Visual Search: Computer Vision II

Pedro Guijas Bravo
University of Coruña
A Coruña, Spain
p.guijas@udc.es

***Abstract*—Methodology proposed for image retrieval consists of two main steps: calculating the feature vector for both the query image and the reference dataset, and ranking the embeddings by cosine similarity to determine the most similar images. For this report, the *ResNet16* model was selected as a pre-trained model to generate high quality feature vectors initially. The main goal will be to achieve the best possible classification, with fine tuning of the given pre-trained model and customized loss terms (contrastive loss and triplet loss). Finally, the evaluation of the results will be based on information retrieval metrics, obtaining really good results for the flower classification domain.**

***Index Terms*—Convolutional Neural Networks, Embeddings, Ranking, Similarity, Contrastive loss, Triplet loss.**

I. Introduction

In this report we will develop and analyze a method for the identification of the flower family from a picture by means of **visual search**. For this purpose, an already trained convolutional neural network will be used, in this case ***ResNet18*** [1]. Starting from this network, only the part of the network used for feature extraction will be used, analyzing first the performance and quality of the embeddings extracted with the convolutional network on the **102 Oxford Flowers Dataset** [2].

After measuring the performance of the embeddings obtained for the ranking process of the most similar images, we will try to improve the results. For this, a fine tuning of the network will be performed with specific error functions to improve the quality of the feature vectors. Note that this process will be slightly different from the training of a classifier, where the highest precision is sought in the network (for example by minimizing the entropy of the last softmax), however, in this case, error terms that try to maximize the distance between embeddings images of different classes and minimize the same between samples of the same class will be used.

II. Proposed methodology

The search process, as mentioned in the introduction, will consist of 2 steps:

- Feature vector computation for the query image and the images in the reference dataset.
- Ranking by similarity of the embedings to determine the most similar images. Specifically, cosine similarity will be used, since others such as Euclidean distance may cause similar vectors with different modules not to be ranked as the most similar.

As can be assumed, the main part of the work will be to obtain the best possible feature vectors or embeddings, since the performance of the search process depends entirely on them.

Since it is really important to start from a pre-trained model with potential, multiple alternatives have been proposed such as *AlexNet* [3], *MobileNet* [4] or *VGG* [5], however, the *ResNet* was the model selected, specifically the *ResNet16*. This decision was mainly motivated by its high performance and its great rendering capabilities.

As for the evaluation of the results obtained, information retrieval metrics [6] will be applied, since essentially the task to be performed is of this kind. In this way, the evaluation process will take into account the labels of both the query image and the reference images, seeking that the most similar images are first of all those that correspond to the same class. The query images will be the train dataset, while the reference dataset will be the test dataset. The metrics to be computed are the following:

- **Adjusted Mutual Information (*AMI*)**: It is an adjustment of the Mutual Information (MI) score to account for chance. Note that mutual information measures the statistical dependence between two variables. Thus, MI is used to evaluate the quality of the generated embeddings.
- **Normalized Mutual Information (*NMI*)**: Normalization of the Mutual Information score to scale the results between 0 (no mutual information) and 1 (perfect correlation).
- **Mean Average Precision (*MAP*)**: The $precision@K$ computes the % of relevant images (same label as query) in top K. Average Precision (*AP*), computes the average of $precision@K$ for each ranking k. And finally, *MAP* computes the *AP* for each query.
- **Mean Reciprocal Rank (*MRR*)**: Let k be the ranking position of the first relevant document. The Reciprocal Rank (*RR*) will be $\frac{1}{k}$. Finally *MRR* will imply averaging for all queries.
- **R-precision**: It is defined as the $\frac{r}{R}$, being the ratio between all relevant documents retrieved up to the rank that equals the number of relevant documents.
- **Mean Average Precision at rank r (*MAP@r*)**: $MAP@r$ combines the ideas of MAP and R-precision. This is because R-Precision does not take into account the ranking of correct retrievals.
- **Precision at 1**: As its name suggests, it will calculate the number of times the first element of the ranking

corresponds to the same tag of the query image.

Since we will only focus on obtaining the best possible ranking, non-information retrieval metrics such as AMI or NMI will be considered less important, considering the rest more important, especially *MAP@r*.

In summary, the methodology followed will be briefly described below:

- **Search for a pre-trained model**: As mentioned above, the *Resnet16* is the selected pretrained model.
- **Metric selection**: Metrics already described in this same section.
- **Measuring the performance of the pre-trained model**: Measurement of the performance of the feature vectors obtained with the initial model in the ranking.
- **Fine tuning of the model**: This will be the most complex phase and where the main work resides. By fine-tuning the pre-trained model using different techniques, we will try to improve the embeddings and consequently improve the recovery of more similar images. Since this is the bulk of the work, the techniques to be used will be explained in the next section.
- **Measuring the performance of the finetuned model**: Again, performance will be estimated for each improved version of the *ResNet16* architecture.
- **Global analysis of results** Finally, all the results obtained will be analyzed qualitatively and quantitatively, emphasizing the improvements made.

III. Experiment description

As mentioned, the major part of the experiment will be the attempt to improve the quality of the feature vectors obtained initially, as well as to obtain these first results.

In order to try to improve these models, the network will be fine-tuned with two different error terms, which will be briefly explained below. Also, after the implementation and re-training of these models, we will try to maximize the improvements by testing with different optimizers (ADAM and SGD), as well as applying data augmentation.

The first error term to use is **Contrastive loss**. It aims to bring all instances of similar classes closer together in the n-dimensional embedding space, as well as to push away instances of different classes. Concretely, the optimization term can be seen in Formula 1, being the inputs two samples available in the batch. Thus, when the entries are of the same class ($\hat{y}_{ij} = 1$), we will seek to minimize the distance between samples ($L_{contrastive}(i,j) = D_{ij}^2$). In the opposite case ($\hat{y}_{ij} = 0$), we will try to minimize the negative taking into account a certain margin α ($L_{contrastive}(i,j) = [\alpha - D_{ij}^2]_+$).

$$L_{contrastive}(i,j) = \hat{y}_{ij} D_{ij}^2 + (1 - \hat{y}_{ij})[\alpha - D_{ij}^2]_+ \quad (1)$$

The next error term to be applied is **Triplet loss**. Again its formulation can be seen in Formula 2. The input of this error term will be an anchor sample, and two examples: a clear negative and a positive one. What is sought is to contextualize the error function with distances to these samples, seeking the clear creation of 2 distinct clusters in the space of the feature vector.

$$L_{triplet}(a,p,n) = [D_{ap} - D_{an} + \alpha]_+ \quad (2)$$

As can be expected, for the task to be solved, these error terms will have a clearly better performance than other more classification-oriented optimization algorithms such as *Cross-entropy*.

In addition, the use of two really important elements should be highlighted:

- **Distance metric**: The cosine similarity will be used for reasons already explained.
- **Reducers**: To obtain the error term in the batch, the error term will be averaged ignoring negative error terms.
- **Miners**: In the case of triplet loss, representative and clear examples should be selected in order to compute the error term. In this way, the use of miners in each batch for the network optimization algorithm must be emphasized. Specifically, the selection criterion for these triplets used will be *semihard*, which, unlike *all*, will select really representative examples.

IV. Results

In this section, the results for the different approaches followed will be discussed. These are listed in Table I together with the values of the different metrics obtained.

TABLE I: Results of different Resnet models

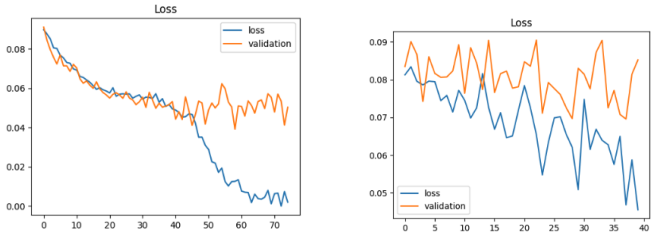
	AMI	NMI	MAP	MAP@r	MRR	Precision at 1	r precision
default resnet18	0.425	0.823	0.655	0.528	**0.991**	0.982	0.593
contrastive loss and sgd	0.351	0.806	0.730	0.609	0.984	0.982	0.667
triplet loss and sgd	0.351	0.806	0.732	0.612	0.984	0.982	0.671
contrastive loss and adam	0.506	0.857	0.632	0.516	0.890	0.839	0.588
triplet loss and adam	**0.672**	**0.905**	**0.950**	**0.919**	0.982	**0.983**	**0.928**
triplet loss, adam and data augmentation	0.521	0.856	0.905	0.859	0.942	0.928	0.874

Being brief and concise with the results, the best values obtained for each metric are highlighted in bold, showing how clearly the best method has been the fine-tuning with Triplet loss, ADAM, and without data augmentation. The values obtained are practically perfect, solving the task with a really high performance.

The data augmentation approach has deteriorated the results, probably because of the additional complexity added to the problem (the original images were generally centered and zoomed to a similar degree.). These results are actually similar to the other approaches using other error terms or optimizers.

Also note the quality of the results for the model with the initial weights, having an acceptable performance and even the best value in the *MRR* metric, demonstrating a great performance and being practically on a par with the rest of the approximations except for the best, already mentioned.

Before continuing with the quantitative results, it is worth mentioning the unclear convergence and the lower train error terms (contrastive and triplet) when using SGD. On the other hand, ADAM has a non-existent convergence despite obtaining the best results with ADAM and triplet loss. This differentiation can be seen in Figure 1.



(a) SGD (b) ADAM

Fig. 1: *ResNet18* fine tuning with different optimizers using triplet loss as error metric

Finally, from a qualitative point of view, Figure 2 shows the ranking by similarity given an image query (from left to right), showing the label of the ranked flowers. It can be seen how the overall performance is good, with the top 9 rank being correct for all models, including the original *ResNet16*. However, after fine-tuning, the models have now introduced only samples of the correct flower (class 0) in the top 20, clearly improving the ranking of the initial model. As for the best-performing approach, there is no noticeable difference as both behaviors are adequate. Nevertheless, there are slight changes such as the top 1, which gives the impression of a better ranking quality for the triplet loss with ADAM.

It should be noted that the ranking for the initial model is not at all out of focus, but its quality is clearly inferior to the other two. This, in the metrics, can be reflected in the *MAP@r*, one of the most important metrics for estimating performance in ranking problems.

(a) Query image

(b) *Resnet18* with initial weights

(c) *Resnet18* retrained with contrastive loss and SGD

(d) *Resnet18* retrained with triplet loss and ADAM

Fig. 2: Visual search results with different versions of the model

V. Conclusions

In conclusion, first of all, we highlight the great initial performance of the *ResNet16* for feature extraction.

Regarding the best model obtained, its performance is extremely good, outperforming the rest and obtaining practically perfect results.

Finally, as future work, the generalization of the problem to different types of images could be analyzed, but surely using more powerful architectures for the extraction of features. In the case of the flowers that are being used, a trivial feature such as the mean color of the image can provide traces for a moderately adequate classification. Therefore, it is concluded that in the given domain of flowers (generally distinct flowers), the classification may be more trivial than in other domains, where classes whose differences are not so trivial have to be segmented.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[2] "102 category flower dataset." [Online]. Available: https://www.robots.ox.ac.uk/ vgg/data/flowers/102/

[3] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," 2014.

[4] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[6] H. Turtle and J. Flood, "Query evaluation: strategies and optimizations," *IP&M*, vol. 31, no. 6, pp. 831–850, 1995.