features but most of all representations need data to represent. In the literature review, commonly used datasets have already been mentioned when necessary. In this section, a short overview will be presented. Datasets will be introduced which were used to train models or which will be used to create representations.

Image Datasets: ImageNet-1k [22] is prevalent in Table 2.1 and Table 2.3 as the dataset that is used to train all the listed supervised and self-supervised image models. ImageNet was released in 2009 but still finds today frequent use. As mentioned previously in the section 2.1, ImageNet kicked off the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) which encouraged substantial development in its field. The ImageNet-1k subset spans 1000 object classes, contains 1,281,167 training images, 50,000 validation images and 10,000 test images [22]. While the models, listed in the Tables above, have been trained on the training images of ImageNet-1k, we will make use of the validation split for evaluation. If not specified otherwise, we will analyse the representation space spanned by the representations of the ImageNet-1k validation set.

A situation in which we inspect an additional dataset would be to compare representation spaces of different datasets. CIFAR-10 [23] is such suitable dataset to compare against ImageNet-1k. Similar to ImageNet, CIFAR-10 is was released in 2009. CIFAR-10 consists of 10 object classes, 50,000 training images and 10,000 test images. The object classes include different types of animals and vehicles (e.g. airplane, truck, bird, horse etc.) [23]. All in all, it is a considerably smaller dataset compared to ImageNet-1k.

Video Datasets: Kinetics-400 [33] can be thought of as the parallel to ImageNet-1k in the video domain. Almost all pretrained models presented in Table 2.2 and Table 2.4 have been trained using Kinetics-400. It was released more recently in 2017 and is a video dataset for human action classification. 400 action classes are included (e.g. arm wrestling, milking cow, tai chi etc.). Kinetics-400 comprises in total 306,245 videos with an average run-time of 10 seconds. The validation set makes up roughly 20,000 of those videos. The videos are obtained from the video platform YouTube. A video is described by its URL, staring time and ending time. Based on their origins, the videos contained in Kinetics-400 vary quite a bit from each other regarding parameters like resolution and frame rate. In this thesis, we will make use of the validation set of Kinetics-400 for comparison purposes.

UCF-101 [63] is a human action video dataset from the year 2012. Kinetics-400 can be considered its successor but UCF-101 still finds regular use in research. It is comprised of 13320 human action videos which have an average clip length of 7.21 seconds. As the name gives it away, the videos are split up into 101 action classes. Same as in the Kinetics-400 dataset, the resource of the videos is YouTube. Different to Kinetics-400, the resolution and frame rate of each video is fixed to 320×240 and 25 frames per second [63]. Therefore, the storage requirement with 7.2 Gigabyte is already significantly less than the storage requirement for the Kinetics-400 validation set alone with 30.6 Gigabyte. UCF's uniformity is helpful for preprocessing the data before evaluation. Additionally, on the GPU described above, the time required for computing the representations on the UCF-101 dataset with a ResNet-50 model is quite manageable with a time of roughly 30 minutes, while it takes almost 3 hours for the Kinetics-400 validation set. Similar to ImageNet, if not stated differently, the UCF-101 dataset is used to compute the representations of video learning methods.

Lastly, we have to quickly mention the IG-Curated dataset [56] since it is used for pretraining in Table 2.4 of the XDC method. IG-Curated is another step up from Kinetics-400. It contains a total of 65 million videos. Those number of videos are almost impossible to annotate and a reason for the development of self-supervised methods. However, the videos are weakly-supervised since their origin is Instagram and hashtags are used for labels. Furthermore, the annotations are based on the classes from Kinetics [16, 56].

3.3 Preprocessing

Since we are only concerned with evaluation, no intricate data augmentations or other special transformations have to be applied to the data. But before handing the videos or images over to the network model to compute their representations, still a bit of work needs to be done. PyTorch is mostly used in this thesis for preprocessing and evaluation. Therefore, if not differently specified, following steps have been followed.

Image Preprocessing:

```python
from torchvision import datasets, transforms
# ImageNet mean and standard deviation values for normalization
normalize = transforms.Normalize(
        mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]
)
# Load and transform ImageNet validation set
dataset = datasets.ImageFolder(
        dataset_path,
        transforms.Compose(
            [
                transforms.Resize(256),
                transforms.CenterCrop(224),
                transforms.ToTensor(),
                normalize,
            ]
        )
)
loader = torch.utils.data.DataLoader(dataset, batch_size = 64)
```

For processing, the PyTorch [70] torchvision library is used which includes many in-demand image transformations and datasets. More so, this procedure follows BYOL's approach for evaluation [13]. First, the image is resized according to its short side (i.e. the smaller dimension is matched to 256 pixels and the other side is rescaled appropriately). Secondly, a center crop is taken from the resized image. Afterwards, the images is converted to a PyTorch tensor and all of its values are rescaled to the range of zero to one. As a last step, the tensor is normalized using the mean and standard deviation computed over the whole ImageNet dataset. Furthermore, these computations are done in batches instead of treating each image individually.

Video Preprocessing:

```
 1 import pytorchvideo.data
 2 from torchvision.transforms import Compose, Lambda
 3 from torchvision.transforms._transforms_video import (CenterCropVideo, NormalizeVideo,)
 4 from pytorchvideo.transforms import (ApplyTransformToKey, ShortSideScale, UniformTemporalSubsample)
 5 # Setting clip parameters for video transformation
 6 num_frames = 8
 7 sampling_rate = 8
 8 frames_per_second = 25
 9 # Composition of video transformations
10 transform =  ApplyTransformToKey(
11     key="video",
12     transform=Compose(
13         [
14             UniformTemporalSubsample(num_frames),
15             ShortSideScale(size=256),
16             CenterCropVideo(crop_size=(224, 224)),
17             Lambda(lambda x: x/255.0),
18             NormalizeVideo([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
19         ]
20     ),
21 )
22 # Clip duration needed to sample the required frames in specified interval
23 clip_duration = (num_frames * sampling_rate)/frames_per_second
24 dataset = pytorchvideo.data.Ucf101(
25             data_path=dataset_path,
26             clip_sampler=pytorchvideo.data.make_clip_sampler("random", clip_duration),
27             video_sampler=torch.utils.data.SequentialSampler,
28             transform=transform,
29         )
30 loader = torch.utils.data.DataLoader(dataset, batch_size = 16)
```

Preparing a video for model evaluation is a bit more involved than for an image. The video pipeline introduces the extra step of first needing to sample a clip from each video. PyTorchVideo is a PyTorch library [70] that can help sample the clip while torchvision is again used to provide transformation functions to augment said clip. The setup shown above is motivated from [34]. Here, UCF-101 is chosen as dataset but also other datasets could be chosen (e.g. Kinetics-400). In line 26, the clip sampler randomly samples a clip from the video. The provided duration in seconds needs to be long enough, so that the clip contains sufficient frames. In line 10, the composition of transformations starts. First, the PyTorchVideo function *UniformTemporalSubsample* takes the provisional clip and the required number of frames. This function creates a new clip containing only the specified amount of frames which are all uniformly separated. Therefore, it is important to set the clip duration correctly, so that the separation equals the sampling rate. Proceeding, the remaining part is akin to the image transformations. The short side of all frames gets matched to 256 pixels, a 224x224 crop is taken, the resulting tensor is rescaled and finally normalized. The mean and standard deviation values are taken from ImageNet.

3.4 Visualizations of Dimensional Collapse

3.4.1 Preamble: Representations

Before we discuss, how we analyse representations for dimensional collapse, it might be worth while to think about how the final representation vector gets constructed. For instance, let us consider a video clip traversing a Slow-only encoder (i.e. 3D ResNet-50). Figure 2.20 describes the path the clip has to travel. The exemplary clip consists of eight frames that are sampled eight frames apart from each other. The clip passes through the residual stages, and spatial as well as temporal features are obtained. The representation shape after the final fifth residual stage is $2048 \times 8 \times 7 \times 7$. After this stage, the only thing that is left, is a global average pooling operation to convert the volumetric tensor shape into a vector form. Therefore, the final representation is a vector, which is further used for downstream tasks, or in our case, to analyse its representation space. More so, for classification, a fully connected linear layer is appended to the pooling layer with as many neurons as classes. To only evaluate the representation vectors, the linear layer can be set to the identity layer (e.g. in PyTorch with `model.fc = torch.nn.Identity()`).

Nonetheless, much feature information about the data is contained in the $2048 \times 8 \times 7 \times 7$ tensor that is recovered after the final stage before it is further reduced through pooling. Analysing such representation would give compelling insight into the data. Incidentally, this also a attractive aspect about the DPC approach [17], since its loss is not based on representation vectors but the uncompressed feature tensors. However, even for DPC, the final representations used for transfer tasks are still pooled representation vectors. In the end, investigating a $2048 \times 8 \times 7 \times 7$ tensor is not feasible. Computing a 802816×802816 sample covariance matrix is computationally out of the question. Those tensors have to be reduced further for analysis. Thus, in this thesis, all data is represented through global average pooled representation vectors, as originally intended in Figure 2.20.

3.4.2 Singular Value Spectrum

The first method that will be used to examine dimensional collapse has already been teased in section 2.5.2 of the literature review and originates from the paper *Understanding Dimensional Collapse in Contrastive Self-Supervised Learning* by Jing et al. [1]. A singular value spectrum is presented to judge the rank of the representation space. Whether the representation vectors fully span the space or fall into a lower-dimensional subspace. Figure 3.1 shows an example plot of a singular value spectrum. The singular value indices are in sorted order listed on the x-axis and the singular vales in logarithmic scale are plotted on the y-axis. The plot indicates how information content is distributed over the dimensions. Ranging from dimensions being of equal importance to all information content being expressed by only a few dimensions.

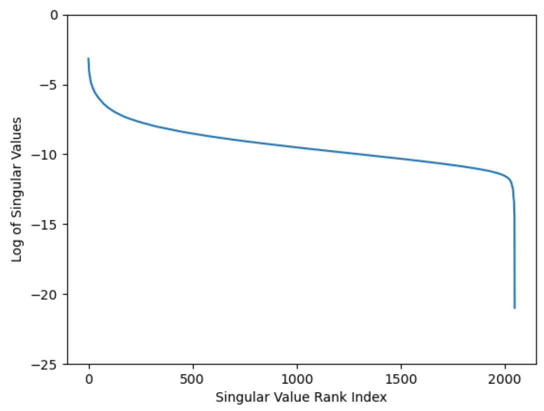


Figure 3.1: Example of Singular Value Spectrum plot

To evaluate the dimensionality of the representation space of a certain self-supervised method, representation vectors have to be collected first. Using such self-supervised pretrained encoder model, the representation vectors can be collected on a chosen dataset or dataset split. Those vectors are then stored in a representation matrix $Z \in \mathbb{R}^{N \times D}$. Here, N stands for the number of samples and D stands for the dimensionality of the representation vector. For example, considering the UCF-101 dataset and a 3D ResNet-50 encoder, the matrix would be of size 13320×2048. The next step consists of computing the sample covariance matrix $C \in \mathbb{R}^{D \times D}$. For every entry of C, the covariance is estimated based on the N samples in Z. As in section 2.5.2 already shown, the covariance matrix C is computed as follows [1]:

$$C = \frac{1}{N} \sum_{i=1}^{N} (z_i - \bar{z})(z_i - \bar{z})^T \quad \text{where} \qquad \bar{z} = \sum_{i=1}^{N} \frac{z_i}{N}$$

With the help of the covariance matrix C, it can be shown how much dimensions vary with each other, but for example, considering a 2048×2048 matrix, the information is not very easy to interpret by just looking at such a large matrix. We want to discern whether every dimensions holds exclusive information about unique features or whether different dimensions share similar information which would make them redundant. To find this out, a Singular Value Decomposition (SVD) is applied to the covariance matrix C. For square and symmetric matrices, like a covariance matrix, the singular values of SVD and the eigenvalues of an eigendecomposition are equal [74]. Therefore, this procedure can also be understood as a Principal Component Analysis (PCA) of representation matrix Z. The covariance matrix of Z is decomposed and we interpret its eigenvalues/singular values. In PCA, principal components (i.e. eigenvectors) are pointed into the direction of most variance and are orthogonal to each other. The respective eigenvalues measure the amount of variance explained by each component. Dimensions in representation space that strongly correlate with each other can be represented in PCA by fewer principal components of higher variance. The variance is large in the first few eigenvalues but the remaining eigenvalues move close to zero. They are not needed to represent the information content of Z. Dimensions, that do not hold unique information and only possess redundant information, collapse with zero variance eigenvalues.

In this method, the information content is described through the singular values of C. The implementation of the singular value spectrum is rather straight forward thanks to help of numerical libraries. Before computing the singular value spectrum, the representations for matrix Z have to be stored.

```python
1  import torch
2  import numpy as np
3  z = torch.load(data_path)
4  z = torch.nn.functional.normalize(z, dim=1)
5  z = z.cpu().detach().numpy()
6  z = np.transpose(z)
7  c = np.cov(z)
8  _, e, _ = np.linalg.svd(c)
9  np.log(e)
```

In the example above, PyTorch is used for representations obtained in PyTorch but using other frameworks is also possible without much change (e.g. using NumPy to load representations from TensorFlow). First, after loading the representation matrix, all the representation vectors are normalized to have a magnitude of one. This helps to compare multiple spectra of different supervised or self-supervised representation learning methods. The next big step is the construction of the covariance matrix, as seen in line seven. Afterwards, the singular values are computed using the SVD implementation of the NumPy library [75]. Lastly, in line nine, the natural logarithm is applied to every singular value. Figure 3.1 shows the example plot of a singular value spectrum in logarithmic scale. This is done to especially highlight scalar differences between numbers of small scale. Since non-zero singular values represent the rank of a matrix, it is even more important to show when dimensions collapse at zero. This kind of visualization helps to display distribution of information content between feature dimensions and detection of sudden collapse.

3.4.3 Cumulative Explained Variance

The second method that will be used in the exploration chapter to examine dimensional collapse has been introduced in the paper *Understanding Collapse in Non-Contrastive Siamese Representation Learning* by Li et al. [20] and also shares some principal similarities with the first described method. The idea is to plot the cumulative explained variance of the representation dimensions.

```
1  import torch
2  import numpy as np
3  from sklearn.decomposition import PCA
4  z = torch.load(data_path)
5  z = torch.nn.functional.normalize(z, dim=1)
6  z = z.cpu().detach().numpy()
7  pca = PCA()
8  pca.fit_transform(z)
9  s = pca.singular_values_
10 np.cumsum(s) / s.sum()
```

The basis lays again a matrix $Z \in \mathbb{R}^{N \times D}$ storing the representation vectors. As instructed in [20], PCA is performed on matrix Z to obtain its singular values, $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_D$. With the scikit-learn library [76], PCA uses SVD and performs the decomposition on Z. Differently than in the first method, where SVD was used on the covariance matrix C, SVD is directly applied on matrix Z. In line eight, Z is centered and subsequently decomposed. Here, the current singular values s of representation matrix Z are related to the previous singular values e of covariance matrix C as $s^2/(N-1) = e$ [74, 76]. After obtaining the vector of singular values s, a cumulative sum vector is formed and divided element-wise by its total sum.

In Figure 3.2, an example plot is drawn with the increasing cumulative sum on the y-axis and the singular value indices in sorted order on the x-axis. This kind of visualization does not use a logarithmic scale. The increasing values are bound between zero and one. Here, it is important to inspect how fast the curve rises to a cumulative value of one. A straight line indicates no collapse because all singular values have an equivalent value. If the curve rises fast, most variance will already have been explained and the remaining indices have not much information content to provide. If a cumulative value of one is reached, all remaining indices (i.e. dimensions) will be collapsed.

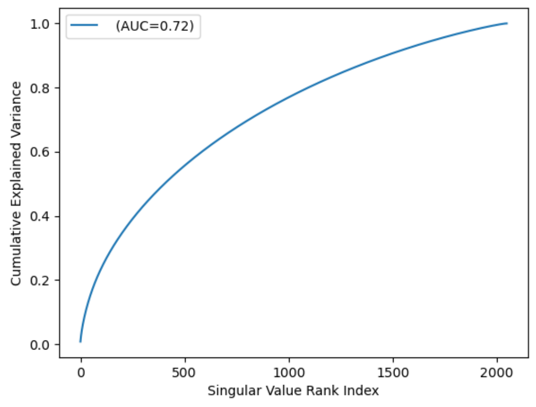


Figure 3.2: Example of Cumulative Explained Variance plot

Additionally, the AUC value of each curve is printed on the plot, as seen in Figure 3.2, to more easily compare multiple curves [20]:

$$\text{AUC} = \frac{\frac{1}{D}\sum_{i=1}^{D}\sum_{j=1}^{i}\sigma_j}{\sum_{k=1}^{D}\sigma_k}$$

As the trend shows, both methods concern themselves with the rank of the representation space to determine collapse but may look at it with different lenses. In the upcoming exploration chapter, we will use these methods to get an idea as to what degree common image and video representation learning approaches suffer from dimensional collapse.

4 Exploration

Let us start exploring dimensional collapse. In this comparative study, we will consider supervised representation learning and self-supervised representation learning. The respective representation learning methods have been introduced in the literature review chapter. Image-based approaches will offer a starting point that will lead into the central video-based approaches. To start from the beginning again, representations of the AlexNet architecture will be covered first.

4.1 Supervised Representation Learning

4.1.1 Image Representation Learning

AlexNet

AlexNet is the first image-based supervised representation learning method presented in section 2.1.1 of the literature review. It started the trend of GPU trained deep convolutional neural networks and performed with a roughly 10% lower error rate than the runner-up in the ILSVRC of 2012 [2]. Therefore, it is just fitting to also begin the exploration with AlexNet. Its performance on ImageNet is by now long passed, as seen in Table 2.1, but many ideas of its have carried over to more current deep CNNs. The architecture of AlexNet is illustrated in Figure 2.1. It is made up of five convolutional layers and three fully connected linear layers with non-linearities in between [2]. The five convolutional layers learn the spatial features and condense the high-dimensional feature down to a more compact representation. After having traversed the first five layers, the shape of the image representation is $256 \times 6 \times 6$. For the rest of the network, the three fully connected layers take over. The last layer is hereby not of interest to us, since it learns the class membership for the classification task. Otherwise, the two preceding fully connected layers further try to improve up on the $256 \times 6 \times 6$ feature maps, to refine the representations or to better classification ability. Therefore, analyzing both for dimensional collapse, the representations obtained after the five convolutional layers and the representations obtained after the next two fully connected layers, could be compelling to detect possible differences in behaviour. The model used for computation is the pretrained model described in Table 2.1 and the representation are computed on the ImageNet validation set.

Figure 4.1 shows the singular value spectra for both sets of representations in logarithmic scale. On the left side, the spectrum of the representations from the first five convolutional layers is shown. Its final representation shape of $256 \times 6 \times 6$ was flattened to construct a representation matrix of size 50000×9216. The ImageNet validation set contains 50,000 image samples. Therefore, its resulting covariance matrix is of size 9216×9216, the first and largest sample covariance matrix computed in this thesis. Let us now inspect the singular value spectrum itself. Similar to the visualizations in [1], the plot is limit on the y-axis to values between -25 and 0, to easier compare plots of different singular value spectra. At first sight, no apparent collapse is visible. A significant drop in between small values is not shown. In the logarithmic scale, the singular

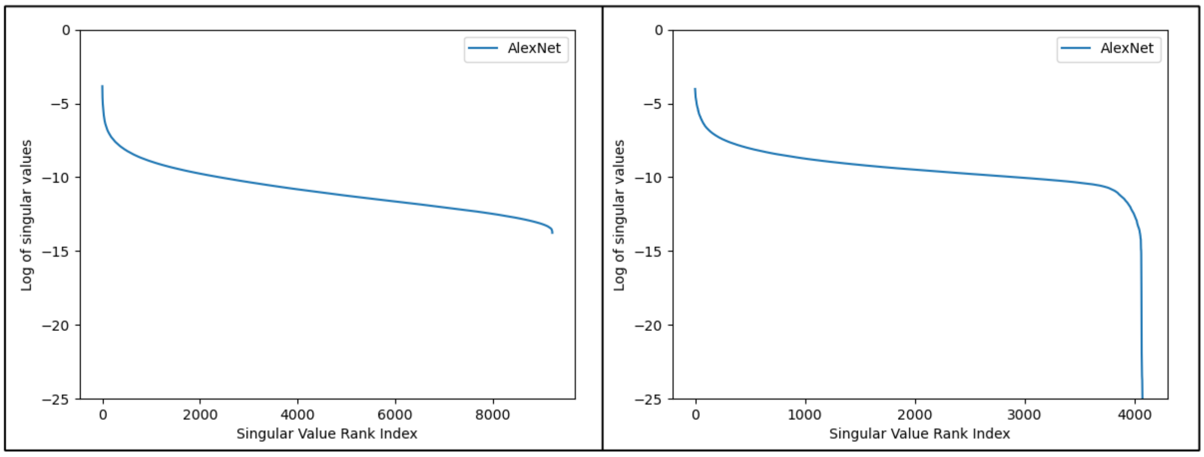


Figure 4.1: Left: Spectrum plot of AlexNet representations obtained after the final convolutional layer. **Right**: Spectrum plot of AlexNet representations obtained before the last fully connected layer.

values decline steadily from dimension to dimension. On the right side of Figure 4.1, the spectrum of the representations before the last fully connected linear layer is presented. In AlexNet, the second to last layer contains 4096 neurons. Therefore, the spectrum presents indices of 4096 singular values in descending order. The picture is very similar to the one displayed on the left side. No drop is visible until the very last dimensions. Before, the values remain consistent. Presently, the two fully connected layers following the first five convolutional layers do not seem to negatively impact the network in terms of collapse.

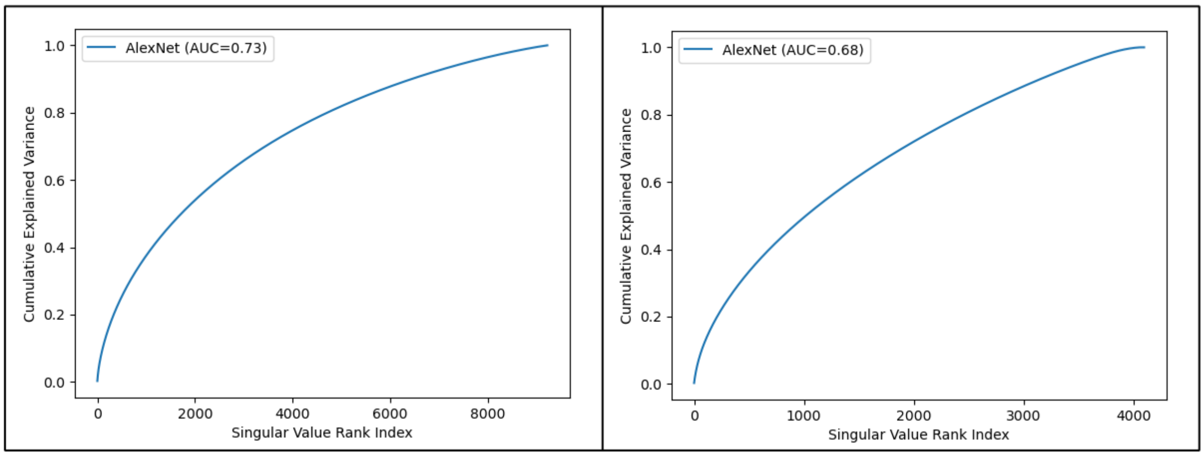


Figure 4.2: Left: CEV plot of AlexNet representations obtained after the final convolutional layer. **Right**: CEV plot of AlexNet representations obtained before the last fully connected layer.

Figure 4.2 shows the Cumulative Explained Variance (CEV) plot for both representation spaces as well as the respective Area under the Curve (AUC) values. Maybe this view paints another picture. Since the representation spaces are of different dimensionality, a direct comparison is more difficult. Both curves appear visually similar but the AUC values indicate a difference. The AUC value for the left plot is slightly larger with 0.73 to the 0.68 of the right plot. With this discrepancy in mind reevaluating Figure 4.1, the right plot of the representation space, which includes the two fully connected layers, appears slightly more horizontal than the spectrum in the left plot. All in all, this difference is negligible and we can assume for both representation spaces no dimensional collapse. So, AlexNet stands the test of time also in this regard.

ResNet and Depth Variation

The second architecture discussed for supervised training in image representation learning was ResNet, the Residual Network [3]. It was introduced in section 2.1.2 of the literature review and lays the foundation as baseline encoder for many self-supervised methods like BYOL, VICReg and SimCLR, just to name a few. ResNet pioneered the idea of residual connections in neural network architectures. The ability to build deeper networks while still being able to train them efficiently [3]. The ResNet-50 construction, made up of 50 convolutional layers, is the most common variant and is most often chosen as baseline encoder in self-supervised image learning. Nevertheless, a ResNet architecture can be build with a various number of residual blocks. It is a trade of between model size and performance improvement.

ResNet Variant	Output Dimensions	Parameters	Top-1 Accuracy	Top-5 Accuracy
ResNet-18	512	11,689,512	69.75%	89.07%
ResNet-34	512	21,797,672	73.31%	91.42%
ResNet-50	2048	25,557,032	76.13%	92.86%
ResNet-101	2048	44,549,160	77.37%	93.54%
ResNet-152	2048	60,192,808	78.31%	94.04%

Table 4.1: Comparison of ResNet models of varying number of layers on ImageNet-1k

Table 4.1 [77] lists ResNet variants of increasing layer depth. The smallest model contains 18 layers and the largest model contains 152 layers. Furthermore, the dimensionality of the resulting image representation vectors is different for the ResNet-18 and ResNet-34 model versus the ResNet-50, ResNet-101 and ResNet-152 model (i.e. 512 versus 2048). Regarding model size, the amount of parameters per model rises linearly with the number of layers, while the validation accuracy does not follow that trend. The improvements in accuracy from model to model saturate slowly after ResNet-50. Probably a big reason as for why the ResNet-50 model is chosen frequently as encoder model. At a certain layer depth, more layers come with diminishing returns. Still, the accessibility to various pretrained ResNet models allows to examine the behaviour of dimensional collapse regarding different levels of network depth. Does the ResNet network remain consistent or vary when consider deeper models like it does for the accuracy metric?

Figure 4.3 presents the spectrum and CEV plots for all ResNet models listed in Table 4.1. ResNet-18 and ResNet-34 with an output dimensionality of 512 are illustrated in the two top plots. ResNet-50, ResNet-101 and ResNet-152 with an output dimensionality of 2048 are shown in the bottom plots. Also, we have to keep in mind that all these models have been successfully pretrained on the same large ImageNet-1k dataset. The dataset size could as well play a part in the dimensional collapse, as shown in [20]. At first sight, the singular value spectra for all five ResNet models in Figure 4.3 look stable. Most noticeably, for each separate 512-dimensional and 2048-dimensional space, the singular value spectra look almost identical. Only the spaces themselves differ from each other. The 2048-dimensional space, belonging to ResNet-50, ResNet-101 and ResNet-152, falls off slightly faster, when inspecting the largest singular values, before flattening out at around singular value index 400. This is also visible in their CEV plot to the

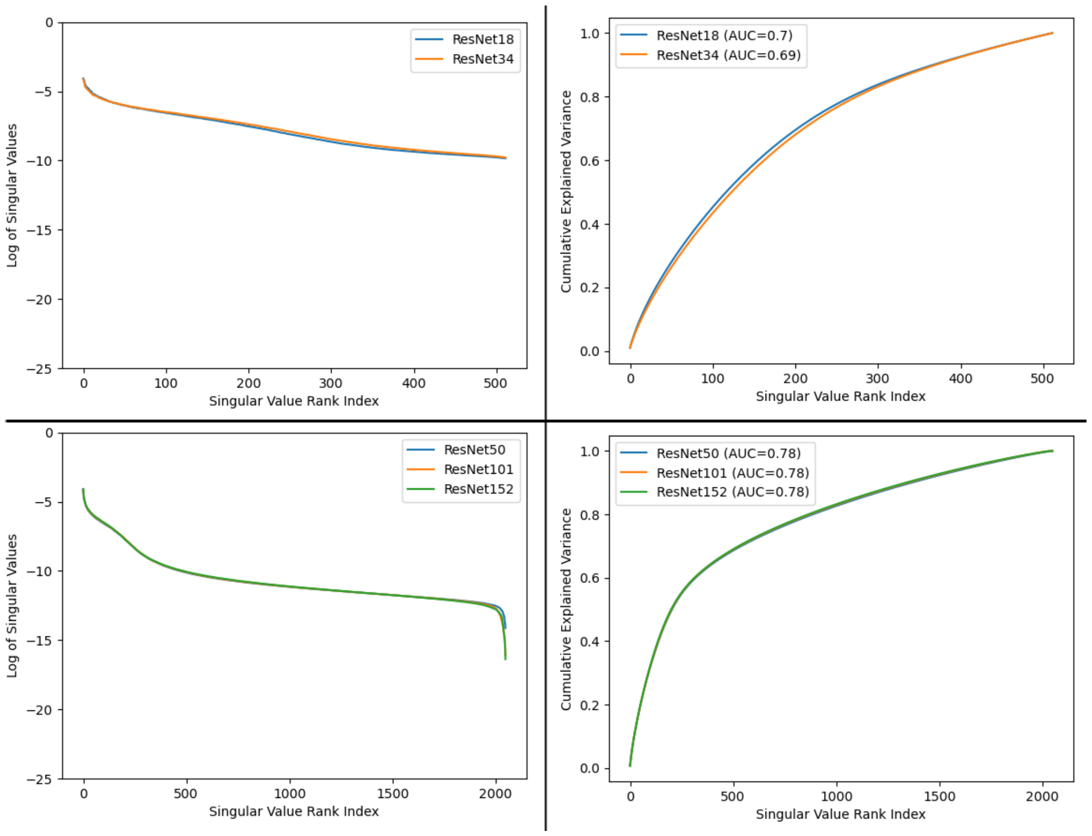


Figure 4.3: Top Left: Spectrum plots of ResNet-18 and ResNet-34. **Top Right**: CEV plots of ResNet-18 and ResNet-34. **Bottom Left**: Spectrum plots of ResNet-50, ResNet-101 and ResNet-152. **Bottom Right**: CEV plots of ResNet-50, ResNet-101 and ResNet-152.

right where the curve starts to rise fast but slows down at around index 400. Furthermore, the AUC levels are moderately elevated compared to the 512-dimensional space. The discrepancy of model parameters between the ResNet-18 model and ResNet-152 model does not have a large impact on the singular value spectrum. But, on the other hand, the output dimensionality does seem to play a role. AUC levels as well as spectrum decay differ sightly between the two. Still, none of the two spaces spanned by ResNet encoders appear to strongly suffer from dimensional collapse. More so, it will be intriguing to see the sensibility of ResNet encoders' representations towards collapse when trained with various self-supervised image methods instead of the supervised counterpart.

4.1.2 Video Representation Learning

Examining dimensional collapse in supervised video representation learning entails new architecture types and datasets. As for datasets, the video dataset UCF-101 will be used to create representations as introduced previously in section 3.2. More so, UCF-101 will be used as standard video dataset throughout exploration. The network architectures that will be analyzed for dimensional collapse are listed in Table 2.2 and have been established in the literature review. All three of them are ResNet-50 type architectures with the inclusion of 3D convolutional layers for video processing. ResNet(2+1)D splits the 3D convolution operation into two separate operations. The first operation is a 2D spatial convolution and the second operation is a 1D temporal convolution to analyze the spatiotemporal video volume [5]. SlowFast [7], on the other hand, is an architecture containing two pathways, a Slow pathway and a Fast pathway. The Slow pathway determines slow temporal and spatial features, while the Fast pathway analyses the fast motion in video clips by considering a faster frame rate but fewer channels. Both pathways share information through lateral connections from the Fast pathway into the Slow pathway. The final video clip representation is constructed by concatenating the Slow pathway output and the Fast pathway output. For the 50 layer variant, which we consider, the Slow pathway has a standard 2048-dimensional output, like a standard 2D ResNet-50, and the Fast pathway has a considerably smaller 256-dimensional output [7]. Lastly, the third network, that will be analyzed for dimensional collapse, is the individual Slow pathway in isolation. Thus, it is named the Slow-only network. It is especially considered, since it is later used as backbone in many self-supervised video learning frameworks.

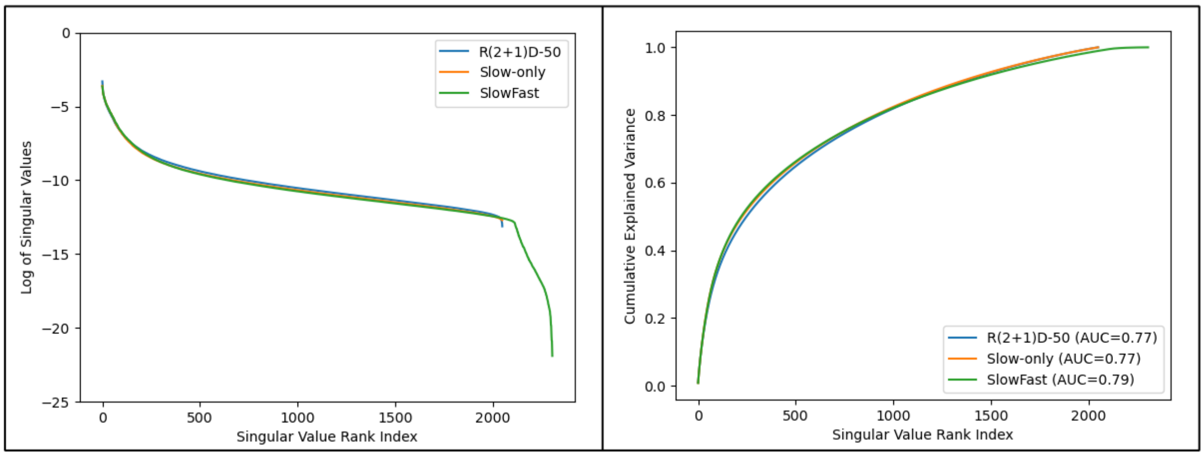


Figure 4.4: Left: Spectrum plot of ResNet(2+1)D-50, Slow-only and SlowFast. **Right**: CEV plot of ResNet(2+1)D-50, Slow-only and SlowFast.

Figure 4.4 shows the spectrum and CEV plots for the representations of all three network architectures. While the representations of the SlowFast network are larger (2048+256=2304) than the ones of R(2+1)D and Slow-only (2048), all three networks are included in the same plot since the difference is not substantial and it allows for further comparison. First of all, the spectra for the R(2+1)D and Slow-only network are identical as well as their curves in the CEV plot. This is not too surprising because of many shared similarities. As seen in the Figure 2.20, the Slow-only architecture similarly separates spatial and temporal convolutions. The difference, that remains, is the late introduction of temporal convolutions in the Slow-only network while R(2+1)D utilizes them in every stage of the network. In the SlowFast architecture, it is split. The Slow pathway introduces temporal convolutions only in the fourth stage onwards and the Fast pathway uses them in every stage. However, all these details do not seem to play a role in terms of dimensional collapse. Additionally, SlowFast's plots are coinciding with the ones of R(2+1)D and Slow-only for the first 2048 singular values. Afterwards, the SlowFast spectrum stays stable for only a few singular values and then collapses. Since the SlowFast spectrum coincides with the Slow-only spectrum for at least 2048 singular values, the assumption is easily made that the remaining singular values mostly represent the 256 dimensions of the Fast pathway. Through lateral connections, as mentioned before, the Fast pathway shares information with the Slow pathway to represent a coherent picture for the representation [7]. Although, this sharing of information between pathways can lead to redundant dimensions that possess similar information. Furthermore, redundant dimensions lead to collapsed dimensions, since not all dimensions are needed to represent the overall variance. While this likely lead to the majority of collapsed dimensions in Figure 4.4, a select few dimensions of the Fast pathway are not collapsed. Therefore, we can hypothesize that these dimensions constitute features of fast motion which are relevant for the SlowFast network to outperform Slow-only and R(2+1)D. Also, if it is only by a small margin.

Similar to 2D ResNet, the Slow-only and R(2+1)D network will reappear as backbone encoders for self-supervised video approaches. Over there, we will examine if those self-supervised methods lead to dimensional collapse of the representation space. In the supervised space so far, both inspected networks are resistant towards dimensional collapse.

4.1.3 Dataset Variation

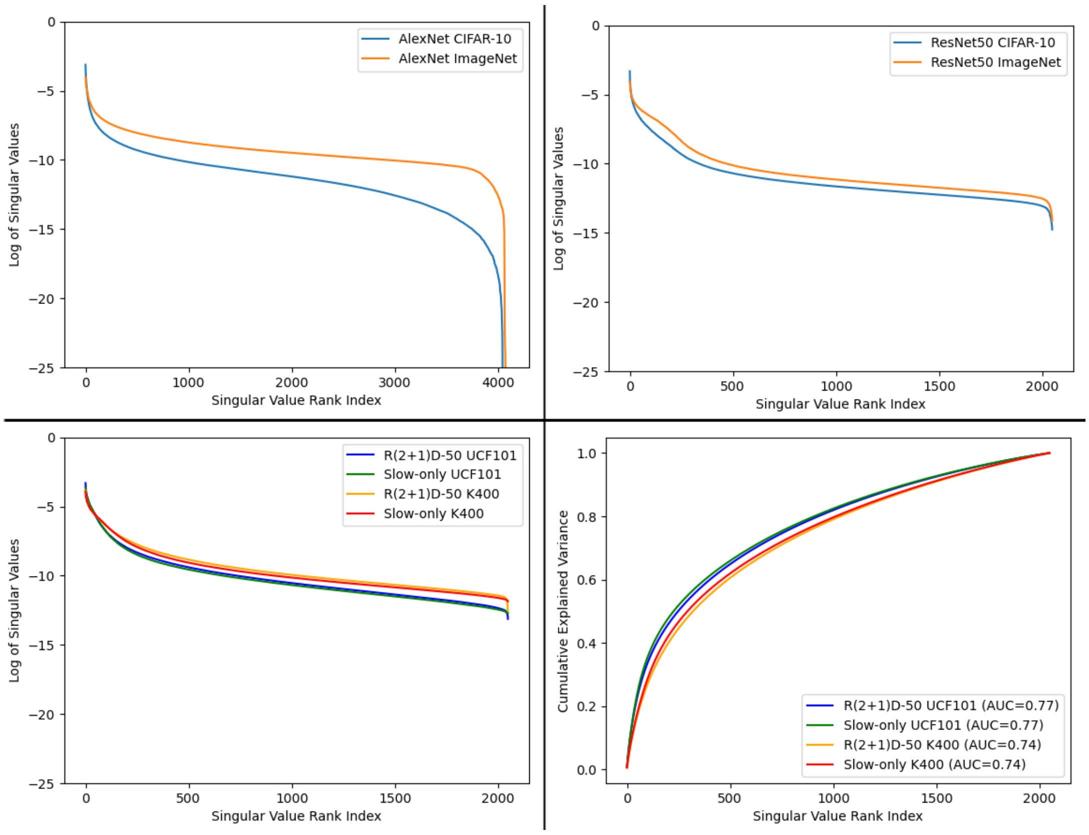


Figure 4.5: Top Left: Spectrum plot of AlexNet on CIFAR-10 train set and ImageNet validation set. **Top Right**: Spectrum plot of ResNet-50 on CIFAR-10 train set and ImageNet validation set **Bottom Left**: Spectrum plot of R(2+1)D and Slow-only on UCF-101 and Kinetics-400 validation set. **Bottom Right**: CEV plot of R(2+1)D and Slow-only on UCF-101 and Kinetics-400 validation set.

So far, for each architecture explored only one set of representations has been produced and examined. For 2D networks, the validation set from ImageNet was used and for 3D networks, the full UCF-101 dataset was applied. Previously, variation in network depth was explored with ResNet on the ImageNet validation set. Now, we are going to attempt to look into dataset variation. Because of the limitation of relying on provided pretrained networks, we are unable to train an introduced architecture on a dataset of smaller or larger size. Most pretrained models are trained on a large dataset with optimal hyperparameter selection to rival their respective state-of-the-art. Thus, we can explore different datasets as representation spaces but can not explore models trained on various datasets.

The ImageNet dataset provides great variety and richness in images to be able to differentiate between 1000 different classes. On the other hand, the CIFAR-10 dataset is a smaller dataset which only contains 10 classes. To exemplify, the training set of CIFAR-10 is as larger as the validation set of ImageNet. In the following, we are going to create representations on the CIFAR-10 training set and the ImageNet validation set and compare them for dimensional collapse. The architectures used to compute these image representations are the already examined AlexNet and ResNet-50 models. Both of them have been trained on the large ImageNet dataset. Therefore, we inspect how the feature representations learned on ImageNet serve to represent images of other origins and how this will impact their sensibility to dimensional collapse. A similar procedure is followed for video datasets. Most 3D networks, be it supervised or self-supervised, are trained on the Kinetics-400 dataset. Hereby, the full UCF-101 dataset and Kinetics-400 validation set are used to create representations. Before, representations on UCF-101 have already been analyzed on selected architectures for dimensional collapse. Presently, we also contrast them to the Kinetics-400 representations. Also to remember, similar to CIFAR-10 and ImageNet, UCF-101 contains less classes and is in general a smaller dataset than the large Kinetics-400 dataset.

Figure 4.5 shows the different spectrum and CEV plots behind representations of various datasets. In the top row, AlexNet and ResNet-50 models are contrasted on CIFAR-10 and ImageNet representations, and in the bottom row, R(2+1)D and Slow-only models are contrasted on UCF-101 and Kinetics-400 representations. To start with the top row, in preceding sections, ImageNet representations produced on AlexNet and ResNet-50 have already shown to be stable against dimensional collapse. Now, we introduce representations on CIFAR-10 into the mix. For the ResNet-50 model, the ImageNet representations, except for the very first few, hold larger singular values than for the CIFAR-10 representations. The difference between the two is not significant but large enough to notice. Still, none of the two representation spaces appear collapsed. The ResNet-50 model trained on ImageNet shows to be expressive enough to also represent images from the CIFAR-10 dataset. This story is a little bit different for AlexNet. A drift is visible, similar to ResNet-50, between the singular value spectra of CIFAR-10 representations and ImageNet representations. However, the gap is considerably more sizeable than it is for the ResNet-50 model. Especially, towards the larger indices, the singular values decrease more rapidly. The CIFAR-10 representations on AlexNet do not suddenly collapse, but between AlexNet and ResNet, the ResNet-50 model seems more generalisable towards new smaller datasets.

For video datasets, in the bottom row of Figure 4.5, R(2+1)D and Slow-only contrast representations on UCF-101 and representations on the Kinetics-400 validation set. The former configuration has already been deemed stable towards dimensional collapse. As well, the latter appears stable. Especially, since the respective models have both been trained successfully on the Kinetics-400 training set. What is visible at first sight, is that both models behave similarly towards each set of representations from UCF-101 and Kinetics-400. The behaviour echos the one of the 2D ResNet-50 model towards CIFAR-10 and ImageNet. There exists a small gap between singular values of representations on UCF-101 and representations on Kinetics-400. The choice of either R(2+1)D or Slow-only does not play a role. For the Kinetics representations, the singular values are slightly more equally distributed than for the UCF-101 representation, thus also the larger AUC value for UCF-101 (0.77 > 0.74). All in all, these differences are almost negligible in terms of dimensional collapse but might point towards a difference between representations of datasets of different expressiveness.

As procedures go, supervised representation learning is pretty straight forward and follows a simple setup, but is also limited by its labelled dataset. Next, we will inspect self-supervised methods for dimensional collapse which do not require a labelled dataset, but have a bit more going on to actively avoid collapse and learn meaningful features, which might even be more representative than the features learned through supervised learning.

4.2 Self-Supervised Representation Learning

4.2.1 Pretext Tasks

Pretext tasks are the first self-supervised approaches addressed in section 2.3.1 of the literature review. Additionally, they are also the longest standing approaches addressed in this thesis by a few years. Pretext tasks are carefully handcrafted tasks such that the neural network can only solve them by learning the necessary semantic information of the data. Mentioned are more specifically Jigsaw [8] and RotNet [9]. Both are image-based pretext approaches. Jigsaw requires to find a permutation of randomly shuffled image patches and RotNet requires to identify the correct rotation applied to the image. Table 2.3 lists the pretrained models for Jigsaw and RotNet that will be inspected for dimensional collapse. Regrettably, the pretrained models provided by the original research teams are either referenced by outdated links or trained on older frameworks. Fortunately, the Vision library for state-of-the-art Self-Supervised Learning (VISSL) by Meta AI provides pretrained weights for both Jigsaw and RotNet [37]. The pretrained model for Jigsaw is a ResNet-50 model trained on ImageNet-1k that had to distinguish 100 different puzzle permutations during training. Additionally, the pretrained model for RotNet is also a ResNet-50 model that is trained on ImageNet-1k. To be able to better compare these two pretrained models, we include also their supervised counter part.

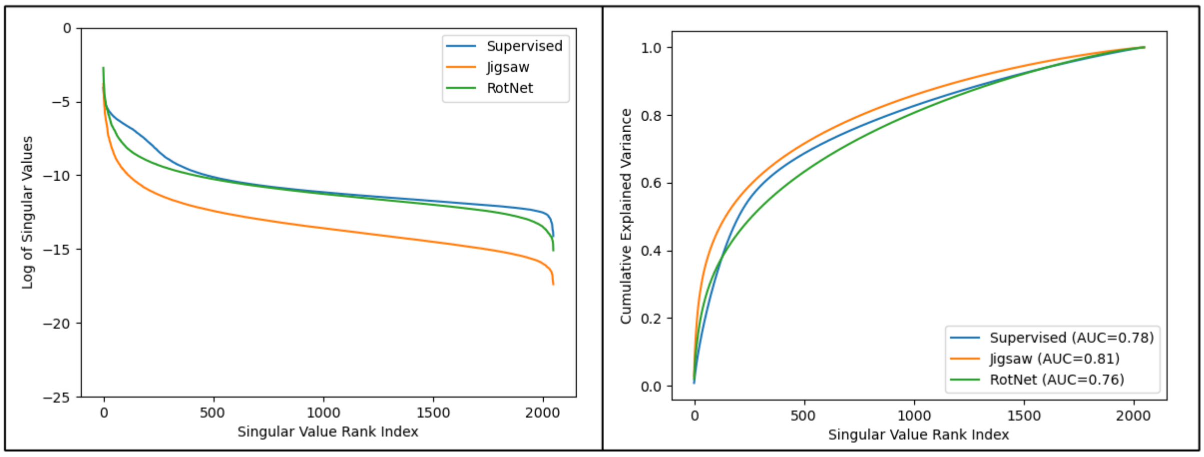


Figure 4.6: Left: Spectrum plots of Jigsaw, RotNet and supervised counterpart. **Right**: CEV plots of Jigsaw, RotNet and supervised counterpart.

Figure 4.6 presents the spectrum and CEV plots for Jigsaw, RotNet and supervised model trained on ResNet-50 with the ImageNet-1k dataset. Inspecting the singular value spectrum plot, RotNet behaves similarly to the model trained with labelled supervision. Only the behaviour of the first and last singular values differs slightly. Jigsaw, on the other hand, holds singular values of lower value for most indices, but otherwise, follows a similar pattern to RotNet. None of the three spectra experience an unexpected collapse in information. The curves in the CEV plot support the spectrum plot. The Jigsaw curve always holds the largest cumulative values while supervised and RotNet curves overtake each other. In the grand total, RotNet achieves even a lower AUC value than the supervised ResNet-50 model.

While Jigsaw and RotNet are no way near the downstream performance of the supervised ResNet-50 model, that gap in performance is not related to their sensibility to dimensional collapse. As seen in Table 2.3, the gap in Top-1 accuracy between the two pretext tasks and the supervised model is almost at 28%. In terms of dimensional collapse, RotNet is as stable as the supervised model and even Jigsaw is not too far off.

4.2.2 Information Maximization

Before talking about self-supervised video representation methods and contrast image-based and video-based approaches of BYOL, SimCLR, MoCo and SwAV, let us inspect the last purely image related experiment. BarlowTwins and VICReg are approaches, which try to prevent, what we are trying to determine. The optimization goal of BarlowTwins [14] and VICReg [15] is to decorrelate the cross-correlation and covariance matrix, respectively. BarlowTwins attempts to decorrelate the cross-correlation matrix constructed by the representations in a batch of differently augmented views. VICReg attempts to drive the covariance matrix of each separate branch of the sesame architecture to the identity matrix. Decorrelation of features has shown to mitigate dimensional collapse [19]. Thus, examining the singular value spectrum of both these approaches will be illuminating. Additionally, this will also be the first time that we inspect a self-supervised method which utilizes an extra projection head on top of its backbone encoder, as introduced in [10]. Jing et al. [1] have already hypothesized that a projector helps to prevent dimensional collapse and Mialon et al. [78] even state that a random projector can enforce pairwise independent features of the representations, when regularized by VICReg. Thus, we compute both the representations of the backbone encoder and the embeddings of the projection head. The resulting representation matrix and embedding matrix allow to inspect either space separately for dimensional collapse. The research by Jing et al. [1] on dimensional collapse even primarily focused on the behaviour of the embedding space. Also to note, the procedure of BarlowTwins and VICReg on information maximization is not limited to the image domain but pretrained models so far only exist for such.

Figure 4.7 shows the singular value spectra and CVE curves for BarlowTwins, VICReg and a supervised counterpart. All three use the ResNet-50 model as backbone encoder. The representations are obtained on the ImageNet validation set as usual. Strikingly, the singular value spectra for both BarlowTwins and VICReg are very flat. The variance is distributed more evenly than in any other representation learning method introduced prior. The set goal of decorrelating the features appears to bear fruit. More so, the spectrum is seemingly identical for the two approaches, despite the marginally different ways in achieving decorrelation. Their AUC values are the same and their curves rise slower than for any other method so far utilizing the ResNet-50 model. Speaking of ResNet-50, the plots also include the spectrum and curve of the supervised counterpart. Most interestingly, in the CEV plot, BarlowTwins and VICReg representations experience less degrees of collapse than the supervised representations. The decorrelation objec-

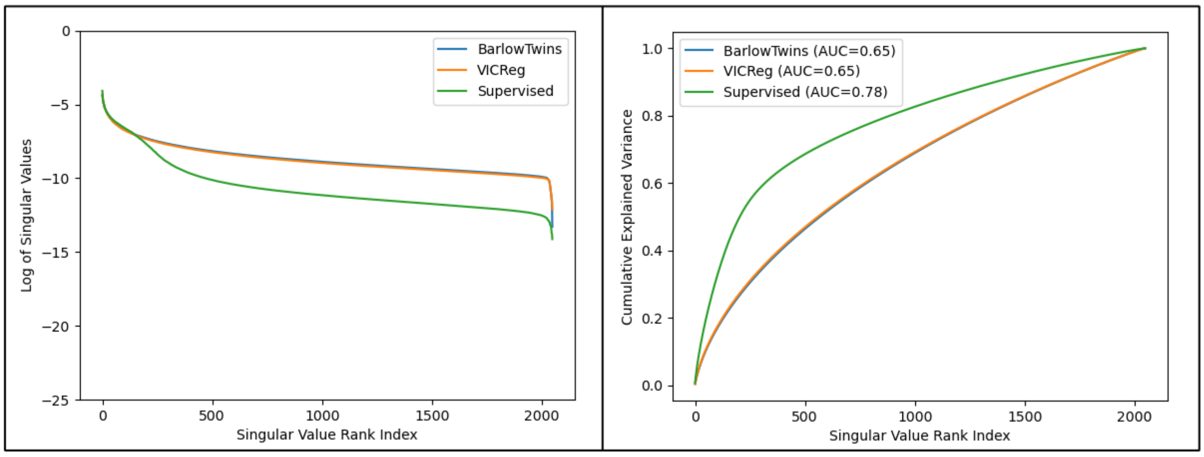


Figure 4.7: Left: Spectrum plots of representations of BarlowTwins, VICReg and supervised counterpart. **Right**: CEV plots of representations of BarlowTwins, VICReg and supervised counterpart.

tive of both approaches appears even more stable to dimensional collapse than the supervised training. To further explore this occurrence, the study on VICReg suggests that decorrelation of the dimensions at the embedding level (i.e. outputs of the projection head) has also a special decorrelation effect at the representation level, which is a non trivial phenomenon [15]. Mialon et al. [78] investigate the idea in terms of pairwise independence of the representation vectors. Additionally, pairwise independence of representations seems to be not unique to VICReg [78]. Many self-supervised learning methods share this criteria, but it can be precisely explained for frameworks using VICReg [78]. Mialon et al. [78] made use of the HSIC (Hilbert-Schmidt Independence Criterion) [79] to test for pairwise independence. As mentioned before, Mialon et al. [78] state that, especially using a projector with random weights, minimizing the variance-covariance terms of the projector's output minimizes the HSIC of the learned representations, and such enforces pairwise independence [78]. Naturally, pairwise independence is a desirable property to have but it is also not solely predictive of test accuracy, according to Mialon et. al [78]. Such, as seen in Table 2.3, BarlowTwins and VICReg still underperform slightly in contrast to supervised training. On linear evaluation, their accuracy falls short by a few percent points.

Next, we will inspect the embedding space of BarlowTwins and VICReg for dimensional collapse. The embedding space for both is very large with a dimensionality of 8192. The embeddings are the outputs of the non-linear MLP on top of the ResNet-50 backbone encoder. This projection head, which is supposedly part of the reason for non-collapsed features, is almost identical for either method and consists of three linear layers of 8192 neurons each. Furthermore, the linear layers are intersected with a BN layer and ReLU activations. The only difference is that BarlowTwins uses bias weights for its first two layers and VICReg uses them in none of the three layers.

Figure 4.8 presents the singular value spectra and CEV plots of the embedding matrices for BarlowTwins and VICReg. Thanks to these visualizations, it can be easily identified that the embedding spaces for both methods suffer from dimensional collapse. Around index 4000, the singular values drop. Especially, the VICReg spectrum drops suddenly. In the CEV plot, the dimensional collapse is described by a curve that saturates its explained variance before considering all singular value indices. The occurrence of dimensional collapse in the embedding

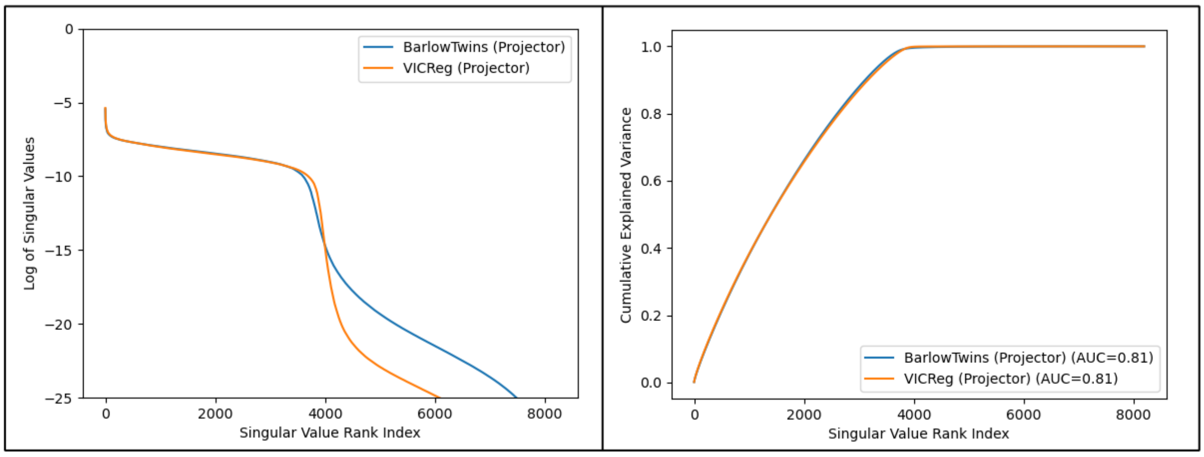


Figure 4.8: Left: Spectrum plots of projector embeddings for BarlowTwins and VICReg. **Right**: CEV plots of projector embeddings for BarlowTwins and VICReg.

space spanned by the projector is particularly interesting since the decorrelation objective worked directly on the embedding vectors and attempted to decorrelate their sample covariance/cross-correlation matrix. However, despite a collapsed embedding space or precisely because of a collapsed embedding space, the representation space experiences none of it. Therefore, since the representations, and not the embedding vectors, are used further for downstream tasks, should not only the rank of the representation space matter?

In the following, we will inspect how differently the embedding and representation spaces behave for in-demand approaches like SimCLR, BYOL, MoCo and SwaV, which do not directly decorrelated their features. Furthermore, we will also start to explore the self-supervised video variants of these approaches, and contrast the image and video medium.

4.2.3 Comparison of Mediums

Besides the pretext task and information maximization approaches, the self-supervised methods of SimCLR, BYOL, MoCo and SwAV have been discussed during the literature review. In addition to that these four methods have also been realized for self-supervised video representation learning, as described in section 2.4.1 of the literature review. Thus, in this section, we well determine the dimensional collapse of all four methods, be it in the image domain or in the video domain, and contrast their respective results. Apart from the representation spaces, we will also examine their embedding spaces and compare them to our findings from the previous section. All four methods make us of the projection head extension introduced in [10]. To further clarify, the respective image and video implementations might differ in certain training procedures or hyperparameter choices, since the research teams, which provide the pretrained models, are different for each method.

To shortly recapitulate, all four methods follow the basic procedure of randomly augmenting two views of the same data sample, encoding both views and trying to find an encoding which is invariant to the random data augmentations, and instead, learns the semantics of the source. The four methods differentiate themselves from each other by how they learn invariance to augmentations, and especially, by how they avoid complete collapse. SimCLR employs a contrastive loss and introduces negative samples to avoid complete collapse [10]. MoCo uses a similar contrastive loss, utilizes a dictionary type structure for negative samples, and additionally, employs a momentum encoder [11]. SwAV learns online clusters to contrast embedding vectors [12]. And lastly, BYOL uses no negative samples, employs a momentum encoder just like MoCo and tries to predict the embedding of the other view [13].

We will begin with examining the representation spaces for dimensional collapse. First, the representation spaces spanned by the image-based versions, and secondly, the representation spaces spanned by the video-based versions of SimCLR, BYOL, MoCo and SwAV. After concluding that analysis, the embedding spaces will be determined for dimensional collapse. Those spaces are spanned by the embedding vectors. The embedding vectors are obtained by using the method's respective projection head on the representations. The pretrained models utilized in this section are listed in Table 2.3 and in Table 2.4 for images and videos, respectively. As mentioned, the work by Feichtenhofer et al. [16] implements video-based variants of all four methods. The pretrained models of the image-based counterparts, on the other hand, have been collected from the repositories of the respective research teams, with exception of SimCLR. The pretrained model provided by the SimCLR research team, unfortunately, did not provide pretrained weights of the projection head. Therefore, a pretrained model was chosen from the VISSL reference implementations [37]. For assurance, the representations of the official implementation and the representations of the reference implementation share the same singular value spectrum.

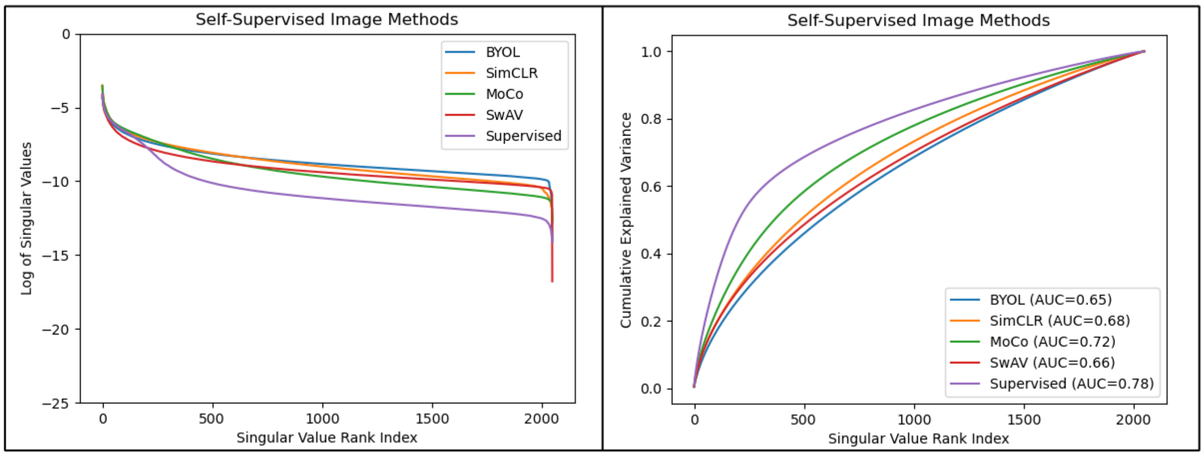


Figure 4.9: Left: Spectrum plots of representations of image-based BYOL, SimCLR, MoCo, SwAV and supervised counterpart. **Right**: CEV plots of representations of image-based BYOL, SimCLR, MoCo, SwAV and supervised counterpart.

Figure 4.9 provides the singular value spectrum plots and CEV plots for image-based BYOL, SimCLR, MoCo, SwAV and their supervised counterpart. The backbone for all these approaches is again the 2D ResNet-50 model. Fortunately, the singular value spectra appear to show no dimensional collapse. All the self-supervised methods experience no sudden drop in information content. Interestingly enough, the representations learned through labelled supervision contain the largest degree of dimensional collapse. The CEV curves illustrate that the variance explained in the first roughly 250 singular values is larger for the supervised counterpart than for any of the self-supervised methods. Additionally, the AUC values for the presented approaches, with the slight exception of MoCo, are all comparatively low. BYOL, which carries the lowest AUC value with 0.65, even achieves a degree of collapse as low as the information maximization methods, displayed in Figure 4.7. While BarlowTwins and VICReg directly attempt to decorrelate their features, BYOL, SwAV, SimCLR and MoCo do not do so. Their procedures more evolve around avoiding complete collapse and spreading out the embedding space. More so, this gives further credence to the statement from Mialon et al. [78] that pairwise independence in representations emerges from many different self-supervised learning criteria.

Next, we will, for the first time, examine self-supervised video representation learning methods. As teased, these are again BYOL, SimCLR, MoCo and SwAV following the strategy of Feichtenhofer et al. [16]. The backbone encoder for all these methods is a Slow-only network of 50 layers as already analyzed previously in section 4.1.2 of the exploration chapter. As for the previous 2D ResNet-50 model used for imaged-based approaches, the representation vector of the Slow-only network has the same dimensionality of 2048, allowing for a more direct comparison with the just inspected image representations.

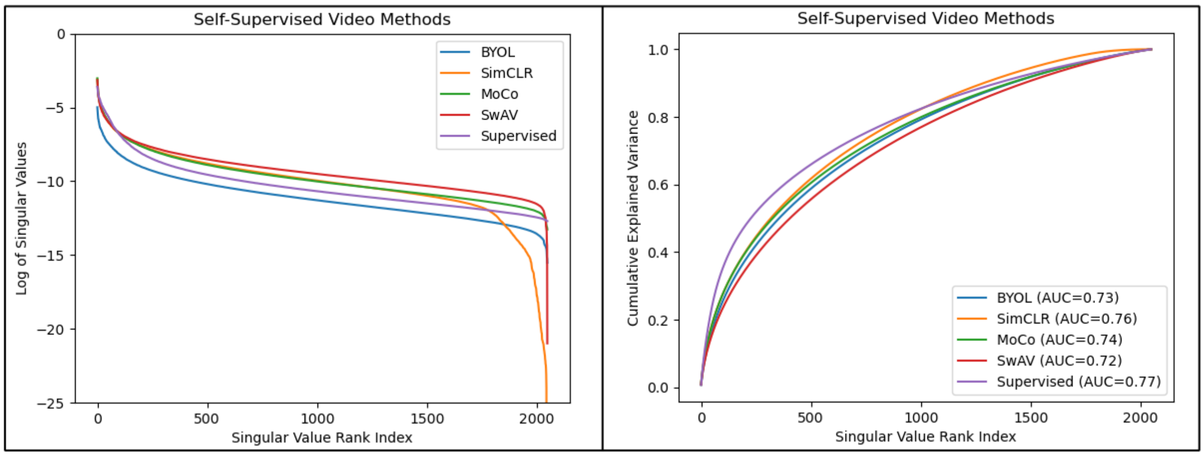


Figure 4.10: Left: Spectrum plots of representations of video-based BYOL, SimCLR, MoCo, SwAV and supervised counterpart. **Right**: CEV plots of representations of video-based BYOL, SimCLR, MoCo, SwAV and supervised counterpart.

This time around, Figure 4.10 provides the singular value spectrum plots and CEV plots for video-based BYOL, SimCLR, MoCo, SwAV and their supervised counterpart. Again no strong dimensional collapse is visible. Although, SimCLR stands out as its singular values for the later indices decay faster than for the other methods. What remains consistent in comparison to Figure 4.9 of the image representations, is that BYOL and SwAV share lower overall AUC values than SimCLR and MoCo. Although, for the video representations, all the values are generally in closer vicinity. This could be an artifact of the pretrained models, shown in Figure 4.10, originating from the same source, whereas for the image representations, the models stem from different research teams with likely differentiating training conditions. Furthermore, the supervised counterpart obtains a larger AUC value than the self-supervised methods, but here, the gap is rather negligible. Additionally, the AUC values are overall larger for the self-supervised video methods than for the image methods. This might partially have to do with the phenomenon encountered in section 4.1.3. At least for the supervised case, as seen in Figure 4.5, representations obtained from a dataset of different origin, compared to the training dataset, raise the degree of collapse by a small margin (i.e. AUC of CEV). The video models have been trained on Kinetics-400 but representations stem from the UCF-101 dataset. Also to note, the performance gap in linear evaluation for self-supervised video methods is larger than for self-supervised image methods. Table 2.4 for video methods in contrast to Table 2.3 for image methods shows that the gap towards the supervised baseline for video methods is still more pronounced. But this performance difference of domains is not discernible through spectrum or CEV plots. As also stated in [20], dimensional collapse is not solely predictive of downstream performance. All in all, these annotations do not display too significant of a difference between the two mediums. BYOL, SimCLR, MoCo and SwAV enable stable representations for both images and videos.

However, as already established in the information maximization section 4.2.2, representations resistant to dimensional collapse do not guarantee a non-collapsed embedding space. Therefore, next, we will examine the role of the projector for all four methods, starting with the image-based versions and concluding with the video-based versions.

Role of Projector

All four methods utilize a projection head. Thus, the loss is not computed directly on the representation vectors but on the embedding vectors of the projection head. The idea of an additional projection head was first introduced by Chen et al. [10] to further improve downstream performance. Jing et al. [1] also hypothesize that the usage of a projection head in contrastive learning is essential to prevent dimensional collapse in the representation space. So far, this appears to be true. None of the multiple representation spaces, which applied a projector, have collapsed. Nonetheless, the embedding space still remains to be examined.

SSL Method	Medium	Hidden Layers	Batch Norm	Bias
SimCLR	Image	2		✓
SimCLR	Video	3		✓
MoCo	Image	2		✓
MoCo	Video	3		✓
SwAV	Image	2	✓	✓
SwAV	Video	3	✓	
BYOL	Image	2	✓	✓
BYOL	Video	2	✓	

Table 4.2: Varying Projector setup for SimCLR, MoCo, SwAV and BYOL

While the backbone encoder for image and video methods, respectively, has been identical (i.e. ResNet-50 for images and Slow-only for videos), the projection head architectures differ to some degree between the various approaches. Table 4.2 displays distinctions in the setup of the specific projection heads. Most notably, SwAV and BYOL utilize BN layers in their projection heads while SimCLR and SwAV do not. Furthermore, Feichtenhofer et al. [16] investigated the effect of using different amounts of hidden layers in their projection heads. Especially for Kinetics-400, using more than two hidden layers improved the linear evaluation performance. Thus, for the video-based methods, three hidden layers are employed, except for BYOL which saw no improvement using more than two hidden layers [16]. Additionally, opposite to the other methods, BYOL's projector has an output dimensionality of 256, while the other methods have an output dimensionality of 128.

Figure 4.11 shows the singular value spectrum plots for the embedding spaces of SimCLR, MoCo,

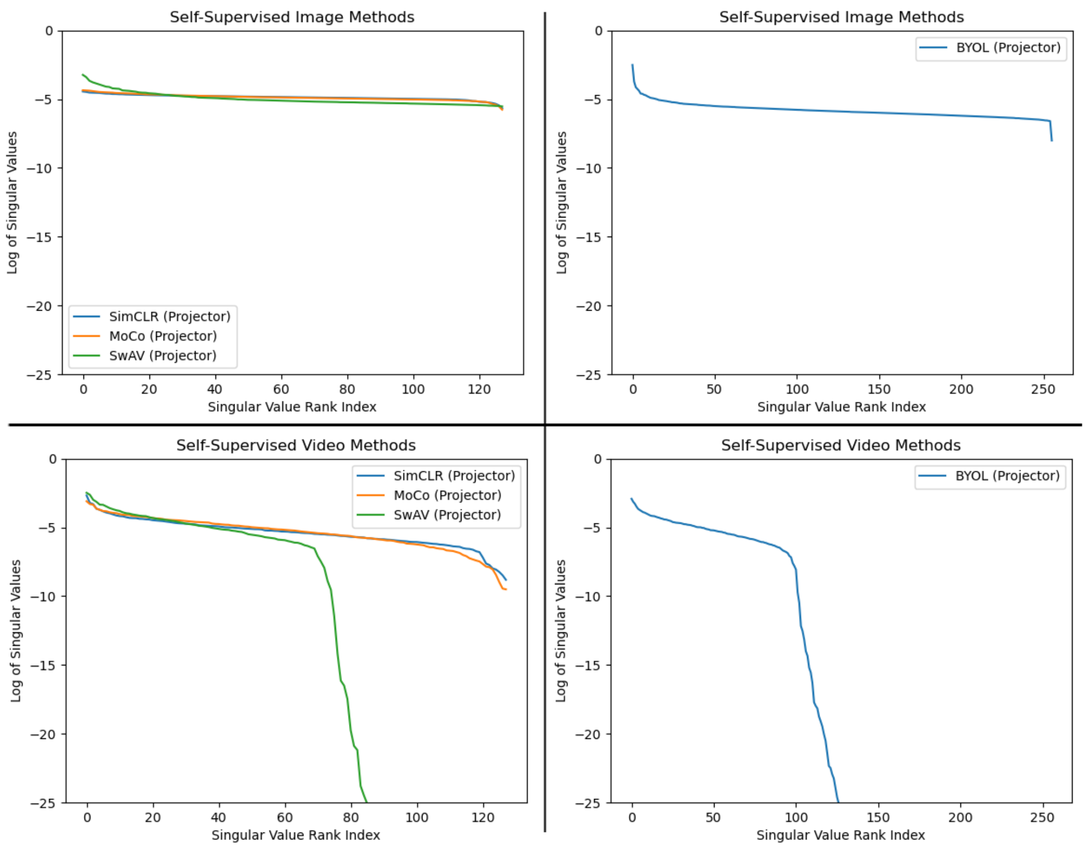


Figure 4.11: Top Left: Spectrum plots of embeddings of image-based SimCLR, MoCo and SwAV. **Top Right**: Spectrum plot of embeddings of image-based BYOL. **Bottom Left**: Spectrum plots of embeddings of video-based SimCLR, MoCo and SwAV. **Bottom Right**: Spectrum plot of embeddings of video-based BYOL.

BYOL and SwAV. The top row displays the plot of the image-based versions and the bottom row displays the plot of the video-based versions. Additionally, since the output dimensionality of BYOL is different than for the rest, its spectrum has been illustrated in a separate plot. First looking at the image spectra in the top row, no collapse can be detected. The spectra appear to be even straighter than any plot inspected so far. On contrast, the video spectra paint another picture. The SimCLR and MoCo spectra bear similarities but the BYOL and SwAV spectra behave differently. Both embedding spaces of BYOL and SwAV experience dimensional collapse. Similar to the information maximization methods, the dimensional collapse takes place at roughly the halfway point. Most surprisingly, only the video-based BYOL and SwAV approaches experience dimensional collapse. In the image domain, none of the two suffer from dimensional collapse. The obvious difference between SimCLR/MoCo and BYOL/SwAV is the use of BN layers in the projection head, as seen in Table 4.2. Furthermore, SimCLR and MoCo are contrastive approaches. Why BYOL and SwAV only experience dimensional collapse in their video implementations is more difficult to interpret. The use of bias weights in their respective projection heads is inconsistent, but that seems like an unlikely solution. Jing et al. [1] also postulate too strong data augmentations as a possible reason for dimensional collapse. SwAV utilizes multi-crop augmentation in the image-based version but does not do so in the video-based version. Otherwise, all implementations seem to follow the SimCLR proposed strategy [10] for data augmentations. For now, a conclusive reasoning for these findings can not be provided. At least, it can again be shown that dimensional collapse in the embedding space does not negatively impact its respective representation space.

4.2.4 Generative Approaches

So far, for video representation learning, we have only examined medium-independent self-supervised methods. None of BYOL, SimCLR, SwAV or MoCo are specialized methods towards the characteristics of videos but rather are generalized learning frameworks. For the remaining approaches, we will analyze methods which make use of video properties to enable supervision.

Generative approaches, for instance, learn the semantic of videos by attempting to generate a likely future state. The partition of a video into present and future state serves as supervision during training. The generative approach examined in this section is Dense Predictive Coding (DPC) [17] as introduced in section 2.4.2 of the literature review. The goal is to determine the potential of dimensional collapse in the representation space. Figure 2.21 exemplifies again the setup of DPC. The representations are obtained from the context representation volume c_t after being passed through an average pooling layer. The final dimensionality of the representation vector is 256. As seen in Table 2.4, DPC performs the worst amongst the inspected video models, but also utilizes a backbone of less capacity than the previous Slow-only architecture of 50 layers. Here, the pretrained model of DPC, used to examine dimensional collapse, consists of a 3D ResNet-34 backbone encoder, similar to [60], and has been pretrained as usual on the Kinetics-400 dataset.

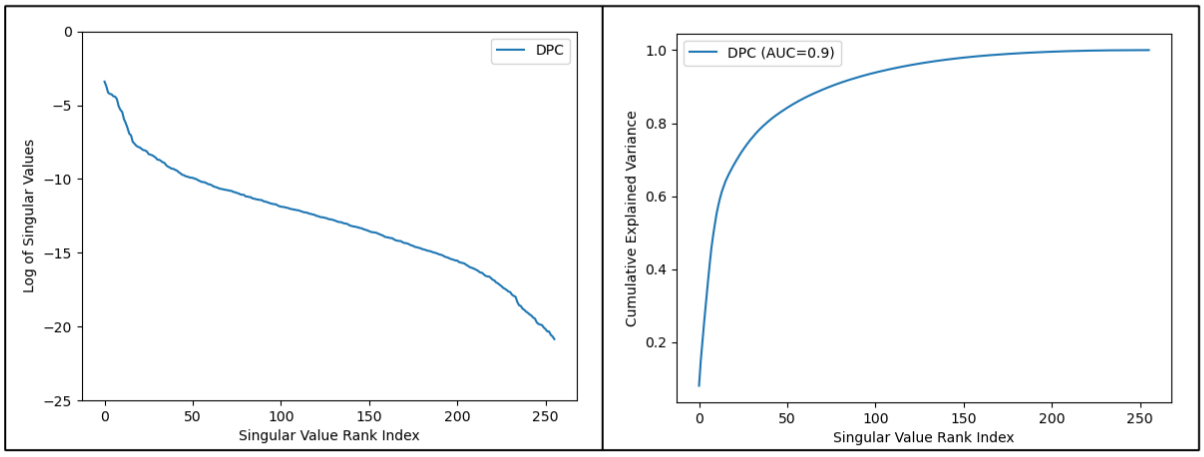


Figure 4.12: Left: Spectrum plot of DPC. **Right**: CEV plot of DPC.

Figure 4.12 shows the singular value spectrum and CEV curve of the mentioned DPC model. The representations for the analysis have been obtained on the UCF-101 dataset. On the left side of Figure 4.12, the spectrum is plotted. It shows a noticeable decay. No sudden collapse is depicted like, for example, in the projector spectrum of VICReg, as seen in Figure 4.8, but certainly a deterioration is evident. Additionally, to the right, the CEV plot presents a fast rising curve and remarkably large AUC value of 0.9, that would be described as a large degree of collapse. In all likelihood, DPC suffers from dimensional collapse. However, its indications for dimensional collapse are different. Naturally, there exist degrees to dimensional collapse as stated by Li et al. [20]. Nonetheless, there remains to be a clear difference between smothering dimensional collapse like observed here for the representation space of DPC, and abrupt dimensional collapse as experienced for the embedding space of BarlowTwins/VICReg, as shown in Figure 4.8, and video-based SwAV/BYOL, as illustrated in Figure 4.11. So far, except for parts of the SlowFast architecture, no dimensional collapse is apparent in the representation space of the inspected video representation learning models. Having said that, DPC, as a first, shows a overarching deficiency in its representation space. Although, the video medium offers with future prediction one of the most intuitive ways of non-labelled supervision, DPC fails at providing a representation space that spans the whole space available.

4.2.5 Cross-Modal Approaches

At last, we will extend our repertoire once more by embracing one intrinsic modality of the video medium. Both datasets, Kinetics-400 and UCF-101, sourced their videos from the video-platform YouTube. Thus, besides only visual frames, these videos also contain an audio track. Cross-model self-supervised approaches utilize this interplay between different modalities for supervision. The video representation learning approach explored in this section is Cross-Modal Deep Clustering (XDC) [18], as introduced in section 2.4.3 of the literature review. XDC encodes and clusters visual frames and audio signals, separately, and uses the opposite modalities' cluster assignments as supervision signal for optimization. The separate backbone encoders for video features and audio features are a R(2+1)D-18 network and a 2D ResNet-18 network, respectively. Thus, a video encoder and audio encoder, which could be used for video and audio downstream tasks, are trained simultaneously. Both encoder architectures have been discussed prior. To note, between the two, we will only analyze the self-supervised trained R(2+1)D-18 network, that learned video representations, for dimensional collapse. Additionally, the representations of a supervised trained R(2+1)D-18 network will be examined for comparison's sake.

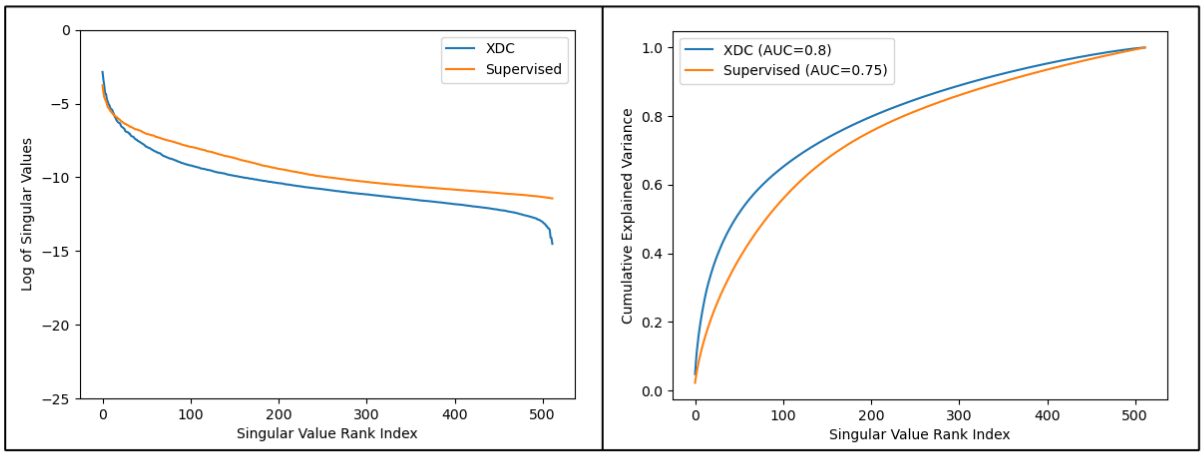


Figure 4.13: Left: Spectrum plot of XDC and supervised counterpart. **Right**: CEV plot of XDC and supervised counterpart.

Figure 4.13 shows the singular value spectrum plots and CEV plots for the XDC pretrained model and the model trained with labelled supervision. For both representations sets, the same type of architecture was used. Apart from that, the models differentiate themselves in one important way. As seen in Table 2.4, the XDC model was trained without labels on the extremely large IG-Curated dataset, while the supervised model was trained on the Kinetics-400 dataset. Besides that, the representations are obtained on the UCF-101 dataset. Let us start with the singular value spectrum plot. Both models share similar behaviour, the spectra are slightly declining without the strong decay of DPC, seen in Figure 4.12, but also without the stability of the information maximization methods, shown in Figure 4.7. More so, the CEV plot shows that the degree of collapse for the XDC model is lower than for DPC, which was indicated as dimensional collapse, but also larger than for the video approaches discussed in section 4.2.3. Additionally, the supervised model expresses lower AUC levels than the XDC model but a direct comparison is somewhat flawed because of the different training datasets used for both models. The inclusion of the supervised counterpart serves more for general contrast. While the behaviour of XDC would not be ruled as dimensional collapse, the degree of collapse is somewhat larger than for other previously inspected video representation learning methods.

5 Conclusion

When encountering complete collapse in representation learning, a deficiency in the approach is immediately noticed. The optimization problem is solved but the actual intention behind the problem, the visual learning process, has not been achieved. Preventing this complete collapse might suggest complete success but the representation space is not that binary. Introducing adapted methods, which clear the initial hurdle of complete collapse, can appear satisfactory and sufficient. Therefore, research into further collapse issues has so far been mostly ignored [19].

In this thesis, we have explored the collapse issue of dimensional collapse in visual data, where only a lower-dimensional subspace of the representation space is utilized [1]. The recent inception of examination of dimensional collapse in image representation learning was extended and the exploration into dimensional collapse of video representation learning has been started. Fortunately, strong dimensional collapse is not a frequent occurrence in state-of-the-art video representation spaces, as witnessed in the exploration chapter 4. However, it is also not an unlikely occurrence either. Like demonstrated by Dense Predictive Coding (DPC) in section 4.2.4 of the exploration, there still exists a wide margin between abrupt collapse and optimal utilization of the representation space.

The exploration of dimensional collapse offers additional insights that can either support standing hypothesis or inspire new ones. Self-supervised approaches like SwAV have shown that supervised training is not the gold standard for learning visual feature representations but that representations learned through self-supervision can prevail in many transfer tasks [12]. Through examination of dimensional collapse, we have shown that many recent state-of-the-art self-supervised methods are also robuster towards dimensional collapse and take better advantage of the full representation space than their supervised counterparts. Furthermore, such findings build credence that self-supervised representations can learn more valuable and intrinsic features than supervised representations do through human assistance.

Evidently, from the experiments conducted in chapter 4, dimensional collapse occurs most often in the embedding space of a projector. On the other hand, every time a projection head has been used, so far, no dimensional collapse was detected in its preceding representation space. These findings coincide with the hypothesis of Jing et al. [1] that projection heads prevent dimensional collapse. Almost like a protective layer, the projector guards the feature representations. Although, this still leaves the question, how to categorize dimensional collapse in the embedding space. Chen et al. [10] first introduced the addition of a non-linear projection head to boost the representation quality of the downstream performance, but the projector also appears to boost the representation utility. The projection head is a compelling and important fragment whose role has not been fully unveiled.

Finally, in its simplest form, dimensional collapse can also serve as a sanity check or metric to verify established methods in representation learning, that otherwise might seem intuitive in their effectiveness.

A Appendix

A.1 DenseNet

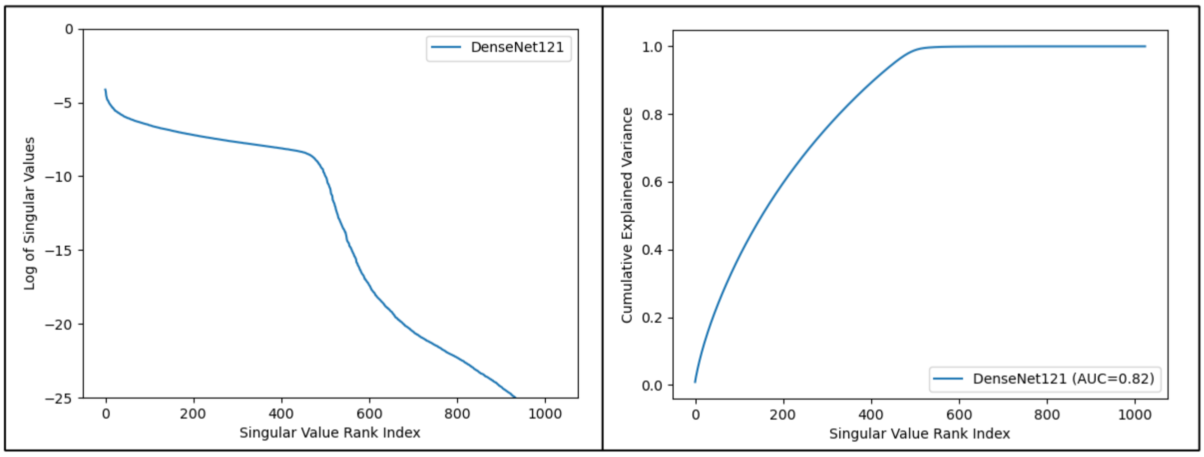


Figure A.1: Left: Spectrum plot of DenseNet121. **Right**: CEV plot of DenseNet121.

DenseNet [80] is a deep convolutional neural network architecture for images. Figure A.1 shows dimensional collapse in its representation space. Here, the pretrained model [81] was trained on ImageNet and the representations obtained on the ImageNet validation set. While very intriguing, this model was not included in the principal thesis because it was explored very late in the process and no time was left for a proper review of the paper. Still, it provides evidence for the possibility of dimensional collapse in supervised trained image models.

A.2 CVRL

CVRL [71] is a self-supervised video representation learning method with parallels to SimCLR. Figure A.2 presents a comparison with video-based SimCLR and the supervised trained Slow-only architecture. The used model [82] was pretrained on Kinetics-400 and the representations obtained on UCF-101. CVRL introduces temporally and spatially consistent augmentations for videos. It was not included in the principal thesis because it did not add much of new relevant information.

A.3 Transformers

Transformers have gained massively in popularity over the recent years and even catched the attention of the public eye with applications like ChatGPT. More so, transformers entered

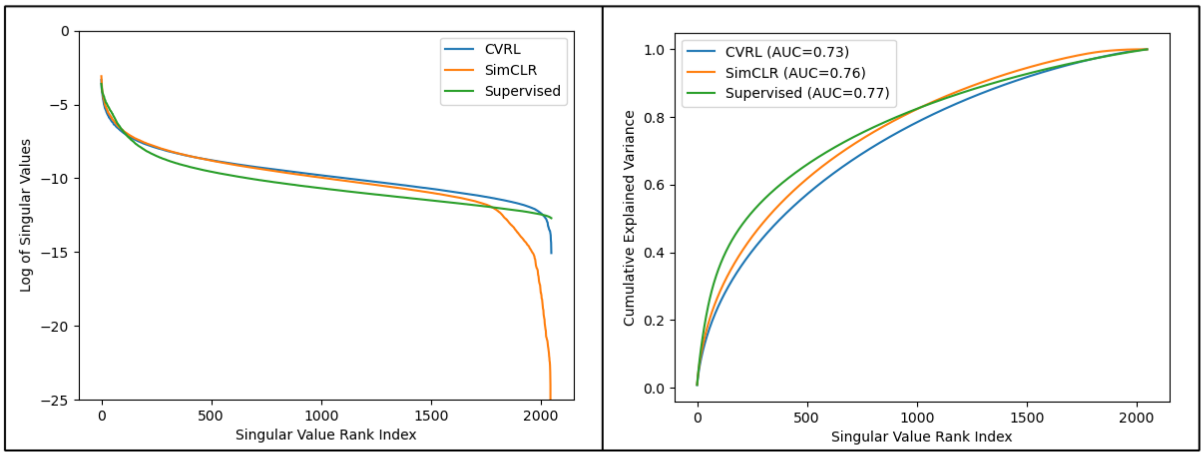


Figure A.2: Left: Spectrum plots of CVRL, video-based SimCLR and supervised Slow-only. **Right**: CEV plots of CVRL, video-based SimCLR and supervised Slow-only.

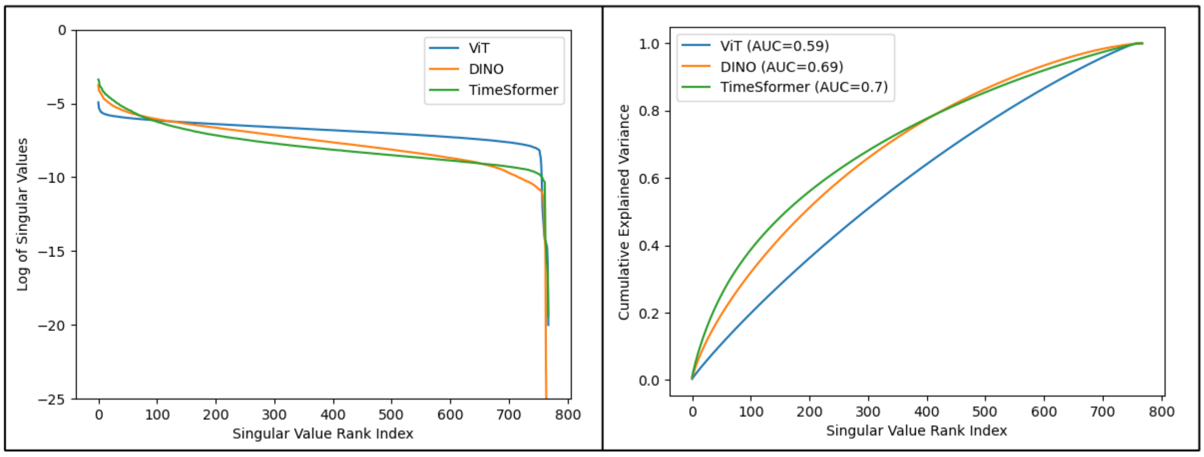


Figure A.3: Left: Spectrum plots of ViT-B/16, DINO-ViT-B/16 and TimeSformer. **Right**: CEV plot of ViT-B/16, DINO-ViT-B/16 and TimeSformer.

the realm of computer vision and established many state-of-the-art architectures [83]. Figure A.3 displays singular value spectra and CEV curves of supervised and self-supervised methods utilizing transformer backbones. Noticeably, all appear very stable to dimensional collapse. The Vision Transformer (ViT, here ViT-B/16) [83] is a image backbone transformer architecture, TimeSformer [84] is a video backbone transformer architecture and DINO [85] is a self-supervised approach, which in this instance utilizes a ViT-B/16 backbone. A section about transformers in video representation learning was not included in the principal thesis, because with the remaining time left, it felt not possible to do it justice.

Bibliography

[1] Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding dimensional collapse in contrastive self-supervised learning, 2022.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[4] Zhiqiang Liu, Paul Chow, Jinwei Xu, Jingfei Jiang, Yong Dou, and Jie Zhou. A uniform architecture design for accelerating 2d and 3d cnnson fpgas. *Electronics*, 8(1), 2019.

[5] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition, 2018.

[6] TensorFlow. Video classification with a 3d convolutional neural network. `https://www.tensorflow.org/tutorials/video/video_classification`. [Online; accessed October 19, 2023].

[7] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition, 2019.

[8] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles, 2017.

[9] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations, 2018.

[10] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020.

[11] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning, 2020.

[12] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments, 2021.

[13] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning, 2020.

[14] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction, 2021.

[15] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning, 2022.

[16] Christoph Feichtenhofer, Haoqi Fan, Bo Xiong, Ross Girshick, and Kaiming He. A large-scale study on unsupervised spatiotemporal representation learning, 2021.

[17] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding, 2019.

[18] Humam Alwassel, Dhruv Mahajan, Bruno Korbar, Lorenzo Torresani, Bernard Ghanem, and Du Tran. Self-supervised learning by cross-modal audio-video clustering, 2020.

[19] Tianyu Hua, Wenxiao Wang, Zihui Xue, Sucheng Ren, Yue Wang, and Hang Zhao. On feature decorrelation in self-supervised learning. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9578–9588, 2021.

[20] Alexander C. Li, Alexei A. Efros, and Deepak Pathak. Understanding collapse in non-contrastive siamese representation learning, 2022.

[21] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives, 2014.

[22] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[23] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[24] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[25] PyTorch. Alexnet. `https://pytorch.org/vision/stable/models/generated/torchvision.models.alexnet.html#torchvision.models.alexnet`. [Online; accessed October 17, 2023].

[26] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks, 2014.

[27] PyTorch. Resnet50. `https://pytorch.org/vision/stable/models/generated/torchvision.models.resnet50.html#torchvision.models.resnet50`. [Online; accessed October 17, 2023].

[28] Dorado list. 21 most cited papers in machine learning. `https://www.doradolist.com/papers/most-cited-papers-in-machine-learning.html`. [Online; accessed October 13, 2023].

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[30] P. Read and M.P. Meyer. *Restoration of Motion Picture Film*. Butterworth-Heinemann Series in Conservation and Museology. Elsevier Science, 2000.

[31] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and

Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[32] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks, 2017.

[33] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset, 2017.

[34] PyTorch. 3d resnet. `https://pytorch.org/hub/facebookresearch_pytorchvideo_resnet/`. [Online; accessed October 17, 2023].

[35] PyTorchVideo. Support pytorchvideo in pyslowfast. `https://github.com/facebookresearch/SlowFast/tree/main/projects/pytorchvideo`. [Online; accessed October 17, 2023].

[36] PyTorch. Slowfast. `https://pytorch.org/hub/facebookresearch_pytorchvideo_slowfast/`. [Online; accessed October 17, 2023].

[37] facebookresearch. Vissl model zoo and benchmarks. `https://github.com/facebookresearch/vissl/blob/main/MODEL_ZOO.md`. [Online; accessed October 21, 2023].

[38] facebookresearch. Moco: Momentum contrast for unsupervised visual representation learning. `https://github.com/facebookresearch/moco`. [Online; accessed October 22, 2023].

[39] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning, 2020.

[40] facebookresearch. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. `https://github.com/facebookresearch/vicreg`. [Online; accessed October 22, 2023].

[41] facebookresearch. Barlow twins: Self-supervised learning via redundancy reduction. `https://github.com/facebookresearch/barlowtwins/tree/main`. [Online; accessed October 22, 2023].

[42] deepmind research. Bootstrap your own latent. `https://github.com/google-deepmind/deepmind-research/tree/master/byol`. [Online; accessed October 22, 2023].

[43] facebookresearch. Unsupervised learning of visual features by contrasting cluster assignments. `https://github.com/facebookresearch/swav`. [Online; accessed October 22, 2023].

[44] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization, 2016.

[45] Michael Tschannen, Josip Djolonga, Paul K. Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning, 2020.

[46] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019.

[47] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman.

The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, jun 2010.

[48] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[49] Agrim Gupta, Piotr Dollár, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation, 2019.

[50] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding, 2016.

[51] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining, 2018.

[52] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transportation distances, 2013.

[53] TengdaHan. Video representation learning by dense predictive coding. `https://github.com/TengdaHan/DPC`. [Online; accessed October 31, 2023].

[54] facebookresearch. A large-scale study on unsupervised spatiotemporal representation learning. `https://github.com/facebookresearch/SlowFast/tree/main/projects/contrastive_ssl`. [Online; accessed October 31, 2023].

[55] HumamAlwassel. Self-supervised learning by cross-modal audio-video clustering. `https://github.com/HumamAlwassel/XDC`. [Online; accessed November 3, 2023].

[56] Deepti Ghadiyaram, Matt Feiszli, Du Tran, Xueting Yan, Heng Wang, and Dhruv Mahajan. Large-scale weakly-supervised pre-training for video action recognition, 2019.

[57] Chunhui Gu, Chen Sun, David A. Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, Cordelia Schmid, and Jitendra Malik. Ava: A video dataset of spatio-temporally localized atomic visual actions, 2018.

[58] Gunnar A. Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding, 2016.

[59] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzyńska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "something something" video database for learning and evaluating visual common sense, 2017.

[60] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?, 2018.

[61] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence*

and Statistics, volume 9 of *Proceedings of Machine Learning Research*, pages 297–304, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[62] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features, 2019.

[63] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild, 2012.

[64] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: A large video database for human motion recognition. In *2011 International Conference on Computer Vision*, pages 2556–2563, 2011.

[65] Karol J. Piczak. ESC: Dataset for Environmental Sound Classification. In *Proceedings of the 23rd Annual ACM Conference on Multimedia*, pages 1015–1018. ACM Press.

[66] Lei Huang, Dawei Yang, Bo Lang, and Jia Deng. Decorrelated batch normalization, 2018.

[67] Anthony J. Bell and Terrence J. Sejnowski. The “independent components” of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338, 1997.

[68] Lars Eldén. *Matrix Methods in Data Mining and Pattern Recognition.* Society for Industrial and Applied Mathematics, 2007.

[69] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning, 2020.

[70] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[71] Rui Qian, Tianjian Meng, Boqing Gong, Ming-Hsuan Yang, Huisheng Wang, Serge Belongie, and Yin Cui. Spatiotemporal contrastive video representation learning, 2021.

[72] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[73] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[74] K. B. Petersen and M. S. Pedersen. The matrix cookbook, nov 2012. Version 20121115.

[75] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and

Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[76] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[77] PyTorch. Resnet. `https://pytorch.org/vision/stable/models/resnet.html`. [Online; accessed November 12, 2023].

[78] Grégoire Mialon, Randall Balestriero, and Yann LeCun. Variance covariance regularization enforces pairwise independence in self-supervised representations, 2022.

[79] Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In Sanjay Jain, Hans Ulrich Simon, and Etsuji Tomita, editors, *Algorithmic Learning Theory*, pages 63–77, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[80] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2018.

[81] PyTorch. Densenet121. `https://pytorch.org/vision/stable/models/generated/torchvision.models.densenet121.html#torchvision.models.densenet121`. [Online; accessed November 18, 2023].

[82] Rui Qian. Spatiotemporal contrastive video representation learning. `https://github.com/tensorflow/models/tree/master/official/projects/video_ssl`. [Online; accessed November 18, 2023].

[83] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[84] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *Proceedings of the International Conference on Machine Learning (ICML)*, July 2021.

[85] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers, 2021.