Categorizing Equus Members with Deep Learning

Miguel Corona

April 2020

1 Introduction

The objective of the project is to create a convolution neural network that can distinguish the members belonging to the Equus genus. The animals belonging to the Equus genus are horses, donkeys, and zebras. The network shall support the categorization of each pure breed i.e. hybrids such as mules are omitted due to being part horse and part donkey.

2 Data Preparation

2.1 Image Selection

Approximately 1044 total images were collected of horses, donkeys, and zebras. The images were selected such that each one only contains the animal associated with the image's label i.e. there are no images of hybrids nor images that contain that contain more than on class. Images that contain multiple instances of the same class were permitted as entries to the dataset.

2.2 Preprocessing Data

The images were selected to emphasize their label. Any images that contained more than one class were cropped to ensure that a single class is present in the image. All samples were adjusted to become square images of dimensions 256 x 256 pixels with the use PIL. Landscape images were cropped to focus on the label to minimize potential information loss on the resizing of the images.

2.3 Data Distribution and Visualization

The distribution of samples is broken down in the following manner:

Donkey	392
Horse	315
Zebra	340

The reasoning for selecting more Donkey samples than those of the zebra is due to its variability. The breakdown may indicate that sufficient records are needed to distinguish between donkeys and adult horses and adult zebras, but additional donkey samples assist in determining whether a donkey or young horse or young zebra is present. Figure 1 contains a depiction of the data visualization.

Figure 1: Visualization of Equus Samples and Labels

2.4 Data Normalization

Normalization was performed on the images via rescaling of the RGB values such that the values lie in the [0,1] interval. The rescaling of the RGB values on the images may alleviate the skewing that may occur on the grayscale images that may be present in the dataset.

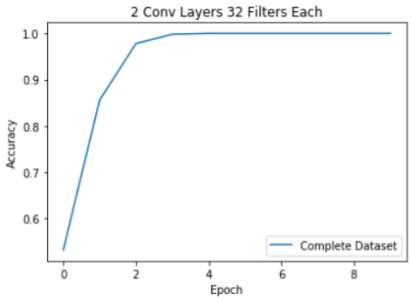
3 Building an Overfitting Model

3.1 Initial overfitting model

An initial convolution neural network was developed to identify whether it was feasible to yield a model that can properly overfit the dataset. All samples were utilized for training the model and no samples were partitioned into a validation set or test set. The initial neural network is composed with the following architecture:

Type	Parameters	Kernel Size	Activation
Convolution	32	3	relu
Convolution	32	3	relu
Dense	3	N/A	softmax

The model was trained across 10 epochs and yields the learning curves:

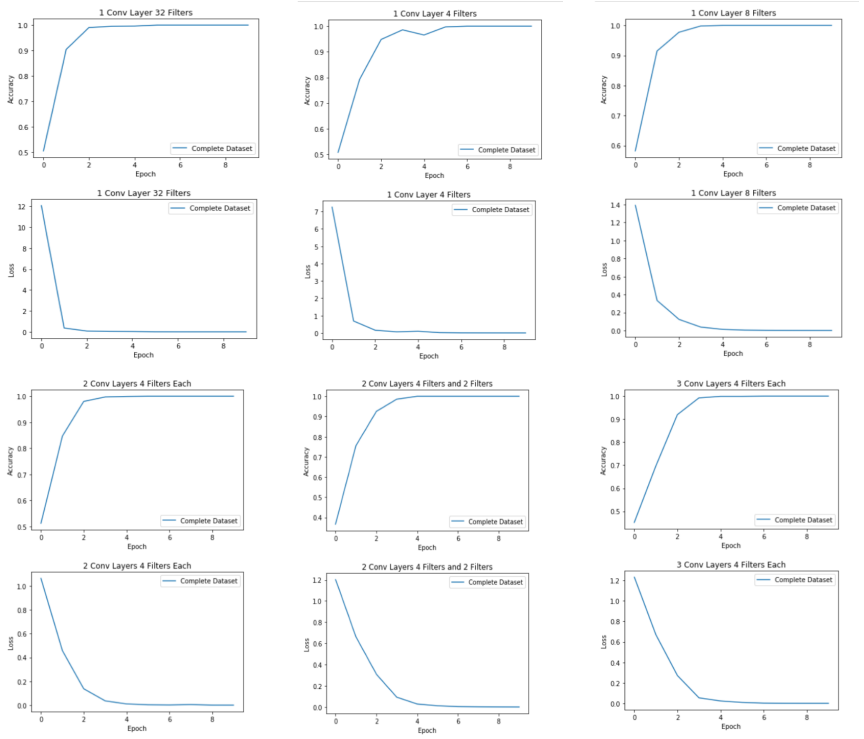


Params	Epoch (s)	Final Acc	Final Loss	Overfit Epoch
6,106,531	84s	1.0000	1.4374e-04	5

The accuracy of the model converged to an overfitted state by the midpoint of total epochs defined. Each epoch was approximately 84 seconds which indicates that the initial overfitting network may not be optimal as less complicated models discussed later yield comparable results. The largest improvement in the accuracy of the model occurred at the early stages of training and the loss converged to its minimum value.

3.2 Performance Dependence on Filters and Layers

Additional networks that differ in the number of filters and the number of layers were constructed once it was determined that overfitting was plausible. Each of the networks is described in the table by the number of convolution layers and number of filters per layer. All networks utilize relu as the activation function for each convolution layer and have a dense layer with a softmax activation as the final layer. The learning curves of the variable networks are depicted below:



Model	Params	Epoch (s)	Final Acc	Final Loss	Overfit Epoch
1 Conv 32	6,194,435	18s	1.0000	3.7188e-04	6
1 Conv 4	774,307	11s	1.0000	0.0031	7
1 Conv 8	3,097,219	13s	1.0000	0.0016	5
2 Conv 4 4	762,311	23s	1.0000	5.7619e-04	6
2 Conv 4 2	381,213	22s	1.0000	0.0017	5
3 Conv 4 4 4	750,411	35s	1.0000	5.3596e-04	7

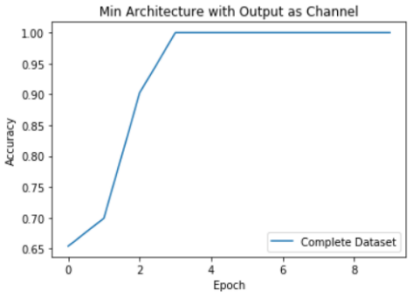
The results of the variable networks seem to indicate that the number of layers do not have a significant effect on the convergence of a model to an overfitted state if sufficient epochs are performed. All models perfectly classified each sample in the dataset albeit at different epochs and with different final loss with sufficient training. The number of layers or filters alone in the network do not

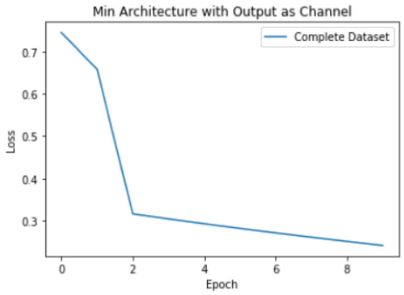
seem to have a significant impact on the final loss compared to the impact of the total number of parameters as the networks with the most parameters yield smaller loss values up to a certain point. The number of layers may affect the loss by favoring the model with less layers when the number of parameters in the network are comparable. It was observed that the number of layers impact the time required to complete an epoch. The observation between the number of parameters and the number of layers are consistent when the results are compared to the initial overfitted model as the number of parameters between 2 Conv 32 32 is comparable to that of the 1 Conv 32 model; Both yield comparable loss values, but with a significant difference in the epoch completion time.

3.3 Minimum Architecture if Label as an Input Channel

An experiment was performed to identify the minimum architecture required to overfit the dataset if each sample contains its label as an additional channel. The images acquired from the ImageDataGenerator were converted from their initial shape of (256, 256, 3) to a shape of (256, 256, 6). The first three elements of the final axis represent the RGB value of the pixel and the last three elements contain the tensor representation of Donkey, Horse, or Zebra. The tensor elements in the final axis remain normalized as the RGB values were normalized via the ImageDataGenerator and the classification tensor contains a single 1.0 value to represent the label.

The network was trained utilizing model.fit() unlike the prior networks due to the transformations required to reshape a sample to shape (256, 256, 6). The network was trained for 10 epochs similarly to the prior models to yield the results:





Params	Epoch Time (s)	Final Acc	Final Loss	Overfit Epoch
387,209	13s	1.0000	0.2416	4

Providing the classification of the sample as an input channel permits a simpler network to overfit the data much sooner than the models observed earlier. An area of concern with having the label as an input channel is the resulting loss as this model overfits sooner, but yields a relatively large loss compared to the prior models. The 2 Conv 4 2 model contains a comparable number of parameters to the minimum architecture with the label as an input channel, but the loss for each respective number differs in magnitude with the 2 Conv 4 2 model yielding a smaller loss value.

4 Split and Evaluate on Test Set

4.1 Split Data Sets

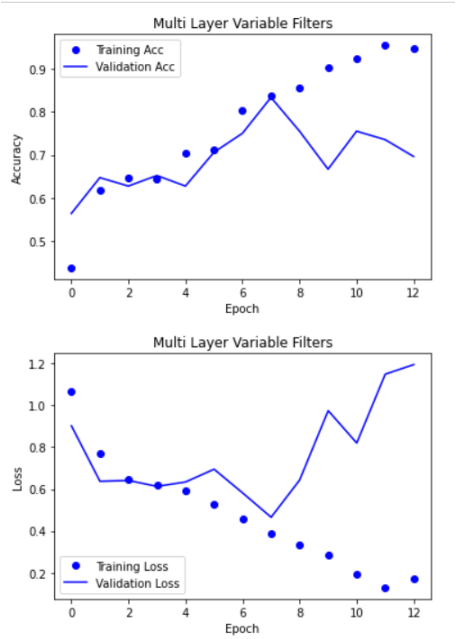
A series of convolution neural networks were constructed and evaluated against a validation and test set. Evaluations were performed via hold-out validation which results a limited training set as the validation set was treated similarly to a test set. The 1044 images composing the dataset were partitioned into a training set of 610 images, a validation set of 204 images, and a test set of 230 images. The data split is provided in the table:

Dataset	Donkey	Horse	Zebra	Total
Training	240	185	185	610
Validation	75	65	64	204
Test	76	64	90	230

Adjustments were made onto the hyperparameters of the model based on the validation loss to yield a model with the architecture:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 254, 254, 12)      336
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 127, 127, 12)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 125, 125, 12)      1308
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 62, 62, 12)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 60, 60, 32)        3488
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 30, 30, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 32)        9248
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 24)                602136
_________________________________________________________________
dense_1 (Dense)              (None, 12)                300
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 39
=================================================================
Total params: 616,855
Trainable params: 616,855
Non-trainable params: 0
_________________________________________________________________
```

EarlyStopping and ModelCheckpoint callbacks were placed onto the model's training. The model checkpoint ensured that the model with the minimal loss on the validation is saved so that it may be retrieved to evaluate on the test set. A preference was made to select the model with the minimal validation loss as the validation loss acts as an estimate for the loss on the test set. It is preferable to have a model that performs equally well on the training set and validation set than one that overfits the training set.



The learning curves indicate that the accuracy and loss for the training data and the validation data followed the same trend through the seventh epoch. The seventh epoch yielded the minimal validation loss and it is the point where the validation accuracy and training accuracy were most comparable at 0.8333 and 0.8361 respectively. The learning curves suggest that the model begins to overfit based on the manner the validation accuracy and loss diverge from the training accuracy and loss. The model with the minimal validation loss was selected and evaluated on the test set to yield:

Loss	Acc	Val Loss	Val Acc	Test Loss	Test Acc
0.3897	0.8361	0.4653	0.8333	0.7365	0.7739

The model performed better on the on evaluating the validation set than on the test set. An imbalance between the validation set and the test set may have contributed to the discrepancy in the loss and accuracy.

4.2 Performance Based on Hyperparameters

The performance of models with varying hyperparameters in the convolution layers were observed. The dense layers and their parameters remained constant. The models observed contain the following architectures along with their learning curves:

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 254, 254, 6)	168
max_pooling2d_3 (MaxPooling2	(None, 127, 127, 6)	0
conv2d_5 (Conv2D)	(None, 125, 125, 6)	330
max_pooling2d_4 (MaxPooling2	(None, 62, 62, 6)	0
conv2d_6 (Conv2D)	(None, 60, 60, 16)	880
max_pooling2d_5 (MaxPooling2	(None, 30, 30, 16)	0
conv2d_7 (Conv2D)	(None, 28, 28, 16)	2320
flatten_1 (Flatten)	(None, 12544)	0
dense_3 (Dense)	(None, 24)	301080
dense_4 (Dense)	(None, 12)	300
dense_5 (Dense)	(None, 3)	39

Total params: 305,117
Trainable params: 305,117
Non-trainable params: 0

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_8 (Conv2D)	(None, 254, 254, 24)	672
max_pooling2d_6 (MaxPooling2	(None, 127, 127, 24)	0
conv2d_9 (Conv2D)	(None, 125, 125, 24)	5208
max_pooling2d_7 (MaxPooling2	(None, 62, 62, 24)	0
conv2d_10 (Conv2D)	(None, 60, 60, 64)	13888
max_pooling2d_8 (MaxPooling2	(None, 30, 30, 64)	0
conv2d_11 (Conv2D)	(None, 28, 28, 64)	36928
flatten_2 (Flatten)	(None, 50176)	0
dense_6 (Dense)	(None, 24)	1204248
dense_7 (Dense)	(None, 12)	300
dense_8 (Dense)	(None, 3)	39

Total params: 1,261,283
Trainable params: 1,261,283
Non-trainable params: 0

Model: "sequential_3"

Layer (type)	Output Shape	Param #
conv2d_12 (Conv2D)	(None, 254, 254, 12)	336
max_pooling2d_9 (MaxPooling2	(None, 127, 127, 12)	0
conv2d_13 (Conv2D)	(None, 125, 125, 32)	3488
max_pooling2d_10 (MaxPooling	(None, 62, 62, 32)	0
flatten_3 (Flatten)	(None, 123008)	0
dense_9 (Dense)	(None, 24)	2952216
dense_10 (Dense)	(None, 12)	300
dense_11 (Dense)	(None, 3)	39

Total params: 2,956,379
Trainable params: 2,956,379
Non-trainable params: 0

Model: "sequential_4"

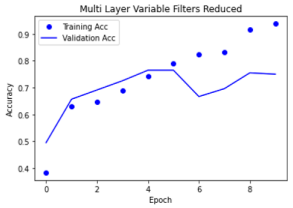
Layer (type)	Output Shape	Param #
conv2d_14 (Conv2D)	(None, 254, 254, 32)	896
max_pooling2d_11 (MaxPooling	(None, 127, 127, 32)	0
conv2d_15 (Conv2D)	(None, 125, 125, 64)	18496
max_pooling2d_12 (MaxPooling	(None, 62, 62, 64)	0
flatten_4 (Flatten)	(None, 246016)	0
dense_12 (Dense)	(None, 24)	5904408
dense_13 (Dense)	(None, 12)	300
dense_14 (Dense)	(None, 3)	39

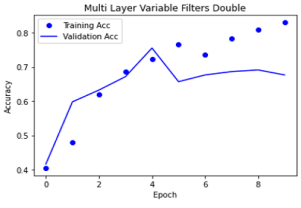
Total params: 5,924,139
Trainable params: 5,924,139
Non-trainable params: 0

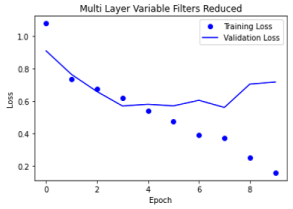
Model: "sequential_5"

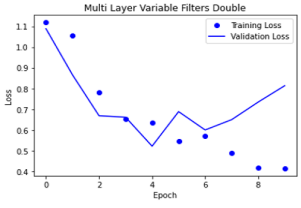
Layer (type)	Output Shape	Param #
conv2d_16 (Conv2D)	(None, 254, 254, 12)	336
max_pooling2d_13 (MaxPooling	(None, 127, 127, 12)	0
conv2d_17 (Conv2D)	(None, 125, 125, 12)	1308
max_pooling2d_14 (MaxPooling	(None, 62, 62, 12)	0
conv2d_18 (Conv2D)	(None, 60, 60, 32)	3488
max_pooling2d_15 (MaxPooling	(None, 30, 30, 32)	0
conv2d_19 (Conv2D)	(None, 28, 28, 32)	9248
max_pooling2d_16 (MaxPooling	(None, 14, 14, 32)	0
conv2d_20 (Conv2D)	(None, 12, 12, 32)	9248
flatten_5 (Flatten)	(None, 4608)	0
dense_15 (Dense)	(None, 24)	110616
dense_16 (Dense)	(None, 12)	300
dense_17 (Dense)	(None, 3)	39

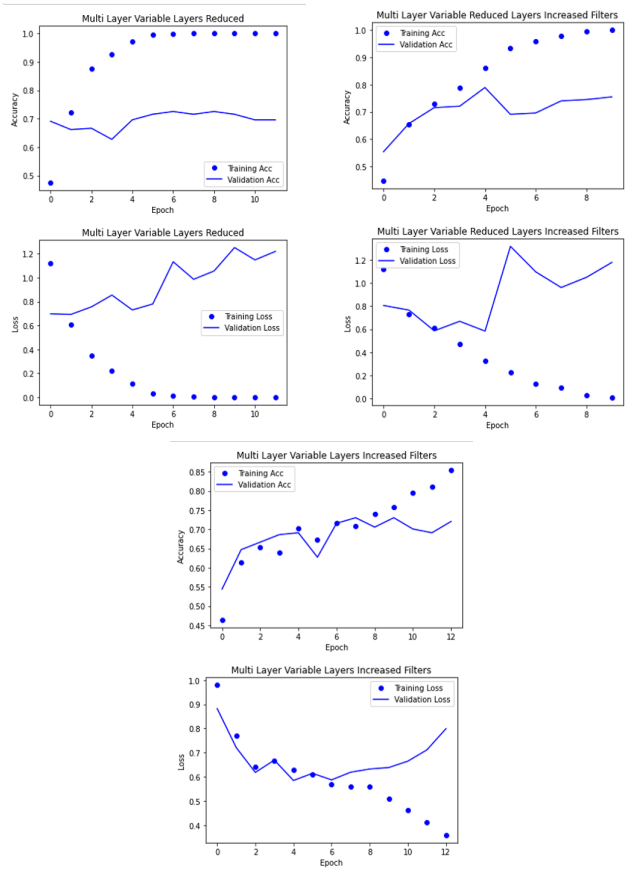
Total params: 134,583
Trainable params: 134,583
Non-trainable params: 0











The metrics of interest are contained in the table in comparison with the initial model developed.

Model	Params	Loss	Acc	Val Loss	Val Acc	Test Loss	Test Acc
0	616,855	0.3897	0.8361	0.4653	0.8333	0.7365	0.7739
1	305,117	0.3736	0.8328	0.5611	0.6961	0.6001	0.7696
2	1,261,283	0.6336	0.7230	0.5212	0.7549	0.5779	0.7348
3	2,956,379	0.6101	0.7230	0.6934	0.6618	0.7433	0.7304
4	5,924,139	0.6113	0.7295	0.5862	0.7157	0.7550	0.7261
5	134,583	0.5693	0.7164	0.5874	0.7157	0.6035	0.7174

The results of the each model's evaluation on the validation and test sets indicate that the number of parameters, which may be influenced by the number of filters per layer and layers in the network, affect the model's performance. Model 0 and Model 1 yield a comparable test accuracy. These two models contain the same number of layers, but differ in that each convolution layer in Model 1 has half the filters of the equivalent layer in Model 0. Model 1 yielded

a lower loss on the test data than Model 0 even with both models having about the same accuracy on the training set. Model 2 was constructed similarly to Model 1 with the exception that each layer has twice the number of filters as the equivalent layer in Model 0. The test loss and validation loss were lower in for Model 2 than in Model 1 and the accuracy between the validation set and the test set are comparable. The comparisons suggest that Model 2 may be the ideal model to select for further observations as it may have a suitable number of parameters to learn from the dataset.

The performance effect of the number of layers is present within Model 3, 4 and 5. Model 3 and Model 4 contain half the number of convolution layers as Model 0 with varying filters. The validation loss and test loss were the greatest with these two models indicating these models tend to overfit the training data much sooner than the previously observed models. The layers themselves may not have been the only contributing factor as MaxPooling was utilized after each layer to reduce the number of parameters in the model. Models 3 and 4 differ in the number of filters and these perform more poorly on the validation and test set than the models with less parameters excluding Model 5. Model 5 is based on Model 0 with an additional layer and MaxPooling yielding a model with the least number of parameters. The results of Model 5 indicate that the model lacks the number of parameters required to learn from the data.

It was observed that more complicated models tend to overfit the training data and do so much more quickly than simpler models. This overfitting of the training data yields poorer evaluations on the validation and test sets due the model's inability to generalize the data. Simpler models should be constructed such that the accuracy and loss for unknown samples are approximately the same for the seen samples. Although a simpler model is desired, the model must contain enough complexity to learn from the data as a model that is overly simple also performs poorly.