12-2023

Enhancing Time Series Hashing Performance via Deep Orthogonal Hashing

Mahmudul Hasan Robin
The University of Texas Rio Grande Valley

ENHANCING TIME SERIES HASHING PERFORMANCE VIA DEEP ORTHOGONAL HASHING

A Thesis

by

MAHMUDUL HASAN ROBIN

Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Major Subject: Computer Science

The University of Texas Rio Grande Valley

December 2023

ENHANCING TIME SERIES HASHING PERFORMANCE VIA DEEP ORTHOGONAL HASHING

A Thesis
by
MAHMUDUL HASAN ROBIN

COMMITTEE MEMBERS

Dr. Yifeng Gao
Chair of Committee

Dr. Emmett Tomai
Committee Member

Dr. Dong-Chul Kim
Committee Member

December 2023

ABSTRACT

Robin, Mahmudul H., Enhancing Time Series Hashing Performance via Deep Orthogonal Hashing. Master of Science (MS), December, 2023, 39 pp., 2 tables, 7 figures, references, 55 titles.

Deep hashing has been widely used for efficient retrieval and classification of high-dimensional data like images and text. However, its application to time series data is still challenging due to the data's temporal nature. To tackle this issue, a new deep hashing method has been proposed that generates efficient hash codes and enhances the time series hashing performance using a ResNet model with Orthohash (Cosine Similarity Loss). The proposed method uses one loss architecture while using ResNet model for efficient hashing. It uses the Character Trajectories dataset to extract discriminative features from the time series data. These features are then converted into binary codes using a quantization function to produce hash codes that can be easily stored and compared. The study evaluated the performance of the hash codes using t-SNE (t-Distributed Stochastic Neighbor Embedding) technique. The classification performance of the time series data are evaluated using accuracy and F1 score. The experimental results show improved deep hashing performance with significantly better distinct clusters for each class within the dataset. The proposed method outperformed other state-of-the-art methods in terms of accuracy and efficiency.

DEDICATION

I would like to dedicate this thesis to my parents for their unconditional love and support. The unwavering love and support of my parents made all of this possible. My parents, who are currently far away from me at the moment but consistently shower me with their blessings and support, give me the willpower to successfully complete my master's degree.

ACKNOWLEDGMENTS

I wish to convey my deep appreciation to all those who have offered their support throughout the course of my work and the completion of this thesis. Their unwavering encouragement, aid, and guidance have played a pivotal role in my accomplishments.

I am profoundly thankful to Dr. Yifeng Gao, my research advisor and chair of the thesis committee, for his expert guidance, valuable feedback, and continuous support throughout this journey. His mentorship and expertise were indispensable in shaping the research questions and methodology, and my achievements would not have been attainable without his assistance.

I would also like to express my thanks to Dr. Emmett Tomai and Dr. Dong-Chul Kim for their backing and for providing me with various opportunities during my tenure at UTRGV.

Furthermore, my appreciation extends to UTRGV for granting me the Presidential Research Fellowship, which has had a substantial impact on my academic and research pursuits. Without this financial aid, my progress would have been severely hindered.

Lastly, I extend my heartfelt gratitude to Dr. Zhixiang Chen and the Graduate College of UTRGV for recognizing my potential and bestowing upon me this esteemed scholarship. Their support has been instrumental in enabling me to pursue my research interests and fulfill my academic objectives.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER I

INTRODUCTION

1.1 Introduction

Deep hashing offers numerous advantages over conventional methods of data representation and retrieval. It excels in handling high-dimensional data with intricate structures, such as images and time series data, by learning distinctive features that capture the inherent data structure. Additionally, deep hashing is versatile, suitable for both supervised and unsupervised settings, allowing flexibility in choosing the training data. Recognizing the significance and potential of deep hashing, it finds applications in diverse domains such as healthcare, finance, and security. For instance, in medical imaging, deep hashing proves valuable for efficiently retrieving similar medical images to aid in diagnosis and treatment planning. Similarly, it finds utility in financial data analysis for tasks like fraud detection and risk assessment. Furthermore, deep hashing contributes to security applications like biometric identification and surveillance. Within the realm of real-world large-scale image retrieval systems, image hashing stands as a vital component. Its primary function is to represent an image's content using a binary code, facilitating efficient storage and precise retrieval. Recent advancements in deep hashing methods [1], [2] have outperformed traditional hashing approaches [3, 4, 5, 6, 7]. These deep hashing methods can be categorized [8] based on how they measure the similarity of the learned hashing codes, namely: pointwise [9, 10, 11, 12, 13], pairwise [2], [14, 15, 16], triplet-wise [17], [18], or listwise [19]. Within these categories, pointwise techniques exhibit a computational complexity of O(N), where N denotes the number of data points. Conversely, the other methods possess a minimum complexity of $O(N^2)$. Consequently, when dealing with large-scale problems, only pointwise methods remain practical [20]. Consequently,

they have become the primary focus of recent research in this field. A deep hashing neural network inherently encompasses multiple learning objectives. Upon receiving an input image, the network produces a continuous code, represented as a feature vector. This continuous code is subsequently transformed into a binary hash code through a quantization layer, often implemented as a sign function. This binary code effectively captures the image's content and plays a pivotal role in tasks involving large-scale image retrieval. Thus, there are two principal objectives to address. Firstly, for the binary codes that serve as the final model output, it is essential for intra-class hamming distances to be small and inter-class distances to be significant. Secondly, to regularize the continuous codes, there is a need for minimizing quantization errors. However, the quantization layer introduces a challenge in the form of the vanishing gradient problem, which limits the learning process. Using relaxation techniques may offer potential solutions to this issue [15] [2] [14], but due to the inherent quantization errors they introduce, they often yield suboptimal hash codes. As a result, the most recent advancements in deep hashing techniques [13] [14] [21] [22] [23] have a clear objective of minimizing quantization errors in their learning process. However, these two primary objectives/loss functions are still insufficient. Many existing systems utilize additional loss functions to ensure the quality of the hash codes. These supplementary losses encompass code orthogonality [24] [25], constraints on weights to maximize Hamming distance [10], and bit balance loss [9] [11] [23]. Furthermore, these losses are designed to address the vanishing gradient issue arising from the sign function employed for converting continuous codes into binary ones [[11] [22] [26]]. Consequently, modern hashing models often contend with a significant number (more than four) of loss functions, which poses optimization challenges and diminishes their effectiveness. Despite several studies focusing on learning to hash from time series data [27, 21, 28, 29, 30], they have not comprehensively addressed the preservation of similarity and temporal relationship properties [31]. Therefore, this research aims to overcome these limitations and enhance the efficacy of deep hashing for time series data. This study introduces a novel deep hashing model named OrthoHash, distinguished by its use of a single loss function tailored for time series data. This eliminates the need for intricate loss weight tuning and simplifies the optimization process. Typically, a deep hashing model necessitates

at least two objectives: enhancing binary code discriminativeness and minimizing quantization errors. However, these objectives are closely interrelated and can be consolidated into a single loss function.

Extensive experiments involving both the cross-entropy loss function and OrthoHash loss techniques illustrate that OrthoHash surpasses the performance of the cross-entropy (CE) loss. Particularly noteworthy are the results in character retrieval tasks from time series data in images (Character Trajectories dataset), where this research attains significantly improved accuracy of 87.81%. This achievement enhances discrimination among distinct characters compared to the CE loss technique.

1.2 Problem Statement

1.2.1 Fact

Deep learning, a branch of machine learning, has made remarkable strides across diverse fields like image recognition, speech processing, machine translation, and gaming, fundamentally transforming the capabilities of artificial intelligence systems.

In the context of image retrieval, previous approaches have introduced various methods based on loss functions, but they fall short when it comes to generating hash codes efficiently. To be more specific, an earlier proposal suggested a single loss function that leverages cosine similarity to streamline hash code retrieval. Surprisingly, there is no other research available that harnesses the cosine similarity loss function to achieve efficient deep hashing, especially when dealing with the character trajectories dataset, which involves time series data.

1.2.2 Problem

Using deep learning models can be quite resource-intensive, requiring a significant amount of time and computing resources. This can be particularly challenging for small businesses and scientific research projects with limited access to powerful computers and funding. Deep hashing techniques have two main objectives: improving the separation between different classes in terms of hash codes and minimizing errors in the quantization process. Achieving both of these goals usually

involves using various loss functions in combination. However, in many instances, this approach is not efficient and ends up consuming more computational resources.

1.2.3 Solutions

The core idea behind the OrthoHash loss-based deep hashing method is to maximize the cosine similarity between continuous codes (that have been L2-normalized) and their corresponding binary orthogonal targets. This optimization is achieved through the use of a unified loss function known as cross-entropy (CE) loss. What's unique is that this method effectively tackles both objectives simultaneously: improving the separation of different classes in terms of Hamming distance and reducing errors in the quantization process.

Additionally, the OrthoHash method offers several benefits due to its single loss function. Firstly, it enhances the variance within the same class by incorporating a margin concept. Secondly, it easily adapts to scenarios involving multi-label classification by applying Label Smoothing to adjust the cross-entropy loss. Lastly, the method deals with code balancing through the introduction of a batch normalization (BN) layer, eliminating the need for a separate loss term.

In this study, the OrthoHash technique has been integrated into the ResNet34 model architecture to produce efficient hash codes specifically for time series data.

1.3 Roadmap

The remainder of this manuscript is structured as follows:

- Chapter 2 studies and compares the previous works regarding the deep hashing approaches for supervised and unsupervised methods.
- Chapter 3 discusses the different key definitions, concepts, and problems in detail.
- Chapter 4 presents the orthohash loss-based deep hashing method.
- Chapter 5 evaluates the proposed work by adopting different methods.
- Chapter 6 concludes the works done in the thesis. Some ongoing and possible future works are also included.

CHAPTER II

RELATED WORKS

2.1 Supervised Learning Based Deep Hashing Method

2.1.1 HashNet

A deep learning to hash framework called HashNet [15] is proposed to learn the image representation from real life data and retrieve efficient hash codes. The major idea of this work is comprised of two parts. The first part generates the hash codes using the sign activation function, which ensures the conversion of the input to a binary form. In addition, the second part solves a real dataset problem named imbalanced similarity data problem because the number of similar pairings in a real retrieval system is substantially fewer than the number of dissimilar pairs. As a result, the number of similar and dissimilar pairs is very imbalanced. To address this problem, a weighted likelihood loss function is used during the training process. Specifically, the loss gives the similar pair of images more weight than the dissimilar pairs. The proposed framework combines these two parts as a single framework which has not been explored in other state of the art works.

2.1.2 Deep Cauchy Hashing (DCH)

A new image data retrieval approach called Deep Cauchy Hashing [16] is proposed to retrieve similar data points and generate accurate hash codes to overcome the issue of misclassifying the similar pair of images, particularly within a small hamming distance threshold. Two loss functions are used in the proposed method: Cauchy cross entropy and quantization loss. Cauchy cross entropy loss function learns the similarities between the pairwise data and outputs lower probability for pairwise data having a hamming distance larger than two. The Loss function outputs a better classification performance for similar pairwise images within a small hamming distance for

efficient hash code retrieval. In the training process, the framework proposes a novel discriminative sigmoid function that solves the issue of achieving a higher probability for dissimilar pairings, particularly where the hamming distance between hash codes is much larger than two. This function differs from the other generalized sigmoid functions utilized in previous hashing algorithms such as HashNet. The second loss function is a novel Cauchy quantization loss that converts an input image to binary data. Additionally, it checks the erroneous hash code in the output layer and goes back and forth through the neural network to correct it.

2.1.3 Deep Supervised Discrete Hashing (DSDH)

This research paper introduces a new supervised approach [32] for generating hashing codes in the final layer of a Convolutional Neural Network (CNN) by leveraging both the pairwise similarity between images and their classification information. To retain this similarity information, it employs Maximum A Posteriori (MAP) estimation and introduces a loss function designed to minimize quantization errors. This loss function, based on the negative log likelihood, has the objective of reducing the Hamming distance between similar points while maximizing it for dissimilar points to the greatest extent possible. Additionally, a linear classifier is utilized to connect the learned binary code with the preserved similarity information of the images.

The main contribution of this paper is the introduction of a unified framework that preserves the similarity information between pairs of images and classifies them using a linear classifier, resulting in a significant enhancement in performance. However, one limitation of this work is its omission of the cross-entropy loss in the loss function, which could potentially further improve performance through backpropagation.

2.1.4 Deep Incremental Hashing Network (DIHN)

This research paper introduces a novel approach called the Deep Incremental Hashing Network (DIHN) method [33]. It aims to learn hash codes for newly added images while keeping the existing training data intact and generating hash codes for query images using a combination of hash functions and a CNN model. To the best of our knowledge, previous methods required

retraining the CNN model to handle new training data, a challenge effectively addressed by the DIHN framework. An incremental hash loss function is utilized to facilitate this process, ensuring that the similarity information between query data and the training data is preserved. One significant advantage of this work is its flexibility in using efficient hashing methods like ADSH and DSDH to train the initial hash codes, as the primary focus of this framework is on learning hash codes for new categories of input data. The simultaneous learning of hash codes for new input data and the generation of hash codes using the trained CNN model for test data constitute the key contributions of this paper, enhancing the effectiveness of supervised hashing techniques for large datasets.

2.1.5 Deep Polarized Network (DPN)

This research paper introduces a new concept known as polarization loss [12] to enhance the effectiveness of learning hash codes. It achieves this by aiming to minimize the significant deviation and maximize the minor deviation in Hamming distances when generating hash codes for pairs of similar and dissimilar data. The proposed loss function essentially pushes the Hamming distances within the same class towards the positive side or upper limit, while distances between different classes are pushed towards the negative side or lower limit. Depending on a threshold margin and a target vector, this method seeks to minimize polarization loss across the entire dataset. In essence, this work addresses and seeks to improve the limitations associated with directly generating hash codes from Hamming distances.

2.1.6 Central Similarity Quantization

This research paper introduces a novel concept known as the 'hash center' [13] to effectively generate and differentiate hash codes for various pairs of data. To achieve this, the Central Similarity Quantization (CSQ) method is employed to optimize these hash centers. Additionally, a CNN is trained using data features and these hash centers to generate precise hash codes within the Hamming space while disregarding local similarities. This approach brings hash codes from similar data pairs closer together and spreads hash codes from dissimilar data pairs apart. Unlike previous methods like HashNet, which relied on weighted pairwise similarity and focused solely on local

similarity, the CSQ approach concentrates on global data pair similarity and dissimilarity through hash centers within the Hamming space. This distinctive feature allows it to distinguish between similar and dissimilar data pairs effectively, making it a significant contribution to this framework.

2.2 Unsupervised Learning Based Deep Hashing Method

2.2.1 DeepBit

A new deep neural network called DeepBit [34] has been developed to output the binary descriptor for a set of images using a data augmentation approach and minimizing the information loss and evenly distributing the number of zero's and one's for each bit of the binary code for each image. This approach uses three major loss functions where the quantization loss function minimizes the information loss through the deduction from binarization of each images to initial output of neural network. Like the first loss function, the second loss function optimize the neural network parameter by taking the even number of ones and zeros from each bit position of the binary code for each image. The last loss function uses an augmentation approach by rotating the training data and updates the parameters of the network by minimizing the distance between binary descriptiors of the reference images and the rotated images (from -R to R). As this is an unsupervised deep learning approach, so this augmentation technique helps the DeepBit network to check the similarity between all the images. As a result, the DeepBit approach can optimize the network to help it find the correct binary descriptor for images.

2.2.2 Unsupervised GAN Hashing Network

An innovative deep unsupervised hashing framework, known as HashGAN [35], has been introduced to convert input images into binary hash codes without requiring any labeled data. This framework comprises three essential components: the Generator, Discriminator, and Encoder. In this setup, the objective of the Generative Adversarial Network (GAN) is to provide random variables (noise) to the binary code of real image data and generate output that closely resembles the real images. Three loss functions are employed to minimize errors in generating the correct hash codes within the network. The process begins with the embedding of the real image dataset, to which

random variables (noise) are added. This augmented data is used by the Generator to produce fake images. The hash loss function plays a crucial role in ensuring that the binary values are close to 0 and 1 while evenly distributing the number of zeros and ones. Furthermore, it minimizes the Hamming distance between the original images and the fake images. This approach enables the HashGAN framework to obtain the actual hash codes for the input dataset, thereby addressing issues related to complexity and overfitting. In summary, this framework employs a novel augmentation approach using noise to enhance the original data and reduce hash code dissimilarities between the augmented and original data, ultimately achieving accurate hash codes in an unsupervised manner.

2.2.3 Contrastive Information Bottleneck Hashing (CIBHash)

CIBHash [36] is an innovative unsupervised hashing technique that differs from conventional approaches that rely on reconstruction. Instead, it adopts a unique combination of joint contrastive learning and the Information Bottleneck framework to craft an effective hash code. The key idea is to maximize the mutual information between the representation and output labels. By modifying the objective function, the method retains crucial discriminative semantic information while filtering out less important background details. To facilitate end-to-end training, the model incorporates a probabilistic Bernoulli representation layer. The connection between this probabilistic hashing model and mutual information ensures that the hash codes closely resemble the output labels within the Information Bottleneck framework. What sets this work apart from existing unsupervised deep hashing methods like DeepBit and HashGAN is its utilization of an efficient hashing model within the Information Bottleneck framework. This approach removes redundant information about the original sample and achieves superior performance.

Unsupervised hashing methods learn hash functions that encode data to binary codes by training from unlabeled data [16, 37, 38, 39]. On the other hand, supervised hashing makes use of semantic labels of data points or relevance feedback from click-through data in online search engines to collect similarity information.

Unsupervised deep hashing methods, such as DeepBit and HashGAN, use data augmentation to minimize hashcode differences between augmented and original data. In contrast, supervised

deep hashing methods do not require data augmentation because they use labeled data to generate hash codes for similar and dissimilar data pairs.

Supervised deep hashing methods utilize various techniques like different activation functions (such as Sigmoid Function for DCH), straightforward simple classifiers (like Linear Classifier for DSDH), or loss functions (such as Polarized Loss for DPN) to create hash codes for similar and dissimilar data pairs in a supervised manner. In contrast, unsupervised deep hashing methods use different augmentation approaches on the original data to minimize the differences in hash codes between augmented and original data, thereby achieving accurate hash codes in an unsupervised manner.

Supervised deep hashing techniques utilize pairwise image similarity and classification information to create hash codes, which can lead to better accuracy and performance. In contrast, unsupervised deep hashing approaches can generate accurate hash codes without relying on labeled data and without needing to retrain the entire model when new data is added. Supervised methods can handle imbalanced similarity data and can improve classification performance for similar images within a small hamming distance, a task that unsupervised methods may not be able to accomplish. However, unsupervised methods may have limitations in terms of accuracy and performance compared to supervised approaches. The main disadvantage of supervised deep hashing is that it requires a significant amount of labeled data, which can be challenging and expensive to obtain. In contrast, unsupervised deep hashing methods do not require labeled data and can be applied to large-scale datasets that lack labels, providing a significant advantage in terms of scalability.

Overall, supervised deep hashing methods are ideal for situations where it is necessary to generate distinct hash codes, but this approach requires labeled data. On the other hand, unsupervised deep hashing methods are preferable in cases where labeled data is not accessible, but it is not essential to generate distinctive hash codes.

CHAPTER III

DEFINITION AND PROBLEM SETTING

This chapter will provide a brief summary of the prerequisite materials, tools, and methods used in this research. They are fundamental to the rest of the thesis and must be understood prior to moving forward.

3.1 Deep Hashing

Hashing is a widely used technique for approximate nearest neighbor searches, especially in large-scale image retrieval [40]. It involves using a specific mathematical function to transform data of any length into a condensed alphanumeric representation of a fixed size, typically a shorter string [41]. Deep Hashing is a method within computer vision that aims to perform fast and efficient approximate nearest neighbor searches in high-dimensional data. Its goal is to convert complex, high-dimensional data into compact binary codes that are easy to compare and store. This is achieved by combining traditional hashing techniques with deep neural networks, which learn a hash function capable of preserving the similarity between data points. By representing input data as binary codes, it becomes possible to quickly and efficiently search for similar items within large datasets. Deep Hashing has found successful applications in various domains, including image retrieval, content-based image retrieval, and face recognition, often outperforming conventional methods. The primary objective of image hashing is to represent an image's content using a binary code, enabling efficient storage and accurate retrieval [8]. Recent advancements in deep hashing methods [1] [42] have demonstrated significant improvements over traditional hashing approaches [3, 4, 5, 6, 7]. Furthermore, deep hashing methods can be categorized based on how they measure the similarity of the learned hashing codes: pointwise [9, 10, 11, 12, 13], pairwise [2, 14, 15, 16],

triplet-wise [17] [18], or listwise [19]. Recent developments in deep learning for hashing [15] [43] have shown that deep neural networks can enable end-to-end representation learning and create nonlinear hash functions [16].

3.1.1 Hashing Methods

Conventional hashing methods come in various categories [8]. Data-independent techniques like Locality-sensitive Hashing (LsH) [44] and its kernelized version (KLsH) [4] have laid the foundation for hashing principles. They emphasize the importance of code balance, uncorrelated bits, and similarity preservation. In contrast, data-dependent methods [6] strive to create hash codes that are more compact and tailored to specific datasets [45]. Recently, deep learning-based hashing methods [1] [14] have gained prominence in hashing research due to the superior learning capabilities of deep neural networks (DNNs). These methods introduce various learning objectives for training hash codes using a dataset. These objectives encompass: 1) Task-specific learning, further divided into pointwise, pairwise [2, 14, 15, 16], triplet-wise [17, 18], or listwise [19], as well as unsupervised approaches [10] [26]; 2) Quantization error minimization, where the goal is to minimize the p-norm (typically with p = 2) between continuous codes and hash codes; 3) Code balancing [11] [26]. For a more comprehensive overview, readers can refer to surveys on learning to hash [46]. Learning-based hashing methods can be categorized into three main groups: unsupervised methods, which exclusively use unlabeled data for learning hash functions, and semi-supervised and supervised methods [1].

3.1.2 Supervised Deep Hashing

Supervised hashing, which leverages similarity and dissimilarity data between pairs of entities, has gained significant attention lately [1]. In Supervised Deep Hashing, a deep neural network is trained to learn a hash function, guided by class labels or ground truth information. During training, the network is fine-tuned to minimize the gap between the binary code produced by the hash function and the actual ground truth labels. This ensures that the hash function maintains the semantic similarities among data points within the same category while effectively distinguishing

between those from different categories. In recent years, various supervised deep hashing methods have been introduced [1, 9, 10, 12, 13, 14, 15, 16, 17, 32, 33, 47], each offering its own set of advantages and limitations.

3.1.3 Unsupervised Deep Hashing

Unsupervised Deep Hashing is a method used to create concise binary representations of high-dimensional data without relying on labeled information. It employs a deep neural network to learn a hash function that captures the inherent relationships and similarities within the data. The network fine-tunes the hash function by minimizing a loss function that measures the similarity between the binary representation and the original data. This approach is widely applied in tasks like image retrieval, image classification, and face recognition due to its efficiency in conducting quick and effective similarity searches.

Compared to traditional unsupervised hashing methods, Unsupervised Deep Hashing takes a step further by employing advanced techniques such as Autoencoder-based, Variational Autoencoder-based, and Generative Adversarial Network-based Deep Hashing. These methods leverage deep neural networks to learn more complex and non-linear hash functions, enabling them to capture intricate relationships and patterns in the data. This results in more accurate binary codes that better preserve the inherent structure and similarities among data points.

Unsupervised Deep Hashing finds extensive use in tasks like image retrieval, image classification, and face recognition. The compact binary codes generated by the hash function serve as efficient feature representations for large-scale data, facilitating swift and effective similarity-based searches and retrievals. Over the years, various unsupervised deep hashing methods have been introduced [34, 35, 36], each offering its own set of advantages and limitations.

3.2 Hamming Distance

The Hamming distance is a metric to gauge the dissimilarity between two binary sequences of the same length. It accomplishes this by counting the positions where corresponding elements in the sequences differ. In the realm of computer vision and image retrieval, the Hamming distance

finds frequent use in assessing the similarity of binary codes generated by deep hashing techniques. It provides a straightforward and effective means of quantifying the distinction between two binary codes. A smaller Hamming distance between two binary codes signifies a higher degree of similarity, while a larger Hamming distance indicates greater dissimilarity. In the context of image retrieval, the Hamming distance associated with a hash code can be indicative of the likeness between images. When the Hamming distance is smaller between two hash codes, it suggests that the images are more similar. Conversely, a larger Hamming distance implies a lower degree of similarity between the images [48].

3.2.1 Deep Hashing Analysis by Hamming Distance

For a given input image pair, denoted as x_i and x_j, a deep hashing model represented as $\mathscr{H}$ generates hash codes, namely, h_i and h_j. These continuous hash codes are then converted into binary codes, labeled as b_i and b_j, and these binary codes belong to the set $\{-1,1\}^K$, which is achieved by applying a sign operation $\left(b_i = \operatorname{sign}\left(\mathrm{h}_i\right), \mathrm{b}_j = \operatorname{sign}\left(\mathrm{h}_j\right)\right)$, respectively. In simpler terms, this sign operation essentially assigns either -1 or 1 to each element in the continuous hash codes, making them binary.

To facilitate retrieval, a Hamming distance, denoted as $\mathscr{D}_{\mathrm{H}}$, is computed using these binary codes [20] as:

$$\mathscr{D}_{\mathrm{H}}\left(\mathrm{b}_i,\ \mathrm{b}_j\right) = \mathrm{XOR}\left(\mathrm{b}_i,\ \mathrm{b}_j\right),$$

where XOR is a bit-wise count operation. It outputs in the range $[0,\ \mathrm{K}]$. From a mathematical point of view, XOR can be interpreted as [20]:

$$\begin{aligned}
\mathrm{XOR}\left(\mathrm{b}_i,\ \mathrm{b}_j\right) &= \frac{1}{2}\left(K - \mathrm{b}_i^T \cdot \mathrm{b}_j\right) \\
&= \frac{1}{2}\left(K - \|\mathrm{b}_i\|_2 \|_{\mathrm{b}}\|_2\, \delta\left(\mathrm{b}_i,\ \mathrm{b}_j\right)\right) \\
&= \frac{K}{2}\left(1 - \delta\left(\mathrm{b}_i,\ \mathrm{b}_j\right)\right),
\end{aligned}$$

where $\|\mathrm{b}_i\|_2 = \left\|\mathrm{b}_j\right\|_2 = \sqrt{K}$, and $\delta(\cdot)$ denotes cosine similarity. This also can be denoted as [20]:

$$\begin{aligned}\delta\left(\mathrm{b}_i,\ \mathrm{b}_j\right) &= \cos\theta_{ij} \\ &= \frac{\mathrm{b}_i^T \cdot \mathrm{b}_j}{\|\ \mathrm{b}_i\|_2 \left\|\ \mathrm{b}_j\right\|_2},\end{aligned}$$

where the angle between b_i and b_j is θ_{ij}. It is important to highlight that $1-\delta\left(\mathrm{b}_i,\ \mathrm{b}_j\right)$ represents a cosine distance measurement between b_i and b_j. This measurement can be approximated using h_i and h_j, as explained in reference [20]:

$$1-\delta\left(\mathrm{b}_i,\ \mathrm{b}_j\right) \simeq 1-\delta\left(\mathrm{h}_i,\ \mathrm{h}_j\right)$$

Where the term $1-\delta\left(h_i, h_j\right)$ represents the cosine distance between h_i and h_j. Consequently, when we minimize this cosine distance between hash codes during deep hashing training, it leads to a reduction in the Hamming distance between their binary codes.

3.3 Loss Function

A loss function, also known as a cost function, is a mathematical formula used in machine learning to measure how far the predicted values of a model are from the actual values. It essentially quantifies the errors or discrepancies in a model's predictions compared to the true data. In machine learning, the goal is to minimize this loss function to improve a model's accuracy and overall performance.

In the context of Deep Hashing, the loss function serves as a way to measure the difference between the binary codes generated by the hash function and the original high-dimensional data. The optimization process involves minimizing this loss function to create binary codes that effectively capture the underlying structure and similarities within the data. The choice of the loss function is crucial as it significantly impacts the quality of the learned hash function and the overall performance of the Deep Hashing model. Different loss functions are designed to capture various aspects of data, such as similarity, dissimilarity, reconstruction, or adherence to specific constraints. Therefore,

selecting the right loss function is a critical step in the development and implementation of Deep Hashing models.

Over the past few years, there has been a variety of loss functions introduced for Deep Hashing [5, 8, 15, 16, 33, 34, 47], each having its unique strengths and weaknesses. Among these, some commonly used loss functions encompass quantization loss, reconstruction loss, triplet loss, contrastive loss, and generative adversarial loss.

3.4 Machine Learning

Artificial intelligence (AI) has become an integral part of our daily lives, offering assistance in numerous tasks like gaming, email management, social media engagement, navigation, music, movie, and product suggestions, virtual reality experiences, self-driving cars, and even in medical applications. Within the realm of AI, machine learning plays a pivotal role, drawing from mathematical principles, computer science, probability theory, and statistics. Its primary objective revolves around developing computational algorithms capable of discerning complex patterns from input data, paired with their corresponding labels, to subsequently classify and predict new data samples within established categories. Identifying the challenges and future research prospects in machine learning-oriented hardware and software IP security serves as an example of a machine learning application [49]. Machine learning algorithms typically fall into three broad categories: supervised learning, unsupervised learning, and reinforcement learning.

3.4.1 Machine Learning Categories

Supervised Learning is a branch of artificial intelligence and machine learning that relies on labeled datasets to train algorithms for accurate data classification or prediction. It's commonly used for large-scale real-world problems, like sorting spam emails into a separate folder. During training, the model adjusts its parameters as it's exposed to input data until it fits the data correctly. Cross-validation is often used to ensure accurate training for data classification or prediction. Supervised learning is valuable for tackling various large-scale real-world problems, including tasks like filtering spam emails into a dedicated folder [50]. In this study, supervised learning was applied

to train and assess the performance of deep hashing using the PAMAP2 time series dataset.

Unsupervised Learning, sometimes referred to as unsupervised machine learning, involves analyzing and clustering unlabeled datasets using machine learning algorithms. These algorithms can uncover hidden patterns or data groupings without human intervention. Unsupervised learning is particularly useful for exploratory data analysis, cross-selling strategies, customer segmentation, and tasks like image recognition, as it excels at identifying similarities and differences within data [50].

Reinforcement Learning is a subfield of machine learning influenced by behaviorist psychology in humans and animals. In reinforcement learning, agents learn to make decisions to maximize cumulative rewards within a given environment. Different types of evaluative feedback can facilitate this learning process [51]. Reinforcement learning is capable of training fully autonomous agents that interact with their environments, learning optimal behaviors through trial and error [52]."

3.5 Deep Learning

Deep Learning, a subset of machine learning, takes inspiration from the structure and function of the human brain, employing artificial neural networks. It directly learns to perform classification tasks involving images, text, or sound. Deep learning has demonstrated remarkable accuracy, sometimes surpassing human capabilities, setting it apart from other machine learning methods. It finds particular utility in computer vision research, where it can tackle various challenges beyond image retrieval to enhance performance. Deep learning models are trained using extensive labeled datasets, allowing them to automatically learn relevant features from the data, eliminating the need for manual feature extraction.

In Fig. 3.1, you can observe an illustration of a deep neural network, featuring interconnected nodes, an input layer, an output layer, and multiple hidden layers. Deep learning encompasses various neural network architectures, each tailored to specific problem types or datasets. For instance, recurrent neural networks (RNNs) excel in natural language processing and speech recognition, while convolutional neural networks (CNNs) are predominantly used in computer

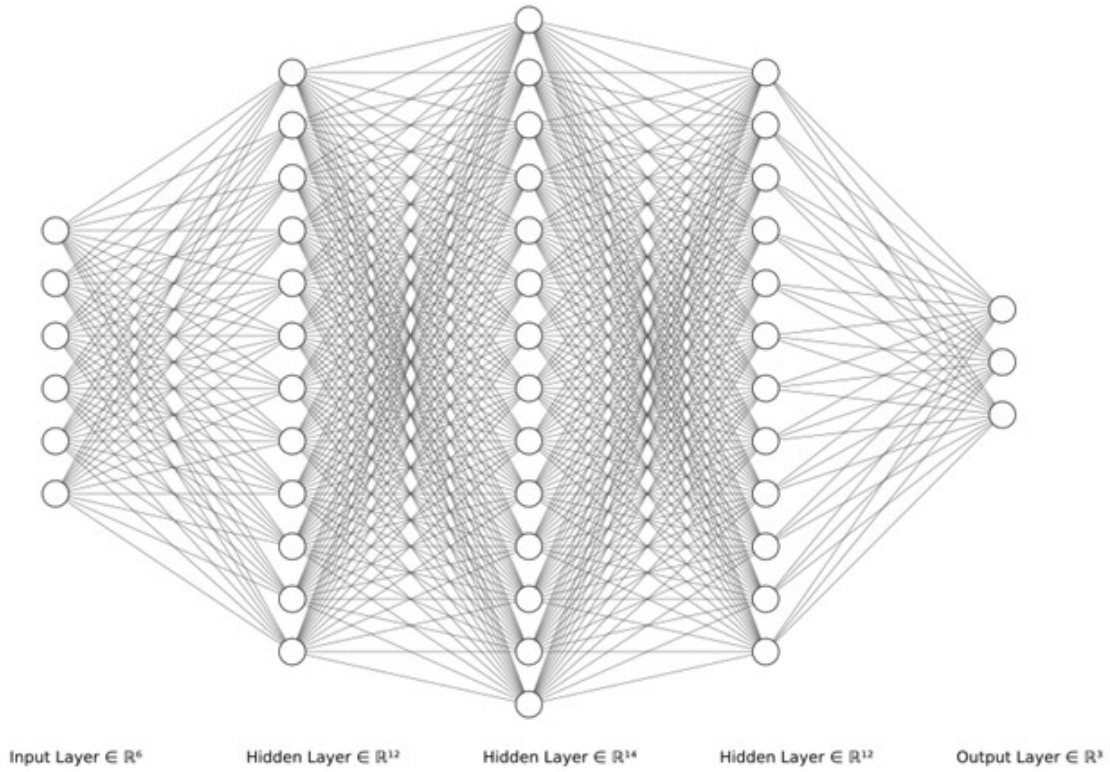

Figure 3.1: Deep Neural Network

vision and classification tasks.

CHAPTER IV

MODEL IMPLEMENTATION

Deep learning models have emerged as powerful tools in the field of artificial intelligence, driving notable advancements across various domains. These models excel at discerning intricate patterns and representations within vast datasets, enabling them to tackle complex tasks with exceptional precision. A key attribute of deep learning is its versatility, allowing it to be applied effectively to a wide range of objects and data types. In this context, deep learning models have found successful applications in various object categories, encompassing images, videos, audio, text, and sensor data. Researchers and practitioners have achieved significant breakthroughs in fields such as computer vision, natural language processing, audio analysis, and sensor data analytics by harnessing the capabilities of these models. Let's delve into each of these five object categories in more detail:

1. Images: Deep learning models have brought about a transformation in the field of computer vision, greatly enhancing their ability to analyze images effectively. Convolutional Neural Networks (CNNs) play a pivotal role in image classification, where they learn to categorize images into predefined classes. Object detection models make use of CNNs to not only identify but also classify multiple objects within an image. Image segmentation models go a step further by assigning labels to individual pixels, providing a highly detailed understanding of the image's content. Furthermore, deep learning models can even generate lifelike images using techniques such as Generative Adversarial Networks (GANs) or variational autoencoders. Images have also proven to be valuable in the domain of data privacy. Utilizing the Advanced Encryption Standard (AES) technique, an image steganography method with three layers of security efficiently enhances both data security and data secrecy [53].

2. Videos: Deep learning models are utilized in the realm of video analysis, facilitating tasks like recognizing actions, adding captions to videos, and creating video summaries. Action recognition models make use of recurrent or 3D convolutional neural networks to identify particular activities or gestures depicted in videos. For example, an iris identification and monitoring system that emphasizes the use of MTCNN, CHT, and morphological approaches achieves significant accuracy in detecting irises from video data [54].

3. Audios: Significant progress has been made in speech and audio analysis using deep learning models. Speech recognition models, such as recurrent neural networks (RNNs) or Transformers, transcribe spoken language into written text. Speaker identification models differentiate between speakers based on voice characteristics, while music classification models can classify songs into genres or identify musical instruments. Audio generation models like WaveNet or SampleRNN can produce realistic-sounding speech or music.

4. Texts: Deep learning models have had a significant impact on various natural language processing (NLP) tasks. Models such as RNNs, Transformers, and their variations are extensively used for tasks like classifying text, analyzing sentiments, identifying named entities, translating languages, generating text, and answering questions. These models can capture complex relationships between words, phrases, and sentences, enabling them to comprehend the meaning and structure of textual data. Pretrained language models like GPT or BERT have achieved top-notch performance in various NLP tasks.

5. Sensor data: The adoption of deep learning models for sensor data analysis is on the rise. These models are applied to process data collected from diverse sources such as IoT devices and environmental monitoring systems. For instance, deep learning models can be employed to detect anomalies in sensor data, which helps identify unusual patterns or outliers that deviate from the expected behavior. Regarding vision-based tracking, the TRM technique was used to address limitations associated with sensor-based response identification, particularly in activity recognition [55]. Activity recognition models can use sensor data like accelerome-

ter readings to identify specific activities or movements. Furthermore, deep learning models are utilized to process sensor data from weather stations for predicting weather conditions and environmental phenomena.

4.1 Dataset

First, a substantial dataset known as the Character Trajectories Dataset was sourced from the UCI Machine Learning Repository. This dataset was created using a WACOM tablet, and each data point includes three dimensions: x-coordinate, y-coordinate, and pen tip force. The data was sampled at a rate of 200Hz and then normalized using the consts.datanorm technique. Only characters with a single 'PEN-DOWN' segment were included in the dataset. In total, there are 2858 samples of characters distributed across 20 different classes.

4.1.1 Training data:

There are total 1422 training data samples with dimension 1422 (Sample size) x 182 (Time Step) x 3 (Features).

4.1.2 Test data:

There are total 1436 test data samples with dimension 1436 (Sample size) x 182 (Time Step) x 3 (Features).

4.1.3 Features:

Features of each character can be stated as Coordinate datapoints (x, y) at each time step and pen tip force.

4.1.4 Multiclass:

There are total 20 class labels. Only characters with a single 'PEN-DOWN' segment were considered.

4.1.5 Data Preprocessing:

This work used character trajectories dataset that contains 1422 train data samples and 1336 test data samples. Then the train dataset is splitted in two parts (80% Training dataset & 20% Validation dataset).

1. Training dataset with size (1137, 182, 3)

2. Validation dataset with size (285, 182, 3)

The training data are exclusively employed to train the model. During each training cycle (epoch), the model learns from the training data by repeatedly processing it. The validation data, on the other hand, serves to assess how well the model is learning. While the model undergoes training with the training data, its performance is evaluated using the validation data at the end of each epoch. If the model exhibits improved performance during this validation phase, checkpoints are saved. The primary purpose of the validation set is to prevent the model from overfitting. Test sets, distinct from both training and validation datasets, are reserved for assessing the model's final performance.

4.2 ResNet34 Basics

Deep Convolutional Neural Networks (CNNs) have brought about significant advancements across various domains by harnessing their multi-layered structure to process intricate patterns in data, mimicking the functioning of the human brain. ResNet34, a prominent example of a deep CNN, is the focus of this section, where we delve into its fundamental aspects, including its architecture, training process, performance, and applications.

ResNet34 falls under the ResNet family of models and was designed to overcome the challenges associated with training extremely deep neural networks. It achieves this by introducing skip connections, also known as residual connections.

The main problem with training very deep neural networks was the vanishing gradient issue, where gradients became extremely small as they propagated back through the network. This made

it challenging for earlier layers to learn meaningful representations, often resulting in performance saturation or degradation with increased depth.

In this section, we will explore key aspects of ResNet34, including its architecture, the use of bottleneck blocks, global average pooling, training techniques like transfer learning, and its remarkable performance.

4.2.1 Architecture:

ResNet34 comprises 34 layers, encompassing convolutional layers, batch normalization layers, ReLU activation functions, and fully connected layers. It commences with a 7x7 convolutional layer featuring 64 filters and a stride of 2, followed by a max-pooling layer with a 3x3 kernel and a stride of 2. The core of the network is formed by 16 residual blocks, each containing two or more convolutional layers alongside batch normalization and ReLU activation.

4.2.2 Bottleneck Blocks:

ResNet34 employs the bottleneck architecture within each residual block. This design incorporates three convolutional layers with smaller kernel sizes, reducing computational demands. Specifically, the bottleneck block includes a 1x1 convolutional layer (to decrease the number of channels), followed by a 3x3 convolutional layer, and another 1x1 convolutional layer (to restore the original number of channels). This design optimizes computational efficiency while preserving the network's representational capacity.

4.2.3 Global Average Pooling:

Following the convolutional layers, ResNet34 employs global average pooling, which transforms the spatial dimensions of the feature map into a 1x1xN tensor, where N represents the number of channels (filters). This reduction in spatial dimensions helps prevent overfitting.

4.2.4 Training and Transfer Learning:

ResNet34 can be pre-trained on extensive image datasets like ImageNet and subsequently fine-tuned for specific image recognition tasks. This transfer learning process enables the model to

leverage its acquired feature representations and adapt to new datasets with fewer data samples and computational resources.

4.2.5 Performance:

ResNet34 has demonstrated impressive performance across various image recognition tasks, including image classification, object detection, and image segmentation. Its depth and the inclusion of skip connections enable it to capture intricate image features, leading to enhanced accuracy compared to shallower models.

Due to its remarkable performance and ease of training, ResNet34 has become a preferred choice for various computer vision tasks and serves as a foundational component for more complex architectures.

ResNet34 is essentially a plain-34 layer convolutional neural network enhanced with skip connections. It comes pre-trained on the ImageNet Database, offering several advantages, such as achieving higher accuracy with limited data and reducing training time. Deep neural networks tend to outperform shallow ones, provided challenges like degradation are effectively addressed, which ResNet34 accomplishes through skip connections. These advantages make ResNet34 an attractive choice for classification tasks, as evident in our proposed work [55].

4.3 Proposed Architecture

In this work, we propose a novel deep hashing model for time series data using ResNet34. Here, every layer of a ResNet is composed of several blocks.

When ResNets are made deeper, they usually do so by increasing the number of operations within a block, which includes things like convolution, batch normalization, and ReLU activation. However, the total number of layers stays the same.

This study introduces a deep hashing model that simplifies the training process by using just one loss function, eliminating the need for adjusting loss weights, which can be quite challenging. As mentioned earlier, training a deep hashing model typically involves two main objectives: making the binary codes distinct and minimizing the quantization error. So, how can we achieve both with

just one loss? The solution lies in the close relationship between these two objectives, which can be combined into a single one. To be more precise, we found that we can meet both goals by maximizing the cosine similarity between the continuous codes and their corresponding binary orthogonal targets. This can be expressed as a cross-entropy (CE) loss.

In Our model, OrthoHash technique has been incorporated which uses only one loss function that maximizes the cosine similarity between the L2-normalized continuous codes and the binary orthogonal targets. This single loss simultaneously enhances the distinguishability of hash codes between different classes and minimizes the quantization error.

To clarify, by maximizing the cosine similarity between the continuous codes and their corresponding binary orthogonal codes, we ensure that the hash codes are both distinctive and minimize quantization errors. Additionally, this learning objective allows us to achieve code balance easily by incorporating a Batch Normalization (BN) layer. It also makes multi-label classification straightforward, including the use of label smoothing. The BN layer contributes to faster training, improved model accuracy, and the ability to utilize higher learning rates.

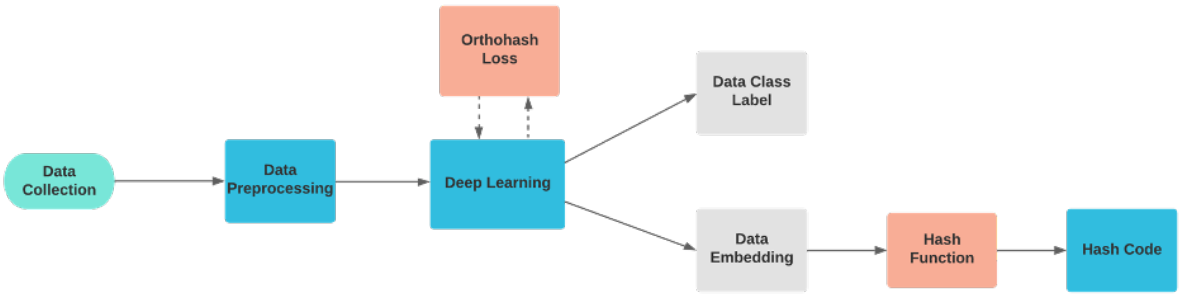

Figure 4.1: Proposed methodology workflow

Fig. 4.1 explains the proposed methodology workflow. We first obtain continuous codes from our backbone network. It is then passed through a batch normalization (BN) layer to obtain zero-mean continuous codes. Next, we compute orthohash loss meaning the cosine similarity between the continuous codes and their binary orthogonal targets. After continuous updates of the loss functions, the model outputs two elements,

1. Data class label (Batch size x number of classes) and

2. Data embedding (Batch size x embedding dimension)

Feeding test data into the trained model outputs two elements.

1. Data class label as the first element that is used to output hash code classification accuracy

2. Data Embedding as the second element to observe the clustering performance

4.4 Orthohash Law

The proposed work used orthohash loss for hash code discriminativeness and quantization error minimization. It reformulated deep hashing in the lens of cosine similarity.

Usually, when converting continuous codes represented by v into binary codes denoted as **b**, there is a loss of information, referred to as quantization error. This quantization error is a common consideration in many existing hashing methods, and it is typically addressed within their learning objectives. The loss function used in Orthohash is as follows [8]:

$$\min L + \lambda Q, \tag{4.1}$$

In the equation provided, L represents the supervised learning objective, such as Cross Entropy, and Q denotes the quantization error between **v** and **b**. Here, λ is used to control the weight of quantization loss. Here, the quantization error is related to the angle θ_{vb} between **v** and **b**. It can be evaluated as follows [8]:

$$\mathrm{Q} = \|\mathbf{v} - \mathbf{b}\|^2 = 2K - 2K\cos\theta_{vb} = 2K\left(1 - \cos\theta_{vb}\right). \tag{4.2}$$

Given that $2K$ remains constant, we can therefore deduce that optimizing the cosine similarity between **v** and **b** will result in a reduced quantization error, thereby yielding a more accurate approximation in the hash codes.

CHAPTER V

EXPERIMENT AND RESULT EVALUATION

In this chapter, we will delve into the examination of the dataset and assess the efficiency and outcomes of different techniques and experiments. We have conducted a comprehensive evaluation of our method's predictive performance, comparing it to several well-established and state-of-the-art alternatives. The effectiveness of the various proposed approaches is evaluated based on the following criteria:

5.1 Evaluation Metrics

Assessing the effectiveness of a Deep Neural Network (DNN) requires the utilization of diverse metrics, which vary depending on the particular task at hand. In the case of classification tasks, typical evaluation metrics encompass:

1. Accuracy: It is a popular metric for evaluating the performance of the classification models. It measures the proportion of correctly classified instances out of all the instances that were classified by the model.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

2. Precision, Recall, and F1-score: Metrics that assess the trade-off between precision (ability to correctly identify positive samples) and recall (ability to capture all positive samples).

3. Area Under the Receiver Operating Characteristic (ROC-AUC): A measure of the classifier's ability to distinguish between classes by plotting the True Positive Rate against the False Positive Rate.

4. Mean Squared Error (MSE): The average of the squared differences between predicted and actual values.

5. Mean Absolute Error (MAE): The average of the absolute differences between predicted and actual values.

These are just a few examples of the evaluation metrics available, and the choice of metric depends on the specific problem and application requirements.

5.2 Experiments Setup

The Character Trajectories dataset is used to train deep learning. The training dataset is divided into two parts: 80% training and 20% validation. The Fig. 5.1 summaries the data selection and splitting process.

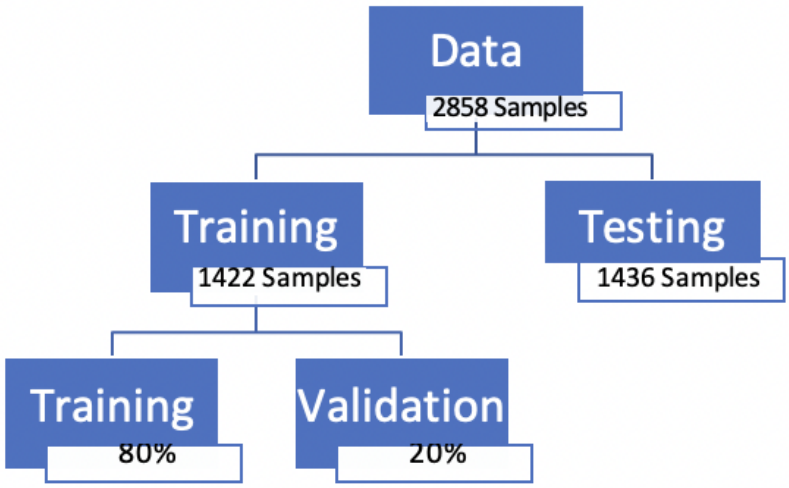

Figure 5.1: Data selection & splitting

Initially, training dataset has a dimension of (1422, 182, 3) where, total number of samples = 1422, time steps = 182, and number of features = 3 (x, y coordinates for each time step, and pen tip pressure)

After splitting, the dimension of the training data becomes (1137, 182, 3) and the dimension of the validation data becomes (285, 182, 3).

5.3 Model Performance

5.3.1 Cross Entropy Loss Function Performance

Adam optimizer with a 0.001 learning rate is used to train the proposed ResNet model. The Cross Entropy loss function was employed, and accuracy was measured during training. The training was conducted three times with 100, 200, and 300 epochs, respectively.

Based on the model training observations, it can be concluded that both the training and validation losses decreased, while the training and validation accuracies increased over the epochs. However, overfitting started after about 13 epochs, as indicated by the rising validation loss and falling validation accuracy. This suggests that the model is not generalizing well and is fitting the training data too closely. At this point, a checkpoint of the model was saved.

Subsequently, the model's performance was tested using the test data and the Accuracy metric. The best training model achieved a classification accuracy of approximately 91.02% on the test data, which is significantly higher than the existing works. The details of the achieved F1 scores by using cross entropy loss are presented in Table 5.1.

Table 5.1: F1 score for different averaging methods

Macro	**Micro**	**Weighted**
0.90%	0.91%	0.91%

The classification performance of the model was visualized using the t-distributed stochastic neighbor embedding (t-SNE) technique. The results were presented in the form of clustering for 20 classes using t-SNE on the test data embedding. Fig. 5.2 illustrates that the trained model forms distinct, dense clusters for the 20 classes, which are significantly different from each other.

To evaluate the model's performance after applying the sign function, the output of the test data was processed by the trained model, and the sign function was applied to the resulting embedding. Fig. 5.3 depicts the model's performance for the 20 classes using t-SNE on the test data embedding with the sign function applied.

It was observed that after applying the sign function to the test data embedding, the data

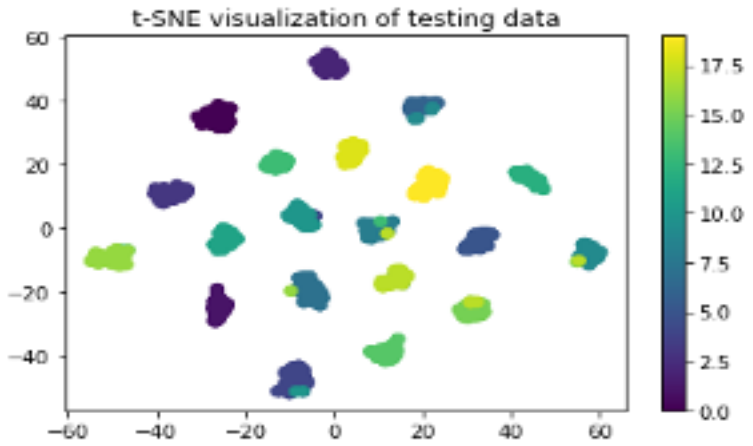

Figure 5.2: t-SNE visualization for test data embedding

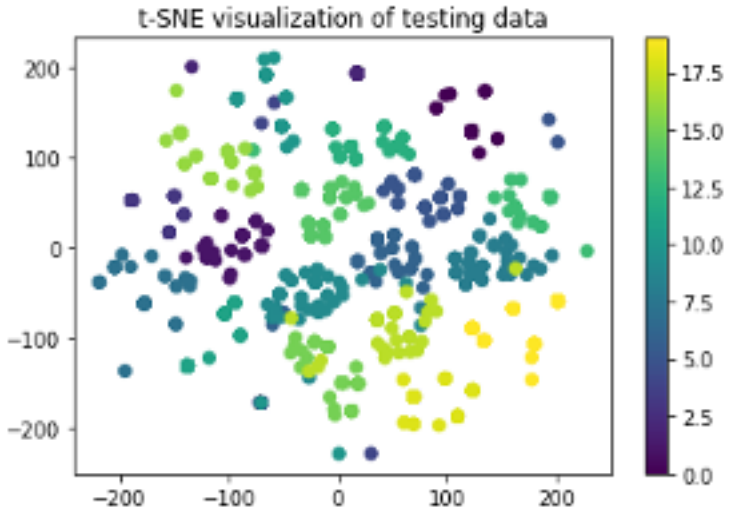

Figure 5.3: t-SNE visualization for test data embedding with sign function applied

points were transformed into hash codes, which resulted in a loss of some information and a weakening of the cluster structure. Consequently, the t-SNE plots showed less distinct clusters for each class. Therefore, the use of the cross-entropy loss function does not yield better hash code outputs for all the classes in this work.

5.3.2 Cosine Similarity Loss Function Performance

After training the proposed ResNet model using the Cross Entropy Loss function, the Cosine similarity loss function was used with the Adam optimizer and a learning rate of 0.001 to further train the model. The Cosine similarity loss function was employed, and accuracy was measured

during training. The training was conducted three times with 100, 200, 300, 500, 1000 and 2000 epochs, respectively.

Based on the model training observations, it can be concluded that both the training and validation losses decreased gradually, while the training and validation accuracies increased slowly over the epochs. At this point, a checkpoint of the model was saved.

Subsequently, the model's performance was tested using the test data and the Accuracy metric. The best training model achieved a classification accuracy of approximately 87.81% on the test data. The details of the achieved F1 scores by using cosine similarity loss function are presented in Table 5.2.

Table 5.2: F1 score for different averaging methods

Macro	**Micro**	**Weighted**
0.87%	0.88%	0.88%

The classification performance of the model was visualized using the t-distributed stochastic neighbor embedding (t-SNE) technique. The results were presented in the form of clustering for 20 classes using t-SNE on the test data embedding. Fig. 5.4 illustrates that the trained model forms distinct, dense clusters for the 20 classes, which are significantly different from each other.

Figure 5.4: t-SNE visualization for test data embedding

To evaluate the model's performance after applying the sign function, the output of the

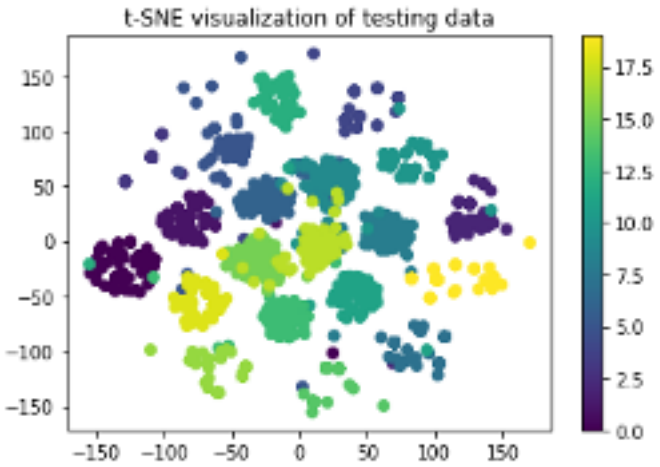

Figure 5.5: t-SNE visualization for test data embedding with sign function applied

test data was processed by the trained model, and the sign function was applied to the resulting embedding. Fig. 5.5 depicts the model's performance for the 20 classes using t-SNE on the test data embedding with the sign function applied.

It was observed that after applying the sign function to the test data embedding, the data points were transformed into hash codes, which resulted in a loss of some information and a weakening of the cluster structure. Consequently, the t-SNE plots showed less distinct clusters for each class and many clusters are overlapping with each other.

However, after comparing Fig. 5.3 and Fig. 5.5, we can conclude that the use of the cosine similarity loss function yields significantly better hash code outputs for all the classes in this work. Therefore, overall cosine similarity loss function generates better hash code outputs in this work.

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

We propose a novel approach to enhance deep hashing training for time series data by introducing a single classification objective, which is called the orthohash loss. This approach focuses on maximizing the cosine similarity between continuous codes and binary orthogonal targets. We have reformulated the deep hashing problem for the character trajectories dataset to emphasize cosine similarity. By applying L2-normalization to continuous codes, we demonstrate that end-to-end training can be achieved without the need for complex constraints. We have also used batch normalization to effectively balance the codes. We tested our method on a character trajectories dataset, achieving an impressive 87.81% accuracy in character identification with just one loss objective. The proposed framework achieves significantly better hash code performance, as evidenced by the t-SNE visualization.

In our future work, we plan to explore ways to improve feature representations for better retrieval performance using hash codes through unsupervised learning. We also aim to enhance hash code performance and incorporate Mean Average Precision (MAP) as a performance metric. Additionally, we intend to conduct detailed analyses of the loss function and the training model to gain deeper insights and make potential refinements.

REFERENCES

[1] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. Supervised hashing for image retrieval via image representation learning. *Proceedings of the National Conference on Artificial Intelligence*, 3:2156–2162, 06 2014.

[2] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. pages 3270–3278, 06 2015.

[3] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2916–2929, 2013.

[4] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing for scalable image search. In *2009 IEEE 12th International Conference on Computer Vision*, pages 2130–2137, 2009.

[5] Mohammad Norouzi and David J. Fleet. Minimal loss hashing for compact binary codes. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 353–360, Madison, WI, USA, 2011. Omnipress.

[6] Mohammad Norouzi, David J Fleet, and Russ R Salakhutdinov. Hamming distance metric learning. *Advances in neural information processing systems*, 25, 2012.

[7] Weihao Kong and Wu-Jun Li. Isotropic hashing. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1646–1654, Red Hook, NY, USA, 2012. Curran Associates Inc.

[8] Jiun Tian Hoe, Kam Woh Ng, Tianyu Zhang, Chee Seng Chan, Yi-Zhe Song, and Tao Xiang. One loss for all: Deep hashing with a single cosine similarity based learning objective, 09 2021.

[9] Huei-Fang Yang, Kevin Lin, and Chu-Song Chen. Supervised learning of semantics-preserving hash via deep convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(2):437–451, 2018.

[10] Chang Zhou, Lai-Man Po, Wilson Y. F. Yuen, Kwok Wai Cheung, Xuyuan Xu, Kin Wai Lau, Yuzhi Zhao, Mengyang Liu, and Peter H. W. Wong. Angular deep supervised hashing for image retrieval. *IEEE Access*, 7:127521–127532, 2019.

[11] Yuming Shen, Jie Qin, Jiaxin Chen, Li Liu, and Fan Zhu. Embarrassingly simple binary representation learning, 08 2019.

[12] Lixin Fan, Kam Woh Ng, Ce Ju, Tianyu Zhang, and Chee Seng Chan. Deep polarized network for supervised learning of accurate binary hashing codes. pages 825–831, 07 2020.

[13] Li Yuan, Tao Wang, Xiaopeng Zhang, Francis EH Tay, Zequn Jie, Wei Liu, and Jiashi Feng. Central similarity quantization for efficient image and video retrieval, 2020.

[14] Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang. Feature learning based deep supervised hashing with pairwise labels. 11 2015.

[15] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S. Yu. Hashnet: Deep learning to hash by continuation. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5609–5618, 2017.

[16] Yue Cao, Mingsheng Long, Bin Liu, and Jianmin Wang. Deep cauchy hashing for hamming space retrieval. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1229–1237, 2018.

[17] Xiaofang Wang, Yi Shi, and Kris M Kitani. Deep supervised hashing with triplet labels. In *Computer Vision–ACCV 2016: 13th Asian Conference on Computer Vision, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part I 13*, pages 70–84. Springer, 2017.

[18] Zhangjie Cao, Ziping Sun, Mingsheng Long, Jianmin Wang, and Philip S Yu. Deep priority hashing. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 1653–1661, 2018.

[19] Fang Zhao, Yongzhen Huang, Liang Wang, and Tieniu Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1556–1564, 2015.

[20] Young Kyun Jang, Geonmo Gu, Byungsoo Ko, Isaac Kang, and Nam Ik Cho. Deep hash distillation for image retrieval, 2022.

[21] Dixian Zhu, Dongjin Song, Yuncong Chen, Cristian Lumezanu, Wei Cheng, Bo Zong, Jingchao Ni, Takehiko Mizoguchi, Tianbao Yang, and Haifeng Chen. Deep unsupervised binary coding networks for multivariate time series retrieval. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(02):1403–1411, Apr. 2020.

[22] Shupeng Su, Chao Zhang, Kai Han, and Yonghong Tian. Greedy hash: Towards fast optimization for accurate hash coding in cnn. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 806–815, Red Hook, NY, USA, 2018. Curran Associates Inc.

[23] Xiangtao Zheng, Yichao Zhang, and Xiaoqiang Lu. Deep balanced discrete hashing for image retrieval. *Neurocomputing*, 403, 04 2020.

[24] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. Deep hashing for compact binary codes learning. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2475–2483, 2015.

[25] Bin Liu, Yue Cao, Mingsheng Long, Jianmin Wang, and Jingdong Wang. Deep triplet quantization. In *Proceedings of the 26th ACM International Conference on Multimedia*, MM '18, page 755–763, New York, NY, USA, 2018. Association for Computing Machinery.

[26] Yunqiang Li and Jan van Gemert. Deep unsupervised image hashing by maximizing bit entropy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2002–2010, 2021.

[27] Dongjin Song, Ning Xia, Wei Cheng, Haifeng Chen, and Dacheng Tao. Deep r -th root of rank supervised joint binary embedding for multivariate time series retrieval. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 2229–2238, New York, NY, USA, 2018. Association for Computing Machinery.

[28] Chenyun Yu, Lintong Luo, Leanne Lai-Hang Chan, Thanawin Rakthanmanon, and Sarana Nutanong. A fast lsh-based similarity search method for multivariate time series. *Information Sciences*, 476:337–356, 2019.

[29] Jwala Dhamala, Emmanuel Azuh, Abdullah Al-Dujaili, Jonathan Rubin, and Una-May O'Reilly. Multivariate time-series similarity assessment via unsupervised representation learning and stratified locality sensitive hashing: Application to early acute hypotensive episode detection, 2018.

[30] Nongmai Inphadung, Suwatchai Kamonsantiroj, and Luepol Pipanmaekaporn. Similarity-preserving hashing for stock analysis. In *Proceedings of the 2019 5th International Conference on E-Business and Applications*, ICEBA 2019, page 94–99, New York, NY, USA, 2019. Association for Computing Machinery.

[31] Zhaoji Fu, Can Wang, Guodong Wei, Wenrui Zhang, Shaofu Du, and Shenda Hong. Hits: Binarizing physiological time series with deep hashing neural network. *Pattern Recognition Letters*, 156:23–28, 2022.

[32] Qi Li, Zhenan Sun, Ran He, and Tieniu Tan. Deep supervised discrete hashing, 2017.

[33] Dayan Wu, Qi Dai, Jing Liu, Bo Li, and Weiping Wang. Deep incremental hashing network for efficient image retrieval. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9061–9069, 2019.

[34] Kevin Lin, Jiwen Lu, Chu-Song Chen, Jie Zhou, and Ming-Ting Sun. Unsupervised deep learning of compact binary descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(6):1501–1514, 2019.

[35] Kamran Ghasedi Dizaji, Feng Zheng, Najmeh Sadoughi Nourabadi, Yanhua Yang, Cheng Deng, and Heng Huang. Unsupervised deep generative adversarial hashing network. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3664–3673, 2018.

[36] Zexuan Qiu, Qinliang Su, Zijing Ou, Jianxing Yu, and Changyou Chen. Unsupervised hashing with contrastive information bottleneck, 2021.

[37] Ruslan Salakhutdinov and Geoff Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In Marina Meila and Xiaotong Shen, editors, *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, volume 2 of *Proceedings of Machine Learning Research*, pages 412–419, San Juan, Puerto Rico, 21–24 Mar 2007. PMLR.

[38] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2916–2929, 2013.

[39] Herve Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

[40] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. Supervised hashing for image retrieval via image representation learning. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, page 2156–2162. AAAI Press, 2014.

[41] Daniel Fraser. A hash based technique for the identification of objectionable imagery. 2014.

[42] H. Lai, Y. Pan, Ye Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3270–3278, Los Alamitos, CA, USA, jun 2015. IEEE Computer Society.

[43] Mingbao Lin, Rongrong Ji, Hong Liu, and Yongjian Wu. Supervised online hashing via hadamard codebook learning. In *Proceedings of the 26th ACM International Conference on Multimedia*, MM '18, page 1635–1643, New York, NY, USA, 2018. Association for Computing Machinery.

[44] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, page 518–529, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[45] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets, 2014.

[46] Jingdong Wang, Ting Zhang, jingkuan song, Nicu Sebe, and Heng Tao Shen. A survey on learning to hash. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):769–790, 2018.

[47] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. Deep supervised hashing for fast image retrieval. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2064–2072, 2016.

[48] Hai Su, Meiyin Han, Junle Liang, Jun Liang, and Songsen Yu. Hard example guided hashing for image retrieval, 2021.

[49] Ashraful Tauhid, Lei Xu, Mostafizur Rahman, and Emmett Tomai. A survey on security analysis of machine learning-oriented hardware and software intellectual property. *High-Confidence Computing*, 3(2):100114, 2023.

[50] IBM. Supervised vs. unsupervised learning: What's the difference? https://citedrive.medium.com/how-to-cite-a-website-in-latex-using-bibtex-and-biblatex-763770172cb5, 2021. 09-12-2023.

[51] Savinay Nagendra, Nikhil Podila, Rashmi Ugarakhod, and Koshy George. Comparison of reinforcement learning algorithms applied to the cart-pole problem. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, sep 2017.

[52] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, nov 2017.

[53] Ashraful Tauhid, Maisha Tasnim, Saima Arifin Noor, Nuruzzaman Faruqui, and Mohammad Abu Yousuf. A secure image steganography using advanced encryption standard and discrete cosine transform. *Journal of Information Security*, 10(3):117–129, 2019.

[54] Mahmudul Hasan Robin, Abu Mohammad Taief, and Md. Minhaz Ur Rahman. A novel approach to detect & track iris for a different and adverse dataset. In *Proceedings of the 2020 4th International Conference on Big Data and Internet of Things*, BDIOT '20, page 24–28, New York, NY, USA, 2020. Association for Computing Machinery.

[55] Md Rahman, Mahmudul Robin, and Abu Taief. A new framework for video-based frequent iris movement analysis towards anomaly observer detection. *International Journal of Image, Graphics and Signal Processing*, 13:13–27, 02 2021.

BIOGRAPHICAL SKETCH

Mahmudul Hasan Robin graduated with a Master of Science in Computer Science from the University of Texas Rio Grande Valley (UTRGV) in December 2023. His academic journey at UTRGV began in the fall of 2021 as a recipient of the highly esteemed Presidential Research Fellowship (PRF) provided by the Graduate College of UTRGV. During this period, he worked as a Graduate Research Assistant (GRA). In his first year at UTRGV, he conducted research using a novel technique to improve deep hashing performance for time series data under the guidance of Dr. Yifeng Gao, an assistant professor of computer science at UTRGV. Alongside his research work, he also served as a Graduate Teaching Assistant for Theory of Computation and Computer Architecture courses in the computer science department.

Upon completing his degree, he plans to join a Doctor of Philosophy in Computer Science program starting in Spring 2024. Robin's research interests include computer vision, data science, data mining, and machine learning.

Robin was born and raised in Bangladesh and earned his Bachelor of Science in Computer Science and Engineering from Ahsanullah University of Science and Technology. Following his successful completion of his BSc, he joined as an AI Engineer at the Cwork Microjob Limited and later served as a Computer Science Lecturer at the Mastermind English Medium School. For further information or assistance, Robin can be reached at mahmudulrobin17@gmail.com.