AITS Internship

Task - 3

Name : Harsh P.V

Email : harsh.pv07@gmail.com

Task :

Implement hand digit recognition in Keras with dataset at https://keras.io/examples/mnist_dataset_api/ and find the effect of changing loss function and optimizer algorithm during model.compile.

Abstract:

The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems.

The database is also widely used for training and testing in the field of machine learning. The dataset consists of different types of handwritten digits and alphabets. The images from the Mnist dataset consist of 28 x 28 normalised images and it also consists of a total of 60000 train images and 10000 test images.

Introduction:

Classification is a technique to categorize our data into a desired and distinct number of classes where we can assign a label to each class. Mnist consists of more than 100 classes of data ie different alphabets and numbers. There are many different types of algorithms for classification.

For this application, we are using Neural Networks for the classification of the different digits in Mnist dataset.

Overview on Deep learning classification Models:

We first group the data into multiple classes ie based on their similarity. This same process should be repeated for both the test and training sets. We need to preprocess the images ie to resize it to a suitable size.

We use CNN to classify the images. CNN consists of many layers that help in extraction of favorable features from the image and learn from the features.

The different filters and layers that is used in CNN are 1)Convolution 2) Max pooling 3) Dropout 4) Dense.

Metrics Used :

1. Accuracy score:

It is the most commonly used score in classification related problems:

$$accuracy\,(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} 1\,(\hat{y} = y_i)\,.$$

2. Confusion matrix:

Confusion Matrix is also used in comparing the Performance of the classification problems.

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FN} + \mathrm{FP}} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{P} + \mathrm{N}}$$

Mnist Dataset :

The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. The database is also widely used for training and testing in the field of machine learning.It was created by "re-mixing" the samples from NIST's original datasets. The creators felt that since NIST's training dataset was taken from American Census Bureau employees, while the testing dataset was taken from American high school students, it was not well-suited for machine learning experiments. Furthermore, the black and white images from NIST were normalized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels.

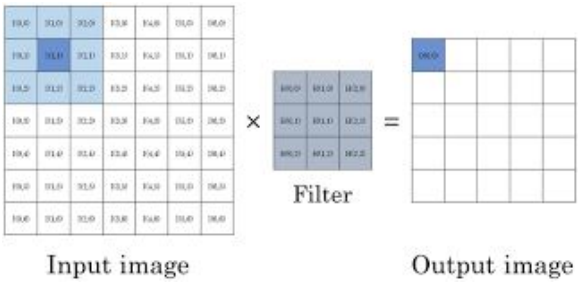
The MNIST database contains 60,000 training images and 10,000 testing images.

Preprocessing:

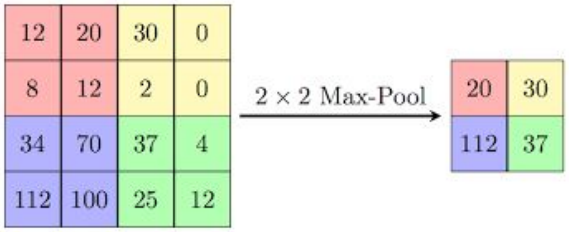
First we change the input shape of the image to 1. This makes all the images to be processed in B/W. We do not resize the image as the images are of standard sizes ie 28 X 28. Hence we pass the same size of all the layers. Hence the preprocessing phase is very less for this dataset.

Classification model:

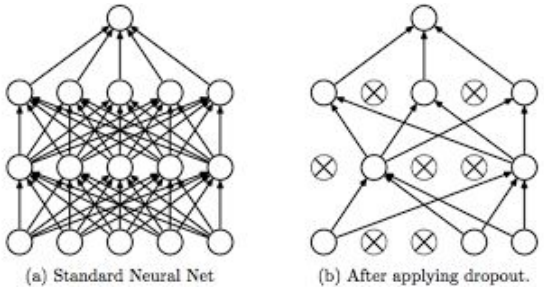
We have used CNN for classification of the Mnist Dataset. As described above, CNN consists of 4 layers. First we pass the image to the convolutional layer. In this layer a 5 X 5 filter is added to the image. This extracts different features from the image.



Next , we apply the Max pooling layer which is aimed to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned. We have added a 2 X 2 filter to the output of the convolutional layer.



Next , we add the dropout layer. The main aim of dropout layer is to prevent overfitting. The dropout layer removes certain hidden layers in the network. This prevents overfitting in the network and therefore improves accuracy.



Next we flatten the network using the flatten function. Flattening is the process of converting all the resultant 2 dimensional arrays into a single long continuous linear vector.

The last layer is the dense layer. A dense layer is just a regular layer of neurons in a neural network. Each neuron recieves input from all the neurons in the previous layer, thus densely connected. The layer has a weight matrix W, a bias vector b, and the activations of previous layer a.

Optimizers :

OPtimizer functions helps us to minimize the cost function.It is simply a mathematical function dependent on the Model's internal learnable parameters which are used in computing the target values(Y) from the set of predictors(X) used in the model. For example — we call the Weights(W) and the Bias(b) values of the neural network as its internal learnable parameters which are used in computing the output values and are learned and updated in the direction of optimal solution i.e minimizing the Loss by the network's training process

and also play a major role in the training process of the Neural Network Model.
The different types of Optimization functions are :
Adagrad - Adagrad is an adaptive learning rate method. In Adagrad we adopt the learning rate to the parameters.It is well suited when we have sparse data as in large scale neural networks.

Adadelta - Adadelta is an extension of Adagrad and it also tries to reduce Adagrad's aggressive, monotonically reducing the learning rate. It does this by restricting the window of the past accumulated gradient to some fixed size of w. Running average at time t then depends on the previous average and the current gradient

RMSProp - RMSProp is Root Mean Square Propagation. It was devised by Geoffrey Hinton. RMSProp tries to resolve Adagrad's radically diminishing learning rates by using a moving average of the squared gradient. It utilizes the magnitude of the recent gradient descents to normalize the gradient

Adam - it calculates the individual adaptive learning rate for each parameter from estimates of first and second moments of the gradients. Adam can be viewed as a combination of Adagrad, which works well on sparse gradients and RMSprop which works well in online and nonstationary settings.

Loss Function:

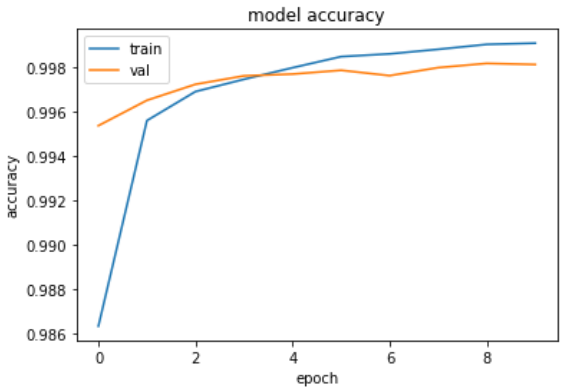
Categorical Cross-Entropy loss - Also called Softmax Loss. It is a Softmax activation plus a Cross-Entropy loss. If we use this loss, we will train a CNN to output a probability over the classes for each image. It is used for multi-class classification.

Binary Cross-Entropy loss - It is a Sigmoid activation plus a Cross-Entropy loss. it sets up a binary classification problem between the 2 classes.

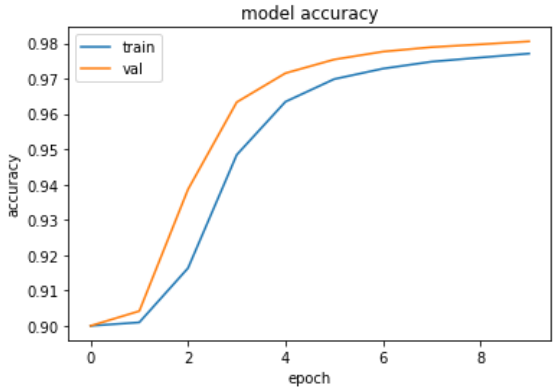
Results:

The graphs given below are the plots between accuracy and epochs for training and testing datasets under different loss and activation functions.

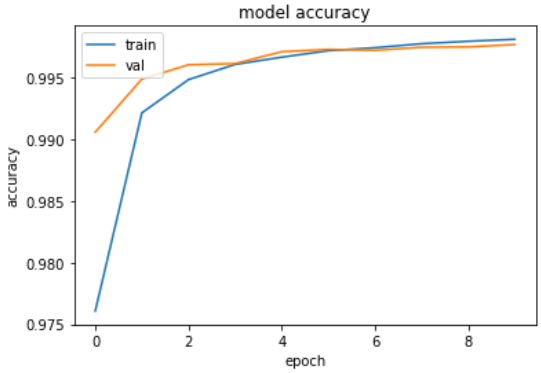
1. **Loss - binary cross entropy**
   Optimizer - adam



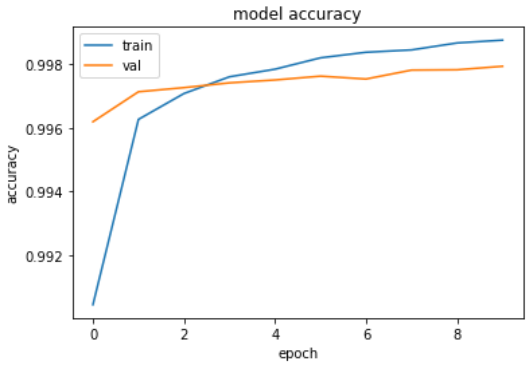
2. **Loss - binary cross entropy**
Optimizer - SGD



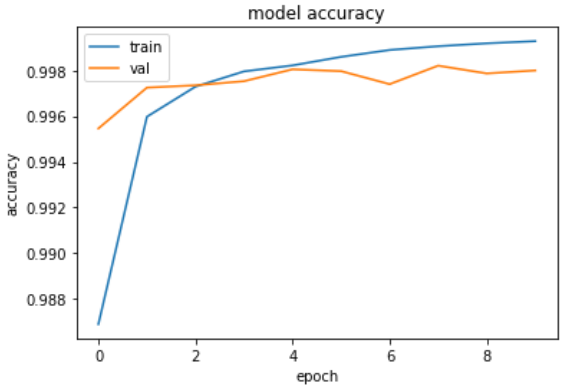
3. **Loss - binary cross entropy**
Optimizer - AdaDelta



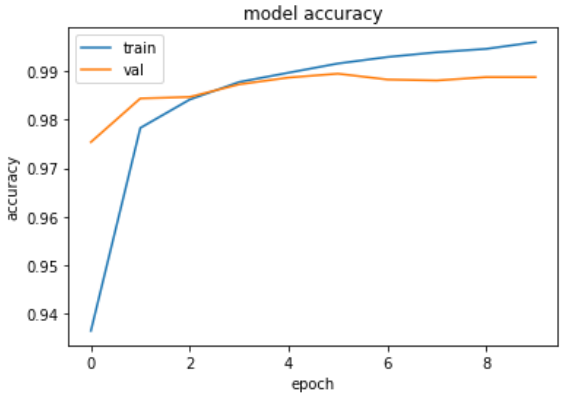
4. **Loss - binary cross entropy**
Optimizer - AdaGrad



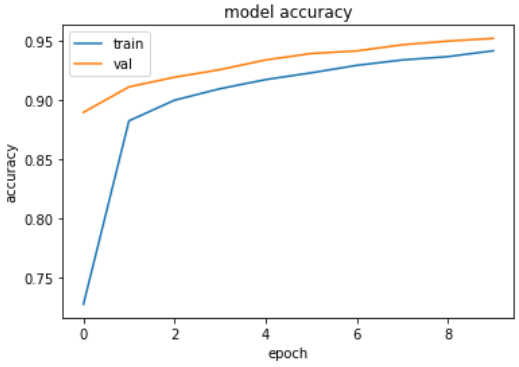
5. **Loss - binary cross entropy**
Optimizer - RMSprop



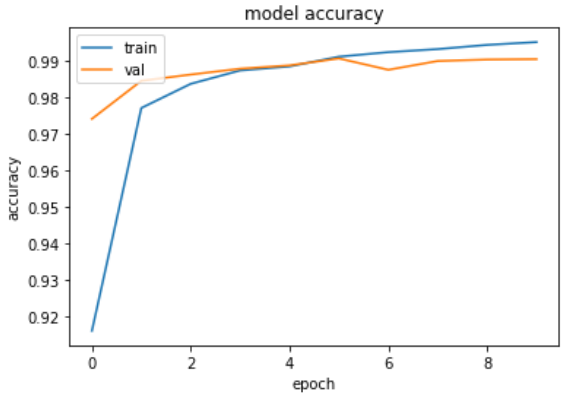
6. **Loss - categorical cross entropy**
Optimizer - adam



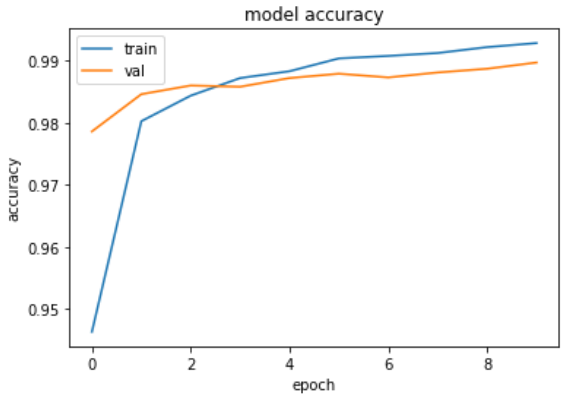
7. **Loss - categorical cross entropy**
Optimizer - SGD



8. **Loss - categorical cross entropy**
Optimizer - AdaDelta



9. **Loss - categorical cross entropy**
Optimizer - AdaGrad



Conclusion:

From the above graphs we can see that, a loss function with categorical cross entropy and optimizer of adadelta gives the best result as both the training and testing data gave us the best accuracy over different epochs.