Domain Specific Feature Representation Learning for Diverse Temporal Data

Farhan Asif Chowdhury
University of New Mexico - Main Campus

Follow this and additional works at: https://digitalrepository.unm.edu/cs_etds

Part of the Computer Sciences Commons, and the Data Science Commons

Recommended Citation

This Dissertation is brought to you for free and open access by the Engineering ETDs at UNM Digital Repository. It has been accepted for inclusion in Computer Science ETDs by an authorized administrator of UNM Digital Repository. For more information, please contact disc@unm.edu.
Farhan Asif Chowdhury

Candidate

Computer Science

Department

This dissertation is approved, and it is acceptable in quality and form for publication:

Approved by the Dissertation Committee:

Abdullah Mueen

Chair

Manel Martinez-Ramon

Member

Trilce Estrada

Member

Bruna Jacobson

Member

i
Domain Specific Feature Representation Learning for Diverse Temporal Data

BY

Farhan Asif Chowdhury
B.S., Electrical and Electronic Engineering, Bangladesh University of Engineering and Technology, 2016
M.S., Computer Science, University of New Mexico, 2019

DISSERTATION
Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

Computer Science
The University of New Mexico
Albuquerque, New Mexico

December, 2023
DEDICATION

This dissertation is dedicated to my parents,
my brother,
and my wife
ACKNOWLEDGMENTS

First of all, I would like to thank my adviser, Abdullah Mueen, for his guidance and support. He taught me how to find meaningful problems and formulate a research question. He has been a constant source of motivation and encouragement, especially during challenging times. My time with Mueen has been greatly enjoyable, and I will always cherish our working memories together.

I would also like to give my sincere thanks to my committee members, Manel Martinez-Ramon, Trilce Estrada, and Bruna Jacobson, for their feedback and support. I am grateful to my internship mentors - Maarten Bos, Yozen Liu, Neil Shah, and Leonardo Neves from Snap Research; Bipob Debnath and Srimat Chakradhar from NEC Labs; Sidney Tsang and Qiaochu Yuan from Meta.

I want to thank my fellow lab mates Ashraf Siddiquee, Rashidul Hasan, Dheenan Saha, Parvez Muntasir Supto, Cynthia Freeman, Nikan Chavoshi, Sheng Zhong and Lawrence Allen for their friendship, support and collaborations. I would also like to thank my co-interns, Koustuv Saha, Nicholas Vincent from Snap Research, Ren Pang, and Sriyesh Yende from Meta, for all the fun memories and support.

I am grateful to my parents for their never-ending support and motivation. They were always there for me to cheer me up and guide me to make the right decision. They believed in me more than I would ever do. I am also grateful to my elder brother for his help, support, love, and mentorship. He stood by my side through thick and thin, took care of me when I was sick, and always shared words of encouragement. Finally, I am forever grateful to my loving wife for her continuous support, care, and patience throughout this journey. She always motivated me and cheered me up during the challenging journey of this dissertation.
Domain Specific Feature Representation Learning for Diverse Temporal Data

by

Farhan Asif Chowdhury

B.S., Electrical and Electronic Engineering, Bangladesh University of Engineering and Technology, 2016

M.S., Computer Science, University of New Mexico, 2019

Ph.D., Computer Science, University of New Mexico, 2023

ABSTRACT

Humans can leverage domain context to recognize novel patterns and categories based on limited known examples. In contrast, computational learning methods are not adept at exploiting context and require sufficient labeled examples to achieve similar accuracy. Many temporal data domain, for example, seismic signals and oil mining sensor data, requires domain expert annotation, which is both costly and time-consuming. The dependency on training data limits the applicability of machine learning algorithms for domains with limited labeled data. This dissertation aims to address this gap by developing temporal mining algorithms that exploit domain context to learn discriminative feature representation from limited samples to achieve improved performance. In this dissertation, I present four domain-specific feature representation learning methods for three diverse domains: a pattern detection algorithm for oil and gas mining, two classification methods for seismic activity monitoring, and a prediction model for user behavior in social media. We show empirical evidence of performance improvement using
real datasets. We demonstrate these methods’ practical usability for multiple real-world applications.
# TABLE OF CONTENTS

List of Figures ................................................................. x  
List of Tables ................................................................. x  

1 Introduction 1  
1.1 Structured Noise Detection in Well Test Pressure Derivative Data ................................................... 3  
1.2 Seismic Phase Identification for Automated Monitoring ................................................................. 5  
1.3 Few-shot Learning for Seismic Phase Detection ................................................................. 5  
1.4 User Behavior Modeling in Social Media ................................................................. 6  

2 Structured Noise Detection in Well Test Pressure Derivative Data 8  
2.1 Introduction ................................................................. 8  
2.2 Related Works ................................................................. 12  
2.3 Singular Spectrum Analysis ................................................................. 13  
2.3.1 Embedding ................................................................. 13  
2.3.2 Singular Value Decomposition ................................................................. 14  
2.3.3 Grouping ................................................................. 14  
2.3.4 Diagonal Averaging ................................................................. 15  
2.3.5 Why SSA? ................................................................. 16  
2.4 Structured Noise Detection ................................................................. 16  
2.4.1 Signal Decomposition ................................................................. 18  
2.4.2 Structured Noise Component Identification ................................................................. 19  
2.4.3 Structured Noise Localization ................................................................. 22  
2.5 Experiment and Results ................................................................. 24
6 Conclusion and Future Work
List of Figures

2.1 (left) An ideal pressure derivative data without any structured noise; (right) A structured noise corrupted pressure derivative data where red labeling indicates the structured noise (both data from Well-A [19])………………. 12

2.2 (left) Smooth trend over the noisy signal; (right) structured noise as residue after subtracting trend from the noisy signal (PBU-13-Well-A [19])…………………………………….. 18

2.3 (a)-(e)Reconstructed components (RC1-RC5) after SSA decomposition (f) Sum of RC2 through RC5 overlapped with structured noise residue. (PBU-13-Well-A [19])………... 20

2.4 SSA eigenvalue spectra with 95% confidence limit for all 38 pressure derivative data of Well-A[19]…………………………………….. 21

2.5 (left) Sum of the structured noise component values as an indicator of structured noise presence; (right) windowed averaging performed on the sum of the structured noise components. (red dashed line indicates threshold limit, PBU-13 Well-A [19])  24

2.6 (left) PBU-13-Well-A, (right) PBU-24-Well-A [19]; Two demonstrations of using F-score as performance evaluation measure for semi-supervised threshold selection. (maximum F-score values are shown by arrow)…………………………………….. 25

2.7 Automatic noise detection result on real pressure build-up data from Well-A [19]; (left) structured noise due to unknown reason; (middle) structured noise due to unknown reason; (right) structured noise due to pressure leak at flow line.……………….. 26
2.8 Automatic noise detection result on real pressure build-up data from Well-C [19]; (left) structured noise due to fluid segregation; (middle) structured noise due to fluid segregation; (right) structured noise due fluid segregation.

2.9 Histogram plot of the number of times a threshold value is selected using single-sample learning: (top) Well-A, (middle) Well-B, (bottom) Well-C [19].

2.10 Accuracy of structured noise detection performance for various threshold values on Well-(A, B, C) [19]. The maximum accuracy values (shown by arrows) vary across wells.

3.1 (left) Typical seismic data processing pipeline. Our objective is to develop a Machine Learning model for phase identification. (right) Travel times with respect to distance for various seismic phases. Phases ending with P can commonly be categorized as P, and phases ending with S can commonly be categorized as S.

3.2 Input images of waveforms are created by taking CWT of individual channels (i.e. BHZ, BHN, BHE).

3.3 CWT of pairs of examples from all phase types. Compressional or longitudinal waves (P, Pn, Pg) are dominantly red due to high vertical component amplitudes. Transverse waves (S, Sn, Lg) are dominantly green/blue due to high horizontal component amplitudes.

3.4 Proposed model architecture.

3.5 (Top) Confusion matrix for the test cases of a randomly split 80-10-10 train-validation test scenario. (Bottom) t-SNE visualization of the same 10% test cases considering the activation values of the last layer before the prediction layer as deep embedding [137]. The compressional and transverse wave signal samples are well separated.
3.6 Performance comparison of FASER with naive nearest neighbor classifier for application in novel stations. The plot shows the cumulative distribution function (CDF) for a percentage of stations having smaller than a given accuracy. FASER achieves an approximately 72% accuracy for the majority of the stations in contrast to 17% for the nearest neighbor classifier.

3.7 (Left) Testing with held-out stations. Each dot is an IMS station. The color bar shows held-out accuracy for a station. (Right) Testing with held-out regions. Each red dot is an event recorded at the NEIC (National Earthquake Information Center). We hold out shaded regions. The average hold-out accuracy across shaded regions is 77.27%, with a standard deviation of 4.13%, suggesting model efficacy at novel source regions.

4.1 Global Monitoring Station network across the world. The size of the node is proportional to the labeled sample number.

4.2 The average accuracy of FASER for stations grouped based on labeled data amount. Here, the accuracy is higher for stations with higher labeled examples and vice versa.

4.3 Station clusters based on geographic proximity. The stations of the same cluster have a similar color.

4.4 Outline of the few-shot seismic phase identification framework.

4.5 The architecture of FUSED, which consists of the few-shot module and the global feature module.
5.1 (top) Diurnal cyclicity in global user activity. The y-axis represents the fraction of activity that happens in each weekday-hour over a single week, aggregated across all users. (bottom) DISCOVER VIEW activity across user clusters by hour (values are z-transformed), darker shades indicate higher activity amount. Here, users are clustered based on the aggregated value of their Snapchat activities (such as frequency and amount of communication and content consumption).

5.2 (a) Cyclicity CDF; most users are cyclic, more than 80% of users having a cyclicity score of more than 0.25. (b)-(f) show hourly average activity (session duration) in the first three weeks (history) and fourth week (current) for five example users, each selected from one of the correlation value buckets ([-1, 0], [0, 0.25], [0.25, 0.50], [0.50, 0.75], [0.75, 1]), as designated with dashed lines in (a). (b) Correlation bucket [-1, 0]. (c) Correlation bucket [0, 0.25]. (d) Correlation bucket [0.25, 0.5]. (e) Correlation bucket [0.5, 0.75] (f) Correlation bucket [0.75, 1]. Here, corresponding with the increasing cyclicity score from (b)-(f), the similarities between historical and current activity increases, too.

5.3 We show the correlation of user’s activity level between consecutive sessions. We categorize each session into three categories based on the level of activity. We show the distribution of current session’s activity level in relation to the previous session’s activity level for two cases; (a) SESSION TIME, (b) DISCOVER VIEW. In both cases, consecutive sessions are more likely to belong to the same category.

5.4 Proposed model architecture.
5.5 (a) Accuracy change ($z$-transformed, 0 shows mean improvement) for more-cyclic and less-cyclic user groups for all four tasks after adding the cyclic module. We gain greater improvement for more-cyclic users. (b) The variation in ephemeral and cyclic attention scores as the ephemeral session number varies. The cyclic attention is greater in fewer ephemeral sessions, and the ephemeral attention increases along with the increase in ephemeral session number.

5.6 The hourly averaged accuracy for all four tasks (a) Task 1 and Task 2, (b) Task 3 and Task 4. (c) The average number of ephemeral and cyclic sessions in each hour. (d) Hourly average ephemeral and cyclic attention weight, and accuracy change after incorporating cyclic module ($z$-transformed).

5.7 F1-score for user-agnostic vs. non user-agnostic training and testing.
## List of Tables

2.1 Accuracy of Structured Noise Detection on real pressure build-up data from three different wells - marked as A, B, and C [19]. 26

3.1 Performance metric comparison of **FASER** against baseline methods. **FASER** outperforms existing baseline methods in all four metrics: Precision, Recall, F-Score, and Accuracy. The best algorithm for each metric is colored in cyan. ............... 47

4.1 Accuracy for the held-out validation samples from in-training stations and for the samples from the novel test stations from a different cluster. ................................. 56

4.2 Performance metric comparison of **FUSED** against baseline methods. **FUSED** outperforms existing baseline methods for both one-shot and five-shot seismic phase identification tasks. The best algorithm for each task is colored in cyan. ........... 67

5.1 Prediction performance (macro f1-score) of **CEAM** on all tasks and both datasets **MAU** and **DAU**. ................................. 89

5.2 Ablation studies to show the contribution to performance improvement (macro f1-score) by different modules. ............ 90
Chapter 1

Introduction

In the current age of big data, temporal data is omnipresent [5, 27, 136, 79]. Temporal or time-series data represents sequential observations or measurements over time [87, 52]. Large sensor networks, social media, smart appliances, self-driving cars, healthcare devices, etc., produce enormous amounts of temporal data [75, 115]. As data storage and computation capacity have advanced and become more affordable, organizations in every industry can collect and store massive amounts of temporal data for real-time monitoring and future analysis [23, 140]. With the emergence of 5G networks, the advancements in Internet of Things (IoT) technology, and the surge in online activities, the volume of temporal data is increasing exponentially [4, 22].

Researchers have devoted significant efforts to derive actionable knowledge from these massive datasets [10]. Many pattern detection, classification, and prediction algorithms have been developed for application in diverse domains, from user behavior prediction in social media to real-time seismic activity monitoring [79]. Lately, machine learning models have significantly improved the performance of various temporal data mining tasks [61, 145, 135]. These models are adept at automating the feature representation learning process irrespective of the data source and are often applied in a domain-agnostic way. However, these models rely on sufficient labeled instances for learning and perform poorly otherwise [7, 94]. For many tem-
poral data domains, the availability of labeled data is limited, mainly due to dependency on domain experts for manual labeling [103]. For example, in seismic monitoring and petroleum mining, manual labeling depends on multiple domain experts [27, 29]. The scarcity of labeled data leads to subpar performance of deep learning models in many temporal data mining tasks [62].

To overcome these limitations, in this dissertation, we aim to exploit data properties and domain knowledge to support learning from limited training examples. Based on the source and specific domain, different datasets can have distinct intrinsic properties, patterns, and context [79, 65, 82]. For example, humans are habitual by nature, and thus their offline activity – for example, commute routine and online activity – for example, social platform usage, show distinct regularities and cycles for different persons [28]. Where else, seismic signals do not contain such distinctive temporal patterns across different phase types; instead, they possess contrasting patterns in the spectral domain depending on their specific phase types [29].

This dissertation’s main goal is to develop temporal data mining algorithms that leverage data properties and domain context to reduce dependency on labeled instances. We developed three temporal data mining algorithms to improve pattern recognition, classification, and prediction tasks. We show an overview of our dissertation objective in Figure 1. First, in Chapter 2, we present a semi-supervised pattern detection algorithm for well-test pressure derivative data that can adapt to novel datasets using only one labeled example. We use Singular Spectrum Analysis (SSA) to decompose the time series into additive components and leverage temporal variance to identify the components that contain the pattern of interest. In Chapter 3, we introduce FASER, an automated seismic phase detection algorithm that exploits the spectro-temporal patterns in the seismic signals. We use deep sequential networks to capture both local structured patterns and long-temporal patterns for improved classification performance. In Chapter 4, we present FUSED, where we address the limitations of FASER in dealing
with limited training data for novel stations. We propose a few-shot learning method using metric-learning based episodic training framework to enable effective learning from a few labeled instances to improve phase identification performance. In Chapter 5, we present CEAM, where we exploit cyclic regularities in human behavior to improve user behavior modeling in social media. We augment the deep sequential model with a cyclic module and attention mechanism to better exploit cyclic patterns in conjunction with the user’s momentary interest. We present extensive evaluation using real datasets and demonstrate multiple practical use cases for all four methods. Finally, we conclude and outline future works in Chapter 6.

1.1 Structured Noise Detection in Well Test Pressure Derivative Data

Real-valued data sequences are often affected by structured noise in addition to random noise. For example, in pressure transient analysis (PTA), semi-log derivatives of log-log diagnostic plots show such contamination of structured noise, especially under multiphase flow conditions. In PTA data, structured noise refers to the response to some physical phenomena not originating at the reservoir, such as fluid segregation in the wellbore or pressure leak due to a brief valve opening. Such noisy responses commonly appear to mix with flow regimes, hindering further reservoir flow analysis.

In this work, we use the Singular Spectrum Analysis (SSA) to decompose PTA data into additive components; subsequently, we use the eigenvalues associated with the decomposed components to identify the components that contain most of the structured noise information. We develop a semi-supervised process that requires minimal expert supervision in tuning the solitary parameter of our algorithm using only one pressure buildup scenario. An empirical evaluation using real pressure data from oil and gas wells shows that our approach can detect a multitude of structured noise with 74.25% accuracy.
Figure 1.1: An overview of the typical temporal data processing pipeline. The temporal data is fed into a learning model, where the goal is primarily to learn a feature representation and finally produce an output that falls under either pattern recognition, classification, or prediction task. In this dissertation, we aim to leverage domain context during the feature representation learning phase to improve these temporal data mining tasks. In Chapter 2, we leverage domain context to identify structured noise from well pressure data; in Chapters 3 and 4, we present our work on seismic signal data; and in Chapter 5, we exploit the regularities in user behavior on social media data.
1.2 Seismic Phase Identification for Automated Monitoring

Seismic phase identification classifies the type of seismic wave received at a station based on the waveform (i.e., time series) recorded by a seismometer. Automated phase identification is an integrated component of large-scale seismic monitoring applications, including earthquake warning systems and underground explosion monitoring. Accurate, fast, and fine-grained phase identification is instrumental for earthquake location estimation, understanding Earth’s crustal and mantle structure for predictive modeling, etc. However, existing operational systems utilize multiple nearby stations for precise identification, which delays response time with added complexity and manual interventions. Moreover, single-station systems mostly perform coarse phase identification.

In this work, first, we revisit the seismic phase identification as an integrated part of a seismic processing pipeline. We develop a machine-learned model FASER that takes input from a signal detector and produces phase types as output for a signal associator. The model is a combination of convolutional and long short-term memory (LSTM) networks. Our method identifies finer wave types, including crustal and mantle phases. We conduct comprehensive experiments on real datasets to show that FASER outperforms existing baselines. We evaluate FASER holding-out sources and stations across the world to demonstrate consistent performance for novel sources and stations.

1.3 Few-shot Learning for Seismic Phase Detection

FASER achieves state-of-the-art accuracy in seismic phase identification tasks. However, the performance decreases for stations in new geographic loca-
tions with limited labeled examples. As new stations are being deployed in novel and previously unmonitored regions due to dynamic monitoring requirements, an effective method to perform phase identification in such stations with few training examples is crucial. In this work, we formulate seismic phase identification as a few-shot learning problem to identify phase types in novel stations using very few-labeled data. We propose FUSED, a metric-learning based meta-learning method that leverages episodic training to simulate learning from limited training data. FUSED leverages both station-specific and station-agnostic features for improved performance. We demonstrate the efficacy of the proposed method for novel stations using real seismic datasets.

1.4 User Behavior Modeling in Social Media

To improve the user experience as well as business outcomes, social platforms aim to predict user behavior. To this end, recurrent models are often used to predict a user’s next behavior based on their most recent behavior. However, people have habits and routines, making it plausible to predict their behavior from more than just their most recent activity. Our work focuses on the interplay between ephemeral and cyclical components of user behaviors. By utilizing user activity data from social platform Snapchat, we uncover cyclic and ephemeral usage patterns on a per user-level. Based on our findings, we imbued recurrent models with awareness: we augment an RNN with a cyclic module to complement traditional RNNs that model ephemeral behaviors and allow a flexible weighting of the two for the prediction task. We conducted extensive experiments to evaluate our model’s performance on four user behavior prediction tasks on the Snapchat platform. We achieve improved results on each task compared against existing methods, using this simple, but important insight in user behavior: both cyclical and ephemeral components matter. We show that in some situations and for some people, ephemeral components may be more helpful for predicting behavior, while for
others and in other situations, cyclical components may carry more weight.
Chapter 2

Structured Noise Detection in Well Test Pressure Derivative Data

2.1 Introduction

Sensor data collection and analysis have become ubiquitous in production and manufacturing operations for continuous surveillance and monitoring [64]. These sensors gather and record data at regular intervals, producing a vast amount of data, mostly in the form of time series [50]. This massive data trove can be transformed into explicit actionable knowledge using data mining techniques. However, these datasets are often contaminated with noise due to inefficient calibration, measurement error, external interruption, etc., and this presence of noise is the main hindrance to further automated analysis and decision-making process.

In the oil and gas industry, permanent downhole pressure gauges are installed in wells drilled in oil and gas fields to monitor production [60]. The deployed pressure gauges continuously record pressure in the well at regular intervals, producing a time series of the downhole pressure. This time series is utilized in pressure transient analysis (PTA) to determine reservoir and well
characteristics [36], to continuously assess reservoir and well condition, and to forecast future production performance. The log-log pressure derivative plot during the shut-in period of the well is widely used in PTA for well property evaluation [16].

Identifying different flow regimes in log-log pressure derivative data during the shut-in period reveals characteristics of the hydrocarbon-bearing formation, i.e., reservoir and condition of the well. Currently, the task of identifying the flow regime is mainly done by manual observation, although some automated PTA methods have been developed recently [105, 37, 125]. One main impediment of fully automated PTA is that flow regime identification in log-log pressure derivative data is misled due to the presence of structured noise.

There is no unified definition of noise in general; instead, it is highly domain and problem-specific as it relates to different events in different observation and measurement systems. We use the terminology structured noise, in the context of pressure data recorded by downhole gauge, to distinguish the usual pressure response from the pressure response to non-reservoir origin physical phenomenon, for example, fluid segregation in the wellbore or pressure leak due to a brief opening of a valve. Mostly, they are deviations from the usual response for a brief time in the original data, affecting the overall trend. These structured noises often maintain a similar pattern between different observations, which implies that there must be a common underlying physical event that may have generated them.

In Figure 2.1, we show two pressure derivative data, often also referred to as pressure buildup (PBU) data. In Figure 2.1(left), the PBU data is free from structured noise, and in Figure 2.1(right), the PBU data is contaminated by structured noises, which are marked in red (automatically labeled using our method, showing an accurate detection case-study). These structured noises often mix up with flow regimes, and in those cases, flow regime identification algorithms generate erroneous results, preventing further down-the-road analysis. Hence, an efficient and robust structured noise detection
algorithm will open new frontiers of data mining applications on PBU data. This work proposes a structured noise detection method for well test pressure derivative data using singular spectrum analysis (SSA).

**Why the problem is challenging?** The pressure derivative data is non-stationary and has variable duration and amplitude offset; thus, using Fourier Transform based filtering techniques is not feasible. The associated random noise in the log-log pressure data is easier to filter out as it poses significantly different spectral properties than the noise-free signal. However, structured noises occur briefly while maintaining a contiguous shape; thus, they appear as a part of the original signal. Again, their frequency spectrum is wide, often overlapping with the noise-free signal spectrum. Moreover, structured noise segments have variable lengths; their shape and statistical properties vary between data observations, which obsoletes using a segment-based classification approach. Wavelet is a powerful spectral filtering tool that produces acceptable results in similar scenarios. One main drawback of Wavelet analysis is that it requires the manual selection of a basis function on which the filtering quality depends greatly. As our data has a high level of variation from one observation to another and one well to another, one selected fixed basis cannot work optimally for all the variations. A more robust method is required to adapt to the variation of data. Moreover, our objective is not only to filter out but rather detect and localize structured noise segments in the noisy observation, which poses new challenges.

**Our Approach.** In this work, we propose a method where we use singular spectrum analysis (SSA) [18] to detect structured noise segments in real oil and gas well log-log pressure derivative data to automate the pressure transient analysis process. The SSA is a non-parametric time series analysis tool and does not make a prior assumption about the data. There are hyper-parameters involved to be adapted to the problem at hand. It has been used in a multitude of problem domains. SSA has been mainly used for time-series modeling [131], structural change detection [91], and forecasting [45]. Recently, it has been widely used in biomedical signal processing for noise
and artifacts removal [39, 129, 85, 84].

SSA decomposes a signal into multiple additive components, which usually can be interpreted as the trend components, amplitude and phase modulated oscillatory components, and the unstructured random noise components [47]. We use SSA to decompose our signal of interest into such components, and we identify the relevant components that capture most of the information about the structured noise present in the data. As structured noise poses oscillatory behavior, it should be projected onto the decomposed oscillatory components, and we use the eigenvalues associated with the decomposed components to identify these components. Once we identify the relevant components, we localize the structured noise segments present in those components using a threshold value. To select the threshold value, we employ a single-sample learning method whenever we use our algorithm on new well data. After detecting the structured noise, we perform further boundary refinement for precise structured noise localization.

Experimental evaluation using real pressure data from oil and gas wells shows that our approach can identify a multitude of structured noise with 74.25% accuracy. Besides, the algorithm is invariant to the amount and types of noise. The algorithm is generalizable to structured noise detection in other kinds of data. By employing single-sample learning to fine-tune the threshold parameter each time we apply our method to new well data, we ensure that our method can adapt to a multitude of structured noise types that show large variations from well to well. Moreover, the algorithmic computation need not be rerun for each threshold value change. In Figure 2.1, we have shown an example of our noise detection result where the left figure shows accurate zero noise detection in the absence of noise, and the right figure shows precise noise detection.
2.2 Related Works

Extensive studies have been carried out to perform noise removal on pressure data due to its usability in evaluating reservoir characteristics. Most related studies have performed noise removal on the original pressure data before calculating the derivative. However, the removal of noise before calculating the derivative often time smooths out finer details. Also, noise is more pronounced in derivative data which requires more rigorous noise removal techniques. To the best of our knowledge, this is the first attempt to detect structured noise in pressure derivative data.

Noise reduction in the original pressure data is usually done using spectral methods and regression analysis. Wavelet transform has been widely used to reduce noise [121]. Some other common filtering methods are Butterworth, Locally Weighted Scatterplot Smooth (LOESS), and Auto-regressive Moving Average (ARMA) [96]. In [90], they have shown a comparative evaluation of the aforementioned denoising methods for well pressure data. Among all these, wavelet performs better. But as mentioned earlier, one of the main challenges in applying wavelets is that we need to select the basis for wavelets.
manually and the performance of the noise detection varies depending on the choice of the basis. As a result of this basis dependency, it fails to adapt to data from different well types associated with a diverse range of structured noise.

2.3 Singular Spectrum Analysis

[40, 18] works with a one-dimensional time series of finite length, and decomposes the time series into additive components. The algorithm consists of two main steps: decomposition and reconstruction. The decomposition stage is composed of embedding and singular value decomposition. The reconstruction stage is composed of grouping and diagonally averaging. Below, we provide a brief description of each of these four steps.

2.3.1 Embedding

Let’s consider a real-valued non-zero time series $S = (s_1, s_2, ... , s_N)$ of length $N$ where $N > 2$. In the embedding step, the time series $S$ is converted into a trajectory matrix $X$ by sliding an $M$-point window over the time series where $M$ is called the embedding dimension. In this step, the one-dimensional time series gets mapped into a sequence of $K = N - M + 1$ multi-dimensional lagged column vectors of length $M$ that constitute the column of the trajectory matrix $X$.

$$X = \begin{bmatrix}
    s_1 & s_2 & \cdots & s_K \\
    s_2 & s_3 & \cdots & s_{K+1} \\
    \vdots & \vdots & \ddots & \vdots \\
    s_M & s_{M+1} & \cdots & s_N 
\end{bmatrix} \quad (2.1)$$

All of the anti-diagonal elements (when $i + j = constant$, where $i$ and $j$ are row and column indices) of the trajectory matrix (2.1) have the same value
(such matrices are also known as Hankel matrix). Embedding dimension $M$ directly affects the decomposition quality. The optimal embedding dimension depends on the purpose of the analysis and the nature of the time series.

### 2.3.2 Singular Value Decomposition

Singular Value Decomposition (SVD) of a real non-zero matrix $X$ of size $M \times K$ decomposes the matrix into a sum of rank one orthogonal elementary matrices ($X_i$).

$$X = \sum_{i=1}^{L} X_i = \sum_{i=1}^{L} \sqrt{\lambda_i} e_i v_i^T$$

(2.2)

Here, $\lambda_i$ are the eigenvalues of the covariance matrix $\Sigma = XX^T$ in descending order, i.e., $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_L \geq 0$, and $e_1, e_2, \cdots, e_L$ are the corresponding eigenvectors. Here, $v_i = \frac{X^T e_i}{\sqrt{\lambda_i}}$ and $L$ is the no of non-zero singular values of $X$, $L = \{ \lambda_i > 0 \}$. The collection $(\sqrt{\lambda_i} e_i v_i)$ is called the eigentriple of the SVD (2.2). The eigenvectors are also called empirical orthogonal functions (EOF) and can be considered as a set of data adaptive orthogonal basis functions for signal decomposition.

The eigenvalue $\lambda_i$ represents the amount of partial variance in the direction of the corresponding eigenvectors or EOF $v_i$ and the sum of the eigenvalues gives the total variance of the original time series. The energy contribution of the $i$-th eigenvector is given by the ratio $\lambda_i \sum_{i=1}^{L} \lambda_i$ and is called the singular spectrum of the time series.

### 2.3.3 Grouping

In the grouping step, we partition the elementary matrices ($X_1, X_2, \cdots, X_L$) into $m$ disjoint groups $I_1, I_2, \cdots, I_m$ and then sum the matrices in each subset. For a group $I_j$ whose member indices are $(i_{j1}, \cdots, i_{jp})$ the resulting matrix would be $X_{I_j} = X_{i_{j1}} + \cdots + X_{i_{jp}}$ and we compute such matrix
(X_{I_1}, X_{I_2}, \cdots X_{I_m})$ for all the groups. Finally, we obtain the expansion of the trajectory matrix $X$ as a summation (2.3) of the grouped matrices:

$$X = \sum_{k=1}^{m} X_{I_k} = \sum_{k=1}^{m} \left( \sum_{i \in I_k} X_i \right)$$

(2.3)

There is no specific rule dictating how to perform the grouping and it depends on the purpose of the time series analysis and the type of signal and noise. Most often, the singular spectrum and the singular vectors are used to form groups of similar components.

### 2.3.4 Diagonal Averaging

In the last step of the reconstruction phase, each of the matrices formed after grouping is converted into a one-dimensional time series of length $N$ by applying a linear transformation called diagonal averaging or hankelization [47], where the cross-diagonal elements are averaged to obtain a single value. If $X_{I_k}$ is a reconstructed trajectory matrix with dimension $M \times K$, we average over the elements $i + j = q + 2$ to calculate the $q$th element of the converted time series, where $i$ and $j$ are row and column indices. If the elements of the matrix $X_{I_k}$ are $Y_{i,j}$, then we can formalize the averaging procedure using the following equations:

$$x_n^k = \begin{cases} 
\frac{1}{n} \sum_{m=1}^{n} Y_{m,n-m+1} & \text{for } 1 \leq n \leq M \\
\frac{1}{M} \sum_{m=1}^{M} Y_{m,n-m+1} & \text{for } M \leq n \leq K \\
\frac{1}{N-n+1} \sum_{m=n-K+1}^{M} Y_{m,n-m+1} & \text{for } K + 1 \leq n \leq N 
\end{cases}$$

(2.4)
This diagonalizing (2.4) of each of the reconstructed trajectory matrices 
\((X_{l_1}, X_{l_2}, \ldots, X_{l_m})\) produces \(m\) resultant time series, whom we refer as reconstructed components (RCs) from here after. These reconstructed components are the final decomposition of the original time series and we can revert back to the original time series by summing them up (2.5).

\[
s_n = \sum_{k=1}^{m} x_n^k
\]  

(2.5)

2.3.5 Why SSA?

SSA is a non-parametric, data-adaptive time series decomposition and analysis method introduced in [18]. SSA decomposes a time series into multiple interpretable additive components, which usually can be considered as the trend components, phase, and amplitude-modulated oscillatory components, and the unstructured random noise components [47]. An important feature of SSA is that trends obtained in this way are not necessarily linear. As SSA is capable of separating out non-linear trends and oscillatory components it is an ideal method to decompose our pressure derivative data into a noise-free signal and structured noise components. Also, it does not make any underlying assumption (i.e.; stationarity, linearity, normality) about the time series of interest. The basis used in SSA to decompose the signal is computed independently for each time series in consideration [142]. Due to the data-adaptive property, SSA is highly suitable for our purpose as our data observations have a high amount of variation in their duration, amplitude offset, rate-of-change, and noise structure.

2.4 Structured Noise Detection

An ideal noise-free pressure data is smooth and slow varying. But, often, they are affected by structured noise and unstructured random noise. Thus,
real-life pressure derivative data are often a combination of noise-free signals, structured noise, and random noise. The smooth, slowly varying noise-free signal can be regarded as the underlying trend of the observed noise-corrupted pressure derivative data, and the structured noises are randomly occurring deviations from the usual smooth signal for a brief interval of time and show oscillatory behavior. In Figure 2.2(left), we show the original pressure derivative data as a combination of noise-free signal and structured noises. In Figure 2.2(right), we subtract the underlying trend from the pressure derivative data and plot the residue which has non-zero values only in those segments that correspond to the structured noise in the original signal. From the spectral perspective, the noise-free signal lies in the lower range of the frequency spectrum, whereas the structured noises are in the mid-range, and the random unstructured noises are in the higher range.

SSA is suitable for decomposing pressure derivative data into additive components that can be meaningfully categorized into trends representing the noise-free signal, phase, and amplitude-modulated oscillatory components that capture the structured and random noise components. However, three algorithmic questions remain unanswered.

1. How do we decompose signals efficiently and accurately so the structured noise is separated from the trends? Poorly performed decomposition may combine structured noise and trend in the same component, hindering the noise removal process.

2. How do we identify components that contain only structured noise? There can be one or more components for structured noise.

3. How do we temporally locate the structured noise segments in the structured noise components? The structured noise can be of variable length and appear at an arbitrary time.

We address these three issues in the subsequent subsections.
Figure 2.2: (left) Smooth trend over the noisy signal; (right) structured noise as residue after subtracting trend from the noisy signal (PBU-13-Well-A [19]).

2.4.1 Signal Decomposition

The separation of components in SSA decomposition is a hard-pressed and widely discussed topic in SSA literature [46, 44, 6]. There are several published techniques to measure the quality of the separation, for example, the weighted correlation of the decomposed components [53]. The quality of decomposition is limited by the nature of the signal as well as by the components to be extracted. Exact separability cannot be achieved for real-world signals as there are usually overlaps in their frequency spectrum.

There are no fixed rules on selecting the embedding dimension; rather, it depends on the purpose of the decomposition and the nature of the time series being decomposed. It is generally considered that the larger the embedding dimension, the better the decomposition quality, and the highest quality of decomposition can be achieved when the embedding dimension (M) is equal to half of the signal length [47]. For periodic signal extraction, there are some specific guidelines; for example, M should capture at least one full cycle of the desired lowest frequency component [129].

For a small signal with a complex structure, a comparatively larger M
may produce component mixing. Hence, it is suggested in [47] to select a small M for such a signal. On [139], it is demonstrated that the embedding dimension is related to the frequency bandwidth of each reconstructed component; the frequency bandwidth of each decomposed component is limited to $f_s/M$ where $f_s$ is the sampling rate of the signal. Hence, selecting M can lead to a trade-off between component mixing vs. decomposition resolution. Also, a larger M corresponds to increased computation and time complexity.

Considering the above-mentioned fact, for a small signal with complex structure, in [139], the embedding dimension is selected according to the rule $M = f_s/f_b(6)$, where $f_b$ is determined by the frequency bandwidth of desired signal structure to be extracted. As our main goal is to separate structured noise components, the selection of M needs to be based on noise property. In our data, the sampling frequency is 100Hz, which means a regular sampling of 100 data points per one log cycle of the time axis, and the aperiodic structured noise components observed in this experiment generally expand for 15-40 data points which leads to a frequency bandwidth $f_b \approx 4$Hz. Thus, based on Eqt (6), we use $M = f_s/f_b = 100/4 = 25$ as the embedding dimension. In Figure 2.3[(a)-(e)], we plot the first five reconstructed components (RC), and in Figure 2.3(f), we plot the summation of RC2 through RC5 overlapped with the structured noise residue for M=25 for PBU-13-Well-A data [19].

2.4.2 Structured Noise Component Identification

Once the time series is decomposed into additive components, the next objective is to identify the components that capture most of the structured noise information. This component of the interest identification step is formally called the \textit{grouping}. Similar to the embedding dimension selection step, there is no strict rule about how to perform this grouping rather heuristics are used based on the type of the time series of interest and the purpose of the analysis [139].

An eigenvalue represents the amount of variance captured by the corre-
Figure 2.3: (a)-(e) Reconstructed components (RC1-RC5) after SSA decomposition (f) Sum of RC2 through RC5 overlapped with structured noise residue. (PBU-13-Well-A [19])
Figure 2.4: SSA eigenvalue spectra with 95% confidence limit for all 38 pressure derivative data of Well-A[19].

In general, the largest eigenvalues are associated with the trend components like the smooth, slow varying noise-free signal, the intermediate ones are related to the mid-frequency components like the oscillatory structured noises, and lower values are associated with the high-frequency random noises. Our noise-free signal is supposed to be captured by the trend components with high eigenvalues associated with them as they dictate the overall shape of the time series and, hence, account for most of the variance present in the signal. The structured noise components should be projected onto the oscillating ones, which have mid-range eigenvalues as they account for the short-duration oscillation around the trend.

Therefore, we use this discriminatory property of eigenvalues associated with the components to identify the trend components and the structured noise-capturing oscillatory components. In Figure 2.3, we have plotted the reconstructed components after performing SSA decomposition on PBU-13-Well-A [19]. In Figure 2.4, we plot the eigenvalue spectra of the associated
decomposed components for Well-A’s 38 pressure derivative data. We observe that the first few eigenvalues account for most of the signal’s variance; after those, most of the eigenvalues are close to zero. These almost zero eigenvalues are usually associated with the noisy components. In [85], they have derived a rule to discard those noisy components. Based on their technique, we reject all the components relating to the eigenvalues $\lambda_i$, if $i \geq L$, where

$$L = a \left\{ \frac{\sum_{k=1}^{n} \lambda_k}{\sum_{j=1}^{l} \lambda_j} \geq 0.95 \right\}$$

The trend components usually have higher values in a similar range, and the structured noise components have comparatively lower eigenvalues. In Figure 2.3(a), we observe that the RC1 represents the trend, and in Figure 2.3(f), the summation of RC2 through RC5 matches with the structured noise residue. Also, there is a significant gap between RC1 and the rest of the RCs in the eigenvalue spectra. Hence, to generalize our procedure, we distinguish between the trend and the structured noise components by identifying the largest gap between two consecutive eigenvalues, and then, based on that differentiating point, we group the trend and structured noise components.

### 2.4.3 Structured Noise Localization

Ideally, after SSA decomposition, the trend component would represent the noise-free smooth signal, and the structured noise component should have non-zero values only in segments corresponding to structured noises in the original signal. But often, due to high amplitudes of the structured noises, component mixing occurs, affecting the trend components. Also, in the structured noise components, noise oscillations expand to neighboring regions, for example, where there are flow regimes between two structured noises or flow regimes close to high-amplitude structured noises. Nevertheless, the amplitude of oscillatory components is still highly correlated to the likelihood of occurrence of structured noise. Thus, to localize structured noise segments
from the structured noise components, we use a threshold value to compare against the absolute value of the summation of the structured noise components, and the sample data points exceeding the threshold value are identified as structured noise.

**Single Sample Threshold Selection**

In our pressure derivative data, in some observations, the structured noises have a high amplitude, whereas, in some observations, they have a low deviation from the actual trend. Though the structured noise properties maintain some similarities in a single well, the variation becomes prominent between the observation of two different wells. So, it is quite impossible to obtain a globally optimal threshold. In summary, we need a mechanism to modify our threshold from well to well. To address this issue, we employ a semi-supervised learning approach to select the threshold by manually evaluating noise detection performance for a range of threshold values using one PBU data per well. Then, we select the threshold which results in the best performance to be used as a fixed threshold for the rest of the observation of that well. Though this process might seem laborious, in practical scenarios, we can automate the noise detection procedure for a few dozen other pressure derivative observations by manually evaluating only a single data observation.

**Boundary Refinement**

The selected structured noise components have high values in the center of the noise segment, and the region where noise transcends into signal has comparatively lower values. In Figure 2.4(left), we have plotted the absolute value of the summation of the oscillatory components, which portrays the aforementioned scenario. To tackle this problem, we calculate a windowed average of the absolute value of the summation of the oscillatory components, which is plotted in Figure 2.4(right). In this case, the boundary region of
structured noise segments has increased amplitude than before, which helps in the precise localization of the whole noise segment.

2.5 Experiment and Results

We use real log-log pressure data and the semi-log derivative of log-log pressure data from two real gas wells and one oil well for quantitative accuracy calculation. Moreover, we use three separate oil well pressure data for qualitative evaluation by the domain expert.

2.5.1 Data Collection and Preprocessing

The pressure data is extracted from the shut-in period of the continuous stream of pressure data recorded by the permanent downhole pressure gauge installed in wells drilled in oil and gas fields. The extracted shut-in period pressure values were converted into pressure change data with reference to
the pressure value at the beginning of the shut-in period. The pressure change data and time axis were converted into a log scale, producing the log-log pressure change data. Finally, the semi-log derivative of this pressure change was calculated using the method of Bourdet [16] to produce the log-log pressure derivative plot. From this plot, we remove the data points with an undefined value, which occurs due to a negative change in pressure value around that point in time. Also, we perform linear interpolation to create evenly spaced data points from irregularly sampled and log-scale transformed data points.

2.5.2 Data Labeling and Accuracy Calculation

We use *F-score* measure to calculate the accuracy of our proposed method. We use log-log pressure derivative data where domain experts have manually labeled structured noise and flow regime segments. We label the flow regimes in addition to the structured noise segments as some segments are present...
Table 2.1: Accuracy of Structured Noise Detection on real pressure build-up data from three different wells - marked as A, B, and C [19].

<table>
<thead>
<tr>
<th>Well</th>
<th>No Lower Bound</th>
<th>Minimum Threshold &gt; 0</th>
<th>Minimum Threshold &gt; 0.02</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>69.73%</td>
<td>74.07%</td>
<td>75.02%</td>
</tr>
<tr>
<td>B</td>
<td>63.22%</td>
<td>66.22%</td>
<td>67.14%</td>
</tr>
<tr>
<td>C</td>
<td>80.36%</td>
<td>80.36%</td>
<td>80.59%</td>
</tr>
<tr>
<td>Average</td>
<td>71.1%</td>
<td>73.55%</td>
<td>74.25%</td>
</tr>
</tbody>
</table>

Figure 2.7: Automatic noise detection result on real pressure build-up data from Well-A [19]; (left) structured noise due to unknown reason; (middle) structured noise due to unknown reason; (right) structured noise due to pressure leak at flow line.
in the pressure derivative data, which are neither noise nor flow. While calculating the accuracy of our algorithm, we consider only the labeled flow regimes and structured noise segments.

We compare our detected structured noise segments with the labeled ground truth data and calculate true positive ($TP$), false positive ($FP$), and false negative ($FN$). Then, we use the ($TP$), ($FP$), and ($FN$) to calculate precision, recall, and finally, the F-score.

\[
\text{precision} = \frac{TP}{TP + FP}, \quad \text{recall} = \frac{TP}{TP + FN}
\]

\[
F\text{-score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}
\]

### 2.5.3 Threshold Selection

The threshold parameter is selected in a semi-supervised fashion where one pressure derivative data is used to select the threshold parameter for the rest of the data of the same well. First, the user manually labels one selected pressure derivative data. Then, noise detection is performed on that data for a range of threshold values, and the F-score is calculated for each of the threshold values to select the threshold corresponding to the highest F-score. This best-performing threshold value is the default for the rest of the
data of that well. In Figure 2.6(left) for PBU-13-Well-A, we observe that
the maximum F-score is achieved for the threshold value 0.06. So, if we
select this PBU observation for the threshold selection step, our automatic
threshold value for the rest of the observation from Well-A would be 0.06.
Similarly, in Figure 2.6(right), we see that for PBU-24-Well-A, the optimal
threshold is 0.05.

2.5.4 Results

Our method’s final noise detection performance depends mainly on the se-
lection of threshold, which can be selected using any PBU data. To get an
overall performance evaluation for each well, we use each PBU once for man-
ual threshold selection. Then, we use that threshold as the fixed threshold
for the rest to perform noise detection and accuracy calculation. Finally, we
calculate the average accuracy for all the thresholds. In table 5.1, we show
the average accuracy for three different wells, where we notice that, without
using any constraint on the threshold value, the average accuracy among the
three wells is 71.1%. However, if we put some lower bound on the threshold
value, it increases up to 74.25%. For blind performance analysis, we run our
method on three more oil well [19] data, which was evaluated by domain
experts to assert the usability of our algorithm in practical application. In
Figure 2.7 and 2.8, we give a few graphical demonstrations of our structured
noise detection performance.

Runtime. The run time of our method is always less than 0.05 seconds for
a single PBU averaged over several runs on different well data (using a core
i5 2.70 GHz desktop computer).

2.6 Discussion

In this section, we discuss various aspects of our proposed method.
Figure 2.9: Histogram plot of the number of times a threshold value is selected using single-sample learning: (top) Well-A, (middle) Well-B, (bottom) Well-C [19].
2.6.1 Parameter Sensitivity

In Figure 2.9, for three different wells, we plot a histogram that shows how many times a particular threshold value is selected from a well. In Figure 2.10, we plot the overall accuracy for a range of threshold values for three wells. In Figure 2.9(top) for Well-A, we observe that the median of the selected threshold value is 0.06, and the noise detection accuracy is also highest around that threshold value. This same scenario is noted for the other wells, too, which implies that if we select an observation that bears some structured noises, it would, in turn, produce a good enough threshold for the rest of the observation of that well.

2.6.2 Interpretability

The solitary parameter of our method, the threshold value, has an intuitive interpretation which the user can readily control to tune the outcome according to specific well characteristics. Here, a higher threshold value implies that the method will be more selective in labeling data segments as noise, whereas a lower threshold value implies that the algorithm will be more inclined toward detecting a particular data segment as structured noise. Due to this interpretable user control mechanism, this method can be effectively deployed in practical field usage where the users can fine-tune the method accordingly without requiring knowledge about the underlying algorithm.

2.7 Conclusion

In this work, we show a practical example of using signal processing techniques to improve the usability of data science methodologies on process monitoring datasets. Our developed method is adaptive to the variation of structured noises, and the run-time is not dependent on the amount of noise present in the data. The proposed method is fast, computationally inexpensive, and requires minimal manual intervention, which makes it perfectly
Figure 2.10: Accuracy of structured noise detection performance for various threshold values on Well-(A, B, C) [19]. The maximum accuracy values (shown by arrows) vary across wells.

suitable for practical day-to-day deployment. The solitary parameter is readily interpretable and intuitive to fine-tune. Experimental evaluation, as well as assessment by domain experts, have validated the accuracy and effectiveness of the method. In the future, we will use this method to increase the accuracy of classification and clustering tasks on additional datasets, which will demonstrate the generalizability of this method.
Chapter 3

Seismic Phase Identification for Automated Monitoring

3.1 Introduction

Real-time seismic signal processing is a key element of the geophysical monitoring required for early warning systems for earthquakes, underground mineral exploration and mining, and nuclear explosion monitoring. Seismic signal processing pipelines involve several sequential steps that start with signal (e.g., from an earthquake) detection from raw seismic signals recorded at a seismic station, and in the end, produce a formalized event bulletin for real-time alarm generation as well as future analysis. Figure 3.1 shows a typical pipeline. Phase identification is a key step in this pipeline subsequent to the signal detection step, which can be framed as a classification problem that takes a detected seismic signal as input, and outputs the phase label. Phase identification is required for proper utilization of the downstream steps of the pipeline, for example, earthquake location estimation, tomographic studies, and understanding of the Earth’s crustal and upper mantle structure [38]. A successful phase classifier must classify a detected seismic waveform into shear or transverse waves (ending with S in Figure 3.1.right) and compressional or longitudinal waves (ending with P in Figure 3.1.right), and all their
subtypes.

**Single Station vs. Array.** At present, in operational systems, seismic phase identification is heavily dependent on the use of multiple close-by seismic monitoring stations, forming an array of stations [86]. High-quality arrays enable better detection and improved signal-to-noise-ratio, and estimation of phase velocity and direction of arrival; which greatly benefit both phase identification and association. Relative arrival times of seismic phases at different arrays and of different detections at the same array, together with their directions of arrival, are used to accurately classify and associate phases [116]. Unfortunately, most new stations added in dynamic response to changing monitoring needs, such as in oil fields or novel seismic sources, will be individual stations rather than arrays. This reduces the processing pipeline’s performance because phase identification is less accurate for single stations than it is for arrays. In this work, we consider automated phase identification on data collected at a single station to enable rapid deployment addressing dynamic needs.

**Figure 3.1:** (left) Typical seismic data processing pipeline. Our objective is to develop a Machine Learning model for phase identification. (right) Travel times with respect to distance for various seismic phases. Phases ending with P can commonly be categorized as P, and phases ending with S can commonly be categorized as S.

**Fine vs. Coarse Identification.** In addition, most existing research work on automated seismic phase identification considers only the two high-level categories, while most monitoring applications require finer identification
For global monitoring, seismic signals are initially classified as teleseismic P (including more complex phases such as PkP and PkikP) or regional P or S (i.e., Pn or Sn). Refinements to the phase identification to add teleseismic S and crustal P and S phases (i.e., S, Pg, and Lg) are made much later in the processing pipeline. In this work, we consider classifying into finer phases at the initial identification step. In addition, phase detection and phase identification are usually sequential steps. Seismic signal detection algorithms are very well developed [99, 93, 117], and signal detection is a key module in the processing pipeline. In this work, we consider phase identification only after detection, unlike some recent works that address detection and identification jointly [106, 107].

**Global vs. Regional:** Lastly, most existing research work focuses on local and near regional methods that only observe two major phases (local P and S, or occasionally regional P and S), and uses these phases to determine source location and origin times, mostly due to constrained focus. However, in a global monitoring application, all kinds of teleseismic (>1000km), regional (>200km), and local waves can arrive at various degrees of temporal overlap with arbitrary arrival order.

Figure 3.1 (right) illustrates some of the complexity of global seismic arrivals. Identifying the correct phase from a complex waveform containing multiple arrivals is difficult for global monitoring applications. We use data from the International Monitoring System (IMS) network, a global network operated by the Preparatory Commission of the Comprehensive Nuclear-Test-Ban Treaty Organization (CTBTO). IMS data are processed in near real-time at the International Data Centre (IDC) in Vienna, with initial detection, phase identification, and association performed automatically and then curated by human analysts. Our method can achieve significant identification accuracy even in such a complex application as presented by global seismic network data.

**Challenges to this research.** First, there is no publicly available dataset of seismic signals categorized into six-phase classes. The main hindrance
to curating such a dataset is the relative rarity of some phases compared to others. For example, in a continuous one-year time frame of the IMS catalog, 128K seismic waveforms are classified as P, and only 1300 are classified as S. Also, the manual labeling of these distinct phase types requires a depth of knowledge and rigorous training [15]. Second, the phases within broader hierarchical categories share highly similar spectral and amplitude properties. Also, seismic signals originating at different geolocations exhibit different propagation effects, resulting in dissimilar signal properties for the same phase type. There are also differences among waveforms of a single type at different distances. The current bottleneck in processing is the association of seismic phases with their most likely sources, which could be improved by more accurate initial phase identification.

**This work.** To tackle these challenges, we focus on a few different aspects. First, we have curated a small-scale yet balanced seismic phase dataset collected from IMS network data. From an imbalanced collection of more than 200K seismic events, we have narrowed it down to 16K events with finer and balanced phase labeling. We use Continuous Wavelet Transforms (CWT) to obtain a time-frequency representation from raw seismic time series to utilize both temporal and spectral information. The CWT representation has been shown to be resilient to dynamic noise in waveforms [92]. We design an end-to-end deep neural network to perform phase identification using these CWT representations. We leverage the power of Convolutional Neural Network (CNN) to capture low-level features from the CWT representations that are invariant to frequency, scale, and position [111]. However, due to locally constrained receptive fields, CNNs are inadequate in modeling long-term temporal dependency, whereelse seismic signals contain distinctive temporal patterns across different phase-types [106]. To mitigate this limitation, we incorporate LSTM on top of the CNN as LSTM can effectively model long-term temporal patterns and dependencies.

Our proposed method **FASER** can perform fine-grained phase identification using single-station data from the global seismic network. Due to
minimal preprocessing requirements and instantaneous output generation, it can be readily integrated into the existing real-time seismic signal monitoring pipeline. We show a comprehensive experimental evaluation of FASER using a real dataset in comparison with existing methods to validate improved performance. We justify the generalizability of FASER by demonstrating case studies for applications in novel operating conditions. To the best of our knowledge, this is the first attempt to perform fine-grained phase identification using single-station seismic signal data.

3.2 Related Work

In general, the methods for seismic phase identification can be broadly categorized into two types, (1) heuristic template matching and statistical analysis based methods, and (2) deep learning based methods.

**Statistical and Heuristic Methods.** Since the early inception and development of seismic signal monitoring, several rule-based and physics-driven methods have been proposed for seismic phase identification. In [104], data-adaptive polarization filtering methods have been used for phase detection tasks. In [8], the difference between the short-term average (STA) and the long-term average (LTA) of the seismic signal has been used for automated detection. Several methods have used higher-order statistics like kurtosis and skewness [78, 76]. Also, few methods have used frequency domain information [144, 92]. However, these methods perform poorly in the presence of noise and when the seismic events are of low magnitude. A few other works have proposed a similarity search based template matching method [102, 41]. However, such methods are heavily dependent on a prior collection of sample signal templates and often fail to generalize when used for phase detection at novel stations. Also, the pairwise similarity search with each sample template renders these methods computationally intensive and inefficient for real-time monitoring.

**Deep Learning based Methods.** Recently, multiple deep learning meth-
ods have been proposed to address the aforementioned shortcomings in the context of phase detection and identification [99, 93, 86, 33]. A deep learning based grid-free phase association method for phase identification has been proposed in [107]. In [106], a CNN based architecture has been used for phase identification from one-dimensional seismic signals. More recently in [34], the use of time-frequency representation and CNN has been explored. These methods have shown promising results compared to previously used statistical and heuristic-based methods as CNNs can effectively model low-level structured patterns into high-dimensional embedding. However, due to locally constrained receptive fields, CNN cannot capture the long-term temporal patterns. Thus, these models cannot fully exploit the higher-order temporal structures in seismic signals, distinctive across different phase types. Moreover, none of these methods perform fine-grained phase identification.

3.3 Methods

In practice, the seismic phases are manually labeled by experienced analysts using multi-modal information, i.e., signal amplitude, frequency components in the signal, the distance between the event origin and monitoring station, depth of the event origin, etc. However, for generalized and real-time phase identification, there exist a few challenges to producing such information a priori. Depth and source location estimation are intricate regression problems requiring complex analysis. Previously, the effectiveness of time-frequency representations has been demonstrated in a multitude of seismic signal processing tasks [118, 92]. Therefore, we use the CWT to obtain spectral-temporal features, as it produces higher spectral resolution and more precise temporal localization than other time-frequency transformations (e.g., Short-Fourier Transformation) [92]. We use a composite CWT image, where the vertical component CWT coefficients are represented by red brightness, and the two horizontal components are represented by the brightness of green and blue (See Figure 3.2).
Figure 3.2: Input images of waveforms are created by taking CWT of individual channels (i.e. BHZ, BHN, BHE).

The time-frequency representation of seismic events contains distinct structured features based on their phase type. However, at a low level, these features are highly overlapping. CNN has been widely used in the domain of Computer Vision [74, 55], Natural Language Processing [35, 59], Speech recognition [2] and other related domains to learn high-level features from raw structured input for better contrastive representation learning. As CNN can model the local correlation of spatial and temporal patterns, it is highly suitable for our two-dimensional CWT feature maps, which contain incremental time information on one axis and frequency information on the other.

However, CNN is inadequate in learning long-range temporal dependencies due to their locally constrained receptive fields [111]. Nevertheless, the long-term temporal patterns in the seismic signals, which are well preserved in CWTs, are vital distinctive features across different phase types as showcased in Figure 3.3. To circumvent this limitation, Recurrent Neural Networks (RNN) have been instrumental in modeling the temporal dependencies by using the cyclic feedback mechanism from previous time-step inputs. LSTM, an improved variant of vanilla RNNs, is capable of learning and modeling long-term temporal patterns and dependencies [68, 88]. However,
Figure 3.3: CWT of pairs of examples from all phase types. Compressional or longitudinal waves (P, Pn, Pg) are dominantly red due to high vertical component amplitudes. Transverse waves (S, Sn, Lg) are dominantly green/blue due to high horizontal component amplitudes.
It has been shown that higher-level features can be helpful in learning the underlying factors of variations within the input, which should make it easier to learn temporal structures between successive time steps [138]. Thus, oftentimes CNN has been successfully used as preceding layers before more complex sequential models to reduce the local temporal and frequency variations [111].

Motivated by these aforementioned successful use cases, we use a combination of CNN and LSTM in an end-to-end network. First, we utilize CNN to identify low-level spectral-temporal features that are invariant to frequency, scale, and position. Afterward, we organize the output features obtained from CNN into sequential features preserving the temporal ordering. We feed these higher-level sequential representations of low-level structured patterns as input into the LSTM. By utilizing the cyclic feedback mechanism in-between consecutive time steps, LSTM can better model the long-term temporal correlation in the seismic signal. Finally, we feed the output from each time step into dense layers to make the final output prediction.

### 3.3.1 Convolutional Neural Network

CNN [74] perform convolution operations on the input feature map using fixed-size kernels (learned during the training step) to produce higher-order representations. Convolution operations are usually followed by a non-linear activation function and max-pooling layers. The use of an activation function introduces non-linearity, and the max-pooling reduces sensitivity to temporal or spatial variation. CNN is adept at learning local structural relationships and are invariant to feature scaling, which reduces the dependency on heavy data preprocessing and feature engineering [146].

### 3.3.2 Long Short-Term Memory Network

LSTM networks are an improved variant of traditional RNN [58]. RNNs can model temporal dependencies in the data by utilizing feedback connection
by considering both inputs at the current time step as well as the output of the last time step’s hidden state. However, vanilla RNNs suffer from the vanishing gradient problem, which prevents the model from learning long-range dependencies. LSTM tackles this problem by introducing three gating mechanisms to update the memory cell $c_t$ and hidden state $h_t$ at each step $t$ based on the current time step input $x_t$ and the previous time step’s hidden state output $h_{t-1}$. Each LSTM unit is composed of a memory cell and three main gates: input, output, and forget. The input gate $i_t$, forget gate $f_t$, output gate $o_t$, memory cell $c_t$ and hidden state $h_t$ at step $t$ are computed as follows:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$  \hspace{1cm} (3.1)  
$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$  \hspace{1cm} (3.2)  
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$  \hspace{1cm} (3.3)  
$$c_t = f_t \odot c_{t-1} + i_t \odot \text{tanh}(W_c \cdot [h_{t-1}, x_t] + b_c)$$  \hspace{1cm} (3.4)  
$$h_t = o_t \odot \text{tanh}(c_t)$$  \hspace{1cm} (3.5)  

Here, $W_i$, $W_f$, $W_o$ are the weight matrix and $b_i$, $b_f$, $b_o$ are the bias of input, forget and output gate, respectively, and $\sigma$ is the logistic sigmoid function, tanh is the hyperbolic tangent function, and $\odot$ denotes the element-wise multiplication. By this architecture, the LSTM manages to create a controlled information flow by deciding which information it must forget and which information to remember. To understand the mechanism behind the architecture, we can view $f_t$ as the function that controls to what extent the information from the old memory cell is going to be thrown away, $i_t$ controls how much new information is going to be stored in the current memory cell, and $o_t$ controls what to output based on the memory cell $c_t$. 

41
Figure 3.4: Proposed model architecture.
3.3.3 Proposed Architecture

Our proposed architecture is comprised of four convolutional layers, followed by two LSTM layers and three fully connected dense layers. In Figure 5.4, we present our proposed architecture. The input to the networks is the CWT representations obtained from the bandpass-filtered seismic signal waveforms. A detailed description of data preprocessing is presented in the following section. In each of the four convolutional layers, we use kernels of size $3 \times 3$ with a stride of $1 \times 1$ and zero paddings. Each convolution layer is followed by a batch normalization layer, a Rectified Linear Unit (ReLU), and a two-dimensional max-pool layer. In the first two max-pool layers, we use a kernel size of $2 \times 2$ with a stride of $2 \times 2$. However, in the last convolution layer, we perform max-pooling only along the temporal dimension with a kernel of size $1 \times 2$ and stride of $1 \times 2$. The first convolutional layer has eight filters, and we double the filter number on each subsequent convolutional layer to keep the number of parameters in each convolutional layer the same as we reduce the input image size by half after each convolution layer due to max-pooling.

The output from the final convolution layer is then passed into the LSTM layers preserving the temporal order. The first LSTM layer consists of 32 hidden units and the second LSTM layer consists of 16 hidden units. We use $sigmoid$ and $tanh$ as the recurrent and output activation function of the LSTM correspondingly. We use a 50% recurrent dropout in the LSTM layers. Both LSTM layers are unrolled for 15 steps as the input feature map to the LSTM has a temporal dimension of fifteen. Both LSTM layers return sequences in each unrolling step. These sequences are flattened before feeding into the dense layers. We stack three dense layers, each with 64, 32, and 6 hidden units consecutively. Each of the dense layers is preceded by Batchnormalization and ReLU activation functions with a 20% dropout rate. We use the softmax activation function in the final dense layer to obtain output probabilities for each phase type.
3.4 Experiments and Results

In this section, we perform experimental analysis on real seismic data to show the effectiveness of our proposed method in comparison with existing baseline methods.

3.4.1 Dataset Description

The dataset is curated from 10 years of continuous seismic data collected at the 155 stations of the International Monitoring System (IMS) [1]. These consist of 46 primary stations, 24 of which were arrays and 105 auxiliary stations, 98 of which were 3-component stations. The dataset includes 80TB of uncompressed seismic waveforms and the comprehensive IMS catalog, with arrival times and phase labels curated by human analysts for over 8 million seismic event detections. This dataset includes the comprehensive IMS catalog, with arrival times and phase labels curated by human analysts for over 8 million seismic event detections. From these 8M seismic events, we filtered out 175K fine-grained seismic phase labeled data. However, among these, the P-phase was predominant, with 128,120 occurrences, while the S-phase had only 1,306 occurrences. To ensure a balanced dataset between crustal, regional, and teleseismic compressional and transverse phases (i.e., Pg, Lg, Pn, Sn, P, and S), we used all labeled S-phases and randomly sampled around 2,500 waveforms from each of the other phases, for a total of 16,304 phases. As the spectro-temporal features of all the phases are highly nuanced, no data augmentation was performed to maintain integrity for practical application scenarios.

3.4.2 Data Prepossessing

Our raw input data are 60-second, three-channel time series sampled at 40Hz. Following conventional seismic signal pre-processing techniques, we filter the waveform from each channel (0.4Hz to 10Hz). As these seismic signals were
generated by events of different magnitudes and recorded at stations spread across the globe, the amplitude is not relevant to phase identification in this first step. We first detrend each sample and remove the mean. We then max-normalize the data across each channel, thus retaining the relative amplitudes among components of a station. Afterward, use the CWT to obtain spectral-temporal feature representation.

3.4.3 Baseline Methods

In order to validate the performance of our method, we have compared our method with the following baseline methods.

- **XGBoost (XGB) [24]**: XGBoost is an optimized ensemble based model that has produced state-of-the-art methods for many classification tasks. We convert the multi-dimensional CWT representations into one-dimensional features as input for XGB. Afterward, we perform standardization across each feature dimension.

- **MLP [54]**: Multi-layer perception (MLP) is a feed-forward neural network. We use a two-layer MLP with the same input feature as XGB.

- **CNN [34]**: CNN-based methods have been previously used in related seismic signal classification tasks. In [34], a CNN-based method has been used for two-class phase identification. We use the same CNN architecture used in this paper to compare against our method.

- **LSTM [58]**: LSTM methods are highly suitable for temporal data modeling and have produced state-of-the-art accuracy in many time series classification tasks. We use the most popular stacked LSTM architecture for comparison. The final output is fed into a fully connected layer to generate an output label.

- **CRED [93]**: In [93], a ResNet-BiLSTM architecture has been proposed for seismic event detection where it achieved state-of-the-art performance. We use the same architecture for our phase identification task.
3.4.4 Experimental Settings

The hyper-parameters of the model were selected empirically by grid-search on the validation set. We use the Adam optimizer [70] with an initial learning rate of 0.01 and with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. We apply $L_2$ regularization with $\lambda = 0.001$, and we use categorical cross-entropy [48] as the loss function with a training batch size of 256. The XGBoost model was trained until convergence. The neural network models were trained for a maximum of 200 epochs with early stopping on the validation set.

We use 10-fold cross-validation to measure the performance of our method and report the average. In each fold, we use 80% of the data for training, 10% for validation, and 10% for testing. We perform random stratification to ensure class balance in the training-validation-test split. All the experiments were performed on a core i5 2.70 GHz desktop computer with 8GB NVIDIA GeForce GTX-1070 GPU.

3.4.5 Evaluation Metrics

In our experiments, following conventional practices for classification tasks, we use accuracy as the primary performance metric. However, as there are minor class imbalances in the dataset, we also calculate macro (calculated individually for each class and averaged afterward) precision, recall, and F-score [27].

3.4.6 Results

In Table 5.1, we report the performance of our method in comparison with the baseline methods, where the highest performance for each metric is cyan-colored. We observe that FASER consistently outperforms all the baseline methods across all performance metrics. To closely probe the performance of FASER across each class, in Figure 3.5(left), we show the confusion matrix for the test cases of a randomly split 80-10-10 train-validation-test scenario.
<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Score</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>XGB</td>
<td>68.4</td>
<td>67.2</td>
<td>67.2</td>
<td>67.2</td>
</tr>
<tr>
<td>MLP</td>
<td>76.2</td>
<td>75.6</td>
<td>75.3</td>
<td>75.6</td>
</tr>
<tr>
<td>CNN</td>
<td>75.2</td>
<td>75.2</td>
<td>75.2</td>
<td>75.2</td>
</tr>
<tr>
<td>LSTM</td>
<td>75.7</td>
<td>74.3</td>
<td>75.0</td>
<td>75.3</td>
</tr>
<tr>
<td>CRED</td>
<td>81.3</td>
<td>80.2</td>
<td>80.7</td>
<td>81.5</td>
</tr>
<tr>
<td>FASER</td>
<td>84.6</td>
<td>81.6</td>
<td>83.1</td>
<td>82.8</td>
</tr>
</tbody>
</table>

Table 3.1: Performance metric comparison of FASER against baseline methods. FASER outperforms existing baseline methods in all four metrics: Precision, Recall, F-Score, and Accuracy. The best algorithm for each metric is colored in cyan.

It is noticeable that the majority of classification error is within the sub-classes of compressional (P, Pg, Pn) and transverse (S, Sn, Sg) waves. This performance is in coherence with the intuitive notion of similar spectral-temporal features within both broader classes.

In Figure 3.5(right), we plot the t-SNE visualization [83] of the same 10% test cases considering the activation values of the last layer before the prediction layer as deep embedding [137]. The compressional and transverse wave signal samples are well separated in the deep embedding space with a high margin with only a few mispositioned overlaps. However, as the spectral-temporal features within the sub-classes of transverse and compressional waves are often overlapping, we notice soft boundaries among the intra-sub-classes along with higher overlaps among the samples. Although the separation among finer-phase types is not well-established, it is evident from this embedding projection that our proposed method is adept at learning higher-order separable representations.
Figure 3.5: (Top) Confusion matrix for the test cases of a randomly split 80-10-10 train-validation test scenario. (Bottom) t-SNE visualization of the same 10% test cases considering the activation values of the last layer before the prediction layer as deep embedding [137]. The compressional and transverse wave signal samples are well separated.
3.5 Case Study: Novel Operating Conditions

In this section, we demonstrate practical use cases for our developed method in real-world applications when various novel scenarios emerge. In particular, we consider the following two novel scenarios: (1) If a monitoring agency adds a new station at a new location, will our method work without any calibration? (2) If new seismic sources occur in historically aseismic regions, will our method identify novel arrivals from a new source?

3.5.1 Performance at Novel Station

In our dataset, we have phase arrival signals recorded at 125 different stations across the world. To test how our method would perform at a novel station, we hold out signals at one station while training on signals at all other stations. We show the empirical cumulative distribution function (ECDF) of stations for various accuracy levels in Figure 3.6. We compare two classifiers in this experiment. The nearest neighbor classifier compares a test image with all training images to find the best match under the Euclidean norm and labels the test image with the phase of the best match. The classifier achieves approximately default classification accuracy of 17% for the majority of the stations. This suggests that the nearest neighbor classifier cannot classify signals at a new station based on historical data at other stations. In contrast, FASER achieves approximately 72% accuracy for the majority of the stations, suggesting single station analyses of seismic data may be useful in response to dynamically changing monitoring needs. The convolutional layers in our architecture extract local features from the images, unlike relying on a global one-to-one alignment of the images in the Euclidean space, as in the case of the nearest neighbor classifier.

The achieved accuracy of 72% for the majority of the station is significant for the IMS processing pipeline, as IDC analysts relabel 62% of the initial phases detected by the current automated algorithm. Moreover, only 38% of the initial phases remain the same, indicating that the initial phases are
Figure 3.6: Performance comparison of FASER with naive nearest neighbor classifier for application in novel stations. The plot shows the cumulative distribution function (CDF) for a percentage of stations having smaller than a given accuracy. FASER achieves an approximately 72% accuracy for the majority of the stations in contrast to 17% for the nearest neighbor classifier.
correct with 38% accuracy. FASER almost doubles the accuracy for a novel station of the current system’s phase identification accuracy for an existing station. We show the held-out performance at each station in Figure 3.7(left). Most stations achieve higher accuracy (>0.7) when there are several closer stations. In contrast, isolated stations such as the one in the South Pacific suffer from poorer performance.

### 3.5.2 Performance on Novel Sources

Most earthquakes originate along fault lines, while the rest of the earth is quieter. Novel seismicity in previously undocumented areas is intriguing. Hence, we evaluate our model by holding out regions of the earth for testing. For each held-out region, we train our model with data from the rest of the world and test the performance of our model on seismic events in that region. For this experiment, we divide the earth into 12×12-degree grids. If a grid cell is not seismically active (i.e., not enough data), we exclude the region from testing. In Figure 3.7(right), we show the world map where the shaded grid cells are held out, one at a time. The average hold-out
accuracy is 77.27%, with a standard deviation of 4.13%. More importantly, this suggests that our model is well suited for novel seismicity with little or no prior recorded events. We test on 46 cells of the 12×12 degree grids, which cover most of the known seismic events recorded at the NEIC (National Earthquake Information Center) for a three-year period.

3.6 Conclusion

In this chapter, we present a method to perform fine-grained seismic phase identification, which can be readily integrated into existing seismic signal processing pipelines. As seismology evolves into a big-data-driven science, deep learning methods are becoming an indispensable part of next-generation seismic monitoring systems. This work shows a practical example of integrating deep-learning methods in an existing semi-autonomous system to achieve complete autonomy. We demonstrate empirical evaluation of our method with a real-world dataset where it outperforms existing methods. Our method reduces the dependency on using array-based methods, which inhibits precise monitoring for regions with limited monitoring stations. It also reduces the dependency on large collections of manually curated template sets and presents the opportunity to use transfer learning for stations with limited labeled data. Due to the minimal preprocessing requirements and faster prediction generation, it is highly suitable for a real-time monitoring pipeline. In the future, with the use of larger datasets, more complex models would produce higher accuracy as well as better generalizability. Moreover, the introduction of interpretable models would be highly suitable for downstream analysis.
Chapter 4

Few-shot Learning for Seismic Phase Identification

4.1 Introduction

In the previous Chapter, we presented FASER, a supervised learning method for seismic phase identification, which achieves state-of-the-art accuracy. However, as the conclusion outlines, FASER has some limitations and improvement opportunities. In this work, we identify the factors contributing to the limitations of FASER in practical use-case scenarios and focus on addressing those limitations for further performance improvement.

In the practical use case scenario, seismic phase identification is performed in stations spread across the globe. The labeled data amount across different stations and regions differs widely due to variances in seismic activity, the lifetime of monitoring stations, and labeling constraints. In Figure 4.1, we plot the location of seismic stations, which are part of the International Monitoring Stations (IMS) [1]. Here, the node size of a station is proportional to the number of labeled instances in that station, and we observe a large variation in the number of labeled instances across stations. This variation in training data introduces biases in the supervised models, for example, FASER. Figure 4.2 depicts the average accuracy for station groups based on
the amount of labeled data. Here, the performance is skewed towards stations with higher labeled examples.

The accuracy further decreases for novel stations from a distinct geographic location from the training stations. In Figure 4.3, we show clusters of stations based on their geographic location, where different clusters have different color labels. In Table 4.1, we report the accuracy when the test and train datasets are sampled from distinct geographic locations or clusters compared to training and testing on the same clusters. The performance is higher when the training and testing samples are from same clusters.

The reason for this performance drop lies in the characteristics of seismic signals. The seismic signals are primarily affected by the characteristics of the seismic event source and the propagation paths from the sources to the stations. The waveforms of the same event may vary when observed at different stations because the waves take different paths through the earth and depend on the relative position of the seismic source and monitoring

Figure 4.1: Global Monitoring Station network across the world. The size of the node is proportional to the labeled sample number.
Figure 4.2: The average accuracy of FASER for stations grouped based on labeled data amount. Here, the accuracy is higher for stations with higher labeled examples and vice versa.

Figure 4.3: Station clusters based on geographic proximity. The stations of the same cluster have a similar color.
Table 4.1: Accuracy for the held-out validation samples from in-training stations and for the samples from the novel test stations from a different cluster.

<table>
<thead>
<tr>
<th>Test-sample</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-training</td>
<td>84.7</td>
</tr>
<tr>
<td>Novel</td>
<td>75.6</td>
</tr>
</tbody>
</table>

station. Therefore, seismic waveforms of similar phase types often show distinct signal properties depending on the geographic region and station location.

In FASER, we focus on learning latent feature representations from raw waveforms to reduce high-dimensionality and overfitting to improve generalizability. However, this approach requires large amounts of labeled data to achieve generalizability across stations. However, the existing seismic data catalog is not sufficient for that purpose. Over time, new stations are being established in novel regions in response to dynamic monitoring needs, such as oil fields or novel seismic sources. These new stations have very few labeled examples and expectedly suffer from lower phase identification accuracy.

To address these challenges, we aim to exploit both station-agnostic high-level similarities and station-centric properties of seismic signals. We present a few-shot learning framework for seismic phase identification that can effectively adapt to novel stations with limited labeled examples via improved utilization of regional data points. We employ an episodic few-shot training setup to simulate learning from limited samples to handle the data distribution imbalance across stations effectively. To adopt a few-shot learning framework, we pose the phase classes in novel stations as novel classes to facilitate learning station-specific features. We jointly optimize toward learning station-agnostic and station-centric features that can effectively handle catastrophic forgetting and overfitting. We mimic the actual test environ-
ment during training by sub-sampling examples from data-rich stations.

We show empirical evaluations using real-world data covering various geographic locations. Our proposed method can perform fine-grained phase identification using single-station data from the global seismic network. Due to minimal preprocessing requirements and instantaneous output generation, it can be readily integrated into the existing real-time seismic signal monitoring pipeline. We show a comprehensive experimental evaluation using a real dataset compared to existing methods to validate improved performance. Our model improves performance for new regions with a limited amount of labeled examples while maintaining the accuracy for data-rich regions. We summarize our main contributions below:

- To the best of our knowledge, this is the first attempt to formulate seismic phase identification as a few-shot learning problem. By leveraging an episodic training setup, we enforce learning from limited training samples to handle the data distribution inconsistency across stations.

- We learn both station-specific and station-agnostic discriminative features, where the former can model geolocation-dependent signal properties and the latter can capture global features to hedge against noise and outliers in the few examples.

- We show significant performance improvement for regions and stations with limited examples while increasing the overall accuracy.

4.2 Related Work

Few-shot Learning. A wide variety of applications require learning a supervised model using a limited number of labeled instances per class [101]. The few-shot learning paradigm deals with training models that can recognize novel classes using a few labeled examples [134]. In this problem domain, the meta-learning or learning-to-learn approach has achieved much attention and success [97]. The meta-learning methods can be roughly categorized into
two main classes: (i) optimization based and (ii) metric-learning based. In optimization-based approaches, the objective is to seek a meta-learner that can adjust the parameters of another learner to a novel task, given only a few labeled examples [123]. Metric-based methods [25] learn a task-agnostic embedding space to measure the similarity between labeled and unlabeled examples of a novel classification task. For example, Matching Networks [133] utilizes a weighted nearest neighbor classifier, while Prototypical Networks [120] uses the mean features of a few labeled or support instances as the prototype for each class. The matching network is susceptible to outliers due to one-on-one matching, but ProtoNet handles noisy handles via averaging across each class [133, 124].

4.3 Problem Definition

In this section, we outline the few-shot phase identification problem definition. We consider the practical use case scenario of seismic phase identification at stations with limited labeled examples. As shown in Figure 4.2, the accuracy of the existing phase identification methods is inferior for stations with limited labeled instances, particularly for stations covering new geographic locations, as shown in Table 4.1. Therefore, we focus on improving the phase identification performance for such data-scarce stations to be on par with data-rich stations and regions.

In the conventional few-shot learning settings, the model must adapt to novel or previously unseen classes with a few (for example, one or five) labeled examples. In seismic phase identification, the model must identify seismic phases at a novel station, with limited labeled examples for each class. Although the high-level or abstract class definition remains the same, changes in seismic source and station location modify the signal properties, leading to novel class representations. Therefore, in contrast to the existing phase identification methods, we propose considering them novel classes and adopting the few-shot learning framework for the novel stations.
Following the conventional few-shot learning notations, we refer to the data-rich stations as base stations and the data-scarce stations as novel stations. The few labeled instances of the novel stations are termed support set, and the unlabeled test samples are called query set. If there are \( C \) unique novel classes in a novel station, and each class has \( K \) labeled examples as the support set, then the target few-shot learning task is called \( C \)-way \( K \)-shot phase identification task. For example, if there are four phase types and five labeled examples for each phase type, it would be a 4-way 5-shot seismic phase identification task.

### 4.4 Method

The key challenge in few-shot seismic phase identification is learning informative representation for stations with limited labeled instances. Learning from a limited number of samples is challenging since the learned model can quickly become overfitted based on the biased distribution formed by only a few training examples. The meta-learning framework has shown considerable improvements in learning from limited labeled data. However, a few recent works in the few-shot image and video classification have shown that extensions of transfer learning methods, consisting of pre-training followed by adaptation, achieve comparable or better performance in few-shot learning compared to the meta-learning methods [25, 148, 32]. These pre-trained and finetuned methods rely on the abundant availability of the pretraining dataset, which is not present for seismic phase identification. Additionally, these methods would require fine-tuning during inference time for each novel station and for additions of new support examples, which is infeasible for practical deployment in the seismic monitoring pipeline.

We propose a metric-learning based few-shot phase identification framework that can learn discriminative signal representation for data-poor stations using limited labeled instances. We use the base stations during the meta-training stage to learn a feature extractor to generate feature repre-
sentations in a metric space that increases intra-class similarity and reduces inter-class similarity. Our framework is grounded on the prevailing episodic training paradigm [120, 133]. The key idea of episodic training is to mimic the actual inference situation to guide the model on learning from limited labeled examples. The consistency between the training and test environment alleviates the distribution gap and improves model generalization capability. We show a high-level overview of our few-shot phase episodic training framework in Figure 4.4. This framework is agnostic to the feature extractor, and any improvement in the feature backbone module would be complementary.

4.4.1 Episodic Training

In each training episode, we construct a N-way K-shot phase identification task for one randomly sampled base station. We create the support set $S_t$ by randomly sampling $K$ examples from each phase class. Similarly, we curate the query set $Q_t$ by randomly sampling $P$ samples from each class, excluding any samples part of the support set examples. The overall training process is based on a set of $T$ meta-training tasks. In each meta-training task, the model is trained to minimize the loss over the classification of the query set ($Q_t$) examples. This training scheme helps in learning naturally generalized knowledge over many meta-training tasks to be transferred to the novel stations.

4.4.2 Feature Extractor

Our proposed framework is agnostic to the backbone feature extractor. In this work, we use a four-layer Convolutional Neural Network (CNN) as a feature extractor to encode the high-dimensional input features into compact feature representations [106]. The architecture is similar to the CNN architecture proposed in the previous Chapter, for FASER. The output from the CNN is passed through two linear layers for dimensionality reduction. Several prior works have demonstrated the effectiveness of time-frequency
representation in seismic signal processing tasks [106, 93]. Therefore, we use Continuous Wavelet Transformation (CWT) to convert raw seismic signals into spectral-temporal representation before passing them into the feature extractor network. A detailed description of data preprocessing is presented in the following section. The input to the feature extractor is composite three-channel CWT images derived from the three-channel seismic waveforms. The output from the feature backbone \( f \) is \( m \)-dimensional feature space, where, \( f : \mathcal{X} \in \mathbb{R}^m \).

### 4.4.3 Prototype Computation

Once the feature embedding is generated from the feature extractor, the next step is to compute the class representative prototype for each class of the support set examples. This notion can be formally expressed as follows:

\[
P_k = \text{Prototype}\left( \{ z_i | A_i \in S_k \} \right)
\]

Here, Prototype is the prototype calculation function, and \( S_c \) denotes the set of labeled support set examples for the class \( k \). Following the idea of [120], we aggregate the feature embedding of all the support examples of a phase class using the following equation.

\[
c_k = \frac{1}{|S_k|} \sum_{(x_i, y_i) \in S_k} f_\phi(x_i)
\]

The next step is finding the nearest class prototype of the query example using a distance function. The final output is a distribution over novel phase classes for a given query input \( x \) based on a softmax over distances to each class prototype, as described in the equation below:

\[
p(y = k | x) = \frac{\exp \left( -d(f_\phi(x), c_k) \right)}{\sum_{k'} \exp \left( -d(f_\phi(x), c_{k'}') \right)}
\]

However, two critical challenges exist in generating prototype representations for stations with limited labels. First, the resulting prototypes often become
Figure 4.4: Outline of the few-shot seismic phase identification framework.

noisy and sensitive to outliers due to the fewer support sets. Secondly, limiting to only a single station’s support set hinders learning more diverse and discriminative feature representations. During the novel phase identification, the model should handle both catastrophic forgettings of old knowledge and overfitting the novel support set. Given a single station with a limited labeled example, extracting knowledge from the most similar station can effectively reduce overfitting. However, finding a one-to-one matching during inference is time-consuming and computationally costly.

Prior works have employed knowledge distillation to reduce overfitting and learn sufficiently expressive discriminative features during meta-training [112], where they introduce inductive bias learned from a large teacher model trained on data-rich classes. However, similar knowledge distillation is not feasible for phase identification due to the low amount of labeled data for data-rich classes. Therefore, we aim to introduce strong inductive bias in the feature extractor by capturing incremental global knowledge across all meta-training tasks. We enforce such inductive bias by adding a generic station-agnostic phase classifier module parallel to the few-shot classifier, termed the global feature module. The purpose of this station-agnostic global feature module is to learn global feature embeddings aside from the station-specific
feature representation. We term these as “basis prototypes”, which work as an anchor to reduce overfitting while introducing generalization across stations.

The global feature module consists of the same feature extractor architecture used in the few-shot module, followed by a classification layer. We use the cosine similarity function to calculate the classification score for each phase class. Although not a proper distance metric, the cosine angle between two features is a common feature similarity measure in the few-shot literature. Here, for an extracted feature $f(x)$ and the learnable weight matrix $W = [w_1, w_2, \ldots w_c]$, we compute the similarity scores for all $C$ class using the following formula:

$$s_j = \frac{f(x)^T w_j}{\|f(x)^T\| \|w_j\|} \tag{4.4}$$

Afterward, we pass the similarity scores through the softmax function similar to Equation 4.3. We scale the cosine similarity scores appropriately to match the non-saturating regimes of the softmax function. To maintain consistency across both task-specific and task-agnostic feature generation, we use the cosine function to compute the distance between the query set and class prototypes in the few-shot module. During each training step, we jointly optimize for the station-centric few-shot module’s prediction loss ($L_p$) and global feature module’s prediction loss ($L_g$), as follows.

$$L = L_p + \lambda L_g \tag{4.5}$$

Here, $\lambda$ is a tunable hyper-parameter. During inference for novel stations, we feed the support set and query set examples of the novel station into the feature extractor of the few-shot module to generate feature embeddings. Afterward, we calculate the class prototype using the support set feature embeddings and calculate the distance with the query set feature embeddings. We apply the softmax function on the distance values to produce normalized prediction probability scores for the few-shot module, as described in equation 4.3. For the global feature module, we feed only the query set examples
to generate feature embeddings. Then, we use the cosine classifier, followed by the softmax function, to generate prediction probability scores for each phase type. Finally, we combine the few-shot module and the global feature module’s prediction scores to identify the phase types for the query example set. We show the framework of FUSED in Figure 4.5.

4.5 Experiments and Results

In this section, we perform an experimental evaluation to measure the performance of FUSED using real seismic data compared to existing methods.
4.5.1 Dataset and Data Preprocessing

We use the same dataset used in the FASER, as described in the previous chapter. The dataset is curated from 10 years of continuous seismic data collected at 129 seismic stations spread across 89 countries around the globe [1]. This dataset includes 80TB of uncompressed seismic waveforms and the comprehensive IMS catalog, with arrival times and phase labels curated by human analysts from eight million seismic event detections. From these 8M seismic events, we filtered out 175K seismic phase labeled data, which were independently verified by at least two domain experts.

The input to the data preprocessing pipeline is three 60-second long time-series from the continuous waveforms of three broadband channels, BHZ (vertical), BHN (north-south), and BHE (east-west) of a single seismometer, sampled at 40Hz. Following conventional seismic signal preprocessing techniques, we detrend each waveform and remove the mean. We pass each waveform through a 0.4Hz to 10Hz second-order Butterworth bandpass filter in both directions to nullify the phase shift. We max-normalize the data across each channel to retain the relative amplitudes. Afterward, we apply a 64-scale Continuous Wavelet Transformation (CWT) on each waveform to obtain a three-channel spectral-temporal features representation.

4.5.2 Compared Methods

We compare FUSED with existing seismic phase identification methods to validate the performance improvement. Additionally, we configure the existing meta-learning methods for seismic phase identification tasks. For a consistent comparison, we use the same backbone feature extractor as FUSED for the off-the-shelf meta-learning methods.

- **Baseline-FASER** [29]: This is the conventional supervised classification approach where a classifier is trained using all the samples from the base stations. We use the current state-of-the-art phase identification method as the baseline method [29]. Following the traditional
phase identification setup, we configure a station-agnostic four-class phase identification problem. During training, we use all the samples from the base stations and only the support sets from the novel stations. During inference, we fine-tune the model using the support set examples from the novel station before testing the query set examples.

- **Baseline++ [25]:** Here, we replace the linear classifier layer of the baseline-FASER model with a cosine distance calculation layer.

- **MatchingNet [133]:** MatchingNet computes the cosine distance between a query feature and each support feature and calculates the average cosine distance for each class to find the closest match.

- **ProtoNet [120]:** ProtoNet computes the Euclidean distance between query features and the class mean of support features.

- **RelationNet [124]:** RelationNet replaces the predefined distance function of ProtoNet with a learnable relation module.

### 4.5.3 Experimental Settings

We use the eight station clusters shown in Figure 4.3 to perform an eight-fold cross-validation. In each iteration of the train-test fold, we use one cluster’s stations as novel test stations and the rest of the cluster’s stations as base training stations. During each iteration of the meta-training stage, we train a maximum of 50,000 episodes for 1-shot and a maximum of 40,000 episodes for 5-shot phase identification tasks. We use the validation set for early stopping and to select the trained model with the best accuracy.

In each training episode, we randomly sample one base station and then randomly sample \( k \) labeled instances as support set and 5 labeled instances for query set from that station for each four phase types to formulate a 4-way \( k \)-shot training task. In the training stage of the Baseline and Baseline++ method, we use all the labeled samples from the base stations for training and validation purposes with a 90 : 10 split between the train and validation.
Table 4.2: Performance metric comparison of FUSED against baseline methods. FUSED outperforms existing baseline methods for both one-shot and five-shot seismic phase identification tasks. The best algorithm for each task is colored in cyan.

<table>
<thead>
<tr>
<th>Method</th>
<th>1-shot</th>
<th>5-shot</th>
</tr>
</thead>
<tbody>
<tr>
<td>MatchingNet</td>
<td>68.7</td>
<td>73.3</td>
</tr>
<tr>
<td>Baseline-FASER</td>
<td>75.2</td>
<td>75.6</td>
</tr>
<tr>
<td>Baseline++</td>
<td>75.0</td>
<td>75.2</td>
</tr>
<tr>
<td>RelationNet</td>
<td>70.3</td>
<td>73.4</td>
</tr>
<tr>
<td>ProtoNet</td>
<td>71.2</td>
<td>74.2</td>
</tr>
<tr>
<td><strong>FUSED</strong></td>
<td><strong>79.3</strong></td>
<td><strong>82.7</strong></td>
</tr>
</tbody>
</table>

set, correspondingly. We use a batch size 256 and train for a maximum of 400 epochs with early stopping on the validation set. We train all models using the Adam optimizer [70] with an initial learning rate of 0.01 and with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. We apply $L2$ regularization with $\lambda = 0.001$, and we use categorical cross-entropy [48] as a loss function.

In the meta-testing stage of meta-learning methods, we test one novel station’s data at a time and randomly sample $K$ labeled instances from each phase as support examples and randomly sample 5 instances as query examples. For each novel station, we perform such a meta-test multiple times to account for bias due to random sampling. For the Baseline and Baseline++ methods, we use all the support set examples from the novel station to fine-tune the four-class classifier for 100 iterations with a batch size of four. Afterward, we test the query examples of novel stations. To maintain consistency, we use the same support and query set examples across all methods during each test iteration.

**Evaluation Metrics.** In our experiments, following conventional practices for classification tasks, we use accuracy as the performance metric.
4.5.4 Results

In Table 4.2, we report the performance of our method in comparison with the existing methods for both 1-shot and 5-shot seismic phase identification tasks, where the highest performance for each task is cyan-colored. We observe that FUSED consistently outperforms all the compared methods for both tasks.

4.6 Conclusion

In this chapter, we formulate seismic phase identification as a few-shot learning task to improve performance for novel stations and stations with limited labeled data. We propose a metric-learning based episodic learning framework, FUSED to enforce learning from limited training samples. We exploit both station-specific and station-agnostic features, where the station-specific feature supports adaptation to regional seismic features, whereas global, station-agnostic feature learning tackles overfitting and catastrophic forgetting. We show empirical evaluations using real-world data where our method outperforms existing methods. Our method reduces the dependency on the time-consuming and costly manual labeling process. FUSED does not require re-training for novel stations, and newly labeled instances can be readily integrated into the support set during inference to improve accuracy, making it suitable for dynamic real-world deployment.
Chapter 5

User Behavior Modeling in Social Media

5.1 Introduction

Online social platforms such as Facebook, Twitter, and Snapchat have become ubiquitous and integral in our daily lives to communicate and interact with others, find entertainment, and learn about current events. Content creators, providers, and consumers benefit when the right content is served to the right person at the right time. Additionally, better content recommendation enhances user engagement and interest in the platforms. Therefore, these social platforms proactively customize and personalize content, interactive features, and ads based on a user’s behavior and preferences to improve user satisfaction and engagement [13]. A prerequisite for successful personalization is predicting user behavior ahead of time. However, human behavior is complex and is confounded by several on-platform and off-platform factors [11]. Those factors include time of the day (e.g. shorter chat sessions during lunchtime vs. longer chat sessions in the evening) [43], day of the week (binge content-watching on weekends), momentary emotional state (exploring contents when bored) [66], social presence (sharing photos when meeting with friends), and other factors [51, 73, 26].
Recent works on modeling user behavior have utilized user activities in the recent past to make a prediction for an upcoming session [56, 63, 14, 147, 110]. Hidasi et al. [56] utilized Recurrent Neural Networks (RNNs) to model the dynamics on sequential behaviors. Jing and Smola incorporate timing of the sessions and time-interval between consecutive sessions in a recurrent model. A few recent works have shown better performance by using Long Short-Term Memory (LSTM) networks with time-gates [147] and exploiting contextual information, such as location and device [14, 126]. These methods exploit the recency or ephemerality in user behavior to dynamically model the temporal variation in online behavior, where ephemerality can be defined as occasional or potentially fleeting behavior.

So, behaviors do vary based on time, and predicting behavior from recent behaviors has proven valuable. There is, however, another insight we can incorporate: As humans, we imbibe into habits of what we do, how we do, and when we do; arguably, these habits make humans predictable.

Habits and routines are — among other drivers — driven by seasons, circadian rhythms [95], and school, work, and workout schedules [72]. Given that our offline and online behaviors are often intertwined, our online activities exhibit daily, weekly, and monthly cycles [109]. Previous works have demonstrated such regularities on multiple platforms [49, 143]. Golder et al. [43] demonstrated consistent weekly and seasonal patterns of social interaction among college students on Facebook. Grinberg et al. [49] identified daily and weekly patterns of food consumption and nightlife activity using Foursquare check-in data. A few methods have utilized cyclicity by incorporating session-timing as contextual information to capture the temporal dependency of user behavior [14, 63]. However, using time as a context with a fixed effect for all users limits the predictive models from learning user-specific temporal patterns. While this is probably less an issue for circadian rhythms and seasonality effects, it may leave user-specific variance unexplained.

In this work, we argue that existing methods cannot fully exploit the
cyclicity in user behavior, and through effective utilization, better predictive performance can be achieved. Moreover, although cyclicity is well-understood in global engagement and platform metrics [143, 42], it is less understood on a per user-level. Therefore, to address these two limitations, we guide our work based on the following research questions:

- **RQ1**: Does individual user behavior on social platforms exhibit cyclical properties, and does cyclicity vary across users?
- **RQ2**: To what extent can user behavior prediction on a social platform be improved by exploiting per user-level cyclicity?

To answer the first research question, we aimed to uncover the temporal dynamics of individual user-level behavior by analyzing user activity from Snapchat, a popular multimedia-driven online social platform. We demonstrated empirically that user behavior is largely driven by regularities (cyclicity) and ephemeral actions. We observed similar temporal variations of user activity levels across user cohorts on a daily and weekly basis. While examining regularities at the individual user level, we also noticed varying levels of cyclicity across users. The variations of temporal activity levels across user cohorts and the difference in cyclicity across users signify the necessity of modeling cyclical temporal dynamics at an individual user level.

To answer the second research question, we aimed to model cyclic and ephemeral behaviors jointly by utilizing a novel end-to-end neural framework which builds on well-validated recurrent networks. We augmented the traditional, ephemeral LSTM module with an additional LSTM head that utilizes historical activity data over a longer period to capture individual user-level cyclicity. Specifically, we aggregated user activities in a particular time frame (i.e., hour of the day) to capture cyclicity in the corresponding prediction time frame. We further employed an attention mechanism to fuse information from both heads adaptively. We evaluated the performance of our model on two anonymized user activity datasets collected from
Snapchat. We defined four behavior prediction tasks that are generalizable to other platforms and compared the performance against existing baseline methods to demonstrate that the simple addition of cyclicity modeling can effectively lead to improved accuracy from baselines on all four prediction tasks.

Through ablation studies, we examined the impact of each module in our model. We also conducted post-hoc sanity analysis on improvements over users with higher cyclicity, attention on cyclic and ephemeral modules when various amounts of data are available, and the performance at different times of the day. To summarize, our contributions are:

- We show that user behavior on social platforms is driven by cyclicity and ephemerality, and the patterns and levels of cyclicity vary across users.

- We leverage per user-level cyclicity by adding a cyclic-LSTM module along with the existing ephemeral-LSTM architecture to jointly model cyclicity and ephemerality through attention-based adaptive fusion.

- We leverage regularities in user behavior to achieve personalization without using any personally identifiable information, which is a timely approach considering societal concerns about data breaches.

- We outperform existing methods by on average 7% (up to 10%) macro $f1$-score on four user behavior prediction tasks using two real-world datasets from Snapchat. We demonstrate that our model can effectively model cyclicity.

5.2 Related Works

User Behavior Modeling. Since the ever-increasing popularity of social platforms, many studies have sought to understand, characterize, and model
people’s usage of these platforms. Such efforts include user engagement prediction, user churn rate prediction, and user intention prediction, among others [126, 69, 141, 80, 132, 81]. However, the majority of these works focus on aggregated user activities in the context of a long period (or window) of time and not on a session level. Recent research revealed the potential of accounting for person-centered modeling on user behaviors [31, 109]. Related to our problem space [72] utilized first-minute user activities in a Facebook session to predict the activity duration for the rest of the session. Another closely related work [77] modeled the action logging of mobile health apps to predict the next action based on the history of actions. This work formulated a probabilistic temporal point process model that considers temporal variation, short-term dependency, and long-term periodic effect. However, they model time-varying action propensity on a global level rather than on an individual level.

**Regularities in Online Behavior.** Several previous works have explored regularities in online and offline human behavior and identified daily, weekly, monthly, and seasonal patterns [42, 30, 95]. Golder and Macy identified daily and seasonal patterns of individual mood based on Twitter posts [42]. Golder et al. found consistent weekly and seasonal patterns of social interaction among college students on Facebook [43]. Grinberg et al.[49] show daily and weekly patterns of eating, drinking, shopping, and nightlife in human behavior using Foursquare check-ins. Moreover, it has been shown that people tend to reply to emails faster in the mornings and on weekdays [71]. Pier-son et al. proposed a Cyclic Hidden Markov Model to detect and model cycles in human menstrual cycle symptoms and physical activity tracking data [100]. A few other works have studied regularities in the context of repeated actions, for example, repeated web search queries [127], web page revisitation patterns [3], music listening [67], and video binge watching [130]. Recently, Saha et al. causally examined the effectiveness of timing ads based on person-centered contextualized modeling of user behavior on online platforms [110]. Several recent works have proposed models that exploit the
Recurrent Models for Behavior Modeling. In the recent past, recurrent models have shown promising results in a multitude of user behavior modeling tasks, mostly in the context of recommendation systems [56, 57, 119] with applications in next basket (or item) recommendation [63], streaming content recommendation [14], check-in location prediction [21] etc. Jing and Smola utilized the session time, time interval between sessions and contextual information as features to improve performance. Zhu et al. introduced a time-gate for LSTM to model time intervals between sessions to improve the recommendation performance. Beutel et al. improved the contribution of contextual features by using a second-order neural network to directly modify the neural network’s hidden states. Although these recurrent models consider recent temporal dynamics and contextual information, they are limited in their capability to capture long-term cyclical effects. Moreover, the majority of these methods learn user embeddings for each user to incorporate user-centric features, which are both static and privacy intrusive. These models are also limited in functionality for new users in cold start situations.

5.3 Task Description

We consider a general social platform where each user $u$ represents a registered user. Each user can engage with the platform by using several in-platform features, such as chatting with a friend in Snapchat, posting a tweet on Twitter, watching video clips on Facebook, or reacting to photos on Instagram; we call these user activities $a$. A session $s$ consists of a continuous sequence of activities, and two sessions are separated by more than a specific time interval. Each session is represented by a feature vector $f$, which contains each activity’s aggregated amounts in that session. For example, the
number of photos shared and the number of videos watched appear in $f$.

We formulate a user activity prediction task where, for each user, we have a sequence of previous sessions consisting of session-activity features along with the timing of the session. Let us consider a set of users $U$, and each user $u \in U$ has a sequence of historical sessions $H_u = H_{u_1}, H_{u_2}, \ldots H_{u_N}$, where $H_{u_i} = \{(f, t)\}$. $f$ represents the activity feature vector, and $t$ represents the timing of that session. Here, $t < t_I$, $t_I = \text{timing of the session to be predicted}$. Our user behavior prediction task can be formalized as follows: [User Behavior Prediction] Given a set of users $U$ and sequences of historical session $H$; for an upcoming session $s$ at time $t$ of user $u$, predict the amount $m$ of user activity $a$.

5.4 Temporal Dynamics and Cyclicity in User Behavior

In this section, we investigate two specific aspects of user behavior (1) routine or cyclic behavior, (2) transient or ephemeral behavior.

5.4.1 Dataset Description

To examine dynamics and patterns of user behavior in social platforms, we conduct our study on an anonymized user activity dataset from the smartphone app Snapchat, a popular social, multimedia, instant messaging platform used by more than 230M users worldwide [122]. First, we randomly sample 20K users who were active at least once in each month over the span of seven weeks from January 6, 2020, to February 23, 2020. Subsequently, we collect longitudinal user activity data for these users in the same period. The longitudinal nature of data allows us to study each user’s on-platform activity spread across several “sessions” of participation on Snapchat. Our study defines a session to start when a user opens the app, and a session to end when they close the app or they leave the app inactive for 15 seconds. We
Figure 5.1: (top) Diurnal cyclicity in global user activity. The y-axis represents the fraction of activity that happens in each weekday-hour over a single week, aggregated across all users. (bottom) Discover view activity across user clusters by hour (values are z-transformed), darker shades indicate higher activity amount. Here, users are clustered based on the aggregated value of their Snapchat activities (such as frequency and amount of communication and content consumption).
Figure 5.2: (a) Cyclicity CDF; most users are cyclic, more than 80% of users having a cyclicity score of more than 0.25. (b)-(f) show hourly average activity (session duration) in the first three weeks (history) and fourth week (current) for five example users, each selected from one of the correlation value buckets ([-1, 0], [0, 0.25], [0.25, 0.50], [0.50, 0.75], [0.75, 1]), as designated with dashed lines in (a). (b) Correlation bucket [-1, 0]. (c) Correlation bucket [0, 0.25]. (d) Correlation bucket [0.25, 0.50]. (e) Correlation bucket [0.50, 0.75] (f) Correlation bucket [0.75, 1]. Here, corresponding with the increasing cyclicity score from (b)-(f), the similarities between historical and current activity increases, too.
Figure 5.3: We show the correlation of user’s activity level between consecutive sessions. We categorize each session into three categories based on the level of activity. We show the distribution of current session’s activity level in relation to the previous session’s activity level for two cases: (a) SESSION TIME, (b) DISCOVER VIEW. In both cases, consecutive sessions are more likely to belong to the same category.

use the first four weeks of data for making empirical observations to motivate our modeling approach. Later (in Section 5.6), we use these four weeks of data to train our predictive models and hold out the subsequent three weeks to evaluate our models.

5.4.2 Cyclicity

Human behavior is largely affected and controlled by circadian rhythms, sleep habits, work, and leisure schedules [72], which is also reflected in online behaviors [95]. Consequently, people’s activities on online social platforms often show diurnal and weekly cycles [43]. To explore regularities at an aggregated level, in Figure 5.1 (top), we show the global usage pattern for three major Snapchat in-app activities (DISCOVER VIEW, STORY POST, SNAP SEND) for all users in our dataset. We can observe obvious daily patterns and notable hourly variations in all three activity types. The observed hourly variation in activity level is highly aligned with the human circadian rhythm.
and work-leisure schedule. Both STORY POST and SNAP SEND activities get higher traction as the day progresses and work-to-leisure transition happens. Additionally, DISCOVER VIEW shows irregularity on weekends with spikes in the mornings that correspond to leisure hours.

Next, we explore temporal patterns at the user cohort level. First, we use $k$-means clustering ($k = 10$) to cluster users based on the aggregated value of their Snapchat activities (such as frequency and amount of communication and content consumption). We use the Elbow heuristic [113] to select the optimal cluster numbers. To visually examine the temporal variability of activities across clusters, in Figure 5.1(bottom), we plot a heatmap of mean aggregated DISCOVER VIEW activity ($z$ normalized) for each cluster over the hours, where the vertical axis represents each cluster of users. Here, we observe that DISCOVER VIEW activity varies strongly both across and within clusters over the hours. We observed similar distinct usage patterns across user cohorts for several other app activities.

Intrigued by the global cyclicity and temporal variations across cohorts, we further investigate cyclicity on a per-user level. To quantify individual users’ cyclical usage patterns, we define a metric, cyclicity, that captures regularities in user behavior. We operationalize the metric per user cyclicity by assigning each user a score, defined as the Pearson correlation of users’ average hourly activity level for the first three weeks and their average hourly activity level in the fourth week.

In Figure 5.2a, we plot the cumulative distribution functions (CDFs) of cyclicity calculated based on SESSION TIME and DISCOVER VIEW. We notice that user behavior is more cyclic based on SESSION TIME than DISCOVER VIEW, which is intuitive as the former encompasses all forms of in-app activities contrary to one specific use case with the latter. From here on, cyclicity refers to cyclicity calculated with session-time, unless otherwise specified. Overall, most users display a certain extent of cyclicity, where 80% of users have a cyclicity score of more than 0.25. Notably, 25% of users have a cyclicity score of more than 0.75. As motivating examples of varying levels of
cyclicity across users, in Figure 5.2b-5.2f, we show hourly average session time in the first three weeks (history) and fourth week (current) for five randomly sampled users from each bucket.

5.4.3 Ephemerality

Several previous works have demonstrated the continuity of human action and intent in online behavior after a short interval [143, 128]. For example, Zhang et al. [143] showed that a session’s length can be an indicator of successive session length on the popular music streaming platform Spotify. To infer this causal relation, they correlated the length of two successive sessions. Similarly, to explore ephemerality on social platforms, we correlate consecutive session length in Snapchat. First, we divide all sessions into three categories (low, mid, and high) based on session duration. We select the session duration thresholds to roughly have a similar number of sessions in each category. Afterward, we calculate the distribution of each category in relation to the previous session’s category. In Figure 5.3a, in the percentage distribution plot, we notice that for all three categories, their likelihood is highest when the previous session is similar.

To further concretize the notion of ephemerality, we perform similar exploration on another core Snapchat activity, DISCOVER VIEW. As before, we quantify the DISCOVER VIEW activity sessions into three classes with equal class proportions. In Figure 5.3b, we show the percentage distribution of categorized DISCOVER VIEW depending on the previous session’s category. Again, we observe that the successive session’s DISCOVER VIEW activity level is similarly correlated as it is for session length. Although these simplified explorations do not consider the more complex dynamics of ephemerality, e.g., the interplay of multiple consecutive sessions or the effect of time-interval, as showcased in prior related studies [143], they can well justify the presence of recency or ephemerality on a social platform in an interpretable way.

**Summary findings.** (1) User behavior in Snapchat shows strong daily patterns and continuity of activities in consecutive sessions. (2) We observe
distinct temporal patterns across user cohorts. (3) We notice that users are in general cyclic; however, the cyclicity level varies across users. These findings motivate us towards our ensuing analyses where we target to better predict user behavior in social platforms by leveraging per user-level cyclicity.

5.5 Jointly Leveraging Cyclicality & Ephemerality

In this section, we formulate a user behavior prediction framework on social platforms to answer the second research question. To improve user behavior prediction by leveraging per user-level cyclicity, we seek to model user behavior in a session as a function of the user’s recent activity and the user’s historical activity around a particular time period (i.e., hour of the day, day of the week). In a modular approach, we model the user’s short-term ephemeral behavior and long-term cyclic behavior separately by two independent modules, termed as ephemeral module and cyclic module respectively, and fuse both for final prediction in the end. Moreover, traditional user behavior models incorporate personalization by utilizing user-typographic and demographic information, which can be privacy-intrusive, biased, and exclusionary. By exploiting the regularities in individual user behavior, we achieve personalization in a privacy-preserving fashion, as it only requires an individual user’s cyclic activity history to learn user-specific temporal preferences. Therefore, our method does not require any user-centric information, identifier, or demographic data, which is a key strength of our method. We discuss this in detail in the Discussion Section.

Ephemeral Module. Prior works on modeling sequential user behavior data have shown the superiority of the Recurrent Neural Networks (RNNs) over latent variable models (i.e., hidden Markov model) in capturing the short-term temporal dynamics of user behavior [56, 63]. Inspired by these successful use cases, we utilize an RNN to model the ephemerality in user
behavior. In particular, we use the Long Short-Term Memory (LSTM) network, an improved variant of traditional RNNs that addresses the vanishing gradient problem by employing a cyclic feedback mechanism from previous time steps. Due to the sequential nature of user behavior data, LSTM can effectively capture their temporal evolution and dependencies. Each LSTM unit is composed of a memory cell, a hidden, and three gating mechanisms: input, output, and forget gate. The input gate $i_t$, forget gate $f_t$, output gate $o_t$, memory cell $c_t$ and hidden state $h_t$ at step $t$ are computed as follows:

$$
i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i),$$
$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f),$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o),$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_c \cdot [h_{t-1}, x_t] + b_c),$$
$$h_t = o_t \odot \tanh(c_t) \quad (5.1)$$

Here, $\sigma$ is the logistic sigmoid function, $\tanh$ is the hyperbolic tangent function, $\odot$ denotes the element wise multiplication, and $t$ is the time step for each individual session.

We implement a two-layer LSTM where the input is a sequence of user activity feature vectors of the sessions preceding the current session of interest. We term these preceding sessions utilized to model the ephemerality in user behavior as ephemeral sessions. In our implementation, we limit the number of ephemeral sessions to a maximum of five within the last four hours. We have varied both the number of ephemeral sessions and ephemeral time-window but observed no significant performance improvement in our joint modeling approach. In the case of less than five ephemeral sessions within four hours, we perform zero paddings. We show the ephemeral module in the bottom-right part of Figure 5.4, where the ephemeral LSTM iterates through the input sequence for five-time steps (each time step corresponds to one session). The final hidden state output from the ephemeral module can be considered as a latent representation of a user’s recent activity summary.
Cyclic Module. Understanding the effect of time has been critical for effective user behavior modeling [63, 14]. However, utilizing temporal dynamics in a user-agnostic manner cannot exploit the individual user-level cyclicity. To accommodate personalized temporal preference, the traditional approaches (e.g., probabilistic models) would require distinct parameters for each user, which is not practically feasible to learn or maintain [77]. In this regard, we argue that a user’s time-specific historical activity can be leveraged to capture personalized temporal preference. To complement the recurrent network used for the ephemeral module and to facilitate end-to-end training, we propose to utilize a recurrent network for this purpose. The intuition behind this simplistic approach is to capture a user’s past activity history at a specific time of the day into a latent representation by iterating through these activity sequences. As the recurrent networks employ a cyclic feedback mechanism to update the current hidden states based on both current input and past hidden states, it is inherently suitable to aggregate overall cyclic history in the final embedding. For instance, for a user who typically engages in a long session at 5:00 PM during his commute, this approach makes it possible to use this knowledge for prediction.

Similar to the ephemeral module, we implement a two-layer LSTM for the cyclic module. However, contrary to the ephemeral module, here we use activities in a particular time frame (i.e., “hour of the day”, “day of the week”) in the past few days or weeks to learn the user’s activity preference in that corresponding time frame. Previous works have modeled temporal dynamics as a function of “hour of the day” to capture daily patterns [77], and as a function of “hour of the day & day of the week (hour-weekday)” to capture weekly patterns [63]. In our empirical observation, we have observed both daily and weekly patterns. Subsequently, we have experimented with both scenarios to find the superiority of modeling daily patterns. We average user activities in each hour in the prior seven days to create an input sequence of length seven for the cyclic LSTM. We term each of these hourly averaged feature vectors as cyclic sessions. We show the cyclic module in
the bottom-left part of Figure 5.4, where the cyclic LSTM iterates through the input sequence for seven-time steps (each time step corresponds to one hourly averaged activity feature). The final hidden state output from the cyclic module can be considered a latent representation of a user’s cyclic activity summary in a particular time frame.

**Figure 5.4: Proposed model architecture.**

**Context Factors.** Inspired by the previous successful use cases of contextual factors (i.e., location, device, software client, or web browser for YouTube video recommendation [14, 119]), we utilize four contextual factors related to Snapchat sessions, which are: (1) *device connectivity*, whether the user is
using WiFi connection or not; (2) *travel mode* if the user is using the app in travel mode or not; (3) *app open state*, whether a user is opening the app by clicking on an app notification or not; (4) *app status*, whether the app was running in the background or not. We merge this context information in the one-hot encoded format with the output hidden embeddings from the ephemeral and cyclic module before feeding into a feed-forward neural network. Although these contextual factors are specific to Snapchat, similar contextual factors are prevalent across other social platforms. Moreover, other contextual factors inherent to a particular platform can be readily utilized in our architecture.

**Modality Attention.** The most intuitive and simplistic approach to aggregate multi-modal information is the naive concatenation of each modality [89]. However, this naive concatenation of ephemeral and cyclic embedding treats both with equal importance in all instances. Nevertheless, in practicality, one modality can be more informative than the other. For example, while predicting a session, the user may not have any prior Snapchat sessions within the last four hours or any historical activities in the last seven days in that hour. Therefore, we employ a generalized modality attention module to attenuate or prioritize each modality for each prediction instance adaptively. This attention mechanism also enables us to quantitatively gauge each modality’s impact on predictive modeling across numerous dimensions, i.e., user cohort, time, etc. (see experiments in section 6.5). We feed in the ephemeral (*e*), cyclic (*c*), and context (*o*) vectors into the attention module as input to generate a soft-attended attention vector for each modality $v \in \{e, c, o\}$ calculated as Equation 5.2:

$$\alpha_v = \frac{\exp(\phi(v))}{\sum_{\nu \in \{e, c, o\}} \exp(\phi(\nu))},$$  

(5.2)

where, $v$ is the embedding vector of modality $v$, and $\phi(\cdot)$ is a mapping function implemented as a feed-forward neural network. Finally, we pass the fused embedding vectors through two feed-forward neural networks before
applying softmax to obtain a final prediction. In the center of Figure 5.4, we show the modality attention fusing ephemeral, cyclic, and contextual information. In coherence with the attention based joint modeling of cyclicity and ephemeralit, we name our model as CEAM: Cyclic Ephemeral Attention Model.

5.6 Experiments

In this section, we evaluate the predictive performance of CEAM using two user activity datasets from Snapchat. We aim to answer the following experimental questions:

• **EQ1**: Can CEAM outperform existing methods in predicting user behavior?

• **EQ2**: How does the ephemeral and cyclic module in CEAM affect performance?

• **EQ3**: Can CEAM effectively model the cyclicity and ephemerality in user behavior?

5.6.1 Datasets and Experimental Setup

We perform experimental evaluations using two datasets collected from Snapchat. As mentioned previously (in Section 4.1), one dataset contains anonymized user activity data for a set of 20K randomly sampled monthly active users who were active at least once in each month from January 6, 2020, to February 23, 2020 (MAU dataset).

We also extract a second dataset for 20K randomly sampled users who were active at least once each day within the aforementioned period (DAU dataset). In both cases, we collect 37 relevant user activity features for each session augmenting the previous studies on Snapchat [126, 141, 80]. We perform min-max normalization [98] of each feature independently before
training and testing. Both datasets span over seven weeks. We use the first four weeks for training purposes and the subsequent three weeks for testing.

**Prediction Tasks.** We define four specific user behavior prediction tasks for the upcoming session, which are the following:

- **Task 1:** Amount of time the user will spend in the session.
- **Task 2:** Number of viewed discover stories.
- **Task 3:** User engagement with subscription content.
- **Task 4:** User engagement with recommended content.

Following Kooti et al.’s [72] work on similar behavior prediction in Facebook, we frame these prediction tasks as classification problems by categorizing each behavior into multiple classes. For the first two tasks, we categorize the activity propensity into three classes: low, medium, and high, proportional to activity level (session length and discover view count). In both cases, the thresholds were selected to maintain roughly equal class balance. However, for the latter two tasks, we employ a binary classification scenario and predict whether the user engaged with the particular content category or not. We note that these user activities are also common on other social platforms, such as Instagram and TikTok, which display individual stories similar to Snapchat and recommended and subscription-based content. Hence, these prediction objectives can be easily transferred onto other social platforms.

**Evaluation Metrics.** We use *macro f1 score* as the main performance evaluation metrics. We ran each training and testing experiment ten times and reported the average.

### 5.6.2 Compared Methods

We compare the performance of our method against the following state-of-the-art methods to validate the accuracy of our user behavior prediction.
- **Copy Model (CM) [9]**: Predicting the current user behavior the same as users’ last session’s. This can be deemed as the most naive version of ephemerality-based prediction.

- **LSTM [63]**: LSTM has shown promising results in several sequential user behavior modeling tasks. We adopt the methods proposed in [63]. Similar to them, we generate embedding vectors for timing and interval of ephemeral sessions after passing through embedding layers and concatenating with the session activity features. We implement a two-layer LSTM that iterates over the ephemeral session’s feature vectors and feeds the output hidden embedding into a two-layer fully connected network to generate the prediction.

- **TLSTM [147]**: In [147], TLSTM has been introduced for user behavior modeling where the time interval between two consecutive actions has been used to moderate a gating mechanism to update the hidden states of LSTM for improved performance. We implement the architecture proposed in [147], and feed in the ephemeral sessions along with session intervals of consecutive sessions.

**CEAM** has access to contextual information to enhance predictions. Although not utilized in the proposed LSTM [63] and TLSTM [147], for a fair comparison, in our implementation, we also include the contextual information in the fully connected layers similar to **CEAM**. Previously, latent variable models (e.g., Markov models, hidden Markov models) and Poisson process models have been used for similar user behavior modeling. However, several recent works employing LSTM-based models have consistently and significantly outperformed the aforementioned approaches [63, 77]. Hence, we omit the comparison against these methods.

### 5.6.3 Model Implementations

We implement a two-layer LSTM network for the ephemeral and cyclic modules, with an embedding size of 32 in both layers. We set the second layer’s
Table 5.1: Prediction performance (macro f1-score) of CEAM on all tasks and both datasets MAU and DAU.

<table>
<thead>
<tr>
<th>Model</th>
<th>MAU</th>
<th>DAU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Task 1</td>
<td>Task 2</td>
</tr>
<tr>
<td>CP</td>
<td>0.375 ± 0.000</td>
<td>0.296 ± 0.000</td>
</tr>
<tr>
<td>LSTM</td>
<td>0.480 ± 0.002</td>
<td>0.482 ± 0.001</td>
</tr>
<tr>
<td>TLSTM</td>
<td>0.470 ± 0.005</td>
<td>0.472 ± 0.003</td>
</tr>
<tr>
<td>CEAM</td>
<td>0.500 ± 0.001</td>
<td>0.521 ± 0.001</td>
</tr>
</tbody>
</table>

dropout rate to 0.5 and use ReLU as the activation function. We implement CEAM and other neural models using PyTorch\(^1\). All the models are optimized using Adam Algorithm [17], with an initial learning rate of 0.001, and an L2 regularization of 1e − 6. We set the batch size to 512. We train all models to a maximum of 50 epochs with early stopping on the validation set. All the hyper-parameters were selected empirically using a grid search on a held-out validation set.

5.6.4 Prediction Performance

To answer the first experimental question, we report the prediction performance (macro f1-score) of CEAM along with the compared methods for all four tasks on both datasets in Table 5.1. We observe that CEAM outperforms all the other methods in all four tasks in both datasets. We note that both LSTM and tLSTM perform on a similar level. However, CEAM outperforms both these models by at most 10%. Although both LSTM and tLSTM

\(^1\text{www.pytorch.org/}\)
use session timing as contextual information, they significantly underperform CEAM, which validates our argument to model cyclicity at the individual user level.

### 5.6.5 Ablation Study

To answer the second experimental question, we perform several ablation studies by developing three variations of the proposed model. (1) **EPHEMERAL**: In the first variation, we use only the ephemeral module and feed the output embedding concatenated with the contextual embedding into a two-layer fully connected network for prediction generation. In contrast with baseline LSTM, we do not utilize session-timing or session interval as features in the ephemeral module. (2) **CYCLIC**: Next, we use only the cyclic module in a similar fashion (3) **COMBINED**: Then, we employ both cyclic and ephemeral modules and concatenate their output embedding along with contextual embedding before feeding into the fully connected layers. In Table 5.2, we report the prediction performance for all variations. Here, we observe that

![Table 5.2: Ablation studies to show the contribution to performance improvement (macro f1-score) by different modules.](image)

<table>
<thead>
<tr>
<th></th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
<th>Task 4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MAU</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ephemeral</td>
<td>.479±.002</td>
<td>.479±.002</td>
<td>.683±.000</td>
<td>.701±.000</td>
</tr>
<tr>
<td>Cyclic</td>
<td>.459±.003</td>
<td>.480±.001</td>
<td>.673±.000</td>
<td>.700±.000</td>
</tr>
<tr>
<td>Combined</td>
<td>.500±.002</td>
<td>.519±.001</td>
<td>.721±.000</td>
<td>.741±.000</td>
</tr>
<tr>
<td><strong>CEAM</strong></td>
<td>.501±.001</td>
<td>.521±.001</td>
<td>.723±.000</td>
<td>.743±.000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
<th>Task 4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DAU</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ephemeral</td>
<td>.478±.002</td>
<td>.474±.002</td>
<td>.680±.001</td>
<td>.694±.000</td>
</tr>
<tr>
<td>Cyclic</td>
<td>.460±.002</td>
<td>.480±.001</td>
<td>.675±.000</td>
<td>.700±.000</td>
</tr>
<tr>
<td>Combined</td>
<td>.499±.001</td>
<td>.517±.002</td>
<td>.720±.001</td>
<td>.742±.000</td>
</tr>
<tr>
<td><strong>CEAM</strong></td>
<td>.500±.002</td>
<td>.518±.002</td>
<td>.722±.000</td>
<td>.744±.000</td>
</tr>
</tbody>
</table>
both cyclic and ephemeral modules are in general predictive across all tasks in both datasets. Their predictive accuracies are in a similar range, with Ephemeral being higher in the majority of the cases. However, the combination of both shows consistent performance improvements than each employed individually.

This further increases with the use of self-attention. We see at most 10% performance improvement by CEAM over a single module approach. This improvement validates the presence of complementary information in both behavioral dynamics and the necessity of joint modeling.

5.6.6 Model Sanity Check

To answer the third experimental question, we design multiple experiments with the goal of understanding for whom, how, and when CEAM improves prediction performance. We perform these experiments on the MAU dataset. First, we investigate for whom, CEAM shows better performance. The underlying motivation behind modeling cyclicity on a per-user basis is to capture an individual user’s app usage regularities for improved predictability. Therefore, we would expect CEAM to be more effective for more cyclic users. To quantify the performance improvement in relation to a user’s cyclicity, we separate users into two groups based on a median split on cyclicity scores (defined in Section 5.4). One group consists of the top 50% cyclic users (more-cyclic), and the rest were in the other (less-cyclic).

We introduce the CYCLIC module in predictions and then calculate the change in accuracy for the user’s prediction in both groups. We transform the raw accuracy change values into $z$-score to reduce sensitivity to inconsistent magnitudes of absolute values [42], used in prior related works [108]. By definition, $z$-score represents the distance between raw value and population mean in units of standard deviation [42]. In Figure 5.5a, we report the mean $z$-score of accuracy change for both user groups for all four tasks. Here, zero $z$-score refers to the mean accuracy im-
Figure 5.5: (a) Accuracy change (z-transformed, 0 shows mean improvement) for more-cyclic and less-cyclic user groups for all four tasks after adding the cyclic module. We gain greater improvement for more-cyclic users. (b) The variation in ephemeral and cyclic attention scores as the ephemeral session number varies. The cyclic attention is greater in fewer ephemeral sessions, and the ephemeral attention increases along with the increase in ephemeral session number.

Improvement for all the users (population) over the ephemeral module-only performance, which is $\approx 10\%$. Above zero z-score indicates greater improvement than the overall population mean improvement and vice versa. We observe that in all four tasks, more-cyclic users show greater accuracy improvement than the population mean whereas less-cyclic users show less accuracy improvement, which resonates with our initial intuition about the model.

Next, we investigate how CEAM better exploits both cyclic and ephemeral aspects of human behavior. In CEAM, we employ a self-attention mechanism to fuse information from both these modalities. We ask whether the attention module can adaptively prioritize one over the other depending on the information of each modality. To quantify dynamic preference, we calculate the shift in attention in relation to the number of available sessions in each module. As example, in Figure 5.5b,

we show the variation of ephemeral and cyclic attention weight as the
Figure 5.6: The hourly averaged accuracy for all four tasks (a) Task 1 and Task 2, (b) Task 3 and Task 4. (c) The average number of ephemeral and cyclic sessions in each hour. (d) Hourly average ephemeral and cyclic attention weight, and accuracy change after incorporating cyclic module (z-transformed).
number of ephemeral sessions vary. Following the intuition, the cyclic module gets greater attention in case of fewer ephemeral sessions. And as the ephemeral session number increases, so does the attention weight for the ephemeral module.

Lastly, we explore the temporal aspect of prediction performance. In Figure 5.6a and 5.6b, we show the prediction accuracy over an hour of the day for Task-1&2 and Task-3&4 respectively. We observe that, for all four tasks, the accuracy remains similar from around 8 AM till midnight. However, there are sharp drops between midnight to around 6 AM in the morning. To better understand it, we plot the average number of ephemeral and cyclic sessions across the day in Figure 5.6c. We notice that both ephemeral and cyclic session average drops during late night to early morning compared to the rest of the day, which correlates with the lower accuracy period. Therefore, we can reasonably attribute the lower accuracy period to the shortage of cyclic and ephemeral activities around that time.

We further explore how CEAM exploits information from both modalities to improve prediction across hours. In Figure 5.6d, we plot the hourly average attention weight for the ephemeral and cyclic modules along with the hourly average accuracy change after adding the cyclic module. Again, we utilize the z-transformation for normalized comparison across several measures. We observe a drop in ephemeral weight and an increment in both cyclic weight and accuracy change between 4 AM to 8 AM. The positive increase in cyclicity indicates that CEAM relies more on cyclic information around that time period when there are fewer ephemeral sessions, as shown in Figure 5.6c. And, due to this adaptive and better utilization of cyclicity, we improve prediction.
5.7 Discussion

5.7.1 Model Robustness

User agnostic. CEAM does not require any user-specific information and does not learn any user-specific parameters. Hence, our proposed model can be readily used for any new users in cold-start situations [114]. To drive this point home, we simulated a similar scenario, where we only evaluate users who do not have sessions in the training set. In Figure 5.7, we show the prediction F1-score for the user-agnostic and non-user-agnostic cases. We observed similar performance in both cases, which shows the usability of our method in cold-start situations.

5.7.2 Ethical and Privacy Implications

Several recent data breaches and reported misuse of sensitive and private user data have resulted in growing public concerns and increases in regulations [20]. Any experimental study dealing with potentially sensitive data
demands a statement regarding ethical conduct and secure data handling. In our study, user activity data from the social app Snapchat was used for empirical and experimental purposes. Following Snapchat’s in-house commitment towards user privacy and data protection policies, our dataset was anonymized before our analyses and was void of any form of personally identifiable demographic or typographic information. Moreover, the user activity data was only comprised of amounts and frequencies of activities: no information such as communication content, type of content, or source of viewed content was used in any part of this study. All the experiments were conducted within Snapchat’s internal secure storage systems, and data was not stored on local computers or outside the Snap Inc. ecosystem.

Given that our models well predicted user behavior without using personally identifiable information and user attributes, this modeling approach is applicable to several scenarios. First, because our behavior modeling does not use any identifiable user attributes, it may be preferable in highly sensitive settings. Second, because the model uses no platform-specific user attributes, it would be relatively straightforward to apply this technique to other platforms. This could be useful to technology designers (e.g. for modeling the behavior of users in other mobile apps) or to online communities that stand to benefit from user behavior modeling.

5.8 Conclusion

In this chapter, we present one way to improve user behavior modeling on online social platforms by including cyclical behavior in the prediction. Using Snapchat data, we demonstrated regularities in people’s behavior, both at the collective and the individual level. We then proposed an end-to-end neural framework that leverages both cyclic and ephemeral aspects of people’s daily lives for improved prediction. Importantly, our method is personalized, but agnostic to privacy-invasive data: We do not use any user-typographic or demographic information, and we avoid any sort of long-term data-based user
profiling. Short-term data and dynamically generated behavioral features are a more ethical approach for responsible user modeling. We evaluated the efficacy of our method using four prediction tasks on two datasets from Snapchat. Our method outperforms existing methods by on average 7% (up to 10%) (macro f1-score). Empirically, we show that our model can successfully capture the cyclicity in individual user behavior. While our work focuses on Snapchat, the demonstrated insights and proposed modeling approach can potentially motivate similar explorations and modeling in other social platforms to increase business value and deliver a better user experience.
Chapter 6

Conclusion and Future Work

We began this dissertation with the aim of leveraging domain context and intrinsic data properties to improve pattern recognition, classification, and prediction tasks involving temporal data. This dissertation shows four practical examples of utilizing domain knowledge to design effective algorithms to improve performance in three different temporal data domains. In the first work, we use signal decomposition techniques to improve the usability of data science methodologies on process monitoring tasks. The proposed semi-supervised method requires one labeled example for adaptation to a new data source. The single parameter has an intuitive interpretation and is easy to fine-tune without any knowledge of the underlying algorithm. The proposed method is suitable for practical deployment as it is computationally inexpensive and requires minimal manual intervention.

In the second work, we present a deep-learning method FASER for fine-grained seismic phase identification using single station data. FASER reduces the dependency on array-based methods requiring multiple nearby stations to enable precise monitoring for regions with limited monitoring stations. We use a combination of CNN and LSTM to exploit local structured patterns and long-term temporal dependencies in seismic signals. Due to the minimal preprocessing requirements and faster prediction generation, it is highly suitable for a real-time monitoring pipeline. In the third work, we develop FUSED, a
few-shot learning method for seismic phase identification that addresses the limitations of FASER to improve performance for novel stations with limited labeled data. We propose a metric-learning based episodic training framework to enforce learning from limited training samples. By leveraging both station-centric and station-agnostic features, we can capture region-specific seismic properties while effectively handling overfitting and catastrophic forgetting.

In the fourth and final work, we utilize regularities in human behavior to improve user activity prediction in online social platforms. First, we demonstrate regularities in people’s online behavior, both at the collective and the individual level, using real-world data from Snapchat. We develop an end-to-end deep sequential network CEAM that jointly models people’s recent and cyclic activity patterns for improved prediction. Due to the user-agnostic modeling approach, CEAM can used in cold-start situations for new users and for highly sensitive settings. The proposed approach can be utilized in other online platforms as it is not dependent on platform-specific user or activity attributes.

The research presented in this dissertation on three different domains can be further advanced in multiple directions. As each domain has its own context, properties, and challenges, we discuss the future work separately.

1. **Structured Noise Detection in Well Test Pressure Derivative Data.** Semi-supervised and human-in-the-loop learning is becoming increasingly popular over time. Our work shows a novel approach of using minimal human input to perform semi-supervised learning and reduce manual labeling overhead. A similar one-shot learning approach can be utilized in other related domains where annotation is expensive and requires in-depth domain expertise. In the future, self-supervised and zero-shot learning methods can

2. **Seismic Phase Identification for Automated Monitoring.** In FASER, we proposed a deep-learning model for seismic phase identification that can be readily integrated into the existing seismic monitoring
pipeline. In FUSED, we address the limitation of FASER by proposing a few-shot phase identification framework to improve performance for novel stations with limited labeled data. However, there are opportunities for improvement in multiple avenues. For example, Generative Adversarial Networks (GAN) can be utilized to address data imbalance by generating synthetic datasets from existing samples. Also, the self-supervised learning method can be explored to utilize large amounts of unlabeled seismic datasets.

3. User Behavior Modeling in Social Media. CEAM utilizes only the propensity of user actions and activities while ignoring their qualitative aspects. For example, we only consider the time spent viewing multimedia content, but we disregard the nature of the content being watched. In future works, the observed temporal activity variations can be further explored to learn each user’s preference towards certain content and ads across the day, which can help in identifying suitable time periods to distribute content and ads. Better personalized content allocation without compromising privacy can benefit the 230B social media industry and the platforms’ users alike. Moreover, our flexible modeling approach is suitable for integrating a multitude of contextual factors (e.g., seasonality, weather, location) when available, depending on the social platform, to improve the prediction performance. Additionally, further experiments can be conducted to examine how the time difference between consecutive sessions affects the ephemerality aspect of user behavior. Future work might explore an online training setting in which user behavioral modeling is integrated with a real-time prediction mechanism.
Bibliography


102


[19] W. D. P. by Upstream Research Company. Well-[a,b,c,d,e,f], 2018.


