TrainFors: A Large Benchmark Training Dataset for Image Manipulation Detection and Localization

Soumyaroop Nandi, Prem Natarajan, Wael Abd-Almageed
USC Information Sciences Institute, Marina del Rey, CA, USA
{soumyarn,pnataraj,wamageed}@isi.edu

Abstract

The evaluation datasets and metrics for image manipulation detection and localization (IMDL) research have been standardized. But the training dataset for such a task is still nonstandard. Previous researchers have used unconventional and deviating datasets to train neural networks for detecting image forgeries and localizing pixel maps of manipulated regions. For a fair comparison, the training set, test set, and evaluation metrics should be persistent. Hence, comparing the existing methods may not seem fair as the results depend heavily on the training datasets as well as the model architecture. Moreover, none of the previous works release the synthetic training dataset used for the IMDL task. We propose a standardized benchmark training dataset for image splicing, copy-move forgery, removal forgery, and image enhancement forgery. Furthermore, we identify the problems with the existing IMDL datasets and propose the required modifications. We also train the state-of-the-art IMDL methods on our proposed TrainFors[1] dataset for a fair evaluation and report the actual performance of these methods under similar conditions.

1. Introduction

Image manipulation and the effects of such acts have become a challenging problem in today's society, owing to the low-cost, publicly accessible image editing tools ([50], [40], [63], [17]) and photo-realistic generative models like GANs([25], [47], [78]) and VAEs([36], [52]) for creating manipulated images. Deceitful attackers may use such tools to spread misinformation like deep fakes ([56]), fake news ([33]), plagiarised academic publications ([57]), internet rumors ([68]), forged satellite images ([30]). A news article ([64]) highlights the usage of manipulated images and videos in the Russia-Ukraine war to spread misinformation. Defending such manipulated misinformation spread is the need of the hour and image manipulation detection and localization is an effort to counter such societal problems.

[1] https://github.com/vimal-isi-edu/TrainFors

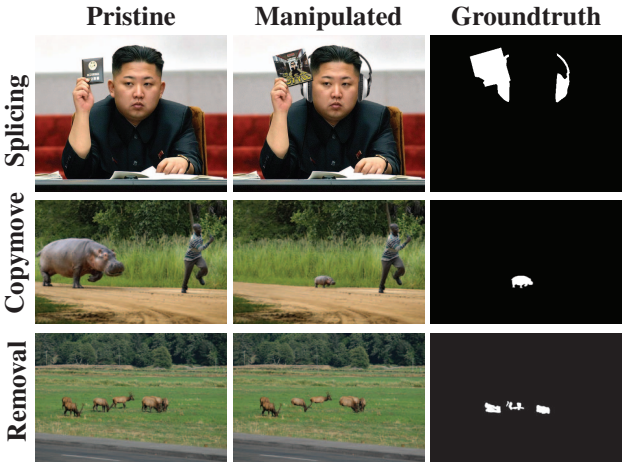

Figure 1: Image Manipulation Localization Examples - From top to bottom, each row shows Image Splicing, Copymove Manipulation, and Removal Manipulation. The binary mask on the third column represents the localized manipulated pixels.

The problem can be divided into two subtasks - *detecting* manipulated images and *localizing* a pixel map of the manipulated region in the forged images. Generally, an image can be tampered in two ways - using image content or without using any image content. Image content can either be moved from one part of the image to another internally (*copy-move forgery*) or externally import certain parts from an alien image (*splicing forgery*). Some forgers try to remove certain image patches or objects and replace them with surrounding inpainted pixels (*removal forgery*). Besides this, image forgeries can also be executed without modifying the image content, but by *image enhancement* (e.g., compression, resampling, blurring, noise, morphing, quantization, histogram manipulation). see Fig. 1 and Fig. 2

The image manipulation process often leads to artifacts around the tampered patches. Researchers have used various clues like noise pattern ([73], [44], [15]), camera model parameters ([10], [9]), edge inconsistencies ([58], [75]), color inconsistency ([20]), EXIF inconsistencies ([33]), visual similarities ([69], [70], [71]) and JPEG compression artifacts ([27], [62], [41]) to detect artifacts in forged im-

Figure 2: Image Enhancement Manipulations, without modifying the image content, are difficult to detect - From left to right: Pristine, Double JPEG Compressed, Lancoz Resampled, Histogram Manipulation, and Dithering Quantization.

ages. Most of the previous works have tried to focus on one or two clues to detect and localize manipulations. Also, they localized the manipulated patches or pixels on the tampered images. But in the real-world scenario, we never know if the image is manipulated or pristine. Neither do we know all the types of manipulations applied on an image and post-processing performed to hide such malicious acts.

Recent DNN-based object detection methods ([80], [19], [29], [32]) try to detect single or multiple objects in an image. But our task is to determine the tampered pixels in a given image. This task is somewhat more challenging because an attacker removes/modifies certain pixels and replaces those pixels with new objects/backgrounds. An object detector may perform semantic segmentation to distinguish all the objects in an image, but we can never know if any of those objects were tampered with in the first place. This necessitates the development of image manipulation detection and localization (IMDL) models, working in parallel with an object detector.

Previous research works do not use a standard training dataset for developing the IMDL models. They have claimed to improve the AUC and F1 scores with a new neural network architecture, trained on a curated synthetic training dataset that varies for each model. Moreover, neither do they release the synthetic training dataset and the training code to reproduce the claimed results nor do they elaborately describe the synthetic training dataset generation procedure. Only the inference code and pre-trained model weights are released for some of the existing research. This violates the repeatability of scientific experimentation in the IMDL tasks. To overcome this massive flaw in the IMDL experimentation, we propose a new benchmark dataset - TrainFors. We hope that TrainFors will standardize the IMDL algorithm designing and software development. TrainFors contain 1 million images (200K pristine, 800K manipulated), belonging to four standardized image forgery tasks - image splicing, copy-move forgery, image inpainting (removal), and image enhancement.

The main contributions of our paper are as follows:

- A large standardized benchmark training dataset with real-world forgeries for the image manipulation detection and localization (IMDL) task.
- Extensive analysis of the baseline IMDL models, trained from scratch on our proposed TrainFors dataset. All the baseline models are trained under similar conditions and evaluated with the same evaluation metrics for a fair comparison.
- The major challenges in curating a training dataset for IMDL tasks are elaborately described and intelligent solutions are proposed to minimize the difference between real-world manipulated images and the forged training images.

2. Related Work

Most of the previous works have focused on one or multiple forgeries like boundary artifacts, noise pattern-based, double JPEG compression-based, color filter array-based to detect and localize the tampered pixels. Some researchers used handcrafted features and others have trained a deep neural network (DNN) to find the image forgeries.

2.1. Handcrafted feature-based methods

The most prominent handcrafted feature-based methods used in image forgery detection and localization are **ELA** [38] (finds the compression error difference between forged regions and pristine regions through different JPEG compression qualities), **NOI1** [45] (models local noise by using high pass wavelet coefficients) and **CFA1** [21] (approximates the camera filter array patterns by using nearby pixels and generates a tampering probability for each pixel). They studied only a single type of forgery (among splicing, copy-move, and inpainting), splicing being the most common.

2.2. Early DNN-based models and training datasets

The earlier DNN-based studies predominantly detected a specific type of manipulation, e.g., splicing ([14], [69], [37]), copy-move ([13], [53], [70], [71], [34]), removal ([79]) and enhancement ([6], [7], [12]). But we do not have apriori information about the type of forgery implemented on real-world manipulated images. So general forgery detection and localization algorithms were proposed that could detect any type of unknown manipulation in an image. **J-LSTM** [4] and **H-LSTM** [5] jointly trained an LSTM and CNN to capture the boundary-discriminative features by training a synthetic dataset (unavailable) sampled from MS-COCO [42], NIST16 [2], and Dresden database [24]. Both methods detected predefined size regions (restricted by preset patch size) and are very time-consuming. **RGB-N** [77] used a synthetic training dataset (unavailable) sampled from MS-COCO [42] and adopted a steganalysis-based SRM kernel model [22] and a two-stream Faster R-CNN [54], but it cannot generate pixel-wise segmentation masks but only bounding boxes around tampered regions. **ManTraNet** [72] used a synthetic training dataset (unavailable) sampled from KCMI [1] and Dresden database [24] and classified 385 unknown manipulation types and trained bipartite end-to-end network to detect image-level manipulations with one part using SRM kernel [22] in feature extraction

and training, while the second subnet used Bayar convolution on synthetic forgery datasets followed by pixel-wise anomaly detection, but failed for double JPEG compression artifacts. **SPAN** ([31]) used the synthetic dataset created by ManTraNet (unavailable) and used a pyramid structure of local self-attention blocks to model the relationships of pixels on varying scales. However, the correlation is only considered in the local regions and fails in capturing spatial correlation on global features. **GSR-Net** [76] used a synthetic training dataset (unavailable) sampled from MS-COCO [42] and Casiav2 [18] and implemented an edge detection and refinement branch that accepts features from different levels. Region segmentation and edge detection are very different tasks and can affect the performance of each other if trained together.

2.3. Recent DNN-based models and training datasets

PSCCNet ([43]) used a synthetic training dataset (unable to download) sampled from MS-COCO [42], KCMI [1], Dresden [24], and Busternet database [70] and extracted hierarchical features along a top-down path and used a bottom-up path to detect manipulated regions in images. **MVSS-Net** [11] used Defacto [46] and Casiav2 [18] datasets for both training and evaluation and tried to address this problem by using an edge-supervised branch. They used noise distribution to create generalizable features and boundary artifacts surrounding tampered regions. They tried to address an important challenge of the IMDL task of balance between sensitivity and specificity, but fails to reach an equilibrium. **CAT-Net** [39] used a synthetic training dataset (unable to download) sampled from MS-COCO [42], Casiav2 [18], IMD2020 [49] and Fantastic Reality [37] and learns forensic features of compression artifacts on RGB and DCT domains and concatenates the features at a middle stage. They employ DCT histograms to detect double JPEG compression for image splicing only. **Transforensics** [28] used Casiav2 [18], Coverage [67] and IMD2020 [49] for both training and evaluation. They used a transformer-based model with dense self-attention encoders (FCN) and dense correction modules to capture global context and pairwise interaction between local patches through spatial self-attention. **RTAG** [8] used Fantastic Reality [37] dataset for training and extended the **MAGritte** [37] model to perform style transfer to create forged images, detect manipulated images, and localize manipulated pixels for splicing forgery using two GANs (faker for retouching and authenticator for localizing forgeries). They Adversarially train both generators to find whitening and coloring transforms. **Trace** [3] created a manipulated database with controlled image cues (noise models, image demosaicing, color correction, JPEG compression) and evaluated the image forensic tools. They generated user-defined endo-mask and exo-mask for each manipulation cue and the database has 5000 manipulated images, which is not enough to train a large DNN. This database is not very impactful when trying to detect real-life manipulated images. **Objectformer** ([66]) used a synthetic training dataset sampled from MS-COCO [42] and Paris-Street-View [51] and detected the manipulation artifacts by extracting and training multimodal patch embeddings with high-frequency features and RGB features combined and then used object prototypes to model object-level consistencies and find patch-level inconsistencies. They did not release their code or training dataset and hence the claimed results cannot be verified.

We found three major problems with the prior research work done in the IMDL task:

- The synthetic training datasets for most of the methods are unavailable. Only the pre-trained weights and inference code is released for a few of them. No detail is provided for creating a standardized training dataset.
- All the previous methods used varying datasets of different sizes and tasks to create a synthetic training dataset. Some methods used the same dataset for training and evaluation. This may induce bias in evaluation.
- All the previous methods used varying backbone deep neural networks for pretraining and finetuning. We propose to verify the performance of all the models, when pretrained on a single backbone network.

3. TrainFors: Benchmark IMDL training set

The main contribution of this paper is the development of a large-scale image manipulation dataset - *TrainFors*. The IMDL community uses benchmark evaluation datasets for manipulation detection and localization. But large-scale training dataset with real-world manipulations for the IMDL task is non-existent. We used open-source images to curate TrainFors. We need both manipulated and pristine images for the image manipulation detection task. The manipulated images can have four types of manipulations as discussed in section 1. Hence, we created five sets of images - pristine and the four manipulated types in the TrainFors dataset, see Tab. 1. Most of the previous works used one or more of these five image sets for training. TrainFors is an effort to accumulate all the manipulation types and pristine images under the same umbrella with the goal to diminish the difference between the curated training set tampered images and real-world manipulated images.

3.1. Image Collection

There exist a few image manipulation datasets that are released. But they are not sufficient and most of the models create their own synthetic training datasets to train the large DNNs. TrainFors is a large database that contains images from existing manipulated datasets and also some

Source Dataset	#Pristine	#Manipulated	#Splicing	#Copy-Move	#Removal	#Enhancement
Existing Training						
Trace [3]	1K	10K	5K	5K	0	0
Defacto[46]	0	149K	105K	19K	25K	0
Casiav2[18]	0	5123	1828	3295	0	0
Dresden[24]	0	35K	35K	0	0	0
Newly Generated Training						
MS-COCO [42]	172K	541K	25.2K	172.8K	171K	172K
Socrates [23]	3.2K	6.4K	3.2K	0	0	3.2K
Vision [59]	12.3K	24.6K	12.3K	0	0	12.3K
FODB [26]	7.6K	15.2K	7.6	0	0	7.6K
KCMI [1]	916	1832	916	0	0	916
Paris-Street-View [51]	3K	12K	4K	0	4K	4K
Total: TrainFors	200K	800K	200K	200K	200K	200K
Evaluation						
Columbia [48]	183	180	180	0	0	0
Coverage [67]	100	100	0	100	0	0
CASIAv1 [18]	800	920	461	459	0	0
NIST16 [2]	0	611	225	282	104	0
IMD2020 [49]	414	2010	1810	100	100	0

Table 1: Training and Evaluation Image Distribution - Number of images for each type of manipulation

new manipulated images were generated to create an exclusive superset of IMDL training database. We used manipulated images from Trace [3], Defacto [46], Casiav2 [18] and Dresden [24] datasets and created another set of pristine and manipulated images using MS-COCO [42], Socrates [23], Vision [59], FODB [26], KCMI [1], and Paris-Street-View [51] datasets. A detailed description of the source images used to generate the TrainFors database is presented in Tab. 1.

3.2. Dataset Description

Socrates [23], Vision [59], FODB [26], and KCMI [1] are Camera Identifying datasets and are not manipulated directly, neither do they possess ground-truth binary masks of manipulated pixels. To create meaningful tampered images from these images, we have externally added manipulated pixels and generated the ground-truth binary masks. They were used to generate splicing and image enhancement manipulations. The external manipulated pixels were added from MS-COCO [42] objects because MS-COCO provides a ground-truth mask of two million objects. Semantically meaningful objects were created from MS-COCO annotations and were used to generate splicing dataset. Paris-Street-View [51] dataset was directly used for creating removal manipulated images following the inpainting protocol mentioned in [51]. Copy-move images can only be generated if the ground-truth masks of objects are available and hence only MS-COCO was used to generate them.

3.3. Robust Forgery Pipelines with TrainFors

MS-COCO [42] has 12 supercategories and 80 categories of object-type images. We have used different combinations of these categories to generate manipulated pixels for splicing, copy-move, and removal images. We tried to make sure that the manipulated images look like real-world images and should be difficult to detect such manipulations by the naked eye. For each of the manipulation types, different pipelines are used to create the manipulated images.

Splicing: We generated spliced images using four types of combinations of MS-COCO categories. In the first combination, two images from the same supercategory were chosen. For example bird and cat categories under the same supercategory animal. In the second combination, objects from two different supercategories were chosen, for example, bird and stop sign from animal and outdoor supercategories respectively. In the third combination, we created spliced images from the same supercategory and category. For example, two different images of the person category were spliced. We generated 100 instances of each combination to generate the spliced images.

In image splicing, we have a pair of images - a donor image and a target image. To make the spliced images more realistic, we refined the segmented objects from the donor images using *MGMatting* [74], before inserting them into the target images. The manipulated objects were placed along the X and Y axes to make them look more realistic. We varied the size, and rotation of the donor pixels and randomly added JPEG compression (quality factor 50-99) before inserting them in target images in some cases. For the first and second types of combinations, we tried to choose the donor object to be small, flyable, and most commonly found to ensure a convincing representation. For example, a sports ball can be convincingly placed flying in any outdoor or indoor images.

In the fourth combination, we used the donor objects from MS-COCO and spliced them into target images from the camera identification datasets [23], [59], [26], [1], and [51] as presented in Tab. 1, using a similar setup as the previous combinations.

Copy-move: The copy-move manipulated image generation process is similar to image splicing, except that there is only a single image, and objects are duplicated in the same image. The camera identification datasets [23], [59], [26], [1], and [51] could not be used for generating copy-move forgeries. We used two kinds of combinations of MS-COCO categories to create the copy-move images. In the first combination, a single category image was chosen and the object in that image was duplicated and inserted into a different position in the same image. For example, a bird's pixels were duplicated and inserted at a different location in the same image. Similar to splicing, we refined the segmented objects using *MGMatting* [74], before inserting the duplicated pixels along the X and Y axes, depending on the height and width of the objects, after resizing and rotation.

In the second combination, we extracted images with multiple objects from MS-COCO and duplicated one of the objects to insert them in the source image, similar to the first combination. We repeated the process with the other object in the same image to create another variant of copy-move images. For example, an image may contain a backpack

and a handbag. We duplicated the backpack in the first set and then duplicated the handbag in the second set to create two instances of copy-move images.

Removal: In the removal manipulated images, one or multiple objects are removed from an image and substituted by background pixels by inpainting. We have used MS-COCO and Paris-Street-View datasets to generate the removal images with three combinations. In the first combination, we simply choose images from any category, and after removing the object, we inpaint it with the background pixels. We used an exemplar-based blending method [55] for inpainting. We tried to make sure that the chosen objects are not very cluttered or complex. For example, if we remove a hat from a person's head, it may look visually unrealistic if we cannot inpaint the hair and head properly. For the second combination, we selected MS-COCO images with multiple objects and repeated the removal procedure of the first combination with two or more objects, creating multiple instances from a single source image. In the third combination, we created removal images from [51] by removing regions from the images and inpainting them using [55]

Image Enhancement: We have performed a series of image enhancements using major image processing methods - adding noise, image morphing, image compression, image resizing, and image blurring. We added Gaussian Noise, Uniform Noise, Poisson Noise, and Impulse Noise. We performed Open Morphing, Erode Morphing, Dilate Morphing and Closed Morphing. We did Area Resize, Cubic Resize, Lanczo Resize, Linear Resize, and Nearest Resize. We created images with JPEG Compression, JPEG Double Compression, and WEBP Compression. We added Box Blur, Gaussian Blur, Median Blur, and Wavelet Blur. Finally, we created images with Quantization, Dithering, Posterization, Histogram Equalization, and Auto Contrast. We used opencv functions to perform the image enhancements and applied *MGMatting* [74] blending when required in certain cases to make the enhanced images look natural.

Pristine Images: Image manipulation detection training requires negative samples, represented by unmanipulated pristine images. We have used a sample set of original images from [[42], [23], [59], [26], [1], and [51]] datasets to create the non-manipulated pristine image set of 200K.

In total, TrainFors have 800K manipulated images (positive samples) and 200K pristine images (negative samples) as depicted in Tab. 1.

4. Experimental Evaluation

We trained four state-of-the-art IMDL models on our proposed TrainFors dataset and evaluated the IMDL performance on five benchmark evaluation datasets. The evaluation mainly comprises two tasks - detecting manipulated images and in the latter task, we generate a manipulated pixel map of the positively detected manipulated images.

4.1. Benchmark Evaluation Datasets

We have evaluated the manipulation localization task on five benchmark datasets: **Columbia**[48], **Coverage**[67], **CASIAv1**[18], **NIST16**[2] and **IMD20**[49]. Columbia[48] is an image-splicing dataset, consisting of 180 images. Coverage[67] is an image copy-move detection dataset composed of 100 images. CASIA[18] has both copy-move and splicing images: 5123 images in v2.0 and 921 images in v1.0. NIST16[2] dataset includes splicing, copy-move, and image enhancement/reduction manipulations with 611 images. IMD20[49] includes 2,010 manipulated images scraped from the internet. Refer to Tab. 1 for the detailed summary of the number of images of each type of manipulation for all the evaluation datasets.

4.2. Evaluation Metrics

We have evaluated the model performance for both image manipulation localization and detection tasks. For manipulation localization, pixel-level Area Under Curve (AUC) and F1 scores are reported. While for manipulation detection, image-level AUC and F1 scores are reported.

4.3. Baseline Models

Most of the previous works followed a two-step training method in generating the IMDL models. Firstly they pretrained a backbone network with a synthetic training dataset and then fine-tuned the pretrained model with the train-split of the evaluation datasets. **PSCCNet[43]** [2] used HRNetV2p-W18 [65] as a backbone network for pretraining their own synthetic data. **ObjectFormer[66]** used EfficientNetb4 [60] as the backbone network. **MVSS-Net**[11] [3] used ResNet-50 [29] as the backbone network. **CAT-Net** [39] [4] also used HRNet [65] as the backbone network. All the state-of-the-art models used varying input image sizes for pretraining the network. We ran two sets of experiments - firstly pretraining all the models on TrainFors using the backbone network specified by each of them respectively and then fine-tuned using the models from the inference code, if released by the authors, else we wrote the model code referencing their respective papers. In the second set of experiments, we fixed the backbone network as EfficientNetV2 [61] (pretrained on ImageNet [16] weights) for all the models and fine-tuned it in a process similar to the first set of experiments. We were not able to report the performance of the early DNN-based models from Sec. 2.2 and some models from Sec. 2.3 because either the code is not released or is no longer publicly accessible.

4.4. Implementation Details

All the images in TrainFors are resized to 256X256 before feeding them to the backbone network. We imple-

[2] https://github.com/proteus1991/PSCC-Net
[3] https://github.com/dong03/MVSS-Net
[4] https://github.com/mjkwon2021/CAT-Net

Method	Columbia[48]		Coverage[67]		CASIAv1[18]		NIST16[2]		IMD20[49]	
	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1
Author-Specified Backbone										
MVSS-Net[11]	61.1±1.2	52.9±1.1	51.9±1.3	36.8±1.3	54.6±1.7	35.1±1.8	38.7±1.8	19.3±2.1	47.9±1.7	30.2±1.6
Cat-Net[39]	56.3±2.1	44.7±2.3	23.4±1.9	7.6±0.7	28.7±1.6	8.9±0.6	29.6±1.4	11.4±1.1	19.3±1.6	6.4±0.3
PSCCNet[43]	58.7±1.3	49.4±1.2	63.9±2.6	46.8 ±2.3	60.9 ±1.1	42.9±0.9	48.6 ±0.8	29.3 ±1.3	41.3±1.7	28.5±2.2
ObjectFormer[66]	55.8±1.1	46.3±1.1	64.2±1.8	47.2±1.7	61.3±1.6	43.2±1.1	48.7±1.2	29.4±1.1	42.5±1.2	28.7±1.3
EfficientNetV2 [61] Backbone										
MVSS-Net[11]	68.3±1.2	58.4±1.2	56.7±1.3	42.6±1.3	59.3±1.8	41.5±1.7	42.1±1.9	24.4±2.1	**53.2±1.6**	**34.1±1.6**
Cat-Net[39]	**70.4±1.4**	**64.2±1.9**	25.6±1.1	9.2±0.9	33.5±1.8	10.6±0.8	37.2±1.5	14.8±1.2	24.8±1.8	9.2±0.5
PSCCNet[43]	62.7±1.3	53.4±1.3	**67.1±1.9**	**50.8±2.1**	**63.8±1.2**	**46.7±1.1**	**52.6±1.2**	**35.7±1.1**	44.5±1.6	32.6±2.3
ObjectFormer[66]	55.8±1.1	46.3±1.1	64.2±1.8	47.3±1.7	61.3±1.6	43.2±1.1	48.7±1.2	29.4±1.1	42.5±1.2	28.7±1.3

Table 2: Manipulation Localization AUC (%) and F1 (%) scores of **Pre-trained models**, when trained with author-specified backbone networks and EfficientNetV2 [61] backbone network respectively: Upper and Lower limits over 6 runs

Method	Columbia[48]		Coverage[67]		CASIAv1[18]		NIST16[2]		IMD20[49]	
	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1
Author-Specified Backbone										
MVSS-Net[11]	71.2±1.3	63.8±1.3	62.9±1.2	45.3±1.1	63.3±1.8	45.2±1.8	47.8±1.9	29.2±1.7	56.6±1.7	39.8±1.7
Cat-Net[39]	67.8±2.3	55.5±2.1	32.3±1.2	12.9±0.6	33.7±1.4	13.6±1.1	41.3±1.8	17.9±1.5	28.4±1.6	10.1±0.9
PSCCNet[43]	69.5±1.2	60.6±1.3	72.3±2.1	57.4±2.1	71.6±1.6	51.2±1.1	59.3±1.2	38.7±2.6	52.7±1.6	38.4±1.7
ObjectFormer[66]	66.3±1.2	56.7±1.1	73.1±1.2	56.4±1.3	72.7±1.7	52.6±1.7	59.1±1.2	37.6±1.2	51.9±1.4	38.3±1.1
EfficientNetV2 [61] Backbone										
MVSS-Net[11]	78.9±1.2	62.7±1.2	69.7±1.2	53.2±1.1	70.4±1.6	52.9±1.7	59.2±1.7	36.7±2.3	**62.3±1.8**	**46.4±1.7**
Cat-Net[39]	**79.5±1.7**	**65.3±1.1**	37.6±2.1	14.6±1.3	38.6±1.6	15.2±1.2	50.1±1.7	22.3±1.5	34.6±1.5	12.8±1.1
PSCCNet[43]	76.9±1.3	62.2±1.2	**78.8±2.2**	**58.6±2.1**	**76.7±1.3**	**54.1±1.1**	**62.3±1.4**	**41.7±2.1**	57.4±1.5	40.3±1.5
ObjectFormer[66]	66.3±1.2	56.7±1.1	73.1±1.2	56.4±1.3	72.7±1.7	52.6±1.7	59.1±1.2	37.6±1.2	51.9±1.4	38.3±1.1

Table 3: Manipulation Localization AUC(%) and F1(%) scores of **Fine-tuned models**, when trained with author-specified backbone networks and EfficientNetV2 [61] backbone network respectively: Upper and Lower limits over 6 runs

Method	Columbia[48]		Coverage[67]		CASIAv1[18]		IMD20[49]	
	AUC	F1	AUC	F1	AUC	F1	AUC	F1
Author-Specified Backbone								
MVSS-Net[11]	82.1	62.3	42.6	24.3	78.6	59.8	65.7	35.1
Cat-Net[39]	81.6	61.7	37.4	29.8	64.7	38.0	59.1	35.4
PSCCNet[43]	83.4	63.8	81.5	62.8	84.6	76.9	78.6	57.3
ObjectFormer[66]	84.8	64.5	82.7	63.9	86.1	77.4	79.3	58.2
EfficientNetV2 [61] Backbone								
MVSS-Net[11]	**85.6**	**65.9**	57.6	41.5	83.6	64.3	**69.8**	**40.2**
Cat-Net[39]	83.5	63.9	38.6	29.9	68.9	43.2	64.3	38.6
PSCCNet[43]	85.4	65.6	**83.6**	**64.7**	**87.2**	**79.6**	67.4	39.6
ObjectFormer[66]	84.8	64.5	82.7	63.9	86.1	77.4	79.3	58.2

Table 4: Manipulation Detection AUC(%) and F1(%) scores on CASIA-D dataset[18], when trained with author-specified backbone networks and EfficientNetV2 [61] backbone network respectively

mented all the models in PyTorch and trained them on NVIDIA GeForce RTX 2080 Ti GPU. We used Adam [35] optimizer with a batch size of 24 and a learning rate periodically varying between 1×10^{-3} to 1×10^{-6}. We trained all the models for 100 epochs. As pointed out by [43], it is inefficient to pre-train the models with the entire 1 Million images in TrainFors. We sampled 0.1 Million images randomly for training in each epoch.

4.5. Image Manipulation Localization

We used the evaluation protocol defined in [31] to evaluate the localization performance using two modules: Firstly, the pretrained model is trained on the TrainFors dataset and evaluated on the entire test set. Secondly, the pre-trained model is fine-tuned on the train-split of the evaluation datasets and evaluated on the test-splits of the datasets. We reported the upper and lower limits of the metrics on 6 runs to evaluate the performance variance.

Pre-trained model: Tab. 2 reports the localization performance of the pre-trained models from Sec. 4.3 on the five benchmark evaluation datasets from Sec. 4.1, reporting the pixel-level AUC and F1 scores. The Cat-Net performance is best on the Columbia dataset and showed very poor performance on the Coverage dataset. This could be attributed to the fact that the Cat-Net model is designed for a Splicing dataset and Columbia and Coverage datasets have all-splicing and no-splicing images respectively (see Tab. 1). The pre-trained PSCCNet model achieves the best localization performance on Coverage, CASIAv1, and NIST16 datasets, when all the models were pre-trained with the same backbone network. The pre-trained MVSS-Net model achieves the best performance on the IMD20 dataset because it has real-world images and PSCCNet was pretrained on synthetic datasets. The PSCCNet has the best generalization ability compared to the other models as it is pre-trained on a large amount of synthetic training data.

Fine-tuned model: According to the second protocol, we

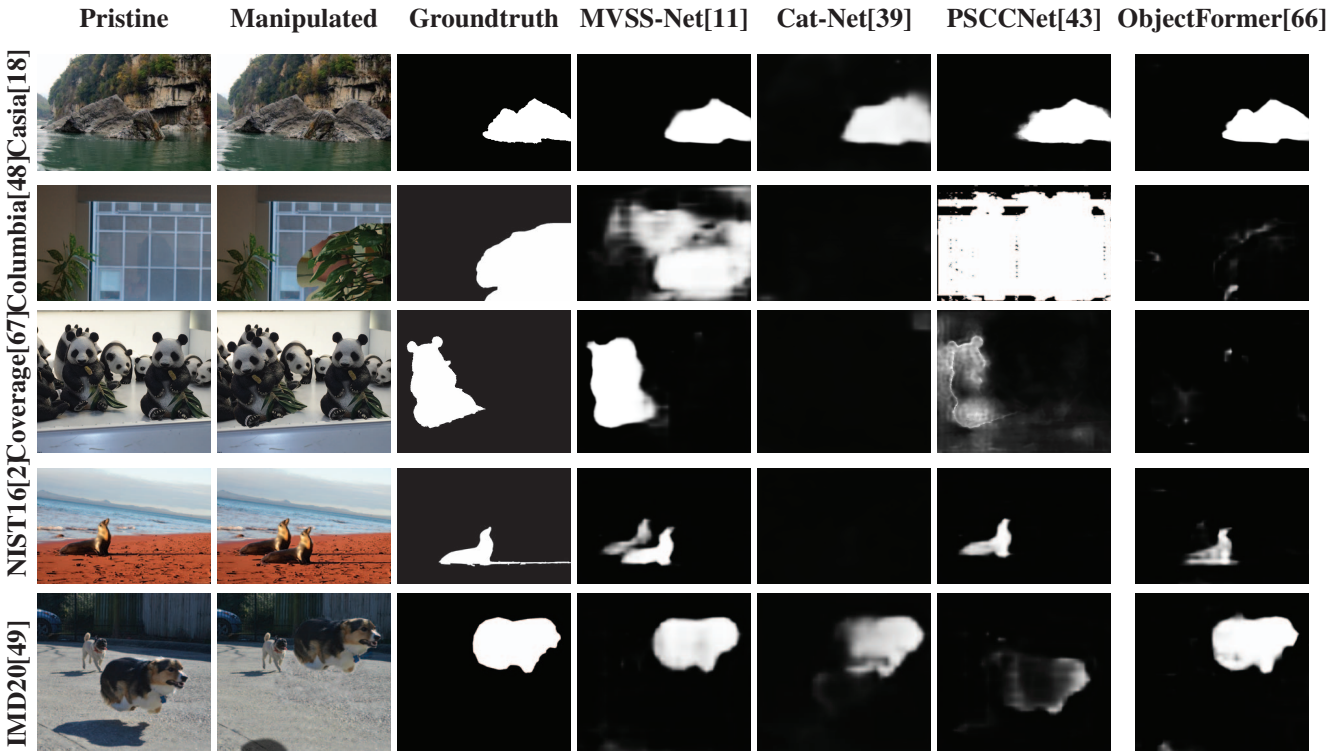

Figure 3: Image Manipulation Localization Prediction Visualization - From top to bottom, we show examples from 5 benchmark evaluation datasets: Casia, Columbia, Coverage Nist16, and IMD20 by 4 baseline IMDL models: MVSS-Net, Cat-Net, PSCC-Net, and ObjectFormer, trained on our proposed TrainFors dataset.

further fine-tuned the pre-trained model on the train-split of the evaluation datasets. The training strategy of fine-tuned models is similar to that of the pre-trained models. We compare the pixel-level AUC and F1 scores of fine-tuned models on Tab. 3. All the fine-tuned models behaved similarly to the results portrayed by the pre-trained models. The fine-tuned PSCCNet model achieves the best localization performance on Coverage, CASIAv1, and NIST16 datasets. The fine-tuned Cat-Net and MVSS-Net models achieve the best localization performance on Columbia and IMD20 datasets respectively. It should be noted that the fine-tuned model performance improves when we train them with the EfficientNetV2 backbone network.

It is clearly evident from Tab. 2 and Tab. 3 that the performance of pre-trained and fine-tuned models vary if the backbone network is altered. The Objectformer performance was on par with the other baseline models for using a superior backbone pre-training model, but it was outperformed, when the backbone network was fixed. The evaluated results reported in the baseline papers deteriorate when pre-trained with a more generalized dataset, showcasing the importance of a generalized TrainFors dataset.

4.6. Image Manipulation Detection

Similar to [43] and [66], we used the pre-trained models for the image manipulation detection task and reported the image-level AUC and F1 scores in Tab. 4 on four benchmark evaluation datasets. We cannot evaluate the manipulation detection results on the NIST16 dataset as it does not have any pristine (negative) images (see Tab. 1). All the previous research work reported the detection results (AUC score only) on the Casia dataset only, but for a fair comparison, we reported the results (both AUC and F1 scores) on all four datasets. With the author-specified back-

	No Distortion	Resize (0.78X)	Resize (0.25X)	Gau-Blur (k=3)	Gau-Blur (k=15)	Gau-N (σ=3)	Gau-N (σ=15)	JPG-Comp (q=100)	JPG-Comp (q=50)	Mixed
Columbia										
Author-Specified Backbone										
MVSS-Net[11]	61.1	60.9	60.9	59.8	58.7	57.8	56.4	59.3	59.5	56.3
Cat-Net[39]	56.3	56.2	56.2	55.4	54.8	53.2	52.7	55.6	55.7	52.6
PSCCNet[43]	58.7	58.6	58.6	57.9	57.2	56.5	55.6	57.8	57.9	56.4
ObjectFormer[66]	55.8	55.8	55.7	53.2	52.8	48.4	47.8	54.6	54.7	47.5
EfficientNetV2 [61] Backbone										
MVSS-Net[11]	68.3	67.9	67.8	66.7	65.6	64.7	63.5	66.8	66.7	63.2
Cat-Net[39]	**70.4**	**70.3**	**70.2**	**68.5**	**67.7**	**67.2**	**66.6**	**69.2**	**69.4**	**66.2**
PSCCNet[43]	62.7	62.5	62.4	60.7	59.8	57.7	57.1	60.3	60.4	56.6
ObjectFormer[66]	55.8	55.8	55.7	53.2	52.8	48.4	47.8	54.6	54.7	47.5
NIST16										
Author-Specified Backbone										
MVSS-Net[11]	38.7	38.7	38.7	38.6	36.4	35.6	34.7	37.6	37.7	34.2
Cat-Net[39]	29.6	29.6	29.6	29.5	28.6	27.9	27.2	29.1	29.2	26.8
PSCCNet[43]	48.6	48.6	48.6	48.5	47.1	46.2	45.3	47.6	47.5	45.1
ObjectFormer[66]	48.7	48.7	48.6	48.5	46.4	45.7	44.9	48.6	48.6	44.7
EfficientNetV2 [61] Backbone										
MVSS-Net[11]	42.1	42.1	42.0	41.9	40.2	39.8	39.3	41.9	41.9	39.1
Cat-Net[39]	37.2	37.2	37.1	36.9	35.1	34.7	33.8	37.1	37.2	33.6
PSCCNet[43]	**52.6**	**52.5**	**52.4**	**51.7**	**50.9**	**50.1**	**49.6**	**51.8**	**51.8**	**49.4**
ObjectFormer[66]	48.7	48.7	48.6	48.5	46.4	45.7	44.9	48.6	48.6	44.7

Table 5: Robustness Comparison of Pixel-level Manipulation Localization AUC(%) with various distortions evaluated on Columbia[48] and NIST16[2] datasets, when pretrained with author-specified backbone networks and EfficientNetV2 [61] backbone network respectively

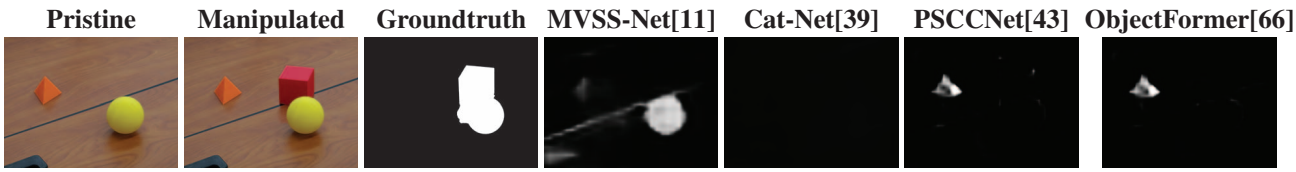

Figure 4: Image Manipulation False Alarm Predictions by 4 baseline models: MVSS-Net, Cat-Net, PSCC-Net, and ObjectFormer, trained on our proposed TrainFors dataset.

bone network pre-training, Objectformer achieved the best performance, but with the EfficientNetV2 backbone network, all the other baseline models outperformed Objectformer. PSCCNet gave the best manipulation detection performance on the Coverage and Casia datasets and MVSS-Net on Columbia and IMD20 datasets. For a fair evaluation of the baseline models, we reported the image manipulation localization and detection tasks separately. But for an efficient IMDL task, manipulation detection should be performed before manipulation localization and only the detected images should be checked for the manipulated pixels.

4.7. Qualitative Analysis

We reported the manipulation predictions of the baseline models, after training them with TrainFors dataset in Fig. 3. The models can easily predict manipulated pixels if large objects are tampered with in an image (eg, Casia and IMD20). For copymove examples, if multiple similar objects are present in an image (eg, more than one panda in Coverage), model prediction is not very accurate. MVSS-Net showed promising predictions in this case. However, if a single object is copymoved, almost all the baseline models can predict the tampered pixels, except Cat-Net (eg, sealion copymoved in Nist16). The reason for Cat-Net's poor performance may be attributed to the fact that it was designed for image splicing and may fail for other types of manipulations. In removal images, a good blending method can result in model failure. It can be concluded that the baseline IMDL models' performance changes when pretrained with the same training dataset and the same backbone pre-training network. Further discussion in the supplementary material.

4.8. Robustness Evaluation

All the previous methods [72], [31], [43], [66] used different image distortion methods on raw images from the NIST16[2] and/or Columbia[48] datasets, and evaluated the robustness of the models. Similarly, we added the following distortions to the manipulated images: **Image scaling** with scales=0.78X, 0.25X, **Gaussian Blurring** with a kernel size k=3, 15, added **Gaussian Noise** with a standard deviation σ= 3, 15, and **JPEG Compression** with a quality factor q=50, 100. We also added a mix of these distortions in the mixed column and the No Distortion column. We compared the manipulation localization performance (AUC scores) of the pre-trained models with all the baseline methods on these distorted images and report the results on Columbia and NIST16 datasets in Tab. 5. Cat-Net and PSCCNet demonstrated the best robustness against various image distortions, on the Columbia and NIST16 datasets respectively, when the EfficientNetV2 backbone network was used for pre-training. Image manipulation detection robustness analysis is discussed in the supplementary material.

4.9. Runtime Analysis

We measured the runtime in terms of frames per second (FPS) on the inference models and evaluated them on NVIDIA GeForce RTX 2080 Ti GPU. Cat-Net is quite inefficient with 4.2 FPS, when compared to 18.6 FPS in MVSS-Net and 51.3 FPS in PSCCNet. Objectformer is much slower than the other counterparts at 1.9 FPS.

4.10. Limitations

The biggest challenge for the IMDL researchers is failed predictions and false alarms, see Fig. 4. The presence of multiple objects can sometimes misguide the IMDL models to easily predict a pristine object as manipulated. Other cases where an image is enhanced, without content modification, may also lead to manipulation detection failures.

5. Conclusion

We introduced TrainFors, a large-scale training dataset for the image manipulation detection and localization task. We hope that TrainFors will be used as a benchmark training dataset for the IMDL task, similar to the benchmark evaluation datasets. We performed extensive experiments with the state-of-the-art baseline IMDL models and showcased fair comparisons under a similar training setup that was not done by the IMDL community previously. The experimental results authenticate that the IMDL model performance is dependent on the training data and the backbone networks used for pre-training and fine-tuning. Along with standardized training and evaluation datasets, standardized evaluation metrics (both pixel-level and image-level AUC and F1 scores) were used to fairly compare the IMDL models. We will release the dataset, which will be available to the research community for future research on detecting and localizing manipulated images.

6. Acknowledgement

This work is based on research sponsored by the Defense Advanced Research Projects Agency under agreement number HR0011-21-C- 0164. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

References

[1] Ieee's signal processing society - camera model identification, 2018. https://www.kaggle.com/competitions/sp-society-camera-model-identification.

[2] Nimble challenge 2017 evaluation — nist. https://www.nist.gov/itl/iad/mig/nimble-challenge-2017-evaluation. (Accessed on 11/14/2020).

[3] Quentin Bammey, Tina Nikoukhah, Marina Gardella, Rafael Grompone von Gioi, Miguel Colom, and Jean-Michel Morel. Non-semantic evaluation of image forensics tools: Methodology and database. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3751–3760, 2022.

[4] Jawadul H Bappy, Amit K Roy-Chowdhury, Jason Bunk, Lakshmanan Nataraj, and BS Manjunath. Exploiting spatial structure for localizing manipulated image regions. In *Proceedings of the IEEE international conference on computer vision*, pages 4970–4979, 2017.

[5] Jawadul H Bappy, Cody Simons, Lakshmanan Nataraj, BS Manjunath, and Amit K Roy-Chowdhury. Hybrid lstm and encoder–decoder architecture for detection of image forgeries. *IEEE Transactions on Image Processing*, 28(7):3286–3300, 2019.

[6] Belhassen Bayar and Matthew C Stamm. A deep learning approach to universal image manipulation detection using a new convolutional layer. In *Proceedings of the 4th ACM workshop on information hiding and multimedia security*, pages 5–10, 2016.

[7] Belhassen Bayar and Matthew C Stamm. Constrained convolutional neural networks: A new approach towards general purpose image manipulation detection. *IEEE Transactions on Information Forensics and Security*, 13(11):2691–2706, 2018.

[8] Xiuli Bi, Zhipeng Zhang, and Bin Xiao. Reality transform adversarial generators for image splicing forgery detection and localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14294–14303, 2021.

[9] Luca Bondi, Luca Baroffio, David Güera, Paolo Bestagini, Edward J Delp, and Stefano Tubaro. First steps toward camera model identification with convolutional neural networks. *IEEE Signal Processing Letters*, 24(3):259–263, 2016.

[10] Luca Bondi, Silvia Lameri, David Guera, Paolo Bestagini, Edward J Delp, Stefano Tubaro, et al. Tampering detection and localization through clustering of camera-based cnn features. In *CVPR Workshops*, volume 2, 2017.

[11] Xinru Chen, Chengbo Dong, Jiaqi Ji, Juan Cao, and Xirong Li. Image manipulation detection by multi-view multi-scale supervision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14185–14193, 2021.

[12] Hak-Yeol Choi, Han-Ul Jang, Dongkyu Kim, Jeongho Son, Seung-Min Mun, Sunghee Choi, and Heung-Kyu Lee. Detecting composite image manipulation based on deep neural networks. In *2017 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 1–5. IEEE, 2017.

[13] Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. Efficient dense-field copy–move forgery detection. *IEEE Transactions on Information Forensics and Security*, 10(11):2284–2297, 2015.

[14] Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. Splicebuster: A new blind image splicing detector. In *2015 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2015.

[15] Davide Cozzolino and Luisa Verdoliva. Noiseprint: a cnn-based camera model fingerprint. *IEEE Transactions on Information Forensics and Security*, 15:144–159, 2019.

[16] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[17] Helisa Dhamo, Azade Farshad, Iro Laina, Nassir Navab, Gregory D Hager, Federico Tombari, and Christian Rupprecht. Semantic image manipulation using scene graphs. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5213–5222, 2020.

[18] Jing Dong, Wei Wang, and Tieniu Tan. Casia image tampering detection evaluation database. In *2013 IEEE China Summit and International Conference on Signal and Information Processing*, pages 422–426. IEEE, 2013.

[19] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[20] Yu Fan, Philippe Carré, and Christine Fernandez-Maloigne. Image splicing detection with local illumination estimation. In *2015 IEEE international conference on Image processing (ICIP)*, pages 2940–2944. IEEE, 2015.

[21] Pasquale Ferrara, Tiziano Bianchi, Alessia De Rosa, and Alessandro Piva. Image forgery localization via fine-grained analysis of cfa artifacts. *IEEE Transactions on Information Forensics and Security*, 7(5):1566–1577, 2012.

[22] Jessica Fridrich and Jan Kodovsky. Rich models for steganalysis of digital images. *IEEE Transactions on information Forensics and Security*, 7(3):868–882, 2012.

[23] Chiara Galdi, Frank Hartung, and Jean-Luc Dugelay. Socrates: A database of realistic data for source camera

recognition on smartphones. In *ICPRAM*, pages 648–655, 2019.

[24] Thomas Gloe and Rainer Böhme. The'dresden image database'for benchmarking digital image forensics. In *Proceedings of the 2010 ACM symposium on applied computing*, pages 1584–1590, 2010.

[25] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[26] Benjamin Hadwiger and Christian Riess. The forchheim image database for camera identification in the wild. In *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part VI*, pages 500–515. Springer, 2021.

[27] Jong Goo Han, Tae Hee Park, Yong Ho Moon, and Il Kyu Eom. Efficient markov feature extraction method for image splicing detection using maximization and threshold expansion. *Journal of Electronic Imaging*, 25(2):023031, 2016.

[28] Jing Hao, Zhixin Zhang, Shicai Yang, Di Xie, and Shiliang Pu. Transforensics: image forgery localization with dense self-attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15055–15064, 2021.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[30] János Horváth, Sriram Baireddy, Hanxiang Hao, Daniel Mas Montserrat, and Edward J Delp. Manipulation detection in satellite images using vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1032–1041, 2021.

[31] Xuefeng Hu, Zhihan Zhang, Zhenye Jiang, Syomantak Chaudhuri, Zhenheng Yang, and Ram Nevatia. Span: spatial pyramid attention network for image manipulation localization. In *European Conference on Computer Vision*, pages 312–328. Springer, 2020.

[32] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[33] Minyoung Huh, Andrew Liu, Andrew Owens, and Alexei A Efros. Fighting fake news: Image splice detection via learned self-consistency. In *Proceedings of the European conference on computer vision (ECCV)*, pages 101–117, 2018.

[34] Ashraful Islam, Chengjiang Long, Arslan Basharat, and Anthony Hoogs. Doa-gan: Dual-order attentive generative adversarial network for image copy-move forgery detection and localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4676–4685, 2020.

[35] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[36] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[37] Vladimir V Kniaz, Vladimir Knyaz, and Fabio Remondino. The point where reality meets fantasy: Mixed adversarial generators for image splice detection. *Advances in Neural Information Processing Systems*, 32, 2019.

[38] Neal Krawetz and Hacker Factor Solutions. A picture's worth. *Hacker Factor Solutions*, 6(2):2, 2007.

[39] Myung-Joon Kwon, In-Jae Yu, Seung-Hun Nam, and Heung-Kyu Lee. Cat-net: Compression artifact tracing network for detection and localization of image splicing. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 375–384, 2021.

[40] Bowen Li, Xiaojuan Qi, Thomas Lukasiewicz, and Philip HS Torr. Manigan: Text-guided image manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7880–7889, 2020.

[41] Ce Li, Qiang Ma, Limei Xiao, Ming Li, and Aihua Zhang. Image splicing detection based on markov features in qdct domain. *Neurocomputing*, 228:29–36, 2017.

[42] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[43] Xiaohong Liu, Yaojie Liu, Jun Chen, and Xiaoming Liu. Pscc-net: Progressive spatio-channel correlation network for image manipulation detection and localization. *arXiv preprint arXiv:2103.10596*, 2021.

[44] Siwei Lyu, Xunyu Pan, and Xing Zhang. Exposing region splicing forgeries with blind local noise estimation. *International journal of computer vision*, 110(2):202–221, 2014.

[45] Babak Mahdian and Stanislav Saic. Using noise inconsistencies for blind image forensics. *Image and Vision Computing*, 27(10):1497–1503, 2009.

[46] Gaël Mahfoudi, Badr Tajini, Florent Retraint, Frederic Morain-Nicolier, Jean Luc Dugelay, and PIC Marc. Defacto: image and face manipulation dataset. In *2019 27Th european signal processing conference (EUSIPCO)*, pages 1–5. IEEE, 2019.

[47] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[48] Tian-Tsong Ng, Jessie Hsu, and Shih-Fu Chang. Columbia image splicing detection evaluation dataset. *DVMM lab. Columbia Univ CalPhotos Digit Libr*, 2009.

[49] Adam Novozamsky, Babak Mahdian, and Stanislav Saic. Imd2020: A large-scale annotated dataset tailored for detecting manipulated images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision Workshops*, pages 71–80, 2020.

[50] Taesung Park, Jun-Yan Zhu, Oliver Wang, Jingwan Lu, Eli Shechtman, Alexei Efros, and Richard Zhang. Swapping autoencoder for deep image manipulation. *Advances in Neural Information Processing Systems*, 33:7198–7211, 2020.

[51] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[52] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. Variational autoencoder for deep learning of images, labels and captions. *Advances in neural information processing systems*, 29, 2016.

[53] Yuan Rao and Jiangqun Ni. A deep learning approach to detection of splicing and copy-move forgeries in images. In *2016 IEEE international workshop on information forensics and security (WIFS)*, pages 1–6. IEEE, 2016.

[54] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[55] Viktor Reshniak, Jeremy Trageser, and Clayton G Webster. A nonlocal feature-driven exemplar-based approach for image inpainting. *SIAM Journal on Imaging Sciences*, 13(4):2140–2168, 2020.

[56] Ekraam Sabir, Jiaxin Cheng, Ayush Jaiswal, Wael AbdAlmageed, Iacopo Masi, and Prem Natarajan. Recurrent convolutional strategies for face manipulation detection in videos. *Interfaces (GUI)*, 3(1):80–87, 2019.

[57] Ekraam Sabir, Soumyaroop Nandi, Wael Abd-Almageed, and Prem Natarajan. Biofors: A large biomedical image forensics dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10963–10973, 2021.

[58] Ronald Salloum, Yuzhuo Ren, and C-C Jay Kuo. Image splicing localization using a multi-task fully convolutional network (mfcn). *Journal of Visual Communication and Image Representation*, 51:201–209, 2018.

[59] Dasara Shullani, Marco Fontani, Massimo Iuliani, Omar Al Shaya, and Alessandro Piva. Vision: a video and image dataset for source identification. *EURASIP Journal on Information Security*, 2017(1):1–16, 2017.

[60] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[61] Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *International conference on machine learning*, pages 10096–10106. PMLR, 2021.

[62] Tiberio Uricchio, Lamberto Ballan, Irene Roberto Caldelli, et al. Localization of jpeg double compression through multi-domain convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 53–59, 2017.

[63] Yael Vinker, Eliahu Horwitz, Nir Zabari, and Yedid Hoshen. Deep single image manipulation. *arXiv preprint arXiv:2007.01289*, 2020.

[64] Jane Wakefield. Deepfake presidents used in russia-ukraine war. *BBC News*.

[65] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3349–3364, 2020.

[66] Junke Wang, Zuxuan Wu, Jingjing Chen, Xintong Han, Abhinav Shrivastava, Ser-Nam Lim, and Yu-Gang Jiang. Objectformer for image manipulation detection and localization. *arXiv preprint arXiv:2203.14681*, 2022.

[67] Bihan Wen, Ye Zhu, Ramanathan Subramanian, Tian-Tsong Ng, Xuanjing Shen, and Stefan Winkler. Coverage—a novel database for copy-move forgery detection. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 161–165. IEEE, 2016.

[68] Haiwei Wu, Jiantao Zhou, Jinyu Tian, and Jun Liu. Robust image forgery detection over online social network shared images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13440–13449, 2022.

[69] Yue Wu, Wael Abd-Almageed, and Prem 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*, pages 1480–1502, 2017.

[70] Yue Wu, Wael Abd-Almageed, and Prem Natarajan. Busternet: Detecting copy-move image forgery with source/target localization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 168–184, 2018.

[71] Yue Wu, Wael Abd-Almageed, and Prem Natarajan. Image copy-move forgery detection via an end-to-end deep neural network. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1907–1915. IEEE, 2018.

[72] Yue Wu, Wael AbdAlmageed, and Premkumar Natarajan. Mantra-net: Manipulation tracing network for detection and localization of image forgeries with anomalous features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9543–9552, 2019.

[73] Qiuwei Yang, Fei Peng, Jiao-Ting Li, and Min Long. Image tamper detection based on noise estimation and lacunarity texture. *Multimedia Tools and Applications*, 75(17):10201–10211, 2016.

[74] Qihang Yu, Jianming Zhang, He Zhang, Yilin Wang, Zhe Lin, Ning Xu, Yutong Bai, and Alan Yuille. Mask guided matting via progressive refinement network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1154–1163, 2021.

[75] Zhongping Zhang, Yixuan Zhang, Zheng Zhou, and Jiebo Luo. Boundary-based image forgery detection by fast shallow cnn. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 2658–2663. IEEE, 2018.

[76] Peng Zhou, Bor-Chun Chen, Xintong Han, Mahyar Najibi, Abhinav Shrivastava, Ser-Nam Lim, and Larry Davis. Generate, segment, and refine: Towards generic manipulation segmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13058–13065, 2020.

[77] Peng Zhou, Xintong Han, Vlad I Morariu, and Larry S Davis. Learning rich features for image manipulation detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1053–1061, 2018.

[78] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE*

international conference on computer vision, pages 2223–2232, 2017.

[79] Xinshan Zhu, Yongjun Qian, Xianfeng Zhao, Biao Sun, and Ya Sun. A deep learning approach to patch-based image inpainting forensics. *Signal Processing: Image Communication*, 67:90–99, 2018.

[80] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.