Key Notes, Key Features: A Temporal Convolutional Approach to Pianist Technique Classification

Max Rodriguez*
Stanford University
Stanford, CA
maxrod@stanford.edu

Renn Su*
Stanford University
Stanford, CA
rrsu@cs.stanford.edu

Abstract

Automated piano technique assessment presents unique challenges due to the complexity of fine motor movement and the need to model temporal dynamics. We propose a system that combines multi-branch convolutional neural networks (CNNs) with optical flow-based feature modeling to address the limitations of existing static frame-based classification methods. Our model architecture achieves a 75% classification accuracy on holdout data when trained on a small dataset, suggesting that our featurization methods and classifier architecture have the potential to advance assessment of fine-motor skill.

1. Introduction

Achieving professional-level technique in fine motor skills such as piano requires consistent feedback from trained instructors, as early-stage errors can hinder progression and cause chronic injury [1]. High-quality piano instruction is often inaccessible, causing students to rely on self-guided practice or online resources and become prone to errors and misinformation. While online education has expanded access to music learning, there remains a critical need for automated, accurate feedback on piano technique.

Previous attempts to address fine motor skill assessment using computer vision have relied on static frame-based analysis [2, 3], failing to capture the inherently sequential nature of piano playing. To address this gap, we present a multi-branch convolutional neural network (CNN) with optical flow-based temporal modeling, enabling fine-grained assessment of piano technique by modeling both local and global motion patterns.

There is a growing body of existing work surrounding activity detection and skill classification. With this project, we wanted to contribute by exploring convolutional approaches to skill classification in a context that is (1) dependent on motion over time and (2) takes advantage of the features that define fine motor movement.

2. Related Work

2.1. Computer Vision in Piano Pedagogy

Work by Lee et al. [2] and Johnson et al. [3] uses varying computer vision architecture to analyze pianist technique. We noticed that Lee et al. do not use temporal information in their visual classifier, and Johnson et al. use a single-frame approach for their video feed. However, piano technique cannot be sufficiently analyzed through static images, as piano feedback should consider dynamic transitions rather than isolated postures [1].

2.2. Modeling Temporal Information in Machine Learning

Temporal relationships in computer vision remain a significant challenge, particularly when modeling contextual motions [4]. Although Hidden Markov Models (HMMs) provide a promising framework for capturing the relationships between motion states and observed feature representations [5, 6], they tend to under-perform when predicting probabilities for motion sequences with high activity variance—such as those exhibited by pianists performing a diverse range of motions. In contrast, optical flow has emerged as a widely adopted method for extracting temporal features prior to classification. For instance, one approach employed optical flow to detect violence in video clips [10], whereas our project uses optical flow first to classify gross motor activity and then to detect subtle differences in fine motor skills.

2.3. Fine Motor Movement Detection in Computer Vision

Previous work has demonstrated varying methods for activity detection in videos. Limitations of existing hand detection approaches often fall into two categories (1) they are general purpose activity classification architectures not optimized for fine-motor movement [4], and (2) the models do not account for temporal information and instead rely on single frame analysis [13].

Our method extracts temporal data via optical flow and convolution across images sequences extracted from video. Secondly, we only use keypoints that represent the most significant movement across frames as opposed to using all keypoints, eliminating resting hand keypoints that are not as informative for technique classification.

3. Methodology

3.1. Data collection

We collected 100 recordings of pianists sourced from in-person recordings and YouTube videos (50 beginner, 50 advanced). All videos were taken from a bird's-eye view (BV) to ensure consistency between angles. Pianists were recorded performing multiple motion sequences such as scales, arpeggios, and chord progressions.

Approximately 10 sequences of 20 images were taken from each video. Each image in the sequence was separated by a 1 frame gap. Both left and right hands were isolated for further video processing. See appendix for further details.

3.1.1 Data Featurization

Prior to featurization, hand-images were normalized through use of the Google MediaPipe Hands method to detect 21 key landmarks per hand. The vector formed by subtracting the wrist landmark $3D$ point from the corresponding middle finger MCP $3D$ point was used to calculate the angle between a y-axis aligned unit vector for each image. With the unique angle calculated for each image, a rotation matrix was formed to orient all hands parallel to a common axis.

The post-orientation featurization pipeline is described in the following subsections.

3.1.2 Harris Corner Detection

Once videos are segmented into 20-image, chronologically-ordered sequences, keypoints are calculated across all frames. First, a Sobel 3-by-3 kernel is used to compute pixel gradients. To calculate the structure tensor M for each pixel, a rectangular 2-by-2 window is used. Finally, corner response is calculated at each pixel using,

$$R = \det(M) - k \cdot \mathrm{trace}(M)^2$$

The corners are then amplified using morphological dilation:

$$I'(x,y) = \max_{(u,v)\in K} I(x+u, y+v)$$

where the neighborhood K is defined as a 3×3 window, which helps to reduce noise.

The keypoints are further refined by applying a maximum corner response threshold of 0.01. Finally, the centroids of the keypoints are determined using `cv2.connectedComponentsWithStats()`.

Keypoints detected via Harris corner detection are ideal for optical flow, as they contain multiple gradients in unique directions—eliminating those located on edges or in flat regions [7].

3.1.3 Filtering Keypoints Using SIFT and BF Match

Using the keypoints computed with Harris corner detection, the `cv2` SIFT function is used to assign each point a 16×16 pixel window. For each pixel in this window, the gradient magnitude is computed as

$$M(x,y) = \sqrt{(I(x+1,y) - I(x-1,y))^2 + (I(x,y+1) - I(x,y-1))^2}$$

and the orientation is calculated by

$$\theta(x,y) = \tan^{-1}\left(\frac{I(x,y+1) - I(x,y-1)}{I(x+1,y) - I(x-1,y)}\right).$$

These values are stored in a 128-dimensional feature vector for each keypoint.

Next, `cv2`'s brute force match method is used to compute the Euclidean distance between all keypoint descriptor vectors between adjacent frames. Descriptors with at least two candidate matches are then confirmed as matches using Lowe's ratio test with an empirical constant threshold of 0.85.

This SIFT and BF match method is applied prior to optical flow calculations for the keypoints, as optical flow assumes that points remain within the frame bounds across consecutive frames and that the surrounding patch for each keypoint exhibits constant flow. Constant flow requires two conditions:

1. Flow is locally smooth [8]: all points in the neighborhood of a keypoint do not experience abrupt changes between frames.

2. Neighboring pixels of a keypoint have the same displacement across frames [8].

Together, SIFT detection and BF match filter out points that are unstable across frames.

3.1.4 Calculating Optical Flow for Select Keypoints

Next, optical flow is applied to capture the temporal evolution of keypoint features across sequences. The Lucas–Kanade (LK) method is employed to calculate the optical flow for each keypoint that has been confirmed to be in-frame and scale/rotation invariant through SIFT and BF match.

A larger 10×10 patch is used to compute optical flow via the LK method, as hand motion between frames is assumed

to be smooth. Assuming that brightness remains constant between frames, we use the following constraint equation

$$I_x u + I_y v + I_t = 0$$

to estimate the displacement between frames, where (x, y) is the keypoint in question and I_t represents the difference in intensity at (x, y) between frames.

Since the equation is under determined (with two unknowns, u and v), we use the surrounding points in the neighborhood to compute the displacement vector $\hat{x}$ by solving

$$\hat{x} = \arg\min_x \|Ax - b\|^2,$$

which simplifies to solving

$$A^T A\, \hat{x} = A^T b.$$

(See [8].) Where A is the 100×2 matrix of (I_x, I_y) gradients for each point and b is the $1D - vector$ for I_t at each point.

The optical flow outputs are refined iteratively using pyramidal affine tracking [9].

Since keypoints were computed using Harris corner detection, both eigenvalues of the matrix $A^T A$ satisfy the condition $\lambda_1 \geq \lambda_2 > 0$ and are well-conditioned (i.e., $\lambda_1/\lambda_2 \approx 1$).

3.2. Data Preprocessing:

Once the optical flow (u, v) is computed for each keypoint, x, y, z spatial coordinates, intensity, and optical flow for each keypoint per frame are stored in a matrix with dimensions $19 \times KP$. 19 represents the temporal frames of each image (the 20th frame is excluded due to lack of optical flow). KP is a representation of all the features selected after the principal component analysis (PCA) process.

In order to maintain uniformity in the number of features across sequences and reduce computational cost, only the keypoints with the top optical k (*in our case* $k = 10$) flow magnitudes are kept for each frame.

Additionally, Keypoints, greyscale values, and optical flow values were standardized via Z-score normalization across the dataset prior to clustering and classification.

3.2.1 K-Means Activity Classifier:

After data featurization and preprocessing, we were left with 200 beginner and advanced holdout samples and 400 beginner and advanced train samples. Data sets were shuffled and fed through the CNN for classification, initially, without prior clustering.

However, to better separate data into similar motion patterns, hierarchical k-means clustering was used to sort data into unsupervised groups.

3.3. Convolutional Neural Network Architecture

The convolutional neural network (CNN) needed to address two major considerations. First, we needed to address both finger-level and hand-level motions in our data. Second, our CNN input consisted of a collection of features as opposed to images. However, conventional CNN inputs have been used for images. Key points have been used in CNNs by Li et al., where a two-branch architecture was trained on 3D Cartesian motion and skeleton motion [11].

3.3.1 Network Architecture

Our network had 10 layers total, consisting of 3 convolutional layers, batch normalization, and pooling layers, followed by a global average pooling layer and a fully connected classifier.

The CNN takes an input of $(B, 5, 1, 150)$, where B represents the batch size. It employs two parallel convolutional branches: Branch 1 processes finer movements using a Conv2D (1×3, 16 filters), followed by batch normalization and ReLU activation, before downsampling via a 1×2 max pooling layer. Branch 2 captures larger movements with a Conv2D (1×5, 16 filters), followed by batch normalization, ReLU activation, and 1×2 max pooling.

The MaxPool (1×2) operation in both branches reduces input width from 150 to 75, downsampling the data while preserving the dominant motion patterns. The aggregation of these two branches through concatenation and feature fusion steps effectively identifies interactions between fine and broad movements, enabling the network to capture complex motion patterns that span varying temporal scales.

Both branches generate output feature maps of size $(B, 16, 1, 75)$, which are concatenated to form a $(B, 32, 1, 75)$ feature representation.

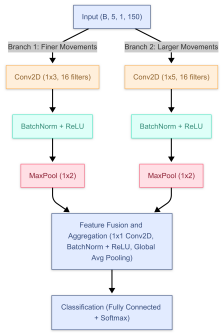

Figure 1. Architecture of the two-branch CNN model.

3.3.2 Alternate Single-Stream Architecture

A single-branch variation of our CNN architecture was tested in comparison to our two branch solution. The baseline single-stream architecture consisted of three con-

secutive convolutional blocks for fine movement extraction, broad motion extraction, and high-level features. The first convolutional block is designed to capture fine-grained movement details using a Conv2D (1×3) with 16 filters and padding $(0, 1)$ to preserve the temporal dimension. The second block utilizes a Conv2D (1×5) with 32 filters and padding (0,2) to increase the receptive field. Both blocks utilize Batch Normalization, ReLU activation, and a Max-Pooling (1×2) layer that reduces the input width from 150 to 75. Global average pooling is applied to the results, followed by flattening, a fully connected layer, and softmax activation.

3.3.3 Cluster-Based Pipeline

In order to investigate the effectiveness of using a k-means activity classifier, we performed trials in which the two-branch CNN (see Section 3.3.1) was run separately on each activity cluster. We then tracked overall performance across clusters.

4. Results

Note on the MLP Baseline: Initial tests were carried out by running MediaPipe key-landmark data directly through a multi-layer perceptron (MLP) binary classifier. This baseline was run to test novel featurization and classification methods presented through this project. As seen in Table 1, the baseline model achieved an accuracy of 60.3% on the training/validation data and 53.9% on the holdout set. The results highlight potential limitations of using MLP for motion classification tasks.

Table 1. Comparison of model performance on training/validation and holdout data.

Model	Training/Validation	Holdout Data
Baseline Using MLP	60.3%	53.9%
Single Branch CNN	100.0%	54.1%
Two-Branch CNN	98.7%	**75.0%**
Two-Branch CNN with Clustering	100.0%	72.7%

The two-branch CNN architecture significantly outperformed the single-branch CNN and the MLP baseline. As presented in Table 1, the two-branch CNN achieved an accuracy of 98.7% on training/validation data and 75.0% on holdout data, compared to 54.1% achieved by the single-branch CNN. The 20.9% improvement suggests effectiveness in using a two-branch CNN architecture to leverage both fine and broad movement patterns.

Though clustering reduced available data, performance was comparable at 72.7% on the holdout set. The result suggests that the model architecture remains robust even with reduced training data.

The confusion matrix in Table 3 shows that the two-branch model accurately identified 91 out of 100 "Advanced" players, with only 9 misclassified as "Beginner," demonstrating the model's ability to recognize advanced movements. Beginner classification accuracy was lower, with 41 out of 100 instances mislabeled as "Advanced." A potential cause of the discrepancy is the broader variability in beginner movement patterns compared to well-defined advanced technique.

Our model exhibited overfitting in the validation stage, highlighting the challenges in differentiating between fine motor features with limited training data. Small reductions in variance or increases in bias can result in fluctuations in holdout and validation accuracy, justifying the prioritization of higher overall accuracy at the cost of increased overfitting.

5. Conclusion

Our project demonstrates a promising, low computational-load, method for classifying the fine motor-controlled hand movements of pianists. With a much smaller data set, we were able to achieve comparable results to published work in CNN approaches to fine motor skill classification in specialized contexts such as surgery [12].

Data featurized using temporal flow far outperformed static hand landmark data. However, to confirm the dominance of temporal featurization methods for proficiency level classification, more tests (varying classification methods and static feature extraction techniques) are needed.

Despite promising outcomes, our model showed a disparity in classifying beginner movements compared to advanced ones, likely due to greater variability in beginner patterns. Furthermore, a key consideration for our CNN model was overfitting, especially with limited datasets, emphasizing the need for more diverse data and improved generalization techniques.

6. Future Work

Future research should address the following areas:

- **Granular Skill Level Classification:** Address discrepancies in beginner classification by implementing a multiclass system (e.g., beginner, intermediate, advanced, expert) to better capture the spectrum of pianist skills.
- **Further Architecture Changes and Hyperparameter Tuning:** Explore variations in model architecture and fine-tune hyperparameters to maximize performance and generalization in a broader range of activities and contexts.

With larger, higher-quality datasets and refined modeling techniques, these enhancements motivate further exploration in the development of a more robust and optimized classification system.

7. Individual Contributions

7.1. Max Rodriguez's Contribution:

The main tasks for which I was responsible include:

1. **Sequencing and labeling of video frames:** Azure blob storage was set up to organize and store video and image data. I wrote Python scripts for the extraction of 20-frame sequences.

2. **Featurization and preprocessing of data:** Responsible for methods implemented from section **3.1.1** through **3.1.4**. The Python script for this featurization pipeline can be found in the GitHub repository listed in the ***Appendix***.

3. **Agglomerative Activity Clustering:** Responsible for generating training and holdout CSVs sorted via hierarchical clustering of k-means. The script for this method can also be found in the **Appendix**.

4. **MLP Baseline tests:** Additionally, I performed baseline MLP classification tests to compare against the final CNN accuracy results. The MLP architecture was designed during a separate Stanford CS229 project.

7.2. Renn Su's Contribution:

The main tasks for which I was responsible include:

1. **Design and Implementation of CNN Architectures:** Reviewed existing literature and methods surrounding convolutional neural networks, designed and implemented various architectures optimized for our data set and use case (i.e. two-branch, single branch, clustered approach from **3.3.1** to **3.3.3**)

2. **Iteration and Evaluation of Data-to-CNN Pipeline:** Responsible for modifying post-processing data in preparation for the CNN. Iterated on approaches of interpreting multi-dimensional and multi-feature keypoint data and evaluated performance and robustness.

3. **Optimization of CNN Model Runtime:** Developed and implemented strategies to reduce computational complexity and improve runtime efficiency, including model pruning, parameter tuning, and leveraging GPU acceleration.

4. **Review of Anatomical and Pedagogical Approaches:** Conducted an in-depth review of anatomical and pedagogical approaches in previous literature to ensure the relevance of our movement recognition criteria and CNN architecture. Reviewed criteria for skill-based classification.

Both team members contributed equally and significantly to preliminary literature review, project design and ideation, data collection, process and result documentation, presentation, and the authorship of this paper. Both authors share first authorship, and the names are listed in alphabetical order.

8. Appendix:

The code and data used in this project are available in our GitHub repository: https://github.com/MaxLuisRodriguez/Updated-CS131-FinProj

The repository contains:

1. The main scripts.

2. Processed data used during experiments.

Table 2. Classification report for the two-branch CNN model performance on holdout data.

Class	Precision	Recall	F1-Score	Support
Beginner	0.87	0.59	0.70	100
Advanced	0.69	0.91	0.78	100
Macro Avg	0.78	0.75	0.74	200
Weighted Avg	0.78	0.75	0.74	200
Accuracy	0.7500			

Table 3. Confusion matrix for the two-branch CNN model performance on the holdout data.

Predicted / Actual	Beginner	Advanced	Total
Beginner	59	41	100
Advanced	9	91	100
Total	68	132	200

Table 4. Classification report for the two-branch CNN model performance on the test set.

Class	Precision	Recall	F1-Score	Support
Beginner	0.97	1.00	0.99	35
Advanced	1.00	0.98	0.99	41
Macro Avg	0.99	0.99	0.99	76
Weighted Avg	0.99	0.99	0.99	76
Accuracy	0.9868			

Table 5. Confusion matrix for the two-branch CNN model performance on the test set.

Predicted / Actual	Beginner	Advanced	Total
Beginner	35	0	35
Advanced	1	40	41
Total	36	40	76

Note on video sourcing from YouTube: Beginner data was primarily sourced from videos containing keywords

such as ”beginner”, ”novice”, ”self-taught”. Much of the advanced data was sourced from footage from the International Chopin Piano Competition in BV.

To address the question of categorization, our team binarized the training data into beginner and advanced. Each member of our research team (both having been classically trained in piano) manually reviewed the categorizations of skill level.

References

[1] S. Furuya and E. Altenmüller, ”Flexibility of movement and improvement of motor performance in musicians,” *Annals of the New York Academy of Sciences*, vol. 1252, no. 1, pp. 222-228, 2013. 1

[2] H. Lee, J. Kim, and S. Choi, ”Pianist technique analysis using computer vision,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, 2019, pp. 1234–1243. 1

[3] D. Johnson, D. Damian, and G. Tzanetakis, ”Detecting Hand Posture in Piano Playing Using Depth Data,” in Comput. Music J., vol. 43, no. 1, pp. 59–78, Jan. 2020 1

[4] J. C. Niebles, H. Wang, and L. Fei-Fei, ”Modeling temporal structure of decomposable motion segments,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Anchorage, AK, USA, 2008, pp. 1–8. 1

[5] K. Tang, R. Sukthankar, and J. Li, ”Temporal action localization in unstructured videos,” in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Florence, Italy, 2012, pp. 198–213. 1

[6] A. Wilson and A. Bobick, ”Parametric hidden Markov models for gesture recognition,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Fort Collins, CO, USA, 1999, pp. 223–230. 1

[7] B. D. Lucas and T. Kanade, ”An iterative image registration technique with an application to stereo vision,” in *Proc. 7th International Joint Conference on Artificial Intelligence (IJCAI)*, Vancouver, BC, Canada, 1981, pp. 674–679. 2

[8] Carnegie Mellon University, ”16-385 Computer Vision Lecture 21,” in *Course Lecture Notes, CMU*, Pittsburgh, PA, USA, 2015. [Online]. Available: https://www.cs.cmu.edu/ 16385/s15/lectures/Lecture21.pdf. 2, 3

[9] J. Y. Bouguet, ”Pyramidal implementation of the affine Lucas-Kanade feature tracker,” Intel Corporation, 2001. 3

[10] M. A. Khan, S. H. Raza, and H. M. Islam, ”Optical flow-based violence detection in video surveillance,” *Expert Systems with Applications*, vol. 133, pp. 160–171, 2019. 1

[11] C. Li, Q. Zhong, D. Xie, and S. Pu, ”Skeleton-based action recognition with convolutional neural networks,” *arXiv preprint arXiv:1704.07595*, 2017. 3

[12] D. Kitaguchi, N. Takeshita, H. Matsuzaki, T. Igaki, H. Hasegawa, and M. Ito, ”Development and validation of a 3-dimensional convolutional neural network for automatic surgical skill assessment based on spatiotemporal video analysis,” *JAMA Network Open*, vol. 4, no. 8, pp. e2120786, Aug. 2021. 4

[13] A. Suteparuk, ”Detection of Piano Keys Pressed in Video,” Ph.D. dissertation, Stanford University, 2021. Available: https://stacks.stanford.edu/file/druid:bf950qp8995/Suteparuk.pdf. 1