MVSS-Net: Multi-View Multi-Scale Supervised Networks for Image Manipulation Detection

Chengbo Dong*, Xinru Chen*, Ruohan Hu, Juan Cao and Xirong Li, *Member, IEEE,*

Abstract—As manipulating images by copy-move, splicing and/or inpainting may lead to misinterpretation of the visual content, detecting these sorts of manipulations is crucial for media forensics. Given the variety of possible attacks on the content, devising a generic method is nontrivial. Current deep learning based methods are promising when training and test data are well aligned, but perform poorly on independent tests. Moreover, due to the absence of authentic test images, their image-level detection specificity is in doubt. The key question is how to design and train a deep neural network capable of learning generalizable features *sensitive* to manipulations in novel data, whilst *specific* to prevent false alarms on the authentic. We propose multi-view feature learning to jointly exploit tampering boundary artifacts and the noise view of the input image. As both clues are meant to be semantic-agnostic, the learned features are thus generalizable. For effectively learning from authentic images, we train with multi-scale (pixel / edge / image) supervision. We term the new network MVSS-Net and its enhanced version MVSS-Net++. Experiments are conducted in both within-dataset and cross-dataset scenarios, showing that MVSS-Net++ performs the best, and exhibits better robustness against JPEG compression, Gaussian blur and screenshot based image re-capturing.

Index Terms—Image manipulation detection, multi-view feature learning, multi-scale supervision, model sensitivity and specificity

1 Introduction

DIGITAL images can now be manipulated with ease and often in a visually imperceptible manner [1]. *Copy-move* (copy and move elements from one region to another region in a given image), *splicing* (copy elements from one image and paste them on another image) and *inpainting* (removal of unwanted elements) are three common types of image manipulation that could lead to misinterpretation and thus malicious use of the visual content [2], [3], [4], [5]. Auto-detection of the presence of these sorts of manipulations in a given image is crucial for media forensics and trustworthy information sharing in the cyberspace. We aim to not only discriminate manipulated images from the authentic, but also pinpoint tampered regions at the pixel level.

While pictorial content tampering has been long existing, media forensics is a relatively new research field [5]. Traditionally, carefully hand-crafted features are extracted from a given image to capture subtle differences between its tampered and authentic regions. The differences are calculated by varied approaches, including media-format based compression artifacts [6], [7], physics-based lighting inconsistency [8], [9], statistical modeling [10], local noise estimation [11], *etc*. However, due to the variety of possible attacks on the digital content, a major challenge in the field is that manipulation detection may not be resolved by a single approach with a single source of information. What makes the problem even more challenging is that when images are uploaded and circulate on social media platforms, regular low-level image processing such as re-sizing, re-compression, re-capturing and aesthetic image enhancement, inevitably weakens forensic traces [12]. Towards conquering the challenges, unsurprisingly, the state-of-the-arts are deep learning based [13], [14], [15], [16], [17], [18], specifically focusing on pixel-level manipulation detection [13], [15], [16], also known as manipulation localization [19]. With only two classes (*manipulated* versus *authentic*) in consideration, the task appears to be a simplified case of image semantic segmentation. However, an off-the-shelf semantic segmentation network

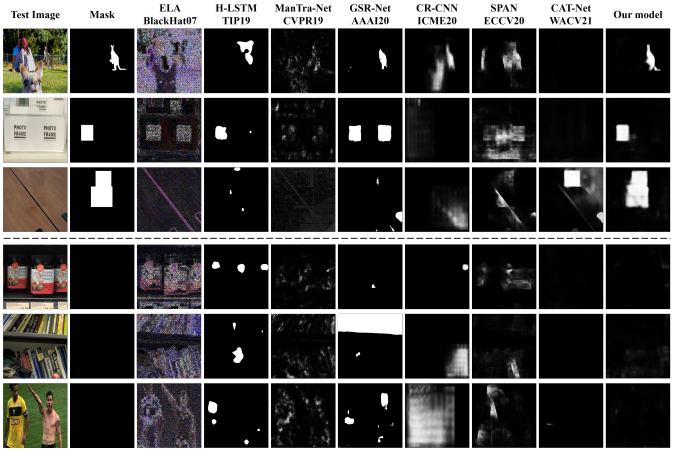


Fig. 1. **Image manipulation detection by the state-of-the-art**. Test images in the first three rows are manipulated by splicing, copy-move and inpainting, respectively. Test images in the last three rows are authentic (thus with blank mask). Our model (*MVSS-Net++*) strikes a good balance between detection sensitivity (lower miss detection on the manipulated) and specificity (lower false alarm on the authentic).

- *Chengbo Dong, Xinru Chen, Ruohan Hu and Xirong Li are with the Key Lab of Data Engineering and Kowledge Engineering, Renmin University of China and the AIMC Lab, School of Information, Renmin University of China, Beijing 100872, China.*
  E-mail: dongchengbo@ruc.edu.cn, chen_xinru1999@163.com, huruohan1126@163.com, xirong@ruc.edu.cn
- *Juan Cao is with Institute of Computing Technology, Chinese Academy of Sciences and the the Key Laboratory of Media Convergence Production Technology and Systems, Beijing 100864, China.*
  E-mail: caojuan@ict.ac.cn
- *Chengbo Dong and Xinru Chen contributed equally to this work. Corresponding author: Xirong Li.*

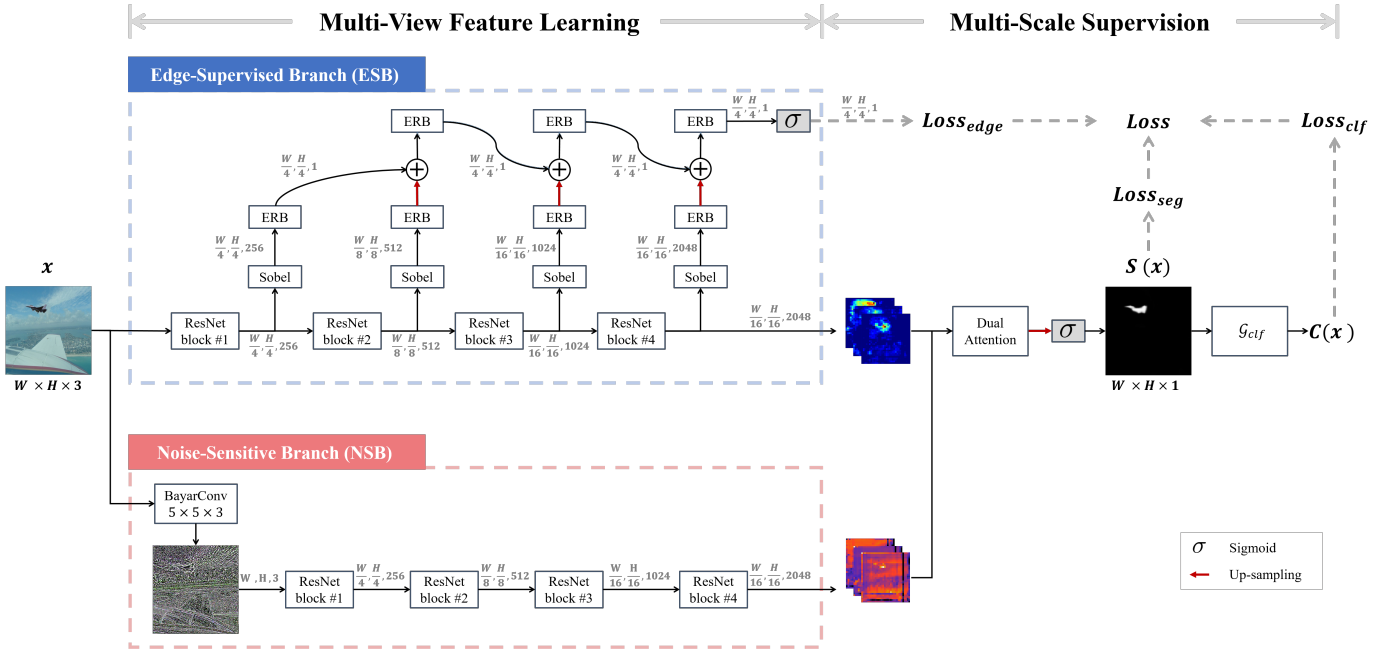


Fig. 2. **Conceptual diagram of the proposed multi-view multi-scale supervised networks for image manipulation detection**. We use the edge-supervised branch (ESB) and the noise-sensitive branch (NSB) to learn semantic-agnostic features for manipulation detection, and multi-scale supervision to strike a proper balance between model sensitivity and specificity. The non-trainable sigmoid (σ) layer is shown in gray. The $\mathcal{G}_{clf}$ module is responsible for converting a pixel-level segmentation map $S(x)$ to an image-level prediction $C(x)$. Depending on how the module is implemented, we have *MVSS-Net* which uses global max pooling (GMP) and *MVSS-Net++* which uses ConvGeM.

is suboptimal for the task, as it is designed to capture semantic information, making the network dataset-dependent and do not generalize. Prior research [16] reports that DeepLabv2 [20] trained on the CASIAv2 dataset [21] performs well on the CAISAv1 dataset [22] homologous to CASIAv2, yet performs poorly on the non-homologous COVER dataset [23]. A similar behavior of FCN [24] is also observed in this study. It has been increasingly recognized that deep neural networks (DNNs) perform well when the training and test data are well aligned in terms of their data source and manipulation methods, but often perform badly on independent tests [19]. Hence, the key question is how to design and train a DNN capable of learning *semantic-agnostic* features *sensitive* to manipulations, whilst *specific* to prevent false alarms?

In order to learn semantic-agnostic features, image content originally presented in the RGB view has to be suppressed. Depending on at what stage the suppression occurs, we categorize existing methods into two groups, *i.e.* noise-view methods [14], [15], [17], [18], [25] and edge-supervised methods [13], [16]. Given the hypothesis that novel elements introduced by splicing and/or inpainting differ from the authentic part in terms of their noise distributions, the noise-view methods aim to exploit such discrepancy. A noise map of an input image, generated either by pre-defined high-pass filters [29] or by their trainable counterparts [25], [30], is fed into a DNN, either alone [17], [25] or together with the input image [14], [15], [18]. Note that the methods are ineffective for detecting copy-move which introduces no new element. The edge-supervised methods try to find boundary artifacts around a tampered region, implemented by using an object-detection head to regress a bounding box to cover the region [14], [17] or an auxiliary branch to reconstruct the region's edge [13], [16]. Note that the prior arts uniformly sum [13] or concatenate [16] features from different layers of the backbone as input of the auxiliary branch. As such, there is a risk that deeper-layer features, which are responsible for manipulation detection, remain semantic-aware and thus not generalizable.

To measure a model's generalizability, a common evaluation protocol [13], [15], [16], [18] is to first train the model on a public dataset, say CASIAv2 [21], and then test it on other public datasets such as NIST16 [31], Columbia [32], and CASIAv1 [22]. To our surprise, however, the evaluation is performed exclusively on manipulated images, with pixel-level metrics reported. The specificity of the model, which reveals how it handles authentic images and is thus crucial for real-world usability, is ignored. As shown in Fig. 1, both traditional Error Level Analysis (ELA) and current deep learning methods [13], [15], [18] make serious false alarms on authentic images. As the current methods mainly use pixel-wise segmentation losses to which an authentic example can contribute is marginal, it is difficult for these methods to exploit the authentic data as holistic context to improve their specificity.

Given the need of exploiting the noise view along with the original RGB view and the need of jointly considering both local edge information and holistic context, it is nontrivial to design a DNN that performs the manipulation detection task in general. We get inspiration from contemporary advances in other research domains. In the context of generic semantic segmentation, the Border Network [33] aggregates features progressively to predict object boundaries. We adapt that technique for tracing subtle boundary artifacts around manipulated regions. In the context of medical image analysis, LesionNet [34] incorporates an image classification loss for segmenting retinal lesions in color fundus photographs. We borrow this idea to take authentic images into account. We propose *multi-view* feature learning with *multi-scale* supervised networks (MVSS-Net series) for image manipulation detection. Note that several previous approaches can potentially

TABLE 1
A taxonomy of the state-of-the-art for image manipulation detection. Methods marked with † are open-sourced or with models released, and will be compared in our experiments. Star (*) indicates edge information is implicitly considered via bounding-box regression. Edge and image labels used by this paper are automatically extracted from pixel-level annotations. So our multi-scale supervision requires no extra manual labeling.

Method	Views			Semantic segmentation backbone	Scales of supervision			Level of evaluation	
	RGB	*Noise*	*Fusion*		*pixel*	*edge*	*image*	*pixel*	*image*
MFCN, Salloum *et al*. 2017 [13]	+	-	-	FCN	+	+	-	+	-
RGB-N, Zhou *et al*. 2018 [14]	+	SRM filter	late fusion (bilinear pooling)	Faster R-CNN	-	*	-	+	-
H-LSTM†, Bappy *et al*. 2019 [3]	+	-	-	Patch-LSTM	+	-	-	+	-
ManTra-Net†, Wu *et al*. 2019 [15]	+	SRM filter BayarConv	early fusion (feature concatenation)	Wider VGG	+	-	-	+	-
HP-FCN†, Li & Huang 2019 [25]	-	High-pass filters	-	FCN	+	-	-	+	-
GSR-Net†, Zhou *et al*. 2020 [16]	+	-	-	DeepLabv2	+	+	-	+	-
CR-CNN†, Yang *et al*. 2020 [17]	-	BayarConv	-	Mask R-CNN	+	*	-	+	-
SPAN†, Hu *et al*. 2020 [18]	+	SRM filter BayarConv	early fusion (feature concatenation)	Wider VGG	+	-	-	+	-
MM-Net, Yang *et al*. 2021 [26]	+	BayarConv	middle fusion (attention guidance)	Mask R-CNN	+	-	-	+	-
JPEG-ComNet, Rao & Ni 2021 [27]	+	SRM filter	early fusion (feature concatenation)	Siamese FCN	+	+	-	+	-
CAT-Net†, Kwon *et al*. 2021 [28]	+	DCT	middle fusion (feature concatenation)	HRNet	+	-	-	+	-
Proposed *MVSS-Net*†	+	BayarConv	late fusion (dual attention)	FCN	+	+	+	+	+

and indirectly learn the boundary artifacts along with the noise view, *e.g*. via a bounding-box regression task [14]. To the best of our knowledge (Table 1), we are the first to jointly exploit the noise view and the *explicitly* extracted boundary artifacts to learn manipulation detection features. With multi-scale supervision, we also make an initial endeavor to learn from the authentic data. Note that the above joint exploitation is technically nontrivial. For instance, simply adding the image classification loss improves the model specificity, but at the cost of considerable degrade in pixel-level detection performance, as our experiments show. To combine the best of the two worlds, new networks are needed.

To sum up, our major contributions are as follows:

• **Proposed *MVSS-Net* as a new network for image manipulation detection**. As Fig. 2 shows, the technical strength of *MVSS-Net* lies in its capability to jointly exploit the multi-view input, the explicitly extracted boundary artifacts and the holistic information in an end-to-end manner. Multi-view feature learning is designed to extract semantic-agnostic and thus more generalizable features.

• **Network training by multi-scale supervision**. This allows us to learn effectively from authentic images, which are ignored by the prior arts. Consequently, the manipulation detection specificity is improved substantially.

• **Superior to the SOTA on multiple benchmarks**. As extensive experiments on two training sets and six test sets show, *MVSS-Net* compares favorably against the SOTA. The inclusion of authentic test images reveals a model's detection specificity at the image level. Code and models are available at GitHub[1].

A preliminary version of this work was published at ICCV 2021 [35]. The journal article improves over the conference paper in multiple aspects. First, for converting pixel-level manipulation detection to an image-level prediction, we propose ConvGeM to replace global max pooling (GMP) used in [35]. The new module effectively overcomes two downsides of GMP, *i.e*. the bottleneck in back propagating the image-scale loss and the lack of ability to consider the amount and the spatial distribution of positive responses. This results in a better model *MVSS-Net++*. Second, we strengthen our evaluation by including three more baseline methods, *i.e*. H-LSTM [3], SPAN [18] and CAT-Net [28], and a recently released dataset, *i.e*. IMD [36]. In addition, we present a pilot study on how the current models react to manipulated images given re-capturing by screenshot, a common operation when images circulate on the Internet.

[1] https://github.com/dong03/MVSS-Net

2 Related Work

We are inspired by a number of recent works that made novel attempts to learn semantic-agnostic features for image manipulation detection, see Table 1. We describe in brief how these attempts are implemented and explain our novelties accordingly. We focus on deep learning approaches to copy-move / splicing / inpainting detection. For the detection of low-level manipulations such as Gaussian Blur and JPEG compression, we refer to [30].

In order to suppress the content information, Li and Huang [25] propose to implement an FCN's first convolutional layer with trainable high-pass filters and apply their HP-FCN for inpainting detection. Kown *et al*. [28] model quantized DCT coefficient distribution to trace compression artifacts. Yang *et al*. use BayarConv [30] as the initial convolutional layer of their CR-CNN [17]. Although such constrained conv. layers are helpful for extracting noise information, using them alone has the risk of losing other useful information in the original RGB view. Hence, we see an increasing number of works on exploiting information from both the RGB view and the noise view [14], [15], [18], [26], [27], [28]. Zhou *et al*. [14] develop a two-stream Faster R-CNN,

coined RGB-N, which takes as input the RGB image and its noise counterpart generated by the spatial rich model (SRM) [29]. Rao and Ni also use SRM [27], whilst Wu *et al*. [15] and Hu *et al*. [18] use both BayarConv and SRM. Given features from distinct views, the need for feature fusion is on. Feature concatenation at an early stage is adopted by Mantra-Net [15], SPAN [18] and JPEG-ComNet [27], while CAT-Net [28] concatenates the features at a middle stage. Alternatively, MM-Net [26] performs feature fusion at an intermediate stage, where features from the noise-view branch are used as attention maps to re-weight features from the RGB-view branch. Our *MVSS-Net* is more close to RGB-N as it performs feature fusion at the late stage. However, different from the non-trainable bilinear pooling used in RGB-N, Dual Attention used in *MVSS-Net* is trainable and thus more selective.

As manipulating a specific region in a given image inevitably leaves traces between the tampered region and its surrounding, how to exploit such edge artifact also matters for manipulation detection. Salloum *et al*. develop a multi-task FCN (MFCN) to symmetrically predict a tampered area and its boundary [13]. GSR-Net has an edge detection and refinement branch which accepts features from different levels [16]. The more recent JPEG-ComNet [27] applies boundary attention on RGB view features to predict edges of manipulated areas, and subsequently utilizes the prediction to refine manipulation segmentation. Given that region segmentation and edge detection are intrinsically two distinct tasks, the challenge lies in how to strike a proper balance between the two. Directly using deeper features for edge detection as done in JPEG-ComNet has the risk of affecting the main task of manipulation segmentation, while putting all features together as used in MFCN and GSR-Net may let the deeper features be ignored by the edge branch. Our *MVSS-Net* has an edge-supervised branch that effectively resolves these issues.

Last but not least, we observe that the specificity of an image manipulation detector, *i.e*. how it responds to authentic images, is seldom reported. In fact, the mainstream solutions are developed within a semantic segmentation network. Naturally, they are trained and evaluated on manipulated images in the context of manipulation segmentation [16]. The absence of authentic images both in the training and test stages naturally raises concerns regarding the detection specificity. In this paper we make a novel attempt to include authentic images for training and test, an important step towards real-world deployment. In addition, different from the previous common practice that selects a model's decision threshold based on test data, we advocate the use of a default threshold of 0.5. Such an evaluation also matters practically.

3 Proposed Model

Given an RGB image x of size $W \times H \times 3$, we aim for a multi-head deep network $\mathcal{G}$ that not only determines whether the image has been manipulated, but also pinpoints its manipulated pixels. In particular, we let $\mathcal{G}$ have an semantic segmentation head, denoted by $\mathcal{G}_{seg}$, for producing a full-size probability map, denoted by $S(x)$, which indicates the probability of manipulation at the pixel level. We have access to the pixel-level scores via $S_{i,j}(x)$, $i = 1, \ldots, W, j = 1, \ldots, H$. Meanwhile, the network has an image classification head $\mathcal{G}_{clf}$ to output $C(x)$ the probability of the image being manipulated. As the image-level decision is naturally subject to pixel-level evidence, we derive $C(x)$ from the segmentation map:

$$\begin{cases} S(x) & \leftarrow \mathcal{G}_{seg}(x), \\ C(x) & \leftarrow \mathcal{G}_{clf}(S(x)). \end{cases} \tag{1}$$

Eq. 1 provides a high-level sketch of our network.

In order to extract generalizable manipulation detection features, $\mathcal{G}$ is designed to accept both the original RGB-view and an extra noise-view of the input image. To strike a proper balance between detection sensitivity and specificity, the multi-view feature learning process is jointly supervised by annotations of three scales, *i.e*. pixel, edge and image. All this results in Multi-View multi-Scale Supervised Networks (*MVSS-Net*).

3.1 Multi-View Feature Learning

MVSS-Net has two branches, both with ResNet-50 [37] as their backbones. The edge-supervised branch (ESB) at the top of Fig. 2 is specifically designed to exploit subtle boundary artifacts around tampered regions, whilst the noise-sensitive branch (NSB) at the bottom is to capture the noise inconsistency between tampered and authentic regions. Both clues are meant to be semantic-agnostic.

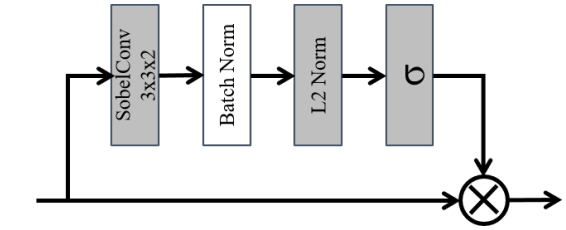
3.1.1 Edge-Supervised Branch

Ideally, with edge supervision, we hope the response area of the network will be more concentrated on tampered regions. Designing such an edge-supervised network is nontrivial. As noted in Sec. 2, the main challenge is how to construct an appropriate input for the edge detection head. On one hand, directly using features from the last ResNet block is problematic, as this will enforce the deep features to capture low-level edge patterns and consequently affect the main task of manipulation segmentation. While on the other hand, using features from the initial blocks is also questionable, as subtle edge patterns contained in these shallow features can vanish with ease after multiple deep convolutions. A joint use of both shallow and deep features is thus necessary. However, we argue that simple feature concatenation as previously used in [16] is suboptimal, as the features are mixed and there is no guarantee that the deeper features will receive adequate supervision from the edge head. To conquer the challenge, we propose to construct the input of the edge head in a shallow-to-deep manner.

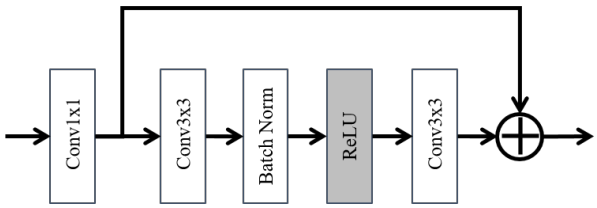
As illustrated in Fig. 2, features from different ResNet blocks are combined in a progressive manner for manipulation edge detection. In order to enhance edge-related patterns, we introduce a Sobel layer, see Fig. 3(a). The basic idea behind the Sobel layer is to discriminate edge-related pixels from others in a given feature map by attending to them with edge-related weights. In order to obtain such an attention map, we let the feature map go through the classical Sobel filter, which is widely used for identifying candidate edge pixels [38], followed by a Batch Normalization layer and an L2 Norm layer, and eventually a sigmoid (σ) layer. The feature map is then re-weighted using the attention map with element-wise multiplication.

The feature map produced by the block #i, enhanced by the Sobel layer, then goes through an edge residual block (ERB), see Fig. 3(b), before being combined (by summation) with its counterpart from the block #i+1. To prevent the effect of accumulation that unwittingly makes features from the last blocks slighted, we let the combined features go through another ERB (top in Fig. 2) before the next round of feature combination. We believe such a mechanism helps prevent extreme cases wherein deeper features

are either over-supervised or fully ignored by the edge head. By visualizing feature maps of the last ResNet block in Fig. 4, we observe that the proposed ESB indeed produces more focused responses near tampered regions.



(a) Sobel Layer



(b) Edge Residual Block (ERB)

Fig. 3. **Diagrams of (a) Sobel layer and (b) edge residual block**, used in ESB for manipulation edge detection.

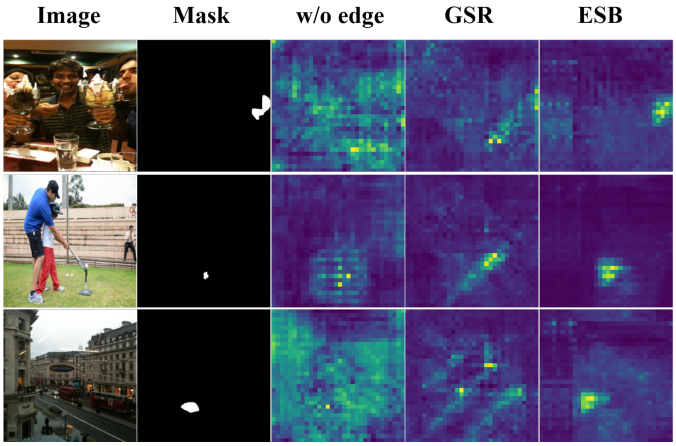


Fig. 4. **Visualization of averaged feature maps of the last ResNet block**, brighter color indicating higher responses. Manipulation from the top to bottom is inpainting, copy-move and splicing. Read from the third column are *w/o edge* (standard ResNet with no edge-related block), *GSR* (ResNet with the GSR-Net alike edge branch) and our *ESB*, which produces more focused responses near tampered regions.

The output of ESB has two parts: feature maps from the last ResNet block, denoted as $\{f_{esb,1}, \ldots, f_{esb,k}\}$, to be used for the main tasks, and the predicted manipulation edge map, denoted as $S_{edge}(x)$, obtained by transforming the output of the last ERB with a sigmoid (σ) layer. The key data flow of the ESB branch is conceptually expressed by Eq. 2,

$$\left.\begin{matrix}\{f_{esb,1}, \ldots, f_{esb,k}\} \\ S_{edge}(x)\end{matrix}\right\} \leftarrow \text{ESB}(x). \tag{2}$$

3.1.2 Noise-Sensitive Branch

In parallel to ESB, we build a noise-sensitive branch (NSB). NSB is implemented as a standard FCN (another ResNet-50 as its backbone) except for its input, which is a noise view of a given image rather than the original RGB view. Regarding the choice of noise extraction, we adopt BayarConv [30], which is found to be better than the SRM filter [17].

According to Bayar and Stamm [30], BayarConv is developed to enhance the noise inconsistency between manipulated and authentic regions within a given image. To that end, the BayarConv layer is implemented as a set of trainable prediction error filters. The response of each filter is designed to be the error between the center-pixel value of the filter window and the linear combination of the remaining pixel values within the window. More concretely, given a specific convolutional filter parameterized by ω with $\omega(0,0)$ as its center element, BayarConv imposes two constraints, *i.e.* $\omega(0,0) = -1$ and $\sum_{i,j\neq 0} \omega(i,j) = 1$. The constraints are applied on ω after each training iteration.

As Fig. 2 shows, letting the given image x go through a BayarConv layer with kernel size of $5 \times 5 \times 3$, we obtain its full-sized noise view as BayarConv(x). The output of the NSB branch is an array of k feature maps from the last ResNet block of its backbone, *i.e.*

$$\{f_{nsb,1}, \ldots, f_{nsb,k}\} \leftarrow \text{ResNet}(\text{BayarConv}(x)). \tag{3}$$

3.1.3 Branch Fusion by Dual Attention

Given two arrays of feature maps $\{f_{esb,1}, \ldots, f_{esb,k}\}$ and $\{f_{nsb,1}, \ldots, f_{nsb,k}\}$ from ESB and NSB, we propose to fuse them by a trainable Dual Attention (DA) module [39]. This is new, because previous work [14] uses bilinear pooling for feature fusion, which is non-trainable.

The DA module has two attention mechanisms working in parallel: channel attention (CA) and position attention (PA), see Fig. 5. CA associates channel-wise features to selectively emphasize interdependent channel feature maps. Meanwhile, PA selectively updates features at each position by a weighted sum of the features at all positions. The outputs of CA and PA are summed up, and transformed via a 1×1 convolution into a feature map of size $\frac{W}{16} \times \frac{H}{16}$, denoted as $S^{'}(x)$. With parameter-free bilinear upsampling followed by an element-wise sigmoid function, $S^{'}(x)$ is transformed into the full-size segmentation map $S(x)$. The DA based branch fusion is conceptually expressed as

$$\begin{cases} S^{'}(x) \leftarrow \text{DA}([f_{esb,1}, \ldots, f_{esb,k}, f_{nsb,1}, \ldots, f_{nsb,k}]), \\ S(x) \leftarrow \sigma(\text{bilinear-upsampling}(S^{'}(x))). \end{cases} \tag{4}$$

3.2 ConvGeM for Image-Level Prediction

Concerning $\mathcal{G}_{clf}$ in Eq. 1, a straightforward implementation is Global Max Pooling (GMP) as previously used in our conference paper [35]. GMP takes the maximum of $S(x)$ as $C(x)$, *i.e.* $C(x) = S_{i^*,j^*}(x)$, with $(i^*, j^*) = \arg\max_{i,j} S_{i,j}(x)$. While GMP links $C(x)$ directly to $S(x)$, we argue that this operation is suboptimal due to the following two downsides. First, as an image classification loss is practically computed based on $S_{i^*,j^*}(x)$, the gradient w.r.t. the loss is back-propagated exclusively via the sole point (i^*, j^*). Such a bottleneck not only slows down the training of the classification head, but also impedes the head from guiding the entire network. Second, GMP is invariant to the amount of positive responses and how they are spatially distributed. However, both properties matter for the pixel-level detection result to be meaningful. According to Gestalt theory [40], humans perceive visual patterns in connection with their spatial context. Following

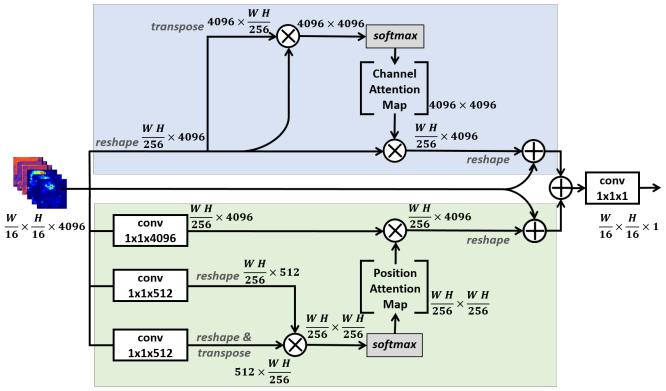


Fig. 5. **Dual Attention** (DA), with its channel attention module shown in blue and its position attention module shown in green. While DA was originally developed for capturing long-range contextual dependencies of feature maps produced by a single-branch network [39], we repurpose it for fusing feature maps from two distinct branches.

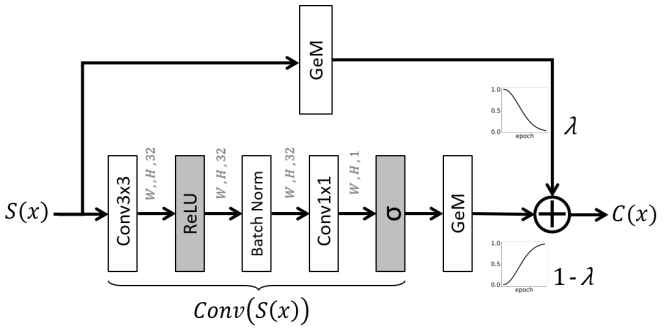


Fig. 6. **Illustration of the proposed ConvGeM module** that converts the pixel-level manipulation detection result $S(x)$ to the image-level prediction $C(x)$. The hyper parameter λ that balances GeM and GeM(Conv) decays w.r.t. the training epochs, and is dynamically determined in the training process.

this theory, for an effective deception, a certain amount of pixels in a given image have to be manipulated simultaneously with certain configurations. As such, positive responses occurring sporadically are more likely to be noise than their spatially grouped counterparts. Telling them apart is beyond the capability of GMP.

We notice that Generalized Mean pooling (GeM), originally proposed for image retrieval [41], can be used to overcome the first downside of GMP. As shown in Eq. 5, GeM uses a trainable positive parameter p to strike a balance between global mean pooling ($p = 1$) and GMP (say a larger p of 100):

$$\text{GeM}(S(x)) = \frac{1}{W \times H}\left(\sum_{i=1}^{W}\sum_{j=1}^{H} S(x_{i,j})^p\right)^{\frac{1}{p}}. \tag{5}$$

As more pixels contribute to $C(x)$, GeM effectively breaks the bottleneck of GMP in back propagation. We empirically observe that substituting GeM for GMP saves 10 training epochs approximately. Nonetheless, GeM remains invariant to the spatial distribution of the positive responses.

As convolution naturally captures spatial correlation among pixels, one might consider adding a convolutional block, denoted by $\text{Conv}(S(x))$, in advance to GeM. Consequently, $C(x)$ is obtained as $\text{GeM}(\text{Conv}(S(x)))$. Notice that in the early training epochs, the network, in particular its segmentation head $\mathcal{G}_{seg}$, has not been well trained, and thus mostly produces meaningless $S(x)$. Such noisy input to $\mathcal{G}_{clf}$ will be further exaggerated by Conv, making the classification head and consequently the entire network difficult to train. In order to suppress such a negative effect, we add $\text{GeM}(S(x))$ to $C(x)$ through a *decayed* skip connection weighed by a nonnegative hyper parameter λ as

$$C(x) = \lambda \cdot \text{GeM}(S(x)) + (1-\lambda) \cdot \text{GeM}(\text{Conv}(S(x))) \tag{6}$$

where λ is initialized with a value close to 1, and decayed nonlinearly w.r.t. the number of epochs. As illustrated in Fig 6, using a close-to-one λ lets $\mathcal{G}_{clf}$ temporarily ignore the Conv block at the early training stage. Then, as $\mathcal{G}_{seg}$ continuously improves to provide more accurate and reliable $S(x)$, λ decreases more rapidly to let $\mathcal{G}_{clf}$ count more on Conv to exploit $S(x)$ sufficiently. As Eq. 6 shows, the convex combination of GeM and GeM(Conv) with their weights dynamically determined in the training process effectively tackles the drawbacks of GMP. We coin the new module ConvGeM.

3.3 Multi-Scale Supervision

We consider losses at three scales, each with its own target, *i.e.* a pixel-scale loss for improving the model's sensitivity for pixel-level manipulation detection, an edge loss for learning semantic-agnostic features and an image-scale loss for improving the model's specificity for image-level manipulation detection.

Pixel-scale loss. As manipulated pixels are typically in minority in a given image, we use the Dice loss, found to be effective for learning from extremely imbalanced data [34]:

$$loss_{seg}(x) = 1 - \frac{2\sum_{i,j} S(x_{i,j}) \cdot y_{i,i}}{\sum_{i,j} S^2(x_{i,j}) + \sum_{i,j} y_{i,j}^2}, \tag{7}$$

where $y_{i,j} \in \{0, 1\}$ is a binary label indicating whether pixel (i, j) is manipulated.

Edge loss. As pixels of an edge are overwhelmed by non-edge pixels, we again use the Dice loss for manipulation edge detection, denoted as $loss_{edg}$. Since manipulation edge detection is an auxiliary task, we do not compute the $loss_{edg}$ at the full size of $W \times H$. Instead, the loss is computed at a smaller size of $\frac{W}{4} \times \frac{H}{4}$, see Fig. 2. This tactic reduces computational cost during training, and in the meanwhile, improves the performance slightly.

Image-scale loss. In order to reduce false alarms, authentic images have to be taken into account in the training stage. This is however nontrivial for the current works, *e.g.* Mantra-Net [15], HP-FCN [25] and GSR-Net [16], as they all rely on certain semantic segmentation losses. Consider the widely used binary cross-entropy (BCE) loss for instance. An authentic image with a small percent of its pixels misclassified contributes marginally to the BCE loss, making it difficult to effectively reduce false alarms. Also note that the Dice loss cannot handle the authentic image by definition. Therefore, an image-scale loss is needed.

As the two classes at the image level are more balanced than their counterpart at the pixel level, we adopt the BCE loss, widely used for image classification, for computing the image-scale loss:

$$loss_{clf}(x) = -(y \cdot \log C(x) + (1-y) \cdot \log(1 - C(x))), \tag{8}$$

with $y = \max(\{y_{i,j}\})$. It is worth pointing out that the usefulness of $loss_{clf}$ is not limited to improving model specificity. Through ConvGeM, the image-scale supervision can now be back propagated more effectively than our previously used GMP [35] for improving feature learning.

Combined loss. Given the losses computed at three distinct scales, we obtain a combined loss by a convex combination, *i.e*.

$$Loss = \alpha \cdot loss_{seg} + \beta \cdot loss_{clf} + (1 - \alpha - \beta) \cdot loss_{edg} \quad (9)$$

where $\alpha, \beta \in (0, 1)$ are positive weights. The combined loss is minimized by stochastic gradient descent, where authentic images in a specific mini-batch are used to compute $loss_{clf}$ only.

4 Experiments

4.1 Experimental Setup

Datasets. For a head-to-head comparison with the state-of-the-art, we adopt CASIAv2 [21] for training and widely used COVER [23], Columbia [32], NIST16 [31], CASIAv1+[2] [22] and the more recent IMD [36] for testing. Meanwhile, we notice DEFACTO [4], a recent large-scale dataset, containing 149k images sampled from MS-COCO [43] and auto-manipulated by copy-move, splicing and inpainting. Considering the challenging nature of DEFACTO, we choose to perform our ablation study on this new set. As the set has no authentic images, we construct a training set termed DEF-84k, by randomly sampling 64k positive images from DEFACTO and 20k negative images from MS-COCO. In a similar manner, we build a test set termed DEF-12k, by randomly sampling 6k positive images from the remaining part of DEFACTO and 6k negatives from MS-COCO. Note that to avoid any data leakage, for manipulated images used for training (test), their source images are not included in the test (training) set. In total, our experiments use two training sets and six test sets, see Table 2. Metadata of these sets is available at our project website[1].

TABLE 2
Two training sets and six test sets used in our experiments. The symbol – indicates information unavailable. Copy-move, splicing and inpainting are shortened as *cmpv*, *spli* and *inpa*, respectively. DEF-84k and DEF-12k are used for training and test in the ablation study (Section 4.2), while for the SOTA comparison (Section 4.3) we train on CASIAv2 and evaluate on all test sets.

Dataset	Negative	Positive	cpmv	spli	inpa
Training					
DEF-84k [4]	20,000	64,417	12,777	34,133	17,507
CASIAv2 [21]	7,491	5,063	3,235	1,828	0
Testing					
COVER [23]	100	100	100	0	0
Columbia [32]	183	180	0	180	0
NIST16 [31]	0	564	68	288	208
CASIAv1+ [22]	800	920	459	461	0
IMD [36]	414	2,010	–	–	–
DEF-12k [4]	6,000	6,000	2,000	2,000	2,000

Evaluation Criteria. For pixel-level manipulation detection, following previous works [13], [14], [16], we compute pixel-level precision and recall, and report their F1. For image-level manipulation detection, in order to measure the miss detection rate and false alarm rate, we report sensitivity, specificity and their F1. AUC as a decision-threshold-free metric is also reported. Authentic images per testset are only used for image-level evaluation.

[2]Note that the original CASIAv1 has 782 authentic images in common with CASIAv2. We fixed the issue by replacing these common images in CASIAv1 with the same amount of images randomly sampled from Corel [42], which is the data source of CASIAv1. We term the fixed version CASIAv1+.

Note that previous works commonly report performance with the decision threshold selected per testset [16], [18], [28], allowing one to compare models under their optimal conditions. However, this setting leads to overly optimistic performance estimates, as in practice, a model's decision threshold (or its operating point) has to be pre-specified and fixed. Towards real-world evaluation, for both pixel-level and image-level F1 computation, we propose to use a default threshold of 0.5, unless otherwise stated.

The overall performance is measured by Com-F1, defined as the harmonic mean of pixel-level and image-level F1. Com-F1 is sensitive to the lowest value of pixel-F1 and image-F1. In particular, it scores 0 when either pixel-F1 or image-F1 is 0, which does not hold for the arithmetic mean.

For a more complete comparison, we additionally report accuracy per test set, *i.e*. the percentage of correctly classified samples in a test set. Pixel-level / image-level accuracy is obtained by treating every pixel / image as a sample. Note that accuracy is not a reliable metric when the class distributions are highly imbalanced. So we report MCC (Matthews Correlation Coefficients) [44], a more balanced measure of a classifier's ability on both classes.

Implementation. Our models are implemented in PyTorch and trained on an NVIDIA Tesla V100 GPU. The input size is 512×512. The two ResNet-50 used in ESB and NSB are initialized with ImageNet-pretrained counterparts. We use an Adam [45] optimizer with a learning rate periodically decays from 10^{-4} to 10^{-7}. For the two hyper-parameters in the combined loss, we empirically set $\alpha = 0.16$ and $\beta = 0.04$, see a parameter sensitivity analysis in the online supplementary material[3]. As for ConvGeM, the initial value of p in the GeM used in the decayed skip connection is set to 10 so as to get a similar effect of GMP, while p of the GeM in the Conv branch is initialized to be 3 to make this GeM more close to global mean pooling. The weight λ in Eq. 6 is decayed nonlinearly w.r.t. the number of training epochs e as $\lambda = 0.9975^{(e \cdot e)}, e = 1, 2, \ldots$. An early stop occurs once the loss on a held-out validation set from DEFACTO does not decrease in 10 consecutive epochs. As such, the number of training epochs depends on the training data in use: training on CASIAv2 takes 16 epochs, meaning λ of 0.527 in the inference mode, while training on DEF-84k requires a larger number of 30 epochs, resulting in a smaller λ of 0.105.

We apply regular data augmentation for training, including flipping, blurring, compression and naive manipulations either by cropping and pasting a squared area or using built-in OpenCV inpainting functions [46], [47].

4.2 Ablation Study

To reveal the influence of the individual components, we evaluate the proposed model in varied setups with the components added progressively. All results reported in this section are obtained with DEF-84k as the training set and DEF-12k as the test set.

4.2.1 On Trainable Components

Influence of the semantic segmentation backbone. We depart from FCN-16 without multi-view multi-scale supervision. Recall that we use a DA module for branch fusion. So for a fair comparison, we adopt FCN-16 with DA, making it essentially an implementation of DANet [39]. Such an improved FCN-16 scores better than its more advanced counterparts, *e.g*. UNet [48],

[3]https://tinyurl.com/mvssnet-extra

DeepLabv3 [49] and DeepLabv3+ [50], see Table 3. The result confirms our conjecture in Section 1 that the state-of-the-art semantic segmentation networks are indeed suboptimal for manipulation detection. The competitive baseline (FCN-16 with DA) is referred to as *Seg* (Setup #0) in Table 4.

TABLE 3
Performance of different semantic segmentation backbones, trained with the segmentation loss only. F1 scores are in percentage.

Backbone	Pixel-F1	Image-F1	Image-AUC	Com-F1
U-Net	13.2	51.7	0.540	21.0
DeepLabV3	24.9	52.6	0.645	33.8
DeepLabV3+	27.9	50.9	0.651	36.0
FCN-16	33.7	69.9	0.774	45.5
FCN-16 with DA	**54.6**	**70.9**	**0.840**	**61.7**

Influence of the image classification loss. Comparing *Seg+Clf* and *Seg*, we see a clear increase in specificity and a clear drop in sensitivity, suggesting that adding $loss_{clf}$ makes the model more conservative for reporting manipulation. This change is not only confirmed by lower Pixel-F1, but is also observed in the fourth column of Fig. 7, showing that manipulated areas predicted by *Seg+Clf* are much reduced.

Influence of NSB. Since *Seg+Clf+Noise* is obtained by adding NSB into *Seg+Clf*, its better performance verifies the effectiveness of NSB for improving manipulation detection.

Influence of ESB. The better performance of *Seg+Clf+Edge* against *Seg+Clf* justifies the effectiveness of ESB.

ESB versus GSR-Net. *Seg+Clf+GSR* is obtained by replacing our ESB with the edge branch of GSR-Net [16]. The overall performance of *Seg+Clf+GSR* is lower than *Seg+Clf+Edge*. Moreover, there is a larger performance gap on *cmpv* (ESB of 0.405 versus GSR-Net of 0.363). The results clearly demonstrate the superiority of the proposed ESB over the prior art.

Influence of two branch fusion. The full setup, with ESB and NSB fused by dual attention, performs the best, showing the complementarity of the individual components. To further justify the necessity of our dual attention based fusion, we make an alternative solution which ensembles *Seg+Clf+Noise* and *Seg+Clf+Edge* by model averaging, refereed to as *Ensemble(N,E)*. Comparing Setup #6 and #7 in Table 4, we see that MVSS-Net is better than *Ensemble(N,E)*, showing the advantage of our fusion method. Comparison to fusion by bilinear pooling as used previously [14] is provided in the appendix.

Influence of $\mathcal{G}_{clf}$. We compare three different implementations of $\mathcal{G}_{clf}$, *i.e*. GMP, GeM and the proposed ConvGeM, with their performance shown in the last three rows of Table 4. Compared with GMP, GeM obtains a higher pixel-level F1, indicating a more effective usage of the image-scale supervision for improving the segmentation network. However, GeM averages the responses over pixels, albeit in a nonlinear manner, making it less sensitive, and consequently resulting in a sharp drop in image-level detection sensitivity (from 79.7 to 63.1). Its gain on the pixel-level task and its loss on image-level task cancel out each other, making Com-F1 mostly unchanged compared to GMP. By contrast, ConvGeM strikes the best balance between the two tasks, improving Com-F1 from 64.3 to 66.3.

Edge segmentation versus bounding box regression. An alternative strategy for learning boundary artifacts around manipulated regions is to treat tampering localization as a bounding-box (bbox) regression task, see [14]. To compare with this alternative, we replace the edge segmentation head, *i.e*. the sigmoid layer in Fig. 2, by the object detection head of CenterNet [51]. CenterNet performs anchor-free object detection with a two-branch head, where one branch produces a probabilistic map of each pixel being the center of an object, while the other branch is responsible for predicting object sizes. In our context, the region of an object is defined as the minimum bbox that encloses all pixels in a given tempering area. The performance of MVSS-Net trained with the object detection loss is shown in the second last row of Table 4 (Setup #9.1). Its relatively lower scores than Setup #9 in terms of both pixel-level and image-level manipulation detection suggest that the edge segmentation loss is more suited for learning boundary artifact features.

4.2.2 On Non-trainable Blocks

Influence of Sobel on ESB. *Seg+Clf+Edge/s* is obtained by removing the Sobel operation from *Seg+Clf+Edge*, so its performance degeneration in particular on copy-move detection (from 0.405 to 0.382, *cmpv* in Table 4) indicates the necessity of Sobel.

It is worth mentioning that the benefit of Sobel for tampering edge detection is concluded on the basis of the ResNet-based network architecture. Compared to ResNet, big vision models with billions of trainable parameters have shown superior performance in natural image classification [52]. Replacing the ResNet blocks used in MVSS-Net with their counterparts from the big models is likely to boost the performance of the current task. Whether the non-trainable Sobel is beneficial for such more sophisticated network architectures requires future investigation.

Enhancing NSB using non-trainable blocks? Inspired by the benefit of Sobel to ESB, we attempt to enhance NSB by using non-trainable blocks to progressively extract noise-related artifacts from the output of each ResNet block in NSB. In particular, we use a median filtering residual (MFR) block after each ResNet block. An MFR processes an input feature map by subtracting the median-filtered feature map from the input, and thus acts as a high-pass filter. The output of each MFR is incrementally aggregated in a shallow-to-deep manner similar to the ESB logic in Fig. 2. The output of the last MFR is added to the output of the last ResNet block, before the DA module. We refer to the appendix for more details. The performance of NSB with MRF is shown in Setup #9.2 in Table 4. Compared with Setup #9 (NSB w/o MFR), the higher sensitivity (81.3 *vs* 74.8) and lower specificity (79.9 *vs* 85.7) suggest that high-frequency noise patterns are sensitive to manipulation, but not sufficiently specific, resulting in more pixel-level false alarms. Therefore, the MFR benefit to image-level manipulation detection is obtained at the cost of performance degradation in pixel-level manipulation detection.

4.2.3 Qualitative Visualization

Fig. 7 shows some qualitative results of pixel-level manipulation detection. From the left to right, the results demonstrate how the propose model in varied setups strikes a balance between the detection sensitivity and specificity.

So far, our evaluation is performed on homologous training and test data. Next, we evaluate the generalization ability of our models in a cross-dataset setting, with CASIAv2 as a common training set and COVER, Columbia, NIST16, CASIAv1+, IMD and DEF-12k as the test sets.

TABLE 4
Ablation study of *MVSS-Net*. Training: DEF-84k. Test: DEF-12k. F1, Sensitivity (*Sen.*) and Specificity (*Spe.*) scores are in percentage. Best number per column is highlighted in **bold**. The steadily improved performance justifies the necessity of the individual components used in *MVSS-Net* (Setup #7) and *MVSS-Net++* (Setup #9).

Setup	Component		Pixel-level manipulation detection (F1)				Image-level manipulation detection				Com-F1
	ESB	*NSB*	*cpmv.*	*spli.*	*inpa.*	*MEAN*	*AUC*	*Sen.*	*Spe.*	*F1*	
Conference version [35], with GMP as $\mathcal{G}_{clf}$											
0: Seg	–	–	45.3	72.2	46.3	54.6	0.840	**82.7**	62.0	70.9	61.7
1: Seg+Clf	–	–	34.1	67.3	37.6	46.3	0.858	76.8	77.8	77.3	57.9
2: Seg+Clf+Noise	–	+	39.3	70.6	42.6	50.8	0.871	76.3	82.1	79.1	61.9
3: Seg+Clf+Edge	+	–	40.5	71.5	43.5	51.8	0.870	77.3	81.1	79.2	62.6
4: Seg+Clf+Edge/s	w/o sobel	–	38.2	71.0	42.2	50.5	0.869	79.2	78.9	79.0	61.6
5: Seg+Clf+GSR	GSR-Net	–	36.3	71.4	42.1	49.9	0.864	81.3	77.9	79.6	61.3
6: Ensemble(#2, #3)	+	+	38.4	70.8	43.7	51.0	0.878	73.1	87.6	79.7	62.2
7: Seg+Clf+Noise+Edge	+	+	44.6	71.4	45.5	53.8	0.886	79.7	80.2	79.9	64.3
Journal extension, built on top of Setup #7											
8: GeM as $\mathcal{G}_{clf}$	+	+	48.0	73.5	47.0	56.2	0.871	63.1	**93.0**	75.2	64.3
9: ConvGeM as $\mathcal{G}_{clf}$	+	+	**48.3**	72.8	**49.0**	**56.7**	0.879	74.8	85.7	79.9	**66.3**
9.1: Edge → BBox	+	+	47.7	**73.9**	47.1	56.2	0.884	70.6	89.4	78.9	65.7
9.2: NSB with MFR	+	+	45.5	72.4	46.0	54.6	**0.894**	81.3	79.9	**80.6**	65.1

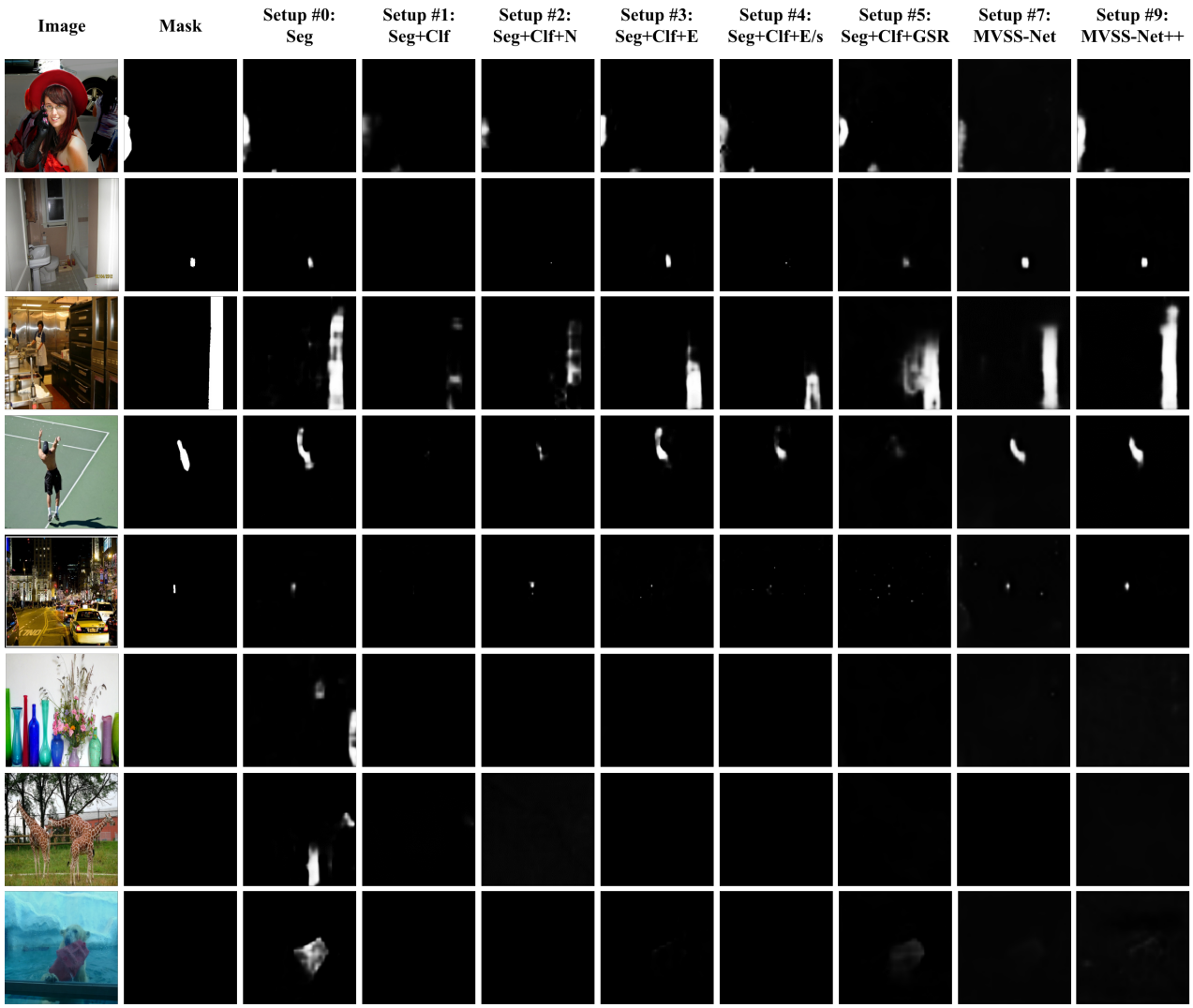


Fig. 7. **Visualizing pixel-level manipulation detection results of the proposed model in varied setups**. Data source: DEFACTO [4]. The test images in the last three rows are authentic. Setups with multi-scale supervision (Seg+Clf and afterwards) improves the detection specificity, yet at the cost of the detection sensitivity, which has to be brought back by multi-view feature learning. Among all the setups, MVSS-Net++ strikes the best balance between the detection sensitivity and specificity.

4.3 Comparison with State-of-the-art

4.3.1 Baselines

For a fair and reproducible comparison, we have to be selective, choosing the state-of-the-art that meets one of the following three criteria: 1) pre-trained models released by paper authors, 2) source code publicly available, and 3) following a common evaluation protocol where CASIAv2 is used for training and other public datasets are used for testing. Accordingly, we compile a list of nine published baselines as follows:

- Models available: H-LSTM [3] (pre-trained on a homemade dataset of 65k manipulated images and finetuned on NIST16 and IEEE Forensics Challenge data[4]), ManTra-Net [15] (trained on a private set of millions of manipulated images[5]), HP-FCN [25] (trained on a private set of inpainted images[6]), CR-CNN [17] (trained on CASIAv2[7]), SPAN [18] (trained on the same data as ManTra-Net and finetuned on CASIAv2[8]), and CAT-Net [28] (trained on a joint dataset including CASIAv2, IMD, Fantastic Reality [53], self-spliced COCO[9]). We use these six models as is.
- Code available: GSR-Net [16], which we train using author-provided code[10]. We cite their results where appropriate and use our re-trained model only when necessary.
- Same evaluation protocol: MFCN [13] and RGB-N [14] with numbers quoted from the same team [16].

For a fair comparison, we have re-trained FCN (Setup#0 in Table 4), *MVSS-Net* (Setup#7) and *MVSS-Net++*(Setup#9) from scratch on CASIAv2. As the previous works seldom report their image-level performance, an image classification head is naturally missing in their implementations. In order to obtain image-level predictions of the baselines yet with no need of hacking into their models or code, we utilize GMP as having been used in *MVSS-Net*.

4.3.2 Pixel-Level Manipulation Detection

Table 5 shows the pixel-level detection performance of the varied models. *MVSS-Net++* is the best in terms of overall performance. We attribute the better performance of ManTra-Net on DEF-12k to its large-scale training data, which was also originated from MS-COCO as DE-12k. The top performer on NIST is H-LSTM, the training data of which contains around 70% of NIST. Compared with baselines trained on the same CASIAv2, *i.e.* MFCN, RGB-N, CR-CNN and GSR-Net, *MVSS-Net++* surpasses them on almost all testsets. Its superior performance in this cross-dataset setting justifies its better generalization ability.

Comparing the left part of Table 5, which shows the models' performance in their optimal conditions, and the right part of the table, which shows the counterpart performance in a real scenario, a clear gap exists. For the best baseline, *i.e.* SPAN, its pixel-level F1 drops from 68.8 to 21.4. As for *MVSS-Net++*, its F1 drops from 73.2 to 38.7. The result shows the challenging nature of the task and the necessity of the proposed evaluation protocol for fairly assessing the technical progress towards real deployment.

4.3.3 Image-Level Manipulation Detection

Table 6 shows the image-level performance of the different models, all using the default decision threshold of 0.5. *MVSS-Net++* is again the top performer. With multi-scale supervision, the *MVSS-Net* series are able to learn from the authentic and obtains higher specificity, and thus lower false alarm rate, on most test sets. Our models also have competitive AUC scores, meaning they are better than the baselines on a wide range of operating points. Fig. 8 shows the performance curves of the individual models w.r.t. the decision threshold. The peak performance of *MVSS-Net++* is obtained at the decision value of 0.46, much closer to 0.5 than its counterparts in the other models. This result again suggests the better generalization ability of our model.

The overall performance of both pixel-level and image-level manipulation detection is provided in Table 7.

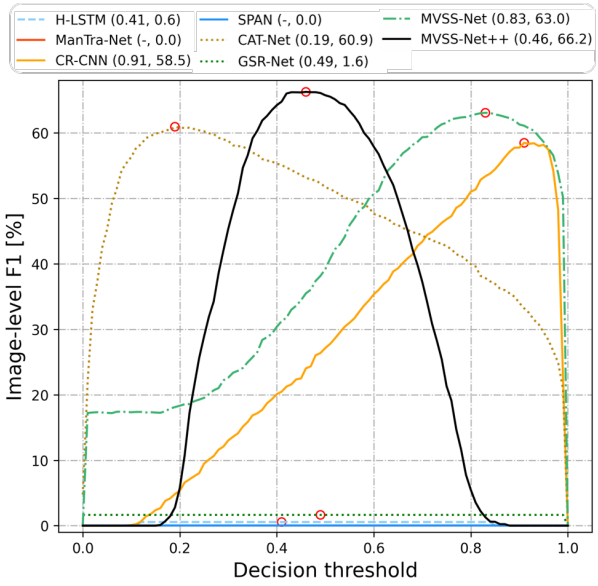


Fig. 8. **Performance curves w.r.t. the decision threshold**. Larger threshold means lower sensitivity and higher specificity. Per model, its image-level F1 score given a specific threshold is obtained by averaging the F1 scores over the five testsets, *i.e.* Columbia, CASIAv1+, COVER, DEF-12k and IMD. The two numbers following each model are its optimal threshold and the corresponding F1, as visualized in red circles.

4.3.4 Robustness Evaluation

Following [14], [15], [18], we evaluate the model robustness against two daily image processing operations when images circulate on the Internet, *i.e.* JPEG compression and Gaussian blur. Furthermore, we investigate how the current models react to manipulated images re-captured by screenshot, which to the best of our knowledge has not been done before.

Comparing the two operations, Gaussian blur affects the detection performance more severely, in particular when a larger kernel size of 17×17 and above are used, see Fig. 9. While such a larger kernel effectively erase manipulation traces, it also noticeably decreases the readability of the pictorial content (data not shown). Both *MVSS-Net* and *MVSS-Net++* exhibit better robustness than the baselines. According to their original papers, ManTra-Net and SPAN used a wide range of data augmentations including compression, while CR-CNN, GSR-Net and CAT-Net did not use such data augmentation. So for a more fair comparison, we also train *MVSS-Net++* with compression and blur excluded from data augmentation. The re-trained model, denoted as *MVSS-Net++* (w/o aug), remains more robust than the baselines.

[4]https://github.com/jawadbappy/forgery_localization_HLED
[5]https://github.com/ISICV/ManTraNet
[6]https://github.com/lihaod/Deep_inpainting_localization
[7]https://github.com/HuizhouLi/Constrained-R-CNN
[8]https://github.com/ZhiHanZ/IRIS0-SPAN
[9]https://github.com/mjkwon2021/CAT-Net
[10]https://github.com/pengzhou1108/GSRNet

TABLE 5
Performance on pixel-level manipulation detection. Performance metric: F1 [%]. Best result per test set is highlighted in bold font. All models are trained on CASIAv2, except for those marked with star (*), the training data of which contains either private (ManTra-Net, SPAN, CAT-Net and HP-FCN) or published but no longer publicly accessible data (H-LSTM).

Method	Optimal threshold per model & testset							Fixed threshold (0.5)						
	NIST	*Columbia*	*CASIAv1+*	*COVER*	*DEF-12k*	*IMD*	*MEAN*	*NIST*	*Columbia*	*CASIAv1+*	*COVER*	*DEF-12k*	*IMD*	*MEAN*
MFCN, VCIR17 [13]	42.2	61.2	54.1	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
RGB-N, CVPR18 [14]	n.a.	n.a.	40.8	37.9	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
H-LSTM*, TIP19 [3]	46.6	14.2	20.9	21.3	12.5	31.0	24.4	**35.4**	13.0	15.4	16.3	5.9	19.5	17.6
ManTra-Net*, CVPR19 [15]	45.5	70.9	69.2	77.2	**61.8**	70.5	65.9	0.0	36.4	15.5	28.6	**15.5**	18.7	19.1
HP-FCN*, ICCV19 [25]	36.0	47.1	21.4	19.9	13.6	16.9	25.8	12.1	6.7	15.4	0.3	5.5	11.2	8.5
CR-CNN, ICME20 [17]	42.8	70.4	66.2	47.0	34.0	60.0	53.4	23.8	43.6	40.5	29.1	13.2	26.2	29.4
GSR-Net, AAAI20 [16]	45.6	62.2	57.4	48.9	37.9	68.7	53.4	28.3	61.3	38.7	28.5	5.1	24.3	31.0
SPAN*, ECCV20 [18]	68.3	77.4	68.8	71.8	57.1	69.6	68.8	22.1	48.7	18.4	17.2	4.8	17.0	21.4
CAT-Net*, WACV20 [28]	59.9	**77.6**	57.3	48.5	44.1	51.7	56.5	17.9	55.5	13.6	12.9	4.6	5.4	18.3
This paper:														
FCN	50.7	58.6	74.2	57.3	40.1	64.5	57.6	16.7	22.3	44.1	19.9	13.0	21.0	22.8
MVSS-Net	**73.7**	70.3	75.3	82.4	57.2	75.7	72.4	29.2	63.8	45.2	45.3	13.7	26.0	37.2
MVSS-Net++	71.5	73.1	**77.1**	**83.2**	55.6	**76.2**	**72.8**	30.4	**66.0**	**51.3**	**48.2**	9.5	**27.0**	**38.7**

TABLE 6
Performance on image-level manipulation detection. Decision threshold: 0.5. NIST16 is excluded as it has no authentic image. *MVSS-Net++* tops the performance with image-level F1 of 68.0 averaged over the five test sets, followed by CAT-Net (51.7) and *MVSS-Net* (51.2).

Method	Columbia				CASIAv1+				COVER				DEF-12k				IMD			
	AUC	*Sen.*	*Spe.*	*F1*	*AUC*	*Sen.*	*Spe.*	*F1*	*AUC*	*Sen.*	*Spe.*	*F1*	*AUC*	*Sen.*	*Spe.*	*F1*	*AUC*	*Sen.*	*Spe.*	*F1*
H-LSTM	0.506	100.0	1.1	2.2	0.498	99.7	0.0	0.0	0.500	100.0	0.0	0.0	0.499	99.9	0.1	0.2	0.501	100.0	0.0	0.0
ManrTra-Net	0.701	100.0	0.0	0.0	0.500	100.0	0.0	0.0	0.500	100.0	0.0	0.0	0.543	100.0	0.0	0.0	0.500	100.0	0.0	0.0
CR-CNN	0.783	96.1	24.6	39.2	0.719	93.0	13.9	24.2	0.566	96.7	7.0	13.1	0.567	77.4	26.7	39.7	0.615	92.9	12.3	21.7
GSR-Net	0.502	100.0	1.1	2.2	0.500	99.4	0.0	0.0	0.515	100.0	0.0	0.0	0.456	91.4	0.1	0.2	0.500	100.0	0.0	0.0
SPAN	0.500	100.0	0.0	0.0	0.500	100.0	0.0	0.0	0.500	100.0	0.0	0.0	0.500	100.0	0.0	0.0	0.500	100.0	0.0	0.0
CAT-Net	0.971	87.2	96.2	91.5	0.647	23.9	92.1	38.0	0.557	28.0	80.0	41.5	0.543	34.2	72.5	46.5	0.586	27.5	81.6	41.1
FCN	0.762	95.0	32.2	48.1	0.770	72.8	64.3	68.3	0.541	90.0	10.0	18.0	0.551	71.1	33.8	45.8	0.502	84.6	15.5	26.2
MVSS-Net	0.980	66.9	100.0	80.2	**0.937**	61.5	98.8	**75.8**	**0.731**	94.0	14.0	24.4	**0.573**	81.7	26.8	40.4	0.656	91.5	22.0	35.5
MVSS-Net++	**0.984**	96.7	89.6	**93.0**	0.862	53.6	98.4	69.4	0.726	69.0	68.0	**68.5**	0.531	37.3	66.6	**47.8**	**0.658**	59.5	63.5	**61.4**

TABLE 7
Overall performance measured by Com-F1, the harmonic mean of pixel-level F1 and image-level F1, on five test sets.

Method	Columbia	CASIAv1+	COVER	DEF-12k	IMD	MEAN
H-LSTM	3.8	0.0	0.0	0.4	0.0	0.8
ManrTra-Net	0.0	0.0	0.0	0.0	0.0	0.0
CR-CNN	41.3	30.3	18.1	19.8	23.7	26.6
GSR-Net	4.2	0.0	0.0	0.4	0.0	0.9
SPAN	0.0	0.0	0.0	0.0	0.0	0.0
CAT-Net	69.1	20.0	19.7	8.4	9.5	25.3
FCN	30.5	53.6	18.9	20.3	23.3	29.3
MVSS-Net	71.1	56.6	31.7	**20.5**	30.0	42.0
MVSS-Net++	**77.2**	**59.0**	**56.6**	15.8	**37.5**	**49.2**

The screenshot oriented evaluation is conducted as follows. A subset of 100 manipulated images are randomly selected from CASIAv1+. Each image is then manually re-captured on a Windows10 laptop with a screen resolution of 1920×1080. Three commercial screenshot tools are used, including Microsoft *Snip&Sketch*[11], Google *Chrome DevTools*[12], and *Snipaste*[13]. Per image and tool, the result image is saved in jpg (with a quality level of 90%) and png formats, respectively, except for Chrome which supports png only. Varying the configuration of the screen tool and the image format results into five variants of the test set, *i.e*. Snip&Sketch (png), Snip&Sketch (jpg), Chrome (png), Snipaste (png) and Snipaste (jpg).

Fig. 10 shows the performance of the individual models on the original test set and its variants. We draw two conclusions from the figure. First, concerning the two factors for image re-capturing, *i.e*. screenshot tool and file format, the latter is more important. Second, while all models suffer from screenshot based image re-capturing, *MVSS-Net++* remains the best.

4.3.5 Efficiency Test

We measure the inference efficiency in terms of frames per second (FPS), tested on two GPU cards, *i.e*. NVIDIA Tesla V100 (32GB GPU memory footprint) and GeForce RTX2080ti (12GB GPU memory footprint), respectively. Depending on the card in use,

[11] https://www.microsoft.com/en-us/p/snip-sketch
[12] https://developer.chrome.com/blog/new-in-devtools-74/#screenshot
[13] https://www.snipaste.com/

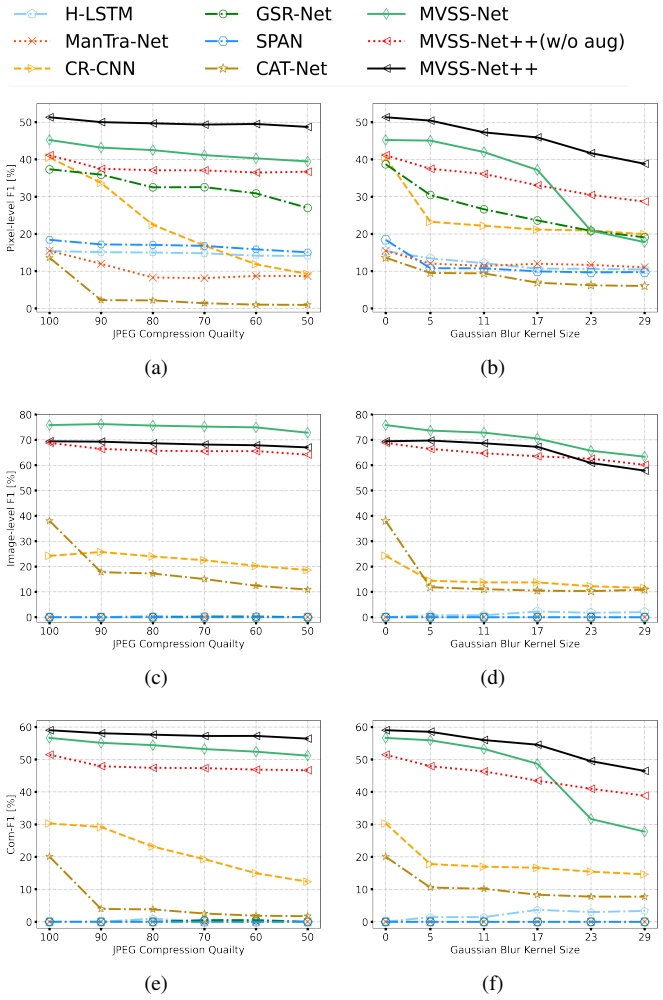


Fig. 9. **Robustness evaluation against two image processing techniques, *i.e.* JPEG compression and Gaussian Blurs**. Test set: CASIAv1+. *MVSS-Net++* (w/o aug) indicates training with JPEG compression and Gaussian blur excluded from data augmentation. The proposed models are more robust than the baselines.

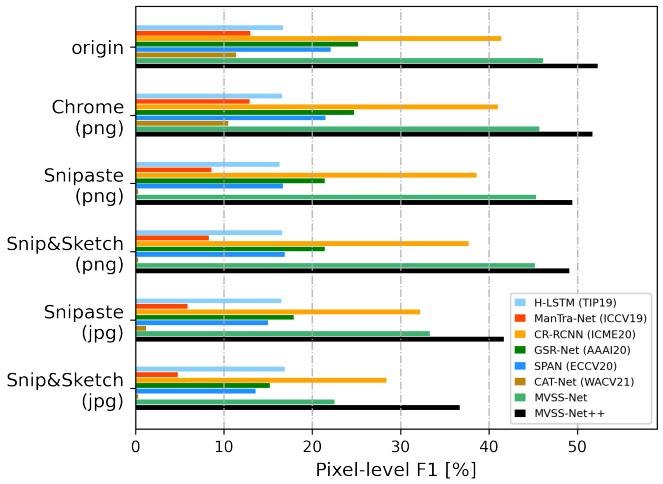


Fig. 10. **Detection performance on images re-captured by three screenshot tools (Snip&Sketch, Chrome and Snipaste) and saved in two different formats (jpg and png)**. Results are sorted by the performance of *MVSS-Net++* in descending order.

the *MVSS-Net* series run at 16 to 20 FPS, see Table 8. The relatively high FPS as compared to the other publicly available models permits the *MVSS-Net* series for real-time application.

TABLE 8
Model inference speed, tested on two NVIDA GPU cards respectively. Performance metric: Frames per second (FPS). Models are sorted in descending order in terms of their FPS on RTX2028ti, which is much cheaper than V100 and thus more affordable.

Model	Tesla V100	RTX2080ti
MVSS-Net	20.1	**19.0**
MVSS-Net++	19.0	16.0
GSR-Net	**31.7**	9.8
SPAN	8.4	8.1
H-LSTM	6.5	5.4
CAT-Net	5.4	4.1
C-RCNN	2.8	2.2
ManTra-Net	3.1	2.1

4.3.6 *Failure Case Analysis*

Given the challenging nature of the task, failures are inevitable, see Fig. 11. The first-row image was manipulated by darkening the frame of the spectacle the kid was wearing. Such manipulation traces appear to be too tiny to be revealed by the current models. The top-right corner of the second-row image was overlaid with certain translucent image patch, with the manipulated traces well melting into the misty scene. As for the last image, manipulation was performed by putting a knight on the back of the dog in the foreground, while blurring the background. All models capture the inconsistency between the foreground and the background, yet all fail to recognize that the background was actually manipulated.

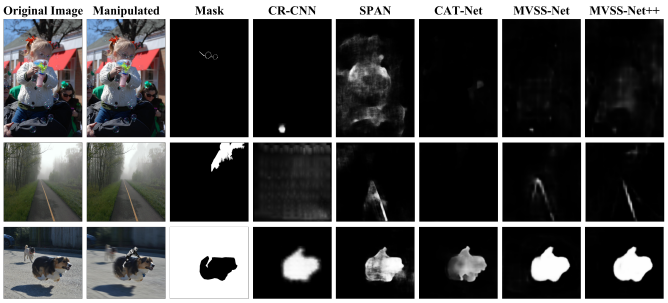


Fig. 11. **Failure cases**. Data source: IMD.

5 Conclusions

For learning semantic-agnostic features, both noise and edge information are helpful, whilst the latter is better when used alone. For exploiting the edge information, our proposed edge-supervised branch (ESB) is more effective than the previously used feature concatenation. ESB steers the network to be more concentrated on tampered regions. Regarding the specificity of manipulation detection, we empirically show that the state-of-the-arts suffer from poor specificity. The inclusion of the image classification loss improves the specificity, yet at the cost of a clear performance drop for pixel-level manipulation detection. To avoid such a loss, multi-view feature learning has to be used together with multi-scale supervision. The resultant *MVSS-Net++* is a new state-of-the-art for image manipulation detection, outperforming the current

methods in both within-dataset and cross-dataset scenarios. It also exhibits better robustness against JPEG compression, Gaussian blur and screenshot based image re-capturing.

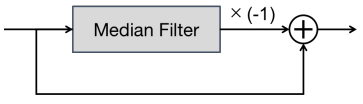
With the initial success of MVSS-Net, we believe it will be promising to design a more complex network that contains more components to cover other information (*e.g.* compression artifacts) and other modalities (*e.g.* associated text) for media forensics.

APPENDIX

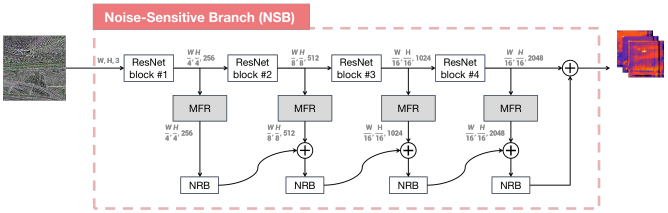
Additional measures. Table 9 shows accuracy and MCC scores of different models. The MVSS-Net series clearly outperform the baselines in terms of the well balanced MCC.

TABLE 9
Detection performance measured by accuracy and MCC.

Method	NIST	Columbia	CASIAv1+	COVER	DEF-12k	IMD	MEAN
Pixel-level accuracy (%):							
H-LSTM	92.8	69.2	90.3	87.4	94.3	90.9	87.5
ManrTra-Net	92.5	74.6	88.2	90.2	96.9	92.3	89.1
C-RCNN	**92.9**	77.2	80.5	89.1	96.8	91.4	88.0
GSR-Net	88.4	80.3	87.9	84.9	93.7	90.2	87.6
SPAN	88.8	77.3	91.6	88.2	95.0	91.0	88.7
CAT-Net	92.1	**82.0**	91.9	90.0	96.8	92.5	**90.9**
FCN	92.4	70.8	93.6	88.0	96.8	**92.4**	89.0
MVSS-Net	90.1	77.6	**94.0**	91.1	**97.0**	91.1	90.2
MVSS-Net++	90.5	66.0	93.1	**91.4**	96.8	91.0	88.1
Image-level accuracy (%):							
H-LSTM	92.8	50.1	53.3	50.0	50.0	**82.9**	63.2
ManrTra-Net	92.5	49.6	53.5	50.0	50.0	**82.9**	63.1
C-RCNN	**92.9**	60.1	56.2	51.9	52.1	79.1	65.4
GSR-Net	88.4	50.1	53.2	50.0	45.8	**82.9**	61.7
SPAN	88.8	49.6	53.5	50.0	50.0	**82.9**	62.5
CAT-Net	92.1	91.7	55.6	54.0	53.4	36.7	63.9
FCN	92.4	63.3	68.8	50.0	52.5	72.8	66.6
MVSS-Net	90.1	83.6	**78.8**	54.0	**54.3**	79.6	**73.4**
MVSS-Net++	90.5	**93.1**	74.4	**68.5**	52.0	60.2	73.1
Pixel-level MCC [-1, 1]:							
H-LSTM	**0.351**	0.124	0.138	0.131	0.046	0.182	0.162
ManrTra-Net	0.000	0.365	0.092	0.313	**0.175**	0.194	0.190
C-RCNN	0.232	0.408	0.380	0.273	0.140	0.254	0.281
GSR-Net	0.257	0.518	0.178	0.228	0.083	0.224	0.248
SPAN	0.203	0.444	0.190	0.164	0.039	0.161	0.200
CAT-Net	0.175	0.518	0.138	0.127	0.048	0.058	0.177
FCN	0.151	0.194	0.425	0.154	0.113	0.212	0.208
MVSS-Net	0.279	0.492	0.447	0.437	0.099	0.256	0.335
MVSS-Net++	0.289	**0.545**	**0.503**	**0.464**	0.097	**0.265**	**0.361**
Image-level MCC [-1, 1]:							
H-LSTM	–	0.074	-0.039	0.000	-0.009	-0.009	0.003
ManTra-Net	–	0.000	0.000	0.000	0.000	0.000	0.000
C-RCNN	–	0.295	0.114	0.084	0.048	0.073	0.123
GSR-Net	–	0.074	-0.053	0.000	-0.208	0.000	-0.037
SPAN	–	0.000	0.000	0.000	0.000	0.000	0.000
CAT-Net	–	0.838	0.216	0.094	0.072	0.078	0.259
FCN	–	0.349	0.372	0.000	0.053	0.001	0.155
MVSS-Net	–	0.710	**0.637**	0.133	**0.102**	0.163	0.349
MVSS-Net++	–	**0.865**	0.569	**0.370**	0.041	**0.174**	**0.404**



(a) A median filtering residual block (MFR)



(b) NSB with MFR

Fig. 12. **Diagrams of (a) non-trainable MFR and (b) NSB with MFR.**

NSB with MFR. Fig. 12 shows how to add non-trainable MFR blocks to NSB, in a shallow-to-deep manner similar to ESB. NRB (noise residual block) is implemented in the same manner as ERB (edge residual block) in Fig. 3(b).

ACKNOWLEDGMENTS

This research was supported by NSFC (62172420, U1703261), BJNSF (4202033), the Fundamental Research Funds for the Central Universities and the Research Funds of Renmin University of China (No. 18XNLG19), and Public Computing Cloud, Renmin University of China. The authors thank the anonymous reviewers for their insightful feedback.

REFERENCES

[1] O. Gafni and L. Wolf, "Wish you were here: context-aware human generation," in *CVPR*, 2020. 1

[2] J. Bappy, A. Roy-Chowdhury, J. Bunk, L. Nataraj, and B. Manjunath, "Exploiting spatial structure for localizing manipulated image regions," in *ICCV*, 2017. 1

[3] J. Bappy, C. Simons, L. Nataraj, B. Manjunath, and A. Roy-Chowdhury, "Hybrid lstm and encoder-decoder architecture for detection of image forgeries," *IEEE Transactions on Image Processing*, vol. 28, no. 7, pp. 3286–3300, 2019. 1, 1, 4.3.1, 5

[4] G. Mahfoudi, B. Tajini, F. Retraint, F. Morain-Nicolier, and M. Pic, "DEFACTO: Image and face manipulation dataset," in *EUSIPCO*, 2019. 1, 4.1, 2, 7

[5] L. Verdoliva, "Media forensics and deepfakes: An overview," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 5, pp. 910–932, 2020. 1, 1

[6] T. Bianchi and A. Piva, "Image forgery localization via block-grained analysis of jpeg artifacts," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 1003–1017, 2012. 1

[7] Z. Lin, J. He, X. Tang, and C. Tang, "Fast, automatic and fine-grained tampered jpeg image detection via DCT coefficient analysis," *Pattern Recognition*, vol. 42, no. 11, pp. 2492–2501, 2009. 1

[8] E. Kee, J. O'Brien, and H. Farid, "Exposing photo manipulation with inconsistent shadows," *ACM Transaction on Graphics*, vol. 32, no. 3, 2013. 1

[9] M. Johnson and H. Farid, "Exposing digital forgeries by detecting inconsistencies in lighting," in *MM & Sec*, 2005, pp. 1–10. 1

[10] W. Fan, K. Wang, and F. Cayre, "General-purpose image forensics using patch likelihood under image statistical models," in *WIFS*, 2015. 1

[11] S. Lyu, X. Pan, and X. Zhang, "Exposing region splicing forgeries with blind local noise estimation," *International Journal of Computer Vision*, vol. 110, no. 2, pp. 202–221, 2014. 1

[12] C. Pasquini, I. Amerini, and G. Boato, "Media forensics on social media platforms: a survey," *EURASIP Journal on Information Security*, no. 4, 2021. 1

[13] R. Salloum, Y. Ren, and C. Kuo, "Image splicing localization using a multi-task fully convolutional network (MFCN)," *Journal of Visual Communication and Image Representation*, vol. 51, pp. 201–209, 2018. 1, 1, 2, 4.1, 4.3.1, 5

[14] P. Zhou, X. Han, V. Morariu, and L. Davis, "Learning rich features for image manipulation detection," in *CVPR*, 2018. 1, 1, 2, 3.1.3, 4.1, 4.2.1, 4.3.1, 4.3.4, 5

[15] Y. Wu, W. AbdAlmageed, and P. Natarajan, "ManTra-Net: Manipulation tracing network for detection and localization of image forgeries with anomalous features," in *CVPR*, 2019. 1, 1, 2, 3.3, 4.3.1, 4.3.4, 5

[16] P. Zhou, B. Chen, X. Han, M. Najibi, and L. Davis, "Generate, segment, and refine: Towards generic manipulation segmentation," in *AAAI*, 2020. 1, 1, 2, 3.1.1, 3.3, 4.1, 4.2.1, 4.3.1, 5

[17] C. Yang, H. Li, F. Lin, B. Jiang, and H. Zhao, "Constrained R-CNN: A general image manipulation detection model," in *ICME*, 2020. 1, 1, 2, 3.1.2, 4.3.1, 5

[18] X. Hu, Z. Zhang, Z. Jiang, S. Chaudhuri, Z. Yang, and R. Nevatia, "SPAN: Spatial pyramid attention network for image manipulation localization," in *ECCV*, 2020. 1, 1, 2, 4.1, 4.3.1, 4.3.4, 5

[19] D. Cozzolino, G. Poggi, and L. Verdoliva, *Data-Driven Digital Integrity Verification*. Springer Singapore, 2022, pp. 281–311. 1

[20] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2018. 1

[21] J. Dong, W. Wang, and T. Tan, "CASIA image tampering detection evaluation database," in *ChinaSIP*, 2013. 1, 1, 4.1, 2

[22] ——, "CASIA image tampering detection evaluation database," http://forensics.idealtest.org, 2010. 1, 1, 4.1, 2

[23] B. Wen, Y. Zhu, R. Subramanian, T. T. Ng, and S. Winkler, "COVERAGE – A novel database for copy-move forgery detection," in *ICIP*, 2016. 1, 4.1, 2

[24] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640–651, 2015. 1

[25] H. Li and J. Huang, "Localization of deep inpainting using high-pass fully convolutional network," in *ICCV*, 2019. 1, 2, 3.3, 4.3.1, 5

[26] C. Yang, Z. Wang, H. Shen, H. Li, and B. Jiang, "Multi-modality image manipulation detection," in *ICME*, 2021. 1, 2

[27] Y. Rao and J. Ni, "Self-supervised domain adaptation for forgery localization of JPEG compressed images," in *ICCV*, 2021. 1, 2

[28] M. Kwon, I. Yu, S. Nam, and H. Lee, "CAT-Net: Compression artifact tracing network for detection and localization of image splicing," in *WACV*, 2021. 1, 2, 4.1, 4.3.1, 5

[29] J. Fridrich and J. Kodovsky, "Rich models for steganalysis of digital images," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 868–882, 2012. 1, 2

[30] B. Bayar and M. Stamm, "Constrained convolutional neural networks: A new approach towards general purpose image manipulation detection," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 11, pp. 2691–2706, 2018. 1, 2, 3.1.2

[31] H. Guan, M. Kozak, E. Robertson, Y. Lee, A. N. Yates, A. Delgado, D. Zhou, T. Kheyrkhah, J. Smith, and J. Fiscus, "MFC datasets: Large-scale benchmark datasets for media forensic challenge evaluation," in *WACVW*, 2019. 1, 4.1, 2

[32] J. Hsu, "Columbia uncompressed image splicing detection evaluation dataset," https://www.ee.columbia.edu/ln/dvmm/downloads/AuthSplicedDataSet/AuthSplicedDataSet.htm, 2009. 1, 4.1, 2

[33] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Learning a discriminative feature network for semantic segmentation," in *CVPR*, 2018. 1

[34] Q. Wei, X. Li, W. Yu, X. Zhang, Y. Zhang, B. Hu, B. Mo, D. Gong, N. Chen, D. Ding, and Y. Chen, "Learn to segment retinal lesions and beyond," in *ICPR*, 2020. 1, 3.3

[35] X. Chen, C. Dong, J. Ji, J. Cao, and X. Li, "Image manipulation detection by multi-view multi-scale supervision," in *ICCV*, 2021. 1, 3.2, 3.3, 4

[36] A. Novozamsky, B. Mahdian, and S. Saic, "IMD2020: A large-scale annotated dataset tailored for detecting manipulated images," in *WACVW*, 2020. 1, 4.1, 2

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016. 3.1

[38] R. Gonzalez and R.Woods, *Digital image processing*. Pearson Education, 2009. 3.1.1

[39] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," in *CVPR*, 2019. 3.1.3, 5, 4.2.1

[40] K. Koffka, "Perception: An introduction to the Gestalt-Theorie." *Psychological Bulletin*, vol. 19, no. 10, pp. 531–585, 1922. 3.2

[41] R. Filip, T. Giorgos, and C. Ondrej, "Fine-tuning CNN image retrieval with no human annotation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 7, pp. 1655–1668, 2018. 3.2

[42] J. Wang, J. Li, and G. Wiederhold, "Simplicity: Semantics-sensitive integrated matching for picture libraries," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 9, pp. 947–963, 2001. 2

[43] T. Lin, M. Maire, S. Belongie, J. Hays, and C. Zitnick, "Microsoft COCO: Common objects in context," in *ECCV*, 2014. 4.1

[44] P. Baldi, S. Brunak, Y. Chauvin, C. A. F. Andersen, and H. Nielsen, "Assessing the accuracy of prediction algorithms for classification: an overview ," *Bioinformatics*, vol. 16, no. 5, pp. 412–424, 2000. 4.1

[45] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. 4.1

[46] A. Telea, "An image inpainting technique based on the fast marching method," *Journal of Graphics Tools*, vol. 9, no. 1, pp. 23–34, 2004. 4.1

[47] M. Bertalmio, A. Bertozzi, and G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting," in *CVPR*, 2001. 4.1

[48] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015. 4.2.1

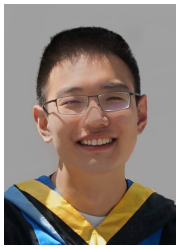
[49] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," in *CVPR*, 2017. 4.2.1

[50] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *ECCV*, 2018. 4.2.1

[51] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," in *arXiv preprint arXiv:1904.07850*, 2019. 4.2.1

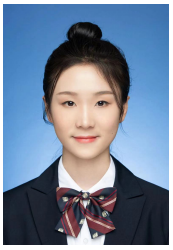
[52] C. Riquelme, J. Puigcerver, B. Mustafa, M. Neumann, R. Jenatton, A. S. Pinto, D. Keysers, and N. Houlsby, "Scaling vision with sparse mixture of experts," in *NeurIPS*, 2021. 4.2.2

[53] V. V. Knyaz, V. Knyaz, and F. Remondino, "The point where reality meets fantasy: mixed adversarial generators for image splice detection," in *NeurIPS*, 2019. 4.3.1

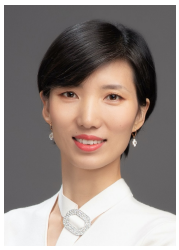


Chengbo Dong received his B.S. degree in automation from Beihang University, Beijing, China in 2020. He is currently a master student at the AIMC Lab, School of Information, Renmin University of China, pursuing his master degree on multimedia forensics.

Xinru Chen received her B.S. degree in Computer Science from the College of Information, Renmin University of China, Beijing, China in 2020. She is currently a master student at the AIMC Lab, School of Information, Renmin University of China, pursuing her master degree on multimedia forensics.



Ruohan Hu is an undergraduate at the College of Information, Beijing Forestry University. She is currently a research intern at the AIMC Lab, School of Information, Renmin University of China.



Juan Cao received her Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2008. She is currently a Full Professor with the same institute. Her research interests include multimedia content analysis, fake news detection and forgery detection.

Xirong Li received the B.S. and M.E. degrees from Tsinghua University, Beijing, China, in 2005 and 2007, respectively, and the Ph.D. degree from the University of Amsterdam, Amsterdam, The Netherlands, in 2012, all in computer science. He is currently an Associate Professor with the Key Lab of Data Engineering and Knowledge Engineering, Renmin University of China, Beijing, China. He leads the AIMC Lab at RUC. His research focuses on multimedia intelligence. Dr. Li was recipient of the ACMMM 2016 Grand Challenge Award, the ACM SIGMM Best Ph.D. Thesis Award 2013, the IEEE TRANSACTIONS ON MULTIMEDIA Prize Paper Award 2012, and the Best Paper Award of ACM CIVR 2010. He served as program co-chair of the Multimedia Modeling 2021 conference and is serving as associate editor of ACM TOMM and the Multimedia Systems Journal.