Assignment 2 – Practical Deep Learning Workshop

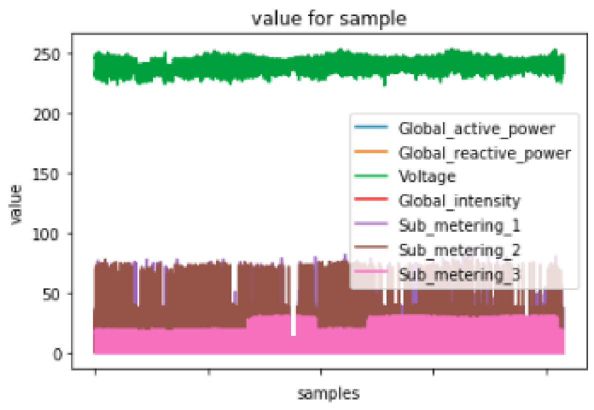
PART 1 – Time Series

The data we decided to use: 'Individual household electric power consumption Data Set'
https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption
description: the data consist measurements of electric power consumption in one household with a one minute sampling rate over a period of almost 4 years.
Size: there are 2,075,259 rows and 9 attributes.
Attribute information:

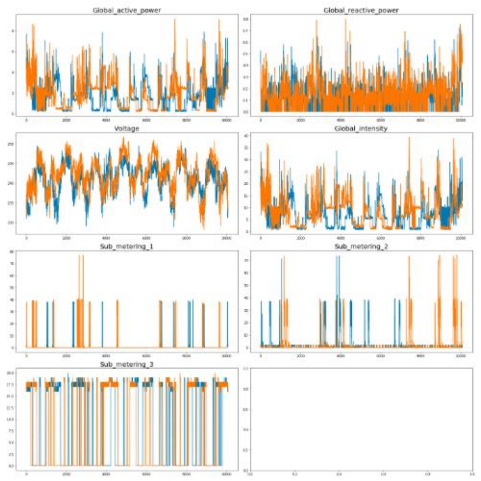
- *global_active_power*: The global minute-averaged active power consumed by the household (kilowatts).
- *global_reactive_power:* The global minute-averaged reactive power consumed by the household (kilowatts).
- *voltage*: minute-averaged voltage (in volt)
- *global_intensity*: household global minute-averaged current intensity (in ampere)
- *sub_metering_1:* Active energy for kitchen (watt-hours of active energy).
- *sub_metering_2:* Active energy for laundry (watt-hours of active energy).
- *sub_metering_3:* Active energy for climate control systems (watt-hours of active energy).
- *Date*: dd/mm/yyyy
- *Time:* hh:mm:ss

Remarks: 1) 1.25% of the data contain missing values. 2) There is option for feature engineering by adding new feature that calculate the active energy that used in the house except sub_metering_1,2,3.

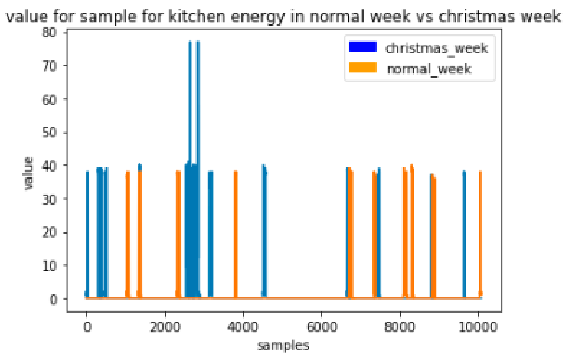
Interesting plots:



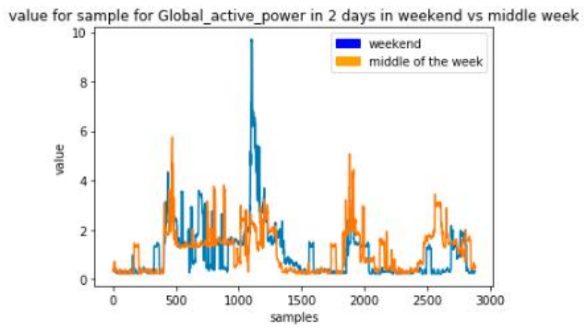
the goal of this plot is to visualize the distribution of values for each feature, and to see more easily that some features have high values and other low. we can notice that not all the features in the same value range. For example voltage is very high in compare to the others.



the goal of this plot is to recognize patterns in the data, each color describe different week, we can see that in some of the days for some features the patterns are almost identical while in others not, for example the lower right plot describe the laudry room and we can see that patterns for each week is very different probably because using washing machine happened in defferent days during different weeks.



In holidays and special dates the electricity consumption can be higher. In this plot we can see that the consumption in the kitchen doubles itself before Christmas evening.

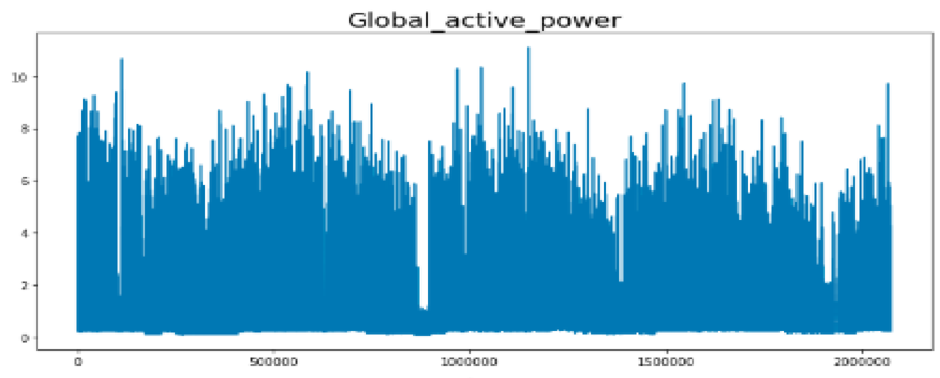


Global consumption during the weekend may be higher than in the middle of the week.

to conclude: there are many similar patterns in the data but also the data may be varied in some cases, we see the following dots as the main challenging obstacles in the data:

- we can see that there are some anomalous but most of them probably because of the Nan values.
- we also have to remember that this data set measure one household so if the family is in holiday for day or more probably high decrease is reasonable.
- we can also recognize patterns that repeat themselves, this is because probably the family use the same amount of electricity in everyday life (for the same season).
- voltage feature has different value range. the rest are almost the same range (scale by watt).
- there is a difference in the electricity consumption during weekend and middle of the week.
- There is a difference in the electricity consumption between normal days and holidays.
- Use of appliances that consume electricity, such as a washing machine is usually fixed in quantity but can change in time

Our goal: we will rearrange the data into hours and our goal will be to predict the next hour consumption, given X hours before.



By getting features of X amount of last hours we will have to predict the 'Global_active_power' of X+1.

Task type: regression

Part 2 – Time-Series

Validation strategy: We decided to take the data and split it to train and validation. We used this split because we have enough data for both train and test sets, and we didn't use any ahuffle because it is time-series data and we didn't want to lose this feature. We used 20% of the train data for validation set.

Using of each data se (train,validation and test)t:

- each ml train on the train set.
- for each ml there are different parameters we can tune, in order to evaluate the best model among them, we will use the validation set. (so there is dependency between the model and the parameter we try to tune)
- after we choose the best tuning parameter, we want to evaluate the model with data that there is not any dependency between the model and the data, for this purpose we will use the test set.

Normalize: we used normalized with minmax method in order that all the features will be on the same range.

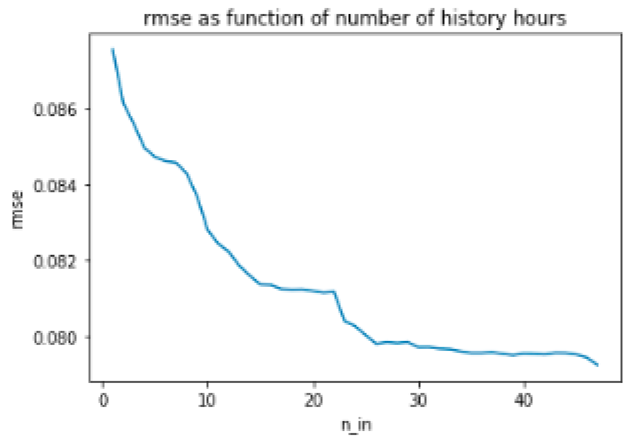
Evaluation method: RMSE

Naïve Baseline: the prediction is the mean of all the X history 'Global_active_power'.

Rmse = 0.2

Classical machine learning: we tried 2 different linear models: Linear regression and Lasolars, we got better result with linear regression so we will elaborate on this model:

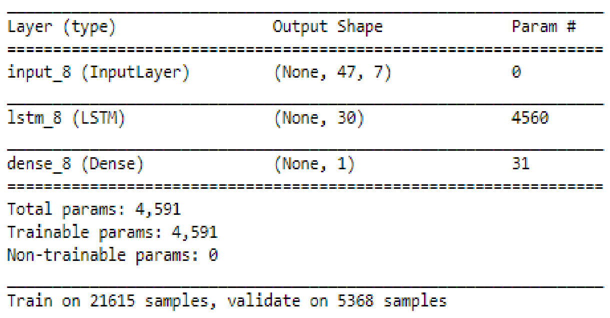
We ran this model with different amount of history hours:



We have got the best result with history of 47 hours, with RMSE = 0.1197 .

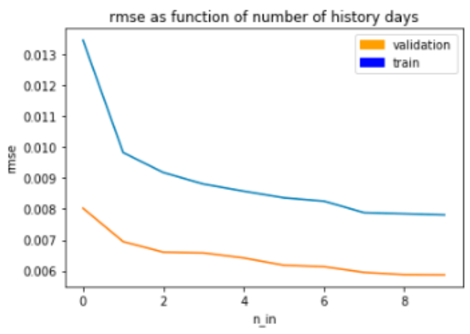
This is almost the same as the naïve baseline.

Fit nn:



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_8 (InputLayer)         (None, 47, 7)             0
_________________________________________________________________
lstm_8 (LSTM)                (None, 30)                4560
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 31
=================================================================
Total params: 4,591
Trainable params: 4,591
Non-trainable params: 0
_________________________________________________________________
Train on 21615 samples, validate on 5368 samples
```

We used simple nn with lstm one hidden layer with 30 units. History hours of 47.

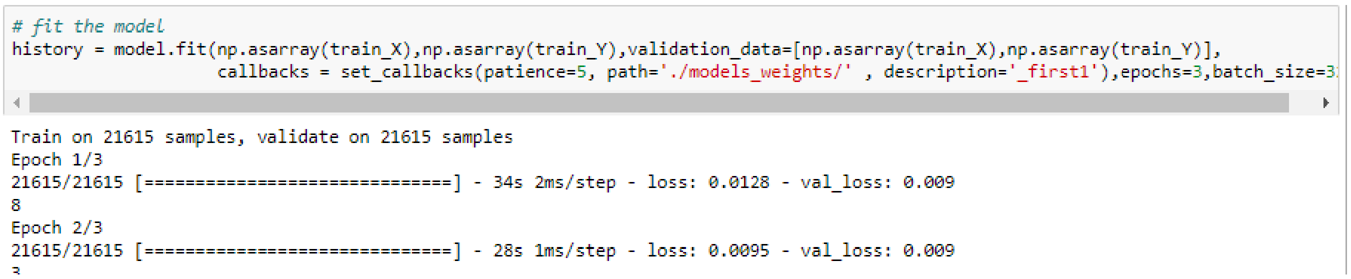


important facts about the plot

- the model has 4,591 parameters, and 21,623 samples to train on.
- we can see that as the epoch advanced the train loss decrease, the same happend in the validation set (except epoch 3).
- the loss of the validation set is lower than the train set since the first epoch, and as the number of epoch increase the train set loss decrease, and the validation set decrease as well.
- we have got an interesting case , which the model fit better to the validation than to the training set, it can happen sometimes and it is not planned, but because we can notice that while the train set fit better, the validation loss decrease and because of the explanation below this case doesn't bother us.

Explanation about the phenomenon that the train loss is higher than the validation loss

while we trained this nn we were very surprising to see the the train loss is higher than the validation loss during all the training process.
first thing we did is using 'Three fingers rule' in order to guess why this happens, we thought about the following reasons:
our split was not good enough - because probably the train set has more 'difficult' cases that the model has to learn, and the validation set doesn't consist them, so the validation consist only the easy cases that the model learn very easy, and hence the loss rate for the validation is lower.
what we did?
we thought maybe there is really difficult cases which can result from transition season, holidays, and many other reasons, and because the validation set cover only few months probably some of this cases doesn't appear there, so we tried to split the train and validation in a different way so the validation will cover one year. the reason for this split is that in the exploration we saw that over the years the use in electricity was similar. the result of this try, was the same, and the train loss was still higher than the validation loss during the train.
because the train set is bigger than the validation set the loss is higher - we calculate the loss by mse which means MEAN square error, so because it mean the square error this assumption is not logical.
the loss of the train set every epoch is calculated before the gradients have been applied while the validation loss calculated after the gradients have been applied
what we did?
we decided to try to train the model and use our train set also for validation, if we will see that the validation loss is less than the train loss we can conclude that this assumption is right.



```
# fit the model
history = model.fit(np.asarray(train_X),np.asarray(train_Y),validation_data=[np.asarray(train_X),np.asarray(train_Y)],
                    callbacks = set_callbacks(patience=5, path='./models_weights/' , description='_first1'),epochs=3,batch_size=3
```

```
Train on 21615 samples, validate on 21615 samples
Epoch 1/3
21615/21615 [==============================] - 34s 2ms/step - loss: 0.0128 - val_loss: 0.009
8
Epoch 2/3
21615/21615 [==============================] - 28s 1ms/step - loss: 0.0095 - val_loss: 0.009
3
```

Understand the errors: for each model we considered prediction with rmse that higher than rmse of the all test set + 0.2 as 'higher rmse' and plot summary tables, here we will just write our conclusion from these tables:

- total errors: 60
- most of them occoured during the evening time. (41/60 => ~68%)
- a large number of the mistakes occoured during the days saturday and sunday. (~38%)

what can we infer from this ?
Unfortunately, we do not know from which house the data is taken and how many people are living at home, and it make it difficult for us to understand the electricity consumption habits of the family members. by the facts we mentioned, most of the errors are during the evening, and in the weekend, so probably we can infer from it that during the day and in the middle of the week most of the time the family members at school, job and etc, so the consumption patterns in these hours are easy to learn. but at the evening and weekend time, they spend more time at home, use electricity often the the consumption pattern is more complex because of these reasons.

RMSE = 0.0797

so... how can we improve our nn model?

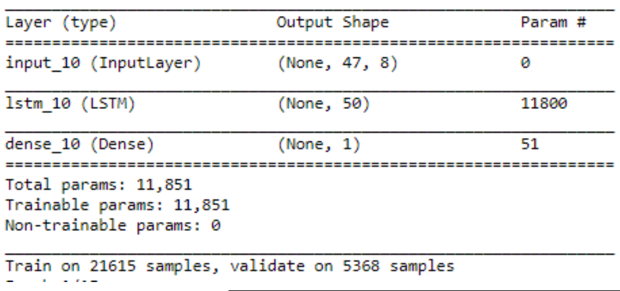
Three Finger Rule (at least three...)

- we can try to add more units to the LSTM layer or add new layers to increase the number of parameters, in order to fit the train data even more.
- we can make our nn model deeper and see if it is helping (as we saw in lectures the deeper the layer, the more complex patterns it learns).
- we can try to change n_in value, possibly with different value the model can generelize better.
- we can try feature engineering by adding new column for the 'active energy consumed every minute (in watt hour) in the household by electrical equipment not measured in sub-meterings 1, 2 and 3.' as new feature as we saw in the exploration.
- we can try to use CNN layers in order to try to recognize the structure of data.
- we can also try to change the batch_size. (we read some articles that using mini-batch can increase the generalization)
- we can try to use feature extraction with cnn (as we saw in lecture, this will be good idea in case cnn will give us good result, so it recognize patterns in the data, and we can use this patterns as features)

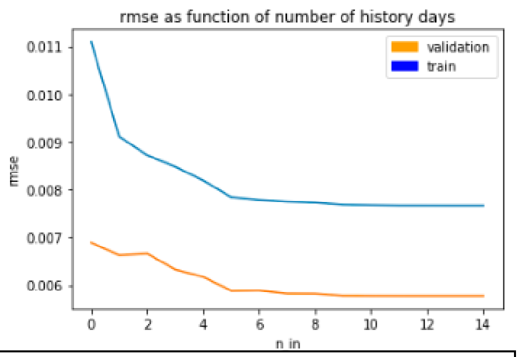
by these steps we can maybe fit the data better and recognize complex patterns in the data even during the evening and weekend time.

We used: nn with feature engineering, find best amount of history hours, deep nn with Conv1D

nn with feature engineering – RMSE = 0.0763



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_10 (InputLayer)        (None, 47, 8)             0         
_________________________________________________________________
lstm_10 (LSTM)               (None, 50)                11800     
_________________________________________________________________
dense_10 (Dense)             (None, 1)                 51        
=================================================================
Total params: 11,851
Trainable params: 11,851
Non-trainable params: 0
_________________________________________________________________
Train on 21615 samples, validate on 5368 samples
```

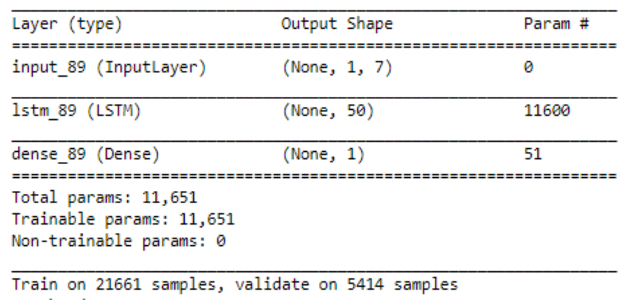


We can see that from epoch ~10 the model doesn't success to improve the train loss result anymore. It's also important to notice also that the loss is very low. Maybe by adding even more parameters or using deeper nn we can fit beven better the train set.

Understand the errors –

- we have 54 errors above the threshold.
- 32 of them occoured during the day (pm), this is ~63%.
- 22 of them occoured during the weekend (saturday,sunday), ~40%

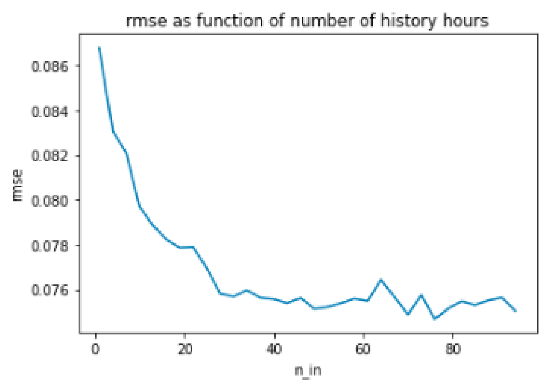
Best amount of history hours – maybe we can get better result by finding the best amount of history hours



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_89 (InputLayer)        (None, 1, 7)              0         
_________________________________________________________________
lstm_89 (LSTM)               (None, 50)                11600     
_________________________________________________________________
dense_89 (Dense)             (None, 1)                 51        
=================================================================
Total params: 11,651
Trainable params: 11,651
Non-trainable params: 0
_________________________________________________________________
Train on 21661 samples, validate on 5414 samples
```

RMSE = 0.0751

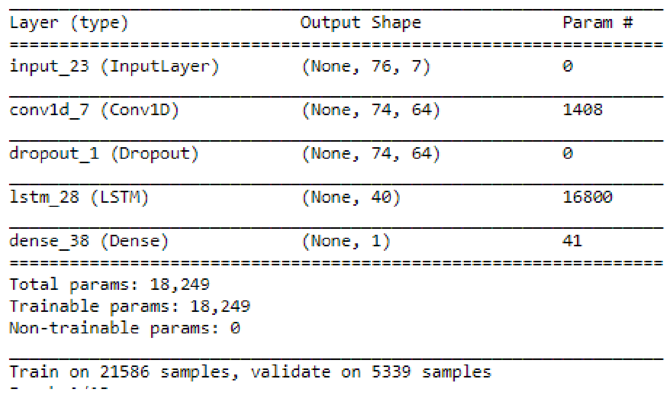
```
*** BEST RESULT WITH 76 HISTORY HOURS WITH RMSE OF: 0.0746921123307941 ***
```



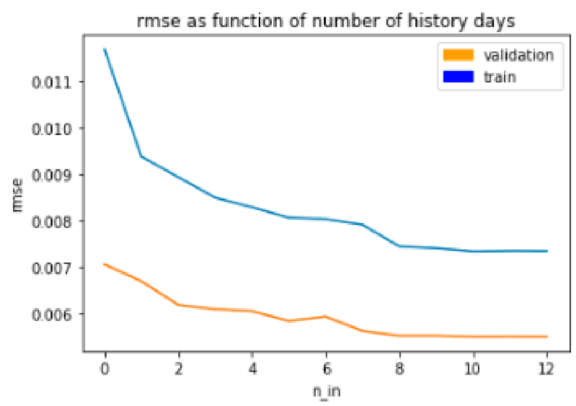
Understand the errors -

- there are only 49 errors.
- 70% at pm.
- 26% during weekend.
  in compare to the last model there are only 49 errors with 0.2 above the rmse (54 in last model), most of the errors are still at pm, but the number of errors during the weekend dropped from 40% to 26%. so by this model we improved the weekend prediction.

Deep nn with Conv1D :



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_23 (InputLayer)        (None, 76, 7)             0         
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 74, 64)            1408      
_________________________________________________________________
dropout_1 (Dropout)          (None, 74, 64)            0         
_________________________________________________________________
lstm_28 (LSTM)               (None, 40)                16800     
_________________________________________________________________
dense_38 (Dense)             (None, 1)                 41        
=================================================================
Total params: 18,249
Trainable params: 18,249
Non-trainable params: 0
_________________________________________________________________
Train on 21586 samples, validate on 5339 samples
```



In this model we tried to use conv1d in order to recognize better patterns in the data because there are many parameters we decided to use dropout to keep it generalize. as we can see in this plot until epoch 8 as long as the train loss decrease the validation loss decrease as well, from this epoch the loss is constant and the model doesn't success to fit the train better.

RMSE – 0.076

Understand the errors:

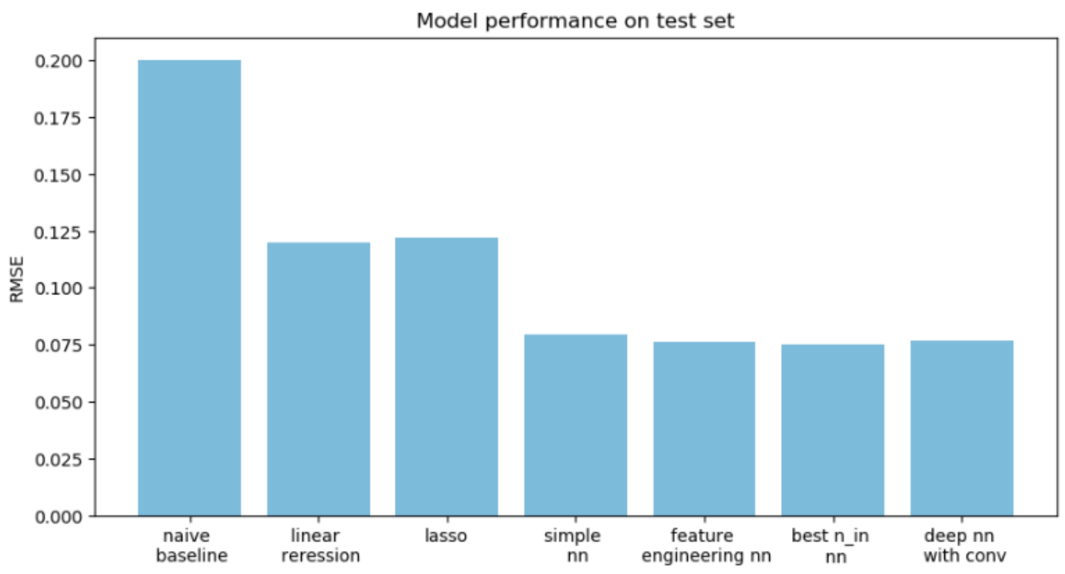
total number of errors: 51
at pm 64%
at weekend 49%

From all the nn this model had the worst result, in both apects, rmse is higher and the model doesn't have any advantage during the weekend or pm in compare to previous models.

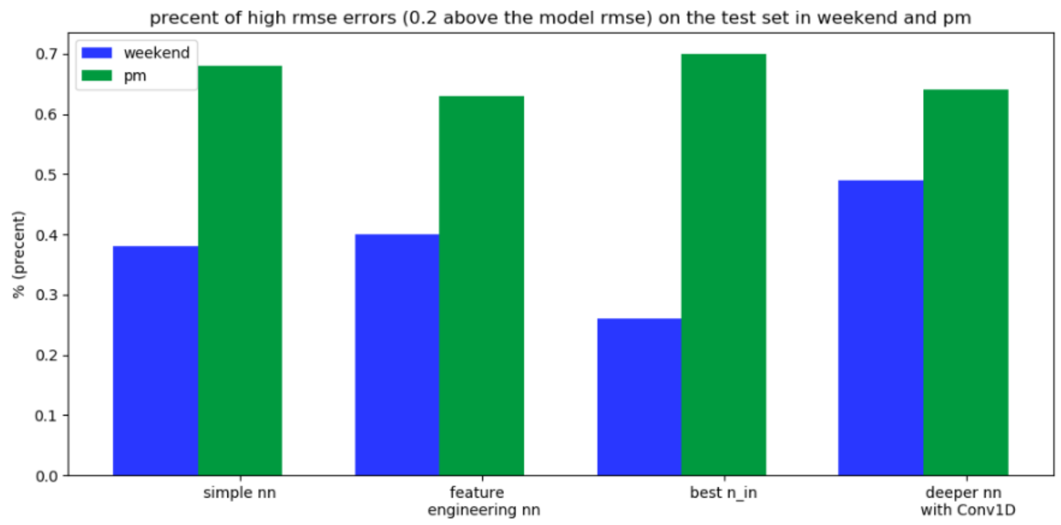
we also tried: change batch size, use deeper architecture, deeper lstm (more than one)

SUMMARY

The following plots presents the performance and advantages of each model:



As we can see the best models were with nn



In reality we can use different models in different times so for example during the weekend we can use the third one for better results while in general the second one will give us good results.

What can we do more?

- We can try feature extraction as we saw in the last assignment.
- As we can see none of the models had really improvement during pm, so we can investigate more the data in this hours, maybe we can understand more accurate which hours are the problematic, and use different feature to improve the results. (for example we can use features of the same hours the week ago of the day before)