Deepfake Video Detection: A Comparative Study

Roberto Giordano
DISI, University of Bologna
roberto.giordano2@studio.unibo.it

Alessio Pittiglio
DISI, University of Bologna
alessio.pittiglio@studio.unibo.it

Abstract

Deepfake detection methods typically rely on CNNs such as Xception, which learn local features but struggle to capture long-range spatial dependencies. More recent transformer-based architectures address this issue through self-attention mechanisms, but they are computationally expensive, and this limits their applicability to high-resolution inputs. In this project, we present a comparative study of deepfake detection approaches. All models are trained and tested on four benchmark datasets: Google DFD, DFDC, FaceForensics++, and Celeb-DF. Our experiments show that our fine-tuned version of MambaVision outperformed Xception and Video Transformer, achieving an AUC of 0.797 on the DFDC test set and obtaining an inference speed-up of 2.5 and a reduction in model size of 2.8 over Video Transformer. We then integrated this model into a prototype detection system and deployed it as a Hugging Face Space. By providing this comprehensive analysis and a publicly available detection tool, this work helps mitigate the malicious use of videos generated with AI.

1 Introduction

Deepfakes are images or videos in which a person's likeness is manipulated or replaced with another, typically using deep learning techniques. The rapid progress in generative models has made it very easy to produce hyper-realistic fake videos that are indistinguishable to the human eye. This raises ethical and social concerns: deepfakes have been used to spread false information, influence political processes, and create non-consensual explicit content. As noted by Rössler et al. [16], fake media can lead to a loss of trust in digital content and facilitate the spread of false information or fake news. Tackling these threats requires reliable deepfake detection methods.

Consequentially, deepfake detection has become a critical research area. The main challenge is to distinguish artifacts or inconsistencies in manipulated videos without being tricked by the increasing quality of generation methods. Early efforts showed that even when detection models achieve high accuracy on the data they were trained on, they often struggle to generalize to new types of deepfakes or different datasets.

We approach this problem by evaluating a range of architectures and training strategies, and by considering the practical aspects of deployment. We also incorporate XAI techniques, such as Grad-CAM [17], within the demo. The demo is available here.

2 Related Work

Early approaches leveraged Convolutional Neural Network (CNN) architectures such as Xception [4] and EfficientNet [18]. These models were typically trained on specific datasets to learn features directly from raw pixel.

When deepfake generation methods improved, spatial detectors emerged. These methods try to identify inconsistencies and artifacts within the spatial domain of an image. Examples include

techniques focusing on biological signals like heart rate [9]. While these methods provided interesting insights, many of these weaknesses have been addressed in newer deepfakes. Moreover, they highlight the importance of thinking beyond classification of raw pixels. An example is combining image analysis with audio analysis to detect mismatches in lip-sync. More recent spatial detectors, such as IID [10], focus on implicit identity features to spot inconsistencies. Studies found that a vanilla ViT, when trained from scratch on deepfake datasets, can underperform CNNs. This is likely due, as underlined by Nguyen et al. [14], to the fact that ViTs tend to focus on global structures and may fail to detect artifacts that CNNs instead catch. To overcome this, recent methods combine the strengths of CNNs and Transformers. For example, FakeFormer [14] extends a ViT with local attention mechanisms to force the model to attend to regions that are more prone to artifacts.

At the same time, frequency domain detectors have explored the analysis of image spectra. Methods like F3Net [15] and SPSL [12] are build on the premise that generative model may leave a sort signatures or high-frequency artifacts that are less visible in the spatial domain.

For video deepfakes, leverage temporal inconsistencies is a research direction. Video detectors analyze sequences of frames to identify anomalies over time. Early methods have used 3D CNNs (e.g., I3D [3]) or Recurrent Neural Networks. More recent video detectors include Transformer-based models like TimeTransformer [2] and VideoMAE [16] that capture long-range temporal dependencies.

The current State-of-the-Art (SOTA) in deepfake detection is focused on generalization across unseen manipulation techniques, datasets, and even beyond face-centric approach. A challenge has been the tendency of detectors to overfit to artifacts specific to the training data. To address this, methods like Latent Space Data Augmentation (LSDA) [19] aim to overcame the specificity of forgery by augmenting representations in the latent space. The latest advancements, exemplified by "Effort" [20], push the boundaries of generalization and efficiency. "Effort" proposes an orthogonal subspace decomposition method thought for generalizable detection of not only face deepfakes but also a wide range of AI-generated images, indicating a shift in the field.

In this context, our work builds on these insights by evaluating baseline CNNs and new architectures, and incorporating these new idea to try to train a detector for all the AI-generated synthetic images.

3 Datasets

We utilized publicly available datasets that have become standard benchmarks for training and evaluating deepfake detection models.

FaceForensics++ (FF++) FaceForensics++ is a dataset introduced by Rössler et al. [16] for the detection of facial manipulation. It contains 1,000 real videos collected from the Internet, each of which has been manipulated using four different methods: DeepFakes, FaceSwap, Face2Face, and NeuralTextures. In total, the dataset includes 4,000 fake videos. It is provided at multiple compression levels (raw, lightly compressed, and heavily compressed) to simulate real-world scenarios. Overall, FF++ offers over 1.8 million manipulated images (extracted frames) for training deepfake detection models. This dataset was also the one on which Xception was originally tested for this task, achieving over 85% accuracy on compressed videos. In our work, we used only the subset generated with the DeepFakes method.

Google DFD The Google Deepfake Detection Dataset (DFD), part of the FaceForensics++ benchmark, was created to advance research in detecting manipulated media. It comprises 363 original videos featuring paid actors across various scenes, alongside 3,068 corresponding deepfake videos.

DeepFake Detection Challenge (DFDC) The DFDC dataset [6] was released in conjunction with a global competition organized by Facebook (Meta) in 2019–2020. It is one of the largest deepfake video datasets, containing over 100,000 video clips. These videos were created using 3,426 paid actors and both GAN-based and traditional face-swapping methods. The DFDC dataset includes diverse settings, lighting conditions, and perturbations. Additionally, some videos contain random distortions or transformations to make detection more challenging. Due to its scale and diversity, DFDC presents a difficult test for the robustness of detection models. Training on DFDC from scratch is computationally expensive. We primarily use it for evaluation: after training models on Google DFD, we assess their performance on DFDC. It is worth noting that the winner of the DFDC challenge

achieved 82.56% accuracy on the public dataset but dropped to 65.18% on the unseen black-box data, highlighting the difficulty of the dataset. Top solutions combined ensembles of EfficientNet-B7 models with face-dropping augmentation and mixup techniques.

Celeb-DF Celeb-DF is a deepfake dataset of celebrity faces released in 2020 by Li et al. [11]. It addresses some limitations of previous datasets by providing high-quality fakes that are harder to detect. The dataset contains 590 real videos of celebrities and 5,639 deepfake videos. The fakes in Celeb-DF are visually very convincing, with fewer observable artifacts, making this dataset difficult for detectors; many models that perform well on FF++ drop significantly in accuracy on Celeb-DF. We included Celeb-DF as a test-only dataset for cross-dataset evaluation. This allows us to measure how well models trained on Google DFD can detect fakes in a different set of videos with different properties.

We used a portion of the data for validation and testing when training on Google DFD, specifically 15% for validation and 15% for testing. These splits were created by first reserving 30% of the data, then dividing it proportionally into validation and test set, while maintaining class stratification to ensure balanced labels. Each dataset was preprocessed to some extent. We extracted a fixed number of 30 frames per video, evenly spaced throughout the entire video duration. For video detectors, we treated consecutive frames as a sequence of frames.

4 Approach

We identified a set of representative architectures from different paradigms to compare:

- **Xception (2D CNN)**, a strong baseline [4].
- **Xception-3D (3D CNN)**, a custom 3D extension of Xception, applying 3D convolutions to consecutive frames to capture temporal features [1].
- **Video Transformer**, a transformer-based model for video that captures spatio-temporal features via attention. We implemented an architecture that processes patches from sequences of frames.

These models represent well-known approaches from literature (CNN vs Transformer, 2D vs 3D). We trained each from scratch on the training datasets and also experimented with fine-tuning using pre-trained weights when available.

4.1 Introducing MambaVision Model

After establishing our baselines, we decided to use MambaVision [8]. Mamba is a new architecture based on Selective State-Space Models (SSM), which has demonstrated strong performance in computer vision tasks while offering better computational efficiency compared to other competing models. NVIDIA's research team introduced MambaVision as a hybrid Mamba-Transformer backbone. The main idea is to use a state-space model to process local information and incorporate a few self-attention layers to capture global dependencies. The result is a hierarchical model designed to be both fast and accurate, pushing the Pareto frontier of accuracy versus throughput in image classification. We used the pre-trained MambaVision model from NVIDIA, available on HuggingFace as `nvidia/MambaVision-S-1K`, and fine-tuned it on our dataset.

4.2 Training

All models were trained using PyTorch. We used four NVIDIA A100 GPUs in parallel, which allowed us to use relatively large batch sizes. We used AdamW [13] as the optimizer. For fine-tuning MambaVision, we set the learning rate to 1e-4 for the head and 1e-5 for the backbone. To improve generalization, we applied augmentations such as random horizontal flips, vertical flips, and color jitter. These transformations were applied only to real samples. We trained our models on Google DFD. The training split is composed of 255 real videos and 2148 fakes at compression c23. We also tried training MambaVision on dataset obtained combining Google DFD and tiny portion of DFDC to increase diversity.

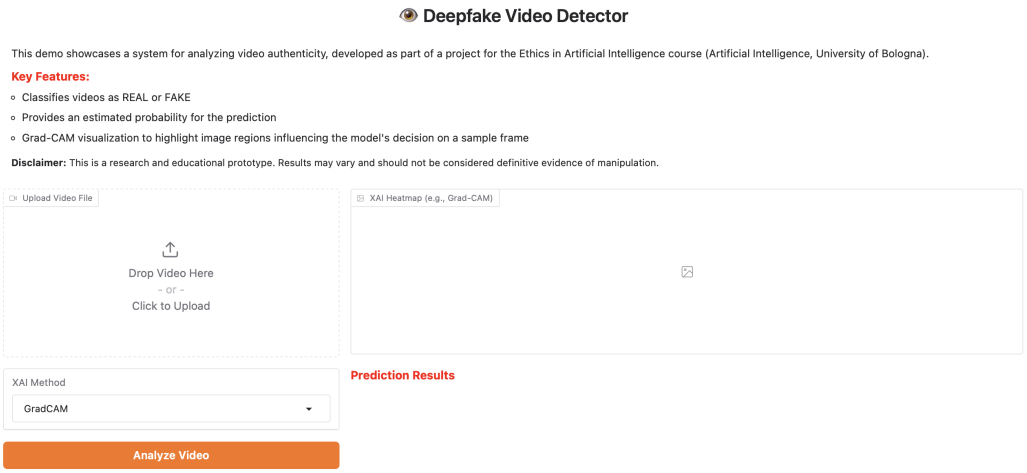


Figure 1: **UI of the demo**. The interface is developed using Gradio and allows a user to upload a video. A set of frames is then automatically extracted from the video. The model processes each frame individually to produce predictions, which are aggregated to determine the final result.

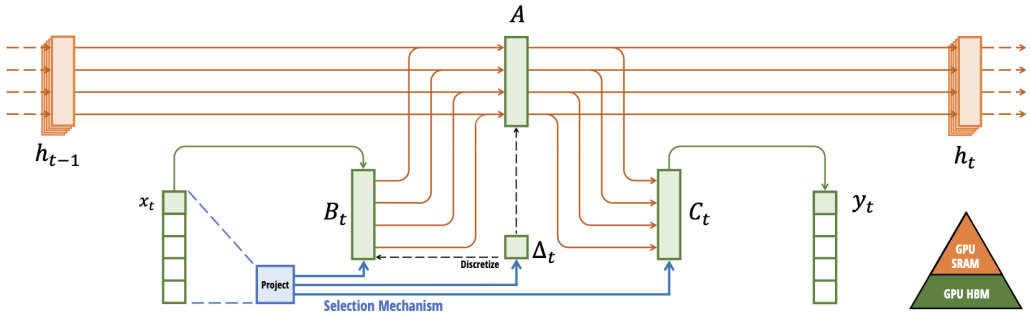


Figure 2: **Mamba SSM**

4.3 Evaluation Methodology

We evaluate model performance using two main setups. In the first, we train on Google DFD and test on FF++, DFDC, and Celeb-DF, none of which the model has seen during training. In the second setup, we expand the training set to include both Google DFD and DFDC, and again test on FF++, DFDC, and Celeb-DF, all treated as unseen during training. We report Accuracy and AUC as our evaluation metrics 1.

4.4 Prototype System

We built a minimal web interface using Gradio. The interface allows a user to upload a video file; we then feed it to our trained model, which returns a probability and a label indicating *Real* or *Fake*. Under the hood, the video is split into frames. This demo serves as a proof-of-concept for how detection could be deployed. For instance, a social media platform could integrate such a model to flag potentially fake videos. In the interface, we underline that the model's output is an aid, so it should be used with judgment.

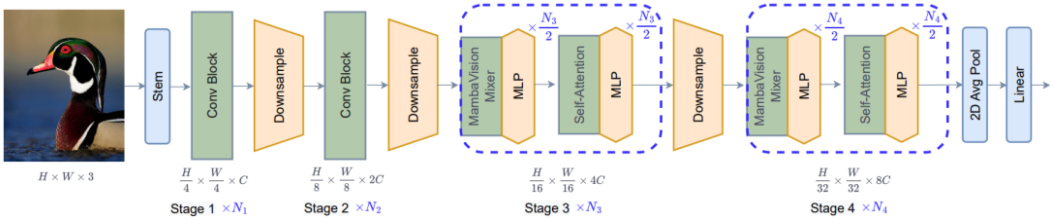


Figure 3: **Mamba Vision**

5 Mamba Solution

5.1 How Mamba Works

Mamba [7] is based on the concept of state-space models (SSM). When dealing with sequences, typically recurrent neural networks (RNNs) or Transformers are used. State-space models offer an alternative to these approaches by defining a state that can be computed in an efficient way, even for long sequences, using linear recurrence and global convolution. The Mamba architecture is essentially an efficient SSM combined with modern deep learning practices. This architecture intrinsically operates with sub-quadratic complexity with respect to the sequence length, unlike standard self-attention, which is quadratic. It incorporates skip connections and a selection mechanism that works similarly to gating, allowing control over the information flow. Additionally, it was designed to be hardware-friendly, with an implementation leveraging GPU acceleration, following a similar approach to FlashAttention [5]. In simpler terms, a Mamba block takes as input a sequence and processes it through an internal state that can carry information. This allows it to capture temporal patterns very efficiently. One reason Mamba is interesting is its efficiency. A self-attention layer with a sequence of length T requires $O(T^2)$ memory for the attention matrix, which, for example, when $T = 100$ can be quite heavy. Mamba's state-space requires only $O(T)$ memory, allowing it to handle longer sequences or higher-resolution data.

5.2 How Vision Mamba Works

Vision Mamba (often referred to as MambaVision) [Figure 3] is the adaptation of the Mamba architecture for computer vision tasks. Since images are two-dimensional data, Vision Mamba replaces the 1D causal convolution used in the original Mamba with regular 1D convolution: early stages use CNNs to extract local features from images, while later stages incorporate MambaVision Mixer blocks. To improve the modeling of long-range spatial dependencies, MambaVision integrates self-attention (Transformer) blocks into the final layers of these stages. According to NVIDIA's report, equipping the Mamba architecture with several self-attention blocks significantly enhances its ability to capture long-range spatial relationships. MambaVision is available in different model sizes. We used a MambaVision-S model, pretrained on ImageNet-1K. Each frame extracted from a video is treated as a separate image, processed through Vision Mamba, and the predictions are then aggregated.

6 Problems and Challenges

6.1 Dataset Imbalance

Deepfake datasets often have an imbalanced class distribution. In our case, the fake class was the majority. Such imbalance can bias a model toward trivial solutions. For instance, a model that always predicts fake may achieve high accuracy if most samples are fake, while still having a high false positive rate.

Downsampling the majority class (fake). We reduced the number of fake samples in each training epoch to match the number of real samples by random selection. While this approach balances class counts, it has a major drawback: we discard a lot of fake data that could have been informative.

Oversampling or Augmenting the minority class (real). We tried oversampling real videos, repeating them many times during training. We also applied augmentations to real videos (e.g., random rotations, flips, small color variations) to increase their diversity. This approach ensured the model saw real examples by upsampling them to match a subset of the fake samples, leading to a balanced 1:1 ratio. However, this made training slower and brought to a loss in performances due to the many repeated samples.

Class-weighted Loss. We modified the loss function to penalize mistakes on real and fake classes differently.

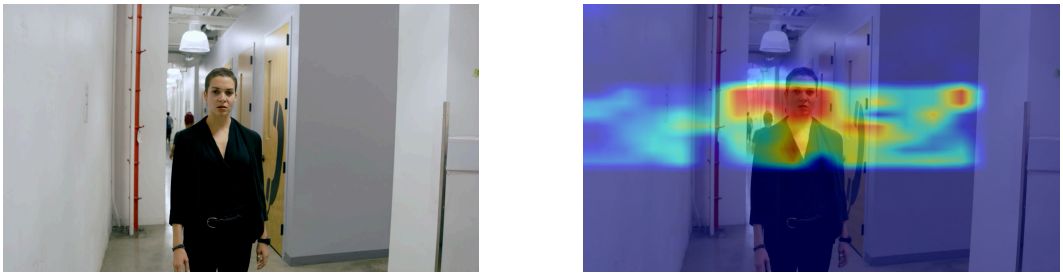
We found that what worked best was a combined approach: downsampling the fake class to achieve a 1:1 ratio, plus a weighted loss. The downsampling was performed by sampling an equal number of elements from each class in every batch, based on the original data distribution. The best-performing Mamba model was trained using this strategy.

6.2 Model Size and Complexity

Some of the models we experimented with are quite large in terms of parameters. Training such large models on high-resolution video data is both memory-intensive and time-consuming. Although we had access to four A100 GPUs (each with 40 GB of memory), naive training could still exhaust memory or take too long. For example, our initial Video Transformer model had approximately 100 million parameters, and feeding 16 frames of 224×224 resolution at once required reducing batch sizes to one or two per GPU, which slowed training.

We found MambaVision converging in just 40 minutes using BF16 mixed precision. In contrast, the 3D Xception model, with 15 million parameters, needed over 3 hours of training from scratch, and even the 105M parameter VideoTransformer took 2 hours to converge. Given the 3D Xception performance and seeing Mamba outperform it after 40 minutes, we ultimately dropped it in favor of our more efficient approach.

The Mamba model was optimized at the CUDA level. NVIDIA reported that MambaVision achieves state-of-the-art performance in terms of both accuracy and throughput, which we can confirm. This allowed for more iterations per second and thus faster progress during training within the same amount of time.



(a) Fake image (b) Grad-CAM overlay

Figure 4: **Side-by-side comparison of (a) a fake image from the FF++ dataset and (b) the corresponding Grad-CAM overlay.** The Grad-CAM highlights regions that the deepfake detection model focuses on to distinguish manipulated content from real.

7 Results

7.1 In-Dataset Performance

Xception. Our implementation of Xception achieved about 80.5% auc on the test set (c23 compression), with an AUC of 0.695. It indicates that Xception can fit the training data and detect the fakes it was trained on. This number it is more a confirmation that our training worked.

Model	Train Data	Test Data	Accuracy	AUC
Xception (2D CNN)	DFD	DFD (same)	80.5%	0.695
Video Transformer	DFD	DFD (same)	89.3 %	0.520
Mamba (Vision)	DFD	DFD (same)	89.2%	**0.973**
Xception (2D CNN)	DFD	Celeb-DF (cross)	67.5%	0.583
Mamba (Vision)	DFD	Celeb-DF (cross)	56.3%	**0.757**
Xception (2D CNN)	DFD	DFDC (cross)	62.0%	0.513
Mamba (Vision)	DFD	DFDC (cross)	47.4%	**0.797**
Xception (2D CNN)	DFD	FF++ (cross)	56.4%	0.640
Mamba (Vision)	DFD	FF++ (cross)	83.3%	**0.921**
Mamba (Vision)	DFD + DFDC mix	Celeb-DF (cross)	58.7%	0.704

Table 1: **Performance comparison across models and datasets.** Accuracy for cross-dataset tests is at a chosen threshold, while AUC is threshold-independent. Best values are in bold.

3D Xception. The 3D version also achieved roughly 89.3% accuracy and an AUC of 0.573. It did not significantly surpass the 2D models. This suggests that for the Google DFD fakes, which often have consistent artifacts per frame, the temporal component didn't add much information. Training it was more computationally expensive, which is why we ultimately did not focus on it.

Video Transformer (ViT-based). When trained on Google DFD, the video transformer processing a sequence of 16 frames per video achieved about 89.3% accuracy and an AUC of approximately 0.51. The ViT tended to train a bit slower and needed more regularization. It might also have focused on global features and sometimes missed fine local artifacts. With more hyperparameter tuning, ViT could potentially match CNNs.

MambaVision. Fine-tuned on Google DFD, the Mamba model got to about 89.2% accuracy (AUC 0.973) on DFD test. Mamba's training curve showed it converged quickly to a good solution, thanks to its pretraining.

7.2 Cross-Dataset Generalization

Train on DFD → Test on Celeb-DF. For Xception, the AUC dropped to around 0.583 on Celeb-DF. This aligns with the findings of Celeb-DF's authors, who noted that detectors struggle on their data. Mamba, when fine-tuned, achieved the best cross-dataset performance, with an AUC of about 0.757 on Celeb-DF. While this is still far from perfect, it represents a clear improvement over the baseline. We attribute this to two factors: (a) pretraining on ImageNet gave Mamba more general features, maybe recognizing inconsistencies that a model trained only on FF++ might miss; (b) Mamba's architecture, being a sequence model, didn't overfit to specific artifacts. In terms of accuracy, Mamba reached approximately 56.3% on Celeb-DF, whereas Xception is around 67.5%.

Train on DFD → Test on DFDC We used a tiny portion of DFDC for evaluation. For Xception, we obtained an AUC of around 0.583 on DFDC, which is worse than on Celeb-DF. Mamba achieved about 0.797 AUC on DFDC.

Train on mixed data → Test on another. We also tried training on a mix of Google DFD and DFDC, then testing on Celeb-DF. The idea was to increase diversity in the training set to improve generalization. Mamba, with mixed training, reached about 0.704 AUC on Celeb-DF.

Overall, the MambaVision model yielded the highest cross-dataset performance among the models we tried, fulfilling our goal of improving generalizability.

8 Conclusion

In this project, we explored several methods for deepfake video detection, implementing a range of models: from CNNs to Vision Mamba. Our efforts led to a prototype deepfake detector that not only

performs well on benchmark datasets but also shows generalization to unseen videos. We confirmed that CNN-based models remain strong baselines for deepfake detection, reaching good accuracy on datasets like DFD. However, these models can overfit to specific artifacts; a detector is only as good as the variety of fakes it has seen. We demonstrated that newer architectures can better equip the model for generalization. In particular, the fine-tuned MambaVision model achieved the best cross-dataset results. This highlights the value of integrating state-space sequence modeling with transformer-style attention for deepfake detection, as well as the benefit of transfer learning from large-scale data. Our best model's success was not only due to its architecture but also the result of careful data sampling. Our prototype system also aims to make verification accessible to everyone. A journalist, fact-checker, or platform moderator could use such an interface to quickly examine videos. That being said, we must remain cautious: no AI system is infallible. As our detector improves, deepfake creators will find new ways to evade detection. Our detector must continuously evolve.

Possible lines of research could include improving generalization through domain adaptation and creating training data with greater variety, as well as extending detection to multimodal features like audio-visual consistency and temporal coherence. Efficiency is also key for deployment at scale, suggesting model distillation and smart frame selection. Moreover, the focus on ethics remains essential: working on how to present results in a more informative way and how to incorporate some form of watermarking for authentic videos.

Limitations

MambaVision currently requires CUDA to work, so this model may be unsuitable in scenarios where a GPU is not available.

Since we used a custom VideoTransformer and 3D Xception model, no pre-trained weights exist. This complicate training. There are techniques such as inflating the weights of Xception to 3D Xception or trying to leverage features from some pre-trained Vision Transformers (ViTs) on video datasets like Kinetics.

References

[1] Amil Khan (amilworks). 3D-Xception: Pytorch implementation of 3d-xception network for video classification. `https://github.com/amilworks/3D-Xception`, 2019. Software repository, accessed June 23, 2025, MIT License.

[2] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *ICML*, 2021.

[3] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[4] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016. URL `http://arxiv.org/abs/1610.02357`.

[5] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022. URL `https://arxiv.org/abs/2205.14135`.

[6] Brian Dolhansky, Joanna Bitton, Ben Pflaum, Jikuo Lu, Russ Howes, Menglin Wang, and Cristian Canton Ferrer. The deepfake detection challenge (dfdc) dataset, 2020. URL `https://arxiv.org/abs/2006.07397`.

[7] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024. URL `https://arxiv.org/abs/2312.00752`.

[8] Ali Hatamizadeh and Jan Kautz. Mambavision: A hybrid mamba-transformer vision backbone, 2025. URL `https://arxiv.org/abs/2407.08083`.

[9] Javier Hernandez-Ortega, Ruben Tolosana, Julian Fiérrez, and Aythami Morales. Deepfakesonphys: Deepfakes detection based on heart rate estimation. *CoRR*, abs/2010.00400, 2020. URL https://arxiv.org/abs/2010.00400.

[10] Baojin Huang, Zhongyuan Wang, Jifan Yang, Jiaxin Ai, Qin Zou, Qian Wang, and Dengpan Ye. Implicit identity driven deepfake face swapping detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4490–4499, 2023.

[11] Yuezun Li, Xin Yang, Pu Sun, Honggang Qi, and Siwei Lyu. Celeb-df: A large-scale challenging dataset for deepfake forensics, 2020. URL https://arxiv.org/abs/1909.12962.

[12] Honggu Liu, Xiaodan Li, Wenbo Zhou, Yuefeng Chen, Yuan He, Hui Xue, Weiming Zhang, and Nenghai Yu. Spatial-phase shallow learning: rethinking face forgery detection in frequency domain. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 772–781, 2021.

[13] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017. URL http://arxiv.org/abs/1711.05101.

[14] Dat Nguyen, Marcella Astrid, Enjie Ghorbel, and Djamila Aouada. Fakeformer: Efficient vulnerability-driven transformers for generalisable deepfake detection, 2024. URL https://arxiv.org/abs/2410.21964.

[15] Yuyang Qian, Guojun Yin, Lu Sheng, Zixuan Chen, and Jing Shao. Thinking in frequency: Face forgery detection by mining frequency-aware clues. In *European conference on computer vision*, pages 86–103. Springer, 2020.

[16] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics++: Learning to detect manipulated facial images, 2019. URL https://arxiv.org/abs/1901.08971.

[17] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016. URL http://arxiv.org/abs/1610.02391.

[18] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019. URL http://arxiv.org/abs/1905.11946.

[19] Zhiyuan Yan, Yuhao Luo, Siwei Lyu, Qingshan Liu, and Baoyuan Wu. Transcending forgery specificity with latent space augmentation for generalizable deepfake detection. *arXiv preprint arXiv:2311.11278*, 2023.

[20] Zhiyuan Yan, Jiangming Wang, Peng Jin, Ke-Yue Zhang, Chengchun Liu, Shen Chen, Taiping Yao, Shouhong Ding, Baoyuan Wu, and Li Yuan. Orthogonal subspace decomposition for generalizable ai-generated image detection, 2025. URL https://arxiv.org/abs/2411.15633.

A Hyperparameters

Hyperparam	**MambaVision**
Card	`nvidia/MambaVision-T-1K`
Input	540×720
Downsample	`True`
Optimizer	AdamW
Learning Rates	1e-5 (backbone), 1e-4 (head)
Weight Decay	0.01
Batch Size	16
Max Steps	89 700
Precision	bf16-mixed
Seed	42

Table 2: Hyperparameters of our model.