Audio Deepfake Detection with RawNet2: Full Documentation

Dataset Used:
(https://www.kaggle.com/datasets/mohammedabdeldayem/the-fake-or-real-dataset)

Part 1: Dataset Overview

Content: Contains human and AI-generated (fake) speech samples.

Format: .wav files (16kHz sample rate, mono).

Structure:

/training/real/

/training/fake/

/validation/real/

/validation/fake/

/testing/real/

/testing/fake/

Key Features:

Balanced classes (50% real, 50% fake).

Diverse speakers and synthesis methods (e.g., TTS, voice cloning).

Part 2: Implementation Challenges & Solutions

1. Data Preparation

Challenge: Variable audio lengths (1–5 seconds).

Solution: Standardized to 3-second clips (trim/pad with zeros).

Challenge: Background noise in some samples.

Solution: Applied amplitude normalization (librosa.util.normalize).

2. Model Training

Challenge: Overfitting on training data.

Solution: Added dropout (nn.Dropout(0.2)) and early stopping.

Challenge: Slow convergence.

Solution: Used ReduceLROnPlateau to adjust learning rate dynamically.

3. Evaluation

Challenge: Limited test samples (no ground-truth labels in original dataset).

Solution: Manually verified 50 random test files for benchmarking.

Part 3: Model Analysis

Why RawNet2?

Advantages:

Processes raw audio (no MFCC/Spectrogram dependency).

Residual blocks capture both local and global artifacts.

Attention mechanism improves focus on forgery cues.

Trade-offs:

Higher GPU memory usage vs. spectrogram-based models.

Performance Metrics

Epoch 5: Train Loss: 0.0147, Acc: 0.9953, Val Loss: 0.0132, Acc: 0.9969

Test accuracy: 0.58

Strengths

Real-Time Capable: ~8ms inference time per sample on NVIDIA T4.

Robustness: Detects artifacts across TTS and voice conversion fakes.

Limitations

Dataset Bias: Performance drops on unseen synthesis methods (e.g., ElevenLabs).

Noise Sensitivity: Struggles with low-SNR samples.

Part 4: Future Improvements

Data-Level:

Augment with synthetic noise/reverb.

Add multilingual samples (current dataset is English-only).

Model-Level:

Hybrid approach: Combine RawNet2 with LSTM for temporal modeling.

Self-supervised pretraining on larger unlabeled datasets.

Deployment:

Optimize with TensorRT for edge devices.

Build a web demo using Gradio/FastAPI.

Part 5: Reflection

1. Biggest Challenge?

Balancing speed (real-time needs) and accuracy. RawNet2's attention block added latency but improved F1-score by 5%.

2. Real-World Viability?

Pros: Works well on clean studio-quality audio.

Cons: Unstable in noisy environments (e.g., phone calls). Needs RIR (Room Impulse Response) augmentation.

3. Critical Improvements?

Data: Include adversarial examples (e.g., perturbed fakes).

Tooling: Integrate Weights & Biases for experiment tracking.

4. Production Plan

Phase 1: Deploy as a batch processing service (AWS Lambda + S3).

Phase 2: Real-time API with WebSocket streaming.

Phase 3: On-device inference using ONNX Runtime.

Appendix: Key Code

Data Loading

```
dataset = AudioDeepfakeDataset(
  root_dir="/kaggle/input/the-fake-or-real-dataset",
  target_sample_rate=16000,
  duration=3  # Seconds
```

```
)
Attention Block
self.attention = nn.Sequential(
  nn.Conv1d(512, 256, kernel_size=1),
  nn.ReLU(),
  nn.BatchNorm1d(256),
  nn.Conv1d(256, 512, kernel_size=1),
  nn.Softmax(dim=2)
)
Inference Example
def predict(audio_path):
  audio = load_audio(audio_path)  # Preprocess
  with torch.no_grad():
      output = model(audio.unsqueeze(0).to(device))
  return "Fake" if torch.argmax(output).item() == 1 else "Real"
```