Truth Behind DeepFakes(TBD)
GROUP - 9

CS 776 : Deep Learning For Computer Vision

Department of Computer Science and Engineering
Indian Institute of Technology Kanpur (IITK)

Expected Outcome and deliverables from Initial Presentation

- Taking computational capacity into account, we will design a ~~Streamlit~~ Gradio based frontend for the detection of Deep Fake images and videos.

- **Phase 1: Deep Fake Image Detection**
  - Utilize a CNN and train a classifier on the publicly available 140k Real and Fake Faces dataset from Kaggle. (~~DenseNET~~/~~VGGFace~~/~~XceptionNet~~/~~Custom CNN Architectures~~/Efficientnet_b0).
  - Conduct robustness analysis to further evaluate the model.

- **Phase 2: Deep Fake Video Detection**
  - Initially employ the ~~mini_face_forensics~~ used original FF++ compressed c23 LQ dataset from Kaggle for model training and evaluation.
  - We'll explore multiple model architectures to capture both spatial and temporal features.:
    - CNN + LSTM combination.
    - ViTs, such as ~~CViT~~/Swin Transformer/~~FasterViT~~
    - Model Ensembling
  - Extract features using state-of-the-art pretrained models such as ~~InceptionV3 or XceptionNet~~ used Efficientnet_b0 and ResNext50 for transfer learning. The pretrained model will be used to obtain a feature vector, further the LSTM layers will be trained using these features and create a baseline model.

- Implementation Strategy:
  - Integrate various techniques from research papers to improve baseline model performance.
  - If feasible, apply the finalized model to a state-of-the-art Faceforensics++ dataset for further validation.
- Deliverables:
  - A web-based interface for Deep Fake detection with Python scripts.
  - Report documenting the implementation.

Deepfake Image Detection

- Deepfake technology is advancing rapidly, posing serious threats to media trust and security.
- Build a model to classify fake image as Real or Fake

Dataset Details

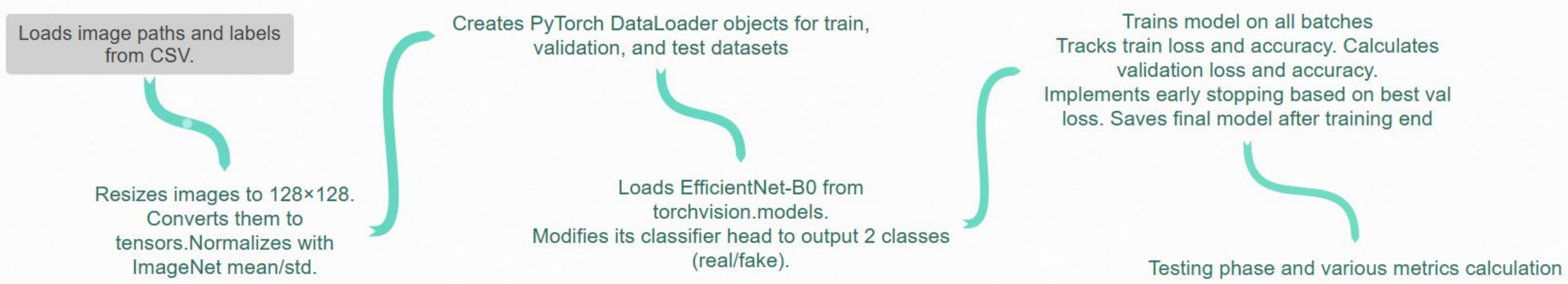
- **Source**: 140K real and fake face images from Kaggle
- **Subset Used**: 80K images total
  - Training : We have used 30k Fake and 30k Real Image.
  - Validation : 10K images
  - Testing : 10k images

Preprocessing Pipeline

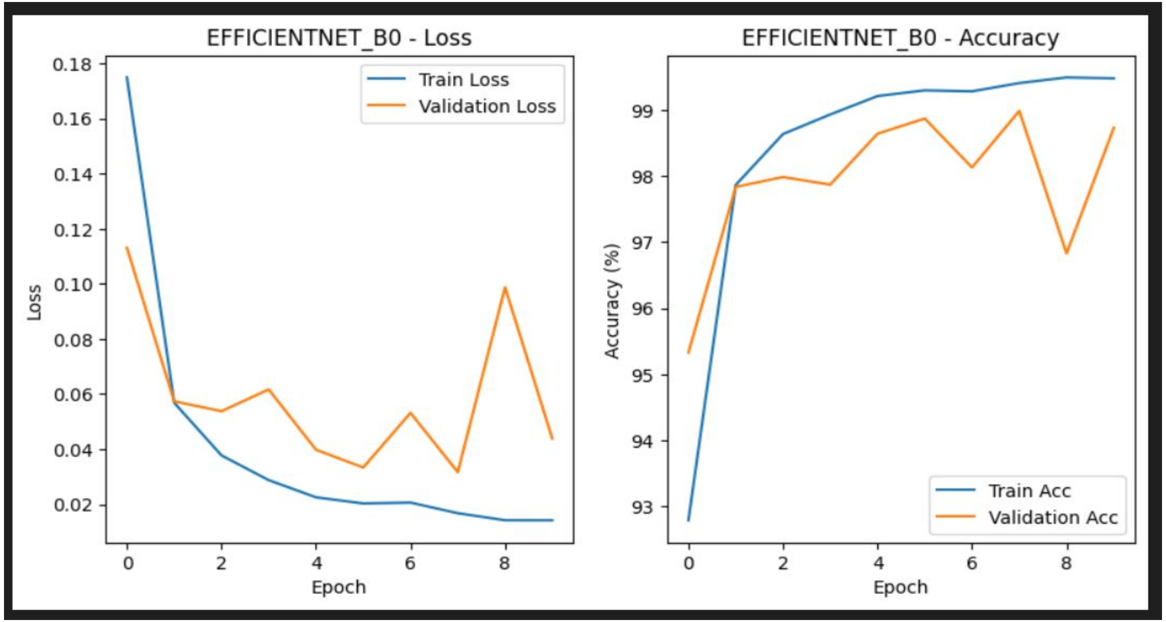
- To ensure consistency and optimize model performance, the following preprocessing steps were applied
- **Resizing** : All images were resized to a fixed dimension of **128 × 128 pixels** to maintain uniformity in input size.
- **Normalization** : Pixel values were normalized using ImageNet statistics
  - Mean [0.485, 0.486, 0.406]
  - Standard Deviation : [0.229, 0.224, 0.225]
  - This helps align the data distribution with what the pretrained EfficientNet-B0 model expects.

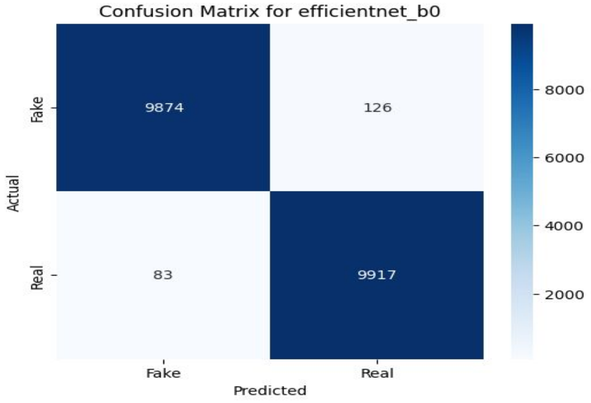
Training Configuration

- Base Model: EfficientNet-B0
  Pretrained on ImageNet (1M+ images, 1000 classes)
- Epochs: 10
- Batch Size: 64
- Loss Function: CrossEntropyLoss
- Optimizer: Adam (lr = 0.001)
- Precision: Mixed-Precision Training (torch.amp)
- Device: GPU (CUDA-enabled if available)



Results & Performance





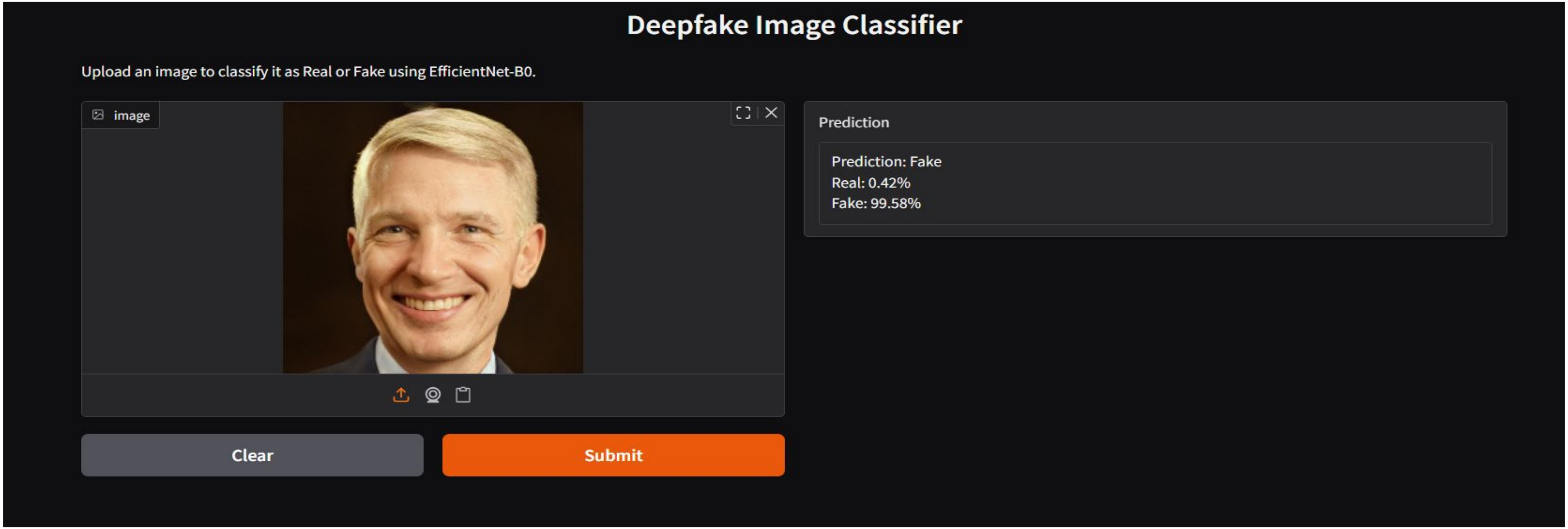
Classification Report:

	precision	recall	f1-score	support
Fake	0.99	0.99	0.99	10000
Real	0.99	0.99	0.99	10000
accuracy			0.99	20000
macro avg	0.99	0.99	0.99	20000
weighted avg	0.99	0.99	0.99	20000

Plotted **loss and accuracy curves** across 10 epochs to monitor learning behavior

Interface for Real/Fake Prediction

- Gradio provides a simple and intuitive web interface for testing our deepfake classification model in real-time.
- Upload image → Model predicts: Real / Fake



Video Deepfake Detection

Pretrained CNN's +LSTM:

- ResNeXt50/EfficientNet-B0 as feature extractor.
- LSTM for temporal aggregation.

Swin Transformer:

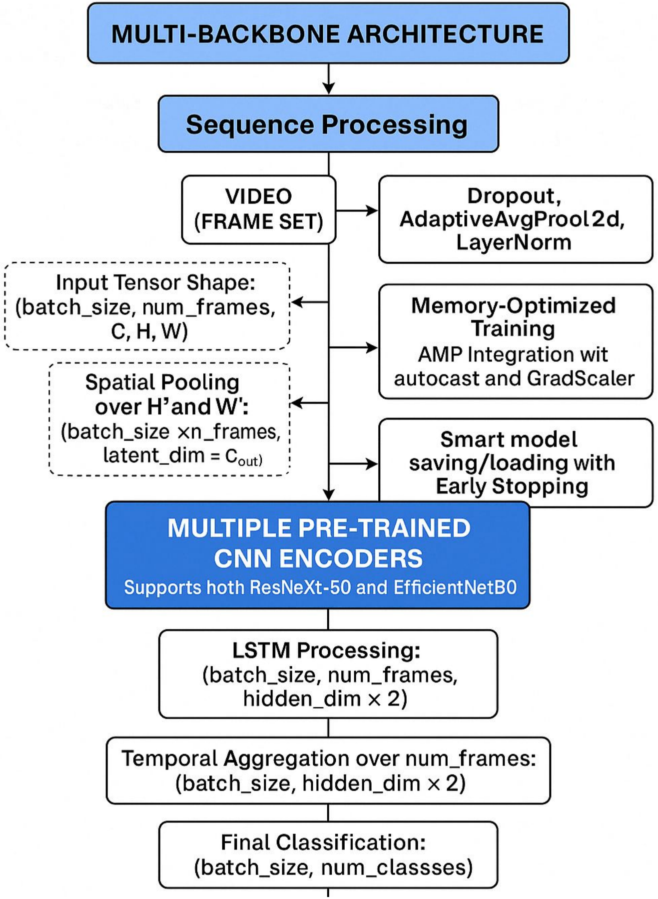
- Patch-based shifted attention for spatial features.
- Sequence averaging for temporal consistency.

Dataset Used for Train/Test/Val: FaceForensics++ c23 (LQ)

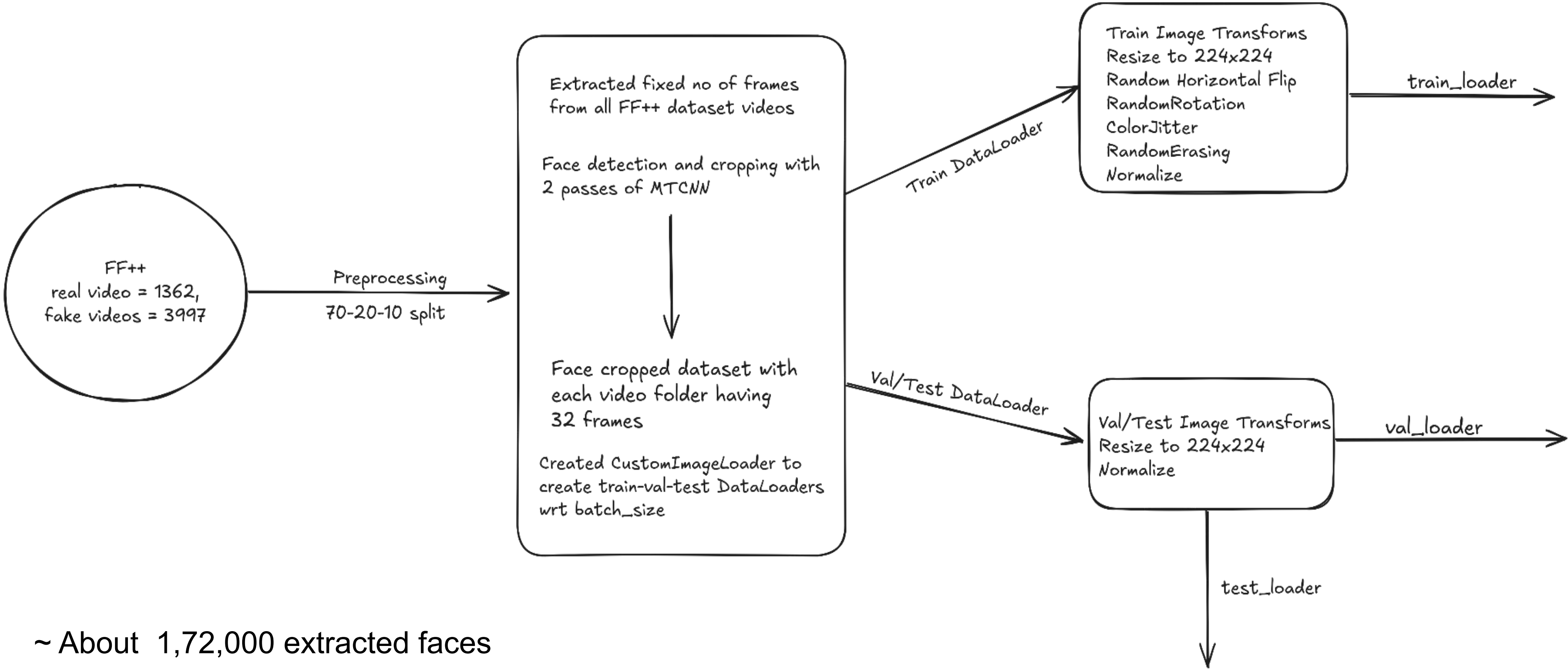
- Real
  - YT 1000 videos
  - Actor 363 videos
- Fake
  - Deepfakes 1000
  - Face2Face 1000
  - FaceShift 1000

Generalization Dataset:

- CelebDFv1
- UADFV Datset

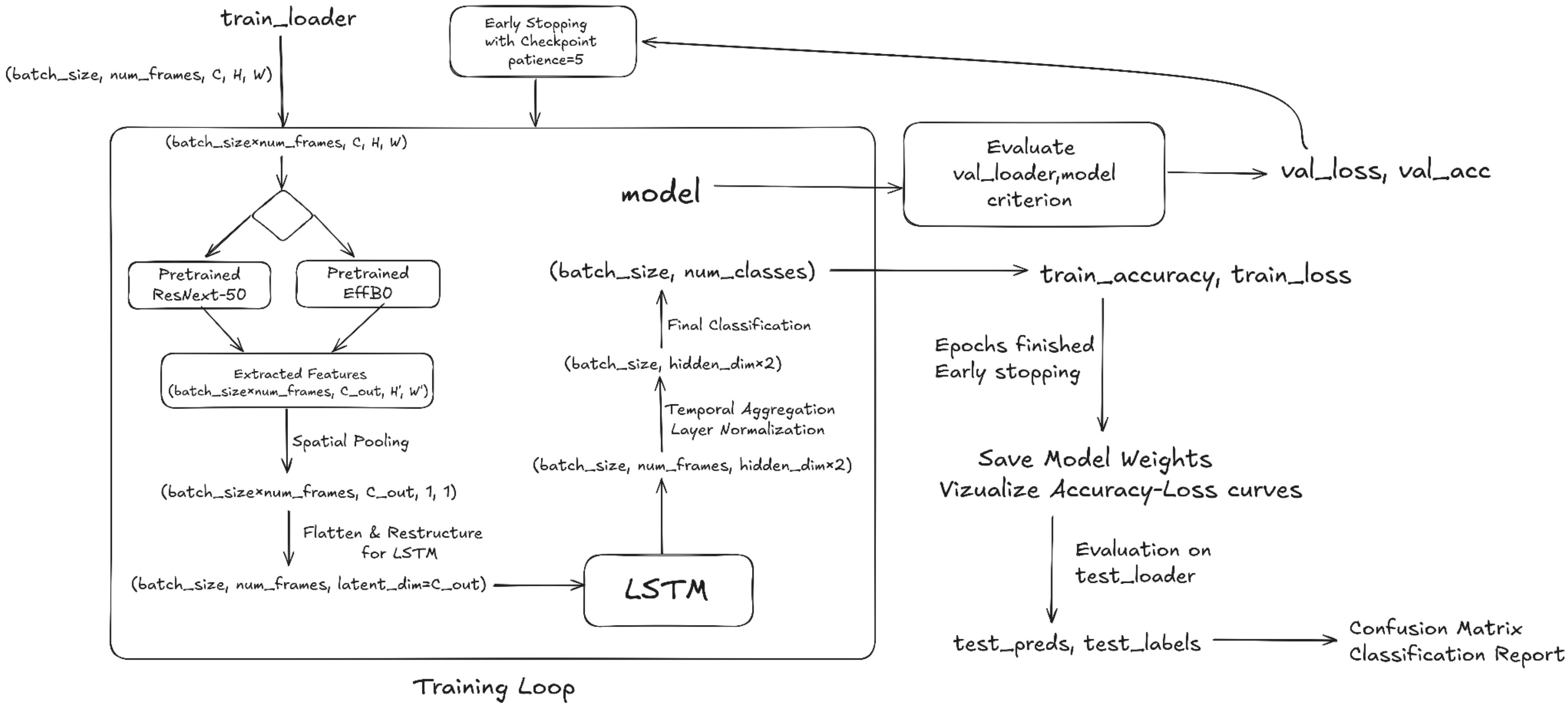


Preprocessing

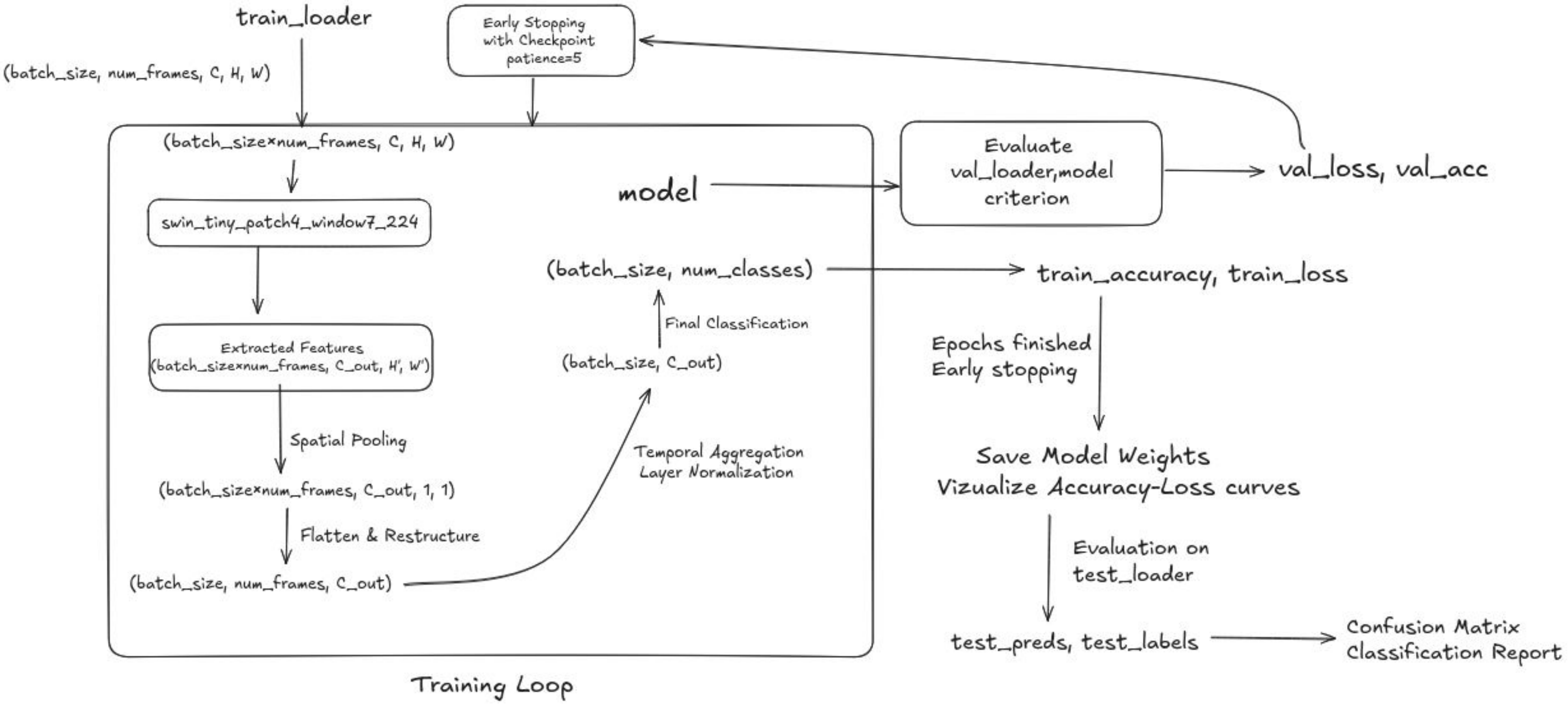


~ About 1,72,000 extracted faces

Training Workflow Pipeline I



Training Workflow Pipeline II



Project novelty

- Dual-Stage Face Verification
  - MTCNN used twice: First for detection, second for verification
  - **Why Better**: Reduces false face detections that plague many deepfake detectors. Cleaner input data than papers using single-pass detection
- Frame Sampling Strategy
  - Linear interpolation-based uniform sampling across entire video duration
  - **Why Better**: Avoids bias toward initial frames seen in many papers that simply take first N frames
  - **Benefit**: Captures temporal patterns from the entire video timeline. Dynamically adapts to videos of any length.
- Extreme Augmentation Pipeline
  - Aggressive train-time augmentations for robustness not commonly used in video deep fake detection
  - **Why Better**: Simulates real-world distortions better than basic flips/rotations.
- Optimized Class Balancing
  - Calculated class weights Based on dataset statistics for CE loss (N_total/2)/N_real
  - **Why Better**: More mathematically grounded than papers using naive sampling.
- Multi-Backbone Architecture and Hybrid Normalization Strategy
  - Combined Spatial Averaging (CNN) and temporal normalization(LSTM)
  - **Why Better**: Handles variance in both facial regions and temporal dynamics. Most papers only use BatchNorm. Combines spatial (channel-wise) and temporal (sequence-wise) normalization.

Project novelty

- Memory-Optimized Sequence Processing with LSTM
  - AMP (Automatic Mixed Precision) integration: autocast and GradScaler enabling 16-bit training without gradient underflow in temporal layers. Seeding numpy random, PyTorch, torch.cuda.manual_seed_all etc.
  - **Why Better**: Enables larger batch sizes and systematically eliminates common reproducibility pitfalls by seeding
  - **Benefit**: 2-3x faster training than typical implementations

Input Video Frames B x F x C x H x W → Reshape B*F x C x H x W → CNN B*F x Features x h x w → Pooling B*F x Features → Reshape B x F x Features → LSTM B x F x LSTM_Hidden → Mean Pool B x Hidden → Linear B x Classes

  - This architecture efficiently bridges 2D CNNs (spatial processing) and LSTMs (temporal modeling)..
- Total Time Complexity (per forward pass) CNN-LSTM: Total Floating Point Operations (FLOPs) per video
  -

$$\mathcal{T}_{\text{total}} = \underbrace{B \cdot F \cdot \mathcal{T}_{\text{CNN}}}_{\text{CNN Cost}} + \underbrace{B \cdot F \cdot \mathcal{T}_{\text{LSTM}}}_{\text{LSTM Cost}}$$

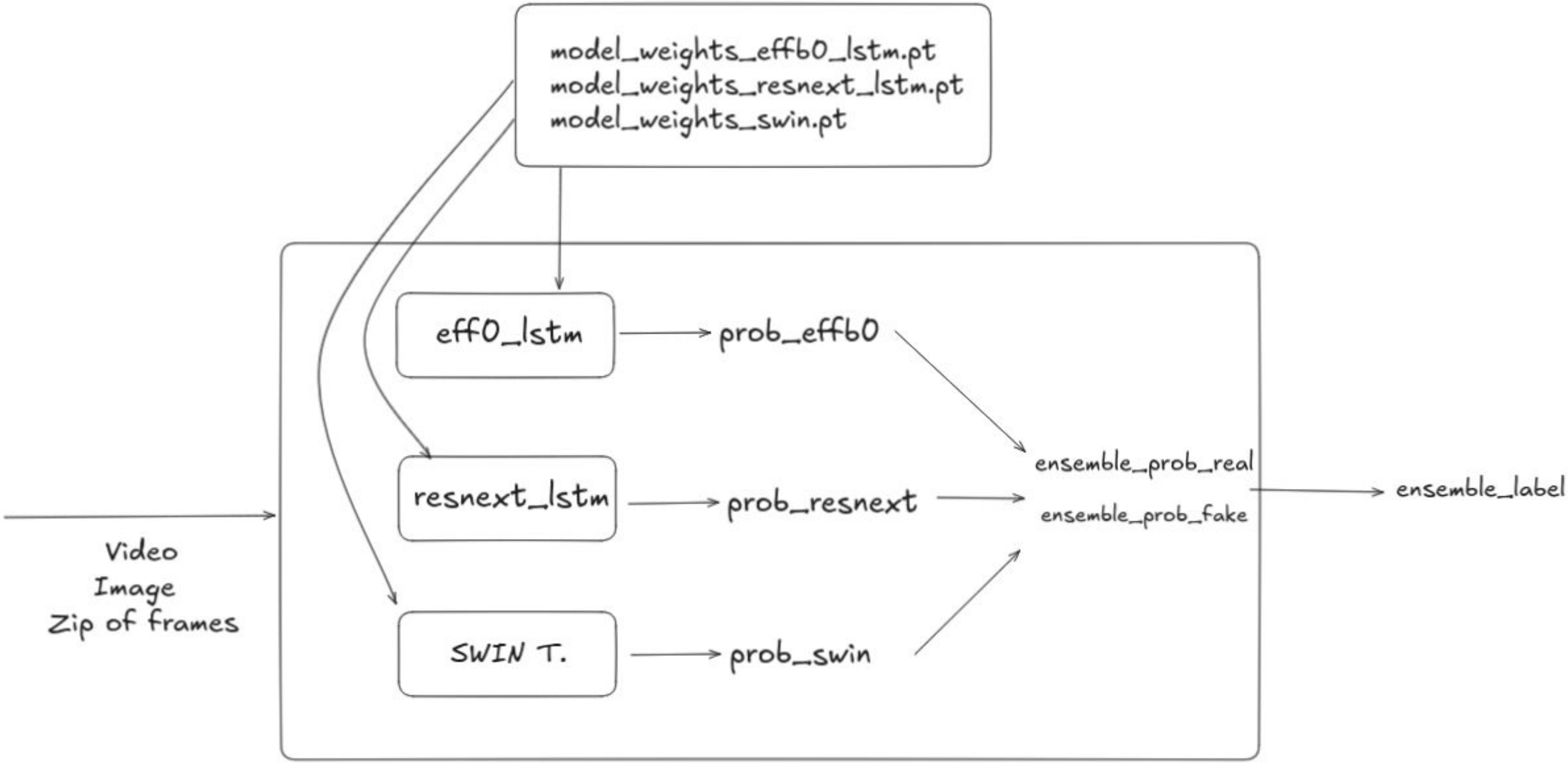
$$\mathcal{T}_{\text{CNN}} = \sum_{\ell=1}^{D} \underbrace{C_{in}^{\ell} \cdot C_{out}^{\ell} \cdot k_{\ell}^{2} \cdot H_{\ell} \cdot W_{\ell}}_{\text{Layer } \ell}$$

$$\mathcal{T}_{\text{LSTM}} = 4 \cdot \underbrace{h^2}_{\text{Hidden-Hidden}} + \underbrace{h \cdot C_{\text{out}}}_{\text{Input-Hidden}}$$

$$\mathcal{T}_{\text{total}} = B \cdot F \cdot \left(\sum_{\ell=1}^{D} C_{in}^{\ell} C_{out}^{\ell} k_{\ell}^{2} H_{\ell} W_{\ell} + 4h(h + C_{\text{out}}) \right)$$

B: Batch Size
F: No of frames used per video
D: CNN depth
h: LSTM hidden units
C_in_ℓ: Input Channels for Layer ℓ
C_out_ℓ: Output Channels for Layer ℓ
C_out: CNN Output Features

Model Ensembling/Prediction Workflow



Training Configuration

The experiments were conducted on an NVIDIA RTX 4060 Laptop GPU, which has 8GB of VRAM.

For SWIN Transformer, experiments were done on Kaggle T4 GPU with 15GB of VRAM.

Parameter	**Value**
Batch Size	2/4/8
Number of frames	10/20/30
Epochs	20
Optimizer	AdamW
Learning Rate	1e-4
Scheduler	Lr_scheduler step_size 5, gamma 0.5
Loss Function	Weighted CrossEntropyLoss
Activation Function	LeakyRelu
Early Stopping Patience	5 epochs
Dropout Rate	0.4

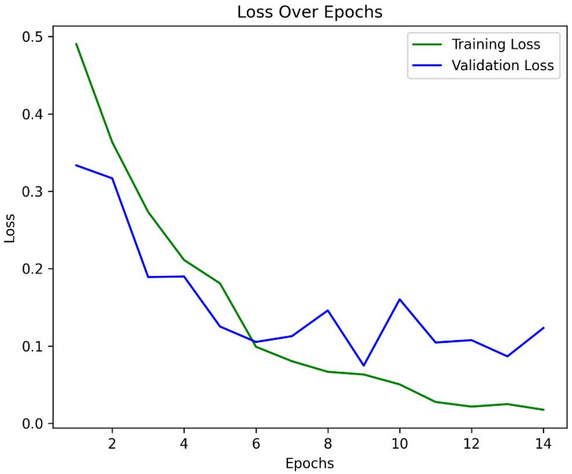
Extensively used CUDA with consistent GPU utilization (98-100%).

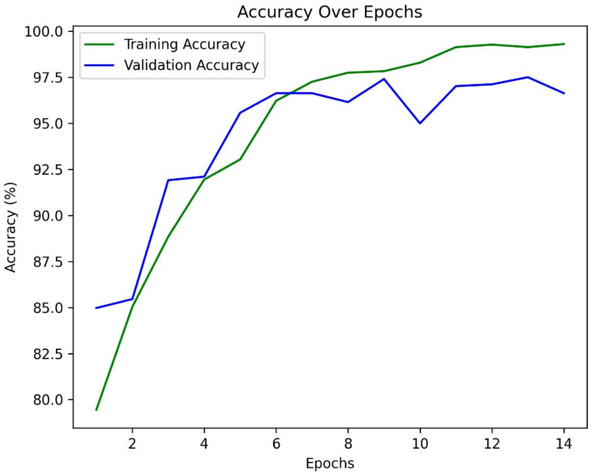
Critical VRAM overflow at sequence length 30 on RTX4060 with batch size = 4.

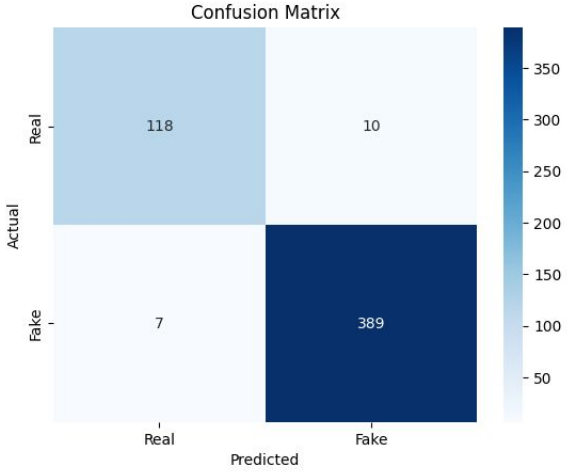
Model	#Parameters	Size
Effb0+Lstm	5.583M	23MB
ResNext+Lstm	25.342M	102MB
Swin	27.52M	110MB

Experimental Results- Efb0+Lstm

BS: 4
F: 30







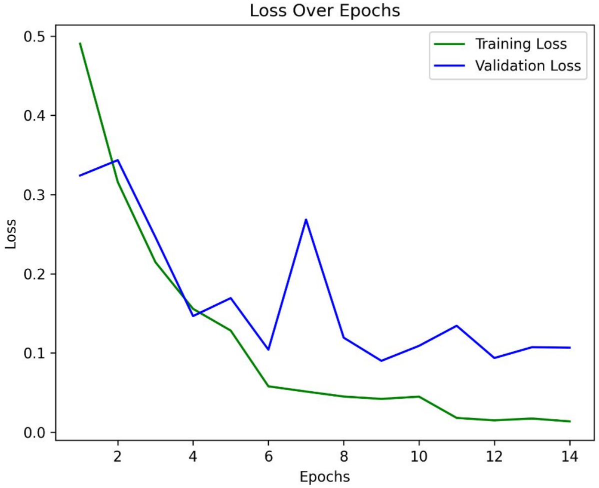
	precision	recall	f1-score	support
Real	0.9440	0.9219	0.9328	128
Fake	0.9749	0.9823	0.9786	396
accuracy			0.9676	524
macro avg	0.9595	0.9521	0.9557	524
weighted avg	0.9674	0.9676	0.9674	524

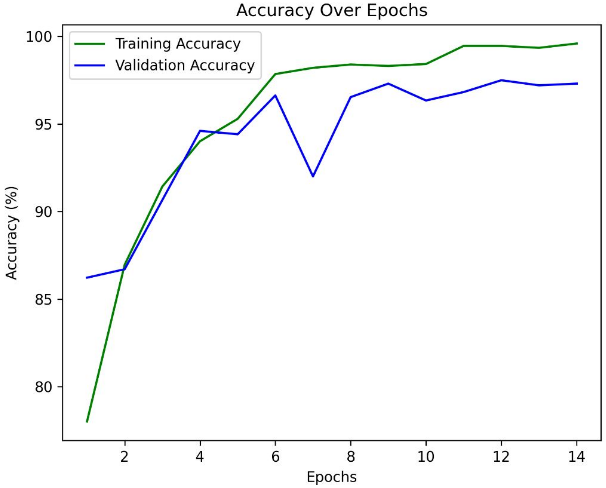
```
Training finished! Total Time: 153.738 min
```

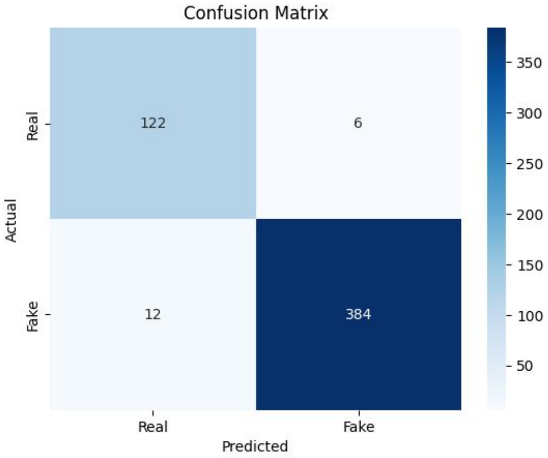
BS=8	F=10	94.79%
BS=4	F=20	96.08%
BS=4	F=30	96.76%

Experimental Results- Resnext50+Lstm

BS: 2
F: 30







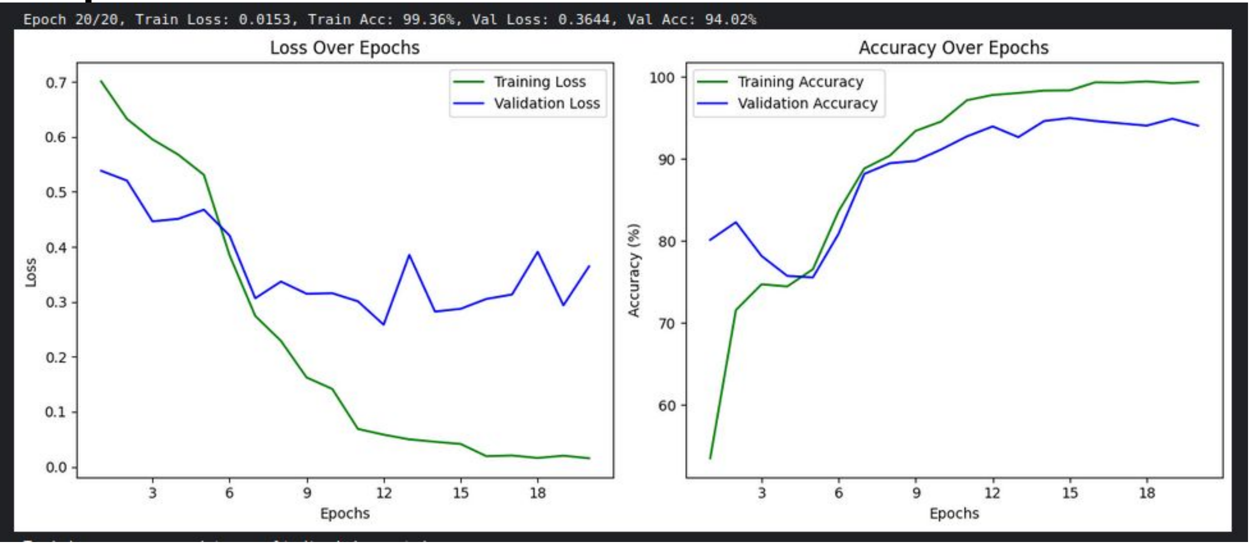
	precision	recall	f1-score	support
Real	0.9104	0.9531	0.9313	128
Fake	0.9846	0.9697	0.9771	396
accuracy			0.9656	524
macro avg	0.9475	0.9614	0.9542	524
weighted avg	0.9665	0.9656	0.9659	524

```
Training finished! Total Time: 224.652 min
==========================================================================
```

BS = 8	F=10	95.16%
BS = 4	F=20	95.52%
BS = 2	F=30	96.56%

Experimental Results - SWIN

BS : 8
num_frames=10




```
Confusion Matrix:
[[117  20]
 [ 13 387]]
Accuracy: 93.85%
```

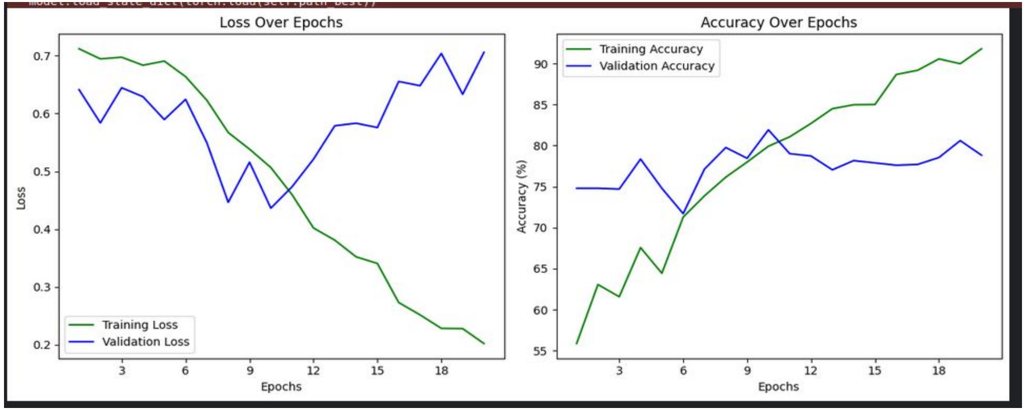
```
Training finished! Total Time: 205.298 min
```

```
Classification Report:
              precision    recall  f1-score   support

        Real     0.9000    0.8540    0.8764       137
        Fake     0.9509    0.9675    0.9591       400

    accuracy                         0.9385       537
   macro avg     0.9254    0.9108    0.9178       537
weighted avg     0.9379    0.9385    0.9380       537
```

BS=8	F=10	93.85%
BS=4	F=20	75.19%

Experimental Results - SWIN

BS : 4
num_frames=20




```
Confusion Matrix:
[[ 62  75]
 [ 58 341]]
Accuracy: 75.19%
```

```
Training finished! Total Time: 403.409 min
```

Classification Report:

	precision	recall	f1-score	support
Real	0.5167	0.4526	0.4825	137
Fake	0.8197	0.8546	0.8368	399
accuracy			0.7519	536
macro avg	0.6682	0.6536	0.6597	536
weighted avg	0.7423	0.7519	0.7462	536

Model	Parameters	Accuracy		Precision	Recall	F1 Score
EfficientNetB0	F=10 BS:8	94.79%	Real	0.90	0.92	0.91
			Fake	0.97	0.96	0.96
	F=20 BS:4	96.08%	Real	0.91	0.86	0.88
			Fake	0.95	0.97	0.96
	F=30 BS:4	96.76%	Real	0.94	0.92	0.93
			Fake	0.97	0.98	0.97
ResNext-50	F=10 BS:8	95.16%	Real	0.91	0.89	0.90
			Fake	0.96	0.97	0.96
	F=20 BS:4	95.52%	Real	0.88	0.94	0.91
			Fake	0.97	0.96	0.97
	F=30 BS:2	96.56%	Real	0.91	0.95	0.93
			Fake	0.98	0.97	0.98
SWIN	F=10 BS:8	93.85%	Real	0.90	0.85	0.87
			Fake	0.95	0.97	0.96
	F=20 BS:4	75.19%	Real	0.51	0.45	0.48
			Fake	0.81	0.85	0.84

Method	FF++ (LQ)
P3D [47]	67.05
R3D [67]	87.72
I3D [3]	93.18
M2TR $_{mean}$	93.95
ST-M2TR	**95.31**

Paper: Multi-modal Multi-scale Transformers for Deepfake Detection (2022)

Testing on other “unknown” datasets

Model	Parameters	Dataset		Precision	Recall	F1Score	Accuracy
ResNext-50	BS:2 F:10	UADFV	Real	0.825	0.97	0.89	0.88
			Fake	0.96	0.79	0.87	
		CelebDFv1	Real	0.53	0.90	0.67	0.70
			Fake	0.92	0.60	0.72	
	BS:2 F:20	UADFV	Real	0.84	0.82	0.83	0.83
			Fake	0.82	0.85	0.84	
		CelebDFv1	Real	0.51	0.90	0.65	0.67
			Fake	0.91	0.55	0.69	
	BS:2 F:30	UADFV	Real	0.96	0.93	0.95	0.95
			Fake	0.93	0.96	0.95	
		CelebDFv1	Real	0.47	0.97	0.63	0.61
			Fake	0.97	0.43	0.59	

Model	Parameters	Dataset		Precision	Recall	F1 Score	Accuracy
EfficientNetB0	BS:8 F:10	UADFV	Real	0.93	0.91	0.92	0.92
			Fake	0.91	0.94	0.92	
		CelebDFv1	Real	0.47	0.82	0.68	0.62
			Fake	0.85	0.52	0.65	
	BS:8 F:20	UADFV	Real	0.96	0.82	0.88	0.89
			Fake	0.84	0.97	0.94	
		CelebDFv1	Real	0.62	0.80	0.70	0.76
			Fake	0.88	0.74	0.80	
	BS:4 F:30	UADFV	Real	0.93	0.87	0.90	0.90
			Fake	0.88	0.93	0.90	
		CelebDFv1	Real	0.52	0.73	0.61	0.69
			Fake	0.83	0.66	0.74	

Model	Parameters	Dataset		Precision	Recall	F1 Score	Accuracy
SWIN	BS:2 F:10	UADFV	Real	1	0.85	0.92	0.92
			Fake	0.87	1	0.93	
		CelebDFV1	Real	0.47	0.70	0.56	0.63
			Fake	0.80	0.6	0.68	
	BS:2 F:20	UADFV	Real	0.96	0.85	0.90	0.91
			Fake	0.86	0.97	0.91	
		CelebDFv1	Real	0.51	0.80	0.62	0.67
			Fake	0.85	0.60	0.71	

UADFV : 95% with ResNext-LSTM_F30
CelebDFv1 : 76% with EffB0-LSTM_F16

Testing on other "unknown" datasets

Methods	Training data	UADFV [58]	Celeb-DF [31]
Two-stream [61]	Private data	85.1	55.7
Meso4 [6]	Private data	84.3	53.6
MesoInception4 [6]		82.1	49.6
HeadPose [58]	UADFV	89.0	54.8
FWA [30]	UADFV	97.4	53.8
VA-MLP [38]	Private data	70.2	48.8
VA-LogReg [38]		54.0	46.9
Multi-task [39]	FF	65.8	36.5
Xception-FF++ [42]	FF++	80.4	38.7

Paper: On the Detection of Digital Face Manipulation (2022)

Table 1: Results on DFDC test

Model	AUC
ViT with distillation [18]	0.978
Selim EfficientNet B7 [37]†	0.972
Convolutional ViT [39]	0.843
Efficient ViT (our)	0.919
Conv. Cross ViT Wodajo CNN (our)	0.925
Conv. Cross ViT Eff.Net B0 - Avg (our)	0.947
Conv. Cross ViT Eff.Net B0 - Voting (our)	0.951

Model	Mean
Convolutional ViT [39]	67%
Efficient ViT (our)	76%
Conv. Cross ViT Wodajo CNN (our)	76%

Model accuracy on FF++

Paper: Combining Efficient Net and Vision Transformers for Video Deepfake Detection (2022)

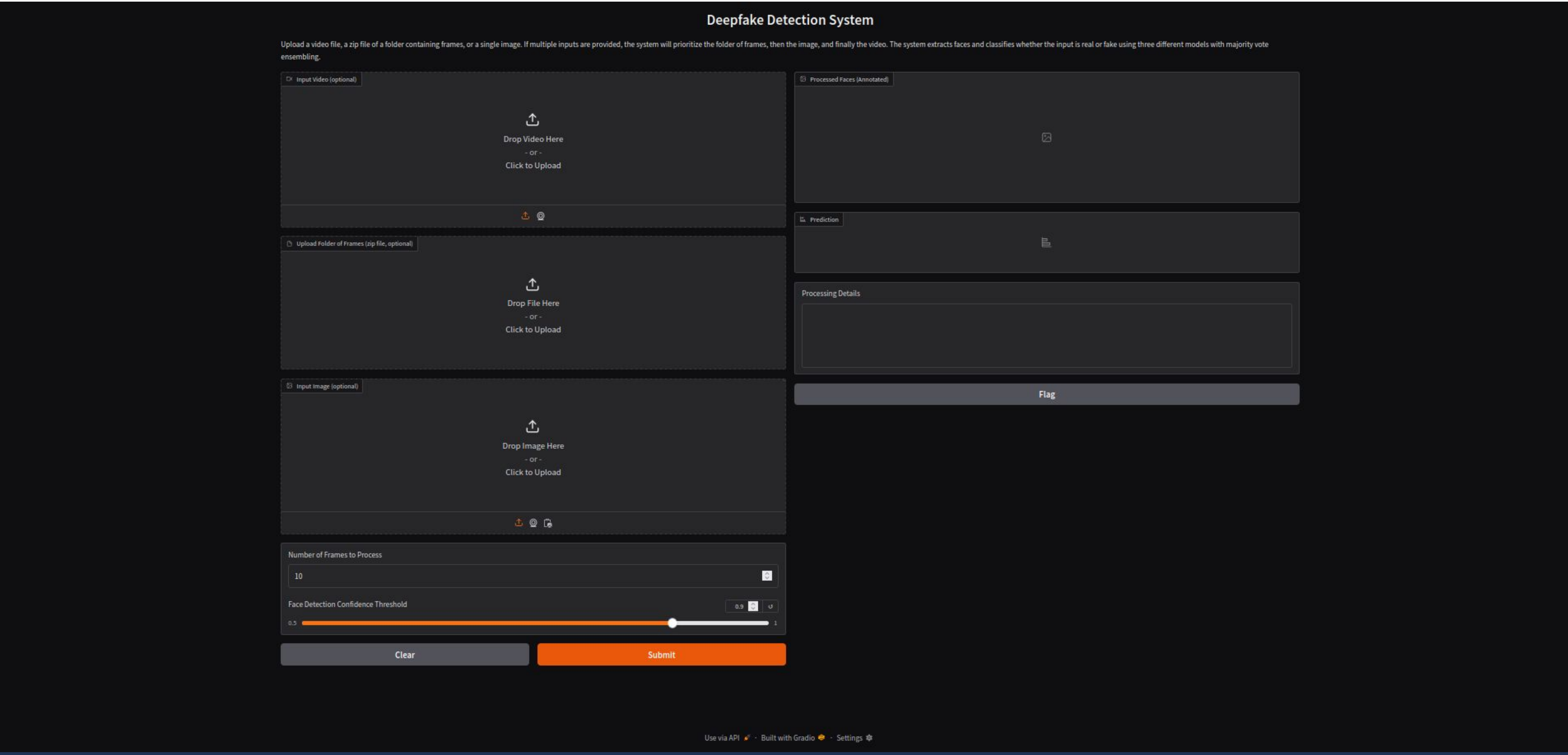
Training Set	Testing Set	M2TR $_{ncl}$	M2TR
FF++	Celeb-DF	65.6	68.2
	SR-DF	60.4	63.7

Paper: Multi-modal Multi-scale Transformers for Deepfake Detection (2022)

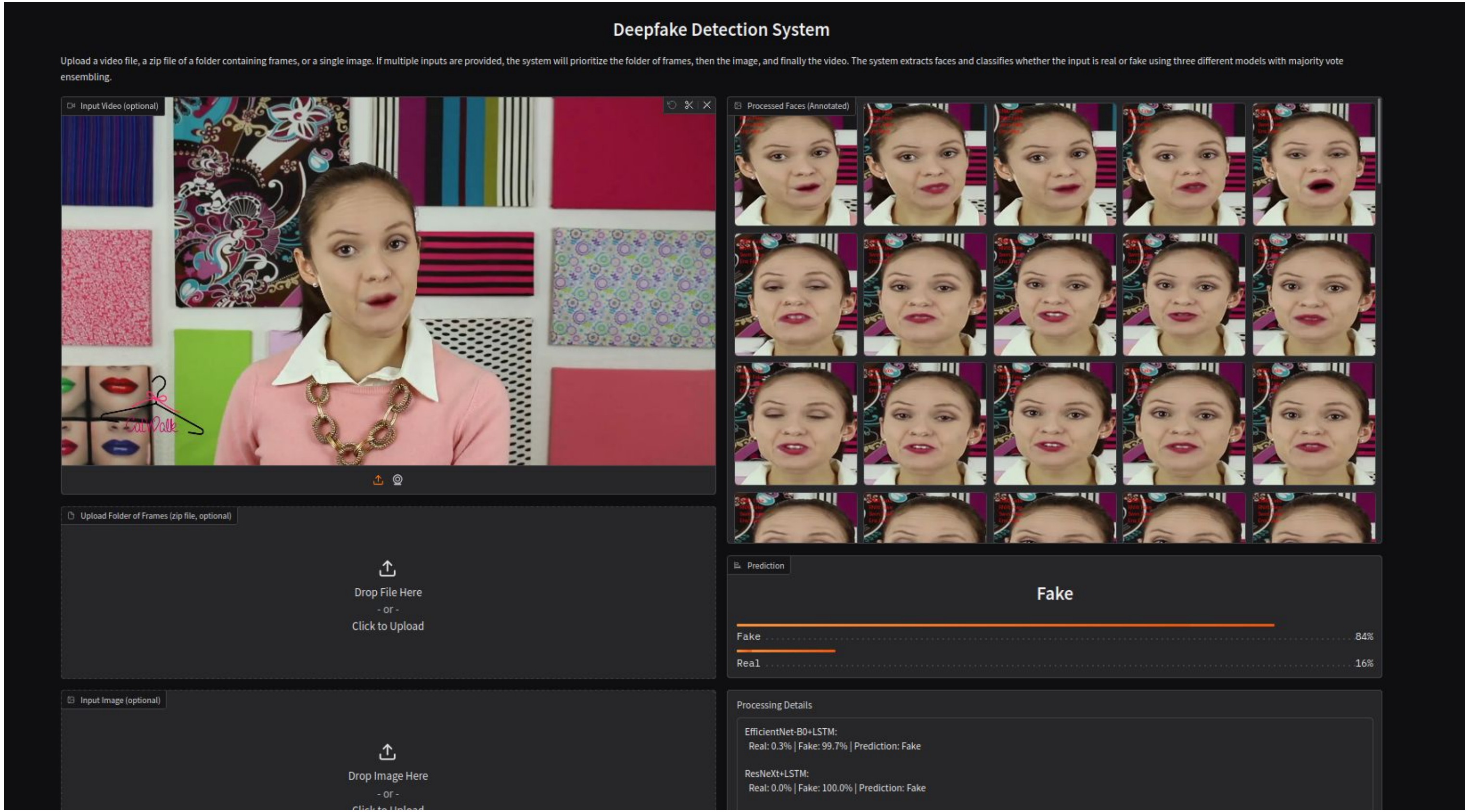
Generalisation Results for FF++ (HQ)

Training Set	Testing Set	Xception [49]	Multi-task [42]	Capsule [43]	DSW-FPA [35]	Two-Branch [39]	F3-Net [46]	MaDD [71]	DCViT [65]
FF++	FF++	99.7	76.3	96.6	93.0	98.7	98.1	99.3	98.3
	Celeb-DF	48.2	54.3	57.5	64.6	**73.4**	65.2	67.4	60.8

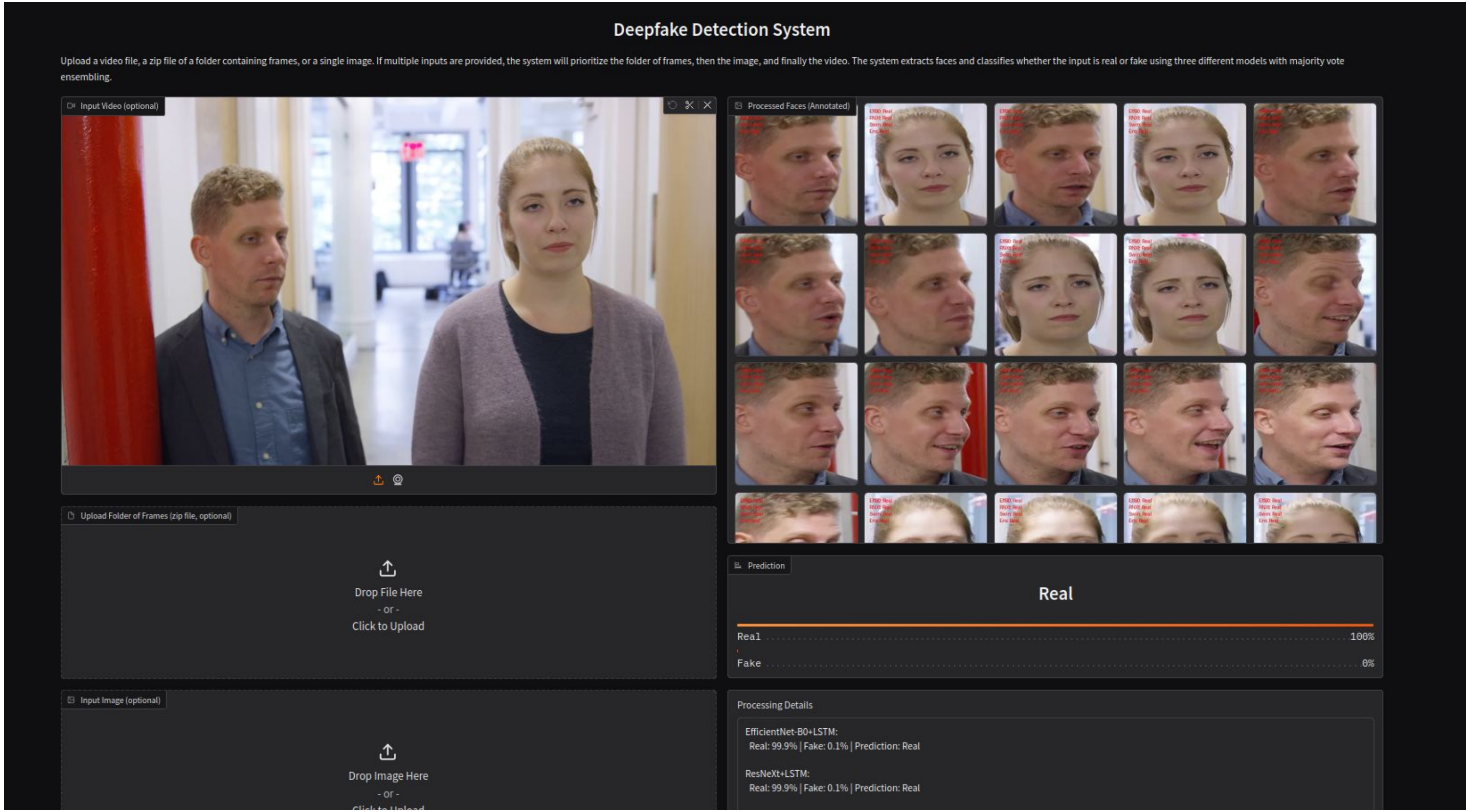
Web GUI (Model Ensembling)



Web GUI (Model Ensembling)



Web GUI (Model Ensembling)



Demo

Project Members

Name	RollNo	Tasks	%
Divyanshu	241110023	Dataset Gathering, Analysis & Preprocessing +UI	16
Khushwant	241110035	CNN+LSTM Deepfake Detection +UI & Generalisation Analysis	18
Krishanu	241110037	Hyperparameter Tuning & Testing of All Models,Noting Observations	16
Rishit	241110056	Image Deepfake Detection + UI	18
Rajan Kumar	241110087	SWIN Transformer Pipeline & Prediction Workflow	16
Senthil Ganesh	241110089	SWIN Transformer Pipeline & Prediction Workflow	16


Thank you!