From the holdings of the Department of Special Collections , Princeton University Library, Princeton, NJ

These documents are either in the public domain, or can only be used for educational and research purposes (Fair use) as per U.S. Copyright law (text below). By accessing this file, all users agree that their use falls in the public domain or within fair use as defined by the copyrightlaw. We do not charge any permission or usage fees for the publication of images of material inour collections, including those provided by Princeton University Library, via our website, catalog, or directly from staff.For more information regarding copyright, credit, and citation please review the information onthe Copyright, Credit and Citation Guidelines page of our website.
https://library.princeton.edu/special-collections/policies/copyright-credit-and-citation-guidelines. If you have any questions, including how to order higher-quality images from the collection, contact Special Collections Public Services via our Ask Us! Form. https://library.princeton.edu/special-collections/ask-us

---

Master's Theses, 2022-2026

Computer Science, 2023

2023

Defending Against Adversarial Attacks with Camera Image Pipelines

Zhang, Yuxuan

Princeton, NJ : Princeton University

---

http://arks.princeton.edu/ark:/88435/dsp01wh246w37v

U.S. Copyright law: The copyright law of the United States (Title 17, §108, United States Code) governs the making of photocopies or other reproductions of copyrighted material. Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be used for any purpose other than private study, scholarship, or research. If a user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of fair use that user may be liable for copyright infringement. This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve a violation of copyright law. https://www.copyright.gov/title17/title17.pdf

MSE Thesis: Defending Against Adversarial Attacks with Camera Image Pipelines

Yuxuan Zhang

Advisor: Professor Felix Heide

Reader: Professor Olga Russakovsky

Submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Engineering

Department of Computer Science

Princeton University

January 2023

I hereby declare that I am the author of this thesis.

I authorize Princeton University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Princeton University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Yuxuan Zhang

Abstract

Existing neural networks for computer vision tasks are vulnerable to adversarial attacks: adding imperceptible perturbations to the input images can fool these models into making a false prediction on an image that was correctly predicted without the perturbation. Various defense methods have proposed image-to-image mapping methods, either including these perturbations in the training process or removing them in a preprocessing step. In doing so, existing methods often ignore that the natural RGB images in today's datasets are not captured but, in fact, recovered from RAW color filter array captures that are subject to various degradations in the capture. In this work, we exploit this RAW data distribution as an empirical prior for adversarial defense. Specifically, we propose a model-agnostic adversarial defensive method, which maps the input RGB images to Bayer RAW space and back to output RGB using a learned camera image signal processing (ISP) pipeline to eliminate potential adversarial patterns. The proposed method acts as an off-the-shelf preprocessing module and, unlike model-specific adversarial training methods, does not require adversarial images to train. As a result, the method generalizes to unseen tasks without additional retraining. Experiments on large-scale datasets, *e.g.*, ImageNet, COCO, for different vision tasks, *e.g.*, classification, semantic segmentation, object detection, validate that the method significantly outperforms existing methods across task domains.

Contents

Abstract iii

List of Tables vi

List of Figures ix

1 Introduction **1**

2 Related Works **4**

2.1 Camera Image Signal Processing (ISP) Pipeline 4

2.2 Adversarial Attack Methods 5

2.3 Defense Methods 6

3 Sensor Image Formation **8**

4 Proposed Defense Method **11**

4.1 F Operator: Image-to-RAW Mapping 13

4.2 G Operator: Learned ISP 14

4.3 S Operator: Conventional ISP 15

4.4 Operator Training 15

5 Experiment Results **17**

5.1 Experimental Setup 18

5.2 Main Results & Analysis 20

5.3 RAW Distribution Analysis 21

5.4 Robustness to Hyperparameter and Operator Deviations 23
5.5 Super White-box Attack 24

6 Conclusion and future works **26**

A Additional Experiment Results **39**
A.1 Additional Qualitative Results 39
A.2 Against BPDA Attack 41

B Attack and Defense Algorithms **43**
B.1 Attack Algorithm Details 43
B.2 Defense Algorithm Details 46

C Implementation Details **48**
C.1 Implementation Details for the S operator 48
C.2 Implementation Details for the F operator 49

D Released Codes & Instruction **51**
D.1 Experiments Environment: 51
D.2 Training 52
D.3 Evaluation 53
D.4 Pre-trained weight 53

List of Tables

5.1 **Quantitative Comparisons on ImageNet:** We evaluate Top-1 Accuracy on ImageNet and compare the proposed method to existing input-transformation methods. The best Top-1 accuracies are marked in bold. Our defense method offers the best performance in all settings, except for the DeepFool attack. 17

5.2 **Quantitative Comparison to SOTA Input-Transformation Defense Methods on the COCO dataset:** We evaluate all methods for mean IoU (mIoU) and mark the best mIoU in bold. Our defense method achieves the best performance in all settings. 18

5.3 **Quantitative Comparison to SOTA Input-Transformation Defense Methods on the ADE20K dataset.** We evaluate all methods for mean IoU (mIoU) and mark the best mIoU in bold. Our defense method achieves the best performance in all settings. 18

5.4 **Quantitative Comparison to SOTA Input-Transformation Defenses on the Pascal VOC dataset**. We evaluate all compared methods on mean average precision (mAP) on the Pascal VOC dataset. The best mAPs are marked in bold. Our defense method achieves the best performance in all settings. 19

5.5 **Quantitative Ablation Study on RAW Training Datasets.** We train F and G operators on three different RAW-RGB datasets and report the Top-1 defense accuracy on the ImageNet dataset. The RAW images in the Zurich-Raw-to-RGB and HDR-RAW-RGB are captured by real cameras, while the ones in MIT-RAW-RGB are synthesized. We see a sharp performance drop when swapping the real RAW training data to synthetic data due to the lack of natural RAW distribution cues. 23

5.6 **Effect of Different Intermediate Mapping Spaces.** We report the Top-1 adversarial defense accuracy on the ImageNet dataset when mapping to different intermediate mapping spaces that are the steps of the image processing pipeline. The performance drops as the intermediate image space moves from RAW to the RGB output space. This validates the importance of exploiting the RAW space in the proposed defense. 23

5.7 **Effect of hyperparameter ω.** We evaluate the impact of the method hyperparameter ω on the effectiveness of the proposed defense method. 24

5.8 **Robustness to Deviations of F and G.** We evaluate the defense accuracy when mixing operator from different training epochs. 24

5.9 **Robustness to Deviations of F.** We perturb the output of operator F with Gaussian noise of different standard deviations and report the defense accuracy. 24

5.10 **Against the super-white attack**: We evaluate the defense performance against a super-white attack setting, where we assume the preprocessing model is exposed and the adversary can calculate the gradient for the differentiable part of the preprocessing model to construct stronger adversarial attacks. In such a setting, the differentiable preprocessing module (*i.e.*, ComDefend and G only) fail while our method does not collapse, demonstrating the benefit of exploiting the conventional ISP operator S in our method. 25

A.1 **Quantitative comparison with baseline input-transformation defense methods under BPDA attack:** The proposed modification improves the defending accuracy from 14.28 to 38.85, and the resulting method significantly outperforms the baselines. 42

A.2 **Quantitative Comparisons for the proposed method w/ and w/o modification for BPDA.** The modification significantly increases the robustness of the proposed method under BPDA attack, at the cost of slightly decreasing the defense performance under other attacks. The relative ranking of the proposed defense method versus baseline defense methods remains the same, with the proposed defense method outperforming baselines with a large margin both before and after modification. 42

C.1 **Architecture description of the F Operator**. 50

List of Figures

1.1 **Proposed Approach:** Existing defense approaches learn an RGB-to-RGB projection from an adversarial distribution (Y') to its natural RGB distribution (Y): $T : Y' \rightarrow Y$. In contrast, our approach learns a mapping via the intermediate natural RAW distribution (X), which is achieved by utilizing three specially designed operators: $F : Y' \rightarrow X$, $G : X \rightarrow Y$, and $S : X \rightarrow Y$. 2

3.1 **Overview of the RAW imaging pipeline model:** The scene light field is captured by compound camera optics, and then it is gathered by a microlens array layer before being captured on the color filter array. The color-filtered photons are converted into electrons based on quantum efficiency before adding dark current and noise. Next, the converted electrons are clipped based on the maximum well capacity, e_s, and scaled by a sensor gain factor κ. Finally, an ADC converts the analog signal into a digital readout with quantization noise n_q, I_{W}. . 9

4.1 **Overview of the proposed defense approach:** see text. We note that the resolution of the RAW image (RGGB) is twice larger than that of the RGB image. We linearly scaled the RAW image in this figure for better visualization. 12

4.2 G **operator:** The architecture of the G operator is adopted and modified from PyNet [33]. The finer operator level exploits upsampled coarser-level features to reconstruct the RGB output. The model is trained sequentially in a coarse-to-fine manner. 12

5.1 **Qualitative Comparison:** Qualitative outputs of the proposed method along with both G and S operators and state-of-the-art defense methods on the ImageNet dataset; see text. 18

A.1 **Additional Qualitative Comparison:** Qualitative comparison of the proposed method along with both G and S operators against state-of-the-art defense methods on the ImageNet dataset, see text. 40

Chapter 1

Introduction

The most successful methods for a wide range of computer vision tasks rely on deep neural networks [10, 29, 30, 35, 94] (DNNs), including classification, detection, segmentation, scene understanding, scene reconstruction, and generative tasks. Although we rely on the predictions of DNNs for safety-critical applications in robotics, self-driving vehicles, medical diagnostics, and video security, existing networks have been shown to be vulnerable to adversarial attacks[78]: small perturbations to images that are imperceptible to human vision system can deceive DNNs to make incorrect predictions[53, 65, 77, 58, 82]. As such, defending against adversarial attacks[4, 52, 53, 62, 89] can help resolve failure cases in safety-critical applications and provide insights into the generalization capabilities of training procedures and network architectures.

Existing defense methods fall into two approaches: they either introduce adversarial examples to the training dataset, resulting in new model weights, or transform the inputs, aiming to remove the adversarial pattern, before feeding them into the unmodified target models. Specifically, the first line of defense methods generates adversarial examples by iteratively training a target model while finding and adding remaining adversarial images as training samples in each iteration[92, 85, 85, 84, 23]. Although the set of successful adversarial examples shrinks over time, iteratively generating

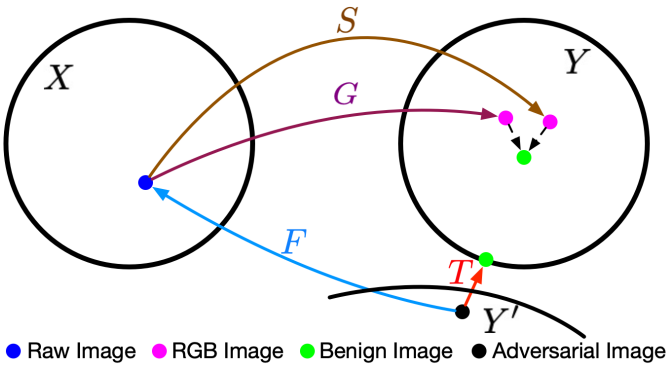


Figure 1.1: **Proposed Approach:** Existing defense approaches learn an RGB-to-RGB projection from an adversarial distribution (Y') to its natural RGB distribution (Y): $T : Y' \rightarrow Y$. In contrast, our approach learns a mapping via the intermediate natural RAW distribution (X), which is achieved by utilizing three specially designed operators: $F : Y' \rightarrow X$, $G : X \rightarrow Y$, and $S : X \rightarrow Y$.

them is extremely costly in training time, and different adversarial images must be included for defending against different attack algorithms. Moreover, the adversarial examples cannot be captured in a single training set as they are model-specific and domain-specific, meaning they must be re-generated when used for different models or on other domains.

Defense methods that transform the input image aim to overcome the limitations of adversarial training approaches. Considering adversarial perturbations as noise, these methods "denoise" the inputs before feeding them into unmodified target models. The preprocessing module can either employ image-to-image models such as auto-encoders or generative adversarial methods[38, 71, 51] or rely on conventional image-processing operations[26, 15, 18, 46]. Compared to adversarial training methods, these methods are model-agnostic and require no adversarial images for training.

All methods in this approach have in **common that they rely on RGB image data as input and output**. That is, they aim to recover the distribution of natural RGB images and project the adversarial image input to the closest match in this distribution, using a direct image-to-image mapping network. As such, existing methods often ignore the fact that images in natural image datasets are the result of several processing steps applied to the raw captured images. In particular, training datasets are produced by interpolating sub-sampled, color filtered (*e.g.*, using Bayer

filter) raw data, followed by a rich low-level processing pipeline, including readout and photon noise denoising. As a result, the raw per-pixel photon counts are heavily subsampled, degraded, and processed in an RGB image. We rely on the **RAW data distribution, before becoming RGB images, as a prior in the proposed adversarial defense method**, which is empirically described in large datasets of RAW camera captures. Specifically, instead of directly learning a mapping between adversarially perturbed input RGBs and "clean" output RGBs, we learn a mapping via the intermediate RAW color filter array domain. In this mapping, we rely on learned ISP pipelines as low-level camera image processing blocks to map from RAW to RGB. The resulting method is entirely model-agnostic, requires no adversarial examples to train, and acts as an off-the-shelf preprocessing module that can be transferred to any task on any domain. We validate our method on large-scale datasets (ImageNet, COCO) for different vision tasks (classification, semantic segmentation, object detection) and also perform extensive ablation studies to assess the robustness of the proposed method to various attack methods, model architecture choices, and hyperparameters choices. Code and data to reproduce the findings in this work are available at https://github.com/princeton-computational-imaging/allyouneedisraw.

Specifically, we make the following contributions:

- We propose the first adversarial defense method that exploits the natural distribution of RAW domain images.
- The proposed method avoids the generation of adversarial training images and can be used as an off-the-shelf preprocessing module for diverse tasks.
- We analyze how the natural RAW image distribution helps defend against adversarial attacks, and we validate that the method achieves *state-of-the-art* defense accuracy for input transformation defenses, outperforming existing approaches.

Chapter 2

Related Works

2.1 Camera Image Signal Processing (ISP) Pipeline

A camera image signal processing (ISP) pipeline converts RAW measurements from a digital camera sensor to high-quality images suitable for human viewing or downstream analytic tasks. To this end, a typical ISP pipeline encompasses a sequence of modules [39], each addressing a portion of this image reconstruction problem. In a hardware ISP, these modules are proprietary compute units, and their behaviors are unknown to the user. More importantly, the modules are not differentiable [80, 57]. Two lines of work rely on deep-learning-based approaches to cope with the significant drawback.

The first flavor of methods directly replaces the hardware ISP with a deep-learning-based model to target different application scenarios, such as low-light enhancement [9, 8], super-resolution [90, 91, 95], smartphone camera enhancement [72, 14, 34], and ISP replacement [45]. Nevertheless, the deep-learning-based models used by these works contain a massive number of parameters and are computationally expensive. Hence, their application is limited to offline tasks. In contrast, another thread of works focused on searching for the best hardware ISP hyperparameters

for different downstream tasks by using deep-learning-based approaches. Specifically, Tseng *et al.* [80] proposed differentiable proxy functions to model arbitrary ISP pipelines and used them to find the best hardware ISP hyperparameters for different downstream tasks. Yu *et al.* [93] proposed ReconfigISP, which uses different proxy functions for each module of a hardware ISP instead of the entire ISP pipeline. Mosleh *et al.* [57] proposed a hardware-in-the-loop method to optimize hyperparameters of a hardware ISP directly.

2.2 Adversarial Attack Methods

Adversarial attacks have drawn significant attention from the deep-learning community. Based on the level of access to target networks, adversarial attacks can be broadly categorized into white-box attacks and black-box attacks.

Among white-box attacks, one important direction is gradient-based attacks[24, 42, 53]. These approaches generate adversarial samples based on the gradient of the loss function with respect to input images. Another flavor of attacks is based on solving optimization problems to generate adversarial samples [77, 6]. In the black-box setting, only benign images and their class labels are given, meaning attackers can only query the target model. With the free query, Black-box attacks rely on adversarial transferability to train substitute models [31, 63, 75, 61] or directly estimate the target model gradients [12, 13, 83] to generate adversarial examples. To avoid the transferability and the overhead of gathering data to train a substitute model, several works proposed local-search-based black-box attacks to generate adversarial samples directly in the input domain [59, 44, 5].

In the physical world, adversarial samples are captured by cameras providing image inputs to target networks. A variety of strategies have been developed to guard the effectiveness of adversarial patterns in the wild [2, 21, 36, 17]. These methods

typically assume that the camera acquisition and subsequent hardware processing do not alter the adversarial patterns. Phan *et al.* [64] have recently realized attacks on individual camera types by exploiting slight differences in their hardware ISPs and optical systems.

2.3 Defense Methods

In response to adversarial attack methods, there have been significant efforts in constructing defenses to counter those attacks. These include adversarial training [53], input transformation [19, 3], defensive distillation [62], dynamic models [86], loss modifications [60], model ensemble [73] and robust architecture [27]. We next analyze the two representative categories of defense methods.

Adversarial Training (AT): The idea of AT is the following: in each training loop, we augment training data with adversarial examples generated by different attacks. AT is known to "overfit" to the attacks "seen" during training and has been demonstrated to be vastly effective in defending those attacks. However, AT does not generalize well to "unseen" attacks [76]. Furthermore, iteratively generating adversarial images is time-consuming, causing 3-30 times longer than standard training before the model converges [74]. Multiple methods have been proposed to reduce the training time, making AT on large datasets (*e.g.*, ImageNet) possible [85, 84, 23, 96]. Even so, for each specific model, the approach still requires an extra adversarial training process and suffers from cross-domain attacks. Besides the target model, it is also worth noting that adversarial examples can be used to train the input preprocessing models. [46].

Input Transformation (IT): IT, as an image pre-processing approach, aims to remove adversarial patterns to counter attacks. A considerable number of IT methods have been proposed, such as JPEG compression [19, 50, 15], randomization [87],

image quilting [26], pixel deflection [66], and deep-learning-based approaches [38, 51, 71]. These IT methods can seamlessly work with different downstream models and tasks. More importantly, IT methods can be easily combined with other model-specific defense methods to offer a stronger defense.

Our work falls into the IT category. Unlike existing IT methods to focus on the preprocessing in the RGB distribution, the proposed approach leverages the intermediate natural RAW distribution to remove adversarial patterns.

Chapter 3

Sensor Image Formation

In this section, we review how a RAW image is formed. When light from the scene enters a camera aperture, it first passes through compound camera optics which focus the light on an image sensor (*e.g.*, CCD and CMOS), where the photons are color-filtered and converted to electrons. Finally, the electrons are converted to digital values, comprising a RAW image. We refer the reader to Karaimer and Brown [40] for a detailed review.

Compound Camera Optics: A compound lens consisting of a sequence of optics is designed to correct optical aberrations. When scene radiance, I_{SCENE} (in the form of a light field) enters a compound lens, the radiance is modulated by the complex optical pipeline and generates the image I_{O}. Compound optics can be modeled by spatially-varying point spread functions (PSFs) [79].

Color Image Sensor Model: A conventional color image sensor has three layers. On the top is a micro-lens array (MLA) layer; the bottom is a matrix of small potential wells; a color filter array (CFA) layer sits in the middle. When I_O falls on a color image sensor, photons first pass through the MLA to improve light collection. Next, light passes through the CFA layer, resulting in a mosaic pattern of the three stimulus RGB colors. Finally, the bottom layer collects the color-filtered light and outputs a single channel RAW image, I_W.

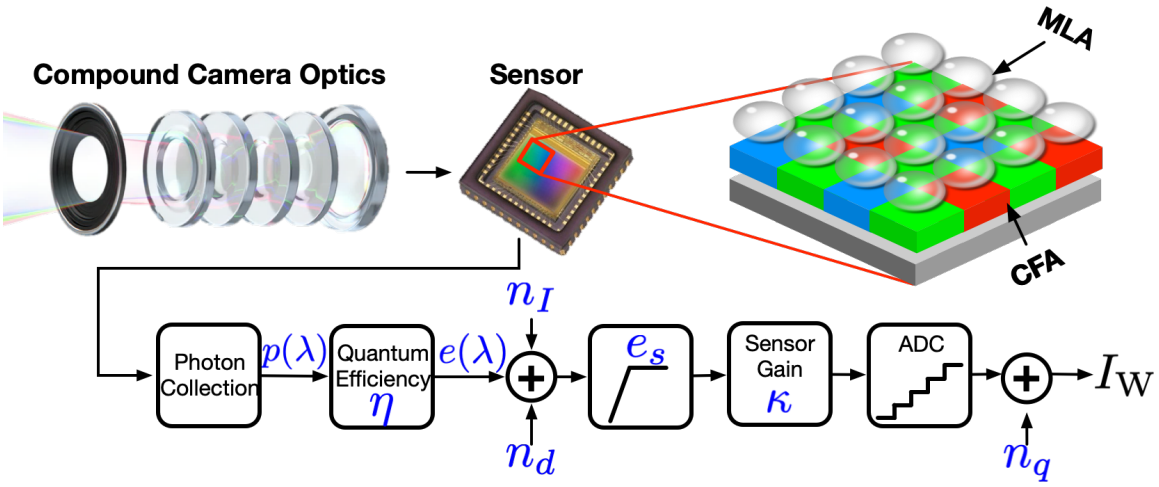


Figure 3.1: **Overview of the RAW imaging pipeline model:** The scene light field is captured by compound camera optics, and then it is gathered by a microlens array layer before being captured on the color filter array. The color-filtered photons are converted into electrons based on quantum efficiency before adding dark current and noise. Next, the converted electrons are clipped based on the maximum well capacity, e_s, and scaled by a sensor gain factor κ. Finally, an ADC converts the analog signal into a digital readout with quantization noise n_q, I_{W}.

The detailed process is illustrated in Figure 3.1. In particular, at the bottom layer, a potential well counts photons arriving at its location (x, y) and converts the accumulated photons to electrons, and the conversion process is specified by the detector quantum efficiency. During this process, electrons can fluctuate randomly which we summarize as electron noise. Two common types of electron noise are the dark noise n_d, which is independent of light; and dark current n_I, which depends on the sensor temperature. These follow normal and Poisson distributions, respectively [79]. Next, the converted electrons are clipped based on the maximum well capacity, e_s, and scaled by a sensor gain factor κ. Finally, the modulated electrons are converted to digital values by an analog-to-digital converter (ADC), which quantizes the input and introduces a small amount of noise, n_q.

Mathematically, a pixel of a RAW image, I_W, at position (x, y) can be defined as

$$I_W(x, y) = b + n_q + \kappa \min(e_s, n_d + n_I + \sum_{\lambda} e(x, y, \lambda)), \tag{3.1}$$

where b is the black level, level of brightness with no light; $e(x, y, \lambda)$ is the number of electrons arrived at a well at position (x, y) for wavelength λ.

This image formation model reveals that besides the natural scene being captured, RAW images heavily depend on the *specific stochastic nature of the optics, color filtering, sensing, and readout components.* The proposed method exploits these statistics.

Chapter 4

Proposed Defense Method

In this section, we describe the proposed defense method, which exploits the distribution of RAW measurements as a prior to project adversarially perturbed RGB images to benign ones. Given an adversarial input, existing defense approaches learn an RGB-to-RGB projection from the adversarially perturbed distribution of RGB images, Y', to the closest point in the corresponding RGB natural distribution, Y. We use the operator $T : Y' \rightarrow Y$ for this projection operation. As this RGB distribution Y empirically samples from the ISP outputs of diverse existing cameras, it also ingests diverse reconstruction artifacts, making it impossible to exploit photon-flux-specific cues, e.g., photon shot noise, optical aberrations, or camera-specific readout characteristics – image processing pipelines are designed to remove such RAW cues.

Departing from existing methods, as illustrated in Figure 1.1, we learn a mapping from Y' to Y via an intermediate RAW distribution, X, which incorporates these RAW statistics of natural images, such as sensor photon counts, multispectral color filter array distributions and optical aberrations. To this end, the approach leverages three specially designed operators: $F : Y' \rightarrow X$, $G : X \rightarrow Y$, and $S : X \rightarrow Y$. Specifically, the F operator is a learned model, which maps an adversarial sample from its adversarial distribution to its corresponding RAW sample in the natural image distribution of RAW images. Operator G is another learned network that

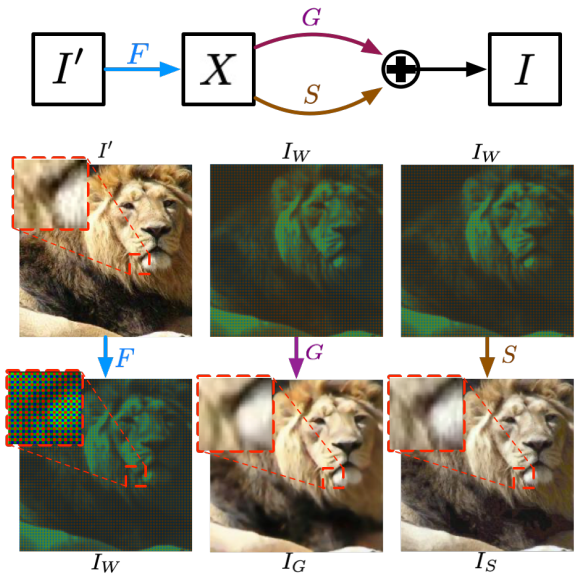


Figure 4.1: **Overview of the proposed defense approach:** see text. We note that the resolution of the RAW image (RGGB) is twice larger than that of the RGB image. We linearly scaled the RAW image in this figure for better visualization.

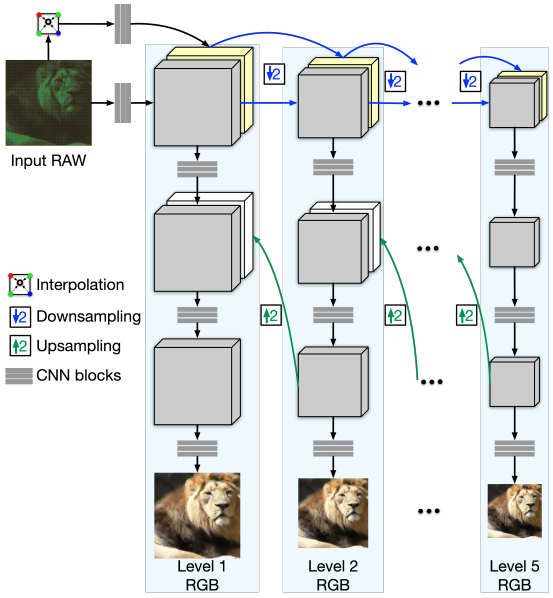


Figure 4.2: G **operator:** The architecture of the G operator is adopted and modified from PyNet [33]. The finer operator level exploits upsampled coarser-level features to reconstruct the RGB output. The model is trained sequentially in a coarse-to-fine manner.

performs an ISP reconstruction task, *i.e.*, it converts a RAW image to an RGB image. In theory, our goal can be achieved with these two operators by concatenating both $G(F(\cdot)) : Y' \rightarrow X \rightarrow Y$. However, as these two operators are differentiable models, the potential adversary may still be able to attack the model if, under stronger attack assumptions, he has full access to the weight of preprocessing modules. To address this issue, we rely on an additional operator S, a conventional ISP, to our approach, which is implemented as a sequence of cascaded software-based sub-modules. In contrast to the operator F, the operator S is non-differentiable. The operators F and G are trained separately without end-to-end fine-tuning. Notably, the proposed defense scheme is *entirely model-agnostic* as it does not require any knowledge of potential adversarial attacks.

For defending against an attack, as shown in Figure 4.1, the proposed approach

first uses the F operator to map an input adversarial image, I', to its intermediate RAW measurements, I_W. Then, I_W is processed separately by the G and S operators to convert it to two images in the natural RGB distribution, denoted as I_G and I_S, respectively. Finally, our method outputs a benign image, I, in the natural RGB distribution, by combining I_G and I_S in a weighted-sum manner. Mathematically, the defense process is defined as

$$I = \omega G(F(I')) + (1 - \omega)S(F(I')), \tag{4.1}$$

where ω is a hyper-parameter for weighting the contributions from the two operators G and S. In the following sections, we introduce each operator in detail.

4.1 F Operator: Image-to-RAW Mapping

We use a small learned encoder-decoder network as the F operator to map an RGB image to its intermediate RAW measurements. The details of network architecture are given in the Supplementary Material.

We train this module in a supervised manner with two $\mathcal{L}_2$ losses. Both $\mathcal{L}_2$ losses are calculated between the ground truth (GT) RAW and estimated RAW images. The only difference between the two losses is the input RGB image corresponding to the estimated RAW. One is using the original input RGB image, while the other is generated by adding Gaussian noise to the original input RGB image. In doing so, F is trained with the ability to convert both benign and slightly perturbed RGB images to their corresponding RAW distributions. We note that the added Gaussian distribution is *different from the correlated noise generated by various adversarial attacks*. Mathematically, given a benign RGB image, I, and its corresponding GT RAW measurements, GT_W, the loss function is defined as

$$\mathcal{L}_F = ||F(I), GT_W||_2 + ||F(I + \alpha\varepsilon), GT_W||_2, \tag{4.2}$$

$$\varepsilon \sim \mathcal{N}(\mu, \sigma), \tag{4.3}$$

where ε is a Gaussian noise with mean, μ, and standard deviation, σ; α is a random number in the range between 0 and 1, weighting the amount of noise added. We empirically set μ and σ to 0 and 1, respectively.

4.2 G Operator: Learned ISP

The G operator is represented by a neural network that converts the I_w generated by the F operator to an RGB image. During this process, we aim to guide local adjustment by global contextual information. This motivates us to devise a pyramidal convolutional neural network to fuse global and local features. To this end, we propose a variant of PyNet [33] consisting of five levels; see Figure 4.2. Here, the $1st$ level is the finest, and the $5th$ level is the coarsest. The finer levels use upsampled features from the coarser levels by concatenating them. We modify PyNet by adding an interpolation layer before the input of each level, interpolating the downsampled RAW Bayer pattern. This facilitates learning as the network only needs to learn the residuals between interpolated RGB and ground truth RGB.

The loss function for this model consists of three components: perceptual, structural similarity, and $\mathcal{L}_2$ loss. Given an input RAW image, I_W, and the corresponding GT RGB image GT_I, the loss function can be defined as

$$\begin{aligned}\mathcal{L}_G^i =& \beta^i \mathcal{L}_{Perc}(G(I_W), GT_I) + \gamma^i \mathcal{L}_{SSIM}((G(I_W), GT_I)) \\ &+ \mathcal{L}_2(G(I_W), GT_I) \qquad \text{for} \quad i \in [1, 5],\end{aligned} \tag{4.4}$$

where i represents the training level. As the model is trained in a coarse-to-fine manner, different losses are used for each level i. $\mathcal{L}_{Perc}$, $\mathcal{L}_{SSIM}$, and $\mathcal{L}_2$ represent

the perceptual loss calculated with VGG architecture, structural similarity loss, and $\mathcal{L}_2$ loss, respectively; β^i and γ^i are the two hyperparameter weights, which are set empirically. The model is trained sequentially in a coarse-to-fine manner, *i.e.*, from $i = 5$ to $i = 1$.

4.3 S Operator: Conventional ISP

The S operator has the same functionality as the G operator, converting a RAW image to an RGB image. Unlike the G operator, the S operator offers the functionalities of a conventional hardware ISP pipeline using a sequence of cascaded sub-modules, which is non-differentiable.

While we may use the ISP pipeline of any digital camera we can extract raw and post-ISP data from, we employ a software-based ISP pipeline consisting of the following components: Bayer demosaicing, color balancing, white balancing, contrast improvement, and colorspace conversion sub-modules. Using the Zurich-Raw-to-RGB dataset [34], we manually tune the hyperparameters of all sub-modules. We refer the reader to the Supplementary Material for a detailed description.

4.4 Operator Training

We use the Zurich-Raw-to-RGB dataset [34] to train the F and G operators. The Zurich-Raw-to-RGB dataset consists of 20,000 RAW-RGB image pairs, captured using a Huawei P20 smartphone with a 12.3 MP Sony Exmor IMX380 sensor and a Canon 5D Mark IV DSLR. Both of the F and G operators are trained in PyTorch with Adam optimizer on NVIDIA A100 GPUs. We set the learning rate to 1e$-$4 and 5e$-$5 for training the F and G operators, respectively. We use the following hyperparameters settings: $\omega = 0.7$ in Eq. 4.1; $\mu = 0$ and $\sigma = 1$ for the Gaussian component in Eq. 4.3; In Eq. 4.4, β^i is set to 1 for $i \in [1, 3]$ and 0 for $i \in [4, 5]$; γ^i is set to 1 for

$i = 1$ and 0 for $i \in [2, 5]$.

Chapter 5

Experiment Results

	FSGM		PGD		BIM		DeepFool		C&W		NewtonFool	BPDA
	2/255 ↑	4/255 ↑	2/255 ↑	4/255 ↑	2/255 ↑	4/255 ↑	L_∞ ↑	L_2 ↑	L_∞ ↑	L_2 ↑	L_∞ ↑	L_∞ ↑
ResNet-101												
JPEG-Defense[19]	33.14	20.71	45.19	21.74	36.78	8.5	53.16	45.69	59.06	52.01	24.65	0.08
TVM[26]	43.75	40.02	45.46	44.35	44.86	41.93	47.69	39.89	45.51	40.44	22.6	6.39
Randomized Resizing & Padding[87]	45.21	34.97	45.38	27.75	40.04	18.04	73.06	62.47	66.53	59.87	27.93	2.66
HGD [47]	54.75	43.85	55.26	50.05	56.74	48.61	64.34	58.13	59.98	52.88	27.70	0.03
Pixel-Deflection[66]	54.56	35.14	60.68	34.86	58.71	41.91	**75.97**	**64.13**	66.29	60.91	28.81	1.87
ComDefend[38]	48.21	36.51	53.28	48.38	51.39	42.01	63.68	55.62	58.53	50.38	26.46	0.03
Proposed Method	**66.02**	**58.85**	**68.34**	**66.17**	**66.91**	**63.01**	72.04	63.52	**71.40**	**67.33**	**40.96**	**38.85**
InceptionV3												
JPEG-Defense[19]	31.97	20.25	43.34	21.15	34.68	8.55	51.20	43.49	55.00	50.39	24.06	0.12
TVM[26]	42.47	37.23	42.75	41.61	42.80	39.71	45.21	37.39	43.27	37.51	23.05	4.58
Randomized Resizing & Padding[87]	41.86	34.49	43.41	25.60	39.42	16.62	70.24	58.65	63.24	55.62	27.55	2.09
HGD [47]	52.83	40.99	50.35	47.62	56.02	47.78	60.33	56.61	59.55	52.0	26.84	0.03
Pixel-Deflection[66]	51.42	34.27	56.13	32.49	56.18	39.13	**71.16**	**61.58**	61.94	57.58	28.01	1.56
ComDefend[38]	47.00	35.34	49.99	46.15	48.74	39.58	60.01	52.47	55.85	47.70	25.44	0.03
Proposed Method	**63.03**	**56.34**	**65.69**	**63.03**	**64.77**	**59.49**	69.25	60.04	**66.97**	**64.69**	**38.01**	**36.43**

Table 5.1: **Quantitative Comparisons on ImageNet:** We evaluate Top-1 Accuracy on ImageNet and compare the proposed method to existing input-transformation methods. The best Top-1 accuracies are marked in bold. Our defense method offers the best performance in all settings, except for the DeepFool attack.

The proposed method acts as an off-the-shelf input preprocessing module, requiring no additional training to be transferred to different tasks. To validate the effectiveness and generalization capabilities of the proposed defense approach, we evaluate the method on three different vision tasks, *i.e.*, classification, semantic segmentation, and 2D object detection, with corresponding adversarial attacks.

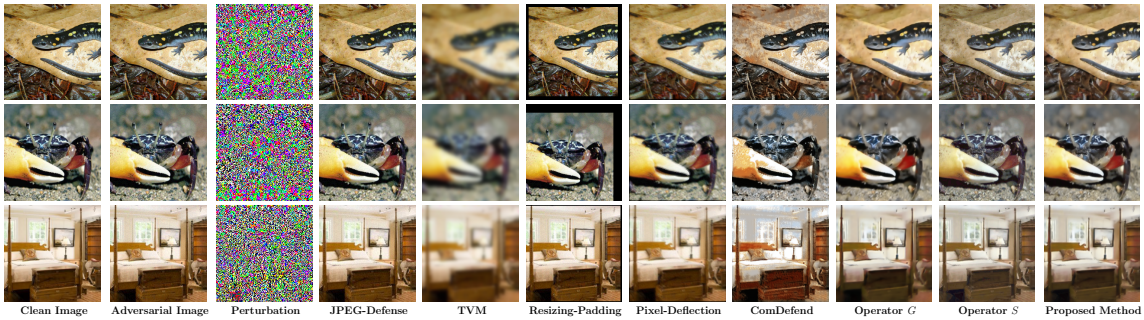


Figure 5.1: **Qualitative Comparison:** Qualitative outputs of the proposed method along with both G and S operators and state-of-the-art defense methods on the ImageNet dataset; see text.

	FSGM		PGD		BIM		DAG	
	$L_\infty = 2/255 \uparrow$	$L_\infty = 4/255 \uparrow$	$L_\infty = 2/255 \uparrow$	$L_\infty = 4/255 \uparrow$	$L_\infty = 2/255 \uparrow$	$L_\infty = 4/255 \uparrow$	$L_\infty = 2/255 \uparrow$	$L_\infty = 4/255 \uparrow$
JPEG-Defense[19]	37.41	32.27	24.53	6.21	25.74	10.18	14.12	5.66
TVM[26]	42.64	41.53	45.55	42.24	44.51	38.44	31.88	25.56
HGD [47]	43.39	40.82	44.54	40.88	40.03	39.95	28.61	22.36
Pixel-Deflection[66]	44.13	41.88	46.38	42.32	44.78	37.22	30.73	24.61
ComDefend[38]	45.57	39.23	44.85	41.14	42.71	36.12	28.94	23.36
Proposed Method	**52.35**	**48.04**	**53.41**	**49.59**	**54.86**	**50.51**	**40.35**	**37.88**

Table 5.2: **Quantitative Comparison to SOTA Input-Transformation Defense Methods on the COCO dataset:** We evaluate all methods for mean IoU (mIoU) and mark the best mIoU in bold. Our defense method achieves the best performance in all settings.

5.1 Experimental Setup

Adversarial Attack Methods: We evaluate our method by defending against the following attacks: FGSM [25], BIM [43], PGD [54] , C&W [7], NewtonFool [37], and DeepFool [56]. For classification, we use the widely used Foolbox benchmarking suite [67] to implement these attack methods. Since Foolbox does not directly support semantic segmentation and object detection attacks, we use the lightweight TorchAt-

	FSGM		PGD		BIM		DAG	
	$L_\infty = 2/255 \uparrow$	$L_\infty = 4/255 \uparrow$	$L_\infty = 2/255 \uparrow$	$L_\infty = 4/255 \uparrow$	$L_\infty = 2/255 \uparrow$	$L_\infty = 4/255 \uparrow$	$L_\infty = 2/255 \uparrow$	$L_\infty = 4/255 \uparrow$
JPEG-Defense[19]	22.18	17.61	23.40	8.37	20.39	6.55	7.09	4.35
TVM [26]	26.72	24.79	23.91	21.07	20.60	18.53	16.12	14.28
HGD [47]	31.15	26.23	29.44	25.78	24.42	23.08	20.17	19.75
Pixel-Deflection [66]	29.40	25.36	30.28	26.59	28.94	21.30	18.82	17.55
ComDefend [38]	30.62	25.89	28.74	27.69	27.08	21.39	20.83	19.42
Proposed Method	**37.83**	**35.15**	**39.22**	**36.20**	**38.63**	**35.18**	**25.09**	**22.47**

Table 5.3: **Quantitative Comparison to SOTA Input-Transformation Defense Methods on the ADE20K dataset.** We evaluate all methods for mean IoU (mIoU) and mark the best mIoU in bold. Our defense method achieves the best performance in all settings.

	FSGM		PGD		BIM		DAG	
	$L_\infty = 2/255 \uparrow$	$L_\infty = 4/255 \uparrow$	$L_\infty = 2/255 \uparrow$	$L_\infty = 4/255 \uparrow$	$L_\infty = 2/255 \uparrow$	$L_\infty = 4/255 \uparrow$	$L_\infty = 2/255 \uparrow$	$L_\infty = 4/255 \uparrow$
JPEG-Defense[19]	39.02	35.88	37.96	33.51	38.85	34.69	30.72	25.07
TVM[26]	48.11	39.66	47.1	44.38	48.94	41.76	39.20	33.18
HGD [47]	50.68	40.06	51.24	45.92	46.80	39.74	41.15	37.23
Pixel-Deflection[66]	53.77	44.82	54.45	47.22	55.32	48.32	46.52	39.87
ComDefend[38]	50.18	42.93	50.46	43.08	52.32	44.2	44.68	37.22
Proposed Method	**61.68**	**59.37**	**64.71**	**60.23**	**66.52**	**61.82**	**57.83**	**54.12**

Table 5.4: **Quantitative Comparison to SOTA Input-Transformation Defenses on the Pascal VOC dataset**. We evaluate all compared methods on mean average precision (mAP) on the Pascal VOC dataset. The best mAPs are marked in bold. Our defense method achieves the best performance in all settings.

tacks library [41] for generating adversarial examples with FGSM, PGD, and BIM attacks. We also evaluate against the DAG [88] attack, a dedicated attack approach for semantic segmentation and object detection tasks. Moreover, we further evaluate against BPDA [1], an attack method specifically designed for circumventing input transformation defenses that rely on obfuscated gradients. Defending against BPDA with our method, however, requires a slight modification at inference time; see Supplementary Material for details. We note that all applied attacks are untargeted. Definitions of all attack methods are provided in the Supplementary Material.

Baseline Defense Approaches: We compare to the following input transformation defense methods: JPEG compression [19], randomized resizing & padding[87], image quilting [26], TVM [26], HGD [47], pixel deflection [66], and Comdefend [38]. We evaluate all baseline methods on the three vision tasks, except that the randomized resizing & padding is omitted in semantic segmentation and object detection tasks as it destroys the semantic structure. We directly adopt the open-source PyTorch implementation for all baseline methods. We use the same training dataset as the one used to train our method for those methods that required training. It is worth noting that all baseline methods do not require adversarial examples for training.

Evaluation Dataset and Metrics: For classification, we use the ImageNet validation set and evaluate the Top-1 classification accuracy of all competing defense approaches. For semantic segmentation and object detection, we evaluate on the

MS COCO [49], ADE20K [97] and Pascal VOC [20] datasets. The effectiveness of all methods for segmentation and detection is measured by mean Intersection over Union (mIoU) and mean Average Precision (mAP), respectively.

5.2 Main Results & Analysis

Classification: We apply a given attack method with ResNet101 and InceptionV3 to generate adversarial samples. For FGSM, BIM, and PGD, we set two different maximum perturbation levels in L_∞ distance, namely 2/255 and 4/255. The maximum number of iterations is set to 100 for both BIM and PGD. For C&W, NewtonFool, and DeepFool attacks, we generate both L_∞ distance based attacks and L_2 distance-based attacks; we choose 100 update steps for C&W and NewtonFool, and 50 for DeepFool; DeepFool requires the number of candidate classes which is set to 10 in our experiments.

The Top-1 classification accuracies of all methods are reported in Table 5.1. Our approach outperforms the baseline methods with a large margin under all experimental settings except those with DeepFool attacks. Notably, under DeepFool attacks, the differences between the best performer pixel-deflection and ours are marginal. Moreover, for PGD and BIM attacks, our defense method offers the lowest relative performance degradation when the more vigorous attack is performed (*i.e.*, maximum perturbation increases from 2/255 to 4/255). Figure 5.1 qualitatively underlines the motivation of combining the G and S operators. The G operator learns to mitigate the adversarial pattern, *i.e.*, it recovers a latent image in the presence of severe measurement uncertainty. In contrast, the S operator can faithfully reconstruct high-frequency details. Note that our method is able to generalize well to images from the ImageNet dataset, which typically depicts single objects, although it is trained on the Zurich-Raw-to-RGB dataset, consisting of street scenes.

Semantic Segmentation: In this task, we conduct experiments with two different types of attacks: commonly used adversarial attacks and attacks specially designed for attacking semantic segmentation models. For the former, FGSM, BIM, and PGD are used, and we employ DAG [88], a dedicated semantic segmentation attack, for the latter. All attacks are based on pre-trained DeepLabV3 models [11]. Two different maximum perturbation levels in L_∞ are used (*i.e.*, 2/255 and 4/255). The corresponding experimental results are reported in Table 5.2 and 5.3. The proposed approach significantly outperforms all baseline methods in all experimental settings. Note that no additional training is required to apply the proposed approach, validating the generalization capabilities of the method.

2D Object Detection: The experimental settings are the same as the ones used for semantic segmentation experiments, except that we use a pre-trained Faster R-CNN [68]. We report the mAP on the Pascal VOC dataset under different experimental settings in Table 5.4. The proposed defense method offers the best defense performance in all experimental settings, indicating that our approach generalizes well to unseen tasks.

5.3 RAW Distribution Analysis

In this section, we provide additional analysis on the function of the RAW distribution as an intermediate mapping space. Fundamentally, we share the motivation from existing work that successfully exploits RAW data for imaging and vision tasks, including end-to-end image processing and camera design [16, 81]. RGB images are generated by processing RAW sensor measurements with an image processing pipeline. This process removes statistical information embedded in the sensor measurements by aberrations in the optics, readout noise, color filtering, exposure, and scene illumination. While existing work directly uses RAW inputs to preserve this information, we exploit it in the form of an *empirical intermediate image distribution*.

Specifically, we devise a mapping via RAW space, thereby using RAW data to train network mapping modules, which we validate further below. As a result, we allow the method to remove adversarial patterns not only by relying on RGB image priors but also RAW image priors. We *validate the role of RAW data* in our method in Table 5.5, discussed in the following, resulting in a large Top-1 accuracy drop (*i.e.*, more than 12%), when swapping the real RAW distribution to a synthesized one. This is further corroborated in Table 5.6, which we also discuss following, where the defense breaks down from 71% to 53%, when gradually moving from RAW to RGB as intermediate image space. These experiments validate that *the "rawer" the intermediate image space is, the better the defense performs.*

Effect of Intermediate Mapping Space: We use the RAW image distribution as the intermediate mapping space in our method. We next map to other intermediate stages in the processing pipeline, such as demosaicing stage, color balance stage, and white balance stage. Specifically, we assess how using different stage values as intermediate mapping space affects the defense performance (*i.e.*, we ablate on the intermediate mapping space used). As reported in Table 5.6, we observe that the defense performance gradually decreases as we map via a less RAW intermediate space.

Real RAW Versus Synthetic RAW: We further ablate on the dataset used to train our model. Specifically, we trained F and G operators on the Zurich-Raw-to-RGB dataset, HDR-RAW-RGB [28] and MIT-RAW-RGB [22] respectively, and assess how the defense performance changes. Similar to the Zurich-Raw-to-RGB dataset, the RAW images in the HDR-RAW-RGB dataset are captured by a real camera; however, the ones offered by MIT-RAW-RGB are purely synthesized by reformatting downsampled RGB images into Bayer patterns with handcrafted Gaussian noise. We note that, as such, the MIT-RAW-RGB dataset does not include the RAW distribution cues. Experimental results are reported in Table 5.5. As observed, both RAW

	FSGM	PGD	C&W	NewtonFool	DeepFool
Zurich-Raw-to-RGB[34]	58.85	66.17	71.40	40.96	72.04
HDR-RAW-RGB[28]	55.57	64.12	71.65	42.36	70.77
MIT-RAW-RGB[22]	40.52	47.29	55.13	28.52	58.49

Table 5.5: **Quantitative Ablation Study on RAW Training Datasets.** We train F and G operators on three different RAW-RGB datasets and report the Top-1 defense accuracy on the ImageNet dataset. The RAW images in the Zurich-Raw-to-RGB and HDR-RAW-RGB are captured by real cameras, while the ones in MIT-RAW-RGB are synthesized. We see a sharp performance drop when swapping the real RAW training data to synthetic data due to the lack of natural RAW distribution cues.

	FSGM	PGD	C&W	NewtonFool	DeepFool
Raw Capture	57.33	65.02	70.86	40.65	70.23
Demosaic Stage	52.93	59.83	63.29	36.76	64.81
Color Balance Stage	48.38	55.41	57.92	33.92	59.37
White Blance Stage	47.35	54.02	56.23	33.01	57.08
contrast Improvement Stage	45.2	52.18	54.18	31.84	55.64
Agamma adjustment Stage	44.4	50.91	53.08	30.57	54.19

Table 5.6: **Effect of Different Intermediate Mapping Spaces.** We report the Top-1 adversarial defense accuracy on the ImageNet dataset when mapping to different intermediate mapping spaces that are the steps of the image processing pipeline. The performance drops as the intermediate image space moves from RAW to the RGB output space. This validates the importance of exploiting the RAW space in the proposed defense.

distributions of Zurich-Raw-to-RGB and HDR-RAW-RGB allow us to learn effective adversarial defenses, while a sharp performance degradation occurs when shifting from real RAW distribution to the synthesized one due to the lack of natural RAW distribution cues.

5.4 Robustness to Hyperparameter and Operator Deviations

Hyper-parameter ω: We introduce a hyperparameter ω for weighting the contributions of the two operators G and S. Here, we evaluate how varying values of ω affect the overall defense accuracy. As reported in Tab. 5.7, we find that, while each attack has a different optimal value of ω, the range 0.6-0.8 provides a good trade-off, and we use 0.7 in our experiments.

Deviations of Operators F and G: The operators F and G are trained separately and used jointly at inference time. We evaluate how deviations in each operator affect the overall performance in two experiments. First, we mix the operators F and G

Hyper-parameter $\omega =$	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Against FSGM Attack	64.25	64.41	64.83	65.27	65.58	65.87	65.93	**66.02**	65.75	65.53	65.39
Against C&W Attack	69.16	69.44	69.93	70.26	70.81	70.96	71.35	71.40	**72.70**	71.28	71.07
Against DeepFool Attack	69.55	69.84	71.19	71.51	71.88	72.35	**72.63**	72.04	71.75	71.69	71.04

Table 5.7: **Effect of hyperparameter ω.** We evaluate the impact of the method hyperparameter ω on the effectiveness of the proposed defense method.

	F-300	F-320	F-340	F-360	F-380	F-400
G-300	66.02	66.08	65.93	66.05	66.03	66.05
G-330	66.11	66.04	65.97	66.01	65.99	66.04
G-360	65.98	66.04	66.02	65.89	66.08	69.01
G-390	65.98	65.95	66.00	66.04	65.99	65.94

Table 5.8: **Robustness to Deviations of F and G.** We evaluate the defense accuracy when mixing operator from different training epochs.

Gausian Noise σ	0 (no noise)	0.01	0.05	0.1	0.3	0.5
Against FSGM Attack	66.02	66.01	65.98	65.90	65.73	65.64
Against PGD Attack	68.34	68.30	68.22	68.10	68.03	67.83

Table 5.9: **Robustness to Deviations of F.** We perturb the output of operator F with Gaussian noise of different standard deviations and report the defense accuracy.

from different training checkpoints and evaluate the effect on the defense accuracy. Tab. 5.8 reports that the checkpoint combinations do not result in a failure but only slight deviations of the defense performance. Second, we add varying levels of Gaussian noise $G(0, \sigma)$ to the output of operator F and evaluate how such deviation affects the following steps and the overall defense accuracy. Tab. 5.9 reports that the perturbations are not amplified in the following steps, and the defense accuracy only fluctuates slightly. The experiments show that the ISP operators G and S themselves are robust to slight deviation in each component.

5.5 Super White-box Attack

When evaluating the performance of the defense algorithm in a white-box setting, most defense papers assume the adversary has full access to the target model (i.e. ResNet classification model), yet not to the preprocessing module used to transform the input. This section considers the case where the preprocessing model is exposed and allows the adversary to calculate the gradient for the differentiable part of the

	Super-FSGM		Super-PGD		Super-BIM		Super-DeepFool		Super-C&W		Super-NewtonFool
	2/255 ↑	4/255 ↑	2/255 ↑	4/255 ↑	2/255 ↑	4/255 ↑	$L_\infty \uparrow$	$L_2 \uparrow$	$L_\infty \uparrow$	$L_2 \uparrow$	$L_\infty \uparrow$
ComDefend[38]	7.43	5.12	1.33	0.80	1.54	0.82	1.83	1.70	2.47	1.02	0.45
Proposed method w/ G only	11.97	7.82	3.68	1.83	4.02	2.35	3.27	2.11	4.93	1.36	0.83
Proposed Method	**62.88**	**55.25**	**65.05**	**63.97**	**64.49**	**60.56**	**68.62**	**59.93**	**68.74**	**65.13**	**38.41**

Table 5.10: **Against the super-white attack**: We evaluate the defense performance against a super-white attack setting, where we assume the preprocessing model is exposed and the adversary can calculate the gradient for the differentiable part of the preprocessing model to construct stronger adversarial attacks. In such a setting, the differentiable preprocessing module (*i.e.*, ComDefend and G only) fail while our method does not collapse, demonstrating the benefit of exploiting the conventional ISP operator S in our method.

preprocessing model to construct strong adversarial attacks. We refer to these attacks as super-white attacks.

We conduct a super-white attack experiment to provide insights into the benefits of the proposed non-differentiable S operator. A white-box setting means the attack methods have full access to a target model, including network architecture, learned weights, and even training data, yet not to the preprocessing module used to transform input. In contrast, in the super-white box attack setting, both preprocessing modules and target models are exposed, allowing the adversary to calculate the gradient for adversarial attacks. In Table 5.10, we demonstrate the defense performances of three competing approaches on the ImageNet in Top-1 accuracy under the super-white box attack setting. We see that the defense methods with differentiable preprocessing modules (*i.e.*, ComDefend and the proposed method with G only) cannot function effectively under the super-white box attack setting. The attack only impacts the proposed method marginally, validating the benefit of the non-differentiable S operator.

Chapter 6

Conclusion and future works

We exploit RAW image data as an empirical latent space in the proposed adversarial defense method. Departing from existing defense methods that aim to directly map an adversarially perturbed image to the closest benign image, we exploit large-scale natural image datasets as an empirical prior for sensor captures – before they end up in existing datasets after their transformation through conventional image processing pipelines. This empirical prior allows us to rely on low-level image processing pipelines to design the mappings between the benign and perturbed image distributions. We extensively validate the effectiveness of the method on existing classification and segmentation datasets. The method is entirely model-agnostic, requires no adversarial examples to train, and acts as an off-the-shelf preprocessing module that can be transferred to diverse vision tasks. We also provide insight into the working principles of the approach, confirming the role of the RAW image space in the proposed method. In the future, we plan to explore RAW natural image statistics as an unsupervised prior for image reconstruction and generative neural rendering tasks.

Bibliography

[1] A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR, 2018.

[2] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok. Synthesizing robust adversarial examples. In *International conference on machine learning*, pages 284–293. PMLR, 2018.

[3] Y. Bahat, M. Irani, and G. Shakhnarovich. Natural and adversarial error detection using invariance to image transformations. *arXiv preprint arXiv:1902.00236*, 2019.

[4] T. Borkar, F. Heide, and L. Karam. Defending against universal attacks through selective feature regeneration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 709–719, 2020.

[5] W. Brendel, J. Rauber, and M. Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *International Conference on Learning Representations*, 2018.

[6] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, 2017.

[7] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks, 2017.

[8] C. Chen, Q. Chen, M. N. Do, and V. Koltun. Seeing motion in the dark. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 3184–3193. IEEE, 2019.

[9] C. Chen, Q. Chen, J. Xu, and V. Koltun. Learning to see in the dark. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 3291–3300. Computer Vision Foundation / IEEE Computer Society, 2018.

[10] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[11] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation, 2017.

[12] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 15–26, 2017.

[13] M. Cheng, T. Le, P.-Y. Chen, J. Yi, H. Zhang, and C.-J. Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. *arXiv preprint arXiv:1807.04457*, 2018.

[14] L. Dai, X. Liu, C. Li, and J. Chen. Awnet: Attentive wavelet network for image isp. In *ECCV Workshops*, 2020.

[15] N. Das, M. Shanbhogue, S.-T. Chen, F. Hohman, L. Chen, M. E. Kounavis, and D. H. Chau. Keeping the bad guys out: Protecting and vaccinating deep learning with jpeg compression. *arXiv preprint arXiv:1705.02900*, 2017.

[16] S. Diamond, V. Sitzmann, F. Julca-Aguilar, S. Boyd, G. Wetzstein, and F. Heide. Dirty pixels: Towards end-to-end image processing and perception. *ACM Transactions on Graphics (SIGGRAPH)*, 2021.

[17] R. Duan, X. Ma, Y. Wang, J. Bailey, A. K. Qin, and Y. Yang. Adversarial camouflage: Hiding physical-world attacks with natural styles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1000–1008, 2020.

[18] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy. A study of the effect of jpg compression on adversarial images, 2016.

[19] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy. A study of the effect of JPG compression on adversarial images. *CoRR*, abs/1608.00853, 2016.

[20] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[21] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1625–1634, 2018.

[22] M. Gharbi, G. Chaurasia, S. Paris, and F. Durand. Deep joint demosaicking and denoising. *ACM Transactions on Graphics (TOG)*, 35(6):191, 2016.

[23] C. Gong, T. Ren, M. Ye, and Q. Liu. Maxup: Lightweight adversarial training with data augmentation improves neural network training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2474–2483, June 2021.

[24] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2015.

[25] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples, 2015.

[26] C. Guo, M. Rana, M. Cisse, and L. Van Der Maaten. Countering adversarial images using input transformations. *ICLR*, 2018.

[27] M. Guo, Y. Yang, R. Xu, Z. Liu, and D. Lin. When nas meets robustness: In search of robust architectures against adversarial attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 631–640, 2020.

[28] S. W. Hasinoff, D. Sharlet, R. Geiss, A. Adams, J. T. Barron, F. Kainz, J. Chen, and M. Levoy. Burst photography for high dynamic range and low-light imaging on mobile cameras. *ACM Transactions on Graphics (ToG)*, 35(6):1–12, 2016.

[29] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[30] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[31] W. Hu and Y. Tan. Generating adversarial malware examples for black-box attacks based on gan. *ArXiv*, abs/1702.05983, 2017.

[32] S.-C. Huang, F.-C. Cheng, and Y.-S. Chiu. Efficient contrast enhancement using adaptive gamma correction with weighting distribution. *IEEE transactions on image processing*, 22(3):1032–1041, 2012.

[33] A. Ignatov, L. V. Gool, and R. Timofte. Replacing mobile camera isp with a single deep learning model, 2020.

[34] A. D. Ignatov, L. V. Gool, and R. Timofte. Replacing mobile camera isp with a single deep learning model. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2275–2285, 2020.

[35] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[36] S. T. Jan, J. Messou, Y.-C. Lin, J.-B. Huang, and G. Wang. Connecting the digital and physical world: Improving the robustness of adversarial attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 962–969, 2019.

[37] U. Jang, X. Wu, and S. Jha. Objective metrics and gradient descent algorithms for adversarial examples in machine learning. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, pages 262–277, 2017.

[38] X. Jia, X. Wei, X. Cao, and H. Foroosh. Comdefend: An efficient image compression model to defend adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6084–6092, 2019.

[39] H. C. Karaimer and M. S. Brown. A software platform for manipulating the camera imaging pipeline. In *ECCV*, 2016.

[40] H. C. Karaimer and M. S. Brown. A software platform for manipulating the camera imaging pipeline. In *European Conference on Computer Vision*, pages 429–444. Springer, 2016.

[41] H. Kim. Torchattacks: A pytorch repository for adversarial attacks, 2021.

[42] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[43] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world, 2017.

[44] Y. Li, L. Li, L. Wang, T. Zhang, and B. Gong. Nattack: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks. *arXiv preprint arXiv:1905.00441*, 2019.

[45] Z. Liang, J. Cai, Z. Cao, and L. Zhang. Cameranet: A two-stage framework for effective camera isp learning. *IEEE Transactions on Image Processing*, 30:2248–2262, 2021.

[46] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1778–1787, 2018.

[47] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1778–1787, 2018.

[48] N. Limare, J.-L. Lisani, J.-M. Morel, A. B. Petro, and C. Sbert. Simplest color balance. *Image Processing On Line*, 1:297–315, 2011.

[49] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. Microsoft coco: Common objects in context, 2015.

[50] Z. Liu, Q. Liu, T. Liu, Y. Wang, and W. Wen. Feature Distillation: DNN-oriented JPEG compression against adversarial examples. *International Joint Conference on Artificial Intelligence*, 2018.

[51] Z. Liu, Q. Liu, T. Liu, N. Xu, X. Lin, Y. Wang, and W. Wen. Feature distillation: Dnn-oriented jpeg compression against adversarial examples. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 860–868. IEEE, 2019.

[52] J. Lu, T. Issaranon, and D. Forsyth. Safetynet: Detecting and rejecting adversarial examples robustly. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 446–454, 2017.

[53] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[54] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks, 2019.

[55] D. Menon, S. Andriani, and G. Calvagno. Demosaicing with directional filtering and a posteriori decision. *IEEE Transactions on Image Processing*, 16(1):132–141, 2006.

[56] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks, 2016.

[57] A. Mosleh, A. Sharma, E. Onzon, F. Mannan, N. Robidoux, and F. Heide. Hardware-in-the-loop end-to-end optimization of camera image processing pipelines. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[58] P. Nakkiran. Adversarial robustness may be at odds with simplicity. *arXiv preprint arXiv:1901.00532*, 2019.

[59] N. Narodytska and S. Kasiviswanathan. Simple black-box adversarial attacks on deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1310–1318, 2017.

[60] T. Pang, K. Xu, Y. Dong, C. Du, N. Chen, and J. Zhu. Rethinking softmax cross-entropy loss for adversarial robustness. *ICLR*, 2020.

[61] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, page 506–519, New York, NY, USA, 2017. Association for Computing Machinery.

[62] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE, 2016.

[63] N. Papernot, P. D. McDaniel, and I. J. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *CoRR*, abs/1605.07277, 2016.

[64] B. Phan, F. Mannan, and F. Heide. Adversarial imaging pipelines. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16051–16061, 2021.

[65] O. Poursaeed, I. Katsman, B. Gao, and S. Belongie. Generative adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4422–4431, 2018.

[66] A. Prakash, N. Moran, S. Garber, A. DiLillo, and J. Storer. Deflecting adversarial attacks with pixel deflection. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018.

[67] J. Rauber, W. Brendel, and M. Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models, 2018.

[68] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

[69] A. M. Reza. Realization of the contrast limited adaptive histogram equalization (clahe) for real-time image enhancement. *Journal of VLSI signal processing systems for signal, image and video technology*, 38(1):35–44, 2004.

[70] A. Rizzi, C. Gatta, and D. Marini. Color correction between gray world and white patch. In *Human Vision and Electronic Imaging VII*, volume 4662, pages 367–375. International Society for Optics and Photonics, 2002.

[71] P. Samangouei, M. Kabkab, and R. Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *ICLR*, 2018.

[72] E. Schwartz, R. Giryes, and A. M. Bronstein. Deepisp: Toward learning an end-to-end image processing pipeline. 28(2):912–923, Feb. 2019.

[73] S. Sen, B. Ravindran, and A. Raghunathan. Empir: Ensembles of mixed precision deep networks for increased robustness against adversarial attacks. *ICLR*, 2020.

[74] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein. Adversarial training for free! In *Proceedings of the*

33rd International Conference on Neural Information Processing Systems, pages 3358–3369, 2019.

[75] Y. Shi, S. Wang, and Y. Han. Curls & whey: Boosting black-box adversarial attacks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6512–6520, 2019.

[76] D. Stutz, M. Hein, and B. Schiele. Confidence-calibrated adversarial training: Generalizing to unseen attacks. In *International Conference on Machine Learning*, pages 9155–9166. PMLR, 2020.

[77] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[78] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks, 2014.

[79] E. Tseng, A. Mosleh, F. Mannan, K. St-Arnaud, A. Sharma, Y. Peng, A. Braun, D. Nowrouzezahrai, J.-F. Lalonde, and F. Heide. Differentiable compound optics and processing pipeline optimization for end-to-end camera design. *ACM Transactions on Graphics (TOG)*, 40(4), 2021.

[80] E. Tseng, F. Yu, Y. Yang, F. Mannan, K. S. Arnaud, D. Nowrouzezahrai, J.-F. Lalonde, and F. Heide. Hyperparameter optimization in black-box image processing using differentiable proxies. 38(4), July 2019.

[81] E. Tseng, F. Yu, Y. Yang, F. Mannan, K. S. Arnaud, D. Nowrouzezahrai, J.-F. Lalonde, and F. Heide. Hyperparameter optimization in black-box image processing using differentiable proxies. *ACM Trans. Graph.*, 38(4):27–1, 2019.

[82] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*, number 2019, 2019.

[83] C.-C. Tu, P. Ting, P.-Y. Chen, S. Liu, H. Zhang, J. Yi, C.-J. Hsieh, and S.-M. Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 742–749, 2019.

[84] J. Wang and H. Zhang. Bilateral adversarial training: Towards fast training of more robust models against adversarial attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6629–6638, 2019.

[85] E. Wong, L. Rice, and J. Z. Kolter. Fast is better than free: Revisiting adversarial training. *ICLR*, 2020.

[86] Y.-H. Wu, C.-H. Yuan, and S.-H. Wu. Adversarial robustness via runtime masking and cleansing. In *International Conference on Machine Learning*, pages 10399–10409. PMLR, 2020.

[87] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991*, 2017.

[88] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille. Adversarial examples for semantic segmentation and object detection, 2017.

[89] C. Xie, Y. Wu, L. v. d. Maaten, A. L. Yuille, and K. He. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 501–509, 2019.

[90] X. Xu, Y. Ma, and W. Sun. Towards real scene super-resolution with raw images. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1723–1731, 2019.

[91] X. Xu, Y. Ma, W. Sun, and M.-H. Yang. Exploiting raw images for real-scene super-resolution. *arXiv preprint arXiv:2102.01579*, 2021.

[92] X. Yin, S. Kolouri, and G. K. Rohde. Gat: Generative adversarial training for adversarial example detection and robust classification. In *International Conference on Learning Representations*, 2019.

[93] K. Yu, Z. Li, Y. Peng, C. C. Loy, and J. Gu. Reconfigisp: Reconfigurable camera image processing pipeline. *ArXiv*, abs/2109.04760, 2021.

[94] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.

[95] X. Zhang, Q. Chen, R. Ng, and V. Koltun. Zoom to learn, learn to zoom. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3762–3770, 2019.

[96] H. Zheng, Z. Zhang, J. Gu, H. Lee, and A. Prakash. Efficient adversarial training with transferable adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1181–1190, 2020.

[97] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017.

Appendix A

Additional Experiment Results

A.1 Additional Qualitative Results

In this section, we provide additional qualitative results from the proposed methods and state-of-the-art defense baselines. For input-transformation-based adversarial defense methods, a balance between transformation fidelity and adversarial pattern removal is essential. A higher transformation fidelity indicates more details of the original image are kept, including adversarial patterns. In contrast, radical pattern removal could result in a severe loss of detail. This balance is achieved by a simple weighted sum between the G and S operators in our method. The S operator is designed for faithfully reconstructing high-frequency details, while the G operator is trained to mitigate adversarial patterns. As shown in Figure A.1, our approach offers better transformation fidelity than Comdefend, TVM, Pixel Deflection, and Resizing & Padding methods. Even though the JPEG-Defense method keeps more high-frequency details than the ones offered by our approach, it suffers from removing adversarial patterns, resulting in the worst performance among all compared methods.

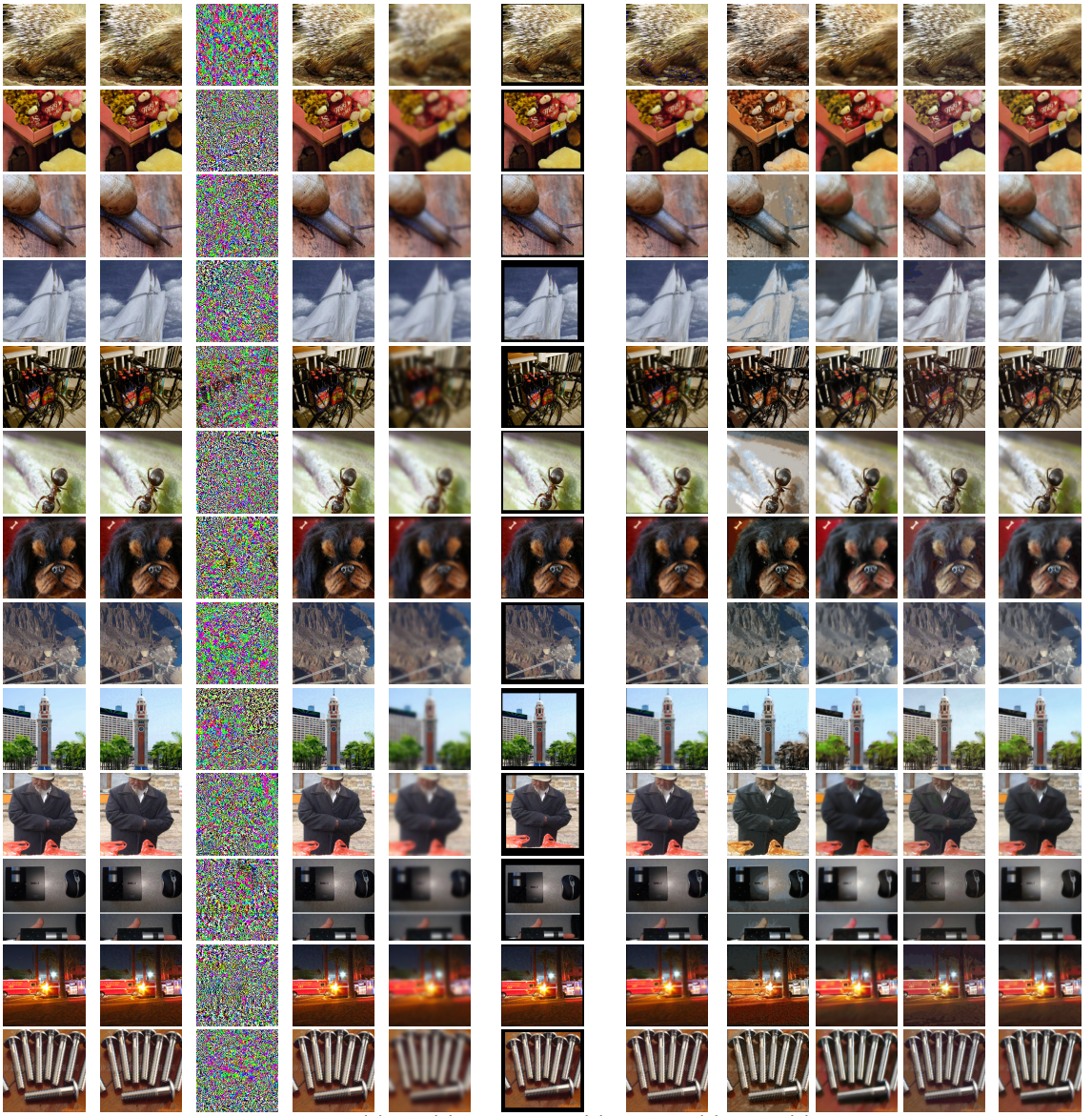


Figure A.1: **Additional Qualitative Comparison:** Qualitative comparison of the proposed method along with both G and S operators against state-of-the-art defense methods on the ImageNet dataset, see text.

A.2 Against BPDA Attack

In this section, we discuss the modifications of the method to defend against BPDA attack. BPDA and derivative methods rely on building a differentiable proxy for a non-differentiable input transformation method (*i.e.*, obfuscated gradients) to approximate gradients. In order to tackle such an attack, we modified our transformation algorithm from a determined algorithm to a randomized algorithm so that the adversary cannot build an accurate proxy.

The modifications are shown listed in Algorithm 2; see original Algorithm 1. Specifically, at the inference time, instead of transforming the input image and directly feeding it to the downstream model, we apply our method N times sequentially, where the output of each transformation is the input for the next one. We add small random noise perturbations to the RGB image and Raw Image during each transformation. Here, two randomization effects are introduced: the number of times of applying transformation, N, is also random. By introducing sufficient randomization, it is hard to build an accurate proxy to get the precise gradient, and then we can defend against the BPDA attack. Note that the new algorithm only requires modification at the inference stage, whereas the training stage remains the same, hence there is no need to retrain the model.

Algorithm 1 Original Inference Algorithm

Require: Input Image I, Operator S, F, G, Hyper-parameter ω

$Raw \leftarrow F(I)$

$I \leftarrow \omega * G(Raw) + (1 - \omega) * S(Raw)$

Algorithm 2 Modified Inference Algorithm

Require: Input Image I, Operator S, F, G, Hyper-parameter ω

$N \leftarrow random(0, 20)$

$i \leftarrow 0$

while $i \neq N$ **do**

    $n_I \leftarrow Gaussian(\mu = 0, \sigma = 0.05)$

    $n_{Raw} \leftarrow Gaussian(\mu = 0, \sigma = 0.05)$

    $Raw \leftarrow F(I + n_I)$

    $Raw \leftarrow Raw + n_{Raw}$

    $I \leftarrow \omega * G(Raw) + (1 - \omega) * S(Raw)$

    $i \leftarrow i + 1$

end while

Defense Method	JPEG-Defense	TVM	Randomized Resizing & Padding	Pixel-Deflection
	0.08	6.39	2.66	1.87
	ComDefend	HGD	Proposed Method (Unmodified)	Proposed Method (modified)
	0.03	0.03	14.28	38.85

Table A.1: **Quantitative comparison with baseline input-transformation defense methods under BPDA attack:** The proposed modification improves the defending accuracy from 14.28 to 38.85, and the resulting method significantly outperforms the baselines.

	FSGM		PGD		BIM		DeepFool		C&W		NewtonFool	BPDA
	2/255 ↑	4/255 ↑	2/255 ↑	4/255 ↑	2/255 ↑	4/255 ↑	$L_\infty \uparrow$	$L_2 \uparrow$	$L_\infty \uparrow$	$L_2 \uparrow$	$L_\infty \uparrow$	$L_\infty \uparrow$
ResNet-101												
JPEG-Defense[19]	33.14	20.71	45.19	21.74	36.78	8.5	53.16	45.69	59.06	52.01	24.65	0.08
TVM[26]	43.75	40.02	45.46	44.35	44.86	41.93	47.69	39.89	45.51	40.44	22.6	6.39
Randomized Resizing & Padding[87]	45.21	34.97	45.38	27.75	40.04	18.04	73.06	62.47	66.53	59.87	27.93	2.66
HGD [47]	54.75	43.85	55.26	50.05	56.74	48.61	64.34	58.13	59.98	52.88	27.70	0.03
Pixel-Deflection[66]	54.56	35.14	60.68	34.86	58.71	41.91	75.97	64.13	66.29	60.91	28.81	1.87
ComDefend[38]	48.21	36.51	53.28	48.38	51.39	42.01	63.68	55.62	58.53	50.38	26.46	0.03
Proposed Method w/o modification	66.02	58.85	68.34	66.17	66.91	63.01	72.04	63.52	71.40	67.33	40.96	14.28
Proposed Method w/ modification	61.87	54.42	63.79	59.03	61.29	57.35	68.80	56.42	68.13	61.65	35.88	38.85
InceptionV3												
JPEG-Defense[19]	31.97	20.25	43.34	21.15	34.68	8.55	51.20	43.49	55.00	50.39	24.06	0.12
TVM[26]	42.47	37.23	42.75	41.61	42.80	39.71	45.21	37.39	43.27	37.51	23.05	4.58
Randomized Resizing & Padding[87]	41.86	34.49	43.41	25.60	39.42	16.62	70.24	58.65	63.24	55.62	27.55	2.09
HGD [47]	52.83	40.99	50.35	47.62	56.02	47.78	60.33	56.61	59.55	52.0	26.84	0.03
Pixel-Deflection[66]	51.42	34.27	56.13	32.49	56.18	39.13	71.16	61.58	61.94	57.58	28.01	1.56
ComDefend[38]	47.00	35.34	49.99	46.15	48.74	39.58	60.01	52.47	55.85	47.70	25.44	0.03
Proposed Method w/o modification	63.03	56.34	65.69	63.03	64.77	59.49	69.25	60.04	66.97	64.69	38.01	12.11
Proposed Method w/ modification	59.89	53.02	61.15	59.44	58.07	57.92	66.48	55.84	61.25	60.30	35.63	36.43

Table A.2: **Quantitative Comparisons for the proposed method w/ and w/o modification for BPDA.** The modification significantly increases the robustness of the proposed method under BPDA attack, at the cost of slightly decreasing the defense performance under other attacks. The relative ranking of the proposed defense method versus baseline defense methods remains the same, with the proposed defense method outperforming baselines with a large margin both before and after modification.

The performance of the defense against BPDA is shown in Table A.1. Specifically, the proposed modification improves the defending accuracy from 14.28 to 38.85. Also, compared to the baseline input transformation methods, the proposed method is the only method that shows robustness to the BPDA attack. We also compared the performance of the proposed method with and without modification for all other attacks in the same Table A.2, and found that the defense performances under other attacks decrease slightly after introducing the modification. In other words, the modification significantly increases the robustness of the proposed method under the BPDA attack, at the cost of slightly decreasing the defense performance under other attacks. Note that the relative ranking of the proposed defense method versus baseline defense methods remains the same, with the proposed defense method outperforming baselines with a large margin both before and after the modification.

Appendix B

Attack and Defense Algorithms

B.1 Attack Algorithm Details

In this section, we provide formal definitions for the attack algorithms which are used for the validation of the proposed defense method.

1. **FSGM:** The FSGM [25] is an attack method that, for each pixel, determines in which direction the pixel intensity should be moved according to the gradient of the loss function. Mathematically, the adversarial perturbation δ is defined by FGSM as:

$$\delta = \epsilon \cdot \text{sign}(\nabla_x L(x, y)),$$

   where ϵ is the maximum perturbation allowed for an attack, which is commonly sufficiently small for undetectable; x and y are a benign image and the corresponding ground truth label, respectively.

2. **BIM:** The BIM [43] attack is an enhanced version of FGSM. Instead of taking one single step of size ϵ, it iteratively searches for an optimal perturbation with

multiple smaller steps α. Then, the obtained perturbation is clipped by the predefined ϵ. Formally, BIM is defined as

$$x_i = \text{clip}_{\epsilon,x}(x_{i-1} + \alpha \cdot \text{sign}(\nabla_x L(x, y))),$$

where $x_0 = x$. The number of iterations used is a hyperparameter, and more iterations typically lead to a stronger attack.

3. **PGD:** Similar to the BIM attack, the PGD [54] attack interatively optimizes the adversarial examples generated. With the initial $x_0 = x$, the method iteratively find x_i by:

$$x^i = \Pi_{x+S}(x^{i-1} + \alpha \cdot \text{sign}(\nabla_x L(f_\theta(x^{i-1}), y))), \tag{B.1}$$

where Π_{x+S} denotes projecting perturbations into the set S. We refer the reader to [54] for additional details. The scalar α is the step size used in each iteration and ϵ denotes the maximum perturbation allowed.

4. **DeepFool:** Similar to BIM and PGD, the DeepFool [56] attack generates adversarial perturbations iteratively, intending to obtain the perturbation with minimal distortion. Specifically, it finds the closest decision boundary by multiple linearizations of the classifier and then generates the perturbations based on the decision boundary.

5. **C&W:** The C&W [7] attack is an optimization-based attack method, which generates the adversarial perturbations by solving the following problem

$$\begin{aligned} \min \quad & ||\delta||_p + c \cdot f(x+\delta) \\ s.t. \quad & x + \delta \in [0,1]^n \end{aligned} \tag{B.2}$$

where f is the objective function that is designed to mislead the example x to be misclassified; $||\cdot||_p$ denotes l_p norm; and c is a constant, estimated by binary search.

6. **NewtonFool:** NewtonFool [37] attack is essentially a variant of DeepFool attack. In contrast to DeepFool generating perturbations based on linearization approximation, the NewtonFool is based on nonlinear constraints, allowing for a significantly faster generation of adversarial patterns. Hence, with the same time budget, more iterations can be performed, resulting in stronger attacks.

7. **DAG:** The DAG [88] attack is designed to attack semantic segmentation and object detection models. Essentially, it is similar to optimization-based classification attacks, generating adversarial perturbation by solving the following problem:

$$\forall n, \arg\max_c \{f_c(\mathbf{X} + \mathbf{r}, t_n)\} \neq l_n, \tag{B.3}$$

where X is an image which contains N recognition targets $\mathcal{T} = \{t_1, t_2, \ldots, t_N\}$; $\mathcal{L} = \{l_1, l_2, \ldots, l_n\}$ is the ground-truth class labels of $\mathcal{T}$, *i.e.*, l_n is the ground-truth label of t_n; $\mathbf{f}(\mathbf{X}, t_n) \in \mathbb{R}^C$ denotes the classification score vector on the n-th recognition target; $\mathbf{r}$ denotes an adversarial perturbation to be estimated. The form of $\mathcal{T}$ is defined based on a specific task. For the image classification task, $\mathcal{T}$ only contains one element, *i.e.*, the entire image. $\mathcal{T}$ becomes all pixels for semantic segmentation tasks, while it becomes all proposals for object detection tasks.

Intuitively, the goal of the Eq. B.3 is to make the predictions of all targets incorrect by estimating an adversarial perturbation $\mathbf{r}$. In doing so, adversarial labels $\mathcal{L}' = \{l'_1, l'_2, \ldots, l'_n\} \quad \forall i, l_i \neq l'_i$ are required, which are generated by randomly sampling from other incorrect classes. Under this setting, the loss function covering all targets can be written as

$$L(\mathbf{X}, \mathcal{T}, \mathcal{L}, \mathcal{L}') = \sum_{n=1}^{N} \big[f_{l_n}(\mathbf{X}, t_n) - f_{l'_n}(\mathbf{X}, t_n) \big]. \tag{B.4}$$

Minimizing L can be achieved by making every target an incorrect prediction, *i.e.*, suppressing the confidence of the original correct class $f_{l_n}(\mathbf{X} + \mathrm{r}, t_n)$, while increasing the confidence of desired (adversarial) incorrect class $f_{l'_n}(\mathbf{X} + \mathrm{r}, t_n))$.

B.2 Defense Algorithm Details

In this section, we provide additional detail on the baseline adversarial defense methods used in our experiment. These baseline methods are state-of-the-art input-transformation defense methods with released code.

1. **JPEG-Defense[19]:** The JPEG-Defense method is a defense method based on input transformation. The intuition behind the method is that the compression process may remove high-frequency adversarial perturbations. A hyper-parameter of this method is the compression ratio, which affects the adversarial defense accuracy. A high compression ratio results in high image quality but also leads to less removal of adversarial perturbations. In our experiment, we set the compression ratio to 75%, which is consistent with the setting proposed in [19].

2. **ComDefend[38]:** Similar to JPEG-Defense, ComDefend also aims to remove unnecessary details of the input image through compression. But unlike JPEG

compression, ComDefend learns an auto-encoder to compress and reconstruct the image. Specifically, an encoder maps the input image to a compressed latent space, which is then fed into the decoder for reconstruction.

3. **TVM[26]:** The method works by randomly selecting a small set of pixels and reconstructing the "simplest" image that is consistent with the selected pixels. The reconstructed image is intended to be "simple" so that it does not contain adversarial perturbations. The "simple" is defined in terms of the total variation. A hyperparameter of this method is the number of pixels to be selected. Selecting a large portion of pixels keeps more image details but also keeps more adversarial perturbations. In our experiment, we tune the hyperparameter to achieve the best performance.

4. **Randomized Resizing & Padding [87]:** The method aims to mitigate the effect of adversarial perturbations through randomization. Given an input image, two randomization operations are performed. In particular, the method first randomly resizes the input image. Then, it randomly pads zeros around the input images.

5. **Pixel-Deflection[66]:** The method works by forcing the image to match natural image statistics. In doing so, the algorithm locally corrupts the image by redistributing pixel values by pixel deflection. Then, a subsequent wavelet-based denoising operation softens this corruption.

Appendix C

Implementation Details

In this section, we provide additional implementation details for the S and F operators.

C.1 Implementation Details for the S operator

The S operator offers the functionality of a conventional (hardware) ISP pipeline using a sequence of cascaded sub-modules. In particular, the proposed S operator consists of the following components: Bayer Demosaicking, color balancing, white balancing, contrast enhancement, Gamma adjustment, and colorspace conversion sub-modules. We provide the implementation details of each sub-module below:

1. **Bayer Demosaicking**: As discussed in the main manuscript, a color filter array (CFA) sits on a matrix of small potential wells. When light passes through the CFA layer, a mosaic pattern of the three stimulus RGB colors is generated, called the RAW image. A Bayer pattern mosaic is the most commonly used one, alternating R-G-G-B 2×2 superpixels. To reconstruct trichromatic intensity values from a RAW image, a demosaicking algorithm is required. In our implementation, we use the DDFaPD demosaicking algorithm [55] offered by

the Python color-demosaicing library.

2. **Color Balance** The purpose of color balancing is to recover the color characteristics of the original scene. We achieve the color balancing by using the SimpleColorBalance [48] algorithm, which is adopted from the OpenCV library.

3. **White Balance** : We further adjust the color temperatures of RGB images with a white balance method to make the color look more natural. We use a white balancing method provided by the OpenCV library, mainly based on the White Patch algorithm and Gray World algorithm [70].

4. **Contrast Enhancement** Contrast plays a critical role in separating the dark and bright areas of an image. An improvement in contrast increases this separation, making objects more distinguishable. In our pipeline, the Contrast Limited Adaptive Histogram Equalization(CLAHE) algorithm [69] is leveraged for contrast enhancement.

5. **Gamma Adjustment**: In this step, we use a non-linear Gamma correction function to adjust the image luminance (*i.e.*, brightness level). In our pipeline, we use the OpenCV's gamma correction algorithm [32].

6. **Colorspace Conversion**: Pixel values are converted to a specific colorspace (e.g., to sRGB) before compression (e.g., to jpg), storage, or further processing.

C.2 Implementation Details for the F operator

The F operator is a small learned encoder-decoder network to map an RGB image to its intermediate RAW measurements. The details of network architecture are shown below in Table C.1

Table C.1: **Architecture description of the F Operator**.

Layer	Type	In-Channel	Out-Channel	Kernel
1st	Conv+Relu	3	32	3×3
2nd	Conv+Relu	32	64	3×3
3rd	Conv+Relu	64	128	3×3
4th	UpConv+Relu	128	64	3×3
5th	UpConv+Relu	64	32	3×3
6th	UpConv+Sigmoid	32	1	3×3

Appendix D

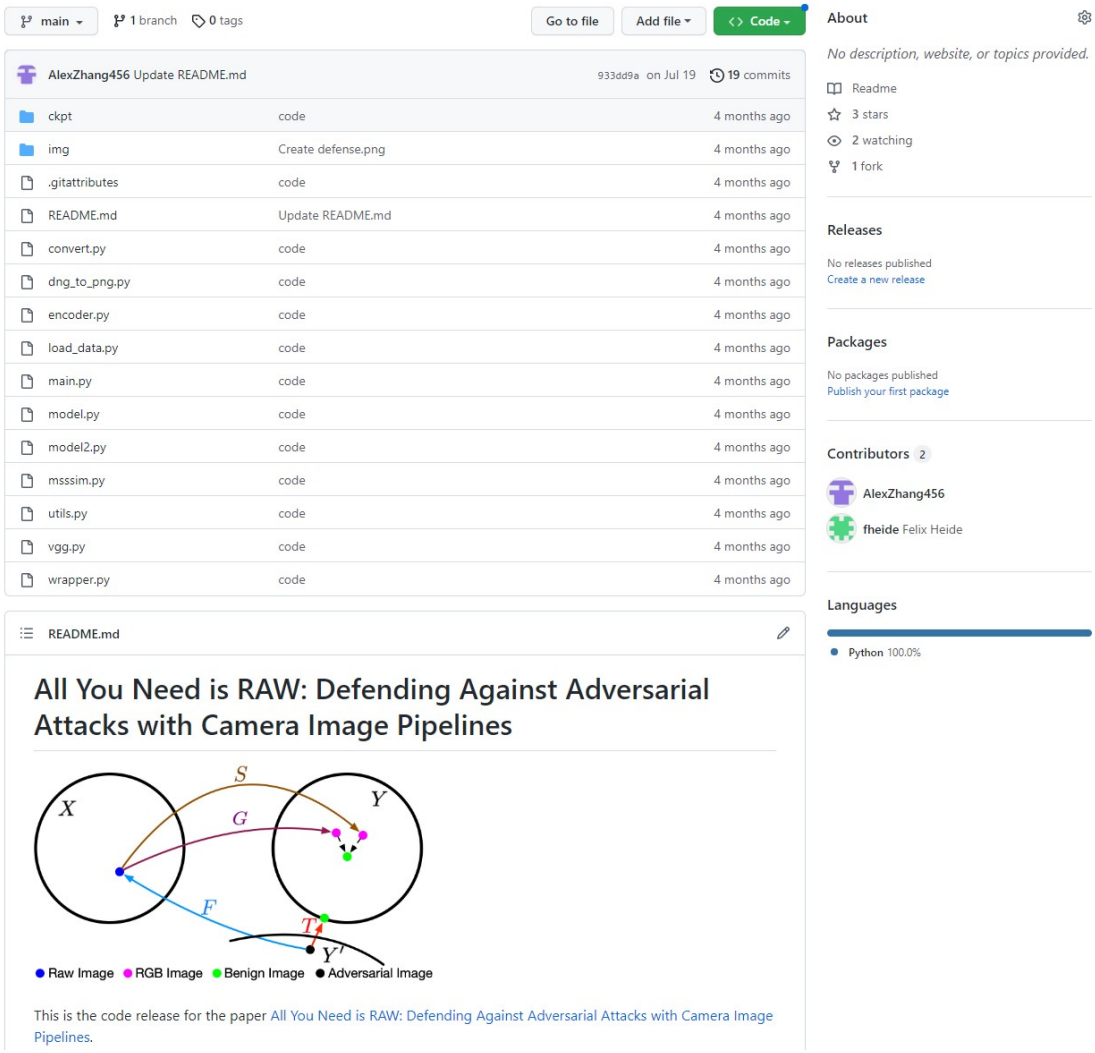
Released Codes & Instruction

In this section, we provide instructions on how to reproduce and utilize our works. The released code is available at https://github.com/princeton-computational-imaging/allyouneedisraw

D.1 Experiments Environment:

We performed our experiments using NVIDIA A100 GPU (40GB memory) Clusters. The model is implemented with the Pytorch framework, under the computing environment of Python 3.8, Pytorch 1.9.0, Torchvision 0.10.0, and CUDA11.1 toolkit. To install the environment run the following pip command:

```
pip install torch==1.9.0+cu111 torchvision==0.10.0+cu111
                                    torchaudio==0.9.0 -f https://
                                    download.pytorch.org/whl/
                                    torch_stable.html
```



D.2 Training

1. Training of mapping network from Raw to RGB: We provide two ways for training the RAW2RGB mapping network, depending on the computational resource you have access to

- For best performance, clone the Pynet training workflow https://github.com/aiff22/PyNET-PyTorch and follow the instruction to perform coarse to fine training. Note that you need to replace the model definition file (model.py) with our model.py file as we slightly modified the model structure.

- The coarse to fine training, however, is computationally heavy. We also provide a fast training flow (see train_decoder_fast method in model class). Run the following command to conduct fast training.

```
python3 main.py --command train_decoder --fast_training True --
                                    visual False --mode train
```

2. Training of mapping network from RGB to RAW: Run the following command to train the mapping network from RGB to Raw

```
python3 main.py --command train_encoder --train_encoder True --mode
                                    train
```

D.3 Evaluation

Run the following command to perform inference. Note you need to download Imagenet and specify the local path of data before running the script. If you have not trained the model by yourself, please download the pre-trained weight before running the script.

```
  python3 main.py --command eval
```

D.4 Pre-trained weight

If you want to use pre-trained weight, download them from https://drive.google.com/drive/folders/1G9ElfOO_pKLTP_EWK8Unsmg-Pzzvpaoy?usp=sharing and put it in ckpt/ folder.