Transfer learning with VGG16

Ronan McCormack S00144576

Institute of Technology Sligo

Abstract

Transfer learning has many benefits in machine learning. It's application with Convolutional Neural networks can increase time spent training convolutional neural networks and allow a better feature extraction process. Transfer learning is examined using the VGG16 convolutional neural network to predict data from the German road traffic sign dataset.

Keywords: Classification, Machine Learning, Convolutional Neural Network

1. Introduction

Convolutional Neural Network (CNN) have become a large area of research in the past number of years. Evolutions in technology have provided the resources to apply CNNs to classification research in a range of fields such as image and pattern recognition. The application of CNNs allow researchers to solve complex tasks which was not possible with classic Artificial Neural Networks. Image classification and CNNs can abstract features at different layers from input images, e.g., face recognition (Albawi, et al., 2018).

2. Methodology

Transfer learning is the application of pre-trained models to help speed up convergence of CNN training. Transfer learning also allows the use of different datasets with the initial dataset used to train the model (Qassim, et al., 2018).

In this project, transfer learning will be applied to a VGG16 convolutional neural network which is pre-trained using the ImageNet database (Krizhevsky, et al., n.d.), which comprises of 1000 categorical images. The theory behind using transfer learning comes from using an already pre-trained CNN and using its layers to extract low level features such as spatial, rotation, edges, and shapes (Qin, et al., 2018).

2.1 Architecture of a Convolutional Neural Network

The VGG16 CNN is 16 layers deep and trained on the ImageNet database which consists of over a million images. The CNN can identify 1000 categorical objects such as animals, keyboards, and pencils.

The convolution layer, marked red in figure 1, places a matrix called a kernel over the input matrix to create a feature map for the next layer. Some refer to this method as many layers sliding across the input image. All kernel sizes in the VGG16 CNN are 3x3 (Istiyadi Swasono, et al., 2019).

The activation layer, marked green in figure 1, is a non-linear activation layer with the one being used by VGG16 is Rectified Linear Unit (ReLU). ReLU is a function which returns the output if the input is positive, otherwise it will output zero (Istiyadi Swasono, et al., 2019).

The pooling layer, marked yellow in figure 1, reduces the size of the input layer and helps prevent over fitting. It also helps ensure small changes to the input and its changes are not largely represented. Kernel sizes in the pooling layer are 2x2 (Istiyadi Swasono, et al., 2019).

The CNN is then completed by the SoftMax layer, marked dark green in figure 1, which highlights the most dominant class as predicted by the CNN. It is usually placed at the output layer of a CNN (Avci, et al., 2019).

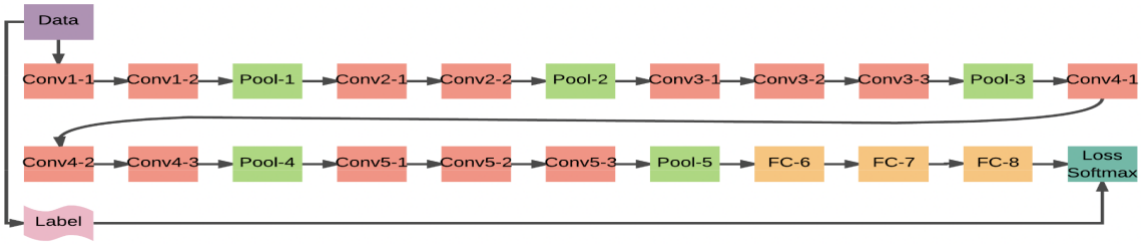

Figure 1: VGG16 Convolutional Neural Network

2.2 Additional Layers

As this project implements Transfer Learning, the addition of output layers is a necessity to complete the CNN. The following layers were added to the VGG16 CNN model:

- Flatten Layer: The pooled feature map is flattened into a column vector of size 512.
- Dense Layer: The flattened layer is fully connected to the Dense layer and uses the 'ReLU' activation. The size specified for this layer is 256.
- SoftMax: This layer use predicts which of the 43 classes the image belongs to.

2.3 Dataset

The dataset is a 43-class, single-image set of images, comprised of German street signs. The dataset itself contains over 50,000 images which are not evenly distributed across the 43 classes. From some basic pre-training analysis, we can see the distribution of the 43 classes in figure 2.

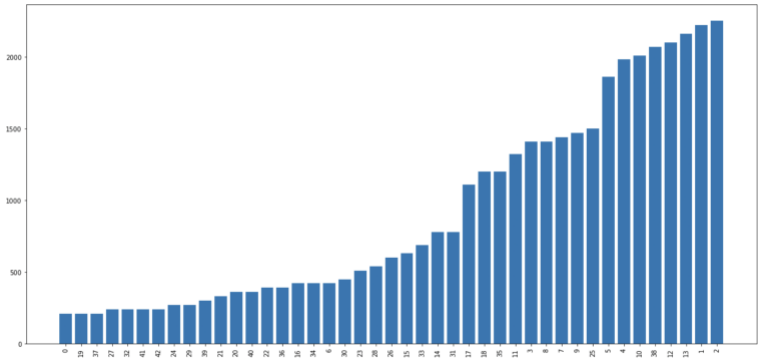

Figure 2: Distribution of images in the dataset

For this project, the dataset has been altered to evenly distribute the images amongst all classes. The dataset was cleaned to ensure all classes length matched the length of the smallest class which had a count of 210 images. The images chosen for each class were randomly selected to ensure further non-bias in the dataset. This new dataset was used to train the CNN for this project with the intention to minimize bias and improve training time.

3. Experiments & Research

Within Google Colab, TensorFlow was the open-source artificial intelligence library used to create the CNN model. Keras was used to provide the Python interface for the CNN, as it acts as an interface for the TensorFlow library.

The images from the dataset were loaded in, at a target size of 224x224, with a color mode of RGB, and batched in sizes of 32. The images were classified by their category of folder they were contained in, and the images were then shuffled to ensure fair bias. The optimizer used in the project was Adam. Adam is a stochastic gradient descent optimizer which according to many is the best adaptive optimizers for most use cases. The loss function applied in this CNN is Categorical Crossentropy. This was chosen because the images were classified using the category. The metric used is Accuracy which calculates how often the prediction equals the label. When fitting the model, the initial steps per epoch was set to default. Within the model's defaults this is calculated by taking the number of images in the training data and dividing it by the batch size. The epoch was set to 100 which according to the graph below was around the time we plateau in terms of accuracy.

The results of training the model are shown in Figure 4. In the beginning the loss was quite high but as each epoch was run, the loss started to converge towards zero while the accuracy converged towards 1, ending up around an accuracy of 98%. When the validation set was used to evaluate the CNN, the results were a validation loss of 0.19 and a validation accuracy of 93%.

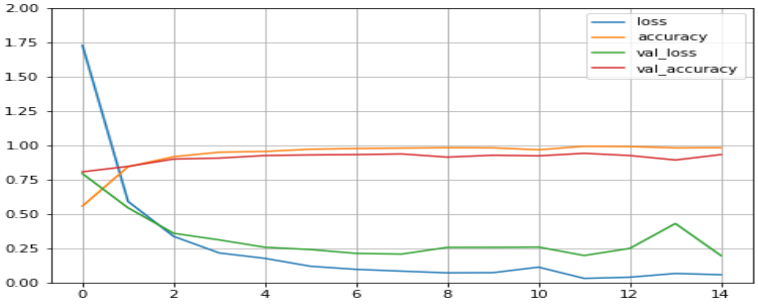

Figure 3: Training & Validation data accuracy and loss.

Using Sklearn's accuracy score function, the correct classes of the test images were compared to the predicted classes by the CNN. The accuracy score of the CNN was 74%.

Figure 4: Predicted classes.

4. Conclusion

In this project, transfer learning was the technique used alongside the VGG16 neural network. The aim was for the CNN model to accurately predict German street signs based on 43 class-images. The CNN model performed exceptionally on the training data and validation data but failed to replicate the same success on the test data (74% accuracy). Based on this, the CNN model may be over fit to the training data. The lack of even distribution amongst the image classes can contribute to over fitting to the training data, with the validation data giving the same indication.

Bibliography

- Albawi, S., Mohammed, T. A. & Al-Zawi, S., 2018. Understanding of a convolutional neural network. *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017,* pp. 1-6.
- Avci, M., Ozyildirim, B. & Sarıgül, M., 2019. Differential convolutional neural network. *Neural Networks 116,* pp. 279-287.
- Istiyadi Swasono, D., Tjandrasa, H. & Fathicah, C., 2019. Classification of Tobacco Leaf Pests Using VGG16 Transfer Learning. *12th International Conference on Information & Communication Technology and System (ICTS) ,* pp. 176-181.
- Krizhevsky, A., Sutskever, I. & Hinton, G. E., n.d. ImageNet Classification with Deep Convolutional Neural Networks.
- Qassim, H., Verma, A. & Feinzimer, D., 2018. Compressed residual-VGG16 CNN model for big data places image recognition. *2018 IEEE 8th Annual Computing and Communication Workshop and Conference, CCWC 2018,* pp. 169-175.
- Qin, X., Wu, Y., Pan, Y. & Yuan, C., 2018. Convolution Neural Network based Transfer Learning for Classification of Flowers. *IEEE 3rd International Conference on Signal and Image Processing,* pp. 562-566.