Exploring Coding with AI for MathDance

Axel Itech Apolonio Flores*, Guest Student; **Maria Fernanda Pichardo Perez***, Guest Student;
CJ Chung, PhD, Professor, College of Arts and Sciences

Lawrence Technological University and (*)Universidad Autónoma del Estado de México

INTRODUCTION

This research project embarks on a journey at this intersection, leveraging Python programming, deep learning techniques, and interactive feedback mechanisms to pioneer an educational mini-game titled "Math Dance [5, 6, 7]." The primary objective is to revolutionize mathematics learning by enabling real-time recognition of basic mathematical function graphs through body gestures captured by a camera.
Traditional methods of teaching mathematics often rely on static representations, which may not fully engage students or cater to diverse learning styles. By harnessing the power of technology, specifically through Python programming and deep learning algorithms, this project seeks to create a dynamic and interactive learning environment. The "Math Dance" mini-game encourages students to actively participate in the learning process, offering a hands-on approach that enhances comprehension and retention of mathematical concepts.

The overarching goals of this research endeavor are multifaceted:

1. Develop a robust system capable of accurately recognizing and interpreting basic mathematical functions, such as constant, linear, negative linear, square, and square x + square y, based on user body gestures.
2. Implement an interactive feedback mechanism utilizing Text-to-Speech (TTS) technology to provide real-time positive reinforcement and guidance during the learning process.
3. Cultivate a dynamic learning environment that fosters active engagement and exploration of fundamental mathematical concepts, promoting a deeper understanding and appreciation for mathematics among students.
4. By achieving these goals, this research aims to not only enhance mathematics education but also pave the way for future developments in interactive and immersive learning technologies.

DATA COLLECTION

Data Collection Equipment:
The data collection process involved using the webcam of a laptop placed on a desk. We utilized Teachable Machine [1] to capture photos of individuals, focusing on the head and torso of the person.

Data Collection Environment:
The images were taken of various individuals in different settings such as a classroom, bedroom, etc., each person assuming different positions for each mathematical function. Additionally, they varied their distance from the camera and moved from left to right to achieve improved results with the model.

Data Classification and Class Settings:
The collected dataset was organized into seven distinct classes, each representing different mathematical functions or scenarios captured by the camera. These classes were used to train the model for multi-class classification. The class settings are outlined as follows:

Class 1: None
no mathematical function gesture is detected
Class 2: 0
Showing the gesture representing the constant function (0)
Class 3: x
Showing the gesture for the linear function (x)
Class 4: minus x
Showing the gesture for the negative linear function (-x)
Class 5: square x
Showing the gesture for the square function (x^2)
Class 6: tangent of x
Showing the gesture for the tangent function (tan(x))
Class 7: square x plus square y
Showing the gesture for the function ($x^2 + y^2$)

Dataset Info	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7
Total Data Size	272	452	381	452	512	461	539

Table 1. Dataset Information

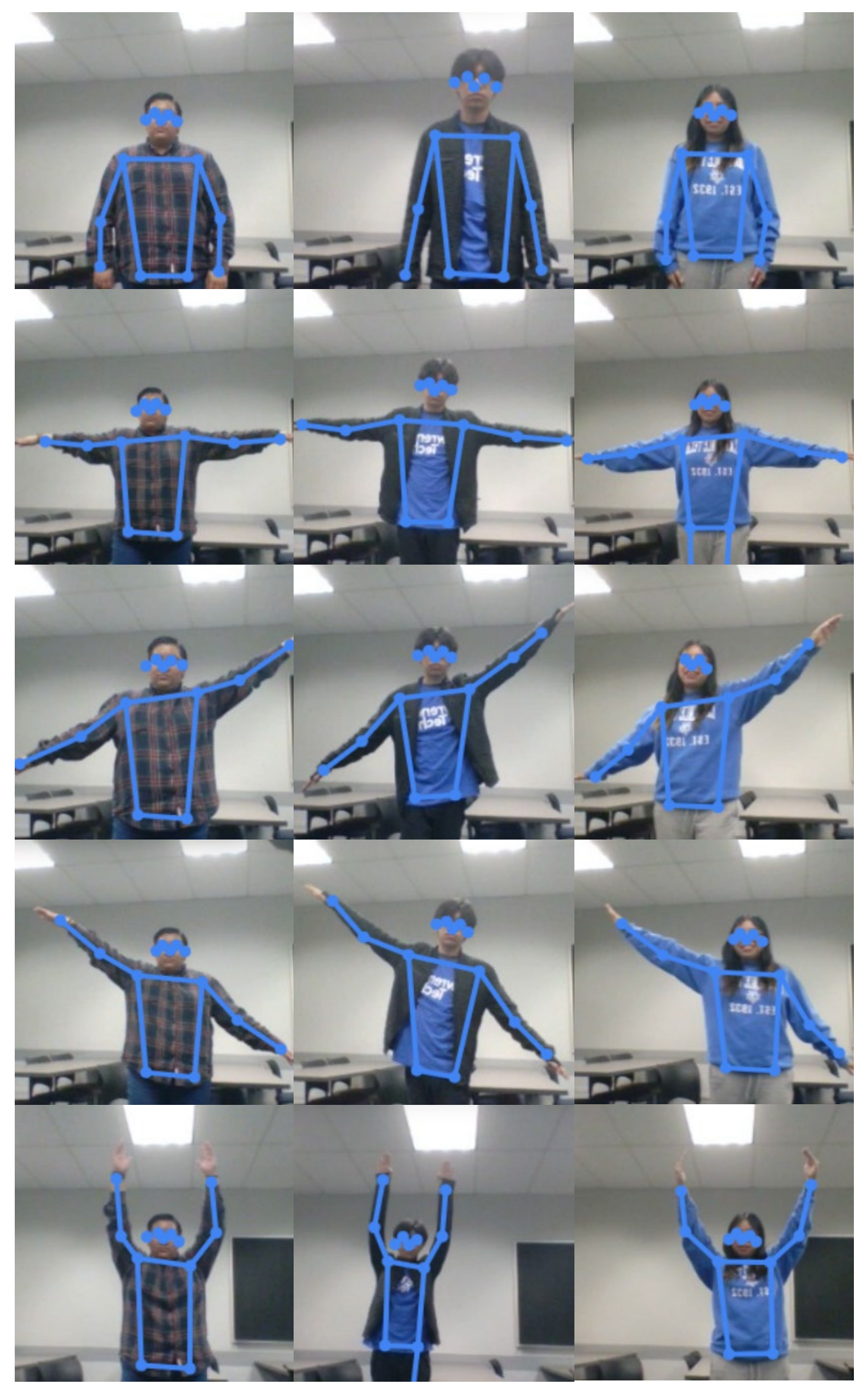


Figure 1. Pose Examples

DESIGN

During the model experimentation phase of the "Math Dance" mini-game, a comprehensive exploration was conducted involving the development of distinct models, each contributing uniquely to the mini-game's evolution and refinement.

The initial model creation commenced with Scratch code, serving as a foundational prototype to delve into basic gesture recognition concepts. However, recognizing the need for more advanced functionality and enhanced accuracy, the project progressed to a more sophisticated stage. This transition involved translating the Scratch code into Python, a decision propelled by the desire to incorporate an image model for gesture recognition. This pivotal shift enabled a deeper exploration of the intricacies involved in recognizing and interpreting user gestures accurately.

However, as the project delved deeper into leveraging advanced models, a significant challenge emerged. The exploration of integrating the pose model within Teachable Machine was initiated but encountered a roadblock. Teachable Machine, unfortunately, restricted its pose model implementation solely to JavaScript, presenting a compatibility issue with the Python-based project structure.

Undeterred by this obstacle, a third model iteration was embarked upon, this time employing JavaScript to harness the capabilities of the pose model for gesture recognition. This approach aimed to leverage the pose model's nuanced understanding of body movements for enhanced accuracy in recognizing mathematical gestures.

Nevertheless, as the project progressed and the desired gameplay dynamics took shape, another hurdle surfaced. The envisioned gameplay scenario necessitated precise timing functionalities to measure the completion time of tasks. Herein lay the crux of the challenge: the integration of the pose model with JavaScript led to periodic page refreshes to load webcam images, resulting in inadvertent time resets and disrupting the seamless flow of the game.

To surmount this intricate challenge and ensure the project's success, a strategic decision was made to revert to the image model within Teachable Machine. This pivot proved pivotal as it provided a stable and reliable solution for gesture recognition without compromising the critical timing functionalities integral to the gameplay experience.

In essence, the model experimentation journey encapsulated a dynamic evolution, transitioning from foundational Scratch code to advanced Python image models and exploring the complexities of pose models in JavaScript. This iterative process underscored the project's commitment to innovation, adaptability, and the relentless pursuit of optimal solutions to enhance the "Math Dance" experience for users.

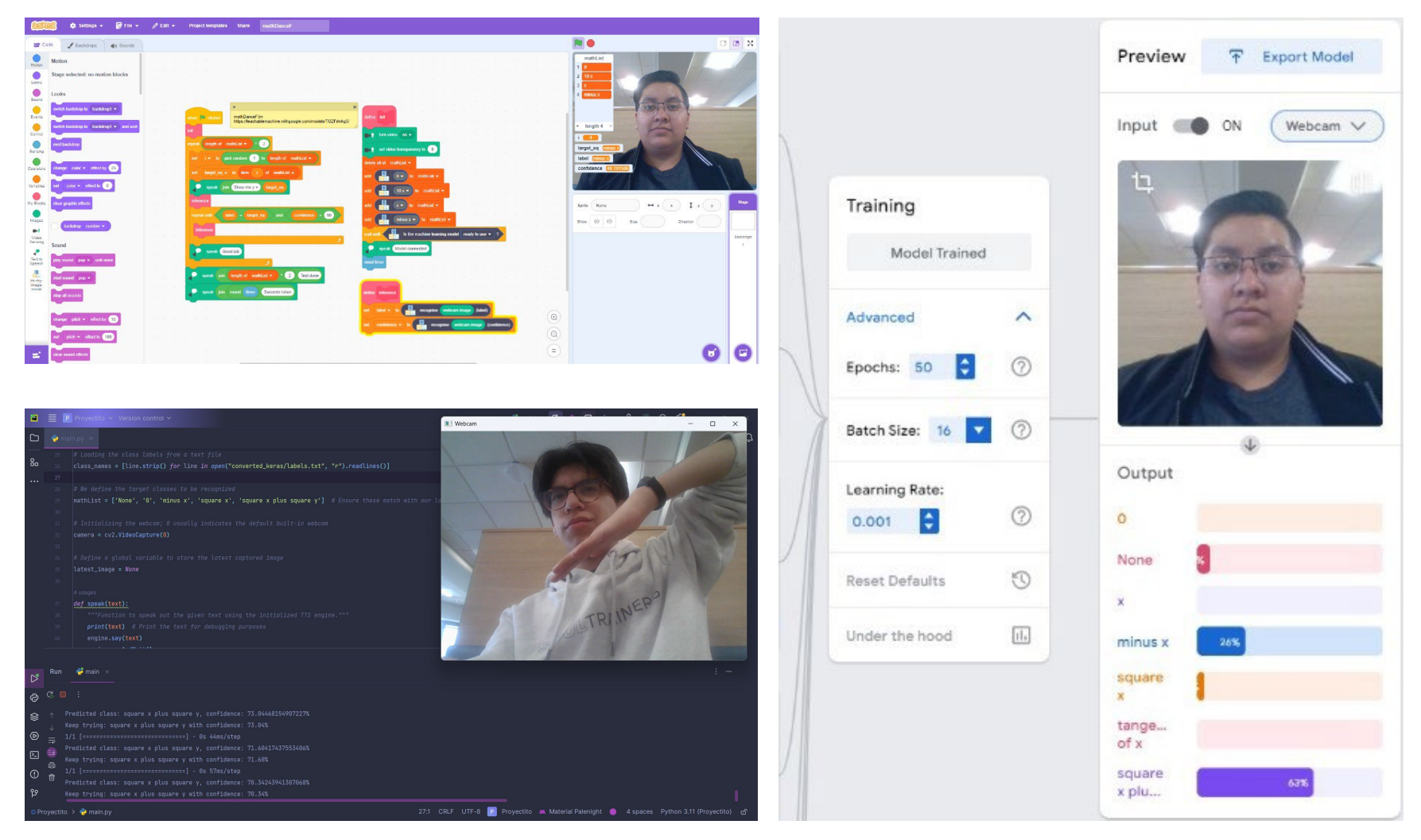


Figure 2. Use of other tools

Experimental RESULTS

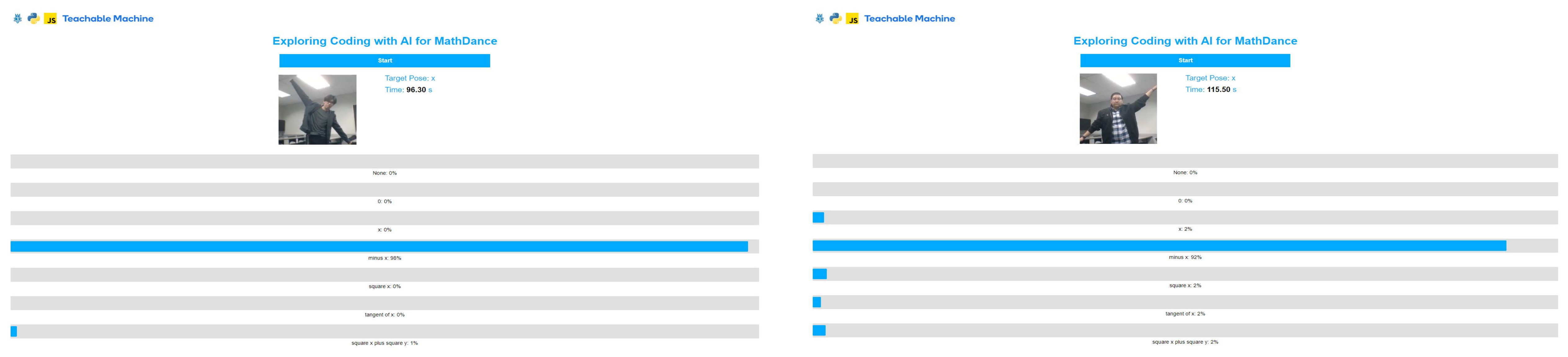


Figure 3. Results of the mini-game using image model and JavaScript

Figure 3 shows the interface in JavaScript, using the image model. This model was used due to the high accuracy obtained, as well as for its ability to adapt to projects with time measurement, as was the case of this project. The interface captures a snapshot of the model in action, indicating the application's recognition of a particular pose labeled in the interface with a confidence score. The pose is likely predefined in the system to correspond to the mathematical concept of the functions and associated with a particular body movement or position. This result illustrates the application's use of Teachable Machine's real-time pose classification capabilities to engage users in an interactive learning experience that combines physical activity with mathematical concepts.

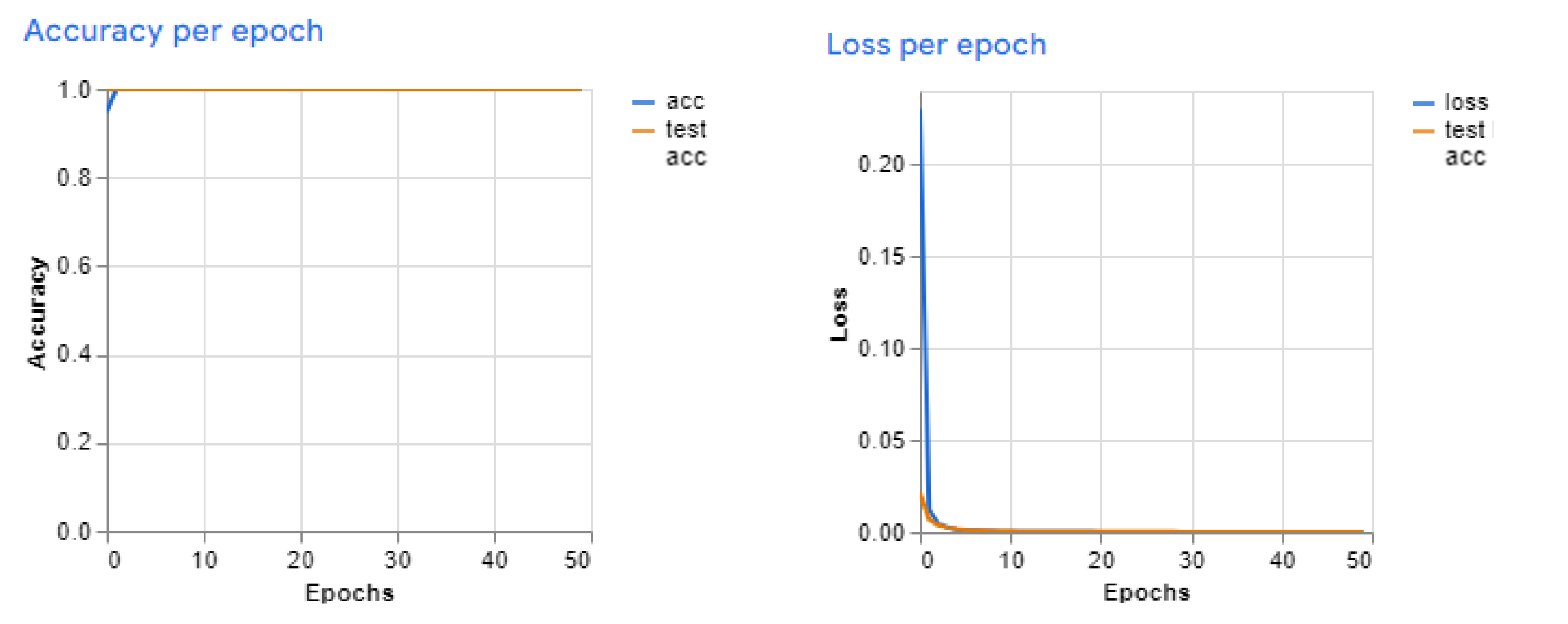


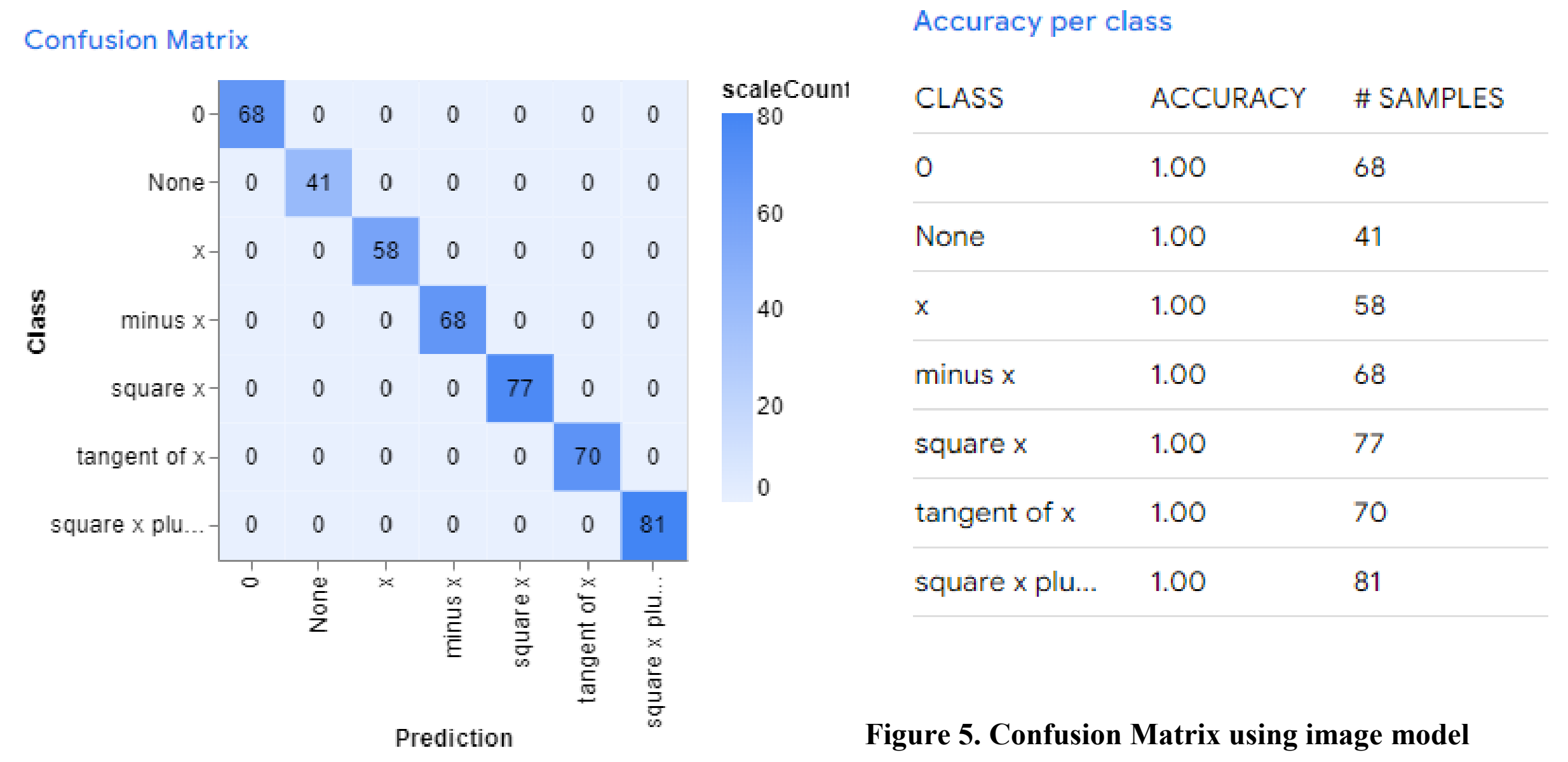
Figure 4. Accuracy and Loss per epoch using image model

Accuracy:
Figure 3 in our results presents the 'Accuracy per Epoch' graph, which is instrumental in understanding the model's capability to correctly classify the dataset over time. The graph plots two lines that represent the accuracy of predictions made on the training data ('acc') and the validation data ('test acc'), respectively. The consistent positioning of these lines at the zenith of the graph, near the maximum possible accuracy of 1.0, demonstrates the model's exceptional performance from the outset. This rapid achievement of high accuracy, maintained throughout the training epochs, suggests that the underlying model has quickly learned the distinguishing features necessary for accurate predictions and that the pretraining on a related task has likely contributed significantly to this swift convergence.

Loss:
Complementing the accuracy trends, Figure 3 displays the 'Loss per Epoch', providing insights into the model's learning efficiency. Both the training loss ('loss') and the validation loss ('test loss') exhibit a precipitous decrease during the initial epoch, indicating an effective gradient descent process that sharply corrects predictive errors. Subsequent epochs show the loss values flattening out close to zero, a sign that the model has reached an optimal state with minimal error. These loss metrics not only reinforce the model's accuracy but also hint at a stable and robust learning process that resists overfitting, despite the model's rapid learning curve.

Confusion Matrix:
The confusion matrix provided in Figure 4 offers a comprehensive breakdown of the classification model's predictive performance across the test dataset. Each row within the matrix delineates the instances of the true classes, while the columns correspond to the predicted classes by the model. The principal diagonal of the matrix, running from the top-left to the bottom-right, enumerates the quantity of correct classifications for each respective class. Notably, the non-diagonal elements, which would typically represent the instances of misclassification, all contain zeros. This indicates an absence of prediction errors across all classes.
The categories, presumably indicative of distinct mathematical functions or gestures, are labeled as 'None', 'x', 'minus x', 'square x', 'square x plus square y', among others. The values within the diagonal cells are substantially high, ranging between 41 to 81, which signifies that the model has unerringly classified all samples within the test set.

Accuracy per Class Analysis:
Complementing the confusion matrix, the accuracy per class data depicted in Table [Y] conveys the precision of the model for each class, alongside the count of samples per class. The accuracy is expressed as a fraction, with 1.00 denoting flawless precision. According to the data, the model achieved an exemplary accuracy score of 1.00 for each class, demonstrating impeccable classification capabilities.
Furthermore, the '# SAMPLES' column indicates the number of test instances for each class, showcasing variance in sample size. Despite the potential imbalance in the dataset, the model has managed to maintain a uniform accuracy of 100% across all classes.



Class \ Prediction	0	None	x	minus x	square x	tangent of x	square x plu...
0	68	0	0	0	0	0	0
None	0	41	0	0	0	0	0
x	0	0	58	0	0	0	0
minus x	0	0	0	68	0	0	0
square x	0	0	0	0	77	0	0
tangent of x	0	0	0	0	0	70	0
square x plu...	0	0	0	0	0	0	81

CLASS	ACCURACY	# SAMPLES
0	1.00	68
None	1.00	41
x	1.00	58
minus x	1.00	68
square x	1.00	77
tangent of x	1.00	70
square x plu...	1.00	81

Figure 5. Confusion Matrix using image model

SUMMARY and DISCUSSION

In the context of JavaScript, the concurrent utilization of a "chronometer" to measure the temporal duration of test execution alongside the use of a pose model poses a challenge. This issue arises from the inherent complexities associated with synchronizing time updates during camera feed processing and concurrent prediction operations. Despite thorough investigation into the matter, it was observed that the predictive accuracy achieved through the use of the image model surpassed that of the pose model. Unfortunately, the challenges posed by Teachable Machine's invocation of the script cannot be directly resolved due to the nature of the JavaScript language.

The "Math Dance" project represents a groundbreaking convergence of advanced technologies and educational methodologies, leveraging Python programming, deep learning algorithms, and real-time webcam-based gesture recognition to revolutionize the realm of mathematics education. This multifaceted initiative aims to create an immersive and interactive learning experience that transcends traditional classroom boundaries, empowering learners with hands-on exploration and immediate feedback mechanisms.

Central to the project's success is the development of a sophisticated dataset comprising seven distinct classes, meticulously crafted to encompass a wide spectrum of fundamental mathematical functions. These classes, ranging from constants to complex expressions like square x plus square y, serve as the backbone for training the system to accurately interpret user gestures captured by a webcam in real-time.

Through an intricate process of data collection and model training, the system has been fine-tuned to seamlessly translate gestures into mathematical functions, providing users with instant visual and auditory feedback. The integration of Text-to-Speech (TTS) technology adds an extra dimension of interactivity, enabling the system to deliver personalized responses and guidance tailored to each user's progress and needs. Furthermore, the "Math Dance" project embraces diversity and variability by incorporating a wide array of backgrounds, distances, and gestures during data acquisition. This comprehensive approach ensures that the system is robust, adaptable, and capable of delivering consistent performance across different environmental conditions and user interactions.

Ultimately, the "Math Dance" project stands as a testament to the transformative potential of technology-enhanced learning experiences. It exemplifies innovation in education, fostering engagement, critical thinking, and confidence-building in learners of all ages. As technology continues to evolve, initiatives like "Math Dance" pave the way for a future where interactive and personalized learning experiences are the norm, revolutionizing the way we approach mathematics education and beyond.

FUTURE WORK

The "Math Dance" project opens avenues for exciting future developments and enhancements that can further enrich the learning experience and expand the system's capabilities. Some potential areas for future work include:

- Advanced Gesture Recognition: Enhancing the system's gesture recognition capabilities to include more complex gestures or expressions, allowing for a broader range of mathematical functions and interactions.
- Multi-User Support: Implementing multi-user support to enable collaborative learning experiences, where multiple users can engage with the system simultaneously, fostering teamwork and peer learning.
- Real-Time Collaboration: Introducing real-time collaboration features that enable users to collaborate on solving mathematical problems or exploring concepts together, enhancing engagement and interactivity.
- Personalized Learning Paths: Developing algorithms to analyze user performance data and create personalized learning paths tailored to each user's strengths, weaknesses, and learning preferences.
- Expanded Functionality: Incorporating additional mathematical concepts, such as trigonometric functions, logarithms, or calculus, to provide a comprehensive learning experience covering a wide range of mathematical topics.
- Integration with Learning Management Systems: Integrating the "Math Dance" system with existing learning management systems (LMS) or educational platforms to seamlessly integrate interactive mathematics learning into formal education settings.
- Accessibility Features: Implementing accessibility features, such as voice commands or alternative input methods, to ensure inclusivity and accommodate users with diverse learning needs or disabilities.
- Gamification Elements: Introducing gamification elements, such as challenges, rewards, and leaderboards, to enhance motivation, engagement, and long-term retention of mathematical concepts.

By exploring these future directions and incorporating advancements in technology and pedagogy, the "Math Dance" project can continue to evolve as a dynamic and impactful tool for mathematics education, offering innovative solutions to enhance learning outcomes and student engagement.

REFERENCES

1. Teachable machine. (2024). Retrieved from https://teachablemachine.withgoogle.com/
2. Googlecreativelab. (2019). Teachablemachine-community/libraries/image. GitHub. Retrieved from https://github.com/googlecreativelab/teachablemachine-community/tree/master/libraries/image
3. TensorFlow.js: Make your own 'Teachable Machine' using transfer learning with TensorFlow.js. (2022). Google Codelabs. Retrieved from https://codelabs.developers.google.com/tensorflowjs-transfer-learning-teachable-machine#0
4. OpenAI. (2024). ChatGPT [Computer software]. https://openai.com/
5. C. J. Chung, Move Your Body, Animate the Movement, and Learn Computer Science, https://www.robofest.net/2018/CSPAA.pdf
6. C. Chung and M. Kocherovsky, "CS+PA2: Learning computer science with physical activities and animation — A MathDance experiment," 2018 IEEE Integrated STEM Education Conference (ISEC), Princeton, NJ, USA, 2018, pp. 262-267, doi: 10.1109/ISECon.2018.8340497.
7. Chan-Jin Chung and Lior Shamir, "Introducing Machine Learning with Scratch and Robots as a Pilot Program for K-12 Computer Science Education," International Journal of Learning and Teaching, Vol. 7, No. 3, pp. 181-186, September 2021. doi: 10.18178/ijlt.7.3.181-186