Parallel Implementation of PCA and LDA for Facial Recognition

Sujal Sanjay Awargand (S20220010213), Rohan Ramesh Vinkare (S20220010244), Jathin Reddey (S20220010241)

Department of Computer Science and Engineering

1 Introduction

Facial recognition has emerged as a cornerstone technology in the realm of biometric authentication, surveillance, and human-computer interaction. As societies increasingly adopt automated systems for security, identity verification, and personalized services, the demand for reliable and real-time facial recognition systems has grown exponentially. However, achieving high accuracy while maintaining low latency remains a significant technical challenge due to the high dimensionality of image data and the computational intensity of underlying algorithms. This project addresses these challenges through the implementation and optimization of classical machine learning techniques—Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA)—enhanced using parallel computing paradigms.

PCA and LDA are well-established statistical techniques used for feature extraction and dimensionality reduction. In the context of facial recognition, PCA is used to identify the directions (principal components) that capture the maximum variance in facial images, effectively transforming high-dimensional input data into a lower-dimensional subspace known as "eigenfaces." On the other hand, LDA is a supervised method that focuses on maximizing class separability, thereby creating a set of discriminative features called "fisherfaces." Both methods have been widely adopted in academic and industrial applications due to their interpretability and computational efficiency compared to deep learning alternatives.

Despite their advantages, the traditional sequential implementations of PCA and LDA are not suited for real-time applications, especially when dealing with large datasets. The primary bottlenecks arise from operations such as covariance matrix computation, eigenvalue decomposition, and matrix multiplications—all of which scale poorly with increased image count and resolution. In high-performance computing (HPC) contexts, these challenges are typically addressed through parallel processing strategies that distribute computational workloads across multiple cores or specialized hardware like Graphics Processing Units (GPUs).

This project explores the use of two major parallel computing models—OpenMP and CUDA—to accelerate PCA and LDA-based facial recognition pipelines. OpenMP facilitates multi-threading on multi-core CPUs, making it suitable for shared-memory systems with minimal programming overhead. CUDA, developed by NVIDIA, enables massive parallelism by leveraging thousands of GPU cores to perform matrix operations and pixel-level transformations simultaneously. By implementing both CPU and GPU-based versions of PCA and LDA, this study provides a comparative analysis of the performance gains, scalability, and accuracy retention across different hardware architectures.

The dataset used for experimentation is the AT&T Face Dataset, a standard benchmark comprising grayscale images of 40 individuals. This dataset provides a manageable yet sufficiently diverse set of facial images, enabling rigorous testing of recognition accuracy, execution time, and throughput across implementations. The algorithms were implemented in C++ for performance, with OpenCV handling image processing tasks and OpenMP/CUDA used for parallelization.

Through this study, we aim to demonstrate that classical algorithms like PCA and LDA—when optimized using parallel computing—can still offer competitive performance in modern facial recognition tasks. We also highlight the trade-offs involved in CPU versus GPU parallelism and establish baseline metrics for future work in real-time, scalable biometric systems.

2 Objective of the Work

The primary objective of this project is to develop an efficient and scalable facial recognition system by implementing and optimizing classical machine learning algorithms—Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA). These algorithms are used to extract meaningful features from facial images and perform identity recognition with high accuracy. However, traditional sequential implementations are computationally expensive and not suitable for real-time applications.

To address this, the project focuses on parallelizing PCA and LDA using two different approaches: OpenMP for multi-core CPU processing and CUDA for GPU-based acceleration. By doing so, the goal is to significantly reduce execution time while preserving recognition accuracy. The project also aims to conduct a comparative performance analysis between sequential, OpenMP-based, and CUDA-based implementations using standard performance metrics. Ultimately, this work seeks to highlight the effectiveness of parallel computing in enhancing traditional facial recognition techniques for real-world deployment.

3 Major Contributions

This project presents a comprehensive implementation and optimization of classical facial recognition algorithms—Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA)—enhanced using parallel computing paradigms. The major contributions of this work span algorithmic adaptations, parallelization strategies, and performance evaluation across CPU and GPU environments.

3.1 Insights from Referenced Techniques

While the project did not rely on any specific academic paper, it draws heavily from foundational methodologies in face recognition using eigenfaces and fisherfaces. The Eigenfaces approach, introduced by Turk and Pentland, forms the backbone of the PCA-based method. It emphasizes projecting high-dimensional facial images into a reduced subspace defined by the top eigenvectors (principal components), allowing for efficient recognition using Euclidean distance. A key takeaway from this approach is that most of the facial variation can be captured with a relatively small number of components.

On the other hand, the Fisherfaces approach improves upon PCA by considering class separability. Derived from Linear Discriminant Analysis, it maximizes between-class scatter while minimizing within-class scatter, making it more robust to variations such as illumination and facial expressions. This dual-stage technique—first applying PCA to reduce dimensionality and then applying LDA—was instrumental in achieving better recognition accuracy in this work.

These foundational methods provided the conceptual basis for the project's goal: to retain classical model accuracy while dramatically improving performance via parallel processing.

3.2 Experimental Setup

All experiments were conducted on a personal machine—ASUS Vivobook 15 Pro, equipped with a 16-core CPU, 16 GB of RAM, and an NVIDIA RTX 3050 GPU (4GB VRAM), which supports CUDA acceleration. This setup allowed for testing both CPU-based parallelism using OpenMP and GPU-based acceleration using CUDA.

The AT&T Face Dataset (previously ORL database) was selected for training and evaluation. This publicly available dataset consists of 400 grayscale images of 40 subjects, each with 10 images, sized 112x92 pixels. The dataset was ideal for benchmarking both the performance and scalability of implemented algorithms.

The entire system was developed in C++, leveraging the OpenCV library for image handling and preprocessing. OpenMP directives were used to implement multithreaded sections for the CPU version, while CUDA kernels were written for parallel computation on the GPU. Key stages of parallelization included mean computation, covariance matrix construction, eigenvector calculation, and projection steps in both PCA and LDA.

The experiments compared three implementations: sequential, OpenMP, and CUDA, focusing on execution time, speedup, and throughput. All performance tests were conducted using the same dataset and machine environment to ensure fairness in comparison.

Summary of Contributions

- Implemented classical PCA and LDA algorithms for facial recognition from scratch.
- Optimized PCA and LDA using OpenMP (CPU parallelism) and CUDA (GPU acceleration).
- Achieved up to 10x speedup in PCA using CUDA over the sequential baseline.
- Validated performance improvements while maintaining recognition accuracy.
- Presented comparative analysis of CPU vs. GPU scaling efficiency on real-world hardware.

4 Performance Measuring Metrics

To evaluate the efficiency and effectiveness of different implementations (Sequential, OpenMP, and CUDA) of PCA and LDA for face recognition, the following performance metrics were employed. Each metric was selected to provide insights into a distinct aspect of system performance:

1. Execution Time

Justification: This is the most direct measure of how quickly an algorithm completes. It was crucial to compare how each version (Sequential, OpenMP, and CUDA) reduced the total time taken for PCA and LDA-based recognition. Lower execution time directly correlates with the system's ability to scale for real-time or near real-time face recognition applications.

2. Speedup

Formula:

$$\text{Speedup} = \frac{T_{\text{sequential}}}{T_{\text{parallel}}}$$

Justification: Speedup quantifies the relative improvement in performance due to parallelization. It shows how much faster the parallel (OpenMP or CUDA) version performs compared to the sequential version. It is especially useful for understanding the effectiveness of utilizing multiple CPU cores or GPU threads.

3. Efficiency

Formula:

$$\text{Efficiency} = \frac{\text{Speedup}}{\text{Number of Cores/Threads}}$$

Justification: Efficiency measures how well the computational resources are utilized. Even with many threads or GPU cores, if the performance doesn't scale accordingly, the efficiency drops. This metric helped identify potential bottlenecks or underutilization in the parallel implementations.

4. Recognition Accuracy

Justification: While optimizing performance is essential, it should not come at the cost of accuracy. The recognition accuracy metric measures how effectively the algorithm identifies the correct individual from test images. This ensures that both PCA and LDA implementations are not just faster but also reliable.

5. Confusion Matrix (for LDA)

Justification: The confusion matrix provides a class-wise breakdown of predictions, showing true positives, false positives, etc. For LDA, which is classification-driven, this matrix was crucial in understanding which classes (individuals) were being misclassified and why. It also provided insight into how well Fisherfaces generalized over varying lighting and expression conditions.

6. Memory Utilization (Qualitative Observation)

Justification: Although not measured numerically, memory consumption patterns were observed, especially in GPU (CUDA) implementations. The high-dimensional nature of face datasets leads to large matrices during covariance and scatter computations, making memory management a key performance consideration.

By combining these metrics, the project holistically evaluated not only raw performance gains through parallelization but also retained a focus on accuracy and real-world applicability. The parallel systems (OpenMP and CUDA) showed measurable speedups, and the CUDA implementation, in particular, significantly reduced execution time while maintaining recognition accuracy — validating the choice of metrics used.

5 5. Results / Outcomes

This section presents the performance comparison and outcomes of implementing PCA and LDA using Sequential, OpenMP, and CUDA approaches.

Sequential PCA

The sequential PCA implementation showed that the execution time increased significantly with the number of images. For 400 images, the total processing time was approximately 40,000 milliseconds. Although the approach maintained recognition accuracy, it was not practical for real-time applications due to its high computational cost.

OpenMP PCA

The OpenMP-based PCA implementation achieved a significant speedup—approximately 3.21×—reducing the execution time to about 12,000 milliseconds. However, CPU efficiency plateaued at around 50%, showing diminishing returns beyond a certain number of threads due to parallel overhead.

CUDA PCA

The CUDA version of PCA demonstrated the best performance, reducing the total processing time to just 4,000 milliseconds—a 10× speedup over the sequential implementation. The massive parallelism provided by GPU threads made CUDA suitable for large-scale and real-time face recognition tasks.

Sequential LDA

Sequential LDA was computationally heavy, taking approximately 35,000 milliseconds for 400 images. While the accuracy remained high, its long processing time made it unsuitable for real-time deployment, emphasizing the need for parallel solutions.

OpenMP LDA

OpenMP implementation reduced the execution time to approximately 25,000 milliseconds, resulting in a speedup of about 1.4×. The performance gains were moderate due to limitations in CPU core utilization beyond 8 threads.

CUDA LDA

The CUDA LDA implementation processed the dataset in roughly 15,000 milliseconds, yielding a 2.33× speedup over the sequential version. CUDA's parallelism effectively handled large matrices, showcasing its potential for real-time and scalable face recognition.

Table 1: Results and Outcomes of PCA and LDA Implementations

Algorithm	Implementation	Exec. Time (ms)	Speedup	Outcome Summary
PCA	Sequential	40000	1.00	Accurate but very slow; impractical for real-time use.
	OpenMP	12000	3.21	Substantial speedup; efficiency 50%; some overhead limits further scaling.
	CUDA	4000	10.00	Excellent speedup; highly scalable for real-time applications.
LDA	Sequential	35000	1.00	High accuracy but computationally intensive; unsuitable for real-time use.
	OpenMP	25000	1.40	Noticeable improvement; limited by CPU core constraints.
	CUDA	15000	2.33	Best performance; ideal for large datasets and real-time deployment.

The outcomes validate the effectiveness of parallel computing in classical face recognition algorithms. CUDA, in particular, demonstrated superior scalability and speed, supporting its use in real-time facial recognition systems.

6 Limitations and Future Scope

6.1 Limitations

- **Hardware Constraints:** The performance of both OpenMP and CUDA implementations was bounded by the available hardware. Although the NVIDIA RTX 3050 GPU offered considerable acceleration, limitations in memory bandwidth and GPU memory could restrict scalability to larger datasets. Similarly, OpenMP showed diminishing returns beyond 8 threads due to CPU limitations and overhead.
- **Dataset Size:** The project utilized the AT&T Face Dataset, which consists of only 40 subjects with 10 images each. While sufficient for testing, it does not reflect real-world complexity or scale. Larger datasets like CelebA or LFW pose additional challenges in terms of diversity and volume.
- **Algorithmic Complexity:** PCA and LDA, though efficient for dimensionality reduction and classification, are inherently linear. They may fail to capture non-linear variations such as lighting changes, pose differences, or facial expressions common in real-world scenarios.
- **Scalability:** Although parallel implementations improved execution time, their scalability is limited. The current approach may not perform efficiently on extremely large datasets without better optimization of memory usage and load distribution.

6.2 Future Scope

- **Hybrid Parallelization:** Future work could explore hybrid models that combine both CPU (OpenMP) and GPU (CUDA) parallelism. Such a model would allow intelligent distribution of tasks between CPU and GPU, leading to better performance and resource utilization.
- **Deep Learning Integration:** Incorporating Convolutional Neural Networks (CNNs) could significantly improve recognition accuracy and robustness. Deep learning models are more effective in handling complex variations and have already outperformed traditional methods like PCA and LDA in many benchmarks.
- **Real-time Processing:** For real-world applicability, extending the system to support live video stream-based recognition is essential. This would involve optimizing algorithm efficiency, reducing

data transfer latency, and managing GPU memory effectively.

- **Dataset Expansion:** Evaluating the performance of the proposed methods on larger and more diverse datasets like CelebA or LFW would provide deeper insights into generalizability and system robustness.
- **Advanced Parallelism Techniques:** Future implementations could benefit from newer parallel computing tools such as CUDA Streams, TensorFlow GPU acceleration, or even Vulkan compute shaders. These technologies may unlock further speedup and flexibility for handling high-dimensional data.

7 Observations from the Study

The study successfully addressed the research questions by exploring the feasibility and performance of parallelized PCA and LDA algorithms for facial recognition. The study design, which involved implementing both sequential and parallel methods using OpenMP and CUDA, provided a comprehensive comparison of performance across different configurations. The research primarily focused on optimizing computational speed without sacrificing accuracy, which was effectively achieved using parallelization techniques.

By testing various methods on a well-defined dataset (AT&T Face Dataset), the study demonstrated that both OpenMP and CUDA offered significant speedups compared to sequential methods, with CUDA outperforming OpenMP, especially for larger datasets.

The study made a notable contribution by providing insights into the practical benefits of parallelizing classical facial recognition algorithms, and it showed the effectiveness of CUDA in reducing execution times for real-time applications.

However, some aspects remain unanswered, particularly regarding the scalability of these methods on much larger and more diverse datasets (e.g., CelebA). Additionally, while parallelization improved performance, further work is needed to address memory bandwidth limitations and to integrate advanced machine learning techniques for handling complex real-world variations in facial recognition tasks.