Model_Training_LSTM_Hyperparameters

February 8, 2022

1 Controlador de vuelo para vehículos aéreos no tripulados multi-rotor basado en técnicas de aprendizaje profundo

1.1 Entrenamiento Red LSTM

1.1.1 Javier Cárdenas - Uriel Carrero

1.2 1. Descripción del Dataset

Importar Librerías

[1]:
```
import os
import sys
import random

import pandas as pd
pd.set_option('display.max_columns', None) #Para mostrar todas las columnas
import matplotlib.pyplot as plt

import numpy as np                    # Cómputo Numérico
print(np.__version__)
assert (np.__version__=='1.19.5'), 'Versión incorrecta de numpy, por favor␣
↪instale 1.19.5'
seed = 5
np.random.seed(seed)
np.seterr(divide = 'ignore')
```

```
1.18.2
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-1-b978317cd0ae> in <module>
      9 import numpy as np                    # Cómputo Numérico
     10 print(np.__version__)
---> 11 assert (np.__version__=='1.19.5'), 'Versión incorrecta de numpy, por␣
↪favor instale 1.19.5'
     12 seed = 5
     13 np.random.seed(seed)
```

```
AssertionError: Versión incorrecta de numpy, por favor instale 1.19.5
```

[2]:
```
os.environ['TF_KERAS'] = '1'
import keras as kr
import keras_tuner as kt

import tensorflow as tf
from tensorflow.keras import models, layers
print(tf.__version__)
assert (tf.__version__=='2.5.0'), 'Versión incorrecta de Tensorflow, por favor␣
 ↪instale 2.5.0'
print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))

from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession

from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split

from IPython.display import clear_output
```

```
2.5.0
Num GPUs Available:  1
```

[3]:
```
print(kr.__version__)
```

```
2.5.0
```

[3]:
```
gpus = tf.config.list_physical_devices('GPU')
config = ConfigProto()
if gpus:
    try:
        config.gpu_options.allow_growth = True
        tf.compat.v1.enable_eager_execution()

        os.environ["TF_CPP_MIN_LOG_LEVEL"] = "2"
        # Currently, memory growth needs to be the same across GPUs
        for gpu in gpus:
            tf.config.experimental.set_memory_growth(gpu, True)
        logical_gpus = tf.config.experimental.list_logical_devices('GPU')
        print(len(gpus), "Physical GPUs,", len(logical_gpus), "Logical GPUs")
    except RuntimeError as e:
        # Memory growth must be set before GPUs have been initialized
        print(e)
session = InteractiveSession(config=config)
```

```
1 Physical GPUs, 1 Logical GPUs
```

[4]:
```
import gc #garbage collector
import gc; gc.enable()
```

1.3 Cargar Datos

1.3.1 Leemos el Dataset

[5]:
```
root = '../logs/Datasets/'
dataset_name = 'Dataset_Final'
rootdir = root+dataset_name
if not os.path.exists(rootdir):
    print(f"{rootdir} not exist")
```

[6]:
```
df = pd.read_csv(os.path.join(rootdir, random.choice(os.listdir(rootdir))))
delete_list = ['timestamps',
               'Q1', 'Q2', 'Q3', 'Q4',
               'uvx', 'uvy', 'uvz',
               'up', 'uq',
               'uwp', 'uwq', 'uwr']
df_list = df.columns.to_list()
rpm_list = [i for i in df_list if ("RPM" in i)]
states_list = [i for i in df_list if not ((i in delete_list) or (i in␣
 ↪rpm_list))]
Ts = df['timestamps'][1]-df['timestamps'][0]
fs = 1/Ts
```

[7]:
```
dataset = []
for filename in os.listdir(rootdir):
    if not filename.endswith(".csv"):
        continue

    df = pd.read_csv(os.path.join(rootdir, filename))
    df =  df.drop(delete_list, axis=1)
    x = df.drop(rpm_list, axis=1)
    y = df.drop(states_list, axis=1)
    dataset.append([x, y])

df = None
x = None
y= None
```

Normalización de Estados (Entradas) y Acciones (Salidas)

[8]:
```
def Norm(df, df_desc):
    for prop in list(df.columns):
        try:
            # 1 ~ Mean  7 ~ Max  3 ~ Min
```

```
            df[prop] = (df[prop]-df_desc[prop]['mean'])/
↪(df_desc[prop]['max']-df_desc[prop]['min'])
        except e:
            print(e)
    return df
```

[9]:
```
norm_data_path = f"{root}/data_description_{dataset_name}.csv"
df_desc = pd.read_csv(norm_data_path, index_col=0)
df_desc
```

[9]:

	x	y	z	p	q	\
count	3.068233e+06	3.068233e+06	3.068233e+06	3.068233e+06	3.068233e+06	
mean	6.352137e-03	3.631930e-03	8.701060e-01	8.113104e-04	7.632901e-04	
std	1.067096e-01	1.015022e-01	7.571069e-01	3.005110e-02	3.288610e-02	
min	-8.166897e-01	-8.176113e-01	2.113373e-04	-4.872289e-01	-4.386099e-01	
25%	-7.023093e-03	-4.566531e-03	3.597720e-01	-7.548420e-04	-1.295726e-03	
50%	9.264773e-05	8.164912e-06	9.371726e-01	0.000000e+00	2.258764e-17	
75%	2.154385e-02	1.321506e-02	1.092425e+00	9.607826e-04	1.469981e-03	
max	8.194435e-01	8.146467e-01	4.000000e+00	3.767827e-01	4.339304e-01	

	r	vx	vy	vz	wp	\
count	3.068233e+06	3.068233e+06	3.068233e+06	3.068233e+06	3.068233e+06	
mean	1.239124e-02	-1.253432e-04	-2.641033e-04	1.198860e-03	4.372706e-06	
std	4.450773e-01	7.177616e-02	6.552093e-02	2.601583e-01	1.942947e-01	
min	-3.141419e+00	-1.156056e+00	-8.621260e-01	-6.841123e+00	-1.045986e+01	
25%	-3.379343e-04	-2.080171e-03	-1.373874e-03	-1.070760e-02	-5.708022e-03	
50%	0.000000e+00	-1.518968e-08	1.364815e-17	3.785519e-05	0.000000e+00	
75%	5.312049e-04	2.095210e-03	1.519517e-03	1.501362e-02	5.440229e-03	
max	3.141577e+00	8.494385e-01	8.597021e-01	6.373837e+00	7.147506e+00	

	wq	wr	ax	ay	az	\
count	3.068233e+06	3.068233e+06	3.068233e+06	3.068233e+06	3.068233e+06	
mean	1.434372e-04	2.558978e-03	-1.772714e-05	3.597617e-05	-2.469848e-05	
std	2.135545e-01	5.107082e-01	3.168708e-01	2.945918e-01	1.454656e+00	
min	-7.981405e+00	-7.181414e+00	-4.279933e+01	-3.656422e+01	-9.800000e+00	
25%	-8.702793e-03	-6.507613e-05	-1.356462e-02	-7.912034e-03	-4.078746e-02	
50%	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	-1.865757e-05	
75%	8.939743e-03	3.879907e-05	1.237756e-02	8.299066e-03	3.374101e-02	
max	1.179763e+01	7.300762e+00	3.906528e+01	3.332884e+01	1.921161e+02	

	ap	aq	ar	RPM0	RPM1	\
count	3.068233e+06	3.068233e+06	3.068233e+06	3.068233e+06	3.068233e+06	
mean	1.224907e-04	5.466858e-05	-3.303912e-05	1.441162e+04	1.441290e+04	
std	7.742095e+00	6.830287e+00	3.325449e+00	1.055152e+03	1.058528e+03	
min	-2.510366e+03	-3.059815e+03	-2.531001e+02	9.440300e+03	9.440300e+03	
25%	-4.151471e-02	-6.752393e-02	-2.696169e-04	1.438274e+04	1.438057e+04	
50%	0.000000e+00	0.000000e+00	0.000000e+00	1.446835e+04	1.446836e+04	

```
75%    4.180967e-02  6.758204e-02  3.539114e-04  1.452948e+04  1.453062e+04
max    2.711637e+03  2.831431e+03  1.630572e+02  2.166645e+04  2.166645e+04

                RPM2          RPM3            ux            uy            uz  \
count   3.068233e+06  3.068233e+06  3.068233e+06  3.068233e+06  3.068233e+06
mean    1.440977e+04  1.441093e+04  6.439717e-03  3.417250e-03  8.782731e-01
std     1.058532e+03  1.058966e+03  1.058331e-01  1.010920e-01  7.641660e-01
min     9.440300e+03  9.440300e+03 -8.000000e-01 -8.000000e-01  0.000000e+00
25%     1.438241e+04  1.437958e+04  0.000000e+00  0.000000e+00  3.549988e-01
50%     1.446834e+04  1.446834e+04  0.000000e+00  0.000000e+00  9.364582e-01
75%     1.452831e+04  1.452817e+04  1.838854e-02  8.750144e-03  1.144531e+00
max     2.166645e+04  2.166645e+04  8.000000e-01  8.000000e-01  4.000000e+00

                ur
count   3.068233e+06
mean    1.285404e-02
std     4.414466e-01
min    -3.140685e+00
25%     0.000000e+00
50%     0.000000e+00
75%     0.000000e+00
max     3.141519e+00
```

[10]:
```
for i, data in enumerate(dataset):
    x, y = data
    Norm(x, df_desc)
    Norm(y, df_desc)
    dataset[i]=[x,y]
```

División del dataset para entrenamiento, validación, prueba

[11]:
```
X = []
Y = []

for sample in dataset:
    x, y = sample
    X.append(x)
    Y.append(y)

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.06,␣
 ↪random_state=42)
X_train,  X_val, Y_train, Y_val  = train_test_split(X_train, Y_train,␣
 ↪test_size=0.1)

del X
del Y
```

[12]:
```
print(f'Total={len(dataset)}, Entrenamiento={len(X_train)}␣
↪({round(100*len(X_train)/len(dataset))}%),'\
      f'Validación={len(X_val)} ({round(100*len(X_val)/len(dataset))}%),'\
      f'Prueba={len(X_test)} ({round(100*len(X_test)/len(dataset))}%)')
```

```
Total=167, Entrenamiento=140 (84%),Validación=16 (10%),Prueba=11 (7%)
```

Generador de Ejemplos de entrenamiento Entrenar un modelo con una señal de 50000 datos en cada iteración sería una tarea que tomaría demasiado tiempo, así mismo, cuando se necesite realizar la inferencia del modelo, se necesitaría esa misma cantidad de datos, por lo que no se utiliza toda la señal de entrenamiento, sino pequeños segmentos de tamaño N, por lo que se generarán M-N (longitud de toda la señal, 50000) señales de longitud N para el entrenamiento, lo que aumentaría el consumo de memoria. Por tal motivo se define un generador.

[13]:
```
class DataGenerator:
    def __init__(self, X=[], Y=[], dataset = None, batch_size=512, window=512,␣
↪sequence_out=False, variable_window=False, delta_window=1, feedback=False,␣
↪window_feedback=1):
        if dataset:
            for data in dataset:
                X.append(data[0])
                Y.append(data[1])
            self.X = X
            self.Y = Y
        elif X and Y:
            if len(X)!=len(Y):
                raise Exception("La longitud de datos de X e Y deben ser␣
↪iguales")
            self.X = X
            self.Y = Y
        else:
            raise Exception("Debe especificar dataset o X, Y")

        self.n  = len(X)                  ### Número de ejemplos de entrenamiento
        x_shape = X[0].shape
        y_shape = Y[0].shape
        self.batch_size = batch_size
        self.window = window
        self.variable_window = variable_window
        self.delta_window = delta_window
        self.feedback = feedback
        self.i  = x_shape[1] if not self.feedback else x_shape[1]+y_shape[1]  ␣
↪### Número de características
        self.j  = y_shape[1]                                                  ␣
↪### Número de salidas
        #self.window_feedback = window_feedback
        if self.variable_window:
```

```
            self.window_max = self.window+self.delta_window
            self.window_min = self.window-self.delta_window
            if self.window_min<1:
                raise IndexError(f'delta_window no puede ser igual o mayor a la␣
↪ventana')
        self.sequence_out = sequence_out
        self.set_shapes()

    def set_shapes(self):
        if self.sequence_out:
            self.shapes = ((self.batch_size, self.window, self.i),
                           (self.batch_size, self.window, self.j))
        else:
            self.shapes = ((self.batch_size, self.window, self.i),
                           (self.batch_size, self.j))

    def buid_init(self):
        if self.variable_window:
            self.window = np.random.randint(self.window_min, self.window_max)
            self.set_shapes()
        self.samples = np.empty(shape= self.shapes[0], dtype='float32')
        self.labels  = np.empty(shape= self.shapes[1], dtype='float32')
        self.batchcount = 0

    def build_data(self):
        self.buid_init()
        if self.feedback:
            i_0 = 1
        else:
            i_0 = 0
        while True:
            try:
                index = np.random.randint(0, self.n-1)            ###␣
↪Trayectoria a seleccionar
                m = len(self.X[index])          ### Número de steps por ejemplo
                if m-self.window-1<=0:
                    raise IndexError(f'El tamaño de la ventana es mayor a la␣
↪trayectoria')
                else:
                    start_index = np.random.randint(i_0, int(m-self.window-1))
                    final_index = start_index+self.window
                    x = self.X[index][start_index:final_index].to_numpy()
                    if self.feedback:
                        y = self.Y[index][start_index-1:final_index-1].
↪to_numpy()
                        self.samples[self.batchcount] = np.concatenate((x,y),␣
↪axis=1)
```

```
                    else:
                        self.samples[self.batchcount] = x

                    if self.sequence_out:
                        self.labels[self.batchcount] = self.
↪Y[index][start_index:final_index].to_numpy()
                    else:
                        self.labels[self.batchcount] = self.Y[index].
↪loc[final_index]
            except IndexError as e:
                print(f'ERROR: Ejemplo {self.batchcount}: {e}')
                raise e

            self.batchcount += 1
            if self.batchcount >= self.batch_size:
                yield self.samples.astype(np.float32), self.labels.astype(np.
↪float32)
                self.buid_init()
```

[14]:
```
window = 64                    ### Número de steps por ejemplo
batch_size = 512               ### Número de ejemplos por batch
sequence_out = False
variable_window=True
feedback = False
delta_window=window/3
```

[15]:
```
train_generator = DataGenerator(X=X_train, Y=Y_train, batch_size=batch_size,␣
↪window=window, sequence_out=sequence_out, variable_window=variable_window,␣
↪delta_window=delta_window, feedback=feedback)
val_generator   = DataGenerator(X=X_val,   Y=Y_val,   batch_size=batch_size,␣
↪window=window, sequence_out=sequence_out, variable_window=variable_window,␣
↪delta_window=delta_window, feedback=feedback)
test_generator  = DataGenerator(X=X_test,  Y=Y_test,  batch_size=batch_size,␣
↪window=window, sequence_out=sequence_out, variable_window=variable_window,␣
↪delta_window=delta_window, feedback=feedback)
```

[16]:
```
dataset_train = tf.data.Dataset.from_generator(train_generator.build_data,
                                               output_types = (tf.float32, tf.float32))
dataset_val = tf.data.Dataset.from_generator(val_generator.build_data,
                                               output_types = (tf.float32, tf.float32))
dataset_test = tf.data.Dataset.from_generator(test_generator.build_data,
                                               output_types = (tf.float32, tf.float32))
```

[17]:
```
for _ in range(5):
    x, y = next(train_generator.build_data())
    print(f'x.shape={x.shape}, y.shape={y.shape}')
```

```
x.shape=(512, 64, 22), y.shape=(512, 4)
x.shape=(512, 81, 22), y.shape=(512, 4)
x.shape=(512, 55, 22), y.shape=(512, 4)
x.shape=(512, 71, 22), y.shape=(512, 4)
x.shape=(512, 60, 22), y.shape=(512, 4)
```

1.4 Keras Model

1.5 Callbacks

[18]:
```
main_metric = 'mean_squared_error'
#metrics = [main_metric, 'cosine_similarity', 'logcosh']
metrics = main_metric
```

Definición del Modelo

[19]:
```
input_dim = len(states_list) if not feedback else len(states_list)+len(rpm_list)
output_dim = len(rpm_list)
print(f'input_dim: {input_dim}, output_dim: {output_dim}')
```

```
input_dim: 22, output_dim: 4
```

[20]:
```
class LSTMHyperModel(kt.HyperModel):
    def __init__(self, input_dim, output_dim, metrics='mean_squared_error',␣
 ↪loss='mean_squared_error'):
        self.input_dim = input_dim
        self.output_dim = output_dim
        self.metrics = metrics
        self.loss = loss

    def build(self, hp):
        model = tf.keras.Sequential()
        model.add(layers.LSTM(units=hp.
 ↪Int('LSTM_units',min_value=32,max_value=512,step=32,default=256),
                              input_shape=(None, self.input_dim),␣
 ↪return_sequences=True))

        hidden_LSTMlayers = hp.
 ↪Int('Hidden_LSTMlayers',min_value=1,max_value=5,step=1,default=5)
        conv1d_layer = hp.Boolean('Conv1D_layer', default=True)
        for i in range(hidden_LSTMlayers):
            if conv1d_layer:
                model.add(layers.Conv1D(filters=hp.
 ↪Int(f'Conv1_filters_{i}',min_value=32,max_value=512,step=32,default=128),
                                        kernel_size=3, padding='same',
                                        activation=hp.Choice(
                                        'Conv1_activation',
                                        values=['relu', 'tanh', 'sigmoid'],
                                        default='relu')
```

```
                                        ))
            if i==hidden_LSTMlayers-1:
                return_sequences = False
            else:
                return_sequences = True

            model.add(layers.LSTM(units=hp.
↪Int(f'LSTM_Hidden_units_{i}',min_value=32,max_value=256,step=32,default=128),
                                  return_sequences=return_sequences))

        hidden_layers = hp.
↪Int('Hidden_FClayers',min_value=1,max_value=5,step=1,default=5)
        for i in range(hidden_layers):
            model.add(layers.Dense(units=hp.
↪Int(f'Hidden_units_{i}',min_value=32,max_value=512,step=32,default=128),
                                   activation=hp.Choice(
                                        'dense_activation',
                                        values=['relu', 'tanh', 'sigmoid'],
                                        default='relu')))

        model.add(layers.Dense(self.output_dim))

        model.compile(
            optimizer=tf.keras.optimizers.Adam(
                learning_rate=hp.Float(
                    'learning_rate',
                    min_value=1e-4,
                    max_value=1e-2,
                    sampling='LOG',
                    default=1e-3
                )
            ),
            loss=self.loss,
            metrics=self.metrics
        )
        return model
```

[21]:
```
hypermodel = LSTMHyperModel(input_dim, output_dim)
```

Tuner

[22]:
```
SEED = 1
MAX_TRIALS = 50
EXECUTION_PER_TRIAL = 2
HYPERBAND_MAX_EPOCHS = 150

tuner = kt.tuners.Hyperband(
    hypermodel,
```

```
    max_epochs=HYPERBAND_MAX_EPOCHS,
    objective=F'val_{main_metric}',
    seed=SEED,
    executions_per_trial=EXECUTION_PER_TRIAL,
    directory='./Models/hyperband/',
    project_name=f'{dataset_name}'
)
```

```
INFO:tensorflow:Reloading Oracle from existing project
./Models/hyperband/Dataset_Final\oracle.json
INFO:tensorflow:Reloading Tuner from
./Models/hyperband/Dataset_Final\tuner0.json
```

Compilado el Modelo

[23]:
```
tuner.search_space_summary()
```

```
Search space summary
Default search space size: 22
LSTM_units (Int)
{'default': 256, 'conditions': [], 'min_value': 32, 'max_value': 512, 'step':
32, 'sampling': None}
Hidden_LSTMlayers (Int)
{'default': 5, 'conditions': [], 'min_value': 1, 'max_value': 5, 'step': 1,
'sampling': None}
Conv1D_layer (Boolean)
{'default': True, 'conditions': []}
Conv1_filters_0 (Int)
{'default': 128, 'conditions': [], 'min_value': 32, 'max_value': 512, 'step':
32, 'sampling': None}
Conv1_activation (Choice)
{'default': 'relu', 'conditions': [], 'values': ['relu', 'tanh', 'sigmoid'],
'ordered': False}
LSTM_Hidden_units_0 (Int)
{'default': 128, 'conditions': [], 'min_value': 32, 'max_value': 256, 'step':
32, 'sampling': None}
Conv1_filters_1 (Int)
{'default': 128, 'conditions': [], 'min_value': 32, 'max_value': 512, 'step':
32, 'sampling': None}
LSTM_Hidden_units_1 (Int)
{'default': 128, 'conditions': [], 'min_value': 32, 'max_value': 256, 'step':
32, 'sampling': None}
Conv1_filters_2 (Int)
{'default': 128, 'conditions': [], 'min_value': 32, 'max_value': 512, 'step':
32, 'sampling': None}
LSTM_Hidden_units_2 (Int)
{'default': 128, 'conditions': [], 'min_value': 32, 'max_value': 256, 'step':
32, 'sampling': None}
Conv1_filters_3 (Int)
```

```
{'default': 128, 'conditions': [], 'min_value': 32, 'max_value': 512, 'step':
32, 'sampling': None}
LSTM_Hidden_units_3 (Int)
{'default': 128, 'conditions': [], 'min_value': 32, 'max_value': 256, 'step':
32, 'sampling': None}
Conv1_filters_4 (Int)
{'default': 128, 'conditions': [], 'min_value': 32, 'max_value': 512, 'step':
32, 'sampling': None}
LSTM_Hidden_units_4 (Int)
{'default': 128, 'conditions': [], 'min_value': 32, 'max_value': 256, 'step':
32, 'sampling': None}
Hidden_FClayers (Int)
{'default': 5, 'conditions': [], 'min_value': 1, 'max_value': 5, 'step': 1,
'sampling': None}
Hidden_units_0 (Int)
{'default': 128, 'conditions': [], 'min_value': 32, 'max_value': 512, 'step':
32, 'sampling': None}
dense_activation (Choice)
{'default': 'relu', 'conditions': [], 'values': ['relu', 'tanh', 'sigmoid'],
'ordered': False}
Hidden_units_1 (Int)
{'default': 128, 'conditions': [], 'min_value': 32, 'max_value': 512, 'step':
32, 'sampling': None}
Hidden_units_2 (Int)
{'default': 128, 'conditions': [], 'min_value': 32, 'max_value': 512, 'step':
32, 'sampling': None}
Hidden_units_3 (Int)
{'default': 128, 'conditions': [], 'min_value': 32, 'max_value': 512, 'step':
32, 'sampling': None}
Hidden_units_4 (Int)
{'default': 128, 'conditions': [], 'min_value': 32, 'max_value': 512, 'step':
32, 'sampling': None}
learning_rate (Float)
{'default': 0.001, 'conditions': [], 'min_value': 0.0001, 'max_value': 0.01,
'step': None, 'sampling': 'log'}
```

Entrenamiento del Modelo

[ ]:
```
%%time
N_EPOCH_SEARCH = 150

#max_queue_size = maximum size of the internal training queue which is used to␣
 ↪"precache" samples from the generator
#workers = number of threads generating batches in parallel. Batches are␣
 ↪computed in parallel on the CPU and passed on the fly onto the GPU for␣
 ↪neural network computations
tuner.search(dataset_train.repeat(),
             steps_per_epoch=len(X_train), # steps per epoch
```

```
            epochs=N_EPOCH_SEARCH,
            validation_data=dataset_val.repeat(),
            validation_steps=1,
            max_queue_size=2, ##
            workers=1,
            callbacks=[kr.callbacks.TensorBoard(f'../Models/tmp/tb_logs/
↪{dataset_name}')]
            )
```

Se guarda el Modelo

[32]:
```
# Show a summary of the search
tuner.results_summary()
```

```
Results summary
Results in ./Models/hyperband/Dataset_Final
Showing 10 best trials
Objective(name='val_mean_squared_error', direction='min')
Trial summary
Hyperparameters:
LSTM_units: 320
Hidden_LSTMlayers: 2
Conv1D_layer: True
Conv1_filters_0: 480
Conv1_activation: relu
LSTM_Hidden_units_0: 128
Conv1_filters_1: 288
LSTM_Hidden_units_1: 128
Conv1_filters_2: 160
LSTM_Hidden_units_2: 32
Conv1_filters_3: 192
LSTM_Hidden_units_3: 64
Conv1_filters_4: 256
LSTM_Hidden_units_4: 160
Hidden_FClayers: 4
Hidden_units_0: 64
dense_activation: tanh
Hidden_units_1: 96
Hidden_units_2: 320
Hidden_units_3: 192
Hidden_units_4: 320
learning_rate: 0.001198186075494075
tuner/epochs: 150
tuner/initial_epoch: 50
tuner/bracket: 3
tuner/round: 3
tuner/trial_id: 86eac072764f89ae45e32d96b5cb8f8e
Score: 0.0001823654820327647
```

```
Trial summary
Hyperparameters:
LSTM_units: 320
Hidden_LSTMlayers: 2
Conv1D_layer: True
Conv1_filters_0: 320
Conv1_activation: relu
LSTM_Hidden_units_0: 32
Conv1_filters_1: 288
LSTM_Hidden_units_1: 128
Conv1_filters_2: 224
LSTM_Hidden_units_2: 224
Conv1_filters_3: 480
LSTM_Hidden_units_3: 64
Conv1_filters_4: 416
LSTM_Hidden_units_4: 256
Hidden_FClayers: 3
Hidden_units_0: 32
dense_activation: tanh
Hidden_units_1: 416
Hidden_units_2: 288
Hidden_units_3: 32
Hidden_units_4: 96
learning_rate: 0.0010771046673067758
tuner/epochs: 150
tuner/initial_epoch: 50
tuner/bracket: 4
tuner/round: 4
tuner/trial_id: 8ed0c7a268133d3179b0554a68f18490
Score: 0.00019305243040435016
Trial summary
Hyperparameters:
LSTM_units: 288
Hidden_LSTMlayers: 2
Conv1D_layer: False
Conv1_filters_0: 160
Conv1_activation: tanh
LSTM_Hidden_units_0: 96
Conv1_filters_1: 512
LSTM_Hidden_units_1: 192
Conv1_filters_2: 480
LSTM_Hidden_units_2: 128
Conv1_filters_3: 192
LSTM_Hidden_units_3: 64
Conv1_filters_4: 256
LSTM_Hidden_units_4: 192
Hidden_FClayers: 2
Hidden_units_0: 128
```

```
dense_activation: relu
Hidden_units_1: 192
Hidden_units_2: 352
Hidden_units_3: 224
Hidden_units_4: 352
learning_rate: 0.0017615796247596443
tuner/epochs: 150
tuner/initial_epoch: 50
tuner/bracket: 3
tuner/round: 3
tuner/trial_id: 2a6a4ed9e8550262304af295349fbf1f
Score: 0.00020064153795829043
Trial summary
Hyperparameters:
LSTM_units: 192
Hidden_LSTMlayers: 4
Conv1D_layer: False
Conv1_filters_0: 160
Conv1_activation: relu
LSTM_Hidden_units_0: 128
Conv1_filters_1: 352
LSTM_Hidden_units_1: 96
Conv1_filters_2: 224
LSTM_Hidden_units_2: 128
Conv1_filters_3: 128
LSTM_Hidden_units_3: 96
Conv1_filters_4: 32
LSTM_Hidden_units_4: 160
Hidden_FClayers: 2
Hidden_units_0: 416
dense_activation: tanh
Hidden_units_1: 128
Hidden_units_2: 64
Hidden_units_3: 320
Hidden_units_4: 320
learning_rate: 0.0008478087418210128
tuner/epochs: 150
tuner/initial_epoch: 50
tuner/bracket: 4
tuner/round: 4
tuner/trial_id: bcb1a1c436e8cc4e8eb4546c04236f17
Score: 0.00020492844487307593
Trial summary
Hyperparameters:
LSTM_units: 320
Hidden_LSTMlayers: 2
Conv1D_layer: True
Conv1_filters_0: 480
```

```
Conv1_activation: relu
LSTM_Hidden_units_0: 128
Conv1_filters_1: 288
LSTM_Hidden_units_1: 128
Conv1_filters_2: 160
LSTM_Hidden_units_2: 32
Conv1_filters_3: 192
LSTM_Hidden_units_3: 64
Conv1_filters_4: 256
LSTM_Hidden_units_4: 160
Hidden_FClayers: 4
Hidden_units_0: 64
dense_activation: tanh
Hidden_units_1: 96
Hidden_units_2: 320
Hidden_units_3: 192
Hidden_units_4: 320
learning_rate: 0.001198186075494075
tuner/epochs: 50
tuner/initial_epoch: 17
tuner/bracket: 3
tuner/round: 2
tuner/trial_id: ad4bd96d87234a355a72e1f4c856b7ad
Score: 0.000290848889562767
Trial summary
Hyperparameters:
LSTM_units: 320
Hidden_LSTMlayers: 2
Conv1D_layer: True
Conv1_filters_0: 320
Conv1_activation: relu
LSTM_Hidden_units_0: 32
Conv1_filters_1: 288
LSTM_Hidden_units_1: 128
Conv1_filters_2: 224
LSTM_Hidden_units_2: 224
Conv1_filters_3: 480
LSTM_Hidden_units_3: 64
Conv1_filters_4: 416
LSTM_Hidden_units_4: 256
Hidden_FClayers: 3
Hidden_units_0: 32
dense_activation: tanh
Hidden_units_1: 416
Hidden_units_2: 288
Hidden_units_3: 32
Hidden_units_4: 96
learning_rate: 0.0010771046673067758
```

```
tuner/epochs: 50
tuner/initial_epoch: 17
tuner/bracket: 4
tuner/round: 3
tuner/trial_id: c1772e4911328d7dd28277e6c388a1b5
Score: 0.0003511321992846206
Trial summary
Hyperparameters:
LSTM_units: 192
Hidden_LSTMlayers: 4
Conv1D_layer: False
Conv1_filters_0: 160
Conv1_activation: relu
LSTM_Hidden_units_0: 128
Conv1_filters_1: 352
LSTM_Hidden_units_1: 96
Conv1_filters_2: 224
LSTM_Hidden_units_2: 128
Conv1_filters_3: 128
LSTM_Hidden_units_3: 96
Conv1_filters_4: 32
LSTM_Hidden_units_4: 160
Hidden_FClayers: 2
Hidden_units_0: 416
dense_activation: tanh
Hidden_units_1: 128
Hidden_units_2: 64
Hidden_units_3: 320
Hidden_units_4: 320
learning_rate: 0.0008478087418210128
tuner/epochs: 50
tuner/initial_epoch: 17
tuner/bracket: 4
tuner/round: 3
tuner/trial_id: fb1a6c88a7ade981ef4616114245385d
Score: 0.00036288380215410143
Trial summary
Hyperparameters:
LSTM_units: 128
Hidden_LSTMlayers: 1
Conv1D_layer: False
Conv1_filters_0: 448
Conv1_activation: relu
LSTM_Hidden_units_0: 32
Conv1_filters_1: 192
LSTM_Hidden_units_1: 128
Conv1_filters_2: 160
LSTM_Hidden_units_2: 96
```

```
Conv1_filters_3: 256
LSTM_Hidden_units_3: 224
Conv1_filters_4: 320
LSTM_Hidden_units_4: 224
Hidden_FClayers: 3
Hidden_units_0: 224
dense_activation: tanh
Hidden_units_1: 448
Hidden_units_2: 64
Hidden_units_3: 256
Hidden_units_4: 256
learning_rate: 0.008080792741293362
tuner/epochs: 17
tuner/initial_epoch: 6
tuner/bracket: 4
tuner/round: 2
tuner/trial_id: 07030eb47085e34ab144238ebc2d1cf2
Score: 0.0003803435683948919
Trial summary
Hyperparameters:
LSTM_units: 128
Hidden_LSTMlayers: 1
Conv1D_layer: False
Conv1_filters_0: 448
Conv1_activation: relu
LSTM_Hidden_units_0: 32
Conv1_filters_1: 192
LSTM_Hidden_units_1: 128
Conv1_filters_2: 160
LSTM_Hidden_units_2: 96
Conv1_filters_3: 256
LSTM_Hidden_units_3: 224
Conv1_filters_4: 320
LSTM_Hidden_units_4: 224
Hidden_FClayers: 3
Hidden_units_0: 224
dense_activation: tanh
Hidden_units_1: 448
Hidden_units_2: 64
Hidden_units_3: 256
Hidden_units_4: 256
learning_rate: 0.008080792741293362
tuner/epochs: 50
tuner/initial_epoch: 17
tuner/bracket: 4
tuner/round: 3
tuner/trial_id: f89b0c6bbf5606ebb8ab326427075eab
Score: 0.00038101735117379576
```

```
Trial summary
Hyperparameters:
LSTM_units: 288
Hidden_LSTMlayers: 2
Conv1D_layer: False
Conv1_filters_0: 160
Conv1_activation: tanh
LSTM_Hidden_units_0: 96
Conv1_filters_1: 512
LSTM_Hidden_units_1: 192
Conv1_filters_2: 480
LSTM_Hidden_units_2: 128
Conv1_filters_3: 192
LSTM_Hidden_units_3: 64
Conv1_filters_4: 256
LSTM_Hidden_units_4: 192
Hidden_FClayers: 2
Hidden_units_0: 128
dense_activation: relu
Hidden_units_1: 192
Hidden_units_2: 352
Hidden_units_3: 224
Hidden_units_4: 352
learning_rate: 0.0017615796247596443
tuner/epochs: 50
tuner/initial_epoch: 17
tuner/bracket: 3
tuner/round: 2
tuner/trial_id: 34c4ed008608df6515f7bd8128c32be5
Score: 0.000404855003580451
```

[47]:
```
EPOCHS = 150
bestHPs = tuner.get_best_hyperparameters(5)
models = []
logs = []
I = 'Tuner'
for i, bestHP in enumerate(bestHPs):
    model = tuner.hypermodel.build(bestHP)
    history = model.fit(  dataset_train.repeat(),
                epochs=EPOCHS,
                steps_per_epoch = len(X_train),
                verbose=1,
                validation_data = dataset_val.repeat(),
                validation_steps= len(X_val)
            )
    model.save(f'../Models/{dataset_name}_{I}_{i}.h5',  include_optimizer=False)
    print(f'../Models/{dataset_name}_{I}_{i}.h5')
```

```
    models.append(model)
    logs.append(history)
```

```
Epoch 1/250
140/140 [==============================] - 58s 393ms/step - loss: 0.0044 -
mean_squared_error: 0.0044 - val_loss: 0.0036 - val_mean_squared_error: 0.0036
Epoch 2/250
140/140 [==============================] - 52s 372ms/step - loss: 0.0016 -
mean_squared_error: 0.0016 - val_loss: 0.0018 - val_mean_squared_error: 0.0018
Epoch 3/250
140/140 [==============================] - 54s 389ms/step - loss: 0.0012 -
mean_squared_error: 0.0012 - val_loss: 0.0012 - val_mean_squared_error: 0.0012
Epoch 4/250
140/140 [==============================] - 55s 394ms/step - loss: 9.2730e-04 -
mean_squared_error: 9.2730e-04 - val_loss: 0.0011 - val_mean_squared_error:
0.0011
Epoch 5/250
140/140 [==============================] - 57s 409ms/step - loss: 8.7532e-04 -
mean_squared_error: 8.7532e-04 - val_loss: 0.0013 - val_mean_squared_error:
0.0013
Epoch 6/250
140/140 [==============================] - 55s 392ms/step - loss: 7.3124e-04 -
mean_squared_error: 7.3124e-04 - val_loss: 0.0014 - val_mean_squared_error:
0.0014
Epoch 7/250
140/140 [==============================] - 55s 393ms/step - loss: 7.8121e-04 -
mean_squared_error: 7.8121e-04 - val_loss: 9.4486e-04 - val_mean_squared_error:
9.4486e-04
Epoch 8/250
140/140 [==============================] - 55s 392ms/step - loss: 7.3918e-04 -
mean_squared_error: 7.3918e-04 - val_loss: 0.0016 - val_mean_squared_error:
0.0016
Epoch 9/250
140/140 [==============================] - 55s 391ms/step - loss: 7.0397e-04 -
mean_squared_error: 7.0397e-04 - val_loss: 0.0011 - val_mean_squared_error:
0.0011
Epoch 10/250
140/140 [==============================] - 56s 400ms/step - loss: 6.1759e-04 -
mean_squared_error: 6.1759e-04 - val_loss: 0.0011 - val_mean_squared_error:
0.0011
Epoch 11/250
140/140 [==============================] - 54s 386ms/step - loss: 6.0942e-04 -
mean_squared_error: 6.0942e-04 - val_loss: 0.0013 - val_mean_squared_error:
0.0013
Epoch 12/250
140/140 [==============================] - 54s 384ms/step - loss: 6.3362e-04 -
mean_squared_error: 6.3362e-04 - val_loss: 0.0014 - val_mean_squared_error:
0.0014
```

```
Epoch 13/250
140/140 [==============================] - 54s 385ms/step - loss: 6.6582e-04 -
mean_squared_error: 6.6582e-04 - val_loss: 8.8052e-04 - val_mean_squared_error:
8.8052e-04
Epoch 14/250
140/140 [==============================] - 55s 390ms/step - loss: 5.9407e-04 -
mean_squared_error: 5.9407e-04 - val_loss: 9.9173e-04 - val_mean_squared_error:
9.9173e-04
Epoch 15/250
140/140 [==============================] - 54s 386ms/step - loss: 5.8243e-04 -
mean_squared_error: 5.8243e-04 - val_loss: 0.0011 - val_mean_squared_error:
0.0011
Epoch 16/250
140/140 [==============================] - 56s 399ms/step - loss: 6.0110e-04 -
mean_squared_error: 6.0110e-04 - val_loss: 9.2219e-04 - val_mean_squared_error:
9.2219e-04
Epoch 17/250
140/140 [==============================] - 56s 401ms/step - loss: 5.6739e-04 -
mean_squared_error: 5.6739e-04 - val_loss: 7.8815e-04 - val_mean_squared_error:
7.8815e-04
Epoch 18/250
140/140 [==============================] - 53s 377ms/step - loss: 5.5281e-04 -
mean_squared_error: 5.5281e-04 - val_loss: 7.7260e-04 - val_mean_squared_error:
7.7260e-04
Epoch 19/250
140/140 [==============================] - 57s 407ms/step - loss: 5.5388e-04 -
mean_squared_error: 5.5388e-04 - val_loss: 9.7347e-04 - val_mean_squared_error:
9.7347e-04
Epoch 20/250
140/140 [==============================] - 55s 390ms/step - loss: 7.0297e-04 -
mean_squared_error: 7.0297e-04 - val_loss: 0.0011 - val_mean_squared_error:
0.0011
Epoch 21/250
140/140 [==============================] - 57s 404ms/step - loss: 5.9121e-04 -
mean_squared_error: 5.9121e-04 - val_loss: 7.7394e-04 - val_mean_squared_error:
7.7394e-04
Epoch 22/250
140/140 [==============================] - 55s 391ms/step - loss: 4.8348e-04 -
mean_squared_error: 4.8348e-04 - val_loss: 7.5322e-04 - val_mean_squared_error:
7.5322e-04
Epoch 23/250
140/140 [==============================] - 56s 401ms/step - loss: 5.7952e-04 -
mean_squared_error: 5.7952e-04 - val_loss: 0.0011 - val_mean_squared_error:
0.0011
Epoch 24/250
140/140 [==============================] - 56s 399ms/step - loss: 4.9846e-04 -
mean_squared_error: 4.9846e-04 - val_loss: 9.5989e-04 - val_mean_squared_error:
9.5989e-04
```

```
Epoch 25/250
140/140 [==============================] - 56s 400ms/step - loss: 5.0662e-04 -
mean_squared_error: 5.0662e-04 - val_loss: 9.5600e-04 - val_mean_squared_error:
9.5600e-04
Epoch 26/250
140/140 [==============================] - 59s 422ms/step - loss: 5.3265e-04 -
mean_squared_error: 5.3265e-04 - val_loss: 0.0010 - val_mean_squared_error:
0.0010
Epoch 27/250
140/140 [==============================] - 57s 408ms/step - loss: 4.9956e-04 -
mean_squared_error: 4.9956e-04 - val_loss: 8.5983e-04 - val_mean_squared_error:
8.5983e-04
Epoch 28/250
140/140 [==============================] - 59s 418ms/step - loss: 4.5934e-04 -
mean_squared_error: 4.5934e-04 - val_loss: 0.0013 - val_mean_squared_error:
0.0013
Epoch 29/250
140/140 [==============================] - 59s 419ms/step - loss: 4.6772e-04 -
mean_squared_error: 4.6772e-04 - val_loss: 0.0011 - val_mean_squared_error:
0.0011
Epoch 30/250
140/140 [==============================] - 57s 405ms/step - loss: 5.2572e-04 -
mean_squared_error: 5.2572e-04 - val_loss: 9.6672e-04 - val_mean_squared_error:
9.6672e-04
Epoch 31/250
140/140 [==============================] - 59s 419ms/step - loss: 5.1214e-04 -
mean_squared_error: 5.1214e-04 - val_loss: 0.0012 - val_mean_squared_error:
0.0012
Epoch 32/250
140/140 [==============================] - 57s 407ms/step - loss: 4.7603e-04 -
mean_squared_error: 4.7603e-04 - val_loss: 8.8936e-04 - val_mean_squared_error:
8.8936e-04
Epoch 33/250
140/140 [==============================] - 60s 426ms/step - loss: 6.0325e-04 -
mean_squared_error: 6.0325e-04 - val_loss: 9.8692e-04 - val_mean_squared_error:
9.8692e-04
Epoch 34/250
140/140 [==============================] - 59s 419ms/step - loss: 5.4696e-04 -
mean_squared_error: 5.4696e-04 - val_loss: 8.2590e-04 - val_mean_squared_error:
8.2590e-04
Epoch 35/250
140/140 [==============================] - 55s 396ms/step - loss: 4.8585e-04 -
mean_squared_error: 4.8585e-04 - val_loss: 8.4676e-04 - val_mean_squared_error:
8.4676e-04
Epoch 36/250
140/140 [==============================] - 56s 403ms/step - loss: 5.0014e-04 -
mean_squared_error: 5.0014e-04 - val_loss: 5.6156e-04 - val_mean_squared_error:
5.6156e-04
```

```
Epoch 37/250
140/140 [==============================] - 56s 396ms/step - loss: 5.1892e-04 -
mean_squared_error: 5.1892e-04 - val_loss: 8.1309e-04 - val_mean_squared_error:
8.1309e-04
Epoch 38/250
140/140 [==============================] - 56s 398ms/step - loss: 4.9914e-04 -
mean_squared_error: 4.9914e-04 - val_loss: 9.2879e-04 - val_mean_squared_error:
9.2879e-04
Epoch 39/250
140/140 [==============================] - 54s 389ms/step - loss: 5.0999e-04 -
mean_squared_error: 5.0999e-04 - val_loss: 7.1492e-04 - val_mean_squared_error:
7.1492e-04
Epoch 40/250
140/140 [==============================] - 61s 438ms/step - loss: 5.2533e-04 -
mean_squared_error: 5.2533e-04 - val_loss: 7.0050e-04 - val_mean_squared_error:
7.0050e-04
Epoch 41/250
140/140 [==============================] - 56s 404ms/step - loss: 4.0881e-04 -
mean_squared_error: 4.0881e-04 - val_loss: 9.6861e-04 - val_mean_squared_error:
9.6861e-04
Epoch 42/250
140/140 [==============================] - 56s 401ms/step - loss: 3.8295e-04 -
mean_squared_error: 3.8295e-04 - val_loss: 6.6541e-04 - val_mean_squared_error:
6.6541e-04
Epoch 43/250
140/140 [==============================] - 55s 395ms/step - loss: 4.8517e-04 -
mean_squared_error: 4.8517e-04 - val_loss: 8.7637e-04 - val_mean_squared_error:
8.7637e-04
Epoch 44/250
140/140 [==============================] - 58s 414ms/step - loss: 5.1094e-04 -
mean_squared_error: 5.1094e-04 - val_loss: 8.9357e-04 - val_mean_squared_error:
8.9357e-04
Epoch 45/250
140/140 [==============================] - 58s 417ms/step - loss: 4.4491e-04 -
mean_squared_error: 4.4491e-04 - val_loss: 6.5753e-04 - val_mean_squared_error:
6.5753e-04
Epoch 46/250
140/140 [==============================] - 57s 404ms/step - loss: 4.3016e-04 -
mean_squared_error: 4.3016e-04 - val_loss: 9.6693e-04 - val_mean_squared_error:
9.6693e-04
Epoch 47/250
140/140 [==============================] - 57s 405ms/step - loss: 4.3935e-04 -
mean_squared_error: 4.3935e-04 - val_loss: 8.2024e-04 - val_mean_squared_error:
8.2024e-04
Epoch 48/250
140/140 [==============================] - 56s 398ms/step - loss: 3.7709e-04 -
mean_squared_error: 3.7709e-04 - val_loss: 8.0742e-04 - val_mean_squared_error:
8.0742e-04
```

```
Epoch 49/250
140/140 [==============================] - 56s 399ms/step - loss: 4.9513e-04 -
mean_squared_error: 4.9513e-04 - val_loss: 6.4758e-04 - val_mean_squared_error:
6.4758e-04
Epoch 50/250
140/140 [==============================] - 57s 405ms/step - loss: 4.3354e-04 -
mean_squared_error: 4.3354e-04 - val_loss: 7.6193e-04 - val_mean_squared_error:
7.6193e-04
Epoch 51/250
140/140 [==============================] - 58s 417ms/step - loss: 4.9146e-04 -
mean_squared_error: 4.9146e-04 - val_loss: 7.6045e-04 - val_mean_squared_error:
7.6045e-04
Epoch 52/250
140/140 [==============================] - 56s 402ms/step - loss: 4.4393e-04 -
mean_squared_error: 4.4393e-04 - val_loss: 7.6561e-04 - val_mean_squared_error:
7.6561e-04
Epoch 53/250
140/140 [==============================] - 58s 411ms/step - loss: 4.6491e-04 -
mean_squared_error: 4.6491e-04 - val_loss: 9.0721e-04 - val_mean_squared_error:
9.0721e-04
Epoch 54/250
140/140 [==============================] - 61s 438ms/step - loss: 4.5897e-04 -
mean_squared_error: 4.5897e-04 - val_loss: 0.0010 - val_mean_squared_error:
0.0010
Epoch 55/250
140/140 [==============================] - 58s 413ms/step - loss: 4.6930e-04 -
mean_squared_error: 4.6930e-04 - val_loss: 0.0011 - val_mean_squared_error:
0.0011
Epoch 56/250
140/140 [==============================] - 56s 399ms/step - loss: 4.3455e-04 -
mean_squared_error: 4.3455e-04 - val_loss: 6.7448e-04 - val_mean_squared_error:
6.7448e-04
Epoch 57/250
140/140 [==============================] - 57s 408ms/step - loss: 4.7501e-04 -
mean_squared_error: 4.7501e-04 - val_loss: 7.9023e-04 - val_mean_squared_error:
7.9023e-04
Epoch 58/250
140/140 [==============================] - 62s 443ms/step - loss: 4.5922e-04 -
mean_squared_error: 4.5922e-04 - val_loss: 7.8044e-04 - val_mean_squared_error:
7.8044e-04
Epoch 59/250
140/140 [==============================] - 65s 463ms/step - loss: 4.4053e-04 -
mean_squared_error: 4.4053e-04 - val_loss: 7.1496e-04 - val_mean_squared_error:
7.1496e-04
Epoch 60/250
140/140 [==============================] - 57s 403ms/step - loss: 3.8563e-04 -
mean_squared_error: 3.8563e-04 - val_loss: 7.8559e-04 - val_mean_squared_error:
7.8559e-04
```

```
Epoch 61/250
140/140 [==============================] - 58s 412ms/step - loss: 3.5781e-04 -
mean_squared_error: 3.5781e-04 - val_loss: 7.4240e-04 - val_mean_squared_error:
7.4240e-04
Epoch 62/250
140/140 [==============================] - 56s 404ms/step - loss: 4.3943e-04 -
mean_squared_error: 4.3943e-04 - val_loss: 0.0010 - val_mean_squared_error:
0.0010
Epoch 63/250
140/140 [==============================] - 57s 405ms/step - loss: 3.7884e-04 -
mean_squared_error: 3.7884e-04 - val_loss: 8.6663e-04 - val_mean_squared_error:
8.6663e-04
Epoch 64/250
140/140 [==============================] - 58s 417ms/step - loss: 5.0454e-04 -
mean_squared_error: 5.0454e-04 - val_loss: 6.3653e-04 - val_mean_squared_error:
6.3653e-04
Epoch 65/250
140/140 [==============================] - 55s 397ms/step - loss: 3.8922e-04 -
mean_squared_error: 3.8922e-04 - val_loss: 6.0955e-04 - val_mean_squared_error:
6.0955e-04
Epoch 66/250
140/140 [==============================] - 56s 403ms/step - loss: 4.0430e-04 -
mean_squared_error: 4.0430e-04 - val_loss: 6.2778e-04 - val_mean_squared_error:
6.2778e-04
Epoch 67/250
140/140 [==============================] - 57s 411ms/step - loss: 3.7102e-04 -
mean_squared_error: 3.7102e-04 - val_loss: 6.6487e-04 - val_mean_squared_error:
6.6487e-04
Epoch 68/250
140/140 [==============================] - 57s 410ms/step - loss: 3.6715e-04 -
mean_squared_error: 3.6715e-04 - val_loss: 7.8082e-04 - val_mean_squared_error:
7.8082e-04
Epoch 69/250
140/140 [==============================] - 58s 415ms/step - loss: 4.4774e-04 -
mean_squared_error: 4.4774e-04 - val_loss: 6.5743e-04 - val_mean_squared_error:
6.5743e-04
Epoch 70/250
140/140 [==============================] - 56s 404ms/step - loss: 4.7002e-04 -
mean_squared_error: 4.7002e-04 - val_loss: 8.4281e-04 - val_mean_squared_error:
8.4281e-04
Epoch 71/250
140/140 [==============================] - 58s 418ms/step - loss: 4.5697e-04 -
mean_squared_error: 4.5697e-04 - val_loss: 5.8149e-04 - val_mean_squared_error:
5.8149e-04
Epoch 72/250
140/140 [==============================] - 59s 424ms/step - loss: 3.5979e-04 -
mean_squared_error: 3.5979e-04 - val_loss: 8.4199e-04 - val_mean_squared_error:
8.4199e-04
```

```
Epoch 73/250
140/140 [==============================] - 59s 420ms/step - loss: 3.9158e-04 -
mean_squared_error: 3.9158e-04 - val_loss: 8.0122e-04 - val_mean_squared_error:
8.0122e-04
Epoch 74/250
140/140 [==============================] - 58s 415ms/step - loss: 4.1176e-04 -
mean_squared_error: 4.1176e-04 - val_loss: 7.4259e-04 - val_mean_squared_error:
7.4259e-04
Epoch 75/250
140/140 [==============================] - 59s 421ms/step - loss: 3.9048e-04 -
mean_squared_error: 3.9048e-04 - val_loss: 7.7237e-04 - val_mean_squared_error:
7.7237e-04
Epoch 76/250
140/140 [==============================] - 57s 409ms/step - loss: 3.6900e-04 -
mean_squared_error: 3.6900e-04 - val_loss: 8.0118e-04 - val_mean_squared_error:
8.0118e-04
Epoch 77/250
140/140 [==============================] - 59s 419ms/step - loss: 3.6014e-04 -
mean_squared_error: 3.6014e-04 - val_loss: 7.5117e-04 - val_mean_squared_error:
7.5117e-04
Epoch 78/250
140/140 [==============================] - 58s 415ms/step - loss: 3.9307e-04 -
mean_squared_error: 3.9307e-04 - val_loss: 6.2659e-04 - val_mean_squared_error:
6.2659e-04
Epoch 79/250
140/140 [==============================] - 57s 408ms/step - loss: 4.0928e-04 -
mean_squared_error: 4.0928e-04 - val_loss: 8.9224e-04 - val_mean_squared_error:
8.9224e-04
Epoch 80/250
140/140 [==============================] - 58s 415ms/step - loss: 3.3651e-04 -
mean_squared_error: 3.3651e-04 - val_loss: 7.8593e-04 - val_mean_squared_error:
7.8593e-04
Epoch 81/250
140/140 [==============================] - 56s 398ms/step - loss: 3.6831e-04 -
mean_squared_error: 3.6831e-04 - val_loss: 7.5252e-04 - val_mean_squared_error:
7.5252e-04
Epoch 82/250
140/140 [==============================] - 56s 399ms/step - loss: 4.1665e-04 -
mean_squared_error: 4.1665e-04 - val_loss: 6.2060e-04 - val_mean_squared_error:
6.2060e-04
Epoch 83/250
140/140 [==============================] - 61s 432ms/step - loss: 3.5533e-04 -
mean_squared_error: 3.5533e-04 - val_loss: 6.9028e-04 - val_mean_squared_error:
6.9028e-04
Epoch 84/250
140/140 [==============================] - 60s 427ms/step - loss: 3.3823e-04 -
mean_squared_error: 3.3823e-04 - val_loss: 9.1992e-04 - val_mean_squared_error:
9.1992e-04
```

```
Epoch 85/250
140/140 [==============================] - 56s 403ms/step - loss: 3.6676e-04 -
mean_squared_error: 3.6676e-04 - val_loss: 5.2520e-04 - val_mean_squared_error:
5.2520e-04
Epoch 86/250
140/140 [==============================] - 58s 413ms/step - loss: 3.7788e-04 -
mean_squared_error: 3.7788e-04 - val_loss: 9.3478e-04 - val_mean_squared_error:
9.3478e-04
Epoch 87/250
140/140 [==============================] - 60s 431ms/step - loss: 3.6811e-04 -
mean_squared_error: 3.6811e-04 - val_loss: 7.8105e-04 - val_mean_squared_error:
7.8105e-04
Epoch 88/250
140/140 [==============================] - 56s 397ms/step - loss: 3.3949e-04 -
mean_squared_error: 3.3949e-04 - val_loss: 7.8545e-04 - val_mean_squared_error:
7.8545e-04
Epoch 89/250
140/140 [==============================] - 56s 402ms/step - loss: 3.1619e-04 -
mean_squared_error: 3.1619e-04 - val_loss: 7.1722e-04 - val_mean_squared_error:
7.1722e-04
Epoch 90/250
140/140 [==============================] - 55s 395ms/step - loss: 3.6526e-04 -
mean_squared_error: 3.6526e-04 - val_loss: 6.2032e-04 - val_mean_squared_error:
6.2032e-04
Epoch 91/250
140/140 [==============================] - 58s 413ms/step - loss: 3.1974e-04 -
mean_squared_error: 3.1974e-04 - val_loss: 9.7495e-04 - val_mean_squared_error:
9.7495e-04
Epoch 92/250
140/140 [==============================] - 57s 408ms/step - loss: 3.2718e-04 -
mean_squared_error: 3.2718e-04 - val_loss: 7.2702e-04 - val_mean_squared_error:
7.2702e-04
Epoch 93/250
140/140 [==============================] - 63s 451ms/step - loss: 3.5729e-04 -
mean_squared_error: 3.5729e-04 - val_loss: 6.3058e-04 - val_mean_squared_error:
6.3058e-04
Epoch 94/250
140/140 [==============================] - 59s 423ms/step - loss: 3.3451e-04 -
mean_squared_error: 3.3451e-04 - val_loss: 4.6428e-04 - val_mean_squared_error:
4.6428e-04
Epoch 95/250
140/140 [==============================] - 57s 410ms/step - loss: 3.0916e-04 -
mean_squared_error: 3.0916e-04 - val_loss: 0.0011 - val_mean_squared_error:
0.0011
Epoch 96/250
140/140 [==============================] - 56s 401ms/step - loss: 3.3311e-04 -
mean_squared_error: 3.3311e-04 - val_loss: 6.7494e-04 - val_mean_squared_error:
6.7494e-04
```

```
Epoch 97/250
140/140 [==============================] - 62s 441ms/step - loss: 4.0816e-04 -
mean_squared_error: 4.0816e-04 - val_loss: 8.7348e-04 - val_mean_squared_error:
8.7348e-04
Epoch 98/250
140/140 [==============================] - 57s 410ms/step - loss: 3.6273e-04 -
mean_squared_error: 3.6273e-04 - val_loss: 8.5619e-04 - val_mean_squared_error:
8.5619e-04
Epoch 99/250
140/140 [==============================] - 58s 411ms/step - loss: 3.2295e-04 -
mean_squared_error: 3.2295e-04 - val_loss: 8.6205e-04 - val_mean_squared_error:
8.6205e-04
Epoch 100/250
140/140 [==============================] - 59s 423ms/step - loss: 3.1464e-04 -
mean_squared_error: 3.1464e-04 - val_loss: 6.7314e-04 - val_mean_squared_error:
6.7314e-04
Epoch 101/250
140/140 [==============================] - 57s 407ms/step - loss: 3.1372e-04 -
mean_squared_error: 3.1372e-04 - val_loss: 6.4211e-04 - val_mean_squared_error:
6.4211e-04
Epoch 102/250
140/140 [==============================] - 59s 418ms/step - loss: 2.9631e-04 -
mean_squared_error: 2.9631e-04 - val_loss: 6.3048e-04 - val_mean_squared_error:
6.3048e-04
Epoch 103/250
140/140 [==============================] - 58s 417ms/step - loss: 3.3239e-04 -
mean_squared_error: 3.3239e-04 - val_loss: 6.0260e-04 - val_mean_squared_error:
6.0260e-04
Epoch 104/250
140/140 [==============================] - 61s 438ms/step - loss: 3.7834e-04 -
mean_squared_error: 3.7834e-04 - val_loss: 7.4056e-04 - val_mean_squared_error:
7.4056e-04
Epoch 105/250
140/140 [==============================] - 57s 404ms/step - loss: 3.5967e-04 -
mean_squared_error: 3.5967e-04 - val_loss: 8.5256e-04 - val_mean_squared_error:
8.5256e-04
Epoch 106/250
140/140 [==============================] - 58s 412ms/step - loss: 3.0838e-04 -
mean_squared_error: 3.0838e-04 - val_loss: 7.4153e-04 - val_mean_squared_error:
7.4153e-04
Epoch 107/250
140/140 [==============================] - 56s 399ms/step - loss: 3.9921e-04 -
mean_squared_error: 3.9921e-04 - val_loss: 7.3467e-04 - val_mean_squared_error:
7.3467e-04
Epoch 108/250
140/140 [==============================] - 59s 421ms/step - loss: 3.4743e-04 -
mean_squared_error: 3.4743e-04 - val_loss: 9.0027e-04 - val_mean_squared_error:
9.0027e-04
```

```
Epoch 109/250
140/140 [==============================] - 55s 395ms/step - loss: 3.2057e-04 -
mean_squared_error: 3.2057e-04 - val_loss: 7.0800e-04 - val_mean_squared_error:
7.0800e-04
Epoch 110/250
140/140 [==============================] - 58s 411ms/step - loss: 3.4034e-04 -
mean_squared_error: 3.4034e-04 - val_loss: 5.1643e-04 - val_mean_squared_error:
5.1643e-04
Epoch 111/250
140/140 [==============================] - 58s 416ms/step - loss: 3.3216e-04 -
mean_squared_error: 3.3216e-04 - val_loss: 6.4328e-04 - val_mean_squared_error:
6.4328e-04
Epoch 112/250
140/140 [==============================] - 59s 426ms/step - loss: 3.4229e-04 -
mean_squared_error: 3.4229e-04 - val_loss: 7.7691e-04 - val_mean_squared_error:
7.7691e-04
Epoch 113/250
140/140 [==============================] - 60s 425ms/step - loss: 3.0333e-04 -
mean_squared_error: 3.0333e-04 - val_loss: 6.9174e-04 - val_mean_squared_error:
6.9174e-04
Epoch 114/250
140/140 [==============================] - 60s 427ms/step - loss: 3.0968e-04 -
mean_squared_error: 3.0968e-04 - val_loss: 8.6470e-04 - val_mean_squared_error:
8.6470e-04
Epoch 115/250
140/140 [==============================] - 60s 430ms/step - loss: 3.1628e-04 -
mean_squared_error: 3.1628e-04 - val_loss: 6.9782e-04 - val_mean_squared_error:
6.9782e-04
Epoch 116/250
140/140 [==============================] - 61s 434ms/step - loss: 3.3830e-04 -
mean_squared_error: 3.3830e-04 - val_loss: 6.8361e-04 - val_mean_squared_error:
6.8361e-04
Epoch 117/250
140/140 [==============================] - 59s 422ms/step - loss: 2.8451e-04 -
mean_squared_error: 2.8451e-04 - val_loss: 7.8010e-04 - val_mean_squared_error:
7.8010e-04
Epoch 118/250
140/140 [==============================] - 57s 411ms/step - loss: 3.8413e-04 -
mean_squared_error: 3.8413e-04 - val_loss: 6.7895e-04 - val_mean_squared_error:
6.7895e-04
Epoch 119/250
140/140 [==============================] - 57s 409ms/step - loss: 3.0491e-04 -
mean_squared_error: 3.0491e-04 - val_loss: 7.4509e-04 - val_mean_squared_error:
7.4509e-04
Epoch 120/250
140/140 [==============================] - 56s 401ms/step - loss: 2.8432e-04 -
mean_squared_error: 2.8432e-04 - val_loss: 6.8904e-04 - val_mean_squared_error:
6.8904e-04
```

```
Epoch 121/250
140/140 [==============================] - 59s 421ms/step - loss: 3.5346e-04 -
mean_squared_error: 3.5346e-04 - val_loss: 5.2428e-04 - val_mean_squared_error:
5.2428e-04
Epoch 122/250
140/140 [==============================] - 54s 387ms/step - loss: 3.2006e-04 -
mean_squared_error: 3.2006e-04 - val_loss: 6.3653e-04 - val_mean_squared_error:
6.3653e-04
Epoch 123/250
140/140 [==============================] - 56s 403ms/step - loss: 3.7756e-04 -
mean_squared_error: 3.7756e-04 - val_loss: 6.0847e-04 - val_mean_squared_error:
6.0847e-04
Epoch 124/250
140/140 [==============================] - 58s 411ms/step - loss: 4.0715e-04 -
mean_squared_error: 4.0715e-04 - val_loss: 9.2913e-04 - val_mean_squared_error:
9.2913e-04
Epoch 125/250
140/140 [==============================] - 56s 398ms/step - loss: 2.9814e-04 -
mean_squared_error: 2.9814e-04 - val_loss: 0.0016 - val_mean_squared_error:
0.0016
Epoch 126/250
140/140 [==============================] - 55s 394ms/step - loss: 3.4980e-04 -
mean_squared_error: 3.4980e-04 - val_loss: 7.7508e-04 - val_mean_squared_error:
7.7508e-04
Epoch 127/250
140/140 [==============================] - 55s 395ms/step - loss: 3.4361e-04 -
mean_squared_error: 3.4361e-04 - val_loss: 7.4165e-04 - val_mean_squared_error:
7.4165e-04
Epoch 128/250
140/140 [==============================] - 54s 389ms/step - loss: 3.2670e-04 -
mean_squared_error: 3.2670e-04 - val_loss: 5.4719e-04 - val_mean_squared_error:
5.4719e-04
Epoch 129/250
140/140 [==============================] - 56s 403ms/step - loss: 2.8709e-04 -
mean_squared_error: 2.8709e-04 - val_loss: 6.8889e-04 - val_mean_squared_error:
6.8889e-04
Epoch 130/250
140/140 [==============================] - 56s 401ms/step - loss: 3.1459e-04 -
mean_squared_error: 3.1459e-04 - val_loss: 5.1764e-04 - val_mean_squared_error:
5.1764e-04
Epoch 131/250
140/140 [==============================] - 59s 426ms/step - loss: 3.4192e-04 -
mean_squared_error: 3.4192e-04 - val_loss: 6.9387e-04 - val_mean_squared_error:
6.9387e-04
Epoch 132/250
140/140 [==============================] - 56s 404ms/step - loss: 3.4433e-04 -
mean_squared_error: 3.4433e-04 - val_loss: 6.7135e-04 - val_mean_squared_error:
6.7135e-04
```

```
Epoch 133/250
140/140 [==============================] - 57s 410ms/step - loss: 3.0408e-04 -
mean_squared_error: 3.0408e-04 - val_loss: 7.8591e-04 - val_mean_squared_error:
7.8591e-04
Epoch 134/250
140/140 [==============================] - 55s 394ms/step - loss: 3.1689e-04 -
mean_squared_error: 3.1689e-04 - val_loss: 0.0010 - val_mean_squared_error:
0.0010
Epoch 135/250
140/140 [==============================] - 55s 393ms/step - loss: 4.0478e-04 -
mean_squared_error: 4.0478e-04 - val_loss: 0.0010 - val_mean_squared_error:
0.0010
Epoch 136/250
140/140 [==============================] - 57s 406ms/step - loss: 3.8729e-04 -
mean_squared_error: 3.8729e-04 - val_loss: 8.8253e-04 - val_mean_squared_error:
8.8253e-04
Epoch 137/250
140/140 [==============================] - 57s 411ms/step - loss: 3.7741e-04 -
mean_squared_error: 3.7741e-04 - val_loss: 7.4766e-04 - val_mean_squared_error:
7.4766e-04
Epoch 138/250
140/140 [==============================] - 59s 424ms/step - loss: 3.4232e-04 -
mean_squared_error: 3.4232e-04 - val_loss: 8.0283e-04 - val_mean_squared_error:
8.0283e-04
Epoch 139/250
140/140 [==============================] - 57s 406ms/step - loss: 3.1491e-04 -
mean_squared_error: 3.1491e-04 - val_loss: 7.5289e-04 - val_mean_squared_error:
7.5289e-04
Epoch 140/250
140/140 [==============================] - 61s 435ms/step - loss: 3.2148e-04 -
mean_squared_error: 3.2148e-04 - val_loss: 7.6344e-04 - val_mean_squared_error:
7.6344e-04
Epoch 141/250
140/140 [==============================] - 58s 411ms/step - loss: 2.7260e-04 -
mean_squared_error: 2.7260e-04 - val_loss: 5.5151e-04 - val_mean_squared_error:
5.5151e-04
Epoch 142/250
140/140 [==============================] - 58s 415ms/step - loss: 3.0779e-04 -
mean_squared_error: 3.0779e-04 - val_loss: 8.4240e-04 - val_mean_squared_error:
8.4240e-04
Epoch 143/250
140/140 [==============================] - 58s 418ms/step - loss: 2.8386e-04 -
mean_squared_error: 2.8386e-04 - val_loss: 7.0422e-04 - val_mean_squared_error:
7.0422e-04
Epoch 144/250
140/140 [==============================] - 58s 415ms/step - loss: 2.4945e-04 -
mean_squared_error: 2.4945e-04 - val_loss: 7.5668e-04 - val_mean_squared_error:
7.5668e-04
```

```
Epoch 145/250
140/140 [==============================] - 57s 408ms/step - loss: 3.7356e-04 -
mean_squared_error: 3.7356e-04 - val_loss: 9.8186e-04 - val_mean_squared_error:
9.8186e-04
Epoch 146/250
140/140 [==============================] - 61s 435ms/step - loss: 4.4625e-04 -
mean_squared_error: 4.4625e-04 - val_loss: 5.8290e-04 - val_mean_squared_error:
5.8290e-04
Epoch 147/250
140/140 [==============================] - 58s 410ms/step - loss: 3.4593e-04 -
mean_squared_error: 3.4593e-04 - val_loss: 6.9454e-04 - val_mean_squared_error:
6.9454e-04
Epoch 148/250
140/140 [==============================] - 60s 431ms/step - loss: 3.1149e-04 -
mean_squared_error: 3.1149e-04 - val_loss: 7.2727e-04 - val_mean_squared_error:
7.2727e-04
Epoch 149/250
140/140 [==============================] - 58s 416ms/step - loss: 2.9335e-04 -
mean_squared_error: 2.9335e-04 - val_loss: 5.5418e-04 - val_mean_squared_error:
5.5418e-04
Epoch 150/250
140/140 [==============================] - 56s 400ms/step - loss: 3.1696e-04 -
mean_squared_error: 3.1696e-04 - val_loss: 7.2713e-04 - val_mean_squared_error:
7.2713e-04
Epoch 151/250
140/140 [==============================] - 61s 433ms/step - loss: 3.0661e-04 -
mean_squared_error: 3.0661e-04 - val_loss: 6.9000e-04 - val_mean_squared_error:
6.9000e-04
Epoch 152/250
140/140 [==============================] - 56s 403ms/step - loss: 2.7745e-04 -
mean_squared_error: 2.7745e-04 - val_loss: 8.0455e-04 - val_mean_squared_error:
8.0455e-04
Epoch 153/250
140/140 [==============================] - 56s 398ms/step - loss: 3.1689e-04 -
mean_squared_error: 3.1689e-04 - val_loss: 7.2201e-04 - val_mean_squared_error:
7.2201e-04
Epoch 154/250
140/140 [==============================] - 55s 393ms/step - loss: 2.8852e-04 -
mean_squared_error: 2.8852e-04 - val_loss: 7.4044e-04 - val_mean_squared_error:
7.4044e-04
Epoch 155/250
140/140 [==============================] - 57s 409ms/step - loss: 2.9902e-04 -
mean_squared_error: 2.9902e-04 - val_loss: 8.9095e-04 - val_mean_squared_error:
8.9095e-04
Epoch 156/250
140/140 [==============================] - 57s 405ms/step - loss: 2.8536e-04 -
mean_squared_error: 2.8536e-04 - val_loss: 0.0010 - val_mean_squared_error:
0.0010
```

```
Epoch 157/250
140/140 [==============================] - 56s 401ms/step - loss: 3.6948e-04 -
mean_squared_error: 3.6948e-04 - val_loss: 6.9546e-04 - val_mean_squared_error:
6.9546e-04
Epoch 158/250
140/140 [==============================] - 58s 415ms/step - loss: 3.3136e-04 -
mean_squared_error: 3.3136e-04 - val_loss: 5.6858e-04 - val_mean_squared_error:
5.6858e-04
Epoch 159/250
140/140 [==============================] - 57s 405ms/step - loss: 3.0637e-04 -
mean_squared_error: 3.0637e-04 - val_loss: 8.1921e-04 - val_mean_squared_error:
8.1921e-04
Epoch 160/250
140/140 [==============================] - 57s 407ms/step - loss: 2.9016e-04 -
mean_squared_error: 2.9016e-04 - val_loss: 6.8450e-04 - val_mean_squared_error:
6.8450e-04
Epoch 161/250
140/140 [==============================] - 55s 395ms/step - loss: 2.7786e-04 -
mean_squared_error: 2.7786e-04 - val_loss: 6.9942e-04 - val_mean_squared_error:
6.9942e-04
Epoch 162/250
140/140 [==============================] - 55s 391ms/step - loss: 2.6122e-04 -
mean_squared_error: 2.6122e-04 - val_loss: 7.4440e-04 - val_mean_squared_error:
7.4440e-04
Epoch 163/250
140/140 [==============================] - 58s 413ms/step - loss: 3.3601e-04 -
mean_squared_error: 3.3601e-04 - val_loss: 7.4242e-04 - val_mean_squared_error:
7.4242e-04
Epoch 164/250
140/140 [==============================] - 61s 439ms/step - loss: 3.3868e-04 -
mean_squared_error: 3.3868e-04 - val_loss: 7.1125e-04 - val_mean_squared_error:
7.1125e-04
Epoch 165/250
140/140 [==============================] - 58s 417ms/step - loss: 3.1068e-04 -
mean_squared_error: 3.1068e-04 - val_loss: 6.1257e-04 - val_mean_squared_error:
6.1257e-04
Epoch 166/250
140/140 [==============================] - 56s 399ms/step - loss: 3.0944e-04 -
mean_squared_error: 3.0944e-04 - val_loss: 7.4061e-04 - val_mean_squared_error:
7.4061e-04
Epoch 167/250
140/140 [==============================] - 56s 403ms/step - loss: 2.7768e-04 -
mean_squared_error: 2.7768e-04 - val_loss: 9.5696e-04 - val_mean_squared_error:
9.5696e-04
Epoch 168/250
140/140 [==============================] - 58s 418ms/step - loss: 3.4807e-04 -
mean_squared_error: 3.4807e-04 - val_loss: 0.0010 - val_mean_squared_error:
0.0010
```

```
Epoch 169/250
140/140 [==============================] - 59s 419ms/step - loss: 3.3591e-04 -
mean_squared_error: 3.3591e-04 - val_loss: 8.2271e-04 - val_mean_squared_error:
8.2271e-04
Epoch 170/250
140/140 [==============================] - 59s 424ms/step - loss: 2.7722e-04 -
mean_squared_error: 2.7722e-04 - val_loss: 6.2358e-04 - val_mean_squared_error:
6.2358e-04
Epoch 171/250
140/140 [==============================] - 57s 407ms/step - loss: 2.5950e-04 -
mean_squared_error: 2.5950e-04 - val_loss: 5.0774e-04 - val_mean_squared_error:
5.0774e-04
Epoch 172/250
140/140 [==============================] - 60s 426ms/step - loss: 3.0637e-04 -
mean_squared_error: 3.0637e-04 - val_loss: 4.3644e-04 - val_mean_squared_error:
4.3644e-04
Epoch 173/250
140/140 [==============================] - 57s 409ms/step - loss: 2.9793e-04 -
mean_squared_error: 2.9793e-04 - val_loss: 4.7644e-04 - val_mean_squared_error:
4.7644e-04
Epoch 174/250
140/140 [==============================] - 57s 405ms/step - loss: 3.1074e-04 -
mean_squared_error: 3.1074e-04 - val_loss: 7.2169e-04 - val_mean_squared_error:
7.2169e-04
Epoch 175/250
140/140 [==============================] - 58s 415ms/step - loss: 2.8341e-04 -
mean_squared_error: 2.8341e-04 - val_loss: 7.7230e-04 - val_mean_squared_error:
7.7230e-04
Epoch 176/250
140/140 [==============================] - 58s 411ms/step - loss: 2.8772e-04 -
mean_squared_error: 2.8772e-04 - val_loss: 0.0016 - val_mean_squared_error:
0.0016
Epoch 177/250
140/140 [==============================] - 60s 427ms/step - loss: 4.5904e-04 -
mean_squared_error: 4.5904e-04 - val_loss: 6.6820e-04 - val_mean_squared_error:
6.6820e-04
Epoch 178/250
140/140 [==============================] - 56s 401ms/step - loss: 3.5800e-04 -
mean_squared_error: 3.5800e-04 - val_loss: 6.5770e-04 - val_mean_squared_error:
6.5770e-04
Epoch 179/250
140/140 [==============================] - 59s 422ms/step - loss: 2.3138e-04 -
mean_squared_error: 2.3138e-04 - val_loss: 6.6163e-04 - val_mean_squared_error:
6.6163e-04
Epoch 180/250
140/140 [==============================] - 56s 398ms/step - loss: 2.8622e-04 -
mean_squared_error: 2.8622e-04 - val_loss: 7.4806e-04 - val_mean_squared_error:
7.4806e-04
```

```
Epoch 181/250
140/140 [==============================] - 56s 403ms/step - loss: 2.9757e-04 -
mean_squared_error: 2.9757e-04 - val_loss: 5.5160e-04 - val_mean_squared_error:
5.5160e-04
Epoch 182/250
140/140 [==============================] - 58s 417ms/step - loss: 2.6431e-04 -
mean_squared_error: 2.6431e-04 - val_loss: 7.0262e-04 - val_mean_squared_error:
7.0262e-04
Epoch 183/250
140/140 [==============================] - 60s 431ms/step - loss: 3.1127e-04 -
mean_squared_error: 3.1127e-04 - val_loss: 5.6846e-04 - val_mean_squared_error:
5.6846e-04
Epoch 184/250
140/140 [==============================] - 61s 438ms/step - loss: 3.0244e-04 -
mean_squared_error: 3.0244e-04 - val_loss: 4.7743e-04 - val_mean_squared_error:
4.7743e-04
Epoch 185/250
140/140 [==============================] - 56s 402ms/step - loss: 2.5604e-04 -
mean_squared_error: 2.5604e-04 - val_loss: 6.4965e-04 - val_mean_squared_error:
6.4965e-04
Epoch 186/250
140/140 [==============================] - 57s 410ms/step - loss: 2.2621e-04 -
mean_squared_error: 2.2621e-04 - val_loss: 6.3308e-04 - val_mean_squared_error:
6.3308e-04
Epoch 187/250
140/140 [==============================] - 58s 418ms/step - loss: 3.1293e-04 -
mean_squared_error: 3.1293e-04 - val_loss: 9.1942e-04 - val_mean_squared_error:
9.1942e-04
Epoch 188/250
140/140 [==============================] - 59s 419ms/step - loss: 2.5976e-04 -
mean_squared_error: 2.5976e-04 - val_loss: 8.1788e-04 - val_mean_squared_error:
8.1788e-04
Epoch 189/250
140/140 [==============================] - 60s 429ms/step - loss: 3.0468e-04 -
mean_squared_error: 3.0468e-04 - val_loss: 9.7914e-04 - val_mean_squared_error:
9.7914e-04
Epoch 190/250
140/140 [==============================] - 58s 412ms/step - loss: 3.0962e-04 -
mean_squared_error: 3.0962e-04 - val_loss: 8.8969e-04 - val_mean_squared_error:
8.8969e-04
Epoch 191/250
140/140 [==============================] - 56s 400ms/step - loss: 2.9913e-04 -
mean_squared_error: 2.9913e-04 - val_loss: 6.9756e-04 - val_mean_squared_error:
6.9756e-04
Epoch 192/250
140/140 [==============================] - 59s 424ms/step - loss: 2.8844e-04 -
mean_squared_error: 2.8844e-04 - val_loss: 7.2889e-04 - val_mean_squared_error:
7.2889e-04
```

```
Epoch 193/250
140/140 [==============================] - 61s 436ms/step - loss: 2.8414e-04 -
mean_squared_error: 2.8414e-04 - val_loss: 6.7858e-04 - val_mean_squared_error:
6.7858e-04
Epoch 194/250
140/140 [==============================] - 58s 412ms/step - loss: 2.9010e-04 -
mean_squared_error: 2.9010e-04 - val_loss: 5.8900e-04 - val_mean_squared_error:
5.8900e-04
Epoch 195/250
140/140 [==============================] - 59s 420ms/step - loss: 2.9705e-04 -
mean_squared_error: 2.9705e-04 - val_loss: 8.7276e-04 - val_mean_squared_error:
8.7276e-04
Epoch 196/250
140/140 [==============================] - 56s 399ms/step - loss: 3.1326e-04 -
mean_squared_error: 3.1326e-04 - val_loss: 6.8726e-04 - val_mean_squared_error:
6.8726e-04
Epoch 197/250
140/140 [==============================] - 61s 440ms/step - loss: 2.4827e-04 -
mean_squared_error: 2.4827e-04 - val_loss: 5.8845e-04 - val_mean_squared_error:
5.8845e-04
Epoch 198/250
140/140 [==============================] - 57s 408ms/step - loss: 2.8528e-04 -
mean_squared_error: 2.8528e-04 - val_loss: 8.3256e-04 - val_mean_squared_error:
8.3256e-04
Epoch 199/250
140/140 [==============================] - 56s 397ms/step - loss: 3.0687e-04 -
mean_squared_error: 3.0687e-04 - val_loss: 6.2955e-04 - val_mean_squared_error:
6.2955e-04
Epoch 200/250
140/140 [==============================] - 57s 407ms/step - loss: 3.4281e-04 -
mean_squared_error: 3.4281e-04 - val_loss: 6.0489e-04 - val_mean_squared_error:
6.0489e-04
Epoch 201/250
140/140 [==============================] - 58s 418ms/step - loss: 3.3605e-04 -
mean_squared_error: 3.3605e-04 - val_loss: 0.0013 - val_mean_squared_error:
0.0013
Epoch 202/250
140/140 [==============================] - 60s 427ms/step - loss: 3.0855e-04 -
mean_squared_error: 3.0855e-04 - val_loss: 0.0022 - val_mean_squared_error:
0.0022
Epoch 203/250
140/140 [==============================] - 60s 433ms/step - loss: 3.1921e-04 -
mean_squared_error: 3.1921e-04 - val_loss: 9.5059e-04 - val_mean_squared_error:
9.5059e-04
Epoch 204/250
140/140 [==============================] - 59s 420ms/step - loss: 3.1870e-04 -
mean_squared_error: 3.1870e-04 - val_loss: 0.0023 - val_mean_squared_error:
0.0023
```

```
Epoch 205/250
140/140 [==============================] - 60s 426ms/step - loss: 3.2691e-04 -
mean_squared_error: 3.2691e-04 - val_loss: 8.6460e-04 - val_mean_squared_error:
8.6460e-04
Epoch 206/250
140/140 [==============================] - 58s 416ms/step - loss: 3.5213e-04 -
mean_squared_error: 3.5213e-04 - val_loss: 8.0437e-04 - val_mean_squared_error:
8.0437e-04
Epoch 207/250
140/140 [==============================] - 54s 389ms/step - loss: 2.6259e-04 -
mean_squared_error: 2.6259e-04 - val_loss: 7.8271e-04 - val_mean_squared_error:
7.8271e-04
Epoch 208/250
140/140 [==============================] - 57s 408ms/step - loss: 2.9908e-04 -
mean_squared_error: 2.9908e-04 - val_loss: 6.7639e-04 - val_mean_squared_error:
6.7639e-04
Epoch 209/250
140/140 [==============================] - 56s 401ms/step - loss: 2.9882e-04 -
mean_squared_error: 2.9882e-04 - val_loss: 8.6074e-04 - val_mean_squared_error:
8.6074e-04
Epoch 210/250
140/140 [==============================] - 58s 411ms/step - loss: 3.1891e-04 -
mean_squared_error: 3.1891e-04 - val_loss: 7.0212e-04 - val_mean_squared_error:
7.0212e-04
Epoch 211/250
140/140 [==============================] - 56s 402ms/step - loss: 3.2713e-04 -
mean_squared_error: 3.2713e-04 - val_loss: 7.7828e-04 - val_mean_squared_error:
7.7828e-04
Epoch 212/250
140/140 [==============================] - 58s 417ms/step - loss: 3.4582e-04 -
mean_squared_error: 3.4582e-04 - val_loss: 7.5946e-04 - val_mean_squared_error:
7.5946e-04
Epoch 213/250
140/140 [==============================] - 59s 424ms/step - loss: 3.3151e-04 -
mean_squared_error: 3.3151e-04 - val_loss: 8.4921e-04 - val_mean_squared_error:
8.4921e-04
Epoch 214/250
140/140 [==============================] - 57s 403ms/step - loss: 3.0695e-04 -
mean_squared_error: 3.0695e-04 - val_loss: 5.8018e-04 - val_mean_squared_error:
5.8018e-04
Epoch 215/250
140/140 [==============================] - 55s 392ms/step - loss: 3.4237e-04 -
mean_squared_error: 3.4237e-04 - val_loss: 5.3482e-04 - val_mean_squared_error:
5.3482e-04
Epoch 216/250
140/140 [==============================] - 57s 406ms/step - loss: 2.5441e-04 -
mean_squared_error: 2.5441e-04 - val_loss: 8.4982e-04 - val_mean_squared_error:
8.4982e-04
```

```
Epoch 217/250
140/140 [==============================] - 59s 422ms/step - loss: 3.4150e-04 -
mean_squared_error: 3.4150e-04 - val_loss: 8.5238e-04 - val_mean_squared_error:
8.5238e-04
Epoch 218/250
140/140 [==============================] - 60s 429ms/step - loss: 2.4841e-04 -
mean_squared_error: 2.4841e-04 - val_loss: 6.5906e-04 - val_mean_squared_error:
6.5906e-04
Epoch 219/250
140/140 [==============================] - 74s 532ms/step - loss: 3.4133e-04 -
mean_squared_error: 3.4133e-04 - val_loss: 8.5479e-04 - val_mean_squared_error:
8.5479e-04
Epoch 220/250
140/140 [==============================] - 86s 616ms/step - loss: 2.6930e-04 -
mean_squared_error: 2.6930e-04 - val_loss: 6.2084e-04 - val_mean_squared_error:
6.2084e-04
Epoch 221/250
140/140 [==============================] - 84s 598ms/step - loss: 2.5794e-04 -
mean_squared_error: 2.5794e-04 - val_loss: 5.3420e-04 - val_mean_squared_error:
5.3420e-04
Epoch 222/250
140/140 [==============================] - 83s 596ms/step - loss: 2.9066e-04 -
mean_squared_error: 2.9066e-04 - val_loss: 7.4741e-04 - val_mean_squared_error:
7.4741e-04
Epoch 223/250
140/140 [==============================] - 84s 603ms/step - loss: 3.0527e-04 -
mean_squared_error: 3.0527e-04 - val_loss: 7.0585e-04 - val_mean_squared_error:
7.0585e-04
Epoch 224/250
140/140 [==============================] - 84s 598ms/step - loss: 2.3627e-04 -
mean_squared_error: 2.3627e-04 - val_loss: 7.4526e-04 - val_mean_squared_error:
7.4526e-04
Epoch 225/250
140/140 [==============================] - 84s 602ms/step - loss: 3.1306e-04 -
mean_squared_error: 3.1306e-04 - val_loss: 7.4500e-04 - val_mean_squared_error:
7.4500e-04
Epoch 226/250
140/140 [==============================] - 83s 596ms/step - loss: 3.0246e-04 -
mean_squared_error: 3.0246e-04 - val_loss: 7.7208e-04 - val_mean_squared_error:
7.7208e-04
Epoch 227/250
140/140 [==============================] - 85s 611ms/step - loss: 3.1184e-04 -
mean_squared_error: 3.1184e-04 - val_loss: 6.7087e-04 - val_mean_squared_error:
6.7087e-04
Epoch 228/250
140/140 [==============================] - 86s 615ms/step - loss: 2.8514e-04 -
mean_squared_error: 2.8514e-04 - val_loss: 8.1590e-04 - val_mean_squared_error:
8.1590e-04
```

```
Epoch 229/250
140/140 [==============================] - 86s 618ms/step - loss: 2.9326e-04 -
mean_squared_error: 2.9326e-04 - val_loss: 5.6527e-04 - val_mean_squared_error:
5.6527e-04
Epoch 230/250
140/140 [==============================] - 85s 608ms/step - loss: 2.9069e-04 -
mean_squared_error: 2.9069e-04 - val_loss: 4.6178e-04 - val_mean_squared_error:
4.6178e-04
Epoch 231/250
140/140 [==============================] - 87s 619ms/step - loss: 2.9200e-04 -
mean_squared_error: 2.9200e-04 - val_loss: 5.9397e-04 - val_mean_squared_error:
5.9397e-04
Epoch 232/250
140/140 [==============================] - 83s 594ms/step - loss: 2.8532e-04 -
mean_squared_error: 2.8532e-04 - val_loss: 7.5934e-04 - val_mean_squared_error:
7.5934e-04
Epoch 233/250
140/140 [==============================] - 86s 618ms/step - loss: 2.2614e-04 -
mean_squared_error: 2.2614e-04 - val_loss: 7.2765e-04 - val_mean_squared_error:
7.2765e-04
Epoch 234/250
140/140 [==============================] - 90s 643ms/step - loss: 3.1072e-04 -
mean_squared_error: 3.1072e-04 - val_loss: 6.3187e-04 - val_mean_squared_error:
6.3187e-04
Epoch 235/250
140/140 [==============================] - 86s 613ms/step - loss: 2.8553e-04 -
mean_squared_error: 2.8553e-04 - val_loss: 6.0258e-04 - val_mean_squared_error:
6.0258e-04
Epoch 236/250
140/140 [==============================] - 86s 613ms/step - loss: 2.8403e-04 -
mean_squared_error: 2.8403e-04 - val_loss: 6.2617e-04 - val_mean_squared_error:
6.2617e-04
Epoch 237/250
140/140 [==============================] - 83s 595ms/step - loss: 3.1282e-04 -
mean_squared_error: 3.1282e-04 - val_loss: 0.0011 - val_mean_squared_error:
0.0011
Epoch 238/250
140/140 [==============================] - 78s 558ms/step - loss: 2.7974e-04 -
mean_squared_error: 2.7974e-04 - val_loss: 5.3167e-04 - val_mean_squared_error:
5.3167e-04
Epoch 239/250
140/140 [==============================] - 59s 420ms/step - loss: 2.3072e-04 -
mean_squared_error: 2.3072e-04 - val_loss: 0.0013 - val_mean_squared_error:
0.0013
Epoch 240/250
140/140 [==============================] - 59s 422ms/step - loss: 2.5334e-04 -
mean_squared_error: 2.5334e-04 - val_loss: 8.3377e-04 - val_mean_squared_error:
8.3377e-04
```

```
Epoch 241/250
140/140 [==============================] - 60s 429ms/step - loss: 2.7558e-04 -
mean_squared_error: 2.7558e-04 - val_loss: 7.3946e-04 - val_mean_squared_error:
7.3946e-04
Epoch 242/250
140/140 [==============================] - 59s 425ms/step - loss: 2.6704e-04 -
mean_squared_error: 2.6704e-04 - val_loss: 0.0012 - val_mean_squared_error:
0.0012
Epoch 243/250
140/140 [==============================] - 58s 416ms/step - loss: 2.9898e-04 -
mean_squared_error: 2.9898e-04 - val_loss: 9.3749e-04 - val_mean_squared_error:
9.3749e-04
Epoch 244/250
140/140 [==============================] - 59s 421ms/step - loss: 3.1200e-04 -
mean_squared_error: 3.1200e-04 - val_loss: 7.1167e-04 - val_mean_squared_error:
7.1167e-04
Epoch 245/250
140/140 [==============================] - 59s 419ms/step - loss: 2.9253e-04 -
mean_squared_error: 2.9253e-04 - val_loss: 7.2935e-04 - val_mean_squared_error:
7.2935e-04
Epoch 246/250
140/140 [==============================] - 58s 417ms/step - loss: 2.9795e-04 -
mean_squared_error: 2.9795e-04 - val_loss: 7.5003e-04 - val_mean_squared_error:
7.5003e-04
Epoch 247/250
140/140 [==============================] - 59s 422ms/step - loss: 2.6536e-04 -
mean_squared_error: 2.6536e-04 - val_loss: 6.8446e-04 - val_mean_squared_error:
6.8446e-04
Epoch 248/250
140/140 [==============================] - 59s 419ms/step - loss: 2.9999e-04 -
mean_squared_error: 2.9999e-04 - val_loss: 6.6603e-04 - val_mean_squared_error:
6.6603e-04
Epoch 249/250
140/140 [==============================] - 60s 428ms/step - loss: 2.5324e-04 -
mean_squared_error: 2.5324e-04 - val_loss: 5.9516e-04 - val_mean_squared_error:
5.9516e-04
Epoch 250/250
140/140 [==============================] - 60s 427ms/step - loss: 2.5951e-04 -
mean_squared_error: 2.5951e-04 - val_loss: 8.4475e-04 - val_mean_squared_error:
8.4475e-04
../Models/Dataset_Final_Tuner_0.h5
Epoch 1/250
140/140 [==============================] - 57s 367ms/step - loss: 0.0039 -
mean_squared_error: 0.0039 - val_loss: 0.0028 - val_mean_squared_error: 0.0028
Epoch 2/250
137/140 [============================>.] - ETA: 0s - loss: 0.0017 -
mean_squared_error: 0.0017
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-47-4f8e8bde56ee> in <module>
      6 for i, bestHP in enumerate(bestHPs):
      7     model = tuner.hypermodel.build(bestHP)
----> 8     history = model.fit(  dataset_train.repeat(),

      9                 epochs=EPOCHS,
     10                 steps_per_epoch = len(X_train),

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\engine\training.py in fit(self, x, y, batch_size, epochs, verbose, callbacks, validation_split, validation_data, shuffle, class_weight, sample_weight, initial_epoch, steps_per_epoch, validation_steps, validation_batch_size, validation_freq, max_queue_size, workers, use_multiprocessing)
   1181                 _r=1):
   1182               callbacks.on_train_batch_begin(step)
-> 1183               tmp_logs = self.train_function(iterator)
   1184               if data_handler.should_sync:
   1185                 context.async_wait()

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\eager\def_function.py in __call__(self, *args, **kwds)
    887
    888       with OptionalXlaContext(self._jit_compile):
--> 889         result = self._call(*args, **kwds)
    890
    891       new_tracing_count = self.experimental_get_tracing_count()

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\eager\def_function.py in _call(self, *args, **kwds)
    915       # In this case we have created variables on the first call, so we run the
    916       # defunned version which is guaranteed to never create variables.
--> 917       return self._stateless_fn(*args, **kwds)  # pylint: disable=not-callable
    918     elif self._stateful_fn is not None:
    919       # Release the lock early so that multiple threads can perform the call

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\eager\function.py in __call__(self, *args, **kwargs)
   3021       (graph_function,
   3022        filtered_flat_args) = self._maybe_define_function(args, kwargs)
-> 3023     return graph_function._call_flat(

   3024         filtered_flat_args, captured_inputs=graph_function.captured_inputs)  # pylint: disable=protected-access
   3025
```

```
~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\eager\functio .
 ↪py in _call_flat(self, args, captured_inputs, cancellation_manager)
   1958         and executing_eagerly):
   1959       # No tape is watching; skip to running the function.
-> 1960       return self._build_call_outputs(self._inference_function.call(
   1961           ctx, args, cancellation_manager=cancellation_manager))
   1962     forward_backward = self._select_forward_and_backward_functions(

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\eager\functio .
 ↪py in call(self, ctx, args, cancellation_manager)
    589       with _InterpolateFunctionError(self):
    590         if cancellation_manager is None:
--> 591           outputs = execute.execute(
    592               str(self.signature.name),
    593               num_outputs=self._num_outputs,

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\eager\execute
 ↪py in quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
     57   try:
     58     ctx.ensure_initialized()
---> 59     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name,␣
 ↪op_name,
     60                                         inputs, attrs, num_outputs)
     61   except core._NotOkStatusException as e:

KeyboardInterrupt:
```

1.5.1 Evaluación del Modelo

Se carga el modelo

[48]:
```
model = tf.keras.models.load_model(f'../Models/{dataset_name}_{I}_0.h5')
model.summary()
```

```
WARNING:tensorflow:No training configuration found in the save file, so the
model was *not* compiled. Compile it manually.

WARNING:tensorflow:No training configuration found in the save file, so the
model was *not* compiled. Compile it manually.

Model: "sequential"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, None, 320)         439040

_________________________________________________________________
conv1d (Conv1D)              (None, None, 480)         461280
```

```
_________________________________________________________________
lstm_1 (LSTM)                (None, None, 128)         311808
_________________________________________________________________
conv1d_1 (Conv1D)            (None, None, 288)         110880
_________________________________________________________________
lstm_2 (LSTM)                (None, 128)               213504
_________________________________________________________________
dense (Dense)                (None, 64)                8256
_________________________________________________________________
dense_1 (Dense)              (None, 96)                6240
_________________________________________________________________
dense_2 (Dense)              (None, 320)               31040
_________________________________________________________________
dense_3 (Dense)              (None, 192)               61632
_________________________________________________________________
dense_4 (Dense)              (None, 4)                 772
=================================================================
Total params: 1,644,452
Trainable params: 1,644,452
Non-trainable params: 0
_________________________________________________________________
```

[50]:
```
model.compile(loss='mean_squared_error', optimizer='adam',␣
↪metrics='mean_squared_error')
```

[64]:
```
model.save(f'../Models/{dataset_name}_{I}_{0}.h5')
```

[65]:
```
print(f'../Models/{dataset_name}_{I}_{0}.h5')
```

```
../Models/Dataset_Final_Tuner_0.h5
```

Evaluación con dataset de prueba

[51]:
```
%%time
N = 0
for i in range(len(X_test)):
    N+=len(X_test[i])
N=N/len(X_test)

n_batches = np.ceil(N/batch_size)
losses = model.evaluate(dataset_test, steps = n_batches)
K = df_desc[rpm_list[0]][7]-df_desc[rpm_list[0]][3] #Ganancia del actuador
print(f'K={"{:.2f}".format(K)}')
if not type(metrics) == list:
    metrics = [metrics]
for i, l in enumerate(['loss']+metrics):
    print(f'{l}: {"{:.2e}".format(losses[i])} -> {"{:.2f}".format(losses[i]*K)}␣
↪RPM')
```

```
45/45 [==============================] - 9s 165ms/step - loss: 5.2549e-04 -
mean_squared_error: 5.2549e-04
K=12226.15
loss: 5.25e-04 -> 6.42 RPM
mean_squared_error: 5.25e-04 -> 6.42 RPM
Wall time: 8.85 s
```

Evaluación con 1 trayectoria

[52]:
```
window_test = 3000
test_traj_generator  = DataGenerator(X=X_test,  Y=Y_test,  batch_size=1,␣
 ↪window=window_test, sequence_out=True, feedback=feedback)
```

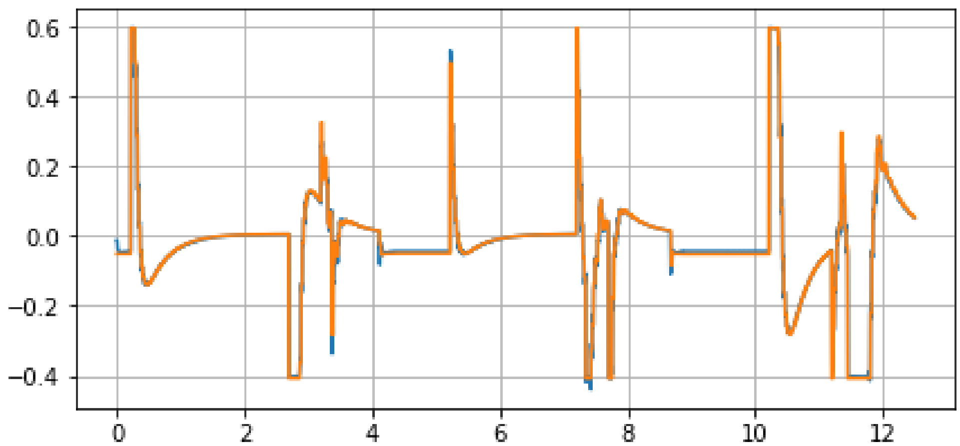
[53]:
```
%%time
X, Y = next(test_traj_generator.build_data())
x = X[0]
y = Y[0]
y_pred = []
for i in range(0, len(x)):
    if i == 0:
        x_temp = x[0].reshape(1,  1, X.shape[2])
    elif i<=window:
        x_temp = x[0:i].reshape(1,  i, X.shape[2])
    else:
        x_temp = x[i-window:i].reshape(1,  window, X.shape[2])
    y_temp = model.predict(x_temp)
    y_pred.append(y_temp)
y_pred = np.array(y_pred).reshape(y.shape)
```

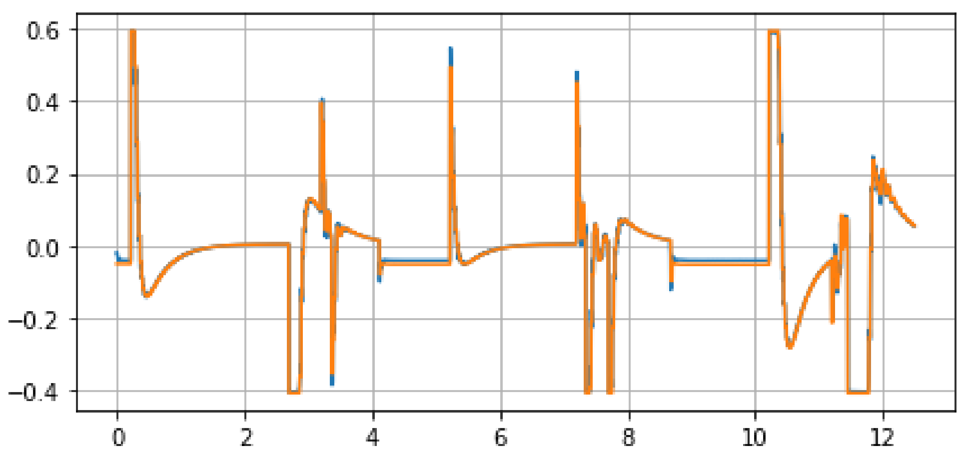
```
Wall time: 2min 11s
```

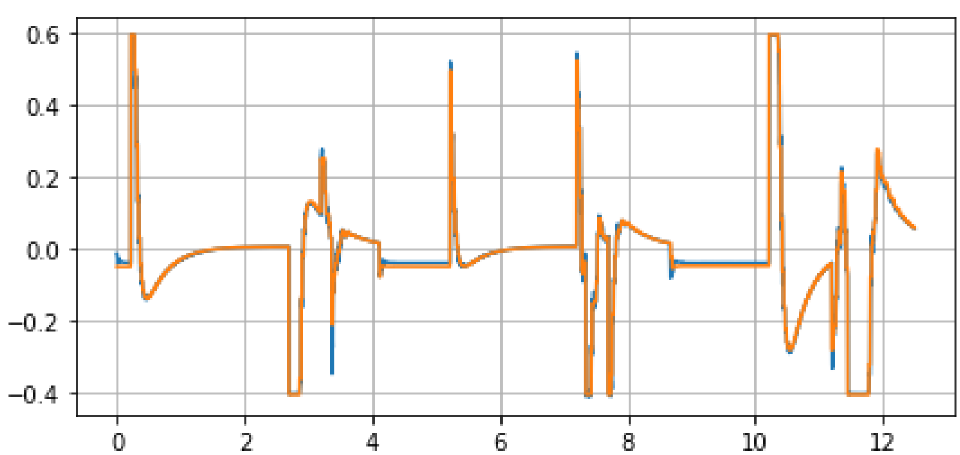
Visualización con 1 trayectoria

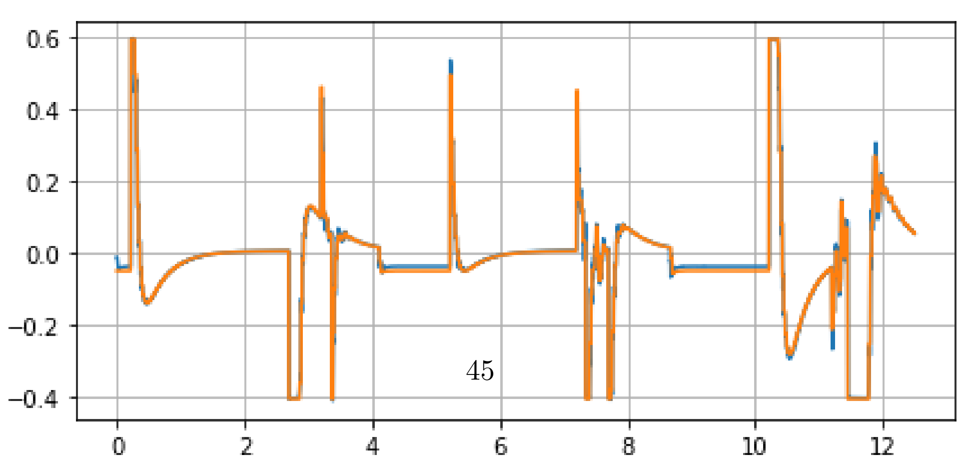
[54]:
```
t = np.arange(0, len(y)*Ts, Ts)
```

[55]:
```
fig, axs = plt.subplots(Y.shape[2], figsize = (7,15))
for i in range(Y.shape[2]):
    axs[i].plot(t, y_pred[:,i], t, y[:,i])
    axs[i].grid()
```

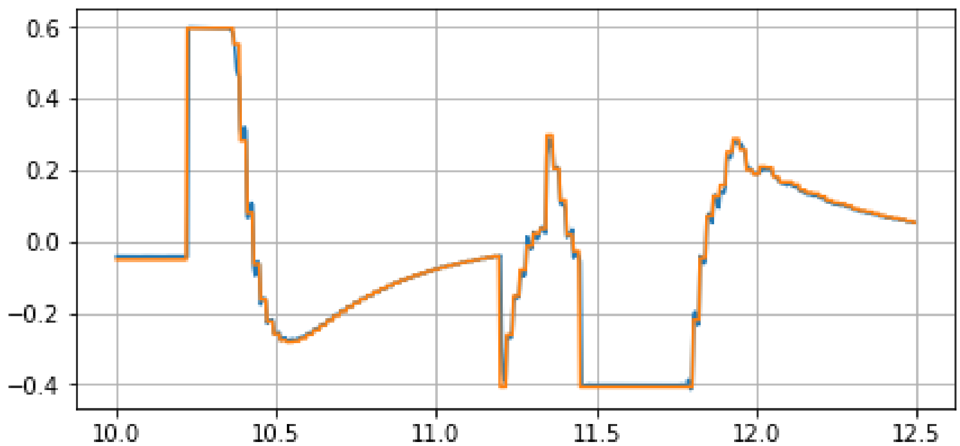

0.6
0.4
0.2
0.0
−0.2
−0.4
0
2
4
6
8
10
12

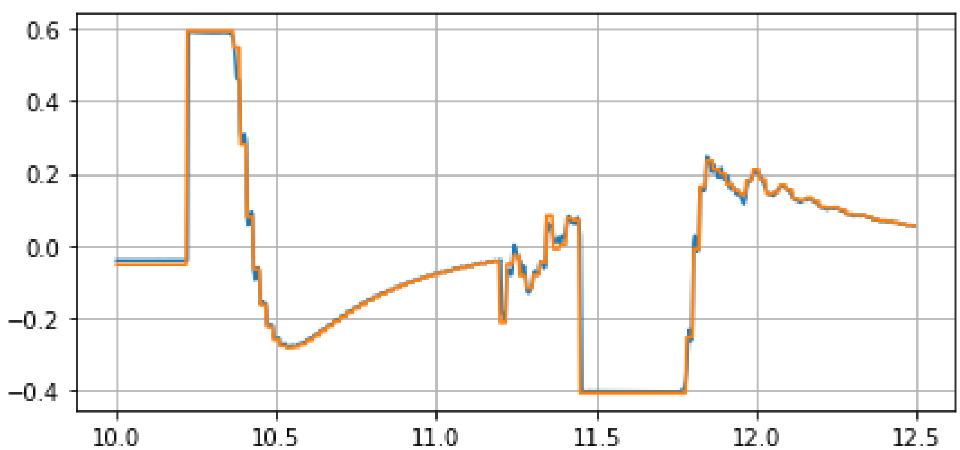

0.6
0.4
0.2
0.0
−0.2
−0.4
0
2
4
6
8
10
12

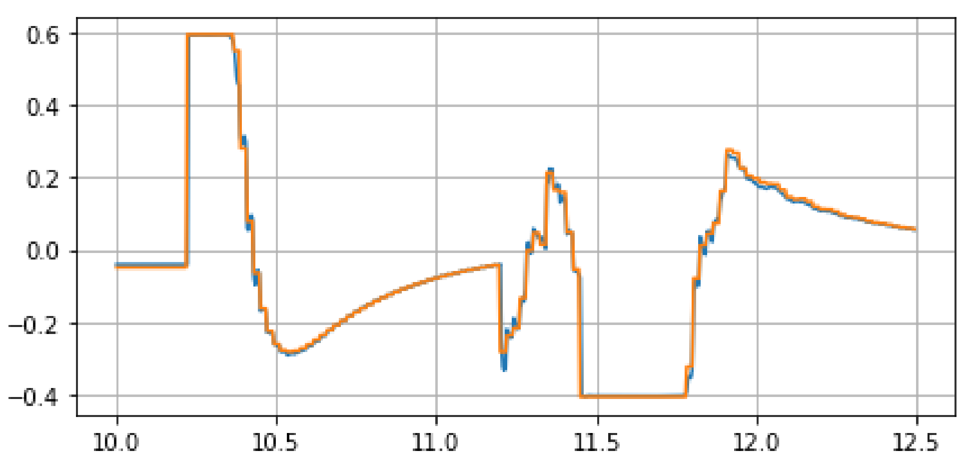

0.6
0.4
0.2
0.0
−0.2
−0.4
0
2
4
6
8
10
12

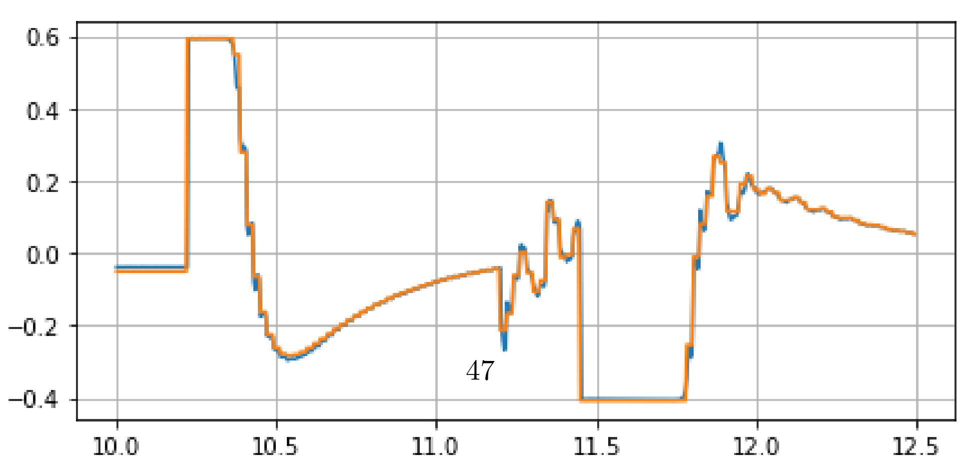

0.6
0.4
0.2
0.0
−0.2
−0.4
0
2
4
6
8
10
12

[56]:
```
fig, axs = plt.subplots(Y.shape[2], figsize = (7,15))
L1 = 4*len(t)//5
L2 = 5*len(t)//5
for i in range(Y.shape[2]):
    axs[i].plot(t[L1:L2], y_pred[L1:L2,i], t[L1:L2:], y[L1:L2:,i])
    axs[i].grid()
```


0.6
0.4
0.2
0.0
−0.2
−0.4
10.0
10.5
11.0
11.5
12.0
12.5


0.6
0.4
0.2
0.0
−0.2
−0.4
10.0
10.5
11.0
11.5
12.0
12.5


0.6
0.4
0.2
0.0
−0.2
−0.4
10.0
10.5
11.0
11.5
12.0
12.5


0.6
0.4
0.2
0.0
−0.2
−0.4
10.0
10.5
11.0
11.5
12.0
12.5

Tiempo de inferencia tamaño entrada

[57]:
```
import time

def MSE(y, y_pred):
    return (y-y_pred)**2

window_test = 1000
test_traj_batch = 1000
test_traj_generator  = DataGenerator(X=X_test,  Y=Y_test, ␣
 ↪batch_size=test_traj_batch, window=window_test, sequence_out=True)

window_len = []
inf_time = []
loss = []

x, y = next(test_traj_generator.build_data())
```

[58]:
```
%%time
accum_time = 0
n_iter = 200
test_range = list(set(np.rint(np.logspace(0, np.log10(x.shape[1]-1),␣
 ↪num=n_iter, endpoint=True))))
test_range.sort()
n_iter = len(test_range)
for k, i in enumerate(list(map(int,test_range))):
    inf_time_aux = []    # Tiempo de inferencia auxiliar
    loss_aux = []    # Costo Auxiliar
    init_time = time.time()
    for j in range(len(x)):
        x_5 = x[j][0:i].reshape(1,  i, x.shape[2])
        start_time = time.time()
        y_pred = model.predict(x_5)
        finish_time = time.time() - start_time
        inf_time_aux.append(finish_time)
        loss_aux.append(
            np.mean(
                MSE(y[j][i], y_pred)
            )
        )

    window_len.append(i)
    inf_time.append(np.mean(inf_time_aux))
    loss.append(np.mean(loss_aux))
    clear_output(wait=True)
```

```
    accum_time += time.time()-init_time
    print(f'iter = {k} de {len(test_range)}, i = {i}, execution time = {"{:.
↪2f}".format(np.max(accum_time))}s')
```

```
iter = 28 de 131, i = 29, execution time = 1201.15s
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<timed exec> in <module>

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\engine\training.
↪py in predict(self, x, batch_size, verbose, steps, callbacks, max_queue_size,␣
↪workers, use_multiprocessing)
   1694                         '. Consider setting it to AutoShardPolicy.DATA. )
   1695
-> 1696       data_handler = data_adapter.get_data_handler(
   1697           x=x,
   1698           batch_size=batch_size,

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\engine\data_adapter.
↪py in get_data_handler(*args, **kwargs)
   1362   if getattr(kwargs["model"], "_cluster_coordinator", None):
   1363     return _ClusterCoordinatorDataHandler(*args, **kwargs)
-> 1364   return DataHandler(*args, **kwargs)
   1365
   1366

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\engine\data_adapter.
↪py in __init__(self, x, y, sample_weight, batch_size, steps_per_epoch,␣
↪initial_epoch, epochs, shuffle, class_weight, max_queue_size, workers,␣
↪use_multiprocessing, model, steps_per_execution, distribute)
   1152     adapter_cls = select_data_adapter(x, y)
   1153     self._verify_data_adapter_compatibility(adapter_cls)
-> 1154     self._adapter = adapter_cls(
   1155         x,
   1156         y,

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\engine\data_adapter.
↪py in __init__(self, x, y, sample_weights, sample_weight_modes, batch_size,␣
↪epochs, steps, shuffle, **kwargs)
    335       return flat_dataset
    336
--> 337     indices_dataset = indices_dataset.flat_map(slice_batch_indices)
    338
    339     dataset = self.slice_inputs(indices_dataset, inputs)

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\data\ops\dataset_ops.
↪py in flat_map(self, map_func)
```

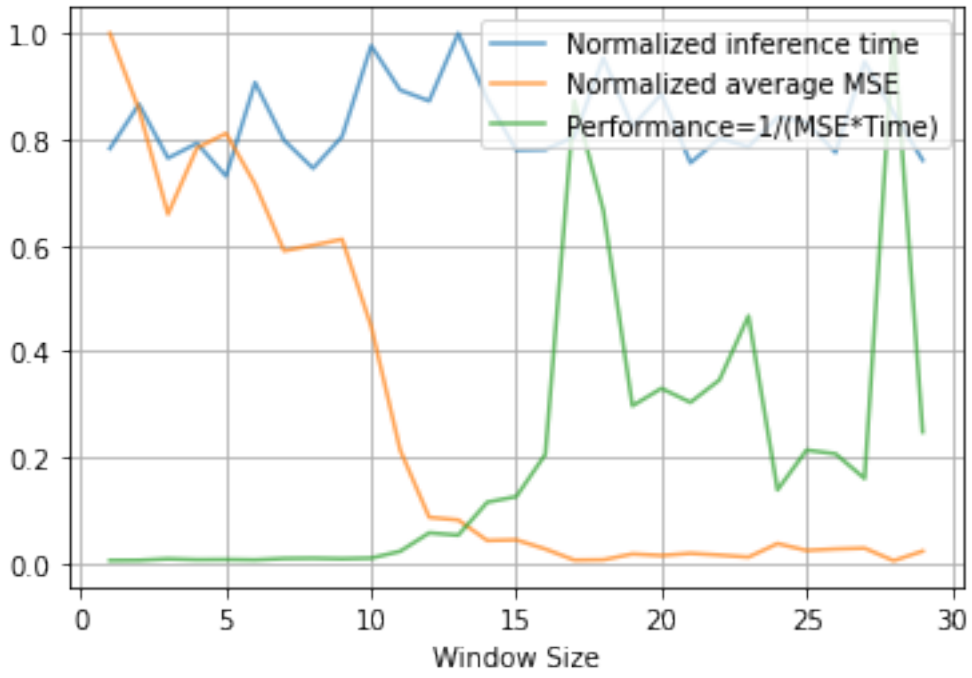
```
   1955       Dataset: A `Dataset`.
   1956     """
-> 1957     return FlatMapDataset(self, map_func)
   1958
   1959   def interleave(self,

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\data\ops\data et_ops.
 ↪py in __init__(self, input_dataset, map_func)
   4562     """See `Dataset.flat_map()` for details."""
   4563     self._input_dataset = input_dataset
-> 4564     self._map_func = StructuredFunctionWrapper(
   4565         map_func, self._transformation_name(), dataset=input_dataset)
   4566     if not isinstance(self._map_func.output_structure, DatasetSpec):

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\data\ops\data et_ops.
 ↪py in __init__(self, func, transformation_name, dataset, input_classes,␣
 ↪input_shapes, input_types, input_structure, add_to_graph, use_legacy_function ␣
 ↪defun_kwargs)
   3710     resource_tracker = tracking.ResourceTracker()
   3711     with tracking.resource_tracker_scope(resource_tracker):
-> 3712       self._function = fn_factory()
   3713       # There is no graph to add in eager mode.
   3714       add_to_graph &= not context.executing_eagerly()

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\eager\functio .
 ↪py in get_concrete_function(self, *args, **kwargs)
   3132          or `tf.Tensor` or `tf.TensorSpec`.
   3133     """
-> 3134     graph_function = self._get_concrete_function_garbage_collected(
   3135         *args, **kwargs)
   3136     graph_function._garbage_collector.release()  # pylint:␣
 ↪disable=protected-access

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\eager\functio .
 ↪py in _get_concrete_function_garbage_collected(self, *args, **kwargs)
   3098       args, kwargs = None, None
   3099     with self._lock:
-> 3100       graph_function, _ = self._maybe_define_function(args, kwargs)
   3101       seen_names = set()
   3102       captured = object_identity.ObjectIdentitySet(

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\eager\functio .
 ↪py in _maybe_define_function(self, args, kwargs)
   3442
   3443           self._function_cache.missed.add(call_context_key)
-> 3444           graph_function = self._create_graph_function(args, kwargs)
   3445           self._function_cache.primary[cache_key] = graph_function
```

```
   3446

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\eager\functio .
 ↪py in _create_graph_function(self, args, kwargs, override_flat_arg_shapes)
   3277     arg_names = base_arg_names + missing_arg_names
   3278     graph_function = ConcreteFunction(
-> 3279         func_graph_module.func_graph_from_py_func(
   3280             self._name,
   3281             self._python_function,

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\framework\fun _graph.
 ↪py in func_graph_from_py_func(name, python_func, args, kwargs, signature,␣
 ↪func_graph, autograph, autograph_options, add_control_dependencies, arg_names ␣
 ↪op_return_value, collections, capture_by_value, override_flat_arg_shapes)
   1038         if x is not None)
   1039
-> 1040     func_graph.variables = variables
   1041
   1042   if add_control_dependencies:

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\framework\aut _control_deps
 ↪py in __exit__(self, unused_type, unused_value, unused_traceback)
    447
    448       # Ensure ordering of collective ops
--> 449       manager_ids = collective_manager_ids_from_op(op)
    450       for manager_id in manager_ids:
    451         if manager_id in collective_manager_scopes_opened:

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\framework\aut _control_deps
 ↪py in collective_manager_ids_from_op(op)
    152     List of CollectiveManager IDs used by the op.
    153   """
--> 154   if op.type == "CollectiveReduce":
    155     try:
    156       return [op.get_attr("_collective_manager_id")]

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\framework\ops
 ↪py in type(self)
   2451   def type(self):
   2452     """The type of the op (e.g. `"MatMul"`)."""
-> 2453     return pywrap_tf_session.TF_OperationOpType(self._c_op)
   2454
   2455   @property

KeyboardInterrupt:
```

[59]:
```
inf_time_norm = np.array(inf_time)/max(inf_time)
loss_norm = np.array(loss)/max(loss)
performance = 1/(np.array(loss)*np.array(inf_time))
performance = np.array(performance)/max(performance)
```

[60]:
```
plt.figure()
plt.plot(window_len, inf_time_norm, label='Normalized inference time', alpha=0.
 ↪7)
plt.plot(window_len, loss_norm, label='Normalized average MSE', alpha=0.7)
plt.plot(window_len, performance, label='Performance=1/(MSE*Time)', alpha=0.7)
plt.xlabel("Window Size")
plt.legend(loc=1)
plt.grid()

print(f'Best window inference time={np.argmin(inf_time_norm)} steps, time={"{:.
 ↪2f}".format(np.min(inf_time)*1000)} ms')
print(f'Best window MSE loss={np.argmin(loss_norm)} steps, MSE={"{:.2e}".
 ↪format(np.min(loss))}')
print(f'Best window Performance (MSExTime)={np.argmax(performance)} steps,␣
 ↪Value={"{:.2e}".format(np.max(loss)/np.min(inf_time))}')
```

```
Best window inference time=4 steps, time=36.19 ms
Best window MSE loss=27 steps, MSE=2.10e-04
Best window Performance (MSExTime)=27 steps, Value=1.13e+00
```



[61]:
```
plt.figure()
plt.semilogx(window_len, inf_time_norm, label='Normalized inference time',␣
 ↪alpha=0.7)
plt.semilogx(window_len, loss_norm, label='Normalized average MSE', alpha=0.7)
plt.semilogx(window_len, performance, label='Performance=1/(MSE*Time)', alpha=0.
 ↪7)
plt.xlabel("Window Size")
plt.legend(loc=1)
plt.grid()
```

```
Error in callback <function flush_figures at 0x000002B27B259D30> (for
post_execute):
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
~\.conda\envs\tesis-pybullet\lib\site-packages\ipykernel\pylab\backend_inline.p␣
 ↪in flush_figures()
    119         # ignore the tracking, just draw and close all figures
    120         try:
--> 121             return show(True)
    122         except Exception as e:
    123             # safely show traceback if in IPython, else raise

~\.conda\envs\tesis-pybullet\lib\site-packages\ipykernel\pylab\backend_inline.p␣
 ↪in show(close, block)
     39     try:
     40         for figure_manager in Gcf.get_all_fig_managers():
---> 41             display(

     42                 figure_manager.canvas.figure,
     43                 metadata=_fetch_figure_metadata(figure_manager.canvas.
 ↪figure)

~\.conda\envs\tesis-pybullet\lib\site-packages\IPython\core\display.py in␣
 ↪display(include, exclude, metadata, transient, display_id, *objs, **kwargs)
    311             publish_display_data(data=obj, metadata=metadata, **kwargs)
    312         else:
--> 313             format_dict, md_dict = format(obj, include=include,␣
 ↪exclude=exclude)
    314             if not format_dict:
    315                 # nothing to display (e.g. _ipython_display_ took over)

~\.conda\envs\tesis-pybullet\lib\site-packages\IPython\core\formatters.py in␣
 ↪format(self, obj, include, exclude)
    178             md = None
    179             try:
--> 180                 data = formatter(obj)
    181             except:
```

```
    182                 # FIXME: log the exception

~\.conda\envs\tesis-pybullet\lib\site-packages\decorator.py in fun(*args, **kw)
    230             if not kwsyntax:
    231                 args, kw = fix(args, kw, sig)
--> 232             return caller(func, *(extras + args), **kw)
    233     fun.__name__ = func.__name__
    234     fun.__doc__ = func.__doc__

~\.conda\envs\tesis-pybullet\lib\site-packages\IPython\core\formatters.py in␣
↪catch_format_error(method, self, *args, **kwargs)
    222     """show traceback on failed format call"""
    223     try:
--> 224         r = method(self, *args, **kwargs)
    225     except NotImplementedError:
    226         # don't warn on NotImplementedErrors

~\.conda\envs\tesis-pybullet\lib\site-packages\IPython\core\formatters.py in␣
↪__call__(self, obj)
    339                 pass
    340             else:
--> 341                 return printer(obj)
    342             # Finally look for special method names
    343             method = get_real_method(obj, self.print_method)

~\.conda\envs\tesis-pybullet\lib\site-packages\IPython\core\pylabtools.py in␣
↪<lambda>(fig)
    246
    247     if 'png' in formats:
--> 248         png_formatter.for_type(Figure, lambda fig: print_figure(fig,␣
↪'png', **kwargs))
    249     if 'retina' in formats or 'png2x' in formats:
    250         png_formatter.for_type(Figure, lambda fig: retina_figure(fig,␣
↪**kwargs))

~\.conda\envs\tesis-pybullet\lib\site-packages\IPython\core\pylabtools.py in␣
↪print_figure(fig, fmt, bbox_inches, **kwargs)
    130         FigureCanvasBase(fig)
    131
--> 132     fig.canvas.print_figure(bytes_io, **kw)
    133     data = bytes_io.getvalue()
    134     if fmt == 'svg':

~\.conda\envs\tesis-pybullet\lib\site-packages\matplotlib\backend_bases.py in␣
↪print_figure(self, filename, dpi, facecolor, edgecolor, orientation, format,␣
↪bbox_inches, pad_inches, bbox_extra_artists, backend, **kwargs)
   2228                         else suppress())
   2229                 with ctx:
```

```
-> 2230                     self.figure.draw(renderer)
   2231
   2232             if bbox_inches:

~\.conda\envs\tesis-pybullet\lib\site-packages\matplotlib\artist.py in␣
↪draw_wrapper(artist, renderer, *args, **kwargs)
     72     @wraps(draw)
     73     def draw_wrapper(artist, renderer, *args, **kwargs):
---> 74         result = draw(artist, renderer, *args, **kwargs)
     75         if renderer._rasterizing:
     76             renderer.stop_rasterizing()

~\.conda\envs\tesis-pybullet\lib\site-packages\matplotlib\artist.py in␣
↪draw_wrapper(artist, renderer, *args, **kwargs)
     49                 renderer.start_filter()
     50
---> 51             return draw(artist, renderer, *args, **kwargs)
     52         finally:
     53             if artist.get_agg_filter() is not None:

~\.conda\envs\tesis-pybullet\lib\site-packages\matplotlib\figure.py in␣
↪draw(self, renderer)
   2778
   2779             self.patch.draw(renderer)
-> 2780             mimage._draw_list_compositing_images(
   2781                 renderer, self, artists, self.suppressComposite)
   2782

~\.conda\envs\tesis-pybullet\lib\site-packages\matplotlib\image.py in␣
↪_draw_list_compositing_images(renderer, parent, artists, suppress_composite)
    130     if not_composite or not has_images:
    131         for a in artists:
--> 132             a.draw(renderer)
    133     else:
    134         # Composite any adjacent images together

~\.conda\envs\tesis-pybullet\lib\site-packages\matplotlib\artist.py in␣
↪draw_wrapper(artist, renderer, *args, **kwargs)
     49                 renderer.start_filter()
     50
---> 51             return draw(artist, renderer, *args, **kwargs)
     52         finally:
     53             if artist.get_agg_filter() is not None:

~\.conda\envs\tesis-pybullet\lib\site-packages\matplotlib\_api\deprecation.py i␣
↪wrapper(*inner_args, **inner_kwargs)
    429                             else deprecation_addendum,
```

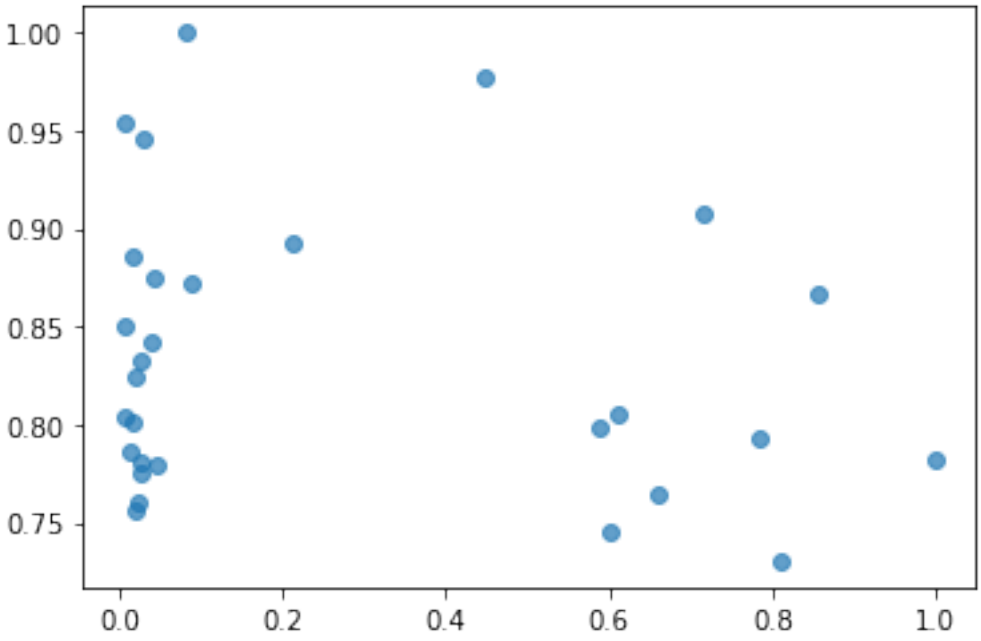
```
    430                 **kwargs)
--> 431         return func(*inner_args, **inner_kwargs)
    432
    433     return wrapper

~\.conda\envs\tesis-pybullet\lib\site-packages\matplotlib\axes\_base.py in␣
↪draw(self, renderer, inframe)
   2919             renderer.stop_rasterizing()
   2920
-> 2921         mimage._draw_list_compositing_images(renderer, self, artists)
   2922
   2923         renderer.close_group('axes')

~\.conda\envs\tesis-pybullet\lib\site-packages\matplotlib\image.py in␣
↪_draw_list_compositing_images(renderer, parent, artists, suppress_composite)
    130     if not_composite or not has_images:
    131         for a in artists:
--> 132             a.draw(renderer)
    133     else:
    134         # Composite any adjacent images together

~\.conda\envs\tesis-pybullet\lib\site-packages\matplotlib\artist.py in␣
↪draw_wrapper(artist, renderer, *args, **kwargs)
     49                 renderer.start_filter()
     50
---> 51             return draw(artist, renderer, *args, **kwargs)
     52         finally:
     53             if artist.get_agg_filter() is not None:

~\.conda\envs\tesis-pybullet\lib\site-packages\matplotlib\axis.py in draw(self,␣
↪renderer, *args, **kwargs)
   1135
   1136         ticks_to_draw = self._update_ticks()
-> 1137         ticklabelBoxes, ticklabelBoxes2 = self.
↪_get_tick_bboxes(ticks_to_draw,
   1138                                                                  renderer)
   1139

~\.conda\envs\tesis-pybullet\lib\site-packages\matplotlib\axis.py in␣
↪_get_tick_bboxes(self, ticks, renderer)
   1061     def _get_tick_bboxes(self, ticks, renderer):
   1062         """Return lists of bboxes for ticks' label1's and label2's."""
-> 1063         return ([tick.label1.get_window_extent(renderer)
   1064                  for tick in ticks if tick.label1.get_visible()],
   1065                 [tick.label2.get_window_extent(renderer)
```

```
~\.conda\envs\tesis-pybullet\lib\site-packages\matplotlib\axis.py in <listcomp> .
↪0)
   1061     def _get_tick_bboxes(self, ticks, renderer):
   1062         """Return lists of bboxes for ticks' label1's and label2's."""
-> 1063         return ([tick.label1.get_window_extent(renderer)

   1064                  for tick in ticks if tick.label1.get_visible()],
   1065                 [tick.label2.get_window_extent(renderer)

~\.conda\envs\tesis-pybullet\lib\site-packages\matplotlib\text.py in␣
↪get_window_extent(self, renderer, dpi)
    901
    902         with cbook._setattr_cm(self.figure, dpi=dpi):
--> 903             bbox, info, descent = self._get_layout(self._renderer)
    904             x, y = self.get_unitless_position()
    905             x, y = self.get_transform().transform((x, y))

~\.conda\envs\tesis-pybullet\lib\site-packages\matplotlib\text.py in␣
↪_get_layout(self, renderer)
    312             clean_line, ismath = self._preprocess_math(line)
    313             if clean_line:
--> 314                 w, h, d = renderer.get_text_width_height_descent(

    315                     clean_line, self._fontproperties, ismath=ismath)
    316             else:

~\.conda\envs\tesis-pybullet\lib\site-packages\matplotlib\backends\backend_agg.
↪py in get_text_width_height_descent(self, s, prop, ismath)
    233         if ismath:
    234             ox, oy, width, height, descent, fonts, used_characters = \
--> 235                 self.mathtext_parser.parse(s, self.dpi, prop)
    236             return width, height, descent
    237

~\.conda\envs\tesis-pybullet\lib\site-packages\matplotlib\mathtext.py in␣
↪parse(self, s, dpi, prop, _force_standard_ps_fonts)
    450         # mathtext.fontset rcParams also affect the parse (e.g. by␣
↪affecting
    451         # the glyph metrics).
--> 452         return self._parse_cached(s, dpi, prop, _force_standard_ps_font )
    453
    454     @functools.lru_cache(50)

~\.conda\envs\tesis-pybullet\lib\site-packages\matplotlib\mathtext.py in␣
↪_parse_cached(self, s, dpi, prop, force_standard_ps_fonts)
    462                 self._font_type_mapping, fontset=prop.
↪get_math_fontfamily()))
    463         backend = self._backend_mapping[self._output]()
--> 464         font_output = fontset_class(prop, backend)
```

```
    465
    466         fontsize = prop.get_size_in_points()

~\.conda\envs\tesis-pybullet\lib\site-packages\matplotlib\_mathtext.py in␣
↪__init__(self, *args, **kwargs)
    585         })
    586         for key, name in self._fontmap.items():
--> 587             fullpath = findfont(name)
    588             self.fontmap[key] = fullpath
    589             self.fontmap[name] = fullpath

~\.conda\envs\tesis-pybullet\lib\site-packages\matplotlib\font_manager.py in␣
↪findfont(self, prop, fontext, directory, fallback_to_default,␣
↪rebuild_if_missing)
   1305             "font.serif", "font.sans-serif", "font.cursive", "font.
↪fantasy",
   1306             "font.monospace"])
-> 1307         return self._findfont_cached(
   1308             prop, fontext, directory, fallback_to_default,␣
↪rebuild_if_missing,
   1309             rc_params)

~\.conda\envs\tesis-pybullet\lib\site-packages\matplotlib\font_manager.py in␣
↪_findfont_cached(self, prop, fontext, directory, fallback_to_default,␣
↪rebuild_if_missing, rc_params)
   1369             result = best_font.fname
   1370
-> 1371         if not os.path.isfile(result):
   1372             if rebuild_if_missing:
   1373                 _log.info(

~\.conda\envs\tesis-pybullet\lib\genericpath.py in isfile(path)
     28     """Test whether a path is a regular file"""
     29     try:
---> 30         st = os.stat(path)
     31     except (OSError, ValueError):
     32         return False

KeyboardInterrupt:
```

[62]:
```
plt.figure()
plt.scatter(loss_norm, inf_time_norm, alpha=0.7)
```

[62]:
```
<matplotlib.collections.PathCollection at 0x2b2edad15e0>
```


1.00
0.95
0.90
0.85
0.80
0.75
0.0
0.2
0.4
0.6
0.8
1.0

[ ]: