Technical Report Capstone Project

Vassily Lombard
Earlham College
Richmond, Indiana, USA
vtlomba22@earlham.edu

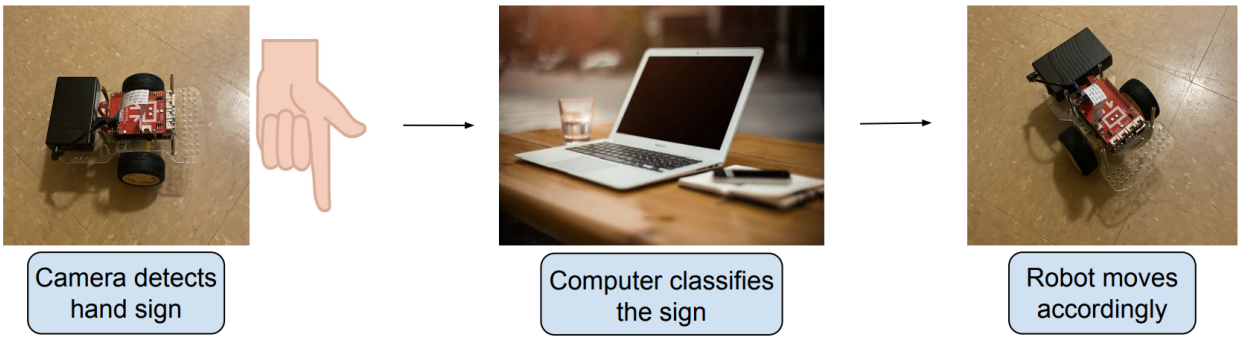

Figure 1: Graphical Abstract

ABSTRACT

Machine learning has played an important role in sign language recognition over the past decade. However, many existing systems are complex, require extensive computational power, and lack precision in real-time use. This project aims to develop a simple and time-efficient sign language-controlled robot that interprets American Sign Language (ASL) gestures to perform movement-based tasks. Built with a Raspberry Pi, the robot will communicate with a nearby laptop to detect and classify signs in real time. Building on prior research, this project will use lightweight computer vision techniques for static sign recognition. It will focus on improving static sign detection accuracy while minimizing latency, enhancing overall system responsiveness. The study highlights the potential of sign language for human-machine interaction, with broader applications in accessibility and inclusion.

Keywords: Neural Network, Deep Learning, Computer Vision, Raspberry Pi, Robotics, Sign Language

1 INTRODUCTION

Human-machine interaction (HMI) is a rapidly evolving field that bridges the gap between humans and technology, enabling intuitive communication and control of machines. As technology becomes more integrated into daily life, effective HMI systems are crucial for making complex systems accessible to a wide range of users. One promising approach to HMI is gesture recognition, which allows users to interact with machines through natural and intuitive movements. This method eliminates the need for traditional interfaces like keyboards or touchscreens, making it particularly valuable in scenarios where hands-free or remote operation is necessary.

Gesture-based HMI systems have transformative potential across numerous fields. In healthcare, they can enable surgeons to control medical devices during operations without physical contact, reducing the risk of contamination. Similarly, in industrial applications, workers can control machinery in hazardous environments without direct contact, increasing safety and efficiency.

However, despite their promise, gesture recognition systems face key challenges. Real-time recognition of complex, dynamic gestures remains difficult due to variations in hand shapes, motion speeds, and environmental lighting conditions. Additionally, accurately mapping these gestures to machine commands requires advanced computational models and efficient hardware integration. These challenges limit the reliability and practicality of gesture-based interfaces in real-world applications.

This project aims to address these challenges by developing a gesture-controlled robot capable of interpreting American Sign Language (ASL) gestures. By leveraging state-of-the-art computer vision techniques alongside Raspberry Pi-based robotics, the system will improve the accuracy and responsiveness of gesture-based HMI. This advancement will contribute to the broader adoption of HMI technology in accessibility, robotics, and beyond. We seek to create a more inclusive, efficient, and human-centered approach to human-machine interaction.

2 RELATED WORK

Controlling a robot without explicitly using a remote controller can be achieved in multiple ways. It can be controlled via voice commands, hand gesture control, or even accelerometer-based control. These technologies can be efficient and life-changing if implemented correctly. A lot of research has been conducted in voice and

sentiment analysis detection, and although significant research has also been done in hand gesture recognition, there is still much to be explored in the field of sign language detection. Effective gesture recognition is foundational for a sign language-controlled robot. Studies have explored ways to recognize hand gestures, especially in the context of sign language recognition. One main challenge many researchers have identified is the complexity and diversity of signs in American Sign Language (ASL). Many signs are static, while others are dynamic. Additionally, many dynamic signs look very similar at the beginning, making them difficult to distinguish.

2.1 Common Approaches for Static Signs Recognition

Many studies have investigated static sign recognition and achieved high accuracy. Barbhuiya et al. (2021) focused on static gesture recognition using pre-trained AlexNet and VGG16 models. Features extracted from the ImageNet dataset were then classified using a support vector machine (SVM), yielding high accuracy in static sign recognition. While Barbhuiya et al. (2021) only used RGB channels in the images to train their neural network, Kuznetsova et al. (2013) highlighted the role of depth cameras in capturing detailed hand gesture features from static hand signs. Their approach utilizes a multi-layered random forest model, taking advantage of depth information in the image to classify each sign. Their model input is a 3D point cloud, which they first cluster to group similar classes together. The first random forest (RF) layer classifies each cluster of similar signs, and the second layer classifies each sign within each cluster. This alternative method to CNNs also achieves good accuracy with low training time. However, similar to Barbhuiya et al. (2021), their experiment is only valid for static images.

2.2 Two Streams Networks

Other methods combining various features of a single image have been investigated and have yielded even better results for static signs. Specifically, Dadashzadeh et al. (2020) proposed a two-stage fusion network, taking advantage of the appearance of an image (RGB color channel) and the segmentation map of the sign. The image is first transformed into a segmentation map, delimiting the hand performing the sign, and then two CNNs are fed with the segmented gesture map and its RGB appearance, respectively. Their outputs are combined at the decision level using summation techniques to produce the final classification. Similarly, Xu et al. (2014) took advantage of depth information to delimit the hand and then used the RGB color channel for skin color detection. Combining depth and RGB features yielded more accurate results than using just one channel.

The previous studies highlight the role of depth cameras as well as regular RGB cameras in recognizing sign languages. Combining these streams of information yields even better results, according to Dadashzadeh et al. (2020) and Xu et al. (2014). This approach allows them to overcome issues such as similar skin and background colors as well as lighting problems. This innovative approach of cleverly combining various streams of information deserves more exploration, as there are various ways to fuse them—at the decision level, feature level, or data level. Additionally, each of the individual streams can have many different architectures, leaving room to explore creative designs and combinations to achieve better results in static sign classification.

2.3 Dynamic Signs Recognition

While a great amount of work has been done in static sign detection, there is still room for improvement, especially in dynamic sign detection. In the quest to recognize larger amounts of signs in ASL, including dynamic signs, researchers have confronted issues such as detecting when a dynamic sign starts and ends. It is crucial to have a rough idea of the window in which the sign occurs. If it is inaccurate, one sign may look too different and be classified as another sign. In addition, it is essential to make sense of the relationship between each frame in a dynamic sign. There is a logical order between each frame that must be considered to achieve high accuracy.

2.4 3D CNNs

Considering the relationship between each frame in a dynamic sign, researchers developed a new type of CNN to add an extra dimension of time. They are called 3D CNNs, in contrast to conventional 2D CNNs, based on the number of input dimensions they can handle. Li et al. (2020) and Kopuklu et al. (2019) leveraged 3D CNNs to address the problem of the time dimension in dynamic sign detection. On testing data, they achieved high accuracy, seemingly addressing the problem of detecting dynamic signs. However, in a real environment, it is crucial to know when a sign is being performed to efficiently pass the specific frames to the 3D CNN for sign classification.

Addressing the issue of knowing the starting and ending points of a sign, Xu et al. (2014) used a specific static gesture to mark these points. This approach allowed them to use previous work on static gesture recognition to show when a dynamic sign starts and ends. While promising, this approach does not explicitly solve the problem of detecting the ending and starting points of a dynamic sign. To address this, Kopuklu et al. (2019) trained a 3D CNN to infer when a sign is being performed. They used a sliding window approach, with 8 frames fed to their 3D CNN to detect if a movement is being performed. Specifically, to avoid multiple activations, they implemented a weighted classification score to activate detection only above a specific threshold. This way, Kopuklu et al. (2019) could detect when a dynamic sign is being performed and use a set number of frames to classify the exact sign being made.

2.5 Limits of 3D CNNs

While innovative and promising, 3D CNNs require extensive amounts of data to train. Since they are used to recognize video patterns, which involve more data than regular images, they need to learn more complex models. Although this problem is partially addressed by Li et al. (2020) by creating a new large dataset for dynamic hand gesture recognition, the training time of 3D CNNs is significantly impacted due to the large amounts of data.

Finally, further work can be done in refining techniques to detect specifically when a gesture starts and ends. It is a complex task because a sign can be confused with a simple unintended movement. However, such advancements in this area are crucial for improving dynamic gesture detection. Being able to accurately detect when a sign starts and ends will enable the sign classifier model to process

the correct frames, containing only the specific sign, and hence perform better classification.

2.6 Application to Robotics

Overall, in the context of creating a robot controlled by sign language, there appears to be little work done in various ways. While Xu et al. (2014) used explicit hand gestures to control their robot, most of the studies do not apply their work to small robots specifically. Most of the work focused solely on recognizing hand gestures or sign language without applying it to robots, which is my goal. After reviewing the current state of the field, there appear to be two main challenges and areas for improvement. The first is the possibility of combining RGB and depth image information at different stages and in innovative ways to achieve higher accuracy in static sign classification. Secondly, there is room to improve efficiency in detecting the starting and ending points of dynamic gestures. These two aspects are interconnected and are both crucial to advancing the field.

3 METHODOLOGY

The proposed system comprises two primary components: the sign recognition system and the robot.

3.1 Dataset

In this project, I used three different datasets to train my model as it was important to increase the variety of the images to achieve higher accuracies. The first one was found on the internet, the second one was created by me and the third was a mix of both.

The first dataset, found on the internet, is the American Sign Language (ASL) Alphabet dataset from Kaggle. It contains 87,000 images of hand gestures representing the 26 letters of the English alphabet. As Figure 2 shows, the images were captured under various lighting conditions and orientations, mostly with white backgrounds. For the sake of this project, I am only interested in four signs in this dataset, which indicate the four possible directions the robot can move. Specifically, the letter L represents the left direction, R the right direction, N the north direction (forward), and S the south direction (backward). These signs were chosen because they have a dual meaning: the letter itself and a direction. Hence, out of all 87,000 images in my dataset, I was only interested in about 8,000 images, which constitute my base dataset for my model. Data augmentation, such as rotation, flipping, and brightness adjustments, was applied to increase the variety within this dataset, allowing the model to generalize well and avoid overfitting.

As Figure 3 shows, the second dataset created by me, differs from the first one due to the variety of background colors, patterns, and lighting conditions. It was crucial to have a dataset with more variety than just a white background so my model could achieve high accuracy regardless of the environment. My dataset consisted of a total of 160 images, with 40 images in each category. Similarly to the first dataset, each category represents a direction the robot can follow. In order to make my dataset more suitable for training, I increased its size through data augmentation, such as rotation, flipping, and brightness adjustments. The final size of my dataset with augmented data was 640, with 160 images per category.

Figure 2: Sample images of kaggle dataset

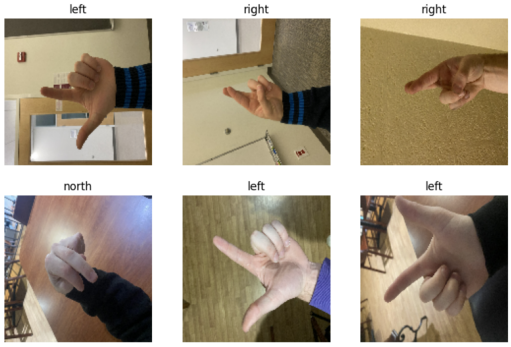

Figure 3: Sample images of my dataset

Lastly, the third dataset was a mix of both the dataset from kaggle and the one created by me. Each of the four classes had the same number of images, 80, for a total of 320 for the entire mixed dataset. Within each class there are 40 images from my dataset and 40 images from kaggle dataset. Balancing the number of images for each class was crucial to avoid bias and overfitting, as well as making sure the overall dataset is a good mix of both dataset. The ideas behind making a mixed dataset is to compare the performance of each model based on the dataset.

Finally, before training, the images of both datasets were resized to 150x150 pixels, and the pixel values were normalized between -1 and 1. This ensured that the model received the same data shape for each image and also contributed to increasing the accuracy of the model.

3.2 Pre-trained Models

Importing Pre-trained Models appeared as the best solution to achieve good performance as their architectures were specifically made for edge detection. They also already perform well on classification task containing hundreds of categories, they are good to detect patterns in image recognition. In addition it requires extensive datasets and computational resources to train such models to achieve good performances. For these reasons I chose to import Pre trained models with their weights trained on the imagenet dataset, assuring a minimal good accuracy. Specifically, vGG16 and DenseNet201 pre trained were chosen for my tasks of classification.

VGG16 was chosen for its strong features extraction capabilities and its high accuracy for classification tasks. On the other hand, DenseNet201 was chosen for its efficiency in complex images processing task as well as its computational efficiency making it perfect for real time application.

3.3 Hybrid Fusion of DenseNet201 and VGG16

Various approaches using these pre-trained models were tried to obtain the best accuracy under different conditions and environments. The selected method consists of fusing the outputs of two pre-trained models at an intermediate stage, taking advantage of their individual strengths.

3.3.1 DenseNet201 Architecture.

Figure 4 shows the general DenseNet architecture. It is a deep Convolutional Neural Network (CNN) that consists of four dense blocks, where each layer receives inputs from all previous layers. DenseNet201 specifically cotains 201 layers making it deeper and potentially more powerful for complex feature extraction.

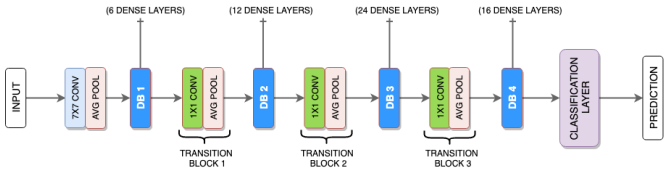

Figure 4: DenseNet Architecture: A Deep CNN with 201 Layers Divided in 4 blocks

3.3.2 VGG16 Architecture.

Figure 5 shows the architecture of VGG16. It is a deep learning model with 16 layers, structured into five convolutional blocks that extract image features step by step. Unlike DenseNet201, which uses global average pooling, VGG16 flattens the feature maps before passing them through three fully connected layers, making it computationally heavier but effective for classification tasks.

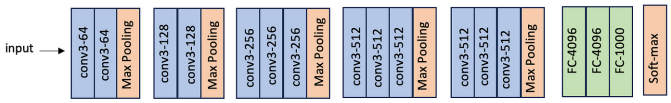

Figure 5: VGG-16 Architecture: A Deep CNN with 16 Layers, Including Convolutional, Max-Pooling and Fully Connected Layers

Both models were imported with the classification layers truncated and instead adding dense and dropout layers.

3.3.3 Fusion Architecture.

Leveraging the strengths of both architectures, I implemented a hybrid fusion model that combines highly abstracted features extracted by DenseNet201 and VGG16. First, each model processes the input image separately, extracting high-level features through their respective layers. Both models generate a compact feature representation of the input, reducing dimensionality while preserving essential information.

Next, the extracted features are concatenated to form a unified feature vector. To refine this fusion, an attention mechanism is applied, learning the importance of each feature and enhancing the most relevant ones. Finally, the fused feature representation passes through a fully connected layer for classification, ensuring that the combined model benefits from DenseNet201's efficient feature reuse and VGG16's deep hierarchical representations.

This hybrid fusion technique differs from decision and feature-level fusion in its approach to combining information from multiple models. Feature-level fusion directly merges features extracted by each model before classification. On the other hand, decision-level fusion combines the final predictions made by each model. In my model, hybrid fusion occurs at an intermediate stage, after they have passed through certain layers but before final decision-making. An attention mechanism is then used to refine and weight the features, allowing the model to take advantage of the complementary strengths of each model. This technique proved to work and achieve high accuracies on both datasets.

3.3.4 Training Specifications.

The model was trained using the RMSprop optimizer, which is a derived version of Adam optimizer, with categorical cross-entropy loss, employing a batch size of 32 and an initial learning rate of 0.001. Early stopping and learning rate reduction were applied to prevent overfitting and optimize convergence.

3.4 Data Framework

Perfectly integrating my sign language classification model in the overall robot computer system was crucial to minimize latency between the two system.

The robot is made of a raspberry and thus is not able to support the classification model due to its limited computational capabilities. To avoid this problem, the model run on a nearby laptop with sufficient computational capacities. A private wifi connection is broadcasted by the robot and the laptop connect to it sending back and forth data between the two devices. The TCP/IP protocol is used with the robot running a server and waiting connection from the laptop (client). The robot is set to listen on a specific port and the laptop send request via a client socket. Once the communication is establish the laptop and robot can exchange data with low latency within a range of a few meters.

Equipped with a camera, the robot sends frames to the connected laptop. Once received, each frame received are processed by the laptop to classify the sign made in front of the camera. Once predicted the output is sent back to the robot and moves accordingly. An extra module in addition to the raspberry pi is added to control the individual motors of the robot, facilitating the control of the robot based on the predicted movement. Figure 6 shows the data flow between the different components of this system.

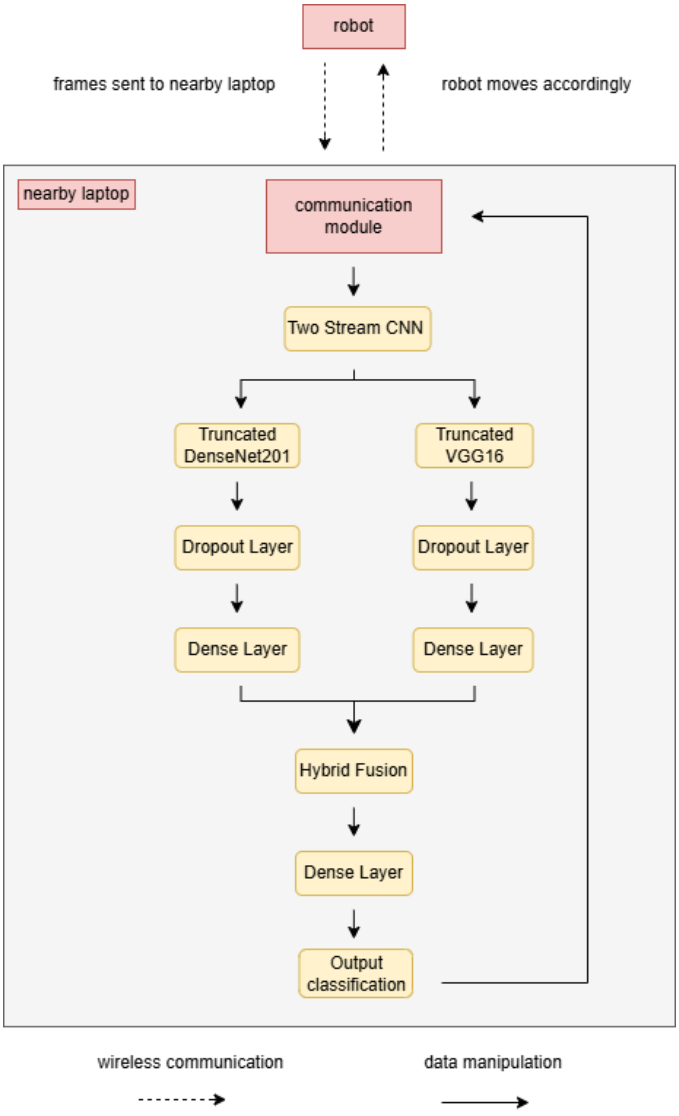

Figure 6: Data Flow Architecture

4 RESULTS

4.1 Training Results

With the three models proposed (VGG16, DenseNet201, and VGG16 x DenseNet201 Fused), I was able to train them on the three datasets I had. Table 1 shows how each model performs on their respective training and validation datasets. VGG16 is optimal on *My Images* dataset, achieving 96% accuracy. On the other hand, DenseNet201 is optimal on *Kaggle Dataset*, with 98% accuracy. Overall, this means that individual pre-trained models perform well on these datasets and are able to recognize general patterns within them. However, when these pre-trained models are tested on datasets other than the ones they were trained on, their accuracy drops significantly. Table 2 shows that VGG16, trained on *My Images* dataset, achieves only 47% accuracy on *Kaggle Dataset*. Similarly, DenseNet201, trained on *Kaggle Dataset*, achieves only 57% accuracy on *My Images* dataset. This clearly demonstrates how these two individual models struggle to generalize their results. They perform well when classifying images similar to those they were trained on but struggle when encountering different images.

The fusion model generalizes better the results learned from one dataset to another. It is due to its architecture taking advantage of the strength of each individual pre-trained, making it easier to learn general pattern and not focus on the background. Table 2 shows how the fusion model performs well even when the validation dataset is different from the dataset it was trained one. When it is trained on *My Images* dataset it achieves an accuracy of 78% on *Kaggle Dataset*, against 47% achieved by VGG16. Similarly, when the fusion model is trained on *Kaggle Dataset* and tested with *My Images* dataset it achieves an accuracy of 82% against 57% achieved by DenseNet201. Although not perfect, the accuracy achieved by the fusion model is better than the individual pre-trained model, showing that they can learn more complex patterns and generalize them.

Model	Training and Validation Dataset	Accuracy
VGG16	My Images	**0.96**
-	Kaggle Dataset	0.97
DenseNet201	My Images	0.91
-	Kaggle Dataset	**0.98**
Fusion	My Images	0.82
-	Kaggle Dataset	0.97
-	Mixed Dataset	0.91

Table 1: Training performance of VGG16, DenseNet201, and Fusion models.

Model	Training Dataset	Validation Dataset	Accuracy
VGG16	My Images	Kaggle Dataset	0.47
-	My Images	Mixed Dataset	0.74
-	Kaggle Dataset	My Images	0.60
-	Kaggle Dataset	Mixed Dataset	0.74
DenseNet201	My Images	Kaggle Dataset	0.49
-	My Images	Mixed Dataset	0.74
-	Kaggle Dataset	My Images	0.57
-	Kaggle Dataset	Mixed Dataset	0.71
Fusion	My Images	Kaggle Dataset	**0.78**
-	My Images	Mixed Dataset	0.74
-	Kaggle Dataset	My Images	**0.82**
-	Kaggle Dataset	Mixed Dataset	0.91
-	Mixed Dataset	My Images	0.80
-	Mixed Dataset	Kaggle Dataset	0.95

Table 2: Cross-dataset evaluation of VGG16 and DenseNet201.

4.2 Live Classification

4.2.1 *Classification.*

The previous tables show encouraging results and performance achieved by the different models developed. Testing live classification in different environments, with multiple backgrounds and

various people performing the signs, is also a crucial part. Live classification matters more than validation tests, as the robot will not always be in the same environment, and the goal is for it to adapt to different environments and users.

After performing 48 signs across various backgrounds and people performing them, 32 were correctly predicted, which results in an accuracy of 66.6%. This is 15–20% lower than what was expected based on the cross-validation accuracy. Nonetheless, this accuracy is still promising, as it provides better results than random guessing, which would achieve an accuracy of 25%.

Table 3 shows the prediction distribution across the four classes. The sign South is the most recognized one, followed by Right and Left. They all achieved decent—even great—scores, apart from the sign North. Its very low score of 16.6% shows that the model struggles to recognize it. This might be due to confusion with the sign Right, as both are similar in the sense that two fingers—the index and the middle finger—are extended either upward or downward and separated from the rest of the hand.

	Incorrectly Predicted	Correctly Predicted	Success %
Left	3	9	75.0%
North	10	2	16.6%
Right	2	10	83.3%
South	1	11	91.6%

Table 3: Live classification performance using the fusion model.

4.2.2 Latency.

Another key parameter to consider is the overall latency of the system. Over 100 classifications, the mean time it took the fusion model to make its prediction was 0.24 seconds. Given that the model was run on an average laptop with no additional or high-end computational resources, and considering the large size of the model, an average prediction time of 0.24 seconds is quite good.

The total time to run all 100 predictions was 4.23 minutes. This includes capturing the frame, sending it to the computer, performing the prediction, sending the result back to the robot, and the robot confirming the movement. On average, the entire process for one prediction, including both communication and computation took 2.63 seconds.

This timing is acceptable given the resources used. Moreover, a response time under three seconds is generally considered sufficient for daily tasks, especially in accessibility applications.

5 CONTRIBUTIONS TO THE FIELD

This project will contribute to the field of computer vision in multiple ways. First, by developing innovative techniques for static and dynamic gesture recognition and combining multiple CNN architectures, I aim to discover novel and efficient methods for recognizing hand gestures. While many architectural types have been explored, there is still room for combining different machine learning techniques in creative ways. Additionally, if I have the opportunity to work on dynamic gesture recognition, my contributions will help advance the understanding of accurately detecting when a gesture starts and ends. This capability is crucial for future improvements in the field of sign language detection, and my work will play a part in addressing this challenge.

Furthermore, my project will demonstrate the integration of gesture recognition with Arduino robotics for real-time applications. While significant progress has been made in facilitating human-computer interaction, my work will contribute specifically to the development of human-robot interaction in the context of a sign-controlled robot. Finally, the scalable framework I propose for creating gesture-controlled robots has potential applications in other areas, such as accessibility and education. For instance, individuals with voice impairments can benefit from advancements in human-computer interaction through sign language, and similar implementations could be valuable in educational settings.

Ultimately, my work aims not only to develop new techniques in the field of sign language detection but also to contribute to the broader effort to enhance accessibility and education through innovative human-robot interaction solutions.

6 FUTURE WORK

Summarize my work and outline future research directions.

REFERENCES

[1] Aldridge, R., Brandt, T., and Parikh, C. "Autonomous Robot Design and Build: Novel Hands-on Experience for Undergraduate Students." *Proceedings of the 2016 ASEE North Central Section Conference*, 2016, pp. 1-9.

[2] Barbhuiya, A. A., Karsh, R. K., and Jain, R. "CNN-Based Feature Extraction and Classification for Sign Language." *Multimedia Tools and Applications*, vol. 80, no. 3, 2021, pp. 3051-3069. Available at: https://link.springer.com/article/10.1007/s11042-020-09829-y.

[3] Chanda, P., Mukherjee, P. K., Modak, S., and Nath, A. "Gesture-Controlled Robot Using Arduino and Android." *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 6, no. 6, 2016, pp. 227-234. Available at: https://www.researchgate.net/publication/304624684_Gesture-Controlled-Robot_Using_Arduino_and_Android.

[4] Dadashzadeh, A., Tavakoli Targhi, A., Tahmasbi, M., and Mirmehdi, M. "HGR-Net: A Fusion Network for Hand Gesture Segmentation and Recognition." *arXiv preprint*, arXiv:2002.08695, 2020. Available at: https://paperswithcode.com/paper/hgr-net-a-fusion-network-for-hand-gesture.

[5] Ghashami Mina. "An Implementation of VGG," Towards Data Science, 2023. Available at: https://towardsdatascience.com/an-implementation-of-vgg-dea082804e14/.

[6] Khanna Mukul. Paper Review: DenseNet – Densely Connected Convolutional Networks, 2019. Available at https://medium.com/data-science/paper-review-densenet-densely-connected-convolutional-networks-acf9065dfefb.

[7] Kopuklu, O., Gunduz, A., Kose, N., and Rigoll, G. "Real-Time Hand Gesture Detection and Classification Using Convolutional Neural Networks." *arXiv preprint*, arXiv:1901.10323, 2019. Available at: https://paperswithcode.com/paper/real-time-hand-gesture-detection-and.

[8] Kuznetsova, A., Leal-Taixé, L., and Rosenhahn, B. "Real-Time Sign Language Recognition Using a Consumer Depth Camera." *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2013, pp. 83-90. Available at: https://doi.org/10.1109/ICCVW.2013.18.

[9] Li, D., Opazo, C. R., Yu, X., and Li, H. "Word-Level Deep Sign Language Recognition from Video: A New Large-Scale Dataset and Methods Comparison." *Papers with Code*, 2020. Available at: https://paperswithcode.com/paper/word-level-deep-sign-language-recognition.

[10] Li, Dongxu, et al. "Word-Level Deep Sign Language Recognition from Video: A New Large-Scale Dataset and Methods Comparison." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020, pp. 1-10. Available at: https://dxli94.github.io/WLASL/.

[11] Ullah, S., Mumtaz, Z., Liu, S., Abubaqr, M., Mahboob, A., and Madni, H. A. "Single-Equipment with Multiple-Applications for an Automated Robot-Car Control System." *Sensors*, vol. 19, no. 3, 2019, p. 662. Available at: https://doi.org/10.3390/s19030662.

[12] Venu, N. "IoT Surveillance Robot Using ESP-32 Wi-Fi CAM Arduino." *International Journal of Food and Agricultural Science (IJFANS)*, vol. 11, no. 5, 2022, p. 198. Available at: https://www.researchgate.net/publication/367380053_IOT_Surveillance_Robot_Using_ESP-32_Wi-Fi_CAM_Arduino.

[13] Xu, Dan, et al. "Online Dynamic Gesture Recognition for Human-Robot Interaction." *Journal of Intelligent & Robotic Systems*, vol. 77, no. 3–4, 2015, pp. 583–596. Available at: https://doi.org/10.1007/s10846-014-0039-4.