Group 24: DSA4212 Assignment 1 Report

MAH SHIAN YEW BRENDAN, A0216349B
TEO MING JUN, A0216305R
SANDRA WONG JIA YI, A0222021H

Introduction

Given a computational budget of 120 seconds of compute time on a standard Google-Colab server with a standard GPU, the goal is to optimize the final validation accuracy of the model. To achieve this, different types of deep learning models can be used, such as logistic regression, neural networks, convolutional neural networks (CNNs), and others. However, starting with a randomized network and considering the constraint on computational time, a lightweight deep learning model is recommended.

Logistic regression is a simple model that can be used for binary classification tasks. However, it may not be suitable for complex image classification tasks.

Neural networks are powerful and flexible models that can learn complex representations from the input data. However, training a neural network from scratch can be time-consuming, and it may require a larger computational budget than what is available.

CNNs are widely used for image classification tasks and are well-suited for the provided dataset. They can learn hierarchical features from the input data and are generally lightweight compared to other neural network architectures. Moreover, there are several pre-trained CNN models, such as ResNet, DenseNet, and Inception, that can be fine-tuned on the provided dataset to achieve better performance.

Considering the constraint on computational time, a lightweight CNN architecture such as LeNet, AlexNet, VGG-16 with a reduced number of filters, MobileNet, or EfficientNet is recommended. These models have fewer parameters than larger architectures and can be trained within the given time frame. Additionally, data augmentation techniques such as flipping, rotating, and scaling images should be employed to increase the size of the training dataset and improve the model's generalization capability. Early stopping can also be used to stop the training process when the validation accuracy stops improving to avoid overfitting and save computation time.

Strategies Explored: Results and Considerations

1. CNN architectures:

In the initial phases, we explored the dataset and noted that there were 10 unique classes. Through this and as the dataset was an image dataset, we identified the need for a Convolutional Neural Network (CNN).

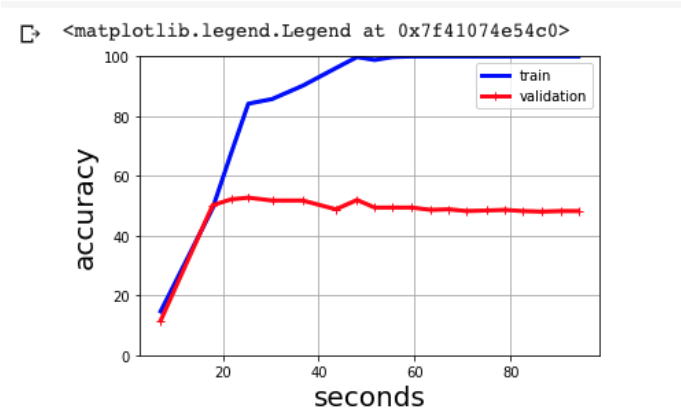
After deciding on the model that we thought was the best fit, we went on to look for different CNN architectures. The 2 architectures we shortlisted were LeNet and AlexNet CNN architectures. We then proceeded to carry out and experiment with both.

For both architectures, we underwent the same data preprocessing as provided in the starter kit. However, we also augmented the data in the pipeline through a Median Blur by blurring and randomly rotating the image by an angle between 0 and 90 degrees. After augmentation, the augmented images and their respective labels are added to the dataset. This was later shuffled again.

The group then proceeded to experiment with the different architectures. We initially created a CNN with the use of TensorFlow and Keras. Through experimentation, the best accuracy we managed to get was 59.9%. This was lower than the default model provided in the starter kit. We then proceeded to experiment with tweaking the model provided in the starter kit.

The first architecture to be tested was the LeNet architecture. We utilized a 9-layer LeNet CNN, with the convolution layers having a filter of 16, 32 and 64 pixels. The best result we received was a 70.0% accuracy.

AlexNet CNN was similar to LeNet CNN, but with 2 additional convolutional layers and an extra fully connected layer. The convolutional layers have filters of 16, 32, 64, 64, and 32 pixels. The best result we received was 59.3% accuracy. The sharp dip in the testing accuracy might likely be due to overfitting as it was noticed in the training phases of this model that it was 100% accurate with the training dataset.



We thus decided to go ahead with the LeNet CNN architecture which has proven to produce a higher accuracy.

2. Optimizer:

We also explored the use of the AdamW optimizer and how its performance compared to the Adam optimizer. Initially, we started off using the Adam optimizer to enhance our algorithm, which aided in providing adaptive learning rates and momentum to update our model's parameters during training. Overall, we found that while the Adam optimizer is effective in many applications, it is not without its limitations.

Some of the main issues we discovered with the Adam optimizer include its non-uniform scaling of the gradients and exponential moving average used which both lead to poor generalization performance and potential overfitting of the training data.

Through additional research on how to tackle these issues, we discovered the AdamW optimizer, which is a modified version of the original Adam optimizer. By introducing a weight decay term in the update rule, it was able to address the limitations we had previously run into and prevent overfitting. More specifically, the new weight decay term was able to penalise large weights, leading to a more robust and generalized model.

Furthermore, the AdamW optimizer also implements a "warmup" to gradually increase the learning rate during the initial stages of training. This helped in preventing the optimizer from getting stuck in poor local minima.

Overall, the AdamW optimizer showed increased effectiveness in a wide range of deep learning applications, and it is generally considered to be an improvement over the Adam optimizer. However, we noted that the choice of optimizer often depends on our specific application and the given dataset for the project, so it was important for us to experiment with different options to find the most suitable option for this assignment.

3. Batch Sizes

Lastly, we experimented with varying batch sizes. Batch size is a hyperparameter that affects the performance of machine learning models. In particular, larger batch sizes can result in faster convergence but may lead to overfitting, while smaller batch sizes deal better with overfitting but succumb to a slower convergence. In our pursuit to find an optimal batch size that would suit our dataset, we decided on 512 in each iteration of training as well as 32 in the initial propagation stage, which we felt was able to best balance convergence speed and generalization performance.

Final Deep Learning Pipeline

After considering the few deep learning pipelines above, our group ultimately decided to go with LeNet CNN architecture, the AdamW optimizer and a batch size of 256 for the reasons outlined above.

Augmented Data

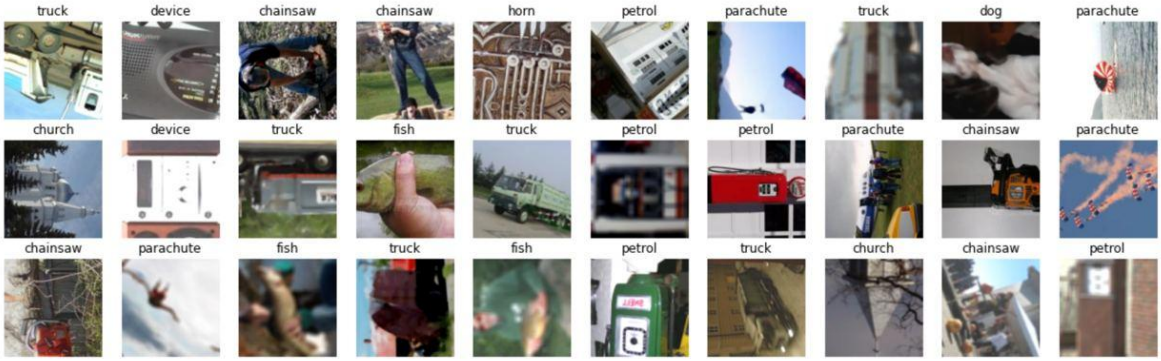
To improve the test accuracy, our group used Data Augmentation to increase the size and diversity of the training dataset. By adding new samples to the training dataset, the model becomes more exposed to variations in the data, which can lead to better learning and higher accuracy on the test data. Increasing the diversity of the training dataset will increase the robustness and generalization of the model, training it to recognize variation and diversity in the data, hence making it less prone to overfitting whilst being better able to generalize when dealing with new unseen data.

Data augmentation also helped create more training data in terms of quantity and variation by generating new samples from existing data. This led to an overall improvement in the performance of the model.

Our group used Albumentations, an open-source Python library for data augmentation.

Figure A: Random sample of the dataset before Augumentation

Figure B: Random sample of the dataset after Augumentation



Observations: The effect of augmentation techniques (*random cropping, flipping, rotation, colour changes and blurring*) can be observed in the images in **Figure B**

It is an open-source Python library for image augmentation, which is commonly used to increase the diversity of data used for training deep learning models. It offers users a wide range of augmentation techniques, that include but are not limited to random cropping, flipping, rotation, colour changes and blurring.

Albumentations offers a quick and effective solution for users looking to augment large image datasets due to its optimized speed and support for various image formats, such as numpy arrays, PIL images, and OpenCV images. Additionally, its user-friendly API allows for easy application of augmentation techniques with minimal code and is compatible with popular deep learning frameworks like TensorFlow, PyTorch, and Keras.

Moreover, besides offering pre-defined augmentation methods, Albumentations also allows users to create their own customized augmentation pipeline by combining different techniques and adjusting their parameters as they see fit. This level of flexibility enables users to tailor their augmentation pipeline to specific datasets and use cases.

Conclusion

In conclusion, given the provided datasets and computational constraints, a lightweight CNN architecture such as LeNet, AlexNet, VGG-16 with a reduced number of filters, MobileNet, or EfficientNet is recommended. Data augmentation techniques and early stopping should also be used to improve the model's performance and save computational time.

Works Cited

7.6. Convolutional Neural Networks (LeNet) — Dive into Deep Learning 1.0.0-Beta0 Documentation. https://d2l.ai/chapter_convolutional-neural-networks/lenet.html. Accessed 16 Mar. 2023.

Alake, Richmond. “Understanding and Implementing Lenet-5 CNN Architecture (Deep Learning).” *Towards Data Science*, 5 Mar. 2022, https://towardsdatascience.com/understanding-and-implementing-lenet-5-cnn-architecture-deep-learning-a2d531ebc342.

Fast.Ai - AdamW and Super-Convergence Is Now the Fastest Way to Train Neural Nets. https://www.fast.ai/posts/2018-07-02-adam-weight-decay.html. Accessed 16 Mar. 2023.

Hargurjeet. “7 Best Techniques To Improve The Accuracy of CNN W/O Overfitting.” *MLearning.Ai*, 27 May 2021, https://medium.com/mlearning-ai/7-best-techniques-to-improve-the-accuracy-of-cnn-w-o-overfitting-6db06467182f.

Mwiti, Derrick. “Image Augmentations with Albumentations.” *Heartbeat*, 11 Aug. 2022, https://heartbeat.comet.ml/image-augmentations-with-albumentations-c1ca8fc78db7.

Saxena, Shipra. “Alexnet Architecture.” *Analytics Vidhya*, 19 Mar. 2021, https://www.analyticsvidhya.com/blog/2021/03/introduction-to-the-architecture-of-alexnet/.