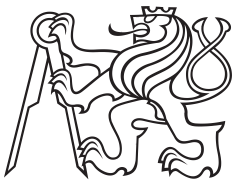
Bachelor Project



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Visual Landmark Recognition with Deep Learning

Ondřej Bouček

Supervisor: Giorgos Tolias, Ph.D
August 2020

Acknowledgements

I would like to thank my supervisor Giorgos Tolias, Ph.D for guiding me through the area of instance recognition. He was patient enough to explain difficult concept over and over again, he did not give up on me in difficult conditions under which this thesis was created. I would also like to thank my parents for supporting me during all my studies.

Declaration

I declare that the presented work was developed independently ant that I have listed all sources of the information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Ondřej Bouček, Prague, August 2020

Abstract

In this work we deal with instance recognition using deep learning. For extracting global descriptors we use neural network model trained with metric learning approach. Various modifications to k-NN classifiers to improve recognition quality were created. We also experiment with using multiple descriptors extracted from rescaled images. To simulate real world application we evaluate the model on created dataset referred to as Tini GLD. We achieved 0.84 Micro Average precision when using multiple descriptors.

Keywords: deep learning, neural networks, instance recognition, computer vision

Supervisor: Giorgos Tolias, Ph.D

Abstrakt

V této práci se zabýváne rozpoznávání instancí pomocí hlubokého učení. Získáváme deskriptory pomocí modelu neuronové sítě, který byl naučený přístupem *metric learning*. Vytvořili jsme různá upravení k-NN klasifikátorů pro vylepšení kvality rozpoznávání. Vyzkoušeli jsme použití více deskriptodů, získaných z různých změn velikostí obrazu. Abychom simulovali použití v reálném světě, k vyhodnocení přístupu používáme data, která jsme vytvořili pod názvem Tini GLD. Pomocí více deskiptorů jsme dosáhli 0,84 *Micro Average Precision*.

Klíčová slova: hluboké učení, neuronové sítě, rozpoznávání instance, počítačové vidění

Překlad názvu: Rozpoznávání orientačních bodů pomocí hlubokého učení

Contents

Figures

Tables

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Bouček Ondřej** Personal ID number: **474654**

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

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Branch of study: **Computer and Information Science**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Visual Landmark Recognition with Deep Learning

Bachelor's thesis title in Czech:

Rozpoznávání orientačních bodů pomocí hlubokého učení

Guidelines:

The goal of this project is to design and train a landmark recognition system with deep learning. Due to the large number of classes, the task is typically handled as an image retrieval problem. Each training image is represented by a high dimensional descriptor and landmarks are recognized through the labels of the closest training images in the descriptor space. The project will seek improvements to this pipeline by representing images with more than a single descriptor and by training pairwise verification models that can infer whether two images show the same landmark with higher certainty.
1. Study the literature on deep metric learning and landmark recognition with k-nearest neighbor classifiers.
2. Train a CNN to extract image descriptors in a metric learning fashion, using existing implementation.
3. Implement a knn classifier based on visual search that uses the learned descriptors.
4. Investigate ways to improve the performance of the search, therefore of the classifier, by employing multiple descriptors per image.
5. Design and implement deep pairwise verification model, i.e. to perform binary classification and predict whether two images show the same landmark or not. This will be used to improve the knn classifier.

Bibliography / sources:

[1] Radenovic Tolias Chum, PAMI 2019, Fine-tuning CNN Image Retrieval with No Human Annotation
[2] Noh Araujo Sim Weyand Han. ICCV2017, Large-Scale Image Retrieval with Attentive Deep Local Features

Name and workplace of bachelor's thesis supervisor:

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

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **10.01.2020** Deadline for bachelor thesis submission: **14.08.2020**

Assignment valid until: **30.09.2021**

Georgios Tolias, Ph.D.	doc. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor'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 bachelor's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

© ČVUT v Praze, Design: ČVUT v Praze, VIC

Chapter 1

Introduction

Computer vision (CV) is one of the most researched areas of artificial intelligence. Most of the work was done in the image classification area. The goal of classification task is to predict class label given a query image (Figure 1.1). Deep learning is wildly used for all CV tasks since the introduction of AlexNet [8] in 2012 which won image classification competition. Deep learning approach to classification is to train a neural network for the task by showing it example images of each class. The model learns features that distinguish classes.

Figure 1.1: Classification visualized. Given query image and defined set of **classes** {Dog, Cat, Parrot, Mouse, etc.} the goal is to predict class label.

Images inside classes have typically low intra-class variance which class based classification does not try to distinguish. However instance level recognition needs to distinguish between examples of single class. In Figure 1.1 image with parrot was labeled from a set of animals (defined classes). In Figure 1.2 we want to recognize the same image, however the set of classes are instances of various classes.

One of the obstacles in such task is low intra-class variation. Because two

Figure 1.2: Instance recognition. Given query image and a set of **instances** {"Hue" (dog), "Jack" (dog), "Ava" (parrot), "Jeffrey" (parrot), "Tom" (cat) and "Jerry" (mouse)} the goal is to predict which instance is seen in query image.

instances of same class are very similar it is difficult to learn features that differs instances. Moreover such features can differ in time, *i.e.*, some animals have winter and summer fur. Therefore instance recognition requires different approach than classification. Deep learning method for instance recognition is usually to train a model with *metric learning* approach. Training this way requires to show image depicting an instance. The model needs an example of image with the same and with different instance. Model learned with metric learning approach does not output class scores, it outputs vector in $\mathbf{R}^p$. This vectors are created in a way that are similar output to vectors nearby each other, different images are transformed to vectors further apart. Afterwards the instance is recognized using, *e.g.*, k-nearest neighbours classifiers.

In this thesis we focus on landmark recognition. Landmarks usually don't change in time and they differ. It does not directly have the obstacles we presented before. However instance recognition became the approach for landmark recognition. The first reason is that we want to recognize a lot of instances and classification does not perform well for huge amount of classes. The second reason, which is more important, is that there is usually very imbalanced dataset. Some landmarks are popular and another have only one image in the training set. Classification requires to show various images from each class and it does not deal well with imbalanced dataset. The third reason is because the landmark set can increase: some landmarks can be added to the database. If there is a metric learning trained model we can expect to be transformed new instance to a vector that is close to similar images. Classification approach requires fine-tuning entire model when added class. If a class with only one training image is added, fine-tuning is practically impossible. Landmark recognition is therefore approached as instance recognition task.

We follow up on the work of Radenovic *et al.* [13]. They trained a model with metric learning approach. In this thesis various modifications to k-nearest neighbours classifier are proposed. Every method is measured in well established metric known as Micro Average Precision. We created subset

of Google Landmarks Dataset [5] referred to as Tiny GLD to simulate real world conditions. The model we use didn't use any images from the Tini GLD during training.

Chapter 2

Related work

Image retrieval is wildly a researched topic since 1990s. In 2004 David G. Lowe published scale-invariant feature transform (SIFT) [14] and it became the state-of-the-art for the next decade. After 2012 and the work of Krizhevsky *et al.* [8], who won Large Scale Visual Recognition Challenge 2012 (ILSVRC, known as ImageNet [15]) with the AlexNet, research area shifted to deep learning and using convolution neural networks (CNN) for various computer vision tasks, image retrieval being one of them.

2.1 Graph Based Methods

One of the very first works was done by Crowly and Parrker (1984) [16]. In this work they proposed a graph based method for finding similarity between grayscale images, or any other two-dimensional shape. They used Difference of Low-Pass (DOLP) Transform to link peaks and ridges in a tree structure. Afterwards two trees could be matched with each other, finding similarity between the images.

In 1997 Wiskott *et al.* [17] created a system for recognizing human faces. The basic representation was a graph structure with labeled vertices with wavelet responses locally bundled in jets and weighted edges representing distances. They also proposed several comparison functions. They achieved state of the art results on ARPA/ARL FERET dataset created by US Army

Research Laboratory. It is worth mentioning that unlike modern datasets FERET consists of predefined lighting, poses and background of the images.

Shokoufande *et al* [1] (1999) proposed a saliency map graph (SMG). Using multi-scale wavelet transform [18] saliency map, which can be represented as SMG, is extracted. Region describing the object is called a scale-space cell (SSC) [19]. They also propose two graph matching algorithms for finding approximate topological and geometrical similarity. This approach achieved a high level of scale, translation and rotation invariance.

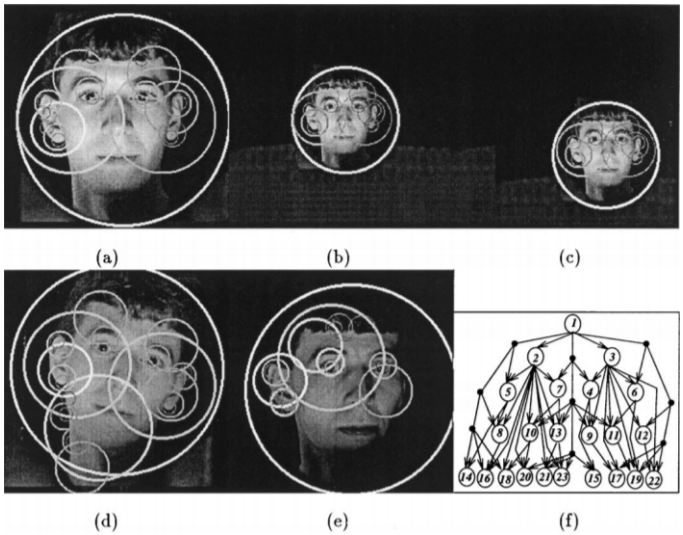


Figure 2.1: From [1]: Extracting the most salient SSCs in an image: (a) original image and its saliency map; (b) scale invariance; (c) translation invariance; (d) image rotation invariance; (c) invariance to rotation in depth (illuminated left side of face exhibits little change in its saliency map); and (f) the saliency map graph (SMG) of the original image in (a).

2.2 SIFT based methods

According to the instance retrieval overview by Zheng *et al.* [20] SIFT based methods are any methods that follow the pipeline: 1) *Local feature extraction*, 2) *Codebook training*, 3) *Feature encoding*

2.3 Local Feature Extraction

In the first part local descriptors are extracted given a feature detector. Each feature is sometimes referred to as "visual word". For D detected features set of descriptors is $\{f_i\}_{i=1}^{D}, f_i \in \mathbb{R}^p$. Counter example is global feature extraction which aims to extract single vector describing entire image. Feature extraction consists of two steps: **keypoint detection** and **feature extraction**. There are methods like corner detectors [21, 22], (Scale Invariant Feature Transform) SWIFT [23], Speeded Up Robust Features (SURF) [24] and Oriented FAST and Rotated BRIEF (ORB) [24].

2.3.1 Codebook Training

SIFT based methods use offline codebook training. Given a set of descriptors $\{f_i\}_{i=1}^{D}, f_i \in \mathbb{R}^p$ we want to divide set into clusters. Visual words are usually clustered using *e.g.*, K-means. We partition descriptors into K clusters so we get codebook of size K.

2.3.2 Feature Encoding

The Goal of feature encoding is to transform multiple descriptors from an image to single vector, so that we can match images between each other. Most wildly used encodings are Bag of Words (BoW [25]), Fisher Vectors (FV) [26] and Vector for Locally Aggregated Descriptors (VLAD) [27]. One can get basic understanding about FV and VLAD from [28].

2.3.3 Methods

Corner Detectors
Although these methods are not only detecting corners (corner is referred to as area in image with strong gradient), they are based on Moravec corner detectors introduced in 1980 [21]. He tried to find corners in image using another image taken from slightly different position. Moravec corner detector was improved by Harris and Stephens (1988) [22]. More recent comparison

between Harris and Moravec corner detection can be found [29]. Harris corner detectors are still being researched and used [30], [31].

SWIFT

SIFT based methods are named after Scale Invariant Feature Transform introduced by Lowe [32] in 1999 and was later improved in one of the most influential papers in computer vision [23]. It has since became standard benchmark for nearly a decade. SIFT algorithm consists of several steps (from [23]):

1. *Scale-space extrema detection*: The first stage of computation searches over all scales and image locations. It is implemented efficiently by using a difference-of-Gaussian function to identify potential interest points that are invariant to scale and orientation.

2. *Keypoint localization*: At each candidate location, a detailed model is fit to determine location and scale. Keypoints are selected based on measures of their stability.

3. *Orientation assignment*: One or more orientations are assigned to each keypoint location based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing invariance to these transformations.

4. *Keypoint descriptor*: The local image gradients are measured at the selected scale in the region around each keypoint. These are transformed into a representation that allows for significant levels of local shape distortion and change in illumination.

Swift was improved in various works, *i.e.*, [33], [34]. It was shown to be superior in performance to SURF and ORB, however ORB was shown to be faster than SIFT [35]. SIFT and SURF are patented [36], ORB is free to use.

Bag of Words

BoW method was developed for natural language processing. However in 2003 Sivic and Zisserman [25] introduced BoW to image retrieval community and it has since became standard method. The basic idea is to create histogram from an image. We consider centres of clusters created during codebook training as bins and we create histogram of features belonging to specific bin. Afterwards two such histograms are comparable.

2.4 CNN Based Recognition

Since introduction of AlexNet [8], convolutional neural networks (CNN) became a wildly researched area in computer vision (CV).CNN showed results in other areas than computer vision, *i.e.*, speech recognition [37]. With image classification being the most dominant area of CV, various architectures were developed.

On ImageNet [38] dataset AlexNet achieved 16.4% top-5 error rate (ER). In 2013 Zeiler and Fergus [37] achieved 14.8% top-5 ER. They developed visualization of inner layers. Simonyan and Zisserman with VGGNet [9] achieved 6.8% top-5 ER. With 16 layers deep model they showed that *deeper architecture is better*. One year later GoogLeNet was introduced [39], achieving 6.67% top-5 ER. GoogLeNet consists of 22 layers. One of the most used architecture is the Residual Network (ResNet) [12]. In this work He *et al.* created multiple models with different depth (18, 34, 50, 101 and 152 layers). With its 152 layers they proved correct Simonyan's and Zisserman's *deeper is better* and achieved astounding 3.57% top-5 ER. ResNet performed better than human with 5.1% top-5 ER, according to [15]. With rapidly increasing performance on image classification CNN research expanded to other CV tasks.

Most of CNNs use convolution and fully connected (FC) layers (as seen of Fig **??**). One can imagine, in a very simplified view, that convolution layers extract features from previous activations, or input image, with keeping spatial awareness. FC layer than takes such information and transforms it into vector which should describe the image reasonably. This idea became the motivation for experiments.
Babenko *et. al* [2] showed that activations within the network provide high-level descriptor of an image. In their experiment they trained a model (shown at Fig 2.2) for classification on ImageNet [38] dataset and extracted descriptors, referred to as Neural codes, as inner layer activation. They evaluated retrieval performance on such obtained descriptors on other datasets (INRIA Holidays [40] , Oxford datasets [41]) and found that performance was relatively well generalized on other datasets. Moreover when training on same dataset as testing, results improved.

Similar approach was used by Razavian *et al.* [42] where they used last convolutional layer. Tollias *et al.* [43] proposed *Maximum Activation Convolution* (MAC) and *Regional MAC* (R-MAC) layer that replace FC layer.

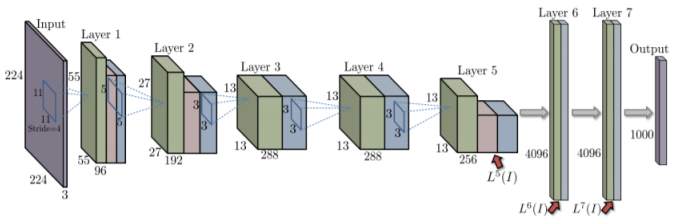


Figure 2.2: From [2]: Purple nodes correspond ti onput and output. Green units correspond to outputs of convolution layer, red units correspond to output of max pooling layer and blue units correspond to outputs of ReLU activation function. Layers 5,6 and 7 were used in their experiment.

Convolutional layer outputs 3D tensor $W \times H \; timesD$ where D refers to the number of output feature channels and W, D are spatial dimensions. 3D tensor can be seen as $X_{ii=1}^{D}$, where X_i is 2D tensor representing activations of i^{th} channel. MAC vector $\mathbf{f}$ is created as

$$\mathbf{f} = (f_1, \ldots, f_i, \ldots, f_D)^T, f_i = \max X_i \tag{2.1}$$

R-MAC layer divides input images into various regions and calculates maximum for each channel and region. They also proposed how to choose regions. With such layers there is no limit for dimension of input image, therefore no transformation is needed to resize input. Using such created descriptors they achieved state-of-the-art performance in image retrieval. R-MAC vectors are being further researched, *i.e.*, R-MAC+ [44]. Gordo *et al.* [45] showed that R-MAC is differentiable architecture therefore model can be trained *end-to-end*.

Chapter 3

Theory Background

3.1 Neural Network Basics

Neural Networks (NN) are computational systems inspired by the brain structure. Basic building block of any NN is called neuron. Neuron is an unit that inputs real valued vector and outputs a linear combination of the input vector and a bias. Neuron can be expressed as $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$ where $\mathbf{w}, b$ are parameters. Neurons are usually expressed with an *activation function* g. Therefore, neuron can be seen as $f(\mathbf{x}) = g(\mathbf{w}^T\mathbf{x} + b)$. Output of a neuron is called *activation*. Neuron can be seen seen in Figure 3.1.

NN is a computational graph created from neurons. Most common graphs

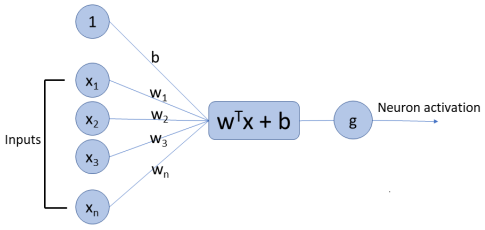


Figure 3.1: Neuron visualization

being *feed-forward neural network*. In such, a network neurons are arranged into layers, with each layer getting inputs only from the previous layer. Last layer is called *output layer*. Every other in between input and output is called *hidden layer*. Simple feed forward NN with one hidden layer can be seen at Figure 3.2. There are other types of NN like Recurrent NN [46], Modular NN [47] and many more. However for CV tasks feed forward architectures are

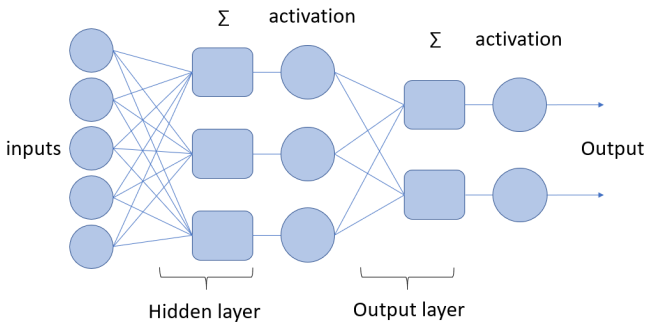


Figure 3.2: NN architecture with one hidden layer

the most wildly used.

3.1.1 Layers

Now that we established the basic structure of feed-forward neural networks, we give descriptions of the most wildly used layers.

Fully Connected Layer
Also *dense* or *linear* layer. Fully connected (FC) layer is presented in Figure 3.2. Every neuron activation from layer n is consumed as the input of every neuron in layer $n + 1$. FC layer takes a vector $\mathbf{x}$ as the input and outputs $W\mathbf{x} + \mathbf{b}$. Suppose layer n has k_n neurons.

Convolutional Layer

Unlike FC layer, Convolutional layer uses multidimensional tensor as an input ,*i.e.*, image with shape $(w \times h \times d)$. Convolution kernel with size $k \times k$ slides across input and for every position it produces an output. We can calculate every output as FC FC layer applied to inputs from receptive field and apply this for every possible position. We can imagine that kernel is able to detect local features in a small area. Moreover, one convolution layer can extract more features, so convolution layer transforms the input as $w \times h \times d_1 \rightarrow w \times h \times d_2$ while extracting d_2 features. There are more parameters to convolution layer: *padding* expands size of input with zeros so convolution does not reduce the size, *stride* is the number of pixels shifts over input matrix.

Pooling Layer

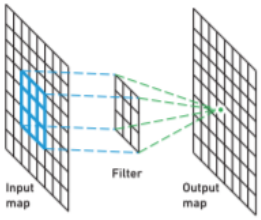


Figure 3.3: From [6]: Receptive field of convolution

Pooling layer is used to reduce the dimensions of data while keeping most of the information. Input data is split into cells of size $n \times n$ and outputs only one value for each cell. The most common versions, max pooling and average pooling can be seen in Figure 3.4. Other pooling methods like Generalized-mean pooling [13] were proposed. **Other layers**

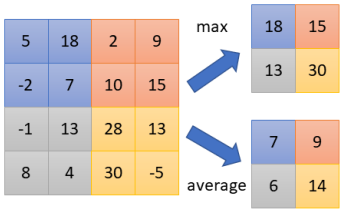


Figure 3.4: Example of max and average pooling

More types of layers exists for different purposes.
Dropout layer [48] aims to prevent over-fitting. During training it sets activations to zero with some probability, therefore next layer is not able to train on this activation, thus it needs to learn its goal on other patterns.
Batch Normalization layer [49] aims to speed up the training process.

3.1.2 Activation functions

Activation functions are what makes NN such a powerful tool. Consider model architecture from Fig 3.2. Without activation function (or with linear activation function $g(x) = \alpha x$) we can express pass through the model as $W_2(W_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2$. No matter how many hidden layers would be added the model, it would still be only a linear transformation. Activation functions is what enables NN to approximate arbitrary function [50] an can be learned. Any differentiable non-linear function can be used as activation function.

Tanh, Sigmoid

One of the very first activation functions were tanh and sigmoid (seen on Fig

3.6). Downsides of both functions are that $exp(x)$ is expensive to compute and both functions lead to *wanishing gradient problem* and are rarely used any more.

$$\begin{aligned} \sigma(x) &= \frac{1}{1+e^{-x}} \\ tanh(x) &= \frac{2x}{1+e^{-2x}} + 1 \end{aligned} \tag{3.1}$$

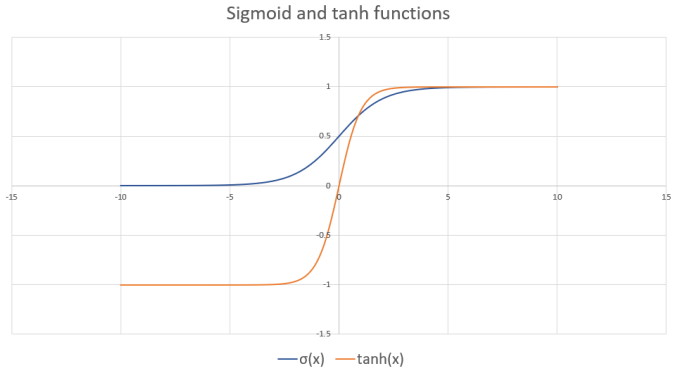


Figure 3.5: Tanh and sigmoid function

ReLU

Rectified Linear Unit (ReLU), proposed in [8], is the default activation function in most architectures.

$$ReLU(x) = max(0, x) \tag{3.2}$$

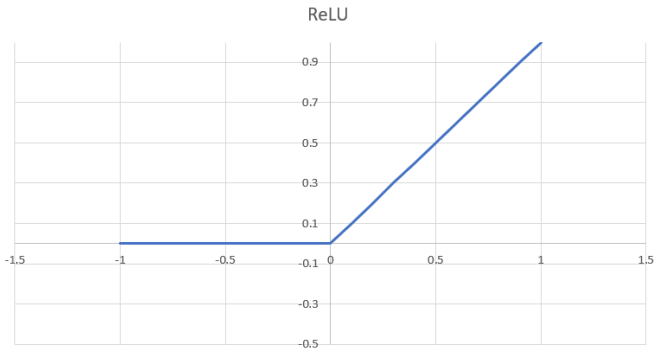


Figure 3.6: Tanh and sigmoid function

It is much simpler to compute than tanh or sigmoid and ReLU is less likely to have vanishing gradient. However ReLU can create *dying ReLU problem*

which happens when most of layer neurons output negative numbers, therefore the ReLU outputs zeros and information is lost. Models with ReLU tend to converge faster than with tanh or sigmoid. There exists a lot of modifications: Leaky ReLU $lReLU(x) = max(\alpha x, x)$, PReLU and many more. Overview can be found [51].

Softmax

Softmax function, defined as

$$S(\mathbf{x}) = \frac{1}{\sum_{i=0}^{n} e^{x_i}} \begin{pmatrix} e^{x_1} \\ e^{x_2} \\ \cdots \\ e^{x_n} \end{pmatrix} \tag{3.3}$$

is vector function. Its outputs a vector that sum to one. It is usually used as an activation function of the output layer to classification task and values in the vector are interpreted as class scores.

3.1.3 Training the NN

Now that we can set up architecture with layers and activation function, we need to train the model. Training NN is optimization task with millions of parameters. For example AlexNet [8] has 60,954,656 trainable parameters.

Loss

Loss is the objective function of optimization task. When training NN we must choose one of possible loss functions. If wrong one is chosen, there is little to no hope for any good results.

In classification tasks we want such a model that given input data the model outputs correct probability distribution. One of the most used loss function for one hot encoding distribution is Negative Log Likelihood (NLL).

$$L(\mathbf{y}) = -log(y_i) \tag{3.4}$$

assuming i is a correctly predicted class. If we try to minimize NLL loss, we also maximize our confidence in the correct prediction. Note that if $y_i = 1, L(\mathbf{y}) = 0$.

Another example is Hinge loss:

$$L(\mathbf{y}) = max(0, 1 - \mathbf{y}^T \mathbf{y}_c) \tag{3.5}$$

assuming $\mathbf{y}_c$ is correct distribution. Shape is seen in Figure 3.7.

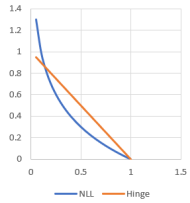


Figure 3.7: NLL and hinge loss

Deep metric learning is important approach for image retrieval and instance recognition with deep learning.
Contrastive loss [52] and Triplet loss [53] are the most frequently used. Triplet loss is defined as:

$$L(A, P, N) = max(0, m + D(A, P) - D(A, N)) \tag{3.6}$$

where m is margin, D is distance function, A is an anchor, P is a positive and N a negative, meaning that A, P is the same class and A, N is different class. Note that loss is 0 when negative data are further away than margin m from positive data. Other loss functions for metric learning like Quadruplet loss [54] have been proposed.

Depending on tasks, we must choose the loss function. There are Mean Square Error(MSE or l_2 error), Mean Absolute Error (MAE or l_1 error) for regression. Allignment Error Rate is used in natural language processing.

Gradient Descent

Gradient descent is optimization algorithm for finding local minima of *objective function*. The goal is to find $\widetilde{\theta}$ so that

$$\widetilde{\theta} = \arg\min_{\theta} \sum_{x \in T^m} L(h(x)) \tag{3.7}$$

where h is NN and T^m is the entire training set.
Single step of basic gradient descent algorithm is

$$\theta_{t+1} = \theta_t - \alpha \nabla f(\theta_t) \tag{3.8}$$

where α is the *step size* and f is the objective function. This process is repeated until algorithm converges. Various versions of gradient descent are studied in mathematical area known as Convex Optimization. Overview of used gradient descents can be seen [55].

Backpropagation

We need to calculate $\nabla f(\theta)$ for every step of gradient descent. Calculating the gradient is known as *backpropagation*. This is why we require to use differentiable activation functions, to calculate partial derivative for every parameter. We can do it by applying using chain rule:

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial x} \tag{3.9}$$

After backpropagation we have partial derivative of every parameter w.r.t. loss. This is repeated for every step of gradient descent.

3.1.4 Additional Techniques

We only scratched surface for reader to know basic idea of learning NN. There goes much more to the process. One must prevent *overfitting*, what is achievable by monitoring training and splitting data into *training* and *validation* sets, using dropout [48] etc. Adjusting learning rate during training was shown to be useful [9]. When using *stochastic gradient descent* one must create *mini-batches*. *Fine-tuning* is wildly used method, where trained model is taken and only last layers are retrained for another task.

3.1.5 Deep Metric Learning

Metric learning is approach where the goal is to find similarity between different inputs. Output of NN is vector in $\mathbb{R}^p$ with defined distance function (Euclidean, Mahalanobis). For such learning *siamese architecture* [56] is used. Such architecture consists of multiple networks with shared weights, needs distinct inputs and for every input calculate output. Metric learning aims to reduce the distance between similar object while increasing the distance between dissimilar objects in the output space. Basic idea is shown on Fig 3.8.

The model is trained on tuple inputs at the same time. In the dataset there

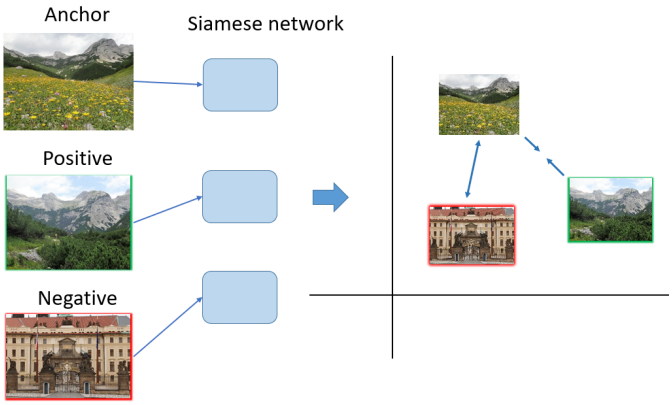


Figure 3.8: Deep metric learning visualized. Training process shifts similar images closer and negative farther

are usually a lot of tuples that yield loss 0. With loss 0 the network is not training and it takes longer to train. Therefore when creating mini-batch for training, we must generate tuples in better way than picking random images. This process known as *data mining*. Wu *et.al.* [57] showed the importance of data mining.

Data Mining

There are multiple approaches of how to generate training tuples.

1. *Random* mining: Our algorithm finds random, positive or negative, image pair. However many combinations yield loss 0. If the loss is 0, the model is unable to train. Therefore it is not used.
2. *Hard negative* mining: Given anchor image we need to find negative image, which generates highest loss. Higher loss increases step size of our training which might speed up the process. However, big step size may cause the training to diverge and not be able to find reasonable parameters.
3. *Hard positive* mining: As expected, hard positive mining stands for finding positive pair to anchor that maximizes loss. This can be hard to achieve. Consider anchor with class c, we have n images to choose from and in entire dataset with size m, there are k images with class c. Suppose we can choose from entire dataset ($m = n$), than we have $(k-1)/n$ options. However if we choose from mini-batch $n \ll m$ than we expect to choose from $(k-1)n/m^2$ possibilities.
4. *Semi-hard* (positive or negative) mining is approach where we generate data that are "good enough". There are multiple options of how to choose such pairs therefore it is most wildly used and researched approach.

Two approaches to generate data are *offline* and *online* mining. Comparison of botch approaches has been done [58]. More types of mining are being proposed, *i.e.*, *easy positive mining* [59].

Offline Mining approach preprocesses data before training. We create training tuples at the start of training and use them during entire process. Advantage of such approach is because we have more data to select from. Main disadvantage is that tuples are not updated to changing parameters of the network.

Online Mining approach is to generate tuples during training. In each training step we choose tuples from mini-batch. It allow us to create training samples that helps network train faster, however we have smaller set from which to choose.

3.1.6 PCA-whitening

PCA-whitening is preprocessing step for various machine learning tasks. Since [60] it is wildly used for feature decorrelation. It is two step process.
Firstly we perform *Principal Component Analysis* (PCA): algorithm used for dimensionality reduction. Given data set $X = (\mathbf{x}_0, \ldots, \mathbf{x}_m)$ we estimate covariance matrix:

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}^{(i)} - \mu)(\mathbf{x}^{(i)} - \mu)^T = XX^T \tag{3.10}$$

where μ is the mean of the data:

$$\mu = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}^{(i)} \tag{3.11}$$

Next we create matrix U containing eigenvectors of Σ. It can be done by , *e.g.*, SVD. PCA is the projection on eigenvectors corresponding to the biggest eigenvalues (when used for dimensionality reduction). When projected on every eigenvector resulting data has diagonal covariance matrix: data becomes decorrelated.

$$X_{PCA} = U^T X \tag{3.12}$$

Second step is *Whitening*. We want each feature to have unit variance. X_PCA has diagonal covariance, therefore we need to rescale each vector as:

$$\mathbf{x}_{PCA-whitened,i} = \frac{\mathbf{x}_{PCA,i}}{\sqrt{\lambda_i}} \tag{3.13}$$

, where λ_i is corresponding variance.

3.2 Popular Architectures

3.2.1 AlexNet

AlexNet [8] was created in 2012. It was the first architecture that achieved major success in image classification. AlexNet consists of 5 convolutional layers and 3 FC layers. They used 11×11 kernel size for the first layer which requires a lot of parameters: AlexNet has 60 million. They also shown that the use of ReLU as the activation performs better than tanh or sigmoid. AlexNet is seen in Figure 3.9 AlexNet was trained in over 5 days on two GTX

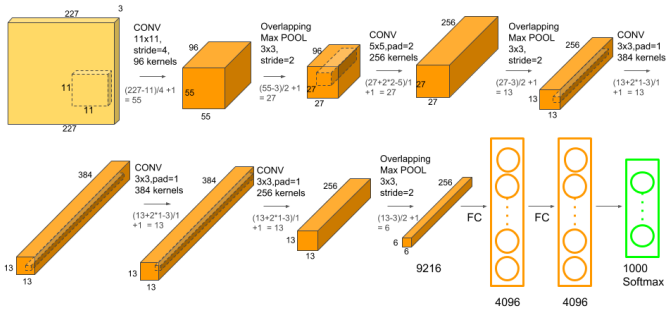


Figure 3.9: From [10] AlexNet architecture.

580 3GB GPUs. They also used dropout and data augmentation techniques.

3.2.2 VGGNet

Two years after introduction of AlexNet, Zisserman and Simonyan created VGGNet [9]. Unlike AlexNet with 11×11 kernel size, VGGNet used multiple layers with kernel only 3×3. They show that, *e.g.*, three convolutional layers with kernel 3×3 outperforms one layer with kernel 7×7 in both discriminative power and number of trainable parameters. VGGNet is seen in Figure 3.10 Although VGGNet has 138 million parameters, training was done in fewer epochs than AlexNet due to smaller kernel sizes and pre-initialization of layers.

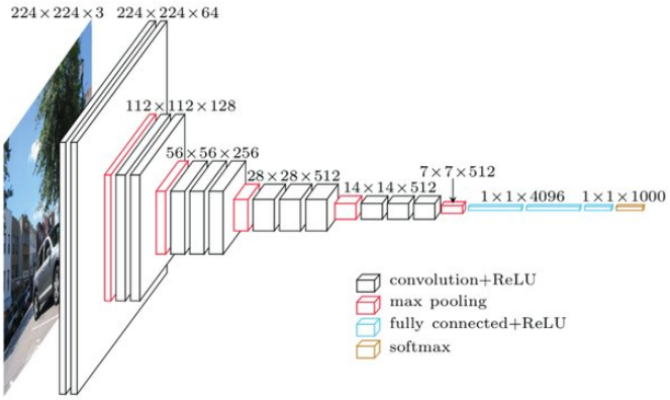


Figure 3.10: From [11] VGGNet architecture.

3.2.3 ResNet

Residual Network (ResNet) [12] was the first model that outperformed human in ImageNet classification. Their main contribution is the usage of identity shortcuts(*residual connections*). The architecture is separated into building blocks and there is identity connection over the block. It is seen in Figure 3.11 Stacking such blocks deals well with *vanishing gradient* therefore they

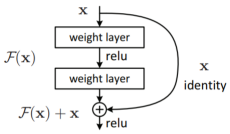


Figure 3.11: From [12] ResNet building block.

were able to build up to 152 layers deep model without vanishing gradient problem.

Chapter 4

An Approach to Instance-level Recognition

Instance recognition deals with data changing over time, *i.e.*, people at a workplace. This is challenging because the models need to recognize instances that were not seen during training. In this thesis we simulate such setting by testing a model on dataset which the model was not trained on. For this purpose Tini GLD dataset was createt as a subset of Google Landmarks Dataset (GLD) [61]. GLD poses a challenge for instance recognition and image retrieval task because of its enormous size and for its imbalanced representations of classes.
We expand on the work of Tolias *et al.* [13]. They trained model with metric learning approach on Paris6k [62] and Oxford5k [41] datasets. We use their model *gl18-tl-resnet101-gem-w* for extracting global descriptors from an image. With such models we can use k-Nearest Neighbours (k-NN) classifiers to recognize instances. Quality of method is measured in established Micro Average Precision (μAP), also known as Global Average Precision (GAP). Calculating μAP requires confidence score in addition to classification. We experiment with various methods of how to get confidence score.
In the first part we experiment k-NN classifiers with non-linear similarity transformation. Mean k-NN and α-Expanded Mean k-NN are proposed with hope of dealing with imbalanced dataset. In the second part we rescale every image to get multiple descriptors. With more descriptors we experiment with finding distance from *similarity matrix* and *multi-scale aggregation*.

Micro Average Precision

µAP is measurement where ranking of predictions is also important. µAP is calculated in two steps: Firstly sort predictions in descending order by

confidence. Secondly calculate

$$\mu AP = \frac{1}{m} \sum_{i=1}^{m} P(i) rel(i) \tag{4.1}$$

where $P(i)$ is precision at i

$$P(i) = \frac{k}{i} \tag{4.2}$$

where k is number of correctly predicted from top i queries and $rel(i)$ is 1 if query i is predicted correctly, 0 otherwise.
When we want to maximize µAP we need to score high confidence on correct predictions and low score on incorrect ones.

Tini-GLD

Because of computational limits Tini-GLD was created from Google Landmarks Dataset [61] for testing. It consists of 10000 training images from 6090 classes. Most of the classes have only one image in the training set and only a few classes have more than 10 images. Distribution of Tini-GLD is seen at Figure 4.1. Test set is 1000 images where 135 are from 51 unique classes and other 865 images do not have corresponding class in training set. One must remember that used models were trained on different datasets and have never seen images from Tini-GLD.

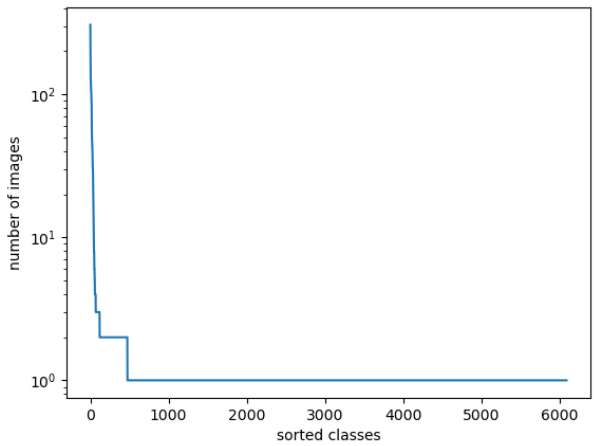


Figure 4.1: Tini-GLD train set distribution. Maximum training samples for a class is 606 and 5616 classes have only one training image.

Classical Approach

In the fist part we evaluate simple approaches. For classification we use variations of k-Nearest Neighbours (k-NN) classifiers. Basic k-NN classifier works as follow:

Firstly we need to create set of k-nearest neighbours. We sort our training set in ascending order by euclidean distance from query vector $\mathbf{q}$:

$$(\mathbf{x}_1, \ldots, \mathbf{x}_m), i < j \rightarrow ||\mathbf{q} - \mathbf{x}_i|| <= ||\mathbf{q} - \mathbf{x}_j|| \tag{4.3}$$

Than we take the first (closest) k vectors from the set, we denote this set as $k(\mathbf{q})$. Secondly classify $\mathbf{q}$ as the mode of the neighbours classes. Classification of query q images is done as follows:

$$\widetilde{c} = \arg\max_{c \in C} |\{\mathbf{x} \,|\, c_x = c,\, \mathbf{x} \in k(\mathbf{q})\}| \tag{4.4}$$

where C is set of classes and c_x is class of the vector $\mathbf{x}$. Such classifier is seen in Figure 4.2

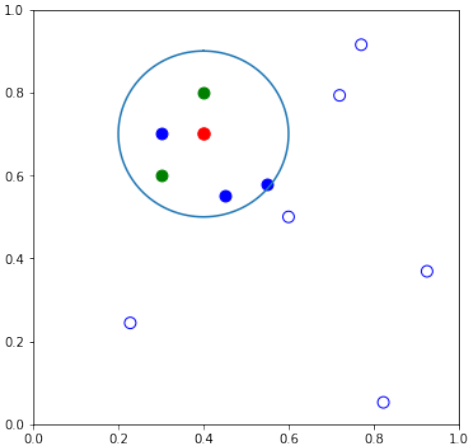


Figure 4.2: Simple 5-NN: There are two classes (blue, green) and query (red). Filled points are only considered in 5-NN. In this case query is classified as green.

However this classification gives us limited possibilities for calculating scores which we need for evaluating μAP. Therefore we use cosine similarity:

$$d(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x}^T\mathbf{y}}{||\mathbf{x}||||\mathbf{y}||} \tag{4.5}$$

We classify as

$$\widetilde{c} = \arg\max_{c \in C} \sum_{\substack{c_x = c \\ \mathbf{x} \in k(\mathbf{q})}} f(d(\mathbf{q},\mathbf{x})) \tag{4.6}$$

where f is distance non-linear transformation. This transformations are used because dataset is highly imbalanced so we want to make the classifier more *top-heavy*. We use the value of maximum as confidence score.

Second variation of k-NN classifier is *Mean k-NN* and its generalization *α-Expanded Mean k-NN*. Mean k-NN is another attempt to deal with long-tailed dataset. Here we calculate class-wise mean from k-nearest neighbours.

We reduce any number of vectors from class represented in $k(\mathbf{q})$ into its mean μ_c:

$$\mu_c = \frac{1}{n_c} \sum_{\substack{c_x = c \\ \mathbf{x} \in k(\mathbf{q})}} \mathbf{x} \tag{4.7}$$

where n_c is number of vectors in $k(\mathbf{q})$ from class c. This approach is seen in Figure 4.3. Classification is done as:

$$\widetilde{c} = \arg\max_{c \in C} f(d(\mathbf{q}, \mu_c)) \tag{4.8}$$

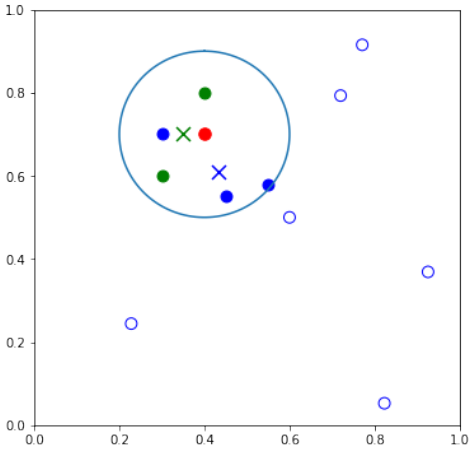


Figure 4.3: Mean 5-NN: In addition to before, crosses denote mean of considered classes. For this classification we calculate only distance to query and means. In this case classifier predicts green class.

α-Expanded Mean k-NN is generalization of Mean k-NN, It is inspired by query expansion (QE) technique used in image retrieval. We expect that training points from highly represented class are similar to each other. Therefore the idea is to take all training vectors that are similar to vectors in the $k(\mathbf{q})$ and classify using class-wise mean of this bigger set. This is seen in Figure 4.4. Here we calculate class-wise mean as:

$$\mu_c = \mu_c = \frac{1}{|M_c|} \sum_{\substack{c_x = c \\ \mathbf{x} \in M_c)}} \mathbf{x} \tag{4.9}$$

where M_c is set of training vectors from class c that are closer to some vector from $k(\mathbf{q})$ from class c

$$M_c = \{y \,|\, \exists \mathbf{x} \in k(\mathbf{q}), ||\mathbf{x} - \mathbf{y}|| < \alpha, c_y = c_x\} \tag{4.10}$$

Classification is performed as in equation 4.3.

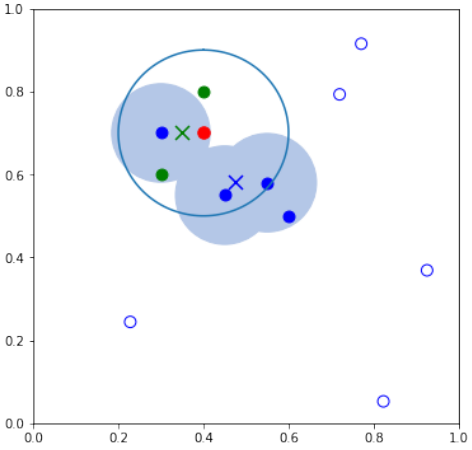


Figure 4.4: α-Expanded mean 5-NN: One blue vector is close to vector in 5-NN. Therefore it is taken into the set of which mean for classification is calculated.

Distance from Similarity Matrix

We rescale every image x five times. For every scale s_i we extract descriptor $\overline{x}_s$. Afterwards we perform PCA-whitening by each scale differently. Each image is than represented as matrix:

$$X = \begin{pmatrix} \overline{x}_{s1} \\ \dots \\ \overline{x}_{sn} \end{pmatrix} \tag{4.11}$$

Afterwards we calculate distance matrix between two images as:

$$S = XY^T \tag{4.12}$$

Which is $n \times n$ matrix. Similarity of images x, y is calculated in various ways:

1. Max: Taking maximum of the matrix

$$d(S) = \max_{i,j} s_{i,j} \tag{4.13}$$

2. Sum of max rows: Get vector row maximums and sum:

$$d(S) = \sum_i \max_j s_{i,j} \tag{4.14}$$

3. Sum of max cols: Get vector of column maximums and sum:

$$d(S) = \sum_j \max_i s_{i,j} \tag{4.15}$$

4. Squared of max rows(cols): Repeat 2 and 3 with squared distance

$$\begin{aligned} d(S) &= \sum_i (\max_j s_{i,j})^2 \\ d(S) &= \sum_j (\max_i s_{i,j})^2 \end{aligned} \tag{4.16}$$

We denote X as descriptor matrix from test set and Y from training set therefore, *e.g.*, we must differ sum of max cols and sum of max rows. With these distance metrics we use k-NN classifier without non-linear function.

Multi-Scale Aggregation

As in before, we extract multiple descriptors corresponding to scale s to get set of descriptors $\{x_s i\}_{i=1}^n$ Afterwards we create single vector

$$\mathbf{x}_w = \sum_{i=1}^{n} w_i \mathbf{x}_{si} \tag{4.17}$$

where $w_i \in \{0, 1\}$. Afterwards we perform PCA-whitening and evaluate with classifiers proposed in the first part.

Chapter 5

Results

We experimented with $k \in \{1, 3, 7, 13, 17, 23\}$. For $k = 1$ (top-1 classifier) non-linear transformations do not have any effect. For 1-NN classifier the non-linear function does not have any effect. This classifier performed poorly achieving only μAP of 0.65. Applying PCA-whitening for top-1 classifier performance even worsened to $\mu\text{AP} = 0.59$. An example of the most similar images for a query is seen in Figure 5.1.

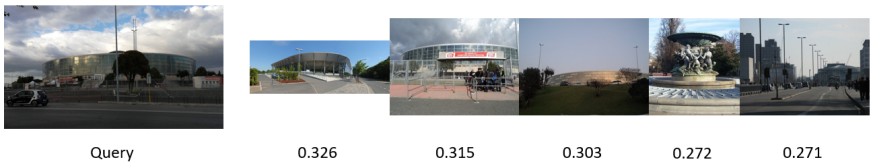


Figure 5.1: 5 images of nearest neighbour from a the query image and their similarity.

5.1 Distance Function Variation

Distance modification functions were chosen as $f(x) \in \{x, x^2, \sqrt{x}, ln(x + e)\}$. Comparison of classifiers with distance modification is seen in Figure 5.2. Best μAP was achieved for $k = 3$, without PCA-whitening and using no distance transformation (identity function). Overall best results used $k = 7$ with PCA-whitening and identity function, μAP of such combination is 0.81. With increasing k performance of all classifiers with non-whitened data worsened, however PCA-whitening seems to slightly increase. Exact values are in Table 5.1.

PCA-w	method	1	3	7	13	17	23
True	sqrt	0.5893	0.7792	0.7913	0.7870	0.7712	0.7536
	pow 2	0.5893	0.7308	0.7760	0.7881	0.7846	0.7847
	identity	0.5893	0.7726	**0.8106**	0.8100	0.8054	0.7966
	$\ln(x+e)$	0.5893	0.7791	0.7916	0.7862	0.7687	0.7455
False	sqrt	0.6493	0.8026	0.7432	0.6697	0.6428	0.5996
	pow 2	0.6493	0.7950	0.7687	0.7182	0.6859	0.6660
	identity	0.6493	0.8058	0.7522	0.6853	0.6639	0.6286
	$\ln(x+e)$	0.6493	0.8027	0.7389	0.6583	0.6224	0.5760

Table 5.1: Performance of descriptor obtained from the original image.

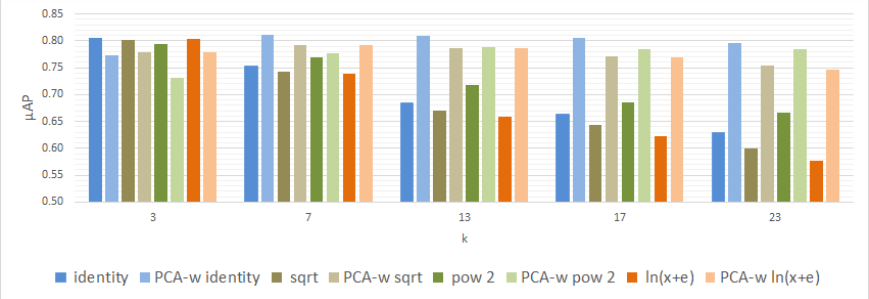


Figure 5.2: Comparison of combinations with distance function and PCA-whitning.

5.2 α-Expanded Mean k-NN

α-Expanded Mean k-NN was outdone by previous methods. This approach only worsened performance. PCA-whitening increase final result up to 0.75 with $k = 7$. Comparison of α-Expanded Mean k-NN is seen in Figure 5.3, exact values are in Table 5.2.

Figure 5.3: α-Expanded Mean k-NN both on non-whitened and whitened data

PCA-w	α	3	7	13	17	23
TRUE	Mean k-NN	0.7415	**0.7524**	0.7487	0.7455	0.7218
	0.7	0.7383	0.7496	0.7465	0.7432	0.716
	0.6	0.7286	0.7316	0.7362	0.7306	0.7168
	0.5	0.7151	0.7163	0.7262	0.7294	0.7138
	0.4	0.7066	0.7061	0.704	0.6978	0.6951
	0.3	0.6875	0.6802	0.6775	0.6779	0.6827
	0.2	0.6535	0.6374	0.6261	0.6298	0.6279
FALSE	Mean k-NN	0.6946	0.7284	0.7356	0.7357	0.7333
	0.7	0.6932	0.728	0.7353	0.7336	0.732
	0.6	0.6889	0.725	0.7331	0.7313	0.7301
	0.5	0.6872	0.7225	0.7316	0.7302	0.7276
	0.4	0.6869	0.7201	0.7287	0.7283	0.7244
	0.3	0.6753	0.7037	0.7145	0.7148	0.7115
	0.2	0.6624	0.6836	0.6808	0.6769	0.6707

Table 5.2: α-Expanded Mean k-NN results.

5.3 Similarity Matrix Based Distance

For calculating similarity matrix and afterwards multi-scale aggregation, rescaling parameters $(0.25, 0.5, \sqrt{0.25}, 1, \sqrt{2})$ were chosen. PCA-whitening has the biggest effect in this method. The highest μAP was achieved for PCA-whitening, sum of max of rows and $k = 7$. Comparison is seen in Figure 5.5 and exact values are seen in Table 5.5. This approach requires to rescale every input image 5 times, feed every scale through the network and calculate

PCA-whitening scale-wise. The computation is therefore expensive, however it shows good results. Examples of similarity matrix between images from Figure 5.4 are seen in Tables 5.3,5.4.

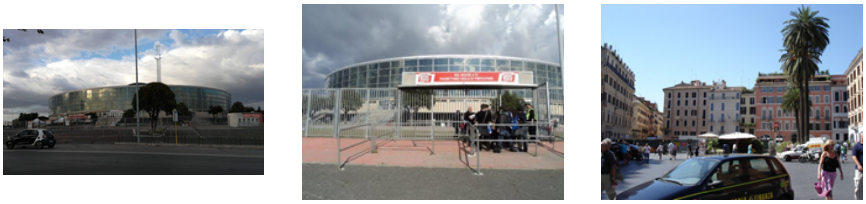


Figure 5.4: Images A, B, C (from left) from which similarity matrix are computed.

		Image B				
		s_1	s_2	s_3	s_4	s_5
Image A	s_1	0.144	0.129	0.095	0.075	0.071
	s_2	0.196	0.229	0.227	0.202	0.187
	s_3	0.226	0.27	0.287	0.267	0.257
	s_4	0.23	0.285	0.33	0.315	0.316
	s_5	0.237	0.279	0.33	0.335	0.348

Table 5.3: Similarity matrix between images A and B

		Image B				
		s_1	s_2	s_3	s_4	s_5
Image A	s_1	-0.003	-0.012	0.041	0.064	0.053
	s_2	0.036	0.017	0.035	0.057	0.056
	s_3	0.012	0.037	0.033	0.057	0.060
	s_4	0.04	0.059	0.055	0.094	0.091
	s_5	0.041	0.064	0.052	0.093	0.095

Table 5.4: Similarity matrix between images A and C

Figure 5.5: Comparison of similarity matrix based methods.

5.4 Multi-scale Aggregation

Multi-scale aggregation revealed that upscaling images by a factor of $\sqrt{2}$ improves over originals. Upsampling achieved μAP of 0.83 for PCA-whitening, no distance transformation and $k = 13$. Out of every aggregation combination the best result yielded summing descriptors corresponding to original images and upscaling by a factor of $\sqrt{2}$. This aggregation is seen in Figure 5.7, exact values are in Table 5.6. With PCA-whitening, both identity and square root functions and $k = 13, k = 7$ respectively, μAP $= 0.84$. In Figure 5.6 there are the 5 most similar images to a query.

PCA-w	method	1	3	7	13	17	23
True	max	0.5088	0.7658	0.7869	0.7826	0.7714	0.7515
	sum of max rows	0.5686	0.7868	0.8270	0.8203	0.8080	0.7782
	sum of max cols	0.5696	0.7838	**0.8303**	0.8269	0.8097	0.7881
	sum of squared max rows	0.5729	0.7294	0.7787	0.7892	0.7900	0.7803
	sum of squared max cols	0.5729	0.7311	0.7820	0.7887	0.7913	0.7845
False	max	0.5584	0.7308	0.7048	0.6447	0.6298	0.5907
	sum of max rows	0.5732	0.7712	0.7280	0.6657	0.6344	0.6123
	sum of max cols	0.5809	0.7544	0.7238	0.6675	0.6451	0.6127
	squared of max rows	0.5802	0.7650	0.7445	0.7050	0.6857	0.6499
	squared of max cols	0.5811	0.7490	0.7433	0.6857	0.6694	0.6580

Table 5.5: Results of similarity matrix approach.

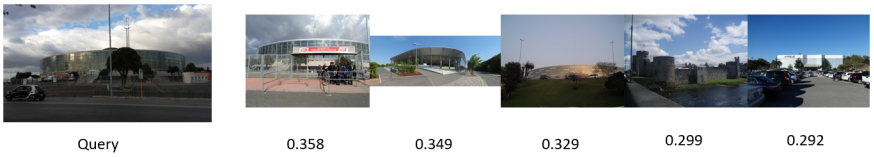


Figure 5.6: 5 nearest neighbours from a the query image and their similarity.

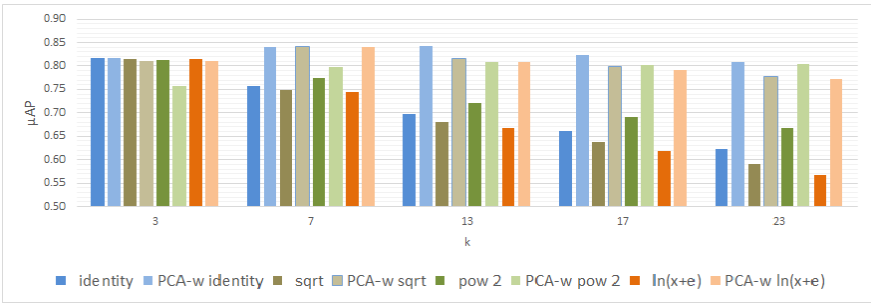


Figure 5.7: Comparison of methods for descriptor obtained from summing descriptor from original image and from upscaled image by factor of $\sqrt{2}$.

PCA-w	method	1	3	7	13	17	23
True	sqrt	0.6142	0.8113	0.8425	0.8156	0.7990	0.7783
	pow 2	0.6142	0.7583	0.7984	0.8094	0.8024	0.8032
	identity	0.6142	0.8168	0.8399	**0.8428**	0.8243	0.8088
	$\ln(x+e)$	0.6142	0.8115	0.8413	0.8094	0.7917	0.7716
False	sqrt	0.6582	0.8153	0.7493	0.6794	0.6384	0.5902
	pow 2	0.6582	0.8117	0.7736	0.7212	0.6915	0.6673
	identity	0.6582	0.8177	0.7571	0.6982	0.6611	0.6226
	$\ln(x+e)$	0.6582	0.8154	0.7453	0.6675	0.6181	0.5679

Table 5.6: Multi scale aggregation. Summation of descriptor from original image and image upscaled by a factor of $\sqrt{2}$ yielded the overall best result.

Chapter 6

Conclusion

In this thesis we explored different possibilities of how to use k-NN classifiers for instance recognition. We used descriptors obtained from a neural network trained by metric learning approach. Evaluation was done on specifically created dataset which the neural network did not use during training. The dataset was created to be highly imbalanced. We proposed and evaluated various methods of how to deal with such dataset.

In the first part we experimented with single descriptor obtained from original image. Firstly we found that non-linear similarity transformations do not improve performance. The best results with original descriptors are for small k, *i.e.*, 3. For 3-NN classifier we obtained μAP 0.8.
We found that in our small dataset small k perform and with larger k the performance worsens drastically. However using PCA-whitening seems to negate this effect and performance on PCA-whitened data does not decrease with increasing k.
Secondly we used α-Expanded Mean k-NN which yielded disappointment results. Moreover α-Expansion lowers μAP even more. Moreover calculating α-Expansion is computationally very expensive. This approach failed and it is not worth looking into the idea.

In the second part we extracted multiple descriptors from each image. We rescaled every image five times and from each scale we extracted descriptor. Firstly we considered calculating similarity from the *similarity matrix*. This

approach requires to calculate similarity between every two descriptors. In our case it is 25 calculations for every two images. We extracted the similarity value in various ways from this matrix. When used PCA-whitening it was calculated scale-wise which is calculate PCA-whitening 5 times. This made the approach the most expensive for calculating. However it outperformed the basic approach. The best μAP was achieved 0.83.
Another method using multiple descriptors is to aggregate them into one. We experimented with simple summing various scaled descriptors and with this single descriptor evaluate the method as in the first part. This approach achieved the best results of μAP 0.84.

We propose three ideas for future research.
Firstly we know that the similarity matrix approach contains a lot of information. In our case we had 25 similarity values and we obtained the overall similarity value in 5 different ways. It is very likely that there exists better ways how to calculate overall similarity. This research might even consider training neural network model to predict similarity score given the matrix. However it would make the similarity matrix approach even more expensive.
Secondly when obtaining multi-scale descriptor we only summed descriptors from different scales. The best results were achieved when summing the descriptor from original image and the descriptor from image upscaled by the factor of $\sqrt{2}$. This approach yielded the best results with only binary weights. It will probably yield better μAP when we learn the weights of the linear combination.
The last direction one might consider is to improve how to obtain multiple descriptors from an image. We extracted a descriptor from every scale of an image. However it is possible to use any transformation on the input image. We might combine various affine transformations. Than it is possible to repeat similarity matrix methods and creating multi-scale descriptor, however it will probably not be called multi-scale descriptor.

Bibliography

[1] A. Shokoufandeh, I. Marsic, and S. Dickinson, "View-based object recognition using saliency maps," *Image and Vision Computing*, vol. 17, pp. 445–460, 04 1999.

[2] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, "Neural codes for image retrieval," in *European conference on computer vision*, pp. 584–599, Springer, 2014.

[3] J. Jeong, T. Yoon, and J. Park, "Towards a meaningful 3d map using a 3d lidar and a camera," *Sensors*, vol. 18, p. 2571, 08 2018.

[4] J. Jordan, "An overview of object detection: one-stage methods.."

[5] H. Noh, A. Araujo, J. Sim, and B. Han, "Image retrieval with deep local features and attention-based keypoints," *CoRR*, vol. abs/1612.06321, 2016.

[6] H. Yakura, S. Shinozaki, R. Nishimura, Y. Oyama, and J. Sakuma, "Malware analysis of imaged binary samples by convolutional neural network with attention mechanism," pp. 127–134, 03 2018.

[7] H. Kataoka, K. Iwata, and Y. Satoh, "Feature evaluation of deep convolutional neural networks for object recognition and detection," 09 2015.

[8] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Information Processing Systems*, vol. 25, 01 2012.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv 1409.1556*, 09 2014.

[10] S. Nayak, "Understanding alexnet," *https://www.learnopencv.com/understanding-alexnet/*.

[11] T. Bezdan and N. Bacanin, "Convolutional neural network layers and architectures," pp. 445–451, 01 2019.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[13] F. Radenovic, G. Tolias, and O. Chum, "Fine-tuning CNN image retrieval with no human annotation," *CoRR*, vol. abs/1711.02512, 2017.

[14] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, Nov 2004.

[15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, "Imagenet large scale visual recognition challenge," *CoRR*, vol. abs/1409.0575, 2014.

[16] J. Crowley and A. Parker, "A representation for shape based on peaks and ridges in the difference of low-pass transform," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-6, pp. 156 – 170, 04 1984.

[17] L. Wiskott, N. Krüger, N. Kuiger, and C. von der Malsburg, "Face recognition by elastic bunch graph matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 775–779, 1997.

[18] E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger, "Shiftable multiscale transforms," *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 587–607, 1992.

[19] I. Marsic, "Data driven shifts of attention in wavelet scale space," *IEEE trans. PAMI submitted*, 1993.

[20] L. Zheng, Y. Yang, and Q. Tian, "Sift meets cnn: A decade survey of instance retrieval," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 5, pp. 1224–1244, 2017.

[21] H. P. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover.," tech. rep., Stanford Univ CA Dept of Computer Science, 1980.

[22] C. G. Harris, M. Stephens, *et al.*, "A combined corner and edge detector.," vol. 15, no. 50, pp. 10–5244, 1988.

[23] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[24] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision – ECCV 2006* (A. Leonardis, H. Bischof, and A. Pinz, eds.), (Berlin, Heidelberg), pp. 404–417, Springer Berlin Heidelberg, 2006.

[25] Sivic and Zisserman, "Video google: a text retrieval approach to object matching in videos," in *Proceedings Ninth IEEE International Conference on Computer Vision*, pp. 1470–1477 vol.2, 2003.

[26] J. Sánchez, T. Mensink, and J. Verbeek, "Image classification with the fisher vector: Theory and practice," *International Journal of Computer Vision*, vol. 105, 12 2013.

[27] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3304–3311, 2010.

[28] P. H.R.Tizhoosh, "Machine intelligence - lecture 5 (computer vision, features, fisher vector, vlad)," 2019.

[29] N. Dey, P. Nandi, N. Barman, D. Das, and S. Chakraborty, "A comparative study between moravec and harris corner detection of noisy images using adaptive wavelet thresholding technique," *arXiv preprint arXiv:1209.1558*, 2012.

[30] P. Hsiao, C. Lu, and L. Fu, "Multilayered image processing for multiscale harris corner detection in digital realization," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 5, pp. 1799–1805, 2010.

[31] J. . Ryu, C. . Lee, and H. . Park, "Formula for harris corner detector," *Electronics Letters*, vol. 47, no. 3, pp. 180–181, 2011.

[32] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2, pp. 1150–1157, Ieee, 1999.

[33] Yan Ke and R. Sukthankar, "Pca-sift: a more distinctive representation for local image descriptors," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 2, pp. II–II, 2004.

[34] R. Arandjelović and A. Zisserman, "Three things everyone should know to improve object retrieval," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2911–2918, 2012.

[35] E. Karami, S. Prasad, and M. S. Shehata, "Image matching using sift, surf, BRIEF and ORB: performance comparison for distorted images," *CoRR*, vol. abs/1710.02726, 2017.

[36] D. G. Lowe, "Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image," 2000.

[37] T. N. Sainath, A. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8614–8618, 2013.

[38] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[39] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014.

[40] H. Jegou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search", booktitle="computer vision – eccv 2008," (Berlin, Heidelberg), pp. 304–317, Springer Berlin Heidelberg, 2008.

[41] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2007.

[42] A. S. Razavian, J. Sullivan, S. Carlsson, and A. Maki, "Visual instance retrieval with deep convolutional networks," *ITE Transactions on Media Technology and Applications*, vol. 4, no. 3, pp. 251–258, 2016.

[43] G. Tolias, R. Sicre, and H. Jégou, "Particular object retrieval with integral max-pooling of cnn activations," *arXiv preprint arXiv:1511.05879*, 2015.

[44] F. Magliani and A. Prati, "An accurate retrieval through R-MAC+ descriptors for landmark recognition," *CoRR*, vol. abs/1806.08565, 2018.

[45] A. Gordo, J. Almazán, J. Revaud, and D. Larlus, "End-to-end learning of deep visual representations for image retrieval," *CoRR*, vol. abs/1610.07940, 2016.

[46] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[47] B. L. Happel and J. M. Murre, "Design and evolution of modular neural network architectures," *Neural networks*, vol. 7, no. 6-7, pp. 985–1004, 1994.

[48] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 06 2014.

[49] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[50] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural Networks*, vol. 6, no. 6, pp. 861 – 867, 1993.

[51] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *CoRR*, vol. abs/1505.00853, 2015.

[52] S. Chopra, R. Hadsell, and Y. Lecun, "Learning a similarity metric discriminatively, with application to face verification," vol. 1, pp. 539–546 vol. 1, 07 2005.

[53] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," *CoRR*, vol. abs/1404.4661, 2014.

[54] W. Chen, X. Chen, J. Zhang, and K. Huang, "Beyond triplet loss: a deep quadruplet network for person re-identification," *CoRR*, vol. abs/1704.01719, 2017.

[55] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016.

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

[57] C. Wu, R. Manmatha, A. J. Smola, and P. Krähenbühl, "Sampling matters in deep embedding learning," *CoRR*, vol. abs/1706.07567, 2017.

[58] M. Sikaroudi, B. Ghojogh, A. Safarpoor, F. Karray, M. Crowley, and H. Tizhoosh, "Offline versus online triplet mining based on extreme distances of histopathology patches," *arXiv preprint arXiv:2007.02200*, 2020.

[59] H. Xuan, A. Stylianou, and R. Pless, "Improved embeddings with easy positive triplet mining," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, March 2020.

[60] H. Jégou and O. Chum, "Negative evidences and co-occurences in image retrieval: The benefit of pca and whitening," in *Computer Vision – ECCV 2012* (A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, eds.), (Berlin, Heidelberg), pp. 774–787, Springer Berlin Heidelberg, 2012.

[61] H. Noh, A. Araujo, J. Sim, and B. Han, "Image retrieval with deep local features and attention-based keypoints," *CoRR*, vol. abs/1612.06321, 2016.

[62] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Lost in quantization: Improving particular object retrieval in large scale image databases," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2008.