Supplementary Material

A Datasets

A.1 DeepFake Detection Challenge

The DFDC dataset was released as part of the homonymous Kaggle challenge [21]. It contains approximately 120k videos, the majority of which are DeepFakes created with different manipulation methods. The dataset comes in different variants of compression quality, which are mostly relevant for training robust classifiers and assessing detection performance. For an analysis of explanation quality, we choose to work on high-quality videos where we expect manipulation artifacts to be more prominent. Table 2 reports how many videos are used for training, validation, testing and explanation evaluation.

Preprocessing. The videos from the dataset might contain one or more faces. Of these, only one is manipulated in the case of "fake" videos. For the purpose of training the DeepFakes classifier, each video is preprocessed as follows:

1. Videos are spatially resized with padding so that each frame is 640×640 pixels;
2. MTCNN is applied every 5 frames, which outputs rectangular bounding boxes tightly cropped around all faces in a frame
3. Face detections are linked across frames using a greedy overlap-based heuristic; namely, if two bounding boxes overlap with IoU > 0.5 they are considered the same face;
4. The longest consecutive sequence of linked boxes is considered the main face and is assumed to be the target for DeepFake manipulation, all other boxes are discarded and the video is clipped to the frames containing the main face;
5. For intermediate frames where MTCNN was not applied, a bounding box for the main face is created by linearly interpolating the corners of the two closest boxes;
6. Boxes are expanded by $1.5\times$ to capture more of the hair, neck and background
7. All frames belonging to the main face sequence are cropped according to their box; rectangular crops are resized to 224×224 and used for training;
8. BiSeNet is applied every 5th frame of a 512×512 version of the aforementioned video, the probabilistic output of BiSeNet is resized with bilinear interpolation to match the original size and then the most likely face part is selected.

Classification. For training, validation, and testing, each video is processed separately. This means that fake videos will not be perfectly aligned with the corresponding real video, neither in space nor in time. Also, it means that detection and parsing might fail on some fake videos, due to the low quality of the manipulation. While this can hinder training, it also represents a realistic scenario where unseen videos are submitted to a trained classifier.

Expalantion evaluation. Explanation metrics based on manipulation detection require perfectly-aligned pairs of real-fake videos. Therefore, face detection and parsing are performed on real videos and applied identically to all corresponding fake videos. Since the pairing between real and manipulated videos is only given for the training split of DeepFake Detection Challenge, and only some videos are perfectly aligned, we use an held-out subset of the training split consisting of 230 fake videos created from 100 real videos.

A.2 DeepFake Detection Dataset

The DeepFake Detection Dataset [57] is constructed by recording actors in various situations and then applying face-swapping DeepFake techniques to the videos. The dataset contains a large number of videos, but only a limited number of them are perfectly aligned and can be used for evaluating explanations on part-based manipulation detection. With respect to the classification task, this dataset is never used during training and it represents a good benchmark for out-of-distribution generalization. Table 2 reports the number of videos used to evaluate classification performance and explanation metrics. The videos are preprocessed in the same way as DFDC. For manipulation detection, perfect alignment is available for 107 fake videos created from 37 real ones.

Table 2: **Dataset sizes:** number of real and fake videos contained in each dataset and split. DFDC is used to train all classifiers in this work, to report classification metrics, and to evaluate explanation metrics. DFD is only used for testing and explanation evaluation.

	DFDC		DFD	
	Real	Fake	Real	Fake
Train	19143	99953	-	-
Validation	1975	1968	-	-
Test	2479	2486	37	107
Explanation	100	230	37	107

B Classification performance

We report relevant classification metrics for the test split of DFDC in Table 3 and for a subset of DFD in Table 4. In addition to cross-entropy loss ($\mathcal{L}_{\text{CE}}$) and area under the receiver operating characteristic curve (A_{ROC}), we also report precision, recall, and F1 scores obtained when the model output is binarized with a threshold of 0.5. Furthermore, we report average precision (AP), *i.e.* the area under the precision-recall curve as the classification threshold changes. For each metric and configuration described in Section 3.2 and Section 4, we report mean and standard deviation of 3 runs.

Table 3: Classification metrics for DFDC, test split. For each configuration and metric, mean and standard deviation of 3 runs are reported. For all metrics except the $\mathcal{L}_{\text{CE}}$ loss, values are given in percentage and a higher value indicates a better result.

	$\mathcal{L}_{\text{CE}} \downarrow$		Precision ↑		Recall ↑		F1 ↑		AP ↑		$A_{\text{ROC}} \uparrow$	
	avg	std	avg	std	avg	std	avg	std	avg	std	avg	std
S3D Baseline	0.447	0.036	80.89	1.82	82.27	4.01	81.50	1.17	88.79	2.19	89.02	1.08
S3D Bilateral	0.696	0.003	54.00	0.15	32.06	8.47	39.91	6.74	52.75	0.58	54.24	0.67
S3D Gaussian	0.542	0.031	78.49	1.51	64.26	2.89	70.65	2.16	80.20	3.30	81.77	2.33
S3D TV Loss	0.460	0.027	78.68	1.53	81.04	3.50	79.82	2.12	88.24	1.78	87.41	1.88
S3D Cutout	0.481	0.037	78.39	0.73	82.72	4.10	80.46	1.87	86.42	3.56	87.19	2.14
MViT	0.430	0.004	83.64	0.28	94.30	1.46	88.65	0.57	96.59	0.32	96.38	0.38

Table 4: Classification metrics for DFD. For each configuration and metric, mean and standard deviation of 3 runs are reported. For all metrics except the $\mathcal{L}_{\text{CE}}$ loss, values are given in percentage and a higher value indicates a better result.

	$\mathcal{L}_{\text{CE}} \downarrow$		Precision ↑		Recall ↑		F1 ↑		AP ↑		$A_{\text{ROC}} \uparrow$	
	avg	std	avg	std	avg	std	avg	std	avg	std	avg	std
S3D Baseline	0.694	0.080	72.48	5.55	61.64	2.81	66.59	3.73	82.88	2.67	80.24	2.31
S3D Bilateral	0.746	0.006	42.37	0.56	26.97	2.51	32.94	2.05	43.90	0.00	45.82	0.29
S3D Gaussian	0.760	0.055	60.51	1.77	49.03	12.73	53.66	8.34	66.03	2.37	66.42	0.90
S3D TV Loss	0.698	0.038	65.21	2.03	66.79	7.17	65.90	4.25	77.66	3.72	75.75	3.59
S3D Cutout	0.655	0.065	72.48	5.26	59.95	4.25	65.44	1.97	82.23	0.91	79.59	0.48
MViT	0.513	0.015	74.75	1.24	83.41	3.50	78.83	2.12	91.73	1.70	90.04	1.79

C Training details

C.1 High-frequencies smoothing

For models trained with smoothing preprocessing, either Gaussian blur or bilateral filtering are applied. Gaussian blur is applied at the video level using a spatial standard deviation of 0.8 and a temporal standard deviation of 0.5. Bilateral filtering is applied per-frame using a spatial standard deviation of 2 and a color range standard deviation of 0.1. These values are empirically chosen so that the filtered videos remain qualitatively similar to the original ones and that common DeepFake artefacts are still visible.

C.2 Default video augmentation

For all models, color augmentations are applied to training videos to improve generalization. Each video is augmented with probability 0.5. If the video is augmented, one of the following transformations is chosen with equal probability: grayscale conversion, RGB shifting, gamma shifting, contrast limited adaptive histogram equalization, hue, saturation and value shifting, brightness and contrast shifting.

C.3 Cutout

When cutout is enabled, each video is augmented with cutout with probability 0.5. Cutout acts on a 64×64 region of the video selected with uniform probability. Once a mask is selected, its contents are blurred with a strong Gaussian filter (standard deviation 4).

C.4 Architectures and training details

Multiscale S3D. The S3D architecture consists of several inception blocks using separable 3D convolution layers. We use a model pretrained on Kinetics 400 as the backbone feature extractor for the DeepFake classifier. On top of the original architecture, we add shortcut connections from intermediate layers to the classification head,to allow easier access to multiscale features which might be relevant for the task. Specifically, we collect the input activations of the 2nd, 3rd, 4th and 5th pooling layers. These activations are first average-pooled to the size of the smallest one, concatenated, processed through $1 \times 1 \times 1$ convolution, and eventually pooled into a 128-dimensional feature vector. The classification head is a simple 2-layer MLP with output size of 2 and softmax activation.

During one epoch of training, real videos are sampled more than once to match the number of fake videos in the training set. From each video, a clip of 64 consecutive frames is extracted at random. Videos shorter than 64 frames are padded by appending black frames. No spatial cropping is performed since the videos already contain centered faces. The optimizer processes mini-batches of 32 videos at the time. The learning rate of the Adam optimizer is set to 10^{-4} for pretrained parameters and to 10^{-3} for the classification head. An additional weight decay loss is applied to all parameters except biases with strength 10^{-5}.

For validation, only the first 64 frames of each video are considered and no augmentations are applied. Training runs for 5 epochs, unless validation loss stops decreasing, in which case early stopping is applied. The results reported in the tables are relative for videos in the training set. For these, we average the output probabilities of 5 equally-spaced 64-frames clips from each video.

Multiscale ViT. As an alternative to the 3D CNN backbone, we experiment with a transformer architecture. Specifically, we use a Multiscale Vision Transformer that combines attention layers with multiscale hierarchical processing. For this model, we maintain the original architecture except for the classification head that is modified to output 2 classes. Similarly to S3D, the weights are initialized from a model pretrained on Kinetics 400.

Training, validation and testing follow the default settings from the authors. Namely, the learning rate follows a cosine annealing schedule without warm-up, random temporal crops of 16 frames are selected from each training video, multiple temporal crops are considered for testing, spatial cropping is disabled.

Compute resources. All models are trained on a single machine equipped with 4 NVIDIA V100 GPUs with 32GB of RAM each, which allow for large batch sizes. Once trained, the model can be ran for both inference and explanations on more modest hardware, *e.g*. a single GPU environment with 12GB of RAM.

D Explanation metrics

This section details how the explanation metrics introduced in Section 3.2 are computed in practice using discretized videos, masks and heatmaps. Furthermore, Table 5 and Table 6 report the metrics for all variations considered in this work as the average and standard deviation of 3 runs each.

The main text uses a compact notation where videos are defined as a mapping from the discretized grid $\mathcal{G} = \{1,\dots,T\} \times \{1,\dots,H\} \times \{1,\dots,W\}$ to RGB pixel values. For this appendix, we choose a more explicit notation based on tensors. We remark the equivalence between the two notations since any function $f : \mathcal{G} \to R^+$ can be uniquely represented as a $T \times H \times W$ tensor whose element at (t,h,w) is $f(t,h,w)$. In this context, it is useful to define the derivative and integral operators ∇ and $\int$ as:

$$(\nabla f(\boldsymbol{\rho}))_{i=1,2,3} = f(\boldsymbol{\rho} + \boldsymbol{e}_i) - f(\boldsymbol{\rho}), \tag{6}$$

$$\int_{\mathcal{G}} f d\lambda = \frac{1}{THW} \sum_{\boldsymbol{\rho} \in \mathcal{G}} f(\boldsymbol{\rho}), \tag{7}$$

where the vector $\boldsymbol{\rho} = (t,u,v)$ denotes the pixel coordinates, and the vectors $\boldsymbol{e}_i$ are the usual orthonormal basis i.e. $(e_i)_j = \delta_{ij}$.

D.1 Total variation

Total variation is used to measure the smoothness of a heatmap and is defined as:

$$\mathrm{TV}(h) = \frac{1}{THW} \sum_{\boldsymbol{\rho} \in \mathcal{G}} \nabla h(\boldsymbol{\rho}), \tag{8}$$

where the discrete gradient ∇h at coordinates $\boldsymbol{\rho} = (t,u,v)$ is computed as:

$$|h(t,u,v) - h(t+1,u,v)| + |h(t,u,v) - h(t,u+1,v)| + |h(t,u,v) - h(t,u,v+1)| \tag{9}$$

D.2 Variance volume

To measure the spatial localization of the heatmap, we first compute its mean and variance:

$$\boldsymbol{\mu} = \sum_{\boldsymbol{\rho} \in \mathcal{G}} \boldsymbol{\rho} h(\boldsymbol{\rho}), \tag{10}$$

$$\boldsymbol{\Sigma} = \sum_{\boldsymbol{\rho} \in \mathcal{G}} (\boldsymbol{\rho} - \boldsymbol{\mu})(\boldsymbol{\rho} - \boldsymbol{\mu})^T h(\boldsymbol{\rho}), \tag{11}$$

where the vector $\boldsymbol{\rho} = (t,u,v)^T$ represents pixel coordinates. Then, to summarize the 3×3 variance matrix as a scalar, we consider its volume given by the determinant $|\det(\boldsymbol{\Sigma})|$. Larger volumes correspond to more spread out heatmaps, while smaller values indicate more localized explanations. Importantly, this metric is particularly indicated for unimodal heatmaps that focus around a single location of the video.

D.3 Gini Index

The Gini Index was initially introduced as an economic indicator of wealth distribution [29], but it is considered a good measure of sparsity due to its properties [31]. The Gini Index of an heatmap measures is defined as:

$$\mathrm{G} = \frac{2}{THW} \frac{\sum_i i \cdot h(\boldsymbol{\rho}_i)}{\sum_i h(\boldsymbol{\rho}_i)} - \frac{THW+1}{THW}, \tag{12}$$

with the indices $i = 1, \ldots, THW$ that select pixel coordinates such that $h(\boldsymbol{\rho}_i) \leq h(\boldsymbol{\rho}_{i+1})$. Notably, the "sparsity" measured by the Gini Index refers to the scalar importance values of each pixel and not their location in the heatmap. The heatmap will have a high Gini Index if most of the explanation mass is concentrated in few highly-relevant pixels while all other pixels have low relevance.

D.4 Faithfulness

Faithfulness is generally used to compare explanation methods on the basis of how closely they identify portions of the input that are meaningful for the classifier and a particular decision. Faithfulness is measured using the *deletion score*, which represents the area under the curve traced by the confidence in $p(\text{FAKE}|v)$ as pixels are removed from the video in decreasing order of relevance. Considering the large amount of pixels in a video, the curve is approximated by removing several pixels in one step. Specifically, we consider the sorted values of an heatmap h and group them in 25 bins. These bins do not necessarily contain the same amount of pixels, but the total relevance in each bin is approximately the same.

In this work, we are interested in the properties of explanations rather than of explanation methods. However, a faithful explanation method is a prerequisite for further evaluation of explanation quality. As a preliminary step, we compare the four explanation methods discussed in Sec. 3.1 and choose the most faithful one based on its deletion scores on the baseline model. The following hyperparameters are used: in SmoothGrad, gradients are averaged over 25 randomly perturbed videos with a noise parameter equal to 0.15 of the RGB range; in Integrated Gradients the path integral is calculated *w.r.t.* a black video baseline using 25 interpolation steps.

With respect to fake videos in DFDC and DFD, the four methods achieve the following average deletion scores: Sensitivity 42.54%, Gradient$\times$Input 43.68%, SmoothGrad 41.25%, Integrated Gradients 43.77%. Therefore, SmoothGrad has the lowest deletion score of the four (p value of paired one-sided t-test $< 10^{-5}$) and is then employed throughout all experiments.

D.5 Manipulation detection

The ability to detect and localize manipulations is measured by considering videos created by overlaying a portion of a fake video v_F to its corresponding original video v_R. The blending mask is obtained by selecting a face part $p \in \{\text{mouth}, \text{nose}, \text{eyes}\}$ from those extracted with BiSeNet [20]. The explanation heatmap can then be compared with the ground-truth manipulation mask to determine whether the model is focusing on manipulated regions of the video. Without loss of generality, only the first 64 frames of each video are considered in all manipulation detection evaluations.

For the main metrics, we measure the precision of 100 most-relevant pixels in the heatmap, and the percentage of heatmap contained in the ground-truth manipulation mask. This approach has the advantage of being threshold independent, as opposed to binarizing the heatmap using an arbitrary threshold and computing metrics such as Intersection over Union. Notably, both metrics penalize explanations that focus outside the ground-truth mask, but can not distinguish whether the heatmap clusters around an actual manipulation artefact or is uniformly scattered inside the mask (Figure 6).

Table 5: Explanation metrics for DeepFake Detection Challenge (DFDC). Mean and standard deviation of 3 independent classifiers whose decisions are explained using SmoothGrad.

	TV ↓		$\sqrt[3]{\sigma}$ ↓		Gini ↑		M_{IN} ↑		P_{100} ↑	
	avg	std	avg	std	avg	std	avg	std	avg	std
S3D Baseline	0.285	0.006	814.4	31.0	74.93	1.00	17.43	0.38	29.43	1.34
S3D Bilateral	0.428	0.044	1372.6	268.8	67.89	3.50	12.15	3.32	13.47	8.77
S3D Gaussian	0.292	0.004	841.4	15.1	75.55	0.44	17.38	0.51	26.31	1.19
S3D TV Loss	0.256	0.003	726.3	54.8	77.40	1.97	17.68	1.00	34.46	1.85
S3D Cutout	0.296	0.005	839.0	40.2	75.26	1.40	17.76	0.63	30.48	1.10
MViT	0.246	0.013	808.3	17.1	80.42	0.36	22.11	0.64	36.03	3.43

Table 6: Explanation metrics for DeepFake Detection Dataset (DFD). Mean and standard deviation of 3 independent classifiers whose decisions are explained using SmoothGrad.

	TV ↓		$\sqrt[3]{\sigma}$ ↓		Gini ↑		M_{IN} ↑		P_{100} ↑	
	avg	std	avg	std	avg	std	avg	std	avg	std
S3D Baseline	0.261	0.009	827.6	34.4	74.84	0.74	17.09	0.15	28.62	0.80
S3D Bilateral	0.413	0.061	1426.3	346.3	68.92	4.95	13.47	4.25	16.10	16.48
S3D Gaussian	0.273	0.011	799.2	11.0	76.54	0.17	16.83	0.82	26.87	1.16
S3D TV Loss	0.244	0.005	742.1	45.3	77.08	1.70	17.20	0.92	30.09	2.72
S3D Cutout	0.274	0.004	838.7	46.5	75.33	1.47	16.87	0.31	29.22	0.96
MViT	0.250	0.016	834.5	27.3	80.02	0.10	20.35	0.50	29.81	3.06

E Additional examples

Additional examples of semantic parsing, manipulation detection (Section 3.2.2), and explanation post-processing for the user study (Section 4.2) are shown below.

Figure 4: User study visualization: real video, fake video, enhanced heat-map, Gaussian matching, blob detection, semantic aggregation.

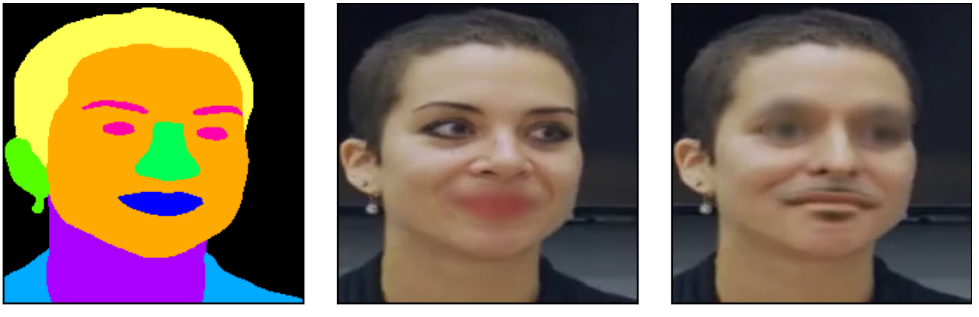

Figure 5: Example of semantic parsing as performed by BiSeNet [90] and of an alternative version of video cutout (Section 3.2.2) where heavy blurring is applied dynamically to a semantic region instead of a fixed square.

Figure 6: Additional examples for manipulation detection. Random frames from random videos. From left to right: original, part-based manipulated video, heatmap.