# Uncertainty-guided Learning for Improving Image Manipulation Detection

Kaixiang Ji[1] Feng Chen[1] Xin Guo[1] Yadong Xu[2] Jian Wang[1*] Jingdong Chen[1]
[1] Ant Group [2] Tsinghua University
{kaixiang.jkx, bangzhu.gx, bobblair.wj, jingdongchen.cjd}@antgroup.com
chenfeng1271@gmail.com xuyd17@mails.tsinghua.edu.cn

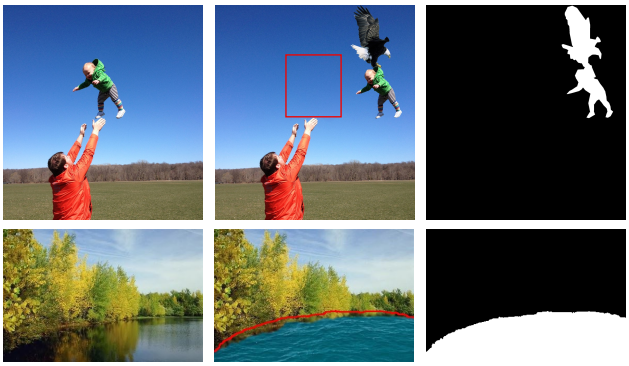
## Abstract

*Image manipulation detection (IMD) is of vital importance as faking images and spreading misinformation can be malicious and harm our daily life. IMD is the core technique to solve these issues and poses challenges in two main aspects: (1) Data Uncertainty, i.e., the manipulated artifacts are often hard for humans to discern and lead to noisy labels, which may disturb model training; (2) Model Uncertainty, i.e., the same object may hold different categories (tampered or not) due to manipulation operations, which could potentially confuse the model training and result in unreliable outcomes. Previous works mainly focus on solving the model uncertainty issue by designing meticulous features and networks, however, the data uncertainty problem is rarely considered. In this paper, we address both problems by introducing an uncertainty-guided learning framework, which measures data and model uncertainties by a novel Uncertainty Estimation Network (UEN). UEN is trained under dynamic supervision, and outputs estimated uncertainty maps to refine manipulation detection results, which significantly alleviates the learning difficulties. To our knowledge, this is the first work to embed uncertainty modeling into IMD. Extensive experiments on various datasets demonstrate state-of-the-art performance, validating the effectiveness and generalizability of our method.*

## 1. Introduction

Seeing is no longer believing! With the development of modern image editing tools, such as Photoshop, Snapseed *etc*., editing and sharing images have become easier and more popular. However, the abuse of manipulated images can also cause various problems that affect our daily life negatively. In addition, manipulated images are often realistic-looking, making them difficult and laborious for humans to discern. Thus, it is critical to develop effective methods for image manipulation detection (IMD).

*Corresponding author: bobblair.wj@antgroup.com



(a) original (b) manipulated (c) ground truth

Figure 1: Data uncertainty refers to noisy labels as shown in the red bounding box of the first row where the child is removed from the original position but is labeled as untampered. Such data uncertainty also shows in the red mask boundary in the second row where the blurring operation makes it hard to label. Besides, model uncertainty is caused by the fact that the same object may have different labels, confusing the model training. In the first row, the same child in the original and manipulated images are labeled differently.

The main challenge of the IMD task is twofold: data uncertainty and model uncertainty. The former is induced by the fact that the manipulated artifacts are often hard to discern, posing a great challenge to annotation and resulting in noisy labels. Humans can effortlessly distinguish manipulated content by comparing original and manipulated images, *e.g*. comparing (a) and (b) in Figure 1, the different regions between them are manipulated areas. However, during the real-word labeling process, original images are usually unavailable, making annotations prone to errors. As shown in the first row, the red bounding box indicates the child has been removed, which is easily overlooked by annotators and marked as untampered. In the second row, we draw the mask boundary in red on the manipulated image. The blurring operation makes the boundary difficult to iden-

tify, resulting in an inaccurate boundary. Such data uncertainty problem poses great challenges to model learning.

As for model uncertainty, it's caused by the fact that labels of the same visual content may be inconsistent in different images. As shown in the top row of Figure 1, for conventional semantic segmentation tasks, the child is always labeled as the same category across different images, which is semantically consistent. While in this task, the same child can be labeled as both tampered and untampered, depending on whether the object is modified, which potentially confuses model learning and leads to unreliable outputs.

For the IMD research community, the primary focus of many methods is to design meticulous network structures and features that can mitigate model uncertainty. Various techniques have been proposed to detect digital image manipulations, such as those based on artifacts from resampling [5], color filter arrays [29], and SRM [43]. MVSS-Net [11] takes this a step further by suppressing image content through the learning of manipulated region edges. However, these approaches tend to address model uncertainty implicitly, and some manipulation artifacts are too subtle to capture, which can lead to over-segmentation, under-segmentation, and phantom-segmentation during the model prediction process. Furthermore, few works focus on data uncertainty. Conventional methods [44, 9, 2] and deep learning based methods [41, 35] are able to generate manipulated data with accurate labels, and may alleviate the data uncertainty issue to some extent. Nevertheless, the generated data is not identically distributed with the read-world manipulation data. So these methods fail to confront the data uncertainty problem in practical cases.

To reveal the two kinds of uncertainties, a straightforward solution is to estimate them via uncertainty estimation techniques. In these techniques, the data uncertainty and model uncertainty are also called *aleatoric* uncertainty and *epistemic* uncertainty respectively [21, 19]. The former refers to the noise inherent in the observations, which can not be reduced even if more data were to be collected. The latter accounts for uncertainty in the model and captures the lack of representativeness of the model, which can be explained away with increasing training data. Plenty of works have been proposed to model the two types of uncertainties. They usually utilize Bayesian Neural Network (BNN) to learn mappings from input data to data uncertainty and compose these together with model uncertainty approximations [21]. The main issue of BNN for uncertainty estimation is the intractable posterior inference, thus most existing uncertainty estimation techniques focus on designing approximate posterior inference [13, 21, 14, 17, 4]. [13] utilizes dropout variational inference as a practical approach for approximate inference in large and complex models. [21] uses Monte Carlo Sampling to approximate posterior inference. Following [21], we adopt the Monte Carlo Sampling for uncertainty approximation.

In our work, we propose an uncertainty-guided learning framework that incorporates a novel Uncertainty Estimation Network (UEN) to capture data and model uncertainty. UEN is comprised of two key components, namely Dynamic Uncertainty Supervision (DUS) and Uncertainty Prediction Refinement (UPR). Specifically, we derive the difference between the predicted results and ground truth as the dynamic uncertainty supervision for UEN. Thanks to the delicate design of DUS, the data uncertainty and model uncertainty maps are precisely estimated and are further integrated to refine the manipulation predictions in UPR. With the above designs, our method is capable of identifying over-segmentation, under-segmentation, and phantom-segmentation regions by assigning high uncertainty to misclassified areas and assigning low uncertainty to correctly classified regions. Extensive experiments demonstrate the effectiveness and generalizability of our method.

The main contributions of this paper are summarized as:

- We develop a new learning framework for image manipulation detection, which is applicable to existing segmentation-based manipulation detection methods. As far as we know, we are the first to introduce uncertainty modeling into image manipulation detection to resolve the challenges.
- We address data uncertainty and model uncertainty by explicitly modeling them. With dynamic supervision, our method is able to output accurate uncertainty estimation results, which are further integrated with the UPR module to improve the manipulation predictions.
- We achieve state-of-the-art results on four benchmarks for image manipulation detection. Extensive experiments on multiple benchmarks demonstrate the superiority and the generalizability of our method.

## 2. Related work

**Image manipulation detection.** Many existing IMD methods are segmentation based [25, 41, 11, 40]. Compared with general semantic segmentation, IMD is more challenging, since delicate image forgery is almost indistinguishable. In order to detect manipulation, many algorithms have been developed to extract the manipulation features and suppress semantic features. Zhou *et al.* [42, 43] harness noise features to find inconsistencies within a manipulated image. RGB-N [43] introduces a two-stream network for manipulation detection, where one stream extracts RGB features to capture visual artifacts, and the other stream leverages noise features to model the inconsistencies between manipulated and pristine regions. PSCC-Net [25] extracts hierarchical features with a top-down path and detects whether the input image has been manipulated using a

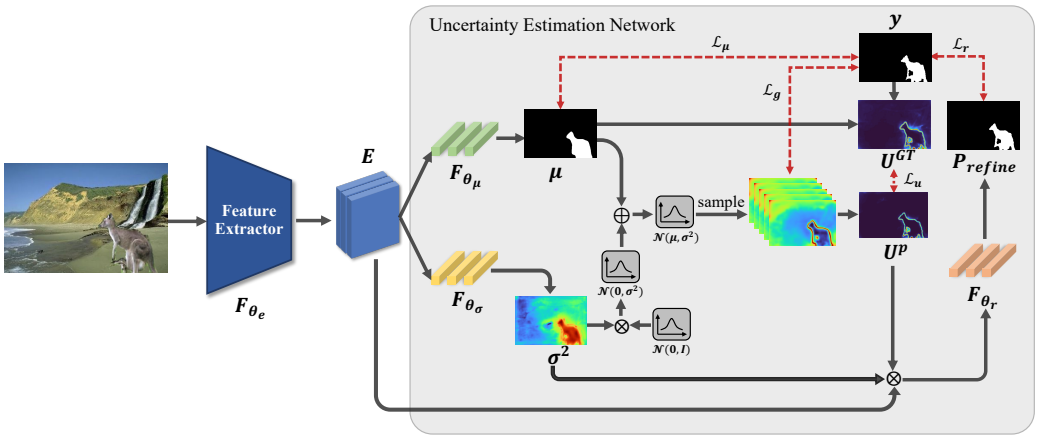



Figure 2: Overview of our Uncertainty Estimation Network for image manipulation detection. Data uncertainty $\sigma^2$ and model uncertainty $U^p$ are precisely estimated with dynamic supervision $U^{GT}$, and are further integrated to guide the refinement process.

bottom-up path. Zhou *et al.* [41] propose a GSR-Net model based on semantic segmentation, which consists of generation, segmentation, and refinement. Hu *et al.* [20] propose a so-called SPAN model to focus on important regions in a target image with attention-based design.

**Model uncertainty estimation.** The model uncertainty estimation aims to estimate the distribution of the model parameter set $p(\theta|D)$, where the $\theta$ follows some specific distribution leading to Bayesian Neural Network (BNN), and $D$ is the training dataset. The main focus of BNN is to achieve effective posterior inference $p(\theta|D)$, which is intractable in practice. In this way, existing techniques mainly work on approximate posterior inference. One line of work proposes to use Markov Chain Monte Carlo (MCMC) methods [26] and its variants [37, 16, 7] as approximation solutions [21]. Another line of work adopts a dropout-based approach [13]. For example, by applying dropout [30] at test time, multiple networks can be sampled from the approximate posterior, and the variance of the outputs can be used as a measure of uncertainty [14].

**Data uncertainty estimation.** The basic assumption for data uncertainty estimation is that the model parameter $\theta$ is fixed and unknown, which leads to non-Bayesian Neural Networks based framework. SDE-Net [22] presents a network that yields a probabilistic distribution as output in order to capture such uncertainty. [23] uses an adversarial perturbation technique to generate additional training data to capture the data uncertainty. Moreover, captured data uncertainty can be used to guide the training flow in different tasks. Wang *et al.* [36] utilize the data uncertainty to guide the data selection of multi-phase training for semi-supervised object detection. Subedar *et al.* [31] use the uncertainty to help the fusion of visual modality and audio modality for audiovisual activity recognition. Chang *et al.* [6] model the data uncertainty to reduce the adverse effects of noisy samples for face recognition.

## 3. Proposed method

To measure the uncertainties, we propose an uncertainty estimation network with DUS that generates accurate uncertainty maps and then integrates them into the network to alleviate learning difficulties through a feature refinement module called UPR. In this section, we will describe the proposed method in detail.

### 3.1. Uncertainty estimation network

For image manipulation detection, given an input image $\mathcal{I} \in \mathbb{R}^{H\times W\times 3}$, the feature embedding $E = \mathcal{F}_{\theta_e}(\mathcal{I}) \in \mathbb{R}^{h\times w\times c}$ is firstly extracted by a feature extractor $\mathcal{F}_{\theta_e}$.

In this work, we build our Uncertainty Estimation Network (UEN) as a probabilistic representational model to measure uncertainty, which is illustrated in Figure 2. Unlike conventional methods [20, 25, 8], which simply map the feature $E$ to uncertainty prediction by single parameterized MLP network, we estimate $\mu$ and $\sigma^2$ from feature $E$ by two parameterized MLP networks, $\mathcal{F}_{\theta_\mu}$ and $\mathcal{F}_{\theta_\sigma}$. The process is formulated as follows:

$$\mu = \mathcal{F}_{\theta_\mu}(E), \quad \sigma^2 = \mathcal{F}_{\theta\sigma}(E), \tag{1}$$

where $\mu \in \mathbb{R}^{h\times w\times 1}$ and $\sigma^2 \in \mathbb{R}^{h\times w\times 1}$ denote the mean map and variance map, respectively. To be specific, each pixel $i$ is predicted as a Gaussian distribution parameterized by mean $\mu_i$ and variance $\sigma_i^2$. Mean map $\mu$ is equivalent to the prediction results of conventional methods and served as coarse predictions in our model, while variance map $\sigma^2$

is Gaussian noise which captures the data uncertainty. We will elaborate on how the variance map $\sigma^2$ represents data uncertainty in Section 3.2.

As mentioned before, the output of each pixel is a random variable following Gaussian distribution. We define the probabilistic logit $z_{i,t}$ as $t$-th sample drawn from the learned distribution for the $i$-th pixel, and probability score $p_{i,t}$ is simply the output from the sigmoid function:

$$\begin{aligned} z_{i,t} &\sim \mathcal{N}(\mu_i, \sigma_i^2), \\ p_{i,t} &= \mathrm{Sigmoid}(z_{i,t}), \end{aligned} \tag{2}$$

where $\mu_i$ and $\sigma_i^2$ are produced by Equation (1). Since the gradients can not be backpropagated during training when directly sampling $z_{i,t}$ from the Gaussian distribution, we adopt the reparameterization trick [3] to perform the sampling. Specifically, we first draw a variable $\epsilon_t$ from the standard Gaussian distribution $\mathcal{N}(0, I)$ randomly, *i.e.*, $\epsilon_t \sim \mathcal{N}(0, I)$, and then obtain a sample $z_{i,t}$ by the following computation:

$$z_{i,t} = \mu_i + \epsilon_t \sigma_i^2, \quad \epsilon_t \sim \mathcal{N}(0, I). \tag{3}$$

where $t$ represents the $t$-th sampling. By doing so, the backpropagation works properly. Through Equation (3), the mean map $\mu$ is then corrupted with Gaussian noise $\sigma^2$. We finally summarise the estimated model uncertainty as an expectation of entropy of $T$ times Monte Carlo sampling:

$$U^p = -\frac{1}{T} \sum_{t=1}^{T} p_t \log p_t + (1 - p_t) \log(1 - p_t), \tag{4}$$

where $U^p$ is the estimated model uncertainty map and is normalized to $[0, 1]$, $p_t$ is the estimated manipulation score map in $t$-th sampling. The intuition is that when the sampled $p_t$ tends to be its upper or lower bound, it means the model makes a highly confident prediction, resulting in low model uncertainty $U^p$. When $p_t$ is close to the value in the middle of its legitimate range, this indicates the model is quite unsure about the result, leading to high model uncertainty $U^p$.

### 3.2. Dynamic uncertainty supervision

We optimize the uncertainty estimation with Dynamic Uncertainty Supervision (DUS), which is derived as:

$$U^{GT} = y \odot (1 - \hat{\mu}) + (1 - y) \odot \hat{\mu}, \tag{5}$$

where $\odot$ denotes Hadamard product, $\hat{\mu}$ is calculated by $\mathrm{Sigmoid}(\mu)$, $y$ is the ground truth corresponding to the input image $\mathcal{I}$. $U^{GT}$ captures difference between ground truth $y$ and $\hat{\mu}$. When the network makes uncertain predictions, Equation (5) assigns high uncertainty to $U^{GT}$, and vice versa. For example, if the label of the $i$-th pixel is 1, denoting a manipulated pixel, while the prediction result is $0.4$, indicating false and uncertain pixel, *i.e.*, $y_i = 1$, $\hat{\mu}_i = 0.4$, $U_i^{GT}$ is assigned with $0.6$ to represent the high uncertainty on this pixel. $\hat{\mu}$ varies along with the model iteration, hence $U^{GT}$ also adapts dynamically in accordance with $\hat{\mu}$ to provide more accurate supervision.

With dynamic supervision, uncertainty estimation is optimized via the following objective function:

$$\mathcal{L}_u = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} \sigma_i^{-2} \underbrace{||U_i^{GT} - U_i^p||^2}_{\mathrm{L2}} + \frac{1}{2} \log \sigma_i^2, \tag{6}$$

where $N$ is the number of total pixels in the predicted result, L2 loss is employed for dynamic model uncertainty supervision. In order to comprehend the design intuition of $\mathcal{L}_u$, let's consider two possible cases that will increase $U_i^{GT}$: 1. ground truth $y_i$ is correct, and our coarse prediction $\mu_i$ is wrong; 2. ground truth $y_i$ is wrong while predicted $\mu_i$ is exactly correct. The former one is expected to be captured by our model uncertainty $U_i^p$, since the model is giving wrong predictions and its confidence is prevalently low, leading to a relatively small L2 value. The latter is the source of data uncertainty. In such a scenario, $U_i^{GT}$ is still large due to the large discrepancy between $y_i$ and $\hat{\mu}_i$, while $U_i^p$ is small as the model's prediction is correct. Then variance $\sigma_i^2$ is maximized to reduce the large L2 value caused by the difference between $U_i^{GT}$ and $U_i^p$, which perfectly fits the definition of data uncertainty, as $\sigma_i^2$ increases when data uncertainty occurs. $\frac{1}{2} \log \sigma_i^2$ is the regularization term to prevent $\sigma_i^2$ from becoming infinitely large. The above reasoning concludes that $\sigma^2$ is a good measurement for data uncertainty. Equation (6) provides direct supervision for uncertainty estimation and improves the reliability of uncertainty estimation.

### 3.3. Uncertainty-guided prediction refinement

Image manipulation detection has different learning difficulties across the image. Firstly, manipulated content usually blends well with the background, and the pixels along the manipulated mask boundaries are fairly hard to distinguish. Secondly, the manipulated artifacts are subtle and are more prone to over-segmentation, under-segmentation, and phantom-segmentation during the model prediction process. We intend to solve such difficulties by designing Uncertainty-guided Prediction Refinement (UPR). Concretely, we propose to couple the feature embedding $E$ with weighted uncertainty maps. The formulation is presented as follows:

$$\begin{aligned} E_w &= E \odot (\alpha \sigma^2 + \beta U^p), \\ E_f &= Concat(E, E_w), \end{aligned} \tag{7}$$

where $\odot$ denotes Hadamard product operation, $\alpha$ and $\beta$ are hyperparameters to adjust the weights between the estimated data uncertainty map $\sigma^2$ and model uncertainty map

**Algorithm 1** Uncertainty-guided IMD Framework

**Input**: Training dataset $\mathcal{D} = \{\mathcal{I}_k, y_k\}_{k=1}^{K}$, where $K$ is the size of training dataset; A feature extractor $\mathcal{F}_{\theta_e}$ with parameters $\theta_e$; Two MLP heads $\mathcal{F}_{\theta_\mu}$ and $\mathcal{F}_{\theta_\sigma}$ with parameters $\theta_\mu$ and $\theta_\sigma$; A refine layer $\mathcal{F}_{\theta_r}$; Maximal training epochs $M$.
**Output**: Parameter set $\theta = \{\theta_e, \theta_\mu, \theta_\sigma, \theta_r\}$ for whole framework

1: **for** $m = 1$ to $M$ **do**
2: Extract feature embedding $E$ by $\mathcal{F}_{\theta_e}(\mathcal{I}_k)$, and then transfer $E$ to mean map $\mu$ and variance map $\sigma^2$ by Equation (1).
3: Sample probabilistic logits $z_{i,t}$ from a specific Gaussian distribution to form a probability score $p_{i,t}$ through Equation (2), and further estimate model uncertainty $U^p$ by Equation (4).
4: Generate dynamic supervision $U^{GT}$ for uncertainty estimation optimization with Equation (5).
5: Integrating both data uncertainty map $\sigma^2$ and model uncertainty map $U^p$ with $E$ to form refined feature $E_f$ by Equation (7), and then produce the final predictions $P_{refine}$ through $\mathcal{F}_{\theta_r}(E_f)$.
6: Update the whole framework with the total loss function in Equation (11).
7: **end for**

$U^p$, $E$ and $E_w$ are concatenated in channel dimension to form the refined feature $E_f$. The final predicted results are then obtained through $P_{refine} = \mathcal{F}_{\theta_r}(E_f))$. $\mathcal{F}_{\theta_r}$ is comprised of two consecutive convolution layers with $1 \times 1$ convolution kernel size, and each is followed by a batch normalization layer and a ReLU [15] activation function. The whole training procedure is illustrated in Algorithm 1, which is applicable to other existing segmentation-based IMD methods.

### 3.4. Loss functions

The total loss of our UEN is composed of four parts. Firstly, for the samples which are composed to estimate the model uncertainty in Equation (2), our expected log-likelihood is given by:

$$\log \mathbb{E}_{\mathcal{N}(z_i;\mu_i,\sigma_i^2)}[p_i]. \tag{8}$$

However, directly computing the expectation is intractable, we thus approximate the objective through $T$ times Monte Carlo sampling, and formulate the following loss to make training more stable:

$$\mathcal{L}_g = \frac{1}{N}\sum_{i=1}^{N} \log \frac{1}{T}\sum_{t=1}^{T} \exp(y_{i,t}\log p_{i,t} + (1-y_{i,t})\log(1-p_{i,t})). \tag{9}$$

Secondly, the coarse prediction $\mu$ and refined prediction $P_{refine}$ are supervised by the binary ground truth $y$ with Binary Cross Entropy (BCE) loss functions:

$$\begin{aligned} \mathcal{L}_\mu &= -(y\log\mu + (1-y)\log(1-\mu)), \\ \mathcal{L}_r &= -(y\log P_{refine} + (1-y)\log(1-P_{refine})). \end{aligned} \tag{10}$$

Finally, we train our framework with a total loss function:

$$\mathcal{L}_{total} = \mathcal{L}_\mu + \mathcal{L}_r + \mathcal{L}_u + \mathcal{L}_g, \tag{11}$$

where $\mathcal{L}_u$ is the loss function for supervised uncertainty estimation introduced in Equation (6).

## 4. Experimental results

### 4.1. Experimental settings

**Datasets.** In this work, we evaluate our method on four standard datasets, *i.e.*, CASIA [12], NIST16 [18], Columbia [27], and COVERAGE [38]. **CASIA** has two versions. CASIA v1.0 contains 921 manipulated images along with ground truth masks provided by original builders [12]. CASIA v2.0 includes 5,123 manipulated samples whose ground truth generated by Pham *et al.* [28] is believed to be noisy despite the wide distribution. Both versions contain spliced and copy-move images in which forged regions are carefully selected. Following [25, 34, 41], we use CASIA v2.0 for training and CASIA v1.0 for testing. **NIST16** is composed of 564 samples manipulated with splicing, copy-move, or removal. Following [25, 34], we use 404 samples for training and 160 samples for testing. **Columbia** provides 180 spliced images. The images are mostly indoor scenes, and a fraction of images are taken outdoors. **COVERAGE** dataset contains 100 images generated by copy-move techniques. Both Columbia and COVERAGE are exploited for testing the model trained on CASIA v2.0 only. Apart from CASIA v2.0, mask labels of the aforementioned datasets are provided by dataset builders. We believe that the labels are accurate and the test results are trustworthy. For a fair comparison, the training and testing protocols of datasets are the same as [25].

**Evaluation metrics.** We evaluate our model at pixel level with the benchmark metrics: F1 score and Area Under the Curve (AUC). F1 score and AUC measure the performance of binary classification for every pixel, where higher scores indicate better performance.

**Implementation details.** Our method is a general paradigm that can be easily extended to existing IMD methods. Without loss of generality, we present a feature extractor with a general design as the *vanilla* baseline and integrate our method into it. Specifically, we adopt HRNetV2 [33] as a primary backbone network with 4 stages to extract RGB features. Our feature extractor also includes practical SRM [43] and resampling features [1]. Concretely, the SRM branch uses a parallel structure as the backbone

| Method | CASIA v1.0 | | NIST16 | |
|---|---|---|---|---|
| | AUC | F1 | AUC | F1 |
| RGB-N* [43] | 79.5 | 40.8 | 93.7 | 72.2 |
| ManTra-Net* [39] | 81.7 | - | - | 79.5 |
| SPAN* [20] | 83.8 | 38.2 | 96.1 | 58.2 |
| MFAN* [32] | - | 59.7 | 91.8 | - |
| PSCC-Net* [25] | 87.5 | 55.4 | 99.6 | 81.9 |
| Constrained R-CNN* [40] | 78.9 | 47.5 | 99.2 | 92.7 |
| MVSS-Net [8] | - | 45.2 | - | - |
| GSR-Net [41] | - | 57.4 | - | - |
| **Ours** | **87.7** | **62.9** | **99.6** | **93.2** |

Table 1: Pixel-level AUC and F1 score comparison on CASIA v1.0 and NIST16 test sets. '-' denotes that the result is not available in the literature. '$*$' denotes training with additional synthetic datasets which include upwards of 42k$\sim$400k images.

network and is fused with the RGB branch at the end of the corresponding stages. The resampling branch is started in the 4th stage and is fused with the other two branches at the end of the 4th stage.

We train our method in an end-to-end manner. The RGB stream network of the feature extractor is initialized with ImageNet [10] classification pre-trained weights, and the weights of the SRM stream and resampling stream are randomly initialized. In the training process, input images are first resized so that the shorter length is 512, and then randomly cropped to $512 \times 512$. To avoid performance degradation caused by size mismatch between training and testing, we resize images to $512 \times 512$ during the test phase. In the following experiments, sampling frequency $T$ is set to 100, and the hyper-parameters $\alpha$ and $\beta$ are set to 0.5 unless otherwise specified. SGD is adopted as the optimizer, with the initial learning rate set to $5 \times 10^{-5}$ and decayed exponentially. The model is implemented with PyTorch deep learning framework and trained for 240 epochs on 8 NVIDIA V100 GPUs with a batch size of 8 per GPU. All training is carried out with the same batch size and random seed.

### 4.2. Comparison with state-of-the-art approaches

We compare our method against existing state-of-the-art approaches on four standard benchmarks. Table 1 shows the performance comparison on the CASIA and NIST16 datasets. Our method achieves the highest performance on both metrics over two datasets. Concretely, in terms of F1 score, ours surpasses the second best by 3.2% on the CASIA v1.0 set and is 0.5% higher than the second best on the NIST16 set. Besides, on Columbia and COVERAGE datasets, since other methods only report the AUC metric, we report the pixel-level AUC results in Figure 3. Apparently, our method outperforms other works by a large margin. It is worth noting that many of the methods are trained with additional synthetic data to achieve competitive performance. For instance, PSCC-Net creates a synthetic dataset with 400k manipulated images by using the segmentation annotations in COCO [24] to randomly select objects and then splice them to other images. With abundant training data, PSCC-Net achieves 98.2% in pixel-level AUC on the Columbia dataset. We believe that adding additional synthetic data for training may further improve the performance of our method, but it is not the focus of our work. Thanks to the uncertainty modeling and uncertainty-guided learning design, our method outperforms these approaches, which demonstrates the superiority of our method.

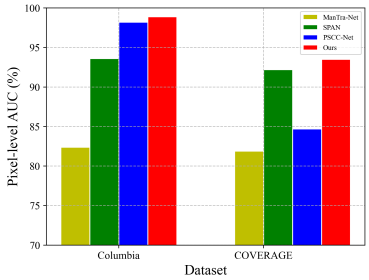



Figure 3: Pixel-level AUC comparison on Columbia and COVERAGE datasets. ManTra-Net, SPAN, and PSCC-Net are trained with additional synthesized data. However, our method is trained on CASIA v2.0 solely.

### 4.3. Ablation study

**Effectiveness and versatility of UEN.** Our UEN is a pluggable module that can be easily extended to other segmentation-based IMD methods. Thus, apart from ablating UEN on our baseline, we further deploy our UEN to PSCC-Net[1] to verify its versatility. The ablation studies and results are shown in Table 2. When UEN is not employed, *vanilla* baseline is a conventional segmentation-based IMD model with 84.6%/59.7% results in F1/AUC. Our UEN further boosts the performance with 3.1%/3.2% gains over AUC and F1, demonstrating its effectiveness. Besides, we integrate UEN to PSCC-Net with two kinds of backbone initialization manners, *i.e.*, initializing with ImageNet classification weights or large-scale synthesized datasets pre-trained weights. For PSCC-Net using the weights pre-trained on large-scale synthesized datasets, we reproduced the results, and get 87.7%/55.1% over AUC and F1, which are close to the numbers in literature, *i.e.*, 87.5%/55.4%. Our UEN further improves the results with 1.6% and 1.4% on F1 and AUC, respectively. When using ImageNet pre-

[1]Few existing IMD works have published trainable open-source code. Thus, we use PSCC-Net for evaluation, whose official code is in https://github.com/proteus1991/PSCC-Net.

| Baseline | Initialization | UEN | AUC / F1 |
|---|---|---|---|
| PSCC-Net | ImageNet | w/o | 84.2 / 52.5 |
| | | w/ | 85.4 / 54.6 |
| PSCC-Net | Synthesized data | w/o | 87.7 / 55.1 |
| | | w/ | 89.3 / 56.5 |
| *vanilla* | ImageNet | w/o | 84.6 / 59.7 |
| | | w/ | 87.7 / 62.9 |

Table 2: Ablation study of our UEN on CASIA dataset. We verify the effectiveness of our UEN by applying it on *vanilla* baseline model and PSCC-Net with two kinds of backbone initialization manners.

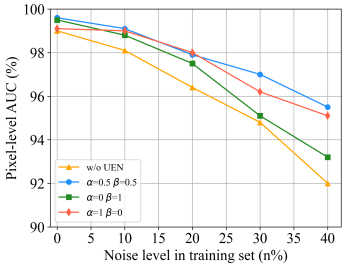



Figure 4: Model robustness against different noise levels on NIST16 dataset. We adopt different $\alpha$ and $\beta$ in UEN.

trained weights, the performances of PSCC-Net are improved by UEN with 1.2% and 2.1%. Significantly, without large synthetic data pre-training, PSCC-Net achieves competitive F1 by integrating UEN (54.6% versus 55.1%). It demonstrates that model and data uncertainties are essential for IMD tasks, and our UEN is a general and adaptable module to boost performance, regardless of framework and initialization variance.

**Robustness of UEN.** To verify the robustness of UEN against label noise, we report the results of our method whose training set is corrupted with label noise in different levels in Figure 4. Given that the labels of NIST16 dataset are provided by the dataset builder, we consider the noise level for this dataset to be 0%. We randomly select $n$% images in the training set and reversely set the labels of their manipulated area to untampered. In this way, we obtain a training set with a noise level of $n$%, involving various degrees of data uncertainty. Besides, we adopt different $\alpha$ and $\beta$ which determine the compositions in the refined feature. As shown in Figure 4, we observe that training with and without UEN, their results gradually decrease according to the increase in noise level. However, our UEN provides more pronounced improvements when the noisy data dominates the whole training set, demonstrating the robustness of the UEN to training noise. Furthermore, when $\alpha = 0, \beta = 1$, indicating the data uncertainty map is not considered in refinement, the performance is much worse

| Method | CASIA v1.0 | NIST16 |
|---|---|---|
| Full model | 87.7 / 62.9 | 99.6 / 93.2 |
| w/o DUS | 86.7 / 61.8 | 99.2 / 91.8 |
| w/o UPR | 85.1 / 61.5 | 99.0 / 92.1 |

Table 3: Ablation study of DUS and UPR on CASIA and NIST16. Pixel-level AUC and F1 (%) score are reported.

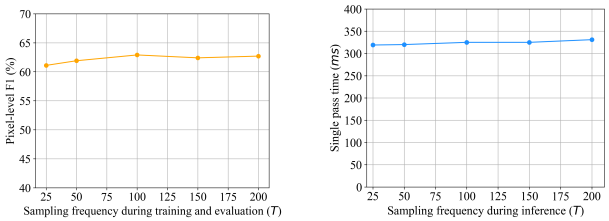



Figure 5: The influence of sampling frequency in performance (left) and inference speed (right) on CASIA dataset.

when the noise level is greater than 30%, proving the effectiveness of the data uncertainty modeling in training against noise.

**Effectiveness of DUS and UPR.** We conduct several ablation studies on DUS and UPR, as shown in Table 3. For the DUS ablation study, we discard the $\mathcal{L}_u$ in Equation (6) during training and observe significant degradation in AUC and F1 on both CASIA v1.0 and NIST16 datasets. For example, without DUS, our method decreases to 91.8% in F1 score with a 1.4% decline in NIST16, which in turn proves the effectiveness of our DUS method. For the UPR ablation study, we remove the refinement process during both training and testing and select the predicted mean map $\mu$ as the predicted result for evaluation. The severe deterioration on two datasets, such as 2.6% descent in AUC of CASIA v1.0 after removing UPR, indicates the effectiveness of UPR. More qualitative analysis results are presented in Section 4.4.

**Impact of sampling frequency.** We investigate the impact of sampling frequency on model performance in Figure 5. The sampling frequency $T$ is set from 25 to 200, and kept the same during training and evaluation. In the left figure, the F1 scores under different sampling frequencies are stable when $T$ is greater than 100, demonstrating that our method is robust to the sampling frequency. Furthermore, we evaluate the effect of sampling frequency on inference speed in the right figure of Figure 5. It can be observed that when $T$ ranges from 25 to 200, the single pass time increases very slightly. This is because we only need to sample from the logits, which is a small fraction of the network's computation and therefore does not significantly increase the inference time.

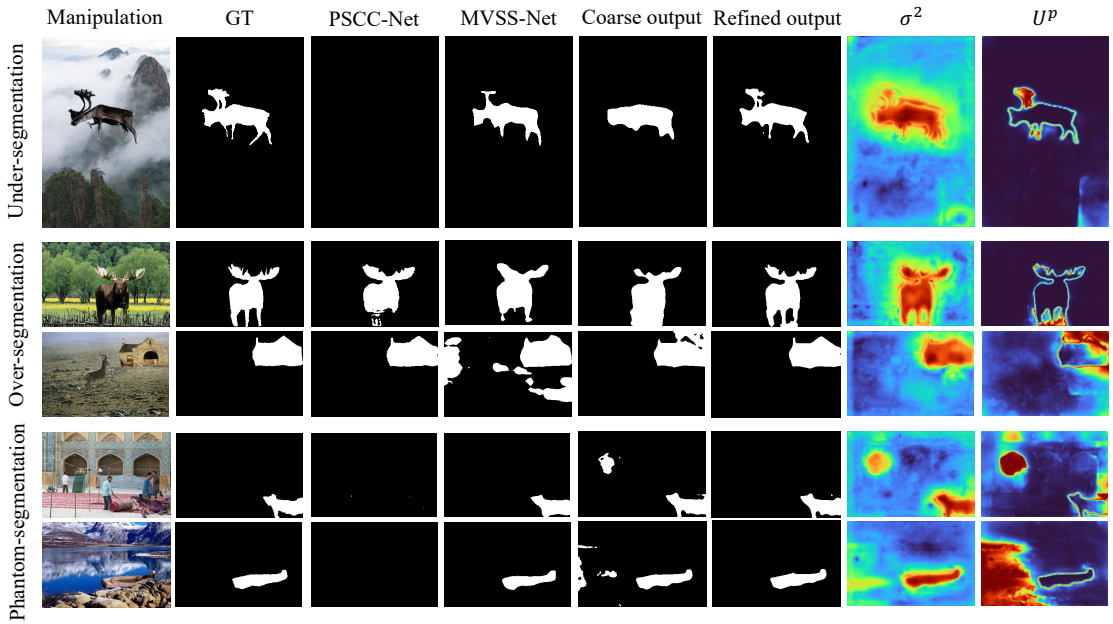



Figure 6: Predictions of PSCC-Net, MVSS-Net and our method on CASIA v1.0. From left to right are manipulated image, ground truth, prediction of PSCC-Net, prediction of MVSS-Net, coarse output of our method, refined output of our method, data uncertainty map $\sigma^2$, and model uncertainty map $U^p$.

### 4.4. Qualitative analysis

To further prove the effectiveness of our method, in this section, we provide qualitative examples which are taken from the CASIA v1.0 for demonstration. In the following analysis, we discuss two aspects: 1. How UEN works? 2. Comparison with SOTA methods. The qualitative results are shown in Figure 6.

*How UEN works?* We visualize the coarse prediction $\mu$, refined prediction, data uncertainty map $\sigma^2$, and model uncertainty map $U^p$ in Figure 6. Due to the inconsistent annotation and the difficulty of boundary labeling, noise often occurs in manipulated regions. Thus, the manipulated objects and corresponding boundaries dominate the data uncertainty signal, which agrees with the highlighted area in our data uncertainty map $\sigma^2$. Model uncertainty is generally concentrated in the misclassified regions of coarse prediction and the corresponding boundaries, which is also captured by our model uncertainty map $U_p$. Compared with coarse prediction and refined prediction, the latter corrects the misclassified regions by integrating the uncertainty maps, and outputs meticulous segmentation results, demonstrating the effectiveness of our UPR. In addition, the first row in Figure 6 illustrates the under-segmentation case of our coarse output, the second and third rows show the over-segmentation cases, and the last two rows demonstrate the phantom-segmentation cases. By coupling UEN, our method is capable of identifying the above cases by assigning high uncertainty to misclassified areas and assigning low uncertainty to correctly classified regions.

*Comparison with SOTA methods.* We also display the qualitative comparison of our method with other state-of-the-art methods in Figure 6. It is clear that our method outperforms the state-of-the-art methods. For example, the result of PSCC-Net in the second row is under-segmented, and the result of MVSS-Net in the third row is phantom-segmented. On the contrary, our results are more accurate, and finer in the boundaries, illustrating the superiority of our method.

## 5. Conclusion

In this paper, we revisit image manipulation detection and consider two uncertainty problems in this task, *i.e.*, data uncertainty and model uncertainty. To solve these problems, we introduce an uncertainty-guided image manipulation detection method, which captures annotation noises by modeling data uncertainty and alleviates semantic inconsistency by modeling model uncertainty. We train our novel Uncertainty Estimation Network under dynamic uncertainty supervision, which benefits accurate uncertainty estimation. Moreover, we integrate both the data uncertainty and the model uncertainty maps to guide the refinement of manipulation predictions. Extensive experiments demonstrate the generalizability of our method and the advantages against existing SOTA methods. In the future, we plan to extend our UEN to more methods and scenarios.

## References


[1] Jawadul H. Bappy, Cody Simons, Lakshmanan Nataraj, B. S. Manjunath, and Amit K. Roy-Chowdhury. Hybrid lstm and encoder-decoder architecture for detection of image forgeries. *IEEE TIP*, 28(7):3286–3300, 2019.

[2] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transaction on Graphics*, 28(3):24, 2009.

[3] Avrim Blum, Nika Haghtalab, and Ariel D. Procaccia. Variational dropout and the local reparameterization trick. In *NeurIPS*, pages 2575–2583, 2015.

[4] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In Francis R. Bach and David M. Blei, editors, *ICML*, volume 37, pages 1613–1622, 2015.

[5] Jason Bunk, Jawadul H. Bappy, Tajuddin Manhar Mohammed, Lakshmanan Nataraj, Arjuna Flenner, B. S. Manjunath, Shivkumar Chandrasekaran, Amit K. Roy-Chowdhury, and Lawrence Peterson. Detection and localization of image forgeries using resampling features and deep learning. In *CVPR Workshop*, pages 1881–1889, 2017.

[6] Jie Chang, Zhonghao Lan, Changmao Cheng, and Yichen Wei. Data uncertainty learning in face recognition. In *CVPR*, pages 5709–5718, 2020.

[7] Tianqi Chen, Emily B. Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *ICML*, volume 32, pages 1683–1691, 2014.

[8] Xinru Chen, Chengbo Dong, Jiaqi Ji, Juan Cao, and Xirong Li. Image manipulation detection by multi-view multi-scale supervision. In *ICCV*, pages 14165–14173, 2021.

[9] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE TIP*, 13(9):1200–1212, 2004.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[11] Chengbo Dong, Xinru Chen, Ruohan Hu, Juan Cao, and Xirong Li. Mvss-net: Multi-view multi-scale supervised networks for image manipulation detection. *IEEE TPAMI*, 2022.

[12] Jing Dong, Wei Wang, and Tieniu Tan. CASIA image tampering detection evaluation database. In *IEEE China Summit and International Conference on Signal and Information Processing*, 2013.

[13] Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*, 2015.

[14] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *ICML*, volume 48, pages 1050–1059, 2016.

[15] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *AISTATS*, volume 15, pages 315–323, 2011.

[16] Wenbo Gong, Yingzhen Li, and José Miguel Hernández-Lobato. Meta-learning for stochastic gradient MCMC. 2018.

[17] Alex Graves. Practical variational inference for neural networks. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *NeurIPS*, pages 2348–2356, 2011.

[18] Haiying Guan, Mark Kozak, Eric Robertson, Yooyoung Lee, Amy N. Yates, Andrew Delgado, Daniel Zhou, Timothee Kheyrkhah, Jeff Smith, and Jonathan Fiscus. Mfc datasets: Large-scale benchmark datasets for media forensic challenge evaluation. In *WACV Workshop*, pages 63–72, 2019.

[19] Hongji Guo, Hanjing Wang, and Qiang Ji. Uncertainty-guided probabilistic transformer for complex action recognition. In *CVPR*, pages 20020–20029, 2022.

[20] Xuefeng Hu, Zhihan Zhang, Zhenye Jiang, Syomantak Chaudhuri, Zhenheng Yang, and Ram Nevatia. SPAN: spatial pyramid attention network for image manipulation localization. In *ECCV*, volume 12366, pages 312–328, 2020.

[21] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *NeurIPS*, pages 5574–5584, 2017.

[22] Lingkai Kong, Jimeng Sun, and Chao Zhang. Sde-net: Equipping deep neural networks with uncertainty estimates. In *ICML*, volume 119, pages 5405–5415, 2020.

[23] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS*, pages 6402–6413, 2017.

[24] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, volume 8693, pages 740–755, 2014.

[25] Xiaohong Liu, Yaojie Liu, Jun Chen, and Xiaoming Liu. Pscc-net: Progressive spatio-channel correlation network for image manipulation detection and localization. *IEEE TCSVT*, 32(11):7505–7517, 2022.

[26] Radford M. Neal. *Bayesian learning for neural networks*. PhD thesis, 2012.

[27] Adam Novozamsky, Babak Mahdian, and Stanislav Saic. Imd2020: A large-scale annotated dataset tailored for detecting manipulated images. In *WACV Workshop*, pages 71–80, 2020.

[28] Nam Thanh Pham, Jong-Weon Lee, Goo-Rak Kwon, and Chun-Su Park. Hybrid image-retrieval method for image-splicing validation. *Symmetry*, 11(1):83, 2019.

[29] Amneet Singh, Gurinder Singh, and Kulbir Singh. A markov based image forgery detection approach by analyzing CFA artifacts. *Multimedia Tools and Applications*, 77(21):28949–28968, 2018.

[30] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014.

[31] Mahesh Subedar, Ranganath Krishnan, Paulo Lopez-Meyer, Omesh Tickoo, and Jonathan Huang. Uncertainty aware audiovisual activity recognition using deep bayesian variational inference. In *CVPR Workshop*, pages 103–108, 2019.

[32] Yu Sun, Rongrong Ni, and Yao Zhao. MFAN: multi-level features attention network for fake certificate image detection. *Entropy*, 24(1):118, 2022.

[33] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep

high-resolution representation learning for visual recognition. *IEEE TPAMI*, 43(10):3349–3364, 2021.

[34] Junke Wang, Zuxuan Wu, Jingjing Chen, Xintong Han, Abhinav Shrivastava, Ser-Nam Lim, and Yu-Gang Jiang. Objectformer for image manipulation detection and localization. In *CVPR*, pages 2354–2363, 2022.

[35] Xinyi Wang, Shaozhang Niu, and He Wang. Image inpainting detection based on multi-task deep learning network. *IETE Technical Review*, 38(1):149–157, 2021.

[36] Zhenyu Wang, Yali Li, Ye Guo, Lu Fang, and Shengjin Wang. Data-uncertainty guided multi-phase learning for semi-supervised object detection. In *CVPR*, pages 4568–4577, 2021.

[37] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *ICML*, pages 681–688, 2011.

[38] Bihan Wen, Ye Zhu, Ramanathan Subramanian, Tian-Tsong Ng, Xuanjing Shen, and Stefan Winkler. Coverage - a novel database for copy-move forgery detection. In *ICIP*, pages 161–165, 2016.

[39] Yue Wu, Wael AbdAlmageed, and Premkumar Natarajan. Mantra-net: Manipulation tracing network for detection and localization of image forgeries with anomalous features. In *CVPR*, pages 9543–9552, 2019.

[40] Chao Yang, Huizhou Li, Fangting Lin, Bin Jiang, and Hao Zhao. Constrained rcnn: A general image manipulation detection model. In *ICME*, pages 1–6, 2020.

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

[42] Peng Zhou, Xintong Han, Vlad I. Morariu, and Larry S. Davis. Two-stream neural networks for tampered face detection. In *CVPR Workshop*, pages 1831–1839, 2017.

[43] Peng Zhou, Xintong Han, Vlad I. Morariu, and Larry S. Davis. Learning rich features for image manipulation detection. In *CVPR*, pages 1053–1061, 2018.

[44] Peiyu Zhuang, Haodong Li, Shunquan Tan, Bin Li, and Jiwu Huang. Image tampering localization using a dense fully convolutional network. *IEEE TIFS*, 16:2986–2999, 2021.