Deeper Networks for Image Classification

Alexander Sworski
ECS795P – Deep Learning and Computer Vision
School of Electronic Engineering and Computer Science - Department of Computer Science
M.Sc. Artificial Intelligence
Queen Mary University
London, United Kingdom
a.sworski@se21.qmul.ac.uk
210456914

***Abstract*—This paper represents a comparison between different Convolutional Neural Networks. In particular, the models GoogLeNet, VGG-16 and ResNet are compared on different datasets, such as MNIST and Cifar10.**

***Index Terms*—GoogLeNet, ResNet, VGG-16, MNIST, Image Classification, Deep Learning**

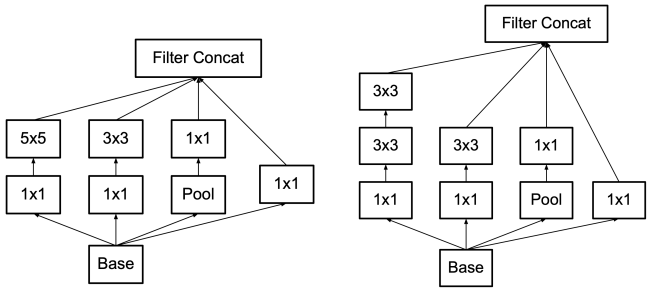
I. Introduction

This document is a model and instructions for LaTeX.

II. Literature Review

A. Neural Network models

1) GoogLeNet: GoogLeNet was initially developed as a submission for the ImageNet Large-Scale Visual Recognition Challenge 2014, which it won. The main hallmark of this Model is the improved utilization of the computing resources inside the network. It has 12 times fewer parameters than the winner of 2012 (AlexNet) but performs significantly better. [7] This Model first introduced the idea of an inception module, which can be seen in Figure 1a. A second version of the Model was published a year later, which then introduced an improved inception module, which can be seen in Figure 1b. This new version reduced computational costs, as bigger convolutions are disproportionately more expensive. Using two 3x3 convolutions instead of 5x5 is computationally less expensive while improving the performance. Although there are is also a V3 and V4 of GoogLeNet, for this paper, the V2 Inception has been used. In total the model has 22 layers.



(a) Original Inception module [7] (b) V2 Inception module [8]

Fig. 1: Three simple graphs

2) VGG-16: VGG-16, is the biggest of the models used in this paper (Figure 2). It also has been developed as a submission for the ImageNet Challenge, in which it won first and second place for the localization and classification tracks, respectively. It consists out of 16 layers and uses only 3x3 convolutions. Although it has fewer layers than GoogLeNet, each layer is computationally more complex. Ultimately, this Model resulted in better scores then the GoogLeNet V1 [6].

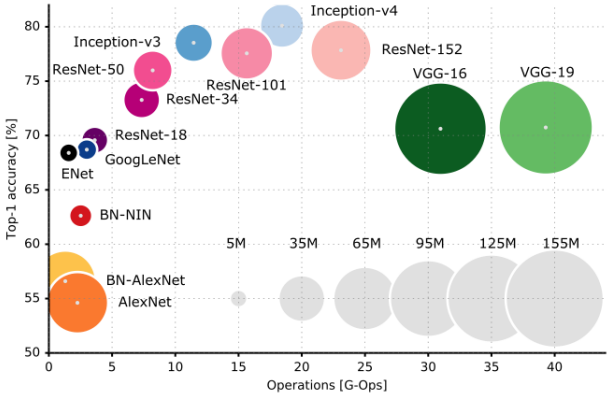


Fig. 2: top-1 one-crop accuracy versus amount of operations required (circle size represents the amount of parameters)

3) ResNet: ResNet, has the highest amount of layers. While there are multiple versions, for this paper a 50 layer version has been chosen. In Figure 2, it can be observed that the model

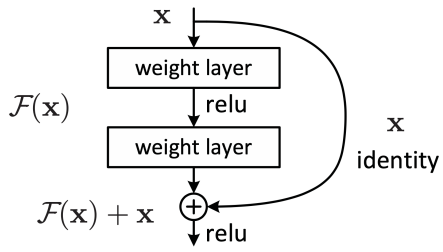


Fig. 3: Residual learning: a building block [5]

size is in between GoogLeNet and VGG-16, yet the best result can be expected according to this comparison. ResNet has been

built with the goal of easing the training of deep networks. The network design is based on chaining multiple residual learning blocks on a row. One residual learning block can be seen in Figure 3. [5]

B. Image datasets

1) MNIST: The MNIST database contains 70,000 28x28 black and white images. 60,000 images are for training and 10,000 images for testing. The images' portrait handwritten numbers from 0 to 9. [9] Examples of the classes can be seen in Figure 4.

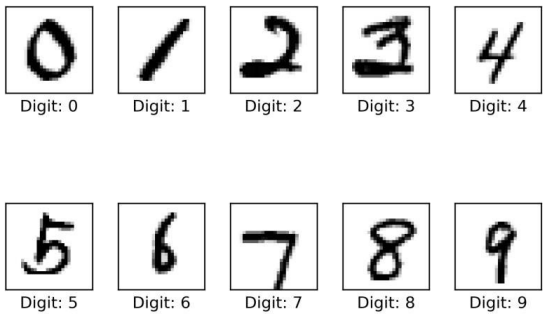


Fig. 4: Cifar10 image samples [2]

2) Cifar10: The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The classes are mutually exclusive.[1] Examples of the classes can be seen in Figure 5.

Fig. 5: Cifar10 image samples [3]

III. Methodology

In the first subsection III-A the implementation of the models using the framework Keras will be explained. In the subsection III-B it will be explained how the datasets have been made compatible with the models.

A. Model implementation

1) GoogLeNet: GoogLeNet considering only the parameters, is the most lightweight Model of the three. Nevertheless, it has 108 operational layers ordered in 22 layers, which can be logically segmented into five stages, excluding input and output. The first two stages of the Model consist of multiple 2D convolutions, followed by a `BatchNormalization` layer and a `MaxPooling2D` layer. Using the `BatchNormalization` layer is a unique attribute of the second version of GoogLeNet. Stage 3 consists of two V2 inceptions, as seen in Figure 1b, followed by a `MaxPooling2D` layer. Stage 4 has five inceptions and two auxiliary modules, followed by a `MaxPooling2D` layer. In Stage 5, we have two inception modules followed by a `GobalAveragePooling2D` layer, equivalent to the 7x7 convolution found in other versions of this network. The Model then finishes with a Dropout of 0.4 and a fully connected layer using `softmax` as its activation function. [8]

2) ResNet: ...

3) VGG-16: VGG-16 is the biggest Model of the three, regarding the parameters. Judging based on layers, this Model is the smallest, as it only consists out of 16 layers, which are segmented into 5 blocks. Each block is structured in the same manner. Two or three 2D Convolution, followed by one 2D MaxPolling layer. The 5 blocks are then followed by 3 dense layers, one dropout of 0.5 and a final fully connected layer using `softmax` as its activation function. The full network architecture can be seen in Figure 6. [6]

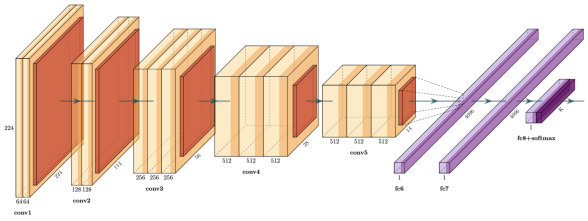


Fig. 6: VGG-16 architecture [4]

B. The Datasets

The dataset used in this paper both have a resolution of 28x28 pixels. On top, the Cifar10 dataset has three colour channels, while the MNIST dataset only has one grey channel. This represents an issue, as the Neural Networks has initially been designed to work with the Microsoft ImageNet dataset. This dataset has 3 colour channels and a resolution of 224x224 pixels. The goal was not to change the input of the models, as this would have significant implications for the layers following, resulting in a significant change in the model design. Therefore, the datasets have been altered as follows to fit the input dimensions of the models.

1) MNIST dataset alterations: The MNIST dataset has the shape (28,28,1), while our models required an input shape of (224,224,3). In order to change the dimensionality, the OpenCV library has been used. Using OpenCV, the image has been interpolated to fit the 224x224 image size. Afterwards, the image was stacked three times to obtain our three-channel input. Although, there is the OpenCV function `cv2.cvtColor(src, cv2.COLOR_GRAY2RGB)`, which can convert from greyscale to RGB. The results are not different from our stacked layers. This is due to the unique properties the MNIST dataset has. Using the stacked layers is computational lightweight.

2) Cifar10 dataset alterations: The Cifar10 dataset is already in RGB. Therefore, we do not need to convert it into a different colour space. Therefore, this dataset has been interpolated from the shape (28,28,3) to (224,224,3) using the OpenCV library.

C. Project structur

One goal of this project, is to create a GitHub repository, that is fully documented and allows for easy replication of the results presented in this paper. In Figure 7, the file structure of this project can be seen. All project files, including the Checkpoints are public on GitHub.

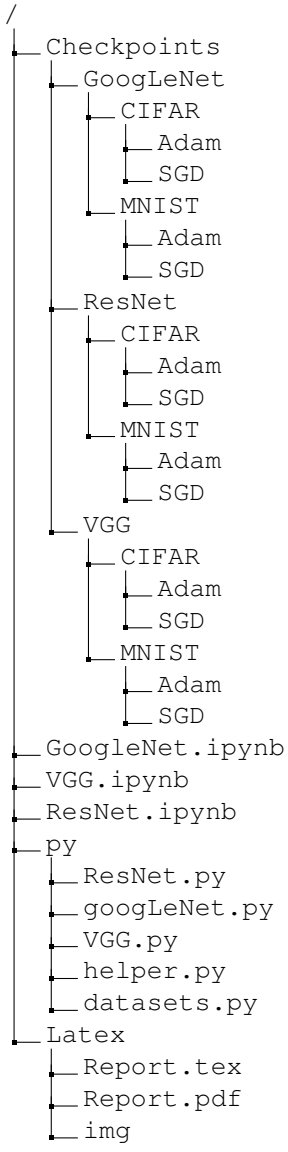


Fig. 7: Project file structure

The project has 3 entry points, which are each a Jupiter Notebook file. Each Notebook is dedicated to one Neural Network. At the top of the Notebooks, the Hyperparameters of the network can be adjusted as well as runtime environment variables. The notebooks have two modes, one for a Google Colab execution and one for a local execution. In case of a Google Colab execution, the notebook will use Google Drive for Checkpoints rather than the Checkpoint folder within the project. In order to minimize the code within the notebooks and create a clean and easing readable code, code has been outsourced into dedicated python files. The `helper.py` file, takes care of Hyperparameters as well as evaluating the model. The `dataset.py` handles the dataset download and the alterations described in Section III-B. The three other python files define the neural networks, corresponding to their names. Every function within the python files is documented, and the notebooks are divided into descriptive sections.

IV. Results

Each Model has been trained over 20 epochs twice for each dataset, once using the Adam optimizer and once using the SGD optimizer. The overall accuracy and the confusion Matrix have been recorded. In the following chapter, the results will be presented.

A. GoogLeNet

Using the SGD optimizer, the Model performed very well on the MNIST dataset but was suboptimal on the CIFAR-10 dataset. If we look at the confusion matrix of the Model using the SGD optimizer on the CIFAR-10 dataset, we can see that the low accuracy comes from a localized inadequate recognition of the labels 2, 3 & 4.

We can see a significant reduction in accuracy and a double in training speed with the Adam optimizer.

TABLE I: GoogLeNet model accuracy

Model accuracy	***SGD***	***Adam***
MNIST	11.02%	98.96%
CIFAR-10	10.17%	66.34%

TABLE II: GoogLeNet training time

time/step	***SGD***	***Adam***
MNIST	133s 354ms	258s 688ms
CIFAR-10	110s 351ms	109s 349ms

B. VGG-16

TABLE III: VGG-16 model accuracy

Model accuracy	***SGD***	***Adam***
MNIST	%	%
CIFAR-10	%	%

TABLE IV: VGG-16 training time

time/step	***SGD***	***Adam***
MNIST		
CIFAR-10		

TABLE V: ResNet-50 model accuracy

Model accuracy	***SGD***	***Adam***
MNIST	%	%
CIFAR-10	%	%

TABLE VI: ResNet-50 training time

time/step	***SGD***	***Adam***
MNIST		
CIFAR-10		

C. ResNet-50

V. Discussion

VI. Conclusion

References

[1] *CIFAR-10 and CIFAR-100 datasets*. URL: https://www.cs.toronto.edu/~kriz/cifar.html.

[2] *Convolutional Neural Networks (CNN) for CIFAR-10 Dataset*. Parneet Kaur. URL: http://parneetk.github.io/blog/cnn-cifar10/.

[3] *Fig. 2. Some samples of the MNIST dataset*. ResearchGate. URL: https://www.researchgate.net/figure/Some-samples-of-the-MNIST-dataset_fig1_342733731.

[4] *Forks · HarisIqbal88/PlotNeuralNet*. GitHub. URL: https://github.com/HarisIqbal88/PlotNeuralNet.

[5] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *arXiv:1512.03385 [cs]* (). version: 1. arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[6] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv:1409.1556 [cs]* (). version: 6. arXiv: 1409.1556. URL: http://arxiv.org/abs/1409.1556.

[7] Christian Szegedy et al. "Going Deeper with Convolutions". In: *arXiv:1409.4842 [cs]* (). version: 1. arXiv: 1409.4842. URL: http://arxiv.org/abs/1409.4842.

[8] Christian Szegedy et al. "Rethinking the Inception Architecture for Computer Vision". In: *arXiv:1512.00567 [cs]* (). version: 3. arXiv: 1512.00567. URL: http://arxiv.org/abs/1512.00567.

[9] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. *MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges*. URL: http://yann.lecun.com/exdb/mnist/.