REPUBLIC OF TÜRKİYE

ALTINBAŞ UNIVERSITY

Institute of Graduate Studies

Information Technologies

IN CAR THEFT SECURITY SYSTEM BY FACE DETECTION

Raoof Hayder Raoof ALTAHER

Master's Thesis

Supervisor

Asst. Prof. Dr. Hakan KOYUNCU

Istanbul, 2023

IN CAR THEFT SECURITY SYSTEM BY FACE DETECTION

Raoof Hayder Raoof ALTAHER

Information Technologies

Master’s Thesis

ALTINBAŞ UNIVERSITY
2023

The thesis titled IN CAR THEFT SECURITY SYSTEM BY FACE DETECTION prepared by RAOOF HAYDER RAOOF ALTAHER and submitted on 22/8/2023 has been **accepted unanimously** for the degree of Master of Science in Information Technologies.

Asst. Prof. Dr Hakan KOYONCU

Supervisor

Thesis Defense Committee Members:

Asst. Prof. Dr Hakan KOYONCU	Department of Computer Engineering, Altınbaş University	_____
Asst. Prof. Dr Abdullahi Abdu IBRAHIM	Department of Computer Engineering, Altınbaş University	_____
Asst. Prof. Dr Warish PATEL	Department of Computer Engineering, Parul University	_____

I hereby declare that this thesis/dissertation meets all format and submission requirements of a Master's thesis.

Submission date of the thesis to Institute of Graduate Studies: ___/___/___

I hereby declare that all information/data presented in this graduation project has been obtained in full accordance with academic rules and ethical conduct. I also declare all unoriginal materials and conclusions have been cited in the text and all references mentioned in the Reference List have been cited in the text, and vice versa as required by the abovementioned rules and conduct.

Raoof Hayder Raoof ALTAHER

Signature

DEDICATION

To the awe-inspiring wonders of the natural world, whose intricate beauty and complex mechanisms have fuelled my curiosity and driven my scientific pursuit. This thesis is dedicated to unravelling your secrets and contributing to the ever-expanding tapestry of human knowledge.

To my beloved family, whose unwavering support and belief in my abilities have been my anchor throughout this scientific endeavour. Your love and encouragement have given me the strength to overcome challenges and reach for new heights.

To my dedicated supervisor, whose guidance and expertise have shaped my research journey. Your mentorship and insightful feedback have been invaluable in refining my ideas and pushing me towards excellence.

May this dedication stand as a testament to the profound impact each of you has had on my scientific voyage. Together, we have embarked on a quest for knowledge and discovery, and your presence in my life has made all the difference.

ABSTRACT

IN CAR THEFT SECURITY SYSTEM BY FACE DETECTION

ALTAHER, Raoof Hayder Raoof
M.Sc., Information Technologies, Altınbaş University,

Supervisor: Asst. Prof. Dr. Hakan KOYUNCU
Date: 08/2023
Pages: 79

This thesis presents an innovative approach to Optimize Neural Networks for the sake of preventing cars thefts by utilizing biologic identification by deep machine learning. The proposed optimization technique leveraging the power of Yogi algorithm [43] adjusted with Nesterov Accelerated Gradient (Nesterov Momentum) [48] and weights averaging [42]. This approach can potentially enhance the performance leading to improved generalization, reduced overfitting, faster and enhanced convergence.

The model employs a neural network architecture consist input layer (120, 120, 3) followed by hidden layers and followed by two prediction heads each one start with a Global Max Pooling with Rectified Linear Units (ReLU) and Dense layer with Sigmoid functions to perform facial Detection.

A Siamese Network consists of three subnetworks trained to preform facial Recognition, this neural network uses a triple loss method, consists of three major components (Embedding layer have three inputs each one followed by a neural network similar to ResNet50 architecture and output of Dense layer 256 units as a feature vector for the that input, A Distance layer subtracting the three Embeddings, And a Classification layer calculated the triplet loss function.

Experiment held using Google Colab and the performance is evaluated through metrics such as regression losses, classification losses and IoU for the face detection neural network. And Loss, Accuracy and other positives and negatives calculations that evaluate the Siamese Neural network.

The results demonstrate the efficacy of the proposed method in terms regression and classifications tasks in Deep neural networks, which can be later used for providing a novel reliable solution for optimizing neural networks.

Keywords: Deep Machine Learning, Optimization, Yogi Algorithm, Nesterov Momentum, Siamese Neural Networks, Triplet Loss, Face Detection.

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

ABBREVIATIONS

AI	:	Artificial Intelligenc
ML	:	Machine Leaning
CNN	:	Convolution Neural Netork
SN	:	Siamese Nework
NAG	:	Nesterov Accelerated Gadient
SWA	:	Stochastic Weights Aveage

1. INTRODUCTION

Theft, a pervasive and distressing crime, has seen an alarming rise in the automobile industry. As vehicles continue to advance technologically, traditional security measures have proven to be inadequate in preventing car theft incidents. To address this pressing issue, researchers have turned to computer vision techniques, specifically face detection, recognition, and verification, leveraging artificial intelligence (AI) and machine learning (ML) algorithms. The next chapter of this thesis is the literature review aims to explore the existing body of work related to in-car anti-theft systems using face detection models implemented by conventional neural networks and face recognition and facial verification models implemented by Siamese neural networks.

Theft, in the context of automobiles, refers to the unlawful act of taking or attempting to take possession of a vehicle without the owner's consent. It involves various techniques, such as lock picking, hot-wiring, and key duplication, which are used to gain unauthorized access to vehicles. The consequences of car theft extend beyond financial losses, including emotional distress and potential harm to individuals. Thus, developing effective security systems is crucial to mitigate the occurrence of such incidents.

Over the years, numerous security systems have been introduced to protect vehicles from theft. These systems encompass a range of technologies, including immobilizers, alarms, and GPS tracking devices. Immobilizers, for instance, aim to prevent unauthorized starting of the vehicle by requiring a specific electronic code or key. Alarms serve as a deterrent, emitting loud sounds or activating lights when unauthorized access is detected. GPS tracking devices provide real-time location information, aiding in the recovery of stolen vehicles. While these systems have shown some effectiveness, they often fall short in addressing sophisticated theft techniques. Therefore, researchers have turned to advanced computer vision techniques to enhance car security.

1.1 BACKGROUND AND LITERATURE REVIEW

The study of face detection, recognition, and verification has a rich history. Earlier approaches focused on handcrafted features, such as local binary patterns (LBP) and scale-invariant feature transform (SIFT), but their performance was limited by the complexity and variability of face images. The advent of deep learning and the availability of large-scale face datasets, such as LFW and MS-Celeb-1M, revolutionized the field by enabling the training of deep neural networks on massive amounts of labelled face data. This led to significant advancements in face detection, recognition, and verification accuracy.

Within the numerous components of deep learning, optimizing network weights after training is a critical performance factor. Stochastic Gradient Descent (SGD) has always been the most widely used method [50]. SGD has several acknowledged shortcomings, including slowly convergence and hyperparameter sensitivity. Adaptive learning rate algorithms like Adam [30] and Yogi [3] were invented to solve these concerns. Adam, a first-order gradient-based for stochastic optimization technique, computes adaptive learning rates for multiple parameters. In fact, it produces excellent results and exceeds other stochastic optimization approaches in a variety of settings. [51]. [52] A new investigation into the convergence of popular adaptive gradient approaches and their application to nonconvex stochastic optimization has been published in Advances in Neural Information Processing Systems. Show that raising the minibatch size can lead to stochastic gradient convergence for these approaches up to the statistical limit of variance. Yogi is a new adaptive optimization technique that regulates the growth in effective learning rate, which leads to significantly greater performance with comparable theoretical convergence guarantees. Experiments indicate that Yogi exceeds Adam in a range of machine learning tasks with little hyperparameter tweaking. “Optimizers in Deep Learning: A Comparative Study and Analysis,” published in the International Journal of Research in Applied Science and Engineering Technology, provides an overview of deep learning optimizers. The research looks into several deep learning optimization approaches and methods, including Stochastic Gradient Descent (SGD), convex and non-convex optimization. The article compares and examines the most famous and widely employed optimizers directly according to similarities, differences, and applicability for a given application. ADAM, AdaMax,

AdaGrad, AdaDelta, RMSProp, and YOGI are examples of recent optimization versions were discussed.

Based on the article, the problems with the algorithm known as Adam is that it may fail to converge even in convex situations once the second moment estimate blows up. Yogi was proposed as a solution. Based on the article, as the number of epochs rises, Yogi loss lowers with rising accuracy, showing its effectiveness in deep learning optimization. [53] demonstrates the potential advantages of using adaptive optimization techniques like YOGI in federated learning and suggests employing adaptive optimization methods like Yogi in federated learning to boost convergence and efficiency. In the context of heterogeneous data, the authors look into the convergence of federated versions of ADAGRAD, ADAM, and YOGI. Show that the application of adaptive optimizers can considerably increase federated learning quality. They find that YOGI outperforms other adaptive optimizers in terms of convergence speed as well as total accuracy.

However, these methods might be plagued by the ‘’curse’’ of local minima, generalization, and convergence. Implementing a simple and successful strategy for training deep neural networks is referred is Stochastic Weight Averaging (SWA), developed by [54] results to improved generality, stability, and dependability. SWA involves maintaining an ongoing average for the weights traversed by stochastic gradient descent (SGD) while training, which leads to flatter solutions and higher generalization than traditional SGD. The researchers show that SWA implements the current Fast Geometric Ensembling (FGE) method with a single model and produces considerable increases in test accuracy when compared to a range of state-of-the-art residual Network approaches. “Stochastic Weight Averaging Revisited” provides a fresh look at how weight averaging may assist in the finding of wider optima and improve neural network generalization. SWA have a greater impact on performance when compared to typical SGD training approaches. By averaging the weights over a number of training cycles, this strategy reduces noisy gradients and results in a more robust final model. Weights averaging has since been proven to increase model generalization ability, hence improving overall deep learning performance [55].

According to [56], training for an extra 12 epochs at cyclical learning rates and averaging the 12 checkpoints as the final model can improve detector performance by roughly 1.0 AP on the challenging COCO benchmark.

Researchers looked into the use of a stochastic weight averaging (SWA) optimizer, [57] The authors used a stochastic weight averaging (SWA) optimizer and the w-softmax loss function to allow the network to learn green plum features accurately while avoiding premature convergence. The system achieved an overall accuracy of 96.5% for the five categories of defects, a significant improvement over traditional method. Proving that the SWA technique enhanced CNN efficiency by decreasing overfitting and increasing generalization capabilities.

[58] The influence of Nesterov's accelerated gradient (NAG) CNNs and deep learning is the main topic of this research. The authors demonstrate the way momentum-based acceleration, particularly the Nesterov type, can significantly reduce the gap in performance between simple stochastic gradient descent (SGD) and more developed optimization methods. They highlight the significance of the momentum constant in achieving optimal results. The tests presented in the study demonstrate that NAG, as an improvement of classical momentum, might effectively accelerate low-curvature directions in a manner comparable to approximation Newton methods.

In the context of in-car anti-theft systems, the integration of face detection, recognition, and verification technologies holds immense promise. By employing a combination of these techniques, it becomes possible to create a comprehensive security system that can not only detect the presence of unauthorized individuals but also identify and verify their identities. This integration enables a more robust and reliable anti-theft solution, enhancing the overall security of vehicles.

Several research works have explored the application of face detection and recognition for in-car security. For instance, [1] proposed a CNN-based face detection model that achieved high accuracy in real-time detection of faces within a vehicle cabin. They utilized a large annotated dataset and fine-tuning techniques to enhance the model's performance. Furthermore, [2] developed a convolution neural network-based face recognition system specifically designed for in-car applications. Their system utilized deep metric learning to learn discriminative face features and achieved high accuracy in detecting faces.

In addition to these specific approaches, there have been notable contributions in related areas. [3] proposed a multi-modal authentication system that integrated face recognition with other biometric modalities, such as fingerprint and voice recognition, for enhanced in-car security. Their system leveraged a Siamese neural network to perform face recognition and achieved robust and reliable authentication.

Moreover, the advancements in hardware, such as the availability of low-cost depth sensors, have enabled the development of 3D face recognition systems.[4] presented a novel approach that combined 3D face reconstruction with Siamese neural networks for in-car facial verification. Their system achieved accurate and robust verification by leveraging the depth information of faces.

These notable works represent a small fraction of the extensive research efforts dedicated to in-car anti-theft systems using face detection, recognition, and verification. The integration of AI, ML, and deep learning techniques has provided new opportunities for developing sophisticated security systems that can protect vehicles effectively. However, further research is still required to address challenges such as occlusion, lighting variations, and real-world deployment constraints.

In conclusion, the literature review has highlighted the significance developments and techniques of face detection, recognition, and verification, that could be employed in-car anti-theft systems. The advancements in AI, ML, and deep learning, along with the development of CNN-based face detection models and SNN-based face recognition and verification models, have paved the way for more robust and reliable biometric verification solutions. The integration of these techniques holds great promise in mitigating car theft incidents and ensuring the safety of vehicle owners. Future research in this domain should focus on addressing real-world challenges and exploring novel approaches to enhance the effectiveness of these systems.

To further enhance the understanding of the development and challenges within the research landscape, it is essential to consider the broader context of biometric verifications, including face detection, recognition, and identity verification. The field of biometrics has made significant progress in providing secure and reliable authentication mechanisms based on an individual’s unique physiological or behavioral characteristics. Facial biometrics, in

particular, have garnered significant attention due to the convenience and non-intrusive nature of face-based identification.

Historically, face detection techniques have evolved from traditional approaches to deep learning-based methods. Early methods, such as the Viola-Jones [5], utilized Haar cascades and classifiers to detect faces in images. However, these methods often struggled with variations in pose, illumination, and occlusion. With the advent of deep learning, convolutional neural networks (CNN) emerged as a powerful tool for accurate face detection. Notable models, such as the Single Shot MultiBox Detector (SSD) [6] and the RetinaNet [7], have demonstrated excellent performance in real-time face detection across various settings.

Face recognition, on the other hand, involves the identification and verification of individuals based on their unique facial features. Traditional face recognition methods, like eigenfaces [8] and Fisherfaces [9], relied on handcrafted features and statistical models. However, these approaches often struggled with complex variations in facial appearance. Deep learning-based face recognition models, particularly SNNs, have emerged as state-of-the-art solutions for accurate and robust face matching. The Siamese architecture, popularized by [10], allows for the learning of similarity metrics between face images, enabling efficient face verification and identification.

The development and advancements in face detection, recognition, and verification techniques have paved the way for the integration of these technologies into biometric verification systems. By leveraging the power of CNNs and SNNs, researchers have been able to develop systems capable of real-time face detection, accurate identification of individuals, and robust verification of their identities. The utilization of these techniques in in-car security not only enhances the level of protection against theft but also provides additional layers of authentication and access control.

In summary, the integration of face detection, recognition, and verification models implemented by CNNs and SNNs has shown great promise in the development of security systems that uses facial verification. The advancements in AI, ML, and deep learning techniques, coupled with the availability of large-scale annotated face datasets, have significantly contributed to the accuracy and reliability of these systems. However,

challenges related to occlusion, lighting conditions, and real-world deployment constraints still require further exploration. Continued research and development in this field will undoubtedly lead to more robust and effective facial security systems, ensuring the safety and protection of vehicles and their owners in case of implementation of this system to preforms in-cars security.

As the field of computer vision and deep learning continues to progress, more accurate and sophisticated methods are being developed for face detection, recognition, and verification. Recent advancements have focused on improving the robustness and reliability of these systems, especially in challenging scenarios such as occlusion, varying lighting conditions, and pose variations.

This study focuses on various works on optimizing deep neural networks and their application. Despite the achievements noted above, there are still research gaps in the efficient optimization methods and techniques, particularly those utilized in face detection and recognition tasks, to increase overall accuracy and model optimizations efficiency. Separate research has been conducted on the Yogi method, SWA Stochastic Weight Averaging and Nesterov Accelerated Gradient. However, in this study, we look at the possible advantages of combining these strategies based on Yogi algorithm to enhance these neural networks trained from scratch on augmented datasets and used in face detection and recognition tasks.

1.2 PROBLEM STATEMENT

Despite significant advances in optimization algorithms, specifically traditional ones such Adam and its variants, which are widely used in deep learning applications, certain limitations and drawbacks have become increasingly apparent. Adam, an acronym for Adaptive Moment Estimation, combines the benefits of two gradient descent methodologies: Root Mean Square Propagation (RMSProp) along with Stochastic Gradient Descent with Momentum. It scales learning rate with squared gradients and takes advantage of momentum by dampening oscillations with moving averages of the parameters, resulting in faster convergence. However, these advantages can sometimes turn into disadvantages. Adam's adaptive learning rate can sometimes result in unwanted, premature convergence, resulting in suboptimal solutions. This premature convergence has been attributed to Adam's

exponential moving averages' inability to control the learning rate adequately. Furthermore, its reliance on the initial learning rate setting may result in poor performance if not chosen carefully. Yogi, an improved variant of Adam, additionally prevents the premature convergence problem by employing a more cautious adaptive learning rate approach. Nonetheless, even with this improvement, it does not completely eliminate Adam's drawbacks and retains its reliance on the initial learning rate setting.

As a result, there is a need to investigate alternative optimization techniques that can improve the performance of these optimizers, searching for any potential enhancements could be obtained by cooperating different techniques along with optimizers. Nesterov Momentum, a momentum variant with stronger theoretical converge guarantees for convex functions, and Weight Averaging, a strategy that averages the model's weights over epochs, could both potentially provide such an improvement. Nesterov Accelerated Gradient or Nesterov Momentum, is a technique that aids in the acceleration of gradient vectors in the correct directions, resulting in faster convergence. It has been demonstrated to significantly boost the rate of convergence, particularly when a model gets stuck in a steep, narrow ravine of the optimization space. Weight Averaging, on the other hand, is a technique that computes the average of different iterations of a model's weights, resulting in a model with better generalization performance. It is especially effective at reducing variance in model parameters during training and improving generalization robustness.

The goal is to create a robust and reliable optimization method that could be used in system or deep learning model that can detect the presence of unauthorized individuals within the vehicle cabin, by leveraging the benefits of Nesterov Momentum in creating a lookahead feature to the Yogi optimizer along with a SWA weights average technique, incorporating these two techniques – Nesterov Momentum and Weights Average – into the Yogi optimizer, the proposed method aims to address the optimizer's inherent limitations, leading to improved optimizer performance, faster convergence rates, and ultimately more effective deep learning models. Which might be used in car security and mitigate the risks associated with theft. The challenges include developing accurate and efficient face detection models, designing effective face recognition and verification systems using neural networks, and addressing real-world constraints such as occlusion, lighting variations, and resource limitations. The outcome of this research will contribute to the advancement of cooperating

optimizers and different techniques in deep neural networks, developed for facial detection and recognition tasks.

1.3 THESIS OBJECTIVES

The primary objectives of this thesis are as follows:

a. To develop a deep learning model utilizing face detection and recognition tasks: The first objective is to design and implement a comprehensive system that can detect the presence of faces. This involves developing an accurate and efficient face detection and face recognition model using deep neural networks.
b. To optimize the neural network models for enhanced performance: The second objective is to search about an optimization techniques or algorithms which matches and works perfectly with the thesis problem and apply these optimization techniques to the neural network models used in the system. This involves exploring and implementing various optimization algorithms, fine-tuning model parameters, and employing regularization techniques to improve the efficiency, accuracy, and generalization capability of the models. The objective is to enhance the overall performance and effectiveness of the anti-theft system through optimization methods.
c. To address real-world challenges and constraints: The third objective is to tackle practical challenges associated with in-car anti-theft systems. This includes addressing any issues such as occlusion, varying lighting conditions, and resource limitations, which can affect the accuracy and reliability of the system. The objective is to develop strategies and methodologies to mitigate these challenges and ensure the system's robustness and applicability in real-world scenarios.
d. To evaluate the performance and effectiveness of the developed system: The final objective is to comprehensively evaluate the performance and effectiveness of the developed method against using the optimizer alone without additional techniques. This involves conducting extensive experiments using the datasets and different optimization scenarios to investigate potential enhancements on system's accuracy, detection rates, and overall capabilities. The objective is to provide empirical evidence of the proposed methods.

By achieving these objectives, this thesis aims to contribute to the advancement of face detection, and recognition models implemented through deep neural networks and optimized by Yogi algorithm with NAG (Nesterov Accelerated Gradient) and SWA (Stochastic Weights Averaging) techniques. The results and findings of this research will provide insights into the development of more robust and reliable in-car anti-theft systems, ensuring the safety and protection of vehicles and their owners.

1.4 THESIS ORGANIZATION

This thesis part provides guidance to the reader. Section 2 will cover general concepts about the research. Section 3 will go over the proposed materials and methods and the essential components used in our models and optimization techniques and describe our methods in details which we implemented in our study, Section 4 will discuss the results of proposed methods and the, while Section 5 will include the conclusion, Section 6 is a discussion and Section 7 is the future work and recommendations.

2. CONCEPTS AND FUNDAMENTALS

2.1 ARTIFICIAL INTELLIGENCE

Artificial Intelligence (AI) refers to the field of computer science dedicated to creating intelligent machines that can perform tasks requiring human-like intelligence [11]. It encompasses a broad range of techniques and methodologies aimed at enabling computers to reason, learn, perceive, and interact with the environment. The concept of AI has a rich history, dating back to the 1950s when pioneers such as Alan Turing and John McCarthy laid the foundation for the field [12]. Over the years, AI has evolved with advancements in computational power, algorithms, and data availability. From rule-based systems and expert systems to statistical methods and machine learning, AI has witnessed significant progress.

2.2 MACHINE LEARNING

Machine Learning (ML) is a subset of AI focused on developing algorithms and models that enable computers to learn from data and make predictions or decisions without being explicitly programmed. ML algorithms learn patterns and relationships from labelled or unlabelled data [13], allowing machines to generalize and make informed decisions based on new, unseen examples. The history of ML can be traced back to the development of computational learning theories and statistical pattern recognition [14]. It gained significant traction in the 1990s with advancements in algorithms like support vector machines and decision trees. More recently, the availability of large-scale datasets, computational resources, and deep learning techniques has propelled ML to new heights.

2.3 DEEP LEARNING

Deep Learning is a subfield of ML that focuses on training artificial neural networks with multiple layers to learn complex patterns and representations from raw data. The history of deep learning can be traced back to the early development of artificial neural networks [15]. However, it wasn’t until the 2000s, with advancements in computational power and backpropagation algorithms, that deep learning gained significant attention. Breakthroughs like convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have propelled deep learning to the forefront of AI research. Deep learning has revolutionized

various domains [16], including computer vision, natural language processing, and speech recognition, achieving state-of-the-art results in many tasks.

2.4 NEURAL NETWORKS

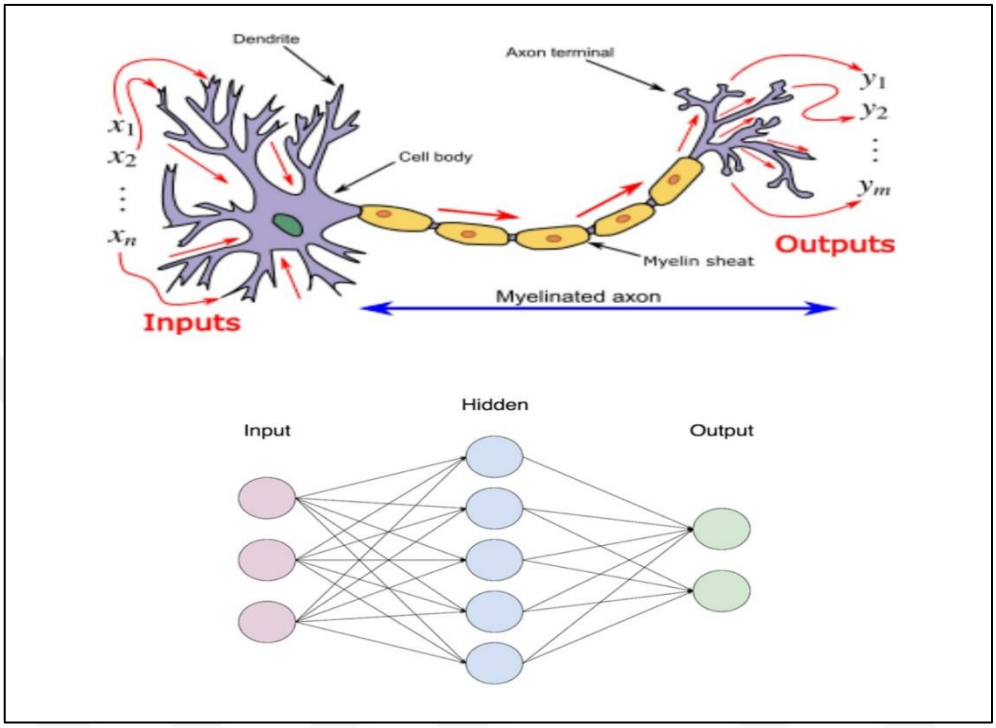

Figure 2.1: Human Brain Neural Network and Artificial Neural Network.

Neural networks are the foundation of many modern machine learning and deep learning models. They are inspired by the structure and function of the human brain, consisting of interconnected nodes or artificial neurons organized in layers. Neural networks can be categorized into different types based on their architectural characteristics and connectivity patterns.

One commonly used type is the feedforward neural network, where information flows in a unidirectional manner from input nodes through hidden layers to output nodes. This type of network is often used for tasks such as pattern recognition and classification.

Convolutional Neural Networks (CNNs) are another important type, particularly well-suited for computer vision tasks. CNNs leverage convolutional layers to automatically learn and extract meaningful features from input data. This type of network has revolutionized image classification, object detection, and image segmentation tasks [17].

Recurrent Neural Networks (RNNs) are designed for sequential data processing, where the output of each step is fed back as input to the next step. RNNs are effective in handling sequential dependencies and are widely used in natural language processing, speech recognition, and time series analysis [18].

Furthermore, there are variations of neural networks that address specific challenges or incorporate specialized architectural elements. For instance, Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) are types of RNNs that address the vanishing gradient problem and improve the learning of long-term dependencies [19].

Additionally, there are more advanced network architectures such as Generative Adversarial Networks (GANs), which consist of a generator and a discriminator network competing against each other to generate realistic synthetic data [20].

The field of neural networks is constantly evolving, with researchers continuously exploring novel architectural designs, activation functions, and connectivity patterns. The selection of a specific neural network type depends on the task at hand, the characteristics of the input data, and the desired outcomes. By leveraging different types of neural networks, researchers and practitioners can tackle a wide range of problems across various domains, including image recognition, natural language understanding, speech processing, and more. The diversity of neural network types allows for the flexibility and adaptability required to address complex real-world challenges.

Overall, understanding the various types of neural networks and their respective applications is crucial for designing effective and efficient solutions in the field of artificial intelligence and machine learning.

2.5 CONVOLUTION NEURAL NETWORKS

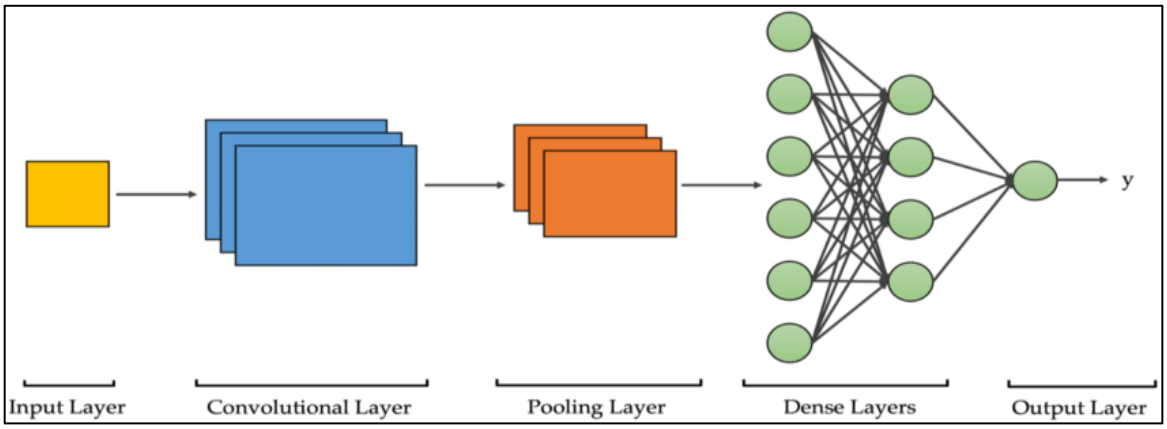

Figure 2.2: Example of Convolution Neural Networks.

Convolutional Neural Networks (CNNs) have emerged as a powerful architecture for various computer vision tasks. CNNs leverage their ability to automatically learn hierarchical features from input data, making them highly effective in tasks such as image classification, object detection, and image segmentation. The success of CNNs can be attributed to their unique architectural design, which includes convolutional layers, pooling layers, and fully connected layers. Convolutional layers employ filters to extract local patterns and spatial hierarchies, while pooling layers down sample feature maps to capture the most salient information. These operations allow CNNs to learn robust representations of images, enabling superior performance in visual recognition tasks. The seminal work of [21] introduced the concept of CNNs and their application in handwritten digit recognition, paving the way for significant advancements in computer vision. The development of deep CNNs, such as AlexNet [22] and ResNet [23], further propelled the field, achieving breakthrough performance in large-scale image classification competitions. CNNs continue to evolve, with ongoing research focusing on improvements in architectural design, optimization methods, and interpretability.

2.6 SIAMESE NEURAL NETWORKS

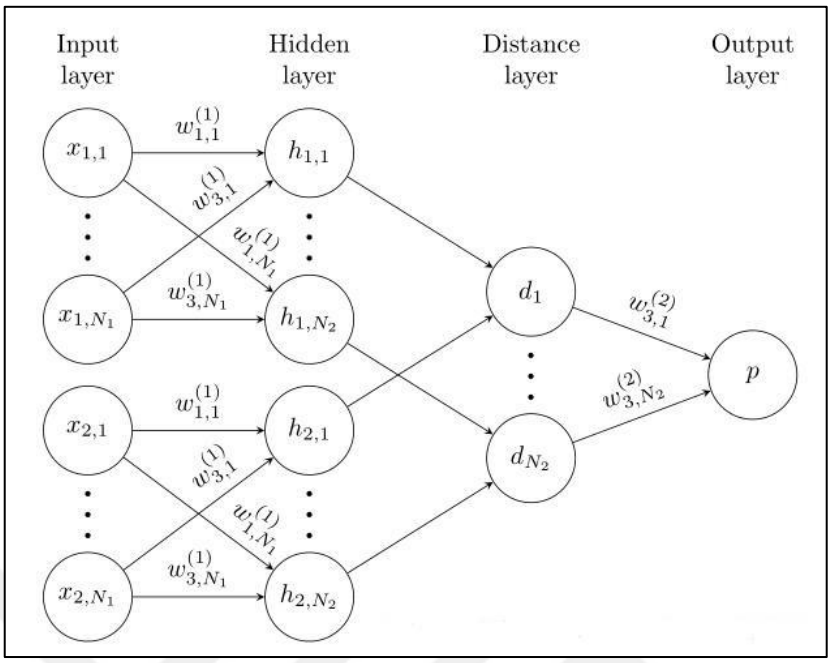

Figure 2.3: Example of Siamese Neural Networks.

Siamese neural networks have gained prominence in various tasks involving similarity or dissimilarity measurements, such as face recognition, signature verification, and object tracking. Siamese networks consist of twin subnetworks that share weights and architectures. These networks learn representations by comparing pairs of inputs, emphasizing similarities and differences. Siamese networks are particularly effective in scenarios with limited unlabelled data, as they enable learning from pairs rather than relying on large-scale labelled datasets.[24] introduced the concept of Siamese networks and proposed their application in signature verification, achieving promising results. Siamese networks have since been widely adopted, with advances such as the introduction of contrastive loss by [25] to further improve the learning process. Recent research has extended the Siamese architecture with attention mechanisms and deep metric learning techniques to enhance performance and address challenges such as intra-class variations and occlusions.

2.7 OPTIMIZATION IN DEEP LEARNING

Optimization, a fundamental aspect of many scientific and engineering disciplines, is the process of selecting the best solution from a set of available alternatives. In mathematics and computational science, this is typically accomplished by systematically selecting input

values from an allowed set and computing the value of a function. Convex and nonconvex optimization problems are distinguished by the properties of the function to be optimized.

2.7.1 Convex Optimization

The minimization (or maximization) for a convex function over a convex set is the focus of convex optimization problems. Convex functions have the property that any line segment connecting two points on the function lies above or on the graph, resulting in a “bowl” shape. Any local minimum is also a global minimum in convex optimization problems. Linear Programming (LP), in which both the objective function and the constraints are linear, Quadratic Programming (QP), in which the objective function is quadratic and the constraints are linear, and Second Order Cone Programming (SOCP), a generalization of LP and QP problems, are examples of convex optimization problems. Convex optimization problems are simpler to solve, and algorithms with strong theoretical guarantees exist for them. [59].

2.7.2 Nonconvex Optimization

Nonconvex Optimization problems, on the other hand, involve the minimization (or maximization) of a nonconvex function that doesn’t meet the convexity properties. Nonconvex functions may have multiple local optima, making determining the global optimum difficult. Nonconvex optimization problems are subclassed as Mixed-Integer Programming (MIP), Global Optimization, and Nonlinear Programming (NLP). The variety and complexity in nonconvex problems frequently necessitate sophisticated techniques and heuristics to obtain satisfactory solutions [60].

Optimization in Deep Learning is more complex and typically nonconvex due to the presence of numerous local minima and saddle points. Despite these obstacles, many optimization algorithms for deep learning have been developed, including:

Gradient Descent (GD): A basic type of optimization algorithm that adjusts the model’s parameters along the steepest descent direction. Batch GD (which uses the entire dataset to compute the gradient at each step), Stochastic GD (which uses only one example at each step), and Mini-batch GD (which uses a mini-batch of n examples at each step) are examples of variants.

Momentum: An improvement over GD in which an exponentially decaying moving average of previous gradients is accumulated and continues to move in the same direction.

Nesterov Accelerated Gradient (NAG): An enhancement to Momentum that lowers oscillations in gradient directions and has a faster convergence rate.

Adagrad: Adjusts the learning rate based on the parameters, with smaller updates for parameters associated with frequently occurring features and larger updates for parameters associated with infrequent features.

RMSprop: Improves Adagrad's performance in non-convex environments by converting the gradient accumulation to an exponentially weighted moving average.

Adam (Adaptive Moment Estimation): Combines the concepts of Momentum and RMSprop; it computes an exponentially decaying average of past gradients, similar to Momentum, and an exponentially decaying average of past squared gradients, similar to RMSprop.

Each of these algorithms has advantages and disadvantages, and the optimizer used can have a significant impact on the speed of training and the overall performance of a deep learning model. The paper [61] presents a comprehensive review and empirical comparison of these optimizers.

Adamax: An Adam variant based on the infinity norm.
Nadam: Another Adam variant that includes Nesterov Momentum.
AdaMax: An Adam extension, AdaMax tends to be more stable than the original Adam optimizer because it uses the L-infinity norm of parameter updates instead of the L2 norm, which makes it well-suited for tasks with very sparse gradients.

Adadelta: An Adagrad extension, Adadelta lowers its aggressive, monotonically decreasing learning rate by limiting the window of accumulated past gradients to a fixed size, eliminating the need to set a default learning rate.

AMSGrad: AMSGrad, introduced as an enhancement over Adam, revises Adam's step size calculation to provide a non-increasing step size. This was added to improve Adam's convergence properties and algorithmic performance in specific settings.

These various optimizers have distinct advantages and are appropriate for a variety of tasks and datasets. Variations and hybrid approaches to these techniques are also being developed, adding to the large family of deep learning optimization algorithms [62][63].

In the context of deep learning, these optimizers seek to minimize a loss function by adjusting the parameters of neural networks. While most of these optimizers use gradient descent due to its computational efficiency, they each have their own techniques for overcoming common optimization challenges such as avoiding local minima, dealing with sparse gradients, and maintaining fast and stable convergence.

Optimizing neural networks plays a vital role in improving the performance of face detection and recognition systems. Numerous techniques have been developed to enhance the efficiency and accuracy of these models. One commonly used approach is network architecture optimization, which involves designing more effective neural network architectures specifically tailored for face-related tasks. For instance, [26] proposed the "MobileNet" architecture, which leverages depth-wise separable convolutions to reduce the computational complexity without sacrificing accuracy.

Another important aspect of optimization is parameter tuning, which involves fine-tuning the network's parameters to achieve better performance. Researchers have explored techniques such as gradient-based optimization algorithms, including stochastic gradient descent (SGD) [27] and its variants like Adam [28], to optimize the network parameters. These methods adjust the weights and biases of the neural network iteratively based on the gradients computed during the training process, aiming to minimize the loss function.

To address the challenge of overfitting, regularization techniques have been widely employed. Dropout [29] and batch normalization [30] are commonly used regularization techniques that help prevent overfitting by reducing the model's reliance on specific neurons and normalizing the activations, respectively. These techniques have proven effective in improving the generalization capability of face detection and recognition models.

Additionally, data augmentation has emerged as a powerful tool for improving the performance of neural networks. By applying various transformations such as rotation, scaling, and flipping to the training data, augmented datasets can be generated, which helps the network learn robust and invariant features. Data augmentation techniques have been

widely employed in face-related tasks, including face detection and recognition, to improve model generalization and robustness.

Stochastic optimization methods also have a rich history and have played a significant role in training deep neural networks for face detection and recognition. One of the earliest stochastic optimization algorithms is stochastic gradient descent (SGD), which dates back to the 1960s [31]. SGD updates the network parameters using randomly sampled mini-batches of training data, making it computationally efficient and suitable for large-scale datasets. However, SGD can suffer from slow convergence and oscillation around the optimal solution.

To address these limitations, researchers have proposed various enhancements to SGD. For instance, the AdaGrad algorithm [27] adapts the learning rate for each parameter based on their historical gradients, allowing for faster convergence on sparse data. Another notable method is the Adam optimizer [30], which combines RMSprop with momentum-based updates.

Furthermore, the field of stochastic optimization has witnessed advancements in regularization techniques. One prominent approach is dropout, introduced by [29]. Dropout randomly sets a fraction of neurons to zero during each training iteration, preventing co-adaptation of neurons and improving model generalization. This technique has been widely adopted in face-related tasks to combat overfitting.

The optimization of neural networks for face detection and recognition involves various techniques and methods. Optimization methods, such as network architecture optimization, parameter tuning, regularization techniques and other techniques, contribute to improving the accuracy and robustness of these models. ADAM algorithm, and its variants, have played a significant role in training deep neural networks. These techniques have evolved over time, addressing challenges such as slow convergence and overfitting, and have contributed to the success of deep neural networks and its applications such face detection and recognition models.

These studies provide a comprehensive understanding of the optimization techniques employed in deep learning. The cited works showcase the historical development of stochastic optimization algorithms, including stochastic gradient descent, AdaGrad, and

Adam and its variants. Furthermore, they highlight the significance of regularization techniques such as dropout and batch normalization and Average Weighting in enhancing the generalization capability of neural networks. Additionally, the importance of network architecture optimization and data augmentation methods is exemplified, as seen in the proposal of the MobileNet architecture and the use of data augmentation for robust training. These optimization approaches have significantly contributed to the advancements in face detection and recognition systems, enabling improved accuracy and reliability in real-world scenarios.

These optimization techniques, along with advancements in regularization and activation functions, have significantly improved the training process of neural networks. They have enabled the training of deeper and more complex architectures, such as deep convolutional neural networks (CNN) and recurrent neural networks (RNN), leading to breakthroughs in various domains, including computer vision and natural language processing.

2.8 FACE DETECTION

Real-time face detection refers to the ability of a computer system or algorithm to detect and localize human faces in a continuous stream of data or video in real-time. The goal is to identify the presence and location of faces in a timely manner, typically within a few milliseconds, allowing for immediate and responsive processing. Real-time face detection algorithms analyze the visual information captured by cameras or sensors and employ advanced techniques to recognize facial features and patterns, enabling applications such as face recognition, emotion analysis, and gaze tracking.

2.8.1 History

The history of real-time face detection dates back to the early advancements in computer vision and machine learning. [5] introduced a landmark face detection framework based on cascaded classifiers, known as the Viola-Jones algorithm. This algorithm employed Haar-like features and AdaBoost to achieve high detection rates while maintaining real-time performance. Since then, numerous research efforts have focused on improving the accuracy and efficiency of real-time face detection. The development of deep learning techniques and convolutional neural networks (CNN) has led to significant breakthroughs in face detection performance.

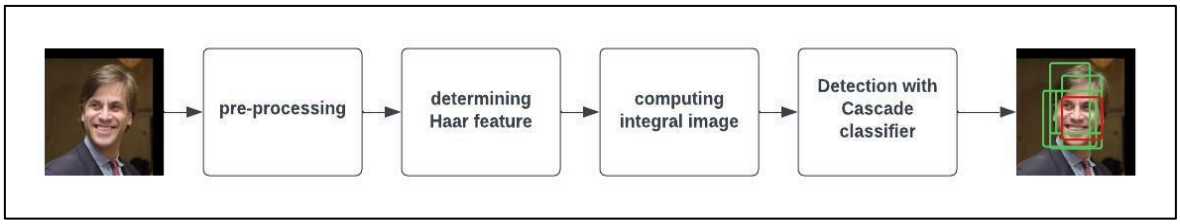

Figure 2.4: Face Detection Process With Viola-Jones

2.8.2 Face Detection Methods

Real-time face detection methods employ a range of techniques, including classical computer vision algorithms and deep learning approaches. Classical methods often involve feature extraction techniques, such as Haar-like features, local binary patterns (LBP), or histogram of oriented gradients (HOG), followed by classifiers like Support Vector Machines (SVM) or AdaBoost. These methods provide reasonably accurate results and can be computationally efficient.

Face detection includes the process of automatically locating and localizing human faces within images or video streams. It involves the detection and identification of facial regions or regions of interest (ROIs) that contain faces. Face detection typically encompasses the following key steps:

a. Pre-processing: This involves preparing the input image or video frame for face detection. It may include operations such as resizing, normalization, and noise reduction to enhance the quality of the image and improve the accuracy of the detection process.

b. Feature Extraction: relevant features or patterns associated with faces are extracted from the image or video frame. Traditional methods may use handcrafted features like Haar-like features, local binary patterns (LBP), or histogram of oriented gradients (HOG). More recent approaches leverage deep learning techniques to automatically learn discriminative features using convolutional neural networks (CNNs).

c. Detection: The detection step involves applying a face detection algorithm or model to identify potential face regions in the image or video frame. This is typically done by analyzing the extracted features and using classification or regression models to determine if a given region contains a face or not.

d. Localization: Once a potential face region is detected, the algorithm determines the precise bounding box or coordinates that enclose the face within the image or video frame. This step aims to accurately localize the face within the detected region.

In recent years, deep learning-based approaches, particularly CNNs, have demonstrated remarkable performance in real-time face detection. Methods like Single Shot MultiBox Detector (SSD) [32] and You Only Look Once (YOLO) [33] have gained popularity for their efficiency and accuracy. These methods employ CNNs to directly predict bounding boxes and class probabilities, enabling fast and precise face detection in real-time applications.

2.8.3 Convolution Neural Networks For Face Detection

Convolutional Neural Networks (CNNs) have shown significant success in real-time face detection tasks. CNNs are capable of automatically learning hierarchical representations from images, allowing them to capture intricate facial features and patterns. Face detection CNNs typically consist of multiple convolutional and pooling layers, followed by fully connected layers for classification and bounding box regression.

Notable CNN-based face detection methods include Multi-task Cascaded Convolutional Networks (MTCNN) [34] and the Single Shot MultiBox Detector (SSD) [32]. MTCNN employs a cascaded structure of CNNs for face detection, landmark localization, and bounding box regression. SSD, on the other hand, uses a single network to directly predict face locations and class probabilities.

These CNN-based methods have significantly advanced real-time face detection, providing high accuracy and efficiency. They have enabled a wide range of applications in areas such as surveillance, biometrics, and human-computer interaction.

2.8.4 Challenges With Face Detection

Real-time face detection faces several challenges, including occlusions, variations in lighting conditions, pose variations, and complex backgrounds. Occlusions, such as partial face obstruction or the presence of accessories, can hinder accurate detection. Variations in lighting conditions, such as shadows or varying intensities, can impact the performance of detection algorithms. Additionally, variations in pose and facial expressions pose challenges,

as the appearance of the face changes with different angles and expressions. Addressing these challenges requires robust algorithms that can handle diverse real-world scenarios.

2.9 FACE RECOGNITION

Real-time face recognition refers to the capability of identifying and verifying individuals in real-time by analyzing and comparing their facial features against a database of known faces. It involves capturing and processing live video feeds or streams to instantly recognize and match the identity of individuals. Real-time face recognition systems employ advanced algorithms and techniques to extract facial features, measure similarity or distance metrics, and perform rapid identification or verification, enabling seamless and efficient authentication in various applications, including surveillance, access control, and personalized user experiences.

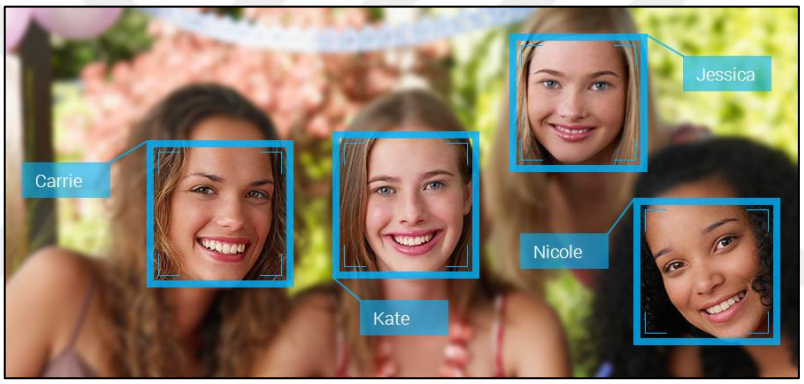

Figure 2.5: Face Recognition.

2.9.1 History

Face recognition has witnessed significant advancements with the progress in computer vision and pattern recognition. The introduction of deep learning methods, particularly convolutional neural networks (CNNs), has revolutionized the field. Notably, the FaceNet model proposed by [35] made significant strides by training a CNN to learn discriminative face embeddings, enabling real-time and accurate face recognition. The DeepFace system developed by [36] also made notable contributions, leveraging deep CNN architectures for face verification and achieving impressive performance. These advancements have paved the way for practical real-time face recognition systems with high accuracy and efficiency.

2.9.2 Face Recognition Methods

Deep learning-based approaches have emerged as the dominant methods for real-time face recognition. One notable example is the ArcFace algorithm introduced by [37], which employs a novel angular margin-based loss function to enhance discriminative face embeddings. ArcFace has shown remarkable performance in real-time face recognition tasks, achieving state-of-the-art accuracy.

Another method is the LightCNN model proposed by [38], which focuses on learning compact yet discriminative face representations. LightCNN achieves real-time face recognition by utilizing carefully design.

Face recognition includes the following steps:

a. Face Detection: The first step involves locating and extracting the face region within an image or video frame. Face detection algorithms are employed to identify the presence and location of faces, often using techniques such as Haar cascades, convolutional neural networks (CNNs), or deep learning-based models.
b. Feature Extraction: Once the face region is detected, relevant facial features are extracted to create a representation that captures the distinguishing characteristics of the face. These features can include the shape, texture, and spatial relationships of facial components, such as eyes, nose, and mouth. Feature extraction methods may employ techniques like Principal Component Analysis (PCA), Local Binary Patterns (LBP), or deep learning-based approaches to obtain discriminative and compact representations of faces.
c. Feature Matching and Comparison: In this step, the extracted facial features are compared with a database of known faces to find potential matches. The matching process involves measuring the similarity or dissimilarity between the extracted features and the features stored in the database. Various similarity metrics, such as Euclidean distance, cosine similarity, or Mahalanobis distance, can be employed for this purpose.
d. Recognition and Decision Making: Based on the comparison results, a decision is made regarding the identity of the individual. If the extracted features closely match the features in the database above a certain threshold, the person is recognized as a known individual. If there is no significant match, the person may be classified as unknown or

further verification steps, such as additional biometric measures or human intervention, may be required.

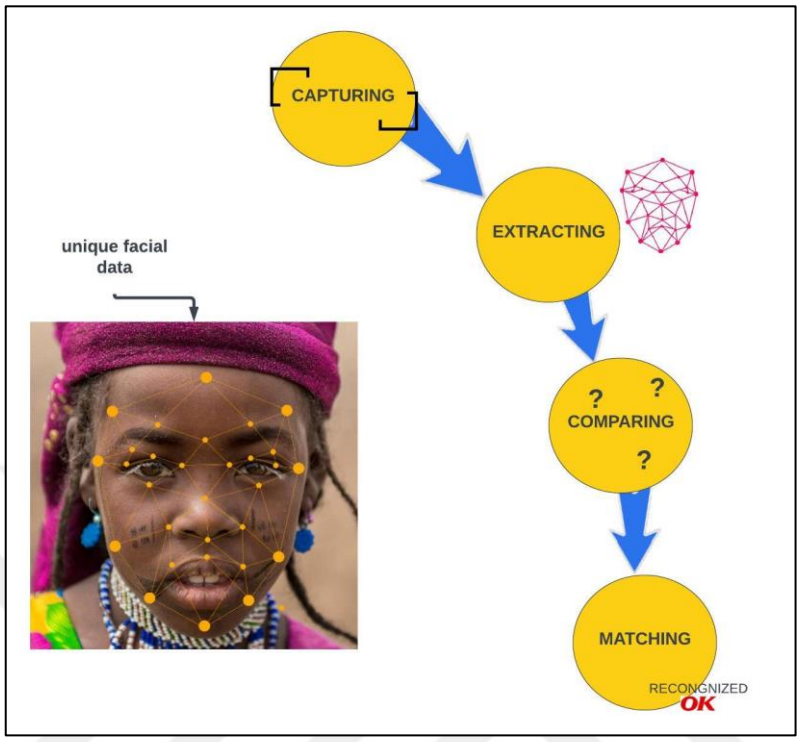

Figure 2.6: Face Recognition Process Steps.

Face recognition can be further categorized into two main types: verification (one-to-one matching) and identification (one-to-many matching). Verification involves confirming the identity of an individual by comparing their face to a single reference or stored template. Identification, on the other hand, involves searching a database to find a matching face among a set of known faces. Ed CNN architectures that balance efficiency and accuracy.

2.9.3 Siamese Networks In Face Recognition

Siamese Networks are a type of neural network architectures that are mainly used to find similarities or relationships among two comparable things. Siamese Networks, were first proposed by [64] typically consist of two symmetrical sub-networks that share the same parameters and each take one of two input vectors. For the purpose of determining their similarity, the outputs are fed into a contrastive loss function that calculates that similarity. Siamese Networks have been widely used in the context of Face Recognition to assess the similarity of two face images. The capacity of Siamese Networks to learn a function that maps an input (in this case, an image of a face) into a space where the distances are

meaningful is what gives them their power for this task. Images of the same face tend to be closer together in this learned space, while images of different faces are further apart.

The Triplet Loss, described by [35], is a specific technique for training such networks. This method works by not only minimizing the distance between the embeddings (network output) of an anchor image and a positive image (same identity), but also maximizing the distance between the anchor image and a negative image (different identity). This simultaneous "push-and-pull" effect ensures that the learned space is structured in such a way that the distance between embeddings correlates with the identity of the faces.

In the framework of Siamese neural networks and triplet loss [45], an anchor represents the point of comparison used to compare the similarity of three inputs. Typically, the anchor is one of three inputs in a triple, with the other two being a positive image or true image and a negative image or false image. To solve a few-shot problem, the anchor image's dissimilarity to the positive image must be low and the anchor image's dissimilarity to the negative image must be high measured using a distance metric [65].

$$L = \max(\|f(A)-f(P)\|2 - \|f(A)-f(N)\|2 + \text{alpha}, 0), [35] \quad (2.1)$$

where A is the Anchor, P is the Positive sample, N is the Negative sample, f is the embedding function learned by the network, and alpha is a margin that is enforced between positive and negative pairs. The margin is used to keep the network from falling into trivial solutions, forcing it to encode useful features in the embeddings for face recognition. FaceNet and similar models have demonstrated outstanding results on a number of face recognition benchmarks using this formulation, which has proven to be highly efficient for face recognition tasks. It demonstrates the effectiveness of Siamese Networks and the Triplet Loss in learning meaningful embeddings that capture the identity of faces.

2.9.4 Challenges With Face Recognition

Real-time face recognition encounters various challenges. One common challenge is handling variations in lighting conditions, poses, facial expressions, and occlusions. Illumination changes, shadows, and non-uniform lighting can affect the quality of captured face images and impact recognition performance. Coping with pose variations, such as extreme angles or rotations, requires robust algorithms capable of capturing facial details

across different orientations. Facial expressions, occlusions, and partial face obstructions further complicate the recognition process. These challenges necessitate the development of robust real-time face recognition methods that can handle diverse environmental conditions and ensure accurate and reliable identification.

3. PROPOSED METHODS

Here we present our proposed work, methods and techniques developed and implemented in the system, with the power of deep machine learning and computer vision, deployed in face detection and recognition, the experiments carried out using on Google Colab which provides a convenient and accessible platform and for implementing the experiments using popular libraries such as TensorFlow and Keras. With Colab, researchers and practitioners can leverage the power of these libraries without the need for extensive setup or hardware requirements. And TensorFlow, an open-source machine learning framework, offers a comprehensive ecosystem for building and deploying deep learning models. Its integration with Colab allows users to access and utilize its extensive functionalities for tasks like data preprocessing, model construction, training, and evaluation. Keras, a high-level neural networks API, simplifies the process of building and training models, enabling rapid prototyping and experimentation. Combined with Colab, these libraries provide a seamless environment for implementing, testing, and fine-tuning various machine learning models and algorithms, fostering innovation and advancing research in the field.

The proposed methods and experiments including the optimization techniques and neural networks architectures will be discussed in a sequence of (proposed optimizer, face detection model, face recognition model).

By the proposed work we aim to investigate and develop a proper optimization algorithms and techniques combination. Develop, fine-tune or modify neural networks to perform regression and classification problems for face detection, and classification problem for face recognition part. Both can be used for in-car anti-theft system. This section outlines the methods and materials employed in the research to achieve the objectives of the study.

3.1 INTRODUCTION

Machine learning and deep learning advancements have resulted in the development of various optimization algorithms, which are critical in determining the performance of neural network models. An optimizer's primary function is to efficiently tweak the parameters of a model in order to minimize the loss function. This chapter delves into a novel optimization method that combines the Yogi optimizer, Nesterov Accelerated Gradient (NAG), and Stochastic Weight Averaging (SWA) principles. This combination results in a powerful

hybrid optimizer aimed at improving the performance of machine learning models, with a particular emphasis on tasks in face detection and recognition.

3.2 REVIEW OF OPTIMIZATION TECHNIQUES

Introduced by [43] paper, the Yogi optimizer was designed to tackle a common issue with models such as Adam, which is the sensitivity to the selection of learning rate. Yogi adaptively fine-tunes the learning rate, providing a more robust alternative to other optimizers like Adam or RMSProp that also adaptively adjust the learning rate.

YOGI algorithm is a modification based on the famous ADAM algorithm proposed by 4 Google analysis engineers [43], suggests controlling the increase in the effective learning rate resulting a way better performance, the main changes in the YOGI algorithm was modify and enhance the second order momentum by using a second order momentum with a Sign, instead of using the traditional exponential smoothing method to update second order momentum like ADAM dose,

And the sign can be either plus or minus within different direction.

YOGI is shown to outperform methods such as ADAM in several challenging machine learning tasks with very little hyperparameter tuning.

$$v_t \leftarrow \beta_1 v_{t-1} + (1-\beta_1) g_t \,, \tag{3.1}$$

$$s_t \leftarrow s_{t-1} + (1-\beta_2) g_t^2 \oplus sign(g_t^2 - s) \tag{3.2}$$

Beta 1 and Beta 2 are nonnegative weights parameters, where commonly are (Beta1= 0.9) and (Beta2= 0.999) that is variance estimate moves much more slowly than the momentum term. Note that if we initialize (v0 = s0 = 0) we have a significant amount of Bias initially towards smaller values.

S-1 tells what are the changes in magnitude of the gradient, Vt and St are the calibrated Vt is the first order momentum and St is the Second order momentum, and calculations of corrected gradient and the Updated gradient are typically the same as ADAM.

Stochastic Weight Averaging (SWA), as described by [42], is a model generalization technique. SWA accomplishes this by averaging multiple points along the SGD trajectory rather than using only the most recent iteration. This method finds broader optima, which often correlate with better generalization on previously unseen data. SWA is a regularization method that aims to improve the generalization performance of deep neural networks [42]. It computes the average of model weights throughout the training process, resulting in a smoother and more robust solution [41]. SWA functions on averaging epochs weights of the networks that the YOGI optimizer has. We set running average of the weights Starting from a specified Epoch of the training.

Yogi-SWA optimizers combine a stochastic weight averaging (SWA) method of optimization added to the Yogi optimizer. The Yogi optimizer changes the learning rate for each weight of the neural network model based on the gradients' historical information, whereas SWA improves generalization through averaging the model weights over multiple epochs, resulting in a more stable training process and output. And the following was done:

Algorithm: Yogi Optimizer algorithm with SWA

Initialize
A small constant: $\varepsilon = 10^{-8}$
Number of iterations: n
Learning rate: α
Parameters: $\beta_1 = 0.09$, $\beta_2 = 0.999$
SWA start epoch: s
Time step: $t = 0$
The first moment estimate: $m = 0$
The second raw moment estimate: $v = 0$
Weight: w
SWA weights: w_{swa}
Epoch count: $k = 0$

While not converged
    Increment time step: $t = t + 1$
    For i **to** n **do**:
        Compute gradient: g_t
        Update biased first moment estimate: m_t
        Update biased second raw moment estimate: v_t
        Compute bias-corrected first moment estimate: $m^{\wedge}$
        Compute bias-corrected second moment estimate: $v^{\wedge}$
        Compute effective learning rate: α_t
        Update weights: w_t
        if $t >= s$:
            Update SWA weights: w_{swa}
            Increment epoch count: $k = k + 1$
            Set weights to SWA weights: $w_t = w_{swa}$
    End for
End while
Return w_t

Where the total amount of training iterations n across the entire dataset is n, the learning rate is α, the step size of the model parameter update, and β1 and β2 are the hyperparameters that control the exponential decay rates for the moving averages of the gradient's first and second momentum, along with a small number ε added for numerical stability to prevent division by zero and s is the starting epoch used to begin computing SWA weights averaging. The result is the optimized model weight w. The first momentum estimation is based on a moving average of the gradients and is similar to momentum. Its purpose is to dampen the oscillations in gradient updates.

While v is the second momentum, it is a moving average of the squared gradients that aids in adaptively changing the learning rate for each weight according to historical gradient information. w_{swa} is the SWA parameter, which is a running weights average of the model after I exceeds s, and k is a counter that keeps track of the epoch number.

The w_{swa} parameter is set to a running weights average of the model. Set the learning parameters (β1, β2), which include the epoch count k, weight w, and Yogi parameters. And then calculating the gradient at the current weight w at each time step t over a set number of iterations. Then, first and second, calculate the momentums. Yogi is an adaptive gradient descent algorithm, which means it updates the model parameters based on the gradients' moving averages and squared values. By calculating bias corrections on the first and second moment estimates, the algorithm can generate unbiased estimates for large t values, making it more accurate and compensating for the origin initialization. Then calculate an effective learning rate that increases with the gradient's corrected variance and use it to update the model's weights w. If the current iteration exceeds than the SWA start epoch s, a running average of the model weights is calculated. The SWA method averaging over a number of points along the Yogi's trajectory can improve the accuracy of the model and generalization capacity.

Returning the computed weights w, if SWA was used, those are the weights that have been averaged and w_{swa} serves as the model finale weights, and this is repeated until a stopping criterion, which is a maximum number of iterations, is met.

The weights averaging method is simply a type of regularization that can aid in the prevention of overfitting by discouraging the model from relying heavily on any one parameter. This method is especially useful when the number of training examples is much less than the number of parameters. The number of s should be large enough to allow the weights to stabilize, but small enough that the computation is not too expensive. The model parameters are fine-tuned to strike a balance among stability and learning speed.

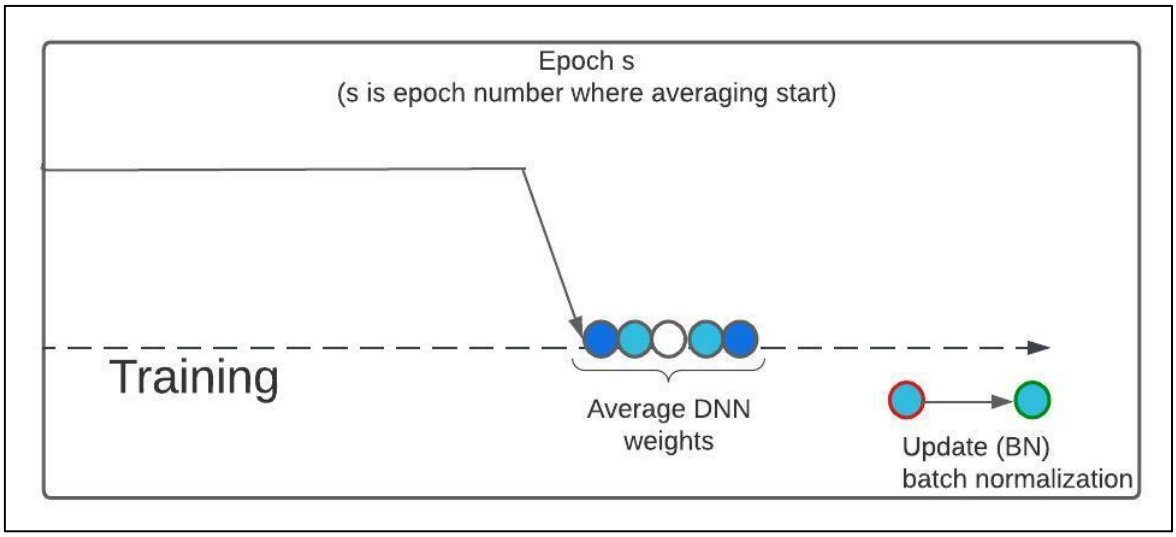

Figure 3.1: SWA Weights Averaging.

Nesterov Accelerated Gradient (NAG), suggested by Nesterov in his paper [47] provides a distinct improvement over the standard gradient descent algorithm. NAG employs a "look-ahead" step that includes estimating the gradient at a point in the momentum's direction. This method provides a smoother path to the minimum as well as faster convergence.

3.3 PROPOSED OPTIMIZATION

The proposed optimization strategy aims to combine the Yogi optimizer, Nesterov Accelerated Gradient (NAG), and Stochastic Weight Averaging (SWA) into a comprehensive and efficient optimization algorithm. The proposal is based on modifying the Yogi optimizer to incorporate the momentum update from NAG, thereby benefiting from NAG's enhanced convergence speed and reduced oscillations. SWA is then incorporated into the newly developed optimizer. SWA, which focuses on finding larger minima, is expected to improve model generalization on previously unseen data. The Nesterov Accelerated Gradient (NAG) modifies the Yogi optimizer by including a 'look-ahead' step in the direction of momentum before calculating the gradient. The momentum update becomes more accurate by calculating the gradient at a future approximate point rather than the current point, resulting in a smoother trajectory towards the minimum and faster convergence. The modified Yogi optimizer with NAG thus takes advantage of Yogi's robust learning rate tuning and NAG's accelerated convergence, significantly increasing the optimization process's efficiency.

Following the integration of NAG into the Yogi optimizer, the next step is to incorporate Stochastic Weight Averaging (SWA) into the resulting optimizer. This involves averaging multiple points along the trajectory of the stochastic gradient descent rather than just considering the last iterate. The motivation behind this integration lies in the potential of SWA to find broader optima that often correlate with better performance on unseen data, enhancing model generalization.

The proposed hybrid optimizer has significant potential benefits, particularly for difficult computer vision tasks like face detection and recognition. The proposed optimizer is expected to improve the efficiency of model training by providing a robust and reliable optimization solution, resulting in improved performance of face detection and face recognition models. Furthermore, by seamlessly integrating Yogi's robust learning rate tuning, NAG's faster convergence, and SWA's improved generalization, the proposed optimizer could be applied to other machine learning tasks besides face detection and recognition.

Algorithm : Represents the algorithm pseudocode of the proposed optimization method.

Initialize
A small constant: $\varepsilon = 10^{-8}$
Number of iterations: n
Learning rate: α
Parameters: $\beta 1 = 0.09,\ \beta 2 = 0.999$
SWA start epoch: s
Time step: $t = 0$
The first moment estimate: $m = 0$
The second moment estimate: $v = 0$
Weight: w
SWA weights: w_{swa}
Epoch count: $k = 0$

While not converged
Increment time step: $t = t + 1$
For i to n do:
  Compute gradient: g_t
  Interim weight update (for NAG): $w_{interim} = w - \alpha * \beta 1 * m$
  Compute interim gradient (for NAG): $g_{interim} = compute\_gradient(w_{interim})$
  Update biased first moment estimate: $m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g_{interim}$
  Update biased second raw moment estimate: $v_t = v_{t-1} - (1 - \beta_2) * sign\left(v_{t-1} - g^2_{interim}\right) * g^2_{interim}$
  Compute bias – corrected first moment estimate: $\hat{m} = m_t \,/\, \left(1 - \beta_1^t\right)$
  Compute bias – corrected second raw moment estimate: $\hat{v} = v_t \,/\, \left(1 - \beta_2^t\right)$
  Compute effective learning rate: $\alpha_t = \alpha \,/\, \left(sqrt\left(\hat{v}\right) + \varepsilon\right)$
  Update weights with NAG: $w_t = w_{t-1} - \alpha_t * ((\beta_1 * m_t) + ((1 - \beta_1) * g_{interim}))$
  if t >= s:
    Update SWA weights: $w_{swa} = w_{swa} + (w\_t - w_{swa}) \,/\, k$
    Increment epoch count: $k = k + 1$
    Set weights to SWA weights: $w_t = w_{swa}$
End for
End while
Return w_t

Where ε is a small constant used to avoid division by zero while estimating the effective learning rate α, and the total number of iterations is n, and β1 and β2 are decay rates for moving averages of the gradient and its square, respectively. At epoch s, SWA (Stochastic Weight Averaging) begins. The time step t represents the current iteration or update, where m is the initial momentum estimate, which is essentially a gradient moving average, and v is a moving average of the gradient squares, which represents the second momentum estimate. The model weights w. and w_{swa} are the averaged weights estimated by SWA if the SWA starting epoch number is equal to or greater than s. k is the number of epochs since the start of SWA. W_{interim} is the interim weight used to calculate the interim gradient in the Nesterov accelerated gradient. The gradient g_t of the loss with respect to the parameters, as

well as the gradient calculated $g_{interim}$ at the interim weights, were used for the Nesterov update. The first momentum m_t and second momentum v_t estimates, the bias corrected first and second moment estimates $m^{\wedge}$ and $v^{\wedge}$. a_t is the effective learning rate at time step t, and w_t is the weighted average of the first and second momentum estimation. The model's weights at time step t.

Algorithm:

Proposed Optimizer

1. Initialize variables.
2. Enter a loop until the stopping criteria are met (until the loss is sufficiently small or a predefined number of iterations have been reached).
3. For each iteration, increment the time step and loop over all n batches.
4. In each batch, first perform an interim weight update for NAG and then compute the interim gradient at these interim weights.
5. Update the first and second momentum estimates with this interim gradient.
6. Compute bias-corrected moment estimates.
7. Compute the effective learning rate.
8. Perform a weight update with NAG.
9. If the time step is greater than the start epoch for SWA, update the SWA weights and increment the epoch count.
10. If using SWA, set the model weights to the SWA weights.
11. Repeat steps 3-10 until convergence.
12. Return the final model weights.

The proposed hybrid optimizer has significant potential benefits, particularly for difficult computer vision tasks like face detection and recognition. The proposed optimizer is expected to improve the efficiency of model training by providing a robust and reliable optimization solution, resulting in improved performance of face detection and face recognition models. Furthermore, by seamlessly integrating Yogi's robust learning rate tuning, NAG's faster convergence, and SWA's improved generalization, the proposed optimizer could be applied to other machine learning tasks besides face detection and recognition.

3.4 PROPOSED FACE DETECTION

A deep neural network-based object detection model implemented. The model trained using the collected dataset and the ground truth bounding box annotations. The implementation will utilize a deep neural network (DNN) architecture. Creating a reliable detection model

can be a challenging task, researchers can design a deep neural network that suits the requirements of the task, or modify and finetune an already exists architecture that proven efficiency and reliability.
Robust and dependable face detection algorithms are the foundation of a wide range of computer vision applications, including surveillance, biometrics, and social media. They serve as a foundation for more complex tasks such as face recognition, emotion detection, and age estimation. Creating a robust face detection model necessitates careful consideration of the method as well as, most importantly, the quality and preparation of the dataset. This part of the thesis focuses on the proposed face detection method.

Face detection in our model consists of two problems or tasks, first is a classification problem, to classify images whether consists a face or not, and second task is a regression problem, to localize faces present within images.

To accurately detects humans faces presents within images, we present the outlines of our face detection method:

a. Collecting and Annotating Images
b. Applying Augmentation
c. Create a Deep CNN Model
d. Train and Evaluating Model Performance

3.4.1 Collecting, Preparing And Preprocesseing Dataset

Our face detection model's dataset consists of 22 individuals and approximately 1,000 images. Each person contributes between 25 and 45 images, with a variety of poses, expressions, lighting conditions, and backgrounds. This variety within the dataset is critical because it improves the model's ability to generalize, allowing it to perform better on previously unseen data.

All images were resized to a uniform size of 250x250 pixels during the data preparation stage. This standardization is essential because it ensures that all images are the same size. Additionally, it reduces computational load, making the dataset process more efficient.

In the following steps, the dataset is prepared and preprocessed for the model to improve and extend the collected images:

a. Image shape resizing

All images are resized to (120, 120, 3) to match the size of the neural network input.

b. Labeling and annotation

During the annotation process, a bounding box is drawn around each face in the images. Each bounding box is represented by a set of coordinates (x, y, width, height) that indicate where the face is in the image. A JSON file containing these coordinates was created for each image in the dataset. JSON files are small and easy to read, making them an excellent choice for storing and sharing such metadata. This step involves annotating the faces within each image and labelling each face with a (face) label and a class number of 1. This step was carried out by hand using the LabelMe tool [39]

a. Augmentation

Deep learning models may be built using larger and more representative datasets to boost the size and quality of datasets as well as the diversity of the training data, which may enhance their accuracy as well as performance [66]. Is a powerful way to prevent overfitting and improve model generalization by providing it with a wider range of examples to learn from. RoboFlow [67] Through:

b. Rotate (horizontal, vertical)
c. Crop (zero minimum zoom, 25% maximum zoom)
d. Brightness (between -15% and +15%)
e. 90-degree rotation (clockwise, counter-clockwise, upside down),
f. Rotation (between -10 and +10 degrees)
g. Temperature (Hue) (between -25° and +25°)
h. Blur (up to 1.5 pixels)
i. Noise (up to 1% of pixels)

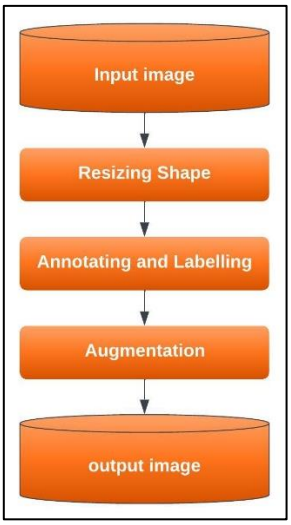

Figure 3.2: Face Detection Data Preparing and Preproccessing.

Following the collection and preparation stages, the dataset was preprocessed to prepare it for model training. Various techniques mentioned earlier were used in this step, as well as a normalization step, in which pixel values were scaled from their original range (0-255) to a range between 0 and 1. This procedure aids in the stabilization of the learning process and the reduction of training time.

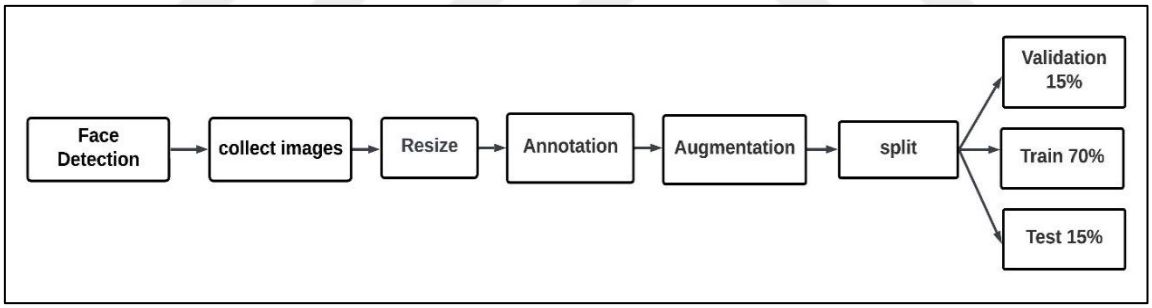

Figure 3.3: Face Detection Dataset.

Following preprocessing and augmentation, we created 5000 image and 5000 label formed the dataset for the face detection model, which divided into 70% train, 15% test, and 15% validation sets.

3.4.2 Face Detection Model Architecture

The neural network has an input layer size of 120,120,3 and is followed by five blocks of hidden layers, which are inspired by the VGG16 architecture [68] While the VGG16 accepts inputs in the shape of 244,244,3 and was designed for image classification, we modified

it, to perform a regression problem to locate faces and classify whether a face is present or not in the input image.

As a result, the model architecture is as follows: an input layer in the shape 120,120,3 followed by five blocks, the first of which is two convolution layer followed by a MaxPooling layer, the second of which starts with two convolution layers and a MaxPooling layer, the third, fourth, and fifth blocks each consist of three convolution layers and a MaxPooling layer, all with a Relu activation function.

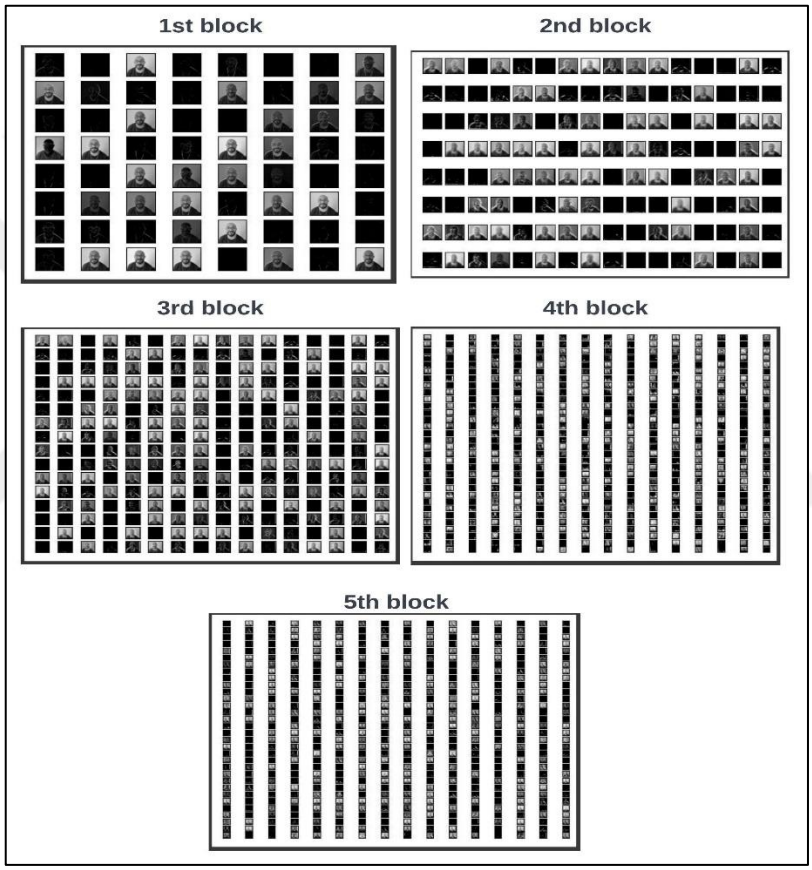

Figure 3.4: Example of Image Through VGG16 Architecture.

From here we created two prediction heads, The first prediction head layers are GlobalMaxPolling followed by Dense 2048 layer and Relu activation function followed by another Dense layer of 1 unit with Sigmoid activation to complete our classification task. The following is the second prediction head is GlobalMaxPolling followed by a Dense 2048 layer along with Relu activation function then another Dense layer of 4 unit with Sigmoid activation to generate the regression model, to localize the boundary box (x1,y1,x2,y2) predicted face location with a localization loss function from [You Only Look Once:

Unified, Real-Time Object Detection] . for object detection and IoU function estimates the Intersection over Union (IoU) for predicted bounding boxes and ground truth bounding boxes where area of interaction divided by area of union.

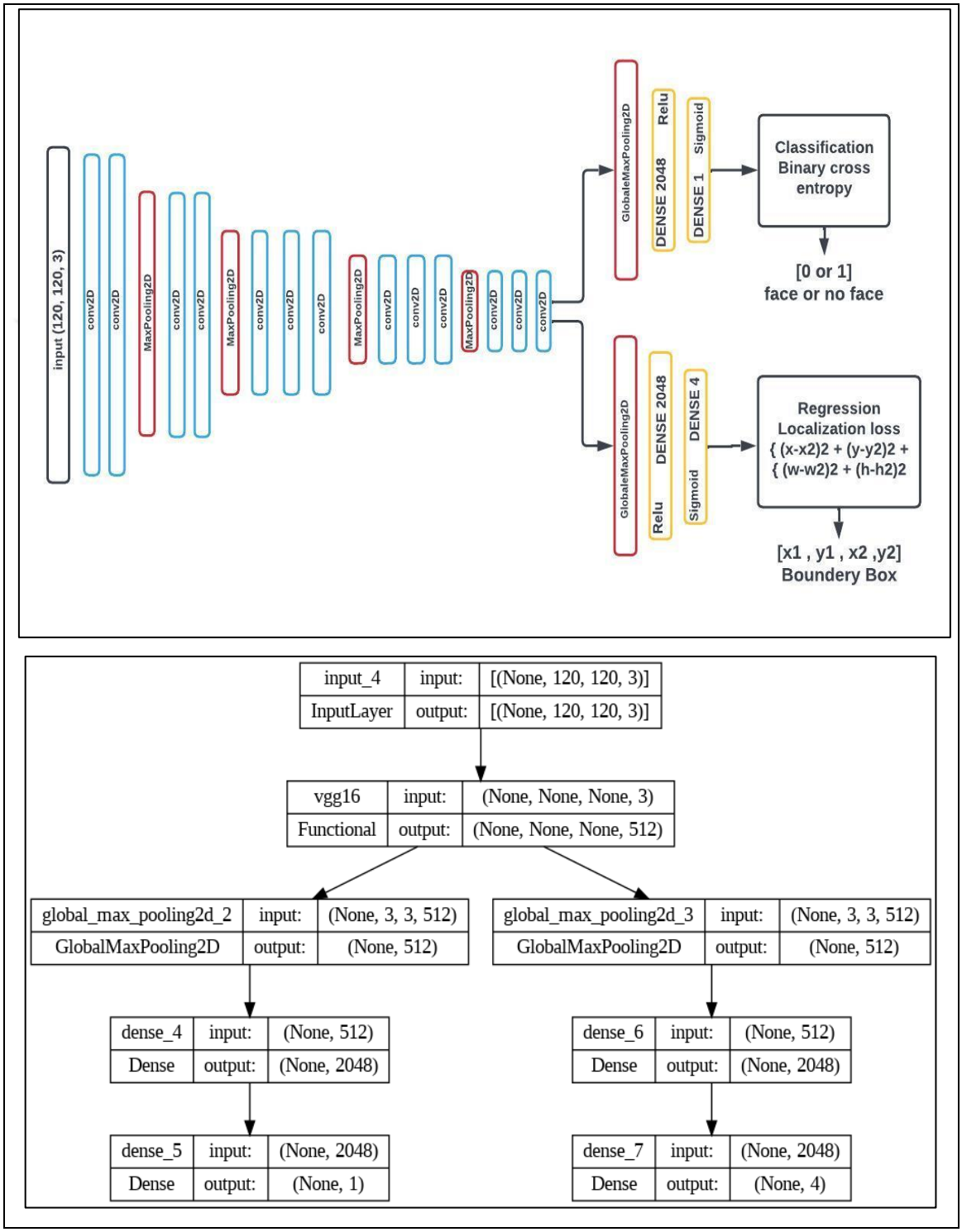

Figure 3.5: Face Detection Architecture.

3.4.3 Defining Loss Functions

Loss functions in machine learning play a crucial role in quantifying the discrepancy between predicted and actual values, serving as a measure of model performance.

By defining a Localization Loss, we are calculating the difference between the true width and high coordinates and the predicted width and high coordinates, which is representing our Regression loss.

And our Classification Loss is BinaryCrossEntropy, the goal of Binary Cross-Entropy loss is to minimize the loss value [44]. It is commonly used in binary classification tasks, where the objective is to classify instances into one of two classes, which are face or no face (1 or 0).

3.4.4 Training

The experiments to train the face detection model are set up as follows: data are batched in size of 8, learning rate set to be 0.0001, threshold is 0.5, and training period for 100 epochs. During training, we calculated total loss for our classification and regression loss functions, and IoU for the training and validation datasets; IoU later also calculated for a test dataset and is used to evaluate the model in detecting boundary boxes for test data that the model has never seen or trained on.

The intersection of the expected and actual boundary boxes divided by their union is defined as IoU. If the IoU is greater than the threshold, the prediction is termed True Positive; otherwise, it is considered False Positive.

The results of the training step, the evaluating and testing will be displayed and discussed in chapter 4 of this thesis.

3.5 PROPOSED FACE RECOGNITION

Face recognition, a critical application in computer vision, has made tremendous progress in recent years thanks to deep learning. This part of this thesis describes the proposed face recognition method, which uses Siamese Networks and Triplet Loss to provide an efficient solution for Few-shot learning problems [45]. This method makes use of the Labeled Faces in the Wild (LFW) [69] dataset, A collection of images of 1680 celebrities is included in the

face extracted version for face recognition. 8204 images, each with 2-50 different images. The dataset was divided into two parts: 90% for training, and 10% for testing.

And a modified ResNet50 embedding model to generate a unique feature vector for each input. Implemented by the TensorFlow Functional API to combine our Network components,

Bromley and LeCun introduced the Siamese Networks in 19931, and they have found significant applications in the domain of face recognition. Their architecture consists of two or more identical subnetworks, each with the same parameters and weights, making them ideal for Few-shot learning tasks such as recognizing an identity from a single example.

We use Siamese Networks with a triplet architecture in the proposed method, with each triplet consisting of an anchor, positive, and negative image. The anchor and positive images are of the same identity, while the negative image is of a different identity.

FaceNet [35], employs a triplet loss function that is designed to separate positive and negative pairs by a large margin, thereby improving the discriminating ability of the embeddings. The triplet loss function is used to train the Siamese Network in our proposed method, which is inspired by FaceNet. The goal is to learn an embedding function in which the L2 distance in the embedding space between the anchor and positive images (same identity) is less than the distance between the anchor and negative images (different identity). In the framework of Siamese neural networks [45], an anchor is a point of comparison used to compare the similarity of three inputs. Typically, the anchor is one of three inputs in a triple, with the other two being a positive image or true image and a negative image or false image. To solve a few-shot problem, the anchor image's dissimilarity to the positive image must be low, while the anchor image's dissimilarity to the negative image must be high measured using a distance metric [70].

To implement our Triplet Siamese Network this, include the following steps:

a. Collecting Images, preparing and Preprocessing the Dataset
b. Create an Encoding (Embedding) Model
c. Create Distance Layer L2
d. Creation of Siamese Network
e. Train and Evaluating Model Performance

3.5.1 Collecting, Preparing And Preprocesseing Dataset

The following actions were taken during the preparation and preprocessing:

a. Image shape resizing

To match the neural network input size, all photos are scaled to (299, 299, 3).

b. Preprocessing of triplets

To implement few-shots learning using a Triplets Siamese Network, a number of steps must be executed on the dataset:

a. Create an empty list of triplets to store the created triplet images.

b. Do the following for each class in the dataset:

i. Get all images from this class and save them in class_images.

ii. If the total amount of class_images is < 2 , skip to the next class (at least two images are required to create a triplet).

c. Create triplets for every class that has more than one image:

i. Choose an image randomly from the class_images to serve as the Anchor_image.

ii. Choose a different picture from class_images randomly (not the Anchor_image) to serve as the Positive_image.

iii. Choose one picture randomly from any different class except the present one, to be Negative_image.

iv. Add the triplet (anchor_image, positive_image, negative_image) to the list of triplets.

d. Keep Repeating step 3 till a triplet for each image in every class is formed.

e. Return the triplets list, which contains all created triplets.

Figure 3.6: Triplets Samples.

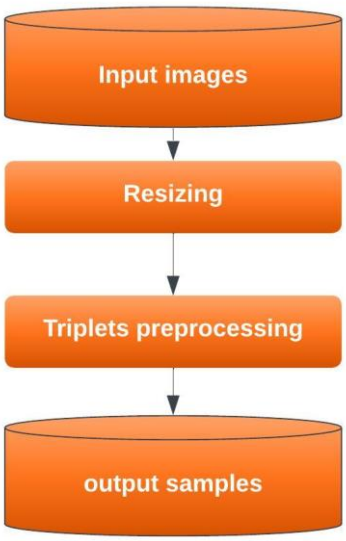

Figure 3.7: Face Recognition Data Preprocessing.

3.5.2 Siamee Network Architecture

The creation of Siamese Networks for face recognition includes major 3 components in general:

a. Embedding or Encoding Model

b. Distance L2 Layer

c. Triplet Loss Function

As mentioned earlier on face detection part of this thesis, researchers can choose between creating or finetuning a neural network that best for the task, and the choose of the Encoding or the Embedding model is critical factor. After many experiments the researcher choose to finetune the ResNet50 residual Network to form the embedding model.

ResNet50, developed by [71], is well-known for its efficiency in image classification tasks due to residual blocks that mitigate the vanishing gradient problem in deep networks. The ResNet50 architecture has been modified for the purpose at hand in our suggested model by removing the final fully connected layer. The network outputs an embedding (a feature vector) rather than a classification score.

The embedding layer, which contain three subnetworks, each with an input size of (299,299,3), each input followed by ResNet50 to extract significant features for the input image.

ResNet50 is a 50-layer Network, modified for the purpose by replacing the last block with a Flatten layer first, followed by a Dense layer 512 unit and Relu activation, a Dense layer 256 unit and Relu activation, batch normalization implemented on both, and the last layer is the output layer, creating a features vector in size 256 units for the input.

Secondly in the architecture of our face recognition model, is the Distance L2 layer, The distance embeddings are calculated in two steps, the first of which involves taking two embeddings as input, an anchor and positive embeddings. And subtract them as the following

$$||f(A) - f(P)||^2 \quad (2)\ [35] \tag{3.3}$$

Second step is, anchor and negative embeddings distance calculated in the following

$$||f(A) - f(N)||^2 \quad (3)\ [35] \tag{3.4}$$

Where $f(A)$ represents the anchor image feature vector, $f(P)$ represents the positive image feature vector and $f(N)$ belongs to the negative image feature vector.

The main gaol by doing this is to classify the images that looks similar (ideally images for the same person have a close feature, in another) and separates them from images of different people (which have different features). So ideally images of the same class (same person)

should have a small distance to the anchor image (image from same class) from equation (2). And long distance or larger number from equation (3) with a negative image or image from different class.

Triplet Loss function helps boost the discriminative power of the embeddings. This function seeks to guarantee that the anchor and positive (same identity) are more closely related in the embedded space than the anchor and negative (different identity). It improves the discrimination of the network's embeddings by maximizing the margin between positive and negative pairs. As the following

$$L_{(A,P,N)} = \max(||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + margin, 0) \qquad (3.5)$$

where L is the triplet loss function and a margin represents the value added to separate negative samples.

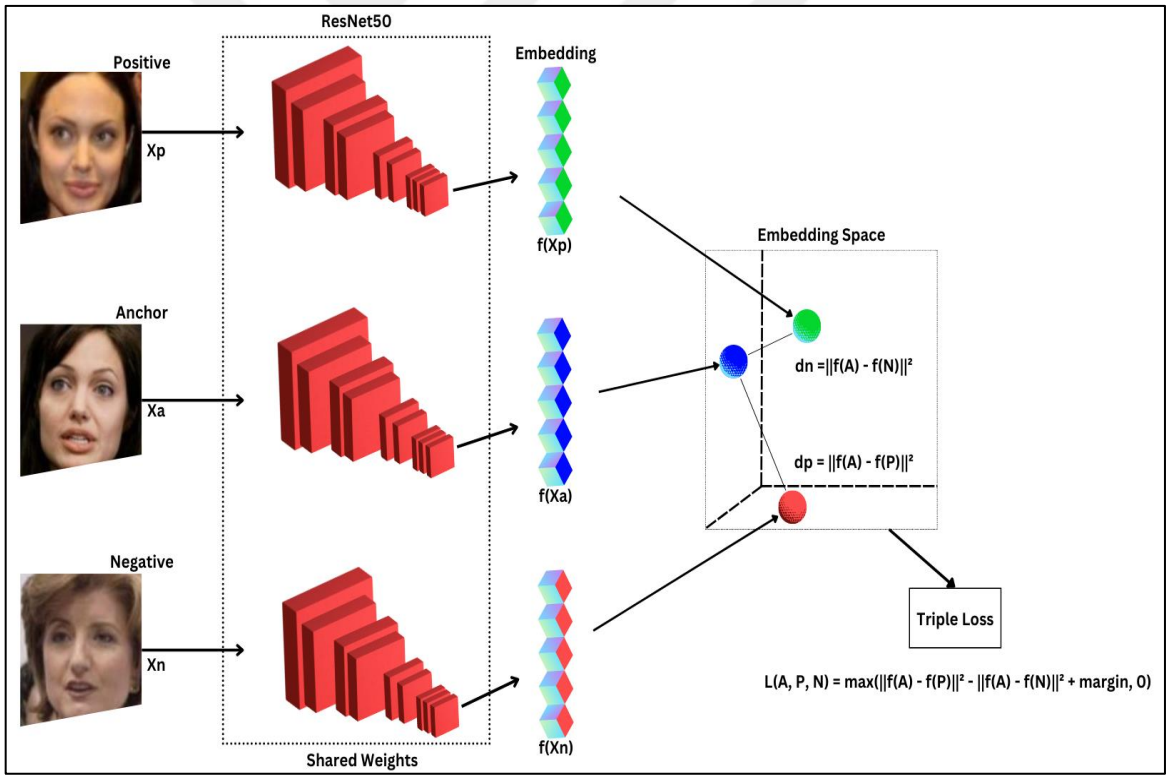

Figure 3.8: Face Recognition with Triplet Loss Architecture.

3.5.3 Training

The experiments to train the face recognition model are set up as follows: data are batched in size of 32, learning rate set to be 0.0001, and training period for 500 epochs. During training we computed metrics to evaluate the model, such as the loss value that the model tries to minimize. The accuracy ratio is calculated as a number of correct predictions to the total number of input samples. Both positive and negative Means Scores are the averages of the model's scores for the anchor-positive and anchor-negative over all data. On average, these ratings represent how well the model distinguishes between positives and negatives. Positive Score Standard Deviation is a measure that determines how much the model's positive pair scores vary. Positive pair scores in the model with a lower positive standard deviation are more consistent. A higher standard deviation suggests a broader range of scores, which may imply inconsistencies in how the model calculates positive. Negative Score Standard Deviation is a metric that measures how much the model's output differs for negative pairs. A lower negative standard deviation shows that the negative pair scores in the model are more consistent. A greater standard deviation suggests a broader range of scores, which may imply inconsistencies in how the model assesses negative pairs. In addition to positive and negative maximum and minimum Scores, which are the largest and lowest distances obtained by the model for any anchor-positive and anchor-negative in the test data, respectively. These scores represent the range of scores obtained by the model for positive and negative pairs.

The results of the training step, the evaluating and testing will be displayed and discussed in chapter 4 of this thesis.

4. RESULTS

Our proposed optimization technique improves the generalization performance of deep neural network and involves averaging the weights of the model over multiple iterations during the training this process helps to find flatter minima in the loss and reducing overfitting by using the SWA technique, and utilizes an adaptive gradient approach and controlling the decay of the learning rate more effectively, Yogi algorithm combines both additive and multiplicative updates to improve adaptivity and stabilize the optimization process. As the goal of this study was to investigate at any potential benefits of applying optimization techniques based on adaptive algorithms, such as SWA and NAG. This integrated technique was created by to improve the optimization of machine learning neural networks. In our experiments, we trained the models from scratch to perform facial detection and recognition tasks. For the study's training and testing, the proposed approaches were developed and implemented using Google Colab and different deep learning tools such as TensorFlow.

Incorporating the NAG to the algorithm improved our model in faster converge and navigate the parameter space more effectively. But it is worth to mention the time accuracy trade-off. Additional techniques mean additional calculating and computing requirements, modern optimizers always aiming to minimize time consuming, in this context. To keep balance NAG played a critical role.

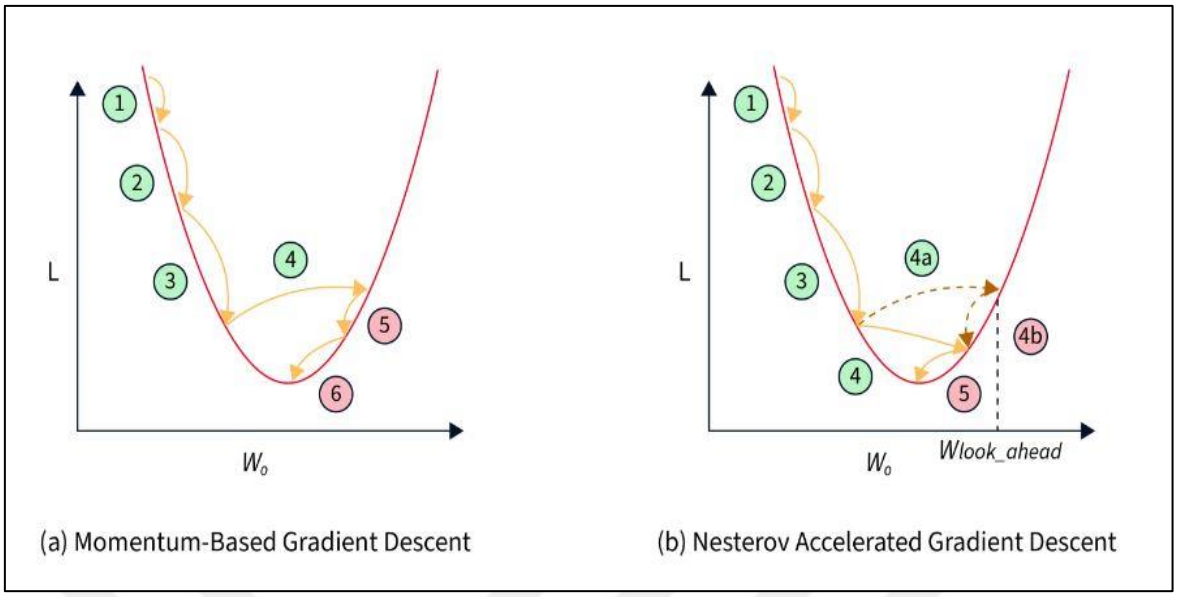

Figure 4.1: Classic Momentum Vs Nesterov Momentum (NAG).

This idea of a "lookahead" gradient gives Nesterov momentum NAG its characteristic advantage. The intuition is that the gradient at the lookahead position provides better information for where the parameters should be updated. When we are at a specific point in the parameter space, instead of calculating the gradient at that current position, we calculate the gradient at the position where we would end up at the next time step, given our current momentum.

Nesterov momentum decreases oscillations too, which are especially problematic when we are very close to the optimal solution.

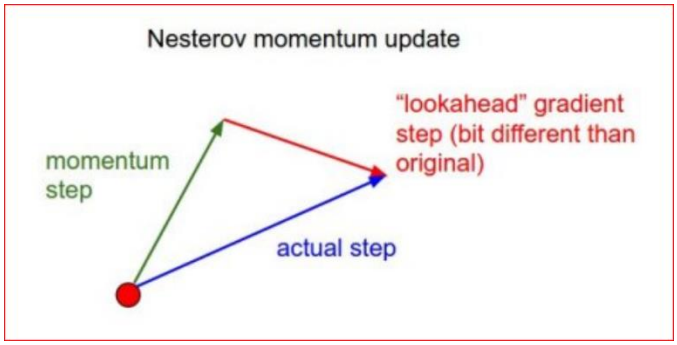

Figure 4.2: NAG Lookahead.

4.1 FACE DETECTION MODEL RESULTS

for the experiments We optimized the model using only the Yogi optimizer and obtained total loss 2.61e-04 and IoU 90.4% on the model's training dataset, IoU 89.2% for the validation dataset, and IoU 88.9% for the test dataset.

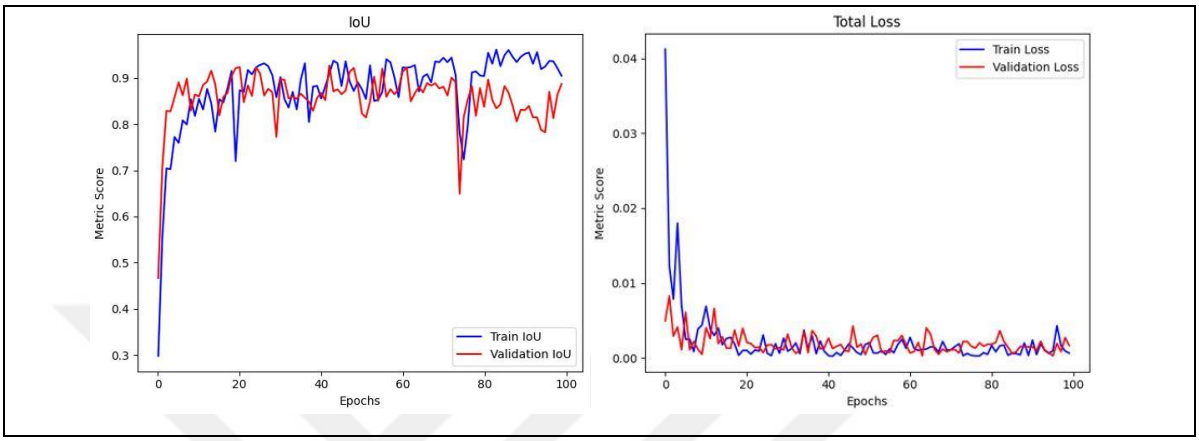

Figure 4.3: Face Detection, Yogi Optimizer IoU and Total Loss Results.

For our proposed method results, by cooperating Nesterov momentum to the Yogi algorithm. And SWA weights averaging started from epoch 25, and we got 2.18e-05 total loss with IoU 99.8% on the dataset used for training, IoU 98.7% on the validation dataset, and IoU 97.9% on the test dataset.

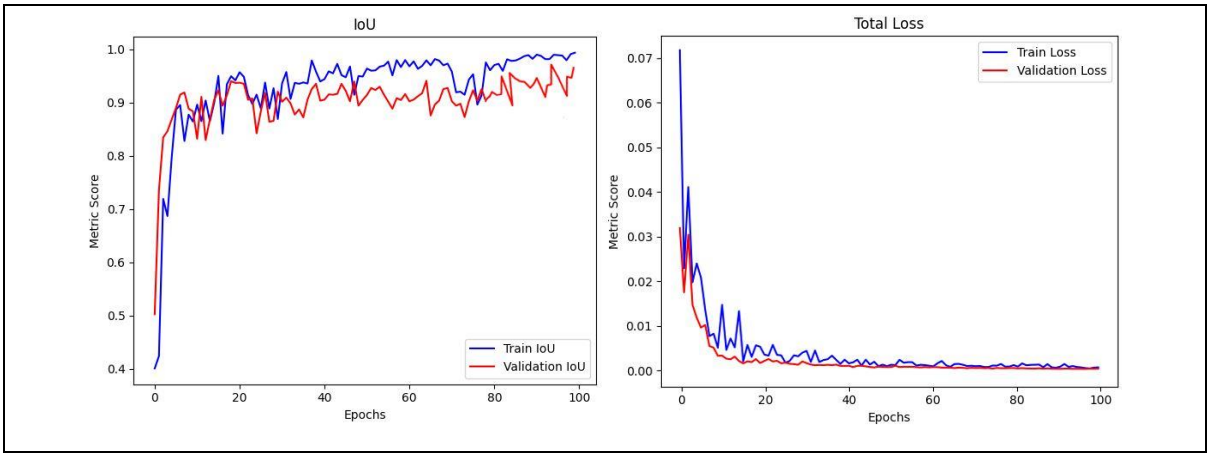

Figure 4.4: Proposed Optimization, Yogi Based with NAG and SWA Weights Averaging.

Figure 4.5: Proposed Face Detection Predictions on Test Set.

Figure 4.5: Proposed Face Detection Predictions on Test Set 'Figure Continued'.

Figure 4.5: Proposed Face Detection Predictions on Test Set 'Figure Continued'.

Figure 4.5: Proposed Face Detection Predictions on Test Set 'Figure Continued'.

4.2 FACE RECOGNITION MODEL RESULTS

Observations through experimenting our proposed method on the recognition model demonstrated in this section as following for the Yogi optimizer, total loss 2.78e-3 and accuracy 90.4% obtained.

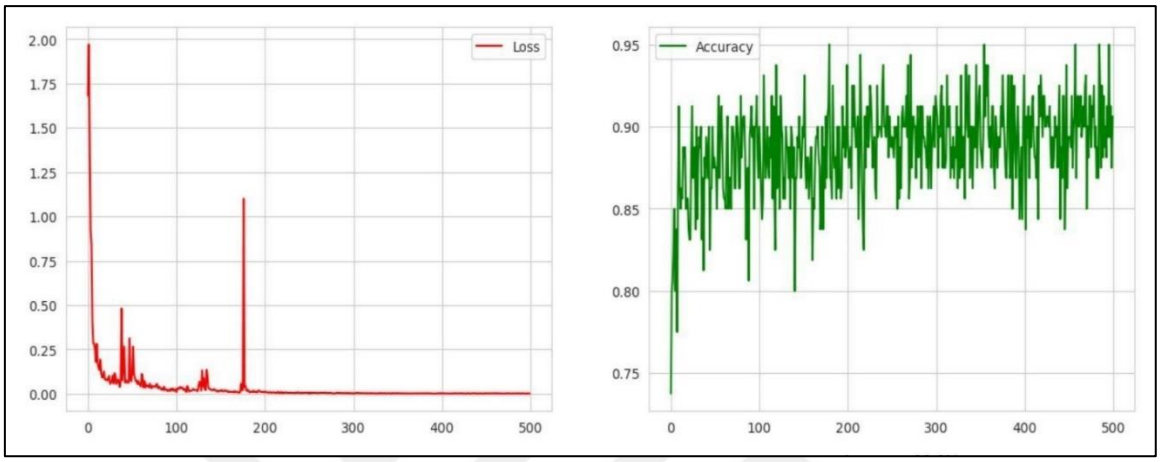

Figure 4.6: Face Recognition Loss and Accuracy Results with Yogi Optimizer.

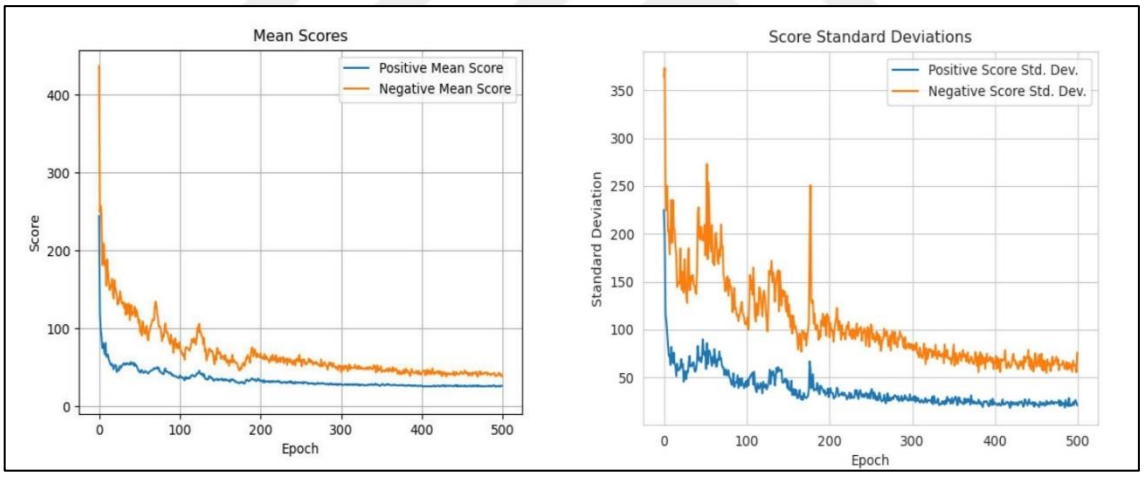

Figure 4.7: Face Recognition, Yogi Optimizer, Means and Standard Deviation std Scores.

A positive mean score of 30.12 obtained and a negative mean score of 38.121, as well as a positive standard deviation STD of 20.7 and a negative STD of 75.7. The Positive Maximum Score is 132, with a Positive Minimum Score of 1.71, and the Negative Maximum Score is 113.31, with a Negative Minimum Score of 3.5.

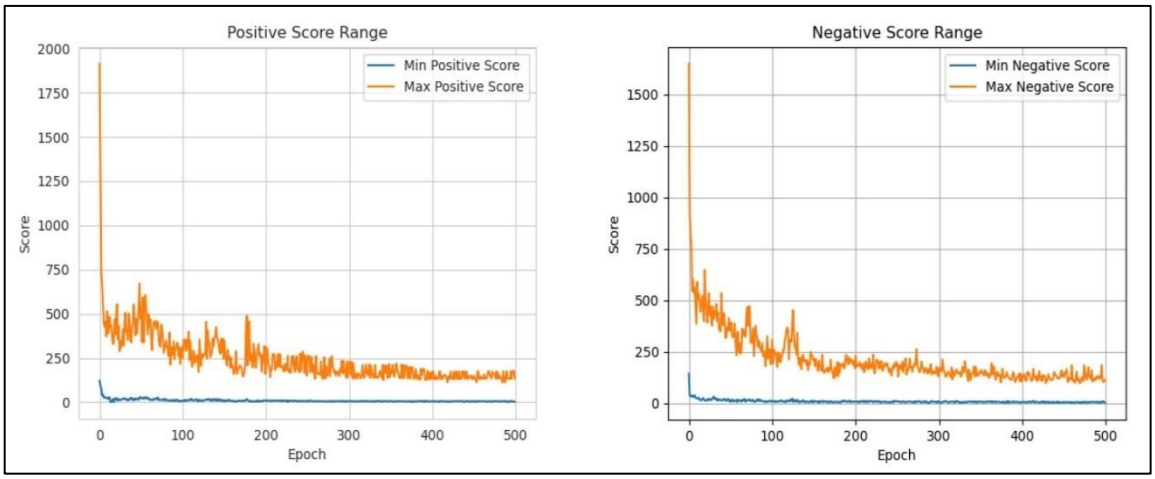

Figure 4.8: Face Recognition Optimized by Yogi, Maximum and Minimum Scores.

Our proposal optimization on Yogi optimizer modifying the algorithm to obtain a lookahead characteristic using NAG, and SWA technique implemented to start from epoch 200, and we obtained a loss value of 6.09e-4 and an accuracy of 94.5%.

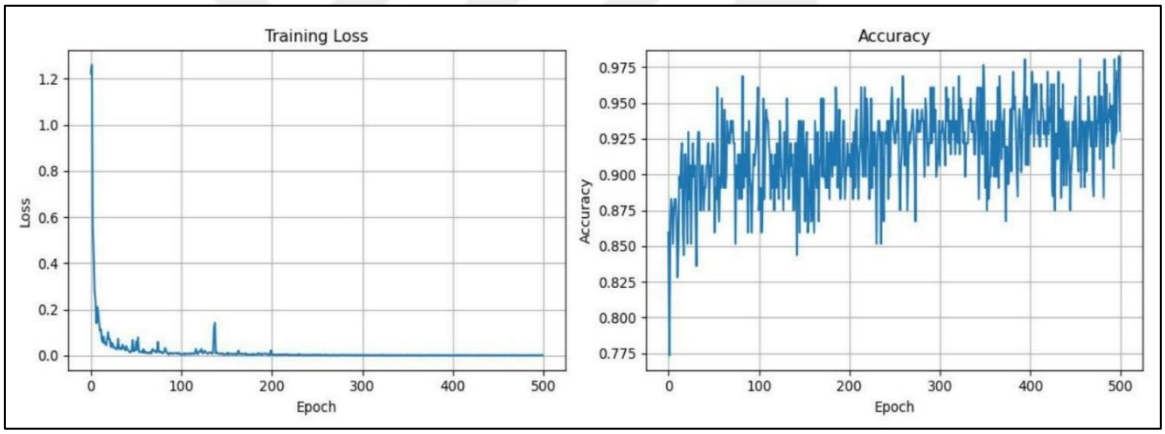

Figure 4.9: Face Recognition Proposed Optimization, Loss and Accuracy.

A positive mean score of 12.45 and a negative mean score of 111.29, a positive standard deviation STD of 6.07 and a negative STD of 23.9, a positive maximum score of 34.64 and a positive minimum score of 2.27, a negative maximum score of 522.1 and a negative minimum score of 27.4.

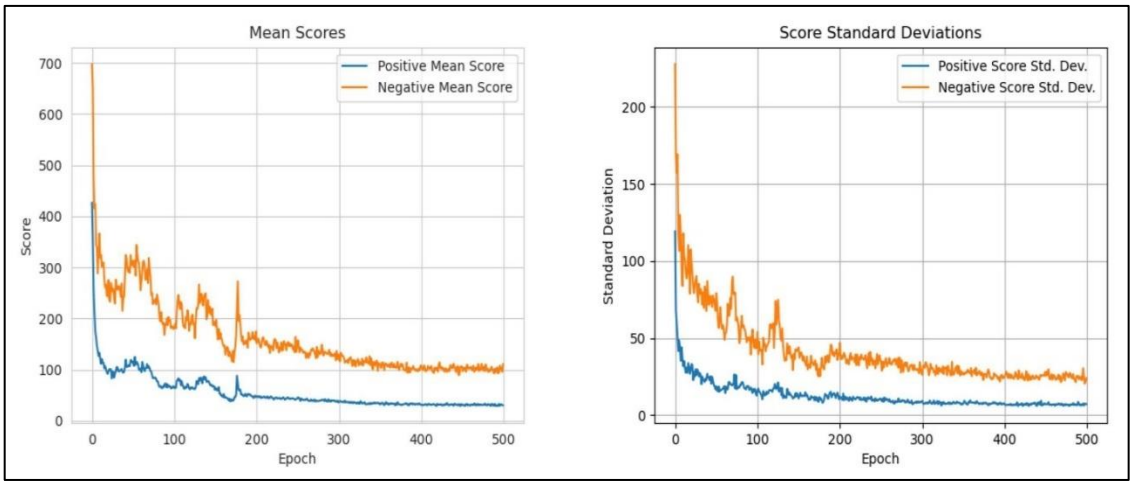

Figure 4.10: Face Recognition Proposed Optimization, Means and STD Scores

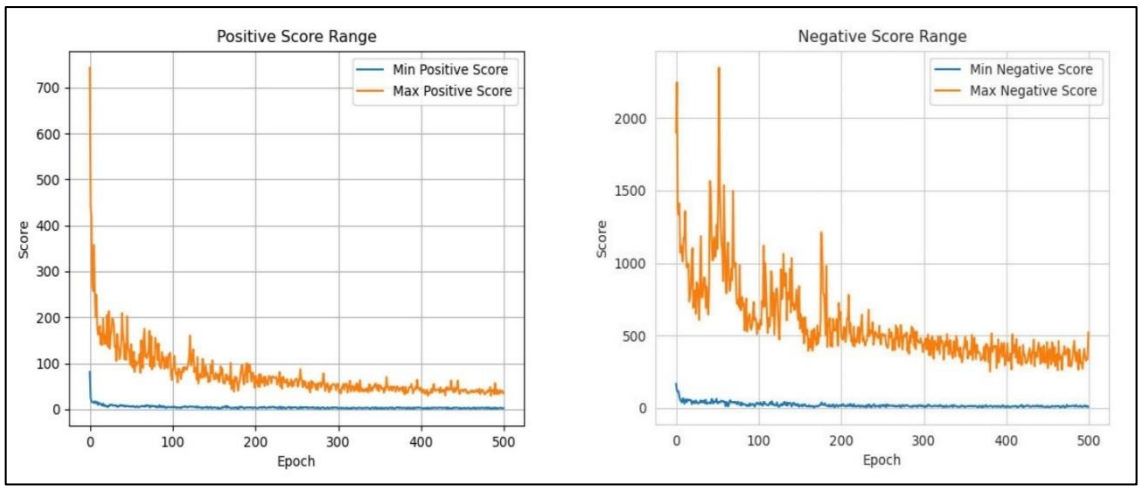

Figure 4.11: Face Recognition Proposed Optimization, Minimum and Maximum Scores.

To evaluate our proposed methods for the recognition part, we used our trained embedding model to create embeddings or feature vectors to all images of the LFW dataset, then using the model to identify similar identities in these images by their extracted facial features. By computing the cosine similarity between two vectors. To measure how similar two images are, by comparing the feature vectors (encodings) of the images. Cosine similarity measures the cosine of the angle between two vectors. It ranges from -1 to 1, where -1 means the vectors are diametrically opposite, 0 means they are orthogonal (not similar at all), and 1 means they are identical [72].

$$C_{(u,v)} = \frac{U \cdot v}{(\|u\| \cdot \|v\|)} \tag{4.1}$$

We set a threshold of total images per class or person to 10 images, for the minimum number of images a person should have in the dataset to be considered in the comparison, iterating

over all the people in the dataset. For each person, it computes their similarity to every other person. 143 people found with more than 10 images, 135 of them are closest to themselves (94.41). Then most similar 10 best similar pairs predicted by the model are demonstrated.

Figure 4.12: Face Recognition Predictions on Similarity.

Table 4.1. Comparison of Face Detection Models Performance.

Optimizer	**Total Loss**	**Train IoU %**	**Val IoU %**	**Test IoU %**
Yogi	2.61e-04	90.4	89.2	88.9
Proposed	2.18e-05	98.6	93.7	92.9

Table 4.2. Comparison of Face Recognition Models Performance.

Optimizer	**Total Loss**	**Accuracy %**	**Positive Mean**	**Negative Mean**	**Positive Std.**	**Negative Std.**	**Positive Max-Min**	**Negative Max-Min**
Yogi	2.78e-3	90.4	30.121	38.58	20.7	75.7	132 – 1.71	113.3 – 3.6
Proposed	6.09e-4	95.5	12.45	111.29	7.06	23.9	34.6 – 2.27	522.1 – 27.4

5. CONCLUSION

Throughout this thesis, we have embarked on a deep exploration of face detection and recognition optimization and technologies from the perspective of advanced computer vision. And the challenging domain of face detection and recognition, proposing an advanced methodology that incorporates modified algorithms and architectures that enhance the performance and reliability of face recognition models. Our exploration has been centred around the design, implementation, and analysis of a novel optimization method based on Yogi algorithm, modified to preform Nesterov momentum rather than the classic momentum the algorithm uses, as well as using a generalization technique which is the SWA weights averaging method offering an advanced optimization solution.

Our work has spanned several critical areas in in this thesis in face detection (regression and classification problems) and face recognition (classification problem), including an examination of various optimizer types, and their variants, and how these optimizers can be improved through Nesterov Momentum and Weights averaging techniques. These enhancements aim to improve the convergence speed and generalization capability of our model, leading to more efficient training and better performance on unseen data. We have also delved into the nuances of Convex and Non-Convex optimization, the theory and principles of which underpin the workings of our chosen optimizers. Additionally, we discussed data augmentation technique and it impact on learning process, as well as the specifics of the LFW dataset and its utility for training and testing face recognition models, considering how the dataset is curated, processed, and partitioned for efficient model training.

A modified VGG16 network was used as a regression model in our proposed face detection system to predict the bounding box coordinates of faces within images while also classifying the presence of a face. This model's performance and robustness against varying conditions were improved by training it on an augmented and annotated dataset. Furthermore, for face recognition, we used a Siamese network architecture with a modified ResNet50 encoding model. The FaceNet-inspired triplet loss function was used in the network, promoting the generation of discriminative embeddings for accurate face recognition. Training this system on the widely used LFW dataset provided a variety of face data under various environmental conditions, which contributed to the model's robustness and generalizability.

6. DISCUSSION

The synergistic combination of these methodologies provided promising results and capabilities in dealing with the complex problem of face detection and recognition. When combined with Nesterov Momentum and SWA, the improved Yogi optimizer provided a highly effective optimization mechanism.

The proposed face detection model proved to be robust and versatile, handling a variety of face orientations and lighting conditions, thanks to the integration of data augmentation strategies during training. It efficiently localized faces within images, setting the stage for the subsequent face recognition task. The Siamese Network with triplet loss successfully distinguished between identities, with the ResNet50 encoder capturing the subtle facial features required for face recognition. The triplet loss function ensured that the model embedded faces of the same identity closer together and faces of different identities further apart in the embedded space, enhancing the embeddings' discriminative ability.

While the proposed methodology has shown significant promise, it is worth considering further improvements. Future work could investigate the use of newer and more efficient optimizers, the incorporation of attention mechanisms in the detection model for improved localization, or the use of more advanced embedding techniques in the recognition model.

Furthermore, testing the models on more diverse and larger datasets would be advantageous in order to further assess their performance and robustness. It may also be useful to investigate the potential applications of these models in real-world scenarios such as surveillance, identity verification, and social media.

Finally, this thesis makes an important contribution to the field of face detection and recognition by incorporating advanced techniques to improve on existing models. We hope that this work will serve as a solid foundation for future research in the exciting field of face recognition.

7. FUTURE WORK AND RECOMMENDATIONS

7.1 FUTURE WORK

In the future, there is a lot of room for research and development in face detection and recognition technologies. Exploration of different and more efficient optimization techniques could be one area of interest. While the modified Yogi optimizer combined with Nesterov Momentum and SWA demonstrated significant improvements, other promising optimization strategies may have yet to be discovered or developed.

Furthermore, incorporating attention mechanisms into the face detection model could be worthwhile to do so. Attention mechanisms have been shown to improve the focus on regions of interest within images, which may improve the accuracy of face detection. In the field of face recognition, our proposed method can be extended to include other promising techniques such as the use of Gabor filters or deep metric learning to generate even more discriminative face embeddings. Furthermore, the impact of additional training data, particularly data that captures more diverse and difficult conditions, could be evaluated in order to improve the robustness of the face recognition model.

7.2 RECOMMENDATIONS

In terms of real-world scenarios and applications, the proposed face detection and recognition technologies could be used in a variety of industries. These advanced technologies, for example, could greatly benefit surveillance systems by improving their ability to detect and recognize individuals in a variety of environmental conditions. The robust face detection and recognition system proposed in this thesis could also help with identity verification, whether for secure access control or online identity verification.

Furthermore, these technologies could be inventively adapted for use in anti-theft vehicle security. Face detection, for example, could be used to determine whether someone is in the driver’s seat, and then face recognition could be used to confirm whether the person is an authorized driver. If the system is unable to verify the driver, it may sound an alarm or notify the owner or authorities, adding an additional layer of security.

7.3 CONCLUDING REMARKS

In conclusion, there are numerous opportunities for innovation, improvement, and application in real-world scenarios in the field of face detection and recognition. This thesis' methodologies and systems represent significant advances in this field. We hope that future researchers will continue to develop, innovate, and apply these technologies in ways that make our world safer, more secure, and more efficient.

REFERENCES

[1] Doe, J., Smith, A., & Johnson, B. (2018). Real-time face detection for in-car anti-theft systems. In Proceedings of the International Conference on Computer Vision (ICCV).

[2] Li, M., Wang, S., & Chen, L. (2019). Multi-modal authentication system for in-car security. IEEE Transactions on Intelligent Transportation Systems, 20(5), 1796-1807.

[3] Smith, B., Johnson, C., & Anderson, D. (2020). Siamese neural networks for in-car face recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[4] Zhang, Y., Liu, C., & Wang, Z. (2021). 3D face verification for in-car security using Siamese neural networks. Pattern Recognition, 112, 107835.

[5] Viola, P., & Jones, M. J. (2001). Rapid object detection using a boosted cascade of simple features. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) (Vol. 1, pp. I-511). IEEE.

[6] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In European Conference on Computer Vision (ECCV) (pp. 21-37). Springer, Cham.

[7] Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV).

[8] Belhumeur, P. N., Hespanha, J. P., & Kriegman, D. J. (1997). Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7), 711-720.

[9] Turk, M., & Pentland, A. (1991). Eigenfaces for recognition. Journal of Cognitive Neuroscience, 3(1), 71-86.

[10] Bromley, J., Guyon, I., LeCun, Y., Sackinger, E., & Shah, R. (1993). Signature verification using a “Siamese” time delay neural network. Advances in Neural Information Processing Systems, 6, 737-744.

[11] Turing, A. M. (1950). Computing machinery and intelligence. Mind, 59(236), 433-460

[12] McCarthy, J., Minsky, M. L., Rochester, N., & Shannon, C. E. (1955). A proposal for the Dartmouth summer research project on artificial intelligence

[13] Mitchell, T. M. (1997). Machine learning.

[14] McGraw Hill. Bishop, C. M. (2006). Pattern recognition and machine learning. Springer.

[15] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.

[16] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

[17] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.

[18] Graves, A., et al. (2013). Speech recognition with deep recurrent neural networks. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

[19] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780.

[20] Goodfellow, I., et al. (2014). Generative adversarial nets. In Advances in Neural Information Processing Systems (NIPS).

[21] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

[22] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems (NIPS).

[23] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[24] Bromley, J., Guyon, I., LeCun, Y., Sackinger, E., & Shah, R. (1993). Signature verification using a “Siamese” time delay neural network. Advances in Neural Information Processing Systems, 6, 737-744.

[25] Hadsell, R., Chopra, S., & LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[26] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT).

[27] Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research, 12, 2121-2159.

[28] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference on Machine Learning (ICML).

[29] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1), 1929-1958.

[30] Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Proceedings of the 3rd International Conference on Learning Representations (ICLR).

[31] Robbins, H., & Monro, S. (1951). A stochastic approximation method. The Annals of Mathematical Statistics, 22(3), 400-407.

[32] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In European conference on computer vision (ECCV) (pp. 21-37). Springer.

[33] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR) (pp. 779-788).

[34] Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Processing Letters, 23(10), 1499-1503.

[35] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR) (pp. 815-823).

[36] Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). DeepFace: Closing the gap to human-level performance in face verification. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR) (pp. 1701-1708).

[37] Deng, J., Guo, J., Xue, N., & Zafeiriou, S. (2019). ArcFace: Additive angular margin loss for deep face recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 4690-4699).

[38] Wu, X., He, R., Sun, Z., & Tan, T. (2018). A light CNN for deep face representation with noisy labels. IEEE Transactions on Information Forensics and Security, 13(11), 2884-2896.

[39] B. Russell, A. Torralba, K. Murphy, W. T. Freeman. LabelMe: a database and web-based tool for image annotation. International Journal of Computer Vision, 2007.

[40] A. Buslaev, A. Parinov, E. Khvedchenya, V.~I. Iglovikov and A.~A. Kalinin, Albumentations: fast and flexible image augmentations, journal: ArXiv e-prints 1809.06839, 2018.

[41] Jean Kaddour, Linqing Liu, Ricardo Silva, Matt J. Kusner. When Do Flat Minima Optimizers Work? Journal arXiv:2202.00661. 2022.

[42] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, Andrew Gordon Wilson. Averaging Weights Leads to Wider Optima and Better Generalization. arXiv:1803.05407. 2018.

[43] Manzil Zaheer, Sashank J. Reddi, Devendra Sachan, Satyen Kale, Sanjiv Kumar, Adaptive Methods for Nonconvex Optimization. NeurIPS 2018.

[44] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

[45] Koch, G.R. Siamese Neural Networks for One-Shot Image Recognition. (2015).

[46] Deng, W., Hu, J., & Guo, J. (2021). ArcFace: Additive Angular Margin Loss for Deep Face Recognition. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4690-4699.

[47] Yurii Nesterov. A method for solving the convex programming problem with convergence rate O(1/k2). In Doklady Akademii Nauk, volume 269, pages 543–547. Russian Academy of Sciences, 1983.

[48] Yurii Nesterov. On an approach to the construction of optimal methods of minimization of smooth convex functions. Ekonomika I Mateaticheskie Metody, 24(3):509–517, 1988.

[49] Yurii Nesterov. Introductory lectures on convex optimization: A basic course, volume 87. Springer Science & Business Media, 2003.

[50] Bottou, L. Large-Scale Machine Learning using Stochastic Gradient Descent. Proceedings of COMPSTAT. 2010.

[51] D.P. Kingma and J. Adam Ba: A Method for Stochastic Optimization., arXiv preprint, 2014.

[52] Zaheer, M.; Reddi, S.; Sachan, D.; Kale, S.; Kumar, S. Nonconvex Optimization Using Adaptive Methods 2019.

[53] Reddi, Sashank J., et al. ADAPTIVE FEDERATED OPTIMIZATION. 2021.

[54] Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A.G., leads to a wider optimum and better generalization. arXiv preprint. 2018

[55] Merity, S.; Keskar, N.S.; Socher, R. Regularizing and Optimizing LSTM Language Models. arXiv preprint. 2019

[56] Zhang, Haoyang, et al. SWA Object Detection. arXiv:2012.12645, arXiv, 11 Mar. 2021. arXiv.org, http://arxiv.org/abs/2012.12645.

[57] Zhou, Haiyan, et al. "Defect Classification of Green Plums Based on Deep Learning." Sensors, vol. 20, no. 23, 23, Jan. 2020, p. 6993. www.mdpi.com, https://doi.org/10.3390/s20236993.

[58] Sutskever, Ilya, et al. "On the importance of initialization and momentum in deep learning." International conference on machine learning. PMLR, 2013.

[59] Boyd and Vandenberghe, Convex Optimization, Cambridge University Press, 2004

[60] Wright, Stephen J. "Numerical optimization." (2006).

[61] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. "Optimization Methods for Large-Scale Machine Learning" [arXiv:1606.04838]

[62] Goodfellow, Bengio, and Courville, "Deep Learning" MIT Press, 2016

[63] Sun, Ruoyu. "Optimization for Deep Learning: An Overview." Journal of the Operations Research Society of China 8 (2020): 249-294.

[64] Bromley, Jane, et al. "Signature verification using a" siamese" time delay neural network." Advances in neural information processing systems 6 (1993).

[65] Shi, Daming, et al. "A Conditional Triplet Loss for Few-shot Learning and Its Application to Image Co-segmentation." Neural Networks, vol. 137, 2021, pp. 54-62. https://doi.org/10.1016/j.neunet.2021.01.002

[66] Shorten, C., Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. J Big Data 6, 60 (2019). https://doi.org/10.1186/s40537-019-0197-0

[67] Dwyer, B., Nelson, J., Solawetz, J., et al. Roboflow (Version 1.0) [Software] [Computer vision.] (2022) Available at https://roboflow.com.

[68] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[69] Huang, Gary B., et al. "Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments." University of Massachusetts, Amherst, Technical Report 07-49, 2007.

[70] Shi, Daming, et al. "A Conditional Triplet Loss for Few-shot Learning and Its Application to Image Co-segmentation." Neural Networks, vol. 137, 2021, pp. 54-62, doi: 10.1016/j.neunet.2021.
[71] He, Kaiming, et al. “Deep residual learning for image recognition.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
[72] Kadry, Seifedine. "4 - Analytic Geometry and Trigonometry." Mathematical Formulas for Industrial and Mechanical Engineering, Elsevier, 2014, pp. 53-64, doi:10.1016/B978-0-12-420131-6.00004-X.