**Single-layer Perceptron**

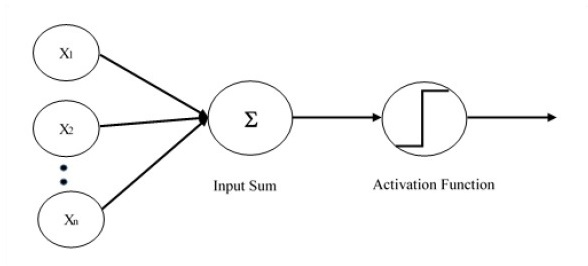
# 

**ABSTRACT**

A ***perceptron***is a primitive neural network created by ***Rosenblatt*** for the task of ***pattern recognition***. A single layer perceptron is a simple feed-forward neural network that is primarily used to classify linearly separable classes.

In this given assignment, a single layer perceptron functioning in ***auto-associative mode*** is designed, trained and tested on a small dataset of alphanumeric characters.This model is built based on ***approach 1***: ***Simple network with scalar output***. The model is also re-tested on ***noise-corrupted*** image dataset to observe its performance.

The goal of this project is to optimize the SLP to give the best possible results.



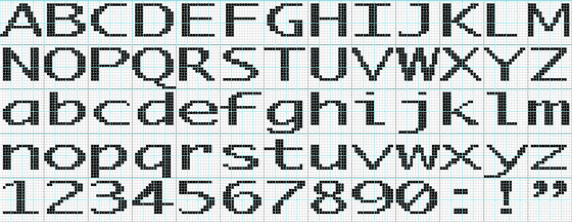
**Figure 1:**SLP model

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CONTENTS** | **PAGE NUMBER** |
| 1 | **Dataset Generation** | 4 |
| 2 | **Network Parameters** | 5 |
| 3 | **SLP Model** | 6 |
| 4 | **Python code for SLP** | 7 |
| 5 | **SLP output results for noiseless input : Graph of FFa Vs Fh** | 10 |
| 6 | **SLP Output Results For Noise-corrupted Input As Table And Graph Of Fh And Ffa** | 11 |
| 7 | **Discussion** | 12 |
| 8 | **Appendix** | 13 |

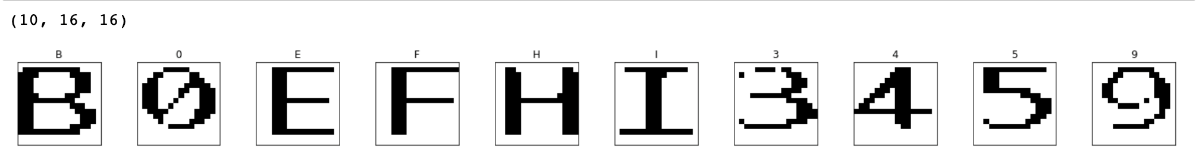
**1. DATASET GENERATION**

The dataset was created by processing the **exemplar image** ( Fig.2) provided. Each alphabet and digit were individually cropped with the help of modules from **Pillow library** in Python. The individually cropped images were then re-sized to **16 X 16**. The images were then converted to **grayscale** from **RGB**. Following, the image matrices were normalized to have either a **0 or 1**. Essentially creating **binary images**.



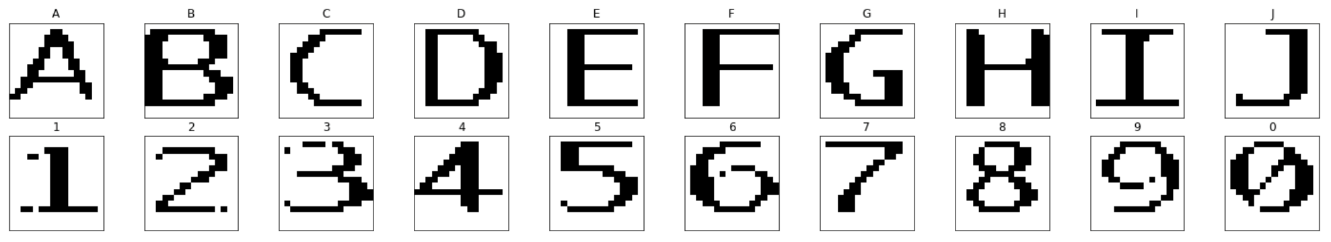
**Figure 2:** Exemplar GIF

The images obtained were then used for testing and training. **10 images** were selected to perform the **training**, they were the alphanumeric characters (**B, E,F,H,I,3,4,5,9,0**). The **testing** was performed on the entire **dataset**.

***TRAINING SET***

**Figure 3:**Training set

***TESTING SET***

**Figure 4**:Testing set

**2. NETWORK PARAMETERS**

1. **Input:**16x16 pixel images

2. **Output:** index associated with the alphanumeric character

3. **Number of weights:** 256, bias 1 is included.

4.**Weight matrix**(Initially,weights were generated randomly using a random function)

5. **Activation function:** linear activation

6. **Learning rate:** 0.001

7.**Epochs**:30,50,100,500,1000,10000.(tabulated below)

8. **Evaluation metrics:** Fh,Ffa

It is observed that the model performed slightly better with a lower learning rate. Although the model didn't precisely predict the alphanumeric index in all cases, it was able to recognize an alphabet as an alphabet and a digit as a digit. For instance, “A” was predicted to closely resemble “E” after training the model with higher iteration count when earlier it was identified as “0”.

|  |  |
| --- | --- |
| **Number of iterations(epochs)** | **Accurate prediction count** |
| 30 | 9 |
| 50 | 10 |
| 100 | 10 |
| 500 | 10 |

**Table 1**: accuracy table for noiseless data

|  |  |
| --- | --- |
| **Standard Deviation** | **Accurate prediction count** |
| 0.001 | 10 |
| 0.002 | 10 |
| 0.05 | 5 |
| 0.1 | 1 |

**Table 2**: accuracy table for noise-corrupted images

The **number of epochs** were chosen in a manner to **improve** the **performance** of the model. It is observed that as the number of epochs changed from **30 to 50**, the number of inputs that were predicted correctly increased from **9 to 10**.Then, as the number of epochs were increased the number of predictions did not increase beyond 10. the Although the number of predictions did not increase after increasing the epochs beyond 100, it was able to identify an alphabet as an alphabet and a digit as a digit(for instance when A was given as an input it predicted it to be a ‘E’ when earlier it was identified as ‘0’). It was also noticed that as the **learning rate** was chosen to be a **smaller value**, then the predictions were more accurate.

**3. SLP Model**

The model is given **256 inputs** which are the elements of the **image matrix.** Followed by **257 weights** (including a **bias** term with **weight 1**). The output will be an **integer output** that will vary from ***i* =1 to *i* = 20** (for the indices of the alphanumeric pattern images).

The governing **equation** is given as:



Then ‘***y’*** is then passed through the **linear activation** function (f(x)=x) to get the final output.

Diagram

Description automatically generated

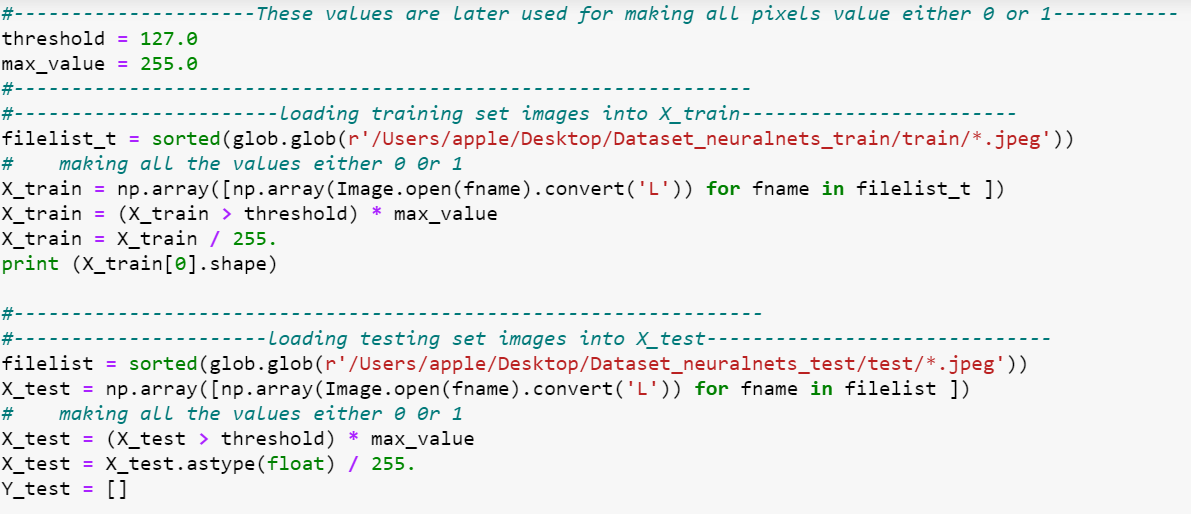
**Figure 5**: SLP architecture

The model was trained with 10 alphanumeric images of our choice on the best interest of better prediction of unseen images in the test set. Later, testing was done on all the 20 images.

**4. Python code for SLP**

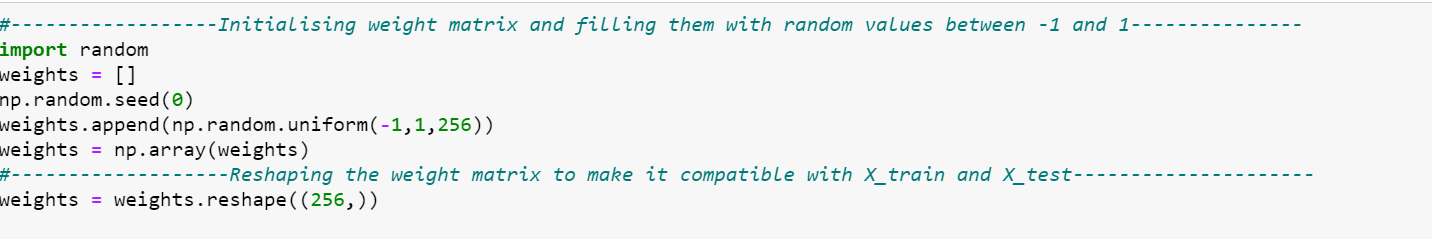
***Generation of Dataset***

The following code snippet depicts the conversion of 20 cropped images into black and white images based on a threshold value (127.0)



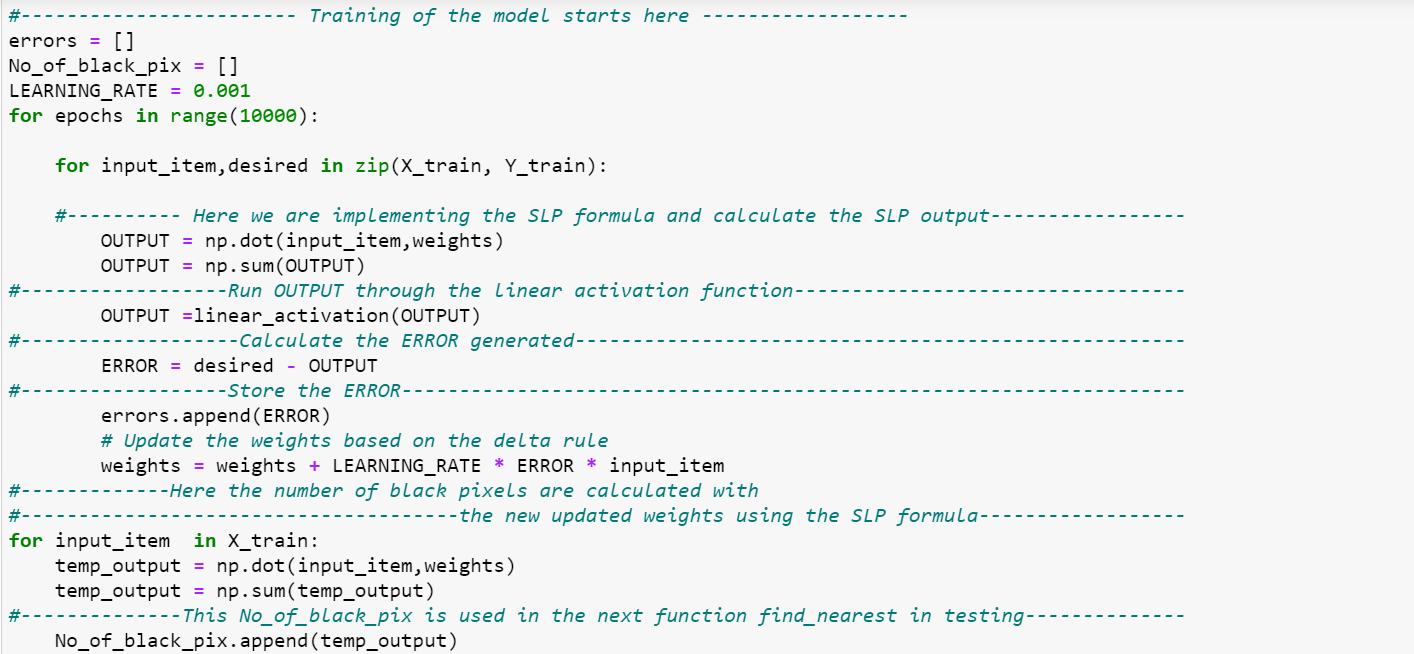
***Generation of Weight Matrix***

The weight matrix was generated randomly with values ranging from [-1,1]. The following snippet of code illustrates the creation of the weight matrix:



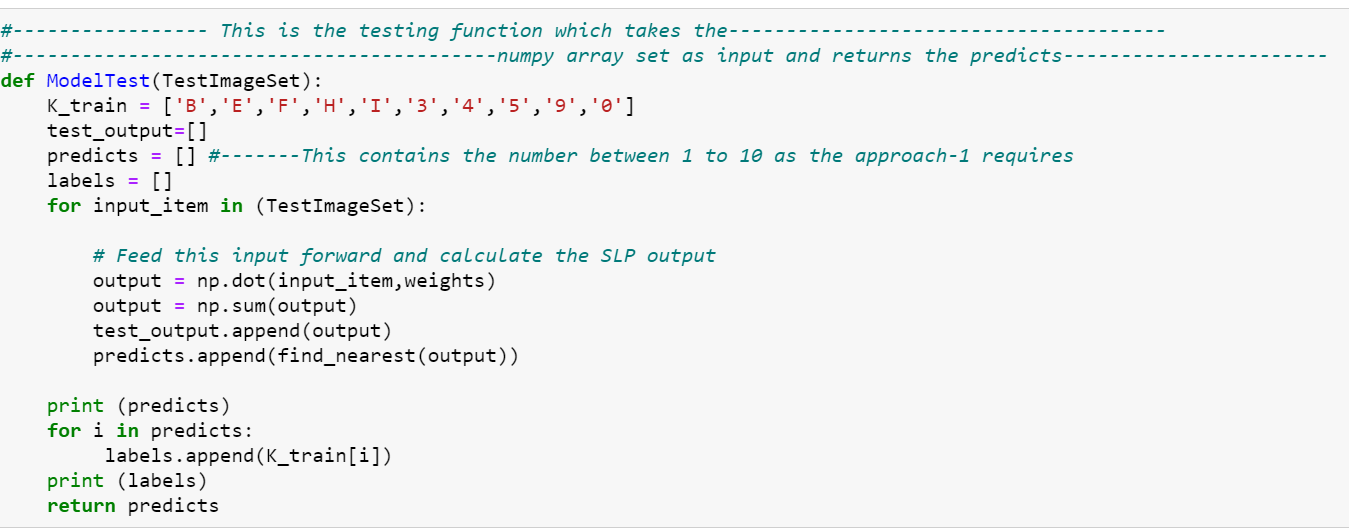
***Training***

The dot product of the image matrix which contains 256 values (0 or 1’s) with the weight matrix is taken which is then passed onto the activation function that is a linear activation function (f(x)=x). The output of the model is correlated to the corresponding value by counting the number of black pixels. So, if the input has a certain number of black pixels which is stored in an array( the same is done for all the inputs), then, the output is the index of the image that has a value which is closest to one of the values in the array.During training the weight matrix is updated using the ***delta learning*** rule.This ***rule*** states that the modification in the weight of a node is equal to the multiplication of error and the input.



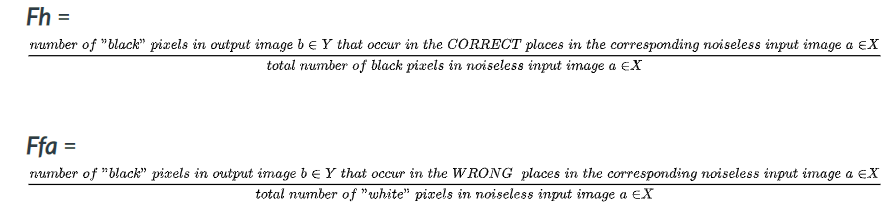
***Testing***

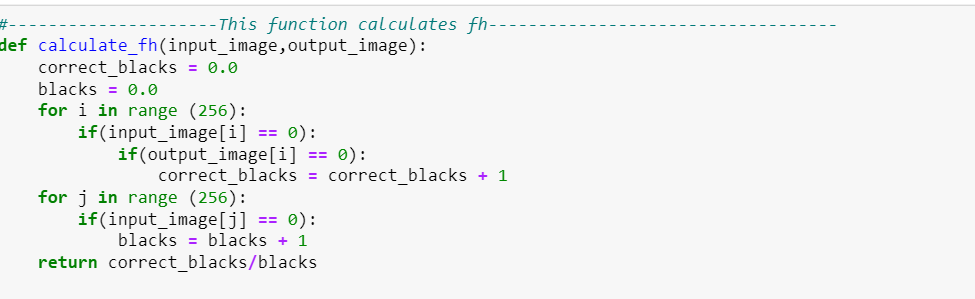
In testing, all the 20 images are each given as input and the output is the index of the alphanumeric pattern image that is (ideally) closest to the input image.

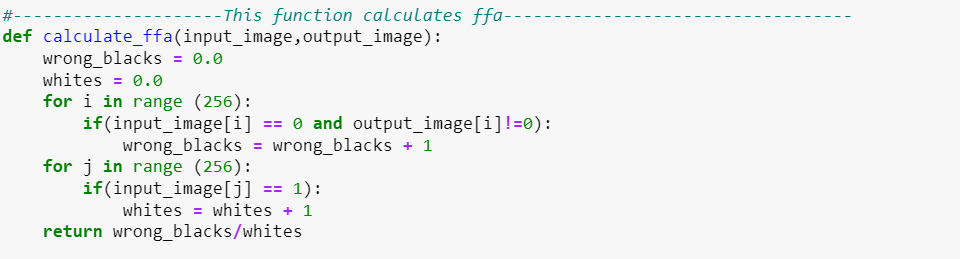
******

***Calculation of performance metrics***

The following metrics **Fraction of hits** and **Fraction of False Alarms** are calculated with the help of the functions displayed in the code snippet. Then, a graph of Ffa vs Fh was plotted which is shown in **Figure 6.**

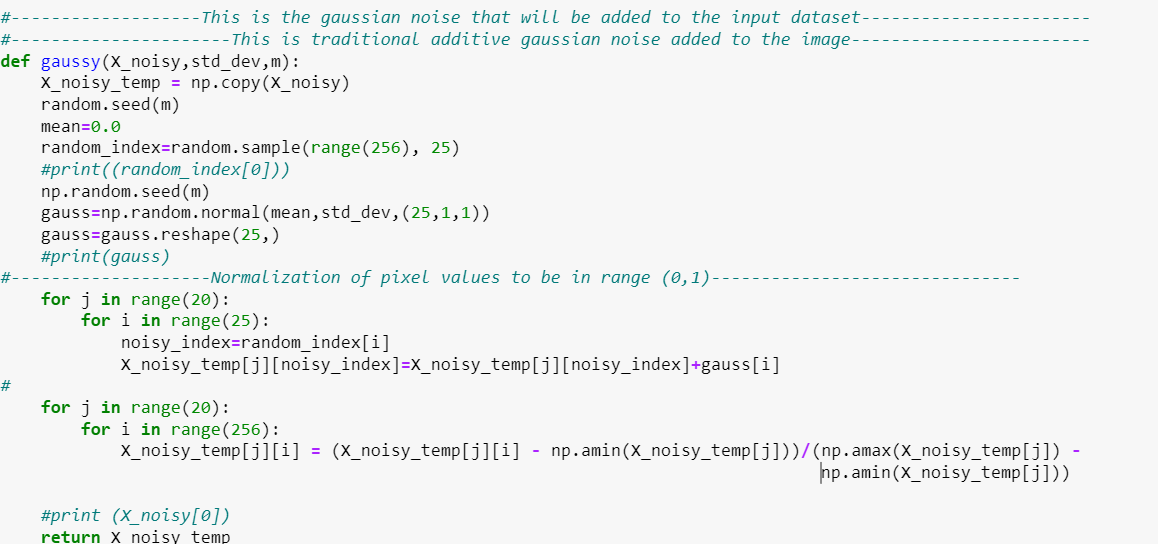
******

******

******

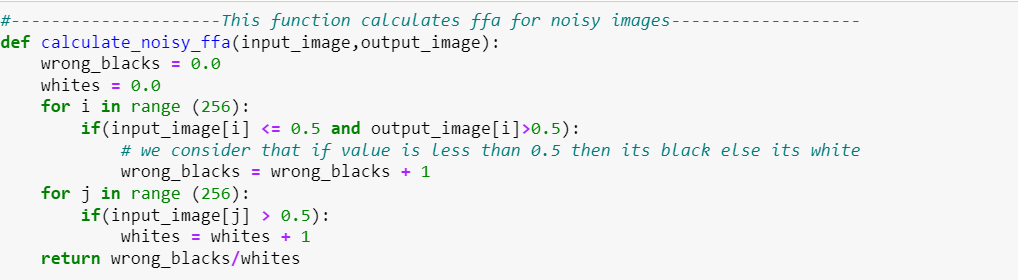
***Introducing noise to the input images***

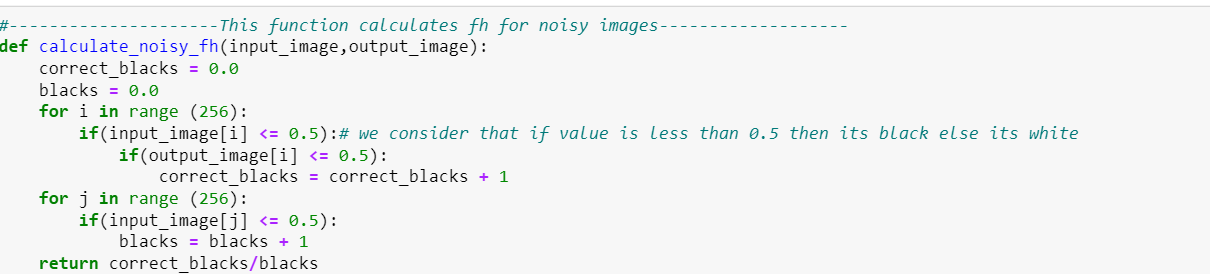
The input images are perturbed by adding gaussian noise. The following function helps in implementing it.



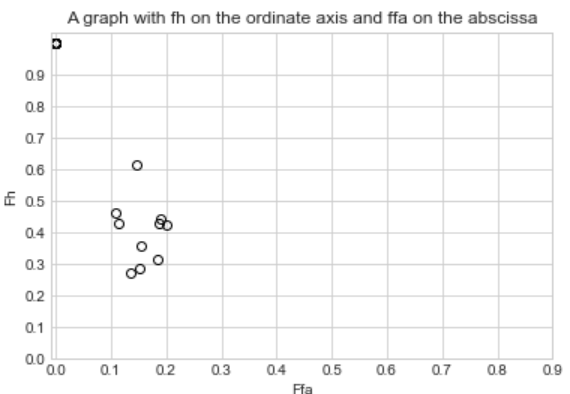
***Calculating performance metrics for noisy images***

Following the training of the noisy images, the Fh and Ffa values are calculated for the noisy images as well with help of the following functions. The functions to calculate Fh and Ffa for noisy images used the **threshold** value which is set to be **0.5** ( that is, **values greater than 0.5 were considered as white pixels and those less than 0.5 were considered as black pixels**).





**5.SLP OUTPUT RESULTS FOR NOISELESS INPUT IN TERMS OF FH AND FFA GRAPH**

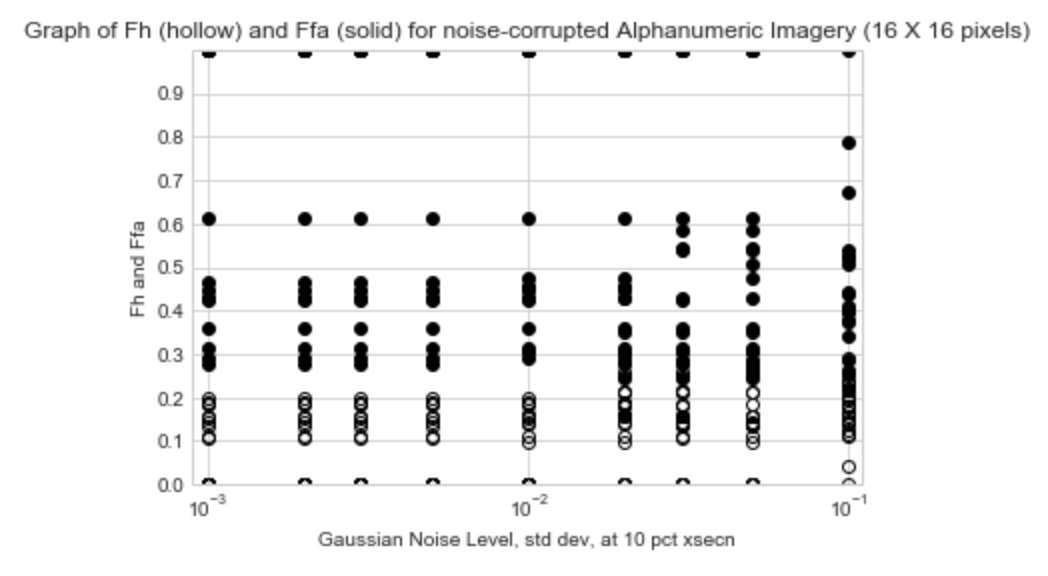


**Figure 6**: Performance graph for noiseless data

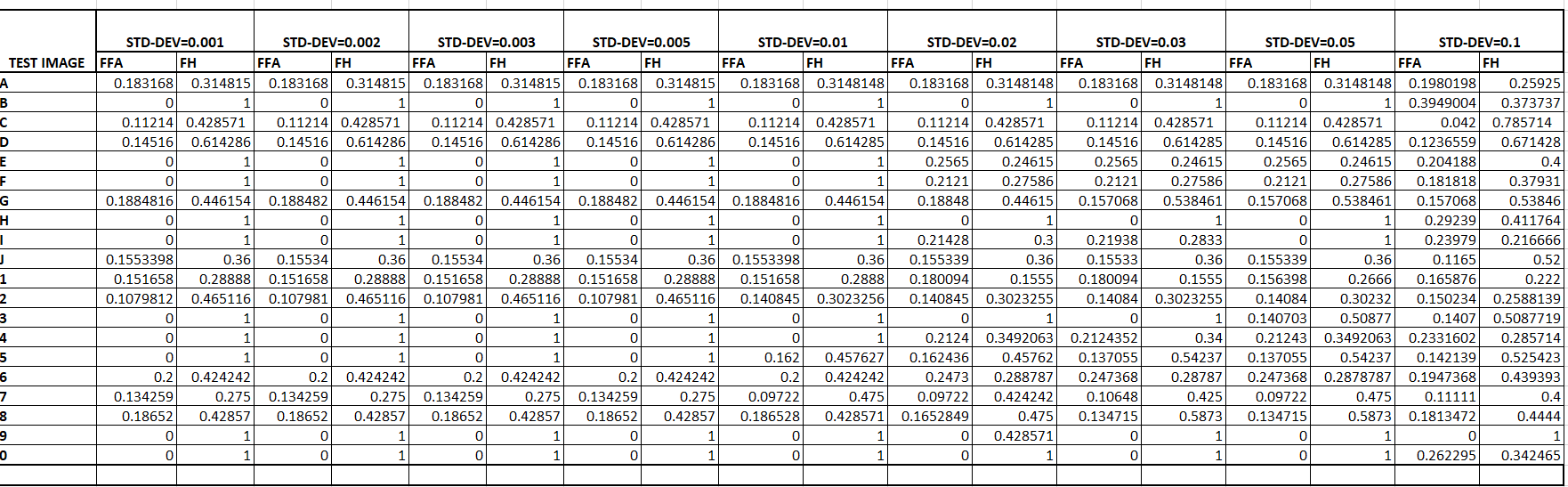
**6. SLP OUTPUT RESULTS FOR NOISE-CORRUPTED IMAGES**

The images were then perturbed with gaussian noise and then the images were then tested. Following which Fh and Ffa values for each standard deviation were calculated and tabulated below in Table 3. A graph was also plotted with **standard deviation** on the **abscissa** and the **Fh** and **Ffa** values on the **ordinate** axis (Figure 7).

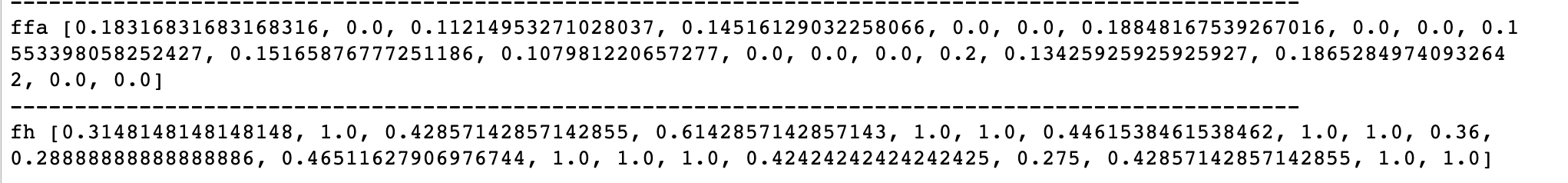
Although different levels of noises were added it was observed that there were miniscule changes in the Fh and Ffa values. Furthermore, as the standard deviation of noise level increased, the Ffa and Fh values had minute changes which possibly indicates that the changes in the image matrix were so minute with current corresponding criteria ( counting the number of black pixels). It is observed that at low standard deviation a similar amount of noise is added at the 25 pixels that are randomly chosen, so there is no much impact on the performance of the model uptil the standard deviation 0.005. From there as the noise amount became diverse (due to high standard deviation) the model started to misinterpret the inputs and we can clearly see the changes in the performance. At higher standard deviations like 0.05 the model correctly interpreted 5 inputs correctly and at 0.1 it only classified 1 input properly. This means that as the deviation is increasing the model is struggling.



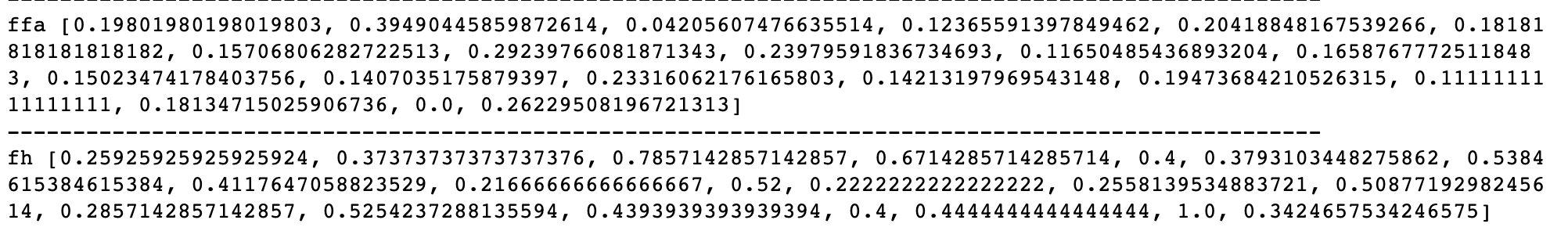
**Figure 7**: Performance graph for noisy data



**Table 3**: SLP response to noisy input



**Figure 8**: Noisy Fh,Ffa values (SD = 0.001)



**Figure 9**: Noisy Fh,Ffa values (SD = 0.1)

**7. Discussion**

The model being a primitive model, performs well as an autoassociative perceptron, it outputs the image that was given as the input during training. Since there are no hidden layers in the model, the scope of learning for the model is minimum. Hence, it can be observed that the model cannot exactly identify the alphanumeric character during testing. Although, it can identify an alphabet as an alphabet and a digit as a digit.The performance can be improved by trying to choose another criteria for correspondence during training. The choice of alphanumeric characters in training may also influence the efficiency of the model. To improve the efficiency while dealing with noisy data that is fed into the input during testing, another form of adding noise could have been chosen, for instance, here additive gaussian noise was implemented, wherein multiplicative gaussian noise could have given better results. We can clearly see that as the noise increased (standard deviation) the ffa value also increased, This can be used to say that the model is in the right track of learning. With more suitable changes as mentioned above it can be made more reliable and precise.

**APPENDIX**

An alternate approach was employed to observe and compare the differences in performance with respect to **approach 1**.

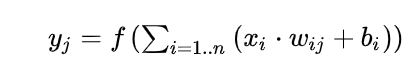
For approach 3 the input is the image matrix and calculations are performed on each pixel meaning each pixel undergoes training. This results in better performance and this is observed from the output of the model predictions. It was observed that the model predicts majority of the input data correctly. In case of noisy input data, the predicted output has reduced noise.

**Approach 3**

**Fully-connected Network with Image Output**

There will be 16x16 = **256 inputs**, and *w=***256x256 = 64K** **weights,** and **256 output nodes** that vary over the interval [0,1]. The **output image** will be **16x16** pixels (256 pixel values), hence the 256 output nodes.

The governing equation will be as:



The **256** output nodes will be assembled in normal (row-wise) scanning order to form the **16x16-pixel** output image.

1. **Input:**16x16 pixel images

2. **Output:** 256 nodes

3. **Number of weights:** 256 X 256 =64K weights.

4.**Weight matrix**(Initially, weights were generated randomly using a random function)

5. **Activation function:** sigmoid activation

6. **Learning rate:** 0.001

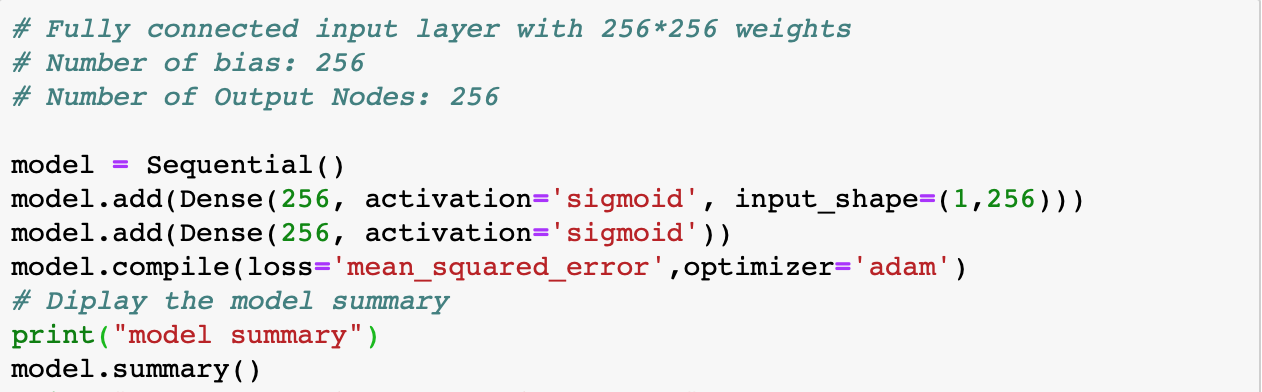
7. **Epochs**: Model trained on 10 images set = 350

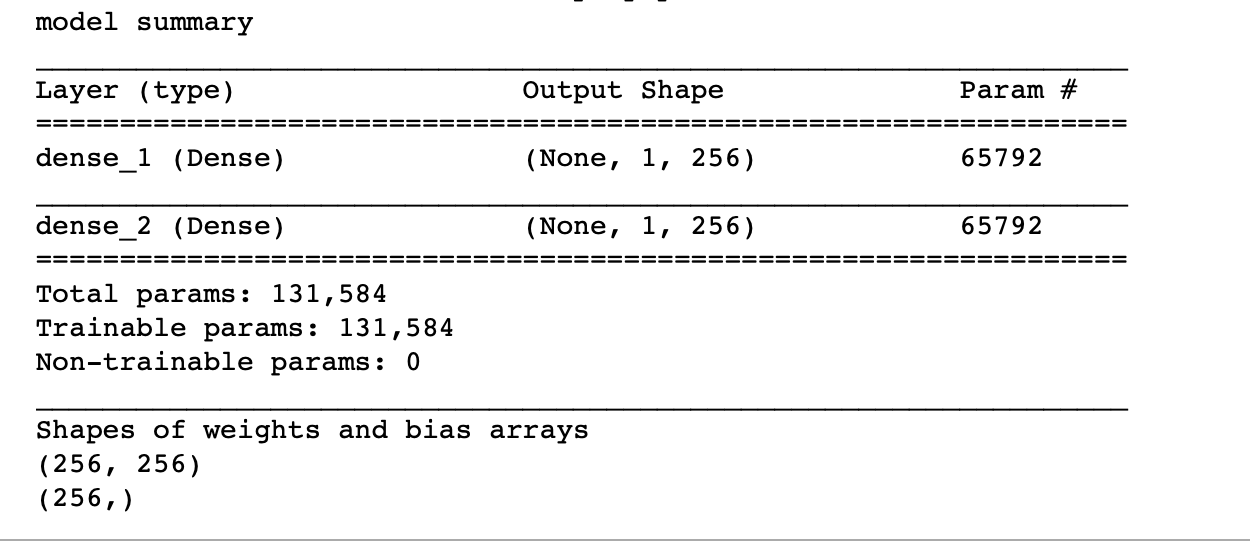
Model trained on 20 images set = 500

8. **Evaluation metrics:** Fh, Ffa

**A. MODEL ARCHITECTURE**

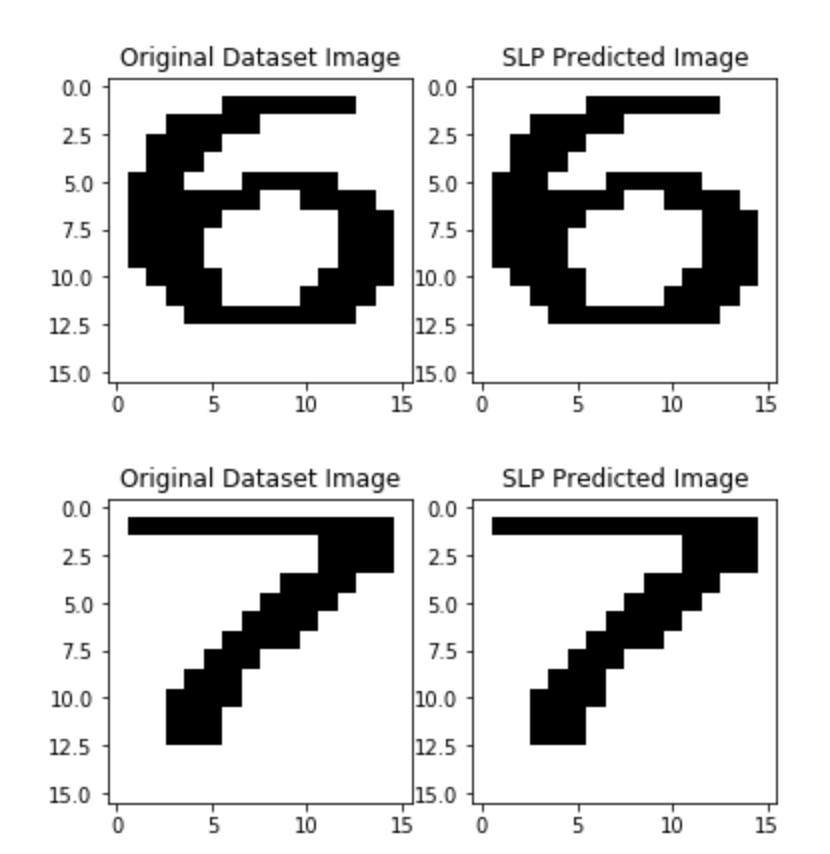
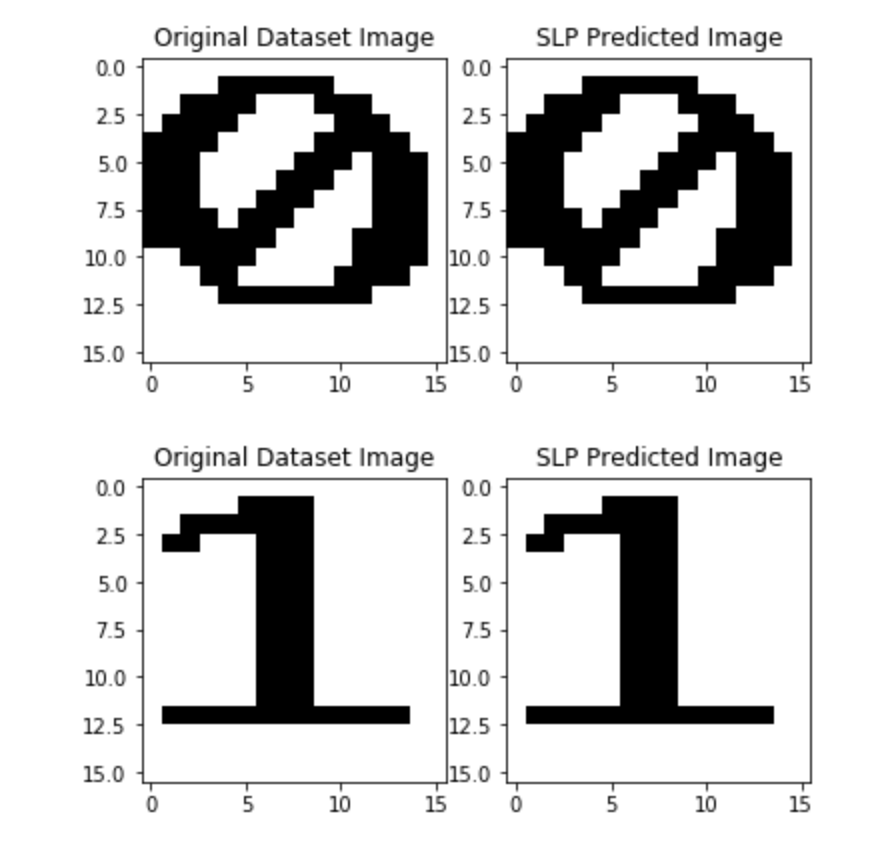
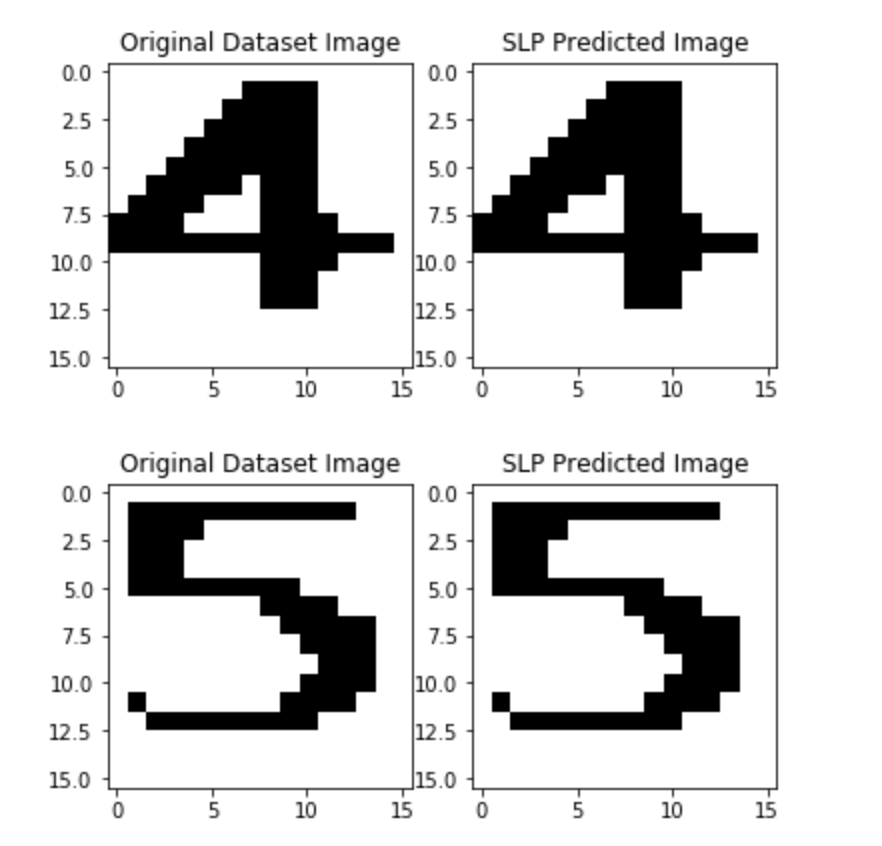
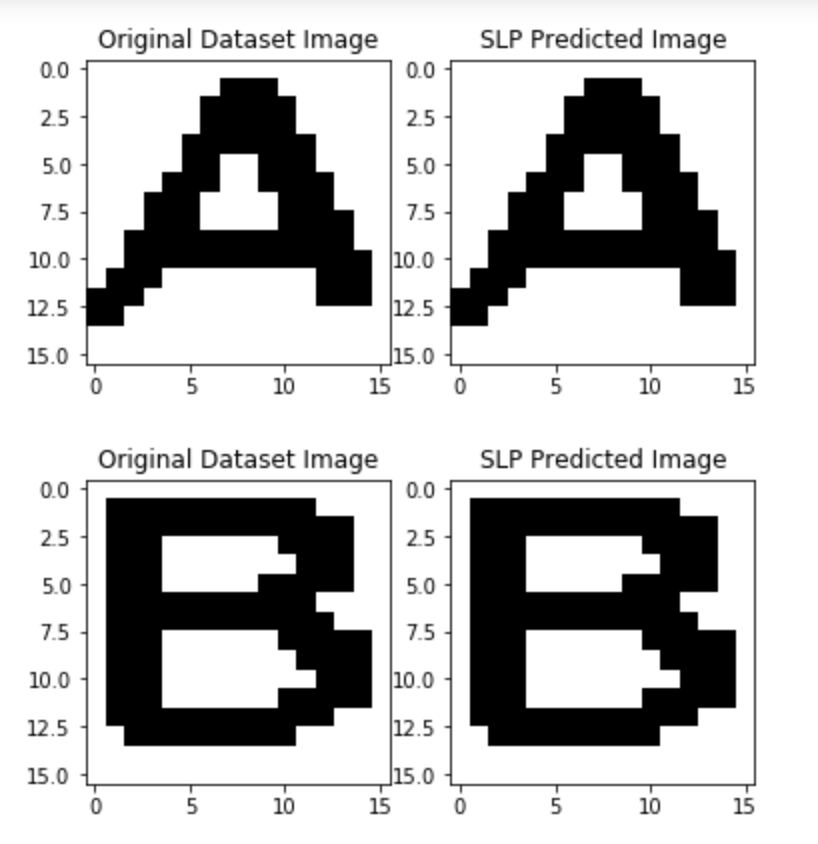
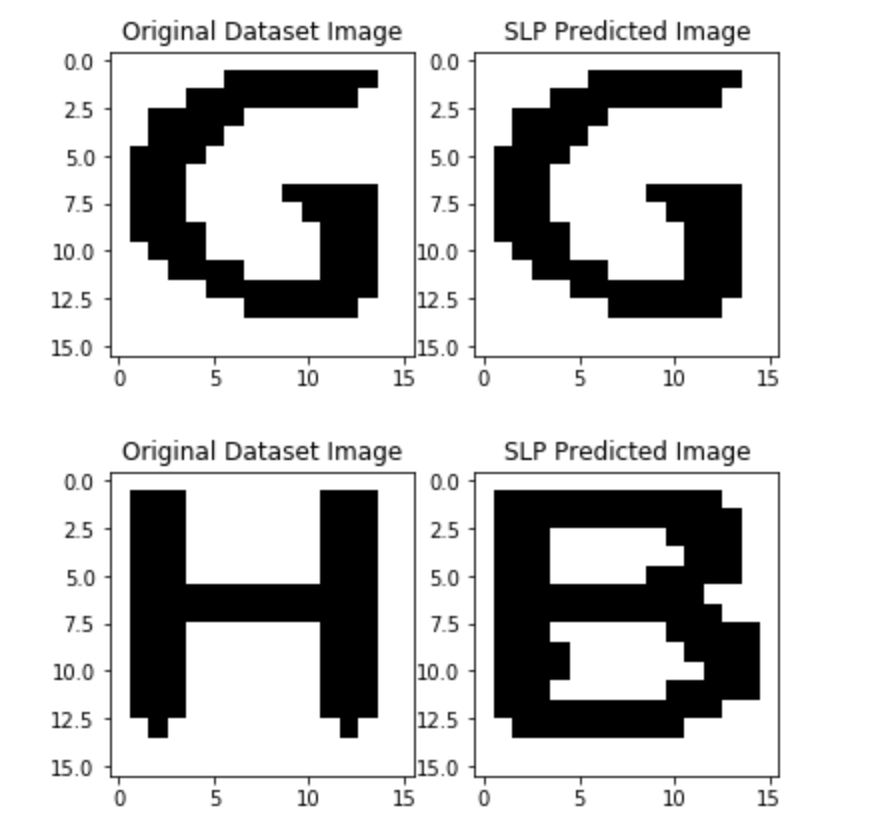
Every node in the input layer is associated with a random weight wi and is fully connected to the next layer. Hence, there are 256 bias values. Sigmoid function was used for the activation. It was noticed that the model performs better on sigmoid activation function than the linear activation function.



****

**B. TRAIN DATASET**

Ten images were selected from the dataset such that each image is uniquely distinguishable from the other. This is done so that better-unseen test images can be predicted while testing the dataset. Hence, the characters 0,1,4,5,6,7,A,B,G,H. In dataset generation, while converting the cropped images into black and white images, 0 is considered as a white pixel and 1 is considered as a black pixel.

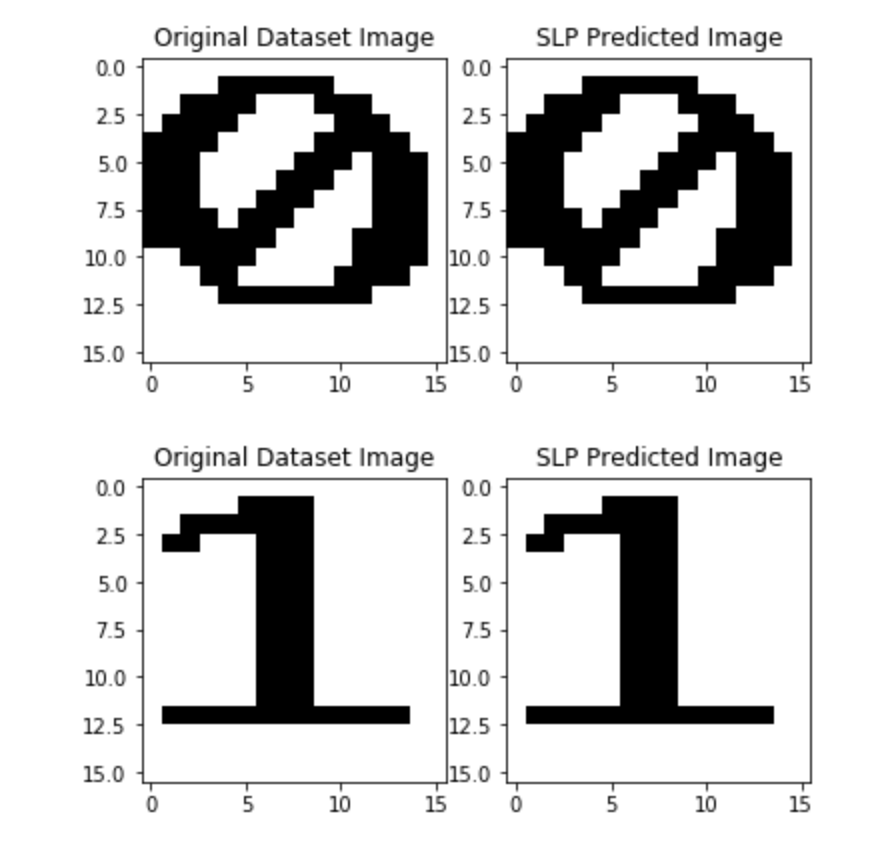
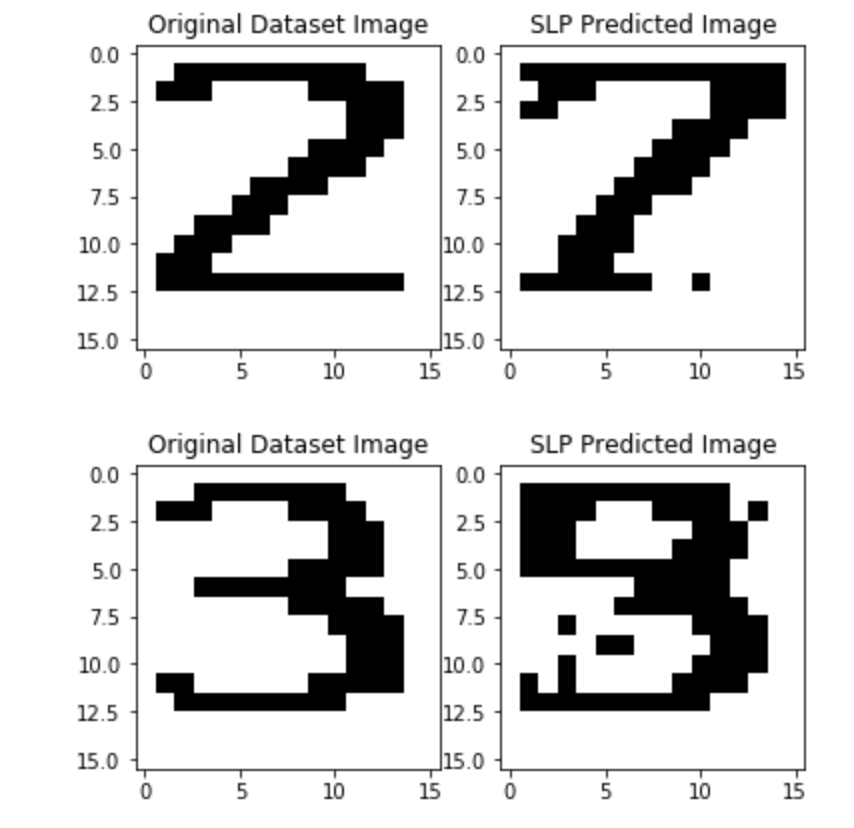
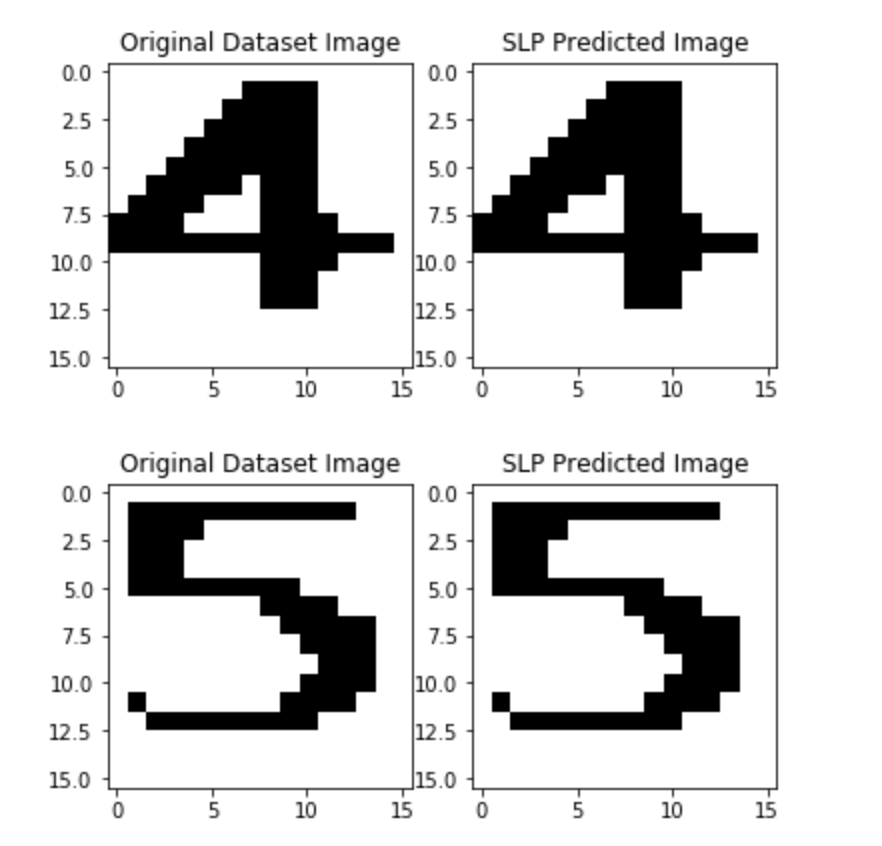
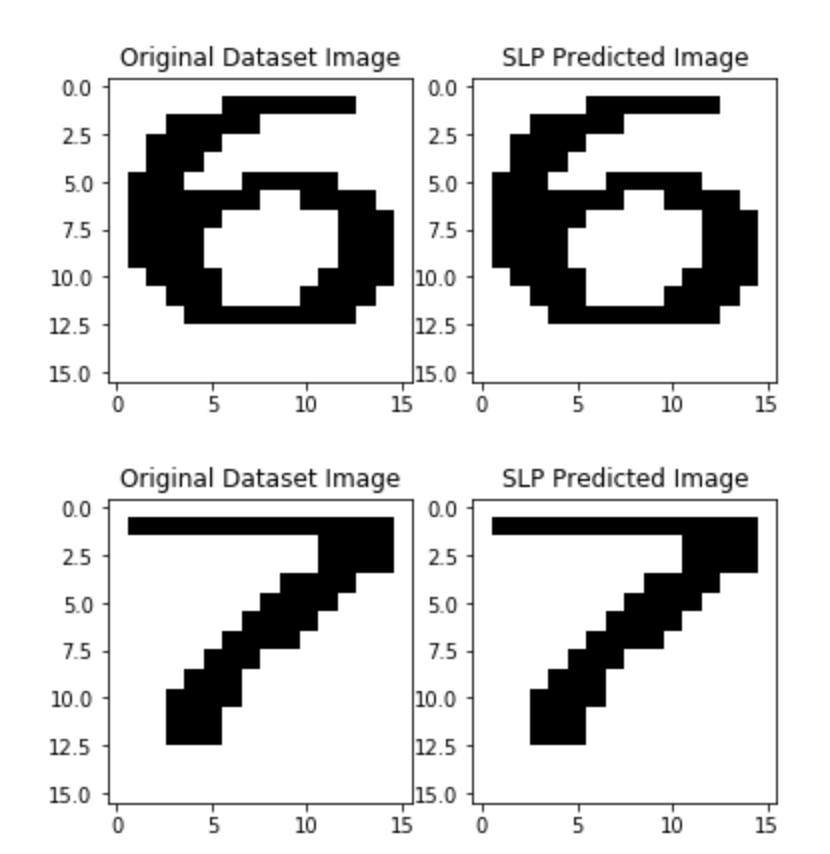
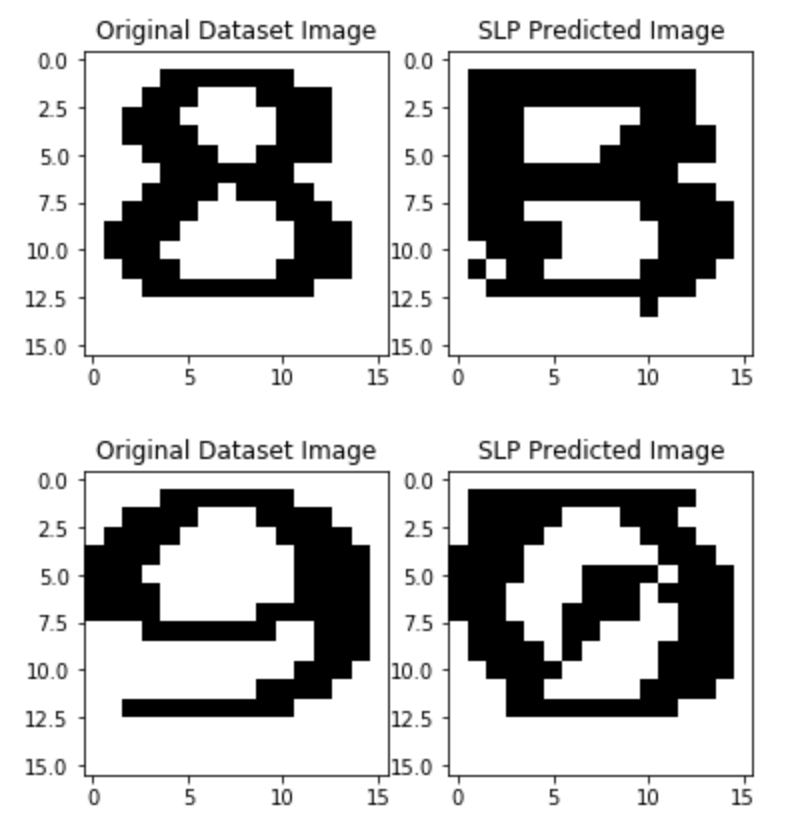
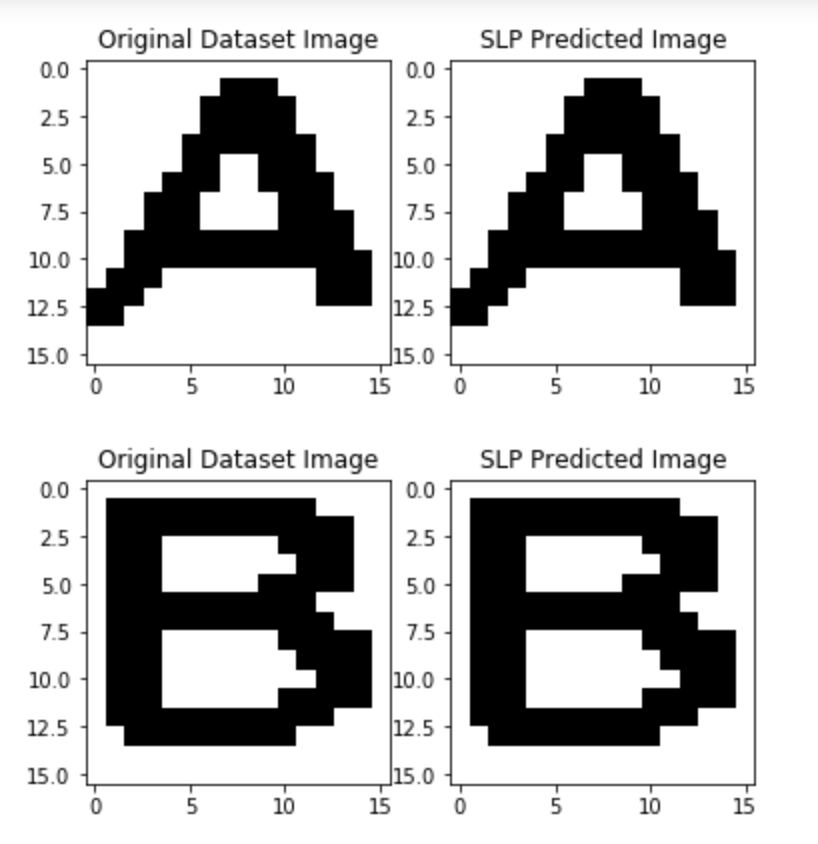
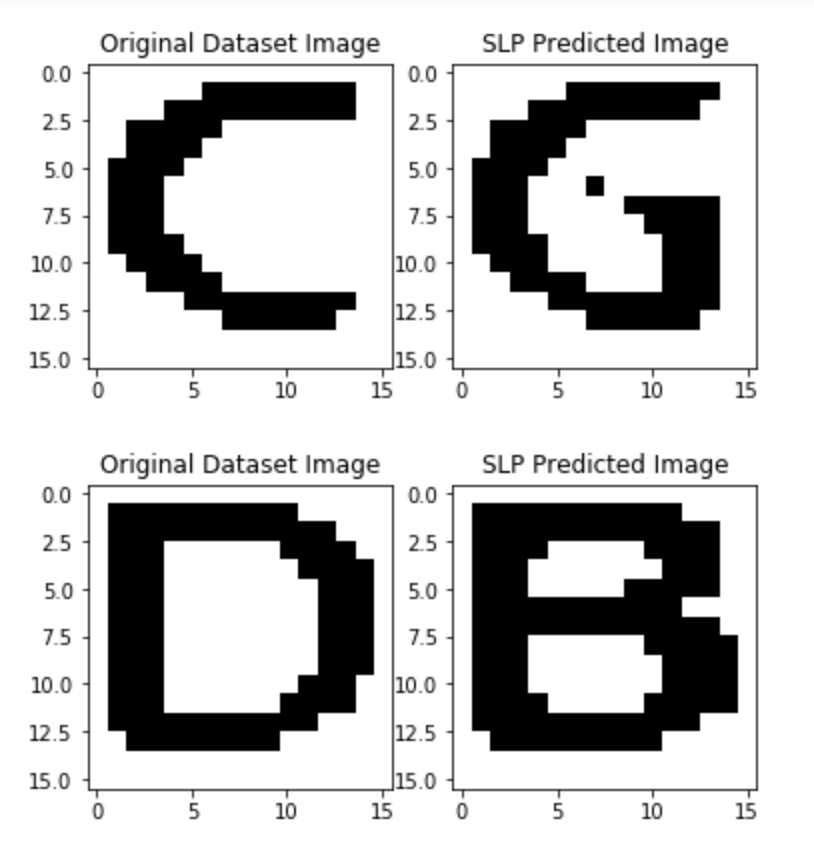
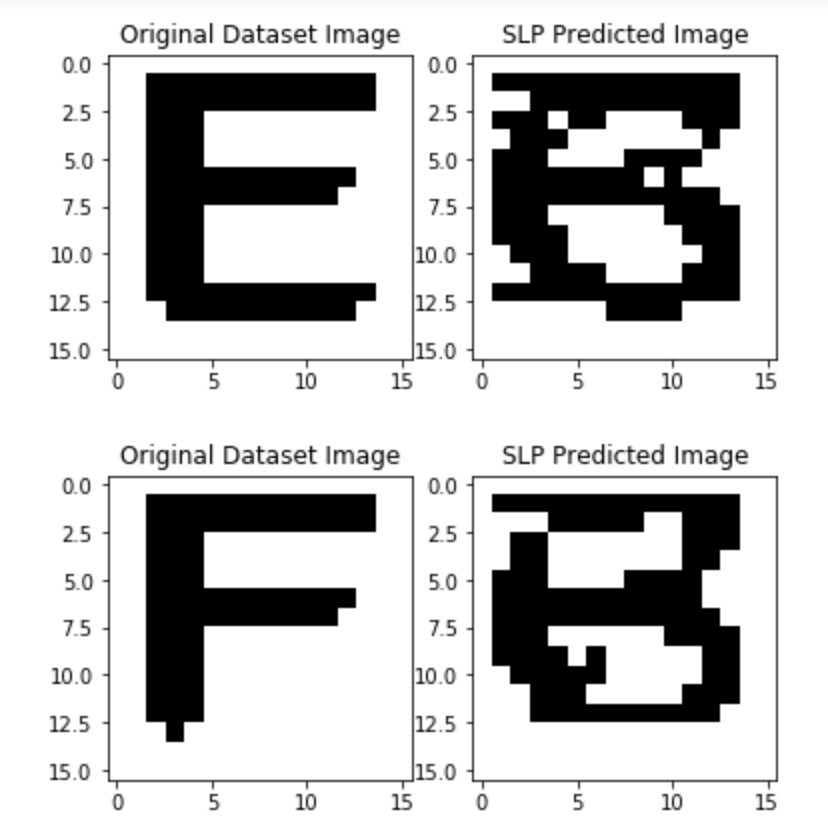
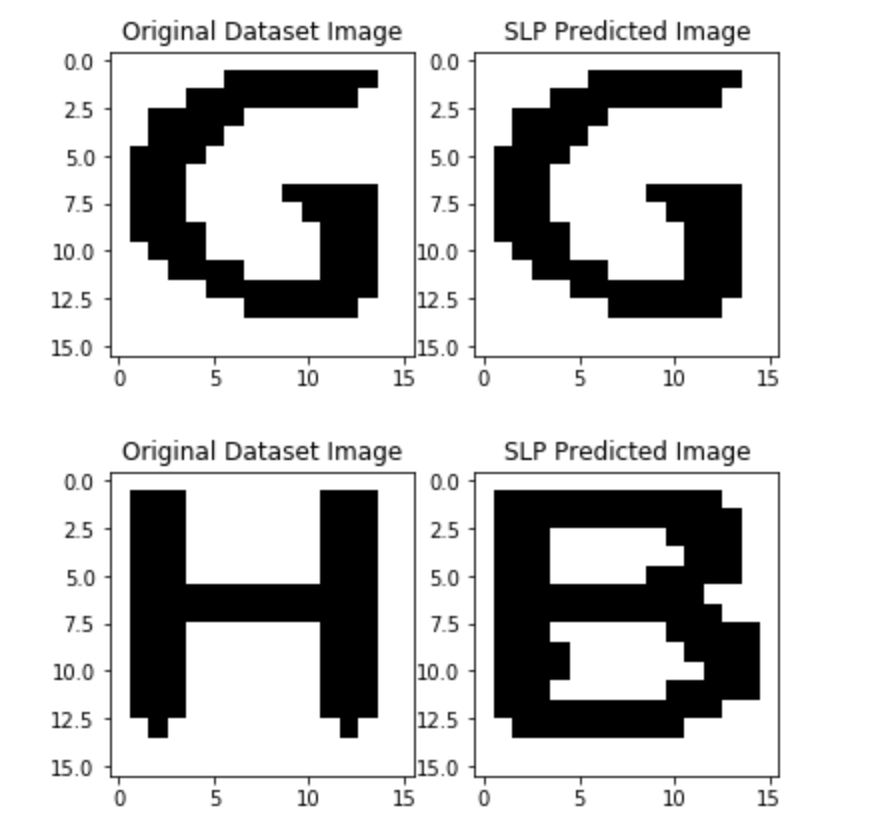
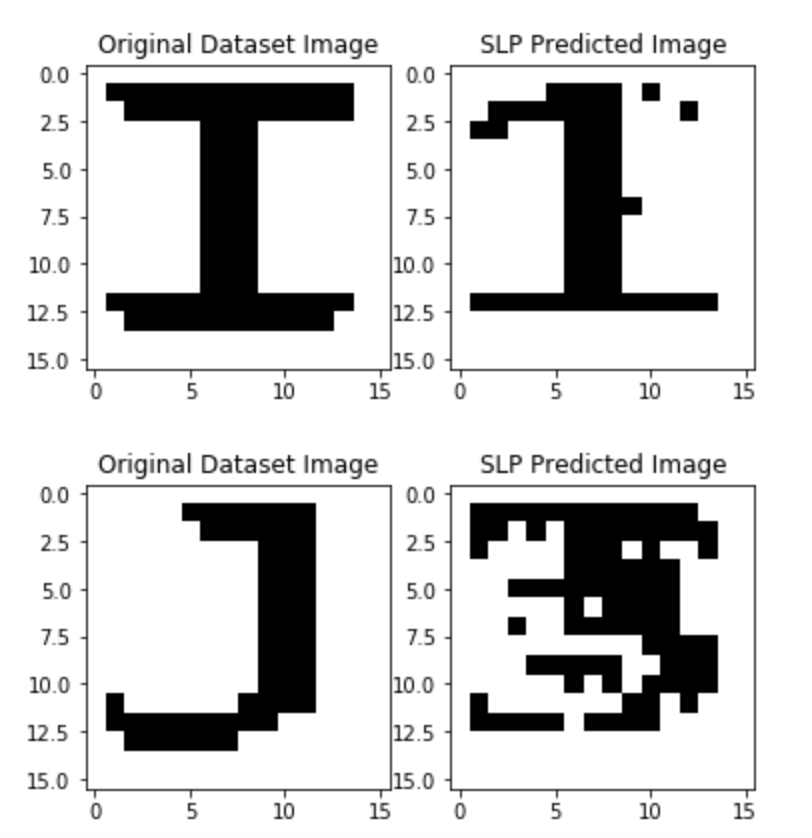


**Figure 10:** Train dataset

**C. SLP PERFORMANCE**

***1. Performance after training on 10 images:***

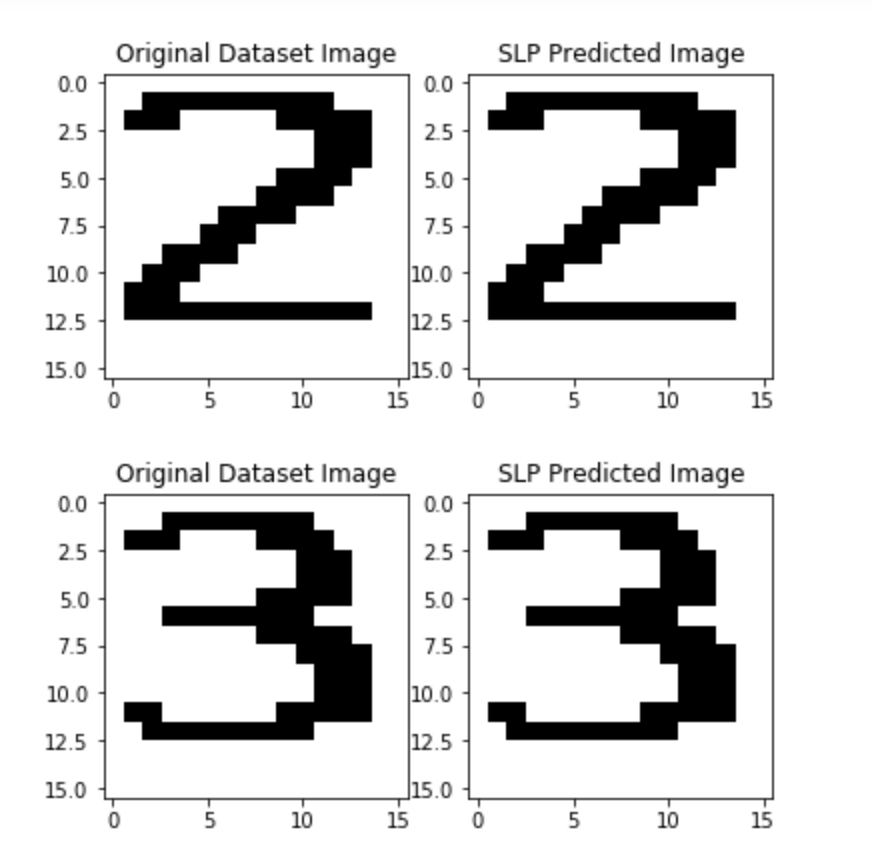
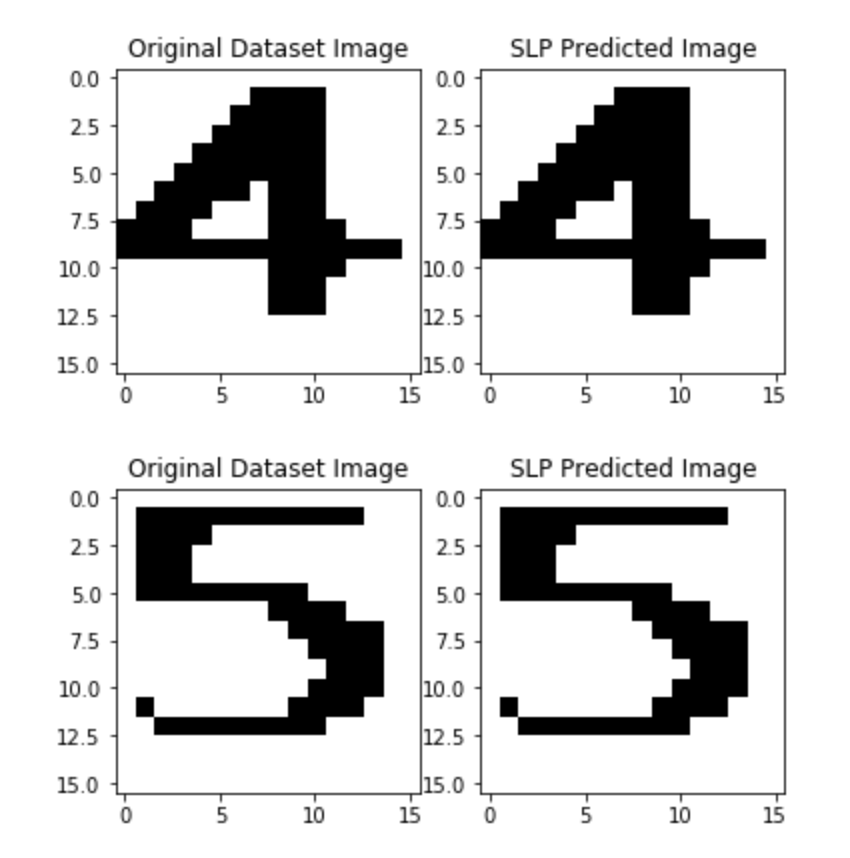
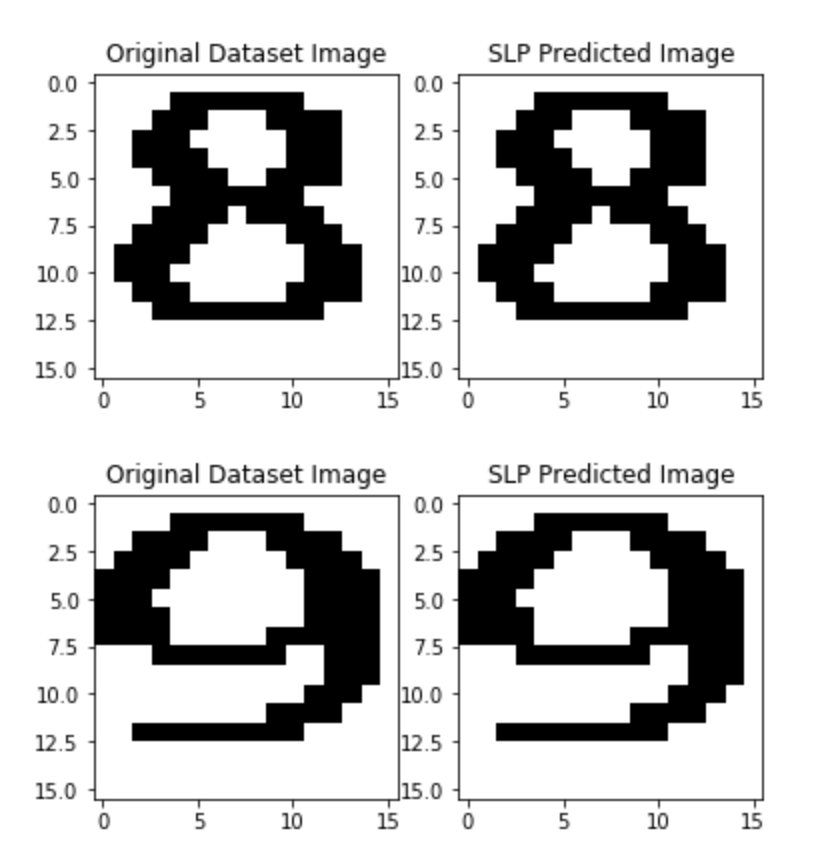
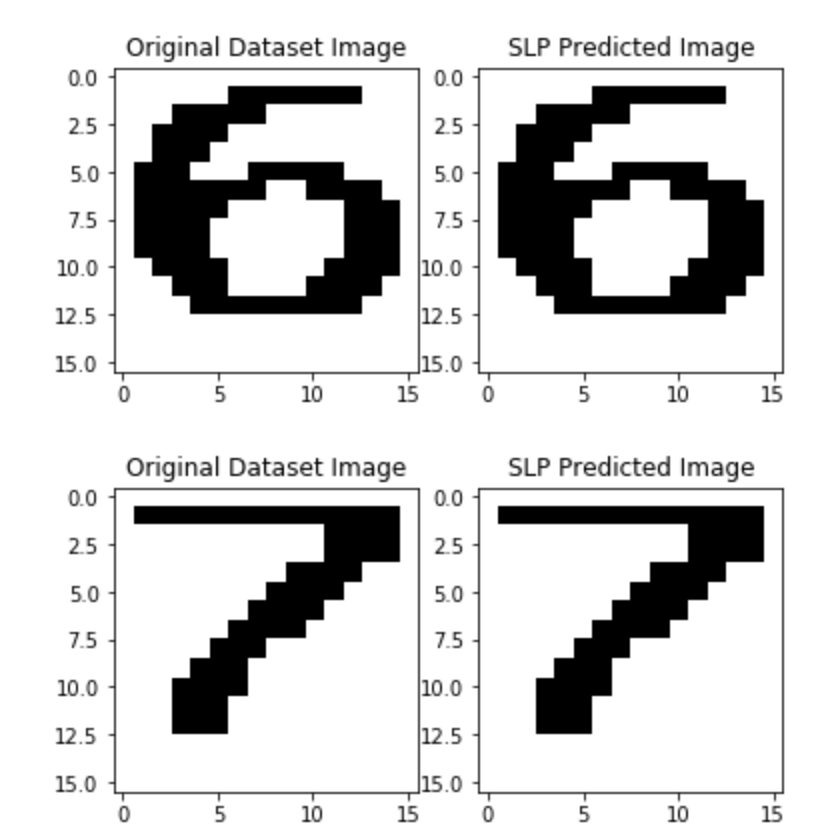
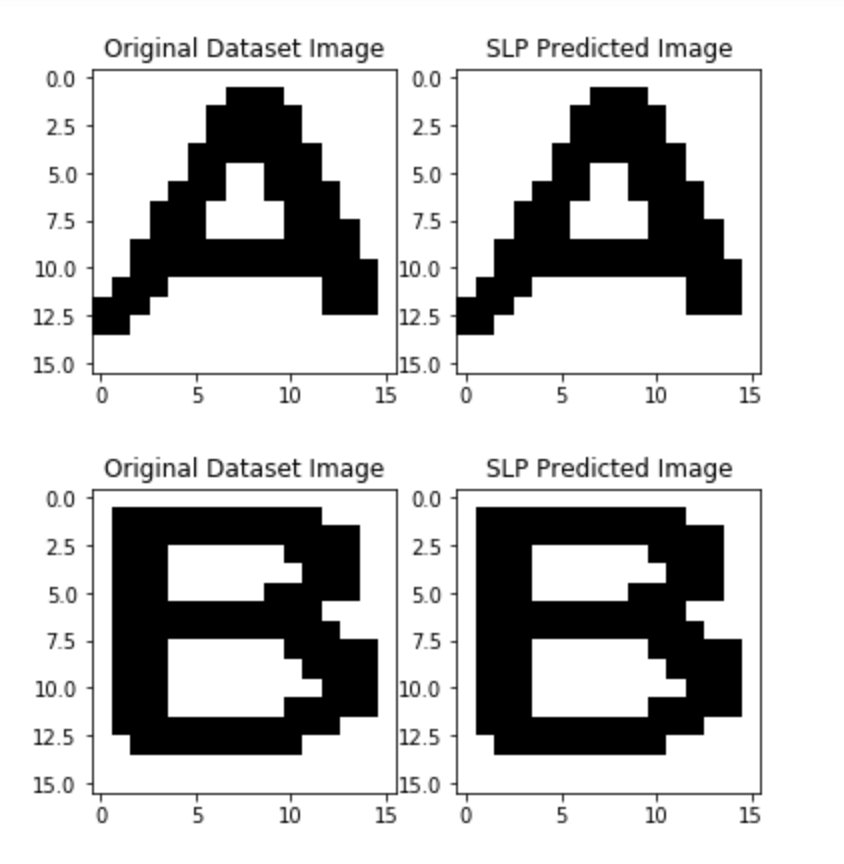
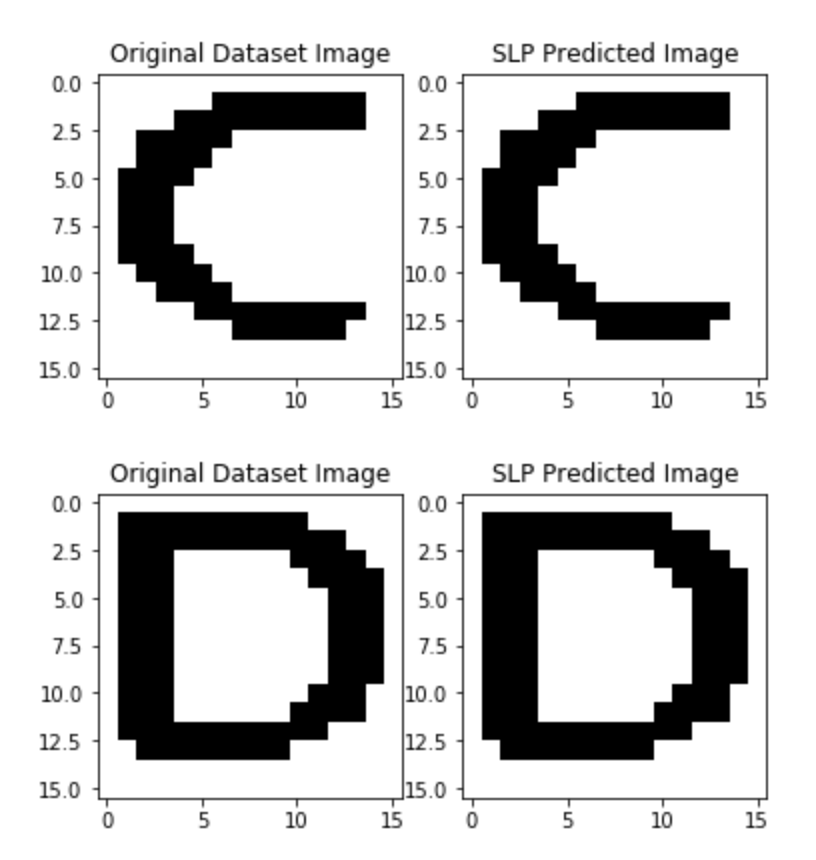
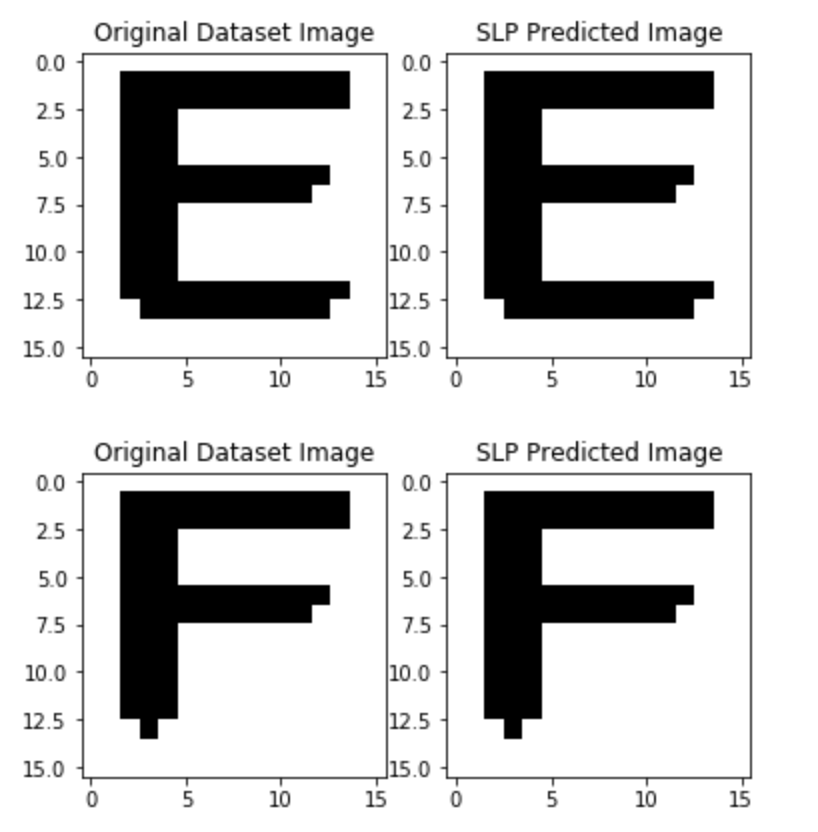
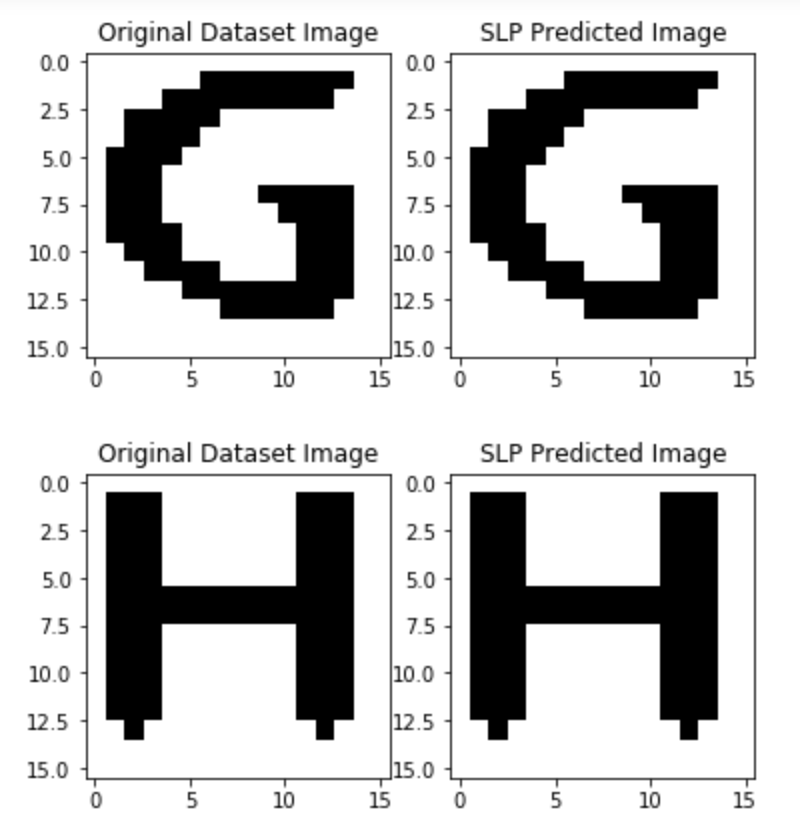
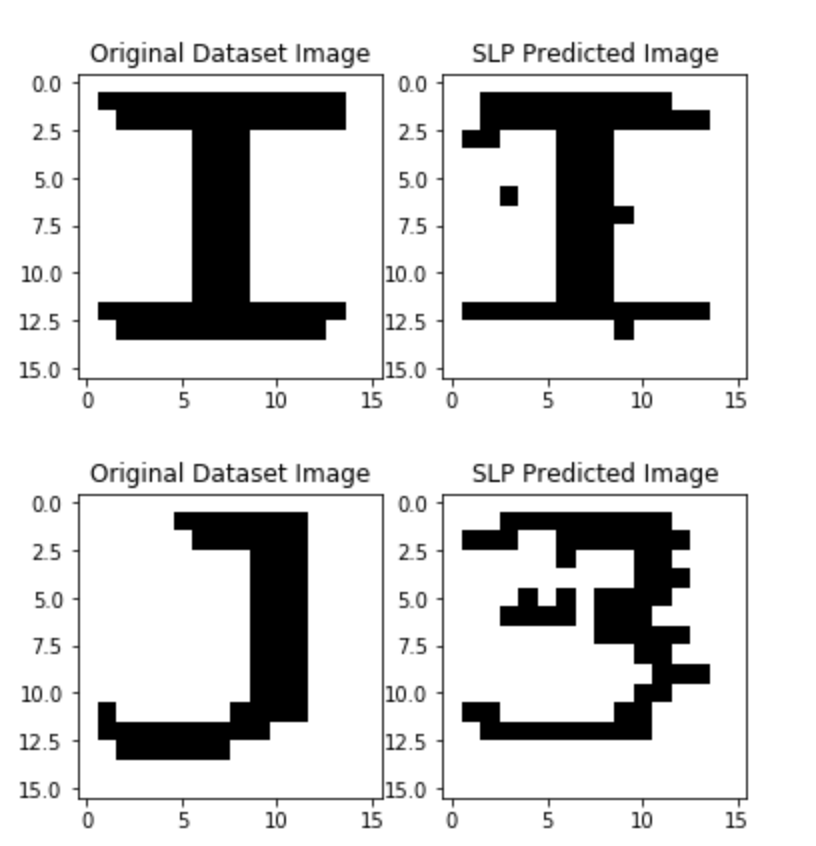
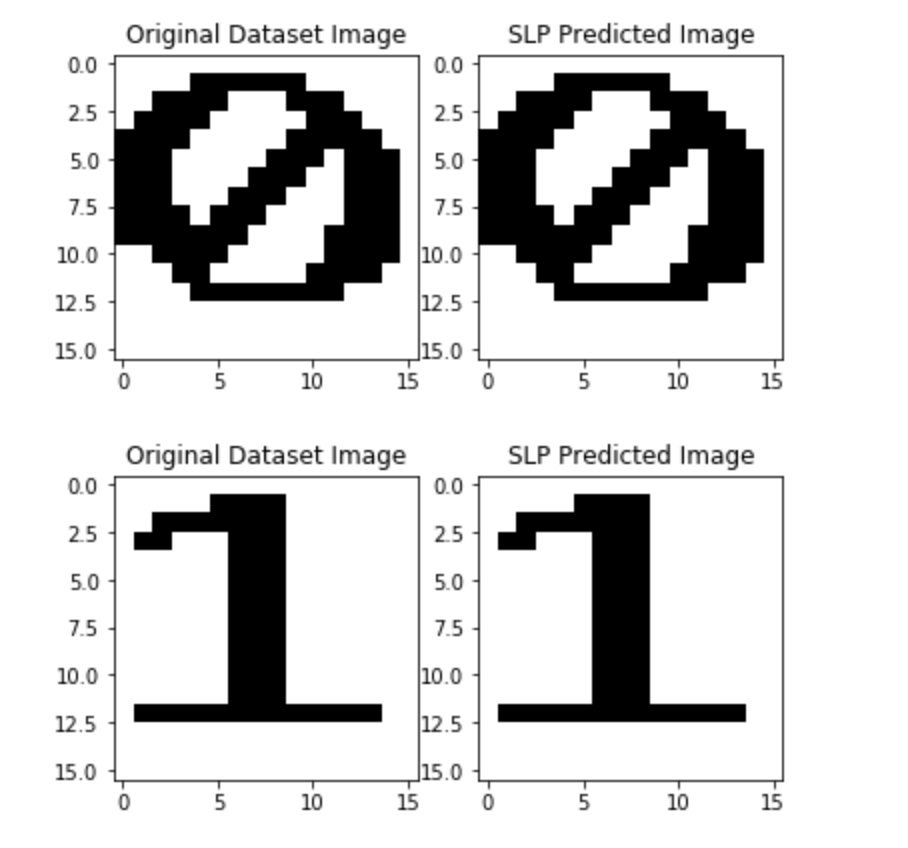
After approximately 300 epochs, the model was correctly able to recognize 40% of the images correctly. The other outputs closely resembled the input character.

****

**Figure 11:** Predictions of trained dataset(10 images)

***2. Performance After Training On Complete Dataset of noiseless images:***

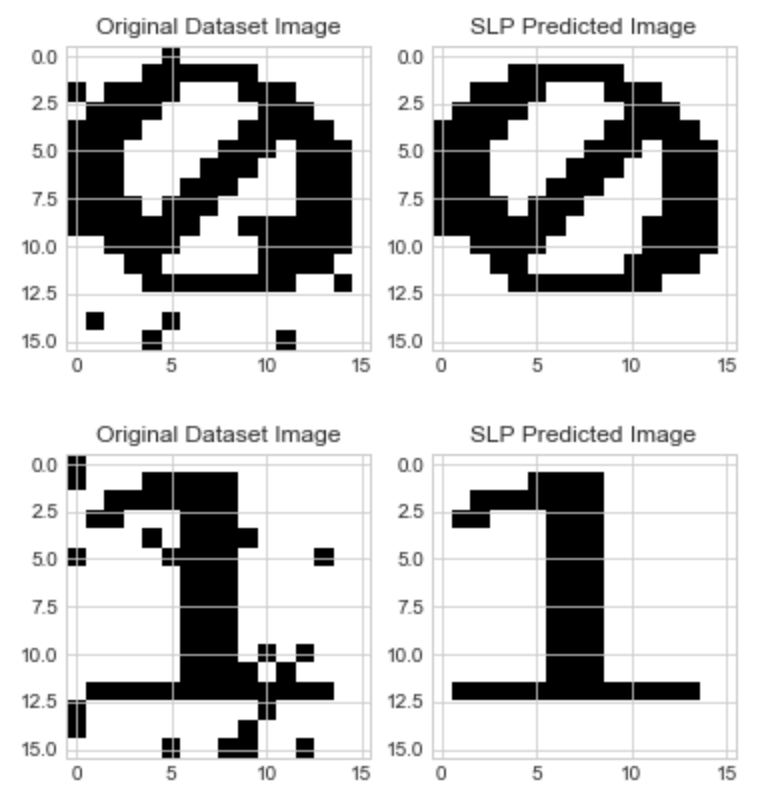
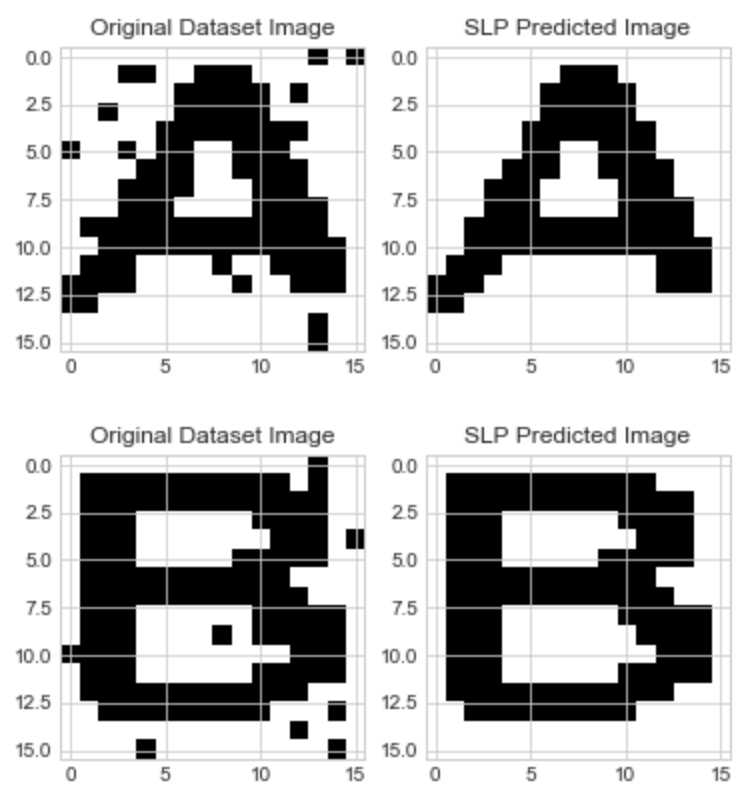
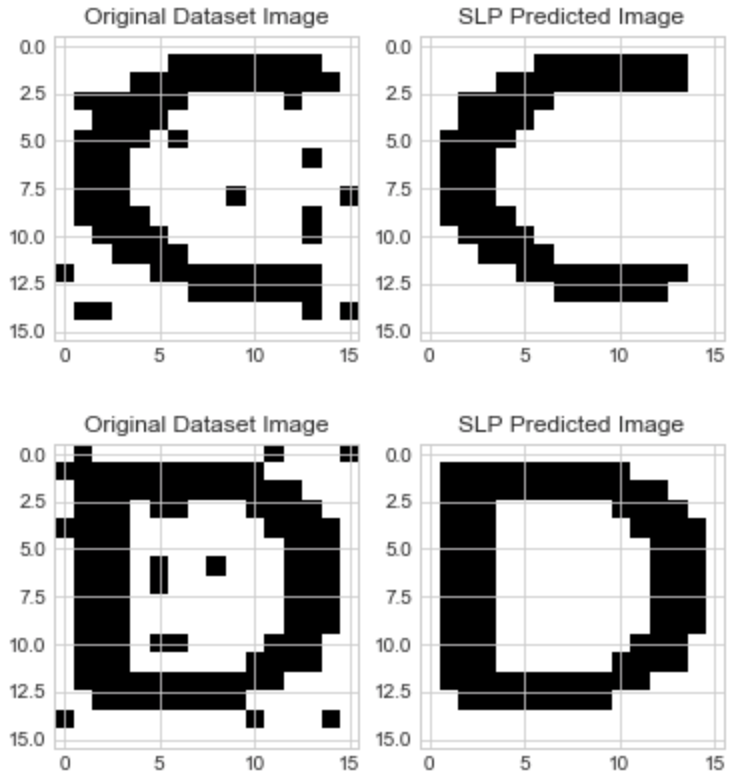
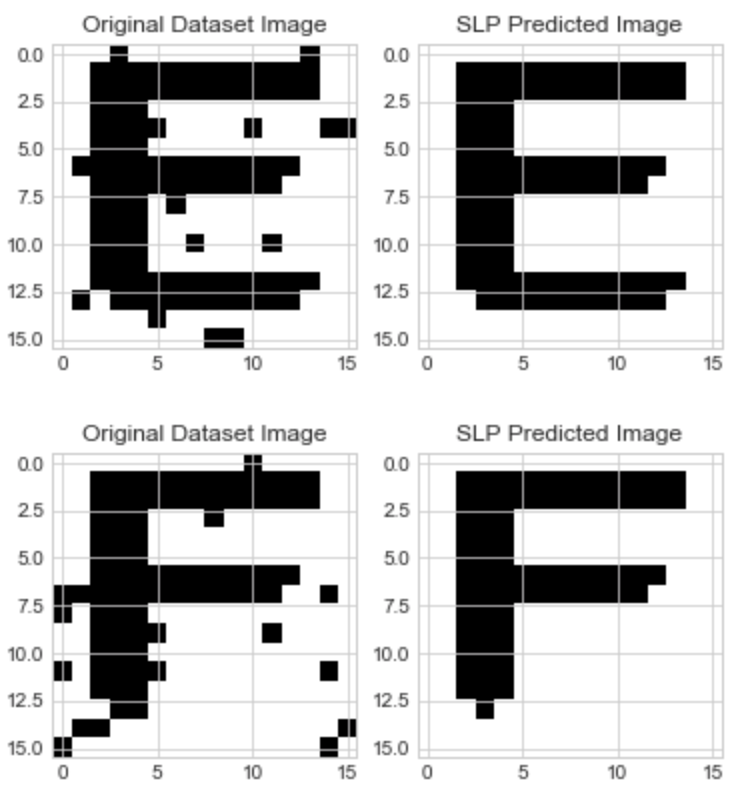
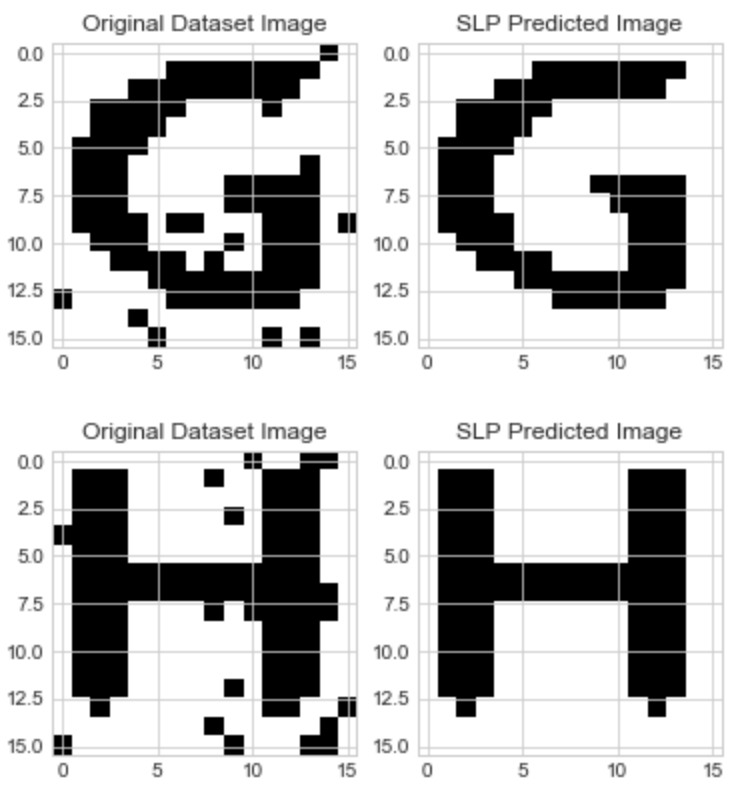
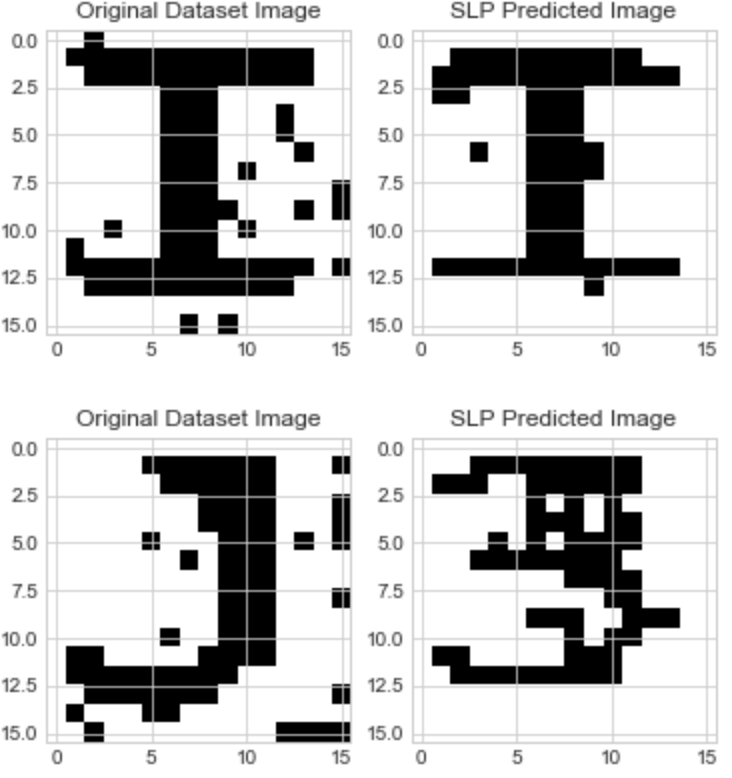
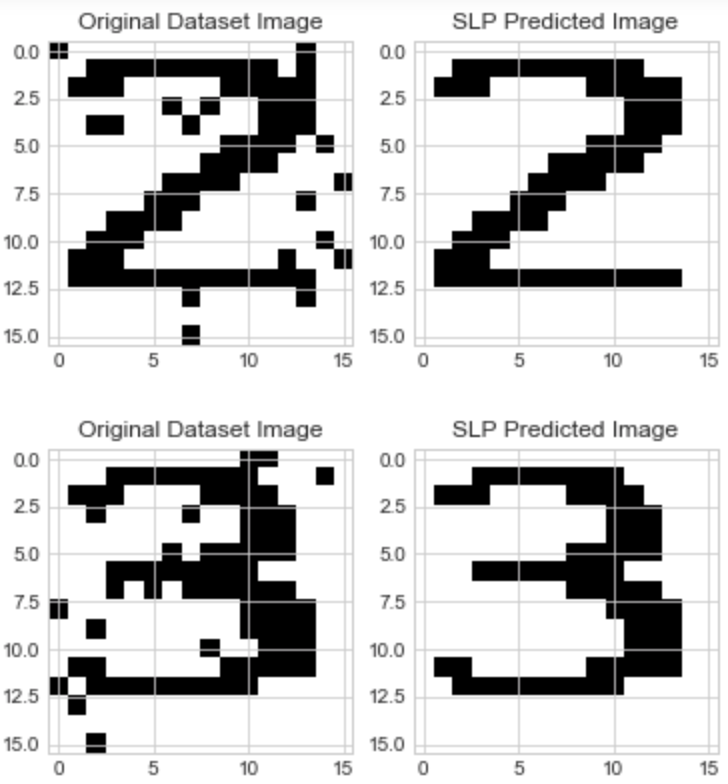
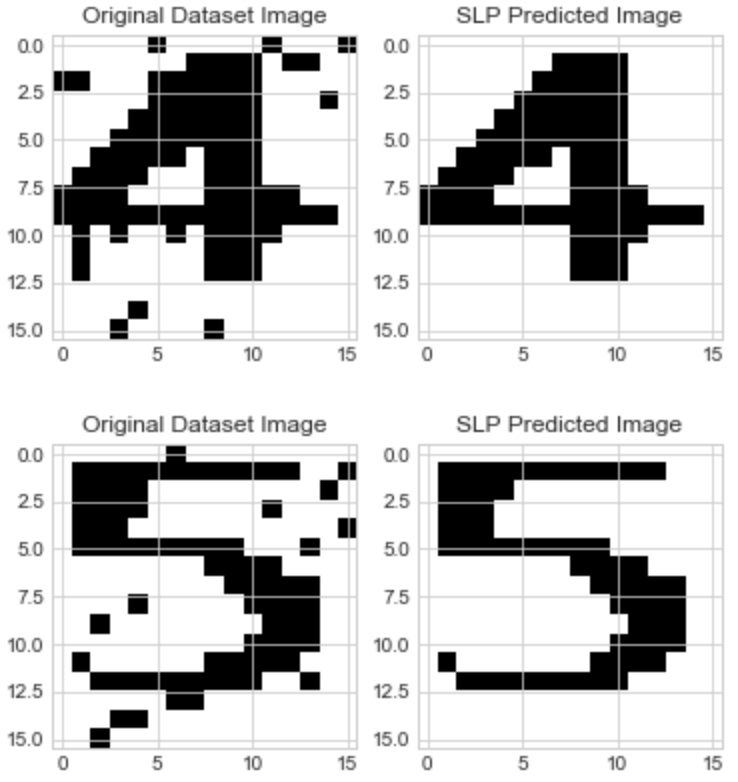
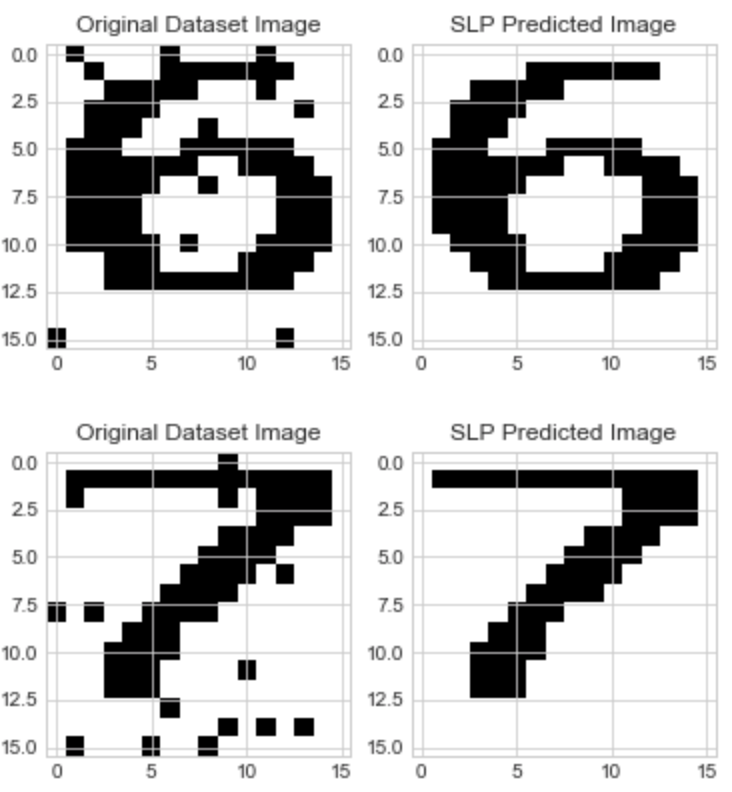
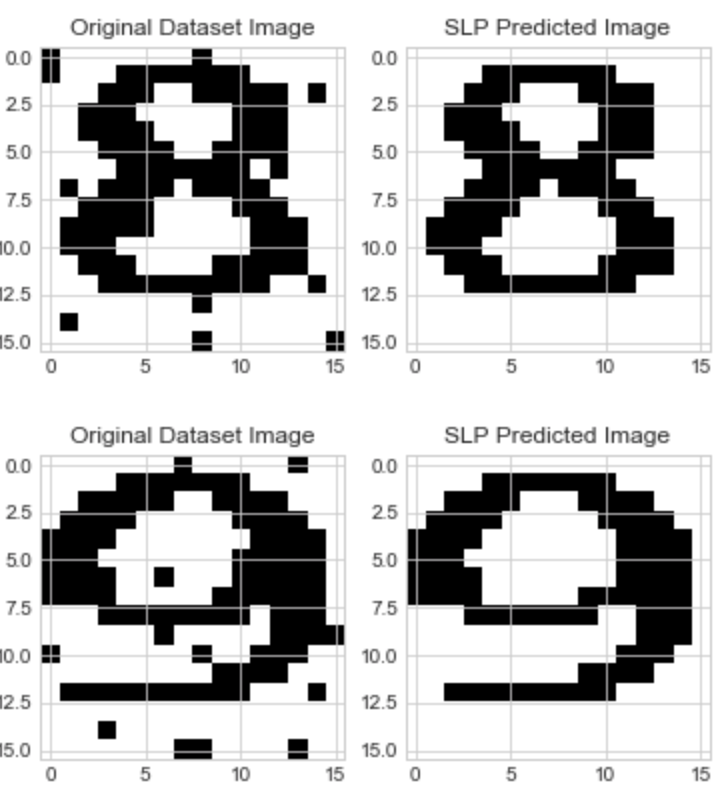
After approximately 500 epochs, the model was correctly able to recognize 90% of the images correctly. The other outputs closely resembled the input character.

****

**Figure 12:** Predictions After Training On Complete Dataset of noiseless images

***3. Performance of SLP on noisy input***

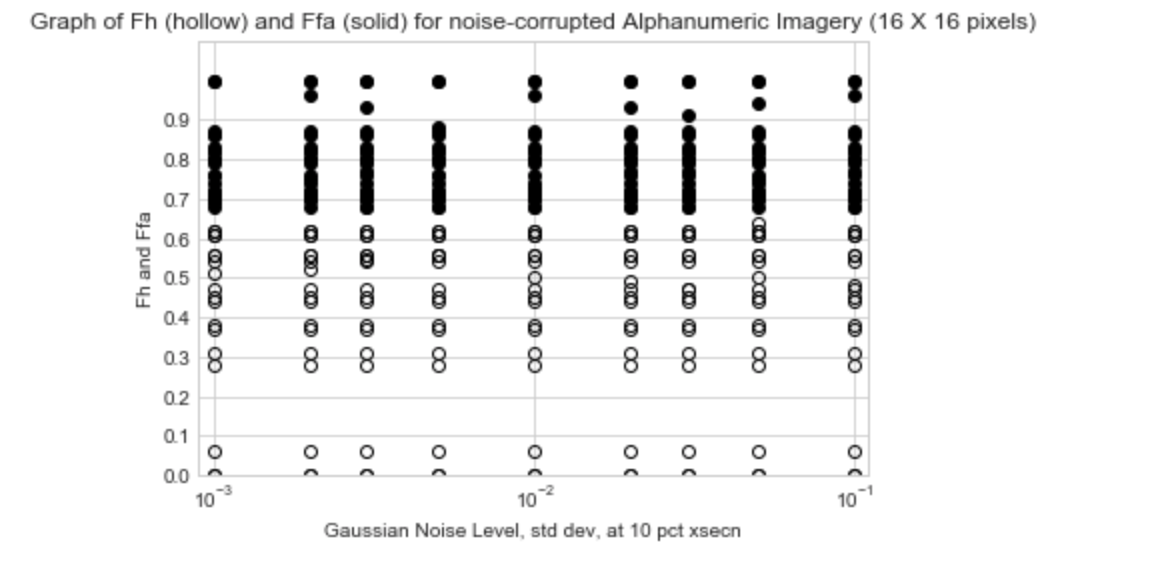
To test the accuracy of the perceptron, we introduced gaussian noise into the images. The model was correctly able to recognize the alphanumeric characters and produce denoised images.



**Figure 13:** Predictions of SLP on noisy input with standard deviation 0.01

**D. PERFORMANCE METRICS ( Fh, Ffa)**

Fh  and Ffa are two metrics calculated to check the accuracy of the perceptron model.

****

**Figure 14:** Performance metrics for noise corrupted images