Master of Science in Informatics at Grenoble
Master Informatique
Specialization Data Science

# Deep learning with Siamese networks for instance search or identification


## Matthias Kohl


June 20 (TODO), 2017

Research project performed at LIG - MRIM

Under the supervision of:
Georges Quénot, Jean-Pierre Chevallet

Defended before a jury composed of:
Head of the jury (TODO)
Jury member 1
Jury member 2

June 2017

**Abstract**

Your abstract goes here... TODO

**Acknowledgement**

I would like to express my sincere gratitude to .. for his invaluable assistance and comments in reviewing this report... Good luck :) TODO

**Résumé**

Your abstract in French goes here... TODO

# Contents

# — 1 —
# Introduction

## 1.1 Motivation

The research presented here is motivated by the GUIMUTEIC project, a collaborative project between industry and LIG. The aim is to develop a smart audio-guide for touristic or cultural sites.

In practice, the final product should offer an augmented reality interface to the user, with information about the object or objects the user is looking at.

One part of the development consists in finding ways of identifying objects the user is looking at. There are multiple possibilities, for example based on the geo-localization of the user and other sensors. In our research, we focus on the recognition of objects based only on visual clues.

## 1.2 Research problem

More specifically, we are interested in the following problem: We are given a collection with reference images for each object, or instance, to be recognized. The task is to develop a system that, given an image of one of the instances, can decide which instance the image represents. We assume that, on average, less than ten images are available for each instance. We will refer to this problem as instance retrieval in the following.

There are a few problems similar to instance retrieval. For one, there is the image classification problem: we are given a collection of images where each image is assigned a class, like dog or fridge. The task is to develop a system that, given an image, decides which class the image represents. This problem is of course similar if we simply consider each instance to be a separate class. However, in classification, we usually assume to have many images per class: hundreds or even thousands. This means our problem is closer to few-shots classification, where only few images are available for each class.

Another similar problem is image retrieval. In image retrieval, we are given a collection of reference images and a query image. We aim to rank the reference images by similarity to the query image. Usually, in image retrieval, reference images are not labeled or loosely labeled and many images are irrelevant to the query image. The challenge is to rank the most similar images on top. Instance retrieval is related to image retrieval in that we aim to develop a notion of similarity between images. However, in instance retrieval, we do not care about the rank of

the returned images, since we only consider the highest ranked image, which should represent the instance to retrieve.

Finally, to allow fast evaluation of the system, our goal is to develop a system that provides a single descriptor for each image, which should be small enough to easily be stored for all reference images and fast to compute. Two images should be compared according to a distance metric, which should also be fast to compute. This requirement is important, since a possible device used as an augment audio-guide will record many images per second, so to be responsive, it should be possible to evaluate the system in real-time or close to real-time.

## 1.3 Challenges

For all of the problems described above, deep learning approaches based on convolutional neural networks (CNNs) have recently obtained the state-of-the-art results. One of the drawbacks of CNNs is that they require large amounts of data to be trained.

In our research problem in particular, we do not have large enough amounts of data available to fully train a CNN. However, we aim to develop a system that can learn a descriptor specific to the reference images, as the system only needs to recognize and differentiate between instances of that particular dataset. So the system should be tuned to the dataset. Since we usually have less than ten images per instance, this is an important challenge to overcome.

Another challenge comes from the nature of the datasets: we were not able to find similar datasets in the literature, with only a few, clean images, for each instance and many instances. Common datasets in image retrieval like Oxford5k [19], Paris6k [20] or Holidays [12] contain many images per instance, but a lot of noise, as they are used to evaluate whether a system can robustly retrieve the correct images out of a set of random images. This means that the approaches used for image retrieval datasets may not work as well for instance retrieval.

## 1.4 Contributions

We made the following three major contributions:

1. We analyze existing approaches to image retrieval, classification and determine their shortcomings in our problem setting.

2. Based on these shortcomings, we propose a novel approach to improve results in our problem setting.

3. We evaluate the novel approach and compare with other approaches.

# — 2 —
# State of the art

## 2.1 SIFT and bag-of-words

Until recently, the state of the art in image retrieval and matching was based on the idea of bag-of-words [19] [18]. The idea behind the approach is to represent an image as a histogram, or collection of frequencies, of visual words. The visual words should be small, representative patches of the images in the dataset. Usually, these visual words are obtained in multiple steps. First, local features are extracted and encoded. The most common feature used is SIFT [17], a 128-dimensional vector representing scale-invariant features. Then, the features are extracted for all images and clustered into clusters of similar features. The representative for each cluster is the center of the cluster, i.e. the mean of all features falling into that cluster. Each image can then be represented as a histogram of the occurrences of these representative features. This histogram forms the image descriptor, which has as many dimensions as there are representative features and clusters.

Finally, for image classification, a classifier can be learned from the descriptors of all images in the dataset. On the other hand, for image retrieval or matching, the descriptors can be directly matched against each other, based on some similarity measure. In both cases, it can be useful to project the descriptors into a Hilbert space using a different similarity measure than euclidean distance in the original descriptor space. For classification, this is done by employing an SVM classifier with a non-linear kernel [27]. Among the most popular kernels are the radial basis function [25] and the chi-squared function [32].

Until recently, this approach has obtained state of the art results for image retrieval tasks [18].

## 2.2 Deep learning with CNNs

Starting with the results of AlexNet for image classification in the 2012 ImageNet challenge [14] [23], image classification tasks have been dominated by CNNs, learned using large amounts of data.

A general trend in image related tasks is to move to an end-to-end approach, where the final objective is directly optimized using gradient descent and the gradient is back-propagated to all previous parts of the system. In contrast, the bag-of-words model requires a choice of features (SIFT, ORB (TODO mention ORB before), ...), a choice of the method for clustering features (k-means), a choice of the classifier (AdaBoost [4], SVM, ...), as well as a choice of the kernel if an SVM classifier is used.

Using a CNN, features are extracted at a low abstraction level by the first convolutional layers, then higher level features are formed by combining low level features from the previous layers. Finally, high-level features are combined into a classifier by linear layers. The advantage of this approach is that the features are learned at all abstraction levels. Furthermore, the modularity of the approach allows us to easily transfer the lower level features learned from a large dataset to a smaller dataset, where there may not be enough data to efficiently learn lower level features.

The following sections describe the high-level architecture of the most widely used CNNs in classification and image retrieval.

### 2.2.1 AlexNet

AlexNet [14] contains five convolutional layers and three linear layers. The first convolutional layer has a large kernel and stride to quickly increase the receptive field of the consequent filters. The first two convolutional layers are followed by a pooling layer to further increase the receptive field.

Finally, an important improvement for AlexNet to avoid over-fit is the addition of dropout layers [9] before the two first linear layers.

Apart from that, AlexNet is a simple adaptation of LeNet by LeCun et al [15] for the ImageNet challenge.

### 2.2.2 VGG

The VGG architecture, introduced by Simonyan et al [28] is the first to use exclusively $3 \times 3$ convolutional kernels. This means that the receptive field of the first filters is much smaller, and thus many more layers are needed, as well as more pooling layers. The most popular VGG architectures are VGG-16 and VGG-19, having 16 and 19 layers in total respectively.

Using smaller kernels in convolutions and consequently more layers allows to reduce the number of trainable parameters used by the network, while increasing the number of non-linear layers. This was shown to allow better generalization properties, and thus less over-fit [28].

### 2.2.3 ResNet, Inception, DenseNet

The idea of increasing the number of layers is further developed by the ResNet, DenseNet and Inception architectures. However, naively increasing the number of layers leads to the problem of vanishing gradient [8].

All of these networks overcome the vanishing gradient problem by introducing skip connections, where the output of a previous layer is added to the output of a layer, skipping some number of layers in between.

The ResNet architecture by He et al [7] always uses blocks of 3 layers along with a skip of those 3 layers. The smaller types of ResNet use blocks of 2 layers.

The Inception architecture by Szegedy et al [29] uses a combination of skip connections of different length and different types.

Finally, DenseNet by Huang et al [10] takes the skip connection idea one step further: the input of each layer is dependent on the output of all $n$ previous layers to some extent.

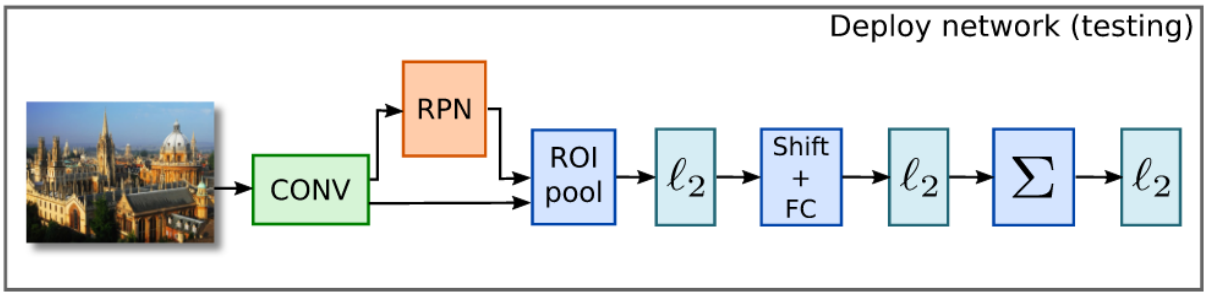



Figure 2.1 – Architecture of a CNN network for image retrieval, developed by Gordo et al [5]

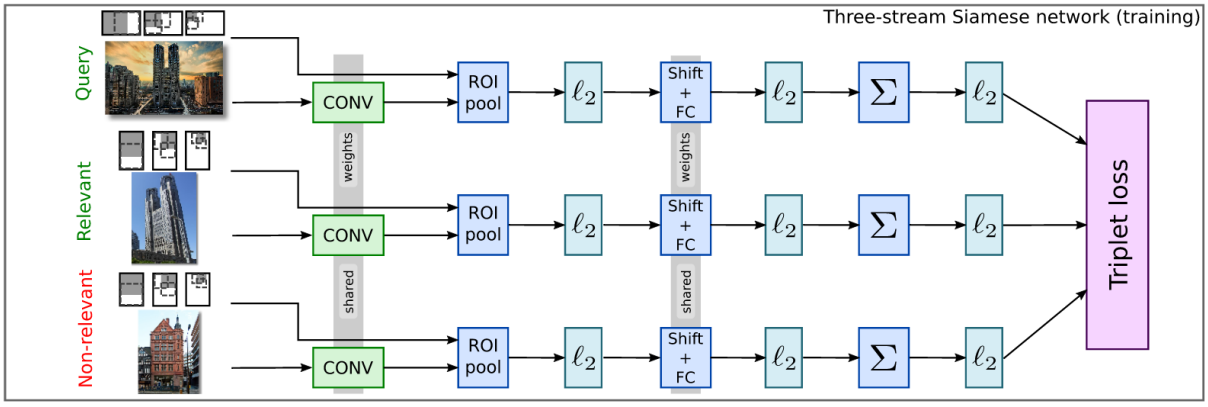



Figure 2.2 – Siamese architecture of a CNN for image retrieval used in training [5]

Finally, all of the very deep architectures use batch norm layers to further reduce over-fit: a batch norm layer simply normalizes the features over each batch, and then applies a learnable scaling and shifting.

## 2.3 Image retrieval using CNNs

For image retrieval, the current state of the art is set by Gordo et al [5]. It is based on an end-to-end approach. The goal is to learn a global descriptor for images that is well suited for comparing images.

Figure 2.1 shows the detailed architecture used. It first extracts the convolutional features of a pre-trained CNN. Then, a Region Proposal Network (RPN) [22] is used to extract the regions of interest. For each region of interest, a shifting and linear layer are used to reduce the dimensionality of the descriptor. The final descriptor is simply a normalized sum of the region-wise descriptors. This network can be learned end-to-end and the obtained descriptor achieves state of the art results, which can be even further improved by using query expansion and database side feature augmentation (TODO either remove or mention before).

Gordo et al [5] found that using very deep networks outperforms the shallower networks. The state of the art results are thus set by a very deep ResNet architecture.

### 2.3.1 Siamese networks and triplet loss

Figure 2.2 shows the architecture used to train the CNN that produces a descriptor for images, as can be seen in Figure 2.1.

The major difference is that during training, a Siamese architecture is used: the CNN is evaluated with multiple images, using the same weights for each. Then, a combined loss is obtained from the descriptors. Finally, the loss is back-propagated once through all streams of the network, since shared weights are used for all images.

In the case of this particular architecture, three images are evaluated and a triplet loss is used. This triplet loss has previously been used in face recognition tasks by Schroff et al [26]. The triplet loss was first introduced by Weinberger et al [34] as a way of learning the best suited distance metric in a k-nearest neighbor classification problem. Section 3.2.1 describes this loss in more detail. This is the reason why the triplet loss was chosen for image retrieval: the goal is to learn a metric which allows to discriminate couples of images containing the same instance from couples of images containing different instances, using the descriptors obtained from the Siamese CNN.

Finally, Gordo et al made further contributions, detailed in their follow-up paper [6].

### 2.3.2 Dataset

For one, an important step is to use a suitable dataset. Datasets like Oxford5k or Paris6k are too small for a network to learn a distance metric applicable to any type of image. Instead, Gordo et al use the much larger Landmarks dataset, created by Babenko et al [2]. This dataset contains almost 200000 images in total of around 600 different landmarks, mostly buildings.

Another important step is the cleaning of the dataset. The Landmarks dataset contains a lot of noise and weak labels: many images do not represent the building they are labeled with. Gordo et al use a meticulous cleaning process: the main idea is to keep only the largest clusters of matching objects for each of the landmarks. The implementation is based on extracting SIFT and Hessian descriptors, matching them and verifying matches with an affine transformation model. This cleaning process is used in order to annotate the cleaned images with regions of interest as well.

The cleaning requires a lot of processing, but is only done once for the dataset. The cleaned dataset is still fairly large, containing 49000 images and 586 instances of landmarks.

### 2.3.3 Triplet choice

Another important step is the choice of triplets during training of the Siamese network. Schroff et al [26] and Weinberger et al [34] already identified that using random triplets for training means that most triplets do not contribute to the loss at all, thus leading to vanishing gradients. So it makes sense to try and choose only the hardest triplets: for each reference image, find the image with the same label that is furthest from the reference image according to the current metric (the hardest positive), and the closest from the reference image having a different label (the hardest negative). However, choosing the hardest positive and hardest negative at each step is computationally too expensive. So Schroff et al [26] propose choosing the hardest triplets for each batch. Since there are only a few positive couples available in each batch, we simply choose all positive couples along with the hardest negative for each reference image in every batch. Since this can lead to a collapse in the model, Schroff et al propose choosing semi-hard

negatives in the first epochs of training: for each reference image and positive sample, choose the hardest negative that is easier (further according to the metric) than the positive.

The disadvantage of this strategy is that it requires large batches. Instead, Gordo et al [6] uses the following strategy: pre-compute all descriptors in each epoch of training. Then, for each reference image, choose the $n$ easiest positive samples and the $m$ hardest negative samples. Compute the loss for all possible combinations and use the $n$ hardest triplets, with highest loss. This allows using smaller batches, while keeping the cost of computation low, since descriptors are only computed every epoch. It may also help eliminate some noise in the data, since only the easiest positive couples are chosen.

### 2.3.4 Limitations and goals

The main limitation of Gordo's method is that it aims at producing a general descriptor for comparing any type of image. This means that it is most effective when trained on large datasets, with many samples per instance. For datasets containing images from museums or tourist sites, this is usually not applicable, as there are usually only very few images available for each instance. While there are large datasets, these datasets usually contain a large number of instances as well. This makes it difficult to directly apply the state of the art on our datasets.

On the other hand, it also means that the state of the art in image retrieval wants to avoid over-fit at all costs: the learned descriptor and metric should be able to provide a good metric for any kind of image. While this is usually desired in machine learning, in our case, one goal is to provide a metric specifically designed to discriminate between the images of the given dataset. This is because an audio-guide in a museum can be fine-tuned for that museum, as long as the fine-tuning is computationally cheap enough. So we do not care if a given audio-guide cannot identify images of multiple museums, as long as it can identify images of one museum with high precision.

Another limitation of the state of the art is that it requires a tedious process to clean the dataset and annotate it with regions of interest. This is not suited to museum datasets which are usually very clean to begin with, by the nature of the dataset. Plus, the notion of region of interest is usually not relevant for this type of image: which part of a painting or piece of art contains the semantically relevant information is not obvious.

Finally, an evaluation of the CNN developed by Gordo et al [5] was carried out. We used the published weights obtained in training this CNN on the Landmarks dataset. Figure 3.2 shows examples of images in our dataset that were correctly matched with high confidence as well as images that were incorrectly matched. From these images, we make the following observations in Section 3.3.2:

1. Images that are very similar visually are well matched

2. Images at different scales are not well matched

Thus, a limitation of the current state of the art is matching images, where the object of interest is at different scales.

# — 3 —
# Contributions

## 3.1 Fine-tuning a CNN

A first possible approach to this problem is to fine-tune a classification network to instance classification. This means that we consider each instance as a class, and optimize using a cross-entropy loss, typical for classification.

$$\mathscr{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i) \tag{3.1}$$

Equation 3.1 describes the cross-entropy loss for a given instance. $N$ is the number of samples, $y_i$ an indicator variable taking the value 1 if sample $i$ belongs to the given instance and 0 otherwise, and $\hat{y}_i$ is the predicted probability that sample $i$ belongs to the given instance.

Fine-tuning a CNN for classification on different data has been studied intensively by Yosinki et al [35]. In particular, their study shows that it is only important to fine-tune the neurons of higher layers of a CNN. Furthermore, they show that fine-tuning can increase generalization even in the fine-tuned model. Both of these properties are desirable in our task, because they reduce the memory and time needed to train a network, as well as the need for a large dataset used for fine-tuning.

The specific considerations we take with regards to fine-tuning are described in the following sections.

### 3.1.1 Data augmentation

Our datasets, presented in Section 4.1, contain an average of less than 10 samples per instance. This is too little to train a typical CNN model designed for classification, even when fine-tuning. One way to overcome this is to augment the data, by randomly applying affine transformations, color perturbations and other random transformations.

Since we specifically identified an issue related to different scales in images in Sections 2.3.4 and 3.3.2, it it natural to augment the data by randomly scaling the images in both dimensions.

We found that randomly rotating and flipping the images improved performance as well, thus we perform this type of data augmentation throughout our experiments.

For data augmentation in order to fine-tune a CNN, we use the following values in our experiments:

1. Rotation: any angle is chosen with the same probability.

2. Scaling: the scaling factor is chosen independently for each dimension in the range $[0.75, 1.25]$.

3. Flipping: with probability 0.5, images are horizontally flipped.

### 3.1.2 Transfer learning

The modularity of a CNN means that we can easily transfer the weights from a pre-trained model, and only retrain the highest abstraction layers. Specifically, we re-train all linear layers in the model, representing the highest-level layers.

We also re-train the highest level convolutional layers, since our datasets contain many visually different images as compared to the ImageNet dataset used for pre-training the models. For the AlexNet architecture, we choose to re-train all layers above and including the last convolutional layer. For a ResNet architecture, we re-train all layers above and including the third to last block of convolutional layers. This contains the nine highest convolutional layers in total.

## 3.2 Simplified Siamese architecture

We follow the methodology of Gordo et al [5] and propose a simplified Siamese architecture. As shown in Figure 2.2, the previous state-of-the-art is set by a Siamese architecture using the triplet loss.

### 3.2.1 Triplet loss

$$\mathscr{L} = \sum_{i=1}^{N} \frac{1}{2} \max(0, \|x_i^a - x_i^p\|_2^2 - \|x_i^a - x_i^n\|_2^2 + 2m) \tag{3.2}$$

Equation 3.2 describes the triplet loss for $N$ triplets. Each triplet is composed of an anchor image with descriptor $x^a$, a positive/relevant image with descriptor $x^p$ and a negative/non-relevant image with descriptor $x^n$. A margin $m$ is added to the loss, which describes the distance at which two images are considered similar enough to be relevant with respect to each other. Throughout our experiments, this margin is set to a value of $m = 0.1$.

For normalized descriptors $x$ and $y$, the squared distance $\|x-y\|_2^2$ is closely related to the cosine similarity and the dot-product through the relationship derived in Equation 3.3.

$$\begin{aligned} &\forall x, y \in \mathbb{R}^n, \|x\|_2 = 1, \|y\|_2 = 1 \\ &\|x-y\|_2^2 \;\; = (x_1 - y_1)^2 + \cdots + (x_n - y_n)^2 \\ &\qquad\quad = (x_1^2 + \cdots + x_n^2) - 2(x_1 y_1 + \cdots + x_n y_n) + (y_1^2 + \cdots + y_n^2) \\ &\qquad\quad = 2 - 2xy = 2 - 2\cos(x, y) \end{aligned} \tag{3.3}$$

With this Equation, we can re-write the triplet loss for normalized descriptors using the dot-product, as shown in Equation 3.4. This equation represents the version of the triplet loss used in our experiments. This means all descriptors must be normalized before the loss is evaluated.

$$\mathscr{L} = \sum_{i=1}^{N} \max(0, x_i^a x_i^n - x_i^a x_i^p + m) \tag{3.4}$$

### 3.2.2 Simplified architecture

We simplify the architecture shown in Figure 2.2 by removing the region proposal network and region of interest pooling. We justify in Section 3.3.1 why these components are not relevant for our datasets and research problem in general. Except for this change, the architecture is kept exactly as described by Figures 2.1 and 2.2.

### 3.2.3 Triplet selection

As noted by previous authors, when using the triplet loss, it is crucial to choose the best triplets during training in order to obtain convergence. In particular, many triplets are irrelevant and do not produce any loss since they are *too easy* for the network.

Hence, the first idea is to choose the hardest triplets, as proposed by Schroff et al [26]. However, as they show in their paper, this can lead to a collapsing model early on in training so they choose *semi-hard* triplets instead. Semi-hard triplets are obtained as follows: use all possible positive couples of images: couples of images from the same instance. For each positive couple, choose the hardest negative that is easier than the positive couple. Hard and easy are defined by the dot-product between the descriptors of the images: a high value of the dot-product for images of the same instance represents an easy positive couple, a high value of the dot-product for images of different instances represents a hard negative couple. The value of all dot-products are determined before each pass over the whole training data during training, for all couples of images.

Gordo et al [6] propose a more complex mechanism. First, they also calculate the values of dot-products for all couples of images before each pass over the training data. Second, for each image, choose the $n$ easiest positive images and the $m$ hardest negatives. Then, calculate the loss for all possible combinations and use the $o$ triplets with the highest loss. This method probably eliminates some noise when choosing the easiest positive couples, for images that are labeled as being the same instance but are not visually similar. However, in experiments, we found that this method does not perform well for datasets with few images per instance, since we either have to choose $n$ as very low or we choose all positive couples after all for most instances.

Hence, in our experiments, we choose the semi-hard triplet selection for the first two passes over the dataset, after which we only choose the hardest negatives for all positive couples.

## 3.3 Analysis of previous approaches

### 3.3.1 Identifying regions of interest

Previous approaches in image retrieval [6] [24] [30] usually deal with regions of interest in one way or another. The idea is that in most cases, only certain parts of each image can be useful for comparison with other images. In addition to this, cropping images at their regions of interest can help with differences in scale of the images to compare: if a building is visible only in a small part of an image, cropping the image at that part and then re-scaling the part should set the building at a normalized scale.

However, in instance search with museum datasets, it is not obvious where the regions of interest should be: most images represent an entire painting or parts of it and only some may contain the painting as part of the image with a wall in the background. This means for most

images, the ground-truth region of interest is simply the entire image, and some may have a ground-truth region of interest which is almost the entire image, excluding only a small part of the background.

On the other hand, a network fine-tuned on classification on such a dataset should be able to easily identify the region containing the painting, since the background wall is contained in almost all classes, which means it is a particularly bad indicator of the class. Thus, if the network is applied in a strided manner across an image, it should produce low maximal activations in parts containing big sections of background wall.

Figure 3.1 shows images, along with the heat map representing the maximal activation of a fine-tuned ResNet-152 at each coordinate, when the network is applied in a strided manner across the input image. The fine-tuning was carried out as described in Section 3.1. From this image, we can see that the highest maximal activations of the network usually occur at the location of the object. This is true even if the object is not correctly classified by some of the highest activations as can be seen in images 3.1d - 3.1f.

In the images 3.1g - 3.1i, it seems like many high maximal activations occur specifically in the background area. However, the corresponding label-map shows that these areas correspond to the labels 38E and 43D. Both of these labels are pieces of art which consist mostly of the background wall. In this sense, it is not entirely wrong to consider 'wall-only' patches of the image as instances of these pieces of art. This simply means that the image consists of two separate regions of interest: one region with the painting (label 30P) and one region with the wall (labels 38E/43D).

From these observations, we can confirm the assumption that the maximal activations of a fine-tuned network are a good indicator of the location of an object, or a combination of different objects. Using this assumption, there is no need for a procedure to annotate regions of interest, as employed by most image retrieval approaches.

On the other hand, using datasets developed for image retrieval, such as Paris6k [20] or Oxford5k [19], this assumption cannot be applied, since the dataset is not clean enough for a fine-tuned network to be a good indicator of location of the query objects.

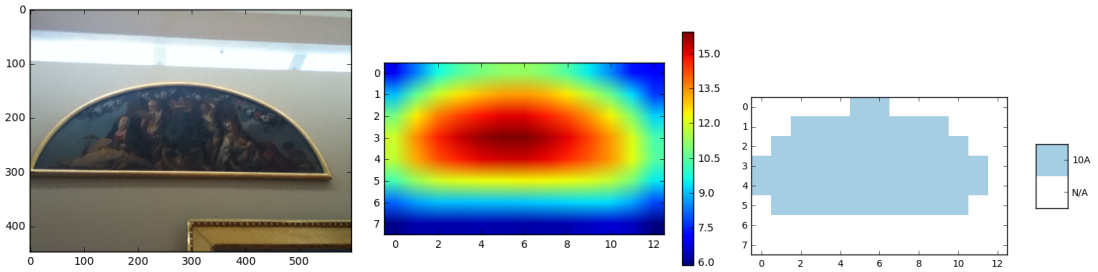
### 3.3.2 Incorrectly identified images

Figure 3.2 shows typical images that were correctly identified versus images that were incorrectly identified, by a fine-tuned network as well as the network published by Gordo et al [5], along with the average precision for the respective queries.

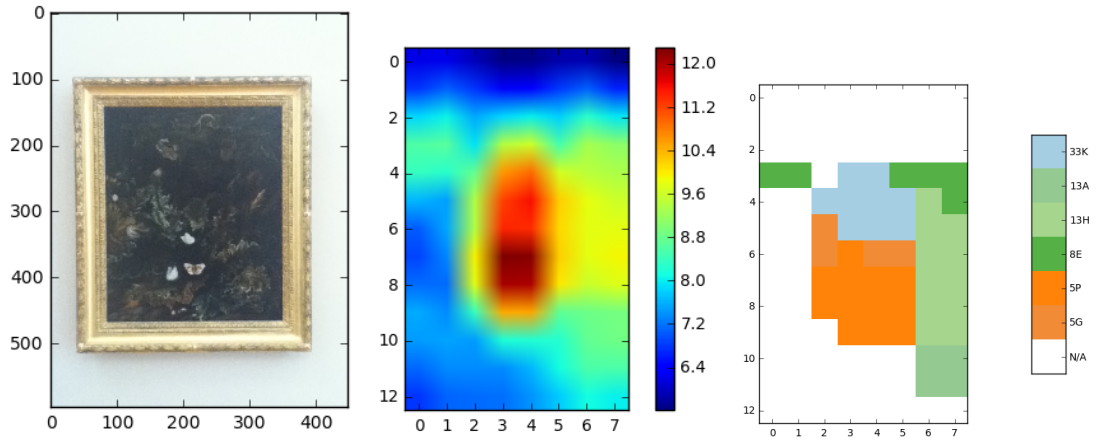
From these results, we see that the networks struggle with images at very different scales, achieving very low average precision, while images with similar scales are usually perfectly matched. Combining both of the properties identified in this section, we propose a novel approach to learn a strong descriptor for instance search in the next section.

## 3.4 Proposed approach

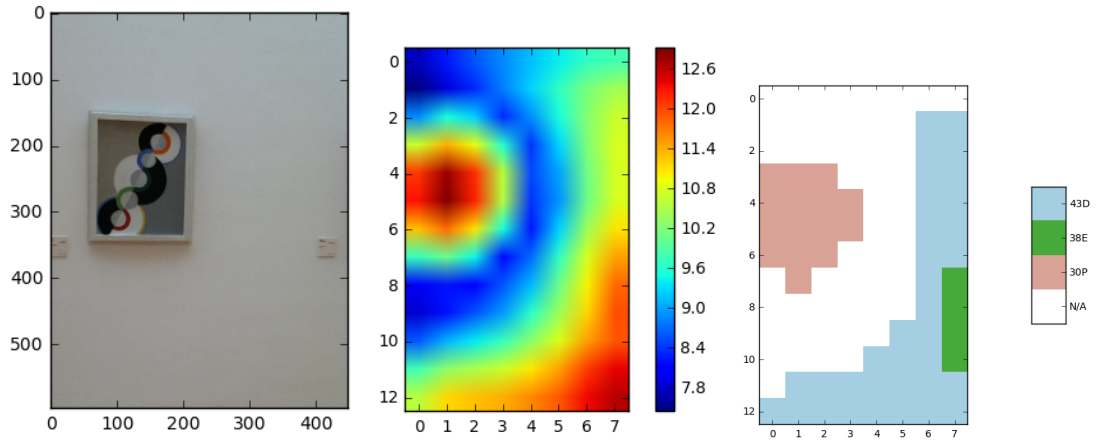
The proposed approach is based on multiple steps as described in the following sections.




(a) Image with label 10A (b) Heat-map for 10A (c) Label-map for 10A




(d) Image with label 5P (e) Heat-map for 5P (f) Label-map for 5P

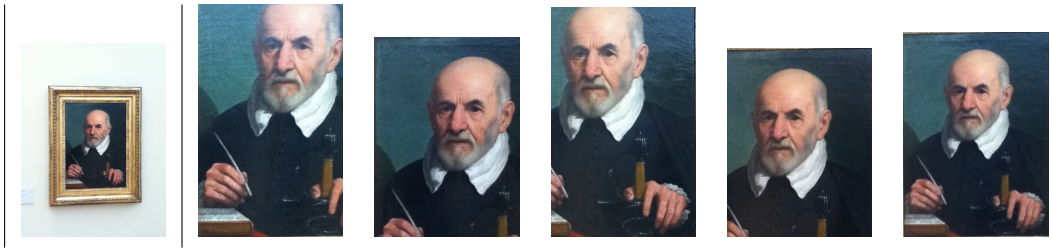



(g) Image with label 30P (h) Heat-map for 30P (i) Label-map for 30P

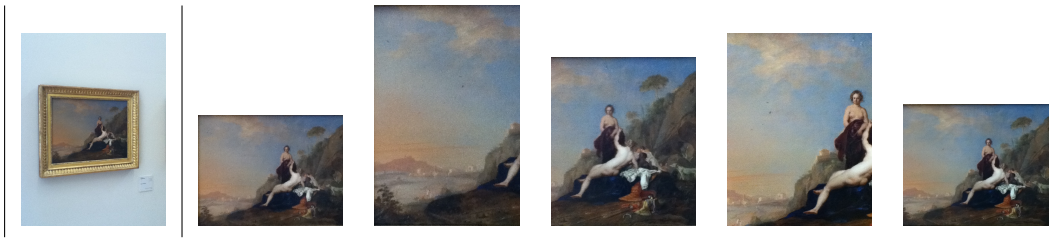
Figure 3.1 – Sample images (scaled to a smaller side of 448 pixels) along with the heat-map of maximal activation values at each location when a fine-tuned ResNet-152 is applied to the image in a strided manner, as well as the labels of all maximal activations that are greater than the mean maximal activation

(a) Correctly identified query with label 11J, along with its reference images.
Average precision Gordo's net: 1.0, fine-tuned ResNet-152: 1.0

(b) Correctly identified query with label 23D, along with its reference images.
Average precision Gordo's net: 1.0, fine-tuned ResNet-152: 1.0



(c) Incorrectly identified query with label 1C, along with its reference images.
Average precision Gordo's net: 0.113, fine-tuned ResNet-152: 0.014



(d) Incorrectly identified query with label 5B, along with its reference images.
Average precision Gordo's net: 0.013, fine-tuned ResNet-152: 0.002

(e) Incorrectly identified query with label 11C, along with its reference images.
Average precision Gordo's net: 0.004, fine-tuned ResNet-152: 0.004

Figure 3.2 – Sample images that were correctly and incorrectly identified by a fine-tuned ResNet-152, as well as the network published by Gordo et al [5], referred to as Gordo's net. For each image, the query image is shown at the very left and all possible correct reference images are shown to the right of the query. The caption contains the associated average precision values.

### 3.4.1 Fine-tuning on classification using an FCN

As shown in Section 3.3, a fine-tuned CNN is already a good indicator of the location of an object in our datasets. Additionally, it seems like scale is a particularly important factor.

Thus, the idea is to start by fine-tuning a network with images at different scales. This can be achieved by using a fully convolutional network (FCN), as introduced by Long et al [16].

In a FCN, the final fully connected layers of a network are replaced by convolutional layers having a kernel which fits the entire domain of the output of the previous layer. This type of convolution is equivalent to a fully connected layer, but allows inputs (and outputs) of any size. The effect is that the network can be applied in one pass to an arbitrarily sized image. The output then represents the activations of the network as if it was applied in a strided manner across the image. The stride of a full network depends on the architecture and is 32 pixels for the architectures used here: AlexNet and ResNet.

Once a FCN is applied to the image, the loss is calculated by averaging the cross-entropy loss across all locations and outputs. The final loss is then obtained by passing images at different scales through the FCN and averaging across all cross-entropy losses of all outputs and scales. Equation 3.5 shows the loss for a single image. $S$ represents the number of scales at which the image is passed through the network. $H_s$ and $W_s$ represent the height and width of the feature map respectively, given the scale $s$ of the input image. $y^{h,w}$ represents the prediction of the network at spatial location $(h,w)$ in the full feature map.

$$\mathscr{L} = \frac{1}{S}\sum_{s=1}^{S} -\frac{1}{H_s * W_s}\sum_{h=1}^{H_s}\sum_{w=1}^{W_s} y^{h,w}\log\hat{y} + (1-y^{h,w})\log(1-\hat{y}) \tag{3.5}$$

We can see from the Equation that we chose to give each scale of the image the same weight in the loss. Additionally, we can see that this loss can only be applied to one image at a time, hence the true label $\hat{y}$ is a constant in Equation 3.5. This is because the images are passed to the network at their true aspect ratio, which means the loss for different images may have different values for the heights and widths of the feature maps $H_s$ and $W_s$.

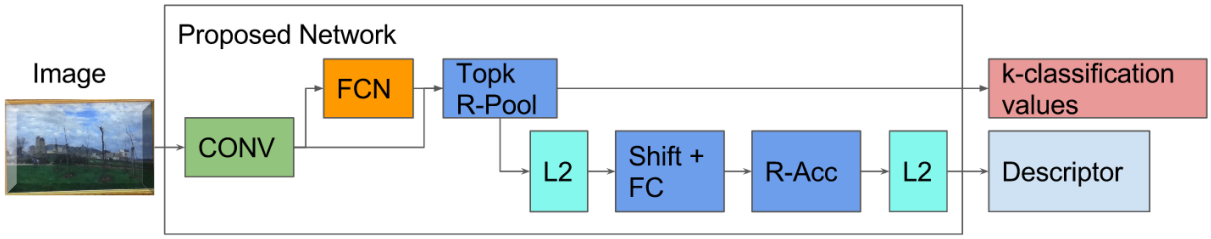
In order to normalize the sizes of the features present in the images, all images are scaled to have the same number of pixels in the smaller side. Note that for large aspect ratios and large scales of the smaller side, the memory consumption of training can be high for single images having a very large aspect ratio. To limit this spike in memory consumption, the aspect ratios are limited by introducing uniform random noise on the smaller side of images with high aspect ratios.

In our experiments, we use a maximal aspect ratio of 2.0 and images at two scales of 448 and 224 pixels for the smaller side. We found that the AlexNet architecture did not have good convergence behavior, thus we used scales of 384 and 224 instead.

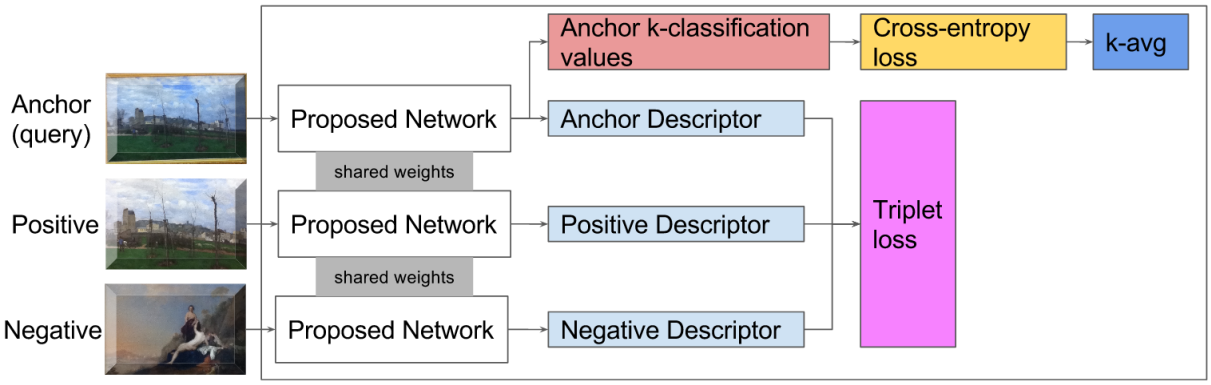
#### Gradient descent using micro-batches

Since only a single image is passed to the network at a time, the stochastic gradient descent algorithm used to optimize the parameters of the model does not converge, as a single image is not representative of the dataset as a whole and does not form a large enough mini-batch.

Instead, we form a mini-batch by accumulating gradients from many *micro-batches*, each of which is made up of only a single image. This allows to form a mini-batch of any size to obtain stable convergence in stochastic gradient descent.




(a) Proposed architecture for instance search, at deploy time




(b) Proposed architecture for instance search, at training time

Figure 3.3 – Proposed architecture for instance search, based on a FCN [16] for region proposals. The descriptor extraction for each region is similar to the architecture by Gordo et al [5], as shown in Figures 2.1 and 2.2

However, using a micro-batch formed of a single image represents an issue with the ResNet architecture: this model uses batch-norm layers, introduced by Ioffe et al [11]. When training a model, the layer normalizes each batch by its mean and variance. However, it keeps a running mean and variance, which approximates the true mean and variance of the dataset and which is applied during testing. With a micro-batch size of 1, this running mean and variance cannot be inferred, which leads to very poor behavior of the model. Instead, we pre-train the model on the dataset as described in Section 3.1 in order to obtain a correct estimate of the running mean and variance. Then, we preset the network with the parameters of this fine-tuned network when training the FCN as described above.

### 3.4.2 Descriptor extraction network

The second step of the proposed approach relies on the FCN, trained as described in the first step in Section 3.4.1.

Figure 3.3 illustrates the proposed architecture. To obtain a descriptor, we first apply the convolutional layers of a previous architecture, such as AlexNet or ResNet. We then obtain all classification outputs at all locations using the pre-trained FCN. We only consider the maximal activation at all locations. The locations with the top $k$ maximal activations will form the descriptor.

For each of these locations, we closely follow the architecture proposed by Gordo et al [5], as shown in Figure 2.1. The convolutional features are reduced by a $\|\cdot\|_2$-normalization, then a shifting and fully connected layer. Finally, all descriptors from the $k$ locations are sum-

aggregated and $\|\cdot\|_2$-normalized again.

When training, the network is applied to a triplet of images. These triplets are chosen as described in Section 3.2 for the simplified Siamese architecture.

Additionally, we regularize the triplet loss by a cross-entropy loss to make sure that the $k$ locations with highest maximal activations are correctly classified. This loss is averaged over the $k$ locations.

Equation 3.6 shows the full loss as used in our experiments to train the proposed model, for a $N$ images. In this equation, $(h_l, w_l)$ represent the spatial coordinates of the $l$-th region of highest maximal activation in the feature map produced by the FCN.

$$\mathscr{L} = \sum_{i=1}^{N} \left( \max(0, x_i^a x_i^n - x_i^a x_i^p + m) + \alpha \frac{1}{N} \frac{1}{k} \sum_{l=1}^{k} -y_i^{h_l,h_l} \log \hat{y}_i + (1 - y_i^{h_l,w_l}) \log(1 - \hat{y}_i) \right) \quad (3.6)$$

In our experiments, we choose the number of regions with highest maximal activation to be $k = 4$ and the regularization hyper-parameter $\alpha = 1.0$. The margin of the triplet loss is $m = 0.1$ as described in Section 3.2.

### 3.4.3 Advantages

As shown in Section 3.3, this approach allows the network to decide which region of interest is best suited for classification and ultimately which regions are best suited for comparison with other images.

Another advantage is that this approach does not require any annotation of the images with regions of interest, which can be a long, manual or automatic process, as evident from the cleaning process used by Gordo et al [6].

Finally, an important property of the descriptor is that it heavily relies on the classification capabilities of the network. This means the descriptor is mostly meaningless for a different dataset and needs to be learned for each dataset. This can be an advantage, since the descriptor can be better suited to a particular dataset and the learning process does not take long, as shown in Section 5.1. On the other hand, this means that the descriptor cannot be applied in a typical image retrieval task.

# — 4 —
# Evaluation

## 4.1 Datasets

The proposed approaches as well as several base-lines are evaluated on two datasets: the CLICIDE and GaRoFou datasets. These datasets are described in detail by Portaz et al [21]. Both datasets are typical of instance search datasets in museums or touristic sites: the objects represented by their images are paintings for one and glass cabinets containing sculptures and artifacts for the other dataset. Both datasets contain a small number of images per instance and a small number of images in total. Table 4.1 lists the content and statistics of the datasets in detail.

### 4.1.1 CLICIDE

The CLICIDE dataset contains photographs taken in the Grenoble Museum of Art. The objects represented in the photographs are paintings and still images, exclusively. The full dataset contains 207 images with no meaningful content, mostly showing walls. These images are labeled as such and have been filtered from the dataset in our evaluation. This explains the difference between total number of images and number of images in Table 4.1. Furthermore, there are 12 query images which contain objects not present in the reference images. These have been filtered out as well.

The CLICIDE dataset is characteristic because the different images for each instance usually consist of at least one global view of the painting and multiple other images representing sub-regions of the same painting.

### 4.1.2 GaRoFou

The GaRoFou dataset contains high-resolution photographs taken in the Museun of Fourvière in Lyon. The objects represented in the photographs are a cultural heritage collection: sculptures, steles and small artifacts grouped together in glass cabinets.

The full GaRoFou dataset is larger than the part used here: along with the dataset of images, it contains a dataset of videos as well as images extracted from these videos. The video collection has not been used as part of our research.

| | *CLICIDE* | *GaRoFou* |
|---|---|---|
| *Type of objects* | Paintings | Glass cabinets containing various objects |
| *#Instances* | 464 | 311 |
| *#Reference images* | 3245 | 1068 |
| *#Query images (#instances)* | 165 (134) | 184 (166) |
| *Median #images per instance* | 6 | 2 |
| *Min #images per instance* | 1 | 1 |
| *Max #images per instance* | 22 | 45 |
| *Total reference images (#instances)* | 3452 (473) | 1068 (311) |
| *Total query images (#instances)* | 177 (143) | 184 (166) |

Table 4.1 – Details of the datasets used for evaluation

## 4.2 Metrics

There are several commonly used metrics in classification and information retrieval. The following sections define these metrics and their use in our research.

### 4.2.1 Precision

$$P = \frac{TP}{PRED} \tag{4.1}$$

$$P_q = \frac{RR_q}{RET} \tag{4.2}$$

$$P_q@k = \frac{RR_q^k}{k} \tag{4.3}$$

$$MP@1 = \frac{1}{Q}\sum_q P_q@1 \tag{4.4}$$

In classification, precision is a measure of how many correct predictions were made out of all predictions. Equation 4.1 shows this relationship: $TP$ are the true positives, the correct predictions, and $PRED$ is the number of all predictions, including incorrect ones. $PRED$ is usually equal to the size of the testing dataset in classification.

In information retrieval, similarly, precision and more generally Precision@$k$ is a measure of how many relevant documents were retrieved out of all retrieved documents. However, the metric is evaluated for each query, instead of the whole testing dataset. Equations 4.2 and 4.3 illustrate this: $P_q$ is the precision for query $q$, $RR_q$ represents the number of retrieved and relevant documents for query $q$. $RET$ represents the total number of retrieved documents. Finally, $RR_q^k$ represents the number of retrieved, relevant documents out of the $k$ first retrieved documents for query $q$.

Thus, Precision@$k$ simply uses a cut-off at $k$: it measures how many relevant documents there are in the first $k$ retrieved documents.

In instance search, only the first retrieved document is important: If the first result is correct, we correctly identify the instance. Thus, we are mostly interested in Precision@1 in our research. In order to generalize this metric to all queries, the mean Precision@1 is used, as defined in Equation 4.4. Here, $Q$ represents the number of queries.

### 4.2.2 Mean average precision

$$R_q = \frac{RR_q}{REL_q} \tag{4.5}$$

$$AP_q = \frac{1}{REL_q} \sum_{k=1}^{RET} rel(RET_k) P@k \tag{4.6}$$

$$MAP = \frac{1}{Q} \sum_{q} AP_q \tag{4.7}$$

In information retrieval, average precision is commonly used, since it allows to consider the order in which documents are returned by a system. Average precision represents the area under the precision-recall curve for a given query $q$. Recall for query $q$ is defined as in Equation 4.5 as the fraction of retrieved, relevant documents for query $q$ as compared to all relevant documents to query $q$.

Equation 4.6 defines average precision for a given query $q$: it can be expressed as a finite sum over the Precision@k values at every possible cut-off $k$, multiplied by an indicator $rel(RET_k)$, which is 1 if the $k$-th retrieved document is relevant. This sum is then normalized by the total number of relevant documents for query $q$ $REL_q$.

Finally, mean average precision $MAP$ represents the mean of all average precision values over $Q$ queries, as defined in Equation 4.7.

In instance search, mean average precision is not important as we only care about the first ranked result. Nevertheless, it can be an interesting metric to allow better comparison between different systems, so we compute the mean average precision for all results along with the mean Precision@1.

## 4.3 Evaluation methodology

$$\begin{aligned} &\forall x, y \in \mathbb{R}^n, \|x\|_2 = 1, \|y\|_2 = 1 \\ &\|x-y\|_2^2 = x_1^2 + \cdots + x_n^2 + y_1^2 + \cdots + y_n^2 - (2x_1y_1 + \cdots + 2x_ny_n) = 2 - 2xy \end{aligned} \tag{4.8}$$

Unless explicitly noted, an evaluation of the system is carried out as follows:

All reference images are encoded with a descriptor extracted from the network to be tested. The query image is encoded the same way. All descriptors are normalized and the descriptor of the query image is compared to all reference images by computing the dot product between it and all descriptors of the reference images. This can be performed by a single vector-matrix multiplication. The resulting values describe the cosine similarity between the descriptor of the query image and the descriptors of reference images.

This is because the cosine similarity between two normalized vectors is equal to the dot product between the two vectors, from the definition of the cosine similarity. Furthermore, Equation 4.8 shows that the dot product is simply inversely proportional to the squared euclidean distance between two normalized vectors. This means that a high cosine similarity, high value of the dot product and low squared euclidean distance between two descriptors have the same meaning in the context of this research.

Hence, after a query image is compared to all reference images through a vector-matrix multiplication, the index of the highest value in the resulting vector indicates the most similar reference image.

| | *CLICIDE* | *GaRoFou* |
|---|---|---|
| *SIFT* | 70.08 | 78.82 |
| *AlexNet IM* | 57.58 | 76.63 |
| *ResNet-152 IM* | 64.24 | 75.54 |
| *Gordo et al [5]* | 90.30 | 95.65 |
| *Gordo et al [5] (multi-resolution)* | 92.73 | 95.65 |

Table 4.2 – Evaluation base-lines set for the CLICIDE and GaRoFou datasets. The results are expressed in percentage points of Mean Precision@1

## 4.4 Baselines

Table 4.2 shows the baselines set for the two evaluation datasets, as described in Section 4.1. The metric used to compare systems is mean Precision@1, as described in Section 4.2.1, and the evaluation is carried out as described in Section 4.3.

The descriptors used for comparing images are extracted using different methods. The SIFT method has been previously evaluated by Portaz et al [21], where it is detailed.

For the *AlexNet IM* and *ResNet-152 IM* methods, an AlexNet and ResNet-152 architecture is pre-trained on the ImageNet dataset. Then, the descriptor is simply extracted by passing an image through the network and normalizing the output. As ImageNet contains 1000 classes, the output has 1000 dimensions. Each image is simply pre-processed by scaling it to the expected size of the network at $224 \times 224$.

The *Gordo et al* [5] method is based on the pre-trained network published by Gordo et al. For this network, images do not need to be pre-processed, as the network works with region proposals and thus accepts any scale of image. This network outputs a descriptor with 2048 dimensions.

## 4.5 Performance

The performance evaluation is carried out on a typical system used for training CNNs. The components approximately represent the highest standard available for a server architecture in 2016.

The server contains the following components:

1. CPU: Intel® Xeon® E5-2623 v4, 16 cores, 10MB cache, 2.60GHz
2. Memory: 142 GB, DDR4
3. Graphics card: NVIDIA® Titan X, Pascal architecture, 12 GB memory

Performance is measured in time in seconds required to execute the task, as well as GPU memory required. If multiple sub-tasks are performed for a task, the reported performance is simply the accumulation of time and the maximum of required GPU memory of the sub-tasks.

### 4.5.1 Training

The training performance is evaluated using the server as described above and the CLICIDE dataset as described in Section 4.1, containing 3245 images.

The training performance was measured only for tasks specific to the dataset, as other tasks can be performed a single time upfront. In particular, this means that pre-training a network on a dataset which is not specific to our task is counted as utilizing no time or memory.

### 4.5.2 Testing

The testing performance is evaluated using the server as described above and the validation dataset of CLICIDE, as described in Section 4.1. This dataset contains 165 images. However, the performance is reported as the average performance for a single image, as this is the most common use-case when applying the instance search system.

# — 5 —
# Results

In this section, we present the experimental results obtained in our research.

## 5.1 Performance

Table 5.1 shows the performance of various approaches when training on the CLICIDE dataset. We assume that pre-trained networks have 0 cost. This table shows that fine-tuning a network on classification on the new dataset is quite fast and has a low memory footprint. This is interesting since fine-tuning already achieves better results than many standard approaches, as can be seen in Section **??**. We can also see that training our proposed architecture is much more costly, although the absolute cost is not excessive: around 48h (TODO) of training time in total for the very deep ResNet-152.

Table 5.2 shows the performance of these approaches when evaluating them on a single image. We can see that all approaches have approximately the same performance, with

Note that the performance reported for testing is not relevant in absolute numbers, since it was established on a high-end server, usually unavailable when testing a system. However, the results are relevant in relative terms when comparing different systems.

| | *CLICIDE training* | |
|---|---|---|
| | Time (s) | GPU memory (MB) |
| *AlexNet IM* | 0 | 0 |
| *ResNet-152 IM* | 0 | 0 |
| *Gordo et al [5]* | 0 | 0 |
| *Gordo et al [5] (multi-resolution)* | 0 | 0 |
| *AlexNet FT* | 0 | 0 |
| *ResNet-152 FT* | 0 | 0 |
| *Proposed AlexNet* | 0 | 0 |
| *Proposed ResNet-152* | 0 | 0 |

Table 5.1 – Performance of different approaches in terms of time in seconds and maximal GPU memory usage in MB when training on the CLICIDE dataset

| | *CLICIDE test per image* | |
|---|---|---|
| | Time (s) | GPU memory (MB) |
| *AlexNet IM* | 0 | 0 |
| *ResNet-152 IM* | 0 | 0 |
| *Gordo et al [5]* | 0 | 0 |
| *Gordo et al [5] (multi-resolution)* | 0 | 0 |
| *AlexNet FT* | 0 | 0 |
| *ResNet-152 FT* | 0 | 0 |
| *Proposed AlexNet* | 0 | 0 |
| *Proposed ResNet-152* | 0 | 0 |

Table 5.2 – Performance of different approaches in terms of time in seconds and maximal GPU memory usage in MB when testing on a single image of the CLICIDE dataset

| | *Mean Precision@1* | | *Mean Average Precision* | |
|---|---|---|---|---|
| | *CLICIDE* | *GaRoFou* | *CLICIDE* | *GaRoFou* |
| *SIFT* | 70.08 | 78.82 | N/A | N/A |
| *AlexNet IM* | 57.58 | 76.63 | 0 | 0 |
| *ResNet-152 IM* | 64.24 | 75.54 | 0 | 0 |
| *Gordo et al [5]* | 90.30 | 95.65 | 0 | 0 |
| *Gordo et al [5] (multi-resolution)* | 92.73 | 95.65 | 0 | 0 |
| *AlexNet FT* | 0 | 0 | 0 | 0 |
| *ResNet-152 FT* | 0 | 0 | 0 | 0 |
| *Proposed AlexNet* | 0 | 0 | 0 | 0 |
| *Proposed ResNet-152* | 0 | 0 | 0 | 0 |

Table 5.3 – Evaluation results for the CLICIDE and GaRoFou datasets. The results are expressed in percentage points of Mean Precision@1

## 5.2 Fine-tuned filters

First, we present the results obtained by fine-tuning filters, as described in Section

## 5.3 Evaluation results

In this section, we present the results obtained on the presented datasets using our network as described in Section 3.4, as well as various other approaches.

Table 5.3 gives an overview of the results obtained. First, the baselines established in Section 4.4 are shown again, with the addition of the relevant results obtained by fine-tuning a classification network (Section 3.1) and our proposed network (Section 3.4). In addition to the mean precision@1, we show the mean average precision obtained by the different approaches. This metric is only indicative of

## 5.4 Additional improvements

In the evaluation results shown above, we consider each approach as if it provides a strong descriptor for each image, and we only evaluate the performance of this descriptor. In information retrieval, many approaches combine descriptors, or use augmentation techniques in order to improve the results provided by a single descriptor.

The following sections quickly describe these approaches and justify whether or not they may improve the results obtained here.

### 5.4.1 Combination of descriptors

Many approaches in information retrieval combine multiple descriptors in order to achieve better results. One of the major motivations for our research is precisely to avoid this combination by using an end-to-end approach based on optimizing a single descriptor globally. On the other hand, combining descriptors always optimizes the individual descriptors for a specific task and only later combines them for more abstract or difficult task. Since we want to avoid choosing many different descriptors to combine, we do not explore this approach.

### 5.4.2 Query expansion

Gordo et al [6] use an approach named query expansion to achieve better results. This approach essentially performs multiple queries: first, the descriptor of a query image is used as-is. Then, the descriptor is combined with the descriptors of the top $k$ retrieved results, by simply adding the descriptors together. This new descriptor is used to perform a second query, which gives the final result.

This technique was introduced by Chum et al [3] and is especially useful when the descriptor of the query can be spatially matched against the returned descriptors. This eliminates some of the returned descriptors from the combined descriptor used to perform a second query. However, as CNNs produce global descriptors of images, this spatial matching cannot be performed.

Furthermore, we do not expect query expansion to provide any major improvements in our research problem, since we expect to have very few images returned. This means the only plausible value of $k$ would be $k = 1$. However, if the best matching descriptor of the first query already matched, the second query cannot improve the result and if it does not match, it is unlikely that the second query would match. Overall, query expansion may improve mean average precision, but it is unlikely to improve mean precision@1.

### 5.4.3 Database-side feature augmentation

Gordo et al [6] use another approach to improve results. This approach is introduced by Turcot et al [31] and Arandjelovic et al [1].

The idea is to combine the descriptors of the reference images in order to form better database-side descriptors. Every reference descriptor is simply replaced by a combination of itself and the $k$ nearest neighbors. This combination is computed as a weighted sum, weighted by the rank of the neighbors with respect to $k$ (the closest neighbor has the highest weight and the $k$-th neighbor the lowest). The neighbors can be pre-computed once all descriptors have been computed for the reference images, by comparing each image to all others. This can be

done efficiently in our case where descriptors can be compared to other descriptors using a matrix multiplication.

However, it is difficult to choose the value for $k$, as it should be low enough to not include many irrelevant images for the descriptors, but high enough for the technique to have an impact at all.

### 5.4.4 Instance-averaging

Since we have the labels of the reference images available, we propose an approach similar to database-side feature augmentation in order to improve the results: for each instance, the descriptors of all images representing this instance are combined by averaging the individual descriptors representing that instance, then normalizing again.

This reduces the number of reference descriptors: instead of having as many reference descriptors as reference images, it leaves only as many reference descriptors as there are instances in the dataset.

Table **??** shows the results obtained using this additional technique.

# — 6 —
# Conclusion

## 6.1 Conclusions from our research

During our research, we analyzed state-of-the-art approaches in image classification and image retrieval and their application to the instance search problem. We found that these approaches struggle with query images and reference images being at different scales. Additionally, the region proposals used by the state-of-the-art do not seem applicable to the problem of instance search.

We have presented a novel approach, which consists of two key steps. First, we leverage the concept of fully convolutional networks in order to perform classification training at different scales, without a heavy computational overhead. Second, we show that the fully convolutional network can be used to obtain region proposals without the need for an additional component in the network and training. This is particularly important, since region proposals are difficult to define manually in our research problem.

Finally, the proposed network keeps all the benefits of state-of-the-art approaches: it can be trained end-to-end and it produces an effective global descriptor, which can be compared using the dot-product. Additionally, it is modular in the sense that it can be built upon any type of CNN, pre-trained for classification. Furthermore, the proposed network is fast to evaluate and the training time for each dataset is reasonable.

Through multiple experiments on two datasets, we show that the descriptor obtained using our proposed network outperforms previous state-of-the-art approaches on the instance search task, while being just as memory-efficient and fast.

## 6.2 Future work

### 6.2.1 Few-shots classification

While we explored different ideas in this report, some have not been considered yet. For one, a paper by Kaiser et al [13] was presented earlier this year, which focuses on few-shots classification and achieves state-of-the-art results.

As noted before, our research problem shares many similarities with the few-shots classification problem. Furthermore, this paper presents a method of combining the triplet loss and cross-entropy loss in order to organize a memory of so-called rare events. This is similar to

our proposed approach and may be adapted to our research problem. At the very least, it could provide significant insight to the instance search problem.

### 6.2.2 Conditional similarities

Another interesting paper presented earlier this year is by Veit et al [33]. It presents a method of capturing semantically different notions of similarity. The main idea is that there are types of similarity (color, category, etc.) and thus images are embedded into many different subspaces, which capture these different types of similarity. Finally, they present a method to jointly learn a global embedding, made up of different sub-embeddings for the sub-spaces representing different similarities.

This idea could be interesting in the instance search problem, as it may allow us to learn different types of categories first and then produce an embedding based on these different categories. For instance, a painting could be described by its main tone (dark, bright, etc.), its art movement (impressionism, dadaism, etc.) or simply its size (big, small, squared, wide, high). Each of these notions may be easily described individually and a joint learning of these different types of descriptors could provide significant improvements over learning a single, global descriptor for each image.

# — A —
# Appendix

# Bibliography


[1] Relja Arandjelović and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2911–2918. IEEE, 2012.

[2] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. Neural codes for image retrieval. In *European conference on computer vision*, pages 584–599. Springer, 2014.

[3] Ondrej Chum, James Philbin, Josef Sivic, Michael Isard, and Andrew Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[4] Yoav Freund and Robert E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.

[5] Albert Gordo, Jon Almazán, Jerome Revaud, and Diane Larlus. Deep Image Retrieval: Learning Global Representations for Image Search. In *Computer Vision – ECCV 2016*, pages 241–257. Springer, Cham, October 2016.

[6] Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus. End-to-end learning of deep visual representations for image retrieval. *arXiv preprint arXiv:1610.07940*, 2016.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*, December 2015. arXiv: 1512.03385.

[8] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[9] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[10] Gao Huang, Zhuang Liu, Kilian Q. Weinberger, and Laurens van der Maaten. Densely Connected Convolutional Networks. *arXiv:1608.06993 [cs]*, August 2016. arXiv: 1608.06993.

[11] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167 [cs]*, February 2015. arXiv: 1502.03167.

[12] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometry consistency for large scale image search-extended version. 2008.

[13] Łukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. Learning to remember rare events. *arXiv preprint arXiv:1703.03129*, 2017.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[15] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[16] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[17] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[18] Andrej Mikulik, Michal Perdoch, Ondřej Chum, and Jiří Matas. Learning Vocabularies over a Fine Quantization. *International Journal of Computer Vision*, 103(1):163–175, May 2013.

[19] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[20] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[21] Maxime Portaz, Johann Poignant, Mateusz Budnik, Philippe Mulhem, Jean-Pierre Chevallet, and Lorraine Goeuriot. Construction et évaluation d'un corpus pour la recherche d'instances d'images muséales.

[22] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

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

[24] Amaia Salvador, Xavier Giro-i Nieto, Ferran Marques, and Shin'ichi Satoh. Faster R-CNN Features for Instance Search. *arXiv:1604.08893 [cs]*, April 2016. arXiv: 1604.08893.

[25] Bernhard Scholkopf, Kah-Kay Sung, Christopher JC Burges, Federico Girosi, Partha Niyogi, Tomaso Poggio, and Vladimir Vapnik. Comparing support vector machines with Gaussian kernels to radial basis function classifiers. *IEEE transactions on Signal Processing*, 45(11):2758–2765, 1997.

[26] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[27] John Shawe-Taylor and Nello Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[28] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556 [cs]*, September 2014. arXiv: 1409.1556.

[29] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *arXiv:1602.07261 [cs]*, February 2016. arXiv: 1602.07261.

[30] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. Particular object retrieval with integral max-pooling of CNN activations. *arXiv:1511.05879 [cs]*, November 2015. arXiv: 1511.05879.

[31] Panu Turcot and David G. Lowe. Better matching with fewer features: The selection of useful features in large database recognition problems. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 2109–2116. IEEE, 2009.

[32] Andrea Vedaldi and Andrew Zisserman. Efficient additive kernels via explicit feature maps. *IEEE transactions on pattern analysis and machine intelligence*, 34(3):480–492, 2012.

[33] Andreas Veit, Serge Belongie, and Theofanis Karaletsos. Conditional Similarity Networks. *arXiv:1603.07810 [cs]*, March 2016. arXiv: 1603.07810.

[34] Kilian Q. Weinberger, John Blitzer, and Lawrence Saul. Distance metric learning for large margin nearest neighbor classification. *Advances in neural information processing systems*, 18:1473, 2006.

[35] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *arXiv:1411.1792 [cs]*, November 2014. arXiv: 1411.1792.