Deep Networks for Image Classification

Performing and evaluating image classification tasks with deep CNN networks

Mughees Asif–180288337
School of Electronic Engineering and Computer Science
Queen Mary, University of London

***Abstract*—Deep learning has instigated a seismic shift in the research associated with the domain of image classification. The use of non-parametric classifiers, such as neural networks, have produced models not only with a higher classification accuracy but also the ability to generalise well on unseen data. This research focuses on the quantitative evaluation of the theoretical abstractions and practical implementations of three convolutional neural networks that have emerged as the de facto models capable of recognising and categorising images.**

***Index Terms*—image classification, deep learning, convolutional neural networks.**

I. Introduction

Image classification is the task of reorganising images into one or more predefined classes using computer vision techniques such as object localisation, image segmentation, and feature detection [1]. The task is achieved by training a model to discern regularities or patterns from input test images with labelled pixels. At first instances the task exudes triviality, however, several complications can arise, for example whether the perception of the features in an image should be objective or subjective, and recognising multiple labels with similar features [2].

Deep learning methods have recently been exploited to introduce non-linear information mapping that enables autonomous feature extraction and pattern analysis [3]. In particular, convolutional neural networks (CNNs) have achieved benchmark progress in image classification through deep learning within the computer vision domain. Deep learning for image classification can be separated into two main groups:

- *Supervised*: Each input image has one label and the model generates one prediction [1]. The model learns from a labelled dataset which requires maximal human intervention (time consuming to annotate the images one-by-one), however the accuracy tends to be higher. Common methods include classification and regression, and applications range from sentiment analysis to weather prediction.
- *Unsupervised*: The input images have no labels and the model utilises algorithms to get insights from a high-volume real-time stream of incoming image data [1]. The model requires minimal human intervention (only need to validate the output variables) but can be computationally expensive as training has to be conducted to cover corner and edge cases. Common examples include clustering and principle component, and applications range from recommendation systems to anomaly detection.

The premise of this research is to develop, explore and test the architecture of several variants of CNNs that will be trained on the `MNIST` dataset to enable supervised image classification. The remaining research is organised into several sections: Section II details the origin of several state-of-the-art neural networks for image classification. Sections III and IV describe the architecture of the experimental networks and the training dataset. Section V and VI highlight the training results and the accompanying empirical analysis, and finally Section VII delivers a conclusion to this research.

II. Background

Conventionally, the image classification problem was solved by using a dual-stage approach, where hand-crafted features were input into a trainable classifier [1]. CNNs were first introduced by LeCun et al. in 1995 to leverage the feedforward architecture of vanilla deep networks with the augmentation of convolution and pooling operations contained within the network as modules [4]. Deep CNNs (DCNNs) were introduced by incorporating deep learning via the usage of Graphical Processing Units (GPUs), easy of access to larger datasets, and development of contemporary networks [1], [5], [6].

ImageNet (also known as AlexNet) was the first network to introduce deep CNNs to the computer vision domain that mimicked the original CNN structure by

LeCun et al., but with the caveat of multiple layers stacked together with non-saturating neurons and regularisation to reduce overfitting [7]. Moreover, Zeiler and Fergus demonstrated an improvement to this network by manipulating the kernel sizes for each convolution layer and retraining the softmax classifier [8].

A major contender for the top spot within image classification came with the development of the Visual Geometry Group (VGG) network that implements small receptive fields (3×3) with a stride length of 1. Effectively, VGG increased depth by adding more layers per network with an increased capacity to parse non-linearity, which ultimately led to better objective performance [9]. However, increasing the depth of the network translated into an increase in the number of trainable hyper-parameters that led to an increase in the computational memory cost.

The problem of declining computational efficiency within the image classification domain was tackled by the GoogLeNet architecture that achieved state-of-the-art results by introducing an 'Inception' architecture that is "… based on finding out how an optimal local sparse structure in a convolutional vision network can be approximated and covered by readily available dense components" [10]. The network introduced dimension reduction to offset the need for over-processing an area with sparse embeddings, and periodic average max-pooling at the top of each layer with a stride of 2 (halves the resolution of the input pixel grid) [10].

More recently, breakthroughs have been made in reducing the computational cost of DCNNs by addressing the degradation problem that can be cited as "… the poor propagation of activation [functions] and gradients because of stacking several non-linear transformations on top of each other" [1]. The Residual Network (ResNet), built on the principles of VGG, introduced deep residual mapping by implementing shortcut connections that enable the layer to fit the mapping (instead of the other way around) by utilising batch normalisation (for the vanishing gradient problem), and skipping layers (degradation) to curtail the training error [11]. Skipping layers renders the overall network open to the modulation of the number of trainable hyper-parameters, therefore increasing the overall computational efficiency.

III. Architectures

A. VGG-16

VGG networks were developed to improve upon the original CNN architecture by exploring the utility of increasing the depth of the network, in terms of the number of layers in various configurations. The associated computational memory cost was mitigated by decreasing the size of the convolution layers to 3×3. Figure 1 displays the blueprint of the network used in this research, namely VGG-16, where the number 16 denotes the number of layers.

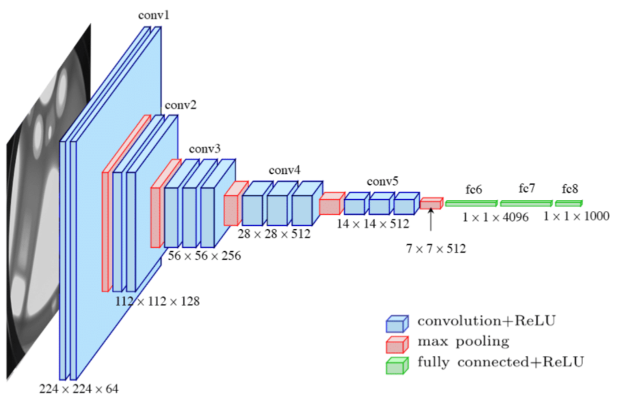


Fig. 1. VGG-16 overall architecture [12]

The input images were loaded as fixed-size 224×224 RGB images with 64 channels for the first layer. The number of channels increases by a factor of two for each consequent layer till the channels equal 512. As a baseline preprocessing step, the mean RGB value from the training set was subtracted from each pixel and the image was funnelled through a stack of convolutional layers with small receptive fields of which 3×3 is the smallest size capable of capturing the key spatial information i.e., up/down, left/right, and center [9]. The stride of the convolutional layers was set to one which is emblematic of the movement of the kernel over the number of pixels, the spatial padding was set such that the layers correlated with the perseverance of the spatial resolution i.e., the padding was one pixel per 3×3 layer, and the spatial pooling was carried out with five max-pooling layers moving over a 2×2 pixel window with a stride equalling 2 [9]. The configuration of the first two layers included 4096 channels each, where the third and fourth layer were responsible for executing the classification and softmax activation respectively. The training was run using different optimiser functions ranging from Stochastic Gradient Descent (SGD) to Adam. The learning rates were also modulated between 0.001 and 0.01.

B. GoogLeNet

GoogLeNet builds upon the VGG network by streamlining the symbiosis between computational budget and deep networks. The network introduced the concept of ‘Inception’ modules that organised the 22 network layers using sustainable (in terms of the learning process) levels of depth whilst maintaining the accuracy of the classifier. Figure 2 highlights the overall blocks of the network used in this research.

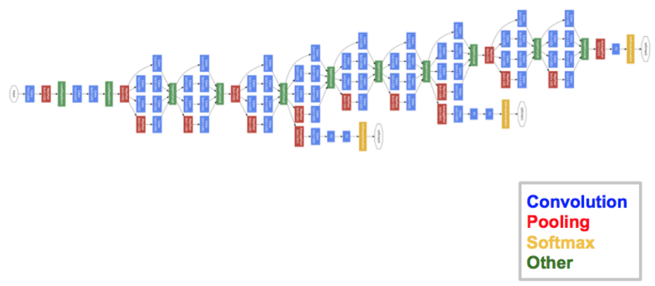


Fig. 2. GoogLeNet architecture with the corresponding layers [10]

The ‘Inception’ blocks form “... a network consisting of modules of the [same] type stacked upon each other, with occasional max-pooling layers with stride [two] to halve the resolution of the grid” [10]. The computational budget of the network is reduced by curtailing the size of the input image as it is propagated through the network, and dimension reduction. The overall model was developed for an input dimension of 224×224 RGB images with mean subtraction and rectified linear activation for the convolution operations. The first convolution layer utilised a 7×7 kernel size with a stride length of two to ensure that the local spatial information is retained before the image size was further reduced [10]. The subsequent convolution layers reduced the image size by a factor of four and eight respectively. Moreover, the second convolution layer used the 1×1 kernel size producing the dimensionality reduction effect of decreasing the computational cost of the model. The network consisted of nine total ‘Inception’ blocks, some of which were followed by a max-pooling layer, with fixed kernel sizes of 5×5, 3×3, and 1×1 [10]. The output of all the blocks were concatenated before passing through the average-pooling layer which computed the mean of each feature map (holds the probability distribution) with a 7×7 kernal and stride of length one. The output was then progressed through the dropout layer with a value of 0.4 to prevent data overfitting, before passing through the final softmax activation layer that converted the output vector values into a probability distribution. Additionally, auxiliary classifiers were also placed between the blocks to mitigate the vanishing gradient problem. Finally, the SGD and Adam optimiser functions were used and different learning rates were implemented to observe the learning behaviour of the model.

C. *ResNet*-18

ResNet was developed to ease the training of deep neural networks by implementing a residual learning framework that inserted identity shortcut connections periodically between the layers [11]. The network optimises the definition of the layers to nullify the vanishing gradient problem by learning residual functions with reference to the inputs, instead of learning from unreferenced functions [11]. Figure 4 shows the overall architecture of the ResNet-18 configuration used in this research.

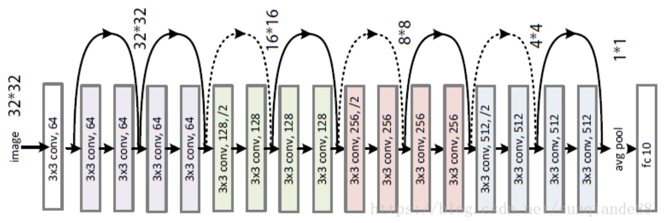


Fig. 3. ResNet-18 architecture [13]

The input image was cropped to 224×224 dimensions with the mean of each pixel subtracted. The first convolutional layer was constructed with a 7×7 kernel size with a stride of two, and the second max-pooling layer maintained a 3×3 kernel size with a stride of two [11]. Furthermore, four consecutive residual blocks with shortcut connections were implemented. The output of each of the blocks was aggregated and reformed as the input for the subsequent blocks. Each block consisted of two 3×3 convolutional layers followed by a batch normalisation layer and the Rectified Linear Unit (ReLU) activation function layer. SGD is also used with a mini-batch size of 256 and the learning rate is kept between 0.001 and 0.1. The aggregated outputs were sifted through the average-pooling layer, before the final feature map was passed through the fully-connected layers [11]. Lastly, the training was run with different learning rates and optimisers.

IV. MNIST DATASET

The MNIST dataset of handwritten digits was used to verify the accuracy of the neural networks. The dataset consists of $60e^3$ training images and $10e^3$ test images, all of which have been dimensionally normalised to a measurement of 28×28 and are represented in the one-channel greyscale format [14].

Fig. 4. MNIST dataset of handwritten digits

V. Results

The models were developed using the industry-standard Python programming language and the PyTorch library that presented a plethora of useful functions for research prototyping. All three models were trained on Google Colaboratory with a basic account and standard GPU allocations for a total of ten epochs while the batch size was constrained to 32. The sklearn classification report was used to evaluate the models and the results have been recorded in Table I with the visual representation of the training process (see Appendix A to access the complete training logs) depicted in Figures 5, 6 and 7.

TABLE I
Evaluation metrics

Model	Learning Rate	Accuracy (%)
VGG-16	0.001	99.1
VGG-16	0.01	98.4
VGG-16 with SGD	0.001	99.1
VGG-16 with Adam	0.001	98.2
GoogLeNet	0.001	98.7
GoogLeNet	0.01	81.5
GoogLeNet with SGD	0.001	99.1
GoogLeNet with Adam	0.001	99.0
ResNet-18	0.001	99.2
ResNet-18	0.01	98.0
ResNet-18 with SGD	0.001	99.4
ResNet-18 with Adam	0.001	98.3

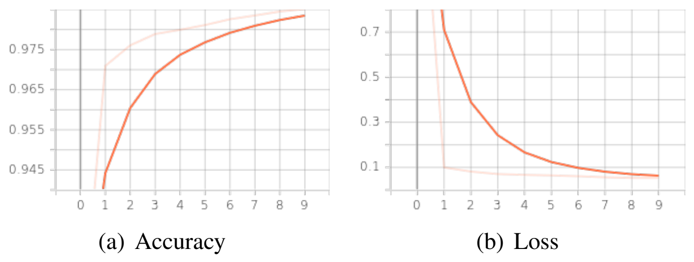


Fig. 5. VGG-16 with SGD

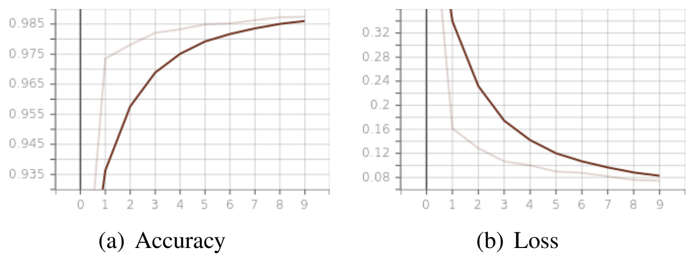


Fig. 6. GoogLeNet with SGD

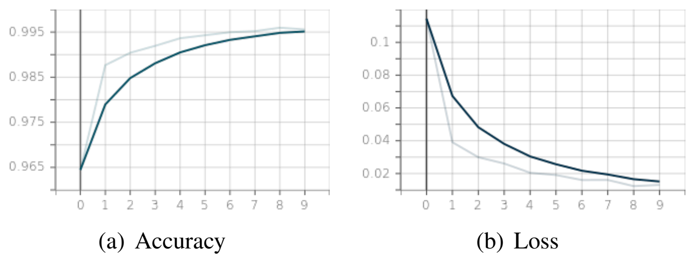


Fig. 7. ResNet-18 with SGD

VI. Discussion

A. Analysis

Observable from Table I and Figure 5, the first batch of models that were developed include the variants of the VGG-16 network. The first two entries in the batch were vanilla VGG-16 models with differing learning rates α, where an α of 0.001 displays optimal training conditions with a percentage increase of 0.7% over the larger α of 0.01. Within the batch, the variant with the highest recorded accuracy was the network with the SGD optimiser. SGD is a well-known deep learning training algorithm that performs well empirically with positive certainties in avoiding the saddle point (critical points in non-convex functions where the algorithm can get stuck), increased generalisation to different functions, and enhanced interpretations using Bayesian inference [15]. The iterative process of SGD is derived as:

$$w_k \leftarrow w_{k-1} - \alpha_{k-1}\hat{\nabla} f(w_{k-1}) \tag{1}$$

where w_k represents the iteration, α_k is the learning rate (how big a step size to take), and $\hat{\nabla} f(w_{k-1})$ represents the previous iteration's SGD. Interestingly, when comparing the results in Table I, the network with the Adam optimiser performs marginally worse than the SGD counterpart. Adam is an adaptive method that was proposed to offset SGD's uniform movement in all directions of the gradient (caused scalability issues for opaquely defined problems) [15]. The basic premise of the method is to use cues from the curvature of the gradient space to diagonally scale the learning loss (vector of values as opposed to SGD's scalar learning rate α) towards the global minima. Although, Adam has been tested to exhibit faster convergence and relative insensitivity to hyperparameter tuning, the generalisation of the developed models was at best sub-par. In fact, it can be noted from Section V that SGD outperforms Adam in every variation, which is also in line with the experimental results from [15], [16]. Moreover, it should be noted that the training time for the VGG-16 model with SGD was the longest (≈ 4 hours) in this complete research and directly correlated with the large number of hyperparameters (15 million).

The second batch consisted of the GoogLeNet architectures where the percentage difference between the accuracy of the SGD and Adam optimiser models with a $\alpha = 0.001$ is evidently negligible. The two baseline models with differing α highlight an accuracy percentage difference of 19% that is considerably more than the VGG-16 models and one of the lowest accuracy's in this research. This can be explained by examining the 'Inception' modules of the network that use variable kernel sizes and fixed-size convolution layers to speed up the training process (see Section III-B). The α effectively modulates the subsequent kernel sizes and plays a vital role in the overall network architecture. Indeed, GoogLeNet with SGD was the third-fastest (≈ 1.5 hours), just after ResNet-18 with SGD (≈ 1.2 hours), in terms of the training time with a total of 10.8 million hyperparameters. This reaffirms the computational processing rapidity of Adam, yet as a result, Table I highlights the slight drop in the accuracy of the optimisation, as compared to SGD, mainly due to the plateauing of the learning process. Furthermore, if the reader visually inspects Figure 6(a) and 6(b), the conventional graphical inferences of good generalisation with an optimal learning rate can be observed.

The final batch consisted of the ResNet-18 models (11 million hyperparameters) and a negligible difference in the accuracy between the two baseline models with different α was observed. The ResNet-18 with SGD model achieved the highest accuracy in this research and is emblematic of the power of using shortcut connections on the computational memory cost and the overall accuracy of the classifier (see Section III-C). The model was also the second-fastest to train, in addition to the visual depictions in Figure 7(a) and 7(b) of favourable training metrics. The ResNet model used batch normalisation to speed up training by making "... the landscape of the corresponding optimization problem significantly more smooth. This ensures, in particular, that the gradients are more predictive and thus allows for use of larger range of learning rates and faster network convergence" [17].

B. Further work

This research can be further improved upon by using a different dataset such as CIFAR-10[1] and drawing comparisons between learning from simple one-greyscale images and color images with different RBG channels. The optimiser should also be investigated by adding momentum-based accelerations and/or random initialisation following work conducted in [18]. Modulating the batch sizes, number of epochs, and the usage of multiple GPU/TPU would also greatly increase the applicability of these models in image classification tasks.

VII. Conclusion

The aim of this research was to develop and analyse several architectural enhancements on vanilla CNNs for the image classification task within the context of the computer vision domain. All the three models that were tested displayed definitive improvement at each iteration of the architectural change, in addition to the tuning of the hyperparameters yielding incremental progress in terms of the computational budget and classifier accuracy. The experimental results have been recorded and analysed with the trends and anomalies highlighted. Lastly, improvements to the experiment have been suggested to further enhance the development of the models.

References

[1] Rawat, W. and Wang, Z. (2017). "Deep convolutional neural networks for image classification: A comprehensive review". Neural computation, 29(9), 2352-2449.

[1]The author could not run training on the CIFAR-10 dataset as the computational load was much higher than the available resources.

[2] Ciresan, D. C., Meier, U., Masci, J., Maria Gambardella, L., & Schmidhuber, J. (2011). “Flexible, high performance convolutional neural networks for image classification”. In Proceedings of the International Joint Conference on Artificial Intelligence (vol. 1, pp. 1237–1242). Menlo Park, CA: AAAI Press.

[3] LeCun, Y., Bengio, Y., & Hinton, G. (2015). “Deep learning”. Nature, 521(7553), 436–444.

[4] LeCun, Y. & Bengio, Y. (1995). “Convolutional networks for images, speech, and time series”. The handbook of brain theory and neural networks, 3361(10), p.1995.

[5] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). “Improving neural networks by preventing co-adaptation of feature detectors”. arXiv:1207.0580.

[6] Deng, L., & Yu, D. (2014). “Deep learning: Methods and applications”. Foundations and Trends in Signal Processing, 7(3–4), 197–387.

[7] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012) “Imagenet classification with deep convolutional neural networks.” Proceedings of the 25th International Conference on Neural Information Processing Systems.

[8] Zeiler, M. D., & Fergus, R. (2014). “Visualizing and understanding convolutional networks”. European conference on computer vision. Springer, pp. 818–833.

[9] Simonyan, K., & Zisserman, A. (2014). “Very deep convolutional networks for large-scale image recognition”. arXiv:1409.1556.

[10] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). “Going deeper with convolutions”. Proceedings of the IEEE conference on computer vision and pattern recognition, 1-9.

[11] He, K., Zhang, X., Ren, S., & Sun, J. (2016). “Deep residual learning for image recognition”. Proceedings of the IEEE conference on computer vision and pattern recognition, 770-778.

[12] Ferguson, M., Ak, R., Lee, Y. T., & Law, K. H. (2017) “Automatic localization of casting defects with convolutional neural networks”. IEEE International Conference on Big Data (Big Data), 1726-1735, doi: 10.1109/BigData.2017.8258115.

[13] Han, S.S., Park, G.H., Lim, W., Kim, M.S., Na, J.I., Park, I. & Chang, S.E. (2018). “Deep neural networks show an equivalent and often superior performance to dermatologists in onychomycosis diagnosis: Automatic construction of onychomycosis datasets by region-based convolutional deep neural network”. PloS one, 13(1), p.e0191493.

[14] Deng, L. (2012). “The mnist database of handwritten digit images for machine learning research [best of the web]”. IEEE signal processing magazine, 29(6), 141-142.

[15] Keskar, N.S. & Socher, R. (2017). “Improving generalization performance by switching from adam to sgd”. arXiv:1712.07628.

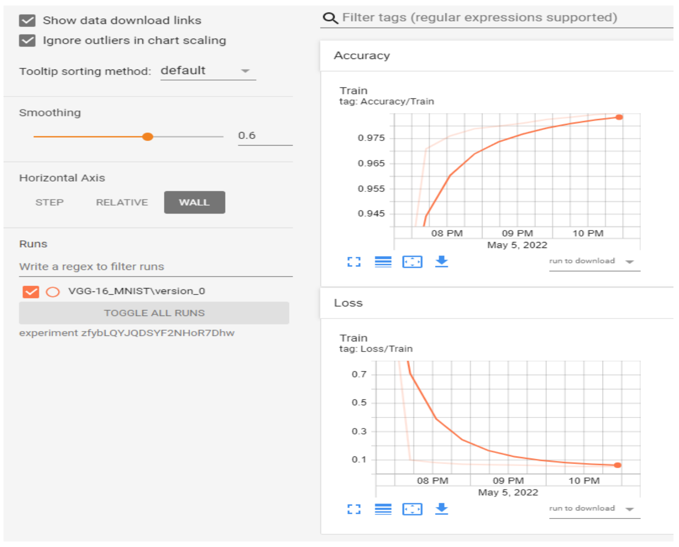
[16] Zhang, K., Wu, Q., Liu, A. & Meng, X. (2018). “Can deep learning identify tomato leaf disease?”. Advances in multimedia.

[17] Santurkar, S., Tsipras, D., Ilyas, A. & Madry, A. (2018). “How does batch normalization help optimization?”. Advances in neural information processing systems, 31.

[18] Sutskever, I., Martens, J., Dahl, G. & Hinton, G. (2013). “On the importance of initialization and momentum in deep learning”. In International conference on machine learning. 1139-1147, PMLR.

Appendix A
Tensorboard Logs

The associated Tensorboard logs for each model have been uploaded to the official `Tensorboard.dev` platform. In addition, the time stamps of the training process can also be accessed by clicking the **WALL** option located in the `Horizontal` section under each tab, on the left-hand side (Figure A).



☑ Show data download links
☑ Ignore outliers in chart scaling
Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis
STEP RELATIVE [✓ WALL]

Runs
Write a regex to filter runs
☑ VGG-16_MNIST\version_0
TOGGLE ALL RUNS
experiment zfybLQYJQDSYF2NHoR7Dhw

A. VGG-16

https://tensorboard.dev/experiment/zfybLQYJQDSYF2NHoR7Dhw/

B. GoogLeNet

https://tensorboard.dev/experiment/iAIMauLrQQiOR1UvaLWiZA/

C. ResNet-18

https://tensorboard.dev/experiment/K3cQFOkjQfWojgPcOgrWZg/