Code for prediction of Multiphase flow

```
# -*- coding: utf-8 -*-
"""MultiphaseFlowPrediction.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1jTWMIJxdXatFkoiNhcMMRX_ZCHQDhR-q
"""

import pandas as pd
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Input
from sklearn.model_selection import train_test_split

from google.colab import files
import io
data = files.upload()

df = pd.read_csv(io.StringIO(data['MULTIPHASE_V2.csv'].decode('utf-8')))

df.head()

df.shape

"""## *Split Data into train and test sets*

"""

train, test = train_test_split(df, test_size=0.2, random_state=1)
```

```
train, val = train_test_split(train, test_size=0.2, random_state=1)

"""## *Data Normalization and structuring*"""

def norm(x):
    return (x - train_stats['mean']) / train_stats['std']

def format_output(data):
    y1 = data.pop('water_kg_min')
    y1 = np.array(y1)
    y2 = data.pop('oil_kg_min')
    y2 = np.array(y2)
    y3 = data.pop('air_kg_min')
    y3 = np.array(y3)
    return y1, y2, y3

"""# *Format Water, Oil and air data as numpy array*"""

train_stats = train.describe()
train_stats.pop('water_kg_min')
train_stats.pop('oil_kg_min')
train_stats.pop('air_kg_min')
train_stats = train_stats.transpose()
train_Y = format_output(train)
test_Y = format_output(test)
val_Y = format_output(val)

print(train_stats)

normalized_train_X = np.array(norm(train))
normalized_test_X = np.array(norm(test))
normalized_val_X = np.array(norm(val))

def build_model():
```

```
    # Model layers.
    input_layer = Input(shape=(len(train.columns),))
    first_dense_layer = Dense(units='128', activation='relu')(input_layer)
    # Y1 output wfrom first dense
    y1_out = Dense(units='1', name='water_kg_min')(first_dense_layer)

    second_dense_layer = Dense(units='128', activation='relu')(first_dense_layer)
    # Y2 output from second dense
    y2_out = Dense(units='1', name='oil_kg_min')(second_dense_layer)

    third_dense_layer = Dense(units='128', activation='relu')(second_dense_layer)
    # Y3 output from third dense
    y3_out = Dense(units='1', name='air_kg_min')(third_dense_layer)

    # Model with input and output layers
    model = tf.keras.Model(inputs=input_layer, outputs=[y1_out, y2_out, y3_out])

    return model

"""# *Model, Optimizer, loss and matrics*"""

model = build_model()

optimizer = tf.keras.optimizers.SGD(lr=0.001)
model.compile(optimizer=optimizer,
              loss={'water_kg_min': 'mse', 'oil_kg_min': 'mse','air_kg_min': 'mse'},
              metrics={'water_kg_min': tf.keras.metrics.RootMeanSquaredError(),
                       'oil_kg_min': tf.keras.metrics.RootMeanSquaredError(),
                       'air_kg_min': tf.keras.metrics.RootMeanSquaredError()})

history = model.fit(normalized_train_X, train_Y,
                    epochs=100, batch_size=10, validation_data=(normalized_test_X, test_Y))
```

```
loss, Y1_loss, Y2_loss,Y3_loss, Y1_rmse, Y2_rmse, Y3_rmse =
model.evaluate(x=normalized_val_X, y=val_Y)

print()
print(f'loss: {loss}')
print(f'water_kg_min: {Y1_loss}')
print(f'oil_kg_min: {Y2_loss}')
print(f'air_kg_min: {Y3_loss}')
print(f'water_kg_min: {Y1_rmse}')
print(f'oil_kg_min: {Y2_rmse}')
print(f'air_kg_min: {Y3_rmse}')

def plot_diff(y_true, y_pred, title=''):
    plt.plot(y_true,label="True Values")
    plt.plot(y_pred,label= "Predicted Values")
    plt.title(title)
    plt.legend()
    plt.show()

def plot_metrics(metric_name, title, ylim=5):
    plt.title(title)
    plt.ylim(0, ylim)
    plt.plot(history.history[metric_name], color='blue', label=metric_name)
    plt.plot(history.history['val_' + metric_name], color='green', label='val_' + metric_name)
    plt.legend()
    plt.show()

Y_pred = model.predict(normalized_test_X)
water_kg_min = Y_pred[0]
oil_kg_min = Y_pred[1]
air_kg_min = Y_pred[2]

plot_diff(test_Y[0],Y_pred[0],title='water_kg_min')
plot_diff(test_Y[1], Y_pred[1], title='oil_kg_min')
```

```
plot_diff(test_Y[2], Y_pred[2], title='air_kg_min')

# Plot RMSE
plot_metrics(metric_name='water_kg_min_root_mean_squared_error', title='Water Flow RMSE', ylim=50)
plot_metrics(metric_name='oil_kg_min_root_mean_squared_error', title='Oil Flow RMSE', ylim=50)
plot_metrics(metric_name='air_kg_min_root_mean_squared_error', title='Air Flow RMSE', ylim=7)
```