Assignment Two Convolutional Neural Network (CIFAR-10 Dataset)

DTSC13-301

Muiz Murad

13599863

Table of Contents

Executive Summary

The task was to build a model that had a high testing accuracy but did not overfit or underfit. The desired outcome was to have a model with a small number of parameters, high testing accuracy but a balance between its training and validation loss. The process in which to find the best model was based off a logical but sequential method in which multiple baselines models were created and refined, building data augmentation model and then using a VGG model and fine tuning to build the best model. The problems faced within the task are based around infrastructure and resource availability, easier and more processing intensive methods are not able to be used as the machine being used is not able to process the code quickly and efficiently but rather take multiple days and become a more exhaustive and time inducive method. This will overcome by using smaller parameters in which the user will reduce the size of the model based on logical and educated thinking and methods. The final and best model to be used will be a Dropout Model with 256 nodes, 20 epochs, batch size of 480 and dropout rate of 0.1.

Problem Definition and Data Assembly

The input data for the model was the CIFAR-10 data set which is a collection of approximately 60 thousand 32x32 colour images categorised split into batches of 6000 images per class. It is organized in 10 different classes and each class is linked to one of the following: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, or truck. The aim is to predict the accuracy of the model to classify an image provided at random with its associating category. The problem at hand is a supervised learning problem as we are categorising data. We also know the output of the data which further reinforces the supervised learning process. The dataset is mutually exclusive and therefore images will only fit into one out of the ten classes.

Evaluation Protocol Decision

For this case, we need to create a validation set to be used. As we will be building a data augmentation model, we need to consider that we are unable to use the validation split input and as a substitute will need to pre-split the data into validation. With the hold-out method we simply split the data up into testing and training sets to provide two randomized sample groups. We then train the model on the training dataset and validate the model on the testing set. The model evaluation technique we are using is the accuracy metric in the keras package. We can also use the loss function in keras to help us guide our optimization of the model. The loss function is a method of evaluation that tells us how well the algorithm can build a model from the given data. However, if the loss function is too large then it would indicate that the image presented to the model is fitted into an unrelated category. For this dataset we will use the categorical cross entropy loss function as we are trying to classify data to one specific category out of ten different categories as defined by num_classes.

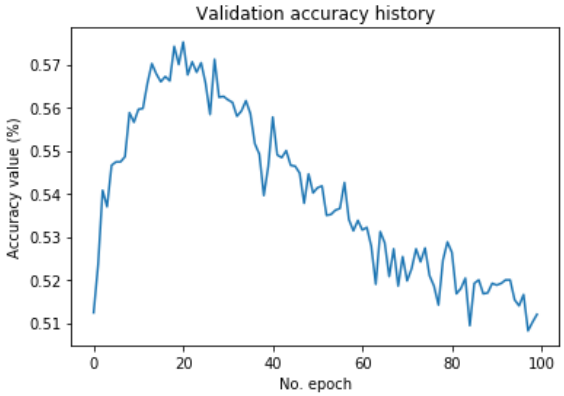
Preparation of Data

We load the data and then determine the shape of one sample, we do this to ensure keras is able to make sense of what type of data it will be using within the input layers of the neural network. After this step we complete two crucial steps to ensure the data can be read quickly and is normalized. Firstly, we convert the data into float32 which reduces the total training process time and then we normalize the data into the same [-1,1] domain as seen in our first assignment with the newsgroup20 dataset.

Development of Model To Beat Baseline

This is a crucial part of the process as we determine the best last-layer activation, loss function and optimization configuration for which the model will run. As we are dealing with a multi-class, single label classification problem we will need to use the soft max as our last-layer activation and sparse categorical cross entropy as our loss function. We use softmax as a it will return probabilities of each target class over all possible target classes. It also summated all probabilities to one and calculated probabilities can easily be interpreted as they are between the range of 0 to 1. Our

optimizer will be adam as it is the easiest to use and combines the most valuable properties of the exisiting RMSProp and AdaGrad optimizers allowing for it to use an optimization fitting process even with noisy data.
To make a model to beat the baseline we will use a 2 VGG Block Style model that will have a fixed set of parameters. As we can build a model that can beat the baseline, we know it is likely that the inputs are useful in predicting the outputs. We have created a basic model by putting together a simple one-layer CNN which demonstrates an approximately 50% accuracy, this is shown in our graph below:



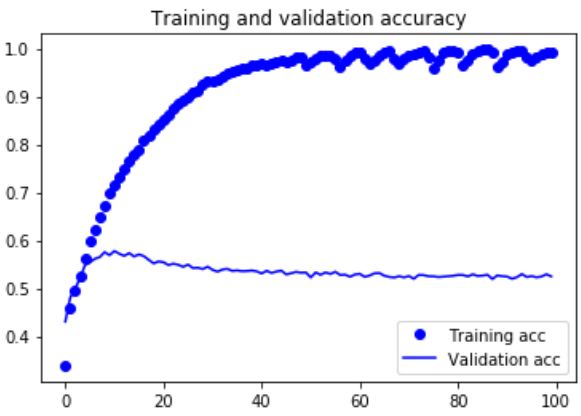
We will now further onto developing a model(s) that will demonstrate higher accuracies and clear signs of over fitting to indicate us how we need to create a balance between our inputs to achieve the best model to ensure it isn't under or overfitting extremely.

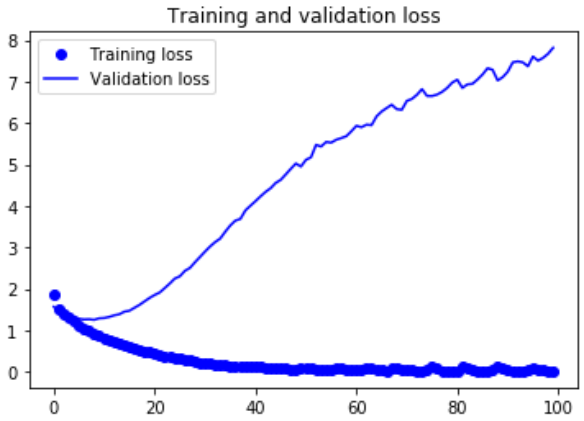
Scaling Up: Developing A Model That Overfits

With a dataset so large we will need to create various baseline models to see what model provides us with the best balance between over and under fitting. To do this we will create 3 basic models with a 1, 2 and 3 VGG Style architecture structure. This will also allow us to see how the data is fitting to the various models and how we are able to use logical and sequential steps to balance the model further. The architecture of the models consists of combining convolutional layers with 3 x 3 filters in addition to a Max Pooling Layer. All these features combined create a block which can then be replicated, and the number of filters can be increased leading to a deeper network each time.

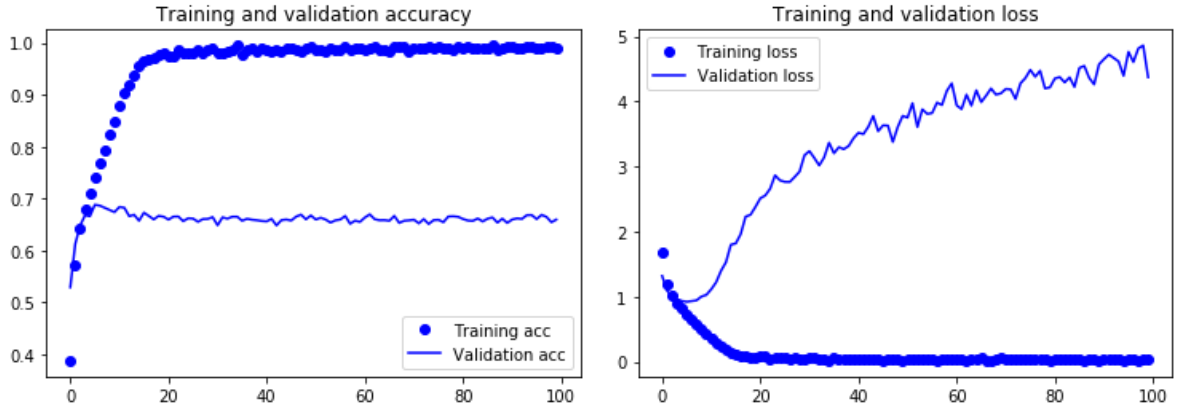
Each of the models will use the same activation function (ReLU) and weight initialization (he_uniform). These are used as they are recommended as best practice for building a model with this type of data.

We will start by developing a simple 1-VGG block style model, which will contain two convolutional layers each with 32 nodes followed by a classifier layer that allows the model to interpret the features of the data and make a prediction to classify the photo to a linking group.

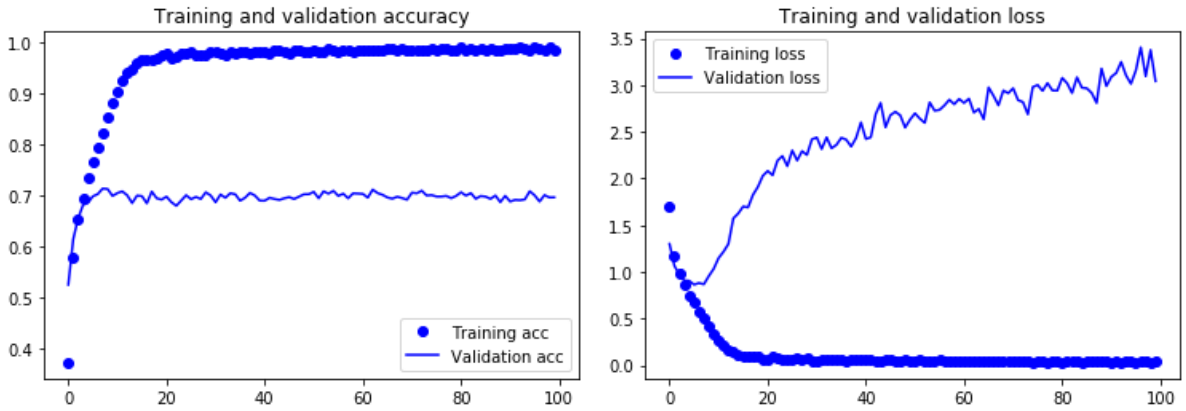




From the plots above we are provided a visualization of the 1 VGG Block style model. We can clearly see that the model is overfitting on the training data. The graph to the right shows us that the validation loss is far greater than the training loss. The training loss is seen to steady off by the 50th epoch however, the training loss is seen to gradually increase after the 50th epoch, drawing the difference between the two and hence making it larger. We can also clearly see from the graph on the left that the training accuracy is nearly 40% greater than the validation accuracy indicating that the model is learning the training data quickly and overfitting very early on, by the 50th epoch as it steadies out. We can also see a similar trend in the validation accuracy which is seen to gradually increase and then steady off by the 50th epoch aswell. This is clearly indicating that the model is quickly overfitting and that the model is learning the data too quickly. Upon looking at the training accuracy, we can see it is 99.42% which is indicating the model is good in classifying the images however is not the best. From here we will now create a 2 VGG Block style model and interpret the results.



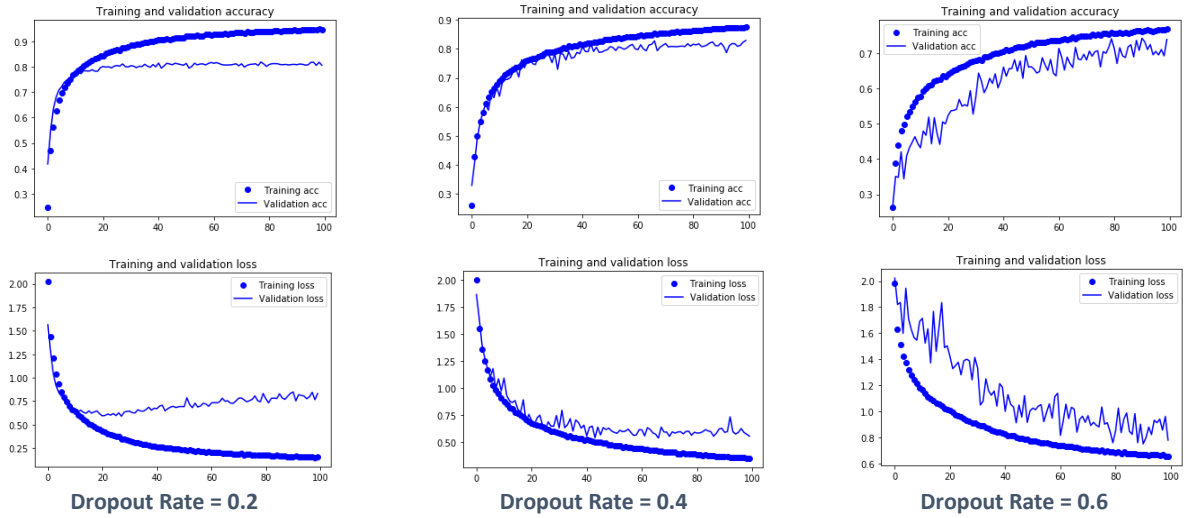
After adding an extra two layers we can see a stark reduction in the maximum validation loss and a higher accuracy on the validation set. However, we need to note that the model is still quickly overfitting with the training accuracy reaching 1.0 within the first few sets of epochs. The gap between the training and validation loss has decreased alongside the gap between the validation and training accuracy indicating that the model is starting to overfit less and create a balance between over and underfitting. A clear indication that this model is better is by looking at the validation accuracy, it has increased by just over 10% on the validation set indicating the model is able to fit hidden data better than the first model whilst retaining its accuracy.



The plot summaries above show us our results from the 3 VGG Block model. We can see that the accuracy has remained the same as before with little change however we can notice a further reduction in the validation loss however we are still overfitting quickly as seen by the quick steadying off from the accuracy curve and training loss curve. The training accuracy is quickly reaching 1.0 indicating the model once again is overfitting quickly. However, in comparison to our previous baseline model we don't see a large enough increase within the accuracy to deem it a

better model and hence we will use the 2 VGG Block model as our Baseline CNN model to further refine and work on as we move onto applying dropout regularization to the model.

We will now begin to investigate the affect of dropout regularization on our best baseline model to further enhance our model. Dropout is largely used within machine learning whereby randomly selected neurons are ignored during the training of the model. This allows an efficient method of averaging neural networks and in our case helps us increase the accuracy of our model as we will see below. We will experiment with three different dropout rates which have been identified as best practice when using a dataset like CIFAR-10, upon building 3 different dropout models we will investigate the plot summaries and summarise our findings leading us onto our data augmentation stage of the process.



From our dropout models above we can clearly see three different models that all performed differently on the best best beat baseline model. The first dropout rate allowed the model to quickly overfit and steady off and not allow a gradual increase in the accuracy or loss whereas the last dropout rate created a lot of noise indicating many inaccuracies and it also had the lowest training and validation accuracy out of the three dropout rates. However, our second dropout rate of 0.4 proved to be the best, this can be seen both in the accuracy and loss. We are able to identify this as the curves of both the validation and train, loss and accuracies are very close to each other and in some instances on top of each other indicating there is no under or overfitting and the model has a very small difference between the validation and train accuracy and loss as the model approaches the 100th epoch. It also demonstrates high training and validation accuracy indicating the data is being fit well onto the model. We will now use this architecture model to further be used on our data augmentation and fine-tuning models.

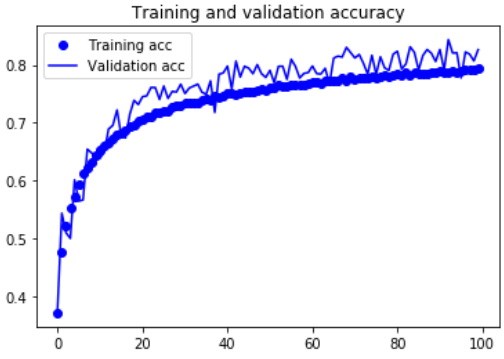
Despite our dropout models having a low number of parameters, it can clearly be seen that the models are performing extremely well on the train set however when run on the holdout set, the accuracy is levelling off around the 40th epoch mark indicating very quick overfitting of the model. We will use the dropout rate within our next models as it has clearly increased the accuracy however we will refine the layers to ensure a more complex model that is able to grow over time and converge at an appropriate time.

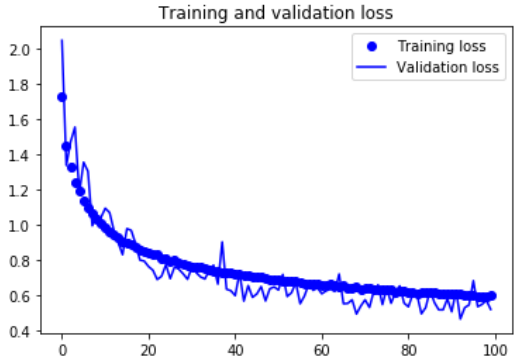
Regularizing Model and Tuning Hyperparameters

For this part of the process we need to start applying different techniques to the model to ensure that we are reducing the number of parameters but at the same time we are increasing the dimensionality and complexity of the model.

We will begin by applying data augmentation to our exisiting best dropout model. Data augmentation will allow us to artificially expand the size of our input data through developing altered versions of the pre-exisiting images within the dataset. We will make use of a variety of parameters that will all hopefully increase our final model accuracy.

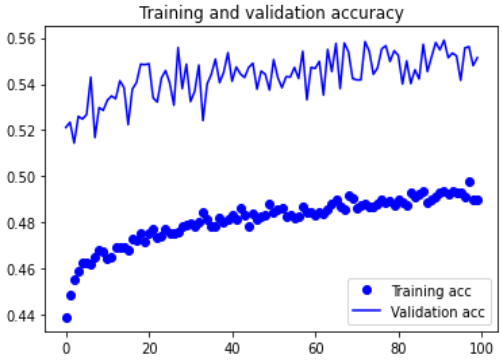
Below we see the results of our data augmentation model which was has used the same architecture as our best refined dropout model.

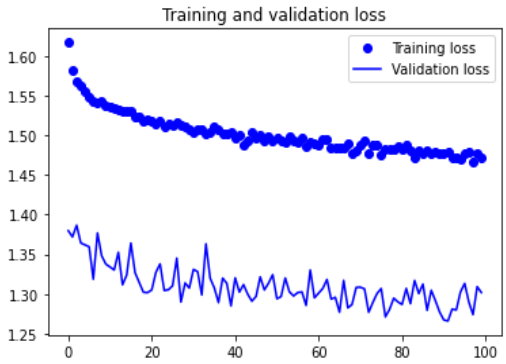




The overall training accuracy of the model was 84.2% which was a 10% reduction from our best dropout model with a dropout rate of 0.4, in addition to its peak validation accuracy of 81%. When looking at total number of parameters, we have reduced them by 10 000 overall however the peak validation accuracy 84.6%. Another clear improvement we could see on the data augmentation model was the loss and accuracy curves. The model is not quickly overfitting as seen by the constant growth of the model and the curves are closer to each other indicating that the model is performing well both on the unseen (holdout) and training set. The graph is showing a noisy set of results indicating fluctuations and minor inconsistencies however overall, it has performed better than our best dropout model. We will now move to investigating the data on a pretrained model (VGG16).

The VGG 16 pretrained CNN is providing a very large model compared to the previous Data Augmentation Model with a total of 15 million parameters, a sheer 1 million more than the data augmentation model. We will analyse the results of the VGG model below however we will not be refining this model as much as our previous ones due to the size and total training time.

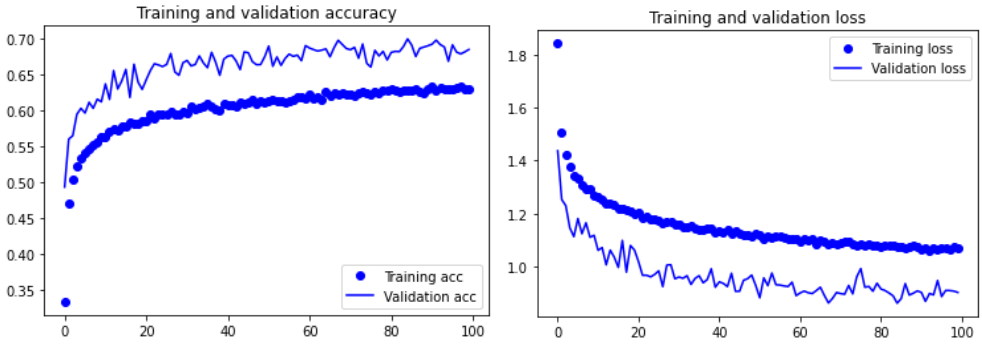




This VGG16 pre-tune model had an overall training accuracy of 56.48% and peak validation accuracy of just over 55%. From the graphs above the model has a large gap between the validation and training losses and accuracies indicating a very unbalanced model. However, we should take note that that validation loss is slightly lower than the training loss and the validation accuracy is higher than the training accuracy. This is interesting and large part in driving this was the removal of the dropout layer. The dropout layer was removed from the VGG model was it was taking a long time to train, and the addition of the dropout layer increased the time further.

We will now investigate the affect of adding an up sampling layer to the VGG model. Why we add it to the model now is that because our accuracy is so low on our VGG16 Model, we can clearly see the effect of the up sampling layer on the model's accuracy. As our first few models were close in their accuracies, it's better to add it to a model whose accuracy is significantly lower than the others to play around with it. The up sampling layer will double the dimensions of our image to 64 x 64 and the best part is that it has no weights.

We clearly see a significant overall increase in the overall train accuracy of the model at 70.56% and a peak validation accuracy of 70%.



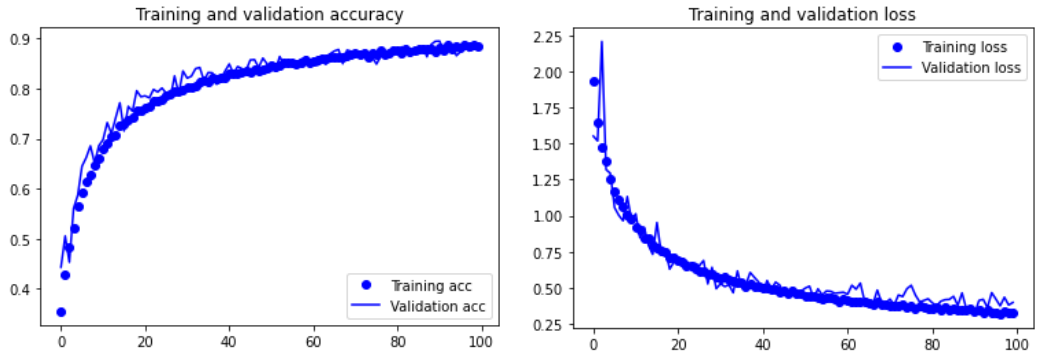
We can see a clear improvement in that the model is taking longer to steady off reducing overfitting, increased accuracy and a reduced gap between the validation and training losses and accuracies. This model has performed well from its base VGG model however due to the size of the parameters it will not be used.

We will now finally tune the VGG model with a different optimizer to see the affect and unfreeze the layers to allow more of the layers to be trained. We get an overall training accuracy of just over 90% and a peak validation accuracy of just over 84%. The model also has reduced overfitting as seen by the graphs. But once again due to the larger number of parameters we will hesitate from using this model. We will use the same structure and architecture from this model to ensure we are keeping the best model layout.

For our final model we will change the optimizer as RMS Prop is considered best use for a CNN Image model, reduce our batch size to increase the model run time and also help reduce the overfitting of the model to make it converge over a longer period of time. We will however increase the complexity of the model by increasing its number of max pooling layers and use the Up Sampling to further increase the accuracy.

The final and best model results are shown below:

Total Number of Parameters = 605 482
Train accuracy = 91.95%
Test Accuracy = 87.05%



The final refined model has performed very well, from the graphs we can see that it has not overfit or underfit indicating a balanced model and we have increased the accuracy. Another interesting point is that the model is not approaching a high accuracy quickly indicating that the number of epochs we have is suitable. The curves are close to each other indicating that the model isn't overfitting or underfitting. Overall, the final model is our best model and has performed very well in comparison to our other ones. We will now do a basic feature visualization.

Best Model Visualization

To extract the feature maps, we want to look at, we will create a Keras model that takes batches of images as input, and outputs the activations of all convolution and pooling layers. To do this, we will use the Keras class model.

The first layer has detected the outline shape of the truck and the size of the body. Looking at the sixth layer, it has detected the edge of the image and the border to identify any features that lie on the edges of the image. It has also pixelized the image more indicating that it was increasing the feature space and diving deeper into the image.

We have just noticed a very important universal characteristic of the representations learned by deep neural networks: the features extracted by a layer get increasingly abstract with the depth of the layer. The activations of layers higher-up carry less and less information about the specific input being seen, and more and more information about the target (in our case, the classes of images such as deer, frog, automobile, truck etc.).

In addition, we have also noticed the below about our model:

- The first layer acts as a collection of various edge detectors. At that stage, the activations are still retaining almost all the information present in the initial picture. It also is identifying the colours of the image as seen by the darker images within the layer.
- As we go higher-up, the activations become increasingly abstract and less visually interpretable. They start encoding higher-level concepts such as “bonnet”, “window” or “body”. Higher-up images represent increasingly less information about the visual contents of the image, and increasingly more information related to the class of the image.
- The sparsity of the activations is increasing with the depth of the layer: in the first layer, all filters are activated by the input image, but in the following layers more and more filters are blank. This means that the pattern encoded by the filter is not found in the input image

Best Model Metrics

By running a simple prediction on the model, we can see that the unseen, test data performed very well on 14 randomly selected images in classifying them. This was to show that the model we had chosen was not struggling to classify the test set images after being trained with the same data.