# DeepSneak: Deepfake Video Detection

Christine Tung
CS231N
Stanford University
chrtung@stanford.edu

David Tung
CS231N
Stanford University
tungd@stanford.edu

## Abstract

*Deepfake detection is critical for preserving the integrity of digital media, yet frame-based approaches often suffer from temporal inconsistency and sensitivity to irrelevant visual noise. We propose a multimodal, region-guided spatiotemporal deepfake detection framework that leverages both video content and associated textual metadata to focus on manipulation-prone segments. Our method performs semantic temporal downscaling to isolate key video moments — such as speech or expressive facial movements — where deepfake artifacts are most likely to appear. Within these moments, a region proposal network extracts high-salience facial regions and assembles them into short clips. These clips are processed using a 3D convolutional neural network (3D CNN) to capture spatiotemporal inconsistencies indicative of manipulation. The resulting features are fused with encoded action descriptions derived from video metadata to produce the final prediction. Our multimodal approach achieves superior accuracy and robustness compared to frame-based baselines with a test accuracy of 95.92%, F1-score of 0.977, and AUC of 0.904, demonstrating high reliability in distinguishing between real and deepfake instances despite class imbalance.*

## 1. Introduction

The proliferation of deepfake videos — synthetically manipulated videos generated using deep learning techniques — poses a growing threat to information authenticity, individual privacy, and public trust. Deepfakes are increasingly realistic and accessible to generate, making it difficult for both humans and traditional automated systems to detect them reliably. This has serious implications in domains such as journalism, politics, cybersecurity, and digital forensics where video authenticity is critical. Our work aims to develop a robust and effective method for deepfake video detection by leveraging both visual and textual modalities.

### 1.1. Motivation

Despite advances in deepfake detection, most existing approaches rely primarily on visual cues, such as frame-level inconsistencies or subtle artifacts. While effective in controlled environments, these methods often fail in real-world settings due to variability in lighting, pose, and editing artifacts. Our motivation stems from the need for more robust and context-aware detection systems that incorporate additional modalities beyond raw visual input.

To address this challenge, we propose leveraging textual metadata, such as video descriptions or subtitles, in conjunction with visual spatiotemporal information. The rationale is that semantic inconsistencies between a video's visual content and its associated textual metadata may serve as strong signals of manipulation. By combining these modalities, we hypothesize that we can achieve more resilient and accurate deepfake detection, particularly under challenging conditions.

### 1.2. Problem Definition

We define the problem as a binary classification task. The input is a video sample $V$, which consists of a sequence of RGB frames $\{f_1, f_2, \ldots, f_T\}$ where $T$ is the total number of frames in the video, accompanied by a textual description (e.g. "outside talking pan laughing"). The output of our algorithm is a binary label $\hat{y} \in \{0, 1\}$, where $\hat{y} = 0$ indicates a real video and $\hat{y} = 1$ indicates a manipulated (deepfake) video.

Our baseline algorithm takes the RGB frames and uses a 2D CNN to process each frame independently, producing a frame-level binary prediction. A majority voting scheme is then applied over all frame-level predictions to output the final video-level binary label.

Our proposed algorithm takes the RGB frames and the associated textual description. For visual feature extraction, we explore two distinct pre-processing methods:

1. Interval Downscaling: We sample frames at a regular temporal interval throughout the video and compress each frame into a low-resolution square image.

2. Region-Guided Cropping: We employ a region proposal network (RPN), specifically leveraging MTCNN's facial confidence scores and a custom window scoring mechanism, to automatically identify and select high-salience facial regions within key frames. This produces high-resolution crops of consistent spatial and temporal dimensions.

The selected frames are then passed through a 3D CNN to extract rich spatiotemporal visual features that capture both motion and localized artifacts. In parallel, the associated textual description is passed through a Transformer-based text encoder to extract semantic textual features. Finally, the extracted visual and textual feature representations are fused and passed through a sequential linear classification head to produce the predicted binary label.

## 2. Related Work

Deepfake detection has rapidly evolved along with deepfake generation and manual human detection is ineffective, shifting the task towards automated detection methods. Existing research falls into several key categories, each contributing to the field's deeper understanding of detection methods. Here, we provide an overview of prominent approaches, their strengths and weaknesses, and how our proposed multimodal, region-guided spatiotemporal framework builds upon and differentiates itself from prior work.

### 2.1. Spatiotemporal Feature Extraction with 3D CNNs

Early efforts in deepfake detection leveraged 3D CNNs to capture both spatial and temporal inconsistencies inherent to manipulated videos. These models are designed to analyze sequences of frames, modeling motion patterns and localized artifacts simultaneously.

1. Ganiyusufoglu *et al*. [5] proposed using 3D CNNs to model spatiotemporal features, showing improved generalization across various manipulation techniques compared to traditional image-based classifiers.

2. De Lima *et al*. [3] introduced spatiotemporal convolutional methods, outperforming frame-based detection methods on the Celeb-DF dataset.

3. Tariq *et al*. [21] developed CLRNet, a Convolutional LSTM-based residual network that learns temporal information from consecutive frames to detect unnatural artifacts present between frames.

4. Guo *et al*. [7] proposed a guided residuals network (GRNet) that fuses spatial-domain and residual-domain features to expose generated face images.

While these methods effectively capture motion and temporal inconsistencies, their primary weakness lies in their susceptibility to very subtle artifacts, particularly those introduced by sophisticated deepfake techniques. Furthermore, they typically require large datasets for training and can be computationally intensive. Our approach similarly utilizes 3D CNNs for spatiotemporal feature extraction, aligning with these foundational methods' strengths. However, we augment this by introducing semantic temporal downscaling and region-guided cropping, enabling our model to focus on more manipulation-prone segments.

### 2.2. Attention Mechanisms and Temporal Modeling

To address the limitations of earlier models, subsequent research has incorporated attention mechanisms and advanced temporal modeling techniques.

1. Gu *et al*. [6] proposed the Spatial-Temporal Inconsistency Learning (STIL) framework, introducing modules to capture spatial and temporal inconsistencies, enhancing detection performance.

2. Chen *et al*. [1] developed a model focusing on localized manipulative signatures using spatial and temporal attention mechanisms, achieving significant performance improvements.

3. Lu *et al*. [14] introduced a spatial-temporal model with a long-distance attention mechanism to capture artifacts in both spatial and temporal domains.

4. Li *et al*. [13] introduced a novel transformer architecture with a texture-aware branch, a bidirectional interaction cross-attention module, and a shape-guided Gaussian mapping strategy.

These attention-driven approaches significantly improve the model's ability to focus on subtle and localized artifacts, making them highly effective. However, their increased complexity can lead to higher computational costs and a greater risk of overfitting without sufficient data. Our framework directly benefits from insights gained from these methods, specifically in our strategy to extract high-salience temporal and region frames. By selecting key moments and regions, our system implicitly employs a form of attention, guiding the 3D CNN to where inconsistencies are most likely to occur, without necessarily adding the overhead of explicit attention layers across the entire video.

### 2.3. Multimodal and Hybrid Learning

Recognizing the importance of integrating multiple modalities and scales, several recent studies have explored multimodal learning frameworks.

1. Wu *et al*. [8] introduced the Spatial-Temporal Deepfake Detection and Localization (ST-DDL) network, combining spatial and temporal features, utilizing the Anchor-Mesh Motion (AMM) algorithm for precise facial micro-expression modeling.

2. Chen *et al*. [2] focused on compressed deepfake videos, using 3D spatiotemporal trajectories to detect manipulations in compressed formats.

3. Saikia *et al*. [20] proposed a hybrid CNN-LSTM model leveraging optical flow features, achieving competitive performance with reduced sample sizes.

4. Mallet *et al*. [15] developed a hybrid model utilizing multilayer perceptron and long short-term memory networks, achieving accuracies up to 74.7%.

The primary strength of these methods is their enhanced robustness across various video qualities, manipulation techniques, and diverse artifact types. However, a common weakness is their requirement for extensive computational resources, complex data synchronization, and extensive fine-tuning across different datasets due to the integration of multiple data streams or models.

Our approach falls directly into this category of multimodal learning, but with a different emphasis. While previous multimodal works often focus on integrating various visual cues, our approach directly leverages semantic inconsistencies between video content and external textual descriptions. We also prioritize identifying and preserving high-resolution features within specific facial regions where deepfake artifacts are most prevalent. This combined strategy of targeted visual analysis and multimodal fusion provides a more comprehensive and robust detection capability than pure visual or hybrid visual approaches.

# 3. Methods

## 3.1. Baseline Method

As a baseline, we implement a frame-level classification strategy leveraging a pre-trained 2D CNN. Each input video is decomposed into individual RGB frames, which are processed independently by a ResNet-18 model [10] $f_\varphi$ and fine-tuned to classify each frame $f_t$. The output $f_\varphi(f_t)$ is a probability score $p_t \in [0, 1]$, which is thresholded at 0.5 into to a binary prediction $\hat{y}_t \in \{0, 1\}$.

$$\hat{y}_t = \begin{cases} 1 & \text{if } f\varphi(f_t) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

To obtain a video-level prediction from frame-level scores, we apply majority voting across all $T$ frames:

$$\hat{y} = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \hat{y}_t > \frac{T}{2} \\ 0 & \text{otherwise} \end{cases}$$

This approach benefits from being computationally straightforward and compatible with existing large-scale image models. However, it presents several key drawbacks:

1. No Temporal Modeling: Each frame is treated independently, preventing the model from leveraging motion cues or detecting inconsistencies over time.

2. Uniform Importance of Frames: The model gives equal weight to all frames, including potentially uninformative frames.

3. Sensitivity to Noise: Frame-level classifiers are vulnerable to compression artifacts and occlusions, which can obscure deepfake traces.

Despite its limitations, this serves as a strong and commonly used baseline in deepfake detection literature, establishing a basis for the improvements we introduce.

## 3.2. Proposed Method

Our proposed system is a multimodal, region-guided, spatiotemporal deepfake detection framework. It consists of three main components: visual feature extraction (with two distinct pipelines), visual and textual feature encoding, and multimodal fusion for the final classification.

### 3.2.1 Visual Feature Extraction

**1. Interval Downscaling (Evenly Spaced Frames)** Given an input video represented as a sequence of RGB frames $\{f_1, \ldots, f_T\}$, we select $k = 16$ evenly spaced frames across the video. The indices $f_i$ are given by $f_i = \text{round}\left((i + 0.5) \times \frac{T}{k}\right)$ for $i = 0, \ldots, k - 1$.

**2. Region-Guided Cropping (Optimal Consecutive Clip)** This pipeline focuses on extracting high-resolution visual features from manipulation-prone facial regions within an optimal consecutive video clip. For each original video, we first apply Multi-task Cascaded Convolutional Networks (MTCNN) [22] at a configurable interval $M$. MTCNN employs a proposal, refine, and output network for efficient face detection. Each detected bounding box is assigned a score based on its MTCNN confidence and we propose to scale this with a spatial integrity component (defined as the fraction of its pixels residing inside the frame). This incentivizes larger and more complete detections. These scores are then used to identify the single optimal 16-frame clip within the video that exhibits the highest overall facial activity and confidence. After, we apply MTCNN densely

across each frame in this clip to extract high-resolution facial crops. This strategy reduces overall computational cost by a factor of $M$ to identify a coarse segment, then focuses dense processing only where most impactful. The output is a single spatiotemporal patch, $V_{\text{facial}} \in \mathbb{R}^{16\times224\times224\times3}$, representing the high-resolution facial crop across the chosen consecutive frames.

#### 3.2.2 Visual and Textual Feature Encoding

We applied adaptive average pooling to each selected frame to resize into $112 \times 112$. This approach preserves global structural information by aggregating spatial features, rather than discarding them. Compared to center cropping, which may remove salient peripheral information, or linear interpolation, which may introduce artifacts or blur fine-grained details, adaptive average pooling offers a content-aware resizing mechanism. It ensures all regions of the frame contribute proportionally to the final representation, thereby retaining more semantically meaningful context. These $k$ frames are prepared as a single input sequence ($V_{\text{linspace}}$ from Interval Downscaling, or $V_{\text{facial}}$ from Region-Guided Cropping) and passed through a 3D CNN, $g_\theta$. Unlike the 2D CNNs used in the baseline, 3D CNNs apply convolutional filters in both spatial and temporal dimensions, allowing the model to capture patterns such as unnatural motion, flickering, or asynchronous lip movements. The 3D CNN processes the 16-frame sequence and outputs a compact visual feature representation, $F_{\text{visual}} \in \mathbb{R}^{d_{\text{visual}}}$.

To further enhance detection, we incorporate semantic features derived from associated video metadata. The video title text is encoded using all-MiniLM-L6-v2 [18], a pre-trained Transformer-based text encoder. This produces a textual feature vector $F_{\text{text}} \in \mathbb{R}^{d_{\text{text}}}$.

#### 3.2.3 Multimodal Classification

The final prediction $\hat{y}$ is computed by fusing the visual and text features through a multi-layer perceptron (MLP):

$$\hat{y} = W_3 \cdot \phi\left(W_2 \cdot \phi\left(W_1 \cdot [F_{\text{visual}}; F_{\text{text}}] + b_1\right) + b_2\right) + b_3$$

- $F_{\text{visual}} \in \mathbb{R}^{512}$ is the visual feature vector extracted from the pre-trained 3D ResNet-18 model, after removing the final classification head.
- $F_{\text{text}} \in \mathbb{R}^{512}$ is the projected textual feature vector of the 384-dimensional output from the all-MiniLM-L6-v2 encoder.
- $[F_{\text{visual}}; F_{\text{text}}] \in \mathbb{R}^{1024}$ denotes concatenation.
- $W_1 \in \mathbb{R}^{1024\times1024}$, $W_2 \in \mathbb{R}^{1024\times512}$, $W_3 \in \mathbb{R}^{512\times C}$ are the weight matrices of the fusion MLP layers and
- $b_1, b_2, b_3$ are the corresponding bias vectors.
- $\phi(\cdot)$ represents the ReLU activation function.

### 3.3. Criterion and Optimizer

#### 3.3.1 Optimization and Loss Function

We use the Adam optimizer [12] to train our models. Adam combines the benefits of RMSProp and momentum by adaptively adjusting learning rates for each parameter based on estimates of first and second moments of the gradients. The parameter update rule at iteration $t$ is:

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

- $m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta L_t$ is the exponentially weighted average of the gradients,
- $v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla_\theta L_t)^2$ is the exponentially weighted average of the squared gradients,
- $\hat{m}_t$ and $\hat{v}_t$ are bias-corrected estimates,
- $\alpha$ is the learning rate and $\epsilon$ is a small constant to prevent division by zero.

We use weighted binary cross-entropy loss to handle class imbalance. Deepfake datasets often contain a skewed distribution of real and fake examples, which can bias the model toward predicting the majority class. To correct this, we apply weights inversely proportional to class frequencies. The weighted cross-entropy loss for a single prediction $\hat{y} \in (0,1)$ and ground truth label $y \in \{0,1\}$ is:

$$\mathcal{L}_{\text{WCE}}(y,\hat{y}) = -w_1 y \log(\hat{y}) - w_0(1-y)\log(1-\hat{y})$$

where $w_1$ and $w_0$ are the weights of the positive and negative classes, respectively. These are computed as:

$$w_c = \frac{1}{\text{frequency of class } c}, \quad c \in \{0,1\}$$

This approach is especially important given the significant class imbalance in the DFDC dataset (363 original videos compared to 3,068 manipulated videos). Without correction, this imbalance would cause the model to disproportionately favor the majority class, leading to high overall accuracy but poor recall on the minority class. By incorporating weighting into the loss function, we guide the model to treat both classes with appropriate significance, improving performance on real video detection and ensuring a robust classifier that can be applied to other deepfake datasets.

### 3.4. Implementation Notes

Our codebase is based on PyTorch and is built upon publicly available MTCNN, ResNet [10], 3D ResNet [9], and Sentence Transformer [18] implementations. We implemented the preprocessing RPN, metadata parser, loss weighting, and multimodal fusion from scratch. The temporal cropping and patch selection pipeline was also customized to interface with pre-trained detection and recognition models.

The raw input consists of MP4 video files uploaded and stored in an Amazon S3 bucket. We set up two pipelines, with variations that work both locally and via cloud. A distributed processing script handled custom preprocessing:

1. Frame extraction: select a specified number of frames using either (1) linearly spacing condensed frames across the duration of the video or (2) high-resolution, continuous facial crops.

2. Description: parse the MP4 title for a text description of the video content. In our dataloader, a pre-trained transformer maps this into an embedding space.

3. Label: parses the directory name to identify the label.

These were then zipped into a tar.gz file for efficient storage and retrieval, and saved into a separate sharded S3 directory. The training infrastructure contains logic to handle:

1. WebDataset loaders to handle data extraction, train-validation-test splits, and data normalization. To address the significant class imbalance, we implemented stratified sampling during the train-validation-test split to ensure each subset maintained approximately the same label distribution as the overall dataset. This mitigates bias in model evaluation and prevents the training process from overfitting to overrepresented classes. We enumerated all available S3 shards and extracted their corresponding labels. Then, we performed splits preserving the original class proportions, with a fixed random seed for reproducibility.

2. Model training that loads a pre-trained 3D ResNet-18 model and all-MiniLM-L6-v2 encoder, runs the data through these models, and fuses the results together to generate the final prediction. We implemented a dual-stream architecture combining visual and textual input information from both modalities. Then, we concatenate the projected text and video embeddings, forming a fused 1024-dimensional feature vector. This vector is passed through a feedforward neural network with two hidden layers (1024 and 512 units respectively), each followed by ReLU activation and dropout (p = 0.2) to prevent overfitting. The final layer maps the fused representation to the number of target classes (binary classification in our case). The model is trained using an Adam optimizer with differential learning rates: a lower learning rate (1e-5) for fine-tuning the pre-trained video encoder and a higher learning rate (1e-3) for the fusion layers and text projection head. This helps retain generalizable features learned from large-scale pretraining while adapting the model to the downstream task.

3. Functionality to save the model state with the best performance on the validation set, per-epoch checkpointing, and the ability to resume training from the last saved epoch checkpoint. This allowed us to perform longer training runs, while protecting against interruptions and avoid starting training from scratch.

## 4. Dataset and Features

We leverage the DeepFake Detection Challenge (DFDC) dataset, a large-scale benchmark designed to support building deepfake detection systems introduced by Dolhansky *et al*. [4]. The dataset exhibits a significant class imbalance, posing a challenge for model training as discussed in Section 3.3.1. Each video is annotated with a short description in the file title. Manipulated content was generated using a variety of facial synthesis and replacement techniques. DFDC's diversity in manipulation methods, actors, scenes, and lighting conditions make it a comprehensive and challenging benchmark to evaluate model robustness.

We implement the following preprocessing pipeline:

1. Frame Sampling: As described in Section 3.2.1, we uniformly sample 16 non-consecutive frames for Interval Downscaling or identify and extract an optimal 16-frame contiguous clip containing high-salience facial activity for Region-Guided Cropping. The choice of 16 frames strikes a balance between computational efficiency and temporal coverage. This number is commonly used in prior video analysis works, including C3D and I3D, and has been shown to provide sufficient temporal context for capturing motion cues and facial artifacts indicative of deepfake manipulations.

2. Frame Resizing: Each sampled frame is resized from the original dimensions of 1920x1080 down to a square spatial resolution of 112×112 [16]. This size is a widely adopted standard in image and video recognition models, such as ResNet and I3D. Working within this resolution enables compatibility with transfer learning from ImageNet or Kinetics-400 pretrained models. Additionally, 112×112 provides a balance between computational efficiency and sufficient spatial resolution to preserve facial features and subtle manipulation artifacts crucial for deepfake detection.

3. Normalization: The resized frames are normalized using the mean and standard deviation corresponding to the pre-trained model used for transfer learning (e.g., ImageNet: mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225] [19]; or Kinetics-400: mean = [0.43216, 0.394666, 0.37645], std = [0.22803, 0.22145, 0.216989] [11]). This step ensures that pixel intensities fall within a consistent range, which is important to ensure compatibility with transfer learning from ImageNet-pretrained models, as these networks expect inputs to follow the same distribution. Normalization also contributes to training stability, faster convergence, and helps prevent issues such as vanishing or exploding gradients during backpropagation.

4. Augmentation: No augmentation was done here. This allows us to preserve the original artifact, as well as ensure cohesiveness with the textual description.

5. Splitting the Dataset: To ensure fair and reproducible evaluation, we perform an 80-10-10 split of the dataset into mutually exclusive train, validation, and test sets of respective sizes 2738, 342, and 343, using a consistent random seed [17]. We stratified by class label to ensure that each fold had representative proportions of real and fake videos. We follow the standard practice of training purely on the training set, using the validation set to identify the best-performing model variant, and running the model with the best performance on the validation set on the unseen test set at the end of training. This provides a solid method of evaluating the effect of hyperparameter tuning.

6. Features: Our model leverages two primary types of features: visual features derived from the preprocessed video frames (either $V\_textID$ or $V$_RGC) and textual features extracted from the associated video descriptions. The visual features are learned implicitly by the 3D CNN from the spatiotemporal sequences, while the textual features are obtained through a pretrained Transformer-based encoder.

7. Dataloader Construction: The preprocessed sequences of frames, along with their corresponding labels, are encapsulated into a PyTorch dataloader to facilitate efficient training and validation operations.

# 5. Results

## 5.1. Experiments and Hyperparameters

Preliminary experiments were conducted on a subset (30%) of the DFDC dataset using Colab. This process was critical in designing the model architecture and training protocol, particularly given the dataset's significant class imbalance.


Figure 1: Example frames from an original (real) video in the DFDC dataset. The associated textual description is "outside, talking, pan, laughing".

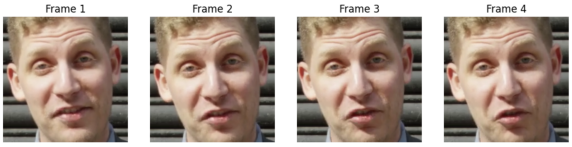



Figure 2: Consecutive video frames preserving resolution determined via MTCNN facial recognition.

We trained our models using the Adam optimizer with a learning rate of $1 \times 10^{-3}$, chosen after preliminary tuning using a held-out validation set and comparison to the RMSProp optimizer. We explored a variety of batch sizes (4, 8, 16, 32, 64, 128, 256) to balance convergence speed with memory constraints on different hardware (NVIDIA T4 and NVIDIA A10G) and mixed precision. All models were trained for 10 epochs, with model checkpoints saved based on the best validation performance.

From these preliminary experiments, our final model was trained using the Adam optimizer with a learning rate of $1 \times 10^{-3}$, a batch size of 64, and non-mixed-precision training on an NVIDIA A10G GPU, balancing convergence speed, stability, and efficient resource usage.

## 5.2. Evaluation Metrics

To rigorously assess performance, we report standard classification metrics commonly used in related literature: Accuracy, Precision, Recall, F1-Score, and AUC.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Accuracy measures the overall correctness by computing the ratio of correctly predicted instances to the total number of instances. $TP$ (True Positives) and $TN$ (True Negatives) are correctly classified videos and $FP$ (False Positives) and $FN$ (False Negatives) are misclassified videos.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

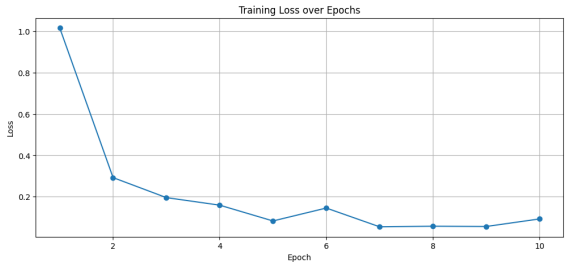



Figure 3: Proposed Method: training loss over epochs.

Precision quantifies the number of correctly predicted positive instances among all instances predicted as positive. High precision indicates a low false positive rate, which is particularly important when minimizing false alarms.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

Recall measures the model's ability to correctly identify all actual positive instances. A high recall value ensures the model captures most of the deepfake content without missing significant instances.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

F1-score is the harmonic mean of precision and recall, providing a single metric that balances both concerns, especially useful in imbalanced datasets.

$$\text{TPR} = \frac{TP}{TP + FN}, \quad \text{FPR} = \frac{FP}{FP + TN} \tag{5}$$

AUC is the area under the Receiver Operating Characteristic (ROC) curve and ranges from 0 to 1, where a value closer to 1 indicates better discriminative capability.

### 5.3. Results

Table 1: Proposed Method: validation classification report.

| Class | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| Real (0) | 0.93 | 0.83 | 0.88 | 35 |
| Fake (1) | 0.98 | 0.99 | 0.98 | 307 |
| **Accuracy** | | 96.20% | | |
| **Macro avg** | 0.96 | 0.91 | 0.93 | 342 |
| **Weighted avg** | 0.97 | 0.96 | 0.96 | 342 |

Table 1 reports the classification performance of the best model, selected based on its highest validation accuracy, which occurred at Epoch 7. The model demonstrates strong performance across both classes, achieving a weighted average F1-score of 0.96 and a macro average of 0.93. Specifically, the classifier achieves 98% precision and 99% recall

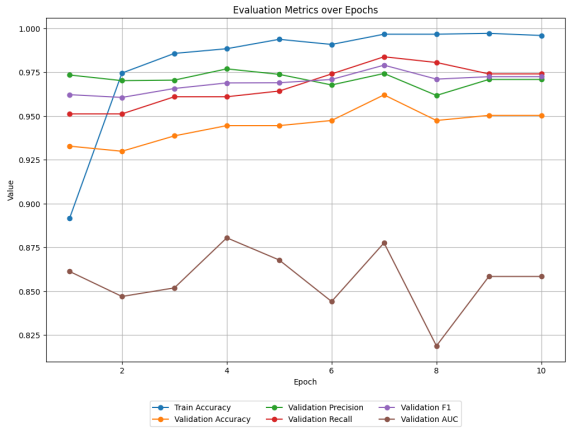



Figure 4: Proposed Method: evaluation metrics over epochs.

on the majority class, and 93% precision and 83% recall on the minority class. Despite the class imbalance, the model maintains relatively strong recall and precision for the underrepresented class.

Table 2: Proposed Method: validation confusion matrix.

| | **Predicted: Real (0)** | **Predicted: Fake (1)** |
|---|---|---|
| **Actual: Real (0)** | 29 | 6 |
| **Actual: Fake (1)** | 3 | 304 |

This performance is further clarified by the confusion matrix shown in Table 2. Out of 35 real instances, 29 were correctly identified, while 6 were misclassified as deepfake. The model made only 3 errors on the deepfake class, showcasing excellent sensitivity.

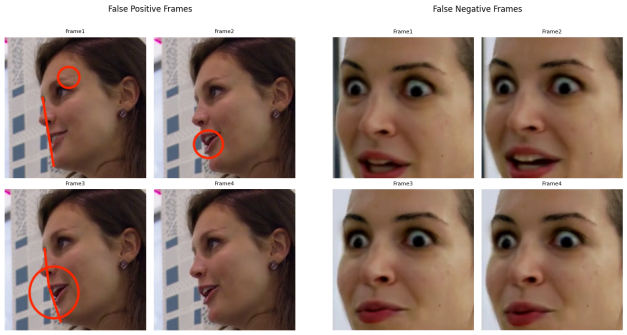
Table 3: Proposed Method: performance of the best model on the test set.

| Metric | Accuracy | Precision | Recall | F1-score | AUC |
|---|---|---|---|---|---|
| **Score** | 95.92% | 0.980 | 0.974 | 0.977 | 0.904 |

To evaluate generalization, we measured model accuracy on a held-out test set. From Table 3, the model achieves 95.92% accuracy, 0.977 F1-score, 0.980 precision, and 0.974 recall. The AUC score of 0.904 indicates the model effective distinguishes the two classes across a range of thresholds, reinforcing its robustness. The close alignment between the validation and test metrics suggests that the model generalizes well and has not overfit. Our results achieve superior accuracy and robustness compared to the baseline model, which achieved only 94.5% accuracy and poor AUC score of 0.542 (shown in appendix 7.1).

Figure 3 and Figure 4 visualize the training loss and key performance metrics over 10 epochs. The training loss decreases sharply from Epoch 1 to Epoch 5, and then continues to decline more gradually, stabilizing around Epoch 7. Similarly, the training accuracy steadily increases and plateaus around 96–97%, with the validation accuracy showing a peak at Epoch 7. Despite the training accuracy reaching nearly 99.7%, the validation accuracy remains stable after Epoch 7. This gap, while non-trivial, is not substantial enough to indicate significant overfitting. The consistency between validation and test performance further supports this conclusion.

To mitigate overfitting, we employed several strategies to ensure the model maintained generalization capability without memorizing the training data. Early stopping was applied based on the validation accuracy, with the best model saved at Epoch 7. Explicit class imbalance handling, implemented via weighted loss function, was crucial in guiding the model to learn representative features for both classes, thereby improving generalization beyond just the majority class. We applied regularization via dropout and ensured that the model architecture was not overly deep.




(a) False positive (b) False negative

Figure 5: Examples of false detections.

Figure 5a illustrates a challenging case from the test set where our model incorrectly classified a real video as fake (false positive). While pinpointing the exact cause of misclassification can be difficult, several factors may contribute here. The subject's side-profile facial orientation might present a more complex learning challenge compared to frontal views, as key facial features are partially obscured. Furthermore, closer inspection reveals subtle visual inconsistencies in her facial features across frames that could mislead the model. For instance, the ends of her eyebrows appear to curve downwards then abruptly upwards, the tilt of her head as she talks varies significantly between frames, and the curvature of her mouth changes as she speaks.

Figure 5b illustrates a case from the test set where our model incorrectly classified a fake video as real. The most prominent characteristic is the extreme zoom level of the subject's face. This presents a direct challenge to our cropping methodology. A common indicator of deepfake manipulation lies in the detection of inconsistencies around edges. However, in this highly zoomed-in scenario, many of these are simply not visible within our constraimed crop. This highlights a limitation of our current approach: while region-guided cropping can focus on high-salience areas, it may exclude important information if the spatial scale differs significantly from the training distribution.

## 6. Conclusion and Future Work

In this work, we presented a multimodal, region-guided spatiotemporal framework for deepfake detection, directly addressing key limitations in traditional frame-based detection approaches. By incorporating both video content and associated textual metadata, our system focuses on manipulation-prone segments, significantly improving detection accuracy and robustness. Our proposed pipeline leverages semantic temporal downscaling to isolate expressive video moments, employs a region proposal network to extract salient facial regions. These are processed by a 3D CNN to capture subtle spatiotemporal inconsistencies. Fusion with metadata-derived action descriptions results in more informed predictions.

Among the components of our framework, the region-guided 3D CNN consistently demonstrated the highest performance. This stems from its ability to preserve temporal continuity and focus on high-salience regions, rather than processing entire, potentially irrelevant frames uniformly. Moreover, the integration of textual metadata provided complementary context, improving accuracy in ambiguous scenarios and under challenging conditions like compression artifacts or occlusions. The failure of frame-only models under these settings highlights the importance of jointly modeling temporal and semantic information.

Looking ahead, several promising avenues could extend this work. Expanding multimodal inputs to include audio cues or speaker identity verification could offer additional, powerful signals for manipulation detection. Based on our observed failure cases, future efforts should also focus on robustly handling diverse video characteristics, such as processing multiple clips within the same video (potentially using the text metadata to guide the region proposal network) and adapting to varying zoom levels. With access to larger-scale annotated datasets and more computational resources, we would also aim to train end-to-end architectures with improved generalization to unseen manipulation techniques. Furthermore, integrating cross-attention mechanisms or transformer-based modules could enhance the framework's ability to model long-range temporal dependencies and complex multimodal interactions.


Figure 6: Baseline Method: training loss over epochs.

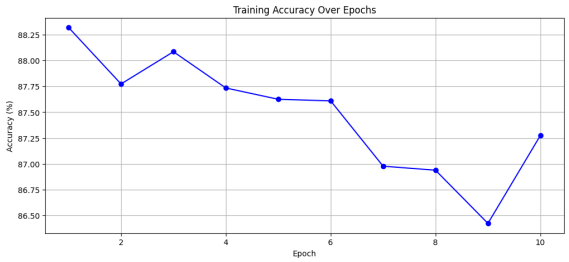



Figure 7: Baseline Method: training accuracy over epochs.

# 7. Appendices

## 7.1. Baseline Results

Figure 6 shows the training loss over 10 epochs. The model's training loss decreased from an initial value of 0.632 to 0.534 by the final epoch, demonstrating moderate learning progress. The use of a weighted loss function — with higher weight assigned to real samples due to their lower representation in the dataset — contributed to the slower convergence, as the model was penalized more heavily for misclassifying real samples.

Figure 7 shows the training accuracy over the same period. The model's training accuracy started at 88.32% and decreased to 87.27% at the final epoch. This fluctuation may indicate the model adjusted to the class imbalance, prioritizing recall for the minority class at the expense of overall accuracy. The weighted loss objective may have led the model to be too conservative in classifying samples as fake, which could benefit real sample recall but reduce short-term accuracy. These training metrics suggest the model is learning under the influence of the weighted loss and may benefit from further fine-tuning or additional epochs to stabilize its performance.

Table 4 reports class-wise evaluation metrics on accuracy, precision, recall, F1-score, and support. The model achieved an overall accuracy of 94.5%, largely attributed to strong performance on the majority class. For the majority class, precision and recall were 94.47% and 100.00%, respectively, resulting in a high F1-score of 97.16%, which indicates the model's consistent success in detecting fake content.

Table 4: Baseline Method: validation classification report.

| Class | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| Real (0) | 1.00 | 0.08 | 0.15 | 12 |
| Fake (1) | 0.94 | 1.00 | 0.97 | 188 |
| **Accuracy** | | 94.50% | | |
| **Macro avg** | 0.97 | 0.54 | 0.56 | 200 |
| **Weighted avg** | 0.95 | 0.94 | 0.92 | 200 |

Table 5: Baseline Method: validation confusion matrix.

| | Predicted: Real (0) | Predicted: Fake (1) |
|---|---|---|
| **Actual: Real (0)** | 1 | 11 |
| **Actual: Fake (1)** | 0 | 188 |

Table 5 shows the model correctly identified all fake videos but misclassified 11 out of 12 real videos. This led to a low recall of only 8.3% for real samples, despite perfect precision for that class due to the absence of false positives. The macro-averaged recall was 54.0%, highlighting the imbalance. This discrepancy is also reflected in the AUC-ROC score of 0.542, suggesting that the model's discriminative capability across decision thresholds is marginally better than random. These results underscore the primary limitation of the baseline approach - its inability to generalize effectively to the minority class. Although a weighted loss function was employed to mitigate class imbalance, it was insufficient to achieve balanced performance.

# 8. Contributions and Acknowledgements

Christine Tung: data preprocessing, Colab framework, models, experiments, paper

David Tung: data preprocessing, AWS framework, models, experiments, paper

We thank the CS231N course staff for providing compute credits on GCP and AWS for final project training; their support enabled extensive experimentation and accelerated model development.

# References

[1] Y. Chen, N. Akhtar, N. A. H. Haldar, and A. Mian. Deepfake detection with spatio-temporal consistency and attention, 2025.

[2] Z. Chen, X. Liao, X. Wu, and Y. Chen. Compressed deepfake video detection based on 3d spatiotemporal trajectories, 2024.

[3] O. de Lima, S. Franklin, S. Basu, B. Karwoski, and A. George. Deepfake detection using spatiotemporal convolutional networks, 2020.

[4] B. Dolhansky, J. Bitton, B. Pflaum, J. Lu, R. Howes, M. Wang, and C. C. Ferrer. The deepfake detection challenge (dfdc) dataset, 2020.

[5] I. Ganiyusufoglu, L. M. Ngô, N. Savov, S. Karaoglu, and T. Gevers. Spatio-temporal features for generalized detection of deepfake videos, 2020.

[6] Z. Gu, Y. Chen, T. Yao, S. Ding, J. Li, F. Huang, and L. Ma. Spatiotemporal inconsistency learning for deepfake video detection, 2021.

[7] Z. Guo, G. Yang, J. Chen, and X. Sun. Exposing deepfake face forgeries with guided residuals, 2022.

[8] W. Haiwei, Z. Jiantao, Z. Shile, and T. Jinyu. Exploring spatial-temporal features for deepfake detection and localization, 2022.

[9] K. Hara, H. Kataoka, and Y. Satoh. Learning spatio-temporal features with 3d residual networks for action recognition, 2017.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[11] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman. The kinetics human action video dataset, 2017.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Y. Li, Y. Li, X. Wang, B. Wu, J. Zhou, and J. Dong. Texture, shape and order matter: A new transformer design for sequential deepfake detection, 2024.

[14] W. Lu, L. Liu, J. Luo, X. Zhao, Y. Zhou, and J. Huang. Detection of deepfake videos using long distance attention, 2021.

[15] J. Mallet, N. Krueger, M. Vanamala, and R. Dave. Hybrid deepfake detection utilizing mlp and lstm, 2023.

[16] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[18] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[19] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge, 2015.

[20] P. Saikia, D. Dholaria, P. Yadav, V. Patel, and M. Roy. A hybrid cnn-lstm model for video deepfake detection by leveraging optical flow features, 2022.

[21] S. Tariq, S. Lee, and S. S. Woo. A convolutional lstm based residual network for deepfake video detection, 2020.

[22] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, Oct. 2016.