Deep Learning for Short-Term Load Forecasting—A Novel Classification-Based CNN Model

Pulak Mehrotra
2021AATS0017P

December 13, 2023

Contents

0.1 Introduction

The idea of Short-Term Load Forecasting (STLF) for self-contained and controllable environments like Smart Grids, is to predict how the power requirements of the households within the environment. The idea behind this was to control the power supplied to the households using the Smart Grid. However, the general way of doing this was simply using prediction based Neural Networks. This poses the following problems:

1. Using prediction based on the total power requirements of all the households within a given environment, we do not know which particular household or which particular group households is the main contributor for the power change. Knowing this information
2. The prediction models take the erroneous readings of the model into consideration as well, when making the final prediction.

We propose a solution that addresses the first problem in entirety and takes some steps to minimise the effect of the second. For prediction for the environment, we can first do it household-wise. Then, we can consider each appliance in a household as having a certain state, for example, "ON" or "OFF". **These states can be predicted by classifying other features received by the device such as current and voltage into various states.**

0.2 Data Collection and Preprocessing

We use the IAWE dataset here because it contains the most informative data out of the freely available datasets, meaning it had the following data readings readily available for all the appliances making up a household:

- Current
- Voltage
- Frequency of Operation
- Active Power
- Reactive Power
- Previous Power Reading

For preprocessing, we assumed a household had three devices (a TV, a fridge and a coffee maker) and each device had three states. Steps taken:

- Rounded down the power reading to integer values.

- Find unique power readings and and the number of occurences of each
- Created labels for every sample data as the weighted average of the power readings and their occurences

This data was truncated and saved to a CSV file. This data is the one worked on in the below code.

0.3 Model Architecture

- Initialize a sequential model (`model1`) with the following layers:
  - 1D Convolutional layer
  - Dense, Fully Connected Layer with 16 neurons
  - 1D Max Pooling Layer
  - Another Dense layer with number of classes/states as output (here, 3)
- The model was compiled using the following settings:
  - Loss function: Sparse Categorical Cross-Entropy
  - Optimizer: Adam
  - Metric: Accuracy
- Train the model on the provided data (`train_fts` and `train_targets`) with validation data (`test_fts` and `test_targets`) using 10 epochs and a batch size of 32.
- Utilize the `ModelCheckpoint` callback to save the best model during training.

0.4 Achieved Performance and Conclusion

This is the Performance Analysis for the Coffee Maker. As shown below, the model is performing to decent extent for the power values classes, but unfortunately the erroneous values are once again causing the error percentage to go up. This can be altered by changing the bucketing of data to include the erroneous values to an extent as well.

Error Type	**Value**
MAE 1	162.3159902
MSE	26414.2554110
MAPE	0.7058395

0.5 Code

```
#PCA

import pandas as pd

data = pd.read_csv('c_final.csv')
data = data.iloc[:,[1,2,3,4,5,6,7]]

# split data
features = ['i', 'v', 'freq', 'reactive_power','active_power',
    'apparent_power']
x = data.loc[:, features].values
y = data.loc[:,'labels'].values

from sklearn.model_selection import train_test_split
train_fts, test_fts, train_lbl, test_lbl = train_test_split( x, y,
    test_size=0.15, random_state=0)

# standardise features
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

scaler.fit(train_fts) #only fit on training set (best practice)

train_fts = scaler.transform(train_fts)
test_fts = scaler.transform(test_fts)

# reduce dimensions from 6 --> 4
from sklearn.decomposition import PCA
pca = PCA(0.95) #preserves 95% variability

pca.fit(train_fts)

train_fts = pca.transform(train_fts)
test_fts = pca.transform(test_fts)

# reshape according to NN
train_fts = train_fts.reshape(train_fts.shape[0], train_fts.shape[1], 1)
test_fts = test_fts.reshape(test_fts.shape[0], test_fts.shape[1], 1)

# encode targets

import numpy as np

arr = np.unique(train_lbl)
i=0
train_targets = np.array([])

while (i<len(train_lbl)):
```

```
    val = train_lbl[i]

    if (val == arr[0]):
        map = 0
    if (val == arr[1]):
        map = 1
    if (val == arr[2]):
        map = 2

    train_targets = np.append(train_targets,map)
    i = i + 1

arr1 = arr
j=0
test_targets = np.array([])

while (j<len(test_lbl)):
    val1 = test_lbl[j]

    if (val1 == arr1[0]):
        map1 = 0
    if (val1 == arr1[1]):
        map1 = 1
    if (val1 == arr1[2]):
        map1 = 2

    test_targets = np.append(test_targets,map1)
    j = j + 1

import tensorflow
from tensorflow import keras

from keras.models import Sequential
from keras.layers import Dense, Conv1D, Flatten, MaxPooling1D
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.datasets import load_iris
from numpy import unique

model1 = Sequential()
model1.add(Conv1D(64, 2, activation="relu", input_shape=(4,1)))
model1.add(Dense(16, activation="relu"))
model1.add(MaxPooling1D())
model1.add(Flatten())
model1.add(Dense(3, activation = 'softmax'))
model1.compile(loss = 'sparse_categorical_crossentropy',
     optimizer = "adam",
              metrics = ['accuracy'])

EPOCHS = 10
```

```
BATCH_SIZE = 32

from tensorflow.keras.callbacks import ModelCheckpoint
cp = ModelCheckpoint('model1/', save_best_only=True)

history = model1.fit(
    train_fts, train_targets, validation_data =
    (test_fts, test_targets), epochs=EPOCHS,
    batch_size=BATCH_SIZE, verbose=2, shuffle=True, callbacks=[cp])

predictions = model1.predict(test_fts, verbose='0')
for i in range(0, 20):
    print('Prediction: ', predictions[i],
          ', True value: ', test_targets[i])

pred = pd.DataFrame(predictions, columns = ['v1','v2','v3'])

# summation(softmax * actual value)
pred1 = pred.copy()

pred1['v1'] = pred1['v1']*arr[0]
pred1['v2'] = pred1['v2']*arr[1]
pred1['v3'] = pred1['v3']*arr[2]
pred1['final'] = pred1['v3'] + pred1['v2'] + pred1['v1']

val_test = pd.read_csv('c_final.csv')
val_test = val_test.iloc[8500:,[6]]

from sklearn.metrics import mean_absolute_error, mean_squared_error,
    mean_absolute_percentage_error

print("MAE")
print(mean_absolute_error(pred1['final'], val_test['apparent_power']))
print("MSE")
print(mean_squared_error(pred1['final'], val_test['apparent_power']))
print("MAPE")
print(mean_absolute_percentage_error(val_test['apparent_power'],
    pred1['final']))
```

Listing 1: PCA and Neural Network Code

0.6 References

[1] H. Shi, M. Xu and R. Li, "Deep Learning for Household Load Forecasting—A Novel Pooling Deep RNN," in IEEE Transactions on Smart Grid, vol. 9, no. 5, pp. 5271-5280, Sept. 2018, doi: 10.1109/TSG.2017.2686012.

[2] A. M. Pirbazari, M. Farmanbar, A. Chakravorty and C. Rong, "Improving Load Forecast Ac-

curacy of Households Using Load Disaggregation Techniques," 2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics), Rhodes, Greece, 2020, pp. 843-851, doi: 10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics50389.2020.00140.

[3] J. Stack, R. G. Harley, P. Springer and J. A. Mahaffey, "Estimation of Wooden Cross-Arm Integrity Using Artificial Neural Networks and Laser Vibrometry," in IEEE Power Engineering Review, vol. 22, no. 12, pp. 66-66, Dec. 2002, doi: 10.1109/MPER.2002.4311942.

[4] C. Dong, C. C. Loy, K. He and X. Tang, "Image Super-Resolution Using Deep Convolutional Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 38, no. 2, pp. 295-307, 1 Feb. 2016, doi: 10.1109/TPAMI.2015.2439281.

[5] A. A. Mamun, M. Sohel, N. Mohammad, M. S. Haque Sunny, D. R. Dipta and E. Hossain, "A Comprehensive Review of the Load Forecasting Techniques Using Single and Hybrid Predictive Models," in IEEE Access, vol. 8, pp. 134911-134939, 2020, doi: 10.1109/ACCESS.2020.3010702.