AN OPEN SOURCE VIDEO ANALYTICS TOOL FOR ANALYZING LEARNER NAVIGATION IN IMMERSIVE SIMULATED ENVIRONMENTS

Noah Soriano

College of Information Sciences and Technology
The Pennsylvania State University
E397 Westgate Building
University Park, PA, USA
nfs5267@psu.edu

Ashkan Negahban

School of Graduate Professional Studies
The Pennsylvania State University
30 E Swedesford Rd
Malvern, PA, USA
anegahban@psu.edu

Sabahattin Gokhan Ozden

Information Sciences and Technology
Penn State Abington
Rydal Executive Plaza Building, 317
Abington, PA, USA
sgo7@psu.edu

Omar Ashour

Department of Industrial Engineering
Penn State Behrend
213 AMIC Building
Erie, PA, USA
oma110@psu.edu

ABSTRACT

In educational research, user-simulation interaction is gaining importance as it provides key insights into the effectiveness of simulation-based learning and immersive technologies. A common approach to study user-simulation interaction involves manually analyzing participant interaction in real-time or via video recordings, which is a tedious process. Surveys/questionnaires are also commonly used but are open to subjectivity and only provide qualitative data. The tool proposed in this paper, which we call Environmental Detection for User-Simulation Interaction Measurement (EDUSIM), is a publicly available video analytics tool that receives screen-recorded video input from participants interacting with a simulated environment and outputs statistical data related to time spent in pre-defined areas of interest within the simulation model. The proposed tool utilizes machine learning, namely multi-classification Convolutional Neural Networks, to provide an efficient, automated process for extracting such navigation data. EDUSIM also implements a binary classification model to flag imperfect input video data such as video frames that are outside the specified simulation environment. To assess the efficacy of the tool, we implement a set of immersive simulation-based learning (ISBL) modules in an undergraduate database course, where learners record their screens as they interact with a simulation to complete their ISBL assignments. We then use the EDUSIM tool to analyze the videos collected and compare the tool's outputs with the expected results obtained by manually analyzing the videos.

Keywords: Simulation-based learning, human-simulation interaction, machine learning, video analytics, virtual reality.

ANNSIM'23, May 23-26, 2023, Mohawk College, ON, CANADA;

1 INTRODUCTION

The work presented in this paper is part of an overarching educational research project (summarized in Figure 1) that aims to develop and assess the effectiveness of Immersive Simulation-Based Learning (ISBL) modules in STEM education. An ISBL module is a learning activity in the form of problem- or project-based learning defined around an immersive simulated system. The learning activity is inspired by and resembles real-world situations that learners may face in a professional setting or future workplace. To complete an ISBL assignment, students will interact with a three-dimensional simulation model that resembles a real system. The simulation serves as the context and enables technology-enhanced Problem-Based Learning (PBL) and risk-free experiential learning. For example, instead of physically visiting a real-world production facility, students perform a virtual tour of a simulated facility to collect the data needed for their course project. In addition to eliminating barriers to on-site visits (say, due to lack of proximity to relevant industries, safety risks, or schedule conflicts), many of the pedagogical and psychological theories supported by traditional PBL also apply to ISBL or are augmented due to the integration of PBL with a simulated environment. As highlighted in the figure, one of the research objectives of our overarching project is to investigate the impact of learner navigation (movement in a virtual space) in the simulation on learning outcomes in ISBL. This impact is the main motivation behind developing the tool proposed in this paper since manually collecting learner-simulation interactions in real-time during the experiments or by having a researcher watch screen recorded videos would be infeasible due to the reasons discussed in Section 2. This paper pertains to the highlighted components indicated in Figure 1.

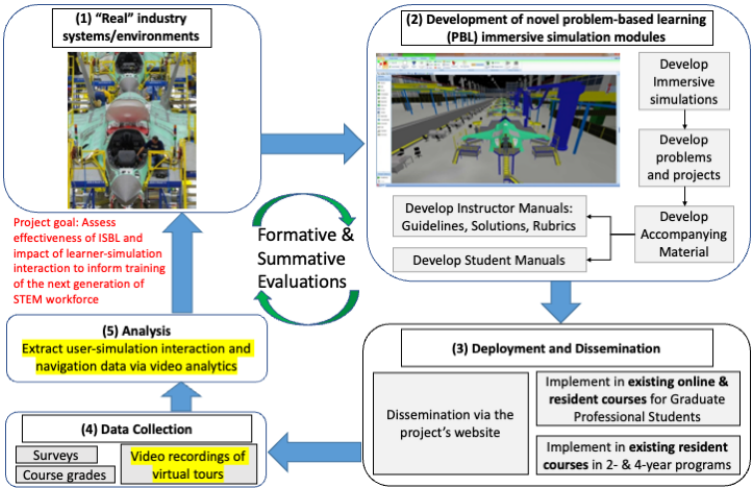


Figure 1: The overarching educational research project

Based on the above motivation, this paper presents an open-source tool for automated environmental detection in simulated settings to assist researchers and instructors with quantification and statistical analysis of user navigation within simulated environments. We call the proposed tool "Environmental Detection for User-Simulation Interaction Measurement" (EDUSIM). The EDUSIM tool runs on a multi-classification Convolutional Neural Network (CNN), which can recognize areas of interest displayed on screen-recorded videos of learners' navigation in the simulation. The tool outputs a data table including time spent at each area in the simulation, average time over all user videos, and deviations between users' time allocations among areas. Due to the inconsistency in user screen recordings, however, not all input videos will contain a clear view of the simulated environment. To combat this inconsistency problem, the EDUSIM tool also implements a rudimentary error handling process via a binary classification model that outputs a predicted flag rating on videos that may contain unrecognizable frames (e.g., user switching to a different program window).

Educational researchers have taken an interest in analyzing the impact of interactions and navigation in simulated environments on students' learning within virtual learning environments such as virtual and augmented reality (Nowparvar et al. 2021, Ozden et al. 2020, Nowparvar et al. 2022). However, simulation software packages generally lack tools for tracking user navigation and interaction within the program, and most programs do not allow modification to the source code to allow for implementation of tracking data. To combat this problem, the EDUSIM tool can be applied to a variety of simulation software programs that lack native tracking functionality by analyzing recorded video data with CNNs.

CNNs are powerful deep learning algorithms most commonly used in visual data analysis. CNNs can recognize and classify images and provide an efficient statistical analysis method on a larger scale as they can process data much faster than humans and with high accuracy. Implementing CNNs in the educational research space is rather cumbersome and requires an extensive background in computer programming and machine learning theory. The proposed EDUSIM tool provides an easy-to-use implementation of CNNs for researchers with little to no background in programming. The tool and its future extensions can be useful for researchers in the field of human simulation interaction for determining user focus in virtual environments, as well as for educational researchers and instructors to assess the impact of learners' usage and navigation within simulation environments on learning outcomes.

The open-source nature of the tool and its ability to receive input from any video data allows it to be modified such that the functions extend beyond interaction and navigation with simulation models. Extensions of the tool can be applied in a variety of applications. For example, in the medical field, the tool may be able to provide information on medical equipment usage by analyzing recordings of student interaction and which objects of interest they primarily focus on. Such tools may also gauge viewer interest in retail design and development by providing time spent by each customer at a particular area in the store and examining visitors' interest in certain art pieces in modern virtual museums.

This paper is structured to provide an overview of the tool's functionality and act as a user manual for future researchers. We begin with a brief literature review on the functionality and performance of common CNN models as well as a discussion on current methods for assessing user-simulation interaction in educational settings (Section 2). Next, in Section 3, we discuss the methodologies behind the functionality of the tool and specific training parameters and provide an explanation of the statistical data in the final outputted table based on actual user videos. Finally, Section 4 presents the conclusions and potential future extensions.

2 RELATED WORK

In this section, we first present the motivation behind the development of the tool and its importance to the educational research space, as well as possible applications in the future by referencing common problems and issues discovered in previous user-simulation interaction research. We then compare of common CNN models to justify our choice of the CNN used in the proposed EDUSIM tool.

2.1 Current Methods for Assessing User Interaction with Simulated Learning Environments

A comprehensive review of VR applications in higher education was conducted in (Radianti et al. 2020), where collected papers were analyzed over their overall VR design elements. The findings suggest that while there is a general interest in studying user interaction with virtual environments, only a small subset of studies (around 7%) collect and analyze usage and navigation data. A handful utilized manual, physical observation of the participants among the papers that do so. This way of observation of the participants can cause a bottleneck during research as logging user information manually is a tedious task and is also prone to human errors by the observer. The remaining papers that study usage and navigation primarily used surveys and written user responses, which are known to be biased and subjective (Brimble 2008) and

only provide qualitative data. In addition, conclusions drawn solely from open-ended response questions of user interaction can be incomplete as the full interactive experience is not tracked, recorded, or evaluated. The limited analysis of user interaction/navigation in virtual/simulated learning environments (despite the general consensus on its importance) can be attributed to the inefficiency and inadequacy of manual and traditional data collection and analysis methods. Integrating machine learning solutions will significantly facilitate data collection and analysis related to user-simulation interaction.

As previously discussed, surveys and manual observation are the most common methods for collecting information on interactions in simulated environments. Here, we mention a sample of these studies. Undergraduate participants were studied in a VR learning environment and were asked to write a paper on how the VR environment compared to viewing a standard classroom slideshow (Parong and Mayer 2018). In another research experiment analyzing player presence in computer games, the research team deemed physical observation of participant interaction too tedious, so they resorted to direct responses from the user (Chung and Gardner 2012). In an investigation of mixed-reality teaching and learning environments, participant behavior was recorded using a screen capture tool and then analyzed by the team through manual labeling of the video data (Plumanns et al. 2016). Our proposed tool automates tedious manual processes to extract user navigation data.

While the current version of our tool is specifically designed to analyze user navigation in simulated environments, it has the potential to extend beyond computerized simulation applications where there is also interest in analyzing usage and navigation. For example, the concept of physical presence in a real-life recording has been studied (Bermejo Berros and Gil Martínez 2021), where 147 undergraduate students participated and were tasked with viewing a 360° video of a YouTuber traveling through a city by car. Within the sample population, one group of students viewed the video through a virtual reality headset, and the other group watched the video on a computer screen monitor. Instead of utilizing a survey response system to quantify their presence in the recorded environment, the team devised space quadrants in the 360° video and measured changes in an omnidirectional movement space to identify when the YouTuber moved outside the focus space on the screen. The researchers manually reviewed all 147 videos and tracked each user's time spent looking at each video quadrant space. Using tools such as the one presented in this paper, researchers can train the machine learning model on these quadrant spaces and output predictions on the amount of time each user has had these spaces in view, effectively bypassing the need for tedious manual measurements.

In the medical research field, students were recorded treating patients in a hospital and were then tasked with analyzing their own videos to evaluate their performance via questionnaire response (Brimble 2008). The responses to these questionnaires were used to evaluate their overall performance on video analysis as a whole. According to the paper, the traditional data collection procedures are "... time-consuming and impractical in the early stage of evaluating." The questionnaires in the research were also determined to be somewhat biased and 'forced', as the students "have felt they had to complete the questionnaires because they were asked to do so by a lecturer." This problem could be solved through automated machine learning tools to provide an unobtrusive evaluation method.

In an experimental design study on rack layout in a retail store, user interest was gauged by manually observing the intensity of exposure (i.e., how long they are seen) (Mowrey, Parikh, and Gue 2017). The research team created a 3D virtual simulation of a store. It quantified interest in store design by measuring the time each user spent interacting with product racks of varying sizes and angles. Through an automated video analytics tool like the one presented in this paper, a significant amount of time could be saved over manual user observation to track the time each participant spends looking at a given location.

In summary, the proposed tool provides a solution and a new way to study user navigation in virtual/simulated environments. Machine learning prediction models have little to no application to automate the observation of full user activity in simulations or virtual reality spaces. Many researchers resort to opin-

ionated responses with a survey system as manual observation techniques have proven to be too inefficient and tedious for data collection tasks. Moreover, important details with a specific environmental focus must be properly captured via traditional data collection methods. Detailed changes in environment/point of interest transitions by the participant are not logged when survey/questionnaire/essay response is used at the end of the experiment. Dedicating time to log these changes introduces delays in the data collection process, especially when the project is scaled up for many participants. The proposed tool in this paper aims to overcome these problems. Using a combination of CNN algorithms and video processing capabilities, our tool can automate the data collection and analysis procedure over large-scale user-simulation interaction video data.

2.2 Common Convolutional Neural Networks (CNNs) for Image Classification

In machine learning applications, image recognition can be tackled in various ways. One of the most common forms is the use of CNNs. CNN models first take an image and pass a pre-defined convolutional filter over a subset of pixel ranges, then identifies patterns according to their weight and outputs a prediction. CNNs can have different parameters and architectures in their convolutional filters. To select an appropriate CNN for the proposed EDUSIM tool, we analyzed performance metrics among three common CNN models: VGG-16, GoogLeNet and ResNet-50, and their ability to predict over the ImageNet database (a collection of over 14 million annotated images).

VGG-16 is a CNN that is 16 layers deep and widely recognized as the winner of the ImageNet recognition competition in 2014. It has been used in a wide range of applications, such as medical research, where it has exhibited ability to identify carcinomas in cytological images and generated predictions with 97.66% accuracy (Guan et al. 2019). GoogLeNet is a CNN model proposed by Google based on the architecture of previous winners of the ImageNet recognition competitions (AlexNet and ZF-Net). It has been shown to have lower error rates compared to VGG-16 by utilizing a combination of convolutional, max pooling, and inception layers devised by the Google research team based on their previous Inception V3 module. The GoogLeNet model has been used in experimentation with face recognition applications in which it exhibited over 91% accuracy (Anand, Shanthi, Nithish, and Lakshman 2020).

ResNet-50 is a convolutional neural network developed by researchers at Microsoft which contains 50 layers of deep residual neural networks. It won the ImageNet classification competition in 2015 and has shown considerably higher prediction ratings in common image classification tasks over VGG-16 and GoogLeNet. As shown by research conducted at Microsoft (He, Zhang, Ren, and Sun 2016), performance in classification over the ImageNet dataset was compared among these three CNN models and evaluated over top-n accuracy (how often a prediction label exists in given top-n classes) as summarized in Table 1. ResNet-50 has shown significantly lower top-1 error rates compared to VGG-16 and lower top-5 error rates compared to both GoogLeNet and VGG-16. It achieves this by minimizing the training parameter size to 26 million instead of 138 million in VGG-16. While this is still higher than GoogLeNet's 7 million parameter size, ResNet-50 employs global average pooling layers over fully-connected layers. It enhances the training accuracy by averaging feature maps to output nodes rather than flattening features to a single fully connected layer (Benali Amjoud and Amrouch 2020). Due to its high accuracy, flexibility over a wide range of image classification applications, and reduced parameter size, we have decided to implement the ResNet-50 architecture as the core framework behind our EDUSIM tool.

Table 1: Training accuracy comparison over ImageNet dataset (He et al. 2016)

Model	Top-1 Error	Top-5 Error
VGG-16	28.07	9.33
GoogLeNet	-	9.15
ResNet-50	22.85	6.71

3 ENVIRONMENTAL DETECTION FOR USER-SIMULATION INTERACTION MEASUREMENT (EDUSIM)

The EDUSIM tool is developed in Jupyter Notebook with the Anaconda programming platform and is available online (Soriano 2022). Installing Anaconda before launching the program is recommended, as it includes the necessary dependencies. Additionally, the following libraries listed below must to be installed in order for the tool to run: OpenCV, Pandas, Numpy, Tensorflow/Keras, and Scikit-learn. Figure 2 shows the graphical user interface (GUI) application wrapper around the tool EDUSIM, allowing the user to run all the required functions without prior knowledge of Python. For simple predictions of the amount of time a recognized area is in a frame, this GUI is sufficient for analyzing user-recorded videos in a simulated environment, as each process is automated via streamlined buttons. However, modifications to the tool for other non-simulation applications will require sufficient knowledge of Python programming, video processing, and CNNs to alter the source code. The following subsections summarize the key components and main steps to use the proposed EDUSIM tool.

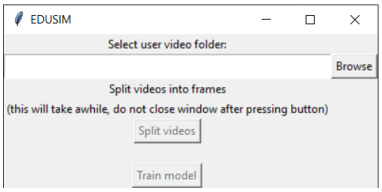
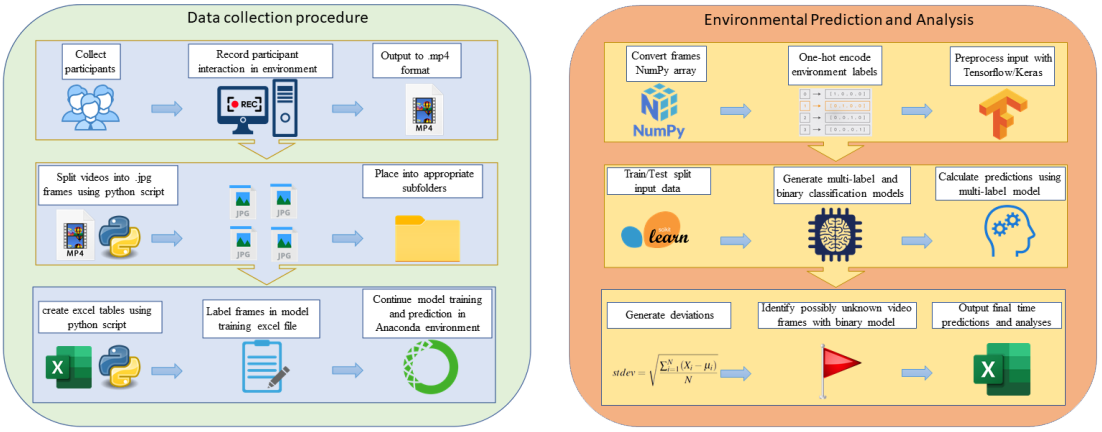


Figure 2: EDUSIM Application GUI.

3.1 Collect Recorded Videos and Label Areas

Figure 3(a) outlines the video data collection and preprocessing steps. The user collects screen recordings of participant interaction in a simulated environment in 1080p format at 30 frames per second. These screen recordings contain useful navigation information, including showing how much time each participant spends looking at areas of interest, which is the primary output in the current version of the EDUSIM tool. The tool accepts .mp4 video format. However, by changing the file extension name in the *glob* function of the *Model_train.ipnyb* file, the tool can also support other formats, such as .avi video formats. In order to protect participant privacy, this data can be used for training the learning model and disposed of after completion.

Before running the code, the user must select a randomized set of videos to train the model. Model training requires manually labeling the specific areas in the virtual environment that participants interact with. For example, the simulation-based learning activity used in our experiments (discussed in subsection 3.4) involves studying different aspects of a simulated airport terminal, including five areas (check-in, security checkpoint, departure gate area, etc.) as well as one for the whole airport when all environments are in view. Each area/environment must be assigned a numerical label starting from a zero-index in the training set in order to allow the model to read the input as categorical labels. When more than one environment is in view, the model is trained to recognize the environment that takes up a majority of the space on the center screen.



(a) (b)

Figure 3: An overview of the EDUSIM data collection and prediction procedures.

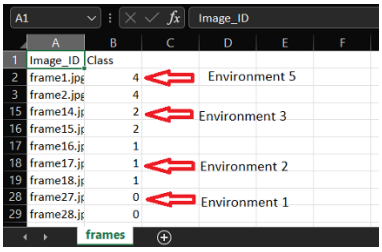
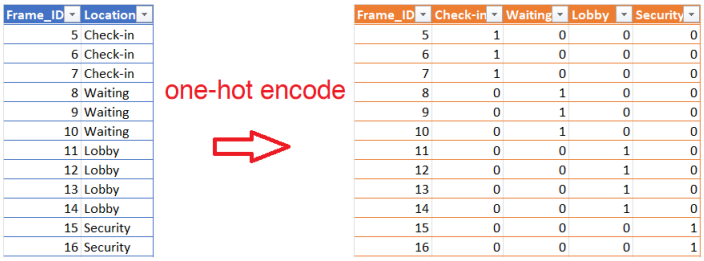


Figure 4: An example of the manual label encoding inputted by the user needed to prepare the training set.

In the *Model_train.ipnyb* Jupyter notebook file, the code utilizes the Python *glob* and *cv2* libraries to extract each selected training video into individual frames for model training. When run, the tool automatically splits these videos into JPEG images and places them in their corresponding folders. Once these frames have been generated, a comma-separated value table or .csv file is created, containing two columns: Image_ID and Class. The *Image_ID* column is automatically generated by the tool, which contains each split video frame separated by a row, while the *Class* will be left blank when generated by the tool for the user to fill in manually. The user must then look through each extracted frame and manually insert the numerical label corresponding to the area they have previously identified into this column, as shown in Figure 4. Once the source training table has been filled out, the model can be trained using the included GUI application wrapper or Jupyter notebook file.

The EDUSIM tool allows the user to train and use a multi-classification CNN for predicting time spent at areas of interest in a simulated environment. However, screen recordings may include video frames in which the participant navigates away from the simulation. Such frames can disrupt the overall accuracy of the multi-classification prediction process as they are not included in the original training set. This irrelevant data set is a common problem in machine learning research known as *open-set recognition* in which the trained model will still output a prediction on unknown frames (e.g., the CNN will still classify a blank screen as an area of interest). A separate binary classification model is included in the EDUSIM tool for identifying frames outside the training set to combat this issue. In order to train this model, the user identifies these 'open-set' frames and places them in the *frames_binary* folder. Next, the *frames_binary* data table must be filled out with '0' representing frames within the training set and '1' representing open-set frames outside of the training input. This filling process will help the tool identify which videos may generate inaccurate prediction results due to a high percentage of open-set recognition frames (i.e., frames



Frame_ID	Location
5	Check-in
6	Check-in
7	Check-in
8	Waiting
9	Waiting
10	Waiting
11	Lobby
12	Lobby
13	Lobby
14	Lobby
15	Security
16	Security

Frame_ID	Check-in	Waiting	Lobby	Security
5	1	0	0	0
6	1	0	0	0
7	1	0	0	0
8	0	1	0	0
9	0	1	0	0
10	0	1	0	0
11	0	0	1	0
12	0	0	1	0
13	0	0	1	0
14	0	0	1	0
15	0	0	0	1
16	0	0	0	1

Figure 5: One-hot encoding.

that are unrecognizable to the multi-classification CNN). This identification is done in the tool by outputting a *flag rate* value for each video (see subsection 3.4 for additional discussion on a sample output data table generated by the EDUSIM tool).

3.2 Run Model Training over Labelled Areas

Figure 3(b) shows the training, prediction, and analysis processes automated by the tool. Once the source files have been properly encoded, both CNNs are trained. The images split in the folder are first converted to Numpy arrays, allowing the CNN to read the image frames in coordinate space. The Matplotlib library converts each image into a pixel array before loading into Numpy, which converts the pixels in the image into numerical values according to the intensity of the RGB color. ResNet CNN takes a 224x224 resolution image on a three-channel RGB color scale (224x224x3), and the tool automatically converts high-definition input video frames into this format. Compared to grayscale images, this generates larger weight and parameter sizes but enables complex prediction methods with color images. Each layer performs a mathematical convolution on an RGB image over a grayscale image. For example, consider a simulated environment containing blue and red doors with the same textures but different colors. On a grayscale image, it is difficult to separate features between these two frames; however, it is much more noticeable when the RGB color scale is kept.

Once the images have been properly arranged into Numpy arrays, the labels are then converted into a one-hot encoded format where location variables are pivoted across the columns and assigned a binary value. This encoding format is then read by the machine learning algorithm. In the context of the EDUSIM training process, the *frames.csv* file is one-hot encoded such that the appearance of each location is assigned to a binary value based on whether it exists at its designated frame (see Figure 5). This information is then loaded into the Tensorflow Keras framework and preprocessed with Keras' built-in ImageNet utilities. This preprocessing allows the CNN model to run convolutions over the pixel arrays and output a valid categorical prediction. Using scikit-learn, the input Numpy array is split into 70/15/15 train, test, and validation size. This splitting will allow the model to generalize over the provided video frames, i.e., learn information from the training set and generate predictions on new, unseen images from the 'test' and 'validation' split.

Once the preprocessing steps are complete, the image array input is compiled for training. A categorical cross-entropic loss function is applied due to the multilabel functionality of the prediction environment. A common problem in machine learning is the introduction of 'overfitting' in which the trained model predicts too closely to its input training images and is unable to generalize and make accurate predictions over unseen images. This overfitting will result in high prediction accuracy over the train set but very low accuracy in the validation set (see Figure 6). If this occurs, the number of training epochs (number of complete passes the model learns over the input) can be lessened or the model training can be stopped early if the user notices the validation accuracy begins to decrease. After the CNN has been trained, both the multi- and binary classification models are then saved to the root 'research' folder for further use in data analysis.

```
Epoch 10/15
104/104 [==============================]
- 57s 538ms/step
- loss: 6.9688
- accuracy: 0.9316
- val_loss: 11.6002
- val_accuracy: 0.5
```

Figure 6: Example of overfitting with 93.16% accuracy on train set but 50% accuracy on validation set.

3.3 Binary Classification of Unknown Video Frames

The general training process for the binary classification model is similar to the multi-classification model. The EDUSIM tool can predict over environments using these two models together and utilize a flagging system on videos with poor prediction accuracy due to large amounts of noise (unknown frames).

3.4 Statistical Analysis and Output

The tool outputs data on time spent in each area of interest. Due to variations in the total interaction time, the tool also provides the *percentage* of time allocated to each area in the simulation. For example, in our experiments, some participants spent upwards of 20 minutes interacting with the airport simulation model while another participant spent 5 minutes of interaction time. This inconsistency of interaction time makes a direct comparison of time allocations infeasible; however, we can still compare the percentage of time allocated to different areas among the participants. The tool also computes the average time and percentage of time spent at each area across all participants (i.e., one average per area). The percentage time averages are then used to calculate a modified measure of standard deviation for each user given by

$$stdev = \sqrt{\frac{\sum_{i=1}^{N}(X_i - \mu_i)}{N}},$$

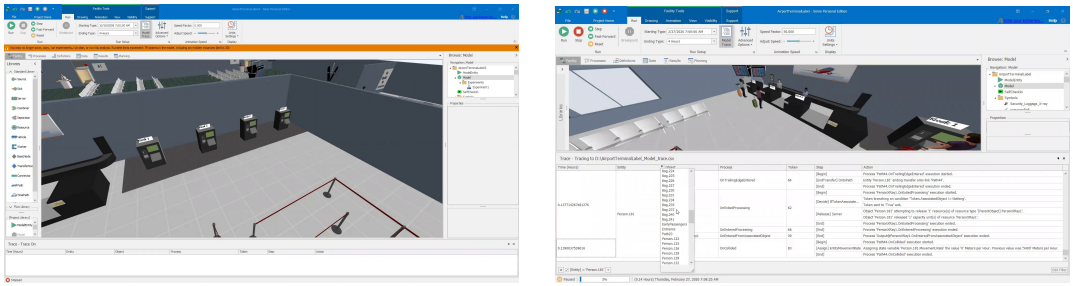
where N denotes the number of areas of interest in the simulated environment, i is the index for the areas ($i = 1, 2, ..., N$), X_i is the percentage of time the user spent in area i, and μ_i denotes the "average" percentage of time each user spends in area i. Figure 7 shows the output statistical data provided by the EDUSIM tool for our research participants who interacted with an airport simulation in the Simio software as part of a simulation-based learning activity. In another work, we will use the *stdev* values and time percentages in conjunction with other data collected on the student participants to determine correlations between student learning and their navigation in the simulation model.

As previously described, an input video's high number of unrecognizable frames can cause inaccurate predictions due to the open-set recognition problem. Using the trained binary model, we can generate a flag rate model to determine which videos contain significant amounts of noise in column N. While the predicted intensity data is left unmodified, these flag rates are included to inform the researcher of possible recording issues due to video noise and alert them to review possible discrepancies in highly flagged input videos manually. The idea is similar to how AI-based online proctoring tools work so that the instructor only needs to review the flagged parts of the recorded video of a student taking a test. Figure 8 provides examples of recognizable and unrecognizable frames in our experiment with the airport simulation model.

We have validated the tool's output by manually performing time studies on user videos and comparing the tool's predictions with the correct values. This validation was done for all videos regardless of their flag rate to further validate the performance of the binary classification model and assess the effect of a high flag rate on the predictions of the multi-classification CNN. Figures 2 and 3 show our comparison with manually collected data for a sample of low and high-flag-rate videos, respectively.

A	B	C	D	E	F	G	H	I	J	K	L	M	N
Video	check in (seconds)	lobby (seconds)	security (seconds)	waiting (seconds)	whole airport (seconds)	check in (%)	lobby (%)	security (%)	waiting (%)	whole airport (%)	Total Time	STDEV	Flag rate
Video 1	57	34	0	1	0	61.9565217	36.95652	0	1.08695652	0	92	26.51462	2.17%
Video 2	3	21	21	6	63	2.63157895	18.42105	18.4210526	5.26315789	55.26315789	114	12.76188	5.26%
Video 3	26	17	0	15	10	38.2352941	25	0	22.0588235	14.70588235	68	15.47325	22.06%
Video 4	31	8	0	4	0	72.0930233	18.60465	0	9.30232558	0	43	28.60133	2.33%
Video 5	22	14	4	2	27	31.884058	20.28986	5.79710145	2.89855072	39.13043478	69	4.210603	2.90%
Video 6	28	14	15	1	54	25	12.5	13.3928571	0.89285714	48.21428571	112	6.516567	0.89%
Video 7	0	5	11	10	55	0	6.17284	13.5802469	12.345679	67.90123457	81	17.66977	2.47%
Video 8	5	24	4	7	56	5.20833333	25	4.16666667	7.29166667	58.33333333	96	11.41119	13.54%
Video 9	0	5	11	10	55	0	6.17284	13.5802469	12.345679	67.90123457	81	17.66977	2.47%
Video 10	31	32	3	1	0	46.2686567	47.76119	4.47761194	1.49253731	0	67	24.51249	2.99%
Video 11	39	51	0	1	2	41.9354839	54.83871	0	1.07526882	2.150537634	93	24.99795	1.08%
Video 12	1	2	1	13	35	1.92307692	3.846154	1.92307692	25	67.30769231	52	18.71326	25.00%
Video 13	24	16	2	2	46	26.6666667	17.77778	2.22222222	2.22222222	51.11111111	90	5.327026	15.56%
Video 14	1	0	0	2	111	0.87719298	0	0	1.75438596	97.36842105	114	28.71989	1.75%
Video 15	6	15	7	6	47	7.40740741	18.51852	8.64197531	7.40740741	58.02469136	81	10.54375	2.47%
Average	18.26666667	17.2	5.266666667	5.4	37.4	24.1391529	20.79067	5.74687054	7.49583452	41.82746778			

Figure 7: Sample table output. Column A: The list of input user videos. Columns B-F: The amount of time at each area of interest predicted by the tool measured in seconds. Columns G-K: Percentage of time spent in the respective area. Column L: Total length of each recorded video. Column M: *stdev* calculation relative to all videos in the input dataset. Column N: Calculated flag rate or percentage of open-set (unrecognizable) frames discovered in the video by the binary classification model. For instance, Video 3 and Video 12 exhibit high flag rates, indicating a large number of open-set frames.



(a) (b)

Figure 8: (a) is an example of a "clean" frame from Video 4 where the simulation window is in full view and the tool can provide an accurate prediction of the given area. (b) shows an example of an unrecognizable/noisy frame from Video 12. The “Model Trace" window covers the majority of the simulation program window and thus outputs inaccurate predictions due to open-set recognition.

4 CONCLUSIONS

We develop a video analytics tool called EDUSIM that analyzes user navigation and time allocation to different areas within a simulated environment via a multi-classification CNN. The tool is publicly available online (Soriano 2022). The current version of the tool is being used in a STEM educational research project to identify the correlation between students' learning outcomes and navigation in simulation-based learning environments. The EDUSIM tool accurately predicts known locations in video recordings of user-simulation interaction as compared to manual stopwatch calculations. The tool also includes a flagging mechanism based on a binary classification model to assist researchers and instructors in identifying videos that suffer from recording issues that may negatively affect the prediction accuracy of the multi-classification model.

There are many possibilities for assessing user-simulation interaction. Future extensions of the tool include quantifying additional measures of navigation and interaction beyond time tracking (e.g., interaction with specific objects within the simulation). With further development, we plan on allowing the tool to recognize a variety of applications as well as analyze interactions in popular virtual reality programs such as VRChat and Mozilla Hubs. Our research team is currently investigating some of these possibilities. Another important area for future research is to exploring other potential remedies to deal with the open-set recognition

	Check in	Lobby	Security	Waiting	Whole Airport
Video 5 (tool)	22	14	4	2	27
Video 5 (manual)	22	14	5	2	26

Table 2: Comparison between outputted predictions (in seconds) by the tool and manual stopwatch (rounded up to nearest integer) of Video 5 which has a 2.90% flag rate. EDUSIM tool exhibits highly accurate predictions across all areas in the simulation model.

	Check in	Lobby	Security	Waiting	Whole Airport
Video 12 (tool)	1	2	1	13	35
Video 12 (manual)	6	3	6	3	21

Table 3: Comparison between the tool's predictions and manual stopwatch time collected for Video 12 which has a 25% flag rate. The multi-classificaion CNN exhibits high degrees of error for all areas due to the high level of noise in the input video, indicating that specified areas of interest cannot be predicted accurately in this video recording due to the high flag rate as recognized by the binary classification model.

problem in multi-classification CNNs to eliminate the need for manually reviewing videos with high flag rates. We hope that this work paves the way for more in-depth analysis of user-simulation interaction in simulation-based learning environments and other contexts where such interactions are of interest.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 2000599. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The preliminary stages of this work were supported by funds from the Office of the Executive Vice President and Provost at Penn State as part of the university's Strategic Seed Grant program on transforming education. We thank Aung Nay Htet Oo, an undergraduate researcher at Penn State, who assisted in developing the ISBL modules used here.

REFERENCES

Anand, R., T. Shanthi, M. S. Nithish, and S. Lakshman. 2020. "Face Recognition and Classification Using GoogleNET Architecture". In *Soft Computing for Problem Solving*, edited by K. N. Das, J. C. Bansal, K. Deep, A. K. Nagar, P. Pathipooranam, and R. C. Naidu, pp. 261–269. Singapore, Springer Singapore.

Benali Amjoud, A., and M. Amrouch. 2020. "Convolutional Neural Networks Backbones for Object Detection". In *Image and Signal Processing*, pp. 282–289. Cham, Springer International Publishing.

Bermejo Berros, J., and M. Gil Martínez. 2021, 12. "The relationships between the exploration of virtual space, its presence and entertainment in virtual reality, 360º and 2D". *Virtual Reality* vol. 25, pp. 1–17.

Brimble, M. 2008, 09. "Skills assessment using video analysis in a simulated environment: an evaluation". *Paediatric Nursing* vol. 20, pp. 26–31.

Chung, J., and H. J. Gardner. 2012. "Temporal Presence Variation in Immersive Computer Games". *International Journal of Human–Computer Interaction* vol. 28 (8), pp. 511–529.

Guan, Q., Y. Wang, B. Ping, D. Li, J. Du, Q. Yu, H. Lu, X. Wan, and J. Xiang. 2019, 08. "Deep convolutional neural network VGG-16 model for differential diagnosing of papillary thyroid carcinomas in cytological images: A pilot study". *Journal of Cancer* vol. 10, pp. 4876–4882.

He, K., X. Zhang, S. Ren, and J. Sun. 2016. "Deep Residual Learning for Image Recognition". In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778.

Mowrey, C. H., P. J. Parikh, and K. R. Gue. 2017. "The impact of rack layout on visual experience in a retail store". *INFOR: Information Systems and Operational Research* vol. 57 (1), pp. 75–98.

Nowparvar, M., O. Ashour, S. Ozden, and A. Negahban. 2022. "An Assessment of Simulation-Based Learning Modules in an Undergraduate Engineering Economy Course".

Nowparvar, M., X. Chen, O. Ashour, S. Ozden, and A. Negahban. 2021. "Combining Immersive Technologies and Problem-based Learning in Engineering Education: Bibliometric Analysis and Literature Review". In *Proceedings of the 2021 ASEE Annual Conference*.

Ozden, S., O. Ashour, and A. Negahban. 2020. "Novel Simulation-Based Learning Modules for Teaching Database Concepts". In *Proceedings of the 2020 ASEE Annual Conference*.

Parong, J., and R. Mayer. 2018, 01. "Learning Science in Immersive Virtual Reality". *Journal of Educational Psychology* vol. 110.

Plumanns, L., T. Sommer, K. Schuster, A. Richert, and S. Jeschke. 2016. *Investigating Mixed-Reality Teaching and Learning Environments for Future Demands: The Trainers' Perspective*, pp. 393–405. Cham, Springer International Publishing.

Radianti, J., T. A. Majchrzak, J. Fromm, and I. Wohlgenannt. 2020. "A systematic review of immersive virtual reality applications for higher education: Design elements, lessons learned, and research agenda". *Computers & Education* vol. 147, pp. 103778.

Noah Soriano 2022, March. "Environmental Detection for User-Simulation Interaction Measurement (EDUSIM)". https://doi.org/10.5281/zenodo.6394885.

AUTHOR BIOGRAPHIES

NOAH SORIANO is a graduate research assistant and an M.S. student in the College of IST at The Pennsylvania State University. He earned his B.S. in Computational Data Science from Penn State. His research interests include machine learning design and application to computer vision. His e-mail address is nfs5267@psu.edu..

ASHKAN NEGAHBAN is an Associate Professor of Engineering Management at the School of Graduate Professional Studies at The Pennsylvania State University (USA). He received his Ph.D. and master's degrees from Auburn University (USA) and his BS from University of Tehran (all in Industrial and Systems Engineering). His research involves stochastic simulation methods, primarily agent-based and discrete-event simulation. He also conducts research related to novel simulation-based learning environments in STEM education. His email and web addresses are anegahban@psu.edu and http://ashkannegahban.com.

SABAHATTIN GOKHAN OZDEN is an assistant professor of Information Sciences and Technology at Penn State Abington. He received B.S. degree in Software Engineering with a double major in Industrial Systems Engineering from Izmir University of Economics in 2009. He received his Master of Industrial and Systems Engineering and a Ph.D. degree in Industrial and Systems Engineering from Auburn University in 2012 and 2017, respectively. His research interests are warehousing, optimization, and information systems. He can be reached at gokhan@psu.edu and http://gokhanozden.com.

OMAR ASHOUR is an Associate Professor of Industrial Engineering at Pennsylvania State University, The Behrend College. He received his B.S. and M.S. degrees in Industrial Engineering from Jordan University of Science and Technology (JUST) in 2005 and 2007, respectively. He received his M.Eng. and a Ph.D. degree in Industrial Engineering and Operations Research from The Pennsylvania State University (PSU) in 2010 and 2012, respectively. Dr. Ashour's research areas include data-driven decision-making, modeling and simulation, data analytics, immersive technologies, process improvement and engineering education .