TARAS SHEVCHENKO NATIONAL UNIVERSITY OF KYIV

Faculty of Computer Science and Cybernetics

Department of Mathematical Informatics

QUALIFICATION WORK

for obtaining a master's degree

in training direction 122 Computer Science

on the topic:

FEW-SHOT LEARNING FOR ONLINE LARGE SCALE FINE-GRAINED IMAGE RECOGNITION

Made by 2nd year student

Holubakha Mykyta Ihorovich

_____
(signature)

Academic adviser:

Professor, Doctor of Physical and Mathematical Sciences

Tereshchenko Vasily Mykolayovych

(signature)

_____

(signature)

I certify that in this work there are no borrowings from the works of other authors without the corresponding references (signature)

Student_____

The work was considered and allowed to be defended at the session of The Department of Mathematical Informatics

«___» _____ 2021 y..

Protocol №____

Head of The Department

V.M. Tereshchenko

(signature)

_____

Kyiv-2021

ABSTRACT

Total page count 58, figure count 22, information source count 45.

FINE-GRAINED IMAGE RECOGNITION, CLASSIFICATION, FEW-SHOT LEARNING, ZERO-SHOT LEARNING, METRIC LEARNING, EMBEDDINGS, ONLINE LEARNING, OUT OF DISTRIBUTION DETECTION, META LEARNING, OPEN WORLD RECOGNITON, SELF-SUPERVISED LEARNING, SEMI-SUPERVISED LEARNING, CLASS DISCOVERY, COMPUTER VISION, IMAGE RECOGNITION, CONVOLUTIONAL NEURAL NETWORK

The object of the work is solving the task of online large-scale image recognition.

The subject of the work is application of the few-shot learning methods to solving the aformentioned problem.

Development methods: training neural networks, transfer learning.

Development tools: Python programming language, PyTorch neural network framework, Vim text editor, Google Colab.

Results: existing methods of few-shot learning, including generative modeling, embedding learning and self-supervised learning were analyzed quantitatively and qualitatively. An architecture of an online learning system for large-scale image recognition based on few-shot learning was proposed.

The solution is still in progress of development and will be deployed in the industry for the task of classifying products in a retail setting.

CONTENT

ABBREVIATIONS

API — Application Programming Interface

BYOL — Bootstrap Your Own Latent

CNN — Convolutional Neural Network

CPC — Contrastive Predictive Coding

CUB — Caltech-UCSD Birds

FSL — Few-Shot Learning

GPU — Graphical Processing Unit

LSTM — Long Short-Term Memory

MoCo — Momentum Contrast

NCA — Neighborhood Component Analysis

NCE — Noise Contrastive Estimation

OOD — Out-Of-Distribution

reID — ReIdentification

SimCLR — Simple Framework for Contrastive Learning of Visual Representations

SKU — Storage Keeping Unit

SOP — Stanford Online Products

SwAV — Swapping Assignments between multiple Views

INTRODUCTION

Current state of the object of development. Object recognition is one of the fundamental problems in computer vision. It involves finding and identifying objects in images, and plays an important role in many real-world applications such as advanced driver assistance systems, military target detection, diagnosis with medical images, video surveillance, and identity recognition. The are quite a few high-performing well-known methods for the problem of image classification, with new methods, which incrementally improve upon the state of the art, consistently appearing on all major computer vision and machine learning conferences.

In some very narrow sense, image classification can be considered a solved problem, with state-of-the-art models having long surpassed human performance on traditional benchmark datasets like ImageNet, and MNIST, which either have a very simple problem structure, or a very large amount of data (ImageNet has over 14 million images for 20000 categories).

Unfortunately, this rosy view invariably breaks down when venturing further ahead from classical problem settings which provide practitioners and researchers well-known and distinct target classes, have a large number of labeled images for each class, having all the target classes known in advance and significantly dissimilar. If we embark upon this journey leading us closer to data which practitioners have and use to solve their real-world problems (with lots of noise, low amount of labeled samples, extremely large amount of classes with miniscule differences, significant class imbalances, with some labels having very few, or even zero known images), we will encounter a number of very challenging problems: few-shot learning, learning from noisy data, semi-supervised learning, zero-shot learning, class discovery, fine-grained image learning etc. Alas, these problems are a very open challenge yet, which is a bit far from the academic mainstream today, although their popularity is slowly rising; new methods, built upon newly developed convolutional architectures, slowly improve the state of the art in the field.

The problem remains extremely difficult when the dataset categories are nearly identical in terms of their visual appearance. Fine-grained object recognition aims to identify subcategory object classes, which includes finding subtle differences among visually similar subcategories such as dog breeds, product brands, car models, etc. The differences between classes are often small but always visually measurable, making visual recognition challenging, but possible. In addition, visual obstructions may cover distinguishing features, which would further complicate the problem.

Relevance of work and grounds for its implementation. With large advances in the field of metric learning in the recent years, it is becoming possible to develop even more accurate and interesting models for zero-shot fine-grained image recognition. Additionally, Alibaba has released a significant dataset: AliProducts, which contains 3 million image of over 50000 classes of retail products. It is both a great training dataset for metric learning, and a great and flexible benchmark.

Therefore, we decided to research and implement an online large-scale fine-grained image recognition system using few-shot learning.

The purpose and tasks of work. We consider the problem of developing an online large-scale fine-grained image recognition system using few-shot learning for retail product recognition. The purpose of the work is to build such a system, which would analyze supermarket shelf images and extract unique product classes.

We set the following tasks to achieve the goal:

- Investigate existing methods for few-shot learning
- Develop and evaluate models on existing benchmark datasets
- Evaluate domain transfer and generalizing from a public benchmark dataset to a proprieatery dataset with significantly different distribution.

Possible areas of application. While the work has a large number of possible area of application, the author professionally tackles the challenging problem of

recognizing retail product images, and is pondering on the solution of the problem in this work.

Intermediate results were published in the “Shevchenko Spring 2021” conference.

SECTION 1. STATEMENT OF THE PROBLEM

In this section, we will analyze, research and describe the existing body of work tackling the challenging problems related to open-world classification: metric learning, few-shot learning, zero-shot learning, fine-grained image recognition and class discovery.

Additionaly, we describe the public benchmark datasets commonly used in the filed of metric learning, as well as the proprietary dataset collected by the author, which would be used to develop and evaluate classification systems further in this work.

1.1. Fine-grained image recognition.

With the advancement of the field of deep learning, fine-grain image classification has received more and more attention, with a large number of deep-learning based approaches proposed in recent years. Fine-grained recognition aims to distinguish objects from different subcategories within a general category, or, alternatively, from a number of very similar categories with marginal visual differences that may be difficult to notice even for humans. In addition, changes of viewpoints, or scales, different backgrounds, occlusion can cause large variations within the same subcategory. This makes the problem very challenging.[1]

Figure 1.1. — CUB-200 dataset example[2]

A common benchmark example is the CUB-200 challenge. The goal is to classify bird species, which can be difficult even for human experts. Categories in the dataset have large intra-class variance and low inter-class variance. [2]Similar problems occure in the retail settings, when the datasets contain photos of very similar products taken from different viewpoints, light conditions and resolutions.

1.2. Few- and zero-shot classification.

Few-shot learning problems are machine learning tasks were the training dataset contains very few samples with a specified target variable.[3]

They are mainly stated as supervised learning problems. Few-shot classification, for example, attempts to learn classifiers with input data containing only a few labeled samples per class. It can be applied to image classification, sentiment classification on short texts and object recognition.

One most commonly considers N-way K-shot classification, where the training dataset consists of K examples per each training class N.[3]

1.3. Metric learning.

The goal of metric learning in computer vision is in finding a similarity measurement on pairs of images which preserves a desired distance structure. This can be used to improve performance of image search, especially with a large or unknown number of categories. Classical methods researched the task of finding a better Mahalanobis metric in Euclidean space. [4]

With the ability of directly learning non-linear representation, deep metric learning has achieved state of the art results in tasks like visual product search, feature image, fine-grained classification, face recognition, zero-shot learning and collaborative filtering. [4]

The majority of the work in deep metric learning can be summarized as optimizing either the contrastive loss (like in Siamese nets)[5] or the triplet loss[6]. It has been observed that directly optimizing such objectives in deep learning is very difficult, requiring special training procedures like triplet mining [7]or multitask learning[4].

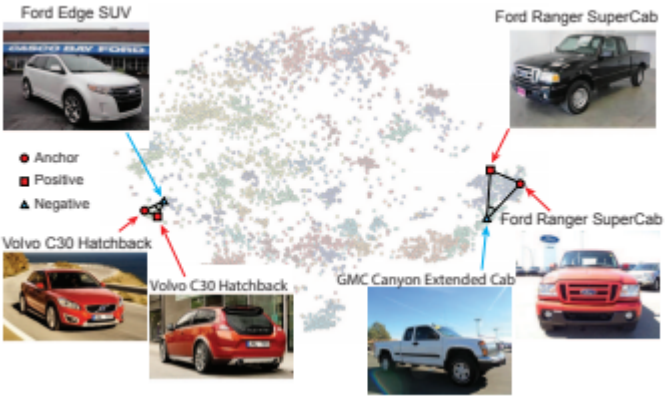

Figure 1.2 — Triplet loss example[8]

1.4. Open world recognition

The open world recognition framework was proposed in 2015 by Bendale at al.[9] It was an attempt to supersede the dominant classification methods which assume a static setting, e.g. that the number of classes and training images if fixed and determined during the initial training. It tries to capture the intrinsically dynamic nature of real-life recognition, like in scenarios where one cannot make an apriori prediction on the number and types of objects the system will have to recognize.

Open world recognitions are differentiated from standard visual classification algorithms in the following key features:

- ability to remodel known classes when the data arrives;
- ability to detect and learn unseen categories without retraining;
- ability to detect whether an image belongs to the seen categories.[10]

The requirement to dynamically update the label set favors metric learning approaches.

To properly model open world recognition scenario, one needs to learn the embedding and the novelty detector concurrently with receiving new instances and classes, instead of doing it initially from a closed set.[10]

1.5. Datasets

With the rise of popularity of metric learning in the last few years, new datasets have been appearing specifically for the task: large-scale fine-grained datasets with few examples per class and significant class imbalance.

During the course of the work we have used three datasets for empirical, quantitative and qualitative evaluation of trained models: Stanford Online Products[11],

AliProducts[12] and our proprietary dataset of supermarket shelf images collected in Germany and Ukraine.

SOP dataset was collected by using the web crawling API from eBay.com to download images. Authors filtered duplicate and irrelevant images (i.e. photos of contact phone numbers, logos, etc). The preprocessed dataset had 120,053 images of 22,634 online products (classes) from eBay.com. Each product has approximately 5.3 images. For the experiments, they split 59,551 images of 11,318 classes for training and 60,502 images of 11,316 classes for testing [11].

AliProducts dataset was published for a large-scale fine-grained image recognition competition which is held as a part of the Retail Vision Workshop[19] at CVPR 2020. It consists of similar isolated product images, colected by Alibaba. In addition, data was noisy, with both noisy labels and categories. The dataset consists of approximately 3M images of 50000 different retail products, with hierarchical annotations. The products are in SKU categories. Additional challenges are arisen by class imbalances [12].

Our proprietary dataset was collected in June 2019 - April 2020 and consists of 2000 unlabeled images of Ukrainian supermarket shelves with different products collected in different stores by the author and his colleagues. It was expanded with a number of images from German stores unseen before during the end of the period to assess generalization and domain transfers. Object detection, based on the RetinaNet[] network with EmMerger postprocessing[] was performed, and the extracted ROIs were used as the input data for the classification system.

Here are some examples of the images from each dataset:

Figure 1.3 — Images from SOP, AliProducts and our datasets

SECTION 2. FEW-SHOT LEARNING

In this section, we will conduct a high-level overview of the methods used to solve the few-shot learning problem.

2.1. Few-shot learning

As stated earlier, few-shot learning problems are machine learning tasks were the training dataset contains very few samples with a specified target variable (oftentimes 1 or 5). [3]

Most researched problems in this area are supervised. The most popular one is the few-shot recognition, which consists of learning a classifier with a small amount of labeled examples per each class. Another well-researched problem is few-shot regression, though we will not further elaborate on this topic.

As employed today, FSL currently fulfills three main goals:

- A pretext task (prerequisite) for human-like learning. Popular tasks include generating unseen samples of seen classes, as well as the few-shot classification. The field is also called meta-learning.[3]
- Long-tail recognition. As it is obviously impossible to collect large amounts of data for rare samples, few-shot learning can be used to train classifiers where such a need arises.
- Reducing data annotation expenses. Few-shot learning require less effort spent on annotating examples. In addition, the methods perusing few labeled samples and extracting prior knowledge from other classes generalize more successfully.

Related problems include:

- Weakly supervised learning [15]. It learns from experiences only with weak supervision (like incomplete, noisy, inaccurate, inexact supervised information). The most relevant one is the weakly supervised learning learning with incomplete

supervision. Only a small amount of samples has the target variable information. These methods are classified into semi-supervised and active learning approaches depending on whether an oracle or expert human are involved.

- Imbalanced learning. It involves learning from a skewed target variable distributions, like when some values of y are very rare (e.g. fraud or catastrophe detection). It trains and tests to consider all possible ys. In contrast, Few-Shot learning trains and tests with few examples for each y value, while considering other samples as priors.
- Transfer learning. It transforms knowledge in the source domain/task with a huge amount of training data, to the target domain/task having scarce training data.
- Domain adaptation. It is a subtype of transfer learning in which the tasks are the same but the domains differ. E.g. in sentiment analysis the source domain data may contain movie reviews, while the target domain is product reviews. Transfer learning methods are often used in FSL for transferring prior knowledge from the source task to the few-shot task.
- Meta-learning improves the performance of a new task T by the provided data set with the help of meta-knowledge extracted across all the tasks by a meta-learner. Meta-knowledge is the generic information gradually learned during all the tasks. Learner applies the meta-learner to a new task T leveraging some task-specific information. These methods can be applied to the FSL problem.

2.2. Few-shot learning methods

Let us now examine the types of approaches used in the area of few-shot learning.

2.2.1. Multitask learning

Multitask learning [16] is an approach involving learning multiple related tasks simultaneously. This allows the model to exploit at the same time both task-generic and task-specific information. This approach can obviously be used in the FSL context. Let us further elaborate on its applications.

Let there be C related tasks $T_1, \ldots, T_C$, with different-sized datasets. Some of the tasks will have a huge training set, while some will have a small one. Thus among the tasks, we denote the few-shot tasks as target tasks, while the rest as source tasks. As they are jointly learned, the resulting parameters for each target task are constrained by the source tasks. An obvious benefit is the possible prevention of overfitting.

The methods can be broadly classified as parameter sharing methods and parameter tying methods.

The first ones directly share a part of the parameters (often called the backbone) between the task. An example is being used for generic feature extraction on the backbone, while specializing the final layers for different output types.

In this setting, target tasks may only update the target-specific shared parameters, while source tasks influence both. [3]

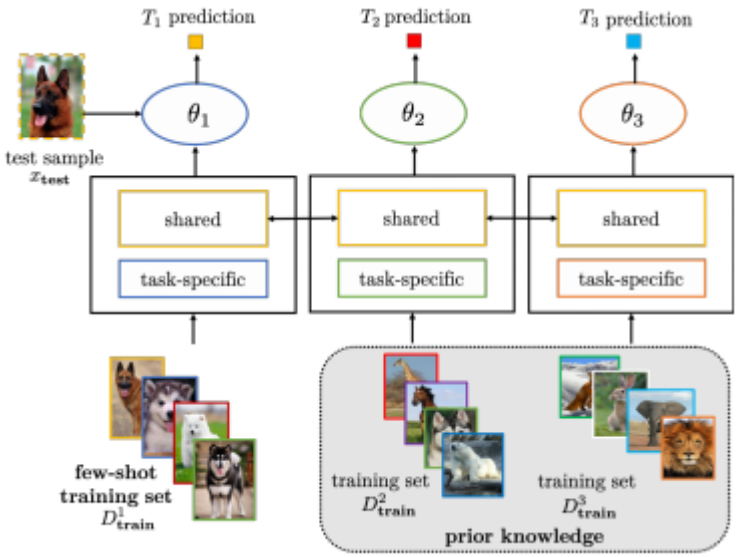

Figure 2.1 — Parameter sharing methods [3]

On the other hand, tying methods do not share the parameters between the tasks. Instead, they are making them as close as possible e.g. by adding a weight difference penalty term to the loss.

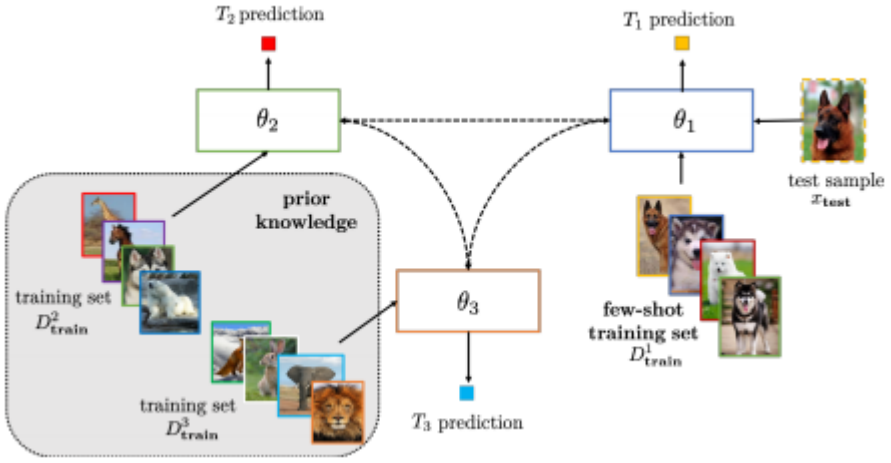

Figure 2.2 — Parameter tying methods [3]

2.2.2. Embedding learning

Embedding learning learns a mapping from samples to a lower-dimensional embedding space, aiming to keep similar samples close and dissimilar ones apart.

This approach has the following key components:

1. An embedding f from test samples to the latent space.
2. An embedding g from the training function to the latent space.
3. A similarity function s measuring the similarity between $f(x_{test})$ and $g(x_{train})$.

The test samples are thus assigned to the class of the most similar training point in the latent space. Although one may use a shared embedding function, learning two separate ones may produce better results. We will further elaborate on the embeddings in SECTION 3.

Depending on whether the parameters of the embeddings vary across the tasks, we classify them as using either (i) a task specific model; (ii) general embedding model (task-invariant); (iii) a hybrid embedding model capturing both task-specific and task-invariant information. [3]

2.2.2.1. Task-specific methods

Task-specific methods learn an embedding adapted to each task using only task-specific information. This increasing the number of useful training samples in some cases allowing to learn an embedding function only using task-specific information. This approach has obvious limitations.

2.2.2.2. Task-invariant methods

Task-invariant methods learn a general embedding function from a very large data set with sufficient samples for various kinds of outputs. They then directly apply these on new few-shot training datasets without retraining the network. Such embeddings may be learned using Siamese [5] or Triplet [6] networks.

Even though task-invariant embeddings do not update the parameters of the embedding model using the few-shot training dataset, many methods simulate the scenario while training it. Let’s assume we have C training datasets which have N

classes each. In each training dataset we only sample U out of N classes during training, while maximizing the performance on the remaining N - U classes. This allows to learn good results with generalization for few-shot tasks.

Recently, increasingly sophisticated embeddings are learned using the following meta-learning methods:

- Matching nets and their variants meta-learn different types of embeddings for train and test samples. Researches propose resLSTM [17] architecture for matching between the mappings. Some approaches combine with active learning by adding a sample selection step, which would use most beneficial unlabeled samples to extend the training dataset.
- Prototypical networks. [18] As opposed to comparing each $f(x_{test})$ with each $g(x_{train})$ this models only compare them with the class prototypes in the training dataset, where the prototype may be the mean of the points belonging to the class. This leads to more stable results and lower computation costs. Semi-supervised variants exist which increase the training dataset using soft-assigned unlabeled samples. [19]
- Relation Nets contain a relation module after the embedding. Their goal is to make the distance function between the points learnable as opposed to a predefined hyperparameter. The input to the relation module consists of an embedding of the query image, and the set of class prototypes, obtaining a relation score for each class. [20]

2.2.2.2. Hybrid embedding models

Although task-invariant embeddings can be very easily applied to new tasks with a negligible computation cost, they do not use any task-specific knowledge. This may not be a viable method for small-scale datasets, especially for long-tail recognition. To alleviate this, hybrid embeddings improve on a generic task-invariant prior embedding

using the task-specific information. This is implemented by learning a function which would map the examples to the parameters of the embedding.

Learnet [21] is an improvement to the task-invariant convolusional siamese network. It incorporates task-specific data from the training dataset. It obtains a meta-learner using multiple meta-training sets, converting training examples to parameters of the learner (the authors used a conventional siamese CNN). Thus the parameters of the embeddings would change for each given training sample, obtaining a hybrid embedding. As another improvement, the classification layer may be replaced by a ridge regression, to allow obtaining the parameters in closed-form. Some approaches train a function which generates the parameters for the full prototypical network taking into account the class prototypes and input data distribution.

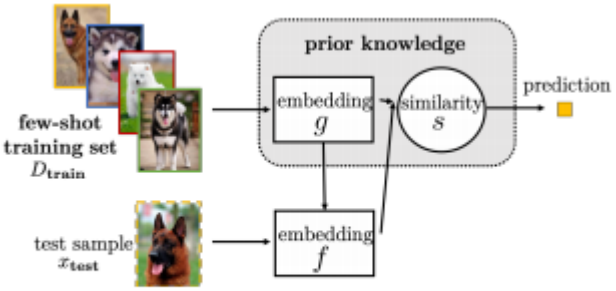

Figure 2.3 — Hybrid embeddings [3]

2.2.3. External memory learning

Learning with external memory may be used to extract the knowledge from the training dataset into external memory. Each new sample is then represented by a weighted average of extracted contents. This limits the representation by the contents in memory thus lowering resource usage. Let the memory be $M \in R^{b \times m}$, with each of its b memory slots $M(i) \in R^{m}$ consisting of a key-value pair M(i) = (Mkey(i), Mvalue(i)). Samples are first processed by an embedding function f. However the result is not used directly as the representatio. It is only used to query for the most similar samples in the

memory instead. Their values are then extracted and combine to generate the resulting representation. It is then fed into a simple classifier to generate a prediction.

As modifying the memory is expensive, it is usually small. When it isn't full, new samples can be stored in vacant slots. One has to choose displaced slots otherwise.

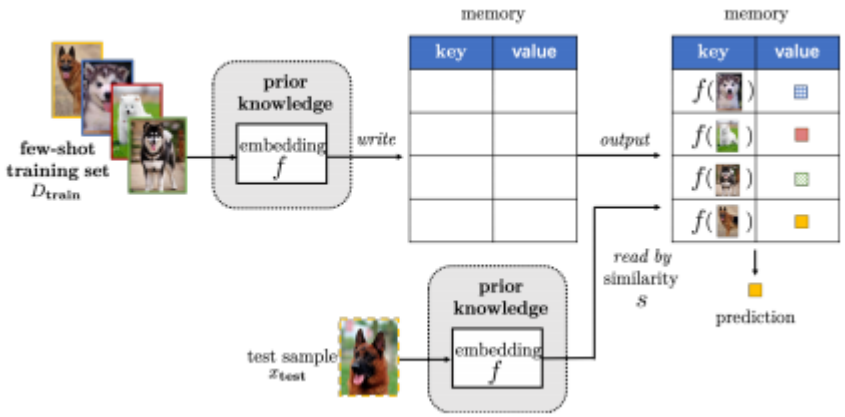

Figure 2.4 — External memory learning [3]

2.2.4. Generative modeilng

Generative modeling methods can be applied to estimate the probability distribution p(x) of the input data with the help of prior knowledge. It usually involves estimating p(x|y) and p(y) as well. This methods can deal with a large amount of tasks, including novel sample generation or recognition, reconstruction, flipping, denoising etc.

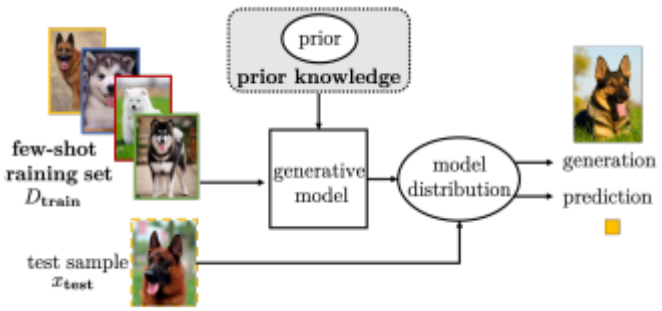

Figure 2.5 — Generative modeling [3]

Generative modeling is performed with the assumption that the observed variable is drawn from a $p(x; \theta)$ distribution parametrized by θ. We usually define a latent variable $z \sim p(z;\gamma)$, so that $x \sim \int p(x|z; \theta)p(z;\gamma)dz$. There prior distribution learned from other dataset is the source of the prior knowledge which is required for the FSL tasks. By drawing the provided training with $p(z;\gamma)$, we greatly constraint the resulting posterior probability distribution. Generative methods are grouped into three types depending on the meaning of the latent variable. [3]

2.2.4.1. Decomposable components

Even though supervised samples are scarce in FSL, they may share some smaller decomposable components within the samples coming from the other tasks.

A demonstrative example is few-shot learning recognition. Even though finding similar faces may be a hard problem, finding photos with similar components like noses, mouths or eyes is much easier. With a large enough number of samples, one can easily learn the models for each of the components. One thus only needs to detect the correct values of the components to decide on the target class. [3]

2.2.4.2. Groupwise Shared Prior

Similar tasks tend to have similar prior distributions which can be exploited in FSL. For example, let’s consider the classification of “orange cat", “leopard" and “Bengal tiger". They are similar, but the Bengal tiger is endangered, while orange cats and leopards are abundant. Thus, one can learn the distribution for the former ones and use this as a prior for the few-shot “Bengal tiger” class. [3]

2.2.4.3. Parameters of Inference Networks

To find the best θ, one has to maximize the posterior.

$$p(z|x;\theta,\gamma) = \frac{p(x,z;\theta,\gamma)}{p(x;\gamma)} = \frac{p(x|z;\theta)p(z;\gamma)}{\int p(x|z;\theta)p(z;\gamma)dz}.$$

Due to the integral in the denominator, it is intractable to solve. A variational distribution q(z; δ), which is learned from data, is then used to approximate p(z|x; θ,γ). It can be approximated efficiently using amortized variational inference. Although z no longer has any semantic meaning whatsoever, the powerful representations learned by these models can significantly increase the performance.

Once learned, the network can be applied to new problems directly, which is efficient and requires less human involvement. As the inference network has a large number of parameters, it is usually trained using some auxiliary large-scale data sets. [3]

SECTION 3. SURVEY OF METRIC LEARNING

In this section, we will perform a comprehensive review of the methods currently solved for the posed problems: metric learning, few- and zero-shot learning, class discovery, online and open-world learning.

3.1. Metric learning.

Let's provide a brief overview of the history of metric learning for the tasks of recognition. We will describe the methods in more detail afterwards.

At first, metric learning with deep networks was used for verification tasks. Afterwards, researchers applied them to face recognition and reID tasks.

First approaches used a learned feature embedding using a contrastive distance metric (so-called Siamese networks) for use with an ordinary classifier (either deep, or shallow). [4]

FaceNet [7] paper introduced the usage of triplet networks for the face verification and recognition tasks. The authors also proposed an approached for unsupervised clustering. Song et al. introduced structured feature embeddings [11]. We will elaborate more on this topics in the following chapters.

Approaches based on triplet ranking [22] and zero-shot learning with attributes [23] were proposed. Others tried to use WordNet to detect labels. These approaches, though, incur significant annotation efforts. We decided not to examine them closer.

As stated previously, most approaches learn either contrastive or triplet embeddings with different loss functions or specific training procedures like multitask learning (e.g. train embeddings and ordinary classification simultaneously). Let us further elaborate on these topics.

3.2. Metric losses.

Let's take a look at some of the existing loss functions for embeddings, which optimize different criteria.

Most of the criteria can be described as either classification-based or distance-based.

3.2.1. Contrastive loss.

Even though first siamese neural net architectures appeared in 1993 for the signature verification task [5], its applications for metric learning came much later. At first, they were used as a method of dimensionality reduction in 2005 [24].

This method was novel in combining all of the following properties in the learned embedding:

- being trained only on a binary equivalence relationship between training samples;
- invariantness to distortion;
- generalizations to unseen (during training) samples;
- smoothness of the learned mapping.

Training process typically consists of pairs of input samples X1, X2 with a label Y., which is set to 0, if samples belong to the same class/are similar/close, and 1, if the samples aren't.

The aim of the training is to learn an embedding which would separate dissimilar samples, while keeping similar ones close. Or, equivalently stated, minimizing inter-class and maximizing intra-class variance.

The following loss function was thus proposed, with Dw denoting the euclidean distance in the embedding space:

$$L(Y, W, X_1, X_2) = (1 - Y)\ 1/2\ (Dw)^2 + (Y)\ 1/2\ \{max(0,\ m - Dw)\}^2$$

This function is still sometimes used in practice, in addition it has great tutorial value. [4]

3.2.2. Triplet loss.

Another option (currently the most popular) is the triplet loss function, which is the base of many of the learning methods mentioned further.

It considers triplets of samples: anchor, positive and negative examples. The goal is to minimize the distance between the positive and the anchor sample, while keeping the anchor and the negative apart.

Let Xi, Xp, Xn denote the anchor, positive and negative samples respectively. Let Dw(x, y) — distance in the embedding space, m — the minimal acceptable intra-class margin.

The following loss function can thus be derived:

$$L(W, X_i, X_p, X_n) = Dw(X_i, X_p)^2 - Dw(X_i, X_n)^2 + m.$$

It should be mentioned that naively optimizing the loss across all the possible triplets is very inefficient. As the neural network would process a lot of easy samples which wouldn't really contribute to the result. Thus, the need for specific triplet selection procedures (also known as triplet mining) arises. They will be introduced in later chapters.

Is was first proposed in application to nearest-neighbors classifiers, and subsequently for face recognition in the FaceNet paper, which applied embeddings to face recognition and clustering. [7]

3.2.3. Soft triple loss.

In the SoftTriple loss paper [25], the authors took a look at the classical softmax loss, which can also be applied successfully sometimes in metric learning. They found out that it is equivalent to a smoothed triplet loss, assuming each class has a single center. They thus derived a new formulation, called the soft triple loss, which allowed each class to have several center points (which models intra-class variance better).

This approach allows to significantly improve the accuracy without requiring complicated triplet mining procedures.

Let us now describe the loss function.

Let S denote sample similarity:

$$\mathcal{S}'_{i,c} = \sum_k \frac{\exp(\frac{1}{\gamma}\mathbf{x}_i^\top \mathbf{w}_c^k)}{\sum_k \exp(\frac{1}{\gamma}\mathbf{x}_i^\top \mathbf{w}_c^k)} \mathbf{x}_i^\top \mathbf{w}_c^k$$

By smoothing the similarity we obtain:

$$\ell_{\text{SoftTriple}}(\mathbf{x}_i) = -\log \frac{\exp(\lambda(\mathcal{S}'_{i,y_i} - \delta))}{\exp(\lambda(\mathcal{S}'_{i,y_i} - \delta)) + \sum_{j \neq y_i} \exp(\lambda \mathcal{S}'_{i,j})}$$

3.2.4. Tuplet margin loss.

In the tuplet margin loss paper [26], the authors generalized the triplet loss to accommodate multiple negative samples Xi1, ...Xin from different classes.

It allows to formulate the following loss function (where d denotes the distance):

$$\mathcal{L}_{tuplet} = \log\left(1 + \sum_{i=1}^{k-1} e^{d(x_a, x_p) - d(x_a, x_{n_i})}\right),$$

Additionally, the authors found that this approach works with random triplet sampling, obviating the need for a computationally-intensive triplet mining scheme if the following loss function is used (where beta denotes the margin hyperparameter):

$$\mathcal{L}_{tuplet} = \log\left(1 + \sum_{i=1}^{k-1} e^{s\left(\cos\theta_{an_i} - \cos(\theta_{ap} - \beta)\right)}\right)$$

3.2.5. Angular loss.

While earlier metric learning approaches focus on optimizing either absolute (contrastive loss) or relative (triplet loss) similarity of image papers, the method proposed in the Angular loss paper is used to limit the angle at the negative points of triplet triangles. [8]

This allows to obtain the following properties not guaranteed by conventional methods:

- scale invariance;
- improved objective robustness (ageainst feature variance);
- capturing additional local structure of triplet triangles (via the imposed third-order geometric constraint).

  As a result, the convergence is much faster than in earlier methods.

  All in all, this method greatly sped up convergence of the algorithms.

$$l_{ang}(\mathcal{B}) = \frac{1}{N}\sum_{\mathbf{x}_a \in \mathcal{B}}\left\{\log\left[1 + \sum_{\substack{\mathbf{x}_n \in \mathcal{B} \\ y_n \neq y_a, y_p}} \exp\left(f_{a,p,n}\right)\right]\right\}$$

$$f_{a,p,n} = 4\tan^2\alpha(\mathbf{x}_a + \mathbf{x}_p)^T\mathbf{x}_n - 2(1 + \tan^2\alpha)\mathbf{x}_a^T\mathbf{x}_p.$$

3.2.6. Circle loss.

In the circle loss paper [27], the authors examine the common loss functions (like triplet and softmax with cross-entropy losses) and observe that most of them tries to

embed samples into similarity pairs and seeks to reduce the distance between the similar ones.

They consider these optimization criteria inflexible, as the penalty strengths on every single similarity scores are equal. The authors decided to emphasize the samples where the similarity score deviates too far from the optimum. To achieve this, they re-weigh each score to highlight the less-optimized ones, thus obtaining:

$$\mathcal{L}_{circle} = \log\left[1 + \sum_{j=1}^{L} \exp(\gamma \alpha_n^j (s_n^j - \Delta_n)) \sum_{i=1}^{K} \exp(-\gamma \alpha_p^i (s_p^i - \Delta_p))\right]$$

3.2.7. FastAP loss.

The authors of the FastAP loss paper [28] revisited the learning to rank approach. They aimed to optimize average precision over ranked lists of examples.

This method has a lower complexity in comparison to other methods, due to a new proposed minibatch sampling method.

The authors explore optimizing the approximated (by histograms) integral of the Average Precision score. The following loss function was thus obtained:

$$\text{FastAP} = \sum_{j=1}^{L} \frac{\frac{H_j^+}{N_q^+} \cdot \frac{N_q^+}{N}}{\frac{H_j}{N}} \cdot \frac{h_j^+}{N_q^+} = \frac{1}{N_q^+} \sum_{j=1}^{L} \frac{H_j^+ h_j^+}{H_j}$$

3.2.8. Lifted structured feature embedding loss.

A possible approach consists of increasing the amount of information extracted from each batch by using the whole pairwise distance matrix between the samples.[11]

$$\tilde{J}_{i,j} = \log\left(\sum_{(i,k)\in\mathcal{N}} \exp\{\alpha - D_{i,k}\} + \sum_{(j,l)\in\mathcal{N}} \exp\{\alpha - D_{j,l}\}\right) + D_{i,j}$$

3.2.9. ProxyNCA loss.

In standard metric learning approaches, the optimization criteria is a loss function defined on distances between anchor, positive and negative samples. Such methods can be challenging optimized. The main obvious issue is the requiment to find informative triplets, which necessitates large batch sizes, hard (or semi-hard) triplet mining, etc. Even this workarounds do not help the slow convergence rate of such methods. In the ProxyNCA paper[29], the authors explored optimizing the triplet loss using a different space of triplets. Instead of the positive and negative samples, they propose to to consider similar and dissimilar proxy points (which are also learned). The proxies approximate the original ones, so that the triplet loss over the proxy points is a tight upper bound for the loss computed with the original points. The authors observed that this loss is empirically better-behaved. It allowed them to improve on the state-of-the-art results on three standard zero-shot benchmarks by up to 15% points. At the same time, they converged three times faster than other triplet-based losses.

The objective is to minimize the loss function on proxy points in NCA formulation.

Algorithm 1 Proxy-NCA Training.

Randomly init all values in θ including proxy vectors.
for $i = 1 \dots T$ **do**
  Sample triplet (x, y, Z) from D
  Formulate proxy triplet $(x, p(y), p(Z))$
  $l = -\log\left(\frac{\exp(-d(x,p(y)))}{\sum_{p(z)\in p(Z)} \exp(-d(x,p(z)))}\right)$
  $\theta \leftarrow \theta - \lambda \partial_\theta l$
end for

Figure 3.1 — ProxyNCA Training

In the recent years, a lot of papers proposing different loss functions appeared, but we believe that the preceding once generally capture the variety of the current state of the art.

3.2.10. SimCLR loss.

The authors of the SimCLR paper [30] reported state of the art results on two widely-used benchmark data using a self-supervised method. Their approach is similar to ordinary siamese neural networks, except two different augmentations of the same image are used for training as positive pairs. We will describe the method in more detail in the later chapters.

They use the following loss function (Normalized Temperature-Scaled Cross-Entropy loss or NT-Xent):

$$\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}$$

.

sim denotes the cosine similarity function, and 1 is the indicator function of samples from the images different from the current one.

3.3. Sample mining

Let us now provide an overview of existing methods of triplet mining. They try to maximize the amount of information learned from each batch of the training data.

Naive methods (like random mining) can be computationally expensive, if random triplets are sampled for each batch, and inefficient, as the triplets can be too easy to learn anything useful, thus slowing down convergence.

All-triplet mining required $O(n^3)$ triplet evaluations for each minibatch.

To solve this, researches from all over the world dare proposing new triplet mining schemes. Triplet mining is the process of extracting useful triplets from the training data.

3.3.1. Batch-hard negative mining.

As mentioned earlier, very easy batches slow down convergence.

The notion of hard triplets can be used to solve this issue. A hard negative triplet is a negative triplet which is closer to the anchor sample than other positive ones and/or the margin. Analogously, an easy negative triplet can be defined.

In the paper “In Defense of the Triplet Loss for Person Re-Identification”, authors have noted that batch-hard mining (training on hard triplets) can significantly speed up convergence. A balance is required, though, as using very hard negatives can prevent meaningful training instead. [31]

Noisily labaled data can also induce significant issues with the training process (or act as a regularizer).

This approach allowed to gain significant improvements to state of the art approaches not using mining (over 20% points).

3.3.2. Semi-hard negative mining.

FaceNet authors also researched the danger of using very hard triplets. They observed, that they can lead the model to getting stuck in bad local optima. The authors introduced the notion of a semi-hard triplet to combat the issue. They define semi-hard examples as examples which are further away from the anchor than the positive examples but still hard, as they lie inside the margin.

They propose to select semihard examples from each minibatch. For good results, this approach requires huge batch sizes (author reported using 1800 triplets), where each batch consists of an equal number of samples from different classes. [7]

3.3.3. Easy positive triplet mining.

As we've seen earlier, most metric learning approaches use the strategy of pushing images from the same class as close as possible. In the Easy positive mining paper [32], the authors proposed a loosened embedding strategy that only requires the embedding function to map each training image to the most similar example from its class. This approach is called "Easy Positive" mining. It leads to more flexible embeddings which better generalize to unseen data. This mining strategy yielded state of the art recall performance (even beating ensemble methods with complicated loss functions) on image retrieval datasets like CUB, SOP, Hotels-50K, and In-Shop Clothes.

The main idea of the approach is that it allows to better model inter-class variance by tackling over-clustering of the embedding space. Easy Positive selection is more likely to push close positive together while being less likely to push far away positives together. It thus maintains the intra-class variances, allowing the classes to have manifold structure in the output space thus helping reduce the over-clustering problem (of mapping very different images to the same point). In addition, this approach empirically improves generalization.

SECTION 4. SELF-SUPERVISED LEARNING

In this section, we will perform a comprehensive review of the methods currently solved for the posed problems: metric learning, few- and zero-shot learning, class discovery, online and open-world learning.

Traditional supervised approaches rely on the amount of annotated training data. Even though large amount of data is available publically, most of it lacks annotations. It has thus spurred the line of research of leveraging non-annotated examples. This is the self-supervise learning task, where the data itself provides its supervision.

Supervised methods have other downsides too:

- generalization error caused by dataset biases or full variance not being captured;
- overfitting, and finding non-existent correlations due to unsatisfactory data;
- adversarial attacks.

Self-supervised learning methods integrate contrastive and generative approaches to utilized unlabeled data to learn underlying representations.[33]

Solving various pretext tasks is an approach to help learn features and then apply pseudo-labeling. Common pretext tasks are instance equivalence, inpainting, colorization, super-resolution, jigsaw, next frame prediction. These tasks can help obtain effective representations without large amounts of training data.

Pretext tasks are self-supervised tasks to learn representations pseudolabels. They are generally generated automatically based on the data. The learned model can be the used for downstream tasks like classification, detection with or without fine-tuning. For a contrastive pretext, the original image is picked as an anchor, while its augmented version servers a positive sample, while the rest of the images in the minibatch or in the training data serve as negatives. [30]

4.1. Contrastive predictive coding

CPC is an unsupervised learning method from high-dimensional data. It works by converting a generative modeling problem into classification. The r InfoNCE loss in CPC, adapted from Noise Contrastive Estimation (NCE), uses cross-entropy to measure how well the model can be used to predict the “future” representations from a set of unrelated “negative” samples. It is in part motivated by the fact that losses like MSE do not have enough capacity while learning a large generative model can be too cost-restrictive.

It uses an encoder to compress the inputs together with an autoregressive decoder to infer high-level context features that may be shared with all future predictions. The end-to-end training relies on the NCE-inspired contrastive loss.

While predicing future information, CPC is optimized to maximize the the mutual information between the input and irs context vector. [34]

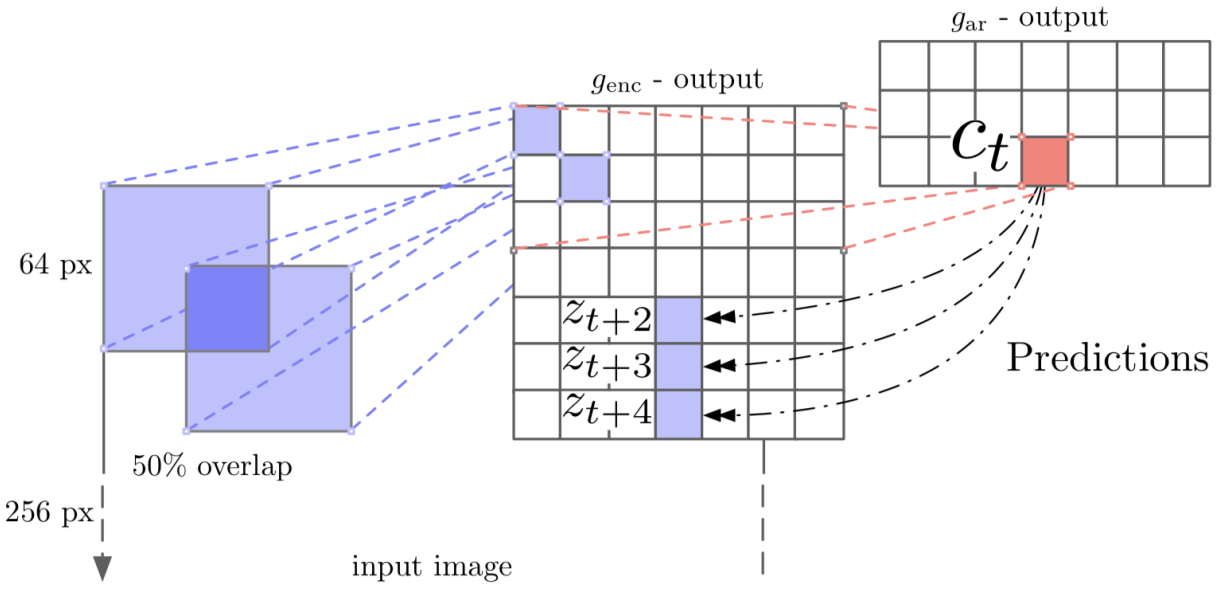

Figure 4.1 — Contrastive Predictive Coding [34]

4.2. MoCO

Momentum Contrast (MoCo) removes the need to store a representation of the whole dataset in the memory bank with the help of a dynamic memory queue. It provides a framework of unsupervised learning visual representation as a dynamic dictionary look-up.

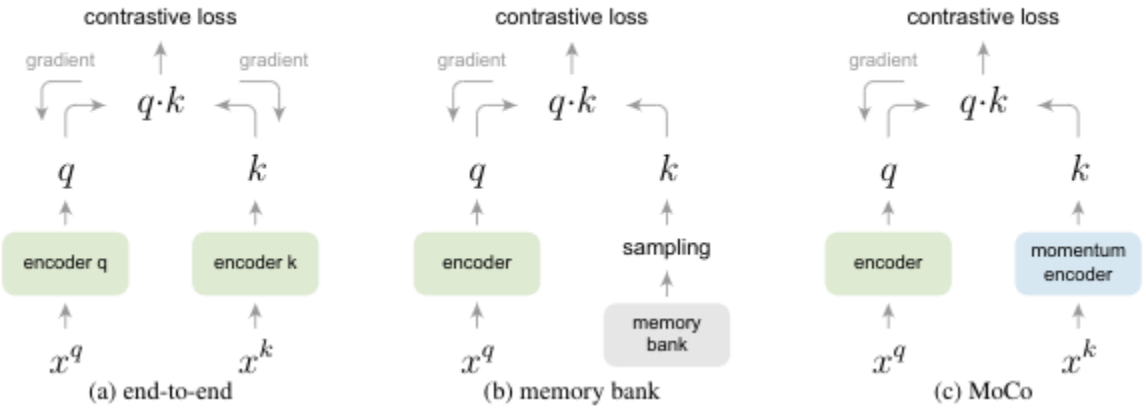

Figure 4.2 — Momentum Contrast [35]

Given a sample x, we can get a query encoding using the encoder f_q. Reference samples (key) are encoded by a momentum encoder k_i to generate a list of representations in the dictionary. Let k^+ denote a single positive key in the dictionary that matches q. It is created using a copy of x_q with a different augmentation. Then the InfoNCE contrastive loss is applied for one positive and negative samples. [35]

4.3. SwAV

Unsupervised representation have made great progress in getting closer to supervised training methods, especially with contrastive learning. Such methods typically require a large number of explicit pairwise feature comparisons online. It is rather computationally complex. SwAV can take advantage of them without requiring any pairwise comparisons. The method clusters the data while ensuring that cluster

assignments for different transformations (or viewpoints) of the same image are consistent, instead of comparing feature representations.

The authors propose a swapped prediction mechanism where the cluster assignment of a view is predicted using the representation of another one. It can be trained with arbitrary batches and scales to large amounts of data, while not requiring momentum networks or a large memory bank. In addition, the authors proposed augmentation strategies which save memory usage.

They achieved 75.3% top-1 ImageNet accuracy with ResNet-50, and surpassed supervised pretraining on all the transfer tasks they considered.

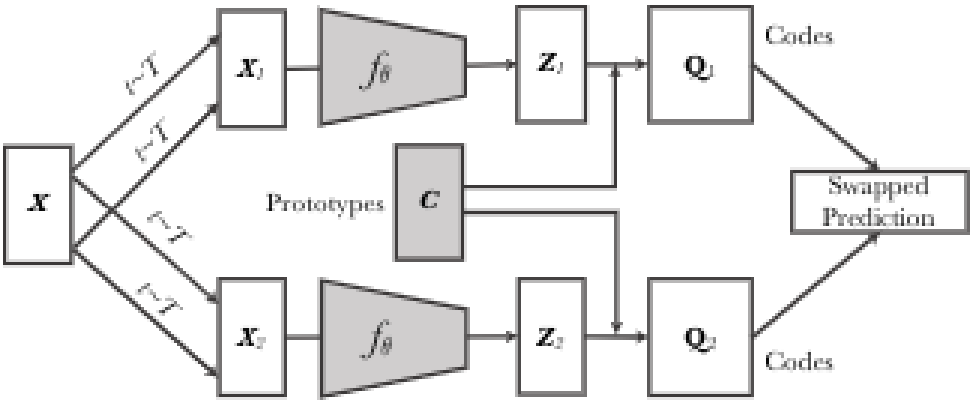

Figure 4.2 — SwAV [36]

4.4. SimCLR.

One of the working paradigms for few-shot learning while utilizing lots of unlabeled data is unsupervised pretraining with supervised fine-tuning. While this paradigm uses unlabeled data in a task-generic way, unlike common semi-supervised in CV, the authors demonstrate that this can be surprisingly effective on many benchmark datasets.

A key ingredient of their approach is using large (both deep and wide) networks during fine-tuning with large batch-sizes. The authors observed, that the effect from

deeper networks only increases the fewer the labels. After fine-tuning, the model can be distilled into a musch smaller one with little loss in accuracy, utilizing the unlabeled examples in a task-specific way.

The proposed algorithm can be stated as three steps: unsupervised pretraining of a large CNN, supervised fine-tuning on a small amount of labeled instances, and finally distillation with unlabeled samples to capture task-spcific knowledge within a smaller network.

The authors achieved 73.9% ImageNet top-1 accuracy with just 1% of labeled instances using ResNet-50, which is a 10x improvement in efficiency over SoTA. While with 10% of labels, they managed to beat then-current supervised approach with all of the labels with 77.5%.

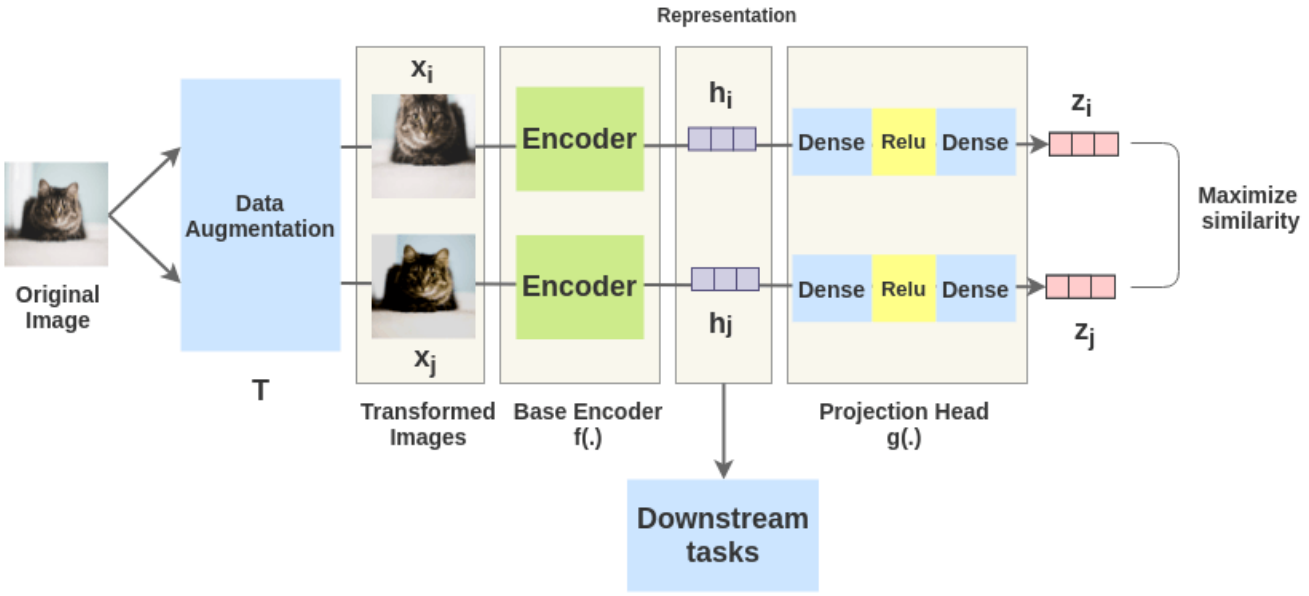

Figure 4.3 — SimCLR Training [30]

4.5. BYOL

Bootstrap Your Own Latent is another self-supervised representation learning approach. It trains a network to predict representations of different augmented views of

the input image. While not using negative pairs in its training objective, it doesn't collapse to the trivial constant solution. The goal is to learn a presentation that can be used in downstream tasks.

It relies on two networks, denoted as online and target networks, that learn from each other.

The online network consists of an encoder, a projector and a predictor.

The target network has the same network architecture, but with different parameters, updated by Polyak averaging.

Great performance without negative samples is quite surprising, though it has been noted that it performs no better than randomness if batch normalization is removed, which can imply that batch normalization may introduce some implicit negative samples.

Though later findings show that same results can be reproduced with batch-independent normalization scheme instead of batch normalization.

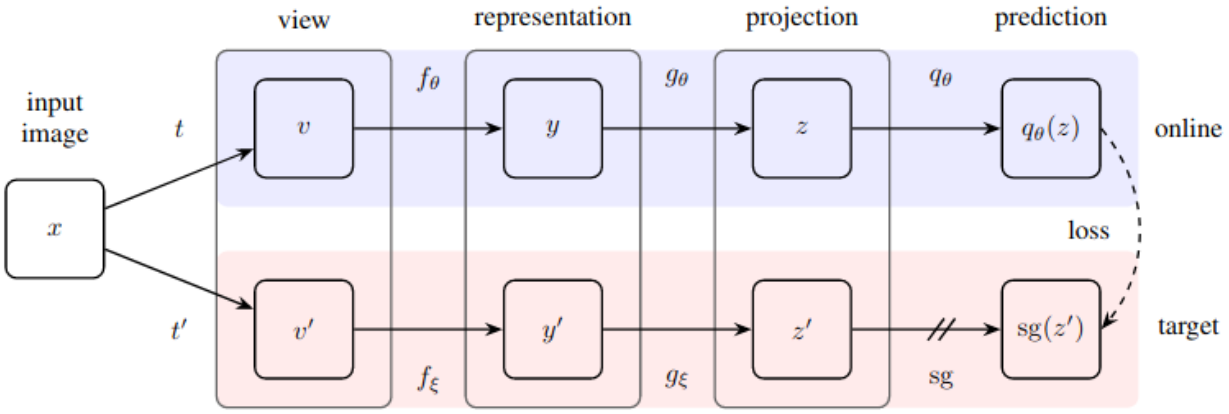

Figure 4.4 — BYOL [37]

4.6. SimSiam

In SimSiam, the authors observed that simple Siamese networks can learn meaningful representations which do not reduce to the trivial constant solution even using none of the following: (i) negative sample pairs, (ii) large batches, (iii) momentum encoders. They determined that a stop-gradient operation may be helpful in preventing the training collapsing.

They managed to achieve competitive self-supervised results on both ImageNet and downstream tasks.

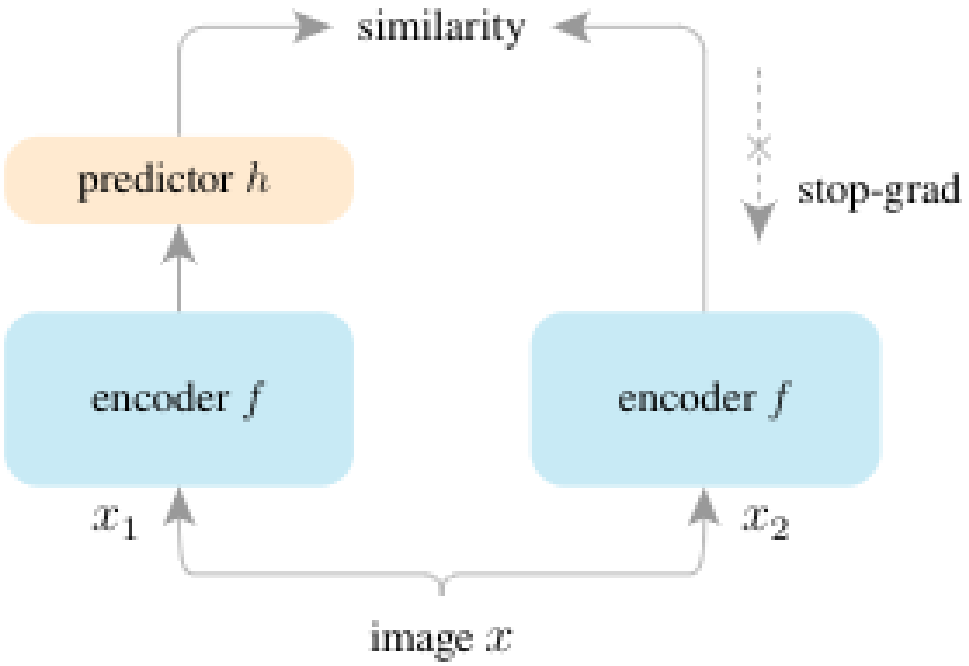

Figure 4.5 — SimSiam [38]

SECTION 5. PREREQUISITES FOR AN ONLINE LEARNING SYSTEM

In this section, we will perform a comprehensive review of the methods currently solved for the posed problems: metric learning, few- and zero-shot learning, class discovery, online and open-world learning.

5.1. Class discovery.

One promising approach to dealing with datapoints that are outside of the initial training distribution (OOD) is to create new classes that capture similarities in the datapoints previously rejected as uncategorizable. Systems that generate labels can be deployed against an arbitrary amount of data, discovering classification schemes that through training create a higher quality representation of data.

To recognize new classes in the data, two elements are needed: an OOD detector and a class creation scheme.

OOD detection can be implemented by a threshold on prediction confidence, while class creation — by a clustering algorithm.[39]

Carefully choosing which new class candidates are worth adding to the training dataset is important to preserving the quality of the training set.

In Semi-Supervised Class Discovery [40], the authors explored two major heuristics. The first is class learnability, which asks whether a model trained on the candidate classes can effectively learn to predict whether a datapoint is from the candidate class. The third is a measure of cluster density, assuming that denser clusters are more likely to contain datapoints that are similar to each other and so will have a higher quality (which turned out to be untrue).

They introduced a Dataset Reconstruction Accuracy metric for evaluating class discovery.

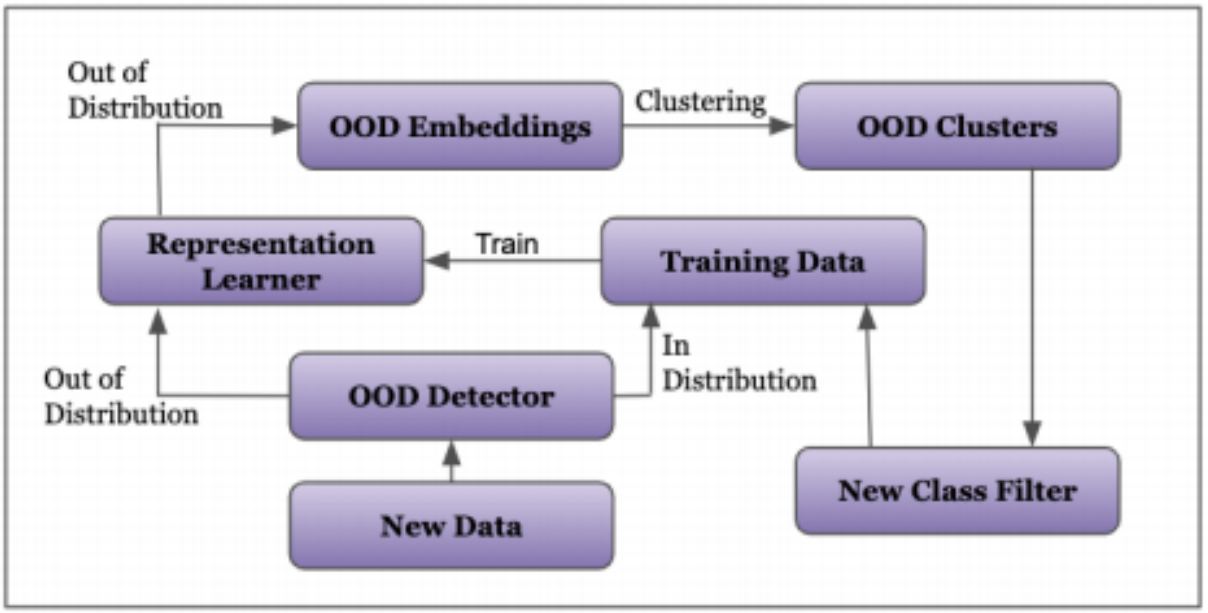

Figure 5.1 — Class Discovery mechanism[40]

SECTION 6. IMPLEMENTING AN ONLINE LEARNING SYSTEM

In this section, we will describe the architecture of a full-cycle online learning system and will then perform experimental evaluations of the methods that can be used to implement different parts of the system.

We will be taking a look at the results from popular benchmark datasets and our proprietary dataset.

6.1. Architecture of an online learning system

We aim to construct an online learning system applicable in the realistic open world classification problem setting, which means that the system should be dynamically updatable which includes the following abilities:

- ability to remodel known classes when the data arrives;
- ability to detect and learn unseen categories without retraining;
- ability to detect whether an image belongs to the seen categories. [10]

Thus we require at least the following components of the system: an out-of-distribution detector (to satisfy the third requirement), an out-of-distribution unsupervised classifier i.e. a clusterizer (to satisfy the second requirement), a generic feature extractor i.e. an embedding (general enough to differentiate between different unseen classes) and a dynamically updatable classifier (so that it can be quickly remodeled when adding or modifying classes).

The components should be arranged as follows. First, we extract the feature embeddings from the image using a learned feature extractor (which would be fixed). The embeddings would be passed into the out-of-distribution detector and, if detected as in-distribution, passed through to the classifiers.

As the data continuously flows in to the model, we would store the embeddings deemed out-of-distribution and a sample of the in-distribution samples. They will later be used to semi-automatically update the model. The out-of-distribution samples would be passed into the unsupervised classifier to detect new proposed class clusters, which the user would manually annotate as being a new example of an existing class (e.g. from a totally new viewpoint) or a totally new class. The in-distribution samples will simply be added to the classifier. After adding in the newly processed samples, we would run a rebalancing procedure to remove redundant points.

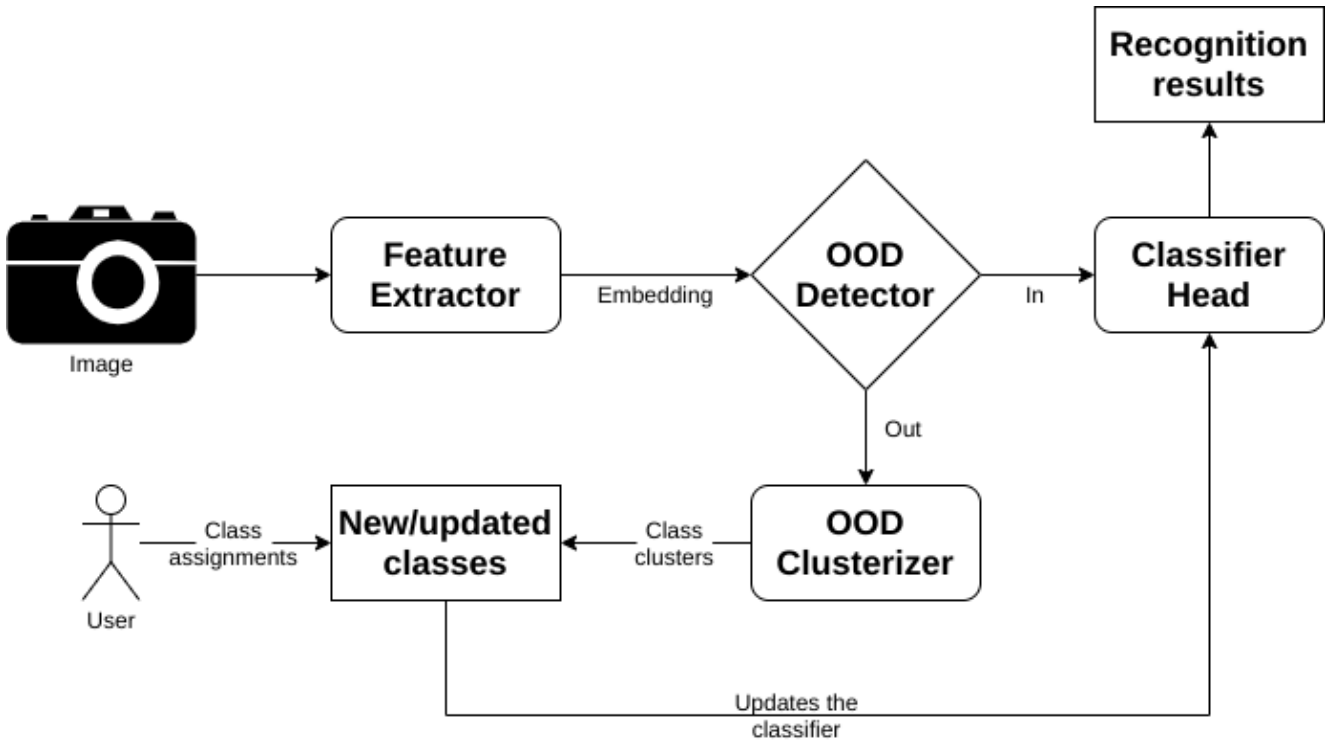

Figure 6.1 — Class Discovery mechanism

6.2. Method selection

As this work mostly concentrated on the task of learning a feature extractor, we would limit ourselves to a k-Nearest Neighbors classifier for the classification head. Not only is it obviously dynamically updatable, with a large number of robust and efficient implementations, the current body of research in the field, show that it is a

surprisingly effective approach, appearing very competitive to the sophisticated meta-learning based methods.

Our baseline out-of-distribution detector is a distance threshold to the nearest neighbors. Although his approach depends on a robust learned embedding, we feel that it is suitable for use in our work, as we mostly focused on the problem of learning a robust feature extractor.

Given a robust feature extractor, geometric methods similar to the ones specified in the Boundary-Based OOD Detector [41] paper may be useful for both detecting and clustering out-of-distribution samples, as it learns to arrange samples belonging to the same object into a smooth manifold on a hypersphere, so that we may use geometric methods as a baseline to cluster/detect new classes.

After reviewing the state of the art in the fields of few-shot and metric learning, we have decided that the most practical method for feature extraction is training either contrastive losses-based methods, or self-supervised classification methods like SimCLR[30] and BYOL[37].

6.3. Evaluating methods for learning feature extractors

We will now perform an experimental review of a number of state-of-the art methods of learning feature extractor.

All the experiments were performed in the Python 3 Google Colab environment with NVIDIA Tesla P100 GPUs and a time-limit of 24 hours per experiment.

We have examined PyTorch implementations of a selection of metric learning methods reviewed earlier. We have performed experiments evaluating their performance on the Few-Shot Classification in CIFAR-10[42] and on our proprietary dataset.

6.3.1. Self-Supervised Learning For Few-shot Image Classification

We have evaluated the performance of an author PyTorch implementation of this paper on the CUB and CIFAR-10 in the 5-shot 5-way formulation.

We have used the AmdimNet architecture proposed im the paper.

The authors proposed combining self-supervised and meta learning.

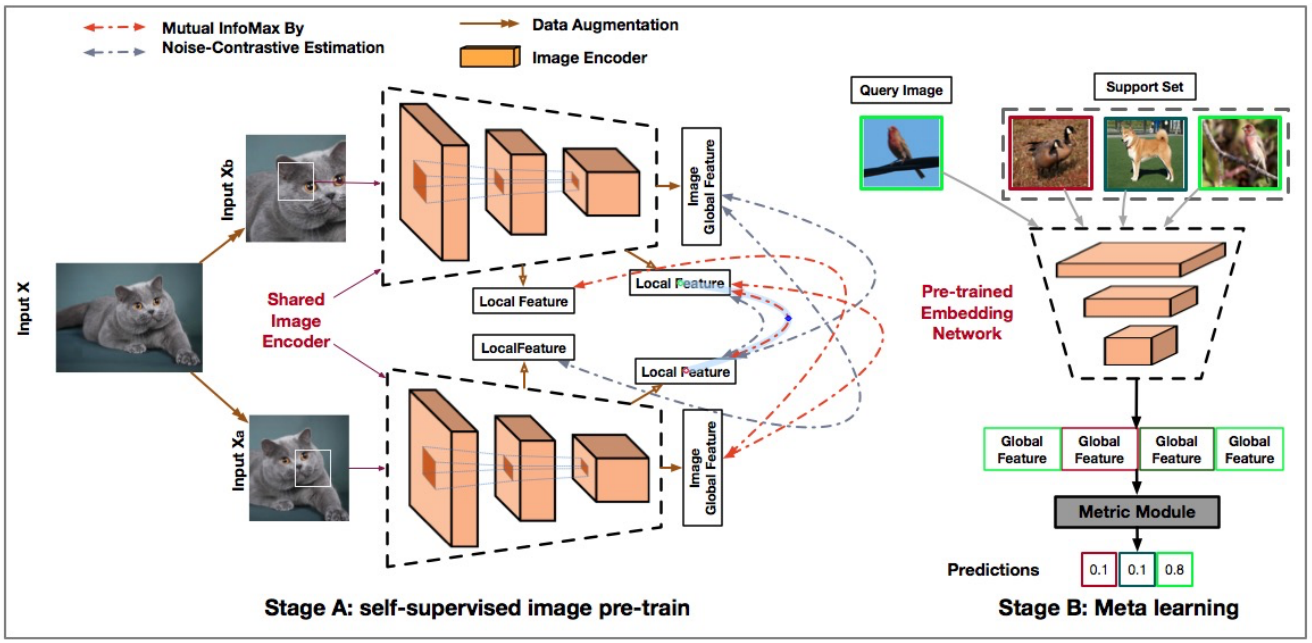

Figure 6.2 — AmdimNet [42]

We obtain the 79% accuracy after 20 epochs of training on the MiniImageNet dataset and fine-tuning and evaluating on the CUB-200 dataset (5-way 5-shot).

We obtain the 85% accuracy after 30 epochs of training on the MiniImageNet dataset and fine-tuning on the CIFAR-10 dataset.

We obtain the 90% accuracy after 30 epochs of training on the MiniImageNet dataset and fine-tuning on our proprietary dataset.

6.3.2. Prototypical Networks for Few-shot Learning

We have tested the performance an author PyTorch implementation of this classic metric learning paper by the authors as the baseline on the CUB in zero-shot setting and CIFAR-10 in the 5-shot 5-way formulation.

We have used a simple convolutional architecture described in the paper.

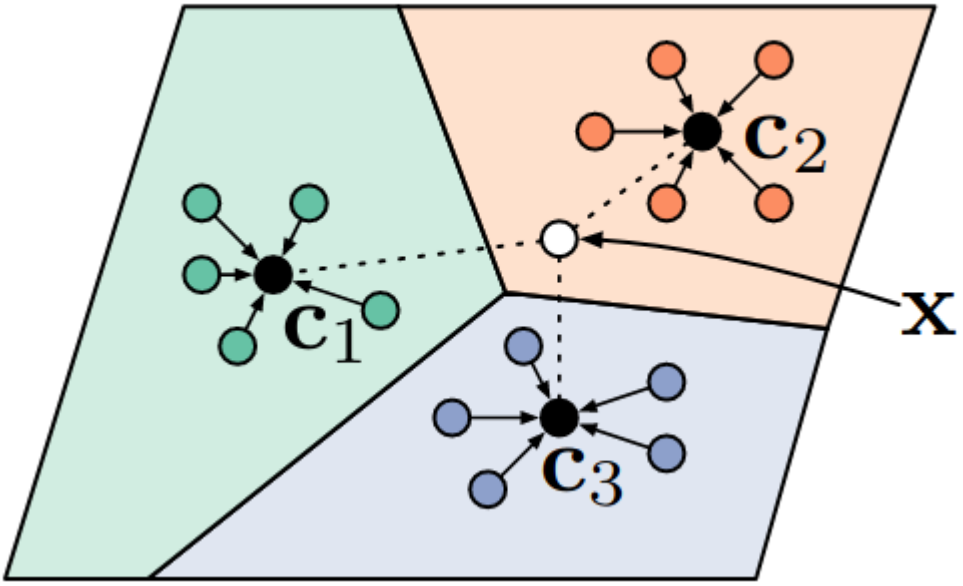

Figure 6.2 — Prototypical classification [18]

We obtain the 53% accuracy after 80 epochs of training on the unlabeled CUB-200 dataset.

We obtain the 70% accuracy after 60 epochs of training on the MiniImageNet dataset and fine-tuning on the CIFAR-10 dataset.

We obtain the 78% accuracy after 60 epochs of training on the MiniImageNet dataset and fine-tuning on our proprietary dataset.

6.3.3. Relation Networks for Few-shot Learning

We have tested the performance an author PyTorch implementation of this classic metric learning paper by the authors as the baseline on the CUB in zero-shot setting and CIFAR-10 in the 5-shot 5-way formulation.

We have used a simple convolutional architecture described in the paper.

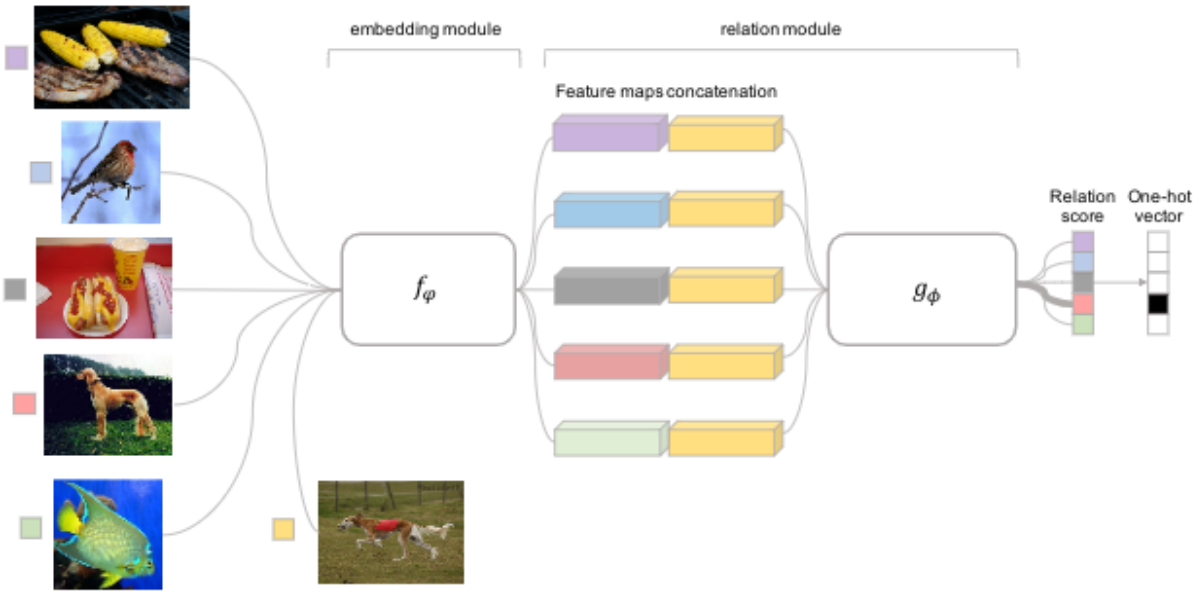

Figure 6.3 — Relation network [20]

We obtain the 50% accuracy after 40 epochs of training on the unlabeled CUB-200 dataset.

We obtain the 67% accuracy after 50 epochs of training on the MiniImageNet dataset and fine-tuning on the CIFAR-10 dataset.

We obtain the 75% accuracy after 50 epochs of training on the MiniImageNet dataset and fine-tuning on our proprietary dataset.

6.3.4. Momentum Contrast

We have tested the performance of a MoCo-based[35] model (in a PyTorch implementation[43]) CIFAR-10 in the 5-shot 5-way formulation.

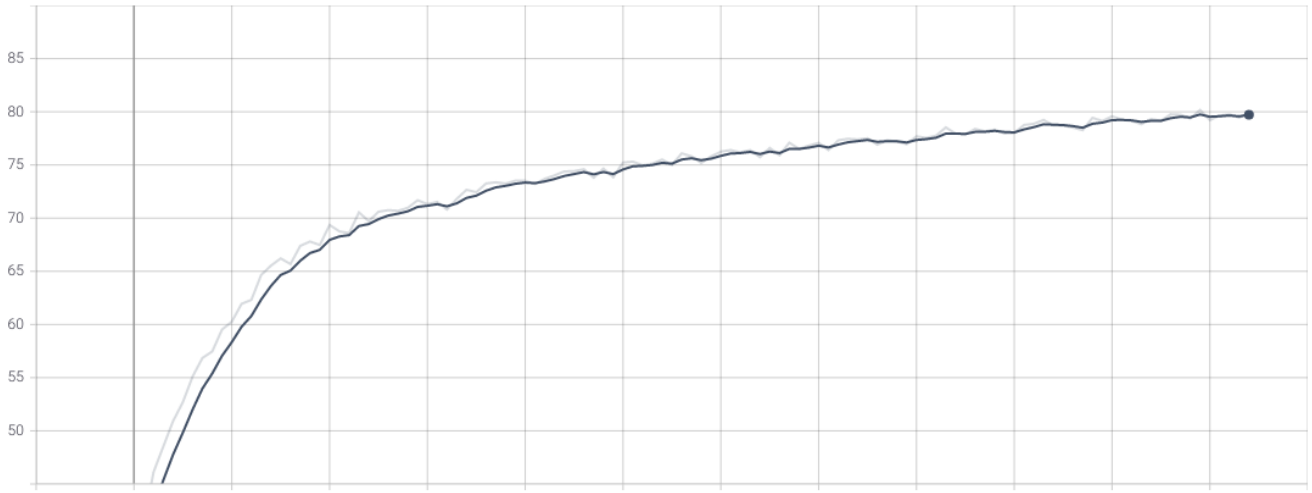

Figure 6.4 — MoCo accuracy plot

We obtain the 83% accuracy after 200 epochs of self-supervised training on the unlabeled CIFAR-10 dataset.

We obtain the 95% accuracy after 200 epochs of unsupervised pretraining on CIFAR-10 dataset and few-shot fine-tuning on our proprietary dataset.

We obtain the 76% accuracy after 200 epochs of unsupervised pretraining on CIFAR-10 dataset and unsupervised fine-tuning on our proprietary dataset.

6.3.5. Meta Pseudo Labels

We have tested the performance of a PyTorch-based implementation of the algorithm [44] in application to CIFAR-10 with a 4000 samples labeled seed.

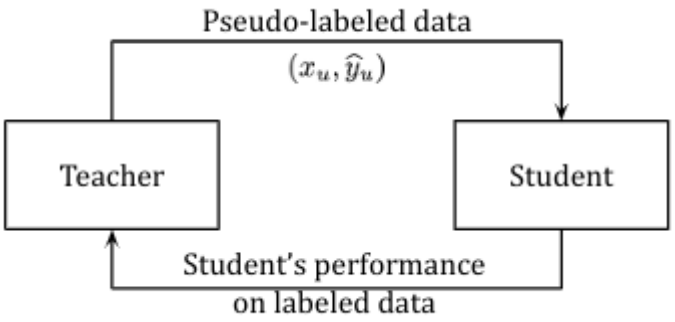

Figure 6.5 — Meta-Pseudo Labeling [44]

While the papers have claimed up results up to 94% accuracy results in this problem formulation, we have not managed to reproduce this result, with the model not making any progress after tens of epochs. We will further investigate these issues.

6.3.6. SimSiam

We have tested the performance of a PyTorch-based implementation of the SimSiam[38] paper with unsupervised CIFAR-10 classification.

We obtain the 80% accuracy after 200 epochs of self-supervised training on the unlabeled CIFAR-10 dataset (the accuracy kept increasing).

We obtain the 92% accuracy after 200 epochs of unsupervised pretraining on CIFAR-10 dataset and few-shot fine-tuning on our proprietary dataset.

We obtain the 74% accuracy after 200 epochs of unsupervised pretraining on CIFAR-10 dataset and unsupervised fine-tuning on our proprietary dataset.

CONCLUSION

So, in this work we have analyzed the existing body of research on few-shot learning and the various methods used to solve the few-shot image classification problem.

We have treated, both theoretically and experimentally, embedding learning methods, multitask methods, different kinds of metric losses. Tried various self- and semi-supervised learning methods like MoCo, SwAV, SimCLR, BYOL and SimSiam.

In addition, we have tackled the problem of performing online large-slace image recognition. It requires solving tasks like Out-Of-Distribution detection and clusterization, class discovery.

As a result of this work, we have proposed a mechanism for implementing a system to solve the problem of fine-grained large-scale image recognition using few-shot learning methods.

We have conducted thourough qualitative and quantitative analysis on the application of various few-shot learning methods for image classification implemented in PyTorch both in benchmark and real-world data.

This methods will further be improved on and tested on real-world product data.

We proposed a novel dataset of product shelf images.

Our next goal is implementing the whole system to work in a fully-online way with minimal human intervention.

Intermediate results were published in the “Shevchenko Spring 2021” conference.

REFERENCES

1. Zhao, B., Feng, J., Wu, X. et al. A survey on deep learning-based fine-grained object classification and semantic segmentation. Int. J. Autom. Comput. 14, 119–135 (2017). https://doi.org/10.1007/s11633-017-1053-3
2. Welinder P., Branson S., Mita T., Wah C., Schroff F., Belongie S., Perona, P. “Caltech-UCSD Birds 200”. California Institute of Technology. CNS-TR-2010-001. 2010.
3. Wang, Y., Yao, Q., Kwok, J., & Ni, L. (2020). Generalizing from a Few Examples. ACM Computing Surveys (CSUR), 53, 1 - 34.
4. Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., & Shah, R. (1993). Signature verification using a" siamese" time delay neural network. Advances in neural information processing systems, 6, 737-744.
5. Chechik, G.; Sharma, V.; Shalit, U.; Bengio, S. (2010). "Large Scale Online Learning of Image Similarity Through Ranking" (PDF). Journal of Machine Learning Research. 11: 1109–1135.
6. Schultz, M.; Joachims, T. (2004). "Learning a distance metric from relative comparisons" (PDF). Advances in Neural Information Processing Systems. 16: 41–48.
7. Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 815-823).
8. Wang, J., Zhou, F., Wen, S., Liu, X., & Lin, Y. (2017). Deep metric learning with angular loss. In Proceedings of the IEEE International Conference on Computer Vision (pp. 2593-2601).
9. Bendale, A., & Boult, T. (2015). Towards open world recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1893-1902).

10. He, J., Mao, R., Shao, Z., & Zhu, F. (2020). Incremental learning in online scenario. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 13926-13935).
11. Song, H. O., Xiang, Y., Jegelka, S., & Savarese, S. Supplementary Material for Deep Metric Learning via Lifted Structured Feature Embedding.
12. TODO
13. Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In Proceedings of the IEEE international conference on computer vision (pp. 2980-2988).
14. Goldman, E., Herzig, R., Eisenschtat, A., Goldberger, J., & Hassner, T. (2019). Precise detection in densely packed scenes. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 5227-5236).
15. Zhou, Z. H. (2018). A brief introduction to weakly supervised learning. National science review, 5(1), 44-53.
16. Caruana, R. (1997). Multitask learning. Machine learning, 28(1), 41-75.
17. Altae-Tran, H., Ramsundar, B., Pappu, A. S., & Pande, V. (2017). Low data drug discovery with one-shot learning. ACS central science, 3(4), 283-293.
18. Snell, J., Swersky, K., & Zemel, R. S. (2017). Prototypical networks for few-shot learning. arXiv preprint arXiv:1703.05175.
19. Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J. B., ... & Zemel, R. S. (2018). Meta-learning for semi-supervised few-shot classification. arXiv preprint arXiv:1803.00676.
20. Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H., & Hospedales, T. M. (2018). Learning to compare: Relation network for few-shot learning. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1199-1208).
21. Bertinetto, L., Henriques, J. F., Valmadre, J., Torr, P. H., & Vedaldi, A. (2016). Learning feed-forward one-shot learners. arXiv preprint arXiv:1606.05233.
22. Ye, M., & Guo, Y. (2018). Deep triplet ranking networks for one-shot recognition. arXiv preprint arXiv:1804.07275.

23. Xian, Y., Lampert, C. H., Schiele, B., & Akata, Z. (2018). Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. IEEE transactions on pattern analysis and machine intelligence, 41(9), 2251-2265.
24. Zheng, L., Duffner, S., Idrissi, K., Garcia, C., & Baskurt, A. (2016). Siamese multi-layer perceptrons for dimensionality reduction and face identification. Multimedia Tools and Applications, 75(9), 5055-5073.
25. Qian, Q., Shang, L., Sun, B., Hu, J., Li, H., & Jin, R. (2019). Softtriple loss: Deep metric learning without triplet sampling. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 6450-6458).
26. B. Yu and D. Tao, "Deep Metric Learning With Tuplet Margin Loss," 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 6489-6498, doi: 10.1109/ICCV.2019.00659.
27. Sun, Y., Cheng, C., Zhang, Y., Zhang, C., Zheng, L., Wang, Z., & Wei, Y. (2020). Circle loss: A unified perspective of pair similarity optimization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 6398-6407).
28. Fatih Cakir, Kun He, Xide Xia, Brian Kulis, Stan Sclaroff; Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 1861-1870
29. Movshovitz-Attias, Y., Toshev, A., Leung, T. K., Ioffe, S., & Singh, S. (2017). No fuss distance metric learning using proxies. In Proceedings of the IEEE International Conference on Computer Vision (pp. 360-368).
30. Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020, November). A simple framework for contrastive learning of visual representations. In International conference on machine learning (pp. 1597-1607). PMLR.
31. Hermans, A., Beyer, L., & Leibe, B. (2017). In defense of the triplet loss for person re-identification. arXiv preprint arXiv:1703.07737.

32. Xuan, H., Stylianou, A., & Pless, R. (2020). Improved embeddings with easy positive triplet mining. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (pp. 2474-2482).
33. Jaiswal, A., Babu, A. R., Zadeh, M. Z., Banerjee, D., & Makedon, F. (2021). A survey on contrastive self-supervised learning. Technologies, 9(1), 2.
34. Oord, A. V. D., Li, Y., & Vinyals, O. (2018). Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748.
35. He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 9729-9738).
36. Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., & Joulin, A. (2020). Unsupervised learning of visual features by contrasting cluster assignments. arXiv preprint arXiv:2006.09882.
37. Grill, J. B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., ... & Valko, M. (2020). Bootstrap your own latent: A new approach to self-supervised learning. arXiv preprint arXiv:2006.07733.
38. Chen, X., & He, K. (2020). Exploring Simple Siamese Representation Learning. arXiv preprint arXiv:2011.10566.
39. DeVries, T., & Taylor, G. W. (2018). Learning confidence for out-of-distribution detection in neural networks. arXiv preprint arXiv:1802.04865.
40. Nixon, J., Liu, J., & Berthelot, D. (2020). Semi-supervised class discovery. arXiv preprint arXiv:2002.03480.
41. Chen, X., Lan, X., Sun, F., & Zheng, N. (2020, August). A Boundary Based Out-of-Distribution Classifier for Generalized Zero-Shot Learning. In European Conference on Computer Vision (pp. 572-588). Springer, Cham.
42. Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images.
43. Chen, D., Chen, Y., Li, Y., Mao, F., He, Y., & Xue, H. (2019). Self-supervised learning for few-shot image classification. arXiv preprint arXiv:1911.06045.

44. Musgrave, K., Belongie, S., & Lim, S. N. (2020). Pytorch metric learning. arXiv preprint arXiv:2008.09164.
45. Pham, H., Dai, Z., Xie, Q., Luong, M. T., & Le, Q. V. (2020). Meta pseudo labels. arXiv preprint arXiv:2003.10580.