Image Classification with Convolutional Neural Networks

Charles Liebenberg

Monday 27th July 2020

Executive Summary

Cifar10 is a dataset containing images that are categorised into 10 different categories. The purpose of this report is to detail the creation of a Convolutional Neural Network trained to classify these images. The final model after optimisation consisted of 500 thousand trainable parameters, and achieved a 90% accuracy on external data.

Data Collection and Preprocessing

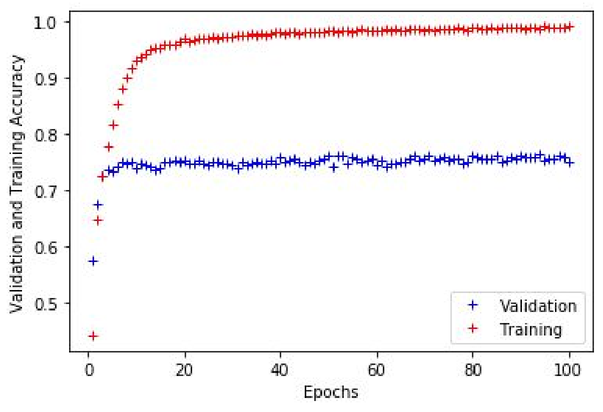
The data is pre-built into the keras library. It was retrieved and then subsequently converted into float data. Once this data was downloaded, it was normalised. This was done through subtracting the mean and dividing by the standard deviation. This normalisation was performed to ensure the gradient descent was not dominated by certain images.

Training, testing and validation sets.

The data was pre-built with training and test datasets. The training and testing data-split was 5:1. The data was further split up after the initial train:test split into train, test and validation data. The validation set was taken as 20% of the training set. This process was done at the start of the analysis as data augmentation is one of the techniques used, and validation data should not be augmented. Therefore, this split allowed for the data to be kept separate and not impede later analysis.

Neural Network Building (Topology, Evaluation Criteria and Training)

An initial basic model was built for a starting point. It was trained purposefully to overfit, and did so after about 5 epochs (see diagram). It was observed that there were no late minima and as such the model was trimmed in epochs so that a systematic analysis could be performed.



The model was optimised using the adam optimisation function due to its momentum attributes. The loss function was sparse categorical cross entropy as the data has not been one-hot encoded. Therefore sparse categorical cross entropy will allow for faster evaluation due to use of a single integer rather than a vector. The metrics used for evaluating the model was accuracy due to the problem's classification nature. To allow for ease of analysis, the same function created previously was used that allowed for automatic plotting of accuracy and validation accuracy against epoch count and both validation accuracy and loss against epoch count.

Initially, the model's architecture was modelled after the VGG16 architecture. This allowed for a better starting point than I would have otherwise have gotten. The structure was as follows: 3 chunks of two convolutional layers followed by a max pooling layer. The number of filters began at 32 for the first chunk and doubled each chunk. This was done as it was believed that the model would need more parameters progressively due to the increasingly specific patterns it learns.

The top of the model consisted of three dense layers. To estimate the optimal node count for these dense layers, model.summary was called on the initial model. It was found that after the final convolutional layer the model moved from roughly 150 thousand to 262 thousand parameters. This was deemed an unnecessary increase. This was because it was believed that the model should have a pyramid-like shape in which after the increasingly specific patterns of the images have been learned in the convolutional layer, the model should then begin funneling node count towards the number of categories. Therefore a decrease in node count was investigated. However, it was found that this did not help, and the node count was left as it was.

The next thing that was investigated was the model’s generalisation. The higher the accuracy before external and internal measures start deviating indicates a better generalisation. From the previous analysis, it was found that dropout and weight regularisation helped greatly. Both l1 and l2 regularisation was trialled. Both of these regularisation techniques increased validation accuracy before overfitting. This means that it can be extrapolated that it will also perform better on new data/test data, and subsequently both were implemented.

In the theme of improving generalisation, dropout was investigated next. A dropout layer of 0.5 was put after each convolutional chunk/dense layer. This promotes the model to not learn noise specific to the training data as some of the information will be dropped each layer. It was found that a dropout value of 0.5 increased generalisation. After this a slightly lower proportions of dropout was trialed. A value of 0.4 proved best in promoting generalisation. Finally, it was thought that perhaps an increasing value of dropout may be helpful. The reasoning behind this was that the model is learning increasingly specific information in each subsequent layer. Therefore it has an increasing chance of overfitting to noise, and will require a larger dropout value. This did not end up working, and a constant level of 0.4 was used for dropout.

The next step in improving the models ability to generalise was to test batch normalisation layers. Batch normalisation is a technique in which the data is normalised after each layer. This helps with normalising gradient descent and helps prevent the vanishing gradient problem. It was found that batch normalisation helped greatly in increasing peak validation accuracy, and was therefore included.

Upon research on classification using convolutional neural networks, the Elu-activation function was discovered. It promised to solve the “dead relu” problem. This is where the gradient during back propagation becomes perpetually zero and effectively means the model stops training. The elu function solves this as it has a value not equal to zero for negative inputs. It was found that the use of this function increased peak validation accuracy and was therefore implemented.

Finally, after this systematic analysis establishing a general shape and topology for the network, parameter count was optimised. This was done through an investigation on layer count and also nodes within each layer. After this investigation was concluded, it was found that the minimal amount of parameters necessary for peak accuracy was roughly 500k parameters. In the theme of maintaining the smallest model for any given accuracy, this structure was implemented.

Once a general shape for the model was found, intensive techniques were trialled. The first of these was data augmentation. This was where the input training data was slightly increased in size through small translations and shifts of the original data. An increase in the amount of data available for training is an effective way of increasing a models ability to generalise and learn patterns, and data augmentation lived up to this. It greatly increased the number of epochs required to be trained before the flattening of accuracy curves, and increased the peak accuracy simultaneously. This final model reached roughly 90% accuracy with 500 thousand parameters.

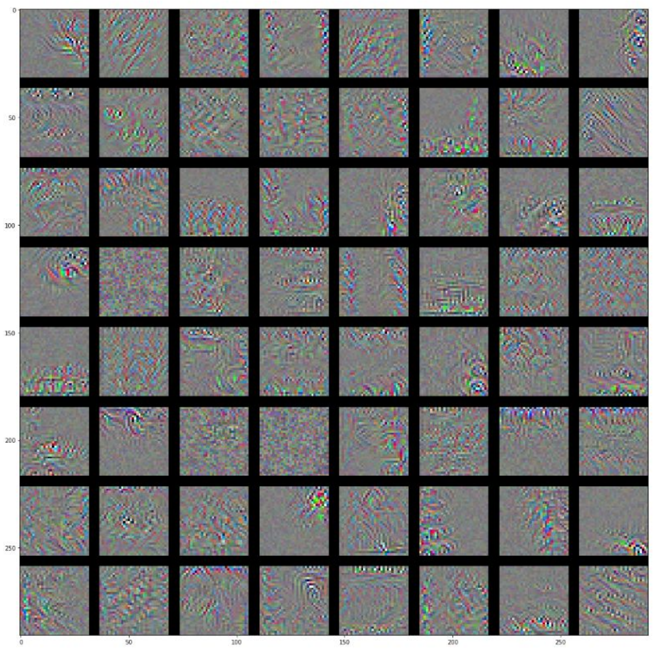
Next, in the aim of maximising accuracy, the use of transfer learning was trialled. The VGG19 model was used for its convolutional base, trained on the ‘imagenet’ dataset. It was hoped that the use of this model as a base for the model would be an improvement on the previously created model due to its training on a much larger dataset. The base was freezed and only the dense layers were trained. The use of an upsampling layer proved extremely helpful as the VGG19 model requires an image size of at least 48x48 and thus the CIFAR10 images needed to be scaled up. On initial training of this model, this did not come to fruition, with a peak validation accuracy of 77%.

However, once this model was trained, fine-tuning could be performed. Fine Tuning is the process in which certain parts of the convolutional base used in transfer learning are made trainable again, in the hopes of increasing the specificity of the layers to the target dataset. This can not be done at the start as training a randomly initialised dense top and a large pretrained base will result in the magnitude of the gradient descent destroying some of the information learned in the convolutional base. This process turned out to be useful and increased peak validation accuracy to just over 90% with 10 million trainable parameters.

Finally, a model had to be chosen. It was decided to use the model without the use of transfer learning due to its massively less amount of trainable parameters and roughly equivalent accuracy.

Visualisation

In the hopes of learning something about how the model makes these predictions, some visualisation techniques were trialled. The following image was produced, visualising what each filter in the final convolutional layer of the data augmented model was looking for:

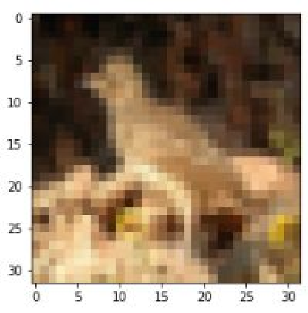


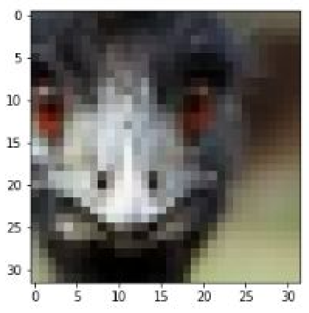
This was plotted as it was believed the final layer would convey the most specific information and therefore the information most likely to be seen but as shown this did not come to fruition. No knowledge was gained about the inherent solution space.

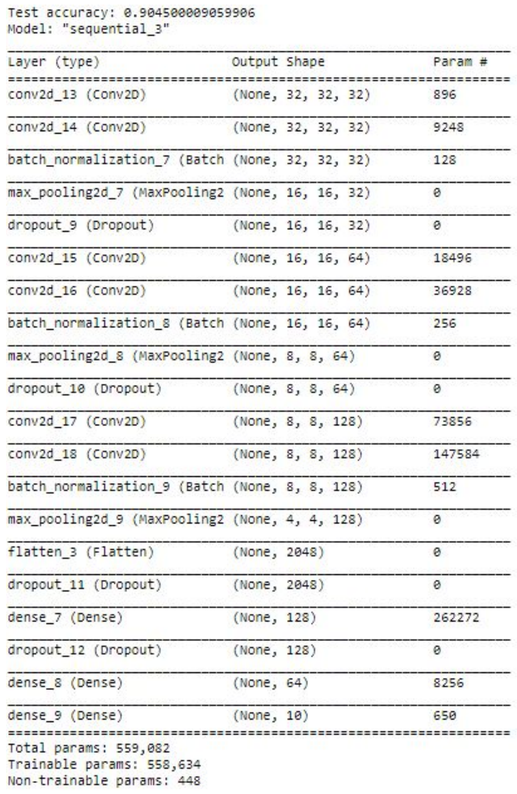
Evaluation and Prediction

The final non-transfer learning model had 500 thousand trainable parameters and achieved a test accuracy of 90.45%. When compared to the base line of 17% presented in the source document for cifar10, the model can be seen as a massive improvement. However, if it is compared to a respective human performance of 97%, the model does not compare. Some investigation of incorrect predictions is as follows:

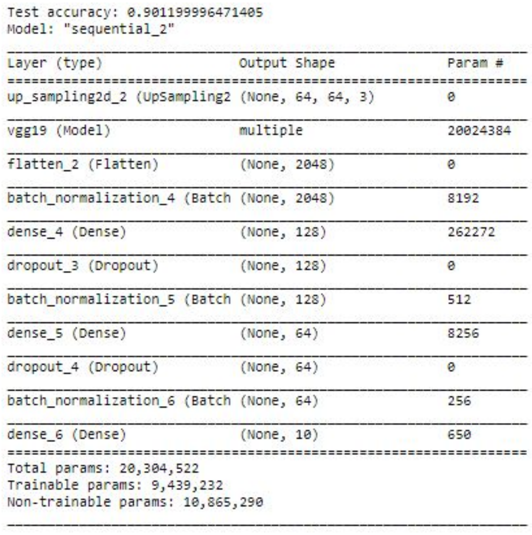
The model has predicted the first image to be a ship. The true category for this is bird. A possible reason for it struggling may be the darker colours and pointy edges all similar to ships. The model has not done terribly, as its second prediction is bird. The second picture has also been incorrectly predicted. Again, the model seems to struggle with the bird category. This time it has predicted it to be a frog. It is hypothesised that the model could achieve near-human performance on categories other than bird or with minimal human supervision.







```
Test accuracy: 0.904500009059906
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_13 (Conv2D)           (None, 32, 32, 32)        896       
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 32, 32, 32)        9248      
_________________________________________________________________
batch_normalization_7 (Batch (None, 32, 32, 32)        128       
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 16, 16, 32)        0         
_________________________________________________________________
dropout_9 (Dropout)          (None, 16, 16, 32)        0         
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 16, 16, 64)        18496     
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 16, 16, 64)        36928     
_________________________________________________________________
batch_normalization_8 (Batch (None, 16, 16, 64)        256       
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 8, 8, 64)          0         
_________________________________________________________________
dropout_10 (Dropout)         (None, 8, 8, 64)          0         
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 8, 8, 128)         73856     
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 8, 8, 128)         147584    
_________________________________________________________________
batch_normalization_9 (Batch (None, 8, 8, 128)         512       
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 4, 4, 128)         0         
_________________________________________________________________
flatten_3 (Flatten)          (None, 2048)              0         
_________________________________________________________________
dropout_11 (Dropout)         (None, 2048)              0         
_________________________________________________________________
dense_7 (Dense)              (None, 128)               262272    
_________________________________________________________________
dropout_12 (Dropout)         (None, 128)               0         
_________________________________________________________________
dense_8 (Dense)              (None, 64)                8256      
_________________________________________________________________
dense_9 (Dense)              (None, 10)                650       
=================================================================
Total params: 559,082
Trainable params: 558,634
Non-trainable params: 448
```



```
Test accuracy: 0.901199996471405
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
up_sampling2d_2 (UpSampling2 (None, 64, 64, 3)         0         
_________________________________________________________________
vgg19 (Model)                multiple                  20024384  
_________________________________________________________________
flatten_2 (Flatten)          (None, 2048)              0         
_________________________________________________________________
batch_normalization_4 (Batch (None, 2048)              8192      
_________________________________________________________________
dense_4 (Dense)              (None, 128)               262272    
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0         
_________________________________________________________________
batch_normalization_5 (Batch (None, 128)               512       
_________________________________________________________________
dense_5 (Dense)              (None, 64)                8256      
_________________________________________________________________
dropout_4 (Dropout)          (None, 64)                0         
_________________________________________________________________
batch_normalization_6 (Batch (None, 64)                256       
_________________________________________________________________
dense_6 (Dense)              (None, 10)                650       
=================================================================
Total params: 20,304,522
Trainable params: 9,439,232
Non-trainable params: 10,865,290
_________________________________________________________________
```