Speech AI Suite - Complete Project Overview

Project Summary

Speech AI Suite is a professional, production-ready multi-task speech analysis platform that performs four critical audio classification tasks using state-of-the-art self-supervised learning models. This project combines cutting-edge deep learning with practical web application design, achieving 79.14% accuracy on emotion recognition and serving as a comprehensive reference for speech representation learning.

Research Paper Foundation:

- "*From Raw Speech to Fixed Representations: A Comprehensive Evaluation of Speech Embedding Techniques*" (IEEE/ACM 2024)

Four Core Tasks

Task	Model	Classifier	Accuracy	Dataset	Classes
Emotion Classification	HuBERT-large	SVM	79.14%	CREMA-D	6 (Neutral, Happy, Sad, Angry, Fear, Disgust)
Gender Identification	WavLM-base-plus	Logistic Regression	—	Mixed	2 (Male, Female)
Intent Classification	WavLM-base-plus	SVM	—	SLURP	20 (Voice Commands)
Speaker Identification	XLSR-53	Logistic Regression	—	Mixed	Variable (Speaker IDs)

System Architecture

```
Speech AI Suite
├── Frontend (Bootstrap 5 + HTML/CSS/JS)
│   ├── Emotion Classification Page
│   ├── Gender Identification Page
│   ├── Intent Classification Page
│   ├── Speaker Identification Page
│   └── About Page (Project Documentation)
│
├── Backend (Flask - Python Web Framework)
│   ├── Audio Input Handler
│   ├── Inference Services (4 parallel modules)
│   └── REST API Endpoints
│
└── ML Pipeline (PyTorch + scikit-learn)
    ├── Data Preprocessing
    ├── Feature Extraction (Self-supervised Models)
```

```
    ├── Model Training & Evaluation
    └── Inference Modules
```

🔄 Complete Data Processing Pipeline

Stage 1: Data Preprocessing

Input: Raw audio files (WAV, MP3, FLAC, OGG, M4A, WebM) **Process:**

1. Audio loading and normalization
2. Resampling to 16kHz (standard for speech models)
3. Duration filtering (remove too short/long samples)
4. Label mapping and standardization
5. Data split: Train (70%), Validation (15%), Test (15%)

Output: Preprocessed dataset metadata (CSV format)

Stage 2: Feature Extraction

Input: Preprocessed audio files **Process:**

1. Load pre-trained self-supervised model (HuBERT, WavLM, or XLSR-53)
2. Extract final hidden layer representations
3. Apply pooling strategy (mean, std, or concatenation)
4. Normalize embeddings using StandardScaler

Mathematical Operation:

```
embedding = Model(audio) → hidden_states[last_layer]
pooled_embedding = Pool(hidden_states)
normalized_embedding = StandardScaler.transform(pooled_embedding)
```

Output: Fixed-dimensional embeddings (1024-2048 dimensions)

Stage 3: Dimensionality Reduction

Input: High-dimensional embeddings **Process:**

1. Apply PCA to reduce to 200 components
2. Fit scaler on training data
3. Transform validation/test data

Mathematical Operation:

```
X_reduced = PCA(n_components=200).fit_transform(X_embeddings)
```

Output: 200-dimensional vectors

Stage 4: Model Training

Input: Reduced embeddings + labels **Process (varies by task):**

- **SVM:** Multi-class SVM with RBF kernel, hyperparameter tuning via GridSearchCV
- **Logistic Regression:** L2 regularization, balanced class weights
- **Cross-validation:** 5-fold CV for robust performance estimation

Output: Serialized model + scaler + label encoder (`.pkl` files)

Stage 5: Inference

Input: New audio file (user upload/recording) **Process:**

1. Load audio with audio processing library
2. Extract embedding using same model as training
3. Scale embedding using training scaler
4. Predict using trained classifier
5. Get probability scores using `predict_proba()`

Output: Predicted label + confidence scores (probabilities)

🔬 Deep Learning Models Used

1. HuBERT-large (Emotion Classification)

- **Full Name:** Hidden-Unit BERT for Self-supervised Speech Representation
- **Publisher:** Meta AI (Facebook)
- **Architecture:** Transformer-based with 24 layers, 1024 hidden dimensions
- **Output Dimension:** 1024-dimensional embeddings
- **Pre-training:** Masked prediction on 960 hours of Libri-Light unlabeled data
- **Why Used:** Best performance for emotion classification in our experiments

2. WavLM-base-plus (Gender & Intent)

- **Full Name:** Large-Scale Self-Supervised Pre-Training for Full Stack Speech Processing
- **Publisher:** Microsoft Research
- **Architecture:** 12 transformer layers, 768 hidden dimensions
- **Output Dimension:** 768-dimensional embeddings
- **Pre-training:** 10,000 hours of speech with masked prediction
- **Why Used:** Efficient, fast inference while maintaining good accuracy

3. XLSR-53 (Speaker Identification)

- **Full Name:** Wav2Vec 2.0 XLSR (Cross-Lingual Speech Representations)
- **Publisher:** Meta AI
- **Architecture:** 12 transformer layers, 1024 hidden dimensions
- **Output Dimension:** 1024-dimensional embeddings

- **Pre-training:** 56,000 hours of multilingual speech
- **Why Used:** Robust speaker identification across languages and accents

---

📐 Mathematical Framework

Feature Extraction Formula

```
E = PoolingStrategy(HiddenStates[-1])

Where:
- E = Output embedding (fixed-dimension vector)
- HiddenStates[-1] = Last transformer layer output (sequence of vectors)
- PoolingStrategy ∈ {mean, max, std, concatenation}

Mean Pooling: E = mean(HiddenStates[-1])
Std Pooling: E = std(HiddenStates[-1])
Concatenation: E = concatenate(mean, std) → 2x dimensions
```

Classification Formula

For SVM:

```
y_pred = sign(Σ α_i * K(x, x_i) + b)

Where:
- K = RBF kernel: K(x,x') = exp(-γ||x-x'||²)
- γ (gamma) = 1/n_features
- C (regularization) = 1.0
```

For Logistic Regression:

```
P(y=class_i) = exp(w_i · x + b_i) / Σ exp(w_j · x + b_j)

Where:
- w_i = weights for class i
- b_i = bias for class i
- x = input embedding (scaled)
```

Cross-Validation Strategy

```
For each of k=5 folds:
  1. Split data into train (80%) and validation (20%)
  2. Train classifier on train fold
  3. Evaluate on validation fold
```

```
    4. Record metrics

  Final_Accuracy = mean(fold_accuracies)
```

💾 Data Formats & Datasets

Input Audio Formats Supported

- **WAV:** Uncompressed PCM, 16-bit
- **MP3:** Compressed, variable bitrate
- **FLAC:** Lossless compression
- **OGG/Opus:** Open-source compression
- **M4A:** Apple audio format
- **WebM:** Browser-native recording format

Datasets Used

CREMA-D (Emotion Classification)

- **Size:** ~7,500 utterances
- **Duration:** ~47 hours total
- **Speakers:** 91 unique speakers
- **Emotions:** 6 (Neutral, Happy, Sad, Angry, Fear, Disgust)
- **Language:** English
- **Recording:** Studio quality, multiple takes per emotion
- **Format:** WAV files at 16kHz

SLURP (Intent Classification)

- **Size:** ~63,000 utterances
- **Task:** Spoken Language Understanding
- **Intents:** 20+ voice command categories
- **Domains:** Smart home, entertainment, productivity
- **Language:** English
- **Format:** Audio files at 16kHz

🛠️ Technology Stack

Backend Framework

- **Flask:** Lightweight Python web framework
- **Werkzeug:** WSGI utilities for request handling
- **Python 3.10+:** Runtime environment

Deep Learning Libraries

- **PyTorch:** Tensor operations, model inference

- **Transformers (HuggingFace):** Pre-trained models loading
- **torchaudio:** Audio processing and resampling
- **librosa:** Audio feature computation
- **soundfile:** Audio file I/O

Machine Learning & Data Science

- **scikit-learn:** SVM, Logistic Regression, PCA, StandardScaler
- **numpy:** Numerical computations
- **pandas:** Data manipulation (CSV handling)
- **scipy:** Scientific computing utilities

Frontend

- **Bootstrap 5:** Responsive UI framework
- **HTML5:** Semantic markup
- **CSS3:** Styling with gradients and animations
- **JavaScript (Vanilla):** Audio recording/playback, form handling
- **Bootstrap Icons:** SVG icon library

Development & Deployment

- **Git:** Version control
- **Git LFS:** Large file storage (for .pkl models)
- **GitHub:** Repository hosting
- **Virtual Environment:** Python dependency isolation

🚀 Project Development Workflow

1. Data Preparation Phase

```
Raw Audio Files (CREMA-D, SLURP)
         ↓
   1_data_preprocessing.py
         ↓
Normalized CSV Metadata
```

2. Feature Extraction Phase

```
Audio Files + Preprocessing Metadata
         ↓
   2_wavlm_feature_extraction.py
         ↓
NumPy Arrays (.npz files)
Embedding Dimensions: 1024 or 768
```

3. Model Training Phase

```
Extracted Embeddings + Labels
         ↓
   3_train_classifiers.py
         ↓
Trained Models (.pkl files)
- SVM/LogReg classifier
- StandardScaler
- LabelEncoder
```

4. Evaluation Phase

```
Trained Models + Test Set
         ↓
   4_evaluation_metrics.py
         ↓
Metrics (Accuracy, F1, Precision, Recall)
Confusion Matrices
```

5. Deployment Phase

```
Trained Models (.pkl)
         ↓
Backend Flask Server
         ↓
REST API Endpoints
         ↓
Web UI (HTML/CSS/JS)
```

Project Structure

```
Emotion-classification/
├── backend/                          # Flask web application
│   ├── app/
│   │   ├── app.py                   # Main Flask application (11 routes)
│   │   ├── templates/               # HTML templates (Jinja2)
│   │   │   ├── base.html           # Base template with navbar
│   │   │   ├── emotion.html        # Emotion classification page
│   │   │   ├── gender.html         # Gender identification page
│   │   │   ├── intent.html         # Intent classification page
│   │   │   ├── speaker.html        # Speaker identification page
│   │   │   ├── about.html          # Project documentation
│   │   │   └── index.html          # Homepage
```

```
│   │   └── static/
│   │       ├── css/
│   │       │   ├── styles.css       # Main stylesheet (550+ lines)
│   │       │   └── custom.css
│   │       ├── js/                  # JavaScript files
│   │       └── images/              # Logo, icons
│   ├── services/                     # Inference services
│   │   ├── emotion.py               # Emotion classification (126 lines)
│   │   ├── gender.py                # Gender identification
│   │   ├── intent.py                # Intent classification
│   │   ├── speaker.py               # Speaker identification
│   │   └── utils/
│   │       └── audio.py             # Audio loading & feature extraction
│   └── config.py                     # Configuration (paths, models)
│
├── ml_models/                        # ML training & artifacts
│   ├── models/                       # Pre-trained models (15 .pkl files, 600+
MB)
│   │   ├── emotion_model_svm.pkl     # SVM classifier (143.78 MB)
│   │   ├── emotion_scaler.pkl        # StandardScaler
│   │   ├── emotion_label_encoder.pkl # LabelEncoder
│   │   ├── gender_classifier.pkl     # Logistic Regression
│   │   ├── intent_classifier.pkl     # SVM classifier (16.28 MB)
│   │   └── [More model files...]
│   ├── src/                          # Training scripts
│   │   ├── 1_data_preprocessing.py   # Data loading & cleaning
│   │   ├── 2_wavlm_feature_extraction.py  # Embedding extraction
│   │   ├── 3_train_classifiers.py    # Model training
│   │   ├── 4_evaluation_metrics.py   # Model evaluation
│   │   └── 5_visualization_umap.py   # t-SNE/UMAP visualization
│   ├── data/                         # Raw datasets
│   │   ├── CREMA-D/                 # Emotion dataset
│   │   ├── IEMOCAP/                 # Alternative emotion dataset
│   │   └── processed/                # CSV metadata
│   ├── results/                      # Training results
│   │   ├── confusion_matrix_*.csv    # Classification matrices
│   │   ├── evaluation_results_*.json # Metrics
│   │   └── test_predictions.csv      # Predictions on test set
│   └── scripts/                      # Utility scripts
│       ├── train_gender_model.py
│       ├── train_intent_model.py
│       ├── verify_setup.py
│       └── download_samples.py
│
├── docs/                             # Documentation
│   ├── PROJECT_OVERVIEW.md           # This file (complete overview)
│   ├── EMOTION_CLASSIFICATION.md     # Emotion task details
│   ├── GENDER_IDENTIFICATION.md      # Gender task details
│   ├── INTENT_CLASSIFICATION.md      # Intent task details
│   ├── SPEAKER_IDENTIFICATION.md     # Speaker task details
│   ├── ARCHITECTURE.md               # System architecture
│   ├── SETUP_GUIDE.md                # Installation guide
│   └── QUICKSTART.md                 # Quick start guide
│
```

```
├── .gitattributes                      # Git LFS configuration
├── requirements.txt                    # Python dependencies (40+ packages)
├── README.md                           # Quick project description
└── Emotion-classification.code-workspace  # VS Code workspace
```

👥 Development Team

Name	Role	Expertise	Contribution
Sk Inthiyaz	ML Engineer & Project Lead	Architecture, Integration	Project design, backend/frontend integration, system architecture
Romith Singh	ML Engineer & Team Lead	Speaker Recognition	Speaker identification model, optimization techniques
Rohin Kumar	ML Engineer	Gender Classification	Gender identification, feature engineering
Sahasra Ganji	ML Engineer & Data Engineer	Intent Recognition	Intent classification, dataset preparation, data engineering
Rashmitha	ML Engineer & Research Engineer	Research, Bug Fixing	Bug resolution, documentation, research support

---

🎓 Key Concepts Explained

Self-Supervised Learning

- Models are pre-trained on **massive unlabeled speech data** (10,000-56,000 hours)
- Learn general speech representations without manual annotation
- Fine-tuning layers frozen; use embeddings as features
- Dramatically reduces annotation cost for downstream tasks

Transfer Learning

- Pre-trained models → Extract embeddings
- Embeddings fed to simple classifiers (SVM, Logistic Regression)
- Avoids training deep models from scratch
- Leverages knowledge from large-scale pre-training

Dimensionality Reduction (PCA)

- Embeddings are 768-1024 dimensional (very high)
- PCA reduces to 200 dimensions
- Removes noise while preserving variance
- Reduces memory footprint and inference time

Cross-Validation

- Protects against overfitting
- Uses 5 folds for robust performance estimation
- Reports mean accuracy across all folds
- 79.14% = average across 5 folds for emotion

✨ Key Features

1. **Real-time Audio Processing**
    - Record directly from browser microphone
    - Upload pre-recorded files
    - Support for 6+ audio formats
2. **Multi-Model Support**
    - 4 different self-supervised models
    - Mix-and-match for different tasks
    - Easy to add new models
3. **High Accuracy**
    - Emotion: 79.14% on CREMA-D
    - Leverages state-of-the-art SSL models
    - 5-fold cross-validation for reliability
4. **Production Ready**
    - Proper error handling and logging
    - Git LFS for large model files
    - Modular architecture for maintenance
    - Comprehensive documentation
5. **Optimized for CPU**
    - No GPU requirement
    - Fast inference (2-5 seconds per audio)
    - Suitable for deployment on modest hardware
6. **Beautiful UI**
    - Responsive design (works on mobile/tablet)
    - Gradient backgrounds with animations
    - Real-time audio visualization
    - Professional styling

🔍 Model Comparison

Why Different Models for Different Tasks?

Task	Model	Reason
Emotion	HuBERT-large	Large model needed for fine-grained emotion distinction (6 classes)
Gender	WavLM-base-plus	Simple binary classification; faster inference sufficient
Intent	WavLM-base-plus	Medium complexity (20 classes); good speed/accuracy tradeoff
Speaker	XLSR-53	Multilingual robustness; handles diverse speaker accents

---

🎯 Interview Confidence Points

When presenting this project in interviews, emphasize:

1. **Research Foundation**
    - Grounded in IEEE/ACM 2024 research paper
    - Evaluated 3 SOTA self-supervised models
    - Published methodology
2. **End-to-End Pipeline**
    - Data preprocessing → Feature extraction → Training → Inference
    - Each stage documented with mathematical formulas
    - Reproducible with provided scripts
3. **Production Considerations**
    - Git LFS for managing large model files
    - Error handling for unsupported formats
    - Configurable batch processing for CPU
    - Comprehensive logging and monitoring
4. **Architecture Decisions**
    - Why each model was chosen
    - Trade-offs between accuracy and speed
    - Dimensionality reduction rationale
    - Cross-validation strategy
5. **Reproducibility**
    - All hyperparameters documented
    - Training scripts with fixed random seeds
    - Version control of all code and data
6. **Scalability**
    - Can add new tasks (follow 4-stage pipeline)
    - Modular service design
    - Easy to deploy on cloud (AWS, Azure, GCP)

Recommended Reading Order

1. **Start Here:** This file (PROJECT_OVERVIEW.md)
2. **Then:** EMOTION_CLASSIFICATION.md (most developed task)
3. **Next:** GENDER_IDENTIFICATION.md, INTENT_CLASSIFICATION.md, SPEAKER_IDENTIFICATION.md
4. **Finally:** ARCHITECTURE.md (system design details)

Getting Started

Installation

```
# Clone repository
git clone https://github.com/sk-inthiyaz/Emotion-classification.git
cd Emotion-classification

# Create virtual environment
python -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate

# Install dependencies
pip install -r requirements.txt

# Download models (via Git LFS)
git lfs pull

# Run Flask app
cd backend/app
python app.py
```

Access the Application

- **URL:** http://localhost:5000
- **Emotion Classification:** http://localhost:5000/emotion
- **Gender Identification:** http://localhost:5000/gender
- **Intent Classification:** http://localhost:5000/intent
- **Speaker Identification:** http://localhost:5000/speaker
- **About/Documentation:** http://localhost:5000/about

Performance Summary

Metric	Value
Emotion Accuracy	79.14% (5-fold CV on CREMA-D)
Emotion F1-Score	0.78 (macro average)
Inference Time	2-5 seconds (CPU, depending on audio length)

Metric	Value
Model Download Size	~600 MB (all .pkl files via Git LFS)
Memory Usage	~2 GB (during feature extraction)
Supported Languages	English (main), multilingual support via XLSR-53

License & Attribution

- **Research Paper:** IEEE/ACM 2024
- **Models:** HuggingFace (open-source pre-trained models)
- **Datasets:** CREMA-D, SLURP (academic use)
- **Code:** Open-source, GitHub repository

Related Links

- **HuBERT Paper:** https://arxiv.org/abs/2106.07447
- **WavLM Paper:** https://arxiv.org/abs/2110.13900
- **XLSR Paper:** https://arxiv.org/abs/2006.13979
- **CREMA-D Dataset:** https://github.com/CheyneyComputerScience/CREMA-D
- **SLURP Dataset:** https://github.com/parietal-io/slurp
- **HuggingFace Models:** https://huggingface.co

Project Completion Status

- ☑ Data preprocessing pipeline
- ☑ Feature extraction (3 models)
- ☑ Model training & evaluation
- ☑ Inference service (4 tasks)
- ☑ Web UI with Bootstrap
- ☑ Documentation & README
- ☑ Git LFS for large files
- ☑ Production-ready code

Created: December 2024 **Last Updated:** December 11, 2025 **Repository:** https://github.com/sk-inthiyaz/Emotion-classification

This documentation serves as a comprehensive reference for understanding, implementing, and extending the Speech AI Suite project.