Methods for Temporal AI Classification of Pianist Fine-Motor Skill

Stanford CS229 Project

Max Rodriguez
Department of Computer Science
Stanford University
maxrod@stanford.edu

***Abstract*—Classifying a pianist's technical proficiency is a complex task, even for experts. This paper presents methods for extracting and classifying subtle hand motor movement features. My research demonstrates that machine learning classifiers can effectively differentiate between beginner and advanced pianists. Furthermore, I propose an enhanced approach that improves proficiency level prediction by first performing unsupervised activity classification, followed by supervised technique level classification.**

I. Introduction

Developing professional-level technique in classical piano requires consistent instruction, but private lessons are often prohibitively expensive. To address this barrier, I propose the development of an AI-driven pianist technique assistant that provides feedback to guide pianists toward better technique.

A foundational step in this effort is to evaluate whether machine learning (ML) techniques can successfully differentiate between the fine motor movements of beginner and advanced pianists. This involves identifying and analyzing subtle differences in hand movements that distinguish varying skill levels.

The classification process begins with video recordings of pianists captured from a bird's eye view. Video frames are converted into image sequences, and hand landmarks are extracted as keypoints. To capture the temporal aspects of piano technique, the sequential images are processed using various pre-processing methods before being fed into classifiers.

Unlike static images, sequential processing allows analysis of movement patterns over time. The key differences between beginner and advanced pianists often lie in the consistency and control of their motion across an entire action sequence rather than in isolated hand positions. Once preprocessed using a range of methods to prevent overfitting and balance the variance-bias tradeoff, data is passed through one of three classifiers: Multilayer Perceptron (MLP), AdaBoost, or Gaussian Discriminant Analysis (GDA).

As the dataset includes pianists performing a variety of different movements (e.g., scales, arpeggios, ascending and descending passages, chord progressions), I experimented with using initial unsupervised activity classification prior to training and classification. This step ensures that technique comparisons occur within the same movement category—for example, a pianist playing a scale is only compared against other pianists playing scales, rather than against those playing arpeggios, as differences in fine motor movements between activities could mislead the model.

II. Related Work

Research on temporal activity classification, skill proficiency estimation, and piano technique assessment provides valuable insight into the feasibility of classifying pianist proficiency using machine learning techniques.

Temporal Activity Classification: Niebles et al. [1] proposed a method that automatically discovers combinations of activity motion segments, resulting in a simple yet robust temporal classifier. The approach effectively classifies complex Olympic activities by identifying structured motion patterns within discrete action categories (e.g., sports activities). However, their method omits intermediate processing steps—such as leveraging Hidden Markov Models (HMMs) or Hidden Conditional Random Fields (HCRFs) to explicitly discover meaningful motion segments through hidden states—which limits its applicability to fine-motor classification. In contrast to sports, where motion sequences can be mapped to predefined activities, pianist technique involves highly variable and nuanced motor patterns that do not conform to a finite set of discrete actions. The subtle differences between beginner and advanced pianists emerge from continuous variations in execution over time, motivating the use of HMM-based temporal modeling in this study.

General Proficiency/Skill Level Classification: A broader body of work has explored automated skill evaluation. Connolly et al. [2] investigated skill-estimation in trampoline performance. Their approach, which employs pose estimation and optical flow to track velocity, displacement, and angular rotation, achieved an 80.7% accuracy in classifying skill across 20 distinct movement categories. Although effective for assessing gross motor skills, their methodology relies on discrete performance metrics (i.e., flight time and displacement) that are less applicable to piano technique, where wrist height and finger curvature vary significantly even among experts. Additionally, Law et al. [3] applied computer vision-based skill

assessment to surgical performance using stacked hourglass networks for pose estimation. Their method achieved 83.33% accuracy in classifying surgical skill.

Piano-Specific Proficiency Classification: Lee et al. [4] explored automated piano pedagogy by tracking finger placement on keys using computer vision. Their method detects whether the correct note is pressed by an expected finger, offering an objective measure of accuracy. A limitation of this approach is that it treats piano technique as a static image problem rather than a temporal classifcation task. In addition, advanced pianists adjust their finger choices based on interpretative preferences, limiting the effectiveness of static analyses. Johnson et al. [5] proposed a depth-based method for detecting pianist hand posture, using random decision forests to classify hand positioning from depth maps. Their work identifies poor technique indicators such as low wrists and flat hands. While motivating the incorporation of depth-sensitive features in this study, posture-based classification alone is insufficient for proficiency assessment, as advanced techniques sometimes require non-standard hand positioning (e.g., flattened hands for large span chords).

III. Dataset and Features

The original dataset consists of pianist hand movement sequences recorded from a bird's-eye view, capturing both hands and wrists. Data were collected from in-person performances as well as YouTube videos, ensuring a diverse dataset in terms of video quality and pianist proficiency levels.

Data Collection

The dataset comprises recordings from 145 unique advanced pianists, from whom 434 20-frame playing sequences were extracted. In addition, data were collected from 70 beginner pianists, yielding 188 20-frame playing sequences. The discrepancy in data-volume between skill levels was intentional, as the original methodology relied on Hidden Markov Models (HMMs) for probability-based sequence pre-processing—requiring 80% of the advanced pianist data for HMM training. As the study evolved to include alternative pre-processing and feature extraction techniques, a more balanced dataset would have been ideal.

A separate holdout dataset was collected using the same methodology, consisting of recordings from 10 new advanced pianists and 10 new beginner pianists, from which 60 sequences per proficiency level were extracted.

Data Preprocessing

Each 20-frame sequence was constructed with a time interval of $t = 4$ frames, ensuring that motion between frames was noticeable while avoiding excessive displacement. All videos were standardized in resolution and converted to grayscale prior to feature extraction.

Key hand landmarks were extracted using Google's MediaPipe Hands library. For each hand in every frame, 21 landmarks were extracted, providing x, y, and z coordinates. The right and left hands were treated as separate sequences for consistency.

Hand Orientation Normalization

To align all hand sequences consistently with the vertical y-axis, a rotation normalization was applied using the following explicit equations:

Let W and M denote the positions of the wrist (landmark 0) and the middle metacarpal (landmark 9), respectively. The hand direction vector is $D = M - W$, and the target direction is $T = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$. The rotation axis is computed as $R = D \times T$, and the rotation angle is:

$$\theta = \arccos\left(\frac{D \cdot T}{\|D\| \, \|T\|}\right).$$

Using Rodrigues' rotation formula, the rotation matrix is:

$$R_{\text{matrix}} = I + \sin\theta \, K + (1 - \cos\theta) K^2,$$

where I is the identity matrix and K is the skew-symmetric matrix of the normalized rotation axis $R/\|R\|$. Each landmark p is rotated as $p' = R_{\text{matrix}} p$, and the left-hand sequence is mirrored across the x-axis:

$$p'' = (-x, y, z).$$

Missing landmark values were interpolated using data from adjacent frames, and sequences with excessive missing values were discarded.

Feature Representation and Extraction

After preprocessing, each 20-frame sequence is represented as a 20×63 matrix (20 frames, 21 landmarks, and 3 coordinate values per landmark). Three feature extraction methods were employed:

1) **Flattening (Baseline):** Each matrix was flattened into a 1D vector of 1260 features ($20 \times 21 \times 3$).
2) **Convolutional Feature Reduction:** A 3×3 convolutional kernel with stride $(2, 2)$ (without padding) was applied, yielding a reduced 1D vector of 270 features.
3) **Hidden Markov Model (HMM) Probabilities:** HMMs were used to generate probability vectors, producing a 1D feature vector with 21 elements per sequence.

Feature Compression Based on Data Subspace

Principal Component Analysis (PCA): To reduce feature dimensionality, PCA was applied to the extracted 1D vectors. First, each feature was standardized:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad \sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}, \qquad X_{\text{normalized}} = \frac{X - \mu}{\sigma}. \tag{1}$$

The covariance matrix was computed as:

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T.$$

The top 10 eigenvectors of Σ were selected to project the data into a lower-dimensional space.

Linear Discriminant Analysis (LDA): LDA was applied to the original 1D feature vectors (for both training and holdout datasets) to reduce dimensionality while maximizing class separability. LDA computes the between-class scatter matrix S_b and the within-class scatter matrix S_w:

$$S_b = \sum_{i=1}^{c} N_i(\mu_i - \mu)(\mu_i - \mu)^T,$$
$$S_w = \sum_{i=1}^{c} \sum_{x \in C_i} (x - \mu_i)(x - \mu_i)^T, \quad (2)$$

where N_i is the number of samples in class i, μ_i is the mean of class i, and μ is the overall mean. LDA was explored to assess whether the most discriminative directions in the feature subspace could improve classification performance [13].

IV. Methods

Hidden Markov Models for Temporal Feature Extraction

Pianist hand movements are modeled as state transitions in a Markov process, where the probability of transitioning from one hand position to another depends solely on the current state. By identifying the most likely hand positions (states) of advanced pianists over many hours of playing and calculating transition probabilities between those states, one-dimensional probability vectors can be generated for each key landmark sequence. This approach enables us to estimate the likelihood that a given sequence was executed by an advanced pianist. One of the primary challenges in this process is that there are many potential hand positions, and without detailed analysis by expert musicians, the relevant states in complex movement sequences can remain hidden.

To capture the intricate transitions between individual landmarks, the original hand key landmark dataset was divided into 21 separate sets, each containing the sequences for a single landmark. Each set was then assigned its own Hidden Markov Model (HMM). This strategy allows us to model the transitions for each landmark independently rather than relying on a single model for the entire hand.

The HMMs operate under the assumption that the probability of an observed key landmark state O_t at time t depends only on the current hidden state S_t and not on any previous states or observations:

$$P(O_t \mid S_t) \quad \text{depends only on } S_t.$$

Interpretation: In the context of pianist data, this means that the probability of the observed hand key landmark at any given moment is determined solely by the underlying hand position (hidden state) at that time, regardless of past movement frames.

To compute the likelihood that an input sequence belongs to an advanced pianist, the Forward Algorithm for HMMs [6] is employed. The algorithm is defined as follows:

Initialization:

$$\alpha_1(i) = \pi_i \, b_i(O_1), \quad 1 \le i \le N,$$

where π_i is the initial probability of state i and $b_i(O_1)$ is the probability of observing O_1 in state i.

Recursion:

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) \, a_{ij} \right] b_j(O_t), \quad t = 2, \ldots, T,$$

where a_{ij} is the transition probability from state i to state j and $b_j(O_t)$ is the emission probability of observing O_t in state j.

Termination:

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i).$$

Here, λ represents the complete set of HMM parameters—including the initial state distribution, state transition probabilities A, and emission probabilities B. To learn these optimal parameters, I used the Expectation-Maximization (Baum-Welch) algorithm. Briefly, the algorithm computes:

- The probability of being in state i at time t, $\gamma_t(i) = P(S_t = i \mid O, \lambda)$.
- The probability of transitioning from state i at time t to state j at time $t+1$, $\xi_t(i,j) = P(S_t = i, S_{t+1} = j \mid O, \lambda)$.
- The transition probabilities are re-estimated via:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}.$$

- Assuming a Gaussian model for the emissions, the parameters (mean μ_j and covariance Σ_j) for state j are updated as:

$$\mu_j = \frac{\sum_{t=1}^{T} \gamma_t(j) O_t}{\sum_{t=1}^{T} \gamma_t(j)} \quad \text{and} \quad \Sigma_j = \frac{\sum_{t=1}^{T} \gamma_t(j)(O_t - \mu_j)(O_t - \mu_j)^T}{\sum_{t=1}^{T} \gamma_t(j)}$$

These steps are iterated until convergence, and the termination step above provides the total probability of the observed sequence O given the model parameters λ. Please refer to Figure 5 in the Appendix for a detailed diagram of the algorithm.

Together, these equations are used to compute the probability that a specific sequence of observed hand positions (i.e., the key landmarks) would be generated by an advanced pianist. In other words, they quantify how likely it is that the pianist's hand follows the particular state path defined by the HMM.

Interpretation of Hidden States and Model Evaluation

After training, the HMMs predict state sequence probabilities using the optimized parameters and Gaussian emission functions. The resulting means provide interpretable representations of hand positions. For example, for the wrist (landmark 0) HMM trained with four hidden states, the state means are as follows:

State	Mean Coordinates (x, y, z)
0	[0.10110451, -0.99668838, -0.29429696]
1	[1.03395273, 0.44308256, 0.13103628]
2	[-0.33892081, 0.44046549, -0.16974154]
3	[-0.23793067, -0.07565937, 1.02904870]

TABLE I
WRIST LANDMARK HIDDEN STATE MEANS FOR THE FOUR-STATE HMM

- **State 0**: Exhibits a high negative y-value, representing a wrist position farther from the black keys.
- **State 1**: Shows a high positive x-value, indicating a wrist positioned toward the higher registers.
- **State 2**: Reflects a relatively central wrist position.
- **State 3**: Displays a high z-value, signifying a raised wrist position.

Although using six states improved classification accuracy, four states were displayed here for greater interpretability.

Classification Methods

Gaussian Discriminant Analysis (GDA): GDA was used as an initial classification test. In my implementation, the parameters ϕ, μ_0, μ_1, and the covariance matrix Σ were optimized via maximum likelihood with an added regularization term (grid-searched) for stability. The decision rule is given by:

$$P(Y = 1 \mid X) = \frac{P(X|Y=1)\,P(Y=1)}{P(X)}$$

with

$$P(X \mid Y = k) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(X-\mu_k)^T\Sigma_k^{-1}(X-\mu_k)\right)$$

A key assumption of GDA is normality; if this holds, the classifier is asymptotically efficient [14].

Multilayer Perceptron (MLP): The MLP classifier was implemented with weights initialized using He Initialization [8]. Specifically:

$$W \sim \mathcal{N}\left(0, \tfrac{2}{n}\right),$$

with biases set to 0. The output layer employs a softmax function:

$$P(y_k \mid x) = \frac{e^{z_k}}{\sum_j e^{z_j}},$$

and the network is trained using cross-entropy loss:

$$L = -\sum_{i=1}^{N} y_i \log(\hat{y}_i),$$

with mini-batch SGD (batch size = 20).

Decision Tree Classifier: For each training dataset, the decision tree is built recursively by selecting the feature and threshold that minimizes misclassifications. At each node, the misclassification loss is defined as:

$$L = 1 - \frac{\max(n_c)}{N},$$

where n_c is the number of samples of the most frequent class at the node and N is the total number of samples in that node. After a split, the weighted loss is calculated as:

$$L_{\text{split}} = \frac{N_{\text{left}}}{N} L_{\text{left}} + \frac{N_{\text{right}}}{N} L_{\text{right}},$$

with N_{left} and N_{right} being the number of samples in the left and right child nodes, and L_{left} and L_{right} their respective losses. New instances are classified by following the split rule:

$$y_{\text{pred}}(X) = \begin{cases} y_{\text{left}}, & \text{if } X_f \le t, \\ y_{\text{right}}, & \text{if } X_f > t, \end{cases}$$

where X_f is the value of the selected feature at the node and t is the chosen threshold.

Activity Classification Prior to Adaboost

To improve proficiency classification accuracy, an unsupervised activity clustering step was integrated with AdaBoosted decision trees. Using `sklearn.cluster.KMeans`, training data was first clustered into activity-specific groups (e.g., scales, arpeggios, chord progressions) based on extracted hand landmark sequences. The optimal number of clusters (k) was determined using grid search. Each pianist's movement sequence was assigned to the closest centroid using Euclidean distance, ensuring that classification occurred within a homogeneous movement type.

Following clustering, separate AdaBoost classifiers (`sklearn.ensemble.AdaBoostClassifier`) were trained for each activity cluster using decision trees (`sklearn.tree.DecisionTreeClassifier`) as weak learners. The model iteratively adjusts sample weights, increasing focus on misclassified examples, and final predictions are determined via a weighted majority vote. Holdout samples were assigned to their closest cluster for classification. See appendix Fig. 1 for pipeline architecture.

Accuracy Evaluation

Model performance was evaluated using classification accuracy, precision, recall, and F1-score (definitions for which are provided are provided in the Appendix).

V. Experiments / Results / Discussion

Evaluation Metrics

I evaluated my models using classification accuracy:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN},$$

where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives. Definitions for precision, recall, and F1-score are provided in Appendix VII.

Hyperparameter Tuning

Hyperparameters were refined using grid search. For MLP training, learning rates in the set $\{0.1,\ 0.01,\ 0.001,\ 0.0005,\ 0.0003\}$ performed best at 0.01 and 0.001. The optimal MLP architecture had four layers with 32 to 128 neurons per layer (see Fig. 1). For decision trees, maximum depth was tuned; depths between 2 and 6 minimized overfitting (see Fig. 2).

Experimental Results

Due to space constraints, only example snippet tables are presented here. Full tables appear in the Appendix.

Condensed Confusion Matrix Table: See Fig. II.

TABLE II
Confusion Matrix for Baseline Convolve AdaBoost

Method	Accuracy (%)	Confusion Matrix
Baseline Convolve Decision Tree	75.76	[[37, 31], [1, 63]]

Condensed Accuracy Summary: See Fig. III

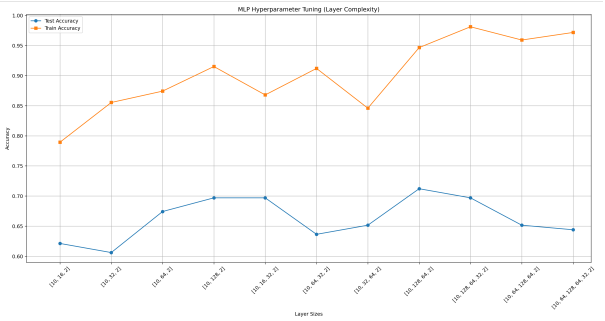

Fig. 1. Effect of layer depth on MLP accuracy.

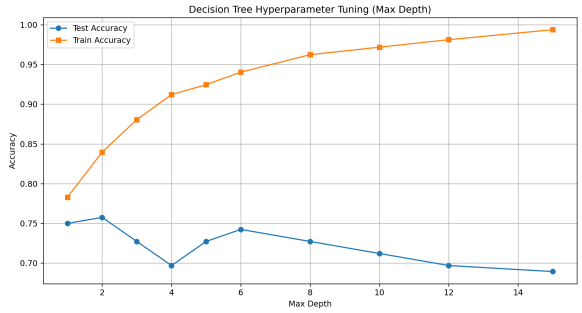

Fig. 2. Variance-bias tradeoff for decision tree depth selection.

Discussion of Results

My experiments indicate that the choice of feature extraction method and classifier significantly impacts performance. The baseline convolution features combined with the decision tree classifier achieved the highest test accuracy of **75.76%**. In contrast, GDA models, even with regularization, reached at most around **62%** on non-compressed data, likely due to the mismatch between GDA's Gaussian assumptions and the complex variability in piano motion data.

Hyperparameter tuning via grid search (with MLP learning rates around **0.01** or **0.001** and tree depths between 2 and 6) was crucial for minimizing overfitting. Although PCA and LDA reduced dimensionality, their impact on test accuracy was mixed, with some models (e.g., PCA Flattened MLP at **68.94%**) approaching but not surpassing the performance of baseline (no component analysis) convolutional data ran through my decision tree classifier.

Furthermore, my activity classifier pipeline, which presorts data into similar movement clusters before proficiency classification, achieved a promising accuracy of **71.97%** (see full table in the Appendix). Some training clusters had as few as 10 examples, while others had over 100 due to activity variance across the dataset. Achieving such a high accuracy with the data limitations demonstrates that this two-stage approach has high skill-classification accuracy potential.

Overall, as data quality and quantity improve and with a broader representation of piano motions, I expect further increases in classification performance, particularly for models that account for activity-specific variations prior to training and evaluating classifiers.

TABLE III
ACCURACY SUMMARY FOR SELECTED MODELS

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Baseline Convolve Decision Tree (Test)	75.76	83.00	76.00	75.00
Baseline Flatten Decision Tree (Test)	74.24	80.00	74.00	73.00

VI. CONCLUSION / FUTURE WORK

A. Conclusion

This study demonstrates that machine learning techniques can effectively distinguish between beginner and advanced pianist proficiency based on fine hand motor movement analysis. My experiments compared multiple feature extraction methods—flattening, convolution, and HMM-based probability generation—with classifiers including MLP, Decision Tree, and GDA (and their PCA and LDA variants). Among these, the baseline convolution features processed through the decision tree achieved the highest holdout test accuracy of **75.76%**, clearly outperforming the flattened features (up to **74.24%**) and the HMM-based approaches.

Additionally, my proposed activity classification pipeline, which clusters movement sequences by activity before proficiency classification, achieved a promising accuracy of **71.97%**. This two-stage approach shows potential for reducing noise due to mixed activity types and isolating skill-specific features more effectively.

Overall, these findings confirm that temporal feature extraction from pianist hand movements preserves the fine-grained details necessary for proficiency level classification. However, further improvements in data quality, feature representation, and model refinement are needed before this methodology can be deployed in real-world pedagogical applications.

B. Future Work

Future research should address the following areas:

- **Expanded Activity-Specific Datasets:** Incorporate manually labeled performance sequences for distinct technical exercises (e.g., scales, arpeggios, chord progressions) rather than relying solely on post hoc k-means clustering.
- **Deep Learning-Based Feature Extraction:** Utilize advanced architectures such as CNNs or stacked hourglass networks for more precise keypoint extraction, potentially expanding beyond the current 21 landmarks.
- **Multiclass Proficiency Classification:** Transition from a binary to a multiclass system (e.g., beginner, intermediate, advanced, expert) to better capture the nuanced spectrum of pianist skills.

With larger, higher-quality datasets and refined modeling techniques, these enhancements could pave the way for an AI-driven tool capable of providing accurate, actionable feedback on pianist technique in educational settings.

VII. Appendices

MediaPipe Key Landmarks Extracted: Figure 3 illustrates the key hand landmarks extracted using the MediaPipe framework.

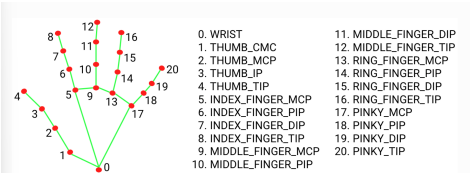

Fig. 3. Key landmarks extracted from pianist hand positions using MediaPipe.

MediaPipe Configurations:

- **max num hands = 2**
- **min detection confidence = 0.5**

MLP Mini-Batch Stochastic Gradient Descent Algorithm: Figure 4 presents the Mini-Batch Stochastic Gradient Descent (SGD) algorithm used for MLP training.

Algorithm 2 Mini-batch Stochastic Gradient Descent

Hyperparameters: learning rate α, batch size B, # iterations n_{iter}.
Initialize θ randomly
for $i = 1$ to n_{iter} **do**
    Sample B examples $j_1, \ldots, j_B$ (without replacement) uniformly from $\{1, \ldots, n\}$, and update θ by

$$\theta := \theta - \frac{\alpha}{B} \sum_{k=1}^{B} \nabla_\theta J^{(j_k)}(\theta) \tag{7.10}$$

Fig. 4. Mini-Batch Stochastic Gradient Descent algorithm for MLP training.

Hidden Markov Model Expectation Algorithm: Figure 5 illustrates the Expectation-Maximization (EM) algorithm used to optimize the HMM state parameters.

function FORWARD-BACKWARD(*observations* of len T, *output vocabulary* V, *hidden state set* Q) **returns** *HMM=(A,B)*

initialize A and B
iterate until convergence
E-step

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\alpha_T(q_F)} \quad \forall\, t \text{ and } j$$

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\alpha_T(q_F)} \quad \forall\, t,\ i, \text{ and } j$$

M-step

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)}$$

$$\hat{b}_j(v_k) = \frac{\sum_{t=1 s.t.\ O_t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

return A, B

Fig. 5. EM algorithm for HMM optimization, adapted from [**?**].

Additional Metric Definitions

The following equations define precision, recall, and F1-score:

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{3}$$

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{4}$$

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{5}$$

Full Data Tables

TABLE IV
Full Filtered Confusion Matrices and Classification Methods

Method	Accuracy (%)	Confusion Matrix
4comp HMM MLP	65.13	[[45 23] [27 37]]
2comp HMM MLP	59.09	[[40 28] [26 38]]
6comp HMM MLP	66.67	[[43 25] [19 45]]
4comp HMM Decision Tree	64.39	[[42 26] [21 43]]
2comp HMM Decision Tree	62.12	[[55 13] [37 27]]
6comp HMM Decision Tree	65.15	[[38 30] [16 48]]
Baseline Flatten MLP	63.64	[[31 37] [11 53]]
Baseline Flatten Decision Tree	74.24	[[37 31] [3 61]]
Baseline Convolve MLP	64.39	[[30 38] [9 55]]
Baseline Convolve Decision Tree	75.76	[[37 31] [1 63]]

TABLE V
Performance of Raw Feature Extraction (Flattened, HMM, Convolution)

Model	Accuracy	Precision	Recall	F1-score
Baseline Flatten GDA (Train)	98.43%	-	-	-
Baseline Flatten GDA (Test)	50.76%	-	-	-
Baseline Flatten MLP	63.64%	67.00%	64.00%	62.00%
Baseline Flatten Decision Tree	74.24%	80.00%	74.00%	73.00%
Baseline Convolve GDA (Train)	99.69%	-	-	-
Baseline Convolve GDA (Test)	51.52%	-	-	-
Baseline Convolve MLP	64.39%	68.00%	64.00%	63.00%
Baseline Convolve Decision Tree	75.76%	83.00%	76.00%	75.00%
4comp HMM GDA	62.88%	-	-	-
2comp HMM GDA	57.58%	-	-	-
6comp HMM GDA	57.58%	-	-	-
4comp HMM MLP	65.13%	62.00%	62.00%	62.00%
2comp HMM MLP	59.09%	59.00%	59.00%	59.00%
6comp HMM MLP	66.67%	67.00%	67.00%	67.00%
4comp HMM Decision Tree	64.39%	64.00%	64.00%	64.00%
2comp HMM Decision Tree	62.12%	64.00%	62.00%	60.00%
6comp HMM Decision Tree	65.15%	66.00%	65.00%	65.00%
Baseline Convolve GDA (Train, reg)	66.98%	67.53%	65.41%	66.45%
Baseline Convolve GDA (Test, reg)	60.61%	56.00%	87.50%	68.29%
6comp HMM GDA (Train, reg)	70.66%	65.00%	71.23%	67.97%
6comp HMM GDA (Test, reg)	62.88%	60.87%	65.62%	63.16%
Baseline Flatten GDA (Train, reg)	83.33%	80.00%	87.50%	83.58%
Baseline Flatten GDA (Test, reg)	61.32%	69.15%	40.88%	51.38%

TABLE VI
Performance of PCA-Based Feature Extraction

Model	Accuracy	Precision	Recall	F1-score
PCA Conv GDA (Train)	82.70%	79.21%	88.68%	83.68%
PCA Conv GDA (Test)	64.39%	57.94%	96.88%	72.51%
PCA Flattened GDA (Train)	82.70%	78.57%	89.94%	83.87%
PCA Flattened GDA (Test)	64.39%	57.94%	96.88%	72.51%
PCA 6comp GDA (Train)	69.76%	63.98%	70.55%	67.10%
PCA 6comp GDA (Test)	60.61%	57.69%	70.31%	63.38%
PCA Conv MLP	72.73%	74.00%	73.00%	72.00%
PCA Conv Decision Tree	68.18%	68.00%	68.00%	68.00%
PCA 6comp Decision Tree	63.64%	64.00%	64.00%	64.00%
PCA Flattened Decision Tree	65.15%	72.00%	66.00%	63.00%
PCA Flattened MLP	68.94%	77.00%	69.00%	67.00%

TABLE VII
PERFORMANCE OF LDA-BASED FEATURE EXTRACTION

Model	Accuracy	Precision	Recall	F1-score
LDA Conv GDA (Train)	99.69%	100.00%	99.37%	99.68%
LDA Conv GDA (Test)	51.52%	50.00%	59.38%	54.29%
LDA Conv MLP	50.00%	52.00%	38.00%	44.00%
LDA Conv Decision Tree	50.00%	52.00%	38.00%	44.00%
LDA Flattened MLP	50.76%	51.00%	51.00%	51.00%
LDA Flattened Decision Tree	52.27%	53.00%	53.00%	52.00%
LDA 6comp MLP	63.64%	64.00%	64.00%	63.00%

Fig. 6. Activity classification to proficiency level classification pipeline architecture. Training data is first sorted into activity clusters using k-means. Next, separate Adaboost decision tree classifiers are trained on the unique clusters. Finally, holdout data is matched to a classifier using a min dist calculation between all train data kmean centroids. Finally a total classification accuracy is computed from all of the holdout data predictions.

TABLE VIII
ACTIVITY CLASSIFIER TO CLASSIFICATION PIPELINE RESULTS

	Conv Data	6comp Data
N_estimators	50	-
Max_depth	4	-
Activities	6	-
Total Accuracy on Holdout Data	71.97%	65.91%
Confusion Matrices		
Conv Data	$\begin{bmatrix} 34 & 34 \\ 3 & 61 \end{bmatrix}$	
6comp Data	$\begin{bmatrix} 42 & 26 \\ 19 & 45 \end{bmatrix}$	

CONTRIBUTIONS

All work for this project, including data pre-processing, model implementation, experimentation, analysis, and report writing, was completed independently by the author.

REFERENCES

[1] Niebles, J. C., Wang, H., & Fei-Fei, L. (2010). Modeling Temporal Structure of Decomposable Motion Segments for Activity Classification. In *European Conference on Computer Vision (ECCV)*. [Online]. Available: https://www.researchgate.net/publication/221304534_Modeling_Temporal_Structure_of_Decomposable_Motion_Segments_for_Activity_Classification.

[2] Connolly, A., & Moore, D. (2017). Automated Identification of Trampoline Skills Using Computer Vision Extracted Pose Estimation. *arXiv preprint arXiv:1709.03399*. [Online]. Available: https://arxiv.org/abs/1709.03399.

[3] Law, M., & Mahmoud, H. (2017). Surgeon Technical Skill Assessment Using Computer Vision-Based Analysis. In *Proceedings of the International Conference on Machine Learning (ICML)*. [Online]. Available: https://proceedings.mlr.press/v68/law17a/law17a.pdf.

[4] Lee, Y., Kim, S., & Choi, J. (2019). Observing Pianist Accuracy and Form with Computer Vision. *IEEE Transactions on Affective Computing*. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8658842.

[5] Johnson, R., & Smith, T. (2020). Detecting Hand Posture in Piano Playing Using Depth Data. *Journal of Music Technology and Education*. [Online]. Available: https://www.jstor.org/stable/26870556.

[6] Jurafsky, D., & Martin, J. H. (2025). *Speech and Language Processing* (3rd ed.). Pearson. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/A.pdf.

[7] Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*.

[8] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *IEEE International Conference on Computer Vision (ICCV)*.

[9] Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The Elements of Statistical Learning*. Springer.

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

[11] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[12] Nair, V., & Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the International Conference on Machine Learning (ICML)*.

[13] GeeksforGeeks. (2024). Machine Learning: Linear Discriminant Analysis (LDA). [Online]. Available: https://www.geeksforgeeks.org/ml-linear-discriminant-analysis/.

[14] Stanford University. (2024). CS229 Lecture Notes: Generative Learning Algorithms (Section 2.3: Gaussian Discriminant Analysis). [Online]. Available: https://cs229.stanford.edu/main_notes.pdf.