ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Machine Learning for Computer Vision

MINING TECHNIQUES FOR SELF-SUPERVISED VISUAL CONTRASTIVE LEARNING

CANDIDATE

Jia Liang Zhou

SUPERVISOR

Prof. Samuele Salti

CO-SUPERVISOR

Andrea Lonza

Academic Year 2021-2022

Session 1st

Abstract

Unsupervised learning for computer vision has recently gained significant progress thanks to the self-supervised contrastive learning methods. Inspired from metric learning, the goal is learning good representations that are more robust and generalisable than the features learned in a supervised manner, when transferring on the downstream tasks. In this work we are going to see the most popular contrastive learning methods in self-supervised representation learning, with a strong focus on the mining techniques devised for picking more informative positive and negative samples. We also propose a new approach combining contrastive learning with an online clustering method in order to mine hard positive samples. We show that on fine-grained datasets, like the Stanford Dogs dataset with 120 dog breeds, it helps to learn more discriminative representations compared to other state-of-the-art contrastive methods.

Contents

List of Figures

1 Introduction

Supervised Learning is a well-established paradigm in machine learning and in artificial intelligence. Given a task with enough data and enough labels, supervised learning can solve it really well in general. Collecting manual labels is, however, expensive and hard to be scaled up. The amount of unlabelled data is much larger than the labelled data, so it is wasteful not to use them. The goal of **Unsupervised Learning** methods is to leverage these unlabelled data and more in particular, **Self-Supervised Learning**, a subset of the unsupervised methods, can create labels for free and can use them in a supervised learning fashion. This idea has been leveraged successfully in Natural Language Processing (NLP), where the default task is to predict the next word given the past sequence. Inspired by ImageNet[31] classification pre-training in computer vision, BERT[10] adds two additional auxiliary tasks, both relying on self-generated labels, achieving however greater performance benefits than vision classification pre-training.

In this dissertation thesis we are going to see on overview of the most popular self-supervised methods used in Computer Vision, with a focus on **Contrastive Learning** methods, which have shown promising results recently, reaching the supervised state-of-the-art techniques when finetuning with a small subset of labeled data. Contrastive learning relies on positive and negative examples, so positive and negative mining methodologies are analysed in order to study the effects of more careful selections. A new framework is also proposed in order to combine the best features of both Clustering-based methods and contrastive learning methods, following the effectiveness claimed in literature of explicitly performing hard negative mining.

1.1 Related work

For **supervised learning**, given a dataset X, for each data X_i in X, there is a corresponding **human-annotated** label Y_i. For the training data $D =$

$\{X_i\}_{i=0}^{N}$, we define the loss function as:

$$loss(D) = \min_{\theta} \frac{1}{N} \sum_{i=0}^{N} loss(X_i, Y_i)$$

Supervised learning methods are able to obtain state-of-the-art results on many computer vision applications with human-annotated labels, but data collection and annotation are usually expensive.

Representation learning aims to learn representations of the data that make it easier to extract useful information when building classifiers or other predictors. Before deep learning, this was mostly done through feature engineering, which is time-consuming and highly dependent on prior knowledge, requiring domain-specific expertise. Hand-crafted algorithms usually failed to extract and organize useful features from the data, especially from complex raw data. "*Good*" representations should make a subsequent task easier, or ideally, we should achieve high performances in more than one task starting from the same representation. The intermediary representations guided by the cross-entropy loss during a classification task are clustered in the embedding space in a way that they are easy to be linearly separable, arranging the representations in long and stretched clusters. In this space distances between elements of the same class can be larger than distances between elements of different classes. The problem arises if we want to reuse these intermediary representations for other applications, for example face recognition, where we throw away the classification head of the network and compute similarities between representation vectors in order to map a certain face to a certain person.

The idea behind **metric learning** is to guide the feature extractor using a specific loss at training time in order to favor a cluster structure of the embedding such that:

- the distance between faces of the same person (intra-class distance) is minimized
- the distance between faces of different people (inter-class distance) is maximized

The **contrastive loss**[8] comes in handy to achieve these desired properties, minimizing the distance between the embeddings if we have images of

the same person, while pushing the distance to be larger if we have images of different people. Contrastive loss takes as input a pair (x_i, x_j) with labels (y_i, y_j) and is computed as follows:

$$L_{cont}(x_i, x_j) = \begin{cases} ||f(x_i) - f(x_j)||_2^2 & \text{if } y_i == y_j \\ -||f(x_i) - f(x_j)||_2^2 & \text{if } y_i \neq y_j \end{cases}$$

The same intuition is used in contrastive learning for **self-supervised learning**, but since we do not have labels, we treat two augmented views of the same image as two different images of the same class. The most common loss is the **InfoNCE** loss, derived from the Noise Contrastive Estimation (**NCE**), where the idea is to run logistic regression to tell apart the target data from the noise. We do not compare directly the feature embeddings as in contrastive loss, but we perform categorical cross-entropy to identify the positive sample among a set of negative samples.

1.2 Structure of the thesis

Chapter 2 explores self-supervised learning and its main tasks used as pre-training phase, before describing the most popular downstream tasks in computer vision. We show the evaluation protocols for benchmarking these methods and we then focus on contrastive learning. The ideal qualities of good representations are presented according to recent works and we talk about the most common neural network architectures used in literature and how the temperature hyper-parameter affects the training. We go more in detail about the popular contrastive learning methods like SimCLR, MoCo and NNCLR, and before moving to the clustering-based techniques, we cover some methods leveraging the use of mining strategies both for positive and negative samples. At the end of the second chapter, we also mention non-contrastive methods that do not rely on negative examples and we report the results on ImageNet of the cited methods.

In **chapter 3** we explain the intuition behind the idea of performing positive and negative mining, detailing all the experimental settings of our approach. A brief description of the datasets used will be presented and the performance of the models used in literature will be compared against our approach.

Chapter 4 presents a conclusion of the experiments and summarizes all

the work done, followed by the **chapter 5** devoted to the future directions of the research.

2 Self-Supervised Learning in Computer Vision

Unsupervised learning refers to learning methods that do not need any human-annotated labels. This type of methods include fully unsupervised learning methods which do not need any labels at all, as well as self-supervised learning methods in which networks are trained with automatically generated pseudo labels, without involving human annotators.

Self-supervised learning lets us exploit a huge amount of labels that come with the data for free. The reason is clear: generating a dataset with curated labels is expensive but unlabeled data is being produced continuously. One way to leverage all this large amount of unlabeled data is to set the learning objective properly as to get supervision from the data itself.

2.1 Pretext tasks

The *self-supervised* task, also known as ***pretext task***, is guided by a supervised loss function. We usually do not care about the final performance of this invented task, because we are rather interested in learning good intermediary representations that carry high quality semantic and structural meanings, helpful for a variety of *downstream tasks*. The models trained on the pretext tasks are not ready for the evaluation on high-level vision tasks, called **downstream tasks**, but they are used as pre-trained models to be fine-tuned on the final tasks.

We can summarize the pretext tasks into five categories: generation-based, context-based, free semantic label-based, cross modal-based and contrastive learning-based.

Generation-based methods: this type of methods involves image generation where features are learned with tasks like image colorization [38], image inpainting [29], image generation with Generative Adversarial Networks (GANs) [39].

Context-based pretext tasks: some of these tasks employ the context features of images such as context similarity between image patches, for example clustering-based methods [3]. Other tasks are based on the spatial relations among image patches, such as jigsaw puzzle [28] and context prediction [11].

Free Semantic Label-based methods: the semantic labels are generated automatically by hard-coded algorithms [13] or by game engines [30]. The pretext tasks include moving object segmentation, contour detection and relative depth prediction.

Cross Modal-based methods: the ConvNets are trained to verify whether two different input channels are corresponding to each other, like Visual-Audio Correspondence Verification [23] or RGB-Flow Correspondence Verification [32].

Contrastive Learning-based methods: the main task of these methods is *Instance Discrimination*, where we learn a representation by maximising agreement of the encoded features (embeddings) between two differently augmented views of the same images, while simultaneously minimising the agreement between views generated from different images.

2.2 Downstream tasks

To assess the quality of the learned visual features by self-supervised methods, the learned parameters are employed as pre-trained models and then fine-tuned on *downstream tasks* such as image classification, semantic segmentation, object detection and action recognition. The performance of the transfer learning on these high-level vision tasks demonstrates the generalization ability of the learned features. If ConvNets of self-supervised learning can learn general features, then the pre-trained models can be used as a good starting point for other vision tasks that require capturing similar features from images or videos. Image classification, semantic segmentation, and object detection usually are used as the tasks to evaluate the generalization ability of the learned image features by self-supervised learning methods, while human action recognition in videos is used to evaluate the quality of video features obtained from self-supervised learning methods.

The most commonly used downstream tasks in computer vision for evaluation are:

- **Semantic Segmentation**: the task of assigning semantic labels to each

pixel of the image. When evaluating with the semantic segmentation downstream task, the FCN is initialized with the parameters trained with the self-supervised pretext task and then fine-tuned on the semantic segmentation dataset;

- **Object Detection**: the task of localizing the position of objects in images and recognizing the category of the detected objects. When using object detection as downstream task, networks that are trained with the pretext task on unlabeled large data are deployed as the pre-trained model for the Fast-RCNN[14];

- **Image Classification**: the task of recognizing the category of objects in a image, where usually only one class label is assigned to each image. When choosing image classification as a downstream task to evaluate the quality of image features learned from self-supervised learning methods, the self-supervised pre-trained model is applied to each image to extract features which then are used to train a classifier such as Support Vector Machine (**SVM**)[33].

2.3 Evaluation protocols

In order to evaluate the self-supervised methods, we need to establish common protocols for using the pre-trained weights for fine-tuning or transfer learning, usually done on the ImageNet dataset:

- ImageNet Linear Classification: after performing self-supervised pre-training, the weights of the network are frozen up to the features layer and a linear classifier (a fully-connected layer followed by softmax) is trained in a supervised fashion for 100 epochs;

- ImageNet Semi-supervised Learning: the network is fine-tuned with either 1% or 10% of ImageNet labeled images.

- Transfer Learning on different datasets: in order to show the effectiveness of the learned features it is useful to evaluate on multiple datasets as Food101[2], CIFAR10[25], CIFAR100[25], STL-10[1] and Cars[24].

2.4 Contrastive learning methods

Contrastive learning can be considered as learning by comparing. A representation is learned by comparing among the input samples, usually performing comparisons between positive pairs of "similar" images and negative pairs of "dissimilar" images. The goal of contrastive learning is simple: the representations of "similar" samples, or from the same class ideally, should be mapped close in the embedding space, while those of "dissimilar" samples should be further away. Contrasting between positive pairs will lead to representations that will be pulled closer together while contrasting negative pairs will push representations farther apart. The most popular pretext task for constrastive methods is the **Instance Discrimination** task, described in figure 2.1 and used for learning visual representations. Instance discrimination has succeeded in learning useful representations that achieve state-of-the-art results on transfer learning for some downstream tasks in computer vision.

2.4.1 Framework formulation

The majority of contrastive learning methods share these common components:

- Data augmentation module that transforms the given input data randomly to output two correlated but different views of the same example, in order to create a pair of positive samples (x_i^1, x_i^2). Generally strong augmentations tend to increase the quality of the learned representations
- A neural network *base encoder* $f(\cdot)$ that extracts representation vectors from augmented data samples. The choice of the architecture for the base encoder can be arbitrary, but it is common practice to use ResNet-50[18] architecture for simplicity
- A small neural network *projection head* $g(\cdot)$ that maps the representations to the space where the contrastive loss is applied. Usually it is implemented as a MLP with one hidden layer and using this additional head has proved to be more effective
- A *contrastive loss function* that guides the augmented views of the same samples to be close in the representations space while pushing away representations of different samples. The *NT-Xent* loss, one of the most

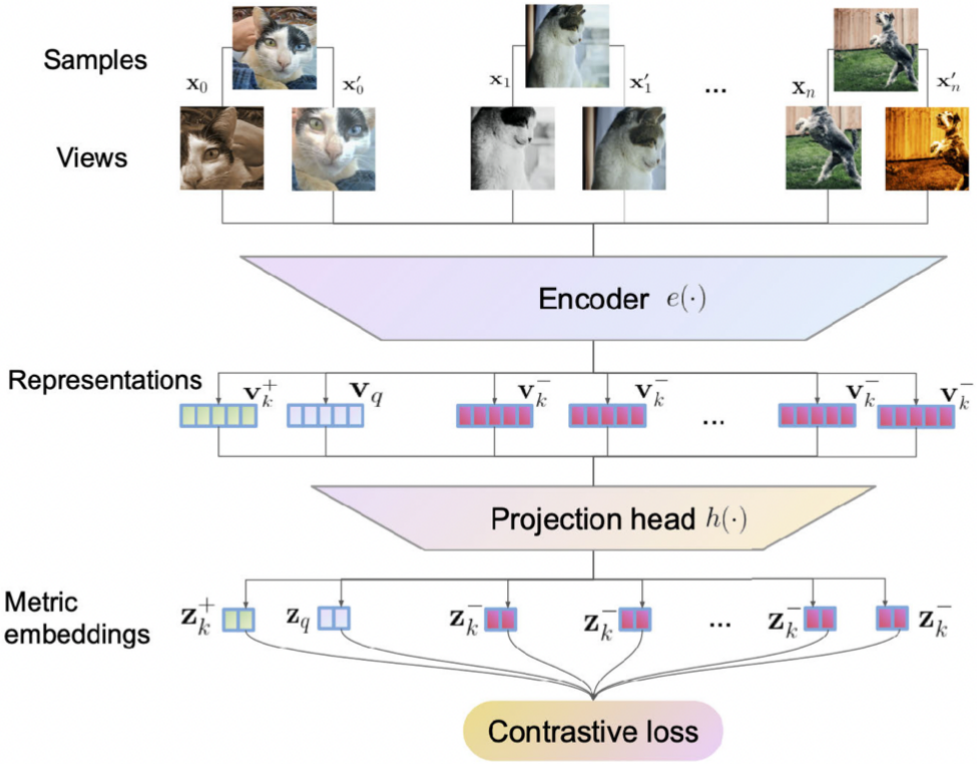

Figure 2.1: Contrastive learning in the Instance Discrimination pretext task for self-supervised visual representation learning. A positive pair is created from two randomly augmented views of the same image, while negative pairs are created from views of two different images. All views are encoded by the a shared encoder and projection heads before the representations are evaluated by the contrastive loss function[26].

used in literature, is a reformulation of a multi-class cross-entropy classification loss

2.4.2 Data augmentation

The most simple yet effective approach to generating different views of the same scene is to use a hand-crafted transformation function operating on the input data domain. Designing and implementing such semantic-preserving transformations requires prior knowledge, but this knowledge is defined once for the entire dataset. For visual data, image augmentation methods such as lighting or color distortion, cropping and padding, adding noise and blur, rotation and perspective transform etc. are efficient methods to transform pixels while preserving the semantic meaning of an image content such as

its class label. Distorting low-level visual cues by image augmentation forces the contrastive method to learn a representation invariant to those changes in the inputs. These techniques have been widely used before in supervised learning to capture invariant features and to improve the robustness of the final models. The recent wave of instance discrimination contrastive methods have demonstrated that the same representation can be learned from these augmentation techniques without the need for a class label.

The most used data augmentation operations are:

- Random cropping and then resize back to the original size
- Random color distortions
- Random Gaussian blur
- Random color jittering
- Random horizontal flip
- Random grayscale conversion
- Multi-crop augmentation: proposed by SwAV[4], uses two standard resolution crops and sample a set of additional low resolution crops that cover only small parts of the image. Using low resolution crops reduces the compute cost

2.4.3 Contrastive Loss

A common way of defining a loss function is to measure the difference between a model prediction and a *fixed target*, such as classifying the image into predefined classes (e.g., cat or dog) by cross-entropy loss. Contrastive losses measure the similarities of sample pairs in the embedding space. Gutmann & Hyvarinen[16] proposed the Noise Contrastive Estimation (**NCE**) loss, a method for estimating parameters of a statistical model, but NCE works with only one positive and one noise (or negative) sample. By formulating the NCE loss as a multi-class classification problem, we automatically incorporate multiple negative samples for comparison. SimCLR[5] proposes the **NT-Xent** loss, or *normalised-temperature cross-entropy*, a slightly different version of the NCE loss with a temperature parameter τ to control the sensitivity of the cosine similarity scoring function. The NT-Xent is defined as follows, given

the query (positive sample) q, the positive key, or the augmented view of the positive sample, k^+ and the set K of the negative samples:

$$L_{NT-Xent}(\mathbf{q}, K) = -log\frac{\exp\frac{q^T k^+}{\|q\|\|k^+\|\tau}}{\sum_{k\in K}\exp\frac{q^T k}{\|q\|\|k\|\tau}}$$

The temperature τ has the same effect of controlling the attraction-repulsion radius around the query. How the temperature parameter affects the learned representations is seen in section 2.4.5.

2.4.4 Representations

It is not clear what makes a good representation. Deep learning has developed some core principles for learning good representations:

- **Distributed**: representations that are expressive and can represent an exponential amount of configurations for their fixed size. In contrast with other types of representations as one-hot encoding. Each direction in representation space can correspond to the value of a different configuration variable. For example one direction in representation space corresponds to whether a person is male or female, while another corresponds to whether the person is wearing glasses.;
- **Abstraction and invariant**: good representations can capture abstract concepts that are invariant to small and local changes;
- **Disentangled representations**: each factor of variation should be as disentangled as possible. This can also be beneficial for explainability, since the ideal representation should disentangle the underlying causal factors of variance that generated the data distribution. Supervised learning uses label y that directly represents one of the factors of variation. With unlabeled data we make use of implicit prior beliefs that we impose to guide the learner.

The paper "***What Makes for Good Views for Contrastive Learning?***"[35] explores the importance of view selection, arguing that the goal is to reduce the mutual information between views, according to the **InfoMin** principle. The optimal representation should maintain all the necessary information to predict y (in case of classification) with smallest complexity, which

makes it more generalizable. We need to find the sweet spot of views that does not keep too many irrelevant variables and that maximizes the mutual information between the examples and the true label, while minimizing the mutual information between the views of the same positive example. The InfoMin principle complements the already popular **InfoMax**[27] principle, which states that a goal in representation learning is to capture as much information as possible about the stimulus. But information maximization is only useful as far as that information is relevant to the task, and beyond that point throwing away information about nuisance variables is preferable as it increases generalization.

2.4.5 Temperature parameter

"***Understanding the Behaviour of Contrastive Loss***"[36] studies the effect of the temperature parameter in the NT-Xent loss. The contrastive loss is a hardness-aware loss function which automatically concentrates on optimizing the hard negative samples. giving penalties to them according to their hardness. The temperature plays a role in controlling the strength of penalties on the hard negative samples. Small temperature tends to penalize much more on the hardest negative samples in a way that the local structure of each sample tends to be more separated, with the embedding distribution likely to be more uniform. The uniformity of the embedding distribution in unsupervised contrastive learning is important to learn separable features, but the excessive pursuit of uniformity may, however, break the underlying semantic structure. On the other hand, contrastive loss with high temperature is less sensitive to the hard negative samples, and the hardness-aware property disappears as the temperature approaches $+\infty$.

Pushing the semantically consistent samples away is harmful to generate useful features. If the contrastive loss uses a very small temperature, the loss function will give large penalties to the nearest neighbours which are very likely to share similar semantic features with the anchor point. There is thus a uniformity-tolerance dilemma, where a good contrastive loss should trade-off to obtain the embeddings to be both locally clustered and globally separated while being distributed uniformly enough to be more separable.

The paper also experiments with a simpler contrastive loss which is not hardness-aware, in combination with explicit hard negative mining by sam-

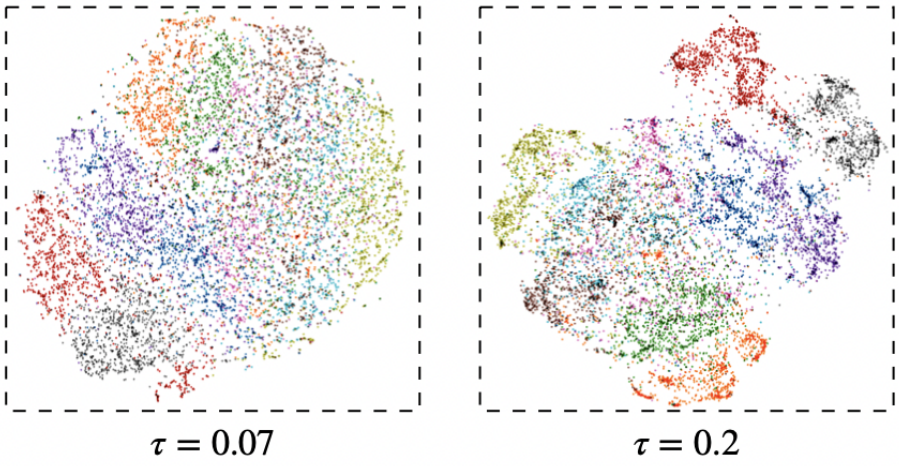

Figure 2.2: T-SNE visualization of the embedding distribution[36]. The two models are trained on CIFAR10. The temperature is set to 0.07 and 0.2 respectively. Small temperature tends to generate more uniform distribution and be less tolerant to similar samples.

pling negatives with nearest neighbor strategy. We will see this strategy later with all the other hard negative mining approaches.

2.5 State-of-the-art contrastive learning methods

2.5.1 SimCLR

A Simple Framework for Contrastive Learning (**SimCLR**)[5] studies the importance of composition of data augmentations and how the quality of the representations is improved thanks to a non-linear projection head between the representations and the contrastive loss. The authors also show that contrastive learning benefits from larger batch sizes and more training steps compared to supervised learning. SimCLR does not require specialized architectures nor a memory bank.

SimCLR learns representations by maximizing agreement between different augmentation views of the same image via a contrastive loss in the embedding space. As illustrated in Figure 2.3, this framework is composed of the following four major elements:

- A stochastic *data augmentation* module, consisting in three simple augmentation applied sequentially: random cropping, random color distortions and random Gaussian blur. In particular, the combination of

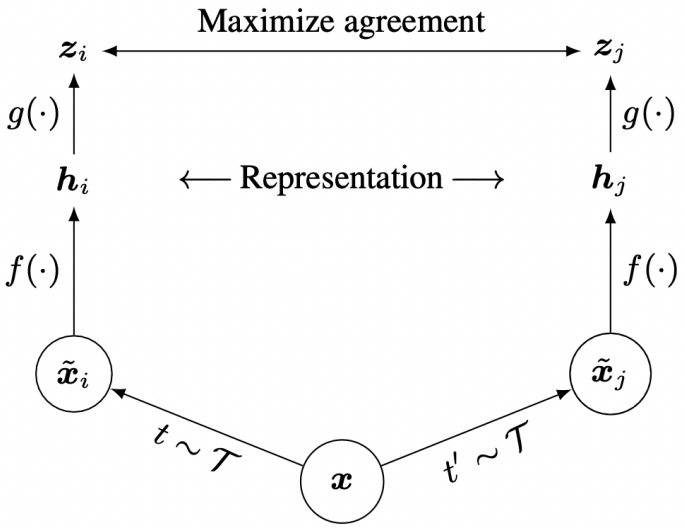

Figure 2.3: A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations (t $\sim$ T and t' $\sim$ T) and applied to each data example to obtain two correlated views. A base encoder network $f(\cdot)$ and a projection head $g(\cdot)$ are trained to maximize agreement using a contrastive loss. After training is completed, we throw away the projection head $g(\cdot)$ and use encoder $f(\cdot)$ and representation h for downstream tasks.

random crop + *random color distortion* works particularly well.

- A neural network *base encoder* $f(\cdot)$ that extracts representation vectors from augmented examples. The commonly used ResNet[18] is adopted here for simplicity.

- A small neural network *projection head* $g(\cdot)$ that maps representations to the space where contrastive loss is applied. A MLP with one hidden layer is used.

- A *contrastive loss function*, **NT-Xent** loss is deployed in this case. Let $sim(u, v) = u^T v/||u||||v||$ denote the dot product between l_2 normalized u and v, then the loss function for a positive pair of examples (i, j) is defined as

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

  where $1_{[k\neq i]}$ is an indicator function evaluating to 1 iff $k \neq$ ß and τ denotes a temperature parameter.

Other interesting studies from the SimCLR paper concern the effect of varying the batch size and the impact of using a nonlinear projection head. By increasing the batch size and the number of training epochs, we also provide the model with more negative examples, which improves the results. Similarly, using a nonlinear projection head is superior to using linear projection heads (+3% in accuracy), and is much better than no projection (>10%).

2.5.2 MoCo

Momentum Contrast (MoCo) [17] tackles contrastive learning as dictionary look-up, building a dynamic dictionary with a queue and a moving averaged encoder. This type of encoder guarantees a large and consistent dictionary on-the-fly, facilitating contrastive unsupervised learning, since the momentum update is more tractable than updating the whole queue with back-propagation.

MoCo closes the gap between unsupervised and supervised representation learning in multiple vision tasks, mainly detection and segmentation task.

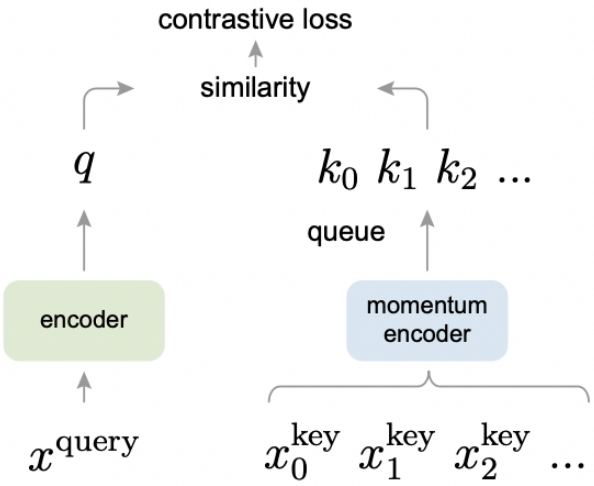

Figure 2.4: Momentum Contrast (MoCo) trains a visual representation encoder by matching an encoded query q to a dictionary of encoded keys using a contrastive loss. The dictionary keys $k0, k1, k2, ...$ are defined on-the-fly by a set of data samples. The dictionary is built as a queue, with the current mini-batch enqueued and the oldest mini-batch dequeued, decoupling it from the mini-batch size. The keys are encoded by a slowly progressing encoder, driven by a momentum update with the query encoder. This method enables a large and consistent dictionary for learning visual representations.

The "keys" of the dictionary are the images and they are represented by

an encoder network. The encoder is trained to perform dictionary look-up: an encoded "query" should be similar to its matching key and dissimilar to others. A contrastive loss has to be minimized during the learning process. The authors of MoCo hypothesize dictionaries should be large and consistent as they change during training. Intuitively, a large dictionary may help to better sample the underlying visual space, while the keys should be represented by the same or similar encoder, so that their comparisons to the query are consistent. Computing the keys for the entire queue each time the encoder is updated may be too slow and too expensive computationally. The dictionary is implemented as a queue of data samples, where the encoded representations of the current mini-batch are enqueued and the oldest are dequeued. A slowly progressing key encoder is used to maintain *consistency*. Formally, denoting the parameters of the key encoder as θ_k and those of the query encoder as θ_q, we update θ_k by:

$$\theta_k \leftarrow m\theta_k + (1-m)\theta_q$$

Consider an encoded query q and a set of encoded samples $\{k_0, k_1, k_2, ...\}$ that are the keys of a dictionary. Assuming there is a single key denoted as k_+ that matches q, with similarity measured by dot product, the InfoNCE, a form of contrastive loss function, is used to :

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

where τ is a temperature hyper-parameter. Ths loss is the log loss of a (K+1)-way softmax-based classifier that tries to classify q as k_+. The value is low when q is similar to its positive key k_+ and dissimilar to all other keys.

MoCo v2[6] just implements two design concepts from SimCLR framework, which are MLP projection head and more data augmentation. These orthogonal integrations makes MoCo v2 even better than SimCLR on some benchmark tasks.

The authors of "*Improving Contrastive Learning by Visualizing Feature Transformation*"[40] experimented in their work with the MoCo framework and they found an interesting insight: varying momentum coefficient m is comparable to varying the hardness of the positive samples. In fact, lower values of m means faster update of the encoder, which leads to representations in the queue to be more similar to the batch, while higher values (0.9 to 0.99)

means slower update of the encoder, leading to higher variance among the representations.

2.5.3 NNCLR

Most of the instance discrimination methods generate positive samples using data augmentations, which however can not provide positive pairs with different viewpoints, deformations of the same object or similar instances from the same semantic class. (**NNCLR**) [12] samples the nearest neighbors from the dataset in the latent space and treats them as positives. This method since it seeks semantic variation by sampling positives in the dataset it is less dependent on data augmentations.

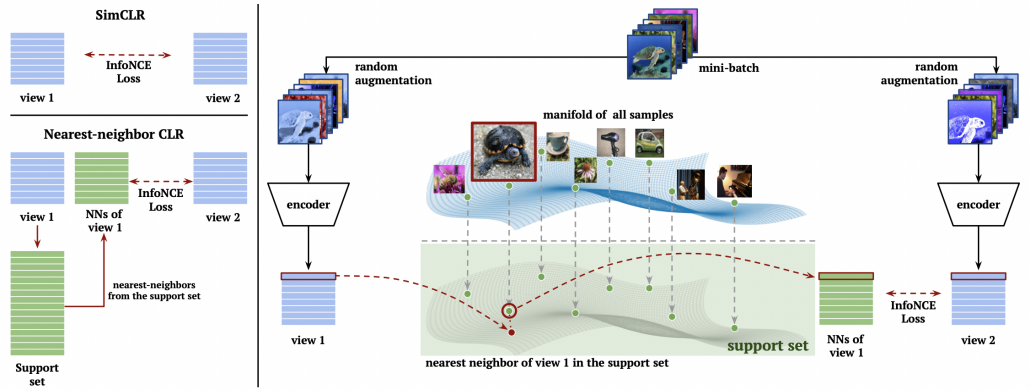

Figure 2.5: Overview of NNCLR Training.

NNCLR is very similar to SimCLR, except for the additional positive mining module. The nearest neighbor is matched in a *support set* implemented as a queue, keeping its size large enough to approximate the full dataset. For each element $x_i$in the batch we compute the similarity scores against each element in the memory bank or queue and we take the most similar example as the augmented view x_i' of x_i. At the end of a training step the queue is updated by appending the embeddings from the current batch and by discarding the oldest elements.

Given the batch $\{z_i\}_{i=1}^{N}$ of size N, the support set Q and the temperature parameter τ, the loss for a positive pair is a variation of the InfoNCE loss:

$$\mathcal{L}_i^{\text{NNCLR}} = -log\frac{exp(\text{NN}(z_i, Q) \cdot z_i^+/\tau)}{\sum_{i=1}^{N} exp(\text{NN}(z_i, Q) \cdot z_k^+/\tau)}$$

where $\text{NN}(z_i, Q)$ is the nearest neighbor operator defined as:

$$\text{NN}(z_i, Q) = \arg\min_{q \in Q} \|z - q\|_2$$

We minimize the average loss over all the elements in the mini-batch to obtain the final loss:

$$\mathcal{L}^{\text{NNCLR}} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_i^{\text{NNCLR}}$$

With linear evaluation and finetuning NNCLR performs better than all the other methods on ImageNet dataset classification.

From the ablation study, NNCLR is less reliant on data augmentations, suffering smaller drops when only cropping is applied. This means that NNCLR could be applied to domains where data augmentations designed for ImageNet are not suitable (like medical images).

The authors of NNCLR also investigate on how often does the NN sampling strategy pick a positive neighbor with the same label as the query. Towards the end of the training, the accuracy of picking the right neighbor (i.e. from the same class) is about 57%. An hypothesized reason for not being higher is due to the random crop augmentation resulting in background windows, which do not contain any object. Thus, there is still possibility of improvement by designing a better NN picking method. In figure 2.6 we report some visual examples of the mined positives with the NN strategy at the end of the training, while figure 2.7 shows how the picking strategy improves over the training epochs.

2.6 Mining strategies in Contrastive Learning

2.6.1 Positive mining in Contrastive Learning

SimCLR[5] already shows that self-supervised contrastive learning benefits from strong data augmentations with respect to supervised learning. More in particular, color transformations are helpful in breaking the similar color distribution in the image patches when random cropping, but a composition of multiple data augmentation operations is crucial for learning good representations. These operations, as seen in the ablation study, when used individually are not sufficient to learn good representations, even if the model can al-

Figure 2.6: Nearest neighbors from support set show the increased diversity of positives in **NNCLR**.

most perfectly identify the positive pairs in the instance discrimination task. When composing augmentations instead, the contrastive prediction task becomes harder, because we introduce more variance in color space and in the spatial structure of the class, so the quality of the representation improves significantly.

Data augmentations should mainly affect the hardness of identifying the positive pair among the negatives, while not impacting as much the quality of the negative samples, because these should already come from different semantic classes, thus carrying enough inter-class variance. Using stronger augmentations however is not sufficient for generating positive pairs which can cover all the variances in a given class. For the aforementioned reasons we present some methods proposed in literature that try to overcome the barriers imposed by using data augmentations only.

"***Improving Contrastive Learning by Visualizing Feature Transformation***"[40] introduces the need of using "hard positives", positive pairs conveying large view variance, by studying how the score similarities between

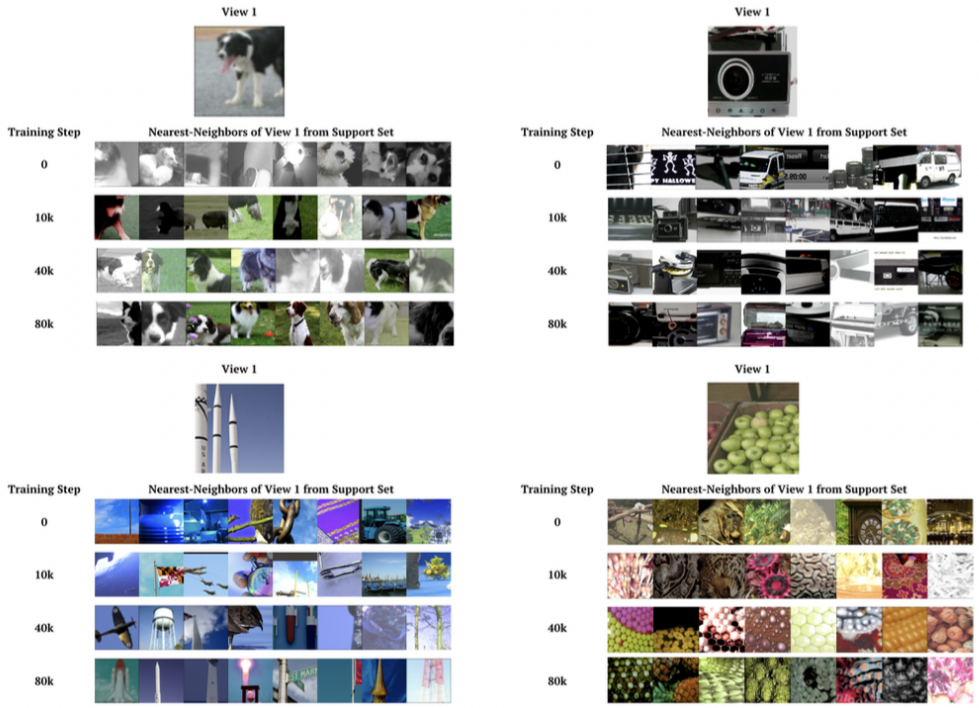

Figure 2.7: Evolution of Nearest-neighbors as the **NNCLR** training proceeds.

positive pairs vary in MoCo under different values of the momentum coefficient m. The case of $m = 0.99$ leads to smaller positive scores than $m = 0.9$, while achieving better performance, as shown in figure 2.8.

A small positive score means less similarity between the pair, which means this positive pair is actually carrying larger view variance. As we have seen in section 2.5.2 larger values for m coefficient means slower update of the encoder, leading to higher variance among the representations.

Starting from these observation, the authors propose a feature transformation method in order to decrease the positive scores. The operation is an extrapolation to transform the original positive pair (z_q, z_k^+) obtained from data augmentation of the same image. Simple weighted addition is used to generate new feature vector for the two positives:

$$\hat{z_q} = \lambda z_q + (1 - \lambda) z_k^+$$

$$\hat{z_k^+} = \lambda z_k^+ + (1 - \lambda) z_q$$

where $\hat{z_q}$ and $\hat{z_k^+}$ are the transformed new features. After extrapolation, the distance between the extrapolated feature vectors is larger. Interpolation

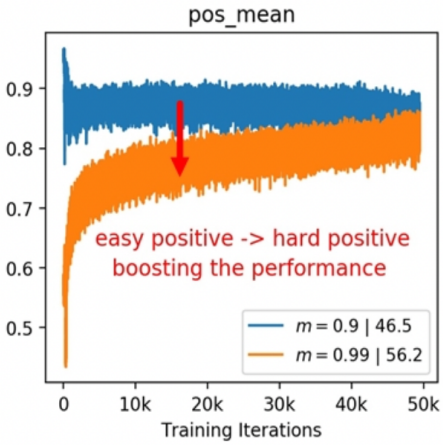

Figure 2.8: The score distribution of positive pairs for m (the momentum in **MoCo**) being 0.99 and 0.9, showing that smaller positive scores generally need longer time to converge and obtain better accuracy.

instead reduces the distance in the embedding space, thus leading to higher scores.

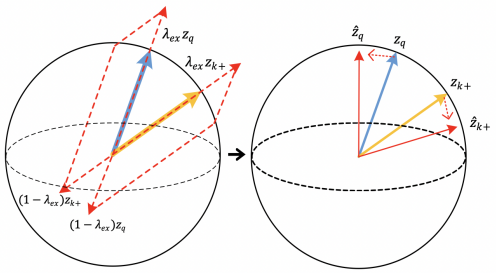

Figure 2.9: **Positive extrapolation**[40]: the two positive features are pushed away from each other using extrapolation, changing easy positives to hard positives, which is better for contrastive learning.

Another method that proposes a strategy for mining more meaningful positive pairs is **NNCLR**, as we have already seen in section 2.5.3. In this case we do not perform any transformation in the feature space, but the directly search for positive candidates in the same semantic class by sampling with nearest neighbor selection. The precision of picking a correct positive with the same semantic class as the anchor has still room for improvement, suggesting that it may benefit from a consistency check in order to make sure we are picking a true positive instead of a false positive.

2.6.2 Negative mining in Contrastive Learning

What makes a good negative? In metric learning it is useful to select negatives that are hard to discriminate, because they give a larger contribution in the loss. We can identify them as the **hard negatives**, with these desired properties:

- Negative samples have different semantic labels from the positive anchor
- Hard negative samples are embedded nearby to the positive anchor in the representation space

Hard negatives are difficult to discriminate because their representation is very similar to the representation of the positive sample. This similarity is reflected in the embedding space as well, where representations that have small distance between each other should share some common or similar features in the pixel representation space.

As we have mentioned in section 2.4.5, the "***Understanding the Behaviour of Contrastive Loss***"[36] paper proposes a method to perform explicit hard negative mining, by slightly modifying the contrastive loss. The loss discriminates between informative and uninformative negatives by thresholding the similarity scores between the positive anchor and the negatives. More in particular, given similarity $s_{i,j}$ between the anchor sample i and the negative sample j, sample j is considered to be *informative* if $s_{i,j} \in [s_\alpha^{(i)}, 1.0]$ given an upper α quantile $s_\alpha^{(i)}$ for the anchor x_i, while the *uninformative* interval is $[-1.0, s_\alpha^{(i)}]$. A negative sample j is then considered "hard negative" if it is similar to the anchor i. Explicit hard negative mining is done by setting the gradients of the similarity scores in the uninformative interval to 0, such that these gradients are not counted in the contrastive loss:

$$\mathcal{L}_{hard}(x_i) = -log \frac{\exp(s_{i,i}/\tau)}{\sum_{s_{i,k} \geq s_\alpha^{(i)}} \exp(s_{i,j}/\tau) + \exp(s_{i,i}/\tau)}$$

The $\mathcal{L}_{hard}$ only penalizes the informative hard negative samples. The results shown are competitive.

"***Improving Contrastive Learning by Visualizing Feature Transformation***"[40] in addition to sampling "hard positive", also devises a technique to generate new negatives, which are diversified negatives but not hard-negatives. New negatives are created with interpolation among negatives in-

stead of extrapolation. Given the negative memory queue $Z_{\text{neg}} = \{z_1, ..., z_K\}$, where K is the size of the memory queue, and Z_{perm}, the random permutation of Z_{neg}, the interpolation between two memory queue to create a new queue $\hat{Z}_{\text{neg}}$ is:

$$\hat{Z}_{\text{neg}} = \lambda Z_{\text{neg}} + (1 - \lambda) Z_{\text{perm}}$$

Interpolation in this case is employed to fully utilize negative samples in each training step, creating diversity in the memory queue. The generated negatives are not necessarily considered harder. The negative interpolation leads to around 2% improvement over MoCo and SimCLR baselines on ImageNet-100 linear classification.

Divide and Contrast (**DnC**)[34] explores the effects of contrastive learning from larger, less-curated image datasets. When training on this type of datasets, the density of informative negatives will be sparse if we sample randomly from the whole dataset. Instead, if we contrast locally between semantically-similar classes, the sampled negatives will be more informative.

DnC performs hard negative mining by alternating contrastive learning and clustering. It can be used with any self-supervised method. The pipeline of DnC consists of three stages:

1. We first train a MoCo model (or any other model) on the given dataset for N_1 epochs. We call this the **base** model. We use this base model to extract representations for a set of samples in the training set and cluster them into K partitions with K-means.

2. For each partition we train a separate MoCo model for N_2 epochs from scratch, called **expert** model.

3. Finally, given a base model that captures the general knowledge of the dataset and expert models focusing on locally similar categories, we distill knowledge from these models into a distillation model. In this stage, we train for N_3 epochs.

The authors want to test the hypothesis that training self-supervised contrastive methods on images from similar classes should let the model learn more fine-grained features, thus improving accuracy. To test this, they train various methods on the canine subset (130 classes) of ImageNet and compare them with models trained on the full dataset by linearly evaluating on

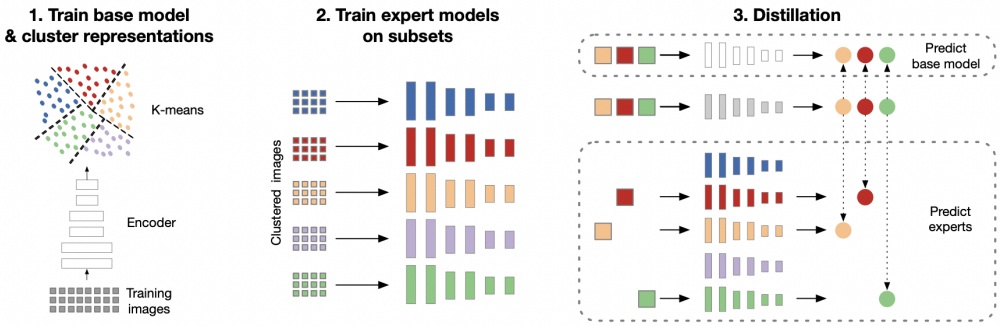

Figure 2.10: **Overview of Divide and Contrast (DnC)**. DnC can be used in conjunction with any self-supervised learning method. In the first step a self-supervised learning method is trained on the whole dataset, which we call the base model. The image representations of the base model are then clustered with k-means into 5, 10 or more groups. In the second step, the clustered dataset is then used to train an expert model on each of the image clusters. In the third step the experts and base model are distilled into a single model by predicting their representations. By splitting the dataset into semantically-similar subsets, contrastive methods need to pay more attention to the differences between the images in those clusters and learn more specific representations.

the canine subset. The models pre-trained on the canine subset significantly perform better.

From the ablation studies, if instead of clustering the subsets for the expert models are created randomly, performance is negatively affected.

The authors of MoCo showed that increasing the memory size, is crucial to getting better and harder negatives. As we can see in figure 2.11, by plotting the highest 1024 matching probabilities p_i for ImageNet-100, we see that in the beginning of the training the logits are flat. As the training progresses, fewer and fewer negatives offer significant contributions to the loss, showing that most of the memory negatives are practically not helping a lot.

"Hard Negative Mixing for Contrastive Learning"[20] proposes to synthesize hard negative features by mixing some of the hardest negative features by creating convex linear combination of pairs. The hardest negatives used for mixing are chosen by sorting the similarity scores between the positive query and the negative keys in decreasing order and by only keeping the first N samples.

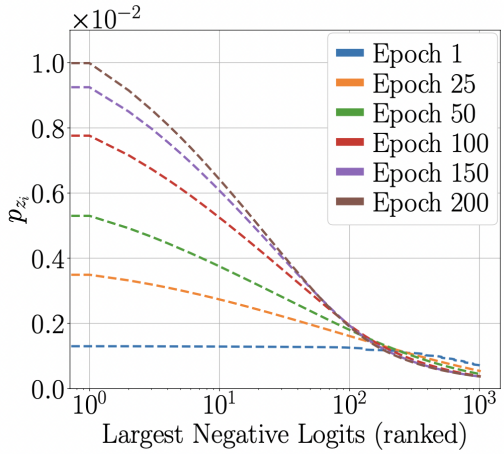

Figure 2.11: [20]A histogram of the 1024 highest matching probabilities p_{z_i} , $z_i \in Q$ for MoCo-v2[6], across epochs; logits are ranked by decreasing order and each line shows the average value of the matching probability over all queries.

2.6.3 Dealing with false negatives

Besides mining hard negative samples for more informative contributions in the contrastive loss, it would be desirable to prevent the contrasting of *false negatives*, as it would induce two issues in representation learning: discarding semantic information and slow convergence. By contrasting these undesirable negative pairs, the network is encouraged to discard their common features in the learned representation, which carry indeed the same semantic content. False negatives, moreover, hinder the convergence of contrastive learning-based representation learning due to the appearance of contradicting objectives. The authors of "*Boosting Contrastive Self-Supervised Learning with False Negative Cancellation*"[19] propose the False Negative Cancellation (**FCN**) method to mitigate the effects of false negatives. Considered that the process of identifying false negatives is fundamentally difficult without the help of class labels, FCN proposes an approach to find false negatives based on the following observations:

- False negatives are samples from different images with the same semantic content, therefore they should hold certain similarity (e.g., dog features).
- A false negative may not be as similar to the anchor as it is to other augmentations of the same image, as each augmentation only holds a specific view of the object.

From the above observations comes the idea of approximating a false positive with more augmented views of the positive anchor.

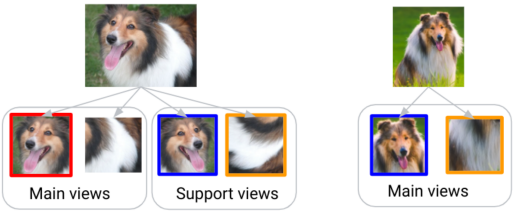

Figure 2.12: **False Negative Cancellation (FNC)**. Finding false negatives with the support views. Negative samples (main views, right) may not have as reliable similarity with the anchor itself (red) as they do with other augmented views of the same image (support views). For instance, the dog's face in the support view (left, blue) is closer to the negative sample (right, blue) in terms of the facial orientation than the anchor (red).

The support set of augmented views is used to match a negative against different orientations and alterations of the positive anchor. By looking at figure 2.12, for example, we see that the negative (right) is more similar to an augmented view of the anchor (left) than it is to the main view, because they have the same face orientation. The overall strategy of FNC of identifying false negatives is structured in the following steps:

1. For each anchor i, generate a support set $\mathcal{S}_i = \{z_i^s\}$ that contains other support views from the same image besides the two main views.

2. Compute similarity scores, $\text{score}_{m,i}^s = \text{sim}(z_m, z_i^s)$, between a negative sample z_m and each sample z_i^s in the support set.

3. Aggregate the computed scores for each negative sample, $\text{score}_{m,i} = \text{aggregate}_{s \in \mathcal{S}}(\text{score}_{m,i}^s)$.

4. Define a set of potential false negatives F_i as the negative samples that are most similar to the support set based on the aggregated scores, $F_i = \text{best}(\text{score}_i)$, where $\text{score}_i = \{\text{score}_{m,i} | m\}$ is the set of scores for each negative sample with respect to anchor i.

It is noteworthy to consider that there are more ways to define the similarity function, the strategy for aggregating scores, and the manner in which the most similar samples are defined. The authors used the *cosine similarity* function with a *max aggregation* function, while for the screening strategy a

combination of top-k and thresholding leads to the most consistent results when varying the value of k. From the ablation studies the false negative detection accuracy steadily increases, reaching approximately 40% accuracy by 100 training epochs, and from the results on ImageNet linear evaluation FNC achieves a competitive accuracy, as shown in table 2.1.

2.7 Clustering-based methods

The contrastive loss explicitly compares pairs of image representations to push away representations from different images while pulling together those from transformations of the same image. Since computing all the pairwise comparisons on a large dataset is not practical, most implementations approximate the loss by reducing the number of comparisons to random subsets of images during training (mini-batches). An alternative to approximate the loss is to approximate the task, or to relax the instance discrimination problem. For example, *clustering-based* methods discriminate between groups of images with similar features instead of discriminating between individual images. One problem of clustering-based methods is that the objective is tractable, but it does not scale well with the dataset as it requires a pass over the entire dataset to form image "codes) (i.e. cluster assignments), which are used during the training phase as targets. We will thus see **SwAV**, a method that proposes an online computation of the codes, which is more efficient and has shown promising results compared to the contrastive methods.

2.7.1 SwAV

SWapped **A**ssignments between multiple **V**iews (**SwAV**)[4] takes advantage of contrastive methods without requiring to compute pairwise comparisons. This method simultaneously clusters the data while enforcing consistency between cluster assignments produced for different augmentations (views) of the same image, instead of comparing the features directly as in contrastive learning. It learns the cluster codes online and it does not need memory banks. The authors of SwAV also propose the usage of multiple views with multi-crop without increasing memory or computation. This augmentation strategy improves performance of other frameworks too.

The core of the method is the "swapped" prediction mechanism, where we compute the cluster assignment from one view and predict it from the other

augmented view. It can be considered fully contrastive because it enforces the cluster codes between different views of the same image to be equal, but it does not use negatives examples explicitly. What it does is to contrast the feature vector against all the prototypes of the clusters, contrasting thus against the prototypes of the negative clusters. Given two image features z_t and z_s from two augmented views of the same image, we compute their codes q_t and q_s by matching these features to a set of K prototypes $\{c_1, ..., c_K\}$. The "swapped" prediction problem is set as:

$$\mathcal{L}(z_t, z_s) = l(z_t, q_s) + l(z_s, q_t)$$

where the function $l(z, q)$ measures the fit between features z and a code q, computed as the dot cosine similarity:

$$l(z_t, q_s) = -\sum_{k \in K} q_s^{(k)} \log p_t^{(k)}$$

where

$$p_t^{(k)} = \frac{\exp \frac{1}{\tau} z_t^T c_k}{\sum_{k' \in K} \exp \frac{1}{\tau} z_t^T c_{k'}}$$

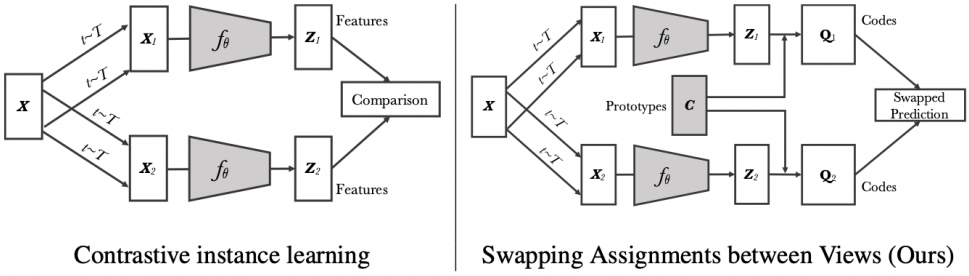

Figure 2.13: **Contrastive instance learning (left) vs. SwAV (right).** In contrastive learning methods applied to instance classification, the features from different transformations of the same images are compared directly to each other. In SwAV, we first obtain "codes" by assigning features to prototype vectors. We then solve a "swapped" prediction problem wherein the codes obtained from one data augmented view are predicted using the other view. Thus, SwAV does not directly compare image features. Prototype vectors are learned along with the ConvNet parameters by back-propagation.

In practice given two augmentations of the same image and their extracted features, we compute their codes by matching against a set of prototypes

Linear evaluation on ImageNet		
Method	**Epochs**	**Top1**
SimCLR[5]	800	70.0
MoCo[17]	800	60.6
MoCov2[6]	800	71.1
DnC[34]	1000	74.5
FNC[19]	800	74.4
NNCLR[12]	800	75.4
SwAV[4]	800	75.3

Table 2.1: **Linear classification results on ImageNet**. Top-1 accuracy for linear models trained on frozen features from different self-supervised methods.

(similar to clustering centroids). The loss essentially compares the features of the two views using the intermediary cluster codes. We perform the swapped prediction where we try to predict the cluster assignment of the other view.

From the ablation studies, the authors show that learning the prototypes is better than using fixed prototypes throughout the training phase.

The SwAV framework is also tested on Instagram 1B dataset, which is uncurated. It's 6% better than SimCLR when evaluating with linear classification on frozen features.

2.8 Results on Imagenet

The results on ImageNet linear evaluation on frozen features are reported here in table 2.1, where all the cited methods use the standard ResNet-50 as backbone. Almost all the methods are pre-trained for 800 epochs, except for DnC which was pre-trained by its authors for 1000 epochs. Only the Top-1 accuracies are reported.

The best frameworks on ImageNet linear evaluation are NNCLR and SwAV, among the selected methods seen in this work. There are also non-contrastive methods present in the research literature like *BYOL*[15] and *SimSiam*[7], but they are not discussed here. These methods are different from the contrastive ones because they do not rely on negative samples for learning representations.

2.9 The impact of the training data

"When Does Contrastive Visual Representation Learning Work?"[9] poses an important problem about the type of data used for the pre-training. Besides ImageNet, we do not know how self-supervised learning methods work on other types of datasets. ImageNet is considered to be "*curated*", since the images underwent a careful selection process by human annotators, who had to pay attention at centering the object of the class in order to facilitate the training of a supervised classifier. In the wild, images are often posted on the web without any goal related to training neural networks on a particular task, and it would be wasteful to discard this huge amount of data that comes for almost free. So the authors wanted to investigate under which conditions self-supervised learning, or self-supervised contrastive learning more specifically, learns good representations.

For the experiments they used datasets that span different visual properties: *curated* vs "*in-the-wild*", *fine* vs *coarse* grained, object centric vs scenes. SimCLR framework is used in all the settings, though it is important to validate also with other methods, for example non-contrastive ones.

More in particular, the impact of the following conditions are being studied with the related results:

- *Data quantity*: decreasing the amount of data used in the training phase by half (1M vs 500k) for ImageNet decreases performance just by 1-2%. Therefore, increasing pre-training size leads to rapidly diminishing returns.

- *Data quality*: artificial image corruption is added to images to study the impact. Down-sampling is the operation that degrades performance the most.

- *Pre-training domain*: same domain in train and test set still proved to more effective when evaluating on the test set. Cross training with ImageNet as pre-training dataset leads to best cross training performances when transferring to different domains for the test set. Mixing different datasets to create a pooled mixed dataset is also worse than using in-domain datasets. It appears that adding pre-training data from different domains degrades performance, the more we add the worse it gets. This

may be due to the pretext task becoming easier, since it is easier to discriminate between different domains.

- *Task granularity*: contrastive loss may not be sufficient to learn fine-grained features. The gap between self-supervised and supervised grows as task granularity increases. The contrastive loss may tend to encourage the learning of coarse-grained features which are insufficient for fine-grained tasks.

3 Experiments

From the inspiration of the NNCLR method, where we sample the nearest neighbors of the positive anchors from a memory bank, we want to generate positive pairs that can carry variation in viewpoint and in deformation too. As a matter of fact, strong data augmentations are not enough to represent different instances from the same semantic class. By sampling positives in the dataset we are also less dependent on data augmentations, which can not be applied in certain domains (e.g. medical images). A problem arises if we sample the nearest neighbor of the anchor coming from a different semantic class distribution. These are in fact considered *false positives*. Since we do not have labels to check for class consistency, we propose to use another unsupervised technique for the validation. We found a good candidate in SwAV, which is able to perform online-clustering in order to assign a pseudo-class to each sample, in a compute-efficient way. We compute a pseudo-class for each image using the prototypes from the clustering module in SwAV, where the prototypes can be compared to centroids of the clusters, but without the need of performing clustering on the whole dataset to compute them.

With the help of the SwAV prototypes we can also sample hard negatives, i.e. negatives belonging to different clusters than the positive anchor considered, but with the centroid close to the centroid of the positive sample. In other words, we do not want easy negatives sampled from clusters located far away in the embedding space, because these are easy to discriminate against and should not contribute significantly to the contrastive loss. This idea of contrasting negative samples with high similarity score with respect to the positive anchor is validated by the authors of "***Understanding the Behaviour of Contrastive Loss***"[36], where the low similarity scores of negative samples are not considered in the loss in order to only use the most informative negatives, or the negatives of high similarity score.

3.1 Proposed approach

The proposed method has two modules, one performing positive mining and one performing negative mining. Since the effects of mining hard negatives has been widely explored in research, we wanted to focus on the less studied positive mining literature in contrastive learning. For this reason, in the experiments we study the eventual benefits of the positive mining module only.

3.1.1 Hard Positive Mining

For "*hard positive*" sample we mean an element from the dataset that has the same semantic label as the positive anchor, but is not the result of applying a data augmentation module to the anchor itself. Including true positives drawn from different images would increase the diversity of the training data and, in turn, has the potential to improve the quality of the learned embeddings. Indeed, Khosla et al.[22] show that supervised contrastive learning (i.e., where an anchor is attracted to samples having the same semantic label) can be more effective than the traditional supervised cross-entropy loss. Following the InfoMin principle cited in section 2.4.4, if the shared information is small, then the learned representation can discard more information about the input and achieve a greater degree of invariance against nuisance variables. By sampling samples from the dataset we can obtain positive pairs that share less information than the augmented views generated from data augmentation.

Similarly to NNCLR, for a given positive anchor we use search for the nearest neighbor in the memory bank but before using it as the second view for the positive pair, we check that it belongs to the same semantic class as the anchor. In our case we consider the clusters created by the SwAV prototypes, so we compute the cluster assignment for the anchor and the cluster assignment for all the samples in the memory bank. We mask out all the samples that have been assigned to a different prototype than the prototype of the positive anchor and we search for the nearest neighbor. By excluding the samples with different cluster assignment we want to make sure that, when performing the nearest neighbor computation, we are selecting a true positive, which should carry the same semantic class as the anchor.

From a practical point of view, in order to find the nearest neighbor for each positive x_i in the batch of size N, among the samples q_j in the memory

bank of size K, we perform the following operations:

1. We first compute the matrix $S_{i,j}$ of the *cosine similarity* scores between the $l2$-normalized embeddings $\{X_i\}_{i=1}^{N}$ of the current batch and the $l2$-normalized embeddings $\{M_j\}_{j=1}^{K}$ of the memory bank:

$$s_{i,j} = \|x_i\|_2 \cdot \|m_j\|_2$$

   where the dot $\cdot$ operator is the dot product.

2. We compute the cluster assignment of the elements of the batch and of the elements in the memory bank, by passing the embeddings of the elements as input to the prototypes layer. The outputs of the prototypes layer are the similarity scores $q_{i,t}$ between the input embedding x_i and each prototype c_t:

$$q_{i,t} = \|x_i\|_2 \cdot \|c_t\|_2$$

3. We add $+10$ to the elements $s_{i,j}$ of the row i in the matrix S, where:

$$s_{i,j} = s_{i,j} + 10 \quad \text{if} \quad q_i == q_j$$

   where q_i is the cluster (prototype) assignment of the element x_i of the batch and q_j is cluster (prototype) assignment of the element m_j of the memory bank. By adding $+10$, we are sure that when computing the nearest neighbor we will select one of the elements in memory bank that have the same cluster assignment of the element x_i from the batch. This is because the cosine similarity scores are in the range $[-1, 1]$.

4. Finally, we compute the nearest neighbor of element x_i by taking the element j from the memory bank M with the highest similarity score in row S_i:

$$\text{NN}(x_i, M) = \arg\max_{m_j \in M} S_i$$

3.1.2 Hard Negative Mining

As we have seen in section 2.6.2, we would define as *hard negatives* the samples that have the following properties:

- Negative samples have different semantic labels from the positive anchor
- Hard negative samples are embedded nearby to the positive anchor in the representation space

Again, we can reuse the same prototypes from the SwAV module to perform online clustering on the samples in the memory bank, obtaining pseudo-labels we can exploit for the class-consistency check. Differently from the positive mining pipeline, we only consider the samples with a different cluster assignment than the positive anchor, and we take the top k samples from the bank with the highest similarity scores with respect to the positive anchor.

Similarly to the positive mining pipeline, in order to find the nearest neighbor for each positive x_i in the batch of size N, among the samples q_j in the memory bank of size K, we perform the following operations:

1. We compute the matrix $S_{i,j}$ of the *cosine similarity* scores between the $l2$-normalized embeddings $\{X_i\}_{i=1}^{N}$ of the current batch and the $l2$-normalized embeddings $\{M_j\}_{j=1}^{K}$ of the memory bank:

$$s_{i,j} = \|x_i\|_2 \cdot \|m_j\|_2$$

   where the dot $\cdot$ operator is the dot product.

2. We compute the cluster assignment of the elements of the batch and of the elements in the memory bank, by passing the embeddings of the elements as input to the prototypes layer. The outputs of the prototypes layer are the similarity scores $q_{i,t}$ between the input embedding x_i and each prototype c_t:

$$q_{i,t} = \|x_i\|_2 \cdot \|c_t\|_2$$

3. We subtract -10 to the elements $s_{i,j}$ of the row i in the matrix S, where:

$$s_{i,j} = s_{i,j} - 10 \quad \text{if} \quad q_i == q_j$$

   where q_i is the cluster (prototype) assignment of the element x_i of the batch and q_j is cluster (prototype) assignment of the element m_j of the memory bank. By subtracting -10, we are sure that when computing the nearest neighbor we will select one of the elements in memory bank

that does not have the same cluster assignment of the element x_i from the batch.

4. Finally, we compute the hardest *num_ negatives* hard negative w.r.t. x_i by taking the *num_ negatives* elements j from the memory bank M with the highest similarity scores in row S_i:

$$\text{Hard-Neg}(x_i, M) = \text{top\_k}(S_i)$$

where *top_ k* takes the first k elements with the highest similitarity scores.

3.1.3 Network architecture

As common practice in computer vision, we use the ResNet[18] architecture for the backbone, but instead of using a ResNet-50 here we deploy a ResNet-18 for simplicity and for shorter training times. The output of the backbone is our embedding of size 512, which is fed to the projection head and to the prototypes layer. Moreover, the output from the projection head is given as input to the prediction head, a simple one-layer MLP which gives a small boost to the performance of the instance discrimination task, as shown by the authors of NNCLR[12]. Both positive and negative mining are possible thanks to the prototypes used in SwAV. We took the prototypes layer from the SwAV framework and added it to the NNCLR framework.

Overall, the components of our framework are:

- *Base encoder* neural network: we use ResNet-18 architecture.
- *Projection head*: a MLP that takes as input the embedding from the base encoder and projects it to the contrastive loss space.
- prediction head: a MLP that takes as input the output from the projection head. explain how it improved results in NNCLR
- *Prototypes layer* from SwAV: a single layer MLP that maps the embedding from the base encoder to the prototypes. The number of output units is equal to the number of the clusters.
- *Memory bank*: shared by the positive and negative mining pipelines to sample both positive and negative samples. The memory bank is

Architecture	
Layer	**Specs**
Input	size=(128,128,3)
ResNet-18	backbone
Embedding	size=512
Prototypes	size=*num_ clusters*, 1 linear layer without biases
Projection	size=256, 2-layer MLP with BatchNorm and ReLU activation
Prediction	size=256, 2-layer MLP with BatchNorm and ReLU for first layer

Table 3.1: The network architecture of our approach with all the layers after the embedding, which is the output from the backbone ResNet-18.

implemented as a queue, where the embeddings of the latest batch are enqueued and the embeddings of the oldest batch are dequeued.

- *Loss*: the loss is given by the sum of the contrastive loss (seen section 2.4.3) and the SwAV loss (seen section 2.7.1).

A summary of the network architecture can be seen in table 3.1.

3.1.4 Loss

The loss is given by a combination of the NT-Xent contrastive loss proposed by SimCLR and of the swapped prediction loss in SwAV. Here we only see the simplest case of summing the two terms, but there is the possibility of using coefficients to weight the individual contributions in the overall loss.

Specify that the temperature parameters could be different for contrastive and swav.

Let z_s and z_t be the normalized representations of the positive pair (s, t) and let $sim(u, v) = u^T v/||u||||v||$ denote the dot product between l_2 normalized u and v. The **contrastive loss** for the positive pair (s, t) is defined as:

$$\mathcal{L}_{s,t}^{\text{contr}} = -log\frac{exp(sim(z_s, z_t)/\tau_{\text{contr}})}{\sum_{k=1}^{2N} 1_{[k\neq s]} exp(sim(z_s, z_k)/\tau_{\text{contr}})}$$

where $1_{[k\neq s]}$ is an indicator function evaluating to 1 iff $k \neq s$ and τ_{contr} denotes a temperature parameter for the contrastive loss term.

Given the codes q_s and q_t computed by matching the features of representation vectors z_s and z_t to a set of K prototypes $\{c_1, ..., c_K\}$, the **"swapped" prediction loss of SwAV** is defined as:

$$\mathcal{L}_{s,t}^{\text{SwAV}} = l(z_s, q_t) + l(z_t, q_s)$$

where $l(z_s, q_t)$ (and respectively $l(z_t, q_s)$) is given by

$$l(z_s, q_t) = -\sum_{k \in K} q_t^{(k)} \log p_s^{(k)} \qquad p_s^{(k)} = \frac{\exp \frac{1}{\tau_{\text{SwAV}}} z_s^T c_k}{\sum_{k' \in K} \exp \frac{1}{\tau_{\text{SwAV}}} z_s^T c_{k'}}$$

The final loss for a positive pair s, t in our approach is simply given by the sum of the above terms:

$$\mathcal{L}_{s,t} = \mathcal{L}_{s,t}^{\text{contr}} + \mathcal{L}_{s,t}^{\text{SwAV}}$$

3.2 Datasets

For our experiments we used smaller datasets than the full ImageNet. The reason of using smaller dataset resides in the limitation of both computational and time resources. Moreover, experiments seen in section 2.9 proved that decreasing the amount of unlabeled training data by half only degrades downstream classification performance by 1-2%. This means that current self-supervised methods may be unable to take advantage of very large pre-training sets. However, we do not deny the advantages of pre-training on the complete ImageNet dataset.

We found good candidates in *Stanford Dogs dataset* and *STL-10*. These are considerably smaller in size and they present different number of classes and characteristics.

3.2.1 Stanford Dogs Dataset

The Stanford Dogs dataset[21] contains images of 120 breeds of dogs. This dataset has been built using images and annotation from ImageNet for the task of fine-grained image categorization. The classes from the training set and from the test set are the same. For our purposes, we split the whole dataset in training set and in validation set. We discard the labels from the

training set in order to perform self-supervised learning, while we keep those from the validation set.

Key features:

- 120 classes of breeds of dogs.
- Images vary in size, color.
- 20.58k total images, split as 16.46k images (80%) for the training set and as 4.1k for the validation set (20%).
- About 150 images per class
- Images were acquired from labeled examples on ImageNet.

Despite representing a very specific dog breed, each class can offer large intra-class variation in the appearance of the dogs, caused by different fur colors of different ages. We can see some examples from 12 classes in figure 3.1.

3.2.2 STL-10

STL-10[1] dataset is an image recognition dataset for developing unsupervised feature learning algorithms. It is inspired by the CIFAR-10 dataset but with some modifications. In particular, each class has fewer labeled training examples than in CIFAR-10, but a very large set of unlabeled examples is provided to learn image models prior to supervised training.

Key features:

- 10 classes: airplane, bird, car, cat, deer, dog, horse, monkey, ship, truck.
- Images are 96x96 pixels, color.
- 800 test images per class.
- 100k unlabeled images for unsupervised learning. These examples are extracted from a similar but broader distribution of images. For instance, it contains other types of animals (bears, rabbits, etc.) and vehicles (trains, buses, etc.) in addition to the ones in the labeled set.
- Images were acquired from labeled examples on ImageNet.

3.2.3 Comparison

Differences betweeen Stanford Dogs Dataset and STL-10:

- Stanford Dogs Dataset has 120 classes while STL-10 has only 10 classes;
- There is a slight distribution shift in STL-10, i.e. the examples seen during the training phase come from different but similar classes than those in the validation set. In Stanford Dogs Dataset images from both training and validation sets come from the same distribution;
- STL-10 has coarse-grained semantic classes, quite different from each other, while Stanford Dogs Dataset has fine-grained semantic classes, with similar features and related meanings;
- STL-10 has roughly 10k images for each class, while Stanford Dogs Dataset has only about 150 images for each class. Despite having more than 10 times the number of classes than STL-10, Stanford Dogs Dataset has only a fifth of total images compared to STL-10.

These two datasets represent diverse experimental scenarios, offering interesting insights from the contrastive methods tested, without the need of eccessive computational resources thanks to the relatively limited amount of data.

3.3 Evaluation metrics

For our experiments we used K-nearest neighbor (k-NN) evaluation after each training epoch. We used k-NN throughout the learning phase because it is faster to compute and provides a consistent metric with respect to the linear evaluation accuracy.

3.3.1 K-nearest neighbor evaluation

At every validation step we predict features on the validation data. After all predictions on validation data we evaluate the predictions on a k-NN classifier on the validation data using the features from the train data. With set the number of neighbors to 200 and the temperature parameter to 0.1.

3.3.2 Linear evaluation

At the end of the self-supervised pre-training period, the weights of the network are frozen up to the features layer of dimension 512 and a linear classifier (a fully-connected layer followed by softmax) is trained in a supervised fashion for 100 epochs with a batch size of 256, learning rate of 0.3 and weight decay of 10^{-6}.

3.4 Results

During the experiments we tried to keep the same hyper-parameters for all the methods for the sake of comparability. For the same motivation, we excluded the MoCo framework since it uses a momentum encoder and it computes the contrastive loss with all the samples in the memory bank, while the methods taken into consideration here use the samples from the batch as negatives. NNCLR uses a memory bank too, but it is a support set from which it mines the positives. Our approach also uses a memory bank for the same purpose.

For the prototypes loss, we follow the authors of SwAV and use a $\tau_{\text{SwAV}} = 0.1$. The number of prototypes is 30 for STL-10 and 120 for Stanford Dogs dataset. On *Stanford Dogs* dataset we train the considered methods for 800 epochs, while on *STL-10* we train them for 200 epochs, since it has almost five times more the number of training images compared to the first one.

3.4.1 Positive mining with ground-truth labels

It is more encouraging to start the experiments by validating the main idea with the true class labels. We first want to prove the effectiveness of the positive mining module by substituting the SwAV prototypes with the class labels. In this way, however, we are not computing similarity scores between the batch elements and the memory bank, so we need to take a positive randomly from the support set. In fact, our positive mining module, defined in section 2.6.2, for each positive anchor, with a certain cluster assignment, takes the nearest neighbor among the samples in bank which have been assigned with the same positive cluster by the SwAV prototypes.

Using other positives from the same class is indeed better than standard SimCLR, SwAV and NNCLR on Stanford Dogs dataset. The k-NN accuracy reached after 800 epochs is 80.1%, while SimCLR, SwAV and NNCLR achieve

respectively 37.5%, 50.4% and 55.8%. This proves that there is room for improvement if we devise a better positive mining strategy.

Besides mining always true positives, we wanted to investigate the differences and the eventual benefits of drawbacks compared to a fully-supervised model. We thus obtained a *supervised training benchmark* for the comparison, where we used the ground-truth labels for training the prototypes layer with a categorical cross-entropy loss. In this case we are just training a single layer linear classifier on top of the backbone features. Since we already have such a linear layer, the prototypes layer, we leverage the same network architecture. Training the prototypes in a supervised manner leads to an accuracy of 96.5%, dominating with a large margin over all the other methods. We can immediately notice that the supervised benchmark performs over 16% better than the positive mining with ground-truth labels. We believe this is due to the difference between the two distributions of the underlying classes, since the supervised training has a more global overview compared to the positive mining approach. The classes seen by the supervised benchmark are associated in a bi-univocal way to the prototypes, since by using the labels we force the mapping from a given prototype to a certain class, similarly to a softmax classifier where each output unit represents the probability of detecting a given class. When using contrastive loss instead, we do not have this fixed mapping, losing the global perspective of the distribution of the classes. The classes are only used to pick correct positive pairs for the attraction in the representation space, pulling closer the two samples from the same class while pushing away all the other instances not belonging to the positive class.

We could not run the same supervised experiments on STL-10 dataset because the training set is not provided with any supervised label, only the validation set does.

3.4.2 Hard positive mining

Now that we have validated the effectiveness of mining better positives, we want to discard the "oracle" supervision from the labels and try to substitute its same function with the SwAV clustering module.

The results on ***Stanford Dogs*** dataset (reported in table 3.2) show that indeed our positive mining strategy is helpful at generating more informative positive pairs for the instance discrimation task, achieving a 3.1% improve-

ment over NNCLR. Since the only difference with respect to NNCLR is the additional consistency check for the semantic cluster, this may have helped at discarding more false positives when mining the nearest neighbor of the positive anchor. As a matter of fact, by detecting mined positives that are not true positives, i.e. with a different semantic label than the anchor, we can speed up the training convergence, possibly hindered by inconsistent pairs. In the same way *false negatives* could negatively affect the training, as seen in section 2.6.3, also false positives can degrade the performance. Contrasting samples from the same semantic class can discard the shared features which characterize all the instances of a given class.

Moreover, our hard positive mining method achieves about 8.9% higher accuracy than SwAV, meaning that in our model the SwAV prototypes could have benefited from the learning signal of the contrastive loss, and vice versa.

On the **STL-10** dataset, however, the results of all the methods are very similar (table 3.2). In this case the best method, SimCLR, is just 0.8% better than our approach. We theorize that since STL-10 has only 10 classes, which are more coarse-grained than the classes in Stanford Dogs dataset, we do not fully exploit the advantage of mining hard positives. A given batch may already provide a well-distributed set of examples, covering all the classes present in the dataset. A simpler method as SimCLR can perform as well as NNCLR: each class is quite different from the other classes, or the intra-class features shared by the samples among a certain class are easy to discriminate against the intra-class features shared by the samples of a different class. The contribution from the negative samples is also enough to form well-separated clusters of samples in the representation space, so pulling different samples of the same cluster closer in this space has no significant impact. On the Stanford Dogs dataset, instead, having more classes and smaller inter-class variance, the representation space may not present clear boundaries between groups of images with different semantic meaning. Therefore, enforcing positive pairs of the same class to be closer and more clustered in space is certainly more helpful for distinguishing between positives and negatives.

3.5 Analysis of the results

We have seen that hard positive mining is helpful to learn better representations, in particular on fine-grained datasets where the classes are very similar

kNN accuracy (%) on the dataset		
Method	**STL-10**	**Stanford Dogs**
Pre-training epochs	200	800
SimCLR[5]	66.1	37.5
NNCLR[12]	65.2	55.8
SwAV[4]	64.5	50.4
PosMining (Ours)	65.3	**58.9**

Table 3.2: **k-Nearest Neighbor classification results on Stanford Dogs and STL-10 datasets of self-supervised methods**. A k-NN classifier is trained on the features extracted by the backbone pre-trained with the various methods.

to each other.

Given the encouraging results from the pre-training of our proposed model on a fine-grained dataset such as Stanford Dogs, we wanted to test the quality of the learned features on a different domain. This time we freeze the features from the backbones pre-trained on the Stanford Dogs dataset and train a linear classifier on top, according to the linear evaluation protocol (described above in 3.3.2). For the linear evaluation tests we use the STL-10 dataset. Our positive mining approach achieves a top-1 accuracy of 52.9%, while SimCLR, NNCLR and SwAV achieve respectively 44.1%, 45.6% and 45.6%. This shows that the learned features, compared to those learned with the other methods, can also transfer better on simpler domains, where the inter-class variance is larger. Having already understood how to discriminate between more fine-grained features, the underlying causal factors of variance that generated the STL-10 domain are easier to determine when each class shows more representative features. Cross-training, however, is still less effective than in-domain training, where all the methods are pre-trained and tested on the same dataset. We can see the results of cross-training and in-domain training with linear evaluation on STL-10 in table 3.3. These results are coherent with the same conclusions presented in section 2.9 by the authors of "*When Does Contrastive Visual Representation Learning Work?*"[9].

Linear evaluation (%) on STL-10 dataset		
Pre-training dataset	**STL-10**	**Stanford Dogs**
SimCLR[5]	68.4	44.1
NNCLR[12]	63.2	44.6
SwAV[4]	66.2	45.6
PosMining (Ours)	66.4	**52.9**

Table 3.3: **Domain adaptation of features learned on Stanford Dogs to STL-10 dataset**. The methods are tested with linear evaluation protocol.

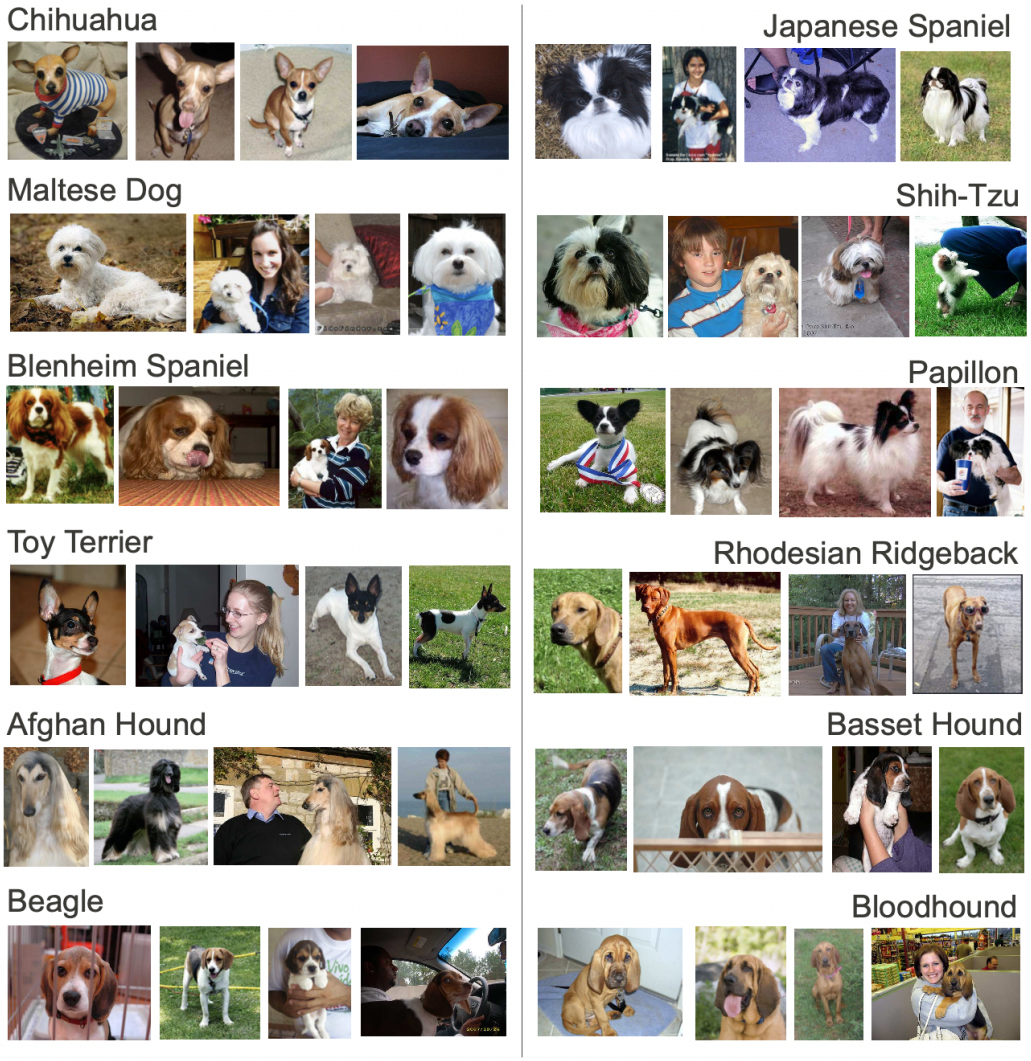

Figure 3.1: **Stanford Dogs dataset**[21]: four random example images from 12 of the 120 dog categories. We observe significant pose/visibility variation and background clutter in all the classes. In addition, there is large intra-class variation in appearance of the dogs. For example, the same species don not share a consistent color of fur or are of different sizes because of age (i.e. puppy vs fully-grown). Furthermore, their appearance is often modified by humans by placing articles of clothing or cutting/growing the fur.

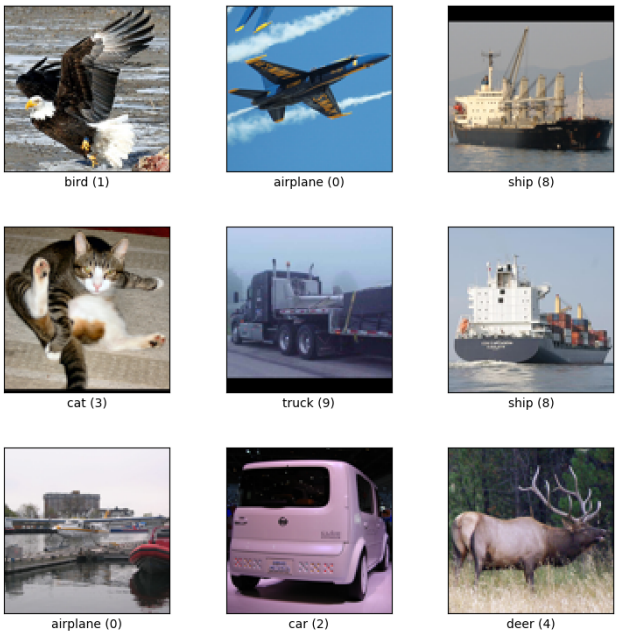

Figure 3.2: **STL-10**[1]: some of the classes used to create the unsupervised subset of the dataset.

4 Conclusions

In this work we have explored the promising results achieved by the self-supervised contrastive learning methods. We have provided a general overview of the contrastive learning frameworks proposed in the research literature and we have seen the main features of SimCLR, MoCo and NNCLR. Contrastive learning methods generally rely on strong data augmentations to generate self-supervision, but in most cases devising better strategies for mining positive and negative pairs can help to improve the quality of the learned representations, as demonstrated by NNCLR.

We also propose a new approach for mining hard positive samples, by combining NNCLR and SwAV. We use the prototypes layer from SwAV to get cluster assignments of the samples in the memory bank in order to mine different positives belonging to the same class. We discovered from our experiments on the STL-10 and Stanford Dogs dataset that positive mining is more beneficial on *fine-grained* domains, where the classes are similar to each other and require more details in order to be discriminated. Our proposed approach, in fact, is superior to other contrastive methods that do not perform explicit positive mining (SimCLR) or deploy a slightly different mining strategy (NNCLR).

We also tested the quality of the learned representations with out method and we saw that it adapts from "harder" to "easier" domains better than the other frameworks. To prove it, we trained a linear classifier according to the linear evaluation protocol on top of the frozen features learned on Stanford Dogs dataset.

The "toy" datasets we used in our work are relatively small, thus it would be interesting to experiment with larger ones like full ImageNet or Instagram 1B[37]. The latter is also more difficult since it is a "less-curated".

In this work we only experimented with positive mining approaches since they are less studied in literature than the negative mining strategies. We would set the future directions of our research to explore this branch as well,

in conjunction with the eventual benefits from removing false negatives in the batch. SwAV prototypes are not perfect but this method is a good starting point for self-labelling techniques. Improvements in this direction as well could help even more the power of mining informative positive samples.

References

[1] Andrew Y. Ng Adam Coates, Honglak Lee. An analysis of single layer networks in unsupervised feature learning. *AISTATS*, 2011.

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

[3] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. *CoRR*, abs/1807.05520, 2018.

[4] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *CoRR*, abs/2006.09882, 2020.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *CoRR*, abs/2002.05709, 2020.

[6] Xinlei Chen, Haoqi Fan, Ross B. Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *CoRR*, abs/2003.04297, 2020.

[7] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *CoRR*, abs/2011.10566, 2020.

[8] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546 vol. 1, 2005.

[9] Elijah Cole, Xuan Yang, Kimberly Wilber, Oisin Mac Aodha, and Serge J. Belongie. When does contrastive visual representation learning work? *CoRR*, abs/2105.05837, 2021.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[11] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. *CoRR*, abs/1505.05192, 2015.

[12] 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. *CoRR*, abs/2104.14548, 2021.

[13] Alon Faktor and Michal Irani. Video segmentation by non-local consensus voting. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2014.

[14] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[15] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. *CoRR*, abs/2006.07733, 2020.

[16] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 297–304, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[17] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. *CoRR*, abs/1911.05722, 2019.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[19] Tri Huynh, Simon Kornblith, Matthew R. Walter, Michael Maire, and Maryam Khademi. Boosting contrastive self-supervised learning with false negative cancellation. *CoRR*, abs/2011.11765, 2020.

[20] Yannis Kalantidis, Mert Bülent Sariyildiz, Noé Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning. *CoRR*, abs/2010.01028, 2020.

[21] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011.

[22] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *CoRR*, abs/2004.11362, 2020.

[23] Bruno Korbar, Du Tran, and Lorenzo Torresani. Co-training of audio and video representations from self-supervised temporal synchronization. *CoRR*, abs/1807.00230, 2018.

[24] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. 11 2013.

[25] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[26] Phuc H. Le-Khac, Graham Healy, and Alan F. Smeaton. Contrastive representation learning: A framework and review. *CoRR*, abs/2010.05113, 2020.

[27] Kwot Sin Lee, Ngoc-Trung Tran, and Ngai-Man Cheung. Infomax-gan: Improved adversarial image generation via information maximization and contrastive learning. *CoRR*, abs/2007.04589, 2020.

[28] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. *CoRR*, abs/1603.09246, 2016.

[29] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. *CoRR*, abs/1604.07379, 2016.

[30] Zhongzheng Ren and Yong Jae Lee. Cross-domain self-supervised multi-task feature learning using synthetic imagery. *CoRR*, abs/1711.09082, 2017.

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

[32] Nawid Sayed, Biagio Brattoli, and Björn Ommer. Cross and learn: Cross-modal self-supervision. *CoRR*, abs/1811.03879, 2018.

[33] Johan Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9:293–300, 06 1999.

[34] Yonglong Tian, Olivier J. Hénaff, and Aäron van den Oord. Divide and contrast: Self-supervised learning from uncurated data. *CoRR*, abs/2105.08054, 2021.

[35] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning. *CoRR*, abs/2005.10243, 2020.

[36] Feng Wang and Huaping Liu. Understanding the behaviour of contrastive loss. *CoRR*, abs/2012.09740, 2020.

[37] I. Zeki Yalniz, Hervé Jégou, Kan Chen, Manohar Paluri, and Dhruv Mahajan. Billion-scale semi-supervised learning for image classification. *CoRR*, abs/1905.00546, 2019.

[38] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. *CoRR*, abs/1603.08511, 2016.

[39] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017.

[40] Rui Zhu, Bingchen Zhao, Jingen Liu, Zhenglong Sun, and Chang Wen Chen. Improving contrastive learning by visualizing feature transformation. *CoRR*, abs/2108.02982, 2021.

Acknowledgements

First of all, I would like to express my gratitude to my supervisor, Professor Samuele Salti, for presenting me the opportunity to work on my thesis in a new and exciting environment.

I am also extremely thankful to my co-supervisor, Andrea Lonza (Addfor S.p.A), for his precious support. His constant feedback helped me refining my ideas and his considerable experience in this field made my debugging process less stressful.

I wanted to thank Rosalia and Enrico from Addfor S.p.A, for guiding me through all the troublesome regulatory procedures between me and my graduation.

My gratitude also goes to all the members of my extended family, of course, for encouraging me during all the journey.

A special thank, finally, goes to my life partner, Giulia, who has always been there to support me with her unconditional love.