Assignment 2: Neural Networks

Summary

Deep Neural Network typically involves the presence of an input layer that receives the raw data, transformed by several input layers and an output layer that produces the final outcome as a prediction. A total of sixteen models, including a base model, was constructed to examine various configurations and its impact on the performance metric of the model. A closer investigation of the metrics, namely, accuracy and loss values ranged between the values of 88.13% to 88.91% and 0.0857 to 0.4145 respectively.

Base model with a loss of 0.1881 and a test accuracy of 0.8843 was used to compare across various models that were built..The increase in the number of hidden layers contributed to greater model complexity, which in turn reduced the number of training iterations i.e epochs needed before overfitting occurred. For example, Model 2 had (three hidden layers, 16 units), achieved 88.38% accuracy in just three epochs, whereas in comparison to Model 1 (that had one hidden layer, 16 units) required four epochs to reach 88.78% accuracy. Similarly, increasing the number of units per layer also reduced the number of epochs needed before overfitting. It is interesting that model 15 with just one hidden layer and 8 units performed comparatively better with an accuracy of 88.69%, that model 4 with three layers and 64 units suggesting that a higher node count does not necessarily increase the accuracy. This also suggests that increase in the number of units can lead to overfitting, as seen in model 4 with three hidden layers and 64 units. A balanced architecture as seen on Model 3 (with three hidden layers, 32 units) may perform better than excessively deep models.

<u>Change in Loss function and its comparison across models:</u>

A change in the loss function from 'binary_crossentropy' to 'mse' resulted in a significant reduction in loss but as seen in Model 8 (MSE loss, 1 hidden layer, 32 units) that had a loss of 0.0857 with an accuracy of 88.46%, which is comparable across many binary cross-entropy models. Similarly, Model 10 (MSE loss, 2 layers of 32 units) that had an additional layer led to a low loss of 0.0862 and an accuracy of 88.26%. These results suggest that while MSE can minimize loss effectively, it does not necessarily enhance classification accuracy, suggesting a lesser number of layers with binary cross-entropy a better choice. It is also important to note that models with MSE as its loss function have lower loss function.

<u>Change in Activation function and its comparison across models:</u>

The two activation functions used were `relu` and `tanh`. The change in activation function of Model 10 (`relu`, two hidden layers, 32 units) to Model 11(`tanh`, two hidden layers, 32 units) by keeping the number of hidden layers and units the same, led to `tanh` models reach faster convergence to an earlier epoch of 2 than an epoch 5 as seen in model 10. This is also indicated by the accuracy of Model 11 with 88.61% than Model 10 with 88.26%. However, it is to be noted that the loss function values of Model 10 and Model 11 are 0.0862 and 0.254 respectively, possibly due to vanishing gradient issues in deeper networks.

Regularization and Dropout results:

Deep Neural Networks constructed here, were firstly allowed to overfit in order for the network to learn and pick up on patterns and trends in the data and then subjected to various techniques such as Regularization and Dropout, to prevent it from overfitting. Consider Model 4 (three hidden layers, 64 units, `relu`, `binary_crossentropy`) and Model 12 (three hidden layers, 64 units, `relu`, `binary_crossentropy`). Model 12 was regularized (L2 = 0.001) and hence reached faster convergence to an earlier epoch of 3, with an increased accuracy of 88.60% in comparison to Model 4 that reached convergence at epoch 4, with a lesser accuracy of 88.29%. This suggests that regularization adds a penalty term to the loss function to constrain model complexity. The base model (two hidden layers, 16 units, `relu`, binary cross-entropy), achieved a loss of 0.1881 and an accuracy of 88.43%, while Model 13, (two hidden layers, 16 units, `relu`, binary cross-entropy) incorporates L1 regularization (1 = 0.0001), resulted in a higher loss of 0.3573 but slightly improved accuracy of 88.61%.

Considering Model 3 (three hidden layers, 32 units, `relu`, binary cross-entropy) and Model 14 (three hidden layers, 32 units, `relu`, binary cross-entropy), Model 14 had an increased accuracy of 88.91% while Model 13 had 88.13% as dropout layers were introduced in Model 14 suggesting that dropout randomly disables neurons during training to improve generalization.

Conclusion and recommendations:

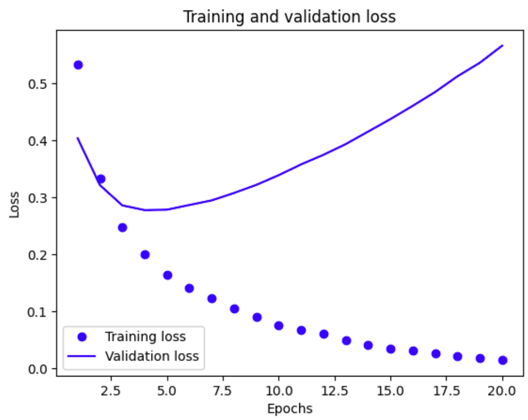
Model learns faster when there is an increase in the number of hidden layers but can lead to overfitting. ReLU activation remains superior compared to Tanh and Binary cross-entropy is preferable for classification problems over MSE loss, when compared across models. L1 regularization is better than L2 as L1 loss value in Model 13 was lesser than L2 loss value in Model 12. Dropout improves generalization and hence increased accuracy, making Model 14 the best choice. It is also important to note that Dropout aids to prevent the model from overfitting. Other methods to address overfitting include regulation of learning rate, batch size in each epoch, tuning of hyperparameters, regulation of model capacity.

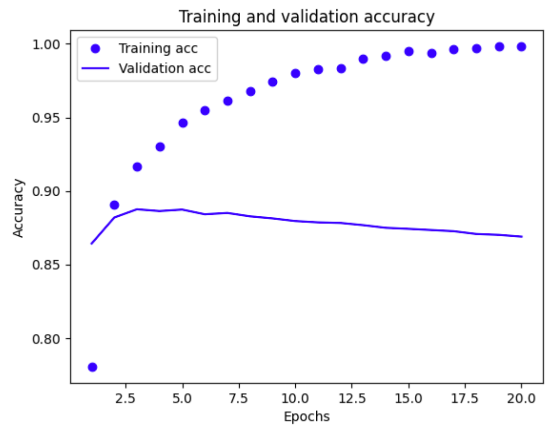
Tabular column of all sixteen models and its configuration:

Each model and its recorded values with respect to hidden layers, units, Activation function, loss value of the test set, number of Epochs (iterations), Loss function and the techniques used to prevent overfitting such as regularization and dropout methods.

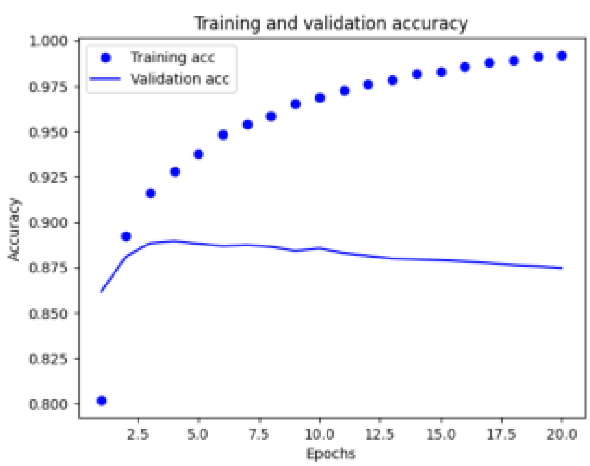
Model	Hidden layer	Units	Activation	Loss function	Epochs	Loss value of Test	Test accuracy in percentage (%)	Technique used
base model	2	(16,16)	relu	binary_crossentropy	4	0.1881	88.43	
model 1	1	16	relu	binary_crossentropy	4	0.2075	88.78	
model 2	3	(16,16,16)	relu	binary_crossentropy	3	0.214	88.38	
model 3	3	(32,32,32)	relu	binary_crossentropy	4	0.1811	88.13	
model 4	3	(64,64,64)	relu	binary_crossentropy	4	0.3014	88.29	
model 5	1	64	relu	binary_crossentropy	4	0.288	88.46	
model 6	1	32	relu	binary_crossentropy	5	0.2875	88.43	
model 7	2	(64,32)	relu	binary_crossentropy	3	0.2969	88.16	
model 8	1	32	relu	mse	7	0.0857	88.46	
model 9	3	(64,64,32)	relu	binary_crossentropy	3	0.2837	88.52	
model 10	2	(32,32)	relu	mse	5	0.0862	88.26	
model 11	2	(32,32)	tanh	mse	2	0.2541	88.61	
model 12 (L2)	3	(64,64,64)	relu	binary_crossentropy	3	0.4145	88.6	Regularization (l2 = 0.001)
model 13 (L1)	2	(16,16)	relu	binary_crossentropy	4	0.3573	88.61	Regularization (l1= 0.0001)
model 14	3	(32,32,32)	relu	binary_crossentropy	4	0.2882	88.91	Dropout
model 15	1	8	relu	binary_crossentropy	7	0.2822	88.69	
model 16	3	(32,32,32)	tanh	binary_crossentropy	3	0.3126	88.22	Dropout+ `tanh`

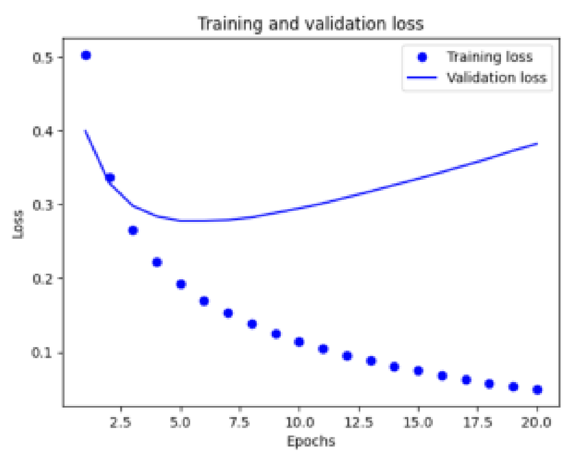
Base Model: The base model, with 2 hidden layers of 16 units each, ReLU activation, and binary cross-entropy loss, achieved 88.43% test accuracy after 4 epochs of training, serving as a benchmark for comparison with other model variations.



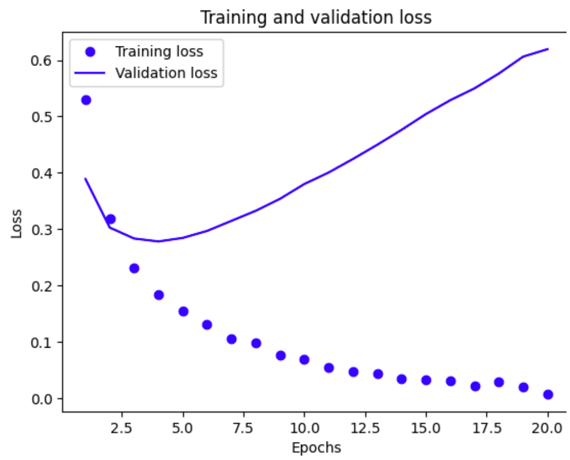


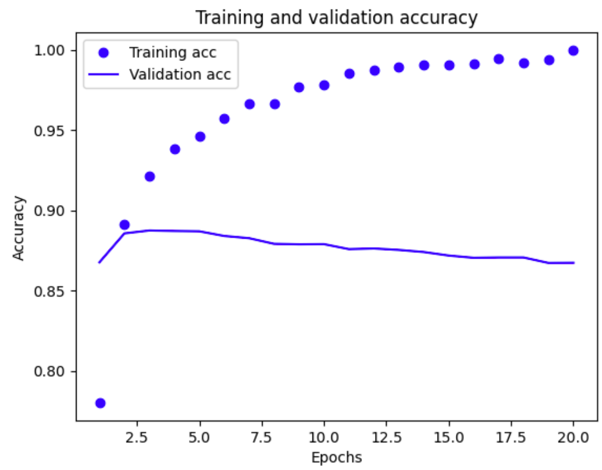
Model 1: Model 1, featuring a single hidden layer with 16 units, ReLU activation, and binary cross-entropy loss, achieved 88.78% test accuracy after 4 epochs of training.



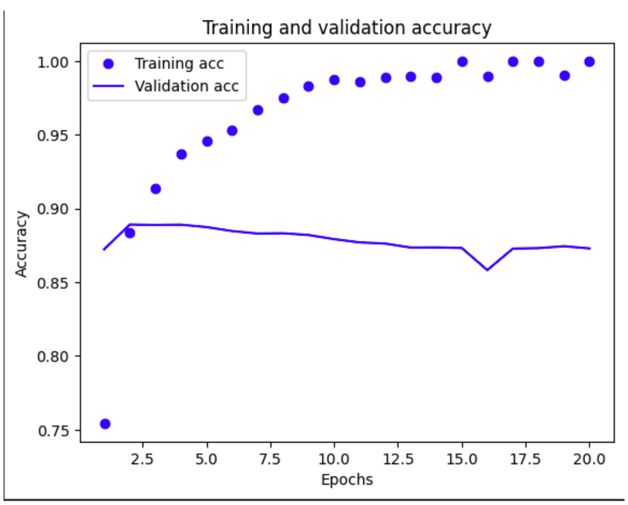


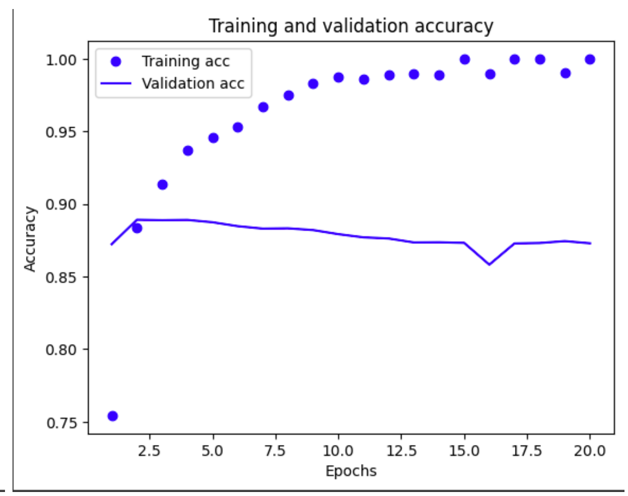
Model 2: **Model 2, employing 3 hidden layers with 16 units each, ReLU activation, and binary cross-entropy loss, attained 88.38% test accuracy after 3 epochs of training.**



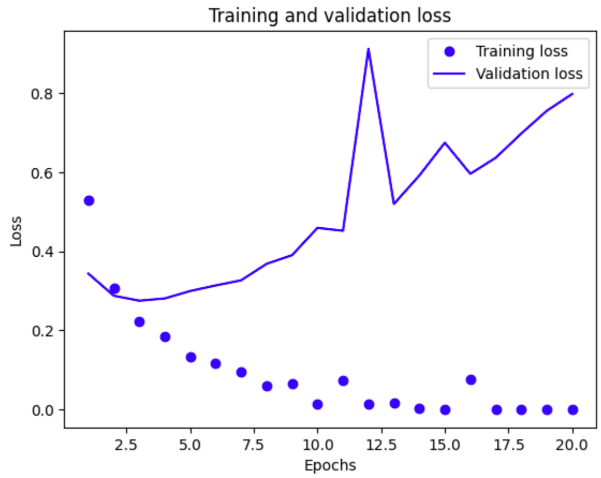


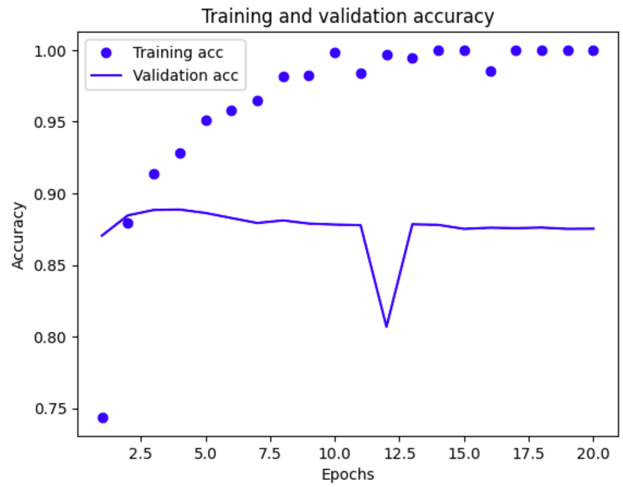
Model 3: **Model 3, utilizing 3 hidden layers with 32 units each, ReLU activation, and binary cross-entropy loss, achieved 88.13% test accuracy after 4 epochs of training.**



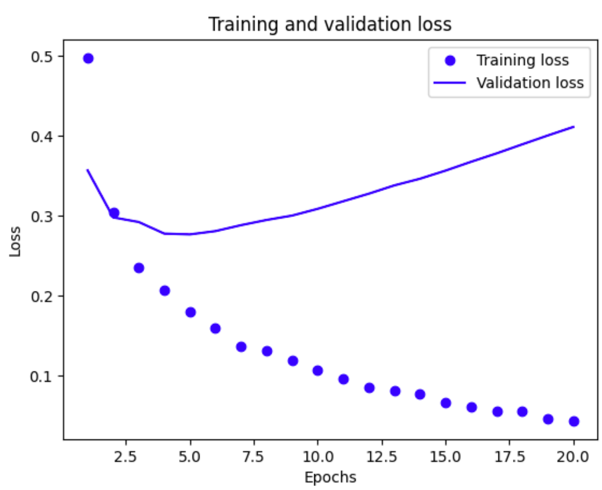


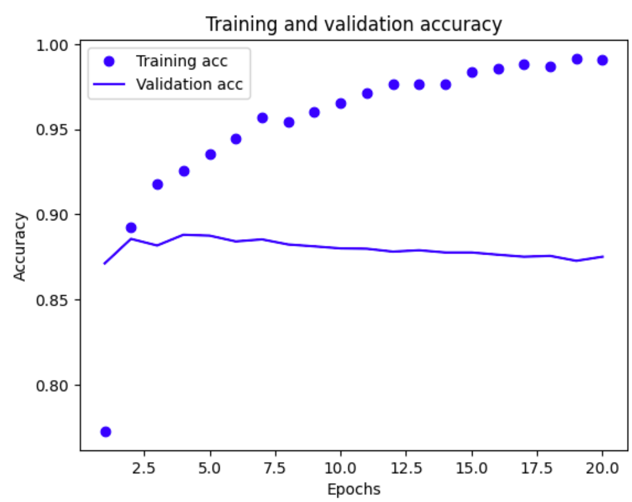
Model 4: **Model 4, incorporating 3 hidden layers with 64 units each, ReLU activation, and binary cross-entropy loss, reached 88.29% test accuracy after 4 epochs of training.**



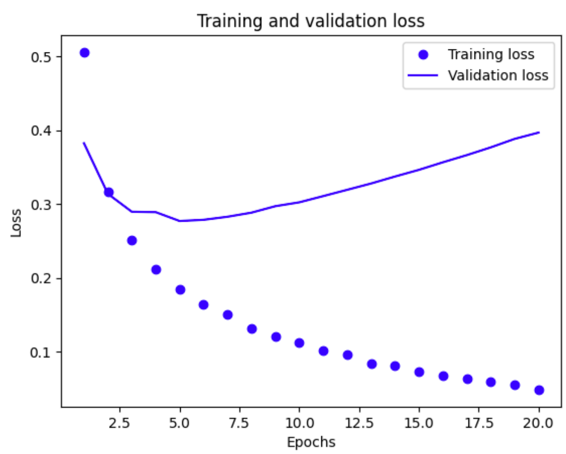


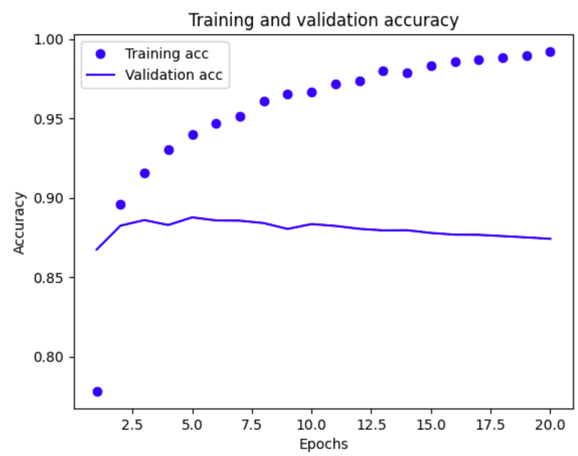
Model 5: **Model 5, consisting of a single hidden layer with 64 units, ReLU activation, and binary cross-entropy loss, achieved 88.46% test accuracy after 4 epochs of training.**



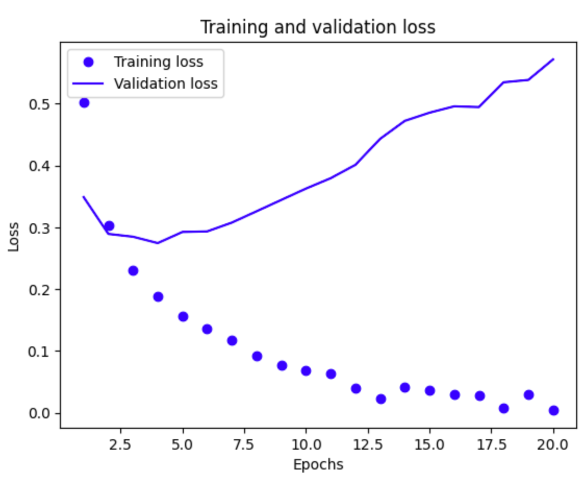


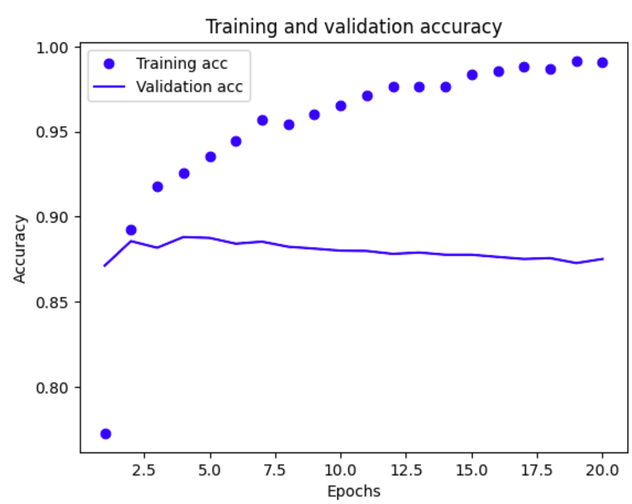
Model 6: **Model 6, featuring a single hidden layer with 32 units, ReLU activation, and binary cross-entropy loss, attained 88.43% test accuracy after 5 epochs of training.**



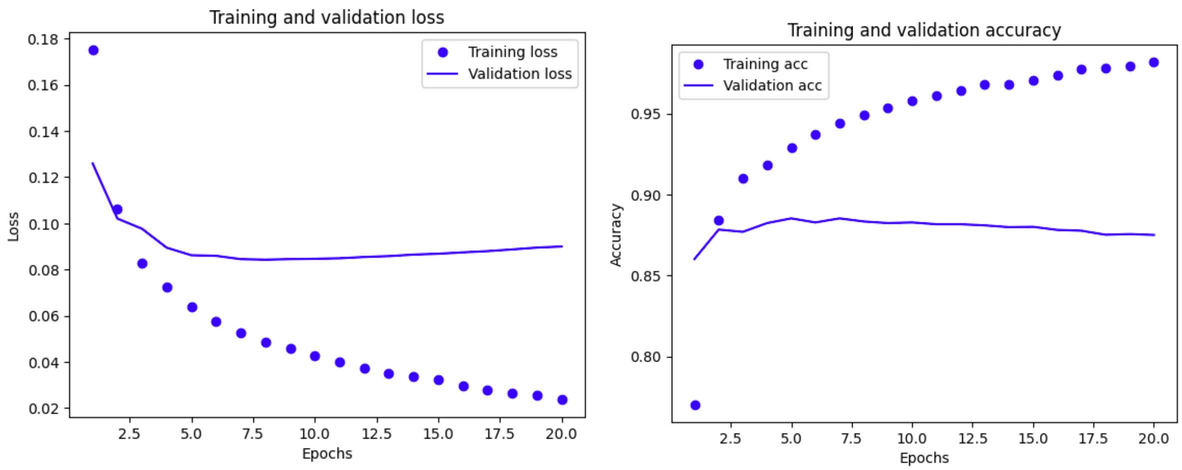


Model 7: **Model 7, employing 2 hidden layers with 64 and 32 units respectively, ReLU activation, and binary cross-entropy loss, achieved 88.16% test accuracy after 3 epochs of training.**

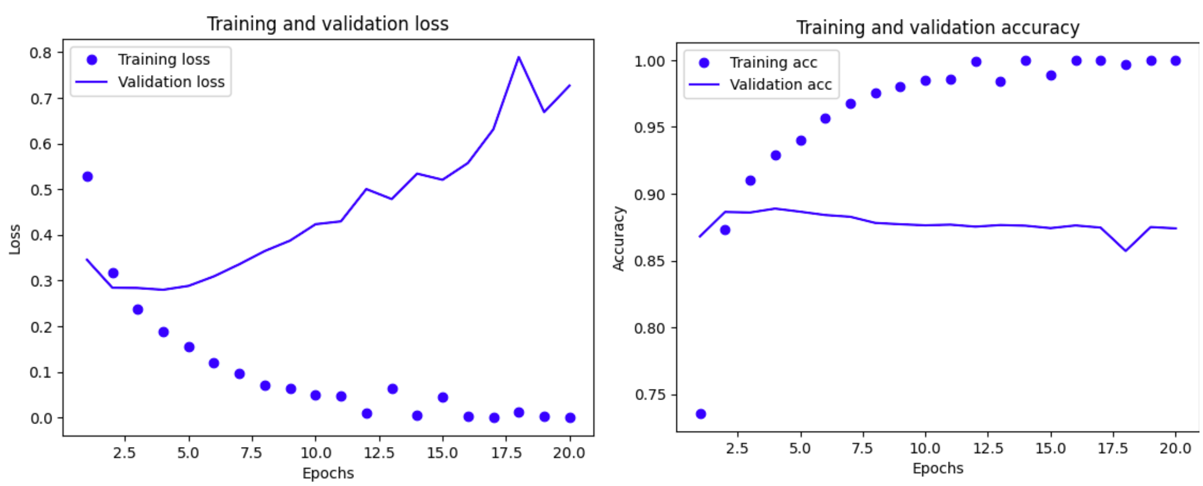




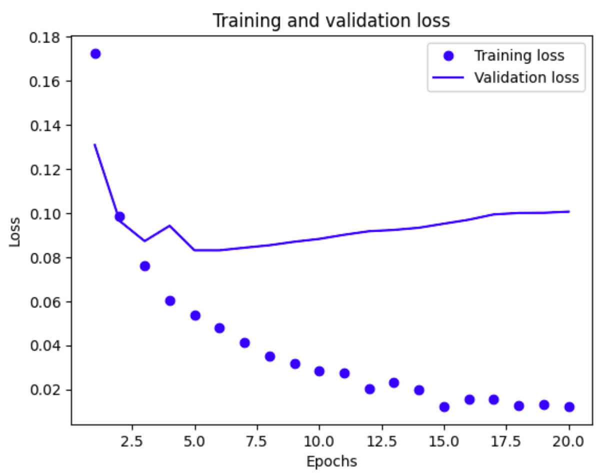
Model 8: Model 8, utilizing a single hidden layer with 32 units, ReLU activation, and mean squared error (MSE) as the loss function, achieved 88.46% test accuracy after 7 epochs of training.

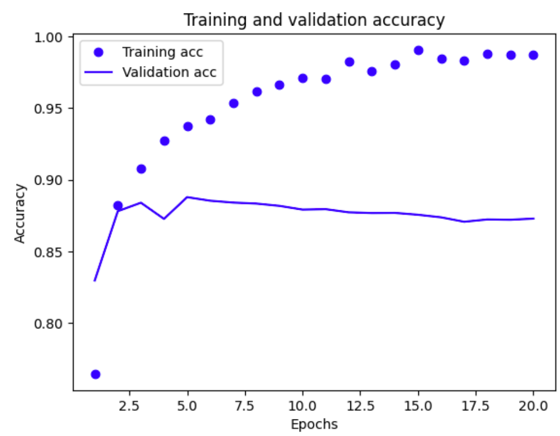


Model 9: Model 9, featuring 3 hidden layers with 64, 64, and 32 units respectively, ReLU activation, and binary cross-entropy loss, attained 88.52% test accuracy after 3 epochs of training.

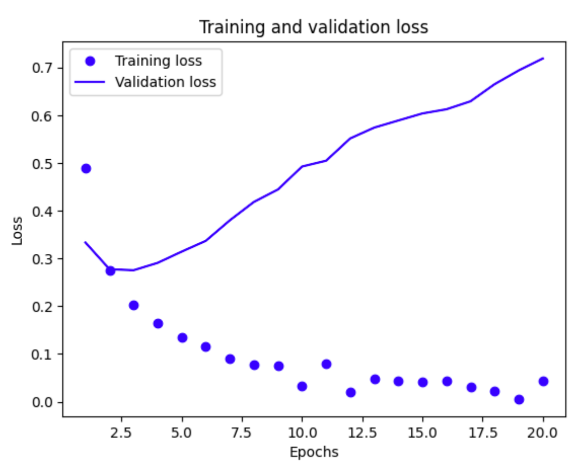


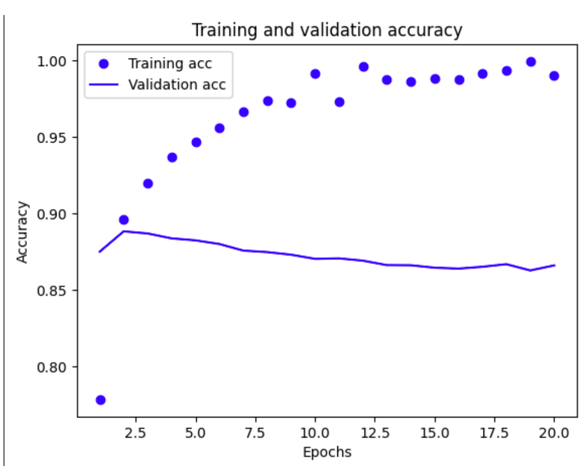
Model 10: Model 10, incorporating 2 hidden layers with 32 units each, ReLU activation, and mean squared error (MSE) as the loss function, reached 88.26% test accuracy after 5 epochs of training.



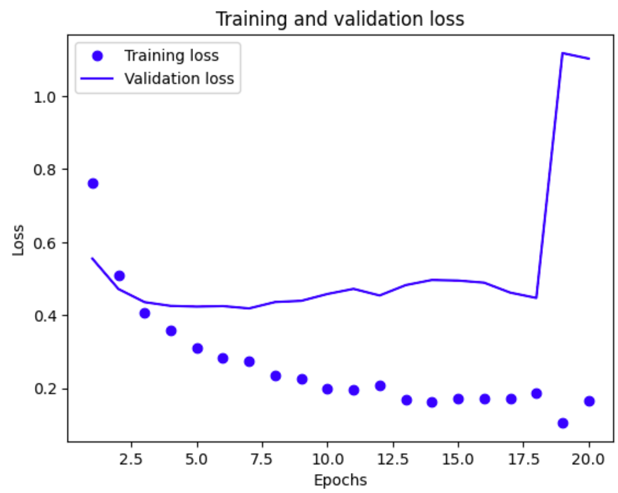


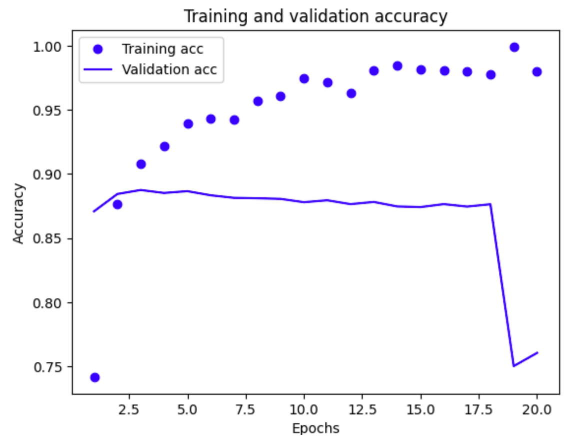
Model 11: Model 11, employing 2 hidden layers with 32 units each, tanh activation, and mean squared error (MSE) as the loss function, achieved 88.61% test accuracy after 2 epochs of training.



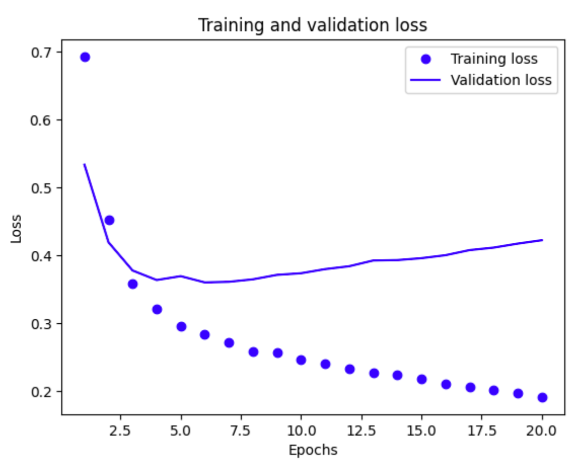


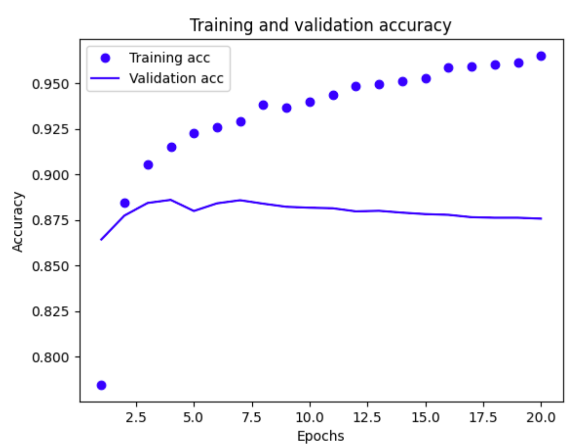
Model 12: **Model 12, utilizing 3 hidden layers with 64 units each, ReLU activation, binary cross-entropy loss, and L2 regularization (l2 = 0.001), achieved 88.60% test accuracy after 3 epochs of training.**



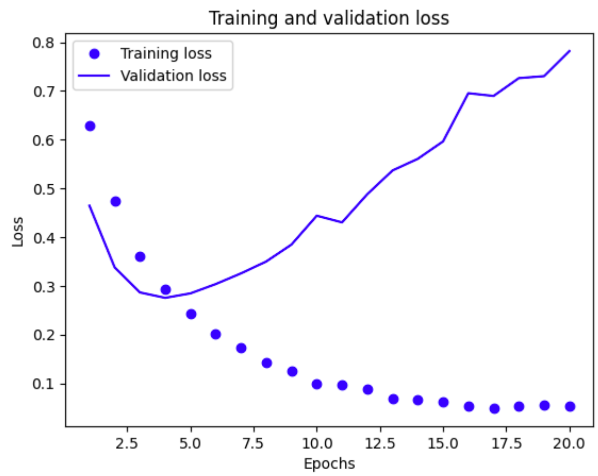


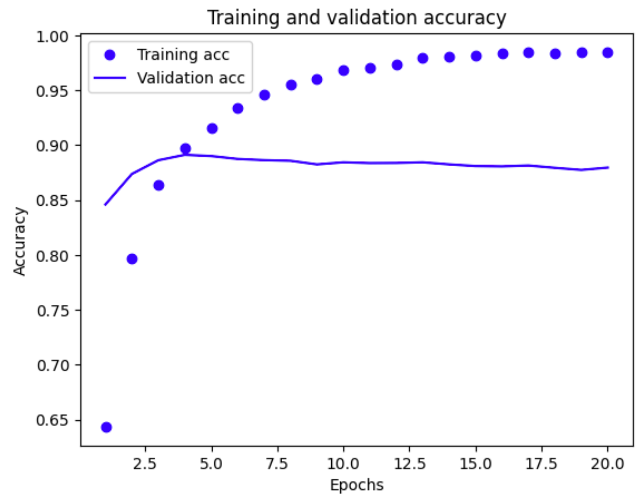
Model 13: **Model 13, featuring 2 hidden layers with 16 units each, ReLU activation, binary cross-entropy loss, and L1 regularization (l1 = 0.0001), attained 88.61% test accuracy after 4 epochs of training.**



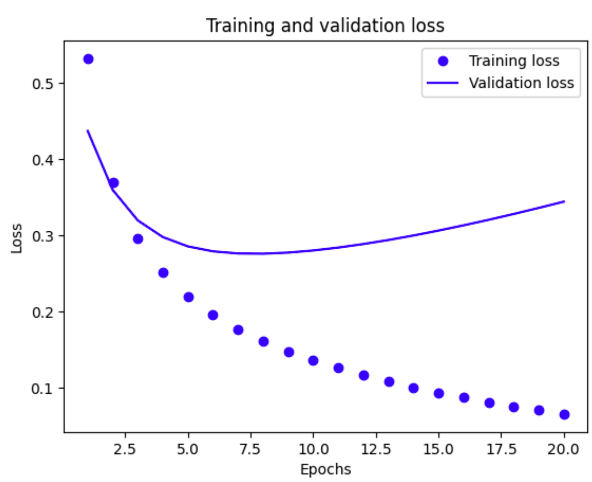


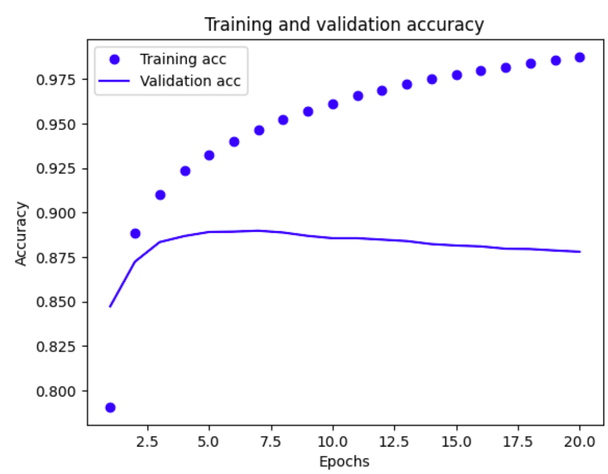
Model 14: **Model 14, employing 3 hidden layers with 32 units each, ReLU activation, binary cross-entropy loss, and dropout regularization, achieved the highest test accuracy of 88.91% after 4 epochs of training.**



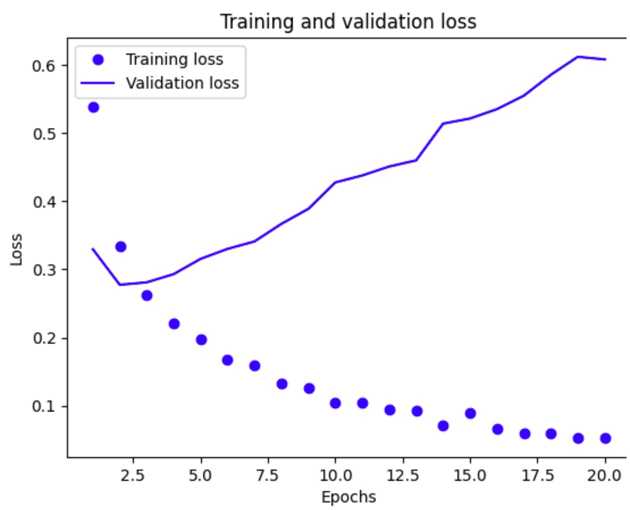


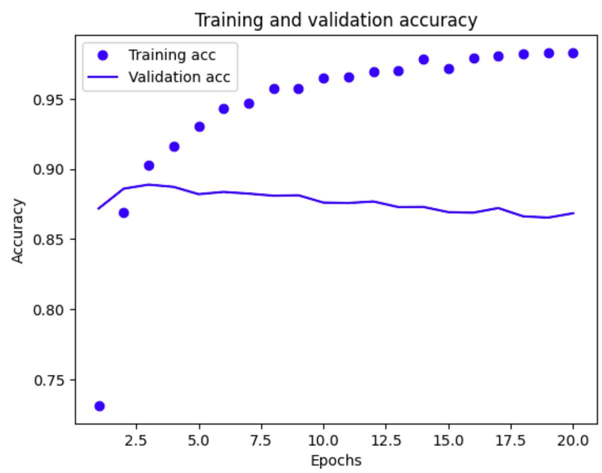
Model 15: **Model 15, featuring a single hidden layer with 8 units, ReLU activation, and binary cross-entropy loss, achieved 88.69% test accuracy after 7 epochs of training.**





Model 16: Model 16, utilizing 3 hidden layers with 32 units each, tanh activation, binary cross-entropy loss, and dropout regularization, attained 88.22% test accuracy after 3 epochs of training.





<u>Appendix:</u>

The following represents the code that was built on google colab to build the various Deep Neural Network. Each of these images depict the various configurations used with respect to hidden layers, activation function, loss values and accuracy with respect to test set, the various techniques used such as regularization and dropout methods to prevent overfitting. This is attached for perusal of various models including the base model, each depicting the model built on the test set after careful selection of epoch during model training.

Base Model:

Retraining a model from scratch

```
tf.random.set_seed(42)
np.random.seed(42)
model = keras.Sequential([
    layers.Dense(16, activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)),
    layers.Dense(16, activation="relu",kernel_initializer=keras.initializers.GlorotUniform(seed=42)),
    layers.Dense(1, activation="sigmoid", kernel_initializer=keras.initializers.GlorotUniform(seed=42))
])
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.fit(x_train, y_train, epochs=4, batch_size=512)
results = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 21ms/step - accuracy: 0.7358 - loss: 0.5557
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - accuracy: 0.8980 - loss: 0.2921
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9198 - loss: 0.2221
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9324 - loss: 0.1881
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8837 - loss: 0.2875
```

```
[ ] results
```

```
[0.2865942418575287, 0.8843600153923035]
```

Model 1:

Retraining a model from scratch

```
tf.random.set_seed(42)
np.random.seed(42)
model1 = keras.Sequential([
    layers.Dense(16, activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)),
    layers.Dense(1, activation="sigmoid", kernel_initializer=keras.initializers.GlorotUniform(seed=42))
])
model1.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model1.fit(x_train, y_train, epochs=4, batch_size=512)
results = model1.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 25ms/step - accuracy: 0.7536 - loss: 0.5359
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 16ms/step - accuracy: 0.8961 - loss: 0.3020
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9168 - loss: 0.2396
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9279 - loss: 0.2075
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8859 - loss: 0.2817
```

```
[17] results
```

```
[0.2802591621875763, 0.8878800272941589]
```

Model 2:

Retraining a model from scratch

```
[16] tf.random.set_seed(42)
     np.random.seed(42)
     model2 = keras.Sequential([
         layers.Dense(16, activation="relu", kernel_initializer=keras.initialize
         layers.Dense(16, activation="relu", kernel_initializer=keras.initialize
         layers.Dense(16, activation="relu", kernel_initializer=keras.initialize
         layers.Dense(1, activation="sigmoid", kernel_initializer=keras.initiali
     ])
     model2.compile(optimizer="rmsprop",
                    loss="binary_crossentropy",
                    metrics=["accuracy"])
     model2.fit(x_train, y_train, epochs=3, batch_size=512)
     results = model2.evaluate(x_test, y_test)
```

```
Epoch 1/3
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 30ms/step - accuracy: 0.7482 - loss: 0.5612
Epoch 2/3
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.8969 - loss: 0.2866
Epoch 3/3
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9198 - loss: 0.2140
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8837 - loss: 0.2881
```

```
[17] results
```

```
[0.2884899973869324, 0.883840024471283]
```

Model 3:

```
tf.random.set_seed(42)
np.random.seed(42)
model3 = keras.Sequential([
    layers.Dense(32, activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)),
    layers.Dense(32, activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)), #Adding another layer
    layers.Dense(32, activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)), #Adding another layer
    layers.Dense(1, activation="sigmoid", kernel_initializer=keras.initializers.GlorotUniform(seed=42))
])
model3.compile(optimizer="rmsprop",
               loss="binary_crossentropy",
               metrics=["accuracy"])
model3.fit(x_train, y_train, epochs=4, batch_size=512)
results = model3.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 21ms/step - accuracy: 0.7129 - loss: 0.5490
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - accuracy: 0.8943 - loss: 0.2781
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9186 - loss: 0.2153
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.9308 - loss: 0.1811
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8797 - loss: 0.3069
```

```
[ ] results
```

```
[0.3057752847671509, 0.8813599944114685]
```

Model 4:

Retraining a model from scratch

```
tf.random.set_seed(42)
np.random.seed(42)
model4 = keras.Sequential([
    layers.Dense(64, activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)),
    layers.Dense(64, activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)), #Adding another layer
    layers.Dense(64, activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)), #Adding another layer
    layers.Dense(1, activation="sigmoid", kernel_initializer=keras.initializers.GlorotUniform(seed=42))
])
model4.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model4.fit(x_train, y_train, epochs=4, batch_size=512)
results = model4.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 28ms/step - accuracy: 0.7012 - loss: 0.5509
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - accuracy: 0.8867 - loss: 0.2870
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9155 - loss: 0.2092
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.9332 - loss: 0.1699
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8820 - loss: 0.3041
```

```
results
```

```
[0.3014732897281647, 0.8829200267791748]
```

Model 5:

Retraining a model from scratch

```
tf.random.set_seed(42)
np.random.seed(42)
model5 = keras.Sequential([
    layers.Dense(64, activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)), #removed one hidden layer
    layers.Dense(1, activation="sigmoid", kernel_initializer=keras.initializers.GlorotUniform(seed=42))
])
model5.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model5.fit(x_train, y_train, epochs=4, batch_size=512)
results = model5.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 24ms/step - accuracy: 0.7290 - loss: 0.5292
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.8973 - loss: 0.2790
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.9155 - loss: 0.2253
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.9260 - loss: 0.2015
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8831 - loss: 0.2907
```

```
[ ] results
```

```
[0.28803542256355286, 0.8846799731254578]
```

Model 6:

Retraining a model from scratch

```
tf.random.set_seed(42)
np.random.seed(42)
model6 = keras.Sequential([
    layers.Dense(32, activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)), # removed one layer
    layers.Dense(1, activation="sigmoid", kernel_initializer=keras.initializers.GlorotUniform(seed=42))
])
model6.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model6.fit(x_train, y_train, epochs=5, batch_size=512)
results = model6.evaluate(x_test, y_test)
```

```
Epoch 1/5
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 28ms/step - accuracy: 0.7401 - loss: 0.5326
Epoch 2/5
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.8982 - loss: 0.2910
Epoch 3/5
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9172 - loss: 0.2321
Epoch 4/5
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9271 - loss: 0.2030
Epoch 5/5
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9347 - loss: 0.1831
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.8819 - loss: 0.2897
```

```
[ ] results
```

```
[0.2875302731990814, 0.8843200206756592]
```

Model 7:

Retraining a model from scratch

```
tf.random.set_seed(42)
np.random.seed(42)
model7 = keras.Sequential([
    layers.Dense(64, activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)),
    layers.Dense(32, activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)), #Adding another layer
    layers.Dense(1, activation="sigmoid", kernel_initializer=keras.initializers.GlorotUniform(seed=42))
])
model7.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model7.fit(x_train, y_train, epochs=4, batch_size=512)
results = model7.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 21ms/step - accuracy: 0.7201 - loss: 0.5384
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - accuracy: 0.8969 - loss: 0.2749
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9171 - loss: 0.2150
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9279 - loss: 0.1832
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8801 - loss: 0.2990
```

```
[21] results
```

```
[0.2969222366809845, 0.8816400170326233]
```

Model 8:

Retraining a model from scratch

```
tf.random.set_seed(42)
np.random.seed(42)
model8 = keras.Sequential([
    layers.Dense(32, activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)),
    layers.Dense(1, activation="sigmoid", kernel_initializer=keras.initializers.GlorotUniform(seed=42))
])
model8.compile(optimizer="rmsprop",
              loss="mse",             #loss funtion was changed to mse from binary_crossentropy
              metrics=["accuracy"])
model8.fit(x_train, y_train, epochs=7, batch_size=512)
results = model8.evaluate(x_test, y_test)
```

```
Epoch 1/7
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 21ms/step - accuracy: 0.7322 - loss: 0.1857
Epoch 2/7
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.8910 - loss: 0.0950
Epoch 3/7
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9090 - loss: 0.0759
Epoch 4/7
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.9199 - loss: 0.0673
Epoch 5/7
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9290 - loss: 0.0610
Epoch 6/7
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9353 - loss: 0.0561
Epoch 7/7
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9420 - loss: 0.0523
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8823 - loss: 0.0868
```

```
[ ] results
```

```
[0.08578227460384369, 0.8846399784088135]
```

Model 9:

Retraining a model from scratch

```
tf.random.set_seed(42)
np.random.seed(42)
model9 = keras.Sequential([
    layers.Dense(64, activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)),
    layers.Dense(64, activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)), #Adding another layer
    layers.Dense(32, activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)), #Adding another layer
    layers.Dense(1, activation="sigmoid", kernel_initializer=keras.initializers.GlorotUniform(seed=42))
])
model9.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model9.fit(x_train, y_train, epochs=4, batch_size=512)
results = model9.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 27ms/step - accuracy: 0.7041 - loss: 0.5605
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 15ms/step - accuracy: 0.8907 - loss: 0.2771
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.9197 - loss: 0.2125
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.9268 - loss: 0.1810
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8829 - loss: 0.2985
```

```
results
```

```
[0.2837504744529724, 0.8852400183677673]
```

Model 10:

Retraining a model from scratch

```
[ ] tf.random.set_seed(42)
    np.random.seed(42)
    model10 = keras.Sequential([
        layers.Dense(32, activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)),
        layers.Dense(32, activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)),
        layers.Dense(1, activation="sigmoid", kernel_initializer=keras.initializers.GlorotUniform(seed=42))
    ])
    model10.compile(optimizer="rmsprop",
                  loss="mse",
                  metrics=["accuracy"])
    model10.fit(x_train, y_train, epochs=5, batch_size=512)
    results = model10.evaluate(x_test, y_test)
```

```
Epoch 1/5
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 22ms/step - accuracy: 0.7283 - loss: 0.1869
Epoch 2/5
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.8897 - loss: 0.0894
Epoch 3/5
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9128 - loss: 0.0688
Epoch 4/5
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.9297 - loss: 0.0574
Epoch 5/5
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - accuracy: 0.9382 - loss: 0.0511
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8808 - loss: 0.0867
```

```
[ ] results
```

```
[0.08627662807703018, 0.88264000415802]
```

Model 11:

Retraining a model from scratch

```
tf.random.set_seed(42)
np.random.seed(42)
model11 = keras.Sequential([
    layers.Dense(32, activation="tanh", kernel_initializer=keras.initializers.GlorotUniform(seed=42)),
    layers.Dense(32, activation="tanh", kernel_initializer=keras.initializers.GlorotUniform(seed=42)),
    layers.Dense(1, activation="sigmoid", kernel_initializer=keras.initializers.GlorotUniform(seed=42))
])
model11.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model11.fit(x_train, y_train, epochs=2, batch_size=512)
results = model11.evaluate(x_test, y_test)
```

```
Epoch 1/2
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 21ms/step - accuracy: 0.7357 - loss: 0.5301
Epoch 2/2
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9012 - loss: 0.2541
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.8847 - loss: 0.2830
```

```
[ ] results
```

```
[0.28107792139053345, 0.8861600160598755]
```

Model 12:

Retraining a model from scratch

```
tf.random.set_seed(42)
np.random.seed(42)
model12 = keras.Sequential([
    layers.Dense(64, kernel_regularizer=regularizers.l2(0.001), activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)),
    layers.Dense(64, kernel_regularizer=regularizers.l2(0.001), activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)),
    #Adding another layer
    layers.Dense(64, kernel_regularizer=regularizers.l2(0.001), activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)),
    #Adding another layer
    layers.Dense(1, activation="sigmoid", kernel_initializer=keras.initializers.GlorotUniform(seed=42))
])
model12.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model12.fit(x_train, y_train, epochs=3, batch_size=512)
results = model12.evaluate(x_test, y_test)
```

```
Epoch 1/3
49/49 ━━━━━━━━━━━━━━━━━━━━ 4s 41ms/step - accuracy: 0.7028 - loss: 0.7862
Epoch 2/3
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 12ms/step - accuracy: 0.8797 - loss: 0.4743
Epoch 3/3
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.9158 - loss: 0.3703
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8846 - loss: 0.4167
```

```
[18] results
```

```
[0.4145466387271881, 0.8860399723052979]
```

Model 13:

Retraining a model from scratch

```
tf.random.set_seed(42)
np.random.seed(42)
model13 = keras.Sequential([
    layers.Dense(16, kernel_regularizer=regularizers.l1(0.0001), activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)),
    #l1 norm regulariztion at 0.0001
    layers.Dense(16, kernel_regularizer=regularizers.l1(0.0001), activation="relu",kernel_initializer=keras.initializers.GlorotUniform(seed=42)),
    #l1 norm regulariztion at 0.0001
    layers.Dense(1,  activation="sigmoid", kernel_initializer=keras.initializers.GlorotUniform(seed=42))
])
model13.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model13.fit(x_train, y_train, epochs=4, batch_size=512)
results = model13.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 5s 30ms/step - accuracy: 0.7320 - loss: 0.7256
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - accuracy: 0.8915 - loss: 0.3987
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9057 - loss: 0.3310
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9122 - loss: 0.3100
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8839 - loss: 0.3598
```

```
[ ] results
```

```
[0.3573884665966034, 0.8861600160598755]
```

Model 14:

Retraining a model from scratch

```
tf.random.set_seed(42)
np.random.seed(42)
model14 = keras.Sequential([
    layers.Dense(32, activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)),
    layers.Dropout(0.5),                                                                                  #Dropout
    layers.Dense(32, activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)),
    layers.Dropout(0.5),                                                                                  #Dropout
    layers.Dense(32, activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)),
    layers.Dropout(0.5),                                                                                  #Dropout
    layers.Dense(1, activation="sigmoid", kernel_initializer=keras.initializers.GlorotUniform(seed=42))
])
model14.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model14.fit(x_train, y_train, epochs=4, batch_size=512)
results = model14.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 6s 43ms/step - accuracy: 0.5973 - loss: 0.6491
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 13ms/step - accuracy: 0.8398 - loss: 0.4110
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.8880 - loss: 0.3126
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.9114 - loss: 0.2506
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8906 - loss: 0.2876
```

```
[ ] results
```

```
[0.2882605195045471, 0.8891599774360657]
```

Model 15:

Retraining a model from scratch

```
tf.random.set_seed(42)
np.random.seed(42)
model15 = keras.Sequential([
    layers.Dense(8, activation="relu", kernel_initializer=keras.initializers.GlorotUniform(seed=42)),
    layers.Dense(1, activation="sigmoid", kernel_initializer=keras.initializers.GlorotUniform(seed=42))
])
model15.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model15.fit(x_train, y_train, epochs=7, batch_size=512)
results = model15.evaluate(x_test, y_test)
```

```
Epoch 1/7
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 20ms/step - accuracy: 0.7536 - loss: 0.5588
Epoch 2/7
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - accuracy: 0.8944 - loss: 0.3355
Epoch 3/7
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9124 - loss: 0.2652
Epoch 4/7
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.9217 - loss: 0.2277
Epoch 5/7
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9296 - loss: 0.2035
Epoch 6/7
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9367 - loss: 0.1857
Epoch 7/7
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9426 - loss: 0.1717
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8848 - loss: 0.2833
```

```
[ ] results
```

```
[0.28220608830451965, 0.8869600296020508]
```

Model 16:

Retraining a model from scratch

```
tf.random.set_seed(42)
np.random.seed(42)
model15 = keras.Sequential([
    layers.Dense(32, activation="tanh", kernel_initializer=keras.initializers.GlorotUniform(seed=42)), #Activation funtion changed from relu to tanh
                                                                                                       #Dropout
    layers.Dropout(0.5),
    layers.Dense(32, activation="tanh", kernel_initializer=keras.initializers.GlorotUniform(seed=42)),
    layers.Dropout(0.5), #Adding another layer
    layers.Dense(32, activation="tanh", kernel_initializer=keras.initializers.GlorotUniform(seed=42)), #Adding another layer
    layers.Dropout(0.5),
    layers.Dense(1, activation="sigmoid", kernel_initializer=keras.initializers.GlorotUniform(seed=42))
])
model15.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model15.fit(x_train, y_train, epochs=3, batch_size=512)
results = model15.evaluate(x_test, y_test)
```

```
Epoch 1/3
49/49 ━━━━━━━━━━━━━━━━━━━━ 6s 43ms/step - accuracy: 0.6862 - loss: 0.5816
Epoch 2/3
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.8821 - loss: 0.3048
Epoch 3/3
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.9086 - loss: 0.2486
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8804 - loss: 0.3107
```

```
[ ] results
```

```
[0.3126238286495209, 0.8822399973869324]
```