EFFICIENT AND SCALABLE DEEP LEARNING BASED FACE AND OBJECT RECOGNITION SYSTEM

by

Vittal Siddaiah

A thesis submitted to the Graduate Council of
Texas State University in partial fulfillment
of the requirements for the degree of
Master of Science
with a Major in Engineering
May 2023

Committee Members:

Semih Aslan, Co-Chair

Damian Valles Molina, Co-Chair

Jesus Jimenez

DEDICATION

This thesis is dedicated to my professors, who have always been my guiding light, inspiring me to strive for excellence and persevere through adversity. Their unwavering belief in my abilities and constant encouragement has been crucial to my academic journey.

I also extend my heartfelt gratitude to my loving family for their camaraderie, love, and laughter. Your presence in my life has made this journey not only bearable but also enjoyable. Your love, understanding, and support have made all the difference in my life. Your companionship and belief in me have provided the motivation to persevere through the challenges of this journey.

Lastly, I dedicate this work to all those who have contributed to my growth and development, both personally and academically. Your support, wisdom, and guidance have been invaluable in helping me reach this milestone.

ACKNOWLEDGEMENTS

First and foremost, I express my deepest gratitude to my thesis advisor, Dr. Semih Aslan, for his invaluable guidance, support, and mentorship throughout this research journey. His extensive knowledge, insights, and unwavering dedication to my progress have been instrumental in shaping my academic growth and completing this thesis.

I am also grateful to the members of my thesis committee, Dr. Damian Valles Molina and Dr. Jesus Jimenez, for their constructive feedback and encouragement during the development of this work. Your expertise and thoughtful suggestions have significantly contributed to the quality of this thesis.

I would like to extend my appreciation to the faculty and staff of the Ingram School of Engineering for providing an enriching academic environment and fostering my intellectual curiosity.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Artificial Intelligence (AI) is the panacea for both prescriptive and predictive analytics through Machine Learning (ML) techniques, demands for computational performance, and snowballing over the decades. Pattern Recognition is increasingly demanding in AI applications that include neural networks-based machine learning. In this research, we are dealing face recognition domain of pattern recognition, popularly termed computer vision. Computer vision enables a wide range of applications spanning across industrial, retail, health care, smart cities in robotics/drones, self-driving cars, augmented reality, optical character recognition, face and gesture recognition, smart Internet of Things, portable/wearable electronics, Law enforcement, and much more. Conventional methods like HAAR and HOG algorithms evolved with improved accuracy; these conventional methods were confined and domain-specific and achieved an accuracy of up to 80% in detection. HAAR and HOG-based algorithms demand expert handcrafting in the design to improve accuracy; they are static and non-scalable. In Deep neural networks (DNN), the algorithms are generic and dynamic. DNN learning enables the model to learn from the data. Traditional learning models are saturated regarding the accuracy, while dynamic Learning improves continually over the quantum of training samples. Today there are DNNs in domains that have achieved over 99% accuracy, which is beyond the ground reality. DNN has established itself as a triumphant set of models for learning relevant connotative representations of data.

Training of deep-learning models is compute-intensive, and there is an industry-wide trend towards hardware specialization to improve performance. This research uses a DNN-based generic, efficient, scalable, and platform-independent framework that can be extendable across platforms. The proposed framework involves computer vision techniques suitable for unsupervised Learning with low latency and high performance. The proposed framework would be open-source, tested across diverse datasets, compatible and scalable across platforms, with low latency and a small footprint. The framework would serve as a benchmark and publish the rating parameters of response times, latencies, and accuracy that grade and differentiates various platforms.

Keywords: Keywords—Artificial Intelligence (AI), Machine Learning (ML), High-Performance Computing (HPC), OpenCV, OpenVINO, OneAPI, Computer Vision, Convolutional Neural Network (CNN), Deep Neural Network (DNN), Industry

1. INTRODUCTION

This research is a manifest of Deep Neural Network-based algorithms to recognize faces and objects in real-time. Face recognition classification is either feature-based, appearance-based, or template-based. Feature-based techniques attempt to find the locations of distinctive image features such as the eyes, nose, and mouth and then validate whether the elements are in a probable geometrical pattern. These techniques include some new strategies with fundamental approaches to the face recognition field[1].

Template-based approaches, such as active appearance models (AAMs)[2], can deal with the broad range of pose and expression variability; typically, they require explicit initialization near a natural face and are, therefore, unsuitable for fast face detectors. Appearance-based approaches scan over the image's small, overlapping rectangular patches, searching for likely face candidates, which refines using a cascade of more expensive but selective detection algorithms.

1.1 Classification of Learning Methodology

Holistic Learning is which treats the image as is as a whole to generate low-dimensional representation. Most commonly used in EigenFace and FisherFace detection methods.

Specialized Learning is when developers and scientists develop localized features for generating a low-dimensional representation. There were non-generic but confined to a particular domain or application, popularly observed in LBP[3], LBPH, and Gabor methods.

Shallow Learning is the amalgamation of the two methods churning into a hybrid. This hybrid method enhanced accuracy. In this method, features are learned independently with arbitrary function approximators. (Shallow networks would have one hidden layer)

Deep Learning is an improved form of shallow Learning, where all the learnings are within Neural Networks, and the key to deep Learning is that features are learned from the data itself. Deep learning approaches have reached a very high level of accuracy, and all modern methods use deep Learning in their pipeline. (*Deep Learning would have two or more hidden layers*)

1.2 Problem Statement

The problem addressed in this thesis is achieving efficient, scalable, and swiftly extendable face and object recognition for implementing a deep learning-based high-performance system. This benchmarking not only compares the performance of different platforms running a broad range of deep learning models but also supports a more profound analysis of the interactions across a broad spectrum of diverse model attributes (hyperparameters) and hardware configuration choices, and software support.

1.3 Proposed Solution

Create a baseline with face and object detection as a domain that would be a benchmark to execute across available hardware environments with the following framework characteristics:

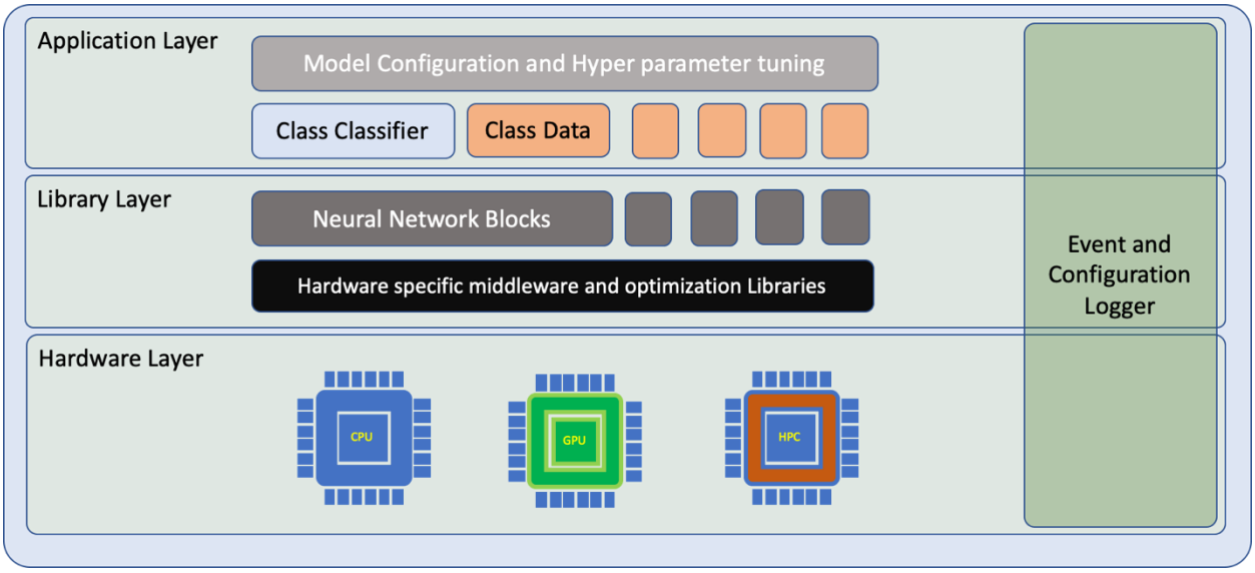

Figure 1 – Proposed Framework

The proposed framework consists of three layers:

Application Layer: This layer would abstract the implementation below with a generic interface for model tuning

Library Layer: The library layer contains the latest and optimized library and drivers, including firmware and other middleware.

Hardware Layer: These above two layers are optimized to the underlying hardware based on the architecture implementations that include a number of computing units, the capability of computing units, cache size, hierarchy, etc.

1.4 Hypothesis

Implementing in C++ with custom controlled optimization of a DNN would yield low latency, highly scalable, swiftly portable generic framework that can be used as a benchmark to rate platforms.

1.5 Research Contribution

The face detection baseline created with the framework described in section 1.7 would be an open-source benchmark that can be used to qualify the system.

1.6 Novelty

There are several open-source and commercial frameworks, but most of them are limited to a particular architecture. High-efficiency algorithms are proprietary.

		Our Framework	Others
	Python	✓	✓
	C++	✓	!
OS Optimized	MacOS	✓	!
	Linux	✓	!
	Windows	✓	!
Hardware Optimized	CPU Only	✓	✓
	GPU Only	✓	✓
	Hybrid	✓	✗
	HPC	✓	✗
Bench marks	Common	✓	✗
	Production Ready	✓	!

Table 1– Framework Features

1.7 Framework characteristics with baselines

Derive a framework with the following baselines.

- ***Efficient***: To maintain the inference accuracy of up to 90%, today, for specialized cases (frontal face recognition), there are inference algorithms that achieve a recorded accuracy of over 99%[4]. However, accuracy is lower in object detection, dark face-face recognition, and other areas. This thesis aims to provide multiple models to the user under the same framework. The proposed framework would enable users to analyze accuracy parameters on the same scale, such as accuracy parameters, hyperparameters, error rate, etc.

- ***Scalable***: Computing systems today consist of a resource-rich with CPU, GPU, and memory, creating a distributed computing environment field[5]. This thesis proposes a framework that can use the available resources efficiently with a precise definition of architectural details[6][7].

- ***Platform Independent:*** The proposed framework will be swiftly portable across Operating Systems, *Linux, MacOS®, and Windows®.* The proposed framework would be capable of independent development of training and inference models[8]. With fine-grained power profiling and scientific motif composition for multi-chip servers[9]. The proposed framework would showcase state-of-the-art compiler optimizations based on heuristics[10].

The above framework is achieved through the following:

- *Decrease in inherent latencies through pipelining and parallelism at all phases*

- *Boost response times in both Learning and inferencing*
- *Achieve real-time inferencing of more than one subject*
- *Create C++-based, portable, scalable, low latency, high-performance libraries for subject detection.*
- *Analyze various image classification methods applicable to the Convolutional Neural Networks (CNN) that demand a fixed resource budget and then scale up with resource availability for better accuracy.*

1.8 Typical applications by use cases:

Industrial:

Auto asset Inspection[11], Automation Machine Vision[12][13], Dynamic Object Detection[12], Visual Robotics[14], Predictive positioning[15], Surveillance[16], all industries that are Industry 4.0 compliant[17].

Retail:

Inventory Management[18] (HEB, Amazon), Point of Sale, Theft Protection[19]

Smart Cities:

Traffic Management, Law Enforcement[20], Public Safety, Autonomous, and Self Driving vehicles[21]

Health Care:

Enhanced Swift Diagnostics, Remote Patient Monitoring[22], Bed Detection for Hospital Patient Monitoring System[23], Drug Discovery [24], Predictive Care[25][26]

Agriculture:

Beehive Monitoring[27],

Others:

Hand Keypoint [28]and Gesture Detection

2. BACKGROUND

Computer Vision enables Machine Vision, and Machine Vision has a significant role to play in the smart factories of tomorrow, as well, where automation in manufacturing lines would allow optimizing to maximize throughput with enhanced quality and profitability. Smart factories are the demand of the day, and it is no more a dream concept; it is a baseline for ***Industry 4.0***.

Today Deep Learning has achieved over 98% accuracy in vision detection, while some commercial companies have reached up to 99.88% accuracy and within a standard error of 0.0004 on Unrestricted, Labeled Outside Data Results[29]. All the algorithms have proven performance and have exceeded human capability and perceptions. The next level of challenges is:

2.1 Challenge 1

Training a subject domain is critical, complicated, and challenging. It would take a few days to weeks of computation based on target domains and the available resources.

2.2 Challenge 2

The efficacy of the training depends on both hardware and software, a perfect match of hardware utilization of available hardware resources for the software stack; most of the software stack is one-off and confined to a particular target hardware system.

- An application that needs large-resolution like medical centers (CT-scans, MRI) and Space Research, where the images are very massive, poses a challenge.

- A typical whole-slide image is approximately 200000 x 100000 pixels on the highest resolution level with a three-byte RGB pixel format. This means 55.88GB of uncompressed pixels of data from a single level[30][31]. We cannot lower the resolution nor resize it into patches, as it results in a loss of data and induces false positives. (Memory, Computing, and Storage are all challenged) Patch Size Vs. Accuracy has been a tradeoff.

2.3 Challenge 3

There are several image classification models, and the challenge is the selection of methodology with which we achieve smaller models with higher accuracy is the demand of the day. GPU-based systems fail when the data rates are higher and rely on CPU-based high-performance systems.

2.4 Challenge 4

Although DNN characteristics accomplish higher accuracy compared to HOG, DNN comes at a massive overhead in energy consumption (~300x-13500x)[3]. The order of magnitude in this gap is the manifest of the fact that DNN architecture is programmable/flexible/non-static. DNN involves more computation, substantially inducing latency in the system. This research is to learn the DNN pipeline and design an optimal framework and achieve the following goals.

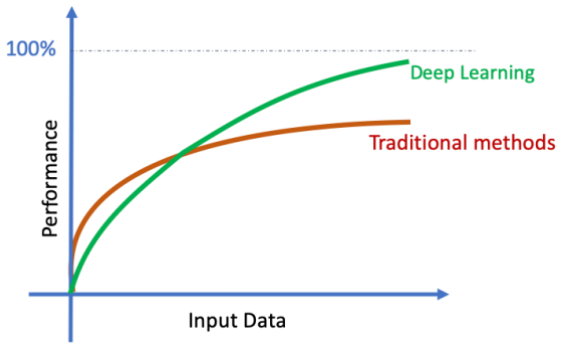

Figure 2 - Traditional Vs. Deep Learning Methods

2.5 DNN in computer vision classification and detection

Deep Neural Network (DNN) based computer vision method outperforms traditional methods. The learning curve in Deep Learning does not saturate, unlike conventional techniques, and it keeps progressing with more data. Traditional methods like HAAR and HOG[32] are domain-specific, while DNN learns irrespective of areas and yields high-performance results. HAAR and HOG might have to be tuned differently for pedestrian detection to direct face detection. HAAR and HOG require a lot of trial and error iterations, and in HAAR and HOG methods, there are more false positives.

Figure 3 – Training based on triplet loss

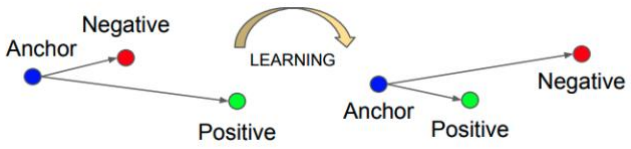

Figure 4 – Triplet loss relationship w.r.t anchor distance

This research proposes the triplet-based network model, which ensures faster convergence and strives to learn valuable observations by relative Euclidean-distance measurements. A comparable model defined by Wang et al. (2014), and Elad Hoffer & Nir Ailon (Dec 2018)[33] confines for learning a ranking for image information retrieval. The procedure is to pick three photos, of which two are from the same source and one different. The triplet loss is high between the two images of the same source, while the distance/triplet loss is minimal across images from different sources.

However, after training, we see the coefficients of the image from the same sources get clustered together. This research demonstrates using diverse datasets that the model learns a better representation than that of the Siamese network. The proposed framework involves computer vision techniques suitable for unsupervised Learning with low latency and high performance.

If the embedding is represented by:

$$f(x) \in R^d \tag{1}$$

Where an image x into a d-dimensional Euclidean space. The constraint on embedding on a d-dimensional hypersphere is given by

$$\|f(x)\|_2 = 1 \tag{2}$$

The described formula in Equations (1) and (2)

Here an image x^a_i (*anchor*) of a specific person is closer to all other images x^p_i (*positive*) of the same person than it is to any image x^n_i (*negative*) of any other person. Thus the expectation is

$$\left\|f(x_i^a) - f\left(x_i^p\right)\right\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2,$$
$$\forall \left(f(x_i^a), f\left(x_i^p\right)\right) \in T \tag{3}$$

Where α is a margin that is enforced between positive and negative pairs. τ is the set of all possible triplets in the training set and has cardinality *N*.

The loss minimized is given by *L*

$$\sum_i^N \left[\left\|f(x_i^a) - f\left(x_i^p\right)\right\|_2^2 + \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha\right] \tag{4}$$

Triplet loss-based deep learning transcends 95% accuracy (more notable compared to human beings). HOG features are static, while DNN is dynamic. It would need handcrafting to improve the performance further in traditional methods (loses flexibility and becomes domain-specific). Hence DNN is dynamic, and progressive Learning is mighty powerful. Although DNN characteristics accomplish higher accuracy compared to HOG, DNN comes at a massive overhead in energy consumption (~300x-13500x)[34]. The order of magnitude in this gap is the manifest of the fact that DNN architecture is programmable/flexible/non-static. DNN involves more computation, substantially inducing latency in the system.

The literature surveyed[34] reveals the hardware specification and measured performance of couple designs for feature extraction, as shown in Table 2 For CNN, Eyeriss programmed to run two CNN models (five convolutional layers of AlexNet and thirteen convolutional layers of VGG-16) to confirm the hardware performance contrasts of operating different CNN features.

Eyeriss delivers almost the same computation throughput (i.e., GOPS) (Giga Operations Per Second) as the HOG design when executing AlexNet features; HOG performs 35x more input PPS (pixels per second); the rift is even more significant between HOG and VGG-16 features. This gap is due to the variations in computational complexity (i.e., operations per pixel) between different functions, as shown in Table 3. However, the HOG design also utilizes around 10x lesser power than Eyeriss. Hence, the HOG hardware consumes 311x and 13,486x less energy per pixel than Eyeriss operating AlexNet and VGG-16 features, respectively. Regarding energy per operation, the HOG hardware is 10× less than Eyeriss.

Table 2 - Relative Hardware specification and performance parameters

Implemented Feature	HOG	CNN (AlexNet)	CNN(VGG-16)
Technology	65nm	65nm	
Gate Counts (kgates)	893	1176	
Memory (kB)	159	181.5	
Multiplier Bitwidth	5×11 – 22×22	16×16	
Throughput (Mpixels/s)	62.5	1.8	0.04
Throughput (GOPS)	46	46.2	21.4
Power (mW)	29.3	278	236
DRAM access (B/pixel)	1	74.7	2128.6
Energy Efficiency (nJ/pixel)	0.5	155.5	6742.9
Energy Efficiency (GOPS/W)	1570	166.2	90.7

Table 3 - Computational Complexity

Feature		GOP/MPixel	Ratio
Hand-crafted	HOG	0.7	1.0x
Learned	CNN (AlexNet)	25.8	36.9x
	CNN (VGG-16)	610.3	871.9x

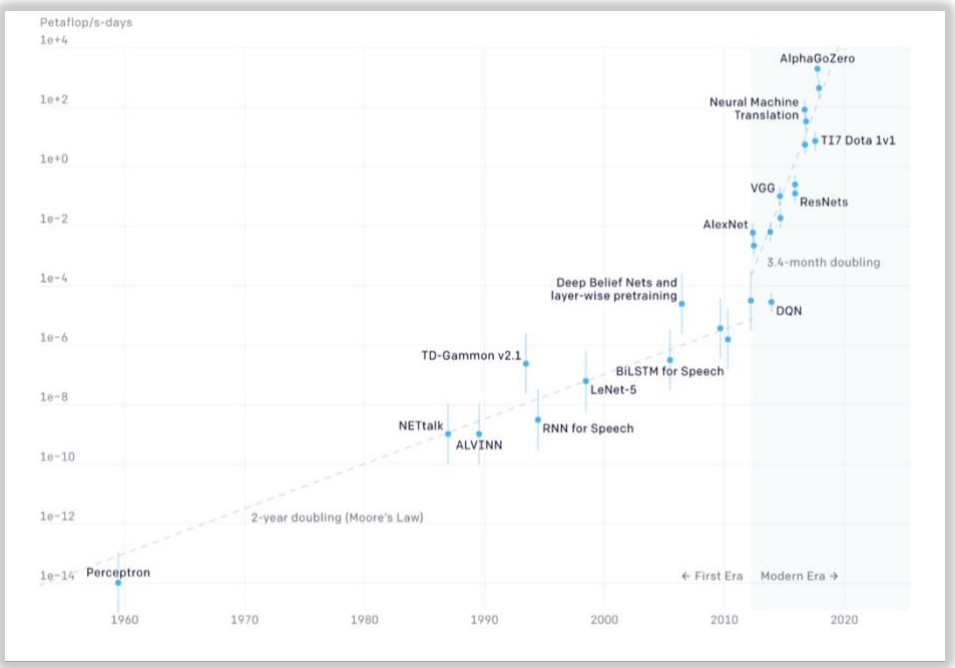

Figure 5 - Two Distinct Eras of Compute usage in Training AI Systems

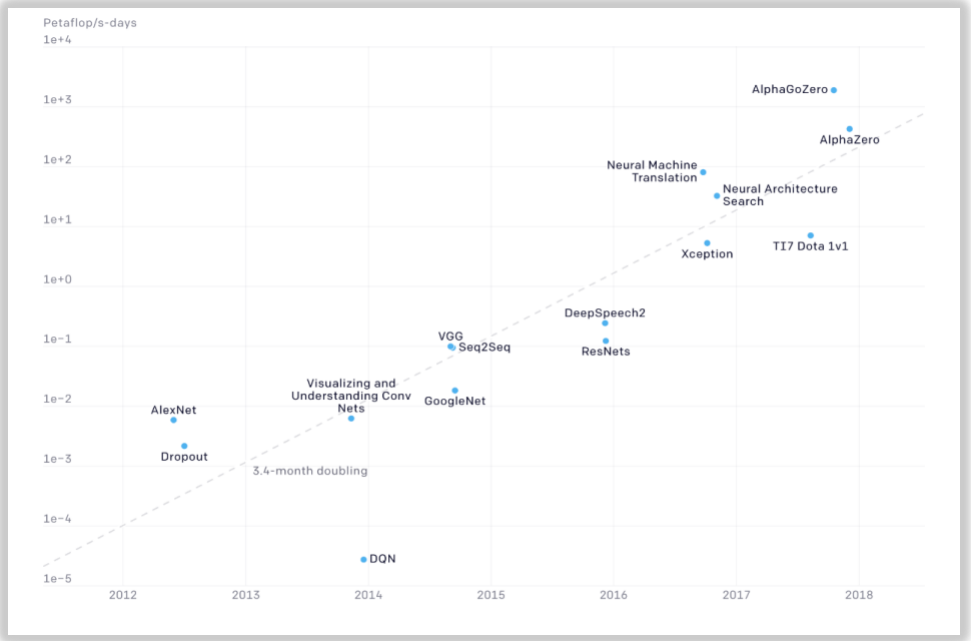

Figure 6 - AlexNet to AlphaGo Zero: A 300,000x increase in Compute (Log Scale)

The demand for computing over time is exponentially increasing; refer to Figure 5 and Figure 6

3. LITERATURE REVIEW

Over a couple of decades, there are various types of research conducted on face recognition and inferred with multiple algorithms, including Principal Component Analysis(PCA)[35], LDA, Histogram of Oriented Gradients (HOG)[6], Local Binary Pattern Histogram (LBPH), algorithm Sparse Coding algorithm. One of the most famous face detection algorithms is the Viola-Jones algorithm[7], and it is the basis of many other face detection techniques. This algorithm uses Haar-like[8] rectangle features to detect a face. They have used an image size of 320x240 pixels and got comparatively inferior performance concerning the software-based system executed on CPUs. Chakrasali and Kuthale [36] used Haar features and the AdaBoost algorithm for face detection.

In Principle Component Neural Network (PCNN) based face recognition system, is capable of recognizing up to 1400 faces in an image frame and is suitable for access control and video surveillance. They have presented a face recognition system with a real-time image of a low-resolution of 32x32 pixels only. Lately, Schaffer has a face recognition system that utilizes a software-based Viola-Jones face detection algorithm and FPGA-based PCA algorithm for the face recognition part. They have reported that real-time face recognition at an accuracy of about 95% and can process 13026 faces per second. In this system, 20 face images from each of the 153 people have been considered for the recognition database.

Zhao and Wei exhibited MLBPH based real-time face recognition system with various facial deflections and attitude and achieved recognition accuracy of about 48-55% for 300 angular positions of the face. Ahmed used LBPH architecture for face recognition at a low resolution of 35px. Using 500 images per person in the database, they have reported a recognition efficiency of 94% at 45px and 90% at 35px of the input image. PCA vs. low-resolution recognition system has

been shown in [18] where the image resolution was 17x18 pixels, but they have yet to present real-time recognition.

In our proposed method of face recognition, we have utilized Local Binary Pattern Histogram (LBPH) algorithm, Principle Component Analysis (PCA) based Eigen face algorithm, and Fisherface algorithm for real-time face recognition showing a reduced number of face images in the database with improved recognition accuracy even with various attitude deflection. We have used two sets of created image databases.

Local Binary Pattern (LBPH), an effectual feature for texture classification in computer vision, is a simple yet very efficient operator to describe the contrast information of a pixel concerning its neighboring pixels. It labels the pixels of an image by thresholding the neighboring pixels and considers the result as a binary number. When combined with the Histograms of Oriented Gradients (HOG) descriptor, LBP considerably improves performance on some datasets. LBP combined with the HOG descriptor can represent a facial image as a simple data vector and be used for face recognition purposes.

The Fisherface algorithm is another famous face recognition algorithm that originated in 1997. It is built upon Eigenfaces and based on the Fisher Linear Discriminant Analysis (FLDA) derived from Ronald Fisher's linear discriminant analysis (LDA) technique. Although both PCA and LDA use linear projection for dimension reduction, the results are highly dissimilar.

LDA maximizes the ratio of the between-class scatter matrix and within-class scatter matrix. For this reason, the Fisherface algorithm has a limited effect on the classification process involving PCA under different lighting conditions of the image.

Comparing Deep Neural Networks (DNN) with traditional methods Deep Neural Networks

Workflow pipelines HOG is static; it is with a fixed descriptor. They are hand-crafted and hence become confined to a domain, the result being non-generic

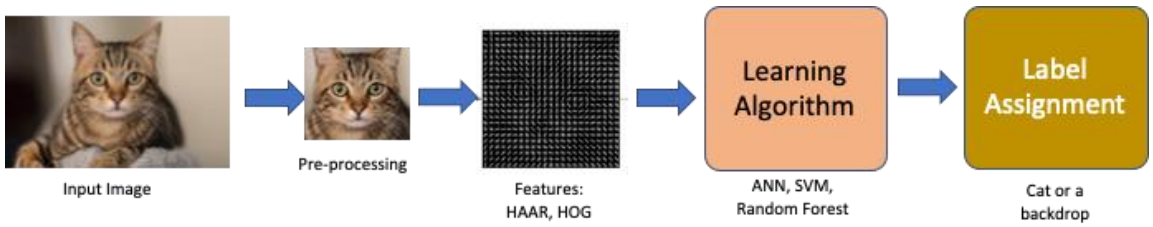

Figure 7 - Traditional WorkFlow

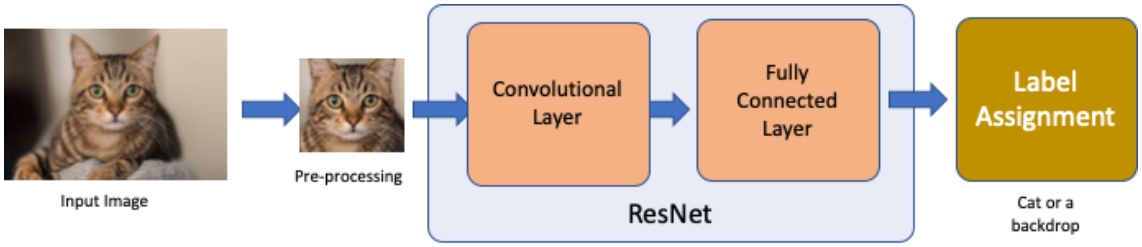

Figure 8 - ResNet (Bank of CNN) Workflow Pipeline

In this research, we enhance face recognition's discriminative and generalization ability, and we adapted VarGFaceNet (Variable Group Convolutional Network)[37]. VarGNet resolves the ambivalence between small computational costs and the unbalance of computational intensity inside a block.

The power of the Active Appearance Models (AAM)[2] lies in its behavior of the mean intensity error, as shown in Figure 9- Proportion of searches to initial displacements as we continually train multiple models.

The search and error reduction enable active clustering. Moreover, the proportion of searches converges rapidly during inference with lower iterations. The deductions are much faster than the other algorithms in the domain.

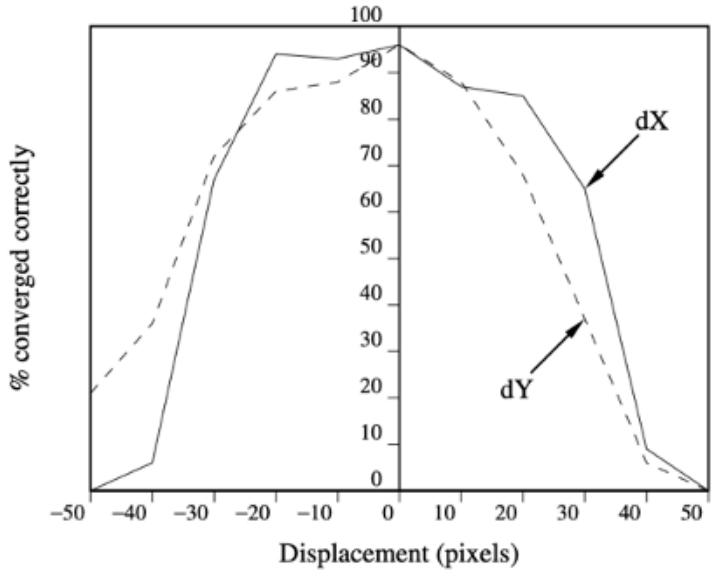

Figure 9- Proportion of searches to initial displacements

3.1 Accuracy and performance comparisons

Accuracy and performance comparisons between OpenFace[38] as shown in Figure 10 - Classification of accuracy with the quantum of training and prior non-proprietary face recognition implementations (from OpenCV)

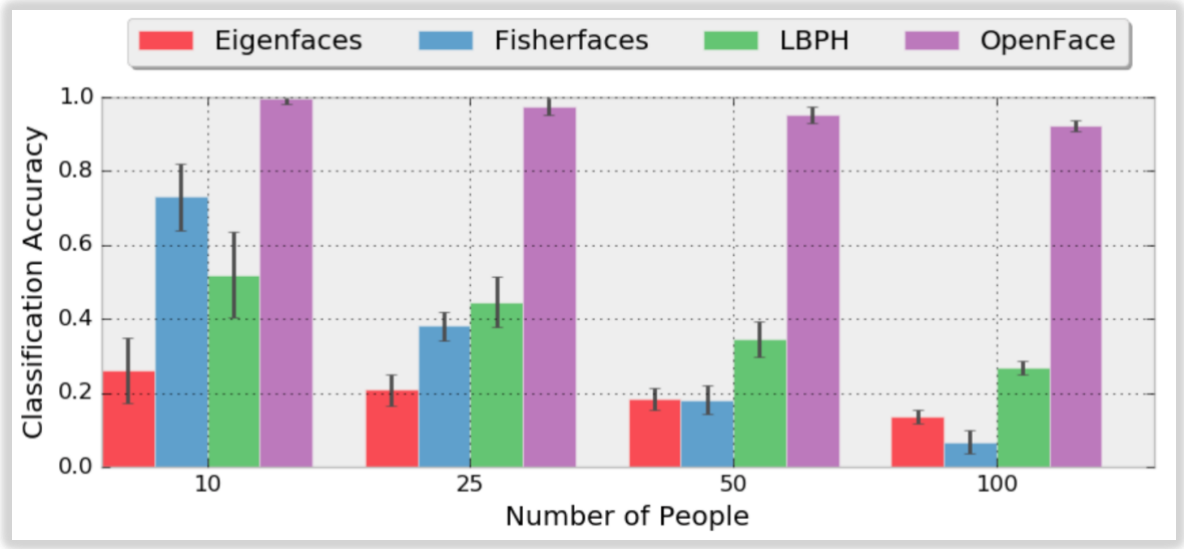

Figure 10 - Classification of accuracy with the quantum of training

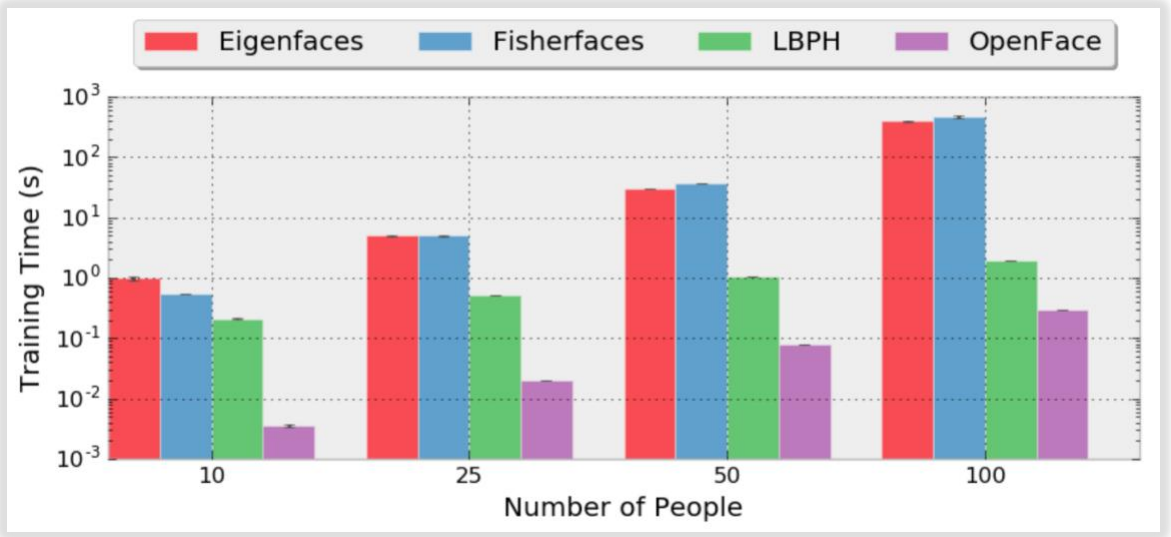

Figure 11 - Training Time w.r.t quantum of images

Performance and Accuracy are not correlated (refer to Figure 12). There is an impact of non-linearities and various residual connections. Substituting one matrix multiplication with different smaller ones hurts runtime performance due to progressed cache misses. This approach is the most effective, with t being a low constant between 2 and 5. It significantly reduces the memory requirement but still allows one to utilize most of the efficiencies gained by using highly optimized matrix multiplication and convolution operators provided by deep learning frameworks. It remains hidden if particular framework-level optimization may lead to further runtime improvements.

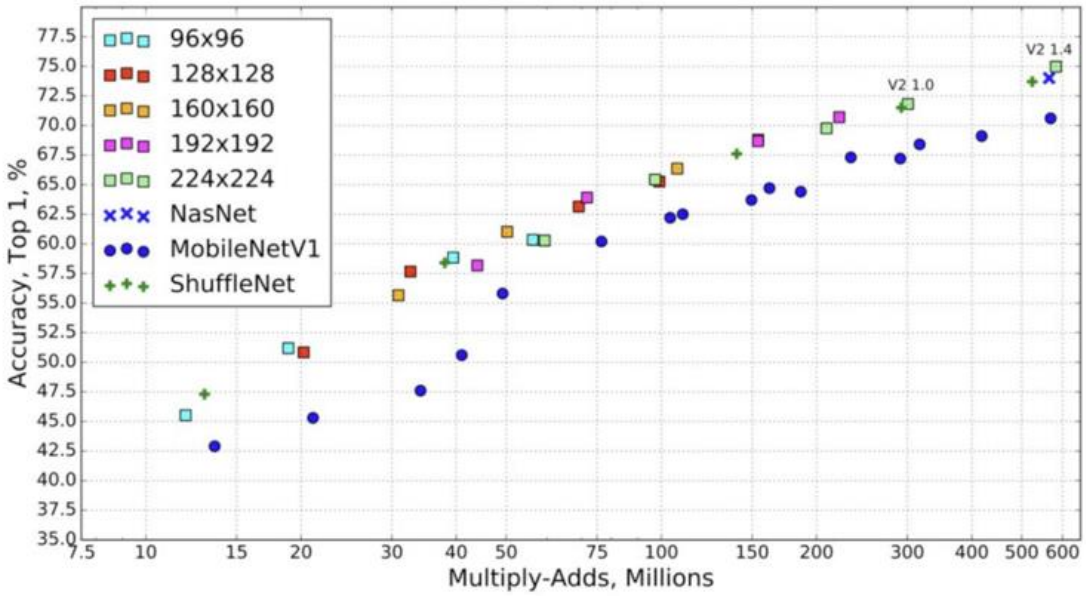

Figure 12 - Performance Vs. Accuracy

4. METHODOLOGY

DNN is the most accurate, while the training time is costly and compute-intensive. Our research proposes that the training be done on an HPC pool, and the inference could be made on a remote setup.

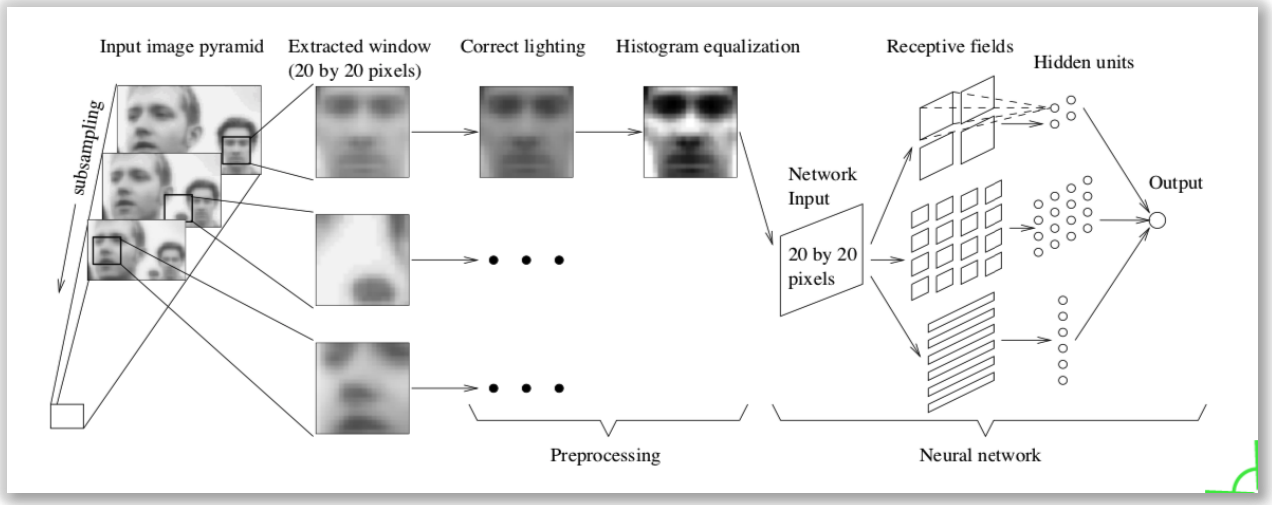

Figure 13 - Neural network-based filter and merging

Based on the IEEE paper, Neural Network-Based Face Detection based on an IEEE paper from Henry A. Rowley, Shumeet Baluja, and Takeo Kanade operates in two stages, The Neural network-based filter and merging overlapping detection and arbitration, as shown in Figure 13.

The hierarchical design stitches together pixels into an invariant face representation. As depicted in Figure 14 - Neural network-based filter and merging, the deep model consists of various complicated layers of simulated neurons that convolute and pool input, during which the receptive-field size of simulated neurons is enlarged to integrate the low-level primary elements into different facial attributes, eventually feeding the data-forward to one or more fully connected layer at the head of the network. The output is a compressed feature vector that represents the face. Such deep

representation is popularly considered the state-of-the-art technique for the face recognition field[39].

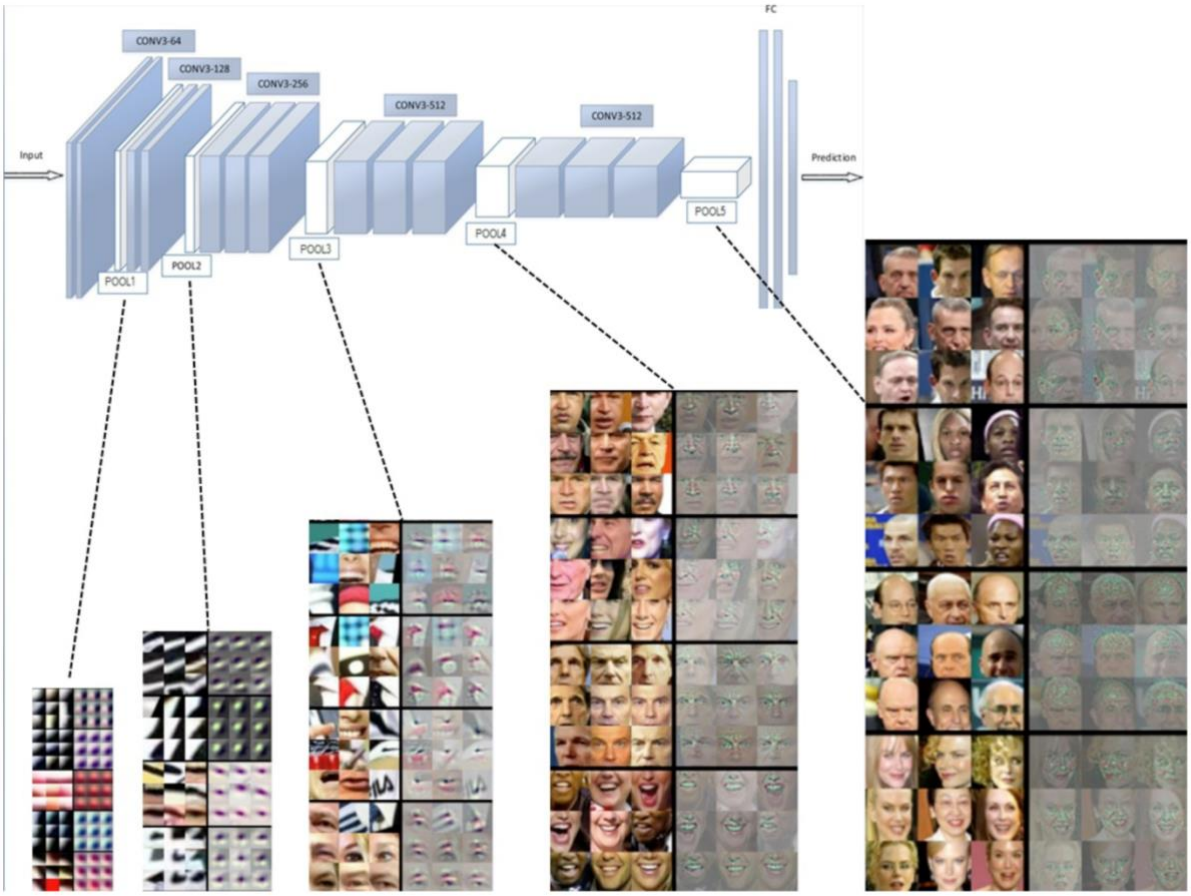

Figure 14 - Neural network-based filter and merging

The evolution of loss functions defines and marks the origin of deep-FR that Deepface[40] and DeepID [41] introduced in 2014 (refer to Figure 15). Following that, Euclidean-distance-based loss ever played an essential role in the loss function, such as contractive loss, triplet loss, and center loss. In 2016 and 2017, L-softmax[42] and A-softmax[43] additionally promoted the

development of significant margin feature learning. In 2017, feature and weight normalization also started to show exceptional performance, which led to the study of variations of softmax.

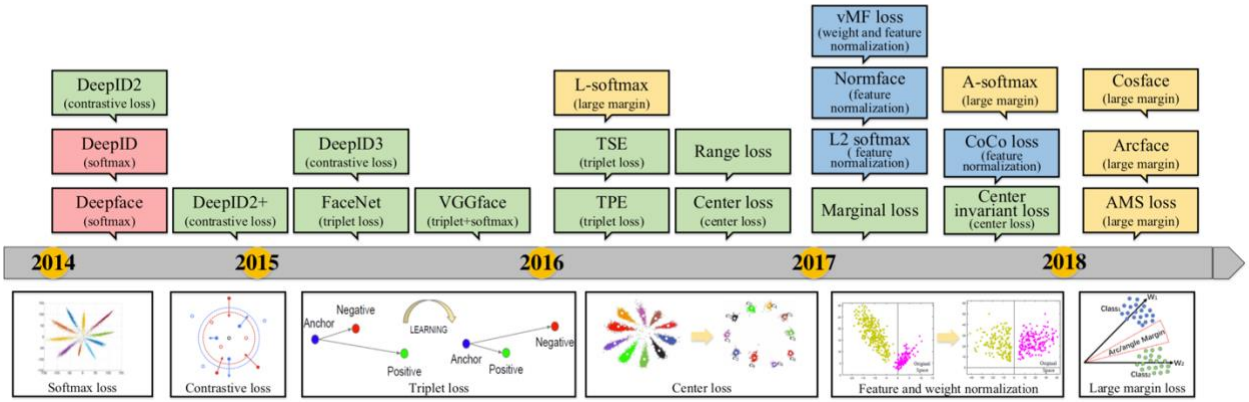

Figure 15 - Time Scale of Loss Functions Definition

Python, along with SciKit learn, OpenCV, and DLib libraries, will be used to develop and deploy the initial prototype. C++-based implementation would be used for production quality, low latency, and low footprint package.

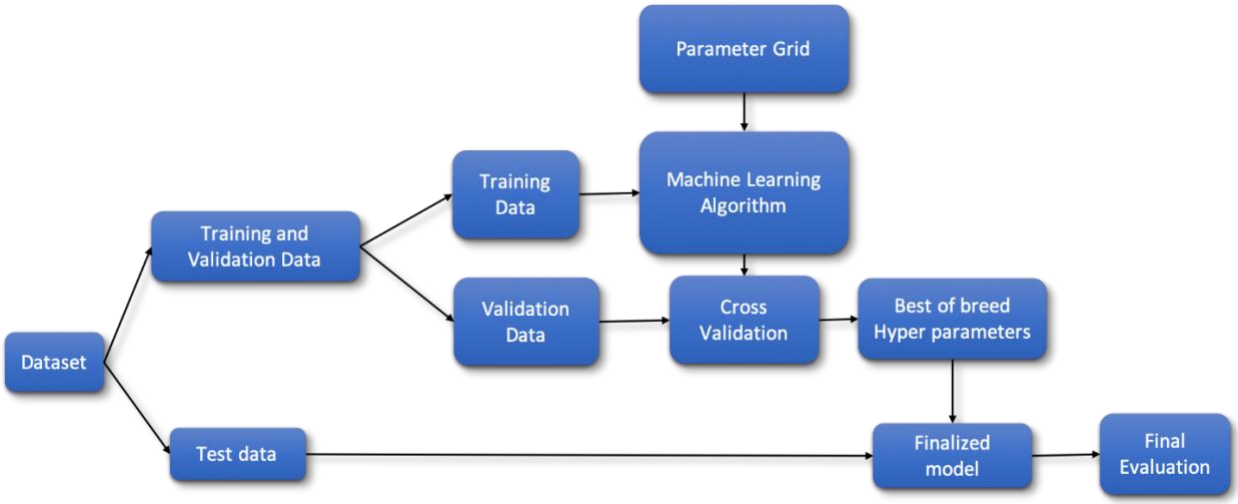

Figure 16 - Training and Optimization Workflow

5. EXPERIMENTAL SETUP

5.1 Preprocessing Phases

Image preprocessing can significantly enhance the integrity of optical inspection. Various filtering techniques are adopted that further enhances or reduce particular image information. Image preprocessing decreases processing time, increases accuracy, and enables a feature extraction field[44].

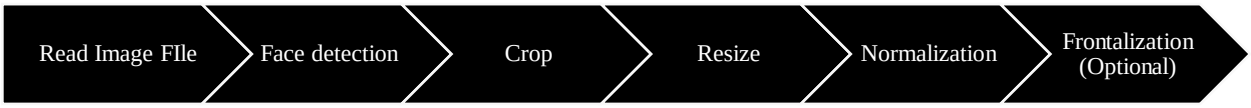

Figure 17 - Preprocessing Phases

Reading Or Parsing Of Image

The incoming files into the system could be in various formats. The design should be capable of reading more than one file format and capable of decoding the image information from the same.

Face Detection: There could be several images being scanned in the picture frame along with the background; In this phase, the region of the image which correlates to a face(s) based on the face-boundary-detected feedback, face regions are detected fed to the cropping stage.

Cropping the face: The image region where the frontal face is identified is cropped, and only this cropped region is used in the face recognition process. All the new images need similar dimensions.

Therefore, after cropping the face from the original image, the new images are normalized to a standard size of m x n pixels.

Resizing: Images are resized into various scales to determine the impact of the size variances. Every resized image carries different information; hence, the best image size needs detailed analysis.

Normalization:The rationale of face normalization is to reduce the impact of irrelevant and redundant information from the image, which increases accuracy. Several techniques, like geometric normalization and photometric normalization, are used for facial recognition in controlled environments to adapt these techniques to un-controlled ones [45].

Frontalization: Even though there are a plethora of modern approaches in face recognition using DNN, we observe a significant drop in accuracy for pose variations in unconstrained environments. "Frontalization" is the process of synthesizing frontal-facing views of faces appearing in single unconstrained photos[46]. Frontalization can significantly boost face recognition performance and help recognize faces in the wild[47]. Frontalization is optional but an instrumental step when analyzing images in the wild.

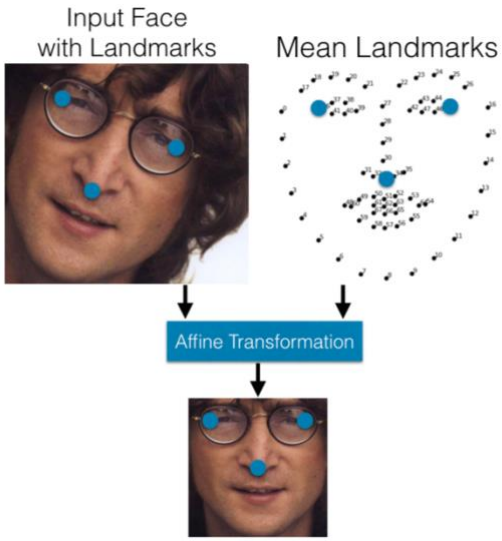

Figure 18 - Affine Transformation in Preprocessing

Figure 18 - Affine Transformation in Preprocessing depicts how the Affine Transformation[48] normalizes faces. The detected landmarks with affine transformation make the eye corners and nose close to the mean locations. The affine transformation also resizes and crops the image to the edges of the landmarks, so the input image to the neural network is with the desired size.

5.2 Experiment-1: Face Detection

This research is a manifest of Deep Neural Network-based algorithms to recognize faces and objects in real-time. Face recognition is classified as feature-based, appearance-based, or template-based. Feature-based techniques attempt to find the locations of distinctive image features such as the eyes, nose, and mouth and then validate whether the elements are in a probable geometrical pattern. These techniques include some of the new strategies to face recognition. These techniques include some of the fundamental approaches to face recognition (Neural Network-Based Face Detection [49])

Today Deep Learning has achieved over 98% accuracy in vision detection. In comparison, some commercial companies have reached up to 99.88% accuracy and within a standard error of 0.0004 on Unrestricted, Labeled Outside Data Results[50]. All the algorithms have proven performance and have exceeded human capability and perceptions. The next level of challenges is:

Challenge-1: Training a subject domain is critical, complicated, and challenging. It would take a few days to weeks of computation based on target domains and the available resources.

Challenge-2: Efficacy of the training depends on both hardware and software, a perfect match of hardware utilization of available hardware resources for the software stack; most of the software stack is one-off and confined to a particular target hardware system. An application that needs large-resolution like medical centers (CT-scans, MRI) and Space Research, where the images are very massive, poses a challenge. A typical whole-slide image is approximately 200000 x 100000 pixels on the highest resolution level with a 3-byte RGB pixel format. This means 55.88GB of

uncompressed pixel data from a single level. We cannot lower the resolution nor resize it into patches, as it results in a loss of data and induces false positives. (Memory, Computing, and Storage are all challenged) Patch Size Vs. Accuracy has been a tradeoff.

Challenge-3: There are several image classification models, and the challenge is the selection of methodology with which we achieve smaller models with higher accuracy, the demand of the day. GPU-based systems fail when the data rates are higher and rely on CPU-based high-performance systems.

Challenge-4: Although DNN characteristics accomplish higher accuracy compared to HOG, DNN comes at a massive overhead in energy consumption (~300x-13500x). The order of magnitude in this gap is the manifest of the fact that DNN architecture is programmable/flexible/non-static. DNN involves more computation, substantially inducing latency in the system. This research is to learn the DNN pipeline and design an optimal framework and achieve the following goals.

Framework characteristics with baselines

Derive a framework that is Extremely Efficient: The baseline is to maintain an inference accuracy of over 95%. Highly Scalable: with a baseline maintaining scalability with system resources (multiprocessing, multithreaded system). Easily Extendable: Ensure the framework can be swiftly portable across platforms; Intel® and NVDIA® platforms are under consideration.

Based on the IEEE paper, Neural Network-Based Face Detection based on an IEEE paper from Henry A. Rowley, Shumeet Baluja, and Takeo Kanade[49], it's a two-stage operation, The Neural network-based filter and merging overlapping detection and arbitration.

The optimized neural network resulted in almost 99% accuracy in detecting faces. Actress Angelina Jolie was enrolled along with other celebrities. Users were told to select any picture of Angela Jolie from the wild (These pictures were not trained/enrolled, but we are to predict the celebrity name).

The optimized neural network resulted in almost 99% accuracy in detecting faces. Actress Angelina Jolie was enrolled along with other celebrities. Users were told to select any picture of Angela Jolie from the wild (These pictures were not trained/enrolled, but we are to predict the celebrity name).

Case 1:

The subject was picked up randomly from the web, a magazine cover page. The model recognized the subject accurately.

Inference time: 81 msec

Case 2:

The subject is not facing the camera, and the face is tilted to the side, and the model recognizes the subject accurately.

Inference time: 24 msec

Case 3:

The next challenge was to pick the subject in disguise. Our DNN model was still able to recognize the face accurately.

Inference time: 54 msec

5.3 Experiment-2: Image Stitching

Image stitching is a complex process that requires a combination of algorithms and techniques, and the code can be quite involved. Here is a general overview of the steps involved in image stitching:

Image alignment: Determine the transformation required to align the input images, such as translation, rotation, or scaling. Standard image alignment techniques include RANSAC (Random Sample Consensus) and homography estimation.

Image blending: Blend the aligned images to create a seamless final stitched image. Common blending techniques include multi-band blending and feathering.

Feature detection and matching: Detect key features in the input images, such as corners, edges, or blobs, and match them across images to determine the relative position and orientation of each image. Popular feature detection algorithms include SIFT (Scale-Invariant Feature Transform) and SURF (Speeded Up Robust Features).

The incorrect matching of images may significantly impact the results of the geometric transformation model. To increase recording speed and precision, RANSAC is the acronym for "Random Sample Consensus." This filtering algorithm was published by Fischler and Bolles in 1981[51]. It is a non-deterministic algorithm because it does not ensure to return of acceptable results; the probability of success increases if more iteration is made to discard the false matches.

RANSAC is an iterative method used by a group of observed data, which contains outliers for estimating a mathematical model's parameters and finding the best-fit results.

Here's a general code structure for image stitching using OpenCV:

```
import numpy as np
import cv2

# Read in input images
image1 = cv2.imread('image1.jpg')
image2 = cv2.imread('image2.jpg')


# Identifying key points and descriptors for the given image
detector = cv2.xfeatures2d.SIFT_create()
key_point_img1, descriptor_img1 = detector.detectAndCompute(image1, None)
key_point_img2, descriptor_img2 = detector.detectAndCompute(image2, None)


# Match features between images
matcher = cv2.FlannBasedMatcher()
matches = matcher.knnMatch(descriptor_img1, descriptor_img2, k=2)


# Apply ratio test to filter matches
good_matches = []
```

```
for m, n in matches:  # ratio test as per Lowe's paper[52]
    if m.distance < 0.7 * n.distance:
        good_matches.append(m)


# Homography using RANSAC estimation
if len(good_matches) > 10:
    src_pts = np.float32([key_point_img1[m.queryIdx].pt for m in good_matches]).reshape(-1, 1, 2)
    dst_pts = np.float32([key_point_img2[m.trainIdx].pt for m in good_matches]).reshape(-1, 1, 2)
    M, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC, 5.0)
    # Apply homography to align images
    aligned_image = cv2.warpPerspective(image1, M, (image2.shape[1], image2.shape[0]))
    # Blend aligned images using multi-band blending
    final_image = cv2.seamlessClone(aligned_image, image2, mask, (0, 0, 0), cv2.NORMAL_CLONE)


# Display final stitched image
cv2.imshow('Stitched Image', final_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Homography in image processing refers to a mathematical transformation that maps points in one image to corresponding points in another. Homography is a critical concept in image stitching, where it is used to estimate the geometric transformation required to align multiple images.

In image processing, a homography matrix is a 3x3 matrix representing the transformation between two images. This matrix maps the coordinates of a point in one image to the corresponding point in another image. The homography matrix can be computed using a set of corresponding points in the two images.

The homography matrix can be used to perform various image processing tasks, such as image registration, stereo rectification, and panoramic image stitching. For example, in panoramic image stitching, the homography matrix is used to align multiple images with overlapping regions so that they can be stitched together to create a single panoramic image.

Homography can be represented using projective geometry, which allows for the transformation of objects from one plane to another, preserving straight lines and angles. The transformation matrix can be obtained using different algorithms, such as direct linear transformation (DLT) or ***RANSAC*** (Random Sample Consensus).

RANSAC (Random Sample Consensus) is a robust method used in image processing and computer vision to estimate the parameters of a mathematical model from a set of data points that may contain outliers. RANSAC aims to identify the inliers, which are the data points that best fit the mathematical model while discarding the outliers that do not fit the model.

RANSAC is commonly used in image processing to estimate homography between two images to align them. Homography is a transformation that maps points in one image to corresponding points in another image. However, due to factors such as camera movement or image distortion, there may be outliers in the set of corresponding points. RANSAC can be used to estimate the homography matrix in the presence of these outliers.

The RANSAC algorithm works by randomly selecting a subset of the corresponding points and estimating the homography matrix using these points. The homography matrix is then used to transform the remaining points, and those points that fit the model within a specified tolerance are classified as inliers. This process is repeated for a specified number of iterations, and the homography matrix that produces the largest number of inliers is selected as the best estimate.

RANSAC is a powerful algorithm for estimating the parameters of a mathematical model in the presence of outliers. It is widely used in image processing and computer vision for tasks such as image alignment, object recognition, and stereo matching. However, it can be computationally expensive and may require a large number of iterations to obtain a good estimate.

Figure 19 - Left and the right image to be stitched

Figure 20 – Stitched Output Image

5.1.3 Image stitching and advantages

Image stitching combines multiple images with overlapping regions to create a larger panoramic or wide-angle image. The advantages of image stitching include the following:

Increased field of view: Image stitching allows you to create a larger panoramic or wide-angle image, giving you a wider field of view than what is possible with a single image. This can be useful for landscape photography, architectural photography, and other applications where you want to capture a large scene.

High resolution: Image stitching can also increase the resolution of the final image, as each image contributes to the overall resolution of the final image. This can result in a final image that is much higher in resolution than what is possible with a single image.

Reduced distortion: Wide-angle lenses can often produce image distortion, such as barrel distortion or pincushion distortion. Image stitching can help reduce this distortion by stitching together images taken from different perspectives.

Improved dynamic range: Image stitching can also help improve the dynamic range of the final image by combining images with different exposures. This can result in a final image with more detail in the highlights and shadows.

Artistic expression: Image stitching can also be used to create artistic images, such as wide-angle landscapes or cityscapes. It allows photographers to capture impossible scenes with a single image and can add a unique perspective to their work.

Image stitching allows us to create larger, higher-resolution images with reduced distortion and improved dynamic range. It can be helpful in a wide range of applications, from landscape and architectural photography to artistic expression.

5.4 Popular algorithms for image stitching

There are several image stitching algorithms, each with advantages and limitations. Here are some of the most commonly used algorithms for image stitching:

Feature-based algorithms: Feature-based algorithms detect key features in the input images, such as corners, edges, or blobs, and match them across images to determine the relative position and orientation of each image. Popular feature-based algorithms include SIFT (Scale-Invariant Feature Transform) and SURF (Speeded Up Robust Features).

Phase correlation-based algorithms: Phase correlation-based algorithms use Fourier transform to detect and match phase correlations between input images. These algorithms are fast and accurate but require precise alignment of the input images. The most common phase correlation-based algorithm for image stitching is the Lucas-Kanade algorithm.

Optical flow-based algorithms: Optical flow-based algorithms detect motion between consecutive video frames and use it to align images. These algorithms can handle small misalignments and produce smooth image transitions but may struggle with large perspective changes. Standard optical flow-based algorithms include the Lucas-Kanade algorithm and the Farnebäck algorithm.

Graph-cut-based algorithms: Graph-cut-based algorithms partition the input images into overlapping regions and use graph-cut optimization to determine the best alignment of the input images. These algorithms produce accurate and seamless image stitching but can be computationally expensive. Popular graph-cut-based algorithms include the Blended-Pyramid algorithm and the Multi-Band Blending algorithm.

Each image stitching algorithm has its advantages and limitations, and the choice of algorithm depends on the application's specific requirements. Feature-based algorithms are robust and widely used, while phase correlation-based algorithms are fast and accurate. Optical flow-based algorithms can handle small misalignments and produce smooth transitions, and graph-cut-based algorithms produce accurate and seamless image stitching but can be computationally expensive.

5.5 Challenges in Image Stitching

Image stitching is a complex process that involves aligning and blending multiple images to create a single panoramic or wide-angle image. Here are some of the challenges that can arise during the image-stitching process:

Perspective distortion: Perspective distortion can occur when images are taken from different perspectives, resulting in distortions in the final stitched image. This can be particularly challenging when stitching images with complex geometry or architecture.

Exposure and color variations: Exposure and color variations can occur between images due to lighting conditions or camera settings changes. These variations can result in visible seams or differences in color and brightness in the final stitched image.

Parallax error: Parallax error occurs when the camera's viewpoint changes between images, resulting in misalignments between the images. This can result in ghosting or blurring in the final stitched image.

Moving objects: Moving objects in the scene can be challenging when stitching images, as they can cause misalignments or ghosting in the final stitched image.

Limited field of view: The camera's field of view may be limited, which can result in gaps or missing parts of the scene in the final stitched image.

Computational complexity: Image stitching can be computationally complex, particularly when dealing with large numbers of images or high-resolution images. This can result in extended processing times or memory limitations.

Image stitching can be challenging due to issues such as parallax error, perspective distortion, exposure and color variations, moving objects, limited field of view, and computational

complexity. Addressing these challenges requires careful planning, preparation, and optimization of the stitching process, as well as the use of appropriate algorithms and tools.

5.6 Experiment-3: Object detection with Depth measurements

Object detection is a crucial task in computer vision, and artificial intelligence (AI) plays a vital role in achieving accurate and efficient detection. AI-based object detection involves training a deep learning model on a large dataset of labeled images, allowing the model to learn the features that distinguish between different objects.

Passive stereo is similar to that of human eyes. Human brains (subconsciously) estimate the depth of objects and scenes based on the difference between what our left eye sees versus what our right eye sees. On OAK-D cameras, it is the same in a stereo camera pair - left and right monocular cameras - and the VPU does the disparity matching to estimate the depth of objects and scenes.

A stereo pair camera's disparity is the difference (in pixels) between the same point on the left and right images. A great example of disparity can be seen below - a red line is drawn between facial landmarks (eyes, nose, mouth) to demonstrate disparity.

Active stereo depth perception: The OAK Pro cameras use conventional active stereo vision. A dot projector projects many small dots in front of the device, which helps with disparity matching, especially for low-visual-interest surfaces (blank surfaces with little to no texture), such as a wall or ceiling.

The stereo-matching process is performed the same way as passive stereo perception, and the dots only help with the accuracy.

There are several AI-based object detection techniques, including:

Region-based Convolutional Neural Networks (R-CNN): This approach involves identifying regions of an image that could contain an object, then applying a CNN to each region to classify the object.

You Only Look Once (YOLO): YOLO is a real-time object detection system that divides an image into a grid and predicts the object class and location for each cell.

Single Shot Detector (SSD): SSD is another real-time object detection system that involves predicting the object class and location at multiple scales.

AI-based object detection has many applications, including surveillance, autonomous driving, and medical imaging. In surveillance, AI-based object detection can help detect suspicious behavior, such as someone leaving a package in a public space. In autonomous driving, AI-based object detection can help the vehicle identify obstacles and navigate the environment safely. AI-based object detection can help identify tumors or other abnormalities in medical imaging.

Despite the significant advances in AI-based object detection, there are still challenges, such as detecting objects in low light conditions, occlusion, and dealing with variations in object

appearance. However, researchers are constantly developing new techniques and approaches to addressing these challenges.

AI-based object detection is a powerful technology with several applications. As technology advances, we can expect more innovative applications in robotics, augmented reality, and many more.

5.7 Comparing YOLO with SSD

You Only Look Once (YOLO), and Single Shot Detector (SSD) are popular and widely used deep learning-based object detection methods. Both techniques are designed to detect objects in real-time and are optimized for speed and accuracy. Here is a comparison of YOLO and SSD for object detection:

Architecture: YOLO and SSD use a convolutional neural network (CNN) architecture. However, YOLO uses a single neural network to predict the object class and location, whereas SSD uses multiple convolutional layers at different scales to detect objects.

Speed: YOLO is faster than SSD because it only needs to make one pass through the network to predict the object class and location. In contrast, SSD must process multiple image scales to achieve high accuracy.

Accuracy: SSD is generally more accurate than YOLO because it uses multiple scales to detect objects. This means that SSD can detect smaller objects that YOLO may miss. However, YOLO

can be more accurate in detecting large objects because it uses a single neural network to detect them.

Training time: YOLO requires less training time than SSD, as it trains the model end-to-end in a single stage. In contrast, SSD requires multiple stages of training to detect objects at different scales.

Object detection in dense scenes: SSD can perform better than YOLO in dense scenes with overlapping objects because it uses multiple scales to detect objects. YOLO can sometimes miss objects in such scenes.

Object localization: YOLO is better at localizing objects accurately than SSD, as it predicts the object location directly in the image. In contrast, SSD uses anchor boxes to predict object location, which can sometimes result in imprecise object localization.

YOLO and SSD are both efficient object detection methods with their strengths and weaknesses. YOLO is faster and better at localizing large objects, while SSD is generally more accurate and better at detecting smaller objects. The choice between YOLO and SSD depends on the specific requirements of the application and the tradeoff between speed and accuracy.

In our experiment we built the system to compare the performace variance between python implementation on both Jetson-nano and X86 platform as shown in Figure 21. Python implementation are good for proto typing while it has an inherient latencies involved due to

interpreter. Our analysis include comparing both YOLO and SSD for comparision of performance as shown in Figure 21 and Figure 23

C++ implementation would produce production quality binaries and we have observed improvement in performance (as show in Figure 25 and Figure 26) for the same.

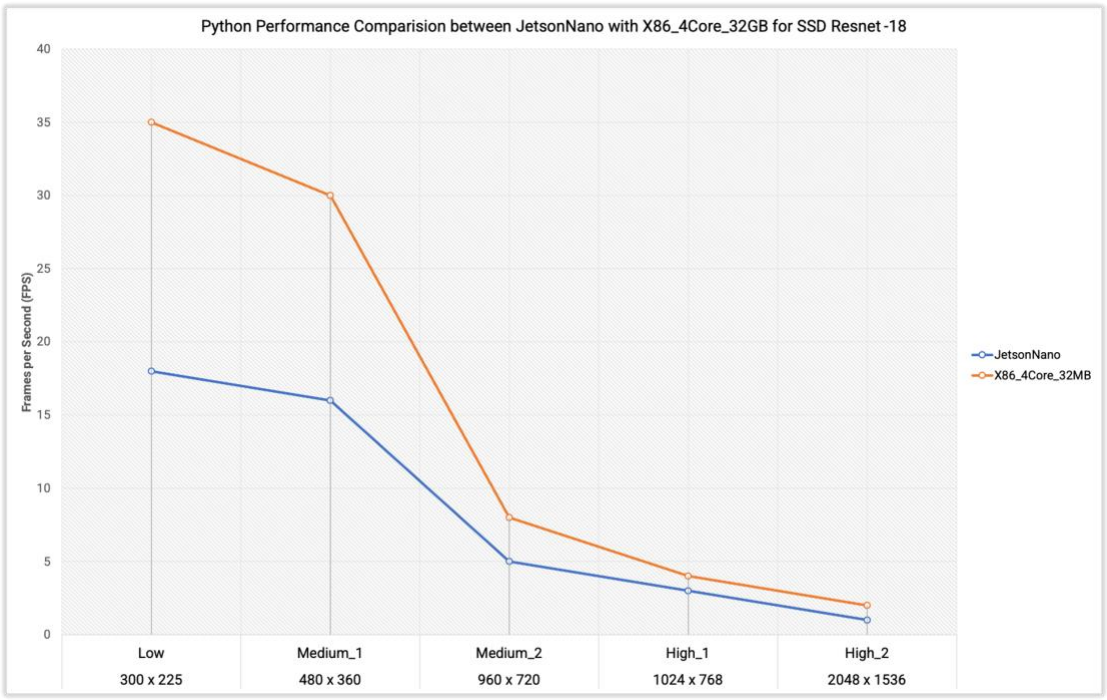

Figure 21 - Comparison of Python implementation between JetsonNano and X86_32GB

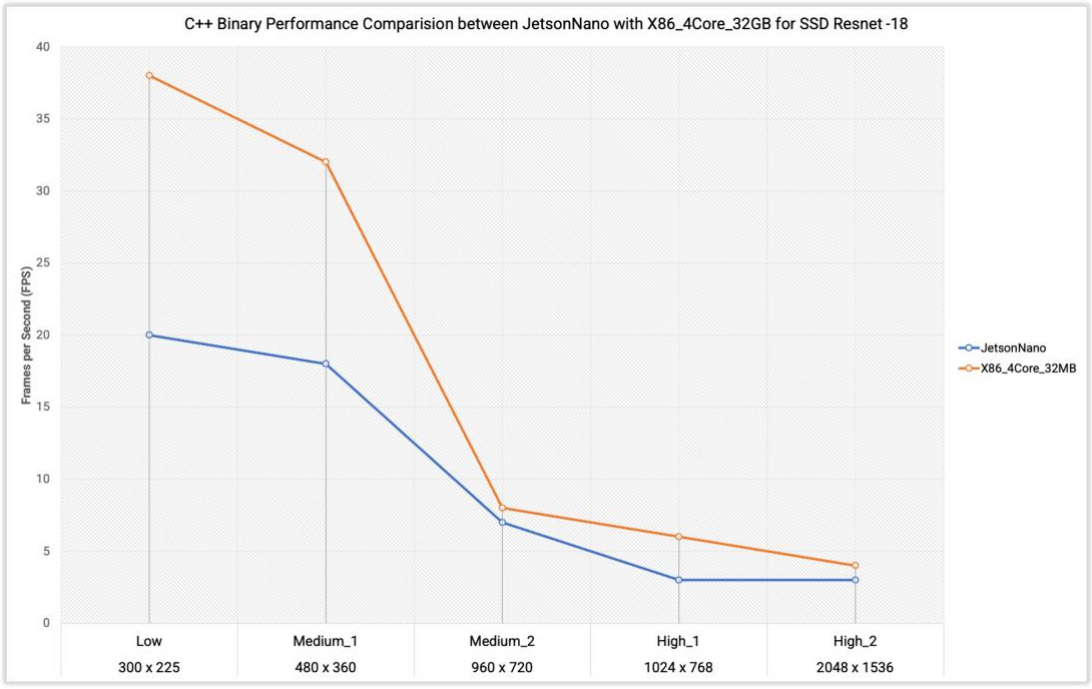

Figure 22 - Comparison of C++ implementation between JetsonNano and X86_32GB

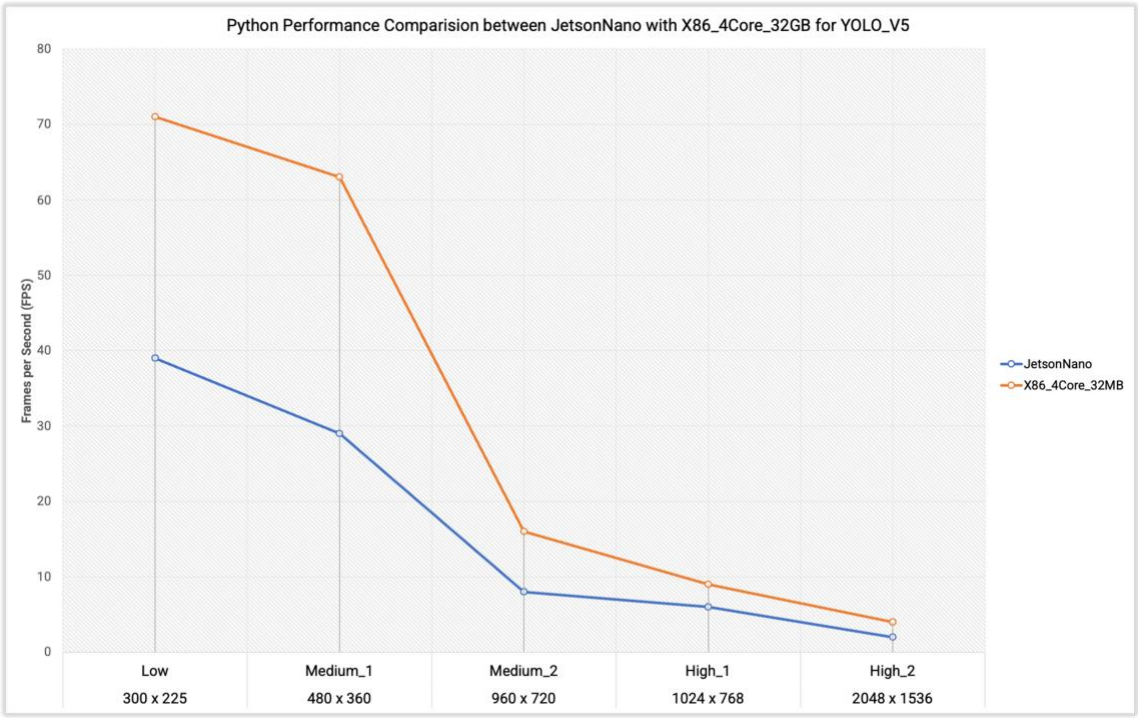

Figure 23 - Comparison of Python implementation between JetsonNano and X86_32GB

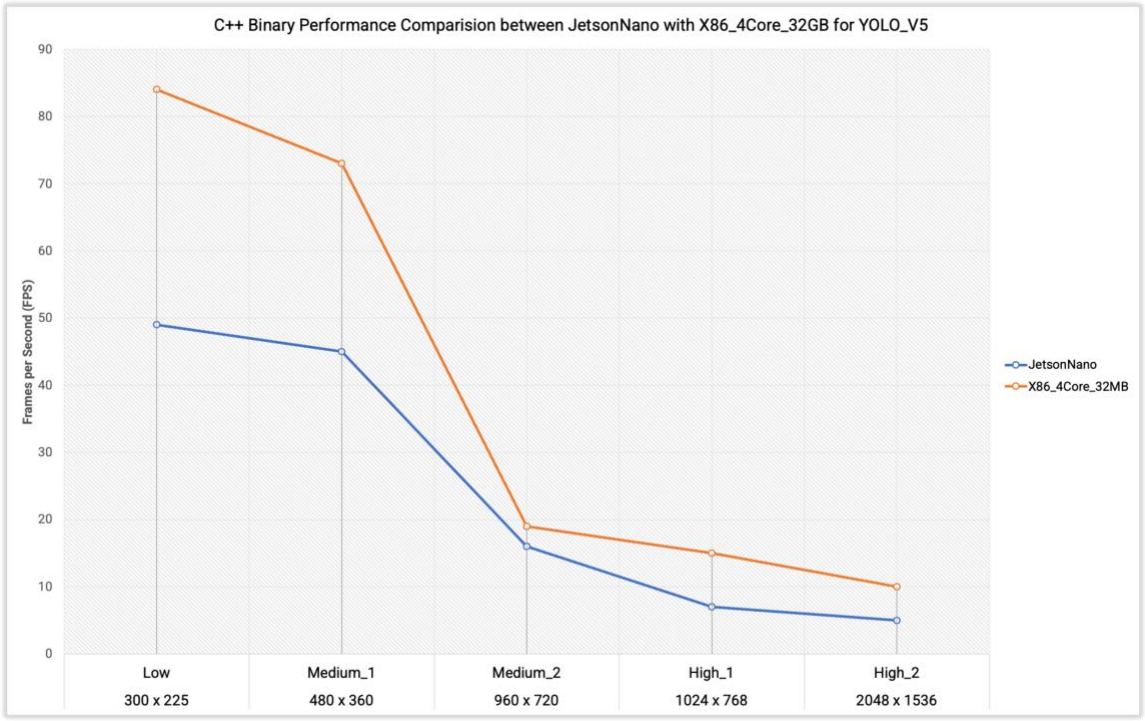

Figure 24 - Comparison of C++ implementation between JetsonNano and X86_32GB

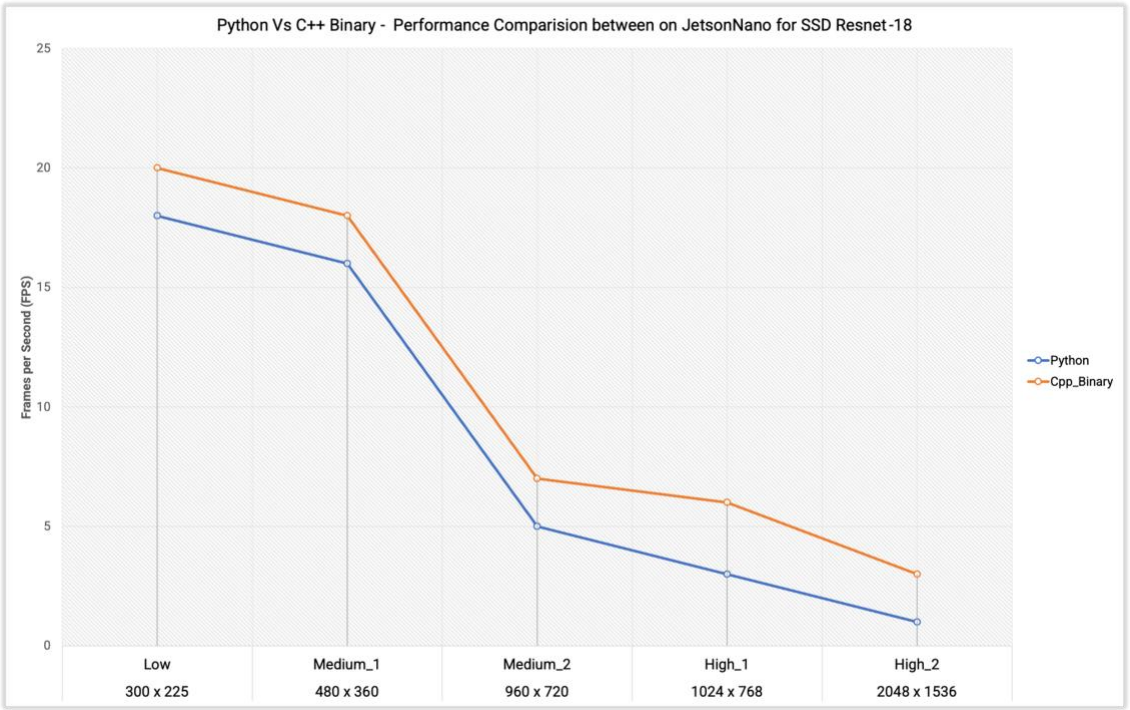

Figure 25 - Comparison of Python Vs. C++ Binary implementation on SSD Resnet-18

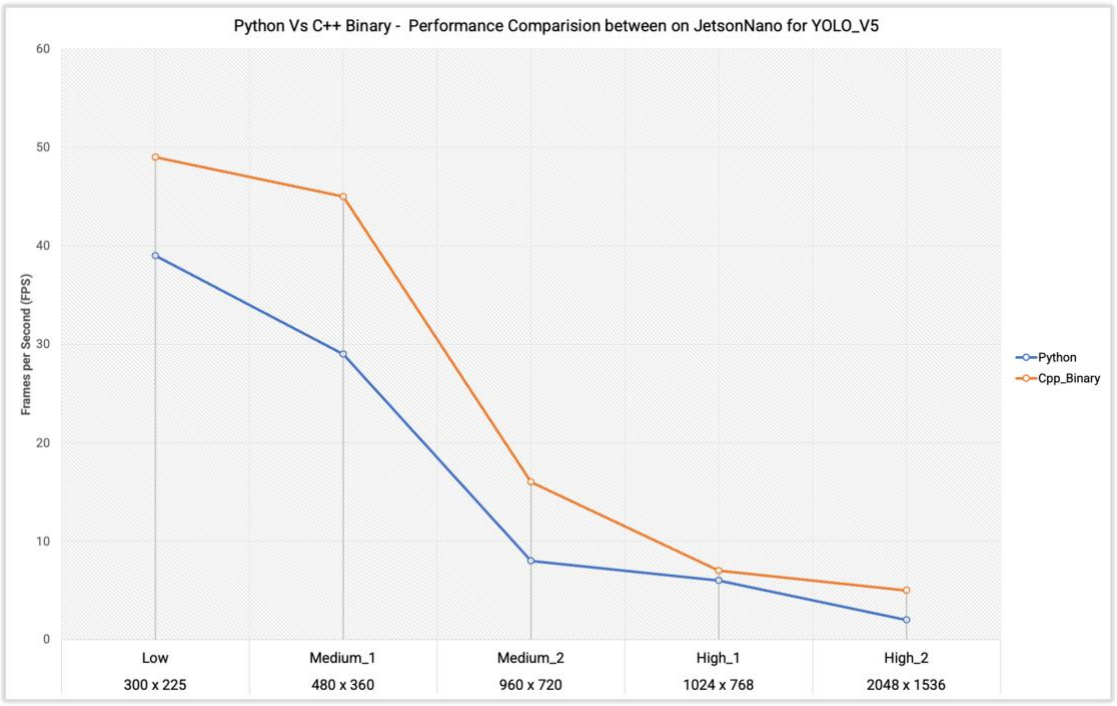

Figure 26 - Comparison of Python Vs. C++ Binary implementation on YOLO 5

Detection of large objects: YOLO is better at detecting more prominent objects, such as cars or people, because it uses a single neural network. This can be an advantage in applications where large objects are of interest.

Object detection in images with low resolution or low contrast: YOLO can perform better than SSD in images with low resolution or low contrast because it uses global information from the entire image to detect objects.

On the other hand, here are some scenarios where SSD may be preferred over YOLO because of the following:

High-precision object detection: SSD is generally more accurate than YOLO because it uses multiple scales to detect objects. This can be an advantage in applications where high-precision object detection is required.

Detection of small objects: SSD can detect smaller objects better than YOLO because it uses multiple scales to detect objects. This can be an advantage in applications where small objects are of interest, such as in medical imaging or microscopy.

Object detection in images with high clutter: SSD can perform better than YOLO in images with high clutter, such as crowded scenes, because it uses anchor boxes to predict object locations. This can result in more precise object detection in such scenarios.

YOLO and SSD have their own strengths and weaknesses, and the choice between them depends on the application's specific requirements. YOLO is faster and better at detecting larger objects, while SSD is generally more accurate and better at detecting smaller objects.

5.8 YOLO vs. SSD for human identifications

Both You Only Look Once (YOLO) and Single Shot Detector (SSD) are capable of detecting and identifying humans in images and videos. However, the choice between YOLO and SSD for human identification depends on the application's specific requirements.

YOLO may be preferred for human identification in scenarios that require real-time processing and detection speed. Since YOLO is faster than SSD, it can process images and videos in real-time and is ideal for applications that require quick identification of humans, such as surveillance systems, security systems, or crowd monitoring.

SSD, on the other hand, may be preferred for applications that require more precise detection and identification of humans. Since SSD is generally more accurate than YOLO, it can detect humans with higher precision, making it ideal for applications that require high-precision human identification, such as medical imaging or forensics.

It is also worth noting that both YOLO and SSD can be trained on large datasets of human images to improve their accuracy and identification capabilities. This means that the accuracy of both methods can be improved by fine-tuning the models on specific human identification tasks.

In summary, both YOLO and SSD can be used for human identification, and the choice between them depends on the application's specific requirements. YOLO may be preferred for real-time processing and detection speed, while SSD may be preferred for higher-precision human identification.

5.9 Challenges in porting YOLO on NVIDIA Jetson Nano

The NVIDIA Jetson Nano is a small, low-power computing platform that runs deep learning models like You Only Look Once (YOLO) for object detection. However, some challenges may arise when porting YOLO to the Jetson Nano. Here are some of the main challenges:

Limited processing power: The Jetson Nano is a low-power platform, which means it has limited processing power compared to more powerful computing platforms. YOLO is a computationally intensive algorithm that requires significant processing power, so it may need to be optimized for the Jetson Nano to run efficiently.

Limited memory: The Jetson Nano has limited memory compared to more powerful computing platforms, which can be challenging when running deep learning models like YOLO. Large models like YOLO may require more memory than is available on the Jetson Nano, so that memory optimization techniques may be required.

Compatibility issues: YOLO is designed to run on a variety of computing platforms, but there may be compatibility issues when porting it to the Jetson Nano. Ensuring that the software libraries and dependencies required by YOLO are compatible with the Jetson Nano is crucial.

Power consumption: The Jetson Nano is a low-power platform, which means it has limited power consumption compared to more powerful computing platforms. YOLO is a computationally intensive algorithm that requires a significant amount of power, so it may need to be optimized for power consumption to run efficiently on the Jetson Nano.

Model optimization: YOLO is a large and complex model requiring significant processing power and memory. To run efficiently on the Jetson Nano, the YOLO model may need to be optimized and compressed, which can be a complex process.

Porting YOLO to the NVIDIA Jetson Nano can be challenging due to its limited processing power, memory, and power consumption. However, with appropriate optimization techniques and compatibility checks, running YOLO efficiently on the Jetson Nano for object detection tasks is possible.

5.10 Experiment-4 : Integrating Multiple POE Depth Cameras

Integrating multiple cameras can be a complex process, and several challenges must be addressed.

Setup: There are three depth based cameras as shown in the Figure 27 . There are two front facing cameras and one rear facing cameras.

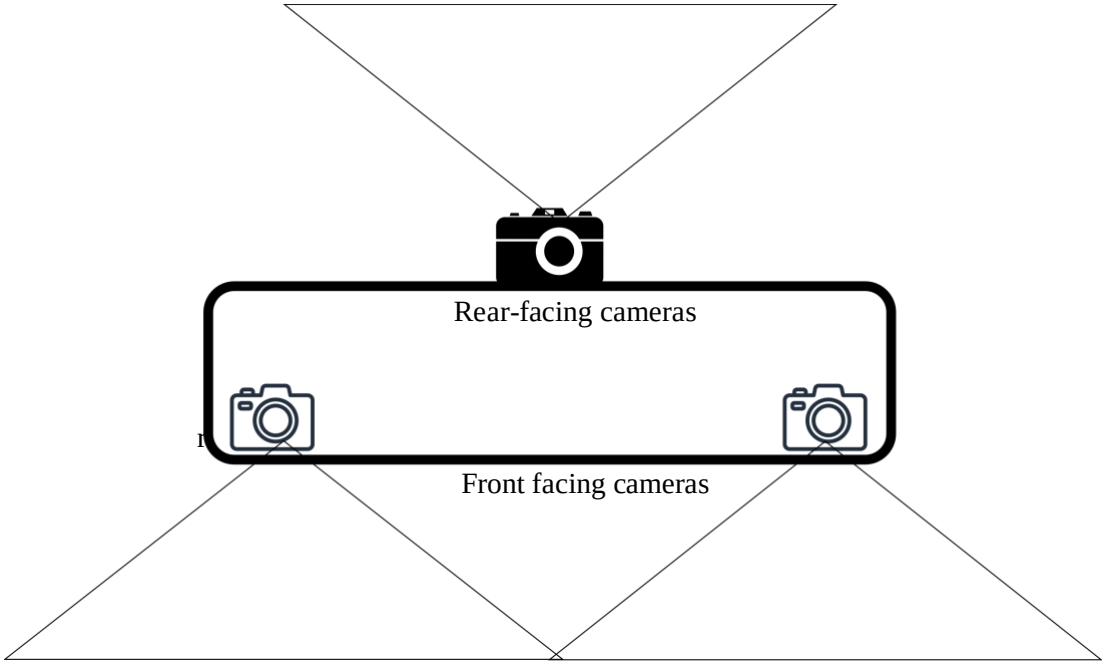

Figure 27 - Three camera setup schematic

Here are some of the main challenges in integrating multiple cameras:

Calibration: Calibration is one of the most critical challenges in integrating multiple cameras. This involves aligning the different cameras' viewpoints so that they can provide accurate and consistent data. It requires careful positioning of the cameras, setting up the right angles, and adjusting the focal length to ensure that the images captured by the different cameras match each other.

Synchronization: Another major challenge is synchronizing the cameras' output. To ensure that the images captured by the different cameras are aligned accurately, they need to be triggered simultaneously. This requires the use of specialized software or hardware to ensure that the cameras are in sync with each other. The data streams must be synchronized to capture the data simultaneously. This can be challenging, as the cameras may have different frame rates or internal clocks.

Data management: Integrating multiple cameras generates a large amount of data, which can be difficult to manage. The images captured by each camera need to be stored, processed, and analyzed, and this requires a robust data management system that can handle large amounts of data. Image processing: Once the data is captured, it must be processed to extract meaningful information. This can be a complex process, particularly if the cameras are capturing images from different angles or with different lighting conditions.

Cost: Integrating multiple cameras can be expensive, particularly if high-quality cameras are required. The cost of hardware, software, and data management systems can add up quickly, making it challenging to justify the expense.

Maintenance: Maintaining multiple cameras can also be a challenge. The cameras need to be regularly serviced and calibrated to ensure that they are functioning correctly. This can be time-consuming and expensive, particularly if the cameras are located in hard-to-reach places.
Camera placement: The placement of the cameras is critical for achieving accurate and reliable data capture. Cameras must be placed in such a way that they capture the desired field of view and do not interfere with each other.

Software integration: Integrating multiple cameras requires camera hardware and software coordination. This can be a challenge, as different cameras may have different drivers or software interfaces that need to be integrated.

Integrating multiple cameras on the NVIDIA Jetson Nano can be further challenging due to limited bandwidth, synchronization, camera placement, data processing, and software integration. Careful planning and optimizing hardware, software, and data streams can help overcome these challenges and ensure accurate and reliable data capture.

Observations: The three depth cameras, two in the front and one in the rear, mounted on a rack, produced promising results.

a. Multiple subjects were detected concurrently as shown in Figure 30 with over 95% accuracy.
b. Multiple subjects were detected on multiple cameras concurrently as shown Figure 31 in with over 95% accuracy.

c. Multiple subjects, multiple cameras with live streaming as shown in Figure 32 with over 95% accuracy

Figure 28 - Three camera setup front view

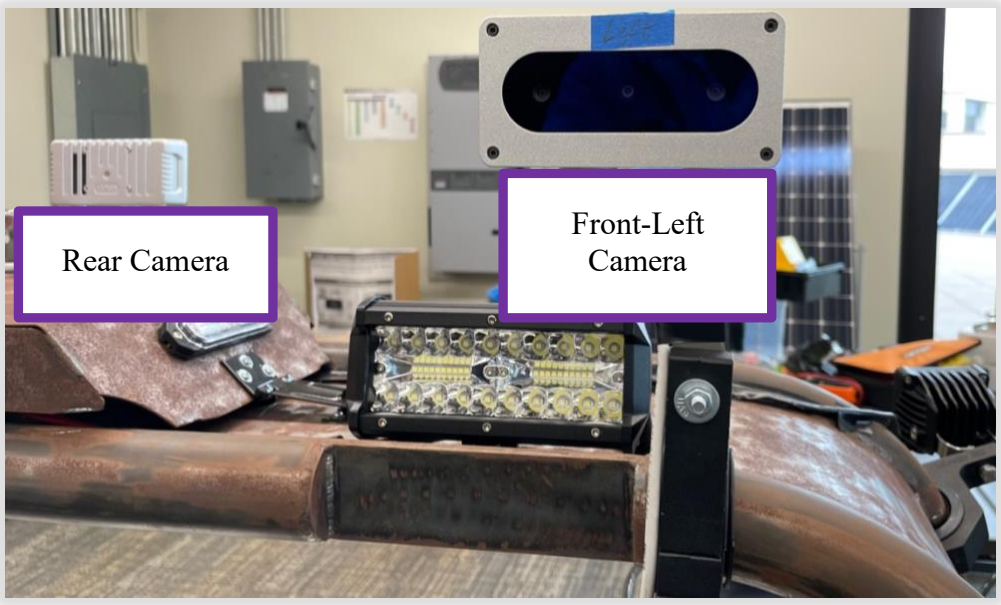

Figure 29 - Camera Mounted

Figure 30 - Multiple subject detection on a single camera with over 95% accuracy

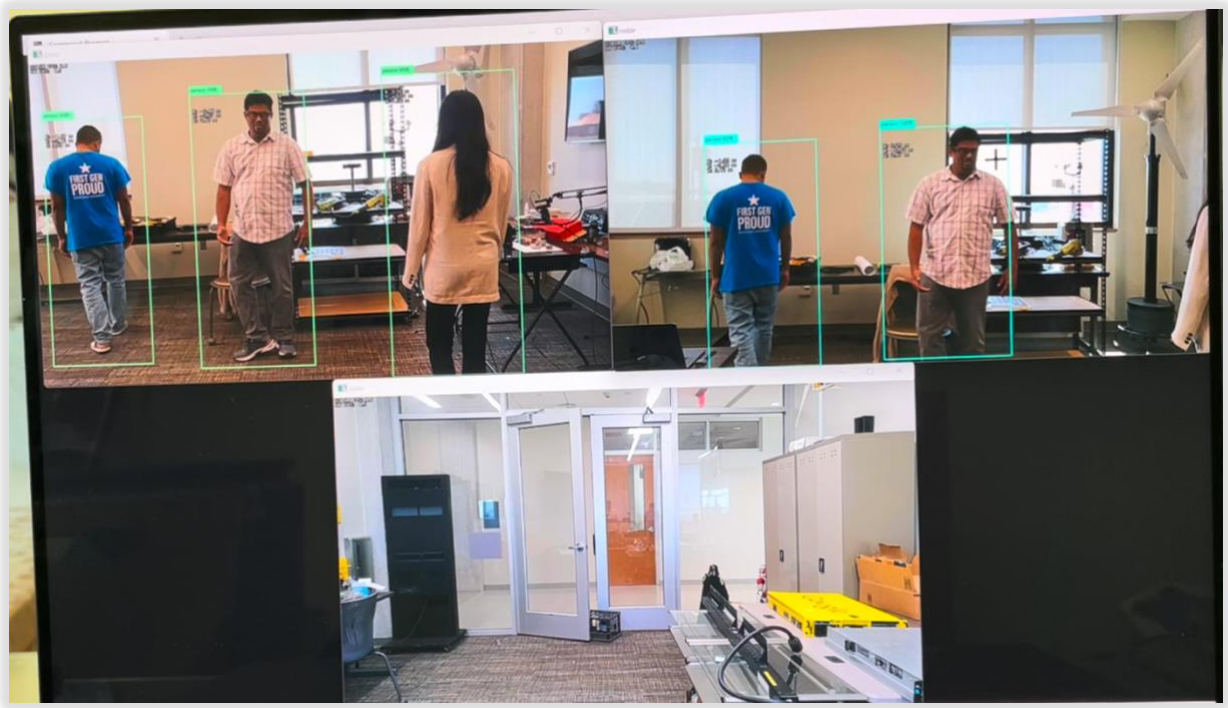

Figure 31 - Concurrent detection on multiple cameras with over 95% accuracy

Figure 32 - Concurrent detection on multiple cameras with live streaming over 95% accuracy

5.11 When to use YOLO as compared to SSD

The choice between You Only Look Once (YOLO) and Single Shot Detector (SSD) for object detection depends on the application's specific requirements. Here are some scenarios where YOLO may be preferred over SSD:

Real-time object detection: YOLO is faster than SSD, making it ideal for applications that require real-time object detection. Examples include surveillance systems, autonomous vehicles, and robotics applications where speed is critical.

Disparity in image depth detection refers to the apparent difference in the position of an object in a scene when viewed from two different perspectives or cameras. This concept is crucial in estimating depth information from stereo images, essential for various computer vision tasks such as 3D reconstruction, autonomous navigation, and obstacle detection. In the context of stereo vision, two cameras are used to capture the same scene from slightly different viewpoints, generating a pair of images known as the left and right stereo images. Since a baseline distance horizontally separates the cameras, objects in the scene will appear at slightly different positions in the two images. This positional difference, or disparity, is inversely proportional to the object's depth from the cameras.

Depth estimation using disparity involves the following steps:

Calibration: To ensure accurate depth estimation, it is essential to calibrate the stereo camera system to account for any misalignments, lens distortions, or differences in intrinsic camera parameters.

Rectification: This process involves aligning the left and right stereo images so that corresponding points lie on the same horizontal scanlines, simplifying the subsequent disparity computation.

Matching: In this step, the algorithm finds corresponding points or features in the left and right images. This can be done using various methods, such as block matching, semi-global matching, or graph-cut algorithms.

Disparity computation: Once corresponding points are identified, the disparity can be calculated as the difference in their horizontal coordinates in the left and right images.

Depth estimation: Using the calculated disparity values, camera geometry, and the baseline distance between the cameras, the depth of each point in the scene can be estimated via triangulation.

Estimating disparity and depth from stereo images can be challenging in cases of occlusions, repetitive textures, or low-textured surfaces. However, Convolutional Neural Networks (CNNs), have significantly improved the accuracy and robustness of disparity estimation and depth detection from stereo images.

Disparity and Error in depth estimation The correlation of depth error with the disparity in image depth detection is an essential factor to consider when evaluating the performance and accuracy of stereo vision algorithms. Depth error refers to the difference between the estimated depth and the true depth of an object or point in the scene. Disparity, as previously explained, is the apparent difference in the position of an object in the scene when viewed from two different perspectives or cameras.

In general, depth error and disparity are inversely correlated, as the depth error typically increases when the disparity decreases. The correlation can be as follows:

a. Small disparity values correspond to objects far from the cameras. In this case, even minor errors in disparity estimation can lead to significant depth errors due to the nonlinear relationship between depth and disparity. As objects get farther from the cameras, the relative change in depth becomes more prominent for a given change in disparity.

b. Contrariwise, large disparity values correspond to objects closer to the cameras. In this case, errors in disparity estimation have a more negligible impact on depth estimation because the relationship between depth and disparity is less sensitive to changes in disparity for nearby objects.

c. Additionally, the depth error can be influenced by the quality of the stereo matching algorithm, camera calibration, and rectification, which affect disparity estimation. Errors in any of these steps can lead to inaccuracies in disparity calculation and, consequently, depth estimation.

d. Noise and other artifacts in the stereo images can also impact the accuracy of disparity estimation. For example, occlusions, repetitive textures, or low-textured surfaces can make it difficult to find corresponding points in the left and right images, resulting in errors in disparity estimation and depth estimation.In summary, depth error and disparity are generally inversely correlated in image depth detection, with depth errors typically increasing as disparity values decrease. This relationship can be affected by various factors, such as the quality of the stereo-matching algorithm, camera calibration as depicted in Figure 34 and Figure 35 , rectification, and noise in the stereo images.

5.12 Hardware Specifications

Camera Hardware Specifications: This OAK camera uses Power-over-Ethernet (PoE) for communication and power. It offers full 802.3af, Class 3 PoE compliance with 1000BASE-T

speeds (1 Gbps), and has a micro SD (uSD) card connector. A PoE injector/switch is required to power the device. It also features IP67 rated enclosure.

Camera module specifications:

Table 4 - OAK Camera Specifications

Camera Specs	Color camera	Stereo pair
Sensor	IMX378 (PY004 AF, PY052 FF)	OV9282 (PY003)
DFOV / HFOV / VFOV	81° / 69° / 55°	82° / 72° / 50°
Resolution	12MP (4032x3040)	1MP (1280x800)
Focus	AF: 8cm - ∞, FF: 50cm - ∞	FF: 19.6cm - ∞
Max Framerate	60 FPS	120 FPS
F-number	1.8 ±5%	2.0 ±5%
Lens size	1/2.3 inch	1/4 inch
Effective Focal Length	4.81mm	2.35mm
Distortion	< 1% AF, < 1.5% FF	< 1%
Pixel size	1.55µm x 1.55µm	3µm x 3µm

RVC2 Performance

- Robotics Vision Core 2 (RVC2 in short) is the second generation of our RVC. OAK Series 2 and our initial devices are built on top of the RVC2.
- RVC2 encapsulates two main components:
- DepthAI features that are fine-tuned for the particular SoC

- A performant SoC and all it's support circuitry (HS PCB layout, power delivery network, efficient heat dissipation, etc.)
- This OAK device is built on top of the RVC2. Main features:
- 4 TOPS of processing power (1.4 TOPS for AI - RVC2 NN Performance)
- Run any AI model, even custom architectured/built ones - models need to be converted.
- Encoding: H.264, H.265, MJPEG - 4K/30FPS, 1080P/60FPS
- Computer vision: warp/dewarp, resize, crop via ImageManip node, edge detection, feature tracking. You can also run custom CV functions
- Stereo depth perception with filtering, post-processing, RGB-depth alignment, and high configurability
- Object tracking: 2D and 3D tracking with ObjectTracker node

Streaming Hybrid Architecture Vector Engines – SHAVES: The SHAVES are vector processors in DepthAI/OAK. The 2x NCE (neural compute engines) were architected for a slew of operations, but there are some that are not implemented. So the SHAVES take over these operations.

These SHAVES are also used for other things in the device, like handling reformatting of images, doing some ISP, etc.

So the higher the resolution, the more SHAVES are consumed for this.

- For 1080p, 13 SHAVES (of 16) are free for neural network stuff.
- For 4K sensor resolution, 10 SHAVES are available for neural operations.

- There is an internal resource manager inside DepthAI firmware that coordinates the use of SHAVES, and warns if too many resources are requested by a given pipeline configuration.

StereoDepth: Stereo vision is a technique used to create a perception of depth by using two cameras, placed side by side, to capture two images of a scene from slightly different angles, mimicking how our eyes perceive the world. A computer then processes the images to calculate the differences in the position of objects in the two images, which allows the system to determine their depth.

The process of stereo vision involves several steps:

1. ***Image capture***: Two cameras are used to capture two images of a scene simultaneously.
2. ***Image rectification***: By transforming the images, they appear to have been captured from the same perspective, simplifying subsequent processing.
3. ***Correspondence matching***: The computer algorithm identifies corresponding points in the two images with the same position in 3D space. This is typically done by comparing the brightness and texture of each pixel in the two images.
4. ***Depth estimation***: Using the information from the correspondence matching, the distance between each corresponding point can be calculated, allowing the system to estimate the depth of each point in the scene.
5. ***3D reconstruction***: By combining the depth information from all the points in the scene, a 3D model of the scene can be reconstructed.

Depth detection using a single camera is also possible. However, it requires additional information about the scene, such as the size and position of objects or the amount of light reflected from them. One standard method for depth detection is to use structured light, where a pattern of light is projected onto the scene, and the deformation of the pattern is used to calculate the depth of objects. Another method is time-of-flight, where the time taken for a light pulse to travel from the camera to the scene and back is used to calculate the distance. Stereo vision and depth detection are important techniques for creating a more immersive and interactive visual experience. They have many practical applications in robotics, automation, and computer vision.

Disparity refers to the distance between two corresponding points in a stereo pair's left and right image. StereoDepth node calculates the disparity/depth from the stereo camera pair (2x MonoCamera/ColorCamera).

Stereo depth FPS:

Table 5 - Stereo depth FPS Comparision

Stereo depth mode	FPS for 1280x720	FPS for 640x400
Standard mode	60	110
Left-Right Check	55	105
Subpixel Disparity	45	105
Extended Disparity	54	105
Subpixel + LR check	34	96

Extended + LR check	26	62

On the block diagram Figure 33 , red rectangle are firmware settings that are configurable via the API. Gray rectangles are settings that that are not yet exposed to the API. We plan on exposing as much configurability as possible, but please inform us if you would like to see these settings configurable sooner.

If you click on the image, you will be redirected to the webapp. Some blocks have notes that provide additional technical information.

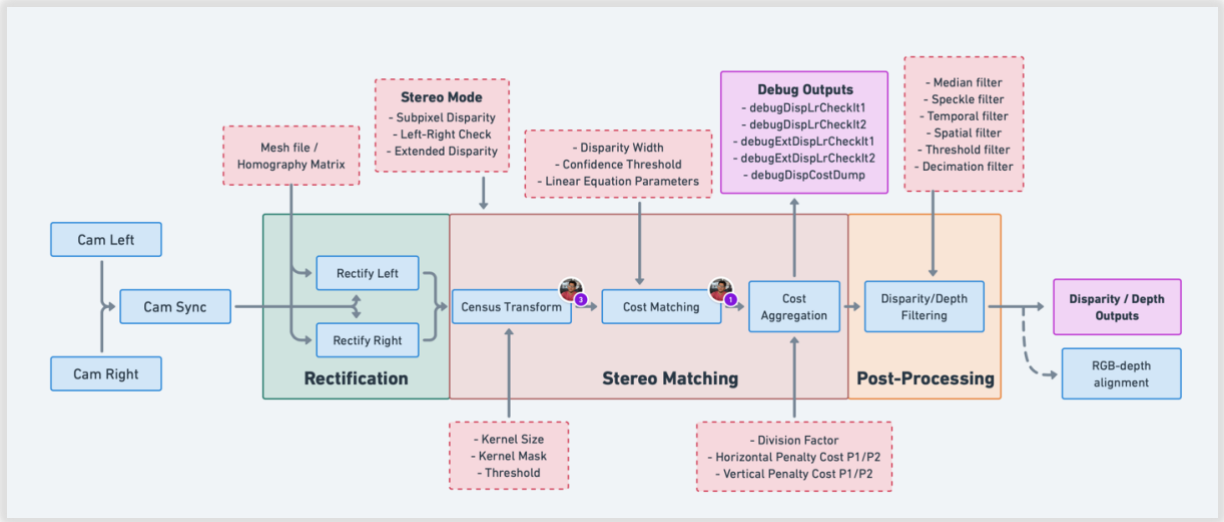

Figure 33 - Blockdiagram of Stereo depth node (ref: docs.luxonis.com)

Subpixel mode improves the precision and is especially useful for long range measurements. It also helps for better estimating surface normals.

Besides the integer disparity output, the Stereo engine is programmed to dump to memory the cost volume, that is 96 levels (disparities) per pixel, then software interpolation is done on Shave,

resulting a final disparity with 3 fractional bits, resulting in significantly more granular depth steps (8 additional steps between the integer-pixel depth steps), and also theoretically, longer-distance depth viewing - as the maximum depth is no longer limited by a feature being a full integer pixel-step apart, but rather 1/8 of a pixel.

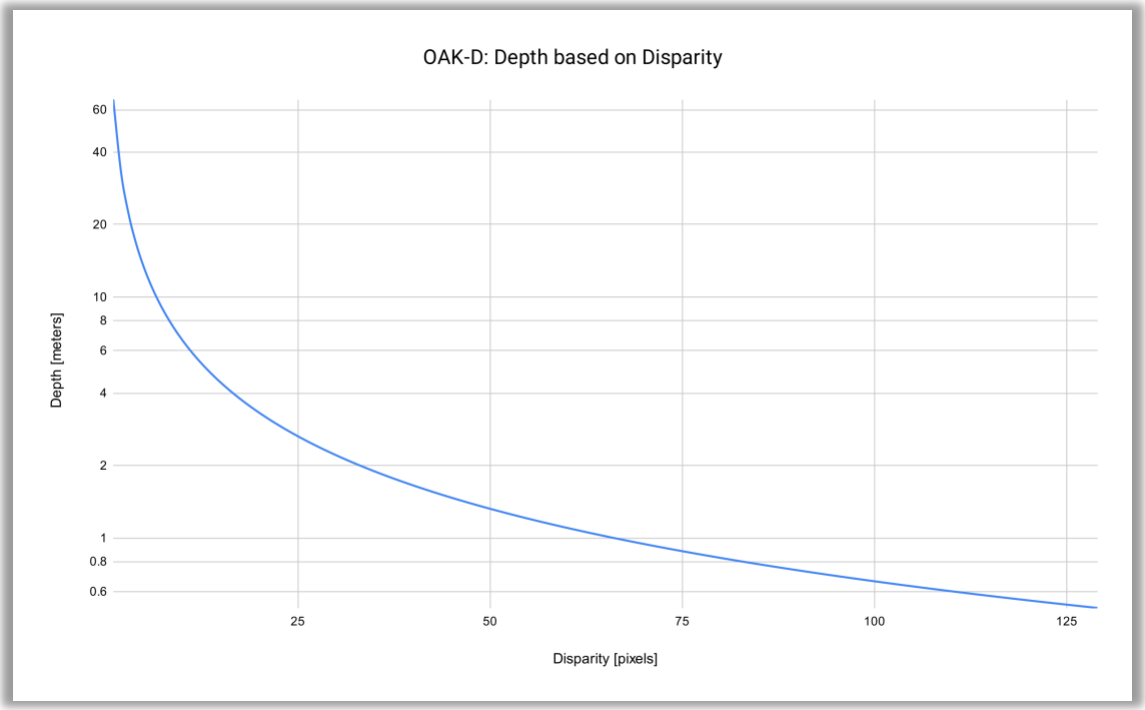

Figure 34 - Variation of depth vs. disparity

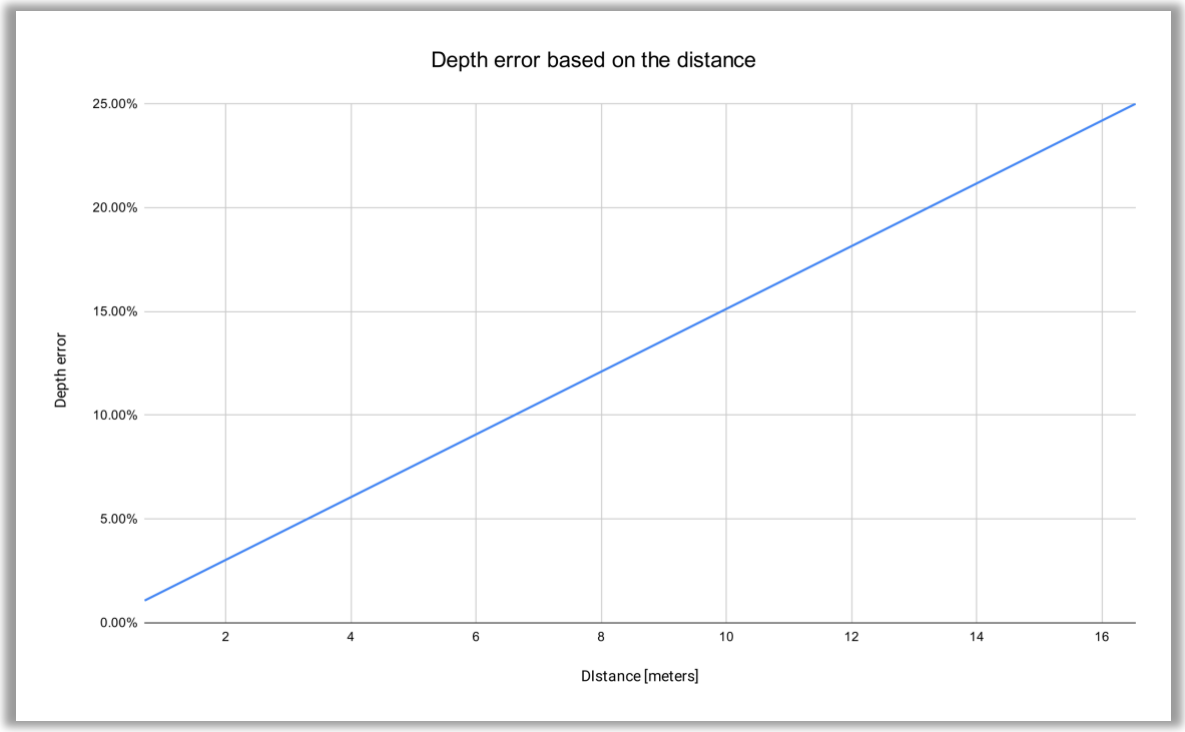

Figure 35 - Variation of depth error vs. distance

6. CONCLUSION

Using Deep Neural Network-Based Face Detection based on an IEEE paper from Henry A. Rowley, Shumeet Baluja, and Takeo Kanade[49] and implementing its effort of fast real-time low latency high-speed face recognition system. This research focuses on process optimization for quick response time (less than a second) and the software with a low footprint and maintaining an accuracy of up to 99%. Problems with spatial dependencies between variables arise in multiple domains of computer vision, such as semantic image labeling, single image depth prediction, and object detection. Future work will extend our architecture to these problems.

The optimized neural network resulted in almost 99% accuracy in detecting objects, including faces. Actress Angelina Jolie was enrolled along with other celebrities. To take part in the game, users were called upon to choose any image of Angela Jolie from the Internet.

Truly AI is the panacea; in this research, the model learned from the incoming data is dynamic and accurate. Furthermore, pre-processing is vital in the inference response time and accuracy. VarGFaceNet is competent in gaining a better tradeoff between performance and efficiency. The hyperparameters are specific to face recognition that helps retain critical information while reducing parameters. Furthermore, to improve the interpretation VarGFaceNet applies a parity of angular distillation loss as an objective function and presents a recursive knowledge distillation strategy. Tests on random images from the wild led to an accuracy of 99% to ground reality, which is beyond human perception.

7. FUTURE WORK

Object recognition is a critical technology in industrial 4.0, where automation and intelligent machines are revolutionizing manufacturing and production. Object recognition allows machines to identify and interact with objects in their environment, making it possible to automate tasks that were previously done manually.

Here are some ways object recognition is being used in industrial 4.0:

Quality control: Object recognition is used to improve manufacturing processes' quality control. For example, cameras and machine vision systems can be used to inspect products and identify defects such as cracks, scratches, or incorrect labeling.

Pick and place: Object recognition can be used to automate pick and place tasks, where robots are used to move objects from one location to another. Cameras and machine vision systems can be used to identify and locate objects, making it possible for robots to pick them up and move them without human intervention.

Assembly: Object recognition can be used to automate assembly tasks, where robots are used to put together products. Cameras and machine vision systems can be used to identify and locate parts, making it possible for robots to assemble products accurately and quickly.
Inventory management: Object recognition can be used to improve inventory management in warehouses and factories. Cameras and machine vision systems can be used to identify and track

products as they move through the supply chain, making it possible to manage inventory more efficiently.

Safety: Object recognition can be used to improve safety in manufacturing environments. Cameras and machine vision systems can be used to identify people and objects in hazardous areas, making it possible to prevent accidents and reduce the risk of injury.

Overall, object recognition is a critical technology in industrial 4.0, enabling machines to interact with the physical world and automate tasks that were previously done manually. By improving quality control, reducing errors, and increasing efficiency, object recognition is helping to drive the transformation of manufacturing and production.

Digital Pathology: The field of digital pathology utilizes whole-slide scanners to digitize glass slides at high resolution in a relatively new and rapidly expanding field of medical imaging. The availability of whole-slide images (WSI) has garnered the interest of the medical image analysis community, resulting in increasing numbers of publications on histopathologic image analysis. WSI is in a multi-resolution pyramid structure. The aim is to evaluate new and existing algorithms for the automated detection and classification of cancer metastases in whole-slide images of histological lymph node sections. In Digital Pathology, the size does matter. Each image is in a few in the order of gigabytes. Lossy compression is unacceptable while hatching and patching techniques would induce noise and latency.

Other Applications include but are not limited to:

- Pose and posture detection and estimation
- Object Detection and Tracking

- Edge detection
- License Plates and several vehicle attribute recognition
- Posture detection
- Sign language recognition
- Optical Character Recognition (OCR)
- We can also customize to detect any object by appropriate training

REFERENCES

[1] “Le et al. - 2012 - Interactive Facial Feature Localization.pdf.” Accessed: Jan. 03, 2020. [Online]. Available: http://www.ifp.illinois.edu/~vuongle2/helen/eccv2012_helen_final.pdf

[2] “Active Appearance Models (AAM) T.F. Cootes, G.J. Edwards and C.J. Taylor.” Accessed: Oct. 16, 2019. [Online]. Available: https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/cootes-eccv-98.pdf

[3] M. Pietikäinen, “Local Binary Patterns,” *Scholarpedia*, vol. 5, no. 3, p. 9775, Mar. 2010, doi: 10.4249/scholarpedia.9775.

[4] “Schroff et al. - 2015 - FaceNet A unified embedding for face recognition .pdf.” Accessed: Dec. 01, 2019. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2015/app/1A_089.pdf

[5] Y. E. Wang, G.-Y. Wei, and D. Brooks, “Benchmarking TPU, GPU, and CPU Platforms for Deep Learning,” *arXiv:1907.10701 [cs, stat]*, Oct. 2019, Accessed: Jun. 11, 2020. [Online]. Available: http://arxiv.org/abs/1907.10701

[6] P. Mattson *et al.*, “MLPerf Training Benchmark,” *arXiv:1910.01500 [cs, stat]*, Mar. 2020, Accessed: Jun. 11, 2020. [Online]. Available: http://arxiv.org/abs/1910.01500

[7] R. Adolf, S. Rama, B. Reagen, G.-Y. Wei, and D. Brooks, “Fathom: Reference Workloads for Modern Deep Learning Methods,” *2016 IEEE International Symposium on Workload Characterization (IISWC)*, pp. 1–10, Sep. 2016, doi: 10.1109/IISWC.2016.7581275.

[8] T. Chen *et al.*, “MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems,” *arXiv:1512.01274 [cs]*, Dec. 2015, Accessed: Jun. 16, 2020. [Online]. Available: http://arxiv.org/abs/1512.01274

[9] A. Ganapathi, K. Datta, A. Fox, and D. Patterson, “A Case for Machine Learning to Optimize Multicore Performance,” p. 6.

[10] J. Cavazos, G. Fursin, F. Agakov, E. Bonilla, M. F. P. O’Boyle, and O. Temam, “Rapidly Selecting Good Compiler Optimizations using Performance Counters,” in *International Symposium on Code Generation and Optimization (CGO’07)*, Mar. 2007, pp. 185–197. doi: 10.1109/CGO.2007.32.

[11] R. S. Andersen, C. Schou, J. S. Damgaard, and O. Madsen, “Using a Flexible Skill-Based Approach to Recognize Objects in Industrial Scenarios,” in *Proceedings of ISR 2016: 47st International Symposium on Robotics*, Jun. 2016, pp. 1–8.

[12] K. Kim, J. Cho, J. Pyo, S. Kang, and J. Kim, “Dynamic Object Recognition Using Precise Location Detection and ANN for Robot Manipulator,” in *2017 International Conference on Control, Artificial Intelligence, Robotics Optimization (ICCAIRO)*, May 2017, pp. 237–241. doi: 10.1109/ICCAIRO.2017.52.

[13] M. Abou-El-Ela and F. El-Amroussy, “A machine vision system for the recognition and positioning of two-dimensional partially occluded objects,” in *Proceedings of 8th Mediterranean Electrotechnical Conference on Industrial Applications in Power Systems, Computer Science and Telecommunications (MELECON 96)*, May 1996, vol. 2, pp. 1087–1092 vol.2. doi: 10.1109/MELCON.1996.551397.

[14] H. Zhang, X. Lan, X. Zhou, Z. Tian, Y. Zhang, and N. Zheng, “Visual Manipulation Relationship Network for Autonomous Robotics,” in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, Nov. 2018, pp. 118–125. doi: 10.1109/HUMANOIDS.2018.8625071.

[15] J. de la Casa Cárdenas, A. S. García, S. S. Martínez, J. G. García, and J. G. Ortega, "Model predictive position/force control of an anthropomorphic robotic arm," in *2015 IEEE International Conference on Industrial Technology (ICIT)*, Mar. 2015, pp. 326–331. doi: 10.1109/ICIT.2015.7125119.

[16] S. Ray, S. Das, and A. Sen, "An intelligent vision system for monitoring security and surveillance of ATM," in *2015 Annual IEEE India Conference (INDICON)*, Dec. 2015, pp. 1–5. doi: 10.1109/INDICON.2015.7443827.

[17] "Vogel-Heuser and Hess - 2016 - Guest Editorial Industry 4.0–Prerequisites and Vis.html." Accessed: Dec. 29, 2019. [Online]. Available: http://ieeexplore.ieee.org/document/7410115/?arnumber=7410115

[18] R. Hebbalaguppe, G. Garg, E. Hassan, H. Ghosh, and A. Verma, "Telecom Inventory Management via Object Recognition and Localisation on Google Street View Images," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Mar. 2017, pp. 725–733. doi: 10.1109/WACV.2017.86.

[19] Zhixiong Liu and Guiming He, "A vehicle anti-theft and alarm system based on computer vision," in *IEEE International Conference on Vehicular Electronics and Safety, 2005.*, Oct. 2005, pp. 326–330. doi: 10.1109/ICVES.2005.1563666.

[20] A. Stylianou, J. Schreier, R. Souvenir, and R. Pless, "TraffickCam: Crowdsourced and Computer Vision Based Approaches to Fighting Sex Trafficking," in *2017 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, Oct. 2017, pp. 1–8. doi: 10.1109/AIPR.2017.8457947.

[21] D. Dong, X. Li, and X. Sun, “A Vision-Based Method for Improving the Safety of Self-Driving,” in *2018 12th International Conference on Reliability, Maintainability, and Safety (ICRMS)*, Oct. 2018, pp. 167–171. doi: 10.1109/ICRMS.2018.00040.

[22] F. Cardile, G. Iannizzotto, and F. L. Rosa, “A vision-based system for elderly patients monitoring,” in *3rd International Conference on Human System Interaction*, May 2010, pp. 195–202. doi: 10.1109/HSI.2010.5514566.

[23] M. Inoue, R. Taguchi, and T. Umezaki, “Vision-based Bed Detection for Hospital Patient Monitoring System,” in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Jul. 2018, pp. 5006–5009. doi: 10.1109/EMBC.2018.8513460.

[24] G. Urban *et al.*, “Deep Learning for Drug Discovery and Cancer Research: Automated Analysis of Vascularization Images,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 16, no. 3, pp. 1029–1035, May 2019, doi: 10.1109/TCBB.2018.2841396.

[25] J. Thevenot, M. B. López, and A. Hadid, “A Survey on Computer Vision for Assistive Medical Diagnosis From Faces,” *IEEE Journal of Biomedical and Health Informatics*, vol. 22, no. 5, pp. 1497–1511, Sep. 2018, doi: 10.1109/JBHI.2017.2754861.

[26] A. G. P. H., J. Anitha, and J. N. R. J., “Computer Aided Diagnosis Methods for Classification of Diabetic Retinopathy Using Fundus Images,” in *2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET)*, Dec. 2018, pp. 1–4. doi: 10.1109/ICCSDET.2018.8821200.

[27] D. J. Kale, R. Tashakkori, and R. M. Parry, “Automated beehive surveillance using computer vision,” in *SoutheastCon 2015*, Apr. 2015, pp. 1–3. doi: 10.1109/SECON.2015.7132991.

[28] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, “Hand Keypoint Detection in Single Images using Multiview Bootstrapping,” *arXiv:1704.07809 [cs]*, Apr. 2017, Accessed: Dec. 15, 2019. [Online]. Available: http://arxiv.org/abs/1704.07809

[29] G. B. Huang and E. Learned-Miller, “Labeled Faces in the Wild: Updates and New Reporting Procedures,” p. 5.

[30] “CAMELYON17 - Digital pathology.” https://camelyon17.grand-challenge.org/Background/ (accessed Dec. 07, 2019).

[31] “BRAINMAPS.ORG - BRAIN ATLAS, BRAIN MAPS, BRAIN STRUCTURE, NEUROINFORMATICS, BRAIN, STEREOTAXIC ATLAS, NEUROSCIENCE.” http://brain-maps.org/index.php?p=how-big-is-brainmaps (accessed Dec. 07, 2019).

[32] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, Jun. 2005, vol. 1, pp. 886–893 vol. 1. doi: 10.1109/CVPR.2005.177.

[33] E. Hoffer and N. Ailon, “Deep metric learning using Triplet network,” *arXiv:1412.6622 [cs, stat]*, Dec. 2018, Accessed: Dec. 04, 2019. [Online]. Available: http://arxiv.org/abs/1412.6622

[34] A. Suleiman, Y.-H. Chen, J. Emer, and V. Sze, “Towards Closing the Energy Gap Between HOG and CNN Features for Embedded Vision,” *arXiv:1703.05853 [cs]*, Mar. 2017, Accessed: Dec. 04, 2019. [Online]. Available: http://arxiv.org/abs/1703.05853

[35] M. Agarwal, H. Agrawal, N. Jain, and M. Kumar, “Face Recognition Using Principle Component Analysis, Eigenface and Neural Network,” in *2010 International Conference on Signal Acquisition and Processing*, Feb. 2010, pp. 310–314. doi: 10.1109/ICSAP.2010.51.

[36] S. V. Chakrasali and S. Kuthale, “Optimized face detection on FPGA,” in *2016 International Conference on Circuits, Controls, Communications and Computing (I4C)*, Oct. 2016, pp. 1–6. doi: 10.1109/CIMCA.2016.8053269.

[37] M. Yan, M. Zhao, Z. Xu, Q. Zhang, G. Wang, and Z. Su, “VarGFaceNet: An Efficient Variable Group Convolutional Neural Network for Lightweight Face Recognition,” *arXiv:1910.04985 [cs]*, Nov. 2019, Accessed: Dec. 13, 2019. [Online]. Available: http://arxiv.org/abs/1910.04985

[38] B. Amos, B. Ludwiczuk, and M. Satyanarayanan, “OpenFace: A general-purpose face recognition library with mobile applications,” p. 20.

[39] M. Wang and W. Deng, “Deep Face Recognition: A Survey,” *arXiv:1804.06655 [cs]*, Feb. 2019, Accessed: Dec. 01, 2019. [Online]. Available: http://arxiv.org/abs/1804.06655

[40] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “DeepFace: Closing the Gap to Human-Level Performance in Face Verification,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, Jun. 2014, pp. 1701–1708. doi: 10.1109/CVPR.2014.220.

[41] Y. Sun, X. Wang, and X. Tang, “Deep Learning Face Representation from Predicting 10,000 Classes,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, Jun. 2014, pp. 1891–1898. doi: 10.1109/CVPR.2014.244.

[42] W. Liu, Y. Wen, Z. Yu, and M. Yang, “Large-Margin Softmax Loss for Convolutional Neural Networks,” *arXiv:1612.02295 [cs, stat]*, Nov. 2017, Accessed: Dec. 29, 2019. [Online]. Available: http://arxiv.org/abs/1612.02295

[43] L. Lin, G. Wang, W. Zuo, X. Feng, and L. Zhang, “Cross-Domain Visual Matching via Generalized Similarity Measure and Feature Learning,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1089–1102, Jun. 2017, doi: 10.1109/TPAMI.2016.2567386.

[44] “Barnouti - 2016 - Improve Face Recognition Rate Using Different Imag.pdf.” Accessed: Dec. 07, 2019. [Online]. Available: http://www.ajer.org/papers/v5(04)/E0504046053.pdf

[45] I. Andrezza, E. Borges, I. Almeida, R. Mota, R. Marques, and L. Batista, “Normalization Methods Analysis Applied to Face Recognition,” in *2017 Workshop of Computer Vision (WVC)*, Oct. 2017, pp. 108–113. doi: 10.1109/WVC.2017.00026.

[46] T. Hassner, S. Harel, E. Paz, and R. Enbar, “Effective Face Frontalization in Unconstrained Images,” *arXiv:1411.7964 [cs]*, Nov. 2014, Accessed: Dec. 06, 2019. [Online]. Available: http://arxiv.org/abs/1411.7964

[47] X. Yin, X. Yu, K. Sohn, X. Liu, and M. Chandraker, “Towards Large-Pose Face Frontalization in the Wild,” *arXiv:1704.06244 [cs]*, Aug. 2017, Accessed: Dec. 06, 2019. [Online]. Available: http://arxiv.org/abs/1704.06244

[48] X. Li, Y. Xu, Q. Lv, and Y. Dou, “Affine-Transformation Parameters Regression for Face Alignment,” *IEEE Signal Processing Letters*, vol. 23, no. 1, pp. 55–59, Jan. 2016, doi: 10.1109/LSP.2015.2499778.

[49] H. A. Rowley, S. Baluja, and T. Kanade, “Neural network-based face detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 23–38, Jan. 1998, doi: 10.1109/34.655647.

[50] “LFW : Results.” http://vis-www.cs.umass.edu/lfw/results.html (accessed Dec. 07, 2019).

[51] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981, doi: 10.1145/358669.358692.

[52] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004, doi: 10.1023/B:VISI.0000029664.99615.94.