Handwritten equation solver using Neural Network

Introduction

Machine learning and deep learning are becoming increasingly important in today's world. Machine learning and deep learning techniques are now used in a variety of fields, including handwriting recognition, robotics, and artificial intelligence. To create such a system, we must first train our machines with data, allowing them to learn and make the necessary predictions. Our project will demonstrate a Handwritten Equation Solver that uses a Convolutional Neural Network for classification of specific character and image processing techniques to train on handwritten digits and mathematical symbols. In this project various libraries such as Numpy, Pandas, Keras, Matplotlib and Seaborn were used to perform this project.

By obtaining a dataset that contains digits and mathematical symbols from publicly available resources, processing of the data, and balancing it (removing bias). After the creation of a model of a neural network with appropriate width and depth after data processing and achieved the 98.54% accuracy. Now that the time has come for machines to learn, some optimizers such as AdaGrad, RMSprop, and Adaptive Momentum Estimation (Adam) can assist in data model optimization and loss minimization. Project can use Adam optimizer and Relu and Softmax activation function for this model. Finally testing of the model is also done by providing handwritten equations from paints.
This project will help us immerse ourselves in the world of profound learning and provide us with an idea of neural networks and how they operate. Neural networks are currently employed in various places, for example image processing, character recognition, forecasting etc. Together with this, our project specifies some basic writing ideas, which many mobile applications like Photomath are using. Such, potential of this project motivates us to go further in it.

Methodology

Importing Data

The data is downloaded in the form of images and creates a dataframe of its pixels with the help of pandas.

Training the data

The data is trained by creating a neural network ,the neural networks are multi-layer networks of neurons (the blue and magenta nodes in the chart below) that we use to classify objects, predict outcomes, and so on. The below diagram can depict the function of Neural Network.

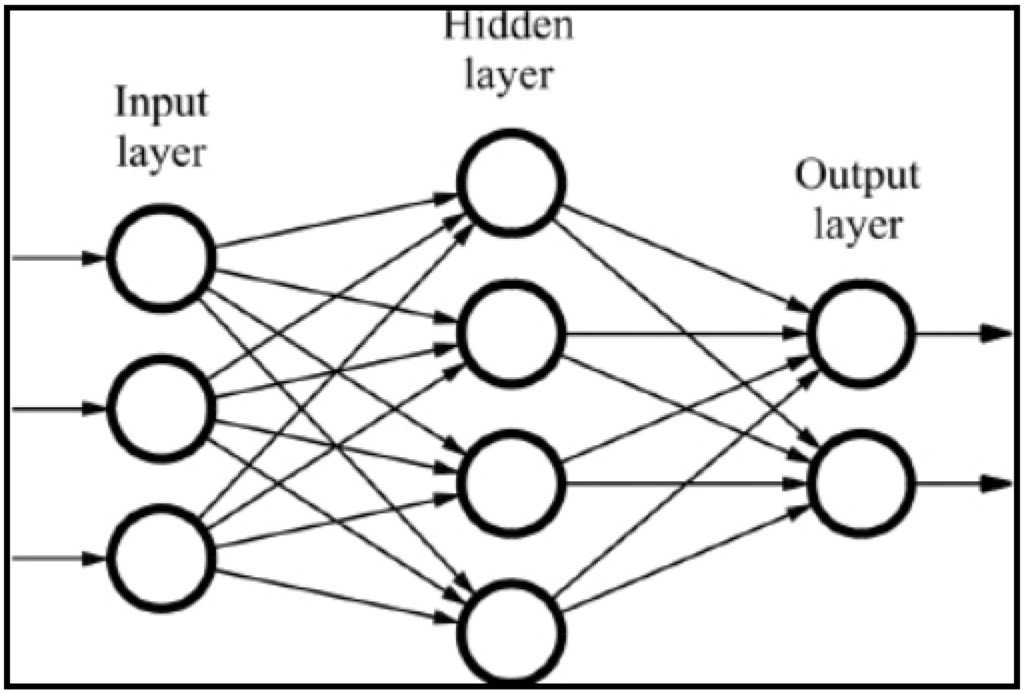


Fig: Flow Diagram of Neural Network

Adaptive Moment Estimation(Adam)

It must either evolve from another algorithm, or overcome some shortcomings. Each algorithm certainly does not come out of nowhere. It is the same for adam. Algorithms of optimization have evolved approximately:

```
vanilla gradient descent => sgd + momentum => adaptive gradient
```

Another method for calculating adaptive learning rates for each parameter is Adam (Adaptive Moment Estimation). Adam keeps an exponentially decaying average of past gradients v, similar to Adadelta and RMSprop, in addition to an exponentially decaying average of past squared gradients s. Adam behaves like a heavy ball with friction, preferring flat minima in the error surface, whereas momentum behaves like a ball running down a slope.

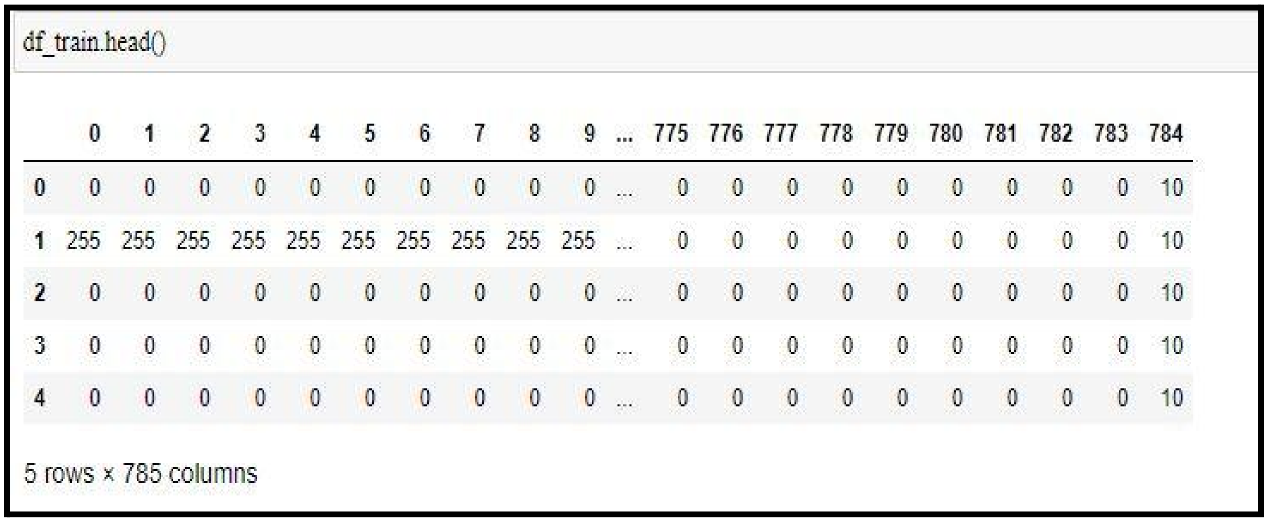
Activation Function

It's just a feature you use to get the node output. It is also known as Transfer Function. There mainly two types of activation function i.e linear activation function and Non-linear activation function. So, here mainly two activation functions are used which are Relu and Softmax i.e non-linear activation function.

Screen shots:

Data implementation

The data source consists of several images of all the numericals from 1 to 9 and some arithmetic operators of multiplication, addition, and subtraction. The following data is concatenated and then converted into the csv file in the later part of the model. The obtained csv file has 784 columns in which the last column is the target attribute which needs to be dropped from the file to perform algorithms.



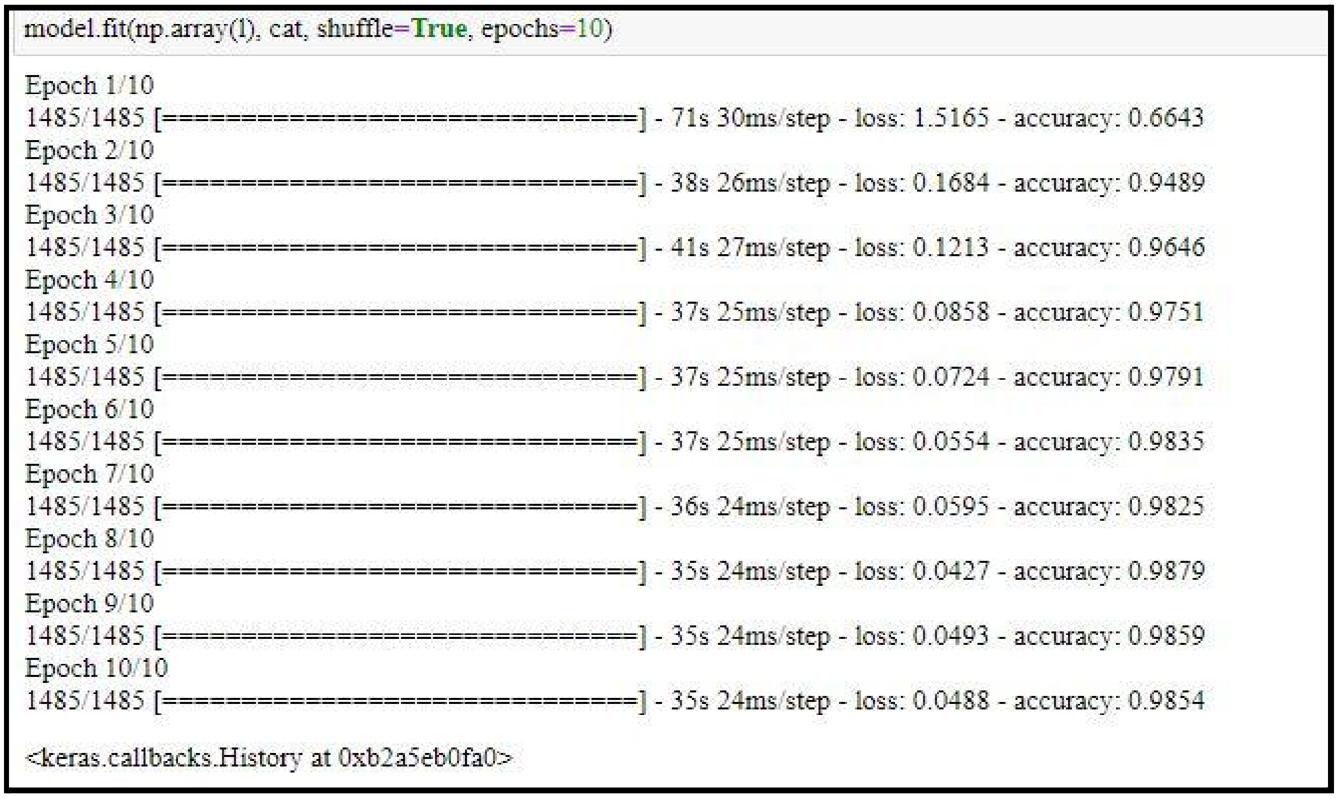
```
df_train.head()

     0    1    2    3    4    5    6    7    8    9 ... 775 776 777 778 779 780 781 782 783 784
0    0    0    0    0    0    0    0    0    0    0 ...   0   0   0   0   0   0   0   0   0  10
1  255  255  255  255  255  255  255  255  255  255 ...   0   0   0   0   0   0   0   0   0  10
2    0    0    0    0    0    0    0    0    0    0 ...   0   0   0   0   0   0   0   0   0  10
3    0    0    0    0    0    0    0    0    0    0 ...   0   0   0   0   0   0   0   0   0  10
4    0    0    0    0    0    0    0    0    0    0 ...   0   0   0   0   0   0   0   0   0  10

5 rows × 785 columns
```

Accuracy of model

To create a model which is related to deep learning Keras library is important and with the help of Keras library the model can be generated. After importing several modules (optimizers, input, dense, sequential, dropout, conv2d, maxpooling2D, flatten) the model can be made with a better accuracy by doing operations with these libraries. There are a total 13 classes in our csv file including arithmetic operators. The model is generated with an accuracy of 98.54% as shown in the below figure by fitting. After completion and checking of model accuracy it can also be saved as json file.



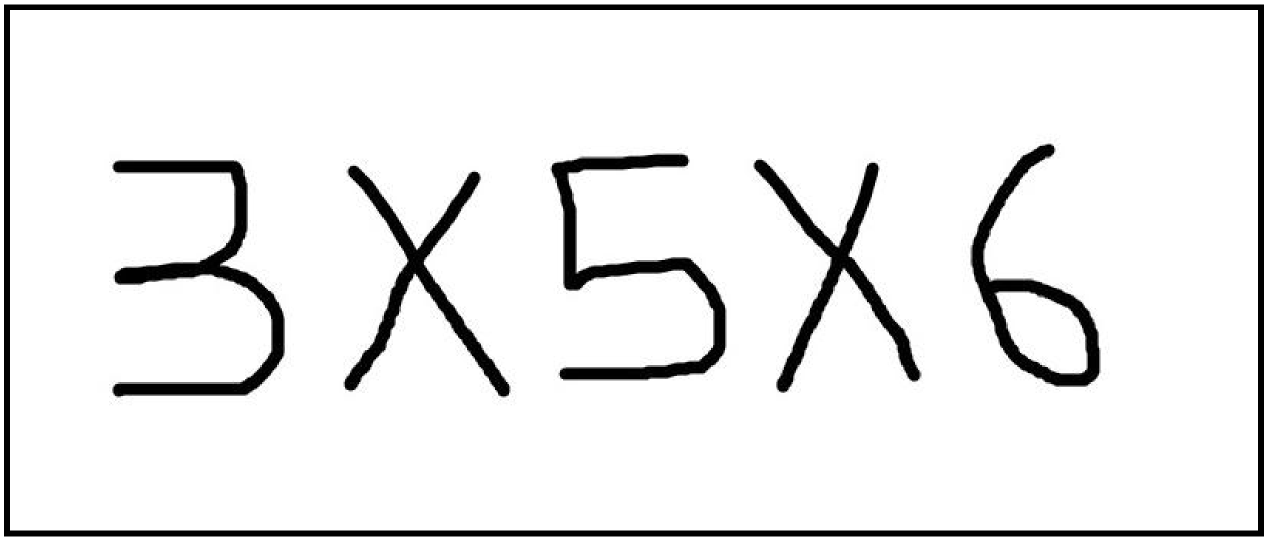
```
model.fit(np.array(l), cat, shuffle=True, epochs=10)

Epoch 1/10
1485/1485 [==============================] - 71s 30ms/step - loss: 1.5165 - accuracy: 0.6643
Epoch 2/10
1485/1485 [==============================] - 38s 26ms/step - loss: 0.1684 - accuracy: 0.9489
Epoch 3/10
1485/1485 [==============================] - 41s 27ms/step - loss: 0.1213 - accuracy: 0.9646
Epoch 4/10
1485/1485 [==============================] - 37s 25ms/step - loss: 0.0858 - accuracy: 0.9751
Epoch 5/10
1485/1485 [==============================] - 37s 25ms/step - loss: 0.0724 - accuracy: 0.9791
Epoch 6/10
1485/1485 [==============================] - 37s 25ms/step - loss: 0.0554 - accuracy: 0.9835
Epoch 7/10
1485/1485 [==============================] - 36s 24ms/step - loss: 0.0595 - accuracy: 0.9825
Epoch 8/10
1485/1485 [==============================] - 35s 24ms/step - loss: 0.0427 - accuracy: 0.9879
Epoch 9/10
1485/1485 [==============================] - 35s 24ms/step - loss: 0.0493 - accuracy: 0.9859
Epoch 10/10
1485/1485 [==============================] - 35s 24ms/step - loss: 0.0488 - accuracy: 0.9854

<keras.callbacks.History at 0xb2a5eb0fa0>
```

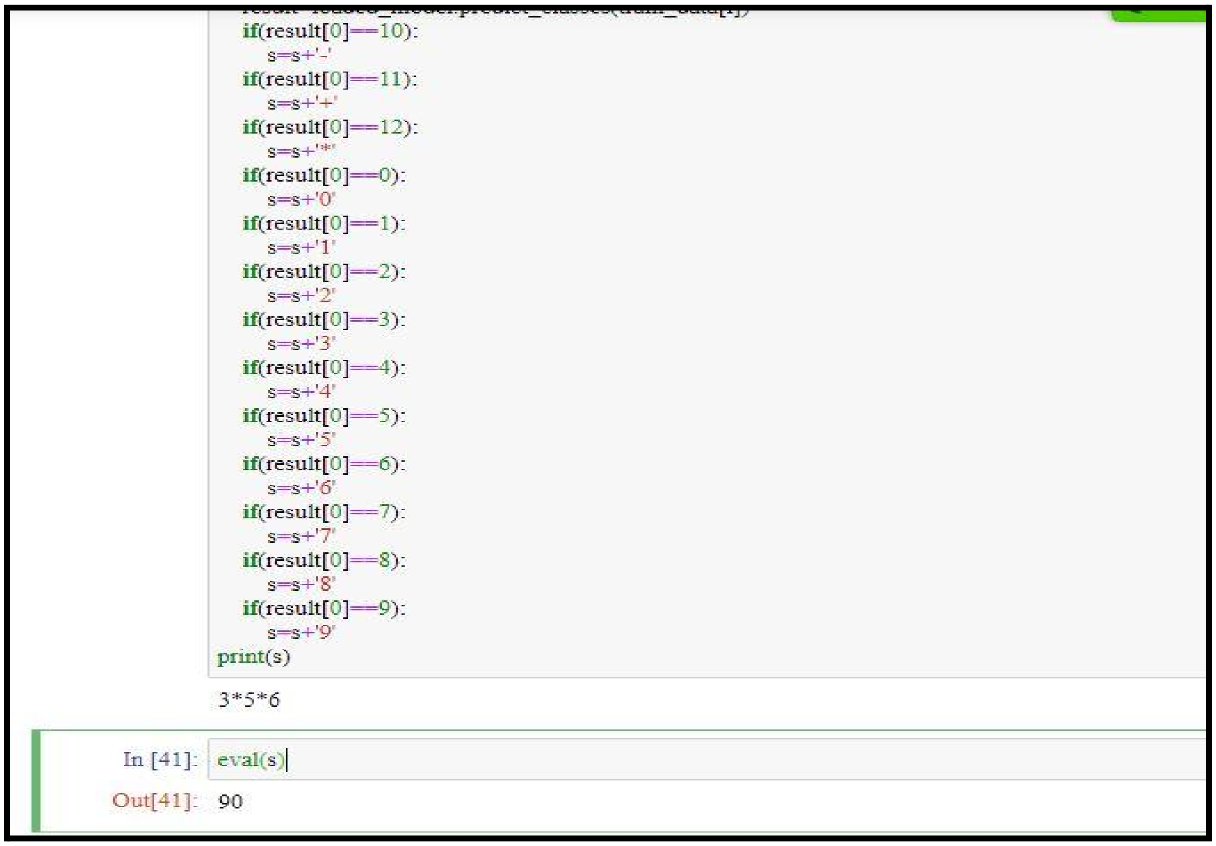
Sample image

The below figure is written to give input to our final model to check whether the results are in good accuracy. The given image is uploaded and a lot of code is used to relate this image to the model and to scan this image so that our model can predict the exact values. Cv2 library is imported in the model and image is uploaded. Width, height, and other parameters of the rectangle are defined in the code which is used to scan the image and distinguish the numbers and arithmetic operators in the given image. Finally, the image is scanned in the code and ready to show the results in the later part.



Final Results

After the image uploading and scanning by the model. Our model is ready to read this image and find out the results of the given handwritten equation solver model. As shown in the below figure the above figure is converted into the numerical values and arithmetic operators also also settled. The result shown by the model is well evaluated with the evaluate function and results can be calculated.



```
    if(result[0]==10):
        s=s+'-'
    if(result[0]==11):
        s=s+'+'
    if(result[0]==12):
        s=s+'*'
    if(result[0]==0):
        s=s+'0'
    if(result[0]==1):
        s=s+'1'
    if(result[0]==2):
        s=s+'2'
    if(result[0]==3):
        s=s+'3'
    if(result[0]==4):
        s=s+'4'
    if(result[0]==5):
        s=s+'5'
    if(result[0]==6):
        s=s+'6'
    if(result[0]==7):
        s=s+'7'
    if(result[0]==8):
        s=s+'8'
    if(result[0]==9):
        s=s+'9'
print(s)

3*5*6

In [41]:  eval(s)
Out[41]:  90
```

References

1. http://yann.lecun.com/
2. https://towardsdatascience.com/introduction-to-neural-networks-advantages-and-applications-96851bd1a207