Homework 3
Chris Lee

Problem 1

System Specification

For the parameters of my perceptron, I used 15 epochs with an eta of 0.001. I chose this because I wanted to have the data visualize more nicely even though it would converge faster with similar results using an eta of 0.005 and fewer epochs. It also created a lower error fraction than the other combinations I tried. My weights were randomized using np.random.rand (and divided by 2 to have all initial weights between [0, 0.5)), and I seeded it at 7 for consistent weights.

Results

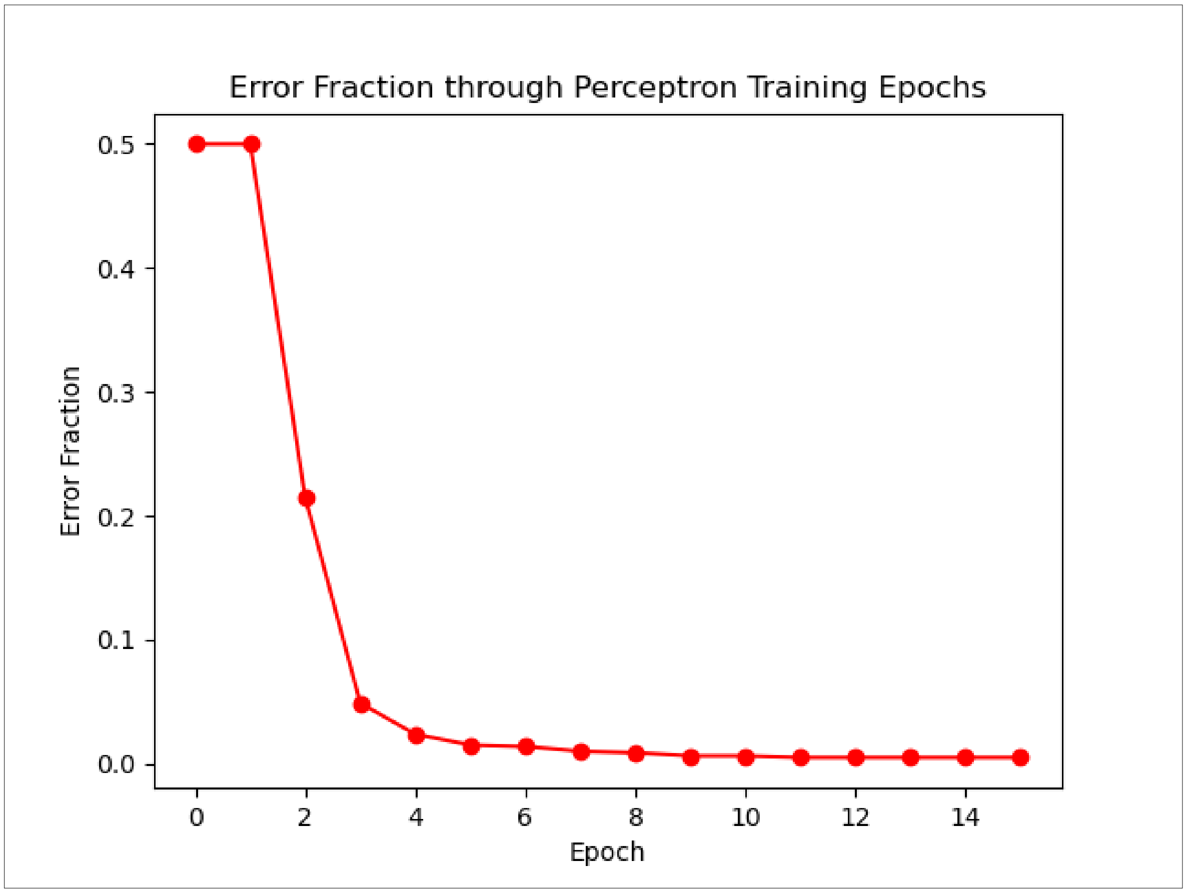


Figure 1.1: Error fraction as a function of passes through the training set. Note that it mostly converges after 5 passes.

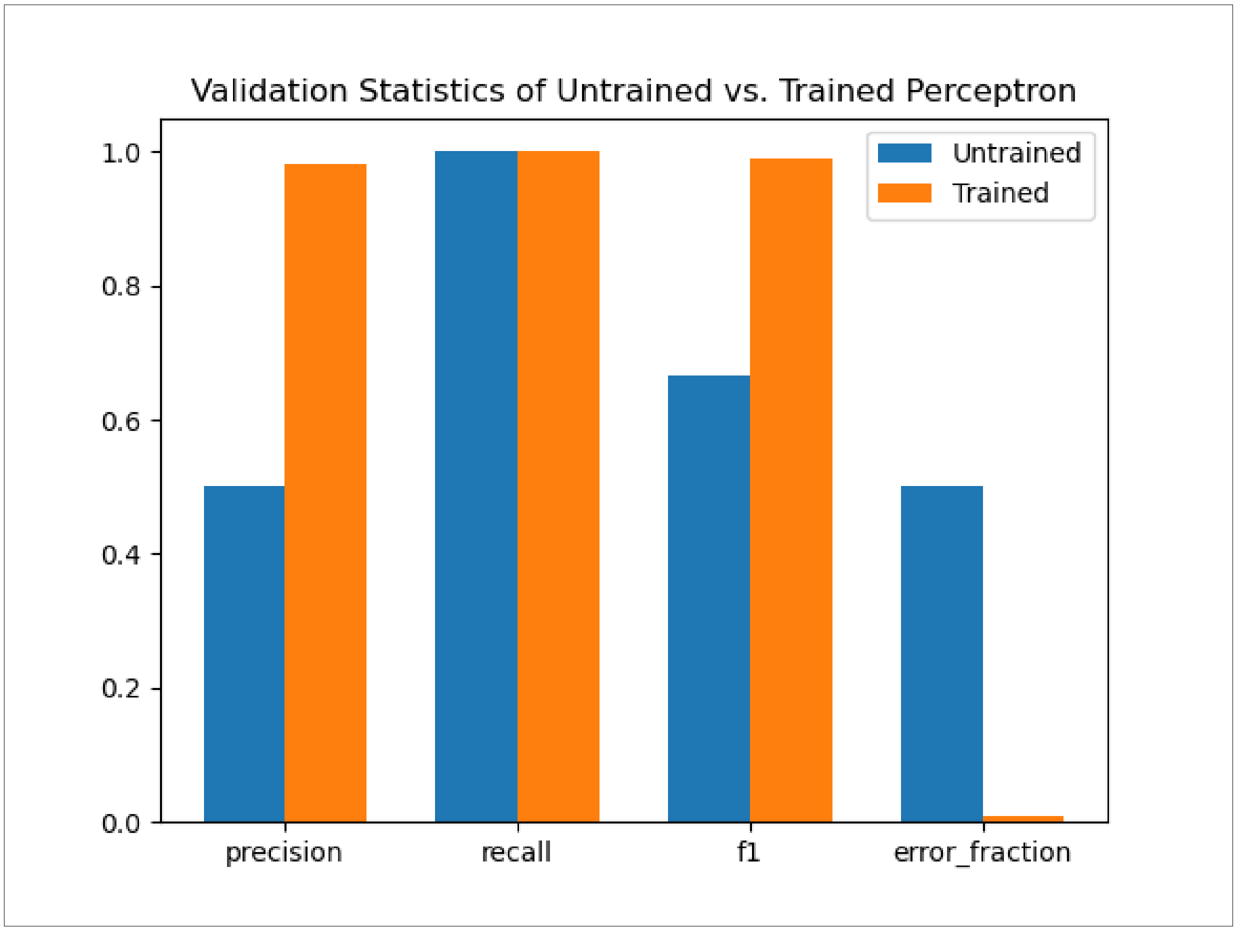


Figure 1.2: The precision, recall, F1 score, and error fractions before and after training. The precision is the biggest improvement after training.

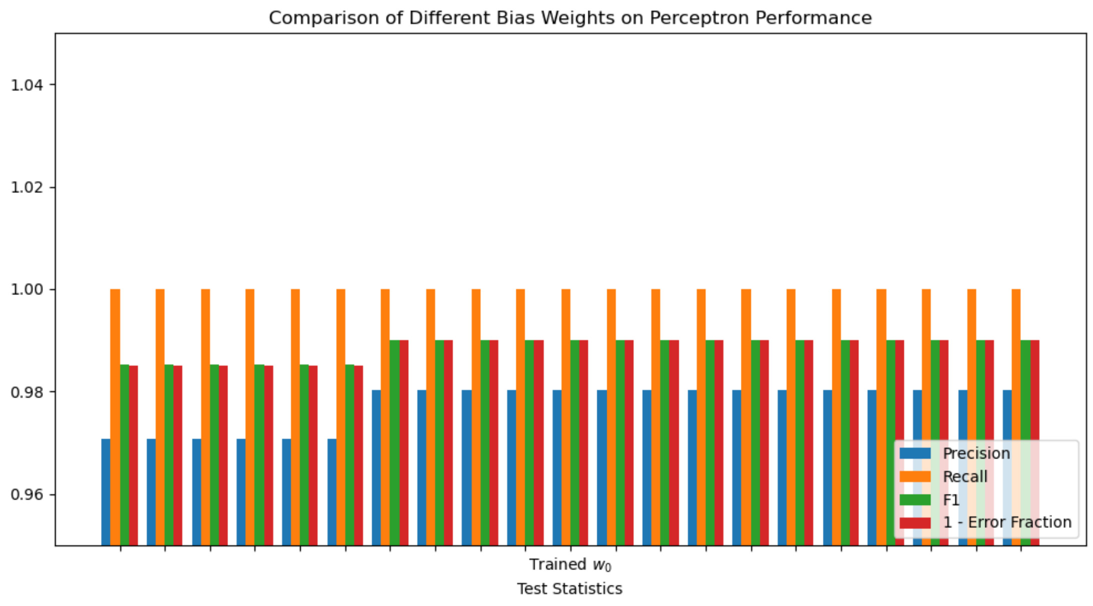


Figure 1.3: A measurement of test statistics after using a range of different bias weights, with the original bias weight statistics in the middle as reference.

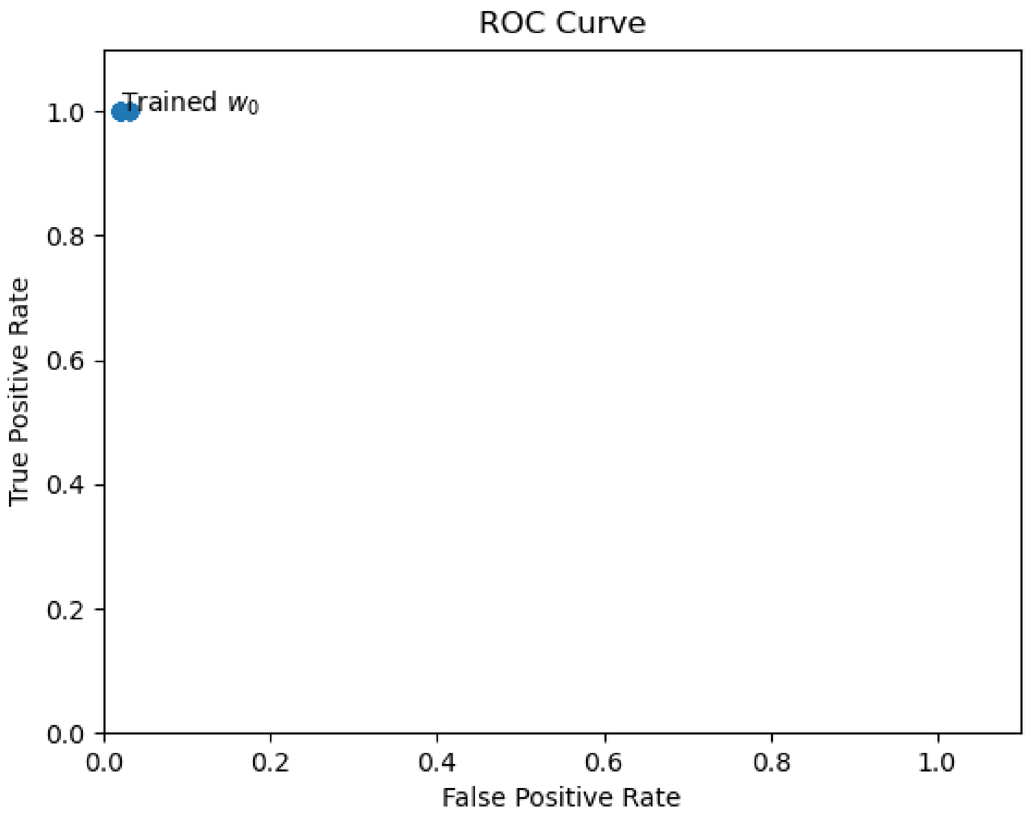


Figure 1.4: The ROC curve for the range of bias weights. Their performances are extremely similar.

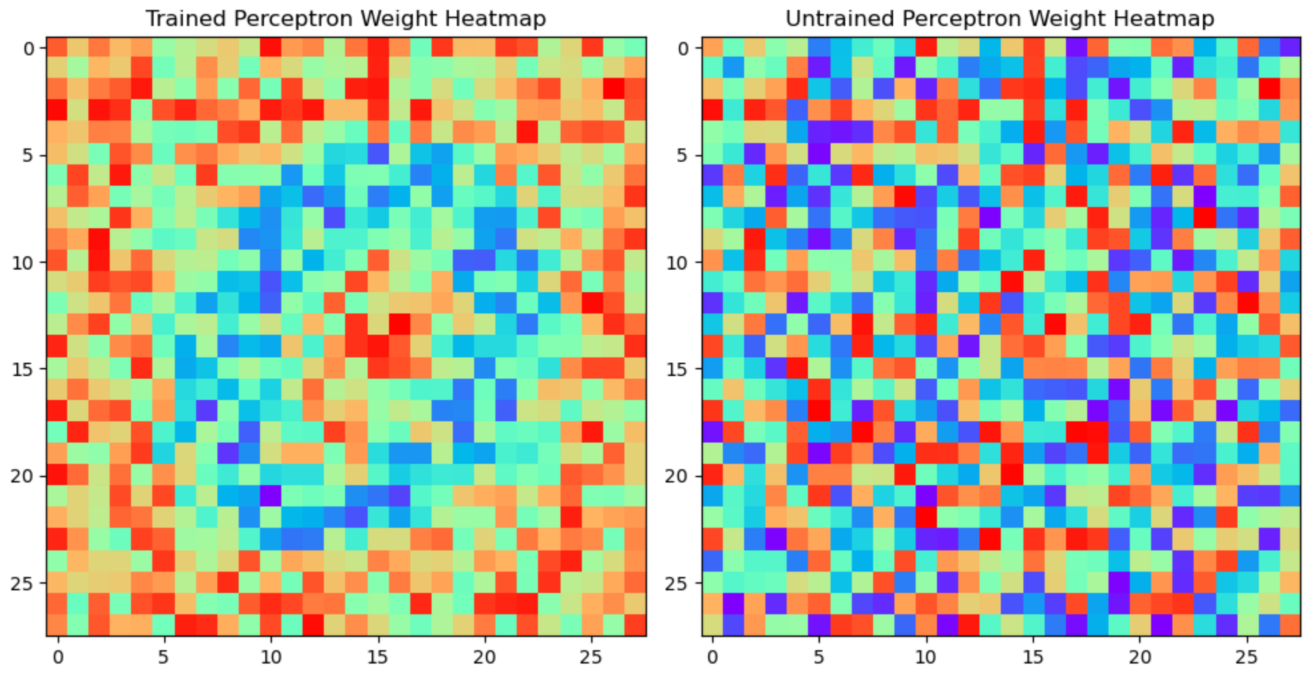
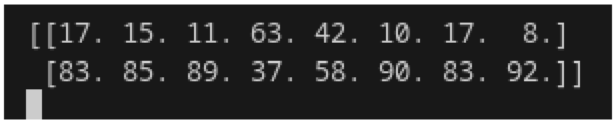


Figure 1.5: A comparison of heatmaps of the weights for an untrained and trained perceptron. Deep blue indicates negative weights for the trained heatmap but zero weight on the untrained heatmap.



```
[[17. 15. 11. 63. 42. 10. 17.  8.]
 [83. 85. 89. 37. 58. 90. 83. 92.]]
```

Figure 1.6: A table of values. The rows correspond to how many were classified as 0 or 1 out of a sample of 100, and the columns correspond to the 8 digits from 2 to 9.

Based on Figure 1.3., it appears that variation from the bias weight only negatively impacts performance when the bias weight is multiplied by a factor smaller than around 0.9. Otherwise, it does not impact the performance of the perceptron.

Analysis of Results

The untrained test statistics shown in Figure 1.2 surprised me. However, it makes sense because all of the weights were random numbers in the interval [0, 0.5). Since there's no input that causes inhibition, any picture that wasn't completely black would almost definitely register as a 1 by the untrained perceptron. This means that there would never be a false negative, so the recall would be perfect, and all of the zeros would be false positives, so the precision would be ½.

Secondly, the results shown in Figure 1.6 are only a little surprising. Any digit that has a vertical line or a strong "presence" in its middle would strongly excite the perceptron, which explains the results for 2, 3, 4, 7, 8 and 9. However, any digit with a round shape would also strongly inhibit the perceptron. This would explain why a lot of 5s and 6s were interpreted as 0s; the only part of their shape in the middle is a horizontal line. However, most of the digits share some characteristics of the shapes of both 0 and 1, so there exists some ambiguity.

Problem 2

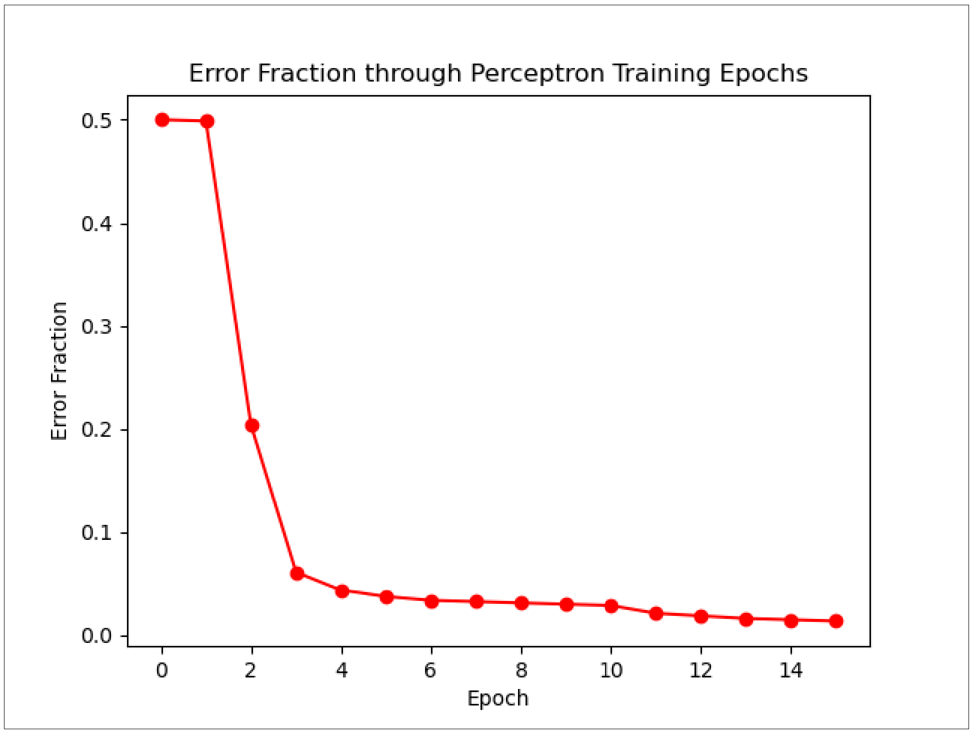


Figure 2.1: Error fraction as a function of passes through the training set of 0s and 9s. Note that the error fraction converges at a higher number than for Figure 1.1

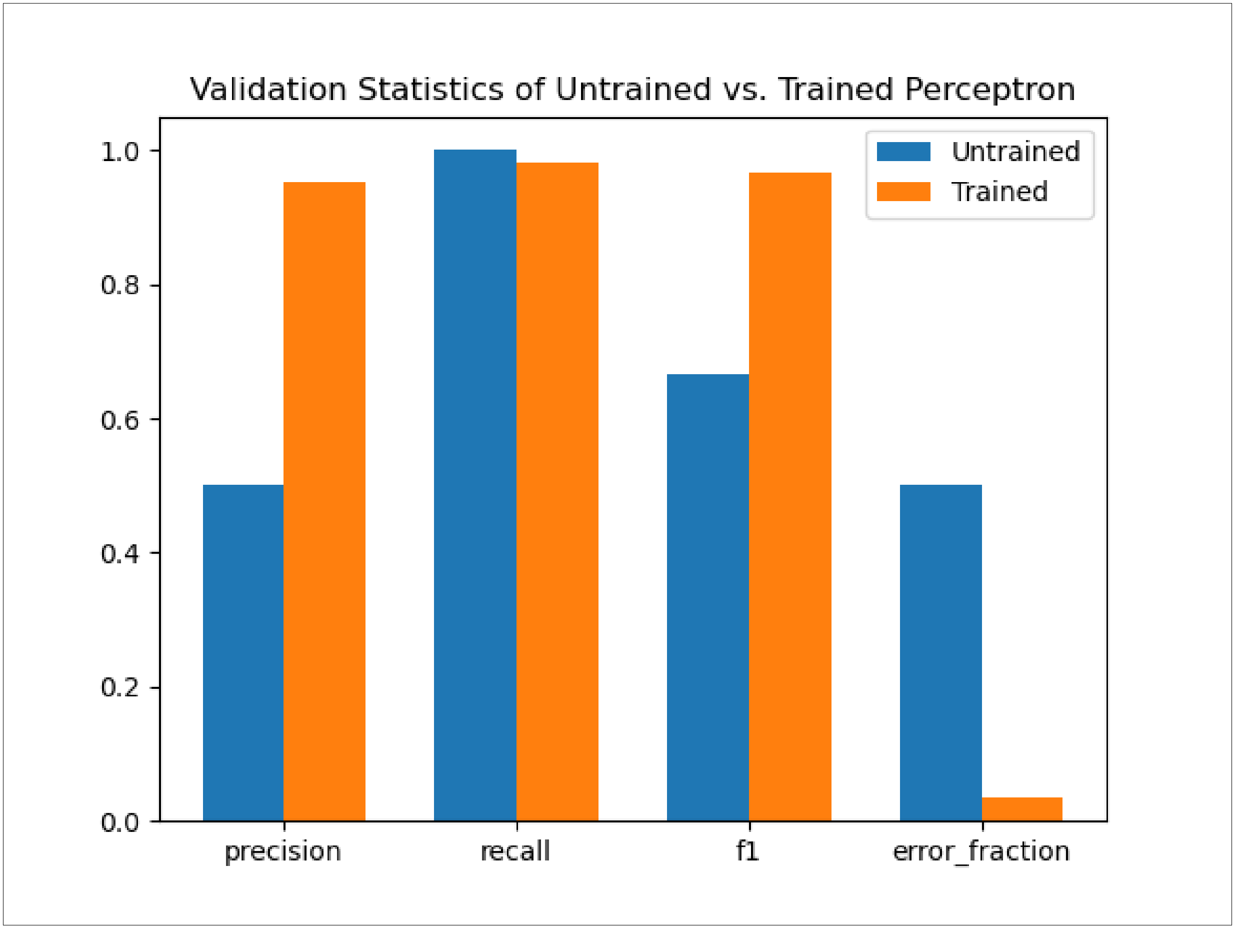


Figure 2.2: Statistics of a pass through the test set of 0s and 9s. Note that the performance after training is slightly worse than in Figure 1.2.

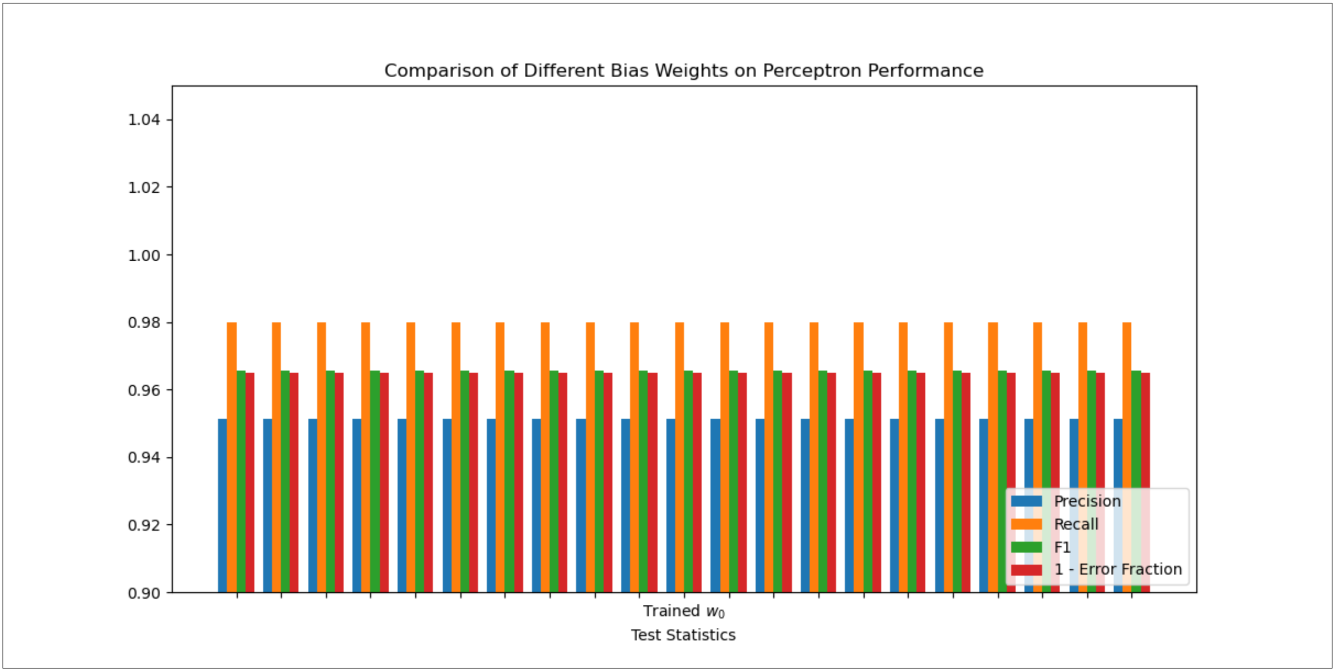


Figure 2.3: A comparison of different bias weights on the performance of the perceptron. Changing the bias weight seems to not affect performance for distinguishing 0s and 9s.

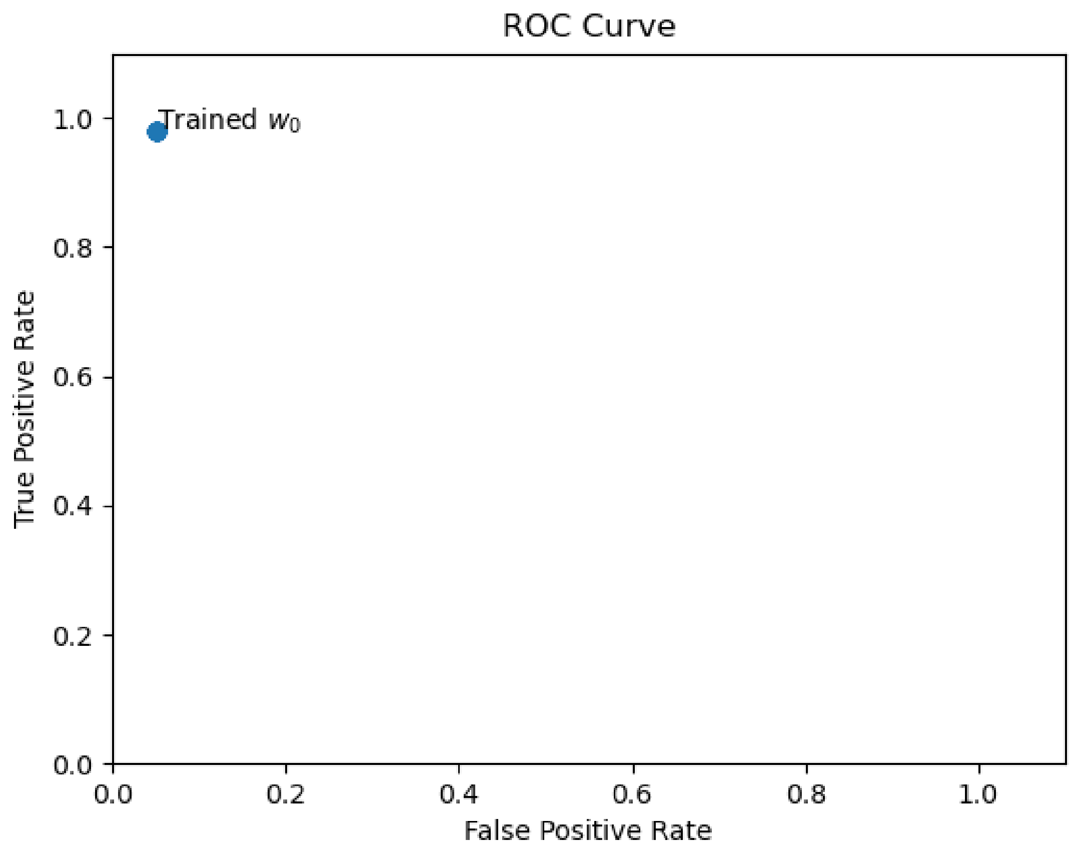


Figure 2.4: As the performances for all 21 bias weights was the same, there is only one distinct point in the ROC curve.

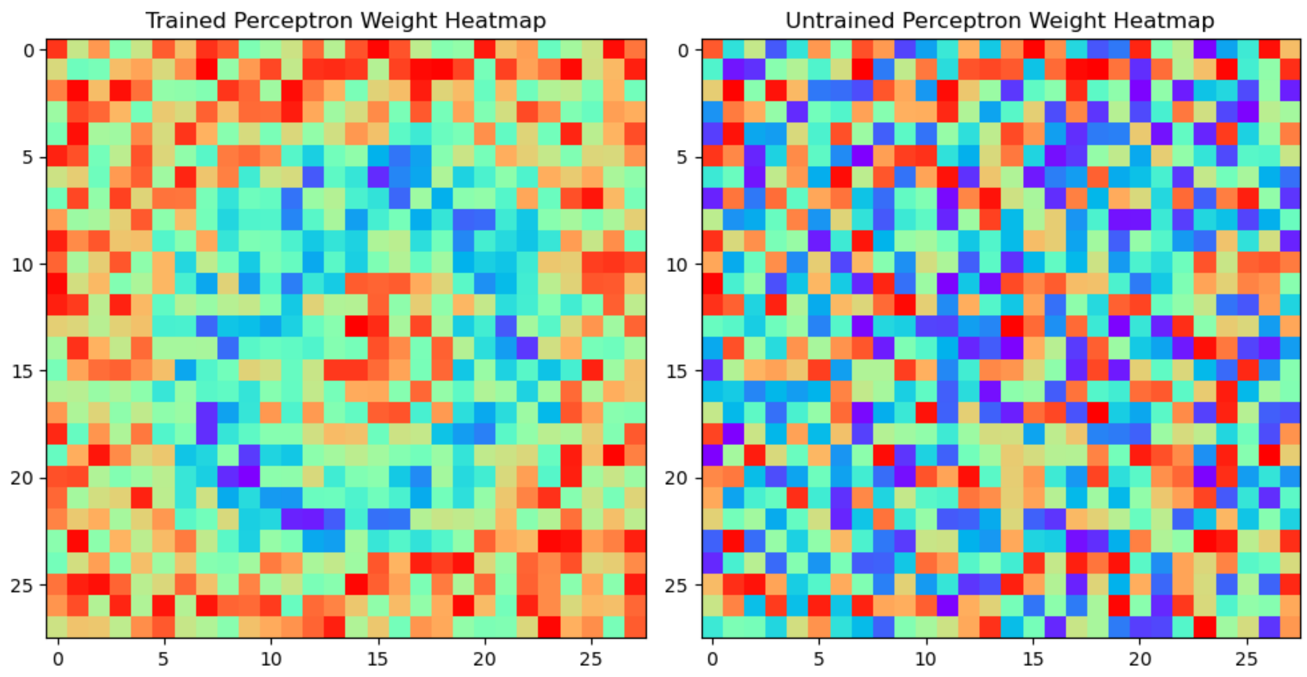
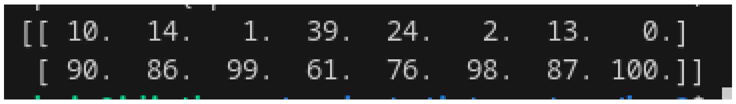


Figure 2.5: The heatmaps before and after training a perceptron on classifying 0s and 9s.



```
[[ 10.  14.   1.  39.  24.   2.  13.   0.]
 [ 90.  86.  99.  61.  76.  98.  87. 100.]]
```

Figure 2.6: The results of testing the challenge sets. Note that the columns represent the digits 1-8, and the rows represent whether it was classified as 0 or 1, respectively.

Analysis of Results

The perceptron had slightly inferior performance in classifying 0s and 9s compared to 0s and 1s. As shown in Figure 1.2 and Figure 2.2, both the precision and recall for the classification of 0s and 9s was worse. The perceptron probably had inferior performance in the test set because 0 has a more similar shape to 9 than it does to 1, so it must be harder to separate just using a linear boundary. For the challenge set, it classified most of the digits as 9s, with the notable exception of the digits 4 and 5 having more than 20 classified as 0s. This is largely similar to the results in Figure 1.6, where it classified most digits as 1s, except that in Figure 2.6, the digit 4 was classified as a 0 somewhat often, and the digit 6 is almost always classified as a 9. This probably happened because the negative weights, as shown in Figure 2.5, corresponded to a lot of pixels in some of the 4s, and the strongly positive weights correspond to a lot of pixels in the 6s.