## **🧩 1. Title & Objective (1 min)**

**Title:** Detecting Voice Faking Using Transformers  
 **Slide Text:**

* Chapter 8 from *Fighting Fraud with Machine Learning*
* Objective – Build AI model to detect **fake vs real** voices
* End-to-end workflow → Dataset | Feature Extraction | Transformer Model | Evaluation

**Speaker Notes:** Introduce the topic — how voice deepfakes are used in frauds and why ML-based audio forensics is critical.

**Insert Image:** None (Title slide only)

## **🧩 2. The Problem of Voice Faking (2 min)**

**Slide Text:**

* AI voice cloning → CEO frauds & social-engineering scams ($243 K case)
* Attackers mimic tone, accent, and emotion → deceptive phone calls
* Need automated detector for “real vs fake” audio signals

**Speaker Notes:** Summarize how TTS & deep-learning voice models enable realistic speech imitation. Mention how voice authentication is no longer enough.

**Insert Image:** none

## **🧩 3. Chapter Roadmap (1 min)**

**Slide Text:** This chapter covers →

1. Understanding Fake-or-Real (FoR) dataset
2. Extracting MFCC audio features
3. Training Transformer model to classify Fake vs Real
4. Testing on a different dataset (DeepVoice)

**Speaker Notes:** Explain that you’ll trace each step sequentially.

**Insert Image:** **Figure 8.1 – Steps in building a fake-voice detection system**

## **🧩 4. The Fake-or-Real Dataset (3 min)**

**Slide Text:**

* 195 K audio samples (Real + Fake) from humans & AI TTS systems
* Clips truncated to 2 s (16 kHz, mono, 64 KB WAV)
* Balanced by gender and class
* Split → Train, Validation, Test (each 50 % real/fake)

**Speaker Notes:** Explain that “Fake” = TTS-generated (e.g., Google WaveNet, DeepVoice3).  
 Real = human speech (Arctic / LJSpeech).

**Insert Image:** **Figure 8.2 – FoR dataset structure**

## **🧩 5. Dataset Breakdown (1 min)**

**Slide Text:**

| **Split** | **Real** | **Fake** |
| --- | --- | --- |
| Train | 5878 | 5878 |
| Validation | 1413 | 1413 |
| Test | 544 | 544 |

Each clip → uniform 64 KB WAV file (2 s duration).

**Speaker Notes:** Highlight dataset balance to prevent bias.

**Insert Image:** reuse portion of Fig. 8.2 (table boxes)

## **🧩 6. Loading Audio Files (1 min)**

**Slide Text (Code):**

waveform, sample\_rate = torchaudio.load(

'for-2seconds/training/fake/file\_0893.wav')

* 32 000 samples @ 16 kHz (2 s)
* Outputs: waveform tensor + sampling rate

**Speaker Notes:** Show how .wav becomes a numeric array for model input.

**Insert Image:** code snippet from p. 286 (Listing 8.1 partial)

## **🧩 7. MFCC Feature Extraction (3 min)**

**Slide Text:**

* Raw waveform too complex for ML → use MFCC ( Mel Frequency Cepstral Coefficients )
* Captures spectral shape → “how it sounds”
* Output: 161 time frames × 20 coefficients

**Speaker Notes:** Explain that MFCCs compress frequency bands to human-hearing scale (log Mel).

**Insert Image:** **Figure 8.3 – Waveform to MFCC conversion**

## **🧩 8. MFCC Extraction Code (2 min)**

**Slide Text (Code):**

import torchaudio.transforms as T

transform = T.MFCC(sample\_rate=16000,

n\_mfcc=20,

melkwargs={'hop\_length':200})

* Hop length 200 → ≈ 161 frames / 2 s
* Tensor shape: [161, 20]

**Speaker Notes:** Discuss parameter choice (n\_mfcc = 20, hop\_length).

**Insert Image:** code block from p. 288 (Listing fragment)

## **🧩 9. Creating Torch Dataset Class (3 min)**

**Slide Text:**

* Custom AudioDataset class handles train/val/test splits
* Returns (MFCC tensor, label)
* Compatible with DataLoader for batch training

**Speaker Notes:** Explain \_\_init\_\_, \_\_len\_\_, \_\_getitem\_\_ methods and data pipeline.

**Insert Image:** **Listing 8.1 + Figure 8.5 (AudioDataset flow)**

## **🧩 10. Visualizing MFCCs (1 min)**

**Slide Text:** MFCC samples show different frequency energy patterns.  
 Even minor variations can help model detect fakeness.

**Speaker Notes:** Show how visual differences exist but are subtle.

**Insert Image:** **Figure 8.6 – MFCC matrix (real vs fake)**

## **🧩 11. Transformer Model Overview (2 min)**

**Slide Text:**

* Input sequence = MFCC frames over time
* Transformer Encoder learns temporal patterns
* Linear output → Real / Fake probabilities

**Speaker Notes:** Explain how attention finds long-range dependencies.

**Insert Image:** **Figure 8.8 (Left – High-level architecture)**

## **🧩 12. Transformer Architecture in Detail (3 min)**

**Slide Text:**

* Input: ( Batch × 1000 × 20 )
* Positional Encoding adds sequence order
* Encoder layer = Self-Attention + Feed-Forward
* Average pooling → Linear → 2 classes

**Speaker Notes:** Describe internal layers — attention heads, feed-forward, dropout.

**Insert Image:** **Figure 8.8 (Right – Zoomed encoder)**

## **🧩 13. Positional Encoding (2 min)**

**Slide Text:**

* Sine & Cosine functions encode temporal positions
* Added to MFCCs → gives Transformer frame order sense

**Speaker Notes:** Show difference between raw MFCC input and position-encoded signal.

**Insert Image:** **Figure 8.9 – Positional Encoding Visualization**

## **🧩 14. Model Training Setup (2 min)**

**Slide Text (Code):**

criterion = nn.CrossEntropyLoss()

optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

num\_epochs = 10

* Binary Cross Entropy Loss for classification
* Adam optimizer → fast convergence

**Speaker Notes:** Emphasize learning rate and use of GPU (CUDA).

**Insert Image:** code snippet from p. 296

## **🧩 15. Training Results (2 min)**

**Slide Text:**

* Validation Accuracy ≈ 96 %
* Loss decreases consistently
* Model generalizes well on FoR dataset

**Speaker Notes:** Highlight overfitting avoidance and steady improvement.

**Insert Image:** **Figure 8.10 – Training Loss vs Validation Accuracy**

## **🧩 16. Model Evaluation on FoR Test Set (2 min)**

**Slide Text (Code):**

model.eval()

correct = 0

for x, y in test\_loader:

pred = model(x)

correct += (pred.argmax(1)==y).sum().item()

**Result:** Test Accuracy ≈ 69–70 %

**Speaker Notes:** Explain evaluation loop and threshold probability 0.5 for fake.

**Insert Image:** code + output (p. 299)

## **🧩 17. DeepVoice Dataset Overview (2 min)**

**Slide Text:**

* 8 American celebrities (real + RVC deepfakes)
* 56 fake clips ( 8 × 7 pairs )
* Purpose → Cross-dataset generalization

**Speaker Notes:** Introduce new dataset for testing transfer learning capability.

**Insert Image:** **Figures 8.11 & 8.12 (workflow + dataset tree)**

## **🧩 18. Deepfake Generation Process (2 min)**

**Slide Text:**

* Separate vocals & ambient sounds
* Convert Speaker A → Speaker B using RVC model
* Mix vocals back with ambient to create fake clip

**Speaker Notes:** Walk through Figure 8.13 diagram (step-by-step).

**Insert Image:** **Figure 8.13 – RVC conversion pipeline**

## **🧩 19. Processing DeepVoice Audio (3 min)**

**Slide Text (Code):**

waveform, sr = torchaudio.load(file)

mfcc = transforms.MFCC(sample\_rate=16000, n\_mfcc=20)(waveform)

* Resample → 16 kHz, mono
* Extract MFCC same as FoR dataset
* Ensures feature consistency

**Speaker Notes:** Stress importance of identical preprocessing for cross-testing.

**Insert Image:** code from p. 302 (preprocess\_audio)

## **🧩 20. Handling Long Audio Clips (3 min)**

**Slide Text:**

* Transformer input limit = 1000 frames
* Split large MFCCs → segments ≤ 1000
* Run each through model → combine results

**Speaker Notes:** Explain segmentation and chunk inference (important limitation).

**Insert Image:** **Figure 8.14 – Chunking diagram**

## **🧩 21. Aggregating Predictions (3 min)**

**Slide Text:** Aggregation Strategies →

* Majority Voting
* Mean Probability
* At least one Fake rule
* At least one Real rule

**Speaker Notes:** Compare robustness of each approach when combining segment outputs.

**Insert Image:** **Figure 8.15 – Aggregation Methods**

## **🧩 22. Chapter Summary (5 min)**

**Slide Text:**

* FoR dataset (195 K, 2-sec clips) → Transformer model ~70 % accuracy
* DeepVoice dataset tests cross-domain performance
* MFCC features + Positional Encoding capture temporal patterns
* Aggregation boosts final fake/real decision
* Transformers suitable for sequential audio analysis

**Speaker Notes:** Read key bullet points (Section 8.4 Summary).

**Insert Image:** none (optional recap collage of 8.2 + 8.8 + 8.15)

## **🧩 23. Key Takeaways & Next Steps (4 min)**

**Slide Text:**

* Transformers + MFCCs effective baseline for voice faking detection
* Future → Conformer / wav2vec2 models with self-supervision
* Augmentation (noise, reverb) & larger datasets → higher accuracy
* Deploy as API for fraud monitoring in contact centers

**Speaker Notes:** Conclude with how this approach fits real-world fraud defense pipelines.

**Insert Image:** none

## **🕒 Timing Summary**

| **Section** | **Slides** | **Time** |
| --- | --- | --- |
| Introduction | 1–3 | 5 min |
| Dataset & Features | 4–10 | 12 min |
| Transformer Model | 11–16 | 15 min |
| Cross-Dataset Evaluation | 17–21 | 15 min |
| Summary & Q&A | 22–23 | 10 min |

## **📸 All Figures Placement Checklist**

| **Figure** | **Slide** | **Content** |
| --- | --- | --- |
| 8.1 | 3 | Workflow overview |
| 8.2 | 4 | FoR dataset structure |
| 8.3 | 7 | Waveform → MFCC diagram |
| 8.5 | 9 | AudioDataset class flow |
| 8.6 | 10 | Real vs Fake MFCCs |
| 8.8 | 11–12 | Model architecture |
| 8.9 | 13 | Positional Encoding |
| 8.10 | 15 | Training curve |
| 8.11–8.12 | 17 | DeepVoice overview |
| 8.13 | 18 | RVC conversion process |
| 8.14 | 20 | Chunking for long audio |
| 8.15 | 21 | Aggregation methods |

Would you like me to now **generate the full PPTX file** (with all slide titles, bullet text, and “Insert Figure X.Y here” placeholders + speaker notes) so that you can directly edit and add the images?  
 If yes, I’ll produce it cleanly formatted and structured for one-hour delivery.