Kavya Shah SUID: 330126500 PIN: 68

Introduction to ML HW-04

Introduction:

I tried a 1-hidden layer **neural network (5-3-1 architecture)** using the back-propagation algorithm and predicted the values of cit_2022 based on all the 2017 to 2021 citations.
I have used Tensor flow , keras and Relu activation functions on both the hidden and output nodes.
Utilised Stochastic gradient descent (SGD) optimization technique of gradient descent.

Steps:

1. Imported the important and useful libraries.
2. Loaded the data set into the python notebook using pandas.
3. Explored and analysed the data.
4. Splitting the data into training (80 %) and testing (20 %) set using sklearn.
5. Normalising both the train and test data.
6. Defining the neural network 5-3-1 architecture using TensorFlow.
7. Defining hyper parameters i.e learning rate and epochs.
8. Calculate the Mean Absolute Error for the predicted and tested value.

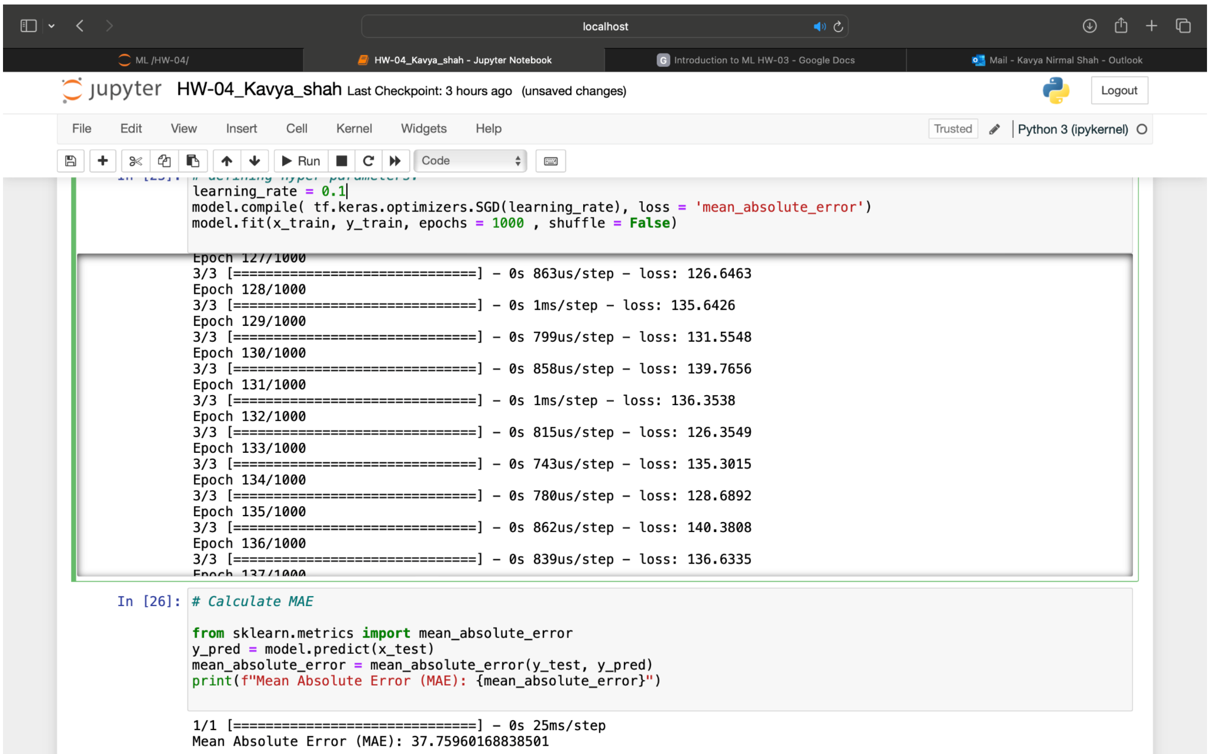
Result:

Epoch	**Learning rate**	**Output**
1000	0.1	37.75960168838501
1000	0.01	50.229731702804564
1000	0.001	180.7489749908447
1000	0.6	122.49656753540039
2000	0.1	37.766401863098146

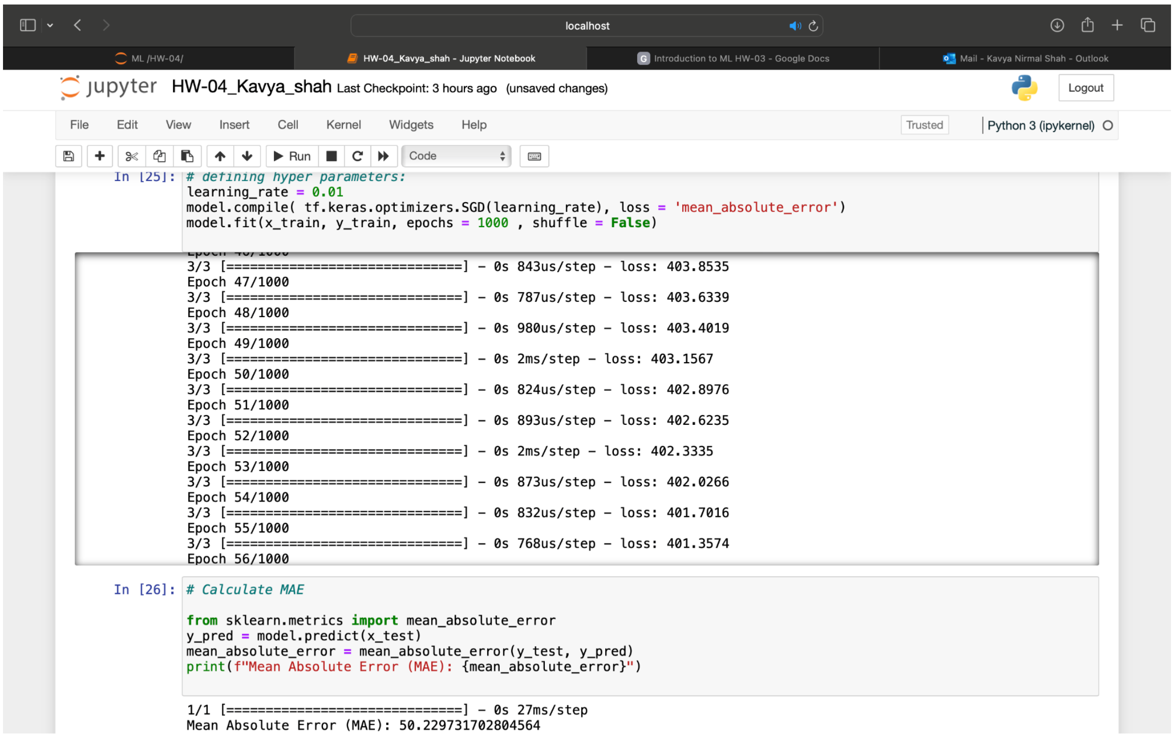
Kavya Shah SUID: 330126500 PIN: 68

Output:

1. *For epoch: 1000, learning rate = 0.1:*

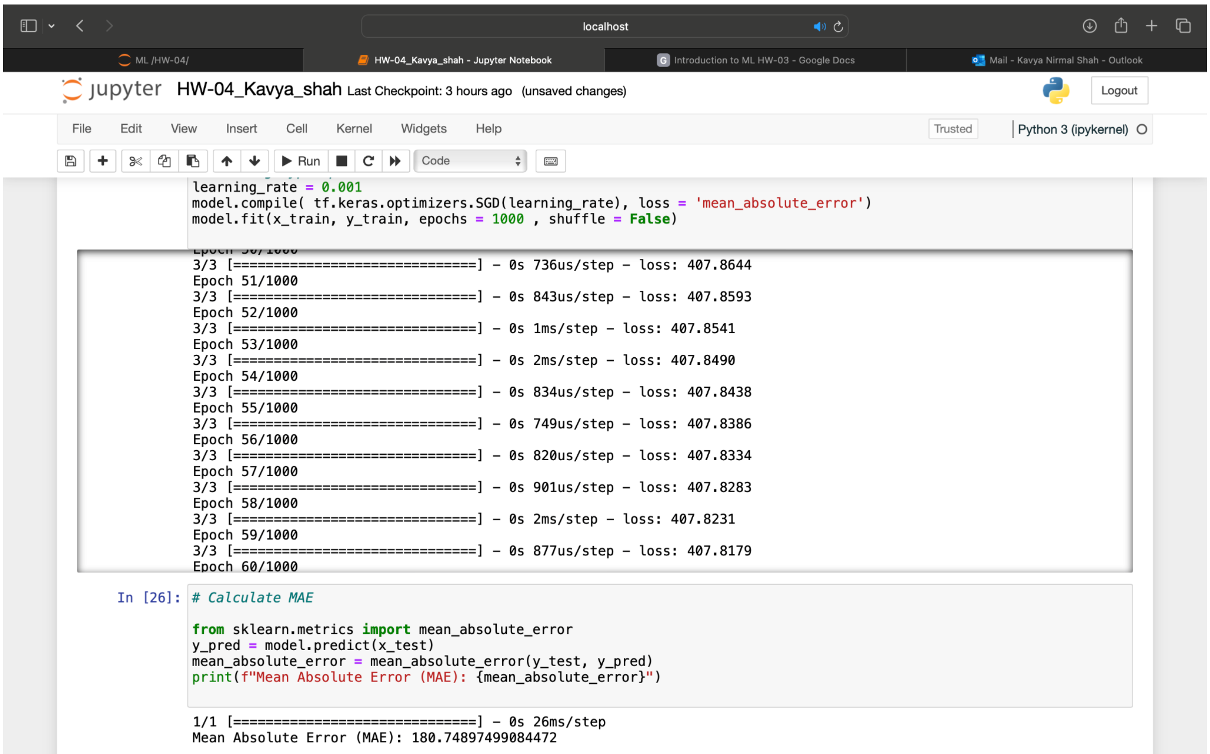


2. *For epoch: 1000, learning rate = 0.01:*



Kavya Shah SUID: 330126500 PIN: 68

3. *For epoch: 1000, learning rate = 0.001:*




```
learning_rate = 0.001
model.compile( tf.keras.optimizers.SGD(learning_rate), loss = 'mean_absolute_error')
model.fit(x_train, y_train, epochs = 1000 , shuffle = False)
```

```
3/3 [==============================] - 0s 736us/step - loss: 407.8644
Epoch 51/1000
3/3 [==============================] - 0s 843us/step - loss: 407.8593
Epoch 52/1000
3/3 [==============================] - 0s 1ms/step - loss: 407.8541
Epoch 53/1000
3/3 [==============================] - 0s 2ms/step - loss: 407.8490
Epoch 54/1000
3/3 [==============================] - 0s 834us/step - loss: 407.8438
Epoch 55/1000
3/3 [==============================] - 0s 749us/step - loss: 407.8386
Epoch 56/1000
3/3 [==============================] - 0s 820us/step - loss: 407.8334
Epoch 57/1000
3/3 [==============================] - 0s 901us/step - loss: 407.8283
Epoch 58/1000
3/3 [==============================] - 0s 2ms/step - loss: 407.8231
Epoch 59/1000
3/3 [==============================] - 0s 877us/step - loss: 407.8179
Epoch 60/1000
```

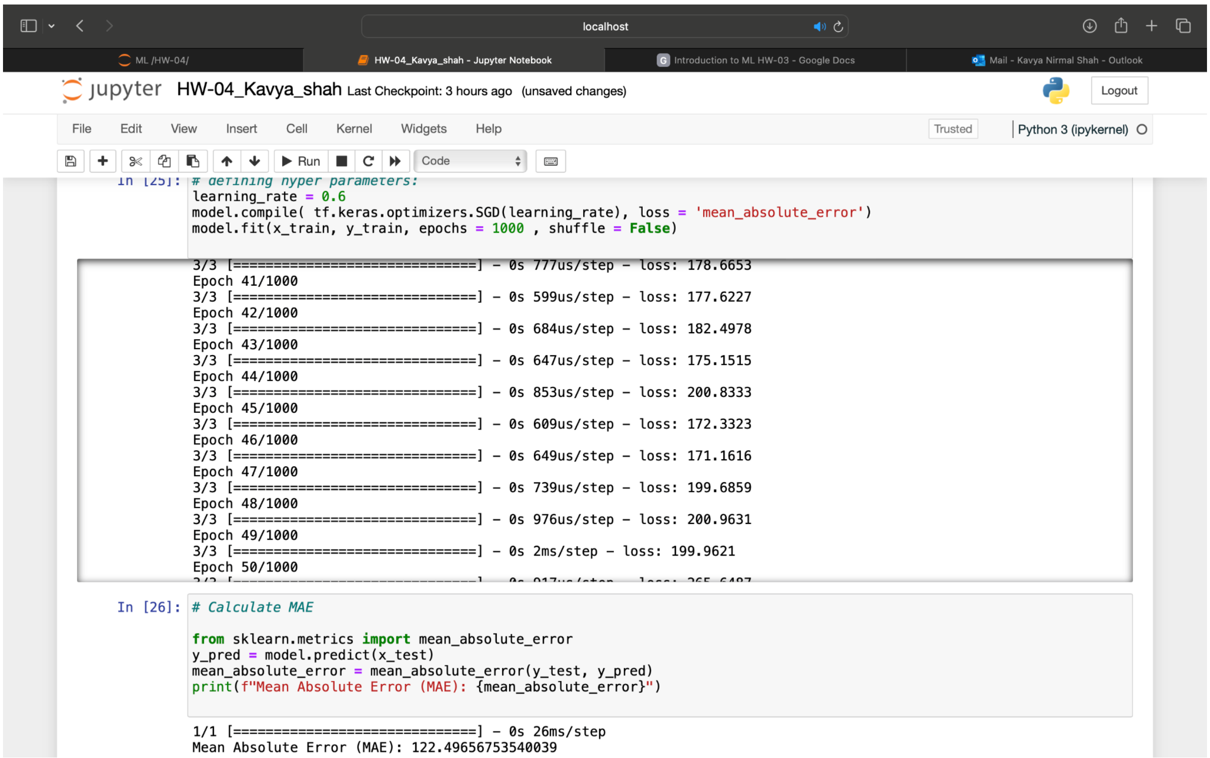
In [26]:

```
# Calculate MAE

from sklearn.metrics import mean_absolute_error
y_pred = model.predict(x_test)
mean_absolute_error = mean_absolute_error(y_test, y_pred)
print(f"Mean Absolute Error (MAE): {mean_absolute_error}")
```

```
1/1 [==============================] - 0s 26ms/step
Mean Absolute Error (MAE): 180.74897499084472
```

4. *For epoch: 1000, learning rate = 0.6:*



In [25]:

```
# defining hyper parameters:
learning_rate = 0.6
model.compile( tf.keras.optimizers.SGD(learning_rate), loss = 'mean_absolute_error')
model.fit(x_train, y_train, epochs = 1000 , shuffle = False)
```

```
3/3 [==============================] - 0s 777us/step - loss: 178.6653
Epoch 41/1000
3/3 [==============================] - 0s 599us/step - loss: 177.6227
Epoch 42/1000
3/3 [==============================] - 0s 684us/step - loss: 182.4978
Epoch 43/1000
3/3 [==============================] - 0s 647us/step - loss: 175.1515
Epoch 44/1000
3/3 [==============================] - 0s 853us/step - loss: 200.8333
Epoch 45/1000
3/3 [==============================] - 0s 609us/step - loss: 172.3323
Epoch 46/1000
3/3 [==============================] - 0s 649us/step - loss: 171.1616
Epoch 47/1000
3/3 [==============================] - 0s 739us/step - loss: 199.6859
Epoch 48/1000
3/3 [==============================] - 0s 976us/step - loss: 200.9631
Epoch 49/1000
3/3 [==============================] - 0s 2ms/step - loss: 199.9621
Epoch 50/1000
```

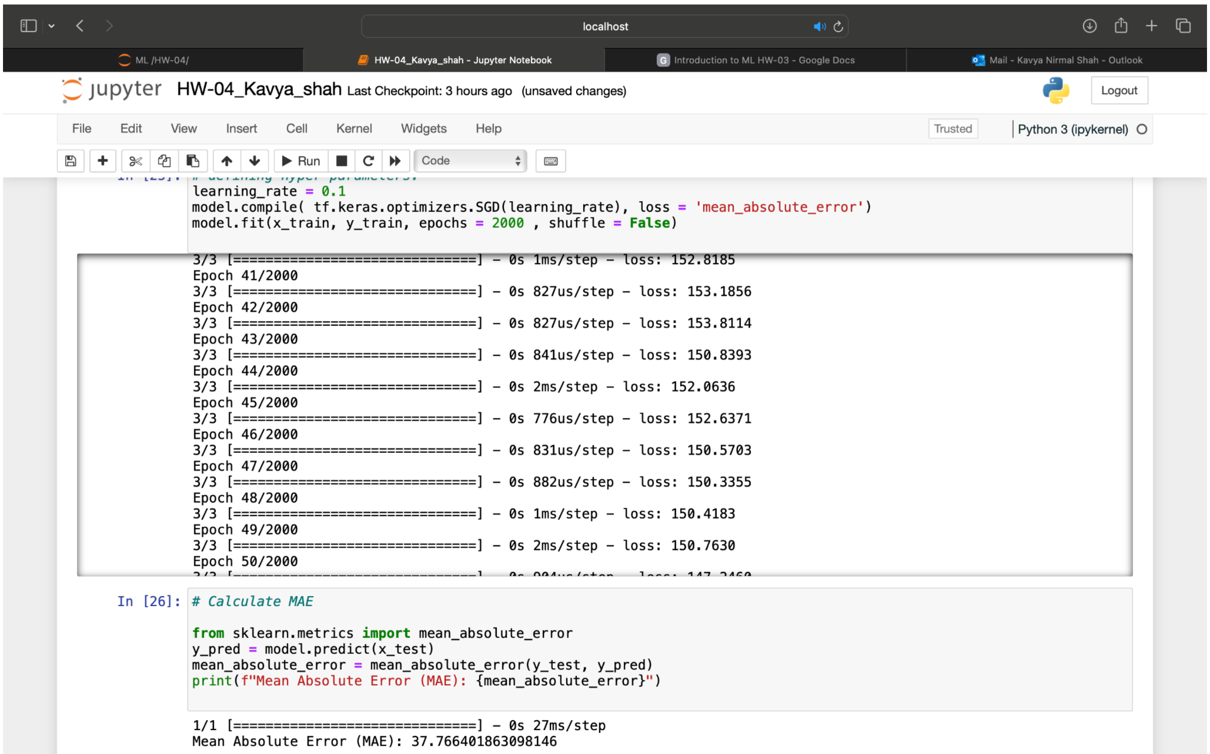
In [26]:

```
# Calculate MAE

from sklearn.metrics import mean_absolute_error
y_pred = model.predict(x_test)
mean_absolute_error = mean_absolute_error(y_test, y_pred)
print(f"Mean Absolute Error (MAE): {mean_absolute_error}")
```

```
1/1 [==============================] - 0s 26ms/step
Mean Absolute Error (MAE): 122.49656753540039
```

5. *For epoch: 2000, learning rate = 0.1:*



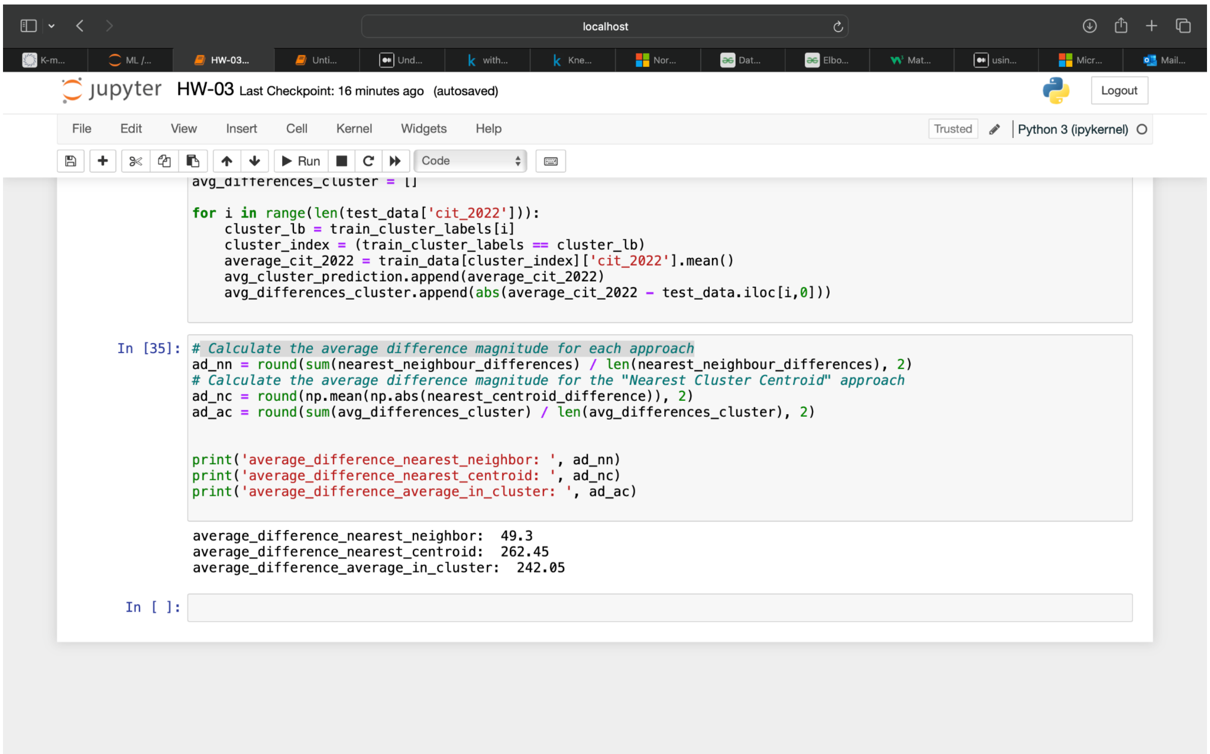
Now, In Hw-03,

Using k = 3, I labelled the normalised data into KMeans Clustering Model and then repeated the process for the normalised test data set

Questions:	**Method used**	**Output**
Question 1	Average difference magnitude for nearest neighbour	49.3
Question 2	Point nearest the cluster centroid	262.45
Question 3	Average of all others from the training set	242.05

Kavya Shah SUID: 330126500 PIN: 68

HW-03 output:




```
avg_differences_cluster = []

for i in range(len(test_data['cit_2022'])):
    cluster_lb = train_cluster_labels[i]
    cluster_index = (train_cluster_labels == cluster_lb)
    average_cit_2022 = train_data[cluster_index]['cit_2022'].mean()
    avg_cluster_prediction.append(average_cit_2022)
    avg_differences_cluster.append(abs(average_cit_2022 - test_data.iloc[i,0]))
```

In [35]:
```
# Calculate the average difference magnitude for each approach
ad_nn = round(sum(nearest_neighbour_differences) / len(nearest_neighbour_differences), 2)
# Calculate the average difference magnitude for the "Nearest Cluster Centroid" approach
ad_nc = round(np.mean(np.abs(nearest_centroid_difference)), 2)
ad_ac = round(sum(avg_differences_cluster) / len(avg_differences_cluster), 2)


print('average_difference_nearest_neighbor: ', ad_nn)
print('average_difference_nearest_centroid: ', ad_nc)
print('average_difference_average_in_cluster: ', ad_ac)
```

```
average_difference_nearest_neighbor:  49.3
average_difference_nearest_centroid:  262.45
average_difference_average_in_cluster:  242.05
```

In [ ]:

Conclusion:

In this dataset, we can conclude that the predictions generated by a backpropagation neural network with a 5-3-1 architecture, trained with 1000 epochs and at 0.1 learning rate are more accurate and efficient compared to the predictions made by the nearest neighbour algorithm used in HW-03. The neural network's output, which is approximately 37.76, is a lower mean error compared to the nearest neighbour, where mean absolute error was 49.3.

- **Y- test and Y-predicted values using 1000 epochs and at 0.1 learning rate:**

Kavya Shah SUID: 330126500 PIN: 68

Y - test:

In [28]: y_test

Out[28]:

	cit_2022
83	782
53	81
70	80
45	404
44	89
39	69
22	96
80	297
10	422
0	397
18	344
30	42
73	88
33	2
90	53
4	741
76	129
77	95
12	509
31	9

Y -predicted:

In [27]: y_pred

Out[27]:

```
array([[692.8738  ],
       [ 91.59355 ],
       [ 66.454506],
       [401.33453 ],
       [122.34772 ],
       [110.199554],
       [108.72528 ],
       [259.09006 ],
       [335.62357 ],
       [458.32962 ],
       [319.7795  ],
       [ 20.921854],
       [ 58.752457],
       [ 19.690155],
       [ 33.98474 ],
       [628.07434 ],
       [104.128235],
       [106.52757 ],
       [600.3913  ],
       [ 23.404865]], dtype=float32)
```