## **SLIDE 1 – Title & Context**

**Title:** Detecting Fake Voices Using Transformers  
 **Subtitle:** Chapter 8 from *Fight Fraud with Machine Learning* – Ashish Ranjan Jha

**Content box:**

*“Voice-based deepfakes are a modern security challenge.  
 This session demonstrates how transformers can distinguish real vs fake voices.”*

🎨 **Insert:**

* Book cover image *(place right side, 50 % width)*

🗣️ **Speaker Notes:**

“Good [Morning/Afternoon]! Today we’ll explore Chapter 8 of *Fight Fraud with ML*, which focuses on transformer-based fake-voice detection — an emerging area of AI security.”

## **SLIDE 2 – Problem Context**

**Title:** Why Voice Faking Matters  
 **Bullets:**

* AI voice cloning enables fraud — CEO and phone-scam incidents (e.g. $243 K UK case).
* Traditional verification (PINs, OTP) ≠ sufficient.
* Goal → Detect synthetic speech patterns automatically.

🎨 **Insert:** small icon of 🎙️+🔒 *(top-right corner)*

🗣️ **Speaker Notes:**

“Fraudsters now use cloned voices to authorize transfers. We need models that can verify if a voice truly belongs to a person.”

## **SLIDE 3 – Chapter Overview**

**Title:** Scope of Chapter 8  
 **Flow bullets:**

1. Dataset Preparation (ForR & DeepVoice)
2. MFCC Feature Extraction
3. Transformer Model Design
4. Training → Validation → Testing
5. Inference on unseen voices

🎨 **Insert:** simple process diagram *(horizontal across slide)*

🗣️ “We’ll follow the same flow as the book — data, model, and evaluation.”

## **SLIDE 4 – Datasets**

**Title:** Training and Testing Data  
 **Table:**

| **Dataset** | **Description** | **Purpose** |
| --- | --- | --- |
| **FoR** | 195 K 2-sec clips (Real vs Fake) | Train / Validate |
| **DeepVoice** | Celebrities (8 real, 56 fake) | Unseen Testing |

🎨 **Insert:** screenshot 301 (dataset tree) *(bottom-right)*

🗣️ “FoR gives large-scale learning; DeepVoice checks if model generalizes.”

## **SLIDE 5 – DeepVoice Dataset Structure**

**Diagram to insert:** from image 301 (Real / Fake folders + Celeb conversion flow).  
 Place diagram center; label *‘Source voice → RVC → Target voice’*.

🗣️ “Each celebrity’s original voice is transformed into seven other voices using RVC.”

## **SLIDE 6 – Audio Feature Extraction**

**Title:** Converting Audio → MFCC  
 **Bullets:**

* MFCC = Mel Frequency Cepstral Coefficients.
* Captures speech tone patterns (20 bands).
* Steps: Mono → Resample 16 kHz → Compute MFCC.
* Output shape ≈ (161, 20).

🎨 **Insert:** MFCC heatmap image (295 bottom). *(center)*

🗣️ “Think of MFCCs as a fingerprint of the voice.”

## **SLIDE 7 – AudioDataset Class**

**Title:** Building PyTorch Dataset  
 **Code snippet:**

\_\_getitem\_\_ → fetch audio → convert to MFCC → return label

🎨 **Insert:** Screenshot (AudioDataset diagram page 290) *(left half)*

🗣️ “This class wraps data loading & feature extraction so DataLoader can batch efficiently.”

## **SLIDE 8 – Model Workflow**

**Diagram:** from image 292 (Defining → Training → Validation → Evaluation).  
 Place full figure center.

🗣️ “Figure 8.7 shows how we build and evaluate the AudioTransformer.”

## **SLIDE 9 – Model Architecture**

🎨 **Insert:** Figure 8.8 (page 294) *(left)*

**Bullets (right):**

* Input Tensor → Positional Encoding → Encoder → Linear → Output (Real/Fake).
* Encoder = Multihead Attention + LayerNorm + FeedForward.
* Positional Encoding keeps temporal order.

🗣️ “Attention layers learn which time-frequency zones indicate fakeness.”

## **SLIDE 10 – Architecture Details**

**Table:**

| **Parameter** | **Value** |
| --- | --- |
| n\_mels | 20 |
| Feedforward Dim | 64 |
| Dropout | 0.1 |
| Encoder Layers | 1 |
| Classes | 2 (Real/Fake) |

🎨 **Insert:** code screenshot (AudioTransformer definition from your photo #7). *(bottom)*

🗣️ “A lightweight yet powerful transformer for audio.”

## **SLIDE 11 – Model Code Flow**

🎨 **Insert:** code screenshot (library installation photo #3) *(top)*

**Bullets:**

1. Install torch + torchaudio + graphviz etc.
2. Define AudioTransformer Class.
3. Train for 10 epochs with Adam optimizer (lr = 0.001).

🗣️ “The notebook mirrors a typical MLOps pipeline from dependencies to model execution.”

## **SLIDE 12 – Training Loop**

🎨 **Insert:** image 297 (training loop code + annotations) *(center)*

**Highlight:** Loss ↓ from 0.48 → 0.09 ; Accuracy ↑ to 96.7 %.

🗣️ “We see steady learning without overfitting.”

## **SLIDE 13 – Save / Load Model**

**Code block:**

torch.save(model.state\_dict(),'audio\_classification\_model.pth')

model.load\_state\_dict(torch.load('audio\_classification\_model.pth'))

🎨 **Insert:** your photo #8 (save & load section) *(right)*

🗣️ “Persistent models enable re-training and deployment without loss of weights.”

## **SLIDE 14 – Model Testing**

🎨 **Insert:** image 299 (Test Accuracy 69.48 %) *(center)*

🗣️ “Test set = unseen voices. Even then ~70 % accuracy shows generalization.”

## **SLIDE 15 – Inference on Real Sample**

🎨 **Insert:** your photo #9 (ryan-original.wav) *(left)*

**Caption:** Output: *All segments → Real*

🗣️ “Model recognizes authentic voice patterns for Ryan clip.”

## **SLIDE 16 – Inference on Fake Sample**

🎨 **Insert:** your photo #2 (ryan-to-margot.wav) *(right)*

**Caption:** Output: *All segments → Fake*

🗣️ “The fake audio is correctly flagged as manipulated.”

## **SLIDE 17 – Aggregation Methods**

🎨 **Insert:** figure 305 (Majority/Mean/At-least-One) *(center)*

🗣️ “For long audio files, we aggregate segment results by majority vote or probability average.”

## **SLIDE 18 – Performance Summary**

**Table:**

| **Metric** | **Value** |
| --- | --- |
| Validation Accuracy | 96.74 % |
| Test Accuracy | 69.48 % |
| Loss Trend | Stable |
| Overfitting | None |

🎨 **Insert:** training curve figure 298 (bottom).

🗣️ “Strong validation means model learned fakeness patterns effectively.”

## **SLIDE 19 – Chapter Takeaways**

**Bullets:**

1. Transformer + MFCC = effective audio fraud detector.
2. Works on unseen voices → robust generalization.
3. Provides template for future voice-based security ML projects.

🗣️ “A compact but insightful chapter bridging deepfake audio and fraud AI.”

## **SLIDE 20 – Start / Stop / Continue**

| **Action** | **Description** |
| --- | --- |
| **Start** | Prototype Transformer on IVR recordings for fraud alerts. |
| **Stop** | Manual voice authenticity checks. |
| **Continue** | Using AI for call-center insight and tone analysis. |

🎨 Icons: 🟢 🛑 🟡 *(left margin)*

🗣️ “These actions help embed voice AI into our existing workflows.”

## **SLIDE 21 – Applicability to Our ML Models**

| **Area** | **Use Case** | **Benefit** |
| --- | --- | --- |
| IVR Fraud Detection | Detect cloned caller voices | Reduce risk |
| Agent Performance | Tone authenticity checks | Quality |
| Customer Verification | Voice ID auth | Security |

🎨 add icons 📞🎧🔐

🗣️ “We can extend this architecture into SmartRouteAI or Gamma-3.”

## **SLIDE 22 – Three Managerial Takeaways**

1. Transformer architecture is production-feasible for audio.
2. MFCC features simplify audio pre-processing.
3. Model accuracy indicates scalability to real world fraud cases.

🗣️ “We can start a PoC using our call recordings to quantify benefit.”

## **SLIDE 23 – Next Steps**

**Flow arrows:** Dataset → Feature Pipeline → Model → Validation → Deployment

🗣️ “Next, we’ll re-implement this pipeline on our data and benchmark against existing speech classifiers.”

## **SLIDE 24 – Summary**

**Title:** Closing Summary  
 **Bullets:**

* End-to-end fake voice detection pipeline implemented.
* 97 % validation, 70 % test accuracy.
* Applicable to fraud monitoring & voice authentication.

🎨 background: waveform + lock icon.

🗣️ “Transformer models are not just for text — they’re effective for audio fraud defense.”

## **SLIDE 25 – Q & A**

**Content:** “Let’s discuss how this fits into our roadmap.”  
 🎨 use background image of microphone or sound waves.

🗣️ “Open for questions — from technical implementation to integration ideas.”