Kari Palmier
CSC 578 Section 710
Final Project
Nov. 12, 2018

Kaggle ID = **KPalmier**
Kaggle Competition 9 in Public Leaderboard, 7 in Private Leaderboard

Link to video (also included in class shared document): https://photos.app.goo.gl/WaETJdbW377Nfy8x8

Modeling Process

I started out by creating a simple single layer LSTM model with 50 nodes, followed by one output dense layer with 1 node (one because only one temperature output is desired from the model per input set). I used this model to integrate my code for model input creation and model analysis. This model performed fairly well, with a prediction MAE (mean absolute error) loss of approximately 0.4755. Since this model did well, I decided to use it as a base for the rest of the models. Next, I generated 33 different models, the first of which is the base model and the rest with slightly different parameters. I changed the number of LSTM layers, the number of LSTM nodes, the number of dense layers, the number of dense layer nodes, the batch size, the number of epochs, the compiler optimizer, use of LSTM input dropouts, use of LSTM recurrent dropouts, dense layer dropouts, use of LSTM kernel regularization (L1 an L2 with 2 different lambdas each), use of LSTM recurrent regularization (also L1 and L2 with 2 different lambdas each), LSTM activation type, dense layer activation type, and combinations of multiple LSTM layers and dense layers. I changed only one parameter at a time in order to determine the effect of each parameter on the model. Out of these 33 models, I chose the 13 best (models with less than 0.51 prediction MAE loss). I then ran these 13 models two more times to examine the repeatability of these models. During the entire modeling processe, I saved off data frames with all of the loss results (model numbers, MAE values, scores, training minimum values, testing values corresponding to the training minimum values, last training values, and last testing values). All models were generated with the same training x and y data and were validated with the same testing x and y data. The training data consisted of all data points before 1/1/2015 00:00:00 (so up until 12/31/2014 11:00:00). The testing data consisted of all data points including and after 1/1/2015 00:00:00. There was no overlap in the training and testing data used for modeling. I specifically did not include overlap because I did not want the modeling validation losses to be influences by points that were used in both the training and testing sets (using any of the same points for training and testing can make the model performance look better than it is). I created a third data set which was used to calculate the temperature predictions required for the Kaggle competition. This third data set consisted of all points before and including 12/31/14 00:00:00. Once each model was made, Kaggle predictions were generated and the MAE of those predictions versus the true temperature data was calculated, then used for analysis. Note that I inversed the normalization of the prediction values that were generated from the predict function before calculating the MAE since the predictions were normalized (predictions were normalized because model inputs were normalized, final prediction MAE must be calculated on non-normalized predictions). I also looked at the difference between the minimum training loss and the corresponding testing loss as well as the difference between the last training loss and last testing loss to see if overfitting was an issue (I also used a plot of training and testing losses to evaluate overfitting). Once I compiled all of the results from the 33 original models and the 26 repeated models (13 models run twice), I evaluated which models had the lowest prediction MAE loss, which had the lowest standard deviation (were most repeatable), and which had low overfitting. I chose my final model to be the model with low MAE prediction losses for all 3 runs (was not the lowest for any run, but was very close to the low value for each), had the lowest standard deviation, and did not exhibit overfitting. This model ended up being 2 LSTM layers with 50 nodes each, followed by one output dense layer with one node. This was the third model I ran during the full 33 model execution.

<u>Results Analysis</u>

The final model chosen was the third model. It consisted of 2 LSTM layers with 50 nodes each and 1 output dense layer with 1 node. I chose this model because it had the lowest mean prediction MAE over the 3 runs performed, the lowest MAE standard deviation, and the lowest MAE range (maximum – minimum MAE values over all runs). It also did not exhibit any overfitting, as seen in the training and testing loss plot over epoch (included in this document in the at the end of this section) and also by the very low differences between the last training loss and last testing loss as well as the minimum training loss and its corresponding testing loss. Note that in this document, when any prediction MAE losses are referenced, these are referencing the MAE losses of the predictions from the Kaggle training set. There were several models that had loss values similar to the final model (between 0.47 and 0.51), which made the choice in final model difficult. I ended up basing my choice on the repeatability results. The model chosen had the best repeatability across the 3 runs (had the most stable results across runs), as well as having very low MAE loss for each run (even though it did not the lowest losses). The first table below in this section contains the MAE and overfitting results for all of the models run. The second and third tables contain parameter and network structure information for each model.

I found that using an ReLU activation on the output dense layer caused a very high prediction MAE loss value (model 9 in the results table). This is most likely due to the ReLU activation limiting the output to be positive. In hindsight, I should have chosen the tanh activation instead of ReLU since it allowed for negative values. Performing 20% input dropouts in the LSTM layer caused high prediction MAE loss and overfitting (see training and testing loss plot at the end of this section). The model loss plot shows the modeling testing losses increased over epoch number while the training modeling losses decreased. Also, the training and testing loss differences were much higher than all other models (0.09 versus less than 0.001 for the rest). LSTM L1 kernel regularization with 0.1 lambda also displayed high prediction loss. The rest of the kernel regularizations performed (L2 with 0.001 lambda, L2 with 0.1 lambda, and L1 with 0.001 lambda, models 14 through 16) had slightly higher prediction MAE losses (approximately 0.55 to 0.58). This means shows that kernel regularization did not positively affect model results and is not necessary. The prediction MAE loss for LSTM L1 and L2 recurrent regularization (models 18 through 21) had slightly lower loss than those of kernel regularization, but were higher than most of the repeated models without regularization. This shows that recurrent regularization was not beneficial to modeling and predictions. The model with standard gradient descent (SGD) optimizer exhibited higher prediction MAE loss than most models, which means that the adam optimizer is the better optimizer to use with the data. Using a low number of nodes in the LSTM layer (10 versus 50 in model 4) also lead to slightly increased prediction MAE losses. A low batch size (5 versus 10 in model 10) also resulted in slightly increased prediction MAE losses. Increasing the number of nodes from 50 to 150 in the LSTM layer (model 2), increasing the batch size from 10 to 100 (model 11), and increasing the number of epochs from 20 to 100 (model 30) did not have a significant effect on the prediction MAE losses. Adding additional dense layers, dropout layers after additional dense layers, and adding additional LSTM layers all resulted in models with very similar performance to the final model. The difference is that these prediction MAE losses of these models had more variation than that of the final model.

I found through experimentation that the MAE loss for a given model had variation from run to run. This is most likely due to the initial model weights being randomly chosen during each model run. The variation of MAE losses for models across different runs were seen to be between approximately 0.01 and 0.07 for most of the 13 models that were re-run. While integrating my code, I ran the first model several times more than are recorded in this document (I only included results after I had the final analysis code set). During these runs, I saw the prediction MAE loss of the first model vary as high as 0.6

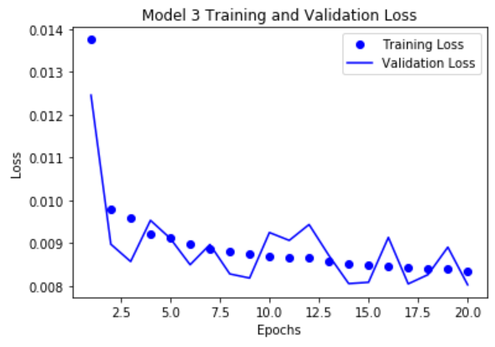
on some runs. This high of loss was not common though (most were between 0.45 an 0.51). This does mean that it is possible that the losses that occurred during the recorded runs in the table are not the most extreme cases. In other words, it is possible for a model to have slightly higher or lower loss than was seen during my experimentation. This means that since there were several models with losses between 0.47 and 0.54 and because a variation of up to 0.07 was not uncommon, the results for all of these models could be interpreted as being essentially the same. If I were to have more time, I would investigate different types of Keras LSTM weight initializers to see if any provided more stable modeling results. There are a few initializers that use constant values (not random). These initializers would provide common initial weights for all models which would make models more repeatable (would have less prediction MAE loss variation). It would also allow for the results of different models to be compared to each other more accurately.

Model Results Summary											
Model Number	**Kaggle MAE Run 1**	**Kaggle MAE Run 2**	**Kaggle MAE Run 3**	**Kaggle Run Min MAE**	**Kaggle Run Max MAE**	**Kaggle Run Mean MAE**	**Kaggle Run MAE Standard Deviation**	**Kaggle Run MAE Range (Max - Min)**	**Run 1 Last Training and Last Validation Loss Difference**	**Run 2 Last Training and Last Validation Loss Difference**	**Run 3 Last Training and Last Validation Loss Difference**
1	0.5066281	0.4857692	0.4985891	0.4857692	0.5066281	0.4969955	0.0105203	0.020859	6.54E-05	0.000223499	1.16E-05
2	0.4924663	0.4906202	0.5390017	0.4906202	0.5390017	0.5073627	0.0274157	0.048381	0.000145599	0.000112575	0.000680816
3	0.476677	0.4795039	0.4775549	0.476677	0.4795039	0.4779119	0.0014469	0.002827	0.000413599	0.000331926	0.000449219
4	0.5589072								0.000806038		
5	0.4858919	0.4779715	0.4907483	0.4779715	0.4907483	0.4848706	0.0064494	0.012777	0.000305354	0.000377405	0.000112898
6	0.4856954	0.4992349	0.5211398	0.4856954	0.5211398	0.5020234	0.017886	0.035444	0.000413515	0.00019912	0.000147122
7	0.5383673								0.004602872		
8	0.5472194								0.000759147		
9	3.7108578								5.07E-05		
10	0.6016788								0.001639471		
11	0.4898162	0.5655785	0.5425674	0.4898162	0.5655785	0.532654	0.0388418	0.075762	0.000165014	0.001215488	0.000668319
12	6.245388								0.093863061		
13	0.4965485	0.5378301	0.4900929	0.4900929	0.5378301	0.5081572	0.0258994	0.047737	0.000544238	0.000241226	0.000578397
14	0.5511582								0.000337878		
15	0.588183								0.000783053		
16	0.554052								0.0002584		
17	0.9063122								0.001162695		
18	0.5073069	0.5376025	0.5194372	0.5073069	0.5376025	0.5214489	0.0152477	0.030296	0.00051505	0.000171845	0.00031624
19	0.5534666								0.00048753		
20	0.5258702								0.000559186		
21	0.5780157								0.000926358		
22	0.4768676	0.523303	0.4737075	0.4737075	0.523303	0.4912927	0.0277667	0.049596	0.000663179	0.000218697	0.000521862
23	0.5361216								0.000364931		
24	0.5407775								0.000442399		
25	0.5309176								0.00311849		
26	0.4961271	0.4926743	0.4894262	0.4894262	0.4961271	0.4927425	0.0033509	0.006701	8.50E-05	0.000164046	0.000164467
27	0.486758	0.4832546	0.4859295	0.4832546	0.486758	0.485314	0.001831	0.003503	0.000211525	0.000306733	0.000246626
28	0.4757567	0.4885632	0.4930471	0.4757567	0.4930471	0.485789	0.0089728	0.01729	0.000502392	0.000201195	9.27E-05
29	1.0081046								0.006372659		
30	0.5564488								0.00149406		
31	0.5314566								0.000335831		
32	0.5258389								0.000337628		
33	0.4878216	0.5571622	0.5594167	0.4878216	0.5594167	0.5348002	0.0407002	0.071595	0.000529293	0.000601665	0.000764297

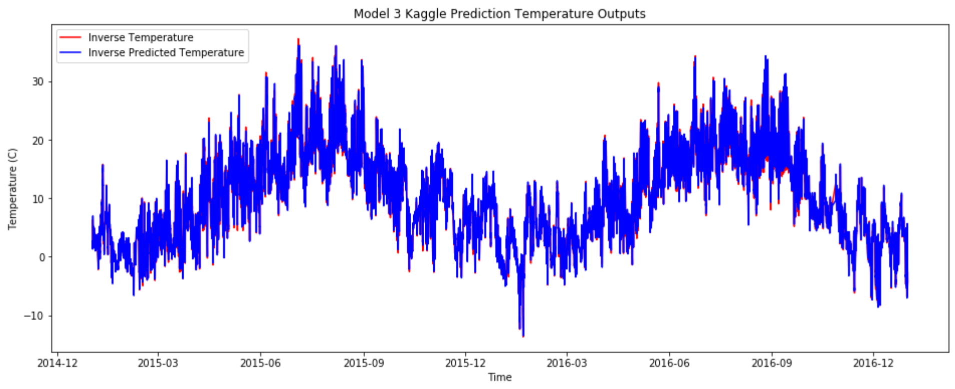
Model LSTM Parameter Descriptions									
Model Number	Number LSTM Layers	Number LSTM Nodes Per Layer	LSTM Activation	LSTM Input Dropout Percentage	LSTM Recurrent State Dropout Percentage	L2 Kernel Regularizer Lambda	L1 Kernel Regularizer Lambda	L2 Recurrent Regularizer Lambda	L1 Recurrent Regularizer Lambda
1	1	50	None (Linear)	0	0	None	None	None	None
2	1	150	None (Linear)	0	0	None	None	None	None
3	2	50	None (Linear)	0	0	None	None	None	None
4	1	10	None (Linear)	0	0	None	None	None	None
5	2	20	None (Linear)	0	0	None	None	None	None
6	1	50	None (Linear)	0	0	None	None	None	None
7	1	50	None (Linear)	0	0	None	None	None	None
8	1	50	ReLU	0	0	None	None	None	None
9	1	50	None (Linear)	0	0	None	None	None	None
10	1	50	None (Linear)	0	0	None	None	None	None
11	1	50	None (Linear)	0	0	None	None	None	None
12	1	50	None (Linear)	20	0	None	None	None	None
13	1	50	None (Linear)	0	20	None	None	None	None
14	1	50	None (Linear)	0	0	0.001	None	None	None
15	1	50	None (Linear)	0	0	0.1	None	None	None
16	1	50	None (Linear)	0	0	None	0.001	None	None
17	1	50	None (Linear)	0	0	None	0.1	None	None
18	1	50	None (Linear)	0	0	None	None	0.001	None
19	1	50	None (Linear)	0	0	None	None	0.1	None
20	1	50	None (Linear)	0	0	None	None	None	0.001
21	1	50	None (Linear)	0	0	None	None	None	0.1
22	2	50	None (Linear)	0	0	None	None	None	None
23	2	20	None (Linear)	0	0	None	None	None	None
24	1	50	None (Linear)	0	0	None	None	None	None
25	1	50	None (Linear)	0	0	None	None	None	None
26	2	50/25	None (Linear)	0	0	None	None	None	None
27	2	50/150	None (Linear)	0	0	None	None	None	None
28	3	50	None (Linear)	0	0	None	None	None	None
29	1	50	None (Linear)	0	0	None	None	None	None
30	1	50	None (Linear)	0	0	None	None	None	None
31	3	50	None (Linear)	0	0	None	None	None	None
32	3	20	None (Linear)	0	0	None	None	None	None
33	3	20	None (Linear)	0	0	None	None	None	None

Model Basic Neural Network Parameters							
Model Number	Number of Dense Layers (Not Output Layer)	Number of Filters Per Dense Layer (Not Output Layer)	Dense Layer Activation (Not Output Layer)	Dense Layer Dropout Percentage	Optimizer Type	Number of Epochs	Batch Size
1	0	None	None (Linear)	None	adam	20	10
2	0	None	None (Linear)	None	adam	20	10
3	0	None	None (Linear)	None	adam	20	10
4	0	None	None (Linear)	None	adam	20	10
5	0	None	None (Linear)	None	adam	20	10
6	1	50	None (Linear)	None	adam	20	10
7	1	50	None (Linear)	30	adam	20	10
8	0	None	None (Linear)	None	adam	20	10
9	0	None	ReLU	None	adam	20	10
10	0	None	None (Linear)	None	adam	20	5
11	0	None	None (Linear)	None	adam	20	100
12	0	None	None (Linear)	None	adam	20	10
13	0	None	None (Linear)	None	adam	20	10
14	0	None	None (Linear)	None	adam	20	10
15	0	None	None (Linear)	None	adam	20	10
16	0	None	None (Linear)	None	adam	20	10
17	0	None	None (Linear)	None	adam	20	10
18	0	None	None (Linear)	None	adam	20	10
19	0	None	None (Linear)	None	adam	20	10
20	0	None	None (Linear)	None	adam	20	10
21	0	None	None (Linear)	None	adam	20	10
22	1	50	None (Linear)	None	adam	20	10
23	1	50	None (Linear)	None	adam	20	10
24	1	100	None (Linear)	None	adam	20	10
25	1	100	None (Linear)	30	adam	20	10
26	0	None	None (Linear)	None	adam	20	10
27	0	None	None (Linear)	None	adam	20	10
28	0	None	None (Linear)	None	adam	20	10
29	0	None	None (Linear)	None	sgd	20	10
30	0	None	None (Linear)	None	adam	100	10
31	1	50	None (Linear)	None	adam	20	10
32	0	None	None (Linear)	None	adam	20	10
33	1	50	None (Linear)	None	adam	20	10

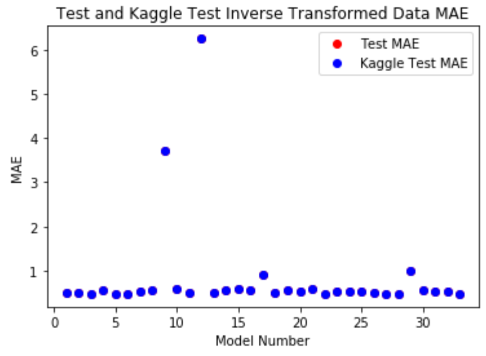
Final Model (Model 3) Training and Testing Losses Over Epochs:



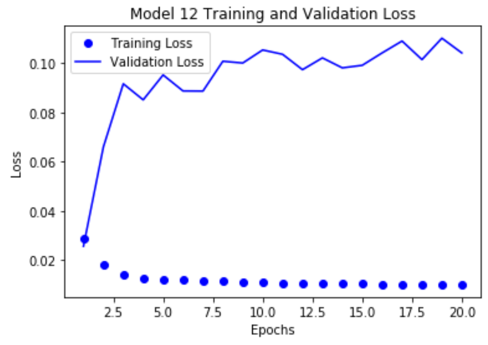
Final Model (Model 3) Kaggle Predictions and Actual Temperature Values Over Time:



All Model (Model 3) MAE Losses (Test MAE is for the test set used in modeling and Kaggle Test MAE is for the test set used to generate the Kaggle predictions):



Model 12 (20% Input LSTM Dropouts) Exhibiting Overfitting:



Final Project Review

I found this project to be very interesting and useful. It was very informative to be able to see how all of the different model hyperparameters affect modeling. It was also informative to learn the limitations of modeling using Keras, one of which I ran into being the random initial weight assignments. I did not expect there to be as much variation in loss values from the same model. Through this project I was able to get a better understanding of how each parameter effects the model, such as how lowering the batch size from 10 to 5 actually increased the prediction loss or how increasing the number of epochs did not impact the model (probably because the model had already converged in 20 epochs). It was also interesting to see how some parameters made the model worse, such as adding input dropouts, L1 kernel regularization with lambda 0.1, or using ReLU dense output activation. One of the challenges I had with this project was choosing a best model when there were several models with very similar losses. I think the models had similar losses because I was only altering the model a little at a time, which means there weren't large variations in the model results. I did this purposefully though in order to easier see the impact of each individual parameter. In hindsight I should have changed the parameter values more drastically (increased the number of LSTM nodes from 50 to 500 instead of 150 or increased the batch size from 10 to 1000 instead of 100, for example). This may have resulted in more different model results (results for models would may have more variation). This would have made interpreting how each parameter effected the model easier.