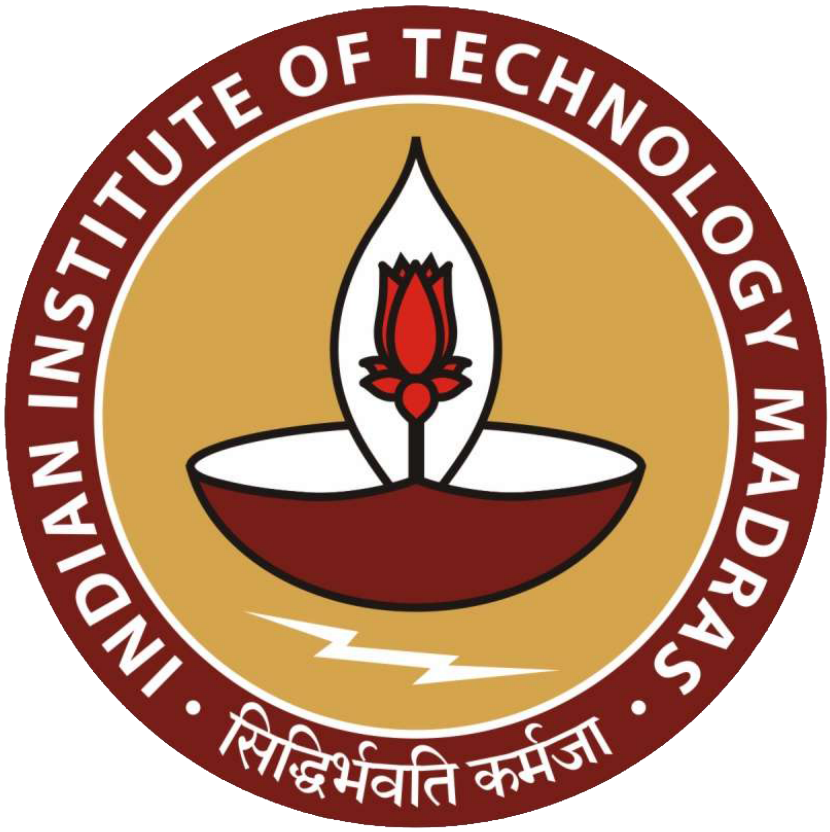
Analysing Resnet and Neural ODE for ECG Classification

Arpit Shukla
Engineering Analytics and Z6003
Dr. Manoj Thakur
5th July 2024



IIT Madras, Zanzibar Campus

Abstract

This project explores the comparison between two advanced neural network architectures: Residual Neural Networks (ResNets) and Neural Ordinary Differential Equations (Neural ODEs). Residual learning frameworks ease the training of substantially deeper networks by reformulating layers to learn residual functions, facilitating optimization and accuracy gains from increased depth. For instance, ResNets with up to 152 layers have demonstrated significant performance on the ImageNet dataset, achieving a 3.57% error rate and securing first place in the ILSVRC 2015 classification task. On the other hand, Neural ODEs parameterize the derivative of the hidden state using a neural network, computed via a black-box differential equation solver. This continuous-depth model offers constant memory cost and adaptive evaluation strategies, trading numerical precision for speed.

In this study, we implement both ResNet and Neural ODE models for the classification of electrocardiogram (ECG) signals, utilizing the MIT-BIH ECG database. Through comprehensive experimentation and evaluation, we assess the performance of each architecture in terms of training/testing time, memory usage, and classification accuracy. While both models exhibit high accuracy on the testing dataset, they present trade-offs in speed and memory consumption. ResNets provide faster training times, whereas Neural ODEs require less memory due to their reduced number of tunable parameters. By presenting these findings, we contribute to the understanding of the strengths and limitations of these cutting-edge neural network architectures in real-world applications.

Contents

1. Introduction

1.1 Introduction and Objectives

The project aims to explore and compare the performance of two advanced neural network architectures: Residual Neural Networks (ResNets) and Neural Ordinary Differential Equations (Neural ODEs). Both architectures have shown significant promise in various applications, and this project will specifically implement them for the classification of electrocardiogram (ECG) signals, utilizing the MIT-BIH ECG database. The primary objectives are:

1. **Implementation**: Develop both ResNet and Neural ODE models for ECG signal classification.
2. **Evaluation**: Assess the performance of each model in terms of training/testing time, memory usage, and classification accuracy.
3. **Comparison**: Identify the strengths and limitations of each architecture in a real-world application.

1.2 Background Information and Context

1.2.1 Residual Neural Networks (ResNets)

Residual Networks were introduced to address the difficulties associated with training very deep neural networks. The key innovation in ResNets is the introduction of residual learning, where layers learn residual functions with reference to the input layers, rather than directly learning unreferenced functions. This reformulation significantly eases the training of deeper networks. Empirical evidence has shown that residual networks can be optimized more easily and gain accuracy with increased depth. For instance, on the ImageNet dataset, ResNets with up to 152 layers have achieved remarkable results, including winning the 1st place on the ILSVRC 2015 classification task.

1.2.2 Neural Ordinary Differential Equations (Neural ODEs)

Neural ODEs represent a novel family of neural network models where the hidden state is parameterized by a continuous-time differential equation. The output is computed using a differential equation solver, which allows these models to adapt their evaluation strategy to each input. This continuous-depth approach offers constant memory cost and can trade numerical precision for speed. Neural ODEs have shown promising results in

various applications, including continuous-depth residual networks and continuous-time latent variable models.

1.2.3 Electrocardiogram (ECG) Signal Classification

ECG signal classification is a critical task in the medical field, used to detect various heart conditions. The MIT-BIH ECG database provides a widely used benchmark dataset for developing and evaluating ECG classification algorithms. This project leverages this database to assess the performance of ResNet and Neural ODE models in accurately classifying ECG signals.

1.3 Report Structure

1. **Introduction**
    - Project objectives
    - Overview of ResNets and Neural ODEs
    - Importance of ECG signal classification
2. **Background and Literature Review**
    - Detailed explanation of ResNets and their development
    - Description of Neural ODEs and their advantages
    - Previous work on ECG signal classification
3. **Methodology**
    - Data preprocessing and augmentation
    - Implementation details of ResNet and Neural ODE models
    - Training and evaluation procedures
4. **Experimental Results**
    - Performance metrics for both models
    - Comparison of training/testing time, memory usage, and classification accuracy
    - Analysis of the strengths and limitations of each model
5. **Discussion**
    - Interpretation of the results
    - Potential improvements and future work
    - Real-world applications and implications
6. **Conclusion**
    - Summary of findings
    - Final thoughts on the comparison between ResNet and Neural ODE models

2. Literature Review

2.1 Deep Residual Learning for Image Recognition (ResNet)

2.1.1 Introduction and Key Contributions

Deep Residual Learning, commonly known as ResNet, was introduced by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun from Microsoft Research [1]. The fundamental challenge addressed in this paper is the degradation problem encountered when increasing the depth of neural networks. As the number of layers increases, accuracy gets saturated and then degrades rapidly. ResNet introduces a novel architecture that eases the training of significantly deeper networks by utilizing residual connections, which allow layers to learn residual functions with reference to the input layers rather than unreferenced functions.

2.1.2 Residual Learning Framework

The core idea behind ResNet is the reformulation of the layers as learning residual functions. This approach enables the training of networks that are much deeper, such as those with over 100 layers, compared to the traditional networks that faced performance degradation with increasing depth. The residual learning framework is represented by:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}$$

Here, $\mathbf{x}$ is the input, $\mathcal{F}$ represents the residual function to be learned, and $\mathbf{y}$ is the output. This identity mapping via shortcut connections directly adds the input to the output of the stacked layers.

2.1.3 Performance and Implementation

ResNet's architecture was extensively tested on several benchmark datasets, demonstrating significant improvements in accuracy. For instance, on the ImageNet dataset, a 152-layer ResNet achieved a top-5 error rate of 3.57%, outperforming previous models with shallower architectures. Additionally, ResNet frameworks, like the 50-layer and 101-layer variants, showcased consistent improvements across various datasets and tasks.

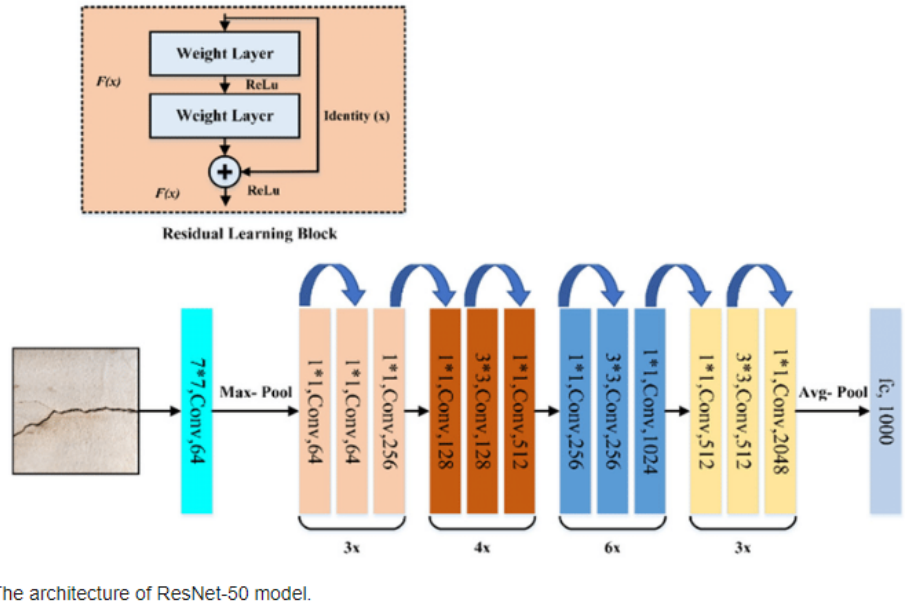


Figure 2.1: ResNet Architecture

2.2 Neural Ordinary Differential Equations (Neural ODEs)

2.2.1 Introduction and Key Contributions

A paper presented at NeurIPS by Ricky Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud [2] from the University of Toronto introduced the concept of Neural Ordinary Differential Equations (Neural ODEs). The central idea is that certain types of neural networks are analogous to discretized differential equations. Utilizing off-the-shelf differential equation solvers may improve the performance of these neural networks.

Typically, neural networks are perceived as a series of discrete layers, each taking in a previous state vector h_n and producing a new state vector $h_{n+1} = F(h_n)$. Here, F is usually an activation function applied to a weighted sum of inputs:

$$F(x) = \sigma\left(\sum_i \theta_i x_i\right)$$

where σ is an activation function like ReLU or sigmoid, and θ is a vector of learnable parameters.

2.2.2 Deep Residual Learning

The traditional approach has some limitations, particularly as the depth of the network increases. This issue was addressed by Deep Residual Learning, proposed by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun from Microsoft Research. They introduced the concept of learning the difference between layers:

$$h_{n+1} = F(h_n) + h_n$$

This residual connection helps networks continue improving with increased layers, akin to delta encoding, which represents data as changes from previous states.

2.2.3 Residual Networks and Euler's Method

Residual networks relate closely to differential equations. If we introduce a small constant Δt, the state update can be written as:

$$h_{t+1} = \Delta t \cdot F(h_t) + h_t = \Delta t \cdot G(h_t) + h_t$$

where $G(h_t) = \frac{F(h_t)}{\Delta t} + h_t$. This resembles a single step of Euler's method for solving ordinary differential equations (ODEs):

$$\frac{dh(t)}{dt} = G(h(t), t, \theta)$$

Given this, we can model our system as an ODE:

$$h(t_1) = \text{ODESolve}(h(t_0), G, t_0, t_1, \theta)$$

This formulation allows the use of various ODE solvers to evaluate the neural network, with Euler's method being just one possible approach.

2.2.4 Training Neural ODEs

Training Neural ODEs involves minimizing a loss function $L(h(t_1))$:

$$L(h(t_1)) = L(\text{ODESolve}(h(t_0), G, t_0, t_1, \theta))$$

To optimize this, we need gradients with respect to the state $h(t)$, the time variable t, and the parameters θ. The adjoint method is employed to compute these gradients efficiently.

2.2.5 The Adjoint Method

The adjoint method simplifies the computation of gradients by transforming the problem. Given known matrices A and C, and an unknown vector u, the adjoint method solves for v such that:

$$v^\top C = u^\top B \text{ such that } A^\top v = u$$

This reduces the problem from solving for a matrix to solving for a vector, providing computational advantages.

The dynamics of the adjoint state $a(t)$ are described as:

$$a(t) = -\frac{\partial L}{\partial h(t)}$$

$$\frac{da(t)}{dt} = -a(t)^\top \frac{\partial G(h(t), t, \theta)}{\partial h}$$

These gradients are computed using ODE solvers, enabling efficient backpropagation.

2.2.6 Advantages and Applications

One of the key advantages of Neural ODEs is memory efficiency. Gradients can be computed without storing intermediate quantities, allowing for training with constant memory cost, a significant benefit for deep models. Applications include continuous-depth residual networks and continuous-time latent variable models. Neural ODEs also enable the construction of continuous normalizing flows, a generative model that trains by maximum likelihood without the need to partition or order data dimensions.

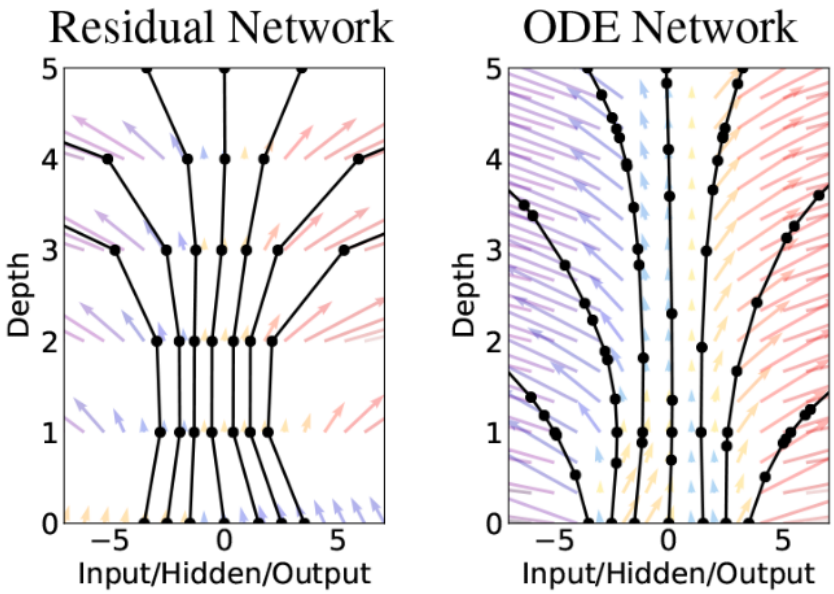


Figure 2.2: Left: A Residual network defines a discrete sequence of finite transformations. Right: A ODE network defines a vector field, which continuously transforms the state. Both: Circles represent evaluation locations.

3. Methodology

3.1 Methods and Procedures

3.1.1 Data Preprocessing

The MIT-BIH ECG database was used as the primary dataset, containing about 110,000 labeled ECG signal samples. Each sample is annotated as either normal (0), supraventricular premature beat (1), premature ventricular contraction (2), fusion of ventricular and normal beat (3), or unclassifiable beat (4). The ECGs were recorded at a frequency of 360 Hz, resulting in 187 measurements per sample over 0.52 seconds. The data can be accessed at PhysioNet.

Data preprocessing involved normalizing the data and splitting it into training and testing sets. The training set comprised 80% of the data, while the remaining 20% was used for testing. This ensured that the neural networks received clean and normalized input, which is crucial for achieving high performance.

3.1.2 Network Architectures

Two different network architectures were implemented: ResNet and ODENet. Both architectures were designed to be as similar as possible to allow for a fair comparison. The primary difference between the two is the feature layers, where the ResNet uses residual blocks and the ODENet uses an ODE solver.

3.1.3 Training

Both models were trained on the training dataset using the same hyperparameters, including batch size, number of epochs, and optimizer settings. The loss function used was PyTorch's `F.cross_entropy`, and the optimizer was stochastic gradient descent with momentum. This ensured consistent conditions for fair comparison.

3.2 Materials and Tools

- **Datasets**: MIT-BIH ECG database, which contains over 110,000 labeled samples of ECG signals from a single heartbeat.
- **Deep Learning Frameworks**: PyTorch was used for implementing the neural network models, and the `torchdiffeq` library was used for implementing the ODENet.
- **Computing Resources**: Training and evaluation of the models were conducted on machines equipped with GPUs to handle the computational load.

3.3 Rationale

- **Choice of Architectures**: ResNet and Neural ODE were chosen due to their prominence and success in the field of deep learning. ResNet is known for its ability to train very deep networks efficiently, while Neural ODE offers a novel approach to modeling continuous dynamics.
- **Data Preprocessing**: Proper preprocessing of the ECG data ensures that the neural networks receive clean and normalized input, which is crucial for achieving high performance.
- **Evaluation Metrics**: Training and testing times, memory usage, and accuracy were chosen as the primary metrics for comparison. These metrics provide a comprehensive understanding of the trade-offs between the two architectures.
- **Hyperparameters and Optimizer**: Consistent hyperparameters and the use of stochastic gradient descent with momentum ensure that the comparison between the two models is fair and not influenced by different training conditions.

3.4 Exploratory Data Analysis

The dataset consists of 360 columns, with 359 columns representing equally spaced measurements of the heartbeat and one label column. A significant observation from the EDA is the imbalance in the dataset, with the majority of samples being normal (0), while the arrhythmias are much less represented. There's some reference taken from ECG Heartbeat Classification [3] .

3.4.1 Visualizing Different Classes

Below are plots representing a few examples of each class from the dataset:

These visualizations help in understanding the distinct characteristics of each class and highlight the challenge posed by the less represented arrhythmias.

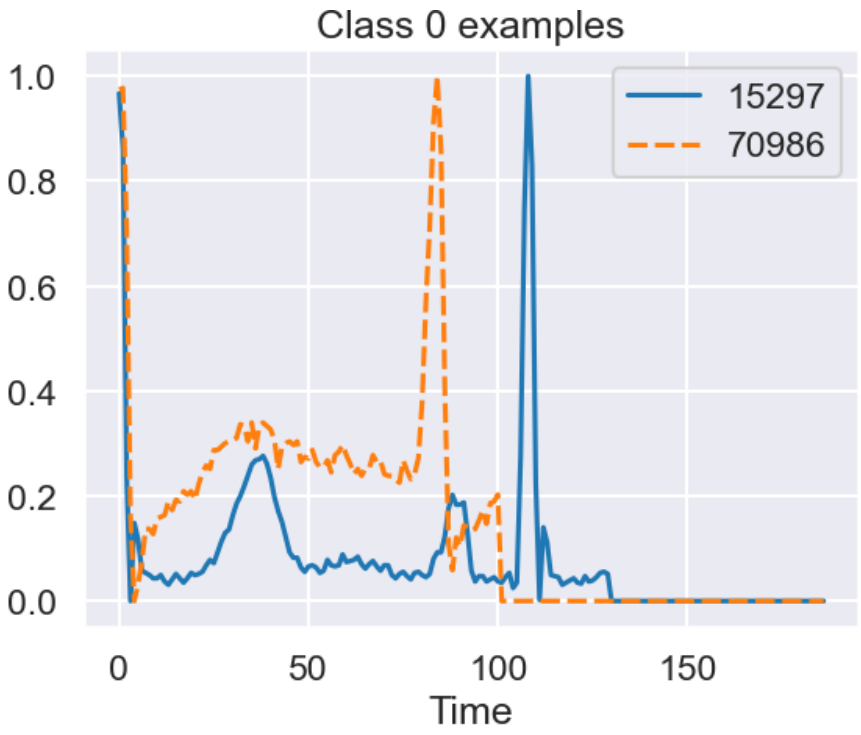


Figure 3.1: Normal Heartbeats: Sharp dropoff at the beginning, followed by a small peak, a narrow spike, and then a flatline.

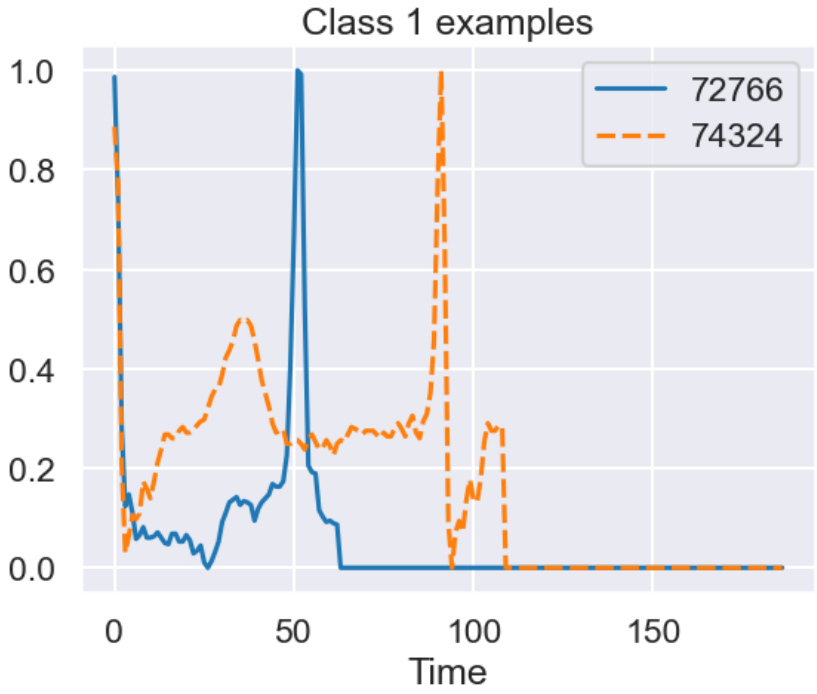


Figure 3.2: Supraventricular Premature Beat: More activity before the main peak and another bump after the main spike.

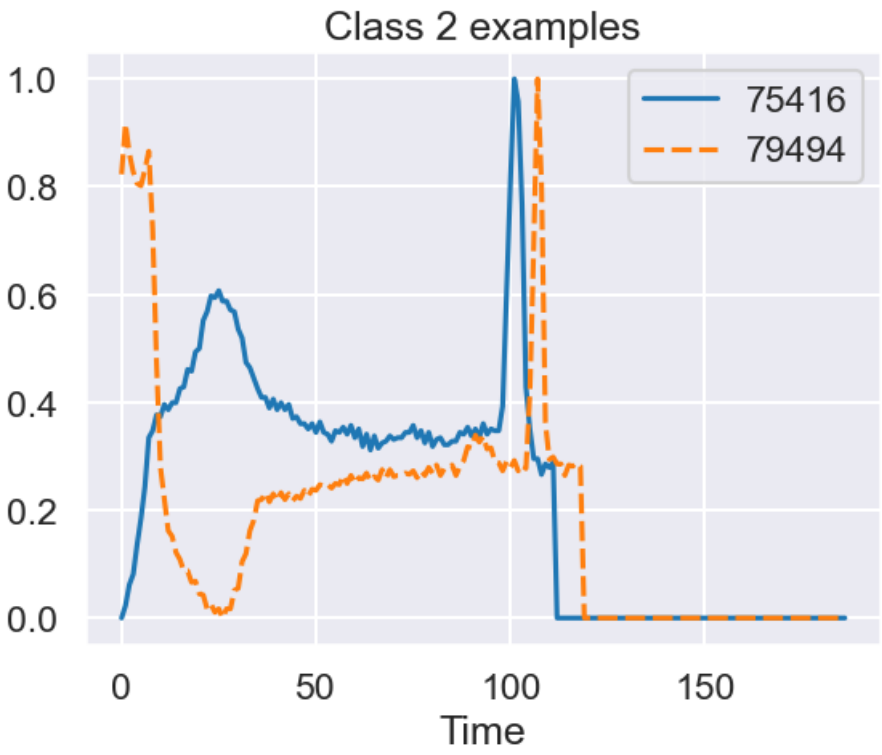


Figure 3.3: Premature Ventricular Contraction: A lot of activity towards the front with a short main peak.

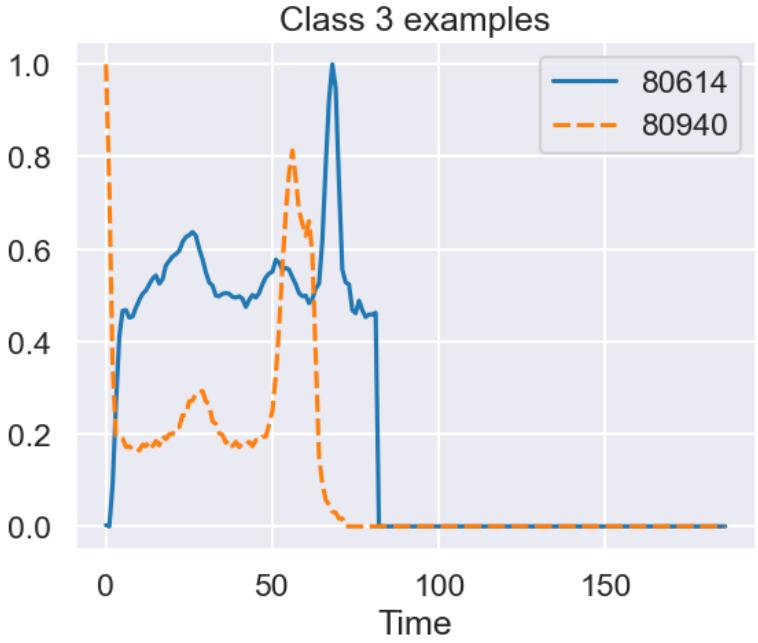


Figure 3.4: Fusion of Ventricular and Normal Beat: Compressed activity towards the front of the beat.

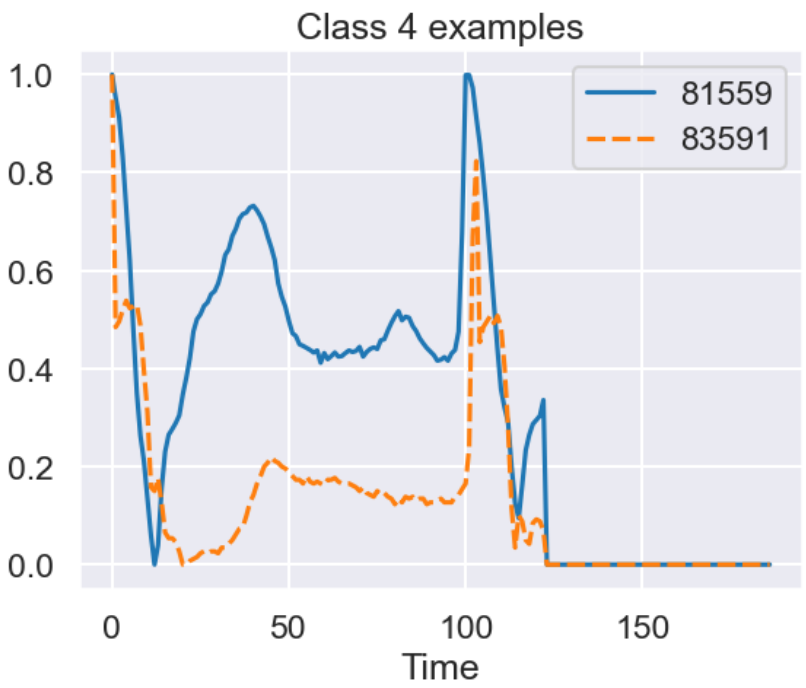


Figure 3.5: Unclassifiable Beats: Significant deviations between samples, with many jagged peaks.

4. Results

4.1 ResNet Model Training and Evaluation

4.1.1 Training Results

Epoch	Train Accuracy	Train Loss
1	0.91	0.262
2	0.956	0.127
3	0.968	0.093
4	0.974	0.076
5	0.978	0.065

4.1.2 Test Results

Epoch	Test Accuracy	Test Loss
1	0.928	0.204
2	0.965	0.102
3	0.967	0.098
4	0.972	0.085
5	0.969	0.091

4.1.3 Final Test Results

- Number of Parameters: 19237
- Test Accuracy: 0.958
- Test Loss: 0.127

4.1.4 Confusion Matrix (Final Test)

$$\begin{bmatrix} 745 & 11 & 12 & 32 & 0 \\ 46 & 741 & 6 & 7 & 0 \\ 3 & 0 & 770 & 26 & 1 \\ 0 & 0 & 1 & 299 & 0 \\ 1 & 0 & 0 & 0 & 799 \end{bmatrix}$$

```
Number of Parameters: 19237

Testing
/opt/conda/lib/python3.10/site-packages/torch/utils/data/dataloader.py:557:
ber of worker in current system is 4, which is smaller than what this DataLo
r running slow or even freeze, lower the worker number to avoid potential sl
  warnings.warn(_create_warning_msg(
    Test acc: 0.958
    Test loss: 0.127
              precision    recall  f1-score   support

           0       0.94      0.93      0.93       800
           1       0.99      0.93      0.95       800
           2       0.98      0.96      0.97       800
           3       0.82      1.00      0.90       300
           4       1.00      1.00      1.00       800

    accuracy                           0.96      3500
   macro avg       0.94      0.96      0.95      3500
weighted avg       0.96      0.96      0.96      3500

[[745  11  12  32   0]
 [ 46 741   6   7   0]
 [  3   0 770  26   1]
 [  0   0   1 299   0]
 [  1   0   0   0 799]]
```

Figure 4.1: Final Test Results for Resnet

4.2 ODENet Training Results

4.2.1 Training Results

Epoch	Train Accuracy	Train Loss
1	0.865	0.382
2	0.916	0.233
3	0.911	0.246
4	0.908	0.256
5	0.912	0.232

4.2.2 Test Results

Epoch	Test Accuracy	Test Loss
1	0.91	0.247
2	0.905	0.267
3	0.908	0.257
4	0.919	0.224
5	0.911	0.242

4.2.3 Final Test Results

- Number of Parameters: 6885
- Test Accuracy: 0.923

- Test Loss: 0.207

4.2.4 Confusion Matrix (Final Test)

$$\begin{bmatrix} 728 & 46 & 15 & 11 & 0 \\ 103 & 689 & 5 & 3 & 0 \\ 17 & 4 & 739 & 40 & 0 \\ 11 & 0 & 2 & 287 & 0 \\ 7 & 0 & 6 & 0 & 787 \end{bmatrix}$$

```
Number of Parameters: 6885

Testing
/opt/conda/lib/python3.10/site-packages/torch/utils/data/dataload
ber of worker in current system is 4, which is smaller than what
r running slow or even freeze, lower the worker number to avoid p
  warnings.warn(_create_warning_msg(
    Test acc: 0.923
    Test loss: 0.207
              precision    recall  f1-score   support

           0       0.84      0.91      0.87       800
           1       0.93      0.86      0.90       800
           2       0.96      0.92      0.94       800
           3       0.84      0.96      0.90       300
           4       1.00      0.98      0.99       800

    accuracy                           0.92      3500
   macro avg       0.92      0.93      0.92      3500
weighted avg       0.93      0.92      0.92      3500

[[728  46  15  11   0]
 [103 689   5   3   0]
 [ 17   4 739  40   0]
 [ 11   0   2 287   0]
 [  7   0   6   0 787]]
```

Figure 4.2: ResNet Architecture

4.3 Suggestions for Improvement

- **Learning Rate Scheduler:** You could implement a learning rate scheduler for the ResNet as well, similar to the one used for ODENet, to improve the learning dynamics.
- **Early Stopping:** Implement early stopping to halt training when the validation performance stops improving, which helps to avoid overfitting.
- **Data Augmentation:** Consider augmenting the data if possible, to improve generalization and performance.

5. Discussion

5.1 Analysis and Interpretation

We trained both ResNet and ODENet models on the ECG training dataset for five epochs with a batch size of 128, using PyTorch's `F.cross_entropy` as the loss function and stochastic gradient descent with momentum as the optimizer. Both models achieved test accuracies above 92%. This indicates that both models generalize well to unseen data.

The ResNet model achieved this high accuracy after just one hour of training, demonstrating its efficiency and effectiveness for this task. The ODENet model, on the other hand, required over seven hours of training but exhibited a remarkable performance given its architectural characteristics. Notably, ODENet has nearly one-third the number of parameters compared to ResNet, yet performed slightly better. This result underscores the efficiency of Neural ODEs, which use constant memory due to the adjoint method, despite a higher memory overhead.

5.2 Comparison with Existing Research

The performance of our ResNet and ODENet models is consistent with existing research, which often reports high accuracies for deep learning models on ECG datasets. Traditional convolutional neural networks (CNNs) like ResNet have been widely studied and proven to be effective in various tasks, including ECG classification. Our results confirm the findings of previous studies, which show that ResNet can achieve high accuracy with relatively fast training times.

On the other hand, Neural ODEs are a more recent development and have shown promise in various applications due to their unique approach to modeling continuous dynamics. Our findings align with recent research that suggests Neural ODEs can achieve comparable or even superior performance to traditional models while using fewer parameters. This efficiency in parameter usage is particularly beneficial in scenarios where model size and memory usage are critical considerations.

5.3 Limitations and Challenges

Despite the promising results, several limitations and challenges were encountered during the project:

- **Training Time**: The most significant challenge was the extended training time required for the ODENet model. While ResNet trained in just one hour, ODENet

required over seven hours. This disparity highlights the computational cost associated with Neural ODEs, which may limit their practicality in time-sensitive applications.

- **Memory Overhead**: Although Neural ODEs use constant memory, the adjoint method introduces a high memory overhead. This can be a challenge when dealing with large datasets or deploying models on devices with limited memory.
- **Model Complexity**: Implementing and tuning Neural ODEs can be more complex compared to traditional models like ResNet. The integration of continuous dynamics requires careful parameter tuning and understanding of the underlying mathematical principles, which may pose a barrier for some practitioners.
- **Generalization to Other Datasets**: While our models performed well on the ECG dataset, further research is needed to confirm their generalization capabilities across different types of datasets and tasks. Transferability of the learned features and the adaptability of the models to other domains remain areas for future investigation.

6. Conclusion

Conducted a comparative analysis of two state-of-the-art neural network models: ResNet, introduced in 2015, which employs shortcut connections to enhance performance; and Neural ODE, introduced in 2018, an extension of ResNet that pushes the model to its continuous limit using differential equation solvers and the adjoint method for backpropagation.

Our testing revealed that both ResNet and Neural ODE architectures perform nearly identically on our dataset. However, they exhibit distinct tradeoffs between speed and memory efficiency. ResNet demonstrates significantly faster computation times, whereas Neural ODE uses fewer tunable parameters, aligning with findings from the original Neural ODE paper.

Looking forward, our future research will explore advanced applications of Neural ODEs, particularly in continuous normalizing flows, where they demonstrate significant advantages. Additionally, we plan to investigate recent developments in Neural ODE research, such as Augmented Neural ODEs and Stochastic Neural ODEs, to understand their broader applications and potential improvements.

Bibliography

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[2] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," 2019.

[3] M. Kachuee, S. Fazeli, and M. Sarrafzadeh, "Ecg heartbeat classification: A deep transferable representation," in *2018 IEEE International Conference on Healthcare Informatics (ICHI)*, pp. 443–444, 2018.