Daniel Camilo Fuentes
Angel Delgado
5/16/2025

# Computer Vision Final Project

## Appendix



## Problem Statement:

The lack of accessible, interactive tools for learning American Sign Language creates barriers for both deaf and hearing individuals preventing effective communication between communities. Traditional learning methods provide no immediate feedback on technique, leading to poor skill development and reduced motivation. Our platform solves this problem by creating an engaging application that uses computer vision to provide real-time feedback on sign language gestures, making learning more effective, accessible, and enjoyable for everyone.

# Project Idea:

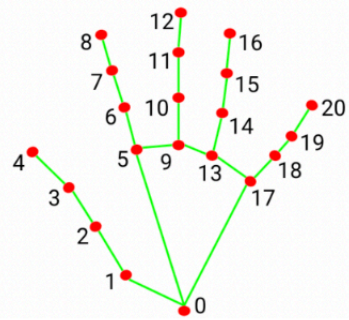
We propose developing an interactive sign language learning platform that functions like "Duolingo" but for sign language. The application will display letters of the alphabet and prompt users to perform the corresponding American Sign Language (ASL) signs in front of their webcam. Using computer vision techniques, the system will analyze the user's hand gestures in real-time, provide and track progress through gamified elements.

# Evidence (Approach and proof that it will get done):

Datasets:https://www.kaggle.com/datasets/grassknoted/asl-alphabet
https://www.kaggle.com/datasets/ayuraj/asl-dataset

Google Mediapipe:
https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker

The hand landmark model bundle detects the keypoint localization of 21 hand-knuckle coordinates within the detected hand regions. The model was trained on approximately 30K real-world images, as well as several rendered synthetic hand models imposed over various backgrounds.




0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

We are using concepts that we have covered in class to implement this:

- Depth and Segmentation for better retrieval of hands from images
    - Can assume that the hand will be closer to the screen compared to the rest of the body
- Feature extraction using mediapipe's hand tracking will provide 21 landmarks points per hand

    - Extract geometric features between landmarks like distances and angles
- Neural Networks:
    - Use the CNN model which was used for image classification to be able to classify sign language to letters from the alphabet
- Interface:
    - Use code from our webcam capture videos
    - Apply masking like the donquixote assignment in order to apply the text for the letter of the alphabet, score, and helpful image of what the actual sign language looks like
    - Apply masking, geometric transformations for the celebration video similar to the UFO video

# Ideas:

**Dataset:**
- Find a dataset with images of sign language and the correct alphabet letter for that sign language
- Use depth/segmentation to isolate the hand/arm and having a blacked out background for each image in the dataset
- Pass each image of the dataset through mediapipe's hand detection and grab the keypoints
- Create a new dataset that instead of using the full image and label, we will use the keypoints and distances between the points and the label

**Neural Network:**
- Train a neural network model that will be trained on the new dataset we created
- Try and achieve a high accuracy
- Could potentially use the CNN model that we used in class as it was an image classification model which is similar as we trying to classify based on the points

**Live Recording:**
- Capture frame by frame of our live video
- Use the depth model or seg to isolate our hands from the screen and apply a blacked out background
- Run the mediapipe on this updated frame of just our hand
- Grab the key points and joints
- Use our neural network we trained to predict what letter the person is doing

**Video Game Aspect:**
- Show a letter of the alphabet to the user
- The user attempts and does the steps in the live recording
- If the model predicted over a certain threshold increase the user's score and show another letter

- When they succeed show a small celebration animation on the screen (mask, kind of like the ufo-parking lot video)
- Could also show the correct sign language on the frame as well to help the user (masking)

**Stretch Goals:**
- If we get letters working, start working on words (as these take full hand gestures not just static holds)
- If we get words, try sentences
- Add a streak of sorts

# 1st Iteration Approach

**CNN Image Classification with Full Image Dataset**

In our initial approach, we attempted to use a Convolutional Neural Network (CNN) trained on an 87,000-image dataset. The dataset contained 29 classes, 26 of which corresponded to the letters A-Z. However, during training on Google Colab, the environment continually crashed due to memory constraints imposed by the size of the dataset. As a result, we sliced the dataset to a more manageable size.

Once preprocessed, we trained the CNN model on this reduced image dataset. During implementation, we attempted to use depth segmentation techniques to isolate the hand in real-time video frames. However, this approach proved ineffective: the real-time processing was too slow and the segmentation was highly inaccurate due to inconsistent image conditions (e.g., hand blur, positioning, and lighting).

At this point, we pivoted to **MediaPipe**, an open-source framework by Google for building multimodal ML pipelines. MediaPipe offers high-performance, real-time pose, face, and hand tracking, and significantly improved the accuracy and speed of hand landmark extraction.

# 2nd Iteration Approach

**Logistic Regression on MediaPipe Keypoints**

For our second iteration, we shifted focus from image classification to hand landmark recognition. We replaced our CNN with a logistic regression model, based on our research suggesting that simpler models often perform better with structured numerical data like keypoints.

We also changed datasets: the new set included 70 clear, labeled images per letter (A-Z), significantly smaller and more manageable. MediaPipe was able to reliably extract 21 hand landmarks per image.

Using these 21 landmarks (each with x, y, z values), we trained the logistic regression model. However, this model struggled with accuracy, often misclassifying clear hand signs such as "I", "O", and "C". This highlighted the limitations of treating raw landmark positions as features.

To improve, we moved away from raw coordinates and instead calculated **relative distances** between key hand points.

## 3rd Iteration Approach

**Using Landmark Distances as Features**

We extracted 17 specific distances per image, including:

- Fingertip-to-wrist distances
- Selected segment distances between finger joints

This improved the model performance notably, as it reduced sensitivity to hand position and orientation in the frame. Signs like "I", "O", and "C" were now more consistently recognized. However, signs involving fist formations or minimal finger separation were still misclassified.

We next returned to using CNNs, applying the CNN techniques we had used successfully in CIFAR-10. But instead of training on image data, we applied the CNN to numerical features (distance vectors). This hybrid approach allowed us to retain the benefits of feature engineering with the learning power of CNNs.

We used the same 70-image dataset and preprocessed each image to extract 21 landmarks with MediaPipe. However, this version of the CNN plateaued at low accuracy.

## Final Iteration: 1D CNN on Full Distance Vectors

We improved further by expanding our distance vector to include all 20 distance features from fingertips to the wrist and between key finger segments. This richer representation helped capture hand shape and pose more accurately.

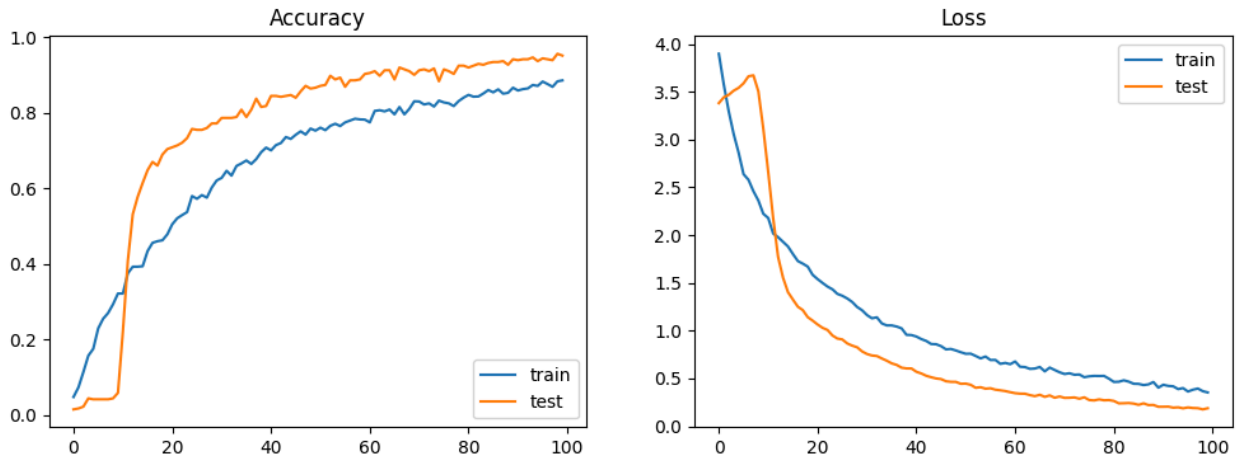
We also:

- Applied **batch normalization** to stabilize learning

- Created **mirrored data** to augment the dataset and improve generalization

The final 1D CNN architecture trained on this distance-based dataset achieved our best accuracy yet. It performed well during real-time implementation, accurately predicting most signs with robust performance across lighting and hand positioning variations.

## Final Training Results




---

The highest training accuracy was 0.8858 obtained in epoch 100
The final training accuracy was 0.8858
The highest validation accuracy was 0.9563 obtained in epoch 99
The final validation accuracy was 0.9515
The lowest training loss was 0.3538 obtained in epoch 100
The final training loss was 0.3538
The lowest validation loss was 0.1764 obtained in epoch 99
The final validation loss was 0.1884
**13/13** ━━━━━━━━━━━━━━━━━━━━ **1s** 30ms/step
**Final Test Accuracy: 0.9515**

## Implementation Overview

The final application combines MediaPipe hand tracking with a trained 1D CNN model for real-time ASL letter recognition. A webcam feed captures live hand gestures, which are processed as follows:

1. **Hand Landmark Extraction**: MediaPipe detects the hand and extracts 21 landmarks per frame.
2. **Feature Engineering**:
   Instead of using raw coordinates, the code computes 20 distance-based features:
   - 5 distances from wrist to each fingertip
   - 3 distances per finger segment (thumb, index, middle, ring, pinky)
3. **Model Inference**:
   These distances are reshaped and passed into the trained 1D CNN. The model returns class probabilities for all 26 letters.
4. **Prediction Filtering**:
   Letters that have already been guessed correctly are masked out to avoid repetition. The model selects the highest-confidence remaining prediction.
5. **User Interface**:
   - Displays the current target letter
   - Shows the current score
   - Provides hint images on demand (via the "H" key)
   - Celebrates correct predictions using a celebrity green-screen overlay video
6. **Game Loop**:
   The user attempts to sign each letter. Upon a correct prediction, the score is updated, the letter is marked as seen, and a new target letter is selected.

The real-time feedback and engaging visual elements make this a complete and interactive ASL learning tool.

## Summary of Results

| Iteration | Approach | Model Type | Input Data | Performance |
|---|---|---|---|---|
| 1st | CNN on raw images (sliced dataset) | CNN | Full images | Crashed on Colab, poor real-time performance |
| 2nd | Logistic Regression | Logistic Regression | 21 landmarks | Poor accuracy on several signs |
| 3rd | Distance-based features | Logistic Regression | 17 distances | Moderate improvement |
| Final | Distance-based with 1D CNN | 1D CNN | 20 distances + batch norm + augmentation | Best performance |

## Conclusion

This project evolved through several well-structured iterations, progressively refining input representations and model architectures. The final 1D CNN using MediaPipe-extracted distance vectors provided the best accuracy and efficiency for real-time ASL letter classification.