Cat vs Dog Image Classifier Neural Network Comparison

Alex Bailey, Anant Pathak, Payal Deora, William Olsen

CS 545 Machine Learning Final Project Report

Introduction

Is this picture a cat or a dog? That is the question that we wish to answer. But more importantly we want to compare how two types of neural networks handle this type of data. To this end we implemented two different machine learners, a multilayer perceptron and a convolutional neural network to classify the image.

Background

There was a machine learning contest, held by Kaggle, that asked the participants to create a neural network that could distinguish between a cat and dog photo, which is where we obtained the data from. To implement our neural networks we use Keras, a convenient Python learning library.

Multi-Layer Perceptron

A Multilayer-Layer Perceptron (MLP) is a grouping of perceptrons
that are densely connected to all nodes in the next layer. There are often one or more hidden layers that help the MLP to determine what features are truly important. While a single

perceptron has a linear decision boundary, with an MLP you can get a non-linear decision boundary allowing for more complex classifications.

Convolutional Neural Network

CNNs are regularized versions of multilayer perceptrons. Unlike MLP, CNN can take a 2D/3D matrix as an input, thus images(in RGB e.g. 64X64X3) can be directly given as input, thus requiring less transformation apart from scaling down their resolution. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns.

Implementation

The first step is to preprocess the input images to the neural network that will be given to the program. We scaled down the images to 64x64 and made them grey scale and divided all the pixel values by 255, so they are normalized between zero and one.

MLP

The basic construction of the mlp neural network was one input layer, accepting the image as a flattened vector of 4096 elements, three fully connected hidden layers, comprised of 200 nodes, 100 nodes, and 10 nodes respectively, resulting in roughly 800,000 parameter nodes, specifically chose to be similar to the CNN’s parameter count, and an output layer that classified the result to be either a cat or a dog.

CNN

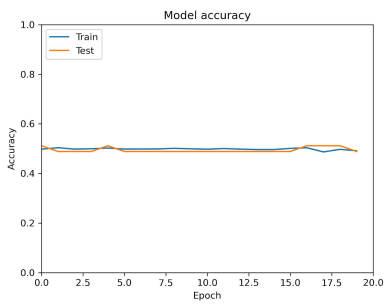
For CNN, we used 2 Convolutional 2D layers, with 64 filters for both the layers and a kernel size of (3,3). After each Conv layer we added a Pooling layer(Max pooling 2D layer) of size (3,3), to decrease the number of parameters used. A dense layer with 64 filters and a dense output layer with 1 node and a batch size of 128.

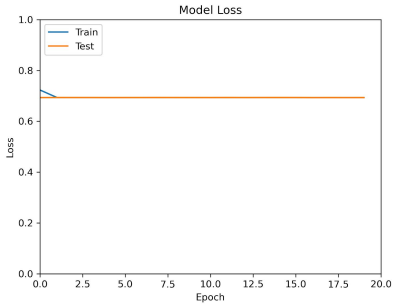
To further decrease the number of parameters, we also tried adding a dropout layer after each conv layer, but that made the model bulky and hence taking more time for training.

Results

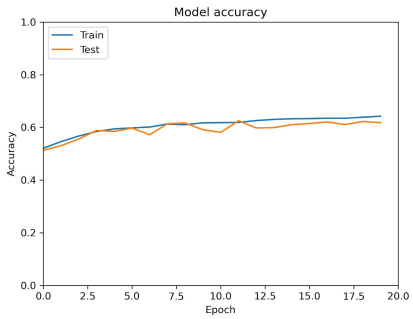
For our experiment, we tried three different combinations of activation function and loss function, mean-squared loss and binary entropy loss functions, and sigmoid and relu activation functions results shown below.

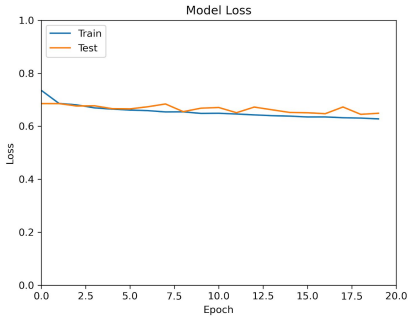
MLP graph results



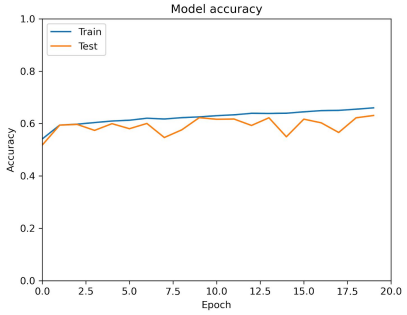


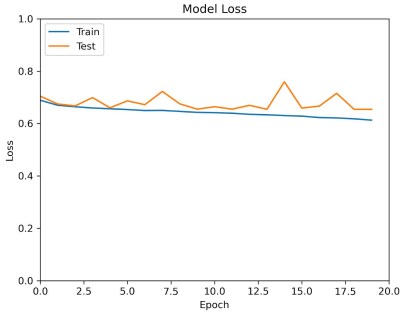
Binary-cross entropy error, relu, 20 epoch





Mean squared error, relu activation, 20 epochs





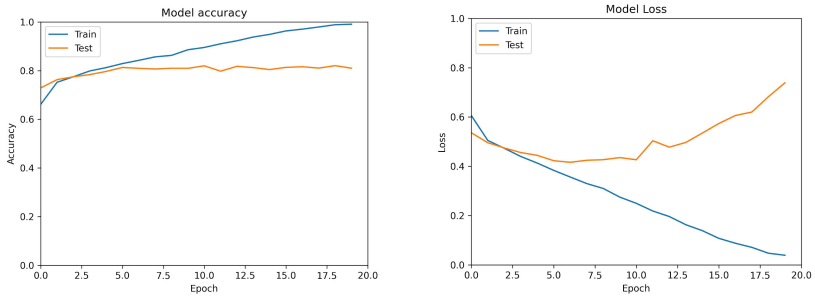
Binary cross entropy error, sigmoid activation, 20 epochs

Multi-Layer Perceptron

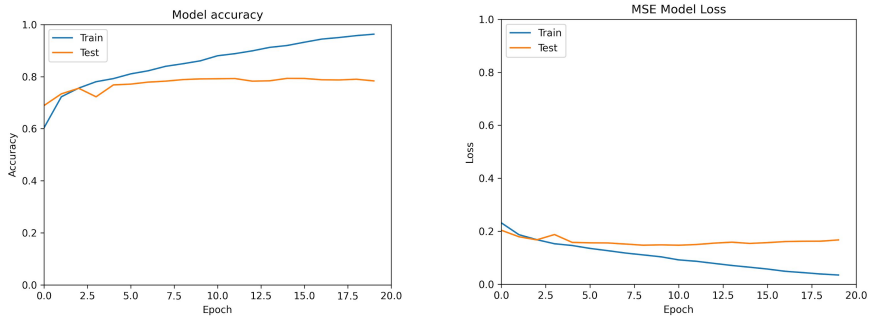
The best training accuracy we could obtain for the MLP was about 60% which is only a little better than flipping a coin. We are suspicious that this poor performance is due to that fact that MLP is not well suited for general image classification. This is due to the input images not being uniform in what is represented. For example there is no constraint that the input images have the cat or dog in the center of the image, nor is there a constraint on which direction the animal is facing.

It is interesting to point out that our MLP does do better than a complete guess. This might be because the cats and dogs are mostly around the center of the photos which give MLP something to extract features from, though extremely poorly.

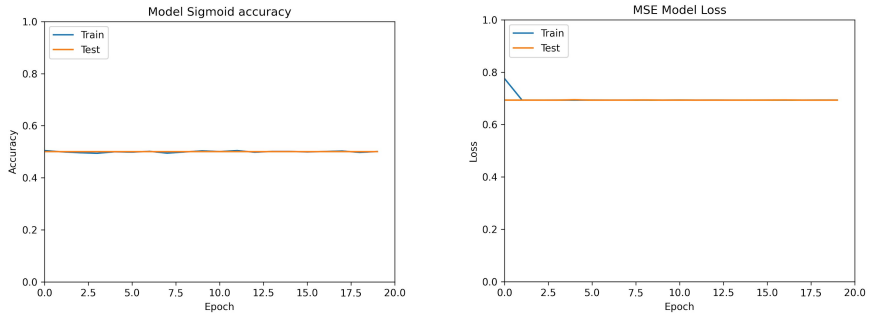
CNN Graph Results



Binary-cross entropy error, relu, 20 epoch



Mean squared error, relu activation, 20 epochs



Binary cross entropy error, sigmoid activation, 20 epochs

Convolutional Neural Network Performance:

The best training accuracy we could obtain for CNN was about 96%. And the best testing accuracy with CNN was around 80%. We got the best results for accuracy with “Relu” as activation function for Convo2D layer, “Binary-crossEntropy” as loss function and “Sigmoid” function for the output layer. But this configuration gave greater testing loss, which means the model’s confidence was low.

Then we tried the model with “MeanSquaredError” as a loss function instead of “Binary-crossEntropy”, there was a decrease in the testing loss, hence increasing the model confidence.

Thirdly, we tried the model with Sigmoid activation function for the Conv layers and “Binary-crossEntropy” as a loss function which gave poor results. This is because of the low convergence performance of the sigmoid function. Also, relu is more computationally efficient to compute than the sigmoid function, since Relu just needs to pick max(0,x) and not perform expensive exponential operations as sigmoid.

Comparison

As can plainly be seen by the graphs, CNN outperforms the MLP in almost every case. While the change in parameters had a fairly significant effect on the output of CNN, the MLP was affected only somewhat. It is interesting to note that both MLP and CNN have one odd case, where the accuracy drops by a noticeable amount. With the MLP, in the case of the binary cross entropy loss function and the relu activation function, the accuracy goes from 60% to 50%. With the CNN, the case of binary cross entropy loss function and the sigmoid activation function the accuracy also drops to 50%. We are unsure of why this happens, though it is interesting that these strange cases don't occur on the same set of parameters.

Conclusion

We created two different neural networks to classify whether an image contains a cat or a dog. To this end we created two neural networks, a MLP and a CNN. The results are unsurprising in that the MLP performed poorly, with 60% accuracy, while the CNN did well, with greater than 80% accuracy. We can also conclude that CNN is better at handling images which have a lot of details other than our object of interest. The next steps for this project would be to focus on improving the CNN to aim for greater accuracy by fine tuning the parameters of the model, for example changing thresholds for activation or adding lots of Convolutional blocks each consisting of varied Conv2D layers , dropout layers, normalization layers & maxpoolign layers. We believe that CNN's results can be improved because other neural networks created for the contest on this data by others have achieved 95% and higher.