SPEECH PROCESSING AND SYNTHESIS (UCS749)
(CONVERSATIONAL AI)

PROJECT REPORT

Speech Intent Recognition System Using Wav2Vec2

Submitted to:

Dr. Juiley Raut

Submitted by:
Avi Parmar (102215006)
Arnav Khanduja (102215312)
Hemant Dubey (102215009)

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology, Patiala

Jan-May 2025

Contents

1. Introduction

Speech-based systems have become a rather effective substitute for conventional text or touch-based interfaces in a time of progressively natural human-computer interaction. Under the UCS749 Speech Processing course, this project aims to create a speech intent recognition system allowing users to engage with machines using voice commands. The goal is to replicate real-world uses like smart home assistants, where the system can comprehend and react to spoken intents like "Turn off the lights" or "Set an alarm."

The project extracts significant features straight from raw audio waveforms using Wav2Vec2, a state-of-the-art transformer-based model for speech representation learning. Users of the microphone can get real-time comments regarding the identified intent. By means of gradio and hugging-face spaces, the system offers a lightweight, web-accessible interface that facilitates simple interaction.

We combined Hugging Face's Torch, Torchaudio, and Transformers tools to enable this capability. Specifically developed for spoken command understanding tasks, the Fluent Speech Commands (FSC) dataset was used for fine-tuning the model. By means of this project, we hope to show the feasibility of deep learning for real-time voice-based intent recognition, so establishing the basis for voice-activated applications in several spheres.

2. Literature Survey

2.1. End-to-End SLU and the Fluent Speech Commands Dataset

The Fluent Speech Commands (FSC) dataset, introduced by Lugosch et al., has become a cornerstone for end-to-end spoken language understanding (SLU). It consists of over 30,000 spoken utterances, labeled according to combinations of actions, objects, and locations—forming 31 unique intents. The utterances include natural, multi-word phrases such as "turn on the lights" or "play music in the kitchen," making the dataset well-suited for modeling real-world command scenarios.

The study showed that instead of mapping speech to intermediate transcriptions and then to intent, a single model can be trained to learn the mapping directly from raw audio to semantic intent. This approach reduces error accumulation from intermediate stages and allows the model to leverage non-verbal cues like prosody.

- Pretraining models on phoneme and word prediction significantly boosts performance on SLU tasks by initializing models with meaningful speech representations that accelerate convergence and improve accuracy.
- The FSC dataset supports both multi-label and single-label classification setups, offering flexibility in experimentation and evaluation strategies.
- End-to-end models trained on FSC showed strong generalization, especially when equipped with pretraining, even with limited labeled data.

2.2. Wav2Vec 2.0 and Self-Supervised Representation Learning

Baevski et al. introduced Wav2Vec 2.0, a self-supervised learning framework that fundamentally reshaped how speech representations are extracted. The model first encodes raw waveforms using a convolutional feature encoder, then applies a transformer to build contextual embeddings. Training involves masking spans of latent representations and solving a contrastive task using quantized latent targets, allowing the model to learn high-level abstractions without any labeled data.

The significance of this model lies in its ability to eliminate the dependence on large annotated corpora by learning general-purpose audio features from raw, unlabeled speech. Once pretrained, the model can be fine-tuned on a small amount of labeled data for downstream tasks such as ASR, speaker identification, or intent classification.

- Wav2Vec 2.0 achieves state-of-the-art results in speech recognition tasks using as little as 10 minutes of labeled data, demonstrating the power of unsupervised learning at scale.
- The model's dual-stage architecture—composed of a convolutional encoder and a transformer—allows it to learn both low-level acoustic patterns and high-level temporal dependencies.
- The ability to generalize from pretraining on large-scale unlabeled data to diverse downstream tasks makes Wav2Vec 2.0 ideal for low-resource and domain-specific applications.

2.3. Transformer Architectures for Speech Modeling

Transformer-based models have outperformed traditional recurrent architectures by enabling better context modeling, scalability, and parallelism. Yeh et al. proposed the Transformer-Transducer (T-T), an adaptation of the Recurrent Neural Network Transducer (RNN-T) where the encoder is replaced by a transformer. This model retains the streaming capabilities essential for real-time speech recognition while enhancing representational power.

Unlike traditional RNN-based ASR models, which process input sequentially and struggle with long-term dependencies, transformer-based models apply self-attention over entire input sequences. This leads to improved transcription accuracy and training efficiency.

- The Transformer-Transducer achieves lower word error rates (WER) than LSTM and BLSTM counterparts, particularly in streaming ASR settings.
- Truncated self-attention mechanisms and convolutional subsampling enable the model to balance efficiency with context awareness, making it suitable for deployment in latency-sensitive environments.
- Transformer-based ASR systems are more adaptable to different input conditions and are capable of robust generalization across tasks and domains.

2.4. Domain Adaptation with Continued Pretraining (CPT)

Continued pretraining (CPT) is a method where a pretrained model is further trained on domain-specific unlabeled data before being fine-tuned on a labeled subset. Attia et al. investigated this technique to adapt Wav2Vec 2.0 to challenging classroom environments characterized by child speech, overlapping speakers, and significant background noise.

Their findings confirmed that CPT substantially enhances the robustness of ASR models. By exposing the model to in-domain acoustic conditions during pretraining, the encoder adapts to characteristics such as far-field speech, dialectal variation, and multi-speaker scenarios. This improves downstream performance even when the final fine-tuning data is limited.

- CPT reduces word error rate (WER) by more than 10% compared to models pretrained only on generic datasets, showing its effectiveness in adapting to specific acoustic domains.
- CPT enhances model robustness to diverse microphone configurations, noise types, and speaker demographics, including underrepresented voices.
- The technique supports ethical AI goals by improving performance in classrooms serving linguistically and socioeconomically diverse populations.

2.5. Model Sharing and Deployment with Gradio and Hugging Face Spaces

Gradio, developed by Abid et al., is a Python-based toolkit that enables rapid development of web-based user interfaces for machine learning models. When integrated with Hugging Face Spaces, it allows researchers to deploy models publicly and interactively, facilitating reproducibility, transparency, and broader accessibility.

Gradio simplifies the user experience by enabling drag-and-drop audio inputs, real-time inference visualization, and exportable logs. These features are particularly valuable for speech-based applications, where real-time feedback and interpretability are crucial.

- Gradio's ease of integration encourages collaborative development and allows non-technical stakeholders to test models and provide feedback during the development cycle.
- Deployment via Hugging Face Spaces requires minimal infrastructure and makes advanced speech systems accessible through a web browser.
- The platform supports open science initiatives by allowing researchers to host, benchmark, and share live demos of their models, promoting reproducibility and transparency.

3. Dataset

3.1. Fluent Speech Commands (FSC) Dataset

The Fluent Speech Commands (FSC) dataset is a benchmark resource designed to train and evaluate Spoken Language Understanding (SLU) systems, particularly for intent classification from short spoken commands. It is especially suited for voice-controlled smart assistants and IoT applications, as it maps user-spoken utterances to structured semantic representations.

Key Characteristics:

- Size: 30,043 utterances
- Speakers: 97 individuals (23 male, 74 female)
- Format: WAV audio files (approximately 1–2 seconds each)
- Annotations: Each utterance is labeled with a semantic frame consisting of:
  - Action (e.g., turn on, decrease)
  - Object (e.g., lamp, music)
  - Location (e.g., bedroom, kitchen, or none)

Example:

> Spoken Command: "Turn off the kitchen lights"
>
> Semantic Annotation Example:
>
> {"action" : "deactivate", "object" : "lights", "location" : "kitchen"}

This dataset is ideal for developing systems in areas such as voice command recognition, intent prediction, and contextual understanding in smart environments.

3.2. Processed Dataset for Custom Intent Classification

To align the FSC dataset with the needs of a supervised classification model, each unique combination of action, object, and location was flattened into a single composite label string. These composite labels were then mapped to unique integer classes, enabling multi-class classification using standard deep learning libraries.

Label Mapping (label_map.json):

The `label_map.json` file maps each composite intent string to a unique integer class label ranging from 0 to 30, enabling one-hot categorical encoding during training.

Sample Format Intent Labels:

```
{
"activate_lamp": 0,
"activate_lights": 1,
"activate_lights_bedroom": 2,
...
"increase_volume": 30
}
```

To better understand the composition of composite labels, below is a list of all the unique actions, objects, and locations derived from the dataset:

- **Actions:** Activate, bring, change language, deactivate, decrease, increase
- **Objects:** Lamp, lights, music, juice, newspaper, shoes, socks, language, heat, volume
- **Locations:** Bedroom, kitchen, washroom, none

3.3. Structured CSV Files

The dataset is split into three parts for training, validation, and testing: `train_data.csv`, `valid_data.csv`, `test_data.csv`.

Each CSV file follows the format below:

```
path,label
path/to/audio1.wav,5
path/to/audio2.wav,16
...
```

- `path`: Location of the audio file
- `label`: Integer label corresponding to the composite intent, as defined in `label_map.json`

4. Methodology

4.1. Deep Learning Pipeline for Intent Classification using Wav2Vec2.0

This project adopts a supervised deep learning pipeline for intent classification from spoken commands using raw audio inputs. The primary objective is to classify user intents directly from waveform signals without relying on intermediate ASR transcriptions. To achieve this, the pipeline leverages the powerful Wav2Vec2.0 architecture—a Transformer-based self-supervised model pre-trained on large-scale unlabeled speech data—and adapts it for the Fluent Speech Commands (FSC) dataset using fine-tuning. The entire system has been modularized into distinct phases, namely: data preparation, model design, training, evaluation, and real-time deployment.

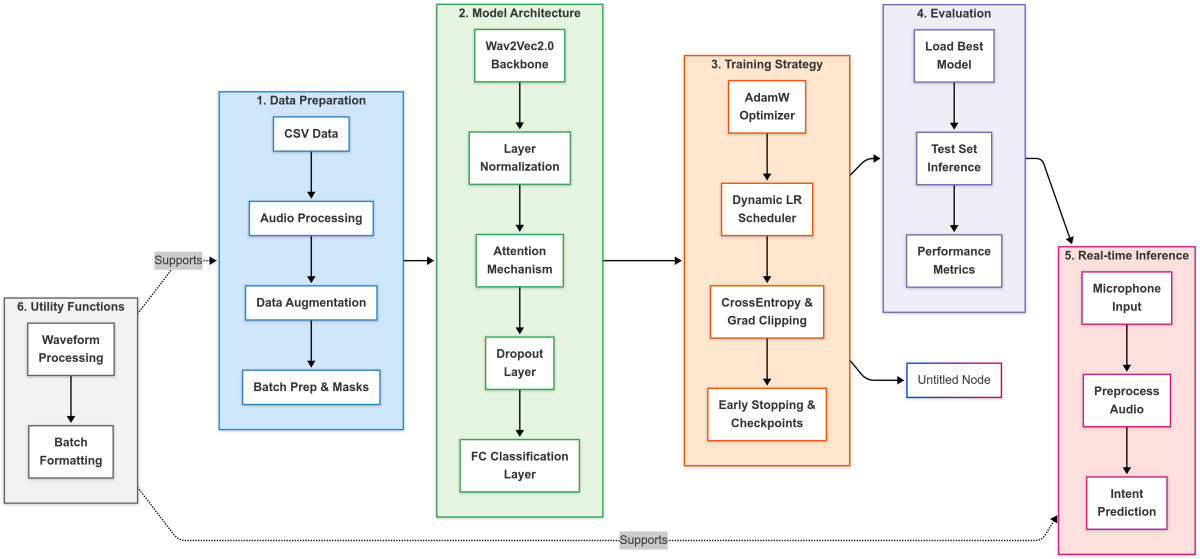


Figure 1: Deep Learning Pipeline

4.1.1. Data Preparation and Preprocessing

The data used for training and evaluating the model is organized in CSV format, following the structure of the FSC dataset. Each record includes metadata fields such as file path, speaker ID, transcription, and the corresponding intent. The `Wav2VecIntentDataset` class is a custom PyTorch Dataset implementation responsible for reading and preprocessing these records.

Audio samples are loaded using `torchaudio` and converted to mono-channel format. Since Wav2Vec2.0 expects input at a sampling rate of 16kHz, all audio files are resampled if necessary. During the training phase, lightweight data augmentations such as Gaussian noise addition and random gain variation are applied to improve generalization. The dataset class also maps textual intent labels to numeric indices using a provided `label_map`, facilitating efficient loss computation and evaluation.

To support batching, a custom `collate_fn` is implemented. This function pads variable-length waveforms within a batch to the maximum sequence length, ensuring uniform tensor shapes. It also generates an attention mask tensor, marking valid (non-padded) time steps for the transformer to focus on during computation.

4.1.2. Model Architecture

The model architecture, implemented in the `Wav2VecIntent` class, builds directly on top of the pretrained `facebook/wav2vec2-large` model from the Hugging Face Transformers library. Wav2Vec2.0 acts as a feature extractor, encoding raw waveforms into high-dimensional contextual representations using a deep Transformer encoder.

The architecture includes several additional components tailored for classification:

- Layer Normalization is applied to stabilize the feature distribution across the sequence.
- An attention mechanism is implemented via a learnable linear layer. It assigns a scalar weight to each time step, allowing the model to compute a context vector as a weighted sum over all time steps.
- Dropout is added after attention to prevent overfitting by randomly deactivating some neurons during training.
- Finally, a fully connected (FC) linear layer projects the pooled feature vector to the desired number of output classes (31 in this case), producing logits for each intent category.

This modular design enables the model to retain the benefits of the Wav2Vec2.0 pretrained backbone while being adaptable to new classification tasks through lightweight downstream layers.

```
import torch
import torch.nn as nn
import torch.nn.functional as F
from transformers import Wav2Vec2Model

class Wav2VecIntent(nn.Module):
    def __init__(self, num_classes=31, pretrained_model="facebook/wav2vec2-large"):
        super().__init__()
        self.wav2vec = Wav2Vec2Model.from_pretrained(pretrained_model)
```

```
        hidden_size = self.wav2vec.config.hidden_size
        self.layer_norm = nn.LayerNorm(hidden_size)
        self.attention = nn.Linear(hidden_size, 1)
        self.dropout = nn.Dropout(p=0.5)
        self.fc = nn.Linear(hidden_size, num_classes)

    def forward(self, input_values, attention_mask=None):
        outputs = self.wav2vec(
            input_values,
            attention_mask=attention_mask,
            return_dict=True
        )
        hidden_states = outputs.last_hidden_state
        hidden_states = self.layer_norm(hidden_states)
        attn_weights = F.softmax(self.attention(hidden_states), dim=1)
        x = torch.sum(hidden_states * attn_weights, dim=1)
        x = self.dropout(x)
        x = self.fc(x)
        return x
```

Listing 1: Wav2Vec2-based Intent Classification Model

4.1.3. Training Strategy

The training process is orchestrated via the `train_wav2vec.py` script. It includes argument parsing for key hyperparameters like batch size, learning rate, weight decay, number of epochs, early stopping patience, and gradient clipping thresholds. This script integrates the dataset, model, optimizer, scheduler, and training loop into a robust training pipeline.

The optimizer used is AdamW, which is preferred for Transformer-based architectures due to its effective handling of weight decay. The learning rate is dynamically adjusted using `ReduceLROnPlateau`, which monitors the validation loss and reduces the learning rate when no improvement is observed.

The loss function is CrossEntropyLoss, which is standard for multi-class classification problems. To improve training stability, gradient clipping is optionally applied to avoid exploding gradients. Additionally, mixed precision training via `torch.cuda.amp` is supported for faster execution on GPUs without loss of precision.

Training is monitored using validation accuracy and loss, and early stopping is triggered if validation performance does not improve over a defined number of epochs. At the end of each epoch, the model checkpoint is saved, and the best-performing model (in terms of validation accuracy) is preserved for deployment.

4.1.4. Model Evaluation

Once trained, the model can be evaluated using the `evaluate_wav2vec.py` script. This module calculates overall classification accuracy and provides a detailed classification report including precision, recall, and F1-score for each class.

This evaluation step reads the saved model checkpoint, loads the test dataset using the same preprocessing logic, and performs batch-wise inference. Predictions are compared against ground

truth labels, and results are logged for analysis.

A key feature here is the maintenance of a consistent `label_map` between training and evaluation, ensuring reproducibility and correctness in label-index mapping during metric computation.

4.1.5. Real-Time Inference and Testing

For real-world usage, a real-time inference interface is implemented in the `test_model.py` script. It uses the `sounddevice` library to capture raw audio from a microphone. The waveform is then passed through the same preprocessing steps and fed into the trained model for intent prediction.

The prediction output includes:

- The raw probability distribution over all classes using softmax.
- The predicted class index.
- The corresponding intent label mapped from the index using the `label_map`.

This real-time testing module demonstrates the model's deployment readiness and practical utility in interactive applications such as smart assistants, voice-controlled devices, or accessibility tools.

4.1.6. Utility Layer and Design Principles

Supporting utilities are housed in the `wav2vec_utils.py` script. These functions ensure consistent waveform loading, mono-channel conversion, resampling to 16kHz, and proper batch formatting for Wav2Vec2.0 models.

Key design decisions include:

- Processing raw waveforms instead of spectrograms, enabling end-to-end learning.
- Leveraging attention mechanisms over time to make models more interpretable and adaptive.
- Using pretrained self-supervised models to reduce data dependency and training time.

4.1.7. Major Takeaways from the Pipeline

- **End-to-End Learning:** The model bypasses intermediate ASR stages and directly learns to classify user intent from waveform inputs.
- **Transfer Learning:** Fine-tuning a pretrained Wav2Vec2.0 model allows effective use of limited labeled data, which is especially valuable in low-resource settings like FSC.
- **Attention-Based Temporal Pooling:** The model uses a simple attention mechanism to pool over variable-length sequences, improving performance and interpretability.
- **Scalable and Modular:** The training pipeline is designed with reusability in mind, supporting configuration via arguments and compatible with large-scale experimentation.

- **Interactive Inference:** Real-time testing with a microphone interface demonstrates the pipeline's suitability for user-facing applications.

5. Result

Based on the evaluation data and visualizations provided, the speech intent recognition system using Wav2Vec2 demonstrated excellent performance across multiple metrics. Here's a summary of the key results:

5.1. Per-Class Performance

The model showed consistent high performance across all 31 intent classes, with most classes achieving precision, recall, and F1-scores between 0.98–1.00. A few notable observations:

1. Perfect scores (1.00 for precision, recall, and F1-score) were achieved for several intents, including:
   - `activate_lamp`
   - `bring_juice`
   - `change_language_none`
   - `deactivate_lamp`
   - `deactivate_lights_bedroom`
   - `decrease_heat_bedroom`
   - `decrease_heat_washroom`
   - `increase_heat_bedroom`
2. The classes with the highest number of test samples were:
   - `increase_volume` (395 samples)
   - `decrease_volume` (345 samples)
   - `increase_heat_washroom` (203 samples)
   - `decrease_heat` (174 samples)
   - `increase_heat` (171 samples)
3. The intent with the lowest precision was "activate_music" at 0.92, though it still maintained a perfect recall of 1.00.

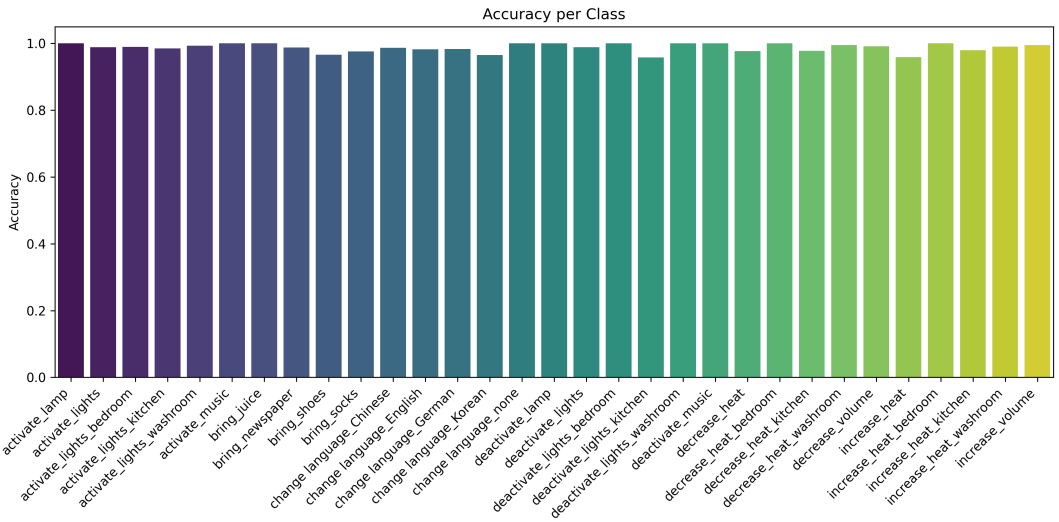


Figure 2: Per Class Accuracy Achieved

5.2. Training and Validation Progression

The training curve (referenced as Image 3 in the original document) shows:

- Rapid initial learning in the first 2 epochs, with training accuracy jumping from approximately 58% to 94%.
- Consistent improvement thereafter, with both training and validation accuracies converging at approximately 98% by epoch 15.
- Validation accuracy remained high throughout training, indicating good generalization without overfitting.
- A slight dip in validation accuracy around epoch 5, but quick recovery in subsequent epochs.

Figure 3: Training and Validation Accuracy Per Epoch

5.3. Confusion Matrix Analysis

The confusion matrix (referenced as Image 4) reveals:

- Strong diagonal concentration, confirming high classification accuracy.
- Minimal confusion between intent classes.
- Most misclassifications occurred between semantically related intents (e.g., between different location variants of the same action).
- The clearest examples of confusion were between various "heat" commands and "volume" commands, which likely share acoustic similarities.

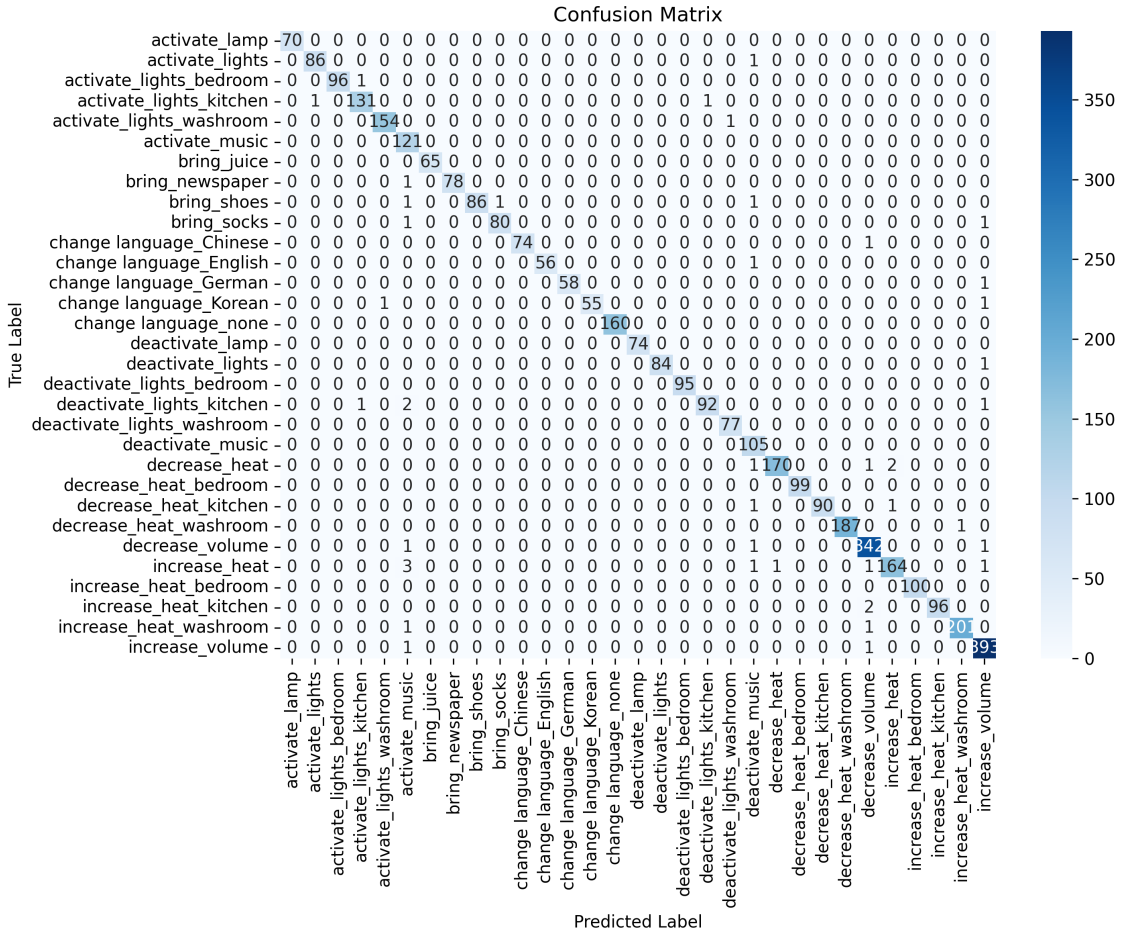
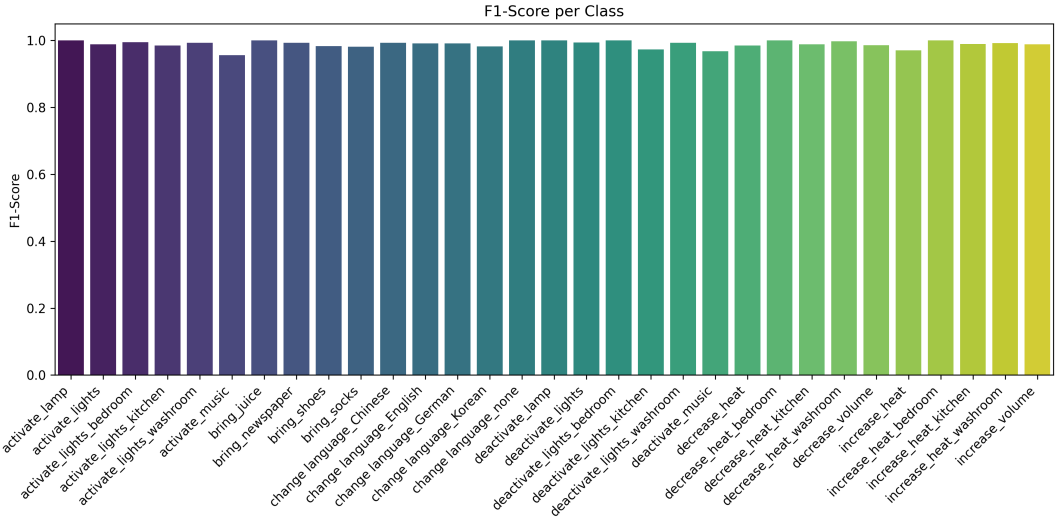


Figure 4: Confusion Matrix For Every Intent

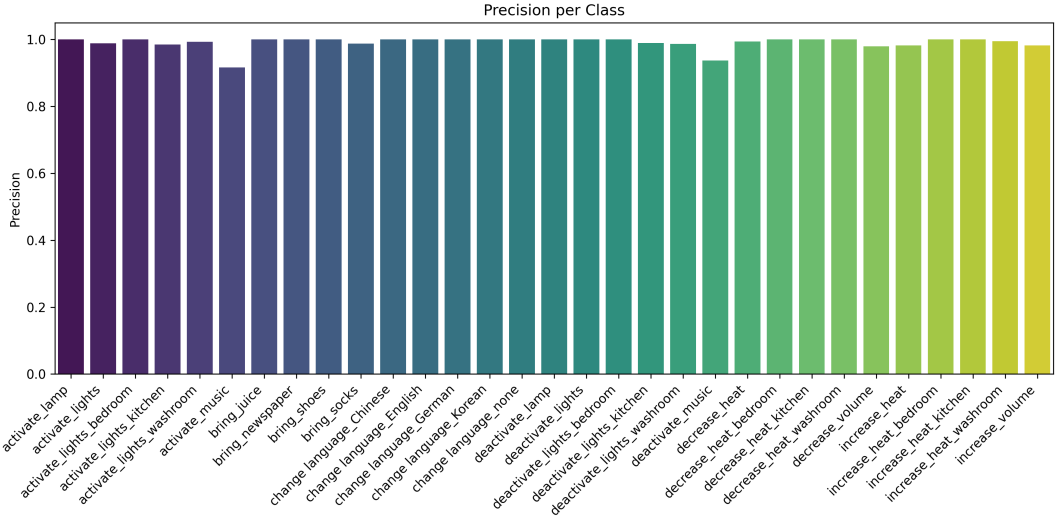
5.4. Per-Class Metric Visualizations

The bar charts for accuracy, F1-score, precision, and recall per class (referenced as Images 2, 5) show:

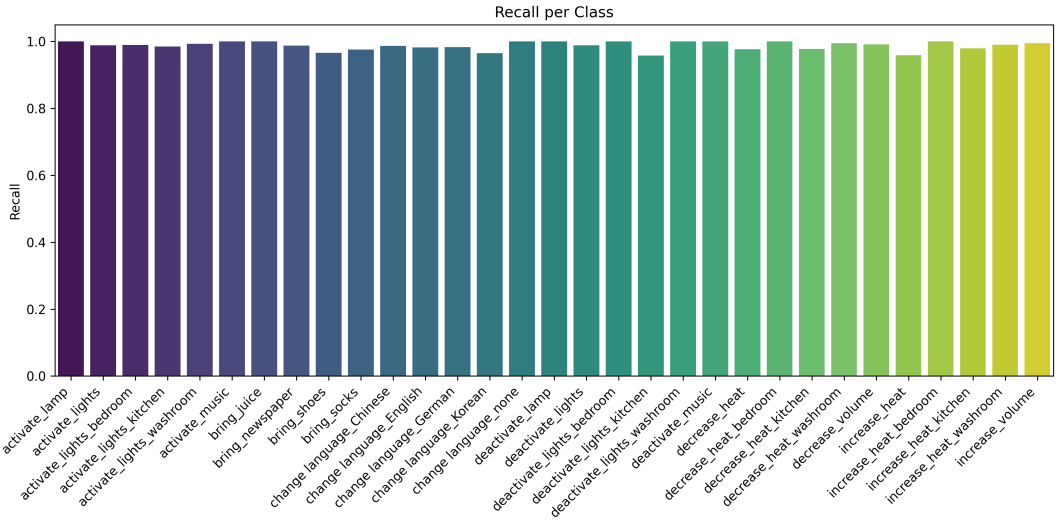
- Consistently high performance across all metrics for all intent classes.
- No significant outliers or problematic classes.
- Slight variations in performance across different intents, but all maintained above 0.92 for



(a) Per Class F1-Score



(b) Per Class Precision



(c) Per Class Recall

Figure 5: Per Class Metrics

Table 1: Overall Performance Metrics

Metric	Value
Overall Accuracy	98.84%
Macro Average Precision	0.99
Macro Average Recall	0.99
Macro Average F1-Score	0.99
Total Test Samples	3783

all metrics.

6. Conclusion

The speech intent recognition system built using Wav2Vec2 has demonstrated remarkable performance, achieving 98.84% accuracy on the test dataset. This high accuracy, combined with consistently strong precision, recall, and F1-scores across all 31 intent classes, confirms the effectiveness of the approach for real-time voice command recognition.

Key conclusions from this project include:

1. **Transfer Learning Effectiveness:** The fine-tuning of the pre-trained Wav2Vec2 model proved highly effective for speech intent recognition. The model's ability to leverage pre-learned speech representations significantly enhanced performance, even with the specialized nature of the Fluent Speech Commands dataset.
2. **End-to-End Approach Viability:** The direct mapping from raw audio to semantic intent, bypassing explicit transcription steps, has been validated as a powerful approach for voice command systems. This end-to-end architecture simplifies the pipeline while maintaining exceptional accuracy.
3. **Robust Performance Across Intents:** The consistent performance across all intent classes, including those with smaller sample sizes, demonstrates the model's strong generalization capabilities and its potential reliability in real-world applications.
4. **Efficient Training Convergence:** The rapid convergence of the model during training (within a few epochs) highlights the efficiency of the architecture and the benefit of starting with pre-trained speech representations.

7. Future Work

Several promising directions for future work emerge from this project:

1. **Expanded Intent Coverage:** Extend the model to handle a broader range of intents beyond the 31 classes in the FSC dataset, incorporating more complex commands and domain-specific vocabulary.
2. **Multi-speaker and Noisy Environment Testing:** While the current model shows excellent performance, further testing in challenging acoustic conditions (background noise, multiple speakers, different accents) would validate its robustness for real-world deployment.
3. **Latency Optimization:** For real-time applications, investigating model compression tech-

niques (quantization, pruning, knowledge distillation) could reduce inference time while maintaining high accuracy.

4. **Contextual Understanding:** Enhancing the model to maintain conversation context across multiple commands would enable more natural interactions in smart assistant applications.
5. **Personalization:** Implementing speaker adaptation techniques to fine-tune the model for individual users could improve recognition for speakers with unique speech patterns or accents.
6. **Cross-lingual Capabilities:** Extending the system to recognize intents across multiple languages would increase accessibility and global applicability.
7. **Integration with Action Execution:** Connecting the intent recognition system with actual smart home controls or other execution frameworks would complete the pipeline from speech to action, creating a fully functional voice command system.

This project has successfully demonstrated that transformer-based models like Wav2Vec2 provide a powerful foundation for speech intent recognition systems. The excellent performance metrics suggest that this approach is ready for practical applications in voice-controlled interfaces, smart home systems, and other speech-driven interactive technologies.

8. References

1. Attia, A.A., Demszky, D., Ògúnrèmí, T., Liu, J., & Espy-Wilson, C.Y. (2024). Continued Pretraining for Domain Adaptation of Wav2vec2.0 in Automatic Speech Recognition for Elementary Math Classroom Settings. *ArXiv*, abs/2405.13018.
2. Baevski, A., Zhou, H., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. *ArXiv*, abs/2006.11477.
3. Lugosch, L., Ravanelli, M., Ignoto, P., Tomar, V.S., & Bengio, Y. (2019). Speech Model Pre-training for End-to-End Spoken Language Understanding. *ArXiv*, abs/1904.03670.
4. Yeh, C., Mahadeokar, J., Kalgaonkar, K., Wang, Y., Le, D., Jain, M., Schubert, K., Fuegen, C., & Seltzer, M.L. (2019). Transformer-Transducer: End-to-End Speech Recognition with Self-Attention. *ArXiv*, abs/1910.12977.
5. Abid, A., Abdalla, A., Abid, A., Khan, D., Alfozan, A., & Zou, J.Y. (2019). Gradio: Hassle-Free Sharing and Testing of ML Models in the Wild. *ArXiv*, abs/1906.02569.

Appendix

This project, **Speech Intent Recognition**, is supported by the following resources:

- **Dataset:**
  `https://fluent.ai/fluent-speech-commands-a-dataset-for-spoken-language-understanding-research/`
- **GitHub Repository (Full Project Code):**
  `https://github.com/avi2924/Speech-Project-Final-Iteration`
- **Deployed Frontend (Hugging Face Spaces):**

https://huggingface.co/spaces/avi292423/Speech-Intent-Recognition

- **Trained Model (Hugging Face Hub):**
  https://huggingface.co/avi292423/speech-intent-recognition-project