University of Surrey

Department of Electronic Engineering

MSc Computer Vision, Robotics and Machine Learning

EEEM063 Image Processing and Deep Learning Coursework

By

Kongkea Ouch

6664607

Submitted to

Professor John Collomosse

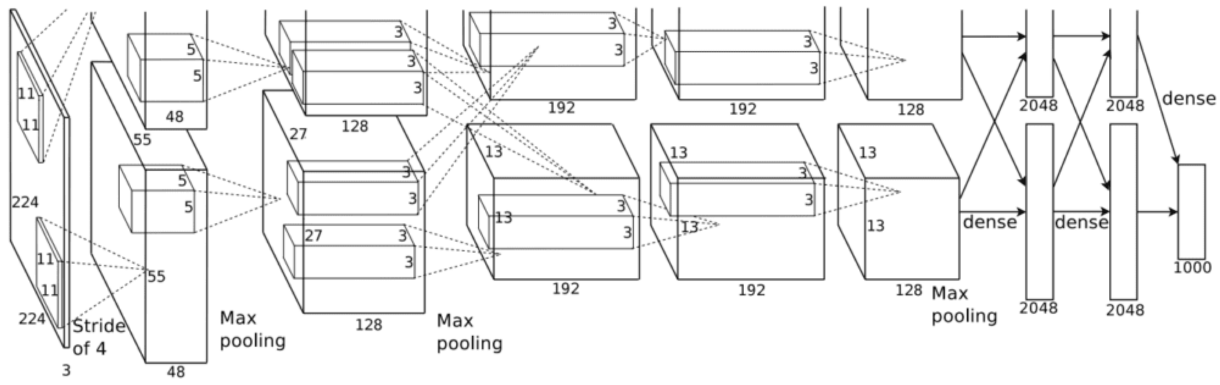
Abstract

In the last 15 years, advanced in the field of deep learning, particularly convolutional neural networks (CNNs), are proven to give outstanding results in several use cases, including image classification [5]. This assignment tackles the task of image classification using Standford car dataset [4] and concentrates on exploring various training strategies and hyperparameters on two modern CNN architectures: AlexNet [1] and GoogleNet [2]. An optimised variant of GoogleNet utilising batch normalization is also introduced for comparison. An evaluation of top-1 accuracy and top-5 accuracy is performed at each step of the fine-tuning process.

Introduction

Deep Learning is making great strides in addressing complex challenges, as computer vision solutions can be tailored to fit the needs of problems and their performances therefore win over the other established state-of-the-art solutions in many areas [5]. Deep learning can be understood as the application of artificial neural networks (ANNs) with a massive amount of layers, which contrasts how the traditional ANNs work by employing only a small number of layers [5]. The key strength of deep learning is being able to learn from raw unstructured data with no need for conventional feature extraction [5]. In deep learning, convolutional neural network (CNN) is a class of deep artificial neural network which is capable of performing complicated works such as image classification and object detection [5]. In this experiment, two major CNNs will be used which are AlexNet [1] and GoogleNet [2].

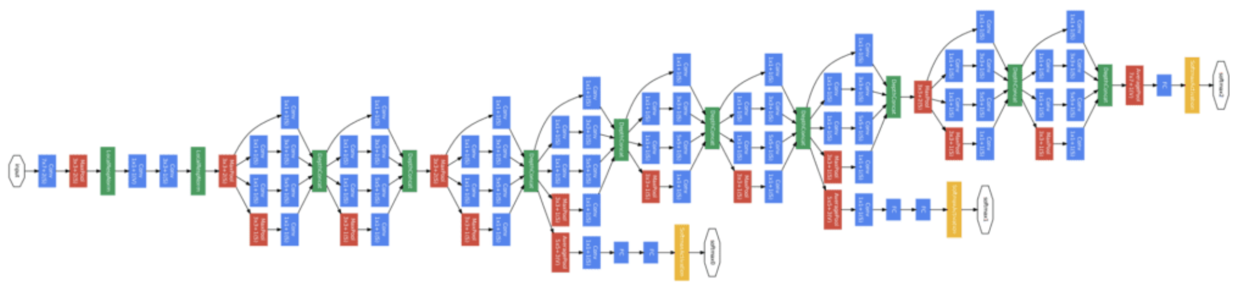
AlexNet [1], the winner of the 2012 ImageNet Large Scale Visual Recognishment (ILSVRC), is made of five convolutional layers for feature extraction and three fully connected layers for classification of the model. Its architecture is inspired and extended from LeNet-5 [1]:



AlexNet's key strengths of the architecture can be summarised as below [3, 1]:

- Use of the ReLU activation function after convolutional layers and softmax for the output layer.
- Use of Max Pooling instead of Average Pooling.
- Use of Dropout regularization between the fully connected layers.
- Pattern of convolutional layer fed directly to another convolutional layer.
- Use of Data Augmentation.

GoogleNet or Inception [2], the winner of the 2014 ImageNet Large Scale Visual Recognition (ILSVRC), just ‘goes deeper with convolutions’ and contains 27 layers. Below is GoogleNet architecture as taken from the original paper [2]:



As can be seen from above, there is a new layer called inception which denotes a breakthrough factor of GoogleNet and as explained by the author of GoogleNet [2, 8], ‘Inception Layer is a combination of all those layers (namely, 1×1 Convolutional layer, 3×3 Convolutional layer, 5×5 Convolutional layer) with their output filter banks concatenated into a single output vector forming the input of the next stage.’

GoogleNet’s key strengths of the architecture can be summarised as below [3, 2]:

- Development and repetition of the Inception module.
- Heavy use of the 1×1 convolution to reduce the number of channels.
- Use of error feedback at multiple points in the network.
- Development of very deep (22-layer) models.
- Use of global average pooling for the output of the model.

AlexNet and GoogleNet will be conducted using DIGIT system [6] on multiple experiments with varying training strategies and hyperparameters including:

- Dataset split
- Dataset augmentation

- Architecture
- Number of epochs
- Solver type or training optimiser,
- Learning rate
- Learning schedule
- Batch size
- Batch accumulation

Evaluation of training will be shown in accuracy of top one and top five classification.

Default configuration

Throughout all experiments, default configuration of training will be mentioned and it is based on DIGIT's default training configuration [6] which is proven as an overall safe and optimal combination for most training circumstances. The configuration used is as below:

- Learning rate: 0.01
- Learning rate policy: step down policy, 33% step size, 0.1 gamma
- Batch size: network defaults
- Batch accumulation: none
- Solver type: Stochastic gradient descent (SGD)

Abbreviation

- 'o': Standford original dataset.
- 'a': Augmented Stanford original dataset
- 'dataset_split': the dataset partitioning used. For example, 'o_80' or 'a_80' means 80% training, followed by the remaining potion equally divided into 10% validation and 10% training.
- 'lr': learning rate
- 'e': epoch
- 'google': standard GoogleNet network without any customization
- 'google_bn': standard GoogleNet batch normalization network
- 'alex': standard AlexNet network without any customization
- 'bs': batch size
- 'ba': batch accumulation
- 'bn': batch normalization
- 'rm': remove a particular layer

- 'top-1': Top-1 accuracy from network classification
- 'top-5': Top-5 accuracy from network classification

Dataset

Standford car dataset [4] will be used for all experiments except those involve augmentation. Augmented dataset is obtained by rotating each image in the original dataset by 45 degrees using MATLAB. As a result, the size of the augmented dataset is double of the original one.

Epoch

The experiment starts by training the original dataset on default configuration with both AlexNet and GoogleNet using 100 and 200 epochs. The dataset split uses here is selected using a general rule of thumb of 80/10/10 partition which means 80% for training set, 10% for validation set and 10% for testing set.

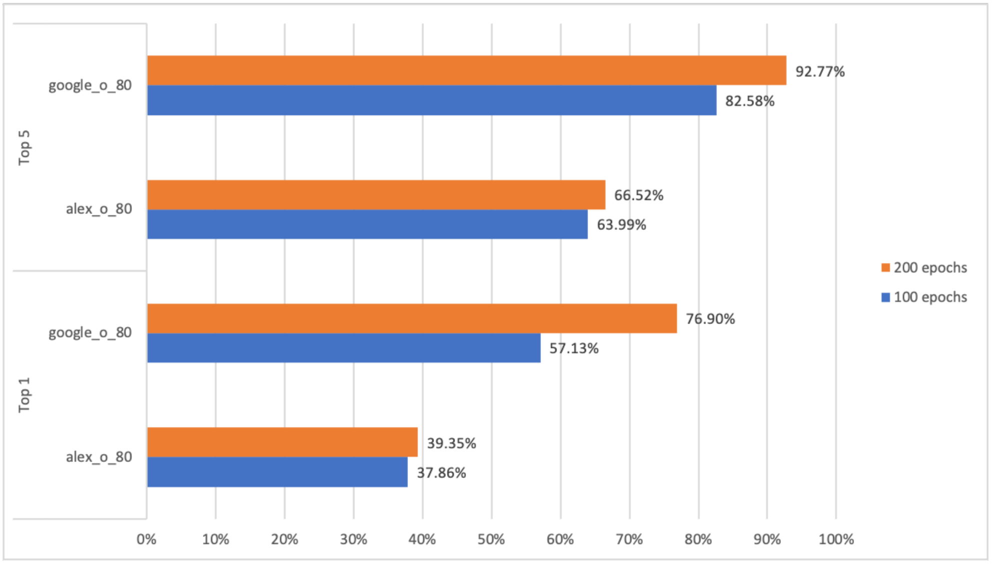


Figure 1: Epoch

From above results, GoogleNet significantly outperforms AlexNet in all scenarios due to its deep architecture being able to learn more feature details than AlexNet's shallow network. It is noticed that Alexnet's performance improves minimally with the change from 100 to 200 epochs, so AlexNet has converged good enough by 100 epochs considering both performance and efficiency. On the other hand, GoogleNet's gigantic network benefits greatly from longer training time and more exposure to feature finery, giving a superb 20% boost in top-1 accuracy and 10% in top-5 accuracy.

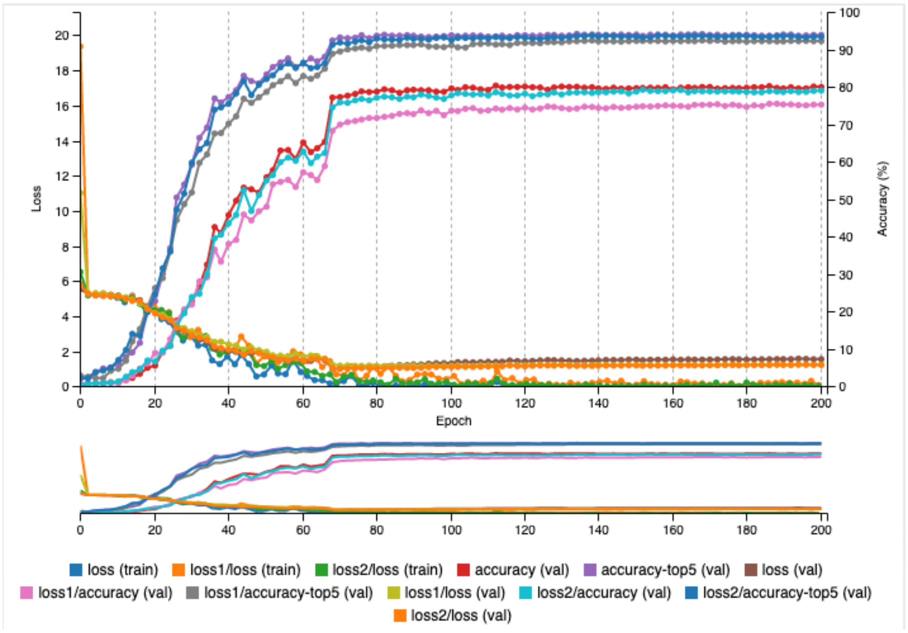


Figure 2: Google_o_80_e_200

Even 200 epochs yield great outcome for GoogleNet, it is noticed during the training (Figure 1) that the network already has converged by 100 epochs, indicating that the large performance gap between 100 epochs and 200 epochs is resulted from biased test set and does not fully represent the overall data distribution. Therefore, 100 epochs will be selected as the default choice for further trainings due to its optimal convergence performance for both networks, overall decent classification performance and lower training time which is ideal for performing multiple experiments in a limited resource environment.

Augmented Dataset

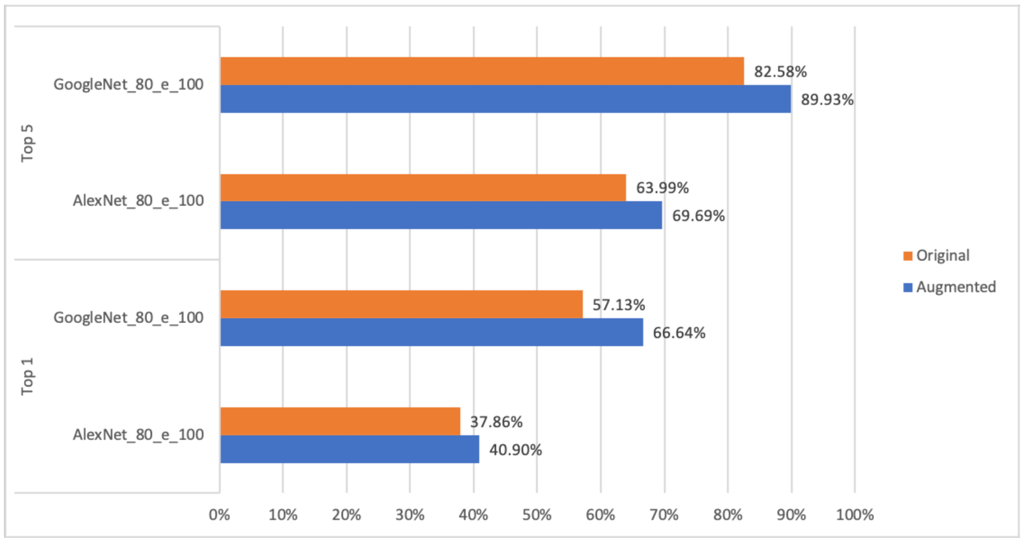


Figure 3: Augmented dataset

Both AlexNet and GoogleNet are trained on augmented dataset using default configuration and 100 epochs. As demonstrated in the results below, both networks improve performance from the augmentation as it allows them to see more training data in different rotation and generalizes better as a result. AlexNet improves slightly with the top-5 rising over 6% from the top-5 of the original network training on the same setting (Table 1). GoogleNet's performance increases considerably as top-1 rises around 10% and top-5 rises around 8%. Hence, it is proven that data augmentation helps boost the model's accuracy due to different exposure to the same data in a different setting. Though, to capture higher accuracy from data augmentation, the model needs to be trained on a bigger number of epochs due to the increased size of dataset.

Dataset Split

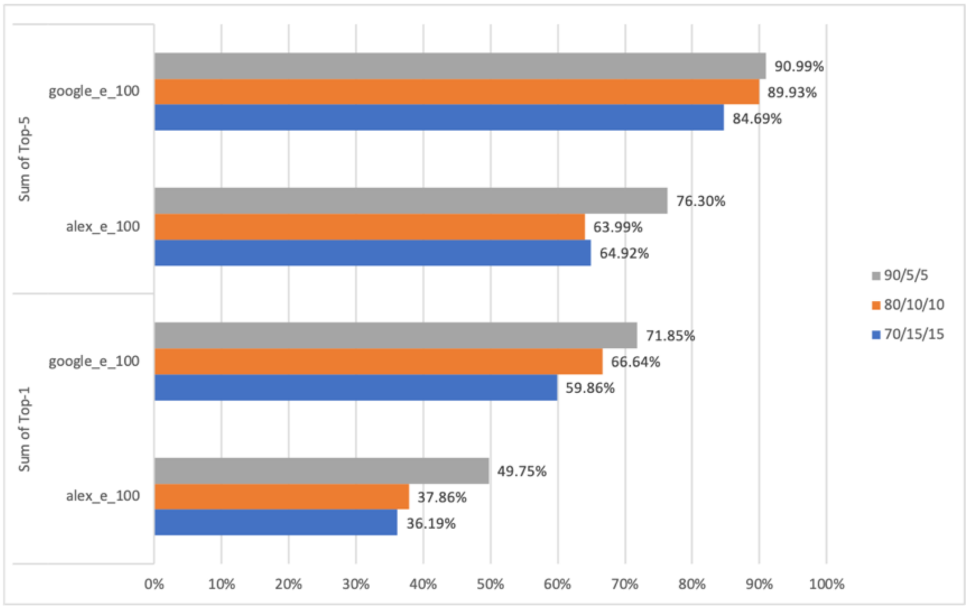


Figure 4: Dataset split

Three dataset splits are tested on default GoogleNet and AlexNet using 100 epochs. Based on Figure 4 results, 90/5/5 split gives the highest test accuracy because the network is exposed to more data and able to generalize better than the splits with less training data. However, this high performance might suffer from biased and insufficient test val/split that does not represent the whole dataset. Hence, an optimal choice for dataset split would be the 80/10/10 as it has more test/val split and can still capture a decent performance.

Learning Rate

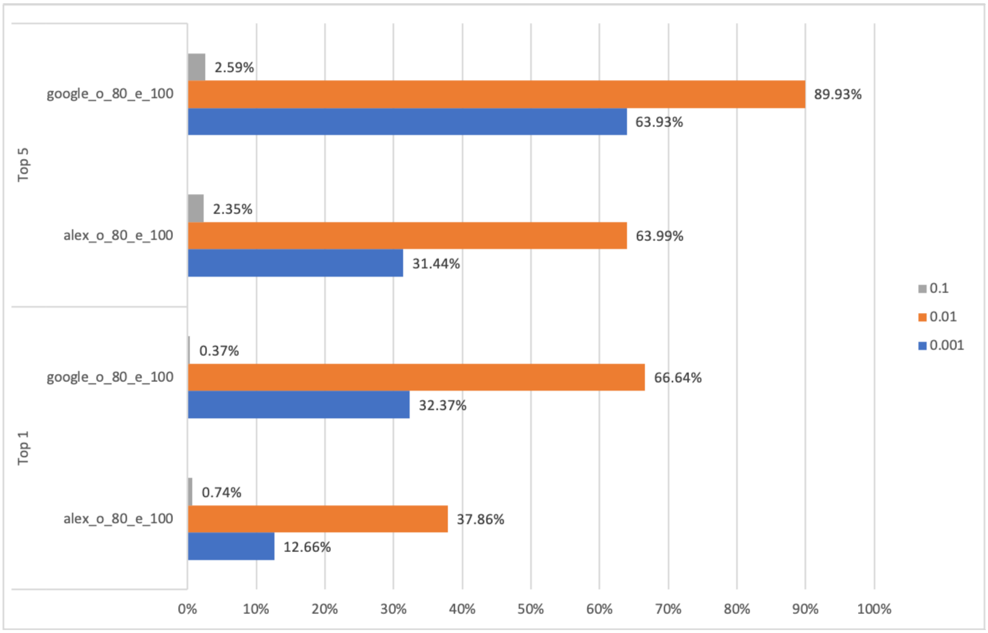


Figure 5: Learning rate

A grid search of learning rates is performed by changing order of magnitude from 0.1 to 0.001. From Figure 5, it is clear that the optimal learning rate that yields the best performance is 0.01. It offers a middle ground between training too fast resulting in high loss (0.1 learning rate) and training too slow resulting in no convergence (0.001) which are proven by above figure to result in poor performance of the system.

Learning Rate Policy

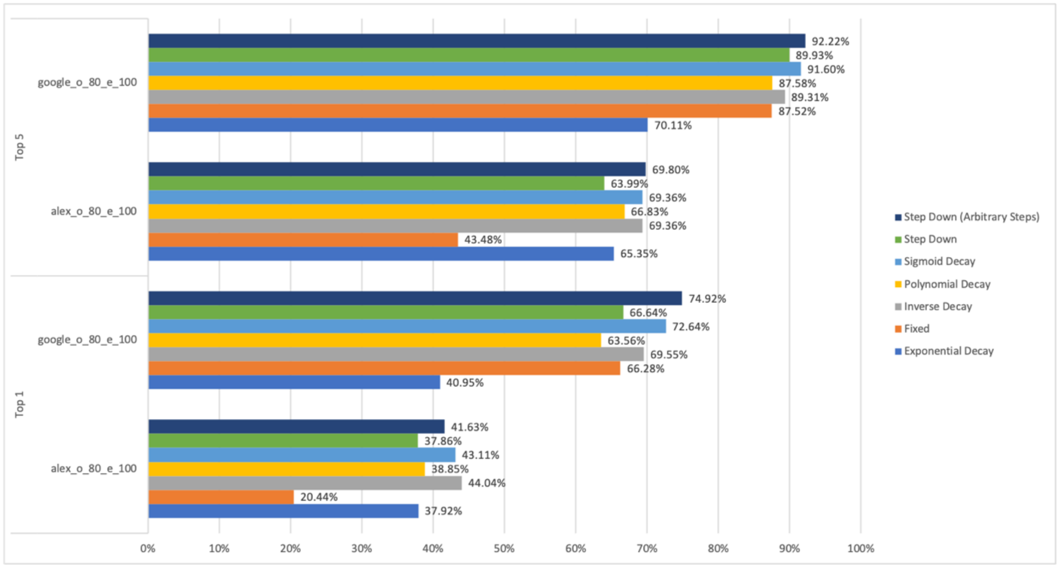


Figure 6: Learning rate policy

Default AlexNet and GoogleNet are trained with varying learning rate policies. Based on Figure 6, most learning rate policies go toes to toes with each other except fixed and exponential decay policy which noticeably gives the lowest performance. Besides exponential decay, other decay policies hold up pretty well against step down based policies. It appears that the policy of step down with arbitrary steps outperforms others policies as it can be customised to optimally target specific periods of the training that might benefit from decreased learning rate.

Solver Type

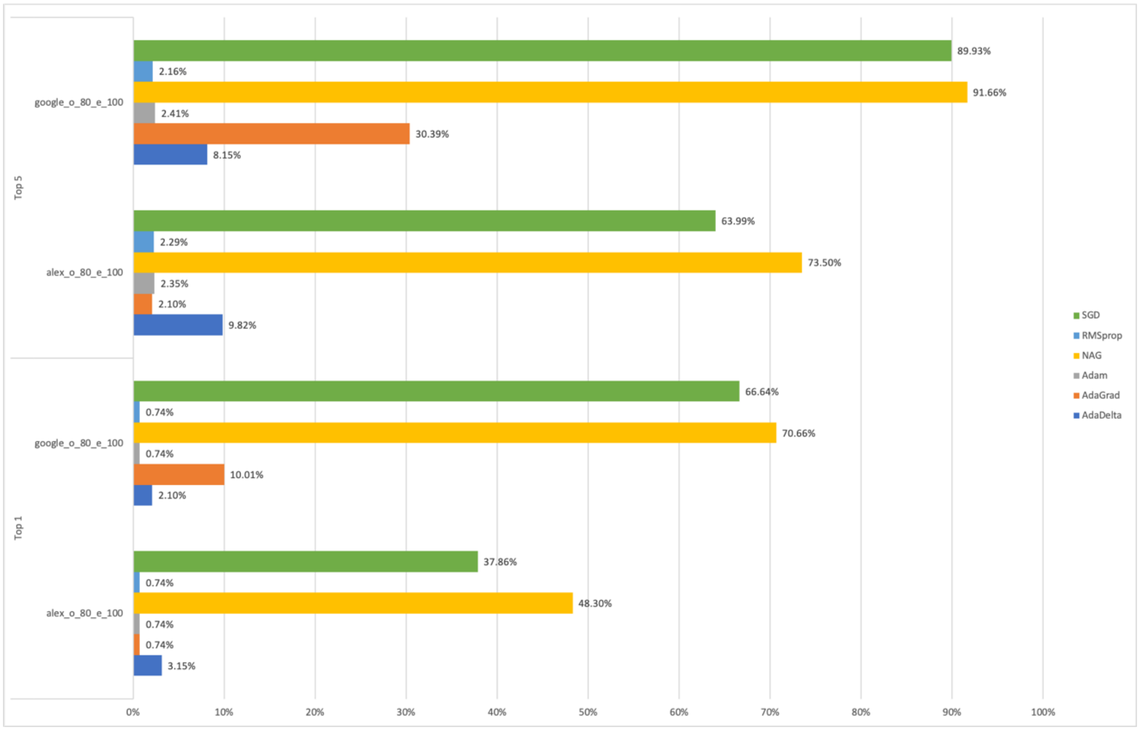


Figure 7: Solver type

Multiple optimisers are used to train default AlexNet and GoogleNet. As can be seen from Figure 7, the best performing optimisers are SGD and NAG. Though, it observed during the training that SGD results in the least training time out of all optimisers, hence it can be safely concluded that SGD is the optimal choice of optimiser overall. Other optimisers besides SGD and NAG seem to flat out fail and result in poor unusable models possibly due to the fact that these optimisers are not fully tuned with other hyperparameters to deal with the utlised CNNs yet.

AlexNet Layer

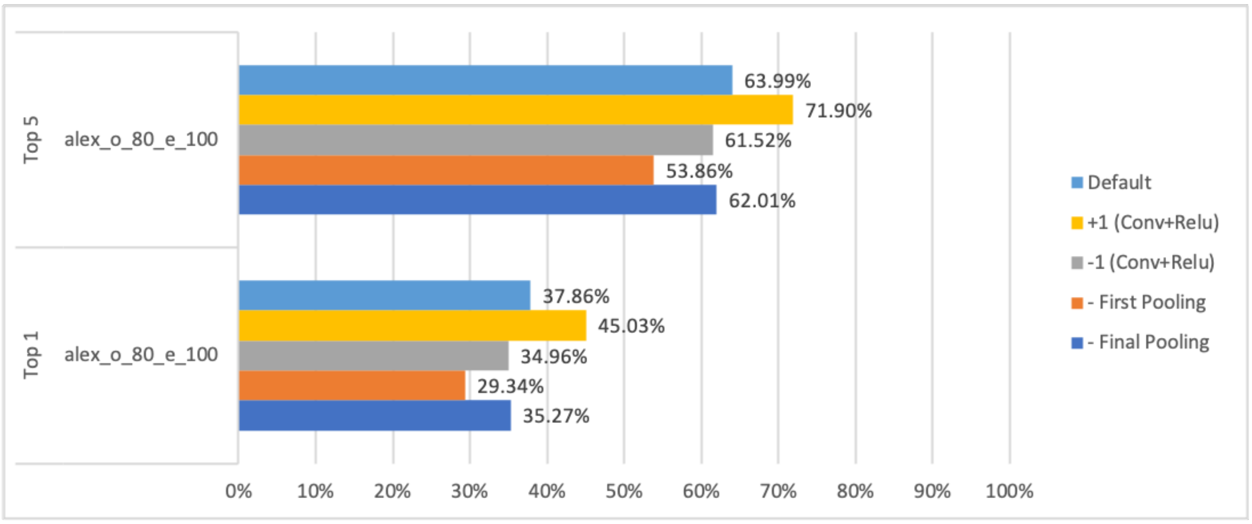


Figure 8: AlexNet layer

The default AlexNet is experimented on a wide range of layer customisations. As can be seen from Figure 8 results, the only customisation that improves upon the original model is the addition of one block of convolutional layer and RELU layer which allows the network to learn more feature details and generalise better. Removing any layer seems to have a drawback on the performance instead because it hinders the network's ability to learn complex information and take more time to train due to more workload resulting from reduced layer.

GoogLeNet Batch Normalization Network (GoogleNet BN)

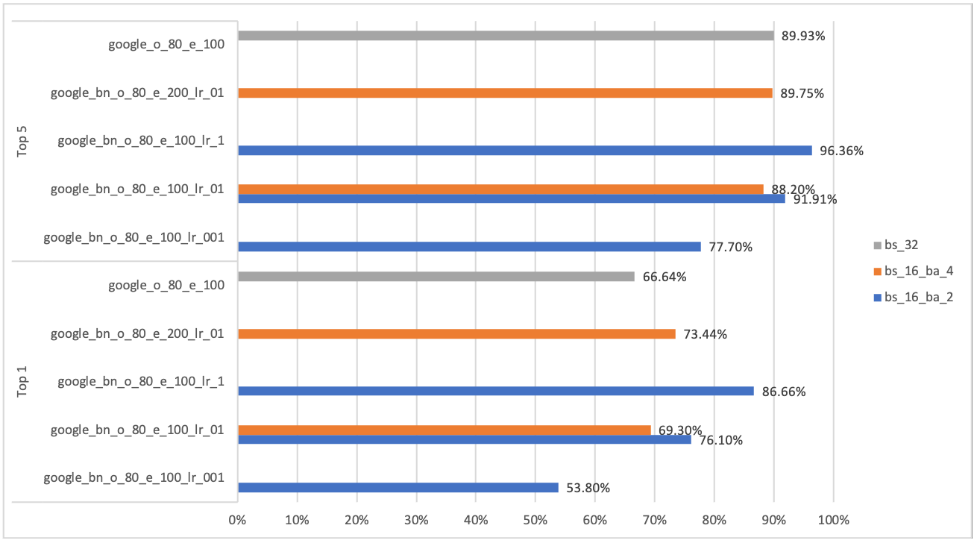


Figure 9: GoogleNet Batch Normalization

Last but not least, GoogleNet Batch Normalization Network (google_bn), a refined version of GoogleNet is experimented on a variety of batch sizes and learning rates. The difference to original GoogleNet is that this improved network incorporates batch normalization which allows the use of high learning rate which speeds up the training process and gives a better performance overall [7]. As demonstrated in Figure 9, in all scenarios that GoogleNet BN are trained with 0.1 and 0.01 learning rate, the batch normalization network beats the default GoogleNet by a significant margin. The default GoogleNet fails to train on 0.1 learning rate according to Figure 5 but GoogleNet BN can finds the best success with high learning rate to batch normalization.

However, GoogleNet BN requires more computational resource than GoogleNet, so a memory technique called batch accumulation is used to alleviate this problem by allowing the network to accumulate gradients over consecutive batches and update the model parameters later [5]. Typically, a batch size of 16 with batch accumulation of 2 is equivalent to batch size of 32 [5]. An attempt is made to increase the batch size to 64 as seen in Figure 9, but it degrades the model performance instead.

Conclusion

Deep learning has proven successful in several fields especially image classification. This assignment mainly focuses on training two major convolutional neural networks (AlexNet and GoogleNet), an advanced version of artificial neural network found in deep learning, using various training strategies and hyperparameters. Most successes are found with GoogleNet which is deeper and more complex than AlexNet, but an enhanced GoogleNet with batch normalization manages to outperform GoogleNet in all scenarios. In future works, more experiments will be done on later start-of-the-art CNNs such as VGG, ResNet and so on.

References

[1]. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* 25 (2012): 1097-1105.

[2]. Szegedy, Christian, et al. "Going deeper with convolutions." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

[3]. J. Brownlee, "Convolutional Neural Network Model Innovations for Image Classification," Machine Learning Mastery, 05-Jul-2019. [Online]. Available: https://machinelearningmastery.com/review-of-architectural-innovations-for-convolutional-neural-networks-for-image-classification/.

[4]. J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3D Object Representations for Fine-Grained Categorization," *2013 IEEE International Conference on Computer Vision Workshops*, 2013.

[5]. LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." *nature* 521.7553 (2015): 436-444.

[6]. "NVIDIA Deep Learning DIGITS," NVIDIA Deep Learning DIGITS Documentation. [Online]. Available: https://docs.nvidia.com/deeplearning/digits/index.html.

[7]. Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *International conference on machine learning*. PMLR, 2015.

[8]. F. Shaikh, "Deep Learning in the Trenches: Understanding Inception Network from Scratch," *Analytics Vidhya*, 18-Oct-2018.