Deep Learning In Practice
TP 2 : Image Retrieval

Dorian Baudry, Alexandre Filiot

March 4, 2019

1 Training

1.1 Creating a network with the AlexNet architecture

Question 1: What does each row of the matrix feats represent?

Each row of the matrix *feats* represents the encoding of an image (from the 5063 set of training images). For each image, this encoding corresponds to the output of the last hidden layer, a 4096×4096 FC layer. It is a 4096-dimensional real vector, which can be seen as an image-embedding vector. *Feats* thus contains, for each training image, a very compact and informative feature vector belonging to a 4096-dimensional semantic space where similar images with similar features are close to each other. We can talk about feature representations of images.

Question 2: Where does the dimension of these lines comes from and how do we extract these features?

As said just above, the dimension of these lines comes from the dimension of the last hidden FC layer. These features can be extracted in recovering the output from this layer instead of the output from the softmax layer which provides probabilities to belong to a specific class.

Question 3: What can be observe from the t-SNE visualization? Which classes 'cluster' well? Which do not?

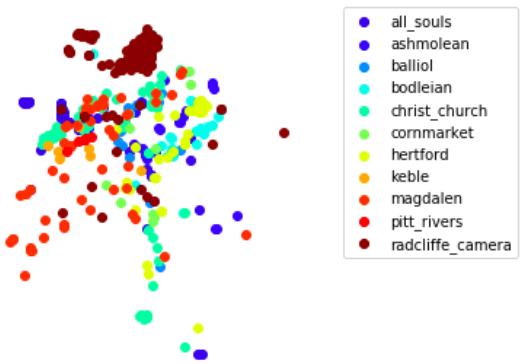

Figure 1: t-SNE applied on AlexNet output features pretrained with ImageNet dataset

t-SNE algorithm aims at projecting each 4096-dimensional feature vector into a 2 or 3-dimensional vector in order to model similar images as nearby points and vice versa. From the 2-dimensional t-sne plot, we can observe that clustering performs very badly except for the *radcliffe camera* class. If one run the algorithm several times, the conclusion stays the same. From one run to another, clusters are slighty different. Conclusion: features from ImageNet are not good at intra-class discrimination.

1.2 Finetuning the created network on the Landmarks dataset

Question 4: Should we get better results? What should change? Why?

Convolutional Neural Networks such as AlexNet may have a huge number of parameters, sometimes in the range of millions ($\approx$ 60 millions here). That's why training a CNN on a small dataset (typically in cases where the number of training samples is far more inferior than the number of parameters), greatly affects the CNN's ability to generalize and thus results in overfitting. Accordingly, we firstly pre-trained our network on the ImageNet dataset, and then, continue training it (*i.e.* running back-propagation) on our dataset of interest, namely Oxford dataset. Provided that **our dataset is not drastically different in context to the original dataset (e.g. ImageNet), the pre-trained model will already have learned features that are relevant for the classification part of our problem**. Here, Oxford corresponds to a very specific domain. One should therefore use a different original dataset than ImageNet to pretrain our network. That's why we would expect better results from pretraining with Landmarks dataset. Nevertheless, one should also be careful about the size of this dataset to avoid overfitting, especially when the last layers are FC layers! As Landmarks dataset is very noisy, it is necessary to automaticly clean it. As highlighted in the slides, this technique results in a 40k spatially verified images with approximate bounding box annotations. 40k images is already satisfying but, combined with 2 FC layers at the end of our network, this could perhaps lead to overfitting.

In practise, there are a few general guidelines to follow when implementing fine-tuning:

- Truncate the last softmax layer of the pretrained network and replace it with our new softmax layer which is relevant with our new problem (see next question).
- As we use t-SNE algorithm and FC features to draw similarities between images, we shall keep the same number of features.
- One may also use a smaller learning rate but we will discuss it later on.

Question 5: Why do we change the last layer of the AlexNet architecture

To sum up, we are simultaneously:

- Fine-tuning our model: pretrain AlexNet on Landmarks dataset in mainly a classification purpose (this is one of the limitations to our current approach). This Neural Network aims at classifying between the **586 distinct classes (landmarks) of this dataset. That's why we changed 1000 (number of ImageNet classes) to 586.** However, one should not omit to add a softmax layer in order to output probabilities.
- Applying t-SNE on last FC layer's features: we project the 4096-dimensional feature space into a smaller space and make similarities between images. As such, we are able to first display the top-n most similar images from an input query and, using the previous probabilities, see if there is a right or wrong matching (same landmarks).

Question 6: How do we initialize the layers of *model_1b* for finetuning?

It is a common practice to *freeze* the weights of the first few layers of the pre-trained network (AlexNet for now). Indeed, the first few layers capture universal features like curves, edges, etc... that are also relevant to our new problem. That's why we want to keep those weights intact. Instead, we will get the network to focus on learning dataset-specific features in the subsequent layers. One may expect the last layers to capture some compact characteristics of Oxford landmarks architecture. That's why we should also use a smaller learning rate to continue the training with Oxford training samples. Since we expect the pre-trained weights to be quite good already as compared to randomly initialized weights, we do not want to distort them too quickly and too much. A common practice is to make the initial learning rate 10 times smaller than the one used for scratch training.

Question 6 bis: How does the visualization change after finetuning? What about the top results?

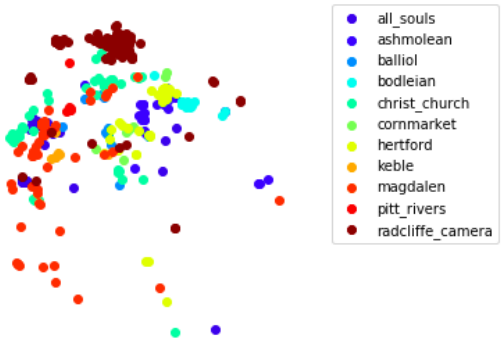

Figure 2: t-SNE applied on AlexNet output features pretrained with Landmarks dataset

We notice an improvement in the clustering. Globally, each class presents a small bunch of $5 \sim 10$ points. This was not observed previously. *Radcliffe Camera* is also better clustered than before: we observe, for this class, less outliers with smaller distances with the main group of points. An important aspect is that clusters are more distinguishable comparing to the previous visualization. Some classes are still hard to cluster, such as *Pitt Rivers, Christ Church, All Souls*. Regarding top results, we notice a huge improvement of the AP on a set of 11 queries (see table 1). Top results are visually more relevant (figure 3).

	Images											
	1	6	11	16	21	26	31	36	41	46	51	Mean
Model 1a	10.96	13.04	12.73	25.47	44.30	14.51	61.62	47.65	5.49	69.5	65.77	33.73
Model 1b	**31.08**	**28.93**	**24.67**	**67.69**	**55.58**	**15.72**	**62.50**	**68.42**	**27.57**	**83.76**	**76.99**	**49.36**

Table 1: Average Precision obtained on 11 images for the 2 models used so far. Model 1a: AlexNet + ImageNet + 4096FC - Model 1b: AlexNet + LM + 4096FC

Figure 3: Top most similar images. Top: ImageNet pretraining. Bottom: Landmarks pretraining.

Question 7: Why images need to be resized to 224×224 before they can fed to AlexNet? How can this affect results?

There are two possible answers. First one is simplistic: images are resized to 224×224 because the original AlexNet architecture takes as input 224×224 images. Second is more practical: A. Krizhevsky et al. (2012) needed to proceed to data augmentation in order to reduce overfitting (due to the 60 million parameters of AlexNet). According to the authors: "The first form of data augmentation consists of generating image translations and horizontal reflections. We do this by extracting random 224×224 patches (and their horizontal reflections) from the 256×256 images and training our network on these extracted patches". One could also add that resizing images allows to considerably decrease computational time. In any case, resizing images to this size may alter their resolution, distort them and remove details. Those details are usually learned by the last layers of AlexNet and are essential to build good feature representation of the images. Loosing these details thus alter the performance as well, despite fine-tuning on a more specific dataset. And this is what we observe: performance is better but not quite satisfying, in particular images clustering. Conclusion: working on small crops and low resolution is a non-negligible caveat of our current modelling.

1.3 Replacing last max pooling layer with GeM layer

Question 8: Why does the size of the feature representation changes?

The size of the feature representation changes because we removed the FC layers. Let's consider the work of Radenovic et al. (2018) on Generalized-mean pooling (GeM). Now, the output of our network - before GeM pooling, is a 2D convolutional layer composed of 256 2D convolutional filters: $\mathcal{X} = W \times H \times K$ where $K = 256$ and W and H are the resulting width and height of the these filters (or feature maps). We apply to it ReLU activation in order to make the values $x \in \mathcal{X}$ non-negative. A GeM pooling layer has K parameters $p_1, ..., p_K$ that can be manually set or learned in a differential way using back-propagation. We can define GeM as the following K-dimensional vector for each image:

$$f^{(g)} = [f_1^{(g)}, ..., f_K^{(g)}]^\top \qquad \text{with } f_k^{(g)} = \left(\frac{1}{\mathcal{X}_k} \sum_{x \in \mathcal{X}_k} x^{p_k} \right)^{\frac{1}{p_k}} \tag{1}$$

where $\mathcal{X}_k$ is the k-th filter. So, as said by the authors, the feature vector "finally consists of a single value per feature map, *i.e* the generalized-mean activation, and its dimensionality if equal to K". That's why here the dimensionality of the feature space is $K = 256$.

Question 9: Why does the size of the feature representation is important for a image retrieval task?

In image retrieval, the key problem is how to efficiently measure the similarity between images. This measure strongly depends on the feature representation. As part of it, the size of the feature space is important. Indeed, if the feature space is too small, the extracted information will be too compact to provide a complete description of the image. Whereas a too large feature space will contain redundant information, more painful computations and, the most important, will result in overfitting. There is no general tips on how to adjust the size of the feature space (except that it is generally a power of 2: $256, 512, 2048$ or 4096. However, it really depends on the architecture of the network and the data relevancy (data cleaning, data augmentation, etc...). In other words, choosing a big feature size is not crippling. In fact, actual top performing model proposed by Gordo et al. (2007), "End-to-end Learning of Deep Visual Representations for Image Retrieval", uses a 2048-dimensional feature space.

Question 10: How does the aggregation layer changes the t-SNE visualization?

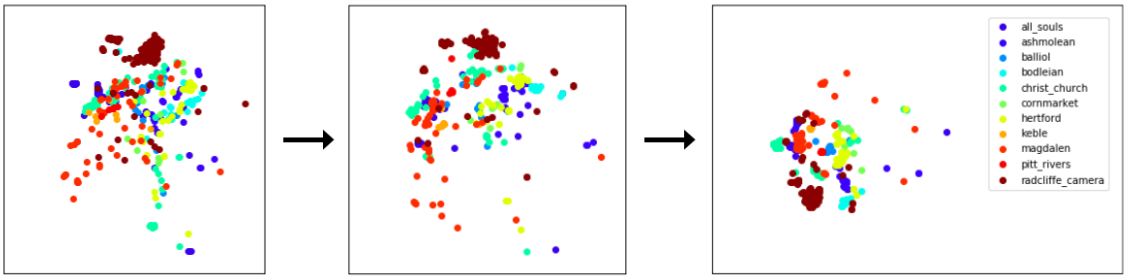

Figure 4: t-SNE visualizations. From left to right: Off-the-shell ImageNet, Landmarks fine-tuning, Landmarks fine-tuning with GeM pooling layer.

If we sum up the t-SNE visualizations obtained so far, we have clearly made progress (figure 4). Especially between the second and third steps. At first, with our first network, we were facing up to a very bad clustering, with no bunches of points and many outliers. Moreover, points from all classes were very mixed up. Then, fine-tuning the model with Landmarks dataset allowed to distinguish a bit more the different clusters. Now, the visualization starts being satisfying: each class presents a clear bunch of points that is the core of the cluster, with few outliers. Clusters are way more distinguishable (the overlaps are reduced) even if they still remain close to each other.

This improvement is clearly due to the introduction of the GeM pooling layer which has replaced the Fully Connected layers. Indeed, 4096-dimensional feature vectors contained too much information: the intra-class variance was too big to yield good clustering. One can add that this improvement in clustering is also observed through top result query. For the same image (q_idx=11), the AP is now around 69 with 8 correct matches over 10! Moreover, on average over the same 11 images as before, GeM pooling layer leads to a gain of 12 points in AP (see table 2).

	Images											
	1	6	11	16	21	26	31	36	41	46	51	Mean
Model 1a	10.96	13.04	12.73	25.47	44.30	14.51	61.62	47.65	5.49	69.5	65.77	33.73
Model 1b	31.08	28.93	24.67	**67.69**	55.58	15.72	62.50	68.42	**27.57**	83.76	76.99	49.36
Model 1c	**39.22**	**61.40**	**69.01**	49.89	**68.10**	**31.41**	**70.73**	**92.14**	17.03	**89.58**	**89.22**	**61.61**

Table 2: Average Precision obtained on 11 images for the 3 models used so far. Model 1a: AlexNet + ImageNet + 4096FC - Model 1b: AlexNet + LM + 4096FC - Model 1c: AlexNet + LM + 256 GeM

Question 11: Can we see some structure in the clusters of similarly labeled images?

Indeed. Each cluster of similarly labeled images is composed of a main bunch of points with some outliers. An important fact is that each bunch of points (the core of the clusters) are almost not overlapping with each other. If we removed the outliers from the visualization, we would observe distinct and dense clusters. It means that we must improve our feature representation to adapt to the specificity of those outliers.

1.4 ResNet18 architecture with GeM pooling

Question 12: Why do we change the average pooling layer of the original Resnet18 architecture for a generalized mean pooling?

There are 2 main reasons. The first one is quite pragmatic: GeM pooling layer revealed itself to be the top performing approach, combined with classical architecture (VGG, ResNet), and compared to previous state of the start (see figure 5).

Net	Method	F-tuned	Dim	Oxford5k	Oxford105k	Paris6k	Paris106k	Holidays	Hol101k
Compact representation using deep networks									
VGG	MAC [9]†	no	512	56.4	47.8	72.3	58.0	79.0	66.1
	SPoC [10]†	no	512	68.1	61.1	78.2	68.4	83.9	**75.1**
	CroW [11]	no	512	70.8	65.3	79.7	72.2	85.1	–
	R-MAC [12]	no	512	66.9	61.6	83.0	75.7	86.9‡	–
	BoW-CNN [48]	no	n/a	73.9	59.3	82.0	64.8	–	–
	NetVLAD [16]	no	4096	66.6	–	77.4	–	88.3	–
	NetVLAD [16]	yes	512	67.6	–	74.9	–	86.1	–
	NetVLAD [16]	yes	4096	71.6	–	79.7	–	87.5	–
	Fisher Vector [49]	yes	512	81.5	76.6	82.4	–	–	–
	R-MAC [26]	yes	512	**83.1**	**78.6**	**87.1**	**79.7**	**89.1**	–
	⋆ GeM	yes	512	**87.9**	**83.3**	**87.7**	**81.3**	**89.5**	**79.9**
Res	R-MAC [12]‡	no	2048	69.4	63.7	85.2	77.8	91.3	–
	R-MAC [27]	yes	2048	**86.1**	**82.8**	**94.5**	**90.6**	**94.8**	–
	⋆ GeM	yes	2048	**87.8**	**84.6**	92.7	86.9	93.9	**87.9**

Figure 5: Performance (mAP) comparison with the state-of-the-art image retrieval using VGG and ResNet (Res) deep networks, and using local features. Credits: Radenovic et al. (2018) "Fine-tuning CNN Image Retrieval with No Human Annotation". Note: *F-tuned* corresponds to the use of the fine-tuned network (yes), or the off-the-shelf network (no). The previous state of the art is highlighted in bold, new state of the art in red outline.

Then, one may use GeM pooling layer instead of an average pooling layer because it generalizes max and average pooling. When we look at equation (1), we see that, if we set $p_k = p = 1 \ \forall k = 1, ..., K$, then we reproduce the average pooling. If $p_k = p \to \infty \ \forall k = 1, ..., K$, then we reproduce the max pooling. Thus, we use GeM pooling layer to have a more flexible

use of the different means, and not only the previous two ! Indeed, $p = 1$ corresponds to the case of the arithmetic mean, $p = 2$ the quadratic mean, $p = -1$ the harmonic mean and $p \to 0$ the geometric mean. Moreover, the great advantage of GeM is that its parameters can be trained using back-propagation in a differentiable fashion.

Question 13: What operation is the layer *model_1d.adpool* doing?

If we take the same notation as before, we can write that *model_1d.adpool* is doing the following operation:

$$f^{(g)} = [f_1^{(g)}, ..., f_K^{(g)}]^\top \qquad \text{with } f_k^{(g)} = \left(\frac{1}{\mathcal{X}_k} \sum_{x \in X_k} x^3 \right)^{\frac{1}{3}}$$

where X_k is the k-th 2Dconv filter, and $K = 512$ now.

Question 14 & 15: How does this model compare with *model 1c*, that was trained in the same dataset for the same task? – How does it compare to the finetuned models of 1b?

ResNet18 shows worse results than AlexNet on a average of 11 images. This must be checked on a larger set but, it seems that model 1d has not as good performance as our previous model. For some images, ResNet18 outperforms AlexNet. However, it seems that the feature representation is not as rich as previously. This is surely do to the number of parameters that are learned during training (about 18 millions now). For example, top results for query 16 are really bad (see figure 7). The clustering has been improved in a sense that the different clusters are more disjoint / distant from one another. However, there is still a lot of common areas shared by the clusters, and also a lot of outliers. This can be linked with what we just said: some images are very difficult to analyse and lead to poor results (outliers), but, in general, clusters are more distant and thus the first top results for a given query correspond, in average, to more good matches (see figure 6).

Regarding the comparison with part 1b, ResNet18 is better than the 2 first models trained. Clustering is clearly less confuse and, on average, the AP is 5 points higher.

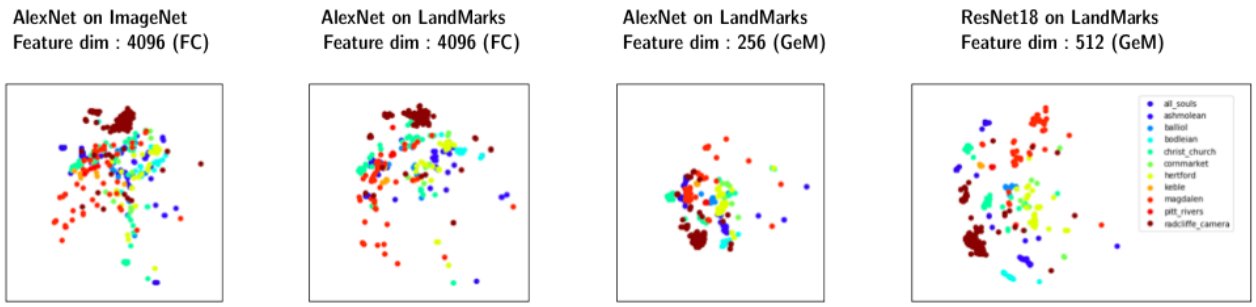

Figure 6: t-SNE clustering for the 4 models used so far.

	Images											
	1	6	11	16	21	26	31	36	41	46	51	Mean
Model 1a	10.96	13.04	12.73	25.47	44.30	14.51	61.62	47.65	5.49	69.5	65.77	33.73
Model 1b	31.08	28.93	24.67	**67.69**	55.58	15.72	62.50	68.42	**27.57**	83.76	76.99	49.36
Model 1c	39.22	61.40	**69.01**	49.89	**68.10**	31.41	70.73	**92.14**	17.03	89.58	**89.22**	**61.61**
Model 1d	**48.20**	**84.46**	39.82	1.28	32.03	**47.47**	**73.36**	70.76	11.38	**100**	85.47	54.02

Table 3: Average Precision obtained on 11 images for the 4 models used so far. Model 1a: AlexNet + ImageNet + 4096FC - Model 1b: AlexNet + LM + 4096FC - Model 1c: AlexNet + LM + 256 GeM - Model 1d: ResNet + LM + 512 GeM

Figure 7: Comparison of top results on query 16 between AlexNet with GeM 256 (top) and ResNet18 with GeM 512 (bottom)

1.5 PCA Whitening

Question 16: What can we say about the separation of data when included unlabeled images?

Labeled clusters separation is worsened by the use of unlabeled data.

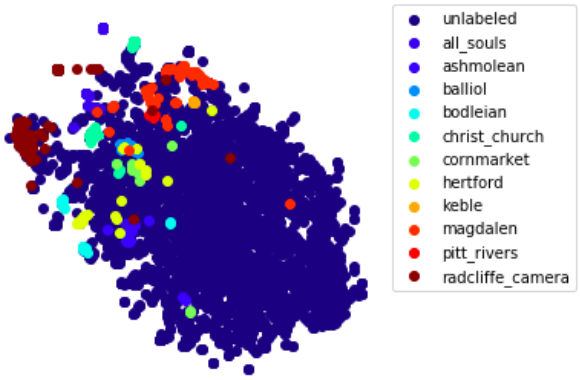

Figure 8: t-SNE visualization with unlabeled data.

Question 17: And the distribution of the unlabeled features?

The distribution of the unlabeled features is very diffuse and dense. Points seem to cover the entire 2D space. Most of them are disjoint from the labeled clusters, but some others not. This suggests two things: first, most of unlabeled images are unrelevant to the retrieval task. But, some others could be useful. Usually, one may distinguish the distributions followed by labeled and unlabeled data (a lot of approaches consider the contribution of the unlabeled examples by using a generative model for the classifier and employing EM algorithm to model the label estimation or parameter estimation process). However, unlabeled data may belong to different irrelevant *and* relevant classes. This is a critical point that we'll see just after.

Question 18: How can we train a model to separate labeled from unlabeled data?

This question is more subjective than the previous ones. We do not guarantee to bring an exhaustive answer to it, but the next developments aim at organising the information we gathered and what we understood. We try to make a connection with image retrieval.

First, we would like to refer to the article from Zhi-Hua Zhou (2004), "Learning with Unlabeled Data and Its Application to Image Retrieval". The author provides useful insights on how to handle unlabeled data for Instance-Level Image Retrieval (ILIR). The starting point is that only a limited number of labeled training examples are available since obtaining the labels require human effort. In fact, we can see the retrieval challenge as a machine learning problem, which attempts to train a learner to classify the images in the database as two classes, *i.e.* positive (relevant) or negative (irrelevant). Some techniques based on *co-training* for exploiting unlabeled data in ILIR exist, with two learners that gives a rank to each unlabeled images. We detail next the principle of SSAIRA algorithm (Semi-Supervised Active Image Retrieval with Asymmetry) proposed by Zhou et al. (2006), "Enhancing Relevance Feedback in Image Retrieval Using Unlabeled Data". This method integrates semi-supervised learning and active learning into the relevance feedback process (process where a user judges if results query are relevant or not). Figure 9 provides a pseudo code of co-training technique.

Cotraining

1. Define two separate nonoverlapping feature sets for the labeled data, X_l.
2. Train two classifiers, f_1 and f_2, on the labeled data, where f_1 is trained using one of the feature sets and f_2 is trained using the other.
3. Classify X_u with f_1 and f_2 separately.
4. Add the most confident $(x, f_1(x))$ to the set of labeled data used by f_2, where $x \in X_u$. Similarly, add the most confident $(x, f_2(x))$ to the set of labeled data used by f_1.
5. Repeat.

Figure 9: Co-training method of semi-supervised classification. X_u denotes the unlabeled data. Source: Han et al. (2012) "Data Mining (Third Edition)"

"Each learner will give every unlabeled image a rank which is a value between -1 and $+1$, where positive/negative means the learner judges the concerned image to be relevant/irrelevant, and the bigger the absolute value of the rank, the stronger the confidence of the learner on its judgement. Then, each learner will choose some unlabeled images to label for the other

learner according to the rank information. After that, both the learners are re-trained with the enlarged labeled training sets and each of them will produce a new rank for the unlabeled images. The new ranks generated by the learners can be easily combined, which results in the final rank for every unlabeled image. Then, unlabeled images with top ranks are returned as the retrieval results which are displayed according to descending order of the real value of their ranks."

For now, we explained how to use unlabeled data for image retrieval. Did we fully answered the question, namely, how to train a classifier that separates labeled from unlabeled data ? Not really. It has been shown that when the model assumption does not match the ground-truth, unlabeled data can either improve or degrade the performance, depending on the complexity of the classifier compared with the size of the labeled training set. For example, if the distribution of the unlabeled data is different from that of the labeled data, using unlabeled data may degrade the performance (Tian et al. 2004). Moreover, introducing unlabeled data (potentially false positives) into the query expansion, a technique that has shown performance boost for image retrieval for the recent years, can be problematic[1]. We thus want to obtain an unlabeled set, which will be a mixture of positives and negatives samples from the top of retrieval shortlist, in order to increase our information about relevant samples. **There is thus a necessity to automatically distinguish between labeled and unlabeled images**. A way to do it is to add a class "Other" to the landmark classes and train the network with $n + 1$ classes. The problem with this approach is that the distribution of the unlabeled images, as we saw, is very diffuse and intersects that of the labeled images. So, there's a risk that the network will predict of lot of unlabeled images, certainly even too much, and plausibly overfits a lot.

We talked about co-training for distinguishing between labeled and unlabeled data. Other approaches to semi-supervised learning exist like two-step strategies: first, identify a set of reliable negative images from the unlabeled set, then build a classifier by iteratively applying a classification algorithm, *i.e.* EM or SVM. We can also talk about PU-learning (Positive and Unlabeled learning[2]), a two-class classification problem which is part of cotraining.

In our opinion, we think that we should keep the bone of our network but only consider 2 classes: labeled (0) and unlabeled (1). Then, get feature representation for each image and apply a classifier like non-linear SVM on it. Indeed, the 2D representation shows that there is some global different spatial structure between the two classes. Using a RKHS could help projecting the feature space into another more discriminative space (of potentially higher dimension). Using large margin SVM, which prevents too much overfitting, could be helpful in a sense that, if a point belongs to one of the 2 clusters, it will belong to it with high confidence as the separating hyperplane would be very discriminative. One could also use the same classifier but with the previous classes +1 for "other".

[1]The baseline of query expansion is the following: find top results from original query, spatially verify those results, generates new queries from it and get new results.

[2]Li et al., "Learning from Positive and Unlabeled Examples with Different Data Distributions"

1.6 Finetuning on Landmarks for retrieval

Question 19: Compare the plots with unlabeled data of the model trained for retrieval (with triplet loss) and the model trained for classification of the previous subsection. How do they change?

First thing, one may highlight the performance gain brought by the triplet loss !

	Images											
	1	6	11	16	21	26	31	36	41	46	51	Mean
Model 1a	10.96	13.04	12.73	25.47	44.30	14.51	61.62	47.65	5.49	69.5	65.77	33.73
Model 1b	31.08	28.93	24.67	**67.69**	55.58	15.72	62.50	68.42	27.57	83.76	76.99	49.36
Model 1c	39.22	61.40	69.01	49.89	68.10	31.41	70.73	92.14	17.03	89.58	89.22	61.61
Model 1d	48.20	**84.46**	39.82	1.28	32.03	47.47	73.36	70.76	11.38	100	85.47	54.02
Model 1f	**58.24**	84.27	**71.42**	0.76	**82.15**	**65.96**	**77.13**	**94.78**	**55.24**	**100**	**98**	**71.63**

Table 4: Average Precision obtained on 11 images for the 5 models used so far. Model 1a: AlexNet + ImageNet + 4096FC - Model 1b: AlexNet + LM + 4096FC - Model 1c: AlexNet + LM + 256 GeM - Model 1d: ResNet + LM + 512 GeM - Model 1f: ResNet18 + LM + GeM + Triplet Loss

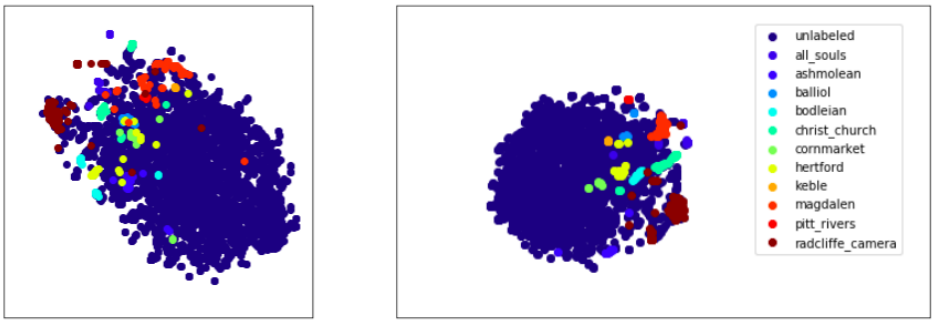

Figure 10: t-SNE comparison with unlabeled data. Left: ResNet trained for classification. Right: ResNet trained for retrieval.

From figure 10, we notice that first, labeled clusters are way better separated. There is also a clear diminution of the number of outliers. Regarding the plots with unlabeled data, we see that unlabeled data 2D distribution is more homogeneous and concentrated. Still, labeled clusters are included in unlabeled cluster. Some classes (and the same as before) remain hard to distinguish from unlabeled data.

1.7 Data augmentation and multi-resolution

Question 20: What is the difference in AP between a model that has trained with and without data augmentation?

On average, even if we didn't test on the whole dataset, performance are better with data augmentation than without. Indeed, the average AP on the same 11 images as before (see table 5) is about 3 points higher, which is huge [3] for this level of sophistication of the two previous models. It is interesting to see that more people/faces are present on the top results

[3] To confirm on the whole dataset.

queries. This is a direct consequence of cropping. As said in the notebook, data augmentation generally increases performance but, in some cases like cropping, may lead to poor results on few images, like image 11. It is also interesting to see that despite data augmentation, query 16 has still a dramatic AP ! This seems to be linked with the dimension of feature vectors. Query 11 is a complex image as it contains a lots of details and is very zoomed in. Consequence: the contribution of the stained-glass window in the top results feature representation is usually low and very local. Despite their good performance, the 2 last networks are still unable to correctly retrieve some images like image 11 due to their more compact representation (see figure 13).

	Images											
	1	6	11	16	21	26	31	36	41	46	51	Mean
Model 1a	10.96	13.04	12.73	25.47	44.30	14.51	61.62	47.65	5.49	69.5	65.77	33.73
Model 1b	31.08	28.93	24.67	**67.69**	55.58	15.72	62.50	68.42	27.57	83.76	76.99	49.36
Model 1c	39.22	61.40	69.01	49.89	68.10	31.41	70.73	92.14	17.03	89.58	89.22	61.61
Model 1d	48.20	84.46	39.82	1.28	32.03	47.47	73.36	70.76	11.38	100	85.47	54.02
Model 1f	58.24	84.27	**71.42**	0.76	82.15	65.96	77.13	94.78	55.24	100	**98**	71.63
Model 1g	**58.24**	**91.52**	59.88	2.63	**87.99**	**86.77**	**82.77**	**94.81**	**58.36**	**100**	97.90	**74.62**

Table 5: Average Precision obtained on 11 images for the 6 models used so far. Model 1a: AlexNet + ImageNet + 4096FC - Model 1b: AlexNet + LM + 4096FC - Model 1c: AlexNet + LM + 256 GeM - Model 1d: ResNet + LM + 512 GeM - Model 1f: ResNet18 + LM + GeM + Triplet Loss - Model 1g: Model 1f with Data Augmentation

Question 21: What about the clustering? Why do you believe some of the classes have not been adequately clustered yet?

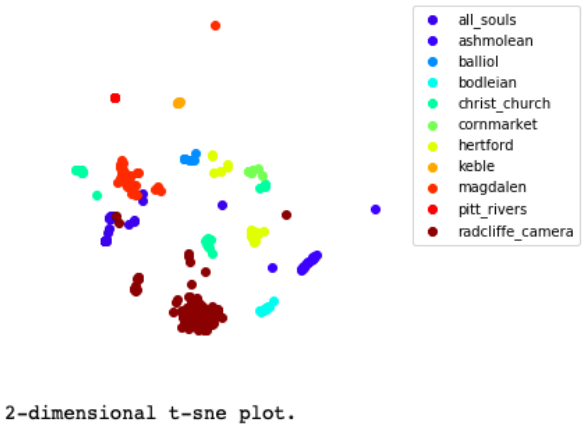

Figure 11: t-SNE visualization with data augmentation.

The clustering with no unlabeled data is tremendous comparing to what we had before ! Clusters are spread on the whole space, with few outliers, and distant from one another. 2 of the classes have not been adequately clustered yet: *Christ-Church* and *Hertford*. This is due to the strong similarity of those 2 landmarks with other landmarks. For example, Christ Church tour is very similar to Radcliffe Camera. Moreover, the walls surrounding its courtyard are also similar to All Souls College ones. Depending on the view, those structural similarities can be very confusing even for the naked eye.

Question 22: What other data augmentation or pooling techniques would you suggest to improve results? Why?

Let's first start with other data augmentation techniques. The data augmentation protocol applied here is quite complete. Nevertheless, one may add some other tricks to boost the performance. The first one is a little long in the tooth as it was introduced by Krizhevsky et al. (2012) with AlexNet. It consists in randomly adding multiples of the principle components of the RGB pixel values throughout the dataset in order to capture the invariance in illumination and color. Specifically, the authors performed PCA on the set of RGB pixel values throughout the ImageNet training set. To each training image, they then added multiples of the found principal components, "with magnitudes proportional to the corresponding eigenvalues times a random variable drawn from a Gaussian with mean zero and standard deviation 0.1". According to the authors, this scheme reduced the top-1 error rate by over 1% on ILSVRC-10 dataset.

Then, we could also advise to apply shearing. This technique was used by Tensmeyer et al. (2017)[4] and has proved to be the "state-of-the-art on the RVL-CDIP dataset". When images are sheared (either horizontally or vertically), the relative locations of layout components are perturbed, but unlike rotation transforms, either the horizontal or vertical structure is preserved. According to the authors, "shearing best models the types of intra-class variations in [image] datasets".

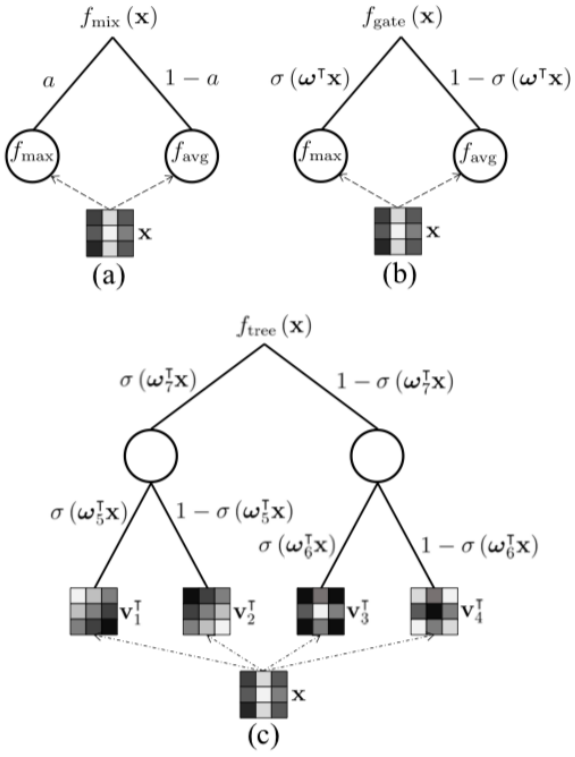

Figure 12: Illustration of pooling operations: (a) mixed max-average pooling, (b) gated max-average pooling, and (c) Tree pooling (3 levels in this figure). Source: Lee et al. (2017) "Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree".

[4] In "Analysis of Convolutional Neural Networks for Document Image Classification".

Before wondering about if those techniques could be helpful, we would also add some suggestions about pooling techniques. To do that, we refer to the work of Lee et al. (2017) "Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree". According to the authors, their technique "provides a boost in invariance properties relative to conventional pooling and set the state of the art on several widely adopted benchmark datasets; they are also easy to implement". Briefly, the first contribution of the article is to learn a non-linear combination of max and average pooling (figure 12.b). The second is to allow the pooling operations that are being combined to themselves be learned (tree-pooling) in a differentiable fashion (i.e parameters $\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}, \mathbf{v_4}, \omega_5, \omega_6, \omega_7$ on figure 12.c). Specifically, the learning is performed within a binary tree in which each leaf is associated with a learned pooling filter. Note that the most performing configuration was to apply first a gated max average pooling and then a 2-level tree pooling.

Now, would this transformation be helpful for our retrieval task ? Let's start with pooling. Testing the described method is worth it as the actual top performing pooling technique is GeM pooling, which is a flexible pooling layer with learnable parameters. It is thus structurally similar to Lee et al.'s generalization. Regarding data augmentation tricks, one may emphasise the potential benefit of shearing. Indeed, there are many configurations where the angle and the view characterizing the pictures are such that the perspective is strong. Thus, a transformation that would allow to "anneal" this effect may generate more similarities between images from the same class.

At last, we could talk about GAN's and neural style transfer but these approaches are computationally very expensive (GAN's are also hard to train) and potentially not suited for our problem as, for instance, GAN's generate images that are more artistic than realistic. This would potentially not bring much further information to the image comparing to the effort put in the training.

Figure 13: Illustration of perspective.

Question 24: Why using a larger architecture results in a higher AP? Is this always going to be the case?

In table 6, we compare the AP obtained on 11 queries for the 7 models used so far. We see that training with ResNet50 leads to the best average results [5]. If we compare models 1g and 1h, the only difference lies in the depth of the architecture (as we both do DA, GeM and use a ranking loss). In such conditions, the feature space size went up from 512 to 2048. The number of parameters increased from roughly 18 to 25 millions. Using ResNet50 allows to build more complex representation of the images thanks to a bigger amount of non-linearities. **However, enlarging the architecture does not automatically guarantee higher AP**. Complexity

[5] To confirm on the whole dataset.

does not solve everything. This is of particular interest when choosing between ResNet18, 50 or 101. Through their experimental results, Gordo et al. (2017), "End-to-end Learning of Deep Visual Representations for Image Retrieval" well illustrate this problematic of complexity. In fact, what really matters is the architecture as a whole (pooling layers, query expansion, diffusion, etc...), the data (data augmentation is determinant), and how the problem fits with the task of image retrieval (triplet loss). This last point is crucial and enhanced by the authors who compare ResNet101 and VGG16: "the model based on ResNet101 outperforms the model based on VGG16. This gap, however, is not as significant as the improvement brought by the training [with a triplet loss on Landmarks dataset]". In other words, the whole methodology that is applied in this paper - and thus similar to this TP, can not be entirely replaced by *only* a very complex neural network like ResNet101. **We could wonder if, keeping the same level of sophistication as model 1g or model 1h, increasing the depth would lead to better results ?** Probably. ResNet101 would certainly provide better results for this retrieval task. However, the performance gain would be inferior to the one brought by ResNet50 compared to ResNet18. We should not forget that 45 million of parameters (for ResNet101) is huge and favour overfitting if we do not proceed to smart data augmentation. Moreover, some architectures are better suited for image retrieval. For example, we didn't find, in the literature of instance-level image retrieval, any paper using Inception(maybe we're wrong?).

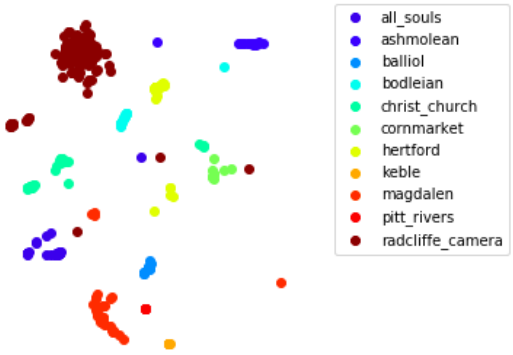

Figure 14: t-SNE visualization with ResNet50 and data augmentation.

	Images											
	1	6	11	16	21	26	31	36	41	46	51	Mean
Model 1a	10.96	13.04	12.73	25.47	44.30	14.51	61.62	47.65	5.49	69.5	65.77	33.73
Model 1b	31.08	28.93	24.67	**67.69**	55.58	15.72	62.50	68.42	27.57	83.76	76.99	49.36
Model 1c	39.22	61.40	69.01	49.89	68.10	31.41	70.73	92.14	17.03	89.58	89.22	61.61
Model 1d	48.20	84.46	39.82	1.28	32.03	47.47	73.36	70.76	11.38	100	85.47	54.02
Model 1f	58.24	84.27	**71.42**	0.76	82.15	65.96	77.13	94.78	55.24	100	**98**	71.63
Model 1g	58.24	**91.52**	59.88	2.63	87.99	**86.77**	82.77	94.81	58.36	100	97.90	74.62
Model 1h	**92.80**	80.91	55.38	10.33	**96.56**	81.79	**95.59**	**100**	**64.59**	**100**	**99.08**	**79.73**

Table 6: Average Precision obtained on 11 queries for the 7 models used so far. Model 1a: AlexNet + ImageNet + 4096FC - Model 1b: AlexNet + LM + 4096FC - Model 1c: AlexNet + LM + 256 GeM - Model 1d: ResNet + LM + 512 GeM - Model 1f: ResNet18 + LM + GeM + Triplet Loss - Model 1g: Model 1f with Data Augmentation - Model 1h: ResNet50 + LM + GeM + Triplet Loss + Data Augmentation