AML Competition Report

Leonardo Venturoso
Università degli Studi di Trento
leonardo.venturoso@studenti.unitn.it

Mithat Sinan Ergen
Università degli Studi di Trento
mithatsinan.ergen@studenti.unitn.it

Cesare Barbera
Università degli Studi di Trento
cesare.barbera@studenti.unitn.it

Gabriele Ghisleni
Università degli Studi di Trento
gabriele.ghisleni@studenti.unitn.it

1 INTRODUCTION & MOTIVATION

In this work, the main goal was to address the image retrieval problem by using an image search algorithm. In other words, given a query image in a first set, this should match the images in another set, the gallery. Specifically, the type of end result should be a list of ranked matches (top-k images) between input and output images. This type of algorithms generally includes two main stages: *filtering*, that is where a specific technique classifies database images according to their similarity to the query and *re-ranking*, where a limited number of the most similar first stage images are investigated deeply and re-classified. At present, several deep learning techniques and methods have been proposed for these two different stages. One of these is the use of global image representations based on convolutional neural networks (CNNs). CNN can be defined as a specific class of deep neural network, inspired by the organization of the Visual Cortex, usually applied to analyze visual imagery [6]. This algorithm can take in an image, assign weights and biases to specific objects present inside and differentiate one from the other. Unlike a multilayer perceptron used for classification purposes, this method, thanks to the filtering stage, considers also spatial and temporal dependencies in an image. As a result, it can generate efficient and fast similarity computation in the first stage (filtering) and perform better and more sophisticated fitting to the image dataset due to a significant reduction of the affected parameters [1, 5]. Thus, the main goal of the Convolution Operation is to extract high-level features from the input image by using a specific filter, also called kernel, and moving through different convolutional layers, extracting first basic information and then more complex ones to get all the complicatedness of the image. All these aspects and features convinced us to adopt this type of approach to solve the problem.

2 RELATED WORK

In this section, we describe the previous work done in the field that is related to our project.

Firstly, as said mentioned above, Convolutional Neural Networks(CNN) are formed to automatically and adaptively discover spatial hierarchies of features by using backpropagation on multiple building blocks. These building blocks include convolution layers, pooling layers, and fully connected layers. CNN is based on deep learning methods and it has become prominent in multiple computer vision tasks in various fields such as radiology and meteorology. There are many differences between CNN and conventional machine learning methods. First of all, CNN does not use manual feature extraction. Secondly, CNN architecture does not require expert opinions in every case. Also, CNN is dependent on big data since it has a myriad of parameters to estimate thus it requires considerably high computational resources compared to traditional machine learning methods. This results in the use of graphical processing units(GPU) while training CNN models[8].

Secondly, Kaiming et al. presents residual learning framework to solve the problem of training difficulty of deeper neural networks[2]. They reformulate the layers as learning residual functions by taking the layer inputs as reference. After the reformulation, they prove that residual networks are easier to optimize and increased depth in the network comes back as accuracy. They evaluate the results of residual nets up to 152 layers depth on ImageNet dataset and prove lower complexity and better results compared to VGG nets.

Thirdly, Shorten et al. explains in detail the reliance of CNN on big data and the overfitting problem[3]. It is mentioned that these problems can be solved by introducing more data to the network but more data might not be always available. It is when data augmentation becomes useful. Data augmentation is described as "a suite of techniques to enhance the size and quality of training datasets such that better Deep Learning models can be built using them". There are various algorithms to perform data augmentation such as geometric transformations, color space augmentations, kernel filters, mixing images, random erasing, feature space augmentation and neural style transfer. They have considerable proven impact on the performance of the models and on solving the overfitting problem.

3 PROBLEM STATEMENT & DEFINITION

To implement this type of algorithm we decided to follow a series of steps that we can generally summarize to these five: get and preprocess large amount of data, use trained convolutional neural networks as feature vector generators, implement feature vectors differences to calculate similarity and finally adopt a list of top-k images. Regarding the first steps, a large image set was required, so we perform a data augmentation to increase the number and the diversity of our training set by applying random transformation, we gathered data (JPG images) in a final folder and we preprocess them to fit the necessary input requirements of the nets, which are 224x224 resolution at least, RGB format, image values within the range 0 and 1 and specific mean and standard deviation values. Subsequently, we calculate feature vectors for the normalized images of a specific net and we calculate the similarity. Regarding the choice for the similarity parameter, we decided to adopt the cosine similarity and finally store the top-k lists. All steps were implemented using a CoLab notebook that supports GPU, Pytorch, that is an open source library and framework for machine learning purposes and specific residual networks (ResNet) for the neural network implementation.

4 METHODS

In this section we present the methods and the steps that we followed to complete the project. We will start by illustrating the image augmentation that we performed. After have manually created 50 classes with at least 15 images each we implemented different techniques that are used to create as many images as possible while keeping under control the trade-off between similarity and the data set dimension, since we need that those images differ one from another. These techniques principally consists of:

(1) Adding different kind of noises as: gaussian noise, salt and paper noise, paper noise, averageing blur, median blur, gaussian blur, bilateral blur, black hat, top hat.
(2) Change the colors of the images modifying: saturation, contrast, light, sharpe and introduce grayscale images.
(3) Change the proportion of the image doing operation as crop, dilatation, flip, superpixel, resize, open (focus) and change the padding.
(4) Highlight features as: edge detection, edge detect and canny, erosion and morphological gradient.

The main idea that we had to complete our task consists of using models that are trained and set up in a classification context as features extractor. Basically once the model is trained, we cut off the last layer so to access to the raw features data matrix which represent the feature extracted by the model and use those to compare different images.

In particular we trained the models on our custom data set, after the data augmentation it consisted of 50 classes containing specific buildings and having around 600 images each. We tried to train different models in order the reach the best accuracy on our validation set, for instance we choice some models that were not pre-trained, others having all the layers freezed except one and lastly others that were using all the layer and were pre trained.

Configure models that are able to extract feature from the images was the main part of the assignment, after have done that we just create some functions that helps to create a rectangular matrix having as row the images on the the query folder and as columns the images from the gallery folder sorted by matrix similarity, this similarity as we said is obtained comparing the two representing matrices that the model have extracted, using the cosine similarity. We also tried different kind if distances but we saw that the cosine similarity overperforms the others so we kept only that.

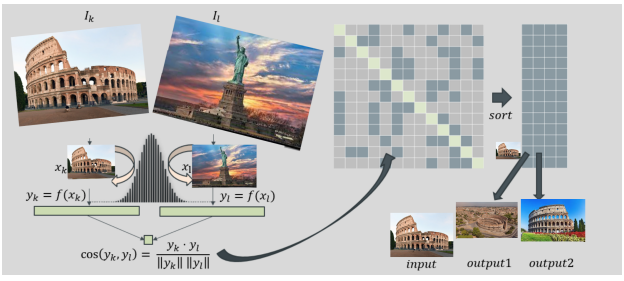

Figure 1: Image similarity search system

5 RESULTS AND DISCUSSION

In this section, we present our results and discuss the performance of all the different models according to top-k evaluation. All the performance data of different models is presented in Table 1. The sample output is shown in Figure 2.

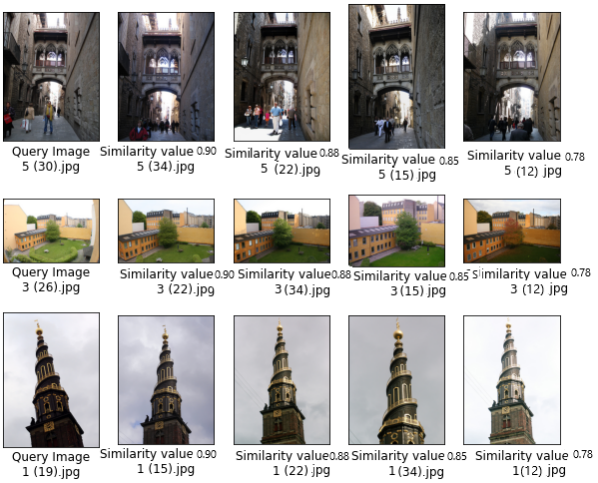

Figure 2: Sample output

The characteristics of these models are as follows:

- Resnet18-small-df is trained on a relatively smaller dataset.
- Resnet-full_layer-1 and Resnet18-full_layer-2 are trained on two different datasets.
- Resnet-full_layer-not_pretrained is not pretrained and it is trained on the entire dataset for at least 30 epochs.
- Resnet-last_layer is a pretrained model with frozen weights except the last layer. The last fully connected layer is trained.
- Resnext is a different version of resnet developed by Xie et al. [7].
- Wide resnet is the same as Resnet but the bottleneck number of channels is twice as large in every block[9].
- GoogLeNet is a convolutional neural network with 22 layers developed by Szegedy et al. [4] .

Among the resnet18 models, the best performing model in top_1 and top_3 was resnet18-full_layer-1 with 0.864 accuracy while for the top_10, the best performer was Resnet18-full_layer-2. Regarding the resnet50 and resnet101 models, pretrained resnet101 has the best top_1 performance while for the top_3 resnet50 and resnet101 have similar performance. Lastly, for the top_10, resnet50 is outperforming resnet101.

Among all models, the best top_1 performance belongs to resnext101 with the last layer trained, the best top_3 and top_10 performances belong to wide_resnet101 with the last layer trained.

By looking at the results, we can state that the pretrained models including the ones in which only the last layer was trained performed better. They all had consistent performances varying by approximately 5%. In the article of He et al. [2], it is shown that as the depth of resnet increases, the error rate decreases. In our case, the error rate increases as we move to depth of 50 from resnet18. Only resnet152 is outperforming resnet18 in top_1 category.

Models	Top_1	Top_3	Top_10
Resnet18_small-df	0.83	0.864	**0.92**
Resnet18-full_layer-1°	0.864	**0.909**	**0.932**
Resnet18-full_layer-2°	0.852	**0.909**	**0.943**
Resnet18-full_layer-not_pretrained	0.511	0.67	0.852
Resnet18-last_layer	0.852	0.886	**0.932**
Resnet50-full_layer	0.784	0.886	**0.932**
Resnet50-last_layer	0.83	0.898	**0.932**
Resnet101-full_layer-not_pretrained	0.409	0.58	0.693
Resnet101-last_layer	0.841	0.898	**0.909**
Resnet152-full_layer	0.865	0.886	**0.909**
Resnet152-last_layer	0.852	**0.932**	**0.932**
Resnet152-full_layer-not_pretrained	0.443	0.648	0.739
Resnext101_32x8d-full_layer	0.795	0.886	**0.909**
Resnext101_32x8d-last_layer	0.898	**0.932**	**0.943**
Wide_resnet101_2-not_pretrained	0.273	0.409	0.682
Wide_resnet101_2-last_layer	0.842	**0.989**	**0.955**
GoogleNet-full_layer	0.818	0.886	**0.932**

Table 1: Results of our models on a query composed by 100 images and 22 classes and 600 gallery images

REFERENCES

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

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385 (2015). arXiv:1512.03385 http://arxiv.org/abs/1512.03385

[3] Connor Shorten and Taghi M. Khoshgoftaar. 2019. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data* 6, 1 (jul 2019). https://doi.org/10.1186/s40537-019-0197-0

[4] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1–9. https://doi.org/10.1109/CVPR.2015.7298594

[5] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. 2015. Particular object retrieval with integral max-pooling of CNN activations. *arXiv preprint arXiv:1511.05879* (2015).

[6] Maria V Valueva, NN Nagornov, Pave A Lyakhov, Georgiy V Valuev, and Nikolay I Chervyakov. 2020. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Mathematics and Computers in Simulation* 177 (2020), 232–243.

[7] Saining Xie, Ross Girshick, Piotr Dollar, Z. Tu, and Kaiming He. 2017. Aggregated Residual Transformations for Deep Neural Networks. 5987–5995. https://doi.org/10.1109/CVPR.2017.634

[8] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. 2018. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging* 9, 4 (jun 2018), 611–629. https://doi.org/10.1007/s13244-018-0639-9

[9] Sergey Zagoruyko and Nikos Komodakis. 2016. Wide Residual Networks. (05 2016).