Major Project Report on

# Classification of Fever Patterns using Deep Learning

Submitted in partial fulfilment of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
ELECTRONICS & COMMUNICATION ENGINEERING
by

**Anirudh Prabhakaran (201EC106)**

*under the guidance of*

**Dr. Sumam David**

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA
SURATHKAL, MANGALORE - 575025
April, 2024

## DECLARATION

I hereby *declare* that the Major Project Work Report entitled ***"Classification of Fever Patterns using Deep Learning"*** , which is being submitted to the **National Institute of Technology Karnataka, Surathkal**, for the award of the Degree of Bachelor of Technology in Electronics & Communication Engineering, is a *bonafide report of the work carried out by me.* The material contained in this Major Project Report has not been submitted to any University or Institution for the award of any degree.

*Name of the Student (Registration Number) with Signature*

(1) Anirudh Prabhakaran (201EC106)

Department of Electronics and Communication Engineering
Place : NITK, Surathkal
Date : 15/04/2024

## CERTIFICATE

This is to *certify* that the Major Project Work Report entitled ***"Classification of Fever Patterns using Deep Learning"*** submitted by

*Name of the Student (Registration Number)*

(1) Anirudh Prabhakaran (201EC106)

as the record of the work carried out by them, is *accepted as the B.Tech. Major Project work report submission* in partial fulfillment of the requirement for the award of degree of Bachelor of Technology in Electronics and Communication Engineering in the Department of Electronics and Communication Engineering, NITK Surathkal.

**Dr. Sumam David**
Professor (HAG)
Department of E & C Engineering
NITK Surathkal

**Dr. N.S.V Shet**
Professor and Head
Department of E & C Engineering
NITK Surathkal

# ACKNOWLEDGEMENT

The successful completion of this project would not have been possible without the invaluable contributions of several individuals, whom I would like to extend my sincere gratitude.

First and foremost, I express my deepest appreciation to Dr. Chakrapani M of Kasturba Medical College, Mangalore, whose meticulous and comprehensive dataset formed the cornerstone of this endeavor. I am also indebted to Prof. Deepu V, whose guidance and assistance were instrumental at every stage of this journey. His expertise and encouragement significantly contributed to the project's success. Special gratitude is reserved for my mentor, Prof. Sumam David, whose steadfast support and mentorship were invaluable throughout the project's development. Without her wisdom and guidance, navigating this endeavor would have been markedly more challenging. Finally, I extend my thanks to the faculty, classmates, and friends whose support and encouragement were invaluable. Whether through helping in debugging code or offering a receptive ear for brainstorming, their assistance was greatly appreciated.

I am deeply grateful to all those who contributed to this project, directly or indirectly. Their collective efforts have been indispensable in bringing this work to fruition. Thank you.

# ABSTRACT

Tympanic temperature is one of the most fundamental indicators for diagnosis of diseases. In a hospital setting, temperature it is an fundamental metric that tracks a patient's status. Due to its importance, it would be very useful to use temperature data of patients to aid in the diagnostic process. This project aims to leverage temperature data collected from various patients to classify different diseases. We first do a comparative study of various types of models to understand which model architectures independently perform well. After our experiments, we note that using a mixture of architectures produces the best results for our usecase. From the temperature data, extraction of important features is necessary so that the downstream layers only have to consider important features, and not miscellaneous information. This feature extraction is done using two methods - Convolution Neural Networks and Autoencoders. This difference has lead to different models with different architectures. We introduce these three new models for Explainable Temperature analysis - ExTemp-Conv-SM, ExTemp-Conv-LG, and ExTemp-Auto. Along with this, we use explainable AI tools to try to identify distinguishing patterns in temperature fluctuations that can characterize diseases. Using GradCAM, we note the distinct temperature patterns that are unique to particular diseases.

***Keywords***— Machine Learning, Medical Machine Learning, Deep Learning, Multiclass Classification, Temperature Data Analysis, Explainable AI

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# CHAPTER 1
# INTRODUCTION

## 1.1 Overview

In the ever-evolving landscape of healthcare, the integration of advanced technologies has become paramount for accurate diagnosis and timely intervention. Amidst these advancements, the utilization of temperature data has emerged as a promising avenue for disease classification. This technical report encapsulates the methodologies, findings, and implications of a pioneering project aimed at harnessing temperature data for the precise classification of diseases.

At the heart of this endeavor lies the recognition of temperature as a fundamental physiological parameter, reflective of the body's intricate response to various pathological conditions. Leveraging this wealth of information, our project endeavors to develop robust machine learning models capable of discerning subtle patterns within temperature data to differentiate between different diseases.

There are multiple indicators used by doctors for diagnosis. Most of the imaging techniques involve complicated and expensive scans like X-Rays, MRI, CAT etc. These methods allow for extremely detailed diagnostics and high performance machine learning models. In contrast to that, temperature data is very easily accessible and a decent indicator of disease progression and patient condition. However, there are drawbacks to using temperature data on its own. This data does not provide enough information to accurately predict diseases. Based on individual patient physiology, temperature patterns for diseases may be different, and not follow the same pattern. For the same disease, different patients may display dissimilar trends in their temperature over a period of time. This leads to an inability to create high accuracy models.

## 1.2 Motivation

Considering the complexity of temperature data and the variations within, in this thesis we try to tackle this problem from two interconnected directions.

One direction is creation of models that can perform with good accuracy given temperature data. Previous attempts have tried to use statistical methods to achieve the same. Their experiments have also been setup as binary classification problems (given a temperature pattern, is it indicative of this disease or not?). We take a new approach, creating models based on advances in machine learning and deep learning. Along with this, we analyze this problem as a multi-class classification problem (given a temperature pattern, which disease is it most likely to be indicative of?), giving us an opportunity to study more diseases, and perform a better analysis.

Another direction is to study the data itself. We recognize that temperature data cannot be the singular source of information for a particular diagnosis. However, there might be some defining characteristics, or patterns in temperature patterns of a particular disease. We try to identify these patterns and trends that have a strong correlation to the diagnosis of a disease. We leverage the latest advances in Explainable AI to help us with this - giving a glimpse into what features the model was giving priority to while making a decision.

# CHAPTER 2

# LITERATURE REVIEW

## 2.1 Background and Related Works

Most previous work in classification of diseases based on temperature data has focused on using statistical techniques. Cuesta-Frau et al. (2019) [1] used Sample Entropy to classify fever patterns. They use one single extracted entropy feature to perform classification. Cuesta-Frau et al. (2020) [2] use Slope Entropy (2019) [3]. This is a novel method of estimating time series complexity based on symbolic patterns and amplitude information. The goal of classification using machine learning techniques has been made. Dakappa et al. (2017) [4] tried to tackle this problem using quadratic support vector machines. There have also been some attempts to tackle this problem using artificial neural networks. However, they are quite limited in their functionality, and focus mainly on classification. Dakappa et al. (2018) [5] introduces artificial neural networks for classification of temperature patterns into two classes - infectious and non-infectious diseases.

There has been intense study on temperature patterns for various diseases as well. In particular, dengue and tuberculosis have been studied on the 24-hour duration. Dakappa et al. (2018) [6] observed the 24-hour continuous tympanic temperature pattern for 15 patients with dengue fever. The note that a tri-phasic fever pattern was seen among a majority of the patients. Similarly, Dakappa et al. (2019) [7] studied 24-hour continuous tympanic temperature in tuberculosis on 81 patients. The noted that majority of the patients exhibited a slow temperature elevation, followed by a slow temperature fall.

## 2.2 Outcome of Literature Review

Classification of diseases using temperature data has been studied before, but there are still places that require attention. Usage of Sample and Slope Entropy for this purpose also denotes that effectively considering smaller temporal variations is crucial for a proper classification attempt. Most previous works leveraging machine learning techniques are utilizing the entirety of data, and not giving specific importance to temporal information. Artificial intelligence methods have also focused on binary classification of diseases.

There has been very little previous work in finding temperature patterns using machine learning. There are biological observation studies that have meticulously calculated and identified trends in temperature variations. However, both publications note that although the majority follow these trends, the minority is not insignificant. Nearly 25% of the patients studied in both cases did not have temperatures following the mentioned trends.

## 2.3 Problem Statement

We note that there are several areas to work on. We aim to work on creating accurate models that can classify diseases using temperature data. We also aim to analyze the model and its predictions, to identify the unique characteristic of temperature trends for every disease.

## 2.4 Objectives of the Project

The problem statement can be broken into two parts. In the first part, we aim to create new models. We try different model architectures to identify which ones perform best. Once experiments are completed, we finalize the models that have the highest performance.

The second part focuses on understanding the model's decision process. With the advance in Explainable AI techniques, deep learning models are no longer a black box. We apply GradCAM on our models for different disease classes and find what patterns encourage the model to make a certain prediction.

# CHAPTER 3

# PROPOSED METHODOLOGY

## 3.1 Data

### 3.1.1 Dataset

The temperature data was provided by Prof. Chakrapani M, Department of Internal Medicine, Kasturba Medical College, Manipal Academy of Higher Education, Manipal, Mangaluru, India. The dataset contains records from 185 patients diagnosed with nine different diseases. Out of these, we are considering only four: Dengue, Non-Infectious Diseases, Non-Tubercular Bacterial Infection and Tuberculosis. This is done because of the availability of data. There are a considerable number of data points for these classes as compared to the others provided.

| **Disease** | **Counts** |
|---|---:|
| Dengue | 47 |
| Leptospirosis | 15 |
| Malaria | 16 |
| Malignancy | 7 |
| Non-Infectious Diseases | 28 |
| Non-Tubercular Bacterial Infection | 37 |
| Pyogenic Sepsis | 2 |
| Thyroiditis | 1 |
| Tuberculosis | 32 |

Table 3.1.1: Counts of classes provided in the data set.

### 3.1.2 Preprocessing

The preprocessing steps for the dataset is straightforward. We extract the data from the classes of interest to us. After creating this new data, we perform simple normalization of the data to bring all the values into a comparable range. We use `MinMaxScaler` to bring the values between 0 and 1. This data is then split into training and testing sets. We use a split of 80% of the data to use as training data, and the remaining 20% as the test set.

The data is provided as a 2D data set, with each record having 1440 entries. This

| **Type** | **Counts** |
|---|---|
| Train | 115 |
| Test | 29 |

Table 3.1.2: Number of samples for each data set.

corresponds to the number of minutes in 24 hours, as the temperature was recorded at every minute for this duration. Below we have a few example plots of the temperature data for the different diseases.

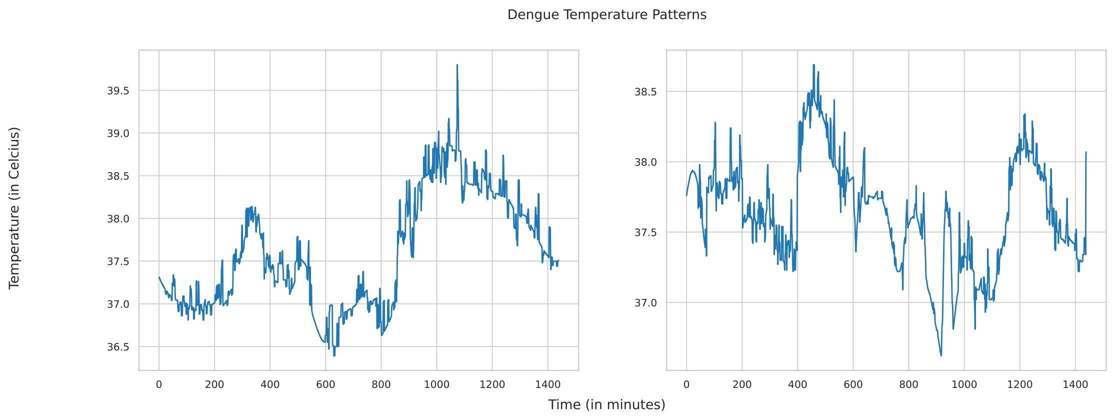

Figure 3.1.1: Sample Temperature Patterns for Dengue

From visualization of the data points provided, we especially note that there is quite a lot of fluctuation in the patterns of diseases like Dengue and Tuberculosis. As compared to that, temperature graphs for Non-Tubercular Bacterial Diseases and Non-Infectious Diseases are comparatively smoother, although they do have fluctuations.

## 3.2 Models

Before finalizing models to use, we performed experiments and compared the performances of all these models. We briefly summarize the characteristics of the variety of models that we used in table 3.2.1.

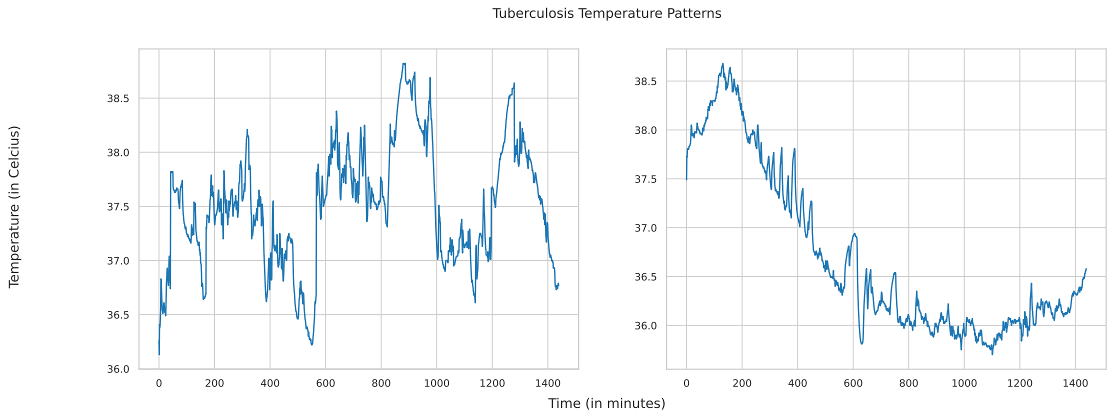

Figure 3.1.2: Sample Temperature Patterns for Tuberculosis

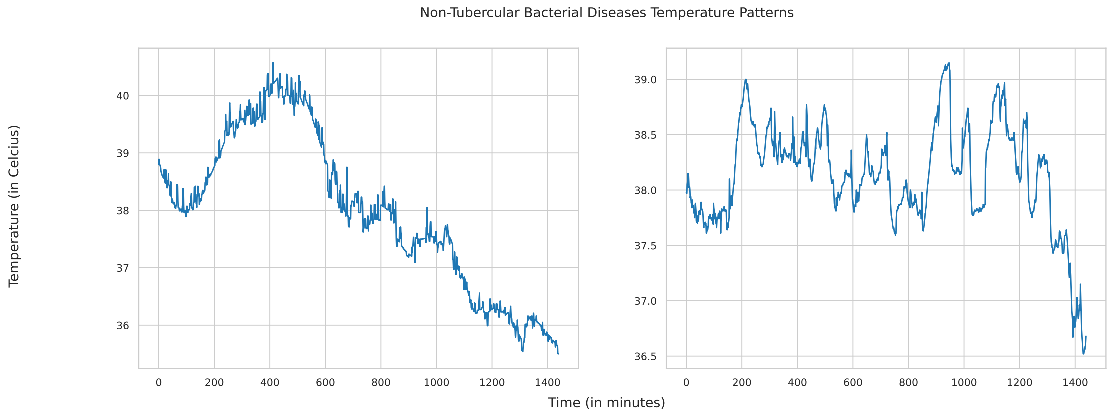

Figure 3.1.3: Sample Temperature Patterns for Non-Tubercular Bacterial Diseases

| **Model Name** | **# of params** | **Model Size (MB)** |
|---|---|---|
| Multi Layer Perceptron | 5,764 | 0.07 |
| Three Layer NN (with/without dropout) | 2,033,476 | 8.28 |
| 1D CNN | 123,012 | 8.39 |
| 1D Two Block CNN | 53,572 | 10.72 |
| 1D Three Block CNN | 63,428 | 14.22 |
| 1 Layer RNN | 4,548 | 5.96 |
| 2 Layer RNN | 12,868 | 6.00 |
| 3 Layer RNN | 21,188 | 6.03 |
| LSTM | 83,972 | 6.28 |

Table 3.2.1: Details of Models used for Experiments

We ran our experiments on each of these models on the decided data set. For these experiments, we focus purely on the classification task, and not on the explainability task.

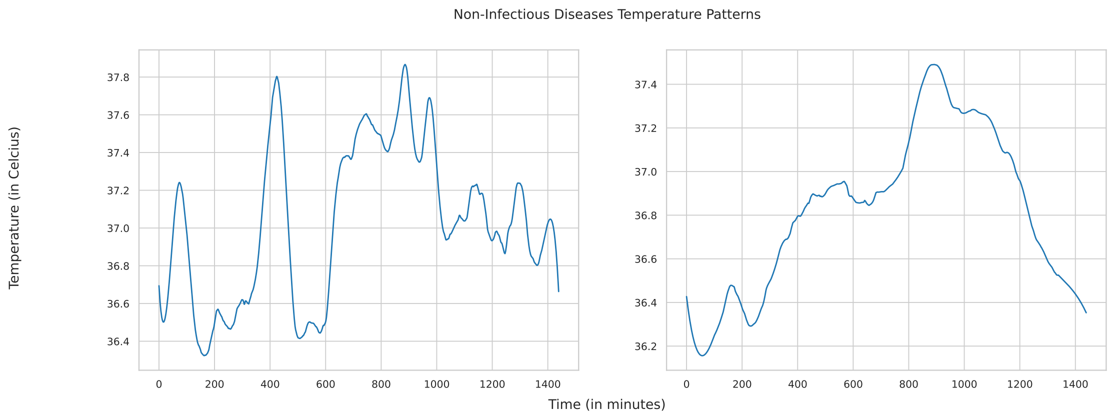

Figure 3.1.4: Sample Temperature Patterns for Non-Infectious Diseases

We use traditional classification metrics, like accuracy, precision, recall and F1 score, along with report loss. We present the findings of our experiments in table. We also experiment with a variety of optimizers, like Stochastic Gradient Descent (SGD) and Adam, to ensure that we can generate the best models.

Based on these experiments, we noted that convolution based models outperform the others. This matched our hypothesis that these model will perform better that the rest. This was because it is important to capture the small temporal differences in the temperature data, rather than trying to use the entire data in one go for classification. Convolution would be able to perform this feature extraction from temporal variations very well.

From this observation, we decided to use a 1D convolution based model. Previously, Singstad et al. [8] have proven that usage of 1D Convolution Neural Networks in medical applications do perform well. We provide two such models, named **ExTemp-Conv-SM** and **ExTemp-Conv-LG**. Both of these are based on convolution blocks, followed by a fully connected neural network to perform classification. The difference in the two models comes from slight differences in the architecture, leading to a difference in the number of parameters for each model. We also note that for these two models, data has to be slightly modified. Instead of using the standard 2D data provided, we need to reshape the data to be used by convolution. This is because the convolution block expects data to be in the form $(num\_channels, num\_features)$, instead of the current shape of data that we have, which is $(num\_features)$. We have reshaped the data so that the number of channels in the input data is 1, i.e. reshaping leads to data input of shape $(1, 1440)$.

There has also been previous work to denote that convolution blocks act as an effective method for dimensionality reduction [9]. We explore this idea more, and look at other promising methods for dimensionality reductions. Hinton et al. [10] show that autoencoders can be effectively used for feature extraction and dimensionality reduction. Using the same thought process, we replace the convolution blocks in our proposed network with the encoder block of an autoencoder. This is the motivation behind the third model - **ExTemp-Auto**.

| Parameter | **ExTemp-Conv-SM** | **ExTemp-Conv-LG** | **ExTemp-Auto** |
|---|---:|---:|---:|
| d_model | (1, 1440) | (1, 1440) | (1440, ) |
| Layers | 24 | 14 | 10 |
| Feature Layers | 15 | 5 | 2 |
| Classifier Layers | 8 | 8 | 8 |
| Feature Params | 1,464 | 40 | 17,556 |
| Classifier Params | 369,236 | 745,892 | 368,896 |
| Total Params | 370,700 | 745,932 | 386,452 |

Table 3.2.2: Details about ExTemp Family of Models

Below, we give more detailed diagrams of the model architectures. We then elaborate about the basics of the theory of convolution and autoencoders, and discuss the optimization criterion and loss functions used.

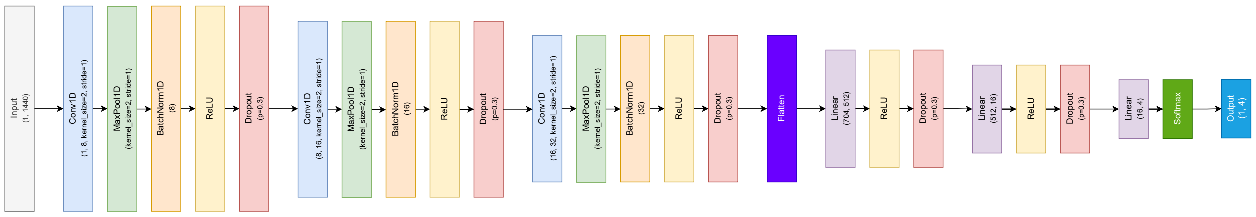

Figure 3.2.1: ExTemp-Conv-SM Model

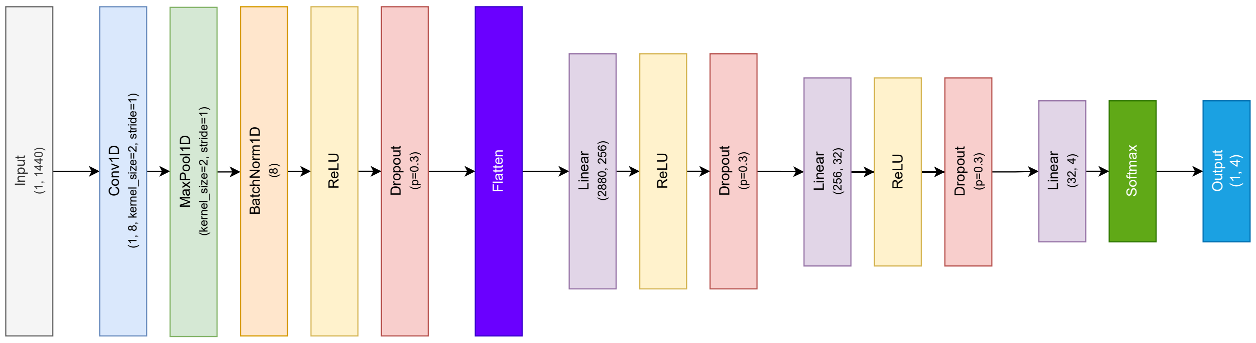

Figure 3.2.2: ExTemp-Conv-LG Model

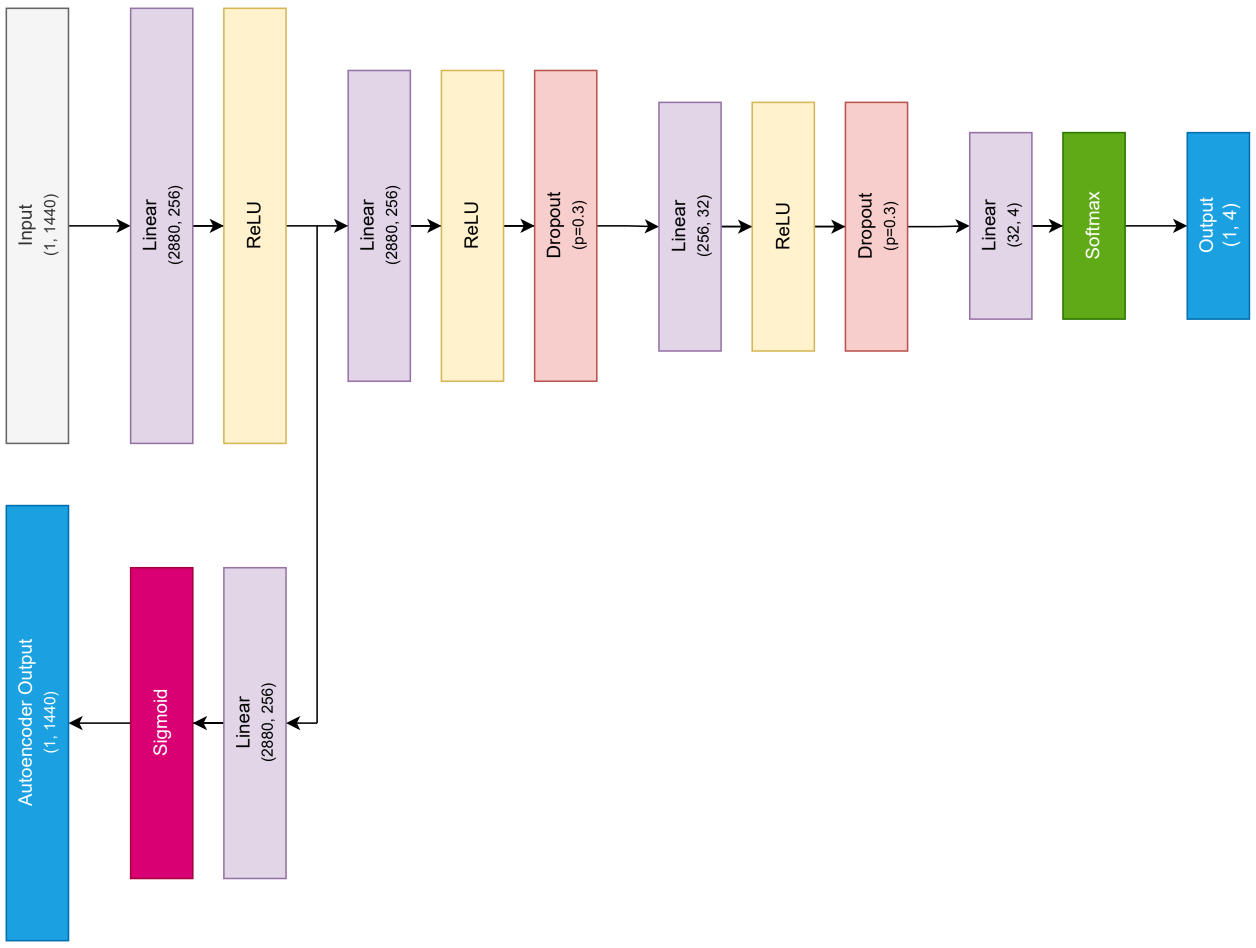

Figure 3.2.3: ExTemp-Auto Model

### 3.2.1 Theory

This section talks in brief about the different technologies used in project as a refresher.

#### Convolution

One-dimensional Convolutional Neural Networks (1D CNNs) [11] are a variant of the traditional CNN architecture, tailored for processing sequential data such as time series, signals, and text. Unlike their 2D counterparts designed for grid-like data such as images, 1D CNNs operate along a single dimension, typically the temporal or spatial axis of the input sequence. They employ convolutional layers that slide one-dimensional kernels along the input sequence, capturing local patterns and dependencies. These layers are often followed by activation functions and pooling operations, enabling hierarchical feature extraction and dimensionality reduction. 1D CNNs excel at learning meaningful representations from sequential data, automatically capturing patterns and temporal dependencies, making them particularly effective for tasks like speech recognition, sentiment analysis, and physiological signal processing. Their ability to learn hierarchical features and exploit local correlations in sequential data has led to their widespread adoption across various domains, highlighting their versatility and effectiveness in modeling and analyzing sequential information.

Convolution at a position $i$ can be expressed as

$$(x * w)_i = \sum_{j=0}^{m-1} x_{i+j} \cdot w_j \tag{3.1}$$

where:

- $x$ is the input sequence.
- $w$ is the filter.
- $m$ is the size of the filter.

#### Autoencoders

Autoencoders [10] are a type of neural network architecture employed in unsupervised learning tasks, primarily for dimensionality reduction, feature learning, and data generation. Consisting of an encoder and a decoder, autoencoders aim to reconstruct their input data as accurately as possible. The encoder compresses the input into a latent space representation, typically of lower dimensionality, while the decoder attempts to reconstruct the original input from this representation. Through this process, autoencoders learn to capture meaningful features and patterns in the data, effectively reducing its dimensionality while preserving important information. Variants such as denoising autoencoders add noise to the input to enhance robustness and prevent overfitting, while convolutional and recurrent autoencoders extend the concept to image and sequential data, respectively. Autoencoders find applications in various domains, including image denoising, anomaly detection, and generative modeling, where they serve as powerful tools for data representation and manipulation in both research and industrial settings.

The following steps are performed in an autoencoder:

- **Encoding:** The encoding process in an autoencoder can be represented mathematically by the mapping from the input $x$ to the latent representation $z$ using the encoder function $E$:

$$z = E(x) \tag{3.2}$$

- **Decoding:** The decoding process maps the latent representation $z$ back to the reconstructed output $x'$ using the decoder function $D$:

$$x' = D(z) \tag{3.3}$$

- **Loss:** We use an appropriate loss function for evaluation of the autoencoder.

$$L = loss(x, x') \tag{3.4}$$

## 3.3 Loss Calculation

The convolution based models are trained using the standard cross entropy loss [12] that is used for other classification tasks. For a multi-class classification task with $K$ classes, we define cross entropy loss as the following:

$$\text{Cross-Entropy Loss} = \sum_{i=1}^{K} y_i log(p_i) \tag{3.5}$$

where:

- $y_i$ is the true probability distribution over the classes (with $y_i = 0$ for all classes except the true class where $y_i = 1$)
- $p_i$ is the predicted probability of the $i$-th class, output from the model's softmax activation function.

For the autoencoder-based model, we use a slightly different approach. Since there are essentially two models in play here, we use two corresponding loss computations and optimization criterion functions. We use the cross-entropy loss for the classification part of the model. However, for the autoencoder model, we use Mean Squared error. We define Mean Squared Error as the following:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.6}$$

where:

- $n$ is the number of samples in the dataset.
- $y_i$ is the true target value for the $i$-th sample.
- $\hat{y}_i$ is the predicted value for the $i$-th sample.

We then use a summation of both these losses, and apply our back-propagation techniques on this computed **Total Loss**. We define the Total Loss as the following:

$$\text{Total Loss} = \text{Cross-Entropy Loss} + \text{MSE} \tag{3.7}$$

## 3.4 Explainability

Finally, we focus on the second part of the project. Explainability in AI plays a crucial role within the realm of medical applications, where the interpretability of AI-driven diagnoses and treatment recommendations is paramount. Transparent AI systems offer healthcare professionals insight into the reasoning behind AI-generated predictions, enabling them to understand and trust the decision-making process. Interpretable AI models are particularly valuable in medical settings, as they provide clinicians with comprehensible explanations of how patient data is analyzed and interpreted, aiding in the validation of diagnoses and treatment plans. Moreover, accountable AI frameworks ensure that the outcomes of AI-assisted medical decisions can be traced back to specific algorithms and data sources, facilitating regulatory compliance and ethical practice. In the healthcare domain, achieving explainability in AI not only enhances trust between patients and practitioners but also promotes the responsible integration of AI technologies into clinical workflows, ultimately improving patient outcomes and safety.

We select one of the most popular Explainable AI technique, **GradCAM** [13], to analyze the decision making process of these models.

### 3.4.1 Gradient-weighted Class Activation Mapping

Gradient-weighted Class Activation Mapping, also known as GradCAM [13], is one of the most popular Explainable AI techniques. It is a technique widely utilized in computer vision and deep learning to interpret the decisions made by convolutional neural networks (CNNs). By computing gradients of the target class score with respect to the feature maps of the last convolutional layer, GradCAM identifies the crucial regions within an input image that contribute most significantly to the model's prediction for a particular class. GradCAM enhances the interpretability and transparency of CNNs, aiding in the validation and understanding of their predictions. It finds extensive application in fields such as medical imaging, where identifying the relevant features contributing to diagnostic decisions is crucial for enhancing clinical trust and confidence in AI-assisted diagnostics.

The GradCAM (Gradient-weighted Class Activation Mapping) method involves several key steps to generate class activation maps highlighting the important regions of an input image. Here are the steps along with their corresponding mathematical equations:

We begin with the **Forward Pass**. The input data $I$ is passed through the model to obtain the final feature maps of the last convolutional layer, denoted as $A^k$.

After this, we **compute Class Score**. The class score $y^c$ for the target class $c$ is computer using the final feature maps $A^k$ and the classification layer weights $W^c$:

$$y^c = \sum_i \sum_j W_i^c A_{i,j}^k \tag{3.8}$$

We then **compute the gradient of class score**. The gradient of the class score $y^c$ with respect to the final feature maps $A^k$ is computer using back-propagation.

$$\frac{\partial y^c}{\partial A_{i,j}^k} \tag{3.9}$$

Then, we find **compute Importance Weights**. The importance weights $\alpha_{i,j}^c$ are computed by global average pooling of the gradients over each feature map.

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{i,j}^k} \tag{3.10}$$

Finally, we **generate Class Activation Map**. The class activation map $L_{GradCAM}^c$ is computed as a weighted combination of the final feature maps $A^k$ using the importance weights $\alpha_{i,j}^c$.

$$L_{GradCAM}^c = \mathrm{ReLU}(\sum_k \alpha_k^c A^k) \tag{3.11}$$

These steps together enable GradCAM to produce class activation maps highlighting the regions of the input data that are most important for predicting the target class $c$, providing valuable insights into the decision-making process of the model.

We leverage the `IntegratedGradients` function provided by Captum [14]. This module employs a few more optimizations to increase the speed of computation of Class Activation Maps. It also helps in easily adapting the GradCAM code to a 1-dimensional use case, similar to what we have used in this project.

# CHAPTER 4

# RESULTS AND ANALYSIS

## 4.1 Model Experimentation

We selected multiple models to run experiments on. Details about the model can be found in table 3.2.1. All of the experiments were run using the same hyperparameters, so that we could accurately compare the different models used. A table containing the values of the hyperparameters used is mentioned in table 4.1.1. Training results can be found in table 4.1.2. Testing results can be found in table 4.1.3. All metrics have been computed using `sklearn` [15].

| Parameter | Value |
|---|---:|
| Number of Epochs | 100 |
| Learning Rate | 0.001 |
| Momentum | 0.9 |
| Dropout Rate | 0.3 |

Table 4.1.1: Experimental Hyperparameter Values

| Model Name | Optimizer | Loss | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|---|---|
| Multi Layer Perceptron | SGD | 1.411 | 0.344 | 0.363 | 0.344 | 0.337 |
| | Adam | 1.401 | 0.310 | 0.324 | 0.310 | 0.311 |
| Three Layer NN without Dropout | SGD | 1.363 | 0.379 | 0.445 | 0.379 | 0.324 |
| | Adam | 1.398 | 0.344 | 0.748 | 0.344 | 0.409 |
| Three Layer NN with Dropout | SGD | **1.288** | 0.413 | 0.551 | 0.413 | 0.459 |
| | Adam | 1.329 | 0.413 | **1.000** | 0.413 | 0.585 |
| 1D CNN | SGD | 1.369 | 0.344 | 0.340 | 0.344 | 0.337 |
| | Adam | 1.369 | 0.379 | 0.393 | 0.379 | 0.369 |
| 1D Two Block CNN | SGD | 1.453 | 0.241 | 0.476 | 0.241 | 0.260 |
| | Adam | 1.426 | 0.275 | 0.384 | 0.275 | 0.286 |
| 1D Three Block CNN | SGD | 1.457 | 0.275 | 0.521 | 0.275 | 0.342 |
| | Adam | 1.416 | 0.310 | 0.439 | 0.310 | 0.360 |
| 1 Layer RNN | SGD | 1.338 | **0.448** | 0.635 | **0.448** | **0.524** |
| | Adam | 1.373 | 0.310 | 0.366 | 0.310 | 0.334 |
| 2 Layer RNN | SGD | 1.352 | 0.413 | 0.770 | 0.413 | 0.534 |
| | Adam | 1.372 | 0.275 | 0.351 | 0.275 | 0.306 |
| 3 Layer RNN | SGD | 1.356 | 0.344 | 0.896 | 0.344 | 0.498 |
| | Adam | 1.413 | 0.344 | 0.896 | 0.344 | 0.498 |
| LSTM | SGD | 1.367 | 0.344 | **1.000** | 0.344 | 0.512 |
| | Adam | 1.460 | 0.206 | 0.455 | 0.206 | 0.283 |

Table 4.1.3: Experimental Models Testing Metrics

| Model Name | Optimizer | Loss | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|---|---|
| Multi Layer Perceptron | SGD | 1.072 | 0.662 | 0.691 | 0.662 | 0.657 |
| | Adam | 1.019 | 0.701 | 0.721 | 0.701 | 0.702 |
| Three Layer NN without Dropout | SGD | **0.847** | 0.718 | 0.744 | 0.718 | 0.685 |
| | Adam | 1.443 | 0.330 | 0.365 | 0.330 | 0.256 |
| Three Layer NN with Dropout | SGD | 0.910 | 0.587 | 0.505 | 0.587 | 0.500 |
| | Adam | 1.423 | 0.361 | 0.269 | 0.361 | 0.243 |
| 1D CNN | SGD | 1.033 | 0.679 | 0.697 | 0.679 | 0.683 |
| | Adam | 1.108 | 0.585 | 0.582 | 0.585 | 0.575 |
| 1D Two Block CNN | SGD | 0.909 | 0.780 | 0.784 | 0.780 | 0.780 |
| | Adam | 0.863 | 0.799 | 0.806 | 0.799 | 0.800 |
| 1D Three Block CNN | SGD | 0.873 | **0.812** | **0.814** | **0.812** | **0.811** |
| | Adam | 1.001 | 0.746 | 0.755 | 0.746 | 0.747 |
| 1 Layer RNN | SGD | 1.378 | 0.284 | 0.145 | 0.284 | 0.164 |
| | Adam | 1.355 | 0.306 | 0.220 | 0.306 | 0.231 |
| 2 Layer RNN | SGD | 1.366 | 0.311 | 0.137 | 0.311 | 0.166 |
| | Adam | 1.331 | 0.342 | 0.284 | 0.34 | 0.292 |
| 3 Layer RNN | SGD | 1.370 | 0.325 | 0.149 | 0.325 | 0.166 |
| | Adam | 1.347 | 0.356 | 0.305 | 0.356 | 0.279 |
| LSTM | SGD | 1.368 | 0.321 | 0.103 | 0.321 | 0.15 |
| | Adam | 1.382 | 0.350 | 0.216 | 0.350 | 0.242 |

Table 4.1.2: Experimental Models Training Metrics

From these tables, we were able to gauge that the best performance is gained when we use convolution models, that can extract temporal features from the given input data.

## 4.2 ExTemp Models

Based on the observations made before, we proposed three models for Explainable Temperature analysis - **ExTemp-Conv-SM**, **ExTemp-Conv-LG** and **ExTemp-Auto**. Hyperparameters for these models can be found in table. These hyperparameters were discovered as ideal with the help of the `RayTune` [16] project. The models were trained using Adam [17] optimizer. Loss functions used are as noted previously - cross-entropy loss for the convolution based models, and total loss consisting of cross-entropy and mean squared loss for autoencoder based model.

| Parameter | **Value** |
|---|---:|
| Number of Epochs | 1000 |
| Learning Rate | 0.006243539059180805 |
| Dropout Rate | 0.4 |

Table 4.2.1: Ideal Hyperparameter values for ExTemp Models

Let us study the individual results of these models.

### 4.2.1 ExTemp-Conv-SM

The following results correspond to the training and testing of the ExTemp-Conv-SM model. The model is called the ”small” model because of the lower number of parameters in the model. However, note that this model contains a higher number of convolution blocks for feature extraction.

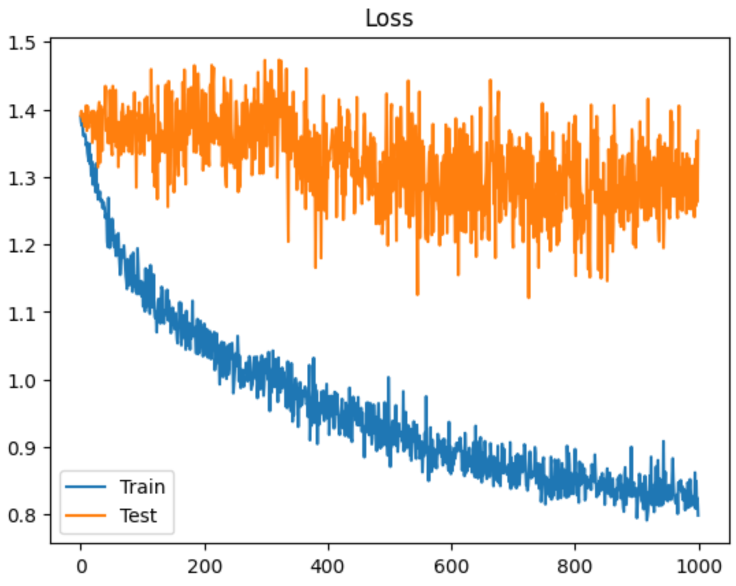

Figure 4.2.1: Loss Curves for ExTemp-Conv-SM

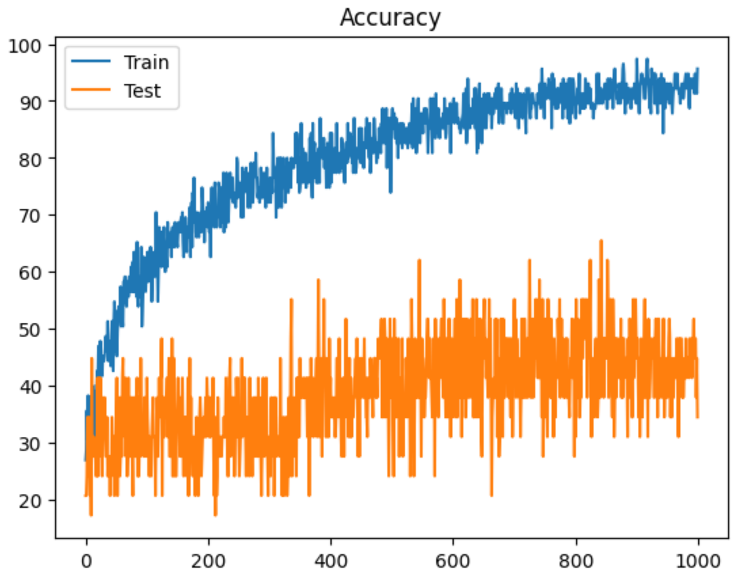

Figure 4.2.2: Accuracy Curves for ExTemp-Conv-SM

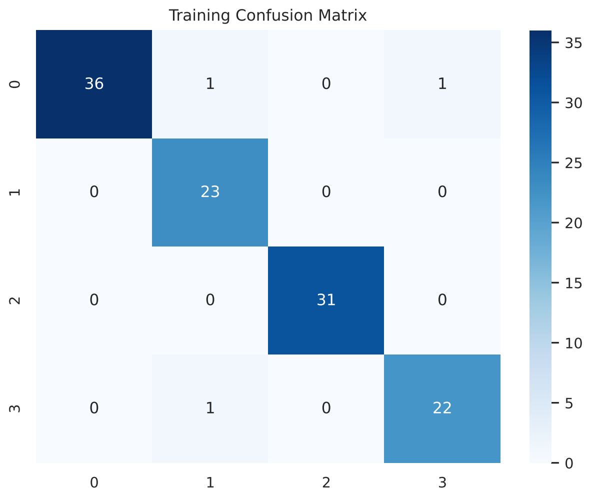

Figure 4.2.3: Training Confusion Matrix for ExTemp-Conv-SM

| | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| 0 | 1.00 | 0.95 | 0.97 | 38 |
| 1 | 0.92 | 1.00 | 0.96 | 23 |
| 2 | 1.00 | 1.00 | 1.00 | 31 |
| 3 | 0.96 | 0.96 | 0.96 | 23 |
| | | | | |
| accuracy | | | 0.97 | 115 |
| macro avg | 0.97 | 0.97 | 0.97 | 115 |
| weighted avg | 0.98 | 0.97 | 0.97 | 115 |

Table 4.2.2: Training Classification Report for ExTemp-Conv-SM

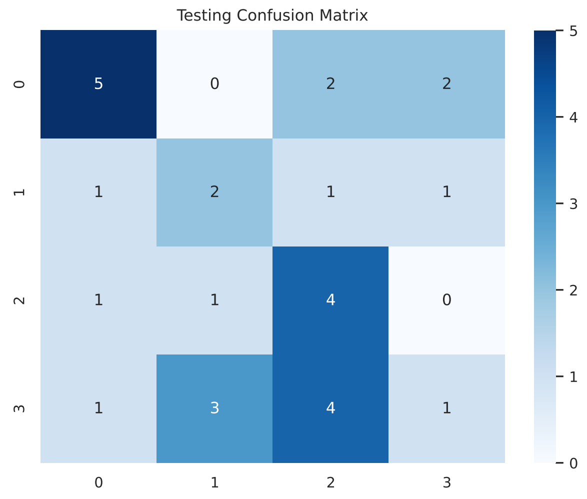

Figure 4.2.4: Testing Confusion Matrix for ExTemp-Conv-SM

| | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| 0 | 0.62 | 0.56 | 0.59 | 9 |
| 1 | 0.43 | 0.40 | 0.36 | 5 |
| 2 | 0.62 | 0.56 | 0.59 | 6 |
| 3 | 0.62 | 0.67 | 0.47 | 9 |
| | | | | |
| accuracy | | | 0.65 | 29 |
| macro avg | 0.62 | 0.56 | 0.52 | 29 |
| weighted avg | 0.62 | 0.55 | 0.52 | 29 |

Table 4.2.3: Testing Classification Report for ExTemp-Conv-SM

### 4.2.2 ExTemp-Conv-LG

The following results correspond to the training and testing of the ExTemp-Conv-LG model. The model is called the ”large” model because of the higher number of parameters in the model. However, note that this model contains a lower number of convolution blocks for feature extraction.

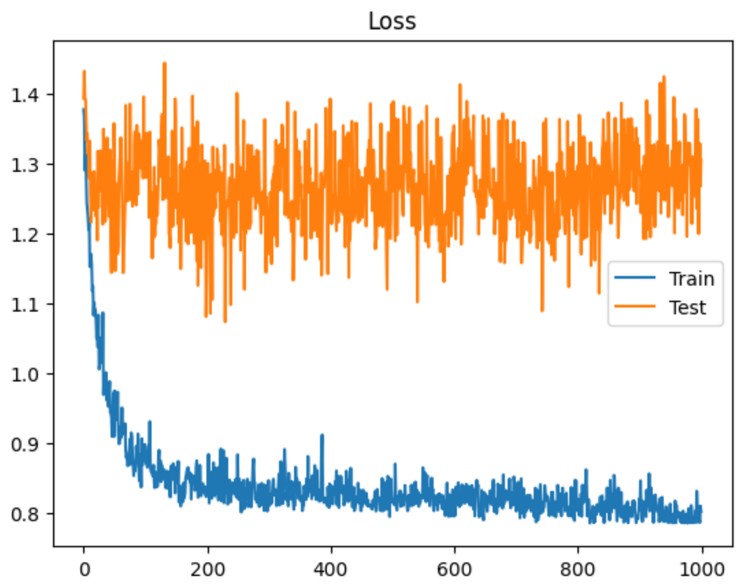

Figure 4.2.5: Loss Curves for ExTemp-Conv-LG

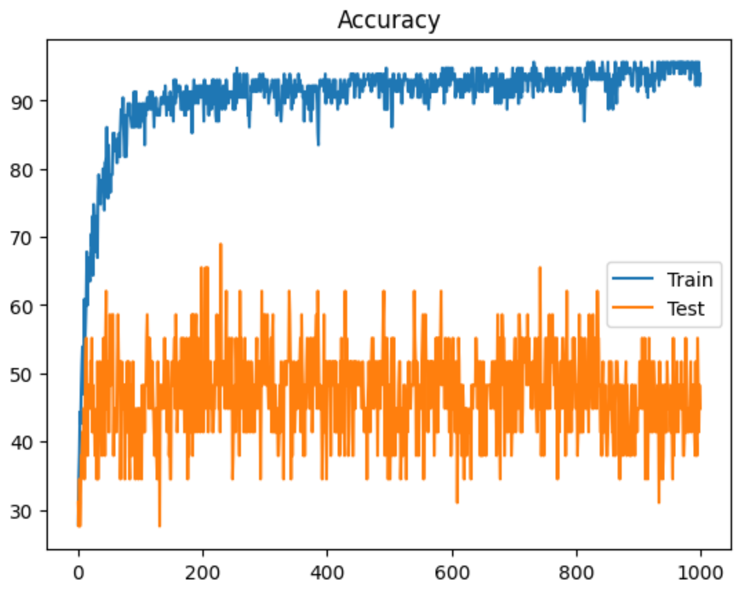

Figure 4.2.6: Accuracy Curves for ExTemp-Conv-LG

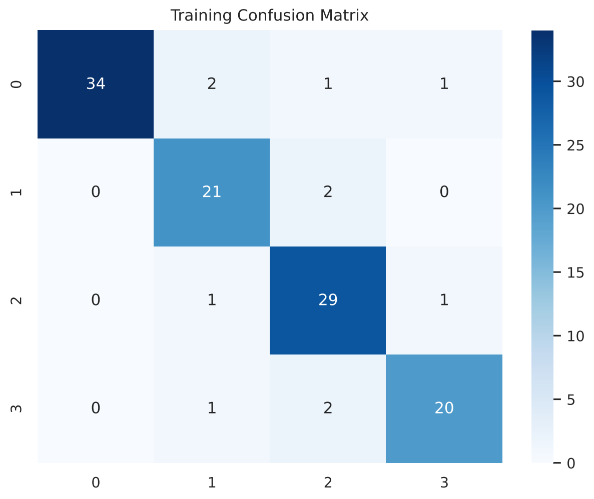

Figure 4.2.7: Training Confusion Matrix for ExTemp-Conv-LG

| | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| 0 | 1.00 | 0.89 | 0.94 | 38 |
| 1 | 0.84 | 0.91 | 0.88 | 23 |
| 2 | 0.85 | 0.94 | 0.89 | 31 |
| 3 | 0.91 | 0.87 | 0.89 | 23 |
| | | | | |
| accuracy | | | 0.90 | 115 |
| macro avg | 0.90 | 0.90 | 0.90 | 115 |
| weighted avg | 0.91 | 0.90 | 0.91 | 115 |

Table 4.2.4: Training Classification Report for ExTemp-Conv-LG

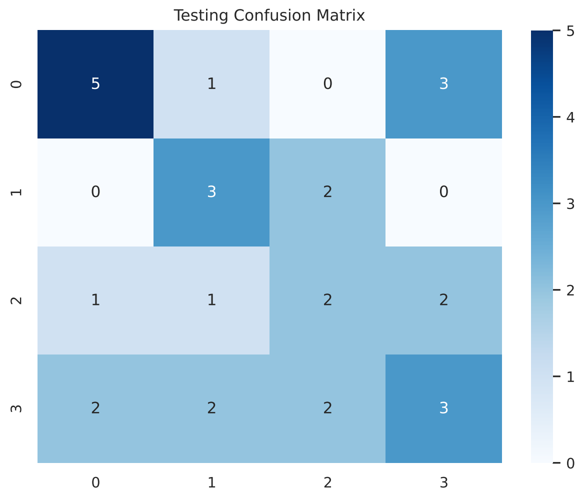

Figure 4.2.8: Testing Confusion Matrix for ExTemp-Conv-LG

| | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| 0 | 0.62 | 0.56 | 0.59 | 9 |
| 1 | 0.43 | 0.60 | 0.50 | 5 |
| 2 | 0.76 | 0.67 | 0.57 | 6 |
| 3 | 0.58 | 0.53 | 0.55 | 9 |
| | | | | |
| accuracy | | | 0.70 | 29 |
| macro avg | 0.60 | 0.58 | 0.56 | 29 |
| weighted avg | 0.60 | 0.57 | 0.55 | 29 |

Table 4.2.5: Testing Classification Report for ExTemp-Conv-LG

### 4.2.3 ExTemp-Auto

The following results correspond to the training and testing of the ExTemp-Auto model. This model uses the encoder of an autoencoder to perform feature extraction.

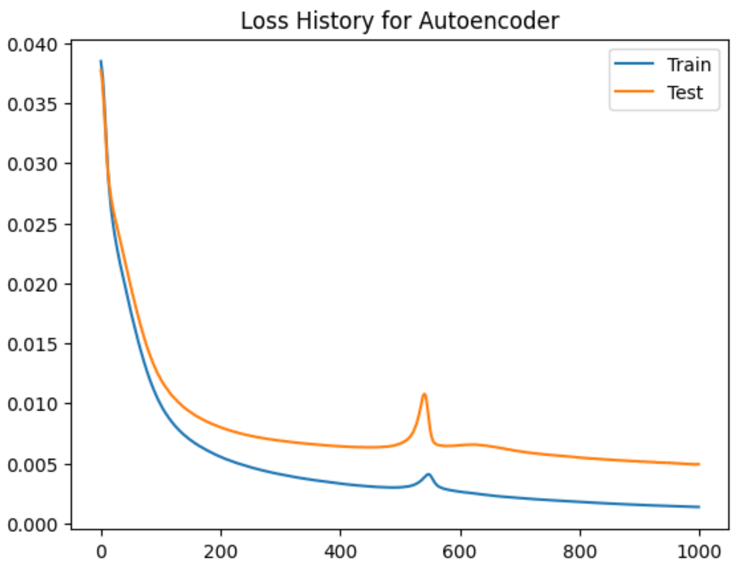

Figure 4.2.9: Loss Curves for Autoencoder in ExTemp-Auto

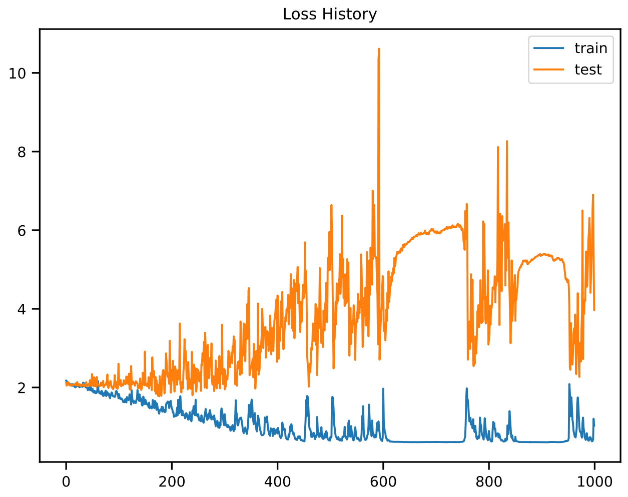

Figure 4.2.10: Loss Curves for ExTemp-Auto

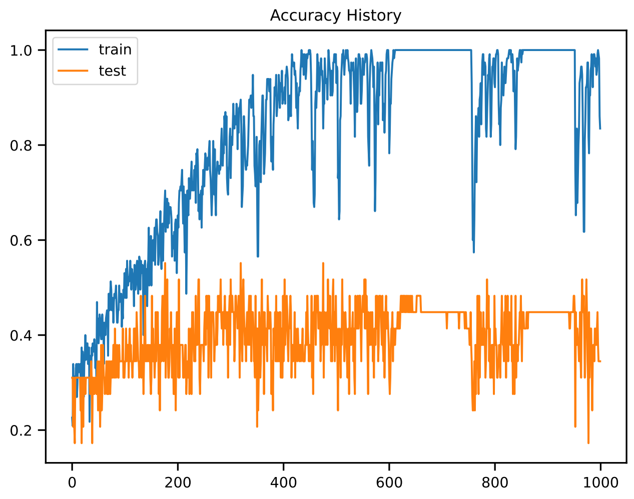

Figure 4.2.11: Accuracy Curves for ExTemp-Auto

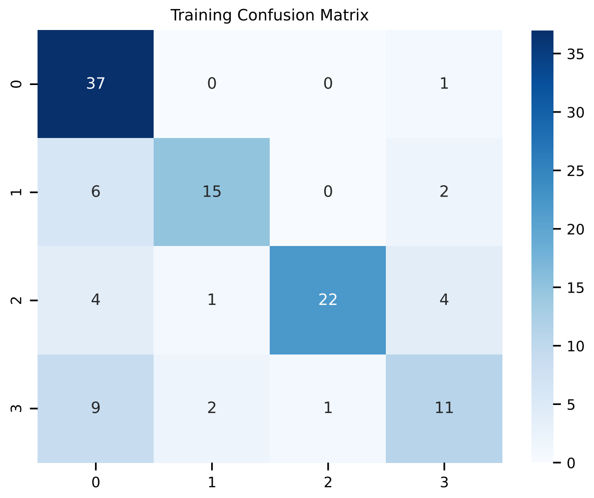

Figure 4.2.12: Training Confusion Matrix for ExTemp-Auto

|  | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| 0 | 0.66 | 0.97 | 0.79 | 38 |
| 1 | 0.83 | 0.65 | 0.73 | 23 |
| 2 | 0.96 | 0.71 | 0.81 | 31 |
| 3 | 0.61 | 0.48 | 0.54 | 23 |
|  |  |  |  |  |
| accuracy |  |  | 0.74 | 115 |
| macro avg | 0.77 | 0.70 | 0.72 | 115 |
| weighted avg | 0.77 | 0.74 | 0.73 | 115 |

Table 4.2.6: Training Classification Report for ExTemp-Auto

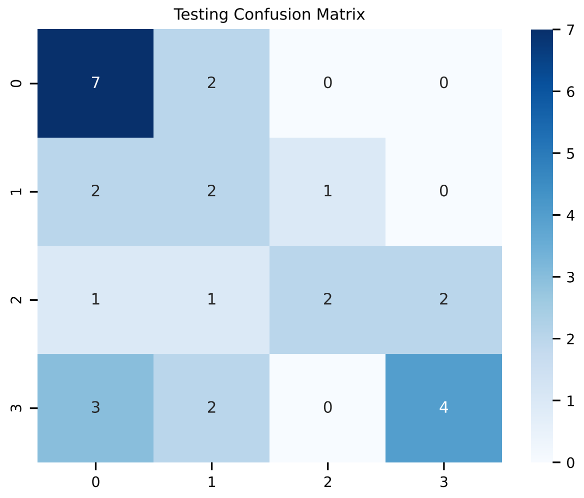

Figure 4.2.13: Testing Confusion Matrix for ExTemp-Auto

|  | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| 0 | 0.54 | 0.78 | 0.64 | 9 |
| 1 | 0.29 | 0.40 | 0.33 | 5 |
| 2 | 0.67 | 0.33 | 0.44 | 6 |
| 3 | 0.67 | 0.44 | 0.53 | 9 |
|  |  |  |  |  |
| accuracy |  |  | 0.52 | 29 |
| macro avg | 0.54 | 0.49 | 0.49 | 29 |
| weighted avg | 0.56 | 0.52 | 0.51 | 29 |

Table 4.2.7: Testing Classification Report for ExTemp-Auto

We note that the performance of this model on the test data is less than compared to the convolution based models. We presume it is due to the lack of data - for the encoding model to learn proper and useful encodings, a lot of data is required for it to learn. Another trend we note with this model is that the class with the least support especially suffers. We attribute this to the inability to properly learn the characteristics of that particular disease's temperature variations.

## 4.3 GradCAM

Below we present the results of running the GradCAM analysis. We plot the points that have strongly positively contributed towards the classification of a particular disease. We define "strongly positively" contributed if the class activation map value is more than the mean positive class activation map value. We only consider the mean of the positive values, as there might be data points that are negatively contributing toward classification. This essentially means points that encourages the model to not consider the selected class. Although this might be of interest in other analytical studies, we do not use this metric. We are more interested in finding which patterns are contributing towards the classification decision.

### 4.3.1 ExTemp-Conv-SM

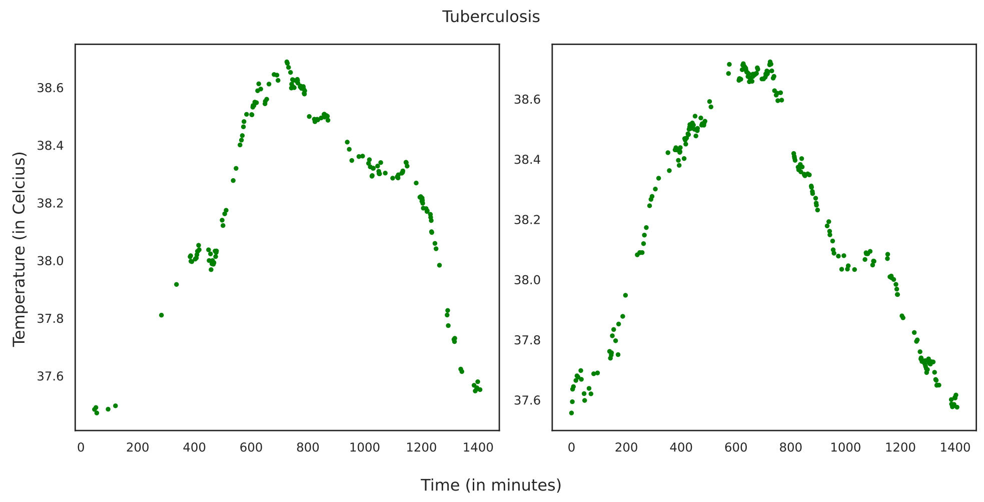

Figure 4.3.1: Plots for Tuberculosis generated by ExTemp-Conv-SM

Examining the graphs, it becomes evident that the model accurately captures the gradual rise in temperature, maintenance of this trend, followed by a gradual decline. This observed pattern aligns closely with the findings documented by Dakappa et al. [7], indicating a consistent temporal rhythm in temperature variations. Furthermore, the model demonstrates attentiveness to finer details such as saddle points and other fluctuations within the overarching trend.

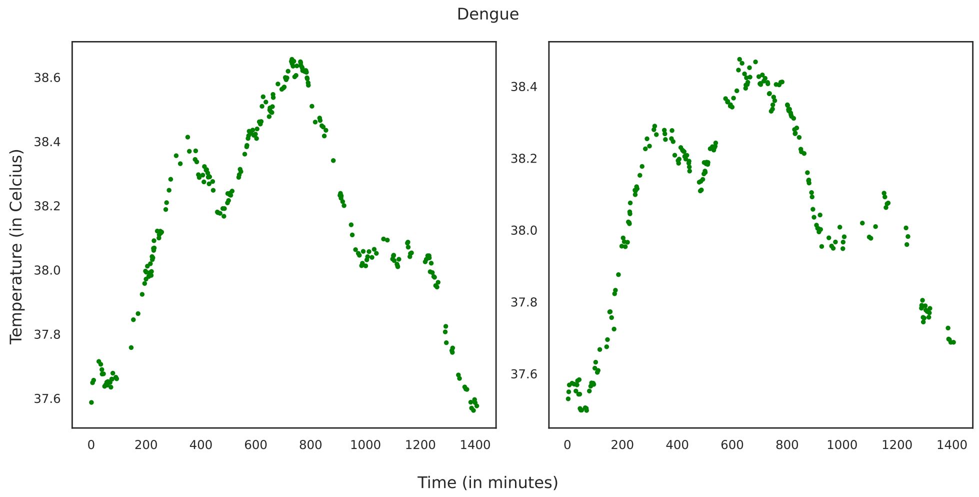

Figure 4.3.2: Plots for Dengue generated by ExTemp-Conv-SM

Here, the model accurately recognizes the temperature's cyclic nature, mirroring observed increases and decreases. These patterns correspond with the tri-phasic temperature phenomena outlined by Dakappa et al [6]. The model is able to do this even in the noisy dengue temperature data.

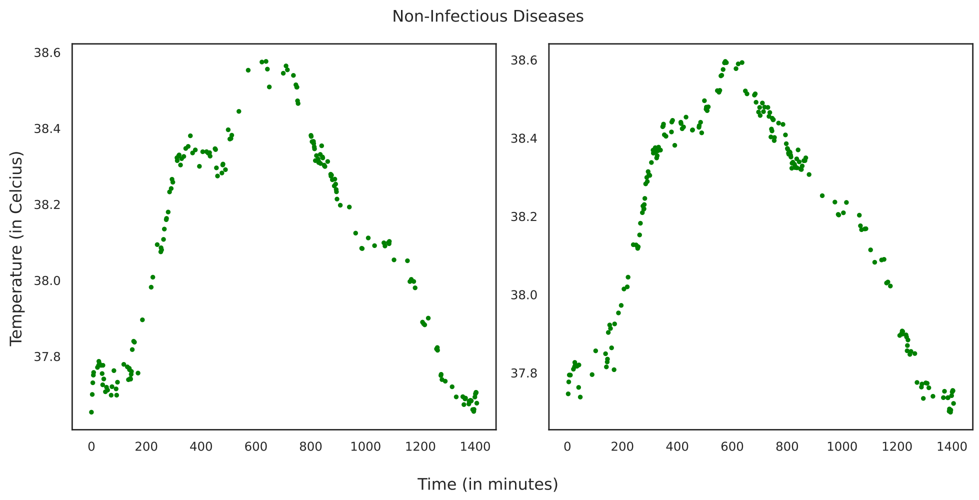

Figure 4.3.3: Plots for Non-Infectious Diseases generated by ExTemp-Conv-SM

Unlike the previous graphs, the temperature here exhibits a distinct increase with fluctuations during the morning hours, reaching its peak around noon. However, what distinguishes this pattern is the rapid decline in temperature observed within five hours after reaching its peak.

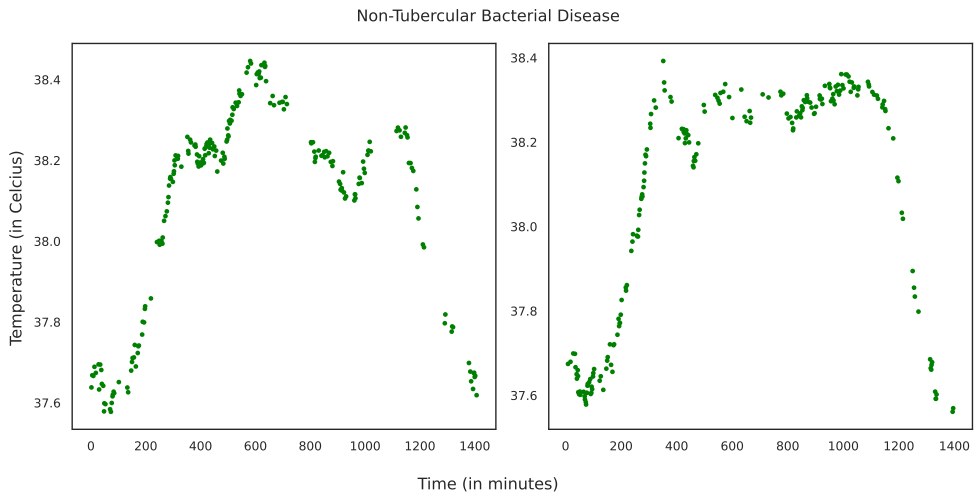

Figure 4.3.4: Plots for Non-Tubercular Bacterial Diseases generated by ExTemp-Conv-SM

In this, we note fluctuations in temperature throughout the day except noon. A mild increase in temperature around mid-morning is followed by a sharp spike around noon, followed by a period of relative stability throughout the afternoon and into the evening, before a gradual decline.

### 4.3.2 ExTemp-Conv-LG

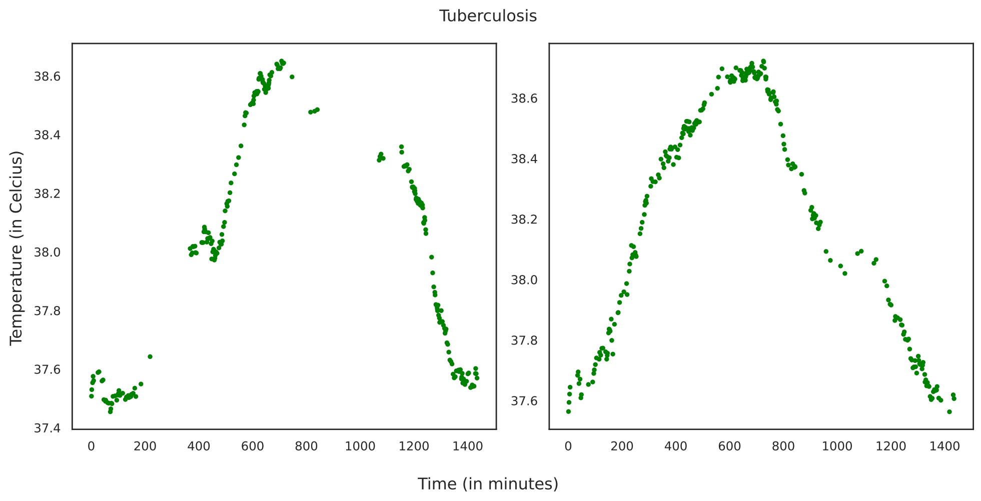

Figure 4.3.5: Plots for Tuberculosis generated by ExTemp-Conv-LG

As identified by the ExTemp-Conv-SM model, we can see the gradual rise of temperature, and the fall. It is more pronounced in this case, as we can see the smooth increase an decrease.

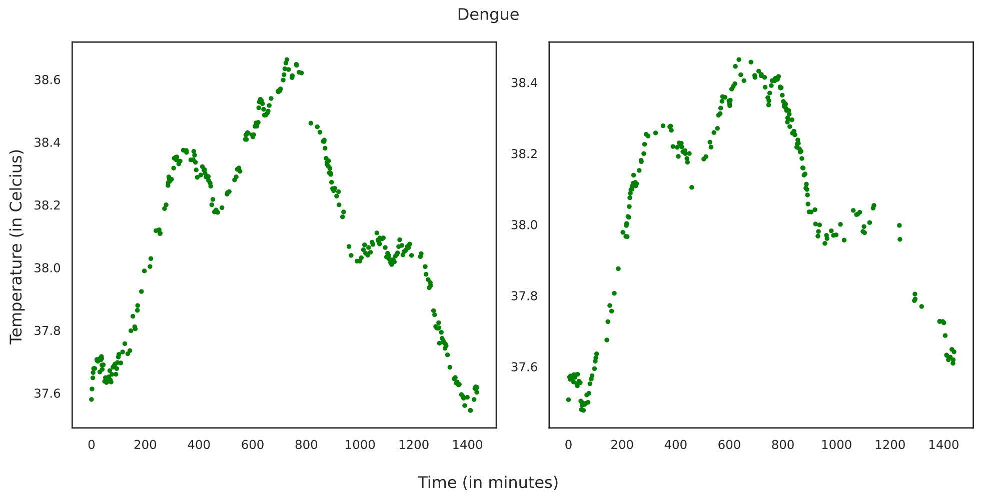

Figure 4.3.6: Plots for Dengue generated by ExTemp-Conv-LG

Similar to the previous model, we also note the tri-phasic temperature pattern here. However, we see more importance being given to the peaks (as evidenced by the higher density of points at those moments). This denotes that this model is capturing this trend well, even in periods of noise.

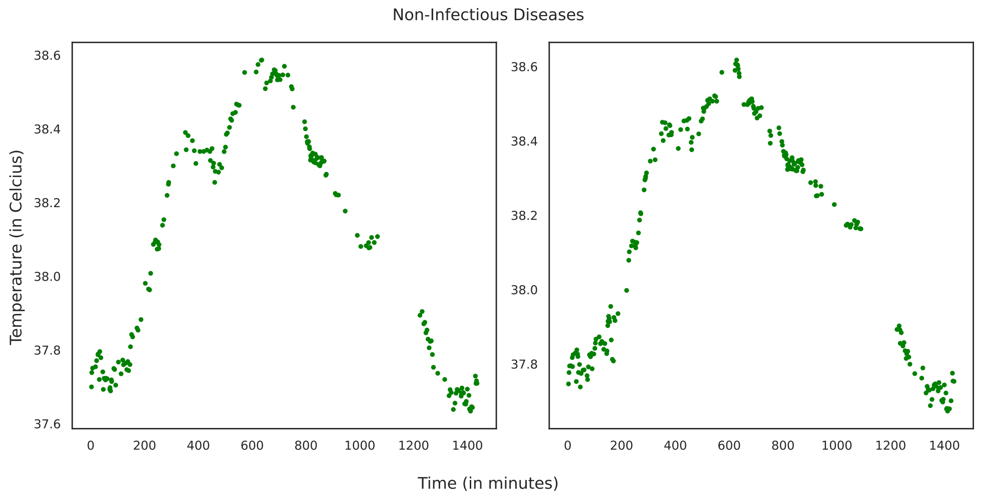

Figure 4.3.7: Plots for Non-Infectious Diseases generated by ExTemp-Conv-LG

In this graph, the temperature fluctuates mildly throughout the day, with a slight increase around mid-morning followed by a gradual decline. However, around noon, there's a sharp spike in temperature, peaking in the afternoon, before gradually returning to normal levels by evening.

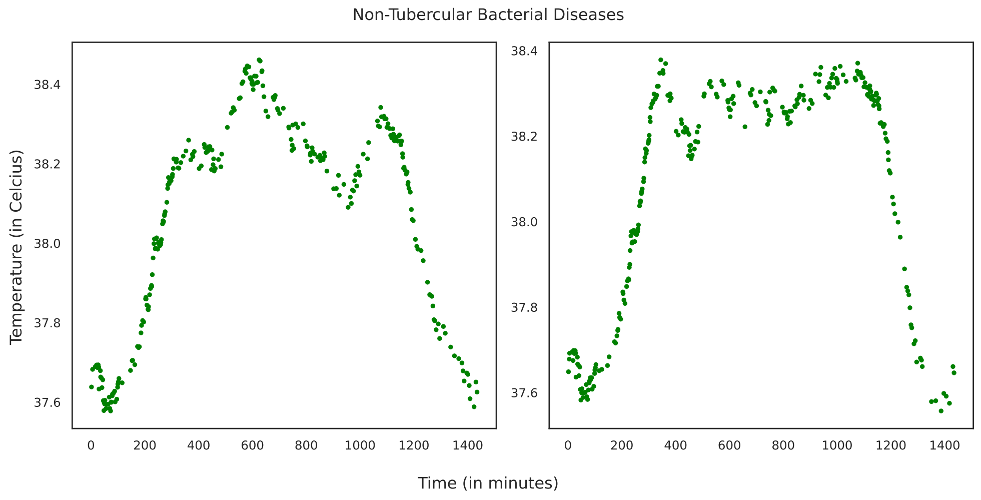

Figure 4.3.8: Plots for Non-Tubercular Bacterial Diseases generated by ExTemp-Conv-LG

Here, the temperature starts to rise steadily around late morning, peaking in the early afternoon before gradually declining toward evening becomes relatively stable across the mean range of temperature. The rise in temperature is more stable at noon compared to the previous graph for the same class.

In general, we note that these observations match those of the smaller convolution model. However, there are advantages in using the larger model. We can see this in the resolution of patterns visible, and the density of points at critical times.

### 4.3.3 ExTemp-Auto

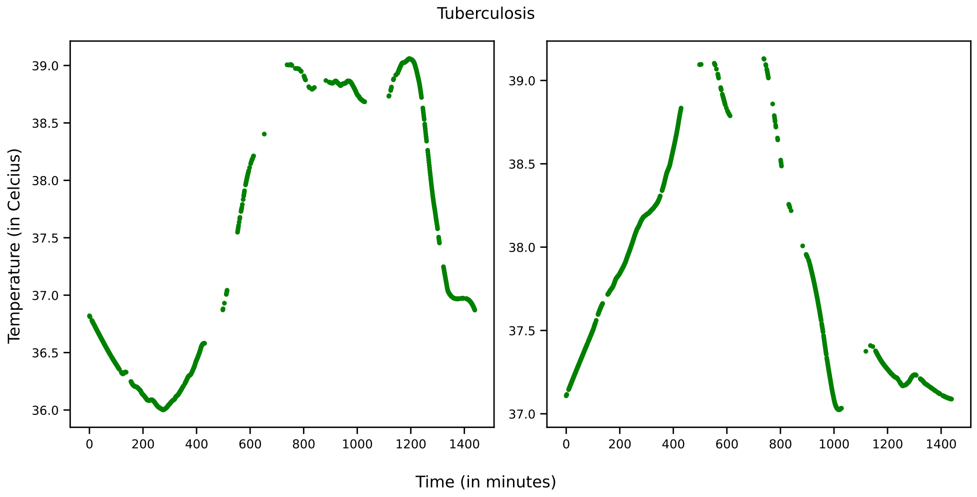

Figure 4.3.9: Plots for Tuberculosis generated by ExTemp-Auto

As identified by the convolution based models, we can see the gradual rise of temperature, and the fall. There is also some focus on the characteristic of temperature fluctuations at its peak.

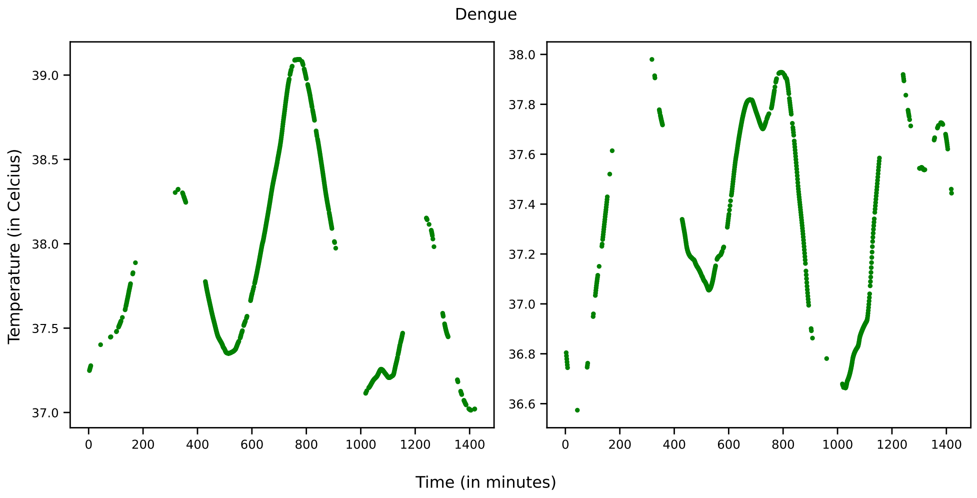

Figure 4.3.10: Plots for Dengue generated by ExTemp-Auto

Similar to the previous model, we also note the tri-phasic temperature pattern here. One thing to note here is the prolonged importance attributed to these peaks. Importance is given to data points well before and after the appearance of the peak.

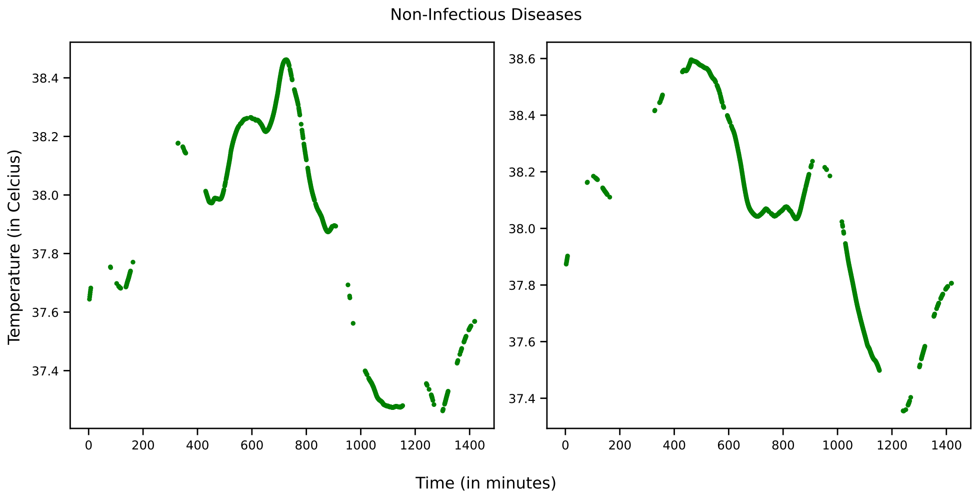

Figure 4.3.11: Plots for Non-Infectious Diseases generated by ExTemp-Auto

In this graph, the temperature fluctuates mildly throughout the day, with a slight increase around mid-morning followed by a gradual decline. This has been captured very well. However, initial characteristics about temperature patterns in the morning has not been captured fully.

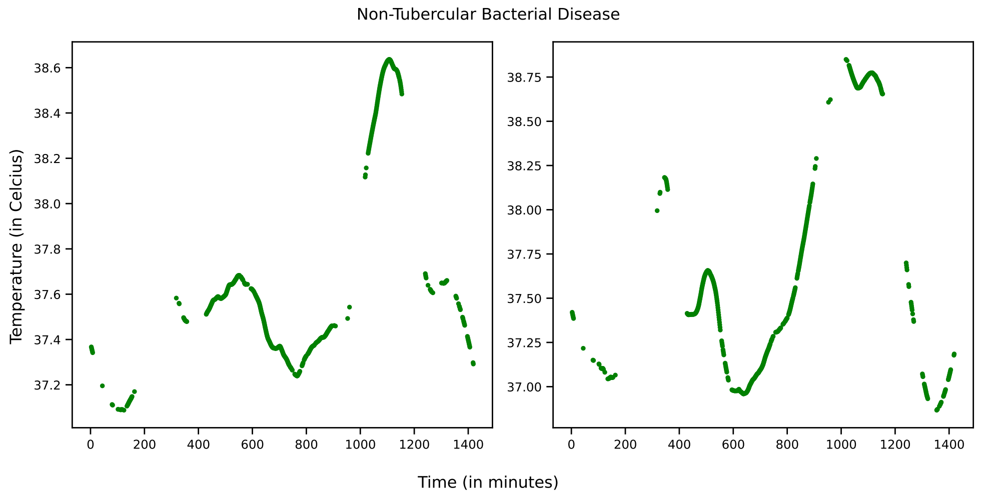

Figure 4.3.12: Plots for Non-Tubercular Bacterial Diseases generated by ExTemp-Auto

Here, we note that unlike other models, there has been importance given to the valleys exclusively. The trend of reaching a minima, and then moving towards the maximum temperature of the day is visible.

We note that the graphs generated by the autoencoder based model is not as smooth as the ones created by the convolution based model. However, there is a very good grasp of temporal information - one could argue that it beats that of the convolution models. With proper amounts of data and training data, we believe that this model would perform very well.

## 4.4 Summary of Results

We introduce three new models for this use case. Their performance is summarized in the tables below.

| **model** | **precision** | **recall** | **f1-score** | **accuracy** |
|---|---|---|---|---|
| ExTemp-Conv-SM | 0.98 | 0.97 | 0.97 | 0.97 |
| ExTemp-Conv-LG | 0.91 | 0.90 | 0.91 | 0.90 |
| ExTemp-Auto | 0.77 | 0.74 | 0.74 | 0.74 |

Table 4.4.1: Summary of training metrics.

| **model** | **precision** | **recall** | **f1-score** | **accuracy** |
|---|---:|---:|---:|---:|
| ExTemp-Conv-SM | 0.62 | 0.55 | 0.62 | 0.65 |
| ExTemp-Conv-LG | 0.60 | 0.57 | 0.55 | **0.70** |
| ExTemp-Auto | 0.56 | 0.52 | 0.51 | 0.52 |

Table 4.4.2: Summary of testing metrics.

We note that the ExTemp-Conv-LG model performs the best out these three models. Using GradCAM analysis, we were also able to find distinctive patterns in the temperature data. We were also able to confirm these findings with previous observations available in literature.

# CHAPTER 5

# CONCLUSIONS AND FUTURE SCOPE

## 5.1 Conclusions

In this project, we explored two tasks. First, we looked into deploying deep learning algorithms and techniques for disease classification from 24-hour tympanic temperature data. We were able to devise three new models for this use-case - two based on convolution and one based on autoencoders. Secondly, we looked at leveraging explainable AI tools to understand how models were making these decisions. By observing those trends, we were able to improve our understanding of temperature patterns for various diseases.

In conclusion, this projects aims to contribute to the ever-growing field of medical machine learning. The results and outcomes of this project aim to be an effective tool available in the arsenal for non-invasive diagnostics. These findings may have important implications in diagnostic and clinical settings, and can be used with other traditional differential diagnoses techniques, and other AI-enhanced medical procedures.

## 5.2 Future Scope

We introduce three new models for consideration of this problem. However, there is still work left to be done. The first step will be to formalize our findings in the form of a publication at a reputed medical machine learning and/or explainable AI for healthcare conference.

There are a couple of improvements that can be made to the models itself. One very important thing would be to gather more data, so that we can train our models better. Another important step would be to get a control set - data points that correspond to non-fever people. This will allow the model to learn what the baselines are, improving its understanding of the patterns. Finally, in the autoencoder model, we have currently implemented a simple Autoencoder. However, we can look into using more sophisticated techniques like Variational Autoencoders (VAE) and Convolutional Variational Autoencoders.

# REFERENCES

[1] David Cuesta Frau, Pau Miró Martínez, Sandra Oltra Crespo, Antonio Molina Picó, Borja Vargas, Paula González, Chakrapani Mahabala, and Pradeepa H Dakappa. Classification of fever patterns using a single extracted entropy feature: A feasibility study based on sample entropy. *Mathematical Biosciences and Engineering*, 17(1):235–249, 2019.

[2] David Cuesta-Frau, Pradeepa H Dakappa, Chakrapani Mahabala, and Arjun R Gupta. Fever time series analysis using slope entropy. application to early unobtrusive differential diagnosis. *Entropy*, 22(9):1034, 2020.

[3] David Cuesta-Frau. Slope entropy: A new time series complexity estimator based on both symbolic patterns and amplitude information. *Entropy*, 21(12):1167, 2019.

[4] Pradeepa H Dakappa, Keerthana Prasad, Sathish B Rao, Ganaraja Bolumbu, Gopalkrishna K Bhat, Chakrapani Mahabala, et al. A predictive model to classify undifferentiated fever cases based on twenty-four-hour continuous tympanic temperature recording. *Journal of Healthcare Engineering*, 2017, 2017.

[5] Pradeepa Hoskeri Dakappa, Keerthana Prasad, Sathish B Rao, Ganaraja Bolumbu, Gopalkrishna K Bhat, and Chakrapani Mahabala. Classification of infectious and non-infectious diseases using artificial neural networks from 24-hour continuous tympanic temperature data of patients with undifferentiated fever. *Critical Reviews™ in Biomedical Engineering*, 46(2), 2018.

[6] D PH, Sathish B Rao, G Bhat, et al. Tri-phasic fever in dengue fever. *Tropical doctor*, 48(2):93–97, 2018.

[7] Pradeepa H Dakappa, Sathish B Rao, Gopalkrishna K Bhat, and Chakrapani Mahabala. Unique temperature patterns in 24-h continuous tympanic temperature in tuberculosis. *Tropical Doctor*, 49(2):75–79, 2019.

[8] Bjørn-Jostein Singstad and Christian Tronstad. Convolutional neural network and rule-based algorithms for classifying 12-lead ecgs. In *2020 Computing in Cardiology*, pages 1–4. IEEE, 2020.

[9] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[10] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[11] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J Inman. 1d convolutional neural networks and applications: A survey. *Mechanical systems and signal processing*, 151:107398, 2021.

[12] Anqi Mao, Mehryar Mohri, and Yutao Zhong. Cross-entropy loss functions: Theoretical analysis and applications. In *International Conference on Machine Learning*, pages 23803–23828. PMLR, 2023.

[13] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[14] Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. Captum: A unified and generic model interpretability library for pytorch, 2020.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[16] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.