Abstract

Time series applications are growing in popularity and importance due to their ability to generate forecasts that influence business operations. In this assignment, the optimal deep learning time series model architecture is explored for a call center data set, with a broader goal of helping to inform call center staffing. After considering baseline models, RNN models, 1D CNN models, and stacked 1D CNN/RNN models, RNN and stacked 1D CNN/RNN models offer the best performance in terms of mean absolute error. Extensive commentary is provided on the model building process and future enhancements to consider when working on any time series or staffing optimization problem.

Introduction

Time series applications are increasing in popularity and importance. Divisions such as sales, marketing, and servicing rely on leveraging past information to predict a future result, often grappling with seasonal trends. This assignment uses daily-summarized data from Los Angeles (LA) city services calls (found on Kaggle) from 2011 – May 2015 to predict future daily call volumes two months in advance. This information can be used to provide feedback into staffing for the LA city services call center. By testing a wide variety of recurrent neural network (RNN) architectures, the power of deep machine learning models can be expressed within the servicing domain.

This assignment focuses on identifying the optimal RNN model architecture for the LA city services data. More specifically, dense, RNN, 1D convolution neural network (CNN), 1D CNN/RNN stacked models, and rudimentary ensembling models are all in scope. In casting a wide net and exploring several different types of model architectures, conclusions are made on key hyperparameters and model frameworks, serving as input for future time series endeavors.

Literature Review

The Chollet Deep Learning with Python textbook served as the primary source of information for this assignment, as it contained both theoretical information and code snippets for various types of time series models. One of the main takeaways from this source was how to approach the modeling process. First and foremost, according to Chollet, it is important to consider a baseline model, as it "will serve as a sanity check, and it will establish a baseline that you'll have to beat in order to demonstrate the usefulness of more-advanced machine-learning models" (p. 212). Chollet also explains the benefits of considering a dense model as a baseline model because it is cheap to implement (Chollet, p. 213). Thus, these baseline models are very important to develop up front in order to show the value of more complex and computationally expensive deep learning models.

Furthermore, Chollet provides a framework for model architectures and hyperparameter tuning. After considering baseline models, additional tuning frameworks should include: the number of nodes in

each layer, the learning rate, and the addition of more dense layers on top of recurrent layers (Chollet, p. 222). Therefore, these levers are used during the model building process. Additionally, Chollet recommends trying RNN, 1D CNN, and stacked 1D CNN/RNN models when working with time series data. Chollet speaks highly of the 1D CNN/RNN stacked model, saying, "it's effective and ought to be more common" (p. 229). Overall, the Chollet book has a profound influence in how the models are developed.

Moreover, the Goodfellow Deep Learning textbook also provided guidance on model architectures and hyperparameters. Although both Goodfellow and Chollet mention some of the challenges when trying to learn long-term dependencies, Goodfellow provides some deeper knowledge of bi-directional RNNs, stating that they "allow the output units to compute a representation that depends on both the past and the future but is most sensitive to the input values around time t" (p. 388-389). Bi-directional RNNs are considered in the model building process to see if they offer an advantage over traditional RNN models.

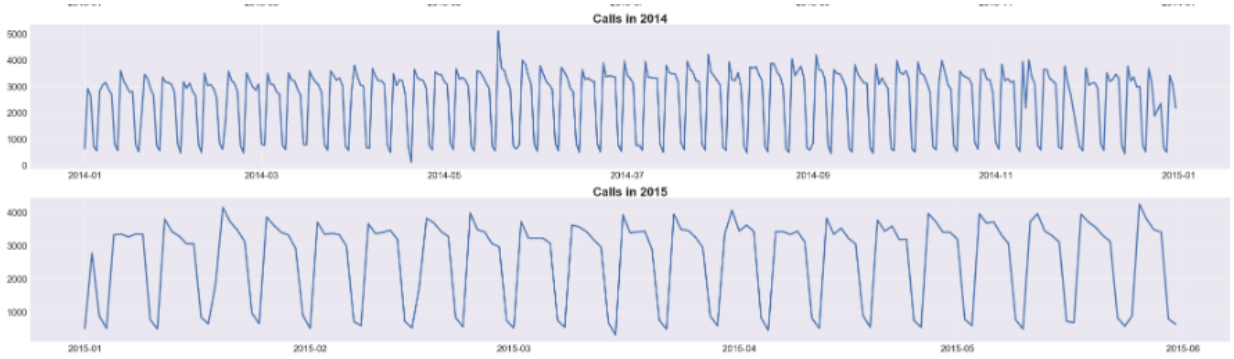
Lastly, Goodfellow goes deep on model building, speaking very highly of regularization and automatic hyperparameter tuning. For instance, Goodfellow says: "you should include some mild forms of regularization from the start...Dropout is an excellent regularizer that is easy to implement" (p. 420). Both dropout and recurrent dropout are considered to be the primary form of regularization for this time series problem. Also, Goodfellow provides commentary on the hyperparameter tuning process: "If you have time to tune only one hyperparameter, tune the learning rate. It controls the effective capacity of the model in a more complicated way than other hyperparameters" (p. 424). Goodfellow further states that the Adam optimizer is a reasonable choice for the optimization algorithm (p. 420). The emphasis on using regularization, tuning the learning rate, and using Adam all influence the model building process undertaken in this assignment. While automatic hyperparameter tuning is not applied in this assignment, it is a major consideration for future enhancements.

Methods

Data Preparation

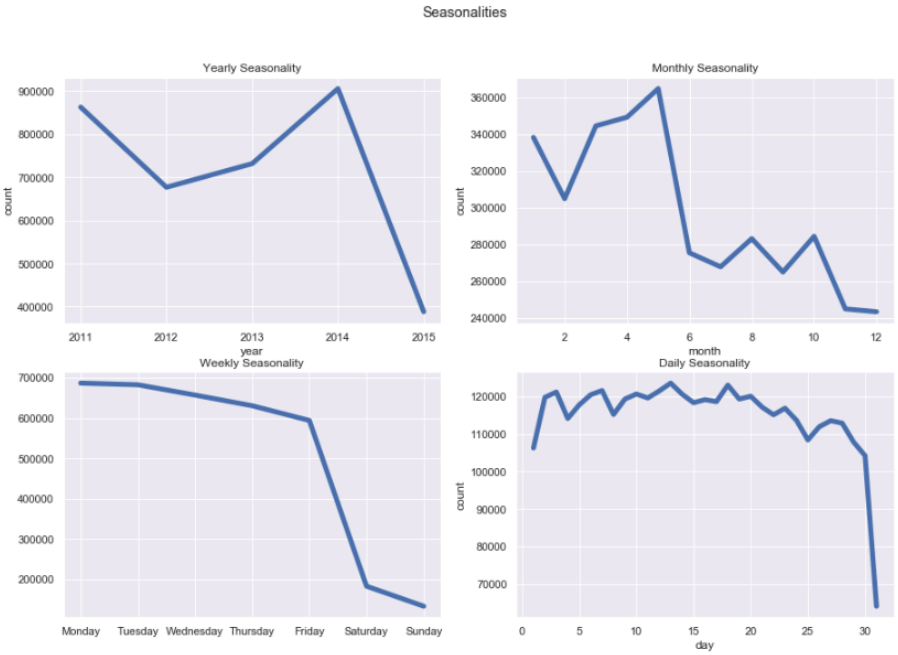
Several different data preparation steps are performed to prepare the data for modeling. First and foremost, the data is cleaned and aggregated at a daily level since the original call data was in its raw form at a call level. Next, the time series patterns are explored in greater detail. Visual 1 contains a snapshot of the overall trends in 2014 and 2015.

Visual 1: 2014-2015 time series



By inspecting the data at a broad level, it is clear that there are cyclical patters within the data. The different types of seasonalities are visualized in greater detail across all years, as shown in Visual 2 below. Through these visuals, it is better understood that the call volumes have been increasing since 2012 and are higher during the first half of the year, peaking in May, which is likely due to the fact that there are more spring cleanup activities. Regardless, it seems like at least three months of data is needed to inform future activity. Moreover, the visuals help to reveal that calls are heaviest on Monday and drop significantly towards the end of the month. In short, there are multiple seasonalities at play here, generating further support to explore deep learning models.

Visual 2: Time series seasonalities



In any time series problem, accounting for holidays is vitally important. Therefore, US Federal Holidays are added to the data frame in an effort to have any model more readily discern trends around holidays. Holiday analysis not a focus of this assignment and was thus cut short. However, more work

could have been done to assess the impact of a holiday on the days surrounding it and feed that into a model.

As a final data preparation step, the data is split into training, validation, and testing sets. Approximately 150 days each are used in the validation and testing sets, leaving the remaining data for training. Most of the data is used for training to ensure that year-over-year trends and other seasonalities can be discerned. Keras generators are used to split the data into their appropriate data frames. These generators help to simplify this process, as this data needs to be structured so that the past three months of data are used to predict the next two months of activity. Thus, it is a large data exercise to transform the data into smaller subsets of relevant inputs and outputs. All data frames have a length, or batch size, of 16. The call volume data, excluding holidays, in their final data frames are normalized as well.

Modeling

Numerous models are built for this assignment. Table and visual output are shown for select models. For all models, mean absolute error is used as the primary metric for model evaluation due to the steady time series pattern observed in the data. Adam is used for the optimizer, and both dropout and recurrent dropout are used to combat against overfitting. Twenty epochs are generated for all models, as this is a small enough number to discern trends yet protect against extremely long processing times.

To begin, a no-nonsense baseline model is generated for the validation set, using the prior week's calls (same day) for the next week's call volumes. This approach yielded a validation MAE of .021, which is extremely low. Therefore, it will be important to see if any deep learning models can beat this result.

Next, several different types of RNN models are considered. Focus is given towards models with multiple layers, as those models consistently displayed superior performance. Additionally, dropout and recurrent dropout are used throughout many of the models due to increased complexity inherent from using multiple layers. Table 1 contains a subset of RNN models and their performance.

From this table, it is evident that LSTM models are superior to a simple dense model, which makes sense given the nature of the data and the need to incorporate both short-term and long-term dependencies. Furthermore, models with three layers and node sizes of 32 consistently have better performance and lower processing times, while increasing the learning rate to .1 had a negative impact on performance. The best validation MAE value is .043, which is not better than the no-nonsense baseline. It seems that adding an extra dense layer at the end of a model slightly improves its performance. Bidirectional RNN models are better than the dense model but do not offer improved performance over other types of RNN models.

Table 1: RNN Training Results

RNN Training Results				
Model	Layers/Nodes	Train MAE	Val MAE	Processing Time
Dense	32	0.15	0.26	21.6
RNN w/ LSTM - Basic	32	0.03	0.045	110.7
RNN w/ LSTM - Basic	32 dropout .2,.2	0.029	0.044	124.3
RNN w/ LSTM - Basic	32,32	0.029	0.044	203.0
RNN w/ LSTM - Basic	32,32 dropout .2,.2	0.029	0.043	229.6
RNN w/ LSTM - Basic	32,32,32	0.028	0.043	4488.1
RNN w/ LSTM - Basic	32,32,32 dropout .2,.2	0.028	0.042	443.8
RNN w/ LSTM - Basic	64,64,64 dropout .2,.2	0.028	0.043	355.8
RNN w/ LSTM - Basic	128,128,128 dropout .2,.2	0.027	0.042	406.8
RNN w/ LSTM - Basic	256,256,256 dropout .5,.5	0.03	0.044	772.6
RNN w/ LSTM - Dropout	32 dropout .1,.5	0.03	0.044	171.3
RNN w/ LSTM - Dropout	32,32 dropout .1,.5	0.029	0.043	6616.6
RNN w/ LSTM - Dropout	32,32,32 dropout .1,.5	0.028	0.042	353.6
RNN w/ LSTM - Extra Dense	32(2 LSTM),64(1 dense) dropout .2,.2	0.028	0.043	229.6
RNN w/ LSTM - Extra Dense	32(3 LSTM),64(1 dense) dropout .2,.2	0.028	0.042	347.2
RNN w/ LSTM - Extra Dense	32(3 LSTM),64(2 dense) dropout .2,.2	0.028	0.042	353.9
RNN w/ LSTM - LR=.1	32(3 LSTM),64(1 dense) dropout .2,.2	0.068	0.066	2561.5
RNN w/ LSTM - LR=.0001	32(3 LSTM),64(2 dense) dropout .2,.2	0.028	0.042	2415.2
RNN w/ LSTM - Extra Dense LR=.0001	32,32,32,64 dropout .2,.2	0.028	0.042	352.7
RNN w/ LSTM - Extra Dense LR=.0001	32,32,32,64 dropout .5,.5	0.028	0.043	336.2
RNN w/ LSTM - Extra Dense LR=.0001	32,32,32,64 dropout .1.,1	0.028	0.043	338.3
RNN w/ LSTM - Extra Dense LR=.001	128,128,128,64 dropout .5,.5	0.029	0.043	412.8
RNN w/ LSTM - Extra Dense LR=.01	32,32,32,64 dropout .2,.2	0.03	0.043	361.7
RNN w/ LSTM - Extra Dense LR=.01	256,256,256,64 dropout .2,.2	0.06	0.044	4960.5
RNN w/ LSTM - Extra Dense LR=.01	256,256,256,64 dropout .5,.5	0.049	0.044	775.4
RNN w/ LSTM - Bidirectional	32	0.03	0.066	7363.8
RNN w/ LSTM - Bidirectional	64,64	0.029	0.065	15586.0
RNN w/ LSTM - Bidirectional	64,64,64	0.028	0.063	440.9
RNN w/ LSTM - Bidirectional	128,128,128	0.028	0.063	2783.7

In briefly examining 1D CNN model results from Table 2, it is clear that these models on their own do not offer improved performance compared to RNN models. Only two 1D CNN models were generated since the focus of this assignment is on RNN models and uncovering their optimal architecture. Furthermore, these 1D CNN models really serve as a baseline. That being said, the results are much improved by stacking 1D CNN and RNN models together. In fact, Table 3 shows that these CNN RNN models offer similar performance at a fraction of the RNN model processing times. The CNN RNN model with a very large processing time seems to be an anomaly. Thus, Chollet's reference to this stacking technique has paid off and lives true to its promise of providing speed while leveraging the order-sensitivity of RNNs (Chollet, p. 229).

Table 2: CNN Training Results

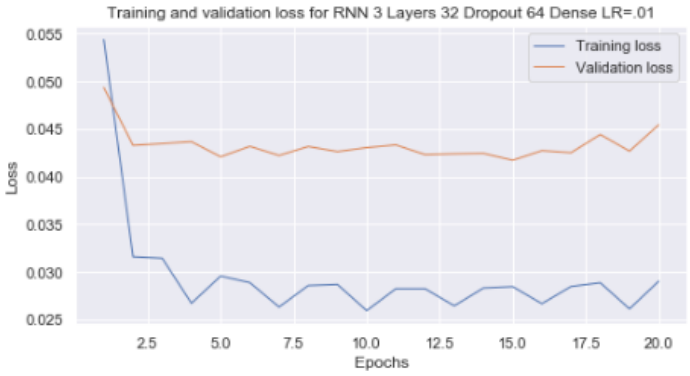
CNN Training Results				
Model	Layers/Nodes	Train MAE	Val MAE	Processing Time
CNN	32,32	0.076	0.14	10.1
CNN	64,64	0.079	0.11	11.7

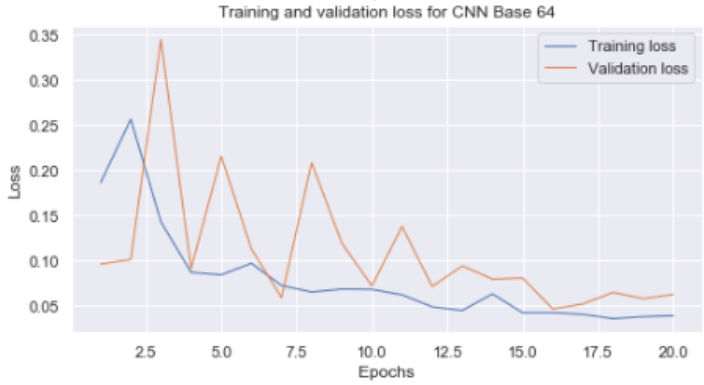
Table 3: CNN RNN Training Results

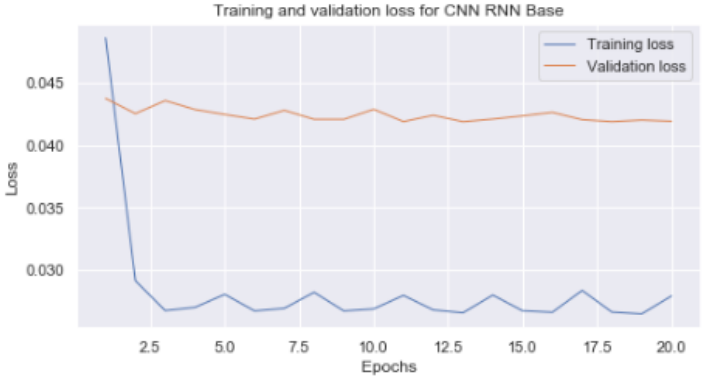
CNN RNN Training Results				
Model	Layers/Nodes	Train MAE	Val MAE	Processing Time
2 CNN/1 RNN	32,32,32 w/ dropout .2,.2	0.028	0.042	24.1
2 CNN/1 RNN	32,32,32 w/ dropout .1,.5	0.029	0.042	25.4
2 CNN/1 RNN	32,32,64 w/ dropout .2,.2	0.028	0.042	25.5
2 CNN/1 RNN	64,64,64 w/ dropout .2,.2	0.029	0.042	29.5
2 CNN/1 RNN	64,64,64 w/ dropout .1,.5	0.029	0.042	29.8
2 CNN/2 RNN	32,32,64,64 w/ dropout .2,.2	0.028	0.042	43.5
2 CNN/3 RNN	32,32,64,64,64 w/ dropout .2,.2	0.027	0.042	53.9
2 CNN/2 RNN Extra Dense	32,32,64,64 w/ dropout .2,.2	0.028	0.043	42.7
2 CNN/3 RNN Extra Dense	32,32,64,64,64 w/ dropout .2,.2	0.028	0.043	2112.6
2 CNN/3 RNN Extra Dense	32,32,32,32,32 w/ dropout .2,.2	0.028	0.042	69.2
2 CNN/3 RNN Extra Dense LR=.01	32,32,32,32,32 w/ dropout .2,.2	0.03	0.044	68.0

For context and reference, select loss or MAE curves on training and validation sets are shown below in Visual 3. These models are considered the best models and will be evaluated on the test set.

Visual 3: MAE or Loss Plots from select models







From these loss plots, it is clear that the RNN models and CNN/RNN model demonstrate a very similar pattern. It is important to keep in mind that the scales within the loss curves make the training and validation curves seem farther apart than they actually are. Regardless, even changing the node size, learning rate, and adding dense layers did not have as significant of an impact on validation MAE as initially anticipated. It is also evident that the CNN model demonstrates greater variability in terms of MAE results, which is why it is an inferior model. With more time and possibly additional data, closer training and validation results could be achieved.

Results

In evaluating the five models on the test set, the CNN model clearly does not perform the best according to Table 4 below. However, the RNN Extra Dense 64 model has the lowest MAE value and is thus the final model. While ensembling methods were not considered during training, simple approaches to ensembling (i.e., equal weights) are shown below and help to illustrate the potential of considering an ensemble model as the final model in the future.

Table 4: Test set results

Test Set Results		
Model	Layers/Nodes	MAE
RNN	32,32,32 dropout .1,.5	0.000288
RNN Extra Dense 64	32,32,32 dropout .2,.2	0.000169
RNN Extra Dense 64 LR=.01	32,32,32 dropout .2,.2	0.00412
CNN	64,64	0.0256
2 CNN/1 RNN	32,32,32 w/ dropout .2,.2	0.000284
Ensemble - All Models		0.00609
Ensemble - No CNN		0.000245
Ensemble - RNN/CNN RNN		0.000227

This assignment helps to reveal several key considerations that can be applied to future time series and staffing optimization problems. First, it is important to utilize a no-nonsense model, as it might be the best model. While the no-nonsense model did have the lowest validation MAE, it is assumed that this model would lack generalizability over the long run, which is why deep learning models are preferred. If models were built using differently sized validation sets, then the poor performance of a no-nonsense baseline would be better revealed. Nonetheless, having this baseline model is an important step in justifying the cost of a deep learning model.

Next, both RNN and stacked CNN/RNN models performed the best on this data set, with an RNN model having a slight edge based solely on test set performance. For future use cases, a stacked CNN/RNN model should be considered, as it has the potential to yield equivalent performance with lower processing times.

If more time permitted, there are several additional items that would have been explored and/or implemented. Specifically, more effort would have been given toward analyzing the time series and removing outliers. Anomaly detection could have been applied to reveal dates that should be altered (i.e., clipped) for modeling. Additionally, inspecting the days surrounding holidays could have been done so as to account for the impact of holidays on call volumes surrounding that given holiday. Instead, this assignment only considered the impact of holiday itself. Furthermore, model ensembling could have been a greater focus, as this approach has the ability to yield more stable results over the long run.

Conclusions

To be successful in any time series project, especially a staffing optimization problem, appropriate time should be given towards understanding business requirements. For instance, in a servicing context like a call center, two potential types of models could be used. One model, with daily predictions, could help supervisors discern if additional staff needs to be hired and trained to support future call volumes. Another model, at an hourly level, could be used to help plan schedules for existing staff for the next day or next week. By utilizing these two models together, supervisors would have the tools to staff for both the near-term and long-term. Additionally, business goals for model accuracy should be documented and used to guide model development.

Furthermore, adding in other data points, if relevant, would aid in model accuracy. While only holidays were considered for this assignment, incorporating key city events that help to explain significant increased or decreased call volumes would boost model performance. It is important to make sure that business goals are known so that the right kinds of data are used and the right kinds of models are built.

Next, it is important to recognize that the model building process takes some time. In using the approaches outlined by Chollet and Goodfellow, the value of deep learning models can be justified in an incremental fashion. To aid in the model development process, automatic hyperparameter optimization, such as outlined in the package hyperas, should be used, as this will speed up model building time. Although traditional time series models, such as ARIMA, were out of scope for this assignment, these types of models should be built as additional baseline models in order to ensure that a deep learning model is truly warranted.

Lastly, an ensemble model should be considered as the final model in sharing predictions with call center supervisors. This type of model will offer stability and help mitigate any potential nuances in the time series. Smoothing techniques could also be considered prior to sharing the predictions with the call center supervisors to ensure that predictions are not too volatile and hiring decisions are truly warranted. In short, by building upon the foundation set forth in this assignment, a successful model can be built to aid in staffing decisions for any call center.

References

Angeles, C. O. (2018, September 04). Los Angeles 311 Call Center Tracking Data. Retrieved May 23, 2019, from https://www.kaggle.com/cityofLA/los-angeles-311-call-center-tracking-data

Chollet, F. (2018). *Deep Learning with Python*. Shelter Island: Manning.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. Cambridge (EE. UU.): MIT Press.