Investigation of Few-Shot Learning for Fall Detection

by

Satyake Bakshi

A thesis submitted to the
Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of
Master of Applied Science in Biomedical Engineering

Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario, Canada

© April 2021, Satyake Bakshi
Abstract

Falls affect seniors’ quality of life, and therefore fall detection and prevention are paramount for the health and safety of aging seniors. Current deep learning-based fall detection methods perform well when a large amount of training data is available. As obtaining fall data from seniors is extremely difficult, training deep learning models is a challenge, and therefore, a few-shot Siamese network is considered in this thesis. A shallow $1 \times 1$ convolutional neural network for Siamese and Triplet networks is proposed in this work. A deeper architecture-based on the Inception and Densenet networks is also considered to improve the fall detection performance. The performances of the proposed few-shot Siamese architectures and Triplet networks are investigated using signals obtained from a wearable sensor. The proposed learning models outperform the traditional deep learning networks, while Siamese architectures also demonstrate generalizability by classifying unseen classes of falls and falls from different sensing modalities.
Acknowledgements

First and foremost I am extremely grateful to my supervisor, Dr. Sreeraman Rajan, for his awesome advice, continuous support, and patience during the course of my masters program. His plentiful experience have encouraged me through out the time of my research and daily life. I thank the examination committee members for their time and efforts to evaluate my work and provide valuable feedback. I am extremely grateful to Dr. Jila Hosseinkhani for meticulously proof-reading my thesis and providing appropriate feedback. I would like to thank the Department of Systems and Computer Engineering, Carleton University, for providing me the opportunity and on-demand support for the completion of my program. I also thank the Natural Science and Engineering Council of Canada (NSERC) for partially funding the work. I also acknowledge the financial support that I got for a term from Department of National Defence’s (DND)- IDEAS- Micronet initiative. These financial supports have made my study and life in Canada a smooth experience through out the ups and downs of this pandemic-ridden world. Finally, I would like to express my gratitude to my parents. Without their financial support, tremendous understanding and encouragement in the past few months, it would be impossible for me to complete my program.
Contents

1 Thesis at a Glance
   1.1 Introduction .............................................. 1
   1.2 Motivation .................................................. 2
   1.3 A Brief Overview of the Proposed Approach ............... 3
       1.3.1 Thesis Contributions ............................... 5
   1.4 Thesis Structure ........................................... 6
   1.5 Summary .................................................... 7

2 Literature Survey on Fall Detection .......................... 8
   2.1 Overview ..................................................... 8
       2.1.1 Fall Detection Strategies ........................... 9
       2.1.2 Traditional Methods for Fall Detection .......... 10
       2.1.3 Machine Learning Methods for Fall Detection .... 13
   2.2 Summary .................................................... 24

3 Proposed Architectures ......................................... 25
   3.1 Loss Functions for Siamese Network ....................... 26
       3.1.1 Contrastive Loss Function ......................... 27
       3.1.2 Binary Cross-entropy Loss ......................... 28
4.6.1 Performance Benchmark on the Face Detection Dataset
(Siamese Inception-Densenet) .......................... 60

4.7 Interclass Classification ............................. 61

4.7.1 Dissimilarity Scores (Class assignment) ............. 62

4.8 Cross Dataset Evaluation ............................. 63

4.9 Few-shot Comparator: Siamese Convolutional Auto-encoder . . 64

4.10 Transfer Learning Comparators ......................... 67

4.11 Original Densenet and Inception ....................... 68

4.12 Triplet Network ....................................... 69

4.13 Parameters ........................................... 72

4.14 Summary ............................................. 73

5 Conclusions and Directions for Future Work 75

5.1 Contributions .......................................... 76

5.2 Limitations of the Current Work ....................... 79

5.3 Future Directions .................................... 80

6 Appendix 83

6.1 Metrics .............................................. 83

6.1.1 Siamese Inception-Densenet ....................... 84

6.1.2 Siamese 1 × 1 .................................. 85

6.1.3 Triplet Network .................................. 85

6.2 Details of the ORL-DB Database: Face-recognition Dataset ....... 86

References 87
## List of Figures

2.1 Phases of a fall event. .................................................. 11

3.1 Structure of Siamese network. ................................. 26

3.2 Demonstration of parameter reduction. ...................... 30

3.3 Proposed architecture. ........................................... 32

3.4 Sample illustration of an Inception module. .................. 34

3.5 Proposed Inception module. ...................................... 35

3.6 Proposed Densenet module. ...................................... 36

3.7 Triplet Network. ................................................... 38

4.1 Spectrograms from the dataset: Non-fall (Left) and Fall (Right). . . . . . 45

4.2 Histogram of F-score (in %) of 15-shot CE (Left) and 1-shot CE (Right) (Similarity testing) (Siamese $1 \times 1$) .................................................. 52

4.3 Histogram of F-score (in %) of 15-shot CLF (Left) and 1-shot CLF (Right) (Similarity testing) (Siamese $1 \times 1$). ................................. 53

4.4 Histogram of F-score (in %) of 15-shot CLF (Left) and 1-shot CLF (Right) (Classification) (Siamese $1 \times 1$). ................................. 54

4.5 Histogram of F-score (in %) of 15-shot CE (Left) and 1-shot CE (Right) (Classification) (Siamese $1 \times 1$). ................................. 55
4.6 Visualization of dissimilarity scores of the first sample 7 similar dissimilar pairs (Siamese $1 \times 1$) (Left) and (Siamese Inception-Densenet) (Right). ......................................................... 56
4.7 Siamese $1 \times 1$ Dissimilarity scores of non-fall/fall - fall/non-fall (anchors). 57
4.8 Siamese Inception-Densenet Dissimilarity scores of non-fall/fall - fall/non-fall (anchors). ......................................................... 57
4.9 Histogram of F-score (in %) Siamese-Inception-Densenet (Similarity). 60
4.10 Histogram of F-score (in %) Siamese-Inception-Densenet (Classification). ......................................................... 60
4.11 Classifying Lateral falls. ......................................................... 63
4.12 Classifying Forward falls. ......................................................... 63
4.13 Classifying Backward falls. ......................................................... 63
4.14 General overview of a Convolutional auto-encoder. ................. 66
4.15 Detailed representation of Convolutional auto-encoder architecture. 67
4.16 Distribution of F-score (in %) Triplet Nets $1 \times 1$ (Classification). 70
4.17 t-SNE evaluation data embeddings before (Left) and after training (Right). ......................................................... 71
4.18 Sigmoid values for sample evaluation-set (15-shot) (Left) and (1-shot) (Right). ......................................................... 72
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Types of ADLs in the SisFall dataset.</td>
<td>42</td>
</tr>
<tr>
<td>4.2</td>
<td>Types of falls in the SisFall dataset.</td>
<td>43</td>
</tr>
<tr>
<td>4.3</td>
<td>F-score (in %) statistics of CE (Similarity) (Siamese $1 \times 1$).</td>
<td>52</td>
</tr>
<tr>
<td>4.4</td>
<td>F-score (in %) statistics of CLF (Similarity) (Siamese $1 \times 1$).</td>
<td>52</td>
</tr>
<tr>
<td>4.5</td>
<td>F-score (in %) statistics of CLF (Classification) (Siamese $1 \times 1$).</td>
<td>55</td>
</tr>
<tr>
<td>4.6</td>
<td>F-score (in %) of CE (Classification) (Siamese $1 \times 1$).</td>
<td>55</td>
</tr>
<tr>
<td>4.7</td>
<td>F-score (in %) statistics on the ORL-Database.</td>
<td>58</td>
</tr>
<tr>
<td>4.8</td>
<td>F-score (in %) statistics of Siamese-Inception-Densenet (Similarity).</td>
<td>59</td>
</tr>
<tr>
<td>4.9</td>
<td>F-score (in %) statistics of Siamese-Inception-Densenet (Classification).</td>
<td>59</td>
</tr>
<tr>
<td>4.10</td>
<td>F-score (in %) statistics on the ORL-DB.</td>
<td>61</td>
</tr>
<tr>
<td>4.11</td>
<td>F-Score in % of Interclass classification of falls (Siamese $1 \times 1$).</td>
<td>62</td>
</tr>
<tr>
<td>4.12</td>
<td>F-Score in % of Interclass classification of falls (Siamese Inception-Densenet).</td>
<td>62</td>
</tr>
<tr>
<td>4.13</td>
<td>F-score (in %) statistics on Mobifall (Classification).</td>
<td>64</td>
</tr>
<tr>
<td>4.14</td>
<td>F-score (in %) statistics of SCAE (Classification).</td>
<td>66</td>
</tr>
<tr>
<td>4.15</td>
<td>F-score (in %) statistics of SCAE (Classification) - Mobifall.</td>
<td>66</td>
</tr>
<tr>
<td>4.16</td>
<td>F-score (in %) statistics of transfer learning architectures.</td>
<td>68</td>
</tr>
<tr>
<td>4.17</td>
<td>F-score (in %) statistics of Original Densenet and Inception architec-</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>tures trained with Full-shot and 1-shot.</td>
<td></td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>4.18</td>
<td>F-score (in %) statistics of Triplet network.</td>
<td>70</td>
</tr>
<tr>
<td>4.19</td>
<td>Trainable parameters based on the spectrograms of size $65 \times 48$.</td>
<td>72</td>
</tr>
<tr>
<td>6.1</td>
<td>Format of Confusion Matrix.</td>
<td>83</td>
</tr>
<tr>
<td>6.2</td>
<td>Mean Precision and Recall in (%) for 1-shot and 15-shot Siamese Inception-Densenets.</td>
<td>84</td>
</tr>
<tr>
<td>6.3</td>
<td>Sum Confusion Matrix for 15-shot Siamese Inception-Densenet.</td>
<td>84</td>
</tr>
<tr>
<td>6.4</td>
<td>Sum Confusion Matrix for 1-shot Siamese Inception-Densenet.</td>
<td>84</td>
</tr>
<tr>
<td>6.5</td>
<td>Sum Confusion Matrix for 15-shot Siamese Inception-Densenet Mobifall (20 trials).</td>
<td>84</td>
</tr>
<tr>
<td>6.6</td>
<td>Mean Precision and Recall in (%) for 15-shot Siamese Inception-Densenet Mobifall Test.</td>
<td>85</td>
</tr>
<tr>
<td>6.7</td>
<td>Sum Confusion Matrix for 15-shot Siamese 1 x 1 Mobifall (20 trials).</td>
<td>85</td>
</tr>
<tr>
<td>6.8</td>
<td>Mean Precision and Recall in (%) for 15-shot Siamese 1 x 1 MobiFall Test.</td>
<td>85</td>
</tr>
<tr>
<td>6.9</td>
<td>Mean Precision and Recall in (%) for 1-shot and 15-shot Triplet Network.</td>
<td>85</td>
</tr>
<tr>
<td>6.10</td>
<td>Sum Confusion Matrix for 15-shot Triplet Network.</td>
<td>85</td>
</tr>
<tr>
<td>6.11</td>
<td>Sum Confusion Matrix for 1-shot Triplet Network.</td>
<td>86</td>
</tr>
</tbody>
</table>
**List of Abbreviations**

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>GANs</td>
<td>Generative Adversarial Networks</td>
</tr>
<tr>
<td>SMOTE</td>
<td>Synthetic Minority Oversampling Technique</td>
</tr>
<tr>
<td>WHO</td>
<td>World Health Organization</td>
</tr>
<tr>
<td>IOT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>HRV</td>
<td>Heart Rate Variability</td>
</tr>
<tr>
<td>ECG</td>
<td>Electrocardiogram</td>
</tr>
<tr>
<td>SPO2</td>
<td>Pulse Oximetry</td>
</tr>
<tr>
<td>MEMS</td>
<td>Micro Electro Mechanical Systems</td>
</tr>
<tr>
<td>MLP</td>
<td>Multi Layered Perceptron</td>
</tr>
<tr>
<td>EMG</td>
<td>Electro Myo Graph</td>
</tr>
<tr>
<td>LDA</td>
<td>Linear Discriminant Analysis</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network</td>
</tr>
<tr>
<td>SNN</td>
<td>Siamese Neural Network</td>
</tr>
<tr>
<td>LSTM</td>
<td>Long-Short Term Memory Network</td>
</tr>
<tr>
<td>SVD</td>
<td>Singular Value Decomposition</td>
</tr>
<tr>
<td>STFT</td>
<td>Short Time Fourier Transform</td>
</tr>
<tr>
<td>TL</td>
<td>Transfer Learning</td>
</tr>
<tr>
<td>ADL</td>
<td>Activities of Daily Living</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>SSFHD</td>
<td>Single Shot Human Fall Detector</td>
</tr>
<tr>
<td>VGG</td>
<td>Video Geometry Group</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean Squared Error</td>
</tr>
<tr>
<td>BCE</td>
<td>Binary Cross-Entropy</td>
</tr>
<tr>
<td>CLF</td>
<td>Contrastive Loss Function</td>
</tr>
<tr>
<td>CE</td>
<td>Cross Entropy</td>
</tr>
<tr>
<td>MFCC</td>
<td>Mel Frequency Cepstral Coefficients</td>
</tr>
<tr>
<td>ReLU</td>
<td>Rectified Linear Unit</td>
</tr>
<tr>
<td>FC</td>
<td>Fully Connected</td>
</tr>
<tr>
<td>STD</td>
<td>Standard Deviation</td>
</tr>
<tr>
<td>IQR</td>
<td>Inter-Quartile Range</td>
</tr>
<tr>
<td>SCAE</td>
<td>Siamese Convolutional Auto-encoder</td>
</tr>
<tr>
<td>FWF</td>
<td>Forward Fall</td>
</tr>
<tr>
<td>BWF</td>
<td>Backward Fall</td>
</tr>
<tr>
<td>LAT</td>
<td>Lateral Fall</td>
</tr>
<tr>
<td>t-SNE</td>
<td>t-Distributed Stochastic Neighbor Embedding</td>
</tr>
</tbody>
</table>
Chapter 1

Thesis at a Glance

1.1 Introduction

Humans have the inherent ability to learn from examples. The human brain is capable of classifying objects with just a single sample [1]. In other words, humans have a robust generalization capability. Machine learning and deep learning have shown the ability to learn and generalize when a lot of data is available. Unlike humans, these algorithms cannot learn from a single example of a class. In most cases, the algorithm over-fits due to limited data. Over-fitting typically means the algorithm has memorized the training data perfectly well, however, it is unable to infer on the unseen test data set [2].

The problem of acquiring an immense amount of data in the real world is difficult, if not impossible, in most detection and classification problems. Furthermore, it may not be feasible to have an equal number of examples of all classes present in the training dataset. Class imbalance is a significant problem when designing learning systems with real-life practical datasets. Often it is not possible to acquire examples of all the classes. For instance, the field of fall detection of elderly suffers from class
imbalance. Fall detection system designers may have to deal with an imbalanced dataset that hardly has any elderly fall in the extreme case. An imbalanced class typically means that one of the class types in the training set is represented more than the other classes. Ideally, the goal is to have equal number of examples or exemplars for each class that means there is an equal class distribution across data samples. In general, slight class-imbalances are typically not an issue, as long as the algorithm can generalize well on the test data. Although, a significant class imbalance requires to be properly addressed.

To overcome class imbalance issue, several methods such as SMOTE (Synthetic Minority Oversampling Technique) [3] exist. In [4], Generative Adversarial Networks (GANS) were utilized to generate synthetic data for training a neural network. Unfortunately, data obtained through such techniques may not precisely simulate real-world conditions. Realizing the demand for learning with limited data, a new area in deep learning has emerged that provides the ability to learn from a few examples (N-Shot Learning), that is, training using few examples. The main focus of this thesis is on learning from few examples to detect fall events. However, this should not be confused with the other emerging areas, such as zero-shot learning, where the model does not use the target class information during training [5].

1.2 Motivation

The primary motivation for the current thesis is to learn feature representations to classify fall detection from limited data. Falls are relatively rare events, and therefore, acquiring data related to falls is challenging. Currently, available datasets mostly contain falls obtained in simulated conditions and consequently cannot resemble real-life situations. This is particularly true for falls of older people. It may be easier
to acquire fall data from the children and youths compared to the elderly subjects. Because it is not feasible to ask elderly subjects to fall due to safety reasons, even in a controlled environment, falls may tend to be fatal; hence, the main issue is to contend with imbalanced data and in the extreme case the lack of elderly fall examples in the dataset. Most of the datasets have a class imbalance, even without fall examples from elderly [5]. The issue of limited data and class imbalance becomes complicated when types of falls need to be identified. Furthermore, as most of the datasets contain only simulated falls, a fall detection system trained using such a dataset may fail to detect falls in real-life scenarios.

Siamese network can provide a potential solution when the training dataset is limited and holds class imbalance. In this thesis, fall detection from signals collected through wearable devices is studied. Based on our conducted research, there is no other existing study in the literature that investigates fall detection using Siamese architecture.

1.3 A Brief Overview of the Proposed Approach

In this section, a brief overview of the two proposed few-shot learning approaches has been provided. The first type of proposed learning architecture is the Siamese network and the second type is the triplet network.

Siamese networks learn to distinguish between similar and dissimilar examples. Siamese networks always use a pair of inputs for learning purposes. This learning procedure is different from predictive learning, where a network learns to identify the classes amongst the examples. The network consists of two identical architectures that can take any neural network architecture; hence, the network has two inputs. The network learns to classify the two inputs based on their similarity/dissimilarity, that
is, the network outputs 0 if similar, and outputs 1 if dissimilar. The trained network can be applied to test inputs. One of the inputs is assumed as an anchor (reference), while the other input is used to present the test input. The class association of the anchor input is assumed to be known. The network outputs a dissimilarity score. The test input is compared to a set of anchors and the lowest dissimilarity score is determined. The class of the anchor that obtained the lowest dissimilarity score is then declared as the class of the test input.

We also extend the idea of the Siamese network, by considering another network, for fall detection using few-shot training examples, known as the triplet network. Triplet networks are trained using three identical networks namely, an anchor network, a positive network, and a negative network. Their corresponding three inputs are the anchor, the positive (which belongs to the same class of the anchor), and the negative (any different class than the anchor class). Thus, network architecture consists of three branches (one input for each branch), accommodating any chosen neural network architecture in each of these branches. These branches compute the embeddings (low-dimensional vectorial representation) of the inputs. The loss function in triplet networks ensures that the distance between the embeddings of the anchor and the positive inputs are lower compared to the anchor and the negative inputs. Once the network is successfully optimized, these branches will generate embeddings that are closer to each other for positive pairs and are farther from the embeddings generated for negative pairs. To test the network, an unknown sample is presented to the network paired with an anchor in which the network outputs two embeddings which may be similar if the unknown belongs class of the anchor’s class, otherwise, the embeddings are farther apart. Then, a softmax function can be applied to provide a probability of class association as an output for both of these pairs. Next, a class declaration can then be made by selecting the resulted class that provides the
maximum probability value. Alternatively, by using a sigmoid activation function, the class of the test example can be directly determined - with the criteria of being closest to the anchor.

Ideally, the choice of a loss function plays a significant role in the network training stage because it helps learn the features. In this work, two existing loss functions are used for learning, and the network’s performances corresponding to the loss functions are compared. The core idea is to learn good representations using a supervised approach with Siamese networks using a limited number of examples. The triplet loss function has been used for triplet networks to learn feature representations.

1.3.1 Thesis Contributions

- This thesis is the first work in the related research literature to demonstrate fall detection with Siamese network using signals acquired through wearable inertial measurement unit.

- A parameter efficient $1 \times 1$ few-shot learning Siamese architecture is proposed for classifying falls using simply a few examples.

- The $1 \times 1$ architecture has been adopted in the triplet networks, which is based on the triplet loss function for comparison purposes. This is the first time in the literature that the triplet loss function has been used for fall detection.

- A deeper network is proposed which is composed of a modified Inception-Densenet architecture in the Siamese branches for the fall detection task using 1-Shot and 15-shot scenarios.

- Triplet networks using the $1 \times 1$ architecture have been proposed and implemented using 15 and 1-shot scenarios.
The accomplished work in this thesis has resulted in the following publications


### 1.4 Thesis Structure

- **Chapter 1** motivates the need for the investigation performed in this thesis, lays the foundation and provides a brief overview of the work and the contributions that are outcomes of this study.

- **Chapter 2** contains a review of the current state-of-the-art in fall detection using wearable devices and provides an overview of machine learning methods related to fall detection. Then, it discusses the issues of classical training regimes and introduces the concepts of few-shot learning from limited data.

- **Chapter 3** explains the Siamese network architectures in-depth with the associated loss functions.

- **Chapter 4** performs an analysis of the results with certain key-points illustrated.

- **Chapter 5** concludes the thesis with a discussion and future research directions of the proposed work.
1.5 Summary

In this chapter, the motivation for the work was explained. Moreover, a brief overview of the proposed work accomplished through this thesis was elucidated. The contributions and the scholarly outcomes of this research were also summarized. The chapter also presented the thesis structure.
Chapter 2

Literature Survey on Fall Detection

2.1 Overview

The World Health Organization (WHO) has recently reported that by 2050, the number of people over 60 years would double that of 2019, outnumbering the younger population aged between 15 and 24 years [6]. Currently, 8.5 percent of the world population is 65 years old or older [6]. With this accelerated growth in the aging population, every nation should be concerned about the healthy aging of its seniors. Seniors of the current decade, at least in the developed nations, prefer independence and, therefore, stay in their homes as they age. Although researches have shown a positive impact on the seniors’ quality of life when they age at the comforts of their home [7], the safety and well-being of aging seniors have become a growing concern. Falling is one of the greatest dangers when seniors live alone. Falls happen to be the driving cause of deaths and hospitalizations for the elderly, and this has led to a significant cost to healthcare systems. The number of injuries caused by falls is expected to double by 2030 due to the aging population [17] if no preventive measures are undertaken[7]. This has led to a new area of research that concerns itself with
the monitoring of activities of the elderly population. With the emergence of the internet of things (IoT), smart non-invasive or minimally invasive wearable devices have gained popularity as activity monitors. Also, concerns of debilitating conditions like dementia and Alzheimer’s diseases have led to the emergence of remote activity monitoring techniques based on active and passive sensors.

The rest of this section discusses existing methods, issues, and challenges in fall detection literature, and then proceeds on to the preprocessing of the signals.

### 2.1.1 Fall Detection Strategies

There are many methods to detect fall events. From a larger perspective, fall detection systems can be classified as environmentally smart systems and wearable devices. Environmental systems include infrared sensors, radars, video cameras, microphones, or pressure sensors. Wearable systems include body-worn sensors such as accelerometers, gyroscopes, and goniometers. Vision-based systems can be considered as environmental sensors, and they can provide information about the environment as well as activities. The information obtained through the environment can be used as feedback to construct fall detection systems [8]. For instance, cameras can be used as part of either wearable sensors or as part of home monitoring. It may be also used to monitor activities inside the house. However, cameras have a field of view constraints, and performance issues based on lighting conditions. Importantly, cameras violate privacy. In addition, radars are another class of environmental sensors that can be used for activity monitoring. Many of the constraints and disadvantages in camera-based sensing systems can be mitigated by using radar-based systems. Such radar-based systems are currently not wearable.

Sensors used for fall detection are generally installed in a wearable device worn by the subject, such as a wristband or a watch. Multiple parameters such as heart rate
variability (HRV), the variation in R-R interval of the QRS complex in the electrocardiogram, electrocardiogram (ECG), pulse oximetry (SPO2), and accelerometer-measured kinematic features can be acquired and tracked by these sensors. In addition, appropriate data from the wearable sensors may be fed as inputs to a threshold-based system, or a set of features may be derived and fed to a machine learning-based system to identify and detect falls. Wearable sensor-based devices are generally inexpensive and consume less power, which eliminates the overhead of prolonged-charging. Moreover, wearable sensors generally have a small form factor as in the form of a band that can be worn around the wrist or thigh, making it less susceptible to isolation from subjects. A description of the contrast between four wearable wristband style devices currently available on the market in terms of their features and cost can be found in [9]. The power consumption in wearable devices depends on the design of the device, the type of sensors, and the technology used for communication. There are also systems where the sensors are worn or embedded within a device like a smartphone. Smartphone-based systems require the subjects to charge their devices as needed and carry it with them to enable falls and other activities to be detected. This situation requires compliance from the subjects (users) as the subject should agree and remember to carry the phone everywhere they go. This compliance cannot be achieved with senior citizens, in general, as many of them may already suffer from dementia.

2.1.2 Traditional Methods for Fall Detection

Most of the researches in fall detection use wearable sensors, mainly triaxial accelerometers and gyroscopes [10]. In this thesis, fall detection using triaxial accelerometers and gyroscopes is considered. In order to understand the fall detection algorithms, it may be necessary to understand the fall event.
Fall events can be considered to be composed of three main phases.

- **Pre-fall Phase**: It is the phase before a fall happens. This phase contains the activity that happens before a fall.

- **Impact Phase**: It is the phase when fall really happens or it is the moment of falling.

- **Post-fall Phase**: It is the phase after the fall has actually happened. It contains the activity that continues after a fall.

![Figure 2.1: Phases of a fall event.](image)

Figure 2.1 shows a time series signal of a person falling down. The pre-fall phase constitutes the person sitting on a chair, following which the person falls forward, which is shown in the impact phase. The impact phase constitutes the moment of a fall and hitting the ground. The moment of a fall has the highest acceleration value. The person then lies on the ground, which constitutes the post-fall phase. From this figure, it is evident that different phases of the fall contain different features or characteristics.
2.1 Overview

One way to detect a fall would be to observe if any feature of the fall signal exceeds a given threshold. The features of fall used in threshold-based methods include fall velocity, fall impact, and orientation [11]. The work by Bagala et al. [12] presented the performance of various threshold-based techniques and observed that most of the models are tuned using simulated falls and cannot generalize on real-world falls. Such algorithms generally have better classification rates, and also tend to have high false-positive rates when experimented on real data. Most of the errors were evident as these algorithms were simply classifying events that exceed a maximum value. Many real-world activities, such as jumping, can resemble a fall and can exceed the set thresholds due to increased acceleration. In [13], an algorithm was developed based on the first differences and the first derivatives of the sum of accelerometer readings along the three axes. The proposed algorithm was real-time and could identify sudden movements. Wu et al. [14] proposed a system that used a tri-axial accelerometer and a gyroscope to obtain the sum of acceleration along with rotational information of the subject. The combined use of threshold-based acceleration and rotation was used to conclude whether a fall has occurred.

Peak acceleration and angular velocity in [15] were captured by a bi-axial gyroscopic sensor and trunk-mounted accelerometer. A multistage threshold-based algorithm was used to identify falls. The resultant angular velocity, angular acceleration, and angle of the trunk had to cross a certain threshold to trigger a final alarm concluding a fall event.

Chen et al. [13] used MEMS-based accelerometers to detect a fall. The study showed that setting thresholds individually for the three axes did not operate well in the experiments. Hence, the norm of the three axes was computed, and a threshold was set for the norm. The authors noted that the design may be customized to improve performance because the acceleration profiles vary from person to person.
based on their physique.

A threshold-based detection is generally a heuristic approach, often based on trial and error. Although automatic thresholding is possible, such methods are non-optimal. Machine learning is an alternate approach that can be used for fall detection. The following subsection describes the machine learning approach for fall detection.

Sensor fusion techniques have also been examined where multiple sensor information is used for inference; for example, the work reported in [11] combined the use of accelerometer and ECG sensors. The data from the accelerometer was used to identify abnormal changes in movements, and the signals from the ECG sensor were used to measure abnormal variations in heart rates induced by anxiety during a fall. The combined detection was threshold-based, and a fall was inferred if both the methods concluded the occurrence of a fall. Recently, proposed NATLIFE FRAMEWORK [16] utilizes cameras, WiFi repeaters, IMU Sensors to track the movement of the subject apart from detecting falls. This union between networking technologies with inertial-based sensors ensures a truly assisted environment for the detection of falls.

### 2.1.3 Machine Learning Methods for Fall Detection

Machine learning has shown to significantly improve performance. Instead of setting parameters for classification manually, a model is trained on a training set. The training set is assumed to contain all representations of the signals of interest, which in the case of this thesis includes all types of fall signals. The model automatically chooses the best-weighted feature combinations and outputs a classification or rating. Specific issues such as selecting the appropriate feature to use, deciding on the correct model type, and having access to an abundance of fall data for training still exist and are being researched. It is important to note that appropriate features must be selected to detect and classify. Features need not be highly predictive individually;
a collection of weak features may be combined to form a powerful composite feature for classification through the learning process. Features such as mean, median, standard deviation, and Inter-Quartile-Ranges may be easily derived manually and used for classification tasks. Instead of manually selecting good features for learning, methods like logistic regression can be used to choose the informative features automatically. Logistic Regression can rate features by assigning non-zero coefficients to useful features and zero for the rest.

The number of parameters in the model is dictated by the number of features. As the number of features increases, the number of parameters in the model should be increased. Consequently, the quantity of training data should also be increased. If the model does not have enough parameters, then it cannot capture all the relationships between the features. This leads to over-generalization or over-fitting, while if there are more parameters in the model than the number of features, then the model will capture unwanted information (also model noise), which leads to under-fitting. During model design, one should ensure that there is enough data available to avoid both over-fitting and under-fitting. Therefore, the choice of model, number of features, and number of data samples will play a role in the design and performance of a machine learning system.

Many traditional machine learning models have been used in fall detection. A review of such traditional machine learning models is given in [8]. Notably, support vector machines, logistic regression, K-nearest neighbour, decision trees, and Multi-Layered Perceptron (MLP) [17] have been used in fall detection by several researchers. In addition, researchers studied gait, balance, postures, and activities that could influence falls. Various types of sensing modalities have been used in fall detection. However, in this work, only body-worn sensors or multi-modal approaches with body-worn sensors are considered. Literature survey is restricted to only body-worn sensors.
or multi-modal sensors, body-worn sensors were also considered for fall detection. In the state-of-the-art for fall detection, multi-modal approaches using various sensors such as videos, microphones, or physiological sensors are used in addition to inertial measurement unit (IMU) sensors.

The work presented in [18] analyzed and compared machine learning and statistical techniques to develop a physiological monitoring system that recorded acceleration and EMG signals to perform analysis on signals during standing. The goal of the system was to assess the behavior of the EMG signals on posture control in terms of balance.

A computer-vision-based approach has been applied to fall detection in conjunction with machine learning. Computer vision part was used to track a subject and identify posture through a two-step process in [19]. In the first step, a silhouette was extracted using the statistical and knowledge-based object detector based on background subtraction. In the second step, a monitoring module was employed to assist with the occlusion after the correct target has been determined. The Naive Bayesian classifier that used the tracking information and the silhouette-associated histogram for classification was developed. The vision-based system produced an alarm if the subject was lying down for a long time.

Another silhouette-based approach was presented in [20], where a Gaussian mixture model was used to classify the deformation of the human body. In this work, a Canny edge detector was used for detection of the edges before utilizing a background subtraction method for feature extraction.

In [21], data was collected using three-dimensional acceleration along with videos. Markers were used on the following anatomical parts: the ankles, waist sternum, and top of the head to track the motion. The mean acceleration and variance of these markers with respect to the initial marker position within video frames were
calculated. The LDA-based classification was performed and compared based on the various marker positions. Classification accuracy resulted in ranging from 54\% to 98\% depending on the number of markers and their location, and the type of classifier.

Acoustic-based approaches were also attempted for fall detection. In [22], sound and floor vibration sensing were used to detect a fall. A tri-axis accelerometer was employed to track floor movements, and a microphone was used to capture the noise caused by fall. A Bayes classification algorithm was then used to determine a fall versus other events.

The current trend in research has moved away from traditional machine learning methods into deep learning-based methods. In these modern methods of fall detection, the sensor data is considered either as a time-series or as an image. The sensor data can first be transformed into images and then used for detection/classification. For instance, Long-Short term memory networks (LSTM) [23] uses the sensor data as a time series while in [24], Convolutional Neural Networks (CNN) uses images as inputs for fall detection.

A multi-sensor-based approach to detect falls’ direction was proposed in [25]. The detection problem was treated as a multi-class classification problem using a CNN ensemble. The model was able to perform with an accuracy of 99.3\%. The tuning of a model and its impact on performance in fall detection has been shown in [26]. In [26] the authors developed a machine learning pipeline for the detection of fall events. The model was able to achieve an F-score of 84.6\%.

1-dimensional CNNs have also found their application in human fall detection. Singular value decomposition (SVD) was applied on fall data obtained from UMAFall, SiSFall, and SHAR in [27]. Raw accelerometer signal and combination of various dimensionality reduction methods like SVD, kernel-based principal component methods were used as inputs for the 1-dimensional CNN model. The model performed with
an accuracy of 75.6% when combined with sum vector magnitude and singular value decomposition. The performance accuracy on the SiSFall and UMAfall datasets using raw signal along with SVD were 63.08% and 64.69%, respectively (based on the leave-one-subject-out cross-validation procedure). The best performance accuracy of 75.65% was achieved with the SHAR dataset using signal magnitude vector of 3-D accelerometer signals through SVD concatenation.

A system for fall detection using radars was proposed in [28]. Time-frequency analysis was used to obtain distinguishable features of human motion. A deep neural network using stacked auto-encoders with softmax was used to classify activities related to falls and non-falls. This classification was a multi-activity classification task. The model performed with an accuracy of 87%.

Variational auto-encoders with 3D-convolutional residual blocks and a region extraction technique were reported in [29]. The proposed system was an unsupervised learning approach that used reconstruction loss to detect falls and non-falls, and achieved an average accuracy of 88.7%.

Deep Convolutional auto-encoders in [30] used three layers of a combination of CNN and auto-encoders that used pre-trained weights to initialize the weights in the subsequent convolutional layers to distinguish 12 indoor activities, which included aided and unaided activities. The aided activities constituted activities performed with external equipment support such as crutches, cane, or wheelchairs. Unaided activities refer to activities that did not require any external equipment. The classification contained two types of falls, forward fall by tripping and lateral fall from a chair. The data was obtained from a Doppler-based radar system. Short-time Fourier transform (STFT) from the radar returns were obtained and spectrograms were computed. Spectrograms were used to train the model. The model provided a classification accuracy of 94.2%.
2.1 Overview

Transfer learning (TL) is another domain that has been well-researched in fall detection literature. Transfer learning has been successfully employed, where a model is trained on one fall detection dataset and tested on another. A TL model was successfully trained and tested on different fall detection datasets in [31]. However, TL models still require considerable amount of data for training during transfer learning, failing which models which transfer learn may overfit. As mentioned previously, the amount of available data to train the model is crucial. Several databases using different sensor types are available for developing human activity detection levels (ADL) and a fall detection system. A summary of the composition of such databases is given in Table 6 in [5]. Although the datasets may be different because of the manner in which they were collected or the type of sensors, it is seen that there is a significant amount of data to classify a fall activity from a non-fall event (binary classification). When the classification is expanded to the types of falls, the number of examples of each type of fall in most of these datasets decreases. Furthermore, the dataset is unbalanced even for the binary classification case. Thus, there is more data related to non-fall activities than fall activities. A thorough study of various datasets used in fall detection is available in [5].

In a recent survey paper [32], various fall detection techniques were analyzed and categorized into two main groups: 1) techniques that derive a model from large datasets (bountiful datasets) with many human fall examples; 2) techniques that derive a model from datasets with limited fall examples (parsimonious datasets).

Many detection techniques including supervised machine learning methods, threshold-based techniques, and single class techniques need large datasets that must contain fall examples for training. Most of the datasets except a few contain more activities of daily living (ADL) than fall examples. Conventional supervised and unsupervised techniques trained with a small pool of fall data with a class imbalance
(due to the under-representation of fall class), will lead to biased results.

Learning methods using the limited dataset containing fewer fall examples than non-fall examples employ oversampling/undersampling techniques as part of data augmentation to increase the representation in the dataset. In the oversampling/undersampling technique, the class that is under represented is oversampled, while the class that is over represented is undersampled. Generally, learning techniques suffer when encountering a class imbalance [33]. Once, a model is trained using oversampled/under sampled data, it can classify a test sample as a human fall or ADL. One of the key drawbacks of this approach is that a good marker for differentiating an ADL from a fall is required. To obtain a good representation, a lot of data comprising of Fall and non-fall activities are required. Alternatively, one can just invoke a one-class classification. The work on fall detection by Diego [34] shows the use of a one-class Support Vector Machine (SVM) in the comparative study, which is a completely unbalanced SVM using only human falls.

Due to the differences in the sensing modality and sensor placement, there is a possibility that a model trained on one type of dataset may not generalize well when applied to another dataset. Furthermore, the effects of different locations for sensor placement have a direct impact on the performance of the model. For example, wearing the sensors on the waist and thigh provided more accurate fall detection compared to sensor placement on the shoulders. [35].

Another challenge seen in fall detection systems is the lack of high-quality data. Fall datasets are collected often in a highly controlled environment; almost all falls are simulated by professional actors or under the guidance of a specialist [36]. For example, there is usually a thick mattress when falling, which affects post-fall characteristics and therefore, the post-fall data. As a consequence, a genuine fall may not have been emulated. Using the learned models by such data may not guarantee generalization
to real-life scenarios. This is the current state of the quality of datasets. Only a few datasets cover different types of falls and span different age groups. Furthermore, learning models created in one monitoring environment may not work well as the environment changes, as shown in [5]. A comprehensive analysis of fall detection devices for people of different ages (excluding children) from a range of viewpoints, including data sources, and fall intervention methods are presented in [37]. In this review, the available research studies were classified into several categories based on the age of participants, methods of evaluation, and tools used in detection. Most of these studies were from synthetic measurements. It was observed that even though simulated data may contain some shared features with actual falls, the reliability was much higher for systems trained on real falls.

The fall detection research field suffers from class imbalance. To overcome the class imbalance, small-sized classes can be over-sampled, or the classes with a large number of samples can be under-sampled. Also, in the case of missing labels, parametric unsupervised learning approaches such as expectation maximization can be employed to label the data. Re-sampling is the most common strategy in terms of time series data. However, this research work was conducted with the real fall data with no resampling to avoid any types of biases due to unequal distribution of data in different classes introduced by the resampling process.

From the literature survey, it may be concluded that data for fall detection is not abundant. As a result, the deep learning methods that require a huge amount of training data may not be suitable under limited data conditions. Therefore a different approach such as a meta learning for designing a fall detection system may be in order. Meta learning helps achieve greater performance (accuracy, stability, etc) by effective combination of multiple classifiers/networks together.

Meta-learning approach can be broadly divided into three categories: metric-
based, model-based, and optimization-based approaches [38]. A detailed survey of meta-learning is available in [38]. Providing a detailed literature survey on meta-learning is out of scope for this work. In this work, a specific meta-learning approach, metric-based approach, is adopted, and a survey of metric-based approach for falls is provided in the following chapter. In a metric-based approach, a distance metric is used to learn the similarity or dissimilarity between training samples. More specifically, the focus of this thesis is to investigate the viability of the Siamese network in fall detection. Learning for fall detection has to be done with limited imbalanced training data. Few-shot learning is considered in this work, and a Siamese network is proposed as a model for fall detection.

2.1.3.1 Few-shot Learning

Since data imbalance cannot be avoided, and one cannot aim for a standard dataset like in the classification of images, an alternate way of training for fall detection is in order. Current research in fall detection employs predictive loss function for classification. A change in the paradigm of training using contrastive loss functions instead of the predictive loss function is utilized in this study. Siamese networks, first introduced by Bromley et al. [39], uses contrastive loss to learn the similarities and dissimilarities in the signature verification task.

The ability of the Siamese network to learn through contrastive loss function[40] was demonstrated in [41], where it was applied for character recognition by training on the Omni-glot dataset. The Omni-glot dataset comprises handwritten characters from 50 different alphabets. This is a standard dataset used by the machine learning community to test various classification tasks. The work in [42] used Siamese networks to compute similarity on the self-back dataset for activity monitoring using different pairs. Siamese network has found applications in signature verification,
image matching [43], natural language processing [44] for sentence regression tasks and for face verification [45]. The concept of Siamese network and few-shot learning as applied to fall detection has not gained enough attention, and only a few research works have appeared in the literature. Siamese auto-encoders and Siamese networks have been reported in [34], where few-shot learning was utilized to detect falls using sound from acoustic microphones. The capability of Siamese networks to learn useful features using a similarity function to recognize activities (using wearable sensors) was demonstrated using the SELF BACK dataset [42].

Another work, Single Shot Human Fall Detector (SSHFD), addresses the generalization issue in occluded or unseen environments [46]. This work addressed the issues of generalization in computer vision based systems due to unseen physical environments, such as background clutter. A pose-based human fall representation was used, which is invariant to these issues. A neural network model for 3D pose estimation and fall recognition is presented, which are resilient to missing joints due to occluded body parts.

Siamese network has been utilized for sound search in [47], where the network was used as a feature extractor. Siamese networks have also been used for object tracking using the Visual Geometry Group (VGG) networks, [48]. Siamese networks have also played an essential role in similarity detection [49].

The process of falling while doing any activity may be viewed as a change and therefore, change detection may be utilized for detecting falls. Sudden changes may also be due to abrupt changes in activities such as bending down while trying to sit or suddenly sitting from standing. Unfortunately, there is no indication in the literature that work has been done from this perspective. However, Siamese networks have been used for change detection. Recently, fully convolutional Siamese networks have been proposed for change detection [50]. Three different variants of fully connected
Siamese networks were studied and applied for detecting changes in Earth observation images. The technique presented in can be [50] adapted to fall detection.

A Fall may be viewed as an anomaly and detection of a fall event may be posed as an anomaly detection problem. Although the Siamese network has been used in anomaly detection, no work has been reported in treating fall as an anomaly event. Anomaly detection is studied extensively in networks under the name of intrusion detection. Traditional anomaly detection approaches are prediction-based. Anomaly detection has recently been studied in video surveillance using a few-shot approach in [51]. In [52], a meta-learning-based approach was considered, the model was trained to learn new tasks quickly with a limited amount of training samples. During testing, a few frames were used to adapt the system further, and with the newly adapted parameters, testing of rest of the frames was done. This was termed as a scene-adaptive meta-learning approach. If there were no anomalies, then the adaptation during the training phase will not change the parameters. Alternate techniques of anomaly detection that use few-shot paradigm by employing engineered features have also appeared in the literature. For instance, transfer learning was used to learn a model with a limited amount of training samples. Statistical features were extracted from the trained model, and a K-nearest neighbour (KNN) approach was utilized to detect nominal scenes from anomalous scenes. Few-shot learning (along with a transfer learning) was employed as a feature extraction module and the statistical KNN was used for decision making task. The few-shot learning module efficiently minimized the training complexity [51].
2.2 Summary

In this chapter, a literature survey was presented describing the existing state-of-the-art architectures relating to fall detection and the use of few-shot architectures in the context of fall detection. The significance and the requirements for few-shot learning have been well-established throughout the literature survey by citing all possible works relating to it.
Chapter 3

Proposed Architectures

Bromley and LeCun first developed a Siamese network in the early 1990s to solve signature verification as an image-matching problem [39]. A Siamese neural network consists of twin networks that accept separate inputs. The outputs of the Siamese network are used to evaluate an energy function. This function calculates pre-defined metrics between the representation of the highest-level feature at each output of the Siamese network. The twin network parameters are shared. A typical Siamese network is shown in Figure 3.1.

Ge (X1) and Se (X2) denote the two sister networks. These two sister networks are identical and have an equal number of parameters. The output of Ge (X1) and Se (X2) are encoded to an array of vectors. Then the distance is calculated between the two vector arrays to generate a score. This score is considered as “dissimilarity index”. This is the core idea of the Siamese network. A Sigmoid layer can be used at the end to generate probability scores from the dissimilarity score. Siamese sister branches can accommodate various types of deep neural networks; however, the original work by [39] on the Siamese network made use of relatively shallow networks.
3.1 Loss Functions for Siamese Network

Most loss functions evaluate the performance of a model across all the examples that form the training set. Regular neural networks have only one input and one output. The output of the network is learnt to either declare a class type or predict the next value that is expected in the output (regression). In order to learn, a loss function is calculated which is often predictive in nature and hence called predictive loss function. For classification related tasks, the most commonly used loss function is the cross-entropy loss function. The cross-entropy loss is also referred to as log loss. The cross-entropy loss is also called binary cross-entropy (BCE) loss for a binary classification task. The most common loss metric used for regression-related tasks, is the mean squared error (MSE) loss [53]. Most types of loss functions used in classification-related tasks are distance-based, where the loss is computed as a measure of L1/L2 distance in the embedding space. For example, embeddings (vectors) of different
classes generated by a neural network can be classified based on the distance between each other. Cosine similarity is another such loss function used to find how close an embedding is to another. The similarity between two embeddings can be computed using cosine similarity [54]. Cosine similarity has mainly found use in natural language processing-based tasks to discover links between sentence embeddings. However, in this thesis, two important loss functions have been used with Siamese Network.

The following two different types of loss functions has been considered for the proposed architecture:

1. Contrastive loss function (CLF);

2. Cross-Entropy loss function (CE).

3.1.1 Contrastive Loss Function

The outputs of the two branches of the Siamese network are used to calculate the contrastive loss. This loss function penalizes the model differently depending on whether the classes of the inputs presented to the network belong to the same class or not. The network parameters are adjusted so that the output vectors produced by the networks are almost “similar” for similar classes and “dissimilar” for dissimilar classes.

Let $X_1, X_2$ be a pair of two input samples and $y(X_1, X_2)$ be the label of that pair. When the samples, $X_1$ and $X_2$ are selected from the same class, then $y(X_1, X_2) = 1$ and if they are from a different class then $y(X_1, X_2) = 0$.

The Euclidean distance, $E_w$, between the outputs of the two branches is first determined as follows: $E_w = \|G_e(X_1) - S_e(X_2)\|$.

The $E_w$ results in a number which is referred to as the “score” of the network.
3.1 Loss Functions for Siamese Network

\[
\text{Loss} = (1 - y) \frac{1}{2} (E_w)^2 + (y) \frac{1}{2} \left\{ \max(0, m - E_w) \right\}^2. \tag{3.1}
\]

A margin parameter, \( m \) exists in the contrastive loss function, which determines the limit of how much the model penalizes dissimilar pairs. Often it may be considered as part of the hyperparameters. Choosing the correct value for \( m \) requires careful analysis. This may be regarded as one of the drawbacks of using the contrastive loss function. The contrastive loss function has been the primary focus in this thesis due to its immunity towards vanishing gradient problem [55].

A different function also exists for learning similarity in the inputs. Unlike the contrastive loss function, this function requires three inputs and is used in a network with three branches instead of two branches. Since there are three branches involved in the computation of the loss function, it is called the Triplet loss function. Triplet loss function was first used for face detection task [56]. Triplet loss function is explained in Subsection 3.3.3.

### 3.1.2 Binary Cross-entropy Loss

The binary cross-entropy loss is given by Equation 3.2.

\[
\text{Cross Entropy Loss} = -t_i \log(f(s_i)) - (1 - t_i) (1 - \log(f(s_i))), \tag{3.2}
\]

where \( t_i = \text{target} \) indicates the labels of the ground truth, \( s_i = \text{scores output from the network} \) and \( f(s_i) \) is the sigmoid activation function given by

\[
f(s_i) = \frac{1}{1 + e^{-s_i}}. \tag{3.3}
\]
3.2 Literature Survey of Siamese Network

The number of literature on Siamese network for fall detection is limited. An application of Siamese network for human fall detection based on audio signals of falling was reported in [34]. This work involved training the Siamese network using Mel Frequency Cepstral Coefficients (MFCC) spectrograms to classify falls. The model performed with an average F-score of 79%. In [57], Siamese Convolutional Autoencoder (SCAE) was used to perform the classification of falls. An unsupervised approach of learning features from audio signals in a few-shot setting was used in this work. The SCAEs performed with an average f-score of 93.58%. However, in both these works, a large dataset for training with 120000 samples was utilized. This does not effectively show the performance of Siamese-style architecture because Siamese is recommended for training using a small dataset. Siamese network for falls using features obtained from surface electromyogram signals (sEMG) was presented in [58]. All the work in Siamese network related to detecting fall using different signal modalities presented training with large amount of data. This thesis is the first work that investigates fall detection under limited training data situation.

3.3 Proposed Architectures

The goal of every model architecture must be geared towards two main things. Firstly, the model is supposed to require as little memory as possible during run time. Secondly, the model should not be “heavy”. The model should be “light” so that the storage requirements are not demanding. The latter facilitates the deployment of learning architectures on small-scale embedded systems such as the raspberry pi, orange pi, and Nvidia Jetson Nano. Generally, deeper models are harder to deploy on
3.3 Proposed Architectures

Embedded miniaturized systems due to their limited processing power.

In this thesis, three few-shot-based learning architectures are proposed. The first architecture is a Siamese network that contains the $1 \times 1$ layer. The second architecture is also a Siamese network that is based on two state-of-the-art deep neural network models, i.e., the Inception and the Densenet. The third is based on the Triplet network, which requires three inputs instead of two to compute the embeddings. The networks used in the Triplet network employed the same $1 \times 1$ model proposed in the first Siamese network.

3.3.1 Siamese 1x1 Network

This proposed network utilizes $1 \times 1$ layers to reduce the overall computational burden during the execution procedure. The $1 \times 1$ convolution reduces the overall number of trainable parameters in a network. This is illustrated in Figure 3.2.

![Figure 3.2: Demonstration of parameter reduction.](image)

The parameters in a neural network are the weights of the nonlinear activation functions. The number of parameters on the network depends on the input matrix (i.e., the number of input nodes within the network) and also the depth of the network.
3.3 Proposed Architectures

(i.e., the number of hidden nodes).

In Figure 3.2, “same” convolution of kernel size five is applied to a 256x256 matrix with 192 feature maps. The computations result in a large number of parameters. The kernel shown in the second row performs the same computation by using a 1 x 1 convolution of 32 feature maps or channels on the 256 x 256 matrix. This computation results in a fewer number of parameters. The number of parameters calculated is achieved by simply multiplying the dimensions of the $N \times M \times W$ with $F \times H \times W$ of the filters, where $N$ and $M$ are the height and width of the input matrix, and $F$ and $H$ are the height and width of the filter kernel. $W$ is referred to as the number of feature maps. Both of aforementioned operations are compared in terms of the number of parameters, using 1 x 1 convolution operation, and a computation efficiency of 96% is achieved. Therefore, the idea of 1 x 1 convolution is used in the proposed architecture. The idea of 1 x 1 convolution is simply scaling the values of the pixel one by one and then passing it through some nonlinear activation function as supplied. Therefore, the idea is that the filter only looks at a single pixel and not the area surrounding it.

The proposed architecture, as shown in Figure 3.3, comprises two sets of four 1 x 1 convolutional layers with a nonlinear rectified linear unit (ReLU) followed by a LeakyReLU activation function between them. The LeakyReLU activation function is defined as $\max(\alpha x, x)$, where $\alpha$ is initialized to 0.03. Then, batch normalization (BN) [59] was applied to the output of the ReLU unit preceding the Fully Connected Layer (FCLayer), to normalize the outputs of the ReLU activation function. The batch normalization can be mathematically expressed as follows:

$$Z = \left(\frac{x - u}{s}\right) \ast g + b,$$

(3.4)
where $u$ is the mean of a batch and $s$ denotes the standard deviation, while $g$ and $b$ are arbitrary learnable parameters. There is a ReLU non-linearity between the fully connected (FC) layers. The last FCLayer has an FC Embedding from an implementation perspective. The FC embedding contains the flattened embeddings generated by the two branches of the network. This minimalist network with just two layers of four $1 \times 1$ layers has a relatively smaller number of parameters (to be discussed in the next chapter).
3.3 Proposed Architectures

3.3.2 Siamese Inception-Densenet Network

The second proposed network is based on the premise that networks learn better as the number of layers increases. This premise was recently applied to Resnet architecture in [60]. On the other hand, a deeper network suffers from the effects of gradient vanishing. Moreover, the requirement for training data increases significantly. The effects of gradient vanishing are more pronounced when the sigmoid activation function is used; for instance, the derivative of a sigmoid may give values from 0 to 0.25.

To take advantage of the fast training of a deep neural network and to simultaneously learn with limited data, two state-of-the-art architectures, namely Inception and Densenet are combined in the proposed network. An Inception-based network keeps the model sparse, while Densenet addresses the issue of vanishing gradient. A lighter variant of these two networks has been combined and used as a network for each arm of the Siamese network.

The deeper architecture consists of a single Inception module followed by a modified Densenet module. The main idea of the Inception module is to use sparsely connected layers and reduce the number of trainable parameters to make the architecture lighter. Densenet module is used to make the model deeper, eliminate the issues of convergence due to vanishing gradient and enable the network to learn faster.

An Inception module is composed of the stacked channel-wise concatenation of parallel operations such as convolutions and max-pooling, as seen in [61]. Fig 3.4 shows a sample general illustration of an Inception module.

As shown in the Figure 3.5, the proposed Inception module is composed of $10 \times 10$, $5 \times 5$, and $2 \times 2$ convolution operations that connect to $1 \times 1$ bottleneck layers. The outputs of the bottleneck layers are concatenated to form the output of the Inception module. The output of the Inception module is passed through a transition layer,
which is comprised of a $50 \times 2 \times 2$ and a $10 \times 1 \times 1$ convolutional layer. This further reduces the number of feature maps generated by the concatenation operation of the Inception module to 10. The output from the transition layer is passed on to the Densenet-style architecture as shown in Fig 3.6.

The principle idea of Densenet originally was to eliminate issues of vanishing gradient and exploding gradient by introducing frequent skip connections between the layers [62]. In Densenet, every layer has access to information from the previous layer resulting in a better gradient flow (also referred to as deep implicit supervision) [63]. Each layer takes the output of the preceding layer as input and produces the output called feature maps. The number of interconnections in a Densenet is given by the relation: $N = \frac{L(L+1)}{2}$, where $L$ denotes the number of layers and $N$ denotes the number of connections. As the interconnections increase, the output of the Densenet becomes wider, that is the number of feature maps increase. To address this, a composite layer comprised of batch normalization and $k \times 1 \times 1$ convolution is used to fix the feature maps as $k$. In the machine learning literature, $k$ is also referred to as the growth rate [62]. Increasing $k$ will lead to longer training times and increased storage requirements.
The number of concatenations was selected such that the growth rate of the feature maps do not grow too large, unlike the original Densenet architecture where there was channel-wise concatenation after every layer. The concatenations in the proposed architecture are made such that the growth rate can never be too large, which makes the model lighter in terms of trainable parameters. Unlike the original Densenet architecture, in the proposed architecture, there are no reduction layers. The output of each layer is fixed as 10 feature maps \((k = 10)\). As opposed to the original Densenet, which has relatively larger kernel sizes, the proposed model has smaller filter kernels \((2 \times 2)\). Smaller kernels can capture adequate details in the input with fewer computations.

The ReLU activation function is used after each convolution layer. The output of
the Densenet is then fed to the final transition layer, which is comprised of flattening and fully connected (FC) layers. The final transition layer also contains $2 \times 1 \times 1$ layers, which ensures that the final output of the Densenet module generates only two feature maps, followed by an FC layer that outputs an embedding of 13824 dimensions.
3.3.3 Triplet Network

The Siamese triplet network is given in Figure 3.7. In this Figure, the terms $A$, $N$, and $P$ refer to the anchor, negative (of the anchor) and the positive (of the anchor) inputs, respectively. The anchor is assumed as a reference input.

\[ \|f(A) - f(P)\|^2 + \alpha \leq \|f(A) - f(N)\|^2. \]  

(3.5)

In Equation 3.5, $f(A)$, $f(P)$, $f(N)$ are the embeddings generated by the Network(A), Network(P), and Network(N) that take anchor, positive, and negative inputs, respectively. The positive and the negative inputs are centered around the anchor. For example, assuming the anchor is a fall, then the positive input could be another variant of the fall (therefore referred to as positive of the anchor). Then, the negative input would be a variant of the non-fall class (referred to as negative of the anchor). In the case that the input is from the same class as the anchor, it is referred to as positive of the anchor. The goal is to ensure that the embeddings of the anchor and the positive of the anchor are closer. This is achieved by minimizing their distance along with maximizing the distance between the anchor and the negative of the anchor by a margin, $\alpha$. The loss function, called the Triplet Loss can be written as Equation 3.6:

\[ L(A, P, N) = \max (\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha, 0). \]  

(3.6)

The pair generation task for the triplet network can be formalized in three main parts: the task of forming easy triplets, hard triplets, and semi-hard triplets. The definition of each of these pair generations is given below:

\[ d(A, N) > d(A, P) + \alpha \quad \text{(easy triplets)}, \]  

(3.7)
In the case of random sampling, the first condition is easily full-filled since there is a high chance that the positive of an anchor would be closer to the anchor compared to the negative one, and hence referred to as easy triplets. For hard triplets, the positive of the anchor is closer to the negative or vice versa, which is equivalent to stating that \( d(A, N) > d(A, P) \) is almost zero. Finally, a sigmoid activation function is used to directly classify if an unknown sample belongs to the class of the anchor. For
semi-hard triplets the negative of the anchor is closer to the anchor than the positive, but still has a positive loss.

3.4 Summary

In this chapter, two different types of few-shot learning networks, namely, the Siamese network and the Triplet network were discussed, and their associated loss functions were explained. The two proposed Siamese networks were the Siamese network that comprised of the $1 \times 1$ architecture and the deeper Inception-DenseNet based architecture. The Triplet network also used the same $1 \times 1$ architecture that was proposed for the Siamese network. Due to hardware and resource limitations, the proposed Inception-Densenet architecture was not implemented within the triplet network and is considered as future work.

In Chapter 4, the prepossessing of data, choice of the initial parameters, training, and validation of the proposed architectures will be discussed and presented.
Chapter 4

Results

4.1 Overview

This chapter discusses the training and testing methodologies adopted for validating the proposed architectures. All detailed descriptions of the types of the dataset are provided, including signal acquisition, sensor placement, and the participants. Also, preprocessing steps applied to the acquired signals in the dataset are presented. The performances of the proposed architectures are compared against known state-of-the-art networks such as the Inception, Densenet, and the stacked convolutional autoencoder (SCAE) architectures. In the last part of this chapter, inter-classification of falls for both proposed architectures are provided along with the cross-data evaluation on another well-known dataset, namely, MobiFall, without any form of retraining. The Triplet network’s performance is presented in a separate section as its performance evaluation needs a pair generation which is different than the Siamese architectures.
4.2 Dataset Description

Two datasets were considered in this thesis and are described in detail in the following subsections.

4.2.1 SiSFall

The main dataset used in this thesis is the SisFall dataset [64]. According to the survey in [5], the SisFall dataset is one of the few datasets that cover the majority of the age demographic while offering a variety of samples for fall and ADL activities. The SiSFall dataset covers both the younger (19-30 years age range) and the older population (65-75 years age range). The dataset contains 15 examples of fall types (with 5-trials of each type) and 20 examples of the non-fall type (with 5 trials of each type). The signals in the dataset were acquired at a sampling rate of 200 Hz. The dataset contains signals acquired from two accelerometers and one gyroscope that were worn on the waist by the subjects performing various activities including falls. The acceleration data from the inertial measurement unit (IMU) is used in this thesis.

4.2.1.1 Participants

The dataset was acquired with 38 volunteers of which 15 (8 males and 7 females) were older adults and 23 (11 males and 12 females) were young adults. Young adults performed ADLs and falls. Older adults did not perform some falls and ADLs due to recommendations by a physician specializing in sports medicine.

In this thesis, falls in the dataset are categorized into forward falls, backward falls, lateral falls based on the direction of the falls while the ADLs are categorized into standing, jogging, sitting, running, and walking based on the type of activity
4.2 Dataset Description

performed. For additional details regarding the type of ADLs and falls refer to Table 4.1 and Table 4.2. There are variants among each of the fall categories. For example, in the case of forward falls, the variants are “Falling while walking forward due to a slip”, “Falling forward while running”, “Falling forward from a chair”, and “Falling laterally from a chair”. Similarly, for lateral falls the variants include “Lateral fall while walking caused by a slip”, “Lateral fall when trying to get up”, and “Lateral fall while slipping caused by fainting or falling asleep”.

For additional details about the specification of the sensor used and the environment in which the activities are performed, readers may refer to [64].

<table>
<thead>
<tr>
<th>Table 4.1: Types of ADLs in the SisFall dataset.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walking slowly.</td>
</tr>
<tr>
<td>Walking quickly.</td>
</tr>
<tr>
<td>Jogging slowly.</td>
</tr>
<tr>
<td>Jogging quickly.</td>
</tr>
<tr>
<td>Walking upstairs and downstairs slowly.</td>
</tr>
<tr>
<td>Walking upstairs and downstairs quickly.</td>
</tr>
<tr>
<td>Slowly sit in a half height chair, wait a moment, and up slowly.</td>
</tr>
<tr>
<td>Quickly sit in a half height chair, wait a moment, and up quickly.</td>
</tr>
<tr>
<td>Slowly sit in a low height chair, wait a moment, and up slowly.</td>
</tr>
<tr>
<td>Quickly sit in a low height chair, wait a moment, and up quickly.</td>
</tr>
<tr>
<td>Sitting a moment, trying to get up, and collapse into a chair.</td>
</tr>
<tr>
<td>Sitting a moment, lying slowly, wait a moment, and sit again.</td>
</tr>
<tr>
<td>Sitting a moment, lying quickly, wait a moment, and sit again.</td>
</tr>
<tr>
<td>Being on one’s back change to lateral position, wait a moment, and change to one’s back.</td>
</tr>
<tr>
<td>Standing, slowly bending at knees, and getting up.</td>
</tr>
</tbody>
</table>
Table 4.2: Types of falls in the SisFall dataset.

<table>
<thead>
<tr>
<th>Fall Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall forward while walking caused by a slip.</td>
</tr>
<tr>
<td>Fall backward while walking caused by a slip.</td>
</tr>
<tr>
<td>Lateral fall while walking caused by a slip.</td>
</tr>
<tr>
<td>Fall forward while walking caused by a trip.</td>
</tr>
<tr>
<td>Fall forward while jogging caused by a trip.</td>
</tr>
<tr>
<td>Vertical fall while walking caused by fainting.</td>
</tr>
<tr>
<td>Fall while walking, with use of hands in a table to dampen fall, caused by fainting.</td>
</tr>
<tr>
<td>Fall forward when trying to get up.</td>
</tr>
<tr>
<td>Lateral fall when trying to get up.</td>
</tr>
<tr>
<td>Fall forward when trying to sit down.</td>
</tr>
<tr>
<td>Fall backward when trying to sit down.</td>
</tr>
<tr>
<td>Lateral fall when trying to sit down.</td>
</tr>
<tr>
<td>Fall forward while sitting, caused by fainting, or falling asleep.</td>
</tr>
<tr>
<td>Fall backward while sitting, caused by fainting, or falling asleep.</td>
</tr>
<tr>
<td>Lateral fall while sitting, caused by fainting, or falling asleep.</td>
</tr>
</tbody>
</table>

Note that vertical fall is not considered in this thesis.

4.2.2 MobiFall

The other dataset used in this work is the MobiFall dataset [65]. The MobiFall dataset has been utilized to validate the proposed Siamese architectures without retraining. The MobiFall dataset was collected using a Samsung Galaxy S3 device with the LSM330DLC inertial module (3D accelerometer and gyroscope). The gyroscope was
calibrated before the recordings. Orientation data is also available through a software-based orientation sensor [65]. The signals were acquired with a sampling rate of 87Hz for the accelerometer and 200Hz for the gyroscope [65].

4.2.2.1 Participants

The MobiFall dataset contained data from 11 volunteers (six males and five females). The age range of male volunteers were 22-32 years, while females were in the age range of 22-36 years [65]. The device was located in a trouser pocket freely chosen by the subject in any random orientation. During conducting falls, the subjects placed the mobile device in the trouser pocket on the opposite side of the fall direction [65]. In this thesis, only the accelerometer signals were considered.

4.3 Data-Preprocessing

A combined total acceleration, $x(n)$ is obtained by summing the acceleration data of all three axes. According to [12], the combined acceleration is robust to variations in the sensor position. In this work, the length of $x(n)$ was selected as 3000 samples and $x(n)$ is considered from $n = 0$ to $n = 2999$. Then, the time-frequency representation of the combined total acceleration, $X(n, \omega)$, is obtained by applying the short-time Fourier transform (STFT) as shown in Equation 4.1, and the spectrogram of the combined acceleration data can be computed using Equation 4.2.

$$X(n, \omega) = \sum_{m=-\infty}^{+\infty} x(m)w(n - m)e^{-i\omega m}. \tag{4.1}$$
Figure 4.1: Spectrograms from the dataset: Non-fall (Left) and Fall (Right).

The window used in this work is von Hann window and is given as follows:

\[ w(n) = 0.5 - 0.5 \cos \left( \frac{2\pi n}{M - 1} \right), \quad 0 \leq n \leq M - 1. \]  

(4.2)
A window length of 128 was used with a 50% overlap in order to ensure that all transitions between activities are captured. Furthermore, the activity before and after the fall event was required because it helps distinguish the different types of falls. The window was translated over the entire signal. The magnitude of $X(n, \omega)$ was concatenated to generate a time-frequency map of the entire signal, following a similar approach to produce a combined representation. The magnitude of STFT (spectrograms) of all the segments was concatenated and presented as input for the Siamese network (and also to the Triplet network). Figure 4.1 shows the spectrogram of a non-fall and a fall event. A $65 \times 48$ spectrogram is presented to the Siamese network. The input spectrogram contains 65 frequency bins starting from 0 Hz to 100 Hz evenly spaced.

For the Mobifall dataset, the considered length is from $n = 0$ to $n = 800$, and a generated spectrogram of $65 \times 14$ is presented to the network.

4.4 Training (Siamese)

In Siamese networks, pair generation is of particular interest as the network requires inputs in pairs. The pair generation process has a direct influence on the classification performance of the few-shot architecture. This section elaborates on the training methodology, the optimizer, and the validation strategy.

4.4.1 Optimizer

The Adadelta optimizer is employed to train the Siamese Network. The network was trained with a learning rate of 0.001. The update rule for weights is as follows:

$$w_t = w_{t-1} - \mu_{t-1} \left( \frac{\partial L}{\partial w} \right)_{t-1}.$$  \hspace{1cm} (4.3)
The learning rate can be updated using Equation 4.4.

\[
\mu_t = \frac{\mu_{t-1}}{\sqrt{w_{t}^{\text{avg}}} + \varepsilon},
\]

(4.4)

and \(w_{t}^{\text{avg}}\) is given by

\[
w_{t}^{\text{avg}} = \gamma * w_{t-1}^{\text{avg}} + (1 - \gamma) \sum_{i=1}^{t-1} \left( \frac{\partial L}{\partial w} \right)_{i}^{2},
\]

(4.5)

where \(L\) denotes the loss function (which is a function of weights) at current iteration, \(w_t\) is the updated weight for the next iteration, \(w_{t-1}\) is the weight at the current iteration, and \(\mu_t\) is the learning rate at iteration \(t\), \(\varepsilon\) is a small positive number and \(\gamma\) is the forgetting factor in the exponential averaging, generally set to 0.95. At the beginning of the training, \(w_{t}^{\text{avg}}\) is initialized to zero. Then, the weights are adjusted using a variable learning rate \(\mu_t\). The weights are adjusted using a larger learning rate in the direction of shallow gradients, while smaller learning rates are used for the weights in the direction of steep gradients. Thus, the Adadelta optimizer enhances the speed of learning; hence, speed of training the networks. After 100 epochs of training, convergence was observed.

Both the networks draw Xavier weight initialization [66] from a normal distribution \(W \approx N[0, a]\) where \(a\) is given by Equation 4.4.1,

\[
a = \sqrt{\frac{2}{\text{Fan}_{\text{in}} + \text{Fan}_{\text{out}}}},
\]

(4.6)

where \(\text{Fan}_{\text{in}}\) refers to the number of inputs in a node while \(\text{Fan}_{\text{out}}\) denotes the number of exit nodes as given in [67]. This optimizer along with the required hyper-parameters initialization were used during the training of all three proposed networks presented in
4.4 Training (Siamese)

this work. For training and testing of the proposed Siamese architectures, the dataset is divided into four different sets: the training set, test set, anchor set and validation set. In this work, the evaluation set is comprised of pairs formed between the anchor set and the test set. The validation set is a small part of the whole dataset, which is used to validate the model after every epoch. The validation set is set at 15% for every trial shuffled each time. The details of how the training set and the validation set are chosen are discussed in the following subsection.

4.4.1.1 Similarity Training and Testing

To evaluate the discrimination ability of the architecture between similar and dissimilar exemplars, the Siamese $1 \times 1$ architecture was trained using both the cross-entropy and contrastive loss functions. The architectures were modified to include a sigmoid activation layer at the end of the architecture, and then trained to classify similarity and dissimilarity. The architectures were trained using 30 similar pairs and 30 dissimilar pairs, and a shuffled evaluation set which comprised of pairs formed between the elements of the anchor and the test set. The entire evaluation set and the training set were shuffled in each trial. For the Siamese $1 \times 1$ and the Inception-Densenet based architecture the number of trials was fixed at 20.

4.4.1.2 N-shot training (Binary class)

In the Siamese $1 \times 1$ architecture, in $N = 15$, (i.e., 15-shot learning), 15 examples from each class are randomly sampled from the dataset to form 30 similar and 30 dissimilar pairs, that were used to form the training set; thus, 50 trials were conducted. During training, to compute the validation loss, 15% of the pairs from the dataset were used to validate the model epoch by epoch.

A single training trial is composed of 30 randomized similar and dissimilar pairs
sampled from 15 examples of falls and non-falls in each trial. In the Siamese-Inception-Densenet based architecture, the number of trials was set at 20.

The training pairs are used to train the Siamese network. To validate and test the Siamese $1 \times 1$ and the Inception-Densenet architecture, an anchor set comprising of a single representation of a fall and non-fall anchor was shuffled over each trial. The testing and the training sets were also shuffled randomly. This allowed the members in the anchor set to be part of the training set and the members of the training set to act as anchors.

All possible pairs are formed among the test and the anchor entries to build the evaluation pairs within each trial. These evaluation pairs are used to test the network during each trial.

In 1-shot ($N = 1$) training, the training pair consist of a single similar and dissimilar pair throughout the training trials. Due to memory and time required for training the Siamese $1 \times 1$, 50 random trials were conducted; however, for the Siamese-Inception-Densenet, only 20 trials were conducted. In the classification stage, the class of the anchor is assigned to the unknown sample based on the minimum dissimilarity score associated with the pairs formed using the anchors.

### 4.4.1.3 Multi-way Classification

The class declaration in a multi-way classification is performed as follows:

$$a^* = \text{arg} \min_c \{ s^{(c)} \},$$  \hspace{1cm} (4.7)

where $s$ indicates the dissimilarity score produced by the network based on the distance metric. Given a test spectrogram $x$ which is classified among $c$ classes. Anchor spectrograms are also provided as $(x_a)^c_{a=1}$, representing $c$ classes. The network can be
tested using $x, x_a$ as the input for a range $a = 1, \cdots, c$, following which a predicted class is declared using equation 4.7. For example, if we have 3 classes, then we can use three anchors, each belonging to a different class to test the chosen input, $x$.

### 4.4.2 Selection of Pairs

Since the pairs selection plays an important role in the performance of the network, it is essential to identify what type of pairs should be used. Ideally, the networks should be trained on hard pairs. This can be achieved by measuring the distance between the embeddings generated by each of the untrained branches. However, this has not been performed in this work. Instead, by forming pairs from 15 examples of falls and 15 examples of non-falls, every single trial will rarely result in only easy pairs. Because in each trial, 60 pairs are used to train the network. Overall, the training encompasses 3000 pairs across the trials. Consequently, the training regime will have a mixture of both easy and hard pairs.

### 4.4.3 Comparators

In this thesis, various comparators were selected to assess the strength of the proposed architectures.

Transfer learning architectures are used as a comparator because it is generally an architecture of choice when training data is limited. However, the architecture still has an initial requirement of large training data. Once the architecture has been trained, the retraining of the architecture can be performed using a limited amount of data. The $1 \times 1$ and Inception-Densenet based Siamese network architectures are evaluated against the transfer-learning architectures.

The proposed Inception-Densenet architecture was compared with two state-of-
the-art benchmark architectures, namely, Densenet and Inception architectures.

Furthermore, a Siamese network using an auto-encoder was introduced in the literature [57] for fall recognition using audio-based sensors. In this thesis, following the work presented in [57], a Siamese auto-encoder was designed and tested as a baseline comparison for few-shot learning.

### 4.4.4 Environment

The networks are implemented using the Python’s Tensorflow, Keras frameworks. Moreover, the Sklearn library was used to produce the training-testing splits. The network is trained on the cloud hosted Google Colab Pro version due to local hardware constraints. The weights after training were also saved locally.

### 4.5 Results and Discussions

F-score was selected as the evaluation metric since it provides a proper interpretation of the fitness of the architecture which the accuracy metric may not provide. The F-score has been used to evaluate all proposed architectures. The F-score which is a combination of precision and recall metrics, is defined by Equation 4.8,

\[
F\text{-}Score = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}},
\]

where Precision and Recall are as follows:

\[
\text{Recall} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}},
\]

and

\[
\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}}.
\]
4.5.1 Comparison of Loss Functions for Siamese 1×1 (Binary)

The proposed architecture was trained using CLF and CE as loss functions for similarity and classification. The number of trials were fixed as 50. Similarity testing was also performed using the same set of loss functions. Table 4.3 and Figure 4.2 show the similarity performance on the evaluation data after training, using the Cross-Entropy loss in 15 and 1 shot settings. Also, Table 4.4 and Figure 4.3 shows the performance using the Contrastive Loss in 15 and 1 shot settings.

![Histogram of F-score (in %) of 15-shot CE (Left) and 1-shot CE (Right) (Similarity testing) (Siamese 1 × 1).](image)

Table 4.3: F-score (in %) statistics of CE (Similarity) (Siamese 1 × 1).

<table>
<thead>
<tr>
<th>Type</th>
<th>Median</th>
<th>Mean ± STD</th>
<th>IQR</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-shot</td>
<td>66%</td>
<td>67.5% ± 9.6%</td>
<td>11.9%</td>
</tr>
<tr>
<td>1-shot</td>
<td>67.4%</td>
<td>68.4% ± 3.3%</td>
<td>3.1%</td>
</tr>
</tbody>
</table>
Table 4.4: F-score (in %) statistics of CLF (Similarity) (Siamese 1 × 1).

<table>
<thead>
<tr>
<th>Type</th>
<th>Median</th>
<th>Mean ± STD</th>
<th>IQR</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-shot</td>
<td>92.15%</td>
<td>91.9% ± 4.25%</td>
<td>4.9%</td>
</tr>
<tr>
<td>1-shot</td>
<td>70.8%</td>
<td>71.9% ± 5%</td>
<td>9.1%</td>
</tr>
</tbody>
</table>

Both 1-shot and 15-shot classifications were carried out using CLF and CE as loss functions. Figure 4.4 and Table 4.5 represent the F-score statistics and distribution obtained throughout the 50 trials for CLF, while Figure 4.5 and Table 4.6 indicate the F-score obtained using CE.

The network trained with CLF loss produces more reliable results than the one trained with CE loss for 1-shot and 15-shot learning. The average and median F-score obtained over 50 trials is better for CLF than CE. The number of shots plays an important role in the classification performance as evident from the two methods. The 15-shot method achieved a lower variance compared to its 1-shot counterpart.
when trained using the CLF function. Based on these observations, it is inferred that CLF is better than CE for the SisFall dataset. Therefore, only CLF is considered for Siamese Inception-Densenet architecture.

A crucial observation between the similarity testing results and the binary classification is that the latter results in a much higher F-score. This is because a classification task is based on the minimum dissimilarity score. In order to compute the minimum, a piece of extra information is required which takes the form of a second anchor. This means that for an N class classification there is a requirement for N likewise anchors. Therefore, simply picking the minimum amongst the pairs formed between the two anchors resulted in a relatively higher performance measure. Another important point is that similarity learning can also be treated as a form of binary classification task. If the network has been optimized to learn the similarity function, then for any given anchor-test pair, the similarity can be interpreted in terms of the probability value indicating that the unknown sample is similar to the anchor. This basically eliminates the need for a second anchor from another class.

Figure 4.4: Histogram of F-score (in %) of 15-shot CLF (Left) and 1-shot CLF (Right) (Classification) (Siamese $1 \times 1$).
4.5 Results and Discussions

Table 4.5: F-score (in %) statistics of CLF (Classification) (Siamese $1 \times 1$).

<table>
<thead>
<tr>
<th>Type</th>
<th>Median</th>
<th>Mean ± STD</th>
<th>IQR</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-shot</td>
<td>93.5%</td>
<td>93% ± 7%</td>
<td>11.5%</td>
</tr>
<tr>
<td>1-shot</td>
<td>70.7%</td>
<td>72.25% ± 10%</td>
<td>13.67%</td>
</tr>
</tbody>
</table>

Figure 4.5: Histogram of F-score (in %) of 15-shot CE (Left) and 1-shot CE (Right) (Classification) (Siamese $1 \times 1$).

Table 4.6: F-score (in %) of CE (Classification) (Siamese $1 \times 1$).

<table>
<thead>
<tr>
<th>Type</th>
<th>Median</th>
<th>Mean ± STD</th>
<th>IQR</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-shot</td>
<td>66.6%</td>
<td>64.87% ± 8%</td>
<td>7.5%</td>
</tr>
<tr>
<td>1-shot</td>
<td>66.8%</td>
<td>62.7% ± 10%</td>
<td>13.0%</td>
</tr>
</tbody>
</table>

4.5.2 Dissimilarity Scores for Siamese $1 \times 1$ and Siamese Inception-Densenet

Figure 4.6 shows the plot of normalized dissimilarity scores resulted for similar and dissimilar pairs when the Siamese $1 \times 1$ and the Siamese Inception-Densenet archi-
4.5 Results and Discussions

Architectures (15-shot-CLF) were trained. In both the graphs, the points on the orange line are the scores generated for the dissimilar pairs, and the points on the blue line denote the scores generated for the similar pairs. In both the graphs, there is a clear separation between similar and dissimilar pairs, thus assuring the discriminating potential of both the architectures. Furthermore, discriminating capability is higher for the Inception-Densenet based architecture. However, in both cases, the architectures were successful in discriminating similar pairs; (i) the anchor was chosen from the fall class while the test signal was also chosen from the fall class and (ii) the anchor was chosen from the non-fall class while the test signal was chosen from the non-fall class as well.

Figure 4.6: Visualization of dissimilarity scores of the first sample 7 similar dissimilar pairs (Siamese $1 \times 1$) (Left) and (Siamese Inception-Densenet) (Right).

Figures 4.7 and 4.8 indicate the role of anchor selection. They show the subset of the evaluation set in discriminating between falls and non-falls for the proposed architectures. The blue line shows the pairings generated for a fall anchor, while the orange line shows the pairings generated for a non-fall anchor. It is observed that in both cases, the network was able to discriminate between pairs of anchors (non-fall/fall) with the examples (non-fall/fall).
4.5 Results and Discussions

4.5.3 Performance Benchmark on the Face Detection Dataset (Siamese $1 \times 1$)

To assess the performance of the proposed model architectures on a standard recognition task, the proposed models have been evaluated on the popular ORL face recognition database from AT & T laboratories [68]. The dataset which has been used has been elaborated in the appendix of this thesis. For this purpose, the architecture was trained on a 10-shot learning task to identify similar and dissimilar faces. The 10-shot learning task shows an extreme case when all the data-points are used for training.
From the 10 examples of each of the class training pairs are formed which include an equal amount of similar and dissimilar pairs. For testing evaluation pairs were formed by creating pairs between the anchor set and the test set. Both the anchor and the test sets have been shuffled with the training set in each of the 50 trials so that the samples in the evaluation set also gets well represented in the training set.

The same ada-delta optimizer discussed earlier was used to train the network with a learning rate of 0.001 and the same weight initializations as discussed in Section 4.4. Table 4.7 shows the average F-score statistics of the architecture for 20 trials.

Table 4.7: F-score (in %) statistics on the ORL-Database.

<table>
<thead>
<tr>
<th></th>
<th>Median</th>
<th>Mean (in %)</th>
<th>IQR (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>98.4%</td>
<td>98.1% ± 1.587%</td>
<td>1.858%</td>
</tr>
</tbody>
</table>

As seen in the Table 4.7, the architecture performed with a very high mean and median F-scores, which effectively indicates the strength of the Siamese network architecture in learning the similarity while trained with a standard benchmark dataset.

4.6 Siamese Inception-Densenet

The Siamese-Inception-Densenet architecture was tested over 20 trials instead of 50 due to memory constraints. The same pair generation process for the Siamese $1 \times 1$ was used as discussed before. The architecture observed a low variance in 15-shot compared to its 1-shot counterpart, which confirms that the number of shots is an important factor in the classification performance of the Siamese network. The spread of the distribution for the proposed architecture is considerably lower when compared to the previous Siamese $1 \times 1$ architecture. The Densenet-Inception based architecture performed much better in comparison with the $1 \times 1$ Siamese architecture. This is
because deeper networks learn better compared to lighter architectures. However, deeper networks require more data than the lighter counterpart. Once a deeper network is brought into the frame work of a few-shot Siamese network, then the architecture can learn effectively.

Figure 4.9 and Table 4.8 show the F-score of the similarity performance for the network. Also, Table 4.9 and Figure 4.10 represent the F-score statistics of the classification performance for 15-shot and 1-shot training scenarios. The Inception-based architecture performed significantly better compared to the Siamese $1 \times 1$ architecture. However, it is not recommended to use these sophisticated architectures for classification when the same performance can be achieved by a very shallow architecture. The reason refers to the total number of parameters. The higher number of parameters indicates more memory during run-time. The Inception-Densenet based architecture featured over 1.4 million parameters while the Siamese $1 \times 1$ that had only 0.7 million.

Table 4.8: F-score (in %) statistics of Siamese-Inception-Densenet (Similarity).

<table>
<thead>
<tr>
<th>Type</th>
<th>Median</th>
<th>Mean (in %)</th>
<th>IQR (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-shot</td>
<td>95%</td>
<td>94.8% ± 3.1%</td>
<td>3.68%</td>
</tr>
<tr>
<td>1-shot</td>
<td>68.35%</td>
<td>70.7% ± 5.5%</td>
<td>10.7%</td>
</tr>
</tbody>
</table>

Table 4.9: F-score (in %) statistics of Siamese-Inception-Densenet (Classification).

<table>
<thead>
<tr>
<th>Type</th>
<th>Median</th>
<th>Mean (in %)</th>
<th>IQR (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-shot</td>
<td>100%</td>
<td>97% ± 4.7%</td>
<td>2.5%</td>
</tr>
<tr>
<td>1-shot</td>
<td>65.11%</td>
<td>68.5% ± 10%</td>
<td>9.7%</td>
</tr>
</tbody>
</table>
4.6.1 Performance Benchmark on the Face Detection Dataset (Siamese Inception-Densenet)

The Siamese Inception-Densenet architecture is evaluated in a similar manner as the evaluation of the Siamese $1 \times 1$ architecture on the AT&T laboratories dataset [68].
using the same optimizer and weight initializations. Table 4.10 shows the average F-score of the architecture for 20 trials.

Table 4.10: F-score (in %) statistics on the ORL-DB.

<table>
<thead>
<tr>
<th>Median</th>
<th>Mean (in %)</th>
<th>IQR (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>99.6%</td>
<td>99.53% ± 0.03%</td>
<td>0.6%</td>
</tr>
</tbody>
</table>

Based on Table 4.10, the architecture performed with a lower variance compared to the previous Siamese $1 \times 1$ architecture (see Table 4.7). The spread of F-scores is also very low compared to those obtained using Siamese $1 \times 1$ architecture. Therefore, by having a deeper Inception-Densenet based architecture, the performance improved significantly on the standard face detection dataset. However, the difference between the mean F-scores is not very remarkable.

### 4.7 Interclass Classification

A Siamese network learns to discriminate in terms of similarity or dissimilarity, but not based on the explicit definition of the class labels. The network does not have prior knowledge about the category of the falls and non-falls subtypes, but is aware of the similar or dissimilar label. To test the architecture for inter-class classification, anchors are selected from the three major groups of falls (forward, backward and lateral), such that each type is represented five times. The unknown sample makes pairs with the anchors, which the trained network outputs a dissimilarity score for each anchor-unknown pair. Then, the minimum is picked, and the class of the anchor is assigned. Tables 4.11 and 4.12 show the performance of the classification of falls using the $1 \times 1$ Siamese network and the Siamese Inception-Densenet network over un-
seen fall classes. Based on our results, the proposed architectures within the Siamese network is able to perform fairly well on new (unknown to the network) samples of unknown classes. This demonstrates the generalization ability in classifying unknown events.

Table 4.11: F-Score in % of Interclass classification of falls (Siamese 1 × 1).

<table>
<thead>
<tr>
<th></th>
<th>FWF</th>
<th>BWF</th>
<th>LAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-scores:</td>
<td>83.3%</td>
<td>60%</td>
<td>83.3%</td>
</tr>
</tbody>
</table>

Table 4.12: F-Score in % of Interclass classification of falls (Siamese Inception-Densenet).

<table>
<thead>
<tr>
<th></th>
<th>FWF</th>
<th>BWF</th>
<th>LAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-scores:</td>
<td>72.7%</td>
<td>85.7%</td>
<td>83%</td>
</tr>
</tbody>
</table>

4.7.1 Dissimilarity Scores (Class assignment)

Figures 4.11-4.13 show the sample dissimilarity scores generated for three correctly classified unknowns after querying the network with the anchor-unknown pairs. Figure 4.11 shows the dissimilarity scores generated between the pairs of the three types of anchors with the unknown sample. The ground-truth of the unknown is a lateral fall sample, that is, the actual description of the event. The network after querying, produced a dissimilarity score of 0.229, 0.350, and 0.150 for forward, backward and lateral fall anchors, respectively. Based on the criterion defined in Section 4.4.1.3, the minimum value is picked, and the sample is correctly classified as a fall of the lateral type.

Similarly, for Figure 4.12, the sample is classified as a forward fall based on the minimum value of 0.28. This minimum value is computed for the pair formed with the
forward fall anchor. In Figure 4.13, the classification is a backward fall based on the minimum value of 0.54 for the pair formed with the backward anchor. The network can be queried using this procedure with any pairs of samples, and classification can be performed simply by ranking.

4.8 Cross Dataset Evaluation

The Siamese architectures were trained using the first 15-shot approach over the SiS-Fall dataset, and tested on the MobiFall dataset. The weights were used from the
4.9 Few-shot Comparator: Siamese Convolutional Auto-encoder

training on the SiSFall and tested on pairs generated on the Mobifall dataset. Evaluation pairs were generated from the Mobifall dataset, which comprised of pairings formed between the shuffled anchor and test set. Table 4.13 shows the performance of the architectures on the Mobifall dataset for the classification of falls and non-falls with an average of 20 trials. As observed, both Siamese architectures performed close to each other without any kind of retraining. This demonstrates the generalization ability of the Siamese network over unseen classes. There is also the difference of data acquisition modality which are different in each of the datasets. The SiSFall dataset was acquired from waist-worn accelerometer, while the Mobifall collected using an accelerometer from a Samsung Galaxy smartphone. This further reinforces that the network was able to adapt on a completely different dataset acquired from a different modality than the training set. This is a classic example of domain adaptation, where the network learns feature representations from a task and employs it to infer on a different task (similar to the original).

Table 4.13: F-score (in %) statistics on Mobifall (Classification).

<table>
<thead>
<tr>
<th>Type</th>
<th>Median</th>
<th>Mean (in %)</th>
<th>IQR (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Siamese 1 × 1</td>
<td>63.4%</td>
<td>63.2% ± 4.7%</td>
<td>5%</td>
</tr>
<tr>
<td>Siamese Inception-Densenet</td>
<td>65.5%</td>
<td>64.3% ± 4.6%</td>
<td>4.75%</td>
</tr>
</tbody>
</table>

### 4.9 Few-shot Comparator: Siamese Convolutional Auto-encoder

To measure the overall strength of the Siamese network, the architecture has been tested against other standard deep learning architectures, and also with another type of few-shot Siamese architecture called SCAE (Siamese Autoencoders) based on [34].
In Siamese auto-encoders, the two Siamese branches feature an auto-encoder that takes an input and compresses it to some dimension. An auto-encoder consists of an encoder which is a feed-forward fully connected neural network that compresses the input into a latent vector representation and encodes the input as a compressed representation. The compressed representation is a distorted version of the original input. Using a decoder which is another feed-forward neural network similar to the encoder, the compressed representation is reconstructed back.

Figure 4.14 shows an intuitive illustration of a convolutional auto-encoder. The network takes in the input and uses a CNN (also referred to as encoder CNN) to encode it to a latent vector (compressed version) and using another CNN (referred to as the decoder CNN), it tries to compute the output from the compressed version. An important point is that the architecture of the encoder CNN is the same as the decoder CNN. This leads to the reconstruction loss between the original input and output. The loss function can be formally defined as shown by Equation 4.11,

$$\mathcal{L}(x, \hat{x}) = \|x - \hat{x}\|^2,$$

(4.11)

where $x$ is the original input and $\hat{x}$ is the reconstructed input. The L2 norm is the error that propagates though the network. The encoder and the decoder are identical with three convolutional layers of $3 \times 3$ dimensions with a ReLU activation function. The feature maps are then flattened and connected to an FC layer which outputs embeddings of 128 dimensions. Once this is generated, the decoder is constructed by reversing the architecture of the encoder. This is done by utilizing the up-sampling layer. The upsampling layer is a weight-less layer that doubles the dimensions of the previous layer. The decoder works by reconstructing the input back from the embedding of 128 dimensions. The resulting reconstruction is then further converted...
to an embedding, which is then used for computing the contrastive loss (as discussed in the previous chapters). The 128-dimensional embedding is referred to as the latent dimension. The space containing these embedding is called the latent space and each compressed vector in this space is called as latent vector. Ideally, this leads to compression of the input by a large factor. Higher dimensions would result in better reconstructions.

In Siamese architecture, the auto-encoders are replicated twice in the branches, and the two reconstructed representations are compared using the distance metric, then the contrastive loss is calculated. If the network is correctly trained, then the convolutional encoder can generate reconstructed representations of the input such that they are similar if the inputs belong to the same class otherwise dissimilar.

Table 4.14 shows the performance of the SCAE architecture in fall detection. As SCAE architecture has fewer parameters (refer to Table 4.19), it has a relatively poor performance compared to the other Siamese architectures discussed in the thesis.

Table 4.14: F-score (in %) statistics of SCAE (Classification).

<table>
<thead>
<tr>
<th>Type</th>
<th>Median</th>
<th>Mean (in %)</th>
<th>IQR (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-shot</td>
<td>85%</td>
<td>79.07% ± 17.4%</td>
<td>19.2%</td>
</tr>
<tr>
<td>1-shot</td>
<td>68.35%</td>
<td>67% ± 2%</td>
<td>1.49%</td>
</tr>
</tbody>
</table>
Table 4.15: F-score (in %) statistics of SCAE (Classification) - Mobifall.

<table>
<thead>
<tr>
<th>Type</th>
<th>Median</th>
<th>Mean (in %)</th>
<th>IQR (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-shot</td>
<td>67.9%</td>
<td>68.5% ± 3.5%</td>
<td>4.3%</td>
</tr>
</tbody>
</table>

4.10 Transfer Learning Comparators

The transfer learning architectures have been included for comparison mainly because transfer learning architectures can be used when data is limited; however, the architectures still have an initial requirement of training data. The transfer learning architectures in this thesis, were trained using the pretrained Imagenet. The Inception architectures that have been used in this thesis is based on the InceptionV3 architecture [61].

Table 4.16 reports the F-scores and standard deviation of commonly used deep learning architectures. Predictions are inferred from the SisFall dataset using three architectures of transfer learning, namely, InceptionV3, Mobilenet, VGG, while simulating both few-shot (1-shot) and full-shot scenarios for comparison. The transfer learning architectures were trained on the last six layers while freezing the rest, for 50 trials of randomized training and testing.

From Table 4.16, the few-shot (1-shot) learning performed poorly and has higher
4.11 Original Densenet and Inception

variance when compared to all the three architectures as indicated by poor F-scores. This observation is even true for full-shot training. Mobilenet performed the best amongst all the transfer learning comparators under full-shot training scenario.

Table 4.16: F-score (in %) statistics of transfer learning architectures.

<table>
<thead>
<tr>
<th>Type</th>
<th>Mean</th>
<th>Median</th>
<th>IQR</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLINCEPTION (Full-shot)</td>
<td>67.55% ± 21.0 %</td>
<td>70.29%</td>
<td>38.35%</td>
</tr>
<tr>
<td>TLINCEPTION (1-shot)</td>
<td>39.9% ± 17.3 %</td>
<td>41 %</td>
<td>43.24 %</td>
</tr>
<tr>
<td>TLMOBILENET (Full-shot)</td>
<td>89.79% ± 9 %</td>
<td>96.8%</td>
<td>6 %</td>
</tr>
<tr>
<td>TLMOBILENET (1-shot)</td>
<td>24.03% ± 7 %</td>
<td>35 %</td>
<td>9 %</td>
</tr>
<tr>
<td>TLVGG16 (Full-Shot)</td>
<td>63.7% ± 10.8 %</td>
<td>67.19%</td>
<td>27.2%</td>
</tr>
<tr>
<td>TLVGG16 (1-shot)</td>
<td>62.03% ± 24.6%</td>
<td>60 %</td>
<td>8 %</td>
</tr>
</tbody>
</table>

4.11 Original Densenet and Inception

The state-of-the-art Densenet and Inception networks have been selected as a suitable comparator based on the premise that the proposed Siamese architecture was inspired by these two architectures. The Densenet121 contains 121 layers. The Inception architecture is the original InceptionV3 architecture proposed by [61]. The architecture was trained to simulate both full-shot and 1-shot scenarios. Table 4.17 shows the average F-score over 50 trials. The 1-shot version’s performance is relatively poor with a low average F-score compared to its Siamese counterpart. The architecture of Densenet121 enables it to learn better than the InceptionV3. This is evident from the F-scores achieved by both the architectures. Densenet121 has lower spread in F-score than InceptionV3.
4.12 Triplet Network

Triplet network is slightly different in the way they work compared to the Siamese networks. A Triplet network has three branches that can accommodate most types of neural network architectures. These three branches produce embeddings of a Triplet network. As mentioned in the previous chapter 3.3.3, a triplet comprise of an anchor, a positive, and a negative sample. The three embeddings are calculated, and a distance-based metric is computed amongst them as \(d(A, P)\), and \(d(A, N)\). The distance metric is not restricted to be Euclidean distance. The loss function ensures that the distance between the positive and the anchor is smaller than the distance between the negative and the anchor. The important difference in a Triplet network is that the distance between the anchors, positives, and negatives are minimized simultaneously during the training process using the triplet loss function. Also, the sigmoid activation function in the last layer after the network has generated the embeddings, directly computes the similarity with a level of confidence for corresponding pairs formed with the anchor. The Triplet network is trained on randomly generated 15 anchor-positive pairs and 15-anchor-negative pairs from 15 examples of falls and non-falls for each of the 25 trials. For the one-shot approach, the architecture is trained using a single
positive and negative pair randomly generated for every 25 trials.

The test process in Triplet network is almost the same as the Siamese approach, that is, picking the minimum dissimilarity score between an anchor and the test sample. The sigmoid/softmax activation function directly enables the network to output predictions (similarity/dissimilarity) with respect to the anchors. This is a binary classification, where the classes are similar or dissimilar with respect to the anchor. If the anchor class is known, then one can directly declare the class of the output. One may also choose to use rank statistics to measure the degree of similarity/dissimilarity. In order to achieve that, the Triplet network should be trained without sigmoid/softmax activation function. Table 4.18 shows the performance of the Triplet network using 15-shot and 1-shot training setting.

Table 4.18: F-score (in %) statistics of Triplet network.

<table>
<thead>
<tr>
<th>Type</th>
<th>Median</th>
<th>Mean (in %)</th>
<th>IQR (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-shot training</td>
<td>96.29%</td>
<td>96.4% ± 3.61%</td>
<td>4%</td>
</tr>
<tr>
<td>1-shot training</td>
<td>63%</td>
<td>68.6% ± 13.1%</td>
<td>23%</td>
</tr>
</tbody>
</table>

Figure 4.16: Distribution of F-score (in %) Triplet Nets $1 \times 1$ (Classification).
In the case that the Triplet network is correctly trained, then the network can generate embeddings that ensure the positive and negative pairs with respect to the anchors are well separated. This is illustrated in Figure 4.17, which shows the embeddings generated by the Triplet network using the t-distribution stochastic neighbor embedding (that is, t-SNE) [69] for a sample evaluation set in a given trial. The t-SNE is used to convert a high dimensional representation to a low dimensional graphical plot. Each of the dots represents a low dimensional embedding for each samples of the evaluation set. The labels 1 and 0 indicate falls and non-falls, which have been well separated by the network post-training. Figure 4.18 shows the sigmoid activations generated after the last layer in 15-shot and 1-shot architectures. Therefore, the network’s ability to separate the falls and non-falls (discriminating potential) is higher, when the network has access to more data than when it is presented with a single example of each class.

![Figure 4.17: t-SNE evaluation data embeddings before (Left) and after training (Right).](image)
### 4.13 Parameters

Once the type of architecture is chosen, the number of parameters is determined by the size of the input. For example, the Inception architecture has only 16.2 M when trained with the SiSFall dataset compared to 46 M in the original Inception architecture, which was trained on the CIFAR-10 dataset [61], [70]. This change in the number of parameters is based on the change in the input feature dimensions. Similarly the VGG-16 network features 138M parameters based on the original imagenet dataset [70], [71].

Table 4.19: Trainable parameters based on the spectrograms of size 65 × 48.

<table>
<thead>
<tr>
<th>Model</th>
<th>Trainable Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Siamese 1 × 1</td>
<td>~ 0.6 M</td>
</tr>
<tr>
<td>Siamese Inception-Densenet</td>
<td>~ 0.7 M</td>
</tr>
<tr>
<td>SCAE</td>
<td>~ 5.45 K</td>
</tr>
<tr>
<td>Densenet121</td>
<td>~ 7 M</td>
</tr>
<tr>
<td>TLVGG16</td>
<td>~ 7 M</td>
</tr>
<tr>
<td>TLMOBILENET</td>
<td>~ 3.47 M</td>
</tr>
<tr>
<td>TLInception</td>
<td>~ 16.4 M</td>
</tr>
</tbody>
</table>

Table 4.19 contains the number of trainable parameters based on the size of the input spectrogram used in this work. The proposed Siamese network with the 1 × 1 architecture and the Triplet network with the 1 × 1 architecture, contain 0.6 M
parameters, which is lower than the other state-of-the-art networks. The proposed Siamese Inception-Densenet architecture has 0.7 M parameters, (only 0.1 M more than the proposed Siamese 1 × 1 architecture), which is attributed to its hybrid structure (including depth for learning) yet ensuring fixed number of feature maps in every layer (as shown in the previous chapter) when comparing to the state-of-art networks that use full-shot learning. The Densenet121 based architecture had 7 M parameters. The SCAE architecture had only 5.45 K parameters, which makes it extremely light.

Moreover, transfer learning technique has been resorted to effectively reduce the number of trainable parameters by freezing some layers and training only on a limited number of layers for the Fall detection task, while simulating both few-shot and full-shot scenarios. In the transfer learning of Mobilenet, the effective number of trainable parameters was 3.47 M parameters after training the last six layers of the network while freezing the rest. Similarly, for VGG-16, the trainable parameters were reduced to 7 M parameters (from 134 M) after training the last six layers of the architecture.

### 4.14 Summary

This chapter summarized the results obtained in this study. The results of the various proposed architectures, along with their associated loss functions were discussed. F-score (fitness) was used to evaluate the performance of the proposed learning models because F-score is a more informative metric than classification accuracy. The F-score provides a better idea of the overall fit of the model compared to accuracy, which is not robust to class imbalance. The ability of the proposed few-shot architectures to distinguishing between similar and dissimilar pairs was confirmed through this work. This chapter also introduced two types of comparators. The first comparator
was the few-shot Siamese Convolutional Autoencoders (SCAE) which was trained in
conditions similar to the proposed architectures. The second distant comparators used
the transfer learning approaches where only a few layers were trained while keeping
the rest unchanged. Results from the comparator studies confirm the robustness of
generalization ability in the few-shot networks despite learning with limited data.
The results also confirm that a larger training dataset will lead to improvement in
performance metrics. This is evident from the number of shots. 1-shot training yielded
an inferior learned Siamese model while 15-shot yielded a better trained model.
Chapter 5

Conclusions and Directions for Future Work

Modern deep learning-based frameworks require an enormous amount of training data for effective learning. The amount of data will determine how well the network has been trained and how well the network can draw inferences from the incoming test data. Unfortunately, in the machine learning field, many architectures are developed assuming that a lot of data is available. For example, many machine learning architectures use the available datasets such as cat-dog images, on the internet. However, not every problem is a cat-dog classification problem where data can be acquired in thousands. In this thesis, the case of learning with limited data was considered. The fall detection datasets are limited and they contain limited number of fall exemplars. The nature of the human fall detection problem makes it particularly difficult to obtain large amount of fall data. Through the literature review, it was observed that the majority of the datasets contained falls performed by actors and did not emulate the actual conditions. This may lead to bias in the inferences made by the detection systems. In the case of human activity recognition, there are relatively fewer issues
related to the availability of data since regular non-fall activities are much easier to acquire. Therefore, it became necessary to find an effective way to train the model using very few data points. Few-shot architectures, particularly Siamese networks, are very robust to learning with few data and have simple construct. Therefore, in this thesis, few-shot learning has been undertaken for fall activity detection. Ideally, very few datasets relating to fall detection exist that comprehensively cover the majority of age groups (from youth to elderly). For example, the SiSFall dataset is one of the comprehensive dataset that cover large age group and therefore has been utilized in this thesis. The SiSFall dataset contains fall and non-fall signals obtained with wearable IMUS sensors.

5.1 Contributions

The prepossessing stages of the time series IMU signals employed a time-frequency representation using the STFT. The STFT was used to generate spectrograms. These spectrograms were used as inputs for the Siamese network. Unlike the traditional approach, these STFTs were not converted into images. Two particular architectures were proposed; one of them was shallow $1 \times 1$ design and other one was a deeper design which was a novel fusion of the Inception and Densenet based modules.

The shallow architecture made the network lighter with respect to the number of parameters efficient and retained the original size of the feature maps as it passes through the convolutional layers.

The Densenet-Inception based module is based on the principle that deeper Networks can learn more reliably than the shallow networks; however, deeper networks encounter vanishing gradient issue which impairs learning. To address this issue, the output of each layer of the Densenet is concatenated with the previous layer. This
leads to stronger gradient flow and feature map diversification as every layer had access to the previous layers’ information. To take it a step further, the Inception module was introduced for multi-level feature extraction. The Inception module performs various operations, such as convolution, pooling, convolution and then stacks the outputs over one another. The Inception module also brings in the notion of $1 \times 1$ filters (which were used solely in the this thesis’s prior model) to reduce the depth of the feature maps, which makes the model parameter efficient.

The proposed architecture is an adaptation of the Inception Module and the Densenet module. In the proposed network, the output of the Inception module is modified by using an intermediate transition layer to set the final number of feature maps to 50. This leads to a reduction in the number of parameters for the Inception module. Next, the transition layer’s output becomes the input to the DenseNet module with a fixed growth rate of 10 feature maps for every convolution layer. This eliminates the need for the composite layer, which exists in the original DenseNet architecture. Another important change to the original Densenet architecture is the reduction in the number of concatenations. The proposed architecture has only three concatenations instead of concatenations in every layer, as seen in the original Densenet.

The Densenet-Inception based module was trained for 20 trials due to memory constraint using the pair generation process, as discussed in the previous chapter. Similarly, the $1 \times 1$ model was trained for 50 trials. Two types of loss functions, contrastive and cross-entropy loss were investigated for few-shot learning. Contrastive loss was found to be effective with the dataset that was used in this study. Therefore, contrastive Loss Function was used to train both models considered in this thesis. Additionally, the Triplet loss function was investigated for the few-shot $1 \times 1$ networks. Triplet Networks are relatively different from Siamese networks. A Triplet network
has three inputs instead of two and it computes the embedding for three of the inputs all at once. There is no reported work in the literature that has used Triplet network for fall detection and this thesis is the first available research work in applying Triplet network for fall detection using wearable sensors.

The Densenet-Inception inspired model performed better than the Siamese $1 \times 1$ architecture in detecting falls. The average F-score of Densenet-Inception model was 97% while the the $1 \times 1$ model had an average F-score of 93%. The 1-shot performances were quite poor, achieving F-scores in the range of 60% for both the models. For Triplet networks, an average F-score of 96.29% was obtained in 15-shot settings. Triplet network had the worst performance across all the proposed architectures in 1-shot settings. When the $1 \times 1$ Siamese network was trained using cross-entropy loss, Siamese a marked reduction in performance was also observed. The Triplet network with the $1 \times 1$ architecture was trained using the Triplet loss.

The generalization power of the Siamese-Nets was investigated by testing on unseen data from the MobiFall dataset (which used a different sensor modality). The model performed with an approximate F-score of 65% without any form of retraining. Furthermore, the model was also tested on unseen sub-classes of falls without any form of retraining to highlight the learning and generalization potential of the $1 \times 1$ Siamese network.

The proposed architecture’s performance was compared with the Siamese convolutional auto-encoders (SCAE). This is the only application of a Siamese network that has been used in fall detection using acoustic data. The proposed network had better performance when compared to SCAE. Moreover, the architecture performance was also compared to the classical ML/DL approaches over a limited dataset compared to standard transfer learning architectures.
5.2 Limitations of the Current Work

Certain significant drawbacks can arise from the Siamese networks which are summarized below:

- **Pair Generation**: It is important to exhaustively generate pairs in order to obtain an accurate performance. In this thesis, only a small portion of the pairs was used to train the architecture due to restrictions of the cloud environment (computational restrictions). This may have not enabled the network to maximize its learning capability. An improvement in performance is expected if all such pairs are used.

- **Variation in the data**: Even though Siamese networks can learn from little data, it should be noted that in 1-shot setting, where only a single example is there in the training set, learning is very limited. If the target problem has a lot of variations among the data (as seen for the fall detection case), then the 1-shot approach is not reliable and results in poor performance, as the network cannot capture these variations. Therefore, the number of shots has to be increased for better performance. It is always advantageous to allow the network access to more data to improve the performance even if the network performs well on limited data as observed.

- **Choice of anchors**: The choice of an anchor is a challenging task as it directly influences the classification performance. Ideally, a true anchor set can be obtained through domain knowledge expertise or via inter-disciplinary collaboration. This problem was tackled by randomly sampling the anchors multiple times to find out the variation in the classification performance. If the variation is unacceptable, a strategy should be in place to collect an effective optimum
set of anchors. Another approach to overcome miss-classification (due to choice of anchors) is to increase the number of anchors and then use rank-order statistics to declare the class association. This allows the possibility of a larger set of classes to be represented with the possibility to improve the classification performance. However, not every problem is the same and the anchors may or may not have high variations between them. For example: if one uses a case of satellite imaging for ship detection the general structure of a ship is almost always the same regardless of the texture or color. If the network can learn effectively then there would be no need for extra anchors.

5.3 Future Directions

The following future directions for fall detection using few-shot learning are proposed based on the findings from this thesis:

- **Study different loss functions**: There are various other loss functions such as quadruplet loss, ranking loss, hinge loss, that can be used with the few-shot architectures. The performance of the architectures trained using these loss functions has not been investigated. Different loss functions may yield different performances. Depending on the performance of the architectures trained using different loss functions, it may be possible to have developed a committee of few-shot networks to detect a fall event reliably.

- **Interclass fall detections and cross data validation using Triplet network**: Due to time limitations, interclass fall detections and cross data validation using MobiFall dataset was not conducted on Triplet Network. Generalization capability of Triplet network should be investigated as a straight-forward extension of this work.
• Inception-Densenet inspired Triplet network: Triplet networks will find applications in various other few-shot domains, from human activity recognition to tracking of subject’s motions and classifying them into falls/non-falls. Triplet networks offer a significant advantage over Siamese in ranking in a way that the embeddings are calculated in one pass instead of querying the network with anchor-positive pair and anchor-negative pairs separately. This has actually sped up training time.

• Unsupervised Siamese network The performance of Triplet networks with deeper architecture may be considered in the future. Furthermore, an unsupervised approach for training Triplet networks may also be attempted. For instance, convolutional auto-encoders could be used, or a more sophisticated architecture like the Variational auto-encoders (VAE) may be used. Variational auto-encoders allow for sampling from the latent space, which can enable sampling of desirable features. This is required to be investigated. Such few-shot architectures have not been proposed for fall detection.

• Siamese transfer learning: Another notable application can be the use of Siamese networks by applying pretrained weights from some other related task and then transfer the learning to a new task. This will allow one to use fewer training examples than the Siamese networks proposed in this thesis, as the network will already have previous knowledge of a task that may be related to the objective task.

• Siamese using individual pretrained networks: In this approach, two separate pre-trained architectures are used in the branches of the Siamese neural network, which may allow for faster training due to training of a fewer number of layers while freezing the rest. It would be preferable to have the architecture pre-
trained to a related task such as human activity recognition. This will allow for meaningful domain-specific transfer learning.

- **Advanced few-shot networks for fall detection**: Recent advances in few-shot learning such as matching networks [72], prototypical networks [73], model agnostic meta-learning for fast adaptation [74] could be a potential future direction.

- **Anchor selection process**: Since there is no explicit ideal anchor available, the networks discussed in the course of the thesis have been focused on training with as many types of anchors possible by performing 20-50 trials. However, as not all anchors are alike, Siamese and Triplet networks must be trained on hard pairs. A hard pair is referred to as a pair whose L1/L2 norm is higher for similar pairs and lower for dissimilar pairs. The goal is to allow the network to learn the separation effectively through training. The type of pairs can be selected by querying a pair to a Siamese/Triplet network. The branches of the network will output embeddings. The embeddings resulted from the untrained network can then be measured using distance-based metrics. If the embeddings are closer for a given dissimilar pair then it can be treated as a hard pair and an easy pair otherwise.
Chapter 6

Appendix

6.1 Metrics

Following tables will illustrate some other metrics of the proposed architectures which were not discussed in the thesis.

Table 6.1: Format of Confusion Matrix.

<table>
<thead>
<tr>
<th>Type</th>
<th>Fall (Actual)</th>
<th>Nonfall (Actual)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall (Predicted)</td>
<td>SUM(TP)</td>
<td>SUM(FP)</td>
</tr>
<tr>
<td>Nonfall (Predicted)</td>
<td>SUM(FN)</td>
<td>SUM(TN)</td>
</tr>
</tbody>
</table>

*Note* F-score computed by scikit-learns confusion matrix has some rounding off.
6.1.1 Siamese Inception-Densenet

Table 6.2: Mean Precision and Recall in (%) for 1-shot and 15-shot Siamese Inception-Densenets.

<table>
<thead>
<tr>
<th>Type</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-shot</td>
<td>55.8</td>
<td>92.3</td>
</tr>
<tr>
<td>15-shot</td>
<td>100</td>
<td>95</td>
</tr>
</tbody>
</table>

Table 6.3: Sum Confusion Matrix for 15-shot Siamese Inception-Densenet.

<table>
<thead>
<tr>
<th></th>
<th>Fall (Actual)</th>
<th>Nonfall (Actual)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall (Predicted)</td>
<td>3200</td>
<td>0</td>
</tr>
<tr>
<td>Nonfall (Predicted)</td>
<td>168</td>
<td>4832</td>
</tr>
</tbody>
</table>

Table 6.4: Sum Confusion Matrix for 1-shot Siamese Inception-Densenet.

<table>
<thead>
<tr>
<th></th>
<th>Fall (Actual)</th>
<th>Nonfall (Actual)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall (Predicted)</td>
<td>2560</td>
<td>212</td>
</tr>
<tr>
<td>Nonfall (Predicted)</td>
<td>2094</td>
<td>1720</td>
</tr>
</tbody>
</table>

Table 6.5: Sum Confusion Matrix for 15-shot Siamese Inception-Densenet Mobifall (20 trials).

<table>
<thead>
<tr>
<th></th>
<th>Fall (Actual)</th>
<th>Nonfall (Actual)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall (Predicted)</td>
<td>95</td>
<td>48</td>
</tr>
<tr>
<td>Nonfall (Predicted)</td>
<td>57</td>
<td>198</td>
</tr>
</tbody>
</table>
6.1 Metrics

Table 6.6: Mean Precision and Recall in (%) for 15-shot Siamese Inception-Densenet Mobifall Test.

<table>
<thead>
<tr>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>65.7</td>
<td>62</td>
</tr>
</tbody>
</table>

6.1.2 Siamese $1 \times 1$

Table 6.7: Sum Confusion Matrix for 15-shot Siamese $1 \times 1$ Mobifall (20 trials).

<table>
<thead>
<tr>
<th></th>
<th>Fall (Actual)</th>
<th>Nonfall (Actual)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall (Predicted)</td>
<td>93</td>
<td>49</td>
</tr>
<tr>
<td>Nonfall (Predicted)</td>
<td>59</td>
<td>197</td>
</tr>
</tbody>
</table>

Table 6.8: Mean Precision and Recall in (%) for 15-shot Siamese $1 \times 1$ MobiFall Test.

<table>
<thead>
<tr>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>65.4</td>
<td>61</td>
</tr>
</tbody>
</table>

6.1.3 Triplet Network

Table 6.9: Mean Precision and Recall in (%) for 1-shot and 15-shot Triplet Network.

<table>
<thead>
<tr>
<th>Type</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-shot</td>
<td>55.1</td>
<td>90.9</td>
</tr>
<tr>
<td>15-shot</td>
<td>97.3</td>
<td>95.5</td>
</tr>
</tbody>
</table>
6.2 Details of the ORL-DB Database: Face-recognition Dataset

The ORL-DB dataset was obtained from AT&T laboratories in the UK. The dataset comprises of different poses of people’s faces smiling. Many researchers used this dataset to evaluate their models’ performance in many standard recognition tasks. Additional details about the way the dataset was acquired can be found at https://camorl.co.uk/facedatabase.html. The first is a smaller version which comprises 10 images of each of the 40 people. This has been done to observe how the designed models perform on a standard recognition task in terms of learning the similarity function. The dataset that was used also had pre-made pairs.

Table 6.10: Sum Confusion Matrix for 15-shot Triplet Network.

<table>
<thead>
<tr>
<th></th>
<th>Fall (Actual)</th>
<th>Nonfall (Actual)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall (Predicted)</td>
<td>725</td>
<td>20</td>
</tr>
<tr>
<td>Nonfall (Predicted)</td>
<td>34</td>
<td>1046</td>
</tr>
</tbody>
</table>

Table 6.11: Sum Confusion Matrix for 1-shot Triplet Network.

<table>
<thead>
<tr>
<th></th>
<th>Fall (Actual)</th>
<th>Nonfall (Actual)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall (Predicted)</td>
<td>725</td>
<td>590</td>
</tr>
<tr>
<td>Nonfall (Predicted)</td>
<td>72</td>
<td>438</td>
</tr>
</tbody>
</table>
References


REFERENCES


