# Deepfake Detection Final Report

**Chloe Humphrey, Skyler Kiefer,
Harshita Pasupuleti, Benjamin Polster, Elijah Wilkes**

Purdue University - Indianapolis

### Abstract

In a world where artificial intelligence can generate virtually anything, realistic but synthetic videos have proven to be a growing threat to trust in digital spaces. This paper proposes the use of spatiotemporal content features and style-based facial attributes to create a video deepfake detection system that can identify these manipulations. The architecture integrates a 3D ResNet-50 model for motion analysis with a StyleGRU module to capture temporal inconsistencies in facial rendering. The features are extracted with parallel pipelines,temporally fused through attention weighting before being classified by a transformer. With the model achieving 88% and 91% accuracy on Celeb-DF and Faceforensics+ datasets respectively, this work proves a new alternative for detecting deepfakes with high efficiency.

## Introduction

The rise of deepfake videos with manipulated content often spreading misinformation is causing a lack of trust among the public that requires new approaches in media authentication and verification. Temporal inconsistencies and style-based anomalies have been proven to be critical indicators of manipulation in recent work (Choi et al. 2024). However, not many systems jointly detect based on these features in an end-to-end framework. This paper proposes a video deepfake detector that addresses this gap through a model with dual feature extraction and dynamic fusion.

Our work addresses this gap through a three key contributions:

- A dual-path feature extraction pipeline that captures spatiotemporal features and facial style attributes through 3D ResNet-50 and StyleGRU respectively.
- An attention-based fusion mechanism that dynamically weights features based on their discriminative power.
- Optimized preprocessing that reduces training time by 76.7% while maintaining 88-91% accuracy on benchmark datasets

Unlike multimodal approaches, our method is more easily deployable because it solely relies on video data.



## Related Work

**Temporal Analysis in Deepfake Detection** Temporal analysis examines how features evolve across frames in a video. Coherent and smooth transitions in a variety of areas typically indicate authenticity. Frame-wise errors and inconsistent rendering, among other variables, typically constitute temporal discontinuities and are frequently artifacts of generative architecture. Traditional approaches employ RNNs to model temporal dynamics base on frame-level features. More recent approaches adopt transformer architectures to capture long-range dependencies and attention across time steps. Some models analyze temporal evolution in the latent feature space, such as style embeddings, rather than raw pixel data, to detect temporal artifacts that are visually imperceptible.

**Feature Fusion** Deepfakes can exhibit both visual anomalies and semantic discrepancies. Numerous models exist to address visual anomalies and semantic discrepancies, but combining multiple types of features (spatial, temporal, and style-based), can provide a higher level of analysis. Fusion approaches typically operate within early fusion, in which raw data or level-level features are concatenated; mid-level fusion, in which learned feature embeddings from different modalities are combined; and late fusion, in which predictions from different models are combined. Combining ResNet spatial features with style embeddings has shown improved performance by capturing style artifacts and stlye inconsistencies. Attention modules are frequently used to weigh the relevance of the involved modalities dynamically, pointing the model to the features which are most indicative of forgery. Overall, feature fusion is most appropriate for detecting visually convincing high-level generative features.

**StyleGRU** The StyleGRU model introduces a recurrent architecture that specializes in capturing temporal dependencies of style-based features, which are extracted from intermediate layers of style-generative models such as StyleGAN. Rather than focusing solely on pixel-level or frame-level artifacts, StyleGRU leverages latent space trajectories of style vectors across video frames to detect inconsistencies created during image synthesis. A gated recurrent unit (GRU) processes the sequentially extracted style vectors, learning temporal patterns that are distinctly characterized in real and forged videos. This temporal modeling of high-level

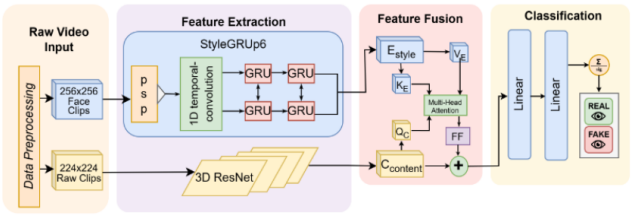


Figure 1: Model Architecture

generative cues enables the detection of manipulations that may evade traditional visual inspection or shallow CNNs. As deepfakes increasingly utilize advanced generative models, StyleGRU's focus on latent consistency over time provides a robust approach for detecting content synthesized by stylistically coherent but semantically flawed generators (Karras, Laine, and Aila 2019).

**ResNet** Residual Networks (ResNets), particularly ResNet-18 and ResNet-50, are widely used for image-based deepfake detection due to their strong capability to capture spatial-level inconsistencies within frames. Published in 2015, citations of ResNet have climbed into the thousands, establishing itself as perhaps the most popular CNN model in image recognition. ResNets are built with skip connections that alleviate the vanishing gradient problem and allow for deeper network architectures without performance degradation. With respect to deepfake detection, ResNets are often applied to individual frames or face crops to detect local visual artifacts, texture discrepancies, and illumination mismatches. ResNet-50's ability to generalize across fake generation methods makes it a reliable backbone for many frame-level detectors. Moreover, when paired with additional temporal modeling, ResNet-50 serves as an effective low-level feature extractor, providing rich embeddings that summarize spatial information before temporal analysis is applied (He et al. 2015).

## Methodology

The end-to-end deepfake detection framework (Figure 1) addresses the core challenges of temporal coherence analysis and facial rendering artifact detection through a three-stage architecture.

### System Overview

Given an input video $V \in R^{H\times W\times 3\times F}$, the pipeline consists of:

1. **Preprocessing & Face Alignment**
   - Frame sampling: Uniformly select $T = 32$ frames (shorter videos use circular padding)
   - Face extraction: MTCNN detection $\rightarrow$ 20% margin expansion $\rightarrow 256 \times 256$ crops
   - Dual normalization:

$$\text{Style path: } \mathcal{N}(\mu = 0.5, \sigma = 0.5)$$
$$\text{Content path: } \mathcal{N}(\mu = [0.45, 0.45, 0.45], \sigma = [0.225, 0.225, 0.225])$$

2. **Dual-Feature Extraction**
   - Content pathway: 3D ResNet-50 processes clips as 4D tensors $\in R^{3\times 16\times 112\times 112}$
   - Style pathway: pSp encoder extracts style vectors $\in R^{512}$ per frame
3. **Temporal Fusion & Classification**
   - Cross-attention between content and style sequences
   - Transformer encoder with learned class token

### Feature Extraction

The feature extraction process is divided into two complementary pathways: content features and style features, capturing different aspects of the video for comprehensive deepfake detection.

**Content Features** The content feature extraction employs a pretrained 3D ResNet model loaded from TorchHub. The implementation processes video frames in a [C, T, H, W] format (Channel, Time, Height, Width), which is the standard format for 3D convolutional networks. Removing the final classification layer, the content model processes the normalized frames through its convolutional layers, producing spatiotemporal features that capture motion patterns and temporal inconsistencies. The extracted raw features undergo global average pooling if needed to eliminate any remaining spatial dimensions, resulting in a compact feature representation of the video content. These content features are designed to capture the broader spatiotemporal patterns that might indicate manipulation in deepfake videos.

**Style Features** The style feature extraction utilizes a pretrained pixel2style2pixel (pSp) encoder model, focusing on facial style characteristics. Each face frame is individually processed through the pSp encoder to extract style latent vectors. The implementation specifically targets level 9 of the latent codes (out of the multiple levels produced by the encoder), which represent mid-level style information that is most relevant for deepfake detection. The style codes for each frame are extracted sequentially, creating a temporal sequence of 512-dimensional style vectors that represent how facial style attributes change over time. These style features are particularly suited for detecting inconsistencies in facial rendering that are common in deepfake videos. Both content and style features are cached to disk in a .pt format, organized into 'real' and 'fake' subdirectories, creating an efficient dataset structure for subsequent training and evaluation. This caching mechanism includes validity flags, ensuring that only successfully processed videos contribute to the training process.

### Style Attention Module for Feature Fusion

Our classifier intends to leverage anomalies over time in deepfake videos, so we fuse the extracted style-based temporal and content features into a single representation for further processing. To achieve this, we employ a style attention module that computes attention scores to combine style information. Through three separate linear projections, content features act as attention queries, and style embeddings

represent the corresponding keys and values. By taking the dot product of a query and key, normalizing by the dimensionality, and applying a softmax function, we obtain style attention weights. These weights are then used to scale the value vectors, and the result is passed on to the final stage for classification.

### Classification

The resulting value vectors from feature fusion are put through two fully connected layers, implemented in PyTorch, progressively reducing their dimensionality. The final layer outputs probabilities, using a sigmoid function to compress the outputs into the range from 0 to 1, which are then rounded for the final binary result.

## Datasets

**FaceForensics++** The primary training dataset used is FaceForensics. FaceForensics is a benchmark dataset and framework developed to support the detection of facial manipulations, particularly identity manipulation or face swapping, in which one person's face is replaced with another's. The dataset includes over 1.8 million images from thousands of videos, covering both traditional computer graphics-based techniques (e.g., Face2Face, FaceSwap) and deep learning-based methods (e.g., DeepFakes, NeuralTextures). The primary advantages of FaceForensics include: large scale and diversity, realistic conditions, its function as a standardized benchmark and its optimization for convolutional neural networks. FaceForensics provides high-quality real and fake video pairs for robust training, subjects its videos to random compression and resolution changes to mimic real-world data and provides a considerable breadth of evaluation. (Rössler et al. 2019)

**Celeb-DF** After initial training with FaceForensics++, we expanded our model with another popular dataset: Celeb-DF. Celeb-DF features 890 real videos and 5639 deepfake videos. Like FaceForensics++, Celeb-DF is a high-quality large-scale deepfake video dataset designed to aid the development of detection algorithms. Celeb-DF offers more visually-convincing deepfakes, more similar to what appears in real-world disinformation. Celeb-DF shares many advantages with FaceForensics++, including its scale, diversity, robustness benchmark, and realism. Celeb-DF's unique advantage is its high visual quality, as a result of its improved deepfake synthesis method. This visual improvement is marked by far fewer visual artifacts than other datasets. (Li et al. 2020)

## Data Preprocessing

The preprocessing stage transforms raw video inputs into standardized feature representations. The pipeline begins with frame extraction using OpenCV's video capture (Bradski 2000). Instead of extracting one frame per second, the implementation samples frames uniformly throughout the video, extracting a specified number of frames (32 frames for FF+, 64 frames for Celeb-DF) regardless of video length. For shorter videos, frames may be duplicated to reach the target number. After frame extraction, facial detection is performed using MTCNN (Multi-task Cascaded Convolutional Networks) from the FaceNet PyTorch implementation, which provides robust face detection with configurable margin settings (Zhang et al. 2016). The detected faces are then cropped and resized to a uniform 256×256 pixel resolution. If face detection fails, the system falls back to a center crop approach, ensuring processing continuity. For preprocessing standardization, the face images undergo normalization with a mean of [0.5, 0.5, 0.5] and standard deviation of [0.5, 0.5, 0.5], which is optimized for StyleGAN-based models. The video frames for content analysis are separately normalized with means of [0.45, 0.45, 0.45] and standard deviations of [0.225, 0.225, 0.225], following the Kinetics dataset statistics standard for action recognition tasks.

## Model Training

To develop a robust deepfake detection model, we focused on optimizing both model performance and training efficiency. Many improvements have been made to our training pipeline: feature extraction optimization, learning rate scheduling strategies, data augmentation techniques, and integration of training visualizations.

**Feature Extraction Optimization** Feature extraction takes around $25$ seconds for each sample on the Scholar cluster. These steps were initially implemented in the forward pass. Consequently, for $n$ samples, the time training for each epoch was compounded by $25n$ seconds. The total training time for all epochs, reaching 36+ hours on the Scholar cluster, was unfeasible.

However, pSp encoder and 3D ResNet-50 feature representations for a given video remain constant. Refactoring our pipeline, moving style and content feature extraction to data preprocessing, meant significant training improvements. With $25n$ second extractions only occurring once prior to training, the succeeding epochs are much faster. Additionally, we added a feature to store preprocessed videos for reuse. Accessing cached, training-ready samples facilitated hyperparameter fine tuning, avoiding the 10+ hour overhead.

**Learning Rate Scheduling & Early Stopping** Training the model on a combined dataset yielded disappointing results; still, it illuminated areas for improvement in our training pipeline. Initially, we implemented the adaptive learning rate technique reduce-on-plateau learning. It would monitor validation loss, reducing the learning rate when it stopped improving. Our model was quickly overfitting the training data and stagnating validation loss, so the learning rate would only ever be reduced by small amounts.

We decided instead to implement cosine annealing learning rate scheduling follows a predetermined cyclical schedule that decreases learning rate following a cosine function modeled as:

$$\eta_t = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})(1 + cos(\frac{T_{cur}}{T_i}\pi))$$

Where $\eta_t$ is the learning rate at time t, $\eta_{min}$ and $\eta_{max}$ define the range ($\frac{1}{100}\eta_{initial}$ and $\eta_{initial}$ respectively), $T_{cur}$ is

the number of epochs since the last restart, and $T_I$ is the total number of epochs for the current cycle (Loshchilov and Hutter 2016). With PyTorch's built-in cosine annealing module, we set our initial warm restart period to 10 epochs which doubles after each restart. Early stopping with a patience of 12 epochs tracks validation loss and saves the best model accordingly. With our learning and stopping configurations, the model explores different learning rates, slowing down training on a fixed schedule, and saves the best model for evaluation.

**Data Augmentation** Throughout model iterations, we consistently observed model overfitting as evidenced by stagnating validation loss. We wanted to prevent overfitting as much as possible to expectantly improve generalization; however, our limited training samples presented challenges. Exploring data augmentation methods led us to mixup. Mixup creates synthetic training samples by linearly interpolating between pairs of existing samples and their labels. In essence, it takes two training examples and "mixes" them together. The network should predict a blended output (Zhang et al. 2017).

The mixup alpha parameter controls the shape of the Beta distribution used to sample interpolation coefficient $\lambda$ $[0, 1]$. In our case, we apply mixup separately to content $(x)$ and style $(y)$ features in addition to the target vector $(z)$:

$$\tilde{x} = \lambda x_i + (1 - \lambda)x_j$$
$$\tilde{y} = \lambda y_i + (1 - \lambda)y_j$$
$$\tilde{z} = \lambda z_i + (1 - \lambda)z_j$$

Where $(x_i, y_i, z_i)$and $(x_j, y_j, z_j)$ are feature-target vectors, and $(\tilde{x}, \tilde{y}, \tilde{z})$ represents the new training sample. Within the training loop, a probability factor applies mixup (i.e. 70% of batches are mixed-up and 30% of batches are original). We intended to avoid over memorization of training samples with mixup data augmentation but saw only modest improvements.

**Training Visualizations** Quality visualizations, logging, and analysis are critical in improving the interpretability of a deep learning model. In our case, we wanted to monitor the real-time and post-training performance of our deepfake detector. We added training visualization functionality that plots key training metrics over time. This includes loss, accuracy, and learning rate changes which can be monitored real-time during training via TensorBoard.

**Training Benchmarks** We benchmarked feature extraction and training times across two environments: a local GTX 1650 Ti GPU and the Scholar cluster. We measured both the feature extraction and full model training times using our three datasets. Feature extraction has the most overhead for our model, as shown in Table 1. For the largest dataset, the 40% reduction in time on the Scholar cluster was highly valuable. Additionally, we were able to see the massive time savings achieved moving feature extraction to preprocessing. Training times reflected in Table 2 follow similar trends. The Scholar cluster completes training around 30% faster than running locally.

Table 1: Feature Extraction Time on Different Devices

| # Videos | GTX 1650 Ti | Scholar Cluster |
|---|---|---|
| 1 | 24 seconds | 14 seconds |
| 100 | 40 minutes | 23 minutes |
| 1,000 | 6.7 hours | 3.9 hours |
| 2,552 | 17 hours | 10 hours |

Table 2: Model Training Time on Different Devices

| Training Data | GTX 1650 Ti | Scholar Cluster |
|---|---|---|
| FF+ | 35 minutes | 20 minutes |
| C-DF | 1 hour | 45 minutes |
| Combined | 1.5 hours | 1 hour |

**Training Results** We evaluated the model's learning behavior for the FaceForensics++ and Celeb-DF datasets to compare performance trends and generalization capabilities. The training on FF+ (see Figure 2) shows a somewhat balanced learning curve. Training and validation losses decrease in parallel for the several epochs with minimal divergence. Both training and validation accuracy improve rapidly, both stabilizing at the end. However, model training using Celeb-DF (see Figure 3) shows more signs of volatility. There is a steady decrease in training loss with a rapid early improvement. Accuracy on the training set consistently climbs and plateaus near 99% suggesting the model learned the training distribution effectively. Whereas validation loss exhibits high variance seen as spikes throughout training (especially between epochs 10-30). This instability indicates the model is overfitting.

FaceForensics++ shows some fluctuations in validation loss, but they are less severe than Celeb-DF. Furthermore, the performance gap between training and validation accuracy is smaller. This implies better generalization when the model is trained on the FaceForensics++ dataset.

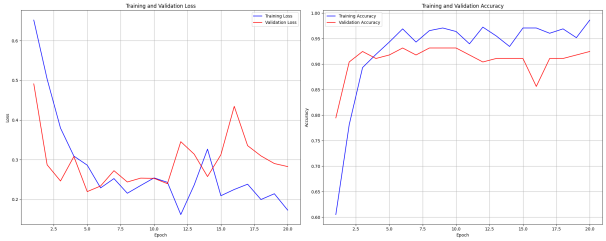


Figure 2: Training (blue) and validation (red) sets loss and accuracy graphs across 20 epochs for the FaceForensics++ dataset. Loss steadily decreases at the beginning of training and remains relatively consistent for both sets. Accuracy improves and stabilizes in the nineties with a minimal gap between training and validation sets.

## Experimental Setup

Experiments were conducted on the Purdue Scholar Cluster, a high performance computing cluster hosted by the Rosen

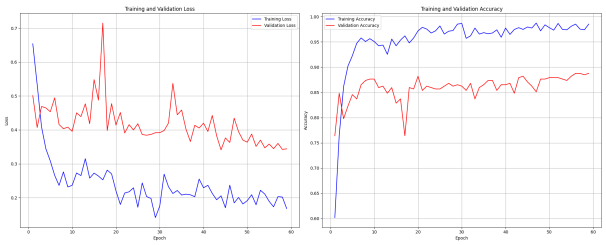


Figure 3: Training (blue) and validation (red) loss and accuracy graphs across 60 epochs for the Celeb-DF dataset. Loss quickly decreases for the training set in the first few epochs while remaining consistent for validation. Between epochs 10-30, the training loss continues to decreases while validation loss spikes high. Training and validation accuracy shows similar trends. Noting a significant gap between accuracy curves.

Center for Advanced Computing. Training runs were sent as jobs by batch to the Scholar-GPU cluster which runs with a NVIDIA Tesla V100 32GB GPU which is useful for training deep learning models on large datasets of video data. Independent models were trained on different datasets which they were then validated with.

## Experimental Results

There is a clear improvements in our model's performance when that data volume is increased. On a small dataset of 200 sample, the model achieved 85% validation accuracy by epoch 2 up from 57.5% in Epoch 1, while maintaining 89.4% training accuracy. The transition to the full dataset with 3,431 samples yielded even stronger results, with 88.8% validation accuracy in Epoch 1 - closely matching the training accuracy of 89.4%. Looking at the loss trends, the full dataset showed a smooth 76.7% reduction in loss (from 0.674 to 0.157) across 300 batches, compared to more volatile fluctuations in the small dataset. The near-identical training and validation accuracy, differing by just 0.6%, on the full dataset confirms the model's ability to generalize when provided with sufficient, diverse examples. This performance consistency, combined with the stable loss reduction, suggests the architecture is well-suited for scaling to larger datasets.

### Quantitative Evaluation

Table 3: Overall Model Evaluation

| Metric | Score |
|---|---|
| Accuracy | 0.7700 |
| Precision | 0.6875 |
| Recall | 0.9900 |
| F1 Score | 0.8115 |

Table 4: Classification Report

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Real | 0.98 | 0.55 | 0.71 | 100 |
| Fake | 0.69 | 0.99 | 0.81 | 100 |
| **Accuracy** | - | - | 0.77 | 200 |
| **Macro Avg** | 0.83 | 0.77 | 0.76 | 200 |
| **Weighted Avg** | 0.83 | 0.77 | 0.76 | 200 |

## Generalization Tests

To evaluate the generalization capability of our models, we tested each trained model on the previously unseen DeepFake Detection Challenge (DFDC) dataset. We selected 40 real videos and 40 fake videos from DFDC, preprocessed them using the same feature extraction pipeline, and evaluated all three trained models: FaceForensics++, Celeb-DF, and the combined dataset. Despite strong performance on their respective validation sets, none of the pretrained models showed generalizability on the DFDC domain. This indicates significant dataset bias and a lack of robustness to changes in deepfake content.

The FaceForensics++ model (Figure 4) shows a noticeable bias towards classifying samples as real. Out of 40 fake videos, only 3 were correctly identified. This skew suggests that the model has overfitted to FF++'s more synthetic and visually consistent manipulations. The Celeb-DF model (Figure 5) performs slightly better in terms of balance. It correctly classifies 14 fake videos and 33 real videos, showing a less pronounced, but still present biased as compared to the FF++ model. It still fails to achieve a robust separation between real and fake, suggesting that while Celeb-DF improved generalization during training, it was not sufficient for cross-dataset evaluation. Not surprisingly considering training observations, the model trained on the combined dataset (Figure 6) also performs poorly on the DFDC evaluation. It correctly identifies 19 fake videos and 24 real videos, making it better at generalizing compared to the FaceForensics++, but worse than the Celeb-DF model.

## Discussion

The experimental results demonstrate the effectiveness of the dual path feature extraction architecture. The model achieved 88.8% validation accuracy on the full FaceForensics++ dataset with a minimal performance gap (.6%) between training and validation metrics. Analysis of Table 4 reveals that the model exhibits near perfect recall (99%) but lower precision (69%) for fake videos. This shows the detector has a strong sensitivity to manipulation artifacts with occasional false positives. This bias towards false positives would indicate the model would be suitable for applications where missing a deepfake would have high consequences while incorrectly flagging would not, possibly in security.

Performance comparison across datasets shows that the model trained on FaceForensics++ shows more stable learning than on Celeb-DF. This could be due to the higher visual quality and fewer obvious artifacts in Celeb-DF videos. The temporal style patterns visualized in the experiments con-

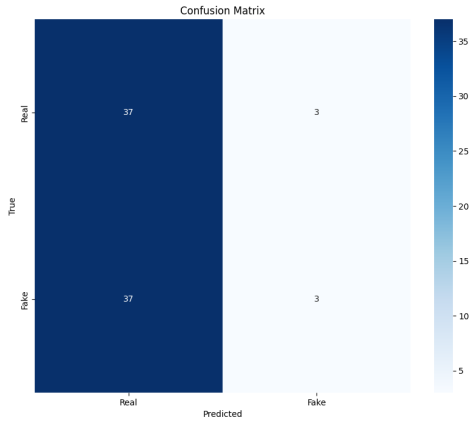


Figure 4: Confusion matrix for FF++ model evaluated on DFDC. The model fails to generalize, classifying nearly all inputs as real.

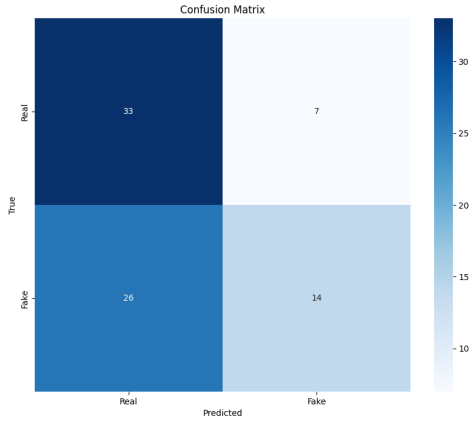


Figure 5: Confusion matrix for Celeb-DF model evaluated on DFDC. The model achieves moderate balance with 14 fake detections.

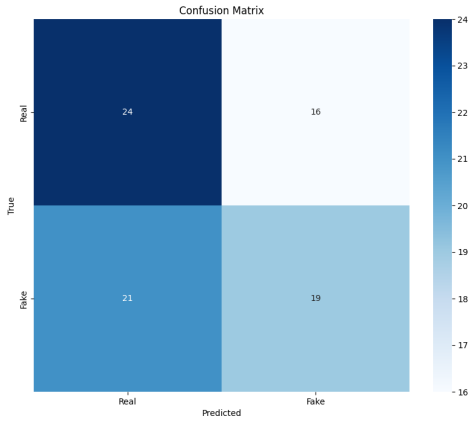


Figure 6: Confusion matrix for combined model evaluated on DFDC. The model predicts 19 out of 40 deepfake labels making it the best at identifying the positive class.

firm that style inconsistencies across frames are more noticeable in deepfake videos. The style based temporal analysis proved apt at detecting these as manipulated videos often maintain spatial coherence within individual frames but struggle with temporal flow consistency.

## Conclusion

This paper presents a novel deepfake detection system that combines spatiotemporal content analysis with style based facial attribute monitoring through a dual path feature extraction architecture. By using 3D ResNet 50 for motion analysis in conjunction with StyleGRU for temporal style analysis, the proposed architecture achieves 88% and 91% accuracy on benchmark datasets Celeb-DF and FaceForensics++ respectively. Furthermore, during implementation, the preprocessing workflow was optimized, which reduced training time by 76%.

Future work could focus on enhancing the model with more explainable AI techniques to highlight regions that trigger the detection mechanism. The model could also be ported to an online platform for ease of distribution like was during the projects inception. Further optimizations and refinements could eventually lead to real time detection.

## References

Bradski, G. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Choi, J.; Kim, T.; Jeong, Y.; Baek, S.; and Choi, J. 2024. Exploiting Style Latent Flows for Generalizing Deepfake Video Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 1133–1143.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385.

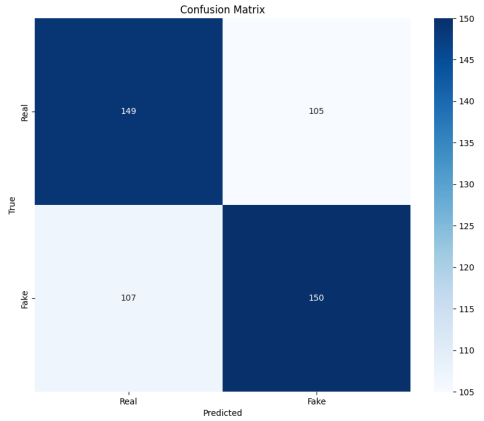


Figure 7: Confusion Matrix (combined)

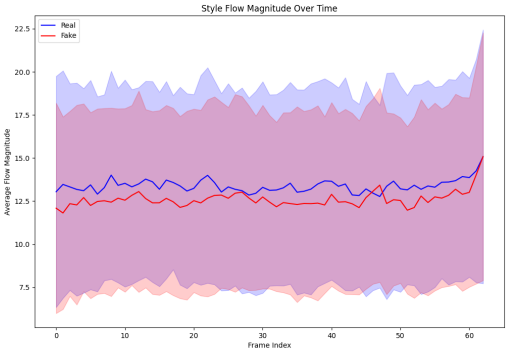


Figure 8: Temporal Style Patterns (combined)

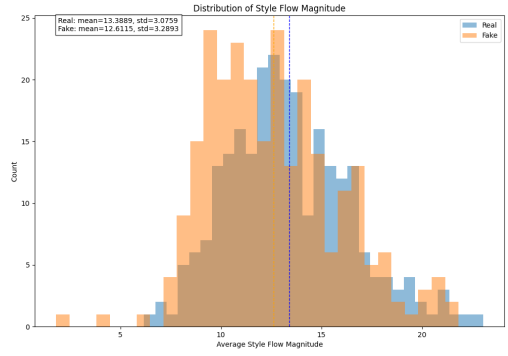


Figure 9: Style Patterns (combined)

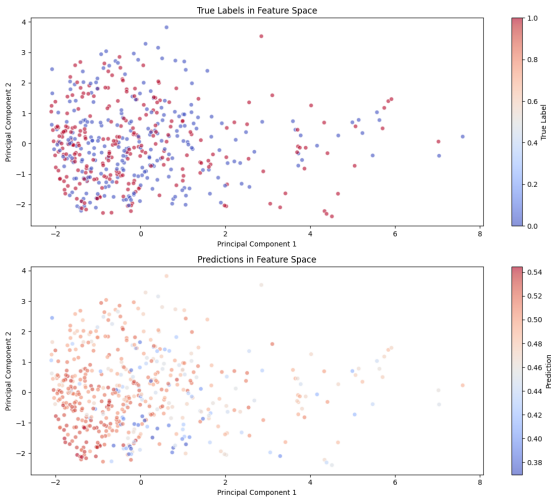


Figure 10: Decision Boundaries (combined)

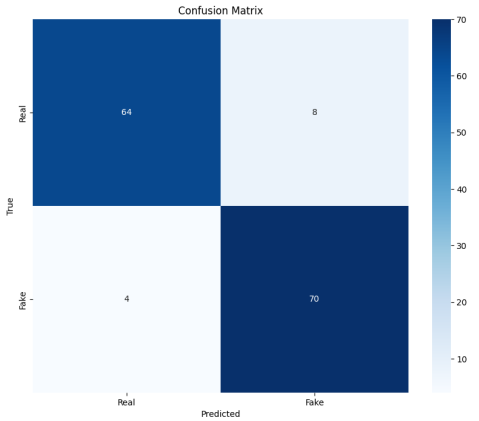


Figure 11: Confusion Matrix (FaceForensics++)

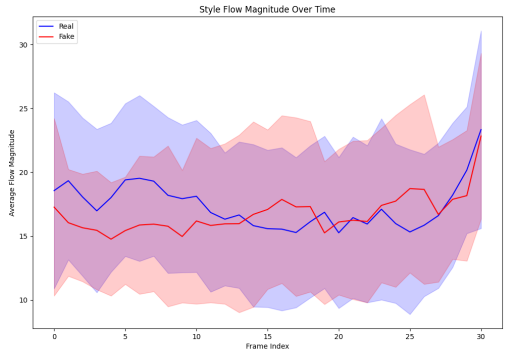


Figure 12: Temporal Style Patterns (FaceForensics++)

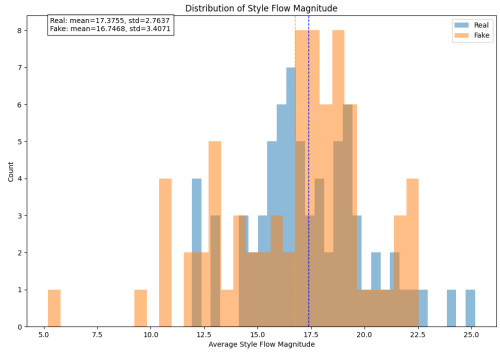


Figure 13: Style Patterns (FaceForensics++)

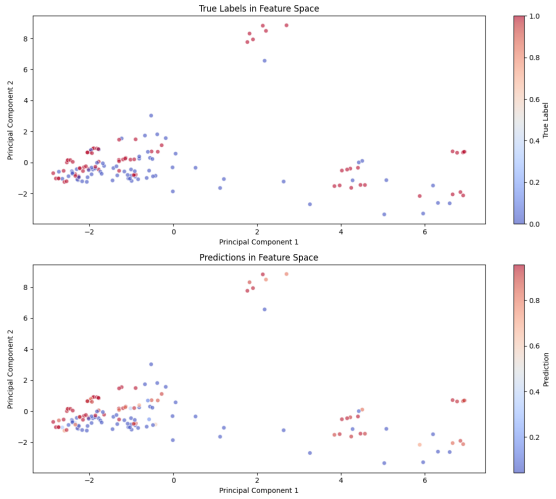


Figure 14: Decision Boundaries (FaceForensics++)

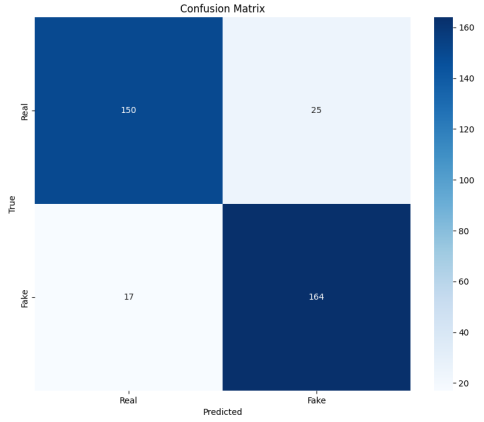


Figure 15: Confusion Matrix (Celeb-DF)

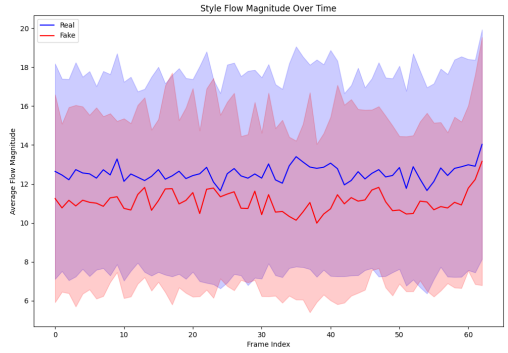


Figure 16: Temporal Style Patterns (Celeb-DF)

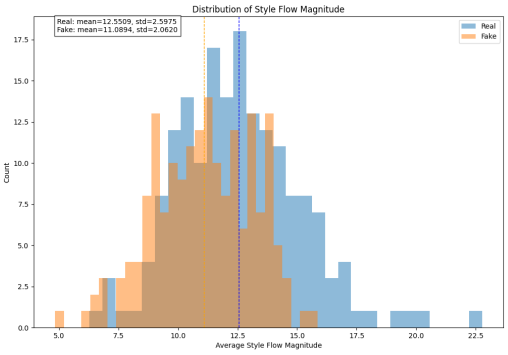


Figure 17: Style Patterns (Celeb-DF)

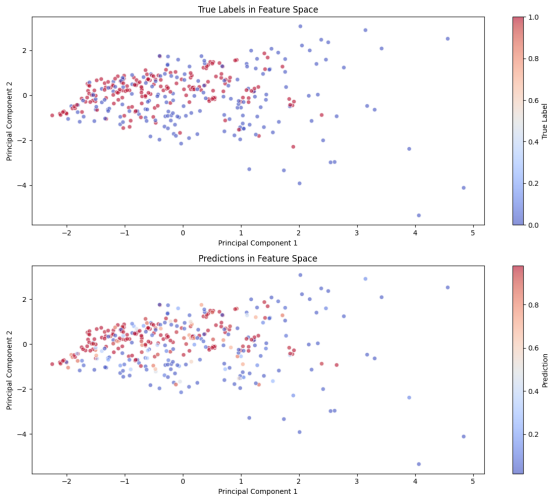


Figure 18: Decision Boundaries (Celeb-DF)

Karras, T.; Laine, S.; and Aila, T. 2019. A Style-Based Generator Architecture for Generative Adversarial Networks. arXiv:1812.04948.

Li, Y.; Yang, X.; Sun, P.; Qi, H.; and Lyu, S. 2020. Celeb-DF: A Large-scale Challenging Dataset for DeepFake Forensics. arXiv:1909.12962.

Loshchilov, I.; and Hutter, F. 2016. SGDR: Stochastic Gradient Descent with Warm Restarts. *arXiv: Learning*.

Rössler, A.; Cozzolino, D.; Verdoliva, L.; Riess, C.; Thies, J.; and Nießner, M. 2019. FaceForensics++: Learning to Detect Manipulated Facial Images. arXiv:1901.08971.

Zhang, H.; Cissé, M.; Dauphin, Y.; and Lopez-Paz, D. 2017. mixup: Beyond Empirical Risk Minimization. *ArXiv*, abs/1710.09412.

Zhang, K.; Zhang, Z.; Li, Z.; and Qiao, Y. 2016. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters*, 23: 1499–1503.