CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Šuma Pavel** Personal ID number: **435317**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Computer Science**

Study program: **Open Informatics**

Specialisation: **Artificial Intelligence**

II. Master’s thesis details

Master’s thesis title in English:

Representation learning for trademark search with limited supervision

Master’s thesis title in Czech:

Učení reprezentací s omezenou supervizí pro vyhledávání ochranných známek

Guidelines:

Trademark registration to patent offices is a time-consuming process, because possibly conflicting cases need to be identified. The goal of this project is to develop an approach based on Convolutional Neural Networks that identifies such cases from a large collection of trademarks. The task is characterized by a large degree of subjectivity, i.e. a conflicting case is not clearly defined. As a consequence only partial annotation exists, which is a limitation for constructing large scale training datasets. This project will exploit a small set of annotations and the structure of the whole trademark collection to automatically generate a richer training set. The student is expected to:
1. Study the basics of representation learning with CNNs
2. Implement a search systém for trademarks that uses CNN derived visual representation
3. Use existing datasets to create a new benchmarks that is suitable for semi-supervised learning
4. Use semi-supervised learning techniques to learn improved visual representations with CNNs

Bibliography / sources:

[1] Turusn Denman Sivipalan Sridharan Fookes Mau, Component-based Attention for Large-scale Trademark[1] Turusn Denman Sivipalan Sridharan Fookes Mau, Component-based Attention for Large-scale Trademark
Retrieval
[2] Tursun Aker Klakan, A Large-scale Dataset and Benchmark for Similar Trademark Retrieval
[3] Iscen Tolias Avrithis Chum CVPR2018 Mining on Manifolds: Metric Learning without Labels
[4] Radenovic Tolias Chum, PAMI 2019, Fine-tuning CNN Image Retrieval with No Human Annotation

Name and workplace of master’s thesis supervisor:

Georgios Tolias, Ph.D., Visual Recognition Group, FEE

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **29.07.2020** Deadline for master's thesis submission: **05.01.2021**

Assignment valid until: **19.02.2022**

Georgios Tolias, Ph.D.
Supervisor’s signature

Head of department’s signature

prof. Mgr. Petr Páta, Ph.D.
Dean’s signature

III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

________.____________________ ____________________________________

Date of assignment receipt Student’s signature

FACULTY
OF ELECTRICAL
ENGINEERING
CTU IN PRAGUE

Master’s thesis

Representation learning for trademark search with limited supervision

Bc. Pavel Šuma

Department of Computer Science
Supervisor: Georgios Tolias, Ph.D.

January 5, 2021

Acknowledgements

I would like to thank my supervisor, Georgios Tolias, Ph.D., for his guidance and valuable advice.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46 (6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity. However, all persons that makes use of the above license shall be obliged to grant a license at least in the same scope as defined above with respect to each and every work that is created (wholly or in part) based on the Work, by modifying the Work, by combining the Work with another work, by including the Work in a collection of works or by adapting the Work (including translation), and at the same time make available the source code of such work at least in a way and scope that are comparable to the way and scope in which the source code of the Work is made available.

In Prague on January 5, 2021 …………………

Czech Technical University in Prague
Faculty of Electrical Engineering

Citation of this thesis

Šuma, Pavel. *Representation learning for trademark search with limited supervision.* Master's thesis. Czech Technical University in Prague, Faculty of Electrical Engineering, 2021.

Abstract

This thesis aims to design methods for recognizing and retrieving similar trademark images from large databases. By embedding the input images into real-valued vector representations, the most similar trademarks can be derived with a simple Euclidean search. Acquiring these representations is done with the help of a deep convolutional neural network. Following a transfer learning paradigm, several network architectures pre-trained on related domains are studied and compared. On top of that, the best performing model is further fine-tuned in a metric learning fashion. A new annotated dataset consisting of more than two thousand trademark images was created for the purpose of the network training. Ground truth evidence for conflicting trademarks is scarce and manually annotating large amounts of data is slow and tedious. For this reason, the thesis also proposes various semi-supervised approaches suitable for deep metric learning to reduce the data requirements and better prepare the solution for possible real-world applications. Both supervised and semi-supervised methods are evaluated on standard trademark benchmark datasets.

Keywords deep metric learning, semi-supervised learning, trademark image retrieval

Abstrakt

Tato práce si klade za cíl navrhnout metody pro rozpoznávání vzájemně podobných obrázků ochranných známek. Zakódováním vstupních obrázků do vektorových reprezentací lze podobné ochranné známky najít pomocí jednoduchého vyhledávání v euklidovském prostoru. Pro získání těchto reprezentací se používá hluboká konvoluční neuronová síť. S využitím paradigmatu přenosu znalostí je v práci prozkoumáno a porovnáno několik síťových architektur, předem natrénovaných v souvisejících doménách. Nejvýkonnější z modelů je dále dotrénován pomocí způsobů učení metrik. Pro účely trénování sítí byla vytvořen nová anotovaná datová sada, skládající se z více než dvou tisíc obrázků ochranných známek. Prokázané případy konfliktních ochranných známek jsou vzácné a ruční anotace velkého množství dat je pomalá a zdlouhavá. Z tohoto důvodu práce také navrhuje několik metod pro učení s omezenou supervizí, vhodných pro hluboké učení metrik, které snižují požadavky na data a lépe připravují řešení pro možné aplikace v reálném životě. Všechny metody jsou vyhodnoceny na standardních datových sadách obrázků ochranných známek.

Klíčová slova hluboké učení metrik, učení s omezenou supervizí, vyhledávání obrázků ochranných známek

Contents

List of Figures

List of Tables

Introduction

Hand in hand with increasing globalization as well as digitalization of all content created today comes frequent copyright infringement, both intentional and accidental. There has already been considerable effort to tackle this issue (DMCA, EU Directive 2019/790). For instance, the YouTube content ID system is used to find unauthorized uses of original work by producing video and music fingerprints using deep learning. This system has generated as of 2018 over $3B for copyright holders [1]. There is a great demand for such automatic systems; the internet is sprawling, and manual checks are no longer sufficient.

A similar issue exists in the domain of company logos. Very often, new company tries to imitate the visual looks to boost their sales, while hurting the brand identity of the established company. Other times, businesses with no malicious intent create a similar logo by accident, leading to the same problems.

This thesis proposes different deep learning techniques to identify similar-looking trademarks, which can be beneficial in finding cases of copyright fraud. The current workflow is predominantly manual, by categorizing each new logo into similarly themed groups and finding collisions among them. Utilizing the strengths of deep learning, which shows exceptional usability for visual tasks, seems like a great fit for this problem. However, the domain of trademark logos poses significant challenges to overcome.

Learning logo representations suitable for automated comparison differs from standard real-life computer vision problems, such as classification or object recognition. A boat is always a boat, but ground truth in logo images is sparse and hard to assess fairly. One difficulty is measuring subjective similarity, as perceived by humans. While the pure visual similarity is computable, there are other kinds such as semantic or name similarity. Many logos might seem obviously similar to us because of some artistic intent, but describing this in an algorithm is non-trivial.

The potential amount of data available is huge by simply scraping the internet, as well as utilizing the government's logo databases (WIPO, USPTO). On the other hand, manually labeling a dataset of logos is a tedious task, as deciding which logo images should belong to the same group is uneasy. For these reasons, the thesis also explores ways of training neural networks in a semi-supervised learning setup for the task of image retrieval. This area is well studied in deep classification, but not in the case of deep metric learning. Moreover, the semi-supervised metric learning approach is also suitable for real-world applications as legal decisions of conflicting trademark law cases provide only limited information, i.e., two images are provably similar, but it tells nothing about other trademarks.

Problem description and objectives

This thesis's task can be presented as automatizing the currently manual process of conflicting trademark assessment. Consider a massive database of logo images and a new trademark application. In order to prevent infringement, the reviewer has to visually observe all relevant categories, whether there is an already existing trademark hardly distinguishable from the application. In my work, I implement an algorithm, which automatically ranks all of the existing logo images in the database by their visual similarity to any given image input.

Generally, the process of searching the database for images based on some attribute is called *image retrieval*, or specifically for this instance, *trademark retrieval*. Storing and searching images in their original formats (jpeg, png) is ineffective and supports only limited comparison tools. A partial solution to this problem is to convert all images into compact representations, typically real-numbered vectors. One can then employ a distance or similarity measure to compare these vectors. However, either the representations or the distance measure have to be tailored to capture the specific aspects that make two trademarks similar.

To produce high dimensional embedding vectors of the images, I train a deep neural network in a metric learning setting. Then, to retrieve the most similar embeddings for a given query, a nearest neighbor search based on cosine similarity is used. There is typically a large amount of human-annotated data needed to train a robust neural network. Hence, several methods are proposed to alleviate the data requirement.

Note that the approach used in this thesis is not tied to trademark images in particular. It can be utilized for any kind of images, but the trademark domain is sufficiently difficult and suitable for semi-supervised applications.

The objectives can be summarized into a couple of key points, which roughly correspond to the contents of each chapter:

1. Review the most relevant work done in the trademark retrieval domain and approaches in the field of deep semi-supervised learning.

2. Study appropriate literature and give a brief summary of the necessary background.

3. Construct a suitable train and validation set covering realistic trademark images and define evaluation measure and test environment.

4. Discuss essential parts of convolutional network training pipeline implementation.

5. Design and compare several semi-supervised methods applicable in deep metric learning.

Chapter 1

Related work

The related work of this thesis can be categorized into two sections. First section takes a look at past research in the domain of logo similarity. Second section briefly summarizes some of the most related recent semi-supervised approaches.

1.1 Logo similarity

Metric learning is a field with vast practical application possibilities. It has been extensively studied for the task of person re-identification [2, 3, 4], face recognition [5, 6, 7], 3D shape retrival and matching [8, 9, 10], or in health related fields [11, 12].

It has also been applied to the domain of trademark similarity. Tursun *et al.* [13] have created a large-scale dataset of trademark images and used it to compare popular methods of image retrieval. The authors utilized both hand-crafted features such as SIFT or SURF and deep neural networks. In the subsequent paper, Tursun *et al.* [14] recognized the text aspects of logo images to be highly distracting for the deep learning methods. By directing the attention of CNN to non-textual parts of the image, they achieved state of the art results in trademark image retrieval.

Lan *et al.* [15] utilized Local Binary Pattern operator on the feature maps outputted by five convolution layers of CNN to create improved feature vectors for logo images. In their more recent work [16], the authors have proposed a combination of a triplet or contrastive loss with Hinge loss to weaken the otherwise strict label constraints, which are often hard to obtain in the trademark domain.

Perez *et al.* [17] used a combination of two deep convolutional networks for the task of trademark image retrieval. One was trained to recognize visual similarity, while the other was built to learn conceptual similarity. The final retrieved images were ranked using the combination of the two individual rankings of computed cosine similarities with the use of feature vectors from both models.

Some research approaches combine multiple aspects of logo similarities to form one final assessment. Trappey *et al.* [18] besides spelling and pronunciation similarity also considered image similarity by using convolutional network trained on Cifar10 dataset. The authors employed a so-called multi-input concept, where a single image is processed through (1) the top 3 layers and (2) the top 7 layers of VGG16. These two feature vectors are concatenated before coming into the loss layer. The evaluation of the model was done on 100 real-life infringement cases, where the ground truth is the verdict of the court in the specific case.

Ko *et al.* [19] specifically focused on the phonetic similarity of trademarks by utilizing convolutional networks on the 2-grams of International Phonetic Alphabet transcripts of the original company names.

1.2 Semi-supervised learning

Recently, the research attention shifted from solely fixating on supervised deep learning with a large amount of labeled data to incorporating some form of self- or semi-supervised techniques. Chen *et al.* [20] showed that pre-training networks with the use of self-supervision before the fine-tuning part on labeled data can boost the model's performance. The gap between self-supervised and semi-supervised learning (SSL) is shrinking, and both perspectives can even be complementary [21].

Approaches based on *consistency regularization* are currently regarded as the state of the art of SSL. Temporal Ensembling [22] enforces consistency between the accumulated average of previous outputs and the current model output. Mean Teacher technique [23] instead uses exponentially accumulated average of network parameters to compute the loss between its output and the real output. Virtual Adversarial Training [24] directly estimates the smallest perturbations that affect the prediction of the network the most. The consistency regularization is then applied between the perturbated samples and their original counterparts.

Another main type of semi-supervision is *pseudo-labeling*, which assigns labels to the unlabeled part of the dataset with the help of the labeled samples. The pseudo-labeled data is then used in training in the same way as the labeled data. Lee [25] has first applied pseudo-labeling in conjunction with deep learning by computing classification probabilities for unlabeled data and regarding the ones with a higher value than a predefined threshold as new labeled training samples. Iscen *et al.* [26] utilized label propagation on the nearest neighbor graph of network output embeddings to obtain new pseudo-labels. The work of Haeusser *et al.* [27] defined transition probabilities as similarities between embeddings of inputs labeled set A and unlabeled set B.

With these probabilities, they observed round-trips as paths starting from A, passing through B, and ending back in A. The final loss is combined standard classification loss with an association loss, which is defined as cross-entropy between the distribution of correct round-trips, starting and ending in the representation of the same class and actual round-trips probabilities.

My implementation part utilizing SSL discussed in this thesis can be seen as a form of pseudo-labeling. However, the new labels in my case are inferred using proximity relations of learned representations in the metric space.

Chapter 2

Background

2.1 Metric learning

Representing data as elements in metric space is often desirable, as we can then easily cluster, classify, or rank our data by utilizing simple algorithms, such as k-nearest neighbors (k-NN). Alternatively, in tasks with a large amount of unlabeled data, we can employ methods using representations to find new labels based on mutual similarity automatically. Clustering the inputs based on resemblances is also necessary in cases where the amount of possible classes is too high, or the class boundaries are broad and standard classification is not applicable. Obtaining distance function fulfilling the preservation of similarity relation between inputs is the primary goal of a machine learning branch called metric learning. In this context, the term metric is interchangeable with distance and distance function.

More formally, let $X = \{x_1, \ldots, x_N\}$ be a set of N inputs from an input space $\mathcal{X}$ and additional information about the relations between inputs is available. In a weakly supervised setting, this information is given by tuples, typically pairs,

$$\begin{aligned} S^+ &= \{(x_i, x_j) \in X \times X : \ x_i \text{ is similar to } x_j\}, \\ S^- &= \{(x_i, x_j) \in X \times X : \ x_i \text{ is not similar to } x_j\} \end{aligned} \tag{2.1}$$

with the requirement $S^+ \cap S^- = \emptyset$. In the terms of standard supervised learning, the information is given as a list of labels $Y = (y_1, \ldots, y_N)$ of N classes, each containing usually multiple data points that are mutually similar:

$$\begin{aligned} S^+ &= \{(x_i, x_j) \in X \times X : y_i = y_j\}, \\ S^- &= \{(x_i, x_j) \in X \times X : y_i \neq y_j\} \end{aligned} \tag{2.2}$$

For the semi-supervised setting, the union of sets S^+ and S^- do not cover the complete set of tuples over X, and there is only a limited list of labels $|Y| < N$. In other words, there is no information about some inputs, which could potentially be similar to some of the labeled items.

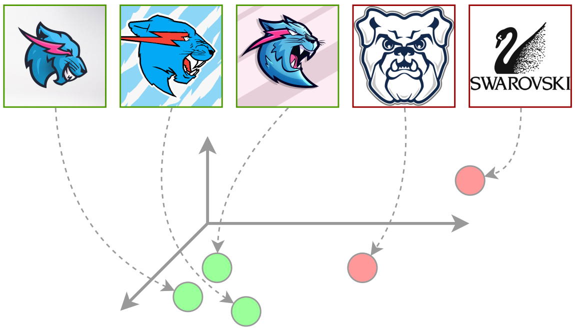


Figure 2.1: Metric learning aims to represent similar (green) images to have smaller mutual distances in the embedding space than dissimilar (red).

2.1.1 Mahalanobis distance

The similarity between input data points can be quantified by a distance function. As there is an infinite amount of such functions, it is important to choose the most optimal for our data, which enforces the concept of resemblance for inputs belonging to the same class. The optimal distance function $d : X \times X \to \mathbb{R}$ would fulfill the following:

$$y_i = y_j \neq y_k \Rightarrow d(x_i, x_j) < d(x_i, x_k) \quad \forall i, j, k \in N \tag{2.3}$$

Having the inputs initially represented as D-dimensional vectors $x_1 \dots x_N \in \mathbb{R}^D$, we can use the well-known Euclidean distance to measure the distance between each point. However, it would not account for the variance and correlations capturing the hidden intraclass relations. To mitigate this problem, we can apply a linear transformation L rescaling the data and replacing each point as $\hat{x} = Lx$. Such projection would optimally map the original vectors to new representations, which carry the additional similarity information. Afterward, we can use Euclidean distance in the standard way. Combining these two parts gives us distance mapping

$$d_L(x_i, x_j) = \|Lx_i - Lx_j\|_2 \tag{2.4}$$

Taking advantage of the Cholesky decomposition, the previous term can be also expressed as [28]:

$$\begin{aligned} \|Lx_i - Lx_j\|_2 &= (L(x_i - x_j))^T (L(x_i - x_j)) \\ &= \sqrt{(x_i - x_j)^T L^T L (x_i - x_j)} \\ &= \sqrt{(x_i - x_j)^T M (x_i - x_j)} \\ &= d_M(x_i, x_j) \end{aligned} \tag{2.5}$$

The matrix $M = L^T L$ is always positive semi-definite, thus distance in the form of d_M obeys all four properties required for a valid metric except distinguishability [29]:

$$\begin{array}{lr} \text{1. non-negativity} & d(x_i, x_j) \geq 0, \\ \text{2. symmetry} & d(x_i, x_j) = d(x_j, x_i), \\ \text{3. triangular inequality} & d(x_i, x_j) + d(x_j, x_k) \geq d(x_i, x_k), \\ \text{4. distinguishability} & d(x_i, x_j) = 0 \Leftrightarrow x_i = x_j \end{array} \tag{2.6}$$

By relaxing the fourth condition to $d(x_i, x_i) = 0$, a pseudo-metric is obtained. The family of pseudo-metrics in the form of d_M are called Mahalanobis distances. To simplify the language, I will refer to pseudo-metrics as metrics in the rest of the thesis.

Metric learning aims to find either linear transformation L or matrix M estimated from input data. Solving the problem with respect to M typically leads to a convex optimization problem, which is often not the case for solving with respect to L. On the other hand, parametrization with L is less restrictive and can be learned to project data to a lower-dimensional space [28].

2.1.2 Deep metric learning

Metric learning, from the standard view of Mahalanobis distances, has been studied for decades. It has been applied to various domains, where the inputs are represented by feature vectors. Since the recent breakthrough in the field of deep learning [30], Convolutional Neural Network (CNN) is the current state of the art for dozens of computer vision problems. One of the main advantages is the possibility of end-to-end training, which means no other algorithm is needed in the training pipeline. We do not have to manually extract feature vectors out of the input images, which typically requires some kind of human expertise on the specific topic. When talking about metric learning using CNNs, the methods are often termed as *deep metric learning.*

Even though convolution is a linear operation, it is followed by a non-linear activation function, making the whole pass through CNN a non-linear operation. Hence, utilizing a neural network to transform the input images into D dimensional space is not completely equivalent to the linear transformation L introduced in Section 2.1.1. Nevertheless, the core metric learning objective remains after replacing L with a non-linear transformation $f(\cdot, \theta) : X \to \mathbb{R}^d$, where θ are the network parameters.

Now the goal shifts to training a neural network, outputting the best representations of our input data. This translates to finding optimal network parameters θ and can be expressed as a non-convex optimization problem. The usual approach is to utilize a gradient-based iterative method, which minimizes some loss $l \in \mathbb{R}$ given by a loss function. Given the set of input images, their class labels, and a distance mapping between two feature vectors

$d_f(x_i, x_j) = \|f(x_i, \theta) - f(x_j, \theta)\|_2$, the objective can be formulated as:

$$\underset{\theta}{\operatorname{argmin}} \quad l(d_f, (x_1, y_1), \ldots, (x_N, y_N)) \tag{2.7}$$

The topic of applicable loss functions in the deep metric learning setting attracts a great deal of interest in recent years, allowing for many options. Two of the most widely used loss functions are described in the following sections.

2.1.2.1 Contrastive loss

Contrastive loss [31] is based on an idea of pulling similar inputs closer together and pushing dissimilar inputs farther in the metric space. An input argument to this loss function is a pair formed of a single input x^a, called *anchor* and either a similar sample x^+ or a dissimilar sample x^- in relation to the anchor. These two are referred to as *positive* and *negative* samples. Using the parametrized distance d_f, the contrastive loss for a positive pair and a negative pair is defined as

$$\begin{aligned} l_c(x^a, x^+) &= \frac{1}{2} d_f(x^a, x^+), \\ l_c(x^a, x^-) &= \frac{1}{2} \left[\alpha - d_f(x^a, x^-)\right]_+ \end{aligned} \tag{2.8}$$

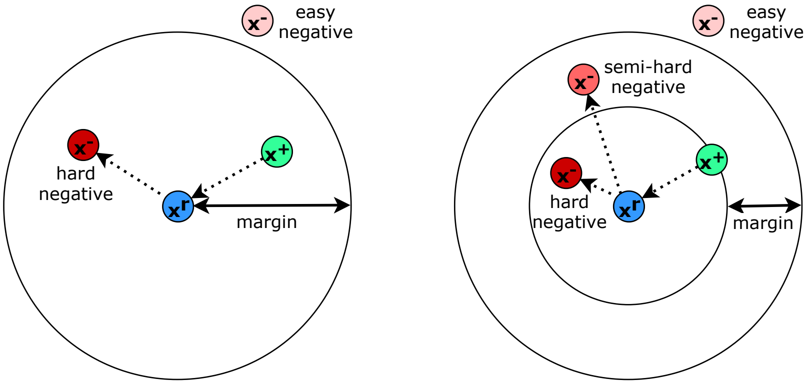
where $[\cdot]_+ = max(0, \cdot)$ and $\alpha \geq 0$ defines the minimal distance, beyond which all negatives should be located. Without this margin, the loss would punish dissimilar examples forever. The loss for the positive sample does not utilize such margin for the sake of stability. Otherwise, the points would reside too close to the margin and the function would be harder to optimize iteratively. The complete loss for all labeled tuples is computed as an average of individual losses.

2.1.2.2 Triplet loss

Even though *triplet loss* [32] is based on a similar idea as contrastive loss, it differs in a few key points. As before, we wish to pull similar points closer and push dissimilar ones farther, but this time we aim to do it only by forcing the positives to keep a predefined distance from the negatives. This specification should alleviate a restrictive property of contrastive loss, which pushes all similar inputs to a single point. Now the input to our loss is in the form of triplets instead of pairs. A set $\mathcal{T} = \{x^a, x^+, x^-)|y^a = y^+ \wedge y^a \neq y^-\}$ contains all such triplets of our input data X, where x^+, x^- refer to a single positive and single negative sample with respect to the *anchor* input x^a [33]. The loss for one such triplet has the following structure:

$$l_t(x^a, x^+, x^-) = \left[d_f(x_i^a, x_i^+) - d_f(x_i^a, x_i^-) + \alpha\right]_+ \tag{2.9}$$

Margin scalar α is again present, but in this case, it represents the minimal distance enforced between the positive and negative points. The difference is also visualized in Figure 2.2.



(a) Contrastive loss

(b) Triplet loss

Figure 2.2: Standard deep metric learning loss functions. Dashed arrows indicate the direction in which the samples will be moved according to each loss.

2.1.2.3 Negative sampling

The set containing all possible pairs for contrastive loss grows quadratically with the input size, and the set with all triplets can even grow cubically. In practice, with a high number of input samples, it is not feasible to optimize the loss function with all of the pairs or triplets. Moreover, most individual losses will equal zero and contribute nothing to the optimization since the negative samples are already too distant. Selecting the most relevant pairs/triplets is a major part of the optimization, and it may be as crucial as choosing the right distance loss [34].

There are commonly three types of negatives distinguished in the dataset, relative to a single anchor:

- *Hard negative* is initially mapped closer than some of the positives.
- *Semi-hard negative* is correctly mapped beyond positives, but the loss function is still non-zero.
- *Easy negative* is already past the margin distance and its loss is equal to zero.

The technique of selecting suitable negative examples is called *negative sampling*. It is also possible to sample optimal positives, however the need for supervision constrains the size of the positive pool, and it is often preferred to use all labeled data available. The idea of algorithmically looking for new suitable positive or negative samples is firmly connected to the semi-supervised methods called *positive/negative mining.*

Semi-hard negatives do not exist in the context of contrastive loss. The pairs are often selected randomly or with the emphasis on selecting hard examples, as it usually leads to faster convergence. For the triplet loss, this may not be the case as hard negative sampling can lead to a collapse of a model, which maps all inputs to a single embedding [34].

2.2 Fully convolutional models

Selecting the right architecture is an essential part of any deep learning task. Many of the most innovative networks were designed for classification tasks as they historically attracted the most attention. Fortunately, we can make use of the CNNs specifically designed for classification by adapting the model to output the desired embedding instead of class probabilities. We would typically omit the last softmax layer and all trailing fully connected (FC) layers, leaving only convolutional layers. This procedure was first described in the paper by Long et al. [35], where the denomination Fully Convolutional Network (FCN) was first coined. Since then, researchers have also designed new FCN architectures from scratch. The next two sections describe two of the most used architectures in computer vision tasks.

2.2.1 VGG16

VGG proposed a revolutionary architecture at the time of its introduction in 2015 [36]. There are several types of VGG, varying in the number of weight layers. This section will only describe the version I use in my implementation, a 16-layered VGG16.

Each convolutional layer preserves the input size by using 3×3 or 1×1 kernels with padding and stride equal to 1. Downsampling is only achieved by five max-pooling layers with 2×2 kernels and stride 2. By stacking multiple convolutions with smaller receptive fields instead of traditionally using larger ones, such as 7×7, the authors achieved the same effective receptive field while decreasing the number of parameters. Moreover, using multiple activation functions makes the decision function more discriminative [36]. The network is originally trained on square RGB images of size 224×224. However, the minimal size of an input can be lowered to $2^5 = 32$ to satisfy the five downsamplings by the factor of 2.

2.2.2 ResNet

The plain architecture of Residual Neural Network (ResNet) [37] is mainly inspired by VGG nets. Once again, there are multiple variations with up to

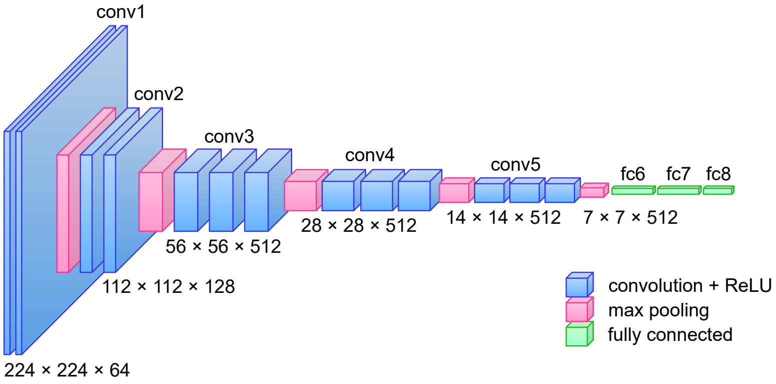


Figure 2.3: VGG16 architecture. To obtain a 1-dimensional embedding, the last FC layers (green) are usually replaced by global pooling. Below each block of layers are specified their output dimensions.

152 layers.

One of the key differences is that the authors opted for downsampling only with convolutions besides the single max-pooling layer in the beginning. The other difference is the implementation of *shortcut connections*, where the input to the convolutional block gets also redirected and added to the output. This should help the non-linear network to approximate identity mappings. Finally, the deeper version of ResNet also use *bottleneck* blocks, which is a stack of 3 convolutions $(1 \times 1, 3 \times 3, 1 \times 1)$. The 1×1 convolution can be seen as a simple scalar multiplication in each channel. In this case, it reduces the number of channels and then restores them once more after the costly 3×3 kernel convolution. The reason for this is mainly economic, as it enables to create very deep networks without losing too much efficiency.

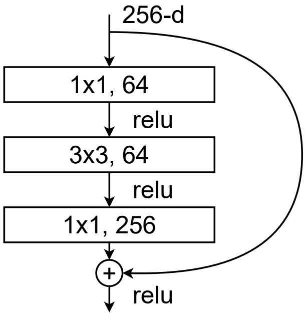


Figure 2.4: Bottleneck block, consisting of three convolutions and ReLU functions, used in deeper ResNet versions. The shortcut connection, on the right, is added to the output as is. Image taken from the original ResNet paper [37].

It may appear that ResNet50, with many more layers, is larger and computationally more complex than VGG16. In reality, VGG16 has roughly 138 million parameters, while ResNet50 has only 25.5 million. These numbers reflect only the network's physical size, as more parameters do not automatically mean longer training time. Nonetheless, ResNet50 also wins in terms of speed, with 3.8 billion FLOPs (Floating point operations) in contrast with 15.3 billion FLOPs for VGG16 [37]. Size disparities for the first two layers (in the case of VGG16 operating on the full image size) play a significant role in the overall time complexity.

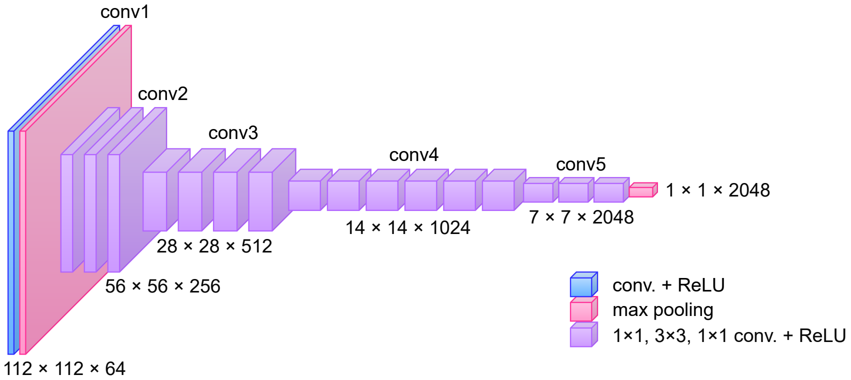


Figure 2.5: ResNet50 architecture. All layers except the first two are the bottleneck blocks, consisting of three convolutional layers. Below each block of layers are specified their output dimensions.

2.2.3 Global pooling

On input, the network expects an image with three dimensions $W_1 \times H_1 \times C_1$, representing width, height, and the number of channels. The number of input channels is equal to 3 for an image in standard RGB color space. When the input progresses through the network, the width and height typically shrink, and the number of channels grows. It can be seen in a simplified way as the network gradually "zooming out" to see less detail and looking through more and more different filters. The last n-th convolutional layer will also yield output in three dimensions $W_n \times H_n \times C_n$. While varying in the specific width and height, depending on the input size and architecture, the amount of output feature channels C_n is always the same for one particular architecture. The goal is to obtain a one-dimensional vector for each image to measure similarities using the predefined metric.

Naturally, we can resize all of our images to the exact size for which the last convolutional layer output dimensions would be equal to $1 \times 1 \times C_n$. This is not practical and would cause us to lose detail or the correct image aspect ratio. Another option would be to flatten the output, i.e., rearrange the 3-dimensional array into a 1-dimensional sequence of $W_n * H_n * C_n$ values.

Lastly, the most prevalent method nowadays is called Global Pooling (GP). It works virtually the same as the standard max-pooling layer with a kernel size set to the whole input. In other words, global pooling projects the 2-dimensional $W_n \times H_n$ feature (activation) map of values onto a single real value corresponding to either maximum, sum, or any other operator.

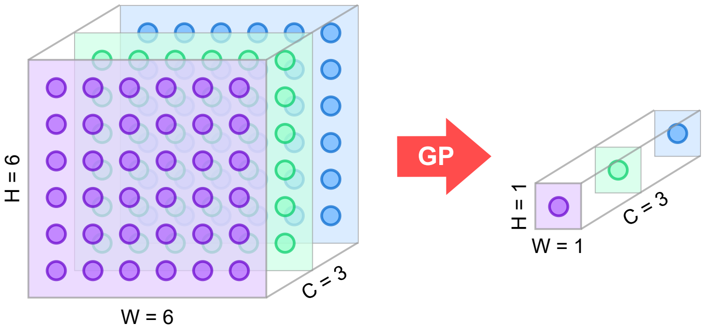


Figure 2.6: Visual representation of global pooling operation, regardless of the specific implementation. Image inspired by Cook [38].

Let $\mathcal{X}_c$ be the set of all activations in the c-th channel, the final feature vector f has the following form:

$$\mathrm{f} = [\mathrm{f}_1 \dots \mathrm{f}_c \dots \mathrm{f}_C] \,, \tag{2.10}$$

where f is the global pooling operator [39]. In this thesis I employ and compare these four types:

- *Maximum Activations of Convolutions* (MAC) [40], also max-pooling, is defined as:
$$\mathrm{f}_c^{(m)} = \max_{x \in \mathcal{X}_c} x \tag{2.11}$$

- *Sum-Pooling of Convolutions* (SPoC) [41] is an aggregation of the outputted features:
$$\mathrm{f}_c^{(s)} = \left(\frac{1}{|\mathcal{X}_c|} \sum_{x \in \mathcal{X}_c} \alpha(x)\, x \right) , \tag{2.12}$$
where α defines a centering prior function, which assigns larger weights to the center of the feature map. If $\alpha = 1$, then the equation equals to the plain sum of all values.

- *Generalized mean* (GeM) [39] is a generalization between the two previous cases:
$$\mathrm{f}_c^{(g)} = \left(\frac{1}{|\mathcal{X}_c|} \sum_{x \in \mathcal{X}_c} x^p \right)^{\frac{1}{p}} \tag{2.13}$$

The result is equal to SPoC when $p = 1$ and equal to MAC when $p \to \infty$. Since the equation is differentiable, the optimal parameter p can be learned during the training.

- *Regional MAC* (R-MAC) [40] extends the idea of MAC by utilizing max-pooling over a set of square regions $\mathcal{R}_c \subseteq \mathcal{X}_c$ at different scales of each feature map.

  $$\mathrm{f}_{\mathcal{R}} = [\mathrm{f}_{\mathcal{R},1} \dots \mathrm{f}_{\mathcal{R},c} \dots \mathrm{f}_{\mathcal{R},C}], \quad \mathrm{f}_{\mathcal{R},c} = \max_{x \in \mathcal{R}} x \tag{2.14}$$

  The individiual region descriptors are then normalized, whitened (Section 2.3), normalized again and summed to form the final descriptor.

It is important to note that global pooling operator is translation invariant. Since only a single value is outputted from each channel, the important region in the image can be shifted and the resulting descriptor will be identical.

2.3 Whitening

The final image descriptors obtained from FCN with global pooling may have a relatively high dimension (2048 for ResNet50). Some of these dimensions will carry redundant information due to mutual correlations. A common postprocessing method that tackles these problems is Principal Component Analysis (PCA). When used after global pooling, sum-pooling (SPoC) tends to perform better when the resulting embedding vectors are PCA-whitened (PCA$_W$), and max-pooling (MAC) performs better when they are not [41].

2.3.1 PCA whitening

Let $X = [x_1, \dots, x_N] \in \mathbb{R}^{D \times N}$ be a matrix containing embedding vectors for all inputs in columns. We can compute covariance matrix of X as

$$\Sigma = \frac{(X - \mu)^T (X - \mu)}{N - 1}, \tag{2.15}$$

where $\mu \in \mathbb{R}^D$ denotes the corresponding vector of means of the rows of X. Using eigendecomposition $\Sigma = P \Lambda P^{-1}$, we obtain eigenvalues as diagonal elements of $\Lambda = \mathrm{diag}(\lambda_1, \dots, \lambda_D)$ and eigenvectors $P \in \mathbb{R}^{D \times D}$. The final projection of X is then given by rotating descriptors by P:

$$\hat{X} = P^T X \tag{2.16}$$

When using only a subset of rows of P, the transformation is onto lower-dimensional space. Typically, when we wish to reduce the number of dimensions from D to K, we keep only those eigenvectors corresponding to the top-K

eigenvalues. This ensures keeping most of the variance in our data while having a more compact representation. After de-correlating and shortening the final embeddings, the problem of over-counting some of the visual patterns remain, which could significantly affect the resulting similarity comparison.

Whitening the data should alleviate this problem by down-weighing co-occurring features in the descriptors [42]. It is attained by scaling the vectors along each dimension to have unit variance, which is also equivalent to dividing each eigenvector by the square root of its corresponding eigenvalue:

$$\hat{X} = \Lambda^{-1/2} P^T X \tag{2.17}$$

As the matrix Σ is merely an empirical estimation of covariance of our input distribution, the eigenvalues may be too extreme or too small, making the whitening operation unstable. Mukundan *et al.* [43] propose to reformulate the eigenvalues as a linear combination of the original value and new parameter β, which corresponds to i-th eigenvalue. Formally, using

$$\Lambda = \mathrm{diag}((1-\beta)\lambda_i + \beta, \ldots, (1-\beta)\lambda_d + \beta) \tag{2.18}$$

in the Equation 2.17 is called PCA whitening with shrinkage.

PCA whitening projection is a linear operation and can also be modeled by a single fully connected layer [39]. This way, the postprocessing can be learned with the network in an end-to-end manner.

Chapter 3

Benchmark

For all experiments in this thesis, I use three datasets containing trademark images. Two of them are already presented in research papers and made available by the authors, and one of them is created by me. This chapter defines measurements needed to evaluate the trained models on test sets and describes the structure of all three datasets along with the process of collecting train and validation data.

3.1 Evaluation measurements

Following the real life analogy from the introduction of this thesis, consider we already have an algorithm which returns an ordered sequence of most similar trademark images to a particular image. Then, to evaluate the uniqueness of the given *query* image, the topmost ranked images would have to be visually examined.

To estimate the performance of a model without human presence, evaluation measurements are imitating the exact process automatically by using a pre-made ground truth with labeled set of query images Q. Each query is also assigned to a similarity group (class) as in the definition of a supervised setting (Equation 2.2). Combining Q with unlabeled images forms the whole test set. These are called *distractor* images and are present in the test set to make the problem harder and testing more robust.

An evaluation algorithm can now retrieve top k most similar images to a query utilizing the model and score it based on the number of correct outputs from the same class. This technique is known as the *Precision@k* (P@k) and is defined for a given query q as:

$$\text{P@k} = \frac{\sum_{i=1}^{k} [y_i = y_q]}{k}, \tag{3.1}$$

where y_i is the label of the image ranked at i-th position and $[\cdot]$ is the Iverson bracket notation.

Recall@k (R@k) represents the ratio of positives in the retrieved top k items to the total amount of existing images coming from the same class S_q as the given query.

$$\mathrm{R@k} = \frac{\sum_{i=1}^{k} [y_i = y_q]}{|S_q|} \tag{3.2}$$

Ranking positive items as close as possible to the query is a significant attribute of the algorithm. Unfortunately, neither precision nor recall accounts for the order of the retrieved positive items. For this reason, *Average precision* is a frequently used evaluation measure, which exploits the well-known trade-off between precision and recall and considers both of these measures simultaneously [44]:

$$\mathrm{AP} = \sum_{k=1}^{N} P@k(R@k - R@(k-1)) \tag{3.3}$$

By computing mean of APs for each query from Q, we obtain mean Average Precision (mAP) measure.

3.2 Test datasets

Two independent sources of test datasets are used to evaluate the obtained models and methods objectively. Having multiple test sets should prove useful, as different authors create them with slightly different views on the meaning of logo similarity. Additionally, it increases the fairness of comparison and mitigates subconsciously introduced bias when deciding which trademarks should be considered similar and fall in the same class. Note that this similarity relation constructed in the datasets can only be zero or one, in the sense that the images are either in the same class or not. Naturally, some of these similarities are clearer than others, but this information is lost by using discrete labels. Having multiple test sets should also help better assess the models' robustness since the number of queries in each dataset is not too large and conveys limited variance.

The first test set is *METU Trademark Dataset* [13]. It consists of 417 query images and almost 1 million distractors, making it the largest dataset for trademark retrieval. Even though it is beneficial to have the option to test on a dataset of such size, the evaluation phase takes a lot of time. Therefore, in some cases, I also use a random small subset of the original distractors and all the queries. I will refer to this limited test set as *tMETU* (tiny METU).

The second test set is *NPU-TM* [16]. In contrast to METU, this dataset's images are already rescaled to a similar resolution and padded with white background. When visually observing the trademark classes in NPU-TM, they are often less strict. In some cases, the same class annotated images are similar only very distantly, which would make them harder to correctly separate.

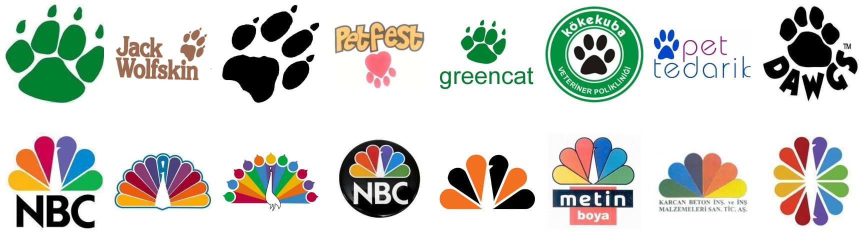


Figure 3.1: Two distinct classes of METU queries, each displayed in one row.

3.3 Train and validation sets

One of the options for a train set was to use either METU or NPU-TM dataset described above. Unfortunately, METU has too few classes to use for the training of the supervised baseline. NPU-TM dataset, on the other hand, consists of quite a lot of classes, but with only a few unique logo images in each of them on average. This fact would complicate the semi-supervised setting, presented later, in Section 5.1. For this reason, I collected new trademark dataset.

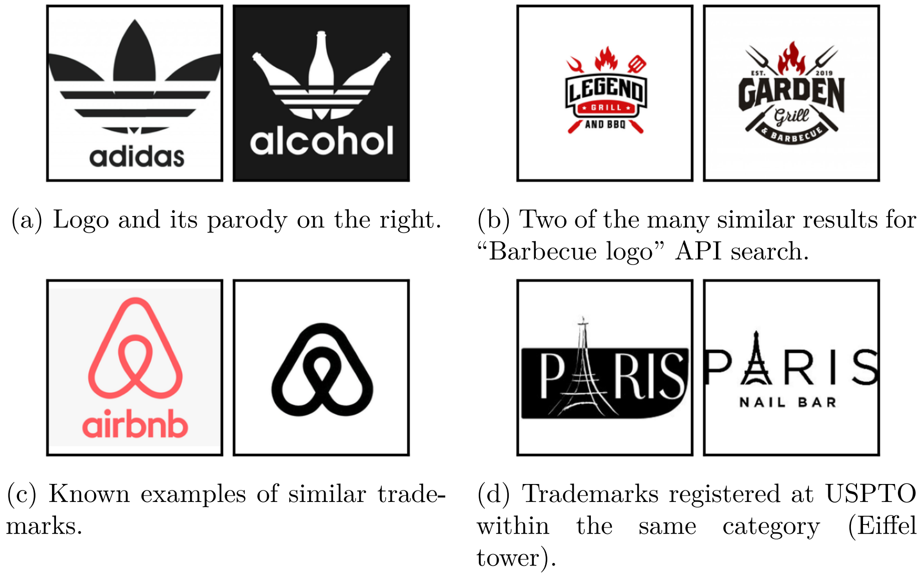
3.3.1 Data collection

To create a diverse training set, I selected four sources of images. Again, the reason for this is to have a lower bias in my data and higher variance. In order to effortlessly download large amount of data, I created a script that utilizes Google Search API and downloads the first 20 images for any given query. Google already ranks the retrieved images based on some similarity measure, which helps in finding relevant results even for broad queries, but it already instigates some prior bias. Still, the collected data contained many unwanted, identical, or low-quality images and had to be manually sorted. The complete collection procedure can be summarized as:

(a) Collected logo parodies of famous brands and their originals. An example of such query to the search API script would be “Adidas logo parody”.

(b) Found a dictionary file of day-to-day English nouns and used it in conjunction with the script to automatically collect the most related logo images to each of the nouns, as perceived by Google engine. In this case, the search query was for example “Barbecue logo”.

(c) Manually searched for known examples of similar trademarks and combined few images of both that look the most alike. METU is built in the similar manner, so some of them had to be skipped.



(a) Logo and its parody on the right.

(b) Two of the many similar results for “Barbecue logo” API search.

(c) Known examples of similar trademarks.

(d) Trademarks registered at USPTO within the same category (Eiffel tower).

Figure 3.2: Examples of the collected similar trademarks for each of the four types.

(d) *United States Patent and Trademark Office* (USPTO) offers Trademark Electronic Search System [45], which allows listing through existing registered trademarks. The trademarks are also grouped by categories, e.g. Animals, with subcategory Lions. The most visually similar images were selected from multiple suitable categories.

The main time-consuming part was to process the downloaded images manually and select only those that should truly fall in the same class. On the flip side, choosing too easy training examples would impair the model’s ability to learn. The images collected from USPTO are visibly the hardest for the computer to mark as similar and the logo parodies and their original counterparts are usually the most indistinguishable. Another challenging aspect is to skip trademarks that overlap with any of the two test sets, even if only distantly, to avoid leaking information from training to the testing phase. After manually selecting the best images for the training set, the rest of the downloaded images are kept aside and utilized in the semi-supervised setting described later.

3.3.2 Validation split

During the training of a network with multiple hyperparameters, it is vital to have a validation set to determine the optimal stopping criterion. Tuning the model (such as network architecture, learning rate, or weight decay) should be done with the aid of this validation set, and the test set should be used

Table 3.1: Detailed comparison of all used datasets. Labeled images are used as queries during the testing phase and as training images in the training phase. Likewise, unlabeled images serve as distractors, or a pool for positive/negative mining in the semi-supervised setting.

	METU	tMETU	NPU-TM	Train	Validation
Labeled/Query	417	417	2025	2160	267
Number of classes	35	35	643	312	46
Avg. logos per class	11.9	11.9	3.1	6.9	5.8
Unlabeled/Distractors	922927	9480	5113	9401	792
Total count	923344	9897	7138	11561	1059

only for the final evaluation. Since obtaining new labeled images is a tedious work and the training set is already small, only around 10 % of classes were selected randomly and put aside as a validation set. As in the case of test sets, all of the distractors in the validation set have to be strictly negative to the query images. For this reason, I manually selected only a small amount of unique distractors.

Chapter 4

Supervised baseline

The presented approaches for trademark image retrieval can be separated into two main parts. In the first one, the focus is on transferring knowledge from existing models and neural network training with labeled data. The second part offers ways of taking advantage of the excess unlabeled data in deep metric learning setting. To show the contributions of proposed methods and validate their reliability, a solid baseline has to be set. This chapter is devoted to the process of setting up such baseline, including the reasoning behind the selection of a particular architecture, neural network training parameters, and postprocessing techniques.

4.1 Transfer learning

Neural network architecture can be adapted or even tailored specifically for a given task, but it requires a significant effort and usually some expertise. A common practice is to choose established networks, which are regularly used and proven to be a good fit for the particular domain. Moreover, training neural networks from scratch is expensive and requires a fair amount of training data. There is a good chance a renowned network architecture was trained on a relevant large dataset, and its weights are made publicly available. I refer to these models as *pre-trained* networks. Usually, we would want to use a network that was trained for the most related application to our problem, yet robust enough to be still able to generalize. The technique of transferring knowledge from one domain to another in the form of reusing pre-trained networks is generally called *transfer learning.* Furthermore, the process of additional training of the pre-trained model on more specific dataset is called *fine-tuning.*

To select the initial model, I examine networks pre-trained on images from the three following datasets:

- The ImageNet [46] is probably the most well-known image database in the field of computer vision. It contains over 14 million images and around 1000 classes. Most of the big deep learning frameworks offer frequently used architectures pre-trained on this dataset.

- SfM120k [47] is a dataset of more than 120 thousand images obtained from the 3D model reconstructions of popular landmarks created by Structure-from-Motion pipeline. The authors also published multiple trained networks, along with code repository [48]. The models were trained for the task of image retrieval, which is almost identical to my setting. In addition, the repository offers trained weights for final whitening layer parameters for several global poolings.

- EdgeMAC [8] is a fully convolutional version of VGG16 trained on edge images extracted from the SfM120k dataset. The original task was also image retrieval with contrastive loss, which closely relates to my setting. To match the original inputs of EdgeMAC as closely as possible, I use the same edge detection method by Dollár and Zitnick [49]. Moreover, the outputted edge images are smoothed by the same edge filtering technique.

I evaluate two types of network architectures – VGG16 and four variations of ResNet, with 18, 50, 101, and 152 layers, pre-trained on all of the mentioned datasets. There are also four possible global pooling techniques defined in Section 2.2.3 and optional whitening layer. Training and tuning all of these combinations would take too much time. To narrow it down, the networks are first compared on both test sets prior to any training. The most important results are listed in Table 4.2.

In all cases, SfM120k pre-trained networks achieve higher mAP score than their ImageNet counterparts. VGG16 does not come close to ResNet, and variations with a higher amount of layers are, not surprisingly, dominating the smaller models. From the global pooling options, SPoC performs substantially worse, followed closely by R-MAC. GeM and MAC are more comparable, although the former is slightly better. Models with whitening in the form of pre-trained, fully connected layer appended after global pooling are also significantly better.

For EdgeMAC, there is only one available pre-trained architecture and pooling. On NPU-TM test set, the network outperforms the rest, while on tMETU, it outperforms only the other VGG16 networks. Nonetheless, the EdgeMAC model proves to be valuable even in the domain of trademark images. It is possible that some of the other networks are complementary and their combination with EdgeMAC will yield further improvements.

Table 4.1: mAP score on both test sets for multiple pre-trained variants. The checkmark in whitening column corresponds to the presence of fully connected layer appended after the global pooling.

Training set	Arch	Pooling	Whiten	tMETU	NPU-TM
SfM120k	ResNet50	GeM	✓	40.2	32.1
SfM120k	ResNet101	GeM	✓	41.8	33.3
SfM120k	ResNet152	GeM	✓	41.8	32.7
SfM120k	VGG16	GeM	✓	31.3	31.2
SfM120k	ResNet101	GeM	✗	32.2	27.8
SfM120k	VGG16	GeM	✗	30.4	30.2
ImageNet	ResNet101	MAC	✓	38.9	29.8
ImageNet	ResNet101	SPoC	✓	31.9	21.1
ImageNet	ResNet101	GeM	✓	38.9	29.3
ImageNet	ResNet101	R-MAC	✓	38.0	27.4
ImageNet	ResNet101	GeM	✗	37.7	27.9
ImageNet	ResNet101	R-MAC	✗	37.7	26.6
ImageNet	VGG16	MAC	✓	32.2	25.3
ImageNet	VGG16	SPoC	✓	22.7	16.5
ImageNet	VGG16	GeM	✓	30.4	24.5
ImageNet	VGG16	R-MAC	✓	31.3	24.0
EdgeMAC	VGG16	MAC	✗	37.7	35.4

4.1.1 Shape matching

Comparison of outline shapes is a significant part of trademark similarity assessment. Imagine two logos that share the same shape and differ only in color or texture. In some cases, the similarity may be distant and not necessarily confusing, e.g., car companies having slightly different cars in their logo. Still, we would want to see them in the top portion of retrieved images. On top of that, research has shown that ImageNet pre-trained networks are biased towards textures [50].

A possible option is to use edge maps as inputs to any pre-trained network in a standard way. However, the model has been trained on colored images with different textures and shades. The learned activations for edge image would not cover the essential distinguishing features of a shape.

Instead, for each input, I obtain a 2048-dimensional descriptor by feeding it into network pre-trained on RGB images and another 512-dimensional descriptor by feeding the input's edge map into EdgeMAC. I concatenate these two vectors, which results in a 2560-dimensional final descriptor. The most similar trademarks are found as before, by taking the inputs with the highest mutual cosine similarity of these enhanced descriptors. Combinations with EdgeMAC provided an additional boost in performance compared to using RGB networks independently, as reported in Table 4.2.

Table 4.2: Test evaluation for concatenated descriptors of EdgeMAC and other RGB networks.

Model		tMETU	NPU-TM
EdgeMAC		37.7	35.4
EdgeMAC	+ SfM120k ResNet101 GeM-W	46.2	37.0
EdgeMAC	+ ImageNet ResNet101 GeM-W	44.7	35.1
EdgeMAC	+ ImageNet VGG16 GeM-W	38.8	31.2

4.1.2 Multiscaling

As humans, we would mark any trademark logo with its rotated self as identical. The same applies to very similar images. Unfortunately, FCN is not rotation invariant, and the outputted descriptors would differ almost entirely. A standard trick to diminish such weakness is to feed the original image and its altered (rotated) versions to the network, producing multiple vectors of the same length. These descriptors can be aggregated using a sum function or element-wise maximum. The alteration can be arbitrary, as long as it preserves the original similarity relations. I have tested rotations, horizontal and vertical flips, and image scaling. Only the latter consistently improved mAP score on both test sets.

The chosen multiscaling technique corresponds to resizing an image three times to facilitate its larger side matches sizes 256, 181, and 362, while preserving the original aspect ratio. These three images are processed through the network and the obtained descriptors are summed to form one final descriptor.

Figure 4.1 visualizes the complete process of acquiring a vector representation for a single input with optional use of EdgeMAC and multiscaling.

4.2 Fine-tuning

Until now, only already trained networks on different types of images were considered. To better adapt the available models on the trademark image domain, I have introduced two improvements to accentuate shape importance and scale invariance. Additional training on the dedicated trademark dataset should enhance the process even further. Out of the tested pre-trained networks, I decided to fine-tune the standard FCN operating with input images in RGB mode, which corresponds to the oblong box on the left part of Figure 4.1. Since ResNet with GeM pooling trained on SfM120k images showed the best

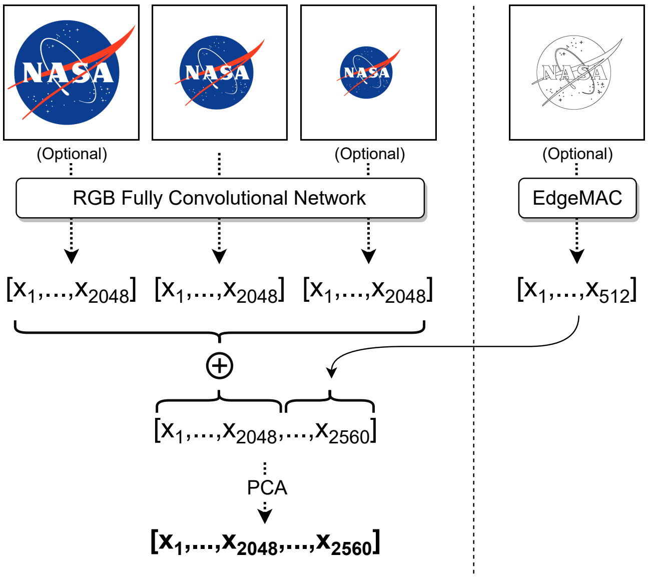

(Optional)
(Optional)
(Optional)
RGB Fully Convolutional Network
EdgeMAC
$[x_1,\ldots,x_{2048}]$ $[x_1,\ldots,x_{2048}]$ $[x_1,\ldots,x_{2048}]$
$[x_1,\ldots,x_{512}]$
$\oplus$
$[x_1,\ldots,x_{2048},\ldots,x_{2560}]$
PCA
$[\mathbf{x_1,\ldots,x_{2048},\ldots,x_{2560}}]$

Figure 4.1: The whole process depicting the embedding of a single input image into real-valued vector (descriptor). Utilization of multiple scales, or the edge image is optional. Whitening and PCA whitening is put in for completeness and is discussed after the network fine-tuning, in Section 4.2.4.

results and the performance gap between versions with 101 and 152 layers was marginal, I settle on the smaller version to save on the time requirements during training.

Deep neural networks are complex models and often viewed as a black box because even a slight change in one of its many parameters may largely affect the results in an unforeseeable way. This phenomenon especially applies to model fine-tuning, where it is desirable merely to enhance the original model in a small number of training epochs. The configuration of parameters is usually done in a trial and error manner, although some prior knowledge can substantially narrow the initial options. The following sections describe the fine-tuning process and justify the selection of the parameters.

4.2.1 Preprocess

Despite the fact that FCN can cope with arbitrarily sized images, the kernels inside have a fixed size, meaning the detail the network percepts changes, and so does the descriptor. Rescaling all inputs to have the same size should ensure more comparable descriptors. However, downscaling of large images would cause a loss of valuable information, and upscaling of small images would have to incorporate some interpolation, adding artifacts. Large image size also dramatically prolongs the training time, if they ever fit on the GPU memory. Another thing which has to be considered is the image size on which the network was pre-trained. Closely matching the original input size should help achieve the results for which the network was trained. The same applies to the evaluation phase, as to get the best results, the test set images should be resized to the original inputs' size used during training. While my training images are mostly high resolution and would benefit from lesser compression, the test sets' images are much smaller. One could argue that I should not pay any attention to the test sets, or else the evaluation will be biased. Although, in this case, the test sets better represent the target domain than my train set, which was explicitly collected in high resolution to have the option of downsizing without adding noise. Lastly, the original aspect ratio is preserved as in multiscaling. The rescaled images only share the same size of their larger side.

Before training, I have evaluated the pre-trained model on a variety of scales. As expected, the mAP score on METU is higher for smaller sizes, and another way around for NPU-TM and the train set. According to these results, all input images are resized to 256 pixels, which still runs reasonably fast during training.

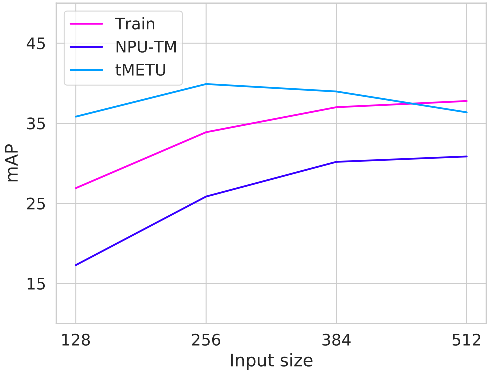


Figure 4.2: The influence of input size on both train and test set score. Input size corresponds to the length of the longer image side.

4.2.2 Random augmentations

When the training set is limited, its variance is also bounded and it probably does not contain all possible images of the domain. For example, a model trained only on colored images might provide unsatisfying representations of greyscale images. Small train set size also often quickly leads to overfitting. A common trick alleviating this problem is to alter the input image with some probability during the training. Such alterations are denoted *random augmentations*. The amount of possible augmentations effectively enlarges the

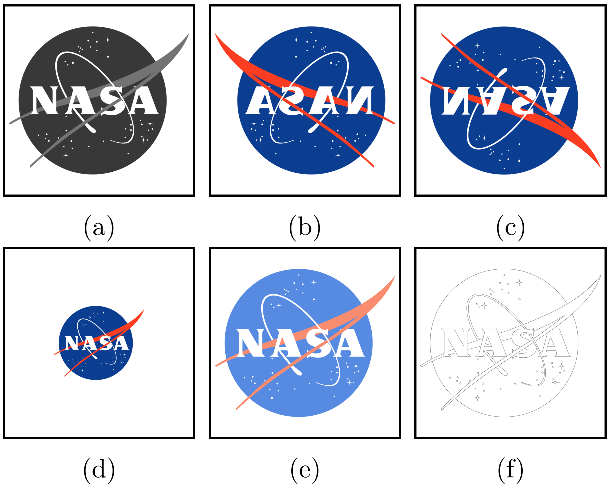

(a) (b) (c)
(d) (e) (f)

Figure 4.3: The effect of random augmentations applied to the same original logo.

input set size by a linear factor. If the augmentations carry any randomness, then the amount of potential inputs is virtually infinite.

There are many possibilities for random augmentations. Naturally, we do not want to change the image too drastically so the images coming from the same class would not share similarities anymore. I consider and test these six:

(a) *Grayscale* makes the image black and white only.

(b) *Horizontal flip* along width dimension.

(c) *Vertical flip* along height dimension.

(d) *Rescale* the image randomly between given upper and lower bound

(e) *Color Jitter* randomly changes the brightness, saturation and contrast up to the given factor.

(f) *Edge detection* converts the image to 0 to 1 values using any standard edge detection method. The higher intensity values indicate sharp transitions between contrasts, i.e., edges.

Due to the translation invariance property of global pooling operation and the fact that logo images are usually centered, I do not use random cropping augmentation, which is often utilized in computer vision scenarios.

Most of the augmentations are already available in the PyTorch library. As the augmentations are applied sequentially with some random probability, more of them may coincide, which creates even more variations. Trying all possible combinations and probabilities would take a lot of training runs.

```
$X \leftarrow$ load train data
$model \leftarrow$ initialize network
$best\_score \leftarrow 0$
if semi-supervised setting then
    positives $\leftarrow$ mine positives from $X$
else
    positives $\leftarrow$ hashmap with labels
end
negatives $\leftarrow$ empty hashmap

for $epoch \leftarrow 0$ to 100 do
    negatives $\leftarrow$ mine negatives from $X$

    foreach $anchor \in X$ do
        $triplets \leftarrow [anchor,$ positives$[anchor],$ negatives$[anchor]]$
        apply random augmentations to each triplet
        $model \leftarrow$ train($model$, $triplets$)
    end

    $current\_score \leftarrow$ evaluate($model$)
    if $current\_score \geq best\_score$ then
        $best\_score \leftarrow current\_score$
        save($model$)
    end
end
```

Figure 4.4: Simplified pseudo code of the main network training loop.

Instead, I tuned the parameters of each augmentation separately and then combined those that showed any improvement.

My final selection consists of: (1) *rescale*, where uniformly sampling from a range around the image size of 256 was slightly better than a random choice between three image sizes; (2) *color jitter* with brightness and contrast values up to 0.4 times the original and saturation up to 0.1; (3) *horizontal flip* and (4) *grayscale*, both with 0.1 probability. Surprisingly, the vertical flip did not provide any performance boost.

Similar to Section 4.1.1, I attempted to enforce the model to focus on the shape information in the form of random augmentation. Unfortunately, the results were poor, presumably because the network cannot grasp the similarity relation in a pair of colored input and drastically different edge image.

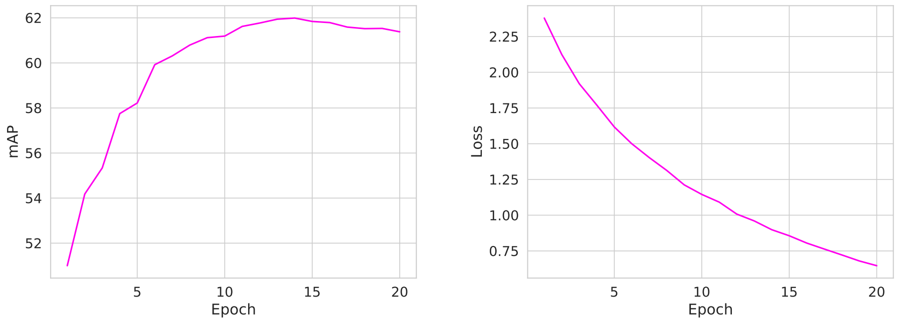
4.2.3 Implementation details

For the implementation of my training and testing scripts, I use Python (version 3.7) with PyTorch (version 1.4) as the main deep learning framework. The examples in *cirtorch* GitHub repository [48] served as the starting point. Figure 4.4 portrays a pseudo code of the simplified main training loop.

The negative sampling for training triplets is done at the beginning of every epoch. For each input image, its negatives are sampled as the most similar labeled images from other classes. The amount of selected negatives is equal to the number of labeled positives to form the corresponding triplets. To secure higher variance, all negatives of a single input are strictly forced to be from distinct classes, but still in the descending order of similarity.

The model is evaluated by mAP on the validation set after each epoch. Whenever the score on the validation set surpasses the previously highest value, the model weights are saved. If the model begins to overfit on the train set, it also performs worse on the validation set and the optimal model remains preserved. It is not a robust approach as it might fail if the training is unstable, but it worked well for my case. An example of validation mAP score and the average loss of the first 20 epochs is visualized in Figure 4.5. The model is also evaluated on tMETU every five epochs for sanity check during training.

I use a standard gradient-based optimizer called Adam, with its two parameters β set to default values (0.9, 0.999). The learning rate starts at 5^{-7} and exponentially decays by -0.01 each epoch. GeM global pooling, as defined in Equation 2.13 is either set to the value provided by the pre-trained network or equal 3 when not. After a couple of trial runs, I have settled on triplet loss and tested the margin parameter ranging from 0.15 to 0.85, where



(a) mAP score on validation set for each epoch. The highest value corresponds to the optimal model, which is obtained after 14 epochs.

(b) Average model loss smoothly decreases to zero.

Figure 4.5: Validation score and network loss per epoch.

the best observed results were for the parameter equal to 0.3. A training batch is comprised of 5 random anchors with all of their triplets.

All networks are trained either on Nvidia 1080Ti GPU or Nvidia Tesla T4. Some parts of the pipeline are processed on Intel(R) Xeon(R) CPU E5-2620 v4 with 256GB RAM. The total time needed for one training epoch using the whole labeled dataset (17992 triplets) is around 35 minutes.

4.2.4 Whitening

I utilize both types of whitening, as described in Section 2.3. First during the training, in the form of a fully connected layer with weights pre-trained on the SfM120k dataset. In this case, the layer preserved the original dimension.

Second, the PCA whitening on the final descriptors obtained by combining multiple scales and EdgeMAC with the fine-tuned network. The eigenvectors are estimated from the whole training set, including the unlabeled data. For this case, I have compared several output dimensions and the optimal β parameter for shrinkage. Keeping the number of output dimensions the same as on input and using shrinkage with β equal to the 64-th largest eigenvalue has yielded the best results. In each case, every outputted descriptor from the global pooling layer is (1) L2 normalized, (2) whitened, and (3) L2 normalized again.

4.3 Results

The results of fine-tuning performed on two most promising pre-trained networks are displayed in Table 4.3. After training, the model initially trained on ImageNet managed to shrink the lead of the second pre-trained model, but ultimately still performed worse. In the rest of the thesis, I refer to the fine-tuned ResNet101, pre-trained on SfM120k, as *baseline*.

Table 4.3: mAP score of both fine-tuned models and their score prior to training for comparison. Fine-tuning network pre-trained on SfM120k provided the best results and will be referred to as baseline in the rest of the thesis.

Training set	Arch	Pooling	tMETU	METU	NPU-TM
ImageNet	ResNet101	GeM	38.0	16.0	29.7
SfM120k	ResNet101	GeM	40.0	17.1	33.9
ImageNet → Trademarks	ResNet101	GeM	53.3	21.5	42.9
SfM120k → Trademarks	ResNet101	GeM	55.6	22.1	43.2

To confirm the contribution of multiscaling and EdgeMAC even after fine-tuning and in combination with whitening of the descriptors, several ablations were carried out. Each of these optional processing methods is beneficial on its own and even in combinations. Using a fine-tuned network with EdgeMAC, multiscaling, and PCA whitening achieved optimal mAP of 26.7 on METU and 47.0 on NPU-TM datasets. All results are printed in Table 4.4.

Table 4.4: mAP for baseline model with optional processing methods.

Model + method	tMETU	METU	NPU-TM
Baseline	55.6	22.1	43.2
Baseline + PCA_W	56.2	22.9	44.1
Baseline + EdgeMAC + PCA_W	61.3	24.4	45.1
Baseline + EdgeMAC + multiscale	64.1	25.8	45.4
Baseline + EdgeMAC + multiscale + PCA_W	64.3	26.7	47.0

Besides the quantitative results in tables, visual observations of retrieved images provide additional validation. The decision-making process of a trained neural network is typically difficult to explain. However, in some cases, a common occurrence can be deduced. The Figures 4.6, 4.7, and 4.8 demonstrate such cases in logo retrieval of both test sets. The query image is depicted in a black border as the leftmost images. The sequence of the most similar images to the query is ordered from left to right.

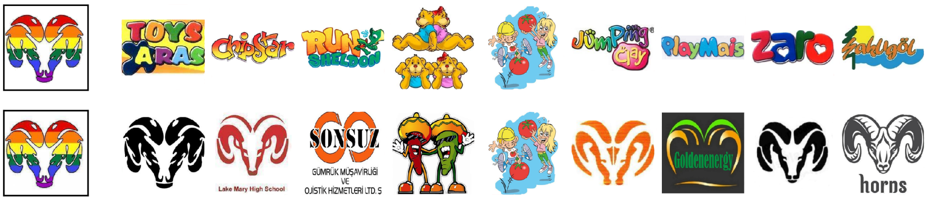


Figure 4.6: Top 9 images retrieved from the whole METU test set for a given query visualized on the left. The upper row is the result of pre-trained SfM120k–ResNet101. The bottom rows are provided by the same network but fine-tuned and combined with EdgeMAC. Despite the expressive colors, the main similarity is captured primarily by a distinct shape.

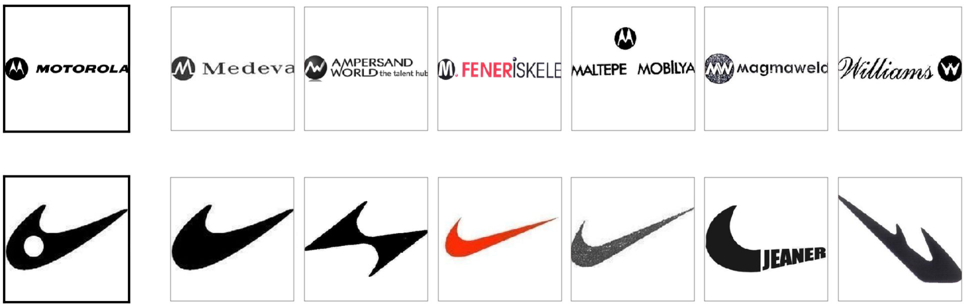


Figure 4.7: Two retrieval examples from the whole METU provided by the fine-tuned network. Simple shapes with solid color are usually well separated in the representation space. The top example shows that the model does not put too much attention to the text.

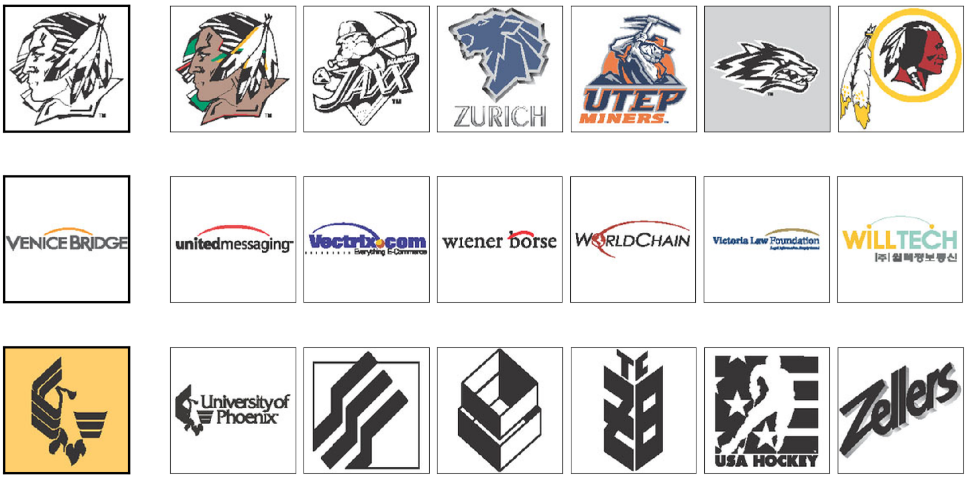


Figure 4.8: Retrieved images from NPU-TM test set by the fine-tuned network. The topmost row shows that the network often finds images with similar texture or style, even though the content is not necessarily conflicting. The examples in the second row share the same idea and are correctly grouped. If the image contains some distinct pattern, then the model frequently puts too much attention on it, which can be seen in the examples with stripes in the third row.

Chapter 5

Semi-supervised methods

I have established a baseline trained on all labeled data in the previous section. Positives for each training anchor were all images from the same class. Negatives were sampled as the most similar images from different classes. This section will discuss possible methods for positive and negative mining, which utilizes both labeled and unlabeled data.

5.1 Semi-supervised setting

During the manual phase of my dataset collection, I had to narrow the downloaded selection so the classes would only comprise the most mutually similar examples. Only around a fifth of the total amount of obtained logos is in the final train set. I kept the remainder of the images as an unlabeled set, denoted X_U, without any processing, which means it is quite noisy and contains duplicate images or images that are not trademarks. Even so, the unlabeled collection is not very large and probably does not carry many potentially positive images to most of my classes. It is only my presumption, as the labeling process is very subjective. In some cases, the network may even benefit more from the images that I did not regard as similar. All in all, I have decided to restrict my baseline further to ensure that potential positives are present and can be mined from the unlabeled set. I randomly selected 100 classes that have at least three positives. From these classes, I randomly chose positive pairs and positive triples. Since the restricted baseline contains only n images per class, the rest of the previously labeled are reckoned as unlabeled.

In Table 5.1, I show the comparison between test scores of networks fine-tuned with only limited amount of classes and either all labeled images, or random pairs/triples in each.

Training model with any other number of images per class n, its resulting mAP would lie between the cases with pairs and all. The two values reported in the table act as a soft lower and upper bound for the semi-supervised fine-tuning. My proposed methods start with the same limited n. If they suc-

Table 5.1: Evaluation score for models trained with a limited train set. The architecture and parameters are equivalent, as defined in Section 4.2.3

Model	classes	n	tMETU	METU	NPU-TM
Baseline	all	all	55.6	22.1	43.2
Baseline	50	2	45.8	18.7	37.5
Baseline	50	3	48.4	19.2	38.3
Baseline	50	all	51.7	19.7	40.2
Baseline	100	2	48.7	19.2	39.2
Baseline	100	3	51.0	19.7	40.8
Baseline	100	all	54.0	20.8	42.6

cessfully mine new positives, the result should also be between the two. If the new positives are not mined correctly, then the test score will be significantly smaller than the lower bound, which I inadvertently empirically verified.

Having prior information about correct labels allows for Precision evaluation measure, which is especially convenient for tuning the parameters of my methods without the need for time-consuming training.

5.2 Positive mining

The only tools available for additional positive mining are the high-dimensional descriptors produced by a pre-trained network and the information about the similarity of a few samples. A first naive approach is to consider the immediate area around each descriptor in the metric space. Utilizing the k-nearest neighbors algorithm (k-NN) gives us k potential new positives for each sample. The problem is that having similar trademarks closely represented, meaning as nearest neighbors out of the whole set of inputs, is the final goal to achieve. If this were already the case, there would be no need for positive mining or even training. The pre-trained networks will typically not have a reasonable precision rate. To see how successful is the current model on finding new positives in the form of k-nearest neighbors, I show the ratio of true and false positives for the setting with 100 triples in the Figure 5.1. Even with k equal only 6, the number of false positives is too high, and training them would produce an unsatisfying network.

5.2.1 Postprocessed descriptors

The initial vectors for the nearest neighbor search can be arbitrary, as long as they provide good results. Since the proposed techniques of shape matching and multiscaling, shown in Sections 4.1.1 and 4.1.2, improved the overall mAP

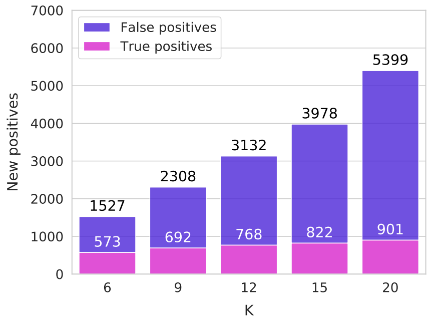
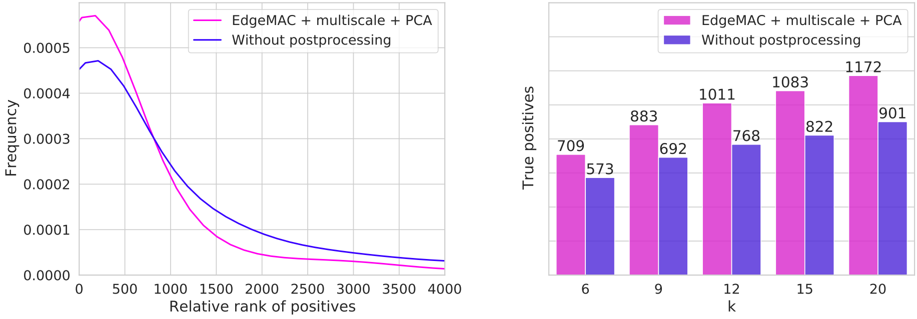


Figure 5.1: Hit rate of simple euclidean k-NN positive mining.



(a) Probability that input's positives will be ranked at a position on x axis, based on the histogram of measured counts.

(b) The amount of original positives present in k-NNs of all inputs.

Figure 5.2: Influence of optional processing steps on positives' ranks.

score on the test set, it means the descriptors are better at conveying similarities. Prior to the mining, I collect the descriptors in the same way as during the postprocessing phase, including edge detection, multiscaling, and PCA whitening. Even though the process takes six times the original amount, the benefits outweigh the additional time complexity. Resulting representations are less noisy, meaning the actual positives have less variance and are more closely grouped together. Figure 5.2a shows that positives of any given input are likely to be ranked at lower positions when the descriptors are postprocessed. The number of true positives found in k NNs of both versions, with and without postprocessing, can be seen in Figure 5.2b. The precision with improved representations is still too low for this approach to be considered for training. Nevertheless, it serves as an unsupervised starting point, which is improved by the proposed semi-supervised methods in the following sections. The goal is to drastically reduce the enormous amount of false positives while keeping most of the true positives. Their values, depicted above columns in the figure, are upper bounds to all of my methods.

5.2.2 Voting based

In my setting, all labeled images in one class are regarded as equally similar. During the creation of my train dataset, adding a new image into a specific class meant to observe whether it is sufficiently similar to all others visually. Making the mining algorithm imitate said procedure means the new positives should be decided collectively by the whole class.

The first considered possibility is to extend every class by descriptors that appear in the k-NN of each image descriptor belonging to it. The original positives are always honored; if S_i is the i-th class, initially consisting of two labeled images, then its extended equivalent

$$\hat{S}_i = S_i \cup \bigcap_{x \in S_i} \mathrm{NN}_k^e(x) \tag{5.1}$$

has $2 \leq |\hat{S}_i| \leq 2 + k$ elements after one run of this positive mining.

This method's strictness can be moderated only by k, i.e., the size of the individual nearest neighbor sets. While having more original positives secures more confidence in the newly mined positives, it may be too uncompromising. The true labels are not homogenous and often contain significant variance. Having a large number of samples to all agree on the same image would either be rare or require large k. Instead, the consensus can be made by using majority voting. The descriptor has to occur in a predefined portion of the NNs of the original group to be considered a new positive. This can be summarized as:

$$\hat{S}_i = S_i \cup \left\{ p \in X_U \mid \sum_{x \in S_i} [p \in \mathrm{NN}_k^e(x)] \geq \frac{|S_i|}{2} \right\}, \tag{5.2}$$

where $[\cdot]$ equals 1 when the condition inside is met and 0 otherwise.

Sometimes, one class's descriptors may form even smaller subgroups in the representation space due to the uneven decision boundary during labeling. An unlabeled image mapped as the proximate NN of a few class members may be a better candidate than an image represented somewhere in between all of them. The former would not pass the majority rule while the latter would. Taking into account both majority and the actual order of NNs could provide a decent trade-off.

I utilize another voting method for this problem, called *Borda count.* It assigns a decreasing amount of points to each preference candidate based on its position. The ordering is then achieved by summing all such points of all voters.

In my setting, the voters are the original labeled images $x \in S_i$, and the preference candidates are their ordered k-nearest neighbors sets $\mathrm{NN}_k^e(x)$. The first NN of x, i.e., the closest descriptor, receives k points, the second receives

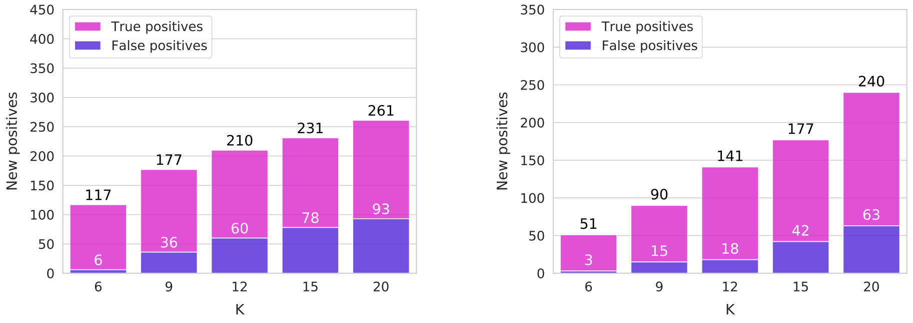
$k-1$, and so on. The majority vote is achieved by allowing only the candidates with a certain amount of points b. For instance, when the class consists only of two labeled inputs and $k = 10$, setting $b > 10$ yields the same output as the majority rule above. On top of that, the resulting candidates are ordered. Instead of all class members sharing the same positives, the mining can be done separately per input by weighing others' preferences while favoring its own choices slightly more. An input may achieve this by assigning more points to its candidate neighbors. Let $\mathrm{B}_x(\cdot) \rightarrow \mathbb{N}_0$ be a mapping assigning integer points to an argument as k minus its zero-indexed position in the ordered sequence of $\mathrm{NN}^e_k(x)$, then the expanded class is

$$\hat{S}_i = \{p \in X_U \mid \sum_{x \in S_i} \mathrm{B}_x(p) \geq b\}. \tag{5.3}$$

Regulating the number of potential candidates by either intersection, majority vote, or Borda count, the total amount of mined positives is greatly reduced compared to the starting k-NN approach. Although it still could have happened that some classes have individually too many positives. The maximal desired number of mined positives per class depends on multiple things, such as the specific domain, size of unlabeled inputs' pool, or the portion of labeled classes. After experiencing the difficulty of obtaining multiple tightly similar images many times while collecting my train set, it does not make sense to mine more than a single digits of new positives per class. When the chosen method is either majority or Borda voting, keeping only the candidates with the highest counts should further improve the quality of mined positives.

In some cases, even the k-nearest neighbors of a sample x are too dissimilar. It is beneficial to bound the potential positives not only by the absolute amount k, but also by some minimal allowed cosine similarity. It is typically difficult to apply a fixed threshold to a similarity measure and cut-off the negatives, as it will not work equally well for all anchors. Therefore, I am using the average similarity between anchor and its original positives $p \in S_i \setminus \{x\}$ as a guide to set a threshold λ.

In total, the first two methods, intersection and majority vote, utilize three parameters: the k in the k-NN algorithm, maximal desired amount of new positives per class m and the coefficient λ. Borda count additionally uses the threshold amount of required points b and the optional multiplier of anchor's own preferences. I have tested extensive amount of these parameters' combinations for all three methods. It is not possible to firmly choose their optimal values, as I cannot reliably assess whether it is worth adding more positives at the expense of precision. Borda count dominated majority vote in all settings, while the approach with intersection was almost on par. Hit rates of the two methods with comparable parameters are depicted in Figure 5.3. As expected, the strict intersection works better when the total amount of mined positives is small and vice-versa. I tried four coefficients for weighing the anchor's own preferences during Borda count – 0.5, 1, and 1.5; none of them yielded better results.



(a) Intersection with $\lambda = 0.7,\ m = 6$

(b) Borda count with $\lambda = 0.7,\ m = 6,\ b = \frac{5}{2}k$

Figure 5.3: Comparison of voting based mining methods.

All three techniques operate in linear complexity with the size of labeled inputs, which makes the added time requirement neglibile in comparison with the whole training.

5.2.3 Graph based

The previous methods exploited the decisive power of a whole class but ignored any other information besides its proximal samples. The space of all representations produced by the FCN is extremely high-dimensional and sparse. As the k-NN algorithm works with only the relative distances, the nearest neighbors of an image will most probably lie in various directions. Even if all inputs of a single class share a common nearest neighbor, it may be located in the wrong direction compared to the rest of the original positives. Besides, the neighbors that are not shared only take the spot of potentially better candidates. Possibly because of these distracting neighbors in the wrong directions, weighing the individual preferences more in the Borda count method did not work. I will introduce a method where the original images take an individualistic approach to finding new positives. By imitating the representation space with a simplified version, I can employ a standard algorithm that searches for new positives in a directed way.

As before, I start with the k-nearest neighbor sets NN_k^e, but this time of all inputs, including the unlabeled images. I construct a graph where nodes are all inputs' descriptors. Undirected edge is present between two nodes if the two inputs are reciprocal nearest neighbors. All edges are weighted by the Euclidean distance between the nodes they are connecting. Thus, every node has a degree of at most k and at least zero. Such graph can be represent by

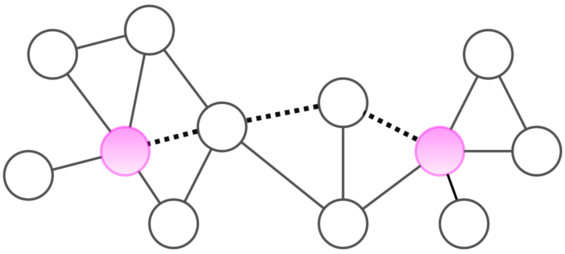


Figure 5.4: Visualization of graph based positive mining. The dashed line illustrates the shortest path between two labeled images. The vertices along this path are chosen to be the new added positives.

symmetric adjacency matrix $A = (a_{ij}) \in \mathbb{R}^{n \times n}$ [51] as

$$a_{ij} = \begin{cases} \left\| x_i - x_j \right\|_2, & \text{if } x_i \in \mathrm{NN}_k^e(x_j) \land x_j \in \mathrm{NN}_k^e(x_i), \\ 0, & \text{otherwise.} \end{cases} \tag{5.4}$$

Having this adjacency matrix allows for the use of a graph searching algorithm. The representations provided by the pre-trained network should already be somewhat clustered by class with additional noise. My idea is that a path-finding algorithm between two good representatives of a class will visit more positive nodes on the way. On the other hand, if the labeled pair is not similar enough, the path is going to be too long.

To further enforce the correct search direction, I incorporate the edge weights and use a variation of the A* search algorithm. Typically, the algorithm starts from a node, traverses closest nodes first, and ends when the goal node is visited.

In my situation, the search is, in total, performed x-times. For each of the labeled inputs $x \in X$, the start node is x, and the goals are its original positives from the same class $S_i \setminus \{x\}$. When the goal node is visited, all nodes along the path are collected and regarded as new positives. Additionally, if the amount of new positives is below the desired amount m, the goal node is removed from the set of visited nodes, and the search continues. The assumption that nodes along shortest paths should be related to both start and goal node will not hold for longer paths. For this reason, I limit their length with a *hop count* parameter. I keep the original positives as before; due to n and the finite number of paths with *hop count* length, the possible size of the extended class $\hat{S}_i$ initially having two images is $2 \leq |\hat{S}_i| \leq 2 + n$. A scenario where two original positives do not share any of their 5-NN is shown in Figure 5.4.

Since classes may contain more than two images, A* searches for multiple goals, where its heuristic estimating the right direction for multiple goals can be set in several ways. Two simple options for such heuristic are:

- *min* heuristic as the euclidean distance to the nearest goal

$$h_m(x) = \min_{g \in G} d(x, g). \tag{5.5}$$

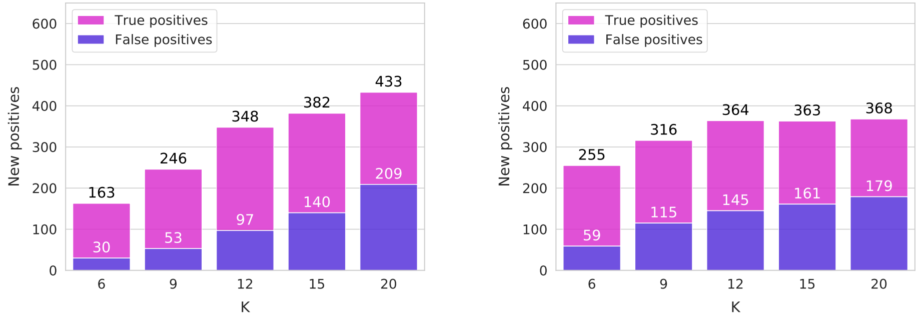
- *sum* heuristic as the sum of distances to all goals

$$h_s(x) = \sum_{g \in G} d(x, g). \tag{5.6}$$

When there is just a single searched goal, then these heuristics are equal. Otherwise, they mostly differ when the goals are located in orthogonal directions. The min heuristic will prefer straight paths to the nearest goal, while the sum heuristic will proceed to space located between the goals.

Same as in the voting methods, I use a threshold value to discard the graph edges with a weight higher than a predefined number λ to filter out overly remote representations. Unfortunately, I cannot set the threshold to depend on the original positives' distance because the graph also contains nodes from the unlabeled set. Also, the edge weights are raised to the power of a parameter p, to force the A* algorithm to prioritize paths with more nodes but lower distances in between.

I have tested the three mentioned parameters and both heuristics to observe the impact on the amount of true and false positives found. The results for the most promising parameters $m = 12$, $\lambda = 0.3$ and $p = 2$ and various values k are shown in Figure 5.5.



(a) Hop count equal 1 and $\lambda = 0.1$.

(b) Hop count equal 2 and $\lambda = 0.3$.

Figure 5.5: Comparison of graph based mining methods.

5.3 Negative mining

Selecting hard negatives is a crucial part of any deep metric learning pipeline using contrastive or triplet loss. In the supervised part, finding the hardest negatives was straight forward. With the assumption the labeled set can be completely trusted, selecting the most similar images from different classes will yield precisely the hardest negatives.

By incorporating the unlabeled set, the pool of potential negatives has increased; however, there is no strict class boundary between similar representations that are mined as positive and those that should be negative as both rely only on the relative closeness. With some additional filtering methods discussed in Section 5.2, the k-nearest neighbors of an image are regarded as the new positives.

For each anchor descriptor x^a of a training triplet, the rest of the inputs is ordered by decreasing cosine similarity. The anchor's positives are mined from the first k descriptors using any of the proposed methods. However, the original positives may be positioned much farther, especially when the descriptors are obtained before any training, as was shown in Figure 5.2a. While some classes are well separated, and the positives are closely clustered in the first ten or twenty positions, most are not. My goal is to find the optimal gap between the positions of positives and the starting position r, where the hard negatives are selected.

If the positives are relatively far from the first position, then the model does not represent this class well, and the inputs ranked at frontal positions are probably negatives. Conversely, when most of the positives are represented close to the anchor, the model already successfully separates this class. I implement this idea as a two-branched condition statement. If the mean position of the anchor's positives $\bar{p}$ is higher than 100, I select r to be relatively close to zero. Otherwise, the gap is defined as a multiple of $\bar{p}$. Naturally, the position r should not be too small to interfere with positive mining, so I offset it with a constant value. The whole approach can be summarized as:

$$r = \begin{cases} 2\bar{p} + 20, & \bar{p} \leq 100, \\ 50, & \text{otherwise.} \end{cases} \tag{5.7}$$

5.4 Evaluation

True and false positive numbers shown in the figures are only indications of the real performance. The total mined amount does not reflect the quality of the positives. If they are too easy, then they hardly contribute to the training. On the other hand, the unlabeled set most probably contains some images that are similar and beneficial for the training, but they cannot be counted towards true positives, as there is no available information about them. Figure 5.6 shows an example of four such positives. The ones found by Borda count method are more conservative, since they have to be agreed upon by both samples in the labeled pair. In contrast, the shortest path from labeled image on the left, to the image on the right, may be traversing through much harder positives.

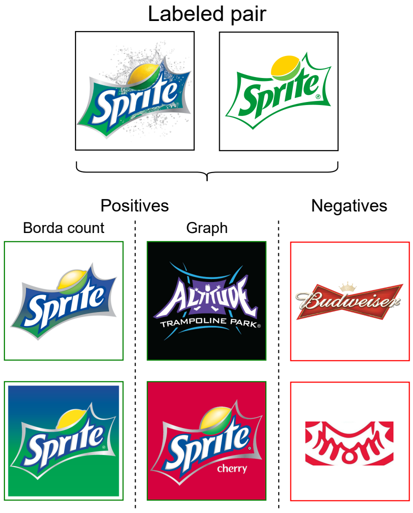


Figure 5.6: Top 2 new positives mined for the given labeled pair according to Borda count and Graph based methods. Mined negatives on the right for comparison.

All of the proposed methods exhibit similar behavior of progressively decreasing precision with the increase of total mined positives. It raises a common trade-off dilemma between precision and recall. It is not easy to assess whether false positives are more harmful than a smaller train set in metric learning. Moreover, the triplet loss does not necessarily punish wrong positives, as long as their corresponding triplet's negative is not genuinely similar.

The positives are mined using the representations provided by the concatenated pre-trained FCN and EdgeMAC descriptors for three scales of image. However, training with a limited train set of two or three positives per class allows for even more precise mining in subsequent epochs. Thus, I select the parameters that result in a small but reliable number of mined positives before the first epoch. The mining is then redone every 5 epochs, which progressively finds more positives, resulting in additional improvement of precision and recall ratio.

The influence of mined positives is limited by the anchor and also by the mined negatives. For instance, adding a single positive to a class S_i of n manually labeled images would only add n more training triplets. In contrast, considering the pseudo-labeled input as an anchor of class S_i would provide n^2 new triples. If it is guaranteed the unlabeled set contains potential positives and the model is already trusted to be representing the images decently, then it might be beneficial to regard the mined samples as equivalent to the original labeled inputs.

Table 5.2 presents mAP scores of all three designed methods. The technique of mining new positives as anchors is marked with an asterisk. As discussed in Section 5.1, the performance of the restricted baseline trained with only labeled pairs serves as a lower bound (row 1) for the semi-supervised methods. Conversely, the restricted model trained with all available labels serves as an upper bound (row 3). Ideally, none of the semi-supervised methods should perform worse with additional mined positives or anchors. This requirement is fulfilled in all cases. Whether the method's final mAP score is closer to the lower bound or to the upper bound depends on the test set. The test score is slightly higher for the rows with pseudo-labeled anchors, which confirms the correctness of the mined samples. The best overall is the intersection method, which is also fairly simple to implement, making it the optimal choice of the three. However, the graph based method performs surprisingly well on the large METU dataset, where it almost catches up with the upper bound baseline.

Table 5.2: Comparison between proposed semi-supervised methods and restricted baseline models. The rows marked with a star correspond to the training with pseudo-labeled anchors, while the same methods without a star used the pseudo-labeled inputs only as positives in the training triples.

Method	k	classes	n	tMETU	METU	NPU-TM
Baseline		100	2	48.7	19.2	39.2
Baseline		100	3	51.0	19.7	40.8
Baseline		100	all	54.0	20.8	42.6
Baseline		all	all	55.6	22.1	43.2
Intersection	6	100	2	51.9	19.9	40.4
Borda count	20	100	2	50.9	19.5	40.2
Graph	9	100	2	51.1	19.8	40.0
Intersection*	6	100	2	52.3	20.4	40.4
Borda count*	20	100	2	51.7	19.8	39.6
Graph*	9	100	2	51.3	20.6	39.5

Lastly, I combine the model trained by the intersection approach with EdgeMAC, multiscaling, and PCA whitening, to verify the improvement is not mitigated. The results are visible in Table 5.3.

Table 5.3: Comparison between restricted baselines and the best semi-supervised method in combination with optional enhancing techniques described earlier.

Method		classes	n	tMETU	NPU-TM
Baseline	+ EdgeMAC + ms + PCA_W	100	2	58.8	43.9
Intersection*	+ EdgeMAC + ms + PCA_W	100	2	61.0	44.7
Baseline	+ EdgeMAC + ms + PCA_W	100	all	63.1	45.9

5.5 Discussion and future work

Even though the proposed semi-supervised methods boosted the baseline performance in this setting, they would have to be adequately tested on other datasets and domains to verify their robustness.

There is a lot of possible improvements to my methods. Both graph and voting rule approaches used heuristics in search of new positives. The heuristics were relatively simple; more options should be tested. For instance, during the A* shortest path search, any information about the graph's density is ignored. Besides the minimal distance, a degree of a node could be incorporated to force the algorithm to search first in populated areas. In the Borda count method, the scoring could be improved by considering the similarity value along with the position.

I have already mentioned the difficulties of labeling images into discrete classes. The newly found positives and negatives are also varying in quality. One way of reducing the strictness of training with discrete labels is to weigh the metric learning losses with the confidence of similarity between anchor, positive, and negative sample [51]. That way, the key factor would not be the quantity ratio of true and false positives but rather the quality of mined samples.

State of the art practice in semi-supervised learning uses a consistence regularization, which is completely orthogonal to my approach. In addition, both techniques can be combined, which could potentially further improve the results.

The mining of positives is done utilizing similarities between the global descriptors. There is frequently only a single distinct aspect that distinguishes the specific trademark. In theory, using local descriptors for the search of positives could prove beneficial.

Conclusion

I have verified a common theme occurring in the related research that the task of trademark image retrieval is dependent on multiple distinct image aspects, such as color, text, or shape. Obtaining a general model sufficiently differentiating trademarks according to all of these at once is a challenging task. A viable approach is to combine several models trained specifically for the individual subtasks. In my work, I have explored this approach by incorporating neural network pre-trained on edge images to better infer the shape information. Presented results, both quantitative and qualitative, proved the network's ability to transfer knowledge to the domain of trademark images even without fine-tuning.

Combining the edge oriented network with a network pre-trained on standard RGB images further improves the model performance. I have compared several network architectures and selected ResNet101, which provided optimal results while being reasonably fast. To adapt the RGB network to this particular domain, I have fine-tuned it on a self-collected and annotated trademark image dataset, consisting of more than two thousand images. Concatenating the embeddings obtained from both edge and RGB networks achieved mAP of 26.7 on METU, the largest trademark dataset available.

Acquiring additional data is relatively straightforward in today's world. However, the manual labeling part is a major complication, especially in domains where the class boundaries are not exact and highly subjective, such as logo images. In the second part of this thesis, I designed and evaluated three semi-supervised methods operating in the deep metric learning setting. All of them take advantage of the similarity relations in the learned representation space. Starting with only a small fraction of original labels, the best performing method managed to almost catch up to the model trained with the whole labeled set.

Bibliography

1. GOOGLE, LLC. *How Google Fights Piracy* [online] [visited on 2021-01-04]. Available from: https://www.blog.google/documents/25/GO806_Google_FightsPiracy_eReader_final.pdf.
2. AHMED, Ejaz; JONES, Michael; MARKS, Tim K. An Improved Deep Learning Architecture for Person Re-Identification. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* [online]. Boston, MA, USA: IEEE, 2015, pp. 3908–3916 [visited on 2020-12-30]. ISBN 978-1-4673-6964-0. Available from DOI: 10.1109/CVPR.2015.7299016.
3. HERMANS, Alexander; BEYER, Lucas; LEIBE, Bastian. *In Defense of the Triplet Loss for Person Re-Identification* [online]. 2017 [visited on 2020-12-30]. Available from arXiv: 1703.07737 [cs].
4. YI, Dong; LEI, Zhen; LI, Stan Z. *Deep Metric Learning for Practical Person Re-Identification* [online]. 2014 [visited on 2020-12-30]. Available from arXiv: 1407.4979 [cs].
5. SCHROFF, F.; KALENICHENKO, D.; PHILBIN, J. FaceNet: A Unified Embedding for Face Recognition and Clustering. In: [online]. 2015, pp. 815–823 [visited on 2020-12-30]. Available from: https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Schroff_FaceNet_A_Unified_2015_CVPR_paper.html.
6. HU, Junlin; LU, Jiwen; TAN, Yap-Peng. Discriminative Deep Metric Learning for Face Verification in the Wild. In: [online]. 2014, pp. 1875–1882 [visited on 2020-12-30]. Available from: https://openaccess.thecvf.com/content_cvpr_2014/html/Hu_Discriminative_Deep_Metric_2014_CVPR_paper.html.

7. LIU, Jingtuo; DENG, Yafeng; BAI, Tao; WEI, Zhengping; HUANG, Chang. Targeting Ultimate Accuracy: Face Recognition via Deep Embedding [online]. 2015 [visited on 2020-12-30]. Available from: `https://arxiv.org/abs/1506.07310v4`.

8. RADENOVIĆ, Filip; TOLIAS, Giorgos; CHUM, Ondřej. *Deep Shape Matching* [online]. 2018 [visited on 2020-11-27]. Available from arXiv: `1709.03409 [cs]`.

9. WANG, Fang; KANG, Le; LI, Yi. Sketch-Based 3D Shape Retrieval Using Convolutional Neural Networks. In: [online]. 2015, pp. 1875–1883 [visited on 2020-12-30]. Available from: `https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Wang_Sketch-Based_3D_Shape_2015_CVPR_paper.html`.

10. DAI, Guoxian; XIE, Jin; FANG, Yi. Deep Correlated Holistic Metric Learning for Sketch-Based 3D Shape Retrieval. *IEEE transactions on image processing: a publication of the IEEE Signal Processing Society*. 2018, vol. 27, no. 7, pp. 3374–3386. ISSN 1941-0042. Available from DOI: `10.1109/TIP.2018.2817042`.

11. NI, Jiazhi; LIU, Jie; ZHANG, Chenxin; YE, Dan; MA, Zhirou. Fine-Grained Patient Similarity Measuring Using Deep Metric Learning. In: 2017, pp. 1189–1198. Available from DOI: `10.1145/3132847.3133022`.

12. ANNARUMMA, Mauro; MONTANA, Giovanni. Deep Metric Learning for Multi-Labelled Radiographs. In: *Proceedings of the 33rd Annual ACM Symposium on Applied Computing* [online]. New York, NY, USA: Association for Computing Machinery, 2018, pp. 34–37 [visited on 2020-12-30]. SAC '18. ISBN 978-1-4503-5191-1. Available from DOI: `10.1145/3167132.3167379`.

13. TURSUN, Osman; AKER, Cemal; KALKAN, Sinan. *A Large-Scale Dataset and Benchmark for Similar Trademark Retrieval* [online]. 2017 [visited on 2020-11-27]. Available from arXiv: `1701.05766 [cs]`.

14. TURSUN, Osman; DENMAN, Simon; SIVAPALAN, Sabesan; SRIDHARAN, Sridha; FOOKES, Clinton; MAU, Sandra. Component-Based Attention for Large-Scale Trademark Retrieval. *IEEE Transactions on Information Forensics and Security* [online]. 2020, pp. 1–1 [visited on 2020-12-28]. ISSN 1556-6013, 1556-6021. ISSN 1556-6013, 1556-6021. Available from DOI: `10.1109/TIFS.2019.2959921`.

15. LAN, Tian; FENG, X.; XIA, Zhaoqiang; PAN, Shijie; PENG, Jinye. Similar Trademark Image Retrieval Integrating LBP and Convolutional Neural Network. In: *ICIG*. 2017. Available from DOI: `10.1007/978-3-319-71598-8_21`.

16. LAN, T.; FENG, X.; LI, L.; XIA, Z. Similar Trademark Image Retrieval Based on Convolutional Neural Network and Constraint Theory. In: *2018 Eighth International Conference on Image Processing Theory, Tools and Applications (IPTA)*. 2018, pp. 1–6. ISSN 2154-512X. Available from DOI: `10.1109/IPTA.2018.8608162`.

17. PEREZ, C. A.; ESTÉVEZ, P. A.; GALDAMES, F. J.; SCHULZ, D. A.; PEREZ, J. P.; BASTÍAS, D.; VILAR, D. R. Trademark Image Retrieval Using a Combination of Deep Convolutional Neural Networks. In: *2018 International Joint Conference on Neural Networks (IJCNN)*. 2018, pp. 1–7. ISSN 2161-4407. Available from DOI: `10.1109/IJCNN.2018.8489045`.

18. TRAPPEY, Charles V.; TRAPPEY, Amy J. C.; LIN, Sam C. -C. Intelligent Trademark Similarity Analysis of Image, Spelling, and Phonetic Features Using Machine Learning Methodologies. *Advanced Engineering Informatics* [online]. 2020, vol. 45, pp. 101120 [visited on 2020-12-28]. ISSN 1474-0346. Available from DOI: `10.1016/j.aei.2020.101120`.

19. KO, Kyung Pyo; LEE, Kwang Hee. 2-Gram-Based Phonetic Feature Generation for Convolutional Neural Network in Assessment of Trademark Similarity, pp. 10.

20. CHEN, Ting; KORNBLITH, Simon; SWERSKY, Kevin; NOROUZI, Mohammad; HINTON, Geoffrey. *Big Self-Supervised Models Are Strong Semi-Supervised Learners* [online]. 2020 [visited on 2020-12-29]. Available from arXiv: `2006.10029 [cs, stat]`.

21. BEYER, Lucas; ZHAI, Xiaohua; OLIVER, Avital; KOLESNIKOV, Alexander. S4L: Self-Supervised Semi-Supervised Learning. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* [online]. Seoul, Korea (South): IEEE, 2019, pp. 1476–1485 [visited on 2020-12-28]. ISBN 978-1-72814-803-8. Available from DOI: `10.1109/ICCV.2019.00156`.

22. LAINE, Samuli; AILA, Timo. *Temporal Ensembling for Semi-Supervised Learning* [online]. 2017 [visited on 2021-01-04]. Available from arXiv: `1610.02242 [cs]`.

23. TARVAINEN, Antti; VALPOLA, Harri. *Mean Teachers Are Better Role Models: Weight-Averaged Consistency Targets Improve Semi-Supervised Deep Learning Results* [online]. 2018 [visited on 2020-12-29]. Available from arXiv: `1703.01780 [cs, stat]`.

24. MIYATO, Takeru; MAEDA, Shin-ichi; KOYAMA, Masanori; ISHII, Shin. *Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning* [online]. 2018 [visited on 2020-12-29]. Available from arXiv: `1704.03976 [cs, stat]`.

25. LEE, Dong-Hyun. Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks. *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)*. 2013.

26. ISCEN, Ahmet; TOLIAS, Giorgos; AVRITHIS, Yannis; CHUM, Ondrej. Label Propagation for Deep Semi-Supervised Learning. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* [online]. Long Beach, CA, USA: IEEE, 2019, pp. 5065–5074 [visited on 2020-12-29]. ISBN 978-1-72813-293-8. Available from DOI: `10.1109/CVPR.2019.00521`.

27. HAEUSSER, Philip; MORDVINTSEV, Alexander; CREMERS, Daniel. Learning by Association — A Versatile Semi-Supervised Training Method for Neural Networks. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* [online]. Honolulu, HI: IEEE, 2017, pp. 626–635 [visited on 2020-12-29]. ISBN 978-1-5386-0457-1. Available from DOI: `10.1109/CVPR.2017.74`.

28. SUÁREZ-DÍAZ, Juan Luis; GARCÍA, Salvador; HERRERA, Francisco. *A Tutorial on Distance Metric Learning: Mathematical Foundations, Algorithms, Experimental Analysis, Prospects and Challenges (with Appendices on Mathematical Background and Detailed Algorithms Explanation)* [online]. 2020 [visited on 2020-12-02]. Available from arXiv: `1812.05944 [cs, stat]`.

29. WEINBERGER, Kilian Q.; SAUL, Lawrence K. Distance Metric Learning for Large Margin Nearest Neighbor Classification. *Journal of Machine Learning Research*. 2009, vol. 10, no. 9, pp. 207–244. Available also from: `http://jmlr.org/papers/v10/weinberger09a.html`.

30. KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks. *Communications of the ACM* [online]. 2017, vol. 60, no. 6, pp. 84–90 [visited on 2021-01-04]. ISSN 0001-0782, 1557-7317. ISSN 0001-0782, 1557-7317. Available from DOI: `10.1145/3065386`.

31. HADSELL, R.; CHOPRA, S.; LECUN, Y. Dimensionality Reduction by Learning an Invariant Mapping. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR'06)* [online]. New York, NY, USA: IEEE, 2006, vol. 2, pp. 1735–1742 [visited on 2020-12-02]. ISBN 978-0-7695-2597-6. Available from DOI: `10.1109/CVPR.2006.100`.

32. SCHROFF, F.; KALENICHENKO, D.; PHILBIN, J. FaceNet: A Unified Embedding for Face Recognition and Clustering. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 815–823. ISSN 1063-6919. Available from DOI: `10.1109/CVPR.2015.7298682`.

33. YU, Baosheng; LIU, Tongliang; GONG, Mingming; DING, Changxing; TAO, Dacheng. Correcting the Triplet Selection Bias for Triplet Loss. In: FERRARI, Vittorio; HEBERT, Martial; SMINCHISESCU, Cristian; WEISS, Yair (eds.). *Computer Vision – ECCV 2018*. Cham: Springer International Publishing, 2018, pp. 71–86. Lecture Notes in Computer Science. ISBN 978-3-030-01231-1. Available from DOI: `10.1007/978-3-030-01231-1_5`.

34. WU, C.-Y.; MANMATHA, R.; SMOLA, A. J.; KRÄHENBÜHL, P. *Sampling Matters in Deep Embedding Learning* [online]. 2018 [visited on 2020-12-02]. Available from arXiv: `1706.07567 [cs]`.

35. LONG, Jonathan; SHELHAMER, Evan; DARRELL, Trevor. Fully Convolutional Networks for Semantic Segmentation. In: [online]. 2015, pp. 3431–3440 [visited on 2021-01-04]. Available from: `https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Long_Fully_Convolutional_Networks_2015_CVPR_paper.html`.

36. SIMONYAN, Karen; ZISSERMAN, Andrew. *Very Deep Convolutional Networks for Large-Scale Image Recognition* [online]. 2015 [visited on 2020-12-03]. Available from arXiv: `1409.1556 [cs]`.

37. HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. *Deep Residual Learning for Image Recognition* [online]. 2015 [visited on 2020-11-24]. Available from arXiv: `1512.03385 [cs]`.

38. COOK, Alexis. *Global Average Pooling Layers for Object Localization* [online] [visited on 2020-12-06]. Available from: `https://alexisbcook.github.io/2017/global-average-pooling-layers-for-object-localization/`.

39. RADENOVIĆ, Filip; TOLIAS, Giorgos; CHUM, Ondřej. *Fine-Tuning CNN Image Retrieval with No Human Annotation* [online]. 2018 [visited on 2020-12-06]. Available from arXiv: `1711.02512 [cs]`.

40. TOLIAS, Giorgos; SICRE, Ronan; JÉGOU, Hervé. *Particular Object Retrieval with Integral Max-Pooling of CNN Activations* [online]. 2016 [visited on 2020-12-06]. Available from arXiv: `1511.05879 [cs]`.

41. BABENKO, Artem; LEMPITSKY, Victor. *Aggregating Deep Convolutional Features for Image Retrieval* [online]. 2015 [visited on 2020-12-06]. Available from arXiv: `1510.07493 [cs]`.

42. JÉGOU, Hervé; CHUM, Ondrej. Negative Evidences and Co-Occurrences in Image Retrieval: The Benefit of PCA and Whitening. In: [online]. 2012 [visited on 2021-01-04]. Available from: `https://hal.inria.fr/hal-00722622`.

43. MUKUNDAN, Arun; TOLIAS, Giorgos; BURSUC, Andrei; JÉGOU, Hervé; CHUM, Ondřej. *Understanding and Improving Kernel Local Descriptors* [online]. 2018 [visited on 2020-12-09]. Available from arXiv: 1811.11147 [cs].

44. ZHU, Mu. Recall, Precision and Average Precision [online]. 2004 [visited on 2020-12-28]. Available from: https://web.archive.org/web/20110504130953/http://sas.uwaterloo.ca/stats_navigation/techreports/04WorkingPapers/2004-09.pdf.

45. *Trademark Electronic Search System (TESS)* [online] [visited on 2020-11-28]. Available from: http://tmsearch.uspto.gov.

46. DENG, J.; DONG, W.; SOCHER, R.; LI, L.; KAI LI; LI FEI-FEI. ImageNet: A Large-Scale Hierarchical Image Database. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. ISSN 1063-6919. Available from DOI: 10.1109/CVPR.2009.5206848.

47. RADENOVIĆ, Filip; TOLIAS, Giorgos; CHUM, Ondřej. CNN Image Retrieval Learns from BoW: Unsupervised Fine-Tuning with Hard Examples. In: LEIBE, Bastian; MATAS, Jiri; SEBE, Nicu; WELLING, Max (eds.). *Computer Vision – ECCV 2016* [online]. Cham: Springer International Publishing, 2016, vol. 9905, pp. 3–20 [visited on 2020-12-13]. Lecture Notes in Computer Science. ISBN 978-3-319-46447-3 978-3-319-46448-0. Available from DOI: 10.1007/978-3-319-46448-0_1.

48. RADENOVIĆ, Filip; TOLIAS, Giorgos. *CNN Image Retrieval in Pytorch* [online] [visited on 2020-12-13]. Available from: https://github.com/filipradenovic/cnnimageretrieval-pytorch.

49. DOLLAR, Piotr; ZITNICK, C. Lawrence. Structured Forests for Fast Edge Detection. In: *2013 IEEE International Conference on Computer Vision* [online]. Sydney, Australia: IEEE, 2013, pp. 1841–1848 [visited on 2020-12-15]. ISBN 978-1-4799-2840-8. Available from DOI: 10.1109/ICCV.2013.231.

50. GEIRHOS, Robert; RUBISCH, Patricia; MICHAELIS, Claudio; BETHGE, Matthias; WICHMANN, Felix A.; BRENDEL, Wieland. *ImageNet-Trained CNNs Are Biased towards Texture; Increasing Shape Bias Improves Accuracy and Robustness* [online]. 2019 [visited on 2021-01-04]. Available from arXiv: 1811.12231 [cs, q-bio, stat].

51. ISCEN, Ahmet; TOLIAS, Giorgos; AVRITHIS, Yannis; CHUM, Ondrej. *Mining on Manifolds: Metric Learning without Labels* [online]. 2018 [visited on 2020-12-21]. Available from arXiv: 1803.11095 [cs].

Appendix A

Acronyms

DMCA Digital Millenium Copyright Act

WIPO World Intellectual Property Organization

USPTO United States Patent and Trademark Office

CNN Convolutional Neural Network

FCN Fully Convolutional Network

k-NN k-Nearest Neighbors

FLOPs Floating Point Operations

FC Fully Connected

GP Global Pooling

SPoC Sum-Pooling of Convolutions

MAC Maximum Activations of Convolutions

GeM Generalized Mean

R-MAC Regional Maximum Activations of Convolutions

PCA Principal Component Analysis

mAP mean Average Precision

DL Deep Learning