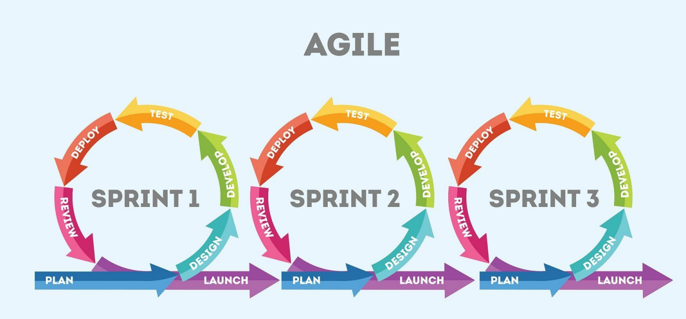
Odyssey - ASL Letter/Word Detection

Project Lead: Tony Gonzalez

Project Vision

- The primary objective is to develop a machine learning model that can translate American Sign Language (ASL) hand signs into words and sentences
- This project aimed to teach and develop the following:
    - A Strong Foundation in Machine Learning
    - Computer Vision Techniques
    - Advanced Model Strategies
    - Effective Teamwork Skills
    - Agile Development Practices

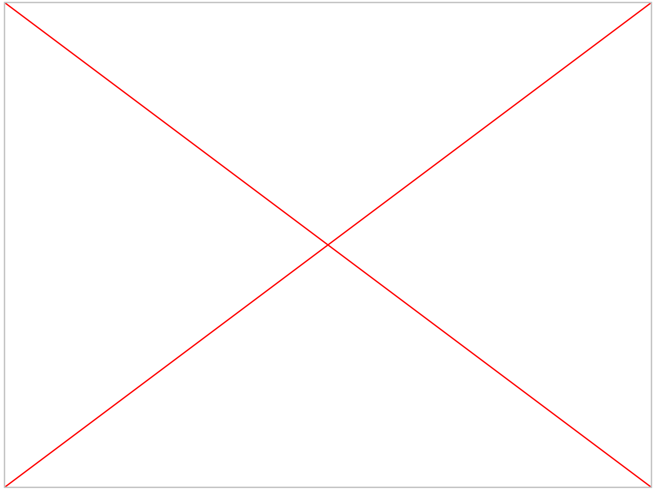
Project Timeline



- Sprint 1: Project Kickoff and Initial Research
- Sprint 2: Dataset Preparation and Data Preprocessing
- Sprint 3: Model Selection and Baseline Implementation
- Sprint 4: Model Tuning and Feature Engineering
- Sprint 5: Integration with MediaPipe and Real-Time Testing
- Sprint 6: Model Fusion and Advanced Techniques (if needed)
- Sprint 7: Finalize Prototype and Prepare for Demonstration

Training Dataset - Videos

Book

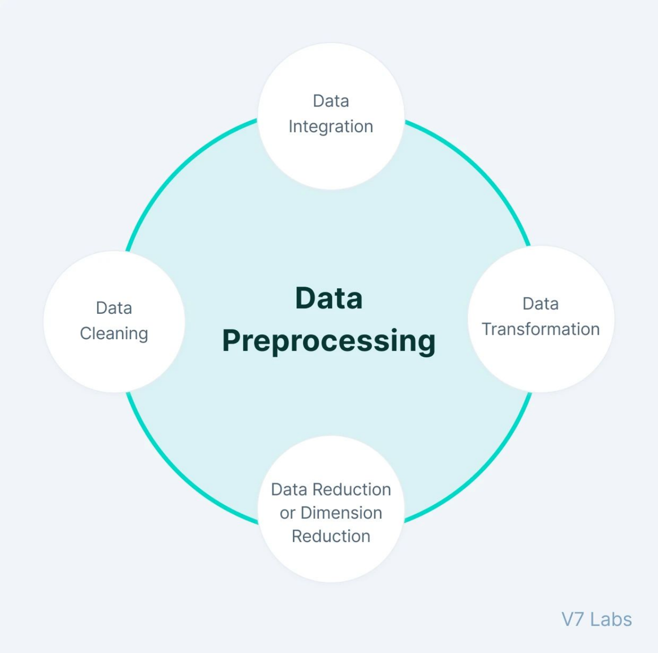


Computer

Source: WLASL (World Level American Sign Language) - Kaggle

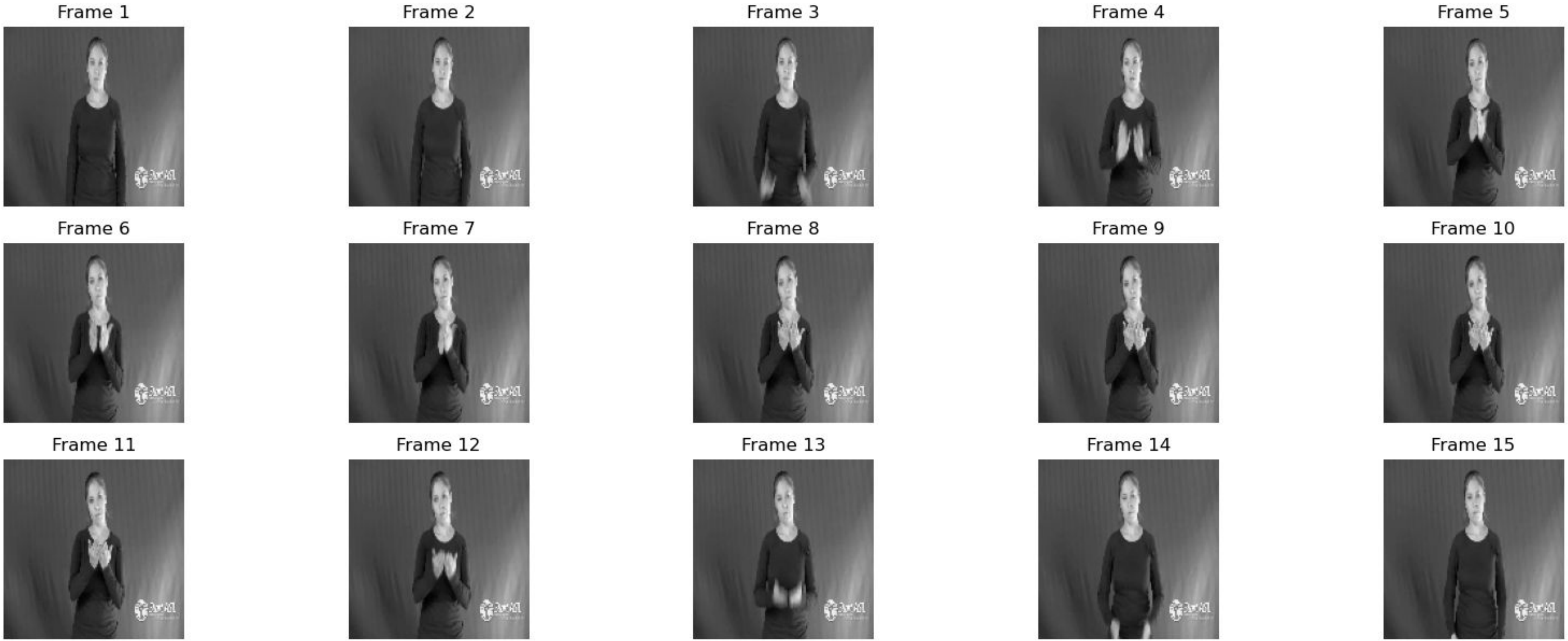
Training Dataset - Preprocessing

- Extracted a fixed number of frames per video (ensuring a consistent input size for the model)
- Resized frames to a consistent resolution
- Normalized pixel values (scaling from [0, 255] to [0, 1] for better neural network performance)



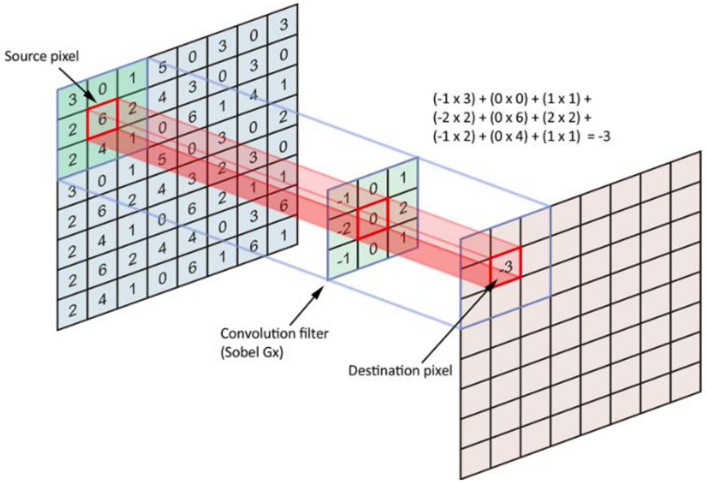
V7 Labs

Training Dataset - Preprocessing



Model Architecture - Convolutional Neural Network

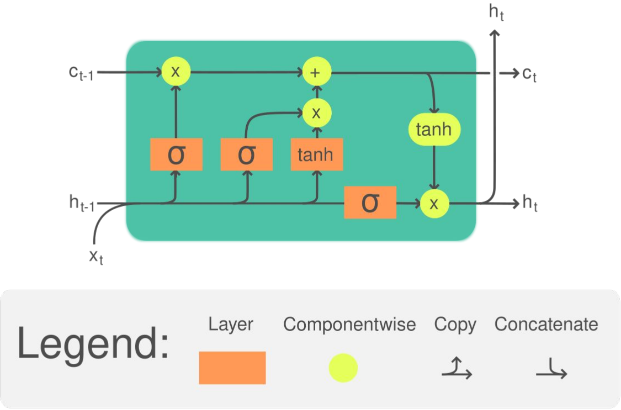
- A type of deep learning model designed to process and analyze image data
- Work by convoluting (sliding) a kernel matrix over an image’s pixels to compute a numerical representation (feature map)
- Different filters detect different features, such as edges, corners, or textures
- Convolutional Neural Networks are often superior for image recognition tasks



(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3

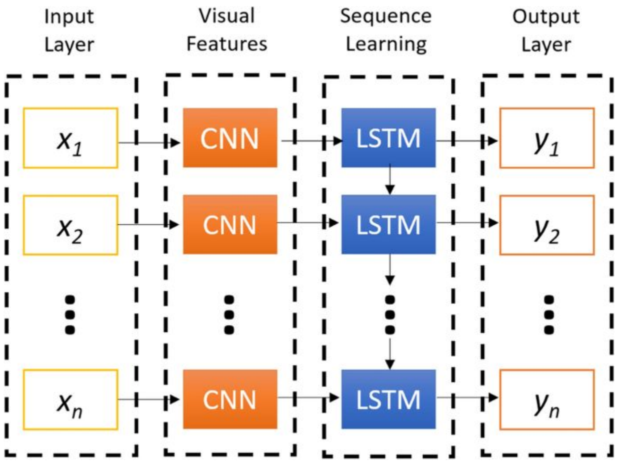
Model Architecture - Long-Short Term Memory (LSTM)

- LSTM (Long Short-Term Memory) is a type of Recurrent Neural Network (RNN) designed to process sequential data
- ASL gestures involve a sequence of movements over time, making LSTMs ideal for recognizing patterns in video frames
- Unlike CNNs, which focus on spatial features, LSTMs track movement transitions across frames



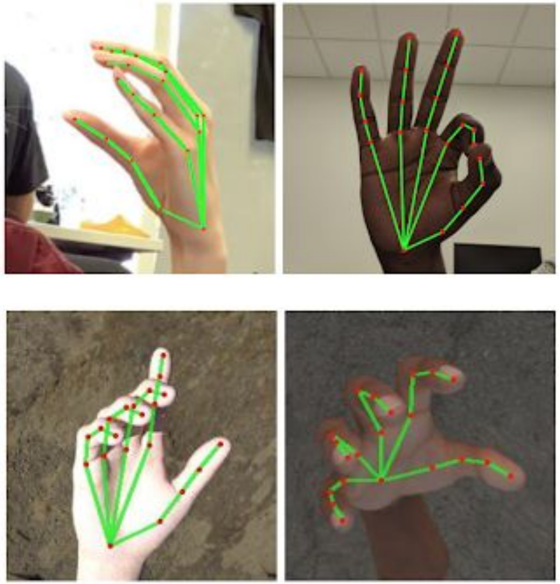
Model Architecture - CNN-LSTM Combination

- CNN is great for extracting spatial features from images, such as hand shapes and finger positions
- LSTM specializes in learning temporal patterns from sequential data, such as how hand movements evolve over time
- Together, they create a powerful hybrid model that can recognize both spatial (shape, position) and temporal (motion, sequence) features in ASL gestures



Live Video Integration and Implementation Details

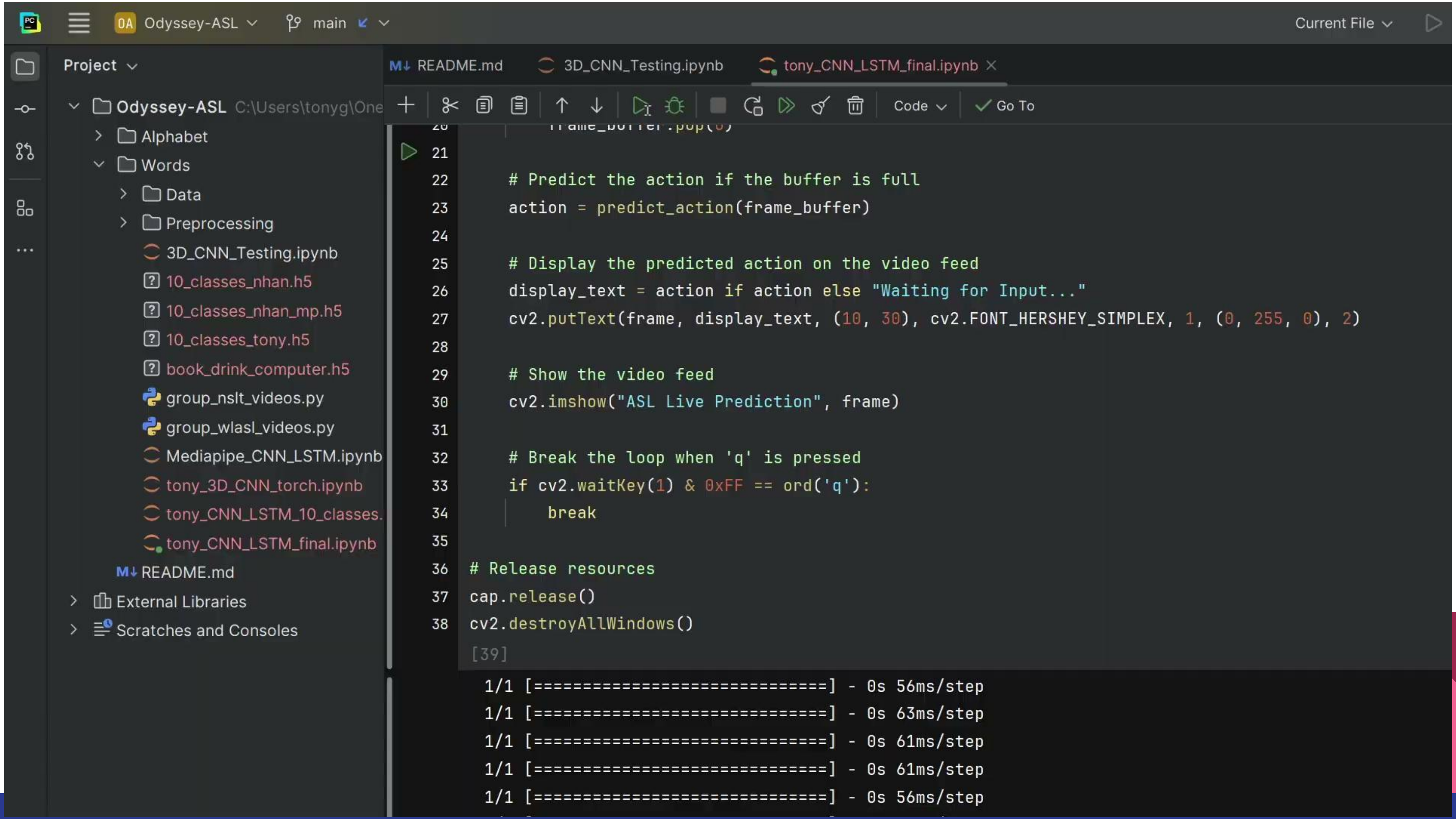
- Constructed the CNN-LSTM model using TensorFlow/Keras
- Used OpenCV to access the webcam in real-time, continuously reading frames from the video stream for processing
- Passed the processed frame sequences through the trained CNN-LSTM model
- Displayed predictions live on-screen using OpenCV



Challenges and Solutions

- Challenge: Limited labeled video data made training harder and increased risk of overfitting.
  - Solution: Used data augmentation (flipping, rotation, brightness adjustments) to artificially expand the dataset
- Challenge: Model performed well on training data but struggled with unseen real-world samples.
  - Solution: Replicated training data conditions (clear, blank background, solid colored clothing, solely upper torso, and similar signing speed)
- Challenge: A deeper model improves accuracy but slows down inference for real-time translation.
  - Solution: Found an optimal trade-off by using batch normalization, reducing LSTM units, and experimenting with fewer convolutional layers

Program Demonstration



```
        frame_buffer.pop(0)

    # Predict the action if the buffer is full
    action = predict_action(frame_buffer)

    # Display the predicted action on the video feed
    display_text = action if action else "Waiting for Input..."
    cv2.putText(frame, display_text, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

    # Show the video feed
    cv2.imshow("ASL Live Prediction", frame)

    # Break the loop when 'q' is pressed
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release resources
cap.release()
cv2.destroyAllWindows()
```

[39]

```
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 56ms/step
```