University of Arkansas – CSCE Department
Capstone II – Final Report – Spring 2022

Team 13: Automatic Action Recognition

Garrett Bartlow, Daniel Miao, Joshua Stadtmueller, Jonathan Zamudio, Braxton Parker

Sponsored by Dr. Khoa Luu

Abstract

Automatic action recognition is one of the primary tasks in video understanding. It has various practical applications in fields such as human behavior analysis, virtual reality, and action/gesture recognition. Advancement in Artificial Intelligence has resulted in the studying of learning techniques called deep learning. Informally, deep learning takes inspiration from the human brain and solves problems with neural nets. Now, in this deep learning era, there are many methods that have been proposed to address the problem of action/gesture recognition. The resultant objective for this project, using these proposed methods, is to create a stand-alone application such that the deep learning technology can be leveraged in the day to day lives of people. This is done by leveraging the *Temporal Shift Module* to recognize a user's gestures and then subsequently controlling certain functions of a computer.

1.0 Problem

Today, there exists many tasks and activities among professional and private enterprises that because of their nature, require the use of common interfacing tools to computers, such as a keyboard and mouse. However, because of advancements in the fields of Artificial Intelligence and Computer Vision, there now exists an alternative way to interface with computers that renders the previous method trivial: Automatic Action Recognition. The problem itself is that of luxury, not of necessity. That is, in general, when using new technologies for interfacing with computers, the new setup (based on the new technology) then renders the previous technologies (and its setup) as an inadequate setup. A similar process occurred with the development of the mouse; the magnitude of inadequacy of the previous state of interfacing (keyboard only) was not necessarily realized until the next setup was in practice. And so, the problem is that with the existence of a new technology that could allow a new and improved interface to a computer, *there does not exist a stand-alone application that allows the use of the new technology to do the interfacing.*

With a better understanding of the problem, the importance of the problem can be seen by the potential benefit from solving it. For example, what if every non-text inputting or selecting action could be executed using gestures? What if every computer system could be calibrated to the actions and gestures of its user such that it allows for the most intuitive use of the computer based on the individual user? Such propositions are vision and conjecture based but are required to fully understand the importance of the fact that there does not exist a stand-alone application that allows for these types of scenarios.

Like most innovative technological advances, the impact of said technologies cannot be understood until after the fact. For example, the printing press revolutionized the media industry around the 15th century. The impact might have been known to the inventor of the press, Johannes Gutenberg. But, the impact across the world and time was only known after the fact. And so, the impact of not having a solution can be found in the potential realized impact of having a solution to the problem. And like many other advances in technology, time will tell on the impact of this problem and its solution.

2.0 Objective

The objective of this project is to provide a stand-alone application that leverages a modern deep learning model to provide action/gesture recognition and computer interfacing options to a user such that the user can integrate the application into any work done on a computer. The stand-alone application consists of a simple GUI designed to display the output of the deep learning model (the recognized gesture) as well as the aforementioned interfacing options designed to help control a computer. Aside from the main objective, other objectives include the following: learn the basic principles of general deep learning as well as its applications, work as a team using the agile development cycle, apply software engineering best practices, and gain experience in writing detailed documentation and analysis.

3.0 Background

3.1 Key Concepts

The foundational topic behind this project is deep learning. Deep learning is a solution to problems that "allow(s) computers to learn from experience and understand the world in terms of a hierarchy of concepts" whereas "each concept (is) defined through its relation to simpler concepts." [7] Counterintuitively, deep learning is more adept at solving abstract problems that humans aren't as good at solving (chess lines that are at least 4 moves long), whereas the inverse is also true; humans are better at solving less-abstract problems (ex. Object detection and recognition). Deep learning is quite adept at solving problems that contain large amounts of data, extrapolating concepts that live in that abstract domain which again might not be intuitive to humans [8]. The only downside is that because deep learning requires large amounts of data to be accurate, the domain of problems it is effective at solving is limited to those that have large amounts of data. At its core, deep learning allows for learning across layers by using the processes of gradient descent and back propagation.

From the principles of deep learning lies a technique called the Convolutional Neural Network (CNN). CNNs are mainly used "in the field of pattern recognition within images" [9][4]. CNN solves what the more general networks called Artificial Neural Networks (ANN) are inefficient at. For example, ANNs can generally solve image-based recognition tasks for small images such as 28x28 images [9]. When operating on larger images, CNNs are more efficient at this task. So, for operating on images from a simple webcam where the image sizes range anywhere from 1920x1080 to 540x303, CNNs are the tool for the task. But at their core, CNNs are "compromised of neurons that self-optimize through learning" [9]. The mentioned self-optimization through learning is done by the deep learning processes of gradient descent and back propagation across layers

Temporal Shift Module (TSM) utilizes the temporal dimension of images to manipulate data to achieve 3D CNN (Convolutional Neural Network) results with 2D CNN complexity. TSM is ultimately a technique using these CNN's where some data is taken in the form of video, then filtered into a collection of images. From these images, the already trained deep learning model predicts the current gestures shown in each frame of the video, if such a gesture exists from the pre-trained model [1]. There is an extension of the TSM module in which gestures can be classified and identified in real time at the expense of only a cache holding 1/8 of current features used in the model [1].

The application of the online version of TSM uses uni-directional TSM in which there is only a shift in the feature data from previous frames to current frames (frames being the current image). This is differentiated from bi-directional TSM which uses futures frames to influence the current frames on the feature data. As seen in figure 2, there is a slight information difference between uni-directional and bi-directional models as the number of spatial dimensions grows.

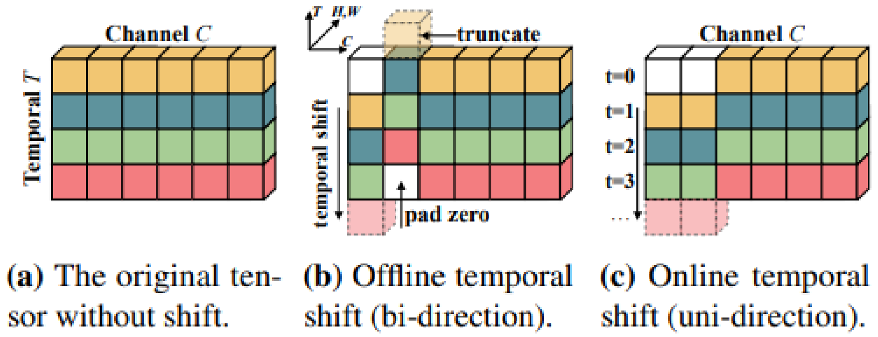
Channel C
Temporal T
T H,W C
truncate
temporal shift
pad zero
Channel C
t=0
t=1
t=2
t=3
...
(a) The original tensor without shift.
(b) Offline temporal shift (bi-direction).
(c) Online temporal shift (uni-direction).

Figure 1: From [1]: Temporal Shift Module (TSM) performs efficient temporal modeling by moving the feature map along the temporal dimension. It is computationally free on top of a 2D convolution but achieves strong temporal modeling ability. TSM efficiently supports both offline and online video recognition. Bi-directional TSM mingles both past and future frames with the current frame, which is suitable for high-throughput offline video recognition. Uni-directional TSM mingles only the past frame with the current frame, which is suitable for low-latency online video recognition.

Within the implementation of TSM, Python3, OpenCV, PyTorch and Scikit-Learn are used. OpenCV is an open-source library that deals with real-time computer vision. It can be used to process images and videos to identify different actors within frames such as objects, people, handwriting, and even hand gestures. OpenCV supports a wide variety of languages which includes Python3.

PyTorch and Scikit-Learn are both machine learning libraries for Python that enable the manipulation, construction, and analysis of data. For the purview of this project, the data gathered from OpenCV is then translated into different array types such that it can then be used with PyTorch. The difference, however, between PyTorch and Scikit-Learn is that PyTorch is much more suitable for deep learning and is used extensively here for the implementation of the TSM model. However, both frameworks are used at some point within the project.

Specifically in this project, the TSM modifies the MobileNetV3 computational model and uses MobileNetV3 as the backbone for training on different datasets. The modifications involved are those described previously in the introduction to the TSM. MobileNetV3 was developed to enable learning models to be used within embedded systems – I.e MobileNetV3 can operate with marginal power resources compared to other models while also achieving better results than, at its time, other modern models. For example, there is an increase in 4.6% accuracy for image classification while also reducing latency by 5% [10].

3.2 Related Work

Related to this project's work, a team at MIT has conducted research by using TSM for video recognition. The MIT team's paper [1] has a focus on the utilization of TSM to increase efficiency, their team implemented it with Google Maps, whereas this project's focus is using gesture recognition to support a stand-alone application that is designed for an increased interfacing experience.

There have also been strides to translate [2] sign language using gesture recognition in video processing. Elmahgiubi's team found that a fitted glove with sensors combined with a CNN allowed for the recognition of most of the letters in the ASL alphabet. The ASL vocabulary is incredibly dependent on the subtle movement and location of the fingers. So, high accuracy and precision is necessary for a successful model. The fitted glove and proposed model yielded a 96% average accuracy as well as recognizing 20 out of 26 letters [2]. Another team proposed a similar system in which infrared images are used to feed to the CNN [5]. While the implementation of each CNN may be different, the input parameters are surely different as one team used sensor data, and another used infrared imaging. Tao's team [5] achieved a 99.7% average accuracy when testing all 24 alphabet gestures with 5 different patients. So, while it is important to note the technology of CNNs, the input selection plays a role.

Not only are known gestures configurable in video recognition, but also a [3] team has discovered how to predict unknown gestures using similar techniques as well as a cognitive behavioral model. More related to TSM, Benitez-Garcia's team proposed a system inspired by TSM in which specifically, TSM is inserted to a temporal shift network (TSN) where the TSN operates only on a sequence of clips within a video instead of the entire video. The final video-level prediction is then considered a summation of accumulation of each shorter clip sequence [6].

These other works have significance to the proposed application because they allow for inspiration and proof-of-concept ideas for new projects. More specifically, the development of the sensor fitted glove shows that wearable technology can be integrated with the ideas put forth in the TSM paper [1]. The advancement of one facet of technology can also lead to the development of innovative technologies as shown by Benitez-Garcia [6].

4.0 Design

4.1 Requirements and/or Use Cases and/or Design Goals

Requirements (this project must --):

- Utilize the online TSM to perform automatic action/gesture recognition
- Consist of a GUI that displays the following:
    - Output of the TSM execution
    - Customization settings for computer controls
    - Quality of Life settings (colors/size of different components in the GUI)
- Be conducted using the agile development cycle as shown by the task schedule

- Be created using software engineering's best practices such as ample documentation and abstraction

Use Cases (this project can be used to --):

- Operate PowerPoint using any of the supported gestures
    - This use case extends to technically any activity that uses any of the supported computer commands, PowerPoint is the specific activity demonstrated by this project

4.2 Detailed Architecture

Overarching, this project consists of two distinct parts – the engine and the user interface. The engine consists of all the computation done by the modified MobileNetV3 model when executing. None of this is explicitly seen by the user. The user interface consists of everything that is seen by the user. This includes the application window itself, the webcam output, the engine output, as well as multiple components that detail the different options made available to the user. The application's architecture follows the methods of the Model-View-Controller (MVC) conventions and integrates the defined engine part with the user interface portion with those conventions. The user interface is designed using component abstraction and is designed so that the application can be expanded upon with ease at a developer's behest.

The technologies used in the creation of the application (not including the technologies used in the TSM exclusively) include Python3, Tkinter, and OpenCV where OpenCV and Tkinter are modules for Python3. OpenCV is used for all image capturing and processing for use in the computation of gesture recognition. It is also used to capture a video stream such that each frame can be displayed inside the user interface. Tkinter is a GUI library. Tkinter is used to create each interface component as well as display each frame from OpenCV. Tkinter is responsible for the "drawing" aspect of the MVC convention. At its core, the application interacts with primarily Tkinter because of its ease of access and low-level support (low level by Python's standards).

Firstly, the engine part of the application is composed of two distinct parts, data gathering and then data processing. Data gathering consists of retrieving a video stream from a webcam. This is done with OpenCV. From there, the data is transformed into the correctly shifted form to be used with the TSM. After that, for each frame in the video stream, the executor is called. The executor simply takes the trained model and then runs the model on the newly transformed data. The model used for the project is the default PyTorch checkpoint that was initially given for the TSM. However, a model that has been trained subsequently. on said checkpoint can also be used here. From the executor, the output is found after a processing function (which is fed the output of the executor) returns the current frame's recognized gesture and the last n-gestures in the form of a list (the history is the list of past gestures where n is the number of frames in the past inclusive to the current frame). For this project, the history length is 20 elements which is the default configuration. When experimenting with other lengths, no difference was found in the efficacy of the heuristic that was applied, as will be detailed later.

Secondly, the user interface in its design is composed of several distinct components. Such components include the webcam display, the engine output of the recognized gesture, a button to toggle between a grayscale or RGB view of the webcam, a button to enable/disable computer control based on the recognized gesture, a section where a user can "record" a sequence of gestures to calibrate the heuristic to themselves for a particular computer control, and a section where the user can define explicitly which gesture controls which computer function. An illustration of the completed interface is below:

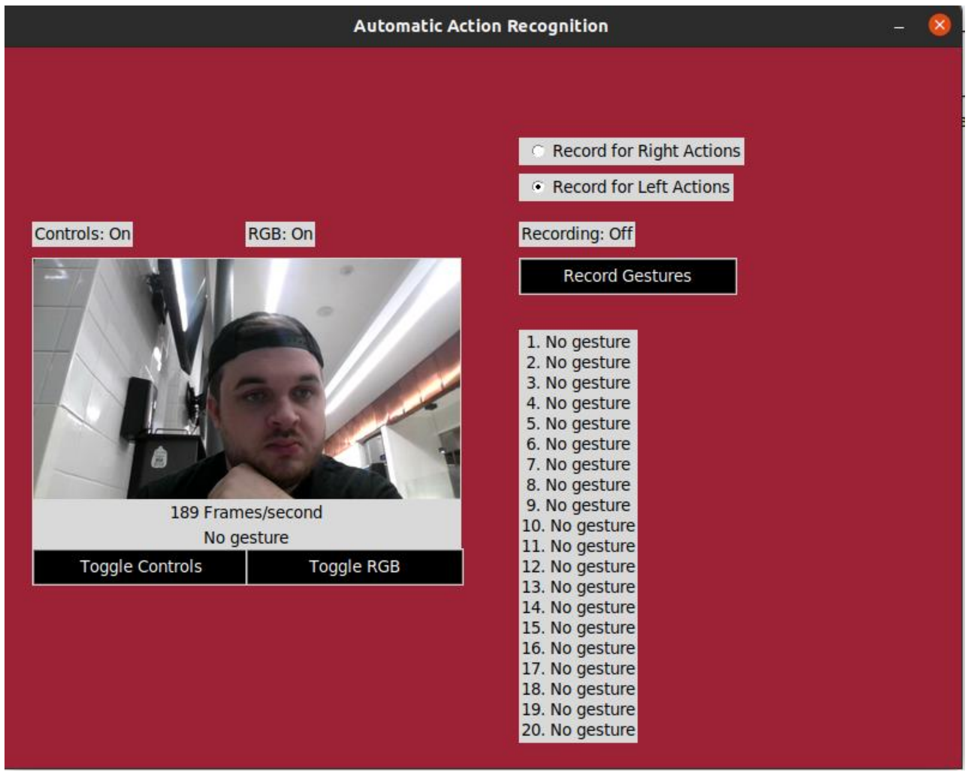

Figure 2. The user interface displays all the implemented components which include Toggle Controls, Toggle RGB, Record Gestures, settings for recording gestures, and the history list of 20 gestures.

Now, within the actual implementation of the above sections, the overall application follows the convention of the Model-View-Controller scheme. Since there exists only one view in the application, the implementation only explicitly uses the model and controller. The system of the application works in the following way. The webcam video stream is a member of the engine. On each iteration of the application's runtime, the engine operates on that iteration's frame that is captured from the video stream. Then, the component inside the model which contains the place for the webcam is fed to the webcam frame used in the engine computation for that iteration of the application. The model contains each component of the user interface and receives inputs from the controller and engine. The controller consists of the user's input into the application itself, not the resulting computer control from the recognized gesture of that current program iteration. So, the controller feeds input actions that change or alter the state of some part of the model. Because of the use of the library Tkinter in the application, not all controller aspects of the application are explicitly in the controller class. For instance, each button component from Tkinter receives an "on click" parameter that determines what to do when that button is clicked. This is logic that would normally be implemented in the controller class. However, since it is already pre-defined, each button's on click behavior is not influenced by the controller class but by the Tkinter module definitions. Though every controlling logic is not in the controller class, the application still follows the Model-View-Controller convention. Different than the controller class, the view class is not utilized in this project explicitly. This is due to the application only containing one view. Since there are not multiple views to be differentiated from, adding an explicit view class to handle the one view used in the application only adds unneeded complexity. In its implementation, the application's components looks and styles are handled automatically as enabled by Tkinter. In this case, handled simply means state updates for text and color/size. Because this functionality exists innately to the library, it allows for the avoiding of the mentioned unnecessary complexity. So all-the-while there is not an explicit view class used within the application, there is a view used. So, the architecture and implementation still follow the convention for the MVC method.

Out of the box, the engine utilizes a PyTorch checkpoint to execute a model using the input video stream from the webcam that is used. However, it is possible to continue training using that checkpoint and then plug it into the engine. Also, it is also possible to completely retrain the model using a different dataset. This is significant because it allows for different gestures to be recognized. But, the original checkpoint is suitable for the scope of this project, so that is what is used in the final version of the application. However, other models were used such that they worked within reason in the application.

Due to the application-specific nature of this project (controlling PowerPoint), an inefficiency became known when testing. Specifically, when a user would try and "swipe right" to get to the next slide, often the model would recognize several different gestures. Some of these include swiping right, two fingers swiping right, as well as swiping left when retracting the arm used to swipe after the swiping was finished. To solve this problem, a heuristic was designed that allows the application to be more efficient for the problem of controlling PowerPoint. The heuristic essentially "calibrates" the application to the user. This is done by recording the gestures executed while performing the desired action. For example, the user would emulate swiping right and the list of recognized gestures during that span are kept as a list. Also, the recognized gesture from the previous program iteration/frame is stored. Per program iteration, the heuristic logic is as follows: If the current recognized gesture is not the previous gesture, (for swiping right) if the current gesture is in the recorded/calibrated "swiping right" list and the previous is not, then swipe right. Else, do nothing. This ensures that only one execution of the computer control happens as well as ensuring that swiping right or left only occurs when the user truly intends to swipe right or left.

Overall, the lesson learned came from a better understanding of the technologies that were used during the creation of the application. For example, in Tkinter, since there is absolute positioning, it is quite easy to explicitly state the position of elements given some current interface state. However, if that interface state were to change, the elements that used absolute positioning would no longer be in the correct state as desired. However, as shown in the implementation details previously mentioned, listing components or elements as relational to another component leads to easier customization of the user interface state as well as easier readability. This is because if there is a relation between components, there can be a hierarchy derived from the relation as opposed to just a large list of relation-less, distinct components. And from this hierarchy, position and other visual styling can easily be changed with the modification of a couple of components, instead of every component. So, creating the application with relating components was a lesson learned. Another lesson learned was that because the application only contains a singular view, explicitly using a View class only adds additional complexity. This is important because with the current vision of the application, no other view is needed. That is, the current display state of the user interface is the only state necessary. So, for this project, learning when to not add additional complexity was a lesson learned. If the application were to be expanded, another view would be introduced. But that is not within the purview of this project, albeit a good point to mention. Lastly, as mentioned about the ability to use different models for execution within the engine, a point to be made is the difficulty of obtaining datasets. Because deep learning is data intensive, it is not always possible to have data that goes to more obscure gestures. So, the gestures that a model can support depends on the availability of gestures within datasets. So, if a very particular gesture was needed for a specific scenario, it may be difficult to obtain enough data necessary to train a model efficiently. But, this project relies on common gestures often found in datasets such as the Something-Something dataset and the Jesture dataset.

Based on the achievement of creating an application that leverages action recognition, the potential impact is the integration of this application within the workflow or daily lives of individuals or companies. Because this project attempts to offer a better way to interface with computers, it is reasonable to conjecture that there exists some activity that would benefit from the functionality that the application from this project provides. And so, at the very least, the impact is the ability to conduct PowerPoint presentations with simple gestures, no hand clicker needed and no need for close proximity to a computer (relatively speaking). This project also serves as a launching point for which future work can be created, with this as a reference.

Next, potential improvements for future works could include better recognition models designed for specific task use when tasks are identified. For example, instead of utilizing an action recognition model that is trained for many actions, perhaps a model specifically trained on one action could be used such that the output of the model is simply, yes or no. This could be helpful if applied to vehicles because the model could discern if the driver is engaged or not. Therefore, it could be decided of the driver was engaged or not. This could be used similarly to a dashcam as a defense in a court system, perhaps. But more relevant to the context of this project, an increased number of actions/gestures could be recognized, an increased number of possible computer controls could be supported, a better user interface could be supplied, and a widening of compatibility could be enabled. This is because the current application only works on Linux based systems.

4.3 Risks

Risk	**Risk Reduction**
Injury from doing an action/gesture	Issue a warning on the application start to be mindful of the surroundings
Not having 100% reliability the gesture recognition and application	Apply a heuristic to better approximate for practical use. Also, a message will be shown explaining the application does not have 100% accuracy
Having several dependencies might lead to conflicts on different machines	A comprehensible list of dependencies and their versions will be made available as well as a packaged image of the application such that it is easy to run out of the box on another machine, eliminating the need to install manually, each dependency

4.4 Tasks –

1. Understand/gain background about using the hand gestures inputs and controlling outside applications/software.
2. Define exactly what actions/controls will be implemented

3. Determine design of GUI
4. Determine architecture for the complete application
5. Implement basic version of GUI
6. Implement a single gesture control with engine integration
7. Debug basic version of application
8. Implement a more advanced version of GUI
9. Implement the remaining hand gesture controls
10. Debug advanced version of application
11. Put final changes on application and allow time for unforeseen errors
12. Complete documentation
13. Work on demonstration of our application

4.5 Schedule –

Tasks	Dates
1. Understand and gain background information about using gesture recognition to control computer functions	1. 11/14/2021 - 11/28/2021
2. Define what computer controls are to be implemented	2. 11/29/2021 - 12/17/2021
3. Determine design of the GUI	3. 1/17/2022 - 1/24/2022
4. Determine architecture and design of the GUI implementation and integration with the engine	4. 1/24 - 1/31
5. Implement the basic version of the GUI	5. 1/31 - 2/28
6. Implement a singular computer control based off gesture recognition	6. 1/31 - 2/14
7. Implement the rest of the defined computer controls	7. 2/14 - 3/21
8. Debug basic version of the application	8. 2/14 - 3/7
9. Implement additional GUI components	9. 3/7 - 4/11
10. Retrain the TSM on a new dataset	10. 3/7 - 4/11
11. Debug the advanced version of the application	11. 4/4 - 4/15
12. Refine the aesthetical aspect of the application	12. 4/14 - 4/15
13. Finish the documentation	13. 4/15 - 4/22
14. Complete the final presentation and practice the demonstration	14. 4/22 - 4/29

4.6 Deliverables

- Design Document: Contains a listing of each major hardware and software component
  - Hardware includes the following: webcam (with model type), and computer used to run the developed application

    - Software includes the following: Linux distribution used and the developed application as well as all of the dependencies of the TSM
- Python codebase for GUI application and TSM implementation
- Presentation slides which are used for the final presentation
- Final Report which is this current document
- GitHub Link which contains via cloud, all of the source code used in the application
- Team Website which contains each member's personal website links, introduction to the project, as well as links to the project schedule, final report and presentation, and task list for this project

5.0 Key Personnel

Daniel Miao – Miao is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Software Engineering courses. He has experience with AI prediction for medical patients as an intern at Arkansas State University. He will be responsible for assisting the bridging of the TSM application to computer commands.

Josh Stadtmueller – Stadtmueller is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Software Engineering and Artificial Intelligence courses. He has experience with developing front-end applications as an intern with Cobb-Vantress. He will assist in front-end development.

Garrett Bartlow – Bartlow is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Artificial Intelligence and Software engineering courses. He gained experience with developing back-end applications, cloud computing, and IoT devices while interning at Dover Fueling Solutions. He will be responsible primarily for implementing actions from hand gestures, however he will be taking a full stack approach. So, he will be involved in all aspects of the project. He will act as team leader.

Jonathan Zamudio – Zamudio is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Artificial Intelligence and Software Engineering courses. He has experience with machine learning through the NACME Google Applied Machine Learning Intensive Summer 2021 Bootcamp. He will assist with bridging the TSM application to the computer commands.

Braxton Parker – Parker is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Artificial Intelligence and Software engineering courses. He has experience in developing intelligent systems in the Unreal framework. He is responsible for bridging the TSM application to the usage of computer commands.

Champion– **Dr. Khoa Luu** received his Ph.D. degree in Computer Science at Concordia University, Montreal City, Canada. His Ph.D. thesis was nominated for the Governor General Gold Medal in Canada. He was the valedictorian for the joint Faculty of Engineering & Computer Science and Faculty of Fine Arts convocation ceremony at Concordia University in 2014. His research interests focus on several topics, including Biometrics, Image Processing, Computer Vision, Machine Learning, Multifactor Analysis, Correlation Filters and Compressed Sensing.

6.0 Facilities and Equipment

The necessary equipment needed for this project is a Linux operating system, a webcam, and a machine that is CUDA-compatible in-order-to run the TVM module used by the application.

7.0 References

[1] Lin, Gan, Han, “TSM: Temporal Shift Module for Efficient Video Understanding,” Arxiv, MIT, 2019

[2] M. Elmahgiubi, M. Ennajar, N. Drawil and M. S. Elbuni, "Sign language translator and gesture recognition," 2015 Global Summit on Computer & Information Technology (GSCIT), 2015, pp. 1-6, doi: 10.1109/GSCIT.2015.7353332.

[3] T. Zhang, Z. Feng, Y. Su and F. Min, "Semantic Gesture Recognition Based on Cognitive Behavioral Model," 2014 International Conference on Information Science & Applications (ICISA), 2014, pp. 1-4, doi: 10.1109/ICISA.2014.6847463.

[4] M.V. Valueva, N.N. Nagornov, P.A. Lyakhov, G.V. Valuev, N.I. Chervyakov,

Application of the residue number system to reduce hardware costs of the convolutional neural network implementation,

Mathematics and Computers in Simulation,

Volume 177, 2020, Pages 232-243, ISSN 0378-4754.

[5] Tao, Wenjin & Lai, Ze-Hao & Leu, Ming & Yin, Zhaozheng. (2018). American Sign Language Alphabet Recognition Using Leap Motion Controller.

[6] Benitez-Garcia, Gibran et al. “Improving Real-Time Hand Gesture Recognition with Semantic Segmentation.” *Sensors (Basel, Switzerland)* vol. 21,2 356. 7 Jan. 2021, doi:10.3390/s21020356

[7] Ian Goodfellow, Yoshua Bengio, & Aaron Courville. *Deep Learning*. MIT Press. (2016).

[8] Najafabadi, M.M., Villanustre, F., Khoshgoftaar, T.M. *et al.* Deep learning applications and challenges in big data analytics. *Journal of Big Data* **2,** 1 (2015). https://doi.org/10.1186/s40537-014-0007-7

[9] O'Shea, Keiron et al. "An Introduction to Convolutional Neural Networks." (2015).

[10] Howard, Andrew et al. "Searching for MobileNetV3." (2019).