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 paper represents a comparison between different Convolutional Neural Networks. In particular, the models GoogLeNet, VGG-16 and ResNet. These three models are compared on the datasets MNIST and Cifar10. This document is split in 6 chapters. The I. Chapter represents a short introduction to the topic and serves as and overview of the document. The II. Chapter presented a Literature Review containing a short overview and history of the models and the datasets. Chapter III describes the implementation of the models, the conducted experiment and the dataset usage. In Chapter IV the Finding of the experiments are presented, which are then discussed in Chapter V. Followed by a conclusion in Chapter VI

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. [9] 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.
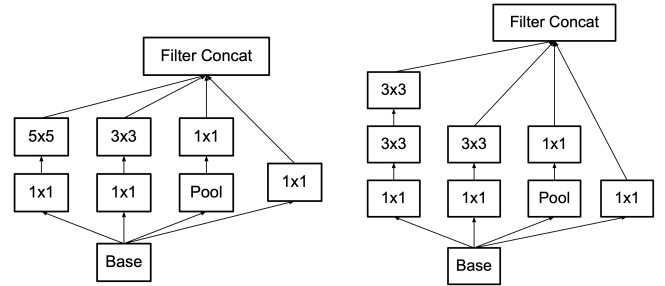
(a) Original Inception module [9]   (b) V2 Inception module [10]

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 [8].
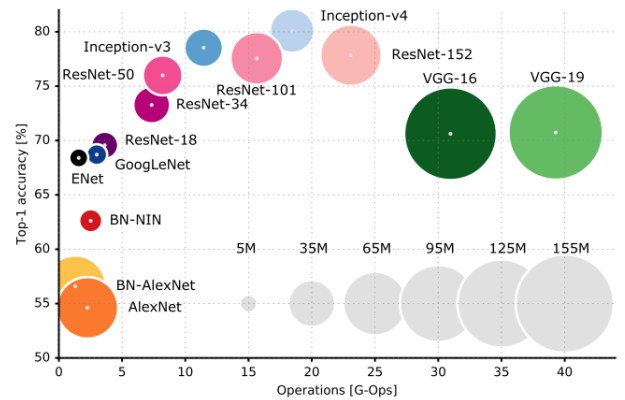

Fig. 2: top-1 one-crop accuracy versus amount of operations required (circle size represents the amount of parameters)

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
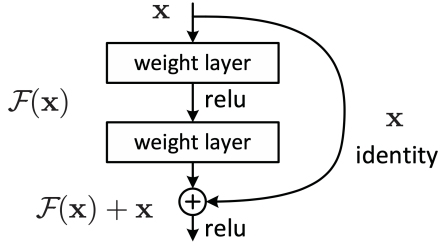
Fig. 3: Residual learning: a building block [7]

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. [7]

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. [11] Examples of the classes can be seen in Figure 4.
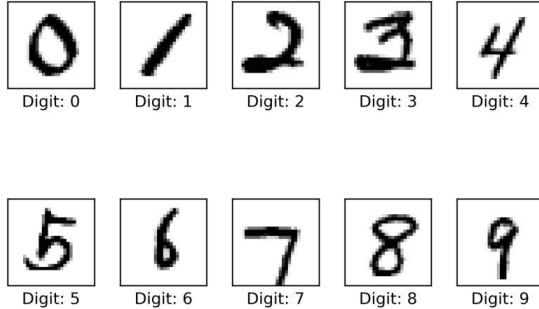

Fig. 4: Cifar10 image samples [2]

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. [10]

2) VGG-16: VGG-16 is the biggest model of the three regarding the parameters. Based on layers, this model is the smallest, as it only consists of 16 layers, which are segmented into 5 blocks. Each block is structured in the same manner: two or three 2D convolutions, followed by one 2D MaxPolling layer. The five blocks are then followed by three 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. [8]

Unfortunately, this model was too complex for the Google Colab environment, as the error `"ResourceExhaustedError: OOM when allocating tensor"` has been given. Therefore, the model was modified using suggestions made in the Stack Overflow forum. [5] The alterations were as follows: each block with three convolutions has been reduced to two, each block with two has been reduced to one convolutional layer, and the final three dense layers have been reduced to only one dense layer. Furthermore, the batch size has been reduced to 32. The rest of the model was not modified. This led to horrible results, that can be found in Table II and Figure 9. [1]

Further, testing also confirmed, that there was no implementation error in the original Keras implementation. Therefore, the only remaining alternative was to implement the model in a different framework, in this case PyTorch, which is more lightweight. The implementation of the model, with the initial architecture using PyTorch instead of Keras, could then be

---

[1]During later stages of the project, while exchanging with classmates, it was brought to my attention, that Google does limit the usage of Colab Pro after a while, which is presumably the cause for this issue.

successfully executed in Colab. The only minor change that had been made, was to reduce the image size of the input for this model to 64x64 pixels, as the environment it would otherwise run out of RAM.
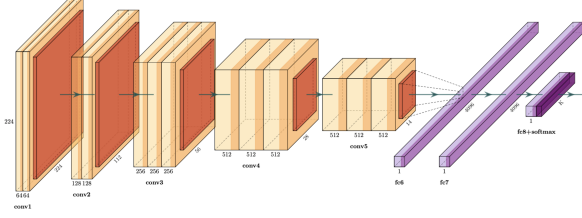

Fig. 6: VGG-16 architecture [4]

3) ResNet: For this paper, the ResNet-50 Model has been chosen. The number 50 means that the model consists of 50 layers of residual blocks (Figure 3) stacked on top of each other. As mentioned before, there are different versions of ResNet, which all hold a different amount of layers. Based on the original paper results, ResNet-50 has the second-best results, behind ResNet-110. [7]. For this paper, the ResNet version designed for the ImageNet dataset has been used. There is a version in the original paper that has been adapted to use (32,32,3). In an effort to keep results between the networks comparable, the version accepting smaller images has not been used. The main final network architecture consists of 4 blocks, each starting with a convolutional block, followed by a varying number of identity blocks (2,3,5,2).

Unfortunately, this ResNet implementation experienced the same issues as the original VGG-16 implementation in Keras. Consequently, the model was reimplemented using PyTorch.

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 structure

This project aims 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, are public on GitHub. The Checkpoints can be downloaded from Google Drive.

```
/
├── Checkpoints
│   ├── GoogLeNet
│   ├── ResNet
│   └── VGG
├── GoogleNet.ipynb
├── VGG_torch.ipynb
├── VGG.ipynb
├── ResNet.ipynb
├── ResNet_torch.ipynb
├── py
│   ├── ResNet.py
│   ├── ResNet_torch.py
│   ├── googLeNet.py
│   ├── VGG.py
│   ├── VGG_torch.py
│   ├── helper.py
│   ├── datasets.py
│   ├── helper_torch.py
│   └── datasets_torch.py
└── Latex
    ├── Report.tex
    ├── Report.pdf
    └── img
```
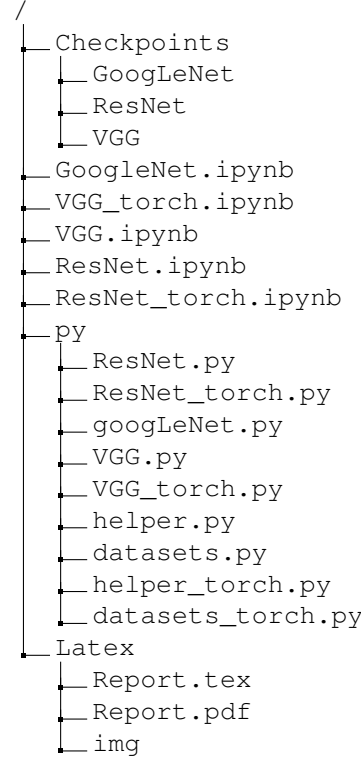
Fig. 7: Project file structure

The project has five 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 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 evaluates 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 bird, cat & deer, which even as a human are very difficult to distinguish from each other.

We can see a significant reduction in accuracy and a double in training speed with the Adam optimizer. In Figure 8a & 8c we can clearly see, that using the Adam optimizer in both cases led to no training at all.
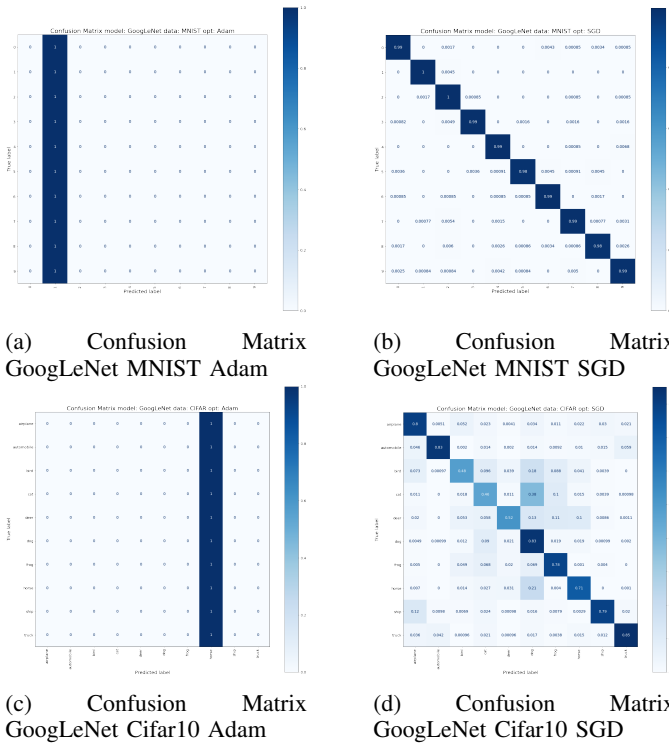

(a) Confusion Matrix GoogLeNet MNIST Adam

(b) Confusion Matrix GoogLeNet MNIST SGD

(c) Confusion Matrix GoogLeNet Cifar10 Adam

(d) Confusion Matrix GoogLeNet Cifar10 SGD

Fig. 8: GoogLeNet Confusion Matrices

TABLE I: GoogLeNet model Results

	Model accuracy		time/step	
	Adam	*SGD*	*Adam*	*SGD*
MNIST	11.02%	98.96%	133s 354ms	258s 688ms
CIFAR-10	10.17%	66.34%	110s 351ms	109s 349ms

B. VGG-16

As mentioned previously, the initial model implementated in Keras was too big to be executed using Google Colab Pro and only a heavily simplified version of the model, which expectedly performs very bed, could be executed. As this poor performance of the simplified version becomes evident very quickly, the model has only been trained on the CIFAR dataset using the SGD optimizer. The results can be seen in Table II and Figure 9.
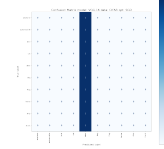

Fig. 9: Confusion Matrix simplified VGG-16 Cifar10 SGD

TABLE II: Simplified VGG-16 model Results

	Model accuracy	time/step
	SGD	*SGD*
CIFAR-10	9.33%	152s 121ms

To fix this issue, the model has been reimplemented in PyTorch. As mentioned previously, this model only uses images with the siez 64x64 pixels as input. Despite that, the model performed well on both the MNIST and the CIFAR dataset using the SGD optimizer.

(a) Confusion Matrix VGG-16 PyTorch MNIST Adam

(b) Confusion Matrix VGG-16 PyTorch MNIST SGD

(c) Confusion Matrix VGG-16 PyTorch Cifar10 Adam
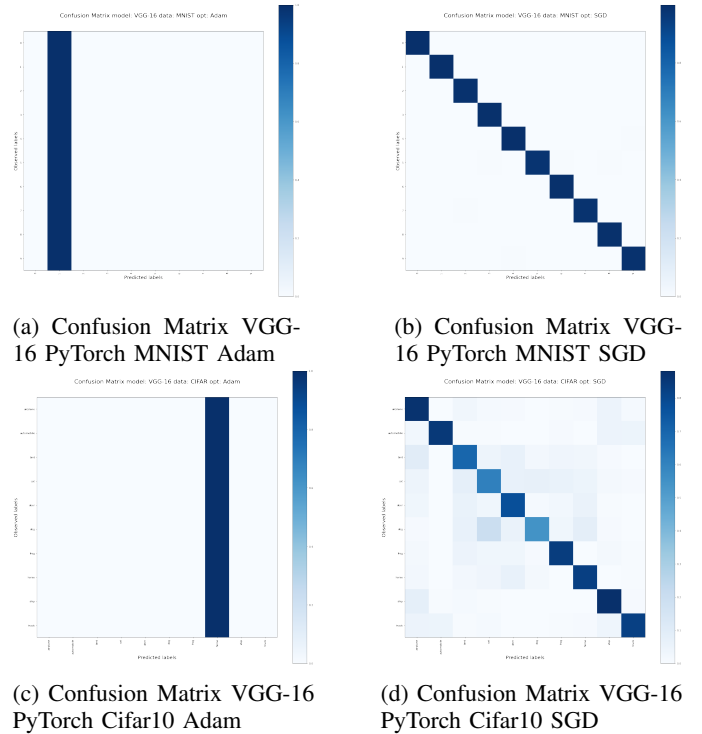

(d) Confusion Matrix VGG-16 PyTorch Cifar10 SGD

Fig. 10: VGG-16 PyTorch Confusion Matrices

The Confusion Matrices for the MNIST and the CIFAR dataset can be found in Figure 10. The results can be found in table III. The results obtained by this mode, are very similar to the GoogLeNet model, which just slightly better accuracies.

TABLE III: VGG-16 PyTorch model Results

Model accuracy			time/step	
	Adam	*SGD*	*Adam*	*SGD*
MNIST	11.02%	99.12%	70s 480ms	70s 200ms
CIFAR-10	9.94%	77.10%	61s 800ms	61s 600ms

C. ResNet-50

While the Keras version of the model was not executable on Google Colab, the PyTorch version did train. In Table IV the accuracy and the training time can be seen, while in Figure 11 the Confusion Matrices for the model can be found. The accuracy of the model differs from the accuracy of the others, as this model performs equally well using both the SGD and the Adam optimiser.

TABLE IV: ResNet-50 PyTorch model Results

Model accuracy			time/step	
	Adam	*SGD*	*Adam*	*SGD*
MNIST	98.73%	99.18%	51s	49s 200ms
CIFAR-10	65.99%	71.72%	42s 600ms	66s 656ms

(a) Confusion Matrix ResNet-50 MNIST Adam

(b) Confusion Matrix ResNet-50 MNIST SGD

(c) Confusion Matrix ResNet-50 Cifar10 Adam

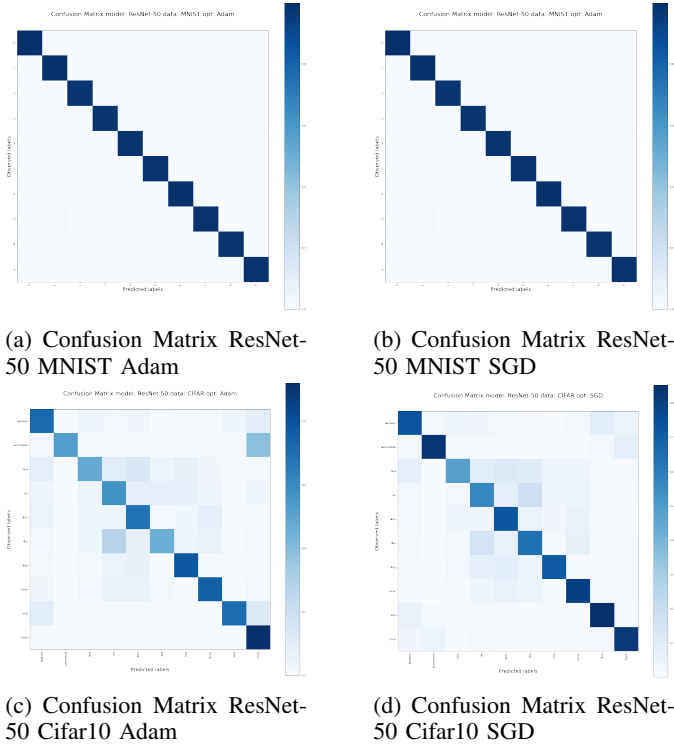(d) Confusion Matrix ResNet-50 Cifar10 SGD

Fig. 11: ResNet-50 PyTorch Confusion Matrices

V. DISCUSSION

Two major observations were made during the experiments. One being the bad performance of the Adam optimizer, the other being the bad performance of all clasifiers on the bird, cat & deer or dog classes if the CIFAR dataset.

While literature suggests that the Adam optimizer has worse generalization performance than SGD [6], the results found in this experiment come at a surprise. As the performance of the models with the SGD optimizer are very well, it could be argued, that this scenario is a special case, in which the Adam optimizer performs particularly bad. The one exception to this is the ResNet-50 model, which performs similarly well using both the Adam and the SGD optimizer.

The second observation was the issues in distinguishing the bird, cat & deer or dog classes if the CIFAR dataset. While the model all perform excellent on the other classes these classes seem to be more difficult to clasify. While this could be an indication of a Minority class, this can be out rules as the dataset are pre-revisioned and each class is equality represented. Neither an overfitting of a particular class can be observed, but rather a well distributed confusion between these classes. These leads to the conclusion, that these classes are too difficult to distinguishing for the models. The argument is, that this images often cointain to little information to distinguish them. To test this arguemnt a small test group of 10 people has been asked individually after been show one picture each of the bird, cat & deer category, given the 10 labels to classify them. While these results are not representative, 30% of the people were not able to correctly classify the category and nobody could classify the bird correctly. Resulting in a performance of 56%. Based on these non representative results, it can be argued, that the results in the CIFAR dataset are very well.

Looking back at the initially assumption based of Figure 2, it can be said, that this did almost correspond to the finding. The ResNet model performed overall the best and the GoogLeNet model performed the worst. With the expection that the VGG-16 model performed 0.06% better on the MNIST dataset compared to the ResNet model. This is within the margin of error.

VI. CONCLUSION

While the results obtained in the experiment differed slightly from the initial assumption, the overall experiment can be considered successfully. All three models have been successfully implemented and trained, using both datasets.

In retrospective the one change that would be made, should this or a similar project be attempted in the future is to use the PyTorch library instead of the Keras framework. While the implementation in both Keras and PyTorch was successfully, the Keras implementation did cause issues when executing on the Google Colab GPU, which did cost a lot of time.

REFERENCES

[1] *CIFAR-10 and CIFAR-100 datasets*. URL: https://www.cs.toronto.edu/~kriz/cifar.html (visited on 04/30/2022).

[2]   *Convolutional Neural Networks (CNN) for CIFAR-10 Dataset*. Parneet Kaur. URL: http://parneetk.github.io/blog/cnn-cifar10/ (visited on 04/30/2022).

[3]   *Fig. 2. Some samples of the MNIST dataset*. ResearchGate. URL: https://www.researchgate.net/figure/Some-samples-of-the-MNIST-dataset_fig1_342733731 (visited on 04/30/2022).

[4]   *Forks · HarisIqbal88/PlotNeuralNet*. GitHub. URL: https://github.com/HarisIqbal88/PlotNeuralNet (visited on 05/01/2022).

[5]   Nicolas Gervais. *Answer to "How to fix "ResourceExhaustedError: OOM when allocating tensor""*. Stack Overflow. Dec. 18, 2019. URL: https://stackoverflow.com/a/59395251 (visited on 05/02/2022).

[6]   Aman Gupta et al. "Adam vs. SGD: Closing the generalization gap on image classification". In: (), p. 7.

[7]   Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *arXiv:1512.03385 [cs]* (Dec. 10, 2015). version: 1. arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385 (visited on 04/30/2022).

[8]   Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv:1409.1556 [cs]* (Apr. 10, 2015). version: 6. arXiv: 1409.1556. URL: http://arxiv.org/abs/1409.1556 (visited on 04/30/2022).

[9]   Christian Szegedy et al. "Going Deeper with Convolutions". In: *arXiv:1409.4842 [cs]* (Sept. 16, 2014). version: 1. arXiv: 1409.4842. URL: http://arxiv.org/abs/1409.4842 (visited on 04/30/2022).

[10] Christian Szegedy et al. "Rethinking the Inception Architecture for Computer Vision". In: *arXiv:1512.00567 [cs]* (Dec. 11, 2015). version: 3. arXiv: 1512.00567. URL: http://arxiv.org/abs/1512.00567 (visited on 04/30/2022).

[11] 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/ (visited on 04/30/2022).