Interactive Physics Experiments using Gesture Recognition

Constantinescu Mario-Cristian
Ovidius University of Constanța
University alley no. 1, Constanța, Romania
mario.constantinescu@365.univ-ovidius.ro

Ionescu Anata-Flavia
Ovidius University of Constanța
University alley no. 1, Constanța, Romania
anata.ionescu@365.univ-ovidius.ro

ABSTRACT

In this paper we propose a web application which provides virtual and interactive physics experiments for secondary school students. The application allows users to perform actions such as creating and manipulating objects in the virtual experiments using both mouse/keyboard and hand gesture input. Within each experiment, supported actions are associated with default gestures, but each user is allowed to customize his/her interaction with the experiment by permuting the associated gestures.

Gesture recognition is performed using a deep neural network classifier with 18 classes of gestures. The classifier is pretrained on a custom dataset of 368 samples. Each sample is a set of 21 hand landmarks extracted using Google's MediaPipe Hands framework. Users are also allowed to fine-tune the weights of the model by adding samples of their own gestures for the supported classes.

The application has the potential of enhancing the quality of experience of students, offering customizable multimodal interaction and the possibility to create affordances. We conclude the paper with a discussion of the contributions, limitations and future research directions.

Author Keywords

virtual learning; gesture recognition; neural network; interactive simulation; physics; motivation; self-paced learning; MediaPipe.

ACM Classification Keywords

H.5.2 User Interfaces; D.2.2 Design Tools and Techniques; I.5.4 Applications •Human-centered computing~Human computer interaction (HCI)~Interaction techniques~Gestural input
•Applied computing~Education~Computer-assisted instruction
•Applied computing~Education~E-learning.
•Human-centered computing~Accessibility~Accessibility technologies

General Terms

Human Factors; Design.

DOI: 10.37789/ icusi.2024.17

INTRODUCTION

Keeping students motivated may be a challenge for physics teachers. This paper reports on the development of a web- based educational software system which provides interactive virtual physics experiments to students.

A growing body of research demonstrates that the use of computers and modern technology motivates students to learn physics. Active learning methods in general, and the ones based on Information and Communication Technologies (ICT) in particular, have been shown to enhance students' interest in physics [5]. If properly integrated within the curriculum and used as a complement for traditional instruction methods, taking into account their interplay with other pedagogical factors, computer-based experiments have been proven to benefit students [14].

Multimodality may be a key feature for the efficiency of interactive learning systems. In particular, gesture control has been proven to have a positive impact on student intrinsic motivation and, through it, on cognitive and affective learning outcomes [16].

Virtual physics simulations can help students understand physical phenomena better than by simply reading theoretical explanations and drawings. Students are given the opportunity to run a virtual experiment as many times as they want and need for a complete and correct understanding of the phenomenon. The solution described in the current paper addresses the need for attractive multimedia content which stimulates active learning through interactive features.

The remainder of the paper is structured as follows. The Related Work section provides an overview of existing solutions in the field. In the Methodology section, we report on the design and implementation of the proposed software architecture, including the neural model architecture for gesture recognition. In the Evaluation section, we present the testing dataset and model performance. The paper ends with a Conclusions section, which reviews the main contributions of the research, followed by a discussion of the limitations of the proposed solution and potential future research directions.

RELATED WORK

There are several similar applications with virtual simulations of physical phenomena. Most of these applications support static file upload of learning materials such as lecture notes, but also include attractive and intuitive GUIs, supporting user interaction using the mouse and keyboard. We offer a brief review of extant solutions in what follows.

Physics at school-HTML5, (https://www.vascak.cz/physicsanimations.php) [10] is a web application with numerous simulations of physical phenomena presented briefly and grouped into categories. The interaction is based on mouse input. The GUI is intuitive, attractive, colorful, with animations that support understanding of the phenomenon. However, in most of the experiments available in this web application, parameters cannot be changed so as for the user to test the effects of such changes.

Physics Applets, by Walter Fendt (https://www.walter-fendt.de/html5/phro/) [2], is a web application that contains simulations of 46 physical phenomena through a simple GUI. These simulations cover the main branches of physics and tackle elementary topics. In most simulations, users are allowed to modify parameter values for experimental purposes. The user is also offered the possibility to check boxes corresponding to experiment parameters he/she wishes to view. The main focus of this app is the math behind the experiments, rather than the attractiveness of the interface.

Physics Education Technology (PHET) Platform (https://phet.colorado.edu/) [11] is a web application that contains simulations of several phenomena, including those related to physics. Each simulation has associated learning objectives and tips for teachers. The objectives refer to what students should learn after completing the experiment, and the teacher tips consist of .pdf files in which teachers are provided with didactic recommendations. The simulations in this application are complex. They receive mouse and keyboard input, through which users are allowed to change the values of experiment-specific parameters. Similar to the previous work in this review, the user is allowed to check which parameters he/she wishes to view. The application allows teachers to create accounts, through which they can download tips on how the phenomenon described by a simulation can be explained in simple terms. The app also has a dedicated section for tips on how it could be used in teaching physics in school classrooms or in distance learning. It has a virtual workshop, with 5 mini-courses in which effective methods for teaching certain learning contents using the application are explained. Also, the graphical interface is attractive, offering suggestive animations that demonstrate physical phenomena.

The Physics Classroom (https://www.physicsclassroom.com/) [18] is an app that contains theory, interactive simulations, student and teacher accounts, and PDF files that teachers can download and print. Theory is grouped by branches of physics and includes a considerable amount of textual information about each phenomenon, backed up by explanatory videos presented by teachers. Students can check their understanding of a phenomenon by answering questions with hidden answers, which are displayed on demand. Interactive simulations have settings through which the parameters of the phenomenon can be changed, as well as buttons to show or hide vectors, where appropriate. Each chapter in the app contains a section that specifies learning objectives. The application supports two types of accounts: teacher and student. Teachers can create classes to later associate their students with those classes, set tasks for their students, and monitor their students' progress. Downloadable PDF files containing details about each lesson, with learning objectives and suggestions for further reading are also available to users. The GUI is attractive and intuitive.

It is also worth mentioning that human interface devices for touchless 3D interaction have sometimes been used for educational purposes. Motion controllers such as Microsoft Kinect [20], discontinued at the time of writing this paper, or the Leap Motion Controller [8] use gesture input. They can recognize complex body motion, but major drawbacks include the fact that they require additional hardware (and inherent costs) and physical space (especially in the case of whole-body motion controllers).

To the best of our knowledge, at the time of writing this paper, there is no platform for virtual physics experiments based on customizable gesture input without specialized hardware. Therefore, an important objective of the application proposed in the present paper is to fill this identified gap. The solution we describe in this paper allows students to control objects in simulations using gesture input based on a fine-tunable neural network classifier. Furthermore, though each simulation has gesture-action pairs associated with it by default, the student is allowed to permute the gestures associated with the actions according to his/her own preferences, through permuting the gestures.

APPLICATION DEVELOPMENT

In this section, we briefly describe the design and implementation of the proposed solution.

Technologies Used

The application was developed using Django [6], which is a Python open source web framework based on the Model-View-Controller (MVC) design pattern. The technologies used for front-end development include HTML5 [7], CSS [12], JavaScript [17], and AJAX [3]. As for the back-end, virtual simulations were mainly based on OpenCV [1] for basic computer vision functions, TensorFlow [4] for creating and training the neural network classifier, and Google's MediaPipe Hands [19] framework.

MediaPipe Hands is a framework for detecting and tracking the user's hands and their landmarks in real time, providing the user with a natural interface without the need for special equipment. The MediaPipe Hands solution consists of two models that are used together to reach the final result:

1. a palm detection model, operating on the entire image to create a bounding box for the palm.
2. a second model that operates on the bounding box output by the first model and determines the positions of the landmarks on the detected palm. This model returns 3 outputs:

(a) 21 landmarks (key points corresponding to hand-knuckles) on the palm, each being described by a triplet of values (x,y,z), where x and y represent the horizontal and vertical position of the landmark, respectively, relative to the dimensions of the bounding box, and z is the distance to the webcam.

(b) A value indicating the probability that a hand is present in the input picture

(c) A binary classification indicating whether the present hand is left or right

The models were trained on datasets that tackle different aspects of the problem:

- A dataset containing 6,000 pictures covering a wide range of conditions such as geographical diversity and different light conditions
- An in-house dataset containing 10,000 images covering angles of a wide range of physically possible hand gestures
- A synthetic dataset with 100,000 pictures created based on a synthetic hand model

Custom Dataset Creation

For the proposed application, we created a small custom dataset consisting of 548 samples collected from the first author and two volunteers. The dataset was made publicly available at https://github.com/corsimar/dataset. The samples were split randomly into 368 training samples and 180 testing samples.

The dataset was created using a GUI which allowed the user to choose the hand and gesture for which he/she wished to add a sample to the dataset. The user was then given a preset number of seconds to perform the gesture in front of the webcam, while the application displayed the webcam input with the 21 key points calculated by MediaPipe marked on it. Upon timeout, if a hand was detected in the last video frame (see Figure 1), then the user was allowed to either save or retry the gesture. Saving the gesture involved saving a record consisting of the 3D coordinates of the 21 key points and the class index for the gesture in a CSV file.

The application supports 18 different classes of gestures that can be made to perform actions in the simulations, 9 for each hand (left and right). These are shown in Table 1.

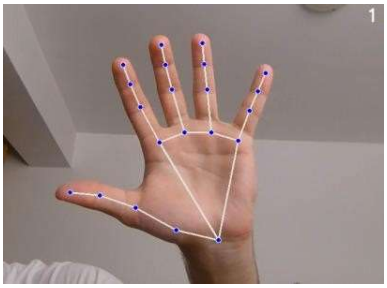

Figure 1. A video frame extracted from the webcam, with hand landmarks obtained with MediaPipe Hands framework

Index	Hand	Gesture
0	Left	Closed fist
1	Left	Index down
2	Left	Index and middle up
3	Left	Index pointing to the right
4	Left	Index up
5	Left	Thumb and index forming a circle and the other fingers up (OK)
6	Left	Open palm
7	Left	Thumb down
8	Left	Thumb up
9	Right	Closed palm
10	Right	Index down
11	Right	Index pointing to the left
12	Right	Index and middle up
13	Right	Index up
14	Right	Thumb and index forming a circle and the other fingers up (OK)
15	Right	Open palm
16	Right	Thumb down
17	Right	Thumb up

Table 1: The 18 classes of gestures in the dataset

The Neural Network Classifier

To choose the best neural model architecture we used KerasTuner [13], a hyperparameter optimization framework which allows the user to define search spaces for hyperparameter values and automatically explore possible combinations to find the best values for the model to be optimized. We defined search spaces for the following hyperparameters:

- The number of fully-connected hidden layers - between 3 and 7
- The number of neurons on each fully-connected layer - between 32 and 512, with a step of 32
- The activation function for each fully-connected layer: ReLU, Leaky ReLU, or tanh
- The optimizer: Adam or SGD
- The presence of a batch normalization layer [15] after each fully-connected hidden layer.

Due to the very large number of possible combinations of hyperparameter values, exhausting the search space would have required a lot of time and computational resources. Therefore, KerasTuner was set to run only 100 trials and used only to confirm assumptions based on theory or isolated empirical results, as well as to suggest suboptimal values for certain hyperparameters, and the model will be refined also based on known results.

The results of the experiments with KerasTuner clearly indicated the tanh activation function for all fully-connected hidden layers and the Adam optimizer as generating superior performances compared to the competing activation functions and optimizers, respectively. Also, the results suggested that models with a smaller number of fully-connected hidden layers (4 or even 3 pairs of fully- connected layers, each followed by a batch normalization layer) tended to achieve higher performances. Furthermore, using more parsimonious models with fewer parameters is expected to reduce the risk of overfitting and lead to better performance in terms of average inference time, which is important for the user experience.

Regarding the number of neurons on each layer, the results were less conclusive, varying substantially between the best models obtained with KerasTuner. As such, for the number of neurons on each layer, a common scheme of halving them from one layer to another (256 - 128 - 64) was adopted, assuming that the shallower layers extract many simpler features, and the deeper layers combine them into a progressively smaller number of complex, implicit features.

The learning rate was set to the default of 0.001.

The final architecture of the model, shown in Figure 2, has 63 neurons on the input layer because the size of the input data is 63 (21 3D landmark coordinates), and the output layer has 18 neurons, because the model classifies 18 different classes of gestures. The model was trained for trained for 50 epochs, because beyond this value the test loss started to increase and the test accuracy started to decrease, indicating that the model was starting to overfit.

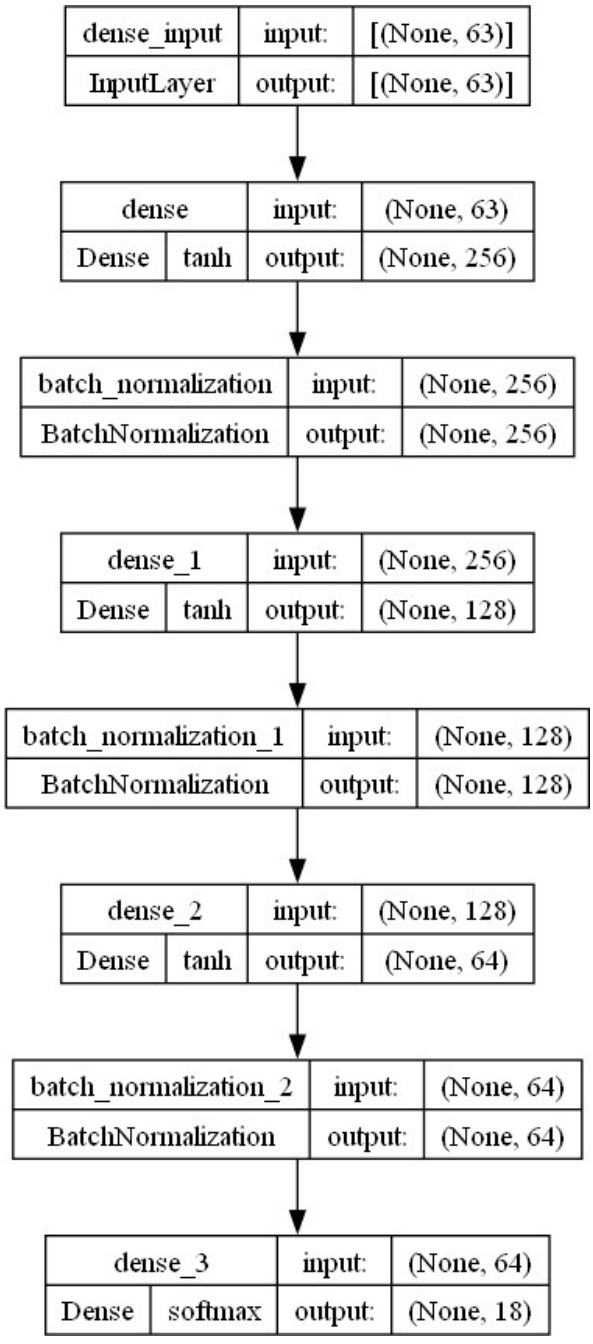

Figure 2: The final architecture of the gesture classifier

Fine-Tuning by the User

For users who feel dissatisfied with the gesture classifier's performance for their own gestures, the application was also designed to offer the possibility to adjust the weights of the neural network used for gesture recognition through a dedicated

fine-tuning module, presented in Figure 3. By accessing this module, the user should be able to use the webcam to add samples of his/her own gestures for the classes supported by the application, and the application fine-tunes the model for gesture recognition, starting from the weights of the base model, adjusted by further training for that user's gestures. This can be useful for improving the classifier's accuracy for each individual user.

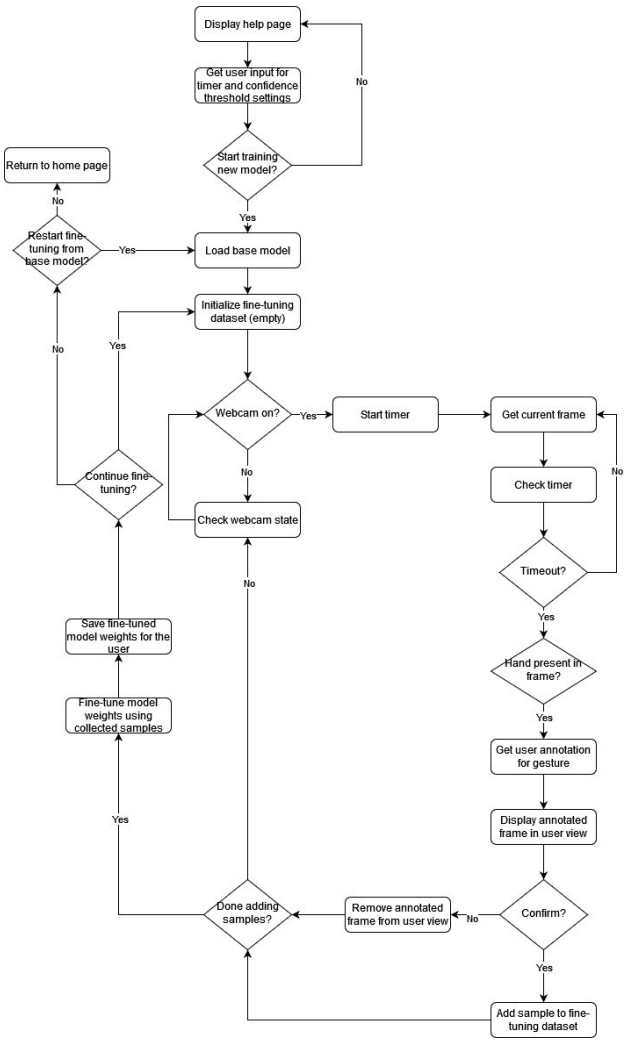

Figure 3. Fine-Tuning Flowchart

Simulations

Four physics simulations were implemented:

- Lever (Figure 4.a), an experiment which allows the user to place objects on a lever, with the goal of keeping the tray balanced
- Density (Figure 4.b), an experiment in which objects are created with only 2 known variables out of the following 3: mass, volume and density. The user can find an object's mass by placing it on a scale, modify the values of the variables dynamically, check whether or not an object is denser than water by throwing it into the water and checking whether or not it floats. The goal is comparing the density of objects to that of water and calculating each quantity depending on the others.
- Windmill (Figure 4.c), an experiment in which a ball is thrown towards a rotating mill. The goal of the simulation is to calculate the speed of the ball, the distance from the ball to the mill and the speed with which the mill rotates so that the ball enters the mill.
- Sliding down an inclined plane (Figure 4.d), an experiment in which an object (a block) in an initial rest state is allowed to slide down a ramp and then on a level surface. The purpose of the simulation is to find out the distance that the object travels after leaving the ramp, when it reaches a target velocity established in the simulation, and knowing the values of the following variables:
    - angle of the ramp
    - height of the ramp
    - frictional coefficient for the ramp
    - frictional coefficient for the level surface

All the listed simulations can be controlled both with the mouse and using gesture input for the webcam, which is classified by the neural network and the application subsequently performs the action associated with the recognized gesture in the simulation. Each simulation allows the user to customize the following settings:

- The weights of the classifier. If the user had previously fine-tuned the model, he/she may choose between this fine-tuned model and the base model.
- The time required to perform the action corresponding to a gesture (default value is 1 second). An action is triggered only after the user continuously makes a gesture towards the camera for the number of seconds set by the user for that particular simulation.
- The threshold value for top-1 probability output of the softmax layer for the input to be considered a gesture (default value is 80%).
- The gesture that triggers each action in the simulation. For each simulation, a default gesture is associated with each action that can be performed. These preset gestures can be modified later by the user, with the help of a GUI. In this GUI (shown in Figure 5), for all actions which can be performed by the user for that particular simulation, a dropdown list containing all gestures and a toggle switch are displayed. The toggle switch allows the user to set whether the gesture is symmetrical or not (i.e., whether performing it with either the right or the left hand triggers the same action). This way, the user may choose to perform a different action using the same gesture, but with the other hand.

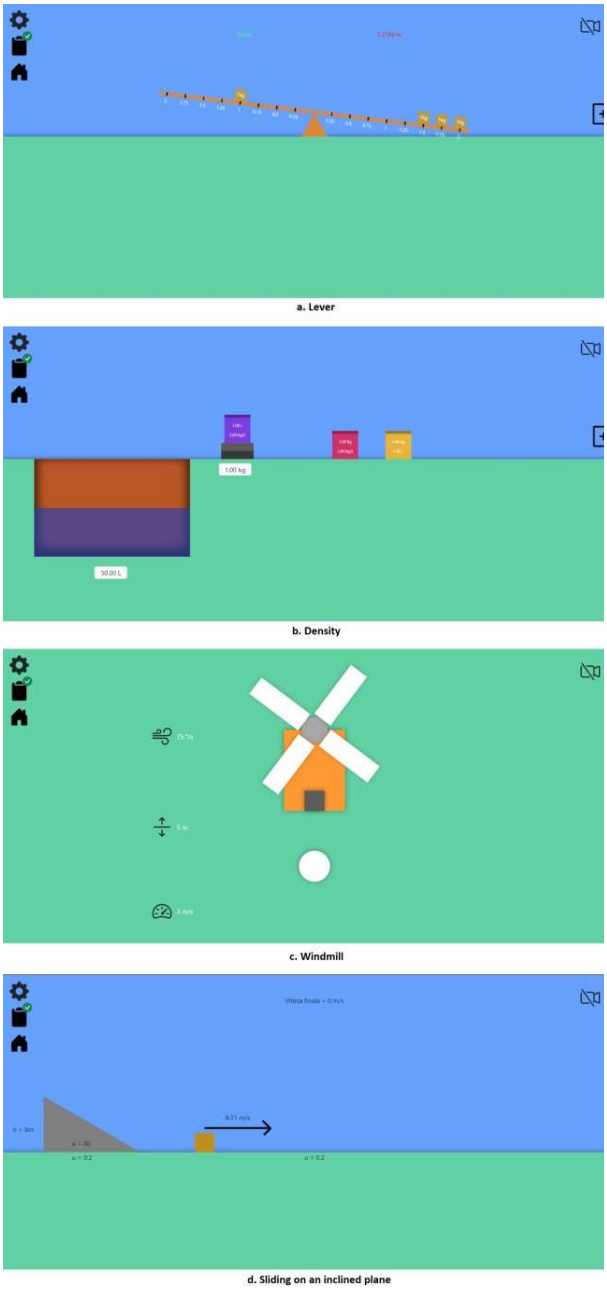

Figure 4: Implemented simulations

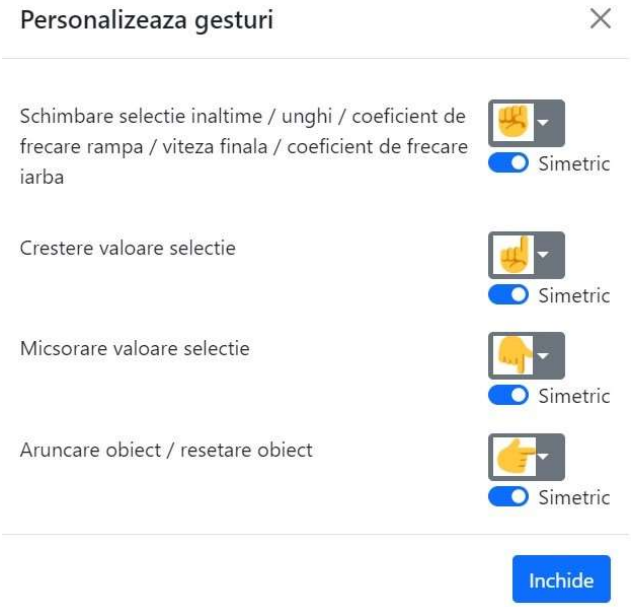

Figure 5: Application Window for Permuting Gestures

EVALUATION

Table 2 shows the classification report for the 180 test samples, obtained using the scikit-learn library [9]. The overall classification accuracy of the model was 98%.

Class Index	Precision	Recall	F1-Score	Support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	9
2	1.00	0.91	0.95	11

3	1.00	1.00	1.00	10
4	1.00	1.00	1.00	10
5	1.00	1.00	1.00	10
6	1.00	1.00	1.00	12
7	1.00	1.00	1.00	10
8	1.00	1.00	1.00	10
9	1.00	1.00	1.00	10
10	1.00	1.00	1.00	10
11	1.00	1.00	1.00	10
12	0.86	0.67	0.75	9
13	0.77	1.00	0.87	10
14	1.00	1.00	1.00	9
15	1.00	1.00	1.00	10
16	1.00	1.00	1.00	10
17	1.00	1.00	1.00	10
Accuracy			0.98	180
Macro average	0.98	0.98	0.98	180
Weighted average	0.98	0.98	0.98	180

Table 2: Classification report for the gesture classifier

The confusion matrix, generated with the Seaborn library, is presented in Figure 6. Note that the model correctly classifies most of the gestures except 12 and 13 (right index and middle fingers up and index up, respectively, which are sometimes confounded) and gesture 2 (left index and middle fingers up).

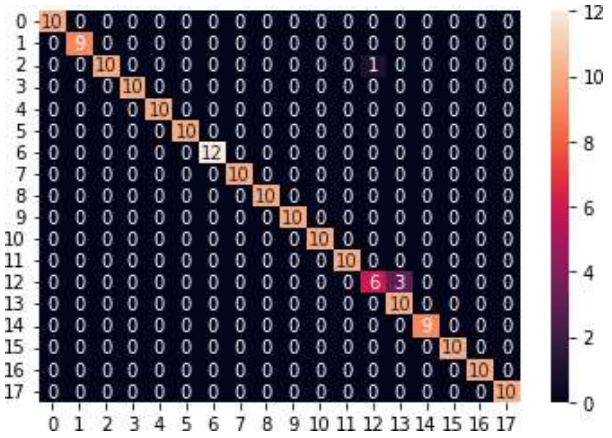

Figure 6: Confusion matrix for the gesture classifier

CONCLUSION

The present paper offers a brief description of the development of an application which serves as a proof of concept for the idea of customizable multimodal interaction with virtual simulation environments. The application allows users to create personalized affordances by associating gestures with actions in virtual physics experiments. The most important contributions of the paper include:

- Proposing a multimodal interaction with virtual experiments, allowing the user to choose between a classical interaction modality using the mouse and keyboard and a more natural interaction modality using gestures in front of the webcam.
- Offering the possibility of customization by permuting gestures associated with actions in each Physics experiment simulation. Each simulation has a default set of gesture-action pairs that can be modified according to the student's preferences for that particular simulation. Moreover, users can also customize both the threshold top-1 probability given by softmax for a gesture to be output by the model (set by default to 80%), and the time required for the gesture input to trigger an action (set by default to 1 second).
- Allowing the user to fine-tune the parameters of the neural network used for gesture recognition. Students may choose to keep the fine-tuned weights or revert to the base model parameters.

The present work also has several limitations. First, there are the psycho-pedagogical limitations inherent to the use of multimedia solutions in education. For example, the interface may capture students' attention at the expense of learning contents. Of course, the aforementioned shortcomings are starting points for future improvements to the application. For example, conditional access to simulations may be implemented. The application may be extended such that it allows experts/teachers to define prerequisites for each simulation and track individual students' completion of the prerequisites. This way, a student's access to an experiment may be conditional on his/her learning outcomes.

Second, from a software perspective, the present application is only a proof of concept, mainly serving demonstrative purposes, and can be improved in multiple ways. For example, the dataset is currently very small, and more samples are clearly needed for training the classifier. Furthermore, the samples should be collected from a larger number of contributors, especially students in the target group. More gestures could be added for the gesture-based interaction modality. A qualitative study of the gestures that students deem natural or intuitive for each of the actions available in the simulations may be a good starting point. Note that gesture input is currently limited to single-frame

single-handed gestures, but future research should consider the development of a recurrent neural network model able to recognize complex user actions based on sequences of frames.

Of course, further validation efforts are needed for the application. Future work should include usability studies on representative student samples. Additionally, to obtain truly relevant feedback from both students and their physics teachers, the simulations of the experiments available through the application should be tested by students immediately after the corresponding physics lessons. Besides usability, student engagement should ideally be surveyed before and after using the application for each experiment. Finally, direct comparisons (in terms of usability, performance, and motivation or engagement) between the proposed application and similar solutions would be desirable in future research.

We can conclude that the presented application, still in an early stage of development, serves as a proof of concept for a customizable natural user interface for physics experiments. It implements didactic means that can become useful for increasing the attractiveness of physics as a discipline, being in agreement with the principles of modern didactics, based on active, student-centered instruction.

REFERENCES

1. Culjak, I., Abram, D., Pribanic, T., Dzapo, H., & Cifrek, M. A brief introduction to OpenCV. In *2012 proceedings of the 35th international convention MIPRO*, IEEE (2012), 1725-1730.
2. Fendt, W. *Physics Applets, by Walter Fendt*, https://www.walter-fendt.de/html5/phro/
3. Garrett, J.J. et al. *AJAX: A new approach to web applications*, 2005.
4. Goldsborough, P. *A tour of Tensorflow*. arXiv preprint arXiv:1610.01178, 2016.
5. Holubova, R. How to Motivate Our Students to Study Physics?. *Universal Journal of Educational Research*, 3, 10 (2015), 727-734.
6. Jaiswal, S. and Kumar, R. *Learning Django Web Development* (volume 336). Packt Publishing Birmingham, United Kingdom, 2015.
7. McLaughlin, B. *What is HTML5?* O'Reilly Media, Inc., 2011.
8. Păvăloiu, I. B.. Leap motion technology in learning. In *Edu world 7th international conference* (2017), 1025-1031.
9. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O. Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., ... and Duchesnay, É. Scikit- learn: Machine learning in Python. *The Journal of Machine Learning Research, 12* (2011), 2825-2830.
10. Physics at school - HTML5, https://www.vascak.cz/physicsanimations.php
11. Physics Education Technology (PHET) Platform (https://phet.colorado.edu/)
12. Pierre Geneves, P., Layaïda, N., and Quint, V. On the analysis of cascading style sheets. In *Proceedings of the 21st international conference on World Wide Web* (2012), 809–818.
13. Pon, M. Z. A., & KK, K. P. (2021). Hyperparameter tuning of deep learning models in Keras. *Sparklinglight Transactions on Artificial Intelligence and Quantum Computing (STAIQC)*, 1 (2021), 36-40.
14. Rutten, N., Van Joolingen, W. R., and Van Der Veen, J. T. The learning effects of computer simulations in science education. *Computers & Education*, 58, 1 (2012), 136-153.
15. Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. How does batch normalization help optimization? *Advances in Neural Information Processing Systems*, 31, 2018.
16. Shakroum, M., Wong, K. W., and Fung, C. C.. The influence of Gesture-Based Learning System (GBLS) on learning outcomes. *Computers & Education*, 117 (2018), 75–101. https://doi.org/10.1016/j.compedu.2017.10.002
17. Suehring, S. *JavaScript Step by Step*. Pearson Education, 2013.
18. The Physics Classroom, https://www.physicsclassroom.com/
19. Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C. L., and Grundmann, M. *Mediapipe Hands: On-device real-time hand tracking*. arXiv preprint arXiv:2006.10214, (2020).
20. Zhang, M., Zhang, Z., Chang, Y., Aziz, E. S., Esche, S., and Chassapis, C. Recent developments in game-based virtual reality educational laboratories using the Microsoft Kinect. *International Journal of Emerging Technologies in Learning (iJET)*, 13(1) (2018), 138-159.