CATS vs DOGS

A Machine Learning Algorithm

CAP 4630: Intro to Artificial Intelligence - Final Project Report

Florida Atlantic University - Summer 2020

– Dr. Oge Marques –

AUTHORS:

Efrem Yohannes – Mason

Beatriz Cangas – Perez

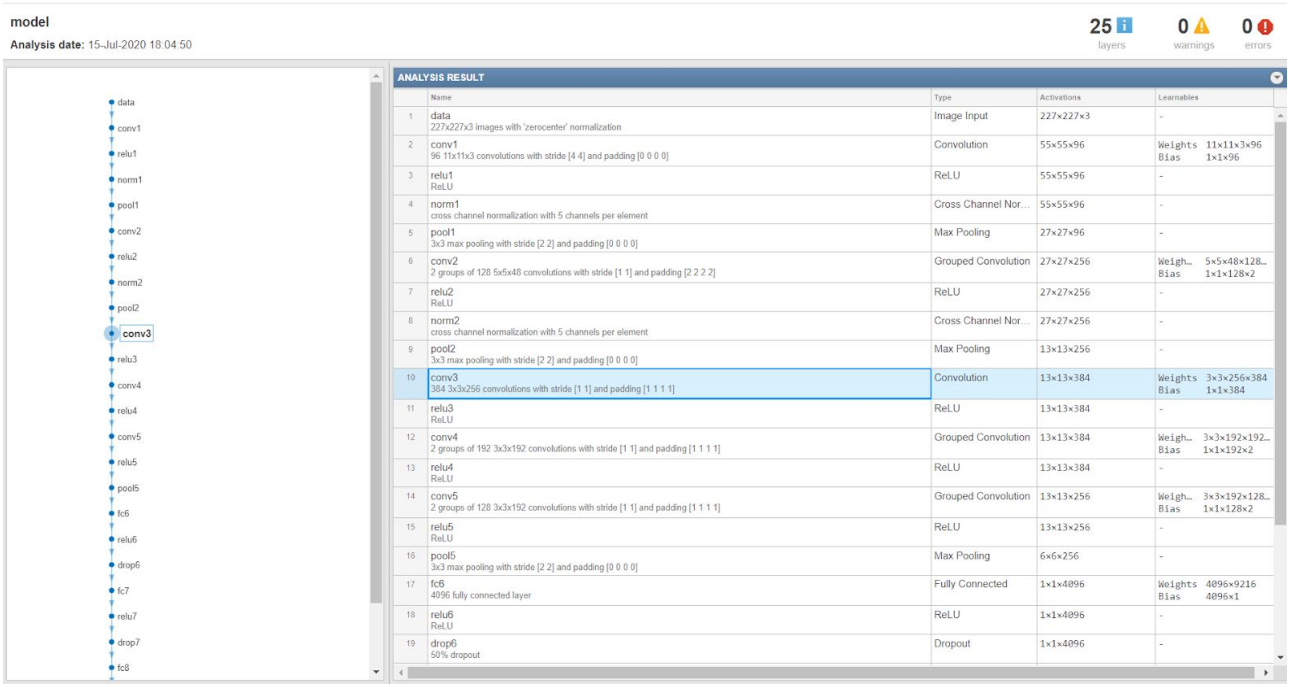
TABLE OF CONTENTS

Setup And Configuration

Section Notes:

- Files added to MATLAB and the pathway was adjusted with “Set Path”
- “Deep Learning Toolbox Model for AlexNet Network” package was downloaded via the link that was provided in the code.
- 1.1: At this point I was able to load the pre-trained AlexNet neural network via
    - model = alexnet;

➔ **1.3: Inspection of the CNN’s layers**



	Name	Type	Activations	Learnables
1	data 227x227x3 images with 'zerocenter' normalization	Image Input	227×227×3	-
2	conv1 96 11x11x3 convolutions with stride [4 4] and padding [0 0 0 0]	Convolution	55×55×96	Weights 11×11×3×96 Bias 1×1×96
3	relu1 ReLU	ReLU	55×55×96	-
4	norm1 cross channel normalization with 5 channels per element	Cross Channel Nor…	55×55×96	-
5	pool1 3x3 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	27×27×96	-
6	conv2 2 groups of 128 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2]	Grouped Convolution	27×27×256	Weigh… 5×5×48×128… Bias 1×1×128×2
7	relu2 ReLU	ReLU	27×27×256	-
8	norm2 cross channel normalization with 5 channels per element	Cross Channel Nor…	27×27×256	-
9	pool2 3x3 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	13×13×256	-
10	conv3 384 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	13×13×384	Weights 3×3×256×384 Bias 1×1×384
11	relu3 ReLU	ReLU	13×13×384	-
12	conv4 2 groups of 192 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]	Grouped Convolution	13×13×384	Weigh… 3×3×192×192… Bias 1×1×192×2
13	relu4 ReLU	ReLU	13×13×384	-
14	conv5 2 groups of 128 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]	Grouped Convolution	13×13×256	Weigh… 3×3×192×128… Bias 1×1×128×2
15	relu5 ReLU	ReLU	13×13×256	-
16	pool5 3x3 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	6×6×256	-
17	fc6 4096 fully connected layer	Fully Connected	1×1×4096	Weights 4096×9216 Bias 4096×1
18	relu6 ReLU	ReLU	1×1×4096	-
19	drop6 50% dropout	Dropout	1×1×4096	-

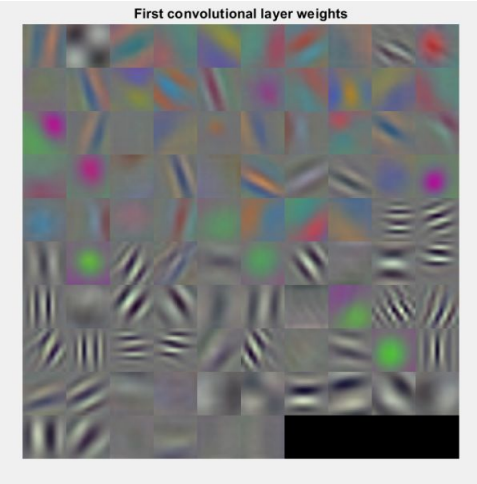
Question 1: What type of preprocessing is performed by the auxiliary function readAndPreprocessImage ?

The main task of this function is formatting the images in a way that AlexNet is able to process. This is achieved by resizing the images to the required 227-by-227 for the CNN and replicating each image three times to create an RGB image in case any of the images are in grayscale.

Section Notes:

- 2.2: At this point the images are pre-processed
  - AlexNet can only process RGB images that are 227-by-227
  - imds.ReadFcn = @(filename)readAndPreprocessImage(filename);
  - imds.ReadFcn is called every time an image is read from the ImageDataStore
  - calls function readAndPreprocessImage() where the following line of code resizes the image to the required size for the CNN
  - Iout = imresize(I, [227 227]);
  - And create an RGB image
  - I = cat(3,I,I,I);

➔ **1.4 Inspecting the weights**



Question 2: What can you say about the montage with network weights for the first convolutional layer (above)?

The montage is representative of the basic image features from the layers at the beginning of the network. The first few layers of the CNN are some of the only layers which are suitable for image feature extraction and capture the basic image features such as edges and blobs. These primitive features are later processed by the deeper network layers to form higher level image features that are better suited for recognition tasks.

Section Notes:

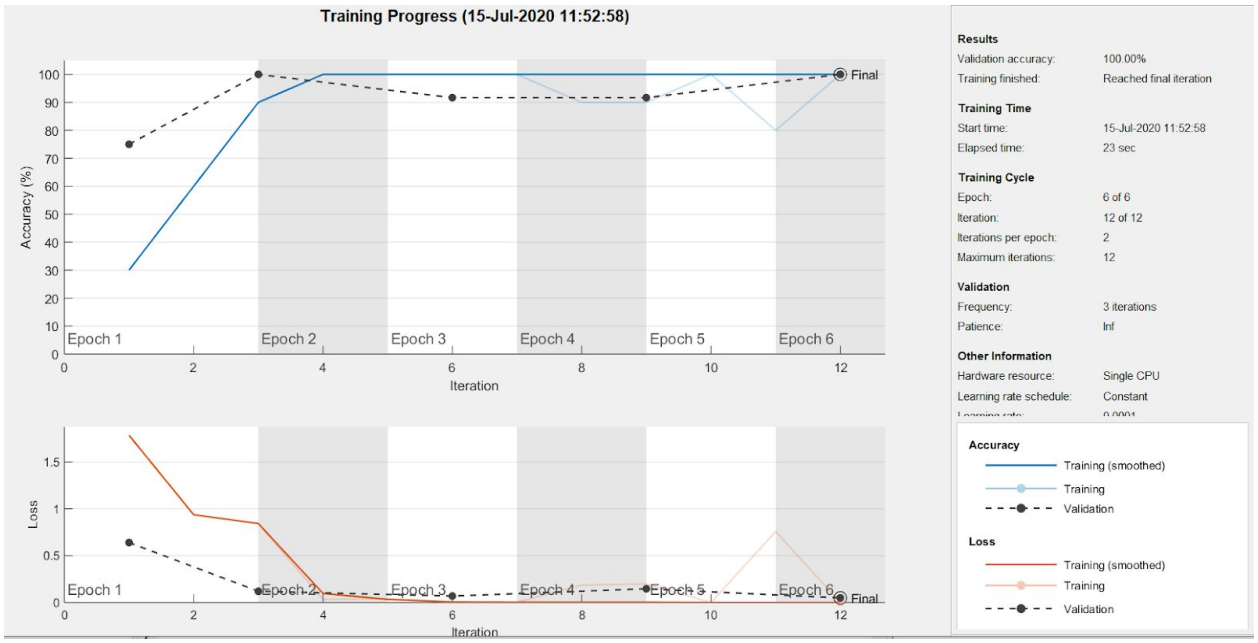
- in Section 2.1: the dataFolder file path was adjusted to include the new file path
  - dataFolder = './CAP_4630_FINALPROJECT_SUMMER2020/data/PetImages';
  - and the categories (cat, dog) were the filenames inside PetImages folder
- 2.3: Divides the data into a training set and validation set
  - In the simple example the data is split into 70% trainingSet
  - And 30% validationSet

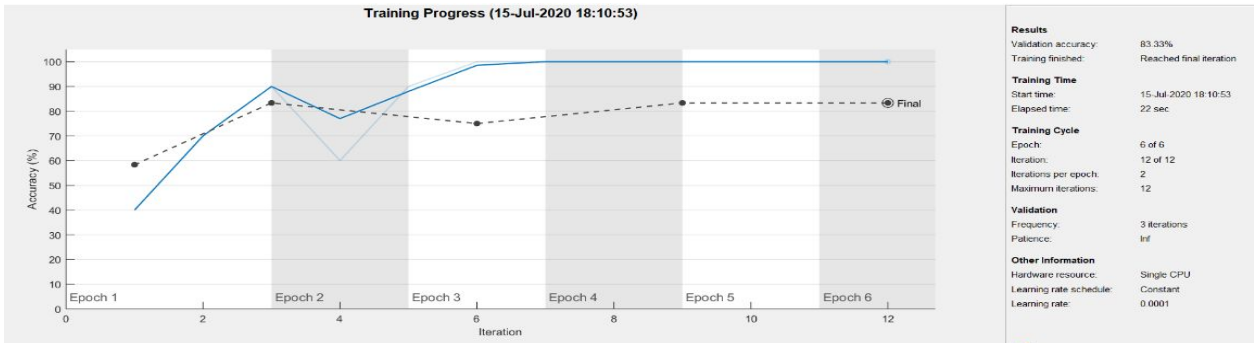
Question 3: How many images are there in each set (training / validation)?

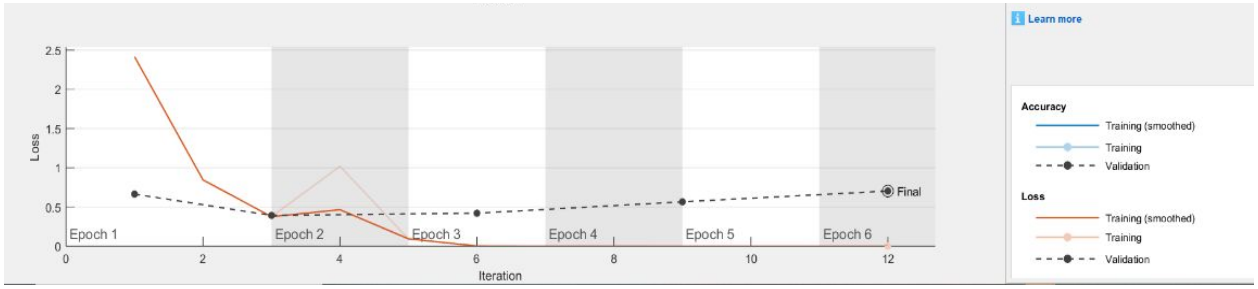
Resulting from the following line of code,
"[trainingSet, validationSet] = splitEachLabel(imds, 0.7, 'randomized');"
The training set takes 70% of the images and the validation set takes the remaining 30%. Therefore, in the 'dog' file of 20 total images, 14 images are in the training set and 6 are in the validation set. Similarly, in the 'cat' file of 20 total images, 14 images are in the training set and 6 are in the validation set.

Section Notes:

- Part 3: we begin the transfer learning process
- 3.1: We freeze all the layers except the last 3
- We assign the number of classes equal to 2 (1. Cats and 2. Dogs)
- Then we configure the last 3 layers to our problem
- 3.2: We now configure the training options
- 3.3: We are Retraining the network
    - This was done twice with far different results
    - One run resulted in 83.33% accuracy
    - While the other resulted in 100% accuracy





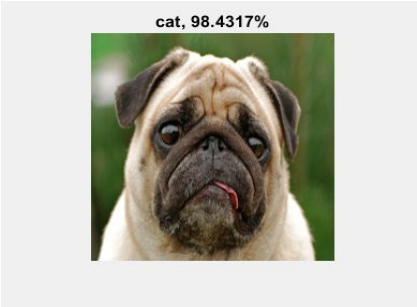


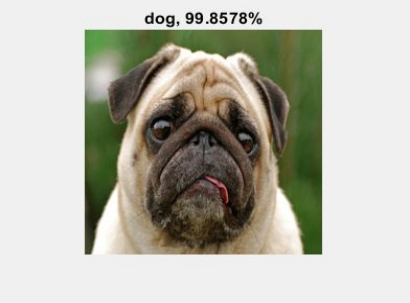
Question 4: Is the validation accuracy (in my example 66.67%) acceptable for a two-class classifier? Why (not)? If not, what could be the problem?

In the first experiment we achieved a validation accuracy of 83.33% which we believe was not acceptable. However, in our second test we received a validation accuracy of 100% which was an optimal result. We think the reason for this problem (on the first run) is an overfitting of the neural network. After the 3rd Epoch even though the accuracy is holding relatively steady at a high degree, the validation loss begins to rise slightly. This could be improved by stopping the network's training after the 3rd Epoch.

Section Notes:

- 3.4 & 3.5: We are classifying the validation images and calculating the accuracy on the validation set
  - As shown above
  - One run resulted in 83.33% accuracy
  - While the other resulted in 100% accuracy
- 3.6: We are testing our new neural network on unseen images
  - File path was changed to './CAP_4630_FINALPROJECT_SUMMER2020/'...'.jpg';
  - For the respective dog.jpg and cat.jpg images provided
- Did this several times and got different results



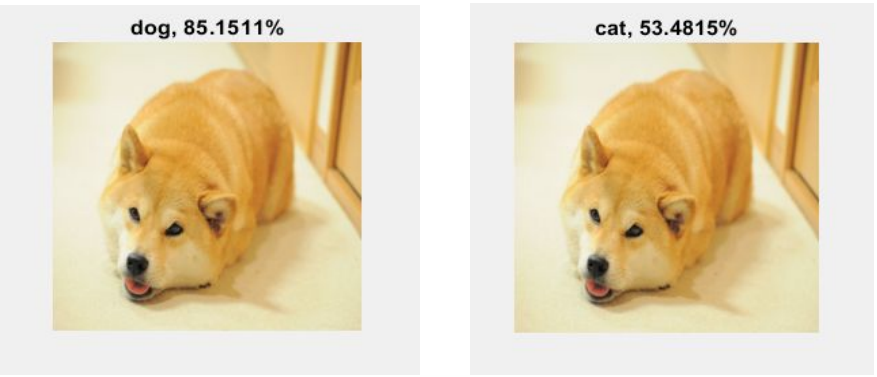


Dog:

First run predicted a cat with 98.43% accuracy
Second run predicted dog with 99.86% accuracy
Third run predicted dog with 99.99% accuracy

Cat:

First run predicted a cat with 99.99% accuracy
Second run predicted a cat with 100% accuracy
Third run predicted a cat with 99.99% accuracy



Doge:

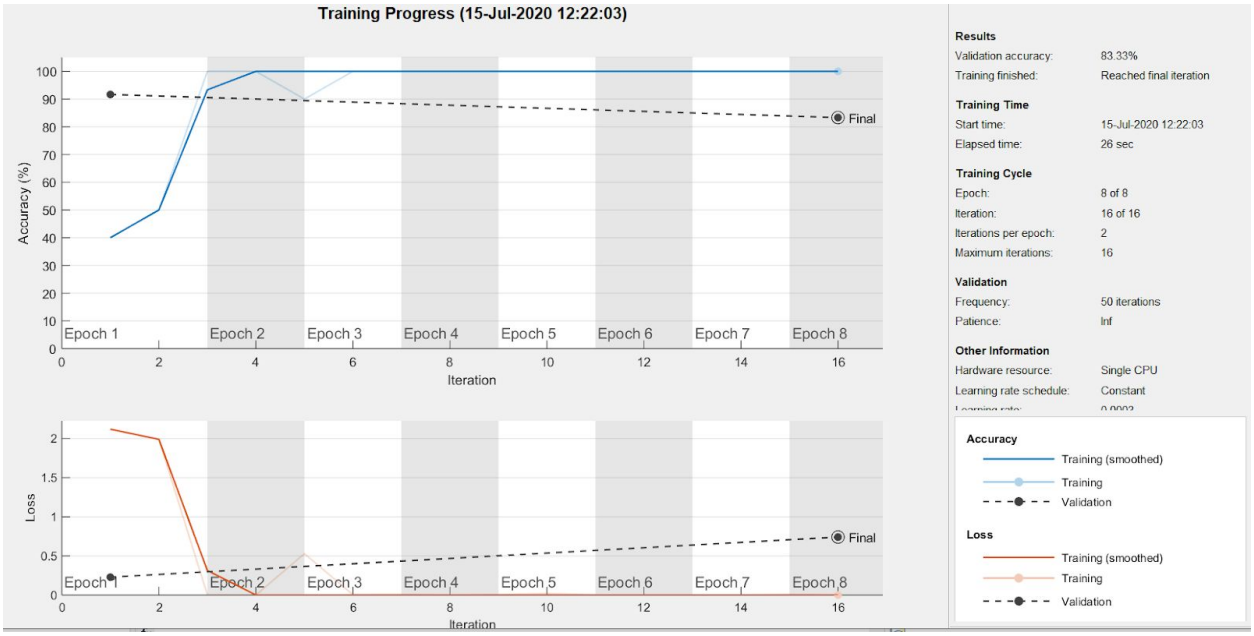
First run predicted a dog with 85.15% accuracy
Second run predicted a cat with 53.48% accuracy

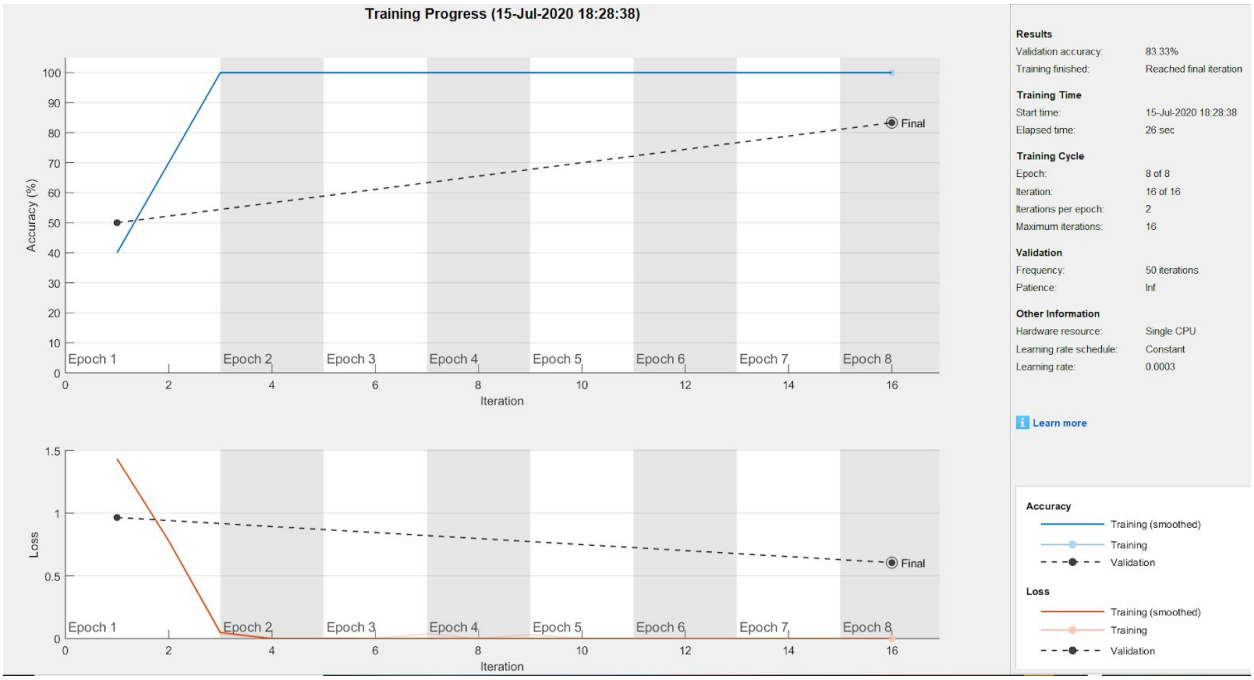
Question 5: Did your classifier recognize 'Doge' as a dog? If not, can you tell why?

On the first run ‘Doge’ was indeed recognized as a dog with 85.12% validation accuracy. However, on a second test ‘Doge’ was then recognized as a cat with 53.48% validation accuracy. Because of the limited sample size of dogs and cats the network was unable to make enough differentiations in size, color, tails, etc. This may have caused an overfitting in a particular dimension, resulting in a misclassification of certain unseen images.

Section Notes:

- Part 4: Data augmentation
- 4.1: assigns a pixel range and scale then randomly flips and translates images
- 4.2: displays the number of observations in the training and validation sets
  - These are the number of images in training set = 28
  - And the number of images in the validation set = 12
- 4.3: We are now retraining the network with the augmented data
  - This was done twice
  - Surprisingly both times yielded the same results 83.33% accuracy





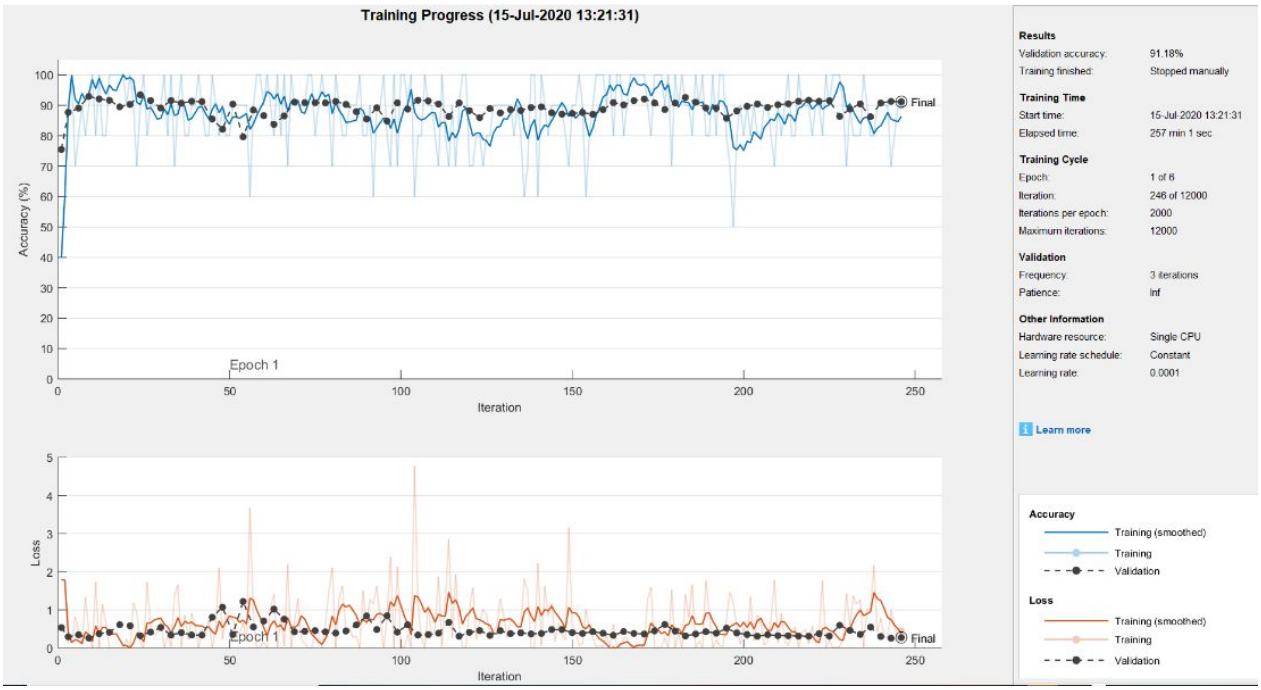
Question 6: Is the validation accuracy (in my example 75%) better than before? What could be the reason(s) behind such (modest) improvement? How could you improve it even further?

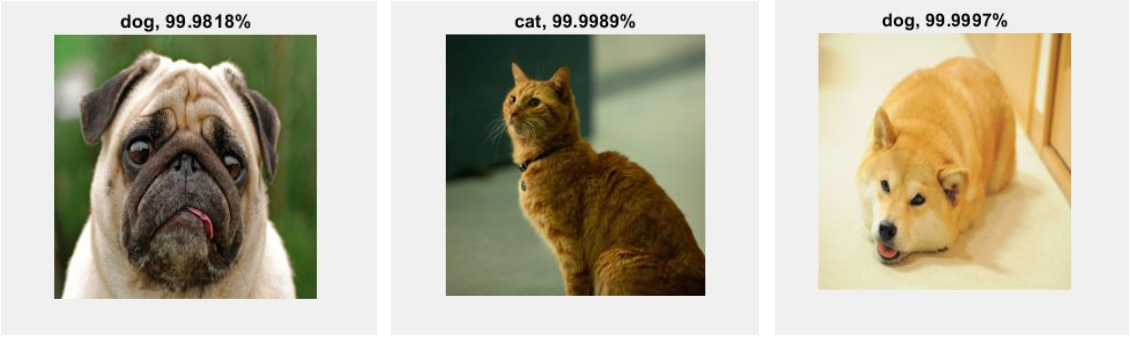
Surprisingly enough, we did not see any improvements in the performance of the augmented dataset. There are no improvements to report on, but we believe it is due to the relatively small dataset that was used in the experiment.

Section Notes:

- Part 5: Experimenting on Larger data set
- **First attempt** at the large dataset
  - The main problem I faced here was the time required to train the network.
  - I allowed training to run for more than 4 hours and was unable to complete a reasonable portion of even the first Epoch.
  - Therefore, the training was stopped early, and the results are as follows.
- With an accuracy of 91.18%
- And the retesting on unseen images resulted in
- Even though the neural network was not completely trained, doge was predicted as a dog at a much greater accuracy than the other two previous predictions

➔ **Large Dataset – First attempt manually stopped due to large training time**



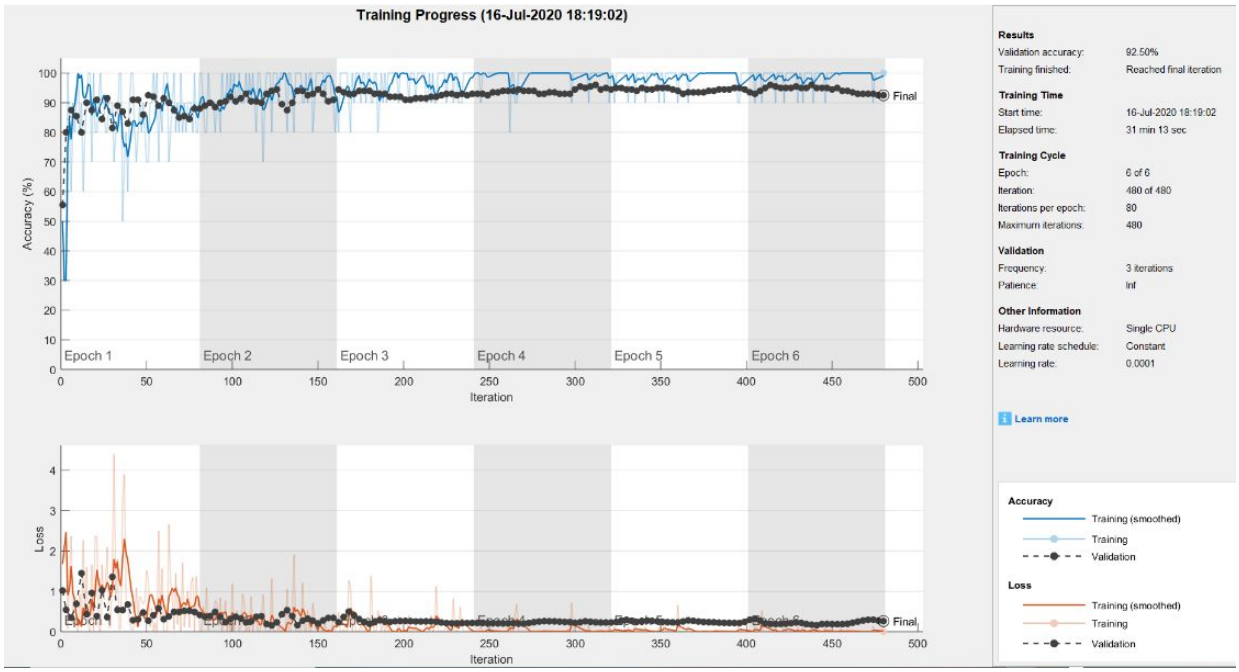


dog predicted as dog with 99.98% accuracy
cat predicted as cat with 99.99% accuracy
doge predicted as dog with 99.99% accuracy

Section Notes:

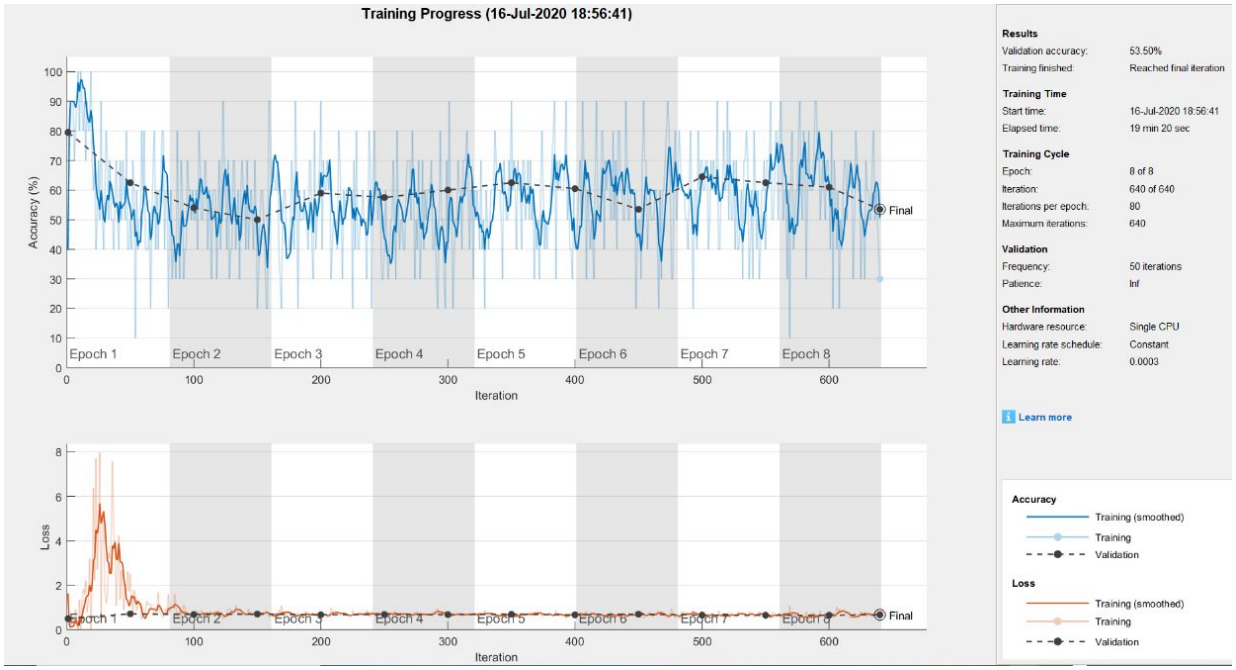
- **Second attempt** at the large dataset
- Due to our groups inability to process 25,000 images within a reasonable time frame we limited the images to 1,000 (500 images of cats and 500 images of dogs) and re-trained the network.
- Although this dataset took a reasonably large amount of time, we were able to train the neural network within 31 minutes 13 seconds which is considerably less than the first attempt.
- We were able to achieve a validation accuracy of 92.50%
- We also trained the neural network with the augmented dataset with a validation accuracy of 53.50%

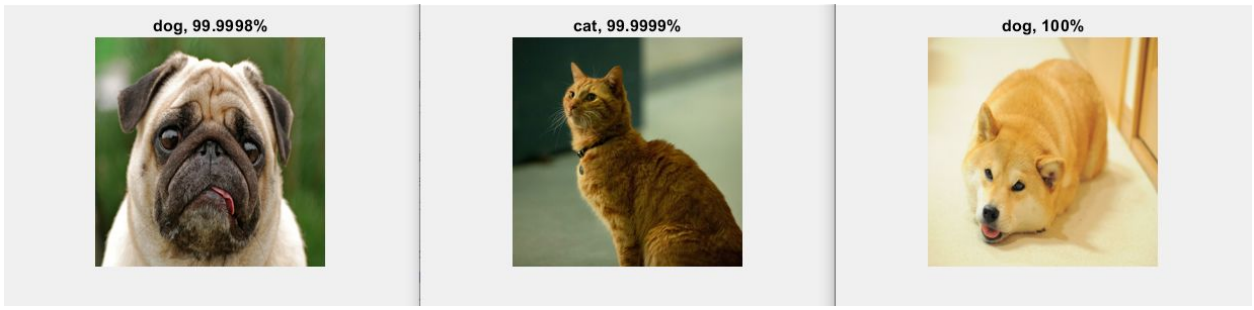
➔ **Large Dataset – Second attempt with 1,000 images**



dog predicted as dog with 99.99% accuracy
cat predicted as cat with 99.99% accuracy
doge predicted as dog with 100% accuracy

➔ **Large Dataset – Second attempt with 1,000 images and Augmented dataset**





dog predicted as dog with 99.99% accuracy
cat predicted as cat with 99.99% accuracy
doge predicted as dog with 100% accuracy

Question 7: Did the validation accuracy improve as a result of using a much larger training dataset? How could you improve it even further?

First Attempt:

Although we had to cut the training short due to the extremely large amount of time that would have been required to complete the training on such a large set. We did see noticeable improvements in the classification of the unseen images. On the small data set, our classifier predicted the dog as cat on one of the runs. Also, 'Doge' was predicted as a dog with 85.15% and a cat with 53.48% accuracy. After training our network on the large dataset we received no misclassifications and most notably, 'Doge' was predicted as a dog with 99.99% accuracy which was a significant improvement over the smaller training set.

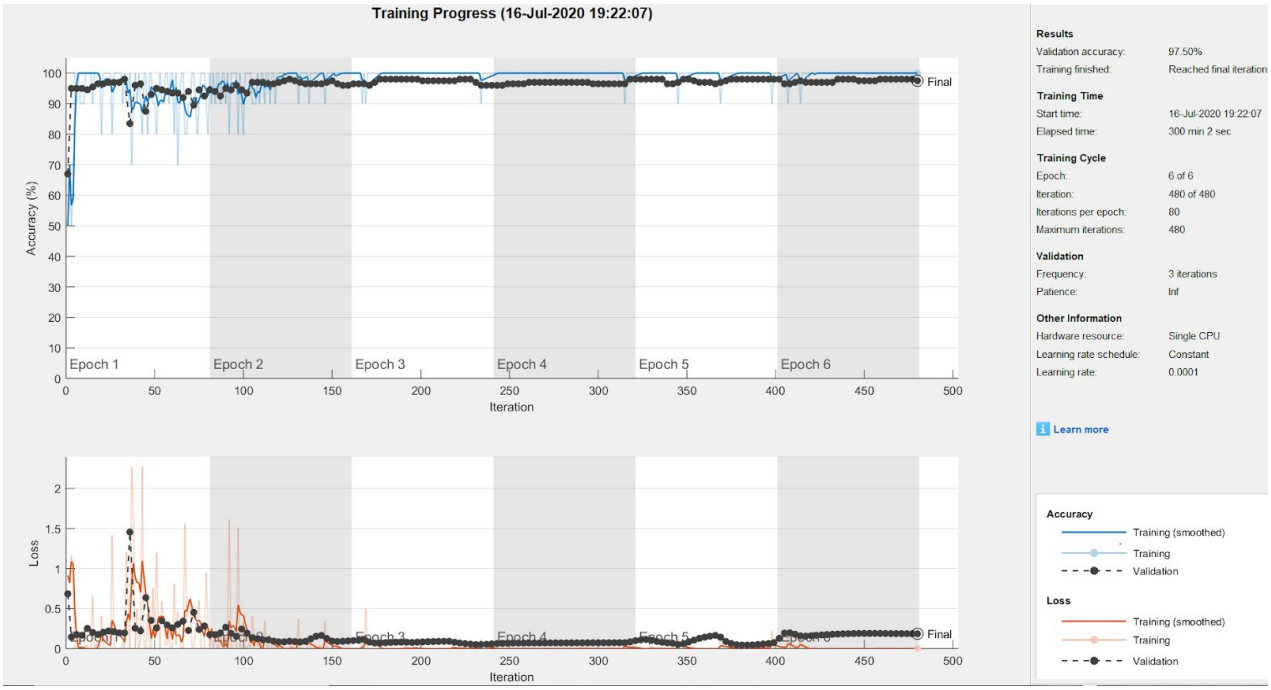
Second Attempt:

Second attempt trained 1,000 images without augmented data performed slightly better than the first attempt with a validation of 92.50% vs 91.18% in the first attempt. However, the augmented set was considerably worse than the other sets with a validation accuracy of 53.50%.

Section Notes:

- Improved Classifier 1: pretrained CNN "VGG-16"
  - This required the support package "Deep Learning Toolbox Model for VGG16 Network"
  - The pre-processing function "readAndPreprocessImageVgg.m" was added
  - VGG16 requires an image size of 224-by-224
  - Training time was much longer than with AlexNet
  - Training time was approximately 300 minutes
  - Resulted in a validation accuracy of 97.50%

➔ **Improved Classifier 1 – pre-trained CNN VGG16**

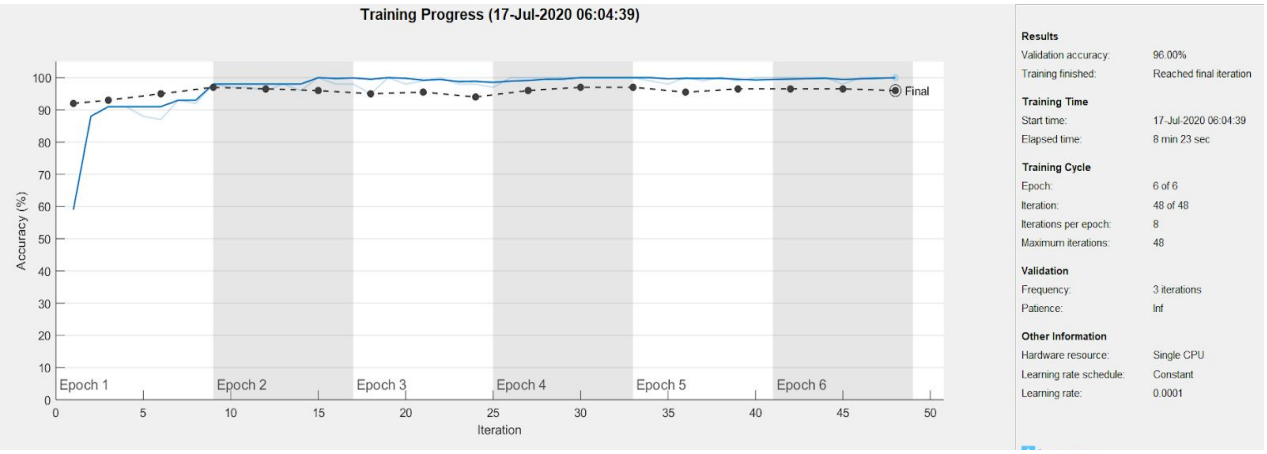


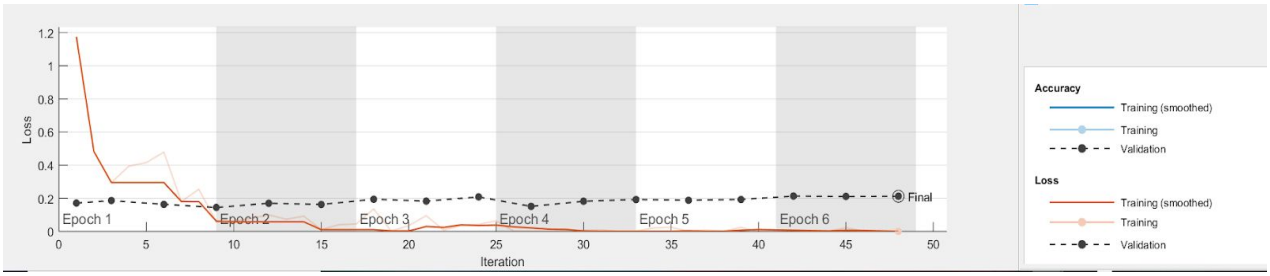
Section Notes:

- Improved Classifier 2: Different Hyperparameters
  - The “adam” optimizer was used
  - The adam optimizer was able to train the set significantly faster than the other two classifiers with a training time of approximately 8 minutes
  - The MiniBatchSize was adjusted to 100
  - Still using pre-trained CNN Alexnet
  - Achieved validation accuracy of 96.00%
- Many variations were attempted with varying results

Optimizer	**MiniBatchSize**	**Accuracy**
sgdm	5	72.50%
sgdm	20	93.0%
sgdm	50	95.00%
sgdm	100	92.50%
rmsprop	10	92.50%
rmsprop	50	94.00%
rmsprop	100	92.50%
adam	10	92.50%
adam	50	94.50%
adam	100	96.00%

➔ **Improved Classifier 2 – “adam” Optimizer MiniBatchSize 100**





➔ **Summary Table**

Classifiers	Model Name	Validation Accuracy	Test Accuracy
Baseline Classifier	AlexNet (includes Augmentation)	92.50%	99.99%
Improved Classifier 1	VGG16 (all other parameters same)	97.50%	99.99%
Improved Classifier 2	AlexNet (Optimizer[adam], MiniBatchSize[100])	96.00%	100%