P1 (40pt): In the following code example explained in Lecture 6, https://colab.research.google.com/drive/13cof4XUULbUqO0s5h-cd17FskWjpyMkm,

make the following changes to the neural network model sequentially in the example:

1. Change the number of neurons on the hidden layer to 256 units. **(10pt)**
2. Use the tanh activation (an activation that was popular in the early days of neural networks) instead of relu for the hidden layer. **(10pt)**
3. Add an additional hidden layer with 256 units and tanh activation function. **(10pt)**

Retrain the newly defined model and evaluate the trained model on the testing dataset to get the accuracy. **(10pt)**

```
#Code
from tensorflow.keras import models
from tensorflow.keras import layers

network = models.Sequential()
network.add(layers.Dense(256, activation='tanh', input_shape=(28 * 28,)))
network.add(layers.Dense(256, activation= 'tanh'))
network.add(layers.Dense(10, activation='softmax'))
```

P2 (60pt): Write a Python code in Colab using NumPy, Panda, Scikit-Learn and Keras to complete the following tasks:

1. Import the Auto MPG dataset using pandas.read_csv(), use the attribute names as explained in the dataset description as the column names, view the strings '?' as the missing value, and whitespace (i.e., '\s+') as the column delimiter. Print out the shape and first 5 rows of the DataFrame. **(5pt)**

   a. Dataset source file: http://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data

   b. Dataset description: http://archive.ics.uci.edu/ml/datasets/Auto+MPG

```
import matplotlib.pyplot as plt
import pandas as pd
import tensorflow as tf
from tensorflow import keras

import numpy as np
np.random.seed(100)

tf.random.set_seed(100)
```

```
# put your answer here
url = 'http://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data'

df = pd.read_csv(url, index_col = False, names = ['mpg','cylinders','displacement','horsepower','weight', 'acceleration', 'model year', 'origin', 'car name'],
                 header = None, na_values = '?', sep = '\s+')
df.head(5)
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130.0	3504.0	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693.0	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150.0	3436.0	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150.0	3433.0	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140.0	3449.0	10.5	70	1	ford torino

2. Delete the “car_name” column using .drop() and drop the rows containing NULL value using .dropna(). Print out the shape of the DataFrame. **(5pt)**

```
# put your answer here
df = df.drop(columns = 'car name')
df.dropna(inplace = True)
```

```
df.shape
```

```
(392, 8)
```

3. For the ‘origin’ column with categorical attribute, replace it with the columns with numerical attributes using one-hot encoding. Print out the shape and first 5 rows of the new DataFrame. **(5pt)**

```
# put your answer here
df = pd.get_dummies(data= df, columns=['origin'])
```

```
df.shape
```

```
(392, 10)
```

```
df.head(5)
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin_1	origin_2	origin_3
0	18.0	8	307.0	130.0	3504.0	12.0	70	1	0	0
1	15.0	8	350.0	165.0	3693.0	11.5	70	1	0	0
2	18.0	8	318.0	150.0	3436.0	11.0	70	1	0	0

4. Separate the “mpg” column from other columns and view it as the label vector and others as the feature matrix. Split the data into a training set (80%) and testing set (20%) using train_test_split and print out their shapes. Print out the statistics of your training feature matrix using .describe(). **(5pt)**

```
from sklearn.model_selection import train_test_split
# put your answer here
X = df.drop('mpg', axis = 1)
y = df['mpg']

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 50, train_size = 0.8)


#y_train = np.asarray(y_train).astype('float32') /255
#y_test = np.asarray(y_test).astype('float32') / 255


X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((313, 9), (79, 9), (313,), (79,))
```

```
X_train.describe()
```

	cylinders	displacement	horsepower	weight	acceleration	model year	origin_1	origin_2	or
count	313.000000	313.000000	313.000000	313.000000	313.000000	313.000000	313.000000	313.000000	313.(
mean	5.389776	190.154952	102.869010	2942.322684	15.697125	76.143770	0.603834	0.178914	0.2
std	1.689378	105.882581	39.110856	861.779719	2.780015	3.707464	0.489883	0.383894	0.4
min	3.000000	68.000000	46.000000	1613.000000	8.500000	70.000000	0.000000	0.000000	0.(
25%	4.000000	98.000000	75.000000	2210.000000	13.900000	73.000000	0.000000	0.000000	0.(
50%	4.000000	140.000000	90.000000	2735.000000	15.500000	76.000000	1.000000	0.000000	0.(
75%	6.000000	258.000000	115.000000	3563.000000	17.500000	79.000000	1.000000	0.000000	0.(
max	8.000000	455.000000	225.000000	5140.000000	24.600000	82.000000	1.000000	1.000000	1.(

```
X_test.describe()
```

	cylinders	displacement	horsepower	weight	acceleration	model year	origin_1	origin_2	origin_
count	79.000000	79.000000	79.000000	79.000000	79.000000	79.000000	79.000000	79.000000	79.00000
mean	5.797468	211.278481	110.810127	3117.291139	14.924051	75.329114	0.708861	0.151899	0.13924
std	1.742234	98.422029	35.457116	788.199541	2.599419	3.536337	0.457190	0.361216	0.3484(
min	4.000000	86.000000	53.000000	1795.000000	8.000000	70.000000	0.000000	0.000000	0.00000
25%	4.000000	119.500000	87.000000	2498.000000	13.500000	73.000000	0.000000	0.000000	0.00000
50%	6.000000	198.000000	98.000000	3039.000000	14.900000	75.000000	1.000000	0.000000	0.00000
75%	8.000000	304.500000	145.000000	3756.000000	16.300000	78.000000	1.000000	0.000000	0.00000

```
y_train.describe()
```

```
count    313.000000
mean      23.873482
std        8.070945
min        9.000000
25%       17.600000
50%       23.000000
75%       30.000000
max       46.600000
Name: mpg, dtype: float64
```

```
y_test.describe()
```

```
count    79.000000
mean     21.751899
std       6.416462
min      13.000000
25%      16.000000
50%      20.500000
75%      26.500000
max      38.000000
Name: mpg, dtype: float64
```

5. Normalize the feature columns in both training and testing datasets so that their means equal to zero and variances equal to one. Note that the testing set can only be scaled by the mean and standard deviation values obtained from the training set. Describe the statistics of your normalized feature matrix of training dataset using .describe() in Pandas. **(5pt)**

- Option 1: You can follow the normalization steps in the code example of “Predicting house prices: a regression example” in Lecture 6.
- Option 2: You can use StandardScaler() in Scikit-Learn as in Homework 2 but you may need to transform a NumPy array back to Pandas DataFrame using pd.DataFrame() before calling .describe().

```
from sklearn.preprocessing import StandardScaler
feature_scaler = StandardScaler()
```

```
X_train = feature_scaler.fit_transform(X_train)
X_test = feature_scaler.transform(X_test)
```

```
X_train = pd.DataFrame(X_train)
X_train.describe()
```

	0	1	2	3	4	5	6	
count	3.130000e+02	3.130000e+02	3.130000e+02	3.130000e+02	3.130000e+02	3.130000e+02	3.130000e+02	3.1
mean	-2.497115e-16	-4.540209e-17	-7.377840e-17	1.589073e-16	9.080418e-17	2.582244e-16	5.107735e-17	6.8
std	1.001601e+00	1.001601e+00	1.001601e+00	1.001601e+00	1.001601e+00	1.001601e+00	1.001601e+00	1.0
min	-1.416855e+00	-1.155531e+00	-1.456375e+00	-1.545002e+00	-2.593025e+00	-1.659789e+00	-1.234582e+00	
25%	-8.239730e-01	-8.717441e-01	-7.137056e-01	-8.511402e-01	-6.474793e-01	-8.493148e-01	-1.234582e+00	
50%	-8.239730e-01	-4.744431e-01	-3.295662e-01	-2.409603e-01	-7.102128e-02	-3.884062e-02	8.099905e-01	
75%	3.617904e-01	6.417834e-01	3.106661e-01	7.213806e-01	6.495512e-01	7.716336e-01	8.099905e-01	
max	1.547554e+00	2.505314e+00	3.127688e+00	2.554245e+00	3.207584e+00	1.582108e+00	8.099905e-01	2.1

```
X_test = pd.DataFrame(X_test)
X_test.describe()
```

	0	1	2	3	4	5	6	7	8
count	79.000000	79.000000	79.000000	79.000000	79.000000	79.000000	79.000000	79.000000	79.000000
mean	0.241713	0.199819	0.203366	0.203357	-0.278528	-0.220086	0.214735	-0.070484	-0.189177
std	1.032938	0.931028	0.908032	0.916083	0.936535	0.955370	0.934759	0.942434	0.844883
min	-0.823973	-0.985259	-1.277110	-1.333473	-2.773168	-1.659789	-1.234582	-0.466796	-0.526831
25%	-0.823973	-0.668364	-0.406394	-0.516413	-0.791594	-0.849315	-1.234582	-0.466796	-0.526831
50%	0.361790	0.074211	-0.124692	0.112363	-0.287193	-0.308999	0.809991	-0.466796	-0.526831
75%	1.547554	1.081652	1.078945	0.945694	0.217208	0.501475	0.809991	-0.466796	-0.526831
max	1.547554	1.985039	3.255734	2.041694	3.279641	1.582108	0.809991	2.142262	1.898142

6. Build a sequential neural network model in Keras with two densely connected hidden layers (32 neurons and ReLU activation function for each hidden layer), and an output layer that returns a single, continuous value. Print out the model summary using .summary(). **(10pt)**

- Hint: You can follow the “Classifying movie reviews” example in Lecture 6, but need to change input_shape and last layer activation function correctly in the model definition.

```
# put your answer here
```

```
from tensorflow.keras import models
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = models.Sequential()
model.add(layers.Dense(32, activation = 'relu', input_shape = (X_train.shape[1],)))
model.add(layers.Dense(32, activation = 'relu'))
model.add(layers.Dense(1))
```

```
model.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_30 (Dense)            (None, 32)                320       

 dense_31 (Dense)            (None, 32)                1056      

 dense_32 (Dense)            (None, 1)                 33        

=================================================================
Total params: 1,409
Trainable params: 1,409
Non-trainable params: 0
_________________________________________________________________
```

7. Define the appropriate loss function, optimizer, and metrics for this specific problem and compile the NN model. **(10pt)**

```
from tensorflow.python import metrics
# put your answer here

model.compile(optimizer='rmsprop',
              loss='mse',
              metrics=['mae'])
```

8. Put aside 20% of the normalized training data as the validation dataset by setting validation_split = 0.2 and set verbose = 0 to compress the model training status in Keras .fit(). Train the NN model for 100 epochs and batch size of 32 and plot the training and validation loss progress with respect to the epoch number. **(10pt)**

- Remember to use GPU for training in Colab. Otherwise, you may find out of memory error or slow execution.
- There is no need to do K-fold cross-validation for this step.

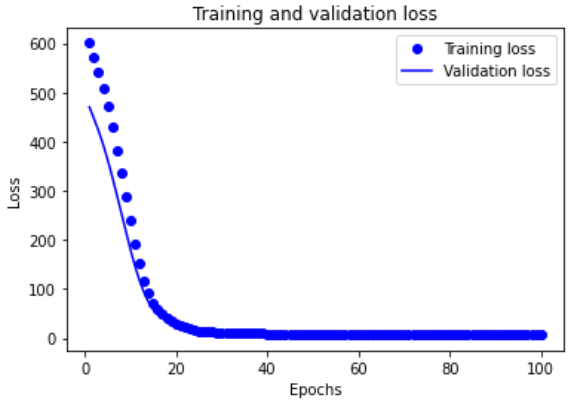
```
# put your answer here
```

```
history = model.fit(X_train,
                    y_train,
                    verbose = 0,
                    validation_split = 0.2,
                    epochs = 100,
                    batch_size = 32,
                    validation_data = (X_test, y_test))
```

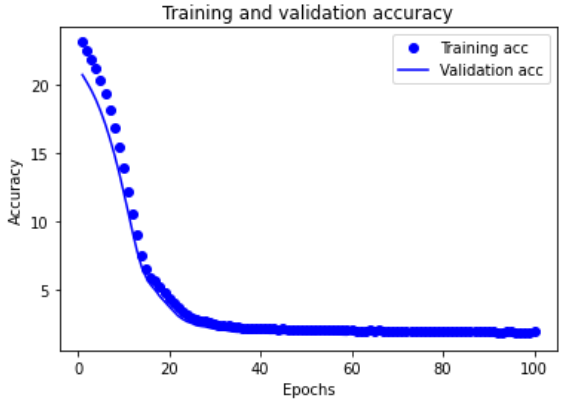
```
history_dict = history.history
history_dict.keys()
```

```
dict_keys(['loss', 'mae', 'val_loss', 'val_mae'])
```

```
import matplotlib.pyplot as plt

acc = history.history['mae']
val_acc = history.history['val_mae']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

# "bo" is for "blue dot"
plt.plot(epochs, loss, 'bo', label='Training loss')
# b is for "solid blue line"
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```



```
plt.clf()   # clear figure
acc_values = history_dict['mae']
val_acc_values = history_dict['val_mae']

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```



9. Use the trained NN model to make predictions on the normalized testing dataset and observe the prediction error. **(5pt)**

```
# put your answer here
results = model.evaluate(X_test, y_test)
```

```
3/3 [==============================] - 0s 4ms/step - loss: 5.2537 - mae: 1.7623
```

```
model.predict(X_test)
```

```
3/3 [==============================] - 0s 3ms/step
array([[24.589079 ],
       [16.694475 ],
       [24.2933   ],
       [15.266783 ],
       [29.239483 ],
       [16.75126  ],
       [17.603123 ],
       [22.764296 ],
       [17.246798 ],
       [18.43931  ],
       [29.572968 ],
       [26.006418 ],
```

```
[14.913834 ],
[17.828634 ],
[13.283282 ],
[16.097935 ],
[31.184515 ],
[29.614803 ],
[24.202991 ],
[36.935925 ],
[28.80614  ],
[37.829014 ],
[13.01804  ],
[17.191717 ],
[20.474588 ],
[13.288854 ],
[18.732338 ],
[20.802275 ],
[31.024765 ],
[14.398551 ],
[12.658205 ],
[30.96376  ],
[31.389847 ],
[23.019953 ],
[12.7838545],
[21.551588 ],
[19.825485 ],
[14.768458 ],
[15.045668 ],
[15.316504 ],
[20.400635 ],
[18.13057  ],
[33.19419  ],
[18.027773 ],
[13.987837 ],
[20.653301 ],
[17.24744  ],
[14.4989195],
[20.54758  ],
[38.29189  ],
[23.690294 ],
[34.929123 ],
[26.811886 ],
[14.154148 ],
[18.133314 ],
[22.431322 ],
[24.867506 ],
```

Colab paid products - Cancel contracts here

✓ 0s completed at 6:04 PM