VideoFACT: Detecting Video Forgeries Using Attention, Scene Context, and Forensic Traces
Supplementary Material

Tai D. Nguyen Shengbang Fang Matthew C. Stamm
Drexel University
Philadelphia, PA, USA
{tdn47, sf683, mcs382}@drexel.edu

A. Dataset Creation

A.1. Set A: Standard Video Manipulations Datasets

In this section, we presents more details regarding the creation of the Standard Video Manipulations datasets used in this paper, specifically, those used for training and validation of our network - VCMS, VPVM, VPIM.

In order to generate samples for each of these datasets, we first needed to make binary masks which specify the to-be-manipulated regions in video frames/images. To facilitate this process, we created a library of 10 basic shapes: rectangle, circle, ellipse, triangle, pentagon, heptagon, 5-pointed star, 8-pointed star, 12-pointed star, and 18-pointed star. Since we wanted to avoid our network overfitting to any particular shapes, we further constructed complex compound shapes by overlapping up-to three basic shapes in a random manner. After the compound shapes for each mask were finalized, it was randomly resized, randomly rotated, and translated to a random location inside the mask's boundary (1080×1920). We note that masks with the manipulated area greater than 75% of the total area were rejected and regenerated using the same procedure.

With the binary masks ready, we then applied either splicing operations (copying content from a source video/image and pasting it into a destination video/image), or in-place editing operation (manipulating each frames of a video or image). With regards to in-place editing operations, we manipulated the regions specified by the mask with at least one of the following operations: changing brightness, contrast, saturation, and hue, adding random Gaussian blur, random motion blur, random box blur, and random Gaussian noise. We used the open-source differentiable computer vision library Kornia to perform these edits. The parameters used for the perceptuall visible datasets (VPVM) and the perceptually invisible datasets (VPIM) are listed in the Table B1.

Finally, we re-encoded the authentic video frames of each authentic video into the set of authentic videos and the manipulated videos frames of each manipulated video into the set of manipulated videos. All videos were re-encoded as H.264 videos using FFmpeg with the constant rate factor of 23 and the frame rate of 30 FPS.

Visible?	Manip. Type	Manip. Parameters	Manip. Prob.
Yes	Brightness	`range=[0.8, 1.6]`	1.0
	Contrast	`range=[0.7, 1.3]`	1.0
	Saturation	`range=[0.8, 1.1]`	1.0
	Hue	`range=[-0.2, 0.2]`	1.0
	Gaussian Blur	`kernel_size=(5,5), sigma=(2,2)`	0.7
	Motion Blur	`kernel_size=(5,5), angle=[-25, 25], direction=[-1, 1], resample='BICUBIC'`	0.7
	Box Blur	`kernel_size=(5,5)`	0.7
	Gaussian Noise	`std=0.05`	1.0
No	Brightness	`range=[0.95, 1.05]`	0.9
	Contrast	`range=[0.95, 1.05]`	0.9
	Saturation	`range=[0.95, 1.05]`	0.9
	Gaussian Blur	`kernel_size=(3,3), sigma=(1.2,1.2)`	0.7
	Motion Blur	`kernel_size=(3,3), angle=[-20, 20], direction=[-1, 1], resample='BICUBIC'`	0.7
	Box Blur	`kernel_size=(3,3)`	0.7
	Gaussian Noise	`std=0.006`	0.9

Table B1. This table lists the different parameters used to manipulate a victim video frame or victim image so that the manipulated area is either perceptually visible or invisible. These parameters are passed into different augmentation modules available in the open-source differentiable computer vision library Kornia.

Note that, in addition to the three video datasets made using the process described above, we also created three Standard Image Manipulation datasets in the exact same procedure. This resulted in three auxilary image datasets: ICMS, IPVM and IPIM.

A summary of all datasets are listed in Table B2, B3, B4

A.2. Set B: In-the-Wild Manipulated Datasets

In addition to evaluating on our Standard Video Manipulation datasets, we tested our proposed network and other

network on six In-the-Wild Manipulated datasets: E2FGVI Inpainted Videos, FuseFormer Inpainted Videos, VideoSham, DeepFaceLab Deepfake Videos, DeepfakeDetectionDataset (DFD), and FaceForensics++(FF+).

The **VideoSham** dataset used in this paper was a subset of the one published by Adobe Research [6]. We excluded videos with audio track or temporal manipulations and videos with resolution less than 1080p. Hence, the remaining manipulated videos were attacked by 1) adding objects/subjects, 2) removing objects/subjects, 3) background/object's color change and 4) adding/removing text. Since the masks indicating the manipulation regions were missing from the original dataset, we created them by first dividing both the manipulated frame and the original frame into small, non-overlapping blocks (16×16), then computing the average luminance-value absolute difference across 3 channels (R, G, B) between each original and manipulated block. After we produced a scalar value for every block, we normalized these values so that they were between 0 and 1 and thresholded them so that they became binarized. We then projected the binary value of each blocks to all the pixel locations belong to that block in order to create a binarized pixel-level masks indicating the manipulated region.

The **E2FGVI and FuseFormer Inpainted Videos** datasets was generated by replacing objects in a scene with their background. In order to leverage existing state-of-the-art video inpainting algorithms like E2FGVI-HQ [4] and FuseFormer [5], we needed videos in which some to-be-removed objects were segmented across multiple frames. Therefore, we chose the Densely Annotated Video Segmentation (DAVIS) dataset [7] for this purpose because it contains both the original videos and the ground-truth segmentation masks for foreground objects in those videos. Inputing a video and its masks into the network code provided publicly by the authors of E2FGVI-HQ and FuseFormer, and using the recommended settings, we generated videos in which the segmented objects were inpainted over. We applied this process over all 90 videos in the DAVIS dataset to get 90 inpainted videos, which we used for evaluation.

The **DeepFaceLab Deepfake Videos** dataset was made by creating deepfaked videos of celebrity interview videos downloaded from YouTube. These downloaded videos were first trimmed down to maximum of 30 seconds, where the only primary subject on the scene was a human body with its face clearly visible. We then chose one source-destination video pair from our set of downloaded videos to perform face swapping. Note that faces of the similar skin tone and gender were more likely to result in better quality deepfakes. We then extracted all faces from the frames of the source and destination video. These faces were then aligned and learned by a deep neural network architecture from DeepFaceLab [2]. After we trained the deepfake network, we used it to perform face-swapping for each uncompressed frame. Finally, these frames were compressed into an H.264 video using FFmpeg with the constant rate factor of 23 and the frame rate of 30 FPS. Using this procedure, we generated 10 deepfaked videos, in which we took out continuous 30 frames chunk from each video to make the Deepfake Video dataset.

The **DeepfakeDetectionDataset** and the **FaceForensics++** dataset were gathered from the original authors' sources available on github [1, 8]. These two datasets contains original videos and videos which were deepfaked using different algorithms (Face2Face [9], FaceSwap [3], etc.).

	Original		Manipulated	
Dataset	# of Frames	# of Videos	# of Frames	# of Videos
VCMS	48000	1600	48000	1600
VPVM	48000	1600	48000	1600
VPIM	48000	1600	48000	1600
ICMS	48000	N/A	48000	N/A
IPVM	48000	N/A	48000	N/A
IPIM	48000	N/A	48000	N/A

Table B2. Summary of our training datasets.

	Original		Manipulated	
Dataset	# of Frames	# of Videos	# of Frames	# of Videos
VCMS	7800	260	7800	260
VPVM	7800	260	7800	260
VPIM	7800	260	7800	260
ICMS	7800	N/A	7800	N/A
IPVM	7800	N/A	7800	N/A
IPIM	7800	N/A	7800	N/A

Table B3. Summary of our validation datasets.

		Original		Manipulated	
Group	**Dataset**	# of Frames	# of Videos	# of Frames	# of Videos
A	VCMS	4200	140	4200	140
	VPVM	4200	140	4200	140
	VPIM	4200	140	4200	140
B	VideoSham	7897	32	12746	64
	E2FGVI Inp. Videos	6208	90	6208	90
	FuseFormer Inp. Videos	6208	90	6208	90
	DeepFaceLab Deepfake Videos	300	10	300	10
	DFD	103056	108	305447	920
	FF++	73770	140	284064	840

Table B4. Summary of the datasets used for evaluating the performance of our network and others.

B. Run Time Analysis

In this section, we provide a preliminary run time analysis of our network and other competing networks: FSG, EXIFnet, Noiseprint, ManTra-Net and MVSS-Net. These run time benchmarks are gathered using unoptimized code, taken directly from other authors' publicly available code repositories. Therefore, these results do not represent the best potential run time of these algorithms. They only show what one may experience the run time of each algorithm by directly using other authors' publicly available code. We performed this analysis on a machine with an NVIDIA RTX

Network	Average Analysis Frame Rate (FPS)
Proposed	**30.80**
FSG	1.82
EXIFnet	0.04
Noiseprint	0.57
ManTra-Net	0.40
MVSS-Net	26.06

Table C1. This table shows the average run time (number of samples per second) of our network and competing networks.

3090 GPU, a 12th Gen Intel i9-12900KF CPU (24 cores at 5.200 Ghz), 64 GB of DDR4 (2800Mhz) RAM, running Ubuntu 22.04.1 with Kernel version 5.15.0-52-generic. We ran 1000 samples individually, each of size $1080 \times 1920 \times 3$, through each network, recorded the total run time, then reported the average run time as frames per second (FPS).

From Table C1, we see that our network achieves the highest FPS, or fastest run time when compared to competing networks. Additionally, with a FPS of over 30, our network is capable of real-time video processing, which can be very useful in practical scenarios.

C. Additional Examples From Each Dataset and Their Localization Results

In this section, we show additional representative examples from each dataset along with their localization results from our proposed network and other competing networks: FSG, EXIFnet, Noiseprint, ManTra-Net and MVSS-Net.

A brief discussion and interpretation of the results for each dataset is provided in each figure's caption. Results for the VCMS dataset are presented in Figure D1, results for the VPVM dataset are presented in Figure D2, results for the VPIM dataset are presented in Figure D3, results for the Deepfake Video dataset are presented in Figure D4, results for the Inpainted Video dataset are presented in Figure D5 and results for the VideoSham dataset are presented in Figure D6.

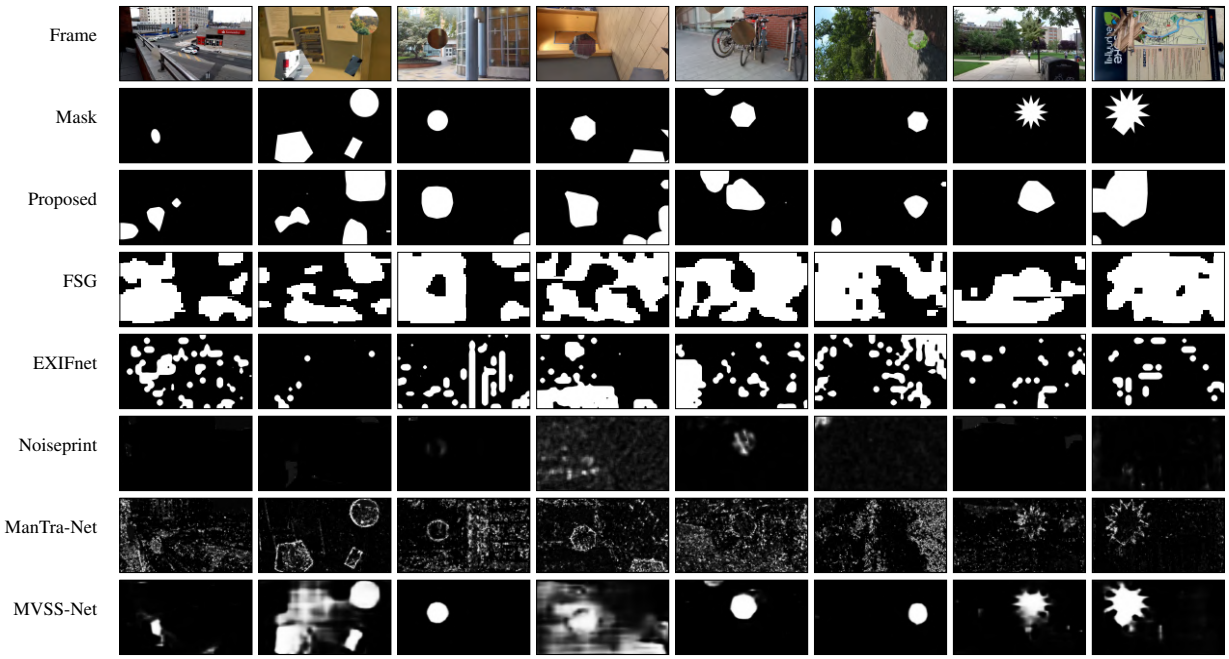

Figure D1. This figure shows the localization results of different networks on the VCMS dataset. Our proposed network's localization results are good, with some minor false alarms on column 1, 3, 6 and 9. We note that, our predicted masks are blobs, which means a large source of our pixel-level localization error comes from the fact that we cannot predict shapes with sharp, concave edges. By contrast, networks, which leverage edge information, like ManTra-Net and MVSS-Net were able to produce masks with sharp edges. Hence, MVSS-Net were comparable to our network in terms of localization performance. On the other hand, FSG, EXIFnet and Noiseprint did not seem like they could make reasonable predictions about the manipulation region.

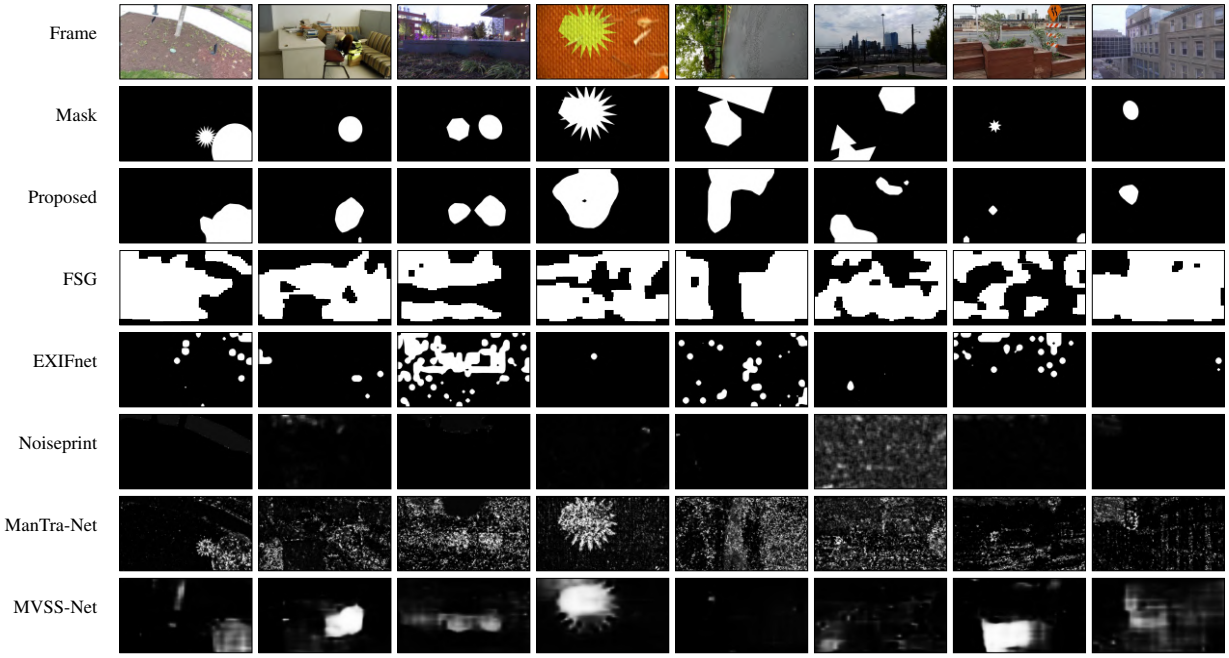

Figure D2. This figure shows the localization results of different networks on the VPVM dataset. Our proposed network's localization results are good, with some minor false alarms on column 2 and 7. Again, our predicted masks are blobs, which means a large source of our error comes from the fact that we cannot predict shapes with sharp, concave edges. Nonetheless, contrast to results on VCMS, strong competitors like MVSS-Net and Mantra-Net could not identify the manipulated region unless it was extremely visible (e.g column 1, 2, 3 and 4). Other examples contained manipulations which resulted in perceptually visible edges but it seemed like competing algorithms false alarmed on regions which had distinct textures versus the rest of the frame, such as they sky (column 3, 6, 8), the colored bricks (column 7), and the building (column 8).

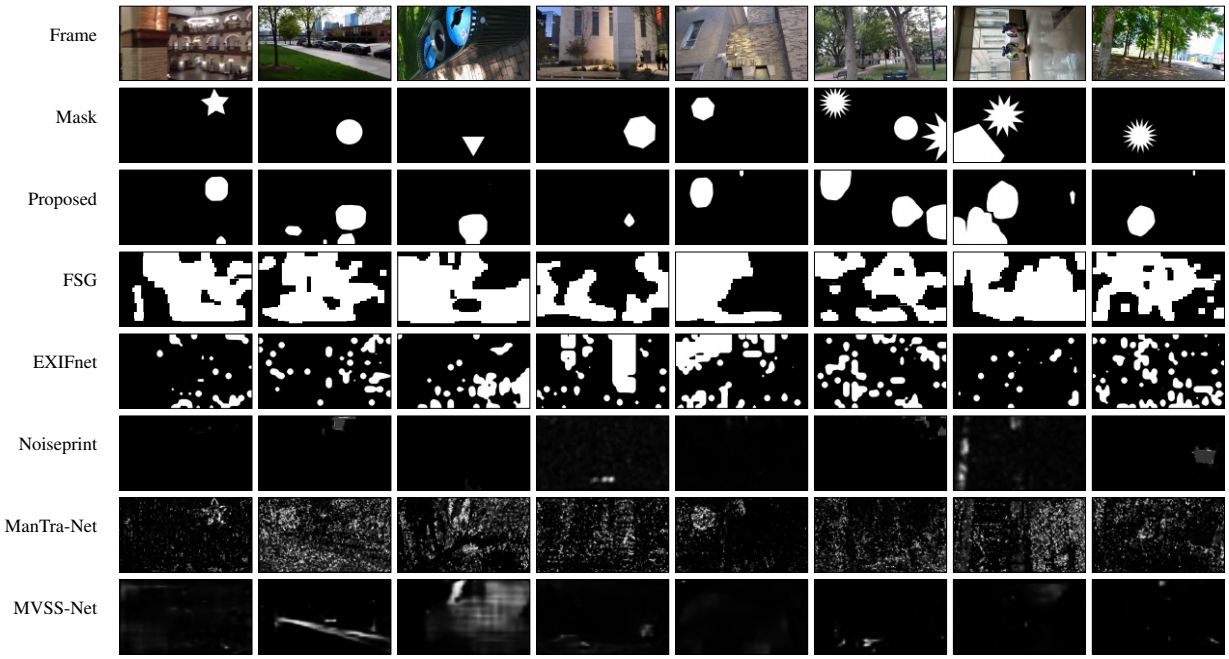

Figure D3. This figure shows the localization results of different networks on the VPIM dataset. Our proposed network's localization results are strong, with some minor false alarms on column 1, 2 and mis-detections on column 4. Differs from VCMS and VPVM, in this dataset, the manipulations' strengths were so low that they are largely perceptually invisible. Hence, in contrast to our network, other competing networks failed to provide any meaningful predictions of the manipulated region.

Figure D4. This figure shows the localization results of different networks on the DeepFaceLab Deepfake Videos dataset. Our proposed network's localization results are reasonable, with minor false alarms on column 1, 2, 3, 7 and mis-detections on column 4. Our network was able to largely identify the deepfaked faces in each of these example. However, for scenes similar to column 4, in which both the manipulated region and its surrounding regions had poor lighting condition, our network's performance seemed to be lower. Notably on column 7, although only the face was manipulated, we were able to detect the watermarked logo on the top right of the frame as well. This prediction is reasonable since the watermark was added in post-processing, hence, it could be considered an additional manipulation. Since other competing networks seemed to be identifying the entire body or the surrounding regions with distinct textures to be manipulated, they did not provide good localization performance in this dataset.

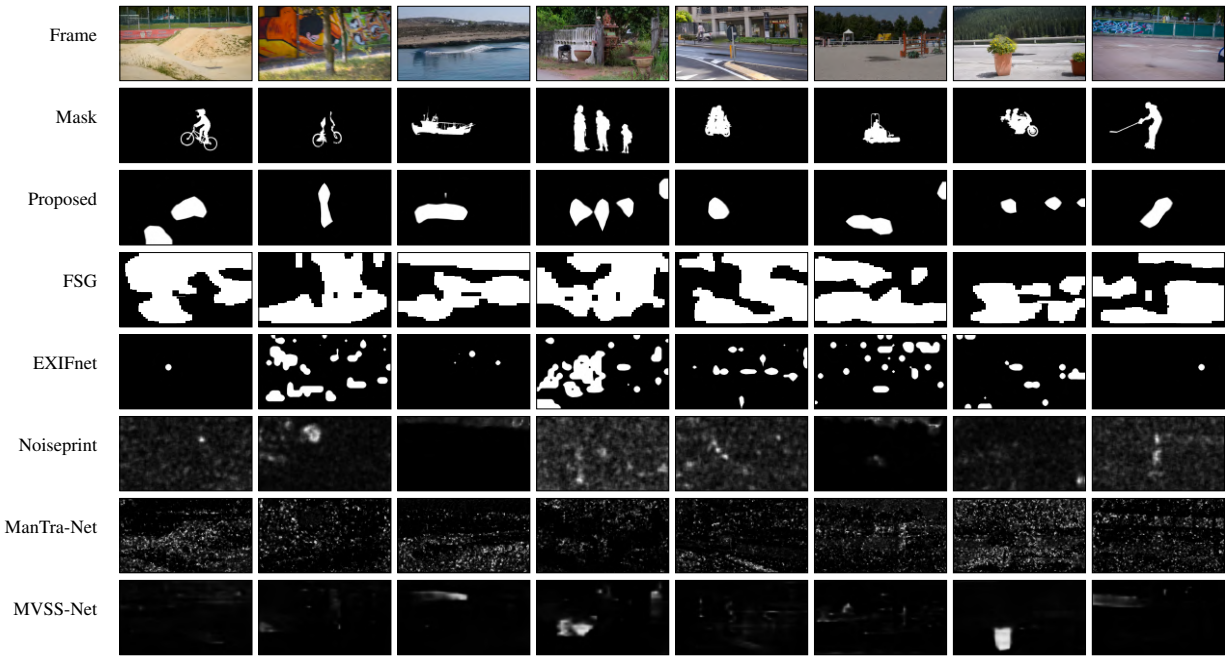

Figure D5. This figure shows the localization results of different networks on the two Inpainted Videos datasets. Our proposed network's localization results are reasonable, with false alarms on column 1, 4, 6, 7 and mis-detections on column 7. Our network were able to largely identify the regions where the segmented objects were removed. On column 7, although our network misdetected the removed motorbike on this frame, we were able to occasionally catch it in other frames of the video. Generally, when the video was stable, without a lot of moving objects, we were able to reliably detect the manipulated regions. However, if the camera moved around too quickly, the frame was blurred, which made our network less likely to provide accurate predictions. While our network generalized well over this dataset, other competing networks failed to identify any manipulations in these videos. We suspected that since objects were removed, there existed no edge information for these networks to rely on, and the residual information might be too different for them to behave properly.

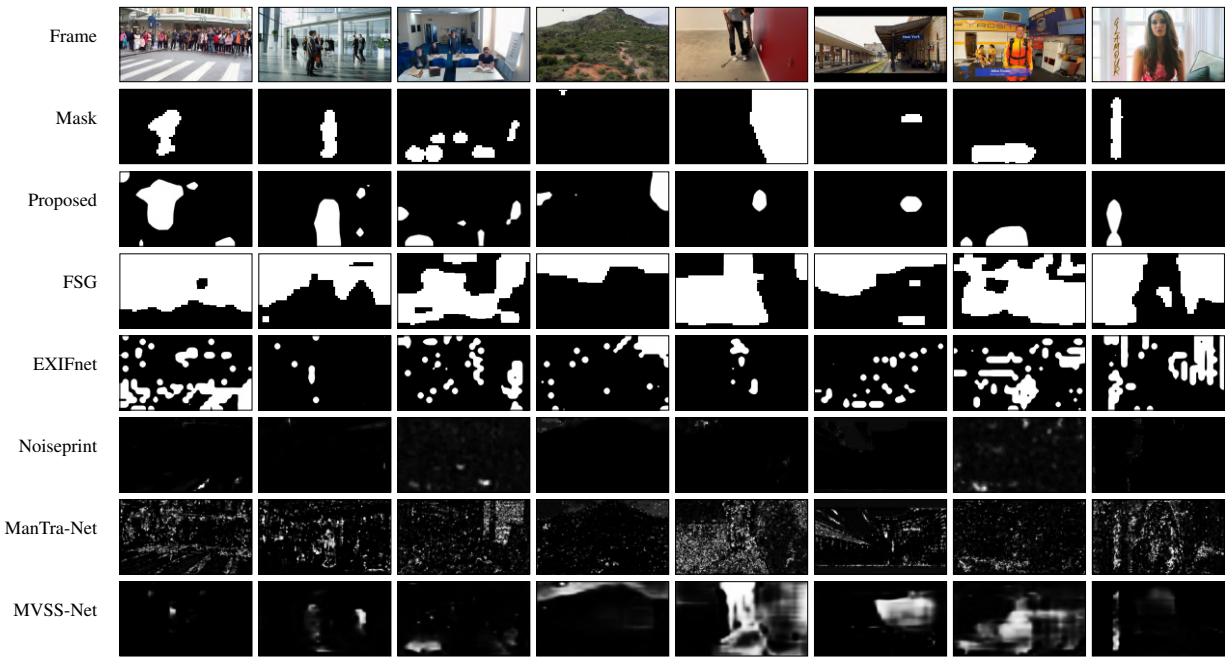

Figure D6. This figure shows the localization results of different networks on the VideoSham dataset. Our proposed network's localization results are reasonable, with minor false alarms on column 1, 2, 3 and mis-detections on column 3, 4, 5. Our network were able to largely identify the manipulated regions in each of these example. However, on scenes like column 4 where the manipulated region was too small, it was not possible for our network and other competing networks to detect. Our network also missed two added books in the back on columns 3, likely because their sizes were small. Additionaly, on column 5, where the color of the wall on the right was changed from blue to red, our network failed to identify the entire manipulated region. Other than these errors, our network produced reasonable localization results, while other competing methods largely failed to identify any manipulations.

References

[1] Nick Dufour and Andrew Gully. Deepfakedetectiondataset - contributing data to deepfake detection research, Sep 2019. 2

[2] Ivan Perov et. al. Deepfacelab: A simple, flexible and extensible face swapping framework. *CoRR*, abs/2005.05535, 2020. 2

[3] M Kowalski. Marekkowalski/faceswap: 3d face swapping implemented in python. *GitHub.[Online]. Available: https://github. com/MarekKowalski/FaceSwap*. 2

[4] Zhen Li, Cheng-Ze Lu, Jianhua Qin, Chun-Le Guo, and Ming-Ming Cheng. Towards an end-to-end framework for flow-guided video inpainting. In *CVPR*, pages 17562–17571, June 2022. 2

[5] Rui Liu, Hanming Deng, Yangyi Huang, Xiaoyu Shi, Lewei Lu, Wenxiu Sun, Xiaogang Wang, Jifeng Dai, and Hongsheng Li. Fuseformer: Fusing fine-grained information in transformers for video inpainting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14040–14049, 2021. 2

[6] Trisha Mittal, Ritwik Sinha, Viswanathan Swaminathan, John Collomosse, and Dinesh Manocha. Video manipulations beyond faces: A dataset with human-machine analysis. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 643–652, 2023. 2

[7] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017. 2

[8] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics++: Learning to detect manipulated facial images. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1–11, 2019. 2

[9] Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2face: Real-time face capture and reenactment of rgb videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2387–2395, 2016. 2