MEDIUM-TERM LOAD FORECASTING USING TEMPORAL CONVOLUTIONAL NETWORKS FOR FUEL SUPPLY PLANNING IN A UNIVERSITY CAMPUS

Abdel Rahman Al Ladiki, Hadi Noureddine, Mohamad Al Tawil, Mohammad Fares El Hajj Chehade and Nadim Younes

Outline

- Problem Formulation
- Data Pre-processing
- ARIMA
- LSTM
- TCN

Lebanese Power Sector

- Long outages
- Fuel shortage
- AUB diesel problem
- Critical load shutdown

Smart Grids

- Digital communication technologies
- Supply and demand-side managements
- Load-forecasting time series models
- Historical data for future power consumption prediction

Relevant Work

Source	Model(s)
Goswami et Al.I	ARIMA
Lekshmi et Al.	ARIMA
NA et Al.	ANNs
Dehalwar et Al.	ANN, Bagged Regression Trees
Lara-Benitez et Al.	Recurrent Networks (LSTM)

Temporal Convolutional Networks (TCNs)

- 1-Dimensional Convolutional Networks
- Electric vehicle charging
- Nationwide demand
- Lower computational and memory demand
- More accurate results

Classes of Load Forecasting

Type	Duration
Short-term Forecasting	1 hour - 1 week
Medium-term Forecasting	1 week - 1 year
Long-term Forecasting	1 year +

Gaps in the Literature

- Rare work on medium-term forecasting
- Possible failure of ARIMA models
- Expensive weather data infrastructure
- Unique campus load profile

Problem Statement

- develop a multi-step temporal convolutional network forecasting model
- predict the electric power consumption in the next week based on historical data
- compare the model to ARIMA and LSTM models

Contributions

(1) TCNs for medium-term forecasting
(2) University campus load
(3) Customized TCN architecture

Dataset

- Dataset for the daily energy consumption of the University of British Columbia (UBC) campus from mid-2019 till mid-2021

```
df.head()
```

	Date	Boulevard Lot Elec Main Meter Energy (kWh)	Asian Centre Elec Main Meter Energy (kWh)	Beaty Elec Main Meter Energy (kWh)	Biomed Elec Main Meter Energy (kWh)	Ansoc Elec Main Meter Energy (kWh)	Bookstore-NCE Elec Main Meter Energy (kWh)	Baseball Training Facility Elec Main Meter Energy (kWh)	Allard Hall Elec Main Meter Energy (kWh)	AMS Nest Elec Main Meter Energy (kWh)	...	Thea Koerner House Elec Main Meter Energy (kWh)	TotemParkRes Salish Haida Elec Main Meter Energy (kWh)	TotemParkRes Shuswap Kwakiutl Elec Main Meter Energy (kWh)	University Centre Elec Main Meter Energy (kWh)
0	6/30/2019	NaN	422.0	9182.50	3743.0	1057.5	2711.0	301.593750	3198.25	7873.5	...	985.5	157.3750	44.781250	829.5
1	7/1/2019	NaN	417.5	9640.00	4170.5	863.0	2706.0	227.062500	3235.25	7607.0	...	961.5	151.9375	40.117188	789.5
2	7/2/2019	NaN	726.5	9830.25	4451.0	1225.5	3165.0	352.414062	5667.50	9429.0	...	1740.5	124.5625	40.171875	1205.5
3	7/3/2019	NaN	702.0	10050.50	4321.5	1238.0	3218.0	358.953125	5658.00	9692.5	...	1750.0	128.8125	40.695312	1234.0
4	7/4/2019	NaN	689.5	9582.50	4325.0	1265.5	3217.0	372.164062	5824.75	9662.5	...	1686.5	147.5000	42.085938	1276.5

5 rows × 108 columns

Data Preprocessing

1. Filling NaN values with their corresponding columns' means

```
for i in range(1,len(df. columns)-1):
    df[columns[i]] = df[columns[i]].fillna(df[columns[i]].mean());
```

2. Getting the total energy consumption for UBC

```
df["Total Energy Consumption"] = df[columns[1:]].sum(axis = 1);
```

3. Generating some important features: Day, Month, Year, Day of the year, Month of the Year and Holiday

Date	Total Energy Consumption	Holiday	Day	Month	Year	Q	Dayofyear	Dayofmonth	Weekofyear	Temperature
2019-06-30	485583.391855	0	6	6	2019	2	181	30	26	12
2019-07-01	502424.567635	1	0	7	2019	3	182	1	27	11
2019-07-02	547598.100838	0	1	7	2019	3	183	2	27	6
2019-07-03	551544.032479	0	2	7	2019	3	184	3	27	7
2019-07-04	550828.395760	0	3	7	2019	3	185	4	27	6

4. Normalizing the data using the mean and standard deviation

```
columns=["Total Energy Consumption","Day","Month","Q","Dayofmonth","Weekofyear"]
df[columns] = (df[columns]-df[columns].mean())/(df[columns].std())
df.head()
```

Date	Total Energy Consumption	Holiday	Day	Month	Year	Q	Dayofyear	Dayofmonth	Weekofyear	Temperature
2019-06-30	-0.284580	0	1.499235	-0.150488	2019	-0.453518	181	1.614774	-0.049708	12
2019-07-01	0.040518	1	-1.495144	0.139400	2019	0.441294	182	-1.673324	0.01609	11
2019-07-02	0.912536	0	-0.996081	0.139400	2019	0.441294	183	-1.559942	0.01609	6
2019-07-03	0.988707	0	-0.497018	0.139400	2019	0.441294	184	-1.446559	0.01609	7
2019-07-04	0.974892	0	0.002045	0.139400	2019	0.441294	185	-1.333176	0.01609	6

ARIMA Model

An Autoregressive Integrated Moving Average (ARIMA) model uses time series data to forecast future trends.

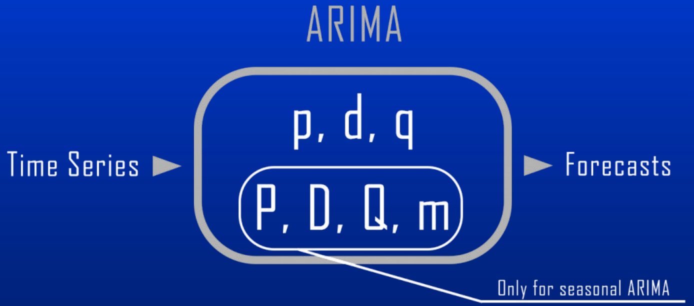
Can be considered as:

- Autoregression (AR): $Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_p Y_{t-p} + \epsilon_1$
- Moving Average (MA): $Y_t = \alpha + \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + .. + \phi_q \epsilon_{t-q}$
- ARIMA: $Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_p Y_{t-p} \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + .. + \phi_q \epsilon_{t-q}$ + Integrated (I)

An ARIMA Model has 3 parameters:

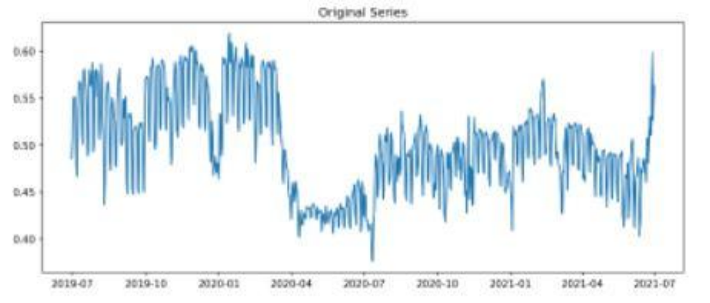
ARIMA (p,d,q):

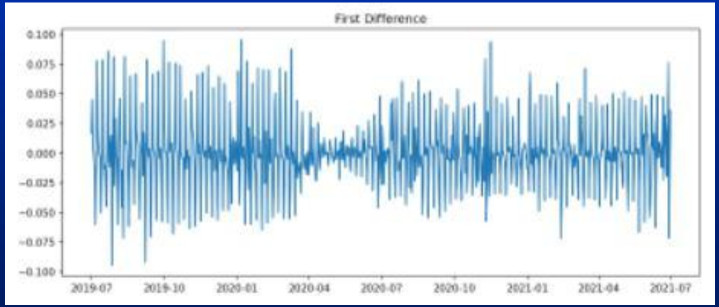
- p : lag order
- d : degree of differencing (lower the p-value)
- q : order of the moving average



Note: p-value can be evaluated using Augmented Dickey Fuller Test - decreases with increase of d

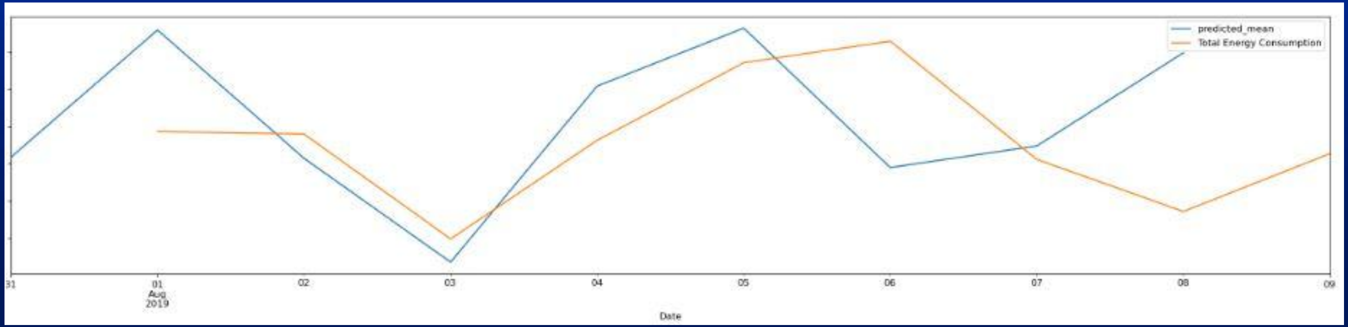
Results of differencing:



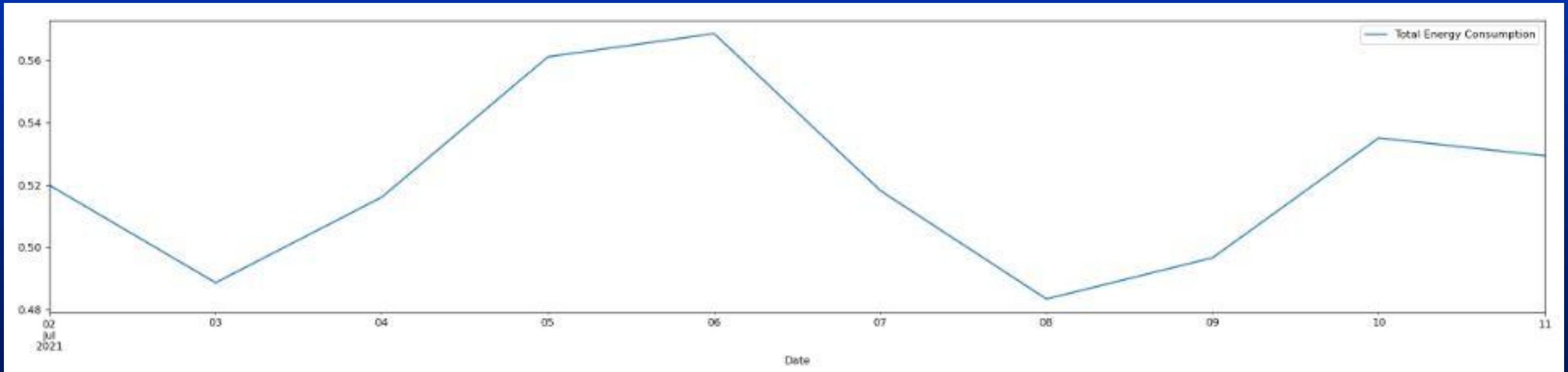


AUTO_ARIMA from PMDARIMA library

- Computes optimal values of parameters on its own by finding the minimum AIC (Akaike's Information Criterion): $AIC = -2\log(L) + 2(p+q+k+1)$; L is likelihood of data and k=1

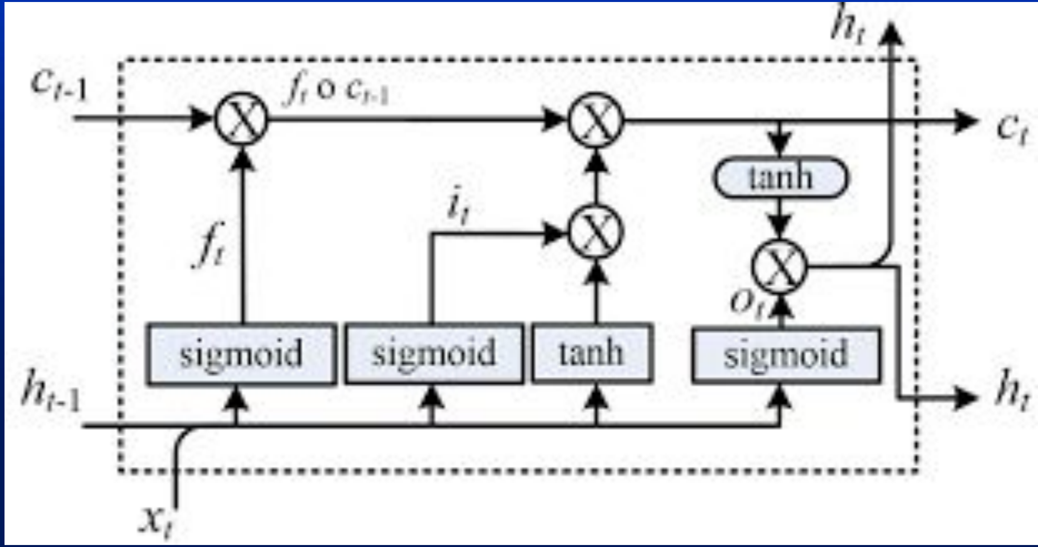


Predictions for the next 10 days using the ARIMA model

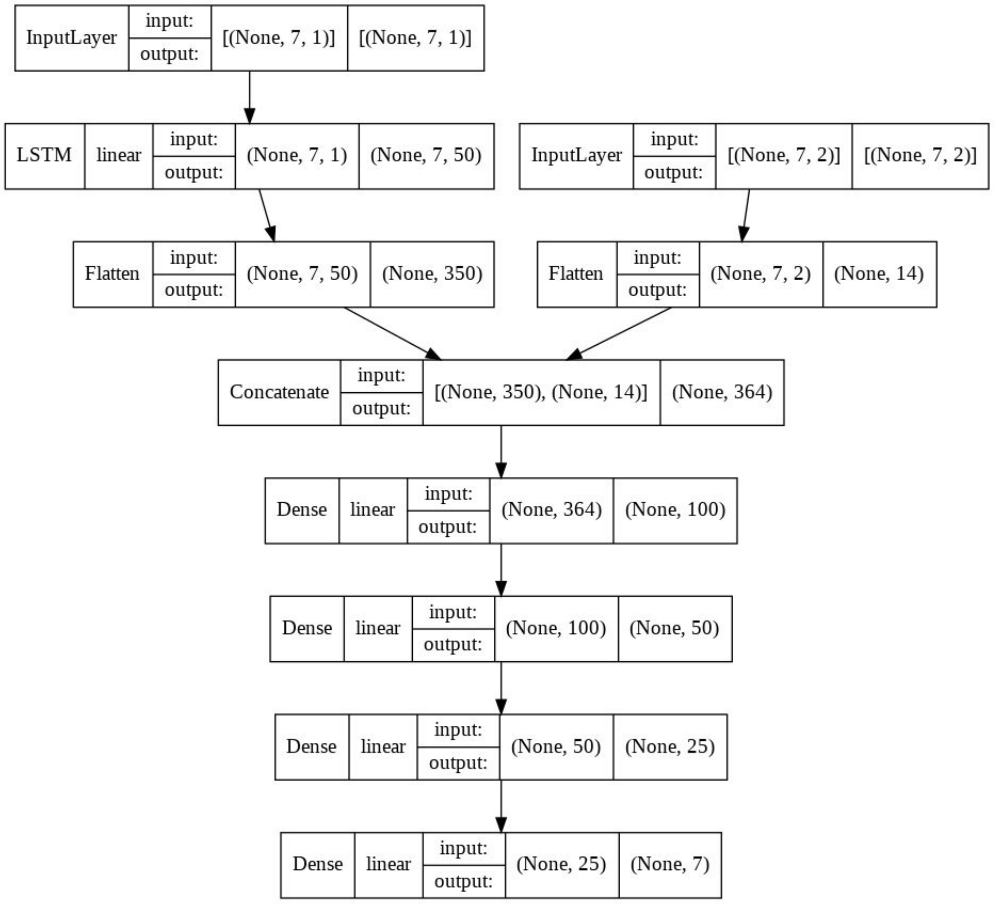


LSTM Model

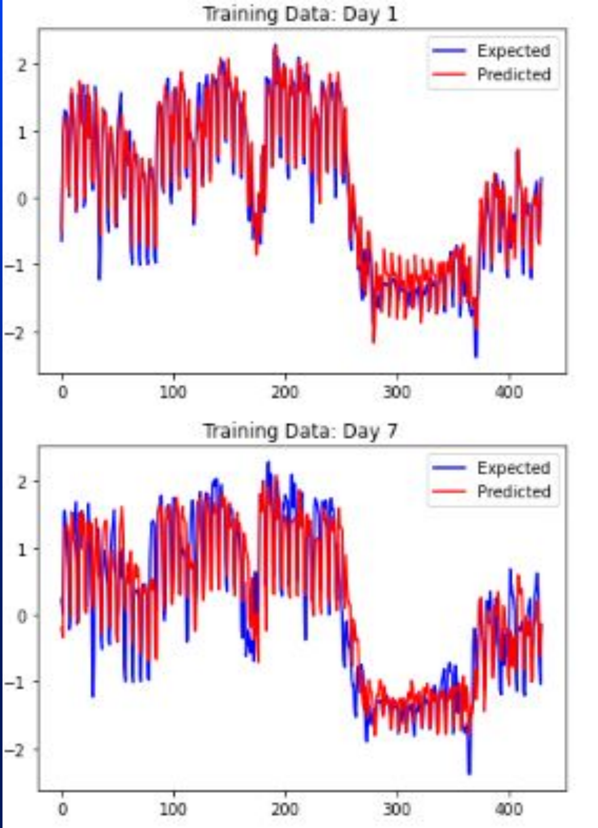
- Long short-term memory (LSTM) is one of the most used RNN architectures for time series predictions. Unlike standard feedforward architectures , LSTM has feedback connections that help identify the importance of new inputs and how much they would impact the next state.

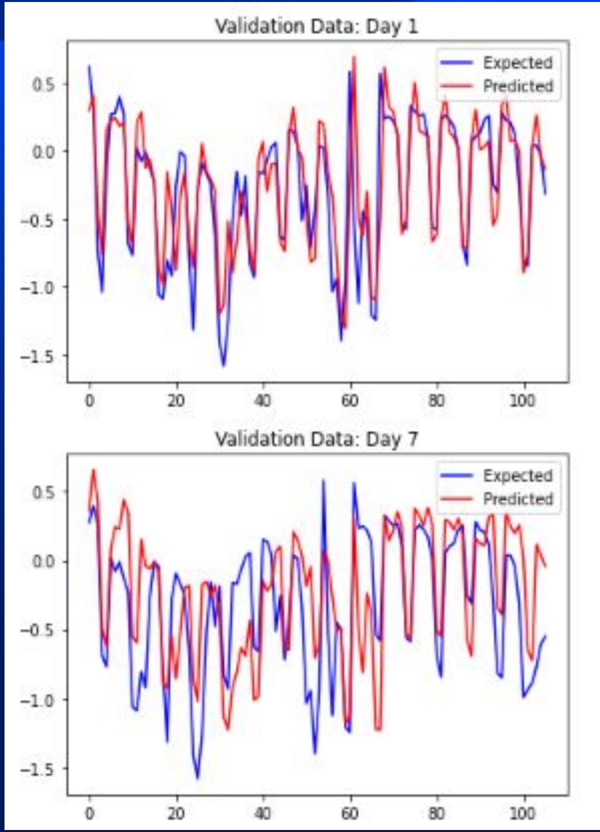


LSTM Model Adapted



LSTM Model Results(Graphically)





LSTM Model Results(Numerically)

Correlation: 0.626
Mean Absolute Error: 17193.955
Mean Absolute Percentage Error: 0.036
Mean Error: 2310.123
Minmax Error: 0.0349
Mean Percentage Error: 0.006
Root Mean Squared Error: 22092.5

TCN Model

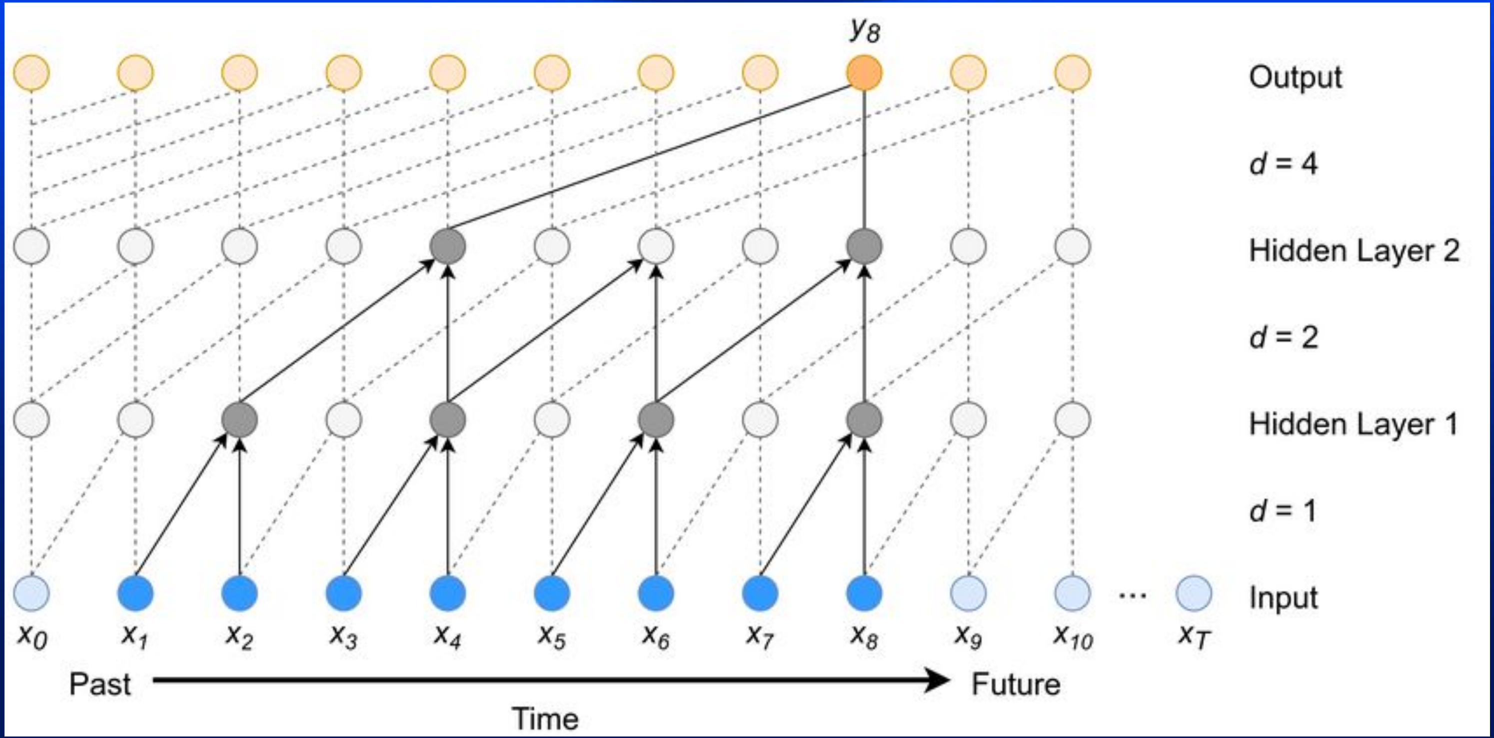
Temporal Convolutional Network: 1 Dimensional CNN

A rather new alternative to RNN for Time Series Forecasting

Based on Causal Convolution and Dilation (explained next slide)

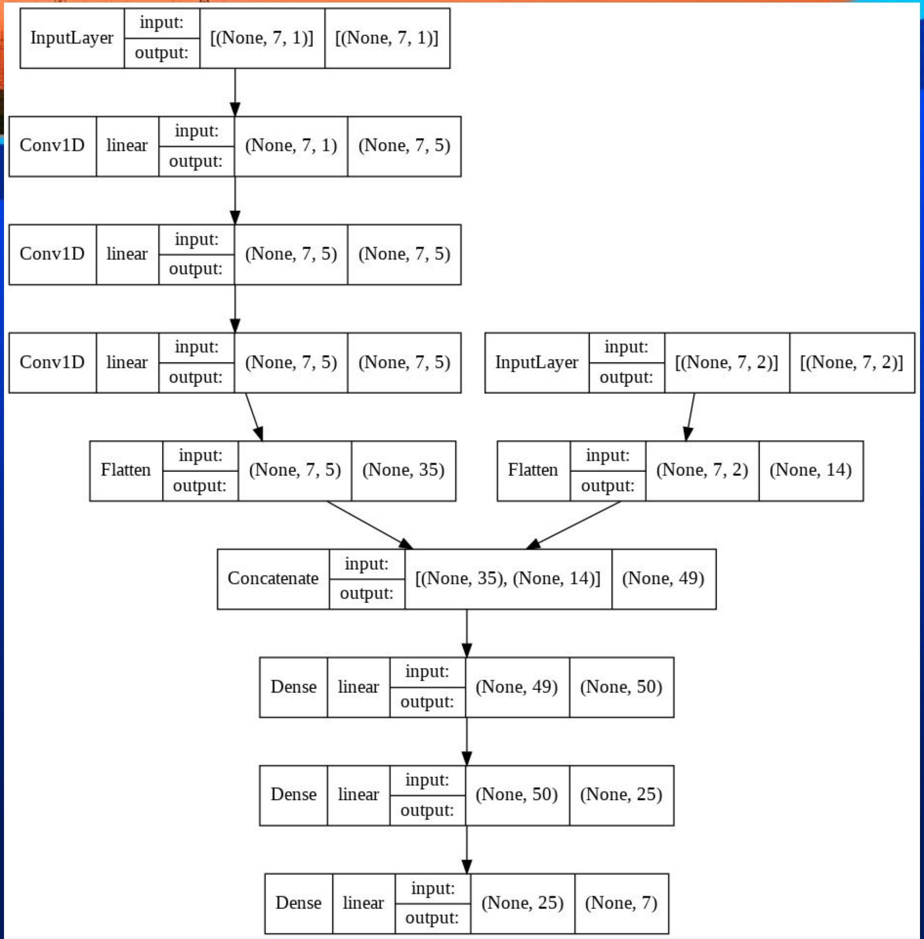
Typical TCN models are multiple blocks of only Convolution layers:
→ Our model takes other features and combines them with the convolution in an ANN

Convolution Block



Our Proposed Architecture

Created as a Functional model
keras.models: Model
keras.layers: Concatenate
keras.utils.vis_utils: plot_model



But this model over complicated the problem…

What went wrong

We tested a model with only one convolution layer with no dilation

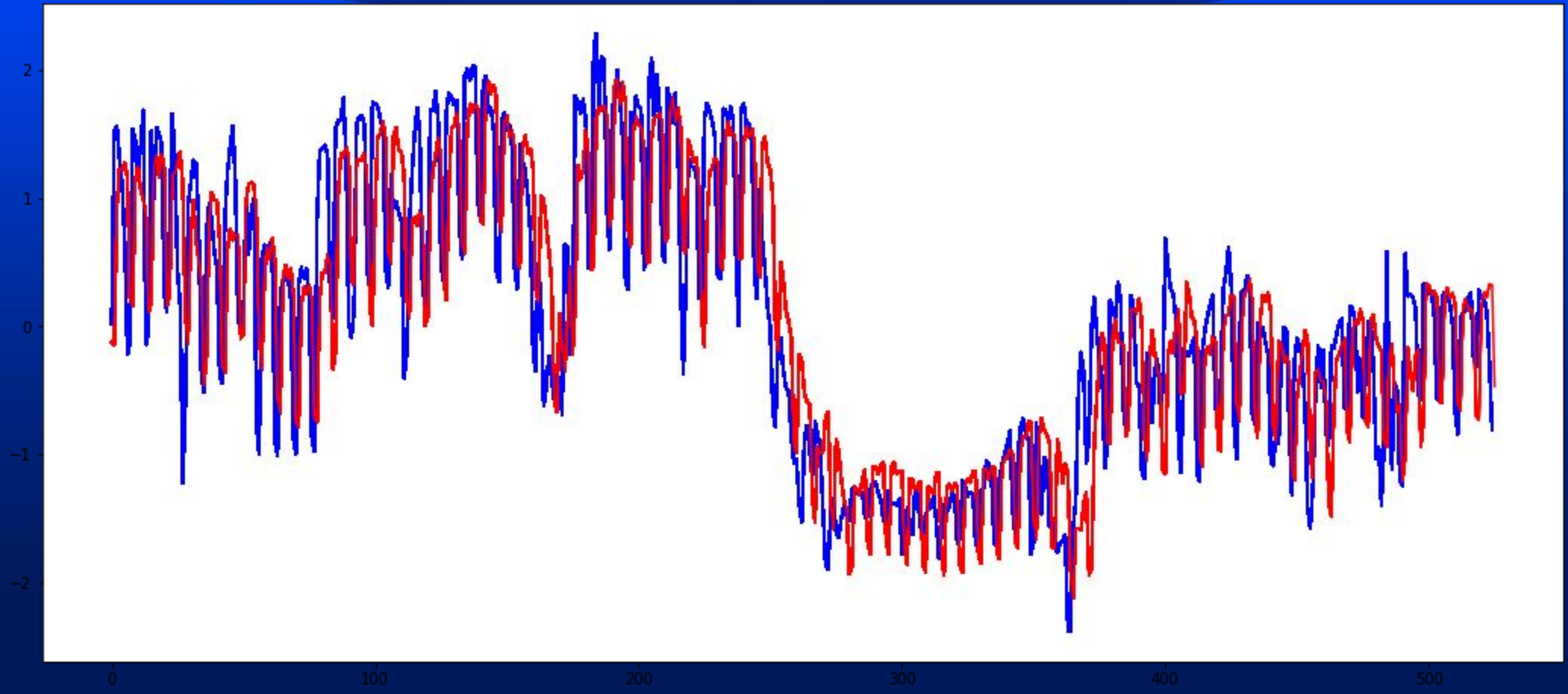
$\longrightarrow$ Gave very similar results!

By taking a closer look at the predictions we concluded that:

Our model is a sophisticated way of copying the previous week!

Sudden changes if average from week to week confused the model

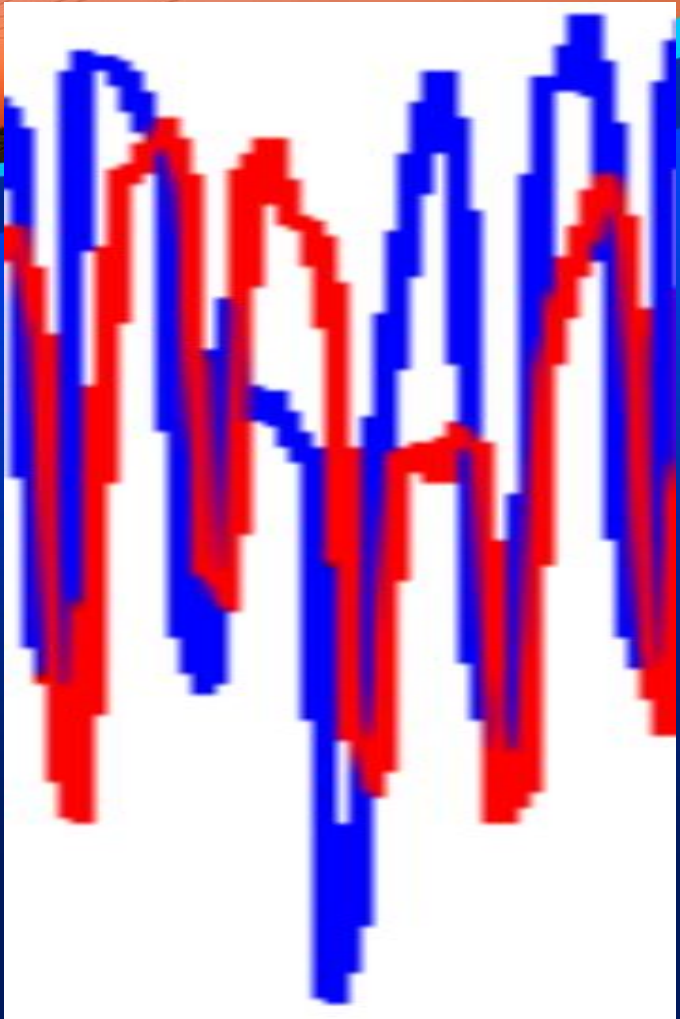
Predictions vs Target



We believe the problem is a missing feature

Proposed Solutions:

- Inputting the Calendrical features of the predicted week not the previous
- Including Temperature as another feature



TCN Model Results(Numerically)

Correlation: 0.6749
Mean Absolute Error: 17125.758
Mean Absolute Percentage Error: 0.0358
Mean Error: 4964.39
Minmax Error: 0.03447
Mean Percentage Error: 0.011
Root Mean Squared Error: 21684.1

Final Words

For the project final report:

- Compare our 3 models with Weighted Absolute Percentage Error (Proposed in one of the referenced publications)
- Improve our input data to deal with sharp week to week changes

Thank you for your attention! We welcome any comment, suggestion, or question.