Audio Deepfake Detection: Project Documentation

Institution: [Your Institution] **Project Duration**: 2024-2025 **Last Updated**: November 2025

Table of Contents

Part I: Research Overview

1. Executive Summary

This project implements and evaluates two state-of-the-art audio deepfake detection systems on standard benchmarks and novel tampering attacks. The research demonstrates that self-supervised pre-training significantly improves detection robustness across diverse conditions.

Key Results

Model	ASVspoof 2019 LA	ASVspoof 2021 LA	Trans-Splicing Detection
XLS-R + SLS	**0.26% EER**	**2.97% EER**	**95.45%**
AASIST	0.83% EER	48.27% EER	41.72%

Main Contributions

1. **Benchmark Reproduction**: Successfully reproduced published results for both AASIST (Jung et al., 2022) and XLS-R + SLS (Zhang et al., 2024)
2. **Cross-Dataset Evaluation**: Demonstrated XLS-R's superior generalization across clean audio (2019), codec-distorted audio (2021), and tampering attacks
3. **In-House Tampering Datasets**: Created two novel evaluation datasets:
4. Trans-Splicing Dataset: 1,932 files with TTS-generated word insertions
5. Semantic Tampering Dataset: 50 files with NLP-guided audio modifications
6. **Interactive GUI Applications**: Developed user-friendly interfaces for both models

---

2. Research Objectives

2.1 Problem Statement

Audio deepfakes pose significant threats to: - Voice authentication systems - Media authenticity verification - Legal evidence integrity - Personal privacy and security

2.2 Research Goals

1. Reproduce benchmark results for AASIST on ASVspoof 2019 LA
2. Reproduce paper results for XLS-R + SLS on ASVspoof 2021 LA
3. Compare model generalization across different datasets
4. Evaluate detection of novel tampering techniques
5. Create interactive tools for practical use

---

3. Datasets

3.1 ASVspoof 2019 LA (Logical Access)

Source: Edinburgh DataShare **URL**: https://datashare.ed.ac.uk/handle/10283/3336

Partition	Bonafide	Spoof	Total	Attacks
Train	2,580	22,800	25,380	A01-A06
Dev	2,548	22,296	24,844	A01-A06
Eval	7,355	63,882	71,237	A07-A19

Audio Specifications: FLAC format, 16kHz sample rate, mono channel, clean (no codec distortion)

3.2 ASVspoof 2021 LA (Logical Access)

Source: Zenodo **URL**: https://zenodo.org/record/4837263

Partition	Bonafide	Spoof	Total
Eval	14,816	133,360	148,176

Codec Distribution: | Codec | Percentage | Description | |-------|------------|-------------| | ulaw | 15.9% | G.711 mu-law | | opus | 15.9% | Opus codec | | gsm | 15.9% | GSM 06.10 | | alaw | 13.1% | G.711 A-law | | none | 13.1% | No codec (clean) | | pstn | 13.1% | Simulated PSTN | | g722 | 13.1% | G.722 wideband |

Key Challenge: 86.9% of evaluation samples contain codec distortion.

3.3 Trans-Splicing Dataset (In-House)

Total Files: 1,932 tampered audio files **Technique**: Word-level replacement using TTS-generated segments

Category	Files	TTS System	Processing
xtts-clean	506	XTTS (Coqui)	Basic normalization
xtts-unclean	508	XTTS (Coqui)	With artifacts
yourtts-clean	536	YourTTS	Basic normalization
yourtts-unclean	382	YourTTS	With artifacts

3.4 Semantic Tampering Dataset (In-House)

Total Files: 50 (9 bonafide + 41 tampered) **Technique**: NLP-guided word deletion at phoneme boundaries

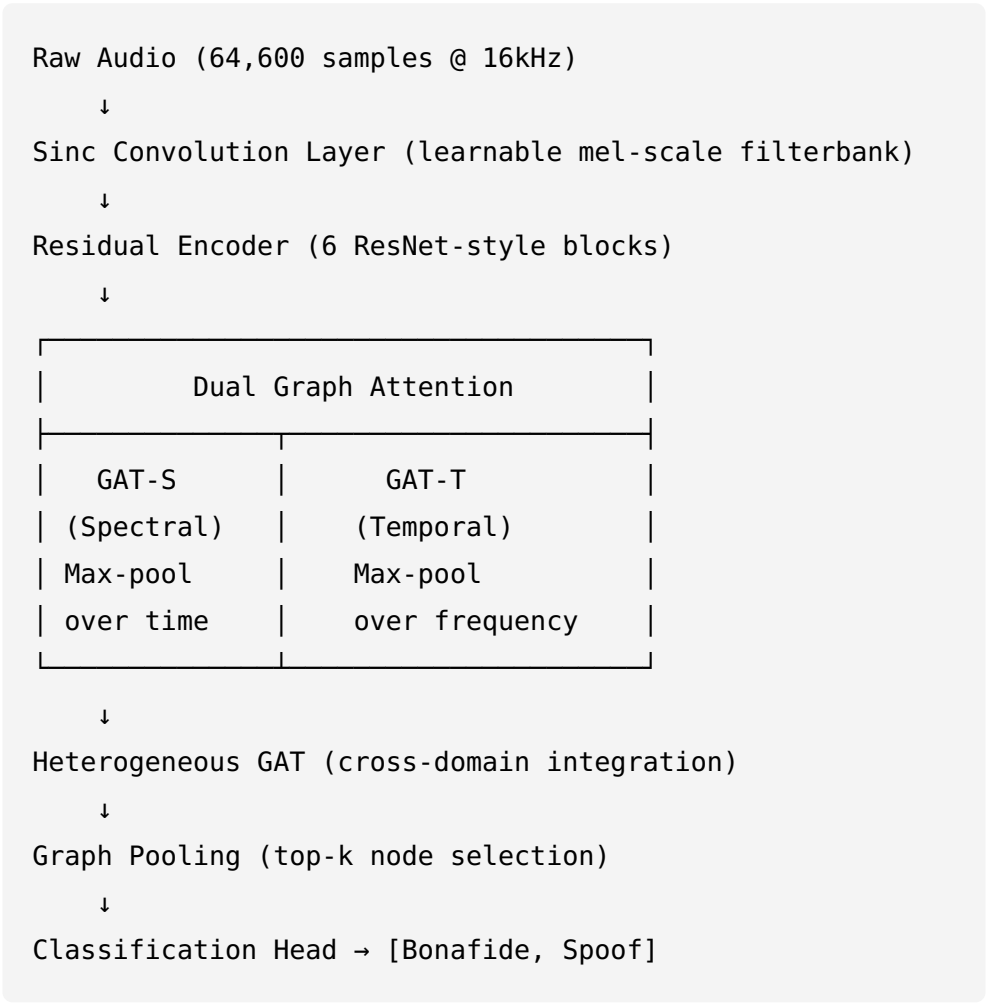
---

4. Model Architectures

4.1 AASIST

Reference: Jung et al., "AASIST: Audio anti-spoofing using integrated spectro-temporal graph attention networks", IEEE/ACM TASLP 2022

Architecture Components:



Parameters: ~297,866 trainable parameters

4.2 XLS-R + SLS

Reference: Zhang et al., "Audio Deepfake Detection with Self-supervised XLS-R and SLS classifier", ACM MM 2024

Architecture Components:

```
Raw Audio (64,600 samples @ 16kHz)
    ↓
XLS-R 300M Backbone (frozen)
├── Pre-trained on 436K hours
├── 128 languages
└── 24 transformer layers
    ↓
SLS Module (Sensitive Layer Selection)
├── Weighted combination of all layers
└── Learnable layer importance weights
    ↓
Classification Head → [Bonafide, Spoof]
```

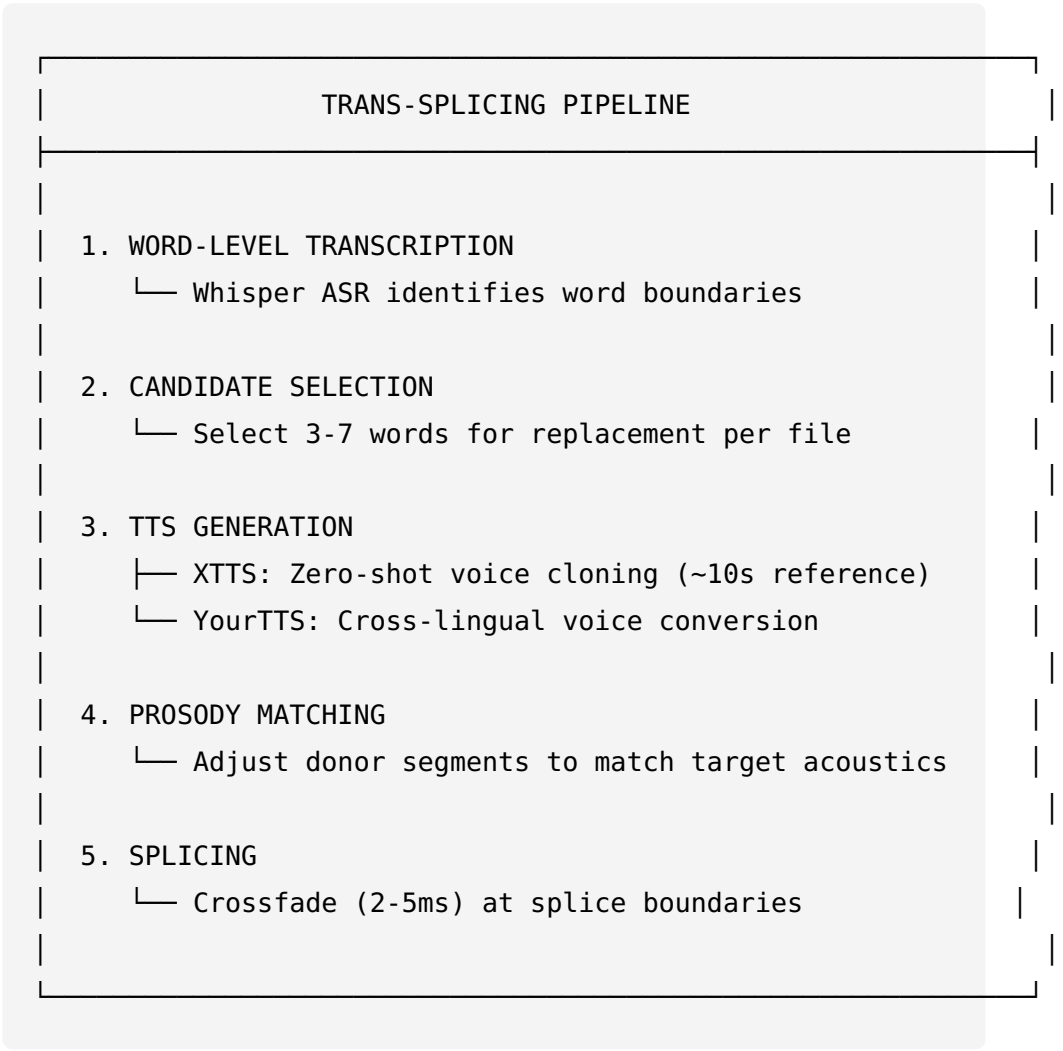
Parameters: ~340M total (mostly frozen pre-trained weights)

---

5. Audio Tampering Methodology

5.1 Trans-Splicing Pipeline

Trans-splicing creates tampered audio by replacing specific words with TTS-generated alternatives:



5.2 TTS Systems Used

System	Description	Voice Cloning Method
XTTS	Coqui X-TTS multilingual model	Zero-shot from ~10s reference audio
YourTTS	Multi-speaker TTS (Casanova et al., 2022)	Cross-lingual voice conversion

5.3 Phoneme-Boundary Editing

For semantic tampering, edits are made at phoneme boundaries to minimize artifacts:

Tools Used: - **Whisper ASR**: Word-level transcription with timestamps - **Montreal Forced Aligner (MFA)**: Phoneme-level

alignment - **spaCy NLP**: Part-of-speech tagging and dependency parsing - **librosa**: Prosody analysis (F0, energy, duration)

Tampering Operations: | Operation | Description | Example | |-----------|-------------|---------| | Deletion | Remove adjectives/adverbs | "always on" → "on" | | Insertion | Add negations | "I agree" → "I not agree" | | Substitution | Swap quantifiers | "all done" → "some done" |

---

6. Training Methodology

6.1 AASIST Training

Configuration:

```
Dataset: ASVspoof 2019 LA
Batch Size: 24
Epochs: 100
Loss: Weighted Cross-Entropy [0.1, 0.9]
Optimizer: Adam
Learning Rate: 1e-4 → 5e-6 (cosine annealing)
Weight Decay: 1e-4
Input Length: 64,600 samples (~4 seconds)
Hardware: NVIDIA RTX 4080 (16GB)
Training Time: ~14 hours
```

Training Results: - Best Dev EER: 0.354% (epoch 97) - Best Eval t-DCF: 0.0384 (epoch 44) - Final Model: SWA (Stochastic Weight Averaging)

6.2 XLS-R + SLS Training

Configuration:

```
Dataset: ASVspoof 2019 LA
Batch Size: 5
Epochs: 4 (early stopping)
Loss: Weighted Cross-Entropy [0.1, 0.9]
Optimizer: Adam
Learning Rate: 1e-6
Data Augmentation: RawBoost Algorithm 3 (SSI colored noise)
Input Length: 64,600 samples (~4 seconds)
Hardware: NVIDIA RTX 4080 (16GB)
Training Time: ~70 minutes
```

Training Results: - Best Model: epoch_2.pth - Training Speed: 4.6-4.8 iterations/second - Early stopping prevented overfitting

Critical Finding: Extended training (100 epochs) degrades performance: | Metric | 4 Epochs | 100 Epochs | |--------|----------|------------| | EER | 2.97% | 44.24% | | Training Loss | 0.00165 | 0.00001 |

---

7. Experimental Results

7.1 ASVspoof 2019 LA Evaluation

Model	EER (%)	min t-DCF	Notes
XLS-R + SLS	**0.26**	N/A	Best overall
AASIST (Pretrained)	0.83	0.0275	Official benchmark
AASIST (Trained)	~0.83	~0.03	Reproduced

7.2 ASVspoof 2021 LA Evaluation

Model	EER (%)	min t-DCF	Notes
XLS-R + SLS	**2.97**	**0.2674**	Paper reproduced
XLS-R (Paper Target)	2.87	N/A	Reference
AASIST (Pretrained)	50.07	0.7704	Near-random
AASIST (Trained)	48.27	0.7212	Codec failure

7.3 Per-Codec Analysis (ASVspoof 2021 LA)

Codec	XLS-R EER (%)	AASIST EER (%)	Difference
alaw	0.97	22.40	+21.43%
g722	0.72	24.98	+24.26%
gsm	2.59	34.16	+31.57%
none	0.36	22.40	+22.04%
opus	2.41	36.17	+33.75%
pstn	2.18	45.69	+43.51%
ulaw	0.81	22.48	+21.67%

7.4 Trans-Splicing Detection

Model	XTTS-Clean	XTTS-Unclean	YourTTS-Clean	YourTTS-Unclean	Overall
XLS-R + SLS	91.11%	91.73%	**99.81%**	**100.00%**	**95.45%**
AASIST (Pretrained)	13.44%	11.42%	88.62%	59.95%	42.96%
AASIST (Trained)	8.70%	9.25%	91.98%	58.12%	41.72%

7.5 Semantic Tampering (EER)

Model	EER (%)	Notes
XLS-R + SLS	44.44	Domain mismatch
AASIST (Pretrained)	48.00	Near-random
AASIST (Trained)	49.15	Near-random

---

8. Key Findings

8.1 Scientific Contributions

1. **Self-supervised pre-training is critical for robustness**
2. XLS-R maintains <3% EER across all codec conditions
3. AASIST fails on codec-distorted audio (22-46% EER)
4. **TTS-specific detection varies by system**
5. XTTS harder to detect than YourTTS for all models
6. XLS-R detects 91% XTTS vs AASIST's 10%
7. **Overfitting in XLS-R fine-tuning**
8. Early stopping at 2-4 epochs is critical

9. Extended training (100 epochs) causes catastrophic forgetting

8.2 Limitations

1. Semantic tampering dataset shows domain mismatch (different source audio)
2. Trans-splicing dataset lacks original bonafide samples for EER calculation
3. Results specific to TTS systems used (XTTS, YourTTS)

8.3 Recommendations

For Researchers: - Test models on diverse tampering techniques beyond ASVspoof - Investigate why XTTS is harder to detect than YourTTS - Develop dedicated audio forensics models for semantic tampering

For Production Systems: - Use XLS-R + SLS for audio deepfake detection - Apply early stopping (2-4 epochs) during fine-tuning - Monitor for novel TTS systems and attack vectors

---

Part II: Reproducibility Guide

9. Environment Setup

9.1 Hardware Requirements

Component	Minimum	Recommended
GPU	8GB VRAM	16GB VRAM (RTX 3080/4080)
RAM	16GB	32GB
Storage	50GB	100GB (including datasets)
CPU	4 cores	8+ cores

9.2 Software Installation

```
# Create conda environment
conda create -n deepfake_detection python=3.8
conda activate deepfake_detection

# Install PyTorch with CUDA support
pip install torch==1.12.1 torchvision==0.13.1 torchaudio==0.12.1 --extra-index

# Install dependencies
pip install soundfile librosa numpy pandas matplotlib scipy scikit-learn
pip install gradio>=3.34.0

# For AASIST
pip install torchcontrib

# For XLS-R (fairseq)
cd xls_r_sls/SLSforASVspoof-2021-DF
pip install -e fairseq-a54021305d6b3c4c5959ac9395135f63202db8f1/
```

---

10. Dataset Preparation

10.1 ASVspoof 2019 LA

```
# Download from Edinburgh DataShare
# URL: https://datashare.ed.ac.uk/handle/10283/3336

# Expected directory structure after extraction:
data/asvspoof/asvspoof2019/LA/
├── ASVspoof2019_LA_train/
│   └── flac/
├── ASVspoof2019_LA_dev/
│   └── flac/
├── ASVspoof2019_LA_eval/
│   └── flac/
└── ASVspoof2019_LA_cm_protocols/
    ├── ASVspoof2019.LA.cm.train.trn.txt
    ├── ASVspoof2019.LA.cm.dev.trl.txt
    └── ASVspoof2019.LA.cm.eval.trl.txt
```

10.2 ASVspoof 2021 LA

```
# Download from Zenodo
# Audio: https://zenodo.org/record/4837263
# Keys: https://www.asvspoof.org/index2021.html

# Expected directory structure:
data/asvspoof/asvspoof2021/
├── ASVspoof2021_LA_eval/
│   └── flac/
└── keys/
    └── LA/
        └── CM/
            └── trial_metadata.txt
```

---

11. Training Procedures

11.1 Training AASIST

```
cd aasist

# Edit config file to set your dataset path
# In config/AASIST.conf, update:
# "database_path": "/path/to/your/data/asvspoof/asvspoof2019/LA/"

# Start training
python main.py --config config/AASIST.conf

# Training will create:
# exp_result/LA_AASIST_ep100_bs24/
# ├── weights/          # Model checkpoints
# ├── metrics/          # Evaluation scores
# └── metric_log.txt    # Training log
```

Monitoring Training: - Check `metric_log.txt` for epoch-wise metrics - Best model saved when dev EER improves - Training completes in ~14 hours on RTX 4080

11.2 Training XLS-R + SLS

```
cd xls_r_sls/SLSforASVspoof-2021-DF

# Download XLS-R pretrained weights (if not present)
# xlsr2_300m.pt should be in the directory

# Edit train_LA.sh to set your paths

# Start training
bash train_LA.sh

# Training will create:
# models/model_LA_WCE_50_5_1e-06_*/
# ├── epoch_*.pth       # Checkpoints
# └── training_output_LA.log
```

Important: Stop training after 2-4 epochs (best EER achieved at epoch 2).

---

12. Evaluation Procedures

12.1 Evaluating AASIST

```
cd aasist

# Using pretrained model
python main.py --eval --config config/AASIST.conf

# Using your trained model
# Edit config to set model_path to your checkpoint
python main.py --eval --config config/AASIST.conf
```

12.2 Evaluating XLS-R + SLS

```
cd xls_r_sls/SLSforASVspoof-2021-DF

# Evaluate on ASVspoof 2021 LA
python eval_LA.py \
    --model_path best_model_4epochs_2.97EER.pth \
    --data_path /path/to/ASVspoof2021_LA_eval/flac \
    --protocol /path/to/keys/LA/CM/trial_metadata.txt

# Output: scores_LA_epoch*.txt
```

12.3 Evaluating on Tampering Datasets

```
cd tampered_evaluation

# Evaluate XLS-R on Trans-Splicing
python eval_tampered.py --model xlsr --dataset trans_splicing

# Evaluate AASIST on Trans-Splicing
python eval_tampered.py --model aasist --model_variant pretrained --dataset tra

# Evaluate all combinations
python eval_tampered.py --model all --dataset all
```

---

13. GUI Applications

13.1 AASIST Detection Interface

AASIST GUI

Launch Command:

```
cd aasist
python gradio_app.py --port 7860
# Access at http://127.0.0.1:7860
```

Features: - Upload audio or record from microphone - Select model checkpoint - View prediction scores and visualizations - Waveform and spectrogram display

13.2 AASIST Multi-Tab Interface

AASIST Multi-Tab GUI

Launch Command:

```
cd aasist
python gradio_app_multitab.py --port 7860
# Access at http://127.0.0.1:7860
```

Tabs: 1. **Single Detection**: Analyze individual audio files 2. **Batch Processing**: Process multiple files with CSV export 3. **Model Comparison**: Compare predictions from different models 4. **Training Monitor**: View training progress from log files 5. **Dataset Explorer**: Browse ASVspoof datasets

13.3 XLS-R + SLS Interface

XLS-R GUI

Launch Command:

```
cd xls_r_sls/SLSforASVspoof-2021-DF
python gradio_app.py --port 7861
# Access at http://127.0.0.1:7861
```

Features: - Upload audio or record from microphone - XLS-R model selection - Color-coded results (green=real, red=fake) - Waveform visualization

13.4 Running Both GUIs Simultaneously

```
# Terminal 1: AASIST (port 7860)
cd aasist && python gradio_app_multitab.py

# Terminal 2: XLS-R (port 7861)
cd xls_r_sls/SLSforASVspoof-2021-DF && python gradio_app.py
```

---

14. Troubleshooting

14.1 Common Issues

Issue	Solution
CUDA out of memory	Reduce batch_size in config
Model not loading	Check checkpoint path is correct
Audio format error	Convert to WAV/FLAC, 16kHz mono
Fairseq import error	Reinstall fairseq from source
Gradio version error	Update to gradio>=3.34.0

14.2 GPU Memory Requirements

Model	Batch Size	GPU Memory
AASIST	24	~8GB
AASIST	12	~4GB
XLS-R	5	~12GB
XLS-R	2	~6GB

14.3 Known Limitations

1. **XLS-R requires fairseq**: Must install specific fairseq version from source
2. **AASIST codec sensitivity**: Poor performance on codec-distorted audio
3. **GPU required**: CPU training not supported

Part III: References

15. Scientific Citations

Models

```
@article{jung2022aasist,
  title={AASIST: Audio anti-spoofing using integrated spectro-temporal graph a
  author={Jung, Jee-weon and Heo, Hee-Soo and Tak, Hemlata and Shim, Hye-jin a
  journal={IEEE/ACM Transactions on Audio, Speech, and Language Processing},
  volume={30},
  pages={1592--1603},
  year={2022}
}

@inproceedings{zhang2024audio,
  title={Audio Deepfake Detection with Self-supervised XLS-R and SLS classifie
  author={Zhang, Qishan and Wen, Shuangbing and Hu, Tao},
  booktitle={Proceedings of the 32nd ACM International Conference on Multimedi
  pages={10873--10877},
  year={2024}
}

@article{babu2021xlsr,
  title={XLS-R: Self-supervised cross-lingual speech representation learning a
  author={Babu, Arun and Wang, Changhan and Tjandra, Andros and others},
  journal={arXiv preprint arXiv:2111.09296},
  year={2021}
}
```

Datasets

```
@article{wang2020asvspoof,
  title={ASVspoof 2019: A large-scale public database of synthesized, converte
  author={Wang, Xin and Yamagishi, Junichi and Todisco, Massimiliano and other
  journal={Computer Speech \& Language},
  volume={64},
  pages={101114},
  year={2020}
}

@article{yamagishi2022asvspoof,
  title={ASVspoof 2021: Towards spoofed and deepfake speech detection in the w
  author={Yamagishi, Junichi and Wang, Xin and Todisco, Massimiliano and other
  journal={IEEE/ACM Transactions on Audio, Speech, and Language Processing},
  volume={30},
  pages={2507--2522},
  year={2022}
}
```

Evaluation Metrics

```
@inproceedings{kinnunen2018t,
  title={t-DCF: a detection cost function for the tandem assessment of spoofin
  author={Kinnunen, Tomi and Lee, Kong Aik and Delgado, H{\'e}ctor and Evans, 
  booktitle={Proc. Odyssey},
  pages={312--319},
  year={2018}
}
```

TTS Systems

```
@inproceedings{casanova2022yourtts,
  title={YourTTS: Towards Zero-Shot Multi-Speaker TTS and Zero-Shot Voice Conv
  author={Casanova, Edresson and Weber, Julian and Shulby, Christopher and Jun
  booktitle={International Conference on Machine Learning},
  pages={2709--2720},
  year={2022}
}
```

Repository Structure

```
deepfake_models/
├── aasist/                          # AASIST model implementation
│   ├── config/                      # Configuration files
│   ├── models/                      # Model architectures
│   │   └── weights/                 # Pretrained weights
│   ├── exp_result/                  # Training outputs
│   ├── main.py                      # Training script
│   ├── gradio_app.py               # Simple GUI
│   └── gradio_app_multitab.py      # Multi-tab GUI
├── xls_r_sls/                       # XLS-R + SLS implementation
│   └── SLSforASVspoof-2021-DF/
│       ├── model.py                 # Model architecture
│       ├── train_LA.sh             # Training script
│       ├── gradio_app.py           # GUI application
│       └── best_model_*.pth        # Best checkpoint
├── tampered_evaluation/             # Tampering evaluation
│   ├── trans_splicing/             # Trans-splicing dataset
│   ├── semantic/                    # Semantic tampering dataset
│   └── eval_tampered.py            # Evaluation script
├── figures/                         # Visualization outputs
└── data/                           # Datasets (not in repo)
```

Document Version: 1.0 **Last Updated**: November 2025