Sign-to-Voice: A Deep Learning Approach to Translating ASL Fingerspelling into Audible Speech

Date: November 4, 2025

Team 7

1. Project Title and Team Members

Project Title: Sign-to-Voice: A Deep Learning Approach to Translating ASL Fingerspelling into Audible Speech

Team Members: Vinod Kumar Kumaravel, Vishal Balaji, Eesha Reddy Alluri

2. Project Description

Deaf or Hard-of-Hearing (DHH) individuals face significant communication barriers with non-signers when interpreters are unavailable. This project aims to develop a practical real-time system that translates American Sign Language (ASL) alphabet gestures into spoken English, bridging communication gaps. The core objective is to build a foundational "Sign-to-Speech Speller" enabling users to spell words letter by letter using hand signs, leveraging deep learning techniques for visual recognition of static ASL alphabet signs and integrating cloud-based text-to-speech services to provide audible output, fostering greater inclusion and demonstrating multimodal AI pipelines connecting computer vision input to speech synthesis output.

3. URL to the Dataset

Dataset: Sign Language MNIST dataset (https://www.kaggle.com/datasets/datamunge/sign-language-mnist). 28×28 pixel grayscale images, 24 classes (A-Y, excluding J and Z), ~27,455 training images, ~7,172 test images.

4. Tentative Deep Learning Algorithms and Techniques

The project will implement a comprehensive deep learning approach utilizing Convolutional Neural Networks (CNNs) and cloud-based AI services. The core recognition algorithm will be a custom lightweight CNN designed for 28×28×1 grayscale image classification of ASL alphabet signs, implemented using TensorFlow/Keras. The model architecture consists of multiple Conv2D layers (32, 64 filters) with ReLU activation, MaxPooling2D for spatial downsampling, Dropout layers for regularization, and Dense layers with softmax activation for 24-class probability distribution. Training strategy employs Categorical Cross-entropy loss, Adam optimizer, L2 regularization, early stopping, and extensive data augmentation (rotations, shifts, zooms, brightness adjustments). The system will utilize Google Cloud Vertex AI for scalable serverless model deployment and Google Cloud Text-to-Speech API for speech synthesis. Local client technologies include Python with OpenCV for webcam capture and image preprocessing, and Google Cloud SDK for API communication. Domain adaptation will be achieved through fine-tuning with webcam-captured samples to bridge the gap between clean dataset and real-world input.

5. Project Timeline and Milestones

The project is planned over a 4 week period, structured into six distinct phases:

Phase 1 (Weeks 1): Environment Setup and Data Preparation - Google Colab and GCP configuration, Sign Language MNIST dataset acquisition, data preprocessing, and augmentation implementation.

Phase 2 (Weeks 1-2): Model Development and Training - Custom lightweight CNN architecture design and implementation in TensorFlow/Keras, training pipeline establishment with regularization (Dropout, L2), early stopping, and hyperparameter tuning to achieve target ≥95% accuracy on the Kaggle test set.

Phase 3 (Weeks 2-3): Cloud Deployment and Integration - Trained model export and upload to Google Cloud Storage, deployment as serverless endpoint on Vertex AI, Google Cloud Text-to-Speech API configuration, and end-to-end cloud pipeline testing.

Phase 4 (Weeks 3): Local Client Development - Python client development for webcam capture and image preprocessing, communication with Vertex AI endpoint, audio playback system integration, and basic user interface implementation.

Phase 5 (Weeks 3-4): Testing and Evaluation - End-to-end latency optimization to ≤2 seconds, live classification accuracy measurement (target ≥85%) through controlled testing, and system integration testing with user experience feedback.

Phase 6 (Weeks 4): Documentation and Finalization - Comprehensive technical and project documentation, final report preparation, presentation materials, demonstration video, and reproducibility package (code repository, setup guide).

6. Task Allocation of Team Members

Tasks are allocated based on expertise and project phase requirements:

Vinod: Phase 2 (Model Architecture Design & Optimization, Training Pipeline & Hyperparameter Tuning, Offline Model Evaluation), Phase 3 (Vertex AI Deployment, API Integration Testing, End-to-End Cloud Pipeline Integration), Phase 5 (Performance Testing and System Optimization), Phase 6 (Final Report Preparation, Future Work Planning).

Vishal: Phase 1 (Google Colab Setup, Google Cloud Platform Setup, Dataset Download), Phase 3 (Google Cloud Storage Setup, Google Cloud TTS Setup), Phase 4 (Basic UI Implementation), Phase 6 (Code Documentation).

Eesha: Phase 1 (Local Development Environment, Data Preprocessing Pipeline, Data Augmentation Strategy, Baseline Analysis), Phase 3 (Model Export and Formatting), Phase 4 (Webcam Integration, Image Preprocessing Pipeline, Cloud Communication, Audio Playback System, User Experience Features), Phase 5 (System Integration Testing, User Acceptance Testing), Phase 6 (System Documentation, Reproducibility Package).

Success Criteria

The project will be deemed successful if it meets the following criteria: offline performance of ≥95% accuracy on the Kaggle ASL Alphabet test set, live performance of ≥85% accuracy for real-time webcam testing, end-to-end latency of ≤2 seconds, functional prototype usable by DHH community members, and complete system implementable using free cloud resources.