Time series analysis and forecasting of energy demand with Recurrent neural networks

MAP573

Ari JÓHANNESSON
Francisco CORREIA
Pedro MACEDO
Raphael MENDES

Data Transformation

- Data had:
  - 'Non-practical' format
  - Missing values
- Zone 9 removed

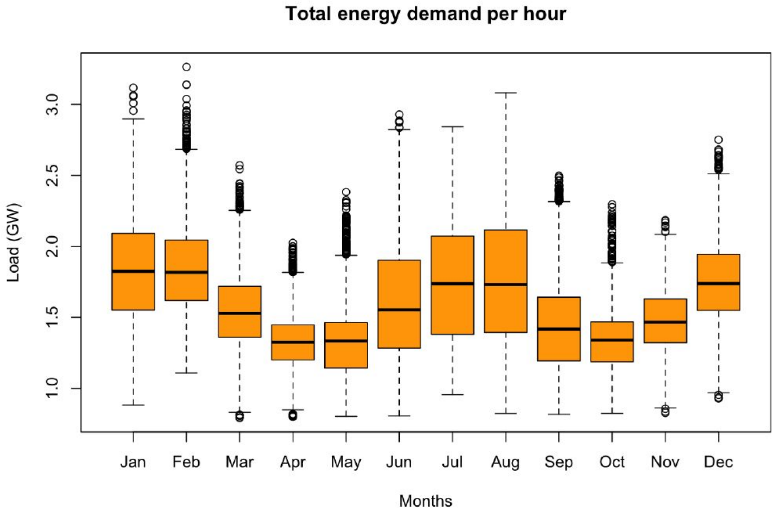
Zone	Year	Month	Date	h1	h2 $\cdots$ h24
1	2004	1	1	16853	16450 $\cdots$
1	2004	1	2	14155	14038 $\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
20	2008	7	7	71263	67560 $\cdots$

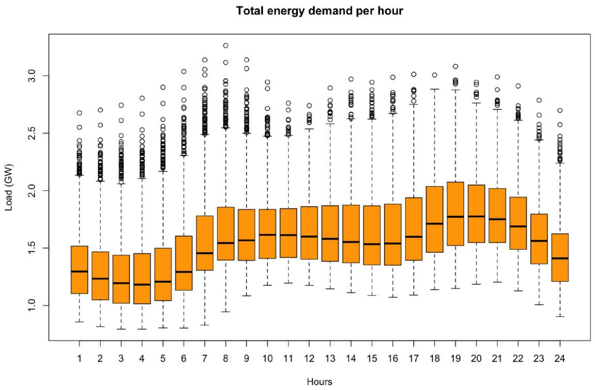
↓

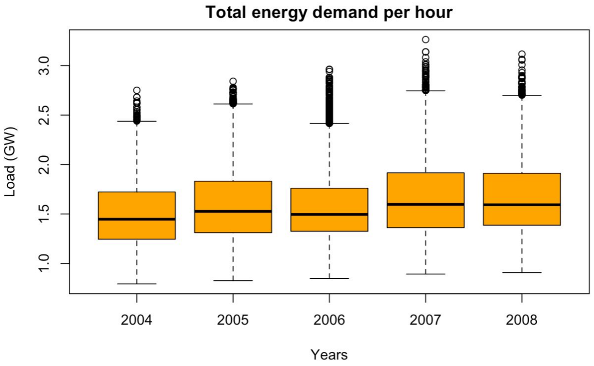
Date	$Z1$	$Z2 \cdots Z20$
2004-1-1 1:00:00	16853	126259 $\cdots$ 79830
2004-1-1 2:00:00	16450	123313 $\cdots$ 77429
$\vdots$	$\vdots$	$\vdots$
2008-7-7 24:00:00	16690	163029 $\cdots$ 85887

Data Statistics

- Relationship between energy demand and temperature
- Energy demand increases throughout the day
- Years do not differ significantly

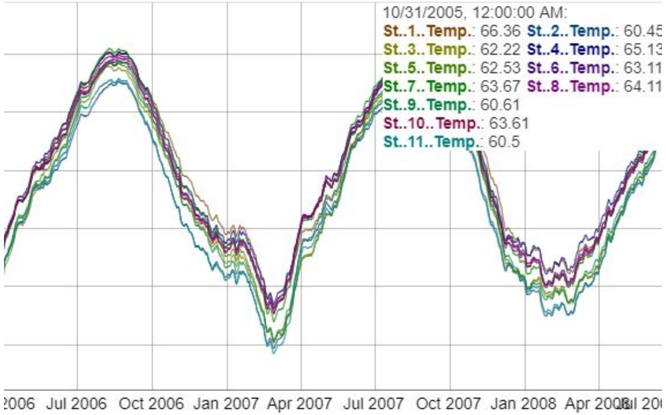


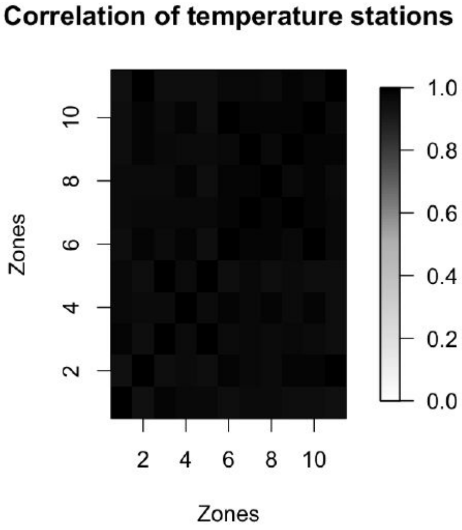


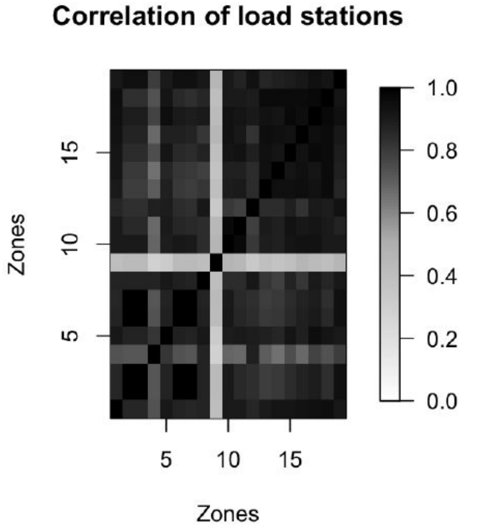


Zone Correlation

- Temperature stations highly correlated
  - r = 0.97
- Load stations correlate
  - On average r = 0.84
  - Except zone 8 (r = 0.4)

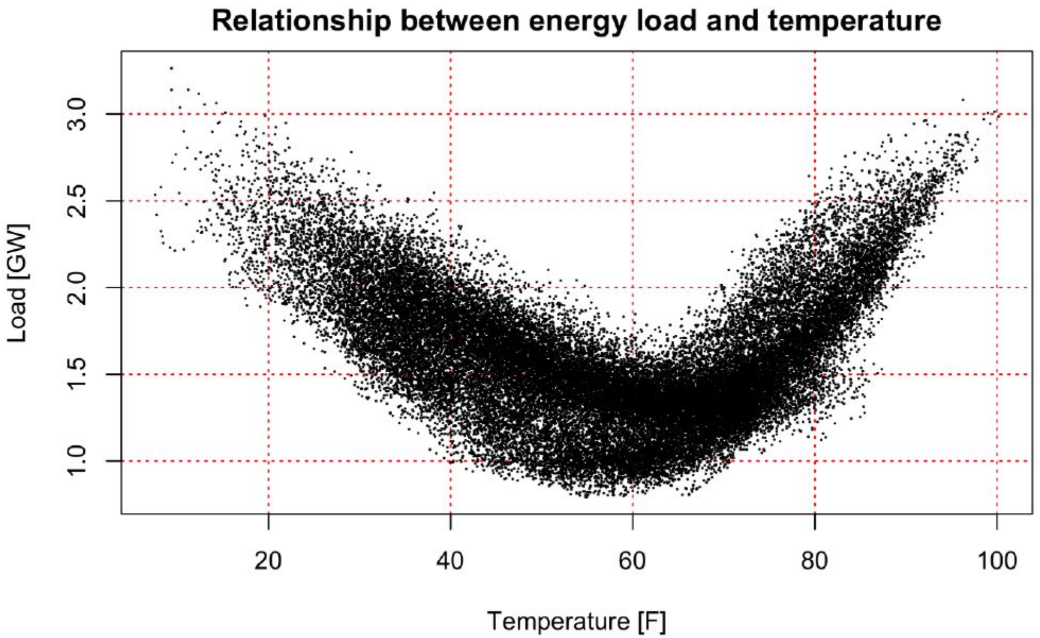






Energy Demand and Temperature

- Further evidence of correlation between energy demand and temperature
  - Hot/cold months have higher demand

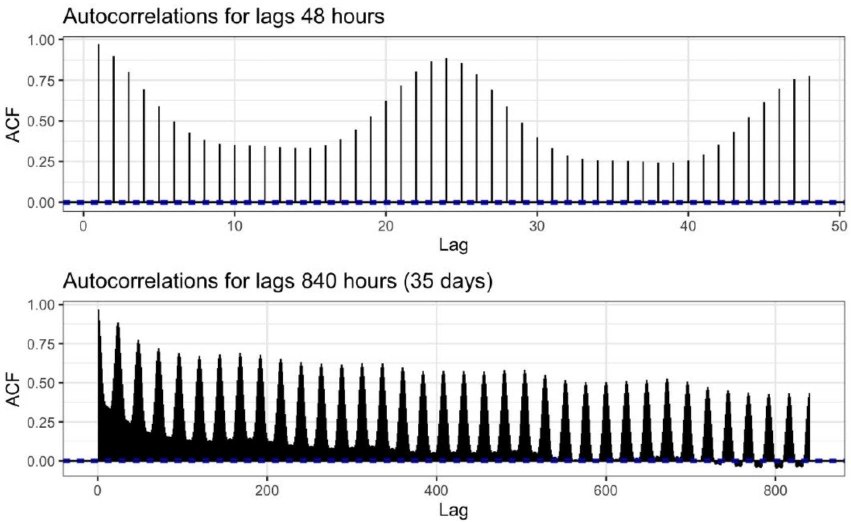


$\mu_{\text{Temperature}} = 57.4$ $\mu_{\text{Load}} = 1,582,393$

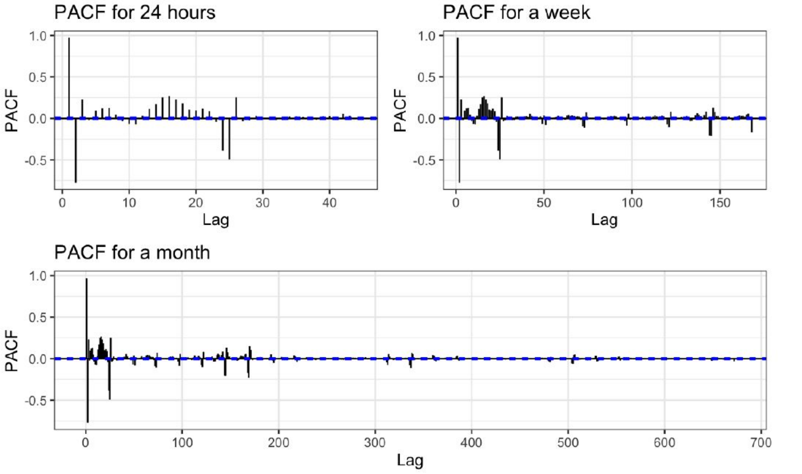
$\sigma_{\text{Temperature}} = 17.2894$ $\sigma_{\text{Load}} = 379245.7$

ACF and PACF

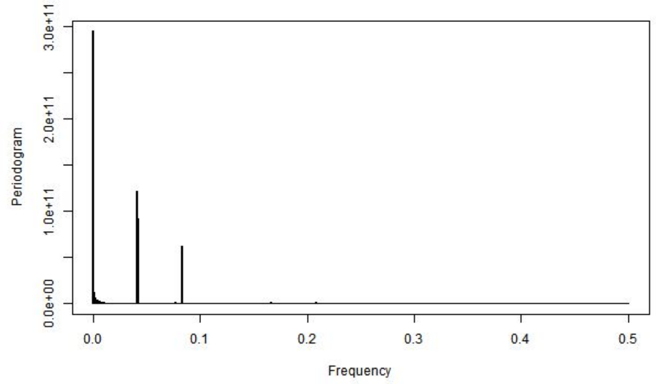
- **ACF:** Correlation of a time series with delayed copy of itself as a function of delay
- **PACF:** Controls for other lags

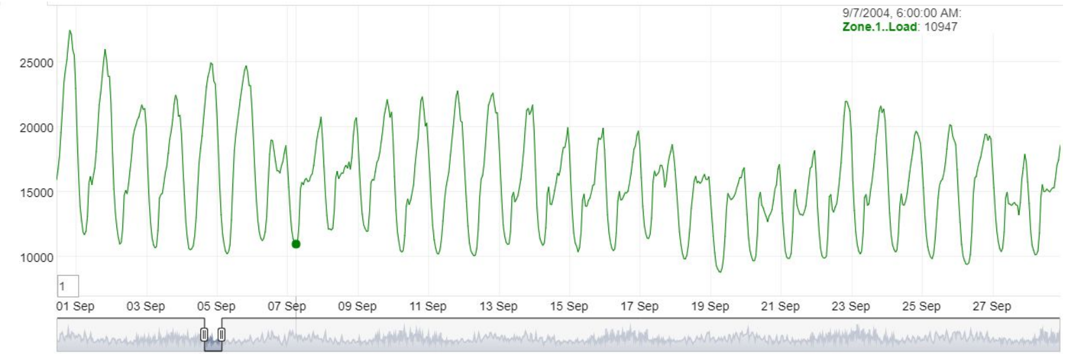


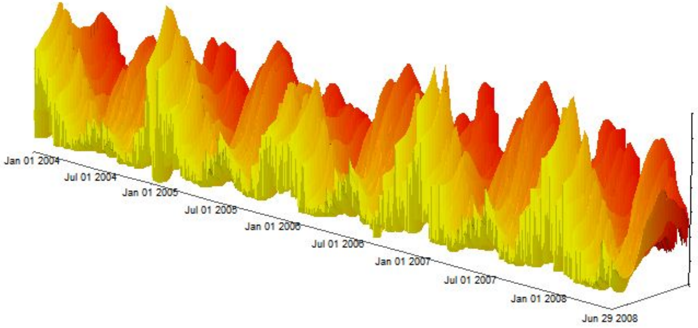
Lags	Time	PACF
1	1 hour	0.9709550
2	2 hours	-0.7737019
25	25 hours	-0.4934457
24	24 hours	-0.3872113
16	16 hours	0.2667820
26	26 hours	0.2520593
15	15 hours	0.2517194
3	3 hours	0.2303735
17	17 hours	0.2301514
169	7 days and 1 hour	-0.2287006
145	6 days and 1 hour	-0.2076716
144	6 days	-0.2046309
18	18 hours	0.1829837
14	14 hours	0.1710621
168	7 days	-0.1709017

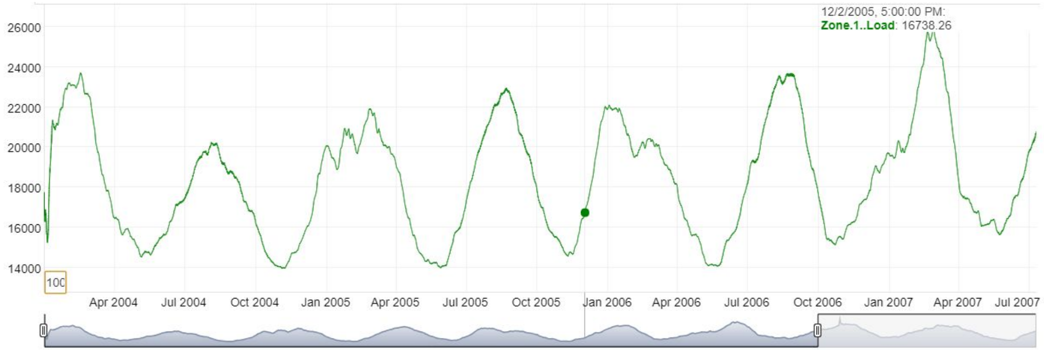


Fourier Analysis- Extracting Seasonalities









Data of Zone Load 1. In the bottom left corner, showing 1 point each 1000 anothers

Fourier Analysis

Data of Load zones and temperatures stations have similar spectral patterns:

Load 3

freq	spec	ttime
0.000225	50397.332	4444.44444
0.041675	910676.799	23.99520
0.005950	7431.144	168.06723
0.083325	37415.358	12.00120
0.041550	7314.805	24.06739
0.041650	238724.696	24.00960
0.041775	41902.423	23.93776
0.000025	26074.008	40000.00000
0.000125	4164825.574	8000.00000

Week

185 days (~ 6 months)

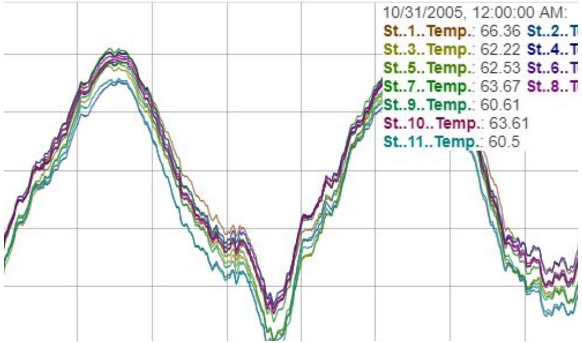
Day/ Half Day

Load 1

freq	spec	ttime
0.000225	74830.326	4444.44444
0.041550	6671.546	24.06739
0.041775	24016.055	23.93776
0.041675	434769.696	23.99520
0.083325	26247.817	12.00120
0.000025	51773.199	40000.00000
0.041650	119681.534	24.00960
0.083350	6708.850	11.99760
0.000675	3717.744	1481.48148

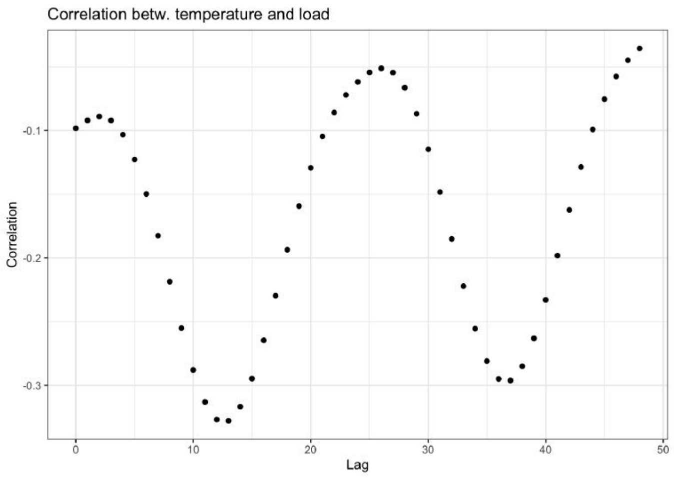
Temp Stations

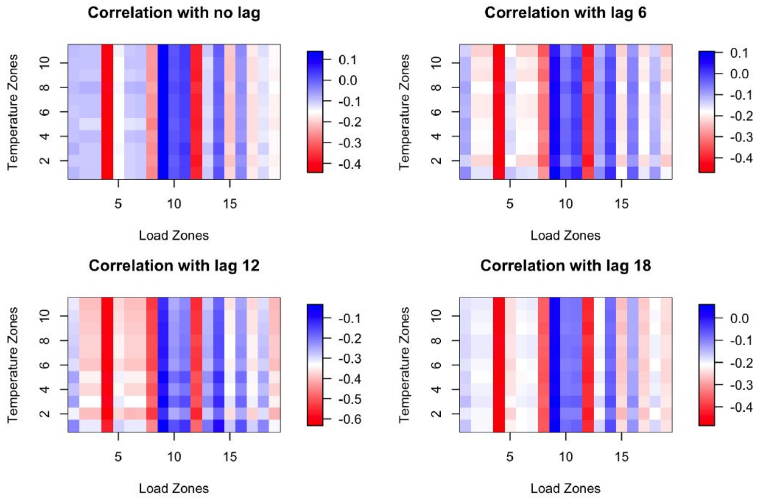
freq	spec	time	
0.000125	4010629.9	8000.0000	11,1 months
0.000100	2166167.8	10000.0000	13,9 months
0.041675	1003809.7	23.9952	
0.000150	372514.3	6666.6667	9,25 months
0.000075	292315.6	13333.3333	
0.041650	287362.0	24.0096	
0.000175	223357.1	5714.2857	8 months
0.000200	159632.7	5000.0000	
0.000250	100546.4	4000.0000	



The Recency Effect

- Energy demand partially determined by temperatures of preceding hours
- Zones might be affected differently
- Lag 12 produced the most correlation
- Few load zones more correlated to all temperature stations





The Recency Effect

Article: "Electric load forecasting with recency effect: A big data approach" **Pu Wang, Bidong Liu, Tao Hong**

https://www.sciencedirect.com/science/article/pii/S0169207015001557?via%3Dihub

- Uses the same data (excluding zone 9)
- Basically a regression model:

$$y_t = \underbrace{\beta_0 + \beta_1 Trend + \beta_2 M_t \beta_3 W_t + \beta_4 H_t + \beta_5 W_t H_t}_{linear\ regressor} + \underbrace{f(T_t)}_{\textbf{polynomial linear regressor}} + \underbrace{\sum_d f(T_{avg(t,d)}) + \sum_h f(T_{t-h})}_{recency\ effect\ terms}$$

$h \backslash d$	0	1	2	3	4	5	6	7
0	4.89	4.10	4.09	4.16	4.23	4.13	4.12	4.25
1	4.55	3.92	3.91	3.97	4.05	3.95	3.96	4.08
2	4.34	3.81	3.80	3.87	3.94	3.85	3.85	3.97
3	4.20	3.76	3.74	3.81	3.88	3.79	3.79	3.91
4	4.09	3.71	3.70	3.76	3.84	3.75	3.74	3.86
5	4.00	3.68	3.67	3.73	3.81	3.72	3.71	3.83
6	3.93	3.65	3.64	3.71	3.78	3.70	3.69	3.81
7	3.86	3.63	3.62	3.69	3.76	3.68	3.67	3.80
8	3.81	3.60	3.60	3.67	3.75	3.67	3.66	3.78
9	3.77	3.59	3.58	3.65	3.73	3.66	3.65	3.78
10	3.74	3.58	3.57	3.64	3.72	3.65	3.64	3.77
11	3.73	3.57	3.55	3.63	3.71	3.64	3.63	3.76
12	3.71	3.56	**3.54**	3.62	3.69	3.63	3.62	3.75
13	3.69	3.56	3.54	3.62	3.68	3.62	3.62	3.74
14	3.67	3.57	3.55	3.63	3.69	3.63	3.63	3.76
15	3.66	3.58	3.57	3.64	3.70	3.64	3.64	3.77
16	3.67	3.60	3.58	3.66	3.71	3.66	3.66	3.79
17	3.67	3.62	3.61	3.68	3.73	3.68	3.68	3.81
18	3.67	3.64	3.63	3.71	3.75	3.70	3.70	3.83
19	3.68	3.67	3.65	3.73	3.77	3.72	3.71	3.86
20	3.68	3.69	3.68	3.75	3.79	3.74	3.73	3.88
21	3.69	3.71	3.70	3.77	3.80	3.76	3.76	3.90
22	3.70	3.73	3.72	3.78	3.81	3.78	3.77	3.92
23	3.72	3.74	3.73	3.78	3.82	3.79	3.79	3.94
24	3.73	3.76	3.75	3.79	3.83	3.80	3.80	3.95
25	3.74	3.76	3.77	3.80	3.84	3.81	3.81	3.96
26	3.75	3.78	3.79	3.81	3.85	3.83	3.82	3.98
27	3.76	3.80	3.81	3.83	3.86	3.84	3.84	4.00

MAPE stats for tuning h and d hyperparameters of the model of the left

The Shiny App

- To visualize and analyse the time series
  - Temperature/Energy demand
  - Compare zone loads and station temperatures
  - See stats for some zones/ some stations
  - See overall stats
  - 3D visualizations
  - Fourier analysis
  - Make automatic Forecast using trained models
- Runs locally, once all the libraries are installed
  - Instructions in README

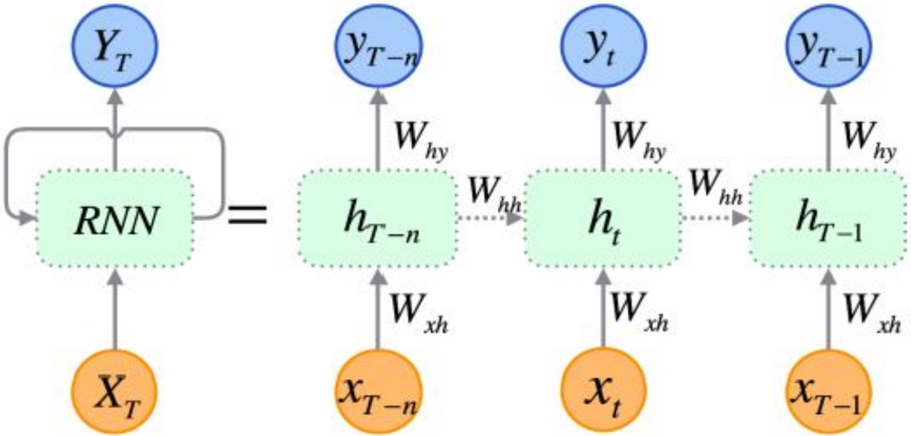
Problem Formulation

Problem 1: Given a time series input X of N hours ($X \in R^{1xN}$) to predict the values Y for the next M hours ($Y \in R^{1xM}$)

Problem 2: Given a time series with a gap Y of M hours ($Y \in R^{1xM}$), use the existing time series X of length N hours ($X \in R^{1xN}$) to predict the gap values.

The models were tested on three tasks (M = 1h, 24h and 168h)

Recurrent neural networks (RNN)



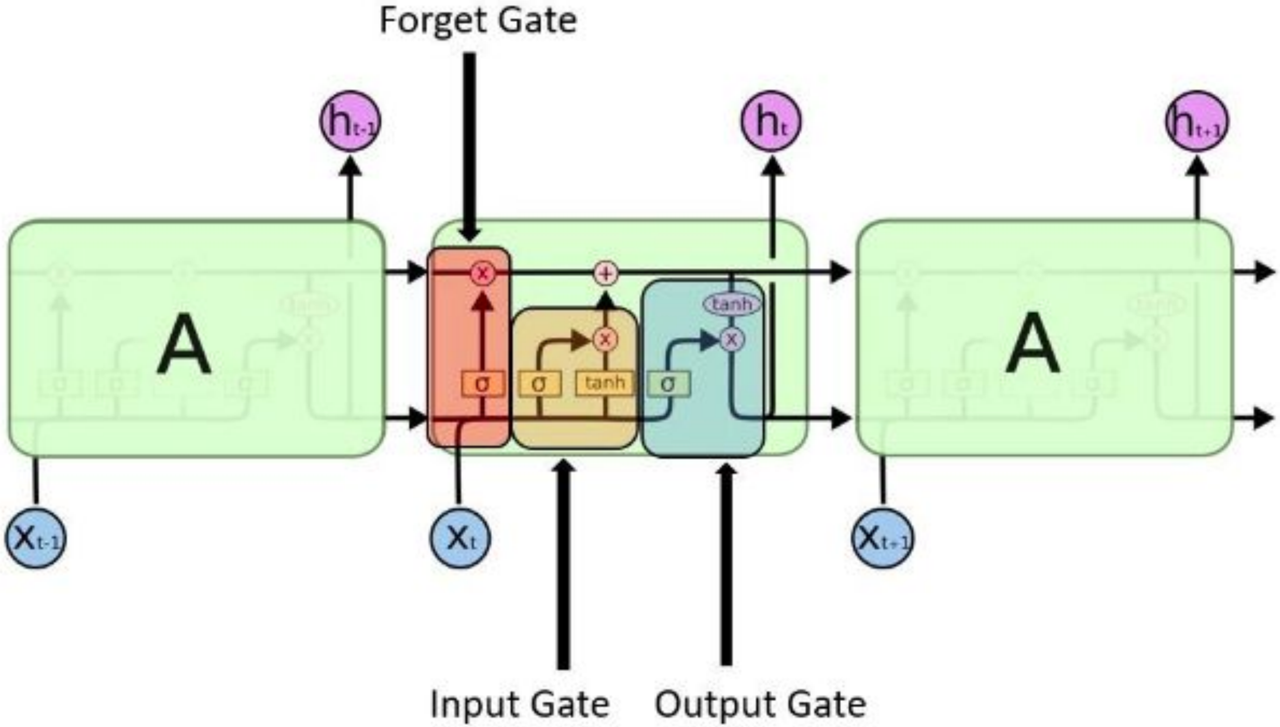
Pro's

- Keeps memory

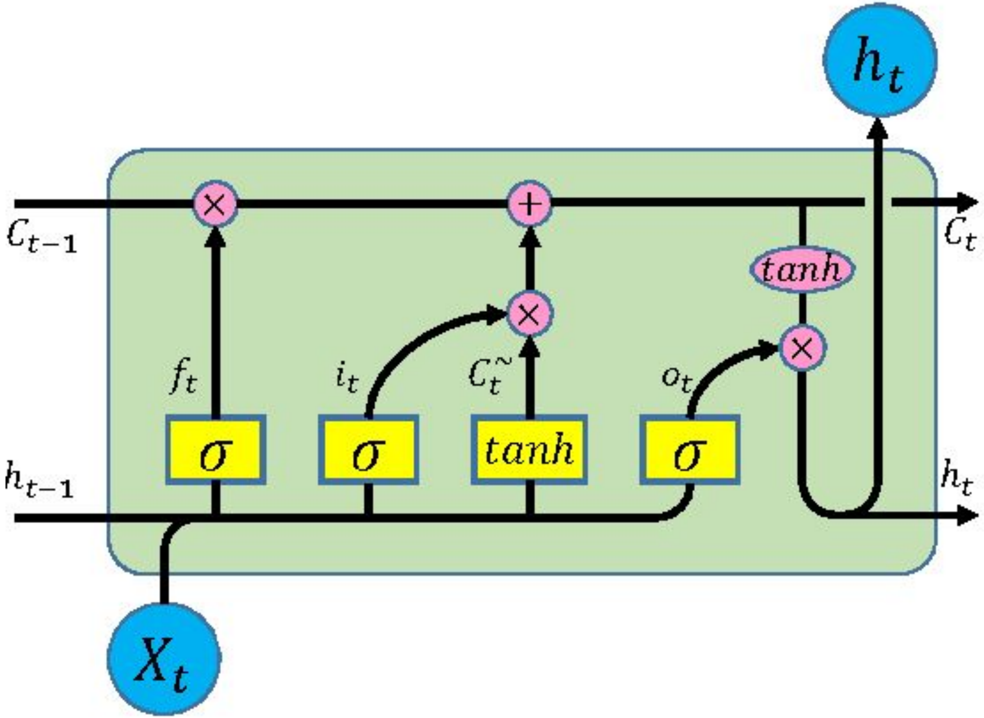
Con's

- Exploding and vanishing gradients problem
- Complex to train

Long short-term memory (LSTM)



Long short-term memory (LSTM)



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

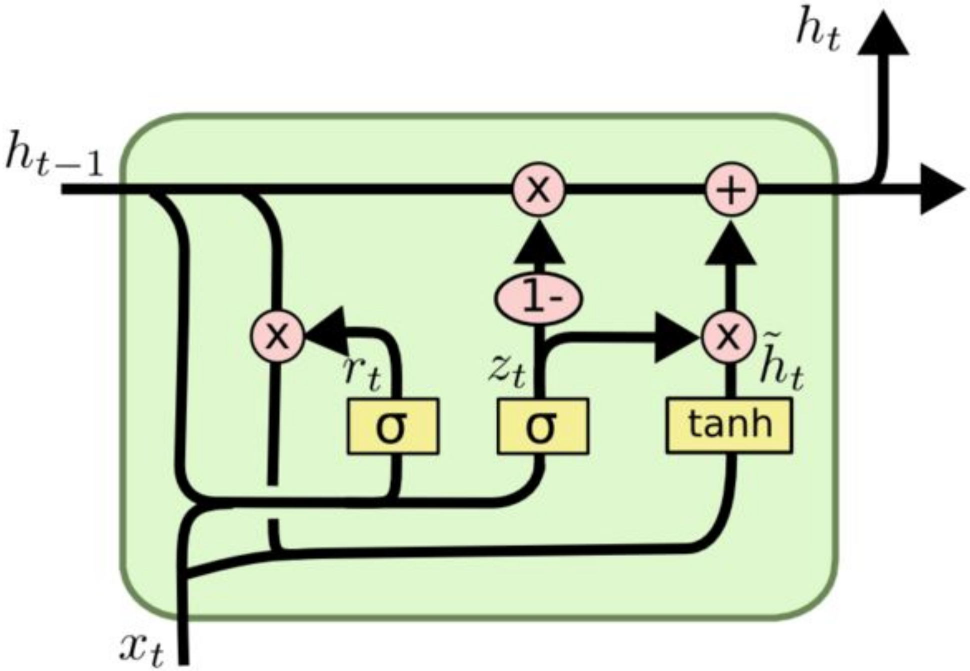
$$\widetilde{C_t} = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C_t}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * tanh(C_t)$$

Gated recurrent unit (GRU)



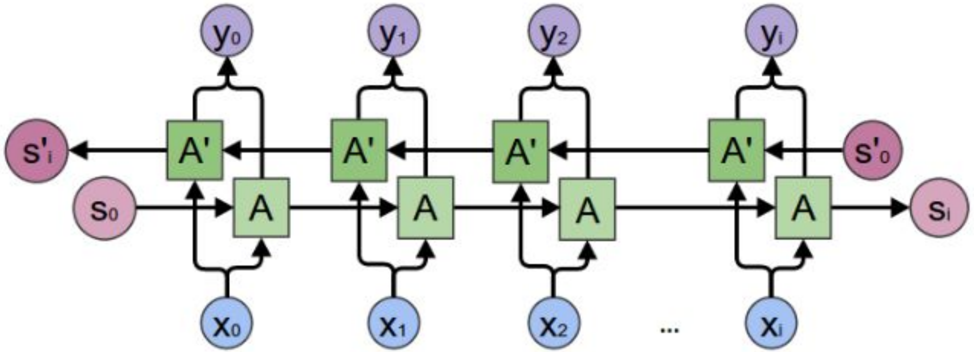
$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

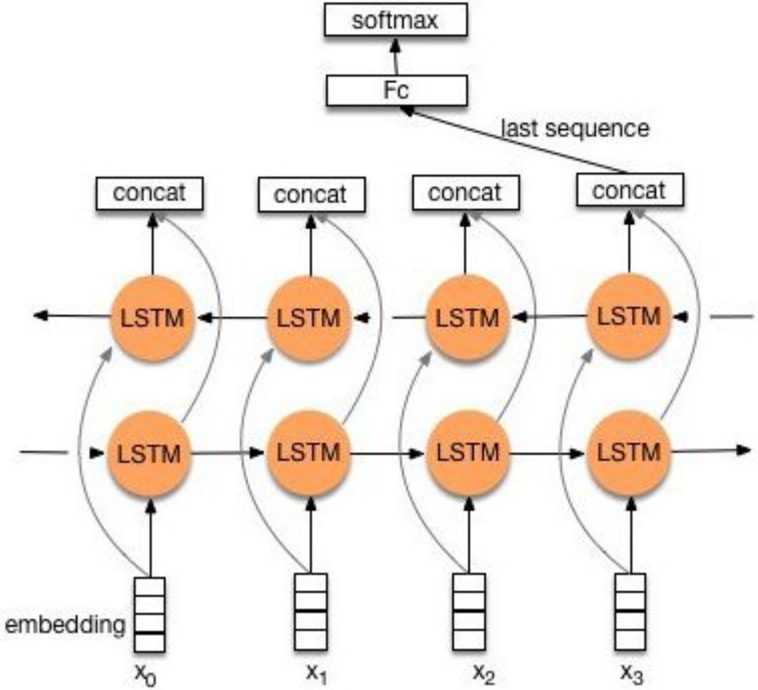
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

$$\widetilde{h_t} = tanh(W \cdot [r_t * h_{t-1}, x_t])$$

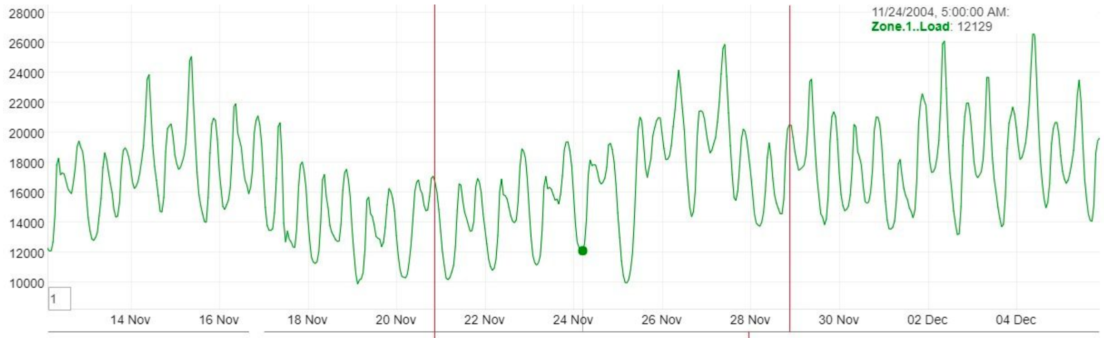
$$h_t = (1 - z_t) * h_{t-1} + z_t * \widetilde{h_t}$$

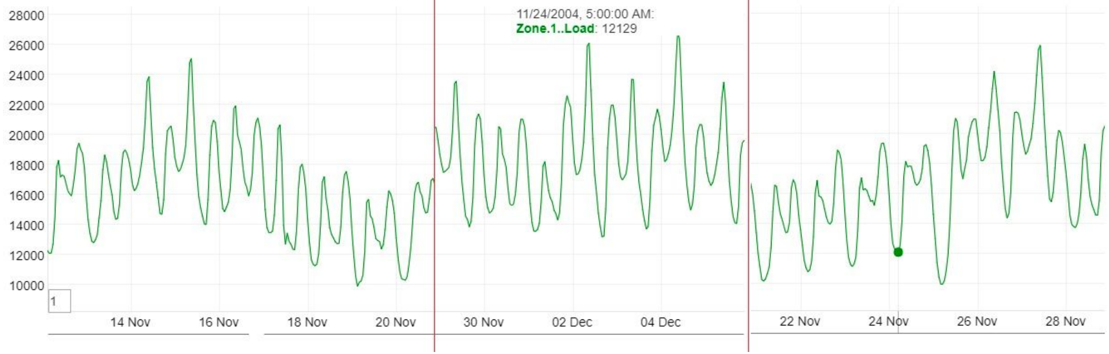
Bidirectional LSTM





Another approach with Bidirectional LSTM





Data Pre-Processing

$$X_{out} = \frac{(X - minX)}{(maxX - minX)}$$

```
# normalize data with min max normalization.
normalizer = MinMaxScaler(feature_range = (0, 1))
dataset = normalizer.fit_transform(data)

# Using 80% of data for training, 20% for validation.
TRAINING_PERCENT = 0.80
train_size = int(len(dataset) * TRAINING_PERCENT)
test_size = len(dataset) - train_size
train, test = dataset[0:train_size, :], dataset[train_size:len(dataset), :]
```

Keras Model Definition

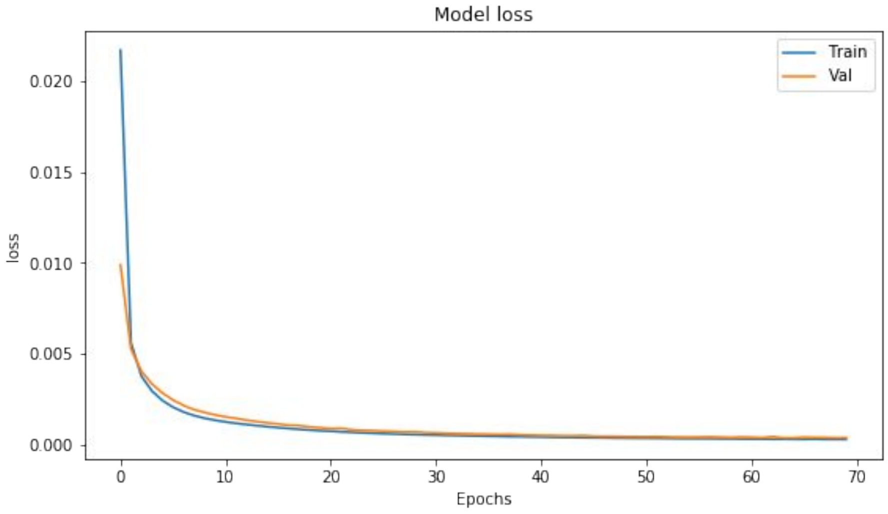
```
def create_model(train_X, train_Y, window_size = 1, nstep = 1):
    vanilla_rnn = Sequential()
    # add an LSTM layer for uni-variate data input
    vanilla_rnn.add(LSTM(20, input_shape = (1, window_size)))
    # output a single decision for the load forecast
    vanilla_rnn.add(Dense(nstep))
    vanilla_rnn.compile(loss = "mean_squared_error", optimizer = "adam", metrics = ['mse', '
        mae'])
    return(vanilla_rnn)
```

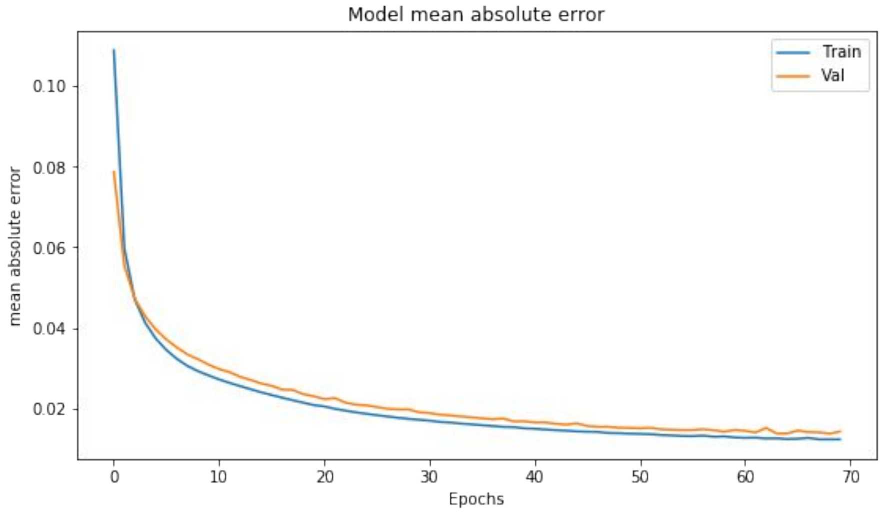
```
vanilla_rnn.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_17 (LSTM)               (None, 20)                5680      
_________________________________________________________________
dense_17 (Dense)             (None, 1)                 21        
=================================================================
Total params: 5,701
Trainable params: 5,701
Non-trainable params: 0
_________________________________________________________________
```

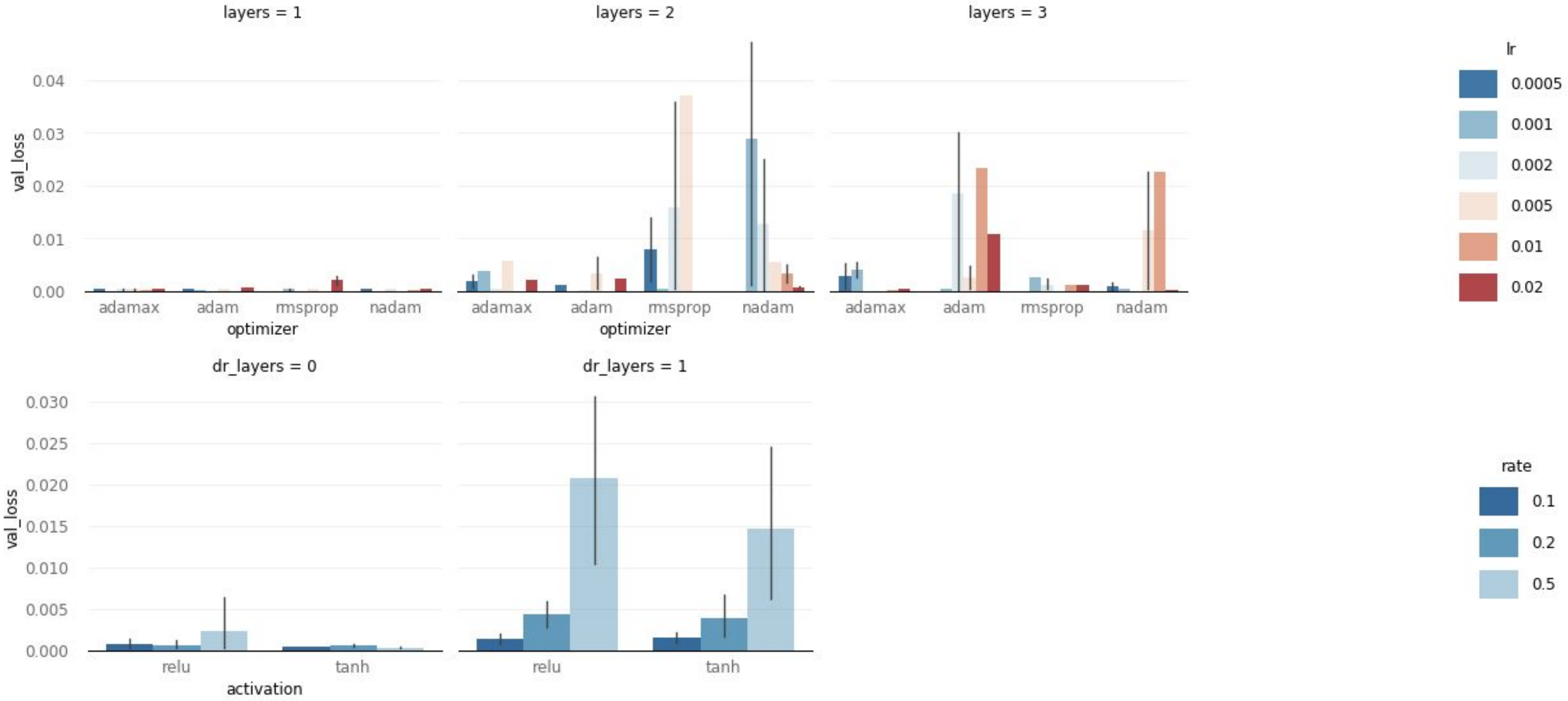
Keras Model Training

```
# train the model with early stopping and mini batches
es = callbacks.EarlyStopping(monitor='val_loss', patience=5, verbose=1)
history = vanilla_rnn.fit(train_X, train_Y, epochs = 100, batch_size = 32, verbose = 2,
    validation_split=0.15, callbacks=[es])
```





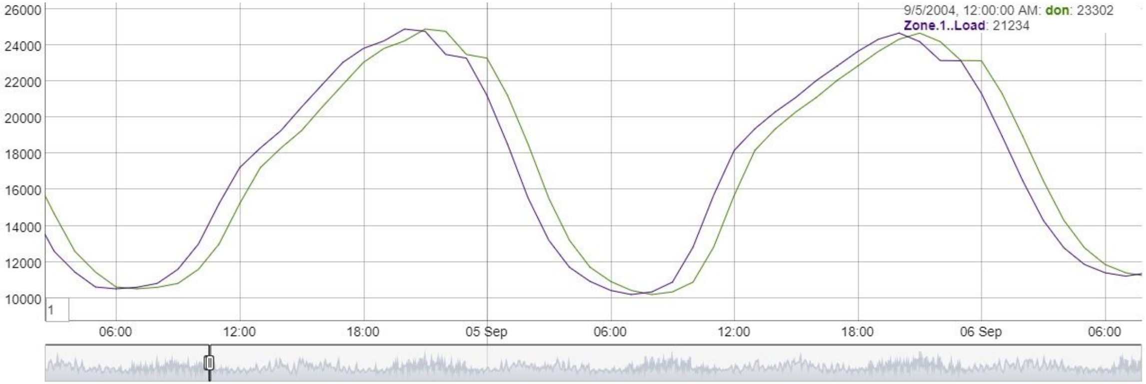
Hyper-Parameter Tuning



Naive Predictors

Table 3: Table showing test errors for naive predictors

	Loss Metric	
Architecture	**MSE**	**MAE**
Naive 1h ahead	1492.99	34.08
Naive 24h ahead	3284.90	48.00
Naive 168h ahead	5281.74	61.24



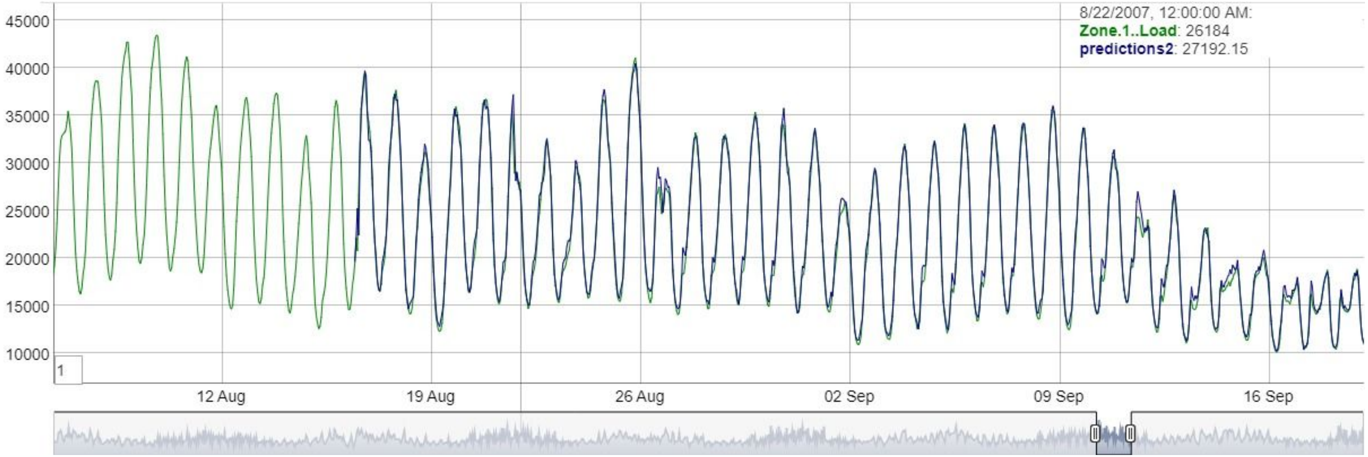
LSTM & GRU

Table 4: Table showing training and test errors for LSTM predictor

		Train Metric		Test Metric	
Architecture	**Window**	**MSE**	**MAE**	**MSE**	**MAE**
LSTM 1h	50	586.99	20.43	636.85	21.38
	100	655.76	22.04	702.24	22.81
LSTM 24h	50	2546.89	42.58	2737.39	44.11
	200	2429.30	41.68	2700.48	43.79
LSTM 168h	200	3530.33	51.61	4013.92	54.47
	500	3390.50	50.28	4157.22	55.25

Table 5: Table showing training and test errors for GRU predictor

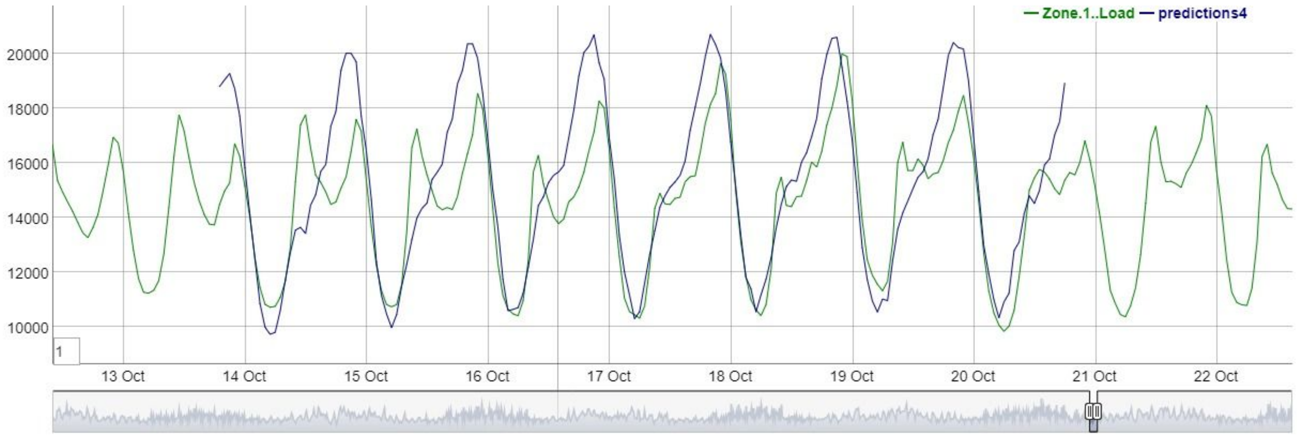
		Train Metric		Test Metric	
Architecture	**Window**	**MSE**	**MAE**	**MSE**	**MAE**
GRU 1h	50	699.08	22.89	753.37	23.83
	100	803.77	25.00	854.61	25.74
GRU 24h	50	2535.07	42.15	2733.19	43.73
	200	2404.36	41.31	2668.59	43.45
GRU 168h	200	3527.94	51.37	4005.74	54.17
	500	3496.78	51.39	4128.64	55.38



Bidirectional LSTM for Future Prediction

Table 6: Table showing training and test errors for bidirectional LSTM predictor

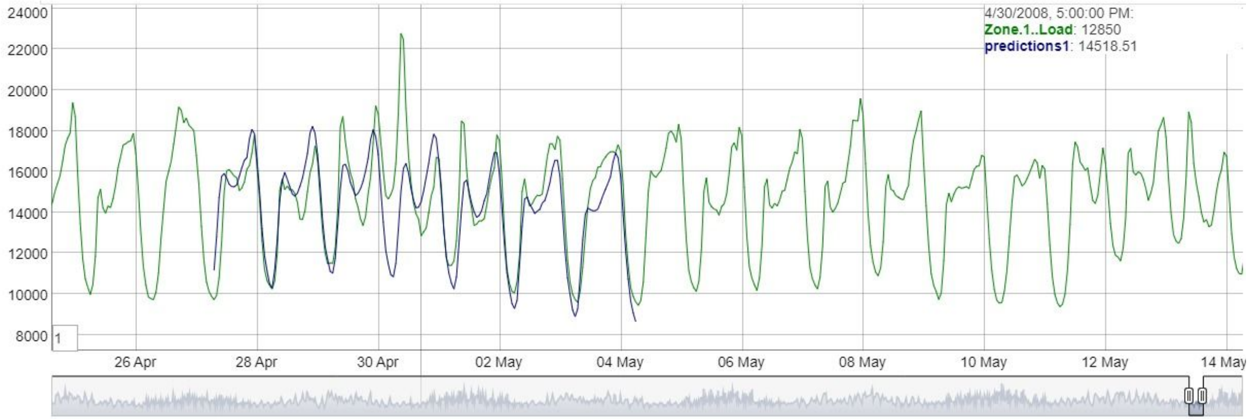
		Train Metric		Test Metric	
Architecture	**Window**	**MSE**	**MAE**	**MSE**	**MAE**
BIDIRECTIONAL-LSTM 1h	50	688.28	22.73	739.65	23.58
	100	616.46	21.52	663.49	22.34
BIDIRECTIONAL-LSTM 24h	50	2546.81	42.70	2737.35	44.19
	200	2426.07	41.53	2722.66	43.78
BIDIRECTIONAL-LSTM 168h	200	3670.45	51.67	4209.39	54.72
	500	3426.25	50.45	4100.67	54.66



Bidirectional LSTM for Gap Prediction

Table 7: Table showing training and test errors for bidirectional LSTM for gap prediction

		Train Metric		Test Metric	
Architecture	**Window**	**MSE**	**MAE**	**MSE**	**MAE**
BIDIRECTIONAL-LSTM 1h gap	50	457.51	19.72	484.20	20.22
	100	311.22	15.30	330.07	15.86
BIDIRECTIONAL-LSTM 1h gap	50	1781.94	35.12	1950.23	36.83
	200	1863.56	36.66	2054.10	38.57
BIDIRECTIONAL-LSTM 1h gap	200	3074.85	47.80	3735.12	52.07
	500	2972.87	46.97	3750.16	52.29



Multivariate LSTM with Temperature Data

Table 8: Table showing training and test errors for multi variable LSTM predictor (with mean temperature)

		Train Metric		**Test Metric**	
Architecture	**Window**	**MSE**	**MAE**	**MSE**	**MAE**
LSTM 1h	50	558.16	19.94	602.18	20.78
	100	625.36	21.64	676.99	22.54
LSTM 24h	50	2392.40	41.47	2606.74	43.16
	200	2369.30	41.44	2744.64	44.29
LSTM 168h	200	3474.56	50.72	4016.03	54.04
	500	3478.85	51.10	4286.81	56.18

Conclusion

GRU is the best performing model on our future prediction task.

Gap prediction can be tackled by using bidirectional LSTM.

Future work

- CNN LSTM
- Alternative implementation for gap prediction