Performance Aware PCA Acceleration for Healthcare Data Analytics

Lesia Mochurad[1,*,†], Roman Triska[1,†] and Oksana Mulesa[2,3†]

1 Department of Artificial Intelligence Systems, Lviv Polytechnic National University, 12 Bandera street, Lviv, 79013, Ukraine
2 Department of Physics, Mathematics and Technologies, Faculty of Humanities and Natural Sciences, University of Prešov, Ul. 17. novembra 1, 080 01 Prešov, Slovakia
3 Uzhhorod National University, Universytetska St 14, Uzhhorod, Ukraine

Abstract

In modern medical decision-support systems, the growing volume and heterogeneity of multimodal data make efficient preprocessing a prerequisite for timely and reliable analytics. Therefore, accelerating dimensionality-reduction techniques while preserving or improving classification quality is essential for scalable and clinically useful healthcare machine learning. Multimodal medical datasets that combine clinical measurements such as UPDRS with acoustic voice features are typically high dimensional, which increases computational cost and can introduce redundancy in downstream machine learning models. The objective of this study is to evaluate how parallel Principal Component Analysis (PCA) implementations on CPU using OpenMP, on GPU using CUDA, and in a hybrid CPU–GPU scheme affect preprocessing time, and to determine how PCA with and without SMOTE augmentation influences classification quality for Parkinson's disease prediction. We use an open Parkinson's Disease Progression dataset that integrates clinical and voice-derived features, perform preprocessing with an 80/20 train–test split, apply SMOTE to the training set, then run PCA and train an XGBoost classifier evaluated using Accuracy, F1, Precision, and Recall. For a large-scale matrix of 200000×200, the hybrid PCA completes in 10.497 s and achieves a 20.6635× speedup relative to the sequential baseline, while the CUDA version reaches 10.0446× speedup, and the hybrid approach reduces end-to-end computation time from more than 78 s to 55 s in the reported configuration. In the classification experiments, the best performance is obtained with SMOTE plus PCA, achieving Accuracy 0.9591, F1 0.9592, Precision 0.9593, and Recall 0.9591, outperforming SMOTE without PCA with Accuracy 0.9529 and the non-augmented settings. Overall, hybrid CPU–GPU PCA improves scalability for high dimensional multimodal medical data while maintaining and in this setup improving predictive performance when combined with SMOTE and XGBoost, enabling practical runtimes for data intensive smart health applications.

Keywords

multimodal medical data, Parkinson's disease, PCA, hybrid CPU-GPU computing, OpenMP, CUDA, SMOTE, XGBoost

1. Introduction

In modern smart health systems [1], [2], [3], [4] decision support increasingly relies on multimodal medical datasets that combine heterogeneous sources such as clinical test results and acoustic voice features. While such fusion can improve diagnostic and prognostic accuracy, it also produces high dimensional feature spaces that amplify redundancy, memory pressure, and computational cost, often exceeding the capabilities of a single workstation and becoming a bottleneck in end to end machine learning pipelines. Dimensionality reduction is therefore not only a modeling choice but also an enabling step for scalable analysis of large medical data streams, and Principal Component Analysis (PCA) remains one of the most widely used approaches due to its ability to preserve the dominant variance structure while compressing the feature space [5], [6].

The relevance of this problem is particularly clear for multimodal learning in medicine. Surveys and taxonomies emphasize that combining modalities can improve robustness and accuracy, yet

[1] *SmartIndustry 2026: 3rd International Conference on Smart Automation & Robotics for Future Industry, March 26–27 , Lviv, Ukraine*
* Corresponding author.
† These authors contributed equally.
lesia.i.mochurad@lpnu.ua (L. Mochurad); roman.m.triska@lpnu.ua (R. Triska); oksana.mulesa@unipo.sk (O. Mulesa)
0000-0002-4957-1512 (L. Mochurad); 0009-0001-6403-1469 (R. Triska); 0000-0002-6117-5846 (O. Mulesa)

they also highlight practical limitations related to scalability as the number of samples and features grows [7], and broader discussions of multimodal systems report consistent gains over unimodal baselines while noting increased complexity of data integration and training workflows [8], [9]. In Parkinson's disease research, non invasive speech based tests have been shown to provide informative markers for monitoring and telemedicine scenarios [10], while recent studies also explore multimodal settings such as combining neuroimaging and electrophysiology features [11]. However, many applied pipelines still treat PCA as a purely technical preprocessing step, without a systematic analysis of how its configuration and computation strategy affect overall runtime and downstream predictive quality in heterogeneous medical data, especially under realistic resource constraints that are typical for applied smart industry and smart health deployments.

A key technical barrier is that PCA becomes expensive at scale. Large scale PCA analyses emphasize the computational burden for high dimensional data and discuss limitations of classical implementations for big data scenarios [12]. Distributed approaches, for example Spark oriented solutions, can accelerate computation but require nontrivial infrastructure and may be impractical for many medical projects that operate under constrained budgets, restricted data governance, or limited access to clusters [13]. As a result, there is strong motivation to study parallel PCA implementations that exploit modern shared memory multicore CPUs and readily available GPUs, aiming to reduce preprocessing latency without sacrificing model quality [14].

This study focuses on Parkinson's disease prediction using the open dataset Unlocking Clues to Parkinson's Disease Progression [15], which includes two complementary modalities, clinical test measurements and voice features. The clinical component leverages established clinical assessment practice, including Unified Parkinson's Disease Rating Scale indicators that reflect motor and non motor symptom severity and are widely used in clinical evaluation [16], while acoustic features can capture early changes that may appear before prominent motor impairment [10]. Such heterogeneity makes the task representative of real multimodal medical analytics, where the combined feature space is large and preprocessing efficiency directly impacts feasibility of repeated training, tuning, and near real time monitoring.

The objective of this work is to evaluate how parallel implementations of PCA on CPU with OpenMP, on GPU with CUDA, and in a hybrid CPU and GPU scheme affect preprocessing time, and how PCA, with and without SMOTE based augmentation, influences classification quality for Parkinson's disease prediction. The classifier is based on gradient boosted decision trees, with XGBoost selected for its strong performance on tabular high dimensional data and its established effectiveness as a gradient boosting implementation [17].

The scientific novelty and main contribution lie in the joint, experimentally grounded assessment of computational scalability and predictive impact in one coherent multimodal medical pipeline. First, the work provides a controlled comparison of sequential PCA and three parallel variants, isolating the computational effect of each implementation and demonstrating that the hybrid approach offers the most favorable scalability for large matrices representative of multimodal medical settings. Second, it links preprocessing design choices to downstream model behavior by evaluating four training configurations, with and without SMOTE [18], [19] and with and without PCA, using Accuracy, F1, Precision, and Recall, and showing that the best reported performance is achieved when SMOTE is combined with PCA, outperforming SMOTE without PCA and non augmented settings. Third, it quantifies end to end efficiency by showing that the hybrid PCA configuration reduces the total computation time from over 78 seconds to about 55 seconds in the reported setup, while also delivering substantial speedup on a large scale benchmark matrix where the hybrid PCA achieves 20.7× speedup over the sequential baseline.

From a practical perspective, the presented results support the development of data intensive smart health applications within the broader smart industry context, where near real time analytics, iterative model updates, and deployment on mixed CPU and GPU platforms are common requirements. By reducing preprocessing latency and maintaining, and in this setup improving, predictive performance, the proposed hybrid PCA strategy contributes to making multimodal Parkinson's disease modeling more computationally tractable for applied environments such as

clinical decision support, telemonitoring, and smart healthcare services that must operate with limited time and compute budgets.

2. Materials and Methods

2.1. Study design and problem formulation

We address a performance-constrained dimensionality reduction problem that arises in multimodal healthcare analytics. After feature fusion, each record is represented by a real-valued vector that combines clinical measurements, including UPDRS-related indicators, and acoustic voice descriptors. Such multimodal fusion is informative for Parkinson's disease prediction [16], [20], but it increases the dimensionality of the input space and makes preprocessing, particularly PCA, a computational bottleneck for repeated model training and time-sensitive decision support [21], [22].

Let the fused dataset be represented by a feature matrix $X \in R^{n \times d}$ and a binary label vector $y \in \{0,1\}^n$, where n is the number of samples and d is the number of multimodal features. The goal of preprocessing is to compute a compact representation $Z \in R^{n \times k}$, where $k \ll d$, using PCA. PCA is adopted because it produces an orthogonal subspace that maximizes retained variance and reduces redundancy in high-dimensional data.

In addition to representation quality, this work explicitly considers computational feasibility on a single node equipped with a multicore CPU and a GPU. Therefore, the methodological task is defined as jointly achieving:

1. variance-preserving compression, selecting k such that the cumulative explained variance satisfies $\rho(k) \geq \tau$ for a predefined threshold τ
2. low preprocessing latency, minimizing the wall-clock time of PCA computation under practical hardware constraints typical of smart-health deployments, without requiring distributed infrastructure

Formally, for each implementation mode $m \in \{CPU(OpenMP), GPU(CUDA), Hybrid\,CPU-GPU\}$, PCA produces a reduced representation $Z^{(m)}$ and incurs preprocessing time $T_{PCA}^{(m)}$. The study compares these modes by analyzing the trade-off between computational cost $T_{PCA}^{(m)}$ and downstream predictive quality obtained after training the same classifier on $Z^{(m)}$.

2.2. Proposed performance-aware hybrid PCA acceleration

PCA is commonly used for variance-preserving compression; however, its end-to-end latency on large multimodal matrices is governed by heterogeneous computational stages whose dominance depends on n, d, and the target hardware. Therefore, accelerating PCA as a single block is suboptimal in practice. The proposed approach is to treat PCA as a structured sequence of kernels with different arithmetic intensity and memory-access patterns, and to apply a performance-aware mapping that assigns each kernel to the computational resource that is best suited for it under single-node constraints. This design is aligned with the main objective of the paper, which is to reduce preprocessing time while preserving the statistical properties of PCA-based compression.

2.3. PCA kernel decomposition and performance model

Given the standardized and centered training matrix X_c computed using training statistics μ, σ, PCA is decomposed into the following stages:

1. Centering: compute feature-wise means $\mu \in R^d$ and obtain X_c
2. Covariance construction: compute $C \in R^{d \times d}$ as

$$C = \frac{1}{n-1} X_c^T X_c. \tag{1}$$

3. Eigen-decomposition: compute eigenpairs (λ_i, υ_i) of C and form

$$W = [\upsilon_1, \ldots, \upsilon_k]. \tag{2}$$

4. Projection: compute the reduced representation

$$Z = X_c W. \tag{3}$$

These stages have different computational profiles. Centering and projection involve regular, data-parallel operations with independent element updates, while covariance construction is a dense accumulation that benefits from high-throughput parallel arithmetic. In contrast, eigen-decomposition, particularly under Jacobi rotations for symmetric matrices, is iterative and control-intensive, which introduces synchronization and dependency constraints.

To formalize the performance objective, the total preprocessing time for a given implementation mode $m \in \{CPU(OpenMP), GPU(CUDA), Hybrid\, CPU-GPU\}$ is written as:

$$T_{PCA}^{(m)} = T_{center}^{(m)} + T_{cov}^{(m)} + T_{eig}^{(m)} + T_{proj}^{(m)} + T_{tr}^{(m)}. \tag{4}$$

Here $T_{tr}^{(m)}$ denotes host–device transfer overheads that are present for GPU-involving modes. This decomposition provides a principled basis to design and compare implementations under realistic single-node constraints, rather than reporting only an aggregate runtime.

The proposed contribution is a performance-aware PCA acceleration methodology that exposes PCA as a sum of measurable kernel times, and uses this structure to design a hybrid CPU–GPU execution strategy that minimizes end-to-end preprocessing latency while keeping the statistical PCA mapping unchanged.

2.4. Step-by-step description of the proposed method

The proposed method consists of the following steps.

Step 1. Centering using training statistics.

Let $\mu \in R^d$ and $\sigma \in R^d$ be feature-wise mean and standard deviation computed on the training subset. Each feature is standardized using training statistics to prevent scale dominance in multimodal fusion:

$$x_{ij}^{'} = \frac{x_{ij} - \mu_j}{\sigma_j + \delta}, \ \mu_j = \frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} x_{ij}, \tag{5}$$

where δ is a small constant for numerical stability. The same μ and σ are applied to the test subset to avoid information leakage.

We denote by X_c the standardized training matrix with entries $x_{ij}^{'}$. The same transformation is applied to the test subset using the training statistics μ,σ.

Step 2. Covariance matrix construction.

Compute:

$$C = \frac{1}{n_{tr} - 1} X_c^T X_c. \tag{6}$$

This operation dominates when n_{tr} is large and d is moderate, which is common in fused multimodal analytics. It is well-suited to parallel execution due to independent computation of C_{pq} elements and high arithmetic intensity.

Step 3. Eigen-decomposition via Jacobi rotations.

Compute eigenpairs of the symmetric matrix C using the Jacobi method. At iteration t, a pivot pair (p, q) is selected, and a plane rotation is applied to reduce C_{pq}. Formally, the update can be written as:

$$C^{(t+1)} = R_{pq}^T C^{(t)} R_{pq}, \; V^{(t+1)} = V^{(t)} R_{pq}, \tag{7}$$

where R_{pq} is an orthogonal rotation acting in the (p, q) plane. The iterations continue until the maximum off-diagonal magnitude becomes smaller than a tolerance ε. Jacobi is selected because it is applicable to symmetric covariance matrices and produces an orthogonal eigenvector basis required by PCA projection.

Since Jacobi eigen-decomposition is iterative and requires repeated pivot selection and synchronization across updates, its efficiency on GPUs may be limited for moderate d due to control overhead and iteration-level dependencies. Therefore, in the hybrid organization used in this study, the covariance matrix C is first constructed on the GPU, then transferred to the host, and the Jacobi eigen-decomposition is performed on the multicore CPU using OpenMP. This mapping preserves GPU acceleration for the throughput-dominant covariance construction while ensuring efficient coordination of pivot selection and convergence control on the CPU for the considered dimensionality regime.

Step 4. Component selection by explained variance.

The eigenvalues are sorted in descending order and the first k components are selected such that:

$$\rho(k) = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i} \geq \tau, \tag{8}$$

where τ is a predefined explained-variance threshold. This ensures that dimensionality reduction preserves the dominant variance structure, which is the central statistical requirement for PCA-based compression.

Step 5. Projection to the reduced subspace.

Compute:

$$Z = X_c W, \; W = [v_1, \ldots, v_k]. \tag{9}$$

This stage is again highly data-parallel and can be efficiently executed on either CPU or GPU depending on the mode m.

2.5. Implementation modes and implementation details

To quantify the computational trade-offs under a consistent PCA formulation, three implementation modes are developed and compared. Each mode follows the same mathematical PCA procedure described in Sections 2.2–2.4, while differing only in how the computational kernels are mapped to hardware resources and parallelized.

Mode 1: CPU (OpenMP).

In the CPU (OpenMP) mode, centering, covariance construction, Jacobi eigen-decomposition, and projection are parallelized using shared-memory multithreading. Centering is parallelized over matrix elements (i, j) after feature-wise mean computation. Covariance construction is parallelized over feature pairs (p, q) while accumulating over samples i; symmetry $C_{pq}=C_{qp}$ is exploited by computing only one triangular part and mirroring the result to reduce redundant work.

Jacobi eigen-decomposition is implemented as an iterative loop. At each iteration, the pivot indices (p, q) corresponding to the maximum off-diagonal magnitude are identified. The pivot search is parallelized by computing thread-local maxima over disjoint matrix blocks followed by a synchronized global selection. After the pivot is fixed, the Jacobi rotation update is applied by parallelizing the independent row and column updates over the index m. The eigenvector matrix is updated in the same way, ensuring that the resulting eigenbasis remains orthogonal. The iterations terminate when the maximum off-diagonal magnitude falls below the tolerance ε. Projection

$$Z = X_c W \tag{10}$$

is computed by parallelizing independent output elements (i, k).

Mode 2: GPU (CUDA).

In the GPU (CUDA) mode, the PCA stages are mapped to CUDA kernels to exploit massive data parallelism. Mean computation is performed via parallel reduction per feature, followed by a centering kernel that updates elements (i, j) independently. Covariance construction is implemented as a kernel that computes entries C_{pq} using dot-product style accumulation over the sample dimension. Projection is executed as a matrix multiplication-like kernel mapping threads to output elements z_{ik}.

Jacobi update steps are also implemented on the GPU for the CUDA-only PCA variant by applying rotation-update kernels. Practical efficiency depends on the relative size of d and on the overhead of coordinating pivot selection and convergence checks across iterations, because the eigen-decomposition stage is iterative and control-intensive compared to throughput-dominant kernels.

Mode 3: Hybrid CPU–GPU (proposed).

The hybrid mode assigns the throughput-dominant covariance construction to the GPU, then transfers the resulting covariance matrix C to the host. Jacobi eigen-decomposition and component selection are performed on the multicore CPU using OpenMP, and the projection $Z = X_c W$ is also computed on the CPU to avoid additional host–device transfers of eigenvectors and to preserve a consistent single-node workflow.

This mapping preserves GPU acceleration for dense accumulation while keeping the iterative, control-intensive Jacobi procedure on the CPU, which is efficient for the considered dimensionality regime. This hybrid mapping is the core of the proposed approach and is designed to minimize $T_{PCA}^{(m)}$ under single-node constraints by combining GPU acceleration for arithmetic-intensive stages with CPU coordination for iterative control.

2.6. Computational complexity and expected performance regimes

The PCA preprocessing stages have different asymptotic costs with respect to the number of samples n and features d. Centering requires $O(nd)$ operations. Covariance construction via

$$C = \frac{1}{n-1} X_c^T X_c \tag{11}$$

requires $O(nd^2)$ operations and becomes dominant when n is large and d is moderate, which is typical for fused multimodal datasets. Eigen-decomposition of a $d \times d$ covariance matrix is

typically $O(d^3)$ in terms of arithmetic updates, while the number of Jacobi iterations depends on the tolerance ε and the off-diagonal structure of C. The projection stage requires $O(ndk)$ operations, where $k \ll d$.

These regimes motivate the proposed performance-aware mapping. When n is large, covariance construction and projection are throughput-dominant and benefit from GPU execution. When d is moderate, the iterative nature of Jacobi eigen-decomposition makes the CPU competitive due to lower control overhead and efficient shared-memory synchronization. The hybrid method therefore targets the regime where covariance dominates and should be accelerated on GPU, while eigen-decomposition remains efficient on multicore CPU with OpenMP.

To quantify acceleration, the preprocessing speedup for mode m is defined as:

$$S^{(m)} = \frac{T_{PCA}^{(seq)}}{T_{PCA}^{(m)}}, \tag{12}$$

where $T_{PCA}^{(seq)}$ is the sequential baseline runtime under the same PCA formulation and component-selection policy. This definition ensures that the comparison reflects true algorithmic acceleration rather than changes in the statistical mapping.

3. Modeling and numerical experiments

3.1. Dataset description

Experiments are conducted using the open dataset Unlocking Clues to Parkinson's Disease Progression. The dataset contains 5875 records collected from 42 subjects and integrates heterogeneous predictors that combine clinical assessments and acoustic voice descriptors. The feature set includes subject identifier, demographic attributes, recording time index, clinical UPDRS-related measurements, and a set of voice micro-variation indicators such as jitter and shimmer families, as well as NHR, HNR, RPDE, DFA, and PPE. This composition represents a multimodal tabular setting where feature fusion increases dimensionality and makes preprocessing cost relevant for repeated modelling.

To examine redundancy in the fused feature space, we analyze pairwise correlations among the main clinical and acoustic variables. The correlation heatmap is shown in Figure 1, illustrating groups of strongly correlated voice micro-variation descriptors.

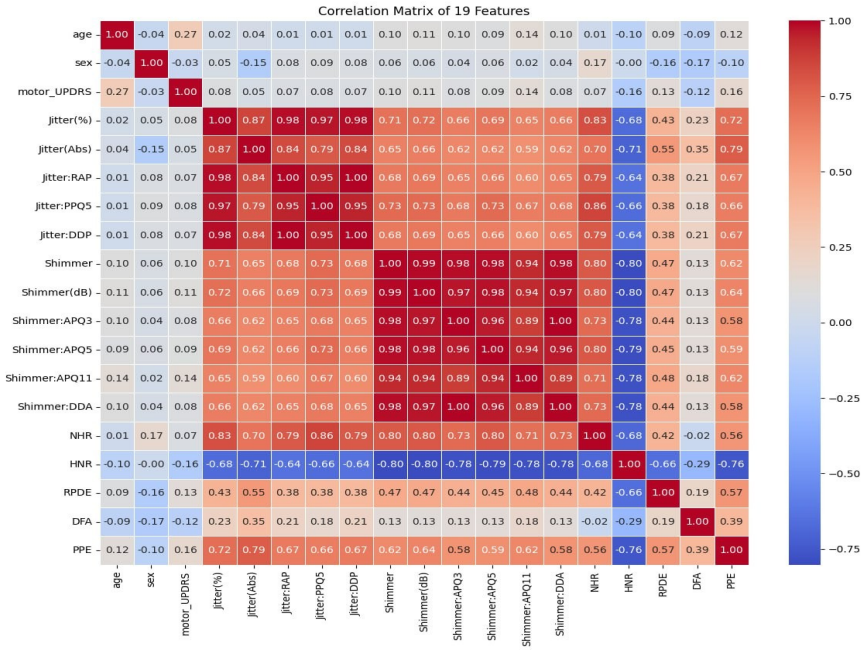

Figure 1: Heatmap of the correlation matrix for clinical and acoustic features.

Strong within-group correlations are visible among jitter-related and shimmer-related features, indicating substantial redundancy and potential multicollinearity. Such structure can increase model variance and training time, and it provides a direct motivation for applying PCA to transform the correlated features into a smaller set of uncorrelated components that preserve the dominant variance directions.

3.2. Experimental setup and preprocessing pipeline

The classification pipeline follows a fixed train–test split of 80 percent for training and 20 percent for testing. To mitigate class imbalance observed in the target distribution, SMOTE augmentation is applied only to the training subset. Class imbalance is observed in the target labels, which can bias a classifier toward the majority class and reduce sensitivity to the minority class. The label distribution is presented in Figure 2, motivating the use of SMOTE on the training subset to improve class balance.

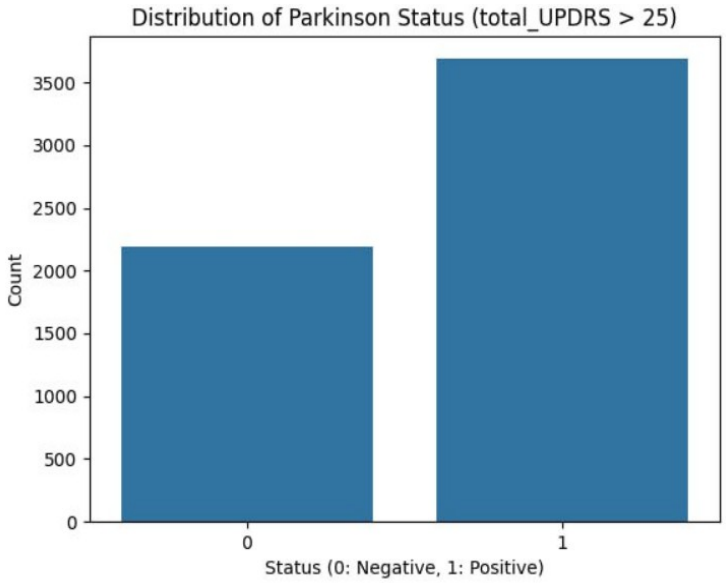

Figure 2: Distribution of Parkinson's disease status in the dataset

Figure 2 confirms that the classes are unevenly represented. Applying SMOTE only to the training data reduces imbalance by generating synthetic minority samples via interpolation between nearest neighbors, which typically improves the stability of decision boundaries without duplicating original records.

PCA is then performed on the training data and the learned transformation is applied to the test data. The number of retained principal components is selected by an explained-variance criterion with threshold $\tau = 0.95$; in the reported configuration this corresponds to retaining 10 components.

The selection of the reduced dimension k is guided by the cumulative explained-variance curve, which shows how much of the original variance is preserved as additional principal components are retained. As shown in Figure 3, the curve increases rapidly for the first components and then saturates, indicating diminishing returns from adding further components.

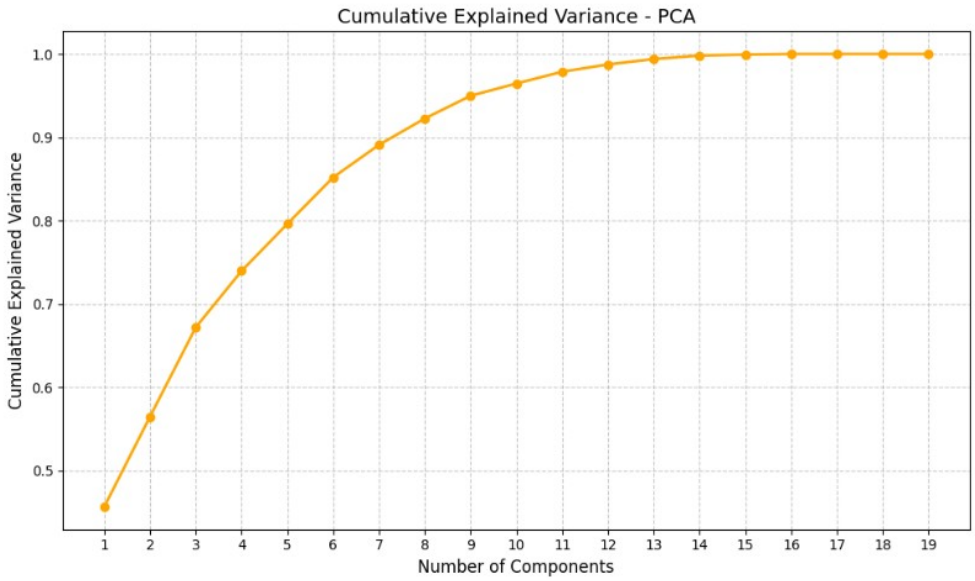

Figure 3: Cumulative explained variance after PCA.

The curve indicates that the first 10 components preserve more than 95 percent of the total variance, which justifies using k=10 as a compact representation that retains the dominant information content while reducing redundancy and computational cost.

The reduced representations are used to train an XGBoost classifier. Predictive performance is evaluated using Accuracy, F1, Precision, and Recall [23], [24], [25].

All performance experiments are executed on a single node equipped with an AMD Ryzen 5 6600H CPU and an NVIDIA GeForce RTX 3050 GPU with 16 GB DDR5 system memory.

3.3. PCA acceleration on synthetic matrices

To evaluate scalability of PCA implementations under controlled matrix sizes, PCA runtime is measured for matrices of sizes 300×300, 500×500, 1000×1000, and 200000×200 for the following implementations: sequential, OpenMP with 2, 6, and 12 threads, CUDA, and the hybrid CPU–GPU approach (see Table 1).

Table 1

Measured PCA runtime (seconds)

Size	Sequential	OpenMP 2	OpenMP 6	OpenMP 12	CUDA	Hybrid
300×300	42.298	36.420	34.930	32.540	70.041	36.599
500×500	307.784	241.583	149.324	121.811	318.076	89.585
1000×1000	1307.149	977.362	490.999	325.852	705.873	281.942
200000×200	216.905	128.648	56.410	47.435	21.594	10.497

The results in Table 1 show that the benefit of GPU and hybrid execution depends strongly on the relationship between the number of samples and the number of features. Increasing the feature dimension makes eigen-decomposition substantially more expensive, which is clearly visible for 1000×1000. For the large tall matrix 200000×200, the hybrid approach achieves the lowest runtime, completing PCA in 10.497 s.

3.4. End-to-end impact on training time

In addition to standalone PCA timing, the full pipeline runtime is measured to quantify how PCA affects overall computation time due to dimensionality reduction before model training (see Table 2).

Table 2

Total runtime of modelling with and without PCA (seconds)

Configuration	Model training time	PCA time	Total time
Without PCA	78.151	-	78.151
With sequential PCA	55.816	8.011	63.827
With hybrid PCA	55.083	0.550	55.633

Applying PCA reduces model training time substantially by decreasing feature dimensionality. The hybrid PCA implementation further reduces preprocessing overhead, producing the lowest end-to-end runtime of 55.633 s.

3.5. Classification quality with PCA and SMOTE

To qualitatively assess how dimensionality reduction organizes the fused multimodal feature space, we visualize the samples in the space of the first two principal components. The two-dimensional PCA projection is shown in Figure 4.

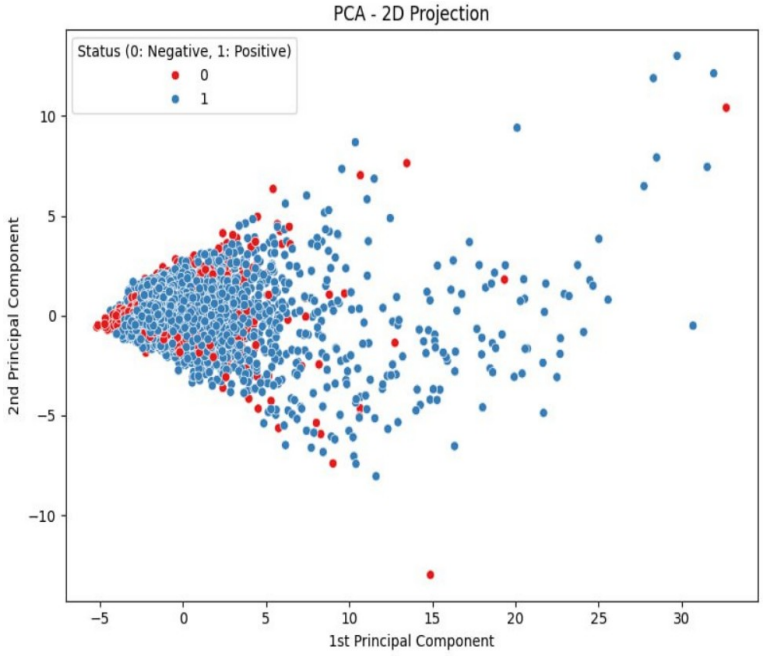

Figure 4: Projection of samples onto the first two principal components.

The projection suggests partial separation tendencies between the two classes, indicating that the leading components capture discriminative structure in the data. Although complete separation is not expected for real clinical data, the observed clustering pattern supports PCA as a meaningful compression step prior to classification.

A three-dimensional visualization provides additional insight into the geometric structure of the reduced space by adding the third component. The 3D PCA projection is presented in Figure 5.

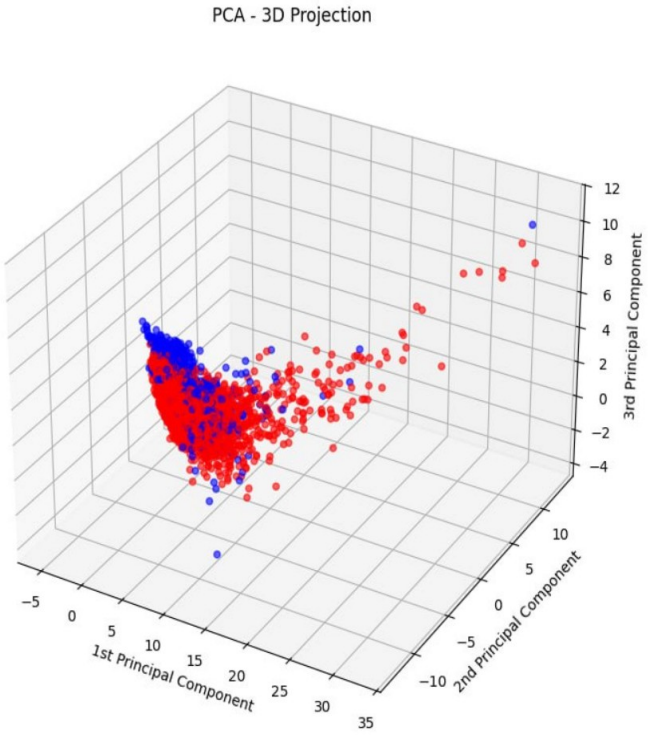

Figure 5: 3D projection onto the first three principal components.

The additional component reveals a richer spatial organization of samples and can highlight separation trends that are not visible in 2D. While overlap remains, the 3D view indicates that PCA preserves meaningful structure of the multimodal data in a compact representation, which is consistent with the observed improvements in classification metrics.

Predictive quality is evaluated for four configurations: without augmentation and without PCA, without augmentation with PCA, SMOTE without PCA, and SMOTE with PCA. All results are presented in Table 3.

Table 3

Classification metrics

Configuration	Accuracy	F1	Precision	Recall
Without augmentation, without PCA	0.9467	0.9469	0.9474	0.9467
Without augmentation, with PCA	0.9543	0.9544	0.9547	0.9543
SMOTE, without PCA	0.9529	0.9529	0.9530	0.9529
SMOTE with PCA	0.9591	0.9592	0.9593	0.9591

As shown in Table 3, the best result is obtained when SMOTE is combined with PCA. PCA also improves results without augmentation, indicating that variance-preserving compression can reduce redundancy and improve generalization in the considered configuration.

The obtained experimental results are further interpreted in the Discussion section, with emphasis on scalability regimes and the interaction between dimensionality reduction and class balancing.

4. Discussion

The experiments confirm that PCA runtime is not determined by a single operation but by stages with different computational characteristics, and that the dominant cost depends on matrix shape.

When the feature dimension increases, eigen-decomposition becomes the main bottleneck, which explains the sharp increase in runtime for square matrices such as 1000×1000 across all implementations. For tall matrices with a moderate number of features, GPU acceleration and hybrid execution become more advantageous, and the hybrid approach provides the best runtime for 200000×200.

From an end-to-end perspective, PCA reduces training time due to dimensionality reduction, so the total pipeline can become faster even after adding the PCA step. This effect is most pronounced for the hybrid implementation, where PCA overhead is reduced to 0.550 s and the total runtime reaches 55.633 s compared to 78.151 s without PCA.

On the modelling side, combining SMOTE with PCA yields the highest classification scores. This suggests that class balancing on the training set and variance-preserving compression act synergistically: SMOTE mitigates imbalance, while PCA reduces feature redundancy and stabilizes the classifier under high-dimensional multimodal fusion.

Future research should extend the proposed performance-aware PCA framework to larger and more heterogeneous neurological datasets in order to assess its robustness across different sample-to-feature ratios, noise levels, and missing-data conditions. From a computational perspective, additional improvements may be achieved by reducing host-device transfer overheads, refining the performance-aware mapping of PCA stages, and exploring randomized or incremental PCA variants for larger-scale or streaming scenarios. Another important direction is the integration of the proposed pipeline into real-time clinical decision-support and telemonitoring systems, where low-latency preprocessing and stable predictive performance are both essential.

5. Conclusions

A performance-aware PCA evaluation was conducted for multimodal healthcare analytics under single-node constraints. The hybrid CPU–GPU strategy used in this work accelerates PCA by executing covariance construction on the GPU and performing Jacobi eigen-decomposition and projection on the CPU with OpenMP. On a large matrix of 200000×200, the hybrid implementation achieves the lowest PCA time of 10.497 s. In the full modelling pipeline, hybrid PCA reduces total runtime to 55.633 s compared to 78.151 s without PCA. In classification experiments, the best predictive performance is achieved with SMOTE combined with PCA, reaching Accuracy 0.9591 and consistent improvements in F1, Precision, and Recall.

Future work will extend the proposed performance-aware PCA acceleration beyond the current single-dataset and single-platform setting by validating it on additional multimodal medical datasets with different sample-to-feature ratios, noise levels, and missingness patterns, and by performing sensitivity analysis of the explained-variance threshold τ and the resulting number of retained components k to quantify the trade-off between compression strength, runtime, and predictive quality. Further studies will examine the interaction between class balancing and dimensionality reduction by comparing SMOTE with alternative imbalance-handling strategies and by analyzing how oversampling affects the covariance structure and stability of principal components. From a systems perspective, future research will focus on reducing host–device transfer overheads, refining profiling-based performance models, and exploring alternative eigensolvers as well as randomized or incremental PCA variants for larger-scale regimes with higher d or streaming-like portion-based updates, which are important for practical smart-health deployments under latency constraints.

Declaration on Generative AI

The author during the preparation of this work, used AI tools to check text spelling. After using these tools, the author reviewed and edited the content as needed and take full responsibility for the publication's this content.

References

[1] V. Shkarupylo, J. A. J. Alsayaydeh, M. F. B. Yusof, A. Oliinyk, V. Artemchuk, and S. G. Herawan. "Exploring the Potential Network Vulnerabilities in the Smart Manufacturing Process of Industry 5.0 via the Use of Machine Learning Methods." *IEEE Access* 12 (2024): 152262–152276. doi:10.1109/ACCESS.2024.3474861.

[2] I. Izonin, R. Tkachenko, Z. Duriagina, N. Shakhovska, V. Kovtun, and N. Lotoshynska. "Smart Web Service of Ti-Based Alloy's Quality Evaluation for Medical Implants Manufacturing." *Applied Sciences* 12.10 (2022): 5238. doi:10.3390/app12105238.

[3] A. Almarri, Z. Hunaiti, and N. Manivannan. "A Review of Smart Healthcare: Concept, Drivers, Characteristics, and Challenges." *Hospitals* 2.4 (2025): 26. doi:10.3390/hospitals2040026.

[4] A. B. Kaminskyi, M. V. Nehrey, and L. M. Zomchak. "COVID-19: Crisis or New Opportunities Time for the Agricultural Sector of Ukraine." *IOP Conference Series: Earth and Environmental Science* 628.1 (2021): 012031. doi:10.1088/1755-1315/628/1/012031.

[5] I. T. Jolliffe and J. Cadima. "Principal Component Analysis: A Review and Recent Developments." *Philosophical Transactions of the Royal Society A* 374.2065 (2016): 20150202. doi:10.1098/rsta.2015.0202.

[6] L. Mochurad, V. Babii, Y. Boliubash, and Y. Mochurad. "Improving Stroke Risk Prediction by Integrating XGBoost, Optimized Principal Component Analysis, and Explainable Artificial Intelligence." *BMC Medical Informatics and Decision Making* 25.1 (2025): 63. doi:10.1186/s12911-025-02894-z.

[7] T. Baltrušaitis, C. Ahuja, and L.-P. Morency. "Multimodal Machine Learning: A Survey and Taxonomy." *arXiv* (2017). doi:10.48550/arXiv.1705.09406.

[8] N. Srivastava and R. Salakhutdinov. "Multimodal Learning with Deep Boltzmann Machines." *Journal of Machine Learning Research* 15 (2014): 2949–2980.

[9] A. Benani et al. "Is Multimodal Better? A Systematic Review of Multimodal versus Unimodal Machine Learning in Clinical Decision-Making." *medRxiv* (2025). doi:10.1101/2025.03.12.25322656.

[10] A. Tsanas, M. Little, P. McSharry, and L. Ramig. "Accurate Telemonitoring of Parkinson's Disease Progression by Non-Invasive Speech Tests." *Nature Precedings* (2009). doi:10.1038/npre.2009.3920.1.

[11] M. Alrawis, S. Al-Ahmadi, and F. Mohammad. "Bridging Modalities: A Multimodal Machine Learning Approach for Parkinson's Disease Diagnosis Using EEG and MRI Data." *Applied Sciences* 14.9 (2024): 3883. doi:10.3390/app14093883.

[12] T. Raiko, A. Ilin, and J. Karhunen. "Principal Component Analysis for Large Scale Problems with Lots of Missing Values." In *Machine Learning: ECML 2007*, Lecture Notes in Computer Science, vol. 4701, J. N. Kok, J. Koronacki, R. L. D. Mantaras, S. Matwin, D. Mladenič, and A. Skowron (Eds.), Springer, Berlin, Heidelberg, 2007, pp. 691–698. doi:10.1007/978-3-540-74958-5_69.

[13] S. Hong et al. "Performance Evaluation of Apache Spark According to the Number of Nodes using Principal Component Analysis." In *Proceedings of the 2015 International Conference on Big Data Applications and Services*, ACM, Jeju Island, Republic of Korea, 2015, pp. 98–103. doi:10.1145/2837060.2837074.

[14] L. Mochurad and A. Solomiia. "Optimizing the Computational Modeling of Modern Electronic Optical Systems." In *Lecture Notes in Computational Intelligence and Decision Making*, vol. 1020, V. Lytvynenko, S. Babichev, W. Wójcik, O. Vynokurova, S. Vyshemyrskaya, and S. Radetskaya (Eds.), Springer, Cham, 2020, pp. 597–608. doi:10.1007/978-3-030-26474-1_41.

[15] The Devastator. "Unlocking Clues to Parkinson's Disease Progression." Kaggle (2023). Available: https://www.kaggle.com/datasets/thedevastator/unlocking-clues-to-parkinson-s-disease-progressi (accessed 13 February 2026).

[16] T. Yamamoto, Y. Yamanaka, S. Hirano, Y. Higuchi, and S. Kuwabara. "Utility of Movement Disorder Society-Unified Parkinson's Disease Rating Scale for Evaluating Effect of Subthalamic

Nucleus Deep Brain Stimulation." *Frontiers in Neurology* 13 (2023): 1042033. doi:10.3389/fneur.2022.1042033.

[17] A. B. Siddique, A. M. M. Alvee, P. Das Ana, L. N. Jahan, and M. Hashan. "Tree-Based, Boosting, and Stacked Models for Accurate Prediction of Total Organic Carbon from Conventional Well Logs." *Energy Reports* 13 (2025): 4491–4513. doi:10.1016/j.egyr.2025.04.021.

[18] S. Saif, P. Das, S. Biswas, S. Khan, M. A. Haq, and V. Kovtun. "A Secure Data Transmission Framework for IoT Enabled Healthcare." *Heliyon* 10.16 (2024): e36269. doi:10.1016/j.heliyon.2024.e36269.

[19] D. Elreedy, A. F. Atiya, and F. Kamalov. "A Theoretical Distribution Analysis of Synthetic Minority Oversampling Technique (SMOTE) for Imbalanced Learning." *Machine Learning* 113.7 (2024): 4903–4923. doi:10.1007/s10994-022-06296-4.

[20] I. Gupta, V. Sharma, S. Kaur, and A. K. Singh. "PCA-RF: An Efficient Parkinson's Disease Prediction Model based on Random Forest Classification." *arXiv* (2022). doi:10.48550/arXiv.2203.11287.

[21] L. Mochurad, N. Shakhovska, J. A. J. Alsayaydeh, and M. F. Yusof. "Parallel and GPU-Based Optimization of XGBoost and Neural Networks for Effective Landmine Classification." *IJSSE* 15.3 (2025): 563–572. doi:10.18280/ijsse.150315.

[22] V. Shkarupylo, I. Blinov, A. Chemeris, V. Dusheba, J. A. J. Alsayaydeh, and A. Oliinyk. "Iterative Approach to TLC Model Checker Application." In *2021 IEEE 2nd KhPI Week on Advanced Technology (KhPIWeek)*, IEEE, Kharkiv, Ukraine, 2021, pp. 283–287. doi:10.1109/KhPIWeek53812.2021.9570055.

[23] I. Izonin, R. Muzyka, R. Tkachenko, I. Dronyuk, K. Yemets, and S.-A. Mitoulis. "A Method for Reducing Training Time of ML-Based Cascade Scheme for Large-Volume Data Analysis." *Sensors* 24.15 (2024): 4762. doi:10.3390/s24154762.

[24] S. Liaskovska, S. Tyskyi, Y. Martyn, A. T. Augousti, and V. Kulyk. "Systematic Generation and Evaluation of Synthetic Production Data for Industry 5.0 Optimization." *Technologies* 13.2 (2025): 84. doi:10.3390/technologies13020084.

[25] S. Liaskovska, O. Gumen, Y. Martyn, and V. Zhelykh. "Investigation of Microclimate Parameters in the Industrial Environments." In *Lecture Notes on Data Engineering and Communications Technologies*, Springer, 2023. doi:10.1007/978-3-031-36115-9_41.