DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2018

Unsupervised Representation Learning with Clustering in Deep Convolutional Networks

MATHILDE CARON

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Unsupervised Representation Learning with Clustering in Deep Convolutional Networks

MATHILDE CARON

Degree Programme in Computer Science and Engineering
Date: September 2017 / January 2018
Supervisor: Atsuto Maki
Supervisors at FAIR: Piotr Bojanowski,
Matthijs Douze, Armand Joulin
Examiner: Hedvig Kjellström
School of Electrical Engineering and Computer Science

Abstract

This master thesis tackles the problem of unsupervised learning of visual representations with deep Convolutional Neural Networks (CNN). This is one of the main actual challenges in image recognition to close the gap between unsupervised and supervised representation learning. We propose a novel and simple way of training CNN on fully unlabeled datasets. Our method jointly optimizes a grouping of the representations and trains a CNN using the groups as supervision. We evaluate the models trained with our method on standard transfer learning experiments from the literature. We find out that our method outperforms all self-supervised and unsupervised state-of-the-art approaches. More importantly, our method outperforms those methods even when the unsupervised training set is not ImageNet but an arbitrary subset of images from Flickr.

Sammanfattning

Detta examensarbete behandlar problemet med oövervakat lärande av visuella representationer med djupa konvolutionella neurala nätverk (CNN). Detta är en av de viktigaste faktiska utmaningarna i datorseende för att överbrygga klyftan mellan oövervakad och övervakad representa- tionstjänst. Vi föreslår ett nytt och enkelt sätt att träna CNN på helt omärkta dataset. Vår metod består i att tillsammans optimera en grup- pering av representationerna och träna ett CNN med hjälp av grupper- na som tillsyn. Vi utvärderar modellerna som tränats med vår metod på standardöverföringslärande experiment från litteraturen. Vi finner att vår metod överträffar alla självövervakade och oövervakade, toppmoderna tillvägagångssätt, hur sofistikerade de än är. Ännu viktigare är att vår metod överträffar de metoderna även när den oöverva- kade träningsuppsättningen inte är ImageNet men en godtycklig del- mängd av bilder från Flickr.

Contents

Chapter 1

Introduction

1.1 Context and motivation

The goal of this master thesis is to develop a model for unsupervised training of deep convolutional neural networks (CNN). Indeed, recent advances in computer vision rely on the use of CNN architectures trained on large manually labeled datasets. For example, ImageNet is a dataset of 1.2M images annotated with 1000 object categories [1]. It has been shown that features obtained from networks trained on ImageNet transfer very well to other computer vision tasks such as fine-grained classification [2, 3], object detection [4], and segmentation [5].

A typical setting [6] consists in pre-training a CNN so that it learns good general-purpose features and then transferring these features to a domain where labels are scarce. This scarcity means that it is impossible to train a neural network from scratch because it has too many parameters. For this setting, the fully supervised part does produce a relatively high level representation. The representations can be used to train a shallower classifier (a logistic regression for example).

In contrast, in our setting we start from a domain where we do not have any annotation. The only information that can be exploited is the distribution of the images in the pixel space. This setting may seem artificial, after all there *are* large and richly annotated datasets like ImageNet. We argue that this setting is interesting both from a theoretical point of view and in practical cases. Let's imagine a new image capture modality appears, for instance RGB + a weak sense of depth (this is what dual-lens cameras on mobile phones produce). It would be extremely tedious to painfully select and re-annotate the million of ImageNet images. This is where fully unsupervised representation learning becomes useful.

At the same time, unsupervised learning might be the right way towards learning better models as it has been argued that human performance in fully supervised image classification and object detection has been reached or is about to be reached [7, 8]. Moreover, an unsupervised approach also should suffer less from the inevitable bias introduced by collecting supervised data. For example ImageNet is a clustered dataset and contains a wide variety of dog breeds. Large unlabeled datasets with wide coverage are easily available on the Internet. The basis of this work is to assume that unsupervised models trained on large datasets such as YFCC100m [9] might provide better representations than supervised models.

1.2 Social and Ethical Impacts

The goals pursued by this master thesis are first and foremost purely scientific. On the other hand, many technological applications could ensue from the implementation of a successful method to pretrain CNN on large unlabeled datasets. It is the duty of the society, the governments, and the researchers themselves to monitor and remain vigilant about the nature of the applications of artificial intelligence algorithms. An example of application that could be allowed thanks to the objectives pursued in this master thesis concerns image recognition in the medical images domain. Indeed, the features learned with training a network on the fully supervised dataset ImageNet are not likely to transfer well to image recognition tasks in medical images. This is due to the fact that the domain of pretaining (in the case of ImageNet: objects) and the domain of the task of application (medical images) are different. An unsupervised training method would allow to acquire good general-purpose features in the specific domain of medical images without requiring the design of a large-scale fully-supervised dataset of medical images. Finally, we have paid attention in this work to use only publicly available data.

Chapter 2

Related work

2.1 Unsupervised learning

Several fully unsupervised learning approaches that attempt to learn good visual features have been proposed in the past few years. In particular, Dosovitskiy et al. [10] consider each sample as a class on its own to train a CNN architecture. Then each class is augmented with data transformation applied to the corresponding sample. However, albeit theoretically interesting, this approach can hardly scale to large datasets. Moreover, Dosovitskiy et al. [10] use shallow custom-tailored architectures while we use deeper standard architectures.

Generative models. In contrast to our work, generative models learn a parametrized mapping from a predefined latent space to the image space. This allows the generation of new images. In particular, the mapping can be obtained with autoencoders [11, 12, 13]. Basically, a convolutional and a deconvolutional networks are trained together with a reconstruction objective. In other words, the composition of the two networks has to reconstruct the input. However, performing well on this task does not necessarily require the networks to acquire a semantical representation of the images.

On the other hand, other works have focused on Generative Adversarial Networks (GANs) [14]. A GAN is made up of two neural networks, namely the generator and the discriminator, that compete against one another. The generator produces images from vectors of the latent space. Then, the task of the discriminator is to predict the provenance of an image: dataset or generator. This way, the generator is encouraged to produce realistic images in order to fool the discriminator. Interestingly, some experiments on GANs [15] have shown that the latent space captures semantic variations in the data distribution. The representations learned by the generator and the discriminator can be transfered to other visual tasks. However, the performances are relatively disappointing. Furthermore, Donahue et al. [16] and Dumoulin et al. [17] approaches both add an encoder to the original GAN model. The purpose of this encoder is to predict the semantic representations of the latent space from images. In other words, it learns the inverse mapping of the generator and allows to produce competitive visual features. Unlike Dumoulin et al., Donahue et al. evaluate the quality of the features learned by the encoder by reporting performances on standard computer vision transfer tasks (ImageNet classification and PASCAL VOC classification, detection, segmentation [18]).

Clustering and representation learning. A Self-Organizing Map [19] allows to associate examples with prototypes arranged in a low dimensional regular structure. These prototypes are trained with competitive learning to fit at best the topological properties of the input space. This provides a simple representation of the data without any supervision. For data mining purpose, Vesanto et al. [20] propose to cluster the prototypes instead of the data directly. In the same line, our work performs clustering on intermediary representations instead of the raw images directly.

Clustering can be used as a means of regularization to train CNNs. Liao et al. [21] benchmark clustering of different natures at different levels in the network. Spatial clustering corresponds to the grouping of the pixels in a given layer over all localizations and examples. On the other hand, channel co-clustering allows to group the different feature maps (channels) in a given layer across examples. Closer to our work, sample clustering refers to the clustering of the set of representations given by a CNN over the whole dataset. However, unlike our method, the clustering objective is used as a regularization term and is not sufficient to learn the network weights. Liao et al. [21] apply their method on unsupervised learning with autoencoder by adding this clustering regularization term to a standard autoencoder loss. They evaluate the performance of this model on relatively small datasets.

Some methods rely explicitly on a clustering loss [22, 23] to learn CNN features and image clusters. Similar to our approach, the approach of Yang et al. [23] iterates over a clustering and a training stages. The former focuses on the optimizing of a grouping of the representations given by the network. The latter corresponds to the training of the network as for a traditional supervised setting with the groups as supervision. Yang et al. [23] use agglomerative clustering which has been shown recently not to be the most effective for large-scale datasets [24]. This clustering method starts from many very small clusters and merges them progressively. In addition, they evaluate their method on small datasets while we are aiming at large-scale datasets.

Discriminative clustering. Discriminative clustering optimizes a clustering of the data that, if used as pseudo-labellisation to learn a classifier f, produces the maximum of separability. The separability is evaluated through the value of the loss for the optimal classifier. In particular, f can be a linear classifier associated with a ridge regression loss [25]. Convex relaxations are operated to solve the problem. It is also essential to add some constraints to avoid the collapsing problem where all the examples are assigned to the same cluster. This framework has been applied to solve object discovery tasks [26, 27].

This work is largely inspired by the work of Bojanowski et al. [28] that uses a discriminative clustering approach to train deep convolutional neural networks. The classifier f is a CNN and the clusters are a set of fixed low dimensional targets uniformly arranged on a sphere in the feature space. Then, the method iterates between training the network with the targets as labels and assigning each representation to a cluster (ie target). Their loss aims at maximizing the difference between each image representation.

2.2 Self-supervised learning

Self-supervised learning [29] methods replace the labels annotated by humans by "pseudo-labels". This is done by finding a pretext task for the network to solve, exploiting signal contained in auxiliary information or in the raw pixels directly. Performing well on the pretext task shall require a semantic understanding of the scene and objects present in the image. This way, the network is encouraged to learn high level features. However, contrary to our work, this approach is somewhat domain dependent, as it requires expert knowledge for the careful definition of the pretext task. In the last few years, several pretext tasks have been proposed. We give a non exhaustive overview of some of these in the remaining of this subsection. Like us, most of these methods assess the semantic quality and transferability of their learned features on standard PASCAL VOC generalization tasks [18].

Pathak et al. [30] leverage motion in videos to produce unsupervised segmentations. Then, a CNN is trained to predict these segmentations from images. Pathak et al. use 205 000 videos from Yahoo/Flickr 100-million dataset [9] that provide them 1.6M images. Their experiments show that their method works especially well for multi-task transfers. They also report the results for PASCAL VOC 2012 object detection in a low shot learning setting, simulating the case where training data to fine-tune is scarce.

Another pretext task is to predict color channels from luminance [31]. The network learns a probability distribution over the color space for each pixel. Zhang et al. [31] also introduce a generalization task on ImageNet: training linear classifiers on top of the different frozen convolutional layers. This evaluation framework is interesting to assess the evolution of the quality of the intermediary representations learned by the network. This setting has been re-used in the literature and we report later in this report our results for this experiment (Table 5.3). In the same vein as Zhang et al [31], it has been tried to predict one subset of the data channels from another one [32].

Pathak et al. [33] propose to train a CNN to generate an image region from its surrounding. An encoder captures the context of the image into a compact latent representation while a decoder uses that latent vector to fill in the missing image patch. The connection between these two networks is performed with a channel-wise fully connected layer that allows propagation of information across channels. A combination of two losses is used: a reconstruction loss that tends to give blurry patches since it is safer to predict the mean of the distribution; and an adversarial loss that encourages sharpest and more realistic results.

A Siamese neural network is the association of two or several identical subnetworks that share the exact same architecture and weights. Siamese networks allow to learn the weights of their subnetworks using a comparative measure instead of labels. This comparative measure can be a similarity function or a relationship between several comparable inputs. In contrast to our work, the following methods use Siamese networks.

Exploiting video consistency, Agrawal et al. [34] predict the camera transformation between pairs of images. Furthermore, an interesting insight of their work is to highlight that a useful visual representation shall be able to perform new tasks by learning from a few labeled examples (low-shot learning). In our work, we propose to evaluate our model in a low-shot setting as well (see Figure 5.1). However, because they perform transfer task on small datasets (MNIST) and because they use auxiliary information, their solution is expensive and impractical. Inspired by word2vec [35], Doersch et al. [36] use spatial context as source of supervisory signal. Given two patches in the same image, the task for a double-Siamese network is to find their relative position. Similar to our work, Doersch et al. [36] report results with pre-training their network on ImageNet [1] and on a subset of 2M images from Yahoo/Flickr 100-million Dataset [9]. They also work with two different architectures: AlexNet [37] and the deeper VGG16 [38]. Besides, an interesting insight of the work of Doersch et al. [36] is the visualization of 5 nearest neighbors of query images in the feature space. We notice that the patches retrieved by a randomly initialized network contain some signal and are not totally random (wheels or legs for example). It supports our assumption that there is a weak signal to exploit in untrained networks. The good performance of random convolutional neural networks is intimately tied to the convolutional structure which corresponds to a strong prior on the input signal. We also report nearest neighbors visualization for our model later in the report (Figure 5.3). In the same vein as Doersch et al. [36], Noroozi et al. [39] propose to train a 9-Siamese network to solve jigsaw puzzles. The task of re-organizing nine shuffled patches cropped from an image is difficult and should force the network to learn feature mappings of objects part and their correct spacial arrangement. They introduce a transfer framework on ImageNet classification that consists in freezing the N first convolutional layers and retraining from scratch the remaining ones. We report results with this evaluation setting (Table 5.2).

Li et al. [40] build a graph of nearest neighbors (NN) on SIFT [41] + Fisher Vector descriptors in order to infer positive and negative pairs of images. The former come from cycle consistency in the k-NN graph while the latter come from large geodesic distances (accumulated weights along the shortest path between the two considered nodes). The task for the Siamese network is then to learn an embedding in which images semantically similar (positive pairs) are close while images semantically different (negative pairs) are distant according to the similarity function (cosine distance). Furthermore, other methods have used metric learning with a triplet-Siamese network [42]. For example, Wang et al. [43] leverage temporal coherence in videos. After mining patches of interest in 100K Youtube videos, they pick a positive matching patch by tracking a given patch along thirty frames in a video. A third negative patch is associated to the pair (randomly or with more sophisticated techniques), in order to feed a triplet Siamese network. Similar to the previous approach [40], the ensemble of networks has to learn an embedding in which the two positive patches are close relatively to the two negative ones.

Finally, the features learned by all the previous diverse methods perform in a comparable range on standard transfer tasks. However, the nature of what the features capture and focus on is likely to differ from a method to another. This way, several strategies for combining multiple cues have been explored recently [44, 45]. Wang et al. [44] propose to construct an affinity graph with two types of edges: inter instance invariance (similar viewpoint) and intra instance invariance (similar instance). The former are found with context prediction [36] while the latter are mined with tracking a patch along thirty frames following the same framework as Wang et al. [43]. From this affinity graph they infer positive and negative pairs that feed a Siamese network. Doersh at al. [45] also aim at leveraging the diversity of the learned representation by combining different pretext tasks. They use a form of regularization that encourages the network to separate features useful for different tasks. In other words, the network determines which combination of layers to use for each of the pretext tasks. Unlike most of the concurrent approaches that use AlexNet networks, they report results with the very deep ResNet-101 architecture [46].

2.3 Large-scale learning

In this work, we are aiming at leveraging the wide coverage and diversity inherent in large datasets. Indeed, it has been shown that more general visual features could be obtained by pre-training networks on very large-scale sources of labeled data using human made annotations [47] or metadata, like hashtags [48] or geolocalization coordinates [49]. In particular, Joulin et al. [48] trains CNNs on the large-scale dataset YFCC100m [9] in a weakly supervised fashion. Captions and hashtags on images provide noisy labels. Their work gives useful insights for this thesis. For example, their experiments show that best network architectures for supervised and unsupervised learning might not be the same. Also, they train their network by sampling data uniformly over targets to make sure that classes with a lot of examples do not dominate the learned features. Finally, they show that roughly 20 millions images from YFCC100m are required to reach performances of ImageNet.

The work of Douze et al. [50] tackles the problem of propagating labels to a large dataset when only a few labeled examples are provided. They designed an efficient method that scales up to datasets of an order of magnitude of several hundreds of millions. Their approach relies on the use of a k-NN graph of the data representations given by a CNN. However, contrary to our work, this method does not focus on the training of the CNN to acquire high-level features but uses a pre-trained (with supervision) CNN to extract descriptors from images.

Very useful for our work, Douze et al. [24] propose a benchmark of the performances of different clustering techniques. They use the unlabelled images of YFCC100m dataset [9] as distractors to augment other annotated datasets. These latter allow them to report quantitative performances such as recall and precision. They conclude that Power Iteration Clustering [51] (graph clustering technique) and k-means show the best performances on YFCC100m dataset [9].

2.4 Power Iteration Clustering

Power Iteration Clustering (PIC), introduced by Lin et al. [51], is a method that allows to cluster graphs. A graph is often represented by an affinity matrix A of size $n * n$ where n is the size of the dataset. A coefficient A_{ij} in this matrix is a measure of how close or similar the nodes i and j are. For example, a Gaussian kernel on ℓ^2 distances between nodes is often used. Standard spectral clustering methods rely on a matrix computed from A, referred to as *Laplacian matrix* L. Thresholding (with k nearest neighbors for example) can be used to make A sparse. Spectral graph theory [52] gives us that the k minimal eigenvectors (associated with minimal eigenvalues) of L are the solution of the Ncut problem (measure on a partition of graph nodes in k clusters). Indeed, the data projected in the subspace spanned by the k smallest eigenvectors are well separated.

PIC focuses on A directly instead of L. We notice that the minimal eigenvectors of L correspond to the maximal ones of the affinity matrix. Generally speaking, it may not seem interesting to use the eigenvector associated with the maximal eigenvalue of A because it is constant. As a matter of fact, to compute it, a common method is to multiply several times a vector to the affinity matrix [51] (and divide by a normalizing constant). The idea of Lin et al. [51] is to stop the iterations before convergence to the constant vector. They show that this low-dimensional embedding (1D) allows to cluster efficiently the data. In this work, we use a simpler variant of the Power Iteration Clustering (described in details by Douze et al. [24]).

Chapter 3

Method

3.1 Notations & specified problem definition

We are given a training set $X = (x_1, x_2, ..., x_n)$ of examples. We call features $f(X)$ the set of representations produced by a certain embedding f. For the remaining of this report, X is a set of images and f is a CNN parametrized by the weights Θ: f_Θ. In the context of supervised classification, one learns the weights Θ of a CNN by using the labels $Y = (y_1, y_2, ..., y_n)$ associated with the examples X. After defining a loss function $\ell(f_\Theta(x_i), y_i)$ on image label pairs, one minimizes it with respect to Θ using stochastic gradient descent and backpropagation [53, 54, 55]:

$$\min_\Theta \frac{1}{n} \sum_{i=1}^{n} \ell(f_\Theta(x_i), y_i). \tag{3.1}$$

In our unsupervised setting, Y is not given in the data. **How can we learn the weights Θ without knowing the labels Y?**

3.2 Proposed approach

With Gaussian random weights Θ, a CNN f_Θ does not constitute good features *per se*, in the sense that these are not discriminative for object classes for example. However, its performance (as reported by Zhang et al. [31]) on standard transfer tasks is, albeit low, far above chance. For example, training a linear classifier on top of the last convolutional layer of a randomly initialized AlexNet [37] gives 12% accuracy while the chance is 0.1%. The key idea of this thesis is to exploit this weak signal, use it to train the CNN and achieve better discriminative properties.

The approach we propose is to iterate over clustering the set of features $f_\Theta(X)$ in order to produce pseudo-labels Y and optimizing Θ using Y. In other words, after every clustering stage, we attribute a label y_i to each data x_i:

$$\forall x_i \in X, \quad y_i = \sum_{k=0}^{C} \mathbb{1}_{f_\Theta(x_i) \in C_k} k.$$

Then we train the network with X, Y and a logistic loss ℓ according to (3.1).

A schematic layout and a brief algorithm of our method can be seen in Figure 3.1 and Algorithm 1. We use two different clustering algorithms: either k-means or PIC [51]. There are very different approaches but turn out to give overall similar performances. With k-means approach, each point has an influence on the whole dataset whereas with PIC this influence is thresholded to the k-NN of the point. Moreover, unlike k-means, PIC does not require to set beforehand the number of clusters.

Algorithm 1 Our approach.

```
Data: dataset $X$
for epoch $\in [1, N]$ do
    Compute the representations $f_\Theta(X)$ for the whole dataset $X$
    Cluster $f_\Theta(X)$
    Assign to each data $x$ the id of the cluster it belongs to
    Train the network $f_\Theta$ with clusters as labels and a logistic loss $\ell$
end for
```

Joint optimization. Jointly learning the pseudo-labels Y and the parameters Θ leads to the trivial solution, where all the images are in the same cluster [56]. Typical solutions to this issue is to add constraints [25] or penalties on the number of images per cluster [57]. A simpler solution is to both reject empty clusters in k-means and sample per pseudo-labels instead of images when learning the parameters.

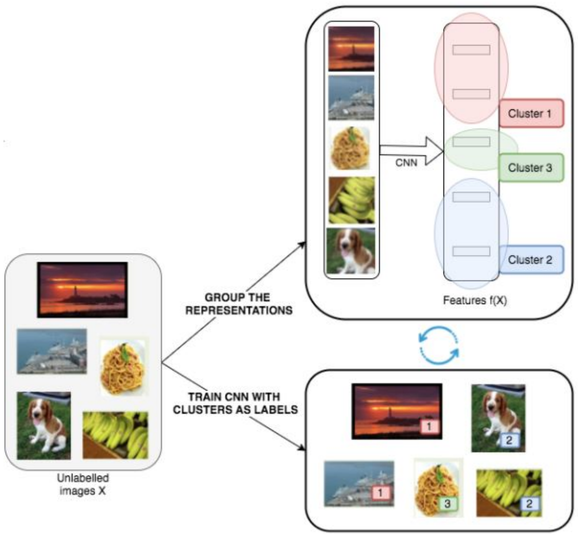


Figure 3.1: Schematic layout of our approach. We iterate over grouping the representations provided by a CNN and training it with the clusters as supervision.

Rejecting empty clustering ensures that the trivial solution is unattainable. In practice, we avoid empty clusters during the optimization of k-means by assigning to an empty cluster half of the images assigned to a randomly picked non-empty cluster. Sampling per classes avoids the domination of a single cluster that would make the classification problem trivial (i.e., always predict the dominating pseudo-labels). This strategy is commonly employed in problems where the label distribution is severely skewed [48].

3.3 Implementation details

Sobel filter. We choose to add a Sobel filter in input of our network. Indeed, as observed in other unsupervised works [28, 58], using the gradients of images as input performs better than using the raw RGB directly. This is in spite of the fact that the first convolutional level could easily integrate it. Our observations suggest that RGB input focuses too easily on strongly dominant colors. Indeed, at the very beginning of our training method, the clustering is performed on highly imperfect representations induced by a network with random weights. These representations are dominated by low-level characteristics such as colors. In practice, we observed that we were basically clustering by color. To avoid training and clustering on colors we choose to feed the network with gradient images. In the remaining of this report, unless specified otherwise, all results are performed with a Sobel filter preprocessing of the input.

Pre-training datasets. We train our model with two large scale datasets: either ImageNet [1] or YFCC100m [9]. We both considered them fully unlabeled. ImageNet counts a total of 1 281 167 images uniformly distributed into 1000 classes. Thus, it is by design a clustered dataset. On the other hand, YFCC100m is a publicly available dataset from the photo-sharing website Flickr, made up of roughly 99 millions images. In this report, *YFCC100m* and *Flickr* refer to the same dataset. In practice, we sample random subsets from this dataset and never work on the 99 millions at once.

Train the CNN. We choose a standard AlexNet [37] architecture for our CNN. It is composed of five convolutional layers with, in order, 96, 256, 384, 384 and 256 filters; and of three fully connected layers. We remove the Local Response Normalization layers and use batch normalization [59]. Except in Section 4.5 where we benchmark different architectures, we use AlexNet architecture throughout the report. We choose to focus particularly on AlexNet in our work because it allows us to be comparable with state-of-the-art concurrent approaches.

We choose to use a multi-class logistic loss ℓ to train the CNN. It allows Y to be a set of classes instead of prototypes. It is especially interesting for the PIC version of our method because this clustering method does not directly provide representatives of clusters (unlike centroids in k-means). Moreover, the cross entropy function forces the probability of the correct class to be be high *in comparison to* the ones of the other classes.

We use standard image preprocessing and data augmentation [37] employed with the original AlexNet architecture [37]. More precisely, we train the network on random crops of size 224*224 from the images. A benchmark of the different types of data augmentation we tried is proposed at Section 4.3. We use dropout [60], a constant learning rate, a ℓ^2 penalization of the weights Θ and a momemtum of 0.9. The size of a mini-batch is 256. As mentioned before, each mini-batch is randomly sampled uniformly over targets. In practice, we pick a cluster at random (with the same probability for each cluster). Then, an example corresponding to this cluster is sampled. It is especially useful if the clusters are unbalanced to prevent big clusters from dominating the learning.

We call *epoch* the training cycle between two reassignments of clusters. In practice it corresponds to a number of updates of Θ equal to the cardinal of the training dataset divided by the batch size for models trained on ImageNet and three times this quantity for models trained on YFCC100m [9]. Indeed, we choose to perform more training steps between two reassignments of clusters in YFCC100m [9] version of our method. Due to our sampling strategy, some examples might be seen several times or, on the contrary, not seen at all during one epoch. After each epoch, we re-set the last fully connected layer of the CNN.

Group the representations. First, we get a set of representations $f_\Theta(X)$ from central crops (size $224 * 224$) for the whole dataset. In practice f_Θ is the intermediate representation given by the penultimate layer of the CNN. We observed that using the representations before the activation function gives better performance because the representations tend to become very sparse (see Section 4.2). We use the k-means and k-NN search implementations available in FAISS [61].

Chapter 4

Detailed analysis

In this chapter, we detail some major steps that have led us to the design of our final approach.

4.1 k-NN graph of representations

Definitions. We propose to generate a graph $G = (V, E)$ of nearest neighbors (NN) from image descriptors. V is the set of representations $f_\Theta(X)$ and E is the set of edges, defined with the k NN relationship. The NN are computed with the Euclidean distance between the representations $f_\Theta(X)$. We arbitrary set the number k of NN to 10. The Faiss [61] library allows us to build efficiently k-NN graphs on datasets of large magnitude (up to 99 millions images for Flickr dataset [9]). Our PIC algorithm implementation relies explicitly on this graph. Moreover, we find that interesting to observe the evolution of some statistics of this graph throughout our training routine.

First of all, to assess the quality of the representations we introduce a supervised measure on any k-NN graph of ImageNet descriptors. For each class C_k, we report the proportion of edges (i, j) between two nodes from C_k with respect to the total number of edges associated with C_k:

$$\frac{\#\{e = (i, j) | i \in C_k, j \in C_k\}}{\#\{e = (i, j) | i \in C_k\}}.$$

We call this measure *ratio edge intra-class*. The higher the ratio edge intra-class the better. Indeed, it implies that there is a high proportion of edges between nodes semantically similar among all edges.

Assess the representations of a CNN with the ratio edge intra-class. We plot in Figure 4.1 the ratios edge intra-class computed for a fixed random subset of 250 classes of ImageNet. We compare the representations obtained with three different AlexNet CNNs: trained with supervision on full ImageNet, trained with supervision on the 750 classes of ImageNet not used for the evaluation, and with random weights. All of them take RGB inputs. We observe in Figure 4.1 that, unsurprisingly, the best model is the one that had been trained on full ImageNet. The model trained on 750 classes performs remarkably well, given that the classes it is evaluated on hadn't been seen during the training stage. It shows that supervised training on ImageNet provides good features that generalize well to new classes. Also, we notice that the mean of the ratio edge intra-class over the 250 classes in the case of a CNN with random weights is 3.0%, which is pretty low but still above the score of a random graph (0.4%). Moreover, the ratio edge intra-class is very unbalanced across the different 250 classes. In particular, there are some classes with a ratio edge intra-class quite high for the random model. For example, the class rapeseed has a score of 69%, the class yellow ladys slipper 51% and the class hammerhead 33%. More precisely, it means that, on the 10-NN graph induced by a *randomly* initialized network, there is 69% of edges between two rapeseed nodes among all the edges coming out rapeseed nodes. Rapeseed, yellow ladys slipper and hammerhead are examples of *easy* classes. Low-level features are sufficient to describe them and discriminate them from other classes. For example, the rapeseed class can be described with simple color and spatial attributes: yellow bottom part plus blue top part.

High density regions. Moreover, we tried to leverage high density regions within the k-NN graph. To do so, we focused on complete subgraphs, namely cliques. Thanks to Networkx library, we retrieved all the maximal cliques of the graph. After sorting and filtering the cliques, we trained a CNN with the cliques as supervision. However, only few samples among the whole dataset belonged to a clique. Thus, we were training the network with a few very similar images. The first trials gave very poor results and we chose not to pursue in that direction.

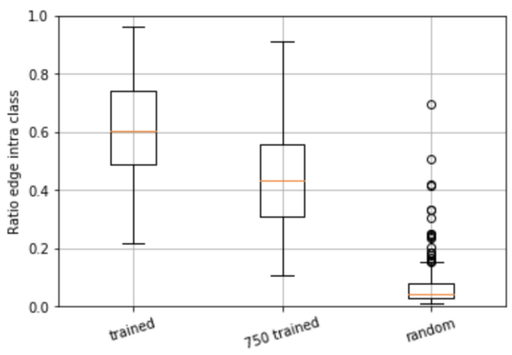


Figure 4.1: Ratio edge intra-class for different 10-NN graph. The orange line represents the median, the rectangles represent the first $Q1$ and third $Q3$ quartiles. If IQR is the interquartile range $(Q3 - Q1)$, then the upper whisker extends to last point less than Q3 + 1.5*IQR. The outliers are data that extend beyond the whiskers.

4.2 The descriptors

An important stage of our method is to compute a set of descriptors from the data, to be clustered thereafter. First, we transform the data with the CNN up to the penultimate layer, before the last fully connected layer and the activation unit. Then, we apply Principal Component Analysis (PCA) whitening to reduce the dimension of the descriptors to 256. Finally, we normalize the representations.

In this section, we focus on the specifics regarding the computation of this set of representations and we motivate some of the choices we made.

Get descriptors before activation. We compare the performance of our method with using intermediate representations at different layer of the CNN for different multi layer perceptron (MLP) architectures at the CNN head. Actually, as it can be seen of Figure 5.4 we benchmark three heads: *Full MLP* (FC - ReLU - FC - ReLU - FC), *2FC MLP* (FC - ReLU - FC) and *no ReLU* (FC - ReLu - FC - FC). We call FC a fully connected layer and ReLU the rectifier activation unit. In this experiment, we get for descriptors the representations obtained just before the last FC. We report as a measure of performance the ratio edge intra-class on a 10-NN graph built on the representations used for the clustering at each epoch.

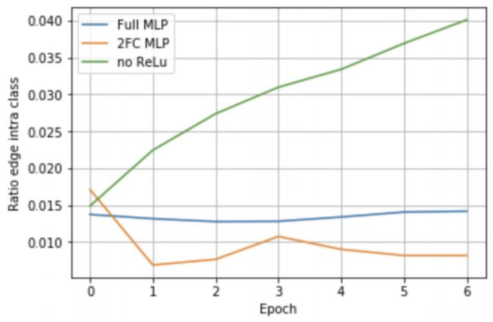


Figure 4.2: Ratio edge intra-class through a few epochs with PIC clustering. *Full MLP*: FC - ReLU - FC - ReLU - FC. *2FC MLP*: FC - ReLU - FC. *no ReLU*: FC - ReLu - FC - FC. FC: fully connected layer. ReLU: Rectifier activation unit.

First, we observe in Figure 5.4 that it gives poorer performance to use a reduced MLP (2FC MLP) compared to the regular AlexNet MLP composed of 3 FC (full MLP). Second, contrary to *Full MLP* and *2FC MLP* cases, the descriptors from *no ReLU* have not been modified with ReLU and contain negative and positive entries. The performance of the green curve (*no ReLU*) is much better than the two other ones. When the clustering stage is based on features after an activation unit, the ratio edge intra-class doesn't improve through the epochs. In the remaining of this report we use a full regular MLP with 3 FC for the head of our model trained with AlexNet and VGG16.

We wonder whether it is better to perform the clustering on descriptors for which activation function has been applied to or not. We report in Table 4.1 the performance on ImageNet classification when retraining the MLP from scratch with frozen parameters for the convolutional layers. The benchmarked models have been trained during 85 epochs with four different clustering techniques: PIC, k-means with $k = 100$, $k = 1000$ and $k = 10000$. For all of them, we compare the performance when the grouping is performed on representations *before* or *after* the ReLU.

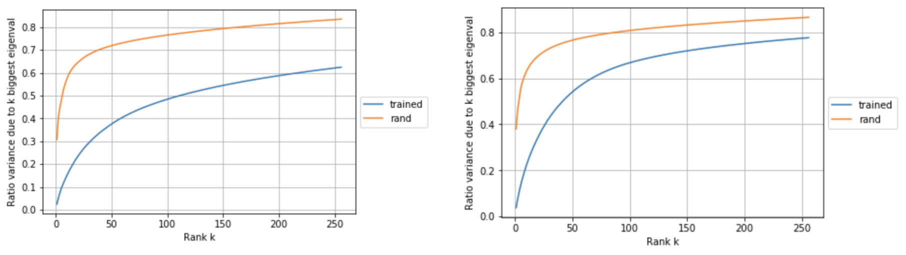
At it can be seen in Table 4.1 the performance, evaluated through ImageNet classification, is better with the grouping without activation except for the model trained with k-means 100. It can be explained by the fact that when a lot of clusters are present, the representations after ReLU tend to become very sparse. To conclude, we decide to get

Table 4.1: Performance on ImageNet classification after retraining the MLP from scratch. The convolutional layers are frozen and obtained from models pre-trained during 85 epochs. The models in the Table correspond to different versions of our approach. They differ in the clustering algorithm used and in the way of computing the descriptors (*before* or *after* activation).

	Before ReLU	After ReLU
PIC	21.9	33.2
k-means 100	33.5	30.2
k-means 1K	33.4	**33.8**
k-means 10K	31.1	33.1

descriptors before both the last fully connected layer and the activation unit.

PCA whitening. PCA allows to reduce the dimension of a set of data while preserving information. The data are projected in a subspace where each vector basis corresponds to a principal direction of variation of the data. In other words, if one projects the dataset on the first principal component, the variance of the resulting projections will be maximal.



(a) AlexNet (b) VGG16

Figure 4.3: Percentage of variance retained by the k (k up to 256) maximal eigenvalues.

In practice, we compute the eigenvectors of the covariance matrix of the data corresponding to the highest eigenvalues. For each rank k corresponding to the sorted eigenvalues, we plot (Figure 4.3) the cumulative sum of the eigenvalues up to λ_k, namely $\frac{\sum_{i=1}^{k}\lambda_i}{\sum_{i=1}^{N}\lambda_i}$, for a network trained with supervision and for a network with random weights. We observe that in Figure 4.3 that the maximal eigenvalue captures 30% of the variance of the data for a random AlexNet and up to 40% for a random VGG16. The maximal eigenvalue in the random case has too much influence, compared to the one in the trained network. We thus decide to whiten the data. In other words, we divide each component of the projected data by the square root of the eigenvalue corresponding to that component. This way, all the directions in the PCA space have the same variance.

4.3 Data augmentation

In our approach, once each data has been assigned to a cluster, we train the network in a conventional supervised manner. It is common to use data augmentation to do so [37].

Figure 4.4 shows the evolution of the performance, measured through the ratio edge intra-class averaged over the 1000 classes of ImageNet, during fifty epochs of training for different settings. These differ in the kind of data augmentation used. We benchmark three different flips: vertical, horizontal (both occurring with a probability of 0.5) and no flip. We also benchmark three different crops in the images: *Central crop*, a crop of fixed size with a random localization (*Random crop*) and a crop of random size (0.08 to 1.0 of the original size) and random aspect ratio (of 3/4 to 4/3 of the original aspect ratio) (*RandomSized crop*). We use the PIC version of our method during this experiment. The pre-training dataset is ImageNet.

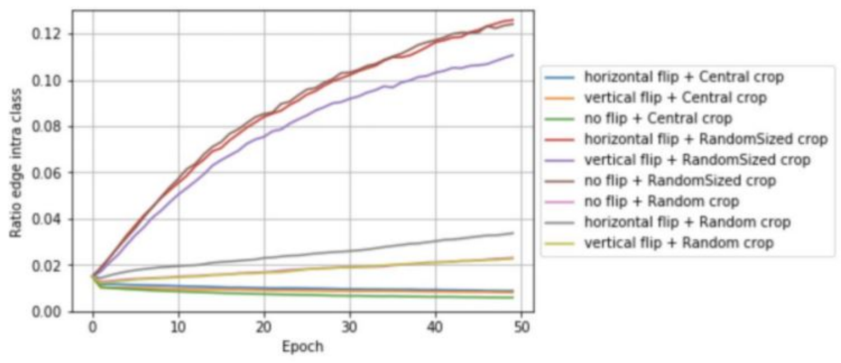


Figure 4.4: Ratio edge intra-class throughout 50 epochs with different kinds of data augmentation during the training of an AlexNet CNN.

We observe in Figure 4.4 we improve the ratio edge intra-class in the k-NN graph of descriptors only for the configurations with RandomSized crops. This way of doing data augmentation is thus the most effective for our method. Surprisingly, we notice that flipping doesn't play a key role and that we could remove it. What's more, vertical flip degrades the performance. One conclusion of this experiment is that data augmentation is crucial in our method. Other unsupervised approaches have been strongly relying on data augmentation [10].

4.4 Frequency of reassignment

In this subsection, we are interested in how frequently to reassign the clusters. A balance has to be found between training the CNN and getting the descriptors to perform a new partition. On the one hand, performing too much training steps between two reassignments would lead to overfitting to the partition. This can be problematic, especially since the clusters are imperfect. On the other hand, reassigning the clusters is quite costly since it involves, among other things, performing a full feed-forward pass for the whole dataset. As defined previously, an epoch corresponds to the training cycle between two reassignments of clusters. Thus, we conduct an experiment that aims at benchmarking different sizes of epoch.

On Figure 4.5, the blue curve (1) corresponds to the version of our method with an epoch size equal to roughly 5005 (size of ImageNet divided by the batch size) updates of Θ. Then, the three other cases correspond to epoch sizes multiple of 5005. For example, the red curve (10) corresponds to the case of reassigning the clusters every 50050 steps of CNN training. We report in Figure 4.5 the ratio edge intra-class averaged over the 1000 classes of ImageNet every 5005 updates of Θ. We perform this experiment with the PIC version of our method and with ImageNet as pre-training dataset.

First, we observe in Figure 4.5 that the ratio edge intra-class grow throughout the training for the four settings. Hence, our unsupervised training method improves the quality of the features provided by the CNN. As a matter of fact, the ratio edge intra-class on ImageNet reaches more than 15% after 100 epochs of training with an epoch size of 5005. Second, we notice that the ratio edge intra-class grows very

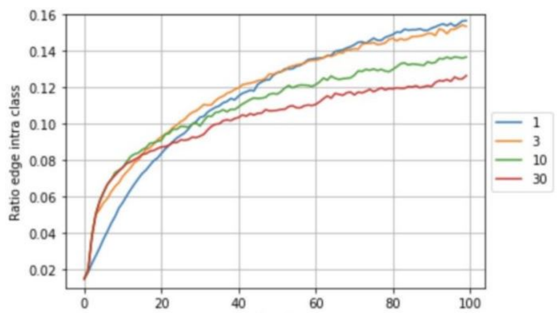


Figure 4.5: Ratio edge intra-class reported every 5005 training steps with different sizes of epoch: 1*5005, 3*5005, 10*5005 and 30*5005. The training is done with PIC, on ImageNet dataset.

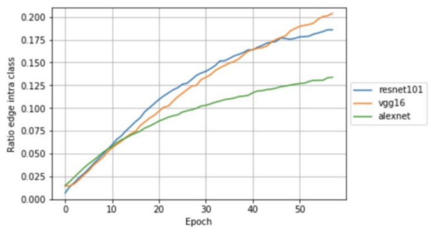
quickly when we let the network overfits to the first (very imperfect) clustering (red curve). After a short time, the performance starts to saturate. Reassigning the clusters allows then to boost the ratio-edge but it re-saturates rapidly afterwards. Thus, we observe that reassigning the clusters frequently (at each epoch) seems to be the best long-term setting when measuring the performance on the k-NN graph.

4.5 CNN architectures

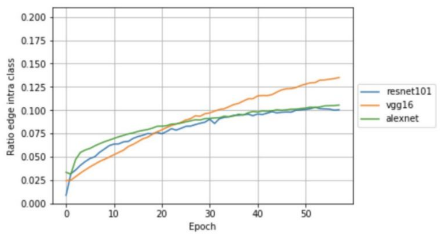
In this subsection, we benchmark three different architectures. We compare AlexNet to much deeper architectures, namely VGG16 [38] and ResNet-101 [46]. Those very deep architectures have been shown to perform remarkably well on classification ImageNet dataset with supervision. For example, VGG16 [38] reaches a top@1 accuracy of 73.4 and ResNet-101 [46] 77.4.

On Figure 4.6 one can see the evolution of the ratio edge intra-class on a 10-NN graph of representations with the three different architectures. The trainings are performed on ImageNet dataset, with clustering the representations with PIC algorithm. We observe in Figure 4.6 that the best performing architecture out of the three we tried is VGG16. Moreover, the performances with a ResNet-101 are quite disappointing in regard to the very good performance of this architecture in supervised learning. In particular, when not applying Sobel filtering to the inputs, (Figure 4.6(b)), AlexNet and ResNet-101 architectures are performing in a comparable range.

As a result, the best performing models on supervised and unsupervised learning might not be the same. Actually, the state-of-art architectures at the present time have been tailored to perform well on classification on supervised ImageNet. It could be a follow-up of this thesis to design a novel architecture dedicated to unsupervised learning.



(a) Sobel Input



(b) RGB input

Figure 4.6: Evolution of the ratio edge intra-class through 50 epochs of training for different architectures.

4.6 Power Iteration Clustering

We use the variant of PIC and the best performing combination of parameters given by Douze et al. [24]. In particular, we start by computing a graph W on the representations:

$$W_{ij} = \begin{cases} 1, & \text{if } j \text{ is one of the 5 NN of } i. \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

Then, we get an affinity matrix by symmetrizing W and applying a Gaussian kernel: $x \mapsto \exp(\frac{-x^2}{\sigma^2})$ where σ is a bandwidth parameter.

Singletons. We observed in practice that clusters obtained with our implementation of PIC tends to be very unbalanced, especially at the beginning of our training routine when the features are still bad. We usually end up with one dominant cluster and a lot of clusters made up of only one data (singletons). That is why we choose to sample the examples uniformly across clusters, to avoid representatives of the dominant cluster to appear too much during the training of the network. Moreover, we introduce a simple heuristic to redistribute singletons. A singleton is reallocated in the cluster of its closest non-singleton 5 nearest neighbors. If there are only singletons amongst them then the singleton remains a singleton. Actually, we observe Figure 4.7 that this heuristic is not essential for our method to work and we can more easily choose not to consider singletons during the training of the network. However, we prefer to keep this redistribution heuristic as it allows not to discard data.

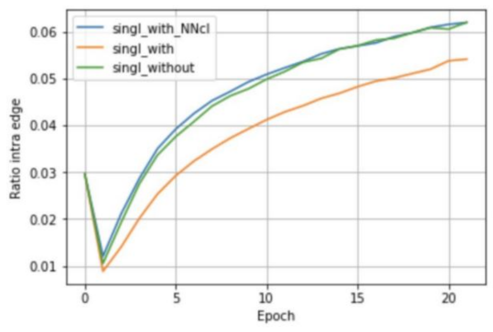
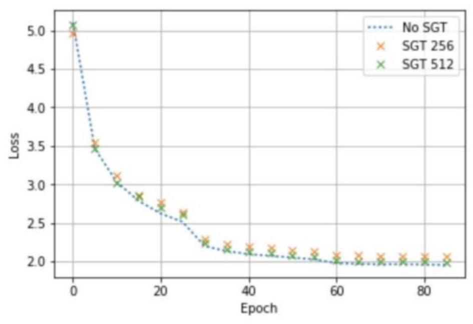


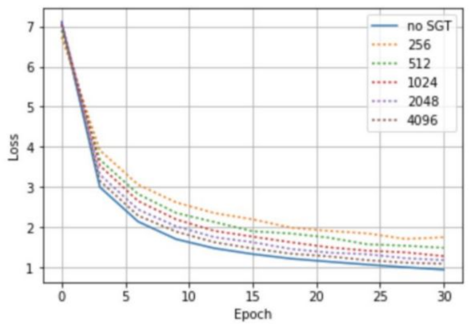
Figure 4.7: Evolution of the ratio edge intra-class during 20 epochs. An AlexNet is trained on ImageNet RGB (without Sobel filter) with PIC. *singl_with_NNcl*: singletons redistribution in their closest clusters. *singl_with*: singletons are kept as they are. *singl_without*: singletons are discarded for the training of the network.

4.7 Approximation for large scale dataset

With PIC algorithm we don't set beforehand the number of clusters. If the clustering gives only singletons, the last fully-connected layer of the model can become very big. Inspired by the work of Joulin et al. [48], we decide to implement a method called *stochastic gradient descent over targets* (SGT). It consists in updating only a subset of the weights of the last fully connected layer. This subset can be of different sizes. At the minimum, we consider only the clusters with representatives present in the mini-batch. At the maximum, we upload all the weights (no approximation). As mentioned before, we are using a multi-class logistic loss. Joulin et al. identify a lower and upper bounds for the approximate loss in this case. For the approximation to be better, we have to update bigger subsets of weights. However, there is a trade-off between quality of the approximation and time per batch.



(a) ImageNet classification



(b) Fixed clusters as labels. The clusters are obtained before the training with PIC algorithm on descriptors

Figure 4.8: Evolution of the loss during training on ImageNet with different approximations of the last fully connected layer of the network. *SGT* stands for stochastic gradient over target. In this experiment, *epoch* has its original meaning and not the one specific to this report.

Table 4.2: Performance on supervised training on ImageNet with different approximations of the multi-class logistic loss with SGT.

	Top@1 accuracy
No approx	56.1
SGT 256	54.2
SGT 512	55.6

First, in order to validate our implementation and method, we train an AlexNet architecture on ImageNet with full supervision. The evolution of the loss can be seen in Figure 4.8(a). On Figure 4.8(a), the dotted blue curve corresponds to the case with no approximation (all the weights are loaded). For the other curves (loss approximations), the neurons corresponding to classes present in a mini-batch are always updated. Then, a subset of random neurons are selected to reach a total size of subset of 256 (orange) or 512 (green). The final top@1 accuracy for these three settings after 90 epochs of training are reported in Table 4.2. We observe that, as expected, the approximation is better when the subset of weights updated is bigger. We also remark that the performance we report (without approximation) is lower than the one reported in the literature: 56.1 % versus 59.3 [37]. This gap can be explained by the fact that we use Sobel inputs and not directly RGB

Table 4.3: Average of the time per mini-batch for our method with different approximations of the multi-class logistic loss.

	Time per mini-batch (ms)
No approx	154
SGT 256	177
SGT 512	211
SGT 1024	277
SGT 2048	396
SGT 4096	713

images. Also, we report the results on the central crop while it is commonly used to perform an average over 10 crops in the image.

On Figure 4.8(b) we plot with dotted curves the evolution of the loss with different degrees of approximation. The solid blue curve corresponds to the loss without approximation. On Table 4.3 we report the times per mini-batch for those different cases. To conclude, we observe that the more updated weights, the better the approximation but the slower the time per mini-batch. In practice, it turns out that we do not need to use any approximation on our multi-class logistic loss.

4.8 YFCC100m dataset

ImageNet is naturally a clustered dataset (images distributed over 1000 classes). Thus, one can expect our clustering-training approach to work well on this dataset. In addition, ImageNet is a supervised dataset so it is an artificial setting to consider it unlabeled. YFCC100m is a bigger dataset, easily accessible and with a wider coverage than ImageNet. That is why we believe that models trained on YFCC100m might produce better and more general features. However, ImageNet is a dataset of a very good quality and has been proven to produce excellent features. It is a difficult challenge to get competitive results with models not relying neither on ImageNet nor supervision.

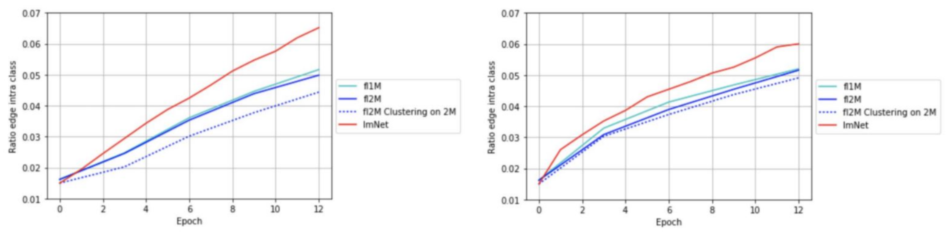
Adapt the method. On Algorithm 2, we introduce a slightly different and more general version of our main approach. In particular, when $\tilde{X} = X$ it is equivalent to the Algorithm 1. This approach allows to have a very large pool of data X and to use only a subset of it, $\tilde{X}$, at each epoch.

Algorithm 2 Approach for very large scale dataset.

```
Data: dataset $X$
for epoch $\in [1, N]$ do
    Sample a subset $\tilde{X}$ of X
    Compute the representations $f_\Theta(\tilde{X})$ for the whole subset $\tilde{X}$
    Cluster $f_\Theta(\tilde{X})$
    Assign to each data $x \in \tilde{X}$ the id of the cluster it belongs to
    Train the network $f_\Theta$ with clusters as labels and a logistic loss $\ell$
end for
```

Comparison with ImageNet. We start with training an AlexNet architecture with either the full ImageNet dataset or with fixed subsets of 1M or 2M of YFCC100m (Figure 4.9). In the two plots Figure 4.9, one epoch corresponds to a total amount of 5005 updates of the network weights. We report the ratio edge intra-class on the 10-NN graph of descriptors throughout a few epochs. The solid curves correspond to the case where a subset of 1.2M of images from the dataset is sampled at each epoch to perform the clustering and the training stages ($|\tilde{X}| = 1.2M$). For ImageNet and Flickr-1M cases, it means considering the whole dataset at each epoch ($\tilde{X} = X$). However, for Flickr-2M a different subset of the whole dataset is sampled from the fixed pool of 2M images at each epoch. The dotted curve for Flickr-2M corresponds to the case where the whole pool of 2M images is clustered but only a set of 1.2M images (with repetition due to our sampling strategy uniform over targets) are sampled to train the CNN on.

First we observe that training on Flickr-2M with a discrepancy of data used for clustering and training (dotted curve: case where Algorithm 2 does not apply) degrades the performance. Second, the performances of Flickr-1M and Flickr-2M are equivalent in term of ratio edge intra-class after a few epochs. It doesn't seem to improve nor alter the performance to use more data. Finally, the ratio edge intra-class is higher when ImageNet is used as training set. It is expected since the performance is measured on ImageNet. Moreover in this experiment we are using a small subset of Flickr. As a matter of fact, Joulin et al. [48] experiments suggest that at least 20 millions of images from Flickr



(a) PIC (b) k-means

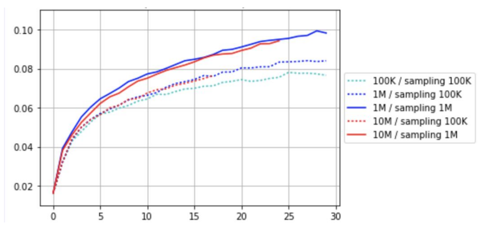
Figure 4.9: Ratio edge intra class on the 10-NN graph of descriptors throughout few epochs for different training datasets.

are required to beat performances of ImageNet on different transfer tasks.

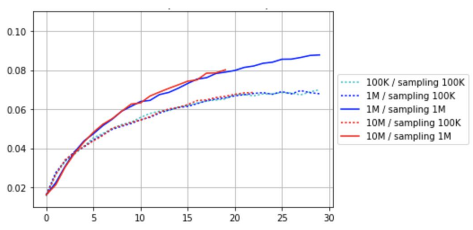
Benchmark of different sizes of subset. In the following experiment (Figure 4.10), we benchmark different choices for $|X|$ and $|\tilde{X}|$ (see Algorithm 2 for a definition of X and $\tilde{X}$) Once again, we report performance through the ratio edge intra-class on a 10-NN graph of descriptors at fixed number of updates of the network weights. We compare three sizes of pool of available images: $|X| = 100000$ (*100K*), $|X| = 1000000$ (*1M*) and $|X| = 10000000$ (*10M*). For each size of X, we compare different sizes of subset $\tilde{X}$ sampled at each epoch. In particular, for Flickr-1M ($|X| = 1000000$), we either sample a subset of 100K images ($|\tilde{X}| = 100000$) or we use the whole 1M dataset ($\tilde{X} = X$). In the same spirit, for Flickr-10M, we test two cases: sampling 100K or 1M images at each epoch. We do not report results with $\tilde{X} = X$ because, albeit feasible, we failed to conduct the experiment as it is computationally heavy and difficult to set up in practice.

We observe in Figure 4.10 that the performances with equal sizes of sampling at each epoch (curves with the same line-style and different colors) are similar. Thus, with a fixed size of data used at each epoch, the size of the pool of available data doesn't seem to matter. Using a bigger dataset doesn't alter nor improve the performance (except for PIC *100K / sampling 100K*).

Moreover, we compare the different sizes of $\tilde{X}$ for a fixed subset of Flickr X (curves with different line-styles and the same color). We observe that sampling more data at each epoch improves the performance.



(a) PIC



(b) k-means

Figure 4.10: Ratio edge intra-class on the 10-NN graph of descriptors every 11700 updates of network weights. The name of each curve is $|X|$ / sampling $|\tilde{X}|$

.

Chapter 5

Experiments

In this chapter, we report results on a diverse set of experiments for five models. These models are CNN with AlexNet architecture trained with different versions of our method. They required 500 (resp. 167) epochs of training on ImageNet (resp. YFCC100m) dataset during about 10 days on one GPU. All in all, they have been trained during the same number of Θ updates. This way, their performances can be compared. A description of the five models is given Table 5.1. We have performed hyper-parameter searches on learning rate, weight decay, and the number of clusters k in the k-means version. The model selection is done by measuring, at fixed regular number of updates of Θ, the accuracy on the validation set of the PASCAL VOC classification transfer task.

Table 5.1: Description of the five models presented in Section 5. LR stands for learning rate while WD stands for weight decay.

Model name	Dataset	Clustering	Sobel	LR	WD
PIC-ImNet-Sob	ImageNet	PIC	Yes	0.05	10^{-5}
km-ImNet-Sob	ImageNet	10K-means	Yes	0.02	10^{-5}
km-ImNet-RGB	ImageNet	1K-means	No	0.02	10^{-6}
km-fl1M-Sob	Flickr-1M	10K-means	Yes	0.02	10^{-5}
km-fl2M-Sob	Flickr-2M	10K-means	Yes	0.02	10^{-5}

5.1 Transfer tasks

In order to assess the quality of the features learned with our method, we proceed to various transfer learning experiments. These experiments consist in adapting our pre-trained models to solve particular visual tasks. It also allows us to compare our approach with state-of-the-art works presented in the related work section notably.

For some of our transfer experiments, we freeze the parameters of the convolutional layers and train a classifier directly on top. This is a strong setting to evaluate the capability of the network to provide semantic and general representations from images. Another setting, often referred to as fine-tuning, consists in training the whole network with our learned parameters as initialization.

Datasets and baselines. We consider three datasets for our transfer tasks: PASCAL VOC (2007 and 2012) [18], ImageNet [1] and Places [62]. The transfer tasks with PASCAL VOC dataset are all the more interesting because it is a small dataset (2.5K images in the train + val set and 5K in the test set). Thus, it reflects well the "real-world" case where a model, after having been trained with heavy computational resources, is publicly released. Then, it can simply be downloaded and adapted to a particular task of interest on a dataset of a much smaller size. We compare our approach to supervised, self-supervised and unsupervised methods that we gave a quick overview of in the related work section.

ImageNet and Places classifications. First, we transfer the features learned by our method to ImageNet classification task. This is particularly interesting for our models pre-trained on Flickr since we aim at learning general-purpose features. When the pre-training and transfer sets are the same, we cannot really assess the ability to generalize of our features. However, we report the results for models pre-trained on ImageNet as well, to allow the comparison with concurrent approaches. This way, it also enables to make a direct comparison between models trained with and without supervision on the same dataset. For all the following experiments, we select the best combination of hyper-parameters with k-fold cross validation on the training set. We compute at each epoch the top@1 accuracy on validation set,

Method	Acc@1
ImageNet, labels	59.7
Random	12.0
Wang et al. [43]	29.8
Doersch et al. [36]	30.4
Donahue et al. [16]	32.2
Noroozi and Favaro [39]	34.6
Zhang et al. [31]	35.2
Bojanowski and Joulin [28]	36.0
Ours. PIC-ImNet-Sob	**45.9**
Ours. km-ImNet-RGB	37.1
Ours. km-ImNet-Sob	44.0
Ours. km-fl1M-Sob	39.6
Ours. km-fl2M-Sob	40.1

Table 5.2: Comparison of the proposed approach to state-of-the-art unsupervised feature learning on ImageNet. A full multi-layer perceptron is retrained on top of the features. We report classification accuracy (acc@1). Except for Noroozi and Favaro [39], all the numbers are taken from Bojanowski and Joulin [28].

averaged over 10 crops. We report the maximum of those accuracies amongst 90 epochs of training.

The first setting consists in freezing the parameters of the convolutional layers and to train the multi-layer perceptron (MLP) from scratch [16]. The results are reported in Table 5.2. All our models have better performance than concurrent approaches. In particular, our best model, trained with PIC on ImageNet dataset with Sobel filtering, outperforms the best concurrent approach with a margin of almost 10%. However, we are still way below the score of a model trained with supervision.

Another setting [31] consists in training a logistic regression classifier on top of the N first frozen convolutional layers. We apply average-pooling to the resulting feature vectors with equal kernel and stride sizes to reach a dimensionality between 9000 and 10000. We do so for ImageNet and Places [62] classification. Places is a dataset of 2.5 millions images spread over 205 scene categories. As it can be seen in Tables 5.3 and 5.4 we outperform state of the art performances for most cases. Our model trained on RGB input performs better than

Method	conv1	conv2	conv3	conv4	conv5
ImageNet labels	19.3	36.3	44.2	48.3	50.5
Random	11.6	17.1	16.9	16.3	14.1
Pathak et al. [33]	14.1	20.7	21.0	19.8	15.5
Noroozi and Favaro [39]	**18.2**	28.8	34.0	33.9	27.1
Donahue et al. [16]	17.7	24.5	31.0	29.9	28.0
Doersch et al. [36]	16.2	23.3	30.2	31.7	29.6
Zhang et al. [31]	12.5	24.5	30.4	31.5	30.3
Zhang et al. [32]	17.7	29.3	35.4	35.2	32.8
Ours. PIC-ImNet-Sob	13.5	32.4	**40.8**	**40.5**	**37.8**
Ours. km-ImNet-Sob	13.4	31.9	40.2	37.7	36.0
Ours. km-ImNet-RGB	18.0	**32.5**	39.2	37.2	30.6
Ours. km-fl1M-Sob	13.5	30.9	38.0	34.5	31.4
Ours. km-fl2M-Sob	13.5	30.4	37.5	34.2	32.3

Table 5.3: Linear classification on ImageNet on AlexNet convolution features. We report accuracy. Noroozi and Favaro [39] use stride 2 instead of 4 in the conv1 layer. All the numbers are from Zhang et al. [32].

our other models when a classifier is trained on top of the first convolutional layer. Unsurprisingly, it helps to leverage color information when low-level features only are used. Remarkably, for Places classification (Table 5.4), we are on par with ImageNet labels. We notice in Table 5.3 that our best models have performance roughly 4% behind the supervised counter part for conv1, conv2 and conv3. However, the gap between features learned with and without supervision increases for conv4 and conv5. Our model doesn't manage to go up a further level of abstraction after conv3. This is where the supervised model performs much better.

Transferring to PASCAL VOC. We follow standard frameworks for classification, segmentation and object detection on PASCAL VOC dataset. The results are reported in Table 5.5. As for the previous experiments, we outperform both unsupervised and self-supervised methods. It shows that the models trained with our method provide good features. However, we are still far below performances of models trained with supervision. Furthermore, we notice that, for our models, the difference between *fc6-8* and *all* (fine-tuning) seems to be shrinking.

Method	conv1	conv2	conv3	conv4	conv5
Places labels	22.1	35.1	40.2	43.3	44.6
ImageNet labels	22.7	34.8	38.4	39.4	38.7
Random	15.7	20.3	19.8	19.1	17.5
Pathak et al. [33]	18.2	23.2	23.4	21.9	18.4
Donahue et al. [16]	21.4	26.2	27.1	26.1	24.0
Zhang et al. [31]	16.0	25.7	29.6	30.3	29.7
Doersch et al. [36]	19.7	26.7	31.9	32.7	30.9
Noroozi and Favaro [39]	23.0	32.1	35.5	34.8	31.3
Zhang et al. [32]	21.3	30.7	34.0	34.1	32.5
Ours. PIC-ImNet-Sob	19.5	32.9	**39.1**	**39.5**	**35.9**
Ours. km-ImNet-Sob	19.6	**33.0**	38.7	38.7	34.4
Ours. km-ImNet-RGB	**23.8**	32.8	37.3	36.0	31.0
Ours. km-fl1M-Sob	19.7	33.0	38.4	39.0	35.2
Ours. km-fl2M-Sob	19.6	32.5	38.5	38.7	34.9

Table 5.4: Linear classification on Places [62] with AlexNet convolution features. We report accuracy. Noroozi and Favaro [39] use stride 2 instead of 4 in the conv1 layer. All the numbers are from Zhang et al. [32].

Encouragingly, it is also a behavior observed for the model trained on supervised ImageNet but not for the models trained with unsupervised approaches.

Those transfer learning experiments allow to draw several conclusions. First, we outperform state-of-the-art self-supervised and unsupervised approaches. Our method allows to acquire good general-purpose features that transfer well to standard visual tasks. Secondly, PIC version of the algorithm gives better performances than k-means. However, PIC is a much more sophisticated algorithm and we prefer our method to be as simple as possible. In addition, the gap between PIC and k-means performances is never higher than 2%. We think that what makes our method work is more the fact of reassigning targets than using a clustering technique to do so. Finally, the model pre-trained with a subset of 2 millions images from YFCC100m is slightly better than the one pre-trained with 1 million images. Thus, using more data from YFCC100m seems to improve the quality of the learned features. However, the gain in performance remains small and

	Classification		Detection	Segmentation
Trained layers	fc6-8	all	all	all
ImageNet labels	78.9	79.9	56.8	48.0
Random	29.0	61.9	47.8	31.3
Pathak et al. [33]	34.6	56.5	44.5	29.7
Donahue et al. [16]	52.3	60.1	46.9	35.2
Pathak et al. [30]	–	61.0	52.2	–
Wang et al. [43]	55.6	63.1	47.2	–
Doersch et al. [36]	55.1	65.3	51.1	–
Bojanowski and Joulin [28]	56.7	65.3	49.4	–
Zhang et al. [31]	61.5	65.9	46.9	35.6
Zhang et al. [32]	63.0	67.1	46.7	36.0
Noroozi and Favaro [39]	–	67.6	53.2	37.6
Ours. PIC-ImNet-Sob	**71.0**	**73.0**	**53.9**	**43.2**
Ours. km-ImNet-Sob	70.9	72.9		
Ours. km-ImNet-RGB	63.6	65.3	49.8	
Ours. km-fl1M-Sob	67.3	69.3	52.8	40.8
Ours. km-fl2M-Sob	67.7	70.2	53.0	41.3

Table 5.5: Comparison of the proposed approach to state-of-the-art unsupervised feature learning on VOC 2007 Classification and detection.

performances of models trained with YFCC100m are still below the ones of our models trained with ImageNet

5.2 Low-shot learning

A common practical case is to have access to only a limited amount of labeled training data. To observe how our model behaves in this setting, we decide to conduct a low-shot learning experiment on ImageNet classification. It consists in training a whole CNN architecture with only n samples per class with our learned parameters as initialization. We report the results averaged over 10 runs of the experiment with different sets of n training data per class. For all cases, the standard deviation is quite low (around 0.1%). We perform k-fold cross validation to set the hyper-parameters and we report top@1 accuracy averaged over 10 crops. We plot as baseline the results for a randomly initialized networks (no pre-training).

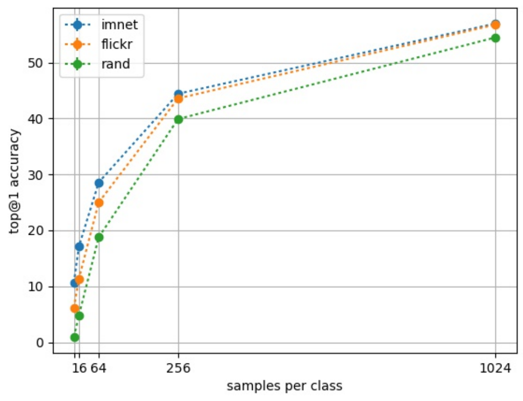


Figure 5.1: Low-shot learning. Top@1 accuracy of ImageNet classification. Results are averaged over 10 runs. The first point corresponds to 4 samples per class.

We can see in Figure 5.1 that using the weights learned with our approach as initialization allows to have better performance than with a randomly initialized network. It can be especially useful when labeled training data is scarce.

5.3 Visualizations

First layer filters. First, we show in Figure 5.2 the filters from the first layer of an AlexNet network trained with our unsupervised method. The filters in Figure 5.2 are the ones of our best models trained on ImageNet with k-means clustering. For the model trained with gradients as input (Sobel), the filters are in greyscale since we show the composition of the Sobel filters with the filters of the first layer. We can see that the learned filters are quite sharp and informative.

Nearest neighbors queries. Our PIC implementation relies on a graph of nearest neighbors. Also, we have been studying the evolution of the statistics of a graph of nearest neighbors of descriptors several times in this report. For these reasons, we decide to visualize nearest neighbors queries in the feature space. We compute the feature representation of a query image x and search for its 3 nearest neighbors according to the ℓ^2 distance. Figure 5.3 shows images and their 3 nearest neighbors

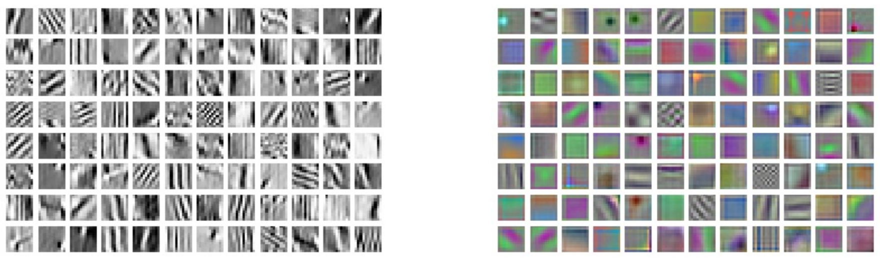


Figure 5.2: Filters from the first layer of an AlexNet trained on unsupervised ImageNet with (left) and without (right) sobel filtering.

with our best trained model (PIC-ImNet-Sob) and with a random network. In particular, we can see that for the random network one the 3-NN of the sunset query (first row) is another image of sunset. The two other NN have the same overall structure than the query. It shows that a not trained network is able to capture low-level structures in an image. After training, the NN matches are largely improved. Finally, this visualization supports the idea that a randomly initialized network captures a weak signal and that our training method is efficient to improve the features induced by the network.

Feature Visualization by Optimization. Second, in order to qualitatively assess what our models have learned, we perform gradient ascent optimization of an input image. It consists in optimizing an activation objective with respect to the input image. Starting from a noise initialization, we apply gradient steps to the input so that, when fed to the network, it fires maximally the neurons in a fixed channel. To have smoother resulting images, we apply standard regularization techniques: ℓ^2 penalization of the input and Gaussian blurring every 5 image updates [63]. In addition, we also display in Figure 5.4, the images in a subset of 1M images from Flickr that maximally activate the corresponding channel. As for supervised networks, along the layers of the network, we observe larger and larger textural structures such as eyes, perspective or trees. In particular, the features of the first layer (first row) correspond to texture and low-level features. Going deeper in the network, more complex textures (tree) and structures (stained-glass window, perspective, car, fur and nose) are revealed. Interestingly, we do not see object-like structures appearing in the last lay-

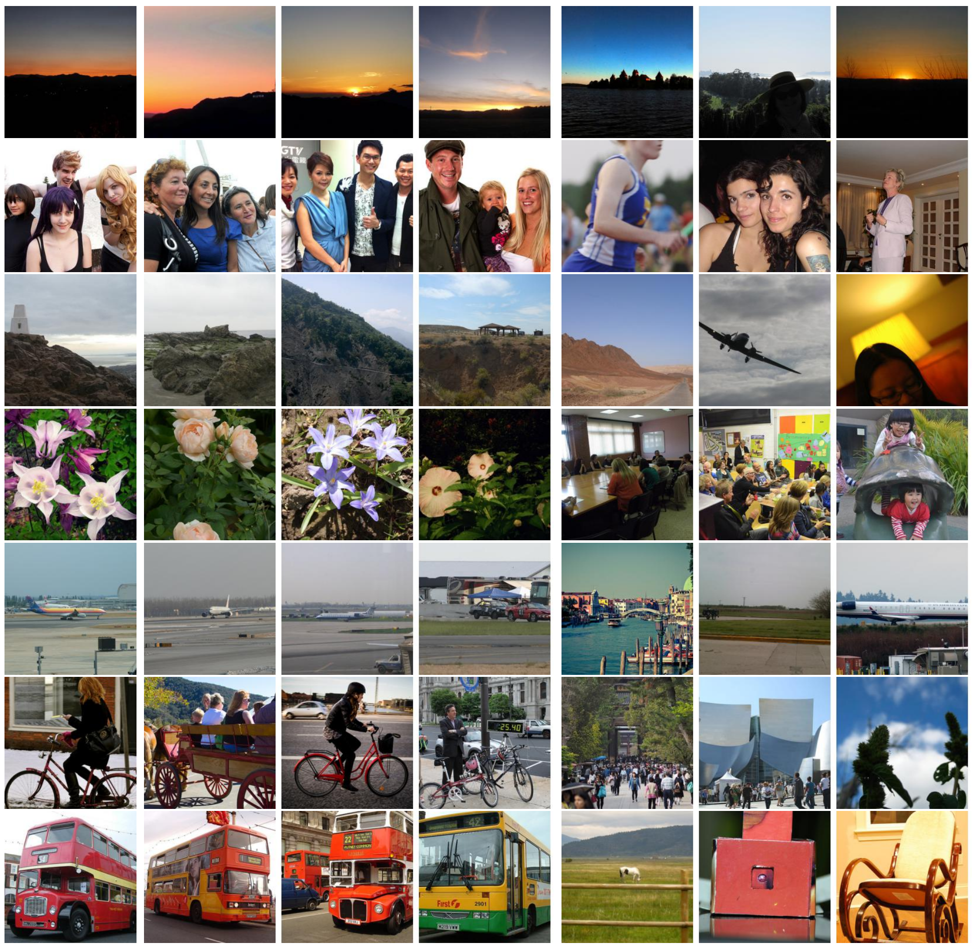
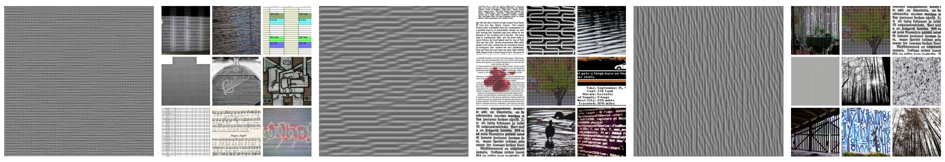
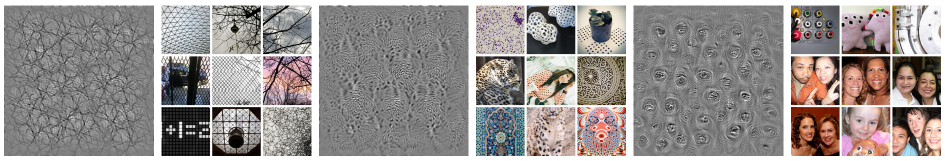


Figure 5.3: Images and their 3 nearest neighbors in a subset of Flickr in the feature space using l_2 distance. The query images are shown on the left column. Then, the following 3 column correspond to a network after training with our method and the last 3 to the same network before training (random weights)

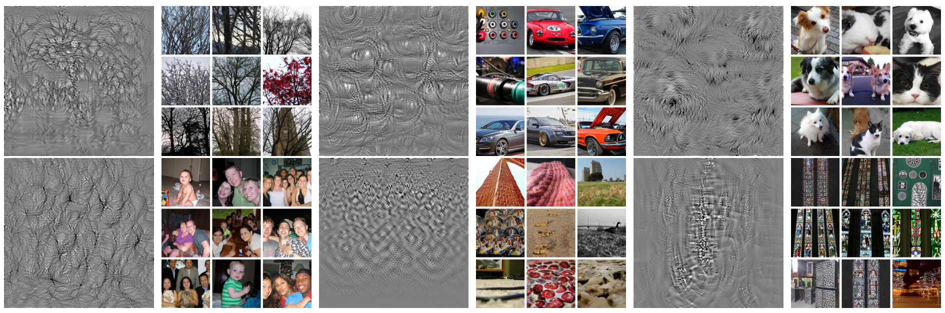
ers, rather slightly noisier version of structures already learned in the lower layers. This qualitative result suggests that our unsupervised learning is better at learning low level layers than high level layers. Moreover this observation is consistent with the results of Tables 5.3 and 5.4: training a linear classifier on top of conv5 doesn't give better performance than on top of conv3.



(a) First convolutional layer



(b) Third convolutional layer



(c) Fifth convolutional layer

Figure 5.4: Optimisation of an input image to fire maximally a channel in a given layer. Top activated 9 images in a subset of Flickr dataset corresponding to this channel.

Chapter 6

Conclusion

In this master thesis, we have proposed a simple and original solution for unsupervised representation learning based on clustering. We have detailed some focal points that have led us towards the design of this approach. Then we have evaluated our best models on standard benchmarks and outperformed state-of-the-art concurrent approaches by a fair margin for all of them. We have also performed visualizations to get an idea of what the features learned with our method look like.

There is still room for improvement regarding our performance on Flickr dataset. More work has to be done to leverage the size and diversity of this dataset. As a matter of fact, we have only shown results with relatively small subsets of this very big dataset.

Finally, this work is a step towards bridging the gap in image recognition between supervised and unsupervised models. Indeed, in domains such as speech or Natural Language Processing, the models pre-trained in an unsupervised manner have been shown to produce excellent general features [64]. The fact that the pre-training stage is performed without any supervision allows much more flexibility in the sense that the pre-training can be done on any dataset. Henceforth, if unsupervised learning was able to provide excellent features in computer vision, one would be able to learn visual features specific to satellite or medical images for example.

Bibliography

[1] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE (2009) 248–255

[2] Jaderberg, M., Simonyan, K., Zisserman, A., et al.: Spatial transformer networks. In: Advances in Neural Information Processing Systems. (2015) 2017–2025

[3] Azizpour, H., Sharif Razavian, A., Sullivan, J., Maki, A., Carlsson, S.: From generic to specific deep representations for visual recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. (2015) 36–45

[4] Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2014) 580–587

[5] Pinheiro, P.O., Collobert, R., Dollár, P.: Learning to segment object candidates. In: Advances in Neural Information Processing Systems. (2015) 1990–1998

[6] Hoffman, J., Guadarrama, S., Tzeng, E.S., Hu, R., Donahue, J., Girshick, R., Darrell, T., Saenko, K.: Lsda: Large scale detection through adaptation. In: Advances in Neural Information Processing Systems. (2014) 3536–3544

[7] He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. (2015) 1026–1034

[8] Stock, P., Cisse, M.: Convnets and imagenet beyond accuracy: Explanations, bias detection, adversarial examples and model criticism. arXiv preprint arXiv:1711.11443 (2017)

[9] Thomee, B., Shamma, D.A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., Li, L.J.: The new data and new challenges in multimedia research. arXiv preprint arXiv:1503.01817 **1**(8) (2015)

[10] Dosovitskiy, A., Springenberg, J.T., Riedmiller, M., Brox, T.: Discriminative unsupervised feature learning with convolutional neural networks. In: Advances in Neural Information Processing Systems. (2014) 766–774

[11] Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. In: Advances in neural information processing systems. (2007) 153–160

[12] Huang, F.J., Boureau, Y.L., LeCun, Y., et al.: Unsupervised learning of invariant feature hierarchies with applications to object recognition. In: Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, IEEE (2007) 1–8

[13] Masci, J., Meier, U., Cireşan, D., Schmidhuber, J.: Stacked convolutional auto-encoders for hierarchical feature extraction. Artificial Neural Networks and Machine Learning–ICANN 2011 (2011) 52–59

[14] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. (2014) 2672–2680

[15] Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)

[16] Donahue, J., Krähenbühl, P., Darrell, T.: Adversarial feature learning. arXiv preprint arXiv:1605.09782 (2016)

[17] Dumoulin, V., Belghazi, I., Poole, B., Lamb, A., Arjovsky, M., Mastropietro, O., Courville, A.: Adversarially learned inference. arXiv preprint arXiv:1606.00704 (2016)

[18] Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International journal of computer vision **88**(2) (2010) 303–338

[19] Kohonen, T.: The self-organizing map. Neurocomputing **21**(1) (1998) 1–6

[20] Vesanto, J., Alhoniemi, E.: Clustering of the self-organizing map. IEEE Transactions on neural networks **11**(3) (2000) 586–600

[21] Liao, R., Schwing, A., Zemel, R., Urtasun, R.: Learning deep parsimonious representations. In: Advances in Neural Information Processing Systems. (2016) 5076–5084

[22] Xie, J., Girshick, R., Farhadi, A.: Unsupervised deep embedding for clustering analysis. In: International Conference on Machine Learning. (2016) 478–487

[23] Yang, J., Parikh, D., Batra, D.: Joint unsupervised learning of deep representations and image clusters. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 5147–5156

[24] Douze, M., Jégou, H., Johnson, J.: An evaluation of large-scale methods for image instance and class discovery. arXiv preprint arXiv:1708.02898 (2017)

[25] Bach, F.R., Harchaoui, Z.: Diffrac: a discriminative and flexible framework for clustering. In: Advances in Neural Information Processing Systems. (2008) 49–56

[26] Joulin, A., Bach, F., Ponce, J.: Discriminative clustering for image co-segmentation. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE (2010) 1943–1950

[27] Joulin, A., Bach, F., Ponce, J.: Multi-class cosegmentation. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE (2012) 542–549

[28] Bojanowski, P., Joulin, A.: Unsupervised learning by predicting noise. Proc. ICML (2017)

[29] de Sa, V.R.: Learning classification with unlabeled data. In: Advances in neural information processing systems. (1994) 112–119

[30] Pathak, D., Girshick, R., Dollár, P., Darrell, T., Hariharan, B.: Learning features by watching objects move. Proc. CVPR (2017)

[31] Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: Proc. ECCV. (2016)

[32] Zhang, R., Isola, P., Efros, A.A.: Split-brain autoencoders: Unsupervised learning by cross-channel prediction. arXiv preprint arXiv:1611.09842 (2016)

[33] Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: Feature learning by inpainting. In: Proc. CVPR. (2016)

[34] Agrawal, P., Carreira, J., Malik, J.: Learning to see by moving. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 37–45

[35] Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)

[36] Doersch, C., Gupta, A., Efros, A.A.: Unsupervised visual representation learning by context prediction. In: Proc. ICCV. (2015)

[37] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. (2012) 1097–1105

[38] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

[39] Noroozi, M., Favaro, P.: Unsupervised learning of visual representations by solving jigsaw puzzles. In: Proc. ECCV. (2016)

[40] Li, D., Hung, W.C., Huang, J.B., Wang, S., Ahuja, N., Yang, M.H.: Unsupervised visual representation learning by graph-based consistent constraints. In: European Conference on Computer Vision, Springer (2016) 678–694

[41] Lowe, D.G.: Object recognition from local scale-invariant features. In: Computer vision, 1999. The proceedings of the seventh

IEEE international conference on. Volume 2., Ieee (1999) 1150–1157

[42] Hoffer, E., Ailon, N.: Deep metric learning using triplet network. In: International Workshop on Similarity-Based Pattern Recognition, Springer (2015) 84–92

[43] Wang, X., Gupta, A.: Unsupervised learning of visual representations using videos. In: Proc. ICCV. (2015)

[44] Wang, X., He, K., Gupta, A.: Transitive invariance for self-supervised visual representation learning. arXiv preprint arXiv:1708.02901 (2017)

[45] Doersch, C., Zisserman, A.: Multi-task self-supervised visual learning. arXiv preprint arXiv:1708.07860 (2017)

[46] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 770–778

[47] Chollet, F.: Xception: Deep learning with depthwise separable convolutions. arXiv preprint arXiv:1610.02357 (2016)

[48] Joulin, A., van der Maaten, L., Jabri, A., Vasilache, N.: Learning visual features from large weakly supervised data. In: European Conference on Computer Vision, Springer (2016) 67–84

[49] Weyand, T., Kostrikov, I., Philbin, J.: Planet-photo geolocation with convolutional neural networks. In: European Conference on Computer Vision, Springer (2016) 37–55

[50] Douze, M., Szlam, A., Hariharan, B., Jégou, H.: Low-shot learning with large-scale diffusion. arXiv preprint arXiv:1706.02332 (2017)

[51] Lin, F., Cohen, W.W.: Power iteration clustering. Proc. ICML (2010)

[52] Gallier, J.: Spectral theory of unsigned and signed graphs. applications to graph clustering: a survey. arXiv preprint arXiv:1601.04692 (2016)

[53] Bottou, L.: Stochastic gradient descent tricks. In: Neural networks: Tricks of the trade. Springer (2012) 421–436

[54] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11) (1998) 2278–2324

[55] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. nature **323**(6088) (1986) 533

[56] Xu, L., Neufeld, J., Larson, B., Schuurmans, D.: Maximum margin clustering. In: Advances in neural information processing systems. (2005) 1537–1544

[57] Joulin, A., Bach, F.: A convex relaxation for weakly supervised classifiers. arXiv preprint arXiv:1206.6413 (2012)

[58] Paulin, M., Douze, M., Harchaoui, Z., Mairal, J., Perronin, F., Schmid, C.: Local convolutional features with unsupervised training for image retrieval. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 91–99

[59] Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning. (2015) 448–456

[60] Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of machine learning research **15**(1) (2014) 1929–1958

[61] Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with gpus. arXiv preprint arXiv:1702.08734 (2017)

[62] Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., Oliva, A.: Learning deep features for scene recognition using places database. In: Advances in neural information processing systems. (2014) 487–495

[63] Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., Lipson, H.: Understanding neural networks through deep visualization. arXiv preprint arXiv:1506.06579 (2015)

[64] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. Journal of Machine Learning Research **12**(Aug) (2011) 2493–2537

www.kth.se