Internship Report

Taichung Traffic Forecasting

Lucas Biéchy

Department of Mathematics, University of Paris-Saclay, Orsay Mathematics Laboratory
Email address: lucas.biechy@universite-paris-saclay.fr

GitHub repository : https://github.com/Biechy/Master-of-Science-Mathematics-and-AI/tree/main/Internship%20-%20NARLabs

Abstract

During my internship, I aimed to improve traffic prediction systems in Taichung. With data formatted tabularly without location information, I focused on individual roads. I developed a method for imputing missing data, and then I have implemented a new state of the art models: Extended Long-Short Term Memory. This model presents better results. I also developed an alternative model, the Extended Gated Recurrent Units, which showed promising results.

Contents

Acknowledgements.

I would like to express my deep gratitude to the NCHC laboratory, a subsidiary of NARLabs, for welcoming me. I really extend my heartfelt thanks to my supervisors, Mr. 黃仲誼 (Schumi) and Dr. 張日昇 (Risheng), for their guidance throughout this experience. A special thanks to Ms. 張惠婷(Melody), who was in charge of international interns and whose support was invaluable, as well as to Mr. 張文鎰(Wen-Yi) for him helpful assistance. I also wish to thank the DataIA institute for introducing me to this international internship program and for providing me with an additional scholarship.

1. Introduction

1.1. Environment

The National Applied Research Laboratories (NARLabs) is a research institute affiliated with the Taiwanese government, dedicated to promoting and supporting applied research and innovation across various technological fields. Its activities span sectors such as environmental technologies, biomedical technologies, and information and communication technologies. NARLabs consists of seven specialized laboratories, including the National Laboratory Animal Center (NLAC), the National Center for Research on Earthquake Engineering (NRCEE), the Taiwan Semiconductor Research Institute (TSRI), and the National Center for High-Performance Computing (NCHC).

I completed my internship at the NCHC, specifically within the Data Science & Technologies division. NCHC has three geographical locations: a main branch in Hsinchu and two secondary branches in Taichung and Tainan. I carried out my internship at the main branch in Hsinchu, alongside other international interns. My assigned mentor was Mr. 黃仲誼(Schumi), who works closely with Dr. 張日昇(Risheng), both of whom are based at the Taichung branch. As a result, I often worked independently and did not have many opportunities to meet them in person. However, I received valuable support from Mr. 張文鎰(Wen-Yi), the division head, whenever it was needed.

1.2. Objective

The National Center for High-Performance Computing (NCHC) manages a broad range of projects, from physical infrastructures such as supercomputers to web platforms for smart cities. These platforms, for example, enable the visualization of various location-specific characteristics in a three-dimensional map, such as noise, temperature, or air pollution. During my internship, I worked on the *Traffic Congestion Prediction System* platform, designed to predict traffic congestion in Taichung. The goal is to anticipate traffic density for the next twenty minutes based on current and historical data. In the long term, the system aims to be shared with authorities and emergency services to avoid delays caused by traffic jams during critical operations. My task was to improve the accuracy of traffic predictions, particularly by better handling unexpected events that could disrupt the system, which involved updating the machine learning models used.

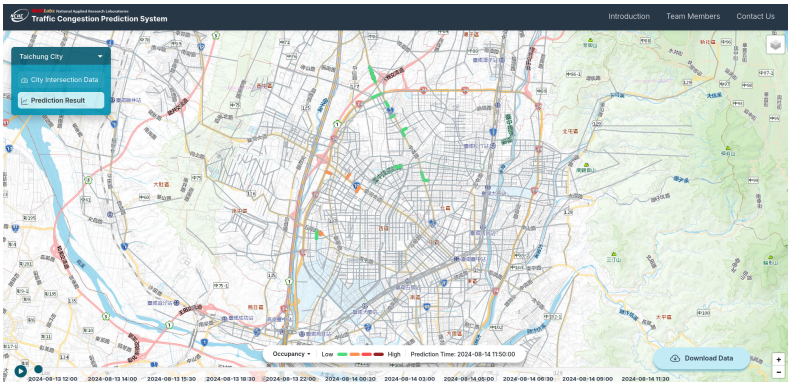

Figure 1. Website of the Traffic Congestion Prediction System

1.3. Data

The system, which is already operational, utilizes sensors installed by NCHC on the streets of Taichung to transmit real-time data to the servers. This information and its associated forecasts are publicly accessible on `https://tcps.nchc.org.tw/`. Currently, only 61 roads are recorded, but due to challenges related to sensor maintenance, only 27 roads were selected for the study as part of my internship.

The collected data is organized in a tabular format, including two necessary variables: time, with a 5-minute granularity, and the road ID. Moreover there are three descriptive variables, which are both explanatory and predictive model outputs:

- AVGSpeed : The average speed of vehicles on the road, measured in km/h.
- AVGOccupy : The occupancy rate of the road by vehicles, expressed as a percentage (0% to 100%).
- Total Volume : The total number of vehicles recorded on the road, expressed in vehicle units.

Although these variables might seem highly correlated and thus potentially challenging for creating a robust and identifiable statistical model, the situation is different in Taiwan, where the high prevalence of scooters significantly reduces this correlation. It is also noteworthy that the absence of data on road locations allows the assumption that each road is independent of the others. This justifies the use of parallelized predictions and employing a distinct model for each road. However, this approach has the drawback of not accounting for potential causal effects of traffic between different roads. Additionally, the dataset contains duplicates and missing lines, as shown in the Table 1. The data is in *.csv* format, with a file size of approximately 35MiB per road for 6 months.

RoadID	AVGSpeed	AVGOccupy	TotalVol	UpdateTime
m2CX17	40	81	134	2024-08-21 09:20:00
m2YI01	38	2	5	2024-08-21 09:20:00
m2YI01	39	1	6	2024-08-21 09:20:00
m30112	51	8	120	2024-08-21 09:20:00
m40131	40	19	82	2024-08-21 09:20:00
m4EO05	32	75	60	2024-08-21 09:20:00
m3EO05	30	79	136	2024-08-21 09:20:00
m40128	57	6	77	2024-08-21 09:15:00
m10448	52	3	85	2024-08-21 09:15:00
m2FS01	38	83	92	2024-08-21 09:10:00
m20317	37	54	139	2024-08-21 09:10:00
...	...	...	...	...

Table 1. Data Table Exemple

Since the results are similar for the other roads, this report will focus on the analysis of road *m20317*, which has the fewest missing data compared to the other roads. The Figure 2 illustrates the three main variables over time for this road.

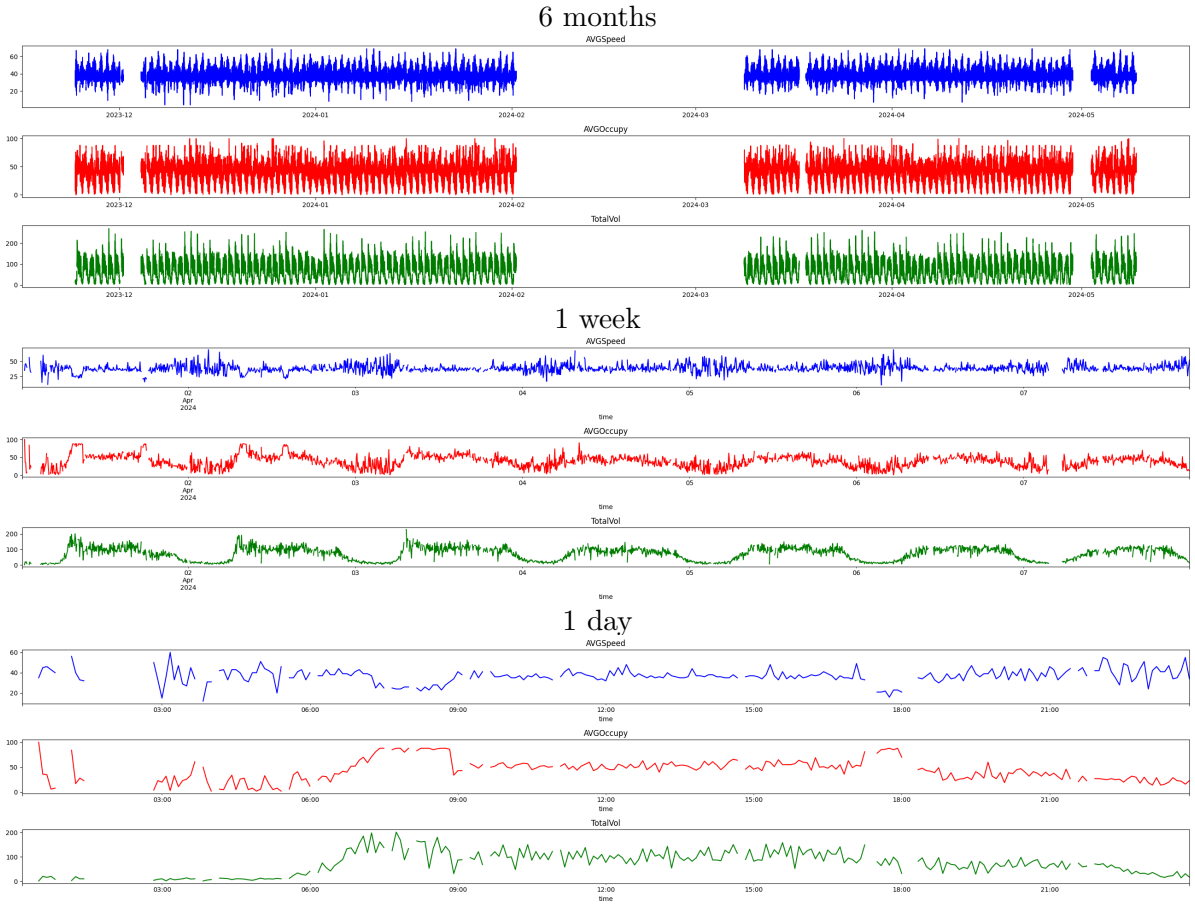

Figure 2. Road *m20317* for 3 different time scales (6months, 1week and 1day)

We will now delve into the most advanced deep learning models in the field of time series prediction to better understand the techniques and methodologies that could be applied to enhance our forecasts.

2. State of the art

The current prediction system at NCHC uses the Long Short Term Memory[1](LSTM) deep learning model for its forecasts. Given recent developments in this field, including the emergence of a new model inspired by LSTM, we have chosen to focus on this new approach rather than exploring traditional statistical solutions. This new model promises significant advancements in prediction accuracy. Therefore, we have decided to evaluate its performance and, if the results are promising, update our system with this cutting-edge technology. Before introducing the new model, it's important to review LSTMs to understand how this innovative model pushes beyond their limitations.

Let $X = (x_t)_{t\in[1,T]} \in \mathbb{R}^{d\times T}$ be a time series, or a subset of this time series, where d represents the dimension of the input vector and T is the sequence length. It is important to note that d may differ from the number of explanatory variables if a projection process has been applied beforehand. Finally, let $y \in R^{\ell}$ be the target to predict. In our study, $\ell = 3$ (AVGSpeed, AVGOccupy and TotalVol) and y represents the vector of the time series at $T + 20$min for this three variables.

2.1. LSTM : Long-Short Term Memory

Assume that

$$\forall t \in [1,T], \quad x_t \mid x_{t-1}, ..., x_1 \sim x_t \mid h_{t-1}, c_{t-1}$$

où h_{t-1} et c_{t-1} are two latent variables in $\mathbb{R}^h$ representing respectively short-term memory and long-term memory.

Long Short-Term Memory (LSTM) learning is a sequential process defined as follows:

- Initialization : $h_0 = c_0 := \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^h$

- $\forall t \in [1,T]$ Forward pass :

  - forget gate : $f_t = \sigma\big(W_f x_t + U_f h_{t-1} + b_f\big)$
  - input gate : $i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$
  - candidate memory : $z_t = \tanh(W_z x_t + U_z h_{t-1} + b_z)$
  - output gate : $o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$

where $\forall j \in [f,i,z,o], W_j \in \mathbb{R}^{h\times d}, U_j \in \mathbb{R}^{h\times h}, b_j \in \mathbb{R}^h$ are the model weights to be optimized during backpropagation (initialized by the chosen method).

  - $c_t = f_t \odot c_{t-1} + i_t \odot z_t$
  - $h_t = o_t \odot \tanh(c_t)$

The output h_T is then the desired prediction. If the dimensions do not match the target y, it can be projected into the desired dimensional space.

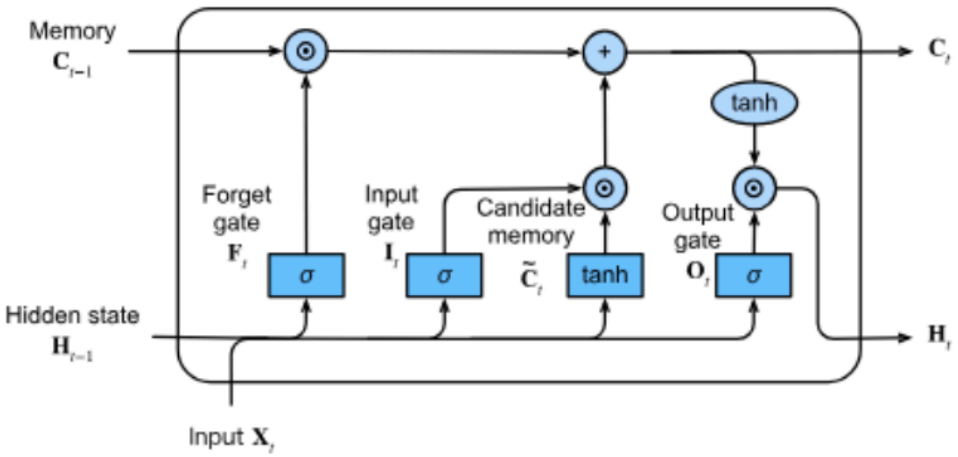

Figure 3. Schematic representation of an LSTM

LSTMs[1] are known for overcoming some of the limitations of Recurrent Neural Networks[2] (RNN), particularly issues with exploding and vanishing gradients. However, two main limitations persist:

- Lack of Parallelization: Due to the mixing of memory and the hidden-to-hidden connections between hidden states at different time steps, LSTMs require sequential processing. This limits their ability to be parallelized, which can slow down training and inference.
- Inability to Revise Storage Decisions: Due to the binary nature of the sigmoid function, decisions regarding the retention or forgetting of information are fixed once made, even if the stored information becomes outdated or blocked. This rigidity in memory management cannot be dynamically adjusted over time.

In this context, Extended Long-Short Term Memory[3] (xLSTM) models come into play.

2.2. xLSTM : extended Long-Short Term Memory

Extended Long-Short Term Memory[3] (xLSTM) models aim to enhance parallelization and provide greater flexibility in revising stored information. To address each of these constraints, two new processes are proposed.

2.2.1. sLSTM : scalarLSTM

The scalarLSTM[3] addresses the issue of the inability to revise storage decisions, which is primarily caused by the binary nature of the sigmoid function. To remedy this, the author proposes replacing the sigmoid function with a smoother function, the exponential function. This approach allows for the creation of a superposition state of information, providing the opportunity to later decide during the sequential process whether it is relevant to remember or forget that information. To avoid exploding gradient, a stabilizer[4] m_t is add in the exponential function.
Assuming that

$$\forall t \in [1, T], \quad x_t \mid x_{t-1}, ..., x_1 \sim x_t \mid h_{t-1}, c_{t-1}$$

where h_{t-1} et c_{t-1} are two latent variables in $\mathbb{R}^h$ representing respectively short-term and long-term memory.
The scalarLSTM learning process is defined sequentially as follows:
- InitInitialization: $h_0 = c_0 = m_0 = n_0 = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^h$
- $\forall t \in [1, T]$ Forward pass (with $\tilde{f}_t = W_f x_t + U_f h_{t-1} + b_f$ and $\tilde{i}_t = W_i x_t + U_i h_{t-1} + b_i$):
  - $m_t = \max\left(\tilde{f}_t + m_{t-1}, \tilde{i}\right)$
  - $f_t = \exp\left(\tilde{f}_t - m_t + m_{t-1}\right)$
  - $i_t = \exp\left(\tilde{i}_t - m_t\right)$
  - $z_t = \tanh(W_z x_t + U_z h_{t-1} + b_z)$
  - $o_t = (W_o x_t + U_o h_{t-1} + b_o)$
  - $n_t = f_t \odot n_{t-1} + i_t$

where $\forall j \in [f, i, z, o], W_j \in \mathbb{R}^{h \times d}, U_j \in \mathbb{R}^{h \times h}, b_j \in \mathbb{R}^h$ are the model weights to be optimized during backpropagation (initialized by the chosen method).

- $c_t = c_{t-1} \odot f_t + i_t \odot z_t$
- $h_t = o_t \odot c_t / n_t$

Proof of Equivalence with or without Stabilizer

Let h_t^{unstab}, c_t^{unstab} and n_t^{unstab} be the latent variables when m_t is not used. We have:

$$c_t^{\text{unstab}} = c_t \exp(-m_t) \text{ et } n_t^{\text{unstab}} = n_t \exp(-m_t)$$

We can verify the equivalence as follows:

$$\frac{c_t}{n_t} = \frac{c_{t-1} \odot f_t + i_t \odot z_t}{f_t \odot n_{t-1} + i_t} = \frac{c_{t-1} \odot \exp\big(\tilde{f} - m_t + m_{t-1}\big) + \exp\big(\tilde{i} - m_t\big) \odot z_t}{\exp\big(\tilde{f} - m_t + m_{t-1}\big) \odot n_{t-1} + \exp\big(\tilde{i} - m_t\big)}$$

$$= \frac{c_{t-1} \odot \exp\big(\tilde{f} + m_{t-1}\big) + \exp\big(\tilde{i}\big) \odot z_t}{\exp\big(\tilde{f} + m_{t-1}\big) \odot n_{t-1} + \exp\big(\tilde{i}\big)} = \frac{c_{t-1}^{\text{unstab}} \odot \exp\big(\tilde{f}\big) + \exp\big(\tilde{i}\big) \odot z_t}{\exp\big(\tilde{f}\big) \odot n_{t-1}^{\text{unstab}} + \exp\big(\tilde{i}\big)} = \frac{c_t^{\text{unstab}}}{n_t^{\text{unstab}}}$$

Thus, we have $h_t = h_t^{\text{unstab}}$

2.2.2. mLSTM : memoryLSTM

The memoryLSTM[3] is somewhat more complex than its counterpart. To address the lack of parallelization in the LSTM[1], it introduces an additional dimension to the latent variable $c_t \in \mathbb{R}^h$, which becomes $C_t \in \mathbb{R}^{h \times h}$ and removes the other latent variable $h_t \in \mathbb{R}^h$. This allows for parallel computation of all h_t (see proof below). A new update function for this latent variable must be defined. The rule for updating the optimal covariance [5] is then:

$$C_t = C_{t-1} + v_k k_t^\top$$

where $v_k \in \mathbb{R}^h$ and $k_t \in \mathbb{R}^h$ are respectively a value vector and a key vector (using terminology from Transformers[6]). The specific feature here is that, at a later time $t + \tau$, the value v_t hould be retrievable by a query vector $q_{t+\tau} \in \mathbb{R}^h$.

Assuming now that

$$\forall t \in [1, T], \quad x_t \mid x_{t-1}, ..., x_1 \sim x_t \mid C_{t-1}$$

where C_{t-1} is a latent variable in $\mathbb{R}^{h \times h}$ representing the memory.

The training process for memoryLSTM is defined as follows:

- Initialization : $h_0 = m_0 = n_0 = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^h$ and $C_0 = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}_{i \in [1,h]} \in \mathbb{R}^{h \times h}$
- $\forall t \in [1, T]$ Forward pass (with $\tilde{f}_t = W_f x_t + b_f$ and $\tilde{i}_t = W_i x_t + b_i$):
  - $m_t = \max\big(\tilde{f}_t + m_{t-1}, \tilde{i}_t\big)$
  - $f_t = \exp\big(\tilde{f}_t - m_t + m_{t-1}\big)$
  - $i_t = \exp\big(\tilde{i}_t - m_t\big)$
  - $o_t = \sigma(W_o x_t + b_o)$
  - $v_t = W_v x_t + b_v$
  - $k_t = W_k x_t / \sqrt{h} + b_k$
  - $q_t = W_q x_t + b_q$
  - $n_t = f_t \odot n_{t-1} + i_t \odot k_t$

where $\forall j \in [f, i, o, v, k, q], W_j \in \mathbb{R}^{h \times d}, b_j \in \mathbb{R}^h$ are the deep learning model weights to be optimized during backpropagation (initialized using the chosen method).

- $C_t = C_{t-1} \odot f_t + i_t \odot v_k k_t^\top$
- $h_t = o_t \odot C_t q_t / \max\big(\big|n_t^\top q_t\big|, 1\big)$

Proof of Forward Pass Parallelization

Consider the following:

$$X = (x_t)_{t\in[1,T]} \in \mathbb{R}^{d\times T},\ \tilde{f}_t = W_f x_t + b_f \in \mathbb{R}^h,\ \tilde{i}_t = W_i x_t + b_i \in \mathbb{R}^h$$

$$F_{i,j} = \begin{cases} 0 & \text{for } j > i \\ 1 & \text{for } j = i \\ \prod_{t=j+1}^{i} \exp\left(\tilde{f}_t\right) & \text{for } j < i \end{cases} \in \mathbb{R}^{h\times T\times T}$$

$$I_{i,j} = \begin{cases} 0 & \text{for } j > i \\ \exp\left(\tilde{i}_j\right) & \text{for } j \leq i \end{cases} \in \mathbb{R}^{h\times T\times T}$$

We then have the queries, keys, and values $Q, K, V \in \mathbb{R}^{T\times h}$,

$$\widetilde{H} \coloneqq CQ, \quad \text{with } C = \frac{\tilde{C}}{\max\left(\left|\sum_{j=1}^{T} \tilde{C}_{i,j}\right|, 1\right)} \quad \text{and } \tilde{C} = \frac{VK^\top}{\sqrt{h}} \odot (F + I)$$

We can then directly compute:

$$H = \sigma(W_o X + B_o) \odot \widetilde{H}$$

where $W_o \in \mathbb{R}^{T\times h\times d}$ et $B_o \in \mathbb{R}^{T\times h}$.

Note that we have just proved this for a forward pass without the stabilizer. By the definition of normalization n_t, this is equivalent to the case with a stabilizer.

2.2.3. xLSTM Blocks

The extended Long Short Term Memory[3] model will not be used directly in this way, but will be used to develop the following blocks:

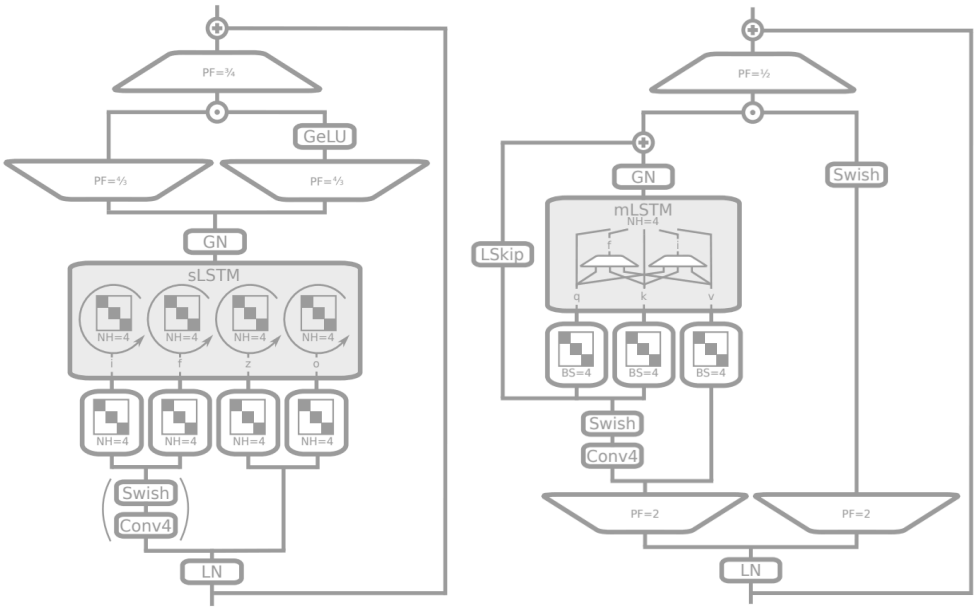

Figure 4. Schematic representation of a sLSTM Block and a mLSTM Block

Schematic representation of an sLSTM Block – Embedded in a pre-LayerNorm[7] residual structure, the input is optionally passed through a causal convolution of window size 4 that includes a Swish[8] activation for input and forget gates. Then, for all input, forget and output gates i, f, o, and the cell update z the input is fed through a block-diagonal linear layer with four diagonal blocks or "heads". These diagonal blocks coincide with the recurrent gate pre-activations from the last hidden state, which corresponds to an sLSTM with four heads depicted with the circular arrows. The resulting hidden state goes through a GroupNorm[9] layer – a head-wise LayerNorm[7] for each of the four heads. Finally, the output is up- and down-projected using a gated MLP, with GeLU[10] activation function and projection factor 4/3 to match parameters.
Schematic representation of an mLSTM block – Embedded in a pre-LayerNorm[7] residual structure, the input is up-projected first with projection factor 2, once for an externalized output gate and once as input for the mLSTM cells. The mLSTM cell input is dimension-wise causally convolved (kernel size 4), before entering a learnable skip connection. We obtain input q and k via block-diagonal projection matrices of block size 4. The values v are fed directly, skipping the convolution part. After the mLSTM sequence mixing, outputs are normalized via GroupNorm[9] – a head-wise LayerNorm[7] for each of the four heads. Finally, the learnable skip input is added and the result is gated component-wise with the external output gate. The output is down-projected.

To sum up, an xLSTM network is a stack of block sLSTM and block mLSTM layers with proportion and depth as hyperparameters.

3. Contribution

During this internship, I contributed in three main ways. Firstly, I developed a script to impute missing data while aiming to preserve the characteristics of the time series as much as possible. With the completed data, I was able to train a new model, the Extended Long Short Term Memory[3] (xLSTM), which I then implemented and compared to the performance of the existing model. This implementation allowed me to identify certain limitations in the xLSTM architecture. In response, I designed a new model, the Extended Gated Recurrent Unit (xGRU), which proved to be more effective for this prediction task but has no theoretical foundation.

3.1. Imputation

In our study, we encounter a mix of missing data types: some are Missing Completely at Random (MCAR) due to occasional sensor anomalies, while others are Not Missing Completely at Random (NMAR) due to prolonged sensor failures. When data is missing, the two associated variables are also missing, which complicates the process of finding solid studies with suitable methodologies, especially considering the temporal dependencies involved. Consequently, I have decided to develop my own Python script, which, while not optimal, is sufficient for our needs. This approach involves three distinct imputation strategies to handle different durations of missing data, with each strategy applied to each variable.

1. For data missing for less than an hour, I applied linear imputation. Drawing inspiration from finance, I assumed that there are few realistic trajectories during such a short period, so linear imputation is reasonably close to reality.
2. For data missing between one hour and one week, I chose to use mean imputation, considering that we were still relatively close to the MCAR regime. In this context, "mean" refers to the average of typical values for each specific time period. For instance, if the data for Monday at 3:55 PM were missing, I replaced it with the average of data collected on Mondays at 3:55 PM. However, this strategy is less effective for long-term missing data, as it considerably reduces the signal variance and can, above all, propagate bias in the observed data to the missing data.
3. For data missing over very long periods, I used a method called "Seasonal Trend Decomposition Imputation." This involves first performing linear imputation, then decomposing the time series into trend, seasonality, and noise using well-known statistical tools, and finally replacing the initial linear imputation with the data inferred from this decomposition.

3.2. Implementation of xLSTM

Once the imputation was complete, I was able to fully focus on implementing the Extended Long Short Term Memory[3] (xLSTM) in PyTorch. This phase was the longest of my internship due to complex time constraints. When I started working on the xLSTM, the paper had been released for a few weeks, but no official code was available yet. The paper was initially unclear on some details, making it particularly challenging to not say impossible to implement the model accurately.

The few unofficial codes available online often diverged in their interpretations of the missing details. It was not until several months later that an official code was shared. Unfortunately, this code was not functional, but it allowed the community to uncover the missing details. As a result, parallel codes, which were often functional, were developed. After more than two months of work, I was finally able to implement the model in a simple and functional way.

During the implementation, I was surprised to find that no regularization techniques, such as dropout[11], were applied to the model. In fact, none of the parts presenting the limits were present in the paper. Given that the model aims to merge LSTMs[1] and Transformers[6]—both known for their limit tendency to overfit and their need for substantial dropout—I decided to experiment with adding dropout to the architecture. This approach completely disrupted the effectiveness of the stabilizer[4] (cf Figure 8). As a result, I explored ways to incorporate dropout more seamlessly within the architecture itself, which led me to develop the model described below.

3.3. Creation of a new model inspired by xLSTM : xGRU

The inability to add dropout[11] to xLSTM led me to explore methods for integrating this regularization technique natively. As a reminder, dropout involves randomly deactivating certain neurons during training, which forces the model to rely on other neurons and helps prevent excessive specialization, thereby improving generalization.

A common approach to incorporating dropout natively is to reduce the number of model parameters. This can be done by decreasing the size h of the parameter matrices, reducing the dimensions of projections in the blocks, or simply removing certain parameter matrices. I chose the latter option, as it can not only maintain the model's accuracy but also potentially improve it.

To implement this, I considered a model where entire parameter matrices could be removed. Since xLSTM is inspired by LSTM, I naturally turned to Gated Recurrent Units[12] (GRU), which are known to be equivalent to LSTM but with less parameter matrices. You can find a description of GRU operations in the Appendix B.

By leveraging the GRU structure and incorporating the same innovations as xLSTM, I created the Extended Gated Recurrent Units (xGRU). Although I attempted to follow a similar mathematical reasoning as with xLSTM to adapt GRU, but stabilization did not work as expected during the experiments. Consequently, I developed the xGRU empirically.

3.3.1. sGRU : scalarGRU

Assume that

$$\forall t \in [1, T], \quad x_t \mid x_{t-1}, ..., x_1 \sim x_t \mid h_{t-1}$$

where h_{t-1} is a latent variable in $\mathbb{R}^h$ which represents memory.
The training process for scalarGRU is defined as follows:

- Initialization : $h_0 := \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^h$
- $\forall t \in [1, T]$ Forward pass (with $\tilde{f}_t = W_f x_t + U_f h_{t-1} + b_f$ and $\tilde{i}_t = W_i x_t + U_i h_{t-1} + b_i$):
  - $m_t = \max\left(\tilde{f}_t + m_{t-1}, \tilde{i}\right)$
  - $f_t = \exp\left(\tilde{f}_t - m_t + m_{t-1}\right)$
  - $i_t = \exp\left(\tilde{i}_t - m_t\right)$
  - $z_t = \tanh(W_z x_t + U_z(h_{t-1} \odot i_t)) + b_z)$

with $\forall j \in [f, i, z], W_j \in \mathbb{R}^{h \times d}, U_j \in \mathbb{R}^{h \times h}, b_j \in \mathbb{R}^h$ are the deep learning model weights to be optimized during backpropagation (initialized using the chosen method).

- $h_t = (\exp(1) - f_t) \odot h_{t-1} + f_t \odot z_t$

The block of the sGRU is exactly the same as the sLSTM one.

3.3.2. mGRU : memoryGRU

Assume that

$$\forall t \in [1, T], \quad x_t \mid x_{t-1}, ..., x_1 \sim x_t \mid C_{t-1}$$

where C_{t-1} is a latent variable in $\mathbb{R}^{h \times h}$ which represents memory.
The training process for memoryGRU is defined as follows:

- Initialization : $h_0 = m_0 = n_0 = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^h$ and $C_0 = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}_{i \in [1,h]} \in \mathbb{R}^{h \times h}$
- $\forall t \in [1, T]$ Forward pass (by noting $\tilde{i}_t = W_i x_t + b_i$):
  - $m_t = \max\big(1 - \tilde{i} + m_{t-1}, \tilde{i}_t\big)$
  - $f_t = \exp\big(1 - \tilde{i} - m_t + m_{t-1}\big)$
  - $i_t = \exp\big(\tilde{i}_t - m_t\big)$
  - $o_t = \sigma(W_o x_t + b_o)$
  - $v_t = W_v x_t + b_v$
  - $k_t = W_k x_t / \sqrt{h} + b_k$
  - $q_t = W_q x_t + b_q$
  - $n_t = f_t \odot n_{t-1} + i_t \odot k_t$

où $\forall j \in [i, o, v, k, q], W_j \in \mathbb{R}^{h \times d}, b_j \in \mathbb{R}^h$ are the deep learning model weights to be optimized during backpropagation (initialized using the chosen method).

  - $C_t = C_{t-1} \odot f_t + i_t \odot v_k k_t^\top$
  - $h_t = o_t \odot C_t q_t / \max\big(\big|n_t^\top q_t\big|, 1\big)$

The block of the mGRU is exactly the same as the mLSTM one.
Note that all the transformers equations could not be modified because they have been proven to be the optimal covariance update rule[5].

4. Results

4.1. Imputation

To validate the imputation, I conducted two stationarity tests, which are crucial for time series prediction: the Kwiatkowski–Phillips–Schmidt–Shin[13] (KPSS) test and the Augmented Dickey–Fuller[14] (ADF) test. The KPSS test assesses the null hypothesis that the series is stationary, which could not be rejected with a p-value of 96. The Dickey–Fuller test, on the other hand, tests the alternative hypothesis that the time series is stationary, which was confirmed with a p-value of 0.005.

To reassure myself further, I also created a validation set where I remove data, then applied linear imputation, mean imputation (as defined in the contribution section) and my own imputation method. I obtained the following results respectively:

	AVGSpeed	AVGOccupy	TotalVol
Linear Imputation	901	774	742
Mean Imputation	62	57	**82**
Method Imputation	**59**	**32**	94

Table 2. Road *m20317* Imputation Results (MSE)

This is reassuring, and means that my preferred imputation method, compared with mean imputation gives equivalent results but does not accentuate bias or reduce variance. Here are the results graphically.

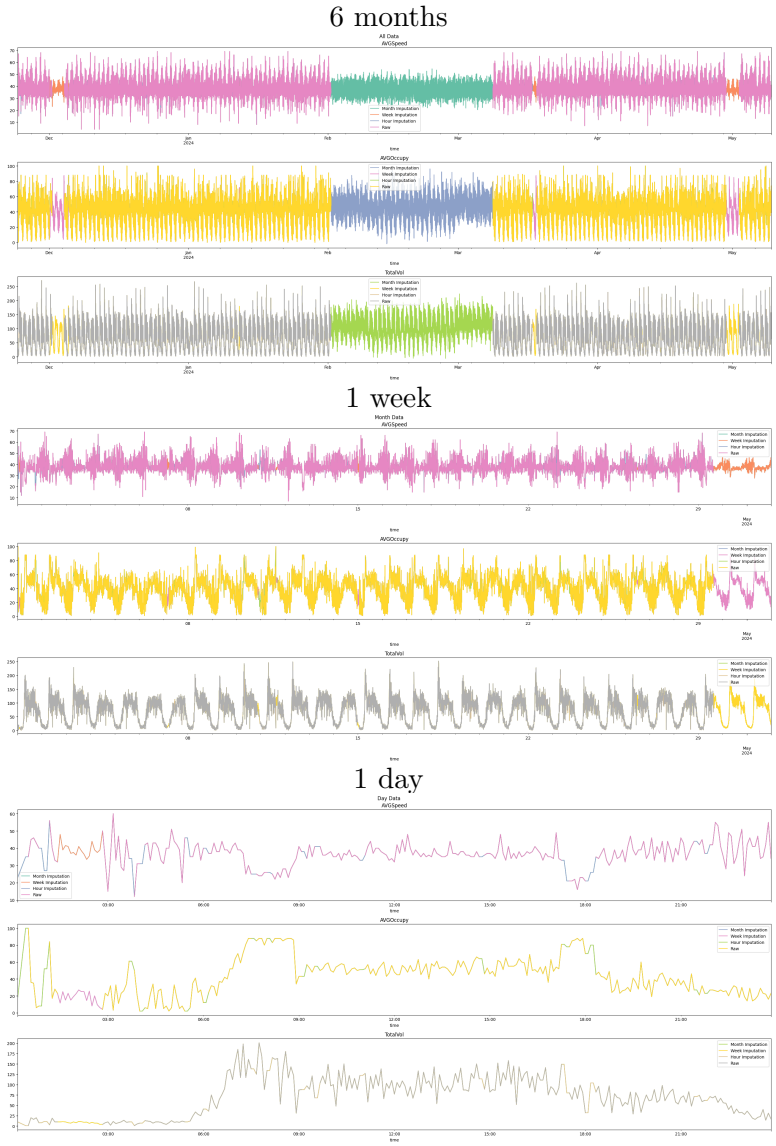

Figure 5. Road *m20317* imputed for 3 time scales (6months, 1week, 1day)

The results are satisfactory, and it is clear that all three types of imputation have been carried out.

4.2. Implementation of xLSTM and xGRU

Each model was trained on an NVIDIA Tesla V100 GPU using the NARLabs supercomputer. To optimize training time, each model was trained on the last 20,000 time steps of the time series (representing 10 weeks), over 100 epochs, with an Adam optimizer, a batch size of 32 and a sequence length of 12 data points (equivalent to 1 hour). I had to restrict the training to 100 epochs because beyond this point, the models became unstable (Figure 9). This instability, noted in the original xLSTM paper, was addressed by enforcing a gradient constraint during descent to ensure it stayed below a certain threshold —a feature I have not yet implemented. The results are presented below with the notation [x:y] for a xLSTM (resp. xGRU) with x mLSTM (resp. mGRU) block layers and y sLSTM (resp. sGRU) block layers.

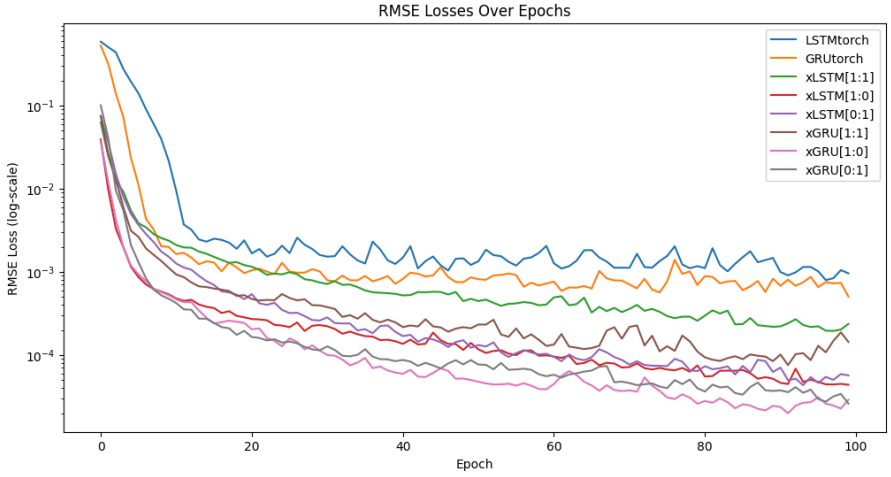

Figure 6. Road *m20317* RMSE Average Losses Over Epochs on Validation Set

We can observe that the performance differences between sGRU (xGRU[0:1]) and sLSTM (xLSTM[0:1]) are quite similar to those between GRU and LSTM. Both sGRU and GRU exhibit a faster initial convergence followed by stabilization, then eventually being matched by sLSTM and LSTM. Additionally, xGRU outperforms xLSTM on the forecasting. Since the results are on a logarithmic scale, the differences are visually minimal, as illustrated in the plot below.

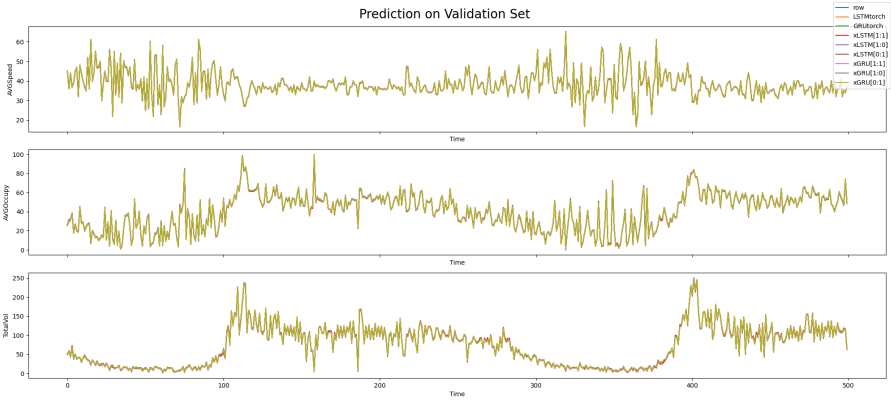

Figure 7. Road *m20317* Predictions on Validation Set

Finally, after spending extensive time on implementation and finding the right training parameters, I was concerned about overfitting by tuning hyperparameters to the validation set. To address this, I took a new dataset from several months on the same road, which had never been used before, and applied the model predictions directly. Because the plot does not add anything, I show the quantitative results directly below.

	LSTM	GRU	xLSTM[1:1]	xLSTM[1:0]
RMSE	0.70	0.51	0.51	0.20

	xLSTM[0:1]	xGRU[1:1]	xGRU[1:0]	xGRU[0:1]
RMSE	0.23	0.25	**0.10**	0.14

Table 3. Road *m20317* Prediction Results on a new Dataset(RMSE)

The results are quite impressive and confirms that xGRU outperforms xLSTM, which outperforms yet LSTM. For the 27 routes, on average, xGRU[1:1], xGRU[1:0], and xGRU[0:1] are 46%, 8%, and 14% lower, respectively, compared to their xLSTM counterparts (xLSTM[1:1], xLSTM[1:0], xLSTM[0:1]) which already far surpasse the LSTM model currently in use.

5. Discussion and Conclusion

The goal of my internship was to improve the traffic prediction systems for Taichung using sensors already deployed in the city by NCHC. The data for each road is in a tabular format without location information. This limitation allows us to focus our methodology on a single road, missing potential causal effects of traffic between different roads. However, the results are satisfactory and generalize well to other roads.

During my internship, I had the opportunity to develop a method for imputing missing data. Given that the situation was not entirely well-defined (MCAR or MNAR), applying a standard methodology was challenging. I acknowledge that there might be more suitable methods that I have not explored yet. However, I believe that the methodology I have presented is simple, effective, and sufficient for the scope of my internship.

I also had the opportunity to work on a newly released research paper on Extended Long-Short Term Memory[3] (xLSTM) models, which aim to replace current Transformers[6]. I thoroughly enjoyed working with such a cutting-edge and recent model. It was exciting to see that others were also exploring this model for new time series prediction [15, 16], making me feel truly involved in the research process. Unfortunately, working with such a new model came with challenges. The official code release was non-functional, further delaying my internship and forcing me to create my own implementation. The paper lacked clarity on some crucial points essential, which made the task of implentation quite difficult. Additionally, the paper did not address the limitations of this new model, and I quickly encountered limitations (Appendix A) when increasing the number of epochs during training or adding regularization techniques like dropout[11]. That is why, It might be interesting to explore statistical learning models.

However, this challenge motivated me to develop my own model, the Extended Gated Recurrent Units (xGRU), which has similar limitations but fewer parameters and delivers superior results, as GRU[12] and LSTM[1]. Unfortunately, the mathematical aspects of the xGRU, inspired by the xLSTM paper, did not align with the experimental performance, forcing me to deviate and conduct more empirical research. This was only possible thanks to the supercomputer of NCHC. Even the results between the xGRU and xLSTM are impressive, they should be interpreted cautiously. Because I may have not coded the xLSTM faithfully, or this difference could also be due to the length of the sequences. Indeed the trend might change with longer sequence lengths, spanning several hours or even days, entering the area of long-term time series forecasting (LSTF). In this context, these new model, especially the mLSTM part with their ability to retrieve data over extended periods through the key-value-query system, could prove to be very meaningful. However, I do not know if stabilization[4] will enable us to reach high lengths, as it does not allow for a large number of epochs. The xLSTM and perhaps the xGRU models will be to know in the future for people passionate about deep learning.

Finally, I am very grateful to DataIA and NARLabs for giving me the opportunity to have this experience. Spending several months in Taiwan was one of the most enriching human and cultural experiences. It was also my first time working in an applied research laboratory rather than a theoretical one. This allowed me to see the direct impact of research on the population, adding a deeper sense of purpose. That made me realize how computer science is poised to revolutionize the world, affecting every geographical area and societal layer. I believe this experience greatly helps in finding one's place and purpose in the research world. I highly recommend this experience to future students of Math&IA.

List of figures and tables

A. Appendix : xLSTM and xGRU limits

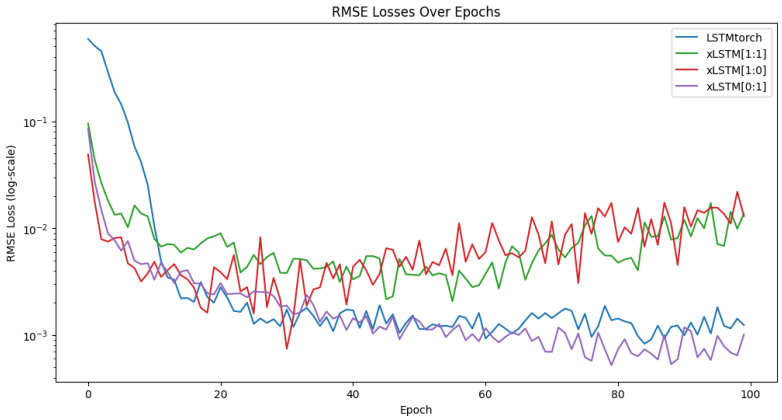

Figure 8. xLSTM efficiency with dropout of 0.1

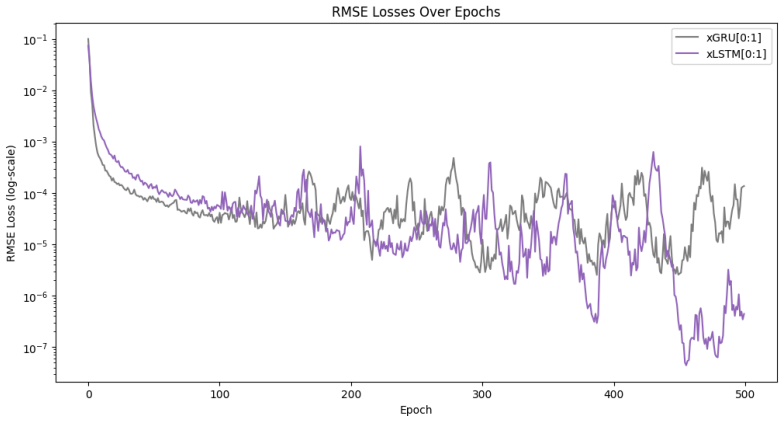

Figure 9. xLSTM and xGRU RMSE over Epochs on Validation Set

B. Appendix : Gated Recurent Unit (GRU)

Assume that

$$\forall t \in [1, T], \quad x_t \mid x_{t-1}, ..., x_1 \sim x_t \mid h_{t-1}$$

where h_{t-1} is a latent variable in $\mathbb{R}^h$ which represents memory.

Learning is a sequential process defined as follows:

- Initialization : $h_0 := \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^h$

- $\forall t \in [1, T]$ Forward pass :

  - $f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$
  - $i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$
  - $z_t = \tanh(W_z x_t + U_z(h_{t-1} \odot i_t) + b_z)$

with $\forall j \in [f, i, z], W_j \in \mathbb{R}^{h \times d}, U_j \in \mathbb{R}^{h \times h}, b_j \in \mathbb{R}^h$ are the deep learning model weights to be optimized during backpropagation (initialized using the chosen method).

  - $h_t = (1 - f_t) \odot h_{t-1} + f_t \odot z_t$

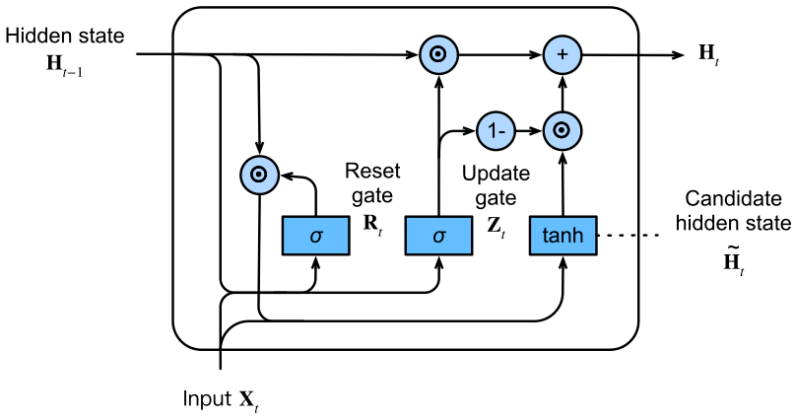

Figure 10. Schematic representation of an GRU

References

1. Hochreiter, S.: Long Short-term Memory. Neural Computation MIT-Press. (1997)
2. Rumelhart, D. E., Hinton, G. E., Williams, R. J.: Learning representations by back-propagating errors. nature. 323, 533–536 (1986)
3. Beck, M., Pöppel, K., Spanring, M., Auer, A., Prudnikova, O., Kopp, M., Klambauer, G., Brandstetter, J., Hochreiter, S.: xLSTM: Extended Long Short-Term Memory. arXiv preprint arXiv:2405.04517. (2024)
4. Milakov, M., Gimelshein, N.: Online normalizer calculation for softmax. arXiv preprint arXiv:1805.02867. (2018)
5. Dayan, P., Willshaw, D. J.: Optimising synaptic learning rules in linear associative memories. Biological cybernetics. 65, 253–265 (1991)
6. Vaswani, A.: Attention is all you need. Advances in Neural Information Processing Systems. (2017)
7. Ba, J.: Layer normalization. arXiv preprint arXiv:1607.06450. (2016)
8. Ramachandran, P., Zoph, B., Le, Q.: Searching for Activation Functions. arXiv preprint arXiv:1710.05941. (2017)
9. Wu, Y., He, K.: Group normalization. In: Proceedings of the European conference on computer vision (ECCV). pp. 3–19 (2018)
10. Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415. (2016)
11. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research. 15, 1929–1958 (2014)
12. Cho, K.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078. (2014)
13. Kwiatkowski, D., Phillips, P. C., Schmidt, P., Shin, Y.: Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root?. Journal of econometrics. 54, 159–178 (1992)
14. Dickey, D. A., Fuller, W. A.: Likelihood ratio statistics for autoregressive time series with a unit root. Econometrica: journal of the Econometric Society. 1057–1072 (1981)
15. Alharthi, M., Mahmood, A.: xLSTMTime: Long-term Time Series Forecasting With xLSTM. AI. 5, 1482–1495 (2024)
16. Gil, G. L., Duhamel-Sebline, P., McCarren, A.: An Evaluation of Deep Learning Models for Stock Market Trend Prediction. arXiv preprint arXiv:2408.12408. (2024)