Received 19 April 2025, accepted 1 May 2025, date of publication 6 May 2025, date of current version 15 May 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3567437

RESEARCH ARTICLE

Fast and Accurate Video Analysis and Visualization of Classroom Activities Using Multiobjective Optimization of Extremely Low-Parameter Models

VENKATESH JATLA[1], SRAVANI TEEPARTHI[1], UGESH EGALA[1], SYLVIA CELEDÓN-PATTICHIS[2], AND MARIOS S. PATTICHIS[1], (Senior Member, IEEE)

[1]Image and Video Processing and Communications Laboratory, Department of Electrical and Computer Engineering, The University of New Mexico, Albuquerque, NM 87131, USA

[2]Department of Curriculum and Instruction, The University of Texas at Austin, Austin, TX 78712, USA

Corresponding author: Marios S. Pattichis (pattichi@unm.edu)

This work was supported in part by the National Science Foundation under Grant 1949230.

This work involved human subjects or animals in its research. Approval of all ethical and experimental procedures and protocols was granted by the UNM Office of the IRB (OIRB).

ABSTRACT The paper considers the problem of video activity recognition in real-life collaborative classroom learning environments. Video analysis of real-life collaborative classroom learning environments faces significant challenges not encountered in current, advanced video recognition datasets. In collaborative learning environments, students are arranged in small groups where they interact within their group. Video analysis needs to deal with long-term activity recognition (of one hour or more session videos), detect multiple simultaneous activities, rapid transitions between activities, occlusions, and numerous individuals performing similar activities in the background that are not part of the group being analyzed. Developing ground truth datasets for analyzing complex video datasets is prohibitively expensive. We dramatically reduce the requirement for large ground truth datasets by creating separate, custom datasets for object detection and video activity recognition. We then introduce a separable, extremely low-parameter system for video activity recognition that can be optimally trained using the derived datasets without the need for transfer learning from larger systems trained on large datasets. We further develop an interactive WebApp for visualizing the results over long video sessions. Overall, the extremely low-parameter activity classification model uses just 18.7K parameters for each activity, requiring 136.32 MB of memory. On a moderate GPU (RTX 5000), the activity classification model runs at an impressive 4,620 (154×30) frames per second. Our approach uses at least 1,000 fewer parameters than several well-established methods for video recognition. Our extremely low-parameter classifiers can process 90 minutes of video in just 26 seconds. Furthermore, our models are much easier to train, they are much faster, and outperform comparable methods.

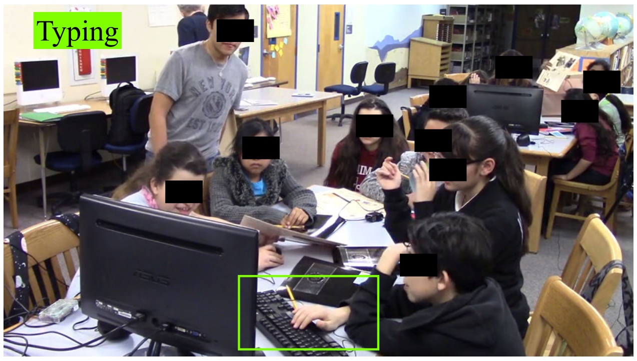
INDEX TERMS Multi-objective optimization, extremely low-parameter neural networks, fast inference, separable models, video activity localization, educational video analysis.

I. INTRODUCTION

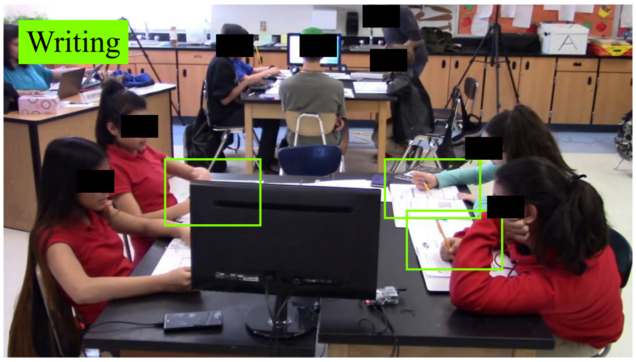
Our paper considers the development of video activity recognition and visualization in real-life middle-school classrooms. Our focus on real-life datasets faces unique challenges that stem from the need to analyze complex real-life classroom video sessions of 60 to 120 minutes.

Overall, our objective is to help educational researchers to analyze classroom videos based on student participation. The videos record students working through an integrated

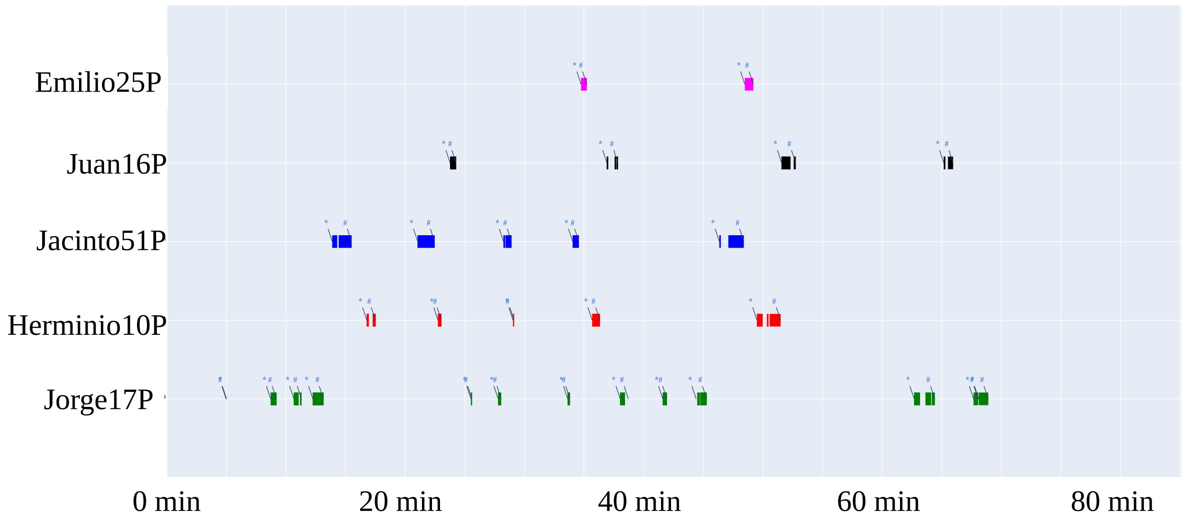
The associate editor coordinating the review of this manuscript and approving it for publication was Senthil Kumar.



(a) Typing activity in collaborative learning environment. The keyboard is partially visible, and the students are very close to each other.



(b) Writing activity in collaborative learning environment. Multiple writing activities with complete or partial occlusion are happening in this example.



(c) An activity map showing typing activity for a 1 hour 23 minute session. The asterisks are web-links that point to corresponding time in the video.

FIGURE 1. Typing and writing activities and expected visualization. The interactive activity map with the activities associated with the person helps the user to get a better understanding of the detected activities.

curriculum that integrates mathematics with Python programming activities. The students engage in writing activities as they work through the mathematics exercises. They engage in typing activities as they work through the programming exercises. Thus, to help assess student participation, our goal is to recognize writing and typing activities.

We present some of our real-life classroom dataset challenges in Fig. 1. In Figs. 1(a)-(b), we are focusing on two classroom activities: typing and writing. Our goal is to analyze the video so that we can determine the video segments where each activity is happening. In Figs. 1(a)-(b), we highlight each activity using a green bounding box. To facilitate interactive video analysis over long videos (>60 minutes), we provide an interactive WebApp (see Fig. 1(c)) that includes links to the video where each activity is happening. Thus, the users can click on the links shown in Fig. 1(c) to bring back the video at the starting point of the activity as shown in Figs. 1(a)-(b).

From our example in Fig. 1, we can see many of the challenges associated with analyzing real-life classroom videos. In Figs. 1(a) and 1(b), we note that our video analysis is focused on analyzing the group that is closest to the camera, appearing in the bottom half of the frame. To avoid confusion due to the complex scene environments, on the first frame of each long video session, we ask the users to select the table and initial table regions associated with each student. This quick annotation on the first frame is part of our interactive system design that allows the users to query the system for specific activities by specific students with minimal required effort by the users. Clearly, in the rare event that the students change their sitting arrangement, the users would be expected to reannotate the initial frame of the new scene arrangement. This is the only input provided to our system. Our goal is to develop an effective video analysis that can process videos fast with moderate, relatively low-price, hardware.

Significant challenges are associated with recognizing activities within the selected video scene. First, we note that we have multiple activities that occur simultaneously (see Fig. 1(b)). Second, once we recognize each activity, we need to attribute each activity to a specific student within the group. In other words, we need to integrate the outputs in our interactive map of all recognized activities against time as shown in Fig. 1(c). In Fig. 1(c), we see that our goal is to process the raw videos so that we can show when each student performs each activity. To make it clear, in Fig. 1(c), each student's activity is represented by a different row plot. Then, the time of each activity corresponds to the length of each bar. Third, from the plot, we note that we have relatively long activities performed by multiple people. Fourth, our videos are collected with low-cost cameras where everyone appears to be in focus. This accounts for structural noise where we do not benefit from background activities being out of focus. Fifth, it is clear that we have peripheral activities and occlusion that can occur at different locations in each video frame.

To avoid biasing our methods, we train on long video sessions collected over one set of dates, validate on another set of dates, and then test on yet another set of dates. The training, validation, and testing dates are distinct. Thus, the testing datasets reflect our goal of measuring generalization on completely unseen scenes and new camera setups.

Modern datasets address some of the challenges that we have found with our real-life classroom videos. Most recently, Korban et al. [1] looked at classroom videos with multiple activities occurring simultaneously. Similarly, videos with simultaneous activities can be found in Dimitriadou and Lanitis in [2], the AVA 2.2 dataset [3], EPIC-Kitchen [3], JHMDB [4], and UCF101-24 [5]. Nevertheless, none of these datasets exhibit the entire range of challenges associated with our dataset. Specifically, none of them require that we process raw (untrimmed) real-life videos that compute activity maps where each activity is mapped to a specific person, as shown in Fig. 1(c). Furthermore, except for specialized sports videos, none of them were designed for processing long duration videos of 1 to 1.5 hours.

For system design, we consider methods to minimize the required amount of ground truth, reduce training time, and model performance optimization. In what follows, we provide a summary of our approach. More details are provided in our methodology.

We minimize the amount of required ground truth by considering each activity as a problem of detecting an object involved in the activity and then classifying activities associated with that object only. For our purposes, we will consider typing activities associated with locating a keyboard and writing activities associated with hand detection. As a result, for object detection, we only require a still image dataset that contains representative examples of keyboards and hands. Then, for each activity, we only need to classify short video segments over each detected object. For typing, we classify short video segments over the keyboard region. For writing, we classify short video segments over the hands regions.

Our single-activity optimization framework provides fundamental advantages over the standard use of transfer learning from large-activity networks. First, we note that it is a lot easier to train for a single activity at a time as opposed to training large models on multiple activities. Second, our separable action recognition is inherently parallel since we can run each video activity detector separately. Third, it is a lot easier to debug and understand what a single activity detector is doing as opposed to understanding how a large video activity system processes any given single activity. Our approach eliminates the need to consider a multitude of large networks trained over different large datasets. Fourth, we note that our approach is scalable. We can easily develop larger networks to train and test on larger datasets that target all of the variations of our single activity approach. This is clearly a much more efficient approach than to keep growing larger models that target all possible activities so that we re-target them on a single activity.

Training time is dramatically reduced by our approach. To see this, note that instead of training on a large video dataset, we only need to train on a small number of image examples and very short and small video segments over each object. Yet, for our final testing, we do report over entire videos.

We adopt a multiobjective framework for optimizing our model. Activity detection is performed based on a Video Activity Proposal Network (VAPN) and a Video Activity Classification Network (VACN). Our VAPN uses optimized object detection and tracking to extract short video segments of possible activities of interest. In terms of our multiobjective framework, the goal of the VAPN is to minimize the false-negatives while tolerating false-positives. For VACN, our multiobjective optimization approaches selects an optimal framework over a family of 3D CNN networks that process the video segments at an optimized frame rate. Compared to the standard use of large networks, our optimized VACN uses 1000$\times$ or less parameters, perform better and run much faster.

We next provide a brief summary of popular Human Activity Recognition (HAR) methods that we will be comparing against. The Temporal Segment Network (TSN) [6] samples the input video through the extraction of equal video segments. It then processes each video segment using ConvNets and computes the final output using a consensus map. The Two-Stream Inflated 3D ConvNet (I3d) [7] extends successful 2D image classification architectures into 3D, aiming at the extraction of spatiotemporal features. The Temporal Shift Module (TSM) [8] aims at developing a high-performance video understanding system by implementing 1D convolutions across frame channels by embedding the time-convolution operations inside efficient 2D CNN networks. The SlowFast network [9] uses a slow and a fast stream to process videos at two different frame rates.

Our approach develops a famitly of 3D CNN architectures that operates at an optimized frame rate that needs to be learned during training. While we do not optimize for GPUs, we demonstrate that our approach is faster than all other methods (including TSM), is easier to train since we are using 1000× or fewer parameters, while performing at-least as good as any of these well-accepted methods. We achieve this performance because we are developing optimized architectures that only need to be trained for recognizing a single activity at a time. We demonstrate our approach on recognizing writing and typing activities.

Preliminary elements of our video activity recognition system have appeared in conference papers [10] and [11]. Specifically, the proposed hand detection method has first appeared in [10] as a conference paper. In the current paper, we incorporate hand detection for writing recognition. However, we also introduce the use of keyboard detection for typing recognition. Earlier versions of the video activity classification system have appeared in [11] as a conference paper. The current paper covers the complete system, with extensive discussion on the optimization process, a significantly larger dataset, inference speed, performance, and video activity visualization over full video sessions. We also mention our related work focused on dynamic participant tracking reported in [12]. This earlier work built upon student recognition reported in [13]. For the current paper, we specifically avoid the need to recognize the students. Thus, our approach allows us to recognize student activities by working with fully anonymized datasets that avoid processing sensitive student information.

In summary, the primary contributions of the paper include:

- **Real-life classroom datasets:** As opposed to the standard end-to-end video datasets for everything, we introduce reduced training and validation datasets for training. At the same time, we test on real-life 90-minute videos.
- **An interactive system for localizing and visualizing typing and writing activities:** Unlike other activity classification approaches, our system recognizes individual activities and provides interactive visualization over the entire video.
- **A multiobjective optimization framework:** We present a multiobjective optimization framework for jointly optimizing accuracy (or AUC), the number of parameters, and inference time. We present different optimization modes that can be used to prioritize different objectives (e.g., accuracy or inference time), or to produce models that balance all objectives.
- **Significant reduction of the number of parameters by 1000×:** Our proposed approach for video activity classification gave better results using more than 1000 times fewer parameters. Given the fact that our model is at least as accurate as any other model that we compare against, it has better generalization capabilities than the models that we compare against (e.g., see Chapter 7 in [14]). At the same time, our extremely low-parameter models are expected to have higher bias and lower variance than much larger models that would have required much larger training datasets (e.g., see bias-variance discussion in [14]). In terms of our multiobjective optimization framework, our use of extremely low parameter models results in fast inference times, a reduction in memory requirements, and faster training.
- **No need for pretraining:** Unlike all other methods, our video activity classification network does not require any pretraining on large datasets.

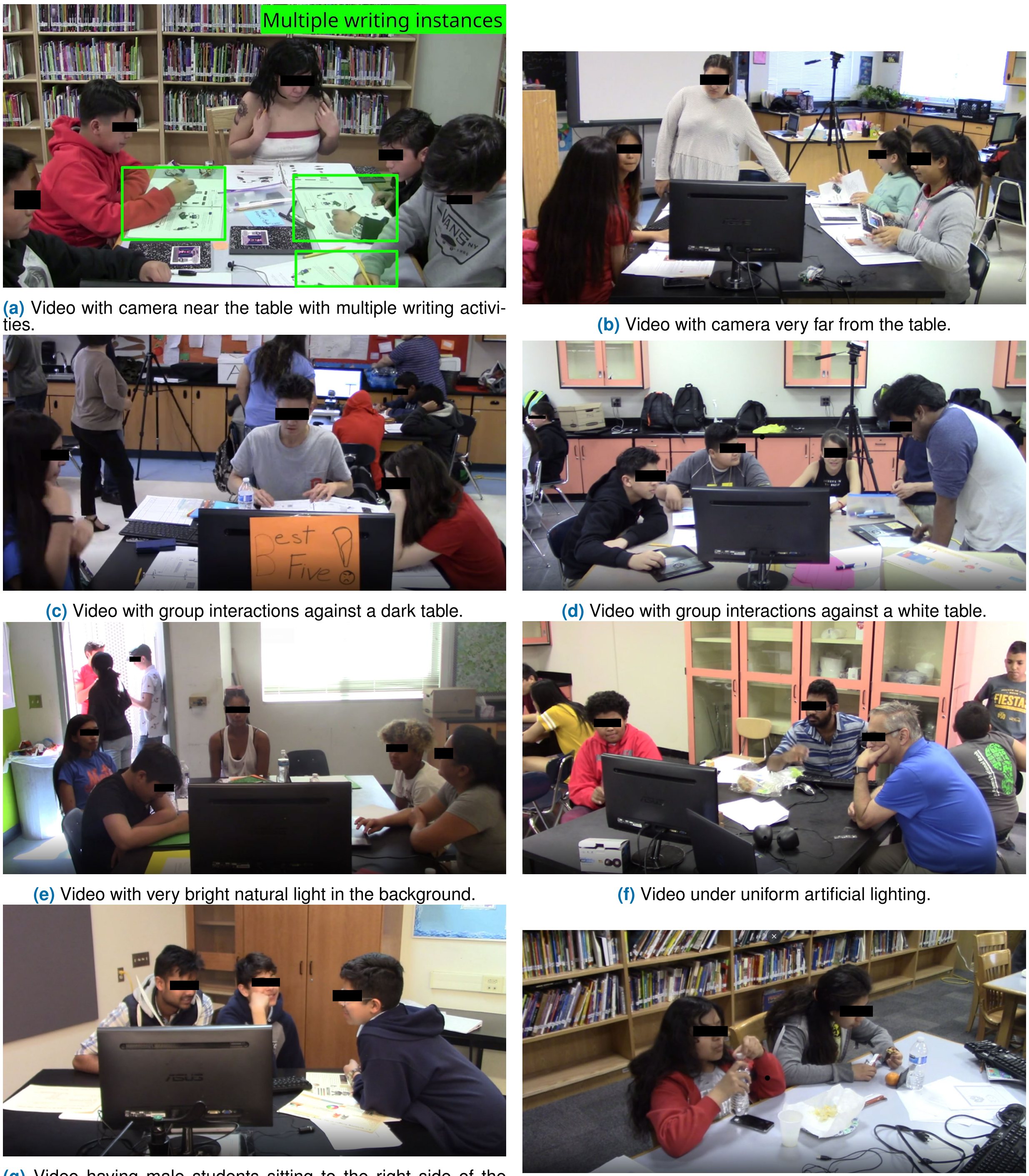
The remainder of this paper is organized into five sections. We summarize our AOLME activity dataset in section II. We provide background information on related research in section III. We describe our proposed approach in section IV. We summarize our results in section V and provide concluding remarks in section VII.

II. AOLME STUDENT DATASETS

We provide several examples of our real-life classroom dataset in Fig. 2. Here, we can see how the students are arranged in different groups (e.g., see Fig. 2(f)). We need to attribute each activity to a specific person as we demonstrate in Fig. 2(a). The videos were recorded using low-cost video cameras with small sensors that keep everything in focus, located at many different angles (see Figs. 2(b) and 2(d)). We have peripheral activities that can occur at the edges of the video (e.g., see Figs. 2(a), 2(b) with strong variations in lighting and appearance (see Figs. 2(a)-2(h)). We provide a comparative summary of our dataset against related modern datasets in Table 1. We will next describe the similarities and differences between our dataset and other datasets in more detail.

A. VIDEO ACTION RECOGNITION DATASETS

The UCF-101 dataset [5] provides a large collection of video clips that are commonly used for action recognition research. It contains 13,320 videos categorized into 101 different action classes, such as "Playing Guitar," "Cricket Shot," and "Soccer Penalty." These videos were collected from YouTube, providing diversity in camera angles, lighting conditions, and background environments. The videos have a frame rate of 25 frames per second (fps), with a typical resolution of 320 × 240 pixels, and average duration of 7 seconds. More recently, the UCF101-24 [5] provided a specialized subset of the UCF101 dataset that includes spatio-temporal annotations covering 24 unique human activities. Unfortunately, UCF101-24 is primarily focused on videos characterized by a single activity. UCF101-24 lacks the



(a) Video with camera near the table with multiple writing activities.

(b) Video with camera very far from the table.

(c) Video with group interactions against a dark table.

(d) Video with group interactions against a white table.

(e) Video with very bright natural light in the background.

(f) Video under uniform artificial lighting.

(g) Video having male students sitting to the right side of the table.

(h) Video having female students sitting to the left side of table.

FIGURE 2. **Figure showing variability in AOLME group interaction videos.**

complexity for describing long, simultaneous activities by multiple students (see Table 1).

Unlike UCF-101, the JHMDB dataset [4] focuses on analyzing human activity using techniques like puppet flow, puppet masks, and per-frame joint positions. As a result, each video in JHMDB is mostly centered around one actor and represents a single specific activity.

The AVA 2.1 dataset [15] contains 80 atomic visual actions across 430 15-minute movie clips, with actions localized in both space and time, leading to 1.62 million action labels. Multiple labels per person often occur, enabling the study of

TABLE 1. Comparison of recent datasets used for recognizing multiple video activities. Videos from different days refers to the requirement that training and testing videos should come from different days. Multiple activities refer to multiple people performing multiple activities. Long multiple activities implies that the videos are over 1 hour long. Low-cost cameras refer to the use of amateur video equipment that do not produce broadcast video quality. Datasets that do not satisfy the requirement are marked by ✗. Datasets that satisfy the requirement are marked by ✓.

Dataset	Group-based person attributed activities	Long multiple activities by multiple persons	Multiple activities at the same time	Low-cost cameras	Peripheral activities
UCF101-24 (2015) [5]	✗	✗	✗	✓	✓
JHMDB (2013) [4]	✗	✗	✗	✓	Mostly ✗
AVA 2.1, 2.2 (2018) [15]	✗	✗	✓	✗	Mostly ✗
EPIC-Kitchen (2018) [3]	✗	✗	✗	✓	✗
Multisports (2021) [16]	✗	✗	✓	✗	Mostly ✗
Dimitriadou *et al.* (2022) [2]	✗	✗	✗	✓	✗
Korban *et al.* (2024) [1]	✗	✗	✓	✓	✓
AOLME (Ours, 2024)	✓	✓	✓	✓	✓

simultaneous actions. Unlike real-world videos, these videos are taken from movies which tend to primarily focus on the activity performed by the actor. In addition, the labeled actions are atomic actions that do not account for long-term, context-driven activities.

The EPIC-Kitchen [3] dataset consists of 100 hours of recordings captured in Full HD (FHD) with 20 million frames, all recorded from a head-mounted camera. The dataset focuses on egocentric, first-person activities in kitchen environments. EPIC-Kitchens does not involve multiple people performing different activities simultaneously, nor does it capture interactions between actors.

The Multisports dataset [16] is designed for action recognition, focusing on athletic activities across various sports. It includes high-quality video footage that captures a wide range of dynamic actions, often involving entire body movements. The dataset emphasizes brief, fast-paced actions within relatively short video segments, making it useful for analyzing complex, high-speed sports scenarios. However, the dataset lacks a clear temporal structure, which presents challenges in modeling and predicting action sequences based solely on time progression. The dataset is particularly suitable for tasks involving unstructured, full-body motion analysis.

Dimitriadou et al. [2] worked with simulated videos where students performed specific actions for 10 seconds. Each video captures one student, consistently centered on the screen. The dataset is carefully controlled, with minimal occlusions, and the subject is always in clear view. This dataset features only one person per video. This provides a well-organized and consistent environment for action recognition tasks. The videos do not capture interactions between the students.

In Korban et al. [1], the authors describe a dataset consisting of nearly 250 hours of classroom videos, collected from elementary mathematics and English language arts lessons. The dataset includes annotations for various instructional activities, such as whole class, small group, individual activities, and transitions. These videos were used to train neural networks to classify activities based solely on visual data, without relying on audio. The dataset provides fine-grained labels for classroom interactions, making it highly suitable for machine learning applications in educational research and classroom activity recognition. While it offers insights into student group dynamics, it does not associate activities with individual students or provide long-term insights into the learning process. In our study, we aim to address this limitation by defining and associating student regions with activities.

B. AOLME VIDEO DATASET PREPARATION

Out of each camera, each video comes compressed at a specific frame rate. We considered different video transcoding parameters to standardize the resolutions, the frame rate, and also to recompress the videos for online video streaming. We compressed a small video dataset at different rates and determined settings that did not compromise video quality while maintaining high-quality audio. Video quality was judged by reviewing the transcoded videos. We then selected the optimized transcoding parameters as described next.

All videos are transcoded using `ffmpeg` as given by:

```
ffmpeg −i <input video> \
  −vf scale=858:480 \
  −c:v libx264 \
  −c:a mp3 −b:a 255k \
  −b:v 2.5M \
  −maxrate 2.5M \
  −bufsize 1.25M \
  −r 30 \
  −x264−params \
   "keyint=30:min−keyint=30:no−scenecut" \
  <output video>
```

We next explain each parameter. Each video is resized to 858 × 480 pixels using `-vf scale=858:480` option. We specify the use of the H.264 video codec as given by `-c:v libx264`. To allow for wider access, we set the bitrate to 2.5Mbps using `-b:v 2.5M` option. We maintain high audio quality by setting the audio bitrate at 255Kbps using `-b:a 255k`. To ensure smooth video delivery during strong bitrate variations, we set the encoder's buffer at 1.25M using `-bufsize 1.25M`. We standardize the video playback frame rate at 30 frames per second using `-r 30`. We use `-x264-params "keyint=30:min-keyint=30:no-scenecut"` to guarantee a keyframe at least once every second (`min-keyint=30`) and use `no-scenecut` to disable scene cut detection. Here, we note that by preventing scene cut detection, we maintain consistent visual quality across the video by avoiding abrupt changes in bitrate or quality due to detected scene changes.

C. AOLME DATASET DESIGN

The greatest challenge with preparing real-life datasets comes from the need to limit the amount of required ground truth. This requirement is further complicated by our need to localize the activity by identifying where it is happening and the need to associate student participants with each activity.

For this paper, we wanted to avoid the need for face recognition of each student. We refer to [12], [17], and [18] for a summary of our efforts to apply face recognition to real-life classroom videos. We note that recognizing typing and writing activities does not require face recognition and tracking. Instead, we use a simple initialization process that requires the users to define rectangular hand regions over a single frame associated with each video scene. Here, we define a video scene to be a particular scene arrangement of the students. On the infrequent occasions when the students get up and move to new positions, we require re-initialization over the first frame of the new scene. We defer to future work on our efforts to automate scene initialization. However, even in future work, we will want to keep developing systems with humans in the loop to review the results, challenge assumptions, and provide for simple and effective scene re-initializations.

We built an efficient collection of ground truth datasets that supported our goal of processing unedited real-life classroom videos. To support activity localization, we associated a single object with each activity. For typing activity detection, we selected keyboard detection. For writing activity detection, we selected hand detection. Our approach is simple. Object detection is used to provide video segments over which an activity can happen.

Our approach of associating activities with a single object proved very effective. First, it allowed us to use fast and reliable object-detection methods to locate the video activities. Second, once the object was detected, over each object region, we extracted small and short video segments for further processing. Third, for the purposes of this section, our approach led to efficient methods for providing ground truth.

By requiring object detection for video activity recognition, we were able to simplify the ground truth process into two basic problems. First, for object detection, we select representative image examples for each problem. Second, for activity recognition, we review short video segments over each representative object example. We describe the derived datasets in the following three subsections.

D. AOLME-FULL VIDEO DATASET

The AOLME project collected 987 hours of videos of student group interactions. We organized the videos into cohorts, levels, schools, and groups. We use cohort-1, cohort-2, and cohort-3 for videos collected in 2017, 2018, and 2019. Each cohort followed a different level of the curriculum. At each school, the students were organized into small groups, with 2 to 5 students per group. We record ten to twelve video sessions per level per student group.

We provide a compact labeling system to define the origin of each video. Our video session labeling provides the cohort (C1, C2, C3), implementation level (restricted to L1 right now), school identifier (P or W), and group letter (A, B, C, D), and date (Month day). Thus, a video session labeled as C1L1P-A, Mar. 02 refers to Cohort 1, Level 1, P school, Group A, Mach 2nd.

We reviewed the entire AOLME dataset to select representative examples of effective teaching practices, ensuring diversity in cohorts, schools, groups, and instructional approaches. Based on our review, we chose 45 sessions for typing and 30 sessions for writing. To evaluate our method's robustness, we split the dataset into training, validation, and testing groups based on sessions from different dates. Additionally, for validation and testing, we made sure to include sessions that hold greater significance for educational researchers.

E. TYPING ACTIVITY DETECTION DATASETS: AOLME-T, AOLME-TP, AOLME-TC, AND AOLME-TD

We designed the AOLME-T dataset to train, validate, and test typing activities (see Table 2). The dataset consists of 332 video segments selected from 45 sessions, selected from three different cohors and 17 separate student groups. For each session, we randomly labeled typing and no-typing activities using bounding boxes that dynamically track the location of the keyboard over time. In total, we have 479,550 frames, 4.4 hours of typing and 1,248,900 frames, 11.5 hours, of no-typing labels (see Table 4). Based on AOLME-T, we create the keyboard object detection dataset AOLME-TP by selecting a number of keyboard instances to form typing proposals (TP). We use the keyboard instances to select video segments from AOLME-T to form the AOLME-TC for training the typing activity classifiers (TC). For final

TABLE 2. Summary of sessions used to train, validate and test typing activity. Validation sessions are shown in yellow background. Testing sessions are shown in green background. Full video sessions for which there is ground truth over the entire duration of the captured video are shown in boldface. The remaining unmarked sessions are used for training.

Group	Dates
C1L1P-A	Apr. 06, Apr. 13, Feb. 16, Mar. 02, Mar. 09, Apr. 20, Feb. 25
C1L1P-B	Apr. 27, Mar. 09, May 06, Mar. 02, Mar. 30, Apr. 06, May 11, May 04
C1L1P-C	Feb. 25, Mar. 09, Apr. 20, **May 04**, **Apr. 13**, Mar. 02, **Mar. 30**, Feb. 16
C1L1P-D	Apr. 06, Mar. 09
C1L1P-E	Feb. 25, **Mar. 02**
C1L1W-A	Feb. 28, Mar. 28, Feb. 21, Apr. 25, Mar. 07
C1L1W-B	May 06
C1L1W-C	Feb. 21
C1L1W-D	Feb. 28
C2L1P-B	**Feb. 23**
C2L1P-C	Apr. 12
C2L1P-D	**Mar. 08**
C2L1W-A	Apr. 10
C2L1W-B	**Feb. 27**
C3L1P-C	**Apr. 11**
C3L1P-D	**Feb. 21**
C3L1W-D	**Mar. 19**

testing of the entire system on raw videos, we select two complete sessions (AOLME-TD) as given in Table 2.

We carefully reviewed the video sessions in AOLME-T to select representative keyboard examples for the AOLME-TP dataset. Our approach was to consider keyboard image samples every minute of every video. We ended up with 1,728,000 images (size = 858 × 480), summarized in Table 4. We then removed images where the keyboard is not visible. At the end, we selected 1,448 representative keyboard instances. The final AOLME-TP consisted of 700 images for training, 100 for validation, and 648 for testing, selected from different sessions.

The AOLME-TP dataset was used to prepare short video segments for typing classification (see Table 4). We use the term AOLME-TC to refer to the video segments extracted from the general dataset (AOLME-T). This dataset includes typing and no-typing instances, each standardized to a duration of 3 seconds. We chose this duration based on the observed range, where typing instances varied from a maximum of 284 seconds to a minimum of 3 seconds. Using the minimum duration as a reference, we applied spatio-temporal cropping. Spatial cropping involved computing

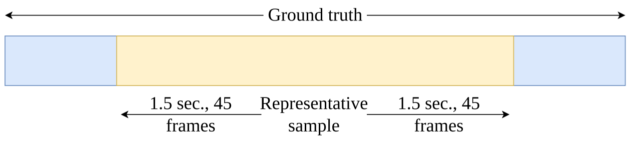


FIGURE 3. AOLME-TC: The creation of typing/no-typing video segments based on temporal cropping around the representative keyboard sample.

TABLE 3. Summary of sesssions used to train, validate and test writing activity. Validation sessions are shown in yellow background. Testing sessions are shown in green background. Full video sessions for which there is ground truth over the entire duration of the captured video are shown in boldface. The remaining unmarked sessions are used for training.

Group	Dates
C1L1P-B	Mar. 03
C1L1P-C	**Mar. 30**, Apr. 06, **Apr. 13**, Feb. 16, Feb. 25, Mar. 09, **Apr. 20**, May 04, May 11
C1L1P-D	Mar. 09, Mar. 02, **Mar. 30**, Apr. 06
C1L1P-E	**Mar. 02**
C1L1W-A	Feb. 14, Feb. 21, Feb. 28, Apr. 04
C2L1P-B	Feb. 23
C2L1P-C	Apr. 12
C2L1P-D	**Mar. 08**
C2L1P-E	Apr. 12
C2L1W-A	Feb. 20, Apr. 10
C2L1W-B	Feb. 27
C3L1P-C	**Apr. 11**
C3L1P-D	Feb. 21, Feb. 14
C3L1W-D	**Mar. 19**

the bounding box that contains the union of bounding boxes from the individual frames. Temporal cropping is centered around the keyboard image as described in Fig. 3. AOLME-TC consisted of a total of 405 typing and 398 no-typing instances, which correspond to approximately 324,000 typing frames and 817,200 no-typing frames. As for all of our datasets, training, testing, and validation are based on different sessions (dates) from AOLME-T.

F. WRITING ACTIVITY DETECTION DATASETS: AOLME-W, AOLME-WP, AOLME-WC, AND AOLME-WD

We provide a summary of the datasets associated with writing in Table 3. Writing detection requires that we label writing activities associated with 2 to 5 students in each session. The AOLME-W dataset consists of 166 videos selected from 30 sessions, taken from three cohorts and 14 student groups. To train the classifiers, we randomly selected writing and no-writing activities using dynamic tracking to track motions through time. In total, we selected 862,710 frames, 7.9 hours

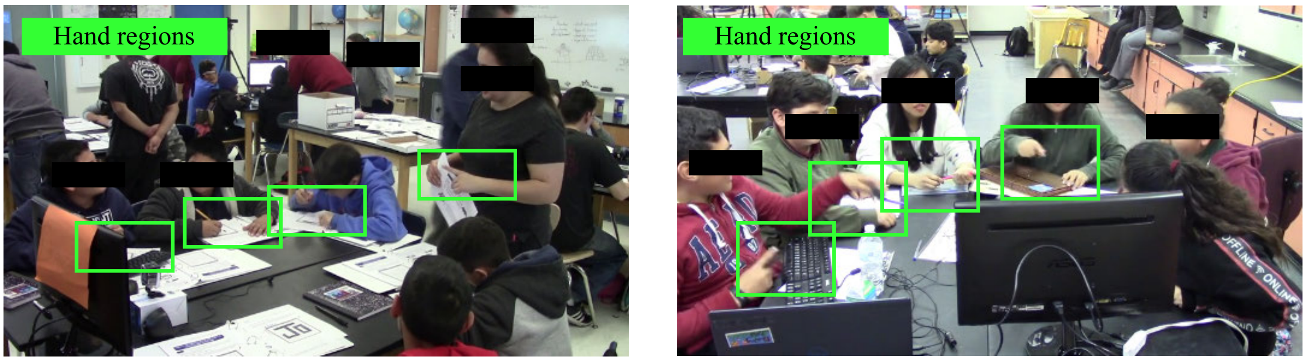
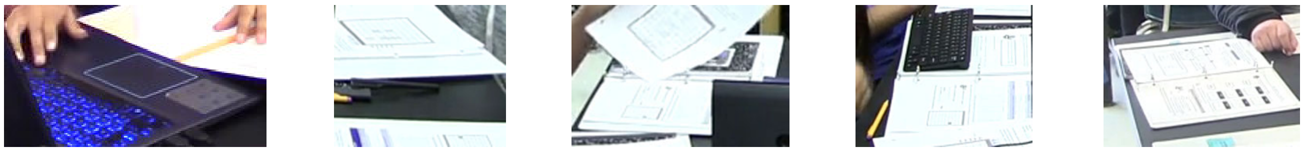
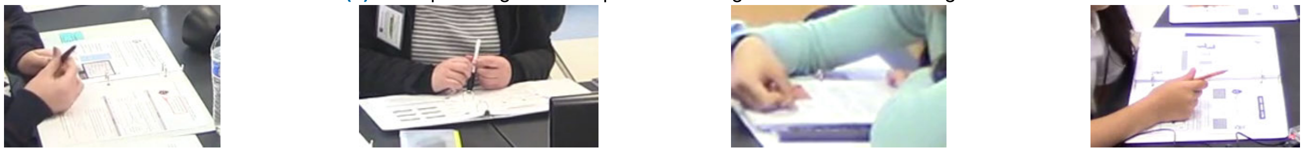


FIGURE 4. AOLME-WD: Sessions used to test our writing detection system.On the left we have the first session (WS1) from group E of cohort 1(2017) and level 1. On the right we have the second session (TS2)from group C of cohort 3 (2019) and level 1.



(a) Example images of samples illustrating the absence of writing.



(b) Challenging examples of samples illustrating the absence of writing. As can be observed in the figure, these samples feature hands executing movements similar to those associated with writing.

FIGURE 5. AOLME-WC: No-writing example segments.

long for writing and 2,639,460 frames, 24.4 hours, for no-writing segments (see Table 4). For writing, we relied on hand instances to extract video segments for training our writing activity classifiers. Similarly, for the final testing, we labeled two complete raw sessions for the final system testing (AOLME-WD). (see Fig. 4).

For hand detection, we developed the AOLME-WP dataset. We selected 1,803 hand instances for training, 714 for validation, and 2,031 for testing, extracted from different video sessions.

We use the hand instances to define short video segments to train the writing classifiers. Each video segment was 3-seconds long and extracted using spatio-temporal cropping as described for the typing dataset. We note that writing is a complex activity that may be confused with different movements (see Fig. 5). The full dataset, termed AOLME-WC, consisted of 1,199 writing and 798 no-writing instances, corresponding to approximately 107,910 writing frames and 71,820 no-writing frames.

III. BACKGROUND

A. OBJECT DETECTION AND TRACKING

Several methods have been developed for object detection and tracking. Popular methods for object detection include Faster RCNN [19], Single Shot Detector [20], and YOLO [21]. These methods are widely available in Detectron 2 [22] and OpenMMDetection [23] libraries. Popular methods for tracking include BOOSTING, MIL, KCF, TLD, MEDIANFLOW, GOTURN, MOSSE, and CSRT available in OpenCV [24]. We will consider all of these background methods for developing our video activity proposals.

B. VIDEO ACTIVITY CLASSIFICATION

We refer to [25] for a recent survey on human activity localization. Here, we provide a brief summary of popular video activity classification systems. In our results, we will compare against all of these popular methods.

We begin with the Two-Stream Inflated 3D ConvNet (I3D [7]). I3D extends successful 2D image classification architectures into 3D for classifying spatio-temporal features from video data. I3D accomplishes this by expanding 2D filters and pooling kernels into 3D layers, inserted between the original 2D layers. It uses two input streams—one for RGB data and another for optical flow— each initialized with 2D network weights before expansion. I3D achieved 80.9% accuracy on HMDB-51 and 98.0% on UCF-101. Similar to I3D, we will be processing short 3D video segments extracted over regions of interest.

TABLE 4. Summary of typing and writing datasets.

	AOLME-T						AOLME-TP	AOLME-TC			
	Typing			No-Typing			Keyboard	Typing		No-Typing	
	#frames	#inst.	Dur.	#frames	#inst.	Dur.	#inst.	#frames	#inst.	#frames	#inst.
Training	325,230	405	10,841	816,900	398	27,230	700	36,450	405	35,820	398
Validation	79,050	72	2,635	90,870	45	3,029	100	6,480	72	4,050	45
Testing	75,270	150	2,509	341,130	202	11,371	648	13,500	150	18,180	202
Total	**479,550**	**627**	**15,985**	**1,248,900**	**645**	**41,630**	**1,448**	**56,430**	**627**	**58,050**	**645**
	AOLME-W						**AOLME-WP**	**AOLME-WC**			
	Writing			**No-Writing**			**Hand**	**Writing**		**No-Writing**	
	#frames	**#inst.**	**Dur.**	**#frames**	**#inst.**	**Dur.**	**#inst.**	**#frames**	**#inst.**	**#frames**	**#inst.**
Training	467,760	727	15,592	1,108,590	311	36,953	1,803	65,430	727	27,990	311
Validation	133,890	189	4,463	334,620	89	11,154	714	17,010	189	8,010	89
Testing	261,060	283	8,702	1,196,250	398	39,875	2,031	25,470	283	35,820	398
Total	**862,710**	**1,199**	**28,757**	**2,639,460**	**798**	**87,982**	**4,548**	**107,910**	**1199**	**71,820**	**798**

TSN [6] is a framework for video-based action recognition that emphasizes long-range temporal structure modeling. In TSN, videos are divided into K equal segments, with frames sampled from each segment and aggregated to make a final action prediction. TSN achieves 69.4% on HMDB51 and 94.2% on UCF101. This sparse sampling approach works well for activities with distinct temporal patterns and defined starting and ending points. Unfortunately, our datasets do not have well-defined starting and ending points.

The Temporal Shift Module (TSM [8]) claims efficient 3D CNN-level performance with the lower complexity of a 2D CNN. TSM performs temporal modeling by shifting a portion of the channels along the temporal axis. The design supports both offline and online video recognition. TSM achieved 74.1% accuracy on Kinetics, 95.9% on UCF101, and 73.5% on HMDB51 offline, with similarly high performance online. While TSM is efficient, its pseudo-3D approach of shifting channels rather than performing full 3D convolution limits its ability to capture the fine-grained temporal details needed for intricate activities.

The SlowFast model is a video analysis architecture with two pathways. The Slow pathway processes videos at a lower frame rate to capture spatial details. The Fast pathway processes videos at higher frame rates to capture video motions. This dual pathway enables SlowFast to excel in action classification and detection by extracting complementary spatio-temporal features. Motivated by the SlowFast model, in our approach, we will learn a single, optimal frame rate that is tuned to each specific activity that we are interested in.

IV. METHODOLOGY

We provide a top-level diagram of the proposed system in Figure 6. We note that the input is a real-life classroom video session. We used independent streams for each activity. Our approach allowed us to optimize each stream for the specifics of each activity. Videos are processed in three stages. First, the video activity proposal network (VAPN) generates candidate video segments for each type of activity. Second, a classifier is used to determine whether the activity is happening within the video segment. Third, we create a video visualization map for each activity. We have obtained informed consent from parents and assent from minors for processing the videos described in this paper.

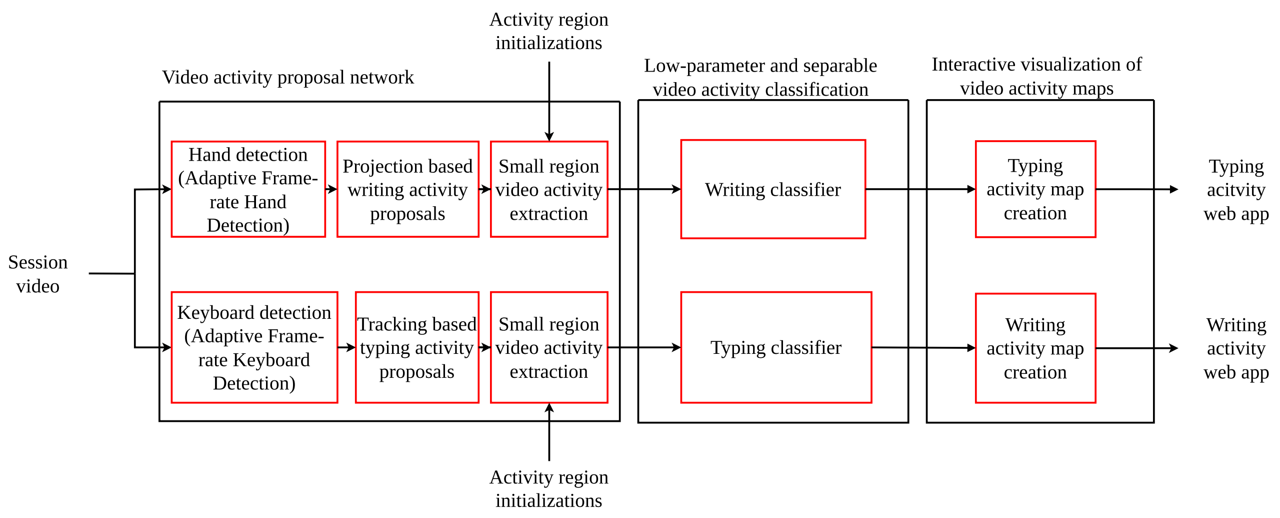
We describe the methodology using four separate sections. In section IV-A, we cover the full multiobjective optimization framework. In section IV-B, we describe the video activity proposal network. In section IV-C, we describe our video activity classification network. In section IV-D, we describe our approach for interactive visualization of video activity maps.

A. A MULTIOBJECTIVE OPTIMIZATION APPROACH TO SYSTEM DESIGN

In this section, we will develop a multiobjective optimization approach for optimizing different requirements and components of our system. Firstly, we are interested in minimizing the amount of ground truth required to train our system. Secondly, we would like to minimize the amount of training time. Thirdly, we are interested in optimizing inference to produce a fast and accurate system that produces generalizable results.

1) MINIMIZING THE REQUIRED AMOUNT OF GROUND TRUTH

We begin with the requirement to minimize the required amount of ground truth. As discussed earlier, our approach is to train separately for each activity and each system component. Our approach is summarized in Table 5. We only need to train for our video activity proposal network (VAPN) and our video activity classification network (VACN). For VAPN, we only need ground truth on still image datasets



Activity region initializations
Video activity proposal network
Low-parameter and separable video activity classification
Interactive visualization of video activity maps
Session video
Hand detection (Adaptive Frame-rate Hand Detection)
Projection based writing activity proposals
Small region video activity extraction
Writing classifier
Typing activity map creation
Typing acitvity web app
Keyboard detection (Adaptive Frame-rate Keyboard Detection)
Tracking based typing activity proposals
Small region video activity extraction
Typing classifier
Writing activity map creation
Writing activity web app
Activity region initializations

FIGURE 6. System diagram of activity detection system for typing andwriting in AOLME group interaction videos.

TABLE 5. Component-based training to minimize the required amount of ground truth.

Method	Typing Activity Recognition Datasets See Sec. II-E.	Writing Activity Recognition Datasets See Sec. II-F
Activity Prop. (VAPN)	AOLME-TP: keyboards annotated using sampled video frames for training and validation.	AOLME-WP: hands annotated using sampled video frames for training and validation.
Classif. Net (VACN)	AOLME-TC: short video segments over keyboard regions for training and validation.	AOLME-WC: short video segments over hands regions for training and validation.
System Visualization	AOLME-TD: full classroom videos for typing visualization and end-to-end testing.	AOLME-WD: full classroom videos for typing visualization and end-to-end testing.

where we annotate the object of interest (keyboards or hands). For VACN, we only need ground truth over short video segments of the keyboard and hand regions. Nevertheless, we still need ground truth over entire video sessions for end-to-end testing of the entire system. However, the key advantage of the proposed approach is that we do not require large ground truth datasets for end-to-end training.

Our component-based training approach has significant advantages over the standard use of end-to-end training and testing. First, we note that our component-based approach requires significantly less amount of ground truth. To see this, we note that end-to-end training requires large video datasets that capture all possible variations and their combinations. Thus, note that the required number of ground truth examples grows as the product of the number of examples required for each considered variation. For example, to capture N object variations and M object motions, we need to generate ground truth for $N \cdot M$ videos. In contrast, our component-based approach uses N object images that capture object variations for training the video activity proposal network and M video segments for the video activity proposal networks. Clearly, our component-based approach grows as $N + M$ where N represent still image examples. Second, our component-based approach is modular, easier to debug, and easier to optimize. In terms of modularity, we note that we can improve performance by simply replacing each component with a better version. In terms of debugging, we note that we can visualize the performance of each component separately, visualizing their successes as failures independent of any other component. In terms of optimizing, we note that we can optimize each component independendly.

Our separable approach enabled us to perform precise troubleshooting and targeted improvements. For instance, when training keyboard detection using a public dataset, we observed that the detector underperformed for wireless keyboards. This was due to the public dataset initially containing more wired keyboard instances. By identifying this imbalance through modular debugging, we were able to add more wireless keyboard training examples, leading to improved detection performance.

2) MINIMIZING TRAINING TIME

We now turn to the problem of minimizing training time through the use of low-parameter models for component-based training. We have already covered how our use of component-based training has dramatically reduced training dataset sizes. In turn, lower dataset sizes resulted in dramatically reduced time for training for a single epoch. Then, our use of low-parameter models requires significantly less epochs than what is typically needed for larger-parameter models. In addition, our use of low-parameter models required significantly less memory for training. As a result, we were able to use significantly larger batch sizes than the larger-parameter models that we were comparing against. Beyond using low-parameter systems, we also consider reducing the sampling rate of the input video. Thus, instead of processing the video at the original sampling rate of 30 frames per second, we also consider processing the video at 10 and 20 frames per second. This approach leads to additional speed-up that is ideally proportional to the frame-rate reduction. Thus, at 10 frames per second, we expect that the videos will get processed three times faster (=30/10). Overall, our low-parameter models required about 10 times less time to train as we document in the results.

3) OPTIMIZING MODEL PERFORMANCE

Beyond optimization training, we are also interested in optimizing model performance. More specifically, given the large video durations, we are interested in fast and accurate inference. Here, once again, we benefit from our use of low-parameter systems. As for training, our low-parameter system requires significantly lower memory and can clearly process larger batches of video segments. Furthermore, our lower-parameter models are also less complex and run much faster. Of course, we also benefit from the use of lower sampling rates as we do for training. Naturally, we would like to develop our fast models without sacrificing model performance. To effectively describe model performance at all possible operating points, we will use the area under the ROC curve (AUC). Our requirements lead us to consider multiobjective optimization, as we describe next.

We summarize our approach for optimizing each component in Algorithm 1. The goal of the algorithm is to jointly optimize for three different objectives: the number of parameters, AUC, and inference speed. Assuming everything else is equal, an architecture with a smaller number of parameters is preferred because it is expected to be better at generalization while leading to faster inference. Similarly, we are interested in systems that deliver the highest AUC and lowest inference time. Ideally, a system can optimize all three objectives at the same time. In this case, the optimal system uses the smallest number of parameters while also providing the highest AUC and requiring the minimum inference time. In this ideal scenario, the optimized architecture is better than any other architecture under consideration.

In general, multiobjective optimization results in a collection of optimized architectures. We refer to the collection of optimized architectures as the Pareto front (refer to [26] for details on the Pareto front). In Algorithm 1, we present the general algorithm that computes the full Pareto front of optimal architectures, frame rates, and corresponding performance metrics (objectives). For the algorithm, Sys refers to a parametrized model that returns a different architecture based on an integer configuration parameter C. In addition to the model Sys, the algorithm accepts C_range that represents a list of all possible values of C that need to be considered. Furthermore, to speed up inference time, we again consider the sampling rate of the input video in terms of the number of frames per second (FPS) and the number of videos used in each batch (B).

As shown in Algorithm 1, we initialize the Pareto front with the number of parameters, AUC, and inference time generated by all possible combinations of neural network levels and input video frame rates. We note that C also determines the number of parameters that will be used by the model. AUC and inference time are measured on the test set.

The Pareto front is calculated through a process of removal of suboptimal configurations. As shown in Algorithm 1, for each configuration on the Pareto front, we search for a better one that improves any one of the objectives without sacrificing any other (see lines 1 to 1). If a better configuration is found, then the currently considered configuration is removed from the Pareto front. At the end, we are left with all of the configurations that cannot be improved upon. They form the final Pareto front.

More precisely, we can formulate our computation of the Pareto front using multiobjective optimization. Our algorithm computes the solution to:

$$\begin{aligned}\min_{\mathrm{C,\,FPS}} \big(&\mathrm{Par}(\mathrm{Sys}(\mathrm{C})),\\ &-\mathrm{AUC}(\mathrm{Sys}(\mathrm{C}),\mathrm{FPS}),\\ &\mathrm{Inf}(\mathrm{Sys}(\mathrm{C}),\mathrm{FPS},\mathrm{B})\big).\end{aligned} \tag{1}$$

In (1), recall that the number of parameters is directly controlled by C. To achieve higher AUC, we place a negative sign in front of AUC so that the minimal values correspond to higher positive AUC values. On the other hand, AUC and inference speed depend both on the parameterized model and the input video frame rate. We assume that the AUC is not affected by the number of videos used in each batch (B). On the other hand, we note that the inference speed is affected by B.

4) SELECTING OPTIMAL CONFIGURATIONS

We now turn to the problem of determining optimal models for different scenarios. Optimal solutions are computed through a process of selecting a specific configuration from the Pareto front that is computed using Algorithm 1. We begin with the standard approach of optimizing for a single objective and then consider the problem of balancing among the objectives.

Algorithm 1 Pareto Front Algorithm

```
1: function OptSys,  C_range,  FPS_range, B_range
▷ Multiobjective optimization framework for
▷ simultaneously optimizing:
▷       the number of system parameters,
▷        AUC and inference time.
▷ Input:
▷ Sys represents the parametrized system
▷        to be optimized.
▷ C_range controls the system configurations.
▷ FPS_range provides a collection of frames per
▷        second for the input video.
▷ B_range provides the range of values for the
▷        batch size.
▷ Output:
▷ ParetoFront that saves the optimal values of
▷        Parameters, AUC, and inference time.
2: ▷ Evaluate all configurations
3:   for   FPS, C, B  in  FPS_range, C_range, B_range do
4:   Train Sys(C) using $B$ videos per batch
5:        sampled at FPS frames per second.
6:        Test Sys(C) using videos at FPS.
7:       Save Pars, Acc, Inf for given FPS, C, B.
8: end for
9:    ▷ Compute Pareto Front
10: Initialize ParetoFront with all Pars, Acc, Inf
11: for Pars_i, AUC_i, Inf_i ∈ ParetoFront do
12:    Look for better Pars_j, AUC_j, Inf_j such that:
13:    (Par_j<Par_i)&(AUC_j≥AUC_i)&(Inf_j≤Inf_i)
14:        or
15:    (Par_j≤Par_i)&(AUC_j>AUC_i)&(Inf_j≤Inf_i)
16:        or
17:    (Par_j≤Par_i)&(AUC_j≥AUC_i)&(Inf_j<Inf_i)

18:    if better Pars_j, AUC_j, Inf_j found then
19:        Remove Pars_i, AUC_i, Inf_i
20:        from ParetoFront
21:    end if
22: end for
23: return ParetoFront
24: end function
```

Consider the problem of maximizing the validation Area Under the Curve (AUC). The standard approach is to select the model configuration that solves:

$$\max_{\text{C, FPS}} \text{AUC(Sys(C), FPS)}. \tag{2}$$

Unfortunately, focusing on maximizing AUC alone can lead to unacceptably slow inference times or the requirement to train a large number of parameters over a large dataset. To avoid this scenario, we impose constraints on the maximum inference time and the number of system parameters. Let P_max denote the maximum number of parameters. Let Inf_max denote the maximum acceptable inference time. We reformulate (2) as a constrained optimization problem using:

$$\begin{aligned} \max_{\text{C, FPS}} \quad & \text{AUC(Sys(C), FPS)} \\ \text{subject to:} \quad & (\text{Par(Sys(C)} \le \text{P\_max) and} \\ & (\text{Inf(Sys(C), FPS, B)} \le \text{Inf\_max})\,. \end{aligned} \tag{3}$$

We refer to (3) as the **maximum AUC configuration**. Similarly, we select the **minimum inference time configuration** using:

$$\begin{aligned} \min_{\text{C, FPS}} \quad & \text{Inf(Sys(C), FPS, B)} \\ \text{subject to:} \quad & (\text{Par(Sys(C))} \le \text{P\_max) and} \\ & (\text{AUC(Sys(C), FPS)} \ge \text{AUC\_min}) \end{aligned} \tag{4}$$

where AUC_min refers to the minimum acceptable testing AUC value. Then, the **minimum parameter configuration** is selected using:

$$\begin{aligned} \min_{\text{C, FPS}} \quad & \text{Par(Sys(C))} \\ \text{subject to:} \quad & (\text{Inf(Sys(C), FPS, B)} \le \text{Inf\_max) and} \\ & (\text{AUC(Sys(C), FPS)} \ge \text{AUC\_min})\,. \end{aligned} \tag{5}$$

Our optimal configurations of equations (3), (5), and (4), can be computed by evaluating all possible configurations of the Pareto front.

Based on the established constraints, we also want to consider selecting a **balanced configuration** that takes into consideration tradeoffs between our three objectives. Once again, we want to impose objective constraints on the final configuration. Here, we consider a convex combination of the normalized objectives as given by

$$-\text{AUC} + \lambda_{\text{Inf}} \frac{\text{Inf}}{\text{Video\_dur}} + \lambda_{\text{Par}} \frac{\text{Par}}{\text{Par\_min}}, \tag{6}$$

where λ_{Inf} refers to the weight assigned to the inference time, λ_{Par} refers to the weight assigned to the number of parameters, Video_dur refers to the duration of the video, and Par_min refers to the minimum number of parameters among all models. Our normalization is aimed at providing intuitive meaning to the weights. For example, an inference time that is the same as the duration of the video implies that we may be able to process and stream the results in real time. In reporting our results, we will refer to the ratio Inf/Video_dur as the **inference speed** (e.g., in the results discussed in Table 8). Also, we will refer to the ratio Par/Par_min as the **parameter ratio** (see Table 8).

The full framework for selecting the **balanced configuration** is given by:

$$\begin{aligned} \min_{\text{C, FPS}} \quad & -\text{AUC(Sys(C), FPS)} \\ & + \lambda_{\text{Inf}} \frac{\text{Inf(Sys(C), FPS, B)}}{\text{Video\_dur}} \\ & + \lambda_{\text{Par}} \frac{\text{Par(Sys(C)}}{\text{Par\_min}} \\ \text{subject to:} \quad & (\text{Inf(Sys(C), FPS, B)} \le \text{Inf\_max) and} \end{aligned}$$

$$\begin{aligned}&(\mathrm{AUC}(\mathrm{Sys}(C), \mathrm{FPS}) \geq \mathrm{AUC\_min}) \text{ and}\\ &(\mathrm{Par}(\mathrm{Sys}(C) \leq \mathrm{P\_max}) . \qquad (7)\end{aligned}$$

Next, we will provide a short discussion about picking different parameters. Given our focus on processing long-duration videos, we are interested in relatively fast inference times. This suggests that we want our inference times to be at-least as long as the duration of the video: Inf_max=Video_dur. For AUC, we want AUC to be well above 50%. We thus set AUC_min=0.6. To support training using common hardware without requiring large training datasets, we want to set the maximum number of possible parameters to about 100M. We note that much larger models can be loaded in memory with more recent hardware. Nevertheless, larger models require large ground truths for training or pre-training for transfer learning applications. Thus, we can set P_max=100M that will be satisfied by all of the models that we consider. For the weights, we note that $\lambda_{\mathrm{Inf}} = \lambda_{\mathrm{Par}} = 1$ will produce a reasonable configuration that balances all of the objectives. To emphasize AUC, we can set $\lambda_{\mathrm{Inf}} = \lambda_{\mathrm{Par}} = 0.1$. To emphasize inference speed, we can set $\lambda_{\mathrm{Inf}} = 10$ and $\lambda_{\mathrm{Par}} = 1$. To emphasize minimizing the number of parameters, we can set $\lambda_{\mathrm{Par}} = 10$ and $\lambda_{\mathrm{Inf}} = 1$.

B. VIDEO ACTIVITY PROPOSAL NETWORK (VAPN)

Our goal for the VAPN is to use fast object detection methods to detect regions of interest without compromising performance. To this end, we consider the use of methods that have been pre-optimized for speed. We note that the VAPN will provide the input to the VACN and hence, the VACN will not be able to detect any activities if the VAPN fails to detect a true object. On the other hand, if VAPN provides false positives, the VACN can be used to remove them. In terms of our multiobjective optimization framework, the VAPN is optimized to have a small number of false positives while allowing false negatives to be later handled by the VACN. In what follows, we summarize the different VAPN configurations that were considered and explain how the final components were selected. We will not detail the full multiobjective optimization approach here. For more details, refer to [27].

For VAPN, we need to consider different configurations for object detection, post-processing, tracking, and the operating frame rate. For object detection, we considered fast methods based on Faster RCNN [19], Single Shot Detector [20], and YOLO [21]. For both keyboard and hand detection, we found that Faster RCNN gave the best results, with a small number of false negatives and a reasonable number of false positives. To improve accuracy for hand detection, we averaged detections over 12 frames. Furthermore, after averaging, we removed detections with small areas (area opening) to reduce the number of false positives, without increasing the number of false negatives.

Following object detection, we considered different methods for object tracking. We consider optimized methods for object tracking based on OpenCV: BOOSTING, MIL, KCF, TLD, MEDIANFLOW, GOTURN, MOSSE, and CSRT. Based on our testing, we got the best accuracy results using KCF [28]. As required by our multiobjective framework, we also determined optimal video sampling rates for each activity. For hand detection, we sample one frame per second. For keyboard detection, we sample one frame every 5 seconds.

To attribute an activity to a particular student, we compute the intersection over union (IoU) between the detected object and the rectangular regions associated with each student (initialized at the beginning of each scene). For IoU>50%, we associate an object detection with a specific student.

C. VIDEO ACTIVITY CLASSIFICATION NETWORKS (VACN)

We are interested in implementing the full multi-objective optimization framework for VACN. In terms of our multiobjective optimization framework, we note that we are interested in fast, low-parameter systems that can operate at optimal frame rates. We are naturally led to consider the optimization of 3D CNN architectures as opposed to significantly higher-parameter transformer networks. Here, we also note our use of VAPN also eliminates the need for transformer models because we do not need to model global interactions. We simply need to use our VACN to classify short video segments over the objects detected by VAPN.

We consider a family of 3D CNN architectures that uses 3D max pooling at each level. As a result, as the number of convolutional levels increases, successful pooling also reduces the number of required parameters for the final fully connected layer. As we document in the results, our approach leads to a parametrized 3D CNN architecture that improves performance as the number of levels increases while reducing the number of parameters at the same time. Furthermore, our models sample the input videos at different frame rates in order to capture coarse temporal characteristics of each activity. Then, at the activity-specific frame rate, we use a family of 3D CNNs to capture fine spatio-temporal features of each activity.

We consider a family of different 3D CNN architectures parametrized by the number of Dyad networks as shown in Fig. 7. Each dyad consists of 3D-ConvNet kernels, batch normalization, ReLU activation, and 3D max-pooling. For the D-th dyad, we use 2^{D+1} 3D-ConvNet kernels, as shown in figure 7. The maximum depth, D_{max} of our architecture depends on the size of the input video. For our dataset videos, which have a size of $3 \times 224 \times 224 \times fr$, where fr is the number of frames, the maximum depth that can be supported is 4. As shown in Fig. 7, we can support up to four depth levels. Overall, we explore three different rates at four depth levels for a total of 12 different processing models.

We will use multiobjective optimization framework to select optimal models as described in the results.

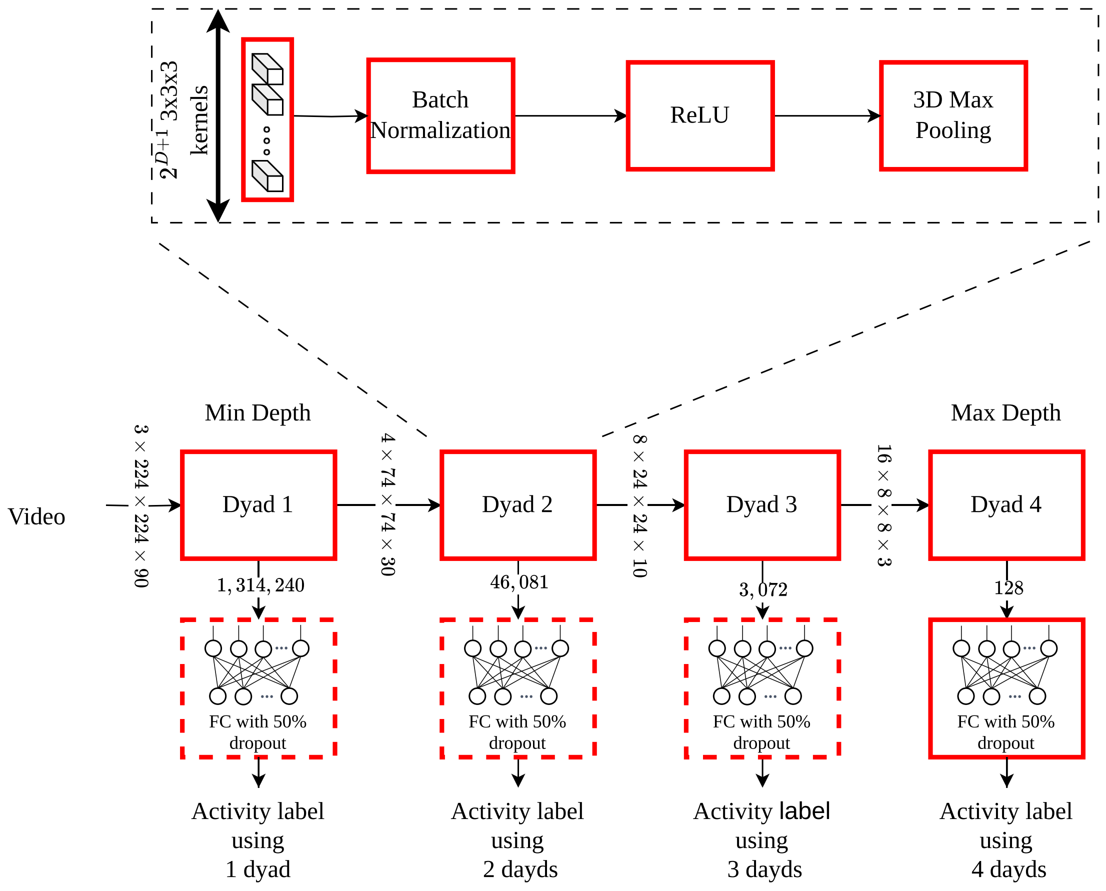


FIGURE 7. **Video activity classification networks based on dyadic architectures. A family of 4 low-parameter architectures were considered for determining whether an activity was present in each video segment.**

D. INTERACTIVE VISUALIZATION OF VIDEO ACTIVITY MAPS

Our interactive visualization of video activity maps is designed to be applicable to any number of activities. However, as stated in the introduction, we focus on recognizing writing and typing activities. We present the WebApp interface to visualize all typing activities over the entire video in Fig. 8.

The WebApp takes the original input with bounding boxes and time stamps that mark the activity within each video session. It then generates an interactive activity map with links back to the original input video. The WebApp associates a student pseudonym with each activity. For each student, we use a different row to summarize the results. Along each row, a sequence of bars marks the beginning and the ending of each activity.

The WebApp is highly interactive. When a user hovers over each bar, they can view the activity time interval. They can then click on each bar to activate a link that loads the video hosted on our AOLME server, allowing users to review the activity in question.

The Webapp provides controlled access. To access the WebApp, users must first register with the AOLME website. The system supports multiple concurrent users.

We provide an example in Figure 8b. The Figure shows several features of the interactive WebApp. We note that the displayed activity maps also provide zoom-in and zoom-out so that the users can focus on specific intervals.

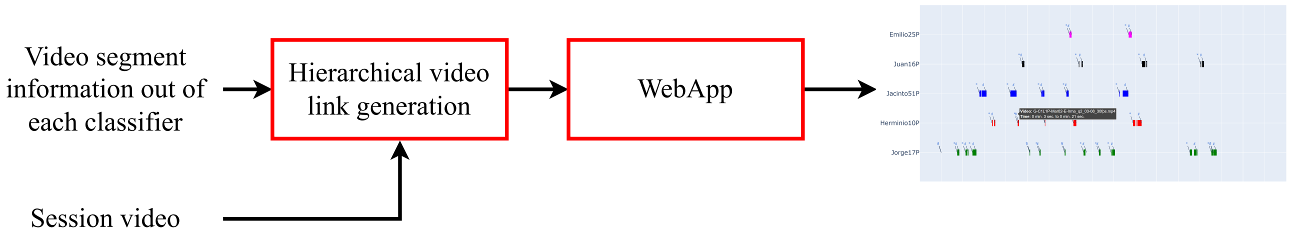
V. RESULTS

In this section, we provide detailed results on the optimization of the video activity classification network, comparisons against other methods, and interactive visualization of the results. In terms of performance, we used an Intel Xeon CPU running at 2.10 GHz and 128 GB of RAM. Our system used an Nvidia Quadro RTX 5000 GPU with 16 GB of video memory, which is considered to be lower-end according to standard benchmarks [29].

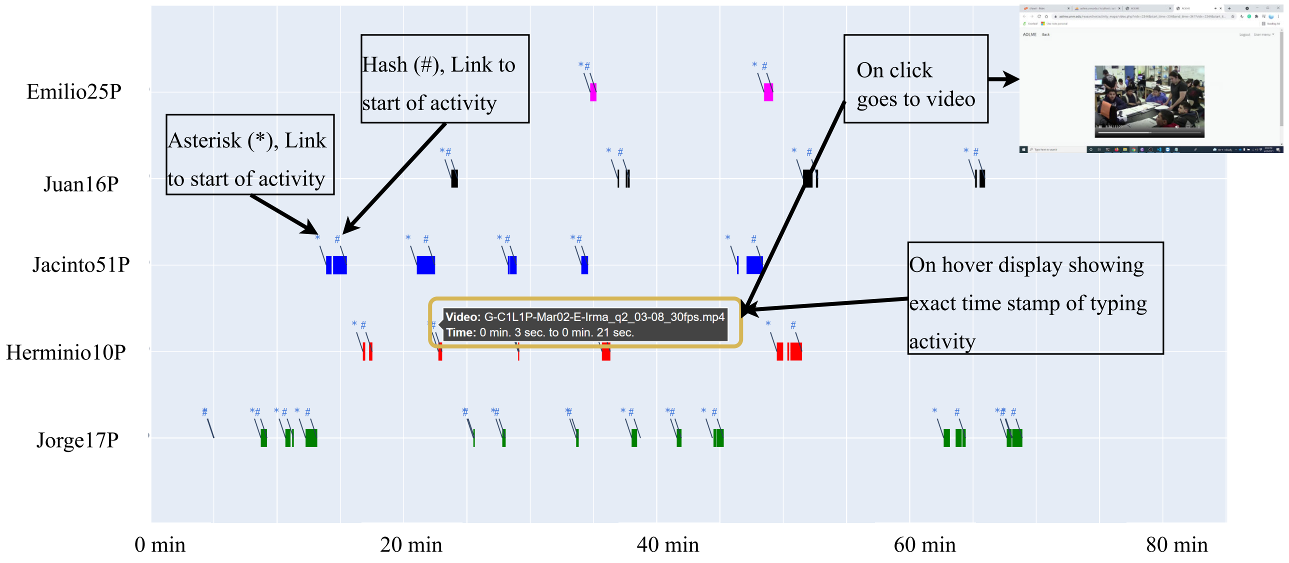
A. VAPN RESULTS

We present results from VAPN in Fig. 9. In what follows, we present results for keyboard and hand detections separately.

We note that training on wired and wireless keyboard examples worked well. As seen in Fig. 9(a), the VAPN can successfully detect wired and wireless keyboards under significant occlusions. On the other hand, the VAPN failed

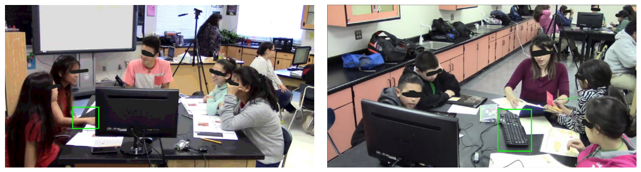

Video segment information out of each classifier
Hierarchical video link generation
WebApp
Session video
Emilio25P
Juan16P
Jacinto51P
Herminio10P
Jorge17P

(a) Interactive activity map visualization system diagram.

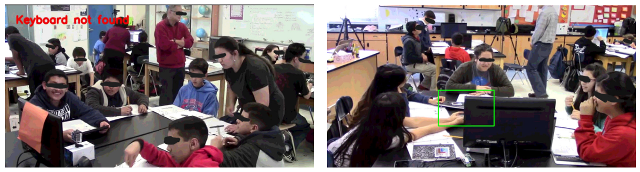


(b) Interactive typing activity map for C1L1P-E, Mar 02 session. It supports "on hover", "zoom", "selection" and "clickable" events. A user can use mouse to hover over the asterisks (*) and hash (#), which display exact location in the video. These symbols also serve as weblinks (requires AOLME account and IRB training), displaying the activity in a web browser.

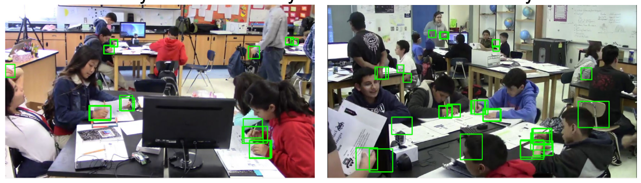
FIGURE 8. **System diagram and example of interactive activity map.**



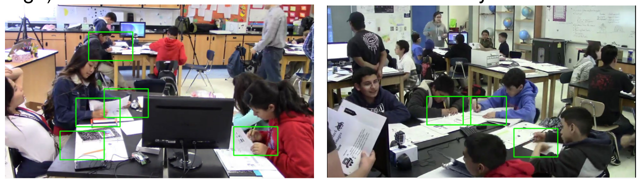
(a) True Positive (left image): Detection of wireless keyboard with significant occlusion. True Positive (right image): Detection of the rotated wired keyboard. Both keyboards were successfully tracked.



(b) False negative example (left image): Significant change in keyboard appearance was not detected. False Positive example (right image): Incorrect detection of dark notebook as a keyboard.



(c) Noisy hand detections before using projections. The images contain several false positives, where many of the detected hands belong to moving people or outside groups (away from the camera).



(d) Corrected hand detections after using projected-based filtering in the VAPN. The left image contains a false-positive in the upper left. All other detections represent true positives.

FIGURE 9. **VAPN results for keyboard detection (top images) and hand detection (bottom images).**

to detect keyboards under significant appearance changes (see left image and in Fig. 9(b)). The VAPN also confused a dark notebook for a keyboard (see the right image and in Fig. 9(b)). Overall, we achieved an average precision (AP) of 0.92 and average recall of 0.676 at 0.5 intersection over union (IOU).

TABLE 6. Multiobjective optimization results for VACN architectures in terms of the number of parameters, test-set accuracy, and inference speed. In this table, we report accuracy over the test-set. This accuracy increased with the test-set AUC. Inference speed is controlled in terms of the frame rate and the number of parameters. FPS refers to the number of frames per second of the input video. Thus, at 10 FPS, the network is applied to the video sampled at 10 frames per second. The number of parameters is controlled in terms of the number of dyads. We choose the highlighted configuration as the most balanced configuration (see text).

Num. of dyads	FPS	Num. Of Params	Typing Acc.	Writing Acc.
1	10	657K	53.33	39.93
2	10	47K	61.25	39.93
3	10	7.8K	61.25	57.34
4	**10**	**18.7 K**	**69.59**	**63.09**
1	20	657K	53.33	38.90
2	20	47K	53.33	61.09
3	20	7.8K	62.08	59.87
4	20	18.7K	65.83	63.22
1	30	657K	53.33	39.93
2	30	47K	53.33	39.93
3	30	7.8K	62.91	61.54
4	30	18.7K	67.91	64.05

Hand detection without the use of projections produced many false positives as shown in Fig. 9(c). On the other hand, our use of projection-based filterings reduced most of the false positives as shown in Fig. 9(d). Overall, hand detection achieved an average precision of 0.72 at 0.5 IOU. Refer to [27] for more details.

In terms of our overall system, we recall that the VACN can reduce the number of false positives by correctly classifying the activity over the wrong objects. On the other hand, our system cannot reduce the number of false negatives since they are not input to the VACN.

B. VACN TRAINING

For training, we use the Adam optimizer with an initial learning rate of 0.001, and use early stopping and video data augmentation techniques to prevent overfitting. During training, we apply data augmentation with 50% probability. Thus, for half the video segments, we apply uniformly random shearing (shearing factor between −0.05 and 0.05), rescaling (scaling factor between 0.9 and 1.2), rotation (angle between −7 and 7 degrees), translation between −5 and +5 pixels, and horizontal flips. Furthermore, we train each model for a minimum of 50 epochs and a maximum of 100 epochs, with early stopping applied after 50 epochs. The early stopping uses a patience of 5 epochs. In other words, after 50 epochs, we select the best model and terminate training if performance over the validation set does not improve after 5 epochs. Here, we note that our early stopping is used to avoid overfitting.

TABLE 7. Fast training using small and short video segments over regions of interest proposed by VAPN. The proposed CNN approach is the fastest, with a batch size of 16.

Method	Typing in minutes	Writing in minutes	Batch size
I3D	79 (3.04×)	75 (2.50×)	8
Slowfast	310 (11.92×)	279 (9.30×)	8
TSM	64 (2.46×)	58 (1.93×)	8
TSN	42 (1.62×)	38 (1.27×)	8
Ours	**26** (1×)	**30** (1×)	16

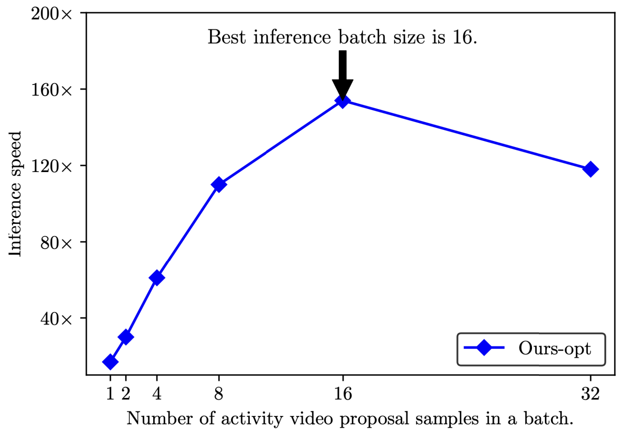


FIGURE 10. Inference speed optimization based on batch size.The optimal batch size is achieved at batch size=16.Inference speed is measured as the ratio of the output video frame rate divided by the input video frame rate. The optimal inference speed is at 154× which corresponds to 4,620 frames per second.

C. MULTIOBJECTIVE OPTIMIZATION OF VACN

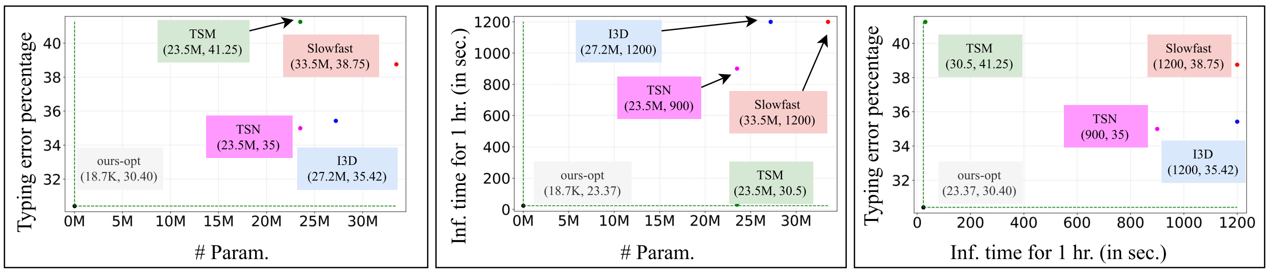
We performed multiobjective optimization to select an optimal VACN for each activity. As described in the methodology, we selected the optimal model-based on the test-set AUC, the number of parameters and inference speed.

We present results from our multiobjective optimization of the VACN architectures in Table 6. In terms of our multiobjective frameworks, we select the *maximum AUC configuration mode* that is represented by the maximum accuracy mode in Table 6. For typing, we achieved a test AUC of 0.95 and for writing, we achieved an AUC of 0.84. In terms of constraints, we note that at sampling at 10 frames per second, our chosen configuration is the fastest possible because it processes a smaller number of frames per second. In terms of the number of parameters, we note that we are using just 18.7K parameters which is a very small number (but more than the 7.8K for 3 dyads). Overall, in terms of balancing the constraints, we sacrifice an increase in the number of parameters to achieve significant improvements in the test-set accuracy.

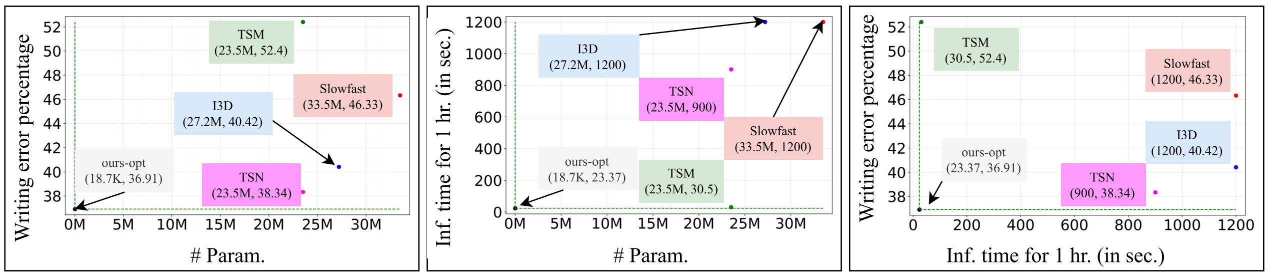
We further consider inference speed optimization by varying the batch size. The results are shown in Fig. 10. Here, batch size refers to the number of video segments.

TABLE 8. Comparative results for video activity classification in terms of the number of parameters, inference speed, and memory requirements. The model performance is measured using AUC and accuracy. We measure inference speed based on the original video playback speed (playback speed = 1×). The best performance is highlighted against a green background. The proposed approach uses over a 1000 times less parameters, requires far less memory, runs faster, and performs better than all other methods.

Method	# Param.	Inf. speed	GPU Mem. in MB	Test AUC	Test acc.	Detection AUC on complete sessions
	Classification on AOLME-TC dataset					**AOLME-TD sessions**
I3D	27.2M (1437×)	3×	5051 (20×)	0.66	64.58	0.75
Slowfast	33.5M (1787×)	3×	6318 (25×)	0.71	61.25	0.48
TSM	23.5M (1252×)	118×	6971 (28×)	0.59	58.75	0.77
TSN	23.5M (1252×)	4×	5593 (23×)	0.74	65	0.76
Ours-opt	**18.7K (1×)**	**154×**	**245 (1×)**	**0.76**	**69.58**	**0.83**
	Classification on AOLME-WC dataset					**AOLME-WD sessions**
I3D	27.2M (1437×)	3×	5051 (20×)	0.66	59.58	0.56
Slowfast	33.5M (1787×)	3×	6318 (25×)	0.57	53.67	0.48
TSM	23.5M (1252×)	118×	6971 (28×)	0.50	47.60	0.41
TSN	23.5M (1252×)	4×	5593 (23×)	0.62	61.66	0.45
Ours-opt	**18.7K (1×)**	**154×**	**245 (1×)**	**0.67**	**63.09**	**0.59**



(a) Pareto plots for typing classification.



(b) Pareto plots for writing classification.

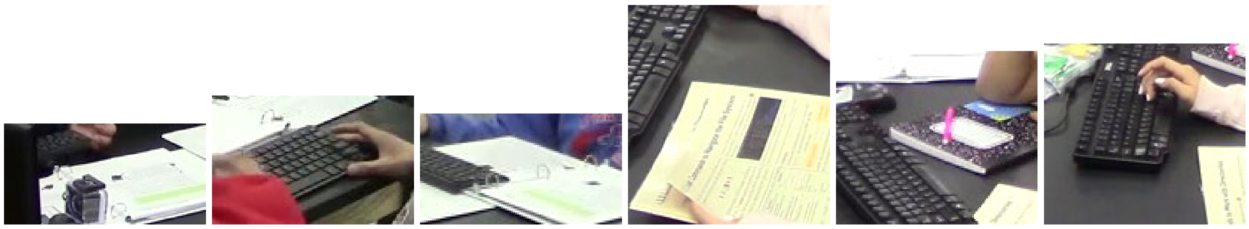
FIGURE 11. Multiobjective optimization results for typing (top images) and writing (bottom images).The results demonstrate that the optimized VACN is significantly better in terms of the number of parameters,testing accuracy (error rates), and inference time.To facilitate visualization, we plot three combinations of any two objectives. For all objectives, lower values are better.

We measure the inference speed as a ratio of the output frame rate to the input frame rate. Thus, in Fig. 10, an inference speed of 1× implies real-time video processing where the output frame rate is the same as the input frame rate. **The optimal network can perform inference at 4,620 frames per second (4,620=154*30) of** 154 × 30 **pixels per frame.**

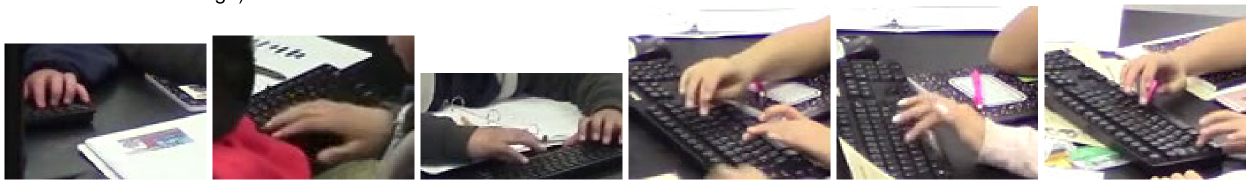
D. TRAINING PERFORMANCE COMPARISONS

Our optimization approach led to very fast training times (see Table 7). We note that this is primarily due to the fact that we train on small and short video segments over hand and keyboard regions.

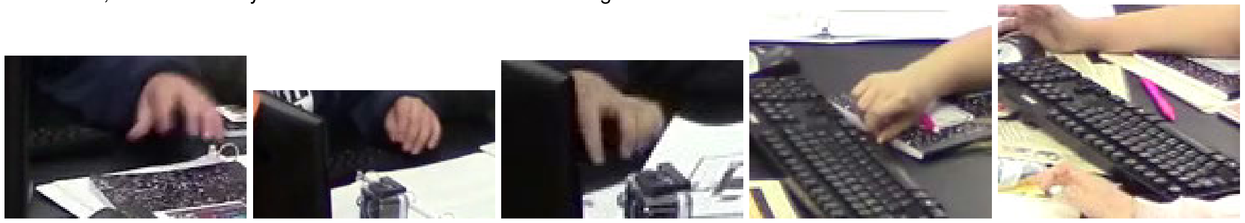
Our fast training results demonstrate the advantages of our approach. First, we recall that our approach does not require any pre-training. Despite the fact that we initialize from random values, our optimized VACN trained faster than any other method. Here, unlike all other methods, we did not perform GPU-based video decoding. Instead, we are only using CPU-based video decoding, which is somewhat slower. Yet, we still outperform every other method during training.



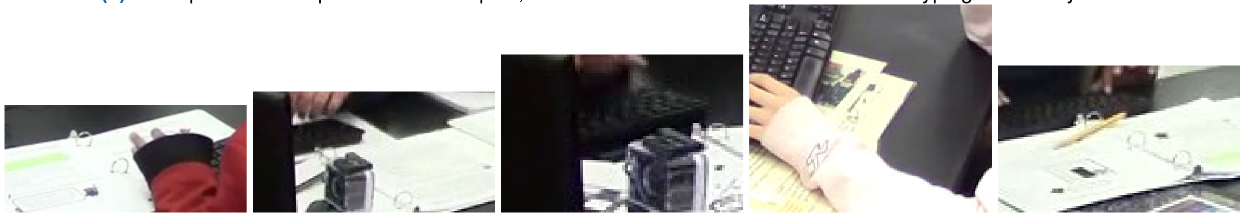
(a) True negative examples: VACN correctly labels the video segments as no-typing activities. We note that the VACN correctly identified the lack of typing activities despite the presence of fingers waiting to type over the keyboard (e.g., the second image from the left and the last image).



(b) True positive examples: VACN can detect typing in cases with severe occlusions, despite strong variations in keyboard and finger placements, and at a variety of different distances and video image sizes.



(c) False positive examples: In the examples, we have hand movements that resemble typing near a keyboard.



(d) False negative examples: In most cases, the keyboard and the hands are severely occluded. There is an example of moderate occlusion where hands movements are misclassified as typing (see 4th image from the left). In this case, the typing finger tips are occluded.

FIGURE 12. Typing classification examples over 90-minute sessions from AOLME-TD dataset.

E. INFERENCE MODEL COMPARISONS

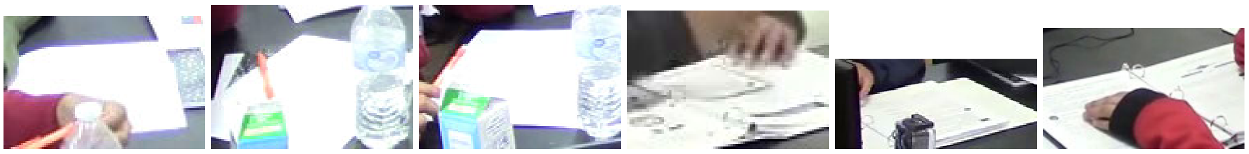
We compare our model against several popular state-of-the-art methods in Table 8 and Fig. 11. For comparison, we have trained all models on the same datasets. Furthermore, as described earlier, we performed inference speed optimization by varying the batch size for each method. As before, we measure performance using AUC and accuracy (for IoU=0.5). We also compare the number of trainable parameters, Graphical Processing Unit memory, and inference speed.

From the results, we note that our classifiers use extremely low parameter models. At just 18.7k, the proposed model uses 1252× to 1437× fewer parameters than any other model. Overall, the proposed models require 20× to 28× less memory than any other popular model considered in the literature.

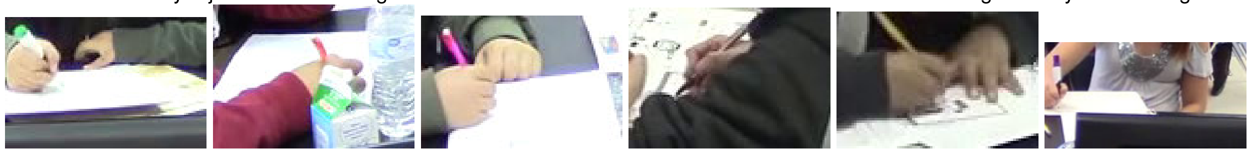
Our proposed models outperform every other model in terms of inference speed and classification performance. Over our full session datasets, our AUC results are significantly better than any other model. For typing, we achieved an AUC of 0.83. For writing, we achieved an AUC of 0.59, which is much lower. Nevertheless, our writing classification performance is significantly higher than any other model. In terms of our combined performance, we note that no other classifier gets close to our performance and speed. For example, TSM is much slower and achieved AUCs of 0.77 and 0.41 (compared against 0.83 and 0.59 for our models).

We demonstrate the significant advantages of our proposed approach in the Pareto plots of Fig. 11. For each plot, we note that our proposed approach is plotted in the lower-left corner. This implies that our approach outperforms all other methods in terms of every possible combination of inference speed, accuracy (given by 1 - error rate), or the number of parameters.

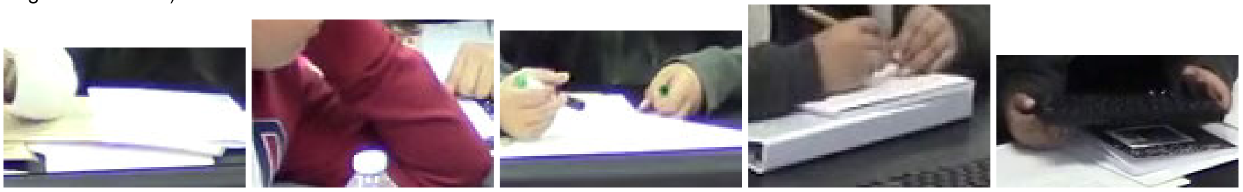
While other methods may approach our models' performance in any one objective, this comes at a significant cost at the other two objectives. And of-course, as mentioned earlier, without any pre-training on larger datasets, our proposed approach outperformed every other method while using more



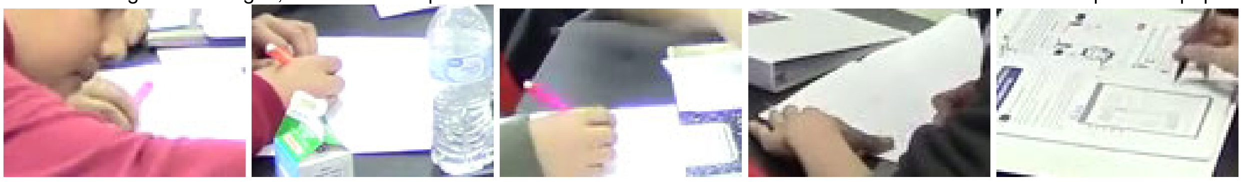
(a) True negative examples: Despite the presence of both hands and paper, the VACN correctly classifies the lack of writing motions. The VACN correctly rejects the lack of significant hand motions or cases where hand movement differs significantly from writing.



(b) True positive examples: The VACN accurately detects writing cases when the hand and pencil/pen are visible and exhibit slow, deliberate movements associated with writing. Note that the VACN works well even in cases with strong occlusions (e.g., the second image from the left).



(c) False positive examples: The VACN was found to be sensitive to hand movements that did not correspond to actual writing. In the leftmost and rightmost images, we show examples where VACN misclassified hand movements that did not involve a pencil or paper.



(d) False negative examples: The VACN missed cases of high occlusion (e.g., fourth image from the left), writing associated with large hand movements, or holding the pencil in a non-standard position (e.g., first image from the left).

FIGURE 13. Writing classification examples over 90-minute sessions of the AOLME-WD dataset.

than 1000× fewer parameters. In terms of inference speed, we note that TSM is also fast (e.g., see right plots in Fig. 11). However, we note that the TSM's error rate is significantly worse than our proposed approach (e.g., by 10 percent or more). Similarly, in terms of accuracy, TSN, although it is slightly worse, approaches our error rate for writing (see bottom-right plot in Fig. 11). However, we significantly outperform TSN in terms of accuracy for typing classification (see top-right plot in Fig. 11). At the same time, we are more than 38× faster in terms of inference speed! As shown in the multiobjective plots of Fig. 11, in terms of objectives, the remaining models perform significantly worse in every objective.

VI. FULL-SESSION VIDEO EXAMPLES (90 MINUTES)

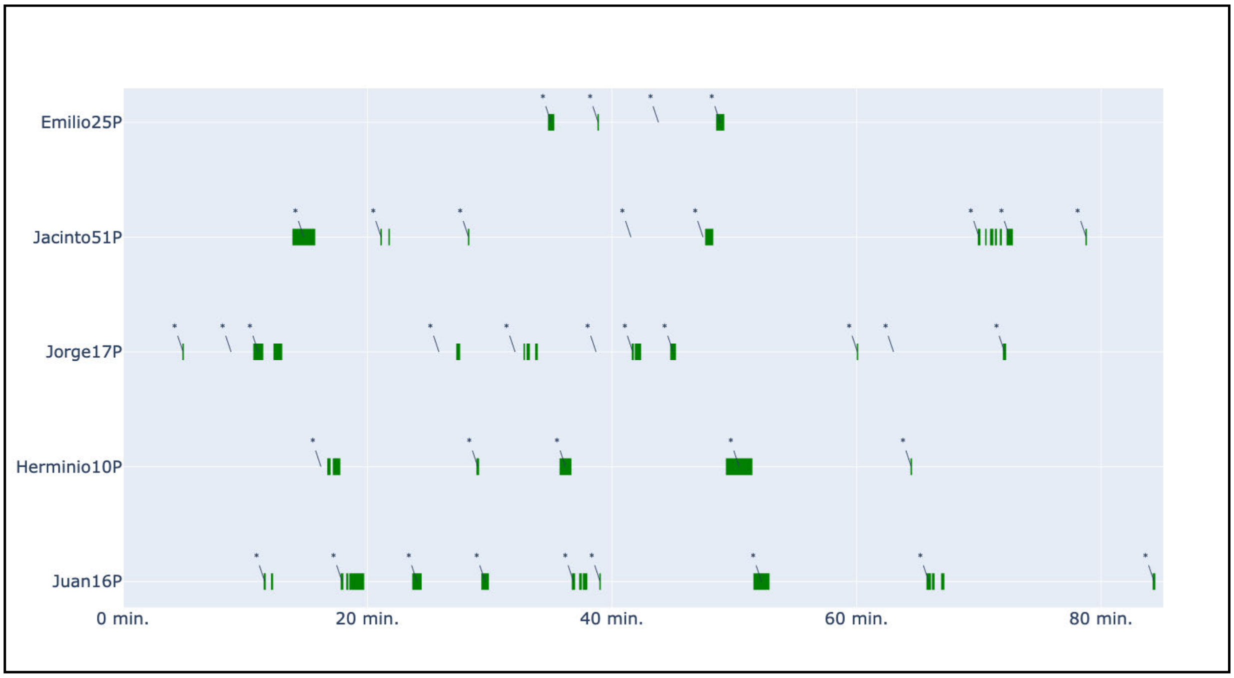
A. TYPING CLASSIFICATION

We present several typing classification examples in Fig. 12. The examples come from testing over 90-minute video sessions from the AOLME-TD dataset.

We note the strong diversity of our examples. In the majority of our examples, keyboards are highly occluded. We also present some examples where hands are severely occluded. Our examples also include complicated hand motions over keyboards. Overall, we note that our dataset captures the strong diversity of angles, positions, and occlusions associated with typing activities.

Our VACN correctly detects typing in a variety of complex scenarios. For example, in Fig. 12(a), we have examples of hand presence over keyboards that were correctly classified as true negatives due to the lack of typing motions. Similarly, despite strong occlusions, typing movements were correctly classified as true positives in Fig. 12(b). We also present complex examples of misclassifications in Fig. 12(c) and Fig. 12(d). In Fig. 12(c), we present examples of hand movements that resembled typing that were misclassified as typing. In Fig. 12(d), we present examples of typing under occlusion that were missed. In the fourth image from the left of Fig. 12(d), we note that the fingers associated with typing fell outside the VAPN detection region.

Overall, our method performed very well in cases with strong occlusions where typing remained clearly visible. Our method performed badly in cases with severe occlusion with limited visibility of the fingers involved in the typing action. We have also found a limited number of cases where finger motions over the keyboard region were misclassified as false positives. To improve performance for such cases, there is a need to include training examples with non-typing finger motions located over the keyboard region. In terms of our



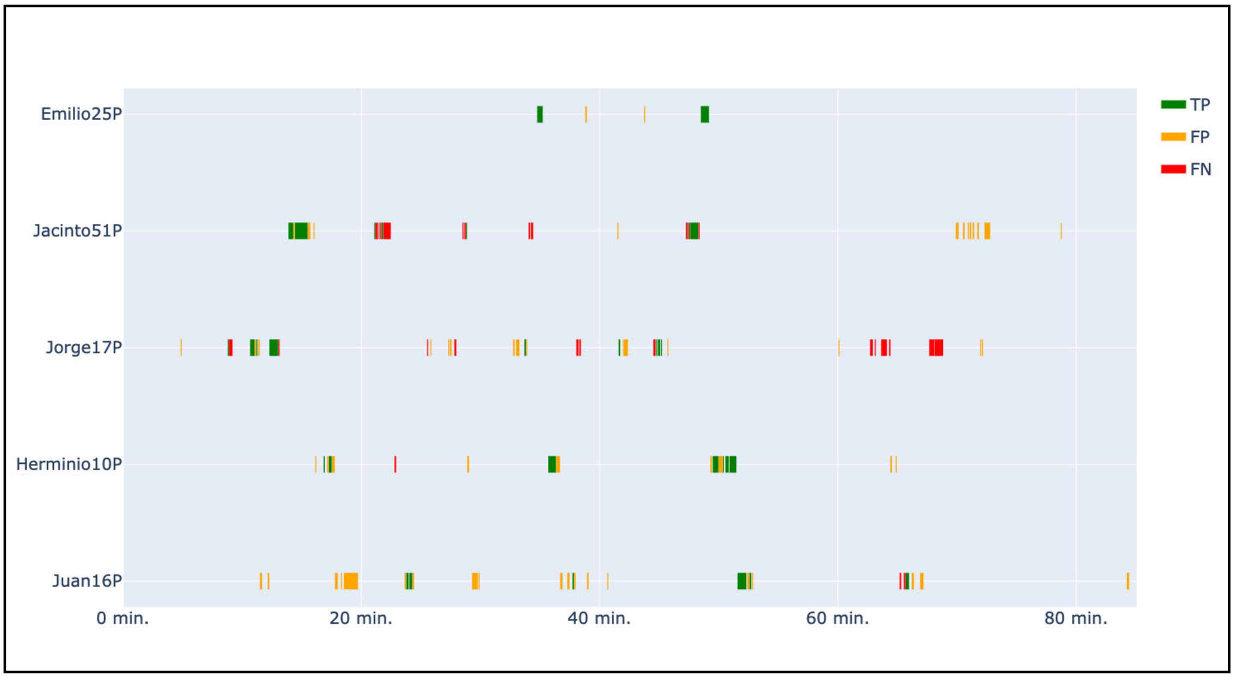


FIGURE 14. AOLME-TD visualization for typing. The interactive WebApp interface is shown in the top figure. Results against ground truth are shown in the bottom figure. For the bottom figure, we have run a 10-second median filter to reflect the intended use of our system. For the bottom figure, TP refers to true positives, FP refers to false positives, FN refers to false negatives. Note that everything else represents a true negative.

overall system, we hope that false positives will be identified during the final review using interactive visualization.

B. WRITING CLASSIFICATION

We present several examples of writing activity classification in Fig. 13. The examples come from testing over 90-minute writing sessions from the AOLME-WD dataset.

We note the diversity of testing examples in the AOLME-WD dataset. Regarding placement, we note severe occlusions, pose variations, and strong variations in camera angles. In terms of motions, in addition to writing, we had strong variations, including students playing with their pencils and making different gestures.

The VACN correctly rejected cases with little or no hand movements as true negatives (see Fig. 13(a)). Despite

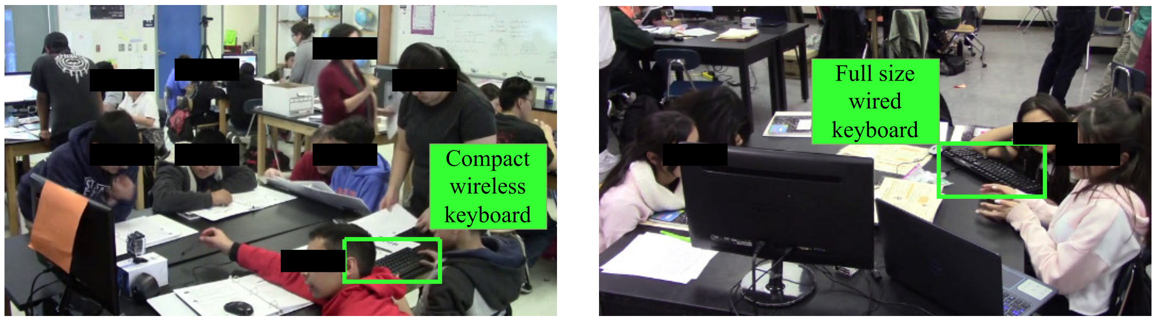
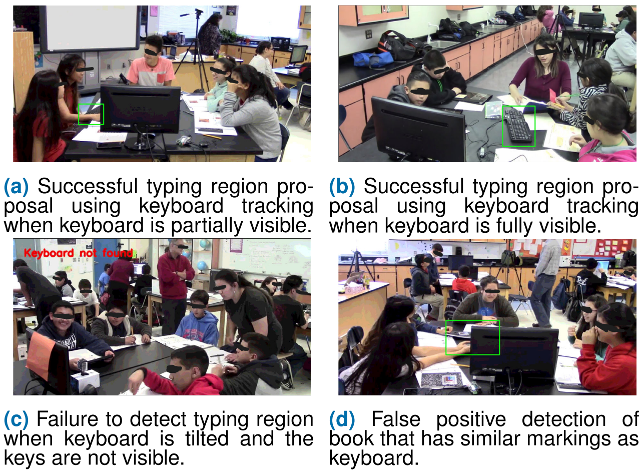


FIGURE 15. AOLME-TD: Typing detection results visualization over real-life classroom videos. Our system was able to detect and track two different types of keyboards.



(a) Successful typing region proposal using keyboard tracking when keyboard is partially visible.
(b) Successful typing region proposal using keyboard tracking when keyboard is fully visible.
(c) Failure to detect typing region when keyboard is tilted and the keys are not visible.
(d) False positive detection of book that has similar markings as keyboard.

FIGURE 16. Typing activity detection visualization examples.

occlusions and strong pose variations, the VACN correctly labeled writing movements as true positives (see Fig. 13(b)). Yet, the VACN incorrectly classified students playing with their pencils and other complex hand movements as writing activities (see Fig. 13(c)). The VACN also missed writing activities in cases of extreme occlusions or large writing movements (see Fig. 13(d)).

Overall, our method performed well in cases where writing is associated with relatively slow hand movements that are visible despite severe occlusions. We missed cases where writing is associated with large movements (e.g., drawing) and cases where the majority of hand movements were severely occluded. To improve performance for such cases, as before, there is a need to extend the training dataset to include examples of complex hand motions that are not associated with writing (e.g., drawing, playing with the pencil, etc). As before, we hope to identify these false positives during the final review using interactive visualization.

C. INTERACTIVE VIDEO ACTIVITY VISUALIZATION USING ACTIVITY MAPS

We present typing activity results over a 90-minute video in Fig. 14. The users can zoom in and out in different parts of the video and click to inspect specific portions of the video activity as demonstrated in 15 and 16. Our interactive system allows users to review educational activity patterns and recognize collaborative learning patterns within each student group. Here, we focus on interpreting the video activity recognition results.

We analyze the global typing activity recognition results using Fig. 14. In the top plot of Fig. 14, we can visualize typing patterns over 90 minutes of real-life classroom videos. The users can then click on the map to verify the results. The bottom plot of Fig. 14 displays a detailed comparison of the detected typing activity against ground truth. From the results, it is clear that our system is able to recognize the general typing patterns. We note that false positives can be easily removed during the interactive review process. Even for our false negatives, we note that there is often a small number of nearby true positives that will direct the users' attention to the general video segment where the activity is happening. Hence, during the interactive review process, we can eliminate several of our false negatives by reviewing longer video segments around our true positives.

We also present results during our interactive review process in Figs. 15 and 16. In Fig. 15, we can see that our system successfully detected typing activity around a compact wireless keyboard (on the left) and a full size wired keyboard (on the right).

We interactively zoom in to visualize specific results as demonstrated in Figs. 15 and 16. In Fig. 15, we can see that the system successfully detected two types of keyboards under partial occlusion. We show more examples in Fig. 16. We note that our proposed system was able to detect typing activities in challenging scenarios (e.g., see Figs. 16(a)-(b)). An example of failure to detect a typing activity is shown in Fig. 16(c). In this example, our system could not identify a keyboard for which the keys were not visible. In Fig. 16(d), we show an example of hand activity being classified as typing.

VII. CONCLUSION

Our paper described the development of a video activity recognition system that can efficiently process

90-minute real-life classroom video sessions. Our approach focused on the development of separable video activity recognition systems that use an extremely low-number of trainable parameters. At 18.7k, our optimized video activity classification system uses more than 1,000 times less parameters than any of the popular state-of-the-art video activity recognition systems. At the same time, our proposed system runs significantly faster and performs significantly better than any of the popular state-of-the-art video activity recognition systems. The key idea here is that our proposed approach allows us to optimize processing frame rates and minimize parameters that allows us to target specific activities, as opposed to the standard practice of using transfer learning from large-scale video activity recognition systems. Furthermore, we have developed an interactive WebApp that allows educational researchers to visualize video activity patterns over very long (90-minute) real-life classroom videos.

REFERENCES

[1] M. Korban, S. T. Acton, P. Youngs, and J. Foster, "Instructional activity recognition using a transformer network with multi-semantic attention," in *Proc. IEEE Southwest Symp. Image Anal. Interpretation (SSIAI)*, Mar. 2024, pp. 113–116.

[2] E. Dimitriadou and A. Lanitis, "Using Student action recognition to enhance the efficiency of tele-education," in *Proc. 17th Int. Joint Conf. Comput. Vis., Imag. Comput. Graph. Theory Appl.*, 2022, pp. 543–549.

[3] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray, "Scaling egocentric vision: The EPIC-KITCHENS dataset," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Jan. 2018, pp. 720–736.

[4] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black, "Towards understanding action recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 3192–3199.

[5] K. Soomro, A. Roshan Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," 2012, *arXiv:1212.0402*.

[6] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks for action recognition in videos," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 11, pp. 2740–2755, Nov. 2019.

[7] J. Carreira and A. Zisserman, "Quo vadis, action recognition? A new model and the kinetics dataset," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4724–4733.

[8] J. Lin, C. Gan, and S. Han, "TSM: Temporal shift module for efficient video understanding," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 7082–7092.

[9] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "SlowFast networks for video recognition," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6201–6210.

[10] S. Teeparthi, V. Jatla, M. S. Pattichis, S. Celedón-Pattichis, and C. LópezLeiva, "Fast hand detection in collaborative learning environments," in *Proc. 19th Int. Conf. CAIP*, Jan. 2021, pp. 445–454.

[11] V. Jatla, S. Teeparthi, M. S. Pattichis, S. Celedón-Pattichis, and C. LópezLeiva, "Long-term human video activity quantification of Student participation," in *Proc. 55th Asilomar Conf. Signals, Syst., Comput.*, Oct. 2021, pp. 1132–1135.

[12] W. Shi, P. Tran, S. Celedón-Pattichis, and M. S. Pattichis, "Long-term human participation assessment in collaborative learning environments using dynamic scene analysis," *IEEE Access*, vol. 12, pp. 53141–53157, 2024.

[13] P. Tran, M. S. Pattichis, S. Celedón-Pattichis, and C. LópezLeiva, "Facial recognition in collaborative learning videos," in *Proc. Int. Conf. Comput. Anal. Images Patterns*, Jan. 2021, pp. 252–261.

[14] T. Hastie, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed., Cham, Switzerland: Springer, 2009.

[15] C. Gu, C. Sun, D. A. Ross, C. Vondrick, C. Pantofaru, Y. Li, S. Vijayanarasimhan, G. Toderici, S. Ricco, R. Sukthankar, C. Schmid, and J. Malik, "AVA: A video dataset of spatio-temporally localized atomic visual actions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6047–6056.

[16] Y. Li, L. Chen, R. He, Z. Wang, G. Wu, and L. Wang, "MultiSports: A multi-person video dataset of spatio-temporally localized sports actions," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 13516–13525.

[17] W. Shi, M. S. Pattichis, S. Celedón-Pattichis, and C. LópezLeiva, "Robust head detection in collaborative learning environments using AM-FM representations," in *Proc. IEEE Southwest Symp. Image Anal. Interpretation (SSIAI)*, Apr. 2018, pp. 1–4.

[18] W. Shi, M. S. Pattichis, S. Celedón-Pattichis, and C. LópezLeiva, "Dynamic group interactions in collaborative learning videos," in *Proc. 52nd Asilomar Conf. Signals, Syst., Comput.*, Oct. 2018, pp. 1528–1531.

[19] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[20] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," in *Proc. Eur. Conf. Comput. Vis.*, Jan. 2016, pp. 21–37.

[21] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[22] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. (2019). *Detectron2*. [Online]. Available: https://github./facebookresearch/detectron2

[23] K. Chen et al., "MMDetection: Open MMLab detection toolbox and benchmark," 2019, *arXiv:1906.07155*.

[24] G. Bradski, "The OpenCV library," *Dr. Dobb's J. Softw. Tools*, vol. 25, pp. 120–125, Jan. 2000.

[25] E. Vahdani and Y. Tian, "Deep learning-based action detection in untrimmed videos: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 4, pp. 4302–4320, Apr. 2023.

[26] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[27] S. Teeparthi, "Long-term video object detection and tracking in collaborative learning environments," Master's thesis, Dept. Elect. Comput. Eng., Univ. New Mex., Albuquerque, NM, USA, 2021.

[28] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *Proc. ECCV*, Florence, Italy. Cham, Switzerland: Springer, Oct. 2012, pp. 702–715.

[29] Lambda Labs. *Deep Learning Gpu Benchmarks*. Accessed: Apr. 1, 2023. [Online]. Available: https://lambdalabs.com/gpu-benchmarks

VENKATESH JATLA received the Ph.D. degree in electrical and computer engineering from The University of New Mexico. He has a profound background in video activity quantification, image processing, machine learning, and video compression standards. His diverse research experiences, including human activity recognition and video compression, are well-supported by his work in both academic and industry settings, notably at MediaTek and UNM. His contributions to the field are documented through numerous publications in esteemed journals and participation in NSF-funded projects, showcasing his technical prowess in neural networks, video analysis, and a range of programming languages.

SRAVANI TEEPARTHI received the Master of Science degree in computer engineering from The University of New Mexico, specializes in image and video processing, boasting a CGPA of 4.17/4.0. Her extensive experience encompasses roles in data science, machine learning, and data engineering across various organizations, including the Fralin Biomedical Research Institute and Cadent. She has developed innovative machine learning models for computational neuroscience and has contributed to the advancement of data pipelines and analytics. Her research, recognized for excellence in video object detection and tracking within collaborative learning environments, has been published in notable conferences and journals.

UGESH EGALA received the M.S. degree in electrical and computer engineering from The University of New Mexico, Albuquerque, NM, USA, specializing in screen activity quantification in collaborative learning environments. His academic journey is marked by a strong focus on image processing, computer vision, and machine learning. His research contributions, aimed at enhancing learning experiences and analyzing nonverbal communication patterns, have led to publications in recognized journals. His professional experience spans both academic research and software engineering roles, showcasing a commitment to advancing educational technologies and collaborative learning analysis.

SYLVIA CELEDÓN-PATTICHIS was a Senior Associate Dean for Research and Community Engagement and the Director of the Center for Collaborative Research and Community Engagement, College of Education, The University of New Mexico. She is currently a Professor of bilingual/bicultural education with the Department of Curriculum and Instruction, The University of Texas at Austin. She prepares elementary pre-service teachers in the bilingual/ESL cohort to teach mathematics and teaches graduate level courses in bilingual education. She taught mathematics at Rio Grande City High School in Rio Grande City, TX, USA, for four years. She was a Co-Principal Investigator (PI) of the National Science Foundation (NSF)-funded Center for the Mathematics Education of Latinos/as (CEMELA). Her current work is a special issue on Teaching and Learning Mathematics and Computing in Multilingual Contexts through *Teachers College Record*. She co-edited three books published by the National Council of Teachers of Mathematics titled *Access and Equity: Promoting High Quality Mathematics in Grades PreK-2 and Grades 3-5* and *Beyond Good Teaching: Advancing Mathematics Education for ELLs*. Her research interests focus on studying linguistic and cultural influences on the teaching and learning of mathematics, particularly with bilingual students.

Ms. Celedón-Pattichis serves as a national advisory board member for several NSF-funded projects and as an Editorial Board Member for the *Bilingual Research Journal*, *Journal of Latinos and Education*, and *Teachers College Record*.

MARIOS S. PATTICHIS (Senior Member, IEEE) received the B.Sc. degree (Hons.) in computer sciences, the B.A. degree (Hons.) in mathematics, the M.S. degree in electrical engineering, and the Ph.D. degree in computer engineering from The University of Texas at Austin, Austin, TX, USA, in 1991, 1991, 1993, and 1998, respectively.

He was a fellow of the Center for Collaborative Research and Community Engagement, UNM College of Education, from 2019 to 2020. He is currently a Professor and the Director of online programs with the Department of Electrical and Computer Engineering, The University of New Mexico (UNM). At UNM, he is also the Director of the Image and Video Processing and Communications Laboratory (ivPCL). His current research interests include digital image and video processing, video communications, dynamically reconfigurable hardware architectures, and biomedical and space image-processing applications.

Dr. Pattichis was elected as a fellow of the European Alliance of Medical and Biological Engineering and Science (EAMBES) for his contributions to biomedical image analysis. He was elected a Senior Member of the National Academy of Inventors. He was a recipient of the 2016 Lawton-Ellis and the 2004 Distinguished Teaching Awards from the Department of Electrical and Computer Engineering, UNM. For his development of the digital logic design laboratories with UNM, he was recognized by Xilinx Corporation, in 2003. He was also recognized with the UNM School of Engineering's Harrison Faculty Excellence Award, in 2006. He was the General Chair of the 2008 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI) and the General Co-Chair of SSIAI, in 2020 and 2024. He was also the General Chair of the 20th Conference on Computer Analysis of Images and Patterns, in 2023. He has served as a Senior Associate Editor for IEEE Transactions on Image Processing and IEEE Signal Processing Letters, an Associate Editor for IEEE Transactions on Image Processing and IEEE Transactions on Industrial Informatics, and a Guest Associate Editor for two additional special issues published in IEEE Transactions on Information Technology in Biomedicine, a Special Issue published by *Teachers College Record*, a Special Issue published by IEEE Journal of Biomedical and Health Informatics, and a Special Issue published in *Biomedical Signal Processing and Control*.