Collaborators: Marco Hendra

1)

A) False. Neural networks are non-convex, meaning gradient descent won't guarantee to provide best results.
B) False as doing so would result in the same values being passed onto the next layer, which would make the final output predictable which removes the purpose of using a neural network.
C) True as we use non-linear activation functions to introduce non-linearity into the network to allow us to model boundaries that are non-linear. If not, then it becomes similar to a linear model.
D) False. While forward pass encompasses matrix multiplication and activation function, backpropagation does something similar where it updates weight/bias and calculates gradients. The difference in number of calculations isn't exponential as they are somewhat similar, so it's not prohibitively larger.
E) False. There are a lot of considerations done when picking a model to best fit its problem(such as designing its structure and picking it's hyperparameters), which may not reflect all possible problems. Furthermore, there could be simpler models that may better suite a problem(such as linear regression for linear problems).

2)

By the definition of kernel:

$$K(x, x') = \phi(x) * \phi(x')$$

For a general i-th component:

$$= \left(\frac{1}{\sqrt{i!}} e^{\frac{-x^2}{2}} x^i\right) * \left(\frac{1}{\sqrt{i!}} e^{\frac{-x'^2}{2}} x'^i\right)$$

$$= \frac{1}{\sqrt{i!}} \frac{1}{\sqrt{i!}} e^{\frac{-x^2}{2}} e^{\frac{-x'^2}{2}} x^i x'^i$$

$$= \frac{1}{i!} e^{\frac{-(x^2+x'^2)}{2}} x^i x'^i$$

Combining all components together(as only the value i changes for each component):

$$= e^{\frac{-(x^2+x'^2)}{2}} \sum_{i=0}^{\infty} \frac{(xx')^i}{i!}$$

Using the definition of Taylor expansion, where $e^y = \sum_{n=0}^{\infty} \frac{y^n}{n!}$:

$$= e^{\frac{-(x^2+x'^2)}{2}} * e^{xx'} = e^{\frac{-(x^2+x'^2)}{2}} * e^{\frac{2xx'}{2}}$$

$$= e^{\frac{-x^2-x'^2+2xx'}{2}}$$

$$= e^{\frac{-(x-x')^2}{2}}$$

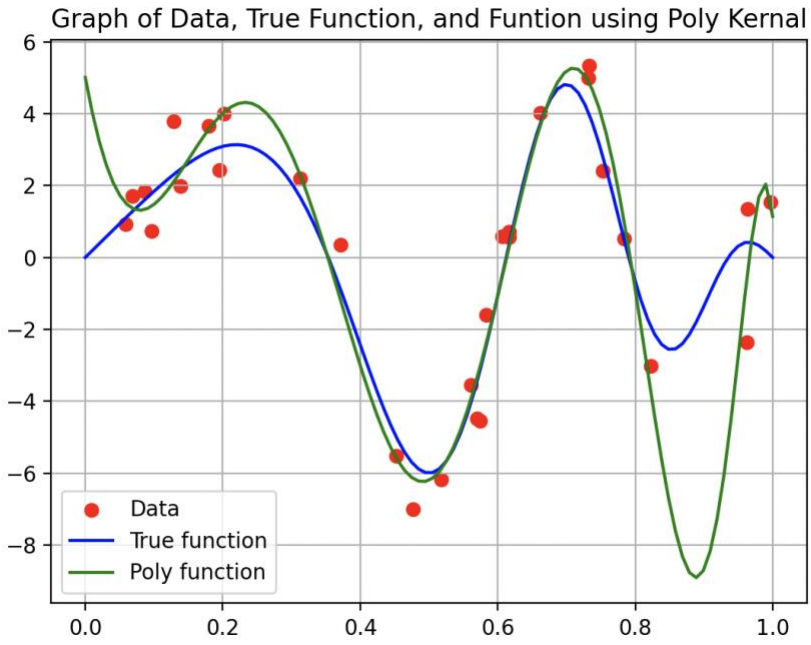
3)

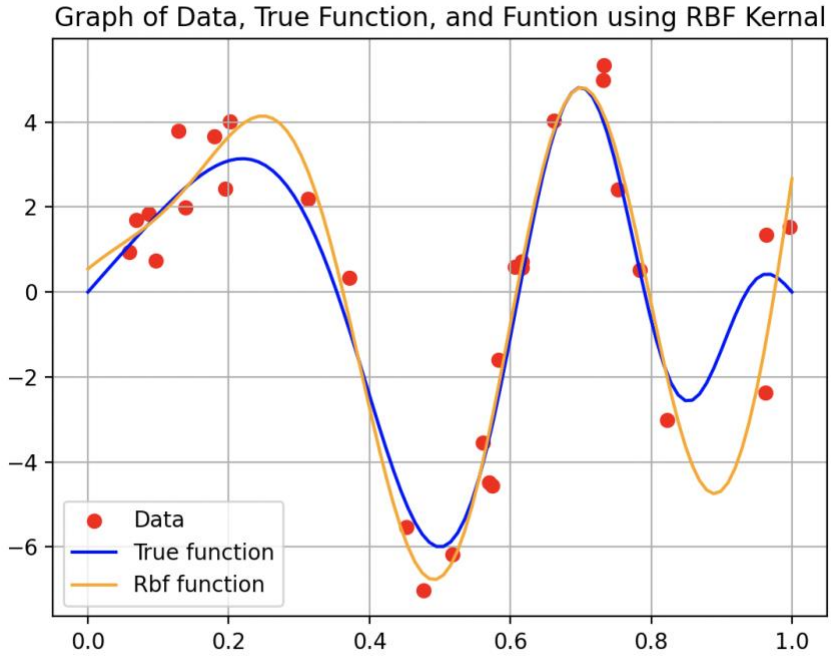
A)

For poly kernel, the best lambda and d hyperparameters are lamba = 1e-05 and d = 15.0.

For RBF kernel, the best lambda is 0.001 and gamma is 11.201924992299844.

B)

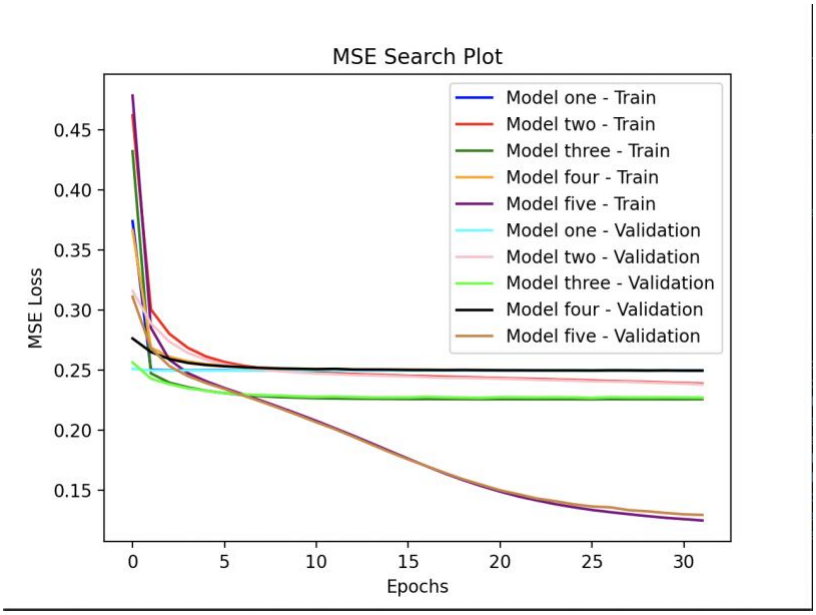


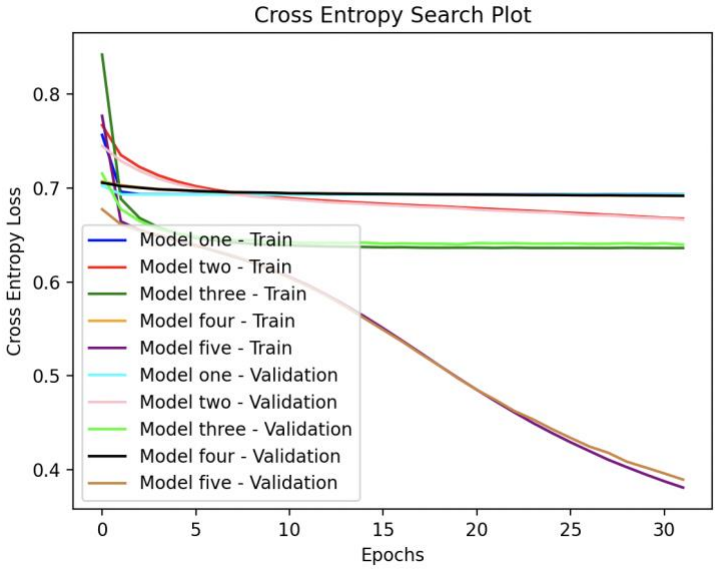


4)

A)

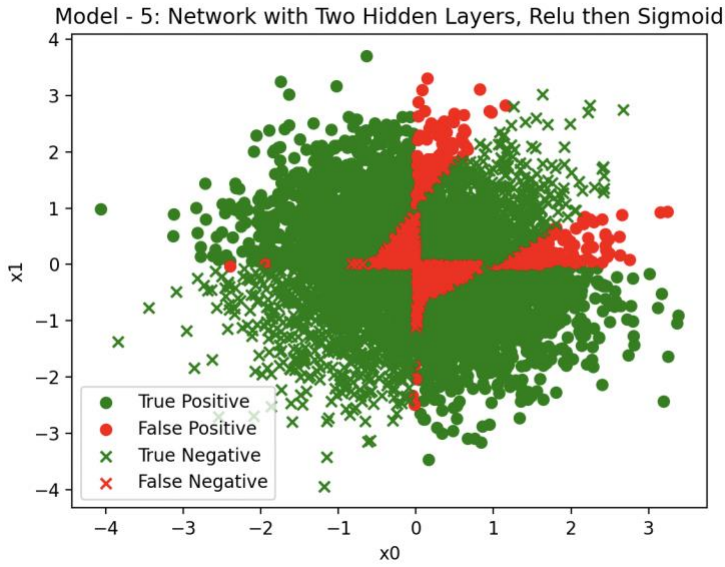
B)


MSE Search Plot
MSE Loss
Epochs
Model one - Train
Model two - Train
Model three - Train
Model four - Train
Model five - Train
Model one - Validation
Model two - Validation
Model three - Validation
Model four - Validation
Model five - Validation

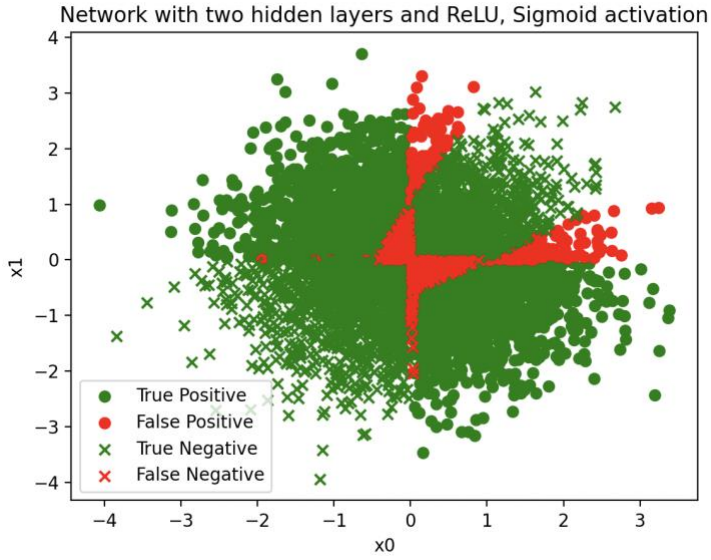

Cross Entropy Search Plot
Cross Entropy Loss
Epochs
Model one - Train
Model two - Train
Model three - Train
Model four - Train
Model five - Train
Model one - Validation
Model two - Validation
Model three - Validation
Model four - Validation
Model five - Validation

C)

For the MSE search, the best performing architecture was model 5, Network with two hideen layers and ReLu, Sigmoid activation after corresponding hidden layers. It's accuracy was 0.8684.

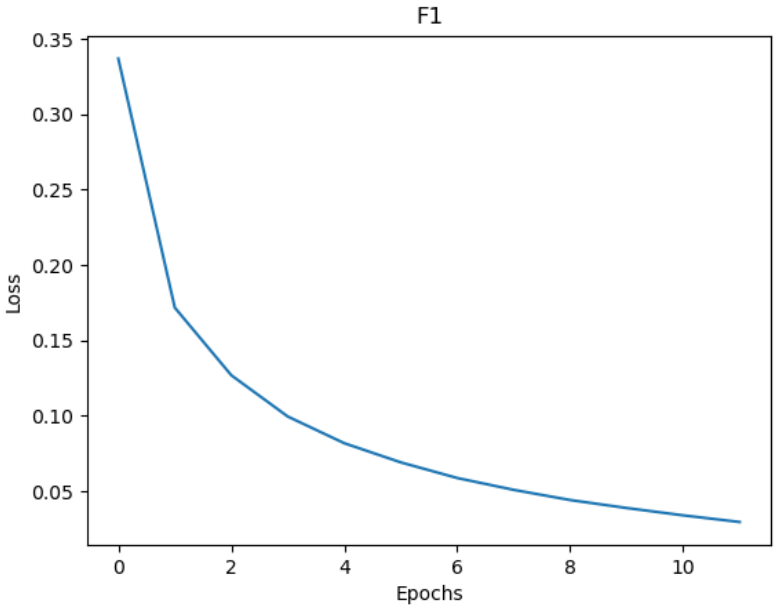


For cross entropy search, the best performing model was the fifth mode, as specified in the spec, as well. Its accuracy was 0.8896.



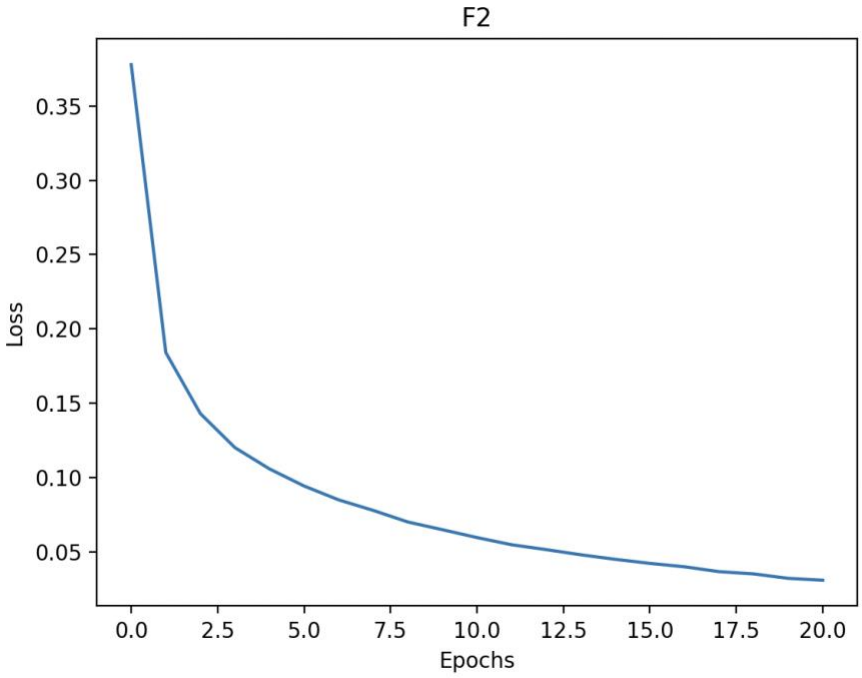
5)

A)



F1 is the wide/shallow network. The loss on the test data was 0.0927. The Accuracy on the test data was 0.9733.

B)



F2 is the deep/narrow network. The loss on the test data was 0.1320. The Accuracy was 0.9684.

C)

The total number of parameters for the shallow, wide model is 50890. The total number of parameters for the narrow, deep model is 26506.

The shallow, wide is a better model than narrow, deep as the large number of parameters causes it to be more complex, which may cause overfitting due to the number of features contributing to the output. Though this may not be a bad thing as it can raise the test accuracy(generalizing the model). Although if you care about memory space, given your constraints narrow/shallow may serve you better.

6)

27 hours