SmartSign — Real-Time Sign Language Translator

One-page step-by-step project summary

SmartSign — Real-Time Sign Language Translator One-page step-by-step project summary

1) Project Goal (short) Build a low-latency system that converts live camera video of sign language into readable text (and optional speech). Start with isolated-word recognition and extend to continuous sentence-level translation.

2) High-level Architecture • Input: Live RGB camera feed. • Preprocessing: MediaPipe (hands + pose + face) → keypoints; optional hand-crop RGB. • Temporal Model: Keypoint sequence → Temporal ConvNet / Bi-LSTM / Transformer. • Decoder: Softmax for isolated; CTC or Seq2Seq for continuous. • UI: Overlay captions + confidence, optional TTS.

3) Datasets • WLASL (word-level ASL), ASLLVD, RWTH-PHOENIX-Weather (continuous), custom recordings for local signs.

4) Data Collection & Annotation • 720–1080p @30–60 FPS; record torso + hands. • Label clips (word-level) or segments (continuous) using CVAT/LabelStudio. • Metadata: signer ID, lighting, camera angle.

5) Preprocessing • Extract frames at 15–30 FPS. • Run MediaPipe to get 2D/3D keypoints; normalize by shoulder width and center on torso. • Optionally crop hands and extract CNN features. • Augment: horizontal flip (careful with handedness), jitter, temporal stretch.

6) Feature Design • Keypoint-only (fast): sequence of normalized coordinates. • Appearance + keypoint (accurate): CNN embeddings fused with keypoints.

7) Model Choices • Backbone: MobileNetV3 / ResNet-18 for hand crops. • Temporal: TCN, Bi-LSTM, or Transformer. • Loss: Cross-entropy (isolated); CTC or cross-entropy Seq2Seq (continuous).

8) Training Strategy • Start with keypoint-only isolated model. • Use pretrained CNNs, mixed-precision, gradient clipping. • Hold-out signer split for generalization. • Metrics: Top-1/Top-5 (isolated), WER & BLEU (continuous).

9) Real-time Inference Pipeline • Capture frame → MediaPipe keypoints → buffer window (SEQ_LEN) → model inference → smoothing/voting → overlay text & TTS. • Optimize: ONNX → TensorRT / TFLite / CoreML, quantize (INT8) if needed.

10) Deployment • Edge: TFLite (mobile/RPi) for keypoint models. • Web: TensorFlow.js + MediaPipe Web. • Server: ONNX + GPU (TensorRT).

11) UI & UX Features • Live captions, confidence bar, show keypoints, save transcripts, correction mechanism for active learning.

12) Human-in-the-loop & Ethics • Involve native signers, obtain consent for recordings. • Clear privacy policy (faces in video). • Use as assistive tool — do not replace professional interpreters in critical contexts.

13) Milestones (iterative) 1. Prototype: keypoint-only isolated classifier (CLI demo). 2. Realtime demo with MediaPipe + model overlay. 3. Add appearance fusion for accuracy. 4. Continuous SLT with CTC/Seq2Seq. 5. Mobile/web deployment and user testing.

Tools & Libraries: MediaPipe, OpenPose; PyTorch or TensorFlow; CVAT/LabelStudio; ONNX, TensorRT, TFLite; OpenCV.

Contact: Harsh Tripathi — use corrections to collect labeled data and improve signer-specific accuracy.