2020 25th International Conference on Pattern Recognition (ICPR)
Milan, Italy, Jan 10-15, 2021

RobusterNet: Improving Copy-Move Forgery Detection with Volterra-based Convolutions

Efthimia Kafali, Nicholas Vretos, Theodoros Semertzidis and Petros Daras
Information Technologies Institute, Center for Research and Technology Hellas, Greece
6thkm Charilaou-Thermi Road, Thessaloniki, Greece
{ekaf, vretos, theosem, daras}@iti.gr

2020 25th International Conference on Pattern Recognition (ICPR) | 978-1-7281-8808-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICPR48806.2021.9412587

***Abstract*—Convolutional Neural Networks (CNNs) have recently been introduced for addressing copy-move forgery detection (CMFD). However, current CMFD CNN-based approaches have insufficient performance commitment regarding the localization of the positive class. In this paper, this issue is explored by considering both linear and nonlinear interactions between pixels. A nonlinear Inception module based on second-order Volterra kernels is proposed, in order to ameliorate the results of a state-of-the-art CMFD architecture. The outcome of this work shows that a combination of linear and nonlinear convolution kernels can make the input foreground and background pixels more separable. The proposed approach is evaluated on CASIA and CoMoFoD, two publicly available CMFD datasets, and results to an improved positive class localization performance. Moreover, the findings of the proposed method imply that the nonlinear Inception module stimulates immense robustness against miscellaneous post processing attacks.**

I. Introduction

Copy-move forgery detection (CMFD) aims to detect and localize cloned regions into the same image. This includes the detection and localization of three distinguished categories of pixels: 1) the source, i.e., the pixels that have been copied, 2) the target, i.e., the pixels that have been pasted and 3) the background, or pristine pixels (Fig. 1).

In reference to image forgery detection, CMFD has lately been an active research area, due to the difficulties arising from the genuine similarities between regions of the same image. While the detection of the target object may come from a straightforward solution, the identification of the source object is a challenging process, since it has similar statistical values with the rest of the pristine image patches [1]. Furthermore, besides pure copy-move manipulation attacks, where an object is just copied and pasted into the same image, additional complexity is often introduced by post processing attacks applied to the target object, or the entire forged image.

Thus far, CMFD has primarily been approached by block-based and keypoint-based methods, with end-to-end deep CNN-based solutions being a fresh entry to the field. Although the so far existing CMFD methods have achieved sufficient results on the detection of copy-move attacks on an image level, their localization performance on a pixel level is still inadequate on benchmark datasets, leaving the localization of cloned objects an open research topic.

Concurrently, taking advantage of research findings of neuroscience, confirming the existence of nonlinear operations in the response of visual cells, progress has also been achieved in various computer vision tasks which make use of the Volterra theory [2]. In [3], the limited expressiveness of the linear convolution operation has inspired the adoption of second-order Volterra kernels for an image classification task, where both the linear and quadratic interactions between input image pixels are exploited, enriching the properties of convolution kernels. Furthermore, Volterra filters have been used for action detection in videos in a nonlinear fusion of the spatial and temporal streams [4]. However, despite the fact that Volterra filters have shown encouraging performance over conventional CNNs, their high complexity has been a constraining factor to their wider use.

In this paper the advantages of end-to-end deep learning CMFD architectures are combined with the extended expressiveness of nonlinear convolutions. A state-of-the-art, end-to-end deep learning CMFD method [5] is fine-tuned and the manipulation and copy-move extracted features are nonlinearly fused, through a Volterra-based Inception module. The novelties of the proposed method are:

- The incorporation of Volterra-based convolutions in an Inception module, where both low and high-level features are captured, while both linear and nonlinear interactions between the input pixels are cooperatively considered during training.
- A training strategy proposition where Volterra convolutions are fully exploited with a clear-cut overall complexity, preventing excessive overparameterization.

The rest of this paper is arranged as follows: In section II, CMFD related approaches are described. In section III, the proposed method is outlined, in alignment with the mathematical basis behind the Volterra theory. Details about the training strategy and implementation of the proposed method are illustrated in section IV, while in section V the experimental results of the proposed method are presented for CASIA and CoMoFoD datasets. Finally, in section VI, conclusions over this work are drawn.

II. Related Work

CMFD approaches usually consist of three major steps: 1) feature extraction, 2) feature matching and 3) postprocessing techniques applied to the matched features, where the source and target objects are seen as a whole, in an attempt to enhance the localization of the positive class (forged pixels) ([6], [7]).

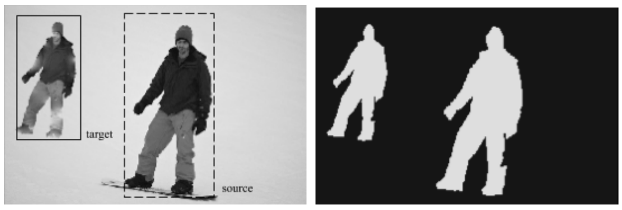

Fig. 1. Image under copy-move attack. The binary mask illustrates the positive class (white pixels) and the negative class (black pixels)

Many CMFD approaches have been proposed that follow block-based methods, where the input image is divided into overlapping patches. The feature extraction process has been addressed by a variety of methods, including DAFMT [8], DWT [9], DCT [10], Zernike moments [11], PCET[12], PCT[13] and RORHFMs [14]. For the matching process, the different blocks are compared based on their similarity and the most similar blocks are decided.

Key-point based solutions are also commonly used for CMFD, with the image being examined for dissimilar regions, in an attempt to localize unusually intense positions. The most common techniques used for feature extraction include scale invariant feature transform (SIFT)([15],[16],[17]), speeded-up robust features (SURF) ([18],[19],[20]) and ORB [21]. The feature matching in key-point based methods is based only on the detected regions, thus their performance is generally better than the block-based methods. In [22], a hybrid framework for CMFD is proposed, where the feature extraction is based on a key-point and the feature matching on a block-based method.

Some deep learning architectures for CMFD have also been proposed. In [23], a DNN is used for feature extraction based on convolutional kernels, while in [24], a DNN-based patch classifier is proposed, for the detection of cloned image regions. However, the aforementioned approaches bear the drawback that the three steps have to be confronted independently and as a result, their optimization is complex. Moreover, regarding the key-point-based and block-based approaches, the feature extraction methods followed are mostly attack-dependent, since most of them have been implemented for detecting copy-move forgeries under explicit postprocessing attacks.

To address these issues, some end-to-end deep learning CMFD methods have lately been introduced which, in contrast to the previously mentioned solutions, can directly be optimized ([25], [5]). BusterNet [5] is an end-to-end deep learning CMFD model following a two-branch architecture. It consists of two segmentation submodels, a Manipulation Detection (Mani-Det) and a Similarity Detection (Simi-Det) branch. The auxiliary branches are responsible for the extraction of manipulation (only target object) and copy-move (source-target objects) features, accordingly. The manipulation and copy-move features are concatenated and fused through an Inception module, to finally produce a three-class copy-move mask, distinguishing source, target and background pixels.

III. RobusterNet: Improving CMFD with Volterra-based Convolutions

With the focus on improving the localization performance of CMFD CNN-based solutions, this work aims to make use of the expressiveness of second-order Volterra kernels. Since Volterra convolutions have shown improvements over image classification tasks, the proposed approach is based on the hypothesis that Volterra convolutions can also ameliorate the results of segmentation-based architectures. Hence, applying this hypothesis to CMFD, the main goal of this work is the exploration of how the nonlinear operations that take place in a receptive field can affect the detection and localization of cloned objects.

The proposed method follows the architecture of BusterNet [5], a state-of-the-art, end-to-end deep learning CMFD architecture. A nonlinear layer which makes use of the second-order Volterra kernels is implemented following [3] and incorporated into a fine-tuning process, where the initial BusterNet Fusion component is replaced by the proposed Volterra Inception module. Thus, **RobusterNet**, a refined version of BusterNet is introduced, which nonlinearly fuses the manipulation and copy-move extracted features.

A. *Volterra-based Convolutions*

The Volterra series are built upon terms of infinite orders. However, the complexity that comes up from the multiplicative operations that take place between the Volterra kernels and the input image pixels, has been a restraining factor for the adaptation of Volterra-based convolutions. Thus, related works principally adopt truncated versions of the Volterra series, as in [3], where the second-order Volterra kernels are employed.

A conventional CNN consists of linear convolution filters. Given an input image patch $\mathbf{I} \in R^{d_h \times d_w}$, reshaped as a vector $\mathbf{x} \in R^N$:

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^T \tag{1}$$

with $n = d_h \cdot d_w$, where d_h and d_w the height and width of the patch in pixels respectively and a linear convolution kernel $\mathbf{w}_1^T$, the output of the convolution operation is computed by:

$$y(x) = \mathbf{w}_1^T \cdot \mathbf{x} + b \tag{2}$$

where $(\cdot)^T$ the transpose operator. The kernel $\mathbf{w}_1^T$ is a vector of n elements, $\mathbf{x}$ is the vectorized image patch with n pixels that the filter is convolved with, and b is the bias.

A linear convolution filter that takes part in the Volterra series is called the first-order Volterra kernel. The input-output function depicting the second-order Volterra filters is written as:

$$y(x) = \mathbf{x}^T \cdot \mathbf{W}_2 \cdot \mathbf{x} + \mathbf{w}_1^T \cdot \mathbf{x} + b \tag{3}$$

where for the quadratic term, $\mathbf{W}_2$ is the second-order Volterra kernel containing $n^2 \cdot n^2$ elements, while $\mathbf{x}$ is the same vectorized image patch that both the linear and quadratic filters are convolved with. The first-order Volterra kernel $\mathbf{w}_1^T$ carries the coefficients of the linear filter's term, while the second-order Volterra kernel $\mathbf{W}_2$ carries the coefficients of the filter's quadratic term. The indices (i, j) agree with the spatial positions of the input pixels (x_i, x_j):

$$\mathbf{w}_1^T = \begin{bmatrix} {w_1}^1 & {w_1}^2 & \dots & {w_1}^n \end{bmatrix} \tag{4}$$

$$\mathbf{W}_2 = \begin{bmatrix} {w_2}^{1,1} & {w_2}^{1,2} & \cdots & {w_2}^{1,n} \\ {w_2}^{2,1} & {w_2}^{2,2} & \cdots & {w_2}^{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ {w_2}^{n,1} & {w_2}^{n,2} & \cdots & {w_2}^{n,n} \end{bmatrix} \tag{5}$$

As stated in [3], in order to tangle the Volterra series into a learning process, the Volterra convolutions have to be implemented as a nonlinear convolutional layer. Hence, the typical backpropagation strategy is adapted to Eq. 3 for the derivation of the equations regarding the backward pass of the Volterra convolutions. The gradients of the layer's output $y(x)$ are computed with regard to the weights ${w_1}^i$ and ${w_2}^{i,j}$, for training the weights of the Volterra kernels. The terms $\frac{\partial y}{\partial {w_1}^i}$, $\frac{\partial y}{\partial {w_2}^{i,j}}$ and $\frac{\partial y}{\partial x_i}$ are used for propagating the error, with regard to the layer's inputs x_i, and optimizing the weight parameters of the Volterra-based convolutional layer. The equations describing the backpropagation scheme are written as:

$$\frac{\partial y}{\partial {w_1}^i} = x_i \qquad \frac{\partial y}{\partial {w_2}^{i,j}} = x_i x_j \tag{6}$$

$$\frac{\partial y}{\partial x_i} = {w_1}^i + \sum_{k=1}^{i} \left({w_2}^{k,i} x_k \right) + \sum_{k=i}^{n} \left({w_2}^{i,k} x_k \right) \tag{7}$$

B. Volterra Inception Module

A Volterra convolution layer can essentially be adopted to all CNN architectures. In [3], a Volterra layer was plugged into a Wide ResNet as the first layer of the network, preventing the overparameterization that would be caused if the layer was plugged deeper into the network. It is well-known that the first convolutional layers of CNN architectures learn low-level features. Nonetheless, the second-order Volterra convolutions in [3] have shown notable performance improvement. In this work, the possibility that a Volterra-based convolutional layer plugged deeper into a CNN can enrich high-level features as well is also explored. Along these lines, the Volterra-based layer is plugged into an Inception module, used for nonlinearly fusing features extracted from BusterNet's Mani-Det and Simi-Det branches. For details regarding the method followed for the extraction of manipulation and copy-move features, the reader may refer to [5].

TABLE I
NUMBER OF PARAMETERS

Architecture	Total number of parameters
BusterNet	15.526.813
RobusterNet	15.552.313

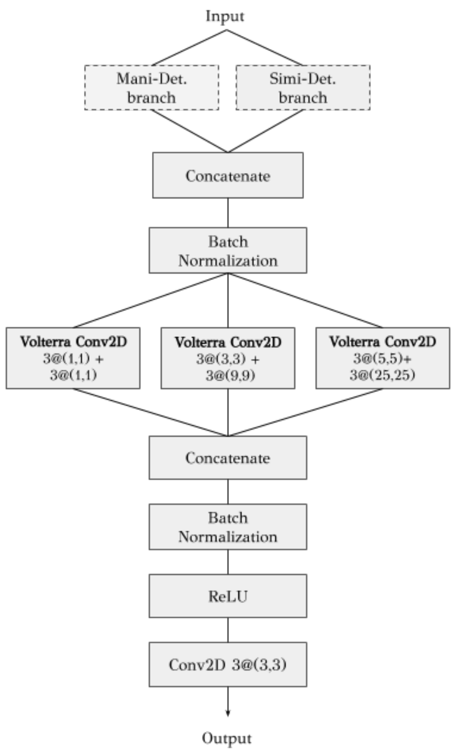

Fig. 2. RobusterNet Overview. The dashed blocks are frozen during training

By plugging our layer into an Inception module, the learning capacity of the proposed model gets wider. Instead of choosing a single kernel size, Volterra-based convolutions with three different kernel sizes (1x1, 3x3, 5x5) are performed for multi-level feature extraction, since in a typical Inception module the convolutions are performed on the same level of the network. This is beneficial for the reduction of the overall complexity that the nonlinearities introduce to the model. Moreover, the multi-level feature extraction that is induced by the different filters of a typical Inception module is further enhanced. Thus, both low and high-level features are captured, while both linear and quadratic interactions between the input pixels are conjointly considered during the fusion of the extracted features.

The architecture of RobusterNet (Fig. 2) can be described as follows: 1) The extracted features from Mani-Det and Simi-Det branches are merged and normalized through a Concatenation and Batch Normalization layer, respectively. 2) The concatenated features are fused through the BN-Volterra-based Inception module and 3) The fused features are sent to a Conv2D layer, for the prediction of three-class masks.

TABLE II
PIXEL LEVEL EVALUATION ON CASIA AND COMOFOD

CASIA						
Methods	[11]	[6]	[7]	[25]	[5]	**Proposed**
Precision	0.227	0.370	0.249	0.239	**0.557**	0.522
Recall	0.133	0.001	0.268	0.137	0.438	**0.525**
F-measure	0.164	0.002	0.254	0.146	0.455	**0.487**
CoMoFoD						
Methods	[11]	[6]	[7]	[25]	[5]	**Proposed**
Precision	0.457	-	0.399	0.362	**0.573**	0.538
Recall	0.343	-	0.476	0.404	0.493	**0.579**
F-measure	0.373	-	0.418	0.311	0.492	**0.516**

For the proposed method, overparameterization caused by the nonlinear operations is omitted, by incorporating the Volterra-based Inception module into a fine-tuning process, where the Mani-Det and Simi-Det modules are frozen during training. Despite the complexity of nonlinear convolutions, the proposed method outperforms former methods on the localization performance by only increasing the number of parameters by 5K. The precise number of parameters for both BusterNet and RobusterNet are illustrated in Table I.

IV. Training Strategy and Implementation Details

A. *Training Strategy*

The learnt weights of BusterNet are kept as initial parameters to the proposed model. Mani-Det and Simi-Det branches are frozen and BusterNet is fine-tuned with a Volterra-based Inception module. Since the proposed method is used for fine-tuning, only a small amount of images is sufficient for training. Thus, RobusterNet is trained on 1000 images from USCISI-CMFD-Full training dataset [1].

Initially, RobusterNet is trained with Adam and a low learning rate of $1 \cdot 10^{-5}$. The loss function to be minimized is the categorical cross entropy, as in [5]. After 10 epochs of very low loss improvement ($< 1 \cdot 10^{-5}$), the learning rate is increased by dividing the current learning rate by $1 \cdot 10^{-1}$. This process continues until the highest learning rate for which the loss is still improving is found. This scheduling strategy ensures that overfitting is prevented at the beginning of the fine-tuning. Moreover, with a low learning rate the gradients are controlled, until the optimal learning rate which evokes a faster convergence is found.

B. *RobusterNet Implementation Details*

The nonlinear convolution layer is implemented in Keras. Initially, the linear and nonlinear kernels of Eq. 4 and Eq. 5 are created and initialized as in [3]. The linear part of Eq. 3 is computed by the built-in convolution operation of Keras backend, while for the nonlinear part, a built-in operation (i.e., *einsum*) is used for modeling the second-order Volterra filter based on Einstein summation conventions. [2]

[1]https://github.com/isi-vista/BusterNet
[2]Code available at https://github.com/efkaf/RobusterNet

TABLE III
CORRECTLY DETECTED COMOFOD IMAGES

Attack	[26]	[11]	[22]	[20]	[23]	[7]	[25]	[5]	**Proposed**
Base	53	90	102	88	97	93	53	117	**139**
BC1	-	91	-	-	-	94	50	116	**139**
BC2	-	89	-	-	-	94	53	115	**136**
BC3	42	89	99	90	94	88	48	109	**132**
CA1	-	93	-	-	-	98	50	117	**139**
CA2	-	93	-	-	-	96	50	116	**139**
CA3	45	92	99	90	94	96	48	116	**141**
CR1	44	92	90	82	72	97	51	117	**139**
CR2	-	91	-	-	-	95	50	116	**139**
CR3	-	90	-	-	-	92	54	116	**137**
IB1	-	90	91	94	104	91	53	113	**126**
IB2	47	87	-	-	-	88	32	98	**113**
IB3	-	84	-	-	-	84	26	93	**107**
JC1	-	43	-	-	-	69	18	60	**86**
JC2	-	63	-	-	-	73	21	77	**94**
JC3	-	72	-	-	-	75	26	86	**104**
JC4	5	73	-	-	-	77	29	103	**120**
JC5	-	76	-	-	-	81	38	99	**118**
JC6	-	80	-	-	-	83	33	101	**121**
JC7	-	86	-	-	-	87	42	107	**123**
JC8	-	88	-	-	-	92	42	109	**128**
JC9	-	81	89	31	78	87	36	106	**119**
NA1	-	24	-	-	-	41	38	100	**123**
NA2	3	42	-	-	-	66	39	102	**120**
NA3	-	-	-	-	-	-	-	124	**131**
Total	239	1899	570	475	539	2037	980	2633	**3113**

V. Evaluation

The proposed method is evaluated on two levels: 1) on its localization performance, i.e., its ability to discriminate between foreground and background pixels and 2) on its robustness against various postprocessing attacks. Both source and target objects are labeled as forged, so that our method can be compared with other CMFD approaches. The results of the compared methods are reported in line with [5].

A. *Evaluation Datasets*

Two benchmark CMFD datasets are used for the evaluation of our method. CASIA is a mixed splice and copy-move dataset, while CoMoFoD [26] is a copy-move dataset where besides clean copy-move manipulation (object cloning), images with postprocessing attacks are also available. The proposed method is evaluated on 1313 CASIA and 5000 CoMoFoD copy-move samples provided by [5].

B. *Evaluation Metrics*

The proposed method is evaluated on a pixel level. We compute per image precision, recall and F-measure and their mean values over the entire dataset are reported. The precision metric (Eq. 8) corresponds to the proportion of the predicted forged pixels that are actually forged, while the recall metric (Eq. 9) indicates the proportion of the actual forged pixels that the model predicted as forged. The F-measure (Eq. 10) is a

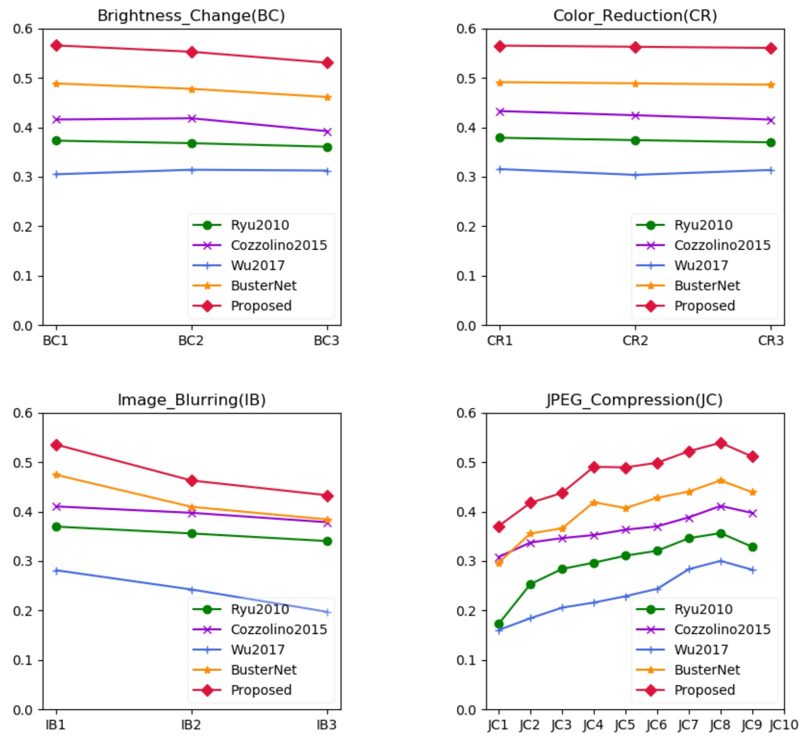

Fig. 3. F-measure over postprocessing attacks on CoMoFoD

metric combining precision and recall and is a useful criterion in cases of unbalanced class distribution, such as CMFD on images with a large number of negative class pixels. This evaluation protocol is reported for both CASIA and CoMoFoD datasets for all the methods available in [5] for he respective datasets.

Moreover, to better capture the behaviour of second-order Volterra kernels, our method's robustness against CoMoFoD images that suffer from postprocessing attacks is explored, as in [5]. An image is counted as correctly detected, if its F-measure is higher than 0.5. Thus, the number of correctly detected images for the total of 5000 samples in 25 different attacks is reported.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{8}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{9}$$

$$F-measure = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{10}$$

C. Localization Performance Analysis

Table II shows the localization performance of the proposed method for CASIA and CoMoFoD datasets respectively. The results for method [6] are not explicitely reported in [5] for CoMoFoD. RobusterNet outperforms the compared methods with a pixel level recall increment of $\sim$ 9%, causing a slight detriment to precision. However, the improvement over recall is the factor causing an improved F-measure which contributes significantly to the proposed method's robustness against post processing attacks, presented in subsection D. The high recall increment shows that RobusterNet has identified a higher proportion of forged pixels among the total number of forged pixels over the entire dataset, than the compared methods.

Given the fact that the inputs to the proposed Volterra Inception module were features extracted from Mani-Det and Simi-Det branches, this score implies that the nonlinear fusion has made a rectifying decision for 9% of the positive class pixels.

Furthermore, the CoMoFoD samples used for measuring precision, recall and F-measure for Table II are mainly images under postprocessing attacks, excluding only the base category (200 out of 5000 images). Thus, the results over CoMoFoD also indicate a better generalization ability.

D. Robustness Analysis Against Postprocessing Attacks

RobusterNet's performance against postprocessing attacks is evaluated on 5000 CoMoFoD samples. Each one of the 25 different categories of CoMoFoD contains 200 images. The different attacks include plain copy-move (Base), brightness change (BC), contrast adjustment (CA), color reduction (CR), image blurring (IB),JPEG compression (JC), and noise adding (NA) [26]. Finally, the different category names are marked with numbers, indicating the parameters used for the attack.

Table III illustrates our method's performance against the total of 5000 CoMoFoD samples, including the base category. For the attack NA3 results were not explicitely reported in [5]. The total number of correctly detected images outperforms the state-of-the-art by 480 images, implying a strong ability of localization performance, regardless of the type of attack.

For Fig. 3 only the postprocessed images of CoMoFoD were considered. It is clear that RobusterNet has an improved F-measure for all the types of attacks. Even in JPEG compression (JC) and image blurring (IB) where BusterNet was not exceptionally better than the compared methods, RobusterNet has a distinctly better performance. The Volterra Inception module proves to be immensely robust against post processing attacks, still managing to detect the positive pixels, disregarding transformations such as the brightness, noise or blurring of the inspected image.

VI. Conclusion

In this work, a Volterra-based Inception module for CMFD is introduced. It is shown that the linear and quadratic interactions between the input pixels increase the localization performance of the cloned objects, i.e., the foreground and background pixels become more separable. Furthermore, the performed experiments validate that Volterra convolutions have a strong localization ability, regardless of the existence of postprocessing attacks.

For the proposed method the Volterra convolutions have been employed into a fine-tuning process and as a result the high overall complexity was omitted. However, it is noteworthy that, although the error was not propagated to the feature extraction branches, our method still achieved a recall increment of $\sim 9\%$, when compared to BusterNet. This rectifying behavior can be further explored as a fast solution for improving the performance of well-established segmentation architectures. Furthermore, the proposed approach proves to be insensitive to post processing attacks by a quite large margin when compared to state-of-the-art methods. Hence, nonlinear convolutional filters can be used for fine-tuning other image manipulation detection architectures, which may be deprived of their ability to localize the manipulated areas, due to post processing attacks.

VII. Acknowledgements

The work presented in this paper was supported by the H2020 project FANDANGO (780355).

References

[1] B. Wen, Y. Zhu, R. Subramanian, T.-T. Ng, X. Shen, and S. Winkler, "Coverage — a novel database for copy-move forgery detection," 09 2016, pp. 161–165.

[2] V. Volterra, "Theory of functionals and of integral and integro-differential equations," *Bull. Amer. Math. Soc*, vol. 38, no. 1, p. 623, 1932.

[3] G. Zoumpourlis, A. Doumanoglou, N. Vretos, and P. Daras, "Non-linear convolution filters for cnn-based learning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4761–4769.

[4] S. Roheda and H. Krim, "Conquering the cnn over-parameterization dilemma: A volterra filtering approach for action recognition," *arXiv preprint arXiv:1910.09616*, 2019.

[5] Y. Wu, W. Abd-Almageed, and P. Natarajan, "Busternet: Detecting copy-move image forgery with source/target localization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 168–184.

[6] V. Christlein, C. Riess, J. Jordan, C. Riess, and E. Angelopoulou, "An evaluation of popular copy-move forgery detection approaches," *IEEE Transactions on information forensics and security*, vol. 7, no. 6, pp. 1841–1854, 2012.

[7] D. Cozzolino, G. Poggi, and L. Verdoliva, "Efficient dense-field copy–move forgery detection," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 11, pp. 2284–2297, 2015.

[8] C.-M. Pun and J.-L. Chung, "A two-stage localization for copy-move forgery detection," *Information Sciences*, vol. 463, pp. 33–55, 2018.

[9] M. Bashar, K. Noda, N. Ohnishi, and K. Mori, "Exploring duplicated regions in natural images," *IEEE Transactions on Image Processing*, 2010.

[10] T. Mahmood, T. Nawaz, A. Irtaza, R. Ashraf, M. Shah, and M. T. Mahmood, "Copy-move forgery detection technique for forensic analysis in digital images," *Mathematical Problems in Engineering*, vol. 2016, 2016.

[11] S.-J. Ryu, M.-J. Lee, and H.-K. Lee, "Detection of copy-rotate-move forgery using zernike moments," in *International workshop on information hiding*. Springer, 2010, pp. 51–65.

[12] M. Emam, Q. Han, and X. Niu, "Pcet based copy-move forgery detection in images under geometric transforms," *Multimedia Tools and Applications*, vol. 75, no. 18, pp. 11 513–11 527, 2016.

[13] Y. Li, "Image copy-move forgery detection based on polar cosine transform and approximate nearest neighbor searching," *Forensic science international*, vol. 224, no. 1-3, pp. 59–67, 2013.

[14] J. Zhong, Y. Gan, and S. Xie, "Radon odd radial harmonic fourier moments in detecting cloned forgery image," *Chaos, Solitons & Fractals*, vol. 89, pp. 115–129, 2016.

[15] I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo, and G. Serra, "A sift-based forensic method for copy–move attack detection and transformation recovery," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 3, pp. 1099–1110, 2011.

[16] A. Costanzo, I. Amerini, R. Caldelli, and M. Barni, "Forensic analysis of sift keypoint removal and injection," *IEEE transactions on information forensics and security*, vol. 9, no. 9, pp. 1450–1464, 2014.

[17] B. Yang, X. Sun, H. Guo, Z. Xia, and X. Chen, "A copy-move forgery detection method based on cmfd-sift," *Multimedia Tools and Applications*, vol. 77, no. 1, pp. 837–855, 2018.

[18] V. Manu and B. M. Mehtre, "Detection of copy-move forgery in images using segmentation and surf," in *Advances in signal processing and intelligent recognition systems*. Springer, 2016, pp. 645–654.

[19] B. Shivakumar and S. S. Baboo, "Detection of region duplication forgery in digital images using surf," *International Journal of Computer Science Issues (IJCSI)*, vol. 8, no. 4, p. 199, 2011.

[20] E. Silva, T. Carvalho, A. Ferreira, and A. Rocha, "Going deeper into copy-move forgery detection: Exploring image telltales via multi-scale analysis and voting processes," *Journal of Visual Communication and Image Representation*, vol. 29, pp. 16–32, 2015.

[21] Y. Zhu, X. Shen, and H. Chen, "Copy-move forgery detection based on scaled orb," *Multimedia Tools and Applications*, vol. 75, no. 6, pp. 3221–3233, 2016.

[22] J. Li, X. Li, B. Yang, and X. Sun, "Segmentation-based image copy-move forgery detection scheme," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 3, pp. 507–518, 2014.

[23] Y. Liu, Q. Guan, and X. Zhao, "Copy-move forgery detection based on convolutional kernel network," *Multimedia Tools and Applications*, vol. 77, no. 14, pp. 18 269–18 293, 2018.

[24] J. Bunk, J. H. Bappy, T. M. Mohammed, L. Nataraj, A. Flenner, B. Manjunath, S. Chandrasekaran, A. K. Roy-Chowdhury, and L. Peterson, "Detection and localization of image forgeries using resampling features and deep learning," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 2017, pp. 1881–1889.

[25] Y. Wu, W. Abd-Almageed, and P. Natarajan, "Deep matching and validation network: An end-to-end solution to constrained image splicing localization and detection," in *Proceedings of the 25th ACM international conference on Multimedia*. ACM, 2017, pp. 1480–1502.

[26] D. Tralic, I. Zupancic, S. Grgic, and M. Grgic, "Comofod—new database for copy-move forgery detection," in *Proceedings ELMAR-2013*. IEEE, 2013, pp. 49–54.