Using MNIST Dataset to Evaluate Convolutional Neural Network Architectures

Erfan Divanizadeh – 160618297
Queen Mary, University of London
e.divanizadeh@se16.qmul.ac.uk

***Abstract* — This research paper evaluates and compares the performance and accuracy of various convolutional neural network architectures and their respective effects on the training and testing set accuracies using the MNIST (Modified National Institute of Standards and Technology database) database. The three architectures used for evaluation in this paper are VGG-16, ResNet-34, and GoogLeNet. Each architecture was evaluated at different learning rates and epochs to provide an adequate comparison.**

I. Introduction

Convolutional neural networks are vital to the development and success of computer vision algorithms and applications, the first of which was the recognition of hand-written characters and digits – a model was built to train the neural network to recognize hand-written characters and digits by cumulating various distinct features to then yield more complex features which enabled the neural network to achieve its task of character recognition. Since the introduction of convolutional neural networks in the 1990s and with the help of more powerful and efficient graphics processing units, there has been a number of more complex architectures and models that have been developed and yield more accurate recognition capabilities. Most, if not all, of these architectures existing today were inspired by the AlexNet architecture in 2012 [1]. The said network models include VGG-16, GoogLeNet, and ResNet-34; this paper will compare these three architectures by evaluating their respective performances on the MNIST dataset at different learning rates and epochs.

II. MNIST Dataset

The MNIST database contains an ample number of hand-written digits; it is used for training and evaluating various deep learning architectures. This dataset was created by mixing image samples from the original NIST image dataset due to the original dataset not being well suited for Machine Learning experiments. Moreover, the MNIST dataset consists of 60,000 training images and 10,000 testing images which is ideal for this paper's evaluation of the three deep learning architectures. This paper predicts that the aforementioned architectures for evaluation will achieve very high accuracy as the majority of the architectures are said to achieve near-perfect accuracy when trained using the MNIST dataset. Furthermore, this dataset has proved its worth, credibility, and reliability in various machine learning and deep learning applications thus proving to be the ideal dataset for training and testing purposes of this paper.

III. CNN Architectures

A. *VGG-16*

The VGG architecture is a convolutional neural network architecture proposed by the Visual Geometry Group of the University of Oxford in 2014. It consists of 13 convolutional layers with three fully connected layers (see figure 1). When comparing to other architectures, VGG has relatively smaller filter sizes at 3x3. The three fully connected layers in the VGG16 architecture consists of two fully connected layers with 4096 channels each and one layer having 1000 channels to detect 1000 different class labels in the default configuration. In this study, a modification is done in the last fully connected layer: changing the total number of output classes from 1000 to 10 as MNIST has 10 different image classes which require classification. Moreover, VGG is trained with images with dimensions 224x224 pixels which could be problematic as the MNIST images are 28x28. Therefore, the MNIST images were resized to 80x80 – if the image dimensions were resized or increased any further, unwanted artefacts would appear within the image.

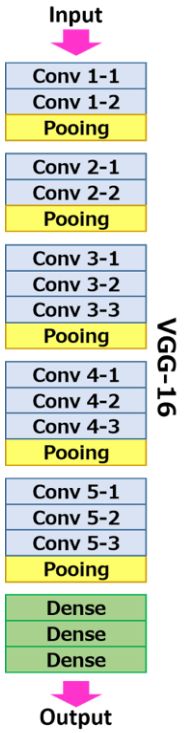

Figure 1: Visual representation of VGG-16 Architecture

The convolution in VGG on the input image is performed with specific parameters (*padding=1, stride=1, kernel size=3x3*) and then maxpooled thus reducing the image dimensions by half; this step iterates throughout the rest of the neural network while changing input and output channel values; ReLU is used as the activation function while using batch normalization to facilitate the initialization of weights to perform better than relatively higher learning rates. The output image of the last convolutional layer is then used as an input to the fully connected layers which outputs 10 channels as different class labels.

B. GoogLeNet

GoogLeNet is a convolutional neural network proposed by Google researchers in 2014 [2]. This architecture demonstrates a lower error rate when compared to architectures such as AlexNet and VGG [2]. The downside of deeper convolutions was that an increase in layers meant an increase in the risk of overfitting and more vanishing gradient scenarios.

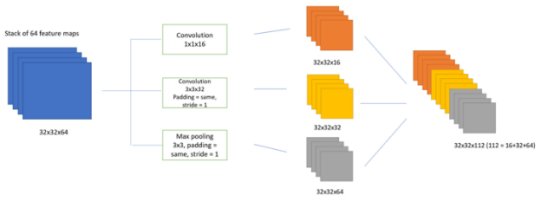

Figure 2: Visual representation of the Inception Module

The GoogLeNet architecture eliminated these problems with the introduction of the Inception module (see figure 1), which makes use of dimensionality reduction techniques to reduce the overall computational cost of the network [3]. Also, it performs dimensionality reduction by adding 1x1 convolution layers before the 3x3 and 5x5 convolution layers thus allowing for faster computation.

Figure 3: Visual representation of GoogLeNet Architecture

The GoogLeNet architecture consists of 22 layers with 5 pooling layers (see figure 2). The input image is passed through the first convolution layer where patch size = 7x7 – this patch size helps reduce the input image dimensions whilst maintaining spatial information. Then, max-pooling is performed. Finally, the resulting image is passed to the second convolutional layer which makes use of 1x1 convolution layers for dimensionality reduction, therefore, facilitating computation.

C. ResNet-34

The ResNet model was proposed to address issues faced during training deeper neural networks such as VGG [4]; deeper neural networks faced the vanishing gradient issue when layers were stacked on top of each one another to create a deep neural network [5]. The vanishing gradient problem occurs when the gradient is repeatedly multiplied, resulting in a very small gradient at the backpropagation stage, negatively affecting the performance of the network. The ResNet architecture (see figure 3) counters this problem by the introduction of the residual block.

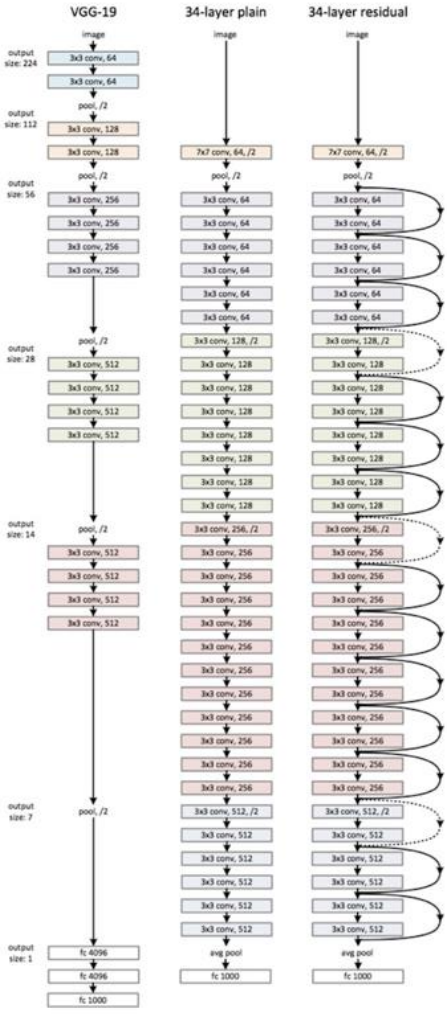

Figure 4: Visual representation of ResNet-34 Archirecture

IV. Critical Analysis and Literture Review

A. The Architectures

Deep learning neural networks yield better results in computer vision and machine learning relative to handcrafted feature descriptors. As aforementioned, the architectures existing today were inspired by the AlexNet architecture in 2012 [1]. This architecture consists of a total of eight layers with weights (five convolutional layers and three fully connected layers). AlexNet maximises the multinomial logistic regression objective which is essentially maximising the average across training cases of the log-likelihood of the correct label under the prediction distribution [1]. Moreover, this neural network architecture has around 60 million parameters; while there are a thousand classes causing each training instance to impose ten bits of constraint on the mapping from image to label, it would not suffice to learn ample parameters without a substantial risk of overfitting. With the implementation of data augmentation and dropout to overcome the said risk of overfitting, Krizhevsky et al managed to score a top-5 classification accuracy of 83%.

In 2014, Simoyan et al [8] extended the AlexNet architecture to have 16 layers including three fully connected layers – this new model is called VGG-16. Unlike AlexNet, VGG-16 contains 138 million parameters (123 million of them belonging to the fully connected layers). This means that the model is required to fit complex non-linear discriminant functions in the feature space into which the input data is mapped. Nevertheless, the risk of overfitting the classifier still exists; this was overcome with the replacement of the 5x5 and 7x7 kernel sizes in the convolutional layers to 3x3 [8]. As a result, the smaller kernel sizes provided the researchers with more freedom to build deeper networks therefore enhancing the network's performance. Furthermore, the VGG-16 architecture yielded a top-5 classification accuracy of 92.7%.

Although the VGG-16 architecture provides a great increase of performance over the AlexNet model thanks to its deeper network, it is not always the case where deeper architectures yield better performance and results. After a particular depth in a neural network, the performance begins to deviate and degrade; this phenomenon is known as the vanishing gradient problem – as the network depth increases, the gradient (used to represent the loss function) would drop to zero. This is where GoogLeNet plays a role; Szegedy et al [2] proposed a 22-layer architecture named GoogLeNet which has only one fully connected output layer, which is the output layer. This architecture is globally known as the lead in various

computer vision tasks such as image classification, face recognition, image recognition, and many more. GoogLeNet is mainly used for image classification as there have been myriad published works in this field. This architecture has solved the vanishing gradient problem through the use of the Inception module – this is a neural network architecture that leverages feature detection at different scales through convolutions with various filters and reduced the computation cost of training a complex network via dimensionality reduction [9]. Part of GoogLeNet 22 layers include nine inception modules. This allows for the utilisation of multiple filter sizes (rather than just one size) in a single image block to which we concatenate and send onto the next layer [2].

Another solution to the vanishing gradient problem is the ResNet architecture (introduced in 2015) which makes use of a skip connection through which the gradient value can flow. He et al [5] presented ResNet as a deep learning neural network architecture that generalises well to the validation data which indicates that the vanishing gradient problem is well addressed thus obtaining higher classification accuracies from increased network layer depth.

B. Improvements

Over the years since the introduction of the aforementioned architectures, there have been several improvements brought forth; one of these improvements is the IVGG model (which is a modification of the original VGG-16 model) where it has a 9-layer architecture. This new model adds max pooling and dropout after multiple convolutional layers simply to extract features from the input to lessen the computational load and decrease training time [10]. This improves the classification or recognition accuracy as the addition of batch normalization and dropout accelerates convergence.

C. Architecture Choice for a Given Dataset

The introduction of convolutional neural networks eradicated the need for handcrafting features thanks to its ability to learn from the input data the specific features. However, not all architectures are suitable for all sorts of datasets as both the depth and width of the network architecture and the dataset could not be suitable. In an attempt to find the most suitable CCN architecture for a given dataset, Basha et al [6] asked three questions in their paper: *1) what is the impact of deeper/shallow architectures on the performance of the CNN to fully connected layer; 2) how the deeper/wider datasets influence the performance of CNN to fully connected layers; 3) which kind of architecture is better or suitable for which kind of dataset.*

While the main objective of Basha et al's work is to evaluate the impact of various hyperparameters related to fully connected layers (specifically the number of fully connected layers and the number of neurons) over the performance, the experiment itself poses more interest to the objective of this paper: the implementation of different CNN architectures. Their training methods also prove crucial to this study's experimentation as the use of various learning rates after every few epochs helps provide empirical justifications for the findings of this paper. Moreover, this study theorises that the use of ReLU as the activation function along with batch normalization for its VGG-16 architecture will produce very accurate results as demonstrated in Basha et al's work [6].

Similar work was done by Palvanov and Cho where they compared deep learning algorithms using the MNIST dataset [7]; the architectures did not comprise a set of convolutional neural networks but included a capsule network, a deep residual learning model, a convolutional neural network, and a multinomial logistic regression model. Palvanov and Cho's work focuses on using said networks and models to recognise hand-written digits and comparing performance and accuracy results obtained in real-time.

V. IMPLEMENTATION

The aim of this paper is to compare the performance and the results of the three deep learning neural network architectures. VGG-16, GoogLeNet, and ResNet-34 were all trained and validated on the MNIST dataset. During training, there were two learning rates (0.01 and 0.001) and two numbers of epochs (10 and 25) utilised. With each architecture getting four runs at different learning rates and epochs, we were able to gain great insight into each architecture's performance. However, as mentioned earlier in this paper, VGG and GoogLeNet are typically trained with images with dimensions 224x224 pixels which could be problematic as the MNIST images are 28x28. Therefore, the MNIST

images were resized to 80x80 – if the image dimensions were resized or increased any further, unwanted artefacts would appear within the image. Moreover, the PyTorch gradient optimiser used is Stochastic Gradient Descent (SGD) as it performs better with longer training time. The original optimiser choice was Adam as it provides very fast performance; this choice was neglected as Adam is known for having myriad convergence issues.

The code for each architecture consists mainly of four parts: 1) importing the dataset and splitting it into training and testing datasets, 2) initialising the respective classes, 3) training the model, and 4) creating plots to record accuracies and losses. During each run, the code displays the current epoch in the iteration as well as its train and test accuracies with the time elapsed since the start of the epoch. All results were recorded with their respective GPU usages as well. The hardware used for this study and experimentation is an NVIDIA RTX 3080 graphics processing unit which has 10,240 CUDA cores and runs at 1950 MHz along with a GDDR6X memory of size 10GB.

VI. RESULTS

The tables below represent the performance and results of each neural network architecture. The statistics were recorded at learning rates 0.01 and 0.001 at the 10^{th} and 25^{th} epochs. Each run has records of the training and testing accuracies along with the time taken for each run.

VGG-16 – Performance and Results				
Epochs	Learning Rate	Training Accuracy (in %)	Testing Accuracy (in %)	Time Elapsed (in sec)
10	0.01	99.063	99.030	839.08
	0.001	99.553	98.960	834.36
25	0.01	99.623	99.080	2046.82
	0.001	99.900	99.200	1999.02
Average GPU Usage: 99%				

Table 1: VGG-16 Performance and Results

GoogLeNet – Performance and Results				
Epochs	Learning Rate	Training Accuracy (in %)	Testing Accuracy (in %)	Time Elapsed (in sec)
10	0.01	99.788	99.440	673.98
	0.001	99.918	99.550	678.69
25	0.01	99.947	99.645	1687.21
	0.001	99.982	99.540	1695.40
Average GPU Usage: 66%				

Table 2: GoogLeNet Performance and Results

ResNet-34 – Performance and Results				
Epochs	Learning Rate	Training Accuracy (in %)	Testing Accuracy (in %)	Time Elapsed (in sec)
10	0.01	99.523	99.260	462.07
	0.001	99.683	99.130	473.50
25	0.01	99.845	99.270	1146.01
	0.001	99.963	99.320	1177.58
Average GPU Usage: 83%				

Table 3: ResNet-34 Performance and Results

As seen in the tables above, each run yields a 99% accuracy on both training and testing datasets. However, attention is drawn to the total elapsed time. Despite having the most number of layers compared to the other architectures of this study, ResNet-24 performed the fastest at both learning rates and both epochs. However, despite having the least number of layers compared to the other architectures of this study, VGG-16 performed the slowest at both learning rates and both epochs. Moreover, the smaller learning rates caused a slightly slower performance on GoogLeNet and ResNet-34, but not on VGG-16.

VII. CONCLUSION

This research paper evaluates and compares the performance and accuracy of various convolutional neural network architectures and their respective effects on the training and testing set accuracies using the MNIST database. The three architectures used for evaluation in this paper are VGG-16, ResNet-34, and GoogLeNet. Each architecture was evaluated at different learning rates and epochs to provide an adequate comparison. Of the three architectures, ResNet-34 performed the fastest with VGG-16 yielded the slowest performance. Nevertheless, all three architectures produced 99% accuracy at various combinations of learning rates and epoch. Further work would include comparing more architectures and running them using various other learning rates at different epochs to further evaluate performance.

VIII. REFERENCES

[1] A. Krizhevsky, I. Sustkever and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks. 2012.

[2] C. Szegedy et al., Going Deeper with Convolutions. 2014.

[3] V. Alto, "Understanding the Inception Module in Googlenet", Medium, 2021. [Online]. Available: https://valentinaalto.medium.com/understanding-the-inception-module-in-googlenet-2e1b7c406106. [Accessed: 17- May- 2021].

[4] Understanding and visualizing ResNets", Medium, 2021. [Online]. Available: https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8. [Accessed: 17- May- 2021].

[5] K. He, X. Zhang, S. Ren and J. Sun, “Deep Residual Learning for Image Recognition,” in CVPR, 2016.

[6] S.H.S. Basha, S.R. Dubey and V. Pulabaigari et al., Impact of fully connected layers on performance of convolutional neural networks for image classification, Neurocomputing, https://doi.org/10.1016/j.neucom.2019.10.008

[7] A. Palvanov and Y. Cho, "Comparisons of Deep Learning Algorithms for MNIST in Real-Time Environment", 2018. [Online]. Available: http://www.ijfis.org/journal/view.html?uid=824&page=&pn=mostread&sort=publish_Date%20DESC&spage=&vmd=Full. [Accessed: 19- May- 2021].

[8] K. Simonyan, A. Zisserman, Very deep convolutional networks for largescale image recognition, arXiv preprint arXiv:1409.1556.

[9] "Deep Learning: GoogLeNet Explained", Medium, 2021. [Online]. Available: https://towardsdatascience.com/deep-learning-googlenet-explained-de8861c82765. [Accessed: 20- May- 2021].

[10] S. Zhou, W. Liang, J. Li and J. Kim, "Improved vgg model for road traffic sign recognition," Computers, Materials & Continua, vol. 57, no.1, pp. 11–24, 2018.