Comp 576 HW 1

Jason Uwaeze
juw@rice.edu

b) Activation Function

1) implement actFun( )

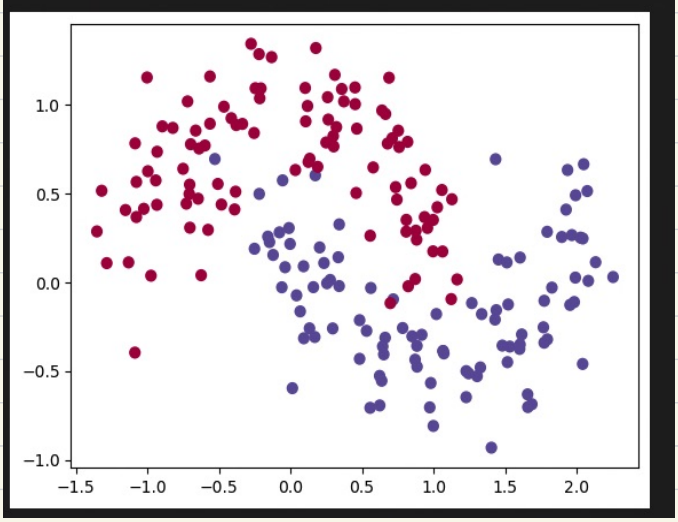
$$\tanh = \frac{e^{2z} - 1}{e^{2z} + 1}$$

$$\text{Sigmoid} = \frac{1}{1 + e^{-z}}$$

$$\text{relu} = \text{maximum}(0, z)$$

2. Derive tanh, Sigmoid, and relu

```
plt.scatter(X[:, 0], X[:, 1], s=40, c=y, cmap=plt.cm.Spectral)
✓ 0.1s
<matplotlib.collections.PathCollection at 0x15d5c6440>
```



a. Dataset

Derivative of ReLU

$f(x) = \max(0, x)$

$\boxed{f'(x) = \max(0, 1)}$

when $x < 0$ $f(x) = 0$

$\frac{dx}{dx} = 1$, so when $x >= 0$ $f'(x) = 1$

Derivative of Sigmoid Function

$\text{Sigmoid} = \frac{1}{1+e^{-z}} = (1+e^{-z})^{-1}$ $\quad \frac{d(1+e^{-z})^{-1}}{dz} = -1 \cdot (1+e^{-z})^{-2} \cdot -e^{-z}$

$= \frac{-e^{-z}}{-(1+e^{-z})^2} = \frac{e^{-z}}{(1+e^{-z})^2} = \frac{1}{1+e^{-z}} \frac{e^{-z}}{1+e^{-z}}$

$= \sigma(z) \cdot \frac{e^{-z}}{1+e^{-z}}$

$= \sigma(z) \cdot \frac{e^{-z} + (1-1)}{1+e^{-z}}$

$= \sigma(x) \cdot \frac{1+e^{-z} - 1}{1+e^{-z}}$

$= \sigma(z) \cdot \frac{1+e^{-z}}{1+e^{-z}} \cdot \frac{1}{1+e^{-z}}$

$\boxed{= \sigma(z) \cdot (1 - \sigma(z))}$

Derivative of Tanh

$$\tanh = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$\frac{d\tanh}{dz} = \frac{(e^z + e^{-z})(e^z + e^{-z}) - (e^z - e^{-z})(e^z - e^{-z})}{(e^z + e^{-z})^2}$$

$$= \frac{1 - (e^z - e^{-z})^2}{(e^z + e^{-z})^2} = \boxed{1 - \tanh^2(z)}$$

d) Backward Pass

1. Derive the following Gradients

$$\frac{dL}{dW_2}, \quad \frac{dL}{db_2}, \quad \frac{dL}{dW_1}, \quad \frac{dL}{db_1}$$

Derivation of $\frac{dL}{dw_2}$

$L = -\frac{1}{N}\sum\sum y_n \log \hat{y}_n$

$\hat{y} = \text{softmax}(z_2)$

$z_2 = w_2 a_1 + b_2$

$\frac{d\hat{y}}{dz_2} = \frac{d\,\text{softmax}(z_2)}{dz_2} = \text{softmax}(z_2)\cdot(1-\text{softmax}(z_2))$

$\frac{dz_2}{dw_2} = a_1$

$$\frac{dL}{dw_2} = \frac{dL}{d\hat{y}}\frac{d\hat{y}}{dz_2}\frac{dz_2}{dw_2}$$

$$\frac{dL}{d\hat{y}} = -\frac{y}{\hat{y}}$$

$$\frac{dL}{dw_2} = \sum -\frac{y}{\hat{y}} \cdot \left[\text{softmax}(z_2)\cdot(1-\text{softmax}(z_2))\right] \cdot a_1$$

$$= \sum -\frac{y}{\hat{y}} \cdot \hat{y}\cdot(1-\hat{y})\cdot a_1$$

$$= \left(\sum -y\cdot(1-\hat{y})\right)\cdot a_1$$

$$= a_1\cdot\left(\sum -y\cdot(1-\hat{y})\right)$$

$$= a_1\cdot\left(\sum -y\cdot 1 + y\hat{y}\right)$$

$$= a_1\cdot\left(-y\sum 1 + \sum y\hat{y}\right)$$

$$= a_1\cdot(-y+\hat{y})$$

$$\boxed{= a_1\cdot(\hat{y}-y)}$$

derivation of $\frac{dL}{db_2}$

$$\frac{dL}{db_2} = \frac{dL}{d\hat{y}} \frac{d\hat{y}}{dz_2} \frac{dz_2}{db_2}$$

$$\frac{dz_2}{db_2} = 1$$

$$\frac{dL}{d\hat{y}} = (\hat{y} - y)$$

$$\boxed{\frac{dL}{db_2} = (\hat{y} - y) \cdot 1}$$

derivation of $\frac{dL}{dw_1}$

$$\frac{dL}{dw_1} = \frac{dL}{da_1} \cdot \frac{da_1}{dz_1} \cdot \frac{dz_1}{dw_1} \rightarrow X$$

$$(\hat{y} - y) \cdot w_2 \left[\frac{d\tanh}{dz_1}, \frac{d\sigma(z_1)}{dz_1}, \frac{d\,\mathrm{ReLU}(z_1)}{dz_1} \right]$$

$$\frac{dL}{dw_1} = \left[(\hat{y} - y) \cdot w_2 \right] \cdot \text{diff\_actfun}(z_1) \cdot X$$

derivation of $\frac{dL}{db_1}$

$$\frac{dL}{db_1} = \frac{dL}{dz_1} \cdot \frac{dz_1}{db_1}$$

$$= \left[(\hat{y} - y) \cdot w_2 \right] \cdot \text{diff\_actfun}(z_1)$$

e) Training

1. Train the network using different activation function

Include the figures generated in report

1. the relu funtion had an almost linear decision boundary

   - was the second best activation function in terms of accuracy

2. the tanh function had an almost perfect decision boundary

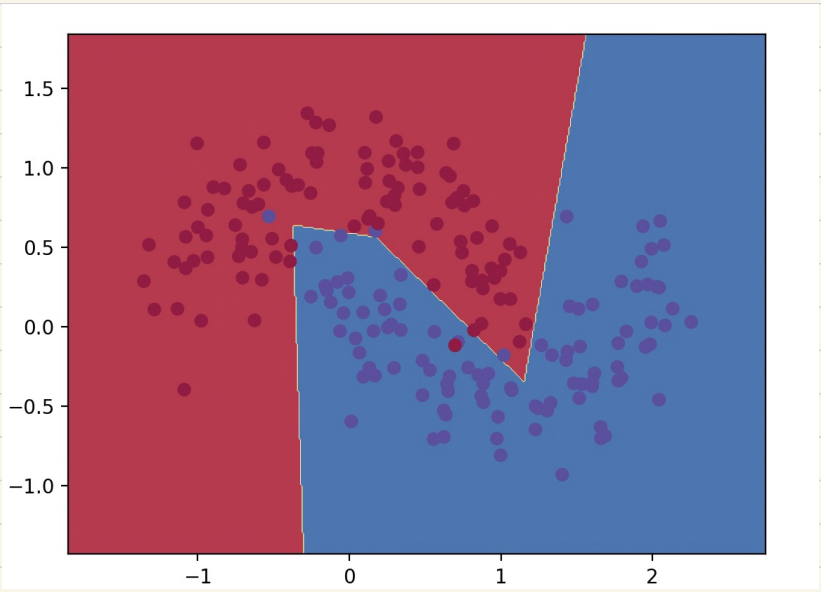
   was the activation function in terms of accurately creating a decision boundary that seperates the two classes

3. the sigmoid function had the worst performance in terms of creating an accurate decision boundary

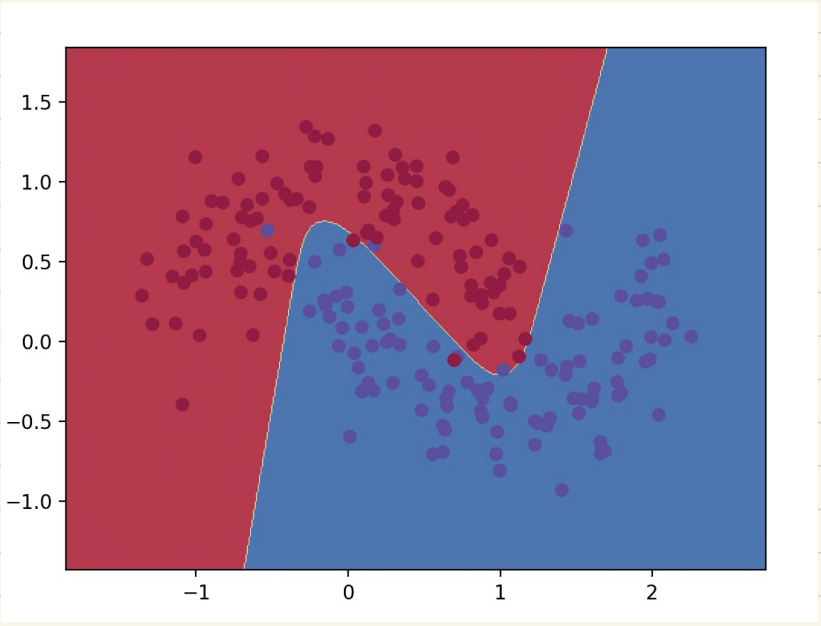
   1. decision boundary was practically none existent
   2. error can be due to numpy not be able to handle values greater than memory allocated

   example: value <= max(int32)

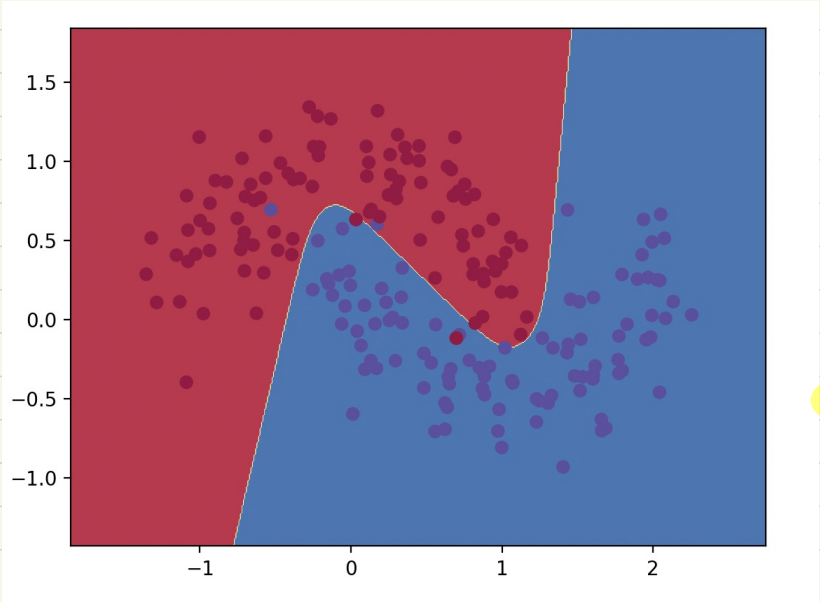
   if value > max(int32)
   return nan



e) relU



e) sigmoid

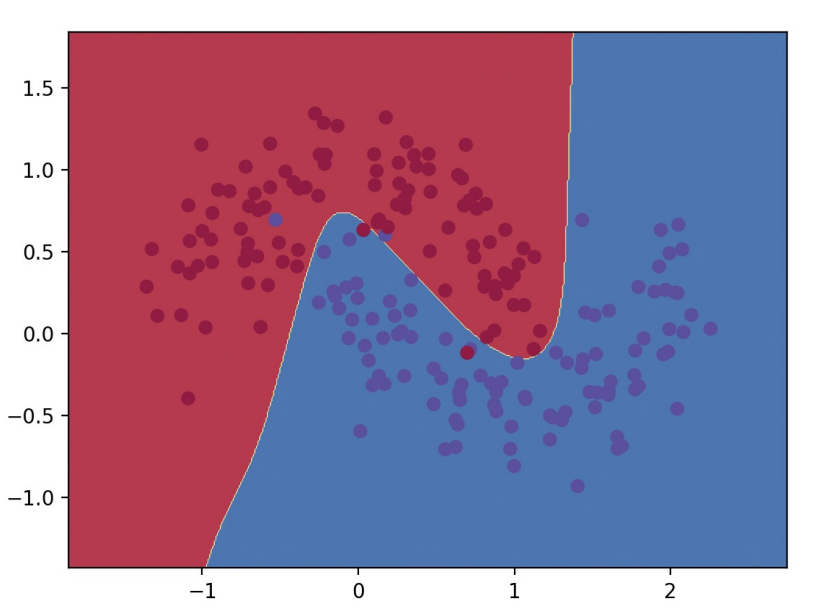


e) tanh

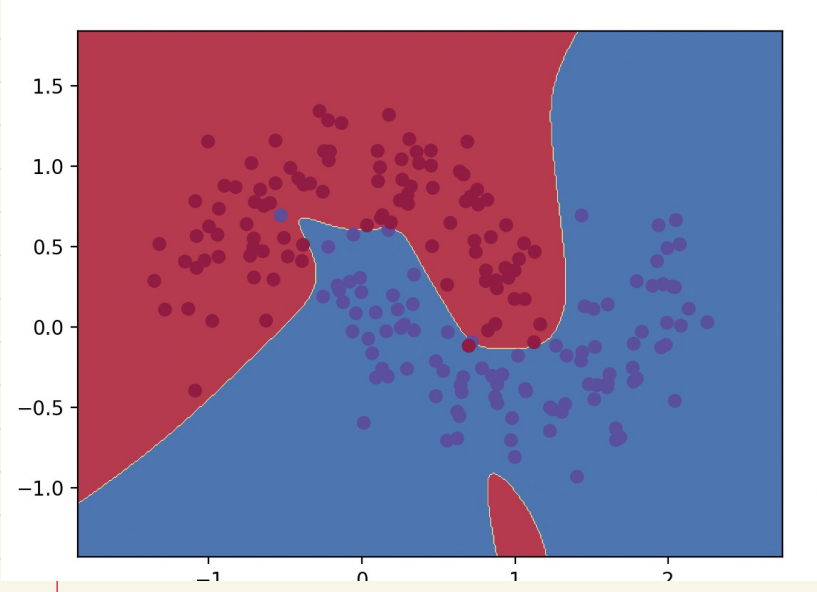
2. Increase the number of hidden units and retrain the network using tanh as the activation function

Include figures in report

- increased the number of hidden units from 3 to 5 then again to 15
- the more hideden units the more accurate the decision boundary gets



e) tanh with [5 hidden layers]



e) tanh with [15 hidden layers]

Training a deeper Network

Write your own n_layer_neural

Requirements

code must be able to accept the following parameters:

(1) number of layers and
(2) layer size

Train on make_moons dataset

- include generated images of
- describe what you observed
- what you find interesting

decision boundary of deep vs shallow neural networks

What I noticed

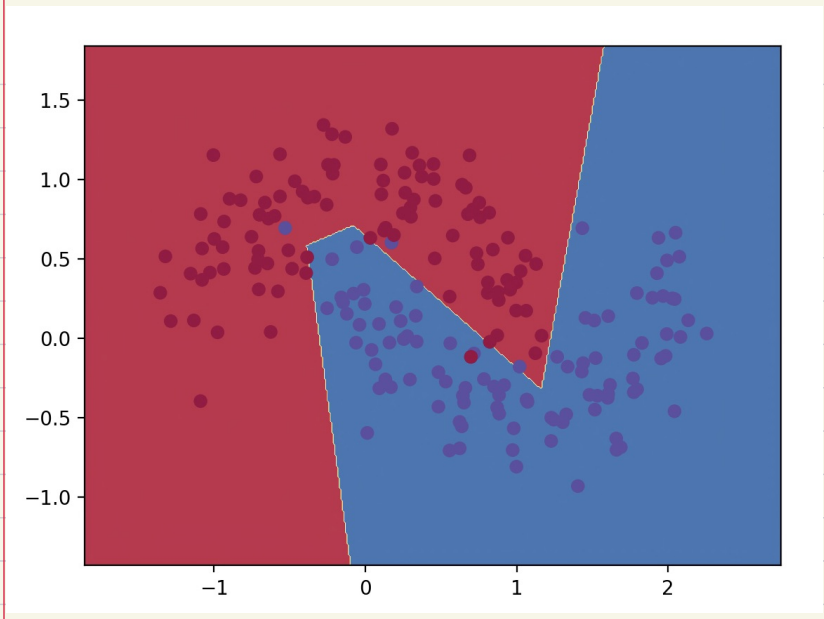
- I saw an increase in accuracy of the decision boundary after increasing the num_hidden_layers from 1 to 5

Why I made the choices I did?

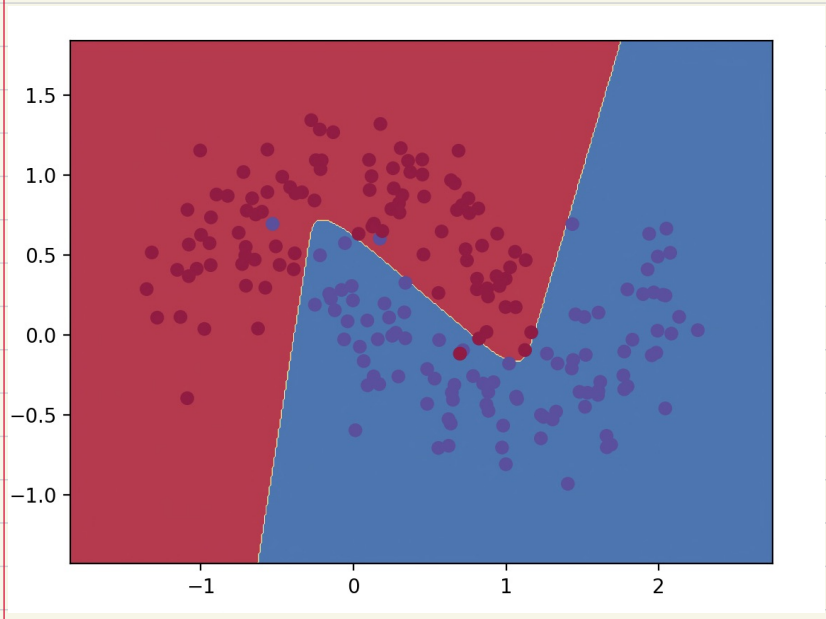
I created a class called layer()

the n_layer_network would create/initialize all the layers specified by the users at the begining to make feedword and backprop easier at the end.

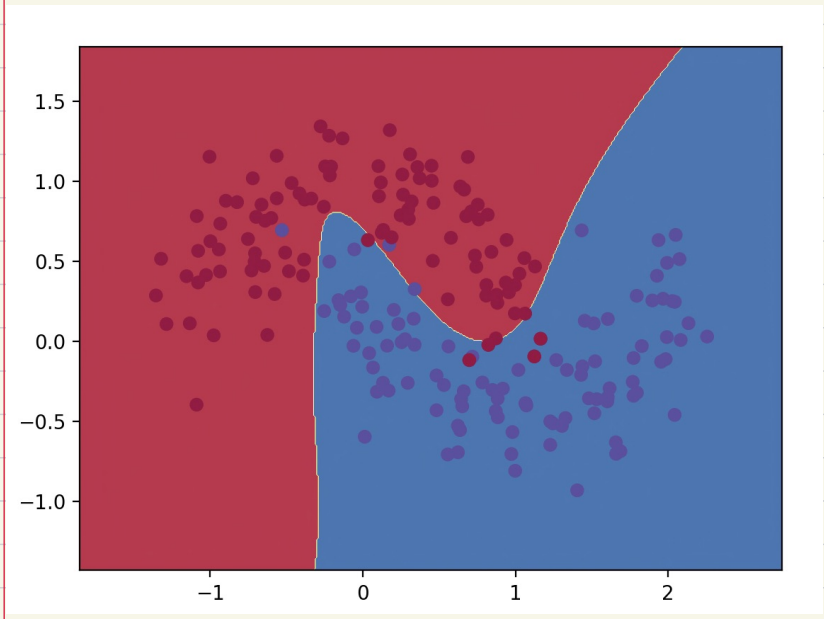
the first and last layer are handled differently than the hidden layers to avoid issues with dimensions later in feedforward & backprop.



f) relu



f) tanh

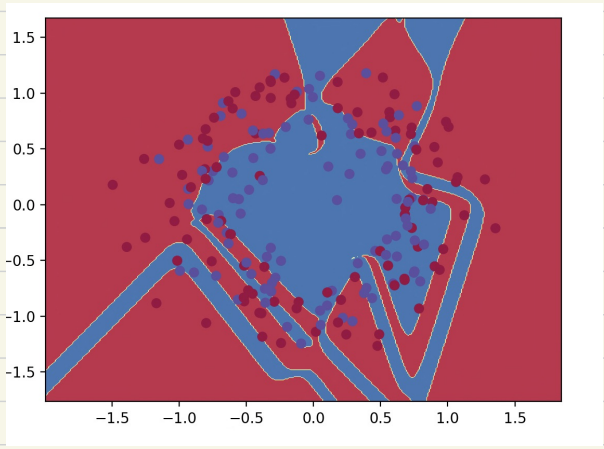


f) sigmoid

Train with Other dataset

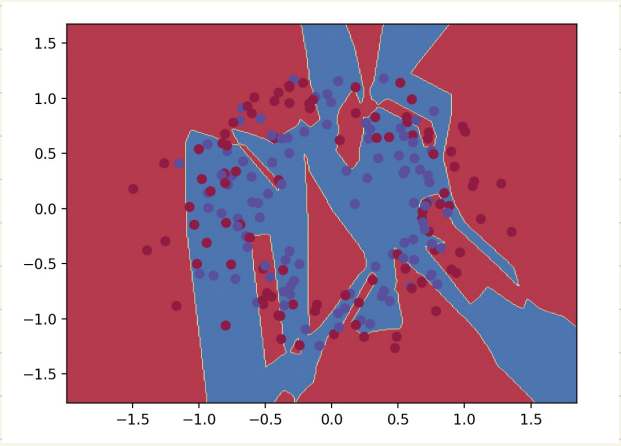
1) using make_circles dataset

input dim = 2 hidden dim = 8 # hidden layers = 3
output dim = 2



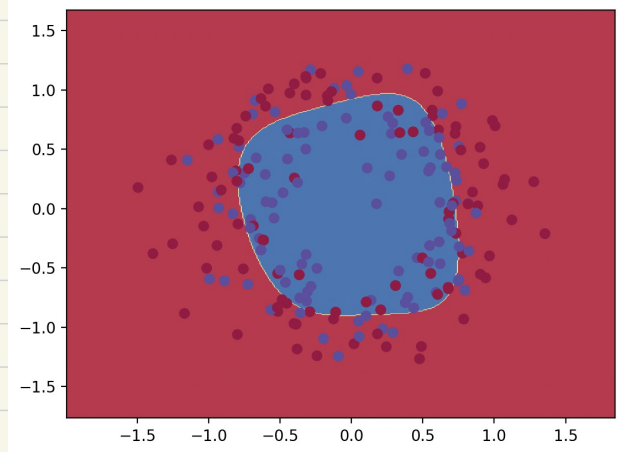
f) tanh

Observation: very scattered unable to make decision boundary due to dataset being random



f) reLU

observation: same as tanh



f) sigmoid

observation: decision boundary is the best among relu & tanh

2 Training a Simple Deep CNN

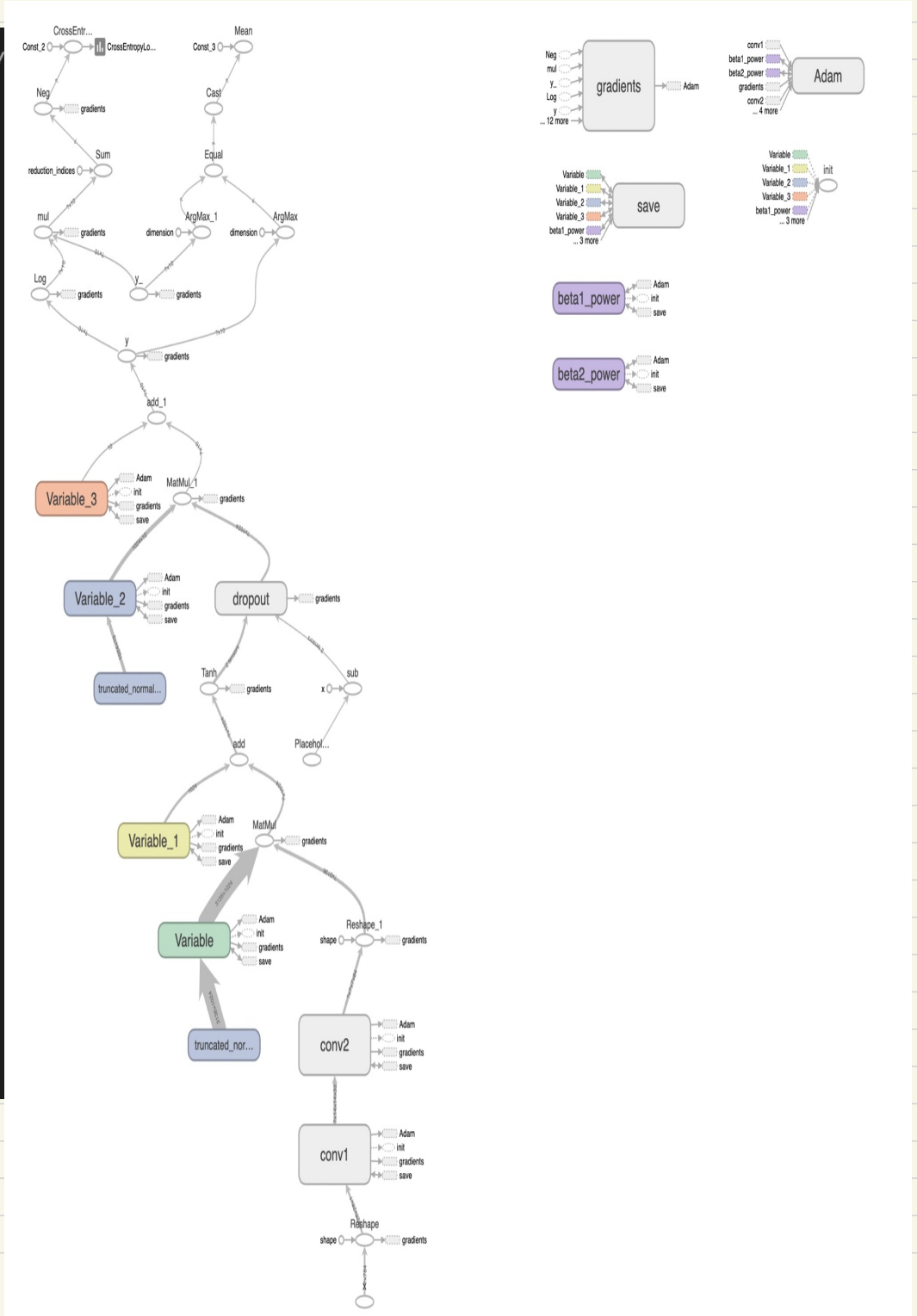
What is the final accuracy?

test accuracy = 0.9837
training took 65.7 seconds

using tanh activation

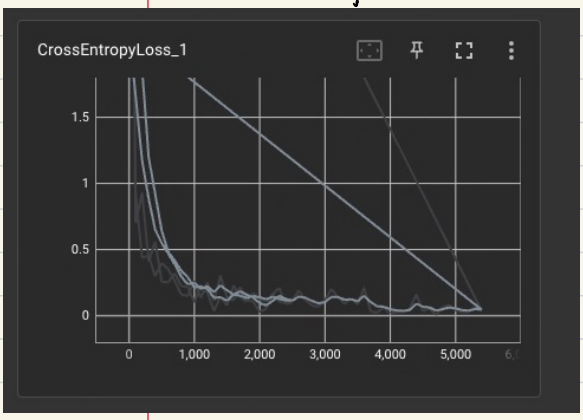
Using python 3.9.12, tensorflow 2.10, and miniconda env

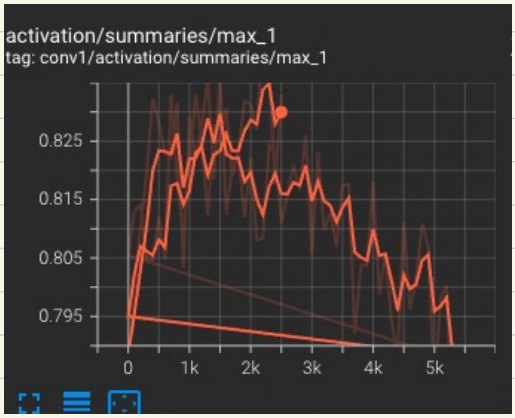
```
Use `tf.global_variables_initializer` instead.
2022-10-05 23:28:41.367707: I tensorflow/compiler/
step 0, training accuracy 0.1
step 100, training accuracy 0.86
step 200, training accuracy 0.86
step 300, training accuracy 0.94
step 400, training accuracy 0.86
step 500, training accuracy 0.96
step 600, training accuracy 0.94
step 700, training accuracy 0.96
step 800, training accuracy 0.98
step 900, training accuracy 0.92
step 1000, training accuracy 0.98
step 1100, training accuracy 0.94
step 1200, training accuracy 0.96
step 1300, training accuracy 1
step 1400, training accuracy 0.94
step 1500, training accuracy 1
step 1600, training accuracy 0.92
step 1700, training accuracy 0.98
step 1800, training accuracy 0.96
step 1900, training accuracy 0.96
step 2000, training accuracy 0.96
step 2100, training accuracy 0.94
step 2200, training accuracy 0.98
step 2300, training accuracy 0.98
step 2400, training accuracy 0.98
step 2500, training accuracy 0.98
step 2600, training accuracy 0.96
step 2700, training accuracy 0.98
step 2800, training accuracy 1
step 2900, training accuracy 0.98
step 3000, training accuracy 1
step 3100, training accuracy 0.96
step 3200, training accuracy 0.96
step 3300, training accuracy 0.98
step 3400, training accuracy 0.98
step 3500, training accuracy 1
step 3600, training accuracy 0.94
step 3700, training accuracy 0.98
step 3800, training accuracy 1
step 3900, training accuracy 1
step 4000, training accuracy 1
step 4100, training accuracy 1
step 4200, training accuracy 1
step 4300, training accuracy 1
step 4400, training accuracy 0.96
step 4500, training accuracy 0.98
step 4600, training accuracy 0.98
step 4700, training accuracy 1
step 4800, training accuracy 1
step 4900, training accuracy 0.98
step 5000, training accuracy 1
step 5100, training accuracy 1
step 5200, training accuracy 1
step 5300, training accuracy 0.96
step 5400, training accuracy 1
test accuracy 0.9846
The training takes 60.394250 second to finish
(tensorflow) jiezy@jiezys-mbp HW1 %
```

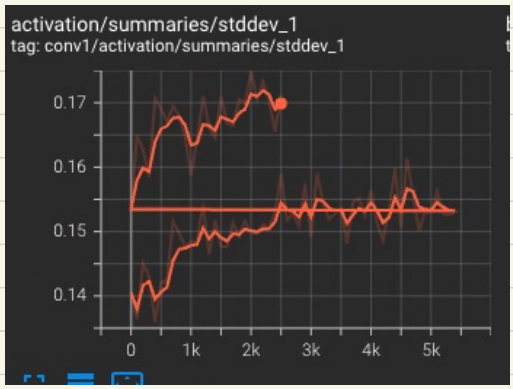


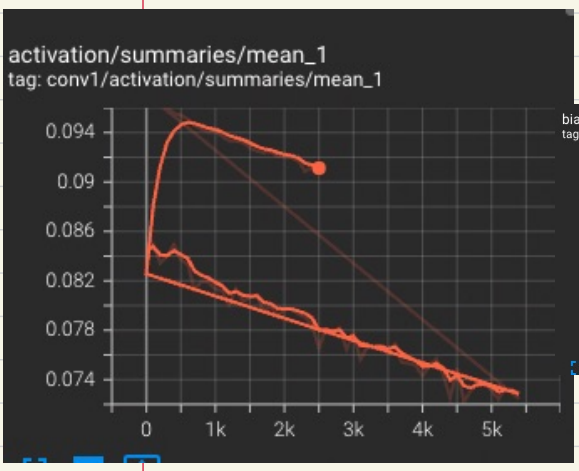
more visualization using tensorboard

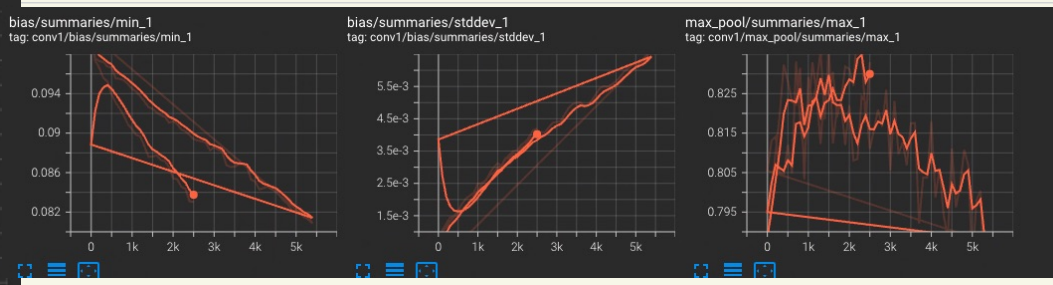
() more fun , tanh activation

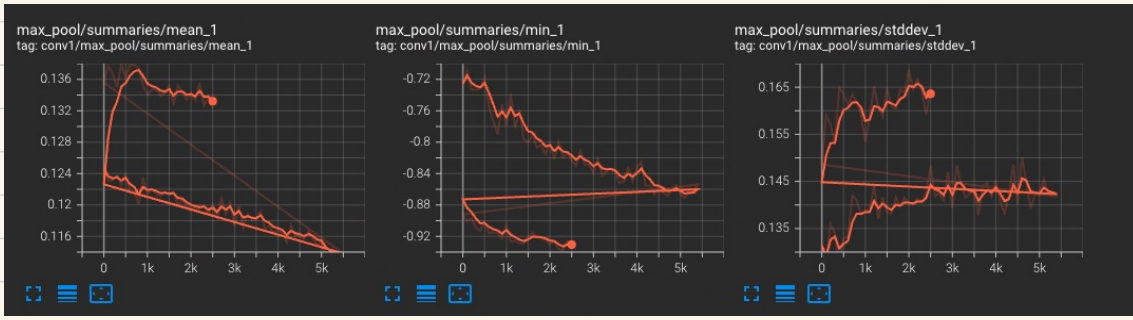


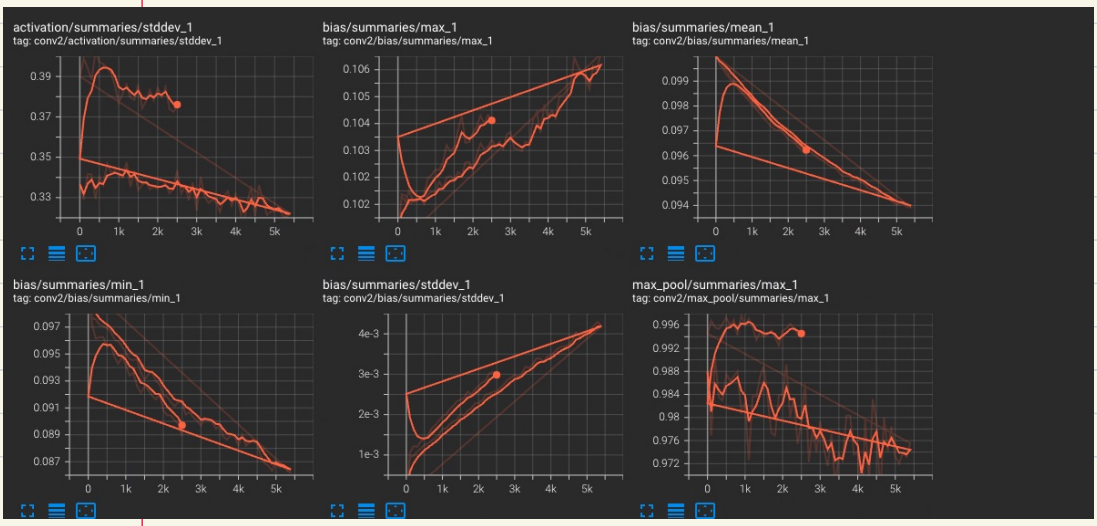












c) using relu activation

```
2022-10-05 23:50:36.945300: I tensorflow/compiler/mlir/ml
step 0, training accuracy 0.02
step 100, training accuracy 0.86
step 200, training accuracy 0.9
step 300, training accuracy 0.94
step 400, training accuracy 0.96
step 500, training accuracy 0.96
step 600, training accuracy 0.94
step 700, training accuracy 0.98
step 800, training accuracy 0.96
step 900, training accuracy 0.94
step 1000, training accuracy 0.94
step 1100, training accuracy 0.98
step 1200, training accuracy 0.94
step 1300, training accuracy 0.98
step 1400, training accuracy 0.98
step 1500, training accuracy 1
step 1600, training accuracy 1
step 1700, training accuracy 0.98
step 1800, training accuracy 0.98
step 1900, training accuracy 0.96
step 2000, training accuracy 1
step 2100, training accuracy 0.98
step 2200, training accuracy 0.94
step 2300, training accuracy 0.98
step 2400, training accuracy 0.98
step 2500, training accuracy 0.98
step 2600, training accuracy 0.98
step 2700, training accuracy 0.98
step 2800, training accuracy 0.98
step 2900, training accuracy 0.96
step 3000, training accuracy 1
step 3100, training accuracy 0.96
step 3200, training accuracy 0.98
step 3300, training accuracy 1
step 3400, training accuracy 1
step 3500, training accuracy 0.98
step 3600, training accuracy 1
step 3700, training accuracy 1
step 3800, training accuracy 0.98
step 3900, training accuracy 0.98
step 4000, training accuracy 0.98
step 4100, training accuracy 1
step 4200, training accuracy 1
step 4300, training accuracy 1
step 4400, training accuracy 0.98
step 4500, training accuracy 1
step 4600, training accuracy 1
step 4700, training accuracy 0.98
step 4800, training accuracy 0.96
step 4900, training accuracy 0.98
step 5000, training accuracy 1
step 5100, training accuracy 1
step 5200, training accuracy 1
step 5300, training accuracy 0.98
step 5400, training accuracy 1
test accuracy 0.9867
The training takes 60.994998 second to finish
(tensorflow) jiezy@jiezys-mbp HW1 %
```