Train/Validation   I have created a 85:15 train/validation split, so that there are 581 points in the training set and 102 points in the validation set.

Network Architecture

- fully connected layer with 512 neurons and ReLU activation

- fully connected layer with 512 neurons and ReLU activation
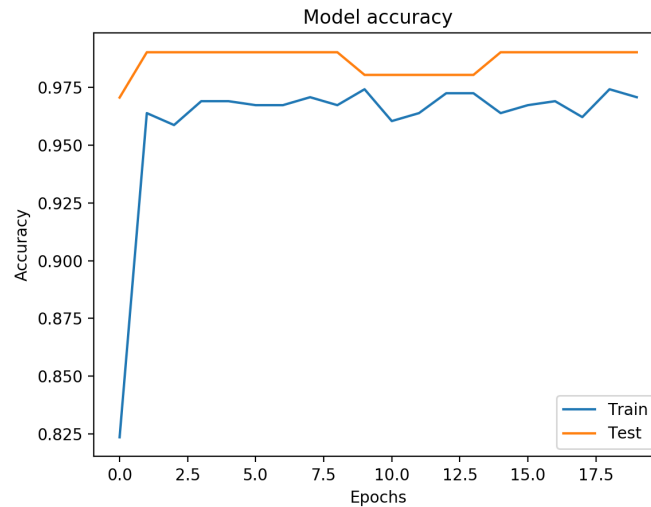
- output layer with 2 neurons and softmax activation

Figure 1: Train and test accuracy over number of epochs.

Data	Label	Prediction
6 1 3 1 2 1 3 1 1	0	1

Table 1: Misclassified example.

Performance   The model is trained for 20 epochs. The train accuracy can reach 99.0% while the test accuracy can reach 97.1% (Figure 1). Training accuracy tends to be lower than test accuracy during experiments. I think that this may be due to lack of training data, or lack of features. Bias values tend to be around 0, but the weight values are not all 0 (Figure 2). Bias and weights for layer 2 have higher values.

Table 1 shows a misclassified point. The points in the Breast Cancer dataset are vectors describing patient's breast tumor, like clump thickness and cell
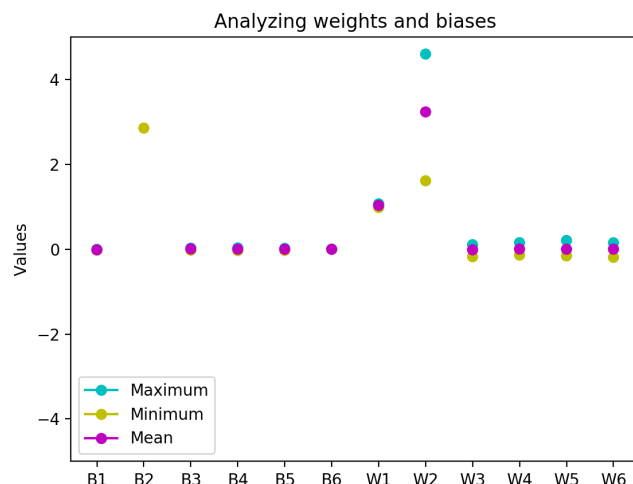
Figure 2: Final weights and biases of trained model. Each weight and bias has 3 points associated with it: maximum, minimum, and mean. These values indicate the largest, smallest, and average of values in the matrix/vector.

shape. These attributes have values ranging from 1 to 10. The two examples are supposed to be labeled as malignant, but instead labeled as benign. I can see how they would be misclassfied as benign since the attributes aren't too large in value.

Explanation I normalized the vectors to fall between 0 and 1 using MinMaxScaler from sklearn. The dataset does not contain images, so I have not used a CNN for this dataset. Instead, I use a feed-forward network with two fully-connected layers of 512 neurons. I have used a ReLU activation function. Batch normalization and reducing the batch size to 10 also helps increase accuracy.

I have not included dropout from this network because I think that the network does not suffer from overfitting. In fact, I find that the network underfits because of lack in training data. The last layer also uses softmax function to output probability distribution over the classes. The loss function is binary cross-entropy because there are only two classes. This network only took less than a minute to train on a CPU.