ASL Translator: Building Intuition for RNN

Xiangyu Liu
jxiangyu@stanford.edu
Department of Computer Science, Stanford

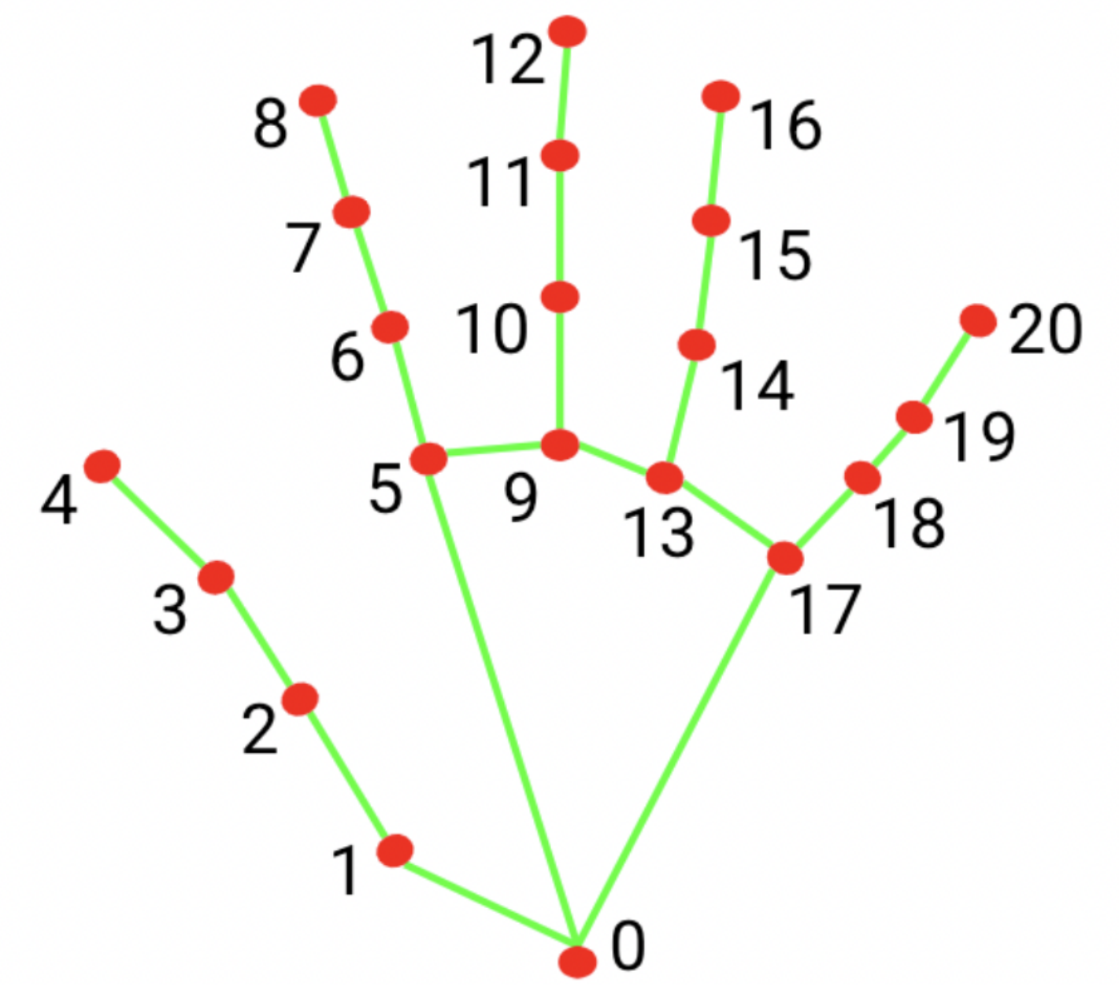
Introduction

ASL is the most commonly used sign language by the deaf and hard of hearing people in U.S. However, learning it is hard and time consuming. Therefore, having an ASL language translator can significantly lower the communication barrier between the deaf and the others. This project aims to solve a 10-class classification problem, a small subset of the overall translation task. Furthermore, it tries to exhibit some common pitfalls raised during training.

Input and Output

Input: Landmark Frame Sequence

Instead of directly working with the raw video, this problem uses the landmark sequence as inputs instead. Landmark for a hand consists of 21 hand-knuckle points as shown in the following figure. Each point has x, y, z coordinates.



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

Figure 1. A Hand Landmark

A training input contains a sequence of frames. Each frame contains a list of landmarks, including left hand, right hand, pose, etc. In the project, I only use the left and right hand landmarks. For instance, the input sequence for "apple" contains 9 frames, each frame contains a captured right hand landmarks. (Note that left hand is not needed to sign "apple", thus not presented in the input.)

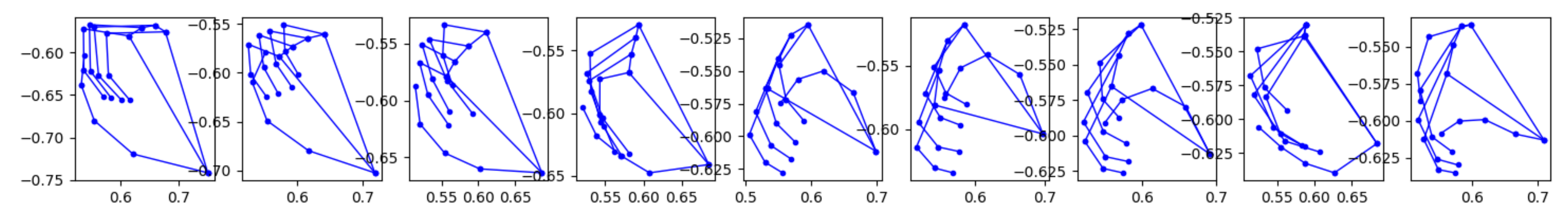


Figure 2. Input sequence for "Apple"

Output: 10-class classification

The output of the model is to predict which word from '['TV', 'after', 'airplane', 'all', 'alligator', 'animal', 'another', 'any', 'apple', 'arm']' the input sequence represents.

Methods

First Model: Nested CNN + RNN network

Intuition: CNN to extract the features and recognize patterns among the points in the landmarks from left and right hands. RNN to process the input sequence.

Model:

$$\sigma = ReLU()$$
$$f(x^{(t)}) = Dropout(MaxPool1d(\sigma(Conv1d(x^{(t)}))))$$
$$h^{(t)} = \sigma_h(W_{hx}f(x^{(t)}) + W_{hh}h^{(t-1)} + \omega_{hb})$$
$$y^{(t)} = \sigma_y(W_{yh}h^{(t)} + \omega_{yb})$$

Second Model: LSTM (Long Short-Term Memory) Model

Directly using a Vanilla LSTM , using hidden state with shape (300,) to process the input directly.

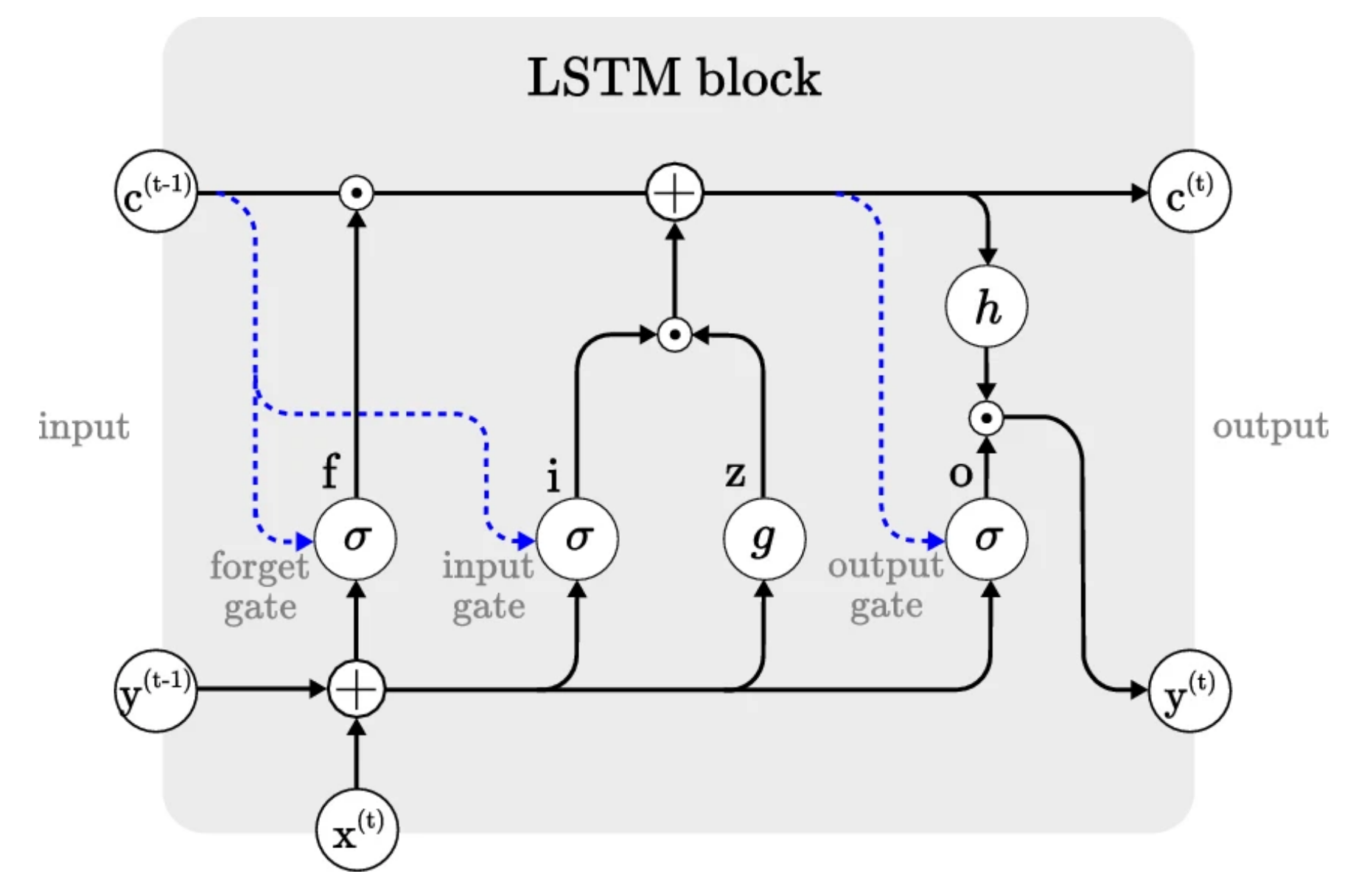


Figure 3. A vanilla LSTM network

Input Gate: $i^{(t)} = \sigma_i(W_{ix}x^{(t)} + W_{iy}y^{(t-1)} + w_{bi})$

Forget Gate: $f^{(t)} = \sigma_f(W_{fx}x^{(t)} + W_{fy}y^{(t-1)} + w_{bf})$

Output/Exposure Gate: $o^{(t)} = \sigma_o(W_{ox}x^{(t)} + W_{oy}y^{(t-1)} + w_{bo})$

New Memory Cell: $g^{(t)} = \sigma_g(W_{gx}x^{(t)} + W_{gy}y^{(t-1)} + w_{bg})$

Final Memory Cell: $c^{(t)} = f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ g^{(t)}$

Final Hidden Layer Output: $y^{(t)} = O^{(t)} \circ \sigma_y(c^{(t)})$

Final Output: $z^{(t)} = \sigma_z(W_{zy}y^{(t)} + w_{bz})$

Experiments and Results Discussion

Vanishing Gradients: Conceptually, the **Nested CNN+RNN Model** should produce a much better prediction accuracy on both training and test datasets. However, in reality it didn't. It suffered significantly from the Vanishing Gradients problem. After training with 400 epochs, the train loss has stayed flat, the gradients for the weights have become 0. On the contrast, the LSTM was able to overcome the vanishing gradient problem and learned successfully.

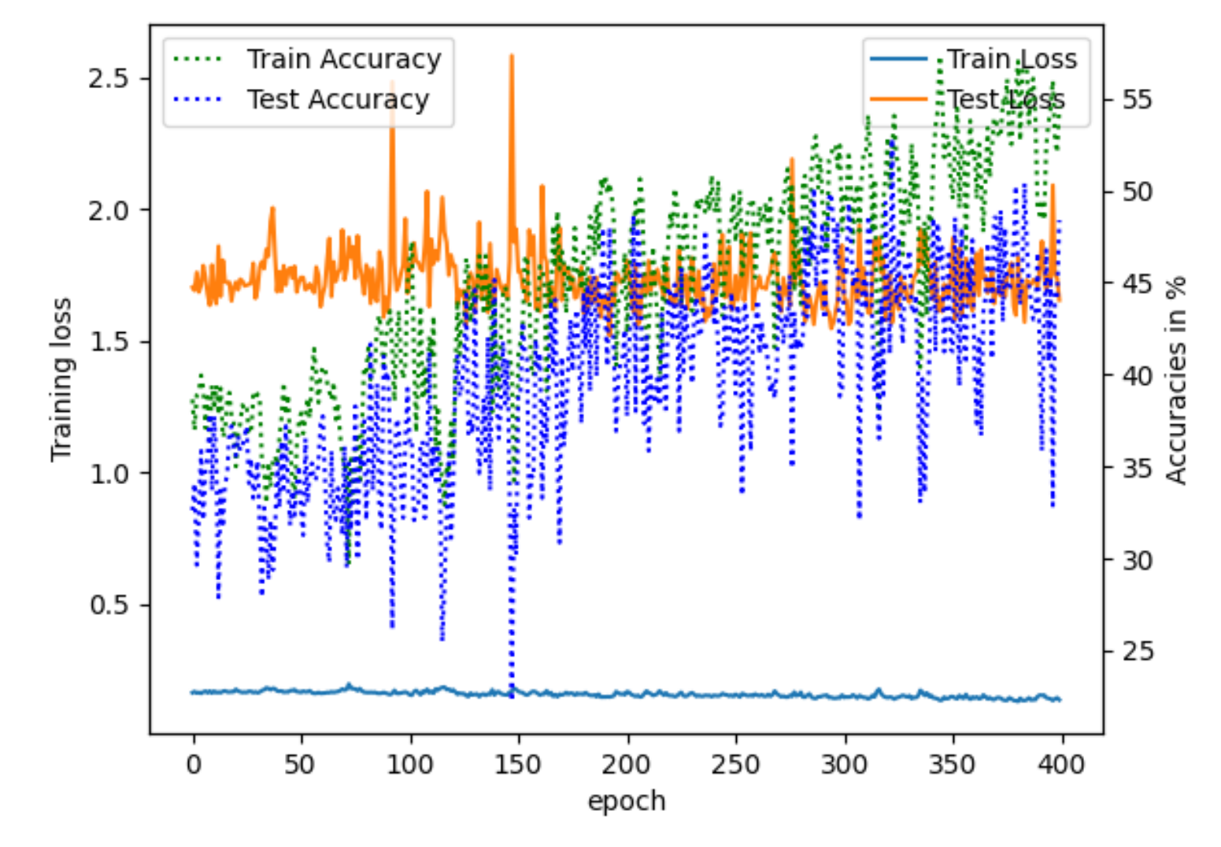


Figure 4. Vanishing Gradients in RNN network

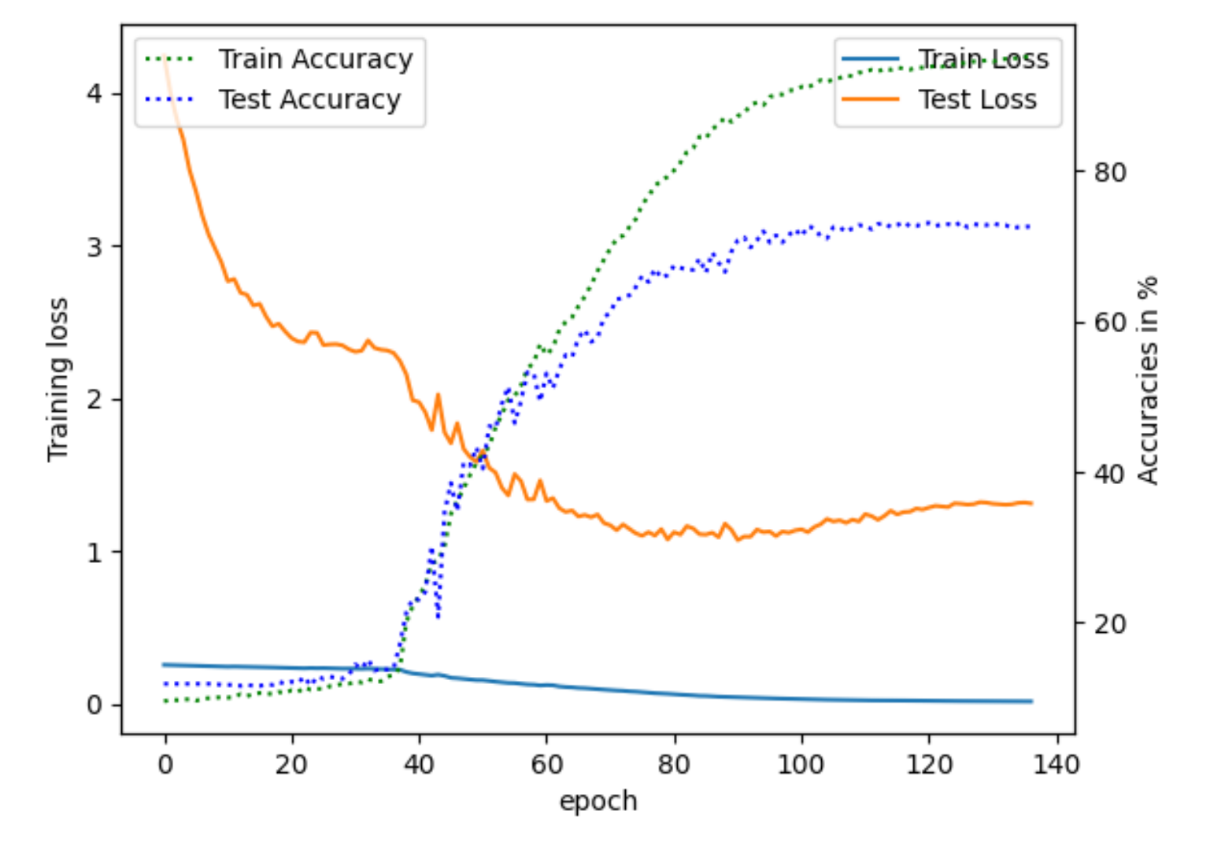


Figure 5. LSTM Learning

Hyper Parameters: Different signs involves different hand movements. Even for the same person, signing the same sign multiple times can still results in multiple different frames. Hence, the hyper-parameters, such as mini-batch size, padding, and learning rate has significant impact on the time it needs to converge, especially when it needs to process a variable-length input sequence.
Padding: because of the variable length input, in order to run mini-batch, the shortest input is padded with 0s to match the longest input length in the batch.
Batch size: prefer smaller batch size to avoid significant padding due to the high variance input length:

Batch Size	Average Epoch Time in Seconds
10	82s
100	216s

Table 1. Mini-Batch Size Time

Learning Rate: StepLR significantly reduces the time to converge, compared to a constant LR.
LSTM Model Result: With StepLR, the LSTM model has converged to 90% accuracy on the training set, 0.028 as the train loss, and 73% accuracy with 1.08 as loss on the test set after 100 epochs without any significant overfitting issue. When validated against the test dataset, the model predicts very well for most of the classes except the "after" class, mainly due to the training data imbalance.

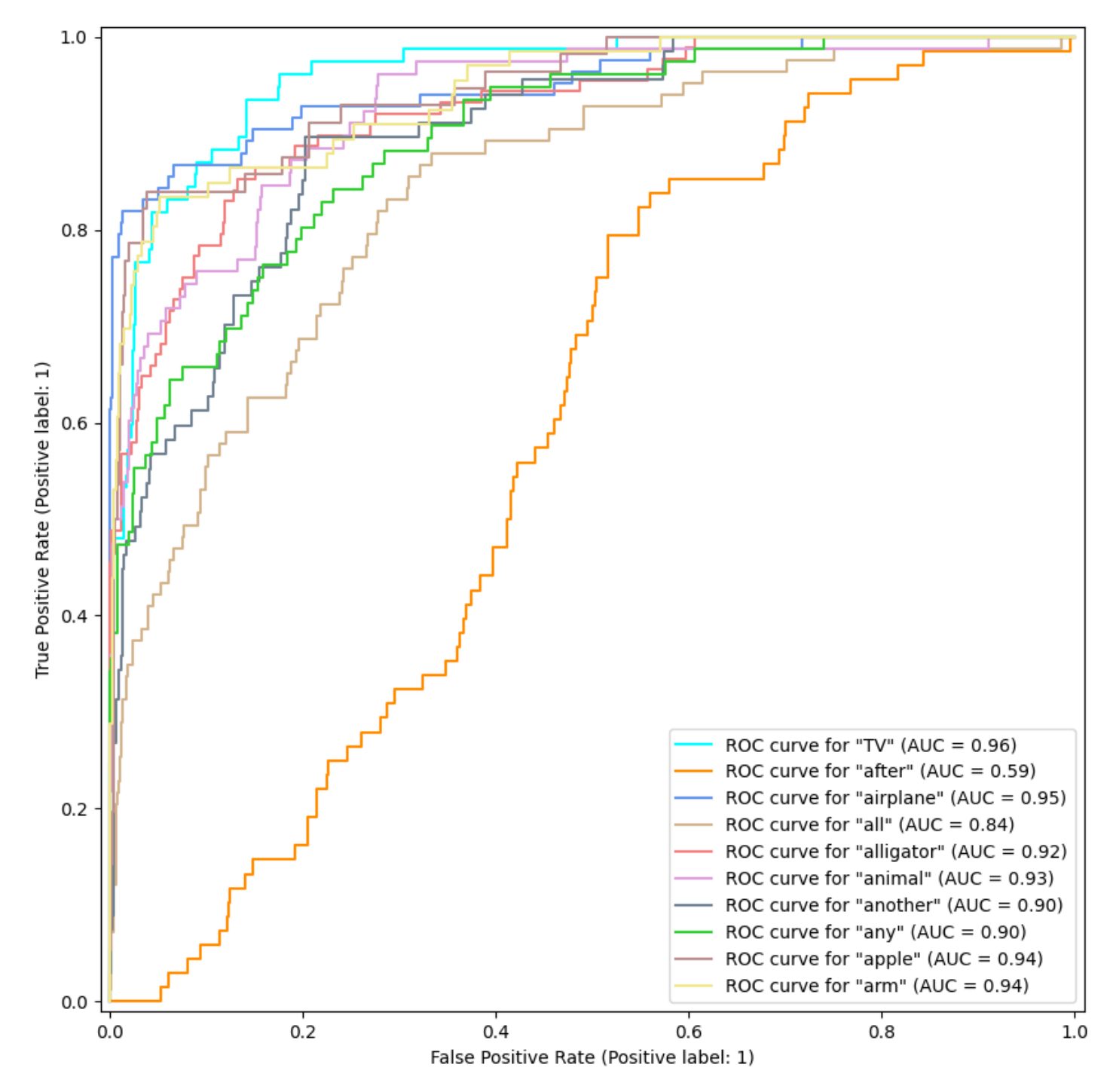


Figure 6. OvR ROC Curve for LSTM

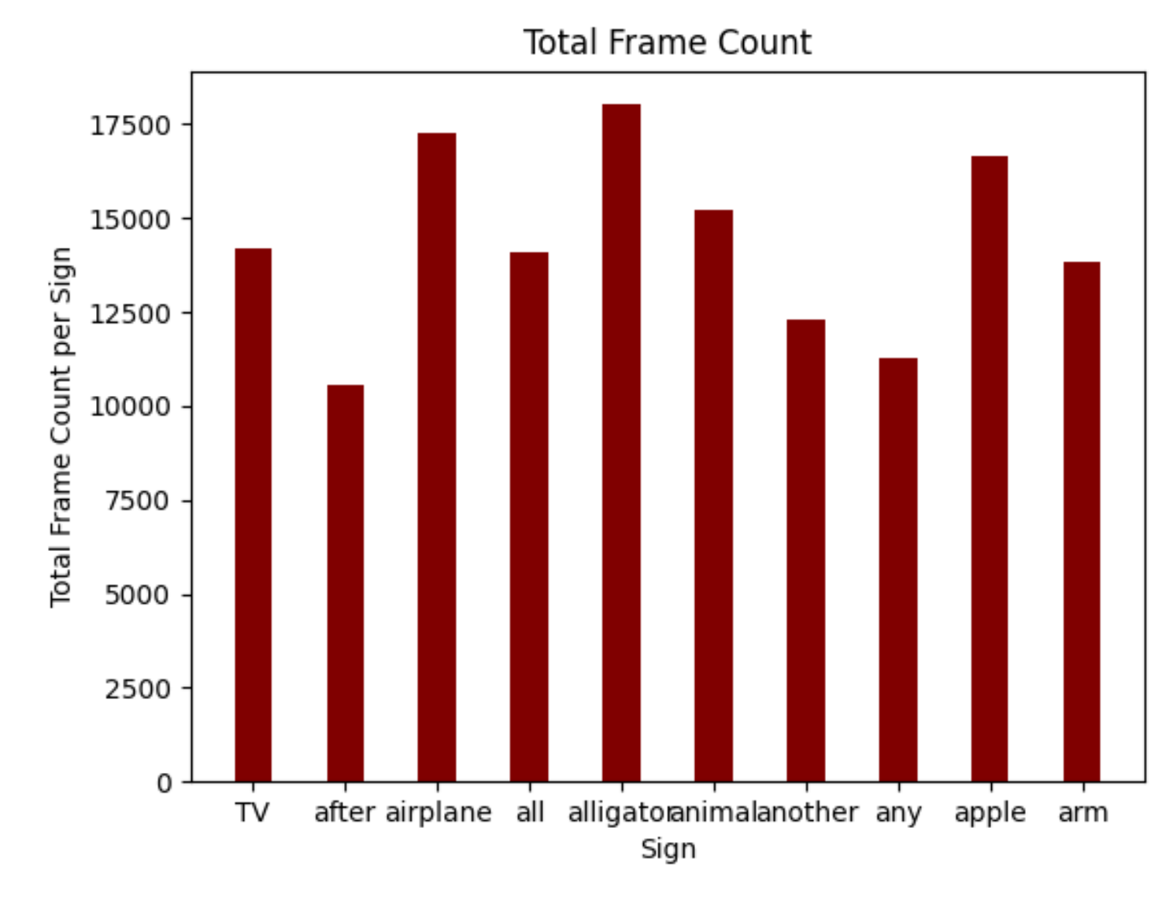


Figure 7. Trainig Data Distribution

Future work

- **Transformer** LSTM is slow due to the internal for loop. Exploring Transformer Architecture with Encoder-Decoder would be interesting.
- **CNN + LSTM** Combining CNN with LSTM to potentially improve accuracy
- **Include other landmark features**: Using Pose and Facial Landmarks can also improve the classification accuracy.
- **Preprocessing**: Upsampling the class with fewer training data.

References

[1] Kaggle Competition.
Google - isolated sign language recognition.

[2] Sepp Hochreiter and Jürgen Schmidhuber.
Long short-term memory.
Neural computation, 9(8):1735–1780, 1997.

[3] Google MediaPipe.
Hand landmarks detection guide.

[4] Mosquera C. Nápoles Van Houdt, G.
A review on the long short-term memory model.
2020.