Contrastive Learning of Visual Representations from Unlabeled Videos

Simion-Constantinescu F. Andrei

Contrastive Learning of Visual Representations from Unlabeled Videos

by

Simion-Constantinescu F. Andrei

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday August 25, 2020 at 3:00 PM.

Student number:	4915739	
Project duration:	November, 2019 – August, 2020	
Thesis committee:	Dr. J.C. van Gemert,	TU Delft, Supervisor & Chair
	Dr. M. Loog,	TU Delft, Core Member
	Dr. N. Tintarev,	TU Delft, External Core Member
	O.S. Kayhan,	TU Delft, PhD Student

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Preface

The current report documents the research in self-supervised learning on videos for my Master Thesis to obtain the Master of Science degree from Delft University of Technology. My work was conducted inside the Pattern Recognition and Bioinformatics research group, under the supervision of Dr. Jan van Gemert, head of the Computer Vision Lab. This document is structured in two parts, a scientific article presenting our novel approach with contrastive learning of human action video features using unlabeled data and supplemental material that provides the background of the thesis.

My first encounter with the Netherlands was in the summer of 2016, when I had an internship at Royal Dutch Shell. Being fascinated by the country and knowing the esteem reputation of TU Delft, my choice for Master studies was straight forward. During the two years of my Master studies, I had to adhere to TU Delft high research standards, gaining a lot of essential Data Science knowledge, both theoretically and practically. My passion for Deep Learning grew further after following Jan van Gemert courses, given his exceptional teaching skills. This resulted in choosing him as my mentor for my Master Thesis.

First, I would like to express my sincerely gratitude to Jan van Gemert for his guidance and useful feedback during the elaboration of this project. Secondly, I would like to point out my appreciation to one of Dr. van Gemert's PhD student, Osman, who acted as my daily supervisor, helping me to overcome all obstacles during my research. Moreover, the support provided by my parents, both financially and emotionally, was essential for my success. Finally, I would like to express my appreciation to the others members of the Thesis Committee, Dr. Loog and Dr. Tintarev, for the interest in my work and for accepting to be a part of my Evaluation Committee.

Simion-Constantinescu F. Andrei
Delft, August 2020

Contents

1 Scientific article **1**

2 The Basics of Deep Learning **13**
2.1 Feed-forward Networks . . . 13
2.1.1 Multi-layered perceptron . . . 13
2.1.2 Activation function . . . 13
2.2 Optimization . . . 14
2.2.1 Stochastic gradient descent . . . 14
2.2.2 Cosine Annealing . . . 15
2.3 Convolutional Neural Networks . . . 16
2.3.1 Convolution operator . . . 16
2.3.2 Kernels as feature detectors . . . 17
2.3.3 Spatial Pooling . . . 17

3 Action recognition **19**
3.1 Common Deep Neural Network architectures for videos . . . 19
3.1.1 Two Stream Networks . . . 19
3.1.2 Long-Term Recurrent Convolutional Networks . . . 20
3.1.3 3D Convolutional Networks . . . 20
3.1.4 3D Residual Networks . . . 21
3.1.5 3D-Fused Two-Stream Networks . . . 21
3.1.6 Two Stream I3D . . . 22
3.2 Action recognition datasets . . . 22
3.2.1 HMDB51 . . . 22
3.2.2 UCF101 . . . 22
3.2.3 ActivityNet . . . 22
3.2.4 Kinetics . . . 22
3.2.5 Moments in Time . . . 23

4 Self-supervised learning **25**
4.1 Learning visual representations from images . . . 25
4.1.1 Generative based methods . . . 25
Generative Adversarial Networks . . . 25
Bidirectional Generative Adversarial Networks . . . 25
Context Encoders . . . 26
Image colorization . . . 26
4.1.2 Relationships between patches . . . 27
Relative position . . . 27
Jigsaw puzzles . . . 27
Feature counting . . . 27
4.1.3 Geometric transformation based methods . . . 27
Exemplar Convolutional Neural Networks . . . 27
Predicting image rotations . . . 28
4.1.4 Contrastive learning based methods . . . 29
Contrastive Predictive Coding . . . 29
Momentum Contrast . . . 29
SimCLR . . . 30
Improved Momentum Contrast . . . 30

4.2 Learning visual representations from videos . . . 31
4.2.1 Generative based methods . . . 31
Video Generative Adversarial Networks . . . 31
Video colorization . . . 31
4.2.2 Relationship between frames . . . 31
Space-Time Cubic Puzzles . . . 31
Temporal Order Verification . . . 31
Temporal Order Prediction . . . 31
Arrow of Time . . . 32
4.2.3 Geometric transformation based method . . . 33
Predicting video rotations . . . 33
4.2.4 Contrastive learning based methods . . . 33
Dense Predictive Coding . . . 33
Temporally Coherent Embeddings . . . 34

List of Figures **35**

Bibliography **37**

1

Scientific article

Contrastive Learning of Visual Representations from Unlabeled Videos

Andrei Simion-Constantinescu
Delft University of Technology
Delft, The Netherlands
andrei.simion.c@gmail.com

Osman Semih Kayhan (Supervisor)
Delft University of Technology
Delft, The Netherlands
o.s.kayhan@tudelft.nl

Jan C. van Gemert (Supervisor)
Delft University of Technology
Delft, The Netherlands
j.c.vanGemert@tudelft.nl

***Abstract*—This paper presents a novel self-supervised approach of learning visual representations from videos containing human actions. Our approach tackles the complex problem of learning without the need of labeled data by exploring to what extent the ideas successfully used for images can be transferred, adapted and extended to videos for action recognition purposes. We begin by giving a brief introduction to the topic of learning features without having access to a labeled corpora, providing the motivation of our work. We continue with presenting the related research in terms of contrastive learning, action recognition from videos with 3D convolutions and self-supervised techniques for both images and videos. Next, we formalize our approach with regards to the sampling method, the types of spatial and temporal transformations and the contrastive loss used. We evaluate videoSimCLR proposed method in terms of linear evaluation, fully fine-tuning and video retrieval. We also explore the extension of another contrastive learning approach to videos, videoMOCO, and compare it to videoSimCLR by means of linear evaluation.**

***Keywords* – self-supervision, contrastive learning, image-based, video-based, deep learning, neural networks, image classification, action recognition, video retrieval, video embedding**

1. Introduction

Convolutional Neural Networks (CNNs) achieve state-of-the-art performance on numerous Computer Vision tasks such as image classification [35], [53], [62], video action recognition [4], [11], [52], object detection [36], [48], [49], instance segmentation [7], [21], [29], etc. CNNs are the standard backbone architecture for most of Computer Vision domains since the revolution of AlexNet [32]. However, their success is highly dependent on having high-quality labeled data, which is not as largely available as unlabeled data. To take advantage of the exiting unlabeled corpora, self-supervised learning [64] techniques are explored. The annotation becomes harder as the complexity of the data increases, with a video is more time consuming and expensive to label compared to an image. This paper aims to tackle this issue by proposing a novel way of feature learning from videos without the need of human data annotation.

Self-supervised pre-training for learning visual representation from images yields very promising results. Methods such as MOCO [6] and simCLR [5] are at a competitive level with the fully supervised counterpart. Compared to images, videos introduce both the temporal dimension and additional modalities such as speech or audio. Cross-modal supervision takes advantage of the audio-visual relation in videos, proposing multi-modal self-supervision schemes [1], [39], [40], [45], [47]. This paper does not focus on the multi-modal learning, but uses a single modality, namely RGB values. We follow another general trend of adapting successful self-supervised ideas from image domain to videos.

Self-supervised video feature learning from RGB flow includes methods such as VideoGAN [56], Video colorization [57], Space-Time Cubic Puzzles [28], Arrow of Time [59], Temporal Order Verification [38], 3D RotNets [24], etc. All of this techniques are using loss functions from supervised learning domain, since they are creating pseudo-labels that can act as normal data annotations. Contrastive learning used for self-supervision is focused on teaching a model to promotes the similarity between relevant data points and to distinguish them from a bunch of distractors. Contrastive learning based methods for videos exists, with Dense Predictive Coding [17] and Temporally Coherent Embeddings [30] as a couple of examples, but this approaches are still indirectly enforcing the similarity by tasks such as predicting the future frames or selecting the temporally adjacent ones. Our paper presents a novel approach of adapting a couple of contrastive learning ideas from images to videos by directly enforcing the similarity between videos. Consider Figure 1, in which an overview of our videoSimCLR method is presented: having two clips we apply a random temporal transformation on each, followed by a random spatial transformation. We then use a contrastive loss to enforce the similarity between the two augmented versions of each clip and to promote the dissimilarity between the versions generated from different clips.

Our work brings the following contributions. First, we extend three state-of-the-art image based methods [5], [6], [20] for videos as videoSimCLR, videoMOCO V1 and V2. This contrastive learning methods from images are used on videos by applying random spatial transformations to the frames of two action recognition datasets, namely UCF101 [51] and HMDB51 [33]. Second, we propose adding novel temporal transformations to learn better video and motion representations. We experiment with multiple types of temporal transformations such as Shifting, Frame Dropping, Reversing and Shuffling. We find out that temporal transformations alone are not enough, with the combination of both temporal transformations and spatial ones being essential for good results.

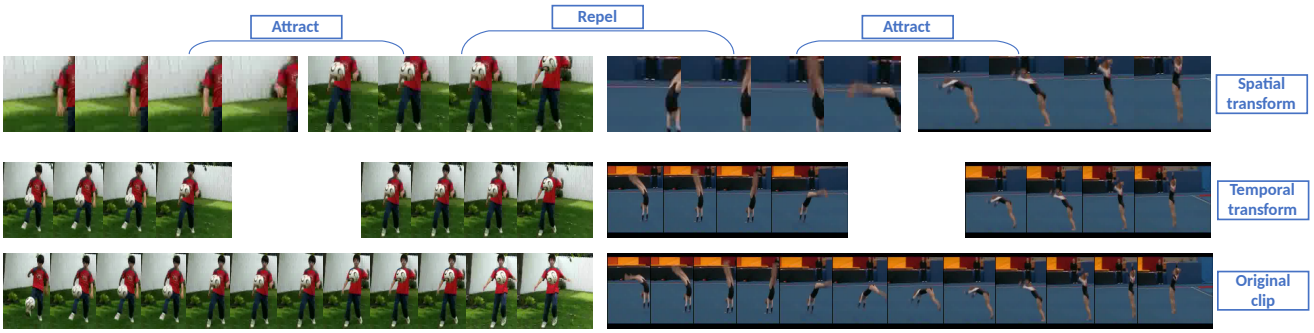

Figure 1: Overview of videoSimCLR method. Starting from two original clips sampled from two different videos, we apply a random temporal transformation on each clip to generate two shifted versions. On each of the shifted versions we further apply a random spatial transformation. At the end, we promote the attraction between the augmented versions from the same clips and we force the ones from different clips to repel each other. Videos taken from UCF101 action classes of Soccer Juggling and Floor Gymnastics.

Having the right combination, we obtain significantly better performance than fully supervised training. Third, we show the learnt representations with a video retrieval task and compare with fully supervised representations. The performance of our methods, videoSimCLR and videoMOCO V1/V2, is measured using the linear evaluation protocol, with the best videoSimCLR model being fine-tuned for comparison purposes with other self-supervised video techniques.

2. Related Work

Action recognition with 3D Residual Network The choice of the backbone architecture used to encode the data to a latent space is important. 3D CNNs [4], [18], [23], [54] are capable to perform the convolution operation on the whole video volume. Motivated by the ability of directly extracting spatio-temporal features, an extension of the 2D residual networks [22] is commonly employed for action recognition when using only the RGB stream. 3D ResNets [19] are proposed in various numbers of layers, from 18 to 200. In our work, we have employed a 3D Resnet18 as our backbone architecture due to training being on small datasets to prevent overfitting. More background information about CNNs can be found in *Chapter 2, The Basics of Deep Learning*. The subject of video action classification is presented in detail on *Chapter 3, Action recognition*.

Contrastive learning. Contrastive learning is a type of deep metric learning [25] which aims to learn a function for measuring the similarity between a pair of data points. The function used to distinguish between similar and dissimilar data examples is called a contrastive loss. There are several contrastive loss functions: Siamese loss [16], Triplet loss [60], Multi-class N-pair loss [50], Supervised NT-Xent loss [27], etc. Self-supervised learning [64] is a type of unsupervised learning which uses a pretext task, also called self-supervised task, to generate pseudo-labels from unlabeled data. By optimizing to solve a pretext task which exploits the intrinsic relationship existent in the data, visual representations can be learned. The optimization can be done using a contrastive loss. Our work uses contrastive learning as a way of creating pseudo-labels without defining an explicit pretext task.

Self-supervised image features learning. The self-supervised techniques for image feature learning can be classified based on the type of the pretext task. The generative based methods has the main goal to create realistic images, meaningfully visual representations being learned as a side effect: Denoising auto-encoders [55], Generative Adversarial Networks [14], Bidirectional GANs [9], Context Encoders [46], Image colorization [63]. The next class of methods exploits the relationship between patches such as predicting the relative position [8], solving Jigsaw puzzles [42] and learning to count features [43]. Another handcrafted pretext task category is geometric transformation based with Exemplar CNNs [10] or predicting the image rotation [13]. Finally, contrastive losses are used for the next group of techniques with Contrastive Predictive Coding [44] being inspired by the Noise Contrastive Estimation used in learning word embeddings [15]. Our work is inspired by two state-of-the-art contrastive based self-supervised methods, namely Momentum Contrast (MOCO) [20] and a Simple Framework for Contrastive Learning of Visual Representations (simCLR) [5]. SimCLR uses contrastive learning to maximize agreement between 2 augmented versions of the same image. In MOCO V1 the number of negative examples is dependent on the queue size, where in SimCLR is related to the batch size. A recent study, MOCO V2 [6], integrates a couple of SimCLR ideas, such as stronger data augmentation and the MLP projection head, into the MOCO framework showing further improvements to the quality of the learned features. This paper takes simCLR and MOCO as starting points for our proposed methods, videoMOCO V1/V2 and videoSimCLR. A detailed review of self-supervised learning from images can be analyzed in *Chapter 4.1, Self-supervised learning*.

Self-supervised video features learning. When looking at approaches for self-supervised video feature learning, many methods are extending the image based version to videos. Similar types of pretext tasks are used: generative based with VideoGAN [56] and Video colorization [57], relationship between frames by solving Space-Time Cubic Puzzles [28], tracking the movement of an object [58], verifying [38] or predicting [34] the frame order, identifying odd-one clip [12] and predicting the Arrow of Time [59], geometric transformation based with 3D rotation networks [24]. More recently, contrastive learning is used for video features learning as well, with Dense Predictive Coding [17] by recurrently predicting future representations and Temporally Coherent Embeddings [30] by directly enforcing temporal coherency in

the embedding space. Following this general trend of extending the image-based methods to video and given the better results of the contrastive learning when compared to handcrafted pretext tasks, we have chosen to explore to what extent SimCLR and MOCO ideas can be applied on videos. The topic of video self-supervised learning is detailed in *Chapter 4.2, Self-supervised learning*.

3. Method

The pipeline of our videoSimCLR method can be analysed in Figure 2. Starting from an N-frames video, we sample a clip with a fixed number of frames. On the selected clip we apply two random temporal transformations to generate two versions of it: clip_i and clip_j. A random spatial transformation is then applied on each version of the selected clip to make $\overline{\text{clip}_i}$ and $\overline{\text{clip}_j}$. Finally, $\overline{\text{clip}_i}$ and $\overline{\text{clip}_j}$ are encoded with a function F composed of a sequence of CNNs followed by one or more fully connected layers (FC). A contrastive loss is used on the encoded versions of the augmented clips, $F(\overline{\text{clip}_i})$ and $F(\overline{\text{clip}_j})$, to promote the similarity between them.

Sampling method. Starting from videos with different number of frames, we need a sampling technique to generate clips with the same number of frames. As the sampling method, a temporal random crop have been used. We select m-consecutive frames starting from a random index. If the clip is shorter than m-frames, we loop until getting the desired number of frames. In the rest of the paper, we future refer this sampling method as *Random* m.

Temporal transformation. For generating two different clips from the sampled m-frames clip, we apply two random temporal transformations. We have 4 different transformations: temporal shifting, frame dropping, reversing the order of frames and shuffling the frames.

For the *Temporal shift*, we start with the m-frames clip and shift it using the original $N-$frames video with a random step s in a random direction, either backward or forward. We will refer it as *RandomShift [-s, s]*. A small example of the temporal shifting can be seen in Figure 3, with $N = 12$, $m = 4$ and $s = 4$.

When implementing the *Temporal Drop*, starting with a $2m$-frames video, we drop either the even or the odd index frames based on a random decision.

At the *Temporal reverse*, we simply take the original clip to be one of the versions and we reverse the frame order to create the other version, which one is clip_i or clip_j being randomly determined.

The *Temporal Shuffle* is creating the two versions of the m-frames clip based on two random permutations of the frame indexes.

Spatial transformation. We apply a sequence of spatial transformations on the frame level to obtain a different correlated view. The spatial transformation pipeline starts with a random cropping of the original frame followed by resizing to the desired frame size, continues with a random horizontal flip, some random color distortions such as color jittering and random grayscaling and finishes with a random Gaussian blur.

Base encoder F. For projecting the augmented clips to a latent space, we use a CNN architecture with 3D kernels, 3D Resnet18. Let C be the encoding function describing 3D Resnet18. Applied on $\overline{\text{clip}_i}$, the output will be $h = C(\overline{\text{clip}_i})$, $h \in \mathbb{R}^{512}$. From the CNN projection, we further propagate the data through a Multi-layer perceptron (MLP) with 2 hidden layers and ReLU [41] non-linearity in between. The final output is $F(\overline{\text{clip}_i}) = \text{MLP}(h) \in \mathbb{R}^{256}$.

Contrastive loss. Let $F(\overline{\text{clip}_i})$ be p_i and $F(\overline{\text{clip}_j})$ be p_j. The contrastive loss function used for videoSimCLR (1) is inspired by the normalized temperature-scaled cross entropy loss (NT-Xent) used in simCLR [5]:

$$\mathcal{L}_{i,j} = -\log \frac{\exp(\text{sim}(p_i, p_j)/\tau)}{\sum_{k=0}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(p_i, p_k)/\tau)} \quad (1)$$

where sim is the cosine similarity function, τ is the temperature parameter that controls the concentration level of the distribution [61] and $\mathbb{1}_{[k \neq i]}$ is $\{0, 1\}$ with 1 if $k \neq i$. For each batch of videos, positive and negative pairs have been created. Given a batch of size N, $2N$ augmented clips are generated forming N positive pairs. For each positive pair (p_i, p_j), $2(N-1)$ negative pairs are made from the rest of the augmented clips from the batch. NT-Xent loss promotes the similarity between two augmented version of the same clip, enforcing the augmented versions of the other clips to be dissimilar to it.

videoMOCO. Another way of viewing the contrastive learning task is as a dictionary look-up problem. Following MOCO [20] approach from images, a similar contrastive loss called InfoNCE (2) is adopted in our work:

$$\mathcal{L}_q = -\log \frac{\exp(\text{sim}(p_q, k_+)/\tau)}{\sum_{i=0}^{K} \exp(\text{sim}(p_q, k_i)/\tau)} \quad (2)$$

where sim is the similarity measured by dot product and k_+ is the positive key of the dictionary that p_q matches. The dictionary is a queue of size K which stores the negative examples k_i. There are two encoders, the query encoder and the momentum encoder: p_q is query_encoder(clip_q) and k_+ is momentum_encoder($\overline{\text{clip}_q}$), with $\overline{\text{clip}_q}$ being an augmented version of clip_q. The dictionary size needs to be adapted to the length of our datasets to limit the number of positives examples from the queue. Both the query encoder and momentum encoder are 3D Resnet18 based, with videoMOCO V1 using the CNN projection of the base encoder F followed by only one fully-connected layer. For videoMOCO V2, the same base encoder F as the one used by videoSimCLR is employed. In regards to the data transformation, videoMOCO V2 uses the same stronger spatial augmentation applied for videoSimCLR.

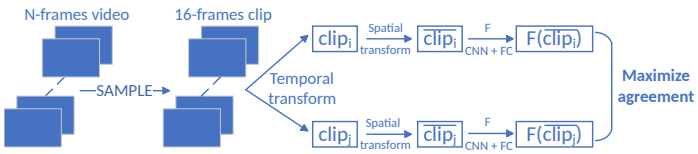

Figure 2: videoSimCLR simplified pipeline which starts with sampling a 16 frame clip from an N-frame video. A sequence of a random temporal transformation and a spatial random transformation is applied to generate two modified versions of the sampled clip, versions which are encoded using CNNs layers followed by FC ones. At the end, the agreement between $F(\overline{clip_i})$ and $F(\overline{clip_j})$ is enforced via a contrastive loss.

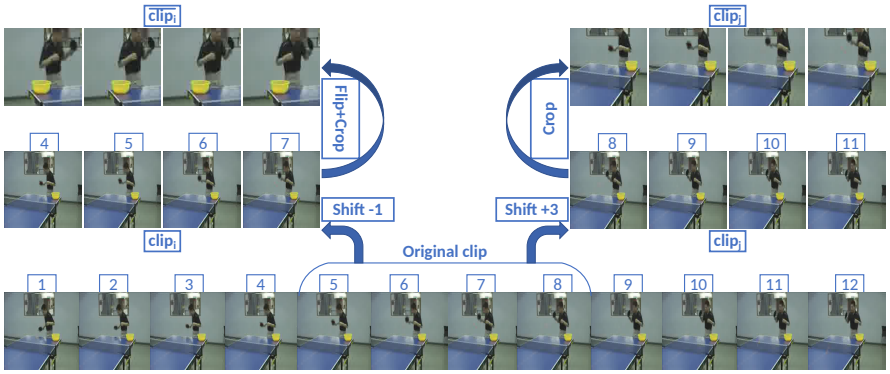

Figure 3: Temporal random shift example on a video of 12-frames, with a sampled clip of 4-frames using a shift step of 4. Starting with a 12-frame video, the original clip is sampled by taking the frames from indexes 5 to 8. Using a random temporal step of 4 frames, we shift the original clip in either backward or forward. $clip_i$ is generated by shifting backward with 1 frame (Shift -1) and $clip_j$ by shifting forward with 3 frames (Shift +3). After that, on both versions of the original clip a spatial random transformation is applied to generate $\overline{clip_i}$ and $\overline{clip_j}$. The example video is from UCF101 action class of Table Tennis Shot.

Evaluating the quality of the learned features. The quality of the learned features is measured by following the linear evaluation protocol, which is the most widely used method for evaluating the learned representations from images [2], [31]. To have a fair comparison with other self-supervised video based techniques, since most of them are reporting their results after fully fine-tuning, we need to fine-tune the whole network starting from our pre-trained weights. For better understanding the learned representation and visualize our results, we also perform a Nearest Neighbour video retrieval task [3]. Based on the $\mathbb{R}^{256}$ feature vector obtained for each video using videoSimCLR, the nearest neighbors of a query video are retrieved using cosine similarity.

4. Experiments

The models are pretrained on two popular yet small action recognition datasets, UCF101 and HMDB51, without using any labels. HMDB51 [33] is divided into 51 action categories with a minimum of 100 clips per category, having 3570 training and 1530 testing clips. UCF101 [51] is a larger action recognition dataset composed of 101 actions divided into 5 types: human-object interaction, body-motion only, human-human interaction, playing musical instruments and sports. It has 13320 videos split in 9537 training and 3783 testing clips.

All the following experiments are done by using 3D Resnet18 [18] architecture with only RGB features. Following the linear evaluation protocol, first, the pre-trained weights are used to initialize the 3D ResNet18 architecture. Afterwards, the weights are frozen to train a linear classifier on the top of the learned representations by using the training set. The classifier performance is evaluated by using the Top-1 accuracy (Acc@1) on the testing set. To note that, this protocol is followed for every training setup of the paper, apart from when comparing to other self-supervised video based techniques which requires fine-tuning the whole network.

For both self supervised pre-training and linear evaluation, we resize the video frames from 320x240 to the standard size of 224x224. The linear classifier is a l2-regularized multinomial logistic regression [5]. For pre-training, we use a batch size of 128 and we decay the learning rate using cosine decay schedule [37]. All of the other configurations are chosen based on an ablation study presented in Section 5.

Pre-training method	Pre-training on	UCF Acc@1	HMDB Acc@1
videoSimCLR Temp. Shift	UCF	**44.4%**	**21.8%**
videoSimCLR Temp. Shift	HMDB	35.9%	20.1%
videoMOCO V1 K=8192	UCF	40.3%	19.7%
videoMOCO V2 K=8192	UCF	41.4%	20.5%
videoMOCO V1 K=1024	HMDB	32.6%	16.7%
videoMOCO V2 K=1024	HMDB	34.2%	18.4%
Random init	-	20.6%	10.2%
Supervised	-	37.4%	14.7%

Table 1: Linear evaluation results for videoSimCLR, videoMOCO V1 and V2 compared to supervised pre-training or random weights. Meaningful features for action recognition are learned without using the labels, the size of the pre-training dataset being important for the quality of the learned visual representations. videoMOCO uses a different queue size K dependant on the size of the pre-training dataset. videoMOCO V2 integrates stronger data augmentation and a projection head with two fully-connected layers into the videoMOCO V1 framework. The three proposed pre-training methods are superior to supervised pre-training by using the labels.

4.1. Evaluating videoSimCLR and videoMOCO learned visual representations

We assess the quality of the learned feature when pre-training on HMDB51 and UCF101 using our first proposed self-supervised method, videoSimCR. We also evaluate our additional method, videoMOCO, in both V1 and V2. The number of epochs for pre-training was 100 for videoSimCLR and 200 for videoMOCO V1/V2. In Table 1, all of the results are summarized, with the best Top-1 linear evaluation accuracy being for videoSimCLR pre-trained on UCF101. The cross testing is used to further show the generalization capabilities of our models. Similar to when applied to images, videoMOCO V2 performs better compared to V1. When pre-trained on UCF101, videoMOCO V2 gives a 1.1% improvement tested on UCF101 and 0.8% on HMDB51. With HMDB51 as pre-training dataset, videoMOCO V2 has 1.6% accuracy improvment on UCF101 and 1.7% on HMDB51. With both videoSimCLR and videoMOCO, pre-training on the larger dataset, namely UCF101, is better when evaluating on the two datasets.

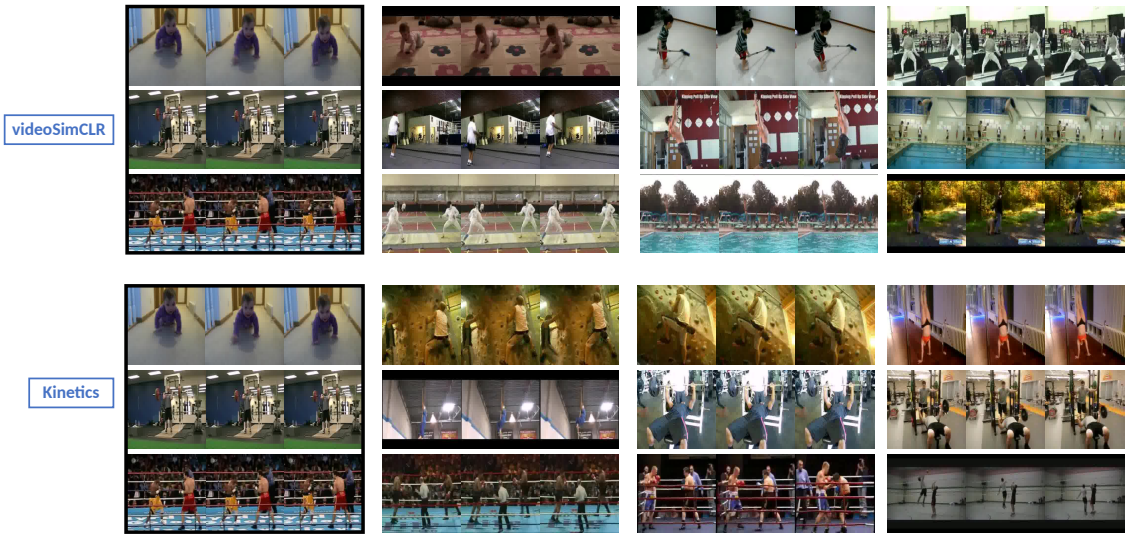

Figure 4: Examples of Nearest Neighbor video clip retrieval on UCF101 testing set. The leftmost column shows 3 frames from each of the query clip. Given the same query clips and based on both videoSimCLR and Kinetics video clip generated embeddings, the 3 nearest neighbors according to cosine distance are retrieved. The last 3 columns contain 3 frames from each of the retrieved clip. The top rows are using videoSimCLR generated embeddings and the bottom rows Kinetics ones. The first rows of both methods show that videoSimCLR is capable on surpassing Kinetics pre-trained, with the retrieved video of the first being from the right Baby Crawling class and the ones from the second having only a similar class to Crawling, Rock Climbing. The other rows shows that overall Kinetics pre-trained generated embeddings are better, but the videoSimCLR disagreements in terms of class are influenced by other relevant semantic cues, with few exceptions in the third row.

Moreover, our pre-trained weights generated by both self-supervised methods, videoSimCLR and videoMOCO, are an excellent starting point for training a classifier. When compared to random initialization and supervised pre-training, our methods surpass with more than 20% on UCF101 and 10% on HMDB the first initialization method and with over 4% on UCF101 and 5% on HMDB51 the second pre-training technique. For small datasets like UCF101 and HMDB51, using a self-supervised method is better than fully-supervised.

4.2. Comparison with other self-supervised methods

For comparing with other existing self-supervised methods, we fine-tune the whole network, not only the classification layer. The fine-tuning of our best videoSimCLR pre-trained model is done for 150 epochs, with different learning rate for the CNN part and the classifier part. Following the approach from DPC [17], the learning rate for the pre-trained network, CNN part, is $1/10$ of the learning rate for the classifier part. In Table 2, we split the results of the other self-supervised methods based on the criteria of using extra data for pre-training. The best Top 1 accuracy after fine-tuning the whole network are obtained using Kinetics as pre-training dataset, all of the methods from the extra data part falling into this category. When self-supervised pre-trained on Kinetics, previous methods achieve accuracies between 60% and 70% on UCF101.

Our method has promising results in the non extra data category, but with lower accuracy in comparison to the current state-of-the-art when pre-training on UCF101. Nonetheless, fine-tuning 3D Resnet18 from scratch starting with random weights is inferior to using our pre-trained weights. Note that DPC uses 40 frames from each video, compared to Random16 for our videoSimCLR. Moreover, TCE has a 2D Resnet architecture which generates embeddings for each frame, instead of making one feature vector for the whole videos as our videoSimCLR approach.

Method	Architecture	Extra data	UCF Acc	HMDB Acc
3D RotNet [24]	3D Resnet18	✓	62.9%	-
3D Cubic Puzzles [28]	3D Resnet18	✓	65.8%	33.7%
DPC [17]	3D Resnet18	✓	**68.2%**	**34.5%**
Shuffle & Learn [38]	CaffeNet	✗	50.2%	18.1%
OPN [34]	VGG M2048	✗	51.9%	23.8%
VideoGAN [56]	C3D	✗	52.1%	-
AoT [59]	AlexNet	✗	55.3%	-
videoSimCLR (ours)	*3D Resnet18*	✗	*56.7%*	*28.6%*
DPC [17]	3D Resnet18	✗	60.6%	-
TCE [30]	2D Resnet50	✗	**68.2%**	**31.7%**
Random init [26]	3D Resnet18	-	38.6%	16.1%
Kinetics [19]	3D Resnet18	-	84.4%	56.4%

Table 2: videoSimCLR fine-tuning results compared to other self-supervised techniques. All of the methods fine-tune the whole network. The best results are achieved when using extra data for pre-training, namely the Kinetics dataset. Our videoSimCLR method performs better compared to a couple of others self-supervised methods from the same no extra-data category, but it is still relatively behind to the current state-of-the-art. videoSimCLR gives significant improvement over fine-tuning by starting with a random initialization. The results of fine-tuning from Kinetics supervised pre-trained weights show that there is still a gap between unsupervised and supervised approaches.

4.3. Qualitative evaluation of videoSimCLR by video retrieval visualization

To further analyze the quality of the learned representation, we visualize the 3 nearest neighbors according to cosine similarity of three query clips from UCF101 testing set. The 256-embeddings are generated by feeding 16 frames to a pre-trained 3D Resnet18 network, either

unsupervised pre-trained using videoSimCLR on UCF101 training set or supervised pre-trained on Kinetics.

As can be seen in Figure 4, in most of the cases the semantic of the human action is well captured with the retrieved clips being semantically diverse (e.g camera viewpoint, different background) when compared to the query clip. When looking at the first row for each method, with the query clip of Baby Crawling, we observe videoSimCLR capturing better the semantic of the action by retrieving a completely different clip in terms of background and camera view-point, but with the same class. However, the other two rows show the superiority of Kinetics pre-trained. The mistakes at class level for videoSimCLR retrieved clips are showing promising semantic information such as up and down movement (videoSimCLR second row) and one to one combat (videoSimCLR third row). The action classes of videoSimCLR retrieved clips from a query clip tend to disagree more compared to Kinetics retrieved clips. Nevertheless, even for the videoSimCLR retrieved clips with a different class as the one of the query clip, the semantics of the retrieved clip and query are similar. The scene appearance impact is minimal, with few exception in videoSimCLR third row having the query clip from a boxing fight with the background floor of blue color: the second retrieved clip showing a person jumping into a pool is only similar in terms of the blue background and the third retrieved video has a news tickers which is confused with the score ticker from the boxing video.

5. Ablation study

To find the best configuration, we run an extensive ablation study presented from 5.1 to 5.5 using a batch size of 256 with smaller frame sizes of 56x56, pre-training all of the models for 100 epochs.

5.1. What is the effect of spatial transformation on learning action representations?

There are two possibilities when applying a random spatial transformation to a video: either keep the same random transformation for the whole video or apply a different transformation on each frame of the video. The sequence of spatial transformations contains a crop of random size $\in [0.08, 1]$ uniform in area, that applied different for each frame could make it impossible for the model to encode relevant information in regards to the action from the video.

To test this assumption, we run the first experiment on UCF101 using Random 8 and Random 16 as sampling techniques. From Table 3 we can observe significant accuracy improvements when using more frames per video, with Random 16 being the best choice for both supervised pre-training approaches, the one using UCF101 labels and the other taking the Kinetics pre-trained model. This accuracy improvement does not hold when using videoSimCLR with a different spatial transformation on each frame for pre-training, Random 8 and 16 performing almost the same. In all of our next experiments we are keeping the same spatial random transformation for each video, since we showed the clear superiority of this approach.

Sampling Method	Pretraining Method	UCF101 Acc@1 Frame	UCF101 Acc@1 Chunk
Random 8	Supervised UCF	28.8%	28.8%
	videoSimCLR	**22.6%**	**29.5%**
	Kinetics 700	29.4%	35.1%
Random 16	Supervised UCF	29.4%	29.4%
	videoSimCLR	**22.1%**	**32%**
	Kinetics 700	43.3%	43.3%

Table 3: Different pre-training methods using 8 or 16 frames from a random frame index with a size of 56x56. Top-1 accuracy is changing from Different column to Same column only for videoSimCLR, since there are two ways of applying a random spatial transformation on a video: Frame-way with a different transformation on each frame and Chunk-way with the same transformation for all of the frames inside a video. videoSimCLR same is the best method for both sampling techniques, showing significant improvement when using more frames per video.

5.2. Is videoSimCLR capable of learning visual features from a different smaller dataset?

In the previous experiment, we used videoSimCLR pre-training method with only random spatial transformations. Having promising results when pre-training on UCF101 keeping the same spatial transformation for all of the frames of each video, we further investigate the capabilities of the method when pre-training on a smaller dataset, HMDB51. In Figure 5, we can observe the same pattern for both datasets used for pre-training. The models are pre-trained on the training set of one of the two datasets and evaluated using the testing set of the dataset used for pre-training. videoSimCLR with only spatial random transformation is better than supervised pre-training on both datasets, but still is not able to achieve the same performance as the Kinetics pre-trained. We will stop comparing with Kinetics for the next experiments, since it will not provide a fair comparison given the massive size difference when compared to our training datasets, UCF101 and HMDB51. Moreover, in all of our next experiments will be be using Random 16 as sampling method, since compared to Random 8 it produced much better results.

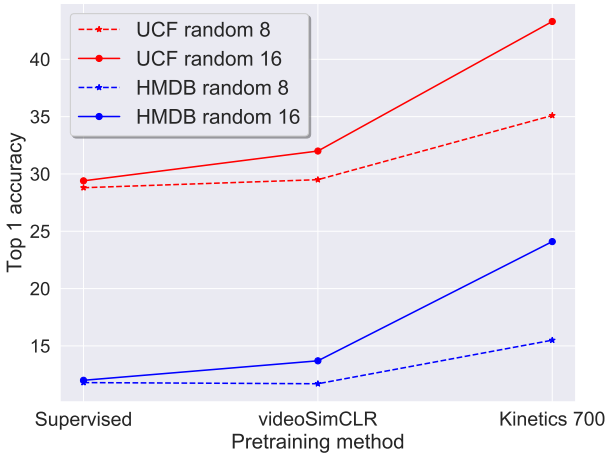

Figure 5: Comparison of 3 pre-training methods on 2 action recognition datasets, HMDB51 and UCF101. For each dataset, two sampling techniques are used with 8 and 16 frames per video. The performance is reported using the linear evaluation protocol, videoSimCLR surpassing the supervised pre-training on both datasets. Experiment using 56x56 as frame size.

5.3. Does adding random temporal shifting before the spatial transformation helps?

The previous experiments were only using random spatial transformations, without having any temporal transformation applied on the sampled clip. Applying the spatial transformations on different versions in the temporal axis of the sampled clip should force the model to better learn the temporal component of the action. To test this hypothesis, we start by adding temporal shifting as a way of increasing how much the models "sees" from the video without having the increase the number of frames sampled from each video. As seen in Table 4, we tried a number of shifting steps for the both datasets, with $s = 4$ being the best for HMDB and $s = 8$ for UCF. With the right shift step, adding random temporal shifting before the spatial transformation improves the performance for both datasets.

Temporal Shift Type	UCF101 Acc@1	HMDB51 Acc@1
None	32%	13.7%
Random [-4, 4]	31.6%	**14.7%**
Random [-8, 8]	**32.8%**	13.5%
Random [-16, 16]	31.3%	11.8%

Table 4: Analysis of the impact of adding random temporal shifting before the spatial transformation. The None-row is only with the random spatial transformation. Several shifting steps are tested, showing that the size of the shift needs to be correlated with the dataset. The Top-1 accuracy improves when using temporal shifting when the step size is set accordingly. Experiment using 56x56 as frame size.

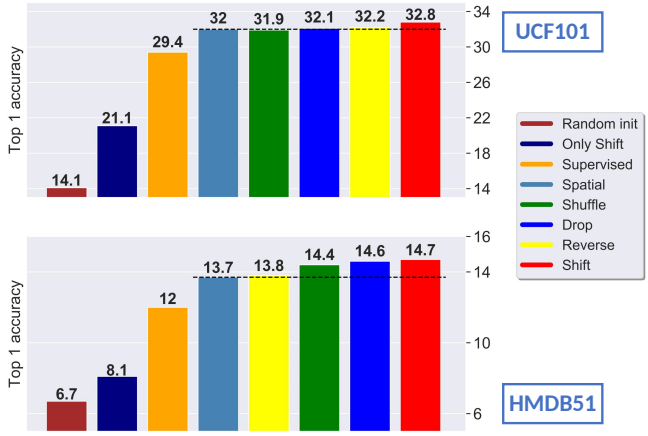

Figure 6: The impact overview of using different random temporal transformations before applying the spatial transformation. In most of the cases, adding temporal transformation improves the performance, with similar relative differences on both UCF101 and HMDB51, an exception being for the Temporal Shuffle. The best improvement is obtained with Temporal Shifting. Without applying the spatial transformation, even the best temporal transformation (Only Shift) performs poorly. When testing using the linear evaluation protocol, the pre-trained weights learned by videoSimCLR are much better than randomly initializing (Random Init) the network or using the weights from supervised pre-training. Experiment using 56x56 as frame size.

5.4. What is the effect of adding other temporal transformations?

With the improvements resulted from adding temporal shifting before the spatial transformations, the question of using other types of temporal transformation arises. We run experiments for both UCF101 and HMDB51 using other three temporal transformation types: Frame Drop, Reverse and Shuffle. Figure 6 indicates the effect of adding different types of temporal transformations before the random spatial transformation. To also validate that the improvements come from the combination of temporal and spatial transformation, we run another test with only the best temporal transformation. Not using the spatial transformation badly affects the performance, the combination of the two being crucial for achieving good results. With all of this results, we can conclude that the best setup is *Random Temporal Shifting of step 4 with Random Spatial Transformations* for HMDB51 and *Random Temporal Shifting of step 8 with Random Spatial Transformations* for UCF101.

5.5. How to modify MOCO dictionary size given the length of our training datasets?

We run a couple of experiments extending another self-supervised contrastive learning image based method, namely MOCO. Different queue sizes were tested dependant on the size of the pre-training dataset, with $K.\{512, 1024, 2048\}$ for HMDB51 and $K.\{1024, 4096, 8192\}$ for UCF101. Controlling the size of the queue is important for both allowing more negative samples to be stored and limiting the number of positive examples from the dictionary. In Table 5, the linear evaluation accuracy on UCF101 and HMDB51 is reported for all of the tested queue sizes. Choosing the right queue size influences videoMOCO performance, with the best results using $K = 1024$ for HMDB51 and $K = 8192$ for UCF101. The Top-1 accuracy is comparable to pre-training in a supervised manner for HMDB51 and slightly lower for UCF101. Hoewever, the overall performance of videoMOCO is lower when compared to even our initial videoSimCLR model that uses only random spatial transformation, without taking into consideration the temporal dimension.

Queue size K	HMDB51 Acc@1	UCF101 Acc@1
1024 / 512	- / 10.1%	24.9% / -
4096 / 1024	- / **12.1%**	25.4% / -
8192 / 2048	- / 10.9%	**26.1%** / -

Table 5: videoMOCO V1 results when pre-training with different queue sizes on HMDB51 (bottom half of each cell) and UCF101 (top half of each cell). The queue sizes K are chosen accroding to the size of the pre-training dataset, ∼3.5k for HMDB and ∼9.5k for UCF. The Top-1 accuracy is on the same level as supervised pre-training for HMDB (12%) and ∼3% lower for UCF (29.4%). For both datasets, videoMOCO V1 performs worst compared to videoSimCLR. Experiment using 56x56 as frame size.

6. Conclusion

In this paper, the challenging task of learning visual features from videos without the need of human annotated data was tackled. We started with extending a self-supervised contrastive based approach from images to videos. This step involved applying only random spatial transformations to the video frames of UCF101, having the question of how to apply the spatial transformation on videos. From the initial version of our method, videoSimCLR, we further explored the model capabilities to learn from another smaller dataset, HMDB51. After that, naturally we addressed the hypothesis of generating better visual features by using temporal information. A variety of random temporal transformation were applied, with the experiments leading to Temporal Shifting being the better choice, but only in combination with the previous spatial transformations. To further increase the importance of our work to the community, we proposed a second self-supervised approach, videoMOCO, in two versions with the latter adding videoSimCLR ideas into videoMOCO framework. The evaluation of the learned features was conducted both quantitatively by reporting Top-1 accuracy on both dataset using linear evaluation protocol and fine-tuning and qualitatively by visualizing a couple of video retrieval results.

Limitations. First, in terms of comparing with the state-of-the-art, videoSimCLR has lower results that can be attribute to the use of a smaller batch size given our GPU memory limitations. The size of the batch determines the number of negative pairs needed for the contrastive loss, therefore a batch of 128 clips is probably not enough to achieved state-of-the-art results. Moreover, the training of our videoSimCLR and videoMOCO models is not done on a very large scale action dataset like Kinetics, which may improve the performance drastically. Secondly, looking at the last row of videoSimCLR video clip retrieval results from Figure 4, the second and the third closest neighbor clips are showing that background information is influencing the learned features without adding additional value in regards to the action cues: the blue background of the pool is similar to the blue floor of the ring (second neighbor), the news ticker from the video of a man walking a dog is considered similar to the score ticker from the boxing video (third neighbor). Finally, videoMOCO comes with the dictionary size limitation adapted to the length of our training datasets. This limitation in relevant, since big dictionaries allow the storing of more negative samples needed for the contrastive loss.

Acknowledgment

This work was supported by Pattern Recognition & Bioinformatics research group from Delft University of Technology. Our code can be found at:
https://github.com/simionAndrei/MScThesis.

References

[1] Humam Alwassel, Dhruv Mahajan, Lorenzo Torresani, Bernard Ghanem, and Du Tran. Self-supervised learning by cross-modal audio-video clustering. *arXiv preprint arXiv:1911.12667*, 2019.

[2] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *Advances in Neural Information Processing Systems*, pages 15535–15545, 2019.

[3] Uta Buchler, Biagio Brattoli, and Bjorn Ommer. Improving spatiotemporal self-supervision by deep reinforcement learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 770–786, 2018.

[4] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[6] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[7] Xinlei Chen, Ross Girshick, Kaiming He, and Piotr Dollár. Tensormask: A foundation for dense object segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2061–2069, 2019.

[8] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015.

[9] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

[10] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1734–1747, 2015.

[11] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 6202–6211, 2019.

[12] Basura Fernando, Hakan Bilen, Efstratios Gavves, and Stephen Gould. Self-supervised video representation learning with odd-one-out networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3636–3645, 2017.

[13] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[15] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010.

[16] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[17] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[18] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Learning spatio-temporal features with 3d residual networks for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 3154–3160, 2017.

[19] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6546–6555, 2018.

[20] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[21] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[23] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2012.

[24] Longlong Jing and Yingli Tian. Self-supervised spatiotemporal feature learning by video geometric transformations. *arXiv preprint arXiv:1811.11387*, 2(7):8, 2018.

[25] Mahmut Kaya and Hasan Şakir Bilge. Deep metric learning: A survey. *Symmetry*, 11(9):1066, 2019.

[26] Osman Semih Kayhan and Jan C van Gemert. On translation invariance in cnns: Convolutional layers can exploit absolute spatial location. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14274–14285, 2020.

[27] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2020.

[28] Dahun Kim, Donghyeon Cho, and In So Kweon. Self-supervised video representation learning with space-time cubic puzzles. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8545–8552, 2019.

[29] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019.

[30] Joshua Knights, Anthony Vanderkop, Daniel Ward, Olivia Mackenzie-Ross, and Peyman Moghadam. Temporally coherent embeddings for self-supervised video representation learning. *arXiv preprint arXiv:2004.02753*, 2020.

[31] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1920–1929, 2019.

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[33] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *2011 International Conference on Computer Vision*, pages 2556–2563. IEEE, 2011.

[34] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 667–676, 2017.

[35] Jungkyu Lee, Taeryun Won, and Kiho Hong. Compounding the performance improvements of assembled techniques in a convolutional neural network. *arXiv preprint arXiv:2001.06268*, 2020.

[36] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[37] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[38] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, pages 527–544. Springer, 2016.

[39] Pedro Morgado, Nuno Vasconcelos, and Ishan Misra. Audio-visual instance discrimination with cross-modal agreement. *arXiv preprint arXiv:2004.12943*, 2020.

[40] Jonathan Munro and Dima Damen. Multi-modal domain adaptation for fine-grained action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 122–132, 2020.

[41] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[42] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016.

[43] Mehdi Noroozi, Hamed Pirsiavash, and Paolo Favaro. Representation learning by learning to count. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5898–5906, 2017.

[44] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[45] Andrew Owens and Alexei A Efros. Audio-visual scene analysis with self-supervised multisensory features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 631–648, 2018.

[46] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[47] Mandela Patrick, Yuki M Asano, Ruth Fong, João F Henriques, Geoffrey Zweig, and Andrea Vedaldi. Multi-modal self-supervision from generalized data transformations. *arXiv preprint arXiv:2003.04298*, 2020.

[48] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[49] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[50] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in neural information processing systems*, pages 1857–1865, 2016.

[51] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[52] Alexandros Stergiou and Ronald Poppe. Learn to cycle: Time-consistent feature discovery for action recognition. *arXiv preprint arXiv:2006.08247*, 2020.

[53] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. Fixing the train-test resolution discrepancy: Fixefficientnet. *arXiv preprint arXiv:2003.08237*, 2020.

[54] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[55] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[56] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Advances in neural information processing systems*, pages 613–621, 2016.

[57] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. In *Proceedings of the European conference on computer vision (ECCV)*, pages 391–408, 2018.

[58] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2015.

[59] Donglai Wei, Joseph J Lim, Andrew Zisserman, and William T Freeman. Learning and using the arrow of time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8052–8060, 2018.

[60] Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pages 1473–1480, 2006.

[61] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2018.

[62] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698, 2020.

[63] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.

[64] Oxford Google Deep Mind Zisser A. Self-supervised learning, 2018.

2

The Basics of Deep Learning

Deep learning is an area of machine learning that uses Artificial Neural Networks (NN) [62] for representational learning. Compared to the classic machine learning approach of designing feature extraction methods, deep learning focuses on making a system to automatically extract features from the raw input data. The learning process can be done either supervised or unsupervised. The most common method is to learn from a labeled training set, which is the supervised case.

2.1. Feed-forward Networks

Feed-forward Networks are a type of NN in which the information flows from the input data x through the intermediate layers to reach the output y, without any feedback connections [17]. Feed-forward Networks are design to approximate some function f^* to map the input to the output: $y = f^*(x)$. The connections from the input to the output have learnable weights to encapsulate the information from the training data. In this way, a mapping $y = f(x, \theta)$ is defined with θ being the learnable parameters.

The input data is fed through the network in the forward pass. A loss function is used to measure the difference between the mapping f and f^*. The parameters θ are updated during the backward pass based on an optimization criteria, given the loss function values on the training set.

2.1.1. Multi-layered perceptron

The most simple way of connecting the inputs to the outputs is directly by a series of weights, this type of Feed-forward Network being called single-layered perceptron. If a number of hidden layers are placed between the input and the output layer, a multi-layered perceptron (MLP) is obtained. The computations can be described by a Directed Acyclic Graph (DAG) [18] as a chain of function compositions. The depth of a MLP is given by the number of layers, with the example from Figure 2.1 having a depth of 3: input, hidden and output layer. The network is drawn in both styles, the explicit expanded form and the compact notation by using vector representations. Each layer consists of units called neurons, which are connected to all the units from the previous layer. A MLP is also called a fully-connected (FC) network.

2.1.2. Activation function

Let's take a look at the output of the previous MLP with one hidden layer, $y = f(x) = \omega^T g(x, W^T) + b$. If the function g that computes the output of hidden layer h is a linear function, $f(x) = \omega^T W^T x$ will be a linear function as well. With this rigid limitation, f cannot approximate a non-linear function like the binary XOR. To solve this problem, non-linear activation functions [7, 27, 47, 51] are used to compute the hidden layer values.

The most trivial activation function is a binary step function based on a threshold value, but this limits the output set to only two values. Sigmoid and Hyperbolic Tangent (Tanh) functions limit the output to an interval: $Sigmoid \in [0, 1]$ and $Tanh \in [-1, 1]$. Sigmoid activation function is generally used for networks to output a probability, on which a threshold can be applied for binary classification purposes. For a multinomial classification problem, a generalized version of Sigmoid exists, the Softmax activation function. The most used function is the Rectified Linear Unit (ReLU) [51], which computes $max(x, 0)$.

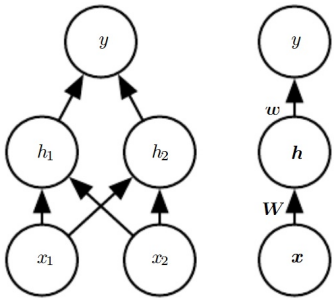

Figure 2.1: Multi-layered perceptron example with a depth of 3 represented in two different styles. The learnable weights are the matrix W describing the mapping from x to h and the vector ω for the mapping from h to y. The network can be written as $y = \omega^T h(x) + b = \omega^T g(x, W^T) + b$, with b being the bias term. Figure courtesy of [17].

The problem with ReLU activation function comes from negative inputs when the neuron turns into zero, affecting the optimization from the backward pass. Leaky ReLU [47] addresses this issue by replacing the constant zero with a very small coefficient for the negative inputs.

The mathematical formulas for Sigmoid, Tanh, ReLU and Leaky ReLU as displayed in Equation 2.1. The plots for all of this activation functions with an input $x \in [-10, 10]$ can be seen in Figure 2.2.

$$\text{Sigmoid:} \quad g(x) = \frac{1}{1+e^{-x}} \qquad \text{Tanh:} \quad g(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$
$$\text{ReLU:} \quad g(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases} \qquad \text{Leaky ReLU:} \quad g(x) = \begin{cases} 0.01x & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases} \tag{2.1}$$

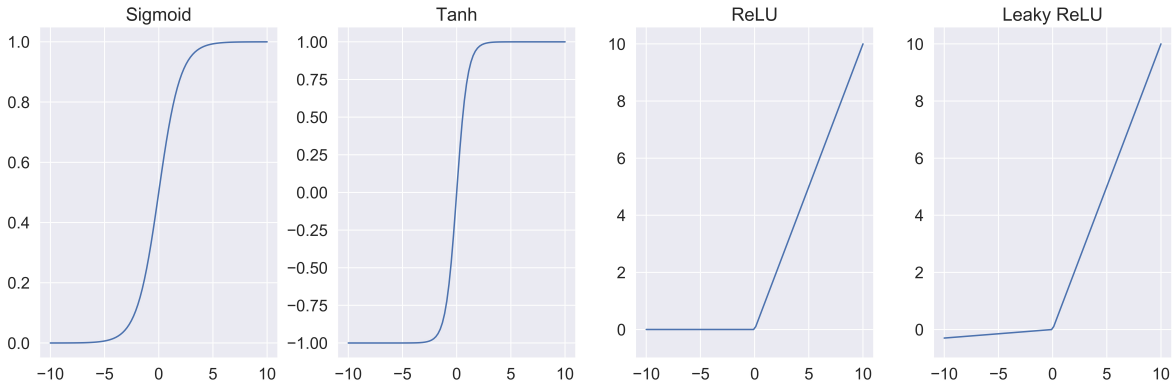

Figure 2.2: The plots of four common used activation functions: Sigmoid, Hyperbolic Tangent (Tanh), Rectified Linear Unit (ReLU) and Leaky ReLU. The graphs are made for an input $x \in [-10, 10]$. Tanh is similar to Sigmoid, but zero-centered. Leaky ReLU is a modified version of ReLU resolving the problem of neurons being deactivated for negative values.

2.2. Optimization

The optimization during the backward pass involves minimizing or maximizing an objective/loss function. Usually, for deep learning networks, the cross-entropy between the data distribution and the model distribution is what needs to be optimized.

When dealing with multinomial classification, for the target $t \in \{1, \ldots, K\}$ the "one-hot" encoding is employed $t = (0, \ldots, 0, 1, 0, \ldots, 0)$. The network output will be a vector of probabilities obtained by applying Softmax and the cross-entropy will be the loss function to be optimized, $\mathcal{L}_{CE}(y, t) = -t^T(\log y)$.

2.2.1. Stochastic gradient descent

Let's assume we have a function $y = f(x)$, with $x, y \in \mathbb{R}$ and its derivative $f'(x)$. The derivative of a function gives its slope at a certain point x. This indicates how a small change ε in the input will affect the output: $f(x + \varepsilon) \approx f(x) + \varepsilon f'(x)$. Gradient descent [45] is the technique of minimizing a function

$f(x)$ by moving x in the opposite sign of its derivative, $f'(x)$, with small steps. Looking at Figure 2.3, having a function $f(x) = \frac{1}{2}x^2$ and its derivative $f'(x) = x$, f has a global minimum at $x = 0$ when $f'(x) = 0$. In this case, the derivative has the same sign as the input x, thus moving rightward for negative values and leftward for positives ones will result in reaching the minimum.

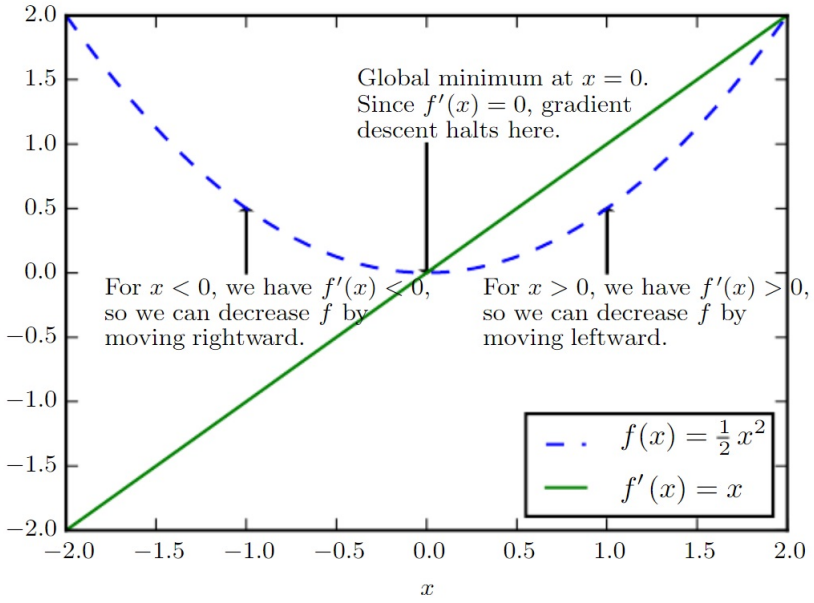

Figure 2.3: Gradient descent algorithm illustration for finding the minimum of a function $f(x) = \frac{1}{2}x^2$. Starting from either positives or negative values, gradient descent optimization will find the solution in $x = 0$ by moving in the opposite sign of the gradient, $f'(x) = x$. Figure courtesy of [17].

For neural networks, the function f is dependant of both the input x and the weights θ: $y = f(x, \theta)$. The gradient $\nabla_\theta f(x, \theta)$ is a vector of all partial derivatives with respect to the weights $\frac{\partial}{\partial \theta_i} f(x, \theta)$. Updating the parameters using gradient descent with step size ε can be written as $\theta' = \theta - \varepsilon \nabla_\theta f(x, \theta)$. In deep learning terminology, the step size is called learning rate. The stochastic approximation [60] idea is behind Stochastic gradient descent (SGD), with SGD being an approximation of the gradient from a small number of samples. The SGD update formula based on a batch of m' samples can be seen in equation 2.2.

$$\theta' = \theta - \varepsilon \frac{1}{m'} \sum_{i=1}^{m'} \nabla_\theta L(x^{(i)}, y^{(i)}, \theta) \quad \text{where } L \text{ is the loss function} \tag{2.2}$$

2.2.2. Cosine Annealing

Cosine Annealing learning rate schedule was first introduced for snapshot ensembles [46] which require an aggressive learning rate changing policy. The learning rate starts at a high value, but it is quickly decreased to a minimum value of almost zero, before being increased again to the previous maximum value. This learning rate schedule acts as a restart of the learning process and on each cycle it continues from the current network learned weights. This is a warm restart, as opposed to a cold restart which resets the network weights to random values at each cycle. The learning rate evolution over 100 epochs, with cycles of 20 epochs and starting with a learning rate of 0.01 can be analyzed in Figure 2.4.

The formula for the learning rate at each step t following the Cosine Annealing schedule can be found in equation 2.3, where $[n_{min}, n_{max}]$ is the range for the learning rate, T_{cur} shows how many epochs have passed since the last restart and T_{max} is the maximum number of iterations.

$$\eta_t = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})\left(1 + \cos\left(\frac{T_{cur}}{T_{max}}\pi\right)\right) \tag{2.3}$$

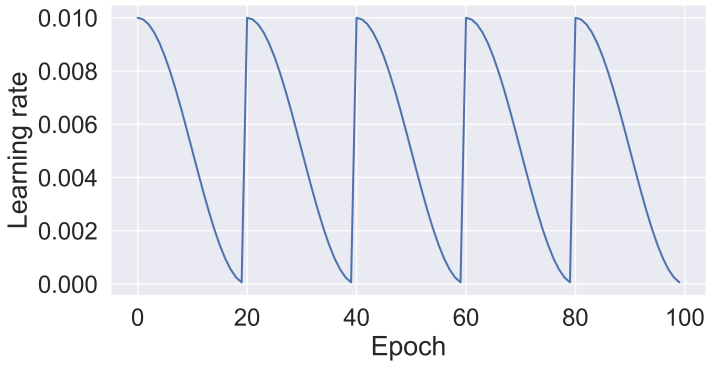

Figure 2.4: Cosine Annealing learning rate schedule example over 100 epoch, with restarts at each 20 epochs.

2.3. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) [43] are the *de facto* standard when dealing with computer vision tasks such as image classification [24, 30, 64], object detection and localization [58, 59], instance segmentation [25, 61], video action classification [3, 23, 35], generative models for images and videos [6, 16], etc.

The paper in which AlexNet [41] architecture was introduced achieved a more than 10% lower error rate on ImageNet [9], compared to the previous state-of-the-art. Surpassing the existing methods by such a wide margin, AlexNet is responsible for the beginning of CNN dominance in Computer Vision tasks.

Typically, a CNN is made of multiple convolutional layers with non-linear activation functions in between and pooling layers. The CNN pipeline from Figure 2.5 shows that starting from the input image, features are learned by the CNN block and the classification is made using fully-connected layers on the flattened learned representations. The end of the pipeline contains a Softmax activation function to generate a vector of probabilities, with the size equal to the number of classes.

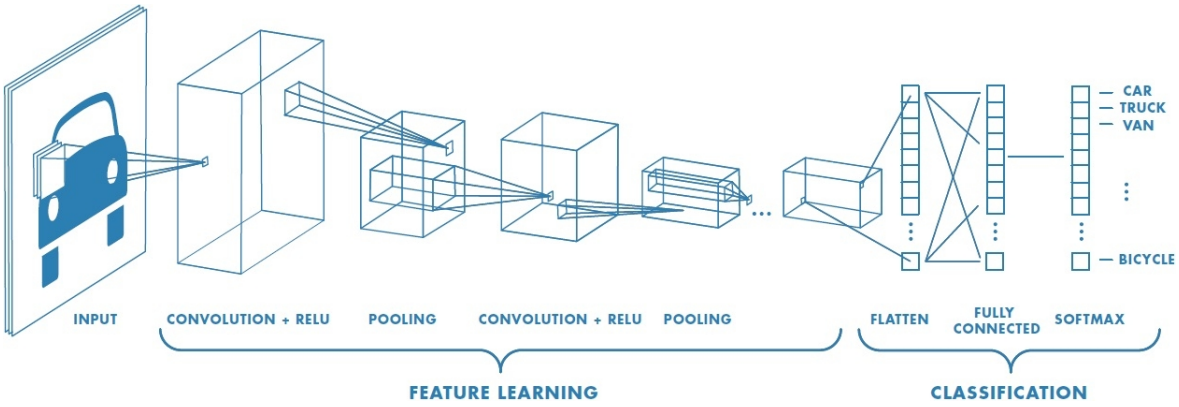

Figure 2.5: CNN overview for a classification problem. From the input image, the feature learning is done by a series of convolutional layers, ReLU and pooling. The classification part uses fully-connected layers with a Softmax activation at the end.[1]

2.3.1. Convolution operator

The convolution is the core operation behind CNNs. In NN terminology, for convolution the most important terms are the input I, the kernel K and the feature map S. The convolution operator is denoted by $*$, having the following formula:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n) \tag{2.4}$$

[1] © https://ch.mathworks.com/discovery/convolutional-neural-network.html

Note that the actual operation described by equation 2.4 is called cross-correlation, but since machine learning terminology refers it as convolution, we will also follow this convention. A matrix K, with the size smaller compared to I, is sliding over the two-dimensional input. We define the stride as the number of pixels we move the kernel when sliding over the input image. Looking at Figure 2.6, each time we slide the 2x2 kernel over the 4x3 image, we calculate the weighted sum between the kernel weights and the pixels from the selected area of the image. At the end, with a stride of 1 we obtain a feature map of size 3x2. To generalize, starting from an input of size $[h, w]$, where h is the height and w the width, moving a kernel of size $[F_h, F_w]$ with a stride of p results in a feature map of size $[\frac{h-F_h}{p} + 1, \frac{w-F_w}{p} + 1]$.

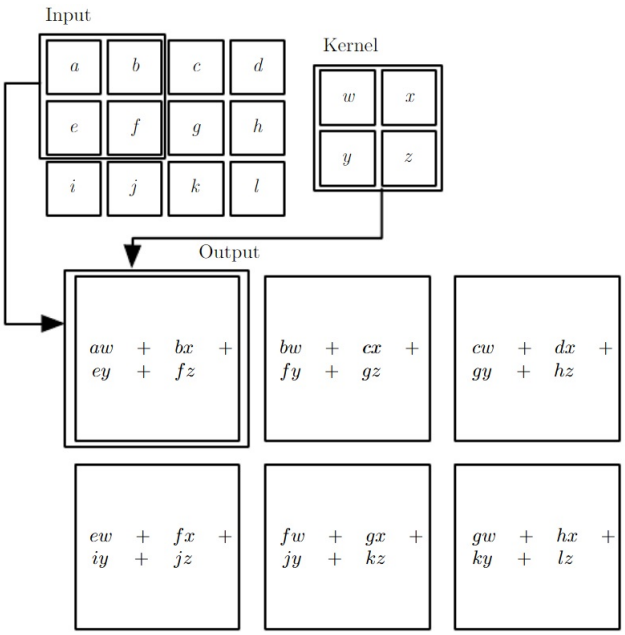

Figure 2.6: 2D Convolution example on an 4x3 input with a squared kernel of 2x2. The output, called the feature map, is the result of moving the kernel with a stride of 1 and computing each time the element-wise matrix multiplication between the kernel and the covered input area. Figure courtesy of [17].

2.3.2. Kernels as feature detectors

In the context of CNNs, kernels are called filters. Each convolutional layer is characterized by the size of the kernels, the number of filters and the stride. Each filter weights act as a particular feature detector, thus the learning of kernel weights translates into learning the features. In Figure 2.7, we can see two feature maps resulted after applying 2 different kernels on two images. For the first image, the filter detects the round shape of the sunflower with that particular feature being highlighted in the activation map. For the second one, we want to detect the position of a character named Waldo, so we use a filter similar to his shirt. From the appropriate feature map, we clearly observe that the most activated part corresponds to Waldo's location.

2.3.3. Spatial Pooling

The last important block of the feature learning layers is the spatial pooling. The pooling layers are used to summarize the outcome over a region by reducing the size of the feature maps. The summary can be achieved by applying different functions such as maximum or average in a similar manner to how kernels are applied on an image. The spatial pooling kernel computes the result of the summarizing function, maximum or average, based on the values from the feature map area covered by the kernel. The stride will affect how much the feature maps will be down-sampled with either maximum pooling or average pooling.

One of the advantages of pooling comes from reducing the memory due to smaller activation maps. Another important aspect in regards to image domain is that it makes the network invariant to local translations. This determines the feature presence to be more important than its location.

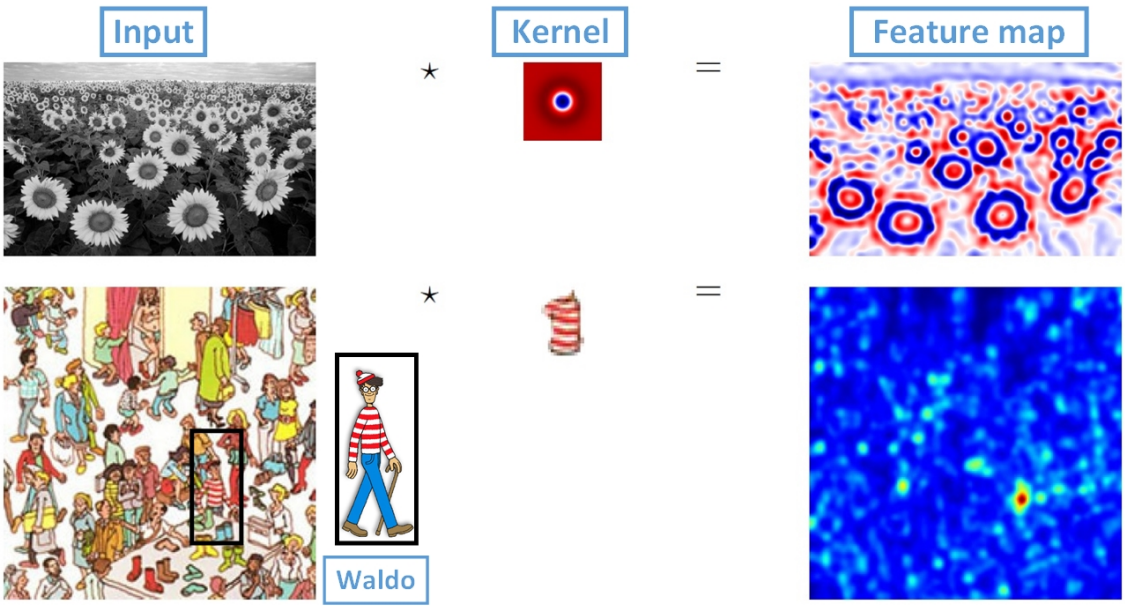

Figure 2.7: Kernels as feature detectors, with two activation maps showing the regions of interest highlighted. The first kernel activates for the round shape of the sunflower head and the second based on the pattern of Waldo's shirt. The second kernel facilitates the localization of Waldo in the scene by identifying the highest activated area from the feature map.[2]

An overview of one complete layer from the feature learning part of a CNN is presented in Figure 2.8. Starting from the input image, using 3 filters F_1, F_2, F_3, we have the Convolution block, followed by the Non-Linearity and finishing with the Spatial Pooling block that produces the output for the next layer. The outputs for each filter resulted after applying the spatial pooling will be concatenated and served as the new image channels to the next layer.

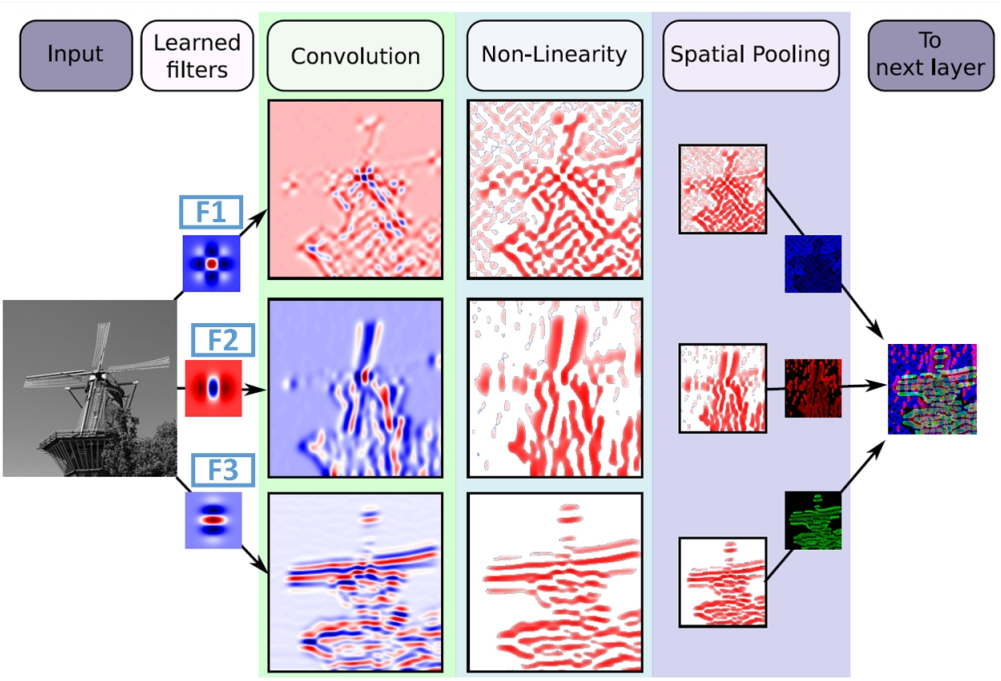

Figure 2.8: The summary of one layer of the Convolutional Network feature learning block, composed of a Convolution operation, ReLU Non-Linearity and Spatial Pooling. Each of the three filters will detect different features, with F1 for the windmill blade, F2 for vertical lines and F3 for horizontal ones.[2]

[2]© Figure adapted from TU Delft CS4180 Deep Learning course slides

3

Action recognition

Action recognition involves assigning an action label to a video clip composed of a sequence of 2D frames, with the action itself not necessarily being capture in all of the frames throughout the video. This task comes with multiple challenges, such as a significant computational cost, the need to capture the spatio-temporal context and the usage of different input flows such as RGB flow or optical flow [29]. The training computation cost comes from the need to use multiple frames for each video or the architecture increased number of parameters from adding 3D kernels. Motion information needs to be capture to obtain relevant features that are robust to changes in camera position or background and encode the meaning of the learned actions.

3.1. Common Deep Neural Network architectures for videos

Video feature learning architectures are design to extract both temporal and spatial information by using either only RGB flow [12, 22, 67] or a combination of both RGB and optical flow [3, 14, 63]. Figure 3.1 presents an schematic overview of the architectures, with the models being recurrent networks based [12], 2D cNNs based [14, 63] and 3D cNNs based [3, 14, 67].

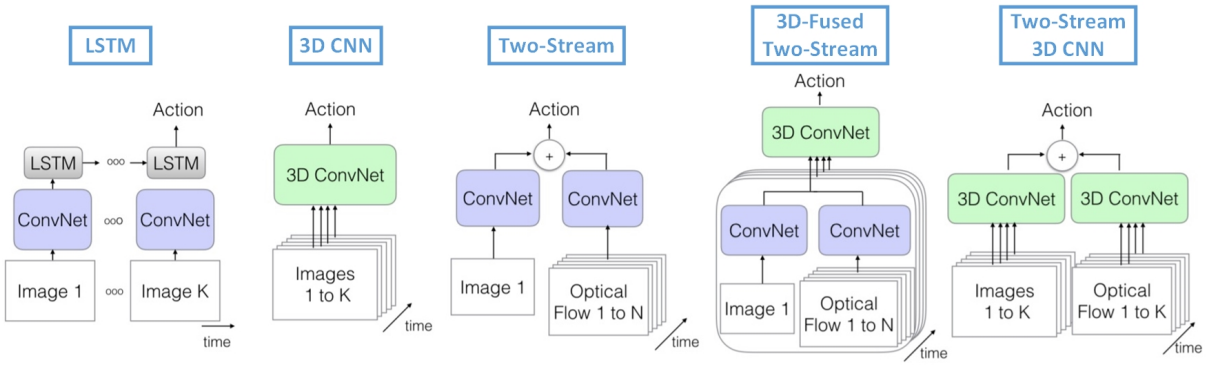

Figure 3.1: Schematic overview of the most common neural networks architectures ideas for video action recognition. All of the methods are using either 2D or 3D CNNs, with the LSTM based one having also recurrent networks incorporated. Figure adapted from [3].

3.1.1. Two Stream Networks

Two Stream Networks [63] are the successor of the failed experiment of Single Stream Networks [34] which used 2D Convolutions with different ways to fuse the temporal information. While Single Stream Networks performed worst compared to the state-of-the-art hand crafted features, Two Stream Networks are able to capture the temporal dimension by using optical flow vectors. As displayed in Figure 3.2, there are two parallel networks, both 2D CNN based, which receive different input in the form of the RBG stream for the Spatial stream ConvNet and optical flow stream for the Temporal stream

ConvNet. The two networks are trained separately and combined at the end by the means of an SVM [8] classifier.

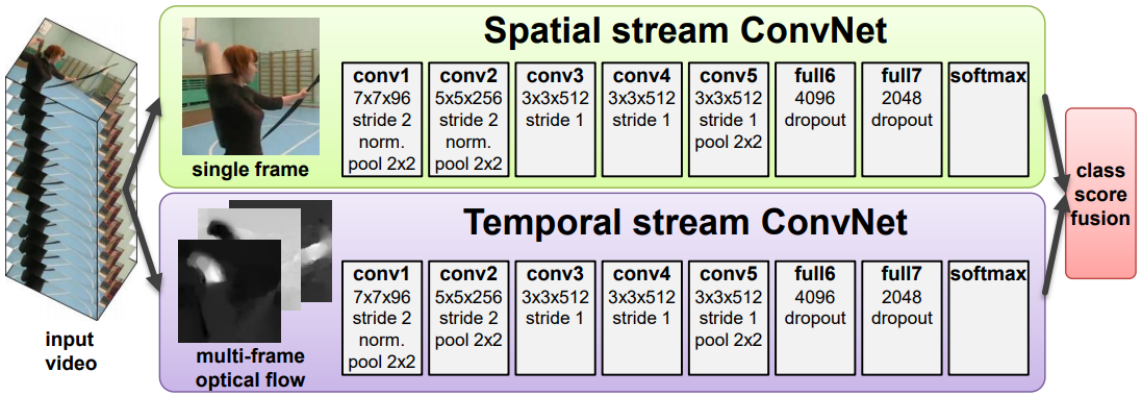

Figure 3.2: The overview illustration of Two stream architecture for video classification. The incorporation of optical flow allows the capturing of motion information and influence the class score in combination with the extracted spatial information from the RBG input. Figure courtesy of [63].

3.1.2. Long-Term Recurrent Convolutional Networks

Long Short-Term Memory (LSTM) [28] are a type of recurrent neural networks (RNN) capable of learning long-time dependencies. LTMs were designed to tackle the vanishing gradient problem encountered when training a classic RNN over a long sequence. Long-Term Recurrent Convolutional Networks (LRCNs) [12] are based on LSTM ability to deal with sequences, being an encoder-decoder architecture for video representations. In Figure 3.3, the LRCN arhitecture is shown with the input being encoded by CNNs to generate visual features which are then decoded by LSTMs. The architecture is trained end-to-end, with the output being averaged when used for action recognition tasks.

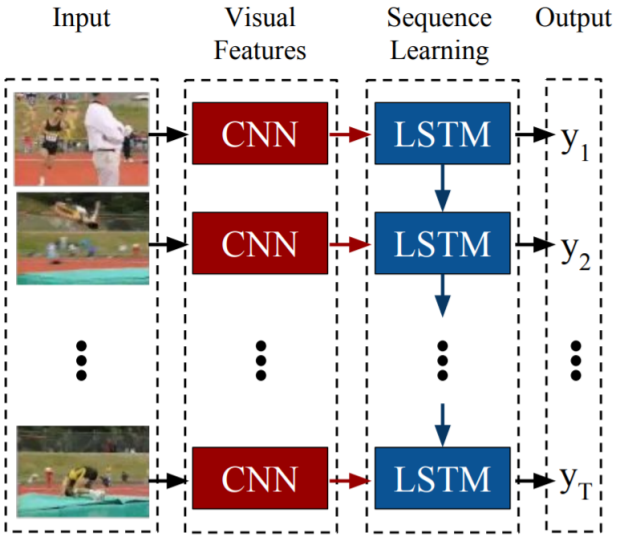

Figure 3.3: Long-Term Recurrent Convolutional Network architecture. The variable length input is processed by 2D CNNs followed by LSTM cells to generate a variable size output. The CNN block is used for visual features extraction and the LSTM block for learning the sequence. Figure adapted from [12].

3.1.3. 3D Convolutional Networks

The novel 3D convolutions [32] are able to directly capture the motion information from multiple adjacent frames. As seen in Figure 3.4, compared to 2D CNNs, 3D CNNs are capable to perform the convolution operation on the video volume. Taking advantage of the 3D kernels, an 11 layers deep 3D CNN architecture [67] with 3x3x3 filters was proposed. The model was trained on a large scale

supervised video dataset [34] and their learned spatio-temporal features, C3D, was able to outperform the state-of-the-art by simply using a linear classifier SVM on the top of the pre-trained C3D architecture used as a feature extractor.

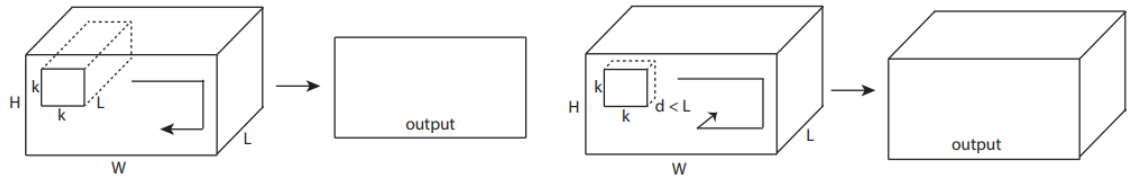

Figure 3.4: From left to right, applying a 2D convolution on each frame of the video volume results in another image compared to applying a 3D convolution on the video volume which results in another volume. Figure courtesy of [67].

3.1.4. 3D Residual Networks

With the grow in the network depth, the problem of increasing training complexity arises. This issue was addressed for image feature learning by presenting a residual learning framework [24] meant to ease the training process of deeper networks. While using deeper model, ResNets made use of skip connections to build residual blocks which are easier to optimize and promote the gradient propagation. Residual nets are capable of taking advantage of considerably increased depth without accuracy saturation. In Figure 3.5, a schematic view of ResNet50 architecture can be visualized.

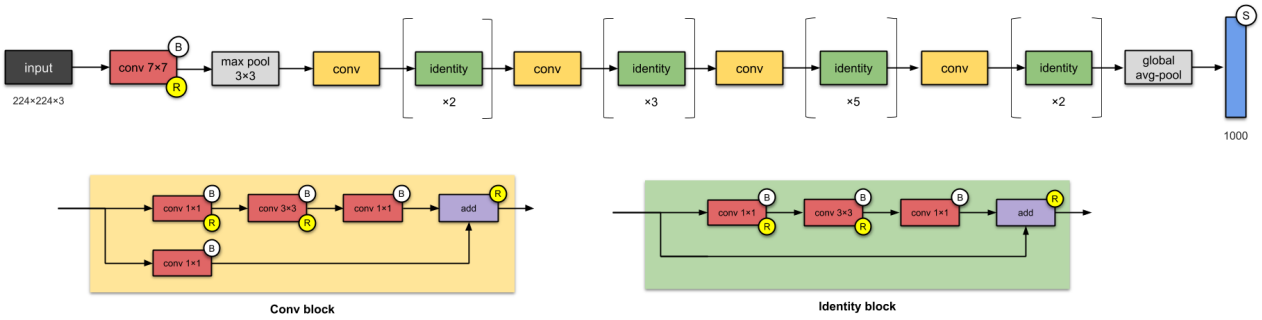

Figure 3.5: Resnet50 arhitecture, B denoting batch normalization [31], R the ReLU [51] activation function and S Softmax. The difference between Conv Block and Identity block is in the addition operation on the skip connection.[3]

Motivated by the success of the Convolutional Neural Networks with 3D kernels to directly extract spatio-temporal features, a version of ResNets with 3D kernels was proposed [22]. 3D ResNets with 18 and 34 layers were implemented, achieving better performance compared to C3D. Later on, deeper model like 3D ResNet50, 101, 152 and 200 were experimented.

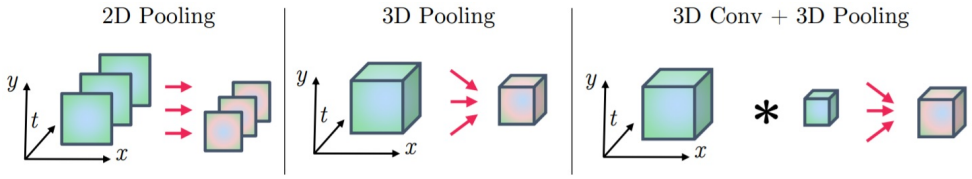

Figure 3.6: Temporal information fusing methods, with 2D pooling, 3D pooling and a combination of a 3D Conv with a 3D pooling by using a fusion kernel. Figure courtesy of [14].

3.1.5. 3D-Fused Two-Stream Networks

Two-Stream architecture was improved by modifying the way spatial and temporal streams are fused. 3D-Fused Two-Stream Networks [14] add two elements of novelty on top of the Two-stream

[3]© https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d

Network. The first one is early fusion at a convolutional layer, rather than at the softmax layer from the end. The second one is from how they performed the fusing by 3D CNNs and 3D pooling, with different ways of fusing temporal transformation illustrated in Figure 3.6.

3.1.6. Two Stream I3D

The 3D Convolutional Networks were also integrated into the Two Stream architecture, replacing both the 2D Spatial Stream ConvNet and the 2D Temporal Stream ConvNet. In the new model, Two-Stream Inflated 3D ConvNet (I3D) [3], the spatial stream input consist of frames stacked in time. The two networks are trained independently based on the RGB input and optical flow input, with the two outputs being average at inference time. Using pre-training on a new proposed large scale video dataset, they improved the state-of-the-art in action classification on several benchmarks.

3.2. Action recognition datasets

In this part we will present the most common video datasets used for human action recognition. The number of action classes varies from 51 all the way to 700.

3.2.1. HMDB51

HMDB51 [42] is an action recognition dataset with 6849 clips divided into 51 action categories with a minimum of 100 clips per category. Clips have a frame rate of 30FPS (frames per seconds) and frames of size 320x240. A class balanced version of the dataset is sampled with 3570 training clips and 1530 testing clips.

3.2.2. UCF101

UCF101 [66] is a larger action recognition dataset composed of 101 actions divided into 5 types: human-object interaction, body-motion only, human-human interaction, playing musical instruments and sports. It has 13320 videos split in 9537 training and 3783 testing clips. The dataset contains 27 hours of videos sampled at 25 FPS frame rate with 320x240 frame size. In Figure 3.7, 12 human action classes examples from both HMDB51 and UCF101 are displayed.

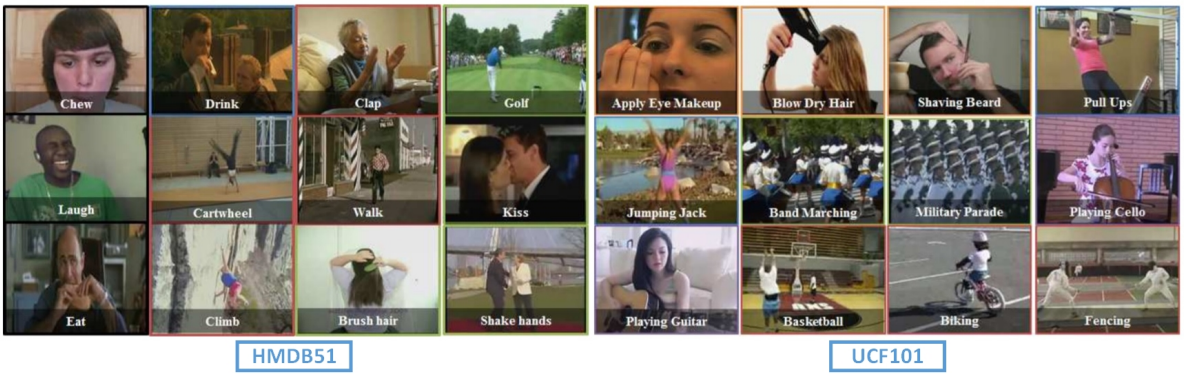

Figure 3.7: A couple of frame examples from HMDB51 and UCF101. For each dataset, 12 action classes are represented by 1 significant frame sampled from the video. Figure adapted from [2].

3.2.3. ActivityNet

ActivityNet [1] is a much bigger dataset having 203 activity categories with an average of 137 untrimmed videos per class and 1.41 activity instances per video. The total video hours of this database is 849. Around 50% of the videos have frames of size 1280x720, with the majority using a frame rate of 30 PFS.

3.2.4. Kinetics

Kinetics datasets aims to be for video action recognition what ImageNet[9] is for image classification. Kinetics 400 [3] is the first version of this dataset, with 400 human action classes and more than 400

clips per class, all collected from YouTube videos. The classes cover three types of human actions: person actions (singular), person-person actions and person-object actions. Each clip lasts around 10s. There are several large versions of this dataset, like Kinetics 600 and 700. Several examples from Kinetics 400 can be visualized in Figure 3.8.

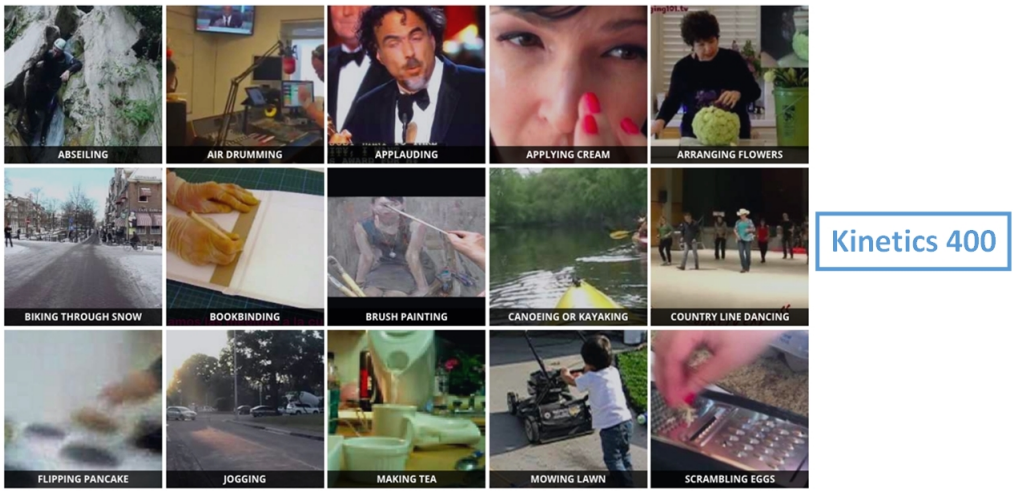

Figure 3.8: Kinetics examples from 15 human action classes, with 1 frame per video being displayed.[4]

3.2.5. Moments in Time

Moments in Time [50] is the latest large-scale human-annotated video dataset, with one million short clips of 3s showing dynamic events. Each video is annotated with one action class among the 339 different possible ones. The visual and auditory events involve different agents: human, animals and nature.

[4]© https://medium.com/datadriveninvestor/a-guide-to-human-activity-recognition-f11e4637dc4e

4

Self-supervised learning

Transfer learning [56] refers to the use of the NN weights learned while solving a problem for a different related task. This is done by pre-training on a available large dataset to encapsulate the knowledge derived from the data and saving the weights to be used on other possible related problem. Pre-training can be done both supervised or unsupervised, depending on the dataset being annotated or not.

Self-supervised learning [74] is a category of unsupervised learning methods in which the labels are generated automatically without the need of a human annotator. The generated labels are called pseudo-labels. To generate labels from unlabeled data a pretext task, also called self-supervised task, is created which takes advantage of the intrinsic relationship existent in the data. By optimizing to solve a pretext task which exploits different cues from the data, visual representation can be learned. During the self-supervised training, we are not necessarily interested in the performance of the pretext task, but rather in the quality of the learned features.

Based on the type of the pretext task, the self-supervised methods can be classified in a number of different categories. We will present a series of self-supervised technique for both image and videos feature generation by covering multiple pretext tasks.

4.1. Learning visual representations from images

4.1.1. Generative based methods

The generative based methods main goal is to create realistic and diverse images, but in this process useful visual representations can also be learned.

Generative Adversarial Networks. Generative Adversarial Networks (GANs) [16] are composed of two networks, the generator (G) and the discriminator (D). The generator G captures the data distribution of the real images in order to generate realistic images, while the discriminator D estimates the probability of an image to be part of the real data, rather than being generated by G. The training is design as a competition between G and D, like a min-max two players game in which both of the networks are helping in improving each other. After the adversarial training, G can be used to create additional realistic data. To accomplish the task of distinguishing between real and fake images, D needs to capture semantic features which could serve as a pre-trained model for fine-tuning to other vision tasks.

Bidirectional Generative Adversarial Networks. Classic GANs does not have the ability to learn the inverse mapping from the data space to the latent space. To tackle this issue, Bidirectional Generative Adversarial Networks (BiGANs) [11] add an encoder E capable of learning the projection from the input space to the latent one. A schematic view of BiGAN architecture can be seen in Figure 4.1.

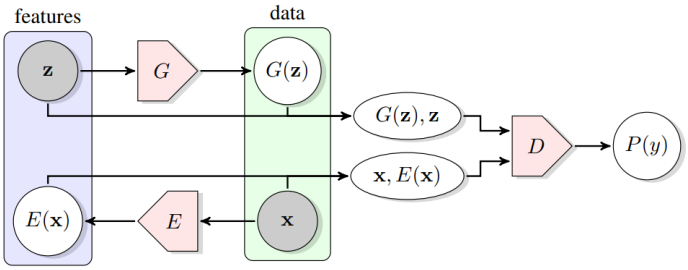

Figure 4.1: Bidirectional Generative Adversarial Networks structure. Encoder E maps data x to latent representation z, which makes D to discriminate in both latent and data spaces: G(z), z vs x, E(x). Figure courtesy of [11].

.

Context Encoders. Context based pixel prediction in another pretext task, with Context Encoders [57] trained to fill a gap from an image. The model training involves both a reconstruction loss and an adversarial one. The context encoder removes the information from all of the color channels of the dropped pixels to form the missing piece. Another method proposes the hiding of only a subset of channels [73], rather than all three RGB channels. Context Encoder architecture can be analyzed in Figure 4.2.

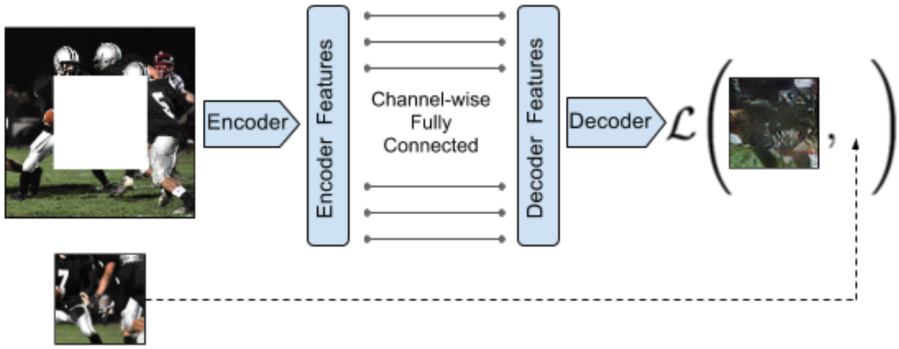

Figure 4.2: Overview of the Context Encoder Architecture. The context image with the missing piece is served as input to the encoder to generate features which are meant to be decoded by the decoder to obtain the absent pixels of the image. The loss function is computed between the decoded missing patch and the real missing piece. Figure courtesy of [57].

Image colorization. Another example of a self-supervised task is training a model to color a grayscale input. Image colorization [72] takes an input lightness channel L for a grayscale image and learns to find a mapping to the two associated color channels ab in CIE L*a*b* color space [54], where L* is lightness from black (0) to white(100), a* represent green(-) to red(+) component and b* is the blue(-) to yellow(+) component. Their CNN based architecture is presented in Figure 4.3. Colorization demonstrates to be a powerful pretext task acting as a cross-channel encoder and serving as a self-supervised pre-training method for object classification, detection and segmentation.

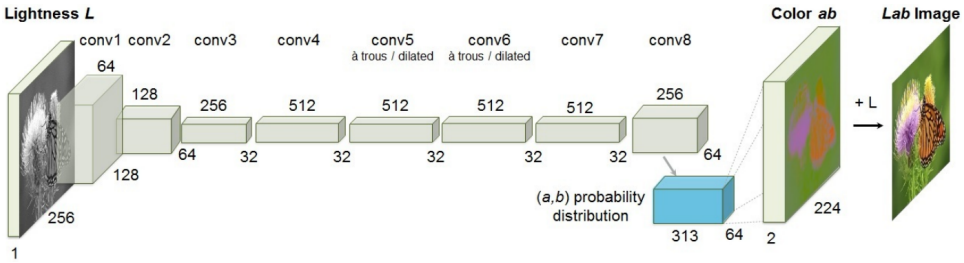

Figure 4.3: Colorful Image Colorization architecture with each conv block representing a series of 2-3 repeated convolutions with ReLU activation and batch normalization. The mapping learned is from the lightness L of a grayscale image to the color ab to form an image in Lab color space. Figure courtesy of [72].

4.1.2. Relationships between patches

This category of self-supervised methods includes pretext tasks focused on making a model learn the relationship between multiple patches selected from an image.

Relative position. Context prediction uses spatial context as the source for the self-supervise task. The model is designed to classify the relative position [10] of two patches from the same image. The patches are sampled as follows: from an input image, the first patch is randomly selected and the other eight ones are cropped around the first to form a 3x3 grid. To make the task more difficult, all of the 8 neighbors of the initial sampled central patch have random noise added and are places in the 3x3 grid with gaps in between, as seen in Figure 4.4. Two CNNs are trained with shared weights, each of the network receiving one patch. In the end, the outputs of the CNNs are combined. The training can be seen as a 8-class classification problem, with the input being the pair of patches and the output the position from the 8 possibilities in the 3x3 grid.

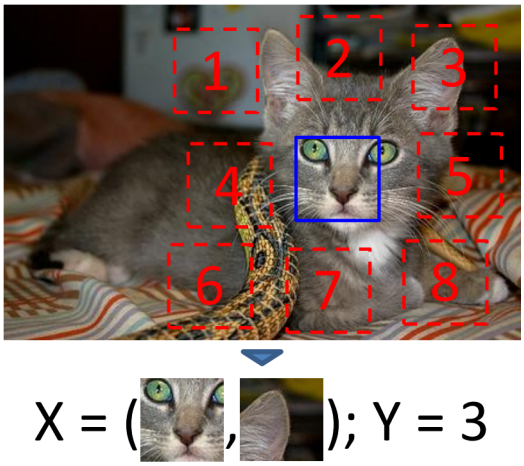

Figure 4.4: The 3x3 grid formed around the first randomly selected patch which generates 8 pairs denoting 8 different classes, with the first member of the pair always being the central patch. Figure courtesy of [10].

Jigsaw puzzles. Having a 3x3 grid already built for each image, the task can be made more difficult by using a training sample that includes all of the 9 patches, instead on only two at a time. With this idea in mind, solving Jigsaw puzzles [52] is the next presented pretext task. The 9 patches formed similar to the previous method are shuffled according to a randomly selected permutation from a predefined set, set which is predefined to control the difficulty of the task. Each patch is then fed to a separate CNN which shares weights with the other eight ones. The output of the nine CNNs is combined to generate a classification problem with the number of classes equal to the size of the permutation set.

Feature counting. The last presented method which exploits the relationship between patches is learning to count features [53]. Two transformations are used in the context of an equivariance relation: scaling and tiling. Scaling an image should not influence the number of visual primitives and tiling an image to a 2x2 grid followed by summing the features of each title should match the number of features from the original image. From Figure 4.5 we can see that starting from an image x, a down-sampling operation followed by a tilling to a 2x2 grid are performed. The 4 patches and the original down-sampled image are fed to separate CNNs with shared weights. The features counted by the 4 CNNs which correspond to the 4 patches are summed and the difference between this sum and the count generated by the CNN which was fed with the original image is minimized. To make the task even more difficult, a contrastive loss is added to promote the dissimilarity between another down-sampled random image y and the sum of the 4 patches generated from x.

4.1.3. Geometric transformation based methods

The pretext task of the next category of self-supervised image based methods is to discriminate between different versions of an image, obtained by applying one or more geometric transformations.

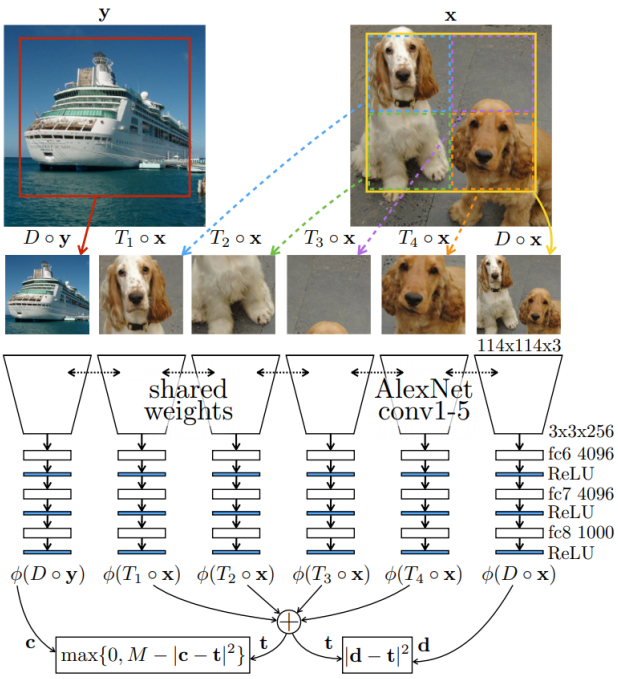

Figure 4.5: Learning to count architecture overview. The CNN used to extract the features which are then summed and compare is AlexNet. The constrative loss uses a experimentally determined constant scalar $M = 10$. Figure courtesy of [53].

Exemplar Convolutional Neural Networks. Exemplar Convolutional Neural Networks [13] are trained to discriminate between a set of surrogate classes. This classes are created by modifying an initial training set. This set was generated from unlabeled data by selecting fix size patches at regions with considerable gradient, mainly edges which are an indication of objects, from different images at multiple scales and positions. On each selected patch a variety of random elementary transformations such as translation, scaling or rotation are applied to generate multiple new patches which are considered part of the same surrogate class. A surrogate class is generated for each patch of the initial training set. A CNN is trained to discriminate between these surrogate classes.

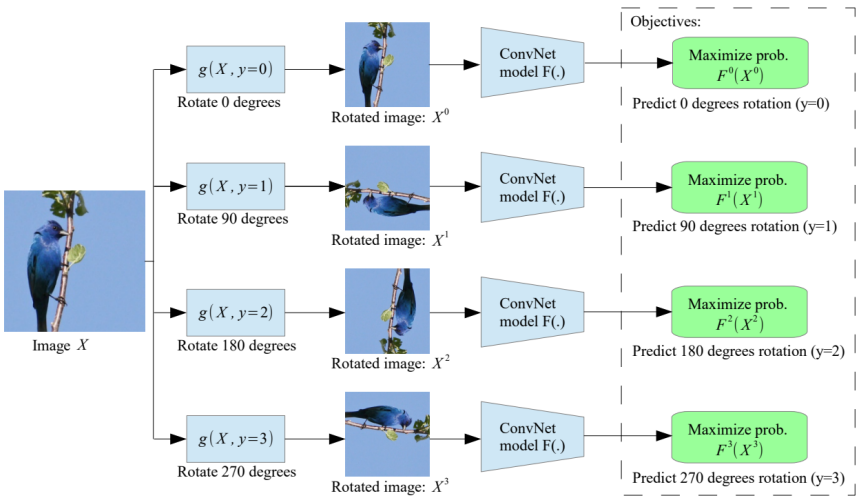

Figure 4.6: 2D RotNet overview with examples of the four possible rotations (0°, 90°, 180°, 270°). The ConvNet is trained as a 4-class image classification problem. Figure courtesy of [15].

Predicting image rotations. A straightforward pretext task which involves geometric transformations is to train a CNN to recognize what 2D rotation was applied on an image [15]. In Figure 4.6, 4 types of

rotation with multiples of 90° are displayed. RotNets are trained as a 4-class classification task and in the process of discriminating between the 4 types of rotations, semantic features are also learned.

4.1.4. Contrastive learning based methods

Contrastive learning represents a category of deep metric learning techniques [36] focused on teaching a deep neural network to distinguish between similar and dissimilar data examples. Positive and negative pairs needs to be created from the unlabeled training data. Based on the created pairs and using a contrastive loss [20, 37, 40, 65, 71], the network parameters are optimized.

Contrastive predictive coding. Contrastive predictive coding (CPC) [55] pretext task is to predict the future in latent space by using autoregressive models. The contrastive loss used by this paper is InfoNCE, loss inspired by the Noise Contrastive Estimation (NCE) employed in learning word embeddings [19]. InfoNCE uses cross-entropy loss to measure the performance of recognising the future samples from a bunch of negative ones. CPC components are an encoder g_{enc} that compresses the input x_t to the latent representation $z_t = g_{enc}(x_t)$ and an autoregressive decoder g_{ar}. When applying CPC on images, starting from a 256x256 image a 7x7 grid is created by using crops with 50% overlap. All patches are encoded using a ResNet architecture for generating a matrix $z_{i,j} = g_{enc}(patches)$. The autoregressive decoder makes predictions in connection to the activations from the latent space, in following rows top-to-bottom. A schematic overview of CPC can be seen in Figure 4.7.

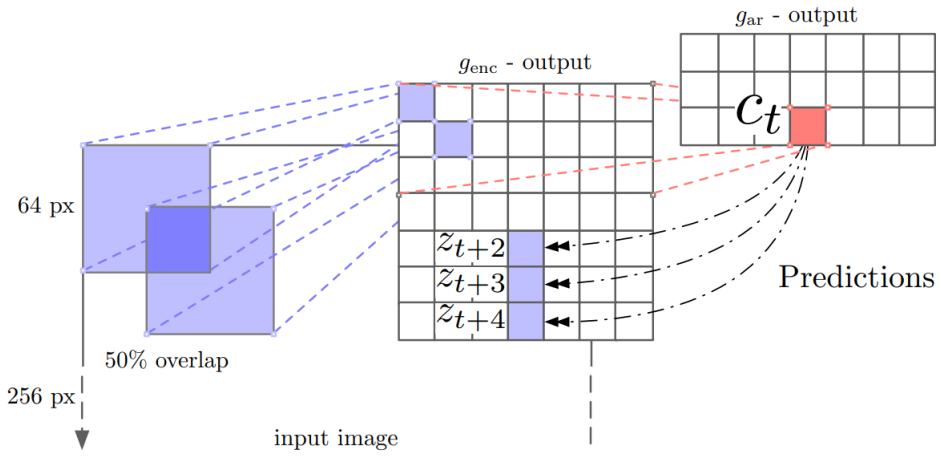

Figure 4.7: Contrastive Predictive Coding (CDC) when applied on an input images using ResNet 101 as the image encoder g_{enc}. From the input image of size 256x256, , crops of 64x64 with 32 pixels overlap are extracted and fed to the encoder. Figure courtesy of [55].

Momentum Contrast. Momentum Contrast (MOCO) [26] tackles contrastive learning as a dictionary look-up problem. The dictionary is represented by a dynamic moving queue of encoded data samples. There is a query encoder f_q and a momentum encoder f_k. From Figure 4.8, we can see f_q encoding x^{query} to $q = f_q(x^q)$ and f_k generating a list of dictionary keys $\{k_0, k_1, k_2...\}$ which are the projection of other data samples into a latent space. Note that both x^q and x^k are data samples. In the queue of size K, a positive example of x^q, namely k^+, exists. The positive sample $k+$ was generated by encoding with f_k a different augmented version of x^q. InfoNCE [55] contrastive loss is used to promote the similarity between q and k^+ given other (K-1) negative keys. The queue is updating using FIFO (First-in, first-out) protocol, with the newest batch replacing the last one when it is full. The gradient flows only through the query encoder and the momentum encoder is moving slowing by a momentum update, based on the query encoder.

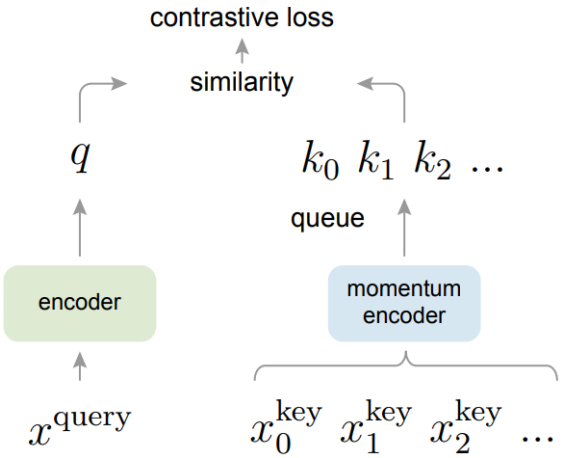

Figure 4.8: Momentum contrast illustration showing both the query encoder and the momentum encoder, with the similarity being computed between the encoded query q and the dictionary keys from the FIFO queue. Figure courtesy of [26].

SimCLR. A Simple Framework for Contrastive Learning of Visual Representations (SimCLR) [4] uses contrastive learning to maximize agreement between 2 augmented versions of the same image. The components of SimCLR can be analyzed in Figure 4.9 and consist of: a random data augmentation to generate t and t', a base encoder f to get the representation h, a projection head to arrive to z and finally a contrastive loss, named NT-Xent, to maximize the agreement between z_i and z_j. Different from MOCO, where the number of negatives pairs, derived from the queue size, is not limited by the batch size, SimCLR loss is dependent on the size of the batch. The negative pairs are created from the augmented versions of the current batch, therefore SimCLR relies on large batch sizes to generate a large enough number of negative pairs. From a batch of N examples, $2(N-1)$ negative pairs are created and the positive ones are represented by the pairs of the two augmented versions of each image from the batch.

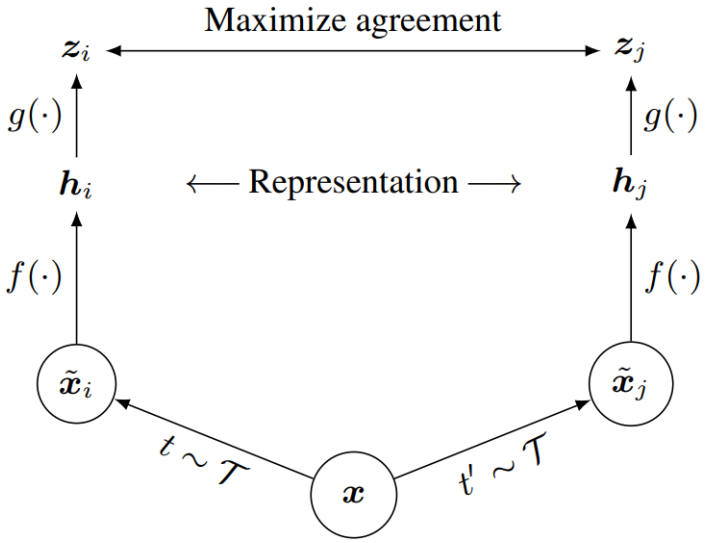

Figure 4.9: SimCLR schematic overview, illustrating the two spatial random transformations t, t', the CNN based encoder f and the fully-connected projector g. The representation h is used for down-stream tasks and the projection z for the contrastive loss during the self-supervised training. Figure courtesy of [4].

Improved Momentum Contrast. The improved version of Momentum Contrast (MOCO V2) [5] applies a couple of ideas from SimCLR to the MOCO framework. The paper finds out that using both a projection head and stronger data augmentation helps in improving the quality of the learned features.

4.2. Learning visual representations from videos

4.2.1. Generative based methods

For videos, generative based models learn visual features in the process of creating realistic videos. The pretext task of generating video is done not using the training videos labels.

Video Generative Adversarial Networks. Video generation with GANs (VGAN) [68] uses a spatio-temporal convolutional architecture to separate the video into two streams, the background and the foreground. In this two-stream architecture, the first stream models the static part of the video (background) and the second one the dynamic objects (foreground). In this way, the method shows that the scene dynamics can be a good signal for representation learning from unlabeled videos.

Video colorization. As seen applied on images, using colorization is a good pretext task for self-supervision. Unlike in images, Video colorization [69] pretext task is to copy colors from a reference colored frame to a target gray-scale frame by taking advantage of the temporal coherency between the reference and target frame. As presented in Figure 4.10, starting from a gray-scale video with a reference and a target frame, which are not far away in the video, a model generates embeddings for both of the frames using a CNN architecture with shared weights. Using the computed embeddings, a relation between f_i and f_j is established, which is going to be used to color c_j based on the referenced color c_i.

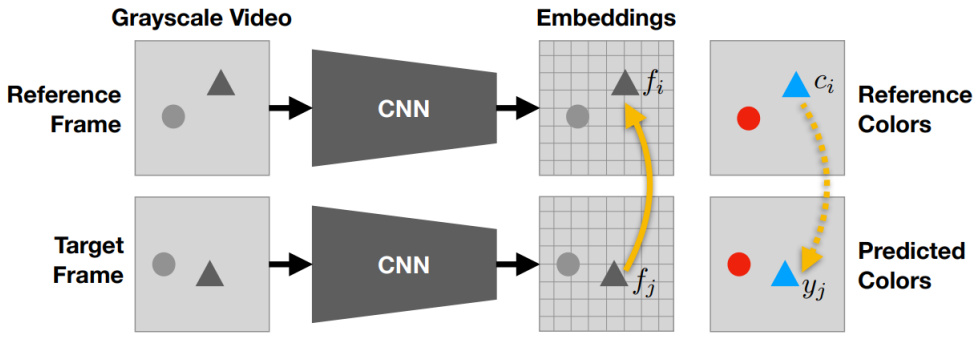

Figure 4.10: Video colorization model overview. The solid yellow line connection based on the learned embeddings is used to generate the dashed yellow arrow to predict the color of the target frame. Figure courtesy of [69].

4.2.2. Relationship between frames

Compared to static images, the relation between the frames of a video contains both spatial and temporal information. The sequentiality of the frames encodes temporal information that can be used to design pretext tasks for self-supervised feature learning.

Space-Time Cubic Puzzles. The Jigsaw puzzles detailed for image visual representation learning were transposed to a 3D version of arranging pieces of a space-time cuboid. Space-Time Cubic Puzzles [38] train 3D CNNs to arrange permuted 3D spatio-temporal crops. In Figure 4.11, we can see that the 3D crops are extracted from a 4-cell grid of either 2x2x1 (displayed in red, spatial dimension) or 1x1x4 (shown in blue, temporal dimension). The crops are randomly permuted and spatio-temporal jittering is applied.

Temporal Order Verification. Shuffle & Learn [48] is another self-supervised methods from videos which teaches a model to determine if a sequence of frames is in the correct order. The technique uses a Triplet Siamese Network with three 2D CNNs which receive a 3-frames tuple as input. The positive (correct order) and negative (incorrect order) tuples are selected from the frames with significant motion, based on optical flow. The tuples are shuffled and grouped in pairs. For the model to be able to correctly verify the temporal order, small differences between the frames needs to be capture, such as the movement of an object or person. By solving this task, semantic features can be learned.

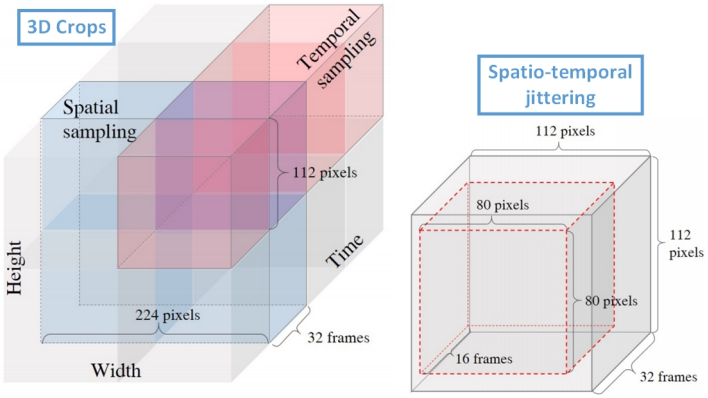

Figure 4.11: Space-Time Cubic Puzzles with both spatial sampling and temporal sampling for extracting the 3D crops. To increase the complexity of the self-supervised task, spatio-temporal jittering is added. Figure courtesy of [38].

Temporal Order Prediction. Order Prediction Networks (OPN) [44] take advantage of the temporal coherence in videos by creating a sequence sorting task. The frames sampled from an input video are shuffled and a CNN is trained to sort them on the temporal axis. To promote the learning of semantics features rather than low-level ones, the data sampling process contains spatial jittering and channel splitting. Figure 4.12 contains an overview of OPN pipeline, with the network architecture being made of two feature extraction stages, the first encoding for each frame and the second for each pair of frames. Starting from a sequence of 4 frames, the order prediction can be viewed as a 12-class classification problem.

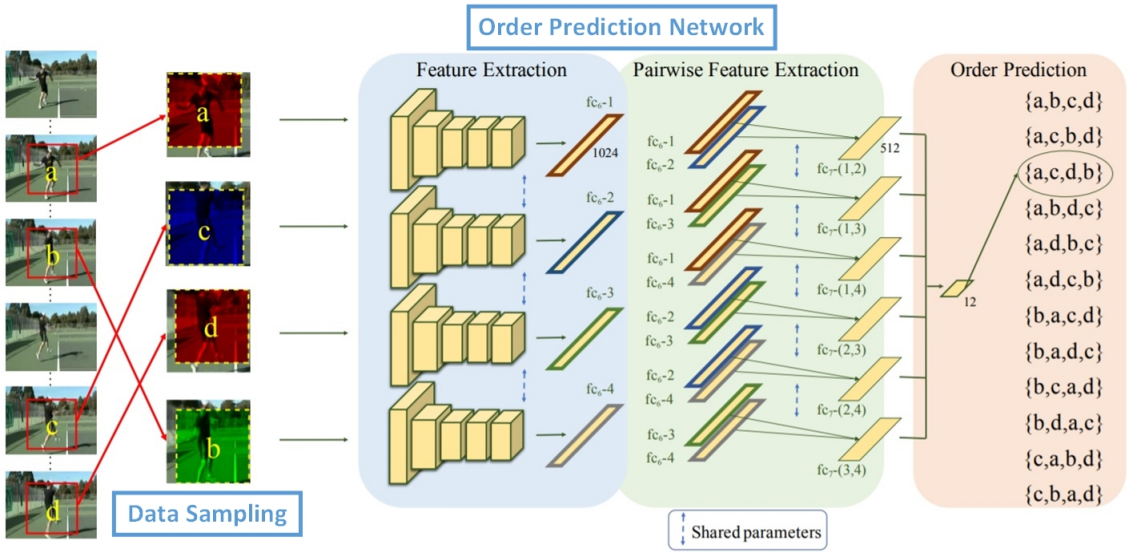

Figure 4.12: Order Prediction Networks overview, with two main stages. The Data Sampling stage creates the tuples by shuffling the sampled frames and applying spatial jittering with channel splitting. The Network stage have CNNs which are trained on the input tuple, first extracting features from frames and then from pairs of frames. The end of the entire pipeline is the Order Prediction by selecting the right sequence from 12 possibilities. Figure courtesy of [44].

Arrow of Time. Another way of exploiting the sequentiality of frames is by understanding the Arrow of Time (AoT) [70] in videos. The AoT is composed of cues that makes videos look like they are playing either forward or backward. This cues can be physics based (e.g. smoke rises up) or event reasoning based (e.q. cracking an egg cannot be reversed). Predicting the AoT, playing forward or backward, is done by using a Temporal Class-Activation-Map Network (T-CAM). The network is design to enable the visualization of the learned features and takes optical flow as input, not the RGB flow. In Figure 4.13, T-CAM network is summarized with the CNNs block introduced for late temporal fusion and global average pooling (GAP) followed by logistic regression for classification.

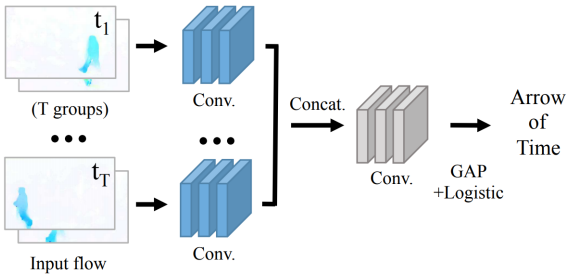

Figure 4.13: Temporal Class-Activation-Map (T-CAM) Network overview. The blue CNNs with shared weights are VGG-16 based and the Conv block at the end contains three convolutional layers followed by global average pooling. Figure courtesy of [70].

4.2.3. Geometric transformation based method

Geometric transformations can also be used as a pretext task for self-supervised feature learning from videos. Similar to the 2D RotNets analyzed at the image section, 3DRotNets [33] were design to learn spatio-temporal features from unlabeled videos. 3DRotNet uses the same list of possible rotations, 0°, 90°, 180°, 270°, replacing the 2D CNN with a 3D one. The training of the network is conducted as a 4-class classification task, with the pipeline displayed in Figure 4.14.

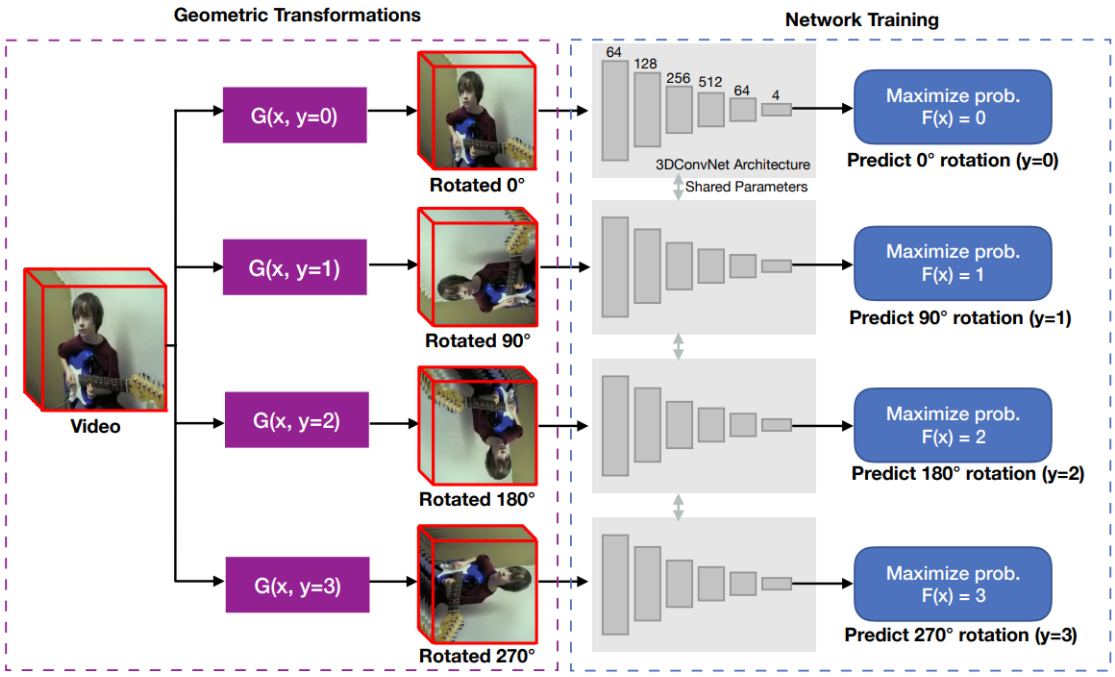

Figure 4.14: 3D RotNet pipeline. In Geometric Transformations stage the input volume is being rotated with four multiplea of 90° and in Network Training stage the 3D CNN learns to recognize the rotation type. Figure courtesy of [33].

4.2.4. Contrastive learning based methods

Dense Predictive Coding. Following the same future prediction idea from the image feature learning proposed by CPC, Dense predictive coding (DPC) [21] learns a dense encoding of spatio-temporal blocks. Pixel-wise prediction of future frames is difficult due to appearance changes, camera shaking, etc. In this way, it does not promote the learning of high-level semantics. Predicting features, instead of frames, within a short temporal windows is needed. The model is trained with a contrastive loss to select the correct future state from a group of distractors. This method overview can be seen in Figure 4.15, with the prediction task observing ta $2.5s$ of video segments from x_1 to x_t and trying to generate the future $1.5s$. The video segments $x_1, \dots, x_t$ are encoded with a function f to the latent representations z_i. An aggregation function g is then used to obtain c_t, as the temporal aggregation of $z_1, \dots, z_t$. The future clip representation is made with the function ϕ, based on the aggregated embedding c_t. The

predicted representation $\hat{z}_{t+1} = \phi(c_t)$ and the ground truth representation z_t will form the positive pair for the contrastive loss. The loss used is a variant of NCE [49] that considers positive and negative pairs in regards to both temporal and spatial sense. The only positive pair for the predicted vector $\hat{z}_{i,k}$ is $(\hat{z}_{i,k}, z_{i,k})$ at the same time step i and the same spatial index k. For fine-tuning on a action recognition down-stream task, they used a ten times smaller learning rate for the CNN backbone architecture compared to the one used for the linear classifier.

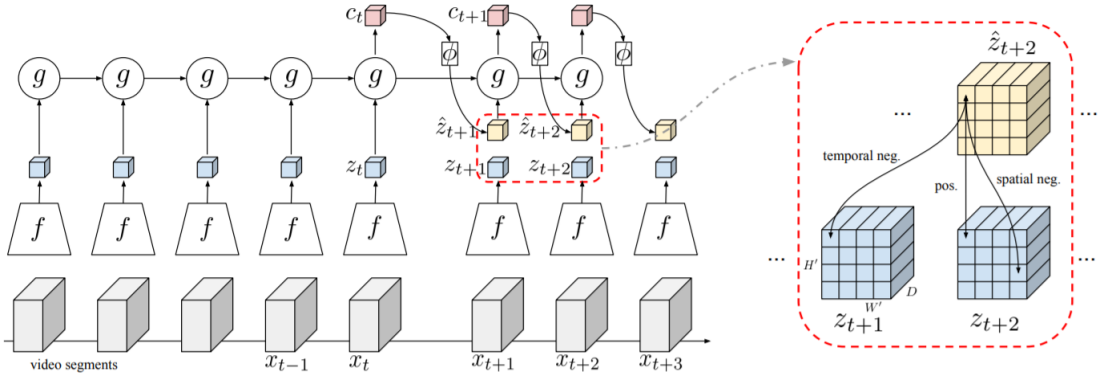

Figure 4.15: Summary of Dense Predictive Coding method, with the left part presenting the architecture for predicting the future clip representation and the right part analyzing the contrastive loss, with positive and negative pairs constructed in regards to both temporal and spatial axis. Figure courtesy of [21].

Temporally Coherent Embeddings. Explicitly enforcing temporal coherency in the embedding space generates high-quality visual representations. Temporally Coherent Embeddings (TCE) [39] promote the similarity between nearby frames from the same video and the dissimilarity with other frames from different videos. The backbone architecture used to encode the video frames is a 2D CNN and the cosine similarity is computed in the resulted embedding space. The Noise Contrastive Estimation (NCE) [49] loss takes the temporally neighbouring frames of one video as positive examples and all frames from other videos as negative samples. Figure 4.16 shows how the contrastive loss works in the embedding space to promote the attraction between frames from $\mathcal{V}^1$ and the separation between frames from $\mathcal{V}^1$ and all other frames from $\mathcal{V}^2, \ldots, \mathcal{V}^m$. Since the backbone architecture is a 2D CNN, the evaluation on action recognition is done by averaging the output of the features generated by passing through the network a couple of video frames. The fine-tuning is done with the same learning rate for both the pre-trained backbone CNN architecture and the linear classifier.

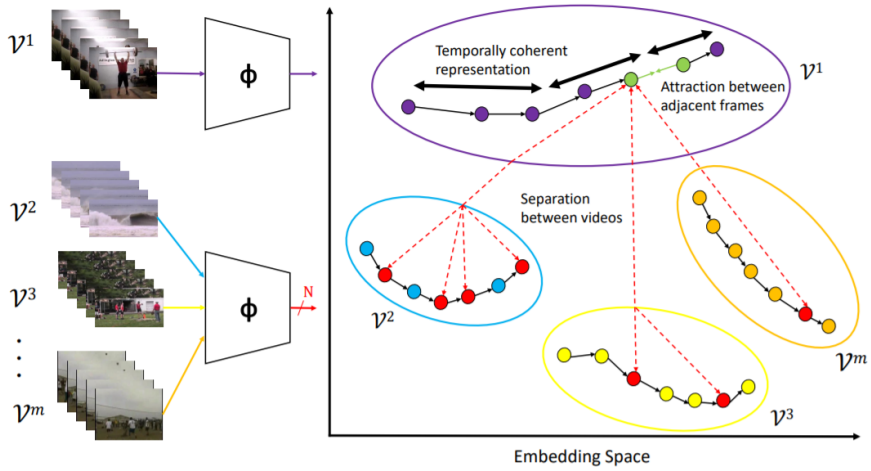

Figure 4.16: Temporally Coherent Embeddings overview with an illustration of the constrastive loss in the Embedding Space. The loss enforces an anchor frame from $\mathcal{V}^1$ to attract with an adjacent frame from the same video and to repel with N negative frames sampled from all other videos $\mathcal{V}^2, \ldots, \mathcal{V}^m$. Figure courtesy of [39].

List of Figures

2.1 Multi-layered perceptron . . . 14
2.2 Activation function . . . 14
2.3 Stochastic gradient descent . . . 15
2.4 Cosine Annealing . . . 16
2.5 Convolutional Neural Networks . . . 16
2.6 Convolution operator . . . 17
2.7 Kernels as feature detectors . . . 18
2.8 Spatial Pooling . . . 18

3.1 Common Deep Neural Network architectures for videos . . . 19
3.2 Two Stream Networks . . . 20
3.3 Long-Term Recurrent Convolutional Networks . . . 20
3.4 3D Convolutional Networks . . . 21
3.5 3D Residual Networks . . . 21
3.6 3D-Fused Two-Stream Networks . . . 21
3.7 HMDB51 and UCF101 . . . 22
3.8 Kinetics 400 . . . 23

4.1 Bidirectional Generative Adversarial Networks . . . 26
4.2 Context Encoders . . . 26
4.3 Image colorization . . . 26
4.4 Relative position . . . 27
4.5 Feature counting . . . 28
4.6 Predicting image rotations . . . 28
4.7 Contrastive predictive coding . . . 29
4.8 Momentum Contrast . . . 30
4.9 SimCLR . . . 30
4.10 Video colorization . . . 31
4.11 Space-Time Cubic Puzzles . . . 32
4.12 Temporal Order Prediction . . . 32
4.13 Arrow of Time . . . 33
4.14 Predicting video rotations . . . 33
4.15 Dense predictive coding . . . 34
4.16 Temporally Coherent Embeddings . . . 34

Bibliography

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

[2] Zhuowei Cai, Limin Wang, Xiaojiang Peng, and Yu Qiao. Multi-view super vector for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 596–603, 2014.

[3] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[5] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[6] Aidan Clark, Jeff Donahue, and Karen Simonyan. Adversarial video generation on complex datasets. *arXiv*, pages arXiv–1907, 2019.

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

[8] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of machine learning research*, 2(Dec):265–292, 2001.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[10] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015.

[11] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

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

[13] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1734–1747, 2015.

[14] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1933–1941, 2016.

[15] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[18] Leo J Grady and Jonathan R Polimeni. *Discrete calculus: Applied analysis on graphs for computational science*. Springer Science & Business Media, 2010.

[19] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010.

[20] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[21] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[22] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Learning spatio-temporal features with 3d residual networks for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 3154–3160, 2017.

[23] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6546–6555, 2018.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[25] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[26] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[27] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[28] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[29] Berthold KP Horn and Brian G Schunck. Determining optical flow. In *Techniques and Applications of Image Understanding*, volume 281, pages 319–331. International Society for Optics and Photonics, 1981.

[30] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[31] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[32] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2012.

[33] Longlong Jing and Yingli Tian. Self-supervised spatiotemporal feature learning by video geometric transformations. *arXiv preprint arXiv:1811.11387*, 2(7):8, 2018.

[34] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[35] Hirokatsu Kataoka, Tenga Wakamiya, Kensho Hara, and Yutaka Satoh. Would mega-scale datasets further enhance spatiotemporal 3d cnns? *arXiv preprint arXiv:2004.04968*, 2020.

[36] Mahmut Kaya and Hasan Şakir Bilge. Deep metric learning: A survey. *Symmetry*, 11(9):1066, 2019.

[37] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2020.

[38] Dahun Kim, Donghyeon Cho, and In So Kweon. Self-supervised video representation learning with space-time cubic puzzles. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8545–8552, 2019.

[39] Joshua Knights, Anthony Vanderkop, Daniel Ward, Olivia Mackenzie-Ross, and Peyman Moghadam. Temporally coherent embeddings for self-supervised video representation learning. *arXiv preprint arXiv:2004.02753*, 2020.

[40] Wouter M Kouw, Marco Loog, Lambertus W Bartels, and Adriënne M Mendrik. Learning an mr acquisition-invariant representation using siamese neural networks. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 364–367. IEEE, 2019.

[41] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[42] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *2011 International Conference on Computer Vision*, pages 2556–2563. IEEE, 2011.

[43] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[44] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 667–676, 2017.

[45] Claude Lemaréchal. Cauchy and the gradient method. *Doc Math Extra*, 251:254, 2012.

[46] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[47] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[48] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, pages 527–544. Springer, 2016.

[49] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in neural information processing systems*, pages 2265–2273, 2013.

[50] Mathew Monfort, Alex Andonian, Bolei Zhou, Kandan Ramakrishnan, Sarah Adel Bargal, Tom Yan, Lisa Brown, Quanfu Fan, Dan Gutfreund, Carl Vondrick, et al. Moments in time dataset: one million videos for event understanding. *IEEE transactions on pattern analysis and machine intelligence*, 42(2):502–508, 2019.

[51] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[52] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016.

[53] Mehdi Noroozi, Hamed Pirsiavash, and Paolo Favaro. Representation learning by learning to count. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5898–5906, 2017.

[54] Noboru Ohta. Correspondence between cielab and cieluv color differences. *Color Research & Application*, 2(4):178–182, 1977.

[55] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[56] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[57] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[58] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[59] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[60] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[61] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[62] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[63] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

[64] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[65] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in neural information processing systems*, pages 1857–1865, 2016.

[66] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[67] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[68] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Advances in neural information processing systems*, pages 613–621, 2016.

[69] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. In *Proceedings of the European conference on computer vision (ECCV)*, pages 391–408, 2018.

[70] Donglai Wei, Joseph J Lim, Andrew Zisserman, and William T Freeman. Learning and using the arrow of time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8052–8060, 2018.

[71] Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pages 1473–1480, 2006.

[72] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.

[73] Richard Zhang, Phillip Isola, and Alexei A Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1058–1067, 2017.

[74] Oxford Google Deep Mind Zisser A. Self-supervised learning. accessed link, 2018. URL `https://project.inria.fr/paiss/files/2018/07/zisserman-self-supervised.pdf`.