Nama : Marita Fauziah

NIM : 2101201046

"TUGAS SISTEM CERDAS TENSOR FLOW & KERAS"

- Memprediksi nilai dari Cosinus suatu dari suatu bilangan yang dirumuskan sebagai berikut :
  y = cosinus (x)

  Tutorial sebagai berikut :
  1. Tahap Pertama, Melakukan instalasi serta mengimport model dan library yang akandigunakan pada simulasi untuk memprediksi fungsi cosinus dari suatu bilangan

Code

```
#Melakukan instalasi os dan juga mendeklarasikan jalur pada model files
import os
MODELS_DIR = 'models/'
if not os.path.exists(MODELS_DIR):
    os.mkdir(MODELS_DIR)
MODEL_TF = MODELS_DIR + 'model'
MODEL_NO_QUANT_TFLITE = MODELS_DIR + 'model_no_quant.tflite'
MODEL_TFLITE = MODELS_DIR + 'model.tflite'
MODEL_TFLITE_MICRO = MODELS_DIR + 'model.cc'
#Melakukan instalasi library tensor flow
! pip install tensorflow==2.4.0rc0
import tensorflow as tf #melakukan pengimportan library tensorflow dimana library tersebut berkaitan dengan deep learning atau digunakan untuk deep learning, library ini dikembangkan oleh Google Brain Team
from tensorflow import keras #melakukan pengimportan libary keras yang terdapat pada tensorflow, libarary keras sendiri kurang lebih fungsi yang sama yaitu untuk deep learning dan dikembangkan oleh Francois Chollet
import numpy as np  #melakukan pengimportan library Numpy adalah library phyton yang fokus kepada scientific computing, biasanya digunakan untuk membuat objek N-dimensional array
import pandas as pd # melakukan pengimportan library Pandas adalah library phyton yang berfokus pada analisa data yang memiliki struktur data yang diperlukan untuk membersihkan data mentah
#ke dalam sebuah bentuk yang cocok untuk dianalisis. Data disini berasal dari dataset yang sudah diimport sebelumnya
import matplotlib.pyplot as plt  #melakukan pengimportan library matplotlib adalah library yang memiliki fungsi sama dengan Seaborn, namun seaborn memiliki beberapa kelebihan dibandingkan matplotlib
import math #melakukan pengimportan library math
seed = 1
np.random.seed(seed)
```

Hasil Running

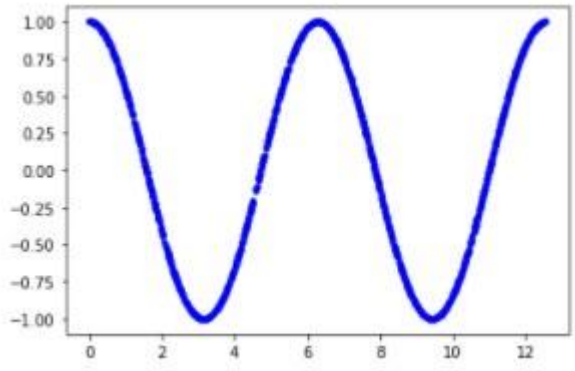
```
Requirement already satisfied: tensorflow==2.4.0rc0 in /usr/local/lib/python3.7/dist-packages (2.4.0rc0)
Requirement already satisfied: keras-preprocessing~=1.1.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (1.1.2)
Requirement already satisfied: grpcio~=1.32.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (1.32.0)
Requirement already satisfied: opt-einsum~=3.3.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (3.3.0)
Requirement already satisfied: numpy~=1.19.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (1.19.5)
Requirement already satisfied: wrapt~=1.12.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (1.12.1)
Requirement already satisfied: typing-extensions~=3.7.4 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (3.7.4.3)
Requirement already satisfied: six~=1.15.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (1.15.0)
Requirement already satisfied: h5py~=2.10.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (2.10.0)
Requirement already satisfied: google-pasta~=0.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (0.2.0)
Requirement already satisfied: astunparse~=1.6.3 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (1.6.3)
Requirement already satisfied: protobuf~=3.13.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (3.13.0)
Requirement already satisfied: flatbuffers~=1.12.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (1.12)
Requirement already satisfied: absl-py~=0.10 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (0.12.0)
Requirement already satisfied: termcolor~=1.1.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (1.1.0)
Requirement already satisfied: wheel~=0.35 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (0.36.2)
Requirement already satisfied: tensorflow-estimator~=2.3.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (2.3.0)
Requirement already satisfied: tensorboard~=2.3 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (2.5.0)
Requirement already satisfied: gast==0.3.3 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (0.3.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (from protobuf~=3.13.0->tensorflow==2.4.0rc0) (56.1.0)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.3->tensorflow==2.4.0rc0) (0.6.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.3->tensorflow==2.4.0rc0) (1.8.0)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.3->tensorflow==2.4.0rc0) (2.23.0)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.3->tensorflow==2.4.0rc0) (0.4.4)
Requirement already satisfied: google-auth<2,>=1.6.3 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.3->tensorflow==2.4.0rc0) (1.30.0)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.3->tensorflow==2.4.0rc0) (3.3.4)
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.3->tensorflow==2.4.0rc0) (1.0.1)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard~=2.3->tensorflow==2.4.0rc0) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard~=2.3->tensorflow==2.4.0rc0) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard~=2.3->tensorflow==2.4.0rc0) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard~=2.3->tensorflow==2.4.0rc0) (2020.12.5)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.7/dist-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard~=2.3->tensorflow==2.4.0rc0) (1.3.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.7/dist-packages (from google-auth<2,>=1.6.3->tensorboard~=2.3->tensorflow==2.4.0rc0) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4; python_version >= "3.6" in /usr/local/lib/python3.7/dist-packages (from google-auth<2,>=1.6.3->tensorboard~=2.3->tensorflow==2.4.0rc0) (4.7.2)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from google-auth<2,>=1.6.3->tensorboard~=2.3->tensorflow==2.4.0rc0) (4.2.2)
Requirement already satisfied: importlib-metadata; python_version < "3.8" in /usr/local/lib/python3.7/dist-packages (from markdown>=2.6.8->tensorboard~=2.3->tensorflow==2.4.0rc0) (4.0.1)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard~=2.3->tensorflow==2.4.0rc0) (3.1.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.7/dist-packages (from pyasn1-modules>=0.2.1->google-auth<2,>=1.6.3->tensorboard~=2.3->tensorflow==2.4.0rc0) (0.4.8)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata; python_version < "3.8"->markdown>=2.6.8->tensorboard~=2.3->tensorflow==2.4.0rc0) (3.4.1)
```

2. Tahap Kedua, Melakukan pembuatan angka dengan jumlah data titik sample yang digunakan 2000 kemudian melakukan pemrosesan fungsi cosinus pada bilangan tersebut dan melakukan ploting bentuk grafik

Code

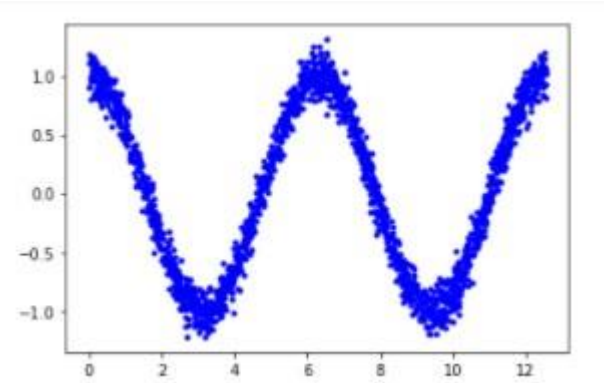
```
seed = 1
np.random.seed(seed) #digunakan untuk membuat angka secara acak yang dapat diprekdisi
SAMPLES = 2000 #mendeklarasikan jumlah titk data sample yang digunakan, pada percobaan ini digunakan 2000 sample
x_values = np.random.uniform(low=0, high=4*math.pi, size=SAMPLES).astype(np.float32) #mengenerata gelombang cosinus yang diinginkan dengan batas 0 hingga 4 phi.
np.random.shuffle(x_values) #melakukan proses shuffle yang digunakan untuk memastikan data tidak berurutan
y_values = np.cos(x_values).astype(np.float32) #melakukan perhitungan untuk fungsi cosinus
plt.plot(x_values, y_values, 'b.') #melakukan ploting dalam bentuk grafik dari hasil kalkulasi fungsi cosinus sebelumnya
plt.show() # menampilkan ploting grafik yang sudah dibuat sebelumnya
```

Hasil Running (Grafik Cosinus)



3. Melakukan penambahan bilangan random pada fungsi y untuk menghasilkan data fungsi cosinus menjadi random (kotor) sehingga dengan menggunakan code deep learning disimulasikan untuk melakukan prediksi dari bentuk bersih dari data cosinus tersebut dan melakukan ploting untuk ha-sil penambahan bilangan tersebut

Code
y_values += 0.1 * np.random.randn(*y_values.shape) plt.plot(x_values, y_values, 'b.') plt.show()

Hasil Running


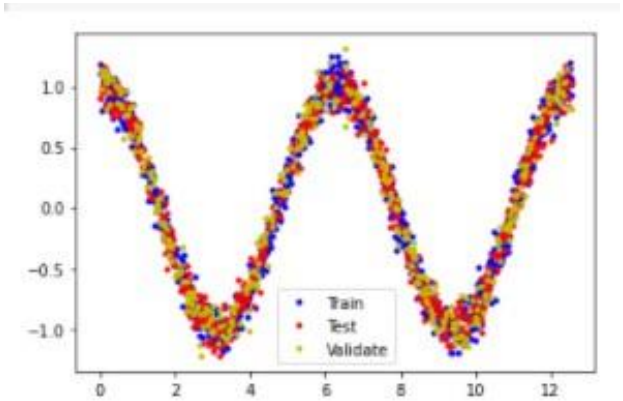
4. Code dibawah menjelaskan proses pembagian data set menjadi tiga bagian yaitu data train, data test dan data validasi. Perbandingan yang dilakukan adalah 0.5:0.3:0.2 dimana data test yang digunakan 30% sedangkan data train 50% dan data validasi sebesar : 20%. Catatan : Data train adalah Data yang digunakan untuk melatih algoritma Data testing adalah Data yang dipakai untuk mengetahui performa algoritma yang sudah dilatih sebelumnya ketika menemukan data baru yang belum pernah dilihat sebelumnya.Setelah data dibuat menjadi lebih kecil, kemudian dilanjutkan dengan proses pelatihan dari data training. Pelatihan dari data training dapat disebut juga model.

Code

```
#mendeklarasikan pembagian jumlah dari setiap bagian data yang akan digunakan yaitu data train, data test dan data validasi
TRAIN_SPLIT =  int(0.5 * SAMPLES)
TEST_SPLIT = int(0.3 * SAMPLES + TRAIN_SPLIT)

#digunakan untuk membagi dataset menjadi 3 bagian yang sudah dideklerasikan sebelumnya jumlahnya
x_train, x_test, x_validate = np.split(x_values, [TRAIN_SPLIT, TEST_SPLIT])
y_train, y_test, y_validate = np.split(y_values, [TRAIN_SPLIT, TEST_SPLIT])
#digunakan untuk memeriksa kesesuaian data yang sudah dibagi
assert (x_train.size + x_validate.size + x_test.size) ==  SAMPLES
#melakukan plotingan terhadap 3 data sebelumnya dengan warna yang berbeda untuk representasikan data tersebut
plt.plot(x_train, y_train, 'b.', label="Train")
plt.plot(x_test, y_test, 'r.', label="Test")
plt.plot(x_validate, y_validate, 'y.', label="Validate")
plt.legend()
plt.show()
```

Hasil Running



5. BAGIAN DEEP LEARNING

- **Skenario satu**

Melakukan pemodelan Keras untuk deep learning serta men-traingin data yang sudah dibagi sebelumnya.
Pada code dibawah dapat dilihat model yang digunakan adalah pemodelan sequential. Dalam codingan dibawah ini jumlah hidden layer ada 1, dimana untuk menambahkan hidden layer menggunakan Dense, (10,,activation='relu'input_shape (1)) sedangkan untuk didalamnya dapat dijelak an bahwa 10 adalah neuron pada hidden layer 1, kemudian 1 adalah neuron pada bagian input sedangkan relu adalah salah satu fungsi aktivasi.Ju mlah layer dan jumlah neuron dapat disesuaikan. Kemudian melakukan konfigurasi pelatihan seperti tertera dibawah. Kemudian traning data dila kukan.
Epoch adalah kondisi ketika seluruh dataset sudah melalui proses training pada Neural Netwok sampai dikembalikan ke awal untuk sekali putara n, karena satu Epoch terlalu besar untuk dimasukkan (feeding) kedalam komputer maka dari itu kita perlu membaginya kedalam satuan kecil (bat ches).Vall_loss adalah nilai fungsi biaya untuk data validasi silang Anda dan kerugian adalah nilai fungsi biaya untuk data pelatihan Anda. Pada data validasi, neuron yang menggunakan drop out tidak menjatuhkan neuron acak. Alasannya adalah selama pelatihan kami menggunakan drop o ut untuk menambah kebisingan untuk menghindari pemasangan yang berlebihan. optimizer adam digunakan dalam tugas klasifikasi

Code

```
model = tf.keras.Sequential()
model.add(keras.layers.Dense(10, activation='relu', input_shape=(1,)))
model.add(keras.layers.Dense(1))
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
history = model.fit(x_train, y_train, epochs=400, batch_size=64, validation_data=(x_validate, y_validate))
```

Hasil Running

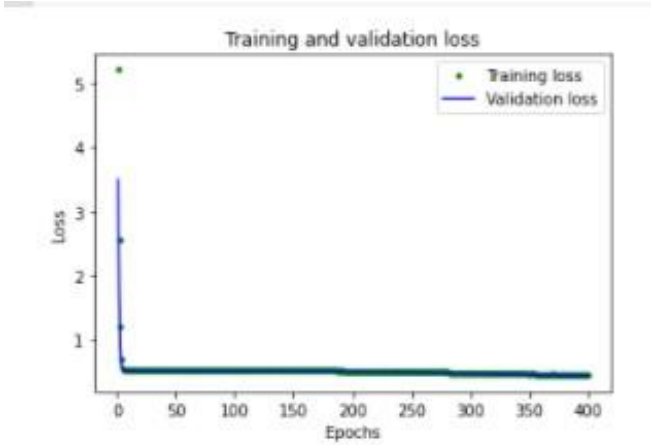
```
Epoch 385/400
16/16 [==============================] - 0s 4ms/step - loss: 0.4522 - mae: 0.5929 - val_loss: 0.4393 - val_mae: 0.5842
Epoch 386/400
16/16 [==============================] - 0s 4ms/step - loss: 0.4351 - mae: 0.5782 - val_loss: 0.4350 - val_mae: 0.5815
Epoch 387/400
16/16 [==============================] - 0s 4ms/step - loss: 0.4582 - mae: 0.6028 - val_loss: 0.4409 - val_mae: 0.5847
Epoch 388/400
16/16 [==============================] - 0s 5ms/step - loss: 0.4600 - mae: 0.6007 - val_loss: 0.4339 - val_mae: 0.5807
Epoch 389/400
16/16 [==============================] - 0s 4ms/step - loss: 0.4530 - mae: 0.5941 - val_loss: 0.4341 - val_mae: 0.5807
Epoch 390/400
16/16 [==============================] - 0s 4ms/step - loss: 0.4504 - mae: 0.5909 - val_loss: 0.4460 - val_mae: 0.5866
Epoch 391/400
16/16 [==============================] - 0s 4ms/step - loss: 0.4647 - mae: 0.6009 - val_loss: 0.4326 - val_mae: 0.5797
Epoch 392/400
16/16 [==============================] - 0s 4ms/step - loss: 0.4521 - mae: 0.5966 - val_loss: 0.4392 - val_mae: 0.5833
Epoch 393/400
16/16 [==============================] - 0s 4ms/step - loss: 0.4563 - mae: 0.5988 - val_loss: 0.4334 - val_mae: 0.5800
Epoch 394/400
16/16 [==============================] - 0s 4ms/step - loss: 0.4452 - mae: 0.5869 - val_loss: 0.4429 - val_mae: 0.5847
Epoch 395/400
16/16 [==============================] - 0s 4ms/step - loss: 0.4512 - mae: 0.5949 - val_loss: 0.4314 - val_mae: 0.5786
Epoch 396/400
16/16 [==============================] - 0s 4ms/step - loss: 0.4485 - mae: 0.5920 - val_loss: 0.4364 - val_mae: 0.5815
Epoch 397/400
16/16 [==============================] - 0s 5ms/step - loss: 0.4534 - mae: 0.5935 - val_loss: 0.4328 - val_mae: 0.5793
Epoch 398/400
16/16 [==============================] - 0s 4ms/step - loss: 0.4501 - mae: 0.5951 - val_loss: 0.4307 - val_mae: 0.5779
Epoch 399/400
16/16 [==============================] - 0s 5ms/step - loss: 0.4553 - mae: 0.5926 - val_loss: 0.4380 - val_mae: 0.5819
Epoch 400/400
16/16 [==============================] - 0s 5ms/step - loss: 0.4671 - mae: 0.6062 - val_loss: 0.4311 - val_mae: 0.5780
```

Menampilkan hasil grafik error dari hasil training dengan memproses loss data training dan loss data validation.

Code

```
train_loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(train_loss) + 1)
plt.plot(epochs, train_loss, 'g.', label='Training loss')
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()
```

Hasil Running

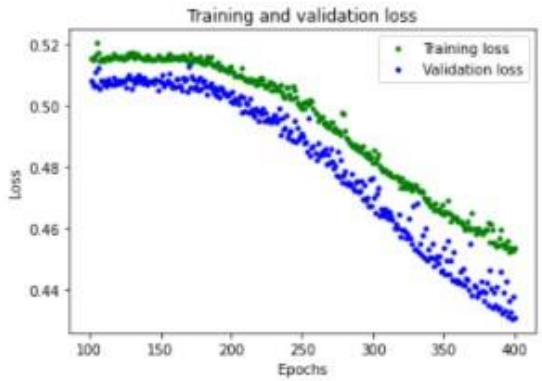


Menampilkan hasil grafik error dari hasil training dengan memproses loss data training dan loss data validation dengan jumlah SKIP= 100

Code

```
SKIP = 100
plt.plot(epochs[SKIP:], train_loss[SKIP:], 'g.', label='Training loss')
plt.plot(epochs[SKIP:], val_loss[SKIP:], 'b.', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

Hasil Running

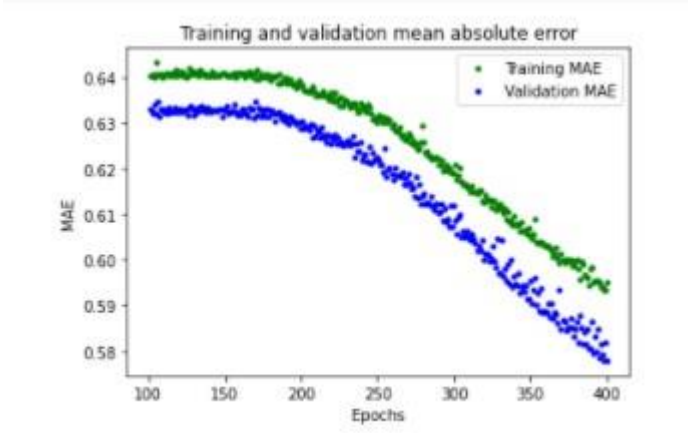


Melakukan kalkulasi jumlah kesalahan pada hasil prediksi dan melihat bentuk grafik dari sisi Mean Absolute error dan memploting hasil perbandingan tersebut.

Code

```
plt.clf()
train_mae = history.history['mae']
val_mae = history.history['val_mae']
plt.plot(epochs[SKIP:], train_mae[SKIP:], 'g.', label='Training MAE')
plt.plot(epochs[SKIP:], val_mae[SKIP:], 'b.', label='Validation MAE')
plt.title('Training and validation mean absolute error')
plt.xlabel('Epochs')
plt.ylabel('MAE')
plt.legend()
plt.show()
```

Hasil Running



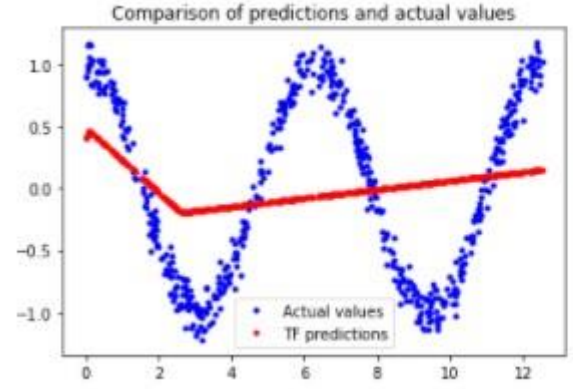
Hasil perbandingan nilai aktual dan juga nilai prekdisi yang sudah melalui tahapan deep learning pada skenario satu

Code

```
test_loss, test_mae = model.evaluate(x_test, y_test)
y_test_pred = model.predict(x_test)
plt.clf()
plt.title('Comparison of predictions and actual values')
plt.plot(x_test, y_test, 'b.', label='Actual values')
plt.plot(x_test, y_test_pred, 'r.', label='TF predictions')
plt.legend()
plt.show()
```

Hasil Running

```
19/19 [==============================] - 0s 1ms/step - loss: 0.4448 - mae: 0.5960
```



- **Skenario Dua**
  Melakukan pemodelan Keras untuk deep learning serta mentraining data yang sudah dibagi sebelumnya.
  Pada code dibawah dapat dilihat model yang digunakan adalah pemodelan sequential.
  Dalam codingan dibawah ini jumlah hidden layer ada 5 ,dimana untuk menambahkan hidden layer menggunakan Dense, (10,,activation='relu'input_shape (1)) sedangkan untuk didalamnya dapat dijelaskan bahwa 8,16,24,32, dan 40 adalah neuron pada hidden layer 1,2,3,4,5 secara berurutan, kemudian 1 adalah neuron pada bagian input sedangkan relu adalah salah satu fungsi aktivasi.Jumlah layer dan jumlah neuron dapat di sesuaikan. Kemudian melakukan konfigurasi pelatihan seperti tertera dibawah. Kemudian traning data dilakukan.
  Epoch adalah kondisi ketika seluruh dataset sudah melalui proses training pada Neural Netwok sampai dikembalikan ke awal untuk sekali putaran, karena satu Epoch terlalu besar untuk dimasukkan (feeding) kedalam komputer maka dari itu kita perlu membaginya kedalam satuan kecil (batches).Vall_loss adalah nilai fungsi biaya untuk data validasi silang Anda dan kerugian adalah nilai fungsi biaya untuk data pelatihan Anda. Pada data validasi, neuron yang menggunakan drop out tidak menjatuhkan neuron acak. Alasannya adalah selama pelatihan kami menggunakan drop out untuk menambah kebisingan untuk menghindari pemasangan yang berlebihan.Optimizer adam digunakan dalam tugas klasifikasi

Code

```
model_1 = tf.keras.Sequential()
model_1.add(keras.layers.Dense(8, activation='relu', input_shape=(1,)))
model_1.add(keras.layers.Dense(16, activation='relu'))
model_1.add(keras.layers.Dense(24, activation='relu'))
model_1.add(keras.layers.Dense(32, activation='relu'))
model_1.add(keras.layers.Dense(1))
model_1.compile(optimizer='adam', loss='mse', metrics=['mae'])
history_1 = model_1.fit(x_train, y_train, epochs=400, batch_size=64,validation_data=(x_validate, y_validate))
model_1.save(MODEL_TF)
```

Hasil Running

```
Epoch 386/400
8/8 [==============================] - 0s 9ms/step - loss: 0.0119 - mae: 0.0882 - val_loss: 0.0122 - val_mae: 0.0911
Epoch 387/400
8/8 [==============================] - 0s 8ms/step - loss: 0.0116 - mae: 0.0890 - val_loss: 0.0140 - val_mae: 0.0941
Epoch 388/400
8/8 [==============================] - 0s 9ms/step - loss: 0.0122 - mae: 0.0891 - val_loss: 0.0139 - val_mae: 0.0945
Epoch 389/400
8/8 [==============================] - 0s 9ms/step - loss: 0.0136 - mae: 0.0933 - val_loss: 0.0153 - val_mae: 0.0977
Epoch 390/400
8/8 [==============================] - 0s 8ms/step - loss: 0.0135 - mae: 0.0926 - val_loss: 0.0140 - val_mae: 0.0957
Epoch 391/400
8/8 [==============================] - 0s 9ms/step - loss: 0.0122 - mae: 0.0907 - val_loss: 0.0133 - val_mae: 0.0919
Epoch 392/400
8/8 [==============================] - 0s 9ms/step - loss: 0.0131 - mae: 0.0929 - val_loss: 0.0130 - val_mae: 0.0917
Epoch 393/400
8/8 [==============================] - 0s 8ms/step - loss: 0.0117 - mae: 0.0875 - val_loss: 0.0130 - val_mae: 0.0920
Epoch 394/400
8/8 [==============================] - 0s 9ms/step - loss: 0.0133 - mae: 0.0925 - val_loss: 0.0122 - val_mae: 0.0897
Epoch 395/400
8/8 [==============================] - 0s 9ms/step - loss: 0.0115 - mae: 0.0864 - val_loss: 0.0123 - val_mae: 0.0912
Epoch 396/400
8/8 [==============================] - 0s 10ms/step - loss: 0.0115 - mae: 0.0884 - val_loss: 0.0126 - val_mae: 0.0907
Epoch 397/400
8/8 [==============================] - 0s 10ms/step - loss: 0.0117 - mae: 0.0886 - val_loss: 0.0125 - val_mae: 0.0917
Epoch 398/400
8/8 [==============================] - 0s 11ms/step - loss: 0.0119 - mae: 0.0887 - val_loss: 0.0124 - val_mae: 0.0917
Epoch 399/400
8/8 [==============================] - 0s 9ms/step - loss: 0.0118 - mae: 0.0879 - val_loss: 0.0126 - val_mae: 0.0909
Epoch 400/400
8/8 [==============================] - 0s 10ms/step - loss: 0.0123 - mae: 0.0888 - val_loss: 0.0138 - val_mae: 0.0940
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:2325: UserWarning: `Model.state_updates` will be removed in a future version. This property should not be used in TensorFlow 2.0, as `
  warnings.warn('`Model.state_updates` will be removed in a future version. '
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/base_layer.py:1397: UserWarning: `layer.updates` will be removed in a future version. This property should not be used in TensorFlow 2.0, as `upda
  warnings.warn('`layer.updates` will be removed in a future version. '
WARNING:tensorflow:FOR KERAS USERS: The object that you are saving contains one or more Keras models or layers. If you are loading the SavedModel with `tf.keras.models.load_model`, continue reading (otherwise, you ma

FOR DEVS: If you are overwriting _tracking_metadata in your class, this property has been used to save metadata in the SavedModel. The metadta field will be deprecated soon, so please move the metadata to a different
INFO:tensorflow:Assets written to: models/model/assets
```

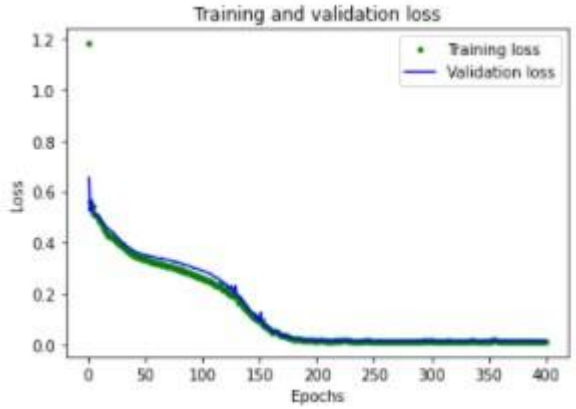
Menampilkan hasil grafik error dari hasil training dengan memproses loss data training dan loss data validation.

Code

```
train_loss = history_1.history['loss']
val_loss = history_1.history['val_loss']
epochs = range(1, len(train_loss) + 1)
plt.plot(epochs, train_loss, 'g.', label='Training loss')
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()
```

Hasil Running

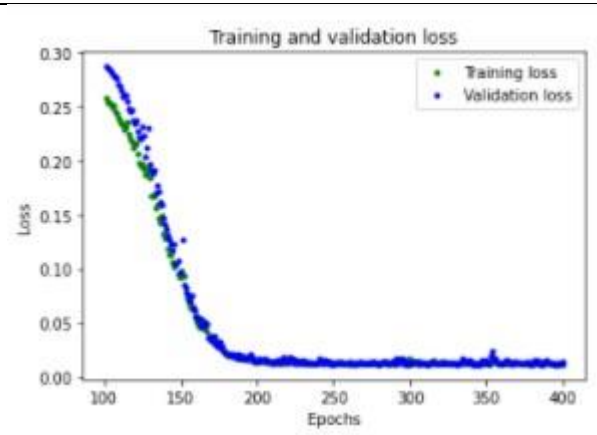


Menampilkan hasil grafik error dari hasil training dengan memproses loss data training dan loss data validation dengan jumlah SKIP= 100

Code

```
SKIP = 100
plt.plot(epochs[SKIP:], train_loss[SKIP:], 'g.', label='Training loss')
plt.plot(epochs[SKIP:], val_loss[SKIP:], 'b.', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

Hasil Running



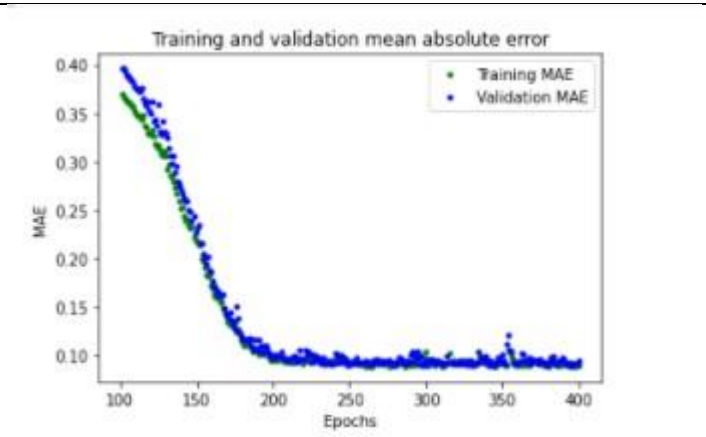
Melakukan kalkulasi jumlah kesalahan pada hasil prediksi dan melihat bentuk grafik dari sisi Mean Absolute error dan memploting hasil perbandingan tersebut.

Code

```
plt.clf()
train_mae = history_1.history['mae']
val_mae = history_1.history['val_mae']
```

```
plt.plot(epochs[SKIP:], train_mae[SKIP:], 'g.', label='Training MAE')
plt.plot(epochs[SKIP:], val_mae[SKIP:], 'b.', label='Validation MAE')
plt.title('Training and validation mean absolute error')
plt.xlabel('Epochs')
plt.ylabel('MAE')
plt.legend()
plt.show()
```

Hasil Running



Hasil perbandingan nilai aktual dan juga nilai prekdisi yang sudah melalui tahapan deep learning pada skenario satu

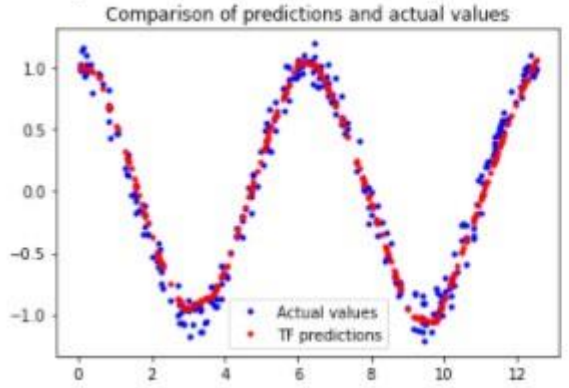
Code

```
test_loss, test_mae = model_1.evaluate(x_test, y_test)
y_test_pred = model_1.predict(x_test)
plt.clf()
plt.title('Comparison of predictions and actual values')
```

```
plt.plot(x_test, y_test, 'b.', label='Actual values')
plt.plot(x_test, y_test_pred, 'r.', label='TF predictions')
plt.legend()
plt.show()
```

Hasil Running

```
10/10 [==============================] - 0s 2ms/step - loss: 0.0112 - mae: 0.0860
```



- **Skenario Tiga**
  Melakukan pemodelan Keras untuk deep learning serta mentraingin data yang sudah dibagi sebelumnya.
  Pada code dibawah dapat dilihat model yang digunakan adalah pemodelan sequential.
  Dalam codingan dibawah ini jumlah hidden layer ada 5 ,dimana untuk menambahkan hidden layer menggunakan Dense, (10,,activation='relu'input_shape (1)) sedangkan untuk didalamnya dapat dijelaskan bahwa 8,16,24,32, dan 40 adalah neuron pada hidden layer 1,2,3,4,5 secara berurutan, kemudian 1 adalah neuron pada bagian input sedangkan relu adalah salah satu fungsi aktivasi.Jumlah layer dan jumlah neuron dapat di sesuaikan. Kemudian melakukan konfigurasi pelatihan seperti tertera dibawah. Kemudian traning data dilakukan.
  Epoch adalah kondisi ketika seluruh dataset sudah melalui proses training pada Neural Netwok sampai dikembalikan ke awal untuk sekali putaran, karena satu Epoch terlalu besar untuk dimasukkan (feeding) kedalam komputer maka dari itu kita perlu membaginya kedalam satuan kecil (batches).Vall_loss adalah nilai fungsi biaya untuk data validasi silang Anda dan kerugian adalah nilai fungsi biaya untuk data pelatihan Anda. Pada data validasi, neuron yang menggunakan drop out tidak menjatuhkan neuron acak. Alasannya adalah selama pelatihan kami menggunakan drop out untuk menambah kebisingan untuk menghindari pemasangan yang berlebihan.optimizer sgd digunakan dalam tugas klasifikasi

Code

```
model_2 = tf.keras.Sequential()
model_2.add(keras.layers.Dense(8, activation='relu', input_shape=(1,)))
model_2.add(keras.layers.Dense(16, activation='relu'))
model_2.add(keras.layers.Dense(24, activation='relu'))
model_2.add(keras.layers.Dense(32, activation='relu'))
model_2.add(keras.layers.Dense(40, activation='relu'))
model_2.add(keras.layers.Dense(1))
model_2.compile(optimizer='sgd', loss='mse', metrics=['mae'])
history_2 = model_2.fit(x_train, y_train, epochs=400, batch_size=64,validation_data=(x_validate, y_validate))
model_2.save(MODEL_TF)
```

Hasil Running

```
Epoch 387/400
16/16 [==============================] - 0s 5ms/step - loss: 0.0469 - mae: 0.1491 - val_loss: 0.1159 - val_mae: 0.2545
Epoch 388/400
16/16 [==============================] - 0s 6ms/step - loss: 0.0775 - mae: 0.1989 - val_loss: 0.1499 - val_mae: 0.3029
Epoch 389/400
16/16 [==============================] - 0s 5ms/step - loss: 0.1197 - mae: 0.2431 - val_loss: 0.2578 - val_mae: 0.3950
Epoch 390/400
16/16 [==============================] - 0s 5ms/step - loss: 0.1831 - mae: 0.3020 - val_loss: 0.0309 - val_mae: 0.1339
Epoch 391/400
16/16 [==============================] - 0s 5ms/step - loss: 0.0454 - mae: 0.1548 - val_loss: 0.0776 - val_mae: 0.2223
Epoch 392/400
16/16 [==============================] - 0s 5ms/step - loss: 0.0517 - mae: 0.1716 - val_loss: 0.4082 - val_mae: 0.4848
Epoch 393/400
16/16 [==============================] - 0s 5ms/step - loss: 0.1976 - mae: 0.3115 - val_loss: 0.0681 - val_mae: 0.2022
Epoch 394/400
16/16 [==============================] - 0s 5ms/step - loss: 0.1160 - mae: 0.2474 - val_loss: 0.0408 - val_mae: 0.1600
Epoch 395/400
16/16 [==============================] - 0s 5ms/step - loss: 0.0676 - mae: 0.1858 - val_loss: 0.0347 - val_mae: 0.1401
Epoch 396/400
16/16 [==============================] - 0s 5ms/step - loss: 0.0828 - mae: 0.2097 - val_loss: 0.0820 - val_mae: 0.2148
Epoch 397/400
16/16 [==============================] - 0s 5ms/step - loss: 0.1001 - mae: 0.2363 - val_loss: 0.0514 - val_mae: 0.1707
Epoch 398/400
16/16 [==============================] - 0s 5ms/step - loss: 0.1254 - mae: 0.2495 - val_loss: 0.0288 - val_mae: 0.1297
Epoch 399/400
16/16 [==============================] - 0s 5ms/step - loss: 0.0314 - mae: 0.1325 - val_loss: 0.0778 - val_mae: 0.2135
Epoch 400/400
16/16 [==============================] - 0s 5ms/step - loss: 0.1193 - mae: 0.2476 - val_loss: 0.0606 - val_mae: 0.1887
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:2325: UserWarning: `Model.state_updates` will be removed in a future version. This property should not be used in Tensor
  warnings.warn('`Model.state_updates` will be removed in a future version. '
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/base_layer.py:1397: UserWarning: `layer.updates` will be removed in a future version. This property should not be used in TensorFlow
  warnings.warn('`layer.updates` will be removed in a future version. '
WARNING:tensorflow:FOR KERAS USERS: The object that you are saving contains one or more Keras models or layers. If you are loading the SavedModel with `tf.keras.models.load_model`, continue reading (oth

FOR DEVS: If you are overwriting _tracking_metadata in your class, this property has been used to save metadata in the SavedModel. The metadta field will be deprecated soon, so please move the metadata
INFO:tensorflow:Assets written to: models/model/assets
```

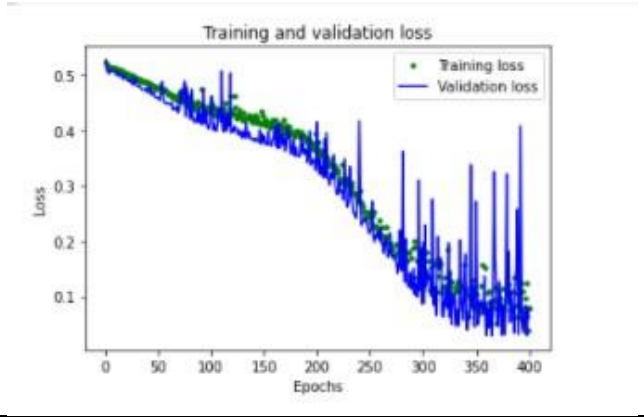
Menampilkan hasil grafik error dari hasil training dengan memproses loss data training dan loss data validation.

Code

```
train_loss = history_2.history['loss']
val_loss = history_2.history['val_loss']
epochs = range(1, len(train_loss) + 1)
plt.plot(epochs, train_loss, 'g.', label='Training loss')
```

```
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()
```

Hasil Running

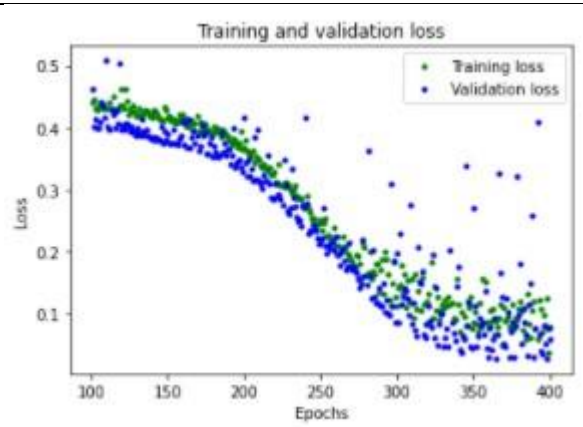


Menampilkan hasil grafik error dari hasil training dengan memproses loss data training dan loss data validation dengan jumlah SKIP= 100

Code

```
SKIP = 100
plt.plot(epochs[SKIP:], train_loss[SKIP:], 'g.', label='Training loss')
plt.plot(epochs[SKIP:], val_loss[SKIP:], 'b.', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

Hasil Running



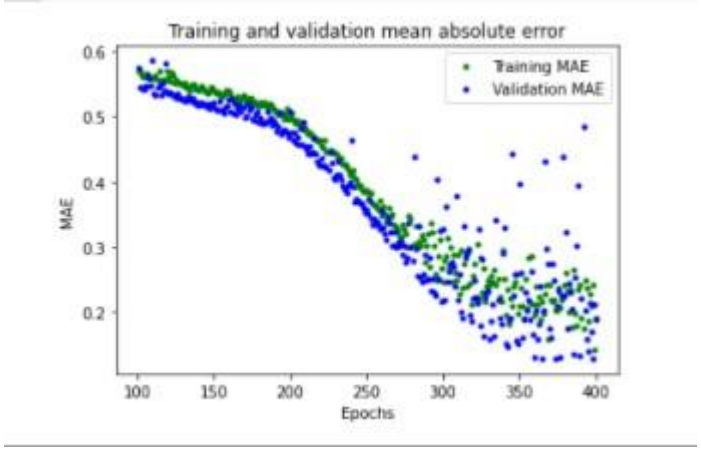
Melakukan kalkulasi jumlah kesalahan pada hasil prediksi dan melihat bentuk grafik dari sisi Mean Absolute error dan memploting hasil perbandingan tersebut.

Code

```
plt.clf()
train_mae = history_2.history['mae']
val_mae = history_2.history['val_mae']
plt.plot(epochs[SKIP:], train_mae[SKIP:], 'g.', label='Training MAE')
```

```
plt.plot(epochs[SKIP:], val_mae[SKIP:], 'b.', label='Validation MAE')
plt.title('Training and validation mean absolute error')
plt.xlabel('Epochs')
plt.ylabel('MAE')
plt.legend()
plt.show()
```

Hasil Running



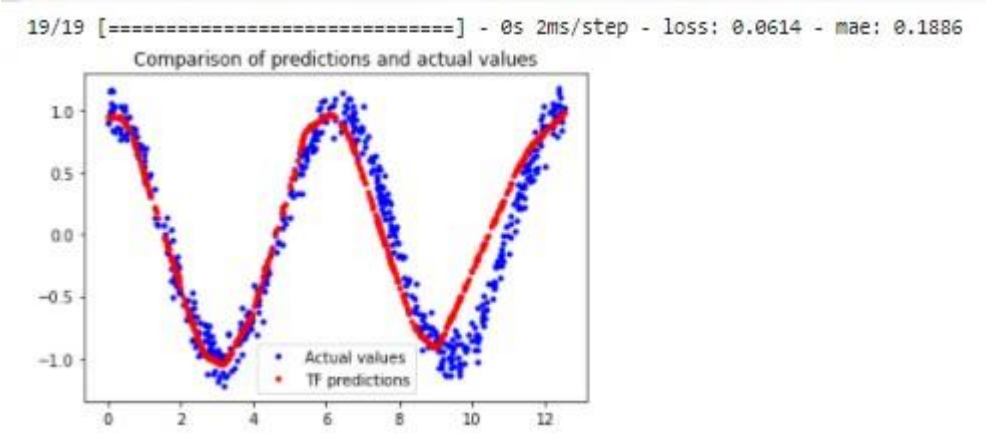
Hasil perbandingan nilai aktual dan juga nilai prekdisi yang sudah melalui tahapan deep learning pada skenario satu

Code

```
test_loss, test_mae = model_2.evaluate(x_test, y_test)
y_test_pred = model_2.predict(x_test)
plt.clf()
plt.title('Comparison of predictions and actual values')
plt.plot(x_test, y_test, 'b.', label='Actual values')
```

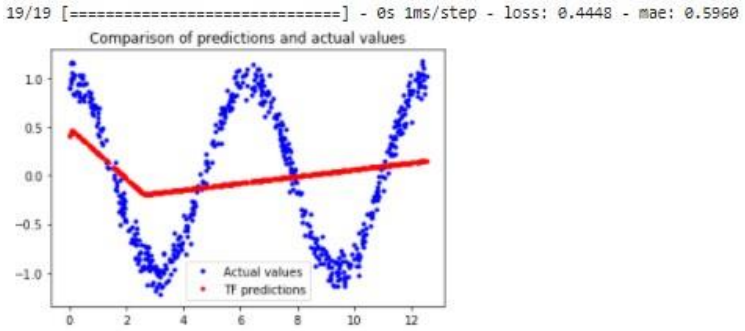
```
plt.plot(x_test, y_test_pred, 'r.', label='TF predictions')
plt.legend()
plt.show()
```

Hasil Running



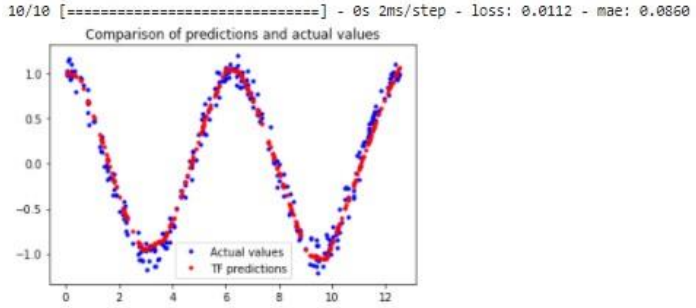
Hasil Analisa dan Kesimpulan.
Setelah melakukan simulasi untuk mentraining data yang dibagi menjadi 3 Skenario didapatkan hasil perbandingan nilai actual dan nilai prediksi yang dilakukan oleh *deep learning* yang menghasilkan bentuk grafik yang berbeda beda pada setiap skenario.

Hasil Skenario 1



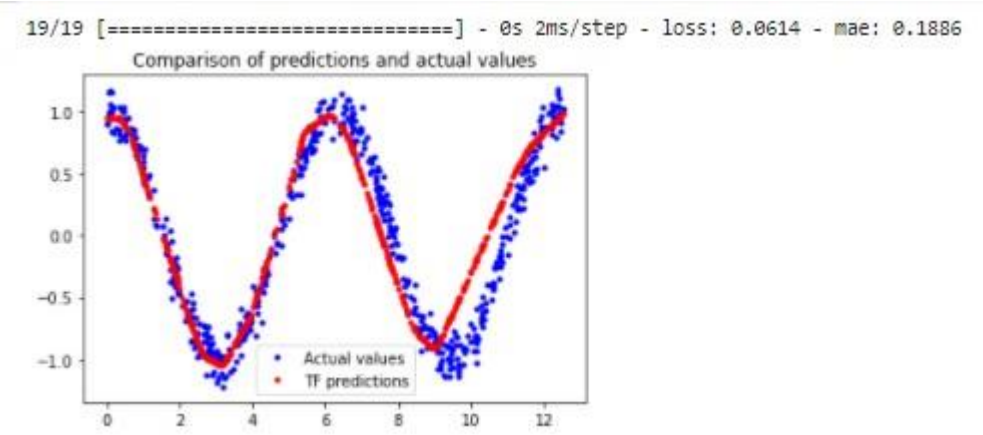
Dari bentuk grafik tersebut dapat dilihay bahwa nilai prediksi yang dilakukan oleh *deep learning* masih memiliki nilai yang jauh berbeda dengan hasil aktual yang diinginkan.

Hasil Skenario 2



Setelah itu skenario digunakan dengan menambahkan 4 hidden layer sehingga deep learning menggunakan 5 hidden layer dan juga 1 ouput layer, dengan jumlah neuron disetiap layer merupakan kelipatan dari nilai delapan, untuk optimizer serta activation masih menggunakan ‘adam’ dan ‘relu’ sama seperti pada skenario satu. Setelah mensimulasikan didapatkan hasil grafik yang menunjukan nilai actual dan nilai prekdiksi dari *deep learning* memiliki bentuk yang hampir sama sehingga dapat disimpulkan bahwa training yang didilakukan pada data random berhasil.

Hasil Skenario 3



Jumlah hidden layer serta jumlah neuron pada setiap hidden layer pada skenario tiga memiliki nilai yang sama dengan skenario dua, namun pada skenario dua menggunakan optimizer =’sgd’ untuk mengoptimalisasikan hasil prediksi dari *deep learning* yang telah dibuat. Hasil grafik menunjukan bahwa bentuk gelombang cosinus yang dihasilkan dari grafik actual dan prediksi memiliki bentuk yang kurang lebih sama namun ada sedikit bergeseran ke kiri dari grafik hasil prediksi.

Dapat disimpulkan bahwa jumlah hidden layer, neuron , optimizer dan activation pada setiap pemodelan keras berpengaruh pada grafik hasil prediksi yang dihasilkan yang berkaitan dengan keberhasilan *deep learning* melakukan training data random ataupun data set yang disediakan.