GTSRB Dataset:
Transfer Learning VGG16 for road signs

Senan Stanley

Dec, 2022

1 Introduction

This paper will attempt to implement transfer learning by retraining VGG16 network to recognise traffic signs. This will be completed by freezing many of the existing layers of the VGG16 network, trained on ImageNet, and adding further classification layers onto the end of the VGG16 network. Then data from the GTSRB dataset will be processed and fed into the augmented VGG16 network for training. The network will then be evaluated on never before seen test images to gauge the performance of the model. This will be iterated to produce the optimal model for classifying traffic sign data.

2 State of the Art

Transfer learning is a machine learning technique where a model trained for a primary task is retrained and repurposed for a secondary task [Weiss et al., 2016]. Transfer learning can be useful for tasks with limited secondary datasets. This method of learning is also useful when the training for the secondary task is computation constrained, as fewer layers can need to be trained.

To accomplish transfer learning, a trained model can have its weights frozen (made immutable) and layers added to the end of the network, which are fine-tuned to the task. A trained model's weights and biases can also be used as the initial state for training to the secondary task. A third way to execute transfer learning is selectively freezing layers of the primary model, and training those unfrozen layers to the secondary task [Cao et al., 2010].

The VGG16 model was created by the University of Oxford Visual Geometry Group (VGG) and consists of a 16 layer neural network model trained on the ImageNet dataset [Simonyan and Zisserman, 2015]. VGG16 is a deep convolutional neural network created for image classification. The original VGG16 network takes a 3 channel (RGB) image of 244 pixels square as input, for the ImageNet dataset it was trained to classify the input into one of 1000 classes [Russakovsky et al., 2014]. It is commonly used as a baseline for transfer learning, for image recognition problems. The model is relatively simple, only consisting of 16 layers, but can be comparatively computationally expensive to run, due to its size of $\approx 138,000,000$ parameters [Simonyan and Zisserman, 2015].

The GTSRB (German Traffic Sign Recognition Benchmark) dataset is a collection of images of traffic signs created by the German Aerospace Centre, to be used in the training of machine learning models [Stallkamp et al., 2011]. The dataset contains $\geq 50,000$ images of German road signs (stop, speed limit, turn here). Though, there is not an equal number of each of the 43 different signs present in the dataset.

The VGG16 model lends itself to being used for transfer learning due to its simplicity, and it's success in image classification. There are several projects online attempting to use transfer learning of the VGG16 model to classify the GTSRB dataset [Persson, 2018, UMAR, 2021].

3 Implementation

The internal Kaggle notebook system was used to run this machine learning project. This particular notebook is publicly available online [Stanley, 2022]. This environment mimics a Jupyter Notebook cell-like scripting Python experience. The GTSRB dataset was linked into Kaggle and processing could begin [Mykola, 2018].

Once the dataset is imported, it must be preprocessed before any machine learning can be accomplished. The preprocessing accomplished was to normalise the image arrays (scale the values to between 0-1, rather than 0-255), resize the images to be square and shuffle the training image order, to encourage generalisation in the model. After this processing on all images, the training data was split into 70% training images, 30% validation images.

The training images were imbalanced in the amount of each of the 43 categories present, as can be seen in Figure 1. The number of images in a given category could vary by a factor of 10. Training a model on imbalanced data can cause the model to ignore minority data in favour of the majority. To attenuate this issue, the number of classes were counted, and ratios were created to reflect the imbalance of each class These class based weights informed the model training of this imbalance [Tensorflow, 2022].

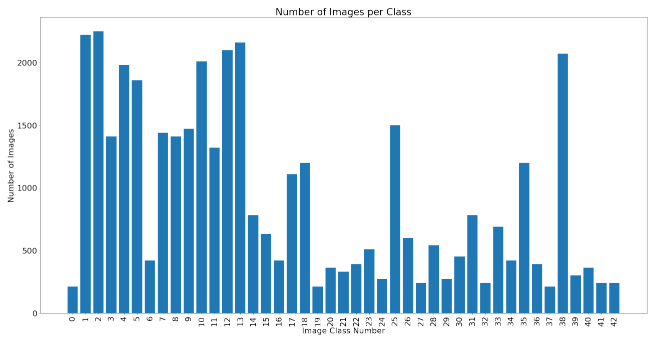


Figure 1: Graphical representation of the class imbalance of the GTSRB dataset.

The VGG16 model was imported with its original input of a 3 channel square 244 pixel image. The final classification layer was removed from VGG16. 4 new layers were added at the end of the VGG16 model to learn the new dataset. These consisted of a flattening layer (to compress the data into one dimension) and three dense layers of size 1024, 512 & 43. The last layer is the softmax activated classification layer.

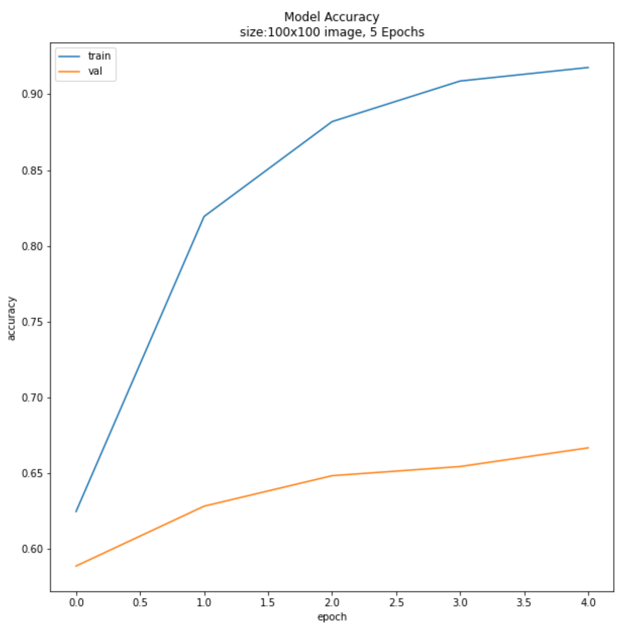
The model was run over varying epochs, at a batch size of 128, using the Adam optimiser and with classes balanced as described above.

4 Results & Discussion

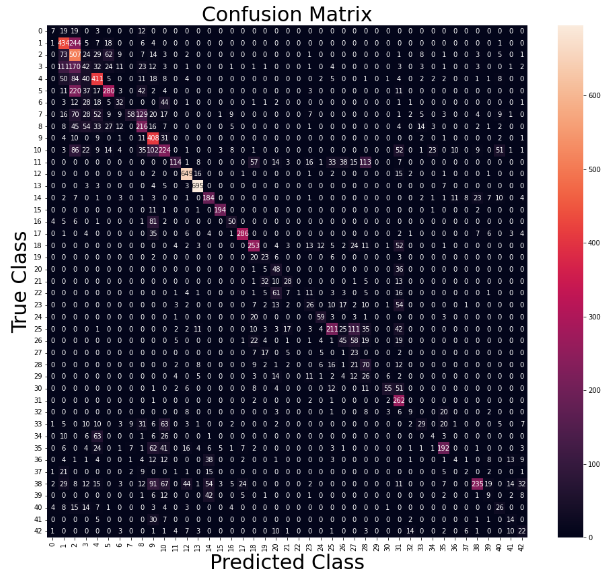
The result of the implementation was underwhelming, as can be seen in the best run Figure 2. Many iterations were created: varying image size, epochs, class weighting, layer complexity, optimiser and many other methods of tuning the model. Models were left running for over a cumulative 100 hours during this process. And the result was a model barely able to achieve 50% accuracy on the test data.

This was far below the accuracy achieved by others attempting the same task, which were usually in the 90th percentile [Persson, 2018, UMAR, 2021]. Many hours were spent varying the importing and fitting of the data in order to locate the code causing the drop in performance. But the cause was not found.

There was a suspicion during this process that the input data must be the problem. As, no level of complexity or tuning on the part of the model could achieve above average accuracy. Finally, towards the end of this project there was an epiphany that not all input images were of equal dimensions, thus they were being warped when resized to a square image.



(a) Best model training and validation accuracy.

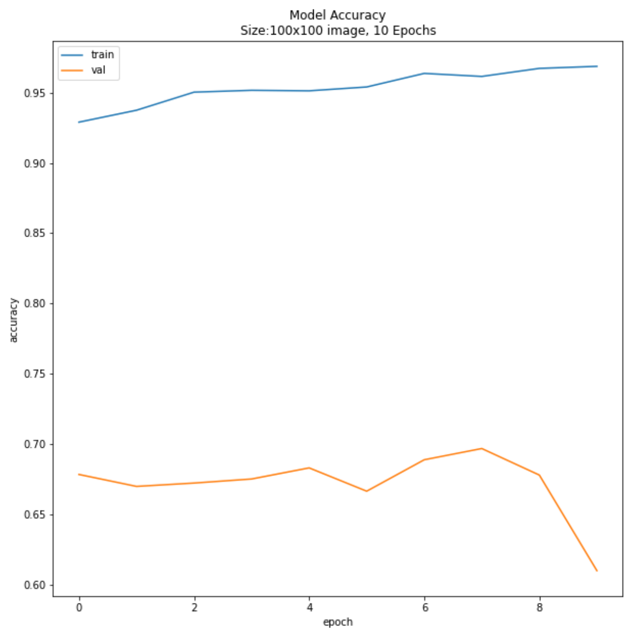


(b) Confusion matrix of best model.

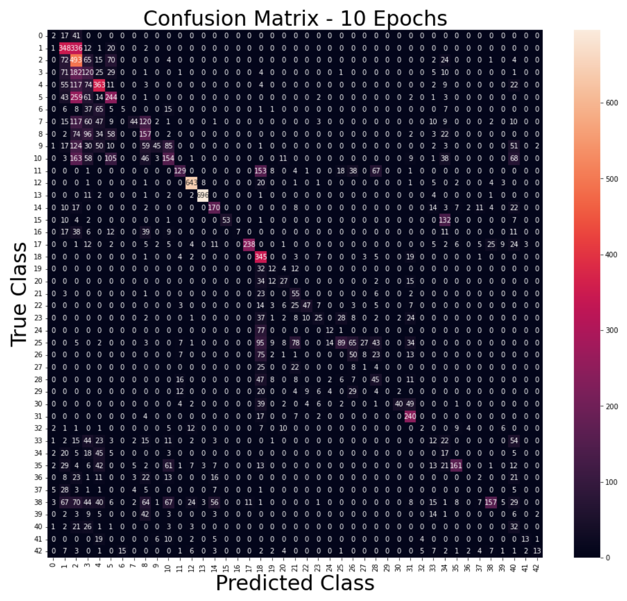
Figure 2: Most successful model, achieving a test accuracy of 51%.

As can be seen in the confusion matrix in Figure 2(b) the test results were scattered in an ordered fashion. With clusters of classes, such as classes 1-5, being highly associated and often mistaken with eachother. The model overestimated the prevalence of some classes, such as class 31, creating many false positive prediction of it. While similarly underpredicting the prevalence of classes, such as class 11.

Interesting, the most successful model was run only over 5 epochs, as can be seen in Figure 2(a). When this model was expanded to run over 10 epochs, the test accuracy worsened, seen in Figure 3.



(a) 10 epoch version: training and validation accuracy.



(b) 10 epoch version: Confusion matrix.

Figure 3: 10 epoch version, achieving a test accuracy of 42%.

The confusion matrices of Figure 2(b) and Figure 3(b) are almost identical upon first inspection. Perhaps the further training epochs reinforced issues with this model. The validation accuracy also suffers in Figure 2(a) and when compared to the 5 epoch version of Figure 3(a).

5 Conclusions

If this project was attempted again, the data must be preprocessed better. It was only late into the project when it was discovered that the dataset images were different dimensions and not all square images. Therefore, when the images were resized the signs were warped, which was reflected in the poor results.

Albeit the VGG16 model is simple, it is by no means a small model. The size of the model, when using a full-sized 244×244 image, caused the program to crash many times while training. This project was transported between three platforms in an attempt to lower training time (from 4+ hour wait timers per layer) and limit crashes, Local notebooks, Google Colab finally Kaggle. This caused many headaches of logistics as datasets and results needed to be migrated too. Even when GPU accelerated, 32 GB RAM systems were rented and implemented, the model often crashed these systems, by over-allocating RAM. To run a full-sized image version of VGG16, a better computing solution must be acquired.

References

[Cao et al., 2010] Cao, B., Pan, S. J., Zhang, Y., Yeung, D.-Y., and Yang, Q. (2010). Adaptive transfer learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 24(1):407–412.

[Mykola, 2018] Mykola (2018). Gtsrb - german traffic sign recognition benchmark.

[Persson, 2018] Persson, S. (2018). *Application of the German Traffic Sign Recognition Benchmark on the VGG16 network using*. PhD thesis.

[Russakovsky et al., 2014] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2014). Imagenet large scale visual recognition challenge.

[Simonyan and Zisserman, 2015] Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition.

[Stallkamp et al., 2011] Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. (2011). The german traffic sign recognition benchmark: A multi-class classification competition. In *The 2011 International Joint Conference on Neural Networks*, pages 1453–1460.

[Stanley, 2022] Stanley, S. (2022). Kaggle notebook `https://www.kaggle.com/sensty/ml-transfer-learning-on-gtsrb`.

[Tensorflow, 2022] Tensorflow (2022). Classification on imbalanced data nbsp;: nbsp; tensorflow core.

[UMAR, 2021] UMAR, A. (2021). Vgg16 transfer learning. *Kaggle.com*.

[Weiss et al., 2016] Weiss, K., Khoshgoftaar, T. M., and Wang, D. (2016). A survey of transfer learning - journal of big data.