Sign Language Detection

1st Jayanth Dasari(*)
B.Tech Computer Science
ABV-IIITM, Gwalior
Anantapur, India
dasarijayanth06042002@gmail.com

2nd Satya Pavan Kalyan Vemula(*)
B.Tech Computer Science
ABV-IIITM Gwalior
Hyderabad, India
vspavankalyan@gmail.com

3rd Jay Shah
B.Tech Computer Science
ABV-IIITM Gwalior
Mumbai, India
jayshah0726@gmail.com

***Abstract*—We aim to close the gap in communication with impaired people and make life much easier for people with disabilities for the rest of the people to interact with them. We have trained different models on the two separate and Augmented American sign language datasets. In the various trained and finetuned models, MobileNet gave the best results, 66.06% accuracy on the augmented dataset1 in the object recognition methods; yolov5 gave the best accuracy in the object detection models in the second dataset, the one with bounding boxes. We chose the best performing model, Yolov5 (89% accuracy), for creating a complete pipeline from sign language to speech. Yolov5 model's output is input to the silero's Text-to-Speech pre-trained model, which gives the speech version as output.**

***Index Terms*—Model, Efficiency, Implementation, Vision-based approach, Data, Sign-to-speech,text-to-speech,sign-to-text**

I. Introduction

According to the WHO's study, more than 7

There isn't one universal sign language for all, every country has its respective sign language, but there might be similarities. Currently, there are more than 300 sign languages worldwide[21]. Most languages we have been exposed to have both a spoken and a written form, whereas sign language has neither, which raises the misconception that sign language is not actual language. Sign languages exhibit almost all features that spoken languages have, such as developing naturally rather than artificially and being a rule-governed communication system. Sign languages do not share the grammar of their spoken counterparts. ASL(American Sign Language) is widely used among various sign languages.

In ASL, there are distinct symbols for many characters, words, and phrases. The language is complex due to the use of so many symbols. Fingerspelling expresses Sign Language by using alphabets and other representational characters to spell out the text.

People with hearing and speaking difficulties face daily social isolation and miscommunication issues. This project is motivated to provide assistive technology that allows differently disabled people to communicate in their language(ASL is used in this project).

Sign language recognition is a vast area for research where much work has been done, but still, various things need to be addressed. Researchers have developed efficient data acquisition and classification methods divided into Sensor-based approaches(or direct measurement techniques) and vision-based methods [3][7]. Vision-based and Sensor-based approaches are the two main approaches to sign language recognition. In a vision-based approach, a camera(in general) reads the human body, typically hand movements, and interprets sign language from the gestures. They are further classified into static Recognition, detection of 2-D images, and dynamic Recognition, real-time live capture of the gestures. In sensor-based methods, real-time hand finger movements are monitored using leap motion sensors, interpreted using pseudo gloves[10]. It is inconvenient to wear gloves in the real world even though it gives high accuracy. Here, we have worked on Vision-based sign language approaches.

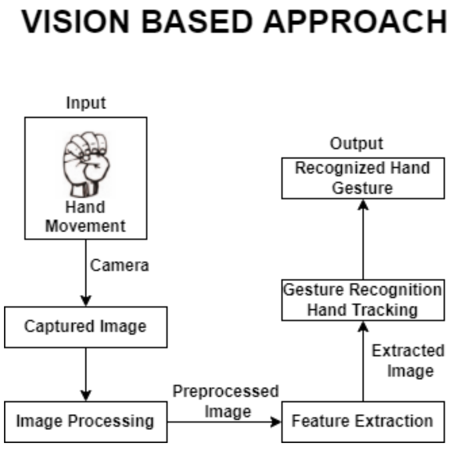


Fig. 1. Vision Based Approach

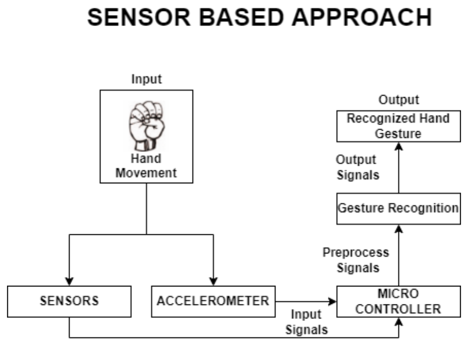


Fig. 2. Sensor Based Approach

II. Literature Survey

A. Vision-Based Approach

Object Detection (One-Stage Object Detection Methods, Two-Stage Object Detection Methods). Key-Related Research Vision-based approaches are classified into two types: Static Recognition and Dynamic Recognition. Static Recognition deals with static gestures (2D images); Dynamic Recognition is a real-time live capture of the gestures. Recognition of these approaches is performed using object detection recognition ML models.

B. Object Detection and its methods

Object detection, an important computer vision task, detects visual objects instances of certain classes in digital images such as photos or video frames. Object detection aims to develop computational models that provide the most basic information computer vision applications need.

Image processing techniques or deep learning techniques are used to perform object detection. Image processing techniques are unsupervised and don't require historical data for training. Deep learning techniques depend on supervised training. The computation power of GPUs limits their performance, advancing rapidly year by year.

Modern state-of-the-art Object detection methods are classified into one-stage and two-stage object detection methods. 1. In general, object detection has two phases: 2. Finding the number of objects. Classifying every object and estimating its size with a bounding box regression. One-stage detection methods(also known as single-shot detectors) combine both tasks into a single step, and bounding boxes are predicted over the images without the region proposal step. These achieve higher performance at the cost of accuracy and are not good at recognizing irregularly shaped or a group of small objects. One-stage methods prioritize inference speed and are comparatively super fast. So, they are used for real-time applications. YOLO family(YOLOv4-Scaled, YOLOR, and soon), SSD, and RetinaNet are some of the most popular one-stage object detection SOTA models.

Two-stage detection methods, at first, find a region of interest using conventional computer vision methods and use this cropped region for classification. These methods have the highest detection accuracy but are slower due to the many inference steps per image. Regional Convolutional Neural Network(R-CNN) and its evolutions Faster R-CNN and Mask R-CNN, G-RCNN are some of the most popular two-stage object detection SOTA models.

YOLO, R-CNN, Mask R-CNN, MobileNet, and SqueezeDet are the most widely used and popular object detection algorithms.

C. Key-Related Research

Video trimming, sign extraction, video background modeling, sign feature representation, and classification are some of the challenges in Sign Language Recognition, an evolving research area in computer vision. In the past[8], all attempted problems have met considerable success and are instrumental in developing the SOTA for SLR. Various 1D/2D/3D models are proposed in the literature with little success to bring the model close to real-time implementation [9].

One of the research studies using a sensor-based approach is Chuan et al.[1] developed an American Sign Language Recognition system using a leap motion sensor. They performed classification using K-Nearest Neighbor and Support Vector Machine and achieved accuracies of 72.78% and 79.83%, respectively.

Andrew Ng, Hinton, LeCun, Bengio, et al. have performed fundamental research on CNNs and improved the CNN algorithm's performance and structural optimization[2]. Yann LeCun et al. in [3] highlighted that deep CNN is a breakthrough in image, video, audio, and speech processing. Extraction of the complex head, hand movements and their constantly changing shapes for Recognition of sign language is considered a challenging problem in computer vision[4]. G. A. Rao et al. [4] has developed a sign language recognition approach based on CNN for a dataset consisting of 200 different sign languages viewed from 5 different angles with varying background environments and achieved 92.88% recognition accuracy.

The computer vision-based sign language recognition approaches rely only on extracting discriminative spatial and temporal data from RGB images. Most computer vision techniques will initially try to track and extract the hand regions before their classification [21]. The current state-of-the-art hand detection methods also use face detection and subtraction and subtraction of background to recognize only the moving parts in a scene [23][24]. 2D CNN extracts spatial features of input images, while RNN captures the long-term temporal dependencies among input video frames. VGG16 pre-trained on ImageNet to extract spatial features and then feed the extracted features to a stacked GRU[5].

3D convolutional networks are used to establish the holistic representation of each frame and the temporal relationship between frames in a hierarchical fashion. Inflate 2D filters of the Inception network trained on ImageNet, thus obtaining well-initialized 3D filters[6].

The unclear temporal boundaries of a specific movement of actions in continuous videos are one of the main challenges. This paper detects their temporal locations from within continuous videos by collecting an ASL dataset that has been annotated with the time intervals for each ASL word[7]. A visual attention-based framework model proposed in [11] works well for constant video backgrounds and is chosen for accuracy and computation time.

S Suresh et al. [12] proposed a sign language recognition system that classifies six different sign languages using a Deep Neural Network (DNN). Two models with Adam and SGD optimizers are compared, Adam optimizer has more accuracy.

Using background elimination, image processing techniques are implemented to extract gestures and CNN models, which achieved 96.20% accuracy under various noisy and illumination conditions. Hand and gesture detection in complex scenes was implemented by Qiang et al. [15] through SqueezeNet with Convolutional networks. The SqueezeNet model ensures the feature extraction network dramatically reduces the parameter weights of the whole network with compression to improve the speed of hand and gesture detection. This lightweight architecture with a prediction fusion network was designed using a residual structure on the deconvolutional network. This system performed gesture recognition at high precision using a single convolutional layer at multiple scales.

Sign Language to Text Conversion Using Deep Learning [14] proposed a model that uses deep learning techniques to solve the problem. They used CNNs for image depiction and classification, split the dataset into the train, and the test data in a 9:1 ratio and achieved an accuracy of 98.67%. The paper does not continuously convert directly from video to text and recommends the use of video pre-processing and frame-wise splitting of data into images to achieve the same.

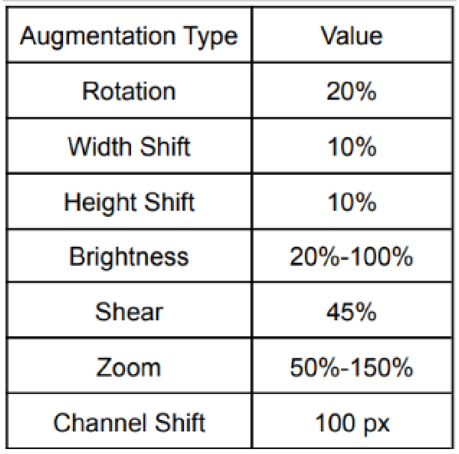
III. METHODOLOGY

In this project, we used two datasets to detect the Signs by any user at any place having any background in real-time. We trained many pre-trained models and a custom CNN model on the augmented dataset versions of the original datasets. We used two datasets taken from the kaggle and Roboflow websites[19][20], one with a common background and the other with versatile backgrounds with bounding boxes. We have used famous SOTA object detection models and finetuned them for sign language recognition. Some of the models are Mobilenet, Yolov5, Faster RCNN. We have trained the total yolov5 model on the dataset with bounding boxes and various backgrounds and achieved 89% accuracy. Converted Text output of signs, making it possible to listen to the sign's text directly.

A. Datasets

We used two datasets, one with typical sign images and the other with bounding boxes included. Both datasets are augmented, making datasets more generalized for better optimal results. Applying image augmentation expands the dataset and reduces overfitting.

1) Dataset1: The data set, a collection of images of alphabets from the American Sign Language, is taken from kaggle[19]. This Dataset Contains 84,000 images of 29 classes[A-Z, del, space, nothing]. It is a clean dataset where all the sign's glosses are almost in the middle of all the images and have clean, same background. Data augmentation is performed to generalize data. Given images of the size 200x200 pixels are scaled to the input size required(224x224 pixels) for pre-trained models used.



Augmentation Type	Value
Rotation	20%
Width Shift	10%
Height Shift	10%
Brightness	20%-100%
Shear	45%
Zoom	50%-150%
Channel Shift	100 px

Fig. 3. Data Augmentation Applied

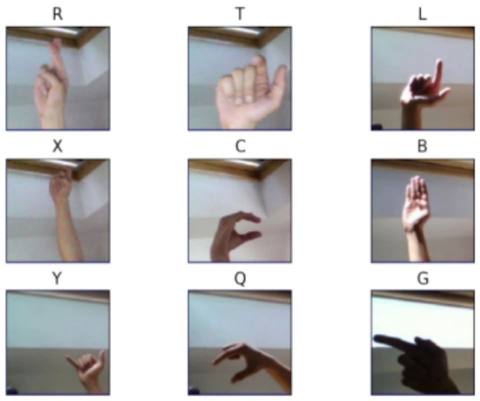


Fig. 4. Dataset1 before augmentation

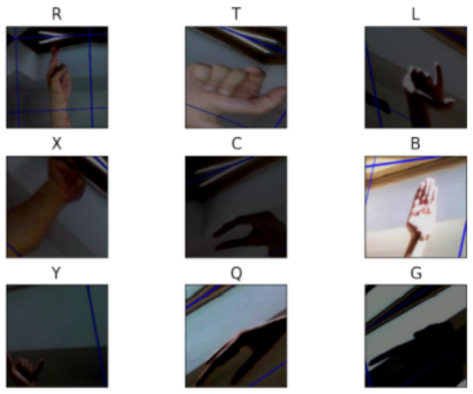
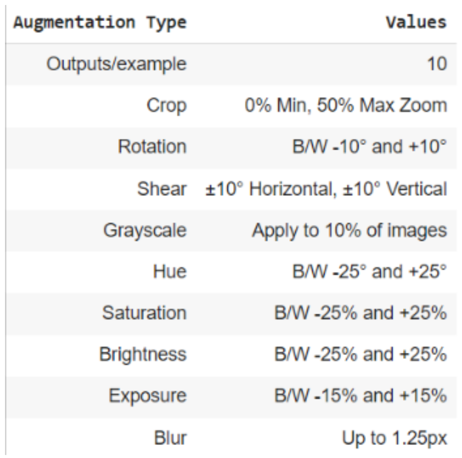


Fig. 5. Dataset1 after the augmentation

2) Dataset2: The object detection dataset of each ASL letter with a bounding box is taken from the Roboflow website. This dataset contains 700 training images with 26 classes[A-Z] and has more diversified backgrounds. Data augmentation is performed to get more data and a generalized one. These images are scaled to 640x640 pixels to match the input size of requirements of pre-trained models.



Augmentation Type	Values
Outputs/example	10
Crop	0% Min, 50% Max Zoom
Rotation	B/W -10° and +10°
Shear	±10° Horizontal, ±10° Vertical
Grayscale	Apply to 10% of images
Hue	B/W -25° and +25°
Saturation	B/W -25% and +25%
Brightness	B/W -25% and +25%
Exposure	B/W -15% and +15%
Blur	Up to 1.25px

Fig. 6. Dataset-2 before augmentation

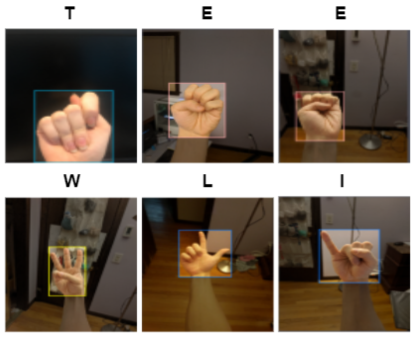


Fig. 7. Dataset-2 before augmentation

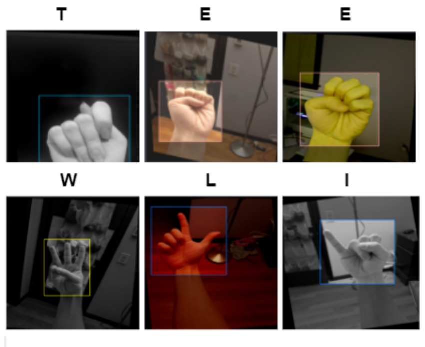


Fig. 8. Dataset-2 after augmentation

the architecture of the top models, we used in this project.

B. YOLOv5

YOLO, an SSD, performs classification and bounding box regression in one step, much faster than most CNN. YOLO is 1000x faster than R-CNN and 100X faster than Fast R-CNN.

The YOLOv5 model is based on the YOLO model pre-trained on the COCO dataset, comprises backbone(New CSP-Darknet53), Neck(SPPF, New CSP-PAN), Hea(YOLOv3 Head). The backbone is a network of convolutional layers to extract rich, informative features from an input image. It is first trained on a classification dataset, like ImageNet, at a lower resolution than the final detection model, as detection requires finer details than classification. In YOLOv5, the Cross Stage Partial(CSP) Networks[26] are used as a backbone. The Model neck is mainly used to generate future pyramids, which helps in object scaling generalization. They use the backbone's convolution layers features with fully connected layers to make predictions on probabilities and bounding box coordinates. In YOLOv5, PANet[27] is used as a neck to get feature pyramids, which helps perform well on unseen data. The head is the network's final output layer, which can be interchanged with other layers with the same input shape for transfer learning.

YOLOv5 uses Leaky ReLU activation for hidden/middle layers and sigmoid activation for the final output layer. It has, by default, an SGD optimizer, which we can change to ADAM. Ultralytics has used Binary Cross Entropy with Logits Loss function[25] for class probability and object score loss calculation. The YOLOv5 loss consists of Classes loss(BCE loss), Objectness loss(BCE loss), and Location loss(CIoU loss).

YOLO struggles to detect small objects in images that naturally appear in groups, as each grid is constrained for only a single entity detection.

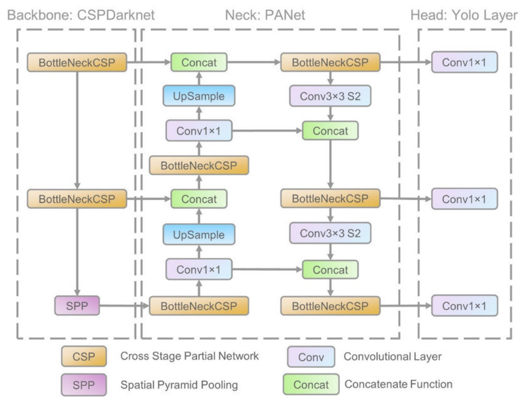


Fig. 9. YOLOv5 Architecture

$$Loss = \lambda_1 L_{cls} + \lambda_2 L_{obj} + \lambda_3 L_{loc}$$

Fig. 10. YOLOv5 Loss Function

```
Model Summary: 191 layers, 7.46816e+06 parameters, 7.46816e+06
gradients
```

Fig. 11. YOLOv5 Loss Function

C. Faster R-CNN

Initially, R-CNN models select several proposed regions(nearly two thousand) from an image and label their categories and bounding boxes based on predefined classes given. After that, they use a convolutional neural network to perform forward computation to extract features from each proposed area. So, this detailed approach takes way more time than YOLO models.

To improve the performance and cut down the training time, Fast R-CNN was developed. Unlike R-CNN, Fast R-CNN runs the neural network once on the whole image for the regions of interest, comparable to YOLO's architecture. Still, YOLO remains faster due to its code simplicity.

Fast R-CNN is more accurate than the original R-CNN as the Region of Interest (ROI) Pooling method, slicing out, reshaping each Region of Interest from the network's output tensor, and classifying is used at the end of the network. Because of this recognition technique, fewer data inputs are required to train Fast R-CNN and R-CNN detectors.

We took the Detectron2 FasterRCNN-FPN model(Feature Pyramid Network) with a Resnet50 backbone pre-trained on the COCO dataset, comprising Backbone network, Region proposal network(RPN), ROI Heads (Box Head). Faster R-CNN computes a score for each RPN region, defining an object's confidence in that region and extracting areas that cross the confidence threshold. ROI Heads output generic predictions based on detection score and regression coefficients used to improve the coordinates of the anchors containing objects generated. These class predictions are then filtered based on the score threshold, and NMS is applied.

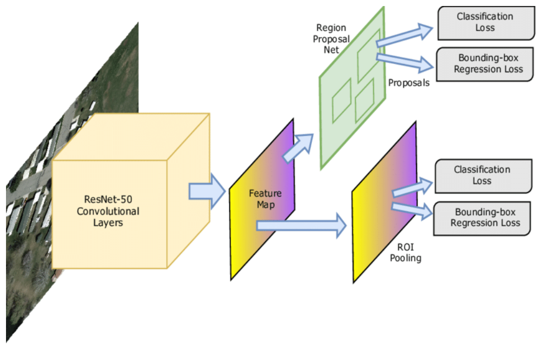


Fig. 12. Faster R-CNN Architecture

D. MobileNet

MobileNet, a single-shot multi-box detection network open-sourced by Google, uses the Caffe framework for object detection tasks. As its name suggests, it is designed for mobile applications usage. It uses depthwise separable convolutions, significantly reducing the number of parameters compared to others resulting in a lightweight deep neural network.

Depthwise separable convolution is a depthwise convolution followed by pointwise convolution. Mobile Nets mainly differ from CNN models in that they split the convolution into a 3x3 depth-wise convolution and a 1x1 pointwise convolution followed by batch normalization and ReLU. The model output is a typical vector of the tracked object data.

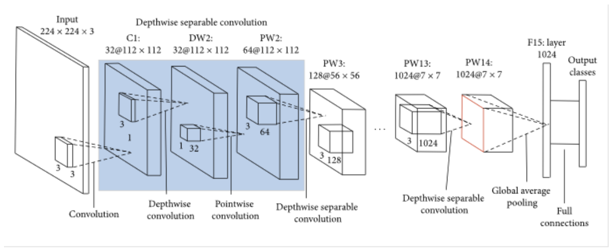


Fig. 13. MobileNet Architecture Overview

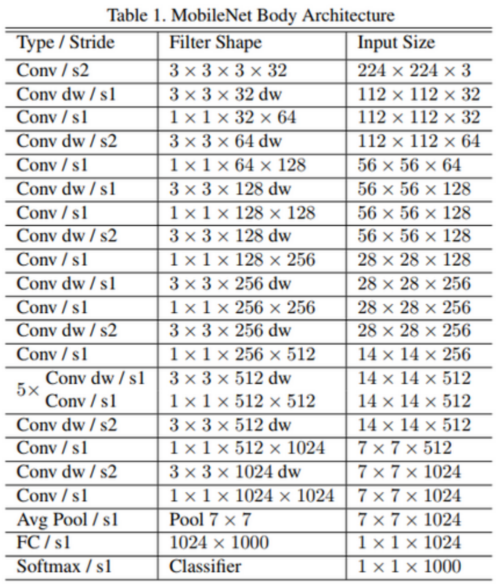


Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Fig. 14. MobileNet Architecture In-Depth

E. Custom Deep CNN Model

We build our own custom deep CNN model for sign language recognition along with pre-trained models used. Our model consists of Deep CNN layers, Batch Normalization layers, dropout regularization layers, used adam optimizer, and ReLU activation layers.

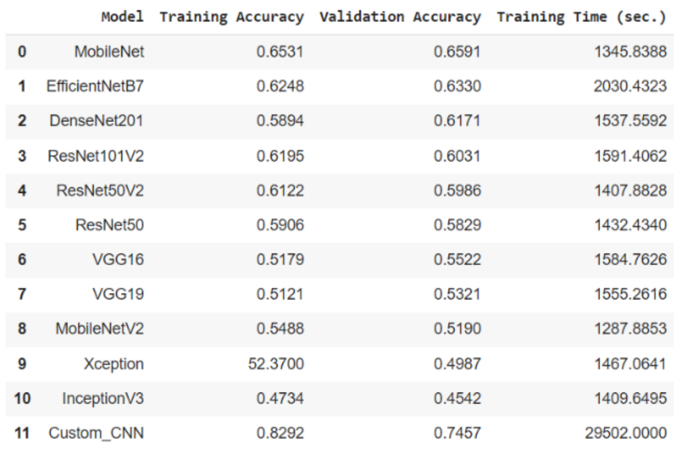
F. Silero TTS

We used the silero model for text-to-speech, trained originally on a private dataset, which results are faster than real-time on one CPU thread. Silero models are licensed under a GPU A-GPL 3.0 License[28], where you have to provide source code if you are using it for commercial purposes

IV. EXPERIMENTS

A. *Experiment 1*

Using Dataset-1,
Aim: To recognize the ASL Hand Signs
Input: Sign Images of 224*224 pixels
Models used: Custom CNN model, MobileNet, and other pre-trained models.
Optimization: ADAM,
Activation : ReLu
Regularization: Dropout,
Pooling: Max Pooling



	Model	Training Accuracy	Validation Accuracy	Training Time (sec.)
0	MobileNet	0.6531	0.6591	1345.8388
1	EfficientNetB7	0.6248	0.6330	2030.4323
2	DenseNet201	0.5894	0.6171	1537.5592
3	ResNet101V2	0.6195	0.6031	1591.4062
4	ResNet50V2	0.6122	0.5986	1407.8828
5	ResNet50	0.5906	0.5829	1432.4340
6	VGG16	0.5179	0.5522	1584.7626
7	VGG19	0.5121	0.5321	1555.2616
8	MobileNetV2	0.5488	0.5190	1287.8853
9	Xception	52.3700	0.4987	1467.0641
10	InceptionV3	0.4734	0.4542	1409.6495
11	Custom_CNN	0.8292	0.7457	29502.0000

Fig. 15. Experiment Results

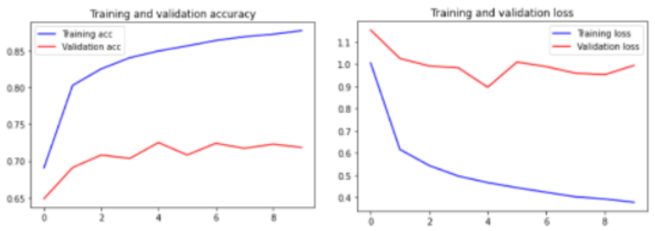
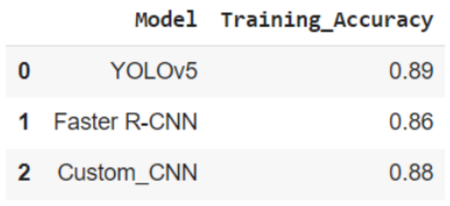


Fig. 16. MobileNet

1) Results: We got the best results with the MobileNet model with 65.91 accuracy, which takes less time to train and is suitable for real-time implementation. We got comparatively low results because of the augmentation. Our Custom model gave the best result with 74.57 accuracy, when we didn't consider time.

B. *Experiment 2*

Aim: To detect and recognize the hand ASL gestures.
Input: Sign Images of size 600*600 pixels.
Models: YOLOv5, Faster R-CNN, Custom Deep CNN Model.
Optimization: ADAM,
Activation : ReLu
Regularization: Dropout, BatchNormalization
Pooling: Max Pooling



	Model	Training_Accuracy
0	YOLOv5	0.89
1	Faster R-CNN	0.86
2	Custom_CNN	0.88

Fig. 17. YOLOv5

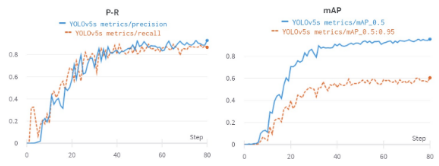


Fig. 18. YOLOv5

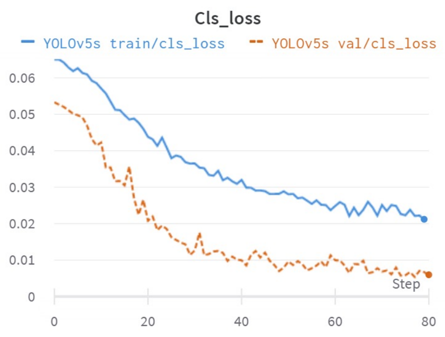


Fig. 19. YOLOv5 BoxLoss

1) Results: We got the best results with the yolov5 model with 0.89 accuracy and less time taken and performed well on the validation set unlike our custom CNN model as we didn't use the bounding box concept.

C. Experiment 3

Aim: Converting Text to Speech, Real-Time Implementation of Sign-to-Speech.
Models: YOLOV5, TTS Silero Model.
Inputs: Live Video, String(Text Stored).
Output: String(Stored recognized Text), Speech format of text.

1) Results: We used the yolov5 model, the best-suited model for real-time implementation, and got good results for the validation set. We merged it with silero tts model giving the best output speech results possible.

D. Inference

Our end-to-end model implements complete sign-to-speech conversion in real-time and works well even with new backgrounds comparatively, maintaining inference speed. We used the yolov5 model (0.89 accuracy) for sign-to-text and silero tts for text-to-speech. Our work has some limitations, like distance from the camera affecting the model's accuracy and misclassifications with some letters due to their similarity. And we have used a dataset with only one sign/character per image, which is challenging to use in real life

E. Future Scope

Implementing facial gesture-based Recognition along with vision-based will improve results. We can train models with more and better data, preferably videos dataset. It is better to train a complex model included with LSTM and GRU from scratch to recognize sequences better. We can implement full end-to-end models for both speech-to-sign and sign-to-speech.

References

[1] Chuan CH, Regina E, Guardino C (2014) American Sign Language recognition using leap motion sensor. In: 13th IEEE international conference on machine learning and applications (ICMLA), pp 541–544.

[2] Yoshua Bengio et al., Learning deep architectures for ai. Foundations and trends® in Machine Learning, vol. 2, no. 1, pp. 1-127, 2009.

[3] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, Deep learning. Nature, vol. 521, no. 7553, pp. 436-444, 2015.

[4] G. A. Rao, K. Syamala, P. V. V. Kishore and A. S. C. S. Sastry, "Deep convolutional neural networks for sign language recognition," 2018 Conference on Signal Processing And Communication Engineering Systems (SPACES), 2018, pp. 194-197, DOI: 10.1109/SPACES.2018.8316344.

[5] K. Simonyan and A. Zisserman. 'Very deep convolutional networks for large-scale image recognition, September 2014, DOI: "https://arxiv.org/abs/1409.1556"

[6] J. Carreira and A. Zisserman. Quo Vadis, action recognition? a new model and the kinetics dataset. CVPR, 2017.

[7] Y. Ye, Y. Tian, M. Huenerfauth, J. Liu, N. Ruiz, E. Chong, J. M. Rehg, S. Palsson, E. Agustsson, R. Timofte, et al., "Recognizing American sign language gestures from within continuous videos," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 2064-2073, 2018. Becky Sue Parton, "Sign language recognition and translation: A multi-disciplined approach from the field of artificial intelligence", Journal of deaf studies and deaf education, vol. 11, no. 1, pp. 94-101, 2005.

[8] Zhengzhou Liu, Fuyang Huang, Gladys Wai Lan Tang, Felix Yim, Binh Sze, Jing Qin, et al., "Real-time sign language-recognition with guided deep convolutional neural networks", Proceedings of the 2016 Symposium on Spatial User Interaction, pp. 187-187, 2016. https://doi.org/10.1145/2983310.2989187

[9] Mukul Singh Kushwah, Manish Sharma, Kunal Jain and Anish Chopra, "Sign language interpretation using pseudo glove", Proceeding of International Conference on Intelligent Communication Control and Devices, pp. 9-18, 2017.

[10] P. Kurhekar, J. Phadtare, S. Sinha and K. P. Shirsat, "Real-Time Sign Language Estimation System," 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), 2019, pp. 654-658, DOI: 10.1109/ICOEI.2019.8862701.

[11] S. Suresh, H. T. P. Mithun and M. H. Supriya, "Sign Language Recognition System Using Deep Neural Network," 2019 5th International Conference on Advanced Computing Communication Systems (ICACCS), 2019, pp. 614-618, DOI: 10.1109/ICACCS.2019.8728411.

[12] C. J. L. Flores, A. E. G. Cutipa and R. L. Enciso, "Application of convolutional neural networks for static hand gestures recognition under different invariant features", 2017 IEEE XXIV International Conference on Electronics Electrical Engineering and Computing (INTERCON), pp. 1-4, 2017.

[13] Kartik, P.V.S.M.S., Sumanth, K.B.V.N.S., Ram, V.N.V.S., Prakash, P. (2021). Sign Language to Text Conversion Using Deep Learning. In: Ranganathan, G., Chen, J., Rocha, Á. (eds) Inventive Communication and Computational Technologies. Lecture Notes in Networks and Systems, vol 145. Springer, Singapore.

[14] https://doi.org/10.1007/978-981-15-7345-3_18

[15] B. Qiang et al., ""SqueezeNet and Fusion Network-Based Accurate Fast Fully Convolutional Network for Hand Detection and Gesture Recognition"", IEEE Access, vol. 9, pp. 77661-77674, 2021.

[16] S. Sridhar and S. Sanagavarapu, "SqueezeCapsNet – Transfer Learning-Based ASL Interpretation Using SqueezeNet With Multi-Lane Capsules," 2021 IEEE 12th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON), 2021, pp. 0001-0007, DOI: 10.1109/UEMCON53757.2021.9666590.

[17] Adaloglou, N., Chatzis, T., Papastratis, I., Stergioulas, A., Papadopoulos, G., Zacharopoulou, V., Xydopoulos, G., Atzakas, K., Papazachariou, D. Daras, P. (2019). A Comprehensive Study on Sign Language Recognition Methods. IEEE Transactions on Multimedia.

[18] Dadashzadeh, A., Tavakoli Targhi, A. Tahmasbi, M. (2018). HGR-Net: A Two-stage Convolutional Neural Network for Hand Gesture Segmentation and Recognition. arXiv:1806.05653.

[19] https://www.kaggle.com/grassknoted/asl-alphabet

[20] American Sign Language Letters - v2 aug (roboflow.com)

[21] Konstantinidis, D., Dimitropoulos, K., Daras, P.: Sign language recognition based on hand and body skeletal data. 3DTV-Conference. 2018-June, (2018).

[22] Zheng, L., Liang, B., Jiang, A.: Recent Advances of Deep Learning for Sign Language Recognition. DICTA 2017 - 2017 Int. Conf. Digit. Image Comput. Tech. Appl. 2017- Decem, 1–7 (2017).

[23] Lim, K.M., Tan, A.W.C., Tan, S.C.: A feature covariance matrix with serial particle filter for isolated sign language recognition. Expert Syst. Appl. 54, 208–218 (2016).

[24] Lim, K.M., Tan, A.W.C., Tan, S.C.: Block-based histogram of optical flow for isolated sign language recognition. J. Vis. Commun. Image Represent. 40, 538–545 (2016).

[25] ttps://pytorch.org/docs/master/generated/torch.nn.BCEWithLogitsLoss.html

[1911.11929] SPNet: A New Backbone that can Enhance Learning Capability of CNN (arxiv.org)

[26] ttps://arxiv.org/abs/1803.01534

[27] ttps://github.com/snakers4/silero-models/blob/master/LICENSE