TransLingualVisionary

Team E6
Kavish Purani, Neeraj Ramesh,
Sandra Serbu

Use Case

Problem:

- Difficult for deaf or hard of hearing (HOH) individuals to participate in live digital environments (online meetings, live streams, etc.)
- Lack of widespread understanding of American Sign Language (ASL)
- Often require assistance from translators to communicate

Solution:

- A real-time ASL speech to English text translator on a user friendly web application

Our Solution

TransLingualVisionary (or TLV) is an ASL-to-Text translator that includes:

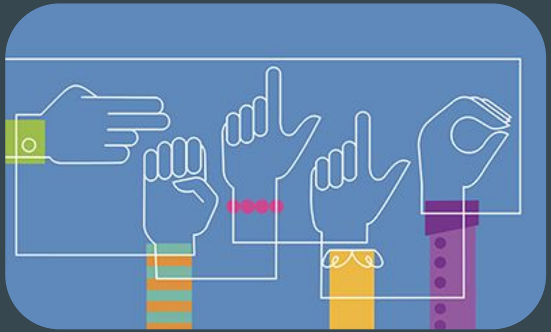
- Live translation of ASL to text
- Accelerated FPGA pre- and post- image processing
- User-friendly web app to visualize processed ASL input and text output

TLV will allow ASL users to:

- Quickly communicate to non-ASL users
- Document their speech in a simple and efficient manner

ECE Focus Areas:

- Software Systems
- Hardware Systems



Use-Case Requirements

Requirement	Metric
Recognize when a user is signing	No output when there is no ASL or user present
Correctly identify ASL words	Recognize 2000 words at ~75% accuracy
Correctly interpret ASL semantics	Translate identified clusters of words into full english sentences with a BLEU score of ~40%

Use-Case Requirements

Requirement	Metric
Classification Distance	Recognize and retain accuracy of the classification model up to 4-5 feet away from the camera.
Text Accessibility	Display and collect the ASL Speech in an accessible user format that can be easily found and read.
Overall Latency ~ real time	Present visual feed and translation on web UI within ~3 seconds

Technical Challenges

ASL Interpretation

- Identify questions, ends of sentences, and other expression and grammar rules inASL
- Variability in gesture speeds and length of words/phrases

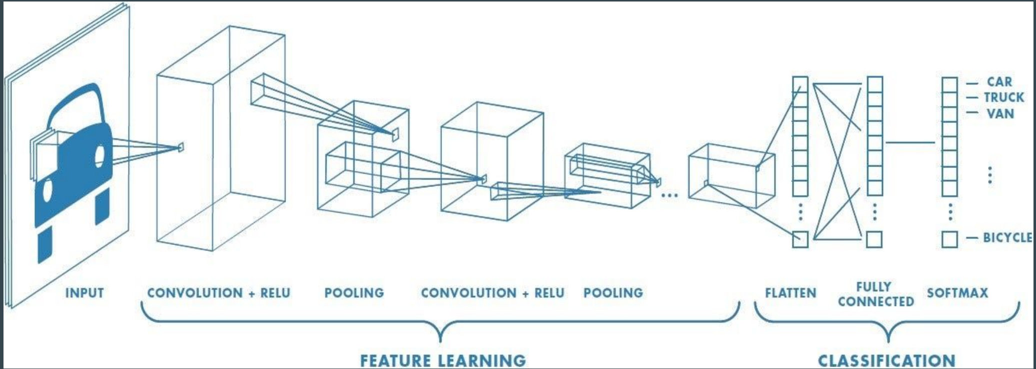
Training the models

- Accounting for overfitting considering the amount of data that we have
- Making sure that we are not training on extraneous details in our training set

Model Inference

- Has to be efficient in order to minimize delay, but also accurate enough to obtain the correct output from the LLM.

Technical Challenges



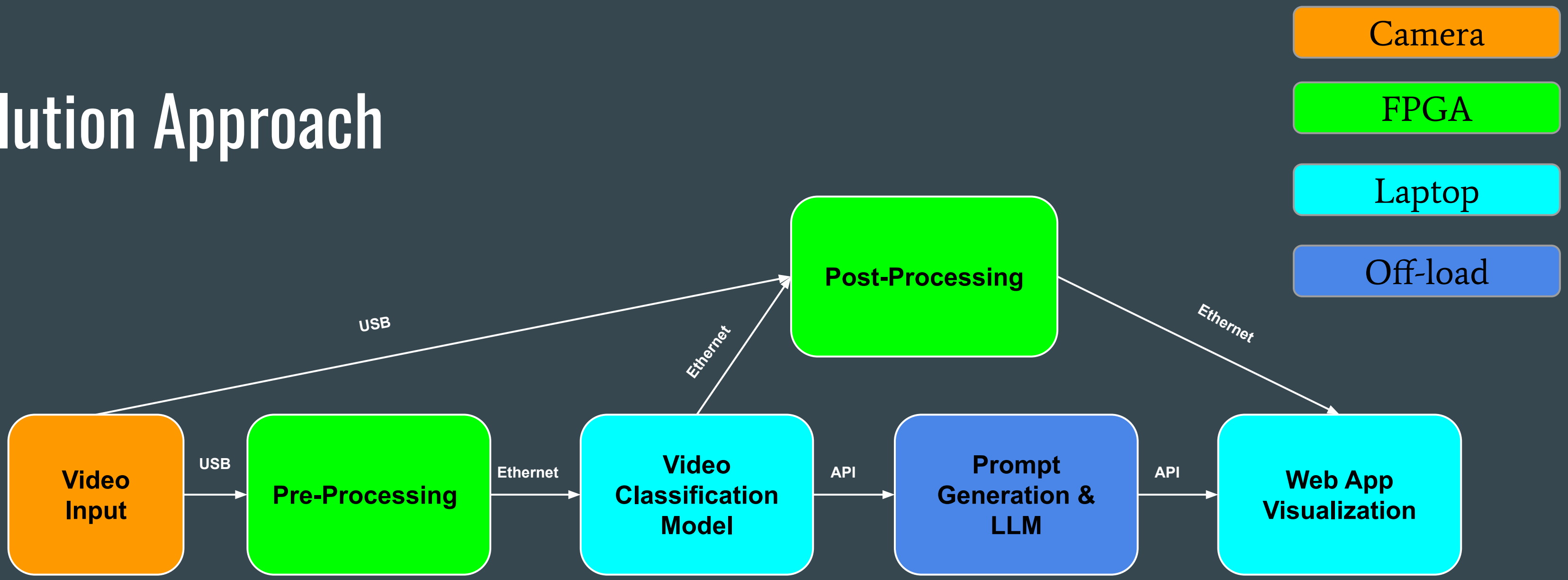
Determining the CNN/RNN layers needed to accurately classify ASL words

Utilizing video feed frames as useful inputs

Hardware:

- Ramp-up FPGA to process video stream from camera
- Maintain low latency to not be a bottleneck on the pipeline
- Maintain high frame rate under memory and communication bandwidth constraints`

Solution Approach



- Capturing live video stream from camera and processing it on the FPGA
    - Frame Extraction, Resizing, Normalization, etc.
- Training a CNN-RNN model for classification of signs into words/phrases
- Prompt generation to utilize LLM for sentence reconstruction and error correction
- Viewing the live text translation and processed video stream on web application

Testing, Verification, and Metrics

ASL Recognition What text output is given when non-ASL gestures/no gestures are occurring.

ASL Identification Calculate the classification error rate of gestures and their word output.

Interpret ASL to English Calculate the overall sentence translation accuracy of the LLM using BLEU scores.

Latency Record the time between when a sign is made to when the text displays.

Text Accessibility User satisfaction feedback survey

Classification Distance Calculate translation accuracy at various distances

BLEU Score	Interpretation
< 10	Almost useless
10 - 19	Hard to get the gist
20 - 29	The gist is clear, but has significant grammatical errors
30 - 40	Understandable to good translations
40 - 50	High quality translations
50 - 60	Very high quality, adequate, and fluent translations
> 60	Quality often better than human

Universal metric for evaluating machine-translated text.

https://cloud.google.com/translate/automl/docs/evaluate

Testing, Verification, and Metrics

Overall Design Verification

Unit testing components within our pipeline to verify individual latency requirements

Classification Verification

Parameter Tweaking: Use accuracy metrics of validation sets to optimize parameters

Hardware Verification

Method Correctness: Make sure that each method obtains the correct values

Number of Cycles per Operation: Determine if further optimization is needed per operation

Tasks and Division of Labor

	Kavish	Neeraj	Sandra
FPGA pre/post processing	✖		
Classification Model	✖	✖	✖
Prompt Generation & LLM		✖	
Web Application			✖
Testing & Integration	✖	✖	✖

Schedule

- Sandra, Kavish, Neeraj
- Kavish
- Sandra, Neeraj

	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13
	2/5	2/12	2/19	2/26	3/11	3/18	3/25	4/1	4/8	4/15
Presentation										
Proposal	SKN									
Design Review				SKN						
Final										SKN
Hardware										
FPGA Ramp-up	K									
Camera I/O		K	K							
Single Image Pre-Processing		K								
Testing and Debugging Pre-Processing			K							
Real-time Pre-Processing				K						
Testing and Debugging Pre-Processing					K					
Single Image Post-Processing						K				
Testing and Debugging Post-Processing							K			
Real-time Post-Processing								K		
Testing and Debugging Post-Processing										
Integration					SKN			SKN		
Software										
Find and test datasets	SN									
Pre-processing datasets for training		SN								
Developing RNN-CNN model			SN							
Training and testing				SN						
Prompt Engineering LLM		SN	SN	SN						
Testing LLM					SN					
Integration						SN				
Simple Web App										
Developement							SN			
Testing								SN		
Final Integration										
Testing									SKN	SKN
Slack									SKN	SKN