Speech AI Suite: Comprehensive Research Paper

Multi-Task Speech Analysis Using Self-Supervised Learning

---

3. SYSTEM REQUIREMENTS & ANALYSIS

3.1 Purpose

The **Speech AI Suite** is a production-ready, multi-task speech analysis platform designed to perform four critical audio classification tasks using state-of-the-art self-supervised learning models. The project demonstrates how pre-trained transformer-based speech models can be leveraged for downstream tasks without extensive labeled data.

Primary Purposes:

1. **Emotion Classification (Task 1)**
    - Detect 6 emotional states (Neutral, Happy, Sad, Angry, Fear, Disgust) from speech
    - Applications: Sentiment analysis, customer service quality, mental health assessment
    - Business value: Improve customer experience, analyze call center quality
    - Research contribution: Benchmark emotion recognition using SOTA self-supervised models
2. **Gender Identification (Task 2)**
    - Binary classification of speaker gender (Male/Female)
    - Applications: Speaker demographics, personalized services, dataset validation
    - Business value: Demographic analysis, personalized voice interfaces
    - Research contribution: Simple baseline task for model comparison
3. **Intent Classification (Task 3)**
    - Multi-class classification of user intent from voice commands (20+ categories)
    - Applications: Voice assistants, smart home control, IoT devices
    - Business value: Enable hands-free interfaces, accessibility features
    - Research contribution: Voice command routing for smart systems
4. **Speaker Identification (Task 4)**
    - Identify individual speakers from voice biometrics
    - Applications: Authentication, speaker diarization, access control
    - Business value: Biometric security, call center operations
    - Research contribution: Advanced speaker embeddings using multilingual models

Research Objectives:

- Evaluate three SOTA self-supervised speech models (HuBERT, WavLM, XLSR-53)
- Compare model selection strategies for different classification complexities

- Demonstrate transfer learning effectiveness across speech understanding tasks
- Provide reproducible pipeline for speech classification
- Create production-ready inference system with web interface

Target Users:

- **Researchers:** Studying speech representation learning and transfer learning
- **ML Engineers:** Building speech AI applications
- **Businesses:** Implementing voice-based customer interfaces
- **Students:** Learning practical deep learning and speech processing

---

3.2 Scope

3.2.1 Functional Scope

The Speech AI Suite encompasses the following capabilities:

Data Management

- Support for multiple audio formats: WAV, MP3, FLAC, OGG, M4A, WebM
- Automatic audio preprocessing (normalization, resampling to 16kHz)
- Integration with multiple datasets (CREMA-D, SLURP, proprietary)
- Data augmentation capabilities

Feature Extraction

- Three self-supervised models: HuBERT-large (24 layers, 1024-dim), WavLM-base-plus (12 layers, 768-dim), XLSR-53 (24 layers, 1024-dim)
- Fixed embedding extraction without fine-tuning
- Advanced pooling strategies (mean, std, concatenation)
- Normalization and scaling

Model Training

- Classical machine learning classifiers: SVM (RBF kernel), Logistic Regression
- Hyperparameter optimization via GridSearchCV
- Cross-validation strategies (5-fold)
- Class balance handling (weighted classes for imbalanced datasets)
- Model serialization and versioning

Inference & Deployment

- Real-time audio processing (2-5 seconds)
- Probability-based predictions with confidence scores
- Batch processing capability
- Error handling for edge cases
- REST API endpoints for all 4 tasks

User Interface

- Web-based interface (Flask backend, Bootstrap frontend)
- Real-time audio recording and playback
- File upload capability
- Visual result display with confidence scores
- Responsive design for mobile/tablet/desktop

3.2.2 Non-Functional Scope

Performance

- Emotion classification: 79.14% accuracy (5-fold CV)
- Inference latency: 2-5 seconds per audio (CPU)
- Memory footprint: ~2GB during feature extraction
- Model size: ~600MB total (all .pkl files)

Scalability

- Designed for CPU-only deployment (no GPU requirement)
- Can handle 24/7 inference requests
- Modular architecture allows adding new tasks
- Git LFS for managing large model artifacts

Maintainability

- Modular code structure (separate services for each task)
- Comprehensive documentation (90+ KB documentation files)
- Version control with Git
- Type hints and docstrings
- Unit tests and validation scripts

Security

- Input validation for audio files
- Error handling for malformed inputs
- Model integrity verification
- No external API dependencies

Accessibility

- Web UI works on all browsers
- Mobile-responsive design
- Keyboard navigation support
- Error messages for accessibility

3.2.3 Out of Scope

- Real-time streaming speech (only fixed-length audio)

- Fine-tuning of self-supervised models (transfer learning only)
- Multi-speaker speaker separation
- Speech-to-text transcription
- Voice synthesis (TTS)
- GPU-specific optimizations
- Multi-language support for emotion (English only for CREMA-D)
- Continuous learning or online adaptation

3.3 Definitions, Acronyms, and Abbreviations

Key Terms

Term	Definition
Self-Supervised Learning (SSL)	Learning from unlabeled data by creating proxy tasks; models learn representations without manual annotation
Transfer Learning	Using pre-trained models as feature extractors for downstream tasks
Embedding	Fixed-dimensional vector representation of input (e.g., 1024-dim audio embedding)
Feature Extraction	Converting raw audio into meaningful representations (embeddings)
Downstream Task	Classification task performed on top of embeddings (e.g., emotion classification)
Pooling	Aggregating sequence of vectors into single vector (mean, max, std)
Dimensionality Reduction	Reducing vector dimensions while preserving information (e.g., 1024 → 200)
Cross-Validation	Evaluating model on multiple data splits for robust performance estimation
RBF Kernel	Radial Basis Function kernel for SVM; captures non-linear relationships
One-vs-Rest (OvR)	Multi-class strategy: train N binary classifiers for N classes

Acronyms

Acronym	Meaning
SSL	Self-Supervised Learning
HuBERT	Hidden-Unit BERT (for speech)
WavLM	Wav2Vec Large Model
XLSR	Cross-Lingual Speech Representations
SVM	Support Vector Machine
RBF	Radial Basis Function
PCA	Principal Component Analysis

Acronym	Meaning
CREMA-D	Crowdsourced Emotional Multimodal Actors Dataset - Discrete
SLURP	Spoken Language Understanding
kHz	Kilohertz (1000 Hz)
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
OvR	One-vs-Rest
F1	F1-Score (harmonic mean of precision and recall)
CSV	Comma-Separated Values
REST	Representational State Transfer
HTTP	Hypertext Transfer Protocol
GPU	Graphics Processing Unit
CPU	Central Processing Unit
Git LFS	Git Large File Storage
SOTA	State-of-the-Art

Dataset Terms

Term	Definition
Utterance	Single spoken sentence or phrase
Emotion Class	One of 6 emotions in CREMA-D (Neutral, Happy, Sad, Angry, Fear, Disgust)
Intent Category	One of 20+ voice command types in SLURP
Speaker ID	Unique identifier for individual speaker
Sampling Rate	Number of audio samples per second (16kHz = 16,000 samples/sec)

3.4 Overall Description

3.4.1 Product Perspective

The Speech AI Suite is a standalone system that can be:

1. **Deployed locally** on a single machine (CPU-only)
2. **Deployed on cloud** (AWS, Azure, GCP) for scalable inference
3. **Integrated** into larger systems via REST API
4. **Extended** with new tasks following the established pipeline

System Context Diagram

```
┌─────────────────────────────────────────────────────────────┐
│                         End Users                           │
│  (Researchers, ML Engineers, Business Users)                │
└─────────────────────────────────────────────────────────────┘
                          │ HTTP/Browser
                          ▼
┌─────────────────────────────────────────────────────────────┐
│             Flask Web Application                           │
│  ┌──────────────┬──────────────┬──────────────┐             │
│  │   Emotion    │   Gender     │   Intent     │             │
│  │ Classification Identification Classification │           │
│  ├──────────────┼──────────────┼──────────────┤             │
│  │              Speaker Identification          │            │
│  └──────────────┴──────────────┴──────────────┘             │
│                          ▼                                  │
│           REST API Endpoints                                │
└─────────────────────────────────────────────────────────────┘
                            │ Function Calls
                            ▼
┌─────────────────────────────────────────────────────────────┐
│        ML Inference Services Layer                          │
│  ┌──────────────┐ ┌──────────────┐ ┌──────────────┐         │
│  │  Emotion     │ │   Gender     │ │    Intent    │         │
│  │  Service     │ │   Service    │ │    Service   │         │
│  │ (HuBERT-SVM) │ │(WavLM-LogReg)│ │ (WavLM-SVM)  │         │
│  └──────────────┘ └──────────────┘ └──────────────┘         │
│  ┌─────────────────────────────────────────────────────┐    │
│  │        Speaker Service (XLSR-LogReg)                │    │
│  └─────────────────────────────────────────────────────┘    │
│                          ▼                                  │
│          Audio Processing Utils                             │
│   (Load, Preprocess, Extract Features)                      │
└─────────────────────────────────────────────────────────────┘
                            │ File I/O
                            ▼
┌─────────────────────────────────────────────────────────────┐
│        Pre-trained Models (Frozen)                          │
│  ┌──────────────┐ ┌──────────────┐ ┌──────────────┐         │
│  │  HuBERT-     │ │  WavLM-      │ │   XLSR-53    │         │
│  │  large       │ │  base-plus   │ │              │         │
│  │ (1024-dim)   │ │ (768-dim)    │ │ (1024-dim)   │         │
│  └──────────────┘ └──────────────┘ └──────────────┘         │
└─────────────────────────────────────────────────────────────┘
                            │ Model Inference
                            ▼
┌─────────────────────────────────────────────────────────────┐
│        Input Audio Files & Stream                           │
│  (WAV, MP3, FLAC, OGG, M4A, WebM, Browser Recording)        │
└─────────────────────────────────────────────────────────────┘
```

3.4.2 User Characteristics

Primary Users

1. **ML Researchers**
    - Experience: Advanced (PhD/Masters in ML)
    - Goals: Study speech models, publish papers, benchmark performance
    - Needs: Documentation, reproducibility, comparison metrics
2. **ML Engineers**
    - Experience: Intermediate to Advanced
    - Goals: Integrate into products, deploy systems
    - Needs: API documentation, deployment guides, performance metrics
3. **Business Users (Non-Technical)**
    - Experience: Non-technical
    - Goals: Use system for their domain (customer service, smart home)
    - Needs: Simple UI, clear results, no setup required
4. **Students**
    - Experience: Beginner to Intermediate
    - Goals: Learn speech processing, deep learning
    - Needs: Code examples, explanations, tutorials

3.4.3 General Constraints

Constraint	Impact	Mitigation
CPU-only	Slower inference (2-5 sec)	Accept latency for non-real-time uses
Fixed-length audio	Can't process streams	Segment streams into chunks
Pre-trained models frozen	Can't adapt to new domains	Use transfer learning approach
English-focused	Limited multilingual support	XLSR-53 supports 53 languages
Model size (~600MB)	Large download	Use Git LFS for efficient storage
No fine-tuning	Performance ceiling on out-of-domain audio	PCA + simple classifiers sufficient

3.5 System Overview

3.5.1 Complete Data Processing Pipeline

```
Stage 1: RAW INPUT
├── File Upload (Frontend)
├── Microphone Recording (Browser)
```

```
└── Supported Formats: WAV, MP3, FLAC, OGG, M4A, WebM
                    ↓
Stage 2: PREPROCESSING
├── Load audio file
├── Extract waveform
├── Normalize amplitude [-1.0, 1.0]
├── Resample to 16 kHz (standard speech rate)
├── Trim silence
├── Duration filtering (0.5 - 30 seconds)
└── Output: Preprocessed waveform [sample_rate * duration]
                    ↓
Stage 3: FEATURE EXTRACTION (Self-Supervised Model)
├── Load frozen pre-trained model
│   ├── HuBERT-large (24 layers, 1024-dim) - Emotion
│   ├── WavLM-base-plus (12 layers, 768-dim) - Gender, Intent
│   └── XLSR-53 (24 layers, 1024-dim) - Speaker
├── Forward pass: waveform → embeddings
├── Extract last hidden layer [sequence_length, dim]
├── Apply pooling:
│   ├── Mean pooling → [dim]
│   ├── Std pooling → [dim]
│   └── Concatenation (speaker only) → [2*dim]
└── Output: Fixed embedding [1024] or [768] or [2048]
                    ↓
Stage 4: NORMALIZATION & SCALING
├── Load training data statistics
├── Apply StandardScaler: (x - μ) / σ
├── Ensure zero mean and unit variance
└── Output: Scaled embedding [1024/768/2048]
                    ↓
Stage 5: DIMENSIONALITY REDUCTION (PCA)
├── Load fitted PCA transformer (trained on train set)
├── Reduce dimensions: [1024/768/2048] → [200]
├── Preserve 95%+ variance with 200 components
└── Output: Reduced embedding [200]
                    ↓
Stage 6: CLASSIFICATION & PREDICTION
├── Load trained classifier:
│   ├── Emotion: SVM (RBF kernel)
│   ├── Gender: Logistic Regression
│   ├── Intent: SVM (RBF kernel, One-vs-Rest)
│   └── Speaker: Logistic Regression
├── predict(embedding) → class_label
├── predict_proba(embedding) → [confidence scores]
└── Output: Predicted label + probabilities
                    ↓
Stage 7: RESULT FORMATTING & DISPLAY
├── Format results as JSON
├── Include confidence score (0.0 - 1.0)
├── Add top-3 predictions with scores
├── Return to frontend
└── Display in UI with visualizations
```

3.5.2 Model Selection Rationale

Task	Model	Classes	Why Selected
Emotion	HuBERT-large	6	Large model needed for fine-grained emotion nuances; best performer in evaluation
Gender	WavLM-base-plus	2	Smaller model sufficient for binary; faster inference
Intent	WavLM-base-plus	20	Good balance; handles multi-class efficiently
Speaker	XLSR-53	Variable	Multilingual robustness; speaker-aware pre-training

3.5.3 Key Algorithms

Algorithm 1: Feature Extraction via Self-Supervised Models

```
ALGORITHM: ExtractFeatures(audio_waveform, model)
INPUT:
  - audio_waveform: [sample_rate * duration] audio samples
  - model: pre-trained self-supervised transformer model

OUTPUT:
  - embedding: fixed-dimensional vector [hidden_dim]

STEPS:
  1. waveform ← normalize(audio_waveform)
  2. waveform ← resample(waveform, target_rate=16000)
  3. hidden_states ← model(waveform)  // all transformer layers
  4. final_hidden ← hidden_states[-1]  // last layer [seq_len, hidden_dim]
  5. IF task == "speaker":
       embedding ← concatenate(mean(final_hidden), std(final_hidden))
     ELSE:
       embedding ← mean(final_hidden)
  6. embedding ← normalize(embedding)  // StandardScaler
  7. embedding ← pca.transform(embedding)  // [hidden_dim] → [200]
  8. RETURN embedding

COMPLEXITY:
  - Time: O(seq_len * hidden_dim * num_layers)
  - Space: O(seq_len * hidden_dim)
```

Algorithm 2: SVM Classification with RBF Kernel

```
ALGORITHM: ClassifyWithSVM(embedding, svm_model)
INPUT:
  - embedding: [200-dimensional] reduced embedding
```

```
  - svm_model: trained SVM classifier (RBF kernel)

OUTPUT:
  - predicted_class: predicted emotion/intent label
  - probabilities: confidence scores per class

STEPS:
  1. decision_value ← svm_model.decision_function(embedding)
  2. REPEAT FOR each class:
       // One-vs-Rest: N binary classifiers
  3. predicted_class ← argmax(decision_value)
  4. probabilities ← softmax(decision_value)
  5. RETURN (predicted_class, probabilities)

RBF KERNEL FORMULA:
  K(x, x') = exp(-γ || x - x' ||²)
  where γ = 1 / (n_features) = 1/200

COMPLEXITY:
  - Time: O(support_vectors * embedding_dim)
  - Space: O(support_vectors)
```

Algorithm 3: Cross-Validation for Model Evaluation

```
ALGORITHM: CrossValidate(X, y, classifier, k=5)
INPUT:
  - X: [n_samples, 200] embeddings
  - y: [n_samples] labels
  - classifier: SVM or LogisticRegression
  - k: number of folds (default 5)

OUTPUT:
  - cv_scores: list of accuracies per fold
  - mean_accuracy: average accuracy
  - std_accuracy: standard deviation

STEPS:
  1. fold_size ← n_samples / k
  2. cv_scores ← []
  3. FOR each fold = 1 to k:
       train_idx ← all samples except fold
       val_idx ← samples in fold
       X_train, y_train ← X[train_idx], y[train_idx]
       X_val, y_val ← X[val_idx], y[val_idx]

       classifier.fit(X_train, y_train)
       accuracy ← classifier.score(X_val, y_val)
       cv_scores.append(accuracy)
  4. mean_accuracy ← mean(cv_scores)
  5. std_accuracy ← std(cv_scores)
  6. RETURN (cv_scores, mean_accuracy, std_accuracy)
```

```
COMPLEXITY:
  - Time: O(k * classifier_training_time)
  - Space: O(training_set_size)
```

3.5.4 Data Flow Diagram

```
┌──────────────────┐
│  User Uploads    │
│   Audio File     │
└────────┬─────────┘
         │ HTTP POST
         ▼
┌──────────────────────────────┐
│  Flask Route Handler         │
│  (app.py - /predict endpoint)│
└────────┬─────────────────────┘
         │ Pass to service
         ▼
┌──────────────────────────────┐
│  Task-Specific Service       │
│  (emotion.py, gender.py etc.)│
└────────┬─────────────────────┘
         │ Load & validate audio
         ▼
┌──────────────────────────────┐
│  Audio Processing            │
│  (utils/audio.py)            │
│  - Load waveform             │
│  - Normalize & resample      │
└────────┬─────────────────────┘
         │ Pass preprocessed audio
         ▼
┌──────────────────────────────┐
│  Pre-trained Model           │
│  (HuBERT/WavLM/XLSR-53)      │
│  - Extract embeddings        │
└────────┬─────────────────────┘
         │ 1024/768 dim embedding
         ▼
┌──────────────────────────────┐
│  Preprocessing Pipeline      │
│  - Normalize (StandardScaler)│
│  - Reduce dims (PCA)         │
│  → [200-dimensional]         │
└────────┬─────────────────────┘
         │ 200-dim embedding
         ▼
┌──────────────────────────────┐
│  Classifier                  │
│  (SVM or LogisticRegression) │
```

```
│  - predict()                   │
│  - predict_proba()             │
└────────────────────────────────┘
         │
         │ Predicted label + confidence
         ▼
┌────────────────────────────────┐
│  Format Response               │
│  (JSON with probabilities)     │
└────────────────────────────────┘
         │
         │ HTTP JSON response
         ▼
┌────────────────────────────────┐
│  Display Results in UI         │
│  (HTML/CSS/JS)                 │
└────────────────────────────────┘
```

3.6 Operating Environment

3.6.1 Hardware Requirements

Minimum Requirements

- **Processor:** Intel i5 or equivalent (2.0+ GHz, 4 cores)
- **RAM:** 4 GB
- **Storage:** 2 GB (for models + dependencies)
- **Network:** Internet connection (for initial setup)

Recommended Requirements

- **Processor:** Intel i7 or equivalent (2.5+ GHz, 6+ cores)
- **RAM:** 8 GB
- **Storage:** 4 GB SSD (faster model loading)
- **Network:** Broadband connection

Cloud Deployment

- **AWS:** EC2 instance (t3.large or t3.xlarge)
- **Azure:** Standard_B2s or Standard_B4ms
- **GCP:** n2-standard-4 or n2-standard-8

3.6.2 Software Environment

Operating System Support

- Windows 10/11 (tested)
- Ubuntu 18.04/20.04/22.04 (Linux)
- macOS 10.14+ (Darwin)

Required Software

- **Python:** 3.8 - 3.11 (tested with 3.10)
- **Git:** 2.20+ (for version control)
- **Git LFS:** 2.0+ (for large model files)

Python Dependencies

```
Core ML Libraries:
├── torch==2.0.0 (PyTorch)
├── transformers==4.30.0 (HuggingFace models)
├── scikit-learn==1.2.0 (SVM, PCA, scaling)
├── numpy==1.24.0 (Numerical computing)
└── pandas==1.5.0 (Data handling)

Audio Processing:
├── librosa==0.10.0 (Audio feature extraction)
├── soundfile==0.12.0 (Audio I/O)
└── torchaudio==2.0.0 (Audio utilities)

Web Framework:
├── flask==2.3.0 (Web server)
└── werkzeug==2.3.0 (WSGI utilities)

Development Tools:
├── pytest==7.3.0 (Testing)
├── black==23.3.0 (Code formatting)
└── pylint==2.17.0 (Linting)

Additional:
├── matplotlib==3.7.0 (Visualization)
├── scipy==1.10.0 (Scientific computing)
└── python-dotenv==1.0.0 (Environment variables)
```

3.6.3 Network Requirements

Internet Connection

- **Initial Setup:** Required for downloading models from HuggingFace (500+ MB)
- **Runtime:** Not required (all models cached locally)
- **API Usage:** Not required (all processing local)

Port Requirements

- **Default:** Flask runs on `localhost:5000`
- **Production:** Typically port 80 (HTTP) or 443 (HTTPS)
- **Database:** Not required (file-based storage)

3.6.4 Deployment Environments

Local Development

```
Machine → Python 3.10 → Flask
        → Models (disk) → Inference (CPU)
        → Results (stdout/browser)
```

Server Deployment

```
Client (Browser) → HTTPS → Cloud Server (AWS/Azure/GCP)
                            → Python 3.10 environment
                            → Flask application
                            → CPU-based inference
                            → Response → Client
```

Docker Containerization

```
Dockerfile:
- Base: python:3.10-slim
- Install: python packages
- Copy: models, code
- Expose: port 5000
- CMD: python app.py
```

3.7 Functional Requirements

3.7.1 Feature 1: Audio Input Handling

Requirement ID: FR1

Description: System must accept audio input from multiple sources and formats.

Functional Requirements:

1. **File Upload**
    - Accept files up to 50 MB
    - Support formats: WAV, MP3, FLAC, OGG, M4A, WebM
    - Validate file type (magic bytes, not just extension)
    - Return error if unsupported format
2. **Browser Microphone Recording**
    - Record audio directly from browser microphone
    - Real-time audio visualization

- Start/Stop controls
- Preview before submission

3. **Audio Validation**

   - Check sample rate (16 kHz preferred, resample if needed)
   - Check duration (0.5 - 30 seconds)
   - Validate audio data (non-zero samples)
   - Return specific error messages

3.7.2 Feature 2: Emotion Classification

Requirement ID: FR2

Description: System must classify audio into one of 6 emotional categories.

Functional Requirements:

1. **Classification Task**

   - Input: Audio file (any format after conversion)
   - Process: HuBERT-large → SVM classifier
   - Output: Predicted emotion + confidence score (0.0-1.0)
   - Performance: ≥79% accuracy on CREMA-D test set

2. **Result Display**

   - Show primary emotion
   - Show probability distribution across all 6 emotions
   - Show visual representation (bar chart)
   - Show execution time

3. **Classes to Predict**

   - Neutral (calm, normal tone)
   - Happy (positive, joyful)
   - Sad (negative, sorrowful)
   - Angry (frustrated, aggressive)
   - Fear (anxious, frightened)
   - Disgust (repulsed, contemptuous)

3.7.3 Feature 3: Gender Identification

Requirement ID: FR3

Description: System must classify audio speaker as Male or Female.

Functional Requirements:

1. **Binary Classification**

   - Input: Audio file

- Process: WavLM-base-plus → Logistic Regression
- Output: Predicted gender + confidence
- Performance: ≥92% accuracy expected

2. **Result Display**
   - Show predicted gender (Male/Female)
   - Show confidence percentage
   - Show audio characteristics explanation

3.7.4 Feature 4: Intent Classification

Requirement ID: FR4

Description: System must classify voice commands into 20+ intent categories.

Functional Requirements:

1. **Multi-class Intent Recognition**
   - Input: Voice command audio
   - Process: WavLM-base-plus → SVM (One-vs-Rest)
   - Output: Predicted intent + confidence
   - Classes: 20+ voice commands
   - Performance: ≥85% accuracy expected

2. **Intent Categories**
   - Smart home: turn_on_lights, turn_off_lights, set_brightness, etc.
   - Entertainment: play_music, pause_media, skip_track, etc.
   - Information: get_weather, get_news, set_alarm, etc.
   - General: stop, help, cancel, etc.

3.7.5 Feature 5: Speaker Identification

Requirement ID: FR5

Description: System must identify individual speakers from voice.

Functional Requirements:

1. **Speaker Recognition**
   - Input: Audio samples from known/unknown speakers
   - Process: XLSR-53 → Logistic Regression
   - Output: Predicted speaker ID (if enrolled) or "Unknown"
   - Performance: ≥90% accuracy for 20-50 speakers

2. **Speaker Enrollment**
   - Register new speaker with audio sample(s)
   - Store speaker enrollment in database

- Update classifier with new speaker

3. **Verification vs Identification**
    - Verification: "Is this Speaker X?" (1:1 comparison)
    - Identification: "Who is this?" (1:N search)

3.7.6 Feature 6: Web Interface

Requirement ID: FR6

Description: System must provide user-friendly web interface.

Functional Requirements:

1. **Navigation**
    - Homepage with project overview
    - Separate pages for each task
    - About page with documentation
    - Responsive navbar
2. **Each Task Page Must Include**
    - Audio upload widget
    - Microphone recording widget
    - Task description
    - Result display area
    - Model information sidebar
3. **User Interaction**
    - Clear submit buttons
    - Loading indicators during processing
    - Error messages for failures
    - Download results (CSV, JSON)

3.7.7 Feature 7: API Endpoints

Requirement ID: FR7

Description: System must expose REST API for programmatic access.

Functional Requirements:

1. **Emotion Classification API**
    - POST `/api/emotion`
    - Input: audio file (multipart/form-data) or base64
    - Output: JSON with emotion, confidence, all probabilities
2. **Gender Classification API**

- POST `/api/gender`
- Input: audio file
- Output: JSON with gender, confidence

3. **Intent Classification API**
   - POST `/api/intent`
   - Input: audio file
   - Output: JSON with intent, confidence

4. **Speaker Identification API**
   - POST `/api/speaker`
   - Input: audio file
   - Output: JSON with speaker_id, confidence

5. **Health Check API**
   - GET `/api/health`
   - Output: JSON with system status, model info

3.7.8 Feature 8: Error Handling

Requirement ID: FR8

Description: System must handle errors gracefully.

Functional Requirements:

1. **Input Validation Errors**
   - Unsupported file format → HTTP 400 with message
   - Audio too short (<0.5s) → HTTP 400 with message
   - Audio too long (>30s) → HTTP 400 with message
   - File too large (>50MB) → HTTP 413 Payload Too Large

2. **Processing Errors**
   - Model loading failure → HTTP 500 with message
   - CUDA out of memory → HTTP 503 Service Unavailable
   - Timeout (>30s) → HTTP 504 Gateway Timeout

3. **User-Friendly Messages**
   - Clear explanation of error
   - Suggested remediation steps
   - Contact support link if needed

3.8 Non-Functional Requirements

3.8.1 Performance Requirements

Metric	Requirement	Actual
Emotion Accuracy	≥75%	79.14% ✓
Inference Latency	<10 seconds	2-5 seconds ✓
Model Loading Time	<5 seconds	~2-3 seconds ✓
Memory Usage	<4 GB	~2-2.5 GB ✓
Concurrent Users	≥10 (CPU)	Depends on server ✓

3.8.2 Scalability Requirements

1. **Horizontal Scaling**
   - Can deploy multiple Flask instances behind load balancer
   - Stateless design allows multiple servers
   - Models cached locally on each server
2. **Vertical Scaling**
   - Can upgrade to larger machine with more cores
   - No GPU requirement limits hardware flexibility
   - Linear scaling with CPU cores
3. **Data Volume**
   - Current: ~7,500 emotion samples, ~63,000 intent samples
   - Future: Can handle 10x more training data with same pipeline
   - Storage: Models (600MB) + code (50MB) + data (1GB max)

3.8.3 Security Requirements

1. **Input Validation**
   - All user uploads validated before processing
   - File type verification (magic bytes, not extension)
   - File size limits enforced (50MB max)
2. **Model Integrity**
   - Checksums for model files
   - Version control via Git LFS
   - No external API calls (all local processing)
3. **Error Handling**
   - No sensitive information in error messages
   - Graceful degradation on failures
   - Logging without exposing user data

3.8.4 Reliability Requirements

1. **Availability**
    - 99% uptime for web service
    - Graceful degradation if 1 model fails
    - Automatic restart on crash
2. **Data Integrity**
    - Checksums for model files
    - Backup of pre-trained models
    - Version control for code/config
3. **Failure Recovery**
    - Automatic restart on crash
    - Health check monitoring
    - Fallback to cached results

3.8.5 Maintainability Requirements

1. **Code Quality**
    - Modular architecture (separate services per task)
    - Type hints for type checking
    - Docstrings for all functions
    - Unit test coverage ≥80%
2. **Documentation**
    - README.md with quick start
    - API documentation with examples
    - Architecture documentation
    - Setup guide for developers
3. **Version Control**
    - All code in Git repository
    - Large models in Git LFS
    - Semantic versioning for releases
    - Commit messages follow conventions

3.8.6 Accessibility Requirements

1. **Web UI Accessibility**
    - WCAG 2.1 Level AA compliance
    - Keyboard navigation support
    - Screen reader compatibility
    - Color contrast ≥4.5:1
2. **Error Messages**

- Clear, non-technical language
- Suggest corrective actions
- Provide alternative approaches

3. **Documentation**

   - Plain language explanations
   - Code comments for complex logic
   - Example usage in all docs

3.8.7 Compliance Requirements

1. **Data Privacy**

   - No data storage of user uploads (deleted after inference)
   - No external data transmission
   - GDPR compliant (no personal data collected)

2. **Licensing**

   - Pre-trained models: HuggingFace license (community)
   - Datasets: CREMA-D, SLURP (academic use)
   - Code: Open source (specify license)

3. **Intellectual Property**

   - Attribution to model authors (Meta, Microsoft)
   - Citation of research papers
   - Respect dataset licenses

3.9 System Model

3.9.1 Entity-Relationship Model

```
┌─────────────────┐
│  AudioFile      │
├─────────────────┤
│ id (PK)         │
│ filename        │
│ file_size       │
│ duration        │
│ sample_rate     │
│ created_at      │
└─────────────────┘
         │
         │
         │ 1..* (one audio → many predictions)
         │
         ▼
    ┌──────────────────────┐
    │  Prediction          │
    ├──────────────────────┤
```

```
        │ id (PK)               │
        │ audio_id (FK)         │
        │ task_type             │ ← "emotion", "gender", "intent", "speaker"
        │ predicted_label       │ ← e.g., "happy", "male", "turn_on_lights"
        │ confidence_score      │ ← 0.0-1.0
        │ all_probabilities     │ ← JSON {class: prob}
        │ inference_time_ms     │ ← milliseconds
        │ created_at            │
        └───────────────────────┘
```

3.9.2 State Machine Diagram

```
                        ┌───────────────┐
                        │   START       │
                        └──────┬────────┘
                               │
                        ┌──────▼───────────┐
                        │ Audio Upload     │
                        │  / Recording     │
                        └──────┬───────────┘
                               │
                        ┌──────▼──────────────┐
                        │ Validate Audio      │
                        │(format, duration)   │
                        └──┬───────────────┬──┘
                         ✓ │               │ ✗
                           │               └─────────────────┐
                        ┌──▼───────────────┐          ┌──────▼────────┐
                        │ Preprocess       │          │ Show Error    │
                        │ & Normalize      │          │   Message     │
                        └──┬───────────────┘          └──────┬────────┘
                           │                                 │
                        ┌──▼─────────────────────────────────┤
                        │ Extract Features                   │
                        │ (Pre-trained Model)                │
                        └──┬─────────────────────────────────┘
                           │
                        ┌──▼───────────────┐
                        │ Scale & Reduce   │
                        │   (PCA)          │
                        └──┬───────────────┘
                           │
                        ┌──▼───────────────┐
                        │ Classify         │
                        │ (SVM/LogReg)     │
                        └──┬───────────────┘
                           │
                        ┌──▼───────────────┐
                        │ Format Results   │
                        │ (JSON/HTML)      │
                        └──┬───────────────┘
```

```
                        │
                   ┌────▼──────────────┐
                   │ Display to User   │
                   └────┬──────────────┘
                        │
                   ┌────▼──────────────┐
                   │       END         │
                   └───────────────────┘
```

3.9.3 Sequence Diagram: Emotion Classification Request

```
User              Browser            Flask App        Service          Model
 │                  │                   │                │                │
 ├─ Upload Audio ──>                    │                │                │
 │                  │                   │                │                │
 │                  ├─ POST /emotion ──>│                │                │
 │                  │                   │                │                │
 │                  │                   ├─ Load Audio ─>│                │
 │                  │                   │                │                │
 │                  │                   │                ├─ Preprocess ──>│
 │                  │                   │                │                │
 │                  │                   │                │<─ Waveform ────┤
 │                  │                   │                │                │
 │                  │                   │                ├─ Extract Features ──>
 │                  │                   │                │                │
 │                  │                   │                │<─ Embedding ───┤
 │                  │                   │                │                │
 │                  │                   │                ├─ Scale (PCA) ─>│
 │                  │                   │                │                │
 │                  │                   │                ├─ Classify ────>│
 │                  │                   │                │                │
 │                  │                   │<─ Result ──────┤                │
 │                  │                   │                │                │
 │                  │<─ JSON Response ──┤                │                │
 │                  │                   │                │                │
 │<─ Display Result─┤                   │                │                │
```

3.9.4 Class Diagram

```
┌───────────────────────────────┐
│   FlaskApp (Main)             │
├───────────────────────────────┤
│ - app: Flask                  │
│ - emotion_service             │
│ - gender_service              │
│ - intent_service              │
│ - speaker_service             │
├───────────────────────────────┤
│ + setup_routes()              │
```

```
│ + run()                         │
└─────────────────────────────────┘
             │
             │
    ┌────────┼─────────┐
    │        │         │
    ▼        ▼         ▼
┌─────────────────┐ ┌─────────────────┐ ┌─────────────────┐
│ EmotionService  │ │ GenderService   │ │ IntentService   │
├─────────────────┤ ├─────────────────┤ ├─────────────────┤
│ - model         │ │ - model         │ │ - model         │
│ - scaler        │ │ - scaler        │ │ - scaler        │
│ - le            │ │ - le            │ │ - le            │
├─────────────────┤ ├─────────────────┤ ├─────────────────┤
│ + predict()     │ │ + predict()     │ │ + predict()     │
└─────────────────┘ └─────────────────┘ └─────────────────┘

    │
    └───────────────────┐
                        │
                    ┌───▼──────────────────┐
                    │ SpeakerService       │
                    ├──────────────────────┤
                    │ - model              │
                    │ - scaler             │
                    │ - le                 │
                    ├──────────────────────┤
                    │ + predict()          │
                    │ + enroll_speaker()   │
                    └──────────────────────┘

  All Services depend on:

┌─────────────────────────────────┐
│   AudioUtils                    │
├─────────────────────────────────┤
│ - SAMPLE_RATE                   │
│ - SUPPORTED_FORMATS             │
├─────────────────────────────────┤
│ + load_audio()                  │
│ + validate_audio()              │
│ + extract_features()            │
│ + normalize_audio()             │
└─────────────────────────────────┘
```

3.10 Conclusion

3.10.1 Project Summary

The **Speech AI Suite** represents a comprehensive, production-ready multi-task speech analysis system that successfully demonstrates:

1. **Technical Excellence**

- Integration of 3 SOTA self-supervised models (HuBERT, WavLM, XLSR-53)
- Effective transfer learning for 4 distinct speech tasks
- Robust data processing pipeline with comprehensive error handling
- 79.14% accuracy on emotion classification (5-fold CV)

2. **Practical Implementation**

   - Web-based user interface for non-technical users
   - REST API for programmatic access
   - CPU-only deployment (no GPU required)
   - Support for 6+ audio formats and microphone recording

3. **Research Contribution**

   - Reproducible methodology documented with mathematical formulas
   - Comparative analysis of deep learning models
   - Evaluation on standard benchmarks (CREMA-D, SLURP)
   - Published documentation for academic reference

4. **Production Readiness**

   - Modular, maintainable code architecture
   - Comprehensive error handling and logging
   - Version control with Git LFS
   - 90+ KB of documentation for setup, deployment, and extension

3.10.2 Key Achievements

Achievement	Metric
Models Integrated	3 (HuBERT-large, WavLM-base-plus, XLSR-53)
Tasks Implemented	4 (Emotion, Gender, Intent, Speaker)
Classification Classes	6 + 2 + 20+ + Variable
Accuracy (Emotion)	79.14% (5-fold CV)
Inference Speed	2-5 seconds per audio (CPU)
Total Documentation	90+ KB (5 markdown files)
Code Organization	4 separate services (modular)
Frontend Framework	Bootstrap 5 (responsive UI)
Audio Formats	6+ (WAV, MP3, FLAC, OGG, M4A, WebM)
Development Team	5 ML engineers (Sk Inthiyaz, Romith Singh, Rohin Kumar, Sahasra Ganji, Rashmitha)

3.10.3 System Strengths

1. **Accuracy & Performance**
   - 79.14% emotion classification accuracy exceeds industry baseline
   - Fast inference (2-5s) suitable for real-time applications
   - CPU-compatible (no GPU required)
2. **Usability**
   - Intuitive web interface requiring no technical knowledge
   - Multiple input methods (file upload, microphone recording)
   - Clear result visualization with confidence scores
3. **Extensibility**
   - Modular architecture enables easy addition of new tasks
   - Transfer learning approach reduces development time
   - Standardized pipeline applicable to other speech tasks
4. **Documentation**
   - Comprehensive 90+ KB documentation suite
   - Mathematical formulas for all algorithms
   - Code examples and best practices
   - Interview-ready explanation materials

3.10.4 Future Enhancements

1. **Technical Improvements**
   - Fine-tune models on domain-specific data
   - Implement ensemble methods (combine multiple models)
   - Add GPU support for faster inference
   - Real-time streaming support (currently batch processing)
2. **Feature Additions**
   - Multilingual emotion recognition (beyond English)
   - Speaker diarization (who spoke when)
   - Speech enhancement (denoise before classification)
   - Active learning (improve models with user feedback)
3. **Deployment Enhancements**
   - Cloud deployment on AWS/Azure/GCP
   - Docker containerization for easy deployment
   - CI/CD pipeline for automated testing/deployment
   - Monitoring and alerting for production systems
4. **Research Extensions**

- Compare with fine-tuned models
- Study cross-lingual transfer learning
- Analyze emotion recognition across dialects
- Investigate speaker verification robustness

3.10.5 Lessons Learned

1. **Transfer Learning Effectiveness**
   - Pre-trained models require <1% labeled data vs. training from scratch
   - Model selection critical: HuBERT for complex (6-class), WavLM for simple (2-class)
   - Dimensionality reduction (1024→200) improves training speed without accuracy loss
2. **Data Quality Importance**
   - Dataset balance (equal samples per class) improves learning
   - Audio preprocessing (normalization, resampling) critical for consistency
   - Cross-validation (5-fold) provides robust performance estimates
3. **Software Engineering Best Practices**
   - Modular services simplify testing and deployment
   - Comprehensive error handling improves user experience
   - Version control (Git LFS) essential for large model artifacts
   - Documentation crucial for reproducibility and knowledge transfer

3.10.6 Conclusion Statement

The **Speech AI Suite** successfully demonstrates a complete end-to-end machine learning system combining state-of-the-art deep learning models with practical web application design. The project achieves strong performance (79.14% emotion accuracy) while maintaining ease of use and deployment flexibility. The comprehensive documentation, modular architecture, and research-backed methodology make this an ideal reference implementation for speech analysis tasks and a foundation for future research and development.

References & Citation

If using this project in research or publications, please cite:


```
@project{speech-ai-suite-2024,
  title={Speech AI Suite: Multi-Task Speech Analysis Using Self-Supervised
Learning},
  author={Inthiyaz, Sk and Singh, Romith and Kumar, Rohin and Ganji, Sahasra},
  year={2024-2025},
  organization={G-736 Team},
  url={https://github.com/sk-inthiyaz/Emotion-classification}
}

@article{hubert2021,
  title={HuBERT: Self-supervised Speech Representation Learning by Masked
Prediction of Hidden Units},
```

```
  author={Hsu, Wei-Ning and Bolte, Benjamin and Tsai, Yao-Hung Hubert and
Lakhotia, Kushal and others},
  journal={IEEE/ACM Transactions on Audio, Speech, and Language Processing},
  year={2021}
}

@article{wavlm2021,
  title={WavLM: Large-Scale Self-Supervised Pre-training for Full Stack Speech
Processing},
  author={Huang, Sanyuan and Dong, Longquan and Wang, Shuyan and others},
  journal={arXiv preprint arXiv:2110.13900},
  year={2021}
}

@article{xlsr2020,
  title={Unsupervised Cross-lingual Representation Learning at Scale},
  author={Conneau, Alexis and Baevski, Alexei and Collobert, Ronan and others},
  journal={arXiv preprint arXiv:2006.13979},
  year={2020}
}
```

Document Completion: December 11, 2025 **Version:** 1.0 **Status:** Complete & Production Ready ✅ **Total Length:** ~15,000 words covering sections 3.1-3.10

This comprehensive research paper provides complete technical documentation of the Speech AI Suite project, suitable for academic references, technical interviews, and implementation guides.