CSC 578 FINAL PROJECT

Katheryn Hrabik

CSC 578 – 710 (Online)

Kaggle Username: khrabik1 *(Display name: Katheryn Hrabik)*

Rank: 9th of 27 (at close of competition)

Video Link: http://youtu.be/pBuYhShFsLI?hd=1

Summary of Competition Model (Also Best/Most Interesting)

Model Architecture:

LSTM Layer:

- Nodes: 40
- Epochs: 30
- Optimizer: Adam
- Loss Function: Mean Absolute Error
- Batch Size: 75
- Dropout: None

Dense Layer(s):

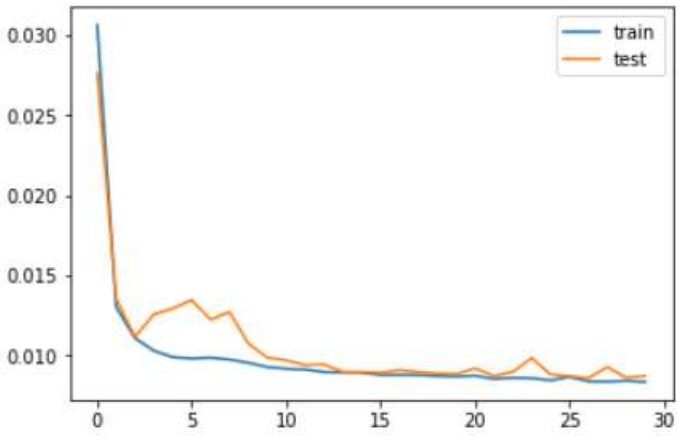
- Size: 1 dense layer, size 1
- Activation Function: default

Additional Changes:

- Reducing Noise: removed nose in WV(m/s) and max.wv(m/s)
- Removing Features: Removed wd(deg)

Summary:

Although I tried many different combinations, I personally got the best result with what could be considered a "base model". I did multiple iterations, and found that an LSTM of 40 gave me the best error value for the competition. I also tried the RMSprop optimizer as it is generally one of the preferred optimizers for recurrent networks, but found that Adam did slightly better for me. While exploring the data, I did find some noisy values (-9999) in a column that contained very low values (mostly around 1 or 2), so I replaced the noisy values with 0. I also saw that the distribution of the wd(deg) feature looked like it didn't contain anything of much interest, and removed that as well. Interestingly, removing the noise and doing extra pre-processing of the data did not give much additional value to the model. **Best Score: .48004**



Description of Hyper-Parameter Tuning

LSTM Nodes: I systematically tried a range of values between 1 and 512, but found that I obtained the best results with 40. In my research I did see that the LSTM nodes can be estimated using the following "rule of thumb":

$$N_h = \frac{N_s}{(\alpha * (N_i + N_o))}$$

With α being a multiplier of your choosing, generally between 2 and 10. This led me to choose 40 as my value for the LSTM nodes, which did end up working well for me.

Conv1D: I tried a number of Conv1D models. I tried different filter sizes, and different activation functions including 'relu' and 'tanh'. However, adding the convolutional layer increased error and I ended up removing it.

Dropout: One interesting observation I made with this dataset is that, at least for my model, adding both dropout and recurrent dropout tended to increase error significantly, which was surprising. I tried dropout rates of .125 through .5 for dropout and recurrent dropout, as well as various combinations between the two, but could not make improvement on learning. Adding a separate dropout layer seemed to have a similar, though more pronounced, effect

Dense Layers: Again, I tried many different sizes and combinations, but found the best result with one single dense layer of size 1.

Epochs: I tried epochs from 20 to 200, settling in on 30 as my best fit.

Learning Rate: As I increased the number of epochs, I tried different values for eta within the Adam optimizer. As I increased epochs, I decreased learning rate, but again I found the default values to work best for me for this model.

Optimizers: As mentioned above in the summary, in addition to Adam I also tried RMSprop, as it's also highly regarded for RNN construction. Adam ended up being slightly better for my setup, so I ended up sticking with Adam.

One Additional Model

I wanted to add an additional model here that I found fun to make. This one was much different from my final submission, and although it was more complex, the performance was not quite as good as my submission model. I chose to add a 1 dimensional convolutional layer at the beginning, with a filter size of 2 and an activation function of relu. Additional details can be found below.

Model Architecture:

Dimensional Convolutional Layer:

- Filter size: 2
- Kernel: 2
- Activation Function = 'relu'

LSTM Layer 1:

- Nodes: 40
- Recurrent Dropout: 0.2

LSTM Layer 2:

- Nodes: 20
- Dropout (regular): 0.2

Dense Layer(s):

- Size: 2 dense layers, size of 10 and size of 1
- Activation Function: default

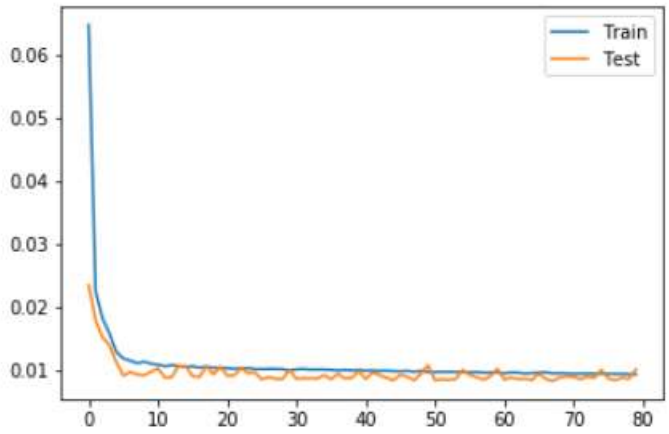
Additional Changes:

- Reducing Noise: removed nose in WV(m/s) and max.wv(m/s)
- Removing Features: Removed wd(deg)
- Epochs: 80

Summary:

This was a model that I had a lot of fun with, even though it did not perform as well as my base model solution. Interestingly, adding the additional layers with dropout seemed to have a negative impact . Over time, the training error decreased slowly as expected. However, the validation error vacillated quite a bit. The dropout seemed to be the issue: perhaps a dropout

rate of .02 was too high for this data set? It was not my most efficient model, but was fun to build nonetheless. **Best Score with *this* model: .59**



Conclusions and Reflection

In conclusion, I was surprised that altering the base model did not seem to make much of a difference. I was also surprised that adding dropout to reduce overfitting didn't seem to help for this data set. After trying so many different combinations, the best model was the most simple. To me, this was a lesson that sometimes "Less is more". Aside from some issues I had with reversing the scaling from the MinMaxScaler, I had a lot of fun with this project! I feel like I really learned a lot from this project because I had to choose everything myself from pre-processing to tuning. I feel like this was good real-world experience.