Reliable Camera Model Identification Through Uncertainty Estimation

Jiarong Pan, Anatol Maier, Benedikt Lorch, Christian Riess
IT Security Infrastructures Lab
Friedrich-Alexander University Erlangen-Nürnberg
Erlangen, Germany
{gary.pan, anatol.maier, benedikt.lorch, christian.riess}@fau.de

***Abstract*—Camera model identification is a standard task in digital image forensics. Learning-based approaches achieve state-of-the-art performance, but they are sensitive to so-called out-of-distribution (OOD) data due to a mismatch between the training and testing distribution. This may result in a significant reduction in classifier performance that is, unfortunately, not easy to anticipate for a forensic analyst.**

In this work, we investigate possibilities for adding reliability measures to the task of camera model identification. We leverage learning architectures that include an uncertainty measure with every prediction that can be reported back to an analyst. To this end, we investigate deep ensembles and Bayesian neural networks (BNNs). We compare both methods against a standard CNN with softmax statistics as uncertainty metric. We demonstrate in several experiments that both probabilistic approaches provide simultaneously state-of-the-art classification performance and reliable uncertainty estimates on OOD data. The uncertainty of deep ensembles is more accurate on OOD camera models, while BNN uncertainties are more accurate on OOD post-processing.

***Index Terms*—digital image forensics, camera model identification, reliability, training-test mismatch**

I. Introduction

Digital image forensics aims to validate the source and authenticity of digital images with applications like fraud detection, journalistic fact checking, and criminal investigations. Most state-of-the-art forensic methods attain impressive performance by leveraging deep learning, *e.g.*, extracting noise fingerprints of camera models [1], detecting image manipulations [2], and identifying computer-generated images [3]. The success of these data-driven approaches can be mainly attributed to the excellent ability of neural networks to learn forensics traces from the training data.

Unfortunately, learning-based methods rely on the assumption that the data on which they are evaluated is drawn from a very similar distribution as the training data. With increasing popularity of smartphone devices and social networks, however, this assumption is increasingly difficult to satisfy. Oftentimes, forensic analysts have limited knowledge about the history of an image, especially when the images of interest stem from the internet. Recent work suggests that classifiers are only able to generalize to features seen during training [4]–[6]. Therefore, unknown cameras and unseen post-processing pose an open challenge for forensic analysis. In the worst case, an image with unseen post-processing or from an unknown camera model is wrongly attributed to a known class with high confidence. Standard neural networks provide barely more information for detecting such failure cases.

Two possibilities to counter such cases of out-of-training-distribution (OOD) inputs are to enlarge the training set by including as many camera models as possible [7, 8], or through extensive data augmentation with commonly seen processing and manipulation operations [1, 6, 8, 9]. However, we argue that both approaches are difficult to scale to the combinatorial multiplicity of real-world hardware and software processing.

Another possibility to cope with OOD data is to explicitly include uncertainty in the classifier development. There are several possibilities to achieve this. Deep ensembles express uncertainty as disagreement of multiple experts. Bayesian neural networks model the network weights as probability distributions, and quantify the variability of the output. Two first works investigate Bayesian methods for forensic tasks, namely for safeguarding resampling detection and JPEG double-compression detection against unseen operations [10, 11].

In this work, we investigate both deep ensembles and Bayesian neural networks (BNNs) for camera model identification. We focus on the reliability of detecting images from unseen sources or with an unseen post-processing history, which we simulate by simple means with additive noise, Gaussian blur, and JPEG double-compression. Both network types are based on the camera model identification CNN by Bayar and Stamm [2], but provide an additional uncertainty metric to detect OOD inputs. In detail, our contributions are:

(i) We compare three network variants for uncertainty quantification: a baseline CNN [2] with softmax statistics, a deep ensemble variant, and a BNN variant of the network.
(ii) We show that deep ensembles and BNNs achieve in-distribution detection accuracies that are comparable to the original network, but they achieve significantly better rejection capabilities for out-of-distribution data.
(iii) We discuss several findings on the advantages and disadvantages of deep ensembles and BNNs, and the impact of the parameters for calculating uncertainty either via

Work was supported by the German Federal Ministry of Education and Research (BMBF) under Grant No. 13N15319, and Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) as part of the Research and Training Group 2475 "Cybercrime and Forensic Computing" (grant #393541319/GRK2475/1-2019).

the predictive entropy or the epistemic uncertainty to aid further development of reliable forensic methods.

This paper is organized as follows. Section II presents related work. Section III introduces deep ensembles and BNNs, and Sec. IV the uncertainty measures. Our experimental results are reported in Sec. V. Section VI concludes this work.

II. Related Work

A. Camera Model Identification

Although information about the camera make and model can be retrieved from the metadata, this information can easily be modified, or is removed, *e.g.*, upon sharing on social media platforms. As an alternative, forensic researchers have identified a number of imperceptible processing traces in the pixel content of an image. Such pixel-level traces are more difficult to modify, and hence enable potentially better imaging device linkage [12].

Early methods use analytically derived cues, that aim, *e.g.*, at color demosaicking [13] and JPEG compression characteristics [14]. However, the increasing complexity of camera processing modules and software variants makes it difficult to develop sufficiently complex analytic models to isolate these traces. As an alternative approach, learning-based methods have recently shown great success in extracting characteristic traces for camera model identification [1, 2, 15].

However, learning-based methods are challenged by images with unseen post-processing [4, 10, 11]. Most studies hence assume a closed-set scenario, in which an image is assumed to come from a specific set of known camera models with known post-processing. In many applications, however, this approach requires extremely large training sets. Hence, the open-set scenario aims at detecting images from unknown models or with unknown processing. To this end, Bayar and Stamm propose a binary classifier [16] that requires a representative set of known-unknown cameras. Júnior *et al.* evaluate different combinations of open-set classifiers, hand-crafted and deep features, and training protocols [17]. While the authors identify several promising methods, these methods require separate feature extraction and classification stages. In contrast, the investigated methods in our work are end-to-end neural networks.

B. Out-of-distribution Detection and Uncertainty Modeling

The machine learning community proposed several methods for estimating uncertainty as a way for detecting OOD inputs. As a simple baseline, the statistics of the network softmax output can be used to identify OOD samples [16, 18]. Neural networks tend to assign higher softmax scores to in-distribution examples. However, these simple statistics are susceptible to incorrect predictions with low uncertainty [10]. To alleviate this issue, some lines of work investigated the calibration of softmax scores [19] and training with perturbed inputs for increasing the discrepancy between in-distribution and OOD softmax scores [20].

A potentially more robust approach to uncertainty modeling is via ensembling. Here, Monte Carlo dropout (*MC-dropout*) uses a standard CNN with dropout at test time to simulate an ensemble of networks [21]. Lakshminarayanan *et al.* trained a standard CNN multiple times with random initialization and random shuffling of the training points [22]. The resulting *deep ensemble* contained sufficient diversity to detect OOD examples from the discrepancy in the predictions of the different networks. In a recent benchmark targeting models under dataset shift, MC-dropout performed relatively weak, while deep ensembles performed best [23].

Another approach to uncertainty modeling uses a Bayesian framework. Here, Bayesian neural networks provide a predictive distribution over possible outcomes where the mean indicates the most likely class and the variance indicates the model's uncertainty. Hence, a BNN can be seen as an infinite ensemble of networks whose weights are drawn from the learned weight distribution [24]. In theory, Bayesian neural networks require an intractable marginalization over the weight space. To this end, the marginalization can be circumvented by approximating the intractable weight posterior by a simpler variational distribution [25, 26] via stochastic variational inference (SVI). Building upon previous work, *Bayes by backprop* provides a tractable objective function to train BNNs via gradient descent [24].

In this work, we compare a standard CNN with softmax statistics with the two most promising approaches, namely deep ensembles and BNNs trained with SVI.

III. Predictive Distributions from Neural Networks

This section describes how deep ensembles and Bayesian neural networks infer the predictive distribution. Let $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$ be a dataset with N training samples that consist of inputs $\boldsymbol{x}_i$ and class labels $\boldsymbol{y}_i$. The goal is to infer a predictive distribution $p(\boldsymbol{y}^*|\boldsymbol{x}^*)$ for a new input $\boldsymbol{x}^*$.

A. Deep Ensembles

Deep neural networks have a complex loss landscape. Hence, if the same network is trained with different random initializations and shuffling of training samples, the training likely converges for each network instance in a different local minimum. These instances form an ensemble of diverse experts [22]. The combination of these experts is a simple technique to improve the accuracy of point estimates, but also to quantify uncertainty. To this end, the predictive distribution $p(\boldsymbol{y}^*|\boldsymbol{x}^*)$ is formed from the outputs of M experts.

B. Bayesian Neural Networks

BNNs differentiate from standard neural networks by learning a posterior distribution $p(\boldsymbol{w}|\mathcal{D})$ over the trainable network weights $\boldsymbol{w}$. The predictive distribution is inferred by marginalizing over all possible weights

$$p(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D}) = \int p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{w}) p(\boldsymbol{w}|\mathcal{D}) \, \mathrm{d}\boldsymbol{w}. \quad (1)$$

However, integration over the weight space is computationally prohibitive. Hence, Blundell *et al.* propose stochastic variational inference to approximate the intractable weight posterior $p(\boldsymbol{w}|\mathcal{D})$ by a simpler variational distribution $q_{\boldsymbol{\theta}}(\boldsymbol{w})$ with parameters $\boldsymbol{\theta}$ [24]. The optimal parameters $\hat{\boldsymbol{\theta}}$ are obtained by minimizing the Kullback-Leibler (KL) divergence between the approximation and the intractable weight posterior, *i.e.*,

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \arg\min_{\boldsymbol{\theta}} \mathrm{KL}\left[q_{\boldsymbol{\theta}}(\boldsymbol{w})||p(\boldsymbol{w}|\mathcal{D})\right] \\ &= \arg\min_{\boldsymbol{\theta}} \mathrm{KL}\left[q_{\boldsymbol{\theta}}(\boldsymbol{w})||p(\boldsymbol{w})\right] - \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{w})}\left[\log p(\mathcal{D}|\boldsymbol{w})\right] \ .\end{aligned} \tag{2}$$

The resulting cost function in the last line of Eqn. 2 is the negative *evidence lower bound* (ELBO) [24], and we additionally multiply the KL term by a hyper-parameter λ to weight its impact. In this cost function, the negative log-likelihood is analogous to traditional neural networks, while the KL term encourages similarity of the variational posterior to a weight prior $p(\boldsymbol{w})$. While the KL term can be computed in closed form, computation of the expectation requires another approximation, for which we refer to the original work by Blundell *et al.* [24].

We empirically found that focusing on hard examples improves the performance. Since the negative log-likelihood after softmax activation is equivalent to the multi-class cross-entropy loss [27], we replace the negative log-likelihood in Eqn. 2 by the focal loss [28]. The focal loss is defined as

$$\mathrm{FL}(\boldsymbol{x}_i, \boldsymbol{y}_i) = -\sum_{k=1}^{K} \left(1 - p(y_{ik}|\boldsymbol{x}_i)\right)^{\gamma} y_{ik} \log\left(p(y_{ik}|\boldsymbol{x}_i)\right) \ , \tag{3}$$

where K is the number of classes and $\gamma > 0$ emphasizes hard examples ($\gamma = 0$ is the original cross-entropy loss).

After training, we denote the learned parameters as $\hat{\boldsymbol{\theta}}$. Then, the predictive distribution from Eqn. 1 becomes

$$q_{\hat{\boldsymbol{\theta}}}(\boldsymbol{y}^*|\boldsymbol{x}^*) = \int p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{w}) q_{\hat{\boldsymbol{\theta}}}(\boldsymbol{w}) \, \mathrm{d}\boldsymbol{w} \ . \tag{4}$$

IV. Predictions and Uncertainty Quantification

The BNN prediction is the mean of the predictive distribution from Eqn. 4, which is evaluated via Monte Carlo sampling

$$\hat{q}_{\hat{\boldsymbol{\theta}}}(\boldsymbol{y}^*|\boldsymbol{x}^*) = \frac{1}{T}\sum_{t=1}^{T} p(\boldsymbol{y}^*|\boldsymbol{x}^*, \hat{\boldsymbol{w}}_t), \ \ \hat{\boldsymbol{w}}_t \sim q_{\hat{\boldsymbol{\theta}}}(\boldsymbol{w}) \ . \tag{5}$$

Here, T Monte Carlo samples draw weights $\hat{\boldsymbol{w}}_t$ from the trained variational distribution $q_{\hat{\boldsymbol{\theta}}}(\boldsymbol{w})$.

Similarly, the prediction of a deep ensemble is obtained by averaging the predictions from M ensemble experts,

$$p(\boldsymbol{y}^*|\boldsymbol{x}^*) = \frac{1}{M}\sum_{m=1}^{M} p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{w}_m) \ , \tag{6}$$

where $\boldsymbol{w}_m$ denotes the trained parameters of the m-th network.

Additionally, an uncertainty measure (or conversely, a confidence) is desirable to predict failure cases. Standard CNNs only offer the softmax probability distribution as uncertainty measure. The predictive distribution from deep ensembles and BNNs provides more possibilities to reason about uncertainty. We examine the two most common ways, namely Shannon entropy and epistemic uncertainty.

TABLE I
Known and unknown camera models from the Dresden image database (DIDB) and Kaggle dataset in this work.

Dataset	**Camera models**
Known DIDB	Canon Ixus70, Canon Ixus55, Nikon D200, Nikon D70, Sony DSC-H50
Unknown DIDB	Agfa DC-830i, Samsung L74wide, Sony DSC-W170, Canon PowerShotA640, Nikon CoolPixS710
Unknown Kaggle	Sony NEX-7, Motorola Moto X, Motorola Nexus 6, Apple iPhone 4s, Apple iPhone 6, HTC One M7, Samsung Galaxy S4, Samsung Galaxy Note 3 Motorola DROID MAXX, LG Nexus 5x

a) Softmax uncertainty: The softmax activations of the last layer of a classification network tend to show lower activations for incorrect and OOD samples [18]. Hence, the maximum activation can be used as confidence score, *i.e.*,

$$c = \max_k \ p(y_k^*|\boldsymbol{x}^*, \boldsymbol{w}) \quad . \tag{7}$$

b) Predictive entropy: Gal [29] proposed to quantify uncertainty on the predictive distribution via the Shannon entropy $H\left[p(\boldsymbol{y}^*|\boldsymbol{x}^*)\right]$. Reusing the BNN notation from Eqn. 5, the variational entropy $H\left[q_{\hat{\boldsymbol{\theta}}}(\boldsymbol{y}^*|\boldsymbol{x}^*)\right]$ is increasingly accurately approximated for increasing T as

$$\begin{aligned}&H\left[q_{\hat{\boldsymbol{\theta}}}(\boldsymbol{y}^*|\boldsymbol{x}^*)\right] \approx \\ &\quad -\sum_{k=1}^{K} \hat{q}_{\hat{\boldsymbol{\theta}}}(\boldsymbol{y}^* = \boldsymbol{e}_k|\boldsymbol{x}^*) \log \hat{q}_{\hat{\boldsymbol{\theta}}}(\boldsymbol{y}^* = \boldsymbol{e}_k|\boldsymbol{x}^*),\end{aligned} \tag{8}$$

where $\boldsymbol{y}^i \in \{\boldsymbol{e}_i\}_{i=1}^K$, and $\boldsymbol{e}_k$ is a K-dimensional unit vector.

c) Epistemic uncertainty: The variance of the predictive distribution in Eqn. 4 captures the overall predictive uncertainty, which can be further decomposed into *aleatoric* and *epistemic* uncertainty [30]. Epistemic uncertainty reflects the uncertainty in model parameters and is reducible with more training data. Aleatoric uncertainty is irreducible and captures inherent statistical noise from the data. Hence, epistemic uncertainty can be used for OOD detection. We calculate the epistemic uncertainty according to Kwon *et al.* [31] as

$$\mathrm{Var}_{\mathrm{ep}}\left[\boldsymbol{y}^*\right] \ = \ \frac{1}{T}\sum_{t=1}^{T}\left[p(\boldsymbol{y}^*|\boldsymbol{x}^*, \hat{\boldsymbol{w}}_t) - \hat{q}_{\hat{\boldsymbol{\theta}}}(\boldsymbol{y}^*|\boldsymbol{x}^*)\right]^{\otimes 2}, \tag{9}$$

where $^{\otimes 2}$ denotes the outer product, *i.e.*, $v^{\otimes 2} = vv^T$, and we take the sum over the diagonal elements as scalar uncertainty estimate.

V. Evaluation

We first state the experimental setup. Then, we evaluate the classification accuracy, compare the detection of OOD samples for unseen cameras and unseen processing, and investigate the impact of ensemble size and the number of Monte Carlo draws.

A. Experimental Setup

Data Preparation: The in-distribution data is composed of 2 453 images from five camera models from the Dresden Image Database (DIDB) [7]. The number of images per camera model ranges between 224 and 752. As OOD data, we choose 2 453 images from five different camera models in the DIDB, and additionally 2 750 images from ten smartphone camera models of the Kaggle-hosted IEEE Signal Processing Camera Model Identification Challenge dataset (Kaggle) [32]. These three datasets are denoted as *Known DIDB*, *Unknown DIDB* and *Unknown Kaggle*. The camera models are listed in Tab. I.

We extract 25 non-overlapping 256×256 pixel patches from the green channel of the central region of each full-resolution image. *Known DIDB* patches are split into 49 075/6 125/6 125 patches for training, validation, and testing, such that patches from a single image only occur in one of these three sets. From *Unknown DIDB* and *Unkown Kaggle*, we randomly choose 576 and 256 patches from each camera model.

To evaluate the detection of unknown post-processing, we form three additional datasets by adding either JPEG compression, Gaussian blur, or additive white Gaussian noise (AWGN) to in-distribution test set patches (Known DIDB), resulting in 6 125 OOD images per distortion (see specific distortion parameters below). As a final pre-processing step, all pixel intensities are scaled to the range $[0, 1]$.

Network Architectures: All networks use the constrained convolutional architecture by Bayar and Stamm [2]. The baseline CNN and deep ensemble are trained with the focal loss in Eqn. 3 with $\gamma = 2$ to compensate for the imbalanced number of samples per class. The CNN weights are initialized from a uniform distribution with Glorot limits. Also the BNN uses the focal loss for the log-likelihood term with balancing factor $\lambda = 0.25/N$. To convert the Bayar and Stamm CNN to a BNN, we replace convolutional and fully-connected layers with Flipout layers [33] from the TensorFlow Probability framework [34]. Also, instead of ReLU activation followed by batch normalization, we use the SELU activation [35]. As prior distribution we assume a multivariate standard normal, and initialize the variational posterior as diagonal Gaussian distribution as suggested by [24]. Hence, the BNN has twice as many trainable parameters as the baseline CNN.

Training Parameters: We use the Adam optimizer with learning rate $lr = 10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-7}$. The baseline CNN and each ensemble member is trained for 100 epochs. whereas the BNN is trained for 300 epochs due to the larger number of parameters. In both cases, we apply early stopping with a patience of 5 epochs.

Evaluation Protocol: To determine the classification accuracy, we use the point estimate of the baseline CNN, and the averaging in Eqns. 5 and 6 for the BNN and the deep ensemble, respectively.

The predicted uncertainties are evaluated by their ability to separate in-distribution data from out-of-distribution data. Following Hendrycks and Gimpel [18], the negative class consists of correct classifications on in-distribution data, and the positive class consists of out-of-distribution data (i.e., misclassifications on in-distribution data do not contribute to the evaluation of uncertainty). We report the receiver operating characteristic (ROC) curves and the area under the ROC curve (AUC). For the baseline CNN, we use the negative confidence of the most likely class as uncertainty (Eqn. 7). For the deep ensemble and the BNN, we evaluate both predictive entropy (Eqn. 8) and epistemic uncertainty (Eqn. 9).

B. In-distribution Classification Accuracy

Each model is trained on image patches from the *Known DIDB* dataset. The baseline CNN achieves an overall accuracy of 98.74%, and a per-class accuracy between 97.33% and 99.78%. Averaging the predictions from $M = 5$ CNNs results in an accuracy of 99.40% for the deep ensemble. A BNN with $T = 50$ Monte-Carlo draws achieves a slightly lower (but overall still quite high) accuracy of 97.42%.

C. Detection of Out-of-Distribution Images

Table II compares the softmax confidence from a single CNN to uncertainty estimates from a deep ensemble and a BNN in detecting five different types of OOD data. We repeated all experiments five times and report the mean and standard deviation. For all datasets, softmax statistics are outperformed by both the deep ensemble and the BNN. The deep ensemble performs better at detecting unknown cameras, while the BNN performs better at detecting unseen post-processing, as reported below.

a) Unseen Camera Models: The left two columns in Tab. II show the AUC (scaled by factor 100) to detect unknown camera models from the DIDB and Kaggle datasets. On both datasets, the deep ensemble outperforms the BNN, which in turn outperforms the baseline CNN. In these experiments, the type of uncertainty calculation does not show a large difference for the deep ensemble; the BNN performs better with epistemic uncertainty.

b) Unseen Post-Processing: The last three columns of Tab. II show different forms of post-processing, namely the detection of JPEG compression with quality 70, Gaussian blur with $\sigma_{\text{blur}} = 1.1$, and additive white Gaussian noise (AWGN) with $\sigma_{\text{noise}} = 0.1$. These levels were chosen based on [36], but we used a smaller σ_{noise} for more realistic distortions. On AWGN, both BNN and deep ensemble perform excellently, with a 2% better performance for the deep ensemble. However, for JPEG-compressed and blurred images, the BNN outperforms the deep ensemble by a significant margins of 12% and 14%. We take this as indication that BNNs better detect unseen post-processing. For both networks, the predictive entropy is better on JPEG post-processing, while epistemic uncertainty is better on blur. For AWGN, both variants are about equal.

Across all experiments, the BNN also exhibits a consistently low standard deviation for the five repetions of each experiment, which makes the performance well predictable. Conversely, the standard deviations of both the baseline CNN and the deep ensemble are much higher. The deep ensembles typically exhibit a standard deviation beyond 1% (except for AWGN), and up to 9% for the deep ensembles on blurring.

TABLE II

Detection of out-of-distribution samples from unknown camera models and unseen post-processing in terms of AUC (scaled by factor 100) with standard deviation over five runs. Both deep ensemble and BNN outperform the softmax statistics. The deep ensemble better detects unknown cameras, while the BNN better detects unseen post-processing.

Architecture	Method	Unknown cameras		Unseen post-processing		
		Unknown DIDB	Unknown Kaggle	JPEG, QF=70	Blur, $\sigma_{\text{blur}} = 1.1$	AWGN, $\sigma_{\text{noise}} = 0.1$
Baseline CNN	Softmax Statistics	63.87(±2.25)	78.85(±1.96)	73.05(±4.45)	59.51(±20.38)	55.87(±34.69)
Deep ensemble, M = 5	Predictive Entropy	72.00(±2.30)	**90.60(±1.42)**	71.33(±1.77)	73.59(±9.08)	99.03(±0.66)
Deep ensemble, M = 5	Epistemic Uncertainty	**72.11(±2.21)**	90.26(±1.38)	69.83(±1.81)	74.78(±8.75)	**99.75(±0.21)**
BNN, T = 50	Predictive Entropy	68.43(±0.11)	86.15(±0.05)	**83.81(±0.08)**	94.39(±0.02)	97.63(±0.04)
BNN, T = 50	Epistemic Uncertainty	72.07(±0.21)	88.40(±0.10)	77.80(±0.20)	**99.00(±0.03)**	97.79(±0.12)

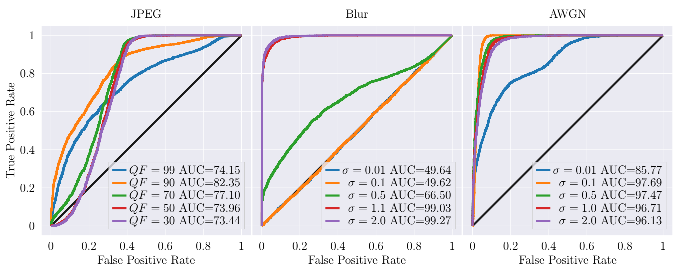

Fig. 1. Detection of post-processed OOD images (JPEG, Gaussian blur, additive white Gaussian noise) based on the BNN epistemic uncertainty with $T = 50$. For Gaussian blur and AWGN, the AUC (scaled by 100) increases with the amount of distortion.

We also report the detection performance for different post-processing strengths for the better performing BNN using epistemic uncertainty. In this experiment, we choose the JPEG quality factors $QF \in \{50, 60, 70, 80, 90, 99\}$, blur factors $\sigma_{\text{blur}} \in \{0.01, 0.1, 0.5, 1.1, 2.0\}$, and AWGN with $\sigma_{\text{noise}} \in \{0.01, 0.1, 0.5, 1.0, 2.0\}$. The resulting ROC curves are shown in Fig. 1. For Gaussian blur, the AUC increases with more distortion, as expected. For AWGN, the AUC peaks at $\sigma_{\text{noise}} = 0.1$, before starting to decline. Nevertheless, the detection performance remains above 96% for $\sigma_{\text{noise}} \geq 0.1$.

For JPEG-compressed images, we observe an unexpected behavior in the ROC curves. The AUC increases for $QF = 90$, but worsens with increasing compression strength. We believe that this behavior is caused by the fact the training images were already JPEG-compressed. In these cases, the BNN predicts uncertainties within the range of in-distribution images, though higher than most in-distribution images. A further analysis will be part of future work.

D. Impact of Monte Carlo Draws and Ensemble Size

As the BNN's posterior predictive distribution is approximated via Monte Carlo sampling, the number of samples T trades off precision of the approximation against computational complexity. In Fig. 2, we investigate the quality of the BNN's epistemic uncertainty metric with $T \in \{3, 5, 10, 20, 30, 50, 100, 200\}$. As expected, we observe that the detection of OOD samples benefits from more Monte Carlo draws. Only for JPEG post-processing, the AUC decreases slightly for higher T. In general, the improvement gains show a logarithmic behavior. Therefore, we used $T = 50$ as a performance trade-off.

TABLE III

Scaling a deep ensemble to $M = 50$ members aids the detection of out-of-distribution images from unknown camera models.

Dataset	Method	AUC
Unknown DIDB	Predictive entropy	74.41(±0.82)
	Epistemic uncertainty	75.81(±0.97)
Unknown Kaggle	Predictive entropy	94.16(±0.39)
	Epistemic uncertainty	95.08(±0.28)
JPEG, QF=70	Predictive entropy	69.06(±1.37)
	Epistemic uncertainty	67.09(±1.60)
Blur, $\sigma_{\text{blur}} = 1.1$	Predictive entropy	67.99(±2.11)
	Epistemic uncertainty	69.23(±2.44)
AWGN, $\sigma_{\text{noise}} = 0.1$	Predictive entropy	99.77(±0.07)
	Epistemic uncertainty	99.96(±0.05)

Out of curiousity, we also took the computational effort to train 100 CNNs for a deep ensemble, out of which $M = 50$ were randomly selected. We repeat this process five times to also report the standard deviation of the predictions. The results in Tab. III show that the detection of unknown cameras and AWGN benefits from more ensemble members, clearly outperforming the BNN with $T = 50$. However, results on JPEG compression and blur do not improve. However, while increasing the number of Monte Carlo samples for the BNN is relatively cheap, the training of a large number of CNNs to increase the ensemble is computationally extremely expensive.

VI. Conclusion

We investigated three deep learning paradigms to anticipate a training-test mismatches and therefore prevent false accusations on the task of camera model identification. Interpreting a CNN's softmax output as confidence yields inferior detection of OOD samples and can be affected by high variance. Both deep ensemble and BNN outperform the baseline CNN by large margins. The deep ensemble shows superior performance for detecting unknown camera models, while the BNN outperforms the deep ensemble for unseen post-processing. Additionally, increasing the quality of the uncertainty of a BNN by more samples is relatively cheap, while scaling a deep ensemble requires significant computational effort. In future work, we plan to train on a larger set of known cameras and evaluate the detection of more types of out-of-distribution data.

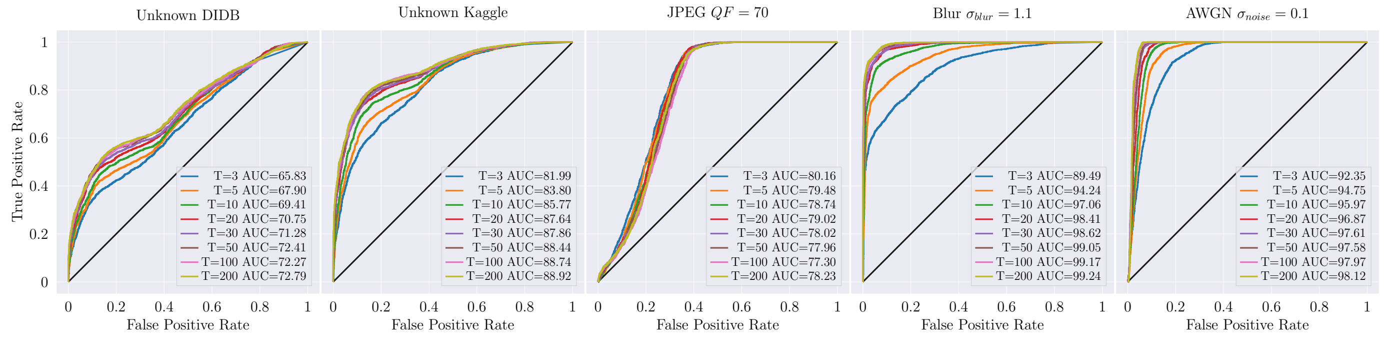

Fig. 2. Evaluation of the BNN's epistemic uncertainty w.r.t. the number of Monte Carlo draws T at test time. Except for the JPEG-compressed dataset, the AUC (scaled by factor 100) improves with increasing number of Monte Carlo samples for unseen camera models and unseen post-processing.

References

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

[2] B. Bayar and M. C. Stamm, "Constrained convolutional neural networks: A new approach towards general purpose image manipulation detection," *IEEE Trans. Inf. Forensics and Security*, vol. 13, pp. 2691–2706, 2018.

[3] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Niessner, "Faceforensics++: Learning to detect manipulated facial images," in *IEEE/CVF Int. Conf. Computer Vision*, Oct. 2019, pp. 1–11.

[4] S. Mandelli, N. Bonettini, P. Bestagini, and S. Tubaro, "Training CNNs in presence of JPEG compression: Multimedia forensics vs computer vision," in *IEEE Int. Workshop Inf. Forensics Security*, Dec. 2020.

[5] B. Diallo, T. Urruty, P. Bourdon, and C. Fernandez-Maloigne, "Improving robustness of image tampering detection for compression," in *Int. Conf. Multimedia Modeling*, Jan. 2019, pp. 387–398.

[6] D. Cozzolino, J. Thies, A. Rössler, C. Riess, M. Nießner, and L. Verdoliva, "ForensicTransfer: Weakly-supervised Domain Adaptation for Forgery Detection," *arXiv e-prints*, p. arXiv:1812.02510, Dec. 2018.

[7] T. Gloe and R. Böhme, "The Dresden image database for benchmarking digital image forensics," in *ACM Symp. Applied Computing*, New York, NY, USA, 2010, pp. 1584–1590.

[8] B. Hadwiger and C. Riess, "The Forchheim image database for camera identification in the wild," in *Multimedia Forensics in the Wild*, 2021.

[9] M. Huh, A. Liu, A. Owens, and A. A. Efros, "Fighting fake news: Image splice detection via learned self-consistency," in *Eur. Conf. Computer Vision*, 2018, pp. 101–117.

[10] A. Maier, B. Lorch, and C. Riess, "Toward reliable models for authenticating multimedia content: Detecting resampling artifacts with Bayesian neural networks," in *IEEE Int. Conf. Image Process.*, 2020, pp. 1251–1255.

[11] B. Lorch, A. Maier, and C. Riess, "Reliable JPEG forensics via model uncertainty," in *IEEE Int. Workshop Inf. Forensics Security*, Dec. 2020.

[12] T. Filler, J. Fridrich, and M. Goljan, "Using sensor pattern noise for camera model identification," in *IEEE Int. Conf. Image Process.*, 2008, pp. 1296–1299.

[13] S. Bayram, H. Sencar, N. Memon, and I. Avcibas, "Source camera identification based on CFA interpolation," in *IEEE Int. Conf. Image Process.*, 2005.

[14] K. S. Choi, E. Y. Lam, and K. K. Y. Wong, "Source camera identification by JPEG compression statistics for image forensics," in *IEEE Region Conf. TENCON*, 2006.

[15] A. Tuama, F. Comby, and M. Chaumont, "Camera model identification with the use of deep convolutional neural networks," in *IEEE Int. Workshop Inf. Forensics and Security*, 2016.

[16] B. Bayar and M. C. Stamm, "Towards open set camera model identification using a deep learning framework," in *IEEE Int. Conf. Acoustics, Speech and Signal Process.*, 2018, pp. 2007–2011.

[17] P. R. Mendes Júnior, L. Bondi, P. Bestagini, S. Tubaro, and A. Rocha, "An in-depth study on open-set camera model identification," *IEEE Access*, vol. 7, pp. 180 713–180 726, 2019.

[18] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," in *Int. Conf. Learning Representations*, 2017.

[19] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *34th Int. Conf. Machine Learning*, vol. 70, 2017, pp. 1321–1330.

[20] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," in *Int. Conf. Learning Representations*, vol. 70, 2018.

[21] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *33rd Int. Conf. Machine Learning*, vol. 48, 2016, pp. 1050–1059.

[22] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6405–6416.

[23] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek, "Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift," in *Advances in Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 13 991–14 002.

[24] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *32nd Int. Conf. Machine Learning*, vol. 37, 2015, pp. 1613–1622.

[25] G. E. Hinton and D. van Camp, "Keeping the neural networks simple by minimizing the description length of the weights," in *Sixth Annual Conf. Computational Learning Theory*, 1993, pp. 5–13.

[26] A. Graves, "Practical variational inference for neural networks," in *24th Int. Conf. Neural Inf. Process. Syst.*, 2011, pp. 2348–2356.

[27] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[28] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *IEEE Int. Conf. Computer Vision*, 2017, pp. 2999–3007.

[29] Y. Gal, "Uncertainty in deep learning," Ph.D. dissertation, University of Cambridge, 2016.

[30] A. Kendall and Y. Gal, "What uncertainties do we need in Bayesian deep learning for computer vision?" in *31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5580–5590.

[31] Y. Kwon, J.-H. Won, B. J. Kim, and M. C. Paik, "Uncertainty quantification using Bayesian neural networks in classification: Application to biomedical image segmentation," *Computational Statistics & Data Analysis*, vol. 142, 2020.

[32] IEEE's Signal Processing Society. (2017) IEEE's signal processing society camera model identification challenge. Visited on 2020-10-17. [Online]. Available: https://www.kaggle.com/c/sp-society-camera-model-identification/data

[33] Y. Wen, P. Vicol, J. Ba, D. Tran, and R. Grosse, "Flipout: Efficient pseudo-independent weight perturbations on mini-batches," in *Int. Conf. Learning Representations*, 2018.

[34] J. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. Hoffman, and R. A. Saurous, "Tensorflow distributions," *arXiv preprint*, 2017.

[35] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 972–981.

[36] B. Bayar and M. C. Stamm, "Design principles of convolutional neural networks for multimedia forensics," *Electronic Imaging*, no. 7, pp. 77–86, 2017.