UNIVERSITY OF SURREY

EEEM063 COURSEWORK

A study into the optimisation of Convolutional Neural Networks

6482709

Table of Contents

Table of Figures

Introduction

This coursework aims to study different Convolutional Neural Network architectures and how changing parameters can affect their performance in image classification. In Section 1, 5 different network architectures are compared to investigate the effect network depth has on classification accuracy of a relatively small dataset. Section 2 compares the effects that data augmentation and increasing validation splits can have on top-1 accuracy. Finally, Section 3 inspect how the tuning of the Metaparameters used in CNN training can improve learning results.

All code used for the coursework including data gathering, data augmentation, and network prototxt files can be found on the project's GitHub page [1].

1 CNN Architectures

In this section we will be looking at the effectiveness of various network architectures when applied to a dataset. To effectively compare the architectures the dataset split will remain at 89%/9%/2% across training, validation, and testing, respectively across the augmented dataset constructed in Section 2.2 to reduce the likelihood of overfitting. Furthermore, the maximum epochs are set at 30 at a base learning rate of 0.01, all other parameters in DIGITS are left as default. The networks looked at in this section are: GoogLeNet, AlexNet, BN-AlexNet, VGG16 and VGG11, the results for which are summarised below:

Accuracy	**GoogLeNet**	**AlexNet**	**BN-AlexNet**	**VGG16**	**VGG11**
Top-1	77.59%	74.14%	86.21%	65.52%	72.41%
Per-class average	77.57%	73.63%	86.36%	66.06%	72.42

1.1 GoogLeNet

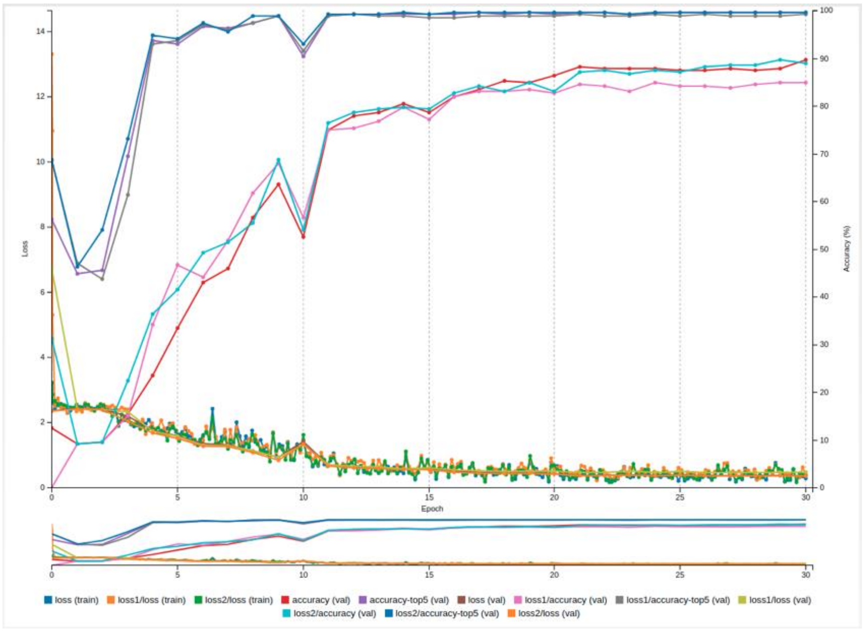

Figure 1: Learning graph for GoogLeNet with base parameters

GoogLeNet is the deepest network studied in this coursework, with a depth of 27 layers. The network shows higher accuracy at the base learning rate when compared to the next deepest network, VGG 16. The results show a plateau in loss after epoch 20 which corresponds to a drop in learning rate to 0.001. Due to the network's effectiveness at base conditions, it will not be used to further explore the effect of altering Metaparameters.

1.2 AlexNet

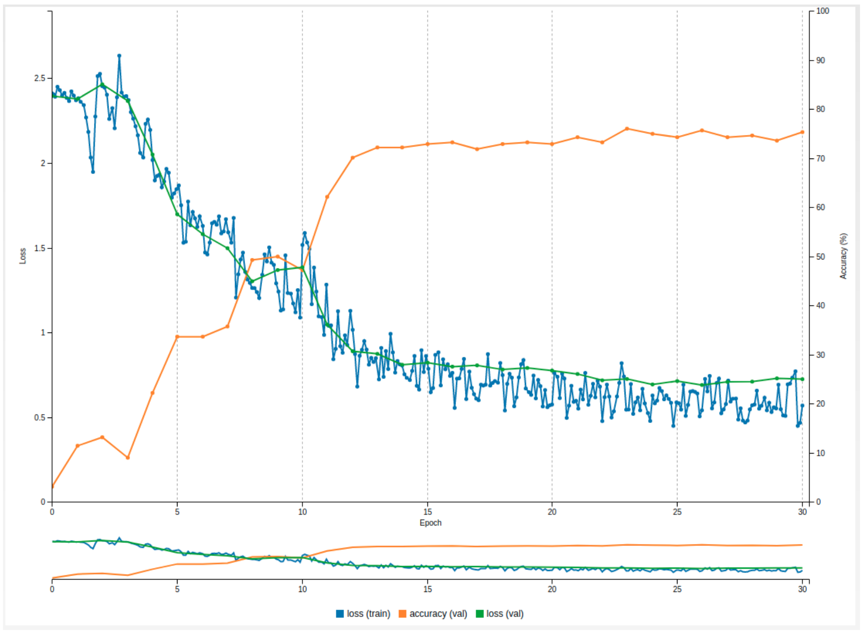

Figure 2: Learning graph for AlexNet with base parameters

AlexNet is the second network which is prebuilt into DIGITS. The network is much less complex than GoogLeNet with only 5 convolution layers. The classification results show a slightly reduced accuracy comparted to GoogLeNet. The data shows a plateau in accuracy after 12 epoch and begins to show signs of early overfitting in later epochs as the training and validation losses begin to diverge.

1.3 BN-AlexNet

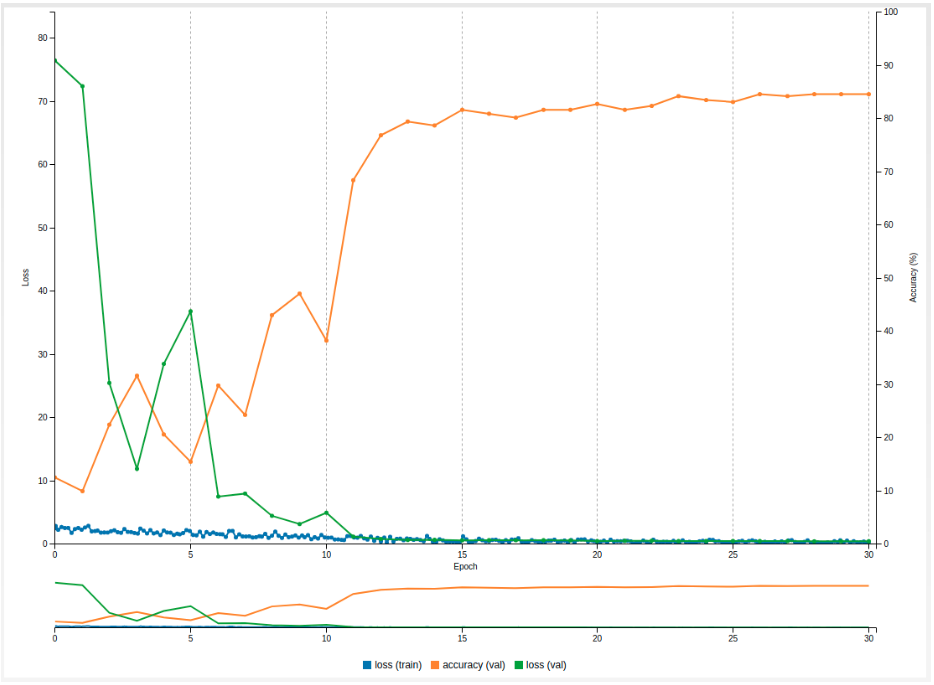

Figure 3: Learning graph for BN-AlexNet with base parameters

BN-AlexNet is a variant on AlexNet which substitutes the Local Response Normalisation (LRN) layers found in the base AlexNet for Batch Normalisation (BN) layers. BN layers have the added advantage of having trainable parameters with the aim of reducing internal covariate shift, which can improve the networks effectiveness for higher learning rates [2]. This network shows better performance than the base AlexNet, however experiences the same plateau by 15 epochs however at a higher top-1 accuracy. The network experiences the highest loss of any other network in early epochs (>10x base AlexNet), potentially owing to the extra tuning required for the BN layer's initial values.

1.4 VGG16 & VGG11

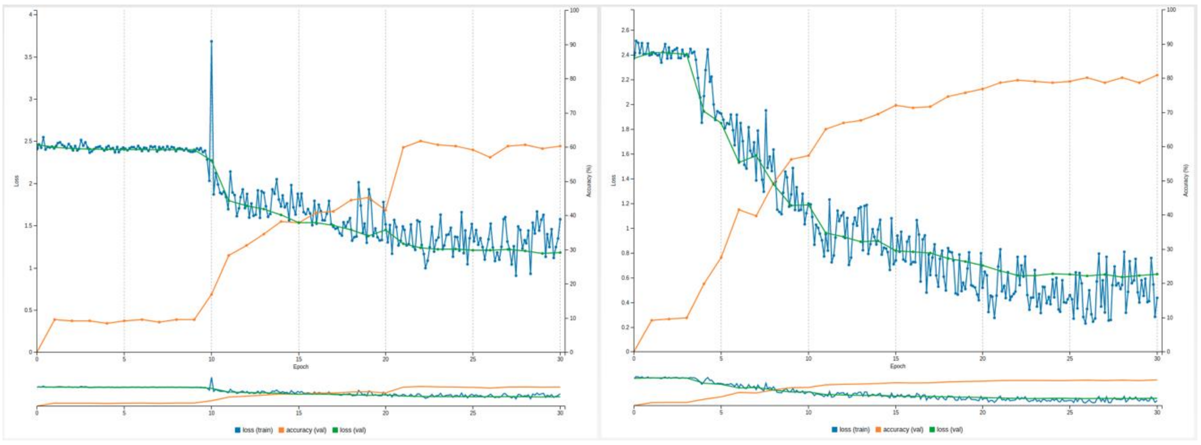

Figure 4: Learning graph for VGG16 (left) and VGG11 (right) with base parameters

The VGG16 and VGG11 networks studied in this section are derived from the ConvNet configurations ‘C’ and ‘A’ found in [3], respectively. The difference in these networks occurs in the number of convolution layers which appear in each stages’ loop, with VGG11 having one less layer per loop, amounting to 5 fewer layers than VGG16. As seen in Figure 4, VGG11 has a far greater performance on this dataset, this can be attributed to a reduced network depth, which is more effective on smaller datasets. The VGG16 network is shown to have the lowest accuracy at base conditions and be the most sensitive to changes in learning rate, with the biggest improvements in loss occurring when the learning rate is reduced by a factor of 10x at epochs 10 and 20.

2 Dataset pre-processing and preparation

2.1 Image Gathering

The first step in training the neural networks was to construct a dataset of images. Based on the ratios of the recommended “Cars” dataset, each class should contain at least an average of 83 images across the training, validation, and testing splits. As we are constructing our own dataset, a reduced set of 11 classes has been chosen. To compensate for the reduced processing time, it was decided to increase the ratio seen in the “Cars” dataset to ~160 images per class. The classes chosen for this dataset are different championship winning Formula 1 cars, some of which are shown below:

McLaren Mp4/4

Ferrari 312T

Ferrari F2004

Mercedes W10

Williams FW18

These cars were chosen as suitable classes owing to their unique designs which arise from the technical regulations of the time such as the tall airbox of the 312T or the introduction of the “halo” in the W10. To collect the data a python script was written utilising the “google_image_download” python library [4] which downloads images matching the specified key words. The script is set to download a total of 200 per class to allow the dataset to be manually cut down to remove images that do not specifically show the desired class. After cutting, each dataset was composed of ~160 images. The script is also set to exclusively download colour images to ensure the colour scheme remains consistent specifically for older cars such as the 312T which have photographs in black and white.

2.2 Dataset Augmentation

Dataset augmentation is a method by which a dataset can be artificially enlarged by introducing augmented variants of the original dataset into the training stage of a CNN, which has been shown to reduce the likelihood of overfitting [5]. To achieve this in our own dataset, a python script was written using the Keras library [6] which would take 50% of each class’ dataset and apply a random combination of augmentations and save it back to the dataset. These augmentations have their limits set to ensure that the processed image will still be recognisable as the ground truth and can include any combinations of shifting (x and y), rotating, shearing, and zooming.

2.3 Data Splits

To test the effect of Train/Validation/Test splits the data has firstly been separated into base and augmented datasets. The tests will be run on 5 split variations on the base AlexNet, VGG11 was considered however the network was found to be too inconsistent, with lower Train percentages occasionally never progressing past the base accuracy assumption of ~9%.

Figure 5 shows the variation in top-1 accuracy across various splits in validation data. As expected, the values for the augmented data set demonstrate higher accuracy than their base counterpart. This can potentially attribute to the augmentation providing less overfitting or simply there being fewer images in the base data set. The most interesting results arise from the base datasets. Firstly, a steeper decline in performance is seen than in the augmented set, suggesting a minimum image count required for AlexNet to successfully learn. Secondly, the base dataset begins to outperform the augmented set at 18% validation split, even though the augmented set has more training images in its lowest training percentage than any of the validation splits studied. A potential cause for this could be the sensitivity of AlexNet to its starting seed, which was set to random for these experiments, when a controlled seed would sometimes cause the network to not learn.

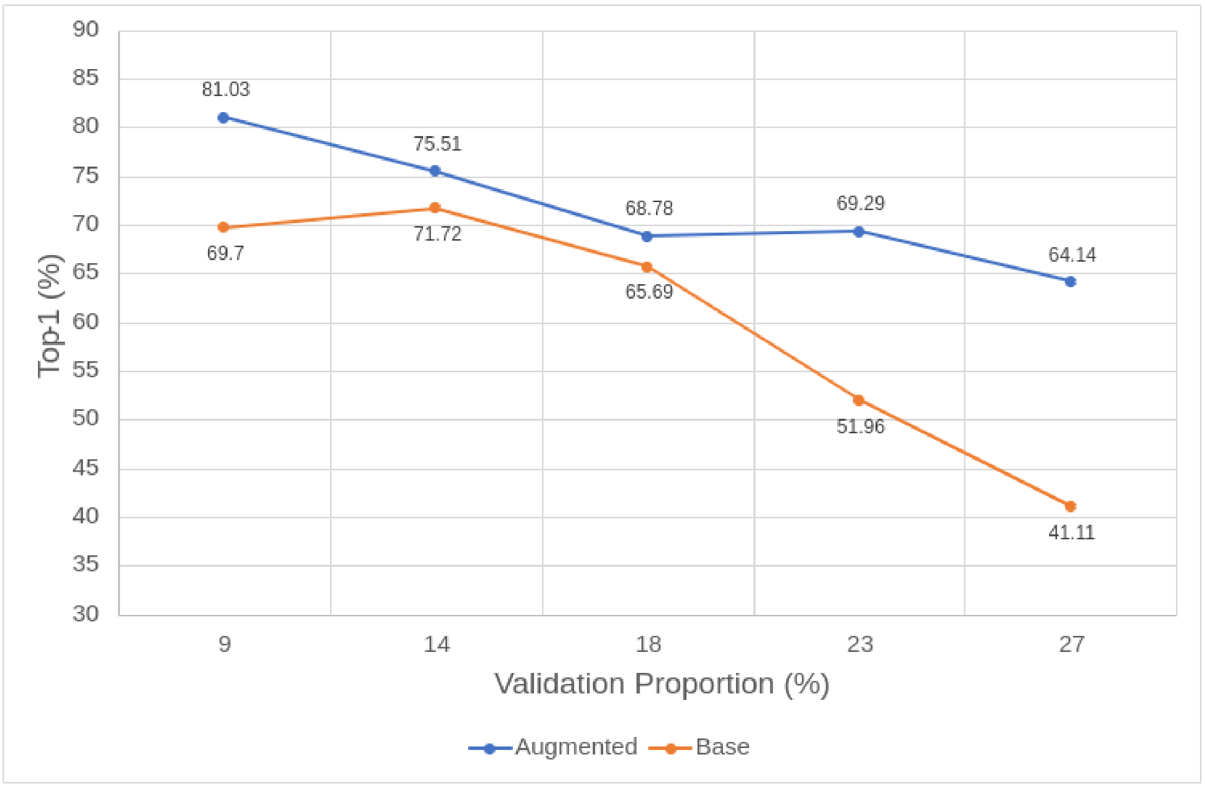

Figure 5: Variation in top-1 accuracy with an increasing dataset validation proportion

3 Metaparameters

This section aims to look at the different Metaparameters which can be altered to provide the best possible learning for a neural network. These parameters include the epoch length, base learning rate and step-down policy. The first architecture studied was AlexNet, which had its learning rate varied by factors of 10x (0.1 and 0.001) versus the base AlexNet parameters studied in Section 1.2 (0.01). Figure 6, shows the training graph for learning rates of 0.1 and 0.001. At 0.1 it is evident that the learning rate is too high, with loss remaining at ~90 throughout the 30 epochs and a top-1 accuracy of 10.34% which is within the range of the networks initial assumptions based on 11 classes. In contrast, the rate of 0.001 does show learning with a top-1 accuracy of 37.93%, however for a duration of 30 epochs this rate is too slow, showing a sweet spot for AlexNet at a rate of 0.01 for a 30 epoch training length.

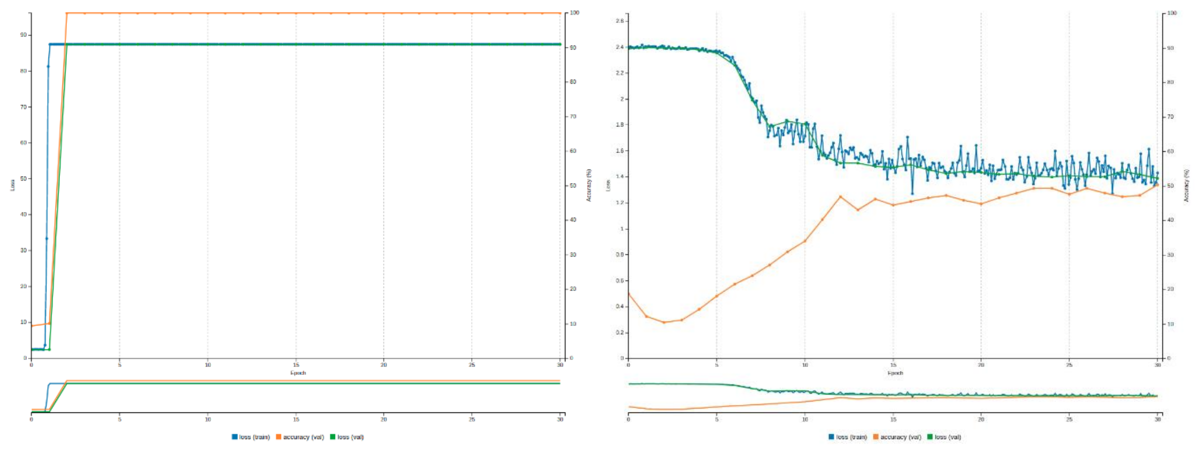

Figure 6: Comparsion of learning rates 0.1 (left) and 0.001 (right) on AlexNet network

The lowest performance for a network in Section 0 was VGG16. The learning graphs show minimal change to loss in the early epochs while the learning rate remains at 0.1, it was therefore experimented to see if other learning rate stepdown policies would provide better results. The two policies studied were Exponential and Sigmoid, the results for which are shown in Figure 7. Exponential decay shows a similar problem to that seen in Step Down, with high loss being maintained in the 0-9 epoch range, resulting in a top-1 accuracy of 64.42%. Sigmoid showed a vastly improved top-1 accuracy of 84.48%, this improvement could potentially be due to the Sigmoid learning rate starting at the base learning rate value, versus Step Down policy which starts at a factor of 10x the base. Furthermore, it was shown in Section 1.4 that VGG16 has its greatest drop in loss when the learning rate hits 0.01, this optimum for Sigmoid which spends a large proportion of its learning time within the order of the base learning rate.

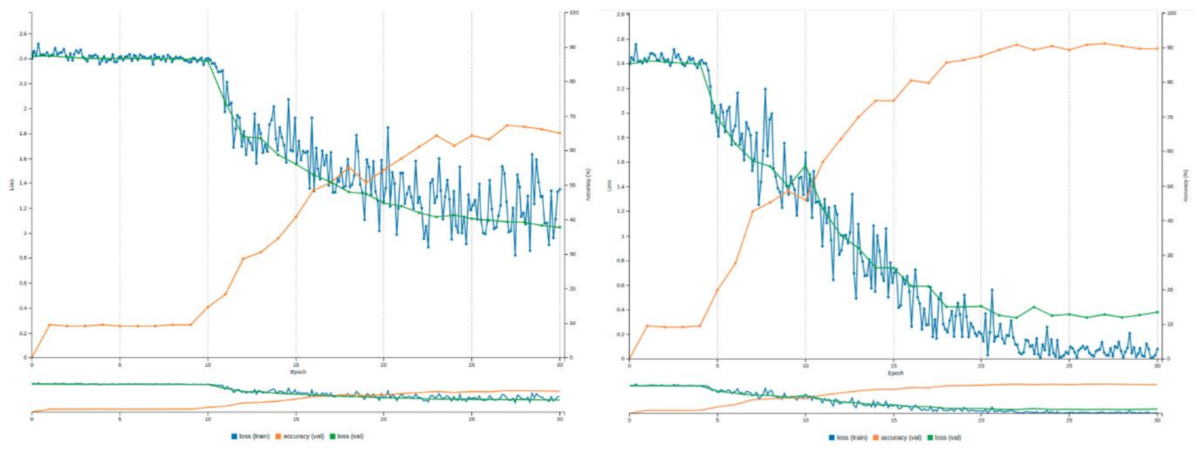

Figure 7: Comparison of learning rate policies exponential (left) and sigmoid (right) on VGG16 network

Now that the Sigmoid policy has been chosen, the next step is to see if changing the learning rate and epoch duration can improve the accuracy. Due to the Sigmoid function starting at the base learning rate, the value of 0.1 provides no learning, consistent with the AlexNet results.

Similarly, when the learning rate was set to 0.001 the network showed reduced learning, with a top-1 accuracy of only 32.76%. With VGG16 performing the best with a Sigmoid policy at a learning rate of 0.01 the effect of epochs can be studied. Figure 8 shows the VGG16 learning when the maximum epochs are set to 15 and 50. It is apparent that an epoch length is 15 is too short for a network of this depth as the loss curve is yet to plateau, resulting in a top-1 accuracy of 58.62%. The epoch length of 50 is useful for deciding the length for which there are diminishing returns, in this case the validation and training losses run in parallel after ~35 epochs and do not show signs of an overfitting arch. The increased training length marginally improves the top-1 accuracy at 91.38% versus the 84.48% seen at 30 epochs.

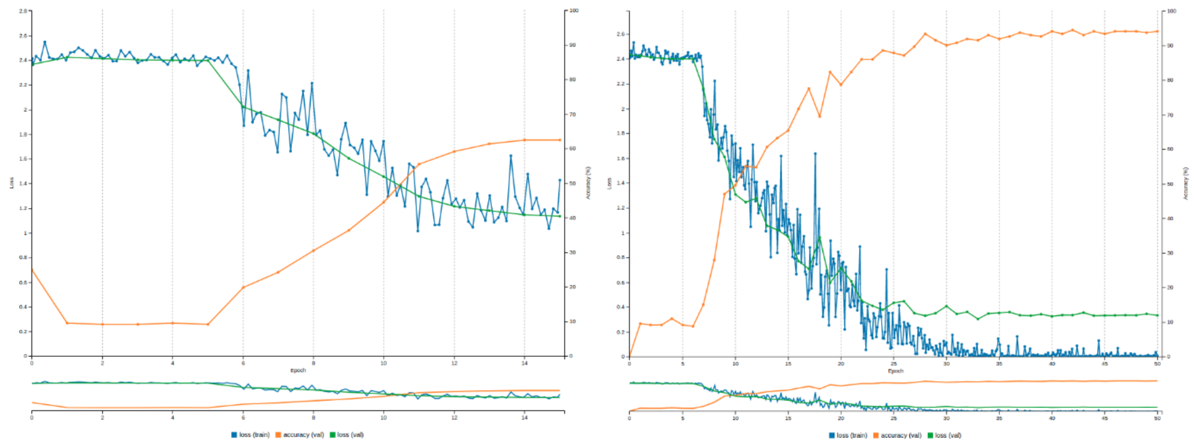

Figure 8: Comparison of epoch lengths 15 (left) and 50 (right) on the VGG16 Network

4 Summary

This report looked at the effectiveness of different CNN architectures and the steps which can be taken to improve learning. In Section 1, BN-AlexNet showed the highest accuracy, with VGG16 showing the lowest accuracy, making it ideal for Metaparameters tuning. Section 2 showed that artificially enlarging a dataset through image augmentation can improve accuracy, and for smaller datasets a larger training improves results. Section 3, demonstrated that by changing a Metaparameter, such as the learning rate policy to Sigmoid, can drastically increase the accuracy of VGG16 from 64.42% to 84.48%.

5 References

[1] 6482709, “EEEM063,” GitHub, 2021 April 13. [Online]. Available: https://github.com/WillJackson99/EEEM063.

[2] F. Bach and D. Blei, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” *Proceedings of Machine Learning Research,* vol. 37, pp. 448-456, 2015.

[3] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” in *International Conference on Learning Representations*, 2015.

[4] H. Vasa, “google-images-download,” 21 May 2019. [Online]. Available: https://github.com/hardikvasa/google-images-download. [Accessed 2 April 2021].

[5] A. Krizhevsky, I. Sutskever and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *Advances in Neural Information Processing Systems,* vol. 25, 2012.

[6] Keras Special Interest Group , “About Keras,” [Online]. Available: https://keras.io/about/. [Accessed 13 April 2021].