

Content Authentication for Neural Imaging Pipelines: End-to-end Optimization of Photo Provenance in Complex Distribution Channels

Paweł Korus^{1,2} and Nasir Memon¹

New York University¹, AGH University of Science and Technology²

<http://kt.agh.edu.pl/~korus>

Abstract

Forensic analysis of digital photo provenance relies on intrinsic traces left in the photograph at the time of its acquisition. Such analysis becomes unreliable after heavy post-processing, such as down-sampling and re-compression applied upon distribution in the Web. This paper explores end-to-end optimization of the entire image acquisition and distribution workflow to facilitate reliable forensic analysis at the end of the distribution channel. We demonstrate that neural imaging pipelines can be trained to replace the internals of digital cameras, and jointly optimized for high-fidelity photo development and reliable provenance analysis. In our experiments, the proposed approach increased image manipulation detection accuracy from 45% to over 90%. The findings encourage further research towards building more reliable imaging pipelines with explicit provenance-guaranteeing properties.

1. Introduction

Ensuring integrity of digital images is one of the most challenging and important problems in multimedia communications. Photographs and videos are commonly used for documentation of important events, and as such, require efficient and reliable authentication protocols. Our current media acquisition and distribution workflows are built with entertainment in mind, and not only fail to provide explicit security features, but actually work against them. Image compression standards exploit heavy redundancy of visual signals to reduce communication payload, but optimize for human perception alone. Security extensions of popular standards lack in adoption [20].

Two general approaches to assurance and verification of digital image integrity include [24, 30, 32]: (1) pro-active protection methods based on digital signatures or watermarking; (2) passive forensic analysis

which exploits inherent statistical properties resulting from the photo acquisition pipeline. While the former provides superior performance and allows for advanced protection features (like precise tampering localization [38], or reconstruction of tampered content [25]), it failed to gain widespread adoption due to the necessity to generate protected versions of the photographs, and the lack of incentives for camera vendors to modify camera design to integrate such features [4].

Passive forensics, on the other hand, relies on our knowledge of the photo acquisition pipeline, and statistical artifacts introduced by its successive steps. While this approach is well suited for potentially analyzing any digital photograph, it often falls short due to the complex nature of image post-processing and distribution channels. Digital images are not only heavily compressed, but also enhanced or even manipulated before, during or after dissemination. Popular images have many online incarnations, and tracing their distribution and evolution has spawned a new field of image phylogeny [13, 14] which relies on visual differences between multiple images to infer their relationships and editing history. However, phylogeny does not provide any tools to reason about the authenticity or history of individual images. Hence, reliable authentication of real-world online images remains untractable [39].

At the moment, forensic analysis often yields useful results in near-acquisition scenarios. Analysis of native images straight from the camera is more reliable, and even seemingly benign implementation details - like the rounding operators used in the camera's image signal processor [1] - can provide useful clues. Most forensic traces quickly become unreliable as the image undergoes further post-processing. One of the most reliable tools at our disposal involves analysis of the imaging sensor's artifacts (the photo response non-uniformity pattern) which can be used both for source attribution and content authentication problems [10].

In the near future, rapid progress in computational imaging will challenge digital image forensics even in

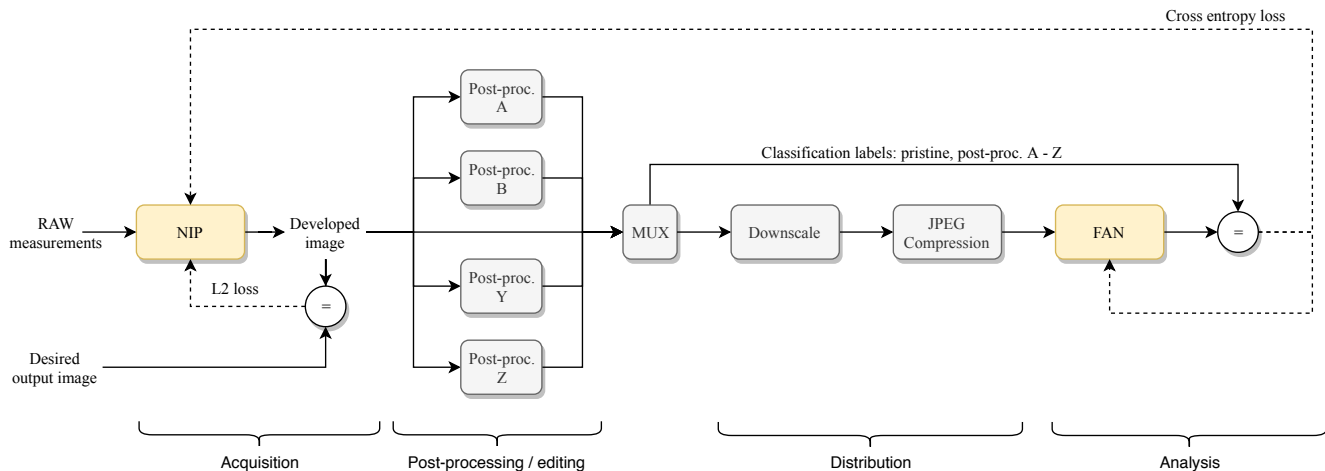


Figure 1. Optimization of the image acquisition and distribution channel to facilitate photo provenance analysis. The neural imaging pipeline (NIP) is trained to develop images that both resemble the desired target images, but also retain meaningful forensic clues at the end of complex distribution channels.

near-acquisition authentication. In the pursuit of better image quality and convenience, digital cameras and smartphones employ sophisticated post-processing directly in the camera, and soon few photographs will resemble the original images captured by the sensor(s). Adoption of machine learning has recently challenged many long-standing limitations of digital photography [9]; (1) high-quality low-light photography [9]; (2) single-shot HDR with overexposed content recovery [15]; (3) practical high-quality digital zoom from multiple shots [37]; (4) quality enhancement of smartphone-captured images with weak supervision from DSLR photos [18].

These remarkable results demonstrate tangible benefits of replacing the entire acquisition pipeline with neural networks. As a result, it will be necessary to investigate the impact of the emerging *neural imaging pipelines* on existing forensics protocols. While important, such evaluation can be seen rather as damage assessment and control and not as a solution for the future. We believe it is imperative to consider novel possibilities for security-oriented design of our cameras and multimedia dissemination channels.

In this paper, we propose to optimize neural imaging pipelines to improve photo provenance in complex distribution channels. We exploit end-to-end optimization of the entire photo acquisition and distribution channel to ensure that reliable forensics decisions can be made even after complex post-processing, where classical forensics fails (Fig. 1). We believe that imminent revolution in camera design creates a unique opportunity to address some of the long-standing limitations of the current technology. While adoption of digital watermarking in image authentication was lim-

ited by the necessity to modify camera hardware, our approach exploits the flexibility of neural networks to learn relevant integrity-preserving features within the expressive power of the model. We believe that with solid security-oriented understanding of neural imaging pipelines, and with the rare opportunity of replacing the well-established and security-oblivious pipeline, we can significantly improve digital image authentication capabilities.

We aim to inspire discussion about novel camera designs that could improve photo provenance analysis capabilities. We demonstrate that it is possible to optimize an imaging pipeline to significantly improve detection of photo manipulation at the end of a complex real-world distribution channel, where state-of-the-art deep-learning techniques fail. The main contributions of our work include:

1. The first end-to-end optimization of the imaging pipeline with explicit photo provenance objectives;
2. The first security-oriented discussion of neural imaging pipelines and the inherent trade-offs;
3. Significant improvement of forensic analysis performance in challenging, heavily post-processed conditions;
4. A neural model of the entire photo acquisition and distribution channel with a fully differentiable approximation of the JPEG codec.

To facilitate further research in this direction, and enable reproduction of our results, our neural imaging toolbox is available at <https://github.com/pkorus/neural-imaging>.

2. Related Work

Trends in Pipeline Design Learning individual steps of the imaging pipeline (e.g., demosaicing) has a long history [21] but regained momentum in the recent years thanks to adoption of deep learning. Naturally, the research focused on the most difficult operations, i.e., demosaicing [17, 23, 34, 35] and denoising [6, 26, 40]. Newly developed techniques delivered not only improved performance, but also additional features. Gharbi et al. proposed a convolutional neural network (CNN) trained for joint demosaicing and denoising [17]. A recent work by Syu et al. proposes to exploit CNNs for joint optimization of the color filter array and a corresponding demosaicing filter [34].

Optimization of digital camera design can go even further. The recently proposed L3 model by Jiang et al. replaces the entire imaging pipeline with a large collection of local linear filters [19]. The L3 model reproduces the entire photo development process, and aims to facilitate research and development efforts for non-standard camera designs. In the original paper, the model was used for learning imaging pipelines for RGBW (red-green-blue-white) and RGB-NIR (red-green-blue-near-infra-red) color filter arrays.

Replacing the entire imaging pipeline with a modern CNN can also overcome long-standing limitations of digital photography. Chen et al. trained a UNet model [31] to develop high-quality photographs in low-light conditions [9] by exposing it to paired examples of images taken with short and long exposure. The network learned to develop high-quality well-exposed color photographs from underexposed raw input, and yielded better performance than traditional image post-processing based on brightness adjustment and denoising. Eilertsen et al. also trained a UNet model to develop high-dynamic range images from a single shot [15]. The network not only learned to correctly perform tone mapping, but was also able to recover overexposed highlights. This significantly simplifies HDR photography by eliminating the need for bracketing and dealing with ghosting artifacts.

Trends in Forensics The current research in forensic image analysis focuses on two main directions: (1) learning deep features relevant to low-level forensic analysis for problems like manipulation detection [3, 41], identification of the social network of origin [2], camera model identification [11], or detection of artificially generated content [27]; (2) adoption of high-level vision to automate manual analysis that exposes physical inconsistencies, such as reflections [33, 36], or shadows [22]. To the best of our knowledge, there are currently no efforts to either assess the consequences

of the emerging neural imaging pipelines, or to exploit this opportunity to improve photo reliability.

3. End-to-end Optimization of Photo Provenance Analysis

Digital image forensics relies on intrinsic statistical artifacts introduced to photographs at the time of their acquisition. Such traces are later used for reasoning about the source, authenticity and processing history of individual photographs. The main problem is that contemporary media distribution channels employ heavy compression and post-processing which destroy the traces and inhibit forensic analysis.

The core of the proposed approach is to model the entire acquisition and distribution channel, and optimize the neural imaging pipeline (NIP) to facilitate photo provenance analysis after content distribution (Fig. 1). The analysis is performed by a forensic analysis network (FAN) which makes a decision about the authenticity/processing history of the analyzed photograph. In the presented example, the model is trained to perform manipulation detection, i.e., aims to classify input images as either coming straight from the camera, or as being affected by a certain class of post-processing. The distribution channel mimics the behavior of modern photo sharing services and social networks which habitually down-sample and re-compress the photographs. As will be demonstrated later, forensic analysis in such conditions is severely inhibited.

The parameters of the NIP are updated to guarantee both faithful representation of a desired color photograph (L_2 loss), and accurate decisions of forensics analysis at the end of the distribution channel (cross-entropy loss). Hence, the parameters of the NIP and FAN models are chosen as:

$$\theta_{\text{nip}}^* = \underset{\theta_{\text{nip}}}{\operatorname{argmin}} \sum_n \left(\|y_n - \text{nip}(x_n | \theta_{\text{nip}})\|_2 \right) \quad (1a)$$

$$+ \sum_c \log(\text{fan}_c(d_c(\text{nip}(x_n | \theta_{\text{nip}})) | \theta_{\text{fan}})) \quad (1b)$$

$$\theta_{\text{fan}}^* = \underset{\theta_{\text{fan}}}{\operatorname{argmin}} \sum_n \sum_c \log(\text{fan}_c(d_c(\text{nip}(x_n | \theta_{\text{nip}})) | \theta_{\text{fan}}))$$

where: $\theta_{\text{nip}/\text{fan}}$ are the parameters of the NIP and FAN networks, respectively; x_n are the raw sensor measurements for the n -th example patch; y_n is the corresponding target color image; $\text{nip}(x_n)$ is the color RGB image developed by NIP from x_n ; $d_c()$ denotes a color image patch processed by manipulation c ; $\text{fan}_c()$ is the probability that an image belongs to the c -th manipulation class, as estimated by the FAN model.

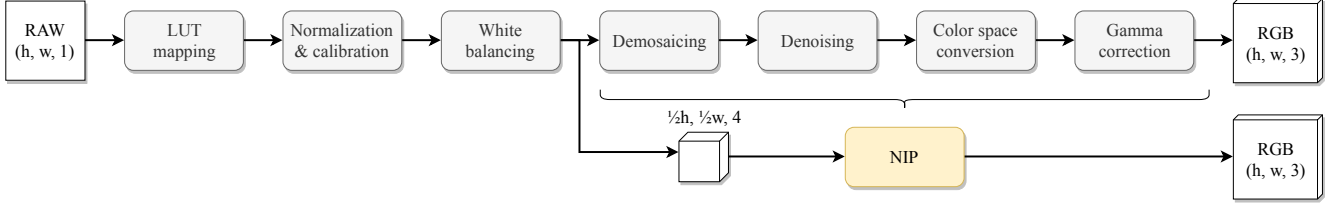


Figure 2. Adoption of a neural imaging pipeline to develop raw sensor measurements into color RGB images: (top) the standard imaging pipeline; (bottom) adoption of the NIP model.

3.1. The Neural Imaging Pipeline

We replace the entire imaging pipeline with a CNN which develops raw sensor measurements into color RGB images (Fig. 2). Before feeding the images to the network, we pre-process them by reversing the non-linear value mapping according to the camera’s LUT, subtracting black levels from the edge of the sensor, normalizing by sensor saturation values, and applying white-balancing according to shot settings. We also standardized the inputs by reshaping the tensors to have feature maps with successive measured color channels. This ensures a well-formed input of shape $(\frac{h}{2}, \frac{w}{2}, 4)$ with values normalized to $[0, 1]$. All of the remaining steps of the pipeline are replaced by a NIP. See Section 4.1 for details on the considered pipelines.

3.2. Approximation of JPEG Compression

To enable end-to-end optimization of the entire acquisition and distribution channel, we need to ensure that every processing step remains differentiable. In the considered scenario, the main problem is JPEG compression. We designed a *dJPEG* model which approximates the standard codec, and expresses its successive steps as matrix multiplications or convolution layers that can be implemented in TensorFlow (see supplementary materials for a detailed network definition):

- RGB to/from YCbCr color-space conversions are implemented as 1×1 convolutions.
- Isolation of 8×8 blocks for independent processing is implemented by a combination of *space-to-depth* and reshaping operations.
- Forward/backward 2D discrete cosine transforms are implemented by matrix multiplication according to DxD^T where x denotes a 8×8 input, and D denotes the transformation matrix.
- Division/multiplication of DCT coefficients by the corresponding quantization steps are implemented as element-wise operations with properly tiled and concatenated quantization matrices (for both the luminance and chrominance channels).

- The actual quantization is approximated by a continuous function $\rho(x)$ (see details below).

The key problem in making JPEG differentiable lies in the rounding of DCT coefficients. We considered two approximations (Fig. 3a). Initially, we used a Taylor series expansion, which can be made arbitrarily accurate by including more terms. Finally, we used a smoother, and simpler sinusoidal approximation obtained by matching its phase with the sawtooth function:

$$\rho(x) = x - \frac{\sin(2\pi x)}{2\pi} \quad (2)$$

We validated our *dJPEG* model by comparing produced images with a reference codec from *libJPEG*. The results are shown in Fig. 3bcd for a standard rounding operation, and the two approximations, respectively. We used 5 terms for the harmonic rounding. The developed module produces equivalent compression results with standard rounding, and a good approximation for its differentiable variants. Fig. 3e-h show a visual comparison of an example image patch, and its *libJPEG* and *dJPEG*-compressed counterparts.

In our distribution channel, we used quality level 50.

3.3. The Forensic Analysis Network

The forensic analysis network (FAN) is implemented as a CNN following the most recent recommendations on construction of neural networks for forensics analysis [3]. Bayar and Stamm proposed a new layer type, which constrains the learned filters to be valid residual filters [3]. Adoption of the layer helps ignore visual content and facilitates extraction of forensically-relevant low-level features. In summary, our network operates on $128 \times 128 \times 3$ patches in the RGB color space and includes (see supplement for full network definition):

- A constrained convolutions layer learning 5×5 residual filters and with no activation function.
- Four 5×5 convolutional layers with doubling number of output feature maps (starting from 32). The layers use leaky ReLU activation and are followed by 2×2 max pooling.

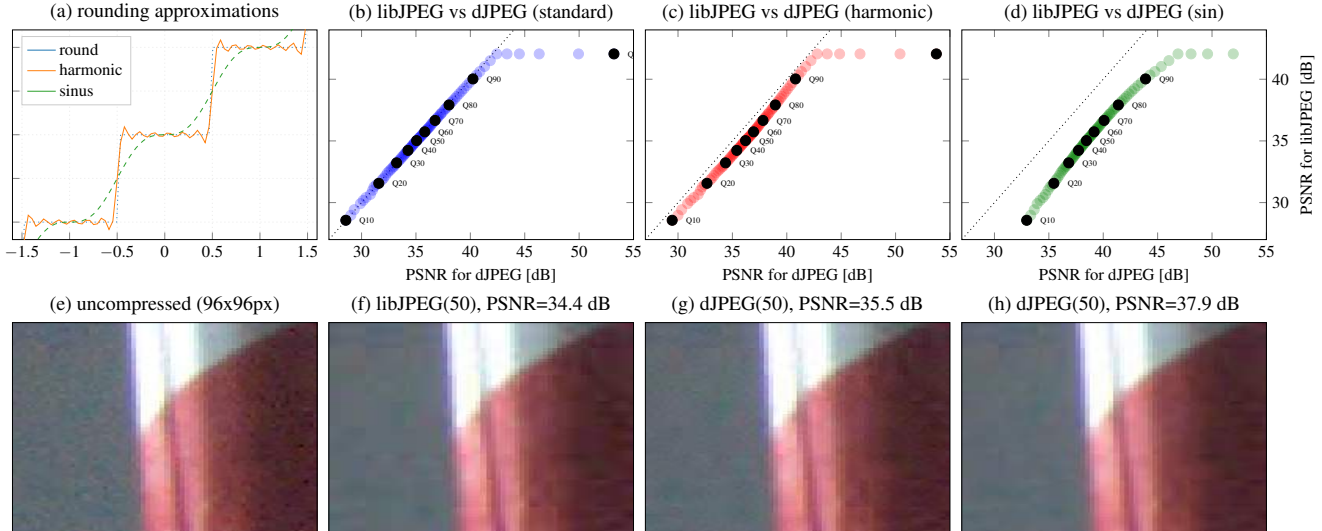


Figure 3. Implementation of JPEG compression as a fully differentiable *dJPEG* module: (a) continuous approximations of the rounding function; (b)-(d) validation of the *dJPEG* module against the standard *libJPEG* library with standard rounding, and the harmonic and sinusoidal approximations; (e) an example image patch; (f) standard JPEG compression with quality 50; (g)-(h) *dJPEG*-compressed patches with the harmonic and sinusoidal approximations.

- A 1×1 convolutional layer mapping 256 features into 256 features.
- A global average pooling layer reducing the number of features to 256.
- Two fully connected layers with 512 and 128 nodes activated by leaky ReLU.
- A fully connected layer with $N = 5$ output nodes and softmax activation.

The network has 1,341,990 parameters in total, and outputs probabilities of 5 possible processing histories (4 manipulation classes & straight from camera).

4. Experimental Evaluation

We started our evaluation by using several NIP models to reproduce the output of a standard imaging pipeline (Sec. 4.1). Then, we used the FAN model to detect popular image manipulations (Sec. 4.2). Initially, we validated that the models work correctly by using it without a distribution channel (Sec. 4.3). Finally, we performed extensive evaluation of the entire acquisition and distribution network (Sec. 4.4).

We collected a data-set with RAW images from 8 cameras (Table 1). The photographs come from two public (Raise [12] and MIT-5k [7]) and from one private data-set. For each camera, we randomly selected 150 images with landscape orientation. These images will later be divided into separate training/validation sets.

Table 1. Digital cameras used in our experiments

Camera	SR ¹	#Images ²	Source	Bayer
Canon EOS 5D	12	864 dng	MIT-5k	RGGB
Canon EOS 40D	10	313 dng	MIT-5k	RGGB
Nikon D5100	16	288 nef	Private	RGGB
Nikon D700	12	590 dng	MIT-5k	RGGB
Nikon D7000	16	>1k nef	Raise	RGGB
Nikon D750	24	312 nef	Private	RGGB
Nikon D810	36	205 nef	Private	RGGB
Nikon D90	12	>1k nef	Raise	GBRG

¹ Sensor Resolution in Megapixels [Mpx]

² RAW file formats: nef (Nikon); dng (generic, Adobe)

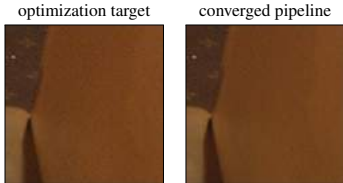
4.1. Neural Imaging Pipelines

We considered three NIP models with various complexity and design principles (Table 2): *INet* - a simple convolutional network with layers corresponding to successive steps of the standard pipeline; *UNet* - the well-known UNet architecture [31] adapted from [9]; *DNet* - adaptation of the model originally used for joint demosaicing and denoising [17]. Details of the networks' architectures are included in the supplement.

We trained a separate model for each camera. For training, we used 120 full-resolution images. In each iteration, we extracted random 128×128 patches and formed a batch with 20 examples (one patch per image). For validation, we used a fixed set of 512×512 px patches extracted from the remaining 30 images. The models were trained to reproduce the output of a stan-



(a) Improved demosaicing example (Canon EOS 40D)



(b) Improved denoising example (Nikon D5100)

Figure 4. Examples of serendipitous image quality improvements obtained by neural imaging pipelines: (a) better demosaicing performance; (b) better denoising.

Table 2. Considered neural imaging pipelines

	INet	UNet	DNet
# Parameters	321	7,760,268	493,976
PSNR	42.8	44.3	46.2
SSIM	0.989	0.990	0.995
Train. speed [it/s]	8.80	1.75	0.66
Train. time	17 - 26 min	2-4 h	12 - 22 h

standard imaging pipeline. We used our own implementation based on Python and *rawkit* [8] wrappers over *libRAW* [28]. Demosaicing was performed using an adaptive algorithm by Menon et al. [29].

All NIPs successfully reproduced target images with high fidelity. The resulting color photographs are visually indistinguishable from the targets. Objective fidelity measurements for the validation set are collected in Table 2 (average for all 8 cameras). Interestingly, the trained models often revealed better denoising and demosaicing performance, despite the lack of a denoising step in the simulated pipeline, and the lack of explicit optimization objectives (see Fig. 4).

Of all of the considered models, *INet* was the easiest to train - not only due to its simplicity, but also because it could be initialized with meaningful parameters that already produced valid results and only needed fine-tuning. We initialized the demosaicing filters with bilinear interpolation, color space conversion with a known multiplication matrix, and gamma correction with a toy model separately trained to reproduce this non-linearity. The *UNet* model was initialized randomly, but improved rapidly thanks to skip connections. The *DNet* model took the longest and for a long time had problems with faithful color rendering. The typical training times are reported in Ta-

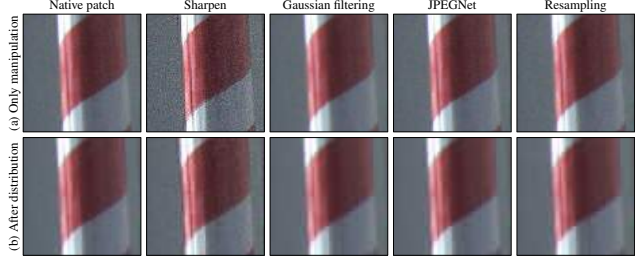


Figure 5. An example image patch with all of the considered manipulations: (a) only manipulation; (b) after the distribution channel (down-sampled and JPEG compressed).

ble 2. The measurements were collected on a Nvidia Tesla K80 GPU. The models were trained until the relative change of the average validation loss for the last 5 dropped below 10^{-4} . The maximum number of epochs was 50,000. For the *DNet* model we adjusted the stopping criterion to 10^{-5} since the training progress was slow, and often terminated prematurely with incorrect color rendering.

4.2. Image Manipulation

Our experiment mirrors the standard setup for image manipulation detection [3, 5, 16]. The FAN simply decides which manipulation class the input patch belongs to. This approximates identification of the last post-processing step, which is often used to mask traces of more invasive modifications.

We consider four mild post-processing operations: *sharpening* - implemented as an unsharp mask operator with the following kernel:

$$\frac{1}{6} \begin{bmatrix} -1 & -4 & -1 \\ -4 & 26 & -4 \\ -1 & -4 & -1 \end{bmatrix} \quad (3)$$

applied to the luminance channel in the HSV color space; *resampling* - implemented as successive 1:2 down-sampling and 2:1 up-sampling using bilinear interpolation; *Gaussian filtering* - implemented using a convolutional layer with a 5×5 filter and standard deviation 0.83; *JPG compression* - implemented using the dJPEG module with sinusoidal rounding approximation and quality level 80. Fig. 5 shows the post-processed variants of an example image patch: (a) just after manipulation; and (b) after the distribution channel (as seen by the FAN module).

4.3. FAN Model Validation

To validate our FAN model, we initially implemented a simple experiment, where analysis is performed just after image manipulation (no distribution channel distortion, as in [3]). We used the *UNet* model

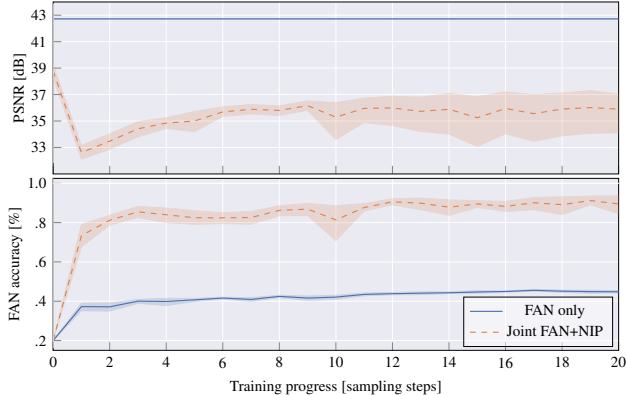


Figure 6. Typical progression of validation metrics (Nikon D90) for standalone FAN training (F) and joint optimization of FAN and NIP models (F+N).

to develop smaller image patches to guarantee the same size of the inputs for the FAN model ($128 \times 128 \times 3$ RGB images). In such conditions, the model works just as expected, and yields classification accuracy of 99% [3].

4.4. Imaging Pipeline Optimization

In this experiment, we perform forensic analysis at the end of the distribution channel. We consider two optimization modes: (F) only the FAN network is optimized given a fixed NIP model; (F+N) both the FAN and NIP models are optimized jointly. In both cases, the NIPs are pre-initialized with previously trained models (Section 4.1). The training was implemented with two separate Adam optimizers, where the first one updates the FAN (and in the F+N mode also the NIP) and the second one updates the NIP based on the image fidelity objective.

Similarly to previous experiments, we used 120 full-resolution images for training, and the remaining 30 images for validation. From training images, in each iteration we randomly extract new patches. The validation set is fixed at the beginning and includes 100 random patches per each image (3,000 patches in total) for classification accuracy assessment. To speed-up image fidelity evaluation, we used 2 patches per image (60 patches in total). In order to prevent over-representation of empty patches, we bias the selection by outward rejection of patches with pixel variance < 0.01 , and by 50% chance of keeping patches with variance < 0.02 . More diverse patches are always accepted.

Due to computational constraints, we performed the experiment for 4 cameras (Canon EOS 40D and EOS 5D, and Nikon D7000, and D90, see Table 1) and for the *INet* and *UNet* models only. (Based on preliminary experiments, we excluded the *DNet* model which rapidly lost and could not regain image representation

Table 3. Typical confusion matrices (Nikon D90). Entries ≈ 0 are not shown; entries $\gtrsim 3\%$ are marked with (*).

(a) standalone FAN optimization (*UNet*) \rightarrow 44.2%

True \ Predicted	nat.	sha.	gau.	jpg	res.
native	27	24	18	20	11
sharpen	*	93	3	*	*
gaussian	7	4	59	12	18
jpg	26	23	18	21	12
resample	14	7	38	20	21

(b) joint FAN+NIP optimization (*INet*) \rightarrow 55.2%

True \ Predicted	nat.	sha.	gau.	jpg	res.
native	41	16	22	18	4
sharpen	7	85	3	4	*
gaussian	18	8	54	10	10
jpg	41	16	21	19	4
resample	5	*	14	*	77

(c) joint FAN+NIP optimization (*UNet*) \rightarrow 94.0%

True \ Predicted	nat.	sha.	gau.	jpg	res.
native	90		*	9	
sharpen		100			
gaussian			97	*	*
jpg	13		3	84	
resample			*		99

fidelity.) We ran the optimization for 1,000 epochs, starting with a learning rate of 10^{-4} and systematically decreasing by 15% every 100 epochs. Each run was repeated 10 times. The typical progression of validation metrics for the *UNet* model (classification accuracy and distortion PSNR) is shown in Fig. 6 (sampled every 50 epochs). The distribution channel significantly disrupts forensic analysis, and the classification accuracy drops to $\approx 45\%$ for standalone FAN optimization (F). In particular, the FAN struggles to identify three low-pass filtering operations (Gaussian filtering, JPEG compression, and re-sampling; see confusion matrix in Tab. 3a), which bear strong visual resemblance at the end of the distribution channel (Fig. 5b). Optimization of the imaging pipeline significantly increases classification accuracy (over 90%) and makes the manipulation paths easily distinguishable (Tab. 3c). Most mistakes involve the *native* and *jpeg* compressed classes. Fig. 7 collects the obtained results for both the *UNet* and *INet* models. While *UNet* delivers consistent and significant benefits, *INet* is much simpler and yields only modest improvements in accuracy - up to $\approx 55\%$. It also lacked consistency and for one of the tested cameras yielded virtually no benefits.

The observed improvement in forensic accuracy comes at the cost of image distortion, and leads to artifacts in the developed photographs. In our experiments, photo development fidelity dropped to ≈ 36 dB

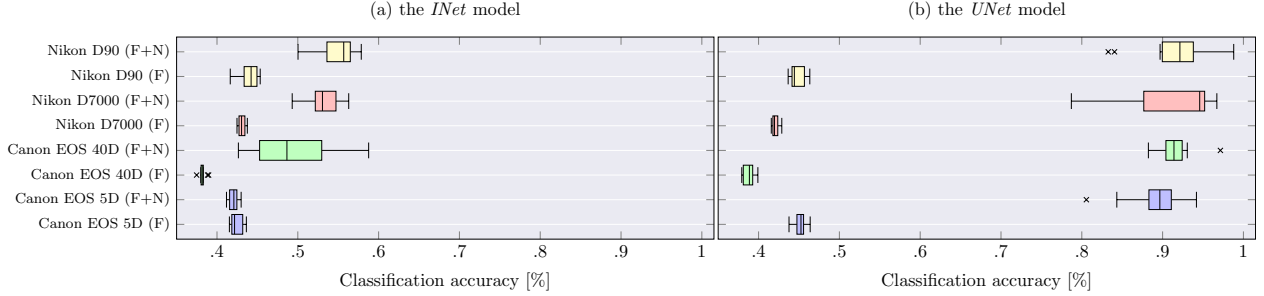


Figure 7. Validation accuracy for image manipulation detection after the distribution channel: (F) denotes standalone FAN training given a fixed NIP; (F+N) denotes joint optimization of the FAN and NIP models.

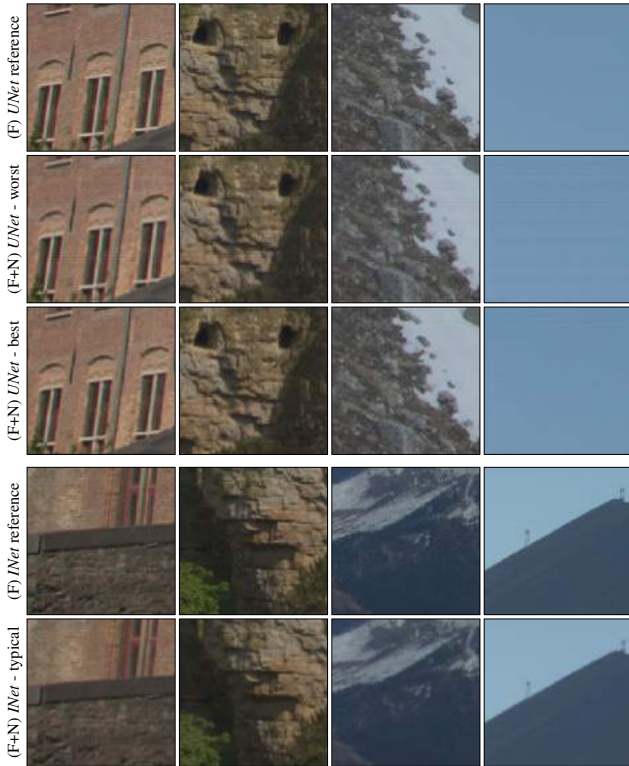


Figure 8. Example image patches developed with joint NIP and FAN optimization (F+N) for Nikon D90. The *UNet* examples show the best/worst artifacts. The *INet* examples show typical outputs.

(PSNR) / 0.96 (SSIM). Detailed results are collected in Tab. 4. The severity of the distortions varied with training repetitions. Qualitative illustration of the observed artifacts is shown in Fig. 8. The figure shows diverse image patches developed by several NIP variants (different training runs). The artifacts vary in severity from disturbing to imperceptible, and tend to be well masked by image content. *INet*'s artifacts tend to be less perceptible, which compensates for modest improvements in FAN's classification accuracy.

Table 4. Fidelity-accuracy trade-off for joint optimization.

Camera	PSNR [dB]	SSIM	Acc. [%]
<i>UNet</i> model			
D90	42.7 → 35.9	0.990 → 0.960	0.45 → 0.91
D7000	43.0 → 35.6	0.990 → 0.955	0.42 → 0.91
EOS 40D	42.8 → 36.1	0.990 → 0.962	0.39 → 0.92
EOS 5D	43.0 → 36.2	0.989 → 0.961	0.45 → 0.89
<i>INet</i> model			
D90	43.3 → 37.2	0.992 → 0.969	0.44 → 0.55
D7000	40.6 → 35.4	0.985 → 0.959	0.43 → 0.53
EOS 40D	41.6 → 33.1	0.985 → 0.934	0.38 → 0.50
EOS 5D	41.5 → 40.7	0.986 → 0.984	0.42 → 0.42

5. Discussion and Future Work

We replaced the photo acquisition pipeline with a CNN, and developed a fully-differentiable model of the entire acquisition and distribution workflow. This allows for joint optimization of photo analysis and acquisition. Our results show that it is possible to optimize the imaging pipeline to facilitate provenance analysis at the end of complex distribution channels. We observed significant improvements in manipulation detection accuracy w.r.t state-of-the-art classical forensics [3] (increase from $\approx 45\%$ to over 90%).

Competing optimization objectives lead to imaging artifacts which tend to be well masked in textured areas, but are currently too visible in flat regions. Severity of the artifacts varies between training runs and NIP architectures, and suggests that better configurations should be achievable by learning to control the fidelity-accuracy trade-off. Since final decisions are often taken by pooling predictions for many patches, we believe a good balance should be achievable even with relatively simple models. Further improvements may also be possible by exploring different NIP architectures, or explicitly modeling HVS characteristics. Future work should also assess generalization capabilities to other manipulations, and more complex forensic problems. It may also be interesting to optimize other components of the workflow, e.g., the lossy compression in the channel.

References

- [1] S. Agarwal and H. Farid. Photo forensics from jpeg dimples. In *Information Forensics and Security (WIFS), 2017 IEEE Workshop on*, pages 1–6. IEEE, 2017.
- [2] I. Amerini, T. Uricchio, and R. Caldelli. Tracing images back to their social network of origin: A cnn-based approach. In *IEEE Int. Workshop Information Forensics and Security*, 2017.
- [3] B. Bayar and M. Stamm. Constrained convolutional neural networks: A new approach towards general purpose image manipulation detection. *IEEE Tran. Information Forensics and Security*, 13(11):2691–2706, 2018.
- [4] P. Blythe and J. Fridrich. Secure digital camera. In *Digital Forensic Research Workshop*, pages 11–13, 2004.
- [5] M. Boroumand and J. Fridrich. Deep learning for detecting processing history of images. *Electronic Imaging*, 2018(7):1–9, 2018.
- [6] H. Burger, C. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with bm3d? In *Int. Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 2392–2399. IEEE, 2012.
- [7] V. Bychkovsky, S. Paris, E. Chan, and E. Durand. Learning photographic global tonal adjustment with a database of input/output image pairs. In *IEEE Conf. Computer Vision and Pattern Recognition*, 2011.
- [8] P. Cameron and S. Whited. Rawkit. <https://rawkit.readthedocs.io/en/latest/>. Visited Nov 2018.
- [9] C. Chen, Q. Chen, J. Xu, and V. Koltun. Learning to see in the dark. *arXiv*, 2018. arXiv:1805.01934.
- [10] M. Chen, J. Fridrich, M. Goljan, and J. Lukas. Determining image origin and integrity using sensor noise. *IEEE Trans. Inf. Forensics Security*, 3(1):74–90, 2008.
- [11] D. Cozzolino and L. Verdoliva. Noiseprint: a cnn-based camera model fingerprint. *arXiv preprint arXiv:1808.08396*, 2018.
- [12] D. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato. RAISE - a raw images dataset for digital image forensics. In *Proc. of ACM Multimedia Systems*, 2015.
- [13] Z. Dias, S. Goldenstein, and A. Rocha. Large-scale image phylogeny: Tracing image ancestral relationships. *IEEE Multimedia*, 20(3):58–70, 2013.
- [14] Z. Dias, S. Goldenstein, and A. Rocha. Toward image phylogeny forests: Automatically recovering semantically similar image relationships. *Forensic Science International*, 231(1):178–189, 2013.
- [15] G. Eilertsen, J. Kronander, G. Denes, R. Mantiuk, and J. Unger. HDR image reconstruction from a single exposure using deep CNNs. *ACM Tran. Graphics*, 36(6):1–15, nov 2017.
- [16] W. Fan, K. Wang, and F. Cayre. General-purpose image forensics using patch likelihood under image statistical models. In *Proc. of IEEE Int. Workshop on Inf. Forensics and Security*, 2015.
- [17] M. Gharbi, G. Chaurasia, S. Paris, and F. Durand. Deep joint demosaicking and denoising. *ACM Tran. Graphics*, 35(6):1–12, nov 2016.
- [18] A. Ignatov, N. Kobyshev, R. Timofte, K. Vanhoey, and L. Van Gool. Wespe: weakly supervised photo enhancer for digital cameras. *arXiv preprint arXiv:1709.01118*, 2017.
- [19] H. Jiang, Q. Tian, J. Farrell, and B. A. Wandell. Learning the image processing pipeline. *IEEE Tran. Image Processing*, 26(10):5032–5042, oct 2017.
- [20] Joint Photographic Experts Group. JPEG Privacy & Security. https://jpeg.org/items/20150910_privacy_security_summary.html. Visited Oct 2018.
- [21] O. Kapah and H. Z. Hel-Or. Demosaicking using artificial neural networks. In *Applications of Artificial Neural Networks in Image Processing V*, volume 3962, pages 112–121, 2000.
- [22] E. Kee, J. O’Brien, and H. Farid. Exposing photo manipulation from shading and shadows. *ACM Trans. Graph.*, 33(5), 2014.
- [23] F. Kokkinos and S. Lefkimmiatis. Deep image demosaicking using a cascade of convolutional residual denoising networks. *arXiv*, 2018. arXiv:1803.05215v4.
- [24] P. Korus. Digital image integrity—a survey of protection and verification techniques. *Digital Signal Processing*, 71:1–26, 2017.
- [25] P. Korus, J. Bialas, and A. Dziech. Towards practical self-embedding for JPEG-compressed digital images. *IEEE Trans. Multimedia*, 17(2):157–170, Feb 2015.
- [26] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila. Noise2noise: Learning image restoration without clean data. *arXiv preprint arXiv:1803.04189*, 2018.
- [27] H. Li, B. Li, S. Tan, and J. Huang. Detection of deep network generated images using disparities in color components. *arXiv preprint arXiv:1808.07276*, 2018.
- [28] LibRaw LLC. libRAW. <https://www.libraw.org/>. Visited Nov 2018.
- [29] D. Menon, S. Andriani, and G. Calvagno. Demosaicking with directional filtering and a posteriori decision. *IEEE Tran. Image Processing*, 16(1):132–141, 2007.
- [30] A. Piva. An overview on image forensics. *ISRN Signal Processing*, 2013, 2013. Article ID 496701.
- [31] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Int. Conf. Medical Image Computing and Computer-assisted Intervention*. Springer, 2015.
- [32] M. Stamm, M. Wu, and K. Liu. Information forensics: An overview of the first decade. *IEEE Access*, 1:167–200, 2013.
- [33] Z. H. Sun and A. Hoogs. Object insertion and removal in images with mirror reflection. In *IEEE Workshop on Information Forensics and Security*, pages 1–6, 2017.
- [34] N.-S. Syu, Y.-S. Chen, and Y.-Y. Chuang. Learning deep convolutional networks for demosaicking. *arXiv*, 2018. arXiv:1802.03769.
- [35] R. Tan, K. Zhang, W. Zuo, and L. Zhang. Color image demosaicking via deep residual learning. In *IEEE Int. Conf. Multimedia and Expo*, 2017.
- [36] E. Wengrowski, Z. Sun, and A. Hoogs. Reflection correspondence for exposing photograph manipulation. In *IEEE Int. Conf. Image Processing*, pages 4317–4321, 2017.
- [37] B. Wronski and P. Milanfar. See better and further with super res zoom on the pixel 3. <https://ai.googleblog.com/2018/10/see-better-and-further-with-super-res.html>. Visited in Oct 2018.
- [38] C. P. Yan and C. M. Pun. Multi-scale difference map fusion for tamper localization using binary ranking hashing. *IEEE Tran. Information Forensics and Security*, 12(9):2144–2158, 2017.
- [39] M. Zampoglou, S. Papadopoulos, and Y. Kompatsiaris. Large-scale evaluation of splicing localization algorithms for web images. *Multimedia Tools and Applications*, 76(4):4801–4834, 2017.
- [40] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Tran. Image Processing*, 26(7):3142–3155, 2017.
- [41] P. Zhou, X. Han, V. Morariu, and L. S. Davis. Learning rich features for image manipulation detection. In *IEEE Int. Conf. Computer Vision and Pattern Recognition*, pages 1053–1061, 2018.

Supplementary Materials for: "Content Authentication for Neural Imaging Pipelines: End-to-end Optimization of Photo Provenance in Complex Distribution Channels"

Paweł Korus^{1,2} and Nasir Memon¹

New York University¹, AGH University of Science and Technology²

<http://kt.agh.edu.pl/~korus>

1. Source Code

To facilitate further research in this direction, and enable reproduction of our results, our neural imaging toolbox is available at <https://github.com/pkorus/neural-imaging>.

2. Implementation Details

We implemented the entire acquisition and distribution pipeline in Python 3 and Tensorflow v. 1.11.0. In all experiments, we used the Adam optimizer with default settings. We proceeded in the following stages:

1. Training and validation of the NIPs.
2. Validation of the FAN for manipulation detection with no distribution channel between post-processing and forensic analysis.
3. Joint optimization of the FAN & NIP models with active distribution channel.

The NIP models trained in stage (1) were then used in (2) and (3) for NIP initialization to speed-up training.

2.1. NIP Architectures

We considered several NIP models with various levels of complexity. The simplest *INet* model was hand-crafted to replicate the standard imaging pipeline. The architecture (Tab. 2) is a simple sequential concatenation of convolutional layers. Each operation was initialized with meaningful values (see Tab. 2) which significantly sped up convergence. (We also experimented with random initialization, but this led to slower training and occasional problems with color fidelity.) The last two layers were initialized with parameters from a simple 2-layer network trained to approximate gamma correction for a scalar input (4 hidden nodes + one output node).

The *UNet* model (Tab. 3) was the most complex of the considered models, but it delivered fairly quick convergence due to skip connections [10]. We adapted the implementation from a recent work by Chen et al. [4]. The configuration of skip connections is shown in detail in Tab. 3. All convolutional layers produce tensors of the same spatial dimensions, eliminating the need for cropping.

We also experimented with two recent networks used for joint demosaicing and denoising (*DNet* - Tab. 4) [7], and for general-purpose image processing (*CNet*) [5]. Overall, the results were unsuccessful. The *DNet* model was able to learn a high-fidelity photo development process, but converged very slowly due to colorization and sharpness artifacts. The trained model proved to be rather fragile, and quickly deteriorated during joint NIP and FAN optimization¹. We were unable to obtain satisfactory photo development using the *CNet* model - the validation PSNR stopped improving around 25-27 dB.

All models were trained to replicate the output of a manually implemented imaging pipeline. (Detailed, per-camera validation performance measurements are shown in Tab. 1.) We used *rawkit* [3] wrappers over *libRAW* [8]. Demosaicing was performed using an adaptive algorithm by Menon et al. [9] from the *colour-demosaicing* package. We used learning rate of 10^{-4} and continued training until the average validation loss for the last 5 epochs changes by more than 10^{-4} . The maximal number of epochs was set to 50,000. Due to patch-based training, we did not perform any

¹In more recent experiments with explicit regularization of the impact of the NIP's L_2 loss, we were able to improve *DNet*'s performance. However, the model still remains more fragile and reaching classification accuracy comparable to *UNet* leads to excessive artifacts. In addition to noise-like artifacts, the model loses edge sharpness. With a visually acceptable distortion, the model yielded accuracy between 70-80%. More detailed results will be available in an extended version of this work.

Table 1. Detailed validation performance statistics for all cameras and all NIPs.

Camera	INet		UNet		DNet	
	PSNR ¹	SSIM	PSNR ¹	SSIM	PSNR ¹	SSIM
Canon EOS 40D	42.7	0.987	43.6	0.990	44.5	0.992
Canon EOS 5D	42.4	0.987	44.8	0.992	48.4	0.997
Nikon D5100	43.7	0.989	45.3	0.990	48.1	0.996
Nikon D700	44.7	0.993	45.6	0.994	47.2	0.997
Nikon D7000	42.3	0.989	44.4	0.992	44.9	0.994
Nikon D750	42.7	0.990	44.8	0.994	45.5	0.996
Nikon D810	39.6	0.984	41.9	0.991	43.6	0.995
Nikon D90	44.6	0.993	44.4	0.991	47.7	0.997

¹ PSNR values in [dB]

global post-processing (e.g., histogram stretching). In each epoch, we randomly selected patches from full-resolution images, and fed them to the network in batches of 20 examples. We used input examples of size $64 \times 64 \times 4$ (RGGB) which are developed into $128 \times 128 \times 3$ (RGB) color patches. The final networks can work on arbitrary inputs without any changes. Fig. 1 shows an example full-resolution (12 Mpx) image developed with the standard (ab) and the neural pipelines (cde).

2.2. JPEG Codec Approximation

The architecture of the dJPEG model is shown in Tab. 5. The network was hand-crafted to replicate the operation of the standard JPEG codec with no chrominance sub-sampling (the 4:4:4 mode). We used standard quantization matrices from the IJG codec [1]. See the main body of the paper for approximation details. The input to the network is an image batch, and should be normalized to $[0, 1]$.

2.3. The FAN Architecture

The FAN architecture (Tab. 6) is a fairly standard CNN model with an additional constrained convolution layer recommended for forensics applications [2]. In contrast to the study by Bayar and Stamm, who used only the green color channel, we take all color channels (RGB). We also used larger patches for better statistics - in all experiments, the input size is 128×128 px.

The network starts with a convolution layer constrained to learn a residual filter. We initialized the layer with the following residual filter (padded with zeros to 5×5):

$$\begin{bmatrix} -1, & -2, & -1 \\ -2, & 12, & -2 \\ -1, & -2, & -1 \end{bmatrix}. \quad (1)$$

We used leaky ReLUs instead of tanh for layer activation, and dispensed with batch normalization due to small network size and fast convergence without it.

2.4. Training Details

In our implementation, we use two Adam optimizers (with default settings) for: (1) updating the FAN (and in joint training also the NIP) based on the cross-entropy loss; (2) updating the NIP based on the image fidelity loss (L_2). The optimization steps are run in that order. To ensure comparable loss values, the L_2 loss was computed based on images normalized to the standard range $[0,255]$. Analogously to standalone NIP training, we feed raw image patches extracted from full-resolution images. In each epoch, the patches are chosen randomly, and fed in batches of 20. We start with learning rate of 10^{-4} and decrease it by 15% every 50 epochs.

References

- [1] Independent JPEG Group. <http://www.ijg.org/>. visited 18 Apr 2017.
- [2] B. Bayar and M. Stamm. Constrained convolutional neural networks: A new approach towards general purpose image manipulation detection. *IEEE Tran. Information Forensics and Security*, 13(11):2691–2706, 2018.
- [3] P. Cameron and S. Whited. Rawkit. <https://rawkit.readthedocs.io/en/latest/>. Visited Nov 2018.
- [4] C. Chen, Q. Chen, J. Xu, and V. Koltun. Learning to see in the dark. *arXiv*, 2018. arXiv:1805.01934.
- [5] Q. Chen, J. Xu, and V. Koltun. Fast image processing with fully-convolutional networks. *arXiv*, 2017. arXiv:1709.00643v1.
- [6] D. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato. RAISE - a raw images dataset for digital image forensics. In *Proc. of ACM Multimedia Systems*, 2015.
- [7] M. Gharbi, G. Chaurasia, S. Paris, and F. Durand. Deep joint demosaicking and denoising. *ACM Tran. Graphics*, 35(6):1–12, nov 2016.
- [8] LibRaw LLC. libRAW. <https://www.libraw.org/>. Visited Nov 2018.
- [9] D. Menon, S. Andriani, and G. Calvagno. Demosaicking with directional filtering and a posteriori decision. *IEEE Tran. Image Processing*, 16(1):132–141, 2007.
- [10] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Int. Conf. Medical Image Computing and Computer-assisted Intervention*. Springer, 2015.



(a) our implementation of the standard pipeline



(b) libRAW with default settings



(c) developed by the INet model



(d) developed by the UNet model



(e) developed by the DNet model

Figure 1. An example full-resolution (12.3 Mpx) image developed with standard pipelines (ab) and the considered NIPs (cde): image *r23beab04t* from the Nikon D90 camera (Raise dataset [6]). The full-size images are included as JPEGs (quality 85, 4:2:0) to limit PDF size. Close-up patches are included as uncompressed PNG files.

Table 2. The INet architecture: 321 trainable parameters

Operation	Activation	Initialization	Function	Output size
Input	-	-	RGGB feature maps	$N \times \frac{h}{5} \times \frac{w}{5} \times 4$
1×1 convolution	-	hand-crafted binary sample selection ¹	Reorganizes data for up-sampling	$N \times \frac{h}{2} \times \frac{w}{2} \times 12$
Depth to space	-	-	Up-sampling	$N \times h \times w \times 3$
5×5 convolution	-	zero-padded 3×3 bilinear kernel	Demosaicing	$N \times h \times w \times 3$
1×1 convolution	-	sample color conversion matrix	Color-space conversion (sRGB)	$N \times h \times w \times 3$
1×1 convolution	tanh	pre-trained model	Gamma correction ²	$N \times h \times w \times 12$
1×1 convolution	-	pre-trained model	Gamma correction ²	$N \times h \times w \times 3$
Clip to $[0,1]$	-	-	output RGB image	$N \times h \times w \times 3$

¹ we disabled optimization of this filter to speed up convergence

² adapted from a 2-layer network trained separately to approximate gamma correction

Table 3. The UNet architecture: 7,760,268 trainable parameters

Operation	Activation	Input	Output	Output size
Input	-	-	x	$N \times h/2 \times w/2 \times 4$
3×3 convolution	leaky ReLU	x	$c_{1,1}$	$N \times h/2 \times w/2 \times 32$
3×3 convolution	leaky ReLU	$c_{1,1}$	$c_{1,2}$	$N \times h/2 \times w/2 \times 32$
2×2 max pooling	-	$c_{1,2}$	p_1	$N \times h/4 \times w/4 \times 32$
3×3 convolution	leaky ReLU	p_1	$c_{2,1}$	$N \times h/4 \times w/4 \times 64$
3×3 convolution	leaky ReLU	$c_{2,1}$	$c_{2,2}$	$N \times h/4 \times w/4 \times 64$
2×2 max pooling	-	$c_{2,2}$	p_2	$N \times h/8 \times w/8 \times 32$
3×3 convolution	leaky ReLU	p_2	$c_{3,1}$	$N \times h/8 \times w/8 \times 128$
3×3 convolution	leaky ReLU	$c_{3,1}$	$c_{3,2}$	$N \times h/8 \times w/8 \times 128$
2×2 max pooling	-	$c_{3,2}$	p_3	$N \times h/16 \times w/16 \times 128$
3×3 convolution	leaky ReLU	p_3	$c_{4,1}$	$N \times h/16 \times w/16 \times 256$
3×3 convolution	leaky ReLU	$c_{4,1}$	$c_{4,2}$	$N \times h/16 \times w/16 \times 256$
2×2 max pooling	-	$c_{4,2}$	p_4	$N \times h/32 \times w/32 \times 256$
3×3 convolution	leaky ReLU	p_4	$c_{5,1}$	$N \times h/32 \times w/32 \times 512$
3×3 convolution	leaky ReLU	$c_{5,1}$	$c_{5,2}$	$N \times h/32 \times w/32 \times 512$
2×2 strided convolution	-	$c_{5,2}$	s_5	$N \times h/16 \times w/16 \times 256$
3×3 convolution	leaky ReLU	$s_5 \mid c_{4,2}$	$c_{6,1}$	$N \times h/16 \times w/16 \times 256$
3×3 convolution	leaky ReLU	$c_{6,1}$	$c_{6,2}$	$N \times h/16 \times w/16 \times 256$
2×2 strided convolution	-	$c_{6,2}$	s_6	$N \times h/8 \times w/8 \times 128$
3×3 convolution	leaky ReLU	$s_6 \mid c_{3,2}$	$c_{7,1}$	$N \times h/8 \times w/8 \times 128$
3×3 convolution	leaky ReLU	$c_{7,1}$	$c_{7,2}$	$N \times h/8 \times w/8 \times 128$
2×2 strided convolution	-	$c_{7,2}$	s_7	$N \times h/4 \times w/4 \times 64$
3×3 convolution	leaky ReLU	$s_7 \mid c_{2,2}$	$c_{8,1}$	$N \times h/4 \times w/4 \times 64$
3×3 convolution	leaky ReLU	$c_{8,1}$	$c_{8,2}$	$N \times h/4 \times w/4 \times 64$
2×2 strided convolution	-	$c_{7,2}$	s_8	$N \times h/2 \times w/2 \times 32$
3×3 convolution	leaky ReLU	$s_8 \mid c_{1,2}$	$c_{9,1}$	$N \times h/2 \times w/2 \times 32$
3×3 convolution	leaky ReLU	$c_{9,1}$	$c_{9,2}$	$N \times h/2 \times w/2 \times 32$
1×1 convolution	-	$c_{9,2}$	c_{10}	$N \times h/2 \times w/2 \times 12$
Depth to space	-	c_{10}	y_{rgb}	$N \times h \times w \times 3$
Clip to $[0,1]$	-	y_{rgb}	y	$N \times h \times w \times 3$

All leaky ReLUs have $\alpha = 0.2$

\mid denotes concatenation along the feature dimension

Table 4. The DNet architecture: 493,976 trainable parameters

Operation	Activation	Input	Output	Output size
Input	-	-	c_0	$N \times h/2 \times w/2 \times 4$
Repeat for $i = 1, 2, \dots, 14$ {				
3×3 convolution + BN	ReLU	c_{i-1}	\hat{c}_i	$N \times h/2 - 2 \times w/2 - 2 \times 64$
Padding (reflection)	-	\hat{c}_i	c_i	$N \times h/2 \times w/2 \times 64$
}				
3×3 convolution + BN	ReLU	c_{14}	\hat{c}_{15}	$N \times h/2 - 2 \times w/2 - 2 \times 12$
Padding (reflection)	-	\hat{c}_{15}	c_{15}	$N \times h/2 \times w/2 \times 12$
Depth to space	-	c_{15}	f_{conv}	$N \times h \times w \times 3$
1×1 convolution	-	c_0	c_{16}	$N \times h/2 \times w/2 \times 12$
Depth to space	-	c_{16}	f_{bayer}	$N \times h \times w \times 3$
3×3 convolution	ReLU	$f_{\text{conv}} \mid f_{\text{bayer}}$	\hat{c}_{17}	$N \times h - 2 \times w - 2 \times 64$
Padding (reflection)	-	\hat{c}_{17}	c_{17}	$N \times h \times w \times 64$
1×1 convolution	-	c_{17}	y_{rgb}	$N \times h \times w \times 3$
Clip to $[0,1]$	-	y_{rgb}	y	$N \times h \times w \times 3$

| denotes concatenation along the feature dimension

Table 5. The dJPEG architecture for JPEG codec approximation

Operation	JPEG Function	Output size
Input	-	$N \times h \times w \times 3$
1×1 convolution	RGB \rightarrow YCbCr	$N \times h \times w \times 3$
Space to depth & reshapes	Isolate 8×8 px blocks	$3N \times 8 \times 8 \times B$
Transpose & reshape	-	$3BN \times 8 \times 8$
$2 \times$ matrix multiplication	Forward 2D DCT	$3BN \times 8 \times 8$
Element-wise matrix division	Divide by quantization matrices	$3BN \times 8 \times 8$
Rounding / approximate rounding	Quantization	$3BN \times 8 \times 8$
Element-wise matrix multiplication	Multiply by quantization matrices	$3BN \times 8 \times 8$
$2 \times$ matrix multiplication	Inverse 2D DCT	$3BN \times 8 \times 8$
Transpose & reshape	-	$3N \times 8 \times 8 \times B$
Depth to space & reshapes	Re-assemble 8×8 px blocks	$N \times h \times w \times 3$
1×1 convolution	YCbCr \rightarrow RGB	$N \times h \times w \times 3$

Table 6. The FAN architecture: 1,341,990 trainable parameters

Operation	Activation	Initialization	Comment	Output size
Input	-	-	RGB input	$N \times h \times w \times 3$
5×5 convolution	-	Standard residual filter ¹	Constrained convolution	$N \times h \times w \times 3$
5×5 convolution	leaky ReLU	MSRA	-	$N \times h \times w \times 32$
2×2 max pool	-	-	-	$N \times h/2 \times w/2 \times 32$
5×5 convolution	leaky ReLU	MSRA	-	$N \times h/2 \times w/2 \times 64$
2×2 max pool	-	-	-	$N \times h/4 \times w/4 \times 64$
5×5 convolution	leaky ReLU	MSRA	-	$N \times h/4 \times w/4 \times 128$
2×2 max pool	-	-	-	$N \times h/8 \times w/8 \times 128$
5×5 convolution	leaky ReLU	MSRA	-	$N \times h/8 \times w/8 \times 256$
2×2 max pool	-	-	-	$N \times h/16 \times w/16 \times 256$
1×1 convolution	leaky ReLU	MSRA	-	$N \times h/16 \times w/16 \times 256$
global average pooling	-	-	-	$N \times 256$
fully connected	leaky ReLU	MSRA	-	$N \times 512$
fully connected	leaky ReLU	MSRA	-	$N \times 128$
fully connected	Softmax	MSRA	Class probabilities	$N \times 5$