Self-supervised representation learning and applications to image and video analysis

Julien Denize

Normandie Université

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN NORMANDIE

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité **INFORMATIQUE**

Préparée au sein de l'**INSA Rouen Normandie**

Self-supervised representation learning and applications to image and video analysis

Présentée et soutenue par

JULIEN DENIZE

Thèse soutenue le 18/12/2023

devant le jury composé de :

M. LAURENT NAJMAN	PROFESSEUR DES UNIVERSITÉS - ESIEE Paris	Rapporteur
M. MASSIH-REZA AMINI	PROFESSEUR DES UNIVERSITÉS - Université Grenoble Alpes	Membre
M. ADRIEN CHAN HON TONG	DOCTEUR - ONERA	Membre
M. JAONARY RABARISOA	INGENIEUR DE RECHERCHE - CEA LIST, Palaiseau	Membre Co-encadrant
MME ASTRID SABOURIN	DOCTEUR - CEA LIST, Palaiseau	Membre
MME CATHERINE ACHARD	PROFESSEUR DES UNIVERSITÉS - Sorbonne Université	Président du jury
M. ROMAIN HERAULT	PROFESSEUR DES UNIVERSITÉS - Université de Caen	Directeur de thèse

Thèse dirigée par **ROMAIN HERAULT** (LABORATOIRE D'INFORMATIQUE DE TRAITEMENT DE L'INFORMATION ET DES SYSTEMES)

MI∫IΣ

litis

Contents

Remerciements

Je tiens à remercier le CEA List, en particulier le service SIALV et le laboratoire LVA, de m'avoir accueilli pendant plus de 3 ans, d'abord lors de mon stage de recherche, puis tout au long de ma thèse. J'ai eu le privilège de découvrir une communauté d'ingénieurs chercheurs brillants dans une atmosphère bienveillante. Cet environnement sain est également dû aux chefs de laboratoire et de service, et je tiens à remercier Angélique, Quoc-Cuong et Patrick. J'ai également pu apprécier la compagnie de nombreux doctorants et stagiaires au sein du service avec lesquels j'ai pu collaborer sur le plan scientifique, mais aussi partager des moments conviviaux autour d'une tasse de café, d'une table de billard ou sur le terrain de foot. Plus proche de moi, je tiens à exprimer ma gratitude envers Rémi, Quentin, Guillaume, Astrid et Alice. Merci pour ces moments, ces discussions et ces rires, ils ont été essentiels à mon bien-être en thèse.

Je souhaite également remercier mes encadrants tout au long de mon séjour au LVA. Merci à Angélique, Romaric et Fabian de m'avoir accordé votre confiance il y a presque 4 ans lorsque j'ai rejoint le LVA en stage. Ce stage a été crucial pour me mettre en confiance et découvrir le monde de la recherche. Ensuite, un grand merci à Jaonary et Astrid de m'avoir encadré en thèse au CEA, mais également à Romain qui depuis Rouen puis Caen, malgré la distance, a su m'accompagner. Chacun d'entre vous m'a guidé et accompagné à sa façon pour m'aider à devenir un jeune chercheur.

Un remerciement à l'ensemble des membres du jury qui m'ont honoré en me permettant de présenter mes travaux de thèse. Je tiens à remercier Catherine et Laurent d'avoir accepté d'être rapporteurs de thèse, un travail conséquent, et d'avoir formulé des retours pertinents et bienveillants. Merci également à Massih-Reza et Adrien d'avoir accepté, tout comme Catherine et Laurent, d'évaluer mon travail.

Je tiens aussi à exprimer ma reconnaissance envers ma famille. Mes remerciements ne se limitent pas à ces trois dernières années de thèse mais s'étendent sur l'ensemble de ma vie, en particulier mes études supérieures qui m'ont amené à aujourd'hui. Merci à mes parents de m'avoir permis d'accéder à ces études, de m'avoir encouragé à me dépasser sans imposer. Je n'ai pas toujours été facile mais vous m'avez accompagné malgré mes peurs, mes doutes et dans mes aspirations. Un merci aussi à mes frères Jérémy et Mickaël

ansi que ma sœur Béatrice également présents dans ces moments. Sans vous, je n'aurais pas pu en arriver là.

Ensuite, merci à mes amis. D'abord un merci spécial à Florentin et Émilien, votre amitié a été un roc sur lequel m'appuyer. J'espère pouvoir contribuer autant à votre bonheur que vous l'avez fait pour le mien. Merci également Maxence pour ces années au lycée jusqu'à maintenant et le plaisir de te revoir quand je me rends disponible en Normandie. Un grand merci à mes amis de prépa Nathan, Antoine, Maël, Théotime, Aurélien, Damien, Camille, Thibault, Gaspard, Morgane, et tant d'autres sans qui je n'aurais pas tenu en prépa et en période de Covid. Merci pour toutes ces journées, weekends, semaines passées ensemble ces 8 dernières années. Je n'oublie pas et inclus également dans ces remerciements les personnes formidables rencontrées grâce à eux, notamment Olivia et Louise. Merci à mes amis d'école Rémi, Maxime, Océane, Marie-Jeanne et Valentin avec qui j'ai affronté des projets, participé à la vie associative et partagé de belles années jusque maintenant. Merci à mes amis d'abord In Game et maintenant In Real Life, en particulier Romain, Benoit, Nicolas, Xavier, Max, Micka et Teddy pour ces moments d'évasions mais également de rien pour les services.

J'ai conscience de la chance que j'ai de vous avoir dans ma vie, d'être entouré de personnes bienveillantes et je vous en remercie.

Lexicon

Acronyms

AI	Artificial Intelligence
AP	Average Precision
AS	Action Spotting
CL	Contrastive Learning
CNN	Convolutional Neural Network
CV	Computer Vision
CS	Computer Science
DL	Deep Learning
DNN	Deep Neural Network
FC	Fully Connected
FN	False Negatives
FP	False Positives
IoU	Intersection over Union
KD	Knowledge Distillation
k-NN	k-Nearest Neighbors
mAP	mean Average Precision
ML	Machine Learning
MLP	Multi-Layer Perceptron
MM	Masked Modeling
MIM	Masked Image Modeling
MVM	Masked Video Modeling
NCE	Noise Contrastive Estimation
NN	Neural Network
OD	Object Detection
OF	Optical Flow
ReL	Relational Learning
ReLU	Rectified Linear Unit
RL	Representation Learning
SCE	Similarity Contrastive Estimation

SGD	Stochastic Gradient Descent
SL	Supervised Learning
SSL	Self-Supervised Learning
TAD	Temporal Action Detection
TAL	Temporal Action Localization
TN	True Negatives
TP	True Positives
UL	Unsupervised Learning
VT	Vision Transformer

Notation

a	A scalar
$\boldsymbol{a}$	A vector
a_i	An element of a vector
$\mathbf{A}$	A matrix
$\mathrm{A}_{i,j}$	An element of a matrix
$\mathscr{A}$	A set
$\mathbb{A}$	A space
$\mathbb{R}$	Space of real numbers
$\|\|\boldsymbol{x}\|\|$	L_2 norm of $\boldsymbol{x}$
$\boldsymbol{x}^{\mathrm{T}}$	Transpose of $\boldsymbol{x}$
$\boldsymbol{x} \sim \mathrm{P}$	$\boldsymbol{x}$ drawn from the probability distribution P
$\mathbb{1}_{\text{condition}}$ $\mathbb{1}(\text{condition})$	Characteristic function evaluated to 1 if condition is true else 0

Chapter 1

Preface

Sommaire

1.1 Context

For the past decade, *Artificial Intelligence* (AI) has been at the forefront of a potential economic revolution. This field can be defined as a collection of theories and techniques that aim to perceive, synthesize, and infer information from machines [Wik23]. While AI initially fell short of industry expectations, it is now becoming increasingly prevalent in our daily lives through the use of facial recognition software, chatbots, automatic speech recognition, social media filters, and more.

The focus of this work lies within the subdomain of AI known as *Computer Vision* (CV), which deals with the processing of images or videos obtained from various sensors. Specifically, this work is focused on *Deep Learning* (DL) methods, where AI is implemented using artificial neural networks.

A *Feedforward Neural Network* (NN) consists of interconnected layers of neurons that perform computations. These computations begin with an input, such as images or videos in the case of CV, and pass through the NN to produce an output. To enable the network to perform a specific task, its neurons are trained to provide outputs that are relevant to that particular task. Examples of such tasks include image recognition (e.g., recognizing cats or dogs), image generation (e.g., creating a fictional Monet painting of a Star Wars spaceship), action spotting (e.g., detecting the timestamp at which a goal is scored in a football match), and various other tasks like action recognition, action localization, depth prediction, facial recognition, reidentification, and object detection.

To train a NN in a supervised manner to solve a specific task such as image classification of cats and dogs, a labeled dataset is used. This dataset contains two elements: A collection of *input* data, the images, and the corresponding *labels* associated with that data, e.g. if the image contains a dog or a cat.

Since a NN trained is specialized to the specific task it is taught on, it would be necessary to train an entirely new network for each different task one wishes to solve. However, even though the final tasks may be different, two NNs trained, for example, to solve action spotting and action recognition, respectively, are likely to learn similar features in some layers of their neural architecture, such as edges, object structures, the arrow of time, or interactions.

Building upon this observation, a subfield of CV has emerged to explore methods for *pretraining* a NN. Pretraining involves training a neural network to acquire a general representation in its layers, which can then be utilized as part of a broader network that is specialized in a particular task. By pretraining a network to learn a general representation, subsequently, specialized networks can focus on learning task-specific features without the need to learn general semantic information from scratch.

This approach significantly reduces the cost of solving tasks for three main reasons:

1. Pretraining is performed once and the learned backbone can be reused for subsequent specialized training sessions.
2. Specializing a pretrained network for a specific task can be accomplished through faster training.
3. Less labeled data is required for specialized tasks since the pretrained model has already learned some general concepts [CTM+21].

To achieve the most general representation for a pretrained network, the task chosen for pretraining should be as general as possible. This can be accomplished through two different approaches: the type of objective used for pretraining and the amount and quality of data seen during the pretraining process.

Regarding the type of objective, two general directions are commonly pursued: *supervised learning* (SL) and *self-supervised learning* (SSL). In SL, the network has access to labeled data for a given task and is pretrained by learning to predict these labels. On the other hand, SSL involves providing unlabeled data and designing a pretext task to provide supervision. This pretext task leverages the entire input or a portion of it to create an associated label. For example, one can rotate images [GSK18] or videos [JT18] by a certain angle and train a network to predict the rotation angle as the output. After pretraining, the task-specific part of the network, usually the last layer, is discarded, while the rest is retained for specializing in other tasks.

Regardless of the learning paradigm, a larger dataset generally leads to better learning of a general representation [DBK+21, ODM+23]. Having more data reduces bias toward the data used for training and helps to generalize to unseen data by learning concepts contained in a larger dataset.

While SSL has demonstrated its superiority over SL in producing general features for text-based tasks [DCLT19], its effectiveness in image and video tasks is still being researched. However, the gap between SSL and SL pretraining has narrowed over the last few years, and SSL shows better generalizability to unseen data [CTM+21, FFX+21, TSWW22]. Furthermore, obtaining labeled data can be extremely expensive, both in terms of financial costs for hiring annotators and time costs for datasets containing a large number of images, or requiring expert annotation for domains like medical or satellite data. Therefore, SSL appears to be the most promising approach for learning general representations for NNs at scale [ODM+23], as it is data-efficient and demonstrates better generalization to unseen data.

Our work lies in this context and proposes original SSL approaches for both image and video modalities. It takes part in a several years collaboration between CEA List, more specifically the Laboratoire de Vision et d'Apprentissage pour l'analyse de scène (LVA), and Normandie Université, more specifically the Laboratoire d'Informatique, de Traite-

ment de l'Information et des Systèmes (LITIS). Both laboratories are interested in image and video analysis for various applications via Deep Learning approaches and successfully led a previous PhD thesis to completion defended by Guillaume Lorre on the subject of self-supervised representation learning for video analysis.

Given that videos are more complex than images, and DL methods already consume significant computational resources for images, our work also focuses on developing resource-efficient methods for pretraining.

1.2 Structure of the manuscript and Contributions

1.2.1 Structure of the manuscript

This Manuscript has been organized to show our work as follows.

- Chapter 1: We present the context of this thesis, the structure of the manuscript and the contributions.
- Chapter 2: We introduce Representation Learning (RL) in the context of Deep Learning for Computer Vision from the basics to Machine Learning to why pretraining matters and how Self-Supervised Learning (SSL) is an interesting approach. It also raises the different challenges faced by the different SSL families. Finally, the different downstream tasks used in this work are presented.
- Chapter 3: We dwelve into the related work of SSL to highlight the consequent research that has been produced in SSL for Images and Videos but also the remaining challenges and issues that necessitate further studies.
- Chapter 4: We introduce Similarity Contrastive Estimation (SCE) a Soft Contrastive Learning objective to perform Image RL. It bridges the gap between CL and Relational Learning (ReL) to consider relations between negative pairs. We show theoretically that our approach aims to solve both objectives and empirically prove the superiority of our approach over the two.
- Chapter 5: We extend SCE to perform Video RL and more specifically to learn one output representation for a short clip of a few seconds. In this study, we showed that for videos, our approach is superior to only CL or ReL, especially for generalization.
- Chapter 6: We present COMEDIAN, a SSL and Knowledge Distillation (KD) approach to perform Video RL of transformers and more specifically to output several local temporal representations in a long clip of half or one minute. It is evaluated on the Action Spotting task. We showed that not only pretraining a transformer is necessary for this task but it also considerably reduced time convergence.

- Chapter 7: We conclude our work and offer perspectives for future work and research directions.

1.2.2 Contributions

The contributions presented in this thesis led to the following peer-reviewed and published works:

- International Conferences:
  - **Julien Denize**, Jaonary Rabarisoa, Astrid Orcesi, Romain Hérault, Stéphane Canu. "Similarity Contrastive Estimation for Self-Supervised Soft Contrastive Learning". In the *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2023. [DRO+23]
- International Journals:
  - **Julien Denize**, Jaonary Rabarisoa, Astrid Orcesi, Romain Hérault. "Similarity Contrastive Estimation for Image and Video Soft Contrastive Self-Supervised Learning". In *Machine Vision and Applications (MVAP)*, 2023. [DROH23]
  - Adrien Maglot, Astrid Orcesi, **Julien Denize**, Quoc-Cuong Pham. "Individual locating of soccer players from a single moving view". In *Sensors*, 2023. [MODP23]
- National Conferences:
  - **Julien Denize**, Jaonary Rabarisoa, Astrid Orcesi, Romain Hérault and Stéphane Canu. "Estimation Contrastive de la Similarité pour un Apprentissage Flou Auto-Supervisé". In the *Conférence sur l'Apprentissage automatique (CAp)*, 2022. [DRO+22]
- International challenges and Workshops:
  - **Julien Denize**, Mykola Liashuha, Jaonary Rabarisoa, Astrid Orcesi, Romain Hérault. "Long-Context Transformer Pretraining Through Spatio-Temporal Knowledge Distillation for Action Spotting". *Action Spotting SoccerNet Challenge 2023*, 5th out of 12 teams.
  - **Julien Denize**, et al. "COMEDIAN: Self-Supervised Learning and Knowledge Distillation for Action Spotting using Transformers". *IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, 2024.[DLR+24]

- Under review:
  - Anthony Cioppa, **Julien Denize**, et al. "SoccerNet 2023 Challenges Results". *arXiv*, abs/2309.06006, 2023. [CGS+23]

It also led to technical contributions some of which were made publically available:

- *SCE* [Den23a]: Repository to reproduce SCE results on images. Available on GitHub[1].
- *Eztorch* [Den23b]: library to perform SSL learning and finetuning to downstream tasks. Our 3 main academic contributions are reproducible thanks to this repo. Available on GitHub[2].
- *Torchaug* [Den23c]: Library to compute efficient CPU/GPU and per-sample/batched data augmentations. Available on GitHub[3].

[1] SCE repository: `https://github.com/CEA-LIST/SCE`.
[2] Eztorch repository: `https://github.com/juliendenize/eztorch`.
[3] Torchaug repository: `https://github.com/juliendenize/torchaug`.

Chapter 2

Introduction to Representation Learning

Sommaire

2.1 Machine Learning basics

2.1.1 What does learning mean in AI ?

Machine learning (ML) is a set of algorithms that aim to learn from data. To design such algorithms, it is necessary to understand the concept of "learning."

> *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E."* - Tom Mitchell [Mit97].

This definition provides a general understanding of learning for a computer program. In practice, when designing machine learning algorithms, one must determine the task, experience, and performance measure to be used.

In the subsequent subsections, we will formally define general concepts of machine learning algorithms applied to Computer Vision (CV) focusing specifically on our work.

2.1.2 Datasets

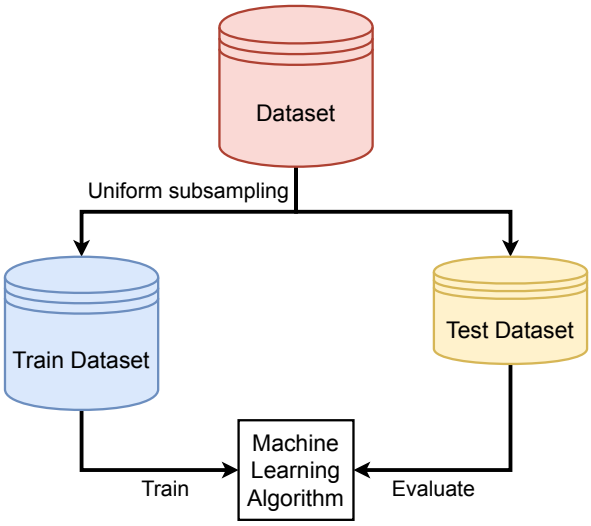


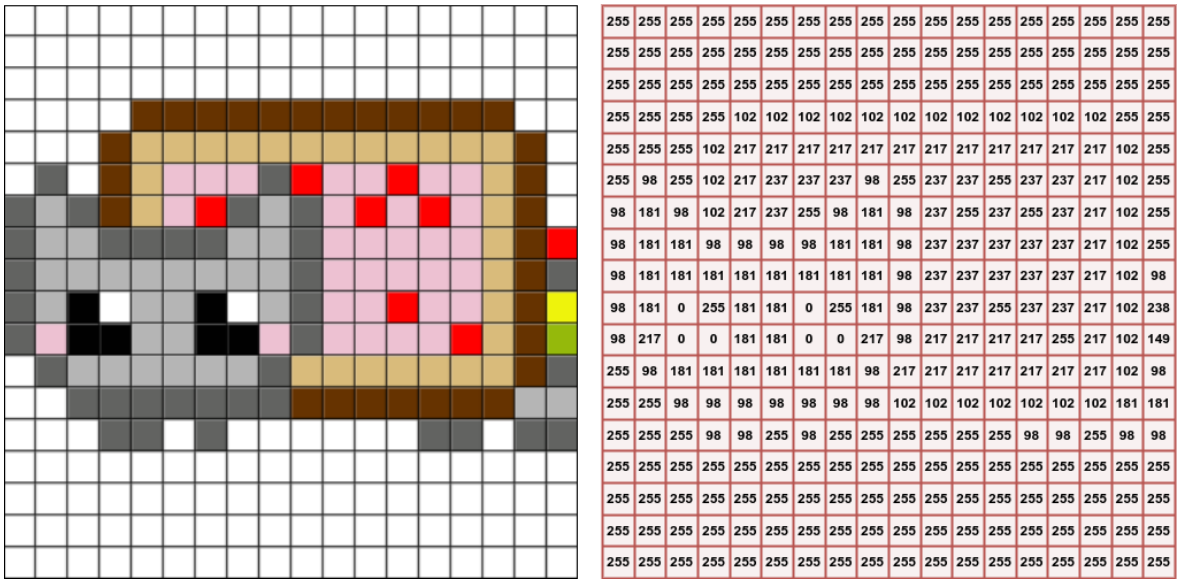
Figure 2.1: A dataset is split into train and test sets. The training set is used to train a Machine Learning algorithm evaluated on the test set.

ML algorithms operate on examples, also referred to as *input* data, to produce an *output* and solve a specific task. An example is represented as a d-dimensional vector, denoted as $\boldsymbol{x} \in \mathbf{R}^d$, where d represents the number of *features*.

A dataset is a collection of examples. A dataset consisting of n examples, each of dimension d, can be described as a matrix $\mathbf{X} \in \mathbb{R}^{n,d}$. Each example can be associated with a vector label of dimension m, denoted as $\boldsymbol{y} \in \mathbb{R}^m$, and all labels can be represented as a matrix $\mathbf{Y} \in \mathbb{R}^{n,m}$.

Typically, the dataset $(\mathbf{X}, \mathbf{Y})$ is divided into at least two splits: the training split $(\mathbf{X}^{(train)}, \mathbf{Y}^{(train)})$ and the test split $(\mathbf{X}^{(test)}, \mathbf{Y}^{(test)})$. This separation is illustrated in Fig. 2.1. The ML model is first trained on the training split and then evaluated on the test split. The purpose of this split is to study the generalization capability of the ML algorithm to unseen data. The evaluation and associated metrics depend on the specific task for which the algorithm is designed.

In this work, our focus is on tasks within the CV domain, specifically image and video analysis, therefore the datasets contain images or videos. We do not consider audio for videos and solely concentrate on the visual aspect. Consequently, videos can be viewed as sequences of consecutive images. In Computer Science (CS), images are represented as two-dimensional arrays of pixels, which are finite discrete quantities that describe the local intensity of an image. In the context of images and videos, each pixel is typically considered as one feature. Thus, a full HD RGB image would have $3 \times 1920 \times 1080 = 6,220,800$ features, and a full HD video with 10 frames would have $62,208,000$ features.



255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	102	102	102	102	102	102	102	102	102	102	102	102	255	255
255	255	255	102	217	217	217	217	217	217	217	217	217	217	217	217	102	255
255	98	255	102	217	237	237	237	98	255	237	237	255	237	237	217	102	255
98	181	98	102	217	237	255	98	181	98	237	255	237	255	237	217	102	255
98	181	181	98	98	98	98	181	181	98	237	237	237	237	237	217	102	255
98	181	181	181	181	181	181	181	181	98	237	237	237	237	237	217	102	98
98	181	0	255	181	181	0	255	181	98	237	237	255	237	237	217	102	238
98	217	0	0	181	181	0	0	217	98	217	217	217	217	255	217	102	149
255	98	181	181	181	181	181	181	98	217	217	217	217	217	217	217	102	98
255	255	98	98	98	98	98	98	98	102	102	102	102	102	102	102	181	181
255	255	255	98	98	255	98	255	255	255	255	255	255	98	98	255	98	98
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255

Figure 2.2: RGB Pixel art of Nyan Cat associated to its corresponding red pixel value grid.

An image is commonly represented in the *R*ed *G*reen *B*lue (RGB) format, as shown in Fig. 2.2, which utilizes three channels to represent the visible spectrum. Each pixel is associated with three values, one for each of the red, green, and blue channels. By leveraging the additive color property, a single pixel can represent a wide range of colors. Each value for each channel is encoded using one byte, allowing for 256 different values. Consequently, a pixel can describe $256^3 = 1,677,216$ different colors.

Concerning the labels, they can be broadly categorized into two categories:

- *discrete labels*: They are labels that can only take a discrete number of values. They are used for *classification tasks* such as image recognition, image segmentation, video action recognition...

- *continuous labels*: They are labels that can take values from a continuous distribution. They are used for *regression tasks* such as object detection, action localization and detection...

2.1.3 Machine Learning frameworks

ML algorithms can be classified into two main frameworks: *supervised learning* and *unsupervised learning* (UL). In SL, the dataset is labeled, and the algorithm is trained to associate each input with its corresponding label or target value. On the other hand, UL involves learning from unlabeled data, where the algorithm aims to extract structures or patterns from the dataset, such as probability distributions or clusters.

In a supervised setting, the goal of ML is to find a function $f_\theta : \mathbb{X} \rightarrow \mathbb{Y}$, parametrized by a set of parameters $\boldsymbol{\theta}$, that maps input data from $\mathbb{X}$ to appropriate labels in $\mathbb{Y}$.

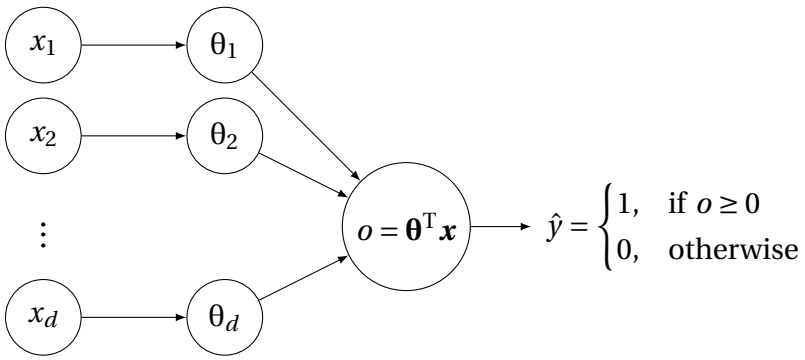


Figure 2.3: Perceptron with parameters $\boldsymbol{\theta}$ and the threshold 0 that receives $\boldsymbol{x}$ as input.

Let's consider the example of a supervised ML algorithm called the Perceptron [SB58], as illustrated in Fig. 2.3. The Perceptron is used for binary classification, which involves assigning one of two different labels (0 or 1) to each input data point. The goal of the Perceptron is to associate each example $\boldsymbol{x}_i$ with its corresponding label y_i using a function f_θ that is parameterized by a set of parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, ..., \theta_d)$, where d is the dimension of the input features $\mathbf{X}$, along with a threshold ϵ. The output $\hat{y}_i$ for a given input $\boldsymbol{x}_i$ is computed using the following operations:

$$o_i = f_\theta(\boldsymbol{x}_i) = \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{x}_i = \sum_{j=1}^{d} \theta_j x_{i,j}, \tag{2.1}$$

$$\hat{y}_i = \begin{cases} 1, & \text{if } o_i \geq \epsilon \\ 0, & \text{otherwise} \end{cases}. \tag{2.2}$$

The output $\hat{y}_i$ is then used as the predicted label by the perceptron model.

2.1.4 Risk minimization

In previous sections, we mentioned what an ML supervised algorithm is used for. Now we will formally define what they seek to learn.

Recall that supervised ML algorithms seek to find a function, also known as a *hypothesis* or *model*, $f_\theta : \mathbb{X} \rightarrow \mathbb{Y}$, parameterized by $\boldsymbol{\theta} \in \Theta$. Depending on the assumptions about the parameters $\boldsymbol{\theta}$, there can be numerous possible functions, and the set of all possible functions is called the *hypothesis space* $\mathscr{H} = \{f_\theta | \boldsymbol{\theta} \in \Theta\}$.

The goal of ML is to find the best $f_{\hat{\theta}}$ from $\mathscr{H}$. However, to define what constitutes the best model, we need to introduce the concepts of *risk* and *loss* function. The loss function $\mathscr{L}$, denoted as $\mathscr{L}\left(f_{\hat{\theta}}(\boldsymbol{x}), \boldsymbol{y}\right)$, where $(\boldsymbol{x}, \boldsymbol{y}) \in (\mathbb{X}, \mathbb{Y})$, quantifies the discrepancy between the predicted output $f_{\hat{\theta}}(\boldsymbol{x})$ and the actual target output $\boldsymbol{y}$.

The risk, denoted as $\mathscr{R}\left(f_\theta\right)$, is the expected value of the loss function over the input space $\mathbb{X}$ and the output space $\mathbb{Y}$, and it can be expressed as:

$$\mathscr{R}\left(f_\theta\right) = \mathbb{E}\left[\mathrm{L}\left(f_\theta(\boldsymbol{x}), \boldsymbol{y}\right)\right] = \int \mathscr{L}\left(f_\theta(\boldsymbol{x}), \boldsymbol{y}\right), \partial \mathrm{P}\left(\boldsymbol{x}, \boldsymbol{y}\right). \tag{2.3}$$

Since we do not have access to the true distribution $(\mathbb{X}, \mathbb{Y})$ of inputs and outputs, but only to observed data stored in datasets $(\mathbf{X}, \mathbf{Y})$ as defined in Sec. 2.1.2, the risk is estimated by the *empirical risk*, which averages the loss function over the training set as follows:

$$\mathscr{R}_{emp}\left(f_\theta\right) = \frac{1}{n_{train}} \sum_{i=1}^{n_{train}} \mathscr{L}\left(f_\theta\left(\boldsymbol{x}_i^{(train)}\right), \boldsymbol{y}_i^{(\text{train})}\right). \tag{2.4}$$

The *Empirical Risk Minimization* (ERM) principle [Vap91] states that the learning algorithm should choose $f_{\hat{\theta}}$ that minimizes the empirical risk:

$$f_{\hat{\theta}} = \underset{f_\theta \in \mathbb{H}}{\arg\min}\, \mathrm{R}_{emp}(f_\theta). \tag{2.5}$$

Therefore, the goal of ML is to solve the optimization problem defined in Equation 2.5.

In practice, we are interested in having a model that is capable of generalizing to unseen data. Thus, the ERM principle is applied to the *testing error*, or *generalization error*, which averages the loss on the test dataset. Consequently, the best model is selected based on its ability to not only fit the training data but also perform well on new data.

2.1.5 Capacity

The *capacity* of an ML algorithm refers to its ability to represent patterns or relationships in the data. For the Perceptron, its capacity is determined by the number of parameters

it has. Models with high capacity are capable of learning complex patterns in the training data. However, when the ML algorithm becomes too specialized on the training set, it may result in *overfitting*. Overfitting occurs when the model learns irrelevant features specific to the training data, leading to lower performance on unseen data which is estimated on a testing set.

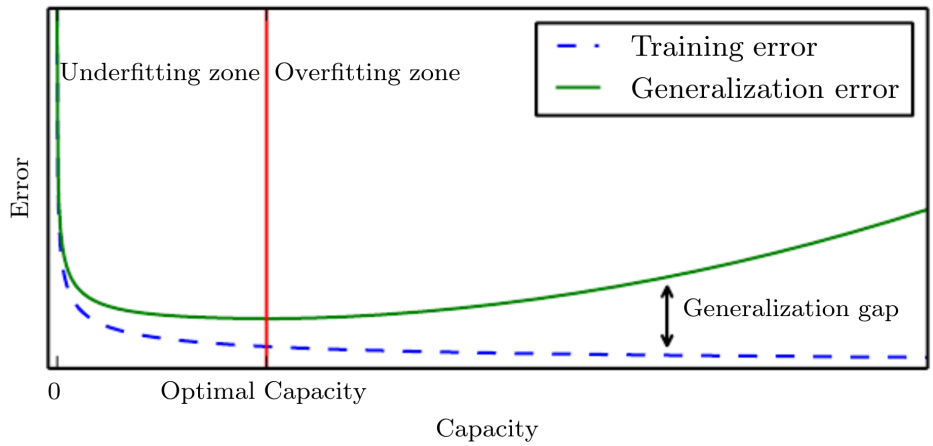


Figure 2.4: Relationship between capacity and error illustrated by [GBC16]. *Underfitting* happens when the model has low capacity so it cannot fit properly training data and has both high training and generalization error, the latter estimated on a testing set. *Overfitting* happens when the model has too much capacity that leads to learning irrelevant training features that minimize training error but increase generalization error causing a wide *generalization gap*.

On the other hand, a low capacity can lead to *underfitting*, where the algorithm is unable to learn patterns that are complex enough to solve the task effectively, both on the training set and the test set. The relationship between capacity and fitting regimes is illustrated in Fig. 2.4.

2.1.6 Regularization

Regularization is a technique used to prevent overfitting of models and improve their generalization to unseen data. It introduces a penalty term $\mathscr{P}(\theta)$, with θ the model parameters, in addition to the loss function $\mathscr{L}\left(f_\theta(\boldsymbol{x})\right)$, to encourage the Machine Learning algorithm to learn simpler and more general models from a global objective $\mathscr{J}\left(f_\theta(\boldsymbol{x}),\theta\right)$ such as:

$$\mathscr{J}\left(f_\theta(\boldsymbol{x}),\theta\right) = \mathscr{L}\left(f_\theta(\boldsymbol{x})\right) + \mathscr{P}(\theta). \tag{2.6}$$

The most common form of regularization is known as *L2 regularization*, *Ridge regression*, or *Weight Decay* (WD) [KH91]. It involves adding the L2 norm of the model parameters to the loss function such as:

$$\mathscr{J}_{wd}\left(f_\theta(\boldsymbol{x}),\theta\right) = \mathscr{L}\left(f_\theta(\boldsymbol{x})\right) + \lambda|\theta|_2^2, \tag{2.7}$$

where λ is a parameter that controls the strength of the regularization. A higher value of λ corresponds to a stronger regularization.

2.2 Deep Learning

2.2.1 Neuron

In Sec. 2.1.3, we defined the Perceptron model. In DL, a *neuron* is based on the Perceptron model. A neuron is composed of *weights* $\mathbf{W} = (\mathbf{W}_1, \ldots, \mathbf{W}_d) \in \mathbb{R}^d$, as well as a *bias* parameter $b \in \mathbb{R}$. Sometimes, the bias parameter is denoted as $\mathbf{W}_0$ and combined with the other parameters in $\mathbf{W}' = (\mathbf{W}_0, \ldots, \mathbf{W}_d)$.

The neuron performs two mathematical operations. First, it multiplies its weights with the input and adds the bias. Then, it applies an *activation function* to produce an output. In the case of the Perceptron, as defined in Sec. 2.1.3, the activation function is a characteristic function that returns 1 for all values greater than 0.

2.2.2 Multi-Layer Perceptrons

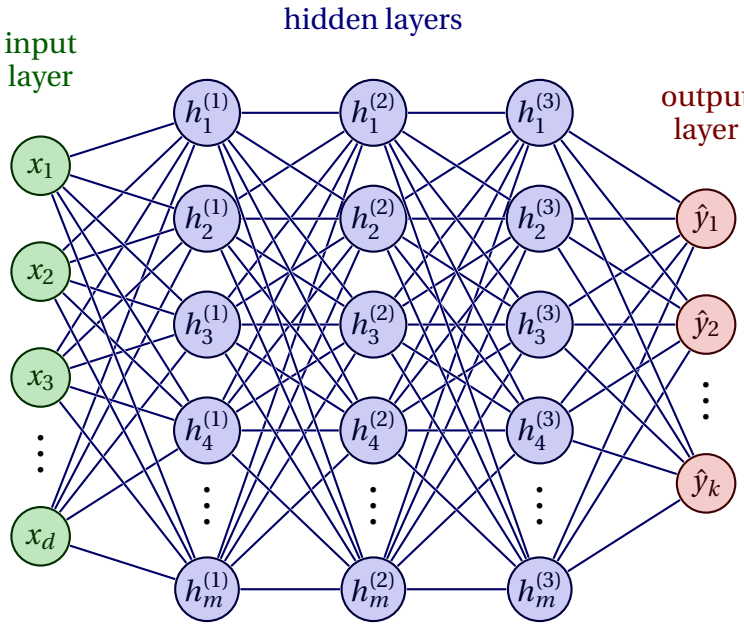


Figure 2.5: Multi-Layer Perceptron composed of an input layer, three hidden layers, and an output layer. It contains fully connected layers, therefore, neurons of each layer are connected to all neurons of the previous layer and all the neurons of the next layer.

Feedforward NNs are composed of stacked layers of neurons. The most basic type of NN is called a *Multi-Layer Perceptron* (MLP) [Mur91], which consists of *Fully Connected* (FC) layers as illustrated in Fig. 2.5. The neural network is structured into three types of layers: the *input layer*, the *hidden layers*, and the *output layer*.

The input layer is the initial input data that is fed to the hidden layers. The hidden layers, as the name suggests, are intermediate layers that perform computations on their input representation. Each hidden layer incorporates weights, bias, and an *activation function* for each of their neurons. Activation functions introduce non-linearity into the network, allowing it to learn and model complex relationships between inputs and outputs. Finally, the output layer produces the final representation or prediction based on its weights and activation function.

For hidden layers, the main used activation functions are *Rectified Linear Unit* (ReLU) and its variants (Leaky ReLU, PReLU, GeLU, ...) but others have been proposed such as Tanh, and Threshold, ... For the output layer, the activation function depends on the task at hand, for binary or multi-label classification the Sigmoid, for multi-class classification the Softmax ...

Each layer of a NN can be viewed as a function that is parameterized by the weights or parameters contained within its neurons. The combination of these functions across all layers forms the complete NN, allowing it to transform input data into desired output predictions.

More formally if each layer $l^{(i)}$ is parametrized by its weights $\mathbf{W}^{(i)}$ and bias $\mathbf{b}^{(i)}$, and activation function $a^{(i)}$ the *forward pass* of a neural network of n layers denoted $f_\theta(\mathbf{x})$ is the following:

$$h^{(0)} = \mathbf{x}, \tag{2.8}$$

$$h^{(i)} = a^{(i)}(\mathbf{W}^{(i)} h^{(i-1)} + \mathbf{b}^{(i)}), i = 1, \ldots, n, \tag{2.9}$$

$$f_\theta(\boldsymbol{x}) = \hat{y} = h^{(n)}, \tag{2.10}$$

with $\boldsymbol{\theta}$ is the set of all $\mathbf{W}^{(i)}$ and $\mathbf{b}^{(i)}$ parameters.

2.2.3 Gradient Descent

During the training process, *gradient descent* [C+47] is commonly used to optimize the neural network's parameters. This optimization algorithm relies on *backpropagation* [RHW86] and the chain rule from calculus to compute the gradients of the loss function concerning the parameters using the entire dataset. By iteratively updating the parameters in the direction of the steepest descent, the NN gradually improves its performance and minimizes the loss. The parameters are updated by subtracting a fraction of the gradients from the current parameter values, multiplied by a *learning rate*. The learning rate determines the step size taken in the parameter update. This process is repeated iteratively until convergence or a predefined stopping criterion is met. It's important to note that NNs being non-linear functions can have multiple local minima, and convergence to the global minimum is not guaranteed.

In practice, using the entire training dataset for each parameter update is often computationally expensive and not feasible due to limited computational resources. Therefore, two different variants of gradient descent have been developed: online gradient descent and batch gradient descent.

Online gradient descent [RM51], computes the gradient and updates the parameters for each training example, one example at a time. This approach introduces more randomness into the parameter updates but can converge faster and is suitable for large datasets.

Algorithm 1 Batch gradient descent

1: Randomly initialize $\mathbf{W}_0^{(i)}$ and $b_0^{(i)}$ **for** $i \in \{1,\dots,n\}$
2: **for** $t \in \{1,\dots,n_{iter}\}$ **do**
3: Select a batch $\mathscr{B}$ from $\left(\mathbf{X}^{(train)},\mathbf{Y}^{(train)}\right)$ shuffled
4: Update $\mathbf{W}_t^{(i)}$ using the formula $\mathbf{W}_t^{(i)} = \mathbf{W}_{t-1}^{(i)} - \frac{\lambda}{\#\mathscr{B}} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathscr{B}} \frac{\partial \mathscr{L}(f_\theta(\boldsymbol{x}),\boldsymbol{y})}{\partial \mathbf{W}_{t-1}^{(i)}}$ **for** $i \in \{1,\dots,n\}$
5: Update $b_t^{(i)}$ using the formula $b_t^{(i)} = b_{t-1}^{(i)} - \frac{\lambda}{\#\mathscr{B}} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathscr{B}} \frac{\partial \mathscr{L}(f_\theta(\boldsymbol{x}),\boldsymbol{y})}{\partial b_{t-1}^{(i)}}$ **for** $i \in \{1,\dots,n\}$
6: **end for**

Batch gradient descent, also known as *vanilla gradient descent*, uses a batch of several samples from the training dataset in each iteration to compute the gradient and update the parameters. Its pseudo-code is in Algorithm 1 with the standard *stochastic gradient descent optimizer* that shuffles the dataset before training. It provides a more accurate estimation of the gradients compared to online gradient descent but can be computationally expensive, especially for large batches. Several optimizers have been proposed to improve convergence such as Adam [KB14] or LAMB [YLR+19] and LARS [YGG17a] for large batch size.

Batch gradient descent suffers from instabilities, especially in deep architectures. Updates from gradient computation can be either very small causing *vanishing gradients* which prevent a good update or very large causing *exploiding gradients* which prevent convergence to a local optima.

2.2.4 Convolutional Networks

Dense MLPs have certain limitations that hinder their performance in certain scenarios. One significant drawback is its inefficiency when dealing with high-dimensional input. As the input dimensionality increases, the computational requirements of the MLP grow substantially, making it computationally expensive and challenging to scale. This limitation can be particularly problematic in applications where large amounts of data with high-dimensional features need to be processed promptly. This is particularly relevant in this work in which we dealt with images and videos that contain lots of pixels.

Another limitation of MLP is its inability to effectively leverage the inherent structure of certain types of data, such as grids in images. Traditional MLPs treat input data as flat

vectors, disregarding the spatial relationships and patterns that may exist within the data. For example, in image data, neighboring pixels often hold valuable information that contributes to understanding the content and context of the image. However, MLPs fail to exploit this structural information, resulting in suboptimal performance for image-related tasks.

Convolutional Neural Networks (CNNs) are neural networks that use convolution in place of general matrix multiplication in at least one of their layers [GBC16]. CNNs deal with data arranged according to a grid by applying convolution on input data using different *filters* or *kernels* and pass the result to subsequent data. The convolution is defined for a two-dimensional input **I** and a two dimensional kernel **K** as:

$$(\mathbf{K} * \mathbf{I})_{i,j} = \sum_m \sum_n \mathrm{I}_{m,n} \mathrm{K}_{i-m,j-n}. \tag{2.11}$$

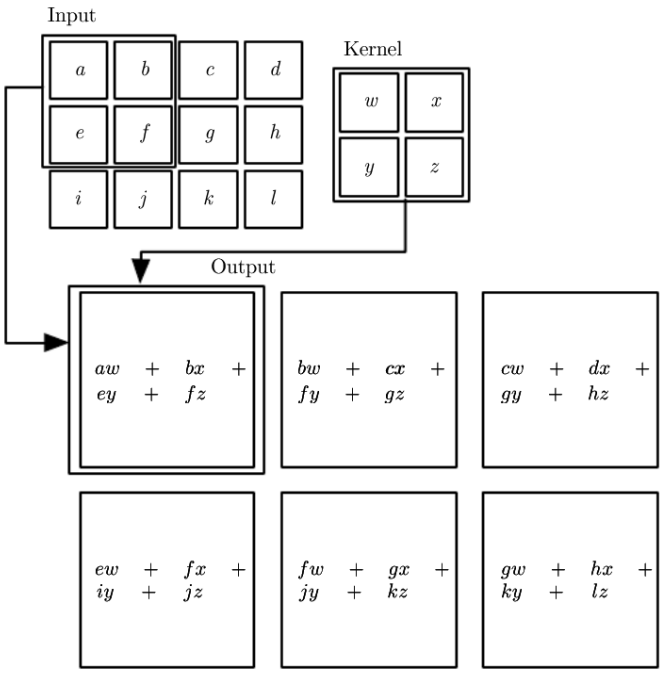


Figure 2.6: Cross-correlations of a 3 × 4 input with a 2 × 2 kernel. Figure from [GBC16].

It holds the following properties: commutativity, distributivity, and associativity. In practice CNNs do not use convolutions but *cross-correlations*, illustrated in Fig. 2.6, and defined as :

$$(\mathbf{K} * \mathbf{I})_{i,j} = \sum_m \sum_n \mathrm{I}_{i+m,j+n} \mathrm{K}_{m,n}. \tag{2.12}$$

Cross-correlations lose the commutativity property that is not required to train a NN whilst making the operation more efficient. Later we refer to cross-correlations as convolutions as it is standard in the field of this work.

Over MLPs, CNNs have the following advantages:

- *Sparse connectivity*: The kernels are smaller than the input which requires fewer parameters and connections.
- *Parameter sharing*: The kernels perform a sliding window over the input meaning the same parameters are used on different locations.
- *Equivariance to translation*: By definition, convolutions are equivariant to translation meaning that applying a translation to the input results in an output after convolution with the same translation.

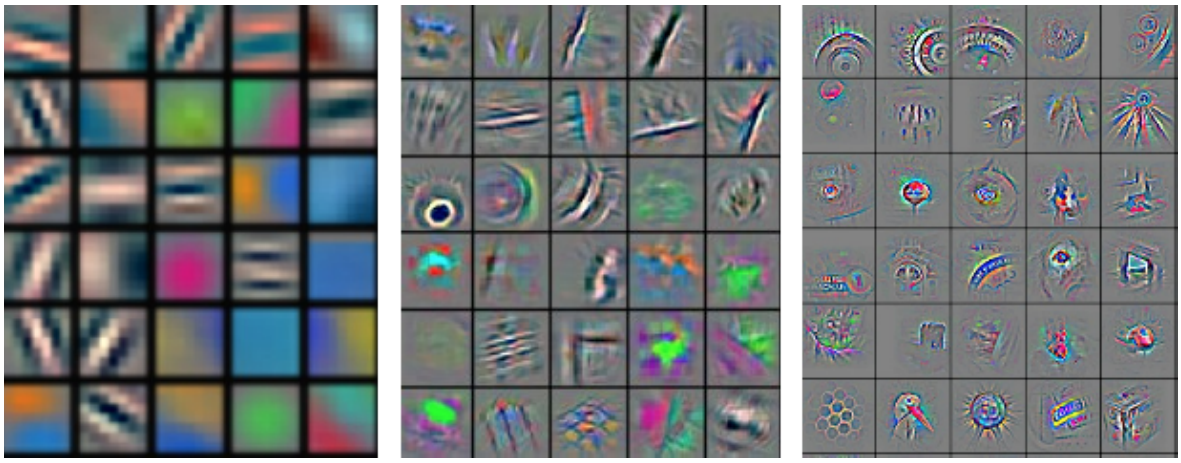


Figure 2.7: Filters learned by successive convolutional layers by AlexNet [KSH12].

Deep CNNs contain stacks of convolutional layers followed by at least one final fully connected layer to make the predictions of the network. Its filters learn more and more complex patterns as the depth grows from detecting edges to body parts, and flowers as illustrated in Fig. 2.7. Convolutions with stride which are generally followed by pooling have an output at lower resolution than the input per construction which permits the final MLP to use fewer features.

For the rest of this section, we will briefly introduce the main architectures that paved the way for current CNN architectures.

LeNet

LeNet [LBBH98], illustrated in Fig. 2.8, was designed to deal with handwritten bank checks and has paved the way to the basics of Deep Learning. It is composed of several convolutional layers that are followed by an averaged pooling and a non-linearity to extract features used by MLP layers to perform digit classification.

ALexNet

AlexNet [KSH12], illustrated in Fig. 2.9, is one of the first *Deep Neural Network* (DNN) and the origin of the Deep Learning era as it won the ImageNet [DDS+09] 2012 compe-

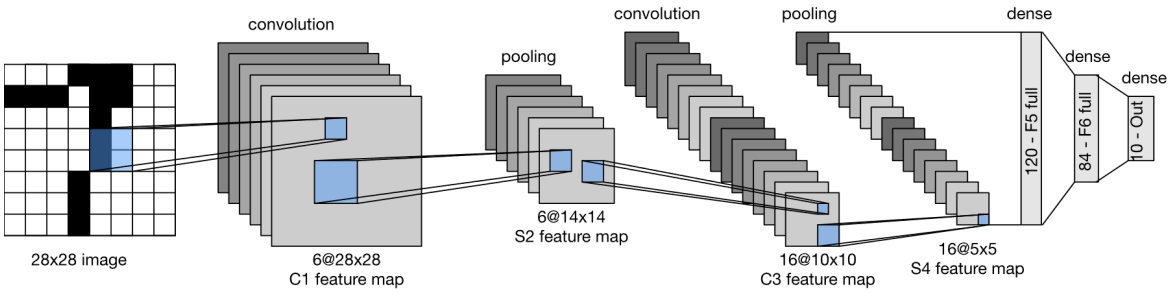


Figure 2.8: LeNet architecture [LBBH98].

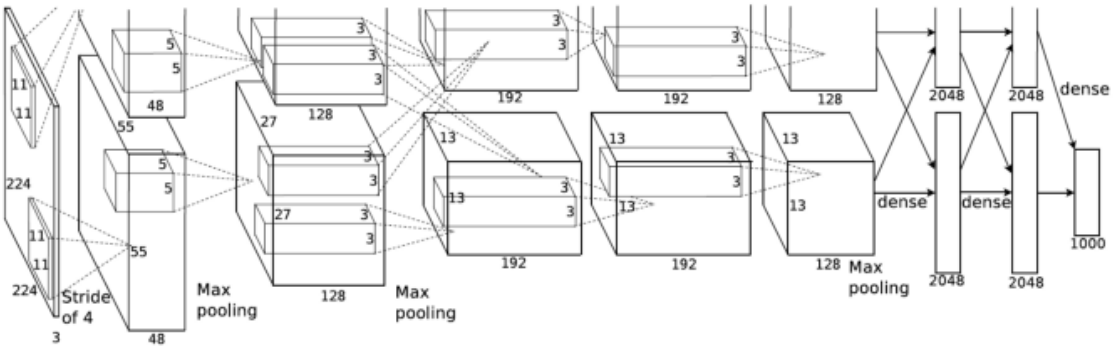


Figure 2.9: AlexNet [KSH12] architecture.

tition with a large margin in comparison with hand-crafted features coupled with ML algorithms. It extended LeNet by using larger images and wider convolutions as well as dropout [SHK+14] to regularize the network.

Blocks

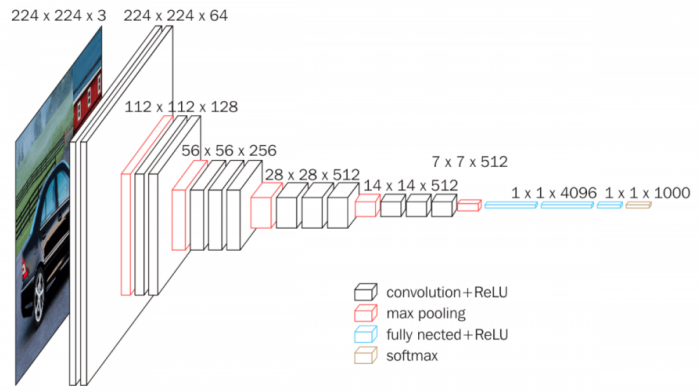


Figure 2.10: VGG [SZ15] architecture. Each block maintains a resolution.

Based on this success, subsequent CNNs were built to improve results by stacking several Convolutional Blocks containing several convolutional layers that maintain the resolution within each block such as VGG [SZ15] illustrated in Fig. 2.10. This allows to perform more subsequent convolutions as the resolution does not decrease too fast.

ResNet

As CNNs grew deeper, some training instabilities appeared such as vanishing gradients and exploding gradients. To avoid this phenomenon, several approaches were proposed such as having several classification layers at different locations in the network [SLJ+15].

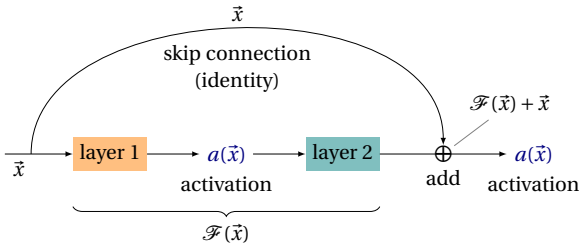


Figure 2.11: Residual connection [HZRS16].

The approach that has been most widely used, even in modern architectures, is implementing *residual connection*, or *shortcut connection* that concatenate the output of a block with its input as illustrated in Fig. 2.11. In particular, it is at the core of the widely used ResNet architecture [HZRS16].

Other image architectures

Newer architectures since ResNet have been proposed such as EfficientNet [TL19], DenseNet [HLVDMW17] that seek to increase the depth or the connections across the backbone. This comes to the cost of increased time for training and inference and dedicated approaches for edge devices have been proposed such as MobileNet [HZC+17].

Video architectures

CNN video architectures have also been developed to deal with sequences of images. Because of the temporal dimension, the convolutions have been extended to 3D convolutions such as ResNet3D [HKS18] and X3D [Fei20]. However such convolutions have a larger computational cost and some approaches developed decoupled spatial and temporal convolutions such as P3D [QYM17], I3D[CZ17], S3D [XSH+18], ResNet(2+1)D [TWT+18].

However, such CNN architectures tend to overfit the spatial dimensions and less on the motion of the videos. To circumvent this issue, two stream architectures [SZ14, FPZ16] have been developed to fuse predictions from two neural networks. Often, one branch is learned on RGB frames and the other one on a motion modality such as Optical Flow to enforce predictions taking into account movement. To avoid dealing with multiple modalities and handling multiple neural networks, SlowFast [FFMH19], illustrated in Fig. 2.12, proposed a two-stream end-to-end approach that solely relies on RGB with one

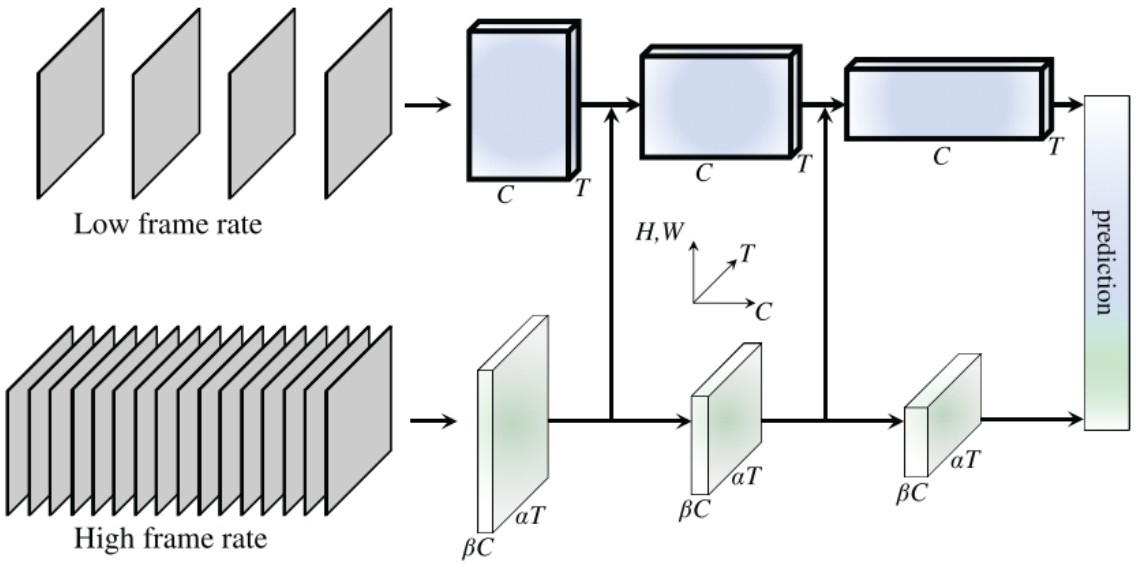


Figure 2.12: SlowFast architecture, from [FFMH19]. The slow path receives a low frame rate video and has wide convolutions while the fast path receives the same video at a fast frame rate with shallow convolutions. The predictions of both paths are fused at their output and the fast path distills motion information in the slow path at different depths.

slow path branch that receives few frames but has wide convolutions to focus on the spatial aspect and a fast branch that receives lots of frames but with shallow convolutions to focus on the temporal aspect. The fast branch has connections with the slow branch at different depths to distill motion information.

2.2.5 Transformers

Transformers [VSP+17] have first been introduced for NLP tasks and have been extended to *Vision transformers* (VTs) [DBK+21] as a promising alternative to CNNs. CNNs have struggled to scale well. Indeed, as the size of the input images increases or the deeper the architecture is, CNNs have difficulties increasing their performance. On the other hand, Vision transformers have shown remarkable scalability, allowing them to handle larger images and huge networks.

The main feature of VTs is the use of the *attention mechanism* [VSP+17]. Attention mechanisms enable the model to capture long-term dependencies within the input data, which is essential for understanding context and relationships between different elements. Unlike CNNs, which rely on local receptive fields and convolutional operations, attention mechanisms allow VTs to attend to the entire image, giving them a global context and facilitating the integration of information across different parts of the image.

Formally, the Scaled Dot-Product Attention Mechanism, illustrated in Fig. 2.13, is defined with three matrices a query $\mathbf{Q}$, a key $\mathbf{K}$ and a value $\mathbf{V}$ as well as three matrices of weights $\mathbf{W}_q$, $\mathbf{W}_k$ and $\mathbf{W}_v$:

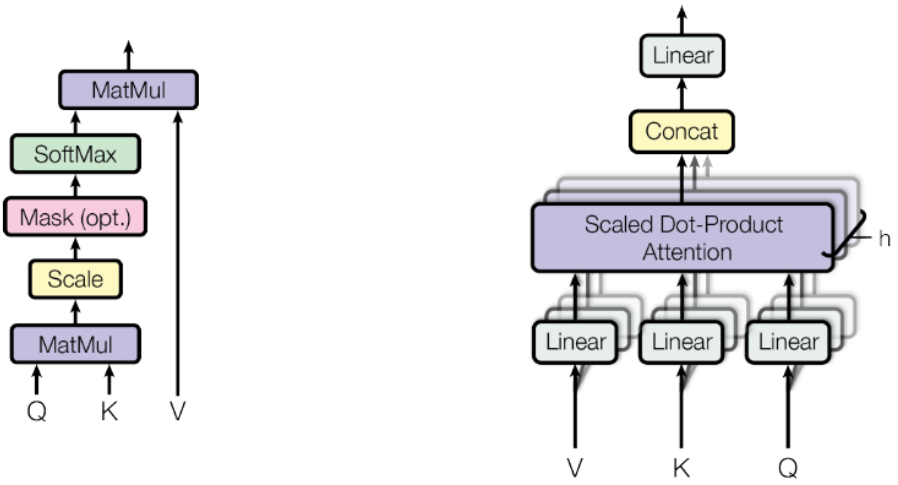


Figure 2.13: From [VSP+17]. (left) Attention mechanism implemented via a Scaled Dot-Product Attention. (right) Multi-head attention consists of several attention layers running in parallel.

$$\text{Attention}_{\mathbf{W}_q,\mathbf{W}_k,\mathbf{W}_v}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{(\mathbf{W}_q\cdot\mathbf{Q})\cdot(\mathbf{W}_k\cdot\mathbf{K})^{\mathrm{T}}}{\sqrt{d}}\right)\cdot(\mathbf{W}_v\cdot\mathbf{V}), \tag{2.13}$$

with d the dimension of the keys $\mathbf{K}$. In case of self-attention $\mathbf{Q}=\mathbf{K}=\mathbf{V}$.

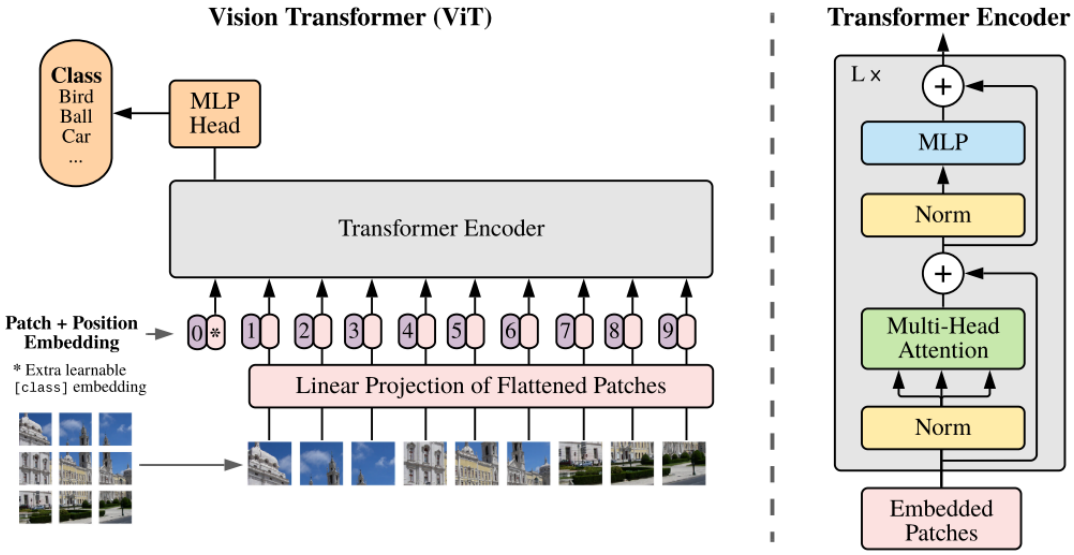


Figure 2.14: From [DBK+21]. The Vision Transformer (ViT) architecture. It is composed of a tokenizer that handles patches of images to form tokens. Then the tokens are passed to a Transformer encoder composed of multiple self-attention layers to output the results.

A Vision Transformer, illustrated in Fig. 2.14, is composed of two main parts:

- a *Tokenizer*: It is generally implemented as a single Convolutional Layer that divides the input image into patches of data to compute input *tokens*. For videos, the patches are spatio-temporal parallelepiped that spans the 2D patches across multiple frames which is often set to 2. These tokens serve as the basic units of com-

putation for the subsequent layers in the model that output as many tokens as they take as input. To maintain positional information lost during the tokenization, a *positional embedding* is added to the tokens.

- an *Encoder*: It is the core of VTs and comprises several stacks of self-attention layers. Each self-attention layer contains multiple attention heads, as illustrated in Fig. 2.13, which learn different features from its input tokens. By attending to various regions, the attention heads can capture different aspects of the image and extract meaningful representations. Additionally, the self-attention layers incorporate residual connections and layer normalization, allowing the model to retain important features and mitigate the vanishing gradient problem.

Self-attention can be computationally expensive, especially as the input resolution size increases and alternative attention mechanisms have been proposed such as Shifted Window Attention [LLC+21]. While VTs have shown impressive performance in computer vision tasks, they require lots of training data which makes them prone to benefits from pretraining [DBK+21].

Depending on the task and the transformer architecture, a *class token* is added during the tokenization process. The goal of this class token is to contain general information on the input data to perform from the output class token task-related supervision such as classification.

2.3 Representation Learning

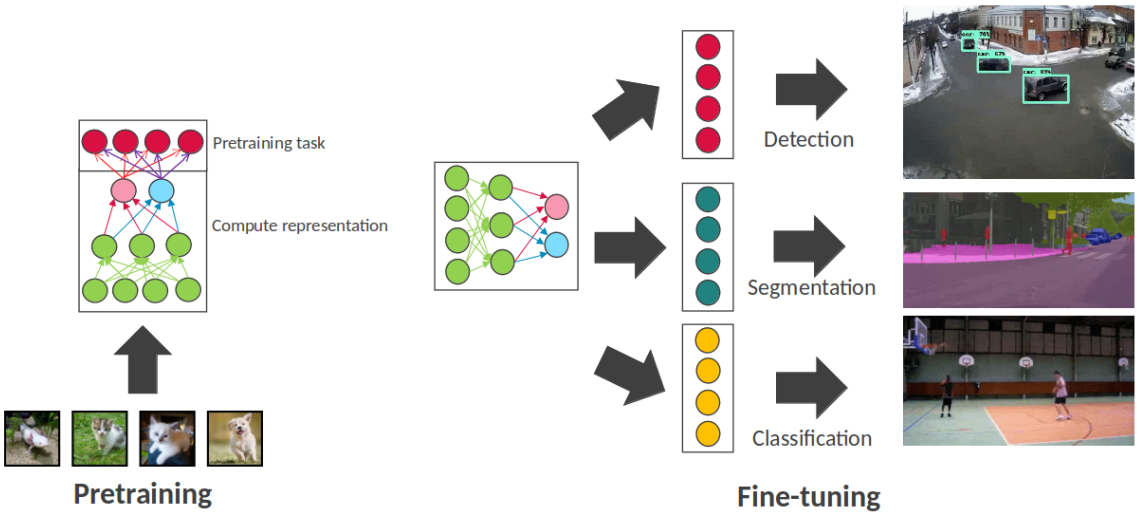


Figure 2.15: (left) Representation learning of a NN separated in two-parts, the *backbone* that learns representation and the *predictor* that solve the pretraining task. (right) Finetuning of the backbone or/and training of an NN to specialize in a task.

2.3.1 Definition

Representation learning [BCV13] consists in learning *features*, or a *latent space* of the data to easily extract information for various tasks. A good representation captures the distribution of explanatory factors of the observed input and should be enough to pass to a supervised predictor for downstream tasks as illustrated in Fig. 2.15.

Therefore, Deep Representation Learning seeks to learn a *backbone*, or an *encoder* that projects its input data to a latent space that meets the criteria of a good representation. Training such backbone is commonly referred to as *pretraining* and can be performed by SL or SSL. For specialization, the backbone can be *fine-tuned* to perform *transfer learning* meaning its weights are updated via the supervised task or *fixed*, and a new NN is learned on its output representation.

Pretraining a Neural Network has the following advantages:

- *Data efficiency*: Pretraining can be performed on a large dataset to learn general representations which can be fine-tuned on smaller, task-specific datasets.
- *Faster convergence*: Pretraining initializes the network with weights that contain general concepts and permits faster convergence to optimize the network for a specific task.
- *Generalization*: Initializing NN weights from a *source domain*, or source dataset, offers a better starting point to the *target domain*, even if it is "far" from the source. Also by pretraining on a large source dataset, general concepts are learned that are beneficial for multiple domains.
- *Feature extraction*: For limiting the cost of fine-tuning to a specific task, using the pretrained network to extract features from the input data and then train on these features instead of the raw images or videos mitigates the cost of latter training.
- *Cost-efficient*: Pretraining once for multiple specializations drastically reduces the time, computational, and environmental resources required to achieve good results in DL.

2.3.2 Supervised Pretraining

For Supervised pretraining, we suppose we have access to a (large) labeled source dataset $\mathscr{D}^{(\mathrm{SL})} = (\mathbf{X}^{(\mathrm{SL})}, \mathbf{Y}^{(\mathrm{SL})})$ and a NN composed of an encoder $f^{(\mathrm{SL})}$ and a predictor $g^{(\mathrm{SL})}$. The goal of supervised pretraining is to learn the best representation for its input data from its encoder via a supervised task based on the labels that the predictor estimates. For the latter specialization, the predictor is ditched.

Most commonly, supervised representation learning is accomplished via a classification task on a large dataset such as ImageNet [KH09] for images and Kinetics [KCS+17] for videos. Therefore the objective functions for a batch $\mathscr{B} \in \mathscr{D}^{(\mathrm{SL})}$ is:

$$\mathscr{L}^{(\mathrm{SL})} = -\frac{1}{\#\mathscr{B}} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathscr{B}} \sum_{c=1}^{\mathrm{C}} \mathrm{y}_c \log\left(g^{(\mathrm{SL})}\left(f^{(\mathrm{SL})}(\boldsymbol{x})\right)_c\right), \tag{2.14}$$

with C the number of classes and $\boldsymbol{y}$ a one hot label over the c classes, meaning $y_c = \mathbb{1}(\text{class of } \boldsymbol{x} \text{ is } c)$. It is a *softmax* loss function that is commonly used for classification.

Having a large enough source dataset is essential to learning a general representation, especially for DNN. This has several drawbacks that prevent such learning paradigm from scaling well:

- *Cost of labelisation*: Labeling data necessitates the efforts of several annotators which not only takes time but requires computational resources and money. Depending on the type of data, hiring experts such as in medical data can be necessary which engenders supplementary cost.
- *Label bias*: Annotators as all human beings are prone to biases. This can be reflected in the data which has a detrimental effect on model performance and/or fairness. Having multiple annotators can help reduce some form of bias but increase costs. Moreover, *data bias* is present and inherent to the collection of data.
- *Error of annotation*: Annotators are subject to making errors that will affect the dataset quality and later performance. Having multiple annotators can help reduce some form of bias but increases costs.
- *Difficulty in scaling*: As the rise of DL approaches on various tasks increase and models become more and more complex, scaling the datasets becomes a necessity that is hard to achieve for labeled ones.
- *Limited availability*: Not only do some datasets require some expert knowledge that is hard to obtain, but some kinds of data can be scarce. Also, because of the cost of obtaining a labeled dataset, several actors prefer keeping them private or accessible with financial compensation.

2.3.3 Self-Supervised Pretraining

SL dependences on labels have multiple issues as explained in precedent Sec. 2.3.2. However, standard Unsupervised Learning algorithms that involve density estimation, dimensionally reduction such as PCA [Pea01] or t-SNE [VdMH08] and clustering like K-means [M+67] are meant to explore and extract the intrinsic structure of the data but do not involve learning a representation for downstream supervised tasks.

Self-supervised learning is a UL approach that seeks to learn a good representation of data from an unlabeled (large) source of data by creating labels from the data and forming the dataset $\mathscr{D}^{(\text{SSL})} = (\mathbf{X}^{(\text{SSL})}, \mathbf{Y}^{(\text{SSL})})$. The model is trained to solve a *pretext task* by predicting certain parts of the data itself from which the labels have been designed. Hence the name Self-supervised learning, as the network is *supervised by the data* itself. As for SL pretraining, the NN is composed of an encoder $f^{(\text{SSL})}$ and a predictor $g^{(\text{SSL})}$.

The advantages of SSL are multiple in comparison to SL:

- *No labelisation*: Not needing labelisation completely removes the cost of labelisation and the error of annotations mentioned for SL. However, it does not completely remove the underlying data bias caused by the discretization of the domain via the creation of a dataset that cannot completely represent the domain and by the choice of the pretext task.

- *Immediat scaling*: Virtually there is no limitation to the scaling of unlabeled datasets. In practice, there is a need for enough storage but also availability for some types of data such as satellite images or submarine videos, and some domains are easier to obtain than others.

- *Broad availability*: A lot of data are available on the internet and continue growing. But also more and more devices are equipped with cameras such as phones or houses, ...

Image and video SSL have shown tremendous success and narrowed the gap with weak SL baselines [RLA+21, CXH21, CTM+21], however, they have yet to surpass Supervised Learning when compared with fairer baselines [SKAL23]. Because of the promising results achieved in NLP and the continuous breakthrough also observed in SSL for CV, this area of research is a hot topic that has gained a lot of attention in recent years.

Over the years different pretext tasks have been proposed that can be divided into several categories. In this thesis, we chose to separate them into 4 main categories:

- *Geometric and Intensity pretext tasks*

- *Contrastive Learning*

- *Clustering*

- *Masked Modeling*

Below, we describe quickly the different categories which will be more thoroughly discussed in Chapter 3.

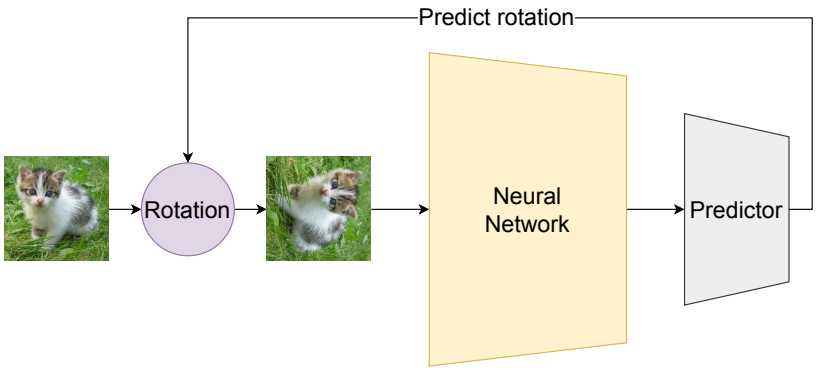


Figure 2.16: Rotation prediction pretext task. Input is rotated and the NN followed by a predictor seeks to predict the rotation applied.

Pretext tasks

Geometric and Intensity pretext tasks perform a transformation of the input and the objective function is designed to predict what transformation has been applied as illustrated in Fig. 2.16. Over the years, multiple transformations have been designed for images and videos such as:

- Color Inpainting [ZIE16]
- Predicting shuffling of image patches [NF16] or video frame order [MZH16]
- Predicting Rotation [GSK18]

These methods have been successful in learning a representation as the NNs, to fulfill the objective function, have to learn the inherent structure and conceptual information among the input. For example, to successfully predict the order of frames, the NN needs to learn the arrow of time. However, since these pretext tasks are hand-crafted, they fail to learn a general representation for two main reasons. First, handcrafted transformations have a discrete support that does not allow for generating a scalable number of transformations. Second, as the NNs are learned to become invariant to one kind of transformation, they cannot generalize to downstream tasks requiring to not be invariant to it and cannot grasp concepts not involved in solving the pretext task used to pretraining.

Clustering

Clustering is an unsupervised task that groups similar objects into *clusters*. In the context of SSL, the similarity is computed on the latent representation of the dataset from the NN to be pretrained generally via a norm function [CBJD18, ARV20]. After the computation of the similarities, a clustering algorithm is applied to the similarity matrix such as K-means to create *pseudo-labels*. The NN followed by a predictor is then trained to predict these pseudo-labels as illustrated in Fig. 2.17.

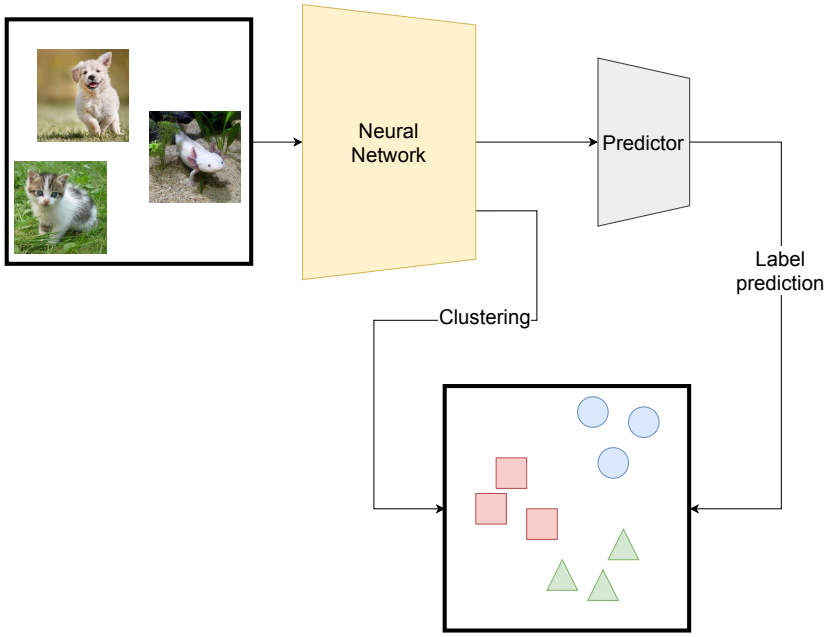


Figure 2.17: Clustering based on a NN representation from a dataset or a batch of data. The SSL task is to predict the pseudo-labels. The clustering can be performed once or multiple times throughout training.

Throughout the training, as the NN learns a better representation, the clustering is performed several times to update pseudo-labels with expected fewer errors. This simple process provides a good data representation in best cases, however, it suffers from several issues. First, passing the clustering algorithm can be quite costly to perform and even intractable for large-scale datasets. Second, the errors on pseudo-labels can harm the representation and not be recovered during training as the NN is trained to predict wrong information. Finally, clustering algorithms generally rely on hyper-parameters that can be difficult to tune and highly depend on the data distribution.

Contrastive Learning

Contrastive Learning [GH10, vdOLV18] aim to *align* latent representations from the same based input data, or *instance* while implicitly or explicitly pushing representations of other instances. Whilst several ways of doing that emerged, the most common one is based on *instance discrimination* and a siamese architecture [CKNH20] as illustrated in Fig. 2.18. In this family of approaches, an instance is transformed twice by one or two distributions of *data augmentations* to form *positive views* of the instance. Positive pairs are to be contrasted with *negatives* which are representations of other instances. To do so, two branches are designed. The first branch contains the encoder and the predictor. The second branch contains an encoder that can be the same as the first branch or not. The contrastive objective functions seek to align the feature representations of the two branches.

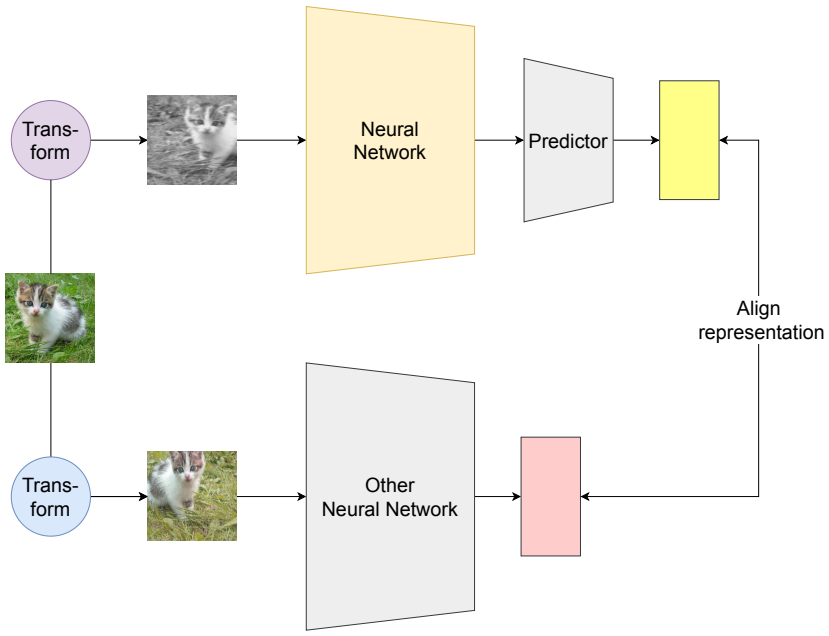


Figure 2.18: Instance discrimination CL. An image is augmented twice and passes through a Siamese NN architecture. The first branch contains the NN to pretrain and a predictor that seeks to predict the representation output of the NN in the second branch. The latter can have the same architecture and weights as the first branch or different.

Two families of contrastive learning emerged which are objective functions taking into account negatives explicitly [CKNH20, HFW+20] or without negatives [GSA+20, CMM+20, CTM+21]. The two families were successful in learning a good general data representation. However, they face different challenges. First, the choice of the data augmentation distributions needs to be carefully designed to maintain enough mutual information between the two views but not too much to make the model learn relevant features [TSP+20]. Second, contrastive with negatives have to sample a lot of negatives to perform best [CKNH20, HFW+20] and among these negatives, there should be *hard* ones which are difficult to distinguish from positives [CFSM20]. Also, some hard negatives can be seen as *False Negatives* (FN) that need to be dealt with to avoid damaging the representation. For contrastive learning without negatives, the main challenge is to design an approach that does not collapse to a trivial solution that prevents the network from learning a relevant data representation [GSA+20, CTM+21]. Finally, CL approaches tend to successfully learn a global representation of its input but negligate local information.

Masked Modeling

Masked Modeling (MM) consists of corrupting the input by masking some part of it and predicting the missing pieces as illustrated in Fig. 2.19. It is inspired by the success of Masked Language Modeling for pretraining NLP transformers [RNS+18, DCLT19]. The idea is that it forces the model to learn the context and relationship between different parts of an image to make an insightful representation. It has similarities with some geometric and intensity pretext tasks that also corrupt the input such as inpainting [ZIE16].

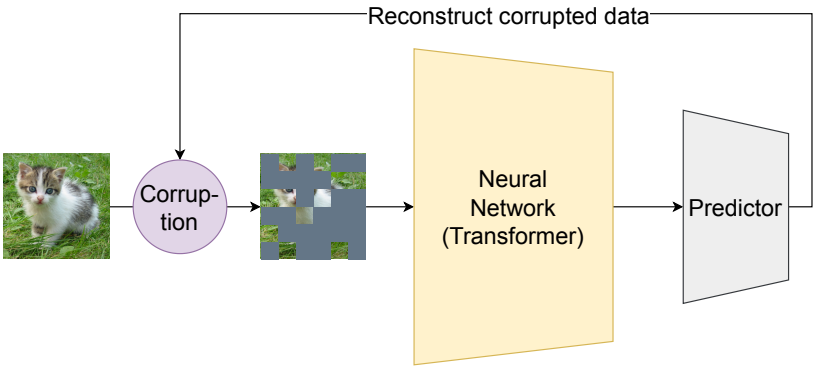


Figure 2.19: Input data is corrupted by masking part of it. The SSL task is to reconstruct the missing parts.

We made the distinction as masked modeling approaches strongly rely on the transformer architecture that makes possible the tokenization followed by the masking process. Also, we make the distinction based on its success and wide use since this family emerged whereas the aforementioned pretext tasks perform poorly in comparison with MM.

As for Language, MM approaches successfully learn a good data representation for Vision-related tasks but face several challenges. First of all, generally, they suppose a lot of redundancies [HCX+22, FFLH22] are present in the input image and video, meaning that several image or video patches contain highly common conceptual information. This is the case for object-centric datasets but this is not true for datasets with sparse visual information such as football matches. Also, because predicting patches can be performed relatively easily with close patches, NN trained via Masked Modeling tends to learn local features but is less efficient in learning more global information.

2.4 Downstream Tasks

In this section, we will discuss several downstream tasks for images and videos that we used as evaluation in our work. Indeed, the goal of Representation Learning is to provide an initialization to a network to later be fine-tuned to a specific task.

2.4.1 Images

Classification

Image classification is a fundamental task in computer vision aimed at categorizing an input image into one or multiple predefined *categories* or *classes*, capturing the overall information of the entire image and emphasizing global characteristics rather than local details. Therefore, the task of image classification encompasses various forms, including binary, multi-class, or multi-label classifications, with corresponding loss functions such as binary cross-entropy (BCE) and categorical cross-entropy (CE) as described in

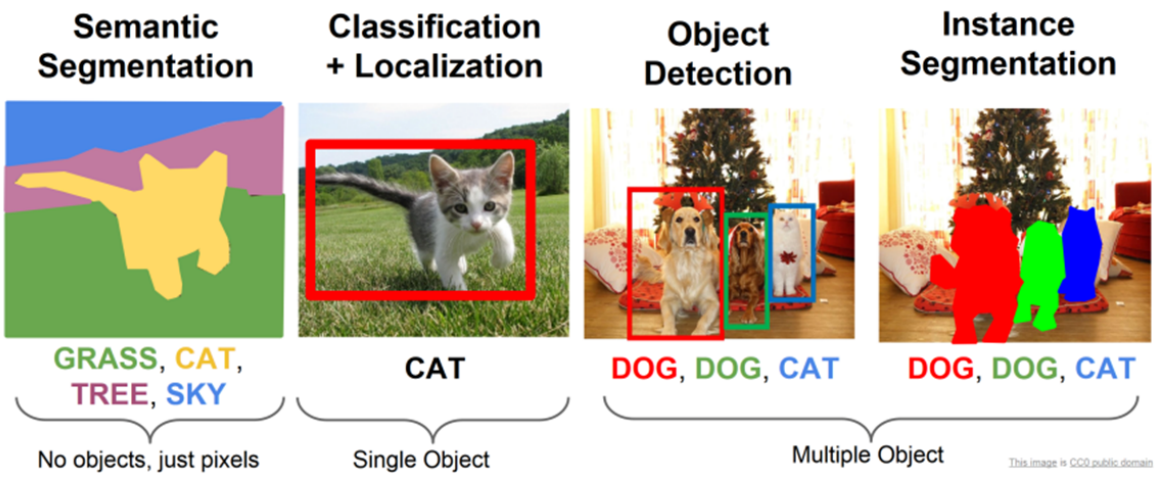


Figure 2.20: Illustration of classification, OD, semantic and instance segmentations. From [RV19].

Eq. (2.14) commonly used. As in standard machine learning, addressing unbalanced data distributions necessitates employing techniques like data rebalancing or class weighting to ensure quality prediction for all classes.

Some applications such as medical data are hard to collect which drastically reduces the number of training samples available. This makes the image classification task more challenging and some special training domains have been developed such as *zero-shot* and *few-shot* learning to deal with these cases. The former involves training on some classes and evaluation on never-seen classes and the latter sees a limited number of training samples.

Object detection

Object detection (OD) is a computer vision task in deep learning that involves identifying and localizing multiple objects of interest within an image as illustrated in Fig. 2.20. Unlike image classification, where the goal is to assign a single label to the entire image, object detection aims to draw *bounding boxes* around individual objects and predict their corresponding class labels. Therefore the object detections's prediction task is composed of two objectives: *classification* and *regression* of the boxes.

Object detection approaches can be broadly separated into two categories:

- *Two-stages approaches* [RHGS15] first use an algorithm or a *Region Proposal Network* (RPN) that proposes a set of candidate regions that are likely to contain an object. It is followed by a second stage involving a NN that refines the regions and makes the actual classification. Whilst these approaches obtain the best performances, they come at a cost of inference time that might not be tolerable.

- *One-stage approaches* [RDGF16] directly predict the bounding boxes and class labels in a single step, without the need for a separate region proposal stage. They

generate a dense set of bounding box predictions across the image and classify them into different object classes. These methods are designed for simplicity, efficiency, and real-time performance.

Segmentation

Segmentation is a computer vision task that involves partitioning an image into multiple segments or regions by assigning each pixel in the image to a specific class label. The label can be a category, an identity of an object or the background it belongs to as illustrated in Fig. 2.20.

Unlike object detection, where bounding boxes are used to localize objects, segmentation provides a more detailed and fine-grained understanding of the image's content as Each pixel is individually classified. Common classes may include various objects identities, and background regions following the three types of segmentation:

- *Semantic segmentation* considers broad categories and assigns each pixel to one category.
- *Instance segmentation* involves identifying and separating individual images from an image.
- *Panoptic segmentation* is a combination of semantic and instance segmentations that not only segment objects and the background but also assign identities to the objects.

2.4.2 Videos

Downstream tasks for images can be extended to video. Also, because this modality incorporates time, it allows for new tasks to be created based on this dimension. We will describe a few below.

Action Recognition

Action recognition is a classification task for which categories are actions. For this task, the action is considered to cover a whole video [KCS+17] that lasts a few seconds. As for images, having a large dataset is important to properly train a highly performant network. However, collecting and annotating videos is more challenging than images as their storage is costly and annotations take more time than images for the same number of samples.

Temporal Action Detection, Spotting and Localization

Temporal Action Detection (TAD) is a task that involves detecting the starting and end points of actions. Therefore, it requires predicting the timestamps of the edge of the actions as well as their categories. It can be seen as a special case of OD where the "object" is the action. Hence as for OD, two-stage [LLL+19] and one-stage [LZS17] methods have been developed.

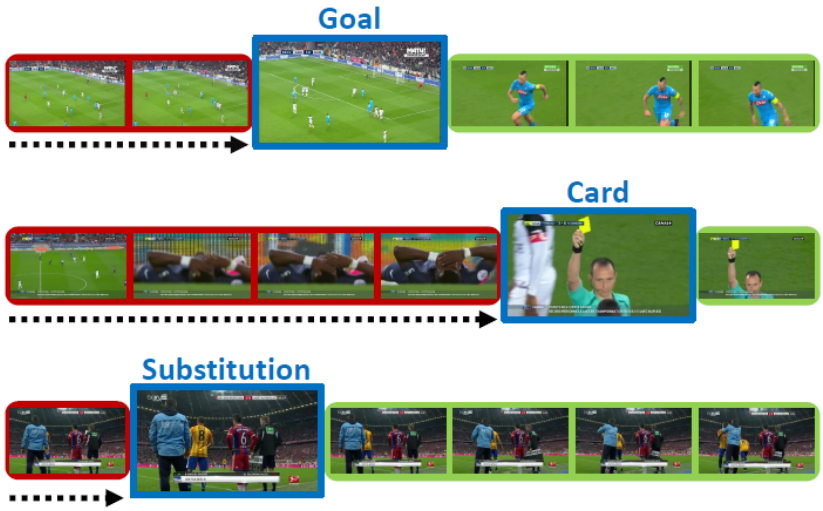


Figure 2.21: Illustration of Action Spotting. From [AIm23].

Action Spotting (AS), illustrated in Sec. 2.4.2, is a specific form of TAD for which the action is short and even collapsed to a specific timestamp-labeled action that has been introduced for football matches by [GADG18]. This requires the NN to provide a precise TAD which brought specific methods.

Temporal Action Localization (TAL) involves first performing a TAD but also predicting the coordinates of where an action occurs by providing bounding boxes as in OD. This requires the NN to not only perform well on the temporal dimension but also learn spatial features capable of solving this task.

The difficulty of these tasks is first to annotate because the starting and endpoint of an action are quite subjective. Then, having a cost-efficient method that allows the detection of the required precision of the actions, and their localization, whilst staying computationally tractable is difficult to manage and often a compromise is made.

The evaluation also suffers from difficulties as the neural network makes predictions at different timestamps that are sometimes not aligned with the truths but shifted and often require the predictions of the same timestamps with different temporal contexts, e.g. past or future. Therefore evaluation often relies on multiple tricks to achieve the best performance such as sampling clips via a sliding window with overlap to make multiple predictions at the same timestamp but from different temporal contexts and applying post-processing to eliminate and shift some predictions.

Video Retrieval

Video Retrieval is a task whose goal is to retrieve videos from a *bank* with respect to a *query*. To perform this retrieval, a similarity measure is computed between the query and all the elements of the bank. The N highest similar required by the user are retrieved with the expected same class as the query if the model performs well.

2.4.3 Metrics

To evaluate the Downstream tasks introduced in Sec. 2.4 metrics have been designed that depend on the task and we describe the ones used in this work below.

Accuracy

Accuracy@k is a classification metric that measures the number of correct class predictions over the number of total predictions. For a problem of C classes, with $\hat{\mathbf{Y}} \in \mathbb{R}^{\mathrm{N,C}}$ the prediction distribution of the model over the C classes and $\mathbf{Y} \in \mathbb{R}^{\mathrm{N,1}}$ the actual labels the accuracy is computed as follows:

$$\text{Accuracy@}k = \frac{\sum_{i=1}^{\mathrm{N}} \mathbb{1}\left(\mathrm{Y}_i \in \operatorname{argmax}\left(\hat{\mathrm{Y}}_i, k\right)\right)}{\mathrm{N}}, \tag{2.15}$$

where $\operatorname{argmax}\left(\mathrm{y}, k\right)$ returns the k indices of the k maximum values of y.

K-Nearest Neighbors (K-NN) accuracy is a special case of the accuracy that evaluates the accuracy of a K-NN algorithm applied to the output features of the data representation.

mAP

mean Average Precision is a metric used to evaluate dense predictions in OD, Segmentation, TAD, TAL. It is built upon several sub-metrics.

Predicted Positive	True Positives	False Positives
Predicted Negative	False Negatives	True Negatives
	Actual Positive	Actual Negative

Figure 2.22: Illustration of a confusion Matrix.

First, for handling the predictions the *confusion matrix*, as illustrated in Fig. 2.22, is created. It is constituted of the following elements:

- *True Positives* (TP): The model predicted the class that is the target.
- *False Positives* (FP): The model predicted a class that is not the target.
- *True Negatives* (TN): The model does not predict a class that is not the target.
- *False Negatives* (FN): The model does not predict the class that is the target.

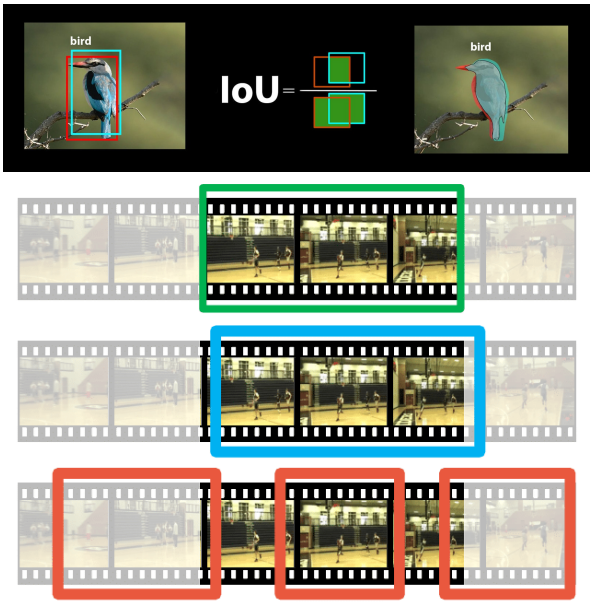


Figure 2.23: Detection examples for (up) a good bird object detection with high IOU with the ground truth from [ope23] (down) temporal action detection with green ground truth, blue good detection and red bad detections from [XZW+17].

Depending on the task, the positives and negatives are assigned differently. For classification, it is the maximum prediction from a set of categories. For regression tasks such as OD, TAD, TAL, an *Intersection over Union* (IoU) is computed between the bound of the predictions and the actual target. Indeed in these tasks, either a bounding box or a temporal segment is regressed as illustrated in Fig. 2.23. To measure the quality of predictions the I*o*U is used to assign a prediction as a positive or a negative with a bounding box. It is computed as follows:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Union of Overlap}}, \tag{2.16}$$

and a threshold is used over the IoU such as 0.5 to assign a prediction as a positive or negative. Assigned predictions are then used to make the confusion matrix.

AP is computed through the *precision* and *recall* metrics as follows:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{2.17}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{2.18}$$

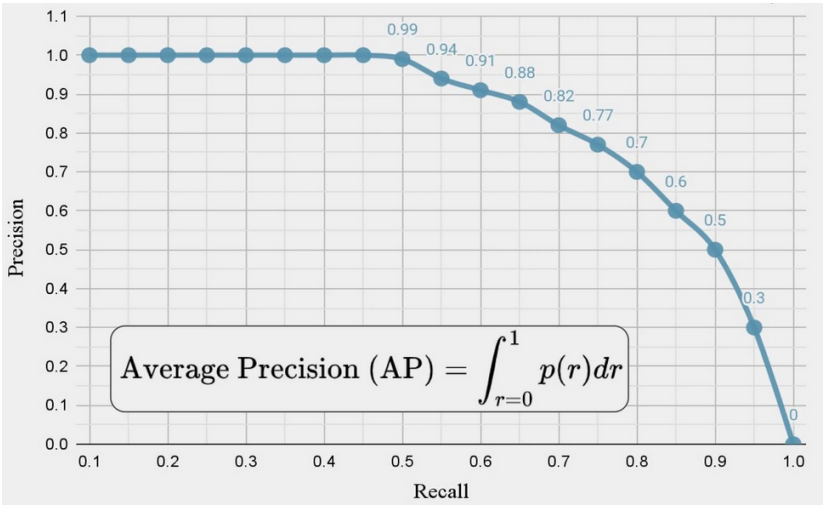


Figure 2.24: Precision given Recall from [Anw23]. AP is the area under the curve. Each point corresponds to a different threshold to consider predictions from 0 to 1.

which measures the model's capacity to predict correct positives. Precision and recall depend on a threshold, different from IoU, and only take into account predictions above this threshold. Therefore it is possible to draw a curve of Precision and Recall given the threshold such as illustrated in Sec. 2.4.3. AP is the *Area Under the Curve* (AUC) for one class and mAP is the average of AP over all classes.

Chapter 3

Self-Supervised Representation Learning for Images and Videos

Sommaire

3.1 Self-Supervised Representation Learning

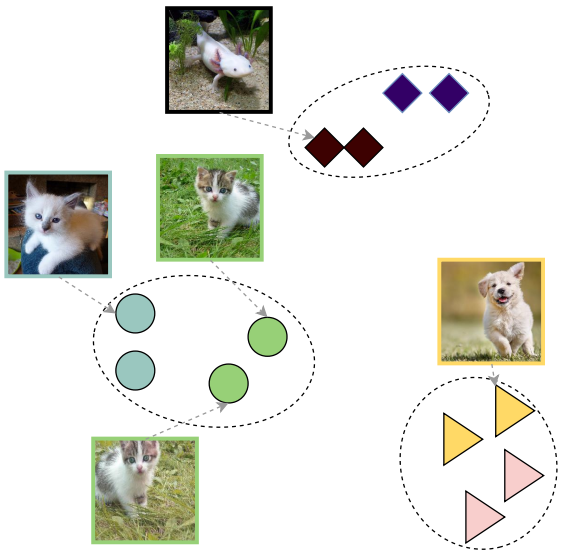


Figure 3.1: Illustration of a good representation. Similar semantic images are grouped whilst positive pairs are aligned closely and unrelated images are contrasted in separate clusters.

Representation learning (RL) [BCV13] consists in learning representations, or a *latent space* of the data to easily extract information for various tasks. A good representation captures the distribution of explanatory factors of the observed input and should be enough to pass to a supervised predictor for downstream tasks.

A good data representation, illustrated in Fig. 3.1 should meet the following criteria:

- *Aligning positives*: A small perturbation of input data should have a representation close to the original input.
- *Clustering similar instances*: Similar instances should be grouped in the same clusters, or groups, as they share highly similar characteristics.
- *Contrasting negatives*: unrelated instances should be contained in different clusters.

Meeting these criteria forces the NNs to learn general concepts such as structures in the data but also relationships between different instances. It allows for later downstream tasks to avoid learning these general concepts and reduce fine-tuning time.

Self-Supervised Learning (SSL) is one approach to performing RL where the model is trained to solve a *pretext task* by predicting certain parts of the data itself. Therefore it does not require prior labelisation. Several families of approaches have been created and this section will cover them and highlight the different challenges they face.

3.2 Geometric and Intensity Pretext-tasks

Geometric and Intensity Pretext-tasks consist of applying a Hand-Crafted transformation on the input data and making a prediction based on this transformation. Geometric pretext-tasks are transformations that change the structure aspect of the image such as rotating, cropping and intensity transformations change the value of pixels such as turning RGB images to gray. More formally, for a batch input $\mathbf{X} \in \mathbb{R}^{n,d}$ and parameters $\{\theta_1, ...\theta_n\} \in \Theta^n$ for a transformation t drawn for each element in $\mathbf{X}$, with Θ the set of possible parameters, an atomic loss function l, the objective function for a model f is:

$$\mathscr{L}^{\text{pretext}} = \sum_{i=1}^{n} l\left(\theta_i, f\left(t\left(\mathbf{X}_i; \theta_i\right)\right)\right). \tag{3.1}$$

Specific pretext tasks have been designed for image and video modalities that we will describe in the next subsections.

3.2.1 Image pretext-tasks

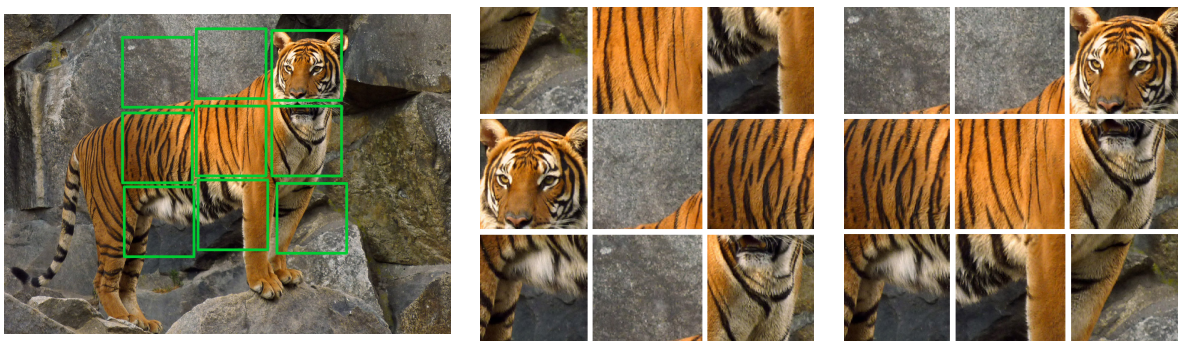


Figure 3.2: Illustration of jigsaw shuffling of image patches from [NF16].

First Self-Supervised approaches on images used position information of patches. [DGE15] takes pairs of patches and predicts where the second one is located based on the first one. To avoid trivial solutions that easily find where is located the second patch just by aligning low-level information such as lines, the patches are data augmented via padding, color jittering, ... Later [NF16], proposed an SSL approach based on the Jigsaw puzzle problem illustrated in Fig. 3.2. An image is randomly cropped in 9 different patches of the same size and shuffled and the objective function is to predict what is the position of each patch. The idea is that to solve these tasks the NN has to learn contextual information and how structural shapes are formed.

Colorization, illustrated in Fig. 3.3, is another approach developed by [ZIE16, LMS16]. First, images are grayscaled meaning the color pixels are transformed to gray values. The pretext task is to predict the missing colors. A close approach [PKD+16] inpaints a part of an image that has been masked. If the NN becomes successful at colorizing or inpainting, it means that it successfully learned semantic information on high-level structures.

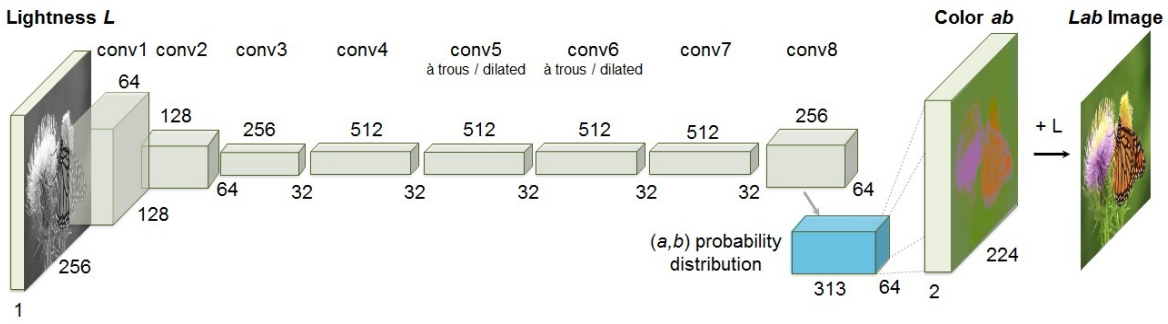


Figure 3.3: Illustration of colorization pretext task from [ZIE16].

However, some objects can have multiple colors which might be detrimental to the supervision because the model will be punished even if it predicted a relevant color.

In [DFS+16], each image in the dataset is considered as a class and the objective function is *instance discrimination*. To create multiple instances per class, the images are first cut in patches and data augmented following various transformations. Then the Neural network learns to predict the class of passed patches.

For [NPF17], the objective task is counting. It is done by splitting the image into separate patches and passing in the neural network all the patches as well as the global image. The representation of the local patches is summed and the objective function seeks to match the global representation and the summed representation.

RotNet [GSK18] rotates an image from a discrete set of 4 angles: $\{0°, 90°, 180°, 270°\}$. The objective function is to predict the rotation applied via a classification task. The Neural Network to solve this task has to recognize the original orientation of the objects.

Geometric	Intensity	Both
Patches order [DGE15, NF16] Rotation [GSK18] Counting [NPF17]	Colorization [ZIE16, LMS16] Inpainting [PKD+16]	Instance discrimination [DFS+16]

Table 3.1: Image geometric and intensity pretext-tasks methods.

Although these approaches sum up in Tab. 3.1 were capable of learning a representation sufficiently to perform well on downstream tasks, they failed to achieve results comparable to SL for two main reasons. First, these hand-crafted pretext tasks can only make NN learn as much as they create informative transformation. For example, rotation is capable of making the network learn the original orientation of the objects, but over time this task becomes quite easy to solve and does not enforce the good representation criteria. Also, the risk of these objective functions is to learn a network that is invariant to some transformations which might be important for downstream tasks. To alleviate these problems some approaches were developed to either learn from several pretext tasks [LLY+21] or combine it with contrastive learning [MvdM20, LLY+21] described in Sec. 3.4.

3.2.2 Video pretext-tasks

Geometric	Intensity
Rotation [JT18] Cubic patches order [KCK19] Frames order [MZH16, LHSY17] Clips order [XXZ+19] Speed prediction [WJL20, BEL+20]	Future prediction [LKC17] Features prediction [WJB+19, WJB+22]

Table 3.2: Video geometric and intensity pretext-tasks methods.

Some Self-Supervised pretext tasks directly took inspiration from images such as applying rotations on the frames [JT18] or solving the jigsaw puzzle on Space-Time Cubic Puzzles [KCK19], but also specifically designed video pretext tasks have been designed and summed up in Tab. 3.2.

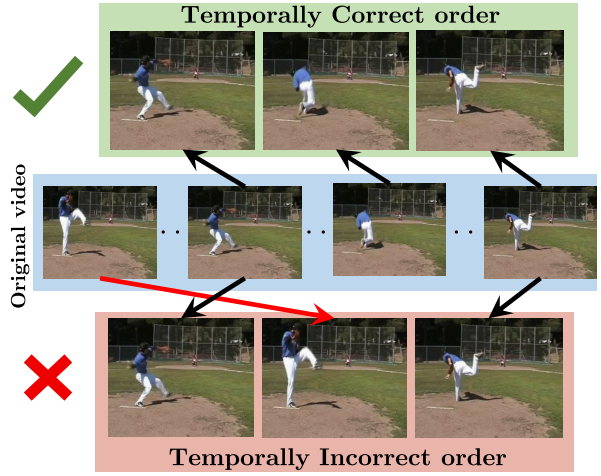


Figure 3.4: Illustration of middle frame shuffling from [MZH16].

First, some approaches focused on changing the order of the frames to make the NN learn the *arrow of time*. [MZH16] generates positives by reversing the frame order and negatives by changing the middle frame coming from another part of the video as illustrated in Fig. 3.4 and predicts which ones are correct or incorrect. [LHSY17] also creates shuffled frame sequences and seeks to predict what is the order of the frames. Another approach [XXZ+19] focused on predicting sub-clips order from a video. It passes several shuffled sub-clips in a 3D-CNN and concatenates the computed representation. Then a classifier seeks to predict the order of the subclips. The goal is also to learn the arrow of time and expect the NN to learn spatio-temporal relationships across time.

Some other approaches were designed on motion. [WJB+19, WJB+22] make use of spatial and motion characteristics from the RGB frames as well as the *Optical Flow* (OF) by extracting statistical features. The goal of the Network is to predict these multiple features as illustrated in Fig. 3.5.

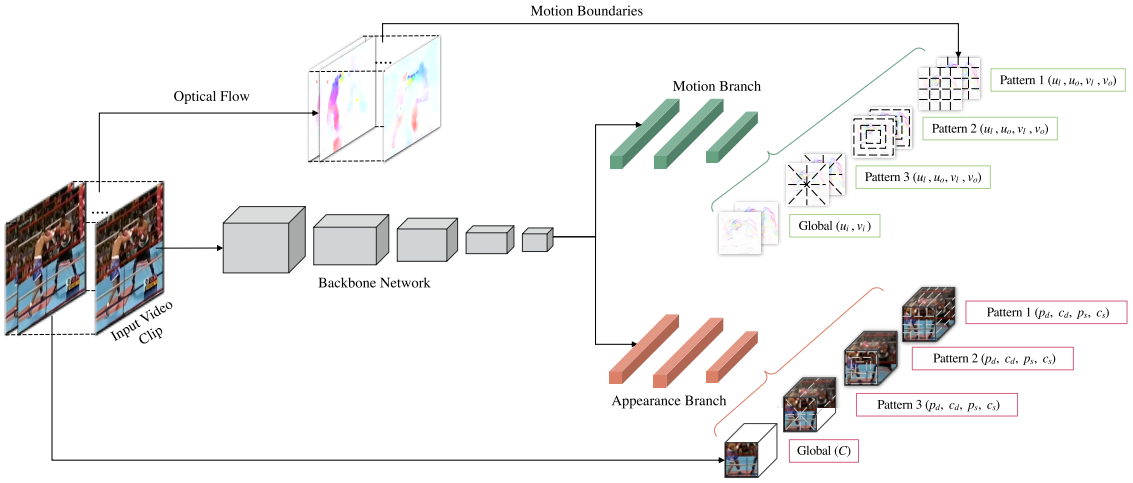


Figure 3.5: Illustration of appearance and motion statistcs prediction from [WJB+19]. Motion and RGB patterns of interest are first extracted to provide supervision to the network.

Notably, other pretext-tasks have been designed such as future prediction [LKC17] which consists of predicting from a subclip the future frames of the video. Therefore, to solve this task the NN has to learn spatio-temporal context to infer what comes next. Also, [WJL20, BEL+20] changes the speed of the sub-clip and predicts what is the speed used which makes the NN learn how fast objects are supposed to move.

However, as for images these SSL approaches failed to compete with SL because of the aforementioned issues, and as images, several approaches were proposed to increase the number of pretext tasks [LLZ+20, JMF20, PAR20, LLY+21, DZCL22] or combine them with Contrastive Learning [YZQ+21, CHH+21, SNTS21, HWHQ21, JJ21, HWH+21, NZQ+22].

3.3 Clustering

3.3.1 Clustering algorithms

Clustering is an unsupervised task that groups similar objects into *clusters*. It can be achieved by different kinds of algorithms operated directly on the input data or on features whether hand-crafted or learned via a NN.

k-Means algorithm [S+56, M+67], illustrated in Fig. 3.6, aims to partition n observations into k clusters. Therefore it strongly relies on the number of clusters. Each observation is assigned to the nearest *mean* cluster or *centroid*. The algorithm minimizes squared Euclidian distances between an observation and its assigned centroid. More formally, given the inputs $\mathbf{X} \in \mathbb{R}^{n,d}$ and the k sets $\mathbb{S} = \{S_1, \ldots, S_k\}$ with centroids $\{\mu_1, \cdots, \mu_k\}$ the

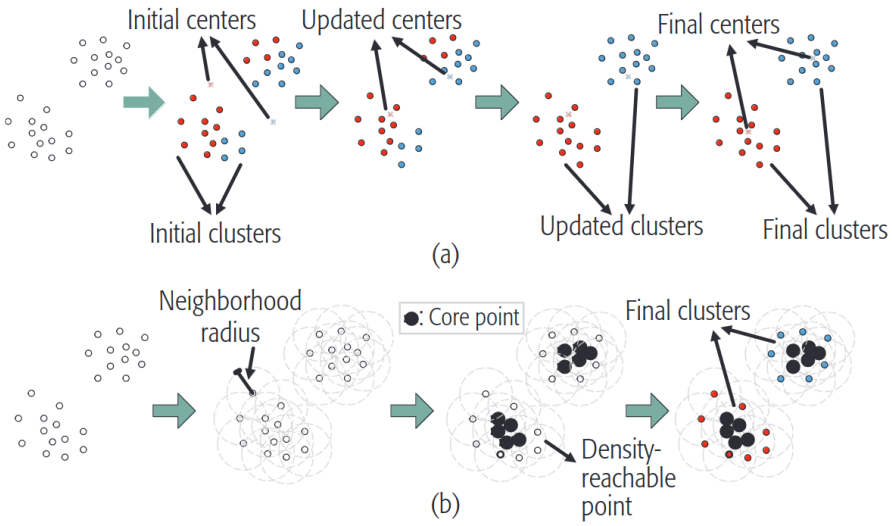


Figure 3.6: (a) k-Means algorithm. (b) DBSCAN algorithm. From [HAM+18].

objective to minimize is:

$$\arg\min_{\mathbb{S}} \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in \mathrm{S}_i} ||\boldsymbol{x} - \mu_i||^2, \tag{3.2}$$

$$\text{with } \mu_i = \frac{1}{\#\mathrm{S}_i} \sum_{\boldsymbol{x} \in \mathrm{S}_i} \boldsymbol{x}. \tag{3.3}$$

k-Means suppose that the clusters are spherical, of the same size but other algorithms do not rely on these assumptions. DBSCAN [EKSX96], illustrated in Fig. 3.6, is a density-based clustering algorithm that groups points close to each other and marks as *outliers* the ones that lie alone. It relies on three hyper-parameters, the minimum number of points to consider a cluster, the distance function, and the distance between two points to consider them related.

One of the main issues of the clustering algorithms is to find the optimal parameters that strongly depend on the shape of the data distribution and the difficulty of scaling to high-dimensional spaces and a large number of data.

3.3.2 Self-Supervised Clustering

Self-supervised clustering approaches rely on clustering algorithms applied to their representation and predicting the assigned clusters. They can be separated into two broad categories:

- *offline clustering*: In this approach, the features of the whole dataset are first extracted. Then a clustering algorithm is applied to assign clusters, or *pseudo-labels* to each example from the dataset. It is feasible when the whole dataset is avail-

able offline, its representation fits in memory and the number of optimal clusters is tractable for performing classification.

- *online clustering*: In this approach, the features of the current passed batch are clustered. Therefore the clusters are continuously updated with data as they arrive. These methods are more memory efficient but not as precise as clustering the whole dataset.

[CN12] introduces an iterative offline clustering approach based on k-means with several tricks to make it work on large-scale data. These tricks involve proper preprocessing of the data and a good initialization of the clusters. It learns sequentially its layers from the first ones to the last ones.

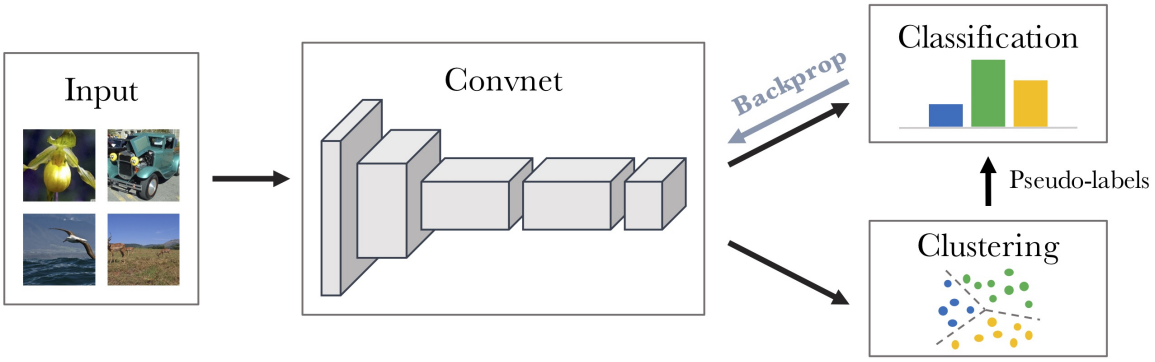


Figure 3.7: Illustration of Deep Cluster from [CBJD18]. Clusters are computed iteratively from the Neural Network image representations.

Deep Cluster [CBJD18] illustrated in Fig. 3.7 introduces an iterative offline clustering approach to pretrain an NN end-to-end, meaning all layers at the same time. At each *epoch*, meaning a complete dataset pass, or at a certain frequency, the whole dataset is passed through the NN to extract features. The k-means algorithm is then applied on top of these features to assign to each image a pseudo-label that is the centroid it has been assigned to.

In parallel, several approaches have been proposed to learn the representations as well as the clusters in an online fashion [YPB16, XGF16, LSZU16]. These approaches learned good data representation for small networks and datasets but were not scalable. [ZXL+20] scaled online clustering by storing in memory features of instances as well as centroids of clusters updated at each iteration.

[ARV20] looked at online clustering as an optimal transport problem. It is composed of two steps at each iteration. First, representation learning is based on the online cluster assignments of the batched instances. Then, self-labeling of the instances to update the clusters based on a fast version of the Sinkhorn-Knopp algorithm [Cut13].

For videos, another direction of work was to cluster the data based on multi-modality such as audio [AMK+20] or optical flow [CZM+22]. The idea is to make use of the different

modalities to denoise labels by making better cluster assignments because they disambiguate assigned clusters using different information about the input data.

Finally several works on images [LZXH21, LHL+21, TZB+22] and videos [AMK+20, CRD+21] combined contrastive learning and clustering to improve contrastive methods by making better use of negatives, or prevent collapse of contrastive without negative. It also helps clustering by learning more discriminative features, producing more robust pseudo-labels, and preventing clustering approaches from being stuck in an error loop.

3.4 Contrastive Learning

3.4.1 Towards Self-Supervised Contrastive Learning

Noise Contrastive Estimation (NCE) has been introduced by [GH10] as a new estimation principle to learn parametrized statistical models. The raw idea is to learn from real input data by contrasting it with generated noise and relies on the fact that if a model is capable of doing that, it has learned the data distribution.

From the NCE principle, one can derive a binary loss function used for classifying whether the input data comes from the true or noisy distributions. Suppose we have two sets, the true data $\mathbf{X} \in \mathbb{R}^{n,d}$ and a generated source of noise $\mathbf{Y} \in \mathbb{R}^{n,d}$ and $\mathbf{U} \in \mathbb{R}^{2n,d}$ the union of the two sets. We assign to each sample in $\mathbf{U}$ a binary class $c_i = \mathbb{1}(\mathbf{U}_i \in \mathbf{X})$ that is 1 if the instance is from $\mathbf{X}$ and 0 if it is from $\mathbf{Y}$. The objective function to minimize for the model f and binary classifier g:

$$\mathscr{L}_{\text{NCE}} = -\frac{1}{2n}\sum_{i=1}^{2n} c_i \log\big(g\left(f\left(\mathbf{U}_i\right)\right)\big) + (1-c_i)\log\big(1-g\left(f\left(\mathbf{U}_i\right)\right)\big). \tag{3.4}$$

Following works on Metric Learning [CSSB10, SC11] were based on the NCE principle to propose a *triplet loss*. It is based on the representations of queries $\{\mathbf{z}_i\}_{i\in\mathbb{N}^n}$, their positive pair representations $\{\mathbf{p}_i\}_{i\in\mathbb{N}^n}$ and the negative representations $\{\mathbf{n}_i\}_{i\in\mathbb{N}^n}$. The concepts of positives and negatives depend on the task and learning paradigm, we define them for instance discrimination which is at the basis of modern contrastive learning at the end of this subsection. The triplet loss forces the *positive pair* of representations $\{\mathbf{z}_i\}_{i\in\mathbb{N}^n}$ and $\{\mathbf{p}_i\}_{i\in\mathbb{N}^n}$ to lie in a close margin C and push negatives $\{\mathbf{n}_i\}_{i\in\mathbb{N}^n}$ further from this margin:

$$\mathscr{L}_{triplet} = \sum_{i=1}^{n} \max(d(\mathbf{z}_i,\mathbf{p}_i) - d(\mathbf{z}_i,\mathbf{n}_i), \mathrm{C}), \tag{3.5}$$

with d a measure distance.

Later, [SXJS16] introduces the *(N+1)-tuple loss* or *InfoNCE loss* that contrasts a *positive* pair with several *negatives*. To do so it uses a categorical cross-entropy loss to identify a positive sample amongst a set of several negative samples.

It has been applied for Self-Supervised Learning by *CPC* [vdOLV18] which stands for Contrastive Predicting Coding. The InfoNCE loss is defined as follows for a batch of representation $\{\mathbf{z}_i\}_{i\in\mathbb{N}^n}$, their positive $\{\mathbf{p}_i\}_{i\in\mathbb{N}^n}$ and a set of negatives $\{\mathbf{n}_i\}_{i\in\mathbb{N}^l}$ that can be the same or different for each sample:

$$\mathscr{L}_{InfoNCE} = -\frac{1}{n}\sum_{i=1}^{n}\log\left(\frac{e^{d(\mathbf{z}_i,\mathbf{p}_i)/\tau}}{e^{d(\mathbf{z}_i,\mathbf{p}_i)/\tau}+\sum_{j=1}^{m}e^{d(\mathbf{z}_i,\mathbf{n}_j)/\tau}}\right), \tag{3.6}$$

where $d(\mathbf{k},\mathbf{l})$ is the cosine similarity between $\mathbf{k}$ and $\mathbf{l}$.

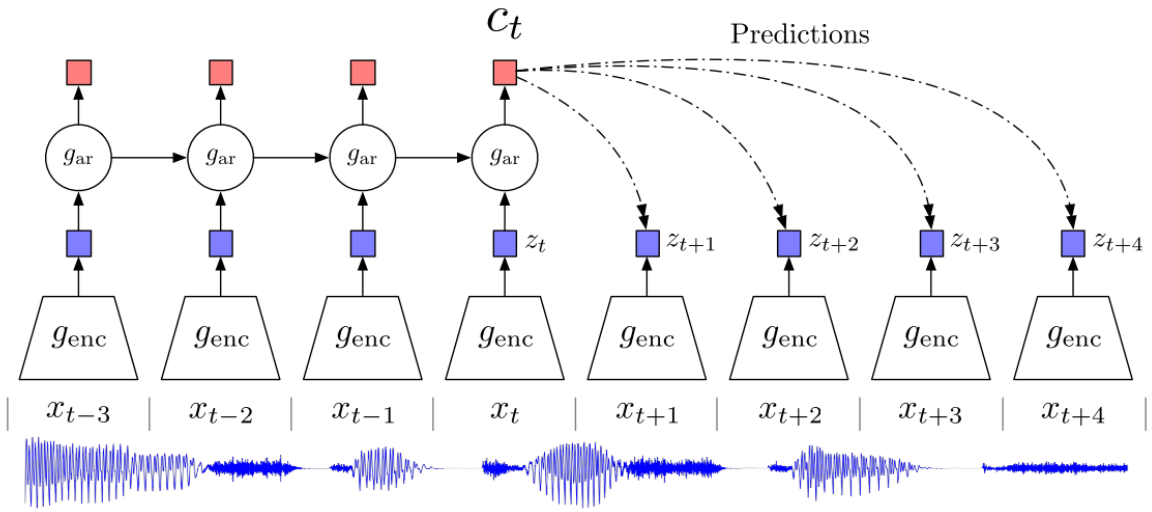


Figure 3.8: From [vdOLV18], a context is built from representations of input data. The generated context is used to predict the future representation and contrasted with negative representations from other instances.

It builds a context with an auto-regressive encoder on top of an encoder, as illustrated in Fig. 3.8, that encodes sequentially parts of input data that can be audio, image, video, ... The goal of CPC is to predict future representations based on the similarities between them and the context contrasted with other instance representations.

CPC by outperforming previously proposed pretext tasks paved the way for plenty of contrastive methods described in the next sub-sections. The main challenges these methods face are:

- How to generate positives.
- How to deal with negatives (for methods using them explicitly).
- Avoid representation collapse.

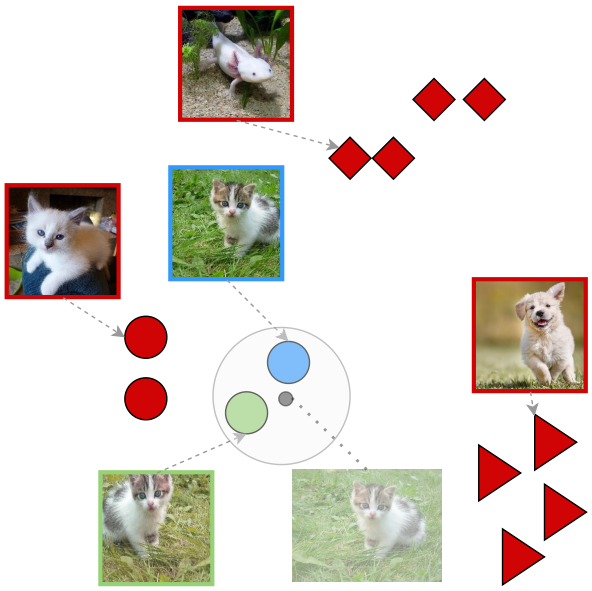


Figure 3.9: Illustration of a query cat (blue circle), its positive (green circle) which is the same cat as the query after some data augmentations, and all the negatives (red shapes). Because there are no labels, even highly similar images are considered as negatives.

Self-supervised contrastive learning methods are built upon Instance Discrimination to construct positives meaning that a positive representation pair is two representations computed from a part of the same based image. In most approaches, negatives are considered as all representations coming from other instances as illustrated in Fig. 3.9.

3.4.2 Contrastive Learning with Negatives

As introduced in the previous sub-section, CPC [vdOLV18] is one of the first methods to employ Contrastive Learning with negatives. It also proves that minimizing $\mathrm{L}_{InfoNCE}$ also minimizes the lower bound of mutual information [Sha48] MI between the input representation $\mathbf{z}$ and its positive representation $\mathbf{p}$:

$$\mathrm{MI}(\mathbf{z},\mathbf{p}) \geq \log(\mathrm{N}) - \mathscr{L}_{InfoNCE}, \tag{3.7}$$

where N stands for the number of negatives. The mutual information measures the dependence between two variables. Therefore, to learn a good data representation, this should be maximized. Eq. (3.7) shows that minimizing the InfoNCE loss as well as increasing the number of negatives helps to do that.

To increase the variety of positives and learn a more general data representation [CKNH20] proposes a siamese contrastive learning approach called SimCLR. It first introduced the basics of modern contrastive learning via several steps, illustrated in Fig. 3.10, and listed here:

- Data augmentations, to construct the positive pair of views.

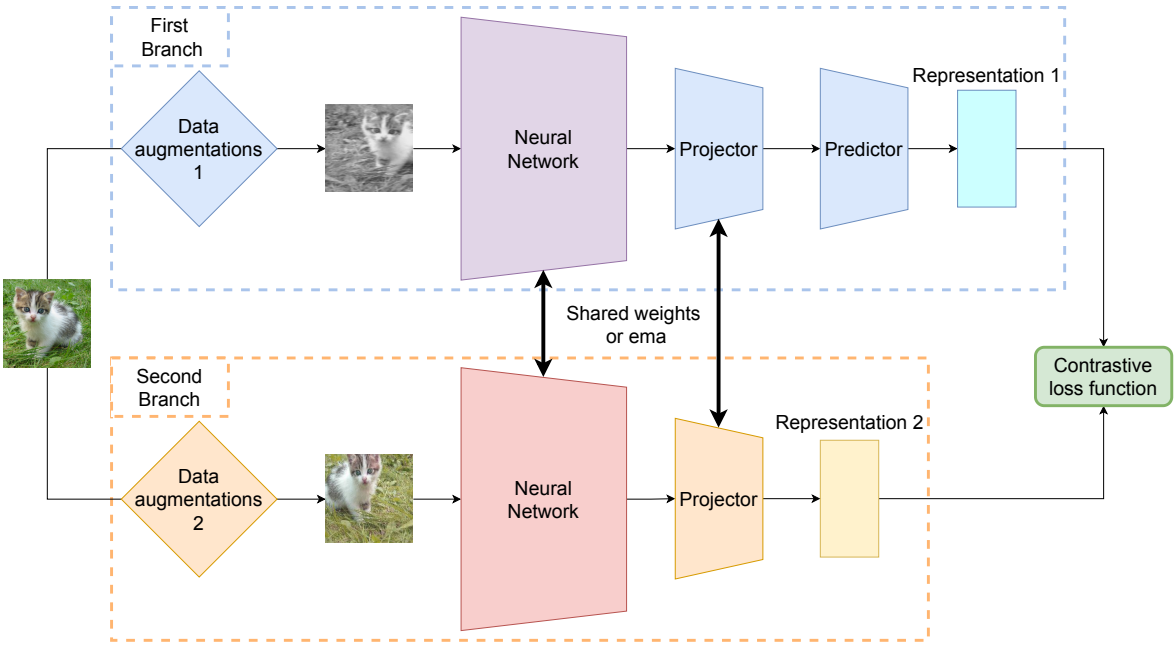


Figure 3.10: Siamese architecture to perform contrastive learning. An input goes through two branches that contain the same architecture: a data augmentation to form positive views, a neural network and a projector. Optionally the first branch can contain a predictor. Depending on the method, the second branch can share the same weights as the first branch or be updated by an Exponential Moving Average (ema) of the first branch. The contrastive loss function is applied to the output positive representations of both branches. The neural network of the first branch is kept at the end of training.

- Non-linear projector to avoid invariant representation to data augmentations.
- Siamese architecture to process the pairs.

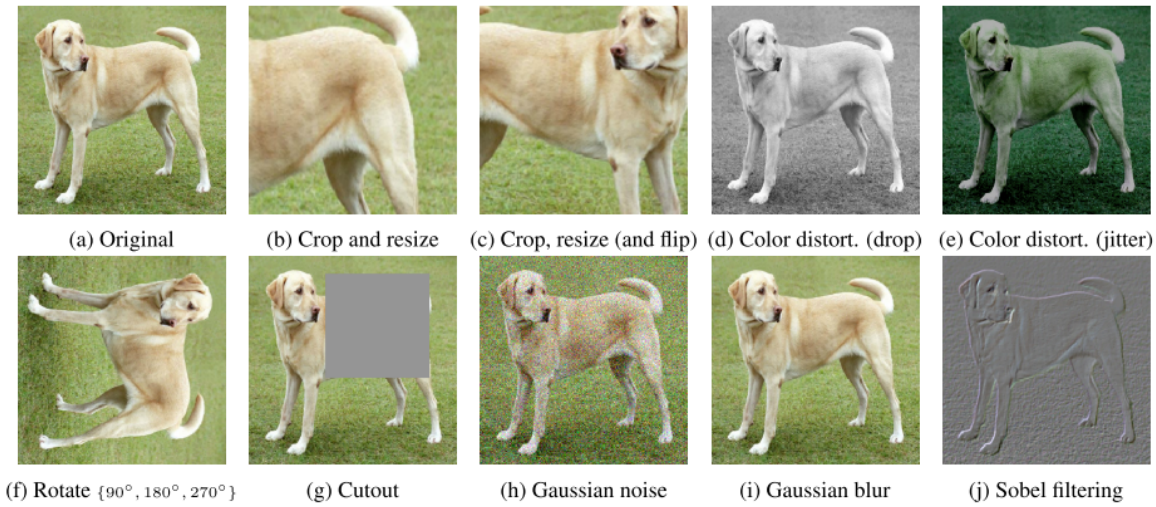


Figure 3.11: Several data augmentations from [CKNH20].

First, data augmentations, some illustrated in Fig. 3.11, have an important role in obtaining the best performance. [CKNH20] show that the crucial one is cropping to avoid complete overlap between the two views that allow the neural network to converge to an easy solution. Increasing the strength of the data augmentations by applying for exam-

ple color jittering, grayscaling, and blurring decreases the mutual information between the positive pairs forcing the neural network to learn high-level concept information to align them. InfoMIN [TSP+20] further pushed this idea by theoretically and empirically proving positive views should have low mutual information to improve generalization and propose a new set of data augmentations. Other works also took interest in the data augmentations strategy to maintain the right amount of mutual information between positive pairs by performing a better cropping strategy than random [PWZ+22] or maintaining task-relevant information [WGDL22].

Second the projector introduced by [CKNH20] is a MLP network that has several purposes. The first one as they empirically show, is that it allows the pretrained neural network to maintain semantic information that could be lost because of the data augmentations and alignment of positive pairs. For example, if color jittering is applied, to align positive pairs a part of the network has to become invariant to the color. This is where the non-linear projector shines as it becomes invariant to the color which helps the pretrained neural network to keep color information. Its second interest is to project to a small dimension the output representations which reduces the computational cost of the similarity matrix between the representations as well as the risk of the curse of dimensionality [Bel66]. Several works studied the effect of the projector and show that the projector is useful when the self-supervised learning task and downstream task are not aligned [BBG+23] and to handle noisy image augmentations [BIS+23].

Finally, the siamese architecture introduced by [CKNH20] involves two branches to handle the input data by augmenting it, passing the augmented input through the neural network, a projector and applying the contrastive loss on the output representations. SimCLR uses share weights between the two branches but other approaches such as MoCo [HFW+20] update the second branch according to an Exponential Moving Average (ema) of the first one. To improve performance MoCov2 [CFGH20] has been developed that follows the projector and augmentations guidelines from SimCLR.

As shown in Eq. (3.7), having a large number of negatives has a role in optimizing the lower bound of the mutual information between the positive pairs. To increase the number of negatives, NPID [WXYL18] uses a bank of representations for all the images contained in the dataset. The positive pair of an image passed in the neural network is the representation of the same instance contained in the bank. MoCo [HFW+20] (v2 [CFGH20]) also makes use of a bank but from a smaller momentum bank of representations that is filled by its second branch data representations to keep relevant fresher representations as illustrated in Fig. 3.12. SimCLR [CKNH20] (v2 [CKS+20]) removes the need for a memory bank by using large batches followed by MoCov3 [CXH21] that keeps nonetheless a momentum encoder to increase performance and adds a predictor to its first branch.

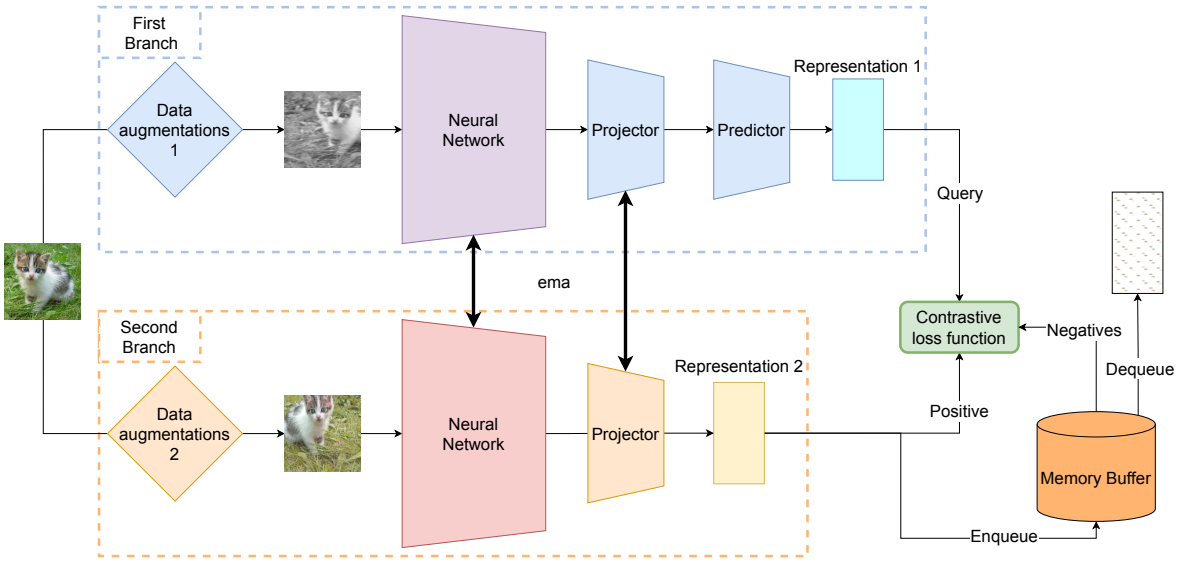


Figure 3.12: Siamese architecture to perform contrastive learning with a momentum memory buffer. It is the same architecture as in Fig. 3.10 with an Exponential Moving Average (ema) update for the second branch and its output of the second branch feeds a momentum memory via a First In First Out (FIFO) update. The momentum memory buffer serves as a collection of negatives for the contrastive loss.

(a) IR, CMC, MoCo (b) Ring

Figure 3.13: Illustration from [WMZ+21] to compare (a) standard contrastive learning negative sampler and (b) Ring negative sampler [WMZ+21]. In black the representation of an example x_i, in gray the representation of its positive and in red the representation of its negative. The gray area is the support of the distribution to sample negatives. Ring only considers negatives further away from the query than its positive and closer than a margin whilst standard contrastive learning does not have a constraint.

However, not all negatives are equal as shown by [CFSM20] and the most difficult, also called *hard negatives*, defined by the ones most similar to the query, are the most crucial to improve performance. Indeed, to learn interesting information by contrasting positives with negatives, they should be sufficiently similar to positives to learn fine-grained information. However, *False Negatives* (FN), which are instances with the same semantic information as the query can degrade performance if they are contrasted on various downstream tasks, which is also called the *class collision problem* [CFSM20, WWSY21, CRL+20].

Several samplers have been designed to deal with such negatives. Some focus on generating hard ones by using mixup [ZCDL18], a data interpolation strategy, to mix positives and negatives [KSP+20, ZHH+21] at the latent representation. Other works debiase the contrastive learning strategy to mitigate the effect of FN [CRL+20] and select the hard negatives [RCSJ21, JW23]. Truncated-triplet [WWW+21] optimizes a triplet loss using the

k-th similar element as a negative. It also proposes to build a soft negative that is an average of the most similar negatives. Ring [WMZ+21], proposes to remove from the negatives the one closer to the query than its positive as they are probably FN and only keep the one close enough to convey useful information and are not too *easy* as illustrated in Fig. 3.13.

Other works focused on properly using positives. i-Mix [LZS+21] considers semi-positive by applying mixup on the input data and applying a contrastive loss with a soft positive class. NNCLR [DAT+21] does not use directly the positive to perform contrastive learning but the most similar negative to the positive from a queue of representation. AdCo [HWHQ21] performs adversarial learning to generate positive views. RCL [TZB+22] uses a hierarchical clustering algorithm to consider several positives at different levels to either push or pull other instances.

To further regularize the contrastive loss, various strategies have been explored. Some added a regularization objective along the contrastive loss that considers the similarities among instances. CO2 [WWSY21] and RELIC [MMW+21] add a consistency regularization term that matches the distribution of similarity for a query and its positive. PCL [LZXH21] and WCL [ZWY+21] combine unsupervised clustering with contrastive learning to tighten representations of similar instances. To do so the regularization objective function is a contrastive learning objective that considers other instances in the same clusters as positives. In a close direction, other works [YLH+21, YLH+23] proposed a denoised contrastive loss that reduces or reverses the gradient for medium and highly similar negatives. They use hard margins between different categories of negatives. Also, DCL [YHH+22] proposes to remove from the denominator the positive term to enforce negative-positive-decoupling in the objective function:

$$\mathscr{L}_{\text{DCL}} = -\frac{1}{n}\sum_{i=1}^{n}\log\left(\frac{e^{d(\mathbf{z}_i,\mathbf{p}_i)/\tau}}{\sum_{j=1}^{m}e^{d(\mathbf{z}_i,\mathbf{n}_j)/\tau}}\right) = -\frac{1}{n}\sum_{i=1}^{n}\left(d(\mathbf{z}_i,\mathbf{p}_i)/\tau - \log\left(\sum_{j=1}^{m}e^{d(\mathbf{z}_i,\mathbf{n}_j)/\tau}\right)\right). \quad (3.8)$$

[LAG+21] uses the Conditional Entropy Bottleneck to control the amount of compression learned in the data representation via SimCLR to improve performance on downstream tasks.

Contrastive learning optimizes two different objectives [WI20, CLL21], alignment of positive views and maximizing the entropy of a prior distribution. Because the majority of contrastive objective functions deal with representations that lie on the hypersphere, the second objective seeks to make the representation fill the whole surface of the hypersphere. The alignment prevents the collapse to a uniform distribution by learning high-level concepts such as structures, and relationships between different parts of an image. Therefore, contrastive learning learns the relations, also called semantic similarity, between instances based on the meaning or semantics they convey by optimizing. A different kind of learning emerged to directly learn these relations which is *relational learn-*

ing. ReSSL [ZYW+21] introduces this explicit relational learning objective by maintaining consistency of pairwise similarities between strong and weak augmented views. As contrastive learning, it is based on a Siamese architecture with the second branch that receives as input the weak augmented view to estimate the relations via pairwise similarities. The first branch objective is to predict the distribution of pairwise similarities between instances via the strong view. Close approaches to relational learning relied on *self-supervised knowledge distillation* [PKLC19, FWW+21, KTP20] for which a student model seeks to predict the distribution of similarities among instances computed by a larger pre-trained teacher.

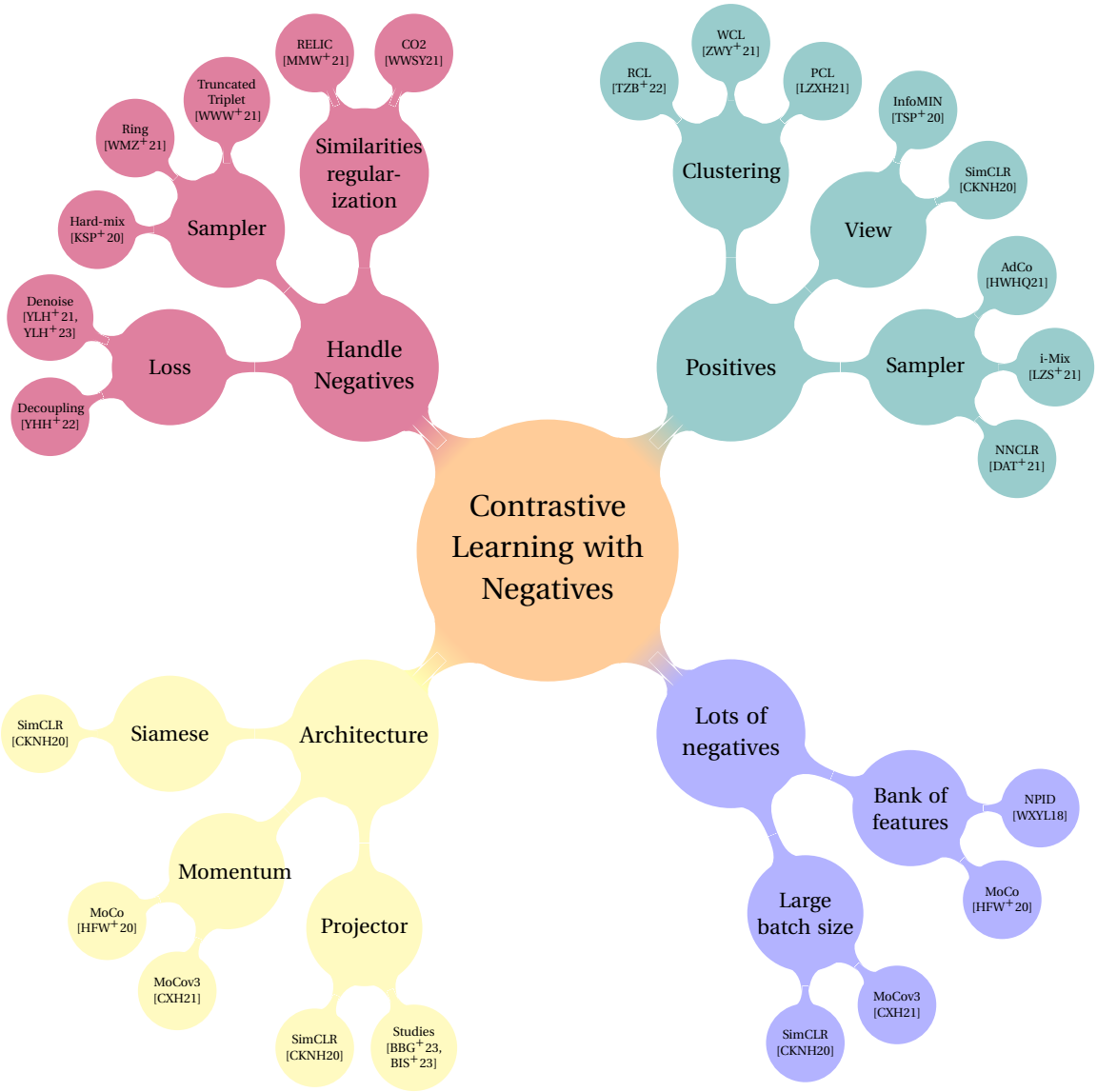


Figure 3.14: Overview of the general concepts and main methods for image contrastive learning using negatives.

We summarize in Fig. 3.14 the different concepts introduced for Contrastive Learning with Negatives associated with the main papers introducing or working on the concepts.

3.4.3 Contrastive Learning without Negatives

To avoid dealing with negatives, several contrastive learning approaches have been developed with an objective function that deals explicitly with only the positive. Whilst the objective function does not explicitly make use of the negatives, they are considered implicitly either from a regularization term or the architecture.

Figure 3.15: Multi-crop strategy. Two global views are formed from a raw image as well as several local views.

SwaV [CMM+20] is the first to propose such an objective by combining online clustering and contrastive learning. To do so it adopts a Siamese architecture with shared weights for two views from the same base image. The output representations are multiplied by a prototype matrix to compute codes for each view which correspond to a soft assignment to each prototype. The objective function seeks to predict from one view what it is the code of the other view by contrasting the prototypes. To make sure the distribution does not collapse to one prototype, the online clustering algorithm enforces a uniform repartition of the data onto the prototypes. SwaV also introduced the *multi-crop strategy*, illustrated in Fig. 3.15 that creates multiple local views, smaller than traditional views (96×96 resolution instead of 224×224), to learn a representation with local information.

BYOL [GSA+20] also adopts a Siamese architecture with asymmetries to avoid collapse [TCG21]. First, the distributions of data augmentations of the two branches are different. Second, it performs *self-knowledge distillation* as MoCo [HFW+20] by updating the second branch parameters, called *target branch* via ema updates of the first branch parameters, called *online branch* or *student branch*. Thirdly, a predictor is added to the online branch to map the output of the online branch to the target branch. The objective function is simply a l_2-normalized reconstruction loss between the output representations q_s and z_t of the online and target branches respectively:

$$\mathscr{L}_{\text{BYOL}} = ||q_s - z_t||_2^2. \tag{3.9}$$

Several works studied what makes BYOL work and what are its critical components. [FA20] showed that originally BYOL could be cast as an implicit contrastive learning method through normalization and it was key to its success. This claim has been refuted by [RGA+20] that instead shows proper initialization is required for BYOL to work. [PZN+22] showed that the momentum weight update of the target branch could only be located on the projector. SimSiam [CH21] replaces the momentum encoder with a stop-gradient which was enough to remove collapse but at the cost of worse performance than BYOL for long pretraining.

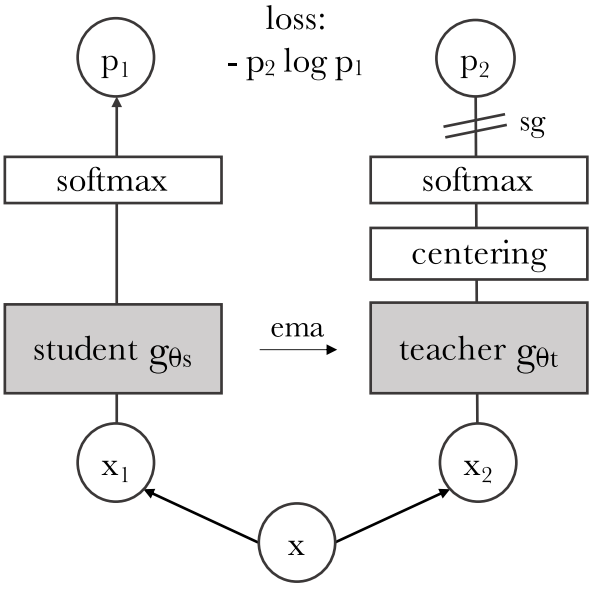


Figure 3.16: Illustration of DINO from [CTM+21]. The target branch produces a soft prediction over a set of pseudo-classes regularized by centering to avoid collapses. The student branch seeks to predict the target distribution.

DINO [CTM+21] illustrated in Fig. 3.16 also relies on a self-knowledge distillation siamese pipeline. Both branches contain an encoder and a projector that in opposition to contrastive learning approaches with negatives and BYOL does not compress the representation but projects to a space with high dimensions that are considered *pseudo-classes* after a softmax operation is applied. For the target branch, a centering operation is performed before softmax to force the batch representations to lie on various pseudo-classes and not collapse on a single class. Hence, this centering operation is an implicit contrastive learning regularization. The student branch seeks to predict the distribution of the target branch for each instance.

W-MSE [ESSS21] and Barlow-Twins [ZJM+21] are based on the idea that the matrix of correlations of the representations of two positives should reach the identity. This would mean that ideally, the representations would learn informative information about the input without redundancies. W-MSE applies a whitening operation on the batch and Barlow-Twins a batch normalization of the inputs which causes implicit contrastive learn-

ing. VicReg [BPL22] free Barlow-Twins from this normalization but add a loss that forces a variance across the batches maintaining the implicit contrastive objective.

3.4.4 Contrastive Learning for Videos

After their success in learning image representation, CL approaches have been extended to videos to learn one global representation of a video clip.

[LRO+20] extended CPC to videos by applying a 2D CNN on each image and building a context on the global representation of the successive images. The objective function seeks to predict the feature representation among a set of representations. Similarly, Mem-DPC [HXZ20a] also extended CPC but used sub-clips and 3D CNNs to compute cubic representations and build the context.

Siamese contrastive learning approaches have also been extended to videos to learn representations. The pipeline illustrated in Fig. 3.10 is essentially the same for videos with the differences being:

- *Input modality*: inputs are sequences of images instead of single images.
- *Data augmentations*: applied on sequences of images and the time dimension offers new types of data augmentations such as changing the speed.
- *The NN architecture*: dedicated architectures have been developed for handling videos such as S3D [ZDWW18], ResNet3D [HKS18], SlowFast [FFMH19], ...

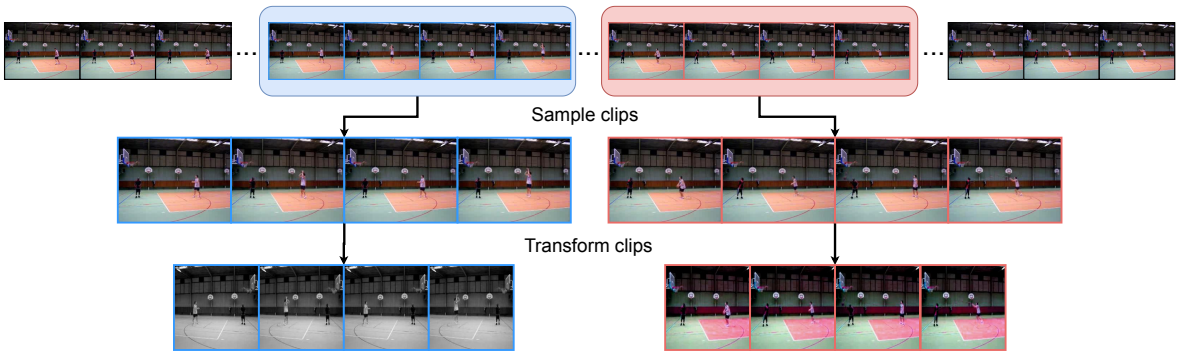


Figure 3.17: Two clips sampled randomly from a video and independently transformed to form positive views.

To form positive views several approaches have been studied to sample two sub-clips from the same videos, as illustrated in Fig. 3.17, and apply a contrastive learning pipeline afterward. CVRL [QMG+21] extended SimCLR to videos and proposed a temporal sampler for creating temporally overlapped views to maintain common information. Yet, the overlap is not total which can avoid spatial redundancy further reduced with different resized crops. Similarly, [FFX+21] extended SimCLR, MoCo, SwaV and BYOL to videos. To form positive views they randomly sample clips from a video uniformly which means that

there might not be overlap. They also generalized the multi-crop procedure introduced for images by [CMM+20] by sampling more than two sub-clips per video. Whilst this considerably improves their results, it comes with a high computational cost.

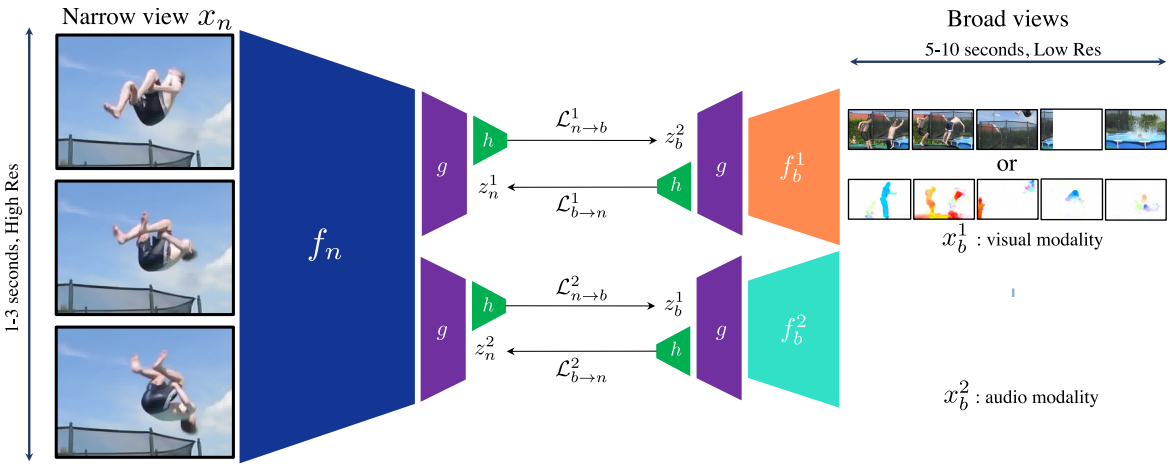


Figure 3.18: Illustration of BraVe from [RLA+21]. Contrastive Learning is applied from an asymmetric pipeline via a high-resolution view with a narrow temporal context and a low-resolution view with a broad temporal context. It supports multi-modality.

Other approaches also made use of the time dimension to build positives with different width temporal contexts. VideoMoCo [PSY+21] adopts a generative contrastive approach where one view is generated and a certain amount of frames of the generated frame are masked removing temporal information. VTHCL [YXDZ20], BraVe [RLA+21] and LSTCL [WBTT22] adopt an asymmetric Siamese pipeline with differently sampled views. VTHCL samples the same clip but with a fast framerate and a low framerate that are passed through the slow and fast networks of the SlowFast architecture. BraVe illustrated in Fig. 3.18 has a broad temporal view with low spatial resolution and one broad temporal view sampled with high resolution. LSTCL as BraVe samples one view with a large temporal context and another with a small temporal context but is applied on transformers whilst Brave was on 3D CNN. Also, LSTCL uses NN-shared weights but different projectors and predictors. Similarly, TCLR [DGRS22] has a global-local contrastive loss that considers part of features from a global view and global features from a local view. VCLR [KZZ+21] explores using intra and inter-contrastive losses by using multiple clips from the same video as positive or negative depending on the loss to learn local and global contexts. SVT [RNK+22] made use of transformers with global spatiotemporal views and local spatiotemporal views that it sought to align.

As for images, different variants of contrastive learning have been proposed such as hierarchical contrastive learning to learn features at different scales [QLL+21] and cascade retrieval [WLHK22]. Provico [PLKS22] proposed a stochastic contrastive loss that constructs positive and negative pairs based on a probabilistic distribution. The loss is weighted by the certainty of positiveness. IIC [TWY20, TWY22] proposes to use multiple views of the same data to form positives. Different clips of the same video are considered intra-

positives but if a temporal transformation such as shuffling the frames is applied, they become hard intra-negatives. [WGL+21] and [DLY+22] propose to change the background of the videos to make the representation less focused on irrelevant spatial information.

Previously mentioned methods focused on building representations from videos to perform global video representation and are evaluated on tasks that focus on general patterns such as Action Recognition on a whole clip. In the opposite direction, CARL [CWLC22] followed by [ZLZS23] proposes a sequence contrastive learning approach to learn frame-wise representations to evaluate fine-grained frame retrieval tasks. Their objective is to capture fine-grained representations whilst aligning close-frame representations. These local-temporal representations are later fine-tuned to tasks taking advantage of such features such as TAD.

Although in our work we solely focused on making use of the RGB video stream, different approaches have been developed to make use of different modalities. These modalities are used in two different ways with contrastive learning: a total replacement of the RGB views or one view from RGB and its positive from another modality. For pretraining, the main idea of using these modalities is that they provide another perspective of the data which permits modeling a better data representation. For example *Optical Flow* contains only motion direction and intensity which require the model to focus on movements rather than spatial information. During fine-tuning and inference for downstream tasks, the predictions of the NNs using the different modalities can be *fused*, or *aggregated* to improve results. The different modalities used are:

- *The text data* [SBMS19, MAS+20] available or generated by Automatic Speech Recognition (ASR) solutions.
- *The Optical Flow* (OF) [HXZ20a, LRO+20, HXZ20b, PAR20, HSL+21, HLW+21, RLA+21, TGSH22, XTM22, NZQ+22, CZM+22] of the video which is the 2D map of pixels' movement between successive images. OF requires a computational cost and a model for preprocessing the RGB data before training and obtaining for each video its corresponding OF. To alleviate this issue, some works approximate the magnitude of the OF by computing the *RGB difference* between frames and use it instead of OF. It can be computed at each iteration with negligible cost.
- *The audio* [AMK+20, PAR20, RLA+21] of the video.

The key concepts of video contrastive learning along with principal methods are summarized in Fig. 3.19.

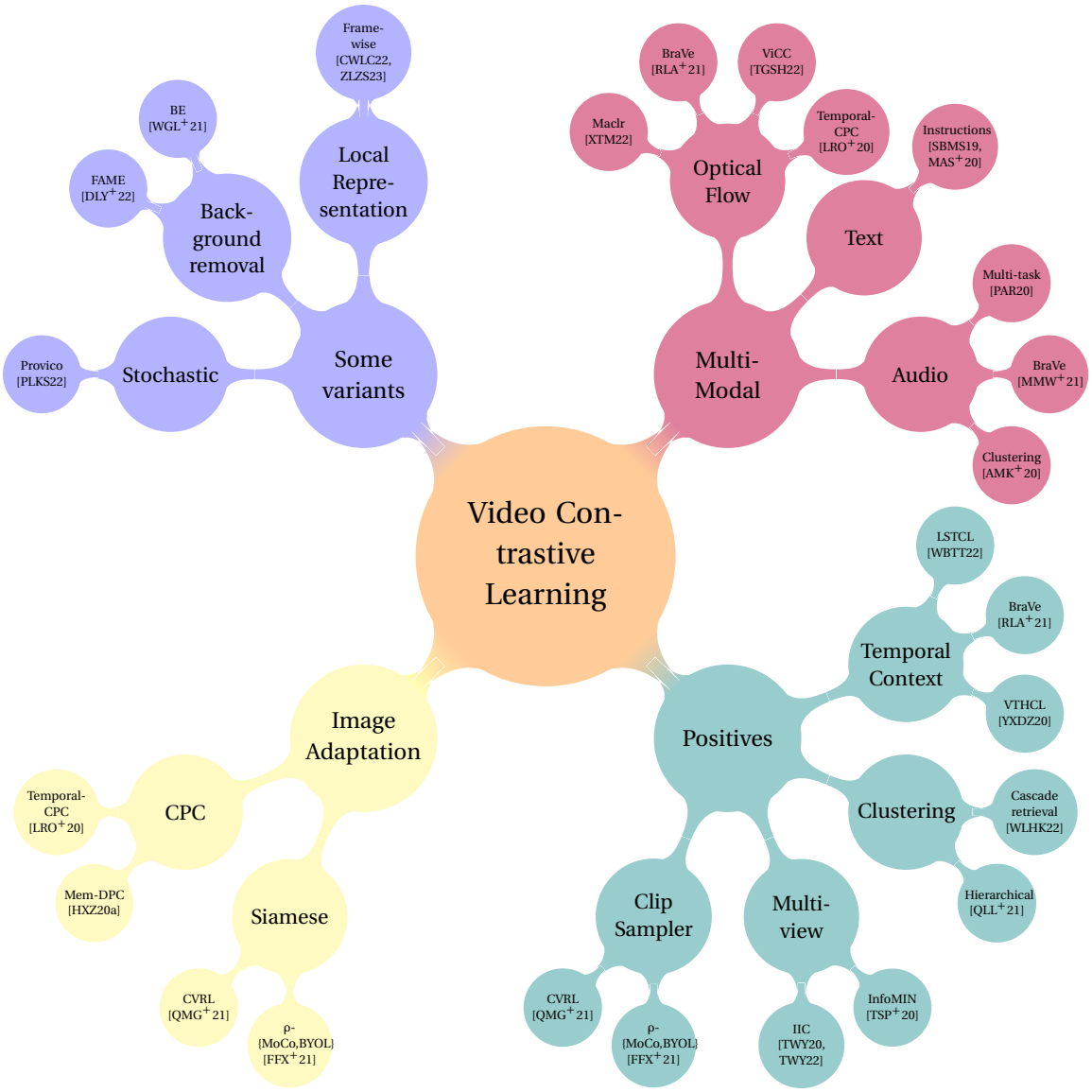


Figure 3.19: Overview of the general concepts and main methods for video contrastive learning.

3.5 Masked Modeling

Masked Modeling (MM) is a generative approach that masks part of an input and tries to predict the missing data. It has first shown tremendous success in NLP representation learning [RNS+18, DCLT19] for the transformer architectures [VSP+17]. For example, BERT [RNS+18] replaces some input tokens with a learned mask token and the objective function is a classification loss to recover the correct token. However, directly applying masked tokenization and positional embeddings on images fails to learn a representation of good quality [CRC+20].

The introduction of Vision Transformers [DBK+21] allowed for Masked Image Modeling (MIM) and Masked Video Modeling (MVM) approaches to emerge and perform competitively with CL approaches as explained in the next sections and summarized in Tab. 3.3.

3.5.1 Masked Modelling for Images and Videos

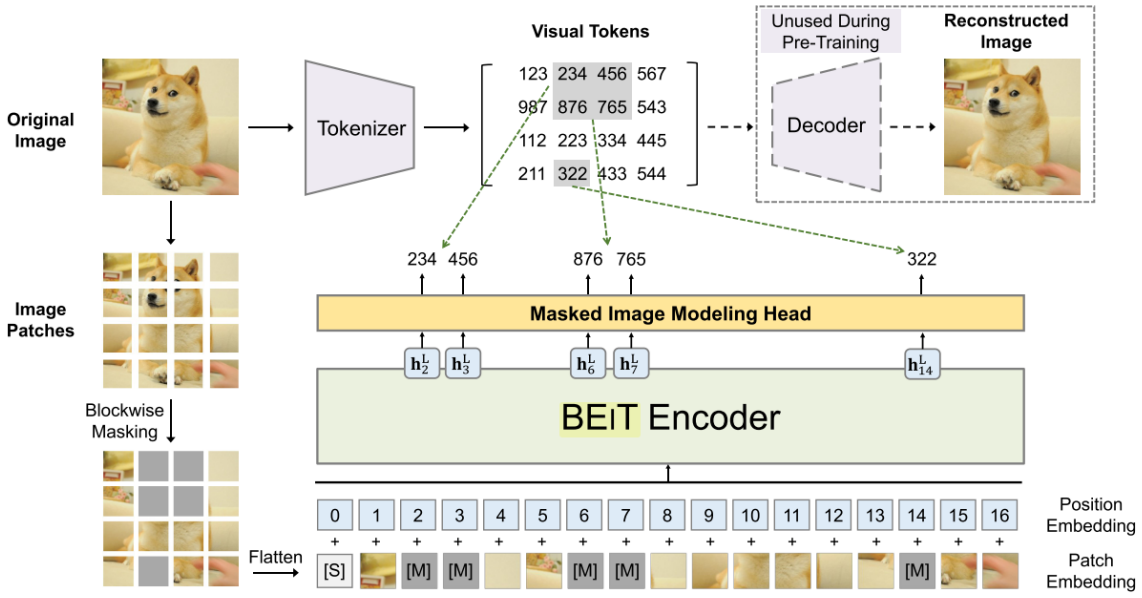


Figure 3.20: Illustration of BEiT [BDPW22] from its paper. The input image is tokenized to input patches, each associated with a visual token computed by a tokenizer. Some patches are corrupted and replaced with a learned masked token. The tokens pass through a transformer encoder and the objective function is to predict the visual tokens of the masked patches.

BEiT [BDPW22], illustrated in Fig. 3.20, is a direct extension of BERT [RNS+18] to images. It consists of first training an image tokenizer via a discrete variational autoencoder (dVAE) [Rol17], trained via a reconstruction loss, to assign to an input image a visual token for each of its patches. Some of the token patches are masked according to blockwise masking, e.g. mask multiple patches according to blocks, and the objective function of BEiT is to predict the visual tokens of the corrupted tokens. Several works built on BEiT to improve the tokenizer, mc-BEiT [LGY+22] soften the classification problem by assigning soft labels instead of hard labels to patches. PeCo [DBZ+23] enforces during the training of the tokenizer perceptual similarity between the original and the reconstructed images.

Using tokenizers has the disadvantage of having two pretraining steps that require careful attention to design choices. Moreover, dVAEs discretize the visual space whereas the visual space is continuous and therefore suffers from information loss. Consequently, end-to-end approaches have been designed to remove the need for a tokenizer.

Masked Autoencoder (MAE) [HCX+22], illustrated Fig. 3.21, proposes an encoder-decoder architecture. Some patches of the input are masked and the visible tokens are passed through a transformer encoder. The output tokens are completed with learnable mask tokens at the positions of the masked patches. The complete set of tokens is fed to a transformer decoder and the objective function predicts the pixels of the masked patches from the output mask tokens. Although this pipeline involves two components, the encoder and the decoder, the cost of pretraining is alleviated by the fact that only visible to-

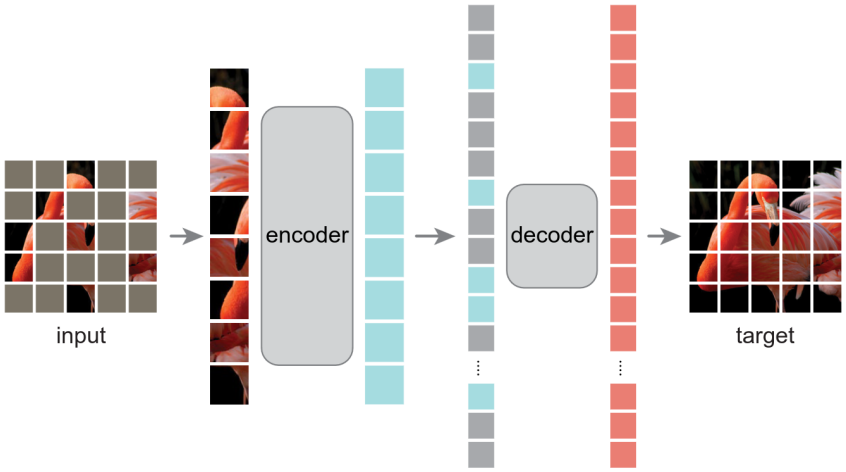


Figure 3.21: Illustration of MAE [HCX+22] from its paper. Some patches are masked. The visible tokens pass through an encoder and its output is complemented by learnable mask tokens at the positions of masked patches. These tokens are fed to a decoder and The objective function is to predict the pixels of the original masked patches.

kens are passed to the encoder and the decoder depth is less significant than the encoder depth. Moreover, the authors point out that the optimal masking ratio is high, around 75%, which further reduces the cost. This high masking ratio is because redundancies make it too easy for the network to predict the masked patches. This is because a lot of redundancies are present in the images from the pretraining dataset, ImageNet [DDS+09], which is object-centric. The decoder is dropped after pretraining as the encoder is the one learning representations.

Similarly, SimMIM [XZC+22] proposes an encoder-decoder pipeline to reconstruct masked images but the encoder takes visible and masked tokens and the decoder only involves one prediction layer. This significantly increases the computational cost in comparison with MAE but it allows the design to be used by hierarchical transformers such as Swin [LLC+21] whereas MAE does. However, some works built upon MAE aim to solve this issue [HYZ+22, LWYY22, ZTH+22]. Hierarchical transformers are useful to reduce computational usage and replace the standard self-attention used in VTs but also to increase performance on local downstream tasks such as Object Detection. As for MAE, SimMIM finds that a high masking ratio is important and investigates several masking strategies to find that its optimal one is big block masking.

As for CL, MIM has been extended to MVM for approaches using tokenizers such as BEVT [WCW+22] and MaskFeat [WFX+22] but also end-to-end approaches such as MAEs video-based models [FFLH22, TSWW22]. Notably, MaskFeat replaces the tokenizer and directly predicts the handcrafted HOG features of the videos. It achieves comparable performance to using a tokenizer, relieving video representation learning to pretrain a tokenizer. [SCC+23] also found that reconstructing motion trajectory hand-crafted features helps to improve performance. As for images, videos require a masking strategy but offer an ad-

ditional dimension that allows for more configurations. Indeed tokens instead of representing a 2D square patch for images, represent a 3D parallelepiped from patches of one or multiple frames for videos. The masking strategies developed are generally uniform masking on all patches or uniform masking of blocks of parallelepiped or masking multiple spatiotemporal tubes in the video. For MAE-based approaches, the masking ratio that performs best for datasets having action-centric videos is very high, around 90% because of the high redundancies present in such videos. This considerably reduces the cost of pretraining.

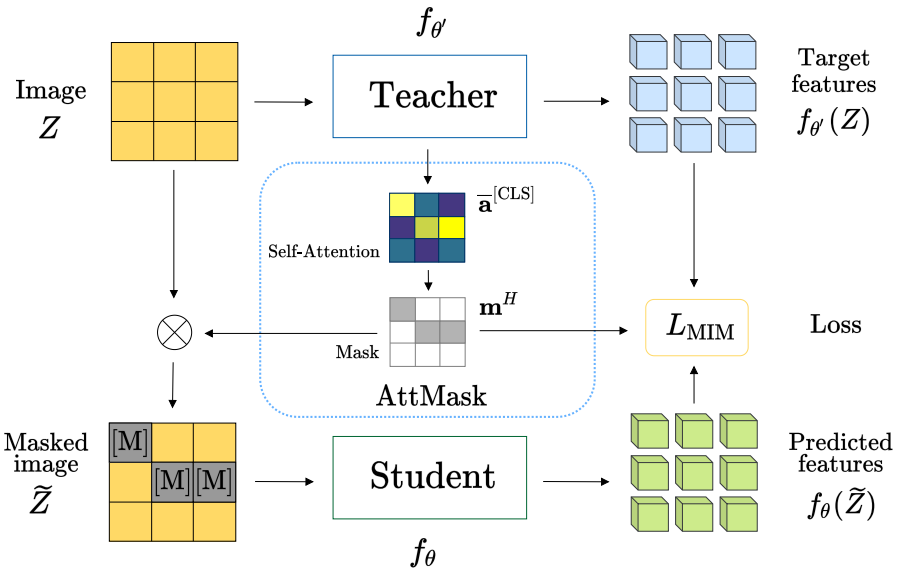


Figure 3.22: Illustration of high-attention teacher-guided masking from [KGP+22].

The question of how and what to mask is inherent to all MIM approaches [BDPW22, HCX+22, XZC+22]. Generally, the masking is random with different strategies such as uniform masking on all patches, or uniform masking on blocks of patches. However, patches in an image have different levels of semantics and some might contain more information than others. For example for a picture of a cat with a grass background, the level of understanding of the scene gained by looking at the cat tokens with more attention than the background tokens is important. Therefore, some methods [LCY+21, KGP+22] developed self-supervised masking strategies. The idea is to use student-teacher supervision of masking. The teacher takes as input the image and computes the attention map of the input tokens to produce the output representation. This attention map is used to mask the tokens according to their attention values. [LCY+21] masks the tokens that have low attention to avoid masking semantically important tokens whilst [KGP+22] illustrated in Fig. 3.22 built upon iBoT [ZWW+22b] introduced in the next subsection, finds that masking tokens with the highest attention improves results as it forces the encoder to exhibit high semantics understanding with short context.

As for CL, several approaches improved performance by using multi-modal inputs. It can be combining images and videos [GES+23, WCW+23] or images, depth and semantic [BMAZ22]...

3.5.2 Mixing Contrastive Learning and Masked Modelling

Masked Modeling approaches performed better than CL on downstream tasks [HCX+22, FFLH22] that require learning local features such as OD and TAL, but they fall short of marking a gap on downstream tasks that require a more global understanding of the scene. This led to methods that sought to combine CL and MM to learn representation good for global and local representations described in this section.

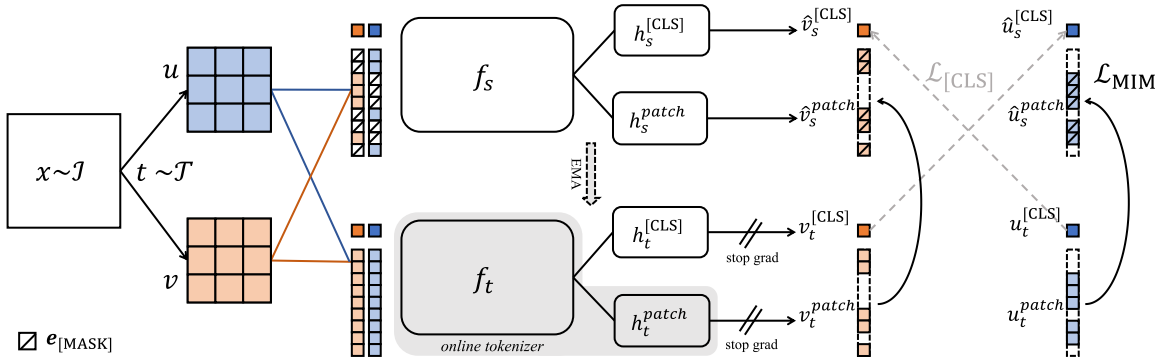


Figure 3.23: Illustration of iBoT from [ZWW+22b]. Some patches going through the student branches are masked to perform Masked features prediction of the teacher's outputs. A global contrastive loss is applied to the class tokens.

iBoT [ZWW+22b] illustrated in Fig. 3.23 combines DINO [CTM+21] and MM by masking some parts of the images going through the student branch. Instead of predicting a visual token or pixels, on top of the DINO global objective function between two positive views. Another objective related to mask modeling enforces the model to predict the token features outputted by the teacher for the respective output tokens from the student for the same view. Therefore, iBoT does not depend on a tokenizer per se, although the authors refer to the teacher as an online tokenizer. DINO-v2 [ODM+23] showed that SSL ViTs can be scaled by pushing objectives similar to iBoT to very large neural networks and datasets.

MSN [ACM+22] combines SwaV [CMM+20] and MM by masking part of the student input and does not associate a new loss to deal with MM. This allows that the masked tokens are not passed through the student encoder which reduces considerably the training time and memory cost. Although it showed great performance for low-shot image classification, it has lower results than iBoT for large-scale data. i-JEPA [ADM+23] proposes a variant of masked feature predictions where the student encoder seeks to predict from one view masked by several blocks, the features computed by the teacher encoder for each masked block via a $\mathscr{L}_2$ loss function as BYOL [GSA+20].

Similarly to iBoT, ConMIM [YGL+23] adopts a teacher-student siamese architecture with asymmetric data augmentations. The teacher receives strongly augmented images, e.g. with more perturbations, without masking and the student receives weakly augmented images that are partially masked. The objective function is a contrastive learning one and is applied on each output masked token from the student to retrieve its aligned target representation among the output target tokens for the same image. Whilst the number of negatives is fairly low, it is compensated by their hardness as they share semantics with the positives and ConMIM achieved better performance than standard Contrastive Learning and iBoT for local and global downstream tasks.

Methods	Tokenizer	Mask in	Prediction	Decoder	Contrast
BEiT, BEVT [BDPW22, WCW+22]	dVAE	✓	Visual tokens	✗	✗
MaskFeat [WFX+22]	✗	✓	HOG	One layer	✗
MAE, VideoMAE [HCX+22, TSWW22]	✗	✗	Pixels	Transformer	✗
SimMIM [XZC+22]	✗	✓	Pixels	One-layer	✗
iBoT, Dinov2 [ZWW+22b, ODM+23]	Teacher	✓	Features	✗	✓
MSN [ACM+22]	✗	✗	✗	✗	✓

Table 3.3: Overview of the main Masked Modeling approaches. For **Tokenizer** column, "Teacher" means updated via the exponential moving average of the neural network learned via backpropagation. **Mask in** column is checked if the methods pass the masked patch tokens in the encoder. If not, the methods considerably reduce the computational cost of an iteration. **Prediction** column refers to the modality to predict for the MM task. **Decoder** column describes the decoder architecture if relevant. If the **Contrast** column is checked, then a contrastive learning objective is also applied which requires a siamese network and data augmentations to create positive pairs.

Although these approaches improve performance over optimizing only Masked Modelling or only Contrastive Learning for general and local representation-dependent downstream tasks, these methods have the disadvantages of Contrastive Learning, meaning designing data augmentations to form positive pairs as well as a Siamese architecture that increases computational cost. In Tab. 3.3 we summarize the main MM approaches to get a grasp of the various methods and their advantages or disadvantages.

3.6 Toward our contributions

This chapter highlighted the various approaches to perform image and video self-supervised learning and the challenges they face. Our works mainly took place in the context of contrastive self-supervised learning and more specifically with approaches involving negatives. As we discussed in the dedicated section, one of its biggest challenges is to deal with hard and false negatives. We proposed to develop a new contrastive learning paradigm that is referred to as soft contrastive learning. It generalizes standard contrastive learning, which can be referred to as hard contrastive learning, and relational learning. Instead of pushing and pulling instances based solely on positiveness, it estimates relations between instances and pulls instances based on the strength of these

relations as well as positives. We introduced a practical implementation of this soft contrastive learning paradigm called Similarity Contrastive Estimation (SCE) in Chapter 4. We applied it to image global representation learning evaluated on image classification, OD, low-shot settings and sports-field registration. We then extended SCE to video global representation in Chapter 5 learning mainly evaluated on action recognition. As discussed in this chapter, some applications rely on local features and our last contribution introduced in Chapter 6 called COMEDIAN proposes a self-supervised learning pipeline that combines SCE, knowledge distillation and temporal mask modeling to pretrain a hierarchical transformer architecture to output local spatiotemporal features enriched in a more global context and study its effectiveness to the action spotting task.

Chapter 4

Similarity Contrastive Estimation for Image Representation Learning

Sommaire

4.1 Introduction to Representation Learning

We discussed in Sec. 3.4.2, how contrastive learning approaches with negatives face several issues. Pairs of views from the same images are generated by carefully designed data augmentations [CKNH20, TSP+20]. Elements from the same pairs are called *positives* and their representations are pulled together to learn view-invariant features. Other images called *negatives* are considered as noise and their representations are pushed away from positives. The negatives are difficult to handle as some of them are semantically false negatives but increasing their number is required to achieve the best performance [vdOLV18] which enhances the probability of retrieving false ones.

Based on the weakness of contrastive learning using negatives, we introduce a self-supervised soft contrastive learning approach called Similarity Contrastive Estimation (SCE), which contrasts positive pairs with other instances and leverages the push of negatives using the inter-instance similarities. Our method computes relations defined as a sharpened similarity distribution between augmented views of a batch. Each view from the batch is paired with a differently augmented query. Our objective function will maintain for each query the relations and contrast its positive with other images. A memory buffer is maintained to produce a meaningful distribution. Experiments on several datasets show that our approach outperforms our contrastive and relational baselines MoCov2 [CFGH20] and ReSSL [ZYW+21].

Our contributions [DRO+23, MODP23] can be summarized as follows:

- We propose a self-supervised soft contrastive learning approach called Similarity Contrastive Estimation (SCE) that contrasts pairs of augmented images with other instances and maintains relations among instances.
- We demonstrate that our framework SCE outperforms on several benchmarks its baselines MoCov2 [CFGH20] and ReSSL [ZYW+21] for a shared architecture and can further be improved using more recent architectures with a larger batch size and a predictor.
- We show that our proposed SCE is competitive with the state of the art on the ImageNet linear evaluation protocol and generalizes to several downstream tasks.

4.2 Related Work

We summarize here the related work to our proposed SCE that has been thoroughly detailed in the contrastive learning section Sec. 3.4. As we performed a study using convolutional networks, we did not perform a comparison with Masked Modeling approaches described in Sec. 3.5 which rely on transformers that require supplementary computational resources.

Contrastive Learning. Contrastive learning is a learning paradigm whose most successful methods rely on instance discrimination with a *positive* pair of views from the same image contrasted with all other instances called *negatives*. Retrieving lots of negatives is necessary for contrastive learning [vdOLV18] and various strategies have been proposed such as maintaining a queue of representations [HFW+20, CFGH20] or large batches [CKNH20, CXH21]. Hard negatives are the most important to sample to improve performance, however, they are potentially harmful to the training because of the "class collision" problem [CFSM20, WWSY21, CRL+20]. Several samplers have been proposed to alleviate this problem such as debiasing negatives sampling [CRL+20], and selecting hard negatives [RCSJ21] or denoising [YLH+21, YLH+23]. Instead, we propose a soft contrastive loss that seeks to estimate relations between instances and consider all negatives equally.

Regularized Contrastive Learning and Relational Learning. Several works regularize contrastive learning by optimizing a contrastive objective along with an objective that considers the similarities among instances [WWSY21] or clusters of close representations [WWSY21, LZXH21, ZWY+21]. ReSSL [ZYW+21] introduces an explicit relational learning objective by maintaining consistency of pairwise similarities between strong and weak augmented views. The pairs of views are not directly aligned which harms the discriminative performance.

In our work, we optimize a contrastive learning objective using negatives that alleviate class collision by pulling related instances. We do not use a regularization term but directly optimize a soft contrastive learning objective that leverages the contrastive and relational aspects.

4.3 Methodology

In this section, we will introduce our baselines: MoCov2 [CFGH20] for the contrastive aspect and ReSSL [ZYW+21] for the relational aspect. We will then present our self-supervised soft contrastive learning approach called Similarity Contrastive Estimation (SCE). All these methods share the same architecture illustrated in Fig. 4.1. We provide the pseudo-code of our algorithm in Appendix A.2.

4.3.1 Contrastive and Relational Learning

Consider $\mathbf{x} = \{\mathbf{x}_k\}_{k\in\{1,\dots,\mathrm{N}\}}$ a batch of N images. Siamese momentum methods based on Contrastive and Relational learning, such as MoCo [HFW+20] and ReSSL [ZYW+21] respectively, produce two views of $\mathbf{x}$, $\mathbf{x}^1 = t^1(\mathbf{x})$ and $\mathbf{x}^2 = t^2(\mathbf{x})$, from two data augmentation distributions T^1 and T^2 with $t^1 \sim \mathrm{T}^1$ and $t^2 \sim \mathrm{T}^2$. For ReSSL, T^2 is a weak data augmentation distribution compared to T^1 to maintain relations. $\mathbf{x}^1$ passes through an online network f_s followed by a projector g_s to compute $\mathbf{z}^1 = g_s(f_s(\mathbf{x}^1))$. A parallel target branch

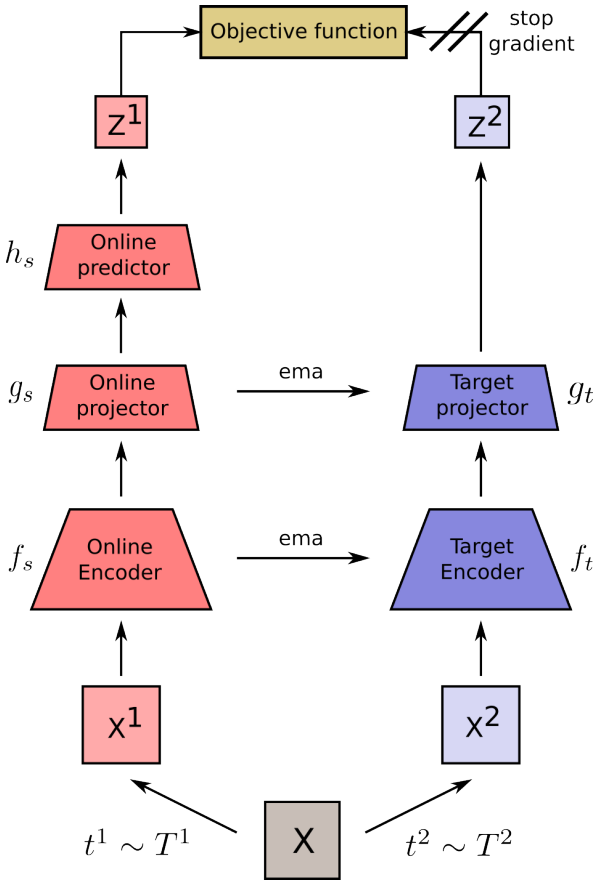


Figure 4.1: Illustration of the siamese pipeline shared by MoCov2 [CFGH20], ReSSL [ZYW+21] and SCE. A batch $\mathbf{x}$ of images is augmented with two different data augmentation distributions T^1 and T^2 to form $\mathbf{x}^1 = t^1(\mathbf{x})$ and $\mathbf{x}^2 = t^2(\mathbf{x})$ with $t^1 \sim \mathrm{T}^1$ and $t^2 \sim \mathrm{T}^2$. The representation $\mathbf{z}^1$ is computed through an online encoder f_s, projector g_s and optionally a predictor h_s such as $\mathbf{z}^1 = h_s(g_s(f_s(\mathbf{x}^1)))$. A parallel target branch updated by an exponential moving average of the online branch, or ema, computes $\mathbf{z}^2 = g_t(f_t(\mathbf{x}^2))$ with f_t and g_t the target encoder and projector. The representations are passed to the objective function that is different for each method.

containing a projector g_t and an encoder f_t computes $\mathbf{z}^2 = g_t(f_t(\mathbf{x}^2))$. $\mathbf{z}^1$ and $\mathbf{z}^2$ are both l_2-normalized.

The online branch parameters θ_s are updated by gradient (∇) descent to minimize a loss function $\mathscr{L}$. The target branch parameters θ_t are updated at each iteration by the exponential moving average of the online branch parameters with the *momentum value* m, also called *keep rate*, to control the update such as:

$$\theta_s \leftarrow optimizer(\theta_s, \nabla_{\theta_s}\mathscr{L}), \tag{4.1}$$

$$\theta_t \leftarrow m\theta_t + (1-m)\theta_s. \tag{4.2}$$

MoCo uses the InfoNCE loss, a similarity-based function scaled by the temperature τ that maximizes agreement between the positive pair and pushes negatives away:

$$\mathscr{L}_{InfoNCE} = -\frac{1}{N}\sum_{i=1}^{N}\log\left(\frac{\exp(\mathbf{z}_i^1\cdot\mathbf{z}_i^2/\tau)}{\sum_{j=1}^{N}\exp(\mathbf{z}_i^1\cdot\mathbf{z}_j^2/\tau)}\right). \tag{4.3}$$

ReSSL computes a target similarity distribution $\mathbf{s}^2$, that represents the relations between weak augmented instances, and the distribution of similarity $\mathbf{s}^1$ between the strongly augmented instances with the weak augmented ones. Temperature parameters are applied to each distribution: τ for $\mathbf{s}^1$ and τ_m for $\mathbf{s}^2$ with $\tau > \tau_m$ to eliminate noisy relations. Indeed, as the temperature decreases, it exponentially increases softmax values for highly similar instances and decreases exponentially values for low similar instances which makes them negligible in the target distribution. The loss function is the cross-entropy between $\mathbf{s}^2$ and $\mathbf{s}^1$:

$$s_{ik}^1 = \frac{\mathbb{1}_{i\neq k}\cdot\exp(\mathbf{z}_i^1\cdot\mathbf{z}_k^2/\tau)}{\sum_{j=1}^{N}\mathbb{1}_{i\neq j}\cdot\exp(\mathbf{z}_i^1\cdot\mathbf{z}_j^2/\tau)}, \tag{4.4}$$

$$s_{ik}^2 = \frac{\mathbb{1}_{i\neq k}\cdot\exp(\mathbf{z}_i^2\cdot\mathbf{z}_k^2/\tau_m)}{\sum_{j=1}^{N}\mathbb{1}_{i\neq j}\cdot\exp(\mathbf{z}_i^2\cdot\mathbf{z}_j^2/\tau_m)}, \tag{4.5}$$

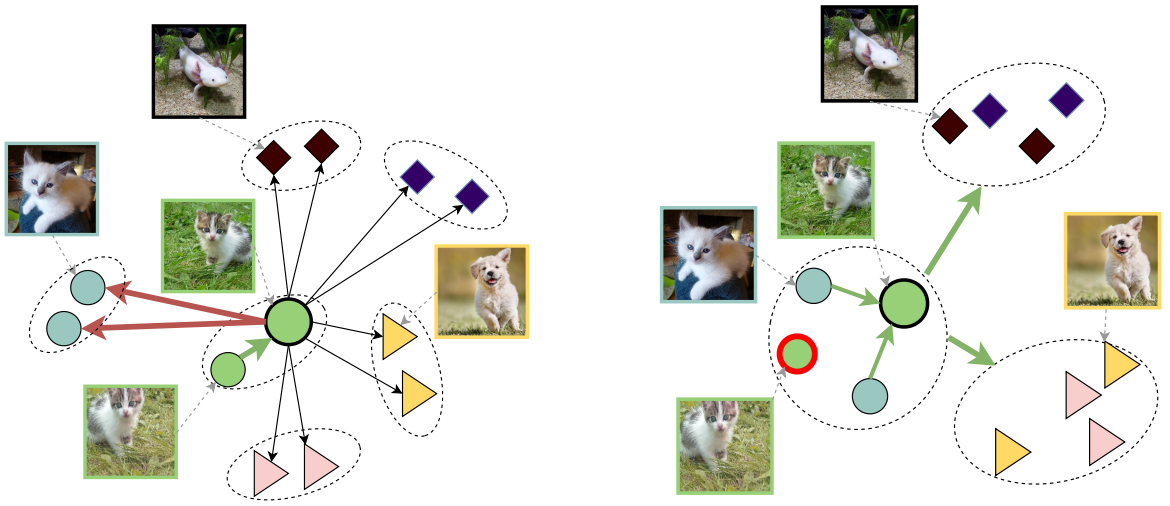
$$\mathscr{L}_{ReSSL} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{\substack{k=1\\k\neq i}}^{N} s_{ik}^2\log\left(s_{ik}^1\right). \tag{4.6}$$

A memory buffer of size M >> N filled by $\mathbf{z}^2$ is maintained for both methods and is used in place of negatives in previous equations.

4.3.2 Similarity Contrastive Estimation

Contrastive Learning methods damage relations among instances, which Relational Learning correctly builds. However, Relational Learning lacks the discriminating features that contrastive methods can learn. If we take the example of a dataset composed of cats and other animals, we want our model to be able to understand that two different cats share the same appearance but we also want our model to learn to distinguish details specific to each cat. We illustrated in Fig. 4.2 the issues of Contrastive and Relational Learning. Based on these requirements, we propose our approach called Similarity Contrastive Estimation (SCE).

We argue that there exists a true distribution of similarity $\mathbf{w}_i^*$ between a query $\mathbf{q}_i$ and the instances in a batch of N images $\mathbf{x} = \{\mathbf{x}_k\}_{k\in\{1,\dots,N\}}$, with $\mathbf{x}_i$ a positive view of $\mathbf{q}_i$. If we had access to $\mathbf{w}_i^*$, our training framework would estimate the similarity distribution $\mathbf{p}_i$ between $\mathbf{q}_i$ and all instances in $\mathbf{x}$, and minimize the cross-entropy between $\mathbf{w}_i^*$ and $\mathbf{p}_i$ which is a soft contrastive learning objective:



(a) Contrastive Learning

(b) Relational Learning

Figure 4.2: Illustration of (a) Contrastive Learning and (b) Relational Learning. Contrastive Learning correctly pushes negatives away and learns positive discriminative features but lacks modeling relations among instances. Relational Learning correctly models relations among instances but lacks discriminative positive features.

$$\mathscr{L}_{\text{SCE}^*} = -\frac{1}{\text{N}} \sum_{i=1}^{\text{N}} \sum_{k=1}^{\text{N}} w_{ik}^* \log\left(p_{ik}\right). \tag{4.7}$$

$\mathscr{L}_{\text{SCE}^*}$ is a soft contrastive approach, illustrated in Fig. 4.3 that generalizes InfoNCE and ReSSL objectives. InfoNCE is a hard contrastive loss that estimates $\mathbf{w}_i^*$ with a one-hot label and ReSSL estimates $\mathbf{w}_i^*$ without the contrastive component.

We propose an estimation of $\mathbf{w}_i^*$ based on contrastive and relational learning. We consider $\mathbf{x}^1 = t^1(\mathbf{x})$ and $\mathbf{x}^2 = t^2(\mathbf{x})$ generated from $\mathbf{x}$ using two data augmentations $t^1 \sim \text{T}^1$ and $t^2 \sim \text{T}^2$. Both augmentation distributions should be different to estimate different relations for each view as shown in Sec. 4.4.1. We compute $\mathbf{z}^1 = h_s(g_s(f_s(\mathbf{x}^1)))$ from the online encoder f_s, projector g_s and optionally a predictor h_s [GSA+20, CXH21]. We also compute $\mathbf{z}^2 = g_t(f_t(\mathbf{x}^2))$ from the target encoder f_t and projector g_t. $\mathbf{z}^1$ and $\mathbf{z}^2$ are both l_2-normalized.

The similarity distribution $\mathbf{s}_i^2$ that defines relations between the query and other instances is computed via the Eq. (4.5). The temperature τ_m sharpens the distribution to only keep relevant relations. A weighted positive one-hot label is added to $\mathbf{s}_i^2$ to build the target similarity distribution $\mathbf{w}_i^2$:

$$\text{w}_{ik}^2 = \lambda \cdot \mathbb{1}_{i=k} + (1-\lambda) \cdot \text{s}_{ik}^2. \tag{4.8}$$

The online similarity distribution $\mathbf{p}_i^1$ between $\mathbf{z}_i^1$ and $\mathbf{z}^2$, including the target positive representation in opposition with ReSSL, is computed and scaled by the temperature τ with $\tau > \tau_m$ to build a sharper target distribution:

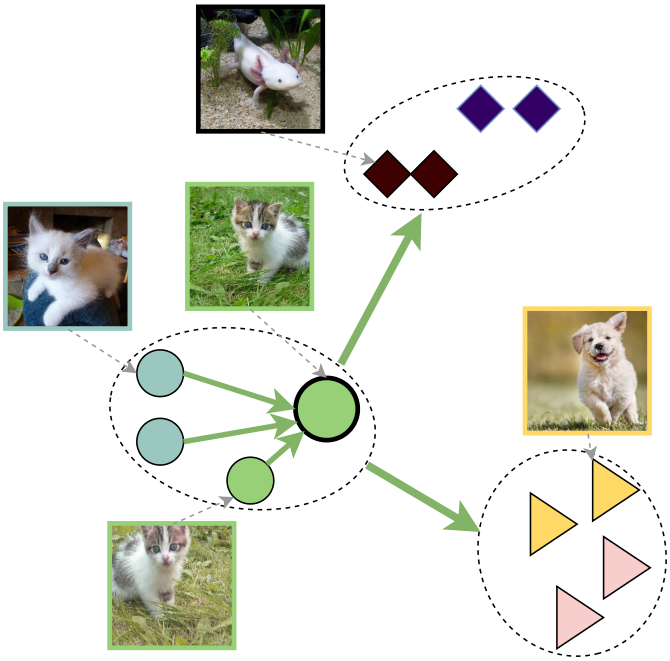


Figure 4.3: Illustration of Similarity Contrastive Estimation (SCE). SCE correctly aligns positives to learn discriminative features and models relations among instances.

$$\mathrm{p}^1_{ik} = \frac{\exp(\mathbf{z}^1_i \cdot \mathbf{z}^2_k / \tau)}{\sum_{j=1}^{\mathrm{N}} \exp(\mathbf{z}^1_i \cdot \mathbf{z}^2_j / \tau)}. \tag{4.9}$$

The objective function illustrated in Fig. 4.4 is the cross-entropy between each $\mathbf{w}^2$ and $\mathbf{p}^1$:

$$\mathscr{L}_{\mathrm{SCE}} = -\frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \sum_{k=1}^{\mathrm{N}} \mathrm{w}^2_{ik} \log\left(\mathrm{p}^1_{ik}\right). \tag{4.10}$$

The loss can be symmetrized by passing $\mathbf{x}^1$ and $\mathbf{x}^2$ through the momentum and online encoders and averaging the two losses computed.

A memory buffer of size M >> N filled by $\mathbf{z}^2$ is maintained to better approximate the similarity distributions and is used in place of negatives in previous equations.

The following proposition explicitly shows that SCE optimizes a contrastive learning objective while maintaining inter-instance relations:

Proposition 1. $\mathscr{L}_{\mathrm{SCE}}$ *defined in Eq.* (4.10) *can be written as:*

$$\mathscr{L}_{\mathrm{SCE}} = \lambda \cdot \mathscr{L}_{InfoNCE} + \mu \cdot \mathscr{L}_{\mathrm{ReSSL}} + \eta \cdot \mathscr{L}_{Ceil}, \tag{4.11}$$

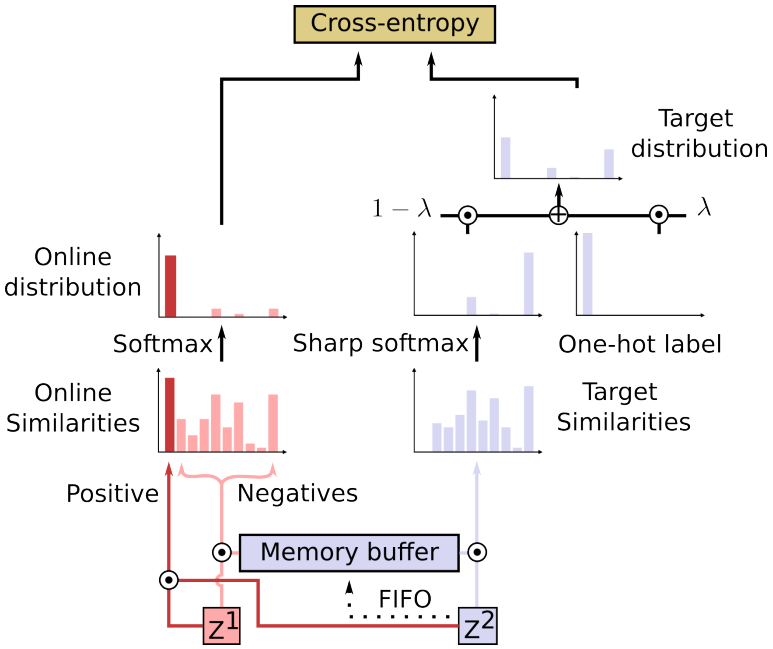


Figure 4.4: Objective function of SCE. The representations from the target branch $\mathbf{z}^2$ are used to compute the inter-instance target distribution by applying a sharp softmax to the cosine similarities between $\mathbf{z}^2$ and a memory buffer of representations from the momentum branch. This distribution is mixed via a $1-\lambda$ factor with a one-hot label factor λ to form the target distribution. Online similarities between $\mathbf{z}^1$ and the memory buffer plus its positive in $\mathbf{z}^2$ are also computed. The online distribution is computed via softmax applied to the online similarities. The objective function is the cross entropy between the target and the online distributions.

with $\mu = \eta = 1-\lambda$ *and*

$$\mathscr{L}_{Ceil} = -\frac{1}{N}\sum_{i=1}^{N}\log\left(\frac{\sum_{j=1}^{N}\mathbb{1}_{i\neq j}\cdot\exp(\mathbf{z}_i^1\cdot\mathbf{z}_j^2/\tau)}{\sum_{j=1}^{N}\exp(\mathbf{z}_i^1\cdot\mathbf{z}_j^2/\tau)}\right). \tag{4.12}$$

The proof separates the positive term and the negatives. It can be found in Appendix A.3. $\mathscr{L}_{Ceil}$ leverages how similar the positives should be with hard negatives. Because our approach is a soft contrastive learning objective, we optimize the formulation in Eq. (4.10) and have the constraint $\mu = \eta = 1-\lambda$. It frees our implementation from having three losses to optimize with two hyperparameters μ and η to tune. Still, we performed a small study of the objective defined in Eq. (4.11) without this constraint to check if $\mathscr{L}_{Ceil}$ improves results in Sec. 4.4.1.

It is also possible to show that SCE optimizes the DCL [YHH+22] defined by:

$$\mathscr{L}_{\text{DCL}} = -\frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{z}_i^1\cdot\mathbf{z}_i^2/\tau - \log\left(\sum_{j=1}^{N}\mathbb{1}_{i\neq j}\exp(\mathbf{z}_i^1\cdot\mathbf{z}_j^2/\tau)\right)\right). \tag{4.13}$$

and ReSSL losses [ZYW+21] as well as the Ceil loss defined in Eq. (4.12).

Proposition 2. $\mathscr{L}_{\text{SCE}}$ *defined in Eq.* (4.10) *can be written as:*

$$\mathscr{L}_{\text{SCE}} = \lambda \cdot \mathscr{L}_{\text{DCL}} + (1-\lambda) \cdot \mathscr{L}_{\text{ReSSL}} + \mathscr{L}_{Ceil}. \tag{4.14}$$

The proof available in Appendix A.4 highlights that $\mathscr{L}_{\text{DCL}} + \mathscr{L}_{Ceil} = \mathscr{L}_{InfoNCE}$. In their papers, DCL authors [YHH+22] empirically prove the superiority of DCL over InfoNCE but in our case as shown in Sec. 4.4.1 we obtain best performances when having $\mathscr{L}_{\text{DCL}}$ and $\mathscr{L}_{Ceil}$ with $\mathscr{L}_{\text{ReSSL}}$. We believe that the three losses play complementary roles required for the neural network to converge to a better data representation.

4.4 Empirical study

In this section, we first make an ablative study of our approach SCE to find the best hyper-parameters on images. Secondly, we compare SCE with its baselines MoCov2 [CFGH20] and ReSSL [ZYW+21] for the same architecture. Finally, we evaluate SCE on the ImageNet Linear evaluation protocol and assess its generalization capacity on various tasks.

4.4.1 Ablation study

To make the ablation study, we conducted experiments on ImageNet100 which has a close distribution to ImageNet, studied in Sec. 4.4.1, with the advantage of requiring fewer resources to train. We keep implementation details close to ReSSL [ZYW+21] and MoCov2 [CFGH20] to ensure a fair comparison.

Dataset. ImageNet [DDS+09] is a large dataset with 1k classes, almost 1.3M images in the training set and 50K images in the validation set. ImageNet100 is a selection of 100 classes from ImageNet whose classes have been selected randomly. We took the selected classes from [TKI20] referenced in Appendix A.1.

Implementation details for pretraining. We use the ResNet-50 [HZRS16] encoder and pretrain for 200 epochs. We apply by default *strong* and *weak* data augmentations defined in Tab. 4.1. We do not use a predictor and we do not symmetry the loss by default. Specific hyper-parameter details can be found in Appendix A.5.1.

Evaluation protocol. To evaluate our pretrained encoders, we train a linear classifier following [CFGH20, ZYW+21] that is detailed in Appendix A.5.1.

Leveraging contrastive and relational learning. SCE defined in Eq. (4.8) leverages contrastive and relational learning via the λ coefficient. We studied the effect of varying the λ coefficient on ImageNet100. Temperature parameters are set to $\tau = 0.1$ and $\tau_m = 0.05$. We report the results in Tab. 4.2. Performance increases with λ from 0 to 0.5 after which it starts decreasing. The best λ is inside $[0.4, 0.5]$ confirming that balancing the contrastive

Parameter	*weak*	*strong*	*strong*-α	*strong*-β	*strong*-γ
Random crop probability	1	1	1	1	1
Flip probability	0.5	0.5	0.5	0.5	0.5
Color jittering probability	0.	0.8	0.8	0.8	0.8
Brightness adjustment max intensity	-	0.4	0.4	0.4	0.4
Contrast adjustment max intensity	-	0.4	0.4	0.4	0.4
Saturation adjustment max intensity	-	0.4	0.2	0.2	0.2
Hue adjustment max intensity	-	0.1	0.1	0.1	0.1
Color dropping probability	0.	0.2	0.2	0.2	0.2
Gaussian blurring probability	0.	0.5	1.	0.1	0.5
Solarization probability	0.	0.	0.	0.2	0.2

Table 4.1: Different distributions of data augmentations applied to SCE. The *weak* distribution is the same as ReSSL [ZYW+21], *strong* is the standard contrastive data augmentation [CKNH20]. The *strong*-α and *strong*-β are two distributions introduced by BYOL [GSA+20]. Finally, *strong*-γ is a mix between *strong*-α and *strong*-β.

λ	0.	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Top-1	81.5	81.8	82.5	<u>82.8</u>	**82.9**	**82.9**	82.2	81.6	81.8	81.8	81.1

Table 4.2: Effect of varying λ on the Top-1 accuracy on ImageNet100. The optimal λ is in [0.4, 0.5] confirming that learning to discriminate and maintaining relations is best. Results style: **best**, <u>second best</u>.

and relational aspects provides better representation. In the next experiments, we keep $\lambda = 0.5$.

Method	Loss coefficients			Top-1	
	λ	μ	η	$\tau_m = 0.05$	$\tau_m = 0.07$
InfoNCE	1.	0.	0.	81.1	81.1
	0.5	0.5	0.	<u>82.8</u>	<u>82.5</u>
SCE	0.5	0.5	0.5	**82.9**	**83.4**
ReSSL	0.	1.	0.	80.8	78.4
	0.	1.	1.	81.5	79.6

Table 4.3: Effect of loss coefficients in Eq. (4.11) on the Top-1 accuracy on ImageNet100. $\mathscr{L}_{Ceil}$ consistently improves performance that varies given the temperature parameters. Results style: **best**, <u>second best</u>.

We performed a small study of the optimization of Eq. (4.11) by removing $\mathscr{L}_{ceil}$ ($\eta = 0$) to validate the relevance of our approach for $\tau = 0.1$ and $\tau_m \in \{0.05, 0.07\}$. The results are reported in Tab. 4.3. Adding the term $\mathscr{L}_{ceil}$ consistently improves performance, empirically proving that our approach is better than simply adding $\mathscr{L}_{InfoNCE}$ and $\mathscr{L}_{ReSSL}$. This performance boost varies with temperature parameters and our best setting improves by +0.9 percentage points (p.p.) in comparison with adding the two losses.

Asymmetric data augmentations to build the similarity distributions. Contrastive learning approaches use strong data augmentations [CKNH20] to learn view-invariant features and prevent the model from collapsing. However, these strong data augmen-

Online aug	Teacher aug	Sym	top-1
strong	*weak*	no	<u>82.9</u>
strong-γ	*weak*	no	**83.0**
weak	*strong*	no	73.4
strong	*strong*	no	80.5
strong-α	*strong*-β	no	80.7
strong	*weak*	yes	<u>83.7</u>
strong	*strong*	yes	83.0
strong-α	*strong*-β	yes	**84.2**

Table 4.4: Effect of using different distributions of data augmentations for the two views and of the loss symmetrization on the Top-1 accuracy on ImageNet100. Using a *weak* view for the teacher without symmetry is necessary to obtain good relations. With loss symmetry, asymmetric data augmentations improve the results, with the best obtained using strong-α and *strong*-β. Results style: **best**, <u>second best</u>.

tations shift the distribution of similarities among instances that SCE uses to approximate w_i^* in Eq. (4.8). We need to carefully tune the data augmentations to estimate a relevant target similarity distribution. We listed different distributions of data augmentations in Tab. 4.1. The *weak* and *strong* augmentations are the same as described by ReSSL [ZYW+21]. *strong*-α and *strong*-β have been proposed by BYOL [GSA+20]. *strong*-γ combines *strong*-α and *strong*-β.

We performed a study in Tab. 4.4 on which data augmentations are needed to build a proper target distribution for the non-symmetric and symmetric settings. We report the Top-1 accuracy on Imagenet100 when varying the data augmentations applied on the online and target branches of our pipeline. For the non-symmetric setting, SCE requires the target distribution to be built from a *weak* augmentation distribution that maintains consistency across instances.

Once the loss is symmetrized, asymmetry with strong data augmentations has better performance. Indeed, using *strong*-α and *strong*-β augmentations is better than using *weak* and *strong* augmentations, and the same *strong* augmentations have lower performance. We argue that symmetrized SCE requires asymmetric data augmentations to produce different relations for each view to make the model learn more information. The effect of using stronger augmentations is balanced by averaging the results on both views. Symmetrizing the loss boosts the performance as for [GSA+20, CH21].

Sharpening the similarity distributions. The temperature parameters sharpen the distributions of similarity exponentially. SCE uses the temperatures τ_m and τ for the target and online similarity distributions with $\tau_m < \tau$ to guide the online encoder with a sharper target distribution. We made a temperature search on ImageNet100 by varying τ in $\{0.1, 0.2\}$ and τ_m in $\{0.03, ..., 0.10\}$. The results are in Tab. 4.5. We found the best values $\tau_m = 0.07$ and $\tau = 0.1$ proving SCE needs a sharper target distribution. In Appendix A.6, this pa-

$\tau = 0.1$		$\tau = 0.2$	
τ_m	Top-1	τ_m	Top-1
0.03	82.3	0.03	**81.3**
0.04	82.5	0.04	<u>81.2</u>
0.05	<u>82.9</u>	0.05	<u>81.2</u>
0.06	82.5	0.06	<u>81.2</u>
0.07	**83.4**	0.07	81.1
0.08	82.7	0.08	80.9
0.09	82.5	0.09	<u>81.2</u>
0.10	82.1	0.10	<u>81.2</u>

Table 4.5: Effect of varying the temperature parameters τ_m and τ on the Top-1 accuracy on ImageNet100. τ_m is lower than τ to produce a sharper target distribution without noisy relations. SCE does not collapse when $\tau_m \to \tau$. Results style: **best**, <u>second best</u>.

rameter search is done for other datasets used in comparison with our baselines. Unlike ReSSL [ZYW+21], SCE does not collapse when $\tau_m \to \tau$ thanks to the contrastive aspect. Hence, it is less sensitive to the temperature choice.

4.4.2 Comparison with our baselines

We compared on 6 datasets how SCE performs against its baselines. We keep similar implementation details to ReSSL [ZYW+21] and MoCov2 [CFGH20] for a fair comparison.

Small datasets. Cifar10 and Cifar100 [KH09] have 50K training images, 10K test images, 32×32 resolution and 10-100 classes respectively. **Medium datasets.** STL10 [CNL11] has a 96×96 resolution, 10 classes, 100K unlabeled data, 5k labeled training images and 8K test images. Tiny-Imagenet [AR19] is a subset of ImageNet with 64×64 resolution, 200 classes, 100k training images and 10K validation images.

Implementation details. Architecture implementation details can be found in Appendix A.5.1. For MoCov2, we use $\tau = 0.2$ and for ReSSL their best τ and τ_m reported [ZYW+21]. For SCE, we use the best temperature parameters from Sec. 4.4.1 for ImageNet and ImageNet100 and from Appendix A.6 for the other datasets. The same architecture for all methods is used except for MoCov2 on ImageNet which kept the ImageNet100 projector to improve results.

Results are reported in Tab. 4.6. Our baselines reproduction is validated as results are better than those reported by the authors. SCE outperforms its baselines on all datasets proving that our method is more efficient to learn discriminating features on the pretrained dataset. We observe that our approach outperforms more significantly ReSSL on smaller datasets than ImageNet, suggesting that it is more important to learn to discriminate among instances for these datasets. SCE has promising applications to domains with few data such as in medical applications.

Method	ImageNet	ImageNet100	Cifar10	Cifar100	STL10	Tiny-IN
MoCov2 [CFGH20]	67.5	-	-	-	-	-
MoCov2 [*]	68.8	80.5	87.6	61.0	86.5	45.9
ReSSL [ZYW+21]	69.9	-	90.2	63.8	88.3	46.6
ReSSL [*]	70.2	81.6	90.2	64.0	89.1	49.5
SCE (Ours)	**70.5**	**83.4**	**90.3**	**65.5**	**89.9**	**51.9**

Table 4.6: Comparison of SCE with its baselines MoCov2 [CFGH20] and ReSSL [ZYW+21] on the Top-1 Accuracy on various datasets. SCE outperforms on all benchmarks its baselines. [*] denotes our reproduction. Results style: **best**, second best.

Method	100	200	300	800-1000
SimCLR [CKNH20]	66.5	68.3	-	70.4
MoCov2 [CH21]	67.4	69.9	-	72.2
SwaV [CMM+20]	66.5	69.1	-	71.8
BYOL [GSA+20]	66.5	70.6	72.5	74.3
Barlow-Twins[ZJM+21]	-	-	71.4	73.2
AdCo [HWHQ21]	-	68.6	-	72.8
ReSSL [ZYW+21]	-	71.4	-	-
WCL [ZWY+21]	68.1	70.3	-	72.2
VICReg [BPL22]	-	-	-	73.2
UniGrad [TWZ+22]	70.3	-	-	-
MoCov3 [CXH21]	68.9	-	72.8	74.6
NNCLR [DAT+21]	69.4	70.7	-	75.4
Triplet [WWW+21]	-	**73.8**	-	**75.9**
SCE (Ours)	**72.1**	72.7	**73.3**	74.1

Table 4.7: State-of-the-art results on the Top-1 Accuracy on ImageNet under the linear evaluation protocol at different pretraining epochs: 100, 200, 300, 800+. SCE is Top-1 at 100 epochs and Top-2 for 200 and 300 epochs. For 800+ epochs, SCE has lower performance than several state-of-the-art methods. Results style: **best**, second best.

4.4.3 ImageNet Linear Evaluation

We compare SCE on the widely used ImageNet linear evaluation protocol with the state of the art. We scaled our method using a larger batch size and a predictor to match state-of-the-art results [GSA+20, CXH21].

Implementation details. We use the ResNet-50 [HZRS16] encoder, apply *strong*-α and *strong*-β augmentations defined in Tab. 4.1. We follow the same training hyperparameters used by [CXH21] and detailed in Appendix A.5.2. The loss is symmetrized and we keep the best hyperparameters from Sec. 4.4.1: $\lambda = 0.5$, $\tau = 0.1$ and $\tau_m = 0.07$.

Multi-crop setting. We follow [HWHQ21] setting and sample 6 different views detailed in Appendix A.5.2.

Evaluation protocol. We follow the protocol defined by [CXH21] and detailed in Appendix A.5.2.

We evaluated SCE at epochs 100, 200, 300 and 1000 on the Top-1 accuracy on ImageNet to study the efficiency of our approach and compare it with the state of the art in Tab. 4.7. At 100 epochs, SCE reaches **72.1**% up to **74.1**% at 1000 epochs. Hence, SCE has a fast convergence and a few epochs of training already provide a good representation. SCE is the Top-1 method at 100 epochs and Top-2 for 200 and 300 epochs proving the good quality of its representation for a few epochs of pretraining.

At 1000 epochs, SCE is below several state-of-the-art results. We argue that SCE suffers from maintaining a λ coefficient to 0.5 and that relational or contrastive aspects do not have the same impact at the beginning and at the end of pretraining. A potential improvement would be using a scheduler on λ that varies over time.

Method	Epochs	Top-1
200 epochs		
SwaV [CMM+20]	200	72.7
AdCo [HWHQ21]	200	73.2
WCL [ZWY+21]	200	73.3
Triplet [WWW+21]	200	74.1
ReSSL [ZYW+21]	200	<u>74.7</u>
SCE (ours)	200	**75.4**
800+ epochs		
WCL [ZWY+21]	800	74.7
SwaV [CMM+20]	800	75.3
DINO [CTM+21]	800	75.3
UniGrad [TWZ+22]	800	75.5
NNCLR [DAT+21]	1000	<u>75.6</u>
AdCo [HWHQ21]	800	**75.7**

Table 4.8: State-of-the-art results on the Top-1 Accuracy on ImageNet under the linear evaluation protocol with multi-crop. SCE is competitive with the best state-of-the-art methods by pretraining for only 200 epochs instead of 800+. Results style: **best**, <u>second best</u>.

We added multi-crop to SCE for 200 epochs of pretraining. It enhances the results but it is costly in terms of time and memory. It improves the results from 72.7% to our best result **75.4**% (+**2.7p.p.**). Therefore, SCE learns from having local views and they should maintain relations to learn better representations. We compared SCE with state-of-the-art methods using multi-crop in Tab. 4.8. SCE is competitive with top state-of-the-art methods that trained for 800+ epochs by having slightly lower accuracy than the best method using multi-crop (−0.3p.p) and without multi-crop (−0.5p.p). SCE is more efficient than other methods, as it reaches state-of-the-art results for fewer pretraining epochs.

4.4.4 Transfer Learning

We study the generalization of our proposed SCE on several tasks using our multi-crop checkpoint pretrained for 200 epochs on ImageNet.

Method	K = 16	K = 32	K = 64	full
MoCov2 [CFGH20]	76.1	79.2	81.5	84.6
PCLv2 [LZXH21]	78.3	80.7	82.7	85.4
ReSSL [ZYW+21]	79.2	82.0	83.8	86.3
SwAV [CMM+20]	78.4	81.9	84.4	87.5
WCL [ZWY+21]	**80.2**	83.0	85.0	87.8
SCE (Ours)	79.5	**83.1**	**85.5**	**88.2**

Table 4.9: Transfer learning on low-shot image classification on Pascal VOC2007. All methods have been pretrained for 200 epochs. SCE is Top-1 when using 32-64-all images per class and Top-2 for 16 images. Results style: **best**, second best.

Method	Food	CIFAR10	CIFAR100	SUN	Cars	Air.	VOC	DTD	Pets	Caltech	Flow.	Avg.
SimCLR	72.8	90.5	74.4	60.6	49.3	49.8	81.4	75.7	84.6	89.3	92.6	74.6
Supervised	72.3	93.6	78.3	61.9	66.7	61.0	82.8	74.9	91.5	**94.5**	94.7	79.3
BYOL	75.3	91.3	78.4	62.2	**67.8**	60.6	82.5	75.5	90.4	94.2	**96.1**	79.5
NNCLR	76.7	93.7	79.0	62.5	67.1	**64.1**	83.0	75.5	**91.8**	91.3	95.1	80.0
SCE (Ours)	**77.7**	**94.8**	**80.4**	**65.3**	65.7	59.6	**84.0**	**77.1**	90.9	92.7	**96.1**	**80.4**

Table 4.10: Linear classifier trained on popular many-shot recognition datasets in comparison with SimCLR [CKNH20], supervised training, BYOL [GSA+20] and NNCLR [DAT+21]. SCE is Top-1 on 7 datasets and on average. Results style: **best**, second best.

Low-shot evaluation. Low-shot transferability of our backbone is evaluated on Pascal VOC2007. We followed the protocol proposed by [ZYW+21]. We select 16, 32, 64 or all images per class to train the classifier. Our results are compared with other state-of-the-art methods pretrained for 200 epochs in Tab. 4.9. SCE is Top-1 for 32, 64 and all images per class and Top-2 for 16 images per class, proving the generalization of our approach to few-shot learning.

Linear classifier for many-shot recognition datasets. We follow the same protocol as [GSA+20, EGH21] to study many-shot recognition in transfer learning on the datasets FGVC Aircraft [MRK+13], Caltech-101 [FFP07], Standford Cars [KSDF13], CIFAR-10 [KH09], CIFAR-100 [KH09], DTD [CMK+14], Oxford 102 Flowers [NZ08], Food-101 [BGG14], Oxford-IIIT Pets [PVZJ12], SUN397 [XHE+10] and Pascal VOC2007 [EGW+10]. These datasets cover a large variety of number of training images (2k-75k) and number of classes (10-397). We report the Top-1 classification accuracy except for Aircraft, Caltech-101, Pets and Flowers for which we report the mean per-class accuracy and the 11-point MAP for VOC2007.

We report the performance of SCE in comparison with state-of-the-art methods in Tab. 4.10. SCE outperforms on 7 datasets all approaches. On average, SCE is above all state-of-the-art methods as well as the supervised baseline, meaning SCE can generalize to a wide range of datasets.

Object detection and instance segmentation. We performed object detection and instance segmentation on the COCO dataset [LMB+14]. We used the pretrained network to initialize a Mask R-CNN [HGDG17] up to the C4 layer. We follow the protocol of

[WWW+21] and report the Average Precision for detection AP^{Box} and instance segmentation AP^{Mask}.

Method	AP^{Box}	AP^{Mask}
Random	35.6	31.4
Supervised	40.0	34.7
Rel-Loc [DGE15]	40.0	35.0
Rot-Pred [GSK18]	40.0	34.9
NPID [WXYL18]	39.4	34.5
MoCo [HFW+20]	40.9	35.5
MoCov2 [CFGH20]	40.9	35.5
SimCLR [CKNH20]	39.6	34.6
BYOL [GSA+20]	40.3	35.1
SCE (Ours)	<u>41.6</u>	<u>36.0</u>
Triplet [WWW+21]	**41.7**	**36.2**

Table 4.11: Object detection and Instance Segmentation on COCO [LMB+14] training a Mask R-CNN [HGDG17]. SCE is Top-2 on both tasks, slightly below Truncated-Triplet [WWW+21] and better than supervised training. Results style: **best**, <u>second best</u>.

We report our results in Tab. 4.11 and observe that SCE is the second best method after Truncated-Triplet [WWW+21] on both metrics, by being slightly below their reported results and above the supervised setting. Therefore our proposed SCE is able to generalize to object detection and instance segmentation task beyond what the supervised pretraining can (+**1.6p.p.** of AP^{Box} and +**1.3p.p.** of AP^{Mask}).

4.4.5 Sports-field registration

Sports-field registration is a task that consists of mapping image pixels of a sports field to their corresponding real-world location via generally a projection thanks to homographies. In this subsection, we pretrain a ViT Tiny encoder [DBK+21] to this task.

Pretraining dataset. We pretrained on the training split of SoccerNetv2 dataset [DCG+21] for action spotting which contains 300 football matches that last around 90 minutes each. The videos are the broadcast live version of the matches. The dataset is extracted at 2 Fps which creates about 3.3M images.

Sports-field registration dataset. We evaluate our approach on the TS-WorldCup dataset [CSH+22] that contains 43 video clips which corresponds to 3812 field images with 2925 images for training and 887 for testing.

Implementation details. Pretraining implementation details are available in Appendix A.5.3. For task-specific implementation details, we refer to the original paper [MODP23].

Metrics. We use four metrics to evaluate the performance of our approach:

- IoU$_{whole}$ is the intersection over union of the binary mask of the whole pitch transformed by the ground truth and the estimated homographies.
- IoU$_{part}$ is the intersection over union of the binary mask of the visible part of the pitch transformed by the ground truth and the estimated homographies,
- *projection error* is the average distance in meters between points randomly sampled in the frame on the visible part of the pitch and projected with the ground truth and the estimated homography,
- *re-projection error* is the average of the distances in pixels, normalized by the frame height, between points randomly sampled on the visible part of the pitch and re-projected with the ground truth and the estimated homography.

Encoder architecture	Encoder pretraining	IoU$_{\text{whole}}$ ↑ (%) mean	IoU$_{\text{whole}}$ ↑ (%) median	IoU$_{\text{part}}$ ↑ (%) mean	IoU$_{\text{part}}$ ↑ (%) median	proj. error ↓ (m.) mean	proj. error ↓ (m.) median	re-proj. error ↓ mean	re-proj. error ↓ median
ViT-tiny	SoccerNet	**95.7**	**96.2**	**98.3**	**98.5**	**0.26**	**0.23**	**0.008**	**0.006**
ViT-tiny	ImageNet	95.4	95.9	98.0	98.4	0.29	0.24	**0.008**	0.007

Table 4.12: Ablation studies on the TS-WorldCup dataset [CSH+22]. The SoccerNet pretraining stands for our self-supervised pretraining on the SoccerNet action spotting dataset [DCG+21] while the ImageNet pretraining stands for a supervised pretraining on the ImageNet 21k dataset [DDS+09]. The metrics show that the ViT-tiny encoder and the self-supervised pretraining improve the homography estimation performance.

We compare our pretrained backbone in 4.12 with a supervised learned one on ImageNet21k [DDS+09] that contains 14 million labeled diverse images. Therefore our domain-specific dataset contains fewer images but from a closer domain as our pretraining datasets contain soccer-related data. Results show that our SCE model consistently provides better-suited initialized weights than an ImageNet supervised on all metrics to evaluate sports-field registration.

4.5 Conclusion

In this chapter, we introduced a self-supervised soft contrastive learning approach called Similarity Contrastive Estimation (SCE). It contrasts pairs of asymmetrical augmented views with other instances while maintaining relations among instances. SCE leverages contrastive learning and relational learning and improves performance over optimizing only one aspect. We showed that it is competitive with the state of the art on the linear evaluation protocol on ImageNet, on video representation learning and to generalize to several image and video downstream tasks. We proposed a simple but effective initial estimation of the true distribution of similarity among instances. An interesting perspective would be to propose a finer estimation of this distribution.
This work has been published in:

- **Julien Denize**, Jaonary Rabarisoa, Astrid Orcesi, Romain Hérault, Stéphane Canu. "Similarity Contrastive Estimation for Self-Supervised Soft Contrastive Learning". In the *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2023. [DRO+23]
- *Adrien Maglot, *Astrid Orcesi, **Julien Denize**, Quoc-Cuong Pham. "Individual locating of soccer players from a single moving view". In *Sensors*, 2023. [MODP23]

Chapter 5

Similarity Contrastive Estimation for Global Video Representation Learning

Sommaire

5.1 Introduction

In Chapter 4, we introduced SCE for image representation SSL to improve contrastive learning and fasten its convergence. Video contrastive self-supervised learning faces similar challenges as images as discussed in Sec. 3.4.4. For videos pairs of views from the same videos are generated by sampling two clips from a video, with or without overlaps, on which are applied carefully designed data augmentations that are spatiotemporal [FFX+21, QMG+21]. Negatives are difficult to handle and require dedicated sampling and selection.

Based on the weaknesses of contrastive learning using negatives introduced in Sec. 3.4.2, we extend our self-supervised soft contrastive learning approach called Similarity Contrastive Estimation (SCE) to video global representation learning, that contrasts positive pairs with other instances and leverages the push of negatives using the inter-instance similarities. It shares the same underlying architecture as images but with changes applied to the data pipeline as well as the neural networks to handle videos.

Our contributions [DROH23] in this chapter can be summarized as follows:

- We extend the concept of relations and our proposed self-supervised soft contrastive learning approach SCE to video representation learning.
- We show that our proposed SCE reaches state-of-the-art results for video representation learning by pretraining on the Kinetics400 dataset as we beat or match previous top-1 accuracy for finetuning on HMDB51 and UCF101 for ResNet3D-18 and ResNet3D-50. We also demonstrate it generalizes to several video downstream tasks.

5.2 Related Work

Video Self-Supervised Learning follows the advances of Image Self-Supervised Learning and often picks ideas from the image modality with adjustment and improvement to make it relevant for videos and make the best use of it. We discussed the state of the art in Sec. 3.4.4 and summarized the related work here for reminder. As for images, we did not perform a thorough comparative study with Mask Modeling as these methods described in Sec. 3.5 rely on transformers and we worked with convolutional networks.

Contrastive Learning. Video Contrastive Learning has been widely studied in recent years as it gained interest after its better performance than standard pretext tasks in videos. Several works studied how to form positive views from different clips [HXZ20a, QMG+21, FFX+21, PSY+21] to directly apply contrastive methods from images. Some works focused on combining contrastive learning and predicting a pretext task [PAR20, WJL20, CHH+21, HSL+21, HWH+21, JJ21]. To help better represent the time

dimension, several approaches were designed to use different temporal context widths [PSY+21, RLA+21, DGRS22] for the different views.

Multi-modal Learning. To improve self-supervised representation learning, several approaches made use of several modalities to better capture the spatio-temporal information provided by a video. It can be from text [SBMS19, MAS+20], audio [AMK+20, PAR20, RLA+21], and optical flow [HXZ20a, LRO+20, HXZ20b, PAR20, HSL+21, RLA+21, TGSH22].

In our work, we extend our soft contrastive learning objective SCE introduced in Chapter 4 to global video representation learning using only RGB frames. It directly generalizes our approach from images with only changes related to data processing and architectures. To the best of our knowledge, we are the first to introduce the concept of soft contrastive learning using relations for video self-supervised representation learning.

5.3 Methodology

Our methodology is the same as for images explained in Sec. 4.3 with changes related to the modality difference. Therefore they concern the data pipeline, e.g. how views are formed, and what kind of backbone that outputs the global representation is used, and the changes are detailed in Sec. 5.4. Because the ablation study for images revealed that the best setting is to use an asymmetric architecture with a predictor for the online branch and a different set of data augmentations for the branches, we keep this setting for videos.

More formally, consider $\mathbf{x} = \{\mathbf{x}_k\}_{k\in\{1,\dots,\mathrm{N}\}}$ a batch of N videos. Two views are created by sampling two clips per video of T frames. The sampling of clips follows a strategy that can be two views with or without overlap [FFX+21, QMG+21]. As for images, we consider $\mathbf{x}^1 = t^1(\mathbf{x})$ and $\mathbf{x}^2 = t^2(\mathbf{x})$ generated from $\mathbf{x}$ using two data augmentations $t^1 \sim \mathrm{T}^1$ and $t^2 \sim \mathrm{T}^2$. Then, the online and target branches compute the $\mathbf{z}^1 = h_s(g_s(f_s(\mathbf{x}^1)))$ and $\mathbf{z}^2 = g_t(f_t(\mathbf{x}^2))$ representations that are l_2-normalized. Because we handle videos the online encoder f_s, and target encoder f_t are designed to deal with such modalities to learn spatiotemporal features. We estimate the target distribution $\mathbf{w}^2$ defined in Eq. (4.8) and the online distribution defined in Eq. (4.9) and compute the $\mathscr{L}_{\mathrm{SCE}}$ loss function defined in Eq. (4.10).

The loss is symmetrized by passing $\mathbf{x}^1$ and $\mathbf{x}^2$ through the momentum and online encoders respectively and averaging the two losses computed. A memory buffer of size M >> N filled by $\mathbf{z}^2$ is maintained to better approximate the similarity distributions.

5.4 Empirical study

In this section, we first make an ablation study of our approach SCE to find the best hyperparameters on videos. Then, we compare SCE to the state of the art after pretraining on Kinetics400 and assess generalization on various tasks.

5.4.1 Ablation study

Pretraining Dataset. To make the ablation study, we perform pretraining experiments on Mini-Kinetics200 [XSH+18], later called Kinetics200 for simplicity. It is a subset of Kinetics400 [KCS+17] meaning they have a close distribution with fewer resources required on Kinetics200 to train. Kinetics400 is composed of 216k videos for training and 18k for validation for 400 action classes. However, it has been created from YouTube and some videos have been deleted. We use the dataset hosted[1] by the CVD foundation. Kinetics200 is then formed from Kinetics400 and contains 200 classes with a selected of 400 videos per class for train and 25 for validation.

Evaluation Datasets. To study the quality of our pretrained representation, we perform linear evaluation classification on the Kinetics200 dataset. Also, we finetune on the first split of the UCF101 [SZS12] and HMDB51 [KJG+11] datasets. UCF101 is an action classification dataset that contains 13,300 different videos for 101 classes and has 3 different training and validation splits. HMDB51 is also an action classification dataset that contains 6,700 different videos from 51 classes with 3 different splits.

Pretraining implementation details. We used the ResNet3D-18 network [HKS18] following the Slow path of [FFMH19]. We kept hyperparameters close to the ones used for ImageNet in Sec. 4.4.3. More details can be found in Appendix B.1. We pretrain for 200 epochs with a batch size of 512. The loss is symmetrized. To form two different views from a video, we follow [FFX+21] and randomly sample two clips from the video that last 2.56 seconds. Then, we uniformly keep 8 frames from these clips.

Linear evaluation and finetuning evaluation protocols. We follow [FFX+21] and details can be found in Appendix B.1. For finetuning on UCF101 and HMDB51 we only use the first split in the ablation study.

Method	K200	UCF101	HMDB51
SCE Baseline	63.9	86.3	57.0
Supervised	**72.0**	**87.5**	**60.1**

Table 5.1: Comparison of our baseline and supervised training on the Kinetics200, UCF101 and HMDB51 Top-1 accuracy. Supervised training is consistently better.

[1]Link to the Kinetics400 dataset hosted by the CVD foundation: `https://github.com/cvdfoundation/kinetics-dataset`.

λ	K200	UCF101	HMDB51
0.000	64.2	86.2	<u>57.5</u>
0.125	**64.8**	**86.9**	**58.2**
0.250	64.3	<u>86.7</u>	**58.2**
0.375	<u>64.7</u>	86.3	56.8
0.500	63.9	86.3	57.0
0.625	63.4	86.2	55.7
0.750	63.1	85.8	56.2
0.875	62.1	85.7	55.3
1.000	61.9	85.0	55.4

Table 5.2: Effect of varying λ on the Kinetics200, UCF101 and HMDB51 Top-1 accuracy. The best λ is 0.125 meaning contrastive and relational leverage increases performance. Results style: **best**, <u>second best</u>.

τ_m	K200	UCF101	HMDB51
0.03	63.4	86.1	56.9
0.04	63.8	**86.6**	56.6
0.05	**64.3**	<u>86.4</u>	**57.1**
0.06	<u>64.1</u>	86.2	56.4
0.07	63.9	86.3	<u>57.0</u>
0.08	63.8	85.9	55.8

Table 5.3: Effect of varying τ_m on the Top-1 accuracy on Kinetics200, UCF101 and HMDB51 while maintaining $\tau = 0.1$. The best τ_m is 0.05 meaning that a sharper target distribution is required. Results style: **best**, <u>second best</u>.

Baseline and supervised learning. We define an SCE baseline that uses the hyperparameters $\lambda = 0.5$, $\tau = 0.1$, $\tau_m = 0.07$. We provide the performance of our SCE baseline as well as supervised training in Tab. 5.1. We observe that our baseline has lower results than supervised learning with -8.1p.p for Kinetics200, -1.2p.p for UCF101 and -3.1p.p for HMDB51 which shows that our representation has a large margin for improvement.

Leveraging contrastive and relational learning. As for the image study, we varied λ from the equation Eq. (4.8) in the set $\{0, 0.125, ..., 0.875, 1\}$ to observe the effect of leveraging the relational and contrastive aspects and report results in Sec. 5.4.1. Using relations during pretraining improves the results rather than only optimizing a contrastive learning objective. The performance on Kinetics200, UCF101 and HMDB51 consistently increases by decreasing λ from 1 to 0.25. The best λ obtained is 0.125. Moreover $\lambda = 0$ performs better than $\lambda = 1$. These results suggest that for video pretraining with standard image contrastive learning augmentations, relational learning performs better than contrastive learning and leveraging both further improves the quality of the representation.

Target temperature variation. We studied the effect of varying the target temperature with values in the set $\tau_m \in \{0.03, 0.04, ..., 0.08\}$ while maintaining the online temperature

Figure 5.1: Illustration of an RGB video and its corresponding Optical Flow and RGB difference for two videos from UCF101 [SZS12]. Optical Flow is predicted by the large RAFT architecture [TD20]. For each video, row 1 is RGB, row 2 is Optical Flow and row 3 is RGB difference. The Optical Flow color indicates motion direction and the intensity its strength. RGB difference does not contain the motion direction information but approximates the motion of Optical Flow.

$\tau = 0.1$. We report results in Sec. 5.4.1. We observe that the best temperature is $\tau_m = 0.05$ indicating that a sharper target distribution is required for video pretraining. We also observe that varying τ_m has a lower impact on performance than varying λ.

Spatial and temporal augmentations. We tested varying and adding some data augmentations to generate the pairs of views. As we are dealing with videos, these augmentations can be either spatial or temporal. We define the *jitter* augmentation that jitters by a factor of the duration of a clip, *reverse* that randomly reverses the order of frames and *diff* that randomly applies RGB difference on the frames. RGB difference consists of converting the frames to grayscale and subtracting them over time to approximate the magnitude of optical flow. We illustrate in Fig. 5.1 the difference between the RGB frames, OF, and RGB difference. In this work, we consider RGB difference as a data augmentation that is randomly applied during pretraining. In the literature it is often used as a modality to provide better representation quality than RGB frames [JT18, LRO+20, DZCL22]. Here, we

only apply it during pretraining as a random augmentation. Evaluation only sees RGB frames.

strength	K200	UCF101	HMDB51
0.50	63.9	86.3	57.0
0.75	<u>64.6</u>	<u>86.8</u>	<u>57.8</u>
1.00	**64.8**	**87.0**	**58.1**

Table 5.4: Effect of strength for color jittering for *strong*-α and *strong*-β augmentations on the Kinetics200, UCF101 and HMDB51 Top-1 accuracy. Strong color jittering improves performance. Results style: **best**, <u>second best</u>.

We tested to increase the color jittering strength in Tab. 5.4. Using a strength of 1.0 improved our performance on all the benchmarks suggesting that video pretraining requires harder spatial augmentations than images.

jitter	reverse	diff	K200	UCF101	HMDB51
0.0	0.0	0.0	63.9	86.3	57.0
0.2	0.0	0.0	64.2	86.4	56.9
0.0	0.2	0.0	64.0	85.7	55.4
0.0	0.0	0.2	<u>65.4</u>	**88.3**	**61.4**
0.0	0.0	0.5	64.1	<u>87.7</u>	<u>60.8</u>
Supervised			**72.0**	87.5	60.1

Table 5.5: Effect of using the temporal augmentations by applying clip duration jittering *jitter*, randomly reversing the order of frames *reverse* or randomly using RGB difference *diff* on the Kinetics200, UCF101 and HMDB51 Top-1 accuracy. The *diff* augmentation consistently improves results on the three benchmarks and outperforms supervised pretraining. The other augmentations unchanged or decreased performance on average. Results style: **best**, <u>second best</u>.

We tested our defined temporal augmentations with *jitter* of factor 0.2, meaning sampling clips between 0.80×2.56 and 1.20×2.56 seconds, randomly applying $reverse$ with 0.2 probability and randomly applying *diff* with 0.2 or 0.5 probability. We report results in Tab. 5.5. Varying the clip duration had no noticeable impact on our benchmarks, but reversing the order of frames decreased the performance on UCF101 and HMDB51. This can be explained by the fact that this augmentation can prevent the model from correctly representing the arrow of time. Finally, applying *diff* with 0.2 probability considerably improved our performance over our baseline with +**1.5p.p.** on Kinetics200, +**2.0p.p.** on UCF101 and +**4.4p.p.** on HMDB51. It outperforms supervised learning for generalization with +**0.8p.p.** on UCF101 and +**1.3p.p.** on HMDB51. Applying more often *diff* decreases performance. These results show that SCE benefits from using views that are more biased toward motion than appearance. We believe that it is particularly efficient to model relations based on motion.

λ	τ_m	diff	strength	K200	UCF101	HMDB51
0.125	0.05	0.2	1.0	65.0	87.4	<u>61.1</u>
0.125	0.07	0.2	1.0	64.7	88.2	60.6
0.500	0.05	0.2	1.0	<u>66.0</u>	<u>88.4</u>	**62.0**
0.500	0.07	0.2	1.0	65.4	**88.6**	61.0
SCE Baseline				63.9	86.3	57.0
Supervised				**72.0**	87.5	60.1

Table 5.6: Effect of combining best hyper-parameters found in the ablation study which are $\lambda = 0.125$, $\tau_m = 0.05$, *color strength*= 1.0 and adding randomly time difference on the Kinetics200, UCF101 and HMDB51 Top-1 accuracy. Using time difference and stronger color jittering increases the optimal λ value which indicates contrastive learning is efficient to deal with harder views and helps relational learning. The best value $\tau_m = 0.05$ performs favorably for Kinetics200 and HMDB51. Results style: **best**, <u>second best</u>.

Bringing all together. We studied varying one hyperparameter from our baseline and how it affects performance. In this final study, we combined our baseline with the different best hyperparameters found which are $\lambda = 0.125$, $\tau_m = 0.05$, color strength = 1.0 and applying *diff* with 0.2 probability. We report results in Tab. 5.6 and found out that using harder augmentations increased the optimal λ value as using $\lambda = 0.5$ performs better than $\lambda = 0.125$. This indicates that relational learning by itself cannot learn a better representation through positive views that share less mutual information. The contrastive aspect of our approach is proven efficient for such harder positives. We take as best configuration $\lambda = 0.5$, $\tau_m = 0.05$, *diff* applied with probability 0.2 and color strength = 1.0 as it provides best or second best results for all our benchmarks. It improves our baseline by +**2.1p.p.** on Kinetics200 and UCF101, and +**5.0p.p.** on HMDB51. It outperforms our supervised baseline by +**0.9p.p.** on UCF101 and +**1.9p.p.** on HMDB51.

5.4.2 Comparison with the State of the Art

Pretraining dataset. To compare SCE with the state of the art, we perform pretraining on Kinetics400 [KCS+17] introduced in Sec. 5.4.1.

Evaluation datasets. UCF101 [SZS12] and HMDB51 [KJG+11] have been introduced in Sec. 5.4.1.

AVA (v2.2) [GSR+18] is a dataset used for spatiotemporal localization of human actions composed of 211k training videos and 57k validation videos for 60 different classes. Bounding box annotations are used as targets and we report the mean Average Precision (mAP) for evaluation.

Something-Something V2 (SSv2) [GKM+17] is a dataset composed of human-object interactions for 174 different classes. It contains 169k training and 25k validation videos.

Pretraining implementation details. We use the ResNet3D-18 and ResNet3D-50 network [HKS18] and more specifically the slow path of [FFMH19]. We kept the best hyperparameters from Sec. 5.4.1 which are $\lambda = 0.5$, $\tau_m = 0.05$, RGB difference with a probability of 0.2, and color strength = 1.0 on top of the $strong-\alpha$ and $strong-\beta$ augmentations. From the randomly sampled clips, we specify if we keep 8 or 16 frames.

Action recognition. We compare SCE on the linear evaluation protocol on Kinetics400 and finetuning on UCF101 and HMDB51. We kept the same implementation details as in Sec. 5.4.1. We compare our results with the state of the art in Tab. 5.7 on various architectures. To propose a fair comparison, we indicate for each approach the pretraining dataset, the number of frames and the resolution used during pre-training as well as during evaluation. For the unknown parameters, we leave the cell empty. We compared with some approaches that used the other visual modalities Optical Flow and RGB difference and the different convolutional backbones S3D [ZDWW18] and R(2+1)D-18 [TWT$^+$18].

On ResNet3D-18 even when compared with methods using several modalities, by using 8×224^2 frames we obtain state-of-the-art results on the three benchmarks with **59.8**% accuracy on Kinetics400, **90.9**% on UCF101, **65.7**% on HMDB51. Using 16×112^2 frames, which is commonly used with this network, improved by +0.9p.p on HMDB51 and decreased by −3.2p.p on kinetics400 and −1.8 on UCF101 and keep state of the art results on all benchmarks, except on UCF101 with −0.5p.p compared with [DZCL22] using RGB and RGB difference modalities.

On ResNet3D-50, we obtain state-of-the-art results using 16×224^2 frames on HMDB51 with **74.7**% accuracy even when compared with methods using several modalities. On UCF101, with **95.3**% SCE is on par with the state of the art, −0.2p.p. than [FFX$^+$21], but on Kinetics400 −1.9p.p for **69.6**%. We have the same computational budget as they use 4 views for pretraining. Using 8 frames decreased performance by −2.0p.p., −1.2p.p. and −4.2p.p on Kinetics400,UCF101 and HMDB51. It maintains results that outperform the three benchmarks ρMoCo and ρBYOL with 2 views. It suggests that SCE is more efficient with fewer resources than these methods. By comparing our best with approaches on the S3D backbone that better fit smaller datasets, SCE has slightly lower performance than the state of the art: −1.0p.p. on UCF101 and −0.3p.p. on HMDB51.

Video retrieval. We performed video retrieval on our pretrained backbones on the first split of UCF101 and HMDB51. To perform this task, we extract from the training and testing splits the features using the 30-crop procedure for action recognition, detailed in Appendix B.1. We query for each video in the testing split the N nearest neighbors ($N \in \{1, 5, 10\}$) in the training split using cosine similarities. We report the recall R@N for the different N in Tab. 5.8.

We compare our results with the state of the art on ResNet3D-18. Our proposed SCE with 16×112^2 frames is Top-1 on UCF101 with **74.5**%, **85.6**% and **90.5**% for R@1, R@5 and

Method	T_p	Res_p	T_e	Res_e	Modality	Pretrain	K400	UCF101	HMDB51
Backbone: S3D / S3D-G									
SpeedNet [BEL+20]	64	-	16	224^2	R	K400	-	81.1	48.8
CoCLR [HXZ20b]	32	128^2	32	128^2	R	K400	-	87.9	54.6
CoCLR [HXZ20b]	32	128^2	32	128^2	R+F	K400	-	90.6	62.9
TEC [JJ21]	32	128^2	32	128^2	R	K400	-	86.9	63.5
ρBYOL (ρ = 4) [FFX+21]	32	224^2	32	256^2	R	K400	-	**96.3**	**75.0**
Backbone: R(2+1)D-18									
VideoMoCo [PSY+21]	32	112^2	-	-	R	K400	-	78.7	49.2
RSPNet [CHH+21]	16	112^2	16	224^2	R	K400	-	81.1	44.6
TransRank [DZCL22]	16	112^2	-	-	R	K200	-	87.8	60.1
TransRank [DZCL22]	16	112^2	-	-	R+RD	K200	-	90.7	64.2
TEC [JJ21]	16	112^2	16	112^2	R	K400	-	88.2	62.2
ρBYOL (ρ = 4) [FFX+21]	32	224^2	32	256^2	R	K400	-	**94.4**	**72.2**
Backbone: ResNet3D-18									
ST-Puzzle [KCK19]	16	-	16	112^2	R	K400	-	65.8	33.7
3D-RotNet [JT18]	16	112^2	-	-	R	K400	-	66.0	37.1
3D-RotNet [JT18]	16	112^2	-	-	R+D	K400	-	76.7	47.0
VTHCL [YXDZ20]	8	224^2	8	224^2	R	K400	-	80.6	48.6
TransRank [DZCL22]	16	112^2	-	-	R	K200	-	85.7	58.1
TransRank [DZCL22]	16	112^2	-	-	R+RD	UCF101	-	88.5	63.0
TransRank [DZCL22]	16	112^2	-	-	R+RD	K200	-	89.6	63.5
TEC [JJ21]	16	128^2	16	128^2	R	K400	-	87.1	63.6
ProViCo [PLKS22]	16	112^2	-	-	R	K400	-	87.2	59.4
ρMoCo (ρ = 2) [FFX+21]	8	224^2	8	256^2	R	K400	56.2	87.1	-
SCE (Ours)	8	224^2	8	256^2	R	K200	–	88.4	62.0
SCE (Ours)	16	112^2	16	128^2	R	K400	56.6	89.1	**66.6**
SCE (Ours)	8	224^2	8	256^2	R	K400	**59.8**	**90.9**	65.7
Backbone: ResNet3D-50									
VTHCL [YXDZ20]	8	224^2	8	224^2	R	K400	–	82.1	49.2
CATE [SNTS21]	8	224^2	32	256^2	R	K400	–	88.4	61.9
CVRL [QMG+21]	16	224^2	32	256^2	R	K400	66.1	92.2	66.7
CVRL [QMG+21]	16	224^2	32	256^2	R	K600	70.4	93.4	68.0
$CORP_f$ [HSL+21]	16	224^2	32	256^2	R+F	K400	66.6	93.5	68.0
ConST-CL [YQC+22]	16	224^2	32	256^2	R	K400	66.6	94.8	71.9
BraVe [RLA+21]	16	224^2	32	224^2	R	K400	-	93.7	72.0
BraVe [RLA+21]	16	224^2	32	224^2	R+F	K400	-	94.7	72.7
BraVe [RLA+21]	16	224^2	32	224^2	R	K600	-	94.1	74.0
BraVe [RLA+21]	16	224^2	32	224^2	R+F	K600	-	95.1	74.3
ρMoCo (ρ = 2) [FFX+21]	8	224^2	8	256^2	R	K400	65.8	91.0	-
ρMoCo (ρ = 2) [FFX+21]	16	224^2	16	256^2	R	K400	67.6	93.3	-
ρBYOL (ρ = 2) [FFX+21]	8	224^2	8	256^2	R	K400	65.8	92.7	-
ρBYOL (ρ = 4) [FFX+21]	8	224^2	8	256^2	R	K400	70.0	94.2	72.1
ρBYOL (ρ = 4) [FFX+21]	8	224^2	16	256^2	R	K400	**71.5**	**95.5**	73.6
SCE (Ours)	8	224^2	8	256^2	R	K400	67.6	94.1	70.5
SCE (Ours)	16	224^2	16	256^2	R	K400	69.6	95.3	**74.7**

Table 5.7: Performance of SCE for the linear evaluation protocol on Kinetics400 and finetuning on the three splits of UCF101 and HMDB51. $\mathbf{Res}_p$, $\mathbf{Res}_e$ means the resolution for pretraining and evaluation. $\mathbf{T}_p$, $\mathbf{T}_e$ means the number of frames used for pretraining and evaluation. For **Modality**, "R" means RGB, "F" means Optical Flow, "RD" means RGB difference. Best viewed in color, gray rows highlight multi-modal trainings and green rows our results. SCE obtains state of the art results on ResNet3D-18 and on the finetuning protocol for ResNet3D-50. Results style: **best**, second best.

Method	Res_p	T_p	Res_e	T_e	Pretrain	UCF101			HMDB51		
						R@1	R@5	R@10	R@1	R@5	R@10
Backbone: ResNet3D-18											
MemDPC [HXZ20a]	40	224^2	40	224^2	UCF101	20.2	40.4	52.4	7.7	25.7	40.6
RSPNet [CHH+21]	16	112^2	16	224^2	K400	41.1	59.4	68.4	-	-	-
MFO [QLL+21]	16	112^2	16	112^2	K400	41.5	60.6	71.2	20.7	40.8	55.2
TransRank [DZCL22]	16	112^2	-	-	UCF101	46.5	63.7	-	19.4	45.4	59.1
ViCC [TGSH22]	16	128^2	16	128^2	UCF101	50.3	70.9	78.7	22.7	46.2	60.9
TransRank [DZCL22]	16	112^2	-	-	K200	54.0	71.8	-	25.5	52.3	65.8
TCLR [DGRS22]	16	112^2	-	-	UCF101	56.2	72.2	79.0	22.8	45.4	57.8
TEC [JJ21]	16	128^2	16	128^2	UCF101	63.6	79.0	84.8	32.2	60.3	71.6
ProViCo [PLKS22]	16	112^2	-	-	UCF101	63.8	75.1	84.8	35.9	55.2	74.3
ProViCo [PLKS22]	16	112^2	-	-	K400	67.6	81.4	90.1	**40.1**	60.6	75.2
SCE (Ours)	16	112^2	16	128^2	K400	**74.5**	**85.9**	**90.5**	37.8	62.1	73.8
SCE (Ours)	8	224^2	8	256^2	K400	74.4	85.6	90.0	**40.1**	**63.3**	**75.4**
Backbone: ResNet3D-50											
CATE [SNTS21]	8	224^2	32	256^2	K400	54.9	68.3	75.1	33.0	56.8	69.4
SCE (Ours)	8	224^2	8	256^2	K400	81.5	89.7	92.8	43.0	67.0	79.0
SCE (Ours)	16	224^2	16	256^2	K400	**83.9**	**92.2**	**94.9**	**45.9**	**69.9**	**80.5**

Table 5.8: Performance of SCE for video retrieval on the first split of UCF101 and HMDB51. $\textbf{Res}_p$, $\textbf{Res}_e$ means the resolution for pretraining and evaluation. $\textbf{T}_p$, $\textbf{T}_e$ means the number of frames used for pretraining and evaluation. We report the recall R@1, R@5, R@10. We obtain state-of-the-art results for ResNet3D-18 on both benchmarks and further improve our results using the larger network ResNet3D-50. Results style: **best**, second best.

Method	views	T	*Linear protocol*	*Finetuning accuracy*		
			K400	UCF101	AVA (mAP)	SSv2
Supervised	1	8	**74.7**	94.8	22.2	52.8
ρSimCLR (ρ = 3)	3	8	62.0 (−12.7)	87.9 (−6.9)	17.6 (−4.6)	52.0 (−0.8)
ρSwAV (ρ = 3)	3	8	62.7 (−12.0)	89.4 (−5.4)	18.2 (−4.0)	51.7 (−1.1)
ρBYOL (ρ = 3)	3	8	68.3 (−6.4)	93.8 (−1.0)	**23.4** (+1.2)	55.8 (+3.0)
ρMoCo (ρ = 3)	3	8	67.3 (−7.4)	92.8 (−2.0)	20.3 (−1.9)	54.4 (+1.8)
SCE (Ours)	2	8	67.6 (−7.1)	94.1 (−0.7)	20.3 (−1.9)	53.9 (+1.1)
SCE (Ours)	2	16	69.6 (−5.1)	**95.5** (+0.7)	21.6 (−0.6)	**57.2** (+4.4)

Table 5.9: Performance of SCE in comparison with [FFX+21] for linear evaluation on Kinetics400 and finetuning on the first split of UCF101, AVA and SSv2. SCE is on par with ρMoCo for fewer views. Increasing the number of frames outperforms ρBYOL on Kinetics400, UCF101 and SSv2. Results style: **best**, second best.

R@10. Using 8×224^2 frames slightly decreases results that are still state of the art. On HMDB51, SCE with 8×224^2 frames outperforms the state of the art with **40.1**%, **63.3**% and **75.4**% for R@1, R@5 and R@10. Using 16×112^2 frames decreased results that are competitive with the previous state-of-the-art approach [PLKS22] for −2.3p.p., +1.5p.p. and −1.4p.p. on R@1, R@5 and R@10.

We provide results using the larger architecture ResNet3d-50 which increases our performance on both benchmarks and outperforms the state of the art on all metrics to reach **83.9**%, **92.2**% and **94.9**% for R@1, R@5 and R@10 on UCF101 as well as **45.9**%, **69.9**% and **80.5**% for R@1, R@5 and R@10 on HMDB51. Our soft contrastive learning approach makes our representation learn features that cluster similar instances even for generalization.

Generalization to downstream tasks. We follow the protocol introduced by [FFX+21] to compare the generalization of our ResNet3d-50 backbone on Kinetics400, UCF101, AVA and SSv2 with ρSimCLR, ρSwAV, ρBYOL, ρMoCo and supervised learning in Tab. 5.9. To ensure a fair comparison, we provide the number of views used by each method and the number of frames per view for pretraining and evaluation.

For 2 views and 8 frames, SCE is on par with ρMoCo with 3 views on Kinetics400, AVA and SSv2 but is worse than ρBYOL, especially on AVA. For UCF101, results are better than ρMoCo and on par with ρBYOL. These results indicate that our approach proves more effective than contrastive learning as it reaches similar results than ρMoCo using one less view. Using 16 frames, SCE outperforms all approaches, including supervised training, on UCF101 and SSv2 but performs worse on AVA than ρByol and supervised training. This study shows that SCE can generalize to various video downstream tasks which is a criterion of a good learned representation.

5.5 Conclusion

In this chapter, we extended our proposed soft contrastive self-supervised learning approach called Similarity Contrastive Estimation (SCE) to video global representation learning. We showed that it is competitive with the state of the art on the widely used Kinetics400 dataset and to generalize to several video downstream tasks. Experiments show that emphasizing the motion aspect via RGB difference is a key component to improve performance opening interesting perspectives to compute a target distribution with more weights applied to motion.
This work has been published in:

- **Julien Denize**, Jaonary Rabarisoa, Astrid Orcesi, Romain Hérault. "Similarity Contrastive Estimation for Image and Video Soft Contrastive Self-Supervised Learning". In *Machine Vision and Applications*, 2023. [DROH23]

Chapter 6

COMEDIAN for Local Temporal Representation Learning

Sommaire

6.1 Introduction

(**a**) Card.

(**b**) Substitution.

(**c**) Goal.

Figure 6.1: Illustration of action spotting from [NRSH+21]. The action is located in the middle frame for each row.

In Chapter 4 and Chapter 5, we introduced SCE for image and video global representation learning. However, there exist some video tasks that require multiple local temporal representations rather than a global one. This is the case for Temporal Action Detection (TAD) which involves the identification of when specific actions occur within a video, enabling comprehensive understanding and meaningful insights into the dynamics of a given scenario. Action Spotting [GADG18], illustrated in Fig. 6.1, is a specific TAD task whose goal is to predict actions at a precise timestamp and therefore requires a temporally precise prediction. Modeling actions in videos faces several issues such as the sparsity of actions and the intricate relationships between them.

Pretraining for such tasks is very interesting, especially via SSL as a lot of video data is readily available but annotating actions presents significant challenges. It suffers from the inherent subjectivity of annotators to interpret when an action starts or ends. Moreover, the process of manual annotation requires considerable time and resources, limiting the scalability of annotating large datasets.

In contrary to previous contributions in Chapter 4 and Chapter 5, this study lies on the recent advancements in vision Transformers [DBK+21, LLC+21] for video analysis

[ADH+21, LNC+22, SJE+23] that showed them surpassing the traditionally used Convolutional Neural Networks (CNNs) but required dedicated architecture design to reduce computational cost [NBZA21, ADH+21, BWT21, XXC+21, WLM+22, YDL+22, MKLF22]. By capturing long-range dependencies and leveraging global context, Transformers analyses better complex sequences. However, their effectiveness crucially depends on proper initialization and access to either ample labeled training data [ADH+21, LNC+22] or self-supervised learning [CTM+21].

In this work, we propose the COMEDIAN approach, which combines self-supervised learning and Knowledge Distillation to initialize a spatiotemporal transformer for action spotting. Knowledge Distillation (KD) [HVD15] initializes a network by transferring knowledge from another network or a collection of models. Depending on how the networks are obtained and the losses used to distill, KD can be considered as SL [TCD+21] or SSL [GZL+22]. Our method involves two transformers: a spatial transformer, which learns short context information from frames extracted from small videos, and a temporal transformer, which enriches the local context with global information. The initialization process consists of two stages: the first stage focuses on the spatial transformer via SSL, while the second stage initializes both spatial and temporal transformers via KD. KD is employed from a pre-computed bank of representations aligned with each output temporal token. Notably, COMEDIAN leverages unlabeled video data for initialization, effectively addressing the aforementioned challenges associated with transformers.

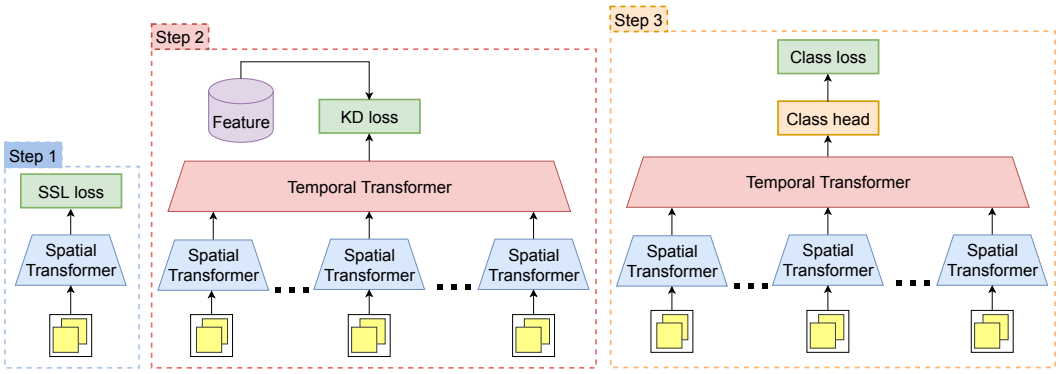


Figure 6.2: Overview of COMEDIAN training pipeline. Step 1: Pretraining of the spatial transformer. Step 2: Pretraining of the spatial and temporal transformers. Step 3: fine-tuning to the action spotting task.

To evaluate our approach, we conducted experiments on the action spotting task on the SoccerNet-v2 [DCG+21] dataset, which contains soccer matches with 17 distinct actions varying in semantics and occurrence. Our contributions [DLR+24] can be summarized as follows:

- We propose COMEDIAN a combined self-supervised learning and knowledge distillation pipeline illustrated in Fig. 6.2 to initialize transformers for action spotting.

- We demonstrate that COMEDIAN achieves state-of-the-art performance for action spotting on the SoccerNet-v2 dataset, showcasing the effectiveness of our self-supervised and knowledge distillation pipeline.
- We provide a comprehensive analysis of the benefits of pretraining with knowledge distillation, including improved performance and faster convergence compared to non-pretrained models.

6.2 Related Work

Video Transformers. Vision Transformers [DBK+21] (ViT) capture long-term dependencies better than recurrent models or convolutional networks. It relies on a tokenizer, that embeds patches of the input, and self-attention [VSP+17]. Standard self-attention is computationally heavy and several other attention mechanisms have been proposed such as Swin [LLC+21] or DeIT [TCD+21]. Transformers can be applied to videos by adapting the tokenizer [ADH+21, BWT21, LNC+22]. However as videos increase the number of tokens, video transformers are computationally heavy, and various strategies have been proposed to reduce their cost. VTN [NBZA21] adds a temporal encoder on top of a ViT while ViViT [ADH+21] and TimeSformer [BWT21] propose several factorizations of space-time attention. ViViT as VTN found a spatio-temporal hierarchical model offers the best trade-off between performance and cost which led to several methods [BPS+21, HHOK21]. Previously mentioned transformers focused on short videos, e.g. ≤ 5 seconds and some architectures have been developed to capture long-range dependencies on longer videos via a sliding window that keeps relevant information from the past with a memory [XXC+21, WLM+22] or a recurrence [YDL+22, MKLF22] mechanism.

In our work, we consider a spatiotemporal hierarchical model without changing specifically the architecture to capture long-term dependencies as we also want to capture bidirectional short-term dependencies.

Pretraining. Pretraining has been crucial to unleashing image [DBK+21] and video [ADH+21] transformers either via Supervised Learning (SL) on a large dataset [DBK+21] or Self-Supervised Learning (SSL) [CXH21, CTM+21]. We discussed in Sec. 3.4 and Sec. 3.5 contrastive learning [vdOLV18] and masked modeling to learn image and video representations. We mainly use in our pipeline contrastive learning methods as masked modeling approaches assume lots of redundancies are present in the video which is true for short videos with few view variations but does not hold for complex videos such as soccer matches. Knowledge distillation (KD) [HVD15] is another approach that distills information from teacher models to students and has been successfully applied for SL [TCD+21] as well as SSL [PKLC19, FWW+21, KTP20, GZL+22]. Most methods learn global representation without consideration for local-temporal representation even though multiple approaches emerged to learn spatial and temporal features separately or decoupled to learn

a global representation [HLW+21, QDLL22, ZWW+22a, ZWL22]. Closer to our approach, CARL [CWLC22] followed by [ZLZS23] proposes a sequence contrastive learning approach to learn frame-wise representations to evaluate fine-grained frame retrieval tasks.

In our work, we pretrain our hierarchical model with a contrastive SSL initialization of the spatial transformer. Then, our global model is pretrained with a KD loss from an extracted bank of features aligned with all the output tokens. This loss leverages temporal masking and soft contrastive learning to maintain local-temporal information enriched in a global context. Therefore, our goal is to learn multiple local-temporal representations, not one global. In opposition to CARL, our local-temporal representation does not concern a frame but a small temporal segment.

Action Spotting. Action Spotting is a timestamp-level Temporal Action Detection (TAD) first introduced for the dataset SoccerNet [GADG18]. This dataset has been extended to more videos and more actions in SoccerNet-v2 [DCG+21] and the tight-Average mean Average Precision (t-AmAP) has been introduced to evaluate precise detection within thresholds of 1 to 5 seconds. Several approaches have been proposed to tackle this task that can be divided into two categories. First, most approaches build a temporal architecture on top of a feature extractor [CDG+20, TBC+20, MHY+21, ZKC+21, MAR21, GG21, SMY+22, CCL+22, CYZ+22, DS22, SSB22], and second, few others train an end-to-end network [ZLL+22, HZG+22]. The first kind of approach reduces the computational cost of experiments however makes them rely on a feature extractor for generalization. Notably, Baidu [ZKC+21] proposed the extraction features of five 3D CNN models pretrained on Kinetics [KGP+22] and finetuned on SoccerNet-v2. The features are then plugged into a temporal action detector. Spivak [SSB22] used these Baidu features coupled with features from a pretrained ResNet-152 to train an anchor-based approach that first classifies actions falling in a few-second temporal radius and shifts the predictions using a temporal regressor. For end-to-end approaches, E2E-spot [HZG+22] proposed to train a CNN spatial encoder on top of a simple recurrent model that performed competitively with the previous approach. As for other domains, the attention mechanism has been studied to improve performance [MHY+21, ZKC+21, ZLL+22, SMY+22, SSB22].

In our work, we propose an end-to-end transformer-based action spotting approach that assigns actions that fall in frames in a small temporal radius.

6.3 Method

6.3.1 Overview

Our approach seeks to learn locally precise temporal features enriched with a larger context for action spotting. Therefore our model is composed of three different parts:

- A spatial transformer embeds the information of a small temporal window and outputs one token embedding.
- A temporal transformer that takes as input the token outputs of the spatial transformer on consecutive windows and outputs the same number of tokens. The temporal transformer enriches the representation of local windows with the knowledge of a larger context.
- A linear head is applied on each temporal output token to perform the classification of the action classes.

To train the model, we perform three different training steps described in the latter subsections: (1) pretraining of the spatial transformer on small windows in Sec. 6.3.2, (2) pretraining of the spatial and temporal transformers on large windows in Sec. 6.3.3, (3) fine-tuning of the model on the action spotting task in Sec. 6.3.4.

6.3.2 Spatial pretraining

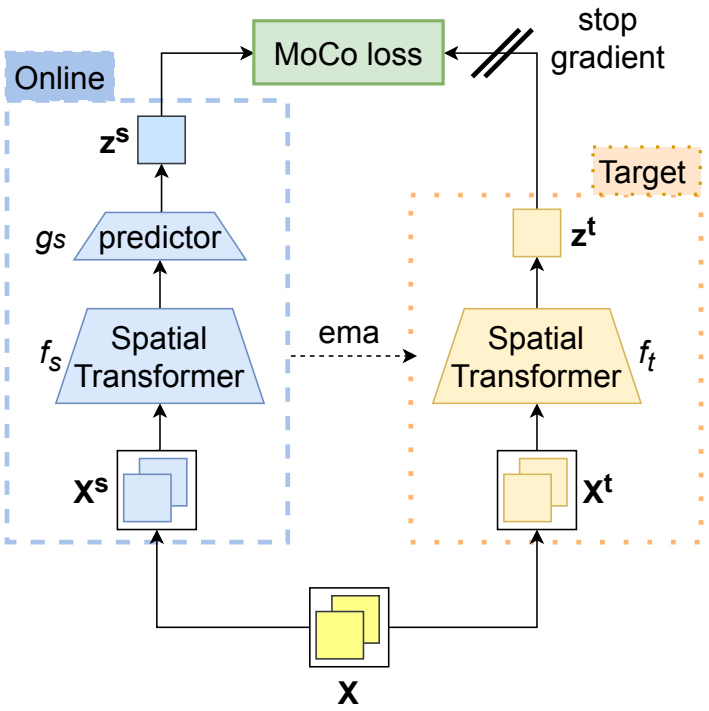


Figure 6.3: Pretraining of the spatial transformer.

To train the spatial transformer, we follow the Self-Supervised Contrastive method MoCo [HFW+20] illustrated in Fig. 6.3. We use a Siamese architecture containing an online and a target branch. For the online branch, the model contains a transformer f_s and a predictor g_s. The target branch contains a copy of the transformer updated by the exponential moving average, or ema, of the online transformer.

Each video $\mathbf{X} \in \mathbb{R}^{T_s \times H \times W \times C}$ of T_s frames, width W, height H, and C channels from the dataset is augmented by two different distributions of data augmentations A^1 and A^2 to

form positive views $\mathbf{X}^1 = a^1(\mathbf{X}), \mathbf{X}^2 = a^2(\mathbf{X})$ with $a^1 \sim \mathrm{A}^1$ and $a^2 \sim \mathrm{A}^2$. We pass both views in both transformers to compute the representations $\mathbf{z}^{1s} = g_s(f_s(\mathbf{X}^1))$, $\mathbf{z}^{2s} = g_s(f_s(\mathbf{X}^2))$, $\mathbf{z}^{1t} = f_t(\mathbf{X}^1)$ and $\mathbf{z}^{2t} = f_t(\mathbf{X}^2)$. A momentum memory buffer $\mathbf{Q}$ of size M >> N is maintained on the target representations to provide negatives.

We apply the MoCo loss on each representation as follows:

$$\mathscr{L}_{MoCo} = -\frac{1}{2}\left(l_{\mathrm{M}}(\mathbf{z}^{1s}, \mathbf{z}^{2t}) + l_{\mathrm{M}}(\mathbf{z}^{2s}, \mathbf{z}^{1t})\right), \tag{6.1}$$

$$l_{\mathrm{M}}(\mathbf{z}, \mathbf{k}) = log\left(\frac{exp(\mathbf{z}\cdot\mathbf{k}/\tau)}{exp(\mathbf{z}\cdot\mathbf{k}/\tau) + \sum_{j=1}^{n} exp(\mathbf{z}\cdot\mathbf{Q}_j/\tau)}\right). \tag{6.2}$$

6.3.3 Spatio-temporal pretraining

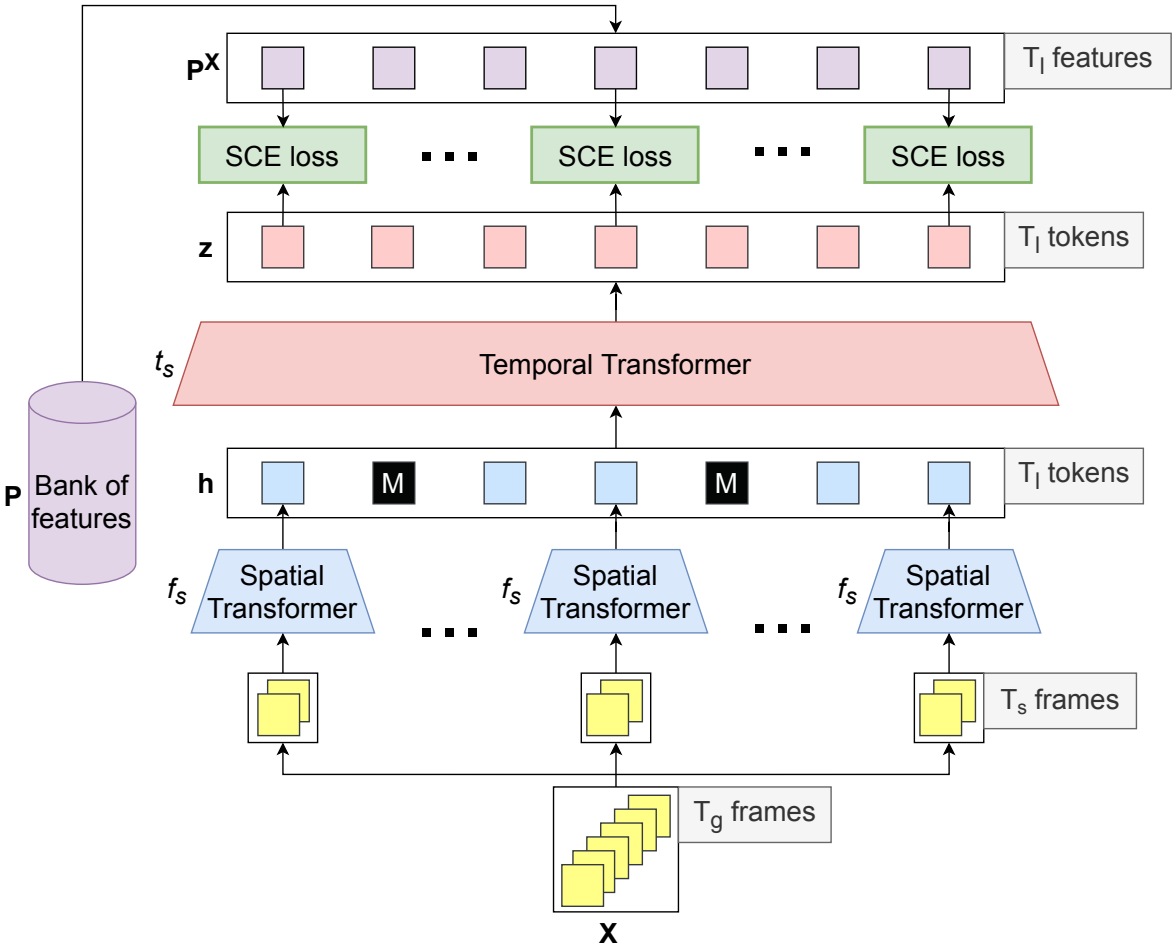


Figure 6.4: Pretraining of the spatial and temporal transformers via knowledge distillation of a bank of features with the SCE loss. Some spatial output tokens are masked.

To pretrain the spatial transformer f_s and the temporal transformer t_s, we adapt the Soft Contrastive Self-Supervised loss SCE to perform knowledge distillation as illustrated in Fig. 6.4. To do so, we consider the following inputs:

- A large video $\mathbf{X} \in \mathbb{R}^{\mathrm{T}_g,\mathrm{C},\mathrm{H},\mathrm{W}}$ of T_g frames that is divisible by T_s with $\mathrm{T}_l = \mathrm{T}_g/\mathrm{T}_s$ sampled from a dataset.
- A bank of spatio-temporal features $\mathbf{P} \in \mathbb{R}^{\mathrm{M_P},\mathrm{D}_t}$ of size $\mathrm{M_P}$ and dimension D_t that can be aligned temporally with small window of T_s frames within any sampled large video. More specifically, each small window is associated with the temporally closest feature of its middle frame. Therefore for each sample $\mathbf{X}$ a set of features $\mathbf{P}^{\mathbf{X}} \in \mathbb{R}^{\mathrm{T}_l,\mathrm{D}_t}$ is selected from $\mathbf{P}$. As a preprocessing stage, this bank of features is extracted from a pretrained model.

The video is transformed via a data augmentation $a^3 \sim \mathrm{T}^3$ such as $\mathbf{X}^3 = a^3(\mathbf{X})$. Each global window is split into $\mathrm{T}_l = \mathrm{T}_g/\mathrm{T}_s$ smaller windows and the input is reshaped to $(\mathrm{T}_l, \mathrm{T}_s, \mathrm{C}, \mathrm{H}, \mathrm{W})$. It passes through the spatial transformer to output $\mathbf{h} = f_s(\mathbf{X}^3)$ with $\mathbf{h} \in \mathbb{R}^{\mathrm{T}_l \times \mathrm{D}}$ and D the output dimension of the spatial transformer. A masking ratio α_1 is applied to replace $\alpha_1 * \mathrm{T}_l$ tokens with a learned mask token. The temporal transformer adds a positional embedding to each token and computes $\mathbf{z} = t_s(\mathbf{h})$ with $\mathbf{z} \in \mathbb{R}^{\mathrm{T}_l,\mathrm{D}_t}$ and D_t the dimension of the temporal output tokens.

The SCE loss is applied on each token of $\mathbf{z}$ with the associated set of features $\mathbf{P}^{\mathbf{X}}$ as follows. First, a target relation distribution $\mathbf{s}^2$ is computed on each of the features in $\mathbf{P}^{\mathbf{X}}$ with the complete bank. A one-hot label is mixed with this distribution with a coefficient λ to form the target distribution $\mathbf{w}^2$. Then, each token in $\mathbf{h}$ predicts this target distribution by computing its similarity distribution $\mathbf{s}^1$ with the complete bank of features. Finally, the loss is applied:

$$s^1_{ik} = \frac{\exp(\mathbf{z}^1_i \cdot \mathbf{P}_k/\tau)}{\sum_{j=1}^{\mathrm{M_P}} \cdot \exp(\mathbf{z}^1_i \cdot \mathbf{P}_j/\tau)}, \tag{6.3}$$

$$s^2_{ik} = \frac{\mathbb{1}_{\mathbf{P}^{\mathbf{X}}_i \neq \mathbf{P}_k} \cdot \exp(\mathbf{P}^{\mathbf{X}}_i \cdot \mathbf{P}_k/\tau_m)}{\sum_{j=1}^{\mathrm{M_P}} \mathbb{1}_{\mathbf{P}^{\mathbf{X}}_i \neq \mathbf{P}_j} \cdot \exp(\mathbf{P}^{\mathbf{X}}_i \cdot \mathbf{P}_j/\tau_m)}, \tag{6.4}$$

$$w^2_{ik} = \lambda \mathbb{1}_{\mathbf{P}^{\mathbf{X}}_i = \mathbf{P}_k} + (1-\lambda) s^2_{ik}, \tag{6.5}$$

$$\mathscr{L}_{\mathrm{SCE}} = -\frac{1}{\mathrm{T}_l} \sum_{i=1}^{\mathrm{T}_l} \mathbf{w}^2_i \log(\mathbf{s}^1_i). \tag{6.6}$$

The KD enforces that each temporal token contains the information of its corresponding smaller window while allowing contextual information from the larger window thanks to the temporal transformer. The SCE loss enables the spatio-temporal token representation to model the relations among spatio-temporal small windows that the bank of features contains. Depending on how the features are extracted, the KD can be considered as supervised or self-supervised as discussed in Sec. 6.4.2.

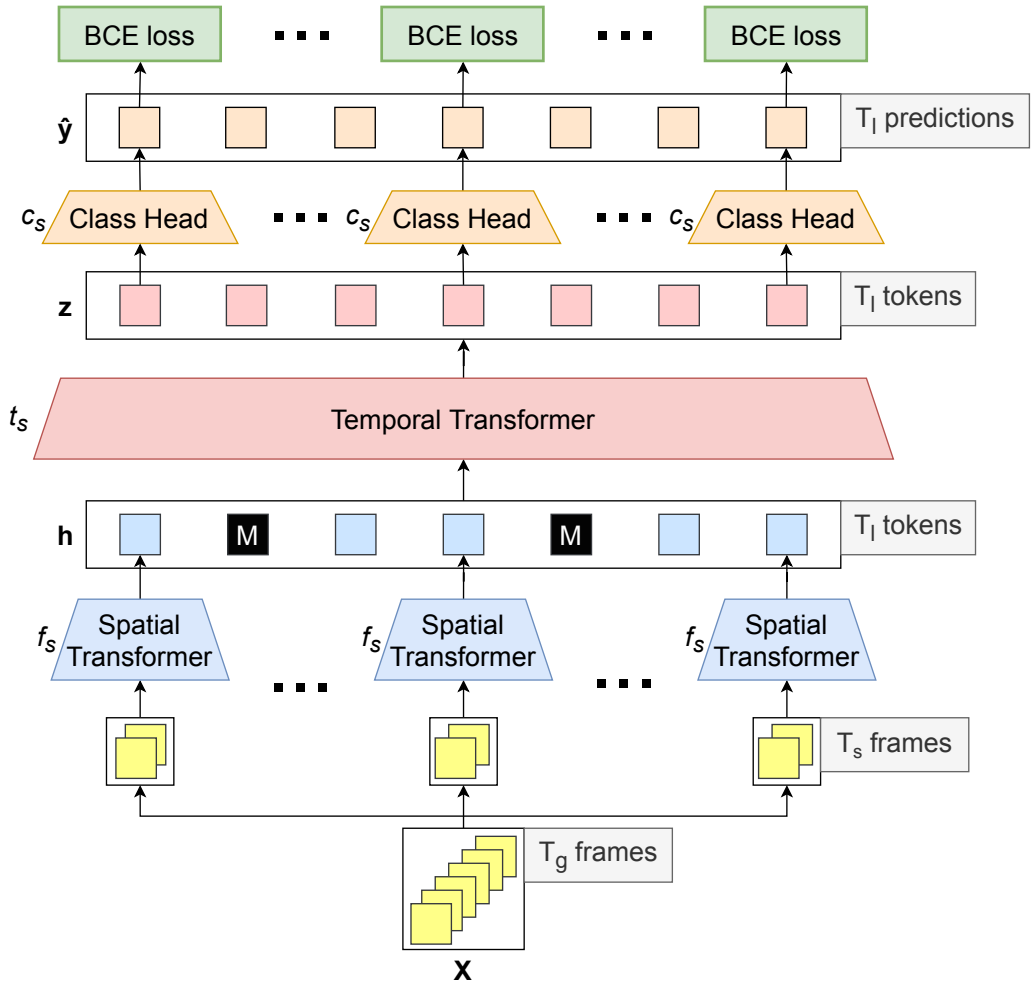


Figure 6.5: Fine-tuning to the action spotting task. Some spatial output tokens are masked.

6.3.4 Fine-tuning

The input video $\mathbf{X}$ considered in this section has the same shape as for spatio-temporal pretraining and passes through the spatial and temporal transformer that outputs $\mathbf{z} = t_s(f_s(a^4(\mathbf{X})))$ with $a^4 \sim \mathrm{A}^4$, a data augmentation.

To train the model to the action spotting task with C classes, a classification head c_s is placed upon the temporal transformer and is applied on each of the temporal output tokens to predict $\hat{\mathbf{y}} = c_s(\mathbf{z})$ with $\hat{\mathbf{y}} \in \mathbb{R}^{\mathrm{T}_l,\mathrm{C}}$ as illustrated in Fig. 6.5. Each token is associated with the average timestamp to the corresponding small window they represent. The masking strategy from the spatiotemporal pretraining is maintained during training by randomly masking $\alpha_2 \times \mathrm{T}_l$ spatial output tokens.

For supervision, each ground truth action that falls into the T_g sampled window is associated with the input frame timestamps that fall into a temporal radius displacement ϵ with the action. For each T_s smaller window, if at least one of its frames is associated with an

action, the label associated with its temporal output token is 1 and otherwise 0 to form the vector of label $\mathbf{y} \in \mathbb{R}^{\mathrm{T}_l,\mathrm{C}}$.

The action classes are considered independently, therefore we apply a Sigmoid activation function to the classifier. During training, the Binary Cross Entropy (BCE) loss is computed for each class at each timestamp as follows:

$$\mathscr{L}_{\mathrm{BCE}} = -\frac{1}{\mathrm{T}_l \times \mathrm{C}} \sum_{t=1}^{\mathrm{T}_l} \sum_{c=1}^{\mathrm{C}} \mathbf{y}_{tc} log(\hat{\mathbf{y}}_{tc}) \tag{6.7}$$

During inference, the predictions are assigned to the output timestamps of their temporal token. A sliding window with overlap is performed on the videos. For overlapped predictions, a strategy is applied to only keep one prediction per class per timestamp, such as keeping the maximum or the average of predictions.

6.4 Empirical study

In this section, we will first review the implementation details of each step, then perform an ablation study on the different parts of our pipeline, and finally compare ourselves with the state of the art.

6.4.1 Implementation details

We launched our experiments on three seeds and averaged the results.

Dataset. We performed our study on the SoccerNet-v2 [DCG+21] action spotting dataset. It contains soccer matches divided into two halves of about 45 minutes. It has three annotated splits of 17 classes with 300 matches for training, 100 for validation, and 100 for testing. There is also a challenge split that contains 50 videos for which annotations are not given. The metric used is the tight-Average mean Average Precision, t-AmAP for short, which evaluates predictions that fall on average between 1-5 seconds. We extracted the video frames at resolution 398×224 at 2 Frames Per Second (FPS). The dataset provides pre-computed Baidu [ZKC+21] features at 1 FPS for all splits. We performed a PCA on these features to reduce the dimension to 512 for distillation.

Spatial and Temporal transformer architectures. For the spatial transformer, two different architectures are used: ViT [DBK+21] and Swin [LLC+21]. The temporal transformer is a stack of 4 attention layers from a ViT architecture. For ViT, the global architecture corresponds to the ViViT model 2 [ADH+21]. We keep this name and refer to the Swin-based architecture as ViSwin. More details can be found in Appendix C.1.1. For optimization, we used the ADAMW optimizer with a weight decay of 0.05. The initial learning rate depends on the step as well as the backbone and is detailed in Appendix C.1.2.

Spatial pretraining. To perform the pretraining of the spatial transformer, we follow the method MoCo [HFW+20] and hyper-parameters from SCE. More specifically we use a projector for the online and target branches and a predictor for the online branch. For data sampling, all sub-videos of 1 second are used. Details can be found in Appendix C.1.3.

Spatio-temporal pretraining. To perform the pretraining of the spatial and temporal transformers, we follow SCE. We apply a projector on top of each output temporal token and we distill information from the reduced Baidu features. Details can be found in Appendix C.1.4. For data sampling, we randomly extract 150 videos of 32 seconds, or 64 frames, per match at each epoch.

Fine-tuning. To perform fine-tuning, common data augmentations are applied as well as mixup [ZCDL18]. For data sampling, 100 videos per match per epoch are sampled uniformly. The classifier is first initialized and then the whole backbone is fine-tuned. Details can be found in Appendix C.1.5.

Inference. During inference, a sliding window with half overlap is applied on all videos. For multiple timestamp classifications, the maximum of predictions per action is kept. No data augmentation is applied. A hard Non-Maximum Suppression (NMS) of a 5-second window is applied. The 6 first and last seconds of each window are ignored to keep predictions with past or future context.

6.4.2 Ablation study

In this subsection, we will make various ablations to highlight the advantages of our 3-step approach, our masking strategy, and how the model and data sampling affect performance. The majority of the ablation study is performed on ViViT Tiny to reduce the cost of training. The models in this section are trained on the training split and evaluated on the validation split.

Model	Params (M)	GFLOPs	t-AmAP (%)
ViViT T	7.5	41.2	64.7
ViViT S	29.1	149.5	65.9
ViSwin T	55.9	145.6	**66.1**

Table 6.1: Influence of the model architecture on the t-AmAP.

Architectures. We test two architectures for the spatial part, ViT [DBK+21] and Swin [LLC+21]. Because the output embedding dimension of the spatial transformer is the one used for the temporal transformer, the number of parameters increases quadratically with the spatial dimension. As going deeper with Swin increases the token dimension, its output dimension token is large which leads to a larger number of parameters for ViSwin's temporal encoder in comparison with ViViT's. We compare the performance of ViViT and

ViSwin in Tab. 6.1. ViSwin Tiny shows an improvement over ViViT Tiny with +1.4 percentage points (p.p). However, this improvement comes with a price of about 7.5 times more parameters. Going from Tiny to Small for ViViT improved by +1.2 p.p for about 4 times parameter. But, its GFLOPs are slightly higher than ViSwin Small suggesting the Swin spatial transformer is more efficient for larger networks.

Depth	Params (M)	GFLOPs	t-AmAP (%)
4	7.5	41.2	64.7
6	8.3	41.3	**65.4**
8	9.2	41.3	65.3

Table 6.2: Influence of temporal depth on ViViT-T for the t-AmAP.

We also test a deeper temporal transformer as the majority of computation comes from the spatial transformer by design [ADH+21]. Indeed for ViT, it sees for a global window 6,272 tokens whereas the temporal transformer only has 32. Therefore, besides increasing the number of parameters, the cost of making a deeper temporal transformer is computationally negligible in comparison with a deeper spatial transformer. The baseline is a depth of 4 blocks of attention and we increase it to 6 and 8. The results are reported in Tab. 6.2 and show that increasing the temporal depth to 6 increases the t-AmAP by 0.7 p.p and going deeper decreases performance. In contrast with ViViT applied to action classification [ADH+21] we increase performance with a deeper temporal transformer probably because action spotting requires modeling more complex temporal dependencies.

Window duration (s)	t-AmAP (%)
32	64.7
64	**66.0**
128	65.5

Table 6.3: Influence of temporal length on ViViT-T on the t-AmAP.

Size of context. Intuitively, the size of the temporal context influences how our model perceives actions. We study this influence in Tab. 6.3 by increasing 2 times and 4 times the temporal context. To keep the computational cost the same between different sizes, we adapt the batch size adequately. Increasing it 2 times improved the results by +1.3 p.p and a larger context shows a slight decrease. This verifies that for a better understanding of soccer actions, a large temporal context is necessary.

Masking. Steps 2 and 3 of our training pipeline incorporate a temporal masking strategy. This masking has two goals: limit the overfitting of our model and make the temporal transformer focus on contextual information instead of just aligning its output with its input. We show the advantage of this masking strategy in Tab. 6.4 by masking only during pretraining, only during fine-tuning, or both. First, masking during only fine-tuning drastically decreases performance by −9.2 p.p. Masking during pretraining increases results

α_1	α_2	t-AmAP (%)
None		
0.00	0.00	64.7
Only fine-tuning		
0.00	0.50	55.5
Only pretraining		
0.25	0.00	64.9
0.50	0.00	65.1
0.75	0.00	64.8
Both		
0.25	0.25	**65.2**
0.50	0.50	65.0
0.75	0.75	63.5

Table 6.4: Influence of masking ratio during spatio-temporal pretraining (α_1) and fine-tuning (α_2) on ViViT-T on the t-AmAP.

by up to 0.4 p.p for 50% tokens masked and masking during both steps increases up to 0.5 p.p for 25% tokens masked. Performance decreases with further masking. These results suggest it is necessary to initialize the mask token during pretraining. Also, the percentage of tokens to mask seems to be different for optimal performance during pretraining and fine-tuning, and fewer masking during fine-tuning seems better.

Steps. Our training pipeline consists of three steps, each adding complexity. Here, we validate the usefulness of each step. We evaluate the quality of our learned representation in step 1 by comparing its performance with a supervised pretrained ViT Tiny model on ImageNet 21k that contains 14 million labeled diverse images. To ensure a fair comparison with spatiotemporal pretrained backbones, when step 2 is not performed, we perform a longer fine-tuning.

Step 1	Step 2	Step 3 epochs	t-AmAP (%)
x	x	100F	48.1
SN	x	100F	54.7
IN	x	100F	57.7
SN	✓	50F	62.2
x	✓	30C + 20F	60.0
SN	✓	30C + 20F	64.7
IN	✓	30C + 20F	**65.0**

Table 6.5: Influence of the pretraining steps and the number of fine-tuning epochs on ViViT-T on the t-AmAP. *SN* stands for SoccerNet-v2 MoCo self-supervised pretraining, and *IN* for ImageNet21k supervised pretraining. *C* stands for training the classifier and *F* for fine-tuning the whole model.

We report results in Tab. 6.5. Each step consistently improves performance. Indeed, training from scratch reaches 48.1% t-AmAP. Adding step 1 increases up to 54.7% for SSL pre-

training and 57.7 % for ImageNet pretraining. This suggests that the spatial transformer takes advantage of initialization from a large diversity of data and that our SSL pretraining can be improved. Step 2 further improves results, even with random spatial initialization which reached 60.0% t-AmAP. With SoccerNet weights, it increases to 64.7%, and with ImageNet weights, it reaches 65.0%. The gap between SoccerNet and Imagenet pretraining in step 1 is almost closed in step 2. As our SSL approach was trained on videos, while ImageNet weights were obtained on images, we argue that our second pretraining stage benefits from having initial spatiotemporal features. Initializing the temporal transformer accelerates convergence and improves results compared to training from scratch or step 1. This reduces the cost of pretraining, allowing future work to perform fast experiments in the fine-tuning phase.

Depth	Sequence	Masking	t-AmAP (%)
x	x	x	64.7
✓	x	x	65.4
x	✓	x	66.0
x	x	✓	65.2
✓	✓	x	66.1
x	✓	✓	66.2
✓	✓	✓	**66.6**

Table 6.6: Influence of best parameters for temporal depth and length, and the masking strategy on ViViT-T on the t-AmAP.

All together. In Tab. 6.6, we test adding together the different best hyperparameters for a deeper temporal transformer, larger temporal context, and temporal masking. Previously, we showed that the larger improvement came from increasing the temporal context so we add other components to it. Increasing temporal depth adds 0.1% whilst using the masking strategy adds 0.2% which makes them marginal in comparison with previous improvements. However, combining the three improves 0.6% to attain our best result of 66.6%. This confirms that a large temporal context is the most determining component to improve performance and that the masking strategy scales with the number of parameters and ensures new information is learned.

Features	Pretraining		t-AmAP (%)
	Dataset	Fine-tuned	
SCE	K400		63.6
SCE	K400	✓	65.7
Baidu [ZKC+21]	K400	✓	**66.6**

Table 6.7: Influence of features to perform KD on ViViT-T on t-AmAP. Supervised features provide the best results and self-supervised features of SCE achieve good performance.

Bank of features. We change the bank of features used from Baidu [ZKC+21], which necessitates fine-tuning of 5 models pretrained on Kinetics400 [KCS+17] to obtain, with two

options: extracted features from SCE pretrained R3D50 on Kinetics400, as explained in Chapter 5, and its fine-tuned version to the action spotting task. The clips used for the R3D50 last 4 seconds and fine-tuning is performed on the middle frame. We report results in Tab. 6.7. Baidu features achieve best performance thanks to its 5 aggregated models, but SCE fine-tuned, which is 1 model, is enough to achieve competitive performance. Also, using the self-supervised model loses $-2.1 p.p$ but opens an interesting perspective toward pretraining a self-supervised model on a closer domain for feature extraction.

NMS	Ignore (s)	Window (s)	t-AmAP (%)
Default inference			
hard	6	5	66.6
Best inference for soft and hard NMS			
hard	12	3	67.1
soft	12	10	**68.0**

Table 6.8: Best inference parameters on ViViT-T on the t-AmAP.

Inference pipeline. The inference has also a huge impact on performance. There are 3 parameters that we take into account: whether to use *hard* or *strong* NMS, the number of seconds to ignore at the beginning and end of each window prediction, and the size of the NMS window. In Tab. 6.8, we provide the results of the best parameters that we found for hard and soft NMS which are detailed in Appendix C.2 and we empirically observe a better performance for soft NMS.

6.4.3 Comparison with the State of the Art

Implementation details. For comparison with the state of the art, we take the best settings found in the ablation study for fine-tuning and inference. The results labeled *ens.* means we use the average predictions of 3 seeds. We evaluate on the test split as well as the challenge split. When we evaluate on the test split, spatiotemporal pertaining and fine-tuning are performed on the training and validation splits, and for the challenge all annotated splits are used.

Comparison on test split. We report our results in Tab. 6.9. We compare ourselves with methods that use a sequence of images as input or a feature extractor. We observe that COMEDIAN with ViViT Tiny provides a significant improvement over the state of the art by +5.6 p.p on t-AmAP. ViSwin Tiny increases performance by +0.9 p.p but at a high cost in terms of computational usage. Finally using an ensemble of our 3 seed, we achieve 72.0% t-AmAP for ViViT Tiny and 73.1 % for ViSwin Tiny. These results empirically prove that our approach even with a small network produces state-of-the-art results by using our initializing pipeline. It is worth noting that we perform a simple fine-tuning stage. Previous approaches only focused on the fine-tuning part and because the two are not

Method	Input	t-AmAP (%)
NetVLAD++ [GG21]	F	11.5
AImageLab RMSNet [TBC+20]	F	28.8
Baidu [ZKC+21]	F	47.1
Faster-TAD [CCL+22]	F	54.1
SpotFormer [CYZ+22]	F	60.9
E2E-Spot [HZG+22]	I	61.8
Spivak [SSB22]	F	65.1
COMEDIAN (ViViT-T)	I	70.7
COMEDIAN (ViSwin-T)	I	71.6
COMEDIAN (ViViT-T - ens.)	I	72.0
COMEDIAN (ViSwin-T - ens.)	I	**73.1**

Table 6.9: Comparison with the state of the art on the test split of SoccerNet-v2. *F* stands for methods using a feature extractor, *I* for methods end-to-end with image inputs.

mutually exclusive, it opens interesting perspectives for future work to build better fine-tuning upon our approach.

Method	Input	t-AmAP (%)
Challenge 2022 leaderboard		
Baidu [ZKC+21]	F	49.56
Transformer-AS [ZLL+22]	I	52.04
Faster-TAD [CCL+22]	F	64.88
E2E-Spot [HZG+22]	I	66.73
Spivak [SSB22]	F	67.81
Challenge 2023 submission		
Spivak* [CGS+23]	F	68.33
COMEDIAN (ViViT-T - ens.)	I	68.38
team_ws_action [CGS+23]	?	69.17
ASTRA [CGS+23]	F	70.10
mt_player [CGS+23]	F	71.10
SDU_VISLAB [CGS+23]	?	**71.31**

Table 6.10: Comparison with the state of the art on the challenge split of SoccerNet-v2. *F* stands for methods using a feature extractor, *I* for methods end-to-end with image inputs and *?* for unknown.

Comparison on challenge split. We report our results in Tab. 6.10. We compare with some participants from the Challenge 2022 and the competitors having results over the baseline Spivak* of the Challenge 2023. Our proposed COMEDIAN achieves 68.38% on global t-AmAP for ViViT Tiny. Contrary to the test split we do not have a gap with state-of-the-art methods and achieve +0.56 p.p in comparison with 2022 best result, +0.05 p.p in comparison with 2023's baseline and $-2,93$ p.p compared to the best method. Because of the opacity of the challenge split's labels, it is difficult to investigate the discrepancy between the test and the challenge. Compared with the best 2022 end-to-end methods E2E-Spot [HZG+22], our approach achieves a more significant improvement of +1.64 p.p on t-AmAP.

At the time of this manuscript, we do not have access to the specifics of the competitors of the 2023 version, making it difficult to compare with COMEDIAN. However, based on the challenge report [CGS+23], these approaches, like previous ones in the literature, focused most on designing specific architectures to solve the action spotting task by using better features than Baidu's [ZKC+21], proposing multi-scale pyramidal backbones, or using different encoders based on the action type. An interesting perspective would be to use these good practices after pretraining an architecture based on COMEDIAN's first two steps to perform better fine-tuning to the action spotting task.

6.5 Conclusion

In this chapter, we introduce COMEDIAN a novel approach for Action Spotting that leverages self-supervised learning and knowledge distillation to initialize a spatio-temporal transformer. It achieves state-of-the-art results on the SoccerNet-v2 action spotting task, demonstrating the effectiveness of the proposed pipeline. By utilizing unlabeled video data for pretraining, we address the subjective and resource-intensive manual labeling processes for action spotting. The pretraining cost is leveraged by a faster and better convergence during fine-tuning. While our approach shows promising results, there are areas for improvement in the pretraining and fine-tuning steps and we hope that our approach will open the path to new methods to increase performance on action spotting and temporal action detection with spatio-temporal transformer models.
This work has been published in or submitted to:

- **Julien Denize**, Mykola Liashuha, Jaonary Rabarisoa, Astrid Orcesi, Romain Hérault. "Long-Context Transformer Pretraining Through Spatio-Temporal Knowledge Distillation for Action Spotting". *Action Spotting SoccerNet Challenge 2023*, 5th out of 12 teams.
- **Julien Denize**, et al. "COMEDIAN: Self-Supervised Learning and Knowledge Distillation for Action Spotting using Transformers". *IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, 2024.[DLR+24]
- **Julien Denize**, et al. "SoccerNet 2023 Challenges Results". *arXiv*, abs/2309.06006, 2023. [CGS+23]

Chapter 7

Conclusion et perspectives

Sommaire

7.1 Conclusion

Self-supervised learning has gained immense popularity in the field of computer vision for image and video analysis. This enthusiasm stems from its ability to utilize vast amounts of unlabeled data without the cost of time, computations and bias associated with manual labeling to pretrain a neural network. Pretrained neural networks contain general concepts and can be specialized in various downstream tasks, such as image classification or action recognition, faster with fewer needed annotated instances.

Within self-supervised learning, various approaches have emerged, each with its strengths and drawbacks. Geometric and intensity pretext tasks are easy to design but may not provide enough information for complex tasks. Clustering methods offer simplicity and surprisingly good results but can be sensitive to hyperparameters and don't scale well. Contrastive learning scales effectively and can produce general representations but requires careful design of positives and handling of negatives. Masked modeling excels at capturing local information but struggles with global context and demands redundant data and specialized architectures.

This thesis has focused on improving contrastive methods using negatives for self-supervised image and video representation learning.

In Chapter 4, we introduced our Similarity Contrastive Estimation (SCE) soft contrastive learning approach which leverages relations among instances to model data structure and enhance contrastive method performance. SCE outperforms traditional contrastive and relational learning methods, as shown theoretically and empirically and has fast convergence.

Expanding SCE to video analysis in Chapter 5, we provide a novel approach to self-supervised video representation learning for global representation learning. This extension addresses the challenges of creating positive pairs by leveraging multiple data augmentations that take into account the spatiotemporal aspect of the videos, outperforming traditional contrastive methods on various benchmarks.

Finally, we introduced the COMEDIAN multi-step pretraining approach in Chapter 6 to propose a self-supervised learning pipeline for temporally localized video tasks. By performing a spatial pretraining using an SSL training objective followed by a spatiotemporal knowledge distillation of local temporal features using SCE and Temporal Masked Modeling, we improve the performance of transformer architectures and fasten convergence on action spotting tasks.

Because pretraining involves a non-negligible computational cost and specific training, we made our contributions accessible and provided code and model checkpoints in a single repository[1] [Den23b].

In summary, this thesis advances self-supervised learning for image and video analysis by enhancing contrastive methods via our new soft contrastive learning approach SCE. Our contributions have proven to reach state-of-art results on various image and video analysis tasks. As self-supervised learning continues to evolve, we hope our work serves as a foundation for future research and applications.

7.2 Perspectives

Although our approaches provided results at the state of the art on various image and video applications there is room for improvement.

To improve our Similarity Contrastive Estimation soft contrastive learning approach we identify the following leads:

- *Better Relation Estimation*: Enhance the relation estimation in SCE by improving positive pair generation. Instead of solely relying on random data augmentations, make use of IoU (Intersection over Union) to classify a pair as positive if the IoU is over a threshold and negative otherwise. For images, only consider a spatial IoU and for negatives a spatiotemporal IoU. This should better capture the relationships between objects in both image and video domains as it will avoid training instabilities caused by non-overlapping views. For video global representation learning, it has been shown experimentally that emphasizing motion improves performance, therefore putting more attention to this aspect for relation estimation is also an interesting perspective.
- *Theoretical comparison with dimension reduction*: SSL can be seen as a dimensional reduction technique as the data are encoded thanks to the learned backbone to a representation that has generally a much smaller dimension than the input. It could be interesting to study under this prism SCE and in particular its link with SNE [HR02] and t-SNE [VdMH08].
- *Integration with Masked Modeling*: Combine SCE with masked modeling techniques on the input data, and not only on temporal input as for COMEDIAN, to learn better local features. To do so a promising masking strategy is using teacher-assisted masking to image and video data by examining which tokens the teacher model attends to during training.

To improve our COMEDIAN pretraining pipeline we thought of the following tracks:

[1] Eztorch repository: `https://github.com/juliendenize/eztorch`.

- *Global Objective with Class Token*: Integrate a global temporal representation learning objective alongside the local temporal distillation by introducing a class token and applying the global objective to it using a standard SSL method such as SCE. This class token can help the model learn a global representation that takes into account the entire input sequence, enhancing its understanding of context and relationships within the data. Pretraining such objectives would allow COMEDIAN to be fine-tuned on downstream tasks that need local temporal representations but also a global temporal representation such as for action recognition.
- *Large-Scale Domain-Specific Pretraining*: Launch a large-scale pretraining phase on domain-specific data for COMEDIAN. This specialized pretraining can help the model become more effective in handling domain-specific tasks and improve its performance in specific applications. Because COMEDIAN pretraining pipeline can be self-supervised entirely, even for the knowledge distillation step, this can be accomplished.
- *Dedicated Architecture for Temporal Action Detection (TAD) and Loss Functions*: We used a standard ViT architecture that is not the best architecture for TAD-related tasks. Pretraining a dedicated architecture tailored for TAD should improve the performance but might necessitate some adjustments.

Appendix A

Supplementary material SCE Image

A.1 Classes to construct ImageNet100

To build the ImageNet100 dataset, we used the classes shared by the CMC [TKI20] authors in the supplementary material of their publication. We also share these classes in Tab. A.1.

100 selected classes from ImageNet					
n02869837	n01749939	n02488291	n02107142	n13037406	n02091831
n04517823	n04589890	n03062245	n01773797	n01735189	n07831146
n07753275	n03085013	n04485082	n02105505	n01983481	n02788148
n03530642	n04435653	n02086910	n02859443	n13040303	n03594734
n02085620	n02099849	n01558993	n04493381	n02109047	n04111531
n02877765	n04429376	n02009229	n01978455	n02106550	n01820546
n01692333	n07714571	n02974003	n02114855	n03785016	n03764736
n03775546	n02087046	n07836838	n04099969	n04592741	n03891251
n02701002	n03379051	n02259212	n07715103	n03947888	n04026417
n02326432	n03637318	n01980166	n02113799	n02086240	n03903868
n02483362	n04127249	n02089973	n03017168	n02093428	n02804414
n02396427	n04418357	n02172182	n01729322	n02113978	n03787032
n02089867	n02119022	n03777754	n04238763	n02231487	n03032252
n02138441	n02104029	n03837869	n03494278	n04136333	n03794056
n03492542	n02018207	n04067472	n03930630	n03584829	n02123045
n04229816	n02100583	n03642806	n04336792	n03259280	n02116738
n02108089	n03424325	n01855672	n02090622		

Table A.1: The 100 classes selected from ImageNet to construct ImageNet100.

A.2 Pseudo-Code of SCE

```
# dataloader: loader of batches
# bsz: batch size
# epochs: number of epochs
# T1: weak distribution of data augmentations
# T2: strong distribution of data augmentations
# f_s, g_s, h_s: online encoder, projector, and optional predictor
# f_t, g_t: momentum encoder and projector
# queue: memory buffer
# tau: online temperature
# tau_m: momentum temperature
# lambda_: coefficient between contrastive and relational aspects
# symmetry_loss: if True, symmetries the loss

def sce_loss(z1, z2):
    sim2_pos = zeros(bsz)
    sim2_neg = einsum("nc,kc->nk", z2, queue)
    sim2 = cat([sim2_pos, sim2_neg]) / tau_m
    s2 = softmax(sim2)
    w2 = lambda_ * one_hot(sim2_pos, bsz+1) + (1 - lambda_) * s

    sim1_pos = einsum("nc,nc->n", z1, z2)
    sim1_neg = einsum("nc,kc->nk", z1, queue)
    sim1 = cat([sim1_pos, sim1_neg]) / tau
    p1 = softmax(sim1)

    loss = cross_entropy(p1, w2)
    return loss

for i in range(epochs):
    for x in dataloader:
        x1, x2 = T1(x), T2(x)

        z1_s, z2_t = h_s(g_s(f_s(x1))), g_t(f_t(x2))
        z2_t = stop_grad(z2_t)

        loss = sce_loss(z1_s, z2_t)
        if symmetry_loss:
            z1_t, z2_s = g_t(f_t(x1)), h_s(g_s(f_s(x2)))
            z1_t = stop_grad(z1_t)
            loss += sce_loss(z2_s, z1_t)
            loss /= 2
        loss.backward()

        update(f_s.params)
        update(g_s.params)
        update(h_s.params)
        momentum_update(f_t.params, f_s.params)
        momentum_update(g_t.params, g_s.params)

        fifo_update(queue, z2_t)
        if symmetry_loss:
            fifo_update(queue, z1_t)
```

Algorithm A.1: Pseudo-Code of SCE in Pytorch style

A.3 Proof Proposition 1.

Proposition. $\mathscr{L}_{\mathrm{SCE}}$ *defined as*

$$\mathscr{L}_{\mathrm{SCE}} = -\frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \sum_{k=1}^{\mathrm{N}} \mathrm{w}_{ik}^2 \log\left(\mathrm{p}_{ik}^1\right),$$

can be written as:

$$\mathscr{L}_{\mathrm{SCE}} = \lambda \cdot \mathscr{L}_{\mathrm{I}nfo\mathrm{NCE}} + \mu \cdot \mathscr{L}_{\mathrm{ReSSL}} + \eta \cdot \mathscr{L}_{ceil},$$

with $\mu = \eta = 1 - \lambda$ *and*

$$\mathscr{L}_{\mathrm{C}eil} = -\frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \log\left(\frac{\sum_{j=1}^{\mathrm{N}} \mathbb{1}_{i \neq j} \cdot \exp(\mathbf{z}_i^1 \cdot \mathbf{z}_j^2 / \tau)}{\sum_{j=1}^{\mathrm{N}} \exp(\mathbf{z}_i^1 \cdot \mathbf{z}_j^2 / \tau)}\right).$$

Proof. Recall that:

$$\begin{aligned}
p_{ik}^1 &= \frac{\exp(\mathbf{z}_i^1 \cdot \mathbf{z}_k^2 / \tau)}{\sum_{j=1}^{\mathrm{N}} \exp(\mathbf{z}_i^1 \cdot \mathbf{z}_j^2 / \tau)}, \\
s_{ik}^2 &= \frac{\mathbb{1}_{i \neq k} \cdot \exp(\mathbf{z}_i^2 \cdot \mathbf{z}_k^2 / \tau_m)}{\sum_{j=1}^{\mathrm{N}} \mathbb{1}_{i \neq j} \cdot \exp(\mathbf{z}_i^2 \cdot \mathbf{z}_j^2 / \tau_m)}, \\
w_{ik}^2 &= \lambda \cdot \mathbb{1}_{i=k} + (1-\lambda) \cdot s_{ik}^2.
\end{aligned}$$

We decompose the second loss over k in the definition of $\mathrm{L}_{\mathrm{SCE}}$ to make the proof:

$$\begin{aligned}
\mathscr{L}_{\mathrm{SCE}} &= -\frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \sum_{k=1}^{\mathrm{N}} \mathrm{w}_{ik}^2 \log\left(\mathrm{p}_{ik}^1\right) \\
&= -\frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \left[\mathrm{w}_{ii}^2 \log\left(\mathrm{p}_{ii}^1\right) + \sum_{\substack{k=1 \\ k \neq i}}^{\mathrm{N}} \mathrm{w}_{ik}^2 log\left(\mathrm{p}_{ik}^1\right) \right] \\
&= \underbrace{-\frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \mathrm{w}_{ii}^2 \left(\mathrm{p}_{ii}^1\right)}_{(1)} \underbrace{-\frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \sum_{\substack{k=1 \\ k \neq i}}^{\mathrm{N}} \mathrm{w}_{ik}^2 \log\left(\mathrm{p}_{ik}^1\right)}_{(2)}.
\end{aligned}$$

First we rewrite (1) to retrieve the $\mathscr{L}_{\mathrm{I}nfo\mathrm{NCE}}$ loss.

$$\begin{aligned}
(1) &= -\frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} w_{ii}^2 log\left(\mathrm{p}_{ii}^1\right) \\
&= -\frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \lambda \cdot \log\left(p_{ii}^1\right) \\
&= -\lambda \cdot \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \log\left(\frac{\exp(\mathbf{z}_i^1 \cdot \mathbf{z}_i^2 / \tau)}{\sum_{j=1}^{\mathrm{N}} \exp(\mathbf{z}_i^1 \cdot \mathbf{z}_j^2 / \tau)}\right) \\
&= \lambda \cdot \mathrm{L}_{\mathrm{I}nfo\mathrm{NCE}}.
\end{aligned}$$

Now we rewrite (2) to retrieve the $\mathscr{L}_{\mathrm{ReSSL}}$ and $\mathscr{L}_{Ceil}$ losses.

$$
\begin{aligned}
(2) &= -\frac{1}{\mathrm{N}}\sum_{i=1}^{\mathrm{N}}\sum_{\substack{k=1\\k\neq i}}^{\mathrm{N}} \mathrm{w}_{ik}^{2}\log\left(p_{ik}^{1}\right)\\
&= -\frac{1}{\mathrm{N}}\sum_{i=1}^{\mathrm{N}}\sum_{\substack{k=1\\k\neq i}}^{\mathrm{N}} (1-\lambda)\cdot s_{ik}^{2}\cdot\log\left(p_{ik}^{1}\right)\\
&= -(1-\lambda)\cdot\frac{1}{\mathrm{N}}\sum_{i=1}^{\mathrm{N}}\sum_{k=1}^{\mathrm{N}} \mathrm{s}_{ik}^{2}\cdot\log\left(p_{ik}^{1}\right)\\
&= -(1-\lambda)\cdot\frac{1}{\mathrm{N}}\sum_{i=1}^{\mathrm{N}}\sum_{k=1}^{\mathrm{N}}\left[\mathrm{s}_{ik}^{2}\cdot\log\left(\frac{\exp(\mathbf{z}_{i}^{1}\cdot\mathbf{z}_{k}^{2}/\tau)}{\sum_{j=1}^{\mathrm{N}}\exp(\mathbf{z}_{i}^{1}\cdot\mathbf{z}_{j}^{2}/\tau)}\right)\right]\\
&= -(1-\lambda)\cdot\frac{1}{\mathrm{N}}\sum_{i=1}^{\mathrm{N}}\sum_{k=1}^{\mathrm{N}}\left[\mathrm{s}_{ik}^{2}\cdot\left(\right.\right.\\
&\qquad \left.\left.\log\left(\exp(\mathbf{z}_{i}^{1}\cdot\mathbf{z}_{k}^{2}/\tau)\right)-\log\left(\sum_{j=1}^{\mathrm{N}}\exp(\mathbf{z}_{i}^{1}\cdot\mathbf{z}_{j}^{2}/\tau)\right)\right)\right]\\
&= -(1-\lambda)\cdot\frac{1}{\mathrm{N}}\sum_{i=1}^{\mathrm{N}}\sum_{k=1}^{\mathrm{N}}\left[s_{ik}^{2}\cdot\left(\right.\right.\\
&\qquad \log\left(\exp(\mathbf{z}_{i}^{1}\cdot\mathbf{z}_{k}^{2}/\tau)\right)-\log\left(\sum_{j=1}^{\mathrm{N}}\exp(\mathbf{z}_{i}^{1}\cdot\mathbf{z}_{j}^{2}/\tau)\right)+\\
&\qquad \log\left(\sum_{j=1}^{\mathrm{N}}\mathbb{1}_{i\neq j}\cdot\exp(\mathbf{z}_{i}^{1}\cdot\mathbf{z}_{j}^{2}/\tau)\right)-\\
&\qquad \left.\left.\log\left(\sum_{j=1}^{\mathrm{N}}\mathbb{1}_{i\neq j}\cdot\exp(\mathbf{z}_{i}^{1}\cdot\mathbf{z}_{j}^{2}/\tau)\right)\right)\right]\\
&= -(1-\lambda)\cdot\frac{1}{\mathrm{N}}\sum_{i=1}^{\mathrm{N}}\sum_{k=1}^{\mathrm{N}}\left[\mathrm{s}_{ik}^{2}\cdot\left(\right.\right.\\
&\qquad \log\left(\frac{\exp(\mathbf{z}_{i}^{1}\cdot\mathbf{z}_{k}^{2}/\tau)}{\sum_{j=1}^{\mathrm{N}}\mathbb{1}_{i\neq j}\cdot\exp(\mathbf{z}_{i}^{1}\cdot\mathbf{z}_{j}^{2}/\tau)}\right)+\\
&\qquad \left.\left.\log\left(\frac{\sum_{j=1}^{\mathrm{N}}\mathbb{1}_{i\neq j}\cdot\exp(\mathbf{z}_{i}^{1}\cdot\mathbf{z}_{j}^{2}/\tau)}{\sum_{j=1}^{\mathrm{N}}\exp(\mathbf{z}_{i}^{1}\cdot\mathbf{z}_{j}^{2}/\tau)}\right)\right)\right]\\
&= -(1-\lambda)\cdot\frac{1}{\mathrm{N}}\left(\right.\\
&\sum_{i=1}^{\mathrm{N}}\sum_{k=1}^{\mathrm{N}}\left[\mathrm{s}_{ik}^{2}\cdot\log\left(\frac{\exp(\mathbf{z}_{i}^{1}\cdot\mathbf{z}_{k}^{2}/\tau)}{\sum_{j=1}^{\mathrm{N}}\mathbb{1}_{i\neq j}\cdot\exp(\mathbf{z}_{i}^{1}\cdot\mathbf{z}_{j}^{2}/\tau)}\right)\right]+\\
&\sum_{i=1}^{\mathrm{N}}\sum_{k=1}^{\mathrm{N}}\left[\mathrm{s}_{ik}^{2}\cdot\log\left(\frac{\sum_{j=1}^{\mathrm{N}}\mathbb{1}_{i\neq j}\cdot\exp(\mathbf{z}_{i}^{1}\cdot\mathbf{z}_{j}^{2}/\tau)}{\sum_{j=1}^{\mathrm{N}}\exp(\mathbf{z}_{i}^{1}\cdot\mathbf{z}_{j}^{2}/\tau)}\right)\right]\left.\right).
\end{aligned}
$$

Because $s^2_{ii} = 0$ and $\mathbf{s}^2_i$ is a probability distribution, we have:

$$\begin{aligned}(2) = &-(1-\lambda)\cdot \\ &\frac{1}{N}\sum_{i=1}^{N}\sum_{\substack{k=1\\k\neq i}}^{N}\left[s^2_{ik}\cdot\log\left(\frac{\mathbb{1}_{i\neq k}\cdot\exp(\mathbf{z}^1_i\cdot\mathbf{z}^2_k/\tau)}{\sum_{j=1}^{N}\mathbb{1}_{i\neq j}\cdot\exp(\mathbf{z}^1_i\cdot\mathbf{z}^2_j/\tau)}\right)\right]- \\ &(1-\lambda)\cdot\frac{1}{N}\sum_{i=1}^{N}\left[\log\left(\frac{\sum_{j=1}^{N}\mathbb{1}_{i\neq j}\cdot\exp(\mathbf{z}^1_i\cdot\mathbf{z}^2_j/\tau)}{\sum_{j=1}^{N}\exp(\mathbf{z}^1_i\cdot\mathbf{z}^2_j/\tau)}\right)\right] \\ =&(1-\lambda)\cdot\mathscr{L}_{\mathrm{ReSSL}}+(1-\lambda)\cdot\mathscr{L}_{Ceil}.\end{aligned}$$

□

A.4 Proof Proposition 2.

Proposition. $\mathscr{L}_{\mathrm{SCE}}$ *defined as*

$$\mathscr{L}_{\mathrm{SCE}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{N} w^2_{ik}\log\left(p^1_{ik}\right),$$

can be written as:

$$\mathscr{L}_{\mathrm{SCE}} = \lambda\cdot\mathscr{L}_{\mathrm{DCL}} + (1-\lambda)\cdot\mathscr{L}_{\mathrm{ReSSL}} + \mathscr{L}_{Ceil}.$$

Proof. As shown in Proposition 1:

$$\mathscr{L}_{\mathrm{SCE}} = \lambda\cdot\mathscr{L}_{Info\mathrm{NCE}} + (1-\lambda)\cdot\mathscr{L}_{\mathrm{ReSSL}} + (1-\lambda)\cdot\mathscr{L}_{Ceil}.$$

Recall that:

$$\mathscr{L}_{\mathrm{DCL}} = -\frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{z}^1_i\cdot\mathbf{z}^2_i/\tau - \log\left(\sum_{j=1}^{N}\mathbb{1}_{i\neq k}\exp(\mathbf{z}^1_i\cdot\mathbf{z}^2_j/\tau)\right)\right).$$

Let us prove that $\mathscr{L}_{Info\mathrm{NCE}} = \mathscr{L}_{\mathrm{DCL}} + \mathscr{L}_{Ceil}$:

$$\begin{aligned}\mathscr{L}_{Info\mathrm{NCE}} - \mathscr{L}_{Ceil} &= -\frac{1}{N}\sum_{i=1}^{N}\log\left(\frac{\exp(\mathbf{z}^1_i\cdot\mathbf{z}^2_i/\tau)}{\sum_{j=1}^{N}\exp(\mathbf{z}^1_i\cdot\mathbf{z}^2_j/\tau)}\right)+\frac{1}{N}\sum_{i=1}^{N}\log\left(\frac{\sum_{j=1}^{N}\mathbb{1}_{i\neq j}\cdot\exp(\mathbf{z}^1_i\cdot\mathbf{z}^2_j/\tau)}{\sum_{j=1}^{N}\exp(\mathbf{z}^1_i\cdot\mathbf{z}^2_j/\tau)}\right) \\ &= -\frac{1}{N}\sum_{i=1}^{N}\left[\mathbf{z}^1_i\cdot\mathbf{z}^2_i/\tau - \log\sum_{j=1}^{N}\exp(\mathbf{z}^1_i\cdot\mathbf{z}^2_j/\tau) - \right. \\ &\qquad \left.\log\left(\sum_{j=1}^{N}\mathbb{1}_{i\neq j}\cdot\exp(\mathbf{z}^1_i\cdot\mathbf{z}^2_j/\tau)\right)+\log\left(\sum_{j=1}^{N}\exp(\mathbf{z}^1_i\cdot\mathbf{z}^2_j/\tau)\right)\right] \\ &= -\frac{1}{N}\sum_{i=1}^{N}\left[\mathbf{z}^1_i\cdot\mathbf{z}^2_i/\tau - \log\left(\sum_{j=1}^{N}\mathbb{1}_{i\neq j}\cdot\exp(\mathbf{z}^1_i\cdot\mathbf{z}^2_j/\tau)\right)\right] \\ &= \mathscr{L}_{\mathrm{DCL}}.\end{aligned}$$

Therefore:

$$\begin{aligned}\mathscr{L}_{\text{SCE}} &= \lambda \cdot \mathscr{L}_{Info\text{NCE}} + (1-\lambda) \cdot \mathscr{L}_{Re\text{SSL}} + (1-\lambda) \cdot \mathscr{L}_{Ceil} \\ &= \lambda \cdot (\mathscr{L}_{Info\text{NCE}} - \mathscr{L}_{Ceil}) + (1-\lambda) \cdot \mathscr{L}_{Re\text{SSL}} + \mathscr{L}_{Ceil} \\ &= \lambda \cdot \mathscr{L}_{\text{DCL}} + (1-\lambda) \cdot \mathscr{L}_{Re\text{SSL}} + \mathscr{L}_{Ceil}.\end{aligned}$$

□

Backbone	Dataset	Projector				Input	Buffer	ema	LR	Batch	WD
		Layers	Hid dim	Out dim	BN						
R-18	CIFAR	2	512	128	hid	32^2	4,096	0.900	0.06	256	$5e^{-4}$
R-18	STL10	2	512	128	hid	96^2	16,384	0.996	0.06	256	$5e^{-4}$
R-18	Tiny-IN	2	512	128	hid	64^2	16,384	0.996	0.06	256	$5e^{-4}$
R-50	IN100	2	4096	256	no	224^2	65,536	0.996	0.3	512	$1e^{-4}$
R-50	IN1k	3	2048	256	all	224^2	65,536	0.996	0.5	512	$1e^{-4}$

Table A.2: Architecture and hyperparameters used for pretraining on the different datasets. **LR** stands for the initial learning rate, **WD** for weight decay, **BN** for batch normalization [IS15], **Hid** for hidden, **Dim** for dimension, **ema** for the initial momentum value used to update the momentum branch. For **BN**: “no" means no batch normalization is used in the projector, “hid" means batch normalization after each hidden layer, “all" means batch normalization after the hidden layer and the output layer.

A.5 Implementation details

A.5.1 Ablation study and baseline comparison for images

Pretraining Implementation details. We use the ResNet-50 [HZRS16] encoder for large datasets and ResNet-18 for small and medium datasets with changes detailed below. We pretrain the models for 200 epochs. We apply by default *strong* and *weak* data augmentations, defined in Tab. 4.1, with the scaling range for the random resized crop set to $(0.2, 1.0)$. Specific hyperparameters for each dataset for the projector construction, the size of the input, the size of the memory buffer, the initial momentum value, the initial learning rate, the batch size and the weight decay applied can be found in Tab. A.2. We use the SGD optimizer [SMDH13] with a momentum of 0.9. A linear warmup is applied during 5 epochs to reach the initial learning rate. The learning rate is scaled using the linear scaling rule and follows the cosine decay scheduler without restart [LH17]. The momentum value to update the target branch follows a cosine strategy from its initial value to reach 1 at the end of training. We do not symmetrize the loss by default.

Architecture change for small and medium datasets. Because the images are smaller, and ResNet is suitable for larger images, typically 224×224, we follow guidance from SimCLR [CKNH20] and replace the first 7×7 Conv of stride 2 with a 3×3 Conv of stride 1. We also remove the first pooling layer.

Evaluation protocol. To evaluate our pretrained encoders, we train a linear classifier following [CFGH20, ZYW+21]. We train for 100 epochs on top of the frozen pretrained en-

coder using an SGD optimizer with an initial learning rate of 30 without weight decay and a momentum of 0.9. A scheduler is applied to the learning rate that is decayed by a factor of 0.1 at 60 and 80 epochs. The data augmentations for the different datasets are:

- **training set for large datasets**: random resized crop to resolution 224×224 with the scaling range set to $(0.08, 1.0)$ and a random horizontal flip with a probability of 0.5.
- **training set for small and medium datasets**: random resized crop to the dataset resolution with a padding of 4 for small datasets and the scaling range set to $(0.08, 1.0)$. Also, a random horizontal flip with a probability of 0.5 is applied.
- **validation set for large datasets**: resize to resolution 256×256 and center crop to resolution 224×224.
- **validation set for small and medium datasets**: resize to the dataset resolution.

A.5.2 Imagenet study

Pretraining implementation details. We use the ResNet-50 [HZRS16] encoder and apply *strong*-α and *strong*-β augmentations, defined in Tab. 4.1, with the scaling range for the random resized crop set to $(0.2, 1.0)$. The batch size is set to 4096 and the memory buffer to 65,536. We follow the same training hyperparameters as [CXH21] for the architecture. Specifically, we use the same projector and predictor, the LARS optimizer [YGG17b] with a weight decay of $1.5 \cdot 10^{-6}$ for 1000 epochs of training and 10^{-6} for fewer epochs. Bias and batch normalization [IS15] parameters are excluded. The initial learning rate is 0.5 for 100 epochs and 0.3 for more epochs. It is linearly scaled for 10 epochs and it follows the cosine annealed scheduler. The momentum value follows a cosine scheduler from 0.996 for 1000 epochs, 0.99 for fewer epochs, to reach 1 at the end of training. The loss is symmetrized. For SCE specific hyperparameters, we keep the best from ablation study: $\lambda = 0.5$, $\tau = 0.1$ and $\tau_m = 0.07$.

Multi-crop setting. We follow [HWHQ21] and sample 6 different views. The first two views are global views as without multi-crop, meaning resolution of 224×224 and the scaling range for random resized crop set to $(0.2, 1.0)$. The 4 local crops have a resolution of 192×192, 160×160, 128×128, 96×96 and scaling range (0.172, 0.86), (0.143, 0.715), (0.114, 0.571), (0.086, 0.429) on which we apply the *strong*-γ data augmentation defined in Tab. 4.1.

Evaluation protocol. We follow the protocol defined by [CXH21]. Specifically, we train a linear classifier for 90 epochs on top of the frozen encoder with a batch size of 1024 and a SGD optimizer with a momentum of 0.9 and without weight decay. The initial learning rate is 0.1 and scaled using the linear scaling rule and follows the cosine decay scheduler without restart [LH17]. The data augmentations applied are:

- **training set**: random resized crop to resolution 224×224 with the scaling range set to $(0.08, 1.0)$ and a random horizontal flip with a probability of 0.5.
- **validation set**: resize to resolution 256×256 and center crop to resolution 224×224.

Dataset	τ	$\tau_m = 0.03$	$\tau_m = 0.04$	$\tau_m = 0.05$	$\tau_m = 0.06$	$\tau_m = 0.07$	$\tau_m = 0.08$	$\tau_m = 0.09$	$\tau_m = 0.1$
CIFAR10	0.1	89.93	90.03	90.06	90.20	90.16	90.06	89.67	88.97
CIFAR10	0.2	89.98	90.12	90.12	90.05	90.13	90.09	90.22	**90.34**
CIFAR100	0.1	64.49	64.90	65.19	65.33	65.27	**65.45**	64.89	63.87
CIFAR100	0.2	63.71	63.74	63.89	64.05	64.24	64.23	64.10	64.30
STL10	0.1	89.34	**89.94**	89.87	89.84	89.72	89.52	88.99	88.41
STL10	0.2	88.4	88.23	88.4	88.35	87.54	88.32	88.80	88.59
Tiny-IN	0.1	50.23	51.12	51.41	51.66	**51.90**	51.58	51.37	50.46
Tiny-IN	0.2	48.56	48.85	48.35	48.98	49.06	49.15	49.66	49.64

Table A.3: Effect of varying the temperature parameters τ_m and τ on the Top-1 accuracy.

A.5.3 Sports-field registration

We use the ViT tiny [DBK+21] architecture and the temperatures $\tau = 0.1$, $\tau_m = 0.07$ and the coefficient $\lambda = 0.5$. The projector and predictor are a 2 and 3-layer Multi-Layer Perceptron (MLP) with a hidden size of 1024 and an output size of 256. For data augmentations, we use *strong*-α and *strong*-β and symmetrize the loss. The random aspect ratio for the random resized crop is sampled between $[1.33, 2.21]$ to deal with source images of ratio $1.77:1$. We use the AdamW optimizer with a batch size of 1024 and the learning rate follows a warmup during 10 epochs to reach the initial value 2×10^{-3} and decrease following a cosine scheduler to 2×10^{-5} throughout 100 epochs of training. The weight decay is set to 0.05.

A.6 Temperature influence on small and medium datasets

We made a temperature search on CIFAR10, CIFAR100, STL10 and Tiny-ImageNet by varying τ in $\{0.1, 0.2\}$ and τ_m in $\{0.03, ..., 0.10\}$. The results are in Tab. A.3. As for ImageNet100, we need a sharper distribution on the output of the momentum encoder. Unlike ReSSL [ZYW+21], SCE do not collapse when $\tau_m \to \tau$ thanks to the contrastive aspect. For our baselines comparison in Sec. 4.2, we use the best temperatures found for each dataset.

Appendix B

Supplementary material SCE Video

stage	ResNet3d-18	ResNet3D-50
conv1	$1\times 7^2, 64$ stride $1, 2^2$	$1\times 7^2, 64$ stride $1, 2^2$
pool1	$1\times 3^2, max$ stride $1, 2^2$	$1\times 3^2, max$ stride $1, 2^2$
res_2	$\begin{bmatrix} 1\times 3^2, 64 \\ 1\times 3^2, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 1\times 1^2, 64 \\ 1\times 3^2, 64 \\ 1\times 1^2, 256 \end{bmatrix} \times 3$
res_3	$\begin{bmatrix} 1\times 3^2, 128 \\ 1\times 3^2, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 1\times 1^2, 128 \\ 1\times 3^2, 128 \\ 1\times 1^2, 512 \end{bmatrix} \times 4$
res_4	$\begin{bmatrix} 3\times 3^2, 256 \\ 1\times 3^2, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times 1^2, 256 \\ 1\times 3^2, 256 \\ 1\times 1^2, 1024 \end{bmatrix} \times 6$
res_5	$\begin{bmatrix} 3\times 3^2, 512 \\ 1\times 3^2, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times 1^2, 512 \\ 1\times 3^2, 512 \\ 1\times 1^2, 2048 \end{bmatrix} \times 3$
pool	global average	global average

Table B.1: ResNet3D-18 and ResNet3D-50 networks.

Backbone	Dataset	Projector				Predictor				Buffer
		Layers	Hid dim	Out dim	BN	Layers	Hid dim	Out dim	BN	
ResNet3D-18	K200	3	1024	256	all	2	1024	256	hid	32768
ResNet3D-18	K400	3	1024	256	all	2	1024	256	hid	65536
ResNet3D-50	K400	3	4096	256	all	2	4096	256	hid	65536

Table B.2: Architecture and hyperparameters used for video pretraining. **BN** stands for for Batch Normalization, **Hid** for hidden, **Dim** for dimension. For **BN**: "hid" means batch normalization after each hidden layer, "all" means batch normalization after the hidden layer and the output layer.

B.1 Implementation details

Pretraining implementation details. We used the ResNet3D-18 and ResNet3D-50 networks [HKS18] following the Slow path of [FFMH19]. The exact architecture details can be found in Tab. B.1. We kept the siamese architecture used for ImageNet in Sec. 4.1.3 and depending on the backbone and pretraining dataset, the projector and predictor architectures as well as the memory buffer size vary and are referenced in Tab. B.2. The LARS optimizer with a weight decay of 1.10^{-6}, batch normalization and bias parameters excluded, for 200 epoch of training is used. The learning rate follows a linear warmup until it reaches an initial value of 2.4 and then follows a cosine annealed scheduler. The initial learning rate is scaled following the linear scaling rule and the batch size is set to 512. The momentum value follows a cosine scheduler from 0.99 to 1 and the loss is symmetrized. To sample and crop different views from a video, we follow [FFX+21] and sample randomly different clips from the video that lasts 2.56 seconds. For Kinetics it corresponds to 64 frames for a frame rate per second (FPS) of 25. Out of this clip we keep a number of frames specified in Sec. 5.4. By default, we sample two different clips to form positives and we apply the *strong*-α and *strong*-β augmentations, defined in Tab. 1 in Tab. 4.1, to the views.

Linear evaluation protocol details. We follow [FFX+21] and train a linear classifier for 60 epochs on top of the frozen encoder with a batch size of 512. We use the SGD optimizer with a momentum of 0.9 and without weight decay to reach the initial learning rate 2 that follows the linear scaling rule with the batch size set to 512. A linear warmup is applied during 35 epochs and then a cosine annealing scheduler. For training, we sample randomly a clip in the video and random crop to the size 224×224 after short scaling the video to 256. An horizontal flip is also applied with a probability of 0.5. For evaluation, we follow the standard evaluation protocol of [FFMH19] and sample 10 temporal clips with 3 different spatial crops of size 256×256 applied to each temporal clip to cover the whole video. The final prediction is the mean average of the predictions of the 30 clips sampled.

Finetuning evaluation protocol details. We follow [FFX+21] for finetuning on UCF101 and HMDB51. We finetune the whole pretrained network and perform supervised training on the 101 and 51 classes respectively for 200 epochs with dropout of probability 0.8 before classification. We use the SGD optimizer with a momentum of 0.9 and without weight decay to reach the initial learning rate 0.1 that follows the linear scaling rule with the batch size set to 64 and a cosine annealing scheduler without warmup. For training, we sample randomly a clip in the video and random crop to the size 224×224 after short scaling the video to 256. We apply color jittering with the *strong* augmentation parameters, defined in Tab. 1 in Tab. 4.1, and an horizontal flip with a probability of 0.5. For evaluation, we follow the 30-crops procedure as for linear evaluation. Specific hyperparameter search for each dataset might improve results.

Appendix C

Supplementary material COMEDIAN

C.1 Implementation details

C.1.1 Architectures

	ViViT Tiny	ViViT Small	ViSwin Tiny
Input dim	$C \times T_g \times H \times W$	$C \times T_g \times H \times W$	$C \times T_g \times H \times W$
Spatial encoder			
Input tokens dim	$\left(\frac{H}{16} \times \frac{W}{16} + 1\right) \times \frac{T_g}{2} \times 192$	$\left(\frac{H}{16} \times \frac{W}{16} + 1\right) \times \frac{T_g}{2} \times 384$	$\frac{H}{4} \times \frac{W}{4} \times \frac{T_g}{2} \times 96$
Num parameters	5.7M	22.0M	27.5M
GFLOPs	41.19	149.30	144.68
Temporal encoder			
Input tokens dim	$\frac{T_g}{2} \times 192$	$\frac{T_g}{2} \times 384$	$\frac{T_g}{2} \times 768$
Num parameters	1.8M	7.1M	28.4M
GFLOPs	0.06	0.24	0.96
Global model			
Num parameters	7.5M	29.1M	55.9M
GFLOPs	41.25	149.54	145.64

Table C.1: Comparison of the ViViT Tiny, ViViT small, and ViSwin Tiny spatial and temporal encoders and global model in terms of computational usage.

Encoders. For the spatial encoder, two different transformer architectures are used: ViT [DBK+21] and Swin [LLC+21]. By default, the temporal encoder is a stack of 4 attention layers as in ViT architecture. For ViT, the global architecture corresponds to the ViViT model 2 [ADH+21]. We keep this name and refer to the Swin based-architecture as ViSwin. In Tab. C.1, we provide the input dimension of tokens for the spatial and temporal encoders, and their number of parameters and GFLOPs for ViViT Tiny, ViViT Small, and ViSwin Tiny.

For all models, the majority of the computations are performed in the spatial encoder which sees a lot of tokens, and the temporal encoder computational cost is negligible. However, the number of parameters does not scale well with the output dimension of the spatial encoder, due to the self-attention mechanism, which is reflected in ViSwin Tiny. It has 4 times more temporal parameters than ViViT small but only 1.25 times more spatial parameters. However, as Swin has less reduced computational usage in comparison with ViT by design [LLC+21] it scales better to deeper spatial architectures.

C.1.2 Optimizers

We use the optimizer ADAMW for pretraining and finetuning with a weight decay of 0.05. The initial learning rate depends on the training step as well as the backbone as detailed below. However, the steps follow different linear scaling rules for an initial learning rate η:

- Step 1: $\eta_{scaled} = \eta \times \frac{batch\_size}{256}$
- Step 2 and 3: $\eta_{scaled} = \eta \times \frac{batch\_size}{256} \times \frac{T_g}{64}$ with T_g the number of global frames per video.

Step 1. The initial learning rate is $\eta = 5 \times 10^{-4}$ with 10 epochs of warmup and a cosine annealing scheduler is applied throughout training.

Step 2. The initial learning rate is $\eta = 0.002$ with 10 epochs warmup and cosine annealing scheduler that ends at $0.01 \times \eta$.

Step 3. The initial learning rate is $\eta = 5 \times 10^{-4}$ for ViViT and $\eta = 3 \times 10^{-4}$ for ViSwin that ends at $0.01 \times \eta$.

C.1.3 Spatial Pretraining.

To perform the pretraining of the spatial encoder, we follow practices introduced by ρMoCo [FFX+21] and SCE. More specifically we use a 3-layer Multi-Layer Perceptron (MLP) on top of the online and target encoders of hidden size 1024 and output size 256 that is discarded after this step. The online predictor is a 2 layers MLP with the same hidden and output size as the projectors. The data augmentation distributions are the standard contrastive ones used on images and the temperature applied is $\tau = 0.1$. The momentum buffer size is $65,536$. For data sampling, all sub-videos of 1 second, or 2 frames, are used. The model is trained for 100 epochs with a batch size of 1024.

C.1.4 Spatio-temporal pretraining.

To perform the pretraining of the spatio-temporal encoder, we follow practices introduced for SCE. More specifically we use a 3-layer MLP on top of the temporal encoder of hidden size 1024 and output size 512 to match the dimension of the Baidu [ZKC+21]

features. The projector is later discarded. For the SCE loss parameters we use $\tau = 0.1$, $\tau_m = 0.07$, $\lambda = 0.5$. The data augmentation used is the $strong_\gamma$ without cropping reported in Tab. 4.1. For data sampling, we randomly extract 150 videos of 32 seconds or 64 frames per game at each epoch. The batch size for 32-second videos at 2 FPS is 64. For longer clips, the batch size is inversely proportional to the length and number of windows. For example, for 64 seconds, the batch size is 32 and the number of windows sampled per match is 75.

C.1.5 Finetuning.

A linear classifier is applied to each output temporal token to perform fine-tuning. Each video sampled is augmented by using color jittering with probability 0.8 and of strength ± 0.4 on brightness, contrast, and saturation and 0 for hue to avoid changing the color of cards. Random Gaussian blur is also applied with probability 0.5 and a kernel size of 23 with $\sigma \in [0.1, 2.]$. A horizontal flip of probability 0.5 is also applied followed by a mixup [ZCDL18] whose mixing coefficient is sampled by a Beta law $\mathscr{B}(0.1, 0.1)$.

The classifier is first trained during 30 epochs for its initialization and then the whole architecture is fine-tuned for 20 epochs for ViViT and 10 for ViSwin. The learning rate is reset for the second part.

For data sampling, 100 videos per match are uniformly sampled whilst enforcing that the beginning and end of each half are selected to avoid missing kickoffs and last-second actions. The batch size for 32-second videos at 2 FPS is 128. For longer clips, the batch size is inversely proportional to the length and number of windows. For example, for 64 seconds, the batch size is 64 and the number of windows sampled is 50 per match.

C.2 Inference hyper-parameters search

During inference, a sliding window with half overlap is applied on all videos. For multiple timestamp classifications, the maximum of predictions per action is kept. No data augmentation is applied. By default, a hard Non-Maximum Suppression (NMS) of a 5-second window is applied. The 6 first and last seconds of each window prediction are ignored to keep predictions that have past and future context.

In Tab. C.2, we study the effect of varying the number of seconds to ignore. Taking all predictions has the worst result of 66.4% t-AmAP showing that it is interesting to remove predictions on edge that do not have access to the context from the past or the future. The results increase up to 66.8% at 10 seconds and are stable for further seconds ignored. The increase in performance is relatively low and can be explained by the fact that the inference sliding window allows for some undetected predictions on edges to be retrieved by past or future windows.

Seconds	t-AmAP (%)
0	66.4
2	66.6
4	66.7
6	66.6
8	66.6
10	**66.8**
12	**66.8**
14	**66.8**
16	66.7

Table C.2: Influence of the number of seconds ignored at the start and end of each window prediction on the t-AmAP.

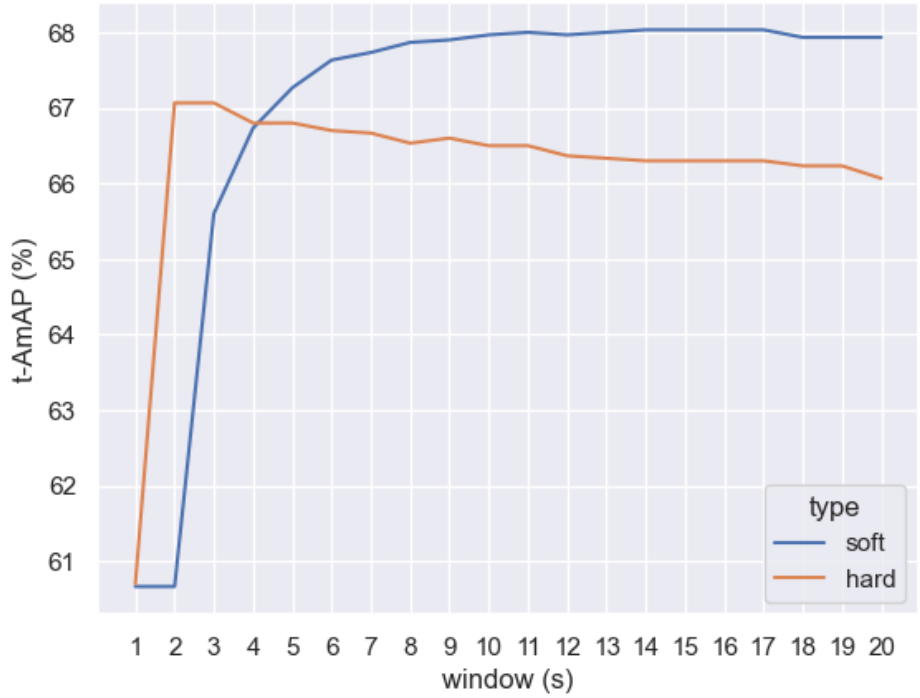


Figure C.1: Influence of the soft and hard NMS and its window size in second on the t-AmAP.

In Fig. C.1, we study the effect of using Hard or Soft NMS. As for [SSB22], we see an increase in using soft NMS over hard NMS. Depending on the NMS type the optimal temporal window size for NMS is not the same. The best results are achieved for a hard NMS with a 4-5 seconds window at 66.8% t-AmAP and 68.0% for a soft NMS with an 11-17 seconds window. The results show that not only does soft NMS perform better than hard NMS but is also more stable.

Bibliography

[ACM+22] Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Florian Bordes, Pascal Vincent, Armand Joulin, Mike Rabbat, and Nicolas Ballas. Masked siamese networks for label-efficient learning. In *European Conference on Computer Vision*, pages 456–473, 2022. 66, 67

[ADH+21] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lucic, and Cordelia Schmid. Vivit: A video vision transformer. In *International Conference on Computer Vision*, pages 6816–6826, 2021. 101, 102, 108, 110, XI

[ADM+23] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael G. Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Conference on Computer Vision and Pattern Recognition*, pages 15619–15629, 2023. 66

[AIm23] AImageLab. Soccer event spotting — aimagelab. `https://aimagelab.ing.unimore.it/imagelab/researchActivity.asp?idActivity=073`, 2023. [Online; accessed 31-July-2023]. 36, XLVI

[AMK+20] Humam Alwassel, Dhruv Mahajan, Bruno Korbar, Lorenzo Torresani, Bernard Ghanem, and Du Tran. Self-supervised learning by cross-modal audio-video clustering. In *Advances in Neural Information Processing Systems*, 2020. 48, 49, 61, 62, 89

[Anw23] Aqeel Anwar. What is average precision in object detection and localization algorithms and how to calculate it? — towards data science. `https://towardsdatascience.com/what-is-average-precision-in-object-detection-localization-algorithms-and-how-to-calculate-it-3f330efe697b`, 2023. [Online; accessed 31-July-2023]. 39, XLVI

[AR19] Zoheb Abai and Nishad Rajmalwar. Densenet models for tiny imagenet classification. *arXiv*, abs/1904.10429, 2019. 80

[ARV20] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *International Conference on Learning Representation*, 2020. 30, 48

[BBG+23] Florian Bordes, Randall Balestriero, Quentin Garrido, Adrien Bardes, and Pascal Vincent. Guillotine regularization: Why removing layers is needed to improve generalization in self-supervised learning. *Transactions on Machine Learning Research*, 2023. 53, 56

[BCV13] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013. 27, 42

[BDPW22] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: BERT pre-training of image transformers. In *International Conference on Learning Representations*, 2022. 63, 65, 67, XLVIII

[Bel66] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966. 53

[BEL+20] Sagie Benaim, Ariel Ephrat, Oran Lang, Inbar Mosseri, William T. Freeman, Michael Rubinstein, Michal Irani, and Tali Dekel. Speednet: Learning the speediness in videos. In *Conference on Computer Vision and Pattern Recognition*, pages 9919–9928, 2020. 45, 46, 96

[BGG14] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 - mining discriminative components with random forests. In *European Conference on Computer Vision*, pages 446–461, 2014. 83

[BIS+23] Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Grégoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A cookbook of self-supervised learning. *arXiv*, abs/2304.12210, 2023. 53, 56

[BMAZ22] Roman Bachmann, David Mizrahi, Andrei Atanov, and Amir Zamir. Multimae: Multi-modal multi-task masked autoencoders. In *European Conference on Computer Vision*, volume 13697, pages 348–367, 2022. 66

[BPL22] Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations*, 2022. 59, 81

[BPS+21] Adrian Bulat, Juan-Manuel Pérez-Rúa, Swathikiran Sudhakaran, Brais Martínez, and Georgios Tzimiropoulos. Space-time mixing attention for video transformer. In *Advances in Neural Information Processing Systems*, pages 19594–19607, 2021. 102

[BWT21] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *International Conference on Machine Learning*, pages 813–824, 2021. 101, 102

[C+47] Augustin Cauchy et al. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847. 18

[CBJD18] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision*, 2018. 30, 48, XLVII

[CCL+22] Shimin Chen, Chen Chen, Wei Li, Xunqiang Tao, and Yandong Guo. Fastertad: Towards temporal action detection with proposal generation and classification in a unified network. *arXiv*, abs/2204.02674, 2022. 103, 114

[CDG+20] Anthony Cioppa, Adrien Deliège, Silvio Giancola, Bernard Ghanem, Marc Van Droogenbroeck, Rikke Gade, and Thomas B. Moeslund. A context-aware loss function for action spotting in soccer videos. In *Conference on Computer Vision and Pattern Recognition*, pages 13123–13133, 2020. 103

[CFGH20] Xinlei Chen, Haoqi Fan, Ross B. Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv*, abs/2003.04297, 2020. 53, 70, 71, 72, 77, 80, 81, 83, 84, VI, XLVIII, LII

[CFSM20] Tiffany Tianhui Cai, Jonathan Frankle, David J. Schwab, and Ari S. Morcos. Are all negatives created equal in contrastive instance discrimination? *arXiv*, abs/2010.06682, 2020. 32, 54, 71

[CGS+23] Anthony Cioppa, Silvio Giancola, Vladimir Somers, Floriane Magera, Xin Zhou, Hassan Mkhallati, Adrien Deliège, Jan Held, Carlos Hinojosa, Amir M. Mansourian, Pierre Miralles, Olivier Barnich, Christophe De Vleeschouwer, Alexandre Alahi, Bernard Ghanem, Marc Van Droogenbroeck, Abdullah Kamal, Adrien Maglo, Albert Clapés, Amr Abdelaziz, Artur Xarles, Astrid Orcesi, Atom Scott, Bin Liu, Byoungkwon Lim, Chen Chen, Fabian Deuser, Feng Yan, Fufu Yu, Gal Shitrit, Guanshuo Wang, Gyusik Choi, Hankyul Kim, Hao Guo, Hasby Fahrudin, Hidenari Koguchi, Håkan Ardö, Ibrahim Salah, Ido Yerushalmy, Iftikar Muhammad, Ikuma

Uchida, Ishay Be'ery, Jaonary Rabarisoa, Jeongae Lee, Jiajun Fu, Jianqin Yin, Jinghang Xu, Jongho Nang, Julien Denize, Junjie Li, Junpei Zhang, Juntae Kim, Kamil Synowiec, Kenji Kobayashi, Kexin Zhang, Konrad Habel, Kota Nakajima, Licheng Jiao, Lin Ma, Lizhi Wang, Luping Wang, Menglong Li, Mengying Zhou, Mohamed Nasr, Mohamed Abdelwahed, Mykola Liashuha, Nikolay Falaleev, Norbert Oswald, Qiong Jia, Quoc-Cuong Pham, Ran Song, Romain Hérault, Rui Peng, Ruilong Chen, Ruixuan Liu, Ruslan Baikulov, Ryuto Fukushima, Sergio Escalera, Seungcheon Lee, Shimin Chen, Shouhong Ding, Taiga Someya, Thomas B. Moeslund, Tianjiao Li, Wei Shen, Wei Zhang, Wei Li, Wei Dai, Weixin Luo, Wending Zhao, Wenjie Zhang, Xinquan Yang, Yanbiao Ma, Yeeun Joo, Yingsen Zeng, Yiyang Gan, Yongqiang Zhu, Yujie Zhong, Zheng Ruan, Zhiheng Li, Zhijian Huang, and Ziyu Meng. Soccernet 2023 challenges results. *arXiv*, abs/2309.06006, 2023. 10, 114, 115

[CH21] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021. 58, 79, 81

[CHH+21] Peihao Chen, Deng Huang, Dongliang He, Xiang Long, Runhao Zeng, Shilei Wen, Mingkui Tan, and Chuang Gan. Rspnet: Relative speed perception for unsupervised video representation learning. In *33rd Conference on Innovative Applications of Artificial Intelligence*, pages 1045–1053, 2021. 46, 88, 96, 97

[CKNH20] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, pages 1597–1607, 2020. 31, 32, 51, 52, 53, 56, 70, 71, 78, 81, 83, 84, VI, XLVII, LI, LII

[CKS+20] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E. Hinton. Big self-supervised models are strong semi-supervised learners. In *Advances in Neural Information Processing Systems*, 2020. 53

[CLL21] Ting Chen, Calvin Luo, and Lala Li. Intriguing properties of contrastive losses. pages 11834–11845, 2021. 55

[CMK+14] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Conference on Computer Vision and Pattern Recognition*, pages 3606–3613, 2014. 83

[CMM+20] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrast-

ing cluster assignments. In *Advances in Neural Information Processing Systems*, 2020. 32, 57, 60, 66, 81, 82, 83

[CN12] Adam Coates and Andrew Y. Ng. Learning feature representations with k-means. In *Neural Networks: Tricks of the Trade - Second Edition*, volume 7700, pages 561–580. 2012. 48

[CNL11] Adam Coates, Andrew Y. Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, pages 215–223, 2011. 80

[CRC+20] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International Conference on Machine Learning*, pages 1691–1703, 2020. 62

[CRD+21] Brian Chen, Andrew Rouditchenko, Kevin Duarte, Hilde Kuehne, Samuel Thomas, Angie W. Boggust, Rameswar Panda, Brian Kingsbury, Rogério Feris, David Harwath, James R. Glass, Michael Picheny, and Shih-Fu Chang. Multimodal clustering networks for self-supervised learning from unlabeled videos. In *International Conference on Computer Vision*, pages 7992–8001, 2021. 49

[CRL+20] Ching-Yao Chuang, Joshua Robinson, Yen-Chen Lin, Antonio Torralba, and Stefanie Jegelka. Debiased contrastive learning. In *Advances in Neural Information Processing Systems*, 2020. 54, 71

[CSH+22] Yen-Jui Chu, Jheng-Wei Su, Kai-Wen Hsiao, Chi-Yu Lien, Shu-Ho Fan, Min-Chun Hu, Ruen-Rone Lee, Chih-Yuan Yao, and Hung-Kuo Chu. Sports field registration via keypoints-aware label condition. In *Conference on Computer Vision and Pattern Recognition Workshops*, pages 3522–3529, 2022. 84, 85, LII

[CSSB10] Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11:1109–1135, 2010. 49

[CTM+21] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *International Conference on Computer Vision*, pages 6706–6716, 2021. 7, 29, 32, 58, 66, 82, 101, 102, XLVII

[Cut13] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, pages 2292–2300, 2013. 48

[CWLC22] Minghao Chen, Fangyun Wei, Chong Li, and Deng Cai. Frame-wise action representations for long videos via sequence contrastive learning. In *Conference on Computer Vision and Pattern Recognition*, pages 13801–13810, 2022. 61, 62, 103

[CXH21] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *International Conference on Computer Vision*, pages 9620–9629, 2021. 29, 53, 56, 71, 74, 81, 102, VII

[CYZ+22] Mengqi Cao, Min Yang, Guozhen Zhang, Xiaotian Li, Yilu Wu, Gangshan Wu, and Limin Wang. Spotformer: A transformer-based framework for precise soccer action spotting. In *International Workshop on Multimedia Signal Processing*, pages 1–6, 2022. 103, 114

[CZ17] João Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In *Conference on Computer Vision and Pattern Recognition*, pages 4724–4733, 2017. 23

[CZM+22] Huseyin Coskun, Alireza Zareian, Joshua L. Moore, Federico Tombari, and Chen Wang. GOCA: guided online cluster assignment for self-supervised video representation learning. In *European Conference on Computer Vision*, pages 1–22, 2022. 48, 61

[DAT+21] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In *2021 International Conference on Computer Vision*, pages 9568–9577, 2021. 55, 56, 81, 82, 83, LII

[DBK+21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 7, 24, 25, 26, 62, 84, 100, 102, 108, 109, VIII, XI, XLVI

[DBZ+23] Xiaoyi Dong, Jianmin Bao, Ting Zhang, Dongdong Chen, Weiming Zhang, Lu Yuan, Dong Chen, Fang Wen, Nenghai Yu, and Baining Guo. Peco: Perceptual codebook for bert pre-training of vision transformers. In *AAAI Conference on Artificial Intelligence*, pages 552–560, 2023. 63

[DCG+21] Adrien Deliège, Anthony Cioppa, Silvio Giancola, Meisam Jamshidi Seikavandi, Jacob V. Dueholm, Kamal Nasrollahi, Bernard Ghanem, Thomas B. Moeslund, and Marc Van Droogenbroeck. Soccernet-v2: A dataset and benchmarks for holistic understanding of broadcast soccer videos. In

Conference on Computer Vision and Pattern Recognition Workshops, pages 4508–4519, 2021. 84, 85, 101, 103, 108, LII

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, 2019. 7, 32, 62

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Society Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 21, 64, 77, 85, LII

[Den23a] Julien Denize. GitHub - CEA-LIST/SCE: Implementation of "Similarity Contrastive Estimation for Self-Supervised Soft Contrastive Learning" WACV 2023. — github.com. `https://github.com/CEA-LIST/SCE`, 2023. [Accessed 17-10-2023]. 10

[Den23b] Julien Denize. GitHub - juliendenize/eztorch: Library to perform image and video self-supervised learning. — github.com. `https://github.com/juliendenize/eztorch`, 2023. [Accessed 17-10-2023]. 10, 119

[Den23c] Julien Denize. GitHub - juliendenize/torchaug: Library to perform efficient vision data augmentations for CPU/GPU per-sample/batched data. — github.com. `https://github.com/juliendenize/torchaug`, 2023. [Accessed 17-10-2023]. 10

[DFS+16] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin A. Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *Transactions on Pattern Analysis and Machine Intelligence*, pages 1734–1747, 2016. 44

[DGE15] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. In *International Conference on Computer Vision*, pages 1422–1430, 2015. 43, 44, 84

[DGRS22] Ishan R. Dave, Rohit Gupta, Mamshad Nayeem Rizve, and Mubarak Shah. TCLR: temporal contrastive learning for video representation. *Computer Vision and Image Understanding*, page 103406, 2022. 60, 89, 97

[DLR+24] Julien Denize, Mykola Liashuha, Jaonary Rabarisoa, Astrid Orcesi, and Romain Hérault. COMEDIAN: self-supervised learning and knowledge distillation for action spotting using transformers. *Winter Conference on Applications of Computer Vision Workshops*, 2024. 9, 101, 115

[DLY+22] Shuangrui Ding, Maomao Li, Tianyu Yang, Rui Qian, Haohang Xu, Qingyi Chen, Jue Wang, and Hongkai Xiong. Motion-aware contrastive video representation learning via foreground-background merging. In *Conference on Computer Vision and Pattern Recognition*, pages 9716–9726, 2022. 61, 62

[DRO+22] Julien Denize, Jaonary Rabarisoa, Astrid Orcesi, Romain Hérault, and Stéphane Canu. Estimation contrastive de la similarité pour un apprentissage flou auto-supervisé. *Conférence sur l'Apprentissage automatique*, 2022. 9

[DRO+23] Julien Denize, Jaonary Rabarisoa, Astrid Orcesi, Romain Hérault, and Stéphane Canu. Similarity contrastive estimation for self-supervised soft contrastive learning. In *Winter Conference on Applications of Computer Vision*, pages 2705–2715, 2023. 9, 70, 86

[DROH23] Julien Denize, Jaonary Rabarisoa, Astrid Orcesi, and Romain Hérault. Similarity contrastive estimation for image and video soft contrastive self-supervised learning. *Machine Vision and Applications*, 34(6), 2023. 9, 88, 98

[DS22] Abdulrahman Darwish and Tallal El Shabrway. STE: spatio-temporal encoder for action spotting in soccer videos. In *International ACM Workshop on Multimedia Content Analysis in Sports*, pages 87–92, 2022. 103

[DZCL22] Haodong Duan, Nanxuan Zhao, Kai Chen, and Dahua Lin. Transrank: Self-supervised video representation learning via ranking-based transformation recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 2990–3000, 2022. 46, 92, 95, 96, 97

[EGH21] Linus Ericsson, Henry Gouk, and Timothy M. Hospedales. How well do self-supervised models transfer? In *Conference on Computer Vision and Pattern Recognition*, pages 5414–5423, 2021. 83

[EGW+10] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, pages 303–338, 2010. 83

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996. 47

[ESSS21] Aleksandr Ermolov, Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening for self-supervised representation learning. In *International Conference on Machine Learning*, pages 3015–3024, 2021. 58

[FA20] Abe Fetterman and Josh Albrecht. Understanding self-supervised and contrastive learning with "Bootstrap Your Own Latent" (BYOL) — generallyintelligent.com. `https://generallyintelligent.com/research/2020-08-24-understanding-self-supervised-contrastive-learning/`, 2020. [Accessed 18-08-2023]. 58

[Fei20] Christoph Feichtenhofer. X3D: expanding architectures for efficient video recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 200–210, 2020. 23

[FFLH22] Christoph Feichtenhofer, Haoqi Fan, Yanghao Li, and Kaiming He. Masked autoencoders as spatiotemporal learners. In *Advances in Neural Information Processing Systems*, 2022. 33, 64, 66

[FFMH19] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *International Conference on Computer Vision*, pages 6201–6210, 2019. 23, 24, 59, 90, 95, X, XLV

[FFP07] Li Fei-Fei, Robert Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, pages 59–70, 2007. 83

[FFX+21] Christoph Feichtenhofer, Haoqi Fan, Bo Xiong, Ross B. Girshick, and Kaiming He. A large-scale study on unsupervised spatiotemporal representation learning. In *Conference on Computer Vision and Pattern Recognition*, pages 3299–3309, 2021. 7, 59, 62, 88, 89, 90, 95, 96, 97, 98, X, XII, LIV

[FPZ16] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 1933–1941, 2016. 23

[FWW+21] Zhiyuan Fang, Jianfeng Wang, Lijuan Wang, Lei Zhang, Yezhou Yang, and Zicheng Liu. SEED: self-supervised distillation for visual representation. In *International Conference on Learning Representations*, 2021. 56, 102

[GADG18] Silvio Giancola, Mohieddine Amine, Tarek Dghaily, and Bernard Ghanem. Soccernet: A scalable dataset for action spotting in soccer videos. In *Con-*

ference on Computer Vision and Pattern Recognition, pages 1711–1721, 2018. 36, 100, 103

[GBC16] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. 16, 20, XLV

[GES+23] Rohit Girdhar, Alaaeldin El-Nouby, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. Omnimae: Single model masked pretraining on images and videos. In *Conference on Computer Vision and Pattern Recognition*, pages 10406–10417, 2023. 66

[GG21] Silvio Giancola and Bernard Ghanem. Temporally-aware feature pooling for action spotting in soccer broadcasts. In *Conference on Computer Vision and Pattern Recognition Workshops*, pages 4490–4499, 2021. 103, 114

[GH10] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010. 31, 49

[GKM+17] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fründ, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "something something" video database for learning and evaluating visual common sense. In *International Conference on Computer Vision*, pages 5843–5851, 2017. 94

[GSA+20] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent - A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, 2020. 32, 57, 66, 74, 78, 79, 81, 83, 84, LI, LII

[GSK18] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations*, 2018. 7, 30, 44, 84

[GSR+18] Chunhui Gu, Chen Sun, David A. Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, Cordelia Schmid, and Jitendra Malik. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *Conference on Computer Vision and Pattern Recognition*, pages 6047–6056, 2018. 94

[GZL+22] Yuting Gao, Jia-Xin Zhuang, Shaohui Lin, Hao Cheng, Xing Sun, Ke Li, and Chunhua Shen. Disco: Remedying self-supervised learning on lightweight models with distilled contrastive learning. In *European Conference on Computer Vision*, pages 237–253, 2022. 101, 102

[HAM+18] Ruisi He, Bo Ai, Andreas F Molisch, Gordon L Stuber, Qingyong Li, Zhangdui Zhong, and Jian Yu. Clustering enabled wireless channel modeling using big data algorithms. *IEEE Communications Magazine*, 56(5):177–183, 2018. 47, XLVI

[HCX+22] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. In *Conference on Computer Vision and Pattern Recognition*, pages 15979–15988, 2022. 33, 63, 64, 65, 66, 67, XLVIII

[HFW+20] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. In *Conference on Computer Vision and Pattern Recognition*, pages 9726–9735, 2020. 32, 53, 56, 57, 71, 84, 104, 109

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask r-cnn. In *International Conference on Computer Vision*, pages 2980–2988, 2017. 83, 84, LII

[HHOK21] Sukjun Hwang, Miran Heo, Seoung Wug Oh, and Seon Joo Kim. Video instance segmentation using inter-frame communication transformers. In *Advances in Neural Information Processing Systems*, pages 13352–13363, 2021. 102

[HKS18] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Conference on Computer Vision and Pattern Recognition*, pages 6546–6555, 2018. 23, 59, 90, 95, X

[HLVDMW17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 23

[HLW+21] Lianghua Huang, Yu Liu, Bin Wang, Pan Pan, Yinghui Xu, and Rong Jin. Self-supervised video representation learning by context and motion decoupling. In *Conference on Computer Vision and Pattern Recognition*, pages 13886–13895, 2021. 61, 103

[HR02] Geoffrey E Hinton and Sam Roweis. Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems*, volume 15, 2002. 119

[HSL+21] Kai Hu, Jie Shao, Yuan Liu, Bhiksha Raj, Marios Savvides, and Zhiqiang Shen. Contrast and order representations for video self-supervised learning. In *International Conference on Computer Vision*, pages 7919–7929, 2021. 61, 88, 89, 96

[HVD15] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *arXiv*, abs/1503.02531, 2015. 101, 102

[HWH+21] Deng Huang, Wenhao Wu, Weiwen Hu, Xu Liu, Dongliang He, Zhihua Wu, Xiangmiao Wu, Mingkui Tan, and Errui Ding. Ascnet: Self-supervised video representation learning with appearance-speed consistency. In *International Conference on Computer Vision*, pages 8076–8085, 2021. 46, 88

[HWHQ21] Qianjiang Hu, Xiao Wang, Wei Hu, and Guo-Jun Qi. Adco: Adversarial contrast for efficient learning of unsupervised representations from self-trained negative adversaries. In *Conference on Computer Vision and Pattern Recognition*, pages 1074–1083, 2021. 46, 55, 56, 81, 82, VII

[HXZ20a] Tengda Han, Weidi Xie, and Andrew Zisserman. Memory-augmented dense predictive coding for video representation learning. In *European Conference on Computer Vision*, pages 312–329, 2020. 59, 61, 62, 88, 89, 97

[HXZ20b] Tengda Han, Weidi Xie, and Andrew Zisserman. Self-supervised co-training for video representation learning. In *Advances in Neural Information Processing Systems*, 2020. 61, 89, 96

[HYZ+22] Lang Huang, Shan You, Mingkai Zheng, Fei Wang, Chen Qian, and Toshihiko Yamasaki. Green hierarchical vision transformer for masked image modeling. *Advances in Neural Information Processing Systems*, 35:19997–20010, 2022. 64

[HZC+17] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv*, abs/1704.04861, 2017. 23

[HZG+22] James Hong, Haotian Zhang, Michaël Gharbi, Matthew Fisher, and Kayvon Fatahalian. Spotting temporally precise, fine-grained events in video. In *European Conference on Computer Vision*, pages 33–51, 2022. 103, 114

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 23, 77, 81, VI, VII, XLV

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *32nd International Conference on Machine Learning*, pages 448–456, 2015. VI, VII, LIV

[JJ21] Simon Jenni and Hailin Jin. Time-equivariant contrastive video representation learning. In *International Conference on Computer Vision*, 2021. 46, 88, 96, 97

[JMF20] Simon Jenni, Givi Meishvili, and Paolo Favaro. Video representation learning by recognizing temporal transformations. In *European Conference on Computer Vision*, pages 425–442, 2020. 46

[JT18] Longlong Jing and Yingli Tian. Self-supervised spatiotemporal feature learning via video rotation prediction. *arXiv*, abs/1811.11387, 2018. 7, 45, 92, 96

[JW23] Taeuk Jang and Xiaoqian Wang. Difficulty-based sampling for debiased contrastive representation learning. In *Conference on Computer Vision and Pattern Recognition*, pages 24039–24048, 2023. 54

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv*, abs/1412.6980, 2014. 19

[KCK19] Dahun Kim, Donghyeon Cho, and In So Kweon. Self-supervised video representation learning with space-time cubic puzzles. In *31st Innovative Applications of Artificial Intelligence Conference*, pages 8545–8552, 2019. 45, 96

[KCS+17] Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *arXiv*, abs/1705.06950, 2017. 28, 35, 90, 94, 112

[KGP+22] Ioannis Kakogeorgiou, Spyros Gidaris, Bill Psomas, Yannis Avrithis, Andrei Bursuc, Konstantinos Karantzalos, and Nikos Komodakis. What to hide from your students: Attention-guided masked image modeling. In *European Conference on Computer Vision*, pages 300–318, 2022. 65, 103, XLVIII

[KH91] Anders Krogh and John A. Hertz. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, 1991. 16

[KH09] Alex Krizhevsky and G Hinton. Learning multiple layers of features from tiny images.(2009). *Cs.Toronto.Edu*, pages 1–58, 2009. 28, 80, 83

[KJG+11] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso A. Poggio, and Thomas Serre. HMDB: A large video database for human motion recognition. In *International Conference on Computer Vision*, pages 2556–2563, 2011. 90, 94

[KSDF13] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *International Conference on Computer Vision Workshops*, pages 554–561, 2013. 83

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012. 21, 22, XLV

[KSP+20] Yannis Kalantidis, Mert Bülent Sariyildiz, Noé Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning. In *Advances in Neural Information Processing Systems*, 2020. 54, 56

[KTP20] Soroush Abbasi Koohpayegani, Ajinkya Tejankar, and Hamed Pirsiavash. Compress: Self-supervised learning by compressing representations. In *Advances in Neural Information Processing Systems*, 2020. 56, 102

[KZZ+21] Haofei Kuang, Yi Zhu, Zhi Zhang, Xinyu Li, Joseph Tighe, Sören Schwertfeger, Cyrill Stachniss, and Mu Li. Video contrastive learning with global context. In *International Conference on Computer Vision*, pages 3195–3204, 2021. 60

[LAG+21] Kuang-Huei Lee, Anurag Arnab, Sergio Guadarrama, John Canny, and Ian Fischer. Compressive visual representations. *Advances in Neural Information Processing Systems*, pages 19538–19552, 2021. 55

[LBBH98] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 21, 22, XLV

[LCY+21] Zhaowen Li, Zhiyang Chen, Fan Yang, Wei Li, Yousong Zhu, Chaoyang Zhao, Rui Deng, Liwei Wu, Rui Zhao, Ming Tang, et al. Mst: Masked self-supervised transformer for visual representation. *Advances in Neural Information Processing Systems*, 34:13165–13176, 2021. 65

[LGY+22] Xiaotong Li, Yixiao Ge, Kun Yi, Zixuan Hu, Ying Shan, and Ling-Yu Duan. mc-beit: Multi-choice discretization for image bert pre-training. In *European Conference on Computer Vision*, pages 231–246, 2022. 63

[LH17] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. *International Conference on Learning Representations*, 2017. VI, VII

[LHL+21] Yunfan Li, Peng Hu, Jerry Zitao Liu, Dezhong Peng, Joey Tianyi Zhou, and Xi Peng. Contrastive clustering. In *Conference on Innovative Applications of Artificial Intelligence*, pages 8547–8555, 2021. 49

[LHSY17] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequences. In *International Conference on Computer Vision*, pages 667–676, 2017. 45

[LKC17] William Lotter, Gabriel Kreiman, and David D. Cox. Deep predictive coding networks for video prediction and unsupervised learning. In *International Conference on Learning Representations*, 2017. 45, 46

[LLC+21] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *International Conference on Computer Vision*, pages 9992–10002, 2021. 26, 64, 100, 102, 108, 109, XI, XII

[LLL+19] Tianwei Lin, Xiao Liu, Xin Li, Errui Ding, and Shilei Wen. BMN: boundary-matching network for temporal action proposal generation. In *International Conference on Computer Vision*, 2019. 36

[LLY+21] Zefan Li, Chenxi Liu, Alan Yuille, Bingbing Ni, Wenjun Zhang, and Wen Gao. Progressive stage-wise learning for unsupervised feature representation enhancement. In *Conference on Computer Vision and Pattern Recognition*, pages 9767–9776, 2021. 44, 46

[LLZ+20] Dezhao Luo, Chang Liu, Yu Zhou, Dongbao Yang, Can Ma, Qixiang Ye, and Weiping Wang. Video cloze procedure for self-supervised spatio-temporal learning. In *Innovative Applications of Artificial Intelligence Conference*, pages 11701–11708, 2020. 46

[LMB+14] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *European Conference on Computer Vision*, pages 740–755, 2014. 83, 84, LII

[LMS16] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *European Conference on Computer Vision*, pages 577–593, 2016. 43, 44

[LNC+22] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. In *Conference on Computer Vision and Pattern Recognition*, pages 3192–3201, 2022. 101, 102

[LRO+20] Guillaume Lorre, Jaonary Rabarisoa, Astrid Orcesi, Samia Ainouz, and Stéphane Canu. Temporal contrastive pretraining for video action recog-

nition. In *Winter Conference on Applications of Computer Vision*, pages 651–659, 2020. 59, 61, 62, 89, 92

[LSZU16] Renjie Liao, Alex Schwing, Richard Zemel, and Raquel Urtasun. Learning deep parsimonious representations. In *Advances in Neural Information Processing Systems*, 2016. 48

[LWYY22] Xiang Li, Wenhai Wang, Lingfeng Yang, and Jian Yang. Uniform masking: Enabling MAE pre-training for pyramid-based vision transformers with locality. *arXiv*, abs/2205.10063, 2022. 64

[LZS17] Tianwei Lin, Xu Zhao, and Zheng Shou. Single shot temporal action detection. In *ACM on Multimedia Conference*, pages 988–996, 2017. 36

[LZS+21] Kibok Lee, Yian Zhu, Kihyuk Sohn, Chun-Liang Li, Jinwoo Shin, and Honglak Lee. i-mix: A strategy for regularizing contrastive representation learning. In *International Conference on Learning Representations*, 2021. 55, 56

[LZXH21] Junnan Li, Pan Zhou, Caiming Xiong, and Steven C. H. Hoi. Prototypical contrastive learning of unsupervised representations. In *International Conference on Learning Representations*, 2021. 49, 55, 56, 71, 83

[M+67] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Berkeley symposium on mathematical statistics and probability*, pages 281–297, 1967. 28, 46

[MAR21] Behzad Mahaseni, Erma Rahayu Mohd Faizal Abdullah, and Ram Gopal Raj. Spotting football events using two-stream convolutional neural network and dilated recurrent neural network. *IEEE Access*, pages 61929–61942, 2021. 103

[MAS+20] Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. End-to-end learning of visual representations from uncurated instructional videos. In *Conference on Computer Vision and Pattern Recognition*, pages 9876–9886, 2020. 61, 62, 89

[MHY+21] Hiroaki Minoura, Tsubasa Hirakawa, Takayoshi Yamashita, Hironobu Fujiyoshi, Mitsuru Nakazawa, Yeongnam Chae, and Björn Stenger. Action spotting and temporal attention analysis in soccer videos. In *International Conference on Machine Vision and Applications*, pages 1–6, 2021. 103

[Mit97] Tom M Mitchell. *Machine learning*. 1997. 12

[MKLF22] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixé, and Christoph Feichtenhofer. Trackformer: Multi-object tracking with transformers. In

Conference on Computer Vision and Pattern Recognition, pages 8834–8844, 2022. 101, 102

[MMW+21] Jovana Mitrovic, Brian McWilliams, Jacob C. Walker, Lars Holger Buesing, and Charles Blundell. Representation learning via invariant causal mechanisms. In *International Conference on Learning Representations*, 2021. 55, 56, 62

[MODP23] Adrien Maglo, Astrid Orcesi, Julien Denize, and Quoc Cuong Pham. Individual locating of soccer players from a single moving view. *Sensors*, 23(18), 2023. 9, 70, 84, 86

[MRK+13] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew B. Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv*, abs/1306.5151, 2013. 83

[Mur91] Fionn Murtagh. Multilayer perceptrons for classification and regression. *Neurocomputing*, 2(5-6):183–197, 1991. 17

[MvdM20] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Conference on Computer Vision and Pattern Recognition*, pages 6706–6716, 2020. 44

[MZH16] 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, 2016. 30, 45, XLVI

[NBZA21] Daniel Neimark, Omri Bar, Maya Zohar, and Dotan Asselmann. Video transformer network. In *International Conference on Computer Vision Workshops*, pages 3156–3165, 2021. 101, 102

[NF16] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84, 2016. 30, 43, 44, XLVI

[NPF17] Mehdi Noroozi, Hamed Pirsiavash, and Paolo Favaro. Representation learning by learning to count. In *International Conference on Computer Vision*, pages 5899–5907, 2017. 44

[NRSH+21] Olav Andre Nergård Rongved, Markus Stige, Steven Alexander Hicks, Vajira Lasantha Thambawita, Cise Midoglu, Evi Zouganeli, Dag Johansen, Michael Alexander Riegler, and Pål Halvorsen. Automated event detection and classification in soccer: The potential of using multiple modalities. *Machine Learning and Knowledge Extraction*, 3(4):1030–1054, 2021. 100, XLIX

[NZ08] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729, 2008. 83

[NZQ+22] Jingcheng Ni, Nan Zhou, Jie Qin, Qian Wu, Junqi Liu, Boxun Li, and Di Huang. Motion sensitive contrastive learning for self-supervised video representation. In *European Conference on Computer Vision*, pages 457–474, 2022. 46, 61

[ODM+23] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael G. Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *arXiv*, abs/2304.07193, 2023. 7, 66, 67

[ope23] Intersection over Union (IoU) in Object Detection and Segmentation — learnopencv.com. `https://learnopencv.com/intersection-over-union-iou-in-object-detection-and-segmentation/`, 2023. [Accessed 12-09-2023]. 38, XLVI

[PAR20] A. J. Piergiovanni, Anelia Angelova, and Michael S. Ryoo. Evolving losses for unsupervised video representation learning. In *Conference on Computer Vision and Pattern Recognition*, pages 130–139, 2020. 46, 61, 62, 88, 89

[Pea01] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901. 28

[PKD+16] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016. 43, 44

[PKLC19] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *Conference on Computer Vision and Pattern Recognition*, pages 3967–3976, 2019. 56, 102

[PLKS22] Jungin Park, Jiyoung Lee, Ig-Jae Kim, and Kwanghoon Sohn. Probabilistic representations for video contrastive learning. In *Conference on Computer*

Vision and Pattern Recognition, pages 14691–14701, 2022. 60, 62, 96, 97, 98

[PSY+21] Tian Pan, Yibing Song, Tianyu Yang, Wenhao Jiang, and Wei Liu. Videomoco: Contrastive video representation learning with temporally adversarial examples. In *Conference on Computer Vision and Pattern Recognition*, pages 11205–11214, 2021. 60, 88, 89, 96

[PVZJ12] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *Conference on Computer Vision and Pattern Recognition*, pages 3498–3505, 2012. 83

[PWZ+22] Xiangyu Peng, Kai Wang, Zheng Zhu, Mang Wang, and Yang You. Crafting better contrastive views for siamese representation learning. In *Conference on Computer Vision and Pattern Recognition*, pages 16010–16019, 2022. 53

[PZN+22] Trung Pham, Chaoning Zhang, Axi Niu, Kang Zhang, and Chang D. Yoo. On the pros and cons of momentum encoder in self-supervised visual representation learning. *arXiv*, abs/2208.05744, 2022. 58

[QDLL22] Rui Qian, Shuangrui Ding, Xian Liu, and Dahua Lin. Static and dynamic concepts for self-supervised video representation learning. In *European Conference on Computer Vision*, pages 145–164. Springer, 2022. 103

[QLL+21] Rui Qian, Yuxi Li, Huabin Liu, John See, Shuangrui Ding, Xian Liu, Dian Li, and Weiyao Lin. Enhancing self-supervised video representation learning via multi-level feature optimization. In *International Conference on Computer Vision*, pages 7970–7981, 2021. 60, 62, 97

[QMG+21] Rui Qian, Tianjian Meng, Boqing Gong, Ming-Hsuan Yang, Huisheng Wang, Serge J. Belongie, and Yin Cui. Spatiotemporal contrastive video representation learning. In *Conference on Computer Vision and Pattern Recognition*, pages 6964–6974, 2021. 59, 62, 88, 89, 96

[QYM17] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *International Conference on Computer Vision*, pages 5534–5542, 2017. 23

[RCSJ21] Joshua David Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. In *International Conference on Learning Representations*, 2021. 54, 71

[RDGF16] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *onference on Computer Vision and Pattern Recognition*, pages 779–788, 2016. 34

[RGA+20] Pierre H. Richemond, Jean-Bastien Grill, Florent Altché, Corentin Tallec, Florian Strub, Andrew Brock, Samuel L. Smith, Soham De, Razvan Pascanu, Bilal Piot, and Michal Valko. BYOL works even without batch statistics. *arXiv*, abs/2010.10241, 2020. 58

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 2015. 34

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. 18

[RLA+21] Adrià Recasens, Pauline Luc, Jean-Baptiste Alayrac, Luyu Wang, Florian Strub, Corentin Tallec, Mateusz Malinowski, Viorica Patraaucean, Florent Altché, Michal Valko, Jean-Bastien Grill, Aäron van den Oord, and Andrew Zisserman. Broaden your views for self-supervised video learning. In *International Conference on Computer Vision*, pages 1235–1245, 2021. 29, 60, 61, 62, 89, 96, XLVIII

[RM51] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951. 19

[RNK+22] Kanchana Ranasinghe, Muzammal Naseer, Salman Khan, Fahad Shahbaz Khan, and Michael S. Ryoo. Self-supervised video transformer. In *Conference on Computer Vision and Pattern Recognition*, pages 2864–2874, 2022. 60

[RNS+18] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018. 32, 62, 63

[Rol17] Jason Tyler Rolfe. Discrete variational autoencoders. In *International Conference on Learning Representations*, 2017. 63

[RV19] Mathias Rieder and Richard Verbeet. Robot-human-learning for robotic picking processes. In *Artificial Intelligence and Digital Transformation in Supply Chain Management: Innovative Approaches for Supply Chains. Proceedings of the Hamburg International Conference of Logistics (HICL), Vol. 27*, pages 87–114. Berlin: epubli GmbH, 2019. 34, XLVI

[S+56] Hugo Steinhaus et al. Sur la division des corps matériels en parties. *Bulletin L Académie Polonaise des Science*, 1(804):801, 1956. 46

[SB58] Terence Sanger and Pallavi N. Baljekar. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, pages 386–408, 1958. 14

[SBMS19] Chen Sun, Fabien Baradel, Kevin Murphy, and Cordelia Schmid. Learning video representations using contrastive bidirectional transformer. *arXiv*, abs/1906.05743, 2019. 61, 62, 89

[SC11] Shiliang Sun and Qiaona Chen. Hierarchical distance metric learning for large margin nearest neighbor classification. *International Journal of Pattern Recognitition Artificial Intelligence*, 25(7):1073–1087, 2011. 49

[SCC+23] Xinyu Sun, Peihao Chen, Liangwei Chen, Changhao Li, Thomas H. Li, Mingkui Tan, and Chuang Gan. Masked motion encoding for self-supervised video representation learning. In *Conference on Computer Vision and Pattern Recognition*, pages 2235–2245, 2023. 64

[Sha48] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948. 51

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 22

[SJE+23] Javier Selva, Anders S. Johansen, Sergio Escalera, Kamal Nasrollahi, Thomas B. Moeslund, and Albert Clapés. Video transformers: A survey. *Transactions on Pattern Analysis and Machine Intelligence*, pages 1–20, 2023. 101

[SKAL23] Mert Bulent Sariyildiz, Yannis Kalantidis, Karteek Alahari, and Diane Larlus. No reason for no supervision: Improved generalization in supervised models. In *ICLR 2023-International Conference on Learning Representations*, pages 1–26, 2023. 29

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. 23

[SMDH13] Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, pages 1139–1147, 2013. VI

[SMY+22] Yuzhi Shi, Hiroaki Minoura, Takayoshi Yamashita, Tsubasa Hirakawa, Hironobu Fujiyoshi, Mitsuru Nakazawa, Yeongnam Chae, and Björn Stenger.

Action spotting in soccer videos using multiple scene encoders. In *International Conference on Pattern Recognition*, pages 3183–3189, 2022. 103

[SNTS21] Chen Sun, Arsha Nagrani, Yonglong Tian, and Cordelia Schmid. Composable augmentation encoding for video representation learning. In *International Conference on Computer Vision*, pages 8814–8824, 2021. 46, 96, 97

[SSB22] João V. B. Soares, Avijit Shah, and Topojoy Biswas. Temporally precise action spotting in soccer videos using dense detection anchors. In *International Conference on Image Processing*, pages 2796–2800, 2022. 103, 114, XIV

[SXJS16] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Conference on Computer Vision and Pattern Recognition*, pages 4004–4012, 2016. 50

[SZ14] 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. 23

[SZ15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. 2015. 22, XLV

[SZS12] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv*, abs/1212.0402, 2012. 90, 92, 94, XLIX

[TBC+20] Matteo Tomei, Lorenzo Baraldi, Simone Calderara, Simone Bronzin, and Rita Cucchiara. Rms-net: Regression and masking for soccer event spotting. In *International Conference on Pattern Recognition*, pages 7699–7706, 2020. 103, 114

[TCD+21] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357, 2021. 101, 102

[TCG21] Yuandong Tian, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning dynamics without contrastive pairs. In *International Conference on Machine Learning*, volume 139, pages 10268–10278, 2021. 57

[TD20] Zachary Teed and Jia Deng. RAFT: recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision*, pages 402–419, 2020. 92, XLIX

[TGSH22] Martine Toering, Ioannis Gatopoulos, Maarten Stol, and Vincent Tao Hu. Self-supervised video representation learning with cross-stream prototypical contrasting. In *Winter Conference on Applications of Computer Vision*, pages 846–856, 2022. 61, 62, 89, 97

[TKI20] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *European Conference on Computer Vision*, pages 776–794, 2020. 77, I

[TL19] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114, 2019. 23

[TSP+20] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? In *Advances in Neural Information Processing Systems*, 2020. 32, 53, 56, 62, 70

[TSWW22] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. VideoMAE: Masked autoencoders are data-efficient learners for self-supervised video pre-training. In *Advances in Neural Information Processing Systems*, 2022. 7, 64, 67

[TWT+18] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018. 23, 95

[TWY20] Li Tao, Xueting Wang, and Toshihiko Yamasaki. Self-supervised video representation learning using inter-intra contrastive framework. In *ACM International Conference on Multimedia*, pages 2193–2201, 2020. 60, 62

[TWY22] Li Tao, Xueting Wang, and Toshihiko Yamasaki. An improved inter-intra contrastive learning framework on self-supervised video representation. *Trans. Circuits Syst. Video Technol.*, 32(8):5266–5280, 2022. 60, 62

[TWZ+22] Chenxin Tao, Honghui Wang, Xizhou Zhu, Jiahua Dong, Shiji Song, Gao Huang, and Jifeng Dai. Exploring the equivalence of siamese self-supervised learning via A unified gradient framework. *Conference on Computer Vision and Pattern Recognition*, 2022. 81, 82

[TZB+22] Shixiang Tang, Feng Zhu, Lei Bai, Rui Zhao, and Wanli Ouyang. Relative contrastive loss for unsupervised representation learning. In *European Conference on Computer Vision*, pages 1–18, 2022. 49, 55, 56

[Vap91] Vladimir Vapnik. Principles of risk minimization for learning theory. *Advances in neural information processing systems*, 1991. 15

[VdMH08] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 28, 119

[vdOLV18] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv*, abs/1807.03748, 2018. 31, 50, 51, 70, 71, 102, XLVII

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, pages 5998–6008, 2017. 24, 25, 62, 102, XLV

[WBTT22] Jue Wang, Gedas Bertasius, Du Tran, and Lorenzo Torresani. Long-short temporal contrastive learning of video transformers. In *Conference on Computer Vision and Pattern Recognition*, pages 14010–14020, 2022. 60, 62

[WCW+22] Rui Wang, Dongdong Chen, Zuxuan Wu, Yinpeng Chen, Xiyang Dai, Mengchen Liu, Yu-Gang Jiang, Luowei Zhou, and Lu Yuan. BEVT: BERT pretraining of video transformers. In *Conference on Computer Vision and Pattern Recognition*, pages 14713–14723, 2022. 64, 67

[WCW+23] Rui Wang, Dongdong Chen, Zuxuan Wu, Yinpeng Chen, Xiyang Dai, Mengchen Liu, Lu Yuan, and Yu-Gang Jiang. Masked video distillation: Rethinking masked feature modeling for self-supervised video representation learning. In *Conference on Computer Vision and Pattern Recognition*, pages 6312–6322, 2023. 66

[WFX+22] Chen Wei, Haoqi Fan, Saining Xie, Chao-Yuan Wu, Alan L. Yuille, and Christoph Feichtenhofer. Masked feature prediction for self-supervised visual pre-training. In *Conference on Computer Vision and Pattern Recognition*, pages 14648–14658, 2022. 64, 67

[WGDL22] Haoqing Wang, Xun Guo, Zhi-Hong Deng, and Yan Lu. Rethinking minimal sufficient representation in contrastive learning. In *Conference on Computer Vision and Pattern Recognition*, pages 16020–16029, 2022. 53

[WGL+21] Jinpeng Wang, Yuting Gao, Ke Li, Yiqi Lin, Andy J. Ma, Hao Cheng, Pai Peng, Feiyue Huang, Rongrong Ji, and Xing Sun. Removing the background by adding the background: Towards background robust self-supervised video representation learning. In *Conference on Computer Vision and Pattern Recognition*, pages 11804–11813, 2021. 61, 62

[WI20] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, pages 9929–9939, 2020. 55

[Wik23] Wikipedia. Artificial intelligence — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Artificial%20intelligence&oldid=1167849562`, 2023. [Online; accessed 31-July-2023]. 6

[WJB+19] Jiangliu Wang, Jianbo Jiao, Linchao Bao, Shengfeng He, Yunhui Liu, and Wei Liu. Self-supervised spatio-temporal representation learning for videos by predicting motion and appearance statistics. In *Conference on Computer Vision and Pattern Recognition*, pages 4006–4015, 2019. 45, 46, XLVI

[WJB+22] Jiangliu Wang, Jianbo Jiao, Linchao Bao, Shengfeng He, Wei Liu, and Yunhui Liu. Self-supervised video representation learning by uncovering spatio-temporal statistics. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3791–3806, 2022. 45

[WJL20] Jiangliu Wang, Jianbo Jiao, and Yun-Hui Liu. Self-supervised video representation learning by pace prediction. In *European Conference on Computer Vision*, pages 504–521, 2020. 45, 46, 88

[WLHK22] Cheng-En Wu, Farley Lai, Yu Hen Hu, and Asim Kadav. Self-supervised video representation learning with cascade positive retrieval. In *Conference on Computer Vision and Pattern Recognition*, pages 4070–4079, 2022. 60, 62

[WLM+22] Chao-Yuan Wu, Yanghao Li, Karttikeya Mangalam, Haoqi Fan, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Memvit: Memory-augmented multiscale vision transformer for efficient long-term video recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 13577–13587, 2022. 101, 102

[WMZ+21] Mike Wu, Milan Mosse, Chengxu Zhuang, Daniel Yamins, and Noah D. Goodman. Conditional negative sampling for contrastive learning of visual representations. In *International Conference on Learning Representations*, 2021. 54, 55, 56, XLVII

[WWSY21] Chen Wei, Huiyu Wang, Wei Shen, and Alan L. Yuille. CO2: consistent contrast for unsupervised visual representation learning. In *International Conference on Learning Representations*, 2021. 54, 55, 56, 71

[WWW+21] Guangrun Wang, Keze Wang, Guangcong Wang, Philip H. S. Torr, and Liang Lin. Solving inefficiency of self-supervised representation learning. In *International Conference on Computer Vision*, pages 9485–9495, 2021. 54, 56, 81, 82, 84, LII

[WXYL18] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2018. 53, 56, 84

[XGF16] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International Conference on Machine Learning*, pages 478–487, 2016. 48

[XHE+10] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *Conference on Computer Vision and Pattern Recognition*, pages 3485–3492, 2010. 83

[XSH+18] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *European Conference on Computer Vision*, pages 318–335, 2018. 23, 90

[XTM22] Fanyi Xiao, Joseph Tighe, and Davide Modolo. Maclr: Motion-aware contrastive learning of representations for videos. In *European Conference on Computer Vision*, pages 353–370. Springer, 2022. 61, 62

[XXC+21] Mingze Xu, Yuanjun Xiong, Hao Chen, Xinyu Li, Wei Xia, Zhuowen Tu, and Stefano Soatto. Long short-term transformer for online action detection. In *Advances in Neural Information Processing Systems*, pages 1086–1099, 2021. 101, 102

[XXZ+19] Dejing Xu, Jun Xiao, Zhou Zhao, Jian Shao, Di Xie, and Yueting Zhuang. Self-supervised spatiotemporal learning via video clip order prediction. In *Conference on Computer Vision and Pattern Recognition*, pages 10334–10343, 2019. 45

[XZC+22] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. In *Conference on Computer Vision and Pattern Recognition*, pages 9653–9663, 2022. 64, 65, 67

[XZW+17] Yuanjun Xiong, Yue Zhao, Limin Wang, Dahua Lin, and Xiaoou Tang. A pursuit of temporal accuracy in general activity detection. *arXiv*, abs/1703.02716, 2017. 38, XLVI

[YDL+22] Jiewen Yang, Xingbo Dong, Liujun Liu, Chao Zhang, Jiajun Shen, and Dahai Yu. Recurring the transformer for video action recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 14043–14053, 2022. 101, 102

[YGG17a] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks, 2017. 19

[YGG17b] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv*, abs/1708.03888, 2017. VII

[YGL+23] Kun Yi, Yixiao Ge, Xiaotong Li, Shusheng Yang, Dian Li, Jianping Wu, Ying Shan, and Xiaohu Qie. Masked image modeling with denoising contrast. In *International Conference on Learning Representations*, 2023. 67

[YHH+22] Chun-Hsiao Yeh, Cheng-Yao Hong, Yen-Chi Hsu, Tyng-Luh Liu, Yubei Chen, and Yann LeCun. Decoupled contrastive learning. In *European Conference on Computer Vision*, pages 668–684, 2022. 55, 56, 76, 77

[YLH+21] Mouxing Yang, Yunfan Li, Zhenyu Huang, Zitao Liu, Peng Hu, and Xi Peng. Partially view-aligned representation learning with noise-robust contrastive loss. In *Conference on Computer Vision and Pattern Recognition*, pages 1134–1143, 2021. 55, 56, 71

[YLH+23] Mouxing Yang, Yunfan Li, Peng Hu, Jinfeng Bai, Jiancheng Lv, and Xi Peng. Robust multi-view clustering with incomplete information. *Transactions on Pattern Analysis and Machine Intelligence*, 45(1):1055–1069, 2023. 55, 56, 71

[YLR+19] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv*, abs/1904.00962, 2019. 19

[YPB16] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2016. 48

[YQC+22] Liangzhe Yuan, Rui Qian, Yin Cui, Boqing Gong, Florian Schroff, Ming-Hsuan Yang, Hartwig Adam, and Ting Liu. Contextualized spatio-temporal contrastive learning with self-supervision. In *Conference on Computer Vision and Pattern Recognition*, pages 13957–13966, 2022. 96

[YXDZ20] Ceyuan Yang, Yinghao Xu, Bo Dai, and Bolei Zhou. Video representation learning with visual tempo consistency. *arXiv*, abs/2006.15489, 2020. 60, 62, 96

[YZQ+21] Ting Yao, Yiheng Zhang, Zhaofan Qiu, Yingwei Pan, and Tao Mei. Seco: Exploring sequence supervision for unsupervised representation learning. In *Conference on Innovative Applications of Artificial Intelligence*, pages 10656–10664, 2021. 46

[ZCDL18] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. 54, 109, XIII

[ZDWW18] Da Zhang, Xiyang Dai, Xin Wang, and Yuan-Fang Wang. S3D: single shot multi-span detector via fully 3d convolutional networks. In *British Machine Vision Conference*, page 293, 2018. 59, 95

[ZHH+21] Yifan Zhang, Bryan Hooi, Dapeng Hu, Jian Liang, and Jiashi Feng. Unleashing the power of contrastive self-supervised visual models via contrast-regularized fine-tuning. In *Advances in Neural Information Processing Systems*, pages 29848–29860, 2021. 54

[ZIE16] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. In *European Conference on Computer Vision*, pages 649–666, 2016. 30, 32, 43, 44, XLVI

[ZJM+21] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pages 12310–12320, 2021. 58, 81

[ZKC+21] Xin Zhou, Le Kang, Zhiyu Cheng, Bo He, and Jingyu Xin. Feature combination meets attention: Baidu soccer embeddings and transformer based temporal detection. *arXiv*, abs/2106.14447, 2021. 103, 108, 112, 114, 115, XII

[ZLL+22] He Zhu, Junwei Liang, Chengzhi Lin, Jun Zhang, and Jianming Hu. A transformer-based system for action spotting in soccer videos. In *International ACM Workshop on Multimedia Content Analysis in Sports*, pages 103–109, 2022. 103, 114

[ZLZS23] Heng Zhang, Daqing Liu, Qi Zheng, and Bing Su. Modeling video as stochastic processes for fine-grained video representation learning. In *Conference on Computer Vision and Pattern Recognition*, pages 2225–2234, 2023. 61, 62, 103

[ZTH+22] Xiaosong Zhang, Yunjie Tian, Wei Huang, Qixiang Ye, Qi Dai, Lingxi Xie, and Qi Tian. Hivit: Hierarchical vision transformer meets masked image modeling. *arXiv*, abs/2205.14949, 2022. 64

[ZWL22] Chen-Lin Zhang, Jianxin Wu, and Yin Li. Actionformer: Localizing moments of actions with transformers. In *European Conference on Computer Vision*, pages 492–510, 2022. 103

[ZWW+22a] Benjia Zhou, Pichao Wang, Jun Wan, Yanyan Liang, Fan Wang, Du Zhang, Zhen Lei, Hao Li, and Rong Jin. Decoupling and recoupling spatiotemporal representation for rgb-d-based motion recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 20154–20163, 2022. 103

[ZWW+22b] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan L. Yuille, and Tao Kong. Image BERT pre-training with online tokenizer. In *International Conference on Learning Representations*, 2022. 65, 66, 67, XLVIII

[ZWY+21] Mingkai Zheng, Fei Wang, Shan You, Chen Qian, Changshui Zhang, Xiaogang Wang, and Chang Xu. Weakly supervised contrastive learning. In *International Conference on Computer Vision*, pages 10022–10031, 2021. 55, 56, 71, 81, 82, 83

[ZXL+20] Xiaohang Zhan, Jiahao Xie, Ziwei Liu, Yew-Soon Ong, and Chen Change Loy. Online deep clustering for unsupervised representation learning. In *Conference on Computer Vision and Pattern Recognition*, pages 6687–6696, 2020. 48

[ZYW+21] Mingkai Zheng, Shan You, Fei Wang, Chen Qian, Changshui Zhang, Xiaogang Wang, and Chang Xu. Ressl: Relational self-supervised learning with weak augmentation. In *Advances in Neural Information Processing Systems*, pages 2543–2555, 2021. 56, 70, 71, 72, 76, 77, 78, 79, 80, 81, 82, 83, VI, VIII, XLVIII, LI, LII

List of Figures

List of Tables

Self-supervised representation learning and applications to image and video analysis

Keywords: *Machine learning; Deep learning; Computer vision; Representation learning (artifical intelligence); Image processing; Video; Self-supervised learning (artificial intelligence); Contrastive*

Abstract

In this thesis, we develop approaches to perform self-supervised learning for image and video analysis. Self-supervised representation learning allows to pretrain neural networks to learn general concepts without labels before specializing in downstream tasks faster and with few annotations. We present three contributions to self-supervised image and video representation learning. First, we introduce the theoretical paradigm of soft contrastive learning and its practical implementation called Similarity Contrastive Estimation (SCE) connecting contrastive and relational learning for image representation. Second, SCE is extended to global temporal video representation learning. Lastly, we propose COMEDIAN a pipeline for local-temporal video representation learning for transformers. These contributions achieved state-of-the-art results on multiple benchmarks and led to several academic and technical published contributions.

Apprentissage auto-supervisé de représentation et applications à l'analyse d'images et de vidéos

Mots-clefs: *Apprentissage automatique; Apprentissage profond; Vision par ordinateur; Apprentissage de représentations (intelligence artificielle); Traitement d'images; Vidéo; Apprentissage auto-supervisé (intelligence artificielle); Contrastif*

Résumé

Dans cette thèse, nous développons des approches d'apprentissage auto-supervisé pour l'analyse d'images et de vidéos. L'apprentissage de représentation auto-supervisé permet de pré-entraîner les réseaux neuronaux à apprendre des concepts généraux sans annotations avant de les spécialiser plus rapidement à effectuer des tâches, et avec peu d'annotations. Nous présentons trois contributions à l'apprentissage auto-supervisé de représentations d'images et de vidéos. Premièrement, nous introduisons le paradigme théorique de l'apprentissage contrastif doux et sa mise en œuvre pratique appelée Estimation Contrastive de Similarité (SCE) qui relie l'apprentissage contrastif et relationnel pour la représentation d'images. Ensuite, SCE est étendue à l'apprentissage de représentation vidéo temporelle globale. Enfin, nous proposons COMEDIAN, un pipeline pour l'apprentissage de représentation vidéo locale-temporelle pour l'architecture transformer. Ces contributions ont conduit à des résultats de pointe sur de nombreux benchmarks et ont donné lieu à de multiples contributions académiques et techniques publiées.