Mini Project 4 : Reproducibility challenge

Luna Dana and Nicolas Fertout
COMP551 - Applied Machine Learning
McGill University

Abstract

This project investigates the reproducibility potential of the paper Very Deep Convolutional Networks for Large-Scale Image Recognition. Indeed, we reproduced some selected experiments and found that some results matched their claims while others did not. Once this first step was made, the methods of the paper were tested with different parameters and modified data. As mentioned in the paper, the effect of the convolutional network depth clearly affects the accuracy in large-scale image recognition setting.

1 Introduction

In this project, we investigate the reproducibility of the paper Very Deep Convolutional Networks for Large-Scale Image Recognition. In particular, we reproduced their experiments and found that some results matched their claims while others did not. Once this baseline was made, the claims of the paper were tested against different datasets not portrayed in their results. As mentioned in the paper, the comparison that we made was not aimed to yield the best results, but rather aimed to validate the chosen paper findings and explore further way of improving models.

2 Paper's Claim

This paper aims to analyze the effect of the depth of a convolutional network depth on its accuracy. It focuses on the application of this model on large scale image recognition settings. This paper underline the role of the ImageNet Large-Scale Visual Recognition Challenge which have provided the data set and means to try models as such. As a result, the paper claims to come up with significantly more accurate ConvNet architectures. It also claim to not only achieve the accuracy on ILSVRC classification and localisation tasks, but also work on other image recognition datasets.

3 Dataset

3.1 Dataset description

The ILSVRC-2012 dataset includes images of 1000 classes, and is split into three different sets: training (1.3M images), validation (50K images), and testing (100K images with their corresponding class labels). As stated in the paper, training the models on the whole set would take around 2-3 weeks and for this reason we used pre-trained weights, and had to use the same data set. We further decided to select different images which are more complex and only run the validation portion.

3.2 Pre-processing tasks performed

In order to make the dataset usable, the authors applied different data processing techniques. They only subtract the mean RGB value, computed on the training set, from each pixel. The motivation is to normalize the data and not have some picture to dark/light. We used this code to preprocess the image and see how did the normalization change the images of the train set.

33rd Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

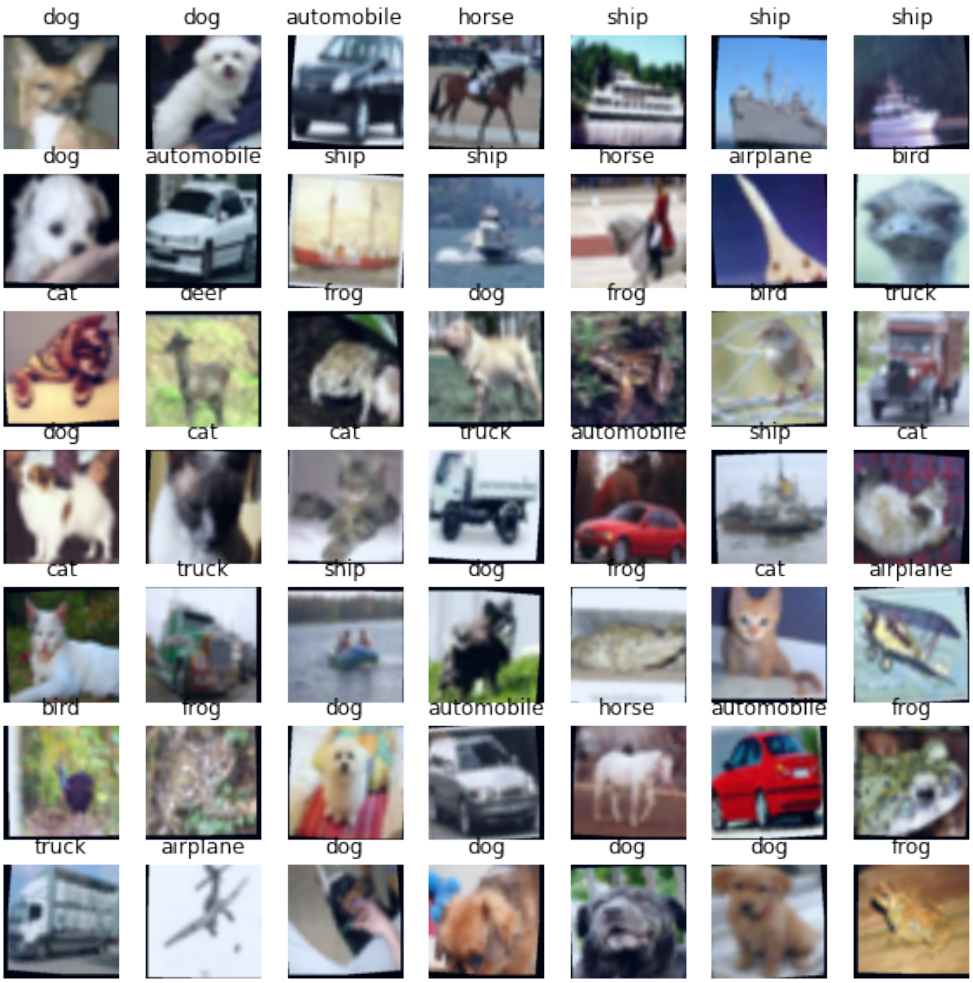


Figure 1: 49 images after normalization script

4 Methods and Design Choices

4.1 Model descriptions

The models used were Convolutional Neural Network using a various number of weight layers. In the table detailing the number of layers, the Softmax layer is not taken into account. Each model is different in the number of layers used. Some layers use ReLu activation (fully nected or convolution) function and other will do a max pooling task.

Model name	Weight layer	Parameters (millions)
A	11	133
A-LRN	11	133
B	13	134
C	16	134
D	16	138
E	19	144

Table 1: Description of the different models trained and tested in the paper

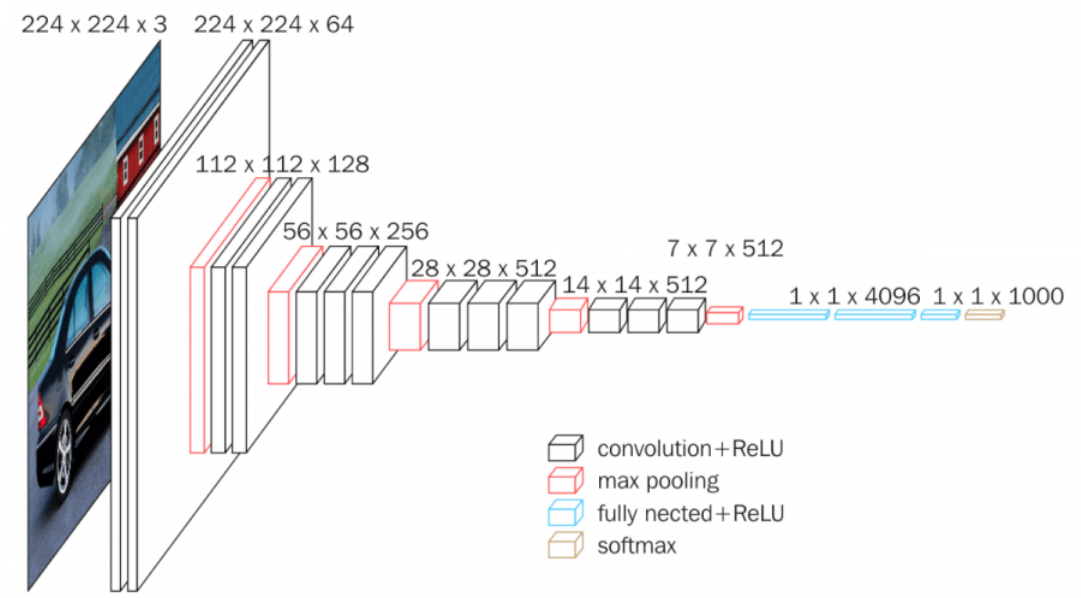


Figure 2: Example of the architecture of model D with 16 layers

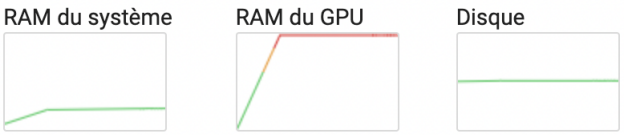
4.2 Experimental setup and code

The code we found provided several implementations, some of them using pretrained weights. The authors of the paper did not provide an official code but we used the following implementations to perform different tasks and grasp the inner workings of the models:

- An implementation which had already pre-trained the model and allowed us to apply further testing strategy.
- A demo of detection on example images using SSD with PyTorch.
- Implementation of models VGG16 and VGG19 (respectively model C and E)
- A notebook with another implementation of the VGG models with further analytics

4.3 Computational requirements

When simply replicating the experiments done in the paper, this code broke after a GPU error when wanting to find the learning rate. We got the following error : CUDA out of memory. Tried to allocate 74.00 MiB. We still got to run the code using the pro-version of google collab with better power.



Since we needed more RAM, we also found that this implementation of the code had pre-trained model and we could use it to perform classification tasks instead of training the models ourselves.

5 Results

Our results first reproduce the initial experiments done in the paper and then verify some claims by going further and testing the model with new images.

5.1 Results reproducing original paper

In order to reproduce the initial paper, we replicated all the experiment. That is, with the helped on pre-trained weight, we re-calculated the 1-crop error rates on imagenet dataset with the different model structures.

Model Structure	Top-1 Error	Top-5 Error
A	30.76%	11.56%
A-LRN	30.12%	10.19%
B	30.45%	10.76%
C	29.34%	9.82%
D	28.53%	9.52%
E	27.56%	9.20%

Table 2: Average accuracy on the MLP classifier using a varying number of 128-unit hidden layers across the ReLu, Tanh, and Leaky ReLu activation functions.

5.2 Results beyond original paper : applying batch norm on model C, D, E.

In this experiment, we introduced here is batch normalization (BN) defined with `BatchNorm2d` and only used if `batch_norm = True` in the function `get_vgg_layers(config, batch_norm)`

Model Structure	Top-1 Error	Top-5 Error
B (with bn)	29.37%	9.82%
C (with bn)	27.54 %	8.55%
E (with bn)	26.67%	8.21%

Table 3: Average accuracy on the MLP classifier using a varying number of 128-unit hidden layers across the ReLu, Tanh, and Leaky ReLu activation functions.

5.3 Results on larger dataset non standard

Often papers don't include enough information to fully specify their experiments. Indeed, most tested images are very clear (subject from category on a background). For this reason, we have tried to compare the accuracies of the models with new images (which are presumably harder for the algorithm). We also modified some images (colours, shape, brightness) to see if the model still works as claimed in the paper.

Figure 3: Example of a non standard image of a black grouse (left) and the same image with augmentation and modification technique (right)

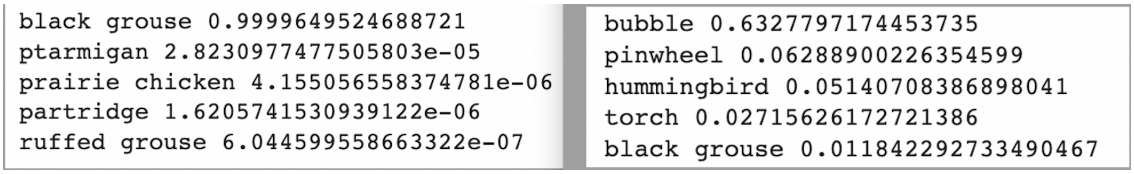


Figure 4: Result for normal image (left) and the results for same image with augmentation and modification technique (right) (Model A with Batch Norm)

6 Discussion

6.1 Key findings

Our first key finding is that when replicating the models and running them as stated in the paper, we get approximately the same results therefore validating their approach and conclusions on this dataset.

However, adding the Batch Normalization improves the actual model from the paper even for larger number of layers (13,16 and 19). Hence, we can say that we found additional results from the one stated in the paper showing the positive effects of the batch norm. Batch Norm is basically adding another layer between a hidden layer and its next hidden layer. It aims to normalize the outputs from the first hidden layer before passing them on as the input of the hidden layer coming next.

We also run experiments which contradict some claims in the paper. The paper state that the models work on non standard image, and have good generalization. Nonetheless, we observed that modified images such as the Black Grouse in figure 3 (right) managed to "trick" the model into classifying it incorrectly. While the original image led to a 99.996% predicted probability of being a black grouse, the modified one gave a 63.28% of being a bubble, with black grouse still being in the top 5 but with only 1.18%. We used similar data augmentation techniques on different images to observe differences in performance, and the majority of our tests resulted in analogous outcomes. This could thus mean that the original paper did not take this type of data modifications into account, or that the way they handled it is not described nor depicted in the codes we used.

6.2 What was easy

Since a lot of code was available online to replicate the experiments, we did not have to do a lot of coding. It was mainly modifying codes and adding/removing parameters. It was therefore easy to run the same experiments as described in the paper. Moreover, the paper only used one data set which is very accessible online which made the task easier.

6.3 What was difficult

The theoretical knowledge needed to replicate the paper is very heavy. Therefore in order to do further experiment, like adding the batch norm and understand the inner mechanisms of this model tuning, we had to conduct a lot of research on CNNs and read papers on how Batch norm is helping optimization.

Besides, the original model being computationally heavy, the use of pretrained model was the best choice, but did not allow us to freely explore all the options we wanted to.

6.4 Directions for Future Investigation

Future experiments could be the following :

- Try to run the models with different activation functions
- Train and test the models on different datasets, and compare results to see if one could highlight some over-fitting.
- Use different values for parameters, such as convolution stride (in the paper it was fixed to 1 pixel).

7 Contributions

Together : Choice of paper, Decision of the experiment to run, writting of the main paper.

Luna : Data set section, transformation of images, methodology section

Nicolas : Running of the different models, discussion of the results