American Sign Language (ASL) Recognition using YOLOv5 and EfficientNet

Cong-Thanh Vu, Truong-Huy Ngo, Trung-Thoi Nguyen, Minh-Khoi Pham, Minh-Triet Tran
Vietnam National University Ho Chi Minh City, Vietnam
Honors Program, Faculty of Information Technology
University of Science, Ho Chi Minh City, Vietnam
Email: {19120374, 19120242, 19120384, 18120043}@student.hcmus.edu.vn, tmtriet@fit.hcmus.edu.vn

***Abstract*—Most deaf and severely hard of hearing individuals in the United States and Anglophone Canada use American Sign Language as a primary means of communication. This sign language has also an alphabet like other languages, but each letter is represented by a unique hand pose. To approach American Sign Language, above all, we have to regconize these hand poses, but it is quite difficult to do it. So, we proposed an American Sign Language alphabet recognition model from static images. We use a combination of the YOLOv5 model to detect the hand and the EfficientNet model to classify the signs in the ASL alphabet. The model we proposed provide a good recognition ability with accuracy up to 94.87% in a quite small validation dataset. In general, the approach is applicable to any sign-based language, however, it is useful to firstly evaluate it with the American Sign Language. Ultimately, we are looking forward to facilitate communication between the handicapped.**

I. Introduction

Everyone's everyday life revolves around communication. Unfortunately, some persons have limited hearing and speech ability. As a result, sign language was developed to overcome these barriers and allow individuals to interact normally. Each region in the world will have its own sign language. However, due to time and data constraints, we were only able to launch our model in American Sign Language.

The alphabet is a key feature of most languages, and American Sign Language is no exception. Recognizing the letters in American Sign Language, on the other hand, is a significant barrier for individuals who do not use sign language on a daily basis because there are 26 signs that must be remembered for recognition. It is necessary to have a system in place to recognize these signs. Deep learning is becoming a more powerful technology due to its success, thus we picked it as the primary solution to solve the sign language recognition challenge.

There is presently no publication that addresses the challenge of sign language recognition by computer vision-based approach utilizing two independent models for two phases of hand detection and sign classification. The main contribution of this paper is a novel approach to the problem of sign language recognition. The problem is separated into two phases using two different models in our approach. Our model will recognize the location of the hand in the image in the first stage, and the results will be passed to the next stage, which will categorize the sign corresponding to the letter of the alphabet. We ran the datasets through multiple models and discovered that combining YOLOv5 for detection with EfficientNet for classification produced the best results. This method achieves a satisfying outcome that is equivalent to the sensor-based approach, but it is substantially less expensive to implement.

Datasets are required to train the models in order to carry out this research. We discovered three datasets for the detection stage and two datasets for the classification step from diverse online sources. We also employ a mixture of the dataset we produced ourselves, which we call SASL, during the classification step. The SASL dataset is made up partly from screenshots from YouTube videos that we found when searching for "ASL alphabet", the others taken by hand.

The rest of the paper is structured as follows: Section II displays related work, while Section III discusses our approach. Section IV describes the experiments in detail, while Section V summarizes our findings and makes recommendations for further research. Finally, in Section VI, we present some results from inference on proposed models.

II. Related Work

Because of the importance of the American Sign Language Alphabet Recognition, there are a lot of studies about this topic. Based on data input, this work can be categorized into two main approaches: computer vision-based and sensor-based methods.

A. Sensor-based methods

In sensor-based methods, more information about data input had been collected by using specialized devices:

- An ASL recognition system use a Cyberglove and a Flock of Birds as input devices in [6]. Theses devices provides the joint angle values and the position and orientation of the hand in 3-D space for input data. Multi-dimensional Hidden Markov models used in this system to learn new gestures from the features extracted from the sensory data.
- In [5], an automatic ASL to text translation is proposed utilising five flex sensors and a 3-axis accelerometer to collect data, which is then digitally stored to compare with a lookup table for recognition.
- The device is called Leap motion controller (LMC) is used in [15] to capture human hands for sign recognition.

It returns skeleton data and two infrared-radiation images. From these data, the angle features and the visual features are extracted by a skeleton module and a vision module. Output of skeleton module which use a two-layer network is a feature vector with the dimension of 16, while output of vision module which use Convolution Neural Network is two feature vectors with the dimension of 1024. Finally, these vectors are fed into a fusion neural network to output the predicted label. However, this approach is difficult to apply because of unnatural user experience, difficulties in setting up the system and high costs.

B. Computer Vision based methods

Nowadays (in 2021), the progress of computer vision and deep learning makes computer vision-based method more popular:

- A method for using deep convolutional network to classify ASL alphabet and digit is presented in [14]. Images for each sign were captured, then they were preprocessed by removing the background using background-subtraction techniques and resizing to 200x200 pixels. A CNN architecture which consists of multiple convolutional and dense layer is used to train data. Data augmentation techniques such as rotation and horizontal flip were applied to improve the performance.
- In [11], dataset consists of images which were resized to 244x244 pixels before being used to train the Deep Neural Network. The model was trained on a Squeezenet architecture which comprises of one squeeze layer, one expand layer and two concatenated layers.
- Once captured, the images in the dataset are automatically cropped and normalised to 50x50 pixels grayscale samples through the use of an OpenCV image processing function in Python [9]. The pre-processed images are then used for neural network training via multiple data processing layers in the CNN architecture built by materials provided by the Keras library. The implementation of Stochastic Gradient Descent (SGD) in the report as an optimizer provides an ease in installation as well as many computational advantages compared to Gradient Descent.
- In [2], 227x227 pixels grayscale images produced by Bicubic interpolation method, which is further augmented using Robert edge method, were fed into deep CNN architecture optimized by SGD.

Most of the approaches mentioned above require an image in which the ratio of hand gesture to background is fixed to be able to recognize exactly. To solve this problem, at stage one, we built a detector to find bounding boxes that contain hands. Object recognition in images is also a common problem, so we have referenced some of the methods used in the past:

- A two-stage deep learning process is presented to detect hands robustly in [8]. At the first stage, they used two individual state-of-the-art object detectors, region-based convolution neural networks (RCNN) and Faster-RCNN (FRCNN) to train data to detect the hand regions. At the second stage, A deep learning based skin segmentation method is applied to reduce false positives.
- In [1], a lightweight model based on You Only Look Once version 3 (YOLOv3) and DarkNet-53 convolution neural networks is proposed to recgconize hand gestures. This approach is performed without additional preprocessing, image filtering, and enhancement of images.
- Hand-CNN, a novel network for detecting hands in unconstrained images is proposed in [13]. This network is developed from MaskRCNN which is a robust state-of-the-art object detection framework with multiple stages and branches. It is extended to include an additional network branch to predict hand orientation. Hand-CNN incorporates a novel attention mechanism to capture the non-local contextual dependencies between hands and other body parts.

III. Proposed method

A. Overview

Due to the lack of data in the training process. Our proposal is to divide the project into two major stages: recognizing the area containing the hand image and classifying that symbol to give the final answer (the character in the ASL alphabet). This method works according to the following flowchart (Fig. 1):

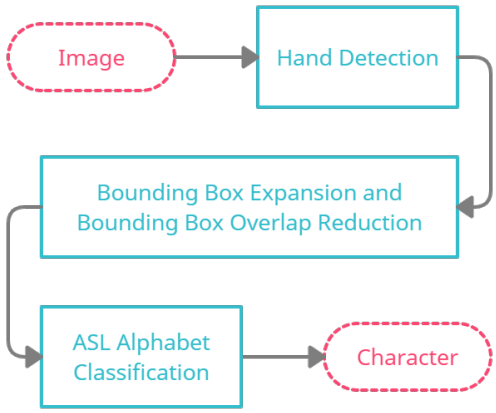

Fig. 1. ASL Alphabet recognization flow

- Hand detection (using the YOLOv5 model): finds the area containing the hand in the user entered images.
- Bounding box expansion and bounding box overlap reduction (bridge between 2 stages): broadens the area of the hand recognized in stage 1 and remove overlapping boxes.
- Sign classification (using the EfficientNet model): from the bounding boxes where the hand has been marked and expanded in the previous stage, classifies the symbol that hand is currently showing and returns the corresponding character in ASL alphabet.

B. Main steps

1) Object detection: Nowadays (in 2021) there are many methods to detect objects in an image, so which model to be chosen is one of our problems. Both one-stage and two-stage

detectors have their own advantages and disadvanges, one of the advantages of two-stage detectors compared to one-stage ones is the model regresses the object location twice (once in each stage) so that the bounding boxes often result better refined than one-stage methods[16]. On the other hand, one-stage detectors provide you a much faster model by skipping the region proposal[12].

With the aim of using our model in real-time to help people who don't know sign language to understand non-verbal conversations, we prioritize inference speed rather than accuracy, so we choose YOLOv5 to optimize our model's inference speed but keeping an acceptable level of accuracy. Fig. 2 shows the relation between average inference time per image and mAP@[0.5:0.95] (mean average precision over different Intersection of Union (IoU) thresholds, from 0.5 to 0.95, step 0.05) of some models:

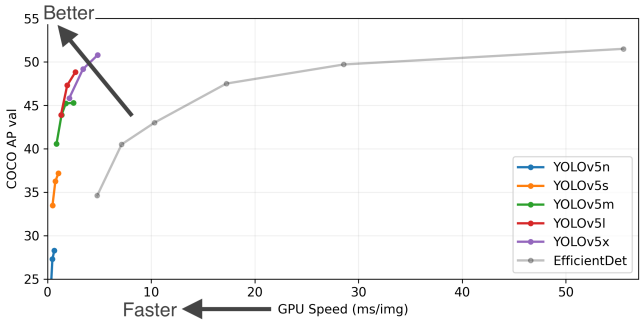

Fig. 2. YOLOv5 in comparison with some models [7]

2) Bounding box expansion: We found that the region of the hand is detected quite well. However, in some cases with extended fingers it may happen that the detector does not return a bounding box containing all the components of the hand. Therefore, we expand the bounding box by 25% in both horizontal directions and by 70% and 25% vertically upward and downward respectively. After conducting scenario-testing, the optimal parameter to expand the bounding boxes has been selected as above. The reason we chose these parameters for the area extension is that we found that most of the symbols in the sign alphabet are facing upwards, and when extended upwards, it will take more space because the hand is more visible. We don't expand the bounding boxes much more, because then the boxes will probably be too wide and we can't control what's in that area. Here are 2 pictures (Fig. 3 and Fig. 4) to compare with and without bounding box expansion, in this case, if the area is not enlarged, the feature will be missing, so it also greatly affects the classifier:

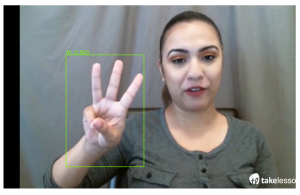

Fig. 3. with Box expansion

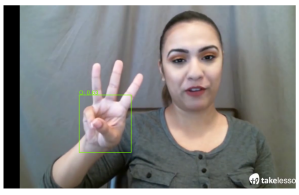

Fig. 4. without Box expansion

3) Bounding box overlap reduction: In some instances, our model detects a hand multiple times. So we've reduced the duplication of the bounding boxes by removing the boxes that have been replaced by 75% or more of the area with another box. The following figures (Fig. 5 and Fig. 6) compare before and after the reduction of the number of boxes that overlap each other:

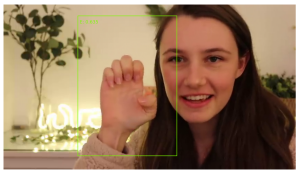

Fig. 5. with Bounding box overlap reduction

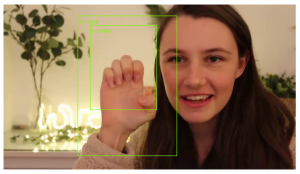

Fig. 6. without Bounding box overlap reduction

4) Image classification: After detecting the hand areas in the image in the first stage and dealing with the bounding box related issues, we crop the bounding boxes. We will then take the cropped image as input to the classifier to indicate the character to be represented in that image. There are many architectures that support this work, after looking at a few architectures, we decided to choose EfficientNet for the classifier because it performed well on the dataset we used.

IV. Experiments

We use Adam optimization algorithm with a learning rate of 0.01 for all models. Our works were conducted on a computer with an nVIDIA Tesla T4 graphics card (15109.75MB of VRAM).

A. Datasets

We have examined quite a lot of different datasets, then we synthesize the results based on mAP metric (for detector) and accuracy on the validation set (for classifier) to choose the best dataset and use it for the final model. The datasets that have been surveyed by us include:

- Hands datasets for object detection (Table I): Hand Dataset[1], EgoHands[2], COCO-Hand[3] and TV-Hand[4]. Overview of these datasets was shown in Fig. 7.
- ASL datasets for classification (Table II): We use 3 datasets, the first one named ASL-Alphabet[5] contains 24 characters (the entire English alphabet except the letter J and Z because of motion when posing those signs) and 3,000 images each. The second one named Significant (ASL) Sign Language Alphabet Dataset[6], its distribution

[1]Hand Dataset by Arpit Mittal, Andrew Zisserman and Phil Torr (University of Oxford): https://www.robots.ox.ac.uk/~vgg/data/hands/

[2]EgoHands by IU Computer Vision Lab (Indiana University): http://vision.soic.indiana.edu/projects/egohands/ from [3]

[3]Microsoft's COCO Dataset: http://vision.cs.stonybrook.edu/~supreeth/COCO-Hand.zip

[4]TV-Hand: http://vision.cs.stonybrook.edu/~supreeth/TV-Hand.zip

[5]Kaggle: ASL-Alphabet by @grassknoted: https://www.kaggle.com/grassknoted/asl-alphabet.

[6]Kaggle: Significant (ASL) Sign Language Alphabet Dataset by @kuzivakwashe: https://www.kaggle.com/kuzivakwashe/significant-asl-sign-language-alphabet-dataset.

of classes was shown in (Fig. 8), it is a combination of the ASL-Alphabet dataset we mentioned above and ASL Finger Spelling Dataset[7]. The last dataset is created by ourselves, some of it was comprised of screenshots of ASL characters taken from YouTube videos using the search phrase "ASL alphabet" while the remainder was captured manually by us. We only use the SASL dataset in conjunction with the other two datasets as the validation and test set.

TABLE I
SUMMARY OF HAND DATASETS FOR DETECTION

Dataset Name	Training Set	Validation Set	Classes	Size
Hand Dataset	4069	738	1	201M
EgoHands	3840	480	1	320M
COCO-Hand	19873	6624	1	885M
TV-Hand	7123	2375	1	2.32G

Fig. 7. Some images and annotations from datasets for hand detection

[7]ASL Finger Spelling Dataset by Nicolas Pugeault: https://empslocal.ex.ac.uk/people/staff/np331/index.php?section=FingerSpellingDataset.

TABLE II
SUMMARY OF ASL DATASETS FOR CLASSIFICATION

Dataset Name	Images	Images/Class	Classes	Size
ASL-Alphabet	87000	3000	29	1.02G
Significant ASL Dataset	80000	2600 to 3400	27	1.89G
SASL	600	3 to 5	24	22MB

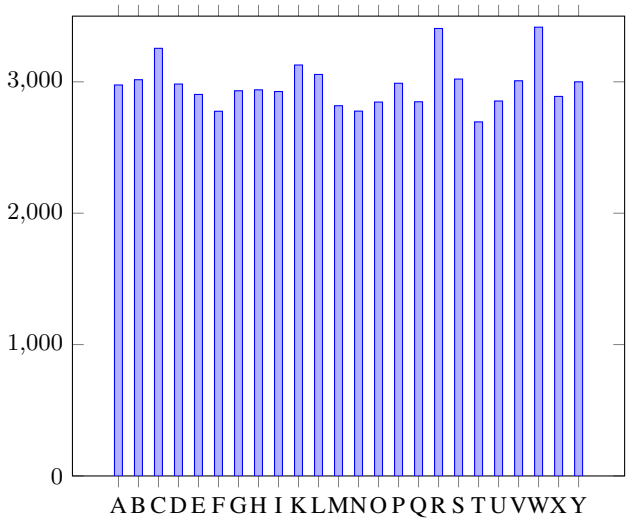

Fig. 8. Distribution of characters in Significant ASL Dataset

B. Data preprocessing

- The datasets for training the hand detector have various annotation formats, so we converted all annotations to the same COCO format for ease of training. We also found that these datasets contain bounding boxes that extend beyond the edges of the image, we have cropped the excess so that the bounding box fits into the correct size of the image.
- In the ASL-Alphabet dataset used to train the classifier, we found that these images were cropped from a video, in addition we also found that 30 consecutive pictures looked quite similar, with very little difference, so we assumed this was a 30FPS (frame per second) video. To serve to divide the training and validation set, we only take 1 image out of 30 consecutive images, which also means we have reduced the number of images per character to 1/30 compared to the original dataset.
- For the Significant (ASL) Sign Language Alphabet dataset, we found it was made up by mixing the ASL-Alphabet dataset with the ASL Finger Spelling Dataset plus a few other pictures, which appears to have some errors in the classification of the characters, so we refined this dataset, along with which we also took more pictures for validation. We will cover why we need to make a brand new validation set in the experiments section below.

C. Use of pretrained model

By examining the training process, we found that using pretrained models trained with the ImageNet set will significantly improve the scores based on the mAP metric (for detector) and the accuracy on the validation set (for classifier), along with that the training time is also consequently reduced.

D. Augmentations

1) Data for detection: The images in the training set are resized to 512x512 pixels, but keep the same aspect ratio, the blanks are filled with black. The training dataset used by us does not ensure that we have sufficient images of hands in different positions, shapes, sizes and angles. As a result, we used random rotations with a rotation limit of 20 degrees and a probability of 30% to better recognize hands from different angles. Along with that, we also use vertical/horizontal flip, perpendicular rotation and visual effects like blur, shift, scale, color distortion, random brightness/contrast to enhance learning and recognition of edited images or photos with different characteristics of the environment during shooting.

2) Data for classification: Similar to the data for the detector, we also resize the images to 256x256 pixels and keep the aspect ratio. The symbols in the ASL alphabet hold for horizontal mirroring, and since the data is not balanced between the two dimensions, in the real test cases there may be symbols in both dimensions, so we use horizontal flip augmentation and rotation augmentation (with a rotation limit of 10 degrees) with 30% probability to fix the imbalance on training data and improve the learning ability. Along with that we also use other augmentations like blur, shift, scale, color distortion, random brightness/contrast to add variety to the training dataset, all of them have a probability of 30% to be applied.

E. Training the object detector

We encountered a number of problems related to the datasets we looked at, whereby we found that the majority of hands tend to be taken on the outside (back of the hand) instead of the inside (palm), while the sign language has a lot of poses captured on the palm, so the ability to detect the palm is quite poor, making the selected area small. The hand may be missed or the bounding box does not cover all hand features such as the shape and position of the fingers. However, this drawback has been overcome by expanding the bounding box that we presented in the Methodology section. We examined these datasets on different sizes of YOLOv5, mAP scores are summarized in the table below (Table III):

TABLE III
MAP SCORES ON DIFFERENT SIZES OF YOLOV5 AND DIFFERENT DATASETS

Model (YOLOv5)	COCO-Hand	TV-Hand	EgoHands	Hand Dataset
S	0.2742	0.0720	0.7400	0.1919
M	0.2941	0.0773	0.7442	0.2150
L	0.3007	0.0753	0.7491	0.2122
X	0.2757	0.0783	0.7436	0.1570

We found the EgoHands dataset to give a pretty good mAP on the validation set. However, when we tested it on a test dataset with different lighting conditions and shot from the opposite side, our model can't accurately detect the position of the hand. In our opinion, the cause of this phenomenon is because the EgoHands set only has pictures of 2 people playing tiles, the images of fingers are almost all together, so there is not enough information for the model to detect the images with open fingers. The photos in this dataset all have roughly the same lighting conditions and resolutions, so the model will make worse predictions if we need to infer a photo with outdoor lighting conditions. The Hand Dataset also gave bad results in the test images because of quite poor image quality, so that the model can't extract enough features to detect hands.

The dataset COCO-Hand and TV-Hand give better results than the other two datasets in the real cases, COCO-Hand have data under a variety of shooting conditions. The number of images is also large enough for the model to learn and detect quite well under many different conditions.

F. Training the image classifier

We began by examining on two sets of data obtained from the internet. The datasets we used did not have a validation set, so we initially tried stratified split the images in the dataset with a ratio of 0.8 and 0.2 for the training set and validation set respectively. We trained with the datasets just split and we get very high accuracy on the validation set. However, when we put this model to use in practice, the ASL classification ability was not as good as the result on the validation set. We think the main reason is because the dataset we have collected has been cut from the same video with adjacent frames, resulting in these frames being almost the same, when validating with the dataset like then the accuracy index does not have much meaning.

We then experimented on the previous two datasets, but instead of splitting them into training, validation, and test sets, we will utilize the SASL dataset to undertake validation and test sets. Although the accuracy on the validation set is not high, when applied in practice, it gives better results than before. This table (Table IV shows the accuracy on the SASL validation set:

TABLE IV
ACCURACY OF DIFFERENT SIZES OF EFFICIENTNET ON THE VALIDATION SET FROM SASL

Model (EfficientNet)	ASL-Alphabet	Significant (ASL) Sign Language Alphabet Dataset
B3	0.4851	0.4908
B4	0.4908	0.8155
B5	0.1881	0.5631
B6	0.4738	0.5583

We found that the EfficientNet-B4 and B5 give acceptable accuracy on a pretty quick training and inference time, so that we choose B4 and B5 as our primary sizes to be trained and examined. Here is a table (Table V) comparing the accuracy of the EfficientNet-B4 and B5 (on the test set containing 154 images from the SASL dataset) with the split validation set from the original training set (Case A) and the SASL validation set (Case B):

TABLE V
ACCURACY COMPARISON BETWEEN DIFFERENT VALIDATION SETS

Model (EfficientNet)	ASL-Alphabet		Significant (ASL) Sign Language Alphabet Dataset	
	Case A	Case B	Case A	Case B
B4	0.4545	0.5259	0.6201	0.7208
B5	0.1104	0.2208	0.4805	0.4675

G. Resampling models with k-fold Cross-Validation

After training and examining the prediction results on the actual test data set, we found that our model often mistakenly recognizes nearly identical poses. Since our training data is full of possible poses, we think this poor recognition is due to the lack of validation data, so we use k-fold Cross-Validation method to improve the results. We initially tested with k=5, but didn't get good results, so then we chose k=7 for our models, each using an EfficientNet architecture.

We split the validation set into 7 parts thanks to sklearn library[4], with the i-th fold we will keep the i-th part of the validation set as the new validation set, and the rest are merged into the training set.

To classify which pose an image contains, each of the 7 trained models will give a prediction result, then we will pass the result through a softmax layer, the result is a 24-dimensional vector now (equal to the number of characters in the ASL alphabet that we trained) contains the probabilities that the image contains a pose of the i-th character. Then we will average those vectors and return the prediction result which is the character with the highest probability. Since we chose k=7, we trained 7 models on the same EfficientNet-B4 architecture with the same augmentations as before. When training each model individually, we recorded the accuracy and F1-score on the validation set as shown in the table below (Table VI):

TABLE VI
ACCURACY AND F1-SCORE ON SASL VALIDATION SET OF 7 FOLDS

Fold	Accuracy	Balanced accuracy	F1-score	Epochs (max accuracy)	Epochs (total)
1	0.9263	0.9375	0.9329	18	23
2	0.9239	0.9257	0.9182	31	35
3	0.9024	0.9014	0.8994	17	22
4	0.9241	0.9347	0.9293	14	20
5	0.9487	0.9444	0.9450	17	28
6	0.9351	0.9451	0.9402	6	12
7	0.9067	0.9132	0.9088	25	32

When 7 models trained on 7 different folds were used to predict signs on the same test set as previously (154 photos from the SASL dataset), the accuracy increased significantly, reaching 90.91%. Meanwhile, the validation set's accuracy can reach 94.87% (according to Table VI).

V. CONCLUSIONS AND FUTURE WORK

Combining hand detection and sign classification is a new approach to the problem of ASL character recognition in images which we want to present. The YOLOv5 is quite effective in hand recognition, and the results are delivered quickly, but the total inference time is still sluggish due to the k-fold cross validation approach employed at the classification stage. To accomplish our initial aim of running this model in realtime, the model must reach a maximum inference time of 1/24s (for 24FPS videos). Although we have improved the detection of the bounding box containing the hand and fixed the cases where a hand is detected multiple times, the ability to classify characters in the ASL alphabet is still not optimized due to the lack of validation data.

The overall inference time can be reduced by putting classification models on an asynchronous flow. That task, in our opinion, should be try in the near future. Additionally, the work is not limited to the ASL Alphabet, but also extends to the expression of ideas and concepts through animation. From there, this work will create a language connection between separate individuals in the most convenient way.

VI. APPENDIX

During the testing process, we discovered that the model is quite effective at detecting signs with numerous unique and easy-to-distinguish characteristics. The model is frequently misunderstood when it comes to signs that are easily confused from different perspectives. These figures (Fig. 10, Fig. 9 and Fig. 11) visualized from our model's prediction results on the test set:

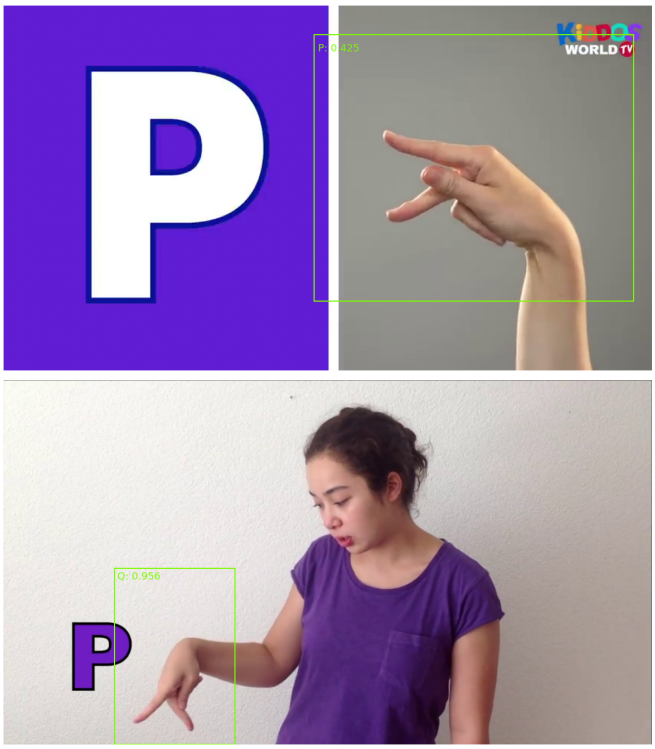

Fig. 9. The letter P is recognized correctly, however the confidence score is not high because the letter has many shooting angles that one or more of our models have not learned well enough. Sometimes the letter P can be mistaken for a Q at the angle of the shot like the photo above, because both the P and Q have thumbs and index fingers pointing down.

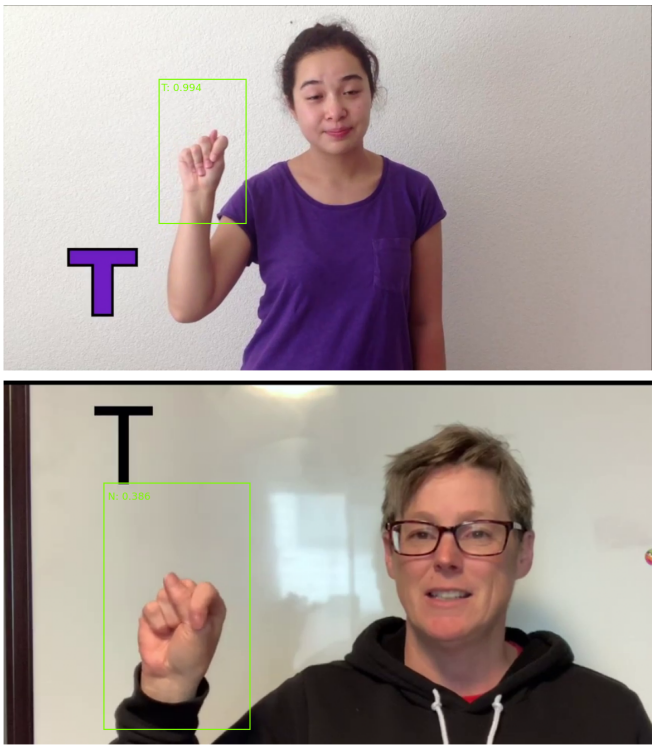

Fig. 10. Letter T was correctly recognized using our method. It can sometimes be mistaken for an M or N because the fingers are not properly placed. Part of it is also because the letters M, N, and T have quite similar features.

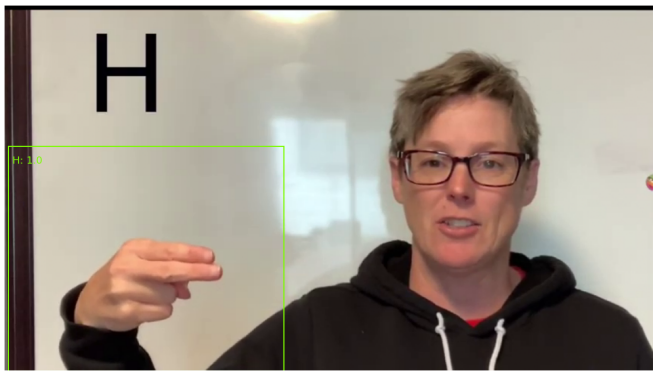

Fig. 11. The letter H has quite clear characteristics and is difficult to confuse with other letters, so it is recognized correctly and has a high confidence score.

REFERENCES

[1] Abdullah Mujahid, Mazhar Javed Awan, Awais Yasin, Mazin Abed Mohammed, Robertas Damasevicius, Rytis Maskeliunas, and Karrar Hameed Abdulkareem. *Real-Time Hand Gesture Recognition Based on Deep Learning YOLOv3 Model*. 2021.

[2] Abdulwahab A. Abdulhussein, Firas A. Raheem. *Hand Gesture Recognition of Static Letters American Sign Language (ASL) Using Deep Learning*. 2020.

[3] Sven Bambach et al. "Lending A Hand: Detecting Hands and Recognizing Activities in Complex Egocentric Interactions". In: *The IEEE International Conference on Computer Vision (ICCV)*. 2015.

[4] *Cross-validation: evaluating estimator performance*. https://scikit-learn.org/stable/modules/cross_validation.html. Online; Accessed: 2021-07-14.

[5] Haydar J., Dalal B., Hussainy S., Khansa L., and Fahs W. *ASL Fingerspelling Translator Glove*. 2012.

[6] Honggang Wang, Ming C. Leu, and Cemil Oz. *American Sign Language Recognition Using Multi-dimensional Hidden Markov Models*. 2006.

[7] Glenn Jocher. *YOLOv5*. https://github.com/ultralytics/yolov5. Online; Accessed: 2021-07-14.

[8] Kankana Roy, Aparna Mohanty, and Rajiv R. Sahay. *Deep Learning Based Hand Detection in Cluttered Environment Using Skin Segmentation*. 2017.

[9] Lean Karlo S. Tolentino, Ronnie O. Serfa Juan, August C. Thio-ac, Maria Abigail B. Pamahoy, Joni Rose R. Forteza, and Xavier Jet O. Garcia. *Static Sign Language Recognition Using Deep Learning*. 2019.

[10] Mingxing Tan, Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2019.

[11] Nikhil Kasukurthi, Brij Rokad, Shiv Bidani, and Dr. Aju Dennisan. *American Sign Language Alphabet Recognition using Deep Learning*. 2019.

[12] Patrick Poirson, Phil Ammirato, Cheng-Yang Fu, Wei Liu, Jana Kosecka, Alexander C. Berg. *Fast Single Shot Detection and Pose Estimation*. 2016.

[13] Supreeth Narasimhaswamy, Zhengwei Wei, Yang Wang, Justin Zhang, and Minh Hoai. *Contextual Attention for Hand Detection in the Wild*. 2019.

[14] Vivek Bheda and Dianna Radpour. *Using Deep Convolutional Networks for Gesture Recognition in American Sign Language*. 2017.

[15] Wenjin Tao, Ze-Hao Lai, Ming C. Leu and Zhaozheng Yin. *American Sign Language Alphabet Recognition Using Leap Motion Controller*. 2018.

[16] Xin Lu, Quanquan Li, Buyu Li, and Junjie Yan. *MimicDet: Bridging the Gap Between One-Stage and Two-Stage Object Detection*. 2020.