Fall 2018

Matthew Dobbin

Northwestern University

MSDS 458 Artificial Intelligence

Assignment 4

Contents

1.0 Introduction

This report describes the process of designing, training and assessing several different convolutional neural network models for the application of classifying bee species. The bee image dataset was part of the Drivendata.org competition titled "Naïve Bees Classifier" which was conducted in 2015. The goal of the competition was to predict for each image whether it is a bumble bee (Bombus) or a honey bee (Apis). The competition was in collaboration with BeeSpotter, which is a partnership between "citizen-scientists and the professional science community designed to educate the public about pollinators by engaging them in a data collection effort" to obtain baseline information on the population status of honey and bumble bees (DrivenData, 2015).

In this report, the computational requirements and classifying accuracy of the networks with varying architectures are compared. The final part of the report discusses a real-world application of using convolutional neural networks for the detection of invasive species. All tables and images referenced in the report are contained within the Appendix.

2.0 Dataset

Due to the competition ending in 2015 the image dataset was not available for download directly from the Drivendata.org website. The images and the training labels spreadsheet was retrieved from a Github user who competed in the competition (Naik 2015). A training label of 0 indicates that the image is a honey bee and a label of 1 indicates that the image is a bumble bee. As there were no test set labels available, the training data set was split into 70% training data, 30% test (validation).

Figure 1 shows the distribution of the labels. In total there are 3969 labelled images and approximately one quarter are honey bee images and three quarters are bumble bee images. The images vary in terms of image quality, distance from subject, background and position of

the bee. Figure 2 shows an example picture for a honey bee and a bumble bee from the training dataset.

Keras pre-processing tools ImageDataGenerator and flow_from_directory were used to prepare the images for modelling. The code to create the training and validation directories and sorting of the images into the respective folders was commented out after the first time it was run.

3.0 Modelling

Four different convolutional neural network models were trialled for classifying the bee images. To reduce computational time requirements, an AWS Deep Learning GPU instance (Ubuntu p2.xlarge) was used to process the models. The following sections provide further detail on the structure and performance of each of the models.

3.1 Baseline Model

The first model that was trialled was relatively simple and was used as a baseline. The network has three convolution layers and its architecture is shown in Figure 3. Max pooling operations are used after each of the convolution layers. The max pooling operation uses a 2x2 window and is used to down sample the feature maps by half. The outputs are then flattened and fed into a dense layer. As this problem is a binary classification problem, the final dense layer has a size of one and uses a sigmoid activation function.

The model was trained for 20 epochs. The training and validation performance are shown in Figure 4. It is visible in the plots that the model starts over fitting somewhere between 7-10 epochs. The model achieved an accuracy of approximately 83%.

To visualize the feature maps learned by the network, 2D images of every channel were independently plotted for each of the layers. To build these plots, an image of each type of

bee from the training data set were used. These plots as well as the original images are shown in Figure 5, Figure 6, Figure 7 and Figure 8. For the honey bee example, the flower is quite prominent in the image while the bee is quite small. In the first convolution layer the flower outline is visible in most of the channels however the bee is harder to detect and appears in far fewer of the channels. In the bumble bee example, the bee is more prominent in the image. In the first convolution layer, some aspect of the bee outline is recognizable. For both the honey bee and bumble bee examples, it appears that the first convolution layer acts as an edge detector. The activations in the final convolution layer are quite abstract and are not visually interpretable.

3.2 Model 2 – Dropout Layer

The second model that was trialled was very similar to Model 1. To mitigate the overfitting that occurred in Model 1, a dropout layer was introduced right before the densely connected classifier. The architecture for Model 2 is shown in Figure 9. Dropout layers are a commonly used regularization technique for neural networks. It consists of "randomly dropping out (setting to zero) a number of output features of the layer during training" (Chollet, 2017).

Similar to Model 1, Model 2 was trained for 20 epochs. The training and validation performance were plotted and are shown in Figure 10. There was a slight improvement in performance compared to Model 1. Overfitting started to occur somewhere between 10-12 epochs and resulted in a validation accuracy of approximately 84%. This is a 1% increase in accuracy compared to Model 1.

3.3 Model 3 – Data Augmentation

The third model that was trialled used the same network architecture as Model 2. However, data augmentation was implemented on the training images. Data augmentation is another

technique that can be used to mitigate overfitting. It is particularly useful when there is a small sample of images. Additional training images are generated by augmenting the current training samples. This is done by performing a number of random transformations on the images. Figure 11 shows an example of a honey bee image that has been augmented. The transformations that were conducted on the image result in augmented images that still appear realistic.

Similar to Models 1 and 2, Model 3 was trained for 20 epochs. The training and validation performance were plotted and are shown in Figure 12. After the 20 epochs the validation accuracy was approximately 84% which is similar to Model 2. However, the model did not appear to be overfitting. Due to this the model was re-run but this time it was trained for 100 epochs. The training and validation performance of the re-trained model is shown in Figure 13. The validation accuracy improved to approximately 90%.

3.4 Model 4 – Pre-Trained Network

The final model that was fitted used the VGG16 convolutional neural network that was trained on the ImageNet dataset. The architecture for Model 4 is shown in Figure 14 and the VGG16 model summary is shown in Figure 15. The layers from the VGG16 were frozen in order to preserve the representations that were previously learned. The densely connected layers were then fine tuned. The training and validation performance of Model 4 are shown in Figure 16. The accuracy of the validation set was approximately 91%.

3.5 Competition Winning Model

Predictions on the test dataset was not conducted as the competition closed in 2015 and the test labels were not available. The top three submissions were able to achieve scores of 0.99 AUC. All three used pre-trained networks with additional fine tuning. The winning model

extracted the features from the inception outputs from the GoogleNet model and then passed them through a logistic regression model (DrivenData, 2017).

4.0 Management Recommendations

The winning model's ability to accurately classify the bee species from an image enables researchers to more efficiently obtain an understanding of the population and potentially determine if the species is in decline. There are many applications for convolutional neural networks in identifying animals and invasive species. Queensland University of Technology researchers have launched a robotic submarine that can detect crown-of-thorns starfish which are destroying the coral on the Great Barrier Reef. Using computer vision and on-board processing, the robot can identify these starfish with 99.4% accuracy.

Once the crown-of-thorns starfish is identified, the robot can instigate an injection which is fatal to the starfish but does not damage anything else on the reef (QUT, 2018). Deploying the robot is financially more economic that using human divers. Six human divers could cover half the length of the reef in a year and would cost $1.44 million to operate. Six robots could cover the length of the reef up to 14 times in a year and would cost $0.72 million a year to operate (Dunbabin, 2018).

5.0 Conclusion

This report investigated several different convolutional neural network model architectures. As the dataset sample was small, dropout layers and data augmentation were used to mitigate overfitting. The model that achieved the highest accuracy in classifying the bee images used a pre-trained network and achieved an accuracy of 91%.

References

Chollet, F., & Safari, an O'Reilly Media Company. (2017). *Deep Learning with Python*(1st ed.). Manning Publications.

Drivendata. (2015). Naïve Bees Classifier. Retrieved November 10 2018, from https://www.drivendata.org/competitions/8/naive-bees-classifier/page/29/

Drivendata. (2017). Competition results for the Naïve Bees Classifier competition. Retrieved December 2 2018, from https://github.com/drivendataorg/naive-bees-classifier/tree/b56b120ff9c51b0ec332e59c769b48c67e43ef39

Dunbabin, M. (2018). Robo reef protector to save the Reef RangerBot. Retrieved December 5 2018 from https://www.barrierreef.org/science-with-impact/swiss-army-knife-for-the-great-barrier-reef

Naik, C. (2015). Bee classifier using cnn. Retrieved December 1 2018, from https://github.com/chetannaik/bee_classifier_using_cnn/tree/master/dataset

QUT. (2018). Robot reef protector sees a new way to check Great Barrier Reef health. Retrieved December 3 2018 from https://www.qut.edu.au/news?id=135108

Appendix

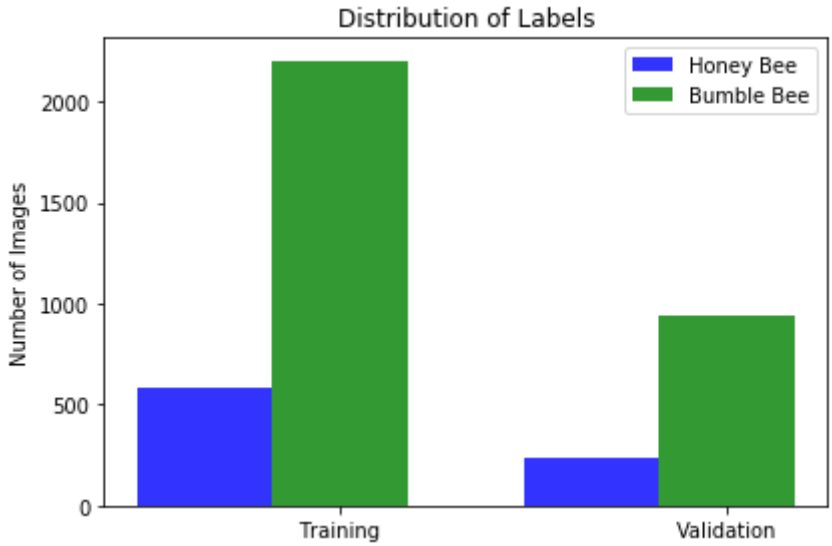

Figure 1 - Distribution of the labels for the bee classifier data set.

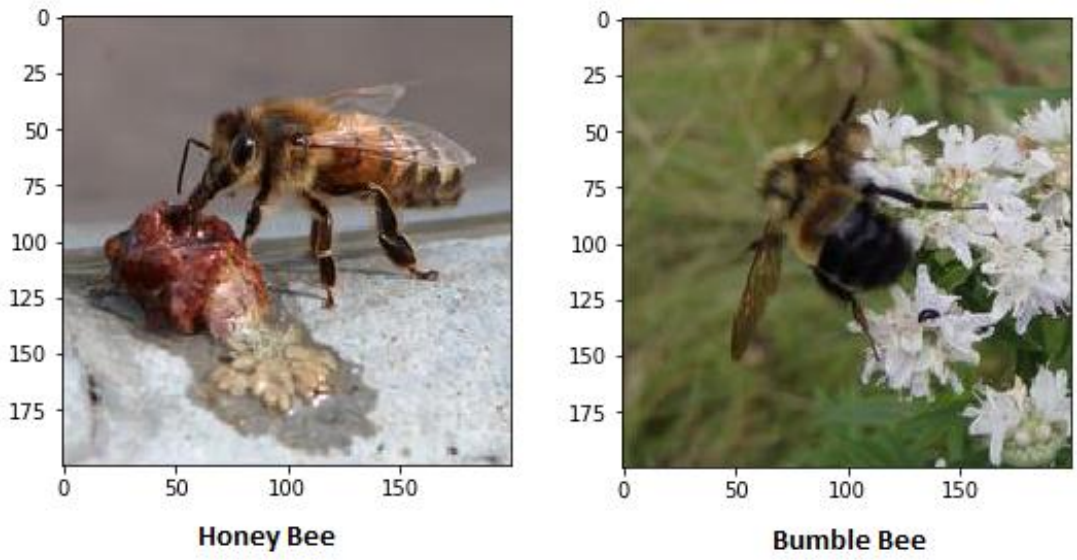

Figure 2 - An example of a honey bee and a bumble image from the data set.

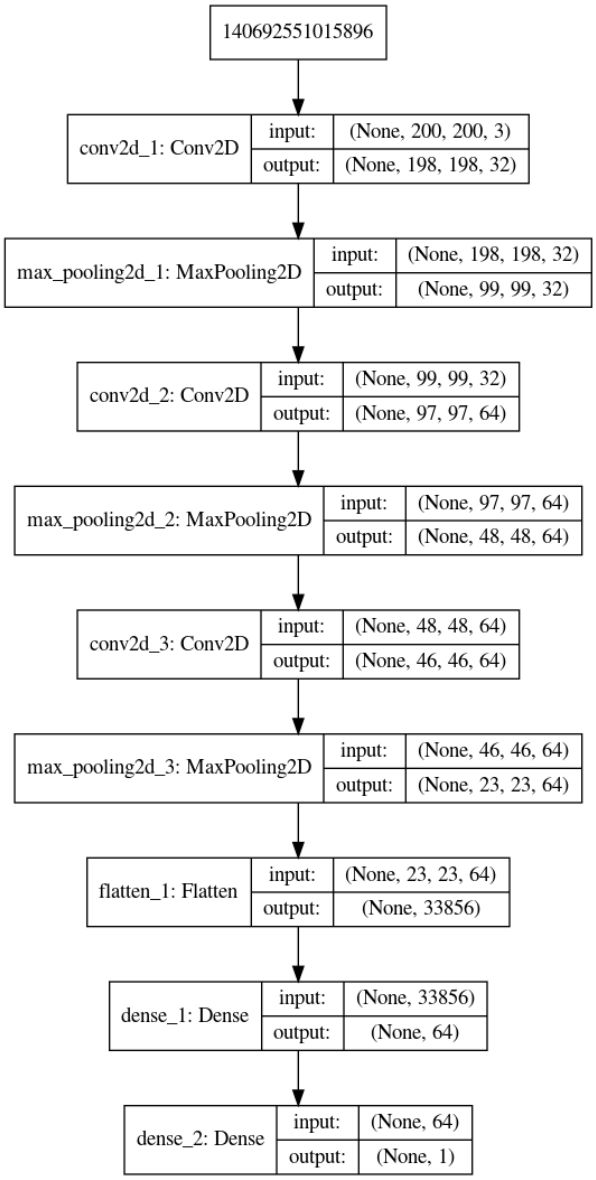

Figure 3 - Model 1 convolutional neural network architecture.

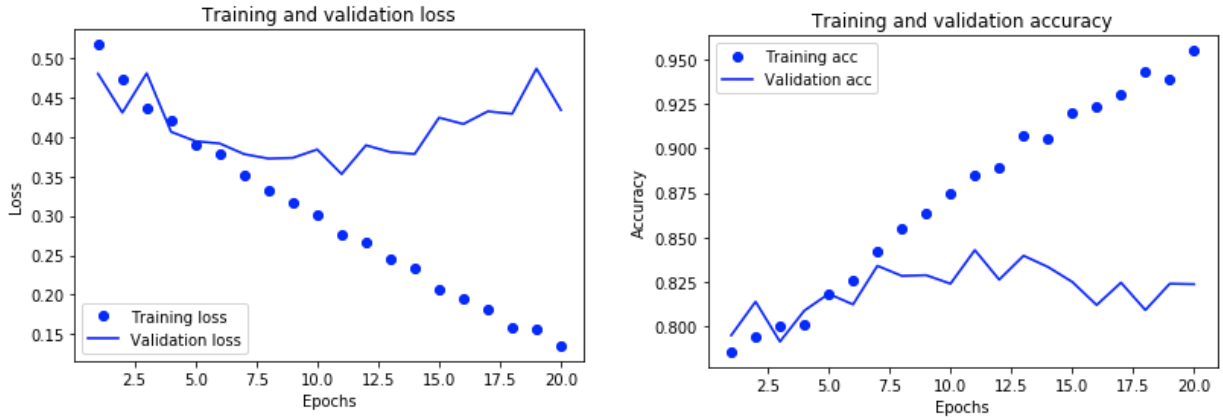

Figure 4 - Training and validation performance for the first model (CNN_1).

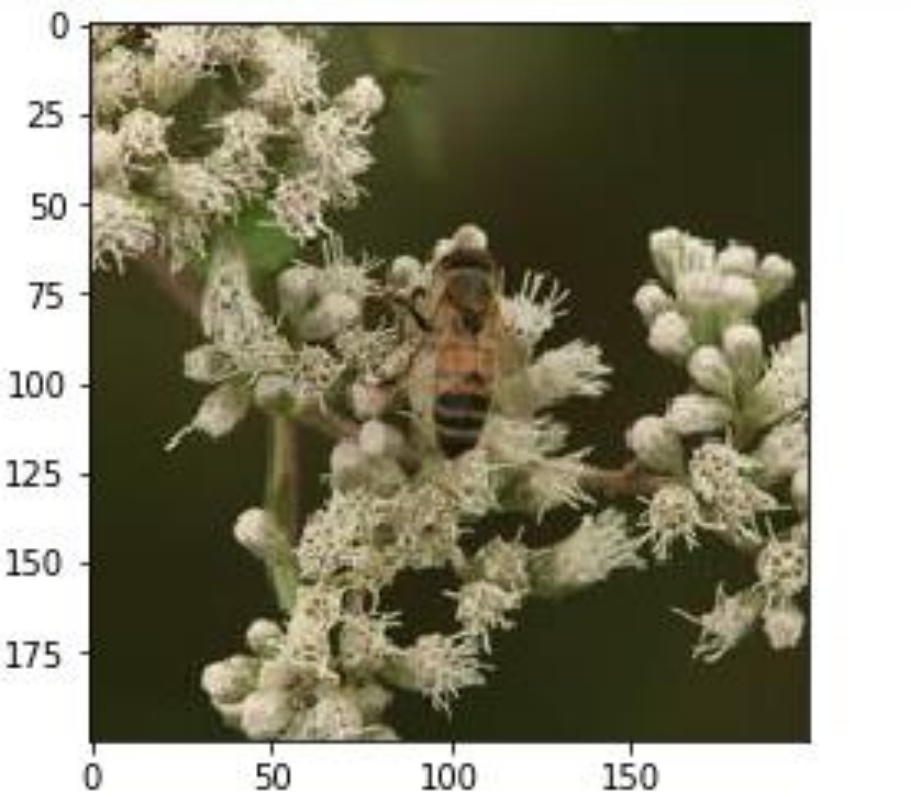

Figure 5 - Honey bee image from training data set.

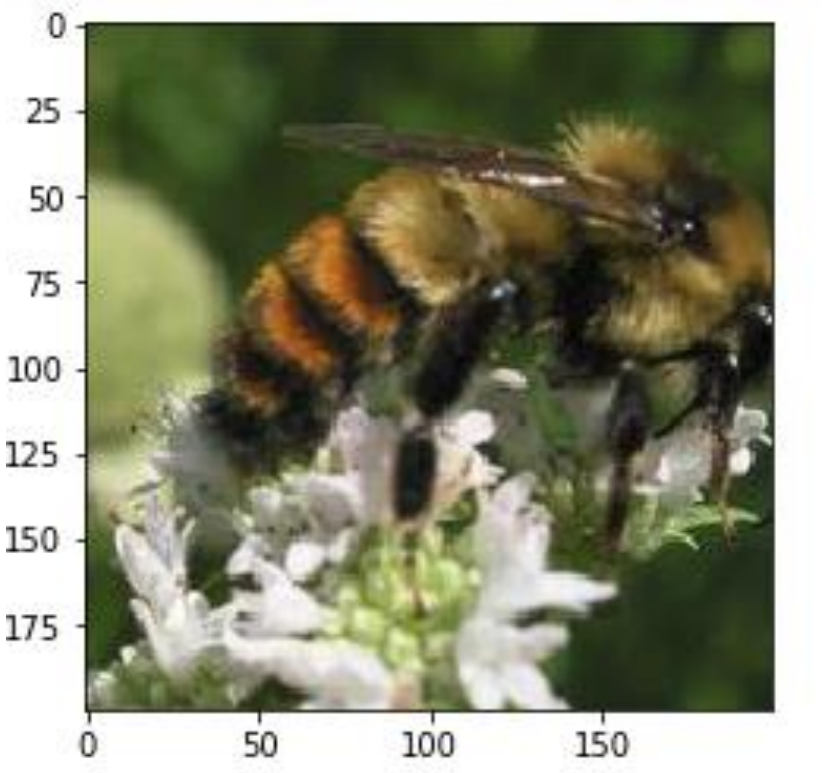

Figure 6 - Bumble bee image from validation data set.

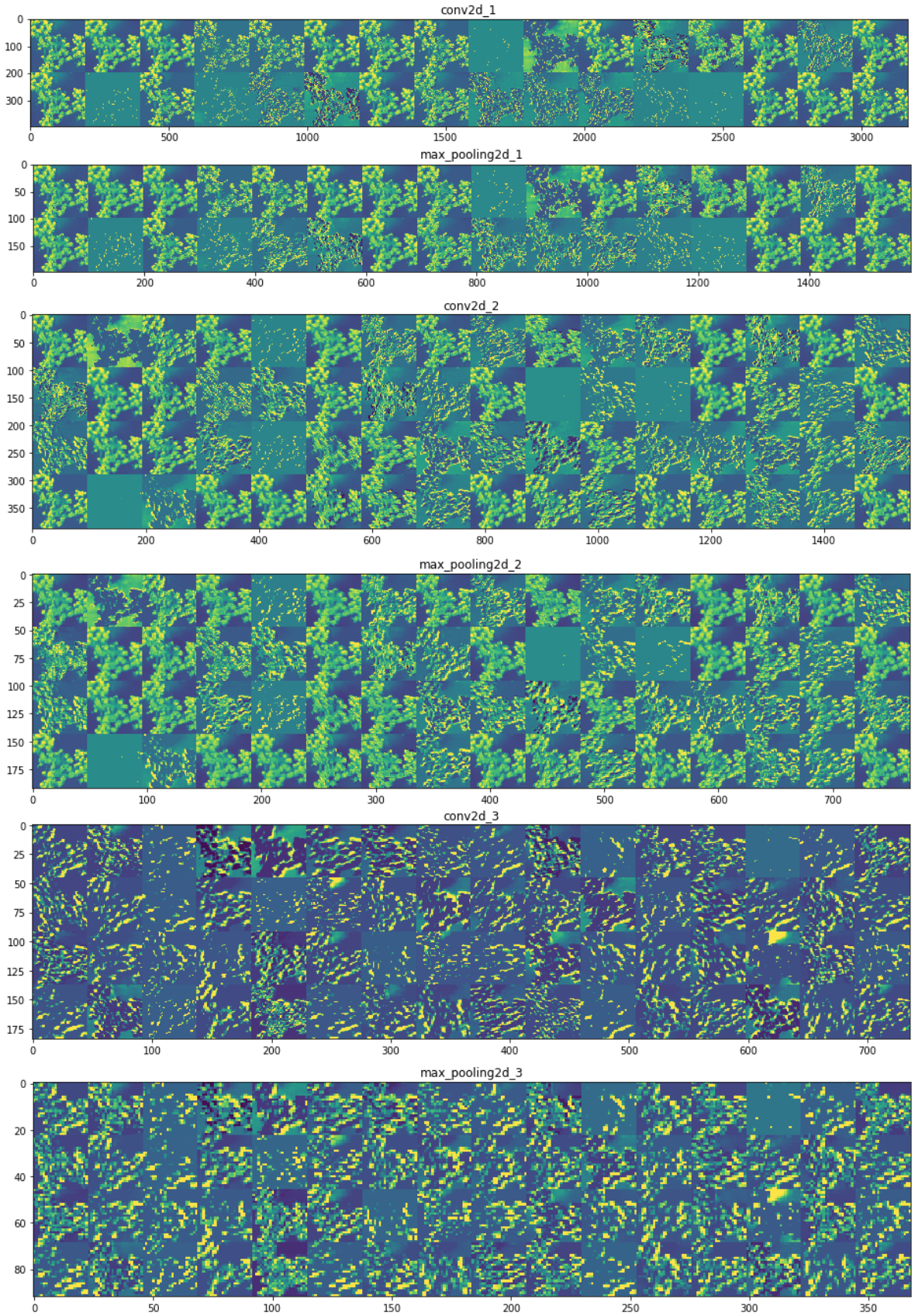

Figure 7 - Visualisation of every channel of every layer activation for the honey bee image.

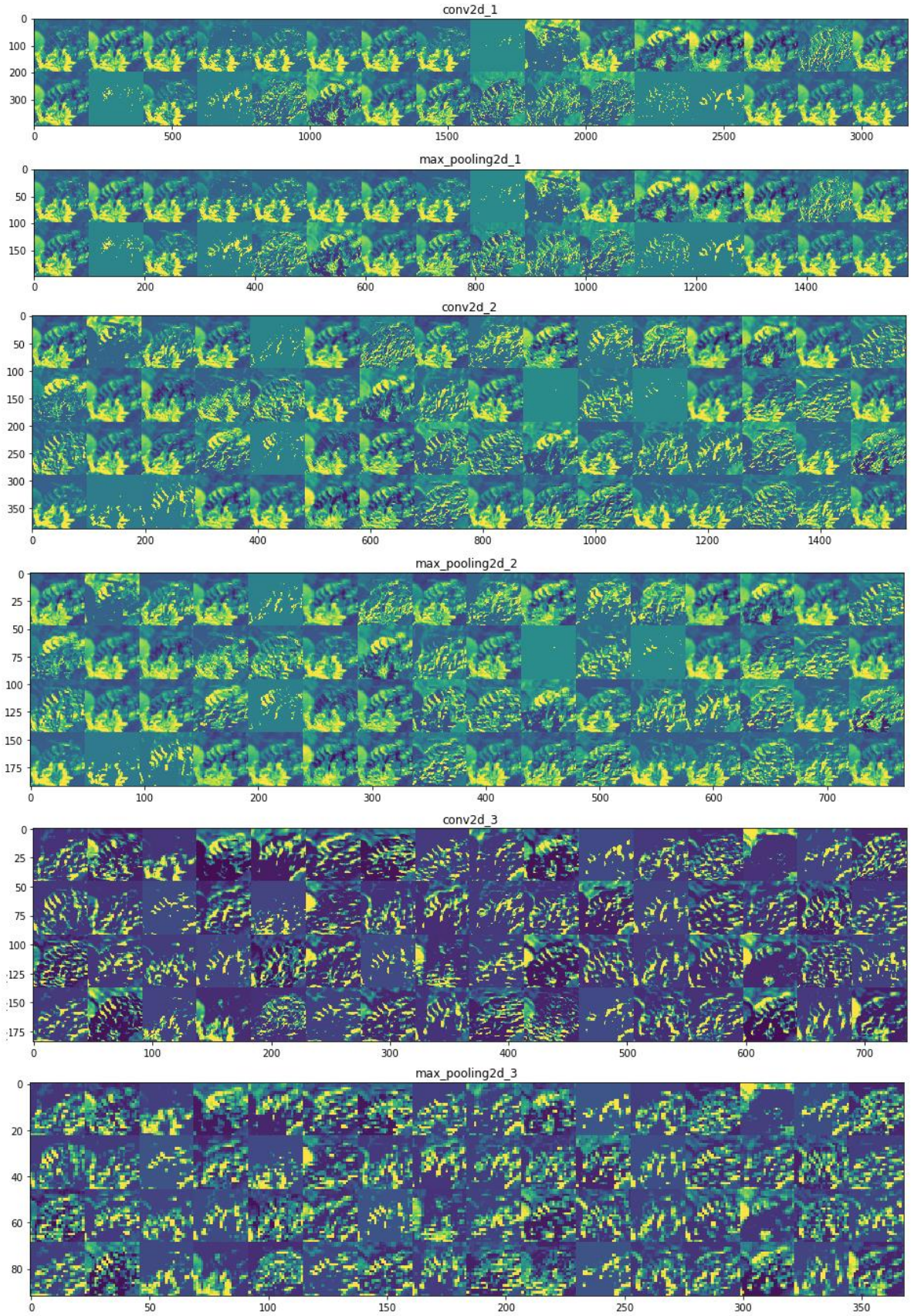

Figure 8 - Visualisation of every channel of every layer activation for the bumble bee image.

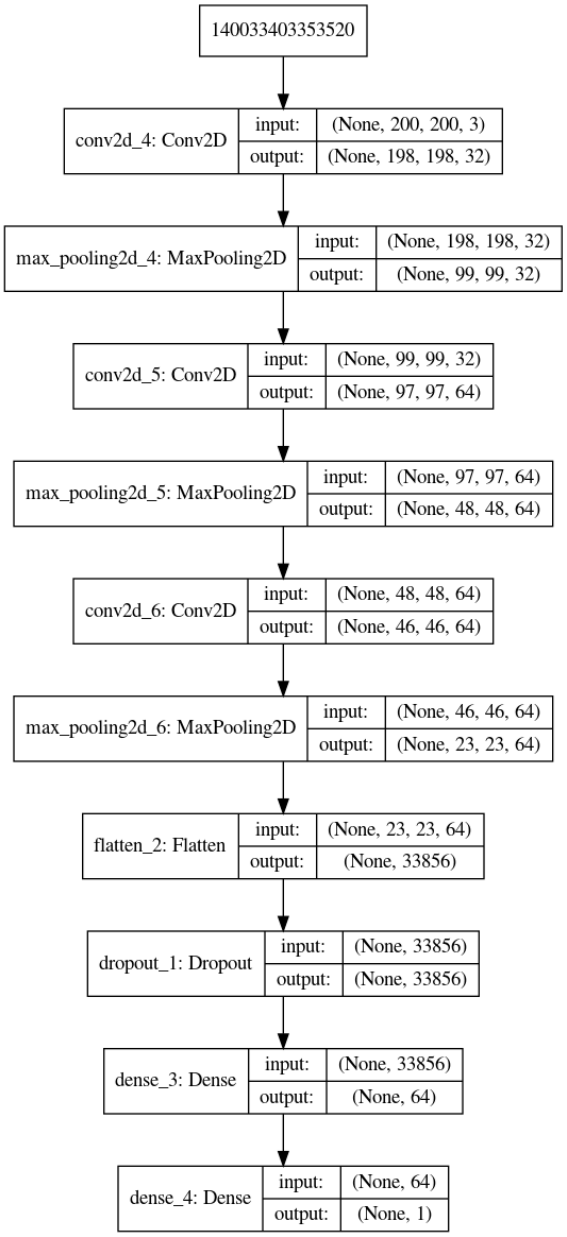

Figure 9 - Model 2 (CNN_D) architecture which uses a dropout layer.

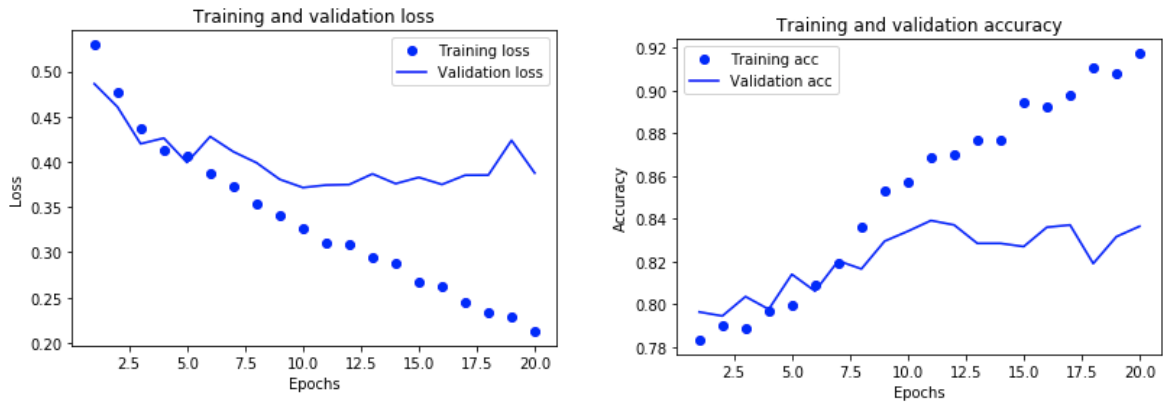

Figure 10 - Training and validation performance for the second model which uses a dropout layer.

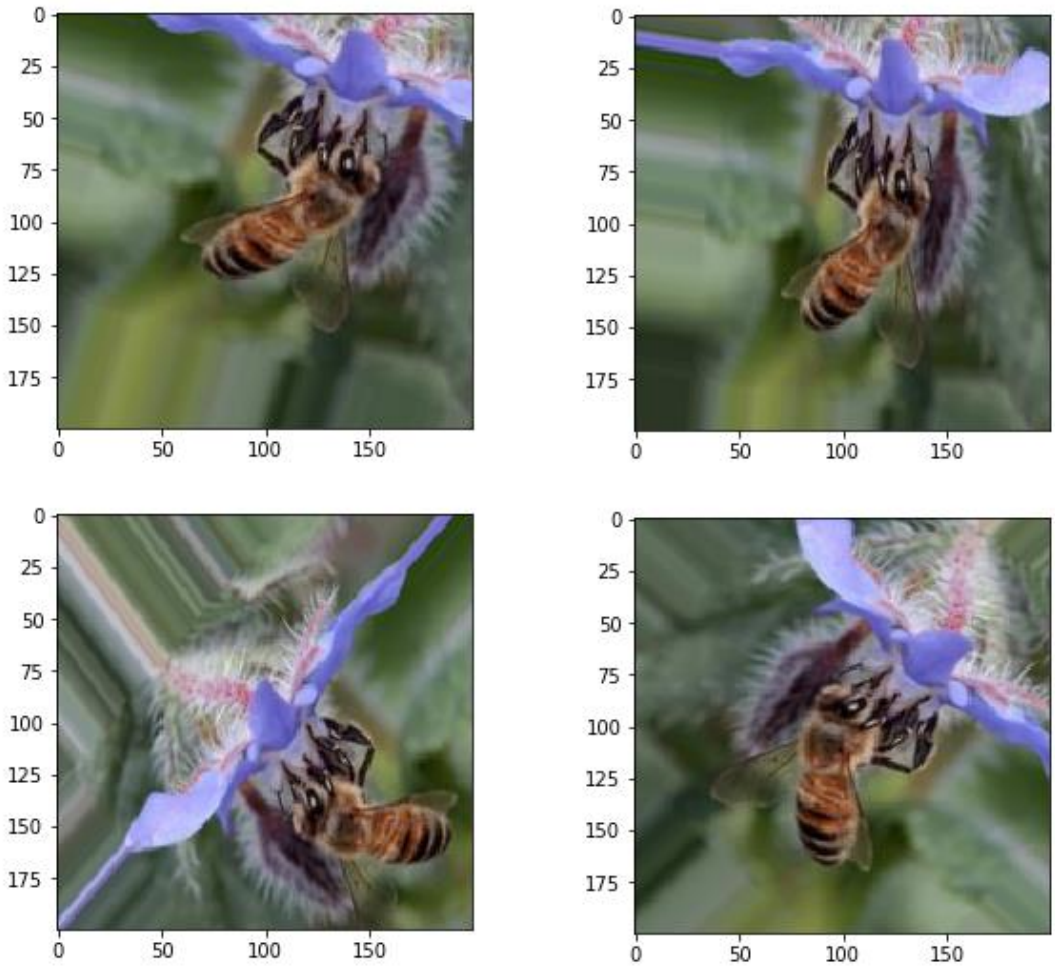

Figure 11 - Data augmentation on a honey bee training image.

Figure 12 - Training and validation performance for third model (CNN_A_20) which has Model 2 network structure plus data augmentation.

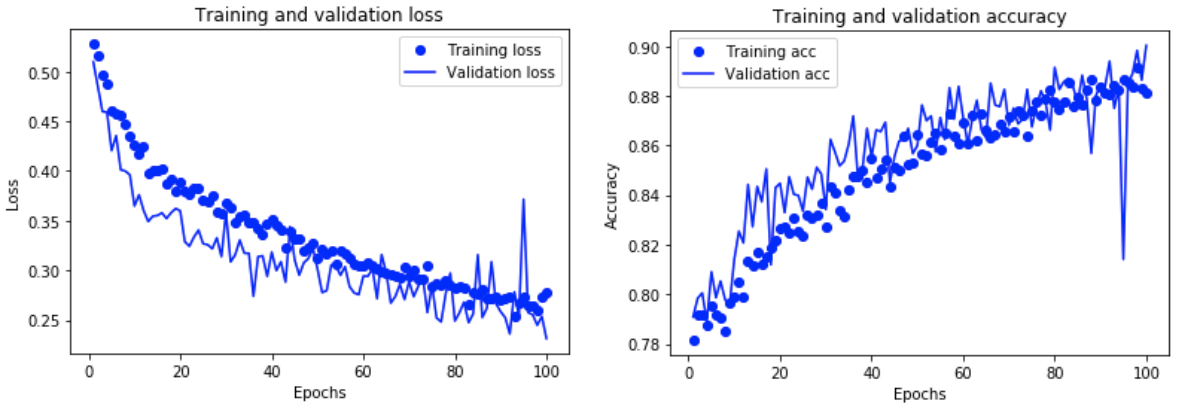

Figure 13 - Training and validation performance for CNN_A_100 (Model 3 run for 100 epochs).

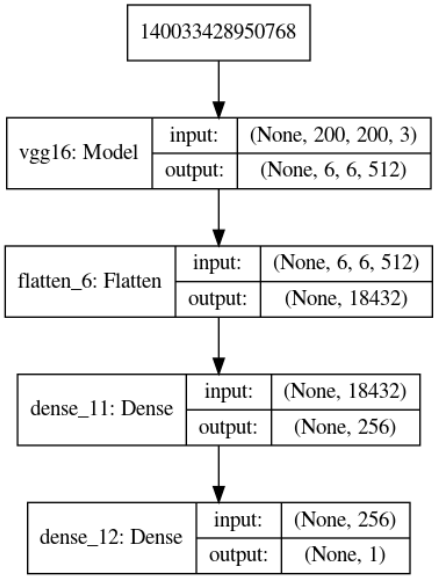

Figure 14 - Network architecture for pre-trained model.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 200, 200, 3)	0
block1_conv1 (Conv2D)	(None, 200, 200, 64)	1792
block1_conv2 (Conv2D)	(None, 200, 200, 64)	36928
block1_pool (MaxPooling2D)	(None, 100, 100, 64)	0
block2_conv1 (Conv2D)	(None, 100, 100, 128)	73856
block2_conv2 (Conv2D)	(None, 100, 100, 128)	147584
block2_pool (MaxPooling2D)	(None, 50, 50, 128)	0
block3_conv1 (Conv2D)	(None, 50, 50, 256)	295168
block3_conv2 (Conv2D)	(None, 50, 50, 256)	590080
block3_conv3 (Conv2D)	(None, 50, 50, 256)	590080
block3_pool (MaxPooling2D)	(None, 25, 25, 256)	0
block4_conv1 (Conv2D)	(None, 25, 25, 512)	1180160
block4_conv2 (Conv2D)	(None, 25, 25, 512)	2359808
block4_conv3 (Conv2D)	(None, 25, 25, 512)	2359808
block4_pool (MaxPooling2D)	(None, 12, 12, 512)	0
block5_conv1 (Conv2D)	(None, 12, 12, 512)	2359808
block5_conv2 (Conv2D)	(None, 12, 12, 512)	2359808
block5_conv3 (Conv2D)	(None, 12, 12, 512)	2359808
block5_pool (MaxPooling2D)	(None, 6, 6, 512)	0

Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0

Figure 15 - Summary of VGG16 model.

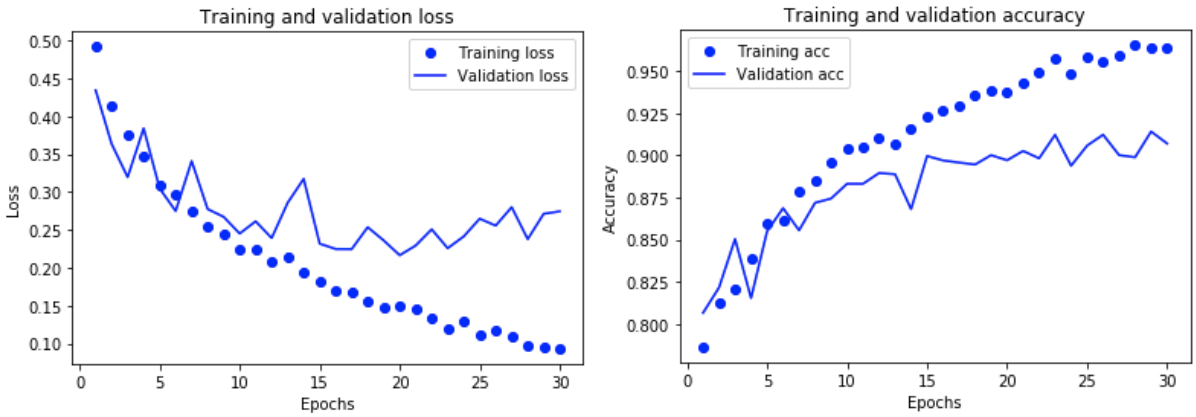

Figure 16 - Training and validation performance for CNN_VGG16 model.

Table 1 - Comparison of overall performance and computational time for each network architecture.

Model	Computational Time	Number of Epochs	Training Set Accuracy	Validation Set Accuracy
CNN_1	4min 6sec	20	0.96	0.83
CNN_D	4min 7sec	20	0.91	0.84
CNN_A_20	12min 53s	20	0.8192	0.84
CNN_A_100	1hr 5min	100	0.88	0.90
CNN_VGG16	20min 41s	30	0.96	0.91