Tugas Sistem Cerdas Tensor Flow & Keras

Nama: Queen Hesti Ramadhamy

NIM: 2101201037

1. Memprediksi nilai dari fungsi sinus suatu bilangan yang dituliskan sebagai berikut: y=sin(x)

   Langkah-langkah sebagai berikut:

   - Langkah pertama adalah melakukan penginstalan dan pengimportan model serta library pada simulasi untuk membuat fungsi yang diinginkan dari suatu bilangan.

Code

```
[ ] 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'

    ! pip install tensorflow==2.4.0rc0

    import tensorflow as tf
    from tensorflow import keras
    import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import 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: protobuf~=3.13.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (3.13.0)
    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: keras-preprocessing~=1.1.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (1.1.2)
    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: absl-py~=0.10 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (0.12.0)
    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: tensorflow-estimator~=2.3.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (2.3.0)
    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: flatbuffers~=1.12.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (1.12)
    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: gast==0.3.3 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (0.3.3)
    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: wheel~=0.35 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (0.36.2)
    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: termcolor~=1.1.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (1.1.0)
    Requirement already satisfied: numpy~=1.19.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (1.19.5)
    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: six~=1.15.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4.0rc0) (1.15.0)
    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: 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: 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<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: 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: 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: 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: 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: 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: 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: 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: 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: 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: 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>=0.1.3 in /usr/local/lib/python3.7/dist-packages (from rsa<5,>=3.1.4; python_version >= "3.6"->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)
    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)
```

Catatan code

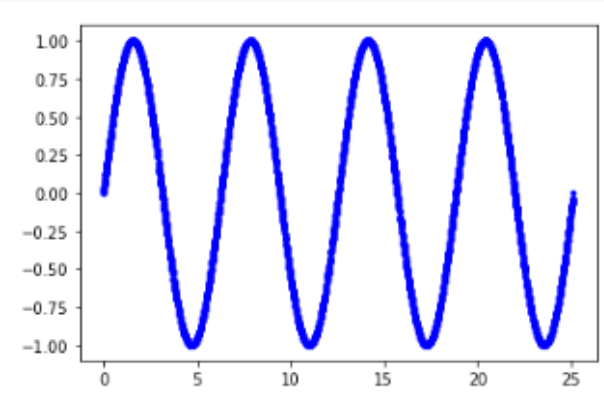
Pada code diatas, pada code '**import os**' merupakan penginstalan os dan memberi jalur pada model file yan akan dijalankan. Lalu, pada code '**! pip instalasi library tensor flow==2.4.0rc0**' merupakan penginstalan terhadap library tensor flow. Kemudian pada code '**import tensorflow as tf**' merupakan pengimportan terhadap library tensorflow yang digunakan sebagai deep learning dan pada code '**from tensorflow import keras**' merupakan pengimportan terhadap library keras yang ada pada tensorflow. Lalu pada code '**import numpy as np**' dan code "**import pandas as pd**' yang merupakan pengimportan terhadap library numpy yang bekerja pada scientific computing dan library pandas yang bekerja untuk menganalisis data yang cocok untuk dianalisis. Selanjutnya ada code untuk '**import matplotlib.pyplot as plt**' dan '**import math**' yang masing-masingnya merupakan pengimprotan terhadap library matplotlib dan library math.

- Langkah kedua adalah pembuatan angka yang menggunakan 5000 jumlah data untuk titik sample dan dilanjutkan dengan pemrosesan fungsi yang diinginkan pada bilangan tersebut dan pemplotingan yang berbentuk grafik.

Code

```
[ ]  seed = 1
     np.random.seed(seed)
     SAMPLES = 5000
     x_values = np.random.uniform(low=0, high=8*math.pi, size=SAMPLES).astype(np.float32)
     np.random.shuffle(x_values)
     y_values = np.sin(x_values).astype(np.float32)
     plt.plot(x_values, y_values, 'b.')
     plt.show()
```

Hasil running



Catatan code

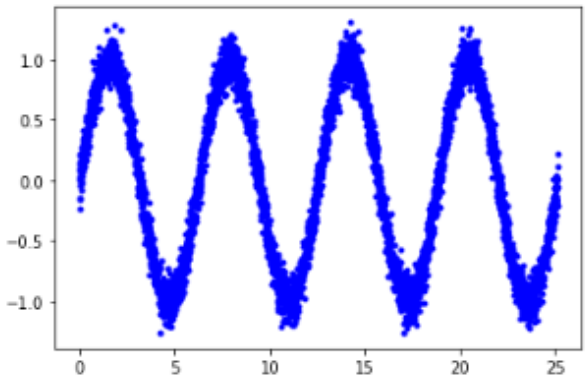
Pada code diatas, terdapat code '**np.random.seed(seed)**' yang digunakan untuk membuat angka yang dapat diprediksi secara acak. Terdapat juga code '**SAMPLES = 5000**' yang merupakan pendeklarasian jumlah data yang digunakan, jumlah ini dapat diubah sesuai dengan kebutuhan. Lalu terdapat juga code '**x_values = np.random.uniform(low=0, high=8*math.pi, size=SAMPLES).astype(np.float32)**' yang merupakan pemrosesan pengenerate an terhadap gelombang yang memiliki batas dari 0 sampai 4 phi, pada code ini dapat diubah-ubah sesuai dengan keinginan untuk melihat gelombang dari berapa sampai berapa. Kemudian terdapat code '**np.random.shuffle(x_values)**' yang merupakan proses shuffle untuk memastikan data tidak berurutan.Lalu code '**y_values = np.sin(x_values).astype(np.float32)**' merupakan perhitungan terhadap fungsi yang diinginkan, pada percobaan ini dilakukan fungsi sinus. Lalu yang selanjutnya adalah code '**plt.plot(x_values, y_values, 'b.')**' untuk memploting dalam bentuk grafik dari hasil yang didapat dari kalkulasi sebelumnya dan code '**plt.show()**' untuk menampilkan grafik dari plotingan tersebut.

- Lalu selanjutnya melakukan penambahan bilangan random pada fungsi y untuk mendapatkan hasil data fungsi sinus (pada percobaan ini) menjadi random dan menggunakan code deep learning yag distimulasikan untuk memprediksi bentuk tidak random dari data sinus dan melakukan pemplotingan terhadap hasil yang didapat.

Code

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

Hasil running



- Langkah selanjutnya adalah proses pembagian data set menjadi tiga bagian yaitu data train, data test, dan data validasi. Data train disini merupakan data yang akan digunakan untuk melatih algoritma yang akan digunakan, lalu data test merupakan data yang digunakan untuk mengetahui performa algoritma yang dilatih pada data train dan jika terdapat data baru didalamnya akan diubah menjadi lebih kecil dari sebelumnya untuk dilakukan proses pelatihan. Dan set terakhir adalah data validasi yang digunakan untuk proses validasi model untuk mencegah overfitting.

Code

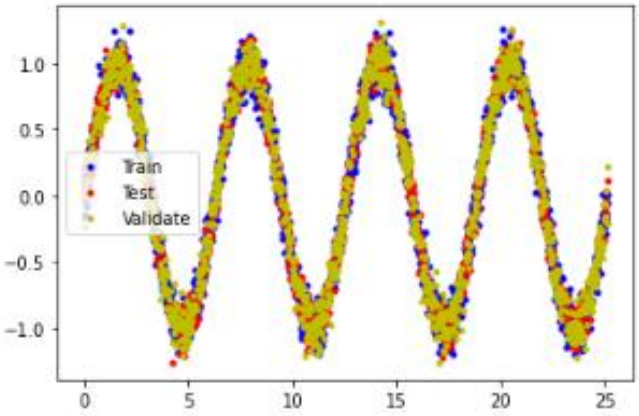
```
[ ]  TRAIN_SPLIT =  int(0.5 * SAMPLES)
     TEST_SPLIT = int(0.2 * SAMPLES + TRAIN_SPLIT)

     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])

     assert (x_train.size + x_validate.size + x_test.size) ==  SAMPLES

     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



Catatan code

Pada code '**TRAIN_SPLIT = int(0.5 * SAMPLES)**' dan '**TEST_SPLIT = int(0.2 * SAMPLES + TRAIN_SPLIT)**' merupakan pendeklarasian terhadap pembagian data set. Pada percobaan ini, perbandingan yang dilakukan adalah 0.5:0.2:0.3 yang dapat dituliskan juga sebagai data set yang digunakan adalah 20%, data train yang digunakan 50% dan 30% untuk data validasi. Kemudian terdapat code '**x_train, x_test, x_validate = np.split(x_values, [TRAIN_SPLIT, TEST_SPLIT])**' dan '**y_train, y_test, y_validate = np.split(y_values, [TRAIN_SPLIT, TEST_SPLIT])**' yang digunakan untuk membagi dataset menjadi 3 bagian sesuai dengan deklarasi sebelumnya. Lalu terdapat pula code '**assert (x_train.size + x_validate.size + x_test.size) == SAMPLES**' yang merupakan pemeriksaan terhadap kesesuaian data yang telah dibagi sebelumnya. Lalu code '**plt.plot(x_train, y_train, 'b.', label="Train")**', '**plt.plot(x_test, y_test, 'r.', label="Test")**', dan '**plt.plot(x_validate, y_validate, 'y.', label="Validate")**' digunakan untuk memploting hasil data dengan menggunakan warna yang berbeda yang ditunjukkan dengan 'r.' untuk test,'y.' untuk validate, dan 'b.' untuk train.

- Langkah selanjutnya adalah deep learning.
  - ***Skenario Satu***

    Langkah selanjutnya adalah membuat skenario. Untuk ***Skenario Satu***, dilakukan pemodelan 'keras' untuk deep learning dan juga melatih data yang telah dibagi seblumnya. Jumlah layer dan jumlah neuron dapat disesuaikan dengan yang dibutuhkan. Setelahnya dapat dilakukan konfigurasi pelatihan lalu data train akan dilakukan.

    Code

```
[ ]  model_1 = tf.keras.Sequential()
     model_1.add(keras.layers.Dense(10, activation='relu', input_shape=(1,)))
     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=500, batch_size=64,
                             validation_data=(x_validate, y_validate))
```

    Hasil running

```
Epoch 1/500
40/40 [==============================] - 1s 6ms/step - loss: 1.1703 - mae: 0.8781 - val_loss: 0.5404 - val_mae: 0.6440
Epoch 2/500
40/40 [==============================] - 0s 3ms/step - loss: 0.5131 - mae: 0.6338 - val_loss: 0.5139 - val_mae: 0.6403
Epoch 3/500
40/40 [==============================] - 0s 2ms/step - loss: 0.5010 - mae: 0.6287 - val_loss: 0.5111 - val_mae: 0.6385
Epoch 4/500
40/40 [==============================] - 0s 2ms/step - loss: 0.5041 - mae: 0.6315 - val_loss: 0.5100 - val_mae: 0.6358
Epoch 5/500
40/40 [==============================] - 0s 2ms/step - loss: 0.5075 - mae: 0.6348 - val_loss: 0.5076 - val_mae: 0.6340
Epoch 6/500
40/40 [==============================] - 0s 2ms/step - loss: 0.4967 - mae: 0.6284 - val_loss: 0.5063 - val_mae: 0.6346
Epoch 7/500
40/40 [==============================] - 0s 2ms/step - loss: 0.4924 - mae: 0.6234 - val_loss: 0.5046 - val_mae: 0.6329
Epoch 8/500
40/40 [==============================] - 0s 2ms/step - loss: 0.4987 - mae: 0.6294 - val_loss: 0.5031 - val_mae: 0.6305
Epoch 9/500
40/40 [==============================] - 0s 3ms/step - loss: 0.4986 - mae: 0.6267 - val_loss: 0.5066 - val_mae: 0.6286
Epoch 10/500
40/40 [==============================] - 0s 3ms/step - loss: 0.4904 - mae: 0.6190 - val_loss: 0.5027 - val_mae: 0.6281
Epoch 11/500
40/40 [==============================] - 0s 3ms/step - loss: 0.4907 - mae: 0.6194 - val_loss: 0.5017 - val_mae: 0.6278
Epoch 12/500
40/40 [==============================] - 0s 2ms/step - loss: 0.4920 - mae: 0.6226 - val_loss: 0.5024 - val_mae: 0.6283
Epoch 13/500
40/40 [==============================] - 0s 2ms/step - loss: 0.4821 - mae: 0.6135 - val_loss: 0.5034 - val_mae: 0.6257
Epoch 14/500
40/40 [==============================] - 0s 2ms/step - loss: 0.4867 - mae: 0.6168 - val_loss: 0.5044 - val_mae: 0.6253
Epoch 15/500
40/40 [==============================] - 0s 2ms/step - loss: 0.4856 - mae: 0.6127 - val_loss: 0.5023 - val_mae: 0.6272
Epoch 16/500
40/40 [==============================] - 0s 2ms/step - loss: 0.4939 - mae: 0.6247 - val_loss: 0.5054 - val_mae: 0.6291
Epoch 17/500
40/40 [==============================] - 0s 3ms/step - loss: 0.4991 - mae: 0.6250 - val_loss: 0.5047 - val_mae: 0.6284
Epoch 18/500
40/40 [==============================] - 0s 2ms/step - loss: 0.5127 - mae: 0.6337 - val_loss: 0.5080 - val_mae: 0.6301
Epoch 19/500
40/40 [==============================] - 0s 2ms/step - loss: 0.4926 - mae: 0.6201 - val_loss: 0.5019 - val_mae: 0.6259
Epoch 20/500
40/40 [==============================] - 0s 2ms/step - loss: 0.4889 - mae: 0.6190 - val_loss: 0.5058 - val_mae: 0.6242
```

Catatan code

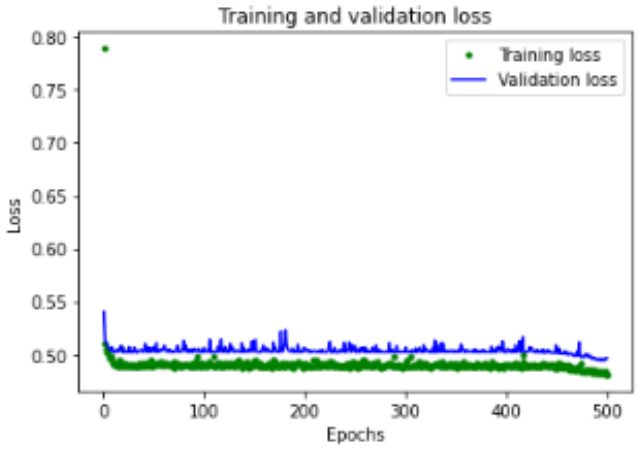
Pada code diatas, terdapat code '**model_1 = tf.keras.Sequential()**' yang merupakan pendeklarasian terhadap model yang digunakan, pada percobaan ini digunakan model sequential. Lalu terdapat pula code '**model_1.add(keras.layers.Dense(10, activation='relu', input_shape=(1,)))**' yang bertujuan untuk memberi tahu bahwa jumlah hidden layer yang sedang digunakan berjumlah 1 layer yang dimana hidden layer tersebut akan ditambahkan lagi menggunakan Dense. Sedangkan angka 10 menunjukkan neuron yang berada pada hidden layer, relu yang merupakan fungsi aktivasi dan angka 1 yang merupakan jumlah neuron pada bagian input. Lalu terdapat code '**model_1.compile(optimizer='adam', loss='mse', metrics=['mae'])**' yang digunakan untuk mengcompile data optimizer yang dimasukkan, pada percobaan ini digunakan optimizer adam. Lalu terdapat pula code '**history_1 = model_1.fit(x_train, y_train, epochs=500, batch_size=64, validation_data=(x_validate, y_validate))**', pada code ini terdapat epochs yang merupakan kondisi dimana seluruh dataset yang sudah melewati proses pelatihan pada neural network dan dikembalikan ke awal untuk satu kali putaran. Lalu pada code 'data_validasi', beberapa neuron yang menggunakan drop out tidak akan menjatuhkan neuron secara random dikarenakan selama pelatihan, penggunaan drop out dilakukan untuk menambah noise untuk menghindari pemasangan yang berlebihan.

Lalu selanjutnya akan menampilkan grafik error dari hasil pelatihan yang dilakukan.

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



Catatan code

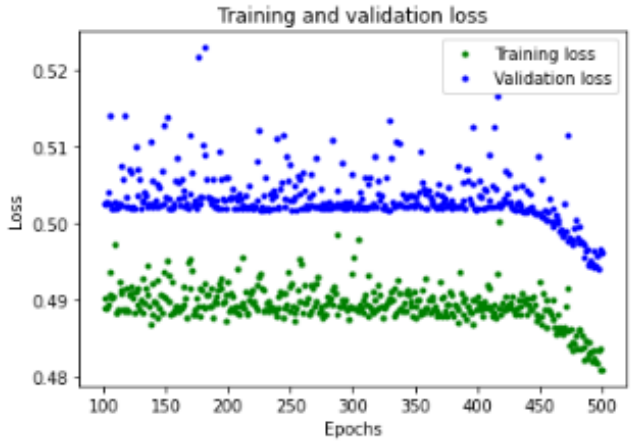
Pada code diatas terdapat val_loss yang merupakan nilai dari fungsi biaya untuk data validasi silang.

Kemudian menampilkan grafik error 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

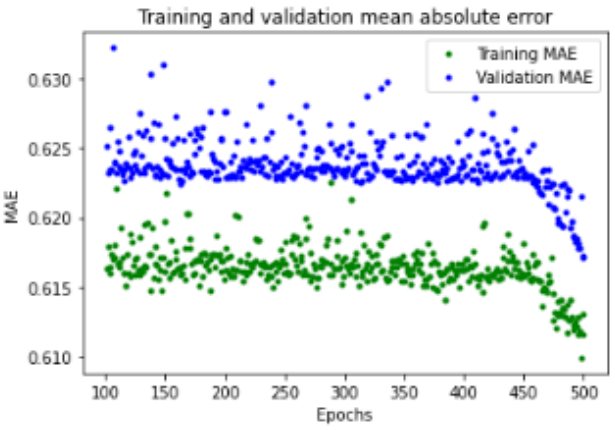


Selanjutnya, dilakukan kalkulasi pada jumlah error yang dihasilkan dan melihat bentuk grafik mean absolute error.

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



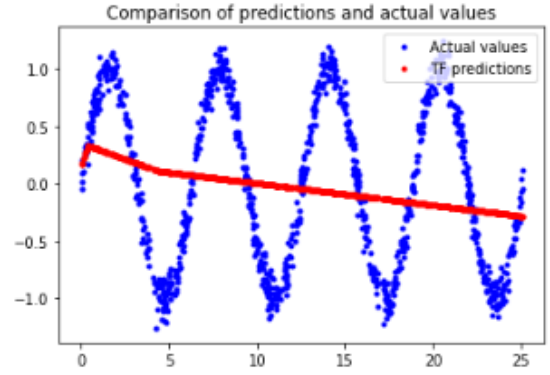
Lalu hasil perbandingan pada ***Skenario Satu*** menggunakan deep learning.

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

```
32/32 [==============================] - 0s 1ms/step - loss: 0.4713 - mae: 0.5983
```



– ***Skenario Dua***

Selanjutnya melakukan ***Skenario Dua*** yang melakukan hal yang sama dengan ***Skenario Satu***. Yang membedakannya adalah jumlah hidden layer yang digunakan.

Code

```
[81] model = tf.keras.Sequential()
     model.add(keras.layers.Dense(9, activation='relu', input_shape=(1,)))
     model.add(keras.layers.Dense(18, activation='relu'))
     model.add(keras.layers.Dense(27, activation='relu'))
     model.add(keras.layers.Dense(36, activation='relu'))
     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))
     model.save(MODEL_TF)
```

Hasil running

```
Epoch 1/400
40/40 [==============================] - 1s 7ms/step - loss: 0.5704 - mae: 0.6494 - val_loss: 0.5036 - val_mae: 0.6323
Epoch 2/400
40/40 [==============================] - 0s 3ms/step - loss: 0.4978 - mae: 0.6258 - val_loss: 0.5031 - val_mae: 0.6291
Epoch 3/400
40/40 [==============================] - 0s 3ms/step - loss: 0.4755 - mae: 0.6068 - val_loss: 0.4852 - val_mae: 0.6133
Epoch 4/400
40/40 [==============================] - 0s 3ms/step - loss: 0.4639 - mae: 0.5983 - val_loss: 0.4871 - val_mae: 0.6043
Epoch 5/400
40/40 [==============================] - 0s 3ms/step - loss: 0.4603 - mae: 0.5876 - val_loss: 0.4693 - val_mae: 0.5958
Epoch 6/400
40/40 [==============================] - 0s 3ms/step - loss: 0.4665 - mae: 0.5880 - val_loss: 0.4692 - val_mae: 0.5861
Epoch 7/400
40/40 [==============================] - 0s 3ms/step - loss: 0.4642 - mae: 0.5810 - val_loss: 0.4648 - val_mae: 0.5868
Epoch 8/400
40/40 [==============================] - 0s 3ms/step - loss: 0.4499 - mae: 0.5736 - val_loss: 0.4551 - val_mae: 0.5783
Epoch 9/400
40/40 [==============================] - 0s 4ms/step - loss: 0.4567 - mae: 0.5788 - val_loss: 0.4521 - val_mae: 0.5756
Epoch 10/400
40/40 [==============================] - 0s 3ms/step - loss: 0.4493 - mae: 0.5696 - val_loss: 0.4493 - val_mae: 0.5712
Epoch 11/400
40/40 [==============================] - 0s 3ms/step - loss: 0.4586 - mae: 0.5757 - val_loss: 0.4455 - val_mae: 0.5687
Epoch 12/400
40/40 [==============================] - 0s 3ms/step - loss: 0.4309 - mae: 0.5544 - val_loss: 0.4483 - val_mae: 0.5724
Epoch 13/400
40/40 [==============================] - 0s 3ms/step - loss: 0.4463 - mae: 0.5662 - val_loss: 0.4431 - val_mae: 0.5690
Epoch 14/400
40/40 [==============================] - 0s 3ms/step - loss: 0.4346 - mae: 0.5613 - val_loss: 0.4366 - val_mae: 0.5615
Epoch 15/400
40/40 [==============================] - 0s 3ms/step - loss: 0.4354 - mae: 0.5586 - val_loss: 0.4357 - val_mae: 0.5586
Epoch 16/400
40/40 [==============================] - 0s 3ms/step - loss: 0.4295 - mae: 0.5522 - val_loss: 0.4270 - val_mae: 0.5556
Epoch 17/400
40/40 [==============================] - 0s 3ms/step - loss: 0.4131 - mae: 0.5417 - val_loss: 0.4220 - val_mae: 0.5505
Epoch 18/400
40/40 [==============================] - 0s 3ms/step - loss: 0.4102 - mae: 0.5363 - val_loss: 0.4131 - val_mae: 0.5438
Epoch 19/400
40/40 [==============================] - 0s 3ms/step - loss: 0.3995 - mae: 0.5287 - val_loss: 0.4092 - val_mae: 0.5400
Epoch 20/400
40/40 [==============================] - 0s 3ms/step - loss: 0.3971 - mae: 0.5264 - val_loss: 0.4004 - val_mae: 0.5319
```

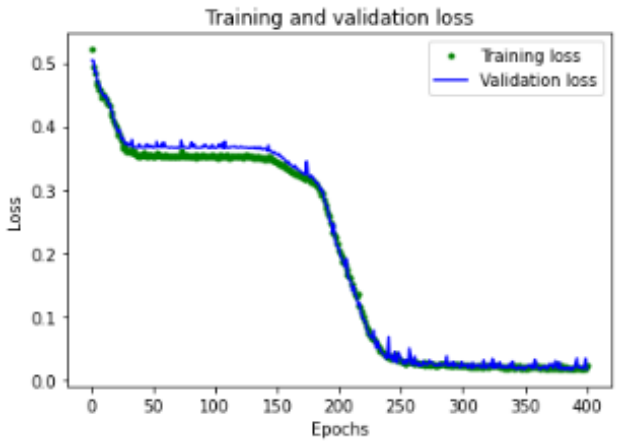
Catatan code

Pada code diatas, terdapat code '**model.add(keras.layers.Dense(9, activation='relu', input_shape=(1,)))**', '**model.add(keras.layers.Dense(18, activation='relu'))**', '**model.add(keras.layers.Dense(27, activation='relu'))**', dan '**model.add(keras.layers.Dense(36, activation='relu'))**' yang menunjukkan hidden layer yang berjumlah 4 layer yang dimana hidden layer tersebut akan ditambahkan lagi menggunakan Dense. Sedangkan angka 9, 18, 27, dan 36 menunjukkan neuron yang berada pada hidden layer, relu yang merupakan fungsi aktivasi dan angka 1 yang merupakan jumlah neuron pada bagian input.

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

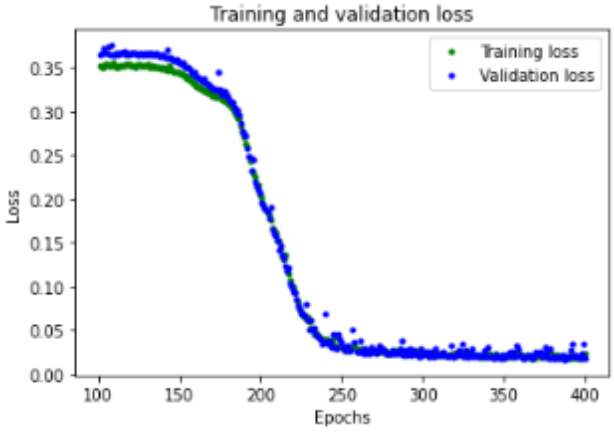


Kemudian menampilkan grafik error dengan jumlah skip 100

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

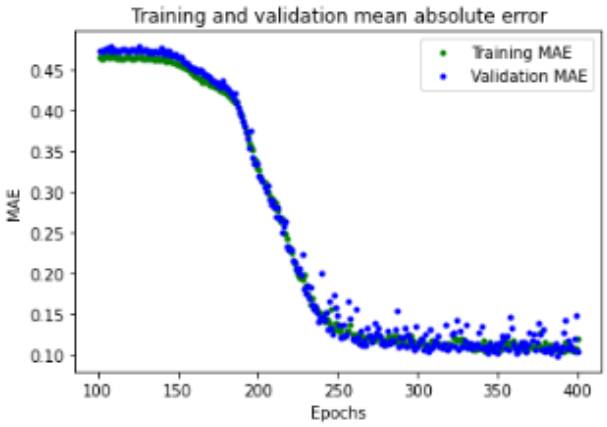


Lalu memploting kembali hasil dari perbandingan kalkulasi error dari sisi mean absolute error

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



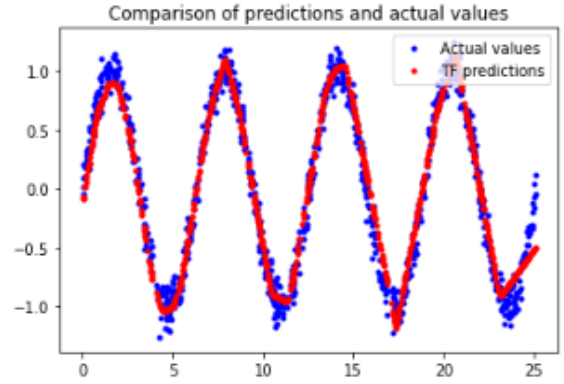
Lalu, didapatkan hasil deep learning untuk ***Skenario Dua***.

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

```
32/32 [==============================] - 0s 1ms/step - loss: 0.0165 - mae: 0.1003
```

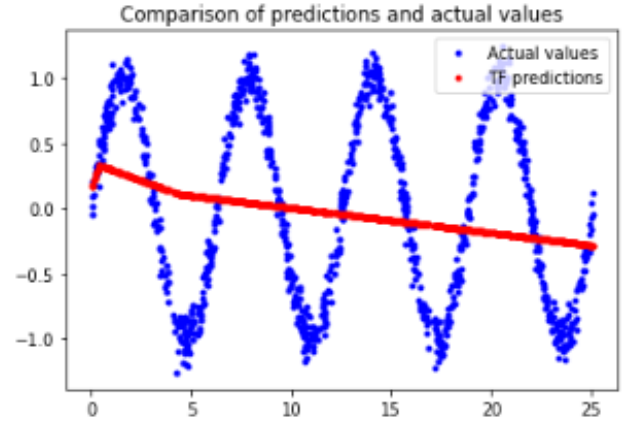


Hasil analisis dan kesimpulan.

Setelah melakukan percobaan diatas dengan 3 skenario didapatkanlah hasil nilai yang dilakukan oleh deep learning yang menhasilkan bentuk grafik yang berbeda pada setiap skenarionya.

Hasil skenario 1:

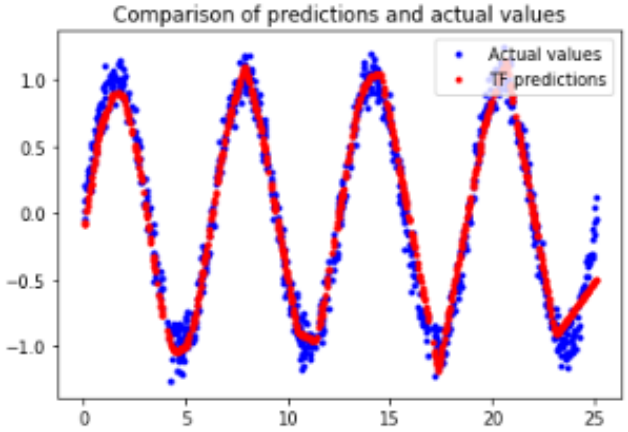
```
32/32 [==============================] - 0s 1ms/step - loss: 0.4713 - mae: 0.5983
```



Dari grafik diatas dapat disimpulkan bahwa nilai yang didapat oleh deep learning masih berbeda jauh dengan hasil yang diinginkan.

Hasil skenario 2:

```
32/32 [==============================] - 0s 1ms/step - loss: 0.0165 - mae: 0.1003
```



Dari hasil grafik diatas didapat nilai oleh deep learning yang hampir sama (walaupun masih harus dilanjutkan untuk mendapat hasil yang sama) dengan hasil yang diinginkan. Dengan demikian dapat disimpulkan bahwa jumlah hidden layer, neuron, dll pada tahap skenario dua berpengaruh pada grafik hasil yang didapat dan juga berkaitan dengan keakuratan serta keberhasilan deep learning saat melakukan pelatihan data random yang ada.