ECE655 Project 04

Author: Stewart Schuler

Due: 10/26/2025

Contents

1 Part A

The pre-augmented dataset used for this project consists of 20 hand drawn 20x20 pixel Q and M, 10 of each letter. They were generated using the paint.NET tool and saved as PNG files. The letters are shown in Figures 1 and 2.

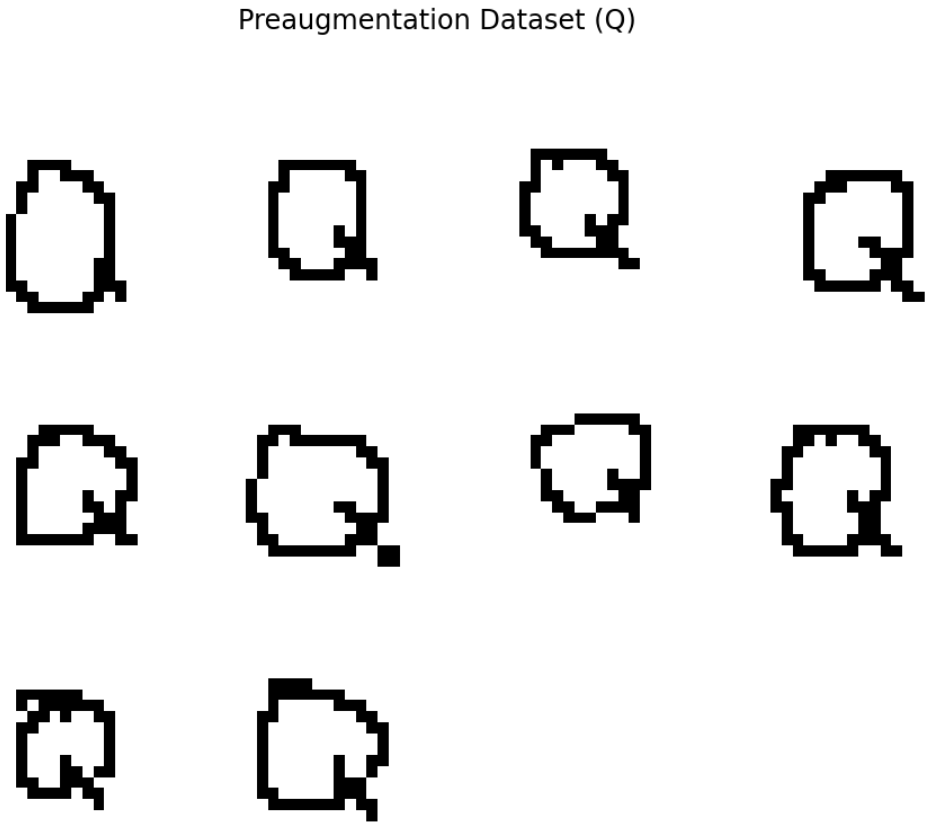

Figure 1: Q Dataset

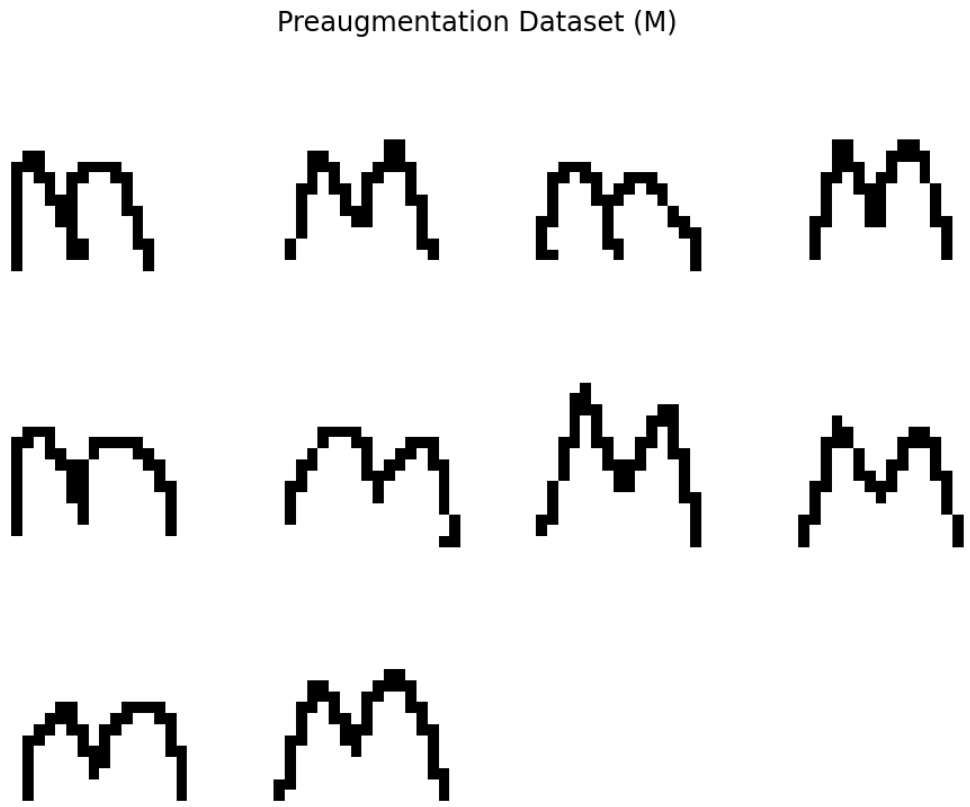

Figure 2: M Dataset

2 Part B/C

Next, before training the model since 20 data samples is insufficient the dataset was upsampled by a factor of 10 to now contain 200 images. Originally these 200 images were copies of the 20 hand drawn images from Part A. To augment the dataset random rotations and scaling were applied to these images. To create 200 unique images, albeit they are linear transforms applied to the original images. The dataset augmentation was acheived by the following code, and the 200 post augmented images can be see in Figures 3 and 4.

```
def upsample_dataset( images, N=100 ):
    idx = np.array([ random.randint(0, images.shape[0]-1) for
    ii in range(int(N)) ], dtype=int);
    rtn_images = np.array([ images[ii] for ii in idx ]);
    return rtn_images

upsampled_Q = upsample_dataset( images[0:10:], N/2 )
upsampled_M = upsample_dataset( images[10::],  N/2 )
images = np.concat((upsampled_Q, upsampled_M), axis=0);

augment_composer = Compose([RandomAffine(degrees=(-15,15),
   translate=(0.1,0.1))])
to_pil = ToPILImage();
augmented_images = np.array( [ augment_composer(to_pil(images
   [ii][0])) for ii in range(images.shape[0])] )
```

Listing 1: part_b.py

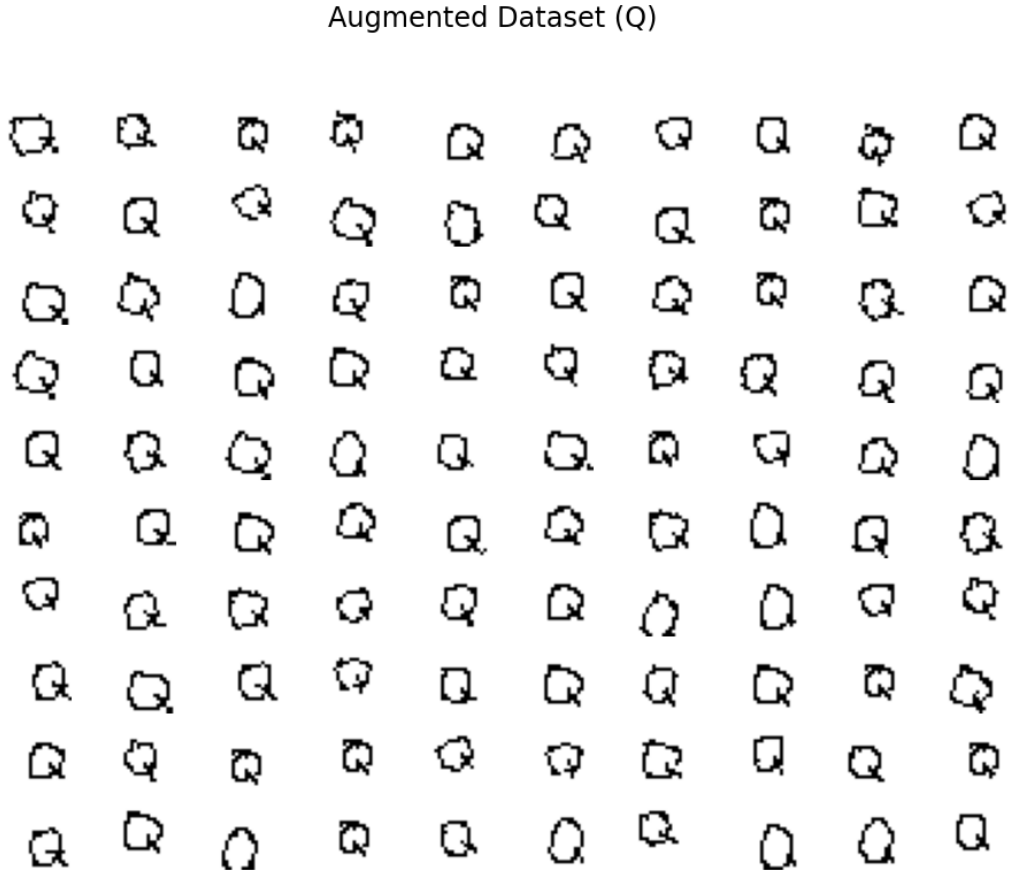

Figure 3: Q Augmented Dataset

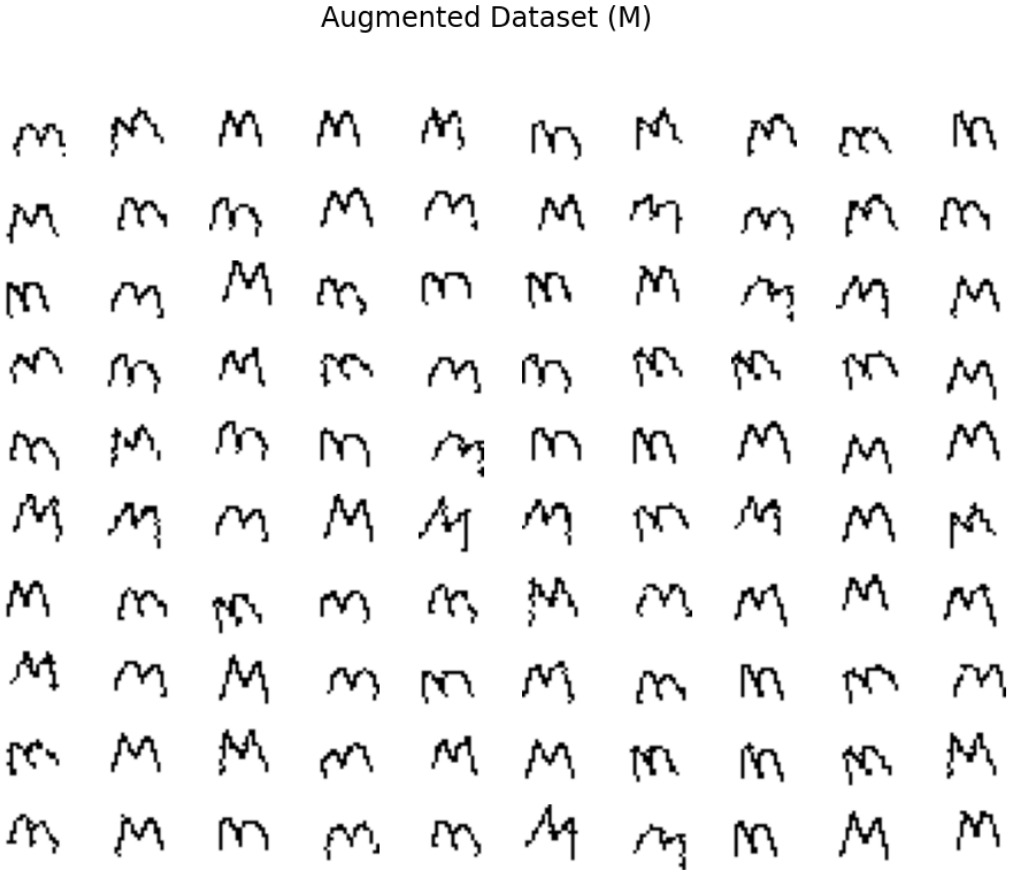

Figure 4: M Augmented Dataset

3 Part D

Next, an 80:20 split was applied to the dataset and binary classifier was trained with two hidden layers of 16 and 4 neurons each. The model can be defined by the following code.

```
model = nn.Sequential()
model.add_module('flatten', nn.Flatten())
model.add_module('linear0', nn.Linear(400, 16, bias=True))
model.add_module('ReLU0', nn.ReLU())
model.add_module('linear1', nn.Linear(16, 4, bias=True))
model.add_module('ReLU1', nn.ReLU())
model.add_module('linear2', nn.Linear(4, 1, bias=True))
model.add_module('sigmoid', nn.Sigmoid())
```

Listing 2: part_d1.py

The model was trained for 50 epochs using the Adam optimize, BCELoss, and a learning rate of 0.001. Figure 5 shows the loss curve.

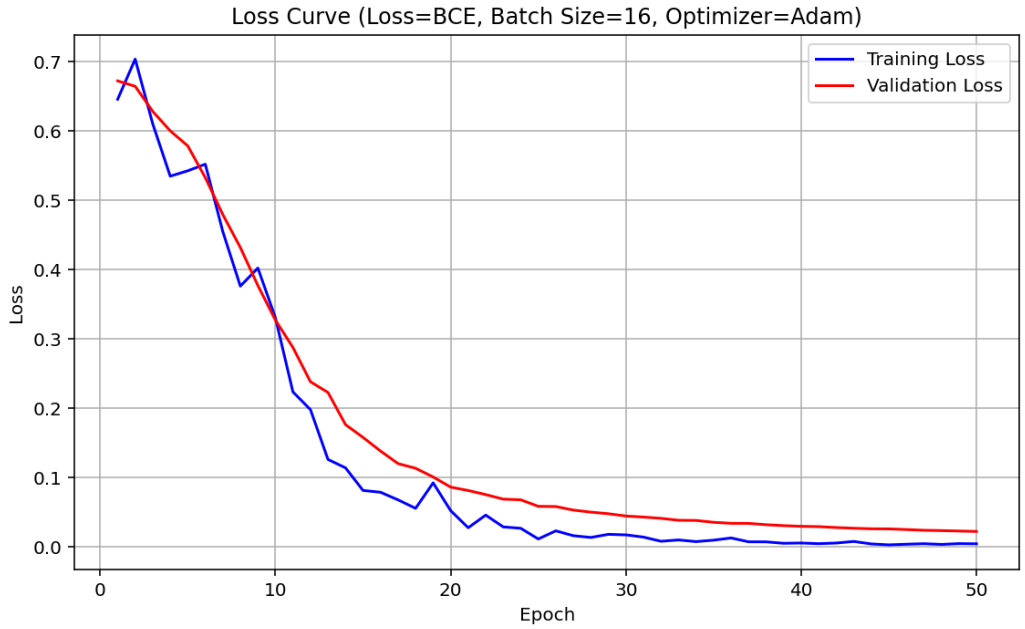

Figure 5: Loss Curve with (**Adam**, **BCE**)

The for this dataset the ROC curve (Figure 6) was found to be ideal and the TPR, FPR was computed to be 1.0 and 0.0 respectively for a threshold value of 0.5. The results of apply the final model to the validation set can be found in Figure 7 where the title of each subplot is the model's output.

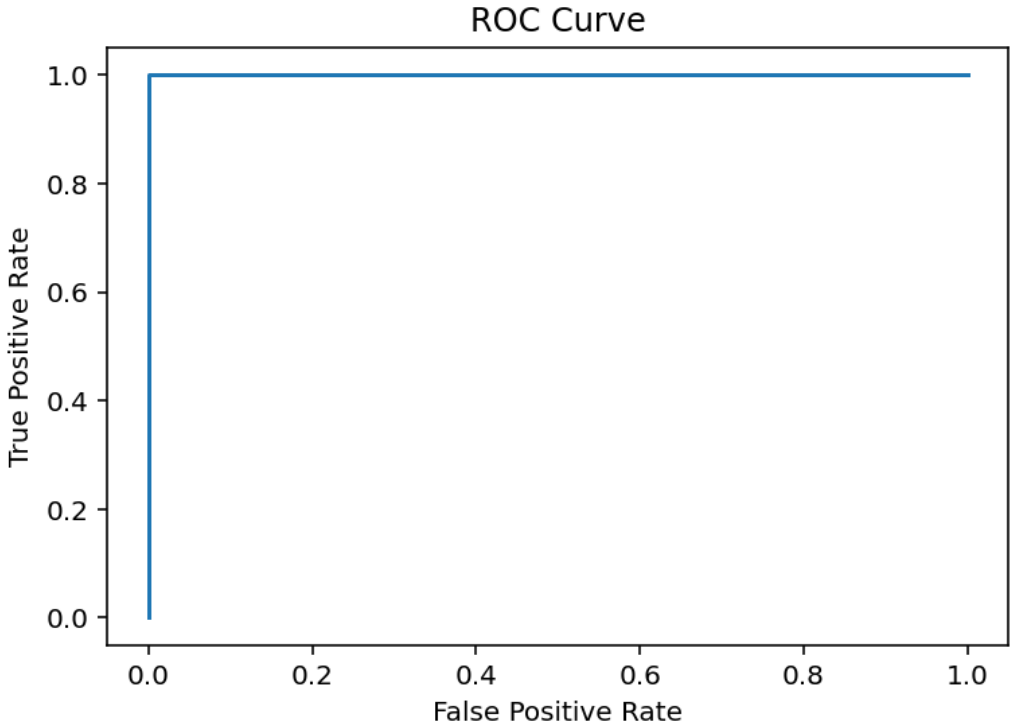

Figure 6: ROC Curve

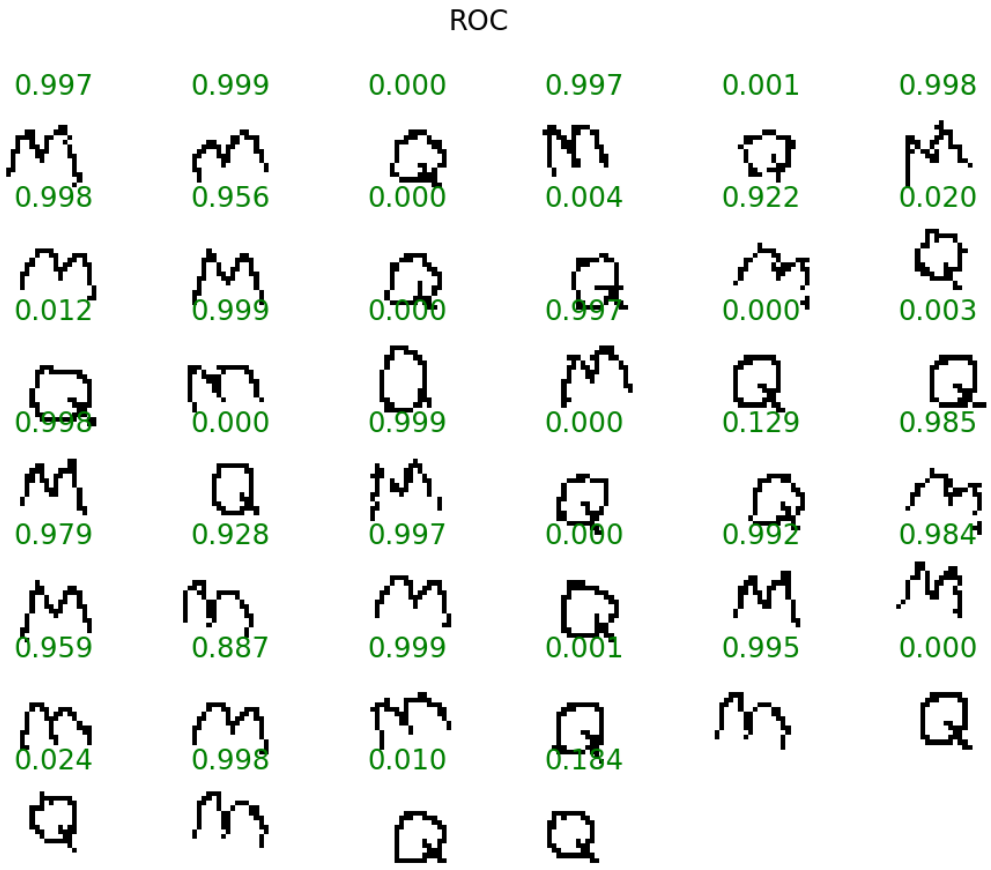

Figure 7: Validation Dataset Results

Given that the experiment resulted perfect validation results I tested on an additional dataset to verify if it was actually learning correctly. I generated a new dataset of 5 Q's and 5 M's, these are not augmented versions of the original 20 like the training set was, they are truly unique. I applied the learned model to these and got the results shown in Figure 8. It can be seen that with a threshold of 0.5 the model correctly classifies all 10 new images, with reasonably high confidence values.

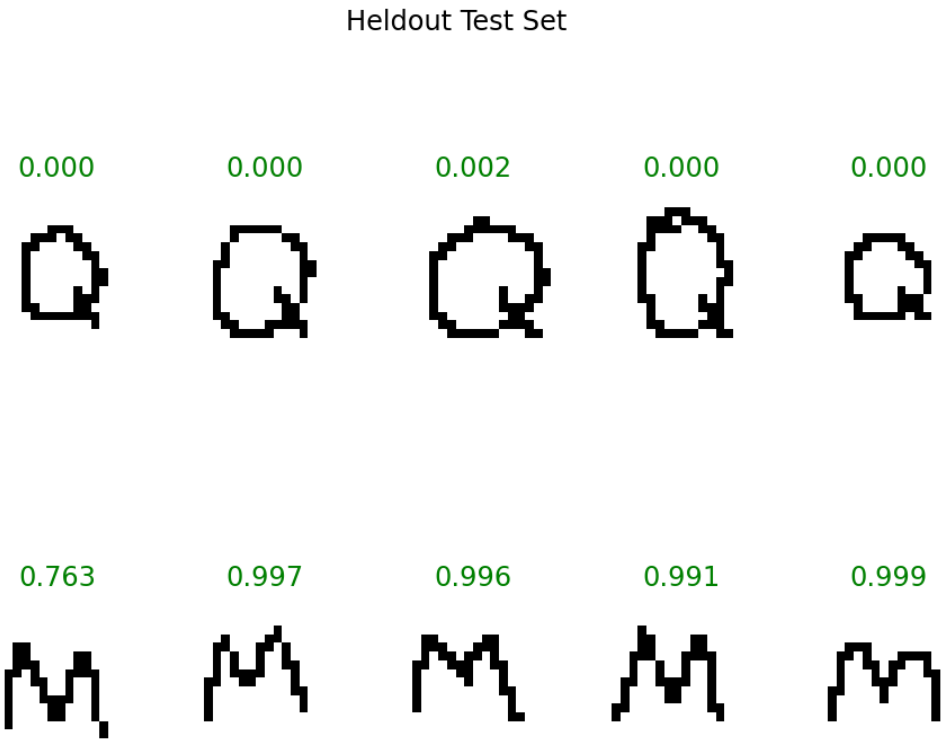

Figure 8: Holdout Dataset Results