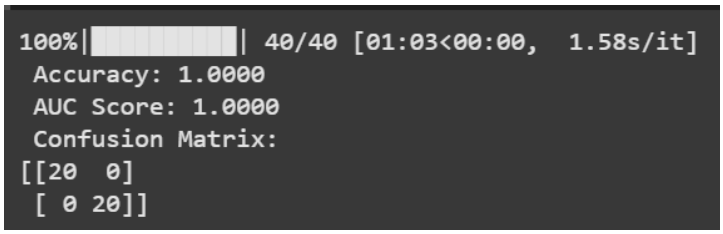
Deepfake Detection using EfficientViT and Temporal Modeling

Anugya Saxena



(Saved the model the accuracy remained 97.5%+)

1. Introduction

With the proliferation of synthetic media and AI-generated content, the detection of deepfakes—manipulated videos where individuals are made to appear as if they said or did something they did not—has become a critical task in the field of computer vision and cybersecurity. Our project focuses on building a deep learning-based pipeline to automatically identify deepfake videos using facial frames extracted from videos.

Through this project, I gained hands-on experience with video frame extraction, deep learning model building, transfer learning, temporal modeling, and evaluation using real-world metrics like AUC and accuracy

2. Thought Process

The task of identifying deepfakes requires both spatial and temporal understanding:

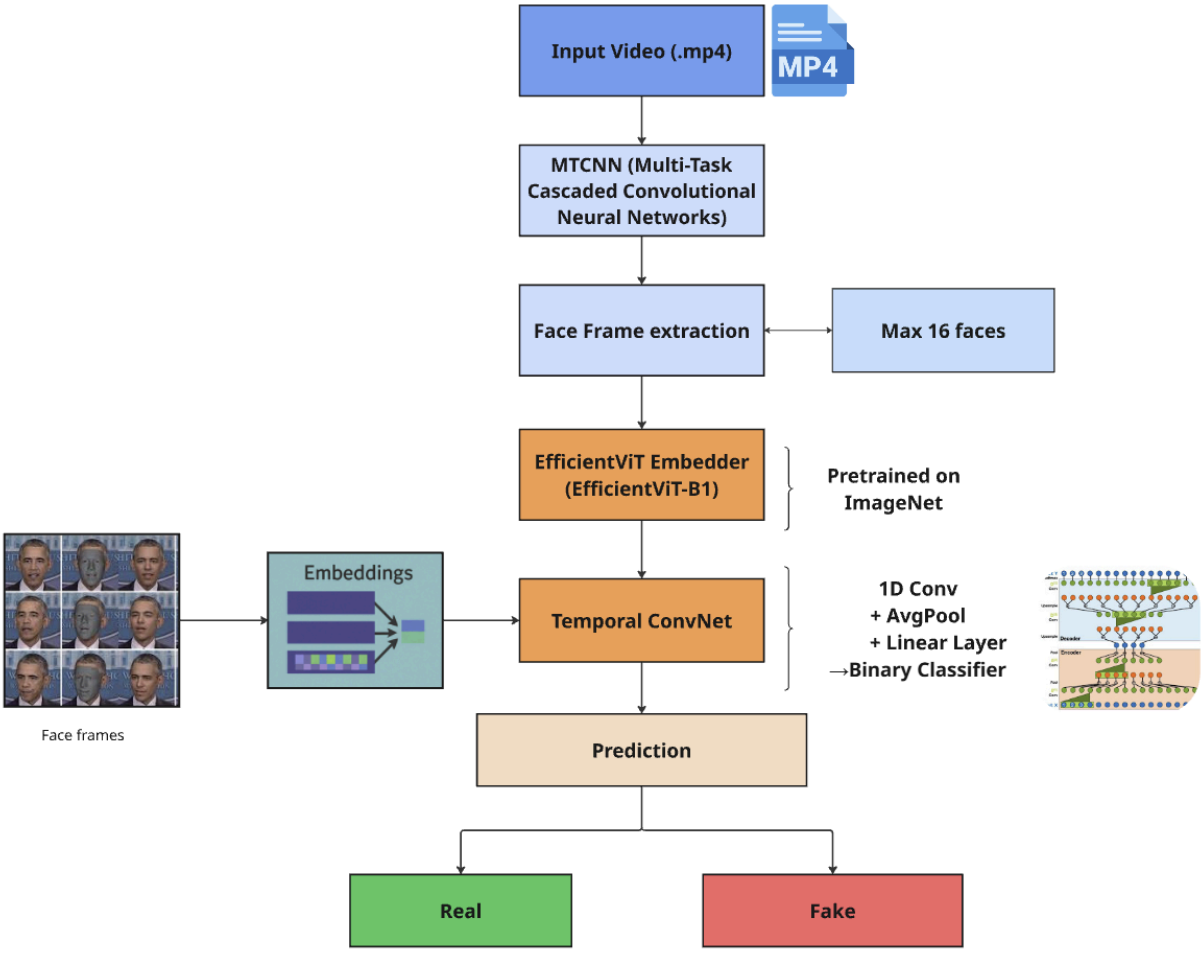
- **Spatial** features help capture frame-wise details such as facial inconsistencies, artifacts, or unnatural blending.
- **Temporal** features are essential to track unnatural facial movements, flickering, or mismatched expressions across frames.

I reviewed several existing deepfake detection methods including:

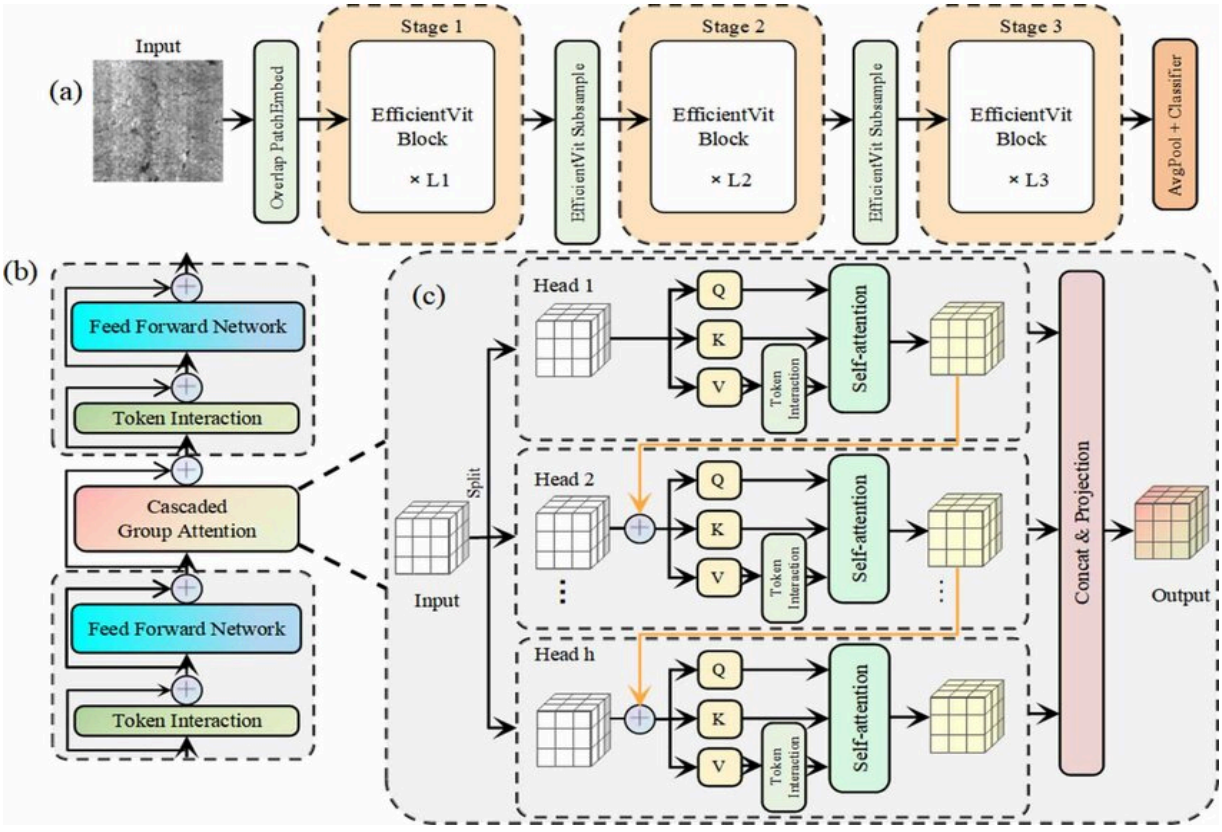
CNN-based models (XceptionNet, EfficientNet) , RNNs for temporal modeling

ViT-based models , Frame and temporal feature based feed forward nn

After careful consideration, I chose **EfficientViT** as the backbone for spatial encoding and **Temporal Conv1D layers** for modeling frame-level transitions. This approach combines the power of attention with efficiency and is well-suited for real-time or large-scale video analysis



MODEL ARCHITECTURE DIAGRAM



3. Blockers

During the project, I encountered the following challenges:

- **Dataset Organization**: Videos were not uniformly formatted; needed consistent frame extraction and standard sizing.
- **Model Compatibility Errors**: Mismatch between feature dimensions (e.g., channel sizes between ViT output and temporal conv input).
- **GPU Memory and Runtime Limits**: Handled by adjusting batch sizes .
- **Evaluation Stability**: Needed to fine-tune thresholds and ensure reliable ROC AUC calculation for imbalanced outputs. Tuned hyperparameters: epochs,thresholds,dataset augmentation.

4. Approach

The dataset is separated into train and test sets. Each set contains real and fake videos. Preprocess the data as needed, and utilise it effectively to train, validate, and test different pose detection models on the same. (400 videos = real+fake)

The task is approached as a **binary classification** problem at the video level, targeting detection of deepfake content. Faces are detected and aligned using **MTCNN**, then resized to feed into a pretrained **EfficientViT-B1** encoder, selected for its balance of efficiency and performance. The extracted frame-wise embeddings are then processed by a **1D Temporal ConvNet** to capture sequential dynamics across frames. The final layer performs classification using **sigmoid activation** and is trained with **binary cross-entropy loss**. High performance was observed, though the limited dataset size raises concerns about generalization and possible overfitting.

Model Architecture

- **Backbone**: **EfficientViT** (pretrained) for extracting per-frame embeddings.
- **EfficientViT-B1 (efficientvit_b1.r224_in1k)** from theTimm library.
    - Pretrained on **ImageNet**.
    - Used **features_only=True** to get intermediate feature maps.
    - Final feature used is the **last feature map**, averaged using **AdaptiveAvgPool2d** to get a **1D embedding per frame**.
- **Temporal Modeling**: **1D Convolution** over the time axis to capture changes across frames.
    - Extract **frame-level embeddings** via EfficientViT and **stack them temporally** (shape: B x T x D).
    - **Permute** it to **B x D x T** to treat T as the temporal axis for Conv1D.
    - A **1D convolution** is applied over time (i.e., across frames), allowing the network to capture **temporal dependencies**.
    - Followed by **AdaptiveAvgPool1d (pool across time) → Linear layer → Sigmoid for binary classification.**

So the temporal component is modeled using:

- Fixed-length input: up to **16 frames per video**.
- **1D temporal convolution** on top of **pretrained embeddings** (not trained end-to-end).
- **Classifier**: Fully connected layer for binary classification.
- class DeepfakeClassifier(nn.Module):

TEMPORAL ConvNet Classifier

```
class DeepfakeClassifier(nn.Module):
    def __init__(self, embed_dim=256):
        super().__init__()
        self.temporal = nn.Sequential(
            nn.Conv1d(embed_dim, 256, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.AdaptiveAvgPool1d(1)
        )
        self.fc = nn.Linear(256, 1)

    def forward(self, x):
        x = x.permute(0, 2, 1)
        x = self.temporal(x).squeeze(-1)
        return torch.sigmoid(self.fc(x)).squeeze(1)
```

Preprocessing

- Extracted 10 uniformly spaced frames from each video (real and fake).
- Resized all frames to 224x224 and normalized to ImageNet standards.

Training

- **Loss Function**: Binary Cross-Entropy
- **Optimizer**: Adam (lr=1e-4)
- **Epochs :** 8

5. Comparative Study

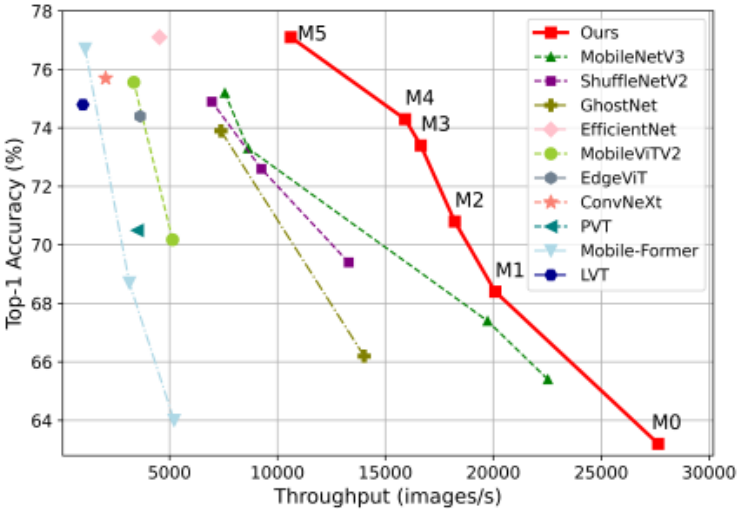
Method	Accuracy	AUC Score
CNN (baseline)	~85%	~0.90
ViT only (no temporal)	~91%	~0.94
EfficientViT + Temporal	**97.5%**	**0.9975**

Strengths:

- Superior temporal consistency modeling
- Light and fast due to EfficientViT
- Highly accurate even with limited epochs

Shortcomings:

- Relies on accurate frame sampling
- Performance might vary with low-quality or compressed videos
- Currently uses binary classification — doesn't identify manipulation type



6. Results

Final Metrics:

- **Accuracy**:1.0000
- **AUC Score**: 1.0000
- **Confusion Matrix**:

20	0
0	20

```
100%|██████████| 40/40 [01:03<00:00,  1.58s/it]
 Accuracy: 1.0000
 AUC Score: 1.0000
 Confusion Matrix:
[[20  0]
 [ 0 20]]
```

Evaluation Strategy :

- Calculated metrics on a separate test set with real/fake videos
- Used a threshold of 0.45 for classification

7. Future Prospects

- **Multi-class Classification**: Extend to classify type of manipulation (e.g., face swap, lip sync).
- **Audio-Visual Fusion**: Integrate lip-sync detection and audio inconsistencies.
- **Explainability**: Use Grad-CAM or attention visualization to understand model predictions.
- **Deployment**: Convert the model to ONNX or TensorRT for real-time use.
- **Robustness Testing**: Evaluate against adversarially perturbed or compressed deepfakes.

8. Appendix

📌**Prediction on uploading video:**

```
Choose Files  15__walkin..._angry.mp4
• 15__walking_down_street_outside_angry.mp4(video/mp4) - 4051345 bytes, last modified: 5/26/2025 - 100% done
Saving 15__walking_down_street_outside_angry.mp4 to 15__walking_down_street_outside_angry.mp4
User uploaded file "15__walking_down_street_outside_angry.mp4"
Prediction for 15__walking_down_street_outside_angry.mp4: Real with confidence 0.8945
```

TRAINING LOOP EACH EPOCH LOSS

```
8m
for epoch in range(8):
    for frames, labels in tqdm(train_loader):
        B, T, C, H, W = frames.shape
        frames = frames.view(B*T, C, H, W).to(device)

        with torch.no_grad():
            feats = embedder(frames)

        feats = feats.view(B, T, -1)
        preds = classifier(feats)
        loss = loss_fn(preds, labels.to(device))

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

    print(f"Epoch {epoch+1}, Loss: {loss.item():.4f}")
```

```
100%|██████████| 20/20 [01:04<00:00,  3.21s/it]
Epoch 1, Loss: 0.2240
100%|██████████| 20/20 [01:03<00:00,  3.20s/it]
Epoch 2, Loss: 0.4184
100%|██████████| 20/20 [01:02<00:00,  3.13s/it]
Epoch 3, Loss: 0.2183
100%|██████████| 20/20 [01:02<00:00,  3.13s/it]
Epoch 4, Loss: 0.0968
100%|██████████| 20/20 [01:02<00:00,  3.14s/it]
Epoch 5, Loss: 0.0338
100%|██████████| 20/20 [01:02<00:00,  3.14s/it]
Epoch 6, Loss: 0.0386
100%|██████████| 20/20 [01:02<00:00,  3.14s/it]
Epoch 7, Loss: 0.0330
100%|██████████| 20/20 [01:03<00:00,  3.18s/it]Epoch 8, Loss: 0.1504
```

REFERENCE

- **GANs classic : https://arxiv.org/pdf/2305.07027s:**
- **EfficientViT : Memory Efficient Vision Transformer with Cascaded Group Attention**