Lab 9: RNN for Hour-Ahead Short-Term Load Forecasting (STLF)

Objectives

- Develop a **Recurrent Neural Network (RNN)** model for predicting electrical load one hour ahead.
- Understand how **temporal dependencies** in time-series data can be modeled using RNNs.
- Prepare sequential data using sliding windows to feed into the RNN architecture.
- Train the RNN model using historical load data and evaluate its forecasting accuracy.
- Use performance metrics like **MAE**, **RMSE**, and **MAPE** for model evaluation.
- Apply techniques like **early stopping** and **model checkpointing** to enhance model robustness.
- Compare the RNN model's performance with other architectures (e.g., 1D-CNN or MLP) in STLF tasks.

Set Working Directory

```
import os
os.chdir(r'C:\Users\PMLS\ML\lab9')
```

Required Imports

This project uses standard ML libraries along with custom utility modules for time-series forecasting.

```python

# Core Libraries

import numpy as np, pandas as pd, matplotlib.pyplot as plt, time, pickle, glob, h5py

# Model Development

from tensorflow.keras.models import Sequential, Model, load_model from tensorflow.keras.layers import Input, Dense, LSTM, SimpleRNN, Conv1D, Flatten, Dropout, BatchNormalization from tensorflow.keras.layers import Activation, MaxPooling1D, GlobalMaxPooling1D, TimeDistributed, Bidirectional from tensorflow.keras.optimizers import Adam, SGD from tensorflow.keras.callbacks import ModelCheckpoint, Callback import tensorflow.keras.backend as K
```

Custom Utilities & Callbacks

from timeseires.utils import * from timeseires.callbacks import EpochCheckpoint, TrainingMonitor

```
from sklearn.metrics import mean_squared_error, mean_absolute_error,
explained_variance_score, r2_score
from timeseires.utils.to_split import to_split
from timeseires.utils.multivariate_multi_step import
multivariate_multi_step
from timeseires.utils.multivariate_single_step import
multivariate_single_step
from timeseires.utils.univariate_multi_step import
univariate_multi_step
from timeseires.utils.univariate_single_step import
univariate_single_step
from timeseires.utils.CosineAnnealingLRS import CosineAnnealingLRS
from timeseires.callbacks.EpochCheckpoint import EpochCheckpoint
from tensorflow.keras.callbacks import ModelCheckpoint
from timeseires.callbacks.TrainingMonitor import TrainingMonitor
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.models import load_model
from tensorflow.keras.layers import LSTM, Bidirectional, Add
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import Conv1D,TimeDistributed
from tensorflow.keras.layers import Dense, Dropout, Activation,
Flatten,MaxPooling1D,Concatenate,AveragePooling1D, GlobalMaxPooling1D,
Input, SimpleRNN
from tensorflow.keras.models import Sequential,Model
import pandas as pd
import time, pickle
import numpy as np
import tensorflow.keras.backend as K
import tensorflow
from tensorflow.keras.layers import Input, Reshape, Lambda
from tensorflow.keras.layers import Layer, Flatten, LeakyReLU,
concatenate, Dense
from tensorflow.keras.regularizers import l2
import glob
import h5py
import matplotlib.pyplot as plt
from keras.callbacks import Callback
```

Model Configuration

```
#lookback = 24
model = None
```

```
start_epoch = 0
time_steps=24
num_features=21
```

RNN Model Architecture

```
def create_rnn():
    input_data = Input(shape=(time_steps, num_features))
    rnn_layer1 = SimpleRNN(8, return_sequences=True)(input_data)
    rnn_layer2 = SimpleRNN(20)(rnn_layer1)
    x = Flatten()(rnn_layer2)
    output_data = Dense(1)(x)
    model = Model(input_data, output_data)
    return model

model1 = create_rnn()
model1.summary()
```

```
Model: "functional"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ input_layer (InputLayer)             │ (None, 24, 21)              │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ simple_rnn (SimpleRNN)               │ (None, 24, 8)               │             240 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ simple_rnn_1 (SimpleRNN)             │ (None, 20)                  │             580 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ flatten (Flatten)                    │ (None, 20)                  │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense (Dense)                        │ (None, 1)                   │              21 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
```

```
 Total params: 841 (3.29 KB)

 Trainable params: 841 (3.29 KB)
```

```
 Non-trainable params: 0 (0.00 B)
```

```
tensorflow.keras.utils.plot_model(model1 )
```

```
You must install graphviz (see instructions at
https://graphviz.gitlab.io/download/) for `plot_model` to work.
```

File Paths for Checkpoints and Training History

Defines paths to save:

- Model checkpoints after each epoch
- Training history plot (`history.png`)
- Training history in JSON format (`history.json`)

```
checkpoints = r'C:\Users\PMLS\ML\lab9\\E1-cp-{epoch:04d}-
loss{val_loss:.2f}.h5'
OUTPUT_PATH = r'C:\Users\PMLS\ML\lab9'
FIG_PATH = os.path.sep.join([OUTPUT_PATH,"\history.png"])
JSON_PATH = os.path.sep.join([OUTPUT_PATH,"\history.json"])
```

Callbacks for Checkpointing and Training Monitoring

```
# construct the callback to save only the *best* model to disk
# based on the validation loss
EpochCheckpoint1 = ModelCheckpoint(checkpoints,
                                   monitor="val_loss",
                                   save_best_only=True,
                                   verbose=1)
```

Model Compilation or Checkpoint Loading

```
# if there is no specific model checkpoint supplied, then initialize
# the network and compile the model
if model is None:
    print("[INFO] compiling model...")
    model =create_rnn()
    opt = Adam(1e-3)
    model.compile(loss= 'mae', optimizer=opt, metrics=["mae", "mape"])
# otherwise, load the checkpoint from disk
else:
    print("[INFO] loading {}...".format(model))
    model = load_model(model)

    # update the learning rate
    print("[INFO] old learning rate:
{}".format(K.get_value(model.optimizer.lr)))
    K.set_value(model.optimizer.lr, 1e-4)
    print("[INFO] new learning rate:
{}".format(K.get_value(model.optimizer.lr)))
```

```
[INFO] compiling model...
```

Loading Dataset and Scaler for Training, Validation, and Testing

```
import os
path_dataset =r'C:\Users\PMLS\ML\lab9'
path_tr = os.path.join(path_dataset, 'AEP_train.csv')
df_tr = pd.read_csv(path_tr)
train_set = df_tr.iloc[:].values
path_v = os.path.join(path_dataset, 'AEP_validation.csv')
df_v = pd.read_csv(path_v)
validation_set = df_v.iloc[:].values
path_te = os.path.join(path_dataset, 'AEP_test.csv')
df_te = pd.read_csv(path_te)
test_set = df_te.iloc[:].values

path_scaler = os.path.join(path_dataset, 'AEP_Scaler.pkl')
scaler         = pickle.load(open(path_scaler, 'rb'))

train_set.shape, validation_set.shape, test_set.shape
```

```
C:\Users\PMLS\anaconda3\envs\myenv\lib\site-packages\sklearn\
base.py:380: InconsistentVersionWarning: Trying to unpickle estimator
MinMaxScaler from version 1.0.2 when using version 1.6.1. This might
lead to breaking code or invalid results. Use at your own risk. For
more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-
maintainability-limitations
  warnings.warn(

((84907, 21), (24259, 21), (12130, 21))
```

Time-Series Input Configuration

```
time_steps=24
num_features=21
```

Preparing Input Data Using Univariate Multi-Step Function

```
start = time.time()
train_X , train_y = univariate_multi_step(train_set, time_steps,
target_col=0,target_len=1)
validation_X, validation_y = univariate_multi_step(validation_set,
time_steps, target_col=0,target_len=1)
test_X, test_y = univariate_multi_step(test_set, time_steps,
target_col=0,target_len=1)
print('Time Consumed', time.time()-start, "sec")
```

```
Time Consumed 0.3962678909301758 sec
```

Model Training with Validation and Callbacks

```
from keras.callbacks import ModelCheckpoint, EarlyStopping

# Define custom checkpoint path format
checkpoint_path = checkpoints

# Define callbacks
callbacks = [
    ModelCheckpoint(filepath=checkpoint_path,
                    monitor='val_loss',
                    save_best_only=True,
                    verbose=1),

    EarlyStopping(monitor='val_loss',
                  patience=10,
                  restore_best_weights=True)
]

# Training configuration
epochs = 60
verbose = 1
batch_size = 32

# Train model
History = model.fit(train_X,
                    train_y,
                    batch_size=batch_size,
                    epochs=epochs,
                    validation_data=(validation_X, validation_y),
                    callbacks=callbacks,
                    verbose=verbose)
```

```
Epoch 1/60
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0120 - mae:
0.0120 - mape: 233.2040
Epoch 1: val_loss improved from inf to 0.01041, saving model to C:\
Users\PMLS\ML\lab9\\E1-cp-0001-loss0.01.h5

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

2653/2653 ━━━━━━━━━━━━━━━━━━━━ 26s 10ms/step - loss: 0.0120 - mae:
0.0120 - mape: 233.2277 - val_loss: 0.0104 - val_mae: 0.0104 -
val_mape: 4.5255
Epoch 2/60
2652/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0114 - mae:
0.0114 - mape: 695.3312
```

```
Epoch 2: val_loss did not improve from 0.01041
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 25s 10ms/step - loss: 0.0114 - mae:
0.0114 - mape: 695.0827 - val_loss: 0.0119 - val_mae: 0.0119 -
val_mape: 5.1713
Epoch 3/60
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0111 - mae:
0.0111 - mape: 141.6864
Epoch 3: val_loss did not improve from 0.01041
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 28s 11ms/step - loss: 0.0111 - mae:
0.0111 - mape: 141.7432 - val_loss: 0.0104 - val_mae: 0.0104 -
val_mape: 4.3675
Epoch 4/60
2651/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0105 - mae:
0.0105 - mape: 71.0487
Epoch 4: val_loss improved from 0.01041 to 0.00941, saving model to
C:\Users\PMLS\ML\lab9\\E1-cp-0004-loss0.01.h5

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

2653/2653 ━━━━━━━━━━━━━━━━━━━━ 24s 9ms/step - loss: 0.0105 - mae:
0.0105 - mape: 71.4098 - val_loss: 0.0094 - val_mae: 0.0094 -
val_mape: 4.1533
Epoch 5/60
2650/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0102 - mae:
0.0102 - mape: 245.6757
Epoch 5: val_loss did not improve from 0.00941
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 25s 9ms/step - loss: 0.0102 - mae:
0.0102 - mape: 245.8632 - val_loss: 0.0103 - val_mae: 0.0103 -
val_mape: 6.0454
Epoch 6/60
2648/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0097 - mae:
0.0097 - mape: 1036.5277
Epoch 6: val_loss improved from 0.00941 to 0.00849, saving model to
C:\Users\PMLS\ML\lab9\\E1-cp-0006-loss0.01.h5

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

2653/2653 ━━━━━━━━━━━━━━━━━━━━ 25s 10ms/step - loss: 0.0097 - mae:
0.0097 - mape: 1034.8300 - val_loss: 0.0085 - val_mae: 0.0085 -
val_mape: 4.2684
Epoch 7/60
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0093 - mae:
```

```
0.0093 - mape: 214.2421
Epoch 7: val_loss did not improve from 0.00849
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 19s 7ms/step - loss: 0.0093 - mae:
0.0093 - mape: 214.2564 - val_loss: 0.0087 - val_mae: 0.0087 -
val_mape: 4.4514
Epoch 8/60
2648/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0090 - mae:
0.0090 - mape: 199.7600
Epoch 8: val_loss did not improve from 0.00849
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 22s 8ms/step - loss: 0.0090 - mae:
0.0090 - mape: 199.6183 - val_loss: 0.0087 - val_mae: 0.0087 -
val_mape: 3.6556
Epoch 9/60
2649/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0089 - mae:
0.0089 - mape: 75.6102
Epoch 9: val_loss did not improve from 0.00849
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 22s 8ms/step - loss: 0.0089 - mae:
0.0089 - mape: 75.8367 - val_loss: 0.0102 - val_mae: 0.0102 -
val_mape: 5.7890
Epoch 10/60
2650/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0088 - mae:
0.0088 - mape: 91.5266
Epoch 10: val_loss did not improve from 0.00849
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 21s 8ms/step - loss: 0.0088 - mae:
0.0088 - mape: 92.0768 - val_loss: 0.0087 - val_mae: 0.0087 -
val_mape: 3.9308
Epoch 11/60
2648/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0087 - mae:
0.0087 - mape: 70.4730
Epoch 11: val_loss did not improve from 0.00849
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 20s 8ms/step - loss: 0.0087 - mae:
0.0087 - mape: 71.1669 - val_loss: 0.0092 - val_mae: 0.0092 -
val_mape: 4.9073
Epoch 12/60
2652/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0086 - mae:
0.0086 - mape: 419.4554
Epoch 12: val_loss improved from 0.00849 to 0.00839, saving model to
C:\Users\PMLS\ML\lab9\\E1-cp-0012-loss0.01.h5

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

2653/2653 ━━━━━━━━━━━━━━━━━━━━ 24s 9ms/step - loss: 0.0086 - mae:
0.0086 - mape: 419.3654 - val_loss: 0.0084 - val_mae: 0.0084 -
val_mape: 4.5436
Epoch 13/60
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0087 - mae:
```

```
0.0087 - mape: 566.1215
Epoch 13: val_loss improved from 0.00839 to 0.00787, saving model to
C:\Users\PMLS\ML\lab9\\E1-cp-0013-loss0.01.h5

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

2653/2653 ━━━━━━━━━━━━━━━━━━━━ 23s 9ms/step - loss: 0.0087 - mae:
0.0087 - mape: 565.9844 - val_loss: 0.0079 - val_mae: 0.0079 -
val_mape: 3.9084
Epoch 14/60
2652/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0084 - mae:
0.0084 - mape: 32.8394
Epoch 14: val_loss did not improve from 0.00787
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 23s 9ms/step - loss: 0.0084 - mae:
0.0084 - mape: 33.0160 - val_loss: 0.0089 - val_mae: 0.0089 -
val_mape: 4.0954
Epoch 15/60
2647/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0084 - mae:
0.0084 - mape: 93.7595
Epoch 15: val_loss improved from 0.00787 to 0.00768, saving model to
C:\Users\PMLS\ML\lab9\\E1-cp-0015-loss0.01.h5

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

2653/2653 ━━━━━━━━━━━━━━━━━━━━ 25s 10ms/step - loss: 0.0084 - mae:
0.0084 - mape: 94.6720 - val_loss: 0.0077 - val_mae: 0.0077 -
val_mape: 3.4059
Epoch 16/60
2648/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0084 - mae:
0.0084 - mape: 151.7834
Epoch 16: val_loss improved from 0.00768 to 0.00745, saving model to
C:\Users\PMLS\ML\lab9\\E1-cp-0016-loss0.01.h5

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

2653/2653 ━━━━━━━━━━━━━━━━━━━━ 26s 10ms/step - loss: 0.0084 - mae:
0.0084 - mape: 152.0507 - val_loss: 0.0074 - val_mae: 0.0074 -
val_mape: 3.2314
Epoch 17/60
```

```
2650/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0083 - mae:
0.0083 - mape: 12.4637
Epoch 17: val_loss improved from 0.00745 to 0.00713, saving model to
C:\Users\PMLS\ML\lab9\\E1-cp-0017-loss0.01.h5

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

2653/2653 ━━━━━━━━━━━━━━━━━━━━ 24s 9ms/step - loss: 0.0083 - mae:
0.0083 - mape: 13.0488 - val_loss: 0.0071 - val_mae: 0.0071 -
val_mape: 3.1315
Epoch 18/60
2647/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0083 - mae:
0.0083 - mape: 139.9209
Epoch 18: val_loss did not improve from 0.00713
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 26s 10ms/step - loss: 0.0083 - mae:
0.0083 - mape: 140.2091 - val_loss: 0.0087 - val_mae: 0.0087 -
val_mape: 4.4855
Epoch 19/60
2651/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0082 - mae:
0.0082 - mape: 297.8125
Epoch 19: val_loss did not improve from 0.00713
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 26s 10ms/step - loss: 0.0082 - mae:
0.0082 - mape: 297.7224 - val_loss: 0.0087 - val_mae: 0.0087 -
val_mape: 4.1722
Epoch 20/60
2650/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0082 - mae:
0.0082 - mape: 28.9557
Epoch 20: val_loss did not improve from 0.00713
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 25s 10ms/step - loss: 0.0082 - mae:
0.0082 - mape: 29.0996 - val_loss: 0.0088 - val_mae: 0.0088 -
val_mape: 4.3417
Epoch 21/60
2652/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0081 - mae:
0.0081 - mape: 1098.5009
Epoch 21: val_loss did not improve from 0.00713
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 26s 10ms/step - loss: 0.0081 - mae:
0.0081 - mape: 1097.8292 - val_loss: 0.0076 - val_mae: 0.0076 -
val_mape: 3.4748
Epoch 22/60
2650/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0080 - mae:
0.0080 - mape: 172.7186
Epoch 22: val_loss did not improve from 0.00713
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 20s 8ms/step - loss: 0.0080 - mae:
0.0080 - mape: 172.7645 - val_loss: 0.0077 - val_mae: 0.0077 -
val_mape: 3.4937
Epoch 23/60
```

```
2648/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0080 - mae:
0.0080 - mape: 444.1331
Epoch 23: val_loss did not improve from 0.00713
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 20s 7ms/step - loss: 0.0080 - mae:
0.0080 - mape: 443.6746 - val_loss: 0.0078 - val_mae: 0.0078 -
val_mape: 3.3120
Epoch 24/60
2649/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0080 - mae:
0.0080 - mape: 197.8779
Epoch 24: val_loss did not improve from 0.00713
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 25s 9ms/step - loss: 0.0080 - mae:
0.0080 - mape: 197.9073 - val_loss: 0.0085 - val_mae: 0.0085 -
val_mape: 3.7542
Epoch 25/60
2647/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0079 - mae:
0.0079 - mape: 228.3519
Epoch 25: val_loss improved from 0.00713 to 0.00706, saving model to
C:\Users\PMLS\ML\lab9\\E1-cp-0025-loss0.01.h5

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

2653/2653 ━━━━━━━━━━━━━━━━━━━━ 25s 9ms/step - loss: 0.0079 - mae:
0.0079 - mape: 228.3618 - val_loss: 0.0071 - val_mae: 0.0071 -
val_mape: 2.9870
Epoch 26/60
2651/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0080 - mae:
0.0080 - mape: 261.0375
Epoch 26: val_loss improved from 0.00706 to 0.00693, saving model to
C:\Users\PMLS\ML\lab9\\E1-cp-0026-loss0.01.h5

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

2653/2653 ━━━━━━━━━━━━━━━━━━━━ 27s 10ms/step - loss: 0.0080 - mae:
0.0080 - mape: 260.9283 - val_loss: 0.0069 - val_mae: 0.0069 -
val_mape: 3.2161
Epoch 27/60
2650/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0079 - mae:
0.0079 - mape: 16.3982
Epoch 27: val_loss did not improve from 0.00693
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 26s 10ms/step - loss: 0.0079 - mae:
0.0079 - mape: 16.4535 - val_loss: 0.0073 - val_mae: 0.0073 -
val_mape: 3.4502
```

```
Epoch 28/60
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0078 - mae:
0.0078 - mape: 48.2686
Epoch 28: val_loss did not improve from 0.00693
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 24s 9ms/step - loss: 0.0078 - mae:
0.0078 - mape: 48.3156 - val_loss: 0.0088 - val_mae: 0.0088 -
val_mape: 3.7741
Epoch 29/60
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0078 - mae:
0.0078 - mape: 319.6461
Epoch 29: val_loss did not improve from 0.00693
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 26s 10ms/step - loss: 0.0078 - mae:
0.0078 - mape: 319.6091 - val_loss: 0.0073 - val_mae: 0.0073 -
val_mape: 3.2263
Epoch 30/60
2651/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0077 - mae:
0.0077 - mape: 328.0288
Epoch 30: val_loss did not improve from 0.00693
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 26s 10ms/step - loss: 0.0077 - mae:
0.0077 - mape: 327.9712 - val_loss: 0.0075 - val_mae: 0.0075 -
val_mape: 3.2782
Epoch 31/60
2650/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0078 - mae:
0.0078 - mape: 673.9037
Epoch 31: val_loss did not improve from 0.00693
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 27s 10ms/step - loss: 0.0078 - mae:
0.0078 - mape: 673.3951 - val_loss: 0.0075 - val_mae: 0.0075 -
val_mape: 3.2763
Epoch 32/60
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0077 - mae:
0.0077 - mape: 89.7443
Epoch 32: val_loss did not improve from 0.00693
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 26s 10ms/step - loss: 0.0077 - mae:
0.0077 - mape: 89.8227 - val_loss: 0.0089 - val_mae: 0.0089 -
val_mape: 4.3261
Epoch 33/60
2650/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0078 - mae:
0.0078 - mape: 649.8865
Epoch 33: val_loss did not improve from 0.00693
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 27s 10ms/step - loss: 0.0078 - mae:
0.0078 - mape: 649.3015 - val_loss: 0.0082 - val_mae: 0.0082 -
val_mape: 4.0103
Epoch 34/60
2648/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0076 - mae:
0.0076 - mape: 273.9116
Epoch 34: val_loss did not improve from 0.00693
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 24s 9ms/step - loss: 0.0076 - mae:
0.0076 - mape: 273.6801 - val_loss: 0.0072 - val_mae: 0.0072 -
val_mape: 3.4193
```

```
Epoch 35/60
2651/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0077 - mae:
0.0077 - mape: 38.4844
Epoch 35: val_loss did not improve from 0.00693
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 25s 9ms/step - loss: 0.0077 - mae:
0.0077 - mape: 38.6578 - val_loss: 0.0073 - val_mae: 0.0073 -
val_mape: 3.2881
Epoch 36/60
2649/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0077 - mae:
0.0077 - mape: 149.7335
Epoch 36: val_loss did not improve from 0.00693
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 24s 9ms/step - loss: 0.0077 - mae:
0.0077 - mape: 149.7380 - val_loss: 0.0071 - val_mae: 0.0071 -
val_mape: 3.1051
```

Model Evaluation on Test Data

```
from keras.models import load_model

model = load_model(r'C:\Users\PMLS\ML\lab9\E1-cp-0026-loss0.01.h5',
compile=False)

y_pred_scaled   = model.predict(test_X)
y_pred          = scaler.inverse_transform(y_pred_scaled)
y_test_unscaled = scaler.inverse_transform(test_y)
# Mean Absolute Error (MAE)
MAE = np.mean(abs(y_pred - y_test_unscaled))
print('Mean Absolute Error (MAE): ' + str(np.round(MAE, 2)))

# Median Absolute Error (MedAE)
MEDAE = np.median(abs(y_pred - y_test_unscaled))
print('Median Absolute Error (MedAE): ' + str(np.round(MEDAE, 2)))

# Mean Squared Error (MSE)
MSE = np.square(np.subtract(y_pred, y_test_unscaled)).mean()
print('Mean Squared Error (MSE): ' + str(np.round(MSE, 2)))

# Root Mean Squarred Error (RMSE)
RMSE = np.sqrt(np.mean(np.square(y_pred - y_test_unscaled)))
print('Root Mean Squared Error (RMSE): ' + str(np.round(RMSE, 2)))

# Mean Absolute Percentage Error (MAPE)
MAPE = np.mean((np.abs(np.subtract(y_test_unscaled, y_pred)/
y_test_unscaled))) * 100
print('Mean Absolute Percentage Error (MAPE): ' + str(np.round(MAPE,
2)) + ' %')

# Median Absolute Percentage Error (MDAPE)
MDAPE = np.median((np.abs(np.subtract(y_test_unscaled, y_pred)/
y_test_unscaled))) * 100
```

```
print('Median Absolute Percentage Error (MDAPE): ' +
str(np.round(MDAPE, 2)) + ' %')

print('\n\ny_test_unscaled.shape= ',y_test_unscaled.shape)
print('y_pred.shape= ',y_pred.shape)
```

```
379/379 ━━━━━━━━━━━━━━━━━━━━ 3s 8ms/step
Mean Absolute Error (MAE): 111.85
Median Absolute Error (MedAE): 89.2
Mean Squared Error (MSE): 21813.82
Root Mean Squared Error (RMSE): 147.7
Mean Absolute Percentage Error (MAPE): 0.77 %
Median Absolute Percentage Error (MDAPE): 0.61 %


y_test_unscaled.shape=  (12105, 1)
y_pred.shape=  (12105, 1)
```

Checkpoint Configuration and Model Initialization

```
checkpoints = r'C:\Users\PMLS\ML\lab9\New folder\\E2-cp-{epoch:04d}-
loss{val_loss:.2f}.h5'
model = load_model(r'C:\Users\PMLS\ML\lab9\E1-cp-0026-loss0.01.h5',
compile=False)
start_epoch= 27
```

Model Compilation or Loading with PC Architecture and Callbacks

```
import os
from keras.models import load_model
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint

# === Model Checkpoint Configuration ===
checkpoints = r'C:\Users\PMLS\ML\lab9\New folder\E2-cp-{epoch:04d}-
loss{val_loss:.2f}.h5'
model_path = r'C:\Users\PMLS\ML\lab9\E1-cp-0026-loss0.01.h5'
start_epoch = 27

# Load model from checkpoint without compiling
print(f"[INFO] loading model from {model_path}...")
model = load_model(model_path, compile=False)

# Recompile the model
print("[INFO] recompiling model...")
opt = Adam(1e-4)  # New learning rate
model.compile(loss='mae', optimizer=opt, metrics=["mae", "mape"])

# Confirm learning rate
print(f"[INFO] new learning rate:
```

```
{model.optimizer.learning_rate.numpy()}")

# Construct the callback to save only the best model to disk
EpochCheckpoint1 = ModelCheckpoint(
    checkpoints,
    monitor="val_loss",
    save_best_only=True,
    verbose=1
)

# Register callbacks
callbacks = [EpochCheckpoint1]

# === Your model is ready to train ===
# model.fit(trainX, trainY,
#           validation_data=(valX, valY),
#           epochs=100,
#           initial_epoch=start_epoch,
#           callbacks=callbacks,
#           batch_size=32)
```

```
[INFO] loading model from C:\Users\PMLS\ML\lab9\E1-cp-0026-
loss0.01.h5...
[INFO] recompiling model...
[INFO] new learning rate: 9.999999747378752e-05
```

```
epochs = 10
verbose = 1 #0
batch_size = 32
History = model.fit(train_X,
                        train_y,
                        batch_size=batch_size,
                        epochs = epochs,
                        validation_data = (validation_X,validation_y),
                        callbacks=callbacks,
                        verbose = verbose)
```

```
Epoch 1/10
2652/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 0.0067 - mae:
0.0067 - mape: 253.7258
Epoch 1: val_loss improved from inf to 0.00683, saving model to C:\
Users\PMLS\ML\lab9\New folder\E2-cp-0001-loss0.01.h5

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

2653/2653 ━━━━━━━━━━━━━━━━━━━━ 52s 17ms/step - loss: 0.0067 - mae:
0.0067 - mape: 253.6871 - val_loss: 0.0068 - val_mae: 0.0068 -
```

```
val_mape: 3.1820
Epoch 2/10
2651/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 0.0066 - mae:
0.0066 - mape: 227.4288
Epoch 2: val_loss improved from 0.00683 to 0.00664, saving model to
C:\Users\PMLS\ML\lab9\New folder\E2-cp-0002-loss0.01.h5

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

2653/2653 ━━━━━━━━━━━━━━━━━━━━ 43s 16ms/step - loss: 0.0066 - mae:
0.0066 - mape: 227.3739 - val_loss: 0.0066 - val_mae: 0.0066 -
val_mape: 2.9391
Epoch 3/10
2651/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 0.0066 - mae:
0.0066 - mape: 31.1545
Epoch 3: val_loss did not improve from 0.00664
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 43s 16ms/step - loss: 0.0066 - mae:
0.0066 - mape: 31.3775 - val_loss: 0.0070 - val_mae: 0.0070 -
val_mape: 3.2917
Epoch 4/10
2652/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0066 - mae:
0.0066 - mape: 239.6783
Epoch 4: val_loss improved from 0.00664 to 0.00653, saving model to
C:\Users\PMLS\ML\lab9\New folder\E2-cp-0004-loss0.01.h5

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

2653/2653 ━━━━━━━━━━━━━━━━━━━━ 37s 14ms/step - loss: 0.0066 - mae:
0.0066 - mape: 239.6617 - val_loss: 0.0065 - val_mae: 0.0065 -
val_mape: 3.0796
Epoch 5/10
2648/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0065 - mae:
0.0065 - mape: 46.8405
Epoch 5: val_loss improved from 0.00653 to 0.00637, saving model to
C:\Users\PMLS\ML\lab9\New folder\E2-cp-0005-loss0.01.h5

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.
```

```
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 16s 6ms/step - loss: 0.0065 - mae:
0.0065 - mape: 47.2282 - val_loss: 0.0064 - val_mae: 0.0064 -
val_mape: 2.8349
Epoch 6/10
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0065 - mae:
0.0065 - mape: 55.3569
Epoch 6: val_loss did not improve from 0.00637
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 17s 6ms/step - loss: 0.0065 - mae:
0.0065 - mape: 55.4267 - val_loss: 0.0068 - val_mae: 0.0068 -
val_mape: 3.1839
Epoch 7/10
2650/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0065 - mae:
0.0065 - mape: 273.6801
Epoch 7: val_loss did not improve from 0.00637
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 17s 7ms/step - loss: 0.0065 - mae:
0.0065 - mape: 273.6450 - val_loss: 0.0067 - val_mae: 0.0067 -
val_mape: 3.1442
Epoch 8/10
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0065 - mae:
0.0065 - mape: 668.7221
Epoch 8: val_loss did not improve from 0.00637
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 17s 6ms/step - loss: 0.0065 - mae:
0.0065 - mape: 668.5556 - val_loss: 0.0065 - val_mae: 0.0065 -
val_mape: 2.8723
Epoch 9/10
2652/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0065 - mae:
0.0065 - mape: 482.3124
Epoch 9: val_loss did not improve from 0.00637
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 17s 6ms/step - loss: 0.0065 - mae:
0.0065 - mape: 482.1121 - val_loss: 0.0066 - val_mae: 0.0066 -
val_mape: 2.9522
Epoch 10/10
2652/2653 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0065 - mae:
0.0065 - mape: 56.6656
Epoch 10: val_loss did not improve from 0.00637
2653/2653 ━━━━━━━━━━━━━━━━━━━━ 18s 7ms/step - loss: 0.0065 - mae:
0.0065 - mape: 56.8069 - val_loss: 0.0068 - val_mae: 0.0068 -
val_mape: 3.0794
```

Model Evaluation on Test Data

```
model = load_model(r'C:\Users\PMLS\ML\lab9\New folder\E2-cp-0005-
loss0.01.h5', compile=False)


y_pred_scaled   = model.predict(test_X)
y_pred          = scaler.inverse_transform(y_pred_scaled)
y_test_unscaled = scaler.inverse_transform(test_y)
# Mean Absolute Error (MAE)
```

```
MAE = np.mean(abs(y_pred - y_test_unscaled))
print('Mean Absolute Error (MAE): ' + str(np.round(MAE, 2)))

# Median Absolute Error (MedAE)
MEDAE = np.median(abs(y_pred - y_test_unscaled))
print('Median Absolute Error (MedAE): ' + str(np.round(MEDAE, 2)))

# Mean Squared Error (MSE)
MSE = np.square(np.subtract(y_pred, y_test_unscaled)).mean()
print('Mean Squared Error (MSE): ' + str(np.round(MSE, 2)))

# Root Mean Squarred Error (RMSE)
RMSE = np.sqrt(np.mean(np.square(y_pred - y_test_unscaled)))
print('Root Mean Squared Error (RMSE): ' + str(np.round(RMSE, 2)))

# Mean Absolute Percentage Error (MAPE)
MAPE = np.mean((np.abs(np.subtract(y_test_unscaled, y_pred)/
y_test_unscaled))) * 100
print('Mean Absolute Percentage Error (MAPE): ' + str(np.round(MAPE,
2)) + ' %')

# Median Absolute Percentage Error (MDAPE)
MDAPE = np.median((np.abs(np.subtract(y_test_unscaled, y_pred)/
y_test_unscaled))) * 100
print('Median Absolute Percentage Error (MDAPE): ' +
str(np.round(MDAPE, 2)) + ' %')

print('\n\ny_test_unscaled.shape= ',y_test_unscaled.shape)
print('y_pred.shape= ',y_pred.shape)
```

```
379/379 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
Mean Absolute Error (MAE): 101.55
Median Absolute Error (MedAE): 80.26
Mean Squared Error (MSE): 18222.1
Root Mean Squared Error (RMSE): 134.99
Mean Absolute Percentage Error (MAPE): 0.7 %
Median Absolute Percentage Error (MDAPE): 0.55 %


y_test_unscaled.shape=  (12105, 1)
y_pred.shape=  (12105, 1)
```