Mediapipe Fingering Classification for Violin

Spencer Wang

Thomas Jefferson High School for Science and Technology, Arlington, Virginia
Email: spencerycwang@hotmail.com

Abstract

The extent of the potential of Mediapipe in analyzing and classifying complex hand patterns, like those found in violin hand movements, remains elusive. This research explores the feasibility of using Mediapipe, an open-source framework that allows the extraction of finger position key points from images/videos, to detect violin fingering positions for playing a virtual instrument. Further, we explore the limitations and potential of Mediapipe for recognizing complex hand movements involved in playing the violin by using a combination of Google Colab and support vector classification models. The repeated training on misclassified images was able to take the F1 score from an initial value of 0.82 to a final value of 0.95. This initial research shows the potential for a video based virtual violin instrument. Further areas of exploration could include expanding the total image training data and testing viability of other classifiers.

1) Introduction

The intersection of artificial intelligence and music technology has opened up new possibilities for innovative applications, from music composition to performance [1-3]. One such application is the creation of a virtual musical instrument that can replicate the nuanced experience of playing a real instrument [4]. This vision involves three critical steps: accurately detecting the fingering positions, recognizing hand placement on the instrument, and capturing the bowing technique. This paper focuses on perfecting the fingering classification process, a fundamental aspect of this vision.

By achieving precise recognition of violin fingering positions through image recognition and machine learning, such as utilizing Mediapipe—an open-source framework developed by Google for real-time hand tracking and gesture recognition [5]—we can open the door to new forms of creative expression and accessibility in music. A virtual instrument capable of mimicking these complex gestures would not only enable musicians to practice and perform without the physical instrument but also allow for new types of musical expression that blend digital and physical realms. This endeavor is worth pursuing because it merges technology with artistic creativity, providing musicians with a powerful tool that expands their creative possibilities while also making music more accessible to a broader audience.

Despite the potential of Mediapipe, its effectiveness in recognizing complex and precise movements, such as those required for playing the violin, remains largely unexplored. Traditional methods of training models for such tasks face challenges related to data collection, computational resources, and varying environmental conditions [6]. This research is a proof of concept that such training is possible for complex motions, like the violin, through just a simple webcam. As shown through Figure 1, we use mediapipe to extract keypoints, simplifying the computer vision process, and then use a support vector machine to classify the key points as this is a technique that requires minimal data and resources as compared to traditional convolutional neural network and direct classification [7, 8].

This study contributes to the field of AI-driven music technology by exploring a novel application of Mediapipe. The findings can inform future research and development of virtual musical instruments, enhancing both educational tools and performance aids.

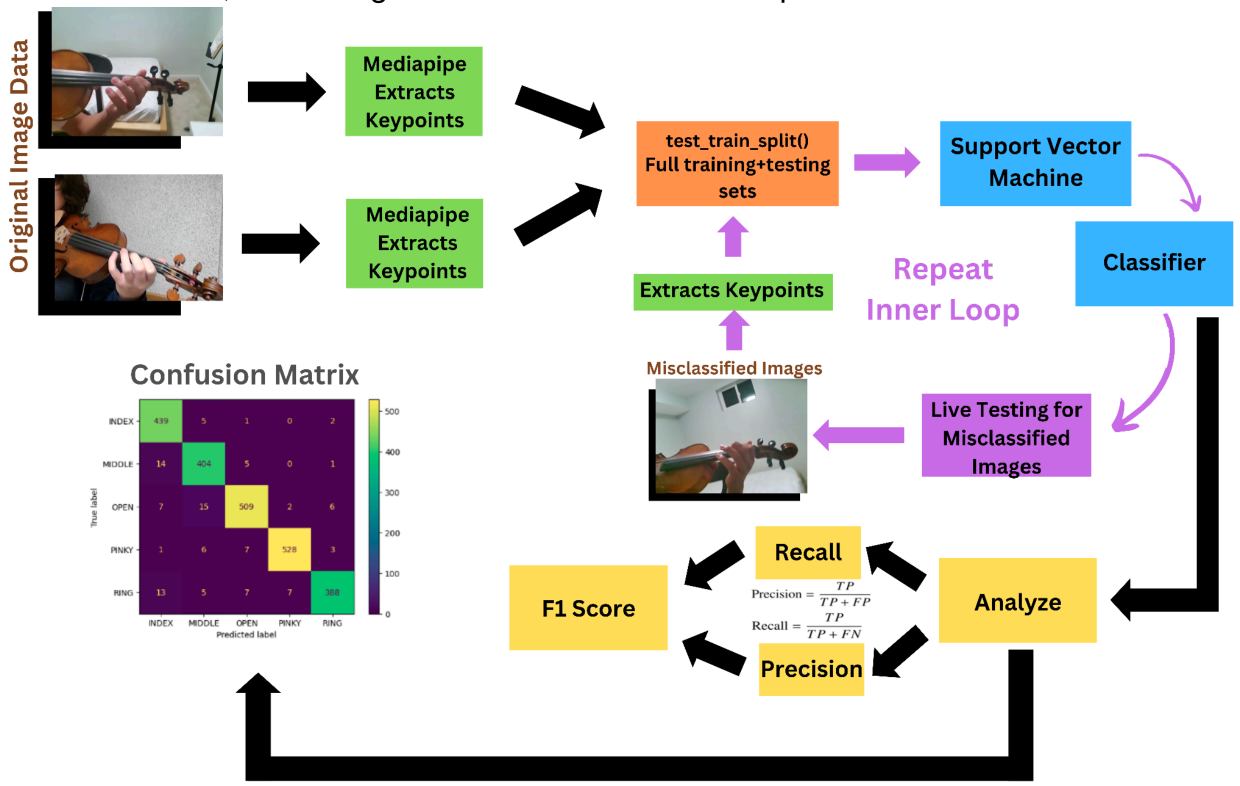


Figure 1: Overview of algorithm and data processing for Classifying Violin Fingering Positions Using Image Data. Camera data is collected which Mediapipe then extracts key point data showing the finger positions. A support vector machine classifier is trained. The classifier is tested on live images and any images that are misclassified go back into the training and testing sets. After augmenting the training set with misclassified images, a F1 score of 0.95 is achieved.

1.1) Objectives

1. Assess the capability of Mediapipe in detecting violin fingering positions.
2. Evaluate the accuracy of a support vector classifier (SVC) trained on hand key points detected by Mediapipe.
3. Identify the limitations and challenges associated with using Mediapipe for this application.
4. Explore the potential for real-time application and performance under varied conditions.

2) Methodology

2.1) Tools and Environment

The tools and environment used in this project include software such as Mediapipe, Google Colab, OpenCV, and scikit-learn, along with a webcam for data collection.

2.2) Data Collection

2.2.1) Enabling Camera Usage in Google Colab

Open-CV does not directly support camera access in Google Colab. To bypass this, a JavaScript code was used to allow camera access in Google Colab. [9]

2.2.2) Automated Photo Capture

We developed a script in Colab to automate the capture and saving of training photos to Google Drive. The script was designed to take photos by slowly waving the camera around, capturing different angles and positions. The specific camera used was the Polycom EagleEye Mini as it was small and could easily be moved around to take photos at different angles. All photos were taken indoors with the camera running at 30 frames per second and each image had the dimensions of 640 x 480 pixels.

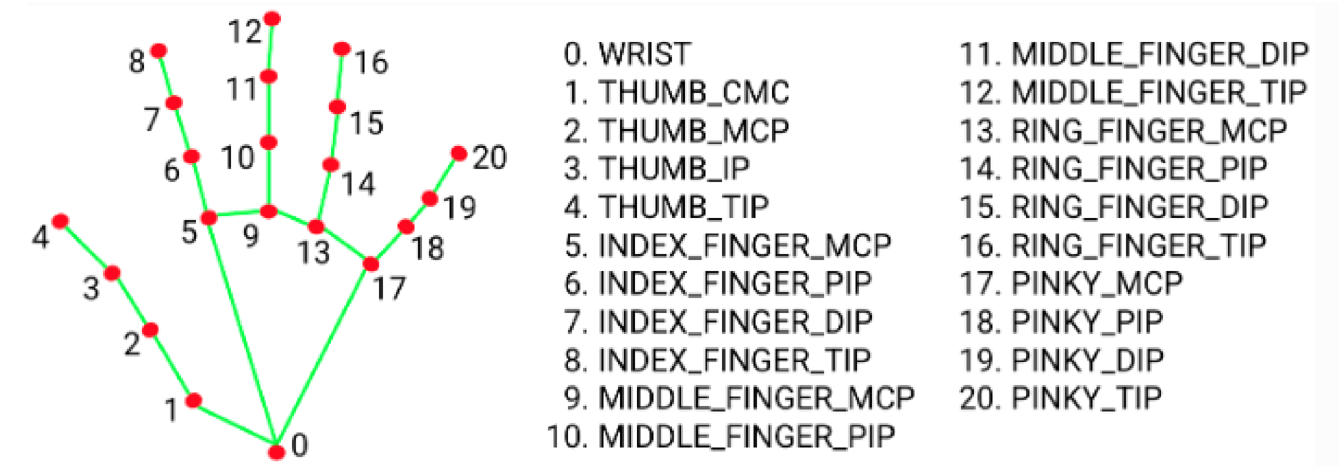
2.3) Data Labeling

Each captured photo was labeled and saved, with Mediapipe detecting and recording the hand key points.

2.4) Data Processing

2.4.1) Keypoint Extraction

For each photo, Mediapipe was used to extract 16 hand key points (x, y coordinates) as shown in Figure 2. These 16 hand key points can be plotted on the respective image, as shown in Figure 3, showing what Mediapipe would “see” when analyzing a hand position. The extracted key points were saved to a text file for subsequent analysis.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figure 2: Specific key points “name” and the index to find them when retrieving all data points[6].

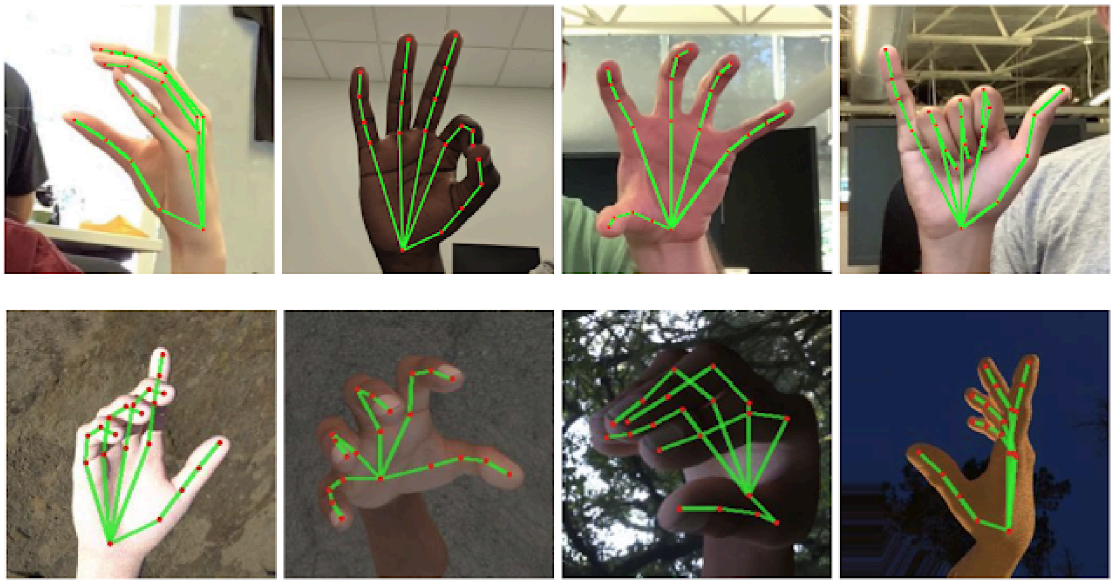


Figure 3: Given an image of a hand, Mediapipe is able to take that image and extract the location of the key points on the hands [6]. What is shown here are examples of what Mediapipe would see to visualize key points on different hand shapes, lighting, and backgrounds

2.5) Model Training

2.5.1) Dataset Preparation

The dataset was split into training and testing sets using the `train_test_split` function from scikit-learn. The model parameters were fine-tuned to optimize performance (80% training set, 20% testing set, C value = 10).

2.5.2) Support Vector Machine

An SVM with an RBF kernel was trained on the training set. SVM typically performs well with high-dimensional and unstructured datasets, which works well in this situation with image data processing. One general downside for using a SVM is computational speed. However, this wasn't a significant issue in this scenario with a considerably low number of training data.

2.5.3) Evaluation

The trained model was evaluated using a confusion matrix to measure accuracy on the test set as well as the F1 score on the total combined test set. The F1 score is an evaluation metric for machine learning models that combines precision and recall, providing a balanced measure of accuracy. Unlike simple accuracy, which can be misleading on imbalanced datasets, the F1 score accounts for both false positives and false negatives*, making it a more reliable metric when class distribution is uneven, which is especially useful during the misclassification backtrack training as the number of images per class gets unbalanced.

$$\text{Precision} = \frac{TP}{TP + FP} \qquad F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

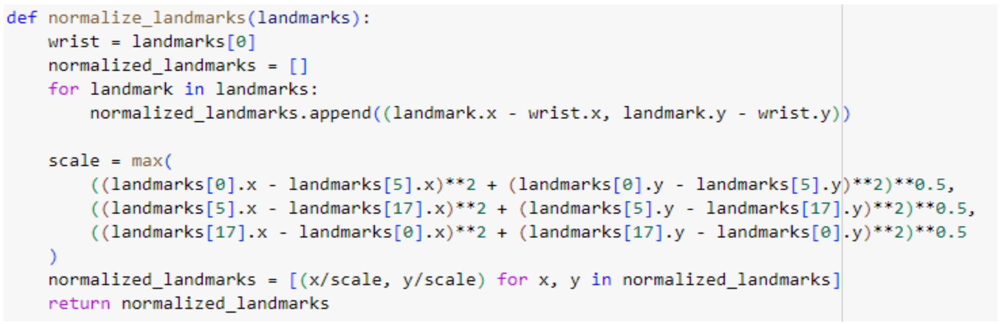
***Note:** TP = True Positive, FP = False Positive, FN = False Negative [10]

2.5.4) Misclassification Backtrack Training

The trained model was then tested in real time and for every frame that the model misclassified, that frame is then stored and later inputted into the total training set to increase real-time accuracy.

2.5.5) Original Issues

The very first model to test whether mediapipe could classify which finger was being pressed down achieved an accuracy of approximately 90%, likely due to the controlled conditions under which the data was collected (same person, environment, and lighting) with only roughly 200 data images per finger (only one person's image data). After collecting data images that contained a different person, lighting, and background, with roughly the same number of data images per finger, the updated model after training achieved an accuracy of around 94% on both subjects. However, when testing the model in real time classifying, there were discrepancies with the results that were given. Due to inconsistencies with relative data points and how data points were processed during the training process and the live testing process, depending on which set of photos the model was trained on, the model was only producing one output, regardless of the input photo that was on the live screen.



```python
def normalize_landmarks(landmarks):
    wrist = landmarks[0]
    normalized_landmarks = []
    for landmark in landmarks:
        normalized_landmarks.append((landmark.x - wrist.x, landmark.y - wrist.y))

    scale = max(
        ((landmarks[0].x - landmarks[5].x)**2 + (landmarks[0].y - landmarks[5].y)**2)**0.5,
        ((landmarks[5].x - landmarks[17].x)**2 + (landmarks[5].y - landmarks[17].y)**2)**0.5,
        ((landmarks[17].x - landmarks[0].x)**2 + (landmarks[17].y - landmarks[0].y)**2)**0.5
    )
    normalized_landmarks = [(x/scale, y/scale) for x, y in normalized_landmarks]
    return normalized_landmarks
```

Figure 4: Python code used to normalize all key points relative to the wrist key point.

As shown in Figure 4; to fix these issues, each datapoint was normalized relative to the wrist key point (index 0 on the key points list) so that the wrist coordinate would always stay at (x, y) = (0, 0) and all other key points would be relative to that coordinate axis. The data points were further normalized to stay within the range of 0 to 1. This method ensures consistency in data processing throughout both the training and live classification stages and also makes the SVM work better. The testing set was carefully split and managed (see section 2.5.2) through the test_train_split() feature so that no model would be trained on any of the image data in the testing set to ensure consistency throughout the analyzing phase.

3) Results

As shown through Table 1, every round of more data inputs significantly increased the overall F1 score as well as improving each individual class's precision and recall. Each class's individual precision and recall was retrieved from testing the classifier on the total test set. Then, the F1 score for that class is calculated through the given formula and the overall classifiers F1 score is calculated by averaging the individual class's F1 scores. Figure 5 shows the resulting confusion matrix for each of the individual classes for each classifier, showing that each class individually improves for every iteration of training.

Table 1: Precision, Recall, and F1 Score (P, R, and F respectively) with respect to different classifiers trained on different datasets.

Data Set → Total Images →		Data on 1st Person (200 Per Class)	Data on 1st and 2nd person (400 Per Class)	Add in first cycle of misclassification	Add in second cycle of misclassification	Add in third cycle of misclassification
Class Open	P	0.55	0.80	0.88	0.92	0.93
	R	0.85	0.93	0.86	0.94	0.97
	F	0.67	0.84	0.87	0.93	0.95
Class Index	P	0.42	0.79	0.87	0.94	0.94
	R	0.56	0.89	0.93	0.94	0.93
	F	0.48	0.84	0.90	0.94	0.94
Class Middle	P	0.77	0.70	0.93	0.95	0.96
	R	0.35	0.75	0.94	0.92	0.94
	F	0.48	0.73	0.94	0.93	0.95
Class Ring	P	0.55	0.94	0.96	0.97	0.98
	R	0.67	0.68	0.96	0.96	0.96
	F	0.60	0.79	0.96	0.97	0.97
Class Pinky	P	0.97	0.90	0.93	0.97	0.96
	R	0.47	0.86	0.92	0.93	0.96
	F	0.63	0.88	0.92	0.95	0.96
Macro-Averaged F1 Score:		**0.57**	**0.82**	**0.92**	**0.95**	**0.95**

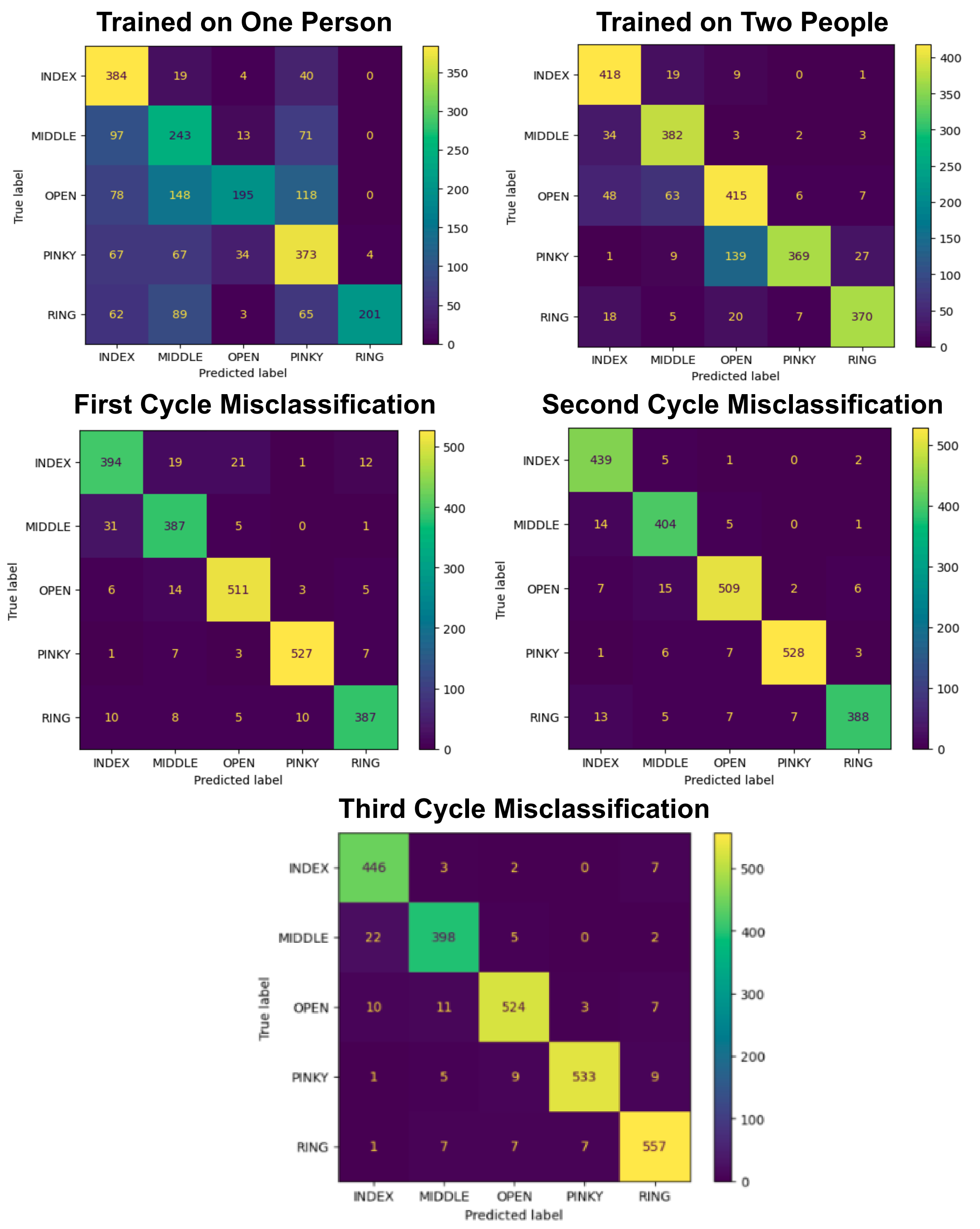


Figure 5: Confusion matrices that show the improvement of each specific class as iterations of training go on, the main diagonal clearly increases in accuracy while the remaining components of the matrix slowly decrease.

Furthermore, to ensure consistency, each model was never trained on any of the image data in the total testing set so that all of the image data in the total testing set is new to each model. The first model that was only trained on one person's data achieved a F1 score of 0.57 and performed poorly during the live testing. Through the iterations of testing by getting an additional person for more data and then running the misclassification backtracking cycle, the final model achieved a F1 score of 0.95 and significantly improved during live testing.

Therefore, through these results, it is clear that the more data that the model is trained on, the more accurate the overall model will be on a more generalized set of inputs. Although these results were only gathered from two people, this shows that this method could be generalized to a wider spread of individuals given the time and resources needed, and acts as a proof of concept for this method of classifying violin finger positions through webcam based images.

4) Discussion

Perspective and camera angle are crucial in training and classifying images for hand position recognition because they directly affect the visibility and clarity of key features, such as finger placement and hand orientation. Variations in camera angle can lead to inconsistencies in the data, making it challenging for the model to accurately interpret hand positions across different perspectives. This variability can be a significant limitation, as ensuring the model performs well with every possible camera angle requires extensive data collection and training, which could be time-consuming and difficult.

Figure 6 is an example of an image being misclassified, you can see that the middle and index fingers are not fully visible to the camera. The index finger is almost completely blocked by the middle finger and the end of the middle finger is blocked by the ring finger. Because of this, it is reasonable that because mediapipe has to “guess” where those remaining keypoints are, the classifier could easily misclassify from those guesses.

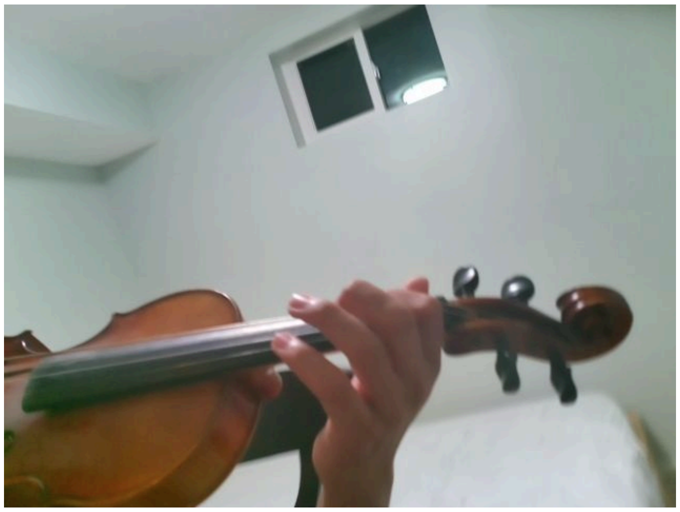


Figure 6: Image that was misclassified - classified as a middle, should have been index

Classifying musical expressions such as vibrato, pizzicato for the right hand, and other nuanced gestures like glissando presents unique challenges due to their subtle and dynamic nature. These expressions involve intricate movements that can vary greatly in speed, intensity, and style, making them more complex to detect and classify accurately using image recognition techniques. While these elements are crucial for capturing the full range of musical expression, they were not considered as variables for this paper. Focusing on the fundamental task of fingering classification first lays the groundwork for future studies to explore these more advanced aspects of musical performance.

Future research could explore the scalability of this approach by incorporating a more diverse dataset, including different lighting conditions, backgrounds, and hand sizes. While this study focused primarily on static fingering classification, extending the model to support live playing would be a significant advancement. However, more work is needed to handle the complexities of real-time performance, such as quickly adapting to dynamic changes and maintaining high accuracy. Additionally, there is a wide range of exploratory work that could be pursued, such as determining which string the fingers are on and analyzing the right-hand movements to identify note values and rhythms.

Furthermore, integrating hand position detection could enhance the system's ability to classify different playing positions on the violin. By examining the wrist's location relative to the tuning pegs and calculating distances, the model could potentially identify the position on the fingerboard. Another exciting area for future research involves using right-hand motion to estimate volume based on the speed of bowing, which could add an expressive dynamic element to the virtual instrument.

Although this project did not achieve a complete virtual instrument capable of replicating every aspect of violin playing, it lays the foundation for ongoing development in this area. There remains a clear path forward to refine and expand the system, bringing us closer to a comprehensive virtual instrument that can fully emulate the nuances of live performance.

5) Conclusion

In conclusion, this research demonstrates the potential and limitations of using Mediapipe and support vector classifiers for detecting violin fingering positions to play a virtual instrument. The study initially achieved a promising accuracy of approximately 90% under controlled conditions. By expanding the dataset and iterating through cycles of misclassification backtracking, the model's performance significantly improved, achieving a final F1 score of 0.95. This research contributes to the field of AI-driven music technology by offering insights into practical challenges and opportunities associated with image recognition in musical applications. While the study focused on a specific application of detecting violin fingering, the methodology and findings can be generalized to other types of hand movements and virtual instrument interfaces.

6) References

[1] Westermann, Dirk, and Johann M. Marz. “Understanding Wind Turbine Condition Monitoring Systems.” Condition Monitoring, 2014, pp. 13–44. Springer, doi:10.1007/978-3-319-12093-5_2.

[2] Stern, Frederick. “Evaluation of System Identification for Operational Modal Analysis of Naval Structures.” MIT DSpace, Massachusetts Institute of Technology, 2023, dspace.mit.edu/bitstream/handle/1721.1/150654/3544549.3585838.pdf?sequence=1&isAllowed=y.

[3] Gil, Manuel, et al. “Applications of UAVs in Forest Fire Management: A Systematic Review.” Journal of Imaging, vol. 6, no. 8, 2020, p. 73, doi:10.3390/jimaging6080073.

[4] Goodfellow, Ian, et al. Deep Learning. MIT Press, 2016, www.deeplearningbook.org.

[5] Verbeke, Mathias, et al. “The Electronic Tabla Controller.” ResearchGate, 2011, www.researchgate.net/publication/221164928_The_Electronic_Tabla_Controller.

[6] Lugmayr, Artur, et al. “MediaPipe: A Framework for Building Perception Pipelines.” arXiv, 19 June 2019, arxiv.org/abs/1906.08172.

[7] Lugaresi, C et al. “Mediapipe: A framework for building perception pipelines.” arXiv preprint, 2019, https://arxiv.org/abs/1906.08172

[8] Cortes, C., & Vapnik, V. Support-vector networks. *Machine Learning,* 1995, *20*(3), 273-297. doi:10.1007/BF00994018.

[9] The AIGuysCode. “Colab Webcam.” GitHub, 2020, github.com/theAIGuysCode/colab-webcam/blob/main/colab_webcam.ipynb.

[10] Evidently AI. “Confusion Matrix: Classification Metrics.” Evidently AI, 2020, www.evidentlyai.com/classification-metrics/confusion-matrix.