CS- 725 - Foundations of Machine Learning

Course Project

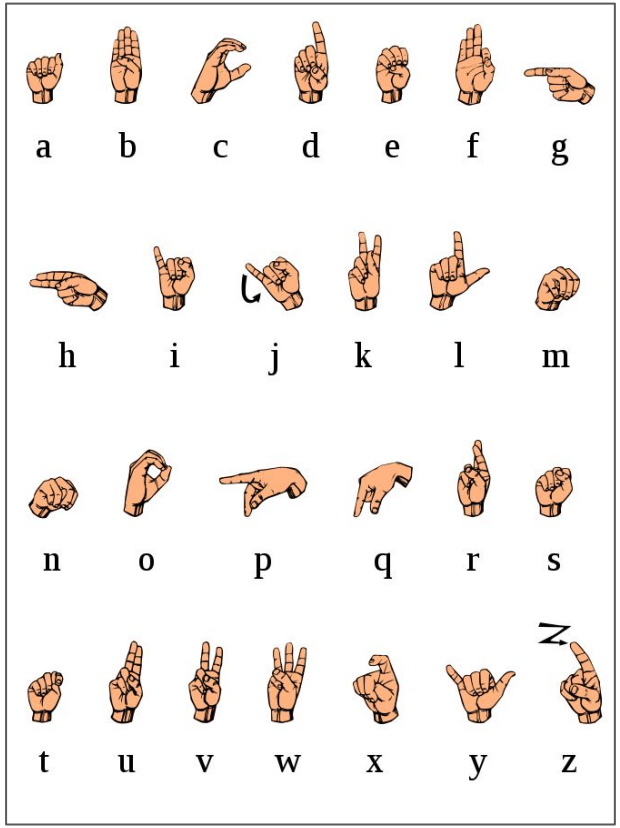
ASL Fingerspelling Detection

Signtists

Task Description / Approach

Aim: To address the challenge of American Sign Language Fingerspelling detection from images.

- To develop a robust and efficient model for detection and classification of ASL Fingerspelling gestures from images.
- The ultimate goal: Real-time classification model.
- First approach: Extract landmarks from input images, use a traditional feed-forward neural network to predict the corresponding English alphabets according to ASL.
- Subsequently, implement YOLO and SSD models to conduct a comparative analysis with the custom model.



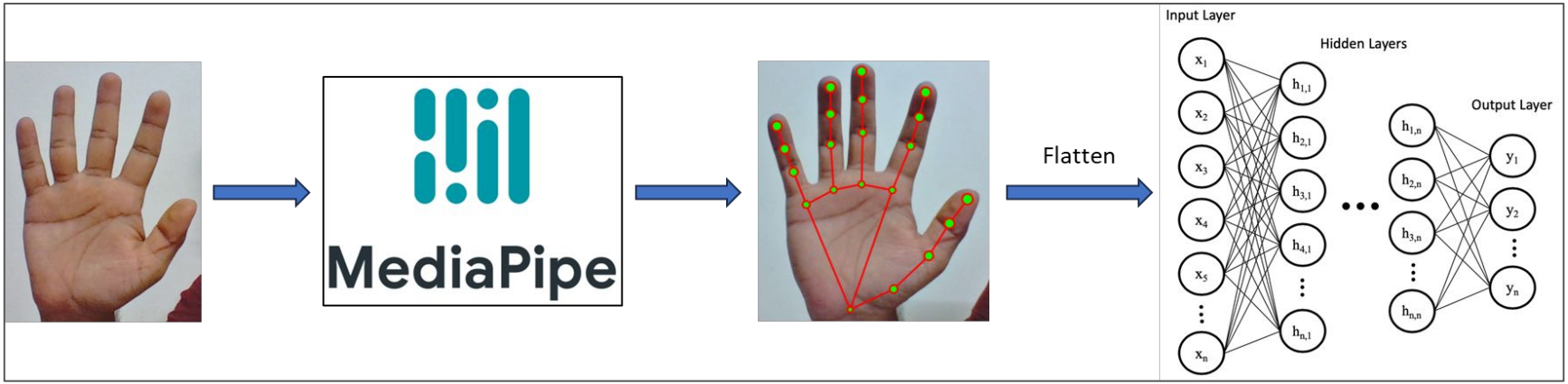
ASL Spelling Gestures

Challenges Addressed

- Stability: Incorrect predictions with slight variations in angle and position.
  - Addressed by data augmentation - created multiple copies of training instances using affine transformations and flipping.
- Inconsistency in signs/symbols/gestures across datasets:
  - Most of the datasets have few signs skipped.
  - Trained two models on two different datasets found online.
  - Trained one model on self-made dataset.
- Overfitting:
  - Regularization
  - Allowing enough variations across training examples.
  - Early stopping

Implementation / Custom Model

- MediaPipe is used to get the 21 landmark / key points from each hand image.
- Each landmark has three components corresponding to spatial coordinates.
- These hand landmarks are flattened and used as the input to the FFNN.
- Categorical cross entropy loss function, with L1 regularization and Adam optimizer, is used for training.



MediaPipe

FFNN

Experiment Details and Main Results

- FFNN Architecture:
  - One hidden layer with 46 neurons.
  - Number of parameters : 4166
  - ReLU activation on hidden layer and softmax on output layer.
  - Number of classes : 26
- Training custom model-3:
  - Randomly picked 400 images for each class from ASL Alphabet dataset.
  - 12 copies of each image are generated through transformations.
- Training custom model-2:
  - Whole dataset(1728 images) from roboflow is used after augmenting with 6 copies of transformed images.
- Training custom model-1:
  - Self-made dataset with around 500 images for each class (~14k images) is used.
- Majority among the predictions from the three models is considered during prediction.

Implementation of YOLO and SSD

- A pre-trained YOLO v8 model is downloaded and fine tuned with one of the datasets for 50 epochs.
- Images are resized to 416x416 for YOLO.
- A pre-trained SSD MobileNetV2 FPN-Lite 320x320 model is downloaded and fine tuned on the same dataset used for YOLO.
- In the MobileNetV2 SSD FPN-Lite, we have a **base network** (MobileNetV2), a **detection network** (Single Shot Detector or SSD) and a **feature extractor** (FPN-Lite).
- Fine tuned the SSD model for 40k epochs.

Comparative Analysis

Performance of models on the custom test dataset:

Metric	Custom Model	YOLO	SSD
Accuracy	0.80	-	-
Precision	0.85	0.85	0.71
Recall	0.80	0.76	0.78
F1 Score	0.82	0.80	0.74

Qualitative Analysis

- Custom Model:
  Custom model - around 80% accuracy
  But Custom Test Dataset is only webcam images with the signs actually facing the camera
  For test images that involve hands facing elsewhere, the performance of the model would depend greatly on Mediapipe's ability to capture the hand landmarks
  For black and white pictures or noisy pictures or when the hand is far away, Mediapipe does not work well. In those cases,predictions are off at times.
  It confuses quite often between U,V and R and M,N.

- YOLO:
  YOLO is background sensitive.
  Custom data includes images with face,body in the background.
  That affects the performance in test evaluation.
  We see in live demo, it fluctuates but it mostly gets the signs correctly.
  But (in general) it works better than custom model for test images with hand facing elsewhere or noisier images.
  It confuses often between M and N. The signs for these are very close.

Conclusions

- Our custom model performs decently well in webcam and custom test data. However, this is not an absolute and this performance alone cannot be used to compare it to YOLO and SSD.
- The custom models individually do not do as well as the combined custom model. The majority vote helps us improve the accuracy by about 5%.
- YOLO performs quite close on the custom dataset. YOLO is more robust in general and is less affected by lighting conditions etc. On the other hand, our model is less affected by background as long as Mediapipe detects well.
- Therefore, the performance of our model is bottlenecked by the performance of Mediapipe.
- YOLO and SSD are still more general and SOTA models. However, our model using the Mediapipe landmark method performs close to them for a scenario where a person is signing at the camera.

Related Works

- Real-time Object Detection and Classification for ASL alphabet
  - This paper presents a comparative study of three models namely Faster RCNN with a ResNet backbone, Faster RCNN with mobileNetv3 backbone, and YOLO v5.
- American Sign Language Detection using YOLO v5 and YOLO v8
  - This papers compares the performance of YOLO v5 and YOLO v8 models on ASL Fingerspelling detection.

Work split

- Custom Model:
  - Manishit Kundu
  - Sai Vivek Mulukuri
  - Vaibhav Patil
- YOLO and SSD:
  - Hemanth Kotaprolu
  - A Snegha
  - Shruthi Akkala
- Everyone is involved during brainstorming and discussions.

Generated on Bing AI

References

1) ASL Alphabet dataset from kaggle
2) Roboflow American Sign Language Letters Dataset
3) MediaPipe by Google
4) Feedforward Neural Network using Keras
5) YOLO v8 documentation.
6) Reference for SSD