Partners:
Vishrut Vanga - Write up and Binary Classification
Vignesh Venkat - Write up and Multiclass Classification and creating the dataset of images to use to demonstrate learning

PART 1:

How are we defining our input space?

Our input space is being defined as a set of images that contain the four lines that overlap each other.

How are we defining our output?

Our output is whether or not the images are dangerous or safe. Dangerous is defined by greater than 0.5 from the sigmoid function and anything else is deemed safe.

Creating the Data Set

To start off, we had an array of the four possible colors that could be included in the drawing. At random, we assigned colors to rows and columns of random indexes. Then we checked if the red wire was placed before the yellow wire to determine if it was dangerous or not. If the image generated was safe, we put the image in the folder of safe, if not we transferred it to the is_dangerous folder. This is how we got the parameters for our dataset

After we got our parameters, we created a train test split method in order to get test and train data for our inputs and outputs.

Creating the Model

The train test was split at 80 20 and from there, we created the model. For binary classification, we used the logistic loss function:

$$\sum_{i=1}^{N}\left[y_{actual}log(y_{\ pred}) + (1 - y_{\ actual})log(1 - y_{\ pred})\right]$$

and we used a logistic regression model, which contained the weights and biases that were updated every batch point. We used the sigmoid function in order to calculate the loss of every batch point, which we set to 32 as it is the industry standard. We are using the gradient descent algorithm to optimize our function, however we do use a rough approximation of the derivative:

$$x_{n+1} = x_n - \alpha\left(\frac{dy}{dx}_n\right)$$

. We used our sigmoid activation function

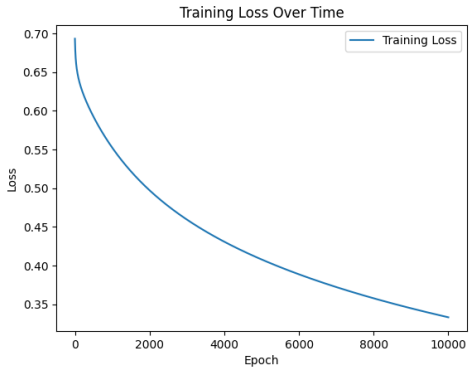
$$\frac{1}{1+e^{-x}}$$

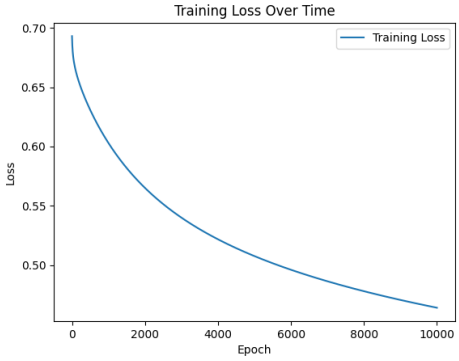
in order to decrease the weights of other input values.

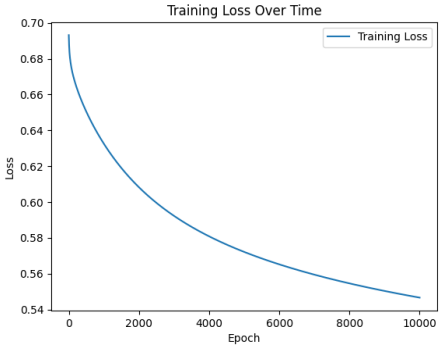
We did not necessarily see overfitting on this stage, so we did not try anything other than optimizing the learning rate and the number of epochs. However, we did notice a bottleneck in performance that we would like to improve on.

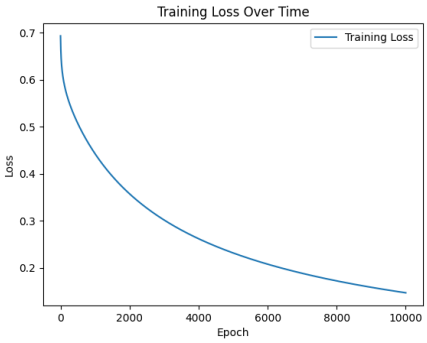
Performance:
500 trials - 70.92% accuracy
1000 trials - 75% accuracy
2500 trials - 72% accuracy
5000 trials - 74% accuracy

Plots:
(1000, 2.5K, 5K, 100) All trained on 10000 Epochs









As you can see based on our loss functions, even after 10000 epochs there seems to be a flattening of the curve which is to be expected. When we tested our data on the test dataset we had prepared, we saw that

Assessment of Trained Model:

Done on testing data

```
Dangerous: predicted value:0.506140204836002,Real value:1
Dangerous: predicted value:0.5929212339906178,Real value:0
Safe: predicted value:0.29247617299353096,Real value:1
Dangerous: predicted value:0.6089241302534906,Real value:1
Dangerous: predicted value:0.6748006411250137,Real value:1
Safe: predicted value:0.2888412065199859,Real value:1
Dangerous: predicted value:0.5397587930056327,Real value:1
Dangerous: predicted value:0.5289115070403609,Real value:0
Dangerous: predicted value:0.7593545333078303,Real value:1
Safe: predicted value:0.19392928872483528,Real value:1
Safe: predicted value:0.4530268202440554,Real value:1
Dangerous: predicted value:0.9575360601458482,Real value:1
Dangerous: predicted value:0.5088121839060363,Real value:1
Dangerous: predicted value:0.988709448254296,Real value:1
Safe: predicted value:0.1792212729374969,Real value:0
Safe: predicted value:0.2807711956869362,Real value:0
Safe: predicted value:0.3400312004430801,Real value:1
Safe: predicted value:0.38397818134609746,Real value:0
Dangerous: predicted value:0.7726098807463011,Real value:0
Dangerous: predicted value:0.7305805888641315,Real value:0
Dangerous: predicted value:0.6253423427826252,Real value:1
Safe: predicted value:0.41574831438840326,Real value:1
Dangerous: predicted value:0.5083369081194864,Real value:1
Safe: predicted value:0.24835402695159808,Real value:0
Safe: predicted value:0.4443667572591412,Real value:0
Safe: predicted value:0.20127444572086756,Real value:0
Dangerous: predicted value:0.5828261909127084,Real value:1
Dangerous: predicted value:0.5551822674370007,Real value:1
Safe: predicted value:0.35037056644851333,Real value:1
Safe: predicted value:0.4446685505580734,Real value:0
Safe: predicted value:0.17613983585241166,Real value:1
Dangerous: predicted value:0.6465824649167881,Real value:1
Dangerous: predicted value:0.6711875154823879,Real value:1
Dangerous: predicted value:0.5996806011296321,Real value:1
Dangerous: predicted value:0.6074129448520366,Real value:0
Safe: predicted value:0.22918264038320577,Real value:0
Safe: predicted value:0.2633942812808622,Real value:0
Dangerous: predicted value:0.6618831309384764,Real value:0
Safe: predicted value:0.4488608689474837,Real value:0
Dangerous: predicted value:0.977660330982761,Real value:1
Dangerous: predicted value:0.5229586188640654,Real value:0
Safe: predicted value:0.3345104404367585,Real value:0
Dangerous: predicted value:0.6535881897693785,Real value:1
Safe: predicted value:0.4074413895028733,Real value:1
Dangerous: predicted value:0.5355249695869464,Real value:1
Dangerous: predicted value:0.5300591556948085,Real value:1
Dangerous: predicted value:0.6398858365986868,Real value:1
Safe: predicted value:0.32560491100774575,Real value:0
Dangerous: predicted value:0.6822503672960227,Real value:0
Safe: predicted value:0.4365225481867346,Real value:0
Safe: predicted value:0.35158947053243955,Real value:0
Safe: predicted value:0.39985752496843513,Real value:0
Safe: predicted value:0.2950918566526424,Real value:1
Dangerous: predicted value:0.5587681570472786,Real value:0
Safe: predicted value:0.13511388463412738,Real value:0
Safe: predicted value:0.37715745903685566,Real value:0
Safe: predicted value:0.2138352040581957,Real value:0
Safe: predicted value:0.16440060414334462,Real value:0
Safe: predicted value:0.3107599564727859,Real value:1
```

PART 2:

<u>How are we defining our input space?</u>

Our input space is being defined as a set of images that contain the four lines that overlap each other.

How are we defining our output?

Our output is which color wire to cut. We aim to find the wire that was laid down third.

Creating the Data Set

To start off, we had an array of the four possible colors that could be included in the drawing. At random, we assigned colors to rows and columns of random indexes. Then we checked if the red wire was placed before the yellow wire to determine if it was dangerous or not. If the image generated was safe, we put the image in the folder of safe, if not we transferred it to the is_dangerous folder. This is how we got the parameters for our dataset

After we got our parameters, we created a train test split method in order to get test and train data for our inputs and outputs.

Creating the Model

The train test was split at 80 20 and from there, we created the model. For binary classification, we used the logistic loss function:

$$\frac{1}{len(y_{true})} \sum_{i=1}^{len(y_{true})} y_{true} * log(y_{pred})$$

and we used a logistic regression model, which contained the weights and biases that were updated every batch point. We used the sigmoid function in order to calculate the loss of every batch point, which we set to 32 as it is the industry standard. We are using the gradient descent algorithm to optimize our function, however we do use a rough approximation of the derivative:

$$x_{n+1} = x_n - \alpha(\frac{dy}{dx}_n)$$

. We used our softmax activation function

$$\frac{e^{w\ of\ class\ dot\ product\ x}}{\sum_k e^{w\ of\ class\ dot\ product\ x}}$$

in order to decrease the weights of other input values. We did notice overfitting so we also did include L1 Regularization. However, we did not notice it had a massive effect on the overfitting problem. We did also try to mess around with the learning rate and epochs:

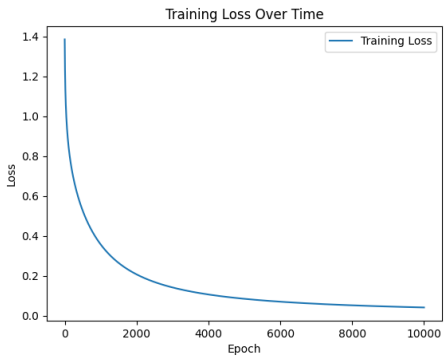
$$Loss(W) = Loss(W) + beta * \sum_{k=1}^{N} \left| w_k \right|$$

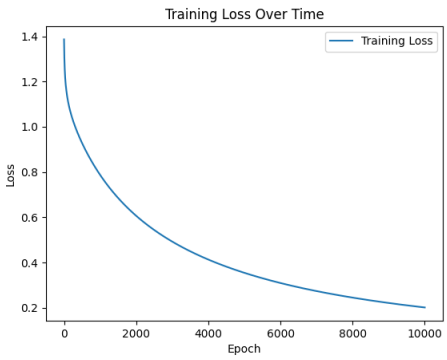
Performance:

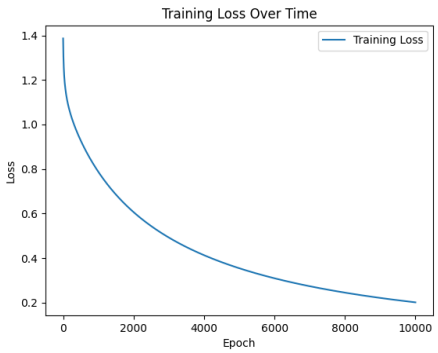
500 trials - 54.2% accuracy
1000 trials - 68.25% accuracy
2500 trials - 74.72% accuracy
5000 trials - 76.9% accuracy

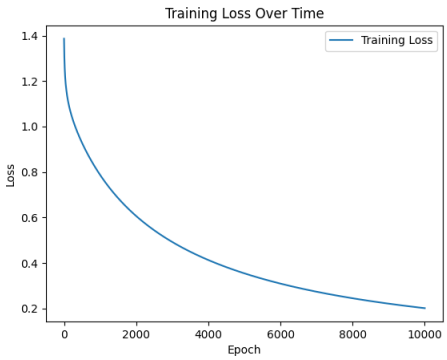
Plots:

(500,1000,2.5K,5K)









Assessment of Trained Model:

Done on Testing Data:

```
Sample 133: Predicted Class: 2, True Class: 0
Sample 134: Predicted Class: 0, True Class: 0
Sample 135: Predicted Class: 2, True Class: 1
Sample 136: Predicted Class: 0, True Class: 3
Sample 137: Predicted Class: 3, True Class: 1
Sample 138: Predicted Class: 0, True Class: 1
Sample 139: Predicted Class: 2, True Class: 0
Sample 140: Predicted Class: 1, True Class: 2
Sample 141: Predicted Class: 0, True Class: 2
Sample 142: Predicted Class: 3, True Class: 3
Sample 143: Predicted Class: 1, True Class: 2
Sample 144: Predicted Class: 0, True Class: 3
Sample 145: Predicted Class: 3, True Class: 2
Sample 146: Predicted Class: 0, True Class: 3
Sample 147: Predicted Class: 0, True Class: 2
Sample 148: Predicted Class: 0, True Class: 0
Sample 149: Predicted Class: 0, True Class: 3
Sample 150: Predicted Class: 3, True Class: 2
Sample 151: Predicted Class: 3, True Class: 3
Sample 152: Predicted Class: 0, True Class: 3
Sample 153: Predicted Class: 2, True Class: 0
Sample 154: Predicted Class: 2, True Class: 2
Sample 155: Predicted Class: 0, True Class: 3
Sample 156: Predicted Class: 1, True Class: 1
Sample 157: Predicted Class: 2, True Class: 3
Sample 158: Predicted Class: 0, True Class: 3
Sample 159: Predicted Class: 3, True Class: 0
Sample 160: Predicted Class: 2, True Class: 1
Sample 161: Predicted Class: 3, True Class: 3
Sample 162: Predicted Class: 0, True Class: 2
Sample 163: Predicted Class: 3, True Class: 2
Sample 164: Predicted Class: 0, True Class: 0
Sample 165: Predicted Class: 1, True Class: 0
Sample 166: Predicted Class: 1, True Class: 1
Sample 167: Predicted Class: 1, True Class: 1
Sample 168: Predicted Class: 1, True Class: 0
Sample 169: Predicted Class: 1, True Class: 1
Sample 170: Predicted Class: 0, True Class: 2
Sample 171: Predicted Class: 2, True Class: 0
Sample 172: Predicted Class: 0, True Class: 1
Sample 173: Predicted Class: 2, True Class: 2
Sample 174: Predicted Class: 1, True Class: 1
Sample 175: Predicted Class: 1, True Class: 3
Sample 176: Predicted Class: 3, True Class: 1
Sample 177: Predicted Class: 2, True Class: 2
Sample 178: Predicted Class: 0, True Class: 1
Sample 179: Predicted Class: 0, True Class: 3
Sample 180: Predicted Class: 1, True Class: 0
Sample 181: Predicted Class: 3, True Class: 3
Sample 182: Predicted Class: 3, True Class: 2
Sample 183: Predicted Class: 2, True Class: 0
Sample 184: Predicted Class: 0, True Class: 1
Sample 185: Predicted Class: 1, True Class: 0
Sample 186: Predicted Class: 1, True Class: 1
Sample 187: Predicted Class: 0, True Class: 1
Sample 188: Predicted Class: 1, True Class: 0
Sample 189: Predicted Class: 1, True Class: 1
Sample 190: Predicted Class: 0, True Class: 0
Sample 191: Predicted Class: 0, True Class: 3
Sample 192: Predicted Class: 2, True Class: 0
Sample 193: Predicted Class: 1, True Class: 3
Sample 194: Predicted Class: 3, True Class: 0
Sample 195: Predicted Class: 0, True Class: 0
Sample 196: Predicted Class: 1, True Class: 1
Sample 197: Predicted Class: 0, True Class: 2
Sample 198: Predicted Class: 0, True Class: 1
Sample 199: Predicted Class: 2, True Class: 0
```

BONUS:

We used Pytorch and we defined our own Logistic Regression model. I did not know much so I learned about how to define models using nn.Module. We also used a Cross Entropy loss and an SGD optimizer. We chose to go with SGD over Adam as SGD has the ability to scale with size. We use data loaders to input our data.