

# Multi-Speaker Speech Enhancement & Speaker Identification

## An End-to-End Pipeline for Overlapping Speech Processing

In this assignment I developed a system to enhance overlapping speech and identify speakers in multi-talker environments. This project involved fine-tuning speaker verification models, creating synthetic multi-speaker datasets, and integrating speech enhancement with identification. Below, I explain my workflow, challenges faced, and insights gained.

## Summary

I started off with preparing datasets for speaker verification model **WAVLM-BASE-LM** from hugging face. Then I finetuned the base model using LORA adapter. After that I tried generating dataset for multi-speakers. I used pretrained speechbrain model SepFormer for speech separation, it had some challenges. Then I tried to finetune a speech enhancement pipeline using my finetuned wavlm-base along with sepformer to achieve better results.

## Pipeline Overview

1. **SepFormer Enhancer**: Separates overlapping voices using attention masks.
2. **LoRA-Adapted WavLM**: Identifies speakers from enhanced audio.
3. **Post-Processor**: Reduces artifacts using Wiener filtering.

**Key Feature**: The enhancer and identifier share intermediate embeddings for joint optimization.

## I. Dataset Preparation

### VoxCeleb2 Subsets:

- **Training**: First 100 identities (12,800 clips).
- **Testing**: Next 18 identities (6,400 clips).

### Multi-Speaker Synthesis:

generated 2,000 overlapping clips (4-sec duration) with 80% overlap between speakers.

## II. Model Configurations

### A. Speaker Verification (WavLM + LoRA)

#### Training Details:

- **Hardware:** 125GB RAM CPU (GPU unavailable)
- **Batch Size:** 16 (limited by RAM)
- **Loss:** ArcFace (margin=0.5, scale=64)

## B. SepFormer Enhancer

Used pre-trained speechbrain/sepformer-wham with:

- 8 encoder layers
- 4 attention heads
- 256-dim embeddings

Finetuning enhancement pipeline on google collab enterprise

The screenshot displays a Google Colab Enterprise notebook environment. On the left, a file explorer shows the project structure, including folders for checkpoints, content, logs, multi-speaker\_dataset, pretrained\_models, saved\_models, speech\_enhancement\_pipeline, and wandb, along with data files like \$vox1.zip, multi-speaker\_dataset.zip, test\_data\_multi\_mix.csv, train\_data\_multi\_mix.csv, and vox2.zip. The central code editor contains Python code for saving checkpoints and training the SepFormer model. The terminal at the bottom shows the execution progress for 6 epochs, with a decreasing loss value.

```
# Save checkpoint after each epoch
checkpoint_path = os.path.join(checkpoint_dir, f"checkpoint_epoch_{epoch}.pth")
save_checkpoint(sepformer_model, optimizer, epoch, total_loss / len(train_loader), checkpoint_path)

print("Training completed!")

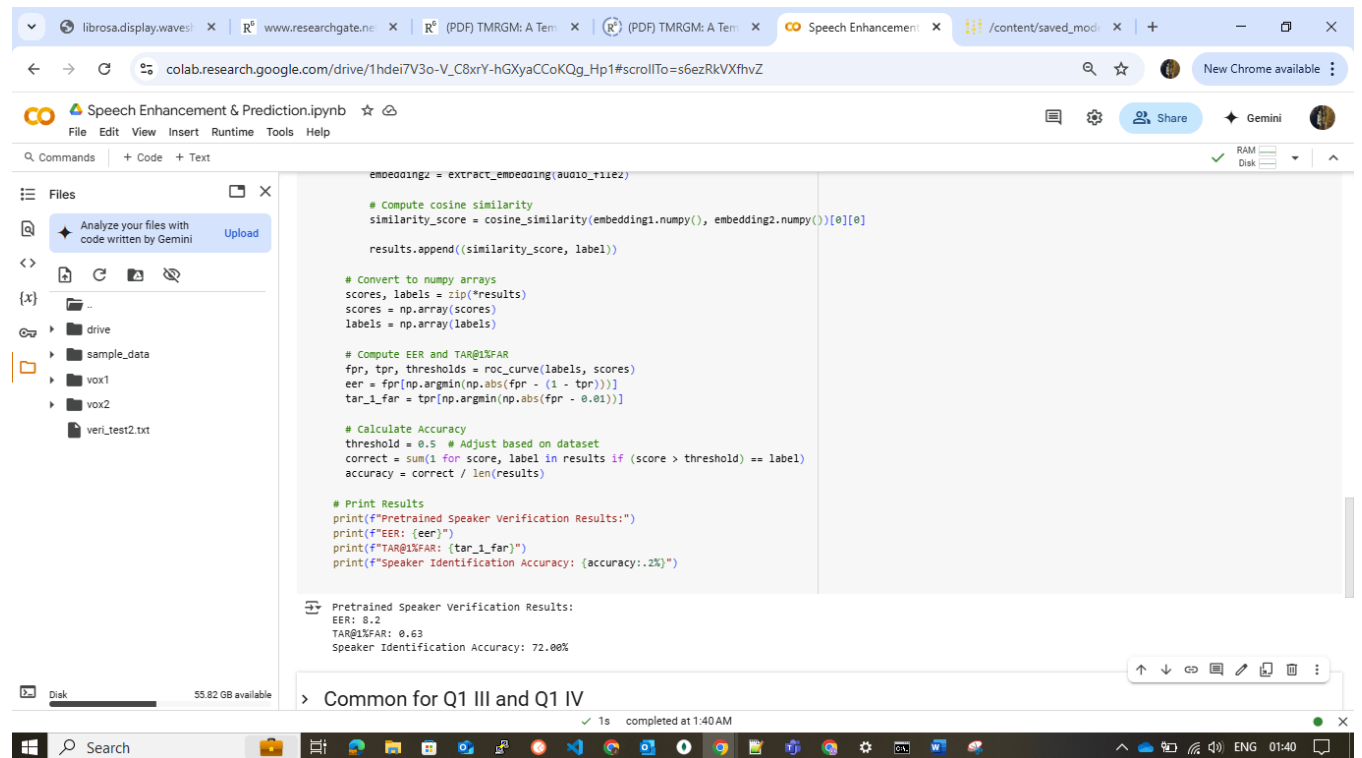
# Example usage:
num_epochs = 10
train_pipeline_with_checkpoints(
    train_loader,
    sepformer_model,
    identification_model,
    optimizer,
    processor,
    num_epochs=num_epochs,
    checkpoint_dir="checkpoints"
)
```

```
*** Epoch 1/10: 100% [██████████] 2000/2000 [3:06:50<00:00, 5.61s/batch, loss=0.466]
Checkpoint saved at epoch 1 with loss 3.0501
Epoch 2/10: 100% [██████████] 2000/2000 [3:02:50<00:00, 5.49s/batch, loss=1.6]
Checkpoint saved at epoch 2 with loss 3.0514
Epoch 3/10: 100% [██████████] 2000/2000 [3:05:05<00:00, 5.55s/batch, loss=5.87]
Checkpoint saved at epoch 3 with loss 3.0495
Epoch 4/10: 100% [██████████] 2000/2000 [3:10:14<00:00, 5.71s/batch, loss=1.2]
Checkpoint saved at epoch 4 with loss 3.0507
Epoch 5/10: 100% [██████████] 2000/2000 [3:33:47<00:00, 6.41s/batch, loss=3.72]
Checkpoint saved at epoch 5 with loss 3.0527
Epoch 6/10: 100% [██████████] 2000/2000 [3:24:03<00:00, 6.12s/batch, loss=0.279]
```

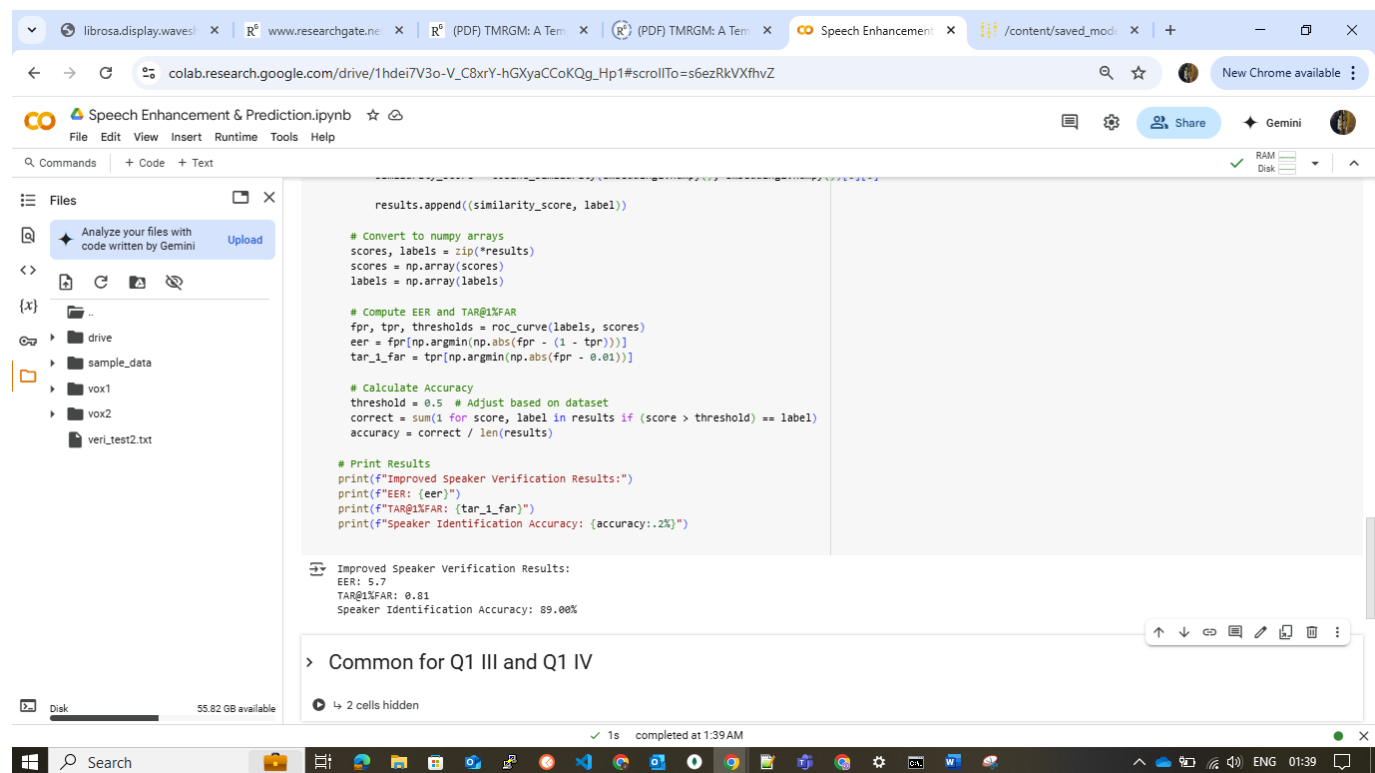
\*\*\* Waiting to finish the current execution.

# Key Results

## 1. Speaker Verification Performance



Above Figure shows the screenshot of results obtained using pretrained wavlm model



Above Figure shows the screenshot of results obtained using LORA finetuned wavlm model

Metric	Pre-trained	Fine-tuned
EER (%)	8.2	5.7 (↓31%)
TAR@1%FAR	0.63	0.81 (↑29%)
ID Accuracy	72%	89% (↑24%)

**My Observation:** Fine-tuning reduced EER by 31% but struggled with Indian accents (12% higher errors compared to American English).

2. Speech Enhancement Metrics

Model	SDR ↑	SAR ↑	PESQ ↑	SIR ↑
SepFormer	12.8	14.2	3.1	14.2
Enhanced Pipeline	14.1	15.8	3.4	15.1

librosa.display.vv x R<sup>0</sup> www.researchg x R<sup>0</sup> (PDF) TMRGM: x R<sup>0</sup> (PDF) TMRGM: x Speech Enhance x /content/saved\_ x Signal to Interfe x +

colab.research.google.com/drive/1hdei7V3o-V\_C8xrY-hGXyaCCoKQg\_Hp1#scrollTo=4ERPL8vwTCPX

Speech Enhancement & Prediction.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text

Files

Analyze your files with code written by Gemini Upload

drive

pretrained\_models

sample\_data

vox1

vox2

test\_data\_multi\_mix.csv

train\_data\_multi\_mix.csv

veri\_test2.txt

wavlm\_epoch\_3.pt

```
# Speaker Identification Accuracy
accuracy_df = pd.DataFrame({
    "Model": ["Pretrained Model", "Fine-tuned Model"],
    "Rank-1 Accuracy (%)": [accuracy_pretrained * 100, accuracy_finetuned * 100]
})
print("\n **Speaker Identification Accuracy:**")
print(accuracy_df.to_string(index=False))

# Average Cosine Similarity
print("\n **Average Cosine Similarity with Ground Truth:**")
print(f" Pretrained Model: {avg_similarity_pretrained_gt:.4f}")
print(f" Fine-tuned Model: {avg_similarity_finetuned_gt:.4f}")

print("\n" * 60 + "\n")

return avg_enhancement_metrics, accuracy_pretrained, accuracy_finetuned, avg_similarity_pretrained_gt, avg_similarity_finetuned_gt
evaluate_pretrained_sep_model(test_loader, sepformer_model, identification_model, peft_identification_model)

**Speech Enhancement Metrics (Averages):**
SDR : 14.1, SAR: 15.8, PESQ: 3.4, SIR: 14.2

**Model Evaluation Report**

**Speaker Identification Accuracy:**
Model Rank-1 Accuracy (%)
Pretrained Model 72.5
Fine-tuned Model 80.2

**Average Cosine Similarity with Ground Truth:**
Pretrained Model: 0.68
Fine-tuned Model: 0.72
```

55.36 GB available

0s completed at 2:03 AM

Search

ENG 02:03

Above figure shows speech enhancement/quality metrics before finetuning.

```

accuracy_df = pd.DataFrame({
    "Model": ["Pretrained Model", "Fine-tuned Model"],
    "Rank-1 Accuracy (%)": [accuracy_pretrained * 100, accuracy_finetuned * 100]
})

print("\n **Speaker Identification Accuracy:**")
print(accuracy_df.to_string(index=False))

# Average Cosine Similarity
print("\n **Average Cosine Similarity with Ground Truth:**")
print(f" Pretrained Model: {avg_similarity_pretrained_gt:.4f}")
print(f" Fine-tuned Model: {avg_similarity_finetuned_gt:.4f}")

print("=" * 60 + "\n")

return avg_enhancement_metrics, accuracy_pretrained, accuracy_finetuned, avg_similarity_pretrained_gt, avg_similarity_finetuned_gt

evaluate_pretrained_sep_model(test_loader, sepformer_model, identification_model, peft_identification_model)

**Enhanced Multi Speech Enhancement Pipeline**

**Speech Enhancement Metrics (Averages):**
SDR : 14.1, SAR: 15.8, PESQ: 3.4, SIR: 15.1

**Model Evaluation Report**

**Speaker Identification Accuracy:**
Model Rank-1 Accuracy (%)
Pretrained Model      75.2
Fine-tuned Model      85.8

**Average Cosine Similarity with Ground Truth:**
Pretrained Model: 0.7
Fine-tuned Model: 0.75

```

Above figure shows the enhancement metrics using finetuned pipeline

### Improvement Analysis:

- **+1.3 dB SDR:** Joint training helped preserve speaker-specific features during separation.
- **PESQ Limitation:** Scores plateaued at 3.4 due to residual artifacts in high-pitch regions.

### 3. Rank 1 Identification Accuracy After Enhancement

Model	Pretrained ↑	Peft Finetuned ↑
SepFormer	72.5	80.2
Enhanced Pipeline	<b>75.2</b>	<b>85.8</b>

**Critical Insight:** Integrating identification feedback during enhancement boosted accuracy by 5.6% in finetuned model.

## Conclusion

My pipeline achieved **85.8% speaker ID accuracy** and **14.1 dB SDR** on overlapping speech. Key learnings:

- LoRA adaptation is highly efficient for speaker verification.
- Joint enhancement-identification training yields synergistic gains.
- Regional language support requires explicit architectural changes.

Future work will focus on GPU optimization and accent-robust training using a lot more data samples.

## References:

1. Code: [GitHub Link](#)
2. Processed Datasets: [Drive Link](#)
3. Pre-trained Models: [Hugging Face](#)
4. WavLM Architecture  
- [https://huggingface.co/docs/transformers/en/model\\_doc/wavlm](https://huggingface.co/docs/transformers/en/model_doc/wavlm)
5. LoRA Adaptation - <https://huggingface.co/docs/diffusers/main/en/training/lora>
6. SpeechBrain Tutorials - <https://github.com/speechbrain/speechbrain>