TFNet: Time and Frequency Modeling for Irregular Multivariate Medical Time Series

A Dissertation Submitted

in Partial Fulfilment of the Requirements

for the Degree of

Master of Technology

in

Computer Science

by

V S Siva Kumar Lakkoju

(Roll No. CS2139)

Under the supervision of

Dr. Swagatam Das

Electronics and Communication Sciences Unit

INDIAN STATISTICAL INSTITUTE

Kolkata

June 17, 2023

DECLARATION

I, **V S Siva Kumar Lakkoju (Roll No: CS2139)**, hereby declare that, this report entitled "**TFNet: Time and Frequency Modeling for Irregular Multivariate Medical Time Series**" submitted to Indian Statistical Institute Kolkata towards partial requirement of **Master of Technology** in **Computer Science**, is an authentic undertaking conducted by me, with the guidance of **Prof. Swagatam Das**, and it has not been used as the foundation for the conferral of any degree or diploma, either at this institution or any other university. I have made genuine efforts to maintain academic integrity and honesty throughout. Proper acknowledgment and citation have been provided for any external information, statements, or results utilized in this work.

Kolkata - 700 108 **V S Siva Kumar Lakkoju**

June 2023

CERTIFICATE

This is to certify that the work presented in this dissertation titled "**TFNet: Time and Frequency Modeling for Irregular Multivariate Medical Time Series**", submitted by **V S Siva Kumar Lakkoju**, bearing the roll number **CS2139**, has been carried out under my supervision in partial fulfillment for the award of the degree of **Master of Technology** in **Computer Science** during the session 2022-23 in the Computer and Communication Sciences Division, Indian Statistical Institute.

To the best of my knowledge, this dissertation or parts of it were not submitted to the award of a degree, diploma, or any academic title anywhere else.

Prof. Swagatam Das

Electronics and Communication Sciences Unit

Indian Statistical Institute, Kolkata

ACKNOWLEDGEMENT

I owe my deep sense of gratitude to my research supervisor Prof. Swagatam Das, ISI, Kolkata for giving me an opportunity to work on this dissertation and for providing me with invaluable guidance, feedback and for providing me with valuable suggestions throughout the period of the project. I am indeed inspired by his attitude and academic discipline which made this dissertation possible.

I am hugely thankful to Faizanuddin Ansari, SRF, ISI Kolkata for his motivation and for helping me at times of difficulty. I am indebted to him for providing me with direction and timely suggestions that helped me towards completing the project. With his technical depth in deep learning methods and in implementing complex models, he provided me with constant help whenever required. I am equally grateful to the Indian Statistical Institute, Kolkata, for allowing me to undertake this project and for providing me with all the required facilities. Lastly, I would like to thank my friends and family for their help and emotional support.

Kolkata - 700 108 **V S Siva Kumar Lakkoju**

June 2023

ABSTRACT

Time Series Classification (TSC) involves assigning a target label based on features involving time series data. TSC arises in a variety of domains, like healthcare, finance, process control, weather pattern prediction, etc. This work is focused on exploiting both frequency and time domains of a time series. Inspired by the TimesNet proposed in [27], which learns multi-periodic variations, we proposed Time-Frequency Network (TFNet), a novel Deep Learning model, and applied it to irregular medical time series data. Earlier methods used either only features captured in the time domain or in the frequency domain. It is difficult to learn both temporal dependencies and understand cyclic or seasonality patterns when analyzed in a single domain. To tackle these limitations, we extend the TimesNet model to perform time domain analysis. Our proposed TFNet achieves an improved performance when applied to in-hospital mortality (IHM) prediction based on 48 hours of ICU stay, on a dataset extracted from Medical Information Mart for Intensive Care (MIMIC-III).

Keywords: *Time Series Classification, TimesNet, MIMIC-III, In-hospital mortality, irregular time series, Time-Frequency Network*

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The history of deep learning dates back to the development of artificial neural networks (ANNs) in the 1940s. The structure and functioning of biological neurons in the human brain inspired the concept of neurons in ANNs. The early years focused on developing basic models and understanding their computational capabilities. In the 1960s and 1970s, significant progress was made in the field of neural networks with the development of the perceptron model by Frank Rosenblatt. After Geoffrey Hinton introduced the backpropagation algorithm in the 1980s, deep learning gained a lot of attention. In 2012, AlexNet [15], a deep convolutional neural network (CNN), won the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [20] and significantly outperformed traditional computer vision approaches; this success illustrated the capabilities of deep learning in image classification and also sparked researchers working on other domains, such as time series, natural language processing, speech recognition, machine translation, and many others. In the coming years, researchers have developed numerous deep learning models that are applied across domains and have shown significant improvement in tackling those problems. RNN-like models, such as LSTM, published

in the 1990s, have been applied to sequential data and have shown state-of-the-art performance, until the introduction of Attention and Transformers, in 2014, which took center stage in different areas with its superior results. Off late, RNN-likes and Transformers are modeled to solve problems like forecasting, classification, etc., based on time series. Time series data occur in many real-world applications like weather forecasting, Electronic Health Records (EHR) [8], next-frame detection in video, pose estimation, electricity load forecasting, etc. One interesting problem with time series data is to classify them by learning good representations. Owing to the success of deep learning and an increase in the scale with which the time series data is available, the deep learning community showed interest in Time Series Classification (TSC) and developed a wide array of models ranging from RNN-like models to applying transformers, to developing more complex domain-specific models which address issues present in a typical time series data, such as missing data, irregular time series etc.

During the same period, the world has seen an increase in the adoption of Electronic Health Record (EHR) systems by hospitals and healthcare facilities, mainly to store patient information digitally and perform administrative tasks. Just In the United States, the percentage of non-federal hospitals equipped with basic digital systems rose from 9.4% to 75.5% between 2008 and 2014 [3]. EHR systems store patient details, demographic data, prescriptions, diagnoses, laboratory test results, clinical notes, microbiology reports, ventilation settings, procedure codes, and so on. Lately, researchers have applied machine learning to several clinical or medical applications such as information extraction, phenotyping, patient de-identification, generating privacy-preserving data, and predictive tasks based on EHR data [22]. However, note that It is to be noted however that applying scoring systems for clinical tasks dates back to the 1950s; notably, the Apgar risk score, which allows

scores based on a predefined chart to evaluate the condition of neonates, was first published in 1952.

Most of the clinical tasks based on EHR data were performed by classical machine learning techniques such as support vector machines (SVM) and random forests until the late 2000s; recently, due to the explosion in the size of EHR data, deep learning approaches have been found suitable to be applied for the clinical tasks. Also, since deep learning learns deep feature representations, the need for hand-engineered features for classical techniques, which would require domain experts, has been mitigated. There are several EHR databases, both public and private, in terms of availability, e.g., Several versions of Medical Information Mart for Intensive Care (MIMIC) and the eICU collaborative research database are two well-known publically available databases. In this work, we perform our experiments on a benchmark dataset extracted from the MIMIC-III database.
MIMIC-III, available on physionet.org [12], [13] clinical database is a relational database generated in a single EHR system from 2001 to 2012; it contains over 40,000 de-identified critical care patients with 60,000 ICU stays and lots of events that include laboratory reports and recorded medical variables. Researchers have been utilizing the database to investigate several tasks, such as predicting clinical outcomes, developing risk prediction models, analyzing treatment patterns, understanding disease progression, and exploring various other aspects of critical care medicine.

In this work, we focus on one such outcome prediction task: In-hospital mortality risk prediction. Such predictions on in-hospital mortality are extensively researched due to their importance in acute care and to improve clinical outcomes of high-risk patients. While most clinical tasks come with the issue of the in-

availability of public benchmarks, Hrayr Harutyunyan et al. has extracted four benchmark datasets from the MIMIC-III [10]. The extraction of such standardized benchmarks has helped various researchers develop newer techniques and perform experiments on those benchmarks. We briefly describe our task and our proposed model in the following sections.

The In-hospital mortality (IHM) task is a binary classification task(target label indicates if the patient died before hospital discharge) that involves predicting mortality based on the first 48 hours of ICU stay; figure 1.1 depicts the IHM task, the majority class indicating the patient has not died has 18342 samples, and the minority class has 2797 samples, which informs that the data is imbalanced.

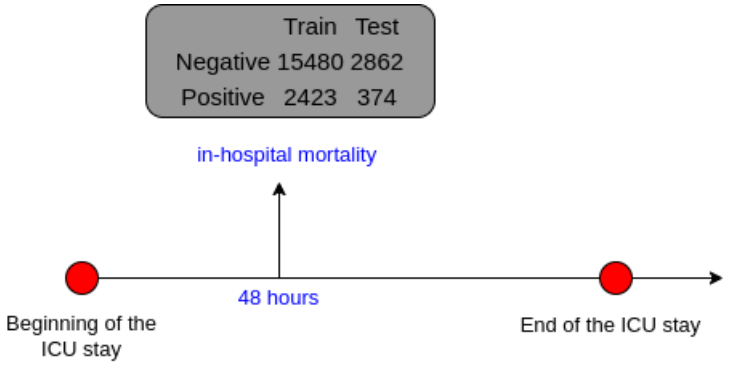

Figure 1.1: Diagram depicting the in-hospital mortality task

The use of both temporal and frequency domain analysis on medical time series is first suggested in UA-CRNN [24]. Inspired by this idea, we propose the Frequency-Time Network or **FTNet**, a novel deep learning model that extracts information in both the time and frequency domain from the multivariate time series. First, we use the TimesBlock [27] module as the component that extracts multi-period variations in the time series by an analysis in the frequency domain. After extracting deep multi-period features, we employ a bidirectional GRU to learn temporal dependencies. Our model reports better recall for the minority

class and improved performance in Area Under Precision-Recall Curve (AUPRC) compared to the baselines. Although existing research compared model performances based on Area Under Reciever Operating Curve (AUC-ROC) scores, due to the presence of class imbalance(13% positive class) in our dataset, AUPRC is more informative of the model's performance, as claimed by Davis et al. in [7]

Outline of the report, in Chapter 2, we briefly discuss various deep-learning approaches applied to the in-hospital mortality prediction task. Chapter 3 begins with a discussion of recent transformer architectures and linear models designed for time series problems. We then discuss in detail the TimesNet architecture [27], which is the building block in our proposed model. Later, the proposed method is then discussed in detail. The chapter that follows includes details of dataset extraction and baseline models. The results obtained by our proposed model and by transformer models and linear models are reported in Chapter 4. We conclude in Chapter 5, discussing possible future directions for the in-hospital mortality prediction task.

Chapter 2

Literature Review

Medical time series data often are irregular with a lot of missing values due to a variety of reasons, since laboratory records and other observations, such as weight, etc., are not recorded at every timestamp or for reducing costs incurred for carrying these observations. In [19], it was observed that missing patterns provide information about the final outcome in tasks like time series classification. Che et al. in [4] term this type of missingness "informative missingness."

Previously lot of approaches were developed to tackle the missing value issue in time series [21]. One solution to address this issue is to simply ignore the data with missing observations and perform the analysis on the observed data. However, this results in a loss of data, especially when the missingness pattern contributes to the target label. Then approaches to fill the missing data, with various methods viz. Interpolation of observed values [14], kernel methods [18], the use of EM algorithm [9], but these approaches don't explore the missing patterns effectively, and often imputations are in contrast to the predictive model, resulting in poor predictions [26].

Lipton et al. in [16], and Choi et al. in [6] tried to resolve the issue of missing values by modifying RNN-like models LSTM and GRU to encode the missing patterns by concatenating missing value masks with the inputs. But directly concatenating the missing masks does not give enough information to the model about the exact location of the missing value in the time series. To better encode the missing mask into the model, Hrayr et al. in [10] have tried to capture missingness patterns with the use of masking information, which is fed alongside the actual time series in a channel-wise LSTM architecture that treats features independently and finally concatenates learned representation for classification. However, this way of treating features independently results in the loss of correlation information among features. Hrayr et al. also proposed a multi-task LSTM model, suggesting that solving multiple clinical tasks simultaneously would be a promising direction to solve individual tasks effectively. On similar lines to the multi-task models, Purushotham et al. in [17] proposed a Multimodal Deep Learning Model (MMDL) that learns shared representations by using an ensemble of feed-forward networks and GRU. The main idea of MMDL is to capture correlations across multiple modalities.

To effectively capture missing patterns, Che et al. in [4] proposed GRU-D, a deep learning model based on the GRU. GRU-D (D for decay mechanism) captures informative missingness with the use of time interval, which encodes the difference between timestamps alongside masking information. The authors noticed that if the last observation was made long ago, the missing value tends to be closer to that last observed value, especially in the case of electronic health records (EHRs).

In 2020 Tan et al. [24] proposed Uncertainty-Aware Convolutional Recurrent Neural Network (UA-CRNN). They first use the Gaussian process to convert irregular time series to regular time series and to estimate uncertainty scores. Once regular time series is obtained, a UA-CRNN is trained in an end-to-end manner. The main components of the UA-CRNN are the Uncertainty Aware Decomposition Layer (UADL) which uses high and low pass filters to extract high-frequency, low-frequency, and mid-frequency components. These components are adjusted based on the uncertainty scores. To the output of UADL, the Residual Network (ResNet) is applied, which captures deep sequential features. Finally, they employed a GRU-RNN to extract temporal features.

In the remainder of this section, we briefly describe some latest state-of-the-art transformers and the DLinear model. These models were earlier not applied to the in-hospital mortality task prediction; we trained the models with our dataset and compared them with our proposed method. **Time Series Transformer** (TST) [30] uses the original transformer [25] encoder to perform time series tasks with two main modifications first, one by replacing the deterministic positional encodings based on sinusoidal with a learnable positional encoding and second, instead of using layer normalization, batch normalization is employed. Zereveas et al. claim that learnable positional encoding performs better for tasks involving time series, and the use of batch normalization eliminates the effect of outliers in the data. Traditional transformers have shown state-of-the-art performance on various time series tasks, like classification and forecasting, but they are unable to capture global features in the time series (for e.g., overall trend, seasonality, etc.). Also, they come with an expensive computational bottleneck (due to $O(n^2)$ self-attention). To solve both these issues, **FEDformer** combines the seasonal-trend decomposition method with the Transformer. The idea is that the decomposition

method captures global features of the time series while transformers capture temporal structures. Also exploiting the fact that time series comes with frequency basis (by Fourier transform), a frequency-enhanced transformer is proposed by Zhou et al. in [31]. The proposed Frequency Enhanced Decomposed Transformer (FEDformer) has only a $O(n)$ time complexity in terms of the input sequence length.

With the idea to improve the performance of long-term forecasting of time series, Haixu Wu et al. in [28] proposed the **Autoformer**, which is based on the decomposition of periods in the time series by using an Auto-correlation block, trained with the deep architecture. The Auto-correlation layer performs dependency discovery at the sub-series level, exploiting the periodicity in the time series, further performing aggregation of the identified discoveries. The authors claim that Autoformer is efficient in both time-complexity compared to other transformer-based models, which are of the order n^2 and improves long-term forecasting performance. There are two primary components of the Autoformer one is the series decomposition block which essentially tries to extract the trend-cyclicity and seasonality components of a time series, while trend-cyclicity explains how the time series could progress in the long term, seasonality captures recurring patterns over a period say, a year. The authors claim, however, that since the future is not known, direct decomposition is not a possibility. To effectively address this issue, a series-decomposition block is used, which with the help of moving average, smoothens out periodic fluctuations and, over time, extracts stationarity from intermediate hidden states. The other component is the Auto-Correlation mechanism which aggregates identical sub-series by time-delay aggregation using series auto-correlation scores.

Ailing Zeng et al., with the following question "Are Transformers Really Effective for Time Series Forecasting?" claim that the permutation invariant self-attention mechanism in the transformers would lose temporal information due to it being anti-order. Although positional encoding adds some temporal ordering to the tokens, applying self-attention to the sequence would inherently mean the loss of order information which may not be that serious of an issue in NLP but would result in poor capture of temporal features in time series. With this discussion, Ailing Zeng et al. proposed a set of one-layer linear architectures in [29] viz. simple one linear layer, NLinear, and **DLinear** to solve long-term time series forecasting (LTSF). Vanilla linear, or simple one linear layer, is just one hidden layer applied to input at each time stamp. DLinear is an extension of the Vanilla linear with a decomposition layer that extracts trend and seasonality-components from the raw input with the help of a moving average kernel, and these two components are separately transformed with two one-layer linear modules, the outputs of which are summed up for the final prediction.

Chapter 3

Proposed Method

3.1 Time-Frequency Network (TFNet)

Inspired by the TimesNet proposed in [27] by Haixun Wu et al., we propose a Time-Frequency Network (TFNet) that performs a 2-fold feature extraction on the multivariate time series. First, we obtain 2D features with the use of the TimesBlock [27] module to learn multi-period variations in the time series by analyzing the time series in the frequency domain. These 2D features are then projected back to 1D space with the use of an Inception Block. We then employ a bidirectional Gated Recurrent Unit to capture temporal dependencies derived from the learned multi-period features. Basically, we analyze the multivariate time series in the frequency domain first and then extract temporal features from the multi-period representations. In the remaining section, we provide a comprehensive description of each component of our model.

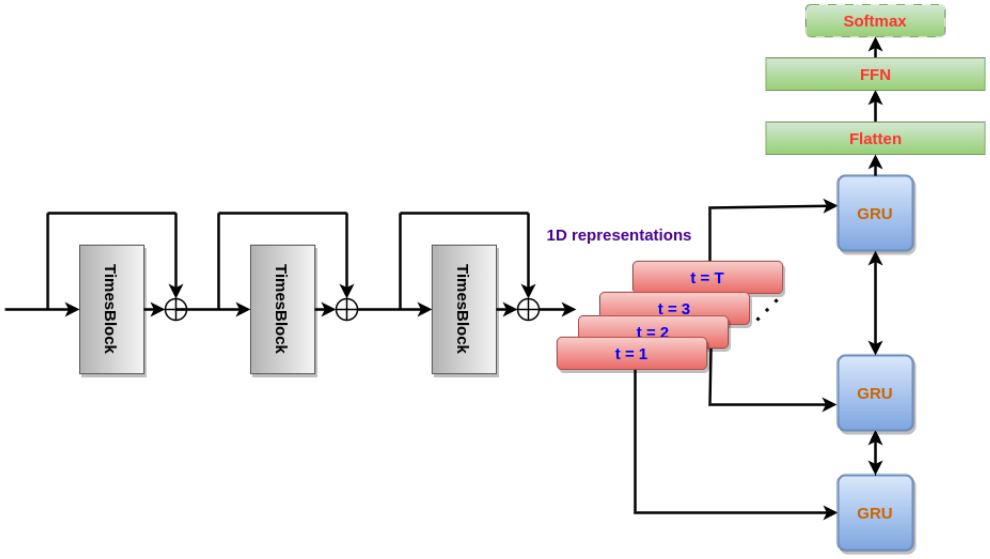

Figure 3.1: Our proposed model, Time-Frequency Network(TFNet)

3.1.1 Fast Fourier Transform

The Fourier analysis states that any signal can be converted into the frequency domain by Fourier transform and back into the time domain by Inverse Fourier transform. In the case of time series signals, analyzing them in the frequency domain can help capture better seasonality and cyclicity information. Fast Fourier Transform (FFT) is an algorithm to perform the discrete Fourier transform (DFT) that decomposes a time series to obtain different frequency components. FFT reduces the number of required computations and thus is faster in terms of time complexity $O(N \log N)$ compared to DFT, which is of $O(N^2)$ for data of size N. FFT decomposes the signal into even and odd subsequences based on indices in a divide and conquer manner, this way DFT can be applied to both the subsequences

simultaneously, thus resulting in faster computation.

$$
\begin{aligned}
x[t] &= \sum_{i=0}^{T-1} x[n] * e^{\frac{-j.2\pi.tn}{T}} \\
x_{even}[t] &= \sum_{k=0}^{\frac{T}{2}-1} x[2k] * e^{\frac{-j.2\pi.t(2k)}{T}} \\
x_{odd}[t] &= \sum_{k=0}^{\frac{T}{2}-1} x[2k+1] * e^{\frac{-j.2\pi.t(2k+1)}{T}} \\
x_{even}[t] &= \sum_{k=0}^{\frac{T}{2}-1} x[2k] * e^{\frac{-j.2\pi.tk)}{T/2}} \\
x_{odd}[t] &= e^{\frac{-j.2\pi.t}{T}} \sum_{k=0}^{\frac{T}{2}-1} x[2k+1] * e^{\frac{-j.2\pi.tk}{T/2}} \\
x[t] &= x_{even}[t] + x_{odd}[t]
\end{aligned}
\tag{3.1}
$$

3.1.2 TimesBlock

In 2023, Haixu Wu et al. proposed TimesBlock as a modular component of the TimesNet [27]. The TimesBlock extracts multi-period patterns by using a convolutional block by first converting the 1D time series into a 2D space and then projecting these 2D features back to 1D space. First, by applying FFT on the input multivariate time series $X_{1D} \in \mathbb{R}^{T \times C}$, frequencies and their corresponding amplitudes are computed.

$$
\begin{aligned}
\mathbf{A} &= Avg(Amp(FFT(X_{1D})), \{f_1, ..., f_k\} \\
&= \arg_{(f \in 1..[\frac{T}{2}])} Topk(\mathcal{A}), p_i = \left\lceil \frac{T}{f_i} \right\rceil \text{for i} \in \{1, .., k\}
\end{aligned}
$$

where, $Amp(.)$ denotes amplitude calculations. The top K frequencies (i.e., the largest frequency components) are chosen to obtain 2D tensors where the columns represent intra-period variation, and rows represent inter-period variation. These 2D tensors are then used to extract local representations by a parameter-efficient inception block. The obtained 2D tensors are then projected back to 1D space to proceed with the adaptive aggregation. For TimesNet, multiple TimesBlocks are stacked layer by layer for the l^{th} layer:

$$X_{1D}^{l} = TimesBlock(X_{1D}^{l-1} + X_{1D}^{l-1})$$

Here $X_{1D}^{l-1} \in \mathbb{R}^{T \times d_{model}}$ is the input from the l-1^{th} TimesBlock. Layer Normalization is applied to the output of each TimesBlock layer. In the remainder of the section, we describe each of the components in the TimesBlock.

The Embed Layer

The Embed Layer has two components, token embedding that transforms input to deep features using a 1D convolutional layer and a positional embedding similar to the positional encoding of the original Transformer [25], that uses two sinusoidal waves to add relative positional information to the input sequence. $X_{emb,1D} = Embed(X_{1D}),$ where $X_{1D} \in \mathbb{R}^{T \times C}$ is length-T 1D time series and $X_{emb,1D} \in \mathbb{R}^{T \times d_{model}}$

Inception Block

Inception block is the idea originally proposed in [23] by Szegedy et al. They proposed the use of a 2-layer convolutional network with residual connections to

learn better local sparse structures from the learned Inception Block uses multiple convolutional filters of varying sizes viz. 1×1, 3×3, 5×5, and so on, shared for each of the k tensors obtained after FFT and 2D projection.

---

for *i:* 0 to num_kernels **do**
  CONV2D(in, out, kernel = 2i + 1, padding = i)(x)
end for

Adaptive Aggregation

With an idea similar to the Auto-Correlation mechanism by Haixu Wu et al. [28], the amplitudes obtained initially by applying FFT reflect the relative significance of their corresponding frequencies. This also means that these amplitudes show us the relative significance of the k 2D tensors, which represent intra-period and inter-period periodic patterns. Thus the authors suggest an aggregation technique based on the amplitudes.

$$\hat{A}^{l-1}_{f_1}, ..., \hat{A}^{l-1}_{f_k} = Softmax(A^{l-1}_{f_1}, ..., A^{l-1}_{f_k})$$

$$X^{l}_{1D} = \sum_{i=1}^{k} \hat{A}^{l-1}_{f_i} \times \hat{X}^{l,i}_{1D}$$

Since the intra-period and inter-period variations are captured in the 2D tensors, TimesBlock can effectively model multi-scale temporal 2D variations simultaneously. TimesNet is constructed by combining all the modules mentioned above and thus captures multiperiod temporal variations effectively. Overall the following equations summarize the TimesBlock:

$$A^{l-1}, \{f_1, ..., f_k\}, \{p_1, ..., p_k\} = Period(X_{1D}^{l-1})$$
$$X_{2D}^{l,i} = Reshape_{p_i,f_i}(Padding(X^{l-1})), i \in 1, .., k$$
$$\hat{X}_{2D}^{l,i} = Inception(X_{2D}^{l,i})), i \in 1, .., k$$
$$\hat{X}_{1D}^{l,i} = Trunc(Reshape_{1,(p_i \times f_i)}(\hat{X}_{2D}^{l,i})), i \in 1, ..., k$$

$X_{2D}^{l,i} \in \mathbb{R}^{p_i \times f_i \times d_{model}}$, is i^{th} transformed 2D tensor. Then a parameter-efficient Inception block is employed on the 2D tensor. The learned 2D features $\hat{X}_{1D}^{l-1} \in \mathbb{R}^{T \times d_{model}}$ are then projected back to 1D space, $X_{1D}^{l-1} \in \mathbb{R}^{T \times d_{model}}$

3.1.3 Bidirectional GRU

To solve problems involving sequential data, such as text, next frame detection in videos, time series forecasting, and classification. When the sequences are too long RNNs suffer from an inherent problem of vanishing and exploding gradients. To address the gradient flow issues Long Short Term Memory(LSTM) network is proposed in [11] by adding gating mechanisms to the RNN cell that help keep the gradient well in range during backpropagation. Later in 2014, Gated Recurrent Unit (GRU) [5] addressed the vanishing and exploding gradient problems by controlling the information flow with gates in a manner similar to the LSTMs, but with lesser parameters. Bidirectional GRU is essentially 2 GRUs, one applied on the input sequence in the forward order and the other applied on the input sequence in the backward order, and the final output is the concatenation of the outputs from the two GRUs. The idea of using a bidirectional GRU is so that it captures contextual information for a given time from both directions. Below are the equations that describe the working bidirectional GRU cell for a given

timestep.

$$
\begin{aligned}
\overrightarrow{z}_t &= \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \\
\overrightarrow{r}_t &= \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \\
\overrightarrow{h}_t &= \tanh(W_h \cdot [\overrightarrow{r}_t \odot h_{t-1}, x_t] + b_h) \\
\overleftarrow{z}_t &= \sigma(\overleftarrow{W}_z \cdot [\overleftarrow{h}_{t+1}, x_t] + \overleftarrow{b}_z) \\
\overleftarrow{r}_t &= \sigma(\overleftarrow{W}_r \cdot [\overleftarrow{h}_{t+1}, x_t] + \overleftarrow{b}_r) \\
\overleftarrow{h}_t &= \tanh(\overleftarrow{W}_h \cdot [\overleftarrow{r}_t \odot \overleftarrow{h}_{t+1}, x_t] + \overleftarrow{b}_h) \\
h_t &= \overrightarrow{h}_t \oplus \overleftarrow{h}_t
\end{aligned}
$$

3.1.4 Classfication Head

The obtained time-frequency representations are flattened to a 1D tensor which is projected to 2 dimensions to finally classify with a softmax. In the equations below, $\mathcal{P}$ denotes the final learned representations after flattening. $\hat{P}$ denotes the projection to 2 dimensions feeding $\mathcal{P}$ to a Linear layer. Finally, we obtain $\hat{Y}$ by applying softmax, whose elements are the probabilities of belonging to a particular class. We finally predict the class by taking an arg-max of $\hat{Y}$.

1D tensor that represents time and frequency level features: $X \in \mathbb{R}^{d_{model} \times seq_{len}}$

$$
\begin{aligned}
\mathcal{P} &= Flatten(\mathcal{X}), \mathcal{P} \in \mathbb{R}^{d_{model} * seqlen} \\
\hat{P} &= Linear(\mathcal{P}), \hat{P} \in \mathbb{R}^2 \\
\hat{Y} &= Softmax(\hat{P})
\end{aligned}
$$

Chapter 4

Experiments and Results

4.1 Details of the dataset

One of the main contributions of [10] is to create benchmarking datasets for four different tasks on the MIMIC-III database, one being the in-hospital mortality benchmark. To create the current benchmark dataset, all hospital admissions who were transferred between different ICU wards were excluded from the database initially. Also, since the physiology of adults and those under the age of 18 vary significantly, patients under 18 are also excluded from the study. Finally, a cohort of 33,798 unique patients having a total of 42,276 ICU stays and 17 laboratory parameters were chosen as features.

From the above cohort, to extract the IHM dataset, all ICU stays for which no event is recorded within the initial 48 hours spent in the intensive care unit (ICU) stay or for which the length-of-stay is not known or if the ICU stay is less than 48 hours are not considered, since the current IHM task defined is based on the

observations made during first 48 hours of the ICU stay. The final benchmark dataset contains 17903 and 3236 ICU stays in the train and test set, respectively. The target label for each ICU stay is obtained by comparing the patient death date with the hospital admission and discharge times recorded. The dataset, which includes a total of 21,139 ICU stays, has a mortality rate of 13.23% or 2,797 out of 21,139. Train and test sets are in the ratio 85% and 15%, respectively. The figure 4.1 below describes the entire process from cohort selection to data extraction; please note that subjects denote patients, events denote any clinical assessment, treatment, or laboratory results recorded, and episodes refer to ICU stays. All the categorical variables, as presented in 4.3, are one-hot encoded. So, we finally get an input $X \in \mathbb{R}^{48\times59}$, where the 59 variates in each of the 48-time steps represent our final feature set, and the binary missing mask $M \in \mathbb{R}^{48\times17}$, 1 representing that a value observed.

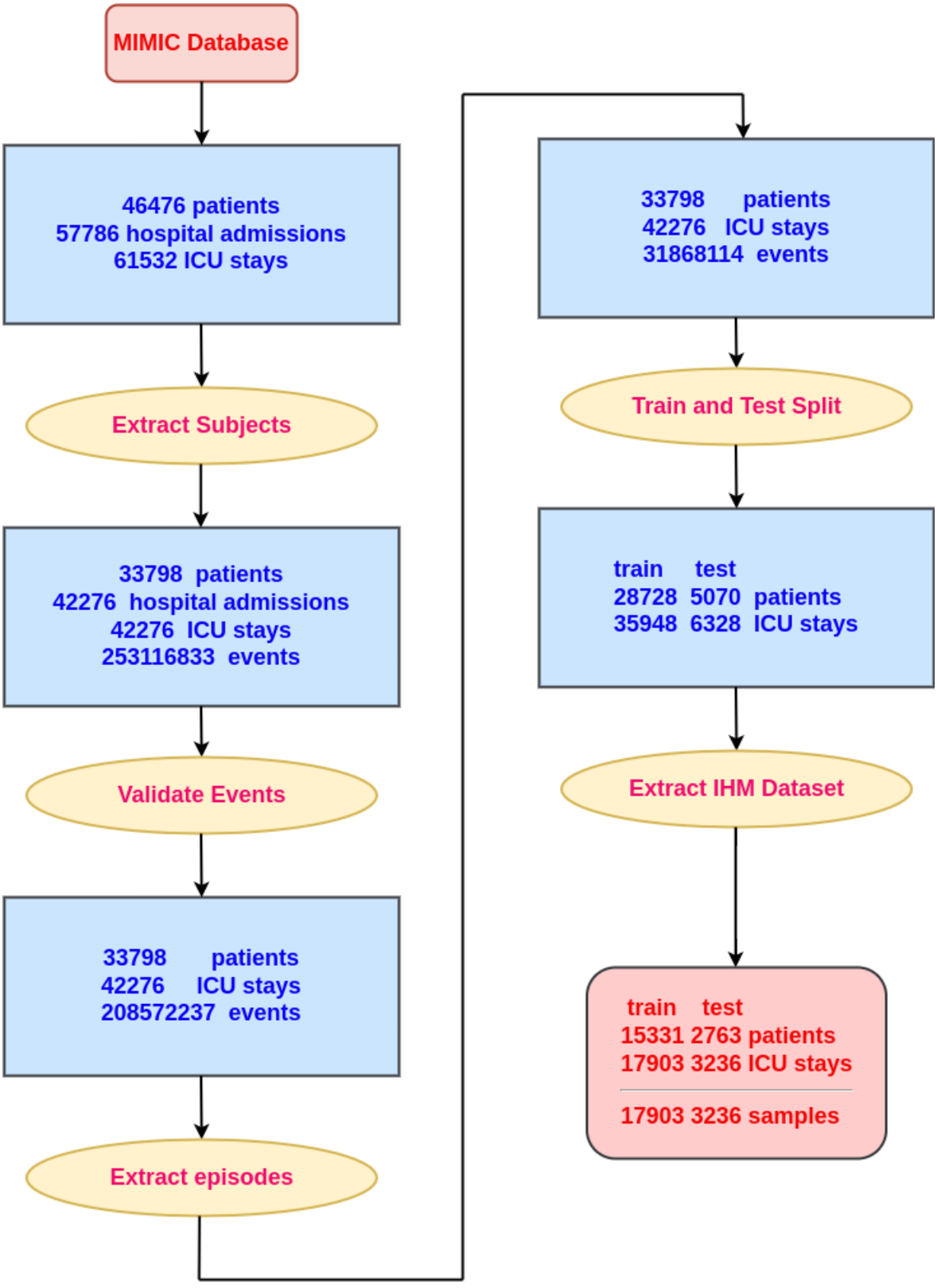

Figure 4.1: The cohort selection and dataset extraction process [10]

4.1.1 The need for data imputation

Figure 4.1 below shows the percentage of missing values for each of the 17 features. To address the missing value problem, the authors of [10] have proposed previous value imputation, where possible, and imputation with normal values as mentioned in 4.3 otherwise. Although there's not much of a discussion on the rationale behind choosing such values as normal values, we believe Hrayr et al. have considered possible feature ranges for, e.g., the weight of a patient can not be in the order of 400kgs except maybe in cases which are rare and chosen a value within that range.

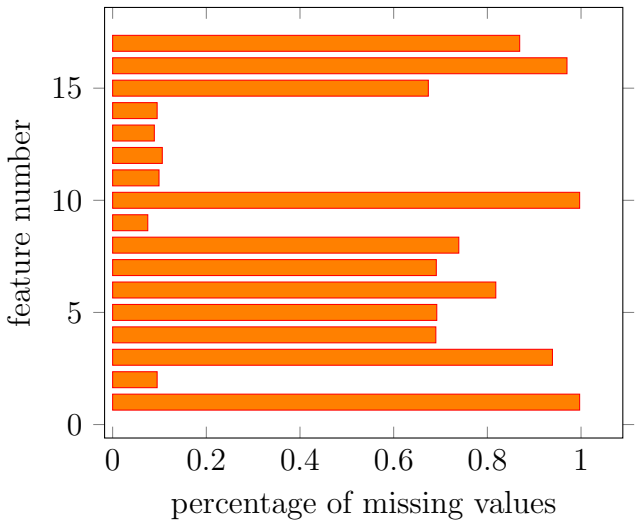

Figure 4.2: Percentage of missing values for each of the 17 variables, calculated for all 48 hours

Variable	Impute value	Modeled as
Capillary refill rate	0.0	categorical
Diastolic blood pressure	59.0	continuous
Fraction inspired oxygen	0.21	continuous
Glascow coma scale eye-opening	4 spontaneously	categorical
Glascow coma scale motor response	6 obeys commands	categorical
Glascow coma scale total	15	categorical
Glascow coma scale verbal response	5 oriented	categorical
Glucose	128.0	continuous
Heart Rate	86	continuous
Height	170.0	continuous
Mean blood pressure	77.0	continuous
Oxygen saturation	98.0	continuous
Respiratory rate	19	continuous
Systolic blood pressure	118.0	continuous
Temperature	36.6	continuous
Weight	81.0	continuous
pH	7.4	continuous

Figure 4.3: Normal values used to impute[10]

4.2 Experimental setup

All the models are trained on an RTX 3090 GPU. We trained all the transformer baselines and our proposed method, including the ablations for 30 epochs, using a batch size of 8. DLinear, however is trained for 100 epochs. We employed EarlyStopping based on the validation accuracy with a tolerance of 10 and saved the best model for prediction on the test set. All the models are trained using the Adam optimizer with an initial learning rate that is equal to 0.0001. Thereafter for every 5 epochs, we use a learning rate scheduler $lr * (0.5)^{(epoch\_num-1)}$. For TFNet

and its modifications, we took K=3 to select the top K frequencies, i.e., the top K largest frequencies, after FFT. For models using TimesBlock, the number of layers of TimesBlock are chosen to be 3, except for our proposed method, which has 5 layers of TimesBlock. The number of kernels for the Inception Block was chosen to be 6, and they are initialized with Kaiming-Uniform initialization. Wherever a dropout is employed, the dropout rate is set to 0.1. All layers use GELU activation. The dimension of the model, d_{model}, is taken to be 32 for all TimesBlock-based models and 128 for transformer baselines and DLinear. Transformer baselines use 3 layers of transformer encoder.

4.3 Baseline models

4.3.1 Logistic Regression with regularization

Logistic regression is a fundamental machine learning algorithm used for binary classification tasks. It works by employing a linear combination of input features and their associated weights. This linear combination is then passed through a sigmoid function, also known as the logistic function. The sigmoid function maps the output to a range between 0 and 1, interpreting it as the probability of belonging to one of the classes. During the prediction phase, if the probability exceeds a threshold, the data point is classified as belonging to the positive class; else, it is assigned to the negative class.

Hrayr et al. in [10] used 714 ($17 \times 6 \times 7$) hand-engineered features per time series as the input to the logistic regression model. First, 7 subsequences of the original sequence are constructed from the original sequence, using features of the first 10% of time, first 25% of time, first 50% of time, last 50% of time, last 25% of

time, and last 10% of time and the full sequence. Then for every feature in the 7 subsequences, six sample statistics, viz. mean, median, maximum, minimum, skewness, and standard deviation, are calculated. If one sample statistic is not measurable, its corresponding feature is marked as missing. A logistic regression with l_2 regularization is then trained on the 714 features after they are standardized.

4.3.2 LSTM

LSTM is a type of recurrent network that captures long-range dependencies effectively in sequential data. It was proposed in 1997 by Sepp et al. in [11] to solve the vanishing and exploding gradient problems in the vanilla RNNs. The authors introduce different gates, viz. input gate, forget gate, and output gate, that makes sure constant error is always propagated. It takes in an input x_t

$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
\tilde{C}_t &= \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\
C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \\
h_t &= o_t \odot \tanh(C_t)
\end{aligned}
$$

4.3.3 Channel Wise LSTM

Instead of analyzing the time series as multivariate data, the authors proposed a channel-wise LSTM that treats each of the features independently of each other

and trains a bidirectional LSTM on individual features + the missing mask of that feature. The authors claim that the channel-wise module has two advantages; first, it can learn useful features about a single variable before combining it with the other data. Although LSTM models were also given as input both the actual data and the missing mask, their ability to learn something meaningful from the missing mask is poor since they may not know which mask belongs to with feature. Thus comes the second advantage of the channel-wise LSTM since the mask information is also an input for each channel it can learn better features from the data masks.

$$
\begin{aligned}
p_t^{(i)} &= LSTM\left(\left[\mu_t^{(i)}, c_t^{(i)}\right], p_{t-1}^{(i)}\right) \\
q_t^{(i)} &= LSTM\left(\left[\overleftarrow{\mu}_t^{(i)}, \overleftarrow{c_t}\right], q_{t-1}^{(i)}\right) \\
u_t &= \left[p_t^{(1)}, q_t^{(2)}, ..., p_t^{(17)}, q_t^{(17)}\right] \\
h_t &= LSTM\left(u_t, h_{t-1}\right)
\end{aligned}
$$

Deep Supervision

Unlike models that do prediction as the last step, deep supervision enforces the model to replicate the target at each time step, this way of replicating contributes to the total gradient flow at each time step. Since the loss is now calculated at all time steps, for a given time step, gradient flows both due to the loss at the given timestep and losses from the future time steps. So overall loss function is as follows:

$$
\mathcal{L} = (1-\alpha) * CE\left(y, \hat{y}_T\right) + \alpha * \frac{1}{T}\sum_{t=1}^{T} CE\left(y, \hat{y}_T\right)
$$

In the above loss function, CE denotes cross-entropy loss, and $\alpha \in (0, 1)$ controls how much different losses would contribute to the final loss. y denoted true label, $\hat{y}_T$ denotes predicted label at time step T.

4.4 Results

4.4.1 Discussion

We compared our proposed work with baseline logistic regression and LSTM-type models; the results are reported in table 4.1. Compared to the best baseline, the CW-LSTM, our model reported a 15.5% improvement on the recall of positive class, i.e., $0.3449 \rightarrow 0.3984$. Since the task at hand is related to the medical field, we believe a high recall for the minority class is desirable since a higher number of positive data are reported positively, meaning fewer false negatives overall.

Compared to CW-LSTM with AUPRC 0.5064, our model reported an AUPRC of 0.5172, an improvement by 2% over the best model reported by Hrayr et al. in [10], the CW-LSTM(except for multi-task models based) on AUROC. Jesse Davis and Mark Goodrich in [7] suggested that although AUROC is usually used to report performance in binary classification, for highly skewed data, and for applications where the positive class is critical, AUPRC gives more information on the model's overall performance because it focuses on the precision-recall tradeoff; it provides a nuanced evaluation by measuring how accurately the model would perform in classifying the positive class(the rare class). On the other hand, AUROC offers a global assessment of the model's ability to classify both classes for all thresholds. An improvement on AUPRC doesn't necessarily improve AUROC, especially in case of class imbalance. This phenomenon is identified in our dataset since our model reports a poorer AUROC than the baselines.

For classification, usually accuracies are estimated to check the model's performance however, when evaluating a classifier on an imbalanced dataset, it can lead

to an optimistic estimate of its performance. One way to address this issue is by replacing accuracy with balanced accuracy, defined as the average between recalls of both classes [1]. Thus, in our work, balanced accuracy is reported instead of accuracy, and our model has shown a 2% improvement on this metric. Our model slightly improved the precision of the majority class.

Table 4.1: Performance of our proposed method compared with baselines [10]

model	bAcc[1]	AUROC	AUPRC	min(P$^+$Se)	Precision_0	Precision_1	Recall_0	Recall_1
Logistic Regression [2]	0.6441	0.8483	0.4811	0.4652	0.9156	0.648	0.978	0.3102
LSTM	0.634	0.8528	0.4891	**0.5**	0.9134	0.6193	0.9766	0.2914
LSTM-DS [3]	0.6402	0.8571	0.4858	0.4868	0.9147	0.6457	0.9783	0.3021
CW-LSTM	0.6565	**0.8632**	0.5064	0.492	0.9188	0.5864	0.9682	0.3449
CW-LSTM-DS [3]	0.6003	0.86	0.5153	0.484	0.9057	**0.7248**	**0.9895**	0.2112
TFNet	**0.6793**	0.8392	**0.5172**	0.4872	**0.9231**	0.5665	0.9602	**0.3984**
Improvement	**3.46%**	-	**2.13%**	-	**0.46%**	-	-	**15.51%**

[1] bAcc, denotes balanced accuracy here and thereafter wherever it is mentioned
[2] Logistic Regression is regularized by l_2 regularization with $\lambda = 0.0001$ [3] DS denotes Deep Supervision

4.5 Ablation Study

4.5.1 Imputation and transformer baselines

We have performed ablation studies analyzing how various imputation techniques would affect the model performance namely, we performed three types of imputation:

- Carry forward imputation
  - For carry forward imputation, where present, we have imputed missing values with observations available in a previous time step till the next available observed value. In case there is no previous observation available, we have imputed with normal values [10]

- Carry-backward imputation
  - For carry backward imputation, where present, we have imputed missing values with observations available from a future time step until an observed value is present in an earlier time step. In case there is no future observation available, we have imputed with normal values [10]
- Imputation with normal values
  - Normal values are values reported in [10]; they consider the possible feature ranges and have chosen a value that results in the best performance of LSTM-type baselines.

With the different imputations, we have also reported results on some recent transformer models viz. TSTransformer [30], Autoformer [28], etc., as well as DLinear [29], which is a simple linear model, and compared with our proposed model. Almost all the models report their best performance when carrying forward imputation is performed. Carry-backward imputation results in the worst performance for each model, validating the claim of Che et al. [4] that values are closer to the previously observed value. Interestingly DLinear [29] showed better performance than the transformer baselines in terms of AUROC, AUPRC, and recall of positive class.

Table 4.2: Results for carry forward imputation

model	**bAcc**	**AUROC**	**AUPRC**	**min(P$^+$Se)**	**Precision_0**	**Precision_1**	**Recall_0**	**Recall_1**
DLinear	0.6477	0.8469	0.5106	0.4813	0.9165	0.6218	0.9745	0.3209
Transformer BN	0.6388	0.8243	0.4317	0.4347	0.9148	0.5367	0.9647	0.3128
Autoformer	0.5614	0.7133	0.3032	0.2895	0.8977	0.3466	0.9598	0.1631
FEDformer	0.6173	0.8228	0.4612	0.4465	0.9097	0.5774	0.9752	0.2594
TimesNet	0.6469	0.8405	0.5133	0.4587	0.9157	0.6297	0.9755	0.3183
TFNet	0.6793	0.8392	0.5172	0.4827	0.9231	0.5665	0.9602	0.3984

Table 4.3: Results for normal value imputation

model	bAcc	AUROC	AUPRC	min(P$^+$Se)	Precision_0	Precision_1	Recall_0	Recall_1
DLinear	0.5948	0.8245	0.4455	0.4465	0.9046	0.5909	0.9811	0.2086
Transformer BN	0.6118	0.8073	0.4105	0.4173	0.9084	0.5897	0.9776	0.246
Autoformer	0.5639	0.7204	0.2864	0.2956	0.8985	0.2917	0.9406	0.1872
FEDformer	0.6197	0.8009	0.4538	0.4305	0.9103	0.5556	0.972	0.2674
TimesNet	0.6385	0.8259	0.4772	0.4346	0.9143	0.5672	0.9696	0.3074
TFNet	0.6452	0.8362	0.4797	0.4577	0.9192	0.5581	0.9668	0.3236

Table 4.4: Results for carry backward imputation

model	bAcc	AUROC	AUPRC	min(P$^+$Se)	Precision_0	Precision_1	Recall_0	Recall_1
DLinear	0.6286	0.8405	0.4872	0.4733	0.9122	0.6105	0.9766	0.2807
Transformer BN	0.6325	0.8253	0.4234	0.4385	0.9135	0.516	0.963	0.3021
Autoformer	0.5264	0.6892	0.2797	0.2701	0.8899	0.3846	0.986	0.0668
FEDformer	0.6511	0.8241	0.4868	0.4603	0.9176	0.56	0.9654	0.3369
TimesNet	0.6329	0.8352	0.4861	0.4599	0.9132	0.5989	0.9745	0.2914
TFNet	0.6443	0.8344	0.4616	0.4519	0.9163	0.5168	0.9598	0.3289

4.5.2 Does the Channel-wise module improve our model?

We have tried different modifications to the TFNet model using a channel-wise module to see if they add to the model's performance. Namely, we performed experiments on the following:

- CWTFNet
  - To the TFNet model, we applied the channel-wise module on the input before analyzing the learned independent channel-wise features in the frequency domain by the TimesBlock.
- CWTFNet_2, with the bi-GRU from the CWTFNet removed.
- CWTNet

– With inspiration from the work of Chang et al. in [2], in the CWTFNet, we employ the TimesBlock for each of the learned representations from the channel-wise module.

  * The channel-wise TimesBlock is trained with and without weight sharing across the channels.

Table 4.5: Ablations performed with channel-wise module

model	Imputation	bAcc	Precision_0	Precision_1	Recall_0	Recall_1	AUROC	AUPRC	min(P^+Se)
CWTFNet	carry forward	0.6453	0.9163	0.5446	0.9644	0.3262	0.8418	0.4744	0.4548
	normal_value	0.6455	0.9163	0.5654	0.9675	0.3235	0.8371	0.4836	0.4572
	carry backward	0.6383	0.9147	0.5472	0.9665	0.3102	0.8381	0.4685	0.4572
CWTFNet_2	carry forward	0.5715	0.8996	0.5455	0.9825	0.1604	0.7938	0.4015	0.4278
	normal_value	0.5619	0.8975	0.5096	0.9822	0.1417	0.761	0.3753	0.377
	carry backward	0.5296	0.8906	0.5319	0.9923	0.0668	0.7858	0.3533	0.391
CWTNet_sharing	carry forward	0.6111	0.9081	0.6642	0.9843	0.238	0.8282	0.4951	0.467
	normal_value	0.5853	0.9026	0.5635	0.9808	0.1898	0.8183	0.4235	0.4011
	carry backward	0.6084	0.9078	0.5449	0.9734	0.2433	0.8294	0.4378	0.4492
CWTNet_no_sharing	carry forward	0.6354	0.9139	0.5773	0.9713	0.2995	0.8345	0.4789	0.4453
	normal_value	0.5891	0.9035	0.5396	0.9776	0.2005	0.8089	0.4163	0.4058
	carry backward	0.6106	0.9082	0.5644	0.9752	0.246	0.8257	0.4488	0.432

Adding a Channel Wise module before TFNet did not result in an improved model. The CWTFNet has shown better results than CWTFNet_2, from this, we deduce that the bi-GRU layer after the 2D multi-period representations are learned is an important component of the TFNet model. It is observed that adding channel-wise modules to the TFNet in a sequential doesn't add much to the performance.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this work:

- We proposed TFNet, a novel deep-learning method that captures representations in both the time and frequency domain from the multivariate time series. First, to extract multi-period variations we used the TimesBlock module, which analyzes the time series in the frequency domain. After the multi-period representations are extracted, we employ a bidirectional GRU to learn temporal dependencies.

- Our model achieved an improved performance on the in-hospital mortality prediction (48 hours) task on the MIMIC-III dataset; reportedly, it shows a 15.5% improved recall of positive class and a 2% improvement in AUPRC than the best baseline, the CW-LSTM.

- We tried 3 different imputation techniques and noted that carry-forward imputation results in the best performance compared to the other imputation methods.

- Ablation studies were performed by adding and removing the channel-wise module and bi-GRU, and we observed that adding a channel-wise did not improve the model significantly. When bi-GRU is removed, it reports a poor performance than TFNet this validates the importance of bi-GRU in learning temporal dependencies from the learned multiperiod representations.

5.2 Future Work

Although a building block of TFNet, the TimesBlock is proposed as a task-general model, the current work was only focussed on in-hospital mortality prediction on MIMIC-III. Extensive experiments need to be performed on general time series classification datasets such as UEA & UCR Time Series Classification Archive.
It needs to be seen if the TFNet would immediately apply to short-term (with 2-12 hour observations) and long-term mortality (with a few months to years of observations) prediction tasks. Performance on other clinical databases, like eICU, should be verified. It is also interesting direction to see how the channel-wise module, which encodes missing mask information more effectively and the TimesBlock should combine in the hope of a model that can better capture missingness patterns.

Bibliography

[1] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M. Buhmann. The balanced accuracy and its posterior distribution. *2010 20th International Conference on Pattern Recognition*, pages 3121–3124, 2010.

[2] Feng-Ju Chang, Martin H. Radfar, Athanasios Mouchtaris, Brian King, and Siegfried Kunzmann. End-to-end multi-channel transformer for speech recognition. *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5884–5888, 2021.

[3] Dustin Charles, Meghan Gabriel, and Michael F Furukawa. Adoption of electronic health record systems among us non-federal acute care hospitals: 2008-2012. *ONC data brief*, 9(1):9, 2013.

[4] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values, 2017.

[5] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[6] E. Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F. Stewart, and Jimeng Sun. Doctor ai: Predicting clinical events via recurrent neural networks. *JMLR workshop and conference proceedings*, 56:301–318, 2015.

[7] Jesse Davis and Mark H. Goadrich. The relationship between precision-recall and roc curves. *Proceedings of the 23rd international conference on Machine learning*, 2006.

[8] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33:917–963, 2018.

[9] Pedro J García-Laencina, José-Luis Sancho-Gómez, and Aníbal R Figueiras-Vidal. Pattern classification with missing data: a review. *Neural Computing and Applications*, 19:263–282, 2010.

[10] Hrayr Harutyunyan, Hrant Khachatrian, David C. Kale, Greg Ver Steeg, and Aram Galstyan. Multitask learning and benchmarking with clinical time series data. *Scientific Data*, 6(1):96, 2019.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[12] Alistair Johnson, Tom Pollard, and Roger Mark. Mimic-iii clinical database (version 1.4). *PhysioNet*, 2016.

[13] Alistair E.W. Johnson, Tom J. Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. Mimic-III, a freely accessible critical care database. *Scientific Data*, 2016.

[14] David Kreindler and Charles J. Lumsden. The effects of the irregular sample and missing data in time series analysis. *Nonlinear dynamics, psychology, and life sciences*, 10 2:187–214, 2006.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60:84 – 90, 2012.

[16] Zachary Chase Lipton, David C. Kale, and Randall C. Wetzel. Directly modeling missing data in sequences with rnns: Improved classification of clinical time series. *ArXiv*, abs/1606.04130, 2016.

[17] S. Purushotham, Chuizheng Meng, Zhengping Che, and Yan Liu. Benchmarking deep learning models on large healthcare datasets. *Journal of biomedical informatics*, 83:112–134, 2018.

[18] Kira Rehfeld, Norbert Marwan, Jobst Heitzig, and Jürgen Kurths. Comparison of correlation analysis techniques for irregularly sampled time series. *Nonlinear Processes in Geophysics*, 18:389–404, 2011.

[19] Donald B. Rubin. Inference and missing data. *Psychometrika*, 1975:19, 1975.

[20] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2014.

[21] Joseph L. Schafer and John W. Graham. Missing data: our view of the state of the art. *Psychological methods*, 7 2:147–77, 2002.

[22] Shickel. and Benjamin et al. Deep EHR: A survey of recent advances in deep learning techniques for electronic health record (EHR) analysis. *IEEE journal of biomedical and health informatics*, 22(5):1589–1604, 2018.

[23] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[24] Qingxiong Tan, Mang Ye, Andy Jinhua Ma, Baoyao Yang, Terry Cheuk-Fung Yip, Grace Lai-Hung Wong, and Pong C. Yuen. Explainable uncertainty-aware convolutional recurrent neural network for irregular medical time series. *IEEE Transactions on Neural Networks and Learning Systems*, 32(10):4665–4679, 2021.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[26] Brian J. Wells, Kevin Chagin, Amy S. Nowacki, and Michael W. Kattan. Strategies for handling missing data in electronic health record derived data. *eGEMs*, 1, 2013.

[27] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The Eleventh International Conference on Learning Representations*, 2023.

[28] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series fore-

casting. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[29] Ailing Zeng, Mu-Hwa Chen, L. Zhang, and Qiang Xu. Are transformers effective for time series forecasting? *ArXiv*, abs/2205.13504, 2022.

[30] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2020.

[31] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 27268–27286. PMLR, 17–23 Jul 2022.