Deep learning image search

Josef Jon <xjonjo00@stud.fit.vutbr.cz>
Tomáš Sýkora <xsykor25@stud.fit.vutbr.cz>

December 29, 2017

1 Introduction

Our goal was to build an architecture for searching images same or similar to the one used as a query image. We could say there are two subsets of this problem of image search or image retrieval. The first one is when having a certain image with a specific object (e.g. Cathedral of Saints Peter and Paul in Brno) taken in some angle, scale, orientation etc. and the goal is to find images containing the same object but taken from a different perspective. Another subset of the image search problem is to find the most similar images to the one used as a query one. That means if there is a church in the image, we would like to find other images with a church (the same church or just a similar one). Our work is focusing on this second problem.

2 Theoretical background

For searching images we need some form of representation of them so that we could easily compare them to each other and choose which ones are more similar than others. For that many forms of hash or feature vectors are used as distances between them can be computed. There are two main categories for obtaining image representations: hand-crafted features and deep architectures for creating image features. We chose deep architectures for their popularity and robustness. There are lots of related works on this topic that inspired us. We found important theoretical information in papers working on face recognition tasks [SKP15] as it is very similar problem to the one we worked on and basically the same techniques are used there. There are also

much more sophisticated approaches like in [QPYM17], from which we also took our inspiration at least partly.

Our deep architecture uses implementation of a triplet loss function also used in the works mentioned above and is trained in this manner: There is a convolutional neural network that learns to predict a feature vector representation for a given image. During the training it takes three images in one iteration. The image triplet consists of a query image (anchor image), an image from the same class as the anchor (positive image) and an image from a different class (negative image). The CNN predicts a feature vector for each of them. The loss function is based on the idea that the distance between the anchor and positive image feature vectors should be much more lower than the distance between the anchor and negative image as in the equation 1. During the training this loss function is minimized.

$$triplet\_loss = \sum_{i}^{N}[||f(x_i^{anchor}) - f(x_i^{positive})||_2^2 + ||f(x_i^{anchor}) - f(x_i^{negative})||_2^2 + \alpha] \tag{1}$$

3 Training

The figure 1 represents the whole architecture used during the training. For the CNN a residual network was used [HZRS15] with pretrained weights on an ImageNet dataset [DDS+09]. We used just the resnet part of the network after which we connected flatten and two fully connected layers of the size 512. Output was a feature vector of 512 values which was normalized with L2 normalization layer. We also tried to select different layers from the trained ResNet, best results were achieved by using all of the layers.

For training we used Places365 dataset [ZLK+17] which is a huge dataset for scene recognition consisting of 500 classes with 5000 images each. As our computational resources were limited (we used metacentrum.cz servers with an NVidia Tesla K20m graphic card), we trained only on 20 and then on 50 classes randomly selected from the whole dataset. Training consisted from two epochs each of which trained just the fully connected layers (training the rest of the pretrained ResNet led to worse results). For both epochs Adam optimizer was used, at first with 0.0004 training step, then with 0.00004 training step. The whole training took approximately one day on the dataset with 50 classes.

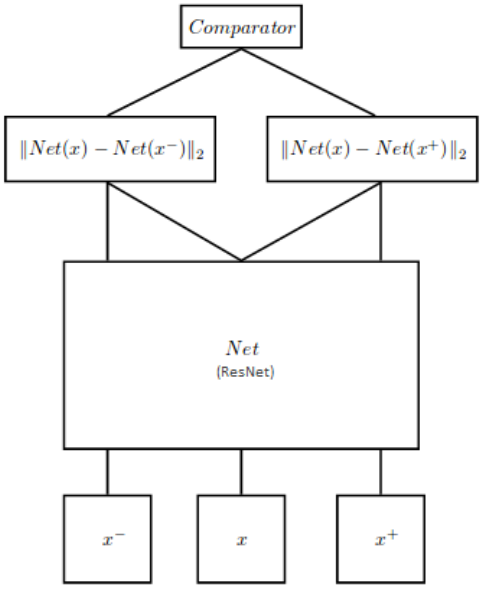

Figure 1: The whole architecture for training.

The problem of training the triplet loss is selection of suitable triplets. To speed up the convergence, we want the positive images to be as least similar to the anchor as possible (negative images vice versa). Choosing such triplets is called hard mining. It is because the model learns quite fast to distinguish between 'easy' triplets and stops training on them. This problem occurs more often while training face recognition models as all the classes are faces. Although we didn't train a face recognition model, we tried to do such hard mining. Before backpropagation the predicted feature vector values were checked whether they fulfill certain condition, if so, they were chosen for backpropagation otherwise the triplet wasn't considered for training. Our attempts resulted in worse result than without the hard mining technique that's why we didn't use it in out final training (results in the next sections are without hard mining).

4 Evaluation

We used mean average precision (MAP) metric to evaluate our model. Firstly we trained the model on 20 classes randomly chosen from the whole dataset. The MAP resulted with 0.85 value here. Then we trained the model with a 50 class subset of the dataset. Here the MAP was worse and resulted with 0.60. Although the results may seem not so great (but not bad), the model behaved very reasonably. It can be seen in 2. Although some images weren't selected correctly by the model, it was often hard to say which class should they belong to even by humans. There were lots of indistinguishable classes an images in the dataset like desert - badlands, volcano - mountain, marketplace - supermarket, forest - forest path and many others. That's why it was impossible to compute precise accuracy in some cases. On the other hand there were some images incorrectly labeled with a wrong class (the last image of the first row of the figure 2 - there is a model of a train instead of a sushi bar as in the query image).

Figure 2: Examples of outputs demonstrating inter-class variability of the dataset.

The figure 2 demonstrates that the output of the system makes sense

although it is not correct according to the dataset ground-truth.

The project also contains a web API to demonstrate the functionality of the model. A query image can be uploaded there and a list of similar images is shown.

5 Conclusion

Our model performed search with 0.6 mean average precision value on 50 classes, which is a reasonable result considering the format and quality of the used dataset. Manually checked image searches confirmed that although some of the found images weren't correct according to the dataset labels, they actually made sense. The work on this project was divided into data preparation and evaluation, which was done by Josef Jon, and into model creation and training which was done by Tomáš Sýkora.

References

[DDS+09] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[QPYM17] Zhaofan Qiu, Yingwei Pan, Ting Yao, and Tao Mei. Deep semantic hashing with generative adversarial networks. August 2017.

[SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832, 2015.

[ZLK+17] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.