**Dog Classification Detector**

**Introduction**

In this project, the main goal of this project is to create a dog classification detector using machine learning. As a side goal, we want to also explore ways of improving the accuracy of the detector. We will be using a dataset to train a neural network for the detector. Multiple tests will be done with improvements to the network, so a comparison can be made as to which is the best model and why.

Dataset: <http://vision.stanford.edu/aditya86/ImageNetDogs/main.html>

**Related work**

For this project we used a few references to get an idea of the approach needed to build the network and improve upon it. The references are below.

Building the network: <http://people.csail.mit.edu/khosla/papers/fgvc2011.pdf>   
Improving the network: <http://www.image-net.org/papers/imagenet_cvpr09.pdf>

**Methods**

The kinds of methods we used are in the forms of data augmentation, regularization and normalization. Each kind of methods has a specific technique we used to represent it. The data augmentation technique used is directional flips. The regularization technique used is dropout regularization. The normalizations technique used is batch normalization. All these methods help make the model more accurate.

Data augmentation

**Directional Flips**

The directional flips used are horizontal and vertical flips. By applying this kind data augmentation, we can add onto the training set. The reason why we want to do this is too reduce the need to annotate more data by collecting more samples since we expand our training sample with flips. The logic of having horizontal flip is to be able to classify accurately even if the image is flip like in a mirror. If we had a training set of dogs on looking left and the testing set had a set of dogs looking to the right, there will be many inaccuracies. The flips help improve accuracy in this way because this improves its ability to generalize. Without it the model will be overfitted to images with dogs looking to the left. Using this technique, however, will make the training process longer since there is a larger set of samples. Additionally, we have tested with horizontal and vertical flips separately as a vertical flip may not always be better.

Network Modifications

**Dropout Regularization**

Note that a neuron is a weighted average if its inputs while will be passed through an activation function where a value is outputted within a range of values. This method essentially ignores randomly selected neurons during the training process. Any weight updates will not be reflected onto the ignored neurons. By doing this, the nodes that do get updated will be tuned to accommodate specific features therefore providing some specialization. This can cause overfitting if the nodes end up being too reliant on these special features. Since the rest of neurons need to make predictions without the ignored neurons, there is some independence in the neural net. The neural net will be less sensitive to specific weights of neurons, so it is improved in generalization causing less overfitting.

**Batch Normalization**

This is a function that is applied to layers in the neural net that normalizes the next layer’s inputs over a mini-batch. This normalizes the output that comes out of an activation function. This process of normalization has 4 values, mean, standard deviation and 2 arbitrary values that are set. These values are trainable and will be optimized during the training process. This allows the weights to be balanced and not be overly influenced by abnormally high/low values. The later layer’s will be constrained to the balanced mean and standard deviation thus not allowing the output to drift off. So, this function as a whole basically improves the stability of the network by not letting abnormal weights cascade throughout the network.

Network Architectures

**FCN with three dense layers**

The first model we tested was a basic fully connected network. It had three layers of sizes 128, 64, and then 6 respectively. The first two layers used ReLU activation and the last layer used SoftMax.

**CNN with 5 single-convolution groups**

The next model we tried had 5 single-convolution groups. Each group was composed of a convolution with a kernel of size 3, followed by batch normalization, and then max pooling with a window of size 2. The number of kernels in the convolution in each group were 32, 64, 64, 96, and 32 respectively. After the 5 single-convolution groups, we applied dropout with a rate of 0.3 and had two fully connected layers with of size 64 and 32 with ReLU activation and dropout of 0.2 in between layers. The final layer had a SoftMax at the end.

**CNN with 3 double-convolution groups**

The next model we tried had 3 double-convolution groups. Each group was composed of two convolutions with a kernel of size 3, followed by batch normalization, and then max pooling with a window of size 2. The number of kernels in the convolution in each group were 32, 64, and 32 respectively. After the convolution groups, we applied dropout with a rate of 0.3 and had two fully connected layers with of size 64 and 32 with ReLU activation and dropout of 0.2 in between layers. The final layer had a SoftMax at the end.

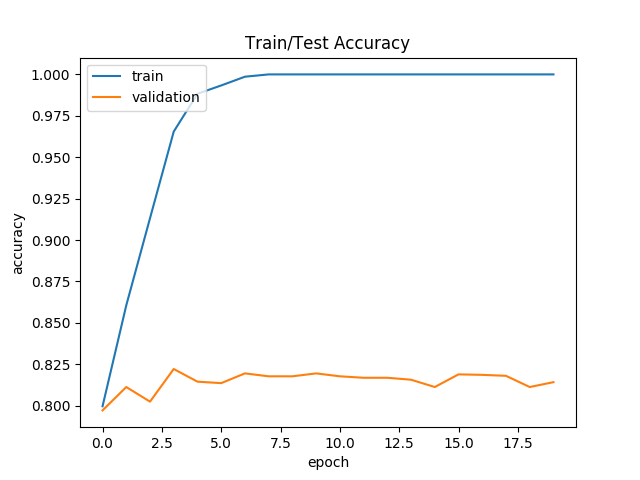
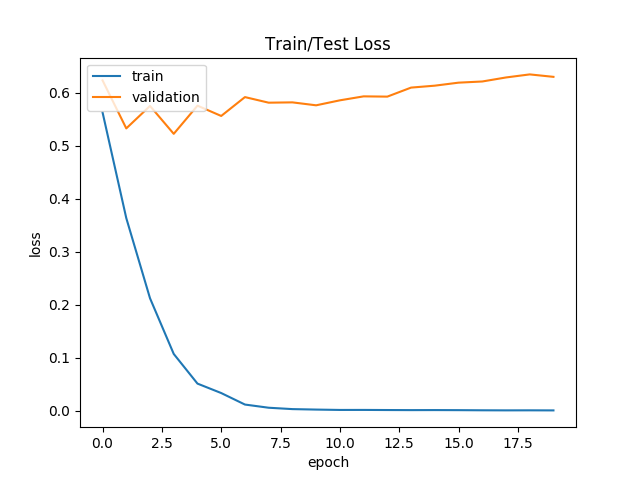
**CNN with 1 double-convolution groups**

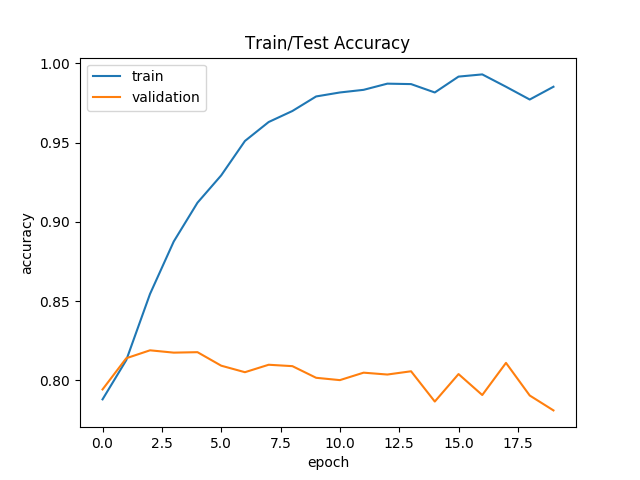
This model was identical to the previous model, except without the second and third convolution groups.

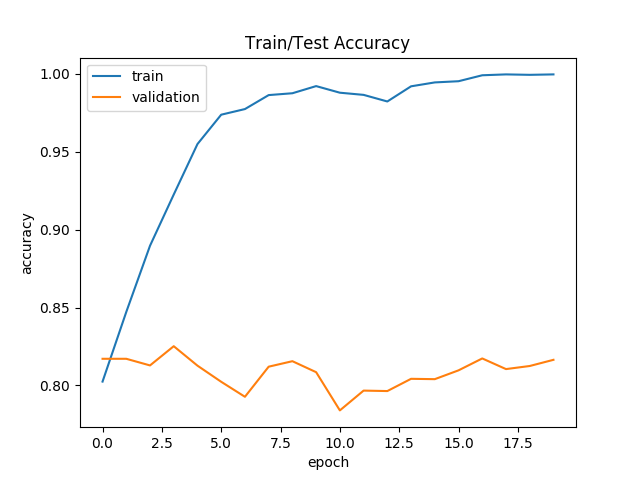
**Experiments**

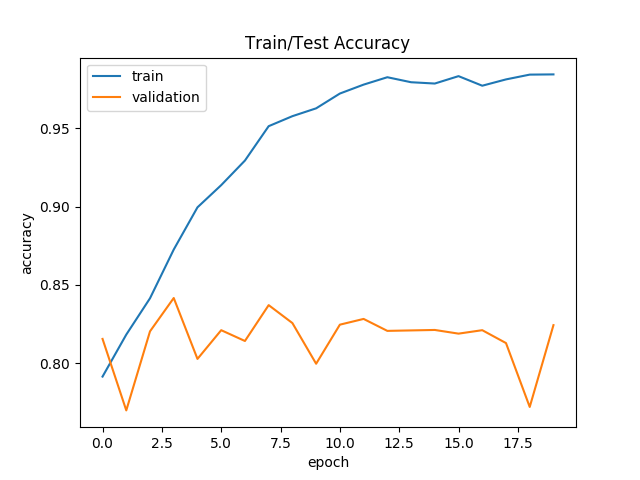
Testing was done with Python with packages that can be found in the requirements.txt file of the zip file. The important packages used are Keras as the library for the functions of features to implement the network and NumPy to generate the graph. We used 20 epochs for all the tests. Below are the accuracy and loss graphs of each test:

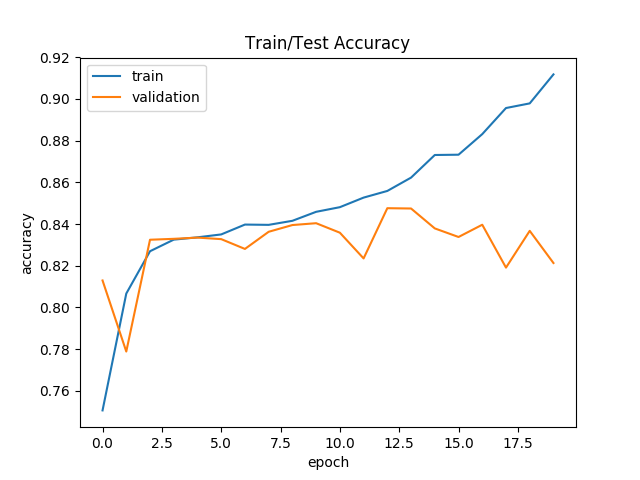
Model: CNN with 5 single-convolution groups and no dropout



Data Augmentation: None  
Analysis: This model obviously overfit heavily. This could be because of the small number of data points or the lack of methods of reducing overfitting. Still the model achieved around 82%  
  
Model: CNN with 5 single-convolution groups with dropout  
  
Data Augmentation: None  
Analysis: This model over fit less at first but it still ended up overfitting over time. The only difference between this model and the previous model is that this model used dropout. This model reached 82% accuracy.

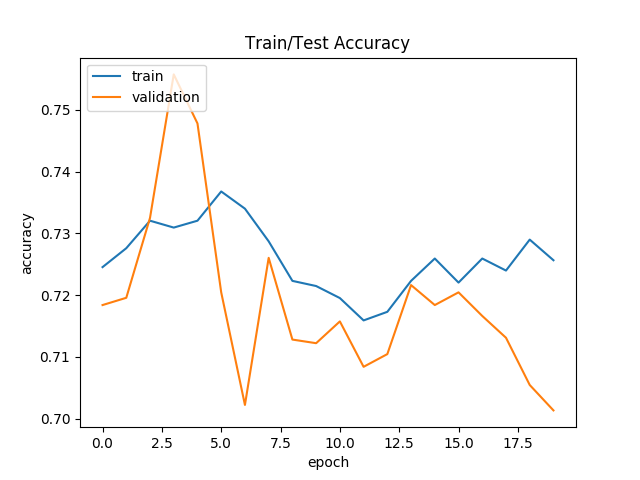
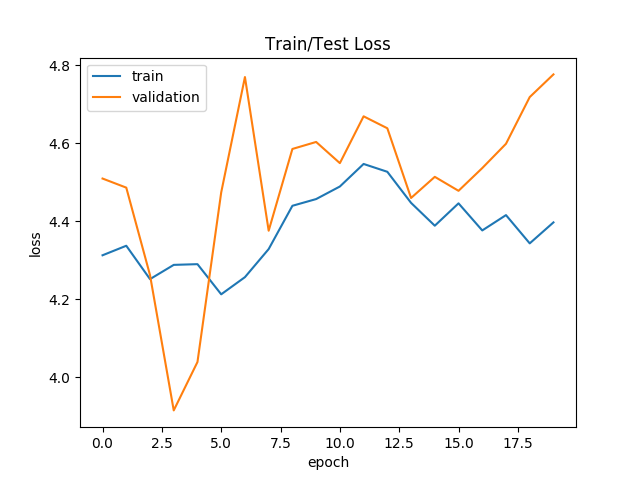
Model: CNN with 5 single-convolution groups with dropout  
  
Data Augmentation: Up-Down Flipped and Right-Left Flipped  
Analysis: This model had both dropout and data augmentation applied. This model also failed to perform any better. One hypothesis we had was the that vertical data flipping was actually hurting the model’s ability to learn as upside-down dogs are less common as images.

Model: CNN with 5 single-convolution groups with dropout  
  
Data Augmentation: Right-Left Flipped  
Analysis: This model had both dropout and data augmentation but because of our previous theory about vertical flipping not helping the model, we only performed horizontal flipping. The Model performed better than previous models, reaching 84%.  
  
Model: CNN with 3 double-convolution groups and no dropout



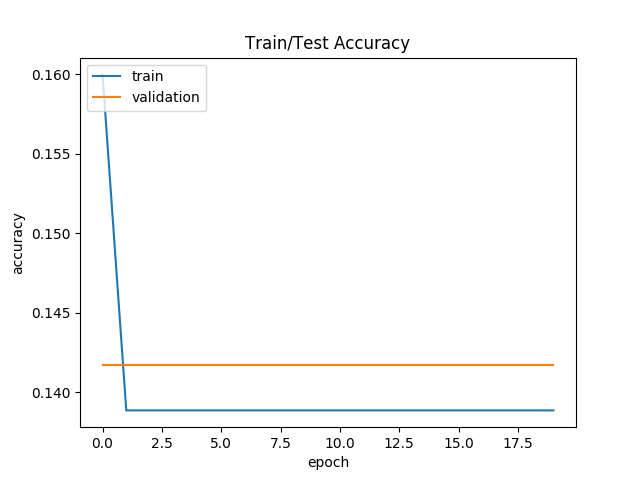
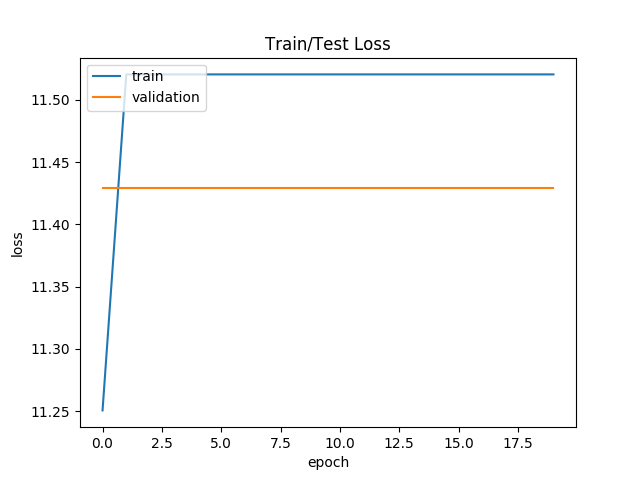
Data Augmentation: Only Right-Left Flipped  
Analysis: We decided to try using a different type of convolutional group, so we added another convolutional layer before applying batch normalization. The model performed about as well as the previous model, but it was a little more consistent. We decided the previous model met our needs.

Model: CNN with 1 double-convolution group and no dropout



Data Augmentation: Only Right-Left Flipped  
Analysis: This model had far fewer layers than the other models. That might be why it didn’t perform very well at all. It reaches 75% accuracy before collapsing.

Model: FCN with three dense layers with no Dropout and no Batch Normalization



Data Augmentation: None  
Analysis: This was just a test to see if we could use simple fully connected networks. It also didn’t use batch normalization or dropout. The network didn’t learn at all, so we decided not to continue down this route.

**Conclusion**

Using the methods from data augmentation, regularization and normalization will improve accuracy greatly. From our results using CNN with 5 single-convolution groups with dropout with only horizontal flipping of images has shown to have the best accuracy of 84%.