FACULTY OF ENGINEERING AND ARCHITECTURE

Ignoring the Decoy: Tackling Forensic Distractions in Fake Image Detection

Xander Staelens
Student number: 02008303

Supervisors: Prof. dr. Peter Lambert, Prof. dr. ir. Glenn Van Wallendael
Counsellor: Dr. ir. Hannes Mareen

Master's dissertation submitted in order to obtain the degree of Master of Science in Computer Science Engineering

Academic year 2024-2025

GHENT UNIVERSITY

Acknowledgements

Last year, I had to choose a topic for my master's thesis. With the sheer amount of topics available, I was overwhelmed. I had no idea where to start. One topic caught my eye. With the title "Game of Fact Checks: attacking image & video forgery detection methods", I was immediately intrigued. After one meeting with my soon-to-be counsellor Dr. ir. Hannes Mareen, I was sold.

I would like to thank my counsellor Dr. ir. Hannes Mareen for his guidance and support throughout this year. His expertise and insights have been invaluable in shaping my research and gaining a deeper understanding of the field. I am grateful for his encouragement and constructive feedback, which have helped me to create a thesis that I am proud of.

I would also like to thank my fellow student Warre Veys, not only for the many discussions we had, which helped me to clarify my thoughts, but also for all the help throughout my entire university career and all the projects we worked on together.

Lastly, I would like to thank my family and friends for their support and encouragement throughout my studies. In particular, I would like to thank my mom for always supporting me and comforting me in moments of stress. I would like to thank my dad for his encouragement and the many talks where I shared what I had just learned at university. And I would like to thank my sister for always being there for me. I am grateful for these last five years at university. I got to meet so many amazing people. Warre, Hendrik, Arthur and Carole-Anne. Thank you for all the fun times we had together.

I am curious to see what the future holds.

Admission to loan

"The author(s) gives (give) permission to make this master dissertation available for consultation and to copy parts of this master dissertation for personal use. In all cases of other use, the copyright terms have to be respected, in particular with regard to the obligation to state explicitly the source when quoting results from this master dissertation."

Toegang tot bruikleen

"De auteur(s) geeft (geven) de toelating om deze masterproef voor consultatie beschikbaar te stellen en delen van de masterproef te kopiëren voor persoonlijk gebruik. Elk ander gebruik valt onder de bepalingen van het auteursrecht, in het bijzonder met betrekking tot de verplichting de bron uitdrukkelijk te vermelden bij het aanhalen van resultaten uit dit afstudeerwerk."

Xander Staelens, May 2025

Note related to the master thesis and oral presentation

"This master's dissertation is part of an exam. Any comments formulated by the assessment committee during the oral presentation of the master's dissertation are not included in this text."

Toelichting in verband met het masterproefwerk en de mondelinge uiteenzetting

"Deze masterproef vormt een onderdeel van een examen. Eventuele opmerkingen die door de beoordelingscommissie tijdens de mondelinge uiteenzetting van de masterproef werden geformuleerd, werden niet verwerkt in deze tekst."

Ignoring the Decoy: Tackling Forensic Distractions in Fake Image Detection

by
Xander Staelens

Master's dissertation submitted in order to obtain the academic degree of Master of Science in Computer Science Engineering

Supervisor: Prof. dr. Peter Lambert, Prof. dr. ir. Glenn Van Wallendael
Counsellor: Dr. ir. Hannes Mareen

Faculty of Engineering and Architecture
Ghent University

Abstract - Creation of fake images has become easier than ever thanks to the growing sophistication and accessibility of image manipulation tools and generative AI. This has led to a surge in fake images being shared online, increasing the need for reliable and robust multimedia forensics methods. Forgery Localization (FL) models aim to detect and localize image tampering, while Synthetic Detection (SD) models aim to identify whether an image is AI-generated or not. However, the robustness of some methods is compromised by forensic distractions that occur in real-world scenarios: benign visual elements such as captions, logos, and watermarks. This thesis investigates the impact of distractions on state-of-the-art FL and SD models, which was never formally done before. We reveal that while some FL models are robust against distractions, others are highly sensitive to them. With regards to SD models, we find that they are relatively robust against distractions. To improve the robustness of FL models against distractions, I propose two methods, both of which use a user-specified distraction cover to indicate distraction regions. First, the filter adapter, a lightweight neural network trained to filter out distraction artifacts from latent feature maps. Second, the masked convolution, a replacement for the standard convolution that ignores distraction regions during inference. The effectiveness of both approaches is evaluated by integrating them into TruFor, a state-of-the-art FL model that is highly sensitive to distractions. The masked-convolution approach significantly improves TruFor's robustness against distractions, nearly restoring original performance, while the filter approach only provides limited improvements. The proposed solutions, and in particular the masked-convolution approach, demonstrate their potential as distraction-aware techniques to enhance the real-world applicability of FL models.

Keywords - Image Forensics, Image Forgery Localization, Synthetic Image Detection, In-the-wild Robustness

Ignoring the Decoy: Tackling Forensic Distractions in Fake Image Detection

Xander Staelens

***Abstract*—Creation of fake images has become easier than ever thanks to the growing sophistication and accessibility of image manipulation tools. This has led to a surge in fake images being shared online, increasing the need for reliable and robust multimedia forensics methods. Recent forgery localization (FL) models (like TruFor) that detect and localize image tampering are compromised by forensic distractions. These distractions are benign visual elements such as captions, logos, and watermarks, that occur in real-world scenarios. We investigate the impact of distractions on state-of-the-art FL models. which was never formally done before. We reveal that while some models are robust against distractions, others are highly sensitive to them, leading to large performance drops. To address this issue, we propose a novel masked-convolution approach for CNN-based FL models, enabling them to ignore user-specified distraction regions during inference. We evaluate the effectiveness of this approach by integrating it into TruFor. The masked-convolution approach significantly improves TruFor's robustness against distractions, nearly restoring original performance. This demonstrates its potential as a distraction-aware technique to enhance the real-world applicability of FL models.**

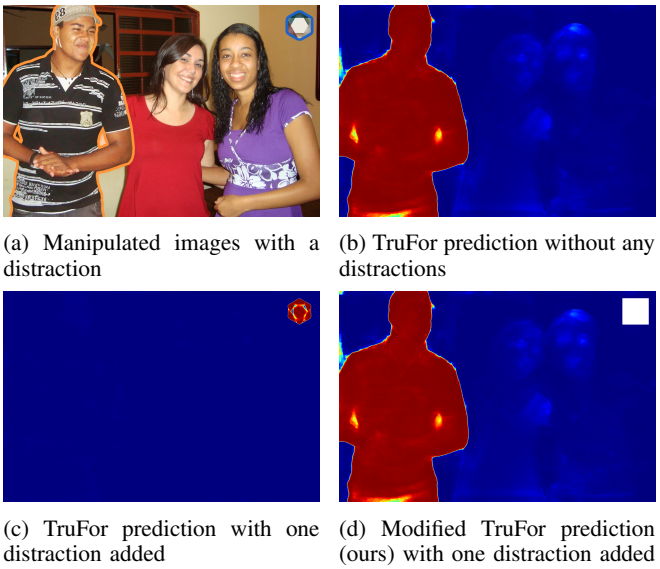
***Index Terms*—Image forensics, forgery localization, in-the-wild robustness**

I. Introduction

In recent years, fake news has emerged as a significant global issue. People are highly susceptible to misinformation, which can shape their opinions, influence their decisions, and ultimately impact large-scale societal domains such as politics, marketing, and public health. Social media platforms have become the primary battlegrounds for the spread of misinformation, false narratives, conspiracy theories, and manipulated multimedia content. The consequences of misinformation can be severe, swaying public opinion, deepening societal divisions, and undermining trust in institutions.

Due to the growing sophistication and accessibility of image manipulation tools, creating high-quality fake images is no longer limited to skilled professionals [1]. These technologies empower creativity and innovation, allowing artists and the entertainment industry to push creative boundaries. However, the same tools have the potential to be exploited for malicious purposes, such as fabricating fake news, spreading misinformation, impersonating individuals, and facilitating fraud.

Given these developments, the need for robust and reliable multimedia forensics tools has become increasingly important. Forensic tools can be divided into two main categories. Forgery Localization (FL) focuses on identifying and localizing manipulated regions in images, while Synthetic Image Detection (SID) focuses on detecting whether an image is real or synthetic. This research focuses on FL techniques.



(a) Manipulated images with a distraction

(b) TruFor prediction without any distractions

(c) TruFor prediction with one distraction added

(d) Modified TruFor prediction (ours) with one distraction added

Fig. 1: Before adding the forensic distraction (in blue), TruFor [2] detects the forgery (in orange) with high accuracy (b). However, after adding the distraction and performing lossy JPEG compression, TruFor fails to detect the forgery (c). The distraction disturbs the model, causing it to lose focus on the forgery. TruFor adapted with the proposed masked-convolution approach is able to ignore the distraction and regain focus on the forgery (d). The white box indicates the masked region. Note that the same compression was also applied to the non-distracted image (b) before the TruFor prediction was made.

Despite the significant progress in the field of multimedia forensics, current state-of-the-art models still face several critical challenges, limiting their effectiveness in real-world applications. One core issue is the limited robustness against post-processing operations, such as compression and resizing.

Another common post-processing operation is the addition of benign visual elements, such as captions, watermarks, and logos. These superficial, non-malicious modifications, referred to as *forensic distractions*, can significantly impact the performance of FL models. Sometimes, resulting in a full loss of focus, causing models to entirely ignore present forgeries. Fig. 1 shows an example of the impact of forensic distractions on the performance of TruFor [2], a state-of-the-art FL model. Even though forensic distractions are a common occurrence in real-world images, their effect has never been formally studied.

The aim of this research is to provide answers to the

following two research questions:

- **RQ1:** What is the impact of forensic distractions on existing FL models?
- **RQ2:** To what extent can models that are sensitive to distractions be made more robust?

This paper's contributions are: (i) Quantifying the impact of forensic distractions on the performance of FL methods. (ii) Proposing the novel masked-convolution approach for distraction-aware CNN-based FL, which replaces all standard convolutions with masked convolutions to explicitly ignore user-defined distraction regions. (iii) Evaluating the effectiveness of the proposed approach by integrating it into TruFor and demonstrating its potential to significantly improve robustness against distractions.

II. Related work

Manipulated images consist of both authentic and forged regions. FL models aim to identify these forged regions, and produce a pixel-level anomaly map, where each pixel is assigned a score indicating the likelihood of being manipulated. This is done by analyzing subtle forensic traces, such as camera noise, compression artifacts, and deepfake traces, introduced by the manipulation process.

The models can be divided into two main categories: model-based methods and data-driven deep learning methods. Model-based methods rely on handcrafted features and explicit assumptions about the statistical or physical properties of authentic images [3]–[6]. Their performance often depends heavily on the validity of their underlying assumptions, leading to limited generalization in real-world scenarios.

Data-driven methods, on the other hand, learn to detect and localize forgeries directly from data, by training on large datasets of authentic and manipulated images [2], [7], [8]. These models typically rely on Convolutional Neural Networks (CNNs) and Vision Transformers to capture both low-level forensic traces and high-level semantic inconsistencies. Deep learning methods show superior performance compared to model-based methods. However, they are black-box models, making it difficult to understand the reasoning behind their predictions. TruFor [2], the method used in Fig. 1, is an example of a deep learning method.

A main challenge of FL is providing robustness against post-processing operations, such as compression, resizing, and forensic distractions (the latter elaborated in Section III-A).

III. Motivation: Forensic Distractions

In this section, an experiment is performed to evaluate the impact of forensic distractions on state-of-the-art FL models. The goal is to quantify the decrease in detection performance due to the addition of distractions, thus answering RQ1.

A. Forensic distractions

As images circulate across online platforms, they are often intentionally modified (e.g., logos, watermarks and captions) or unintentionally modified (e.g., notification bar when taking screenshots) with additional visual elements. These modifications, referred to as forensic distractions, do not semantically alter the image or introduce falsified information. However, in the eyes of FL models, they are still considered manipulations, and can be mistakenly prioritized over the actual forgery.

The forensic models analyzed in this research do not immediately fail when a distraction is added. Instead, they often simply detect both the distraction and the forgery. However, if lossy compression is applied, the distraction artifacts are spread throughout the image, hiding the forensic traces of the forgery, resulting in a loss of focus as illustrated in Fig. 1.

B. Experimental setup

Forensic methods Seven FL methods are evaluated. Four model-based methods (DCT [3], ADQ1 [4], BLK [5], and CAGI [6]), and three data-driven methods (TruFor [2], CAT-Net [7], and PSCC-Net [8]).

Datasets Four datasets are used to evaluate the models: DSO-1 [9], VIPP [10], OpenForensics [11], and IMD2020 [12]. The DSO-1 and VIPP datasets are used in full since they contain a limited number of images — 100 and 62, respectively. From the OpenForensics and IMD2020 datasets, 200 images are randomly selected. All images are resized to 1536x1024 pixels.

Distractions For each image, a position (upper-left, upper-right, lower-left, lower-right) and a logo are randomly selected. Ten logos are used, sourced from www.freepik.com. Logos have a size of 128x128 pixels, while the images themselves are 1536x1024 pixels. The logos are placed in the selected corner, after which the distracted image is JPEG compressed with Q95. The lossy compression is part of the source of the problem, therefore it needs to be included in the evaluation pipeline. The addition of compression does not reduce the generalizability of the results, as lossy compression is a common step when sharing images online.

Detection score The goal of the evaluation is to determine how well the models continue to detect forgeries when distractions are present. Thus, the model should neither be rewarded nor punished for detecting the distraction. To fairly compare the model predictions (P) to the ground truth (G), the *adapted* F1 score (1) is used, which only considers the pixels that are not part of the distraction. For this calculation, a binary distraction mask (M) is created, indicating the pixels that belong to the distraction.

$$\text{F1* score(P, G, M)} = \frac{2\text{TP[M=0]}}{2\text{TP[M=0]} + \text{FP[M=0]} + \text{FN[M=0]}} \quad (1)$$

C. Results

Table I shows the mean F1* scores of the evaluated models on the original and distracted images, and the corresponding decrease, for each dataset.

Model-based methods are highly robust against the forensic distractions, showing only a small decrease in performance of 0.84%. Note that this may be because these methods

TABLE I: Performance of FL models before (orig) and after (distr) adding forensic distractions and the corresponding decrease (decr), measured in F1* score. Most notably, TruFor has a significant decrease, addressed by the proposed Mod. TruFor.

	Model	DSO-1			VIPP			OpenForensics			IMD2020			Average		
		orig	distr	decr	orig	distr	decr	orig	distr	decr	orig	distr	decr	orig	distr	%decr
Model-based	DCT [3]	.231	.231	.000	.189	.187	.002	.112	.111	.001	.147	.143	.004	.169	.168	0.83%
	ADQ1 [4]	.225	.224	.001	.179	.179	.000	.099	.098	.000	.142	.138	.003	.161	.160	0.80%
	BLK [5]	.186	.185	.000	.144	.142	.002	.076	.075	.001	.122	.120	.002	.132	.131	0.87%
	CAGI [6]	.195	.190	.005	.187	.189	-.002	.065	.065	.000	.120	.118	.002	.142	.141	0.84%
Deep learning	PSCC-Net [8]	.138	.087	.051	.165	.139	.026	.112	.082	.029	.074	.060	.014	.122	.092	24.67%
	CAT-Net [7]	.067	.059	.008	.126	.116	.011	.284	.275	.009	.529[1]	.514[1]	.015	.251	.241	4.26%
	TruFor [2]	.435	.161	.274	.330	.028	.302	.548	.222	.326	.558[1]	.426[1]	.133	.468	.209	55.28%
	Mod. TruFor (ours)	.435	.384	.051	.330	.289	.041	.548	.530	.018	.558[1]	.535[1]	.024	.468	.434	7.12%

[1]TruFor and CAT-Net were trained on the IMD2020 dataset, which explains their higher performance and robustness to distractions on this dataset.

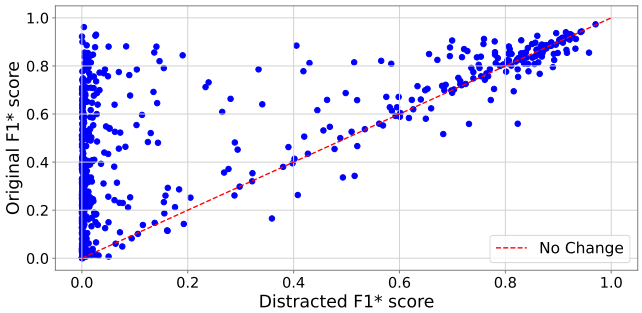


Fig. 2: Scatter plot of the localization scores (F1* scores) of **TruFor**. As points move farther to the left (away from the diagonal), the model's performance increasingly degrades due to the presence of the forensic distraction.

perform poorly in general, with an average F1* score between 0.13 and 0.17.

TruFor displays a significant sensitivity to the distractions, with an average decrease in performance of 55.28%. On the non-distracted images, TruFor performs the best of all evaluated models, across all datasets, with an average F1* score of 0.47.

Fig. 2 shows a scatter plot of TruFor's performance across all datasets. The y-axis and x-axis, respectively, show the F1* score on the images before and after distractions are added. The farther a point moves to the left, the greater the impact of distractions on its detection. A large number of points are located at the left side, indicating a large decrease in performance. However, a cluster of points is also visible in the upper right corner. These are images in which the forgery is easy te detect without distractions and remains easy to detect even with distractions.

Fig. 3 shows three examples of manipulated images with their corresponding TruFor predictions with and without distractions. It can be seen that not every image is equally affected by the distractions. The first and second image show a large decrease in performance, while the third image shows only a small decrease.

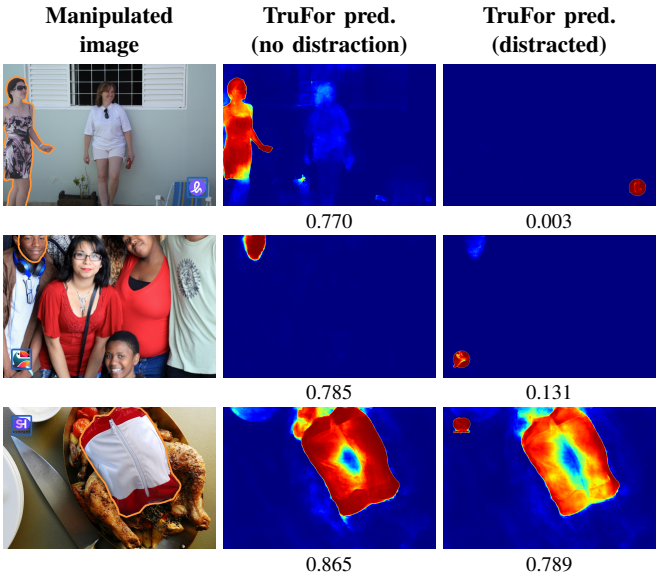


Fig. 3: Examples of manipulated images (manipulation in orange and distraction in blue) with corresponding TruFor [2] predictions and F1* scores, with and without distractions. It can be seen that the impact of distractions on a models performance varies across images.

PSCC-Net scores poorly on all datasets, performing even worse than the model-based methods. This limited performance is further increased by the addition of distractions, leading to an average decrease of 24.67%, showing moderate sensitive to the presence of distractions.

CAT-Net displays a high resilience to forensic distractions, with an average decrease of only 4.26%. This is still a non-negligible decrease, but it is significantly lower than TruFor and PSCC-Net. However, CAT-Net does perform quite a bit worse than TruFor on non-distracted images, with an average F1* score of 0.24 compared to 0.47 for TruFor. This implies that CAT-Net cannot simply be used as distraction-robust replacement for TruFor.

These results highlight the need for distraction-aware approaches to improve the robustness of FL models. As TruFor is the best performing model on non-distracted images, and the most sensitive to distractions, we aim to improve its robustness in Section IV.

IV. Proposed method: Masked Convolutions

This section describes our proposed method of masked convolutions, allowing for distraction-aware forgery localization. This approach was built with the TruFor model in mind, but the concepts can be applied to any CNN-based model.

A. Masking components

Distractions introduce artifacts into the image, which can interfere with forgery detection. To prevent these artifacts from spreading, it is necessary to mask every component in the model that could propagate them. This includes any component that combines information from multiple pixels. Masking a component involves modifying its operation so that it ignores pixels specified by a provided mask, effectively treating the masked pixels as if they do not exist.

The most important building block of a CNN is the convolutional layer, used to extract features from the input image. This component combines information from multiple pixels, meaning it needs to be masked. The same holds for the pooling layer, used to reduce the dimensionality of the feature maps.

The proposed masked components require an additional input in comparison to their unmasked counterparts, namely a mask that specifies the region to be ignored. For simplicity, in this work, the mask is defined as a rectangle, characterized by its top-left corner and its width and height. By ignoring the masked pixels, the model stops the spreading of the distraction artifacts and thus regains focus on the forgery.

As a reminder, a convolution layer slides a kernel over the input image, at each position calculating the dot product between the kernel and the pixels it overlaps with. The output is a feature map, which is used as input for subsequent layers.

The provided mask (a rectangle) represents a hole in the image. All pixels inside the hole are considered *invalid* and should be ignored. This is achieved in three steps, as illustrated in Fig. 4. First, the same amount of padding as added on the outside of the image, is added to the inside of the hole. Adding the padding to the hole sets some invalid pixels to 0. The others remain invalid (pad). Next, the default convolution operation is performed (convolve). Lastly, any convolution window that overlaps with non-padded invalid pixels is itself invalid (i.e., its output should be ignored in subsequent layers). The mask is updated to reflect the new validities (update).

Masking any other component is conceptually the same as masking a convolutional layer. For example, masking the pooling layer is done by applying the pooling operation as normal, but only considering the unmasked values.

B. Integration into CNN-based models

Integrating the proposed masked components into any CNN-based model takes three steps. First, all components that mix information from different pixels, need to be replaced with their masked counterparts. Next, a mask must be created from the distraction cover, and passed through the network together with the original input. Third, every time a masked component is called, the mask must be passed to it and the updated mask must be captured.

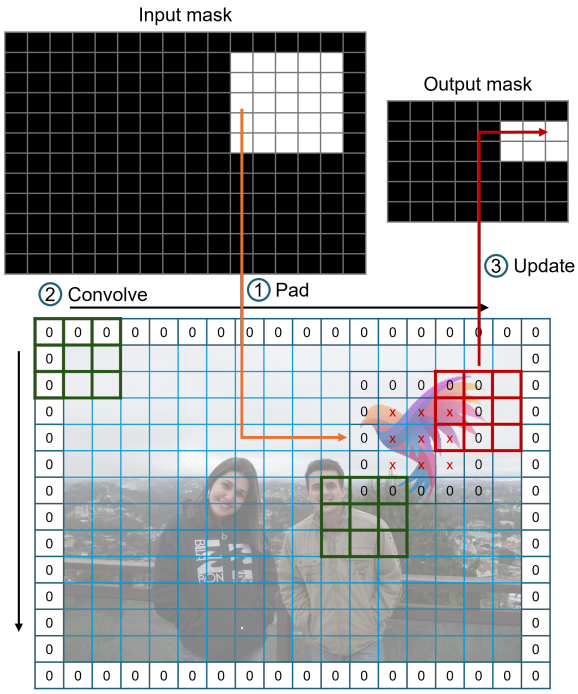


Fig. 4: Illustration of the three steps taken by the proposed masked convolution layer. The illustration shows a 12x16 image with a 3x3 convolutional kernel, a stride of 2 and a padding of 1. Three example convolution windows are shown, two of which are valid (green) and one which is invalid (red).

Many models, such as TruFor, are implemented using the PyTorch [13] framework. To ease the integration process, masked components can be implemented as wrappers of the originals, allowing PyTorch to reuse the original weights.

C. Discussion

The use of masked convolutions as a technique for distraction-aware FL has several advantages. First, it allows the model to focus on the relevant parts of an image while ignoring distractions, and it is toggleable, meaning if no mask is provided, the model behaves as if it were unmodified. Next, it does not require any training. Pretrained weights can be reused. Last, masked convolutions can in theory be added to any CNN-based model, making it a versatile solution. This combination of properties makes masked convolutions a promising approach for improving the robustness of FL models against forensic distractions.

The idea of masked convolutions is similar to the work proposed by Liu et al. (2018) [14]. They used a similarly modified convolution operation for the task of inpainting with

irregular-shaped holes. As opposed to our proposed method of masked convolutions, Liu et al. treat holes in the image differently from pixels outside the image. The convolution operation is performed as normal, but the masked pixels set to 0 and the output rescaled to account for ignored pixels.

In summary, we propose a novel approach to make CNN-based FL models distraction-aware by replacing all components that combine information from different pixels with a masked version.

V. Evaluation

To assess the performance of the proposed masked-convolution approach, we integrated the masked components into the TruFor model and evaluated its performance using the same experiment performed in Section III.

A. Experimental setup

Datasets & Distractions To evaluate the performance of the proposed method, the same datasets and distractions are used as used in Section III-B.

Distraction cover The distraction cover is a binary mask that indicates which pixels should be ignored during the convolution operation. For these tests, the cover is created by selecting the smallest rectangle that contains the distraction padded by 4 pixels on all sides. This cover is given as input to the modified TruFor model, along with the original image, and is used in the calculation of the detection score.

Detection score To quantify the performance of the modified TruFor model, we use the adapted F1 score (1) as defined in Section III-B, with the distraction cover used as the binary mask (M). The adapted F1 score is used for two reasons. First, just as before, the model should neither be rewarded nor punished for detecting the distraction. Second, the modified model generates nonsensical predictions for the masked pixels, which would skew the results if taken into account.

B. Results

Table II shows the mean adapted F1 scores of the modified TruFor model on the distracted images, and the corresponding relative improvement (RI), defined in equation (2), for each dataset.

The relative improvement is used to evaluate the increase in performance of a modified model compared to the original, unmodified, model.

$$\mathrm{RI} = \frac{\text{F1* score [M D]} - \text{F1* score [notM D]}}{\text{F1* score [notM notD]} - \text{F1* score [notM D]}} \qquad (2)$$

with M and D indicating modified and distracted, respectively. The relative percentage increase is a value between 0% and 100%, with 0% indicating no improvement and 100% indicating that the modified model performs as well as the unmodified model on non-distracted images.

The masked-convolution approach is able to significantly improve the performance of the TruFor model in the presence of distractions, with a relative improvement of 87.11%, i.e., nearly matching the original performance on non-distracted images.

TABLE II: Performance of unmodified TruFor vs. TruFor adapted with the proposed masked-convolution approach in the presence of distractions, measured in F1* score. Both models perform equally on undistracted images (orig).

Dataset	Unmodified TruFor [2]			Masked Convolution		
	orig	distr	dec	distr	dec	RI (2)
DSO-1 [9]	.435	.161	.274	.384	.051	81.48%
VIPP [10]	.330	.028	.302	.289	.041	86.34%
OpenForensics [11]	.548	.222	.326	.530	.018	94.54%
IMD2020 [12]	.558	.426	.133	.535	.024	82.22%
Average	.468	.209	.259	.434	.033	87.11%

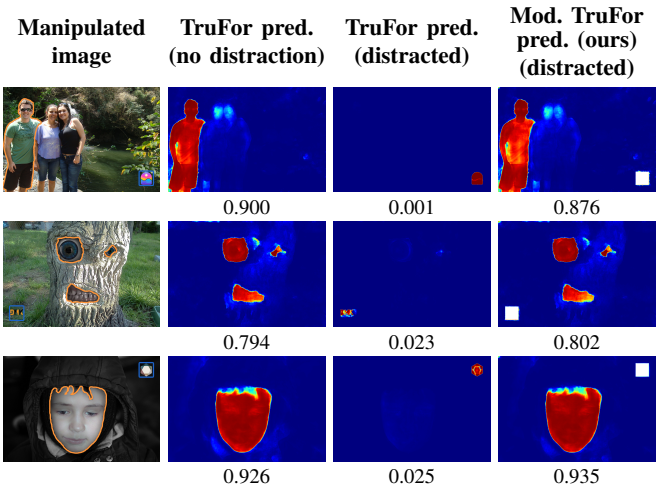


Fig. 5: Examples of manipulated images (manipulation in orange and distraction in blue) with corresponding TruFor [2] and proposed modified TruFor predictions and F1* scores, with and without distractions. The white boxes in the last column indicate the masked regions. The modified model performs significantly better than the original TruFor model in the presence of distractions, showing the effectiveness of the masked-convolution approach as a solution to the problem of forensic distractions.

Fig. 5 shows several distracted images with their corresponding predictions of both the unmodified and modified TruFor model. The approach is able to largely mitigate the impact of forensic distractions on TruFor's performance, allowing it to regain a substantial amount of focus.

The increase in robustness is most apparent when looking at the decrease in performance of the modified model due to addition of distractions. The unmodified TruFor model has an average decrease of 55.28% when distractions are added, while the masked-convolution approach lowers this down to 7.12%.

The increase in robustness can also be seen in Fig. 6, which shows a scatter plot of the F1* scores of the modified TruFor model aggregated over all four datasets. The points lie much

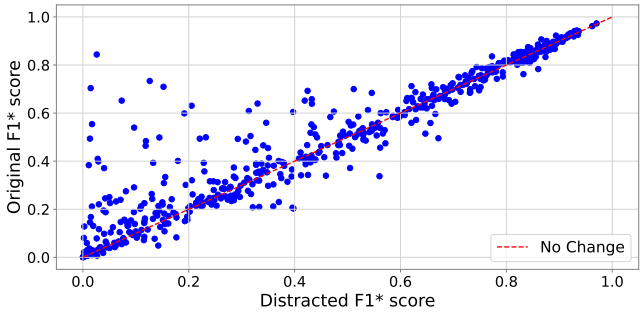


Fig. 6: Scatter plot of the localization scores (F1* scores) of the proposed **modified TruFor** model. Compared to Fig. 2, the points are now more clustered around the diagonal, indicating that the model's performance is significantly less affected by the presence of forensic distractions.

closer to the diagonal compared to the unmodified TruFor model in Fig. 2. This indicates higher resilience to distractions.

When comparing to the other FL models discussed in Section III, the TruFor model with masked convolutions outperforms all other models in the presence of distractions.

Even though the approach seems able to recover most of the original performance, there remains an average decrease in performance of 7.12% when adding distractions. This indicates that the masked-convolution approach is not a complete solution to the problem of forensic distractions, but it is a major step in the right direction. Other approaches should be explored to further improve the robustness of FL models against forensic distractions.

VI. Conclusion

This paper studied the impact of forensic distractions on FL models and proposed a solution to improve their robustness against distractions.

The performed experiments show that forensic distractions can significantly impact the performance of FL models. In particular, TruFor is found to be highly sensitive to distractions, causing a decrease in performance of 55% on average. PSCC-Net is also affected, but to a lesser extent, with an average decrease of 25%. However, not all FL models are equally affected. CAT-Net is found to be quite robust against distractions, and the model-based methods barely show any sensitivity.

To improve the robustness of FL models against distractions, we proposed the use of masked convolutions, which differ from standard convolutions by not convolving over the entire input, but instead only over the parts that are not covered by a provided distraction cover. By replacing all standard convolutions with masked convolutions, the influence of the covered regions on the detection result should be eliminated, meaning that the model can focus on the rest of the image.

The performed evaluation shows that the masked-convolution approach significantly improves the performance of TruFor in the presence of distractions, with a relative improvement of 87.11%. This approach is able to outperform all tested FL models and shows high resilience to distractions, with an average performance decrease of only 7.12% when adding distractions. These results highlight the effectiveness of the masked-convolutions approach as a method for creating distraction-aware FL models.

Future work could focus on improving the proposed method by supporting arbitrarily-shaped masks and automating distraction cover generation, removing the need to manually select distractions. The concept of distractions could also be extended to include other types of regions that distract FL models. For example, this could be useful for images with overexposed regions or multiple manipulations where one draws attention away from the other.

References

[1] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 10 684–10 695.

[2] F. Guillaro, D. Cozzolino, A. Sud, N. Dufour, and L. Verdoliva, "TruFor: Leveraging all-round clues for trustworthy image forgery detection and localization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 20 606–20 615.

[3] S. Ye, Q. Sun, and E.-C. Chang, "Detecting digital image forgeries by measuring inconsistencies of blocking artifact," in *2007 IEEE International Conference on Multimedia and Expo*, 2007, pp. 12–15.

[4] Z. Lin, J. He, X. Tang, and C.-K. Tang, "Fast, automatic and fine-grained tampered jpeg image detection via dct coefficient analysis," *Pattern Recognition*, vol. 42, no. 11, pp. 2492–2501, 2009.

[5] W. Li, Y. Yuan, and N. Yu, "Passive detection of doctored jpeg image via block artifact grid extraction," *Signal Processing*, vol. 89, no. 9, pp. 1821–1829, 2009.

[6] C. Iakovidou, M. Zampoglou, S. Papadopoulos, and Y. Kompatsiaris, "Content-aware detection of jpeg grid inconsistencies for intuitive image forensics," *Journal of Visual Communication and Image Representation*, vol. 54, pp. 155–170, 2018.

[7] M.-J. Kwon, S.-H. Nam, I.-J. Yu, H.-K. Lee, and C. Kim, "Learning jpeg compression artifacts for image manipulation detection and localization," *International Journal of Computer Vision*, vol. 130, no. 8, p. 1875–1895, May 2022.

[8] X. Liu, Y. Liu, J. Chen, and X. Liu, "PSCC-Net: Progressive spatio-channel correlation network for image manipulation detection and localization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 11, pp. 7505–7517, 2022.

[9] T. J. de Carvalho, C. Riess, E. Angelopoulou, H. Pedrini, and A. de Rezende Rocha, "Exposing digital image forgeries by illumination color classification," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 7, pp. 1182–1194, 2013.

[10] M. Fontani, T. Bianchi, A. De Rosa, A. Piva, and M. Barni, "A framework for decision fusion in image forensics based on dempster–shafer theory of evidence," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 4, pp. 593–607, 2013.

[11] T.-N. Le, H. H. Nguyen, J. Yamagishi, and I. Echizen, "OpenForensics: Large-scale challenging dataset for multi-face forgery detection and segmentation in-the-wild," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 10 097–10 107.

[12] A. Novozámský, B. Mahdian, and S. Saic, "IMD2020: A large-scale annotated dataset tailored for detecting manipulated images," in *2020 IEEE Winter Applications of Computer Vision Workshops (WACVW)*, 2020, pp. 71–80.

[13] A. Paszke *et al.*, *PyTorch: an imperative style, high-performance deep learning library*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[14] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image inpainting for irregular holes using partial convolutions," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Table of Contents

Acknowledgements ii

Admission to loan iii

Note related to the master thesis and oral presentation iv

Abstract v

Extended Abstract vi

Table of Contents xii

List of Figures xiv

List of Tables xv

List of Acronyms xvii

1 Introduction 1
- 1.1 Problem Statement . . . 3
- 1.2 Research Questions . . . 4
- 1.3 Thesis Outline . . . 5
- 1.4 Use of Generative AI . . . 6

2 Related work 7
- 2.1 Forgery Localization . . . 8
  - 2.1.1 Forensic Traces . . . 8
  - 2.1.2 Model-Based Methods . . . 10
  - 2.1.3 Data-Driven Deep Learning Methods . . . 11
- 2.2 Synthetic Image Detection . . . 18
  - 2.2.1 Spectrum-Based Methods . . . 18
  - 2.2.2 Local Artifact and Texture-Based Methods . . . 19
  - 2.2.3 Hybrid Feature-Based Methods . . . 19
  - 2.2.4 Foundation Model-Based Methods . . . 20

3 Motivation: The Impact of Forensic Distractions 22
- 3.1 What are Forensic Distractions . . . 23
- 3.2 The Impact on Forgery Localization . . . 25
  - 3.2.1 Experimental setup . . . 25
  - 3.2.2 Results . . . 27
  - 3.2.3 Effectiveness of Distraction Types . . . 33
- 3.3 The Impact on Synthetic Detection . . . 35
  - 3.3.1 Experimental setup . . . 35
  - 3.3.2 Results . . . 36
- 3.4 Conclusion . . . 39

4 Distraction-aware Forgery Localization 41
- 4.1 Possible Solutions . . . 41
  - 4.1.1 Fine-tuning . . . 41
  - 4.1.2 Post-Processing . . . 42
  - 4.1.3 Extra Input and Retraining . . . 43
  - 4.1.4 Filter Adapter . . . 45

4.1.5 Masked Convolutions . . . 46
4.1.6 Overview & Conclusion . . . 48
4.2 TruFor Technical Details . . . 49
4.3 Filter Adapter . . . 50
4.3.1 Architecture . . . 50
4.3.2 Training . . . 51
4.4 Masked Convolutions . . . 53
4.4.1 Masking Components . . . 53
4.4.2 Integration with TruFor . . . 55
4.4.3 Similar Approach used for Inpainting . . . 56
4.5 Conclusion . . . 57

5 Evaluation **58**
5.1 Experiment Setup . . . 59
5.2 Filter Adapter . . . 60
5.3 Masked Convolutions . . . 63
5.3.1 Ablation Study . . . 63
5.4 Conclusion . . . 67

6 Future Work **69**
6.1 Improving the Proposed Solutions . . . 69
6.2 Extending the Scope . . . 70

7 Societal Reflection **71**

8 Conclusion **73**

Bibliography **75**

A Supplementary Study: Effectiveness of Distraction Types **79**
A.1 Textual Distractions . . . 79
A.2 Graphical Distractions . . . 81

B Forensic Distractions on FL Experiment results **83**

C Forensic Distractions on SD Experiment results **95**

List of Figures

1.1 Example of Adobe Photoshop Generative Fill . . . 1
1.2 Example of fake news . . . 2
1.3 Example of insurance fraud . . . 2
1.4 Illustration of the problem of forensic distractions . . . 4

2.1 Illustration showing the difference between FL and SID . . . 7
2.2 Example of PRNUs and Noiseprints of two different cameras . . . 8
2.3 Examples of common visible compression artifacts . . . 9
2.4 Schematic representation of the difference in compression history of a splicing . . . 10
2.5 Example of Noiseprints of spliced images . . . 12
2.6 Example of Comprints of a spliced images . . . 13
2.7 Overview of the TruFor architecture . . . 14
2.8 Performance of Noiseprint++ vs Noiseprint . . . 15
2.9 Example of TruFors confidence maps and forgery score . . . 16
2.10 Example of fourier spectra fingerprints of different GenAI models . . . 18
2.11 Diagram of how Foundation Models can be used for SID . . . 20

3.1 Example of the impact of distractions on TruFor . . . 22
3.2 Impact of JPEG compression on prediction of a image with a distraction . . . 24
3.3 Overview of distraction logos used in the experiments . . . 26
3.4 Example of an image with a applied distraction used in the experiments . . . 26
3.5 Overview of the distracted image creation process used in the experiments . . . 27
3.6 Examples of manipulated images with corresponding TruFor predictions . . . 31
3.7 Percentage plots of TruFor on four datasets . . . 32
3.8 Percentage plots for TruFor and PSCC-net aggregated over four datasets . . . 33
3.9 Scatter plot of localization scores of TruFor on different datasets . . . 34
3.10 Scatter plots for TruFor, CAT-Net and PSCC-net aggregated over four datasets . . . 34
3.11 ROC curves of NPR and PatchCraft over all four datasets . . . 37
3.12 Scatter plots of the predictions of NPR over all four datasets . . . 39
3.13 Scatter plot of the predictions of PatchCraft over all four datasets . . . 40

4.1 Overview of the post processing solution incorporated in the TruFor architecture . . . 42
4.2 Illustration of low-probability remnants in predictions of distracted images . . . 43
4.3 Overview of the extra input solution incorporated in the TruFor architecture . . . 44
4.4 Overview of the adapter solution incorporated in the TruFor architecture . . . 46
4.5 Schematic overview of the observation behind the masked-convolution solution . . . 47
4.6 Training and validation loss and IoU of the adapter . . . 52
4.7 Convolutional layer explanation . . . 53
4.8 Illustration of the three steps taken by the proposed masked convolution layer . . . 54
4.9 Schematic overview of the integration of the masked components into the TruFor architecture 55

5.1 Example of unmodified TruFor vs. the proposed solutions . . . 58
5.2 Examples of manipulated images with corresponding predictions by TruFor and the proposed filter adapter approach . . . 62
5.3 Examples of manipulated images with corresponding predictions by TruFor and the proposed masked-convolution approach . . . 65
5.4 Scatter plots for unmodified TruFor, the adapter solution, and the masked convolution solution aggregated over all four datasets . . . 68

A.1 Examples of distraction at different text sizes . . . 79
A.2 Impact of distraction text size on TruFor . . . 80

A.3 Examples of distraction with different fonts and weights 80
A.4 Overview of logos used to study the impact of shape and color 81

B.1 Scatter plots of localization scores of DCT on different datasets 85
B.2 Percentage plots of localization scores of DCT on different datasets 85
B.3 Scatter plots of localization scores of ADQ1 on different datasets 86
B.4 Percentage plots of localization scores of ADQ1 on different datasets 86
B.5 Scatter plots of localization scores of BLK on different datasets 87
B.6 Percentage plots of localization scores of BLK on different datasets 87
B.7 Scatter plots of localization scores of CAGI on different datasets 88
B.8 Percentage plots of localization scores of CAGI on different datasets 88
B.9 Scatter plots of localization scores of TruFor on different datasets 89
B.10 Percentage plots of localization scores of TruFor on different datasets 89
B.11 Scatter plots of localization scores of PSCC-Net on different datasets 90
B.12 Percentage plots of localization scores of PSCC-Net on different datasets 90
B.13 Scatter plots of localization scores of CAT-Net on different datasets 91
B.14 Percentage plots of localization scores of CAT-Net on different datasets 91
B.15 Examples of manipulated images with corresponding TruFor predictions 92
B.16 Examples of manipulated images with corresponding CAT-Net predictions 93
B.17 Examples of manipulated images with corresponding PSCC-Net predictions 94

C.1 Scatter plots of predictions of CNNDetect 96
C.2 ROC curve of CNNDetect 96
C.3 Scatter plots of predictions of DIMD 97
C.4 ROC curve of DIMD 97
C.5 Scatter plots of predictions of GramNet 98
C.6 ROC curve of GramNet 98
C.7 Scatter plots of predictions of PatchCraft 99
C.8 ROC curve of PatchCraft 99
C.9 Scatter plots of predictions of NPR 100
C.10 ROC curve of NPR 100
C.11 Scatter plots of predictions of Fusing 101
C.12 ROC curve of Fusing 101
C.13 Scatter plots of predictions of UnivFD 102
C.14 ROC curve of UnivFD 102
C.15 Scatter plots of predictions of RINE 103
C.16 ROC curve of RINE 103

List of Tables

2.1 Overview of FL models and their detection strategies 16
2.2 Overview of SID models and their detection strategies 21

3.1 Overview of datasets used for forgery localization experiments 25
3.2 Results distraction experiment on FL models 28
3.3 Overview of datasets used for synthetic image detection experiments 35
3.4 Results distraction experiment on SD models 38

4.1 Overview of possible solutions for distraction-aware FL with advantages and disadvantages 48
4.2 Overview of datasets used for training the adapter 51

5.1 Evaluation of Filter Adapter solution 61
5.2 Evaluation of Masked Convolution solution 63
5.3 Results of ablation study on Masked Convolution solution 64
5.4 Summary: Full results distraction experiment on FL models and proposed solutions . . . 66

A.1 Impact of font type and weight on TruFor 80
A.2 Impact of logo shape and color on TruFor 81
A.3 Impact of logo position on TruFor 82

List of Acronyms

AUC Area Under Curve

CLIP Contrastive Language-Image Pretraining

CM-FRM Cross-Modal Feature Rectification Module

CNN Convolutional Neural Network

DCT Discrete Cosine Transform

DnCNN Denoising CNN

DFD Deepfake Face Detection

DQ Double Quantization

FFM Feature Fusion Module

FL Forgery Localization

GenAI Generative AI

IoU Intersection over Union

LLM Large Language Model

LVM Large Vision Model

MiT-B2 Mix Transformer encoder B2

ML Machine Learning

MLP Multi-Layer Perceptron

NLP Natural Language Processing

NP Noiseprint

NP++ Noiseprint++

NPR Neighboring Pixel Relationship

PEFT Parameter Efficient Fine-tuning

ROC Receiver Operating Characteristic

RQ Research Question

PRNU Photo-Response Non-Uniformity

SOTA state-of-the-art

SD Synthetic Detection

SID Synthetic Image Detection

ViT Vision Transformer

1

Introduction

In recent years, fake news has emerged as a **significant global issue**. People are highly susceptible to misinformation, which can shape their opinions, influence their decisions, and ultimately impact large-scale societal domains such as politics, marketing, and public health. Social media platforms have become the primary battlegrounds for the spread of misinformation, false narratives, conspiracy theories, and manipulated multimedia content. The **consequences of misinformation can be severe**, swaying public opinion, deepening societal divisions, and undermining trust in institutions.

This problem is further amplified by the **growing sophistication and accessibility of image editing technologies**. High-quality image manipulation is no longer confined to experts. Powerful editing tools are widely available and increasingly user-friendly. Moreover, the rise of Generative AI (GenAI) models, such as Stable Diffusion [1], FLUX [2], MidJourney [3], and Sora [4], has introduced new possibilities for creating hyper-realistic synthetic images and videos. These models enable seamless inpainting and full image generation with minimal effort. Even mainstream software like Adobe Photoshop now integrates GenAI-powered tools, such as Generative Fill (see Fig. 1.1), making advanced manipulation capabilities accessible to a wide audience.

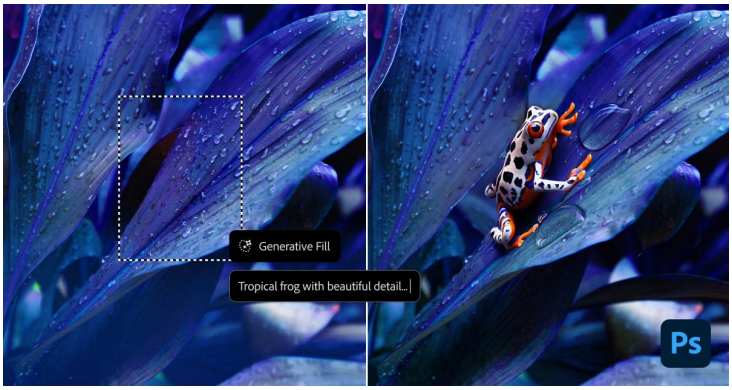


Figure 1.1: Example of Adobe Photoshop Generative Fill

These technologies empower creativity and innovation. They allow artists and the entertainment industry to push creative boundaries, enabling an expression of creativity like never before. While there is clear potential for harmless use, they also pose **significant risks when misused**:

In malicious contexts, these tools can be exploited to **fabricate fake news**, influencing public perception, fueling conspiracy theories, and eroding trust in legitimate news sources. E.g., Figure 1.2 show a viral fake image of Zelenskyy posing with a large amount of cash suggesting there was a high level of corruption in Ukraine, thereby influencing publics perception on the war.

They can also **enable impersonation**, leading to identity fraud, political manipulation, or reputational damage. A recent example occurred when a finance worker was tricked into transferring $25 million after a deepfaked video call with a scammer impersonating their company's chief financial officer [5].



Figure 1.2: Fake images can be used to create fake news and influence public opinion. This image shows Zelenskyy posing with a large amount of money suggesting corruption is at play.

Additionally, these technologies can **facilitate fraud**, allowing scammers can create highly convincing fake profiles, documents, financial records and images. E.g., Figure 1.3 shows a real-life case of insurance fraud, where a person filed a fraudulent insurance claim by manipulating photos of a car and forging a repair invoice [6].



Figure 1.3: Fake images can be used to facilitate insurance fraud. The image of the car was manipulated to make it appear as if it was involved in an accident.

Given these developments, it has become increasingly important to detect forged or synthetic visual media. The field of **multimedia forensics** focuses on analyzing digital content to determine its authenticity. By combining techniques from computer vision, signal processing, and machine learning, this field aims to develop robust tools for detecting image manipulations and synthetic media.

These forensic tools have a **wide range of applications** across various sectors, such as media and journalism organizations (e.g., VRT, European Broadcasting Union), fact-checking platforms (e.g., deCheckers, Factcheck.Vlaanderen, Knack), law enforcement agencies (e.g., Federal Police), cybersecurity institutions (e.g., European Union Agency for Cybersecurity-ENISA), and social media platforms to combat the spread of misinformation and uphold information integrity.

The field of multimedia forensics consists of two main tasks: **Forgery Localization (FL)** and **Synthetic Detection (SD)**. FL focuses on identifying and localizing manipulations, while SD aims to detect full AI-generated media. Both branches rely on the analysis of forensic traces, which are artifacts left behind by manipulation and GenAI processes. These traces can be subtle and difficult to detect, making the development of effective forensic tools a challenging task.

The societal impact of misinformation is far-reaching, affecting public trust, political stability, and social cohesion. As manipulated and synthetic media become increasingly prevalent, the role of multimedia forensics grows ever more critical. By addressing the challenges posed by misinformation, forensic tools play a key role in safeguarding the integrity of information, supporting informed decision-making, and strengthening the resilience of society against manipulation and deception.

1.1 Problem Statement

Despite the significant progress in the field of multimedia forensics, current state-of-the-art detection models still face several **critical limitations** that hinder their practical reliability. While many models demonstrate impressive performance under controlled conditions or specific manipulation types, their robustness and generalizability in real-world scenarios remain limited.

One core issue is **limited generalization across manipulation types**. Two main classes of forensic models exist, being hand-crafted models and deep-learning based models. Handcrafted models are often created with specific manipulations in mind. Deep-learning models, on the other hand, are trained on a select number of datasets, which only contain certain forgeries. This leads to many forensic models only being robust to a narrow set of forgery techniques, such as inpainting, splicing, or copy-move.

They will struggle to maintain performance when confronted with forgeries outside their training domain. For instance, a model optimized for inpainting detection may fail to identify splicing artifacts, and vice versa. This specialization leads to fragility when models are deployed in the wild, where manipulation techniques are often varied and unpredictable.

Another major challenge lies in **degradation under compression and post-processing**. Fake images are often reshared across platforms, undergoing multiple rounds of compression and resolution changes. This process can obscure manipulation traces and cause models to break down or misclassify the content entirely.

Similarly, some forensic models fail when **multiple forgeries** coexist within the same image, for example, when two people are inserted into a scene rather than just one. This lack of robustness limits their effectiveness in complex or layered manipulation scenarios.

Lastly, it has been noticed that some forensic models tend to **become distracted** by superficial, non-malicious modifications. This has, however, never been formally investigated. In real-world contexts, images often carry overlaid content, such as captions, watermarks, logos, or platform branding. These are elements commonly added during social media sharing or publication on news and blog websites. Figure 1.4 illustrates the effect that benign elements such as a logo can have on the performance of a forensic model.

As these additions are technically *manipulations*, they are expected to be detected in FL tasks. Nonetheless, some forensic models disproportionately focus on these distractions, detecting them as fake while overlooking the actual malicious forgery within the image. To a far lesser extent, distractions also impact the performance of some SD models. This behavior **significantly reduces the practical usability** of such models for analyzing images shared online, where non-malicious overlays like captions, watermarks, and logos are prevalent.



(a) Manipulated image with a logo added

(b) Prediction of the non-distracted image

(c) Prediction of the distracted image

Figure 1.4: Illustration of the problem of forensic distractions. The left image shows a manipulated image with a distraction added in the form of a logo. The logo is not malicious in nature, but it does negatively impact the performance of forensic models.
The middle image shows TruFor's prediction (a state-of-the-art FL model) of the non-distracted image (i.e., without the logo present), and the right image shows TruFor's prediction of the distracted image (i.e., with the logo). The addition of the logo causes TruFor to fail detection of the forgery.

These challenges reveal a gap between theoretical performance and practical utility. Improving general robustness and adaptability remains an open research problem. To this end, this thesis specifically focuses on the underexplored susceptibility of forensic models to distraction and explores strategies to mitigate their effects.

Addressing this issue is essential to enhance the reliability, interpretability, and trustworthiness of multimedia forensic tools in realistic, noisy, and content-rich contexts, particularly in high-stakes environments such as journalism, law enforcement, and online content moderation.

1.2 Research Questions

The aim of this thesis is to investigate the impact of forensic distractions, defined as benign, non-malicious overlays such as captions, watermarks, and logos, on the performance of image forgery detection models, and to explore strategies for increasing model robustness in such conditions.

To that end, this thesis aims to answer two Research Questions (RQs):

RQ 1: What is the impact of forensic distractions on existing Forgery Localization (FL) and Synthetic Image Detection (SID) models?

This question aims to evaluate a representative set of, both FL and SD, state-of-the-art forensic methods, to determine which ones are susceptible to distractions and which demonstrate resilience. The goal is to quantify the extent to which these distractions interfere with the accurate localization of manipulated regions of Forgery Localization (FL) models and the classification performance of Synthetic Image Detection (SID) models.

RQ 2: To what extent can models that are sensitive to distractions be made more robust, either through architectural design or training strategies?

Based on the insights from the evaluations, this thesis explores the feasibility of improving model robustness. This involves experimenting with training techniques and model adaptations aimed at reducing distraction sensitivity, and assessing their effectiveness on diverse and realistic test cases.

By answering these questions, this thesis aims to contribute to a deeper understanding of the limitations and potential improvements of current multimedia forensic tools in real-world, content-rich environments.

1.3 Thesis Outline

This thesis is structured into eight chapters. The first two chapters provide an overview of the field of multimedia forensics. The third chapter presents the problem of forensic distractions. The fourth chapter discusses potential solutions, which are evaluated in the fifth chapter. Chapter size outlines potential future work, and chapter seven provides a societal reflection. Finally, chapter eight concludes the thesis. The following is a more extensive overview of the different chapters.

Chapter 1: Introduction

Chapter 1 introduces the problem of misinformation and manipulated media. It outlines the motivation for this research, presents the core research questions, and situates the study within the broader field of multimedia forensics.

Chapter 2: Related Work

This chapter provides an overview of a selection of existing approaches for image forensics. It outlines the core differences between FL and SID and explains how models use forensic traces to perform their task. This chapter lays the foundation for understanding the objectives and workings of forensic models discussed in this thesis.

Chapter 3: Motivation - The Impact of Forensic Distractions

Chapter 3 defines the concept of forensic distractions and illustrates their disruptive effects through empirical evaluation. An experiment is conducted across a variety of models to assess the impact of such distractions, answering RQ 1.

Chapter 4: Distraction-aware Forgery Localization

This chapter discusses potential solutions to increase the robustness of models to forensic distractions and evaluates their trade-offs. It answers RQ 2 by implementing solutions for the TruFor model and presenting the made adaptations and architectural choices.

Chapter 5: Evaluation

In this chapter, the proposed solutions are evaluated on a variety of datasets and compared to the original TruFor model. The results are presented and discussed, highlighting the effectiveness of the proposed methods in improving model robustness against distractions.

Chapter 6: Future Work

Chapter 6 outlines potential directions for extending the performed research.

Chapter 7: Societal Reflection

Chapter 7 presents a reflection on the broader societal relevance of this work.

Chapter 8: Conclusion

This chapter summarizes the main findings of this thesis: the impact of forensic distractions on forensics models, the proposed methods to improve robustness, and possible directions for future research.

1.4 Use of Generative AI

In this final section of the introduction, I would like to discuss the use of generative artificial intelligence (GenAI) in the context of this thesis. GenAI was used in two main ways. That is, to help with the code and to help with the writing of the thesis itself.

During this thesis, a considerable amount of code was written. This includes code for the performed experiments (see Chapter 3 and Chapter 5), code for the proposed solutions (see Chapter 4), and code for the visualizations of results shown throughout the thesis.

To help me with the development of this code, I used two GenAI tools, being ChatGPT[1] and GitHub Copilot[2]. GitHub Copilot is integrated into my IDE (Visual Studio Code), which made it easy to use for small code snippets or to debug small pieces of code. ChatGPT was used to debug larger pieces of code, such as the dataloader and training loops. I want to stress that I did not use GenAI to write entire pieces of code. The main code is always written by me, and GenAI was only used to speed up the process.

All code generated by GenAI was thoroughly reviewed and modified by me. This is important, as GenAI is not perfect and can make mistakes.

Writing an entire thesis can be overwhelming. To ease this process, some GenAI tools were used to help with the writing of the thesis itself.

GenAI was mostly used in the first draft phase. To start, I would create an outline of everything I wanted to say. This could be in bullet points or in some small paragraphs. I then used GenAI to create a first draft of the section. This helped to get over the initial writer's block. Starting a new section can be quite hard. Everything I write feels a bit off, as I can't seem to find the right words. By using GenAI to create a first draft, I was able to first focus on the most important part, being the content and the structure of the section, instead of the wording itself.

All text written by GenAI was reviewed and rewritten by me in a second phase. In this phase, I went over the created first draft, rewrote it in my own words/style, and added any missing information. The help of my counsellor, Dr. ir. Hannes Mareen, was invaluable in this phase.

In a third and final phase, I polished the text, highlighting important information and making sure everything was clear and easy to read. In this phase, GenAI was sometimes used to reword sentences or give synonyms for words to avoid repetition.

The text you are reading now is the result of this careful, iterative process, which combined my own writing with the assistance of GenAI and the feedback from my counsellor.

Lastly, I want to note that this was my first time using LaTeXfor such a large project. As I do not have extensive knowledge of LaTeX, I used both online forums such as `tex.stackexchange.com` and GenAI to help me with the formatting of text, tables, and figures.

[1] `https://chat.openai.com/` (using GPT-4o by OpenAI, `https://platform.openai.com/docs/models/gpt-4o`)

[2] `https://github.com/features/copilot` (using GPT-4.1 by OpenAI, `https://platform.openai.com/docs/models/gpt-4.1`)

2

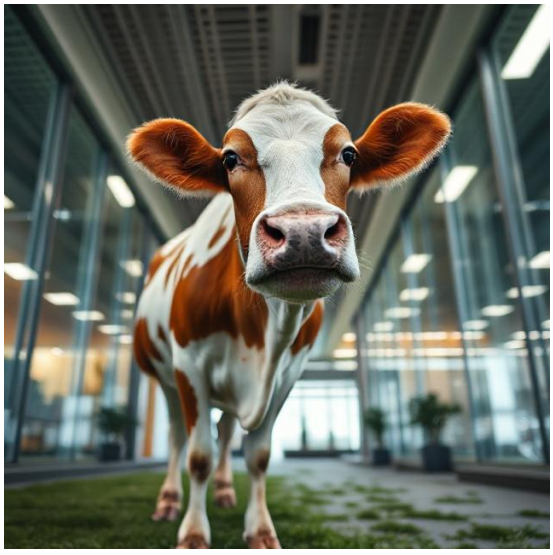
Related work

Image forensic models can be categorized into two types, FL models and SD models. Figure 2.1 illustrates their different goals. Both approaches look for subtle traces left behind during image editing or synthetic image generation. Understanding the fundamental differences between FL and SD in terms of their objectives and their outputs is important, as is grasping the mechanisms that enable the detection and localization of forgeries.

This chapter provides an overview of how FL and SD are performed and highlights a selection of state-of-the-art models in each category. It serves as a reference for the models used in the two experiments performed in Chapter 3. Section 2.1 discusses models for FL, while Section 2.2 explores SD models.



(a) Manipulated image of a pilot holding a selfie stick. The manipulation is highlighted in orange



(b) Synthetic image of a cow in an office building.[1]

Figure 2.1: Forgery localization (a) aims to identify the manipulated region in the image, while synthetic image detection (b) focuses on classifying the entire image as either authentic or synthetic.

[1] Image generated via `https://deepai.org/machine-learning-model/text2img`

2.1 Forgery Localization

The goal of FL models is not just to detect forgeries, but also localize them. The **output** of an FL method is an **anomaly- or forgery mask**. The model calculates a forgery probability for each pixel in the original image. This score is an indication of how likely the pixel is forged (i.e., it was not part of the original image). FL techniques aim to **detect local alterations by analyzing subtle forensic traces**. These traces are small artifacts or inconsistencies across the image which are introduced somewhere along the forgery process. Often used forensic traces are camera noise, compression artifacts and deepfake traces. These traces are discussed in Section 2.1.1. FL models can be split up into two main categories, being model-based methods, discussed in Section 2.1.2 and data-driven deep learning methods, discussed in Section 2.1.3. An overview of the discussed models is provided in Table 2.1.

2.1.1 Forensic Traces

Camera Noise

A commonly used forensic trace is the presence of inconsistencies in camera noise [7, 8]. When taking a picture, the **camera sensor introduces a certain noise pattern** or fingerprint onto the image. This pattern is called a Photo-Response Non-Uniformity (PRNU) fingerprint. It is not visible with the human eye, but it can be used to detect if two images were taken with different cameras. In Figure 2.2, the PRNUs and Noiseprints (discussed later) are shown for two different cameras. It can be seen that for different cameras the fingerprints are different. Forged images are often spliced. Splicing is copying one part of an image (the source image) and pasting it in a different or even the same image (the destination image). When the source and destination image were produced by a different camera, the camera noise fingerprints will be different. This allows detection models to pinpoint the splicing. Examples of models that exploit camera noise are Noiseprint [7] and TruFor [8]. Both are discussed in Section 2.1.3.

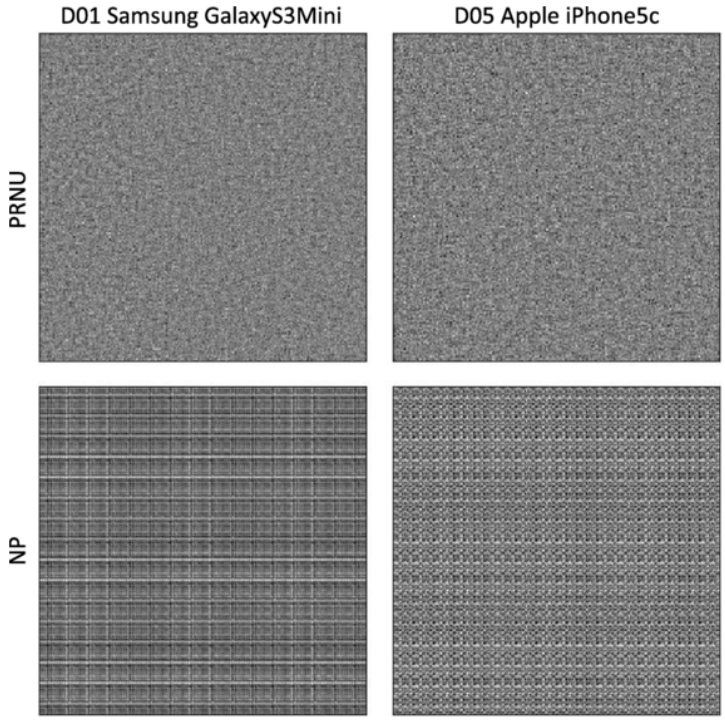


Figure 2.2: The PRNUs (top row) and Noiseprints (bottom row) are calculated for two different cameras. As the cameras are different, the corresponding PRNUs and Noiseprints are also different. By analyzing these fingerprints, manipulations taken by a different camera than the rest of the image can be detected. Image reproduced from Cozzolino et al. (2020) [9]

Compression Artifacts

Another often used forensic trace is the presence of inconsistencies in compression artifacts [10]. Compression artifacts are artifacts introduced by **lossy compression** methods such as JPEG. Information is lost, during lossy compression, and this **creates both visible and invisible changes in the image**. Figure 2.3 shows some common visible compression artifacts, being mosaicing, visible DCT-blocks and ringing. Just as with PRNU fingerprints, splicings can be localized by looking at differences in these compression artifacts. The source and destination image will have a different compression history, as shown in Figure 2.4. This history is composed of a part before the splicing and a common part after the splicing. This difference in history allows detection models to detect the manipulated pixels. A common example is Double Quantization (DQ), where the original image undergoes two rounds of compression, while the spliced region is only compressed once, being after it is inserted.

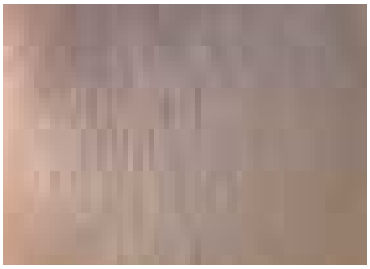
Examples of models that study compression artifacts are model-based methods such as DCT [11], ADQ1 [12], BLK [13] and CAGI [14], and data-driven methods such as Comprint [10] and CAT-Net [15]. TruFor uses Noiseprint++ which also incorporates the compression histories. The model-based methods are discussed in Section 2.1.2, and both Comprint and CAT-Net is discussed in Section 2.1.3



(a) JPEG compressed image



(b) Mosaicing



(c) Visible DCT-blocks



(d) Ringing

Figure 2.3: When an image (a) is lossily compressed, compression artifacts can be introduced. Some common artifacts are mosaicing (b), visible DCT-blocks (b), and ringing (c).

Deepfake Traces

Deepfake Face Detection (DFD) can be seen both as a type of FL or SD. Deepfake techniques use GenAI models to generate highly realistic facial forgeries. Since these manipulations alter facial regions, FL methods can be applied to localize the tampered areas. However, because the primary objective is often to determine whether a face is synthetic, SD approaches are also relevant. As a result, DFD sits at the intersection of FL and SD, and both types of techniques are commonly employed.

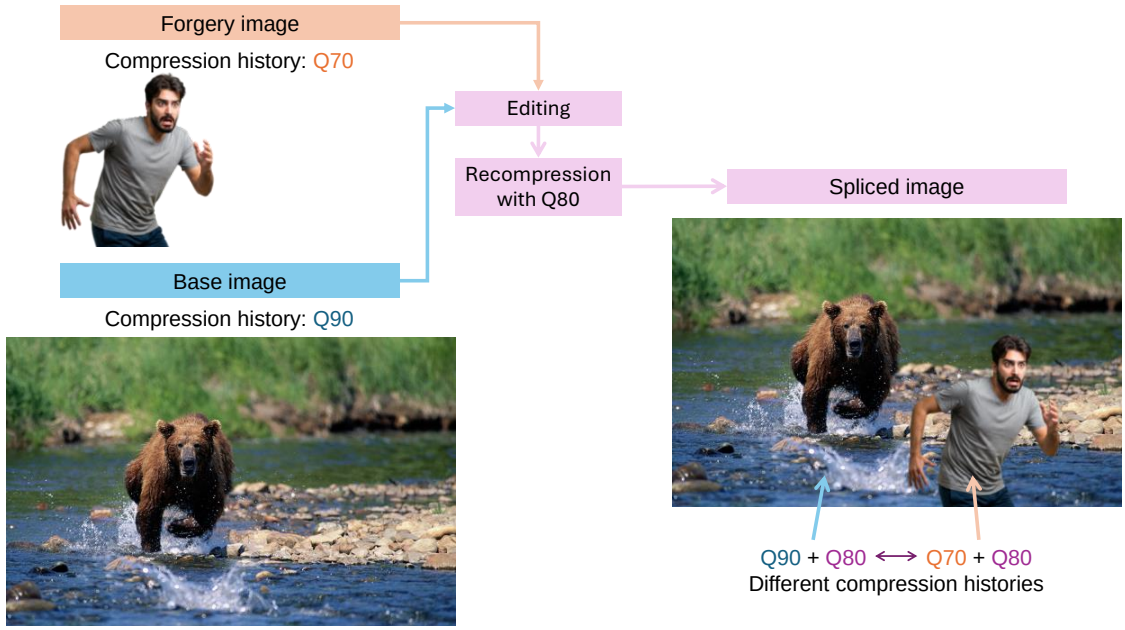


Figure 2.4: A splicing consists of a base image and a forgery. Both the base image and the forgery have been subjected to different compressions in the past, called their compression history. The splicing is later again compressed, altering their histories. However, the base image and forgery still have a different compression history, making it possible to distinguish between both.

It is, however, often discussed separately, as it specifically focuses on detecting manipulated or synthetic faces. DFD models often rely on detecting subtle artifacts introduced during the generation process, such as inconsistencies in facial features, unnatural textures, or mismatched lighting. DFD is not further discussed in this thesis.

2.1.2 Model-Based Methods

Model-based forgery localization methods **rely on hand-crafted features and explicit assumptions** about the statistical or physical properties of authentic images. These models are often based on detecting anomalies in JPEG compression artifacts. Their performance often depends heavily on the validity of their underlying assumptions, which may not hold in real-world, in-the-wild, scenarios where images are frequently recompressed multiple times or edited in unforeseen ways.

DCT

One well known model-based approach to image forgery detection is based on DCT blocking artifact analysis. JPEG compression introduces characteristic blocking artifacts due to the use of Discrete Cosine Transform (DCT) on 8×8 image blocks. The method proposed by Ye et al. (2007) [11] detects inconsistencies in these JPEG artifacts by, first, estimating the underlying JPEG quantization table and, second, measuring the difference between actual DCT coefficients and their expected quantized values for each block in the image. This difference is called the Blocking Artifact Measure (BAM). By analyzing local deviations in BAM across the image, the method can identify tampered regions.

ADQ1

An advancement over traditional DCT-based forgery detection is ADQ1 (2009) [12], which targets fine-grained localization of tampered regions in JPEG images by analyzing Double Quantization (DQ) effects in DCT coefficients. ADQ1 is relatively fast and is able to provide an anomaly mask with block-level resolution.

ADQ1 builds histograms for multiple DCT frequencies across YUV color channels and identifies periodic structures caused by DQ effects. It applies a naive Bayesian approach to estimate the likelihood that each 8×8 block is either tampered or untampered based on how well it aligns with these periodic artifacts. These per-block estimates are aggregated into a Block Posterior Probability Map (BPPM). This BPPM is then thresholded and segmented to localize the manipulated regions. Finally, a trained SVM classifier uses features extracted from the BPPM to confirm image-level tampering.

BLK

BLK (2009) [13] is another model-based method. It focuses on detecting inconsistencies in the Block Artifact Grid (BAG) introduced during JPEG encoding. JPEG compression divides images into 8×8 blocks, and the block boundaries form a weak but regular grid known as the BAG. When an image is manipulated using techniques like copy–paste, inpainting, or cropping, the alignment of these grids can become inconsistent.

The BLK method introduces a blind algorithm to extract BAGs by enhancing weak periodic edge structures using second-order differences and periodic median filtering. The extracted BAG image is analyzed for anomalies. A marking procedure is used to automatically highlight inconsistencies, such as areas where the grid is missing, faint, or misaligned. This approach is particularly effective for detecting tampering that disrupts the natural alignment of JPEG block structures, even in cases where other statistical methods fail.

CAGI

CAGI (2018) [14] stands for Content-Aware detection of Grid Inconsistencies. It works by detecting JPEG grid misalignments, similar to BLK. CAGI identifies such disruptions by exhaustively estimating grid positions and measuring their local confidence scores.

CAGI has two key improvements compared to BLK. First, a grid alignment confidence measure that improves robustness to image content and compression quality. Second, a content-aware filtering step that suppresses false positives caused by strong edges, over- or underexposed regions, or highly textured areas.

CAGI generates heatmaps to highlight tampered areas by analyzing grid alignments and combining auxiliary maps. It also includes an inverse mode (inv-CAGI) for cases where tampering strengthens grid traces rather than weaker grid traces. CAGI offers both accurate and interpretable results.

2.1.3 Data-Driven Deep Learning Methods

In contrast to model-based techniques, data-driven deep learning methods **learn to detect and localize forgeries directly from data** by training on large collections of manipulated and pristine images. These approaches typically rely on Convolutional Neural Networks (CNNs) and Vision Transformers to capture both low-level forensic traces and high-level semantic inconsistencies that may arise from tampering.

Data-driven models can be supervised but can also follow a one-class training paradigm. In **one-class training**, the model learns what pristine, authentic data looks like instead of learning to distinguish forged data and authentic data. As the model does not just learn to detect the kind of forgeries present in the dataset, it generalizes better to unseen forgeries. Both Noiseprint and Comprint are examples of one-class trained models.

Deep learning-based methods have **strong performance**, especially when trained with diverse and realistic forgeries. They often generalize better across manipulation types and image domains, particularly when explicitly designed for in-the-wild robustness.

Noiseprint

One notable model that leverages PRNU-like traces for image forgery localization is Noiseprint (2020) [7]. Unlike traditional PRNU-based methods that attempt to isolate device-specific fingerprints, Noiseprint focuses on extracting camera model-level fingerprints. It introduces a deep learning approach to generate noise residuals that suppress scene content while enhancing forensic artifacts caused by the in-camera processing pipeline. The model utilizes a CNN, based on the Denoising CNN (DnCNN) [16] backbone. This CNN is trained with the Siamese paradigm, to distinguish between image patches from the same or different camera models. By doing so, it learns to highlight consistent artifacts inherent to the camera model and suppresses other unrelated content. The resulting residual, termed the *noiseprint*, can be interpreted as a camera model-specific forensic trace.

These noiseprints can reveal manipulations, such as splicing, through visual or algorithmic analysis, as inconsistencies in the pattern can indicate the presence of forgeries. Importantly, Noiseprint is a one-class learning method, and the training process is completely unsupervised with respect to the test data (i.e., manipulated images), making the model robust across different datasets and unseen camera models.

An example of two spliced images and their corresponding noiseprints is shown in Figure 2.5. The forgeries become visible through inconsistencies in the extracted residual. In Figure 2.2, a comparison between Noiseprint and PRNU is shown. It can be seen that the Noiseprint fingerprint is much more structured than the PRNU fingerprint.



Figure 2.5: Example of spliced images (left side) and their corresponding Noiseprints (right side). The forgeries are not visible in the original images, but do become visible in the extracted Noiseprints due to inconsistencies in the camera noise fingerprint. Image reproduced from Cozzolino et al. (2020) [7]

Comprint

Another fingerprint-based approach is Comprint (2023) [10], which focuses on the compression history of an image rather than the camera model. Comprint introduces a method to extract a compression fingerprint, termed a *comprint*, by training a deep neural network to detect inconsistencies in JPEG

compression artifacts. Similar to Noiseprint, the model employs a CNN based on the DnCNN [16] backbone, that is trained with the Siamese paradigm, but instead of learning camera-specific traces, it learns to distinguish between patches that have different JPEG compression histories. It uses only pristine training data, making it a one-class learning method capable of generalizing to many different manipulations. During inference, the extracted comprint is passed through a blind localization algorithm to produce a heatmap indicating likely tampered regions. This makes Comprint particularly effective in real-world conditions, where recompression often obscures traditional forensic traces. An example of two manipulated images and their corresponding comprints is shown in Figure 2.6.



Figure 2.6: Example of spliced images (left side) and their corresponding Comprints [10] (right side). The forgeries are not visible in the original images, but do become visible in the extracted Comprints due to inconsistencies in the compression artifacts.

CAT-Net

CAT-Net (2022) [15] is a fully CNN-based model. Designed to detect and localize image splicing by jointly analyzing visual features and compression artifacts. The model consists of two parallel streams: an RGB stream for learning visual-level cues, and, a DCT stream for capturing JPEG compression artifacts. CAT-Net uses HRNet [17] as its backbone, allowing it to retain high-resolution features throughout the network and effectively fuse multi-scale information. The outputs of the RGB and DCT streams are merged in a final fusion stage, producing a binary mask that highlights tampered regions.

One of CAT-Net's key strengths is its robustness across both JPEG and non-JPEG images, as it simulates DCT features for non-compressed inputs by using a JPEG encoder with quality 100. By explicitly modeling compression artifacts alongside image content, CAT-Net demonstrates strong generalization across different manipulations.

PSCC-Net

PSCC-Net (2022) [18] is a forgery localization method that does not rely on compression artifacts, but instead models spatio-channel correlations to detect any kind of manipulation. PSCC-Net consists of two paths: a top-down path that extracts multi-scale local and global features, and a bottom-up path

that progressively refines manipulation masks from coarse to fine resolutions using SCCM modules. The Spatio-Channel Correlation Module (SCCM) captures both spatial and channel-wise correlations at each refinement step, producing intermediate feature maps. Each intermediate map is passed to the next SCCM module, progressively improving the localization accuracy at higher resolutions.

TruFor

TruFor (2023) [8] is one of the best state-of-the-art deep learning-based methods. It combines both low-level forensic traces and high-level semantic information using a modular architecture. A key part of TruFor is the integration of a refined residual fingerprint called Noiseprint++ (NP++). In contrast with many state-of-the-art FL models, TruFor also produces a confidence map and a forgery score. The addition of both components leads to an increase in interpretability and reliability, and makes it possible to not only perform localization but also image-level detection. Answering the question "Is this image forged?".

In Chapter 4 the TruFor model is altered to make it more robust against forensic distractions. This requires some knowledge about TruFor's architecture, therefore, a technical overview of the different components that make up TruFor is provided here.

General Architecture TruFor follows a multi-stage architecture, which can be seen in Figure 2.7, consisting of five main components: a Noiseprint++ Extractor, a shared Encoder, two parallel Decoders, respectively, producing an anomaly map and a confidence map, and a final Forgery Detector that estimates a global integrity score. The RGB image and its extracted residual NP++ are jointly encoded to extract features. These features are then decoded along two paths. The Anomaly Decoder is focused on identifying likely forgeries, and the Confidence Decoder on assessing the trustworthiness of the localization. The final forgery score is computed by the Forgery Detector using pooled features from both maps.

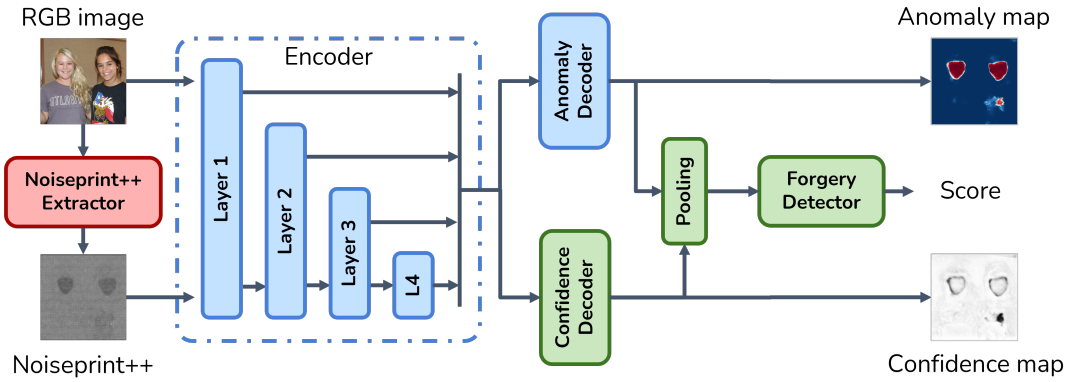


Figure 2.7: The TruFor architecture consists of five main components: The Noiseprint++ Extractor, the Encoder, the Anomaly Decoder (produces the anomaly map), the Confidence Decoder (produces the confidence map) and the Forgery Detector (produces the forgery score). Image reproduced from Guillaro et al. (2023) [8]

Noiseprint++ (NP++) The strength of TruFor is partially due to the use of NP++. NP++ is an improvement of the previously discussed Noiseprint. It combines concepts from Noiseprint and Comprint, i.e., using both camera and compression noise traces, and incorporates more editing history traces. The model is trained via contrastive learning and is based on the DnCNN [16] backbone, similar to its predecessors. Figure 2.8 shows a spliced image with both the Noiseprint++ and the original Noiseprint. It can be seen that NP++ performs noticeably better than the original Noiseprint. Lastly, as NP++ is, just like Noiseprint and Comprint, a one-class learning method, it generalizes well to a wide range of manipulations. This generalization ability is inherited by the full TruFor model.

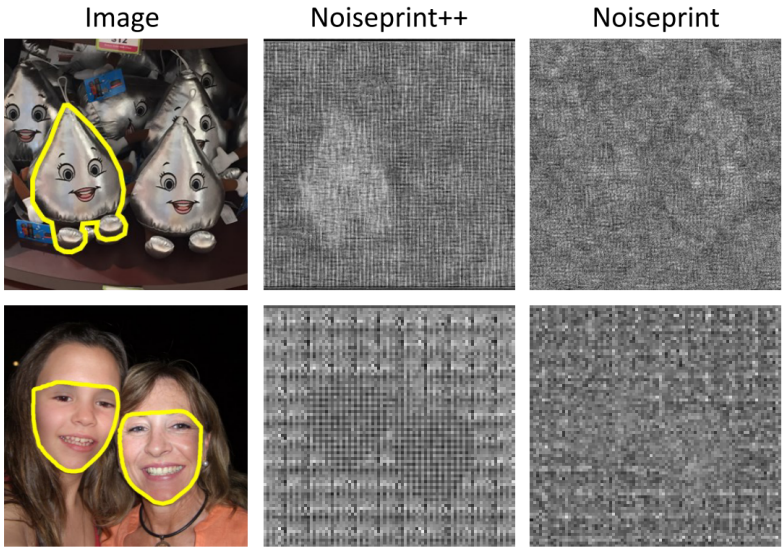


Figure 2.8: Example of spliced images (left) and their corresponding Noiseprint++ (center) and Noiseprint (right). The forged area is more clearly visible in the Noiseprint++ than in the original Noiseprint. Image reproduced from Guillaro et al. (2023) [8]

Encoder The encoder uses a cross-modal transformer-based architecture, adapted from SegFormer [19]. It works in 2 *feature extraction branches*. Each branch passes through a transformer blocked based on Mix Transformer encoder B2 (MiT-B2) [20]. The branches are then processed together using a Cross-Modal Feature Rectification Module (CM-FRM) [19]. This output is then further processed in 3 more similar steps. At each step, the resolution of the feature maps is divided by 4, and the feature maps are passed through a Feature Fusion Module (FFM) to produce one output of the encoder. The 4 resulting outputs are the latent condensed feature maps that are used by the decoders to generate the anomaly mask, the confidence mask and the integrity score.

The CM-FRM block exploits the interactions between the semantic features from the RGB image and the residual features from Noiseprint++. The FFM integrates the RGB and Noiseprint++ features using a lightweight cross-attention mechanism that operates without positional encoding.

Decoders For the two decoders (i.e., the Anomaly Decoder and the Confidence Decoder), All-MLP decoders [20] are used. All-MLP decoders consist only of MLP layers. For every latent feature map, the channels are first unified, and the sets are upscaled to 1/4th the original resolution. Next, the maps are concatenated and passed through 2 more MLP layers, finally resulting in 2 channels, one for real and one for fake pixels.

Anomaly Localization Map The first decoder is the Anomaly Decoder. It produces a high-resolution, pixel-level anomaly map, which highlights the manipulated regions in the image. Higher values indicate larger probability of forgery. In Figure 2.9, the anomaly masks can be seen for both a manipulated and authentic image.

Confidence Map To reduce false positives and increase reliability, TruFor includes a confidence map (generated by the second decoder), which estimates how trustworthy the forgery probability of each pixel is. Bright regions (i.e., high pixel values) in the confidence map indicate high certainty, while dark areas (i.e., low pixel values) suggest low reliability. This can be seen in Figure 2.9.

Forgery Score Both the anomaly and confidence maps are pooled together to form a compact feature vector. This vector is then passed to a binary classifier to generate an integrity score, which provides an image-level prediction of whether the input is manipulated or not. This classifier network consists of 2 fully connected layers with RELU activation. The confidence mechanism plays a crucial role

in filtering out accidental high responses in pristine areas, offering a more interpretable, explainable and robust decision process. This is illustrated in Figure 2.9.

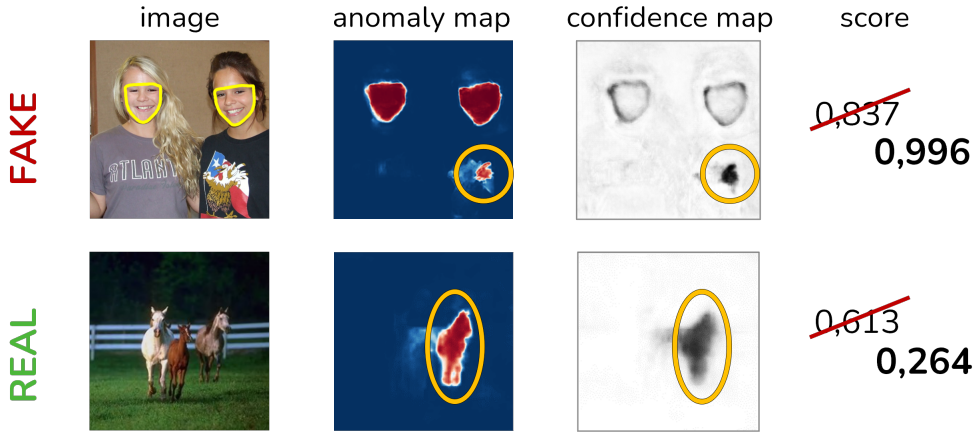


Figure 2.9: Example of both a fake (top row) and real (bottom row) image and their corresponding anomaly map, confidence map and forgery score. White pixels in the confidence map mean the model is confident in its detection, black pixels mean the opposite. The forgery score gets *corrected* based on the calculated confidence map, leading to a more robust output. Image reproduced from Guillaro et al. (2023) [8]

Loss functions TruFor is trained in three steps. In the first step, the Noiseprint++ Extractor is trained using self-supervised contrastive learning. The second step is the joint training of the encoder (pretrained) and the Anomaly Decoder. This is done using a combination of the weighted cross-entropy loss and the dice-loss [21]. The third step is the joint training of the Confidence Decoder and the Forgery Detector. This is done using a combination of the mean squared error loss calculated on the confidence map and the balanced cross-entropy loss on the forgery map.

Training/Validation data The NP++ extractor was trained on a custom dataset, containing about 25,000 publicly available images. The encoder and decoders were trained on a combination of 14 datasets, as proposed by CAT-Net [15], consisting of about 877,000 images. Some included datasets are Fantastic Reality [22], IMD2020 [23], SP COCO, CM COCO, CM RAISE and CM-JPEG RAISE collectively called tampCOCO [15], and JPEG RAISE [15].

	Model	Year	Core Detection Strategy
Model-based	DCT [11]	2007	Double JPEG compression inconsistencies in DCT coefficients
	ADQ1 [12]	2009	Aligned double quantization artifacts in JPEG
	BLK [13]	2009	Block artifact grid (BAG) inconsistencies
	CAGI [14]	2018	Content-aware detection of JPEG grid misalignments
Deep learning	Noiseprint [7]	2020	Camera model fingerprint extraction via Siamese trained CNN
	Comprint [10]	2023	Compression fingerprint extraction via Siamese trained CNN
	CAT-Net [15]	2022	Dual-stream CNN for visual and DCT compression cues
	PSCC-Net [18]	2022	Progressive spatio-channel correlation modeling
	TruFor [8]	2023	Fusion of image content and Noiseprint++ residuals

Table 2.1: Overview of forgery localization models and their detection strategies. The models can be split into three groups. The first four models are model-based methods (Section 2.1.2), while the other models are deep learning-based methods (Section 2.1.3). Noiseprint and Comprint are both fingerprint extraction methods, and are not further discussed in this thesis.

Summary

Forgery Localization (FL) aims to produce **pixel-level anomaly maps** that highlight manipulated regions within an image. FL models **search for inconsistencies** in low-level forensic traces, such as camera noise fingerprints, left behind by the camera sensor, and compression artifacts, introduced by lossy compression algorithms.
These models can be broadly divided into two main categories: **model-based methods**, which rely on **hand-crafted forensic features**, and **data-driven deep learning methods**, which **learn** manipulation patterns directly **from data**. An overview of the discussed models is provided in Table 2.1. While deep learning methods generally achieve higher performance and better generalization across manipulation types, **no method is perfect**. Different models exhibit varying levels of robustness depending on the nature of the manipulation or the presence of post-processing operations such as compression.
Chapter 3 investigates one particular vulnerability of some FL models, the impact of forensic distractions, and analyzes how these distractions affect the localization accuracy of various state-of-the-art approaches.

2.2 Synthetic Image Detection

The second category of image forensic models are SID models. The **output** of these models is **a score, indicating the probability that the image is synthetically generated**. Some techniques also perform model attribution, which involves identifying the specific generative model or architecture used to create the synthetic image [24, 25]. This can provide additional insights into the origin of the image and help in tracing its generation process.

In the **earlier days** of GenAI, it was **quite easy to identify** synthetic images by simply looking for abnormalities, such as extra fingers or distorted faces. A study from 2022 [26] found that the ability of humans to detect synthetic images has significantly declined as generative models have improved. Another study from 2023 [27] found that current AI-generated faces have become so realistic, they are undistinguishable from real faces, often appearing even more realistic than actual human faces.

As **traditional techniques**, based on obvious indication, **do not suffice anymore**, current state-of-the-art models often focus on invisible *fingerprints* introduced by the AI-generation model [25, 28].

For this thesis, eight SID models were chosen for evaluation, representing a diverse set of strategies and techniques. This selection ensures a comprehensive overview of the latest advancements in synthetic image detection. The **models were selected by SIDBench** [29]. SIDBench is a SID benchmark which evaluated a number of state-of-the-art SID models on a large number of different synthetic image datasets. This makes it an excellent resource for selecting relevant and effective approaches. The selected models can be grouped into four categories based on their approach: spectrum-based, local artifact and texture-based, hybrid feature-based, and foundation model-based methods. The following sections briefly describe each category and the selected models. An overview is provided in Table 2.2.

2.2.1 Spectrum-Based Methods

GenAI models leave behind characteristic **fingerprints in the frequency domain**, which can be observed through the image's Fourier spectrum. These spectral fingerprints are unique to the generative model, and can be used not only for detecting synthetic images, but also for model attribution as they are often tied to the architectural choices of the generator. Figure 2.10 illustrates the spectral patterns of different GenAI models. Two spectrum-based methods are discussed, CNNDetect [30] and DIMD [28].

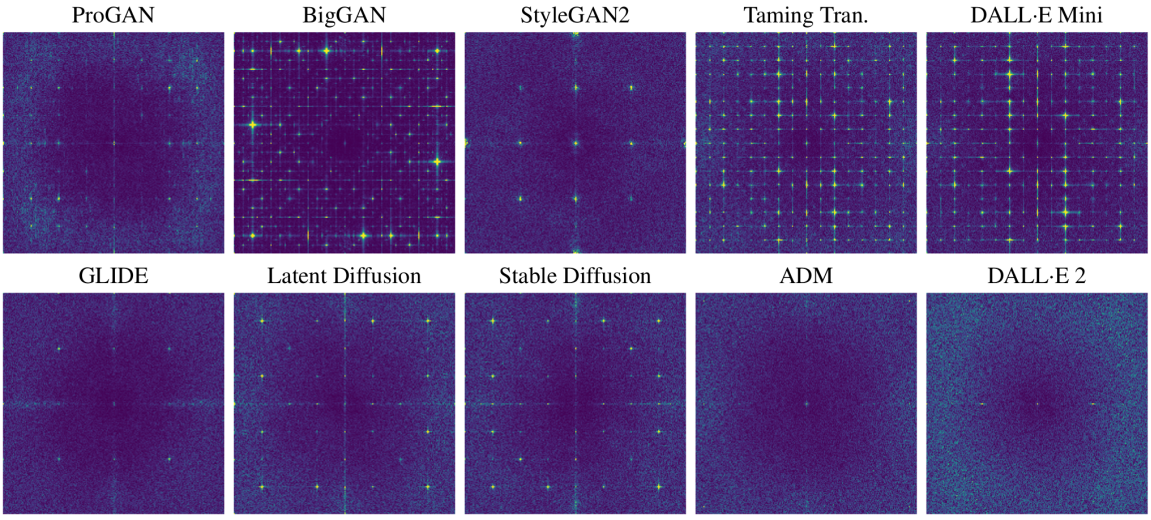


Figure 2.10: Example of the fourier spectra fingerprints left behind by different GenAI models. From left to right, top to bottom: ProGAN [31], BigGAN [32], StyleGAN2 [33], Taming Transformers [34], DALL·E Mini [35], GLIDE [36], Latent Diffusion [1], Stable Diffusion [37], ADM [38], DALL·E 2 [39]. Image reproduced from Corvi et al. (2023) [28]

CNNDetect

CNNDetect (2020) [30] trains a binary classifier on ProGAN images and detects synthetic content by learning common artifacts left by upsampling operations in CNN-based generators. It uses data augmentation and diverse training data to improve robustness and generalization to other generative models.

DIMD

DIMD (2023) [28] targets synthetic image detection specifically for diffusion models. It estimates noise residuals from images and performs spectral analysis to reveal subtle generation artifacts. These fingerprints, though weaker than in GANs, can still distinguish some diffusion models, especially when using detectors trained on similar architectures.

2.2.2 Local Artifact and Texture-Based Methods

Synthetic image generators often leave behind **subtle structural or texture-level inconsistencies** that are difficult to detect using frequency analysis alone. Local artifact and texture-based methods focus on low-level visual cues such as local pixel relationships, textural irregularities, or inter-pixel statistics. These approaches are particularly useful for generalizing across unseen generative models, as they rely on architectural patterns or local irregularities that are common in synthetic content. From this category three models were selected, GramNet [40], PatchCraft [41], and NPR [42].

GramNet

GramNet (2020) [40] enhances fake image detection by incorporating global texture features using Gram matrices. It extends a CNN backbone with *Gram Blocks* that capture long-range texture statistics at multiple semantic levels. These global texture cues help distinguish real from fake images and improve robustness against image editing (e.g., compression, blur) and generalization to unseen GANs or datasets.

PatchCraft

PatchCraft (2024) [41] detects synthetic images by analyzing the contrast in inter-pixel correlations between rich and poor texture regions. It introduces a Smash & Reconstruction step that breaks global semantics and isolates local texture patches, enhancing subtle generative artifacts. High-pass filtering is applied to amplify these artifacts, making the method effective across various GAN- and diffusion-based models.

NPR

Neighboring Pixel Relationship (NPR) (2024) [42] introduces a novel artifact representation that captures local pixel correlations caused by up-sampling layers in GANs and diffusion models. It computes pixel-wise differences within small grids (e.g., 2×2 patches) to model local structural patterns, revealing subtle generation artifacts missed by frequency-based methods. This approach targets generalization by focusing on spatial-domain traces left by upsampling, making it robust across a wide range of generative models.

2.2.3 Hybrid Feature-Based Methods

Hybrid detection methods aim to **combine both global and local visual information** to better capture diverse signs of image synthesis. These models typically extract high-level semantic features while also attending to fine-grained spatial artifacts, allowing them to detect both obvious and subtle generation traces. This use of multiple representation levels improves robustness across a wide variety of generative models and image types. One model in this category is Fusing [43].

Fusing

Fusing (2022) [43] combines global and local features using a two-branch model. A patch selection module identifies regions likely to contain forensic clues, and an attention-based fusion module integrates these with global features for classification. This architecture improves generalization by capturing both large-scale structure and fine-grained artifacts.

2.2.4 Foundation Model-Based Methods

GenAI models can also introduce **semantic inconsistencies that can be detected using Foundation Models**, such as Contrastive Language-Image Pretraining (CLIP) [44]. CLIP is a multimodal model that learns to align images and text in a shared embedding space [44]. By comparing these embeddings, such models can detect semantic anomalies or deviations typical of synthetic content. Figure 2.11 shows a diagram of how Foundational Models can be used for SID. Two CLIP-based methods were selected, UnivFD [45] and RINE [46].

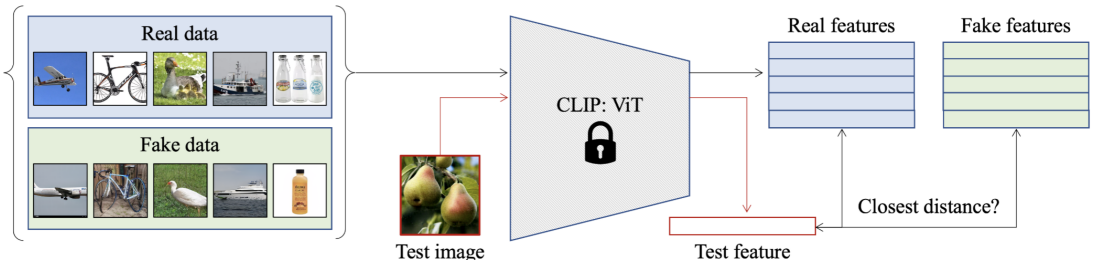


Figure 2.11: Foundation Models can be used to extract a large amount features from images, which contain semantic information. As synthetic images and authentic images generally have different distributions of these features, a model can be trained to distinguish between the two. Image reproduced from Utkarsh et al. (2023) [45]

UnivFD

UnivFD (2023) [45] detects synthetic images by using the feature space of a large pretrained CLIP:ViT [47, 44] model. UnivFD uses nearest neighbor matching or a simple linear classifier on the CLIP embeddings to distinguish real and fake images. It thus compares how similar the tested image is to the known real and fake images. This approach avoids overfitting to specific generative artifacts and achieves strong generalization to unseen models, including diffusion and autoregressive generators.

RINE

RINE (2025) [46] enhances CLIP-based detection by using Representations from Intermediate Encoder-blocks of CLIP, rather than relying solely on the final semantic layer. These intermediate representations capture low-level visual details useful for detecting synthetic traces. The extracted information is combined and projected to a forgery aware vector space using a Trainable Importance Estimator to assign weights to each layer's contribution. This enables the model to form a robust, forgery-aware representation and significantly improves generalization across generative models with minimal training.

	Model	Year	Category	Core Detection Strategy
A	CNNDetect [30]	2020	Spectrum-Based	CNN upsampling artifacts in frequency domain
	DIMD [28]	2023	Spectrum-Based	Spectral fingerprints from diffusion noise
B	GramNet [40]	2020	Texture-Based	Global texture via Gram matrices
	PatchCraft [41]	2024	Local Artifact-Based	Patch-wise texture contrast and filtering
	NPR [42]	2024	Local Artifact-Based	Neighboring pixel correlation patterns
C	Fusing [43]	2022	Hybrid Feature-Based	Global-local feature fusion with attention
D	UnivFD [45]	2023	CLIP-Based	Distance in CLIP embedding space
	RINE [46]	2025	CLIP-Based	Intermediate CLIP layers fused with TIE

Table 2.2: Overview of synthetic image detection models and their detection strategies. The models can be split into four groups, depending on their underlying strategy. In order: spectrum-based (A, Section 2.2.1), local artifact and texture-based (B, Section 2.2.2), hybrid feature-based (C, Section 2.2.3), and foundation model-based (D, Section 2.2.4).

Summary

Synthetic Image Detection (SID) aims to **distinguish between authentic** and **synthetically generated images**. Modern generative models, such as GANs and diffusion models, introduce subtle artifacts or inconsistencies into generated images, which can be exploited for detection. Based on their underlying approach, the discussed SID models can be categorized into **four groups**: spectrum-based methods, local artifact and texture-based methods, hybrid feature-based methods, and foundation model-based methods. An overview of the selected models is shown in Table 2.2.
As these methods are **based on different principles**, they exhibit different strengths and vulnerabilities. Chapter 3 analyzes the potential impact of forensic distractions on these models, and shows that most models are fully robust against these distractions.

3

Motivation: The Impact of Forensic Distractions

This chapter studies the impact of forensic distractions on FL and SID models. This has never before been formally studied. *Forensic distractions* are defined as benign visual additions, such as captions, watermarks, and logos, that can unintentionally (or intentionally) divert the models' attention away from actual forgeries. Although these elements are not malicious in nature, they can pose a significant challenge to multimedia forensic systems (see example in Figure 3.1).

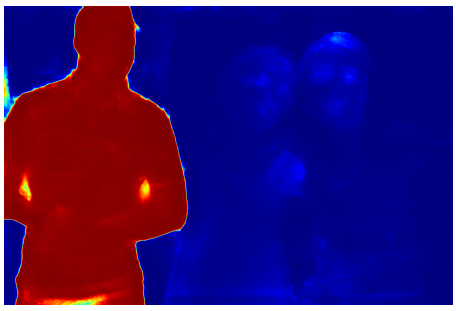
Section 3.1 defines forensic distractions and presents different types commonly found in real-world images. Next, Sections 3.2 and 3.3, respectively, evaluate the performance of several state-of-the-art FL and SID models when exposed to these distractions, providing empirical answers to Research Question 1. Finally, Section 3.4 discusses the findings and motivates for the need of distraction-aware image forensics, which is be the focus of Chapter 4.



(a) Manipulated image, with distraction in upper-right corner



(b) Ground truth mask



(c) Prediction by TruFor *without* distraction, F1* score: 0.9616



(d) Prediction by TruFor *with* distraction, F1* score: 0.0061

Figure 3.1: Example of the impact of distractions on the performance of TruFor [8]. F1* is the adapted F1 score as defined in Section 3.2.1. After a distraction is added, TruFor becomes unable to detect the manipulated region.

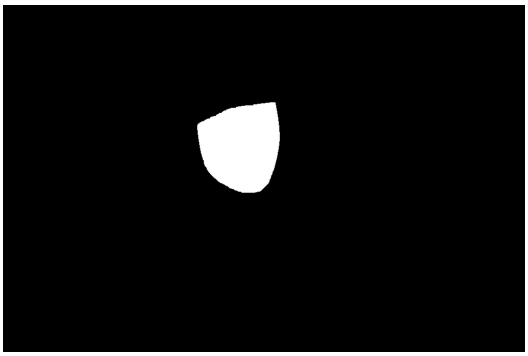
3.1 What are Forensic Distractions

As digital images circulate across online platforms, they are often **intentionally or unintentionally modified with additional visual elements**. For example, users or platforms may intentionally add captions, watermarks, logos, or branding overlays to images. While, these modifications are typically non-malicious and serve communicative or commercial purposes, they pose a unique challenge to multimedia forensic systems. Distraction can also be added unintentionally, for example when taking screenshots. When people take screenshots they often introduce other residual elements, such as the status bar on a phone, the taskbar on a computer, or text overlays from social media platforms (e.g., usernames, descriptions, or platform logos like TikTok).

Although these elements do not semantically modify the underlying visual content or introduce falsified information themselves, they are still technically *manipulations* in the eyes of a forensic model. Since they are an easy target, and probably the most recent manipulation the image underwent, **some FL models mistakenly prioritize these distractions**, flagging them as forged while failing to detect actual malicious alterations elsewhere in the image. Thus, leading to significant reductions in their utility in real-world scenarios. We call these elements *forensic distractions*, and define them as benign modifications that may unintentionally disrupt the performance of forgery detection models.

The **forensic models** analyzed in this thesis **do not immediately become distracted by the presence of distractions**. Simply overlaying a distraction on a PNG image does not disturb these models. FL models, for example, tend to simply detect both the distraction and the forgery independently. However, the **problem arises when lossy compression** (e.g., JPEG compression) **is performed** on the distracted image. The artifacts of the distraction are spread over the entire image, hiding the forensic traces of the forgery. As images shared on social media platforms or uploaded on blog and news websites generally undergo a lossy compression step, the impact of forensic distractions becomes a significant concern for the reliability of forgery detection models in real-world applications.

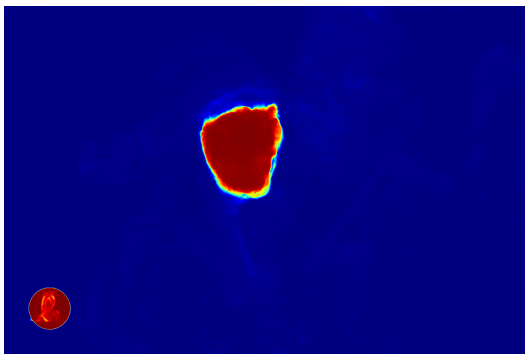
Figure 3.2 shows an example of an image with a distraction and its prediction performed by the TruFor model [8]. Before performing any compression on the image, both the manipulation and the distraction are detected. However, after applying JPEG compression at quality level 95 (Q95), the model starts to fail at detecting the forgery. Increasing the compression further leads to even worse results.



(a) Ground truth mask



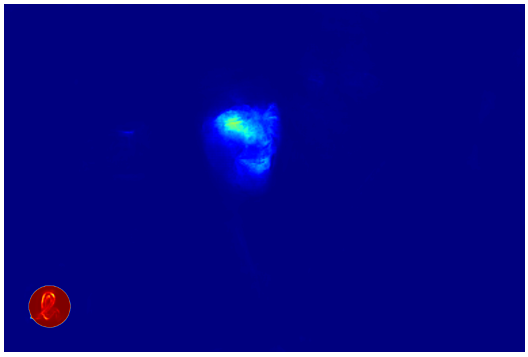
(b) Original image without compression



(c) Prediction of uncompressed image, F1* score: 0.8670



(d) Original image with JPEG Q95



(e) Prediction with JPEG Q95, F1* score: 0.2273



(f) Original image with JPEG Q90



(g) Prediction with JPEG Q90, F1* score: 0.0086

Figure 3.2: Impact of JPEG compression on TruFor's [8] ability to detect forgeries in the presence of a distraction. F1* is the adapted F1 score as defined in Section 3.2.1. Without any compression, TruFor detects both the manipulation and the distraction, achieving a high F1* score. By applying JPEG compression with Q95, TruFor starts losing focus on the distraction, resulting in a decrease in the F1* score. By increasing the compression even further, TruFor becomes fully unable to detect the manipulation.

3.2 The Impact on Forgery Localization

To evaluate the extent to which forensic distractions affect the performance of FL models, an experiment was conducted on a selection of state-of-the-art data-driven deep learning methods and classic model-based methods. This analysis aims to identify which models are particularly sensitive to distractions and to quantify how these elements interfere with the accurate localization of manipulated regions.

Subsection 3.2.1 describes the experimental setup, including tested models, dataset selection, distractions, and evaluation criteria. The results of this analysis are discussed in Subsection 3.2.2.

3.2.1 Experimental setup

Forensic methods The effect of distractions was studied on seven Forgery Localization methods. An explanation of each method can be found in Chapter 2.1. The methods are divided into two categories based on their underlying approach.

The first category consists of **model-based methods**, including DCT [11], ADQ1 [12], BLK [13], and CAGI [14]. These methods exploit handcrafted statistical features or forensic traces, introduced in the image creation and compression process. For example, inconsistencies in DCT coefficient histograms, blocking artifacts, or JPEG grid alignments. The implementations used for these methods are available at `https://github.com/EldritchJS/pyIFD` [48]

The second category are the **data-driven deep learning methods**, which learn to detect and localize forgeries directly from image data. These methods typically rely on CNNs and Vision Transformers to extract and process complex features related to both visual content and forensic traces. Three state-of-the-art deep learning models were analyzed, being CAT-Net [15], TruFor [8] and PSCC-Net [18]. Their implementations can be found in their respective github repositories.

A more in depth explanation of every method can be found in Chapter 2.1 and an overview of all tested FL methods is provided in Table 2.1

Datasets To study the effect of distractions, **four datasets** were used: DSO-1 [49], VIPP [50], OpenForensics [51], and IMD2020 [23]. The DSO-1 and VIPP datasets were used in full, since they contain relatively few images, while 200 images were randomly sampled from both the OpenForensics and IMD2020 datasets, as including all of them in the experiment would have been computationally exhaustive. All images were resized to 1536x1024 pixels. An overview of the used datasets is provided in Table 3.1.

Dataset	Size	Description
DSO-1 [49]	100	Spliced images both indoor and outdoor
VIPP [50]	62	Double compressed images
OpenForensics [51]	18,895	Synthetic face-swapped images
IMD2020 [23]	2,010	Set of in-the-wild manipulated images from the Internet

Table 3.1: Overview of datasets used for forgery localization experiments.

Distractions For this experiment, **ten different logos** were used as distractions. The logos vary in color and shape: some feature sharp peaks, others are more rounded, and some include text. This set of ten logos is representative of those commonly added by social media platforms or news outlets. All logos are sourced from `www.freepik.com` and are 128x128 pixels in size, while the images themselves are 1536x1024 pixels.

All ten used logos are shown in Figure 3.3. To give a better idea of the size of the distractions, an example of their application within a full image is shown in Figure 3.4.

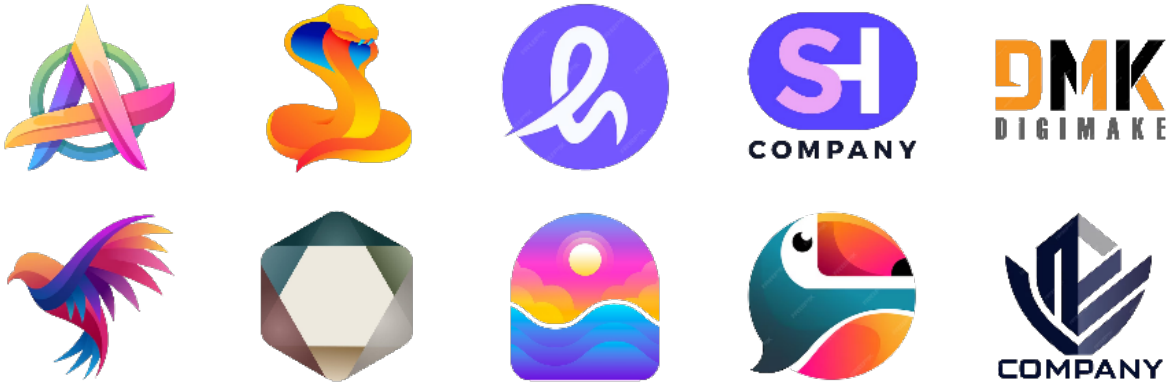


Figure 3.3: To study the effects of forensic distractions on FL, ten logos were used. The logos were selected to represent a variety of shapes, colors, and contents, and are representative of the types of overlays commonly added by social media platforms or news outlets.



Figure 3.4: Example of an image resized to 1536x1024 pixels with a distraction of size 128x128 pixels applied to the upper-left corner. Distraction do not need to be big to have a significant impact on the detection performance of the models.

For each image, a distraction and position (upper-left, upper-right, lower-left, lower-right) was randomly selected. The distraction was overlaid onto the image, after which the image was lossily compressed using JPEG at quality level 95 (Q95). An overview of the **distracted image creation process** is shown in Figure 3.5. As explained in Section 3.1, the JPEG compression step is needed for the distraction to have an impact on the detection performance of the models. The quality level was chosen high to keep the degradation minimal. As any compression reduces the performance of forensic models to a certain extent, adding too much compression can lead to low localization scores, even on non-distracted images. This would make it difficult to assess the impact of adding distractions. The addition of the compression does not reduce the generalizability of the results, as images shared online generally undergo lossy compression.

Detection score To evaluate the impact of distractions on the forensic methods, the F1 score was calculated between the predictions (P) and ground truth masks (G).

The **F1 score** (3.1) was chosen **instead of accuracy** (3.2), as it better reflects the model's performance. Manipulated regions are typically small compared to the entire image, meaning there are far more negative (unaltered) pixels than positive (altered) ones. As a result, even a blank prediction consisting only of negatives could yield a deceptively high accuracy, since the metric would be dominated by true negatives.

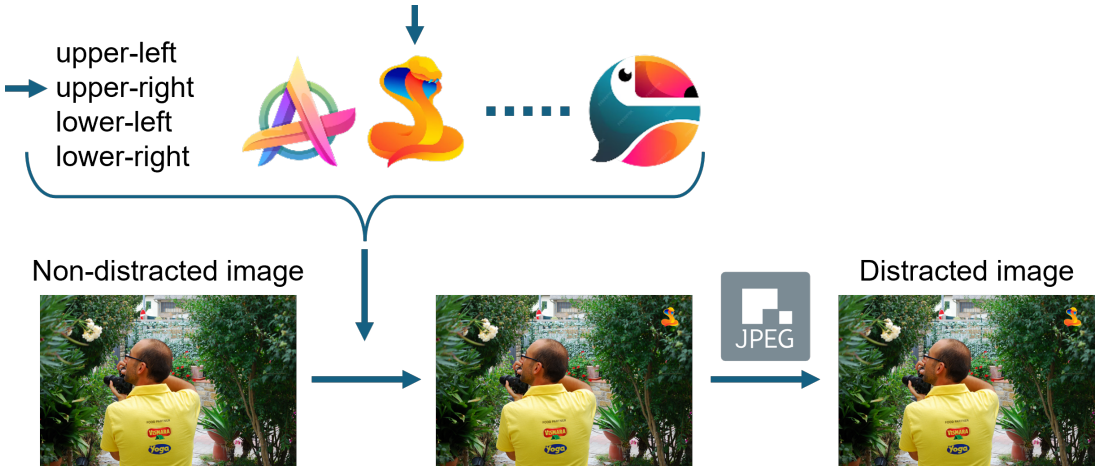


Figure 3.5: Overview of the distraction creation process used in the experiments. A distraction and position (e.g., upper-right) are randomly selected. The distraction is overlaid onto the image, after which the image is lossily compressed using JPEG Q95. The compression step is needed for the distraction to have an impact on the detection performance of the models.

$$\text{F1 Score(P, G)} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \quad (3.1) \qquad \text{Accuracy(P, G)} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (3.2)$$

The goal of this analysis is to measure how well each method continues to detect the actual forgeries after a distraction has been added. Since the distraction itself is not a forgery, methods should neither be rewarded nor punished for detecting it. Instead, the **evaluation focuses solely on the correct detection of the underlying manipulation**.

To ensure this, an adapted version of the F1 score was used. The **adapted F1 score** (F1* score) only considers the pixels that are not part of the distraction. For this calculation a binary mask is created, indicating the pixels that belong to the distraction. The F1* score was computed as follows, with P representing the model prediction, G the ground truth mask and M the created distraction mask, all having values in the range $[0, 1]$:

$$\text{F1* score(P, G, M)} = \frac{2\text{TP[M=0]}}{2\text{TP[M=0]} + \text{FP[M=0]} + \text{FN[M=0]}} \quad (3.3)$$

with

$$\text{TP[M=0]} = \sum_{i=0}^{W}\sum_{j=0}^{H}((P[i,j] \cdot G[i,j]) \cdot (1 - M[i,j])),$$

$$\text{FP[M=0]} = \sum_{i=0}^{W}\sum_{j=0}^{H}((P[i,j] \cdot (1 - G[i,j])) \cdot (1 - M[i,j])),$$

$$\text{TN[M=0]} = \sum_{i=0}^{W}\sum_{j=0}^{H}(((1 - P[i,j]) \cdot (1 - G[i,j])) \cdot (1 - M[i,j])),$$

$$\text{FN[M=0]} = \sum_{i=0}^{W}\sum_{j=0}^{H}(((1 - P[i,j]) \cdot G[i,j]) \cdot (1 - M[i,j])).$$

3.2.2 Results

The results of the experiment are shown in Table 3.2. The table shows the mean localization score of every model both before and after a distraction was added, and the corresponding decrease, on the different datasets. The table also shows the average over all tested datasets.

	Model	DSO-1			VIPP			OpenForensics		
		Original	Distracted	Decrease	Original	Distracted	Decrease	Original	Distracted	Decrease
Model-based	DCT [11]	0.23122	**0.23149**	-0.00026	0.18852	0.18696	0.00156	0.11154	0.11099	0.00054
	ADQ1 [12]	0.22461	0.22355	0.00106	0.17930	0.17885	0.00045	0.09869	0.09824	0.00045
	BLK [13]	0.18553	0.18541	0.00011	0.14365	0.14194	0.00172	0.07594	0.07517	0.00077
	CAGI [14]	0.19477	0.18995	0.00482	0.18724	**0.18925**	-0.00201	0.06512	0.06532	-0.00020
Deep learning	TruFor [8]	**0.43493**	0.16117	0.27376	**0.32978**	0.02787	0.30191	**0.54804**	0.22184	0.32620
	PSCC-Net [18]	0.13760	0.08652	0.05108	0.16504	0.13932	0.02572	0.11167	0.08241	0.02926
	CAT-Net [15]	0.06668	0.05896	0.00772	0.12630	0.11566	0.01064	0.28374	**0.27457**	0.00917

(a) DSO-1 [49], VIPP [50] and OpenForensics [51] datasets

	Model	IMD2020			Average			
		Original	Distracted	Decrease	Original	Distracted	Decrease	% Decrease
Model-based	DCT [11]	0.14660	0.14284	0.00376	0.16947	0.16807	0.00140	0.83%
	ADQ1 [12]	0.14161	0.13838	0.00323	0.16105	0.15976	0.00130	0.80%
	BLK [13]	0.12223	0.12023	0.00199	0.13184	0.13069	0.00115	0.87%
	CAGI [14]	0.11983	0.11765	0.00218	0.14174	0.14054	0.00120	0.84%
Deep learning	TruFor [8]	**0.55829***	0.42579*	0.13250	**0.46776**	0.20917	0.25859	**55.28%**
	PSCC-Net [18]	0.07405	0.05962	0.01443	0.12209	0.09197	0.03012	24.67%
	CAT-Net [15]	0.52894*	**0.51360***	0.01534	0.25141	**0.24070**	0.01072	4.26%

(b) IMD2020 [23] dataset and average across all datasets

Table 3.2: Localization scores (F1* score (3.3)) of the tested FL models on the different datasets before and after adding distractions, and the average across all datasets. The best score for each dataset is highlighted in **bold** and the second best score is underlined. The percentage decrease is shown in parentheses and changes of less than 1% are shown in light gray.
Before adding distractions, TruFor outperforms all models on every dataset. After adding distractions, its performance drops significantly (highlighted in **bold** in the last column), only having competitive scores on the OpenForensics and IMD2020 datasets. PSCC-Net is moderately impacted by the addition of distractions, while the model-based methods and CAT-Net show resilience to distractions, with only minor decreases in performance. As TruFor is the highest performing tested model (before distractions), and it is significantly impacted by distractions, this thesis focuses on improving the robustness of TruFor against distractions.
Remark:* TruFor and CAT-Net were trained on the IMD2020 dataset, which explains their high performance and robustness against distractions in this dataset.

Model-Based Methods

In Table 3.2, it can be seen that the model-based methods are highly resilient against distractions. **Forensic distraction do not pose a threat** to them, with an average decrease in performance of only 0.84%. Note that this may be because these methods perform poorly in general, with an average F1* score between 0.13 and 0.17. Percentage- and scatter plots for each individual method and dataset can be found in Appendix B

TruFor

From the results in Table 3.2 we observe that the TruFor model is strongly influenced by the presence of distractions. For each dataset, the F1* score decreases a significant amount. The average score across all datasets decreases from 0.468 to 0.209, being a drop of 55.3%. This implies that adding distractions to an image **significantly disrupts** the working of the TruFor model, leading to degradation of performance and accuracy.

Examples Figure 3.6 shows three examples of manipulated images with their corresponding TruFor predictions with and without distractions. It can be seen that not every image is equally affected by the addition of distractions. The first and second image show a large decrease in performance, while the third image shows only a small decrease.

Scatter plots The mean localization score does not tell the full story. The TruFor model is inherently better or worse at detecting certain forgeries. These forgeries can skew the average score. Therefore, it is important to look at the individual scores as well.

How to interpret scatter plots
Each dot in the scatter plot represents a single image. The y-axis indicates the localization score before a distraction was applied. The x-axis shows the score after the distraction was applied. The dashed-red line shows y=x or before equals after, meaning the distraction did not influence the detection. If a point lies to the left of the red line, the detection score for that image decreased by adding the distraction.

Example:
Image A's localization score drops from 0.8 to 0.2 after adding the distraction, indicating that the distraction had a significant impact on the detection process. Image B's score remains 0.6, showing the distraction did not influence the detection process.

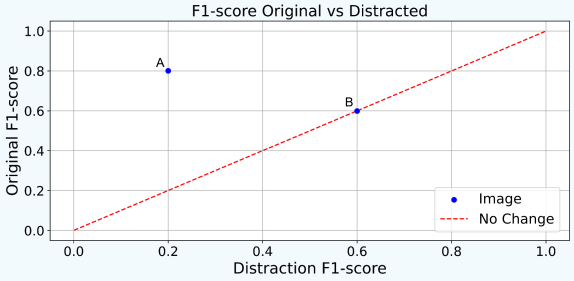


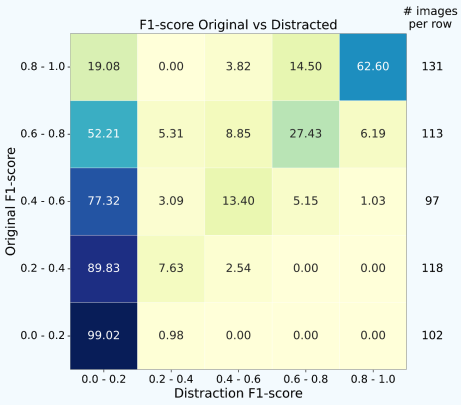
Figure 3.9 shows the scatter plots of the localization scores of all tested images for each dataset, and Figure 3.10a shows the aggregated results. These scatter plots do not only show that the performance of TruFor decreases, which could already be seen from the average scores, but it also shows the relationship between the score before and after a distraction is added. As could be expected, forgeries that are difficult for TruFor to detect become almost impossible to detect after the forensic distractions are added. On the other hand, **forgeries that TruFor can detect very easily, often retain a high localization scores** even after a distraction is added. The scatter plots show that the F1* scores, in general, decrease by adding distractions.

Percentage plots All tested images for each datasets were aggregated and bucketed into percentage plots. Figure 3.7 shows the results for each dataset and figure 3.8a shows the aggregated results of all 4 datasets. It can be seen that 19.08% of all images that were originally detected with a score of 0.8 or higher, i.e. easily detectable forgeries (by TruFor), are now detected with at most a score of 0.2, i.e.

they become undetectable. In general, it can be calculated that **of all images with a score of 0.6 or higher**, i.e. forgeries get correctly identified, **36.89% drops to a score of 0.4 or lower**, i.e. forgeries are *not* correctly identified.

How to interpret percentage plots
Percentage plots are a bucketed version of the scatter plots. Each row in the plot shows a bucket of images with a similar localization score, before any distraction are added. The columns of every row add up to 100% and show what percentage of these images after a distraction is added have a localization score, part of that bucket. The total number of images in each row is shown on the right.

Example:
131 of the 561 (=131+113+97+118+102) images had a localization score of 0.8 or higher before any distractions were added. After adding distractions, 19.08% (or 25) of these images have a localization score of 0.2 or lower. While 62.20% (or 82) of these images kept a localization score of 0.6 or higher.



Datasets In general, all 4 dataset show the same, significant, impact of forensic distractions on the TruFor model. The DSO-1 and OpenForensics datasets show a similar score distribution. The VIPP dataset only contains 61 images, on which TruFor seems to perform quit poorly, leading to a lower localization score of 0.330. Lastly, TruFor seems to perform the best on the IMD2020 dataset, with a localization score of 0.558 on the original images, and 0.426 on the distracted images. This is not surprising, as the IMD2020 dataset is part of TruFor's training data. Percentage- and scatter plots for each individual dataset can be found in Appendix B.

In summary, the TruFor model is **highly sensitive to the presence of forensic distractions**, leading to a decrease in localization performance and accuracy. Adding distractions to forged images **can thus be used as an attack strategy** against one of the best state-of-the-art multimedia forensic models. Therefore, it is important to study the possibility of distraction-aware forgery localization, which is the focus of Chapter 4.

PSCC-Net

The results in Table 3.2 show that PSCC-Net **performs quite poorly on all datasets**, with an average localization score of 0.122. This poor performance could be due to the applied compression, which significantly increases the difficulty of detecting forgeries. When adding distractions, the localization score drops to 0.0920, for a decrease of 24.7%. PSCC-Net is thus also **sensitive to distractions**, lowering the already low detection scores even further.

Examples Examples of manipulated images with their corresponding PSCC-Net predictions with and without distractions can be found in the Appendix B

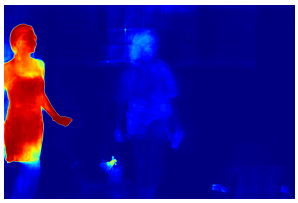
Scatter- & percentage plot Figure 3.10b and Figure 3.8b, respectively, show the scatter plot and the percentage plot of PSCC-Net aggregated over all four datasets. Both plots show that PSCC-Net is affected by the presence of distractions, but to a lesser extent than TruFor.



(a) Manipulated image (1), with distraction in lower-right corner



(b) Ground truth mask (1)



(c) Prediction *without* distraction (1), F1* score: 0.7696



(d) Prediction *with* distraction (1), F1* score: 0.0028



(e) Manipulated image (2), with distraction in lower-left corner



(f) Ground truth mask (2)



(g) Prediction *without* distraction (2), F1* score: 0.7850



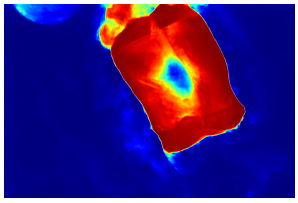
(h) Prediction *with* distraction (2), F1* score: 0.1305



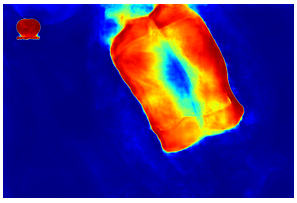
(i) Manipulated image (3), with distraction in upper-left corner



(j) Ground truth mask (3)



(k) Prediction *without* distraction (3), F1* score: 0.8654



(l) Prediction *with* distraction (3), F1* score: 0.7889

Figure 3.6: Examples of manipulated images with corresponding TruFor [8] predictions and F1* scores (3.3), without (3th column) and with (4th column) distractions. It can be seen that the impact of distractions on TruFor's performance varies across images.

Datasets PSCC-Net performs similarly on all datasets, with the worst performance on the IMD2020 dataset. This is likely due to the fact that the IMD2020 dataset contains *in-the-wild* manipulated images, and the IMD2020 dataset is *not* part of PSCC-Net's training data. Percentage- and scatter plots for each individual dataset can be found in Appendix B.

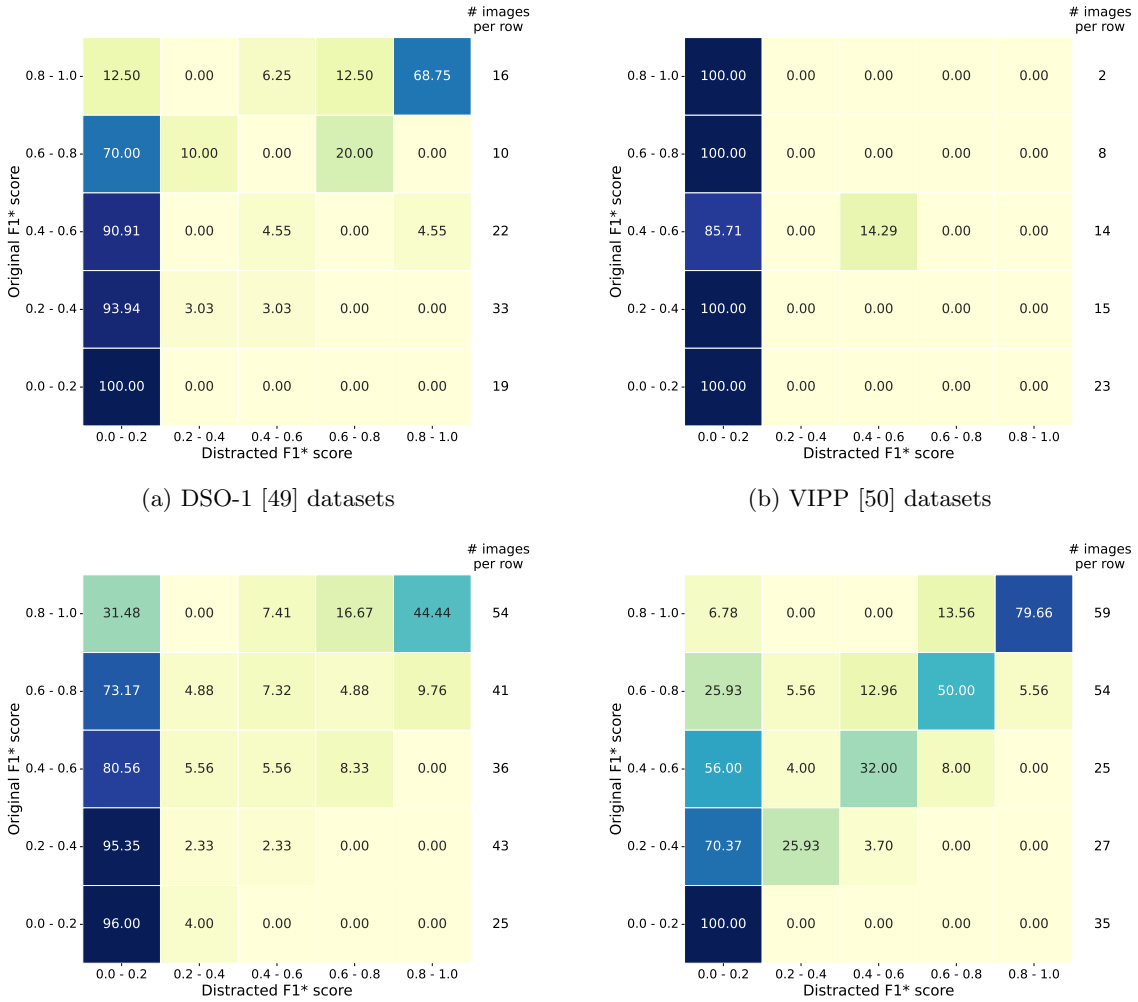
CAT-Net

Table 3.2 shows that CAT-Net is **quite robust to the presence of distractions**. The F1* score decreases from a mean of 0.251 to 0.241, being a decrease of only 4.26%. This is a non-negligible impact, but compared to 55.3% for the TruFor model and 24.7% for PSCC-Net, it is quite low.

Examples Examples of manipulated images with their corresponding CAT-Net predictions with and without distractions can be found in the Appendix B

Scatter plot In Figure 3.10c, it can be observed that most points lie close to the dashed-red line, indicating that the **addition of distractions has minimal impact** on CAT-Net's ability to localize forgeries. This robustness is particularly evident when looking at images where CAT-Net struggles to detect the forgery. These are the points at the bottom of the scatter plot. It is clear that the presence of distractions does not significantly alter the performance, meaning that even though the CAT-Net model already had trouble detecting the forgery, it does not struggle more after a distraction is added. This suggests that CAT-Net's architecture is inherently more resilient to the effects of forensic distractions compared to TruFor and PSCC-Net.

Datasets In general, CAT-Net performs worse than TruFor on every dataset, as can be seen in Table 3.2. Only for the IMD2020 dataset, CAT-Net performs similarly to TruFor, with a localization score of 0.529. This is expected as IMD2020 is also part of CAT-Net's training data. As CAT-Net does perform worse than TruFor, it is not possible to simply stop using TruFor and use CAT-Net as a distraction resilient replacement. Percentage- and scatter plots for each individual dataset can be found in Appendix B.



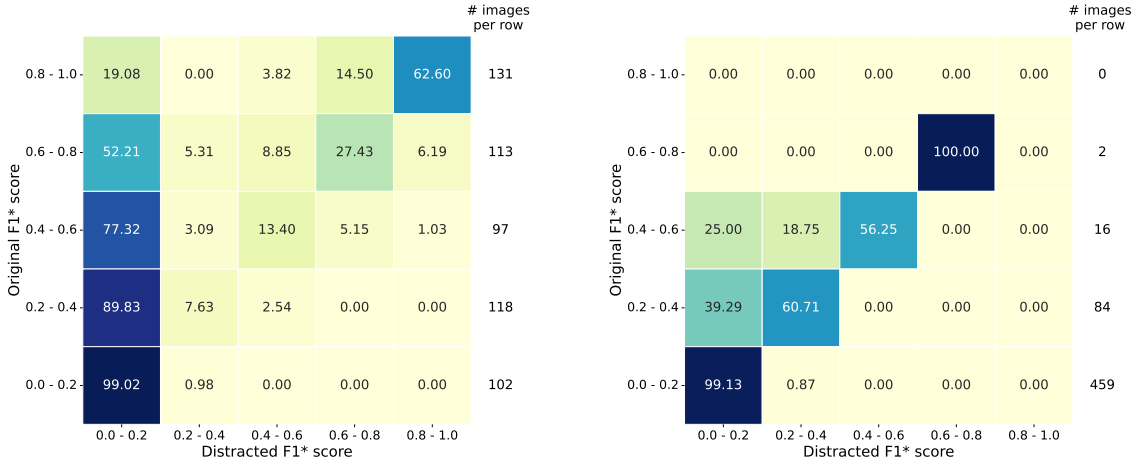
(a) DSO-1 [49] datasets

(b) VIPP [50] datasets

(c) OpenForensics [51] datasets

(d) IMD2020 [23] datasets

Figure 3.7: Percentage plots of the localization scores of TruFor [8] on four datasets (DSO-1, VIPP, OpenForensics, and IMD2020). The plots show that TruFor is significantly affected by the addition of forensic distractions. The IMD2020 dataset (d) seems to show more robust results. This is likely because IMD2020 is part of TruFor's training set.



(a) TruFor [8] (b) PSCC-Net [18]

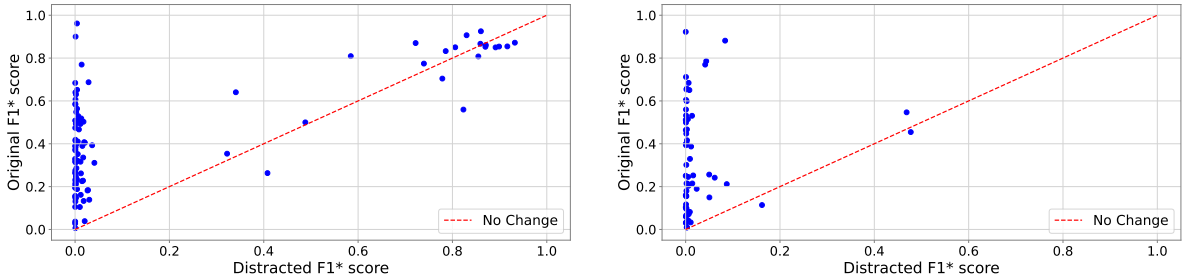
Figure 3.8: Percentage plots of the localization scores of TruFor and PSCC-Net aggregated over all four datasets (DSO-1, VIPP, OpenForensics, and IMD2020). For TruFor (a) It can be calculated that 36.89% of the images that originally had a localization score of 0.6 or higher, now have a localization score of 0.4 or lower.

Summary

In summary, the experimental results show that forensic distractions can significantly impact the performance of deep learning-based forgery localization models, while having **little to no effect on model-based methods**.
Among the deep learning models, **TruFor** is **highly sensitive** to distractions, with an average performance drop of 55.3%, severely reducing its localization accuracy. **PSCC-Net** performs poorly overall and is **moderately affected** by distractions, with a 24.7% decrease. **CAT-Net** exhibits much **greater resilience**, with only a 4.26% decrease, though it generally performs worse than TruFor on undisturbed images. These findings highlight the vulnerability of current state-of-the-art FL models to forensic distractions and motivate the need for distraction-aware localization strategies, which is the focus of Chapter 4.

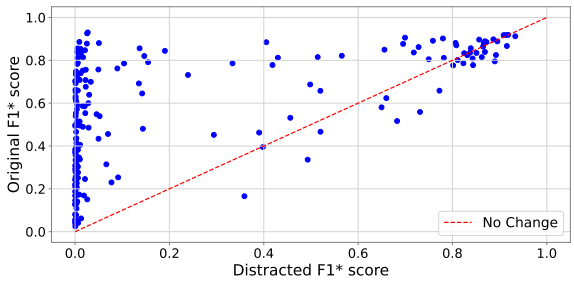
3.2.3 Effectiveness of Distraction Types

A small supplementary study was conducted to better understand which types of distractions are most effective at distracting FL models, particularly TruFor [8]. The study examined both textual and graphical distractions. It analyzed how factors such as text size, font type, and font weight influence distraction potency. It also explored the effects of logo color and shape, as well as the impact of distraction placement. Full results of this experiment are provided in Appendix A. Overall, the findings suggest that larger and bolder textual distractions tend to be more disruptive (up to a certain size threshold of about 10% of the image height). Some graphical distractions are more disruptive than others, although no clear patterns were found. The placement of distractions also plays a role, with upper-left and lower-right placements being the most disruptive. These insights may be helpful for future development of distraction-resilient forensic models.

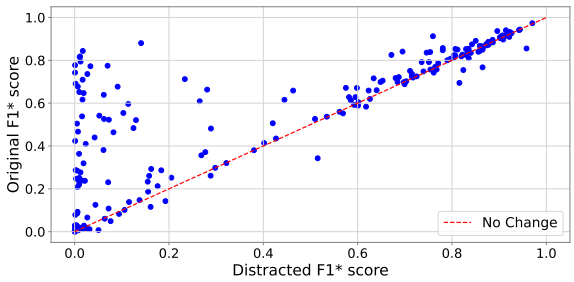


(a) DSO-1 [49] datasets

(b) VIPP [50] datasets

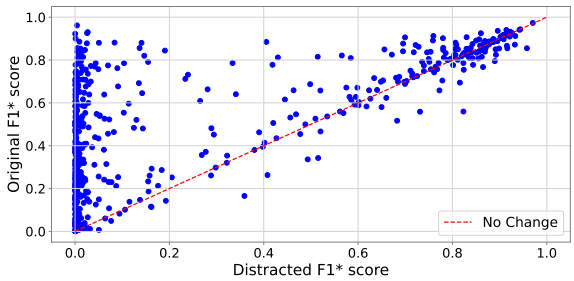


(c) OpenForensics [51] datasets

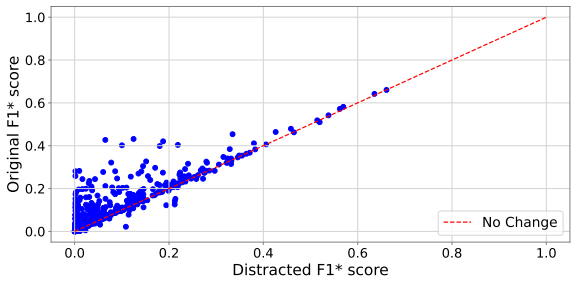


(d) IMD2020 [23] datasets

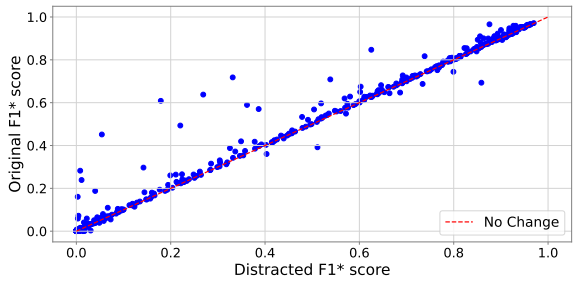
Figure 3.9: Scatter plots of localization scores (F1* score (3.3)) of TruFor [8] on four datasets (DSO-1, VIPP, OpenForensics, and IMD2020)



(a) TruFor [8]



(b) PSCC-Net [18]



(c) CAT-Net [15]

Figure 3.10: Scatter plots of the localization scores (F1* score (3.3)) of TruFor, CAT-Net and PSCC-Net aggregated over all four datasets (DSO-1, VIPP, OpenForensics, and IMD2020). Not all models are affected equally by the forensic distractions. TruFor (a) is significantly affected, causing a drop in localization scores. PSCC-Net (b) is also affected, though this is less pronounced than for TruFor, as the general performance of PSCC-Net is already worse. CAT-Net (c) shows a robust performance.

3.3 The Impact on Synthetic Detection

To assess the impact of forensic distractions on the performance of SID models, a separate experiment was conducted using a range of representative detection methods. This analysis aims to determine whether such distractions interfere with detection performance of synthetic content classification.

Subsection 3.3.1 describes the experimental setup, including tested models, datasets selection, distractions, and evaluation. The findings of this experiment are presented in Subsection 3.3.2.

3.3.1 Experimental setup

Detection models The effect of distractions was studied on a set of eight Synthetic Image Detection models. The experiment was conducted **using the SIDBench framework** [29], a benchmarking suite designed to ease the evaluation of state-of-the-art SID models.

The selected models were **divided in four categories based on their underlying approach**: The spectrum-based methods, CNNDetect [30] and DIMD [28], detect synthetic content by identifying frequency-domain fingerprints left by generative models. The local artifact and texture-based methods, GramNet [40], PatchCraft [41], and NPR [42], analyze spatial inconsistencies or texture-level traces in the image. Fusing [43], categorized as a hybrid method, combines global semantic and local artifact features to improve robustness. Finally, two Foundation model-based methods, UnivFD [45] and RINE [46], detect semantic mismatches between image content and high-level multimodal representations using pretrained foundation model embeddings - in this case CLIP [44] embeddings.

A more in depth explanation of every method can be found in Chapter 2.2 and an overview is provided in Table 2.2

Datasets Four datasets were used in this experiment: **two synthetic** datasets, TGIF [52] and Synthbuster [53], and **two authentic** datasets, Fantastic Reality [22] and JPEG RAISE [15].

TGIF contains approximately 37,500 fully regenerated images generated through text-guided inpainting using Stable Diffusion 2 and XL, and Adobe Firefly. Synthbuster contains 8,000 synthetic images generated using various generative models (1,000 each), DALL·E 2 and 3, Stable Diffusion 1.3, 1.4, 2 and XL, Glide, Adobe Firefly, and Midjourney. Fantastic Reality contains both 16,000 authentic and 16,000 synthetic images, during this experiment only the authentic images were used. Finally, JPEG RAISE is a JPEG compressed version of the RAISE dataset [54], containing about 8,000 authentic images.

From both authentic datasets, 800 images were randomly sampled. For the synthetic datasets, 200 images were randomly selected from each of the following sources: TGIF (sd2-fr and sdxl-fr) and Synthbuster (DALL·E 2 and 3, Glide, Adobe Firefly, Midjourney, and sd1.3). In total, 1,600 authentic images and 1,600 synthetic images were used in the experiment.

Dataset	Size	Type	Source Models
TGIF [52]	∼37,500	Inpainting	SD2, SDXL
Synthbuster [53]	9,000	Text-to-image	DALL·E 2 & 3, Glide, Firefly, Midjourney, SD1.3
Fantastic Reality [22]	16,000	Authentic	N/A
JPEG RAISE [15]	∼8,000	Authentic	N/A

Table 3.3: Overview of datasets used for synthetic image detection experiments.

Distractions The **same set of ten distraction logos**, used in the FL experiment in Section 3.2.1, was also used in the SD experiments. These logos, sourced from `www.freepik.com`, vary in color, shape, and text content, and are representative of typical overlays found in social media or news imagery. An overview of the logos is shown in Figure 3.3. The logos were resized for each image to maintain the same relative area as a 128x128 logo on a 1024x1024 image.

For each image, a distraction and position (upper-left, upper-right, lower-left, lower-right) were randomly selected. The logo was overlaid on the image, which was then JPEG-compressed at quality level 95 (Q95). This simulate typical web compression without introducing strong artifacts and without compromising generalizability.

Detection score SID models output a single prediction score per image, representing the probability that the image is synthetic. To evaluate detection performance, the Area Under the Receiver Operating Characteristic Curve (**AUC-ROC**) was used.

AUC was chosen over accuracy or F1* score because it provides a threshold-independent measure of performance. This is particularly important in SID tasks, where thresholds are rarely consistent across models or datasets. AUC better captures the trade-off between false positives and false negatives, especially in datasets with class imbalance.

The aim is to determine whether the presence of a distraction alters the model's ability to correctly classify the image as synthetic or real. Since the distractions themselves do not change the nature of the underlying image, an ideal synthetic image detector should remain unaffected by their presence, producing the same predictions and maintaining a constant AUC score.

3.3.2 Results

The results of the SID experiment are shown in Table 3.4. It shows the average prediction result on each dataset, and the mean AUC score of each model calculated over all datasets. Full results of all methods can be found in Appendix C.

The results show that the tested SID models are **generally robust** to the addition of distractions. Most models maintained similar predictions after distractions were applied, **with exceptions of NPR and PatchCraft**, and to a much smaller extent, Fusing. The average AUC scores are similar for both distracted and non-distracted images, with the exception of NPR. As only NPR and PatchCraft are influenced by the presence of distractions, their results are discussed in more details.

NPR The scatter plots of NPR, shown in Figure 3.12, consists of **multiple distinct clusters**, almost lines. Each cluster is a different dataset. **Adding distraction shifts the predictions in a controlled, but different way** for each dataset. The ROC curves of NPR with and without distractions are shown in Figure 3.11a. The AUC score drops from 0.59 to 0.25, indicating that NPR is sensitive to distractions. However, due to fact that each dataset is affected in a totally different manner, it could be that if other datasets were used, the AUC score would be a lot higher. Further experimentation is needed to confirm this.

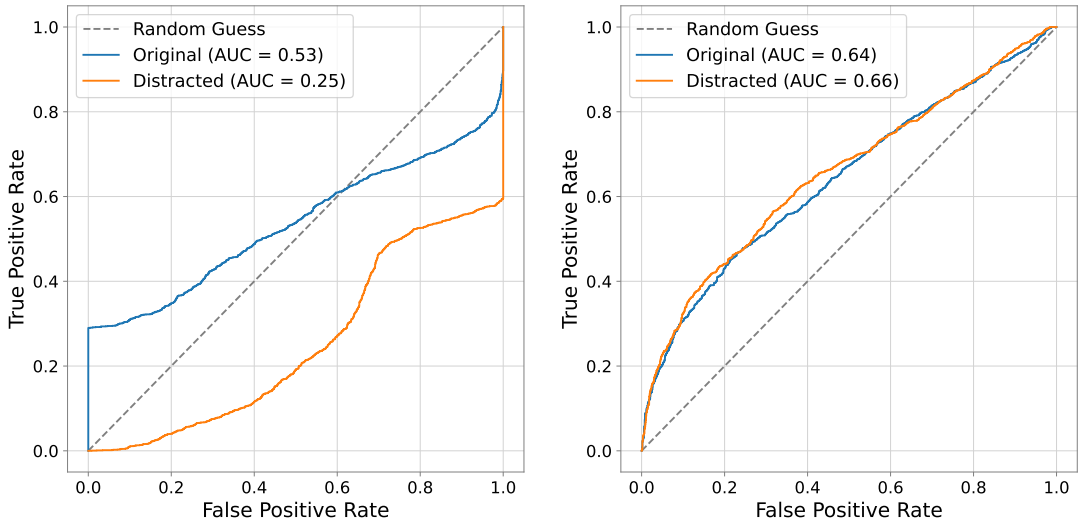
The weird behavior can be explained by how NPR operates. **NPR focuses on the local structural relationships between neighboring pixels**, detecting inconsistencies caused by upsampling artifacts in generative models. The **addition of distractions introduces new local structures** that can disrupt these relationships. Inside a dataset, each image is likely to have similar local structures, leading to a consistent shift in predictions. However, the nature of the distractions and the JPEG compression applied to the images can vary significantly between datasets, resulting in different prediction shifts. Furthermore, the JPEG compression applied to all tested images may partially destroy or blur the subtle neighboring pixel patterns that NPR relies on, further contributing to the model's initially poor performance on both distracted and non-distracted images.

PatchCraft PatchCraft [41], on the other hand, shows a **strong sensitivity to distractions**. This can be observed in the scatter plots shown in Figure 3.13. When distractions are added, the predicted probabilities for both authentic and synthetic images shift in both directions (sometimes higher, sometimes lower), but **without a consistent pattern favoring one class**. As a result, the overall average predicted scores, and consequently the AUC, remain approximately the same.

This behavior aligns with how PatchCraft functions. **PatchCraft analyzes local texture patches**.

The **presence of distractions alters the local texture statistics significantly**, which affects the model's output. However, because the perturbations caused by distractions impact both classes similarly, there is no strong overall effect on the AUC score.

Overall, while most SID models proved resilient to distractions, models that rely heavily on local texture or structural relationships, such as NPR and PatchCraft, are more affected by small localized changes introduced by forensic distractions.



(a) NPR [42]

(b) PatchCraft [41]

Figure 3.11: ROC curves of NPR (a) and PatchCraft (b) over the four datasets (TGIF [52], Synthbuster [53], Fantastic Reality [22], and JPEG RAISE [15]). The area under the curve (AUC) is shown in the legend.

Summary

In summary, the experimental results show that Synthetic Image Detection models are **generally robust** to the addition of forensic distractions. Most models maintained similar detection performance, with only NPR and PatchCraft showing notable sensitivity.
NPR showed **erratic results**, this is likely due to its reliance on pixel-level correlations being disrupted by distractions and the JPEG compression performed in the used workflow for adding distractions. The predictions of **PatchCraft** were **strongly affected** by the distractions. Increasing for certain images, and decreasing for others, but without a consistent pattern favoring one class. This led to an overall similar AUC score.

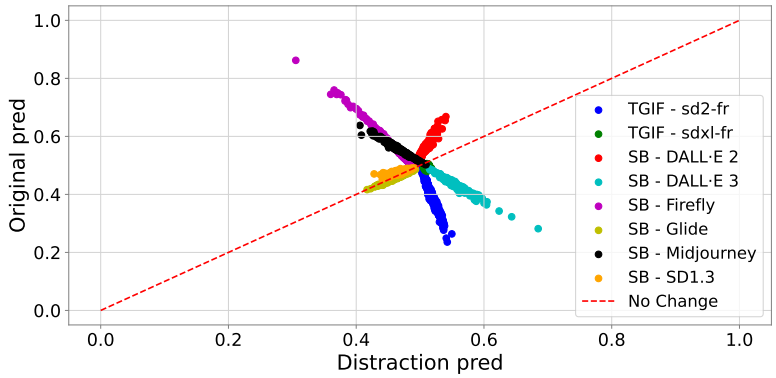
	Model	TGIF - sd2-fr		TGIF - sdxl-fr		SB - DALL·E 2		SB - DALL·E 3		SB - Glide		SB - Firefly	
		Original	Distracted	Original	Distracted	Original	Distracted	Original	Distracted	Original	Distracted	Original	Distracted
A	CNNDetect [30]	0.01059	+0.0	0.00707	+0.0	0.03542	+0.0	0.00029	+0.0	0.13672	+0.0	0.00453	−0.001739
	DIMD [28]	0.93956	+0.0	0.27965	+0.0	0.00923	+0.0	0.00032	+0.0	0.23998	+0.0	0.06313	−0.02636
B	GramNet [40]	0.08085	+0.0	0.08485	+0.0	0.37705	+0.0	0.08684	+0.0	0.24427	+0.0	0.24345	−0.02303
	PatchCraft [41]	0.56368	+0.004916	0.58103	−0.02008	0.31418	−0.06800	0.10377	+0.01153	0.70388	−3.644e-04	0.39932	+0.016985
	NPR [42]	0.40031	+0.1167	0.49735	+0.1383	0.56064	−0.05215	0.43128	+0.1257	0.58270	−0.1329	0.45813	+0.005271
C	Fusing [43]	0.00321	−0.001098	0.00578	−1.544e-05	0.03522	−0.008593	0.00017	−2.885e-05	0.16245	−0.01244	0.01510	−0.006171
D	UnivFD [45]	0.22027	−1.686e-08	0.32846	+0.0	0.56426	+0.0	0.03104	+0.0	0.83886	+0.0	0.14894	+0.01549
	RINE [46]	0.34748	−2.892e-07	0.37732	+0.0	0.24783	+0.0	0.23055	+0.0	0.57628	+0.0	0.79280	−0.04049

(a) TGIF [52] (sd2-fr, sdXL-fr) and Synthbuster [53] (DALL·E 2, DALL·E 3, Glide, Firefly) datasets

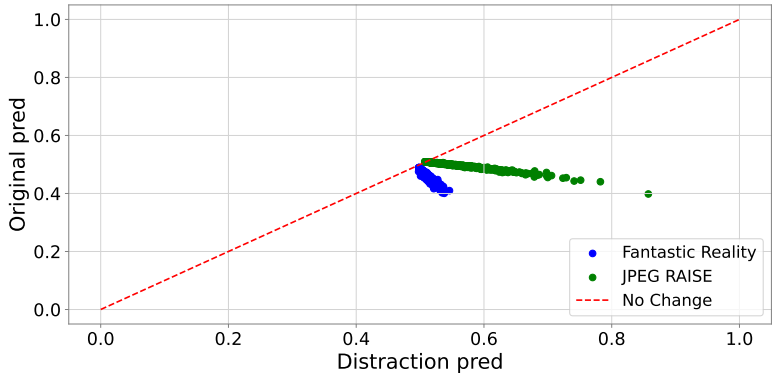
	Model	SB - Midjourney		SB - SD1.3		Fantastic Reality		JPEG RAISE		Mean AUC		
		Original	Distracted	Original	Distracted	Original	Distracted	Original	Distracted	Original	Distracted	Difference
A	CNNDetect [30]	0.00538	+0.0	0.00566	+0.0	0.00335	+6.011e-09	0.04591	+0.0	0.4794	0.4765	−0.0029
	DIMD [28]	0.94876	+0.0	0.99999	+0.0	0.01075	+3.864e-13	0.08113	+0.0	0.7856	0.7810	−0.0046
B	GramNet [40]	0.11722	+0.0	0.35573	+0.0	0.04361	+0.0	0.03864	+0.0	0.7224	0.7220	−0.0004
	PatchCraft [41]	0.12596	−0.001574	0.45765	+0.006321	0.14259	−0.005195	0.32602	+5.650e-04	0.6428	0.6554	+0.0126
	NPR [42]	0.55428	−0.08645	0.47669	−0.009924	0.46081	+0.04907	0.55993	+0.06363	0.5348	0.2518	−0.2830
C	Fusing [43]	0.02877	−0.001722	0.01724	−0.005314	0.00175	−4.291e-05	0.04234	−1.47e-04	0.4910	0.4758	−0.0152
D	UnivFD [45]	0.06544	+0.0	0.22331	+0.0	0.03553	+2.724e-04	0.26570	+0.0	0.6353	0.6410	+0.0057
	RINE [46]	0.40322	+0.0	0.79021	+0.0	0.07258	+4.113e-05	0.32338	+0.0	0.7647	0.7644	−0.0003

(b) Synthbuster [53] (Midjourney, SD1.3), Fantastic Reality [22], and JPEG RAISE [15] datasets, and AUC score calculated over all datasets

Table 3.4: Average prediction of the tested SD models on each dataset, and the mean AUC score of each model calculated over all datasets. (SB = Synthbuster). The SD models are devided into four categories: spectrum-based (A), local artifact and texture-based (B), hybrid (C), and foundation model-based (D).



(a) Synthetic images



(b) Authentic images

Figure 3.12: Scatter plots of predictions of NPR [42] over the four datasets (TGIF [52], Synthbuster [53], Fantastic Reality [22], and JPEG RAISE [15]). The y-axis shows the prediction of the non-distracted images, while the x-axis shows the prediction of the distracted images. Adding distractions introduces new local structures which can disrupt the neighboring pixel relationships that NPR relies on, leading to a shift in predictions. Images inside a set are likely to have similar local structures, resulting in similar changes for each image in the set. Each image set is shown in a different color.

3.4 Conclusion

As images circulate the internet, they are often altered with additional visual elements. People add captions to images, or take screenshots, which introducing other residual elements, such as usernames and descriptions. And platforms add watermarks, logos, or branding overlays to images. All these elements are **benign modifications**, but have the **potential to disrupt the performance of forensic models**. We call these elements *forensic distractions*.

To study the impact of forensic distractions, we conducted two experiments. The first experiment focused on **Forgery Localization models** (see Section 3.2). The results show that while traditional **model-based methods** are mostly **unaffected**, some deep learning-based methods are significantly impacted by distractions. **TruFor** [8] is **highly sensitive**, with a 55.3% drop in performance (F1* score). **PSCC-Net** [18] has an overall weak performance, but still shows **moderate sensitivity**, with a 24.7% decrease in performance. **CAT-Net** [15] is much more **robust**, with a decrease of only 4.26%.

The second experiment targeted **Synthetic Image Detection models** (see Section 3.3). These models were found to be **generally robust** to forensic distractions, maintaining a constant AUC score. Only NPR [42] and PatchCraft [41] exhibited some sensitivity.

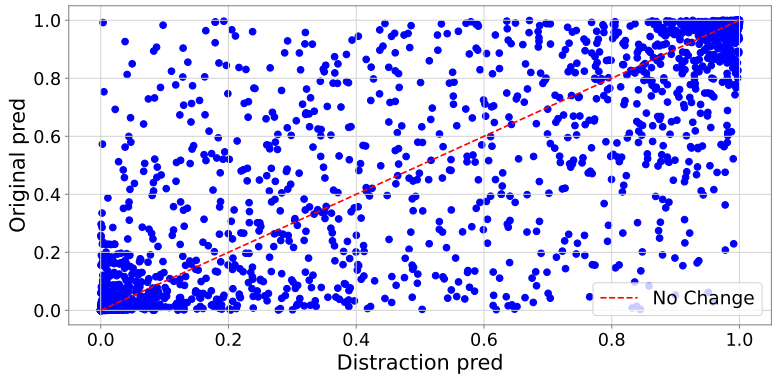
NPR exhibited erratic behavior, likely due to its dependence on pixel-level correlations, which were

disrupted by both the added distractions and the JPEG compression performed during the experiment. PatchCraft's predictions were significantly influenced by the distractions, with scores increasing for some images and decreasing for others, but without a clear trend favoring either class. As a result, the overall AUC score remained largely unchanged.

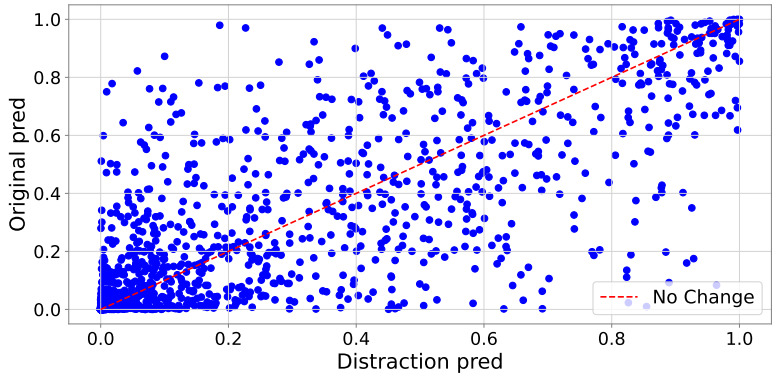
In addition to these core experiments, a supplementary study was conducted to better understand what makes a distraction effective. The full results of this study are presented in Appendix A.

These results show that forensic distractions pose a real **threat to the reliability** of a number of state-of-the-art FL models, particularly TruFor. This highlights the **need for** more robust, **distraction-aware approaches** to improve the reliability and accuracy of existing forensic models.

This need is the focus of Chapter 4, which explores strategies for distraction-aware forgery localization based on the TruFor model.



(a) Synthetic images



(b) Authentic images

Figure 3.13: Scatter plot of predictions of PatchCraft [41] over the four datasets (TGIF [52], Synthbuster [53], Fantastic Reality [22], and JPEG RAISE [15]). The y-axis shows the prediction of the non-distracted images, while the x-axis shows the prediction of the distracted images. PatchCraft works by analyzing local texture patches, and the presence of distractions significantly alters the local texture statistics, leading to wildly different predictions.

4

Distraction-aware Forgery Localization

In Chapter 3, it was shown that forensic distractions pose a significant threat to the effectiveness of the TruFor [8] forensic model. Since TruFor is among the most advanced state-of-the-art models, this threat must be mitigated.

This chapter aims to mitigate the problem of forensic distractions by introducing the concept of *distraction-aware detection*. First, potential solutions are explored, which are discussed in Section 4.1. Next, two solutions are selected and implemented, which aim to enhance TruFor's robustness by making it distraction-aware. The *adapter* solution is discussed in Section 4.3, and the *masked-convolution* solution is discussed in Section 4.4. The created solutions are then evaluated in Chapter 5.

4.1 Possible Solutions

This section explores various potential solutions to address the challenges posed by distractions in FL. As these solutions build on the original TruFor model, it is important to understand its architecture. A full explanation is provided in Section 2.1.3.

Five possible solutions are presented: Fine-tuning (Section 4.1.1), Post-Processing (Section 4.1.2), adding an extra input and retraining (Section 4.1.3), using an adapter (Section 4.1.4), and masked convolutions (Section 4.1.5). Each solution is analyzed in detail, providing an overview of its approach, the rationale behind its plausibility, and a discussion of its advantages and disadvantages. Table 4.1 summarizes the solutions, highlighting their strengths and weaknesses.

4.1.1 Fine-tuning

A potential approach is fine-tuning the existing TruFor model. Fine-tuning a pretrained Machine Learning (ML) model is a common technique for **improving** its **performance on a specific task** by continuing training on a task-specific dataset. In this case, it is possible to adapt TruFor's training data to include distractions such as logos, captions, and watermarks placed in various positions. The associated masks would need to be updated slightly to reflect any overlap between the distractions and the forged regions.

It has been shown repeatedly in the literature that fine-tuning models for domain-specific applications can significantly improve performance with relatively little additional training. Fine-tuning has been **successfully applied in various domains**, including Natural Language Processing (NLP) [55] and computer vision [56]. Therefore, it is reasonable to expect that fine-tuning the TruFor model on a dataset containing distractions will improve its resilience to *these* distractions.

While fine-tuning looks promising, it has certain drawbacks that need to be taken into account. First, fine-tuning typically improves performance on a specific, narrower task. This has the potential of **reducing** the model's **generalization** to other scenarios. In this context, a model fine-tuned on distracted images might become more robust to distractions but **worse at detecting forgeries in general**. This trade-off may be acceptable if the training data is diverse enough and well-balanced between distracted and non-distracted examples. Ideally, though, the "distraction-aware component" would be modular, allowing it to be enabled or disabled as needed to preserve generalization when no distractions are present.

Second, there is a risk that the model only learns to ignore the specific distractions it has seen during training. The model would learn, "If I see this logo, ignore it." The model would not learn to detect a distraction and consequently ignore it. It may thus **not generalize to new, unseen distractions**. Instead, it would be preferred if the model learned to detect and ignore distractions as a concept rather than memorizing specific instances.

Summary: Fine-tuning

Fine-tuning the TruFor model on a dataset containing distractions can improve its resilience to *these specific* distractions. However, the model **may lose generalizability** to other scenarios and **may not generalize** to new, unseen distractions.

4.1.2 Post-Processing

A second possible solution is to add a post-processing step to the TruFor pipeline. The original prediction is passed through an **additional module that filters out distractions, using an externally provided *distraction cover***. This distraction cover indicates regions in the image that likely contain distractions. An overview of the adapted TruFor scheme is shown in Figure 4.1.

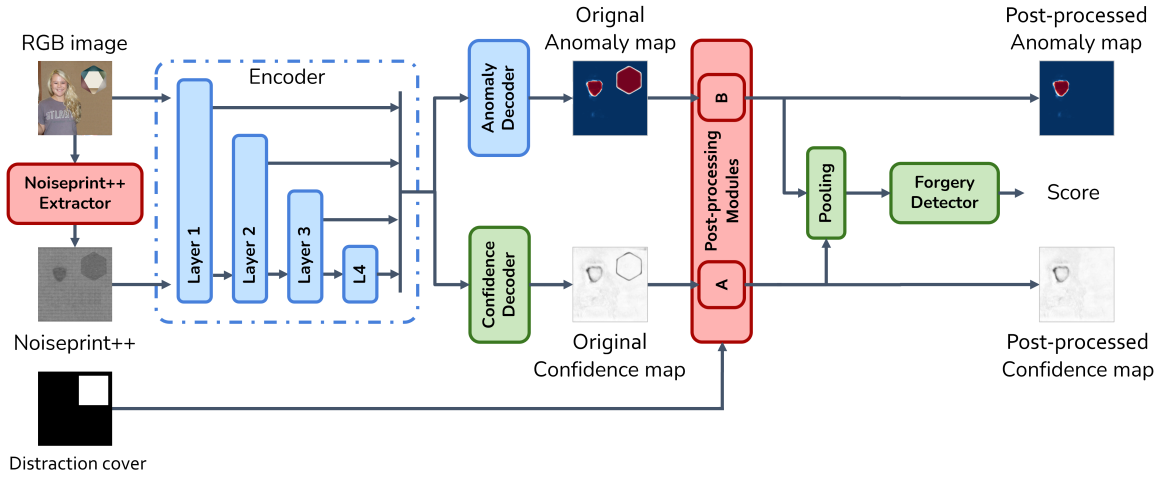


Figure 4.1: Overview of the **post processing** solution incorporated in the TruFor architecture. Adapted from Guillaro et al. (2023) [8]

Although TruFor may fail to detect a forgery in the presence of a distraction, the **forged pixels often still carry weak indications of tampering** — they rarely receive a 0% probability. These subtle cues can be amplified and analyzed to improve detection. Figure 4.2 illustrates this idea. TruFor is unable to detect the forgery in the image due to the presence of the distraction. However, the prediction still contains some low-probability pixels. By adjusting the levels of the prediction, the low-probability pixels become visible, and the forgery becomes more apparent.

In general, many pixels will have a non-zero probability. This does not necessarily mean that the pixel is forged, but it can still be used as an indication of forgery. The goal of the post-processing step would

be to deduce which low-probability pixels are actual forgeries by taking a distraction cover into account.



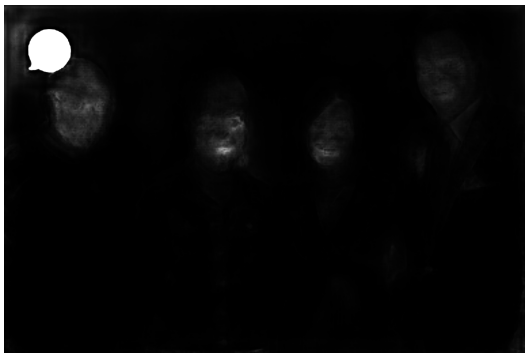
(a) Manipulated image



(b) Ground truth mask



(c) TruFor prediction



(d) TruFor prediction with adjusted levels

Figure 4.2: Illustration showing the low-probability remnants present in the TruFor [8] prediction of a distracted image. Prediction maps are shown in grayscale to better convey the notion of intensity. TruFor is unable to detect the manipulated region in the image. However, the prediction still contains some low-probability remnants. By adjusting the levels of the prediction (e.g., here all pixel values are multiplied by 10, clipping at 1), these remnants become visible.

There are two factors that can limit the effectiveness of this solution. First, distinguishing between low-confidence forged and non-forged pixels in post-processing essentially **reintroduces the forgery localization problem**, but now without access to all the raw data.

Second, by operating after the main model inference, the post-processing step cannot recover **information** that has been **discarded during earlier steps**, reducing its effectiveness. For example, if the tampered pixels get near zero probabilities of being forged, the Post-Processing module will never be able to detect these.

Summary: Post-Processing

When detection fails due to distractions, the **forged pixels often still carry weak indications of tampering**. The post-processing step can use these cues together with a provided distraction cover (which indicates regions that are likely distractions) to improve detection.
However, this essentially **reintroduces the forgery localization problem**, and any **information lost during original inference** cannot be recovered.

4.1.3 Extra Input and Retraining

A third approach is to modify the TruFor architecture to accept an additional input. Similar to the post-processing solution, a distraction cover is passed to the model together with the image to analyze. The cover informs the model about regions that may contain distractions. Now, the RGB image, the NP++,

and, **additionally**, the **distraction cover will be passed through the encoder**, allowing the model to take the distraction cover into account when encoding the image. By adapting the architecture and retraining the model, the final model should be able to ignore selected distractions in the input image. An adapted version of the TruFor architecture is shown in Figure 4.3.

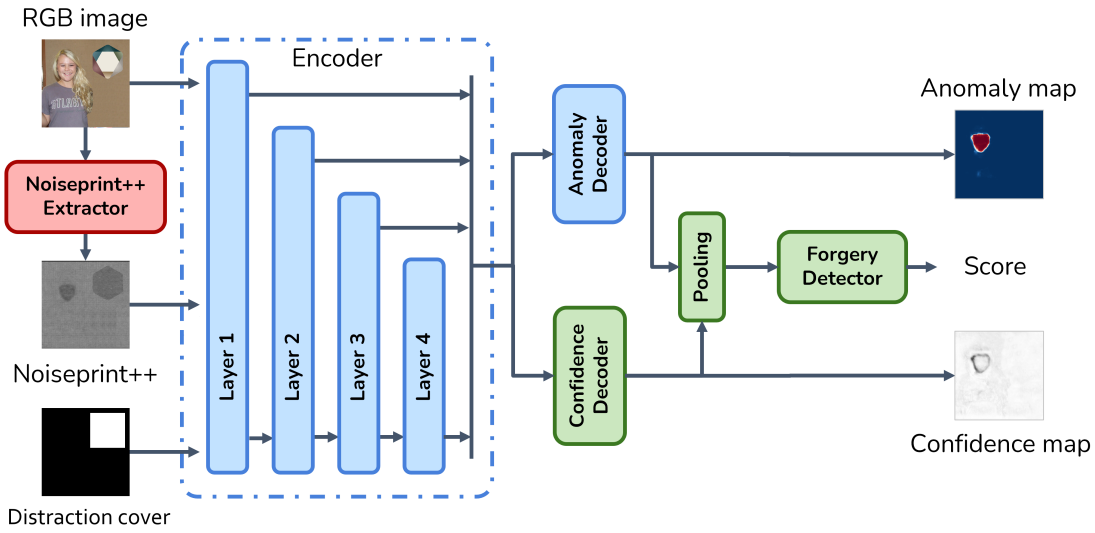


Figure 4.3: Overview of the **extra input** solution incorporated in the TruFor architecture. Adapted from Guillaro et al. (2023) [8]

This method combines the benefits of the previous two approaches. First, by explicitly providing a distraction cover to the model, similar to what was proposed in the post-processing solution, the model can **explicitly learn to ignore regions selected by the cover**. It does not learn specific distractions as the fine-tuning method did.

Second, unlike post-processing, it **leverages raw feature information**. By incorporating the distraction cover early in the network (i.e., before the encoder), the model can learn to suppress distractions before any critical information is lost.

Still, there are limitations. First, in general, **adapting a model to accept an extra input is not a trivial task**. In the specific case of TruFor, there is some redundancy in the Noiseprint++ channels, which can be used to add the extra input. However, as this fundamentally changes the architecture, the original weights cannot be used directly, and thus the **entire model must be retrained**.

Second, this solution has a **lack of freedom**. There are only a few *simple* ways to adapt the existing TruFor model with an extra input (i.e., the distraction cover). The most basic approach is to fuse all inputs at the encoder level and pass everything together through the encoder. This offers little flexibility or experimentation with different architectures.

Lastly, this method is **inherently tied to the TruFor structure**. While model-specific solutions often yield better performance, a more general solution that can be reused across models is preferred. A model-independent solution does not mean that the solution is trained once and reused multiple times. It means that the architectural structure and choices can be reused, only requiring, hopefully limited, training.

Summary: Extra Input and Retraining

The TruFor model can be modified to accept a **distraction cover** (indicating regions that are likely distractions) **as an additional input**. This allows the model to **learn to ignore covered distractions** with **access to the raw feature information**.
However, this requires adapting the model architecture, which is non-trivial, and **retraining the entire model**. The TruFor architecture does allow for a relatively simple adaptation, but this leads to a **lack of freedom** in the design choices, and the solution **cannot be reused** across different models.

4.1.4 Filter Adapter

Another promising solution is to insert an adapter module (or filter) into the existing TruFor architecture. Adapters are **a form of Parameter Efficient Fine-tuning (PEFT)** [57].

Adapters aim to improve task-specific performance while addressing some of the drawbacks of traditional fine-tuning. Fine-tuning typically involves updating all model parameters, which is computationally expensive and can lead to a loss in generalization performance. In contrast, **adapters are lightweight modules** inserted into a pretrained model. They allow for efficient fine-tuning by training only a small subset of parameters (i.e., only the adapter parameters) and leaving other weights unchanged, thereby **reducing computational complexity** and **maintaining generalization**.

Adapters are **popular in the domain of NLP** [57, 58], where they are often used on top of large transformers to create task-specific Large Language Models. In the domain of multimedia forensics, adapters have been used on top of Large Vision Models (LVMs), such as CLIP [44], to perform both FL [59] and SD [45, 46, 24] (two examples provided in Section 2.2.4).

As adapters have been shown to be effective in various domains, they might have the potential to improve the performance of TruFor on the task of distraction-aware forgery localization.

The TruFor model consists of two big parts: an encoder, which transforms the input into a latent feature space, and the decoders, which use the latent features to produce an anomaly map, a forgery score, and a confidence map. This architecture is explained more in depth in Section 2.1.3. This solution would introduce an adapter module, acting as a **filter**, **between the encoder and decoders** in the latent feature space. The filter module takes this latent representation and the distraction cover as input and transforms it into a distraction-filtered latent representation. The goal of the filter is to **change the latent representation based on a distraction cover**, in such a way that the downstream decoders will better localize forgeries while ignoring any highlighted distractions. Figure 4.4 shows the adapted TruFor scheme. A more in-depth explanation of how this solution could practically be implemented in TruFor is provided in Section 4.3.

As the adapter solution is similar to the idea of adding an extra input, the positive attributes are also similar. The adapter uses distraction covers and is trained to **ignore selected regions** rather than specific distractions. As the adapter does **operate after the encoder**, its performance strongly depends on the amount of **information lost** during the encoding process, which was later found to be significant.

There are several additional advantages. First, adapters are **lightweight**, meaning training them is faster and more efficient compared to changing the original model architecture and retraining it.

Second, the architecture of the adapter **can be customized** to match the complexity of the task. The filter must be complex enough to be able to adapt the latent representation in a correct way, but not too complex to overfit on the training data.

Third, the adapter architecture **can be reused** in different models. However, **retraining will be required** for each model, and this specific adapter is **limited to** forensic models with a similar **encoder/decoder architecture**. Fortunately, training an adapter is relatively quick, and many state-of-the-art models adhere to a comparable encoder/decoder design.

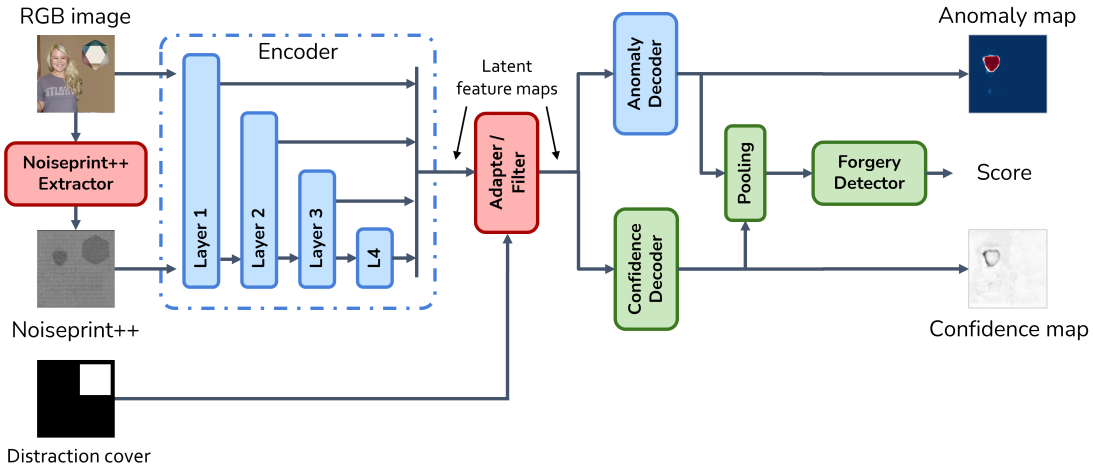


Figure 4.4: Overview of the **adapter** solution incorporated in the TruFor architecture. Adapted from Guillaro et al. (2023) [8]

Finally, since the base model remains unchanged, the adapter **can be easily toggled**, skipped entirely, when no distraction cover is available.

Summary: Filter Adapter

Lightweight neural network modules, called adapters, have been used in several domains to improve the performance of pretrained models on specific tasks. The proposed adapter solution is a filter, **inserted between the encoder and decoders**, which filters out distractions in the latent feature space based on a provided distraction cover.
This solution has several advantages: it is **lightweight**, **reusable**, and **toggleable**. There is **freedom in the design choices**, and the adapter would **ignore selected regions** rather than specific distractions.
As it does **operate after the encoder**, some **information may be lost** during the encoding process, which can limit the effectiveness of the solution. This was later found to be a significant issue.

4.1.5 Masked Convolutions

The final solution proposed in this thesis is to **adapt the convolutional layers** of the TruFor model **to ignore distractions based on a provided mask**. These adapted convolutional layers are called *masked convolutions*. This solution is similar to the adapter solution, but it operates at a lower level, directly modifying the convolutional layers instead of the latent representations.

Existing Solutions

The idea of shifting attention away from distractions and towards more significant parts of the image is not new. CNNs are designed to learn spatial hierarchies of features, making them particularly effective for image processing tasks. However, they can also be sensitive to irrelevant information, such as distractions. **Attention mechanisms** have been proposed to help CNNs focus on the most relevant parts of an image [60, 61]. The idea is that the attention module learns what parts of the image are important for the task at hand, and then focuses on those parts. The opposite strategy, ***reverse attention***, has also been proposed [62], where the model learns to ignore certain parts of the image — again without using a distraction cover of any sort.

The approach of **reverse attention has two main drawbacks**. First, the model must learn to identify what parts of the image are distractions, which is a difficult task in itself. Furthermore, we

assume that a user can provide a distraction cover, which indicates the regions that should be ignored, making this task unnecessary.

Second, the model performs only minor adjustments to the values of these *pixels*. This behavior is expected, as the model often demonstrates uncertainty in its predictions and cannot entirely eliminate the influence of such information. Instead, it adjusts the values closer to 0, which may not be sufficient to mitigate the impact of distractions effectively.

Due to these limitations, this strategy is unsuitable for the task of distraction-aware forgery localization, highlighting the need for a more explicit approach.

Proposed Solution

I propose a solution based on the concept of **masked convolutions**. As opposed to standard convolutions, which convolve the entire image, masked convolutions only convolve the parts of the image that are not covered by a mask. This is done by **treating the masked-out pixels as if they were not part of the image**. The idea is that by replacing all standard convolutions in the TruFor model with masked convolutions, the model should not be influenced anymore by the covered distractions. Thus leading to a regain in focus on the forgery, meaning increased detection performance. A more in-depth explanation of how masked convolutions work and how they are integrated into TruFor is provided in Section 4.4.

The solution of masked convolutions is based on the **following observation**, shown schematically in Figure 4.5. When adding a distraction to the corner of an image, the detection model will often lose focus of the original forgery. However, if we were to **physically crop the distraction out of the image** (i.e., cut away a side or top border), **the model would regain part of its focus**. Only partial focus is regained, as some artifacts of the distraction are still present in the image due to the applied lossy compression, which spread the distraction information throughout the image. Moreover, some valuable forensic information may be lost in the cropped-out part, which may be much larger than the distraction itself.

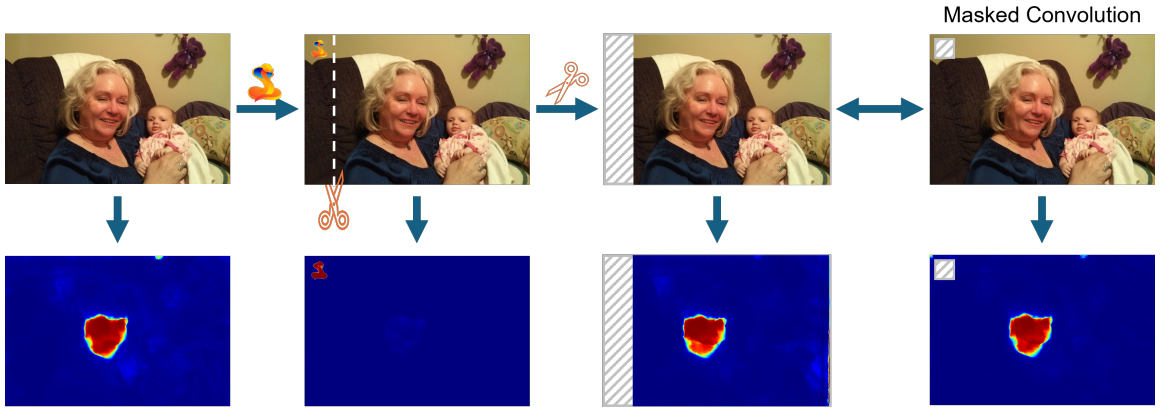


Figure 4.5: Schematic overview of the **observation behind** the masked-convolution solution. By physically cropping the distraction out of the image, the model can regain part of its focus on the original forgery. The masked convolution solution achieves the same effect by *not* convolving over the "cut-out" part, making it possible to cut any distracting region out of the image without needing to physically crop.

Internally, cutting off a part of an image is similar to stopping the convolution when reaching the cut and continuing with the rest of the image. Thus, it is not necessary to physically cut the image, but rather to **stop the convolution when getting to the supposed cut**. This means it should be possible to cut any arbitrarily sized and positioned hole in the image by simply skipping that part of the image during the convolution. This is exactly what the proposed masked convolution does.

Based on the observations made in Figure 4.5, the proposed masked-convolutions approach has high potential of succeeding. Similar to the extra input solution, this method leverages **raw feature information**, and by using a distraction cover, it **ignores regions** — not just some distractions. Furthermore,

this solution requires **no retraining** of the model, as it is simply a modification of the convolutional layers. This means that it can be **added to any model** with a convolutional architecture. Finally, by providing an empty mask, the model will work as it did before, making the solution **toggleable**.

Summary: Masked Convolutions

CNN are sensitive to irrelevant information, such as distractions. Attention mechanisms have been proposed to help CNNs focus on the most relevant parts of an image, but these methods are not suitable for the task of distraction-aware forgery localization.
This thesis proposes **masked convolutions**, which are convolutional layers that ignore regions based on a provided mask. By ensuring that each convolutional layer ignores the distractions, the model as a whole will be able to ignore them.
This solution has several advantages: it uses **raw feature information**, is **toggleable**, and **requires no retraining**. The method is **not model-specific** and can be added to any model with a convolutional architecture, and it **ignores regions** rather than specific distractions.

4.1.6 Overview & Conclusion

Table 4.1 provides a comparison of the four proposed solutions, highlighting their respective strengths and limitations. Based on this analysis, the *filter adapter* and the *masked-convolution* solutions emerge as the most promising.

The use of masked convolution offers key advantages: no training required, access to raw feature information, the ability to be reused on any model with a convolutional architecture, and the option to be toggled on or off. In contrast, the adapter does require some training and operates behind the encoder, limiting its access to the raw feature information.

Sections 4.3 and 4.4, respectively, describe the design and implementation/training of the adapter and masked-convolution solutions. Further investigation of the remaining solutions is left for future work.

Solution	Advantages	Disadvantages
Fine-tuning	+ Uses all information	– Loss of generalization – Learns specific distractions
Post-Processing	+ Ignores regions	– Information already lost – Repeats FL task
Extra Input & Retraining	+ Uses all information + Ignores regions	– High training cost – Limited flexibility – Model-specific
Filter Adapter	+ Uses most information + Ignores regions + No training required + Flexible architecture + Reusable & toggleable	– Needs encoder/decoder structure – Need balance in complexity – Operates behind encoder
Masked Convolutions	+ Uses all information + Ignores regions + No training required + Reusable & toggleable	– Limited to CNNs

Table 4.1: Overview of possible solutions for distraction-aware FL with advantages and disadvantages. The most promising solutions, highlighted in **bold**, are the *filter adapter* and the *masked convolution*. Both solutions are implemented (Section 4.3 and Section 4.4) and evaluated (Section 5.2 and Section 5.3) in this thesis.

4.2 TruFor Technical Details

The TruFor [8] model consists of two big parts: the encoder and decoders (i.e., the Anomaly Decoder, Confidence Decoder, and Forgery Detector). TruFor also contains an NP++ extractor, but this is seen as preprocessing in this chapter and is not changed. A full overview of the technical details is provided in Section 2.1.3. This section provides a short overview of the encoder and decoders, and the used loss functions and datasets during training. This information is needed to understand the distraction-aware solutions.

Encoder The encoder uses a cross-modal transformer-based architecture, adapted from SegFormer [19]. It processes the input (i.e., the rgb image and the NP++) into four feature maps, each with a different resolution. This is done by using a series of transformer blocks, together with CM-FRM and FFM modules [19].

Decoders The four feature maps are processed by the decoders into an anomaly map, confidence map, and forgery score. All decoders are All-MLP decoders [20], except for the Forgery Detector, which is a two layer fully connected network.

Loss functions TruFor is trained in three steps: training the NP++ extractor, training the encoder and Anomaly Decoder, and training the Confidence Decoder and Forgery Detector. Details for the first and third steps are provided in Section 2.1.3. The encoder and Anomaly Decoder are jointly trained using a combination of the weighted cross-entropy loss and the dice-loss [21].

Training/Validation data The encoder and Anomaly Decoder were trained on a combination of 14 datasets, as proposed by CAT-Net [15], consisting of about 877,000 images. Some notable datasets are Fantastic Reality [22], IMD2020 [23], SP COCO, CM COCO, CM RAISE and CM-JPEG RAISE together called tampCOCO [15], and JPEG RAISE [15].

4.3 Filter Adapter

The filter is an adapter that is added in between the encoder and anomaly decoder of TruFor [8]. The goal of the filter is to adapt the feature maps generated by the encoder in such a way that the decoder regains focus of the forgery. A distraction cover is used to identify the pixels that are potentially distracting.

4.3.1 Architecture

The filter is a **small network that is added to the encoder-decoder architecture** of TruFor [8]. As input, it takes the four feature maps generated by the encoder. Additionally, it takes an new input, the *distraction cover*, which is a binary mask with the same size as the input image. The output is a new set of four feature maps, with the same shapes as the input feature maps. The goal is that the artifacts of the distraction are removed from the feature maps so that the decoder can focus on the forgery.

Although multiple architectures can be chosen for the filter, this thesis proposes a **filter consisting of a backbone and four subnetworks**. The backbone is the same SegFormer [20] based architecture as used in the encoder of TruFor. It processes the distraction cover into four feature maps (M_i with $\mathrm{i} = 0, 1, 2, 3$), one for each of the four feature maps produced by the encoder (I_i with $\mathrm{i} = 0, 1, 2, 3$). The motivation behind this is that processing the distraction cover into feature maps in the same way as the encoder processes the input image and NP++ into feature maps, will ease the task of removing distractions. Next, four subnetworks are used to process the four pairs of feature maps (one from the encoder and one from the backbone).

The **four subnetworks** are all identical and consist of 3 fully connected layers with RELU activation and a skip connection. The eight feature maps have the following shapes:

$$\mathrm{I}_0 \text{ and } \mathrm{M}_0 : B \times 64 \times H/4 \times W/4 \qquad \mathrm{I}_1 \text{ and } \mathrm{M}_1 : B \times 128 \times H/8 \times W/8$$
$$\mathrm{I}_2 \text{ and } \mathrm{M}_2 : B \times 320 \times H/16 \times W/16 \qquad \mathrm{I}_3 \text{ and } \mathrm{M}_3 : B \times 512 \times H/32 \times W/32$$

where B is the batch size, H and W are the height and width of the input image, and 64, 128, 320 and 512 are the number of channels in the feature maps (C_i with $\mathrm{i} = 0, 1, 2, 3$).

The **first layer** (L_1) fuses the two feature maps together, going from $2 \cdot C_i$ to 512 channels. The **second layer** (L_2) aims to adapt the feature maps, keeping the number of channels at 512, and the **third layer** (L_3) reduces the number of channels back to C_i. The first 2 layers are followed by a layer normalization and dropout layer. The dropout layer is used to prevent overfitting, and the layer normalization is used to stabilize the training. After all three layers, a RELU activation is applied. Using three layers provides a good trade-off between complexity and performance.

A **skip connection** is added across all 3 layers. The skip connection is used to allow the network to learn the difference between the distracted and non-distracted feature maps instead of learning the non-distracted feature map directly. Only having to learn the residual is a simpler task, and speeds up and stabilizes learning. This is similar to the idea behind the residual networks as proposed by ResNet [63], where the network learns the difference between the input and output of a layer instead of learning the output directly.

The **outputs of the filter** are four distraction-aware feature maps (O_i with $\mathrm{i} = 0, 1, 2, 3$):

$$\mathrm{O}_i = \mathrm{L}_3(\mathrm{L}_2(\mathrm{L}_1(\mathrm{I}_i \oplus \mathrm{M}_i))) + \mathrm{I}_i \qquad \text{for } i = 0, 1, 2, 3 \qquad \text{where } \oplus \text{ is the concatenation operation}$$

In **summary**, the filter consists of a backbone and four subnetworks and is placed between the encoder and anomaly decoder. The backbone processes the distraction cover into four feature maps, and the subnetworks process the pairs of feature maps (i.e., one from the encoder and one from the filter) into distraction-aware feature maps. Each subnetwork consists of three fully connected layers with RELU activation and a skip connection. The skip connection makes the network learn the difference between the distracted and non-distracted feature maps instead of learning the non-distracted feature map directly.

4.3.2 Training

The filter is trained using part of TruFor's [8] **training data**. Being Fantastic Reality [22], tampCOCO [15] (consisting of 4 subdatasets: SP COCO, CM COCO, CM RAISE and CM-JPEG RAISE) and JPEG RAISE [15]. IMD2020 [23] was excluded from training as it was used in Chapter 3 to evaluate the impact of forensic distractions and is again used in Chapter 5 to evaluate the filter's performance. All images were resized and/or cropped to 1536x1024, and the same number of images were used from each dataset, limited by the smallest (i.e., Fantastic Reality), to ensure a balanced training. An overview of the used datasets is provided in Table 4.2.

Dataset	Size	Description
Fantastic Reality [22]	2×16,000	Both authentic and spliced images from various scenes
SP COCO [15]	200,000	Spliced images, compressed
CM COCO [15]	200,000	Copy-move forgeries, compressed
CM RAISE [15]	200,000	Copy-move forgeries with random shapes, compressed
CM-JPEG RAISE [15]	200,000	Extra compressed version of CM-RAISE
JPEG RAISE [15]	24,462	Authentic dataset created by compressing RAISE [54] images

Table 4.2: Overview of datasets used for training the adapter. SP COCO, CM COCO, CM RAISE, and CM-JPEG RAISE are subdatasets of the tampCOCO dataset [15]. The same number of images were selected from each dataset, limited by the smallest (i.e., Fantastic Reality; 16,000), to ensure a balanced training set. "Compressed" in this table refers to JPEG compression with a random quality factor between 60 and 100.

The goal is to learn to ignore distractions based on a provided distraction cover. To do this, the **dataloader was modified** to randomly select a rectangle in the image, with a size between 5% and 20% of the size of the image, and set the pixels in this rectangle to 0. Next, the image was JPEG compressed with Q95, just as was done in the experiments in Chapter 3. Lastly, a distraction cover was generated based on the chosen rectangle.

As explained in Section 3.1, the presence of the JPEG compression step is partially responsible for the problem of forensic distractions. Therefore, it is important to include this step in the training process.

The filter is trained using the same **loss functions** as TruFor used for training its Anomaly Decoder. The loss function is a combination of the weighted cross-entropy loss and the dice-loss [21]. In this thesis, the loss functions were modified to only take into account the pixels that are *not* masked by the distraction cover (similar to the adapted F1 score calculation). The adam optimizer was used with a weight decay of 0.0005 and a base learning rate of 0.0001 with polynomial decay. The training was done using a batch size of 4 due to memory constraints. To prevent unstable training, **gradient accumulation** was used with a weight update every 4 batches. Every 800 batches (or 3,200 samples or 200 weight updates) a **validation round** was performed. The validation set consists of 550 samples, each with a rectangular distraction with a fixed position and size. These positions and sizes were randomly generated before starting the training and remained constant throughout the training to make validation results comparable. Training was done for 7 epochs, with each epoch containing 26,760 batches (or 107,040 samples or 6690 weight updates).

The training loss, validation loss, and validation Intersection over Union (IoU) are shown in Figure 4.6. The **training was stopped after 7 epochs** as the validation loss started to stagnate. At the end of the training, the validation loss was 0.0012 and the validation IoU was 0.95. The validation IoU did not increase a lot during training, indicating that the model was not able to learn a lot. This inability to learn is further evaluated and explained in Section 5.2 where the results of the filter are discussed.



(a) Training loss

(b) Validation loss

(c) Validation IoU

Figure 4.6: Training loss, validation loss, and validation IoU of the adapter. Updates and loss calculations are performed every 16 samples. A validation round is performed every 3200 samples. Plots are smoothed with a running average of 10.

4.4 Masked Convolutions

The proposed masked-convolution approach aims to make TruFor [8] distraction-aware by **modifying the convolutional layers to ignore the distracting pixels**. Figure 4.5 shows that cutting off the distraction from an image leads to partial focus being regained. Instead of physically cutting the distraction out of the image, it should be possible to simply skip this part of the image during the convolution. To do this, a mask needs to be provided to the convolutional layers, indicating which part of the image should be ignored.

The masked-convolution approach has **two parts**. First, the necessary components, including but not limited to the convolutions, need to be modified to correctly process the input based on a provided mask. For simplicity, these are called *masked* components. All components that mix information from different pixels need to be masked. The masked components are discussed in Section 4.4.1. Second, the existing model needs to be adapted to use the new masked components. This includes correctly generating the mask and passing it to the different components. This is discussed in Section 4.4.2.

4.4.1 Masking Components

Understanding convolutional layers

The main component that needs to be masked is the 2D convolutional layer. In its simplest form, shown in Figure 4.7, a convolution operation can be explained as follows: a small region of the input image, called a convolution window X, is selected. The size of this window is determined by the kernel size K. A convolution kernel W is then applied to this window, producing an output value $x' = W^T X + b$, where b is the bias. The convolution window is then shifted by a specified number of pixels, called the stride S, and the process is repeated across the entire image (left to right, top to bottom).

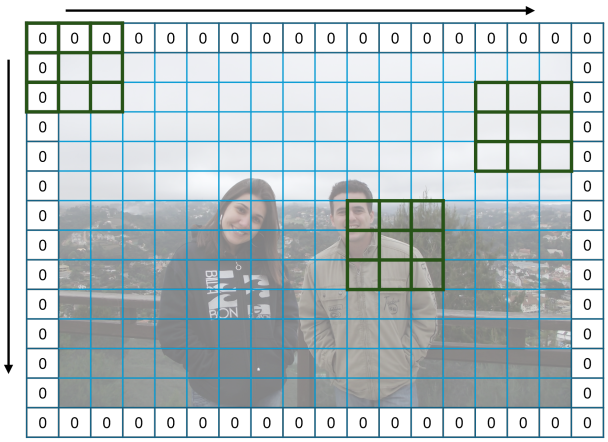


Figure 4.7: Example of a convolutional layer operating on an image of size 12x16 with a kernel of size 3x3, a stride of 2 and a padding of 1. The kernel shifts over the image in steps of 2 pixels (left to right, top to bottom), at each step performing a convolution operation. Three example convolution windows are shown.

To handle **edge cases** where the convolution window extends beyond the boundaries of the image, padding of size P can be added around the image. The final output of the convolution is a new image with dimensions determined by the formula:

$$\text{Output size}_{\text{dim } i} = \frac{L_i - K + 2P}{S} + 1$$

where L_i is the length of the input image in dimension i. This process is repeated for all regions of the input image, resulting in a transformed output image. Kernel size K, stride S, and padding P can be different for each dimension.

In practice, convolution operations work with **multiple channels**. For instance, an RGB image has 3 channels. For an input image with C_{in} channels, the convolution kernel W is a 3D tensor with dimensions $K_r \times K_c \times C_{\text{in}}$. Each kernel processes all input channels simultaneously, and the output is a single channel. To produce an output with C_{out} channels, C_{out} separate kernels are used, each with its own set of weights and biases. Thus, the total number of parameters in a convolutional layer is given by:

$$\text{Number of parameters} = C_{\text{out}} \times (K \times K \times C_{\text{in}} + 1)$$

where the $+1$ accounts for the bias term for each output channel. This allows the convolutional layer to learn complex transformations across both spatial dimensions and channels.

Masked convolutional Layer

For simplicity, in this thesis, the mask is a rectangle defined by its top-left corner (x, y) and its dimensions (h, w) (height and width). This rectangle represents a hole in the image. All pixels inside the hole are considered invalid and should be ignored by the convolution operation. This is done by skipping over the hole during the convolution operation using the following three steps illustrated in Figure 4.8.

- **Pad the hole**: Add the same amount of padding as added on the outside of the image, to the inside of the hole. Adding the padding to the hole sets some invalid pixels to 0. The others remain invalid. Adding the padding ensures that the convolution windows overlapping with the hole are handled similarly to those extending beyond the image boundaries.
- **Convolve**: Perform the default convolution operation.
- **Update the mask**: Any convolution window that overlaps with non-padded invalid pixels is itself invalid. This means that the output of the convolution should be ignored in subsequent layers. To achieve this, the mask is updated to reflect the new validities using the following steps:
  - Count the number of *valid* convolution windows before the hole in both dimensions.
  - Count the number of *invalid* windows within the hole in both dimensions.
  - These counts are, respectively, the new top-left corner coordinates and the new height and width of the updated mask.

This solution currently supports only rectangular holes, but could be extended to irregular shapes if needed. Rectangular holes suffice for the current use case and as a proof of concept.

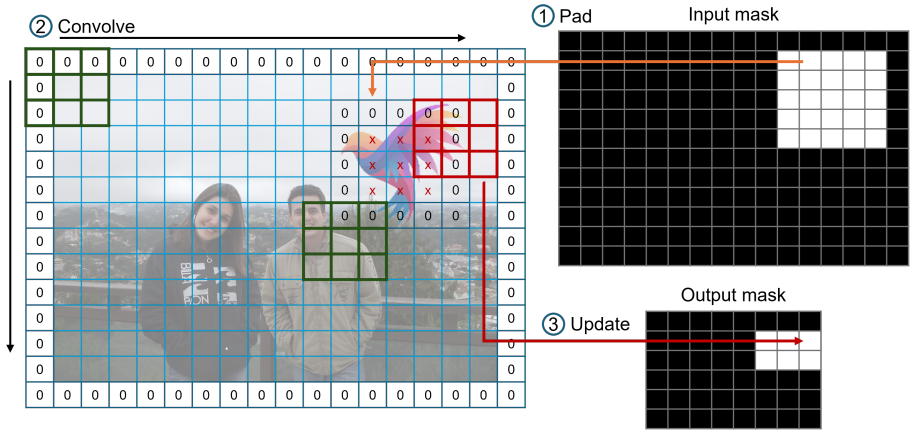


Figure 4.8: Illustration of the three steps taken by the proposed masked convolution layer. The illustration shows a 12x16 image with a 3x3 convolutional kernel, a stride of 2 and a padding of 1. Three example convolution windows are shown, two of which are valid (green) and one which is invalid (red). First, the padding is applied to the hole (pad). Next, the convolution is performed as normal (convolve). Finally, the mask is updated to reflect the new hole (update).

Masked avg pooling and max pooling

The four CM-FRM blocks in the encoder make use of adaptive max pooling and average pooling operations, which compress all information into a single number to capture a feature summary per channel. These are then combined using a small network to calculate channel weights, which are then used, together with spatial weights calculated via a CNN, to rescale the feature maps, allowing the model to emphasize informative regions and suppress less relevant features.

The pooling operations mix information from different pixels and therefore also need to be masked. Masking these operations is very simple. Instead of average over the entire channels of the feature map, the average is only taken over the unmasked pixels. The same applies to the max pooling operation. The mask is not changed during this operation, so it does not need to be returned.

4.4.2 Integration with TruFor

The second part of the solution is to integrate the masked components into the TruFor [8] architecture. The TruFor model is implemented using the PyTorch [64] framework. The masked components are implemented as separate modules, inheriting from the original components, which they replace. By implementing the masked components in this way, the original weights can simply be reused. PyTorch immediately recognizes the new components as wrappers of the original ones and uses the original weights.

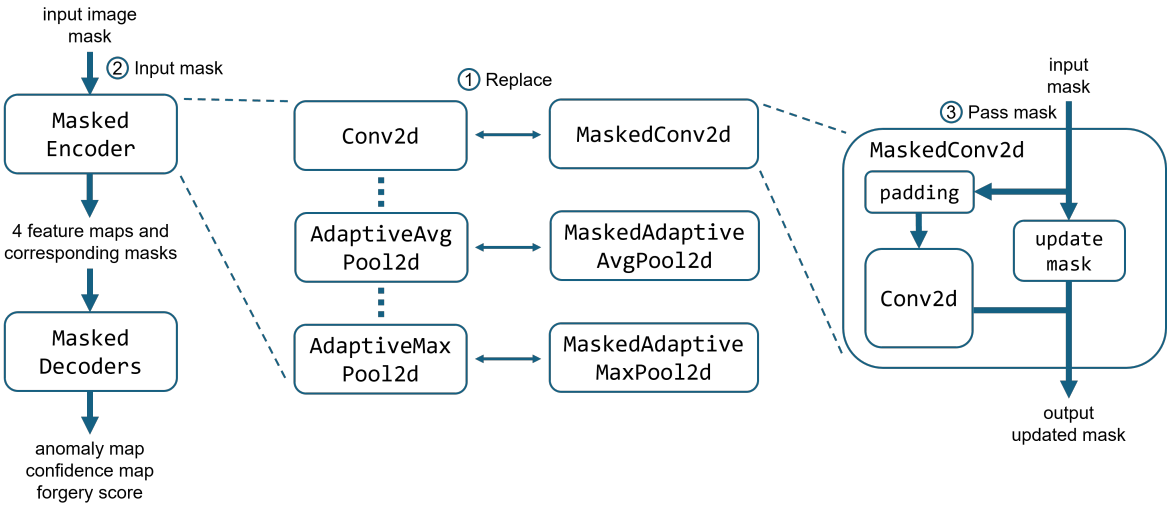


Figure 4.9: Schematic overview of the integration of the masked components into the TruFor architecture. First, components that mix information from different pixels are replaced with their masked versions (replace). Next, the mask is generated and passed through the network (input mask). Finally, the mask is passed to the masked components, which will use it and return an updated version (pass mask).

Figure 4.9 shows a schematic overview of the integration of the masked components into the TruFor architecture. The first step is to replace all components that mix information from different pixels with the newly created masked versions (replace). The second step is to generate the mask from the distraction cover, and pass it through the network together with the input (input mask). The mask follows a similar path to the input. Every time a masked component is used, the mask must be passed to the component, which will use it and return an updated version (pass mask). The output of the encoder is extended to output the corresponding mask for each feature map. The feature maps and their masks are then passed to the decoders, which are also modified to use the masked components.

As the masked components, contrary to the original components, take two inputs and return two outputs (the feature map and the mask), they cannot be used inside a Sequential Module. Sequential Modules are used to group multiple layers together, making it possible to use them as if they were a single layer. Inside, the output of each layer is passed to the next layer. This is only possible if each

layer takes the same number of inputs and returns the same number of outputs, which is not the case if masked and non-masked components are mixed.

This means that any Sequential Module that contains at least one masked and one non-masked component needs to be dismantled (i.e., separating the layers). The implications of this are that the weights of the Sequential Module cannot be simply reused. To solve this, some weights need to be renamed.

For instance, consider a Sequential Module named "seq". Its weights will be named "seq.0.weight", "seq.1.weight", and so on. This module can be dismantled by separating the internal modules and renaming them as "seq_0", "seq_1", etc. The final step is to rename the weights from "seq.0.weight" to "seq_0.weight", and so on.

The explanation provided in this section is not limited to TruFor, and can, in theory, be used to implement the masked components in any CNN-based model. If the model is not implemented using PyTorch, some modifications may be needed, but the concepts remains the same.

4.4.3 Similar Approach used for Inpainting

The idea of masked convolutions is similar to the work proposed by Liu et al. (2018) [65]. They used a similarly modified convolution operation for the task of inpainting with irregular-shaped holes. Their method uses a binary mask to indicate which pixels should be ignored during the convolution — 1 for pixels that should be considered and 0 for pixels that should be ignored.

As opposed to the proposed method of masked convolutions, Liu et al. treat holes in the image differently from pixels outside the image. First, the convolutional operation is performed as normal, but the ignored pixels (i.e., pixels with value 0 in the binary mask) are set to 0. Next, the outputs (i.e., the values for each convolution window) are rescaled to account for ignored pixels. The rescale factor is calculated as the size of the convolution window divided by the number of considered pixels (i.e., pixels with value 1 in the binary mask) in that window.

4.5 Conclusion

In this chapter, **five possible solutions were discussed** to make TruFor [8] distraction-aware. Four of these solutions are based on performing additional training, being finetuning (Section 4.1.1), post-processing (Section 4.1.2), adding an extra input and retraining (Section 4.1.3), and using a filter adapter (Section 4.1.4). The fifth solution does not require any training, only replacing convolutions by a masked counterpart (Section 4.1.5). An overview of the solutions with their advantages and disadvantages is shown in Table 4.1.

The **most promising** solutions were the filter adapter and the masked convolution. Both solutions can be reused on different models (limited to some architectures) and can be toggled on or off. As the **filter solution** does require training, and as it operates behind the encoder, it has limited access to the raw feature information.

The **masked-convolution solution** does not require any training. It can thus be immediately used on any pretrained CNN-based model, apart from the fact that the model needs to be adapted to use the new components.

Both solutions are discussed and implemented in this chapter. The filter adapter solution is implemented in Section 4.3, and the masked-convolution solution is implemented in Section 4.4.

In the **next chapter**, **the two solutions are evaluated**. The goal of the evaluation is to determine if the proposed solutions are able to make TruFor distraction-aware, and if so, how well they perform. The evaluation is done using the same datasets as used in Chapter 3, and the results are compared to the results of TruFor with and without distractions.

5

Evaluation

In Chapter 4, two different solutions were proposed and implemented to solve the problem of forensic distractions. The first solution is a distraction filter that can be added to existing models, limited to encoder-decoder architectures. The second solution is to use masked convolutions that ignore part of the image. This solution does not need to be trained and can be used in any model, limited to CNN architectures. In this chapter, both solutions are evaluated, and the results are compared to the original TruFor [8] model. The goal is to determine whether the proposed solutions can effectively mitigate the impact of forensic distractions on forgery localization tasks. Figure 5.1 shows a manipulated distracted image and the corresponding predictions by the unmodified TruFor model and the two proposed solutions.

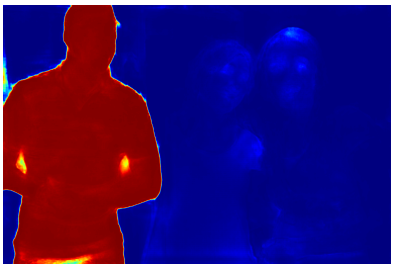
The experimental setup is briefly described in Section 5.1. The results for both solutions are, respectively, discussed in Section 5.2 and Section 5.3. Tables 5.1 and 5.2 show the performance of the two proposed solutions.



(a) Manipulated image, with distraction in upper-right corner



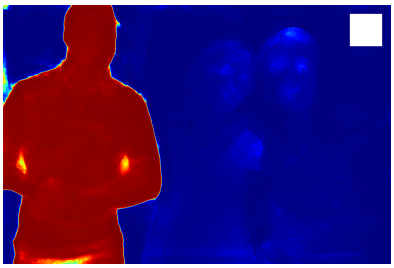
(b) Ground truth mask



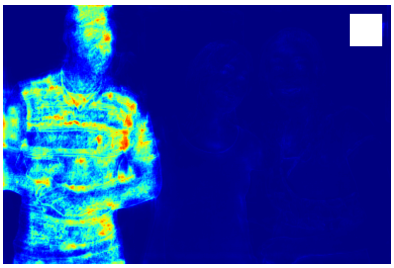
(c) Prediction by unmodified TruFor *without* distraction, F1* score: 0.9616



(d) Prediction by unmodified TruFor *with* distraction, F1* score: 0.0061



(e) Prediction by the **masked-convolution** approach *with* distraction, F1* score: 0.9619



(f) Prediction by the **filter adapter** approach *with* distraction, F1* score: 0.4926

Figure 5.1: Example of the performance of unmodified TruFor [8] vs. the two proposed solutions on the same manipulated image from Chapter 3. The unmodified TruFor model is unable to detect the manipulated region in the presence of a distraction, while both proposed solutions are able to detect the manipulation, albeit with varying accuracy. The blue box in (b) and white boxes in (e) and (f) indicate the masked region.

5.1 Experiment Setup

The experiments conducted in this chapter are based on the **same setup as the experiments in Chapter 3**. A detailed description of the setup can be found in Section 3.2.1. Below is a brief summary of the most important aspects of the setup.

Datasets The experiments were conducted on **four datasets**, DSO-1 [49], VIPP [50], OpenForensics [51], and IMD2020 [23]. An overview of the datasets is provided in Table 3.1.

Distractions **Ten different logos** were used, which vary in color, shape, and content. They were selected to be representative of the types of distractions commonly added by social media platforms or news outlets. The logos are shown in Figure 3.3. Each image was randomly assigned a distraction and a position (upper-left, upper-right, lower-left, lower-right), constant over all experiments. After the distraction was added, the image was lossy compressed using JPEG Q95, which imitates the compression applied by social media platforms while keeping quality high for evaluation.

Distraction cover The distraction cover is a **binary mask that indicates what pixels should be ignored by the model**. In these experiments, the cover is generated by selecting the smallest rectangle that contains the distraction, padded by 4 pixels on all sides. The distraction cover is given as input to the model and is used in the calculation of the detection scores in this chapter.

Detection score The prediction performance was evaluated using the an adapted F1 score as defined in Section 3.2.1, with the distraction cover used as the binary mask (M). The **adapted F1 score** is used for two reasons. First, just as before, the model should neither be rewarded nor punished for detecting the distraction. Second, the modified model generates nonsensical predictions for the masked pixels, which would skew the results if taken into account.

As a reminder, the F1* score is computed as follows:

$$\text{F1* score(P, G, M)} = \frac{2\text{TP[M=0]}}{2\text{TP[M=0]} + \text{FP[M=0]} + \text{FN[M=0]}}$$

with P representing the model prediction, G the ground truth mask and M the distraction cover in this case, all having values in the range $[0, 1]$:

The adapted F1 score used in this chapter takes into account a rectangle on top of the distraction. This is different from Chapter 3 where the distraction it self was taken into account. However, the resulting difference in F1* score is negligible, meaning that the results are still comparable.

5.2 Filter Adapter

The filter solution consists of an adapter placed between the encoder and decoders of the TruFor [8] model. The adapter is a lightweight neural network that is trained to filter out the distraction from the feature maps produced by the encoder based on a provided distraction mask. The architecture and training setup of the adapter are described in Section 4.3. Unlike the TruFor model, the adapter was *not* trained on the IMD2020 dataset. This makes the results more generalizable and comparable to the results in Section 3.2.

The results of the filter solution are shown in Table 5.1. It can be seen that the adapter **slightly improves the performance** of the TruFor model across all datasets, with the most significant improvement observed on the VIPP dataset and quite small improvements on the DSO-1 and IMD2020 datasets. The average F1* score across all datasets increased from 0.21 to 0.25, which is a **non-negligible relative improvement of 16.90%**.

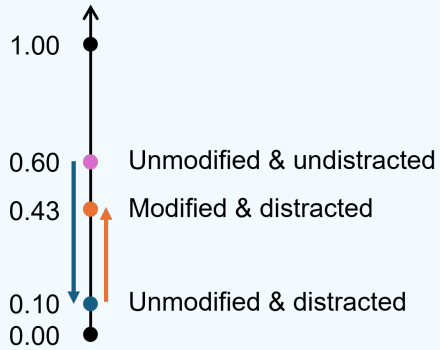
How relative percentage increases are calculated
The relative percentage increase is a metric used to evaluate the increase in performance of a modified model compared to the original model. It is calculated as follows:

$$\text{Relative Percentage Increase} = \frac{\text{F1* score [M D]} - \text{F1* score [notM D]}}{\text{F1* score [notM notD]} - \text{F1* score [notM D]}}$$

with M and D indicating modified and distracted, respectively. The relative percentage increase is a value between 0% and 100%, with 0% indicating no improvement and 100% indicating that the modified model performs as well as the unmodified model on non-distracted images.

Example:
The original unmodified model has an F1* score of 0.60 on non-distracted images and an F1* score of 0.10 on distracted images. The modified model has an F1* score of 0.43 on distracted images. In this case, the relative percentage increase is:

$$\frac{0.43 - 0.10}{0.60 - 0.10} = \frac{0.33}{0.50} = 66\%$$



The model is **evaluated at two different checkpoints**. At the end of training (after 7 epochs) and after 3 epochs. The average performance of the model after 3 epochs is better than the performance after 7 epochs. This points to the **model overfitting to the training data**. The 7-epoch model did, however, perform **better on the VIPP dataset**.

One possible explanation is that due to the compression performed on all images during training, the **model gets better at detecting forgeries post-compression**, and more specifically at detecting DQ artifacts. The VIPP dataset was built to evaluate the performance of an FL model that looks for DQ artifacts, and it thus contains many DQ images. I hypothesize that as the model continues training, it gets better at detecting these DQ artifacts, which leads to a better performance on the VIPP dataset.

Examples Figure 5.3 shows eight examples of manipulated images with their corresponding predictions by both the unmodified TruFor model and the proposed filter adapter solution (the 7-epoch model).

In general, however, the adapter solution is **unable to make very significant improvements** in performance, being stuck at an average F1* score of 0.25, which is quite low compared to the original F1* score (i.e., without distractions) of 0.47. This is likely **due to the fact that the encoder already**

Dataset	Unmodified TruFor		Filter (3 epochs)		Filter (7 epochs)	
	Original	Distracted	Distracted	Improvement	Distracted	Improvement
DSO-1 [49]	0.43493	0.16117	0.17045	3.39%	0.16931	2.97%
VIPP [50]	0.32978	0.02787	0.09052	20.75%	0.10976	27.12%
OpenForensics [51]	0.54804	0.22184	0.31542	28.69%	0.30045	24.10%
IMD2020 [23]	0.55829*	0.42579*	0.44277*	12.82%	0.43191*	4.62%
Average	0.46776	0.20917	0.25479	17.64%	0.25286	16.90%

Table 5.1: Evaluation of the **Filter Adapter** solution on the different datasets compared to the unmodified TruFor [8] model using the F1* score (3.3). The Filter is evaluated at two different checkpoints (3 epochs and 7 epochs). The shown percentage increases are relative to the possible improvement, with 100% being the original TruFor model and 0% being the unmodified TruFor model.
The performance of the adapter is better after 3 epochs than after 7 epochs, which indicates that the model is overfitting to the training data. In general, the improvements are limited, but non-negligible, meaning that the adapter is a valid solution to the problem of forensic distractions.
Remark:* The original TruFor model was trained on the IMD2020 dataset, which results in general higher robustness against distractions in this dataset.

lost most of the information needed to detect the forgery, making it difficult for the adapter to extract any remaining information. This theory is supported by the results of a small ablation study performed in Section 5.3.1, which shows that the encoder is the biggest culprit for the performance drop during the addition of distractions.

Even though improvements are limited, the adapter solution is **still a valid solution to the problem of forensic distractions**.

Summary

In summary, the filter is a lightweight neural network placed between the encoder and decoders of TruFor, with the goal of filtering out distractions based on a provided distraction mask.
The filter **improved the performance** on distracted images across all datasets, achieving a 16.90% relative improvement. However, the improvements were **limited**, likely because the encoder discards crucial forensic information when distractions are present.

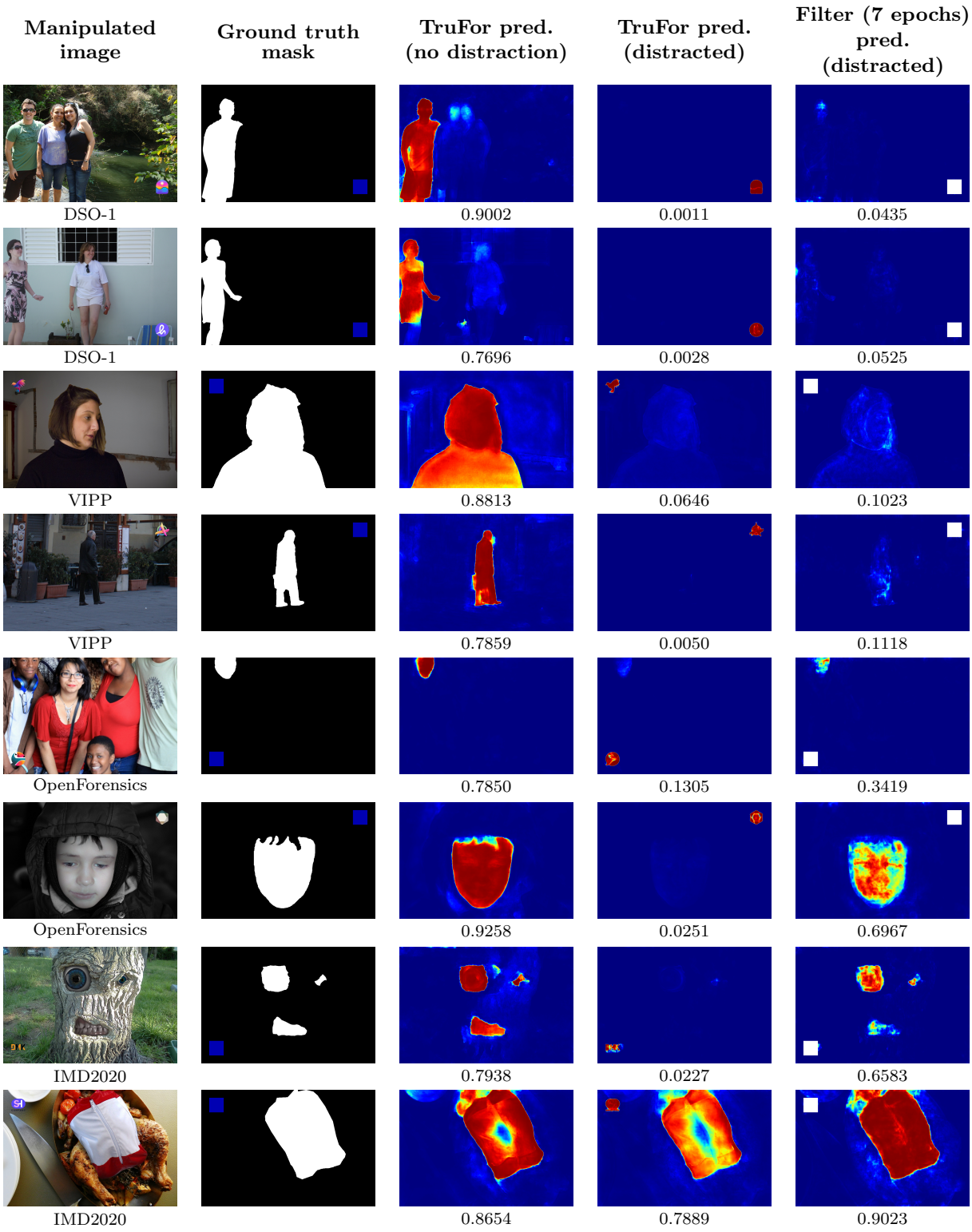


Figure 5.2: Examples of manipulated images with corresponding predictions and F1* scores (3.3) by both unmodified TruFor [8] and the **proposed filter-adapter solution** (the 7-epoch model). The blue boxes in the ground truth and the white boxes in the last column indicate the masked regions. The last column shows the predictions of the filter adapter approach, all other columns are the same as in Figure 5.3. The filter adapter approach is able to improve the performance of TruFor on every image, most notably on rows 6 and 7. In general, the performance increase is limited.

5.3 Masked Convolutions

The masked-convolution solution consists of replacing any standard convolution with a masked convolution. The masked convolution takes a mask as an extra input. The mask defines a hole in the image, which the convolution will treat as if there were no pixels there. The construction of the masked convolution and other necessary components, and the implementation in the TruFor architecture, are described in Section 4.4.

Table 5.2 shows the results of the masked-convolution solution. This solution led to a **significant improvement in performance** across all datasets.

Dataset	**Unmodified TruFor**		**Masked Convolution**	
	Original	Distracted	Distracted	Improvement
DSO-1 [49]	0.43493	0.16117	0.38424	81.48%
VIPP [50]	0.32978	0.02787	0.28855	86.34%
OpenForensics [51]	0.54804	0.22184	0.53023	94.54%
IMD2020 [23]	0.55829*	0.42579*	0.53473*	82.22%
Average	0.46776	0.20917	0.43444	87.11%

Table 5.2: Evaluation of the **Masked-Convolution** solution on the different datasets compared to the unmodified TruFor [8] model using the F1* score (3.3). The shown percentage increases are relative to the possible improvement, with 100% being the original TruFor model and 0% being the unmodified TruFor model.
The masked-convolution solution performs significantly better than the unmodified TruFor model in the presence of distractions, with an average relative improvement of 87.11%. This makes the masked convolution a very effective solution to the problem of forensic distractions.
Remark:* The original TruFor model was trained on the IMD2020 dataset, which results in general higher robustness against distractions in this dataset.

The average F1* score across all datasets increased from 0.21 to 0.43 for a **87.11% relative improvement**, making it the **best performing FL model tested** (see Table 5.4). The masked-convolution solution only drops 7.12% in performance when distractions are added, compared to 55.28% for the unmodified TruFor model (see Table 5.4). This makes it **very resilient to distractions**.

Examples Figure 5.3 shows eight examples (same examples as in Section 5.2) of manipulated images with their corresponding predictions by both the unmodified TruFor model and the proposed masked-convolution solution.

The results show that the masked-convolution approach is a very effective solution to the problem of forensic distractions. As the method is merely a modification of the convolution operation, it can in theory be used in any CNN-based model, making it a very versatile solution.

5.3.1 Ablation Study

To better understand the impact of the masked-convolution solution, a small ablation study was performed. The goal of this study was to **evaluate the relative impact of the encoder and anomaly decoder on the performance**. To this end, two new versions of the masked-convolution solution were created. In the first version, only the encoder was adapted to use masked convolutions, while the decoder was left unchanged. In the second version, only the decoder was adapted, while the encoder was left unchanged. The results of this ablation study are shown in Table 5.3.

The results show that the **encoder is the most important component in the story of forensic distractions**. Adapting only the encoder leads to a significant average performance relative improvement

Dataset	Unmodified TruFor		Masked Convolution		
	Original	Distracted	Encoder only	Decoder only	Fully masked
DSO-1 [49]	0.43493	0.16117	0.38424	0.18348	0.38424
VIPP [50]	0.32978	0.02787	0.28854	0.03419	0.28855
OpenForensics [51]	0.54804	0.22184	0.53022	0.18064	0.53023
IMD2020 [23]	0.55829	0.42579	0.53468	0.49757	0.53473
Average	0.46776	0.20917	0.43442	0.22397	0.43444

Table 5.3: **Ablation study** on the **Masked-Convolution** solution, to evaluate the relative impact of the encoder and anomaly decoder on the performance. The performance is evaluated using the F1* score (3.3).
The results show that only adapting the encoder is sufficient to achieve a significant performance improvement. Adapting only the decoder leads to a very small average performance gain. For the OpenForensics dataset, it even leads to a performance loss, the reason of which may be a topic for future work. This implies that the encoder is the most important component, being responsible for the majority of the performance drop when distractions are added.
Remark:* The original TruFor model was trained on the IMD2020 dataset, which results in general higher robustness against distractions in this dataset.

of 87.11%, while adapting only the anomaly decoder leads to a very small average relative performance of only 5.72%. On the OpenForensics dataset the performance even degrades when only adapting the decoder. The reason for this decrease is unclear and may be a topic for future work. This implies that the **encoder is the main culprit for the performance drop when distractions are added**.

This further **explains the results of the adapter solution** presented in Section 5.2. The adapter was unable to significantly improve the performance of the TruFor model because the encoder already lost most of the information needed to detect the forgery.

Summary

The use of masked convolutions led to a **significant improvement** in performance across all datasets, with an average F1* score increase from 0.21 to 0.43, being a 87.11% relative improvement. This makes it the best-performing FL model tested. With a performance drop of only 7.12% when adding distractions, it is very resilient to distractions.
An ablation study revealed that the **encoder** is the **main source of distraction vulnerability**, further explaining the limited improvements of the adapter solution.
The approach is highly versatile, as it can be applied to any CNN-based model, making it a promising solution for the future of distraction-aware FL methods.

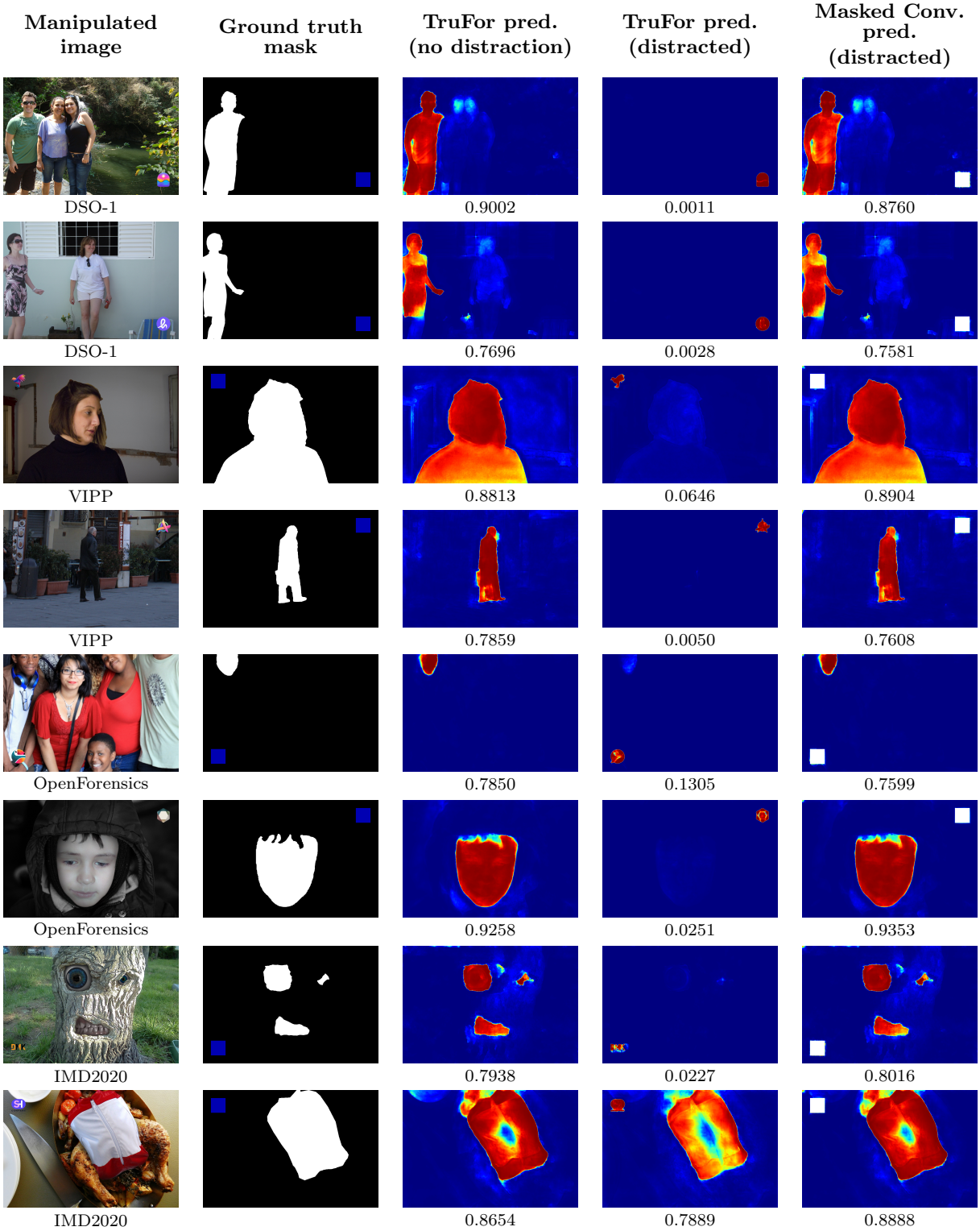


Figure 5.3: Examples of manipulated images with corresponding predictions and F1* scores (3.3) by both unmodified TruFor [8] and the **proposed masked-convolution solution**. The blue boxes in the ground truth and the white boxes in the last column indicate the masked regions. The last column shows the predictions of the masked-convolution approach, all other columns are the same as in Figure 5.2. The masked-convolution approach significantly improves the performance of TruFor in the presence of distractions, demonstrating its effectiveness as a solution to the problem of forensic distractions.

		Model	DSO-1			VIPP			OpenForensics		
			Original	Distracted	Decrease	Original	Distracted	Decrease	Original	Distracted	Decrease
Model-based		DCT [11]	0.23122	0.23149	-0.00026	0.18852	0.18696	0.00156	0.11154	0.11099	0.00054
		ADQ1 [12]	0.22461	0.22355	0.00106	0.17930	0.17885	0.00045	0.09869	0.09824	0.00045
		BLK [13]	0.18553	0.18541	0.00011	0.14365	0.14194	0.00172	0.07594	0.07517	0.00077
		CAGI [14]	0.19477	0.18995	0.00482	0.18724	0.18925	-0.00201	0.06512	0.06532	-0.00020
Deep learning		CAT-Net [15]	0.06668	0.05896	0.00772	0.12630	0.11566	0.01064	0.28374	0.27457	0.00917
		PSCC-Net [18]	0.13760	0.08652	0.05108	0.16504	0.13932	0.02572	0.11167	0.08241	0.02926
	Trufor	Unmodified [8]	**0.43493**	0.16117	0.27376	**0.32978**	0.02787	0.30191	**0.54804**	0.22184	0.32620
		Filter (§4.3)		0.16931	0.26562		0.10976	0.22002		0.30045	0.24759
		Masked Conv. (§4.4)		**0.38424**	0.05069		**0.28855**	0.04124		**0.53023**	0.01782

(a) DSO-1 [49], VIPP [50] and OpenForensics [51] datasets

		Model	IMD2020			Average			
			Original	Distracted	Decrease	Original	Distracted	Decrease	% Decrease
Model-based		DCT [11]	0.14660	0.14284	0.00376	0.16947	0.16807	0.00140	0.83%
		ADQ1 [12]	0.14161	0.13838	0.00323	0.16105	0.15976	0.00130	0.80%
		BLK [13]	0.12223	0.12023	0.00199	0.13184	0.13069	0.00115	0.87%
		CAGI [14]	0.11983	0.11765	0.00218	0.14174	0.14054	0.00120	0.84%
Deep learning		CAT-Net [15]	0.52894*	0.51360*	0.01534	0.25141	0.24070	0.01072	4.26%
		PSCC-Net [18]	0.07405	0.05962	0.01443	0.12209	0.09197	0.03012	24.67%
	Trufor	Unmodified [8]	**0.55829***	0.42579*	0.13250	**0.46776**	0.20917	0.25859	**55.28%**
		Filter (§4.3)		0.43191*	0.12638		0.25286	0.21490	**45.94%**
		Masked Conv. (§4.4)		**0.53473***	0.02356		**0.43444**	0.03333	**7.12%**

(b) IMD2020 [23] dataset and average across all datasets

Table 5.4: Summarizing table showing the localization scores (F1* score (3.3)) of the tested FL models and the two proposed solutions on the different datasets before and after adding distractions, and the average across all datasets. The best score for each dataset is highlighted in bold and the second best score is underlined. The percentage decrease is shown in parentheses and changes of less than 1% are shown in light gray.
The Masked-Convolution solution performs the best (post distraction) across all datasets, highlighting its effectiveness. The Filter solution performs second best on average, closely followed by CAT-Net. The Masked-Convolution solution is far more resilient to distractions than the original TruFor model, with a decrease of only 7.12% on average. CAT-Net is still more resilient to distractions.
Remark:* Both TruFor and CAT-Net were trained on the IMD2020 dataset, which results in general higher robustness against distractions in this dataset.

5.4 Conclusion

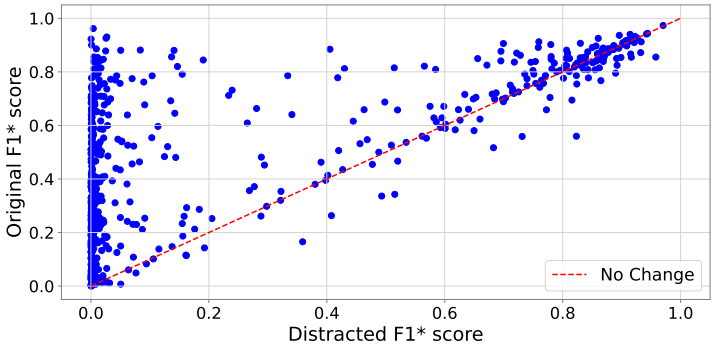
Two different solutions were proposed to solve the problem of forensic distractions: using a distraction filter and using masked convolutions. Both solutions were evaluated on four different datasets, with a **summary of the results and a comparison to all tested FL models presented in Table 5.4**. Figure 5.4 shows the scatter plots of the localization scores aggregated over all four datasets for the unmodified TruFor model, the adapter solution, and the masked-convolution solution.

The **masked-convolution** solution was found to be the **most effective** solution, with an average relative improvement of 87.11%. The final decrease in performance due to the addition of distractions is only 7.12%. This makes it the best performing and one of the most distraction-resilient FL models tested. However, the solution is not perfect. The decrease of 7.12% is non-negligible and highlights that other approaches should be explored to further improve the robustness of FL models against forensic distractions.

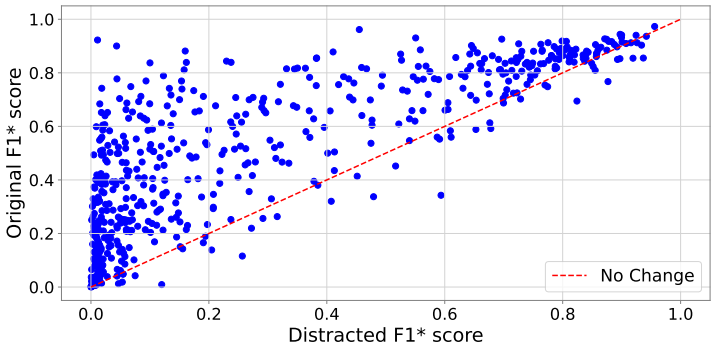
The **filter adapter** solution was also able to improve the performance of the TruFor model, however, to a **much lesser extent**, with an average relative improvement 16.90%, resulting in a decrease in performance due to distractions of 45.94%. The adapter solution is still a valid solution to the problem of forensic distractions, as it on average outperformed every other model tested, with the exception of the masked-convolution solution.

An ablation study was performed to evaluate the relative impact of the encoder and anomaly decoder on the performance. The results showed that the **encoder** was **responsible for** the majority of the **performance loss** when adding distractions, which explains the limited improvements of the adapter solution.

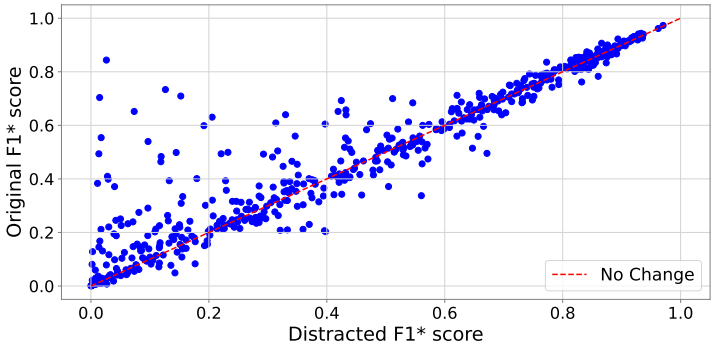
In **conclusion**, masked convolutions are a very effective solution to the problem of forensic distractions. In the future, they can be used to make any CNN-based FL method distraction-aware. Leading to performance improvements in realistic, noisy, and content-rich contexts, such as social media, blog posts, and news articles.



(a) Original unmodified TruFor [8]



(b) Filter adapter solution



(c) Masked-convolution solution

Figure 5.4: Scatter plots of the localization scores (F1* score (3.3)) of the unmodified TruFor model, the **adapter solution**, and the **masked-convolution solution** aggregated over all four datasets (DSO-1, VIPP, OpenForensics, and IMD2020).
It can be seen that for both modified models (b & c), the points shift to the right, indicating a higher resilience to distractions than the unmodified model (a). For the masked-convolution solution (c), the points are more clustered around the diagonal, indicating the model's performance is far less affected by the presence of forensic distractions.

6

Future Work

This thesis represents an **initial step** towards *distraction-aware detection* by proposing two methods, being a filter adapter (Section 4.3) and masked convolutions (Section 4.4), and demonstrating their effectiveness in improving the performance of distraction-sensitive FL models.

This work **opens up several options for future exploration**. Future work could focus on improving the performance and usability of the proposed methods, particularly through combining both methods, supporting arbitrarily-shaped masks, and automated mask generation. Moreover, the implementation of the masked-convolution method could be simplified by creating utility functions or wrapper classes.

Furthermore, the concept of distraction-aware detection can be extended beyond handling *explicit* distractions (i.e., logos, captions, etc.) to include *implicit* distractions, which are not immediately noticeable but still negatively impact detection performance. Additionally, the scope could be broadened to ignoring any irrelevant part of an image, useful in the cases of multiple forgeries where one manipulation is much more prominent than the other

By expanding both the technical capabilities and conceptual scope of distraction-aware detection, future work can help realize its full potential in real-world forensic scenarios.

6.1 Improving the Proposed Solutions

Combining Methods

The **masked-convolution method** shows significant improvements in the resilience to forensic distractions, but it is ***not*** **perfect**. Distractions still lead to an average performance decrease of 7.12%. The adapter method, while less impactful, still yielded non-negligible improvements, mainly in the VIPP (possibly due to the training on compressed images; see Section 5.2) and OpenForensics datasets.

A potential direction for future work could involve **combining both methods**, replacing all convolutions in the model with masked convolutions, and using an adapter to filter the feature maps before passing them to the decoders. By combining both methods, the adapter could learn to filter out the few remaining distractions that the masked convolutions leave untouched. In turn, increasing the performance even further.

Arbitrary Shaped Masks

The current masked convolution implementation is **limited to rectangular masks**. Extending this mechanism to support irregular shaped masks, as done by Liu et al. (2018) [65], could improve the usability of the method in real-world applications. Irregular masks would **allow for more precise targeting of distractions**, enabling the model to ignore complex shapes that are not easily encapsulated by rectangles (e.g., captions around a person's face). In general, the ability to define arbitrary masks would increase flexibility and applicability of the masked-convolution method, making it a more powerful tool for distraction-aware detection.

Automatic Mask Generation

Both **proposed solutions rely on user-provided masks** to indicate distracting regions. A logical next step would be to automate the mask generation process. Several strategies could be explored. A model could be made specifically for detecting distractions, or pretrained models could be used to identify and segment likely distractions.

Simplifying Implementation

The current implementation of the **masked-convolution method requires several modifications to the model's architecture**. Not only do components need to be replaced with the masked versions, but the mask itself also needs to be passed through the model. Moreover, due to the double input and output, masked convolutions cannot be used in the same way as standard convolutions. For example, the Sequential module in PyTorch [64] does not support this and thus needs to be dismantled (i.e., separating the layers).

Future work could focus on **simplifying this process** by developing utility functions or wrapper classes. For instance, a wrapper for *torch.nn.Sequential* that takes two inputs (i.e., the image and the mask) and passes the mask along to all relevant layers. This would also remove the need to remap the weights of all the layers due to the dismantling of the sequential module.

A more ambitious solution would be to create a full wrapper for *torch.nn* that allows all standard layers to accept (input, mask) tuples as input. This would enable distraction-aware models to be implemented with minimal effort. Enabling any existing CNN-based model, implemented using PyTorch, to be converted to a distraction-aware model by simply replacing the standard *torch.nn* import with the masked wrapper and changing the input from a single tensor to a tuple (input, mask).

6.2 Extending the Scope

Ignore Irrelevant Image Regions

The concept of distraction-aware detection can be extended to ignore any irrelevant part of the image. One potential application is in the detection of multiple forgeries. Some models struggle to detect multiple manipulations within a single image. By masking one of the manipulated regions, the model could focus on the other, improving detection accuracy.

Implicit Forensic Distractions

The distractions discussed in this thesis are clearly visible and can be easily distinguished from the rest of the image. Therefore, they can be classified as *explicit* distractions. However, in the real world, **many images also contain *implicit* distractions**, such as overexposed areas. It is not immediately clear to a viewer that these elements are problematic, yet they can still negatively impact detection performance.

As implicit distractions are not always obvious, they can be **difficult to identify and mask manually**. This reinforces the need for automatic mask generation. One possible approach is to generate masks based on the model's **confidence maps**. A confidence map is a heatmap that indicates the model's certainty about its predictions. High uncertainty areas could indicate the presence of implicit distractions. This does, however, require that models produce confidence maps alongside their anomaly maps. Some state-of-the-art models, such as TruFor, already do this, but this is the exception rather than the rule.

7

Societal Reflection

This chapter discusses the societal implications of the research presented in this thesis. It highlights the growing challenges of fake media and the need for reliable multimedia forensics methods to combat the spread of misinformation. Moreover, it emphasizes the potential benefits of the proposed solutions for various societal actors and positions the work within the context of ongoing international and national research.

Fake media detection is a critical area of research, especially in the context of the rapid advancements in image manipulation tools and generative AI technologies.

Tools such as Photoshop and GIMP make it possible to create high-quality fake images. These tools do require some level of expertise, which limits their accessibility to a certain extent. However, **the emergence of generative AI technologies**, have made it possible to create fake images via simple text prompts. This drops the barrier to entry significantly, allowing **anyone with access to these tools to create convincing fake images.** Moreover, in 2023, Photoshop introduced a generative fill feature giving users access to inpainting technologies right in the Photoshop interface, increasing the accessibility of these technologies even further.

There have been many cases of manipulated images being used for **malicious purposes**, such as spreading misinformation, committing fraud, and impersonating individuals, as discussed in Chapter 1. On a **larger scale**, the spread of misinformation can have **significant societal implications**. It can erode trust in institutions, polarize public opinion, and manipulate democratic processes.

Fake media is a driving force behind the spread of misinformation. As it becomes easier to create and share fake images, the danger of misinformation increases. This is particularly concerning in the context of social media, where fake images can go viral, reaching a large audience in a short amount of time. It is clear that developing reliable and robust multimedia forensics methods is crucial in the fight against misinformation.

This thesis proposes two methods to improve the robustness of FL models against distractions. The results of which have the potential to **benefit various societal actors** by enhancing media integrity and improving in-the-wild performance.

- Media and journalism organizations, such as VRT and the European Broadcasting Union, can leverage these improved tools to verify the authenticity of visual content before publication, thereby maintaining public trust and credibility.
- Fact-checking platforms, including Factcheck.Vlaanderen, Knack, and deCheckers, can use these technologies to systematically analyze and debunk false claims, combating disinformation and ensuring that accurate information reaches the public.
- Law enforcement agencies, such as the Federal Police, will be equipped with improved forensic tools to assess the authenticity of digital evidence, thus increasing the reliability of investigations.
- Cybersecurity institutions, like the European Union Agency for Cybersecurity (ENISA), can integrate these forensic methods into their frameworks to address the growing threat of identity fraud and manipulated media in financial and security-related scams.

- Lastly, social media platforms, being distraction-rich environments, can particularly benefit from these distraction-resilient technologies. Employing such methods can enhance the effectiveness of automated moderation systems, improving online safety and platform integrity.

This research **contributes to the ongoing international and national efforts** to combat misinformation and strengthen the reliability of digital media.

Internationally, the detection of fake media is a key focus in research. Projects like InVID, WeVerify, and vera.ai under Horizon Europe have resulted in practical tools such as the vera.ai verification plugin and MeVer's image verification assistant by ITI-CERTH. Standardization bodies, including C2PA and JPEG, are developing metadata-based solutions for media authentication, enabling the tracing of media content. Additionally, networks like the European Digital Media Observatory (EDMO) and its (Flemish-Dutch) BENEDMO hub bring together interdisciplinary expertise to address the problem of disinformation.

On a **national level**, the Flemish reliance plan provided €35M to battle disinformation. This led, among other things, to the creation of the **COM-PRESS Image Manipulation Analysis Dashboard** by IDLab at Ghent University, which aims to make image manipulation analysis accessible to a broad audience.

In conclusion, the research presented in this thesis addresses the growing challenges of fake media and misinformation. The proposed solutions enhance the robustness of FL models against distractions, potentially benefiting various societal actors and contributing to ongoing international and national efforts to combat misinformation.

8

Conclusion

Given the growing sophistication and accessibility of image manipulation tools, the creation of fake images has become easier than ever. This has led to an **increase in the number of fake images being shared online**, making it ever more important to develop reliable and robust multimedia forensics tools.

Despite the increasing performance of **state-of-the-art models**, they **still face several critical challenges**. One of these challenges is the limited robustness against post-processing operations. This thesis focussed on one common post-processing operation: the addition of benign visual elements, called forensic distractions.

Impact of forensic distractions

The first part of this thesis studied the impact of forensic distractions on both FL and SD. This was **never formally studied before**. The performed experiments and their results are discussed in Chapter 3 and an explanation of every evaluated model is given in Chapter 2.

Seven **FL models** were evaluated, of which four model-based and three deep-learning-based. The results show that forensic distractions **can significantly impact the performance** of FL models. In particular, TruFor is found to be highly sensitive to distractions, causing a decrease in performance of 55% on average. PSCC-Net is also affected, but to a lesser extent, with an average decrease of 25%. However, **not all FL models are affected**. CAT-Net is found to be quite robust against distractions, and the model-based methods barely show any sensitivity.

Eight **SD models** were evaluated, categorized into four groups. The results show that SD models are **generally robust** against forensic distractions, with most models maintaining similar performance. However, models based on local artifacts, such as PatchCraft and NPR, are impacted by distractions, although there is no consistent decrease in performance.

Proposed methods

As it was found that FL models and in particular TruFor, can be significantly affected by distractions, the second part of this thesis (Chapters 4 and 5) focused on improving the robustness of FL models against distractions. **Two methods are proposed**: the filter adapter and the masked convolutions. Both methods take a distraction cover as input, which indicates the parts of the image that likely contain distractions.

The **filter adapter** is a lightweight neural network placed between the encoder and decoder of the TruFor model. It is trained to alter the latent feature maps generated by the encoder based on a distraction cover, in such a way that the downstream decoders regain focus on the forgery.

The filter was **able to improve TruFor's performance** in the presence of distractions, with a relative increase of 17%. However, these **improvements are limited**. This is likely because the encoder discards crucial forensics information when a distraction is present, making it impossible for the filter to fully recover focus. This theory is supported by the results of an ablation study on the masked convolution.

Masked convolutions differ from standard convolutions by not convolving over the entire input, but instead only over the parts that are not covered by a provided mask. By replacing all standard

convolutions with masked convolutions, the influence of the covered regions on the detection result should be eliminated, meaning that the model can focus on the rest of the image.

The masked-convolution approach **significantly improves the performance** of TruFor in the presence of distractions, with a relative improvement of 87.11%. This approach is **able to outperform all tested FL models** and shows **high resilience to distractions**, with an average performance decrease of only 7.12% when adding distractions. These results highlight the effectiveness of the masked-convolutions approach as a method for creating distraction-aware FL models.

Future work

Future work (Chapter 6) could focus on **improving the proposed methods** by combining both methods, supporting arbitrarily-shaped masks, and automating distraction cover generation, removing the need to manually select distractions. The **concept of distractions could also be extended** to include other types of regions that distract FL models. For example, this could be useful for images with overexposed regions or multiple manipulations where one draws attention away from the other.

Conclusion

In conclusion, this work addresses the issue of forensic distractions in FL models. It demonstrates that **distractions can significantly impact the performance** of these models, and proposes two methods for improving their robustness. The **masked-convolution approach is particularly effective**, resulting in a significant increase in resilience to distractions. This work contributes to the ongoing efforts to develop reliable and robust multimedia forensics tools by proposing a solution to the problem of forensic distractions, **leading to improved performance in distraction-rich environments**, such as social media, blog posts, and news articles.

Bibliography

[1] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, 2022.

[2] Flux, "Flux [ai design tool]." `https://www.flux.ai/`, n.d. Accessed: 2025-05-18.

[3] Midjourney, "Midjourney [ai image generation tool]." `https://www.midjourney.com/`, 2023. Accessed: 2025-05-18.

[4] OpenAI, "Sora [ai video generation model]." `https://openai.com/sora`, 2024. Accessed: 2025-05-18.

[5] CNN, "Finance worker pays out $25 million after video call with deepfake 'chief financial officer'," February 2024.

[6] The Guardian, "Fraudsters editing vehicle photos to add fake damage in uk insurance scam," May 2024.

[7] D. Cozzolino and L. Verdoliva, "Noiseprint: A CNN-based camera model fingerprint," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 144–159, 2020.

[8] F. Guillaro, D. Cozzolino, A. Sud, N. Dufour, and L. Verdoliva, "TruFor: Leveraging all-round clues for trustworthy image forgery detection and localization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20606–20615, June 2023.

[9] D. Cozzolino, F. Marra, D. Gragnaniello, G. Poggi, and L. Verdoliva, "Combining PRNU and noiseprint for robust and efficient device source identification," *EURASIP Journal on Information Security*, vol. 2020, p. 1, Feb. 2020.

[10] H. Mareen, D. Vanden Bussche, F. Guillaro, D. Cozzolino, G. Van Wallendael, P. Lambert, and L. Verdoliva, "Comprint: Image forgery detection and localization using compression fingerprints," in *Pattern Recognition, Computer Vision, and Image Processing. ICPR 2022 International Workshops and Challenges* (J.-J. Rousseau and B. Kapralos, eds.), (Cham), pp. 281–299, Springer Nature Switzerland, 2023.

[11] S. Ye, Q. Sun, and E.-C. Chang, "Detecting digital image forgeries by measuring inconsistencies of blocking artifact," in *2007 IEEE International Conference on Multimedia and Expo*, pp. 12–15, 2007.

[12] Z. Lin, J. He, X. Tang, and C.-K. Tang, "Fast, automatic and fine-grained tampered jpeg image detection via dct coefficient analysis," *Pattern Recognition*, vol. 42, no. 11, pp. 2492–2501, 2009.

[13] W. Li, Y. Yuan, and N. Yu, "Passive detection of doctored jpeg image via block artifact grid extraction," *Signal Processing*, vol. 89, no. 9, pp. 1821–1829, 2009.

[14] C. Iakovidou, M. Zampoglou, S. Papadopoulos, and Y. Kompatsiaris, "Content-aware detection of jpeg grid inconsistencies for intuitive image forensics," *Journal of Visual Communication and Image Representation*, vol. 54, pp. 155–170, 2018.

[15] M.-J. Kwon, S.-H. Nam, I.-J. Yu, H.-K. Lee, and C. Kim, "Learning jpeg compression artifacts for image manipulation detection and localization," *International Journal of Computer Vision*, vol. 130, p. 1875–1895, May 2022.

[16] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Transactions on Image Processing*, vol. 26, p. 3142–3155, July 2017.

[17] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao, "Deep high-resolution representation learning for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3349–3364, 2021.

[18] X. Liu, Y. Liu, J. Chen, and X. Liu, "PSCC-Net: Progressive spatio-channel correlation network for image manipulation detection and localization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 11, pp. 7505–7517, 2022.

[19] J. Zhang, H. Liu, K. Yang, X. Hu, R. Liu, and R. Stiefelhagen, "CMX: Cross-modal fusion for rgb-x semantic segmentation with transformers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 12, pp. 14679–14694, 2023.

[20] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "SegFormer: Simple and efficient design for semantic segmentation with transformers," in *Advances in Neural Information Processing Systems* (M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, eds.), vol. 34, pp. 12077–12090, Curran Associates, Inc., 2021.

[21] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-Net: Fully convolutional neural networks for volumetric medical image segmentation," in *2016 Fourth International Conference on 3D Vision (3DV)*, pp. 565–571, 2016.

[22] V. V. Kniaz, V. Knyaz, and F. Remondino, "The point where reality meets fantasy: Mixed adversarial generators for image splice detection," in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[23] A. Novozámský, B. Mahdian, and S. Saic, "IMD2020: A large-scale annotated dataset tailored for detecting manipulated images," in *2020 IEEE Winter Applications of Computer Vision Workshops (WACVW)*, pp. 71–80, 2020.

[24] Z. Sha, Z. Li, N. Yu, and Y. Zhang, "DE-FAKE: Detection and attribution of fake images generated by text-to-image generation models," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, CCS '23, (New York, NY, USA), p. 3418–3432, Association for Computing Machinery, 2023.

[25] F. Marra, D. Gragnaniello, L. Verdoliva, and G. Poggi, "Do GANs leave artificial fingerprints?," in *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pp. 506–511, 2019.

[26] F. Lago, C. Pasquini, R. Böhme, H. Dumont, V. Goffaux, and G. Boato, "More real than real: A study on human visual perception of synthetic faces [applications corner]," *IEEE Signal Processing Magazine*, vol. 39, no. 1, pp. 109–116, 2022.

[27] E. J. Miller, B. A. Steward, Z. Witkower, C. A. M. Sutherland, E. G. Krumhuber, and A. Dawel, "Ai hyperrealism: Why ai faces are perceived as more real than human ones," *Psychological Science*, vol. 34, no. 12, pp. 1390–1403, 2023. PMID: 37955384.

[28] R. Corvi, D. Cozzolino, G. Zingarini, G. Poggi, K. Nagano, and L. Verdoliva, "On the detection of synthetic images generated by diffusion models," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2023.

[29] M. Schinas and S. Papadopoulos, "Sidbench: A python framework for reliably assessing synthetic image detection methods," in *Proceedings of the 3rd ACM International Workshop on Multimedia AI against Disinformation*, MAD '24, (New York, NY, USA), p. 55–64, Association for Computing Machinery, 2024.

[30] S.-Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros, "CNN-generated images are surprisingly easy to spot...for now," in *CVPR*, 2020.

[31] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *International Conference on Learning Representations (ICLR)*, 2018.

[32] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *International Conference on Learning Representations (ICLR)*, 2018.

[33] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8110–8119, 2020.

[34] P. Esser, R. Rombach, and B. Ommer, "Taming transformers for high-resolution image synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12873–12883, 2021.

[35] B. Dayma, S. Patil, P. Cuenca, K. Saifullah, T. Abraham, P. Lê Khâc, L. Melas, and R. Ghosh, "DALL-E mini," 2021. `https://github.com/borisdayma/dalle-mini`.

[36] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen, "GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models," *arXiv preprint arXiv:2112.10741*, 2021.

[37] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "Stable diffusion." `https://github.com/CompVis/stable-diffusion`, 2022.

[38] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," in *Advances in Neural Information Processing Systems* (M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, eds.), vol. 34, pp. 8780–8794, Curran Associates, Inc., 2021.

[39] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, 2022.

[40] Z. Liu, X. Qi, and P. H. Torr, "Global texture enhancement for fake face detection in the wild," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8057–8066, 2020.

[41] N. Zhong, Y. Xu, S. Li, Z. Qian, and X. Zhang, "PatchCraft: Exploring texture patch for efficient ai-generated image detection," 2024.

[42] C. Tan, Y. Zhao, S. Wei, G. Gu, P. Liu, and Y. Wei, "Rethinking the up-sampling operations in CNN-based generative network for generalizable deepfake detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 28130–28139, June 2024.

[43] Y. Ju, S. Jia, L. Ke, H. Xue, K. Nagano, and S. Lyu, "Fusing global and local features for generalized ai-synthesized image detection," in *2022 IEEE International Conference on Image Processing (ICIP)*, pp. 3465–3469, 2022.

[44] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *Proceedings of the 38th International Conference on Machine Learning* (M. Meila and T. Zhang, eds.), vol. 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763, PMLR, 18–24 Jul 2021.

[45] U. Ojha, Y. Li, and Y. J. Lee, "Towards universal fake image detectors that generalize across generative models," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24480–24489, 2023.

[46] C. Koutlis and S. Papadopoulos, "Leveraging representations from intermediate encoder-blocks for synthetic image detection," in *Computer Vision – ECCV 2024* (A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, and G. Varol, eds.), (Cham), pp. 394–411, Springer Nature Switzerland, 2025.

[47] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[48] T. U. of Notre Dame Computer Vision Research Laboratory (CVRL) and R. Hat, "Python-based image forgery detection toolkit." `https://github.com/EldritchJS/pyIFD`, 2022.

[49] T. J. de Carvalho, C. Riess, E. Angelopoulou, H. Pedrini, and A. de Rezende Rocha, "Exposing digital image forgeries by illumination color classification," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 7, pp. 1182–1194, 2013.

[50] M. Fontani, T. Bianchi, A. De Rosa, A. Piva, and M. Barni, "A framework for decision fusion in image forensics based on dempster–shafer theory of evidence," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 4, pp. 593–607, 2013.

[51] T.-N. Le, H. H. Nguyen, J. Yamagishi, and I. Echizen, "OpenForensics: Large-scale challenging dataset for multi-face forgery detection and segmentation in-the-wild," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10097–10107, 2021.

[52] H. Mareen, D. Karageorgiou, G. V. Wallendael, P. Lambert, and S. Papadopoulos, "TGIF: Text-guided inpainting forgery dataset," in *2024 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–6, 2024.

[53] Q. Bammey, "Synthbuster: Towards detection of diffusion model generated images," *IEEE Open Journal of Signal Processing*, vol. 5, pp. 1–9, 2024.

[54] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato, "RAISE: a raw images dataset for digital image forensics," in *Proceedings of the 6th ACM Multimedia Systems Conference*, MMSys '15, (New York, NY, USA), p. 219–224, Association for Computing Machinery, 2015.

[55] J.-S. Lee and J. Hsiang, "Patent classification by fine-tuning bert language model," *World Patent Information*, vol. 61, p. 101965, 2020.

[56] Y. Chen, Q. Zhang, Y. Wu, B. Liu, M. Wang, and Y. Lin, "Fine-tuning resnet for breast cancer classification from mammography," in *Proceedings of the 2nd International Conference on Healthcare Science and Engineering* (C. Q. Wu, M.-C. Chyu, J. Lloret, and X. Li, eds.), (Singapore), pp. 83–96, Springer Singapore, 2019.

[57] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for NLP," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799, PMLR, 09–15 Jun 2019.

[58] D. Yin, L. Hu, B. Li, and Y. Zhang, "Adapter is all you need for tuning visual tasks," 2023.

[59] X. Cui, Y. Li, A. Luo, J. Zhou, and J. Dong, "Forensics adapter: Adapting clip for generalizable face forgery detection," 2025.

[60] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[61] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[62] F. Laakom, K. Chumachenko, J. Raitoharju, A. Iosifidis, and M. Gabbouj, "Learning to ignore: rethinking attention in cnns," 2021.

[63] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[64] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, *PyTorch: an imperative style, high-performance deep learning library*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[65] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image inpainting for irregular holes using partial convolutions," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

A

Supplementary Study: Effectiveness of Distraction Types

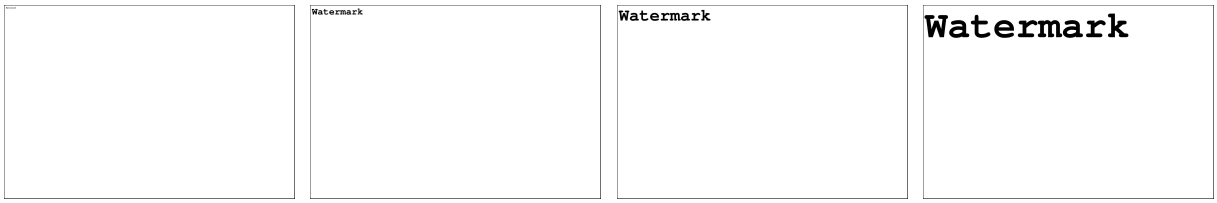
To better understand what makes distractions more or less effective at disrupting the performance of Forgery Localization (FL) models, a focused study was conducted on the DSO-1 [49] dataset using the TruFor [8] model. This study investigates how specific properties of distractions, such as text size, font type, boldness, position, and graphical content, affect the model's localization accuracy.

A.1 Textual Distractions

Effect of Text Size

The impact of varying text sizes was evaluated by applying the same textual distraction at multiple font sizes. Figure A.2 shows how the average F1 score decreases with increasing text size. The larger the text, the more it disrupts the model's performance. However, the effect appears to saturate at larger sizes, leading to the performance to slightly increase again. Figure A.1 provides examples of the text sizes used in the experiment.

Conclusion: Larger textual distractions are more disruptive to detection performance, but the effect levels off at larger sizes.



(a) 10 (b) 50 (c) 90 (d) 200

Figure A.1: Examples of distraction at different text sizes. The numbers indicate the font size in pixels, and the images were 1536x1024 pixels.

Effect of Font Type and Text Weight

To evaluate the impact of different font types and weights, three font types (Free Mono, High School USA Serif, and Moonkids). Each font size was chosen to cover about the same area in the image. Figure A.3 shows the font used in the experiment. Table A.1 shows the average F1* score (as defined in Section 3.2.1) for each font type and weight combination. The results indicate that the denser the font, the more distracting it is. High School USA Serif is more spread out than Free Mono, which is more spread out than Moonkids.

Conclusion: The denser the font, the more distracting it is. Density increases with font weight and is different for each font type.

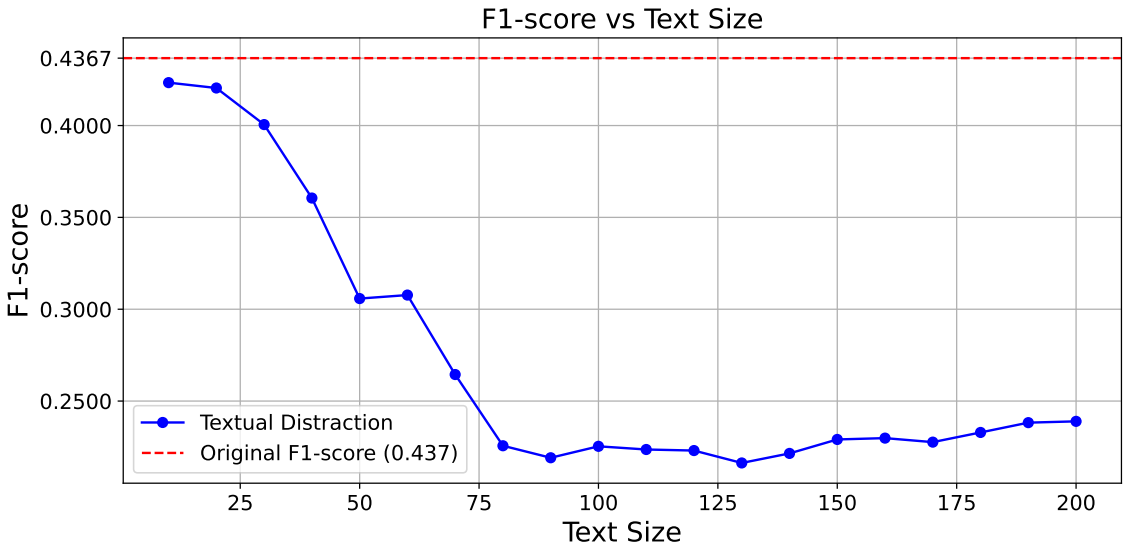
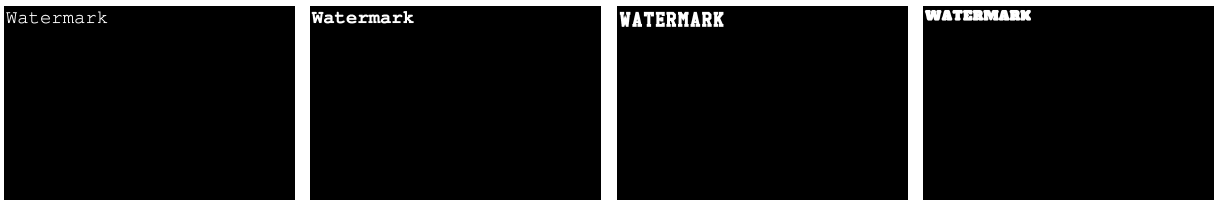


Figure A.2: Impact of distraction text size on the performance of the TruFor [8] model on the DSO-1 [49] dataset. The average F1* score (as defined in Section 3.2.1) is plotted against varying text sizes, with the original detection score (i.e., without any distractions) represented by the dashed red line. Larger text sizes lead to a greater reduction in performance, with the effect saturating beyond a certain size.



(a) Free Mono 100 default (b) Free Mono 100 bold (c) High School USA Serif 100 default (d) Moonkids 80 default

Figure A.3: Examples of distraction with different fonts and weights. The numbers indicate the font size in pixels, and the images were 1536x1024 pixels.

	Font	**Font Size**	**Font Weight**	**F1* score**	**% Decrease**
		Original	Distracted	Distracted	Improvement
No Distraction		N/A	N/A	0.43670	N/A
Free Mono		100	Bold	0.22460	48.57%
High School USA Serif		100	Default	0.22929	47.49%
Moonkids		80	Default	0.20192	53.76%
Free Mono		100	Default	0.33677	22.87%
Free Mono		100	Bold	0.22460	48.57%

Table A.1: Impact of font type and weight on the performance of the TruFor [8] model on the DSO-1 [49] dataset, measured. The average F1* score (as defined in Section 3.2.1) is shown for each font type and weight combination. The original detection score (i.e., without any distractions) is represented by the first row. Rows 2-4 show the impact of 3 different font types (Free Mono, High School USA Serif, and Moonkids), and rows 5-6 show the impact of font weight (default vs. bold) on the Free Mono font.

A.2 Graphical Distractions

Effect of Shape and Color

The experiment in Section 2.1 was performed on ten different logos, each with a different shape and color. The results from that experiment are aggregated per logo in Table A.2. As a reminder, the used logos are shown in Figure A.4.

Conclusion: Some logos are more effective than others, with the hexagon logo being the most effective and the DMK logo being the least effective. No clear pattern emerges regarding the impact of shape or color on the effectiveness of the logos.

Logo	# occurrences	F1* score		
		Original	Distracted	% Decrease
A	38	0.43398	0.18148	58.18%
snake	56	0.49447	0.29847	39.64%
h	49	0.54060	0.22333	58.69%
SH	55	0.49393	0.26412	46.53%
DMK	75	0.54447	0.35148	35.45%
bird	54	0.47389	0.21633	54.35%
hexagon	75	0.51790	0.19019	63.28%
sunrise	59	0.55011	0.31103	43.46%
toekan	67	0.47155	0.25081	46.81%
company	34	0.52877	0.31849	39.77%

Table A.2: Impact of logo shape and color on the performance of the TruFor [8] model measured using F1*score as defined in Section 3.2.1. Results are from the experiment described in Section 2.1. Different logos appeared a different number of times, as indicated in the table. Some logos are more effective than others, with the hexagon logo being the most effective and the DMK logo being the least effective. No clear pattern emerges regarding the shape or color of the logos.



(a) A (b) Snake (c) H (d) SH (e) DMK

(f) Bird (g) Hexagon (h) Sunrise (i) Toekan (j) Company

Figure A.4: Overview of logos used to study the impact of shape and color (see Table A.2). The logos are shown in the same order as in the table.

Effect of Position

The experiment in Section 2.1 was performed by placing logos in four different positions: upper left, upper right, lower left, and lower right. The results from that experiment are aggregated per position in

Table A.3.

Conclusion: Position does influence the effectiveness of a distraction, with the upper left and lower right positions being the most effective.

Position	**# occurrences**	**F1* score**		
		Original	Distracted	% Decrease
Upper left	136	0.50073	0.23453	53.16%
Upper right	125	0.51809	0.29399	43.26%
Lower left	126	0.54754	0.30634	44.05%
Lower right	175	0.47632	0.22930	51.86%

Table A.3: Impact of logo position on the performance of the TruFor [8] model measured using F1*score as defined in Section 3.2.1. Results are from the experiment described in Section 2.1. Different positions where used a different number of times, as indicated in the table. Position does influence the effectiveness of a distraction, with the upper left and lower right positions being the most effective.

Summary

In summary, the findings suggest that larger and bolder/denser textual distractions are more disruptive to FL models, although the effect levels off past a certain size.
Some graphical distractions are more effective than others, but no clear pattern was found regarding the impact of shape or color. Position also influences the effectiveness of a distraction, with the upper left and lower right positions leading to the most effective distractions.

B

Forensic Distractions on FL Experiment results

Appendix B contains all results from the performed experiment, discussed in Section 3.2. This experiment studied the impact of forensic distractions on different FL models. The results are presented in the form of scatter plots and percentage plots, showcasing the performance of the models under forensic distractions.

Following is a list of of scatter- and percentage plots in this appendix:

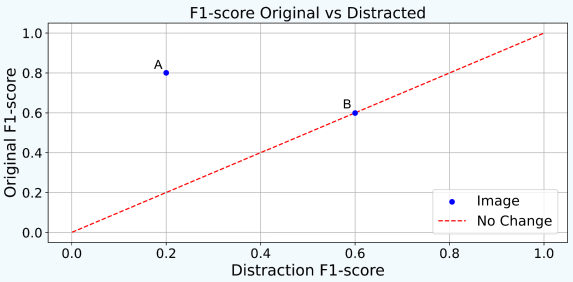
- DCT [11]: Scatter plots in Figure B.1 and percentage plots in Figure B.2.
- ADQ1 [12]: Scatter plots in Figure B.3 and percentage plots in Figure B.4.
- BLK [13]: Scatter plots in Figure B.5 and percentage plots in Figure B.6.
- CAGI [14]: Scatter plots in Figure B.7 and percentage plots in Figure B.8.
- TruFor [8]: Scatter plots in Figure B.9 and percentage plots in Figure B.10.
- PSCC-Net [18]: Scatter plots in Figure B.11 and percentage plots in Figure B.12.
- CAT-Net [15]: Scatter plots in Figure B.13 and percentage plots in Figure B.14.

Next, to the scatter plots and percentage plots, examples of manipulated images with corresponding predictions are shown for TruFor in Figure 3.6, for CAT-Net in Figure B.16, and for PSCC-Net in Figure B.17.

How to interpret scatter plots

Each dot in the scatter plot represents a single image. The y-axis indicates the localization score before a distraction was applied. The x-axis shows the score after the distraction was applied. The dashed-red line shows y=x or before equals after, meaning the distraction did not influence the detection. If a point lies to the left of the red line, the detection score for that image decreased by adding the distraction.

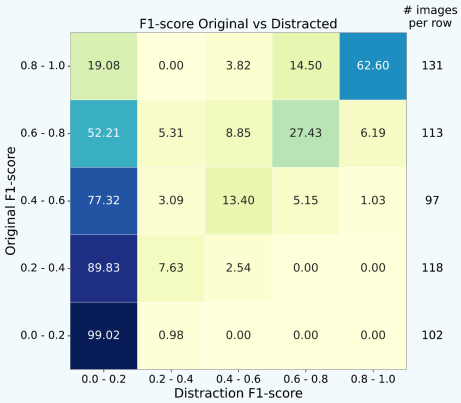
Example:
Image A's localization score drops from 0.8 to 0.2 after adding the distraction, indicating that the distraction had a significant impact on the detection process. Image B's score remains 0.6, showing the distraction did not influence the detection process.

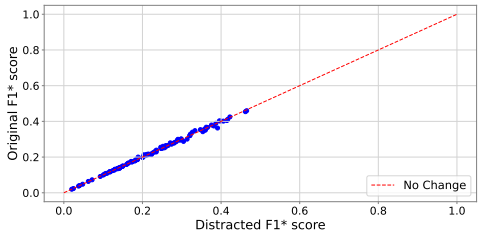


How to interpret percentage plots

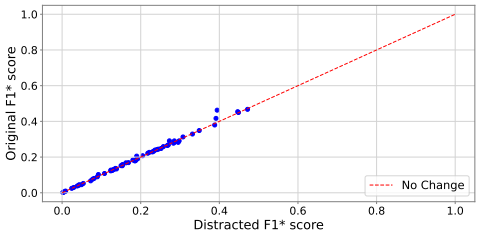
Percentage plots are a bucketed version of the scatter plots. Each row in the plot shows a bucket of images with a similar localization score, before any distraction are added. The columns of every row add up to 100% and show what percentage of these images after a distraction is added have a localization score, part of that bucket. The total number of images in each row is shown on the right.

Example:
131 of the 561 (=131+113+97+118+102) images had a localization score of 0.8 or higher before any distractions were added. After adding distractions, 19.08% (or 25) of these images have a localization score of 0.2 or lower. While 62.20% (or 82) of these images kept a localization score of 0.6 or higher.

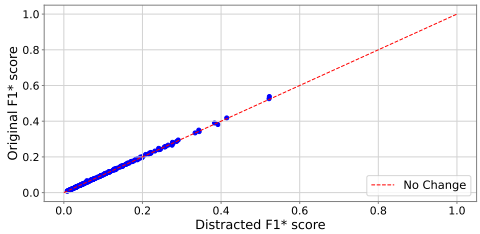




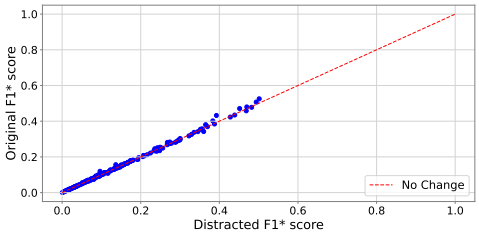
(a) DSO-1 datasets



(b) VIPP datasets

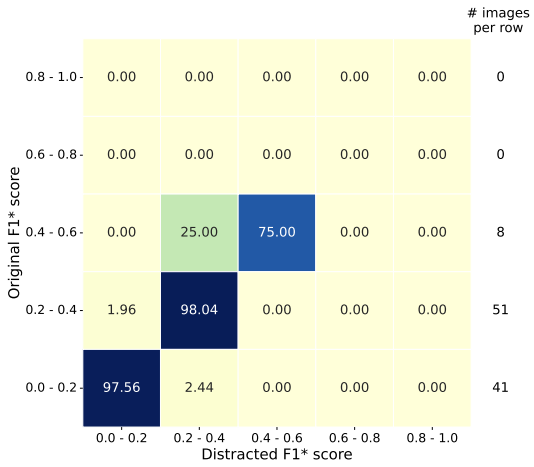


(c) OpenForensics datasets

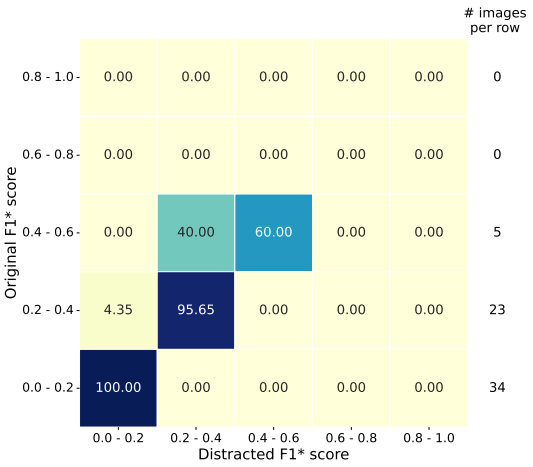


(d) IMD2020 datasets

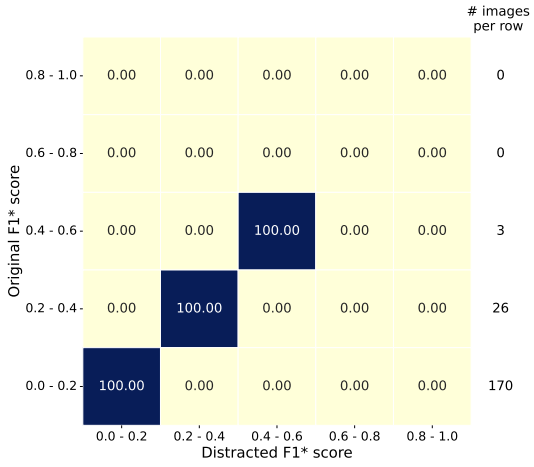
Figure B.1: Scatter plots of localization scores (F1* scores as defined in Section 3.2.1) of **DCT** [11] on different datasets



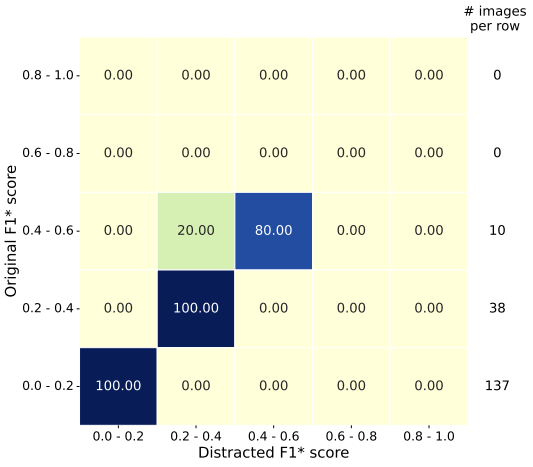
(a) DSO-1 datasets



(b) VIPP datasets

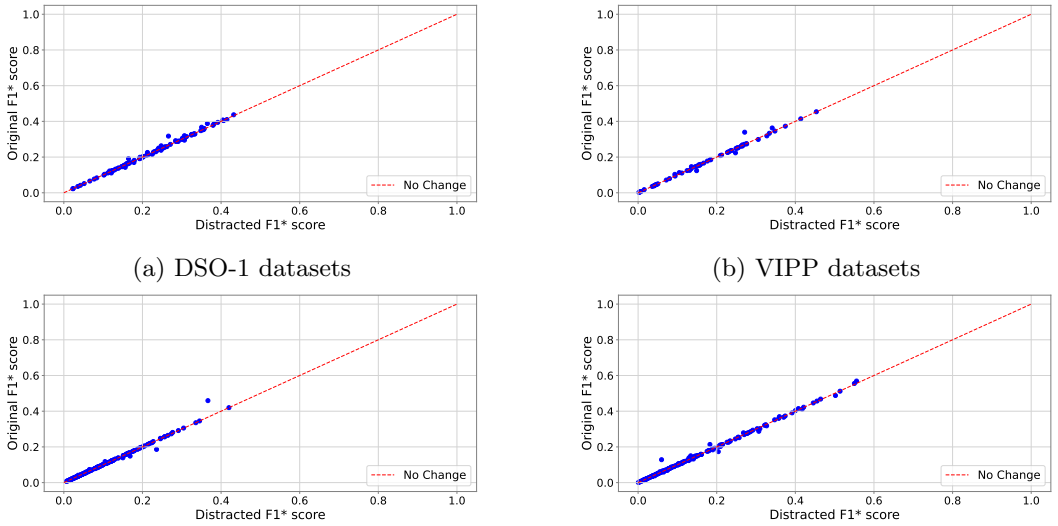


(c) OpenForensics datasets



(d) IMD2020 datasets

Figure B.2: Percentage plots of localization scores (F1* scores as defined in Section 3.2.1) of **DCT** [11] on different datasets



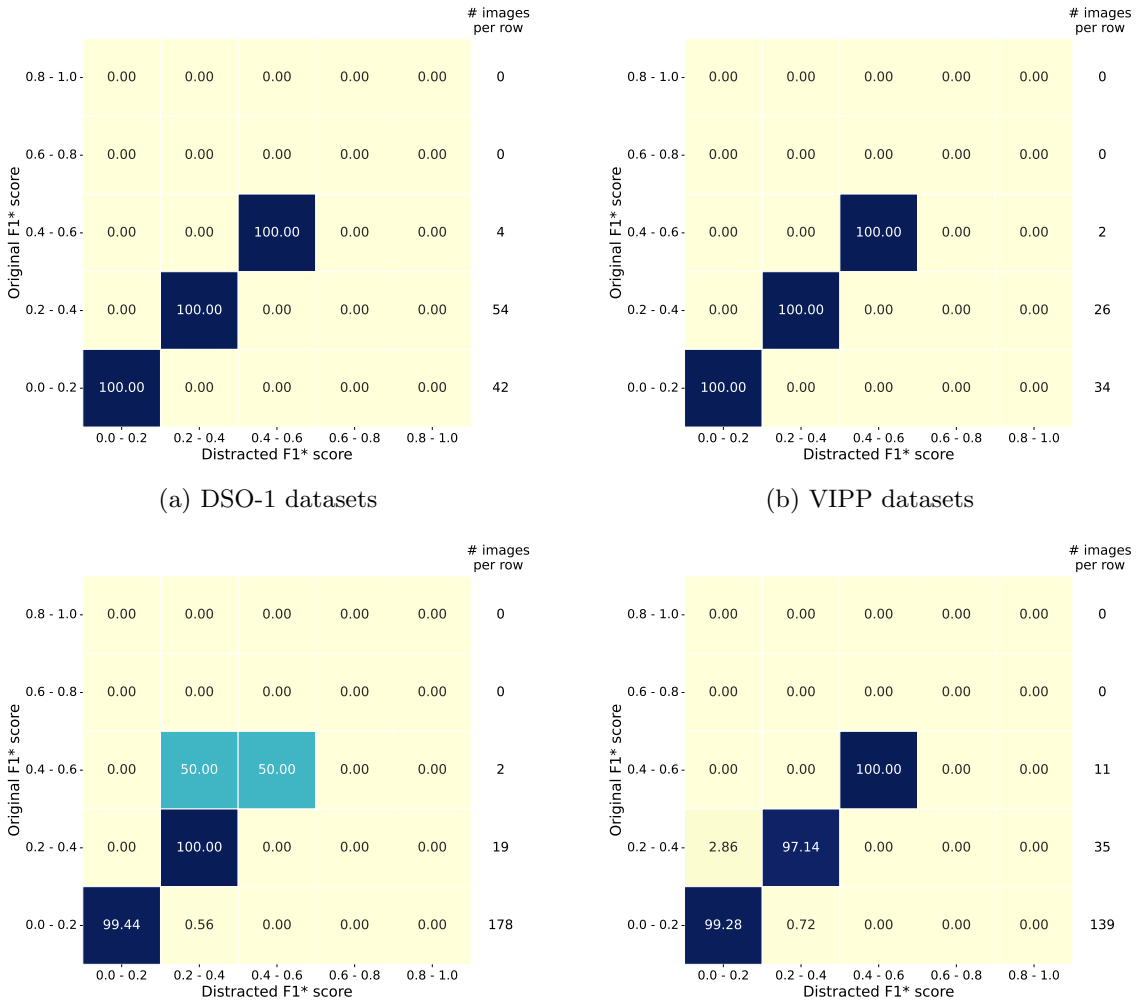
(a) DSO-1 datasets

(b) VIPP datasets

(c) OpenForensics datasets

(d) IMD2020 datasets

Figure B.3: Scatter plots of localization scores (F1* scores as defined in Section 3.2.1) of **ADQ1** [12] on different datasets



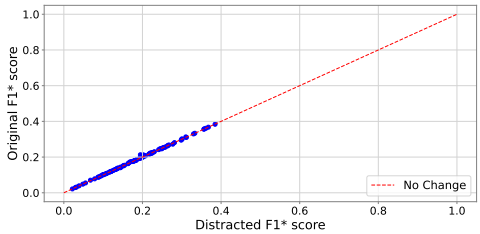
(a) DSO-1 datasets

(b) VIPP datasets

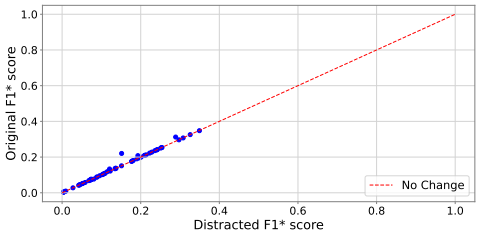
(c) OpenForensics datasets

(d) IMD2020 datasets

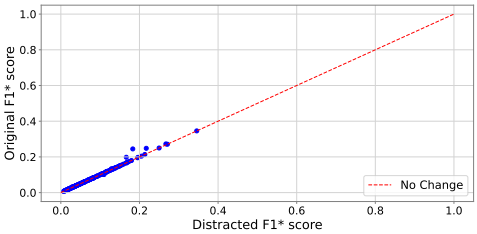
Figure B.4: Percentage plots of localization scores (F1* scores as defined in Section 3.2.1) of **ADQ1** [12] on different datasets



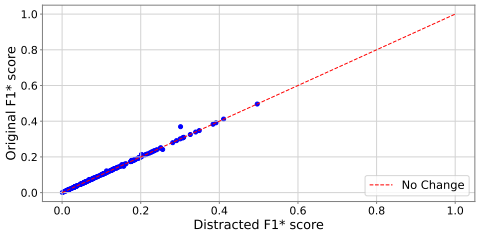
(a) DSO-1 datasets



(b) VIPP datasets

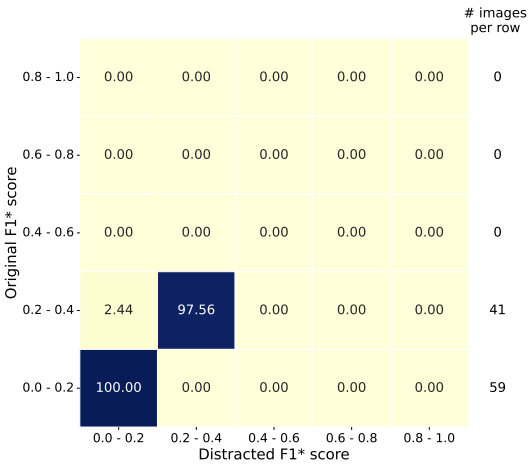


(c) OpenForensics datasets

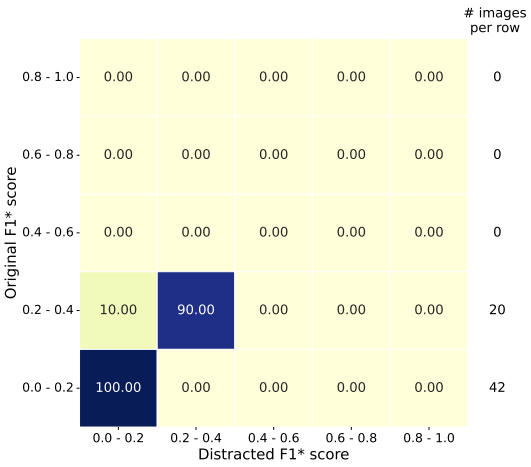


(d) IMD2020 datasets

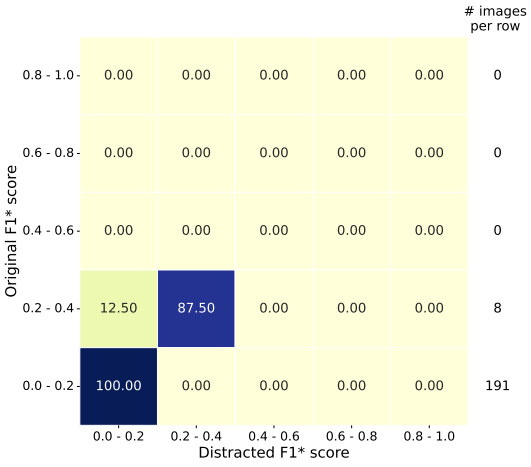
Figure B.5: Scatter plots of localization scores (F1* scores as defined in Section 3.2.1) of **BLK** [13] on different datasets



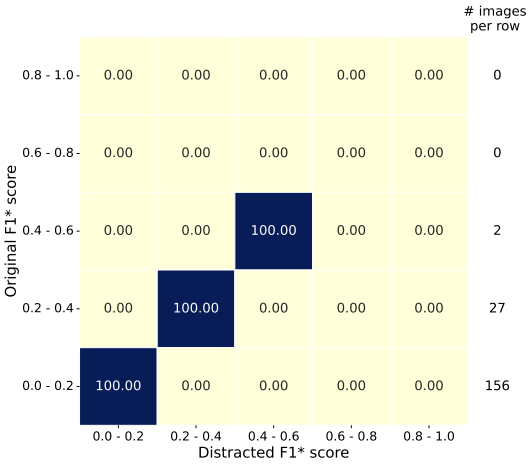
(a) DSO-1 datasets



(b) VIPP datasets

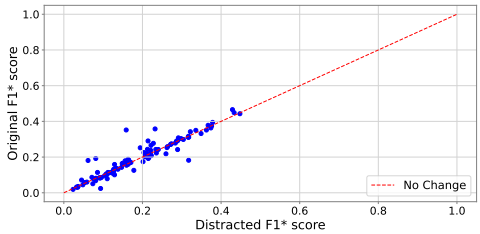


(c) OpenForensics datasets

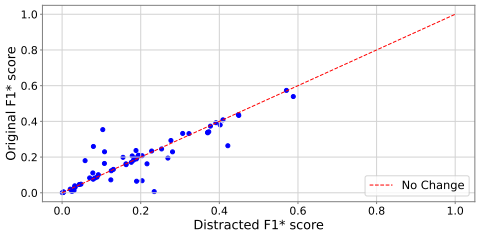


(d) IMD2020 datasets

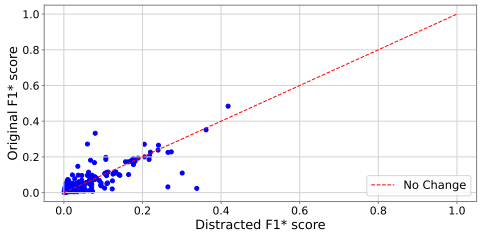
Figure B.6: Percentage plots of localization scores (F1* scores as defined in Section 3.2.1) of **BLK** [13] on different datasets



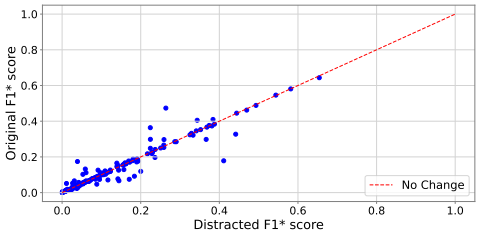
(a) DSO-1 datasets



(b) VIPP datasets

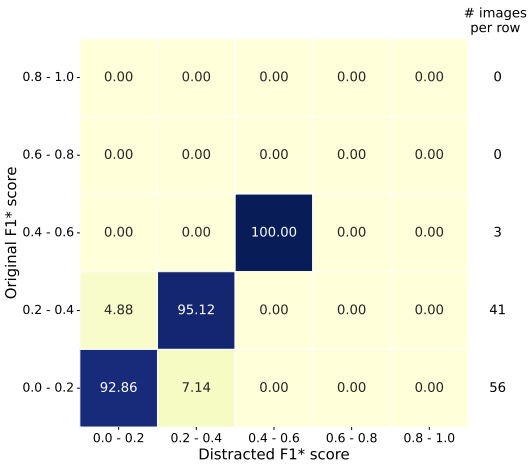


(c) OpenForensics datasets

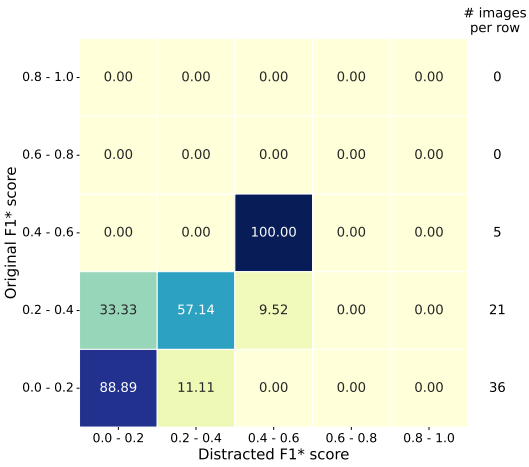


(d) IMD2020 datasets

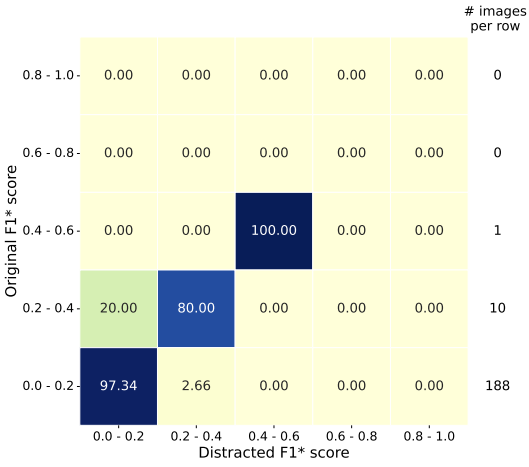
Figure B.7: Scatter plots of localization scores (F1* scores as defined in Section 3.2.1) of **CAGI** [14] on different datasets



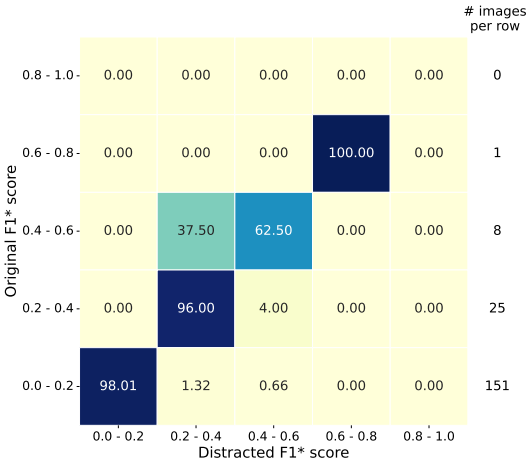
(a) DSO-1 datasets



(b) VIPP datasets

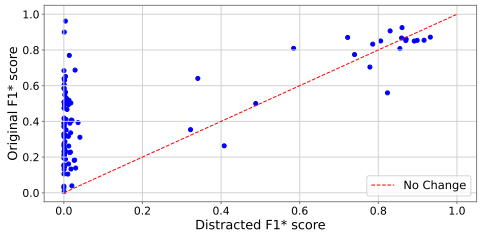


(c) OpenForensics datasets

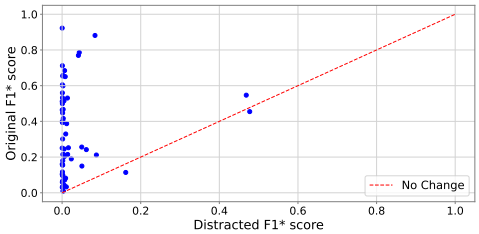


(d) IMD2020 datasets

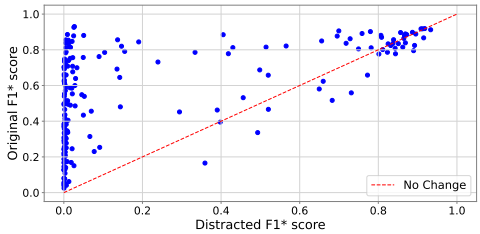
Figure B.8: Percentage plots of localization scores (F1* scores as defined in Section 3.2.1) of **CAGI** [14] on different datasets



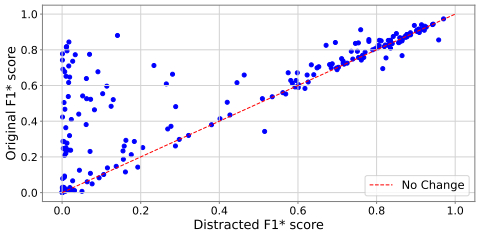
(a) DSO-1 datasets



(b) VIPP datasets

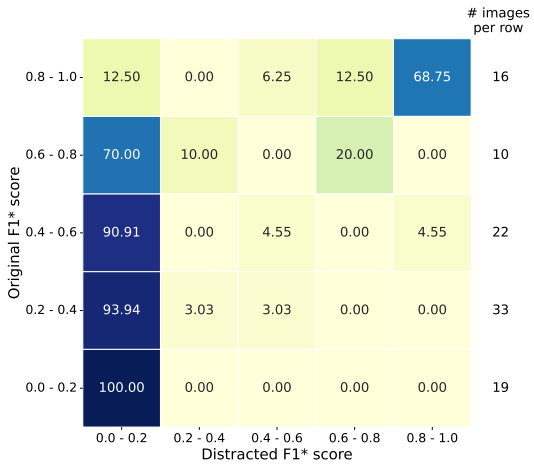


(c) OpenForensics datasets

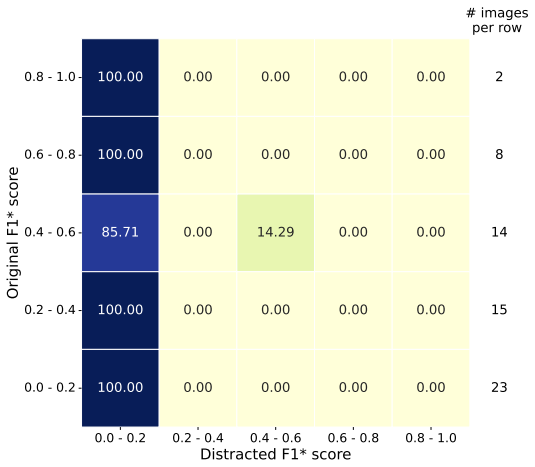


(d) IMD2020 datasets

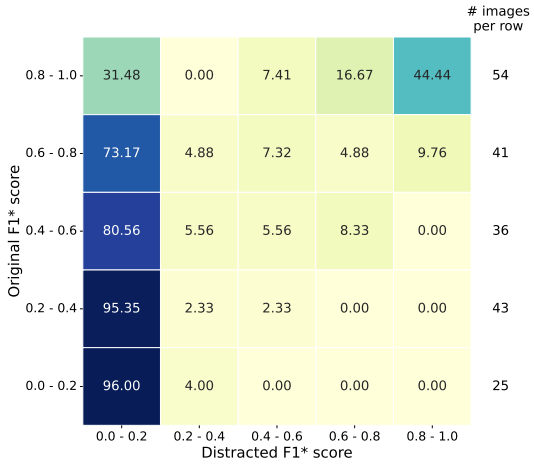
Figure B.9: Scatter plots of localization scores (F1* scores as defined in Section 3.2.1) of **TruFor** [8] on different datasets



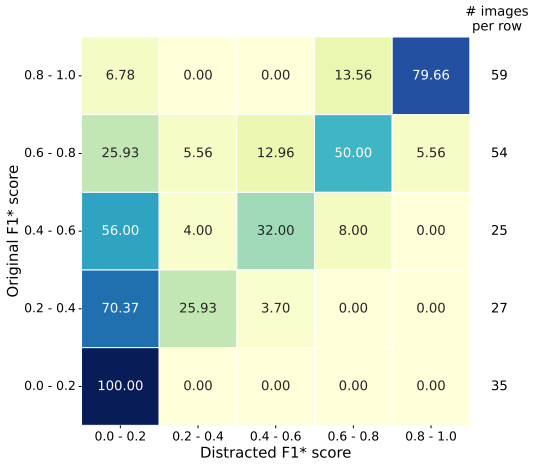
(a) DSO-1 datasets



(b) VIPP datasets

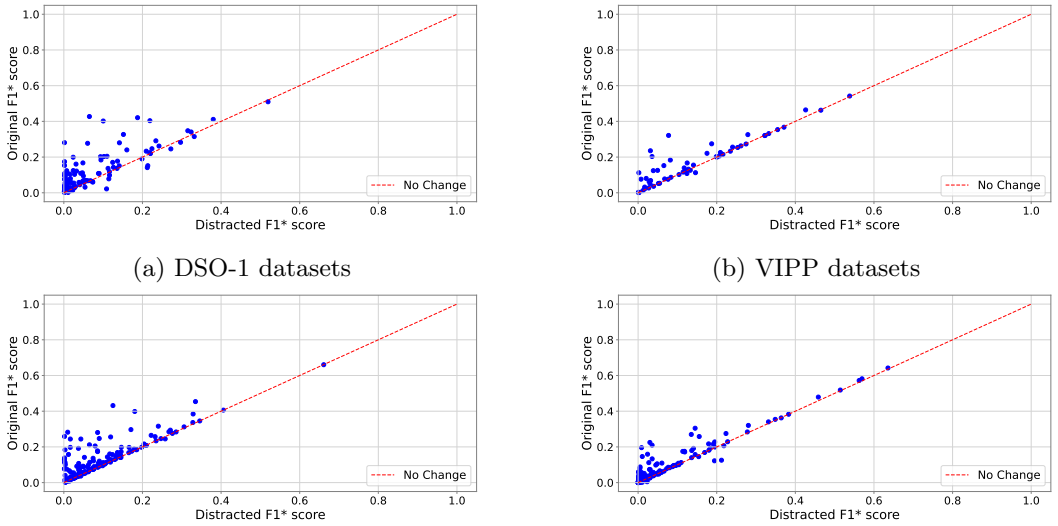


(c) OpenForensics datasets



(d) IMD2020 datasets

Figure B.10: Percentage plots of localization scores (F1* scores as defined in Section 3.2.1) of **TruFor** [8] on different datasets



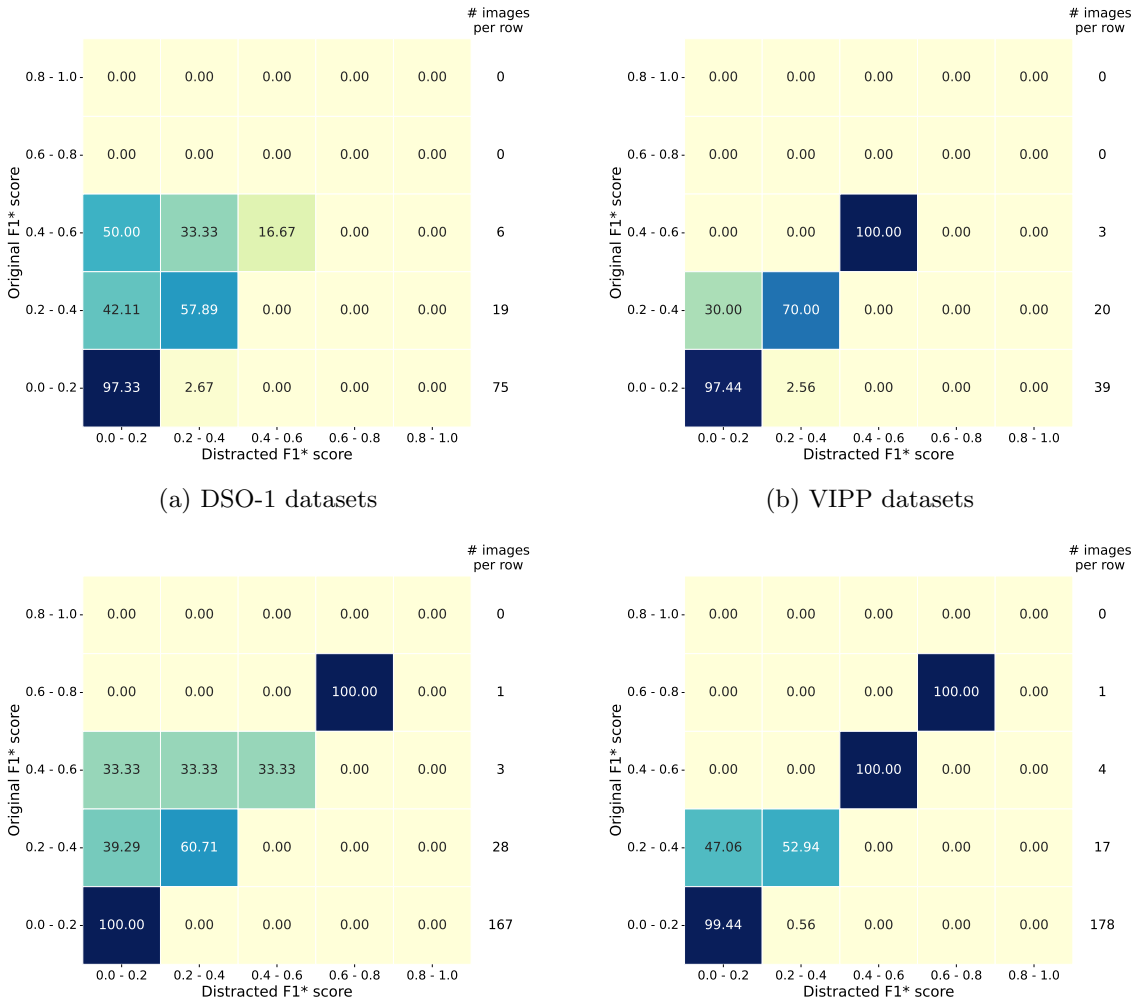
(a) DSO-1 datasets

(b) VIPP datasets

(c) OpenForensics datasets

(d) IMD2020 datasets

Figure B.11: Scatter plots of localization scores (F1* scores as defined in Section 3.2.1) of **PSCC-Net** [18] on different datasets



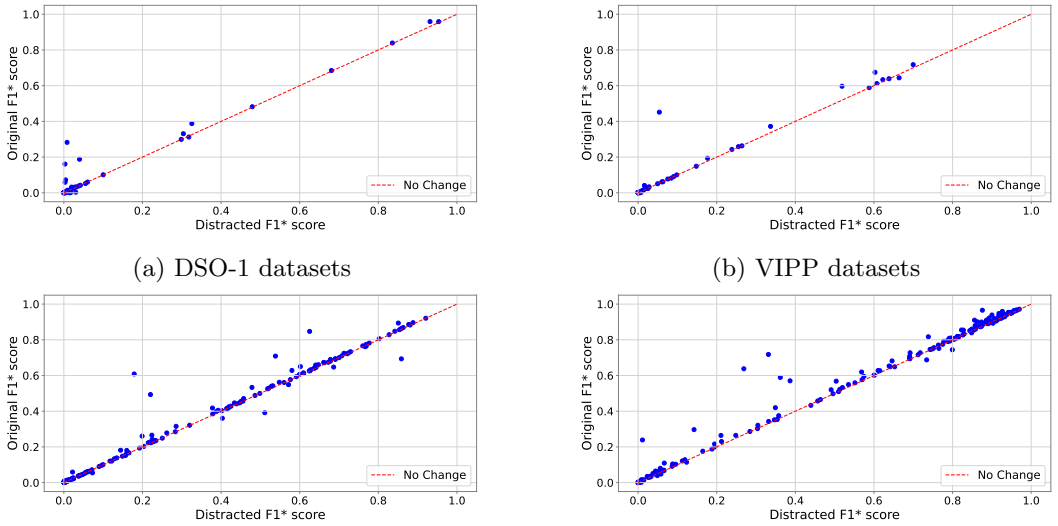
(a) DSO-1 datasets

(b) VIPP datasets

(c) OpenForensics datasets

(d) IMD2020 datasets

Figure B.12: Percentage plots of localization scores (F1* scores as defined in Section 3.2.1) of **PSCC-Net** [18] on different datasets



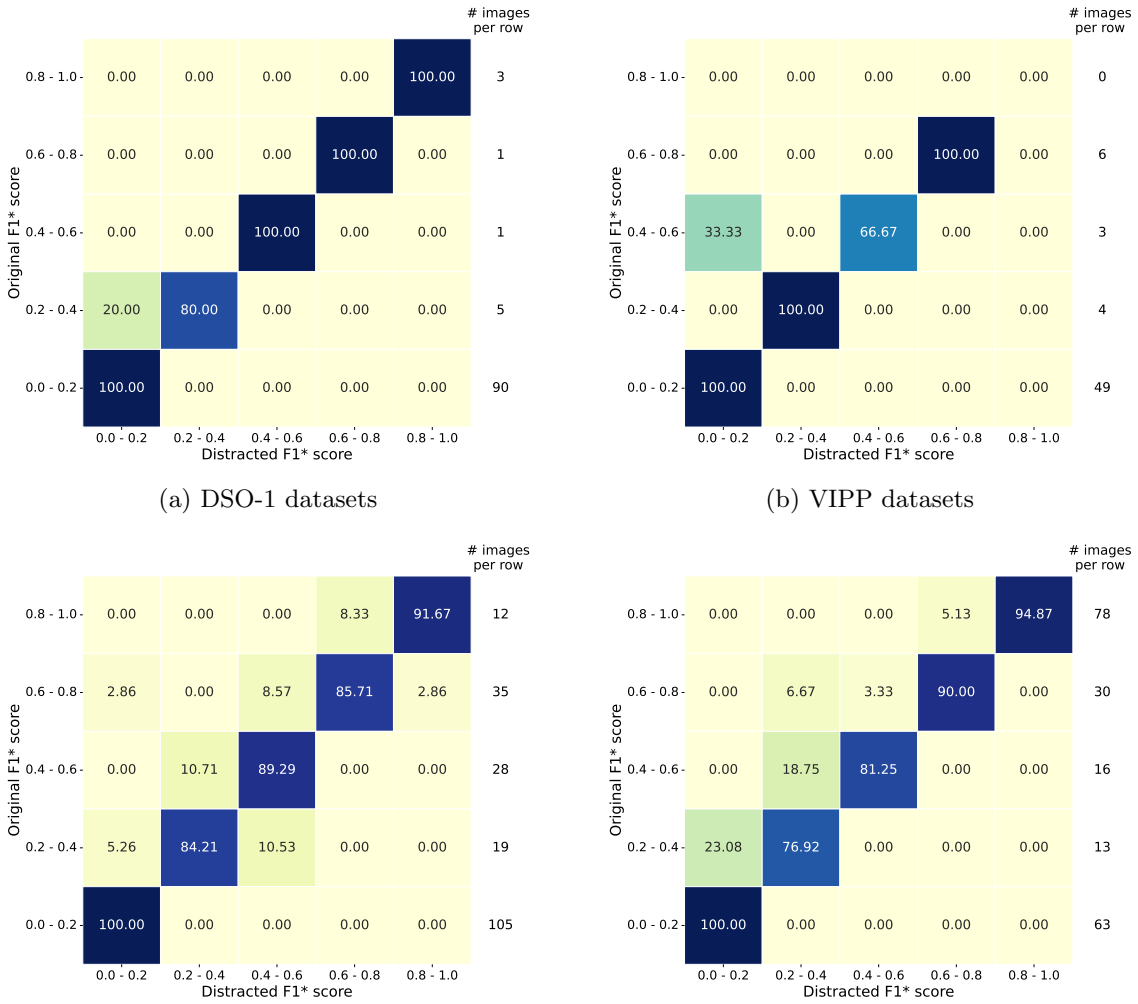
(a) DSO-1 datasets

(b) VIPP datasets

(c) OpenForensics datasets

(d) IMD2020 datasets

Figure B.13: Scatter plots of localization scores (F1* scores as defined in Section 3.2.1) of **CAT-Net** [15] on different datasets



(a) DSO-1 datasets

(b) VIPP datasets

(c) OpenForensics datasets

(d) IMD2020 datasets

Figure B.14: Percentage plots of localization scores (F1* scores as defined in Section 3.2.1) of **CAT-Net** [15] on different datasets



(a) Manipulated image (1), with distraction in lower-right corner



(b) Manipulated image (2), with distraction in lower-left corner



(c) Manipulated image (3), with distraction in upper-left corner



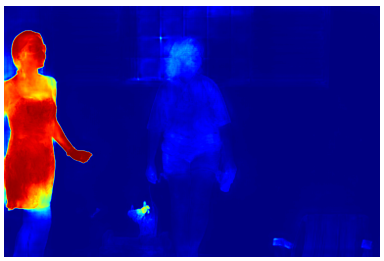
(d) Ground truth mask (1)



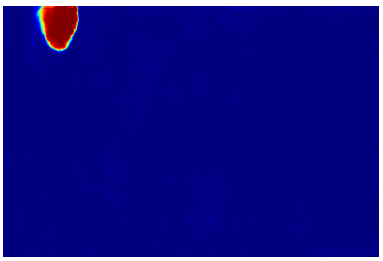
(e) Ground truth mask (2)



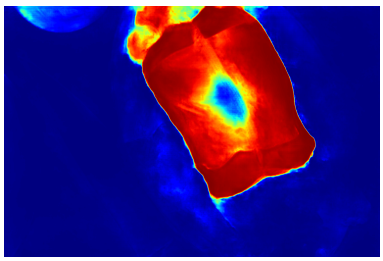
(f) Ground truth mask (3)



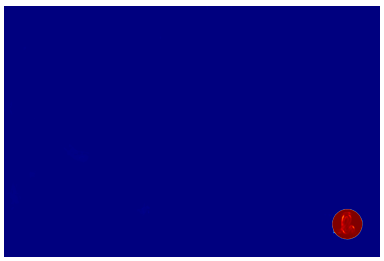
(g) Prediction *without* distraction (1), F1* score: 0.7696



(h) Prediction *without* distraction (2), F1* score: 0.7850



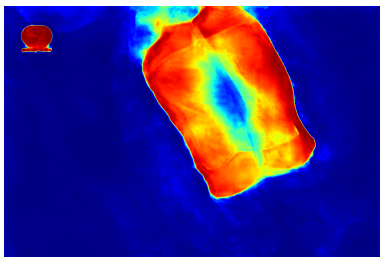
(i) Prediction *without* distraction (3), F1* score: 0.8654



(j) Prediction *with* distraction (1), F1* score: 0.0028



(k) Prediction *with* distraction (2), F1* score: 0.1305



(l) Prediction *with* distraction (3), F1* score: 0.7889

Figure B.15: Examples of manipulated images with corresponding **TruFor** [8] predictions and F1* scores (3.3), without (3th row) and with (4th row) distractions. It can be seen that the impact of distractions on TruFor's performance varies across images.



(a) Manipulated image (1), with distraction in lower-right corner



(b) Manipulated image (2), with distraction in lower-left corner



(c) Manipulated image (3), with distraction in upper-left corner



(d) Ground truth mask (1)



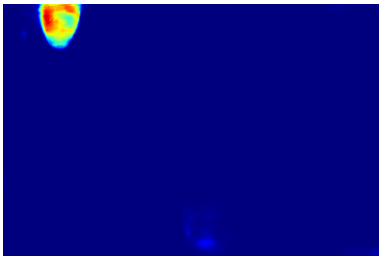
(e) Ground truth mask (2)



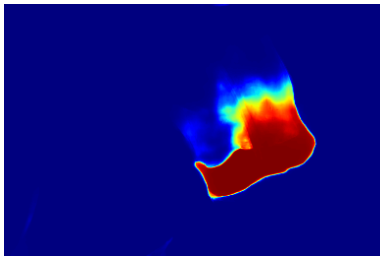
(f) Ground truth mask (3)



(g) Prediction *without* distraction (1), F1* score: 0.0008



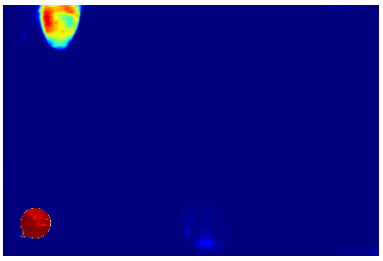
(h) Prediction *without* distraction (2), F1* score: 0.6460



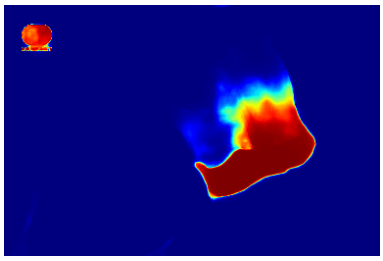
(i) Prediction *without* distraction (3), F1* score: 0.4766



(j) Prediction *with* distraction (1), F1* score: 0.0008



(k) Prediction *with* distraction (2), F1* score: 0.6458



(l) Prediction *with* distraction (3), F1* score: 0.4765

Figure B.16: Examples of manipulated images with corresponding **CAT-Net** [15] predictions and F1* scores (3.3), without (3th row) and with (4th row) distractions. It can be seen that CAT-Net is quite robust to distractions, retaining a similar F1* score. However, the general performance is worse than TruFor.



(a) Manipulated image (1), with distraction in lower-right corner



(b) Manipulated image (2), with distraction in lower-left corner



(c) Manipulated image (3), with distraction in upper-left corner



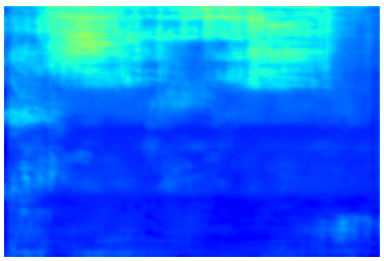
(d) Ground truth mask (1)



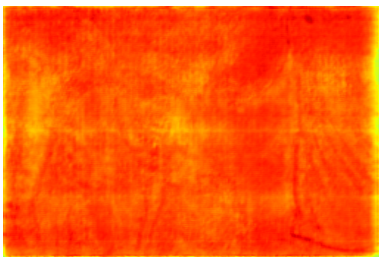
(e) Ground truth mask (2)



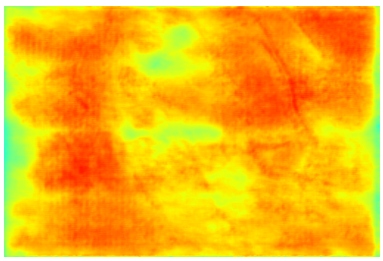
(f) Ground truth mask (3)



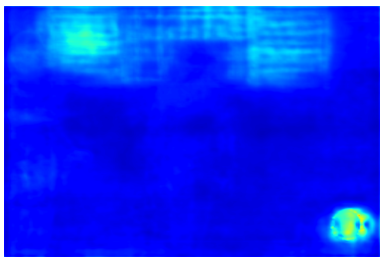
(g) Prediction *without* distraction (1), F1* score: 0.1429



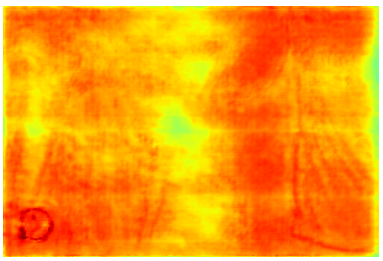
(h) Prediction *without* distraction (2), F1* score: 0.0381



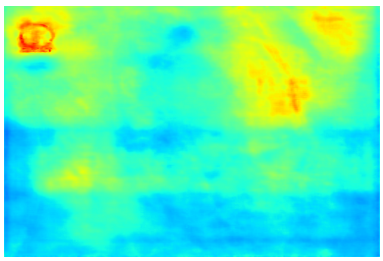
(i) Prediction *without* distraction (3), F1* score: 0.3392



(j) Prediction *with* distraction (1), F1* score: 0.1158



(k) Prediction *with* distraction (2), F1* score: 0.0386



(l) Prediction *with* distraction (3), F1* score: 0.3202

Figure B.17: Examples of manipulated images with corresponding **PSCC-Net** [18] predictions and F1* scores (3.3), without (3th row) and with (4th row) distractions. It can be seen that PSCC-Net performs very poorly on the tested images, both before and after adding distractions.

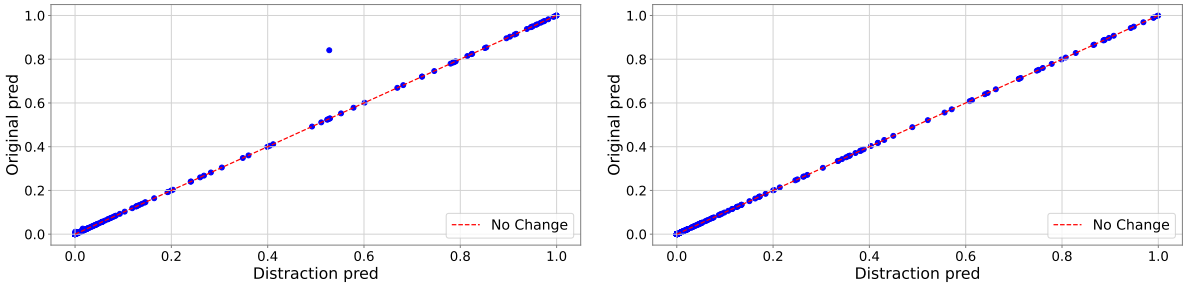
C

Forensic Distractions on SD Experiment results

Appendix C contains all results from the performed experiment, discussed in Section 3.3. This experiment studied the impact of forensic distractions on different SD models. The results are presented in the form of scatter plots showing the predictions both before (y-axis) and after (x-axis) adding distractions, and ROC curves showing the performance of the models with and without distractions present.

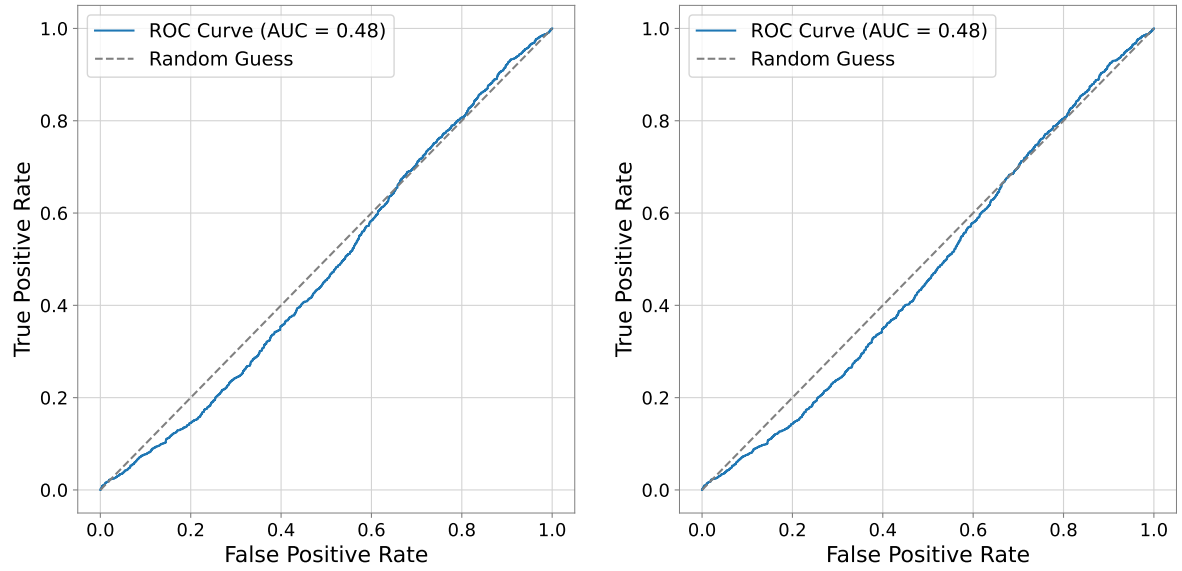
Following is a list of of scatter plots and ROC curves in this appendix:

- CNNDetect [30]: Scatter plots in Figure C.1 and ROC curves in Figure C.2.
- DIMD [28]: Scatter plots in Figure C.3 and ROC curves in Figure C.4.
- GramNet [40]: Scatter plots in Figure C.5 and ROC curves in Figure C.6.
- PatchCraft [41]: Scatter plots in Figure C.7 and ROC curves in Figure C.8.
- NPR [42]: Scatter plots in Figure C.9 and ROC curves in Figure C.10.
- Fusing [43]: Scatter plots in Figure C.11 and ROC curves in Figure C.12.
- UnivFD [45]: Scatter plots in Figure C.13 and ROC curves in Figure C.14.
- RINE [46]: Scatter plots in Figure C.15 and ROC curves in Figure C.16.



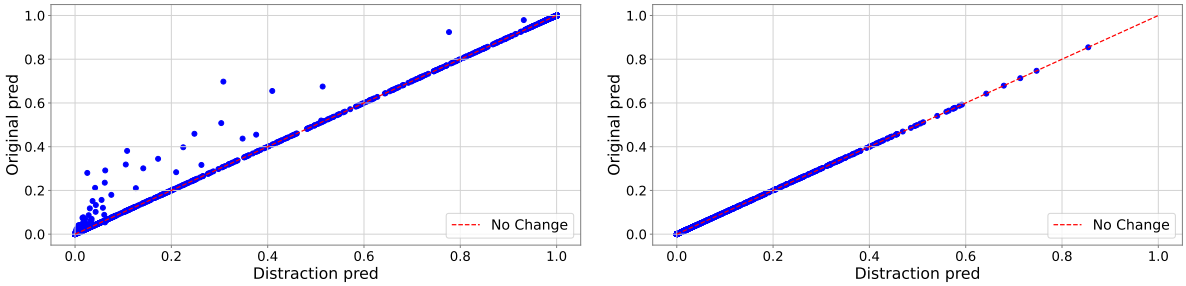
(a) Synthetic images (b) Authentic images

Figure C.1: Scatter plot of predictions of **CNNDetect** [30] on different datasets.



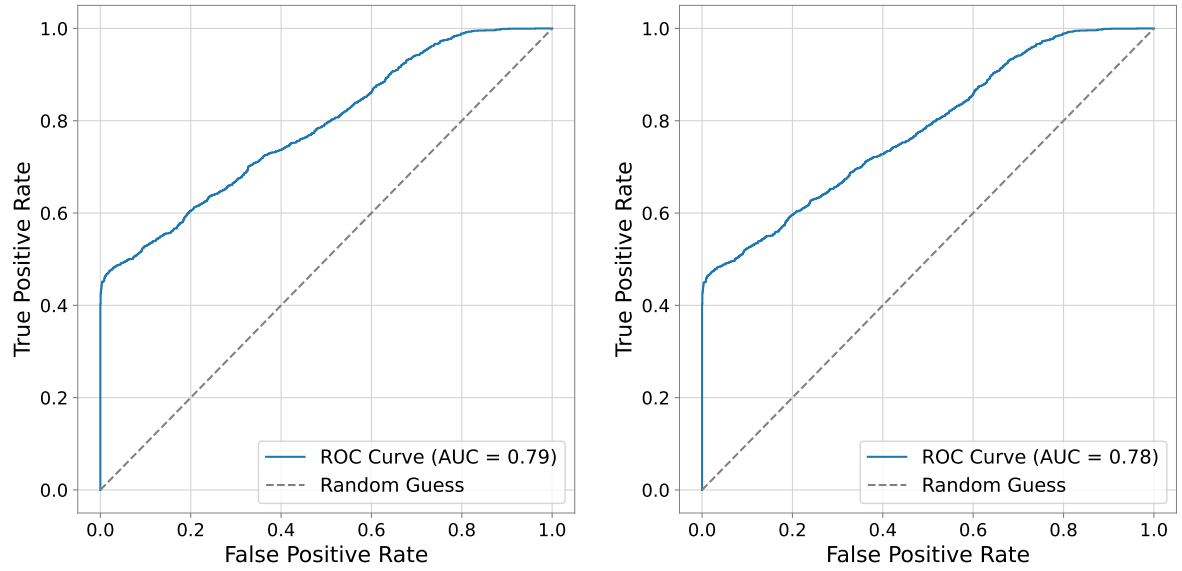
(a) Original (b) Distracted

Figure C.2: ROC curve of **CNNDetect** [30]. Used synthetic datasets: TGIF sd2 (400) and TGIF sdxl (400). Used authentic datasets: Fantastic Reality (800). This model does not get influenced by adding distractions, therefore the ROC curve is the same for both the original and distracted images.



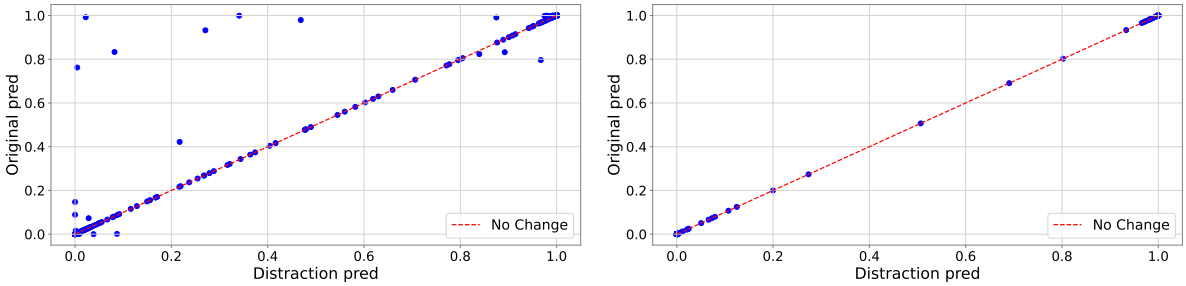
(a) Synthetic images (b) Authentic images

Figure C.3: Scatter plot of predictions of **DIMD** [28] on different datasets.



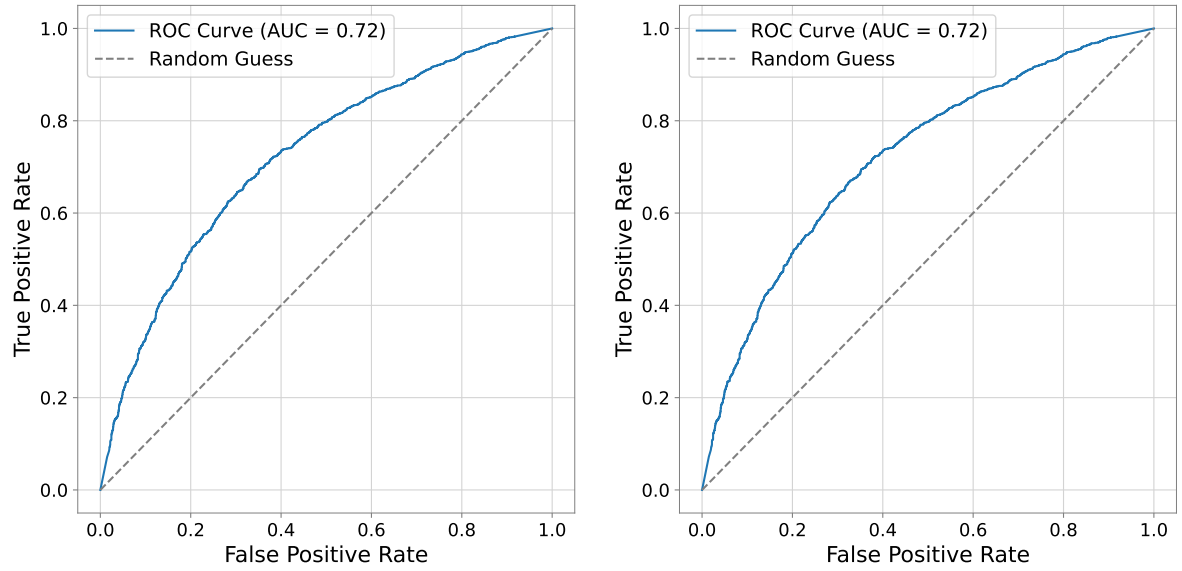
(a) Original (b) Distracted

Figure C.4: ROC curve of **DIMD** [28]. Used synthetic datasets: TGIF sd2 (400) and TGIF sdxl (400). Used authentic datasets: Fantastic Reality (800). This model does not get influenced by adding distractions, therefore the ROC curve is the same for both the original and distracted images.



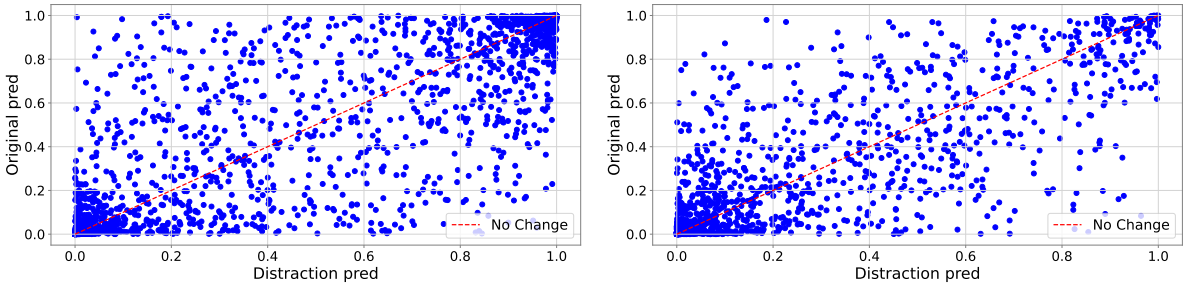
(a) Synthetic images (b) Authentic images

Figure C.5: Scatter plot of predictions of **GramNet** [40] on different datasets.



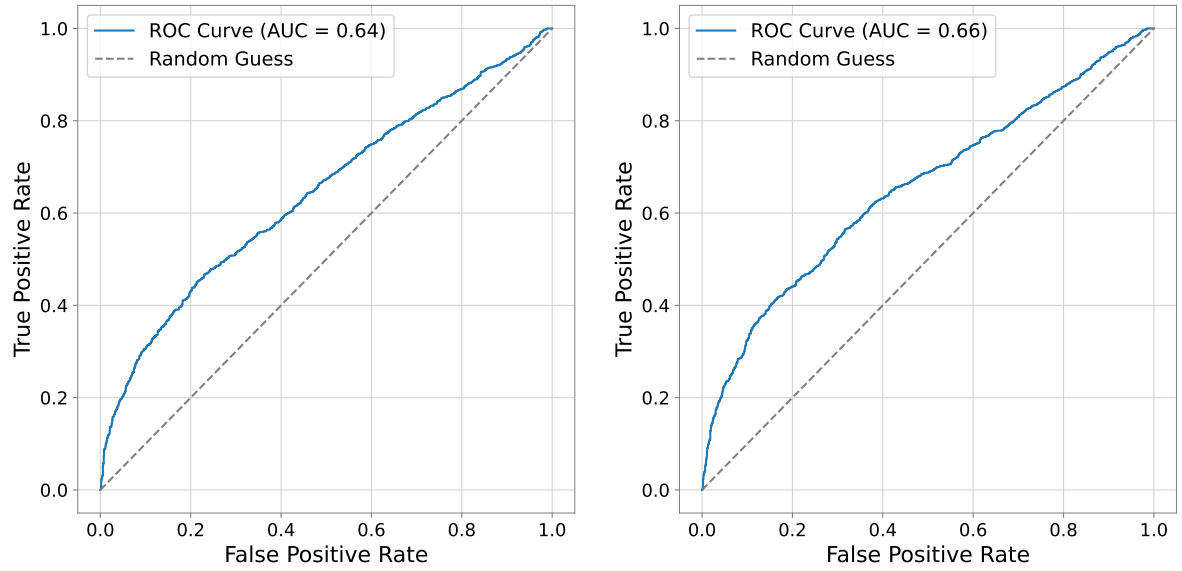
(a) Original (b) Distracted

Figure C.6: ROC curve of **GramNet** [40]. Used synthetic datasets: TGIF sd2 (400) and TGIF sdxl (400). Used authentic datasets: Fantastic Reality (800). This model does not get influenced by adding distractions, therefore the ROC curve is the same for both the original and distracted images.



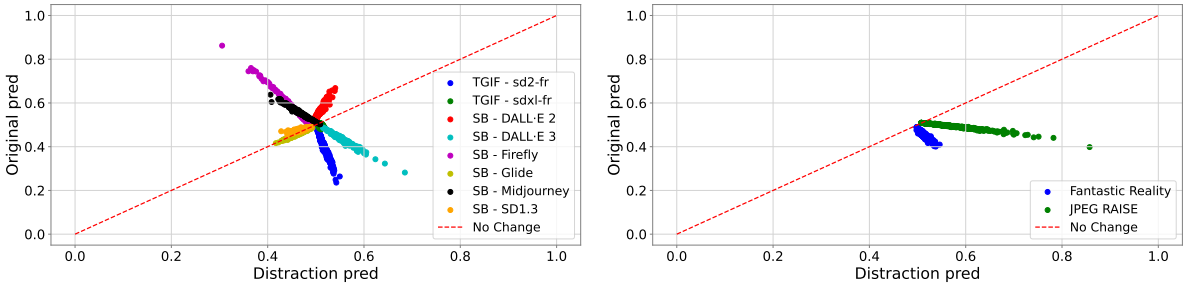
(a) Synthetic images (b) Authentic images

Figure C.7: Scatter plot of predictions of **PatchCraft** [41] on different datasets.



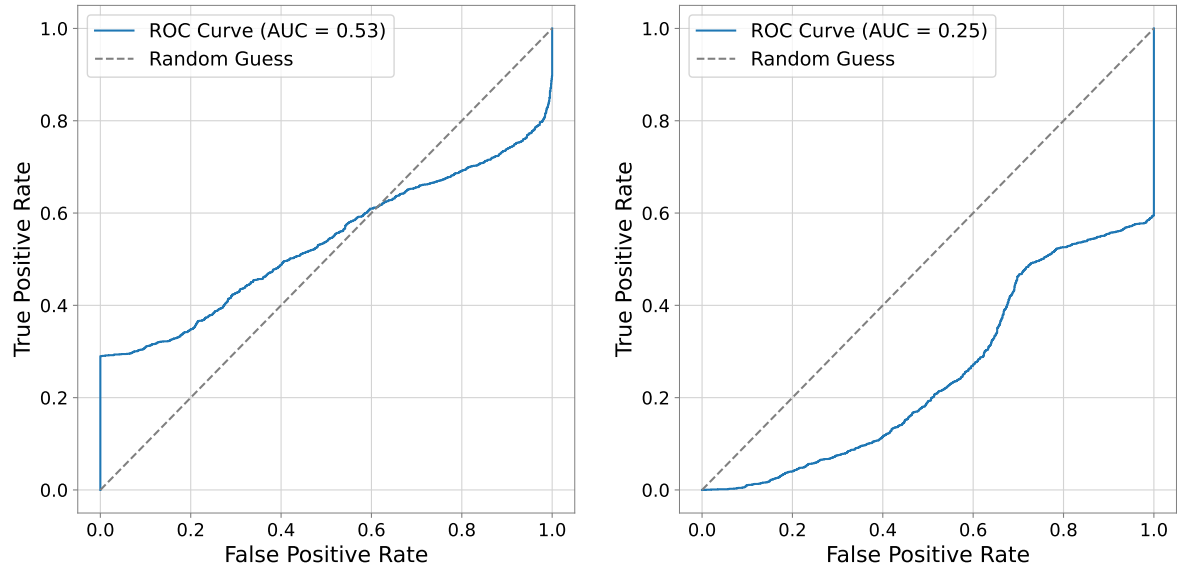
(a) Original (b) Distracted

Figure C.8: ROC curve of **PatchCraft** [41]. Used synthetic datasets: TGIF sd2 (400) and TGIF sdxl (400). Used authentic datasets: Fantastic Reality (800). This model does not get influenced by adding distractions, therefore the ROC curve is the same for both the original and distracted images.



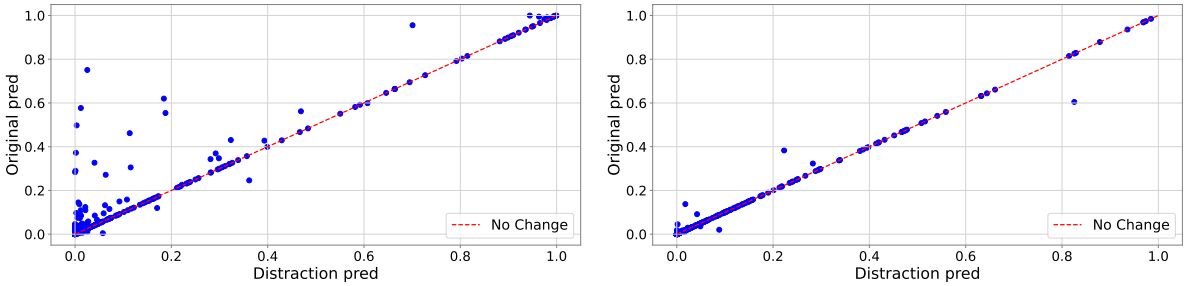
(a) Synthetic images (b) Authentic images

Figure C.9: Scatter plot of predictions of **NPR** [42] on different datasets.



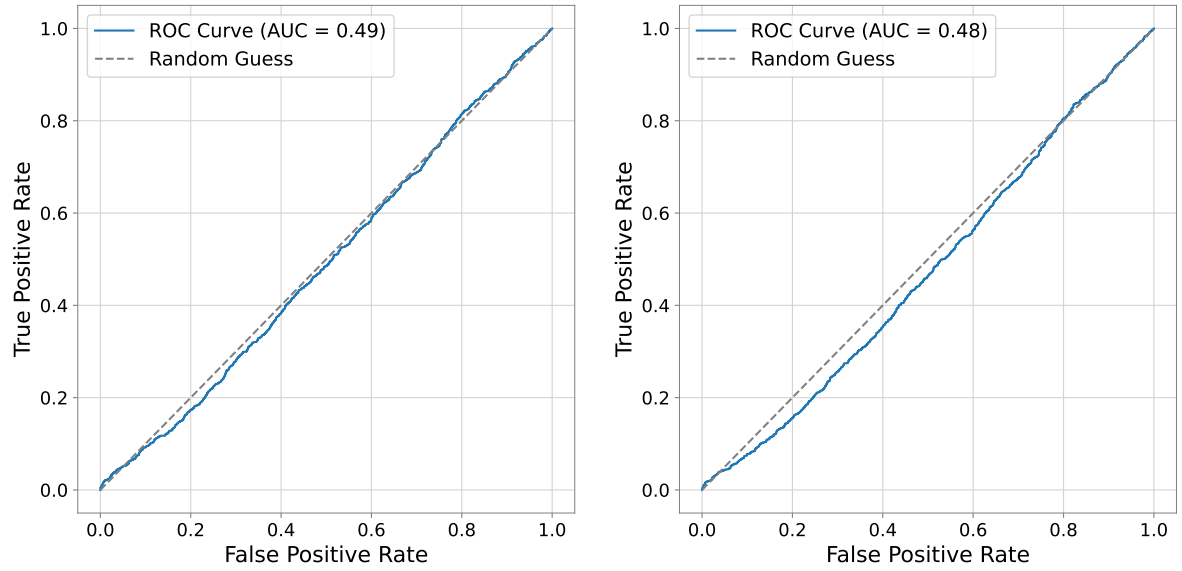
(a) Original (b) Distracted

Figure C.10: ROC curve of **NPR** [42]. Used synthetic datasets: TGIF sd2 (400) and TGIF sdxl (400). Used authentic datasets: Fantastic Reality (800). This model does not get influenced by adding distractions, therefore the ROC curve is the same for both the original and distracted images.



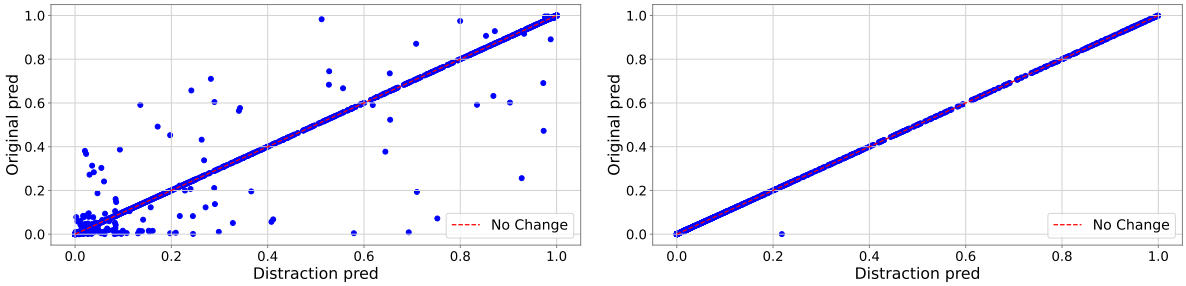
(a) Synthetic images (b) Authentic images

Figure C.11: Scatter plot of predictions of **Fusing** [43] on different datasets.



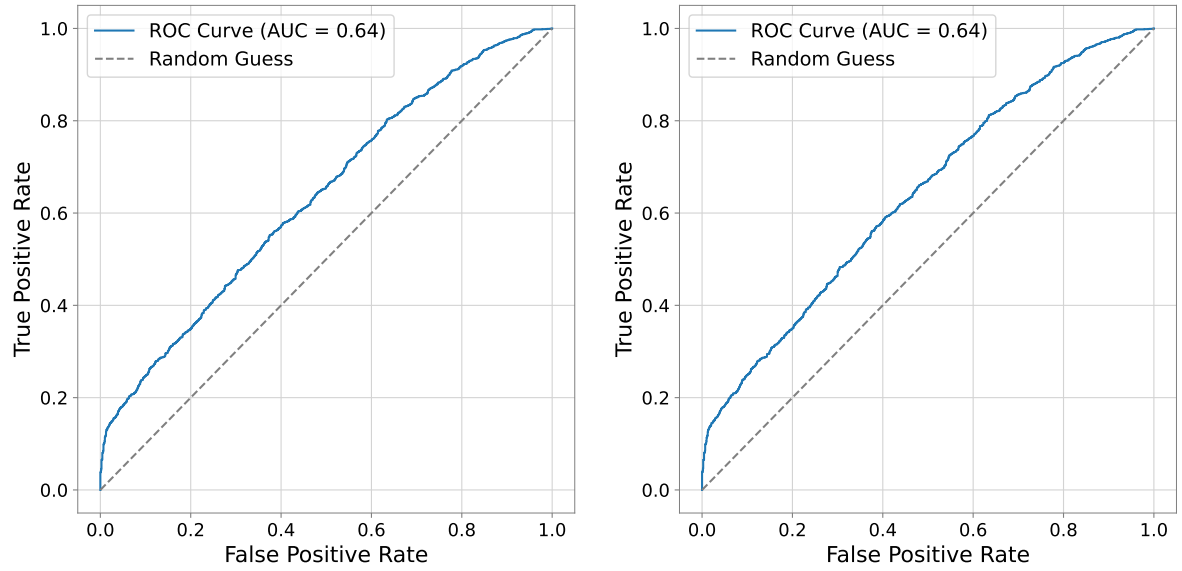
(a) Original (b) Distracted

Figure C.12: ROC curve of **Fusing** [43]. Used synthetic datasets: TGIF sd2 (400) and TGIF sdxl (400). Used authentic datasets: Fantastic Reality (800). This model does not get influenced by adding distractions, therefore the ROC curve is the same for both the original and distracted images.



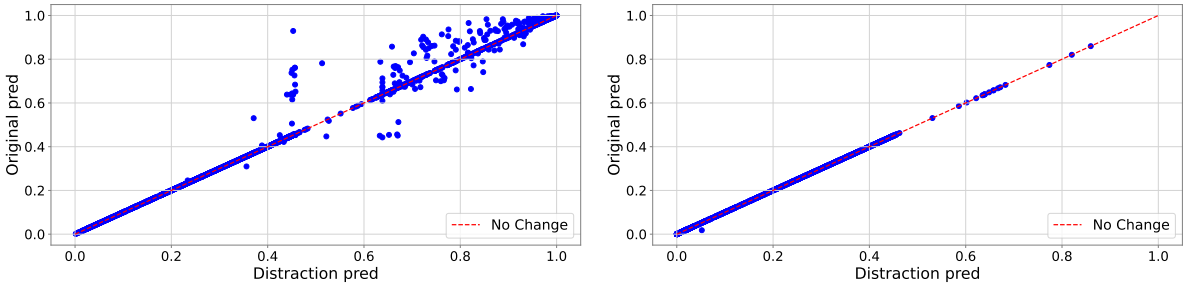
(a) Synthetic images (b) Authentic images

Figure C.13: Scatter plot of predictions of **UnivFD** [45] on different datasets.



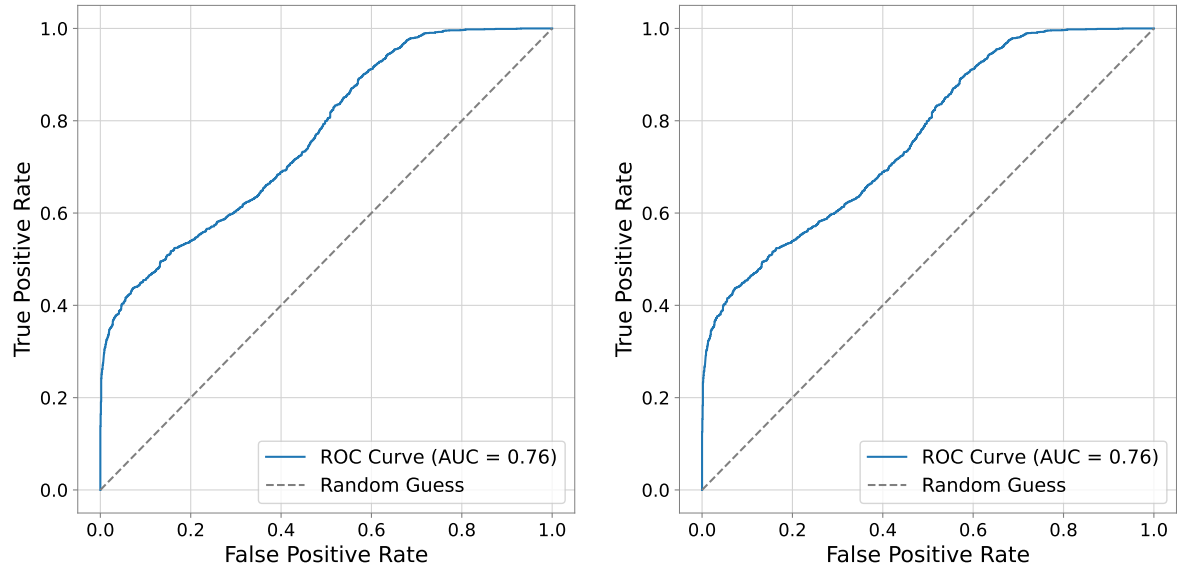
(a) Original (b) Distracted

Figure C.14: ROC curve of **UnivFD** [45]. Used synthetic datasets: TGIF sd2 (400) and TGIF sdxl (400). Used authentic datasets: Fantastic Reality (800). This model does not get influenced by adding distractions, therefore the ROC curve is the same for both the original and distracted images.



(a) Synthetic images (b) Authentic images

Figure C.15: Scatter plot of predictions of **RINE** [46] on different datasets.



(a) Original (b) Distracted

Figure C.16: ROC curve of **RINE** [46]. Used synthetic datasets: TGIF sd2 (400) and TGIF sdxl (400). Used authentic datasets: Fantastic Reality (800). This model does not get influenced by adding distractions, therefore the ROC curve is the same for both the original and distracted images.

This page is intentionally left blank.