**MODEL SUMMARY**

**7th Placed Solution for M5 Forecasting - Uncertainty**

**Background**

Competition Name: M5 Forecasting - Uncertainty

Team Name: Ka Ho\_STU

Private Leaderboard Score: 0.1616

Private Leaderboard Place: 7th

Name: Ka Ho

Location: Hong Kong

Email: [khtsangdavid@gmail.com](mailto:khtsangdavid@gmail.com)

**Team Background**

I am a full-time student studying in the Chinese University of Hong Kong, enrolling in the Master of Philosophy program in Risk Management Science. I joined this forecasting competition in the second year of my study. At that time, I was curious about how to apply a machine learning framework to the forecasting problem. After reading several blogs and papers, I was motivated to join this competition to implement those ideas.

**Model Summary**

The final model is an ensemble model of two seq2seq LSTM models, trained to forecast on two different hierarchical levels (dept\_store level and store\_item level). The ensemble model first gives the point forecast, then the prediction quantiles are computed using model residuals information based on normality assumptions. The major features are the lag-features (i.e. previous 28D time series, encoded by the LSTM encoder model), the weekday variable (for capturing weekly seasonality), as well as the identifier features (e.g. item\_id, dept\_id, store\_id) that are post-processed by the embedding layer. The entire NN model is built using Keras functionality. It takes around 4 hours to train the model on a Kaggle Kernel or 12 hours locally.

**Methodology**

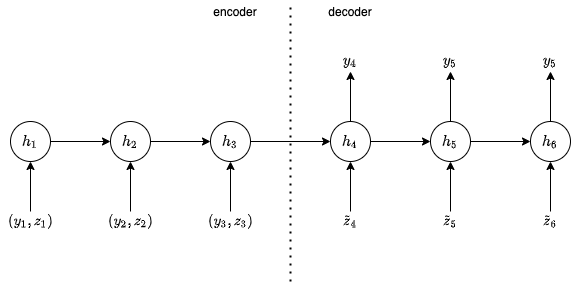
We think the key factor for our success is a reliable local validation scheme, together with the model ensembling technique, which makes our model to deliver stable performance both locally and in public / private LB.

This section can be divided four parts, namely:

1. Model Architecture
2. Features Engineering & Processing
3. Training Method
4. Local Validation Scheme

**Model Architecture**

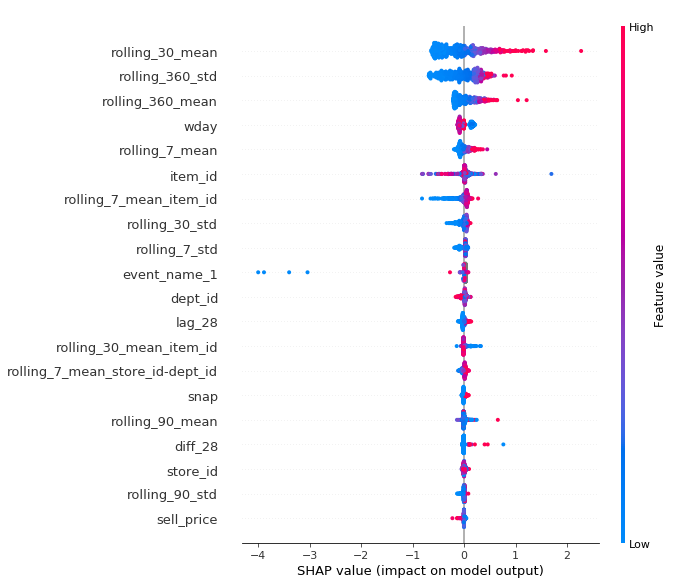
The basic seq2seq LSTM model can be described by the following graph (the input / output window becomes 28D in the competition):



where y denotes the time series (i.e. sales), z denotes the exogenous features (e.g. weekday, holiday\_event, is\_snap, item\_id), h denotes the hidden state (as well as cell state). The above architecture shares a large similarity with encoder-decoder model for machine translation except that I add some exogenous features to the model, and there is no forecast value as input to the decoder model (i.e. trained unconditionally).

In the submission model, both encoder and decoder have 2 stacked LSTM layers. The number of hidden units are (64, 32) respectively.

**Features Engineering & Processing**

The below SHAP plot provides the variable importance of different features when trained using LGBM model:

From the above plot, we observe the followings:

1. The key features are the lag-features (e.g. rolling\_30\_mean, ...)
2. Apart from lag-features, weekday (wday), item\_id, dept\_id also have some contribution to the model performance

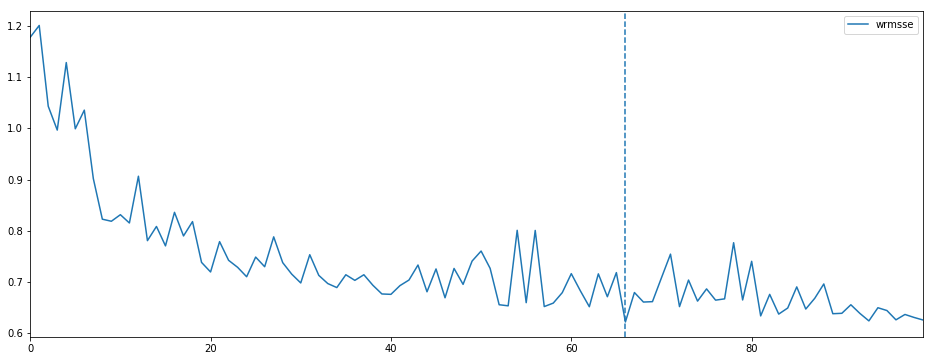
The exogenous variables added to the LSTM model is largely based on the above feature importance plot. The categorical features of high cardinality are further processed by an embedding layer before inputting to the model.

In addition, to train the dept\_store model, all the time series are re-scaled into [0, 1], and the model is trained to predict the increment of sales value since the first day of input window (so as to remove trend). Both rescaling and trend removal are observed to provide a great boost in performance for dept\_store model but not for store\_item model.

**Training Method**

The LSTM models are trained using Adam algorithm. The dept\_store model is trained with weighted MSE loss. The store\_item model is trained with unweighted zero-inflated poisson loss (to deal with cases with many 0s).

During training, an issue of model instability is observed. The following graph shows how the WRMSSE metric (used for accuracy measure) changes during training epochs.



From the above metric curve, we observe that the model performance is highly volatile with respective to training epochs. Moreover, it is observed that the model performance fluctuates dramatically when the random seed for training changes. Both indicate an issue of model instability.

To resolve the instability issue, each seq2seq LSTM model is actually consisting of models trained at 20 different checkpoints (i.e. training epochs) and is a simple average of the 20 checkpoint models. The submission model is trained blindly without any early stopping, and is observed to have stable performance.

While ensembling across model checkpoints is used for improving stability, another ensembling technique (between dept\_store model and store\_item model) is used to produce more accurate forecast. The idea behind is that each model can capture the specific time-series pattern at a specific hierarchical level. It is observed that the store\_item model consistently delivers a better forecast at bottom levels (i.e. store\_item, state\_item, item), and the dept\_store model performs better at top levels.

To combine the model forecast, a simple average of them is used. It is observed that simple averaging already provides a similar performance when compared with any optimal reconciliation approach (<https://otexts.com/fpp2/reconciliation.html>).

**Local Validation Scheme**

I think that a reliable local validation scheme is extremely important in this competition. At the early stage of the competition, I mainly focus on the accuracy metric (i.e. WRMSSE) for the point forecast. The WRMSSE metric can be volatile given only 28D of validation window, because it is equally weighted on all aggregate levels (meaning that the top aggregate levels forecast, even with a few observations, already give a huge proportion of the loss). Thus, throughout the competition, I mainly look at the local validation result but not the public LB.

My local validation scheme consists of forecasting on 4 folds of the most recent data (each fold has 28D validation window). The model is trained once only (i.e. no model re-training) without the 4 folds of validation data, and is used for forecasting the next 4 folds of validation data consecutively. This local validation scheme already gives a close alignment between the local score and the public / private LB.

**Simple Model**

Suppose we are looking for a simpler model, we may consider using only the store\_item model or the dept\_store model. Both model delivers similar performance, and dept\_store model is considerably simpler in terms of the number of features, and the number of time series to be trained with.

The dept\_store model alone can achieve 0.1669 WSPL at the private LB, compared with 0.1616 WSPL for the ensemble model.

**Model Execution Time**

It takes around 2 hrs to fit for each seq2seq LSTM model. The dept\_store model takes 10s to produce a forecast for all dept\_store series in 28D, while the store\_item model spends 2 mins forecasting all store\_item series in 28D.

**Reference**

1. <https://www.kaggle.com/c/web-traffic-time-series-forecasting/discussion/43795>
2. <https://otexts.com/fpp2/reconciliation.html>