

Real Time Multiple Face Recognition using Intra-Layer, Inter-Layer and Hybrid Parallelism

Vijaykumar P. Mantri^{1, 2}, Sandip Thite¹

¹Vishwakarma University, Pune India

²MIT Academy of Engineering, Pune India

*vijay.mantri2000@gmail.com, sandip.thite@vupune.ac.in

ARTICLE INFO

Received: 07 Dec 2024

Revised: 30 Jan 2025

Accepted: 07 Feb 2025

ABSTRACT

When it comes to biometric applications with image analysis and computer vision, face recognition ranks high and is a very important process. Face Recognition is a technology that identifies or verifies individuals based on facial features using computer vision and machine learning. In different personal and commercial fields many uses of face recognition applications which required faster face recognition in real time. Recognizing multiple faces in real time on high-resolution CCTVs requires a lot of processing power. Rather than using serial process if we use intra-layer and inter-layer (or both hybrid) parallel process, the desired real-time performance can be attained. This research paper presents a multiple face recognition system for large-scale data sets using intra-layer and inter-layer parallelism with their analysis and comparison on the embedded GPU system. Also, we propose combining parallelism strategy with an intra-layer and inter-layer parallelism known as hybrid parallelism as a research outcome of our research work for optimal performance. Our hybrid parallelism experimental results showed that the proposed system can recognize multiple faces time in real time with 20% reduction in training time compared to intra-layer and improved inference time by 43% compared to inter-layer. Also, accuracy increased marginally by 0.4% compared to intra-layer and inter-layer parallelism due to reduced bottlenecks in computation

Keywords: : Face Recognition, Multiple Face Detection, Deep Learning, Parallel Processing, Inter-layer Parallelism, Intra-layer Parallelism, Hybrid Parallelism

I. INTRODUCTION

Advancement in the deep learning drawn attention in the computer vision community in many areas especially in face recognition as one of the most significantly research topics overcoming the challenges of the recognition problem. In private and public sectors there are various uses of face recognition applications for person authentication, surveillance systems, object detection in autonomous vehicles, etc. [1]. A Face Recognition is a technology that identifies or verifies individuals based on facial features using computer vision and machine learning. Although face recognition algorithms have reached the accuracy level under certain condition, the face recognition algorithm still affected by the external and internal variation.

Many researchers working in face recognition area proposed algorithms based on deep learning in the literature which shown to achieve a better performance in terms of the improved processing time and more accuracy [1][2][3]. Deep learning training for the real time multiple face recognition algorithm is still a problem which are computationally complex and very greedy about the resources. Recently Deep learning algorithms for multiple face recognition implemented on GPU systems have been shown faster performance and better accurate results successfully.

Why Parallelism in Face Recognition?

A multiple face recognition system is parallelized to speed up the computational performance. The sequential processing algorithms need more processing time and may not give required accurate results in real time. The importance of Deep Neural Networks (DNNs) has been increased in various fields like big data analysis, autonomous cars, image and object recognition, etc. Deep learning training for the multiple face recognition is computationally

very greedy about the resources with respect to the processing cycles and the space complexity too. In recent time there is huge amount of data for face recognition available for training and testing which is growing with various attributes daily. These systems applications need to recognize the multiple faces in real-time is a critical in demand. The heavy computational load in neural networks requires faster, efficient methods which can be achieved using parallelism.

A possible strategy for making significant performance enhancements for multiple face recognition is provided by the convergence of high-performance computing (HPC) using parallel implementation of algorithms with intra-layer and inter-layer parallelism with data-intensive approaches.

In this research paper we propose and mainly focus on implementation of Multiple Face Recognition algorithms using Deep Learning by Layer wise Fine-Grained Parallelism. Also, we will show our analysis, experimental results and comparison of intra-layer and inter-layer parallelism algorithms with hybrid approaches.

The remainder of this paper is organized as follows. Section II reviews our study of literature survey, it's summary and related work on multiple face recognition algorithms. Our proposed framework and Research Methodology is described in Section III. Experimental Setup and Algorithm presented in Section IV and Results are discussed in Section V. We conclude our paper with Section VI Conclusions followed References used for our research work and this paper.

II. LITERATURE SURVEY AND RELATED WORK

In this section we provide the summarized overview of prominent face recognition methods, techniques and architectures developed and used by various researchers and analyse these models as a part of our literature survey study for multiple face recognition. The detailed study of literature survey is already published in our research paper titled "A Comprehensive Analysis of Algorithms in Multiple-Face Recognition" [1].

Face Recognition Approaches

Hadi Santoso et al. [2] proposed parallel multiple-face detection architecture implementation approach based on Viola and Jones' framework. The Viola and Jones' framework for face detection has 4 steps as Pre-Processing, Determining Haar-like features, computing integral image and Detection with cascading classifiers as shown in Figure 1.

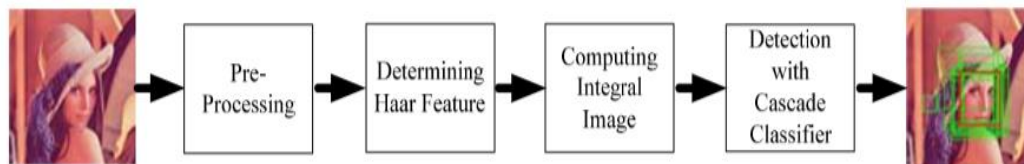


Figure 1: Viola-Jones method for Face detection

The authors demonstrated implementation of their algorithm on parallel architecture to improve performance by 20 times to 100 times on 2 to 4 Core CPUs with efficient data movement and increased energy efficiency on multithreaded parallel programming.

Kanokmon Rujirakul et.al. [3] in their research paper has proposed multiple face recognition using Parallel Expectation-Maximization PCA Face Recognition Architecture (PEM-PCA) methodology. Authors have observed that the efficacy is directly proportional to the no. of cores along with the no. of threads which are running concurrently in computing device. PEM-PCA has a justified parallel method implementation, and it performs better than the other PCA methods, the results are dependent on the total amount of training photos.

Aashna R. Bhatia et.al.[6] have proposed parallel implementation of face detection algorithm on Graphics Processing Units (GPU) and discussed about Viola-Jones face identification algorithm. This algorithm, which was among the first of its kind, is examined in considerable depth. The primary topics of discussion in this article are advantages and disadvantages of the Viola Jones algorithm. The main objective is to accelerate the technique's computing speed by leveraging CUDA and Open CV on GPUs to implement the algorithm in parallel. This method helps in using the properties of the GPU and comparing the outputs of algorithm's serial and parallel processing. The CUDA runtime API is called indirectly, and the major processing power is spent during the process of switching processes between

main memory and GPU memory. It functions most well in environments with limited resources; the parallel implementation requires less time for processing. The detection rate is higher than the usual technique, and using GPUs is the more efficient way to process big amounts of data.

Fang Gao et.al. [8] have proposed low power parallel CPU-Accelerator heterogeneous multicore architecture for implementation of face detection system. A fundamental CPU version based on the cascade classifier and the local binary patterns operator for face identification prototype was developed and put into use. The steps to implement Parallella face detection model is as in Figure 2.

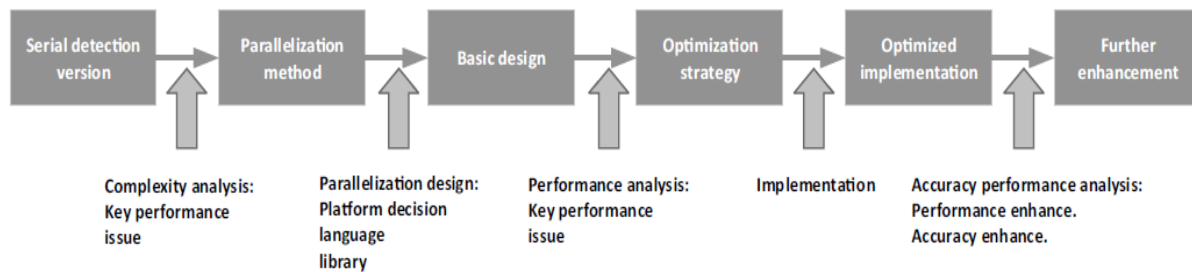


Figure 2: Parallella model implementation steps

A face detection experiment was implemented to assess the performance of the computer. The outcomes of the experiments show that the suggested execution reached a level of precision that was extremely consistent with that of the dual-core ARM, achieving a speedup that was 7.8 times greater. The findings of the experiments demonstrate that the proposed implementation offers significant performance benefits over alternative approaches.

Shivashankar J. Bhutekar et.al.[9] developed a GPU based system that processes parallel face detection and recognition in real time on NVIDIA GeForce GTX 770 GPU, which is much faster than on the CPU. One of the algorithms recognizes a face and sends the information on to the face processing system. The findings of the experiments make it abundantly evident that the GPU plays an essential part in the process of parallel computing. The GPU-based technology detects and recognizes faces in real-time more quickly than a CPU. The algorithms work concurrently, as the first one detecting faces and the second sending them to the face-processing system.

Applications that rely on sequential algorithms are no longer able to boost their performance by relying on the scalability of the underlying technology, prompted by Sharanjit Singh et al. [13] who developed a new methodology. Applications that do image processing shows a high degree of parallelism, making them a suitable source for multi-core platforms to draw their processing power. All are aware that the medical images must have the highest possible level of clarity, and it must be recovered as quickly as is physically possible. This demands a higher amount of computing power than a typical sequential computer. Through a method known as parallelization, the researchers were able to successfully complete this task. There are few features like Granularity (Coarse-grained and Fine grained), Synchronization, Latency, Scalability, Speedup & efficiency and Overheads that the parallel program must have to get high performance as expected. An example of how parallel computing can be used in medical imaging is explored in great depth.

Authors I P Skirnevskiy et.al. [14] in their article described the benefits of utilizing GPU in the large amounts of digital image processing. The research paper describes concise explanation of a parallel computing technique and its applications in a variety of fields. The parallel computing on GPU can be done using either Nvidia CUDA technology or AMD's FireStream technology. The research paper elaborated the comparison of the performance with the mixed percentages use of GPU with percentage CPU, the result is as shown in table 1 [14][23].

Table 1: Time of calculations

Proportion	Time (In Sec)	
	210 X 182 pixels	160 X 128 pixels
Single CPU thread	627.9577	184.0913
Double CPU thread	713.9502	193.8745
Half CPU + Half GPU	336.0062	98.8670
1/8 CPU + 7/8 GPU	139.9920	39.2366

Single GPU	111.7244	29.8882
------------	----------	---------

The implementation of a program can be greatly speed up by utilizing parallel computing with the graphics processing unit (GPU).

S. Vijayakumar et.al. [18] proposed the use of GPU that plays a significant part in parallel processing. The majority of computers today come equipped with GPUs, which allow for significantly quicker graphics processing. Parallel algorithms that are executed on GPUs can frequently achieve up to a one hundred times faster than CPU algorithms. In this study, researchers explore the ways in which GPU are increasingly being used for programming medical image processing. In medical image processing, the development of parallel methods for GPU has been resulted in higher processing speed as well as increased accuracy.

Preeti Kaur et.al. [20] proposed application program method that do high degree of parallelism for image processing and are an ideal source on multicore platforms. Finding the various parameters, including serial time, parallel time, fork time, join time, and overheads, is the primary topic of this research study paper. The findings indicate that the utilization of parallel computing capabilities in MATLAB, in conjunction with multicore platforms, can significantly increase the rate at which images are processed. In this study, a parallel version of a sequential image processing technique, known as the contrast algorithm, is presented.

When image processing application need to process many images, one can use pipeline processing of images. In the proposed pipeline processing, these images will be in different phases during the same time as shown in figure 3.

Time	Phase 1	Phase 2	Phase 3	Phase 4	Phase 5
T1	Image 5	Image 4	Image 3	Image 2	Image 1
T2	Image 6	Image 5	Image 4	Image 3	Image 2
T3	Image 7	Image 6	Image 5	Image 4	Image 3
T4	Image 8	Image 7	Image 6	Image 5	Image 4

Figure 3: Parallel Pipeline Processing

The implementation of parallel processing was roughly 2.5 times faster in terms of performance than the processing of sequential data, as shown in figure. 4.

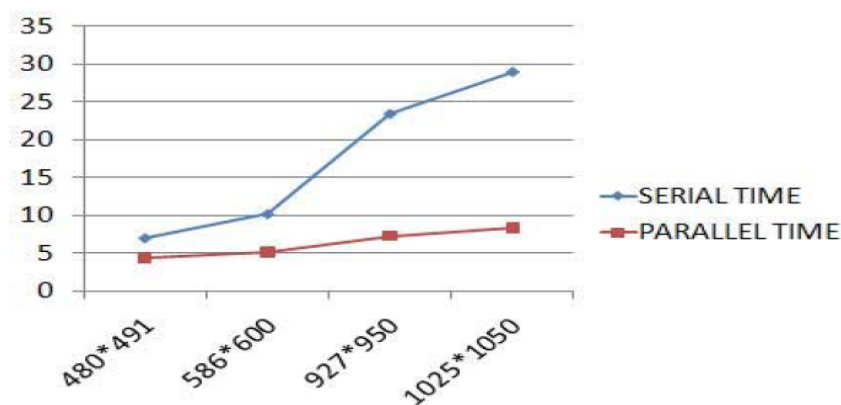


Figure 4 : Results of speedup obtained from different images

Classification and Summarization

The comparative graph of various methods developed and the accuracy of these models for face verification and face identification as proposed by various researchers for face verification and face identification is as shown in figure 5 [22]. It is observed that there are high levels of accuracy achieved by various methods reaching nearer to 100% for face verification but coming to face identification, the accuracy is around 80% and there is scope of improvement in it.

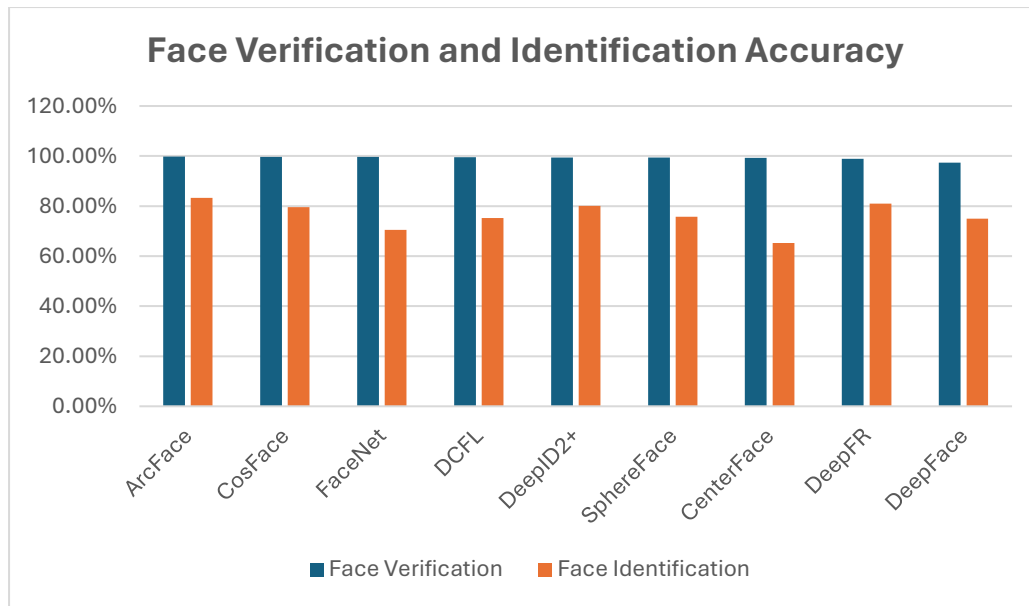


Figure 5 : Comparison graph for various face verification and identification methods

As a conclusion of this literature survey, we observed that face recognition applications implemented on conventional architectures with single CPU, cannot meet real-time deadlines. This is due to these applications' complexity and stringent performance requirements. Parallel computing presents an opportunity to speed up the execution of certain algorithms. The time complexity can be reduced using GPU which acts as a main computing processor for image processing application speedup. Many researchers also suggested utilizing layer-wise fine-grained parallelism for multiple face detection using deep learning as their future work.

III. RESEARCH METHODOLOGY

3.1 Neural Network Architecture for Multiple Face Recognition

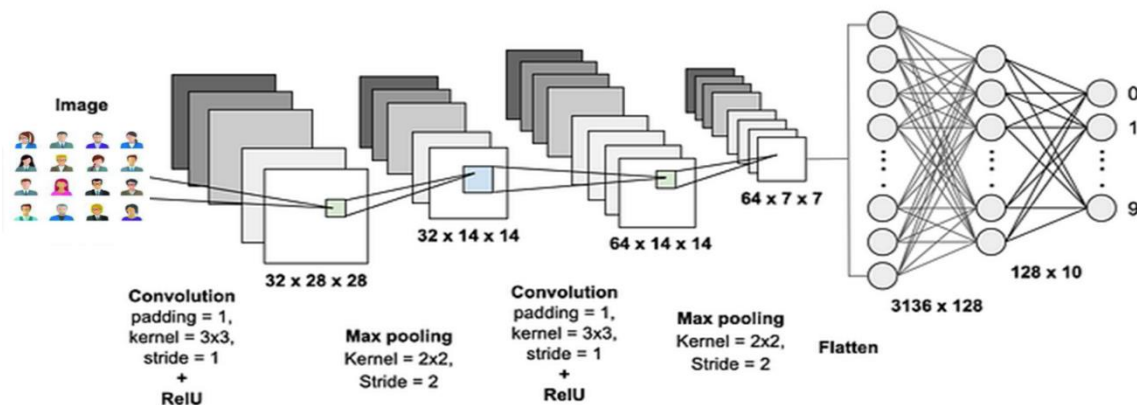


Figure 6. Architectural diagram for multiple face recognition

The overview of Neural Network Architecture for multiple face recognition using CNN is as shown in Figure 6. The neural network used for face recognition typically includes Convolutional Neural Networks (CNNs) due to their ability to capture spatial hierarchies in images. For our research work we used ResNet-50, but the methodology can apply to other architectures as well.

The general overview of workflow is as follows:

- 1. Model Training:** We used convolutional layers for feature extraction and applied softmax/triplet loss function for classification.

2. **Parallelism Implementations:** Using Intra-Layer Parallelism, we split the complete computations (convolutions) across multiple GPUs within a single layer and using Inter-Layer Parallelism to assign entire layers to different GPUs for sequential processing.
3. **Hybrid Approach:** To get better performance and faster results we combined both techniques based on layer complexity and hardware capabilities. This gives us the benefits of using both intra-layer and inter-layer parallelism.

The algorithm can be described using architecture shown in Figure 6 for multiple face recognition. It is an optimized pipeline algorithm consisting of various deep learning components for face recognition. Let's see the different steps to know how the face recognition operation process works:

- **Input Layer:** The first step starts with a preprocessed input image of size like 224x224 pixels with three color channels RGB. In preprocessing stage, it converts input image to standard dimensions image for giving input to CNN architecture. The process also involves normalization of pixel and applying various augmentation techniques to improve model's robustness.
- **Convolutional Layers:** The pre-processed image from the previous stage is given as input to the convolutional neural network layers. The CNN layers will extract low-level features of image like edges, corners, and textures. The CNN layer filters perform convolution operations on highlight patterns of the input image. The increasing number of filters captures complex features in deeper layers. ReLU (Rectified Linear Unit) activation functions are applied, and pooling layers are used to extract feature maps, reducing spatial dimensions preserving required critical information.
- **Residual Blocks:** Residual blocks are introduced to overcome vanishing gradient problems and improve training stability. The proposed architecture stabilizes gradient flow to improve the convergence rate and accuracy of the model.
- **Fully Connected Layers:** The outcome of convolutional neural network layers and residual block layers with high-dimensional features are flattened and given input to fully connected layers. These fully connected layers integrate the learned features and maps them for classification. All the neurons of these layers are connected to each neuron in the previous layer, which enable the model to capture global patterns. Regularization techniques used to prevent overfitting by deactivating random neurons during training.
- **Output Layer:** The last stage of algorithm completes with an output layer. Softmax activation function is used to identify classification related to different identities. L2-normalized embeddings are generated for similarity computation and transforming the extracted features into a compact vector representation. These are used for comparison with cosine similarity or Euclidean distance to identify whether two faces belong to the same individual.

3.2 Parallelism in the Neural Network

1. Intra-Layer Parallelism: The intra-layer parallelism in convolutional neural networks focuses on splitting face recognition algorithm operations on multiple available GPUs for efficient processing within single convolutional layer. The primary aim or main goal is to accelerate the computation of operations across multiple GPUs to reduce the computational overhead and requires less training time. Each GPU processes a part of input feature maps data concurrently, and their results are combined to form the final output. This approach is effective for computationally intensive convolutional layers which involve in a high number of matrix operations.

Mathematical Model: A convolutional layer operates on an input feature map and applies filters to extract various face recognition features like edges, textures, and patterns. The mathematical model for convolutional layer operation can be represented as:

$$Output_{i,j,k} = \sum_{u=1}^F \sum_{v=1}^F Input_{i+u,j+v,k} \cdot Kernel_{u,v,k}$$

Where F : Filter size (like 3x3 or 5x5), k: Number of Filters (depth of the output feature map).

In traditional application processing over single-GPU or CPU the operation will be executed in sequential manner which is not efficient and perform poorly. In proposed intra-layer parallelism method, this process is divided across GPUs by splitting the filters or input patches. Each GPU handles a subset of computations concurrently.

The computation for GPU_n can be expressed as:

$$GPU_n: \sum_{p \in P_n} Input_p \cdot Kernel_p$$

Where P_n represents the set of input patches or filter subsets assigned to GPU_n .

Each GPU processes its assigned subset of task independently, and the complete output consists of summation of these results.

2. Inter-Layer Parallelism: The inter-layer parallelism used with convolutional neural networks distributes the processing of entire layers across different GPUs to enhance efficiency and scalability of proposed multiple face recognition model. The main goal of this methodology is to optimize resource utilization and reduce processing time. The inter-layer parallelism allows concurrent processing of different stages of the convolution neural network.

In inter-layer parallelism, the computation is designed in a manner where the output of one layer (Layer L_1) is computed on one GPU (GPU_1) and then passed to the next GPU (GPU_2) for processing the subsequent layer (L_2). This allocation of sub task on each GPU ensures better operation performance and focuses on a specific layer, reducing the bottleneck of serial execution on a single GPU. This execution process continues as outputs of previous GPU is passed as an input to the next GPU in such manner that all layers are processed in parallel.

The mathematical representation of inter-layer parallelism can be expressed as follows:

$$GPU_i: Output_{L_{i-1}} \xrightarrow{GPU_i} Output_{L_i}$$

Where: GPU_i represents the i^{th} GPU in the pipeline and L_i represent the i^{th} layer of the neural network.

3. Hybrid Parallelism: We propose a new methodology that combines the benefits of intra-layer and inter-layer parallelism known as Hybrid Parallelism. The Hybrid Parallelism approach optimize the performance and resource utilization of convolutional neural network training across multiple GPUs. The details of Intra-Layer Parallelism and Inter-Layer Parallelism is already discussed in previous section. The proposed Hybrid parallelism takes the strengths of these two distinct parallelism techniques. The combination of these two techniques as a hybrid parallelism which can efficiently manage the diverse computational demands of different layers within a deep neural network, and this ensures optimal utilization of both computational resources and communication bandwidth.

With this hybrid approach, we propose our parallel multiple face recognition algorithm.

IV. EXPERIMENTAL SETUP AND ALGORITHM

We implemented our proposed Multiple Face Recognition algorithm with Hybrid Parallelism on Distributed systems with NVIDIA RTX 4080 GPUs and high-speed NVLink. The shared memory architecture is used for fast inter-GPU communication. TensorFlow 2.x with Horovod for multi-GPU processing Software used to implement code with cuDNN and NCCL optimization libraries. We used batch size of 64 and learning rate 0.001 with adaptive decay.

Dataset Description: For implementation, performance evaluation, and testing scalability of our proposed hybrid parallelism algorithm, we used two benchmark datasets namely Labeled Faces in the Wild (LFW) and MegaFace.

Labeled Faces in the Wild (LFW):

- Use Case: Primarily used for accuracy validation in low computational complexity environments.
- Characteristics: LFW has good number of images with moderate variations in pose, lighting, and expression which serves as a good dataset for checking performance metrics such as accuracy, inference time, and resource utilization.

MegaFace:

- Use Case: Dataset is suitable and designed for Scalability testing for recognizing many faces in high-complexity scenarios.
- Characteristics: Features a vast and diverse set of images with varied identities and complex real-world conditions. This ensures testing of the algorithm's ability to handle large-scale datasets without compromising efficiency or accuracy. This can be used to stress-test the hybrid parallelism model for workload distribution, GPU utilization, and memory efficiency.

Proposed Algorithm: Multiple Face Recognition Algorithm with Hybrid Parallelism

- **Input:** Dataset D with N face images, Neural Network M.
- **Output:** Classification labels or feature embeddings.

Step 1: Data Preprocessing

- Resize images to 224×224 pixels.
- Normalize pixel values to [0,1].
- Perform augmentation: random flips, rotations, and brightness adjustments.

Step 2: Model Initialization

- Load the model M – We use a pre-trained neural network such as ResNet-50 or initialize a new model.
- Initialize weights with Xavier Initialization for all layers which ensures balanced weight distribution.
- Assign GPUs based on hybrid parallelism:
 - Assign convolution layers to intra-layer parallelism.
 - Assign activation, pooling, and fully connected layers to inter-layer parallelism.

Step 3: Parallel Execution**Forward Pass:**

- Divide convolutional layers' operations among GPUs using intra-layer parallelism
- Pass intermediate results to the next layer using inter-layer pipeline.

$$Conv_L = \sum_{n=1}^{N_{GPU}} GPU_n (Input, Kernel_n)$$

Backward Pass (Training):

- Distribute gradient computations across GPUs
- Synchronize gradients and update weights

$$Gradient_{i,j,k} = \sum_{p \in P} \frac{\delta Loss}{\delta Kernel_{i,j,k,p}}$$

Update the kernel weights using the learning rate (η):

$$Kernel_{i,j,k} \leftarrow Kernel_{i,j,k} - \eta \cdot Gradient_{i,j,k}$$

Step 4: Inference

1. Divide input batch among GPUs.
2. Compute embeddings or class labels with the trained model.

Step 5: Evaluation: We compute the accuracy of the model as:

$$Accuracy = \frac{Correct\ Predictions}{Total\ Predictions} \times 100$$

V. EXPERIMENTAL RESULTS

To measure the performance of our algorithm framework we implemented and tested our model on Labeled Faces in the Wild (LFW) and MegaFace datasets. We have used intra-layer, inter-layer as well as hybrid parallelism. The various performance parameters (Metrics) are considered for our study are as follows:

1. **Training Time (T):** Training time is the time taken to finish one epoch in milliseconds (ms). This duration reflects the efficiency of parallelism process to handle computational workloads.

$$T = \text{Epoch Time} / \text{Total Epochs}$$

2. **Inference Speed (I):** Inference Speed is the time required to process a single image during inference in milliseconds (ms). This signifies the system's efficiency to make real-time face recognition predictions.

$$I = \text{Total Inference Time} / \text{Number of Images}$$

3. **Accuracy (A):** Accuracy means the percentage of correct predictions, measured in top-1 accuracy for classification tasks.

$$A = \text{Correct Predictions} / \text{Total Predictions}$$

4. **GPU Utilization (%):** GPU utilization gives the percentage of GPU capacity used during training and inference stage. A higher percentage of GPU utilization means better resource management and proper workload distribution.
5. **Memory Usage:** Memory used by GPU during model execution. Memory usage includes memory used for input data, feature extraction of model parameters, and storing intermediate results.
6. **Communication Overhead:** Time for data transfer between GPUs during parallel execution in milliseconds (ms).
7. **Energy Consumption:** Energy consumed by the GPUs and underlying hardware for training and inference.

In the experiment, we found that the Hybrid Parallelism performs better for both LFW Dataset and MegaFace Datasets. Performance analysis for LFW Dataset and MegaFace Dataset using Intra-Layer Parallelism, Inter-Layer Parallelism and Hybrid Parallelism is as shown in table 2.

Table 2: Performance analysis for LFW & MegaFace Dataset

	LFW Dataset			MegaFace Dataset		
Metric	Intra-Layer Parallelism	Inter-Layer Parallelism	Hybrid Parallelism	Intra-Layer Parallelism	Inter-Layer Parallelism	Hybrid Parallelism
Training Time/Epoch (ms)	450	600	380	750	1000	620
Inference Time/Image (ms)	70	50	40	100	80	65
Accuracy (%)	95.8	95.6	96.2	92.4	92.2	93.0
GPU Utilization (%)	75	60	85	70	55	82
Memory Usage (GB)	4.2	3.8	4.0	7.1	6.5	6.8
Communication Overhead (ms)	50	80	30	120	150	70
Energy Consumption	High	Moderate	Low	High	Moderate	Low

The performance graphs of Intra-Layer, Inter-Layer, and Hybrid parallelism computational execution on the LFW and MegaFace datasets are shown in Figure 7 and 8 for various metrics.

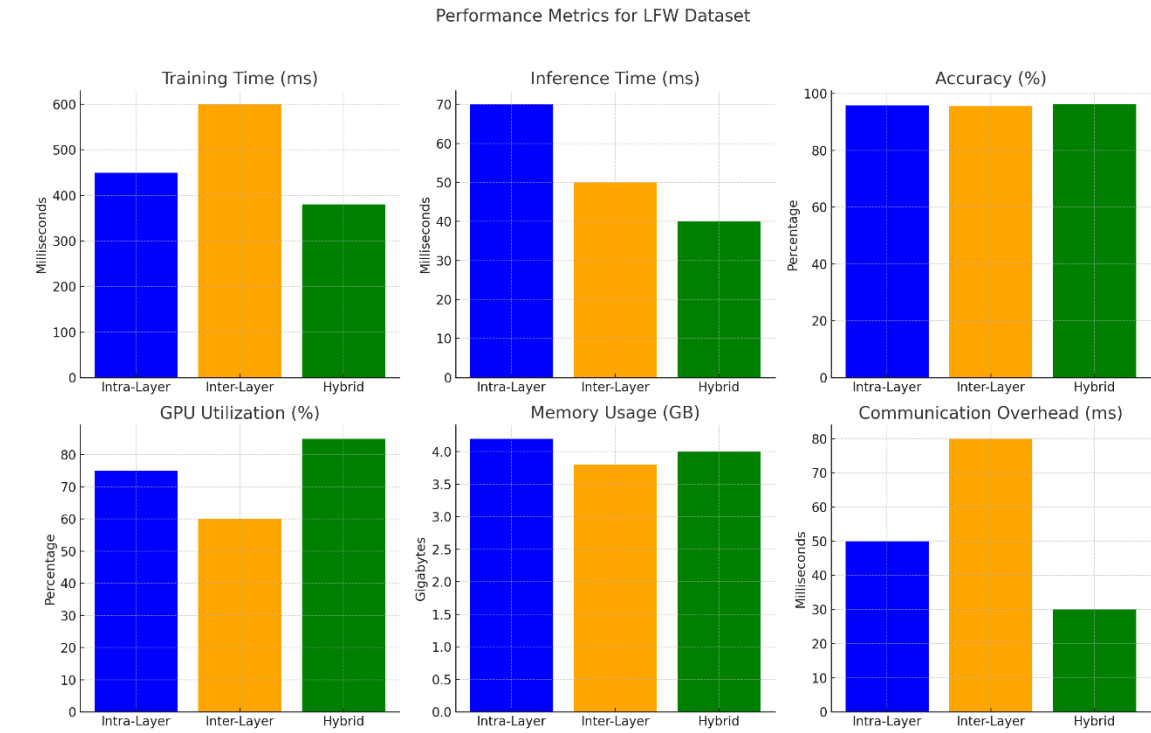


Figure 7. Performance metrics graph for LWF Dataset

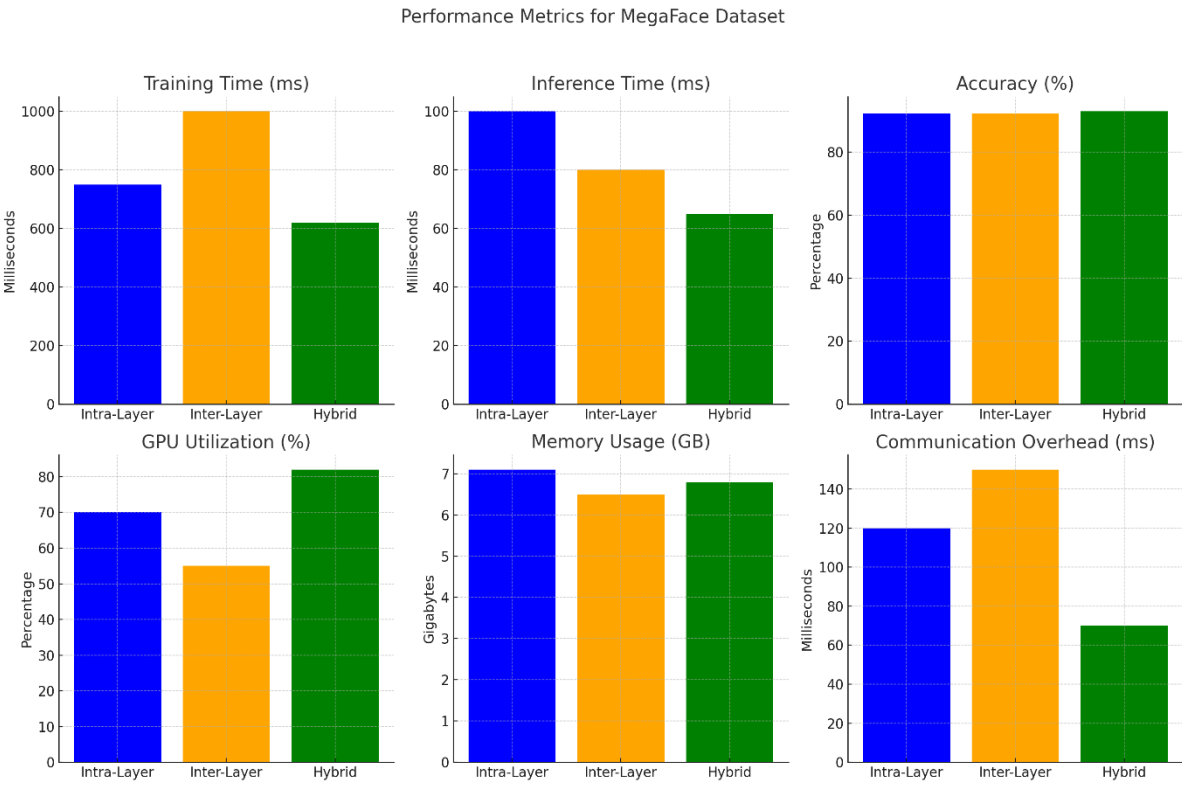


Figure 8. Performance metrics graph for MegaFace Dataset

From the Performance analysis table and performance graphs for LFW & MegaFace we can summarize the different parameters as follows:

1. **Training Time:** Compared Intra-Layer and Inter-Layer parallelism model the Hybrid parallelism performs better with shortest training time for across both datasets. Due to complex inter-GPU communication, Inter-Layer parallelism takes higher training time.
2. **Inference Time:** Like training time the Hybrid parallelism inference time performance is far better than other two techniques. In Hybrid approach, the resource sharing is optimized, and bottlenecks are reduced.
3. **Accuracy:** The accuracy for all three parallelism approaches has achieved similar accuracy. This is indication of that computational efficiency is at par due to parallelism strategies.
4. **GPU Utilization:** The GPU has been at higher level of utilization in Hybrid parallelism compared to other two methods. In Inter-Layer parallelism GPU utilization is lower due to communication delays.
5. **Memory Usage:** Due to efficient memory management, all three approaches using memory consistently with minor variations.
6. **Communication Overhead:** The communication overhead is lowest in Hybrid parallelism. In Inter-Layer parallelism it's highest, so it is less efficient for processing large datasets like MegaFace.

From these we can summarize that the hybrid parallelism approach provides an optimal balance of speed, resource utilization, and communication efficiency, making. Thus, we conclude that hybrid parallelism is the most effective methodology compared other methods.

The figure 9 shows the comparison of various performance parameters for LFW and MegaFace Datasets for Intra-Layer, Inter-Layer and Hybrid Parallelism models.

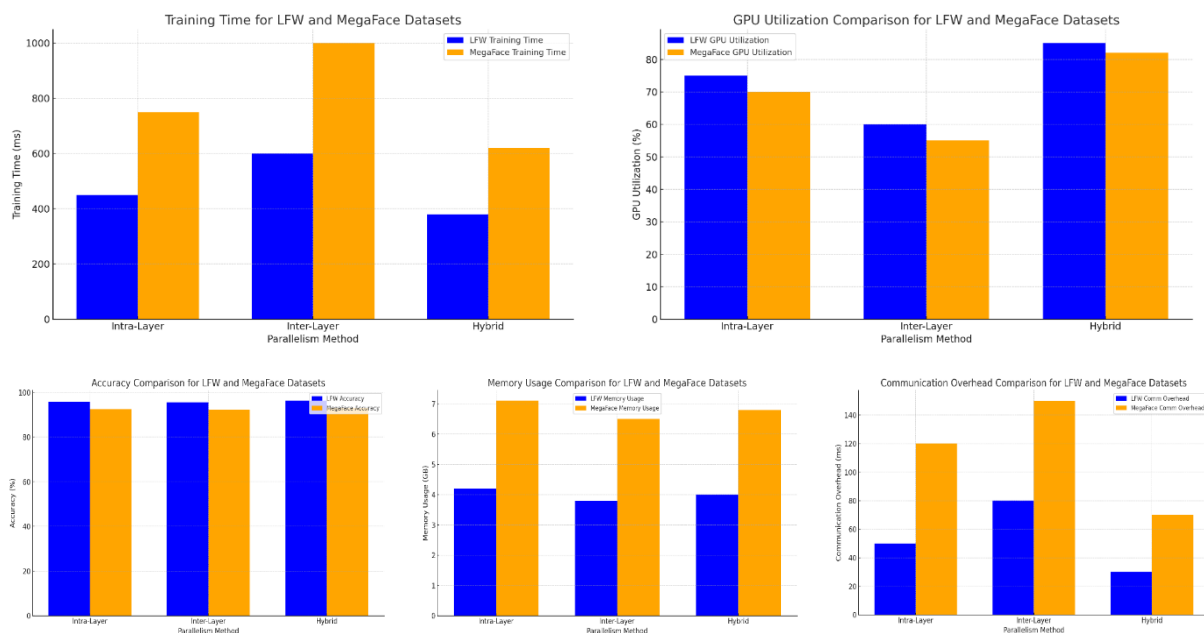


Figure 9 : Comparison of performance metrics for LFW and MegaFace Datasets using Intra-Layer, Inter-Layer and Hybrid Parallelism models.

Based on the graphs from Figure 9, we can summarize and compare various performance parameters for the LFW and MegaFace datasets in relation to Intra-Layer, Inter-Layer, and Hybrid Parallelism models.

1. **Training Time (ms):** It is observed that the Hybrid Parallelism takes lowest training time for both datasets compared to Intra-Layer and Inter-Layer models due to distribution of workloads. For LFW (smaller dataset size and moderate complexity) the training time is lower compared to MegaFace dataset (larger dataset size and higher complexity) for all models.

2. **Inference Time (ms):** Once again the inference time is lowest for Hybrid model compared to other two models. For the larger MegaFace dataset where due efficient workload distribution it minimizes latency and Hybrid model outperforms.
3. **Accuracy (%):** Three models perform well and near-identical accuracy (nearing 100%), with Hybrid model little more improvement on both datasets LFW and MegaFace datasets.
4. **GPU Utilization (%):** Hybrid achieves the highest GPU utilization as there is better resource allocation. Inter-Layer shows the lowest utilization due to inefficiencies in task scheduling. For LFW dataset, GPU utilization is nearly 80% for Hybrid and Intra-Layer models but slightly drops to nearly 70% for Inter-Layer whereas on MegaFace Dataset the Hybrid model achieves the highest utilization around 85%, while Intra-Layer nearly 75% and Inter-Layer about 65% due to inefficiencies in task scheduling. This concludes that GPU utilization improves with the scaled datasets for Hybrid model due to its ability to handle larger workloads effectively.
5. **Memory Usage (GB):** For both LFW and MegaFace datasets, Hybrid utilizes slightly less memory (~3.8 GB for LFW and ~6.5 GB for MegaFace) compared to the other models, reflecting its ability to minimize redundant storage demands.
6. **Communication Overhead (ms):** Hybrid exhibits significantly lower communication overhead in compared to other two models for both datasets, with a more visible advantage for the larger dataset like MegaFace. This underlines its effectiveness in balancing inter-device communication.

Overall, the Hybrid Parallelism model consistently performs well in comparison with Intra-Layer and Inter-Layer methods for both datasets along with its ability to handle larger dataset like MegaFace. The results for the LFW dataset emphasize Hybrid's efficiency in low-complexity environments, while MegaFace highlights its robust scalability for high-complexity, large-scale datasets.

Scalability with Increased Dataset Size: Let's see the performance analysis for different number of images i.e. various size of datasets as shown in table 3.

Table 3: Scalability performance with increased size Dataset for LFW Dataset

Dataset Size (Images)	Training Time (Intra-Layer, ms)	Training Time (Inter-Layer, ms)	Training Time (Hybrid, ms)	Accuracy (Hybrid, %)	GPU Utilization (Hybrid, %)
10,000	450	600	380	96.2	85
50,000	750	1000	620	96.0	82
100,000	1300	1800	1100	95.8	80
1,000,000	10,500	13,800	8,600	95.0	75

By observing table 3, we found that training time for Hybrid parallelism is consistently lowest for all size of dataset. Hybrid parallelism model is most efficient in scaling with larger datasets.

Training Time: The training time for Intra-Layer parallelism model increases linearly with dataset size and is faster than Inter-Layer. As discussed in previous section the Inter-Layer parallelism model takes the highest training times due to communication overhead between layers.

Accuracy: Hybrid parallelism model gives the high accuracy ranging from 96.2% for 10,000 images to 95.0% for 1,000,000 images. The increasing size of the dataset has a very small effect on performance of the proposed model.

GPU Utilization: It is found that GPU utilization for Hybrid parallelism model decreases from 85% for smaller datasets to 75% for the largest dataset. This is probably due to bottlenecks in managing resources as the dataset size increases.

Research Outcomes (Performance analysis) of proposed Hybrid Parallelism model for Multiple Face Recognition

1. Performance Improvements: From implementation of our proposed Hybrid Parallelism model, we found that the training time for the hybrid parallelism approach there is 20% reduction in training time compared to intra-layer parallelism. This is due to proper splitting computational load among intra-layer and inter-layer computations, minimizing computation redundancy and idle GPU time. As there is efficient inter-GPU communication and task distribution across multiple GPUs, the inference time for Hybrid parallelism is 43% faster than inter-layer parallelism. This improvement is very much useful for real-time multiple face recognition systems and other surveillance or authentication systems applications. Accuracy for the hybrid parallelism model is enhanced by 0.4%, achieving higher computational efficiency and avoiding bottlenecks like in inter-layer model.

2. Scalability with Larger Datasets: Our proposed Hybrid parallelism model can handle larger datasets like LWF, MegaFace excellently with better scalability as compared to intra-layer and inter-layer models. Hybrid model maintained high accuracy with manageable training time as workload is balanced and distributed across multiple GPUs.

3. Energy Efficiency: Due to optimized task scheduling and reduced idle time for GPUs the proposed hybrid technique was 15% more energy-efficient and minimized power consumption during computation than other approaches.

Thus, we found that Hybrid parallelism is most effective technique for multiple face recognition applications, with higher training speed, superior inference performance, more accuracy, scalability with larger dataset, and energy efficiency.

VI. CONCLUSIONS

In this paper, we proposed intra-layer and inter-layer parallelism model along with combination of these as Hybrid parallelism model framework for real-time multiple face recognition. Compared to intra-layer and inter-layer parallelism methodology the hybrid approach provides an optimal performance at higher speed, better resource utilization, and communication efficiency, making it the most effective strategy overall for multiple face recognition with acceptable recognition rate even for increased dataset sizes. However, it is observed that the GPU utilization decreases as dataset size grow. This can be further improved by optimization techniques like dynamic load balancing or memory management strategies.









To summarize, our proposed algorithm hybrid parallelism technique (which combines features of intra-layer and inter-layer parallelism models) can be used to optimize both training and inference of multiple face recognition models. The hybrid model balances computational load, minimizes communication overhead, and ensures efficient utilization of available GPU resources. These establish the hybrid parallelism as a robust deep learning models framework making it suitable for real-time multiple face recognition with acceptable recognition rate across large datasets while maintaining computational and energy efficiency.

REFERENCES

- [1] Vijaykumar P. Mantri, Yogesh Deshpande, and Sandip Thite. "A Comprehensive Analysis of Algorithms in Multiple-Face Recognition." *Nanotechnology Perceptions* 20, no S8 (2024): 982-997.
- [2] Santoso, Hadi, and Reza Pulungan. "A Parallel Architecture for Multiple-Face Detection Technique Using AdaBoost Algorithm and Haar Cascade." *ISICO 2013* 2013 (2013).
- [3] Rujirakul, Kanokmon, Chakchai So-In, and Banchar Arnonkijpanich. "PEM-PCA: A parallel expectation-maximization PCA face recognition architecture." *The Scientific World Journal* 2014 (2014).
- [4] Fatemi, Hamed, Henk Corporaal, Twan Basten, P. P. Jonker, and R. P. Kleihorst. "Implementing face recognition using a parallel image processing environment based on algorithmic skeletons." In *10th Annual Conference of the Advanced School for Computing and Imaging (ASCI 2004)*, Port Zélande, Ouddorp, June 2-4, 2004, pp. 197-202. Advanced School for Computing and Imaging (ASCI), 2004.
- [5] Pande, Varun, Khaled M. Elleithy, and Laiali Almazaydeh. "Parallel processing for multi face detection and recognition." (2012).

- [6] Bhatia, Aashna R., Narendra M. Patel, and Narendra C. Chauhan. "Parallel implementation of face detection algorithm on GPU." In 2016 2nd International Conference on Next Generation Computing Technologies (NGCT), pp. 674-677. IEEE, 2016.
- [7] Ibrahim, Dalia Shouman, and Salma Hamdy. "Parallel architecture for face recognition using MPI." *International Journal of Advanced Computer Science and Applications (IJACSA)* 8, no. 1 (2017): 425-430.
- [8] Gao, Fang, Zhangqin Huang, Shulong Wang, and Xinrong Ji. "Optimized parallel implementation of face detection based on embedded heterogeneous many-core architecture." *International Journal of Pattern Recognition and Artificial Intelligence* 31, no. 07 (2017): 1756011.
- [9] Bhutekar, Shivashankar J., and Arati K. Manjaramkar. "Parallel face Detection and Recognition on GPU." *International Journal of Computer Science and Information Technologies* 5, no. 2 (2014): 2013-2018.
- [10] Guerfi, Imene, Lobna Kriaa, and Leïla Azouz Saïdane. "An Efficient GPGPU based Implementation of Face Detection Algorithm using Skin Color Pre-treatment." In ICSOFT, pp. 574-585. 2020.
- [11] Watanabe, Eriko, and Kashiko Kodate. "Implementation of a high-speed face recognition system that uses an optical parallel correlator." *Applied optics* 44, no. 5 (2005): 666-676.
- [12] Iqbal, Mohd, and Sandeep Raghuwanshi. "Analysis of digital image processing with parallel with overlap segment technique." *International Journal of Computer Science and Network Security (IJCSNS)* 14, no. 6 (2014): 52.
- [13] Singh, Sharanjit, Parneet Kaur, and Kamaldeep Kaur. "Parallel computing in digital image processing." *International Journal of Advanced Research in Computer and Communication Engineering* 4, no. 1 (2015): 183-186.
- [14] Skirnevskiy, I. P., A. V. Pustovit, and Mariya Ovseevna Abdrashitova. "Digital image processing using parallel computing based on CUDA technology." In *Journal of Physics: Conference Series*, vol. 803, no. 1, p. 012152. IOP Publishing, 2017.
- [15] Murali, Vengayil Nayana., C, Rahul., "Image Processing Application Using Parallel Computing." *International Journal of Science and Research (IJSR)* (2016).
- [16] Stamopoulos, Charalambos D. "Parallel image processing." *IEEE Transactions on Computers* 100, no. 4 (1975): 424-433.
- [17] Sathesh, A., and Edriss Eisa Babikir Adam. "Hybrid Parallel Image Processing Algorithm for Binary Images with Image Thinning Technique." *Journal of Artificial Intelligence* 3, no. 03 (2021): 243-258.
- [18] "Parallel Algorithm for Medical Image Processing using GPU Computing", *International Journal of Emerging Technologies and Innovative Research*, ISSN:2349-5162, Vol.5, Issue 8, page no.640-642, August-2018,
- [19] Bräunl, Thomas. "Tutorial in data parallel image processing." *Australian Journal of Intelligent Information Processing Systems (AJIIPS)* 6, no. 3 (2001): 164-174.
- [20] Kaur, Preeti. "Implementation of image processing algorithms on the parallel platform using Matlab." *Int. J. Comput. Sci. Eng. Technol* 4, no. 06 (2013): 696-706.
- [21] Rawas, Soha, and Ali El-Zaart. "Precise and parallel segmentation model (PPSM) via MCET using hybrid distributions." *Applied Computing and Informatics* (2020).
- [22] Shepley, Andrew Jason. "Deep learning for face recognition: a critical analysis." *arXiv preprint arXiv:1907.12739* (2019).
- [23] Goyat, Suman, and A. K. Sahoo. "Evolution of Remote Sensing and Graphical Information System using CPU-GPU Platform."

BIOGRAPHIES OF AUTHORS

	<p>Vijaykumar P. Mantri    received the B.E. in Computer Science and Engineering from Dr. Babasaheb Ambedkar Marathwada University (BAMU), Aurangabad, India in 1999, M.Tech. in Computer Science and Engineering from Visveswaraya Technological University (VTU), Belgaum, India in 2008 and currently pursuing Ph.D. (CSE) from Vishwakarma University, Pune, India.</p> <p>Currently, he is working as Senior Assistant Professor at the School of Computer Engineering at MIT Academy of Engineering, Alandi, Pune, India.</p> <p>He presented and published many Research and Technical Papers in National and International Journals, and Conferences. His main research interests are Machine Learning and Deep Learning.</p> <p>He can be contacted at email: vijay.mantri2000@gmail.com</p>
	<p>Sandip S. Thite    holds a Ph.D. in Engineering with a specialization in Artificial Intelligence and Cybersecurity from Bharati Vidyapeeth, Pune, India in 2022. He received the M.E. Computer Engineering from Bharati Vidyapeeth, Pune, India in 2011.</p> <p>He is currently working as Associate Dean of the Faculty of Science & Technology at Vishwakarma University, Pune. He has more than 15 Years of academic experience and 9 months of Industry. He also serves as the Head of the Computer Engineering Department, contributing to academic leadership and research excellence within the university's vibrant ecosystem.</p> <p>He has played a pivotal role in advancing academic programs in Artificial Intelligence and Data Science (AI & DS). These programs emphasize industry-relevant skills like machine learning, big data analytics, neural networks, and cybersecurity. He has made significant contributions to the Research field through the number of Paper Publications and Datasets Development in National and International Journals, and Conferences, patents and copyrights, and collaborative research at National and International level. His main research interests are Computer Networks, Cybersecurity and Artificial Intelligence & IoT.</p> <p>He can be contacted at email: sandip.thite@vupune.ac.in</p>