Nama: Yogantana Arum Panganti

NIM: 2101201043

Tugas: Tutorial TensorFlow dan Keras untuk aplikasi Deep-Learning menggunakan Google Colab

Link: https://github.com/yogantana/Tugas_SistCerdas_Tutorial-TensorFlow-dan-Keras-_2101201043_Yogantana-Arum-Panganti

Memprediksi nilai dari fungsi y=sin(x) dan z=cos(x).

Tutorial:

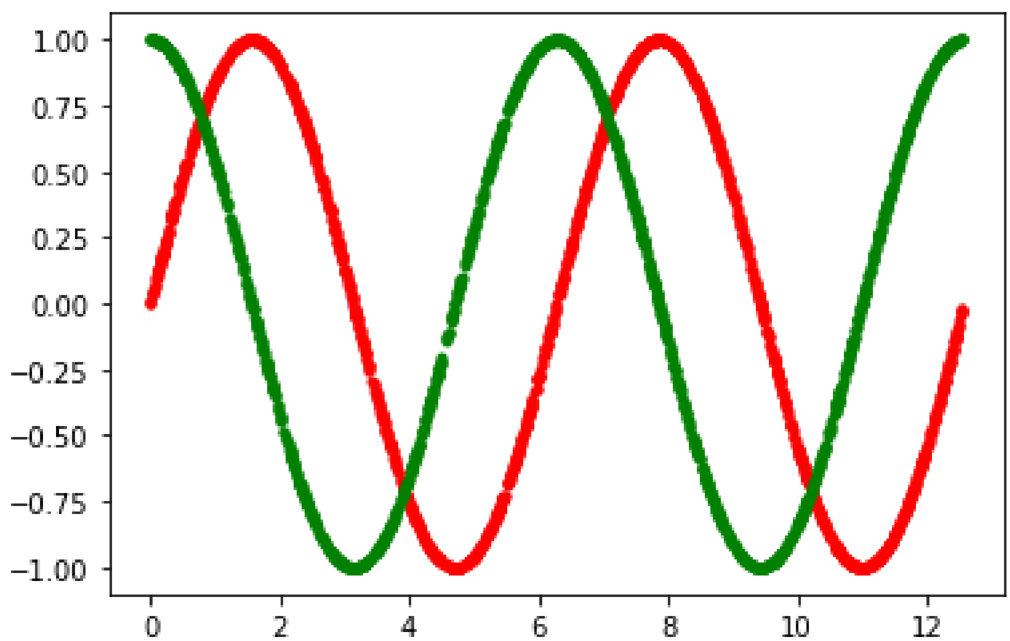
1) Melakukan instalasi dan import model serta library yang akan digunakan pada server google colab.

```
# Define paths to 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'

! pip install tensorflow==2.4.0rc0


import tensorflow as tf

# Keras is TensorFlow's high-level API for deep learning
from tensorflow import keras
# Numpy is a math library
import numpy as np
# Pandas is a data manipulation library
import pandas as pd
# Matplotlib is a graphing library
import matplotlib.pyplot as plt
# Math is Python's math library
import math
```

```
# Set seed for experiment reproducibility
seed = 1
np.random.seed(seed)
```

2) Tahap selanjutnya, membuat fungsi dengan data sin(x) dan cos(2x) dengan titik sample sejumlah 2000 titik. #note: semakin banyak jumlah sampel, maka garis akan semakin jelas.

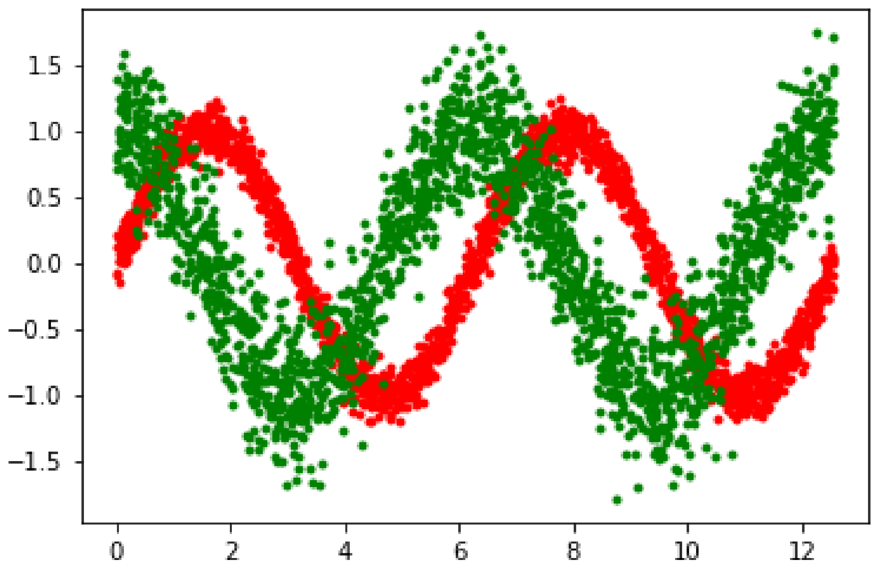
```
SAMPLES = 2000
# Generate random number
x_values = np.random.uniform(low=0, high=4*math.pi,
size=SAMPLES).astype(np.float32)
# Melakukan proses shuffle agar data yang di generate tidak
berurutan
np.random.shuffle(x_values)
# Melakukan pergitungan fungsi
y_values = np.sin(x_values).astype(np.float32)#fungsi sin(x)
z_values = np.cos(x_values).astype(np.float32)#fungsi cos(2x)
# Menampilkan plot data
plt.plot(x_values, y_values, 'r.')#data sin(2x) dengan garis merah
plt.plot(x_values, z_values, 'g.')#data cos(2x) dengan garis hijau
plt.show()
```



3) Melakukan penambahan bilangan secara random agar fungsi menjadi kotor agar Deep-Learning memprediksi hasil fungsi bersih.

```
y_values += 0.1 * np.random.randn(*y_values.shape)
z_values += 0.3 * np.random.randn(*z_values.shape)
plt.plot(x_values, y_values, 'r.') #data sin(2x) dengan garis merah
plt.plot(x_values, z_values, 'g.') #data cos(2x) dengan garis hijau
plt.show()
```

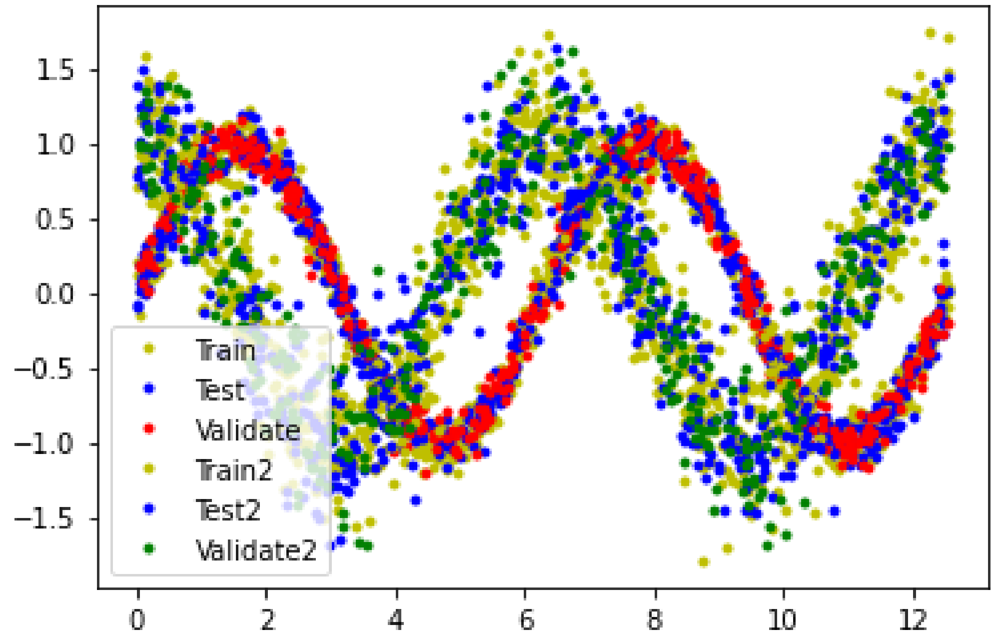
#note: penambahan 0.xx membuat titik sample semakin menyebar. Sebagai contoh penambahan 0.3(hijau) membuat titik semakin tersebar dibanding dengan penambahan 0.1(merah) dimana titik sampel saling menempel.



4) Membuat pembagian data set menjadi tiga bagian, yaitu data train, data test, dan data validasi. Pembagian data dilakukan dengan perbandingan yang dapat diatur atau disesuaikan. #note: pada code dibawah menggunakan perbantdingan 50:30:20.

```
TRAIN_SPLIT =  int(0.5 * SAMPLES)
TEST_SPLIT = int(0.3 * SAMPLES + TRAIN_SPLIT)
#Membagi dataset menjadi tiga bagian
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])
z_train, z_test, z_validate = np.split(z_values, [TRAIN_SPLIT, TEST_SPLIT])
# Memeriksa kesesuaian data yang telah dibagi
assert (x_train.size + x_validate.size + x_test.size) ==  SAMPLES
# Plot data dengan warna dan label yang berbeda
```

```
plt.plot(x_train, y_train,  'y.', label="Train")
plt.plot(x_test, y_test, 'b.', label="Test")
plt.plot(x_validate, y_validate, 'r.', label="Validate")
plt.plot(x_train, z_train, 'y.', label="Train2")
plt.plot(x_test, z_test, 'b.', label="Test2")
plt.plot(x_validate, z_validate, 'g.', label="Validate2")
plt.legend()
plt.show()
```



5) Tahap DEEP LEARNING

a) Percobaan/Skenario Satu

Membuat pemodelan deep-learning “Keras” serta men-training data yang sudah dibagi sebelumnya. Model yang digunakan adalah pemodelan sequential dimana hidden layer pertama dengan 10 neuron dan final layer dengan single neuron.

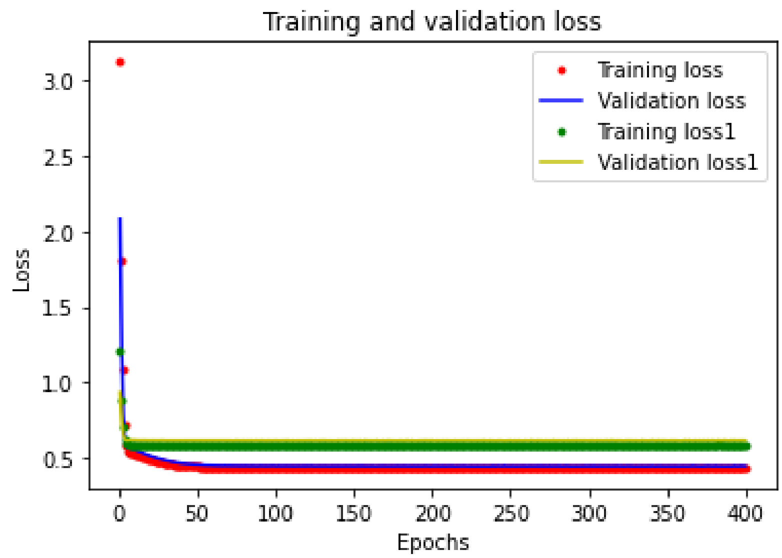
```
model = tf.keras.Sequential()
model1 = tf.keras.Sequential()
model.add(keras.layers.Dense(10, activation='relu', input_shape=(1,)))
model1.add(keras.layers.Dense(10, activation='relu', input_shape=(1,)))
model.add(keras.layers.Dense(1))
model1.add(keras.layers.Dense(1))
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
model1.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))
```

```
history1 = model1.fit(x_train, z_train, epochs=400, batch_size=64,
validation_data=(x_validate, z_validate))
```

```
Epoch 56/400
16/16 [==============================] - 0s 3ms/step - loss: 0.6186 - mae: 0.6763 - val_loss: 0.5982 - val_mae: 0.6713
Epoch 57/400
16/16 [==============================] - 0s 3ms/step - loss: 0.6136 - mae: 0.6745 - val_loss: 0.5976 - val_mae: 0.6711
Epoch 58/400
16/16 [==============================] - 0s 4ms/step - loss: 0.5979 - mae: 0.6642 - val_loss: 0.5976 - val_mae: 0.6710
Epoch 59/400
16/16 [==============================] - 0s 3ms/step - loss: 0.6216 - mae: 0.6845 - val_loss: 0.5967 - val_mae: 0.6708
Epoch 60/400
16/16 [==============================] - 0s 3ms/step - loss: 0.6080 - mae: 0.6724 - val_loss: 0.5964 - val_mae: 0.6707
Epoch 61/400
16/16 [==============================] - 0s 3ms/step - loss: 0.6057 - mae: 0.6711 - val_loss: 0.5955 - val_mae: 0.6705
Epoch 62/400
16/16 [==============================] - 0s 3ms/step - loss: 0.6245 - mae: 0.6853 - val_loss: 0.5954 - val_mae: 0.6704
Epoch 63/400
16/16 [==============================] - 0s 3ms/step - loss: 0.6375 - mae: 0.6887 - val_loss: 0.5952 - val_mae: 0.6703
Epoch 64/400
16/16 [==============================] - 0s 4ms/step - loss: 0.6070 - mae: 0.6772 - val_loss: 0.5946 - val_mae: 0.6702
Epoch 65/400
16/16 [==============================] - 0s 3ms/step - loss: 0.6088 - mae: 0.6708 - val_loss: 0.5943 - val_mae: 0.6700
Epoch 66/400
16/16 [==============================] - 0s 4ms/step - loss: 0.6059 - mae: 0.6695 - val_loss: 0.5939 - val_mae: 0.6699
Epoch 67/400
```

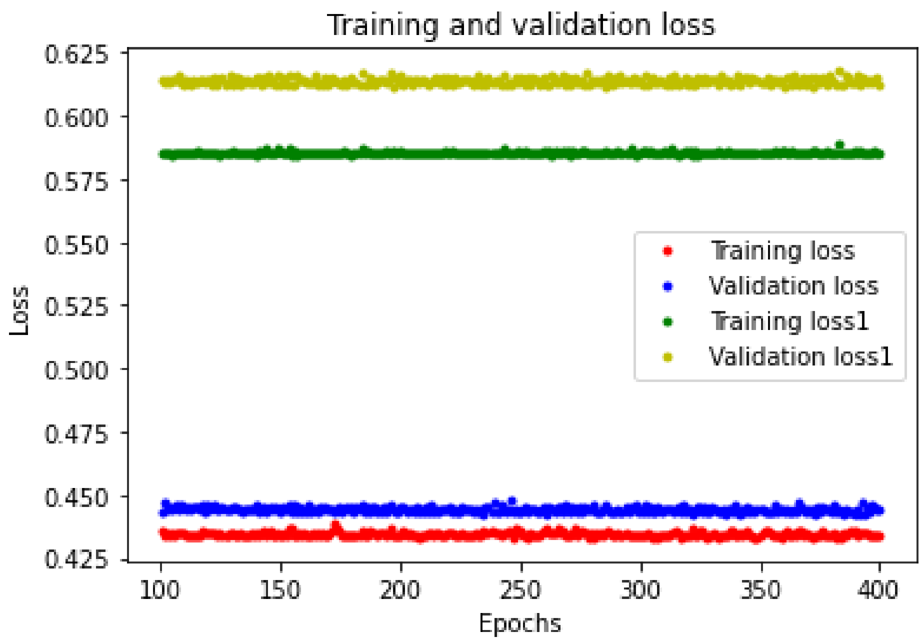
Melihat grafik error hasil training

```
train_loss = history.history['loss']
val_loss = history.history['val_loss']
train_loss1 = history1.history['loss']
val_loss1 = history1.history['val_loss']
epochs = range(1, len(train_loss) + 1)
epochs1 = range(1, len(train_loss1) + 1)
plt.plot(epochs, train_loss, 'r.', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.plot(epochs1, train_loss1, 'g.', label='Training loss1')
plt.plot(epochs1, val_loss1, 'y', label='Validation loss1')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



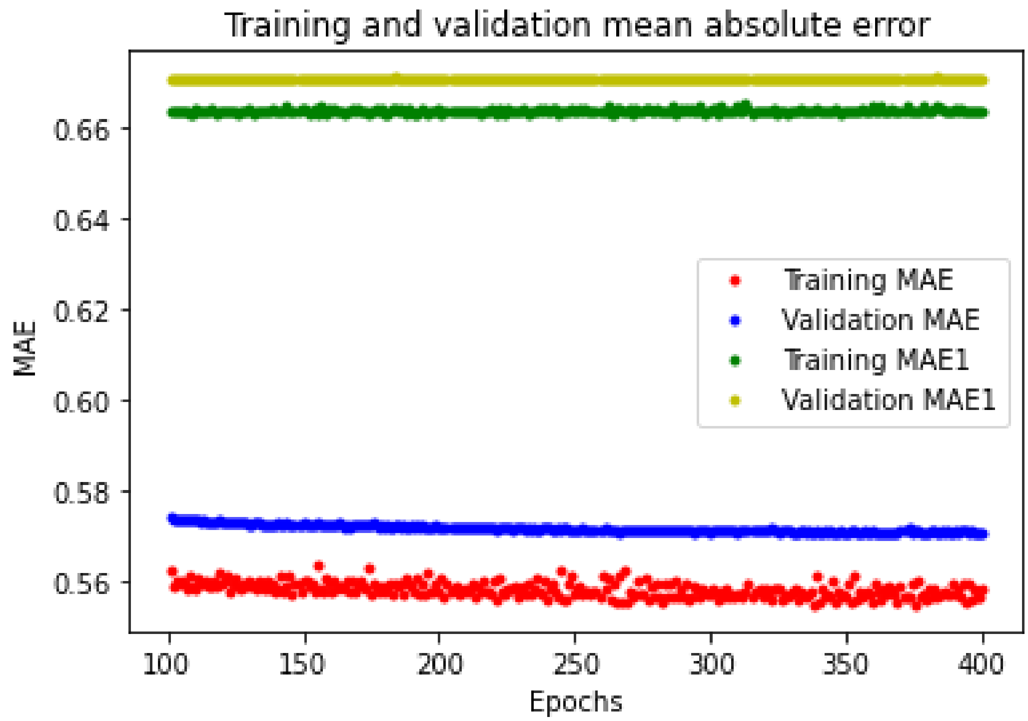
Melihat hasil training dari sisi lain

```
SKIP = 100
plt.plot(epochs[SKIP:], train_loss[SKIP:], 'r.', label='Training
loss')
plt.plot(epochs[SKIP:], val_loss[SKIP:], 'b.', label='Validation
loss')
plt.plot(epochs1[SKIP:], train_loss1[SKIP:], 'g.', label='Training
loss1')
plt.plot(epochs1[SKIP:], val_loss1[SKIP:], 'y.', label='Validation
loss1')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



Melihat laporan training dari sisi Mean Absolute Error

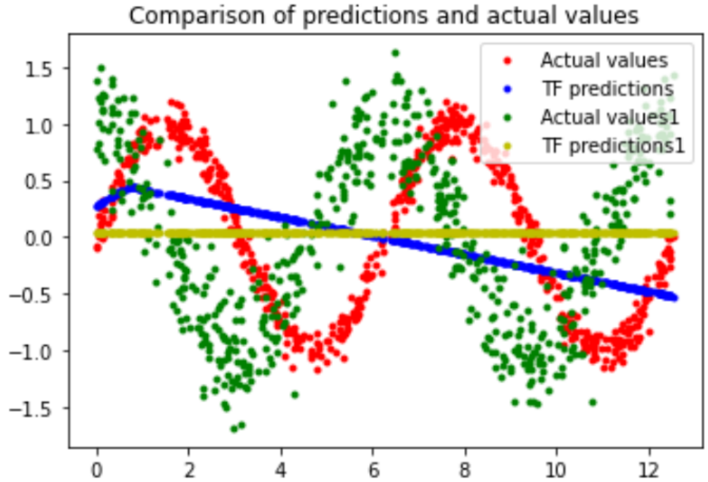
```
plt.clf()
train_mae = history.history['mae']
val_mae = history.history['val_mae']
train_mae1 = history1.history['mae']
val_mae1 = history1.history['val_mae']
plt.plot(epochs[SKIP:], train_mae[SKIP:], 'r.', label='Training
MAE')
plt.plot(epochs[SKIP:], val_mae[SKIP:], 'b.', label='Validation
MAE')
plt.plot(epochs1[SKIP:], train_mae1[SKIP:], 'g.', label='Training
MAE1')
plt.plot(epochs1[SKIP:], val_mae1[SKIP:], 'y.', label='Validation
MAE1')
plt.title('Training and validation mean absolute error')
plt.xlabel('Epochs')
plt.ylabel('MAE')
plt.legend()
plt.show()
```



Evaluasi selisih training

```
test_loss, test_mae = model.evaluate(x_test, y_test)
test_loss1, test_mae1 = model1.evaluate(x_test, z_test)
y_test_pred = model.predict(x_test)
z_test_pred = model1.predict(x_test)
plt.clf()
plt.title('Comparison of predictions and actual values')
plt.plot(x_test, y_test, 'r.', label='Actual values')
plt.plot(x_test, y_test_pred, 'b.', label='TF predictions')
plt.plot(x_test, z_test, 'g.', label='Actual values1')
plt.plot(x_test, z_test_pred, 'y.', label='TF predictions1')
plt.legend()
plt.show()
```

```
19/19 [==============================] - 0s 1ms/step - loss: 0.4208 - mae: 0.5553
19/19 [==============================] - 0s 978us/step - loss: 0.6133 - mae: 0.6793
```



b) Percobaan/Skenario Dua

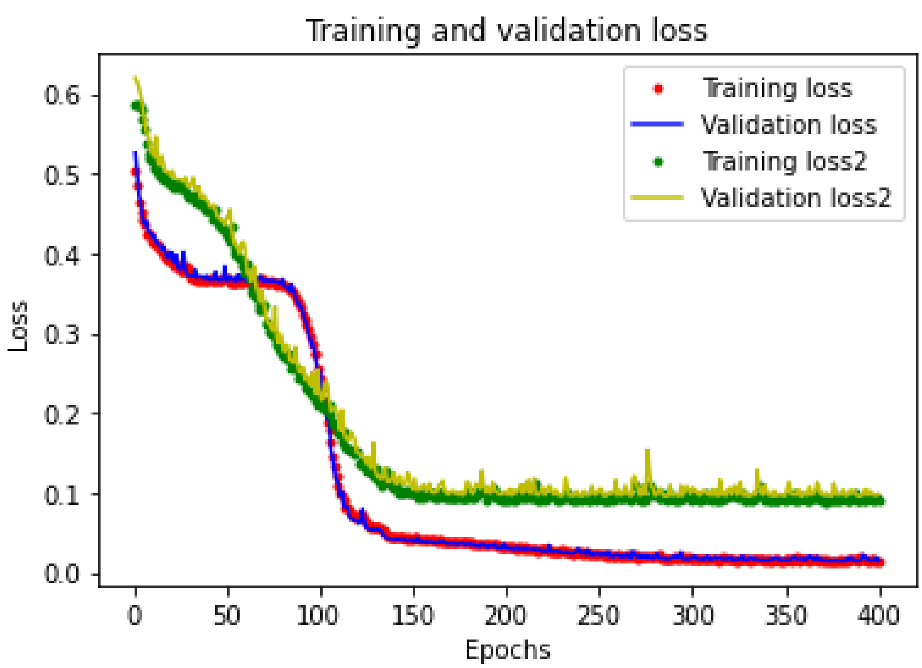
Mengubah mode “keras”, misalnya dengan menambah layer atau node dan melakukan training ulang.

```
model_1 = tf.keras.Sequential()
model_2 = tf.keras.Sequential()
model_1.add(keras.layers.Dense(8, activation='relu', input_shape=(1,)))
model_2.add(keras.layers.Dense(8, activation='relu', input_shape=(1,)))
model_1.add(keras.layers.Dense(16, activation='relu'))
model_2.add(keras.layers.Dense(16, activation='relu'))
model_1.add(keras.layers.Dense(24, activation='relu'))
model_2.add(keras.layers.Dense(24, activation='relu'))
model_1.add(keras.layers.Dense(32, activation='relu'))
model_2.add(keras.layers.Dense(32, activation='relu'))
model_1.add(keras.layers.Dense(1))
model_2.add(keras.layers.Dense(1))
model_1.compile(optimizer='adam', loss='mse', metrics=['mae'])
model_2.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))
history_2 = model_2.fit(x_train, z_train, epochs=400,
batch_size=64,validation_data=(x_validate, z_validate))
model_1.save(MODEL_TF)
model_2.save(MODEL_TF)
```

```
Epoch 379/400
16/16 [==============================] - 0s 3ms/step - loss: 0.5615 - mae: 0.6379 - val_loss: 0.5179 - val_mae: 0.6105
Epoch 380/400
16/16 [==============================] - 0s 4ms/step - loss: 0.5254 - mae: 0.6146 - val_loss: 0.5158 - val_mae: 0.6096
Epoch 381/400
16/16 [==============================] - 0s 4ms/step - loss: 0.5323 - mae: 0.6145 - val_loss: 0.5183 - val_mae: 0.6106
Epoch 382/400
16/16 [==============================] - 0s 4ms/step - loss: 0.5349 - mae: 0.6182 - val_loss: 0.5164 - val_mae: 0.6101
Epoch 383/400
16/16 [==============================] - 0s 4ms/step - loss: 0.5281 - mae: 0.6108 - val_loss: 0.5167 - val_mae: 0.6100
Epoch 384/400
```

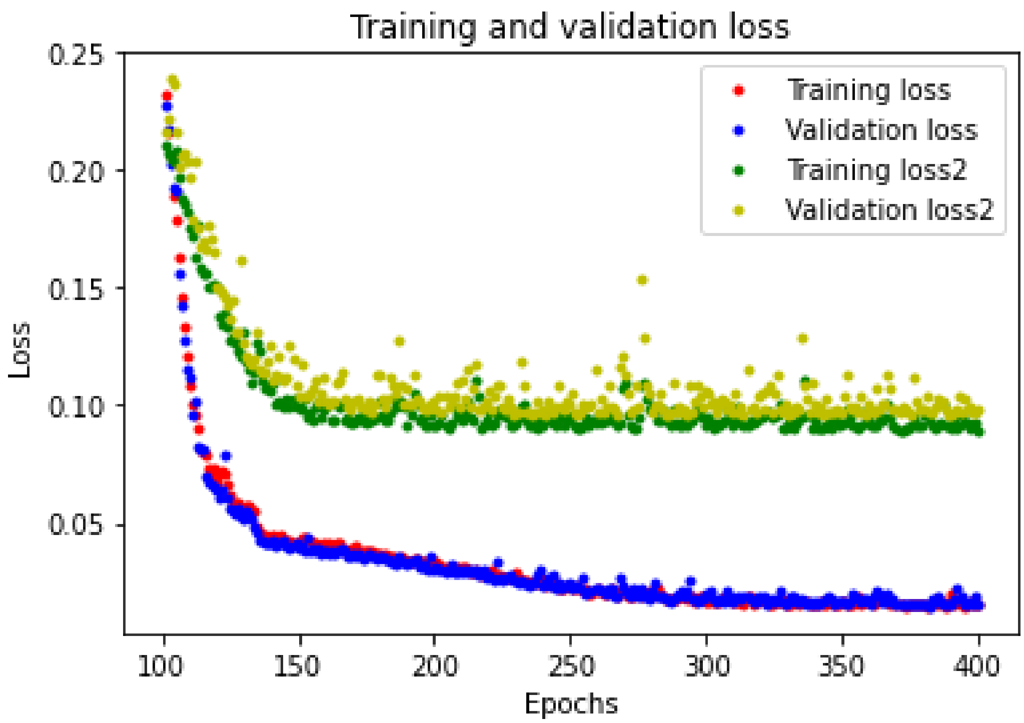
Melihat grafik error hasil training

```
train_loss = history_1.history['loss']
val_loss = history_1.history['val_loss']
train_loss2 = history_2.history['loss']
val_loss2 = history_2.history['val_loss']
epochs = range(1, len(train_loss) + 1)
epochs2 = range(1, len(train_loss2) + 1)
plt.plot(epochs, train_loss, 'r.', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.plot(epochs2, train_loss2, 'g.', label='Training loss2')
plt.plot(epochs2, val_loss2, 'y', label='Validation loss2')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



Melihat hasil training dari sisi lain

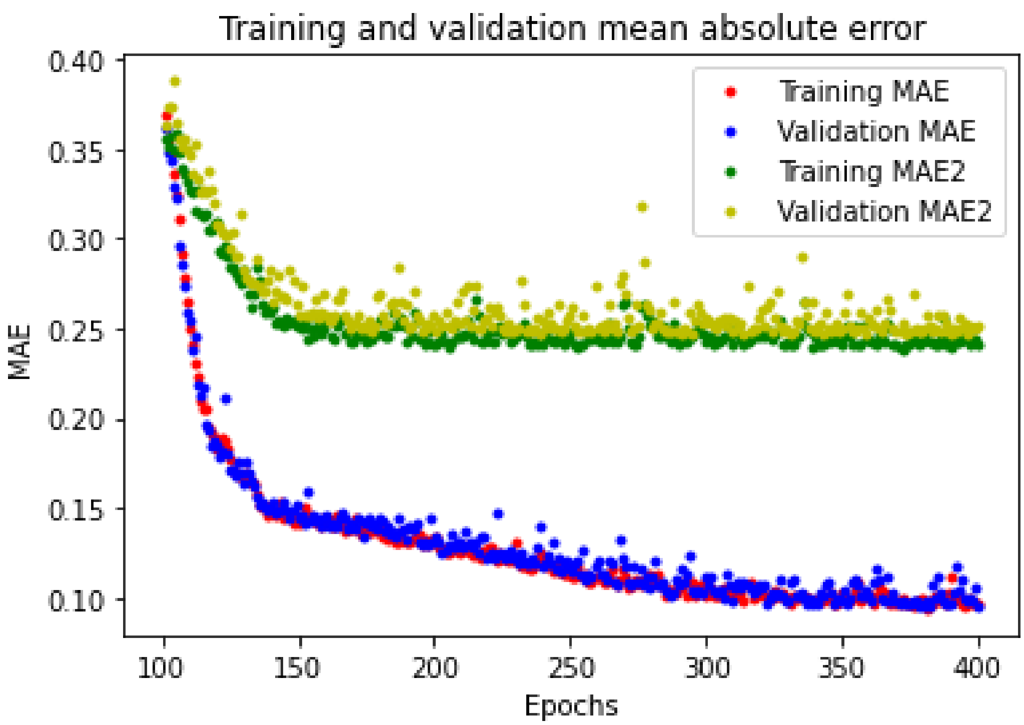
```
SKIP = 100
plt.plot(epochs[SKIP:], train_loss[SKIP:], 'r.', label='Training
loss')
plt.plot(epochs[SKIP:], val_loss[SKIP:], 'b.', label='Validation
loss')
plt.plot(epochs2[SKIP:], train_loss2[SKIP:], 'g.', label='Training
loss2')
plt.plot(epochs2[SKIP:], val_loss2[SKIP:], 'y.', label='Validation
loss2')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



Melihat laporan training dari sisi Mean Absolute Error

```
plt.clf()
train_mae = history_1.history['mae']
val_mae = history_1.history['val_mae']
train_mae2 = history_2.history['mae']
val_mae2 = history_2.history['val_mae']
```

```
plt.plot(epochs[SKIP:], train_mae[SKIP:], 'r.', label='Training
MAE')
plt.plot(epochs[SKIP:], val_mae[SKIP:], 'b.', label='Validation
MAE')
plt.plot(epochs2[SKIP:], train_mae2[SKIP:], 'g.', label='Training
MAE2')
plt.plot(epochs2[SKIP:], val_mae2[SKIP:], 'y.', label='Validation
MAE2')
plt.title('Training and validation mean absolute error')
plt.xlabel('Epochs')
plt.ylabel('MAE')
plt.legend()
plt.show()
```

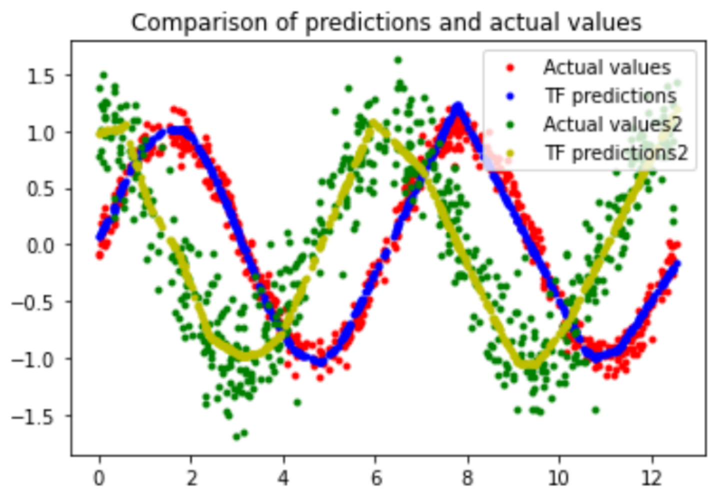


Evaluasi selisih training dengan scenario 2

```
test_loss, test_mae = model_1.evaluate(x_test, y_test)
test_loss2, test_mae2 = model_2.evaluate(x_test, z_test)
y_test_pred = model_1.predict(x_test)
z_test_pred = model_2.predict(x_test)
plt.clf()
plt.title('Comparison of predictions and actual values')
plt.plot(x_test, y_test, 'r.', label='Actual values')
```

```
plt.plot(x_test, y_test_pred, 'b.', label='TF predictions')
plt.plot(x_test, z_test, 'g.', label='Actual values2')
plt.plot(x_test, z_test_pred, 'y.', label='TF predictions2')
plt.legend()
plt.show()
```

```
19/19 [==============================] - 0s 957us/step - loss: 0.0148 - mae: 0.0978
19/19 [==============================] - 0s 950us/step - loss: 0.0974 - mae: 0.2451
```



c) Percobaan/Skenario Tiga

Mengubah mode “keras”, misalnya dengan menambah layer atau node dan melakukan training ulang. Menggunakan Optimizer sgd dalam tugas klasifikasi.

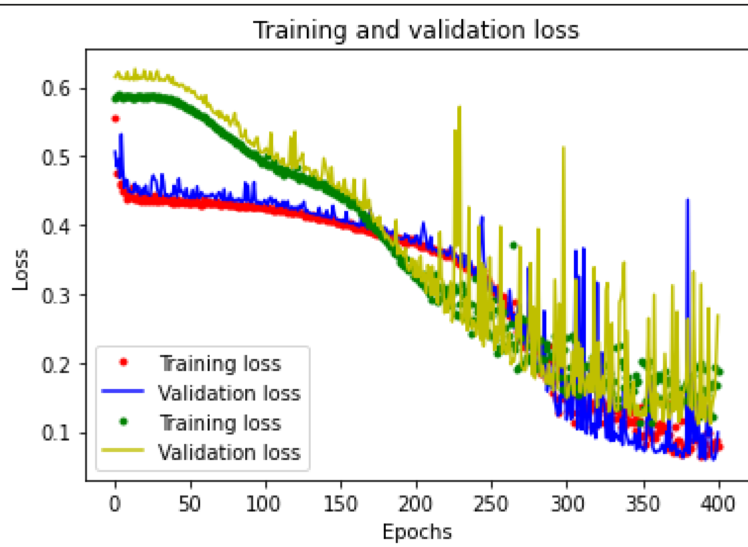
```
model_3 = tf.keras.Sequential()
model_4 = tf.keras.Sequential()
model_3.add(keras.layers.Dense(8, activation='relu', input_shape=(1,)))
model_4.add(keras.layers.Dense(8, activation='relu', input_shape=(1,)))
model_3.add(keras.layers.Dense(16, activation='relu'))
model_4.add(keras.layers.Dense(16, activation='relu'))
model_3.add(keras.layers.Dense(24, activation='relu'))
model_4.add(keras.layers.Dense(24, activation='relu'))
model_3.add(keras.layers.Dense(32, activation='relu'))
model_4.add(keras.layers.Dense(32, activation='relu'))
model_3.add(keras.layers.Dense(40, activation='relu'))
model_4.add(keras.layers.Dense(40, activation='relu'))
model_3.add(keras.layers.Dense(1))
model_4.add(keras.layers.Dense(1))
model_3.compile(optimizer='sgd', loss='mse', metrics=['mae'])
model_4.compile(optimizer='sgd', loss='mse', metrics=['mae'])
history_3 = model_3.fit(x_train, y_train, epochs=400,
batch_size=64,validation_data=(x_validate, y_validate))
```

```
history_4 = model_4.fit(x_train, z_train, epochs=400,
batch_size=64,validation_data=(x_validate, z_validate))
model_3.save(MODEL_TF)
model_4.save(MODEL_TF)
```

```
16/16 [==============================] - 0s 4ms/step - loss: 0.3023 - mae: 0.4309 - val_loss: 0.2991 - val_mae: 0.4190
Epoch 374/400
16/16 [==============================] - 0s 3ms/step - loss: 0.2653 - mae: 0.3988 - val_loss: 0.3162 - val_mae: 0.4232
Epoch 375/400
16/16 [==============================] - 0s 4ms/step - loss: 0.2867 - mae: 0.4152 - val_loss: 0.2919 - val_mae: 0.4159
Epoch 376/400
16/16 [==============================] - 0s 3ms/step - loss: 0.3297 - mae: 0.4430 - val_loss: 0.2878 - val_mae: 0.4173
Epoch 377/400
16/16 [==============================] - 0s 4ms/step - loss: 0.2842 - mae: 0.4149 - val_loss: 0.2877 - val_mae: 0.4164
Epoch 378/400
16/16 [==============================] - 0s 4ms/step - loss: 0.3002 - mae: 0.4249 - val_loss: 0.3028 - val_mae: 0.4293
Epoch 379/400
```

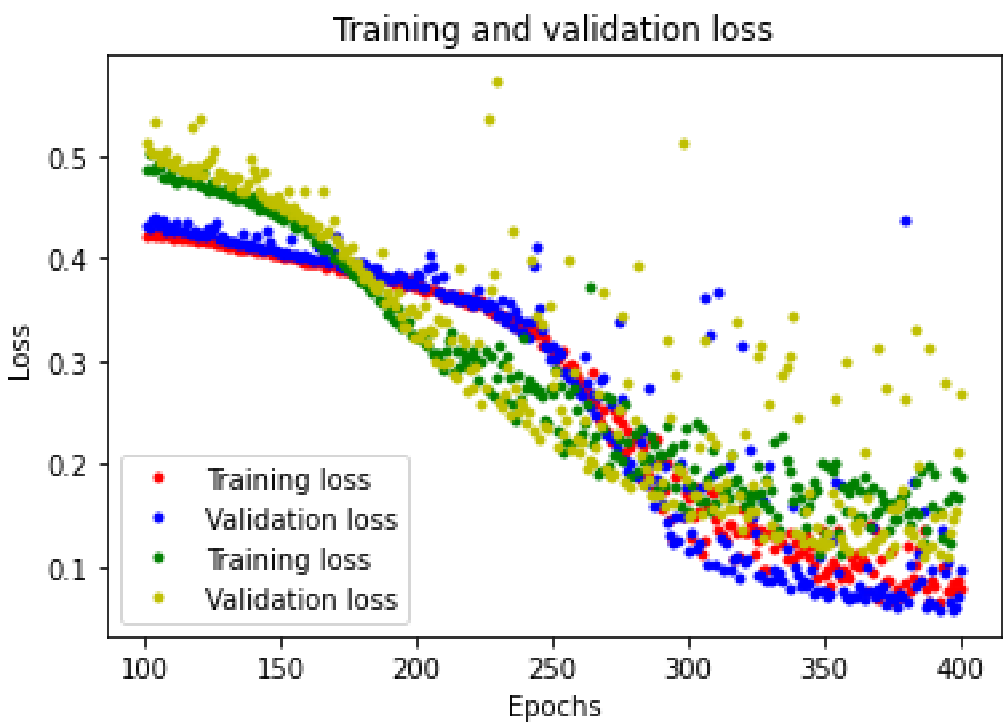
Melihat grafik error hasil training

```
train_loss3 = history_3.history['loss']
val_loss3 = history_3.history['val_loss']
train_loss4 = history_4.history['loss']
val_loss4 = history_4.history['val_loss']
epochs3 = range(1, len(train_loss3) + 1)
epochs4 = range(1, len(train_loss4) + 1)
plt.plot(epochs3, train_loss3, 'r.', label='Training loss')
plt.plot(epochs3, val_loss3, 'b', label='Validation loss')
plt.plot(epochs4, train_loss4, 'g.', label='Training loss')
plt.plot(epochs4, val_loss4, 'y', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



Melihat hasil training dari sisi lain

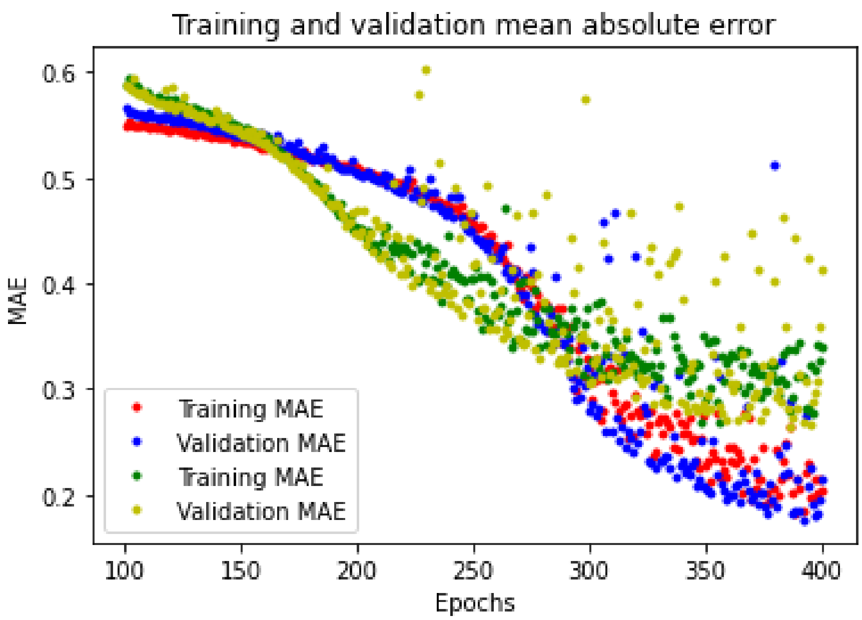
```
SKIP = 100
plt.plot(epochs3[SKIP:], train_loss3[SKIP:], 'r.', label='Training
loss')
plt.plot(epochs3[SKIP:], val_loss3[SKIP:], 'b.', label='Validation
loss')
plt.plot(epochs4[SKIP:], train_loss4[SKIP:], 'g.', label='Training
loss')
plt.plot(epochs4[SKIP:], val_loss4[SKIP:], 'y.', label='Validation
loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



Melihat laporan training dari sisi Mean Absolute Error

```
plt.clf()
train_mae3 = history_3.history['mae']
val_mae3 = history_3.history['val_mae']
train_mae4 = history_4.history['mae']
```

```
val_mae4 = history_4.history['val_mae']
plt.plot(epochs3[SKIP:], train_mae3[SKIP:], 'r.', label='Training
MAE')
plt.plot(epochs3[SKIP:], val_mae3[SKIP:], 'b.', label='Validation
MAE')
plt.plot(epochs4[SKIP:], train_mae4[SKIP:], 'g.', label='Training
MAE')
plt.plot(epochs4[SKIP:], val_mae4[SKIP:], 'y.', label='Validation
MAE')
plt.title('Training and validation mean absolute error')
plt.xlabel('Epochs')
plt.ylabel('MAE')
plt.legend()
plt.show()
```

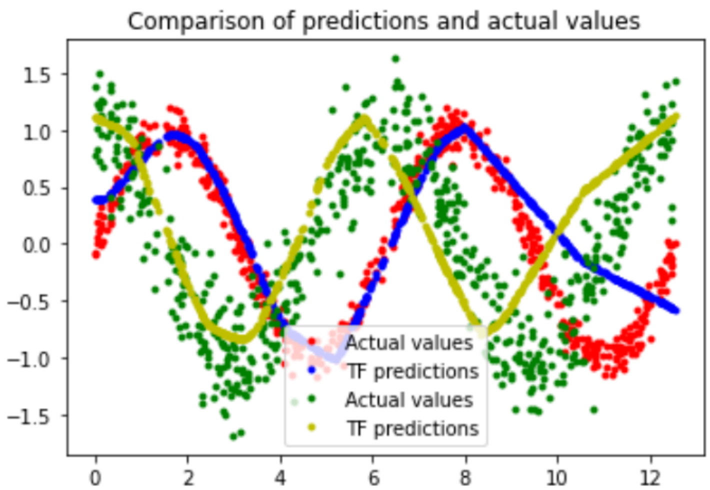


Evaluasi selisih training dengan scenario 3

```
test_loss3, test_mae3 = model_3.evaluate(x_test, y_test)
test_loss4, test_mae4 = model_4.evaluate(x_test, z_test)
y_test_pred = model_3.predict(x_test)
z_test_pred = model_4.predict(x_test)
plt.clf()
plt.title('Comparison of predictions and actual values')
plt.plot(x_test, y_test, 'r.', label='Actual values')
plt.plot(x_test, y_test_pred, 'b.', label='TF predictions')
plt.plot(x_test, z_test, 'g.', label='Actual values')
```

```
plt.plot(x_test, z_test_pred, 'y.', label='TF predictions')
plt.legend()
plt.show()
```

```
19/19 [==============================] - 0s 1ms/step - loss: 0.1009 - mae: 0.2214
19/19 [==============================] - 0s 946us/step - loss: 0.2693 - mae: 0.4104
```

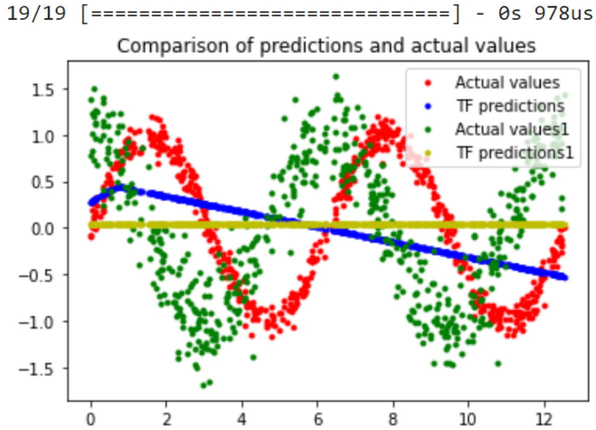


ANALISA

Dari percobaan simulasi yang telah dilakukan menggunakan tiga scenario, didapatkan hasil analisa sebagai berikut:

- Hasil scenario 1

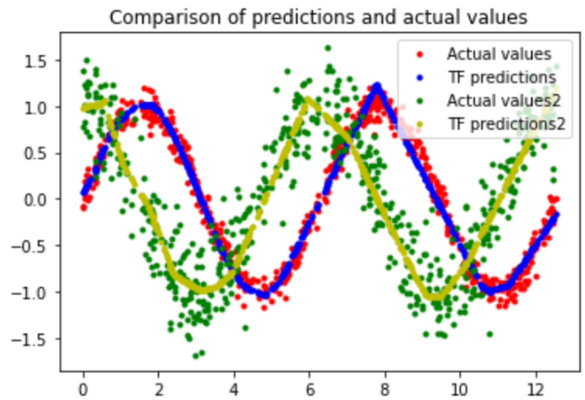
```
19/19 [==============================] - 0s 1ms/step - loss: 0.4208 - mae: 0.5553
19/19 [==============================] - 0s 978us/step - loss: 0.6133 - mae: 0.6793
```



Dari scenario 1 dapat diketahui bahwa nilai prediksi yang dilakukan oleh deep learning masih jauh berbeda dan belum sesuai dengan hasil actual. Hal ini kemungkinan karena jumlah hidden layer dan jumlah neuron yang terbatas. Pada scenario ini deep learning dianggap masih kurang cerdas.

- Hasil scenario 2

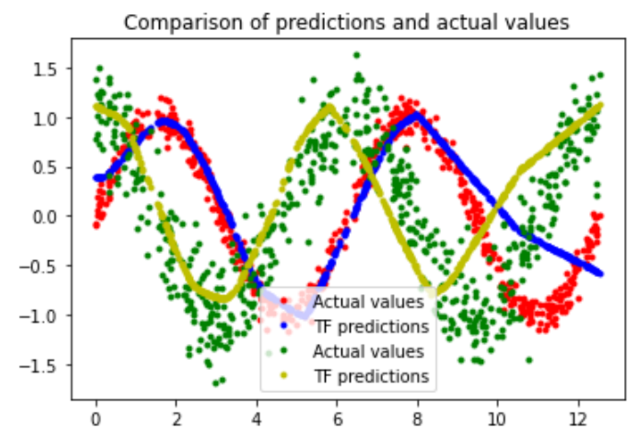
```
19/19 [==============================] - 0s 957us/step - loss: 0.0148 - mae: 0.0978
19/19 [==============================] - 0s 950us/step - loss: 0.0974 - mae: 0.2451
```



Pada scenario 2, jumlah hidden layer dan neuron disetiap layer ditambahkan kelipatan nilai dari neuron pada layer sebelumnya. Setelah dilakukan simulasi, didapatkan hasil yang cukup sesuai dengan nilai actual. Pada scenario ini deep learning dianggap sudah cukup cerdas.

- Hasil scenario 3

```
19/19 [==============================] - 0s 1ms/step - loss: 0.1009 - mae: 0.2214
19/19 [==============================] - 0s 946us/step - loss: 0.2693 - mae: 0.4104
```



Pada scenario ini, jumlah hidden layer dan neuron mirip dengan scenario 2. Yang membedakan adalah penggunaan optimizer sgd pada scenario ini. Setelah dilakukan simulasi didapatkan hasil prediksi yang menyerupai nilai actual akan tetapi gelombang yang dihasilkan lebih bergeser kekiri dibanding nilai aktualnya.