HW4

Node Functions:
In the code, we used the ReLU (Rectified Linear Unit) activation function for the hidden layer with activation='relu'. ReLU is a common choice for hidden layers as it introduces non-linearity by outputting zero for negative values and the input for positive values.

Data Normalization:
In the code, the StandardScaler from scikit-learn is used to normalize the input features (citations) with X_normalized = scaler.fit_transform(X).

Output Interpretation:
In this regression task, we are predicting a numeric value (2022 citations), so the output interpretation is straightforward. The model aims to predict the numeric value as close as possible to the actual 2022 citations.
The activation function for the output layer is set to linear, which means the output directly represents the predicted value.

Optimizer Choices:
The optimizer is a crucial component of training neural networks. It determines how the model's weights are updated during training to minimize the chosen loss function.
In the code, we've chosen Stochastic Gradient Descent (SGD) as the optimizer with a learning rate of 0.01. This optimizer updates the weights based on the gradients of the loss with respect to the weights. We've also tested with Adam as the optimizer.
The loss function chosen for this regression task is Mean Absolute Error ('mean_absolute_error'), which is a common choice for regression problems.

Below are the output screenshots while trying different approaches to get the best results:

Learning Rate as 0.001 – MAE = 374.7608

```
model.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=0.001),  loss='mean_absolute_error', metrics=['mae'])

# Train the model
model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_test, y_test))
```

HW4

```
Epoch 1/100
3/3 [==============================] - 0s 65ms/step - loss: 655.3097 - mae: 655.3097 - val_loss: 375.2338 - val_mae: 375.2338
Epoch 2/100
3/3 [==============================] - 0s 22ms/step - loss: 655.3035 - mae: 655.3035 - val_loss: 375.2293 - val_mae: 375.2293
Epoch 3/100
3/3 [==============================] - 0s 24ms/step - loss: 655.2977 - mae: 655.2977 - val_loss: 375.2249 - val_mae: 375.2249
Epoch 4/100
3/3 [==============================] - 0s 20ms/step - loss: 655.2920 - mae: 655.2920 - val_loss: 375.2204 - val_mae: 375.2204
Epoch 5/100
3/3 [==============================] - 0s 22ms/step - loss: 655.2863 - mae: 655.2863 - val_loss: 375.2159 - val_mae: 375.2159
Epoch 6/100
3/3 [==============================] - 0s 23ms/step - loss: 655.2806 - mae: 655.2806 - val_loss: 375.2115 - val_mae: 375.2115
Epoch 7/100
3/3 [==============================] - 0s 20ms/step - loss: 655.2751 - mae: 655.2751 - val_loss: 375.2067 - val_mae: 375.2067
Epoch 8/100
3/3 [==============================] - 0s 20ms/step - loss: 655.2684 - mae: 655.2684 - val_loss: 375.2021 - val_mae: 375.2021
Epoch 9/100
3/3 [==============================] - 0s 19ms/step - loss: 655.2628 - mae: 655.2628 - val_loss: 375.1976 - val_mae: 375.1976
Epoch 10/100
3/3 [==============================] - 0s 20ms/step - loss: 655.2572 - mae: 655.2572 - val_loss: 375.1931 - val_mae: 375.1931
Epoch 11/100
3/3 [==============================] - 0s 17ms/step - loss: 655.2513 - mae: 655.2513 - val_loss: 375.1886 - val_mae: 375.1886
Epoch 12/100
3/3 [==============================] - 0s 20ms/step - loss: 655.2457 - mae: 655.2457 - val_loss: 375.1840 - val_mae: 375.1840
Epoch 13/100
...
Epoch 100/100
3/3 [==============================] - 0s 20ms/step - loss: 654.6969 - mae: 654.6969 - val_loss: 374.7609 - val_mae: 374.7609
1/1 [==============================] - 0s 53ms/step
Mean Absolute Error on the Test Data: 374.7608947753906
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Learning rate as 0.01- MAE = 372.2100

```
# Compile the model
model.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=0.01),  loss='mean_
absolute_error', metrics=['mae'])

# Train the model
model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_test,
y_test))
```

HW4

```
Epoch 1/100
3/3 [==============================] - 0s 76ms/step - loss: 655.7902 - mae: 655.7902 - val_loss: 375.3941 - val_mae: 375.3941
Epoch 2/100
3/3 [==============================] - 0s 16ms/step - loss: 655.7517 - mae: 655.7517 - val_loss: 375.3592 - val_mae: 375.3592
Epoch 3/100
3/3 [==============================] - 0s 16ms/step - loss: 655.7137 - mae: 655.7137 - val_loss: 375.3249 - val_mae: 375.3249
Epoch 4/100
3/3 [==============================] - 0s 20ms/step - loss: 655.6766 - mae: 655.6766 - val_loss: 375.2909 - val_mae: 375.2909
Epoch 5/100
3/3 [==============================] - 0s 24ms/step - loss: 655.6403 - mae: 655.6403 - val_loss: 375.2573 - val_mae: 375.2573
Epoch 6/100
3/3 [==============================] - 0s 24ms/step - loss: 655.6040 - mae: 655.6040 - val_loss: 375.2237 - val_mae: 375.2237
Epoch 7/100
3/3 [==============================] - 0s 20ms/step - loss: 655.5681 - mae: 655.5681 - val_loss: 375.1905 - val_mae: 375.1905
Epoch 8/100
3/3 [==============================] - 0s 20ms/step - loss: 655.5319 - mae: 655.5319 - val_loss: 375.1567 - val_mae: 375.1567
Epoch 9/100
3/3 [==============================] - 0s 20ms/step - loss: 655.4957 - mae: 655.4957 - val_loss: 375.1237 - val_mae: 375.1237
Epoch 10/100
3/3 [==============================] - 0s 20ms/step - loss: 655.4605 - mae: 655.4605 - val_loss: 375.0910 - val_mae: 375.0910
Epoch 11/100
3/3 [==============================] - 0s 20ms/step - loss: 655.4261 - mae: 655.4261 - val_loss: 375.0586 - val_mae: 375.0586
Epoch 12/100
3/3 [==============================] - 0s 28ms/step - loss: 655.3917 - mae: 655.3917 - val_loss: 375.0260 - val_mae: 375.0260
Epoch 13/100
...
Epoch 100/100
3/3 [==============================] - 0s 24ms/step - loss: 652.3836 - mae: 652.3836 - val_loss: 372.2100 - val_mae: 372.2100
1/1 [==============================] - 0s 56ms/step
Mean Absolute Error on the Test Data: 372.21002197265625
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Learning Rate 0.1 - MAE = 50.66

Till this point this is the best configuration as it has the minimum error value. Also this is better than HW3 approaches. The best value we got in HW3 was 66.2

```
# Compile the model
model.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=0.1),  loss='mean_a
bsolute_error', metrics=['mae'])

# Train the model
model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_test,
y_test))
```

HW4

```
...  Epoch 1/100
     3/3 [==============================] - 1s 68ms/step - loss: 655.3660 - mae: 655.3660 - val_loss: 374.7382 - val_mae: 374.7382
     Epoch 2/100
     3/3 [==============================] - 0s 24ms/step - loss: 654.8558 - mae: 654.8558 - val_loss: 374.1905 - val_mae: 374.1905
     Epoch 3/100
     3/3 [==============================] - 0s 20ms/step - loss: 654.2409 - mae: 654.2409 - val_loss: 373.4483 - val_mae: 373.4483
     Epoch 4/100
     3/3 [==============================] - 0s 23ms/step - loss: 653.4294 - mae: 653.4294 - val_loss: 372.3853 - val_mae: 372.3853
     Epoch 5/100
     3/3 [==============================] - 0s 21ms/step - loss: 652.2924 - mae: 652.2924 - val_loss: 370.8205 - val_mae: 370.8205
     Epoch 6/100
     3/3 [==============================] - 0s 21ms/step - loss: 650.6823 - mae: 650.6823 - val_loss: 368.4176 - val_mae: 368.4176
     Epoch 7/100
     3/3 [==============================] - 0s 21ms/step - loss: 648.1846 - mae: 648.1846 - val_loss: 364.5019 - val_mae: 364.5019
     Epoch 8/100
     3/3 [==============================] - 0s 20ms/step - loss: 644.1442 - mae: 644.1442 - val_loss: 358.3681 - val_mae: 358.3681
     Epoch 9/100
     3/3 [==============================] - 0s 24ms/step - loss: 637.8605 - mae: 637.8605 - val_loss: 348.5956 - val_mae: 348.5956
     Epoch 10/100
     3/3 [==============================] - 0s 27ms/step - loss: 628.7740 - mae: 628.7740 - val_loss: 335.1234 - val_mae: 335.1234
     Epoch 11/100
     3/3 [==============================] - 0s 29ms/step - loss: 617.2496 - mae: 617.2496 - val_loss: 321.6957 - val_mae: 321.6957
     Epoch 12/100
     3/3 [==============================] - 0s 20ms/step - loss: 602.6212 - mae: 602.6212 - val_loss: 304.9284 - val_mae: 304.9284
     Epoch 13/100
     ...
     Epoch 100/100
     3/3 [==============================] - 0s 17ms/step - loss: 70.8724 - mae: 70.8724 - val_loss: 50.6612 - val_mae: 50.6612
     1/1 [==============================] - 0s 48ms/step
     Mean Absolute Error on the Test Data: 50.661170959472656
     Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Changing the optimizer to Adam and keeping the other configurations same as before
MAE = 82.4599

```
# Compile the model
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.1),  loss='mean_
absolute_error', metrics=['mae'])

# Train the model
model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_test,
y_test))
```

HW4

```
...  Epoch 1/100
     3/3 [==============================] - 1s 81ms/step - loss: 655.5745 - mae: 655.5745 - val_loss: 374.7823 - val_mae: 374.7823
     Epoch 2/100
     3/3 [==============================] - 0s 20ms/step - loss: 654.7521 - mae: 654.7521 - val_loss: 374.0492 - val_mae: 374.0492
     Epoch 3/100
     3/3 [==============================] - 0s 16ms/step - loss: 653.8057 - mae: 653.8057 - val_loss: 373.0440 - val_mae: 373.0440
     Epoch 4/100
     3/3 [==============================] - 0s 24ms/step - loss: 652.4332 - mae: 652.4332 - val_loss: 371.8514 - val_mae: 371.8514
     Epoch 5/100
     3/3 [==============================] - 0s 21ms/step - loss: 650.7925 - mae: 650.7925 - val_loss: 370.4333 - val_mae: 370.4333
     Epoch 6/100
     3/3 [==============================] - 0s 23ms/step - loss: 648.5540 - mae: 648.5540 - val_loss: 368.6756 - val_mae: 368.6756
     Epoch 7/100
     3/3 [==============================] - 0s 24ms/step - loss: 646.1489 - mae: 646.1489 - val_loss: 366.6850 - val_mae: 366.6850
     Epoch 8/100
     3/3 [==============================] - 0s 24ms/step - loss: 642.5319 - mae: 642.5319 - val_loss: 364.2446 - val_mae: 364.2446
     Epoch 9/100
     3/3 [==============================] - 0s 20ms/step - loss: 638.4163 - mae: 638.4163 - val_loss: 361.4142 - val_mae: 361.4142
     Epoch 10/100
     3/3 [==============================] - 0s 16ms/step - loss: 633.4517 - mae: 633.4517 - val_loss: 358.2069 - val_mae: 358.2069
     Epoch 11/100
     3/3 [==============================] - 0s 21ms/step - loss: 628.2601 - mae: 628.2601 - val_loss: 354.6049 - val_mae: 354.6049
     Epoch 12/100
     3/3 [==============================] - 0s 24ms/step - loss: 622.1798 - mae: 622.1798 - val_loss: 350.5348 - val_mae: 350.5348
     Epoch 13/100
     ...
     Epoch 100/100
     3/3 [==============================] - 0s 24ms/step - loss: 94.9666 - mae: 94.9666 - val_loss: 82.4600 - val_mae: 82.4600
     1/1 [==============================] - 0s 40ms/step
     Mean Absolute Error on the Test Data: 82.45997619628906
```

Changing optimizer back to SGD as Adam optimizer didn't give better results than SGD.
Changing epoch value to 1000- MAE = 59.2657

```
# Compile the model
model.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=0.1),  loss='mean_a
bsolute_error', metrics=['mae'])

# Train the model
model.fit(X_train, y_train, epochs=1000, batch_size=32, validation_data=(X_test,
y_test))
```

HW4

```
...  Epoch 1/1000
     3/3 [==============================] - 0s 82ms/step - loss: 655.7281 - mae: 655.7281 - val_loss: 374.9934 - val_mae: 374.9934
     Epoch 2/1000
     3/3 [==============================] - 0s 29ms/step - loss: 655.1717 - mae: 655.1717 - val_loss: 374.4603 - val_mae: 374.4603
     Epoch 3/1000
     3/3 [==============================] - 0s 61ms/step - loss: 654.6402 - mae: 654.6402 - val_loss: 373.8889 - val_mae: 373.8889
     Epoch 4/1000
     3/3 [==============================] - 0s 25ms/step - loss: 654.0497 - mae: 654.0497 - val_loss: 373.1460 - val_mae: 373.1460
     Epoch 5/1000
     3/3 [==============================] - 0s 29ms/step - loss: 653.2870 - mae: 653.2870 - val_loss: 372.1059 - val_mae: 372.1059
     Epoch 6/1000
     3/3 [==============================] - 0s 20ms/step - loss: 652.2032 - mae: 652.2032 - val_loss: 370.5202 - val_mae: 370.5202
     Epoch 7/1000
     3/3 [==============================] - 0s 16ms/step - loss: 650.5985 - mae: 650.5985 - val_loss: 368.0124 - val_mae: 368.0124
     Epoch 8/1000
     3/3 [==============================] - 0s 19ms/step - loss: 648.1227 - mae: 648.1227 - val_loss: 364.0886 - val_mae: 364.0886
     Epoch 9/1000
     3/3 [==============================] - 0s 20ms/step - loss: 644.2202 - mae: 644.2202 - val_loss: 357.8642 - val_mae: 357.8642
     Epoch 10/1000
     3/3 [==============================] - 0s 25ms/step - loss: 637.9470 - mae: 637.9470 - val_loss: 347.3245 - val_mae: 347.3245
     Epoch 11/1000
     3/3 [==============================] - 0s 20ms/step - loss: 627.7758 - mae: 627.7758 - val_loss: 331.8119 - val_mae: 331.8119
     Epoch 12/1000
     3/3 [==============================] - 0s 22ms/step - loss: 614.6639 - mae: 614.6639 - val_loss: 315.5286 - val_mae: 315.5286
     Epoch 13/1000
     ...
     Epoch 1000/1000
     3/3 [==============================] - 0s 20ms/step - loss: 62.9983 - mae: 62.9983 - val_loss: 59.2658 - val_mae: 59.2658
     1/1 [==============================] - 0s 40ms/step
     Mean Absolute Error on the Test Data: 59.26579666137695
     Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Best case:

Changing learning rate to 0.01 with epoch as 1000 and SGD optimizer.

MAE = 41.4748

This is the best result we got among all the configurations.

```
# Compile the model
model.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=0.01),  loss='mean_
absolute_error', metrics=['mae'])

# Train the model
model.fit(X_train, y_train, epochs=1000, batch_size=32, validation_data=(X_test,
y_test))
```

HW4

```
...  Epoch 1/1000
     3/3 [==============================] - 0s 76ms/step - loss: 654.6774 - mae: 654.6774 - val_loss: 374.7103 - val_mae: 374.7103
     Epoch 2/1000
     3/3 [==============================] - 0s 20ms/step - loss: 654.6100 - mae: 654.6100 - val_loss: 374.6586 - val_mae: 374.6586
     Epoch 3/1000
     3/3 [==============================] - 0s 20ms/step - loss: 654.5422 - mae: 654.5422 - val_loss: 374.6062 - val_mae: 374.6062
     Epoch 4/1000
     3/3 [==============================] - 0s 16ms/step - loss: 654.4731 - mae: 654.4731 - val_loss: 374.5522 - val_mae: 374.5522
     Epoch 5/1000
     3/3 [==============================] - 0s 22ms/step - loss: 654.4009 - mae: 654.4009 - val_loss: 374.4981 - val_mae: 374.4981
     Epoch 6/1000
     3/3 [==============================] - 0s 29ms/step - loss: 654.3297 - mae: 654.3297 - val_loss: 374.4421 - val_mae: 374.4421
     Epoch 7/1000
     3/3 [==============================] - 0s 25ms/step - loss: 654.2526 - mae: 654.2526 - val_loss: 374.3852 - val_mae: 374.3852
     Epoch 8/1000
     3/3 [==============================] - 0s 16ms/step - loss: 654.1753 - mae: 654.1753 - val_loss: 374.3269 - val_mae: 374.3269
     Epoch 9/1000
     3/3 [==============================] - 0s 24ms/step - loss: 654.0985 - mae: 654.0985 - val_loss: 374.2681 - val_mae: 374.2681
     Epoch 10/1000
     3/3 [==============================] - 0s 24ms/step - loss: 654.0176 - mae: 654.0176 - val_loss: 374.2070 - val_mae: 374.2070
     Epoch 11/1000
     3/3 [==============================] - 0s 21ms/step - loss: 653.9351 - mae: 653.9351 - val_loss: 374.1451 - val_mae: 374.1451
     Epoch 12/1000
     3/3 [==============================] - 0s 24ms/step - loss: 653.8521 - mae: 653.8521 - val_loss: 374.0808 - val_mae: 374.0808
     Epoch 13/1000
     ...
     Epoch 1000/1000
     3/3 [==============================] - 0s 24ms/step - loss: 50.8248 - mae: 50.8248 - val_loss: 41.4749 - val_mae: 41.4749
     1/1 [==============================] - 0s 48ms/step
     Mean Absolute Error on the Test Data: 41.474884033203125
     Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Comparison:

HW3 Result:

```
Average Difference Magnitude for Nearest Neighbour method: 66.2
Average Difference Magnitude for Nearest Centroid method: 167.8077677224736
Average Difference Magnitude for Average Cluster method: 191.3357142857143
```

Neural Network Result:

```
Mean Absolute Error on the Test Data: 59.26579666137695
```

```
Mean Absolute Error on the Test Data: 41.80929183959961
```

The accuracy of the neural network is higher than the accuracy of the nearest neighbor and average cluster methods. The neural network model outperforms the nearest neighbor and average cluster methods on this problem. This is likely because the neural network can learn complex relationships between the input features and the output target.

Hence, neural network model is a good choice for predicting 2022 citations based on all the 2017-2021 citations. It is able to achieve high accuracy and outperforms other simple methods, such as nearest neighbor and average cluster.

References:

github.com/keras-team/keras/issues/12062
https://www.tensorflow.org/api_docs/python/tf/keras/Sequential