Calvin Green
Udacity Machine Learning Nano-Degree
Capstone Project

1. DEFINITION OF CONVOLUTIONAL NEURAL NETWORKS AND THE AMOUNT OF DATA

The following analysis provides an overview of the processes involved in creating an accurate machine learning model using a Convolutional Neural Network architecture. The model should achieve accurate results on testing data without using expensive computational power while capturing necessary features in the data. To examine the data in a responsible matter of time, the experiment contains a Parallel computing CPU architecture for the machine that does not have access to Parallel GPU servers for the framework use of Keras/Tensorflow . Multi-GPU/CPU are highly beneficial in the creation of Convolutional Neural Networks. Modern GPU's hold capabilities to process live Video-Feed,Images/Natural Language Processing as well as using unsupervised learning techniques to help autonomous vehicles navigate streets, human biology for surgery, identity fraud,etc. The necessary module libraries needed for the evaluation will be run on separate CPU/GPU cores to save memory given the volume of the data, while speeding up the overall time on the experiment. It is not always necessary to implement multi-core (GPU/CPU) parallel processing with small datasets but can be very beneficial for larger sets of data.

An important note is that creating an image classifier can be very computationally expensive. Given the dataset (SIGN-LANGUAGE-DIGITS).
****RAM NEED: N_SAMPLES,HEIGHT,WIDTH,32_BIT-FLOATS = (2062*64*64*32) =270,270,464 bits. Given the variables, RAM needed for one instance is 33MB.. Parallel Computing will be a necessary tool for the data to conserve memory based on the high-dimensionality of the data..****

2.TASK AT HAND(PROBLEM STATEMENT)

In the following Sign Language data-set, each of the images have been captured by 218 student participants, each taking 10 pictures. The data consists of 2062 images, with the dimensions of images being height,width,channels (64(h),64(w),3(rgb)), and 10 different target categories with each target being a numerical sign language digit ranging from 0-9. The expected outcome of the model is that while given testing images (data model has not seen), It is possible for the model to generate accurate predictions on numerical digit test images within the correct class of 0-9 (10 classes).

The following report provides an analysis based on the data given in the Sign-Language-Digits data-set via implementation Convolutional Neural Networks. As with the MNIST(LeNet-5) database of handwritten digits being used for pattern recognition , this creates space for experimentation on human digits for numerical pattern recognition as well as more complicated sign language gestures as more data is gathered in the future.

3. RESEARCH LEADING TO DEVELOPMENT OF THE CNN ARCHITECTURES.

Convolutional Neural Networks stems from the research of the visual cortex of cats brains in 1958 which helped to develop the self organizing Neocognitron network in the 1980's to the Convolutional neural networks which are currently used. With this research, algorithms have been created to detect many types of complex patterns in the visual field reception nodes, which in turn leads to achieve results that were once unimaginable until the recent advances in computational power and open source information. This neural network architecture is able to detect many types of complex patterns in the visual receptors that can surpass human capabilities in various domains (if the parameters have been correctly specified).

The convolutional layer is the most important parameter of the CNN architecture as it is a precursor to the number of neurons in the network/layer, as well as the input shape of the data for training (only needed for first layer). As the model progresses through the network it uses the pooling implementation to "shrink" the input image to reduce the use of memory usage as well as overfitting. The CNN architecture TYPICALLY consists of a : *convolution layer, *Relu layer, *Pooling layer (Global Average Pooling or Average Pooling,Max Pooling), *convolution layer, *Relu layer, then fully connected layers including more Relu activation functions, then a softmax layer that outputs the predicted class probabilities. In the case of the SIGN-LANGUAGE-Digits-Dataset there are 10 class probabilities where as the AlexNet CNN has 1000 class probabilities because that specific CNN is built for detecting much more data (cars,horses,buildings,etc.) than the data set used for experiment.

4. METRICS FOR ANALYSIS/ IMPLEMENTATION FOR BENCHMARK MODEL

Given the experiment, the most suitable implementation would be to approach the question as a classification problem rather than regression because the output is expected to return binary/boolean class labels rather than continuous values unless sigmoid(logistic) is used. The appropriate metrics that could be implemented would be to use an accuracy metric, as well as and adaptive learning method. Adding a Learning Scheduler/Adaptive Learning rate to the network is beneficial for adjusting the learning rate during training by reducing it based on the learning schedulers variables. The Step Decay learning rate scheduler has the effect of quickly learning good weights early and fine tuning them later while adjusting the learning rate at the specified training epoch which will possibly help the model create more accurate predictions.

The Sign Language numerical data-set is appropriate to create a predictive model given that it has many training images of multiple classes. This will help the model make accurate predictions given few parameters. There are no missing values as all the pixels are filled in each image (this will cause the model to find variance in the data easily). The outliers that could be considered would be the background in the image surrounding the Numerical Category. Principal component analysis will help to reduce the reduce the features while reducing outliers. Also with the data being images, image pre-processing is a must for the model to accept the input shape of the image as well as the target categories Y(10). Given that the data is a classification task , the data will be linearly separable, but can use non-linear activation functions to help optimize the models performance by connecting the neurons together rather than single perceptrons.

5. VISUALIZATION

After defining the image input shape to make the dimensions suitable for training, we can use the images for data exploration before diving into implementing the neural network. To help explore the data , the Tensorflow framework will be used for its simplicity in visualizing the data. Instead of using the whole mini batch, One sample image will be enough to grasp an intuition of what the Convolutional layers are performing during a Convolution/Max Pooling layer with specified filters and strides.

CREATE EXAMPLE TENSOR FUNCTION THAT CONVERTS THE CHOSEN INPUT IMAGE TO AN ARRAY. THEN RESHAPE THE IMAGE SO THAT A MODEL CAN READ THE IMAGE. THE FOLLOWING FUNCTIONS ARE IMPLEMENTED IN THE FOLLOWING ORDER WITHIN THE FUNCTION (PATH TO TENSOR / PATHS TO TENSOR) - LOAD EXAMPLE IMAGE DATA AS RGB CHANNELS INSTEAD OF THE PREVIOUS IMAGE WHERE IT WAS LOADED AS AN ARRAY - CONVERT LOADED IMAGE TO 3D TENSOR WITH SHAPE (64, 64, 3). Convert 3D tensor to 4D tensor with shape (1, 64, 64, 3) and return 4D tensor. CREATE A VARIABLE THAT RETURNS AN ARRAY OF THE TOTAL NUMBER OF TARGET CLASSES (10) WITH THE IMAGE WIDTH,HEIGHT, AND CHANNELS.

Now that the images have been converted to 4-D arrays, the images can be inputed into a tensorflow CNN model to plot an output of Image(5) at feature map (1 and 2). To achieve creating the first layer in the network, filters and strides need to be defined, as well as the tensorflow variable placeholder and The Convolutional layer to fit the train image into. This tensorflow implentation is used for visualization purposes to thoroughly explain what is happening in each specified parameter in the network.

Filters are a set of vertical and horizontal lines applied to the CNN architecture using the Tensorflow implementation. The strides are a 1D array consisting of 4 elements: 1,stride_height,stride_width,1. The image input variable is the input mini batch 4D tensor[10,64,64,3]. The padding parameter has 2 different categories ("SAME" or "VALID"):

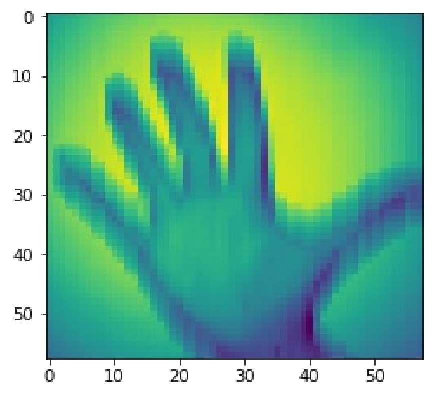
-VALID: DOES NOT use zero padding. Depending on the stride it could miss rows and columns in the data.

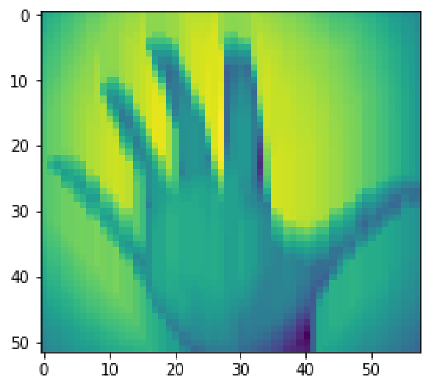
-SAME: DOES USE zero padding if necessary. output neurons = input neurons/ stride (rounded). The zeros are added evenly if possible to the input.

Strides: Connecting a large input layer to a smaller layer by spacing out the receptive fields.The distance between two consecutive receptive fields is the stride. The formula for strides is

The pooling implementation to “shrink” the input image to reduce the use of memory errors. Even though the filters have been defined, A neural network will be able to find the appropriate filters on its own. Experimentation with Cross Validation parameters is also an option to find the best parameters for the model but can cause be time consuming.

This is what the implementation results output for Image(5) at feature map (1 and 2).





6.BENCHMARK ANALYSIS

To thoroughly begin the experiment the data will be reshaped and fit to the Multi Layer Perceptron model to serve as a benchmark to help gain an intuition as to the number of Dense (Hidden) layers to add to the network as well as other hyper-parameters that may be necessary for adjustment to the CNN model architecture. The implementation of an MLP consists of stacking multiple perceptons(LTU's). A single perceptron is when the inputs and outputs are numbers instead of binary on/off values and each input is associated with a weight.(z=w1x1+w2x2+...+wnxn=w^t * x). Multi-layer perceptrons contains one or more perceptrons (hidden layers) and an output layer. Each layer is fully connected to the next and has a bias neuron except the output layer. What classifies the model as a Deep Neural Network is if it consists of 2 or more hidden layers. The use of an MLP classifier can be optimized by adding an adaptive learning optimizer such as the ADAM algorithm, with a number of neurons in hidden layers that will hopefully output a generous accuracy.

7. PREPROCESSING PHASE 1 FOR MULTILAYER-PERCEPTRON CLASSIFIER -TRAINING/TESTING MLP CLASSIFIER

ONE STEP BEFORE TRAINING AND TESTING THE DATA IS THE PREPROCESSING STEP. THE TARGET LABEL (Y) HAS ALREADY BEEN PREPROCESSED INTO THE APPROPRIATE (N_SAMPLES,N_CATEGORIES). THE X VARIABLE HAS 3 DIMENSIONS, WHICH NEEDS TO BE PREPROCESSED (FLATTENED) TO ONLY 2 DIMENSIONS(#SAMPLES,#INPUTS -> (H * W)) SO THE INPUT /OUTPUT DIMENSIONS OF THE IMAGE ARE THE SAME FOR THE ALGORITHM TO RUN CORRECTLY. AFTER PREPROCESSING THE DATA,THE FOLLOWING 2062 IMAGES WILL BE SPLIT INTO A TEST SIZE OF 619 IMAGES AND A TRAIN SIZE OF 1113. SPLITTING THE DATA INTO TRAIN/TEST BATCHES ASSISTS IN REDUCING THE RATE OF OVERFITTING("HIGH VARIANCE") IN THE DATA. ALSO SETTING THE TEST SIZE TO LOW CAN CAUSE UNDERFITTING WHICH WILL CAUSE HIGH BIAS IN THE MODEL .. NOT GENERALIZING THE DATA WELL CREATING A MODEL THAT HOLDS NO BENEFICIAL USE.

The hidden layers parameter in the MLP classifier can predict even some of the most complex problems with only 1 hidden layer. The more complex the problem is one hidden layer may not be acceptable and will cause over-fitting. If decided to increase the number of hidden layers this can cause underfitting. To control over-fitting and underfitting, the tolerance can been adjusted for the model to continue iterating for the accuracy score to increase over the iterations as well as removing/adding hidden layers. Generally, shrink the hidden neuron by half of the previous activated neurons in the layer. The general purpose of using a Multi-layer perceptron against a Perceptron network is that the the MLP can output class probabilities although its term(Regression meaning continuous values) . This is why a MLP seems beneficial for training an image classifier vs a Linear Model , even though the data is linearly separable. We can adjust the complex weights and hyper-parameters in the MLP with it's use of the backpropagation algrotihm which a linear model alone does not have the appropriate step function an MLP does .

8. MLP CLASSIFIER RESULTS

With the given parameters, The MLP classifier reached an accuracy: .44 with the final loss being 1.48. The estimator contains 100 nodes each in the 4 hidden layers, the Adam Learning optimizer, with a tolerance of .000000099 and alpha of .000055. Given the chosen hyper-parameters, the iterations have reached a max of 58 prior to training ending. The tolerance ended the model before over-fitting could occur. Increasing the number of hidden layers could increase the accuracy or hurt the model as it is not certain . A cross-validation execution will be able to give the "best" parameters for the model but can be time consuming. Given the low accuracy for the estimator, this does not seem like a sufficient model to cast predictions. With a total 400 hidden nodes, It is possible we can gain higher accuracy through implementing a CNN using KERAS with less hidden nodes. Dimensionality reduction will be useful for going through the rest of the analysis.

9.PRINCIPAL COMPONENT ANALYSIS DIMENSIONALITY REDUCTION

Principal Component Analysis is a useful statistic for reading complex patterns in high dimensional data to highlight their similarities and differences. Principal component Analysis is implemented after fitting the MLP classifier to reduce the number of dimensions of the image. not all the principal components need to be stored. Keeping only the first *L* principal components/weights gives the truncated transformation. For the procedure of DIMENSIONALITY REDUCTION, the P.C. Analysis equation ($T = X W$) finds the number of dimensions, while allowing the function to keep 95% of the variance of the original images. With each image being (M+N) pixels wide, each image will be held in an image vector. Afterwards all the images are grouped in one image-matrix. The dimensionality reduction implentation has reduced the number of dimensions from 2062 to 316 features. Dimensionality reduction with PCA can be a very useful analysis tool for visualising and processing high-dimensional datasets such as the given experiment, while still retaining as much of the variance in the dataset as possible.

10. CNN KERAS METRICS

Using the KERAS framework for configuring the neural network, there are many implementations that can be involved in generating the predictions. The network will consist of a given number of input neurons connected to a number hidden neurons to connect to the final output softmax layer with the number of classes. The model will be able to output the given test sign-language-digit image from the dataset into a Class/Boolean integer along with the accuracy score for the test image. Defining the network architecture is a very computationally expensive process. Inputting too many neurons can cause the function to run very slow. To solve this issue, The number of neurons can be set to a low value while splitting the number of hidden neurons to a fraction of that input neuron. Setting the number of input neurons to a low value achieve very accurate results compared to a network with neurons set to high values and unnecessary number of hidden layers. Typically the number of input neurons in the convolution layers are set to a size within the size of the inputs(M+N). Within the network, the strides and filters parameters will be specified to set the size of the convolutions as well as a chosen activation function.

For the neurons in the first Convolutional layer ,they are only connected to the receptive field of the second convolutional layer instead of every other pixel located within a small rectangle in the first layer. This architecture allows the model to focus on lower level features in the first hidden layer into the higher-level features in the second layer, and so on.

The activation function takes the decision of whether or not to pass the signal to the next neuron or forget it. Using a non-linear activation function such as a sigmoid (logistic) will enable the network to approximate the complexity of the network. Without the non-linearity introduced by the activation function, multiple layers of a neural network are the same as a single perceptron network..

Sigmoid'(x) = Sigmoid(x)*(1-Sigmoid(x))

The sigmoid function is an 'S' shaped curve, that takes a real-integer value and flattens it into a range between 0 and 1 (0 if integer is negative 1 if integer is positive).It is an accurate representation of a biological neuron behavior. The output shows if the neuron is firing or not. It's usage in the output layer is to predict class probabilities. This means that for the output of the 10 predicted class probabilities for a given input image, if the output probability score is >.5 for a target class (1,2,3,..10) then the target digit has been classified within the an array matrix of images and classes.

The batch normalization technique will be implemented within the network to help optimize performance as well. Batch Normalization,normalizes the activations of the previous layer at each batch(applies a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1). Implentation of a regularization function will also help to optimize the model. Regularization helps tune and control model complexity, ensuring that our models are able to make (correct) classifications on the data (how well the model can generalize). This in turn helps the model to reduce the possibility of underfitting by preferring the smaller weights. Applying regularization can decrease training accuracy while raising testing accuracy..Hence the reasoning for this implementation (reducing overfitting). Regularizing the network does improve the accuracies, but the gain is small.

The implementations discussed will help generate a convolutional neural network. Monitoring the training scores during the epochs will help to determine if the model is overfitting the data or underfitting which will have dramatic impacts on our testing accuracy if the model is not getting the appropriate parameters to train the model. The parameters in the MLP benchmark classifier can be used to help configure the hidden layers in the network if necessary as well as the learning rate for optimization..The accuracy of the MLP was rather low so relying on the MLP classifer may not be the most beneficial way to develop this architecture.

Defining the number of epochs and batch size is important for the classifier. If the number of epochs is set to low the model will not gain enough information to cast accurate predictions as the accuracy generally gets better as the model progresses. Batch size is important because it helps with the training time given the size of the batch. Setting the batch size high can cause data to converge faster but can be very computationally expensive as well.. In general too many epochs may cause your model to over-fit the training data (model is memorizing not learning).

The implementation will consist of two different convolutional neural network models. The first model will be using an ADAM adaptive learning method with specified decay,epsilon rate, and learning rate values. The second implementation will be using the ADAM optimization method, but instead has a learning-rate time-step decay function to explore different different training scores throughout the progression of the model accessing different learning rates by decreasing the learning-rate by a value by a certain number of epochs. Without changing the architecture of the model, The time-step decay function is hoping the model will cast more accurate predictions in the output. This implementation is not a definite way to optimize the model but it is definitely worth a try! The accuracy metrics involved in this classification will be the ROC curve, Mean Squared Error, as well as Mean Absolute Error with the loss metric being CATEGORICAL CROSSENTROPY.

The reasons for using Mean Squared/Absolute is that these metrics both generate different scores but have the same concept in mind, disregarding the equations of these metrics. The difference in the two is that the Squared Error is looking for the average in the epoch while the Absolute Error is looking for the median average. The results from evaluating a metric are not involved in training. The scores for these metrics will also vary on different sets of data as well as the network. In some cases, the score nearing zero, like the loss is the goal.

ROC (Receiver operating) curve implementation is a metric involved in the model for getting a view of the true positive(sensitivity) and false positive(costs) rates at a certain threshold based on train classes against the test classes which creates the curve. The ROC_AUC (rate under curve) is a measure of how well a models parameters can distinguish between the classes. The closer the ROC curve is to the upper left corner of the graph
(100 or 1.0), the higher the overall accuracy of the test .

Categorical crossentropy is a common loss implementation in image classification models. Categorical crossentropy is used for models that will have outputs with multiple samples in multiple categories rather than multi-label classification for binary crossentropy. With binary cross entropy, only two classes can be classified. With categorical crossentropy, any number of classes can be classified.

After defining the metrics, parameters, loss, and functions for the model, it is now possible to start the implementation for the model. This is so that it is possible to analyze the models performance during the training and validation. The implementations discussed earlier will hopefully reveal that machine learning in it's early stages can still hold amazing capabilities such as recognizing sign language using image classification with the given amount of data. Keeping in mind that activated layers and hidden layers will not always optimize the models performance when keeping the model moderately small can usually benefit in the long run.

After training/validating the model and generating the metric scores, there will be data to test on to view how well the model is generalizing. This will help to gain an understanding whether or not increasing the size of strides, filters, size, neurons, etc. will benefit the model. There is no definite way to decide the parameters of the model, but with convolutional neural networks being computationally expensive, out-of-memory errors are likely to happen if the model is too large. If the network is misshaped the model will be rejected and the layers will have to be adjusted. Responsible use of the parameters will help the model optimization and generate outputs that can capture data that can have applications in the real world. For the Sign-language-digits set of data,given the amount of input images for the number of 10 classes, an accuracy of 80% will be the goal of the experiment using the Keras framework. This will make the model accurate for real world use!

11. CONVOLUTIONAL NEURAL NETWORK CONSTRUCTION/ANALYSIS

The strategy for building the network will be as follows: Develop a model that can only store a few features that will have to focus on the most significant features found in the data, and these are more likely to be the most relevant features that optimize generalization. This means the smaller number of neurons - the faster learning, better generalization. For the 64*64 pixel images we've been using, this means our network has 4096 (=64*64) input neurons. All the neurons in the first hidden layer detect exactly the same feature , just at different locations in the input image and so on.

For implementing this neural network, the techniques taken to generate an exceptional model have been described in the writing above. Keeping in mind the size of the data and the network, it is reasonable to keep the size rather small while making sure that the model will generate accurate output for the class categories. One way to make sure the model will make accurate predictions is by watching the verbosity in during the training process. If the training/validation accuracy is not reaching the accuracy measurement needed, adjustments to the parameters in the network may help.

In general, going about creating a neural network can be a complex process, but keeping everything small will help in the long run. To begin, The data needs to be preprocessed. The categorical data has already been preprocessed. As explained above: the input image variable has 4 dimensions (2062,64,64,1) which needs to be flattened to 3 dimensions (n_samples,64,64,1) so that the data has equal dimensions to execute properly with the categorical data (n_samples,10).

After preprocessing the data, it is now possible to tune the hyper-parameters for the model. The hyper-parameters include what have been discussed above as well, but instead will now be specified as well as reasoning behind choosing the appropriate metrics. Afterwards, the results of the achievements of the model will be discussed as well as what can be done to possibly improve the model.

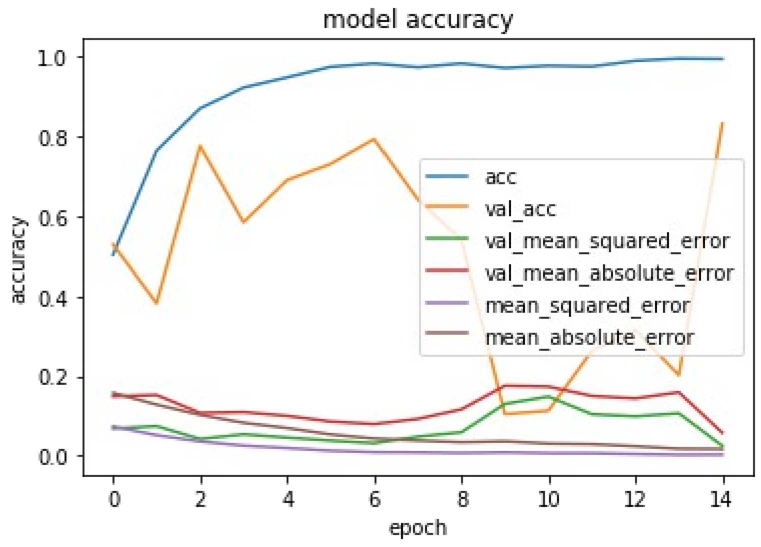
The model will consist of 3 convolutions with each convolutional layer having a size of [8,8]. Two of the convolution layers have 64 neurons each, with an exception of one having 32 neurons. Each convolutional layer will have a SIGMOID activation function, and the use of L2 regularization, followed by the Max pooling layer, and the final output softmax layer for outputting the 10 class probabilities. The use of batch normalization will help to normalize the activations of the previous layer at each batch: maintains a mean activation close to 0 and the activation standard deviation close to 1.   The dropout is a technique where randomly selected neurons are ignored during training in the case of this model (.15 will be the max amount of neurons left out in a layer).This ensures the model becomes less sensitive to the specific weights of neurons. This in turn results in a network that is capable of better generalization and

is less likely to overfit during training. For the final fully connected layer that will consist of the number of categories (10) that are connected to the softmax activation layer for prediction.
THE FOLLOWING IS THE SUMMARY OF THE CNN MODEL GIVEN THE CHOSEN IMPLEMENTATIONS

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_26 (Conv2D)           (None, 57, 57, 64)        4160      
_________________________________________________________________
batch_normalization_32 (Batc (None, 57, 57, 64)        256       
_________________________________________________________________
dropout_22 (Dropout)         (None, 57, 57, 64)        0         
_________________________________________________________________
max_pooling2d_25 (MaxPooling (None, 28, 28, 64)        0         
_________________________________________________________________
conv2d_27 (Conv2D)           (None, 21, 21, 32)        131104    
_________________________________________________________________
batch_normalization_33 (Batc (None, 21, 21, 32)        128       
_________________________________________________________________
max_pooling2d_26 (MaxPooling (None, 10, 10, 32)        0         
_________________________________________________________________
conv2d_28 (Conv2D)           (None, 3, 3, 64)          131136    
_________________________________________________________________
batch_normalization_34 (Batc (None, 3, 3, 64)          256       
_________________________________________________________________
max_pooling2d_27 (MaxPooling (None, 1, 1, 64)          0         
_________________________________________________________________
dropout_23 (Dropout)         (None, 1, 1, 64)          0         
_________________________________________________________________
flatten_8 (Flatten)          (None, 64)                0         
_________________________________________________________________
dense_18 (Dense)             (None, 124)               8060      
_________________________________________________________________
batch_normalization_35 (Batc (None, 124)               496       
_________________________________________________________________
activation_15 (Activation)   (None, 124)               0         
_________________________________________________________________
dense_19 (Dense)             (None, 10)                1250      
_________________________________________________________________
activation_16 (Activation)   (None, 10)                0         
=================================================================
Total params: 276,846
Trainable params: 276,278
Non-trainable params: 568
```

Using the ADAM adaptive learning optimization function, we will set the decay-rate(.0025) which is how much the learning rate decreases each epoch, learning-rate(.0025) increases performance while reducing training time, and a low epsilon factor of (.00025). The model will have a categorical-crossentropy loss function as well as using the mean squared error and mean absolute error for accuracy metrics.

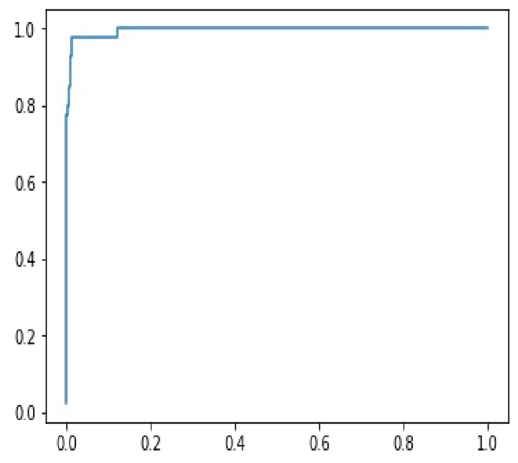
12. RESULTS

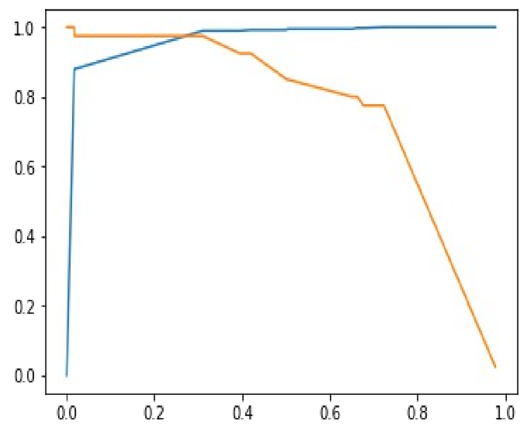


As shown from the above visualization, the training accuracy steadily increased over the epochs while the validation accuracy had fluctuations in scores, then dramatically decreased during epochs 9-13 but had a sharp increase in the final 2 of the 15 epochs. Increasing the number of epochs may help with the accuracy of the model by increasing its training time. The training/validation scores of the mean squared/absolute errors are nearing 0 which is an exceptionally acceptable low value, the scores in these metrics almost resemble eachother which is great, meaning that the bias is low and the model is generalizing well !

FINAL ACCURACY SCORES FOR THE MODEL AFTER 15 EPOCHS OF TRAINING AND TESTING GIVES A FINAL TEST ACCURACY OF 85.47% . THIS MODEL IS TWICE AS GOOD AS THE MODEL CREATED WITH THE MLP CLASSIFIER WITH LESS NEURONS. THE IMPLEMENTATION OF THE NEURAL NETWORK CONFIRMS THE ASSUMPTION OF A SMALLER MODEL GATHERING MORE FEATURES IN THE DATA IS HIGHLY ACCURATE.
max train accuracy ->0.9952606680268925
max validation accuracy-> 0.8316498215150352
min train accuracy-> 0.503317523200365
min validation accuracy-> 0.1043771044774489

VISUALIZATION OF THE TRUE POSITIVE/FALSE POSITIVES WILL HELP US TO GAIN A BETTER INTUITION OF THE CORRECT AMOUNT OF DATA. IN THE ROC CURVE, THE CLOSER THE CURVE IS TO THE TOP LEFT, THE MORE ACCURATE THE DATA. As shown the ROC/ROC_AUC scores are very accurate holding a Crossover at 0.02 with specificity 0.88 with an ROC area under curve of 1.0.

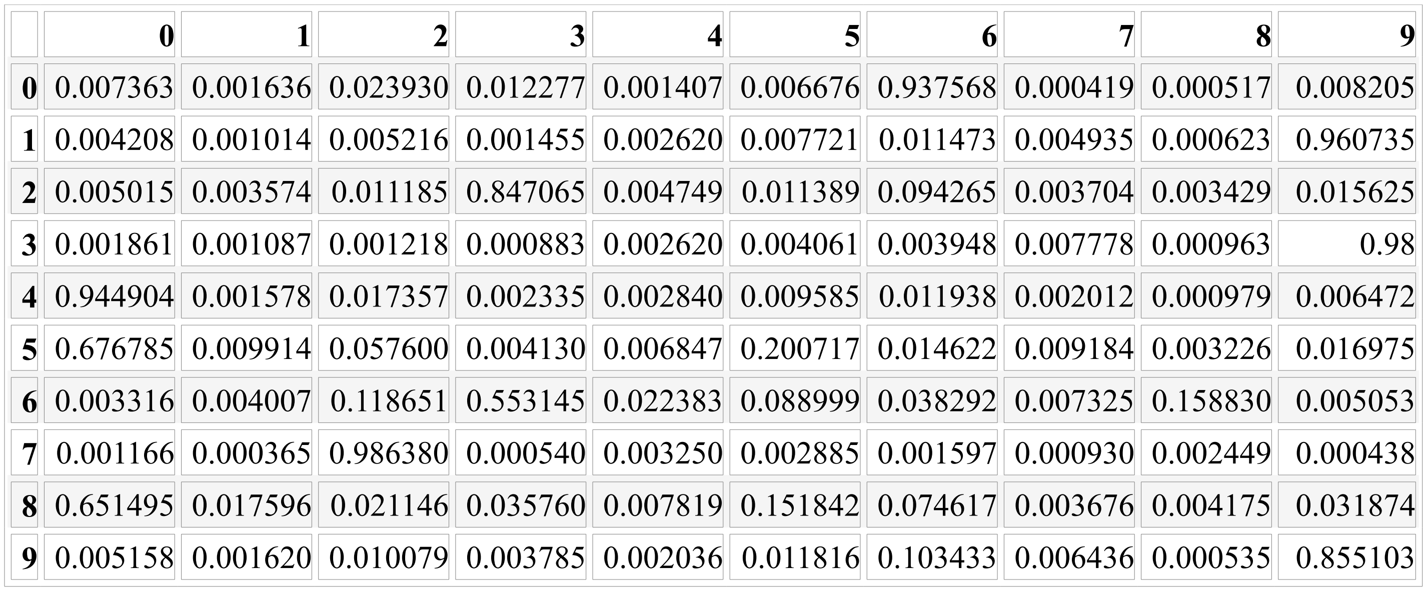




ROC AUC: 1.0

ROC: .88

NOW IT IS TIME TO TEST THE MODEL AGAINST THE FIRST 10 TESTING IMAGES AND GET THE ACCURACY SCORE. THIS IMPLENTATION WILL CAST PREDICTION ON IMAGE 6 FROM THE TESTING DATA AND RETURN THE PREDICTION AS AN ARRAY [0,0,0,0,0,0,0,0,0,0]. FOR EXAMPLE IF PREDICTION IS A 3 , THE OUTPUT WILL BE [0,0,0,1,0,0,0,0,0,0].



	0	1	2	3	4	5	6	7	8	9
0	0.007363	0.001636	0.023930	0.012277	0.001407	0.006676	0.937568	0.000419	0.000517	0.008205
1	0.004208	0.001014	0.005216	0.001455	0.002620	0.007721	0.011473	0.004935	0.000623	0.960735
2	0.005015	0.003574	0.011185	0.847065	0.004749	0.011389	0.094265	0.003704	0.003429	0.015625
3	0.001861	0.001087	0.001218	0.000883	0.002620	0.004061	0.003948	0.007778	0.000963	0.98
4	0.944904	0.001578	0.017357	0.002335	0.002840	0.009585	0.011938	0.002012	0.000979	0.006472
5	0.676785	0.009914	0.057600	0.004130	0.006847	0.200717	0.014622	0.009184	0.003226	0.016975
6	0.003316	0.004007	0.118651	0.553145	0.022383	0.088999	0.038292	0.007325	0.158830	0.005053
7	0.001166	0.000365	0.986380	0.000540	0.003250	0.002885	0.001597	0.000930	0.002449	0.000438
8	0.651495	0.017596	0.021146	0.035760	0.007819	0.151842	0.074617	0.003676	0.004175	0.031874
9	0.005158	0.001620	0.010079	0.003785	0.002036	0.011816	0.103433	0.006436	0.000535	0.855103

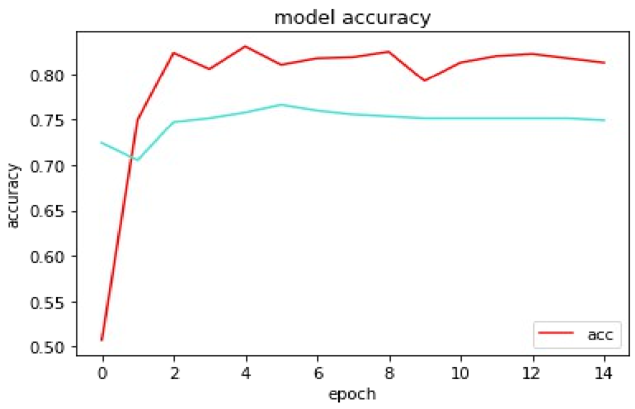
As shown in the table above and below, it shows the output predictions of the test inputs. For image 3 it is predicted to be in class 9 because the value(.98) is the highest out of the 10 classes. This is how the predictions are evaluated. In the below table it clearly represents the predicted class for the output predicted class.

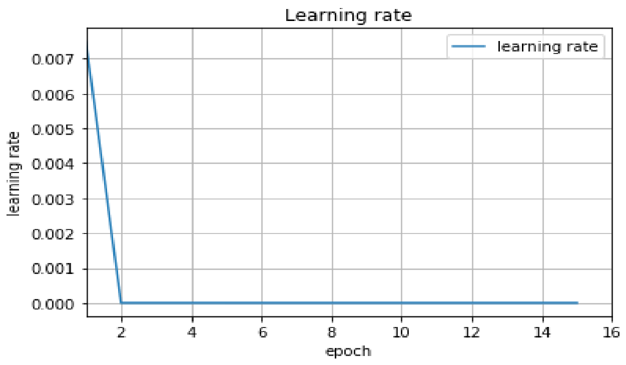
BELOW ARE THE PREDICTED OUTPUT CLASSES BASED ON THE FIRST 10 TEST INPUTS. THE CLASSES HAVE BEEN CLEARLY VISUALIZED ALONG WITH THE INPUTS. WITH THE ABOVE MODEL GENERATING A SCORE OF 85%, WE CAN NOW USE THE MODEL TO TEST AGAINST NEW DATA!

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
2	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
4	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
7	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

IMPLENTING CNN USING KERAS AS WELL AS LEARNING RATE SCHEDULER DECAY WITH AN INTIAL LEARNING RATE OF .0075 based on the previous model which gave a score of 87% AND DECREASING IT BY .0025 EVERY 3 EPOCHS. THIS WELL HELP THE MODEL TRAVEL THROUGH DIFFERENT LEARNING RATES OUTPUTING A GENEROUS ACCURACY SCORE EACH EPOCH WHILE REDUCING OVERFITTING WITH THE SIGMOID ACTIVATION FUNCTION AS WELL AS USING THE DROPOUT IMPLEMENTATION. THE USE OF THE SIGMOID IMPLEMENTATION WILL BE NOTICED IN THE VALIDATION ACCURACY VS THE TRAINING ACCURACY,(WHERE THE VALIDATION ACCURACY GENERALLY HAS A HIGH SCORE THAN THE TRAINING, THIS IS THE OVERFITTING CONTROL IN ACTION..)

The following graphs visualizes the learning-rate scheduler decrease per epoch as well as accuracy for the model. As shown this model still has an accuracy of 76%. Still better than the baseline MLP classifier, but still not as well as an 85% like the defined model not using learning-rate schedulers.





Given the inadequate evaluations of the model implemented with the learning-rate schedulers. We can conclude that the stand-alone defined keras model implanted to achieve an 85% accuracy will be the model of choice.