Fast Video Denoising with Uncalibrated Webcams

Anqi Yang[1]*, Zhichao Yin[2], Rahul Garg[2], Xin Tong[2],
[1] Carnegie Mellon University, [2] Google

Abstract

Deep learning has advanced video denoising, yet state-of-the-art networks are compute intensive, making them unsuitable for real-time applications like video conferencing. Conversely, lightweight networks often assume known camera noise parameters limiting their use for uncalibrated webcams common in conferencing setups. This paper introduces a noise estimation and normalization framework for adapting lightweight neural denoisers to diverse webcams. We train off-the-shelf compact neural networks on synthetically added noise, simplifying denoising via input normalization with known noise parameters. During testing, we perform a one-time noise parameter estimation using initial video frames, which informs subsequent denoising. This optimization entails freezing the trained model and using gradient descent to solve for the noise parameters. We minimize the reconstruction loss, relying on estimated ground truth obtained by temporally averaging static pixels in the test video stream. We show that our method improves denoising performance of lightweight neural networks on publicly available datasets with synthetic noise and also provide examples on real data.

1. Introduction

Digital noise is a common issue in many video conferencing setups. It can be caused by low-quality or small sensors on webcams, or by poor lighting conditions. Video noise degrades the quality of the video, leading to difficulties in discerning participants and undermining the overall user experience.

Advances in deep learning have also improved video denoising algorithms [10, 14, 16, 20–23]. The state-of-the-art approaches use large neural networks trained on a wide variety of noise profiles. However, these approaches are computationally expensive and cannot be used in low-latency real-time video calls.

A common approach to simplify the problem is to train a neural network for a specific camera of which noise parameters are known, e.g., using calibration [12, 19]. While this leads to lightweight neural networks tailored to the noise profile, it does not generalize across cameras. Further noise calibration can be expensive or cumbersome. This is especially problematic for webcams, given the wide variety of webcams available and the lack of a consistent API that can provide the noise parameters of the camera.

We propose a solution that allows off-the-shelf lightweight neural networks to be trained once and then be used on a wide variety of webcams. Our method doesn't require any extra computing, except for a one-time optimization that can be done at the beginning of the video.

We leverage the observation in [19], that given a parameterized noise model and known noise parameters, one can train a lightweight neural network if we normalize the input image so that the noise statistics become invariant to the light level and additive sensor noise. While such networks can be trained on synthetic data where the noise parameters are known, the challenge lies in applying them to real-world webcams where noise parameters remain unknown. Our key insight is that the noise parameters of a webcam can be learned at test time. We exploit the fact that webcams are stationary and many pixels in a typical video stream during a conference call are static, e.g., the pixels in the background. The denoised "ground-truth" values of these static pixels can be obtained by averaging observations across the first few frames of the video. We optimize noise parameters such that the prediction of the pre-trained model aligns with the "ground-truth" values when the model input is normalized by them. Since the network and the normalization steps are differentiable, we can use gradient descent to optimize for these. The low dimensionality of noise parameters alleviates any risk of overfitting even when using a small number of frames. Moreover, our framework extends to estimating additional camera pipeline variables that affect noise statistics, such as the gamma correction factor.

We show that using our approach of normalization and test-time optimization allows various off-the-shelf lightweight neural networks to do better denoising. We perform quantitative experiments on the publicly available dataset REDS [15] using synthetic noise. Quantitative experiments show that the proposed method accurately estimates the three noise parameters, and boosts the denoising

*Work done during internship at Google.

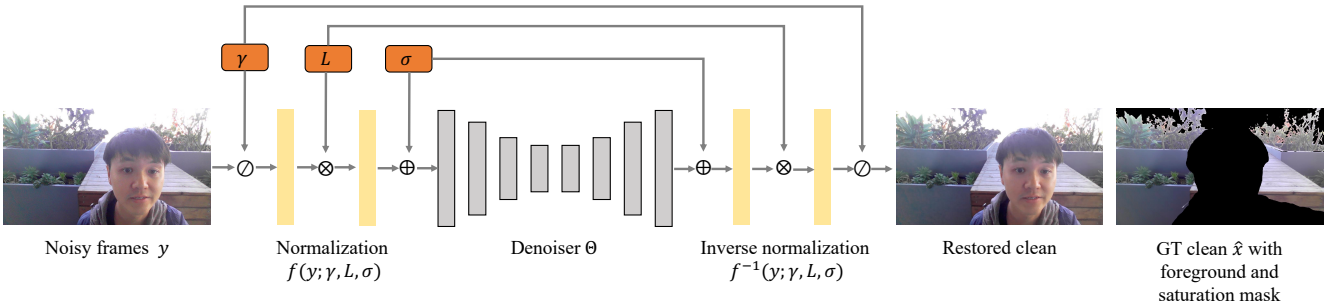


Figure 1. **Overview of the proposed noise estimation and normalization framework.** During training, we use ground-truth noise parameters γ^*, L^*, σ^* to normalize noisy frames and train a lightweight denoiser Θ. During test time, we freeze the denoiser Θ and optimize noise parameters γ, L, σ to minimize the difference between restored and ground-truth clean images in the background.

performance of the original neural network across different light levels and sensors that have different gamma corrections. We also collected several real video conferencing clips, with various webcams from on-the-market laptops, different lighting conditions, and different subjects. We show qualitative results on the collected real video dataset.

Our contributions are as follows:

- We model the noisy images using three noise parameters, light level L, sensor additive noise σ, and gamma correction coefficient γ, and apply a transformation function to normalize the effect of noise parameters.
- We propose a novel test-time noise parameter estimation method, which adapts off-the-shelf compact neural networks to denoise for various unknown webcams in video conferences.
- We conduct experiments on REDS dataset by synthesizing realistic noise. We also collect several video conferencing clips to evaluate the proposed algorithm.

2. Related work

Learning-based denoisers. With the recent advances in learning-based approaches, image and video denoising performance has seen large improvements [10, 14, 16, 20–23]. State-of-the-art (SOTA) learning-based denoisers restore high fidelity images from a wide variety of noisy images, both synthetic and real-world, but often have high computational overhead. MIRNet [23] learns enriched features through a heavy network that extracts multi-scale features, exchanges information among scales, and exploits spatial and channel attention. DEAMNet [16] proposes a maximum a posterior (MAP) method with a new adaptive consistency prior, and unrolls each iteration of solving MAP into a learning-based module. MPRNet [14] is a multi-path residual network that contains several residual concatenation block, adaptive residual blocks, and a two-fold attention module. Yang *et al*. [21] exploits diffusion model for real-world denoising by designing a linear interpolation method that guarantees the diffusion process starts with a clean image and ends with the real-world noisy image, but their method is slow due to the iterative inference in diffusion models. Liang *et al*. [10] and UFormer [20] both use transformers, and more recently, Zamir *et al*. [22] proposes a relatively efficient transformer with two novel network designs, multi-head attention and feed-forward network. However, all above-mentioned methods have a computation complexity of over 40 Gigabytes floating points operations for a 256×256 image [22] and are impractical for real-time video conferences denoising.

Lightweight denoisers. Many works design compact neural network for real-time denoising [12,19]. Maggioni *et al*. [12] exploits predictions from previous frame in videos and designs a three-stage denoising framework, each of which contains a lightweight convolution neural network. Wang *et al*. [19] is the most relevant to our work. They propose a k-Sigma transformation function to normalize the effect of different ISO settings, and train a lightweight UNet [17] with the normalized image pairs. Using this transformation, they demonstrate that the small UNet denoise as well as a much larger UNet. However, both methods require a fully calibrated sensor, RAW noisy images, and ISO values in the meta information. If deploying these methods for video conferencing, one has to calibrate a wide variety of webcams, not to mention that RAW images and metadata are not easily accessible from all webcams. Our innovation lies in developing a test-time noise estimation framework that avoids calibration, and showing that other steps of the imaging pipeline, e.g., gamma correction can be folded into the same framework.

3. Noise Modeling and Normalization

We first introduce the parameterized noise model, and how we normalize the noise statistics across images with different webcams and various lighting conditions.

Given a noisy frame y, the goal is to estimate the clean image x^*. By Gaussian-Heteroskedastic noise model [5], the noisy frame y can be formalized as

$$y = \Gamma\{g(Lx_l^* + n_{\text{shot}} + n)\}, \tag{1}$$

where Lx_l^* is the expected number of photons arrive at each sensor diode, n_{shot} is the shot noise, n is the additive sensor noise before the amplifier, g is the amplifier gain, and $\Gamma(\cdot)$ is the gamma correction function. We ignore post-amplifier readout noise, quantization, and other steps in the ISP pipeline, such as auto white balancing, demosaicing, and tone mapping for simplicity.

The expected number of photon arrivals Lx_l^* is a multiplication between the maximum photon arrival at given exposure time L and the expected clean intensity image x_l^* in linear space, where $x_l^* \in [0, 1]$ and $x^* = \Gamma(x_l^*)$. Shot noise n_{shot} is approximated as a signal-dependent Gaussian random variable [5] and follows $n_{\text{shot}} \sim \mathcal{N}(0, Lx_l^*)$. We refer to L as the light level that characterizes the scene brightness while assuming a fixed exposure time. Additive sensor noise n follows a Gaussian distribution $n \sim \mathcal{N}(0, \sigma^2)$, σ is the additive noise standard deviation in electron counts. Since front-facing webcams often have fixed exposure time and compensate for the brightness of scenes by automatically changing the gain, we let $g = \frac{1}{L}$, such that the maximum intensity of the captured frames always reaches the reference intensity of 1 [2]. We approximate the family of gamma encoding function $\Gamma(\cdot)$ with a power function with different coefficients $1/\gamma$.

We therefore simplify the noisy frame y to be

$$y = (x_l^* + \frac{n_{\text{shot}}}{L} + \frac{n}{L})^{1/\gamma}, \tag{2}$$

and the linearized noisy image y^γ follows a Gaussian distribution [5]

$$y^\gamma \sim \mathcal{N}(x_l^*, \frac{x_l^*}{L} + \frac{\sigma^2}{L^2}). \tag{3}$$

The variance changes with the scene light level L and sensor additive noise standard deviation σ. The variance of y^γ describes the noisiness of a pixel and can guide a denoiser algorithm through determining its denoising strength. Therefore, designing a neural network to denoise videos from various webcams would require a large number of parameters, so that the networks can fit various distributions. And the computation overhead is often too large for video conferencing applications. We hypothesize that a denoising neural network may implicitly estimate these three parameters during the prediction of the denoised image. Since these three parameters do not change for a given webcam in the same scene, estimating them in every frame is a waste of model capacity.

Noise normalization. According to Wang *et al*. [19], normalizing different noise profiles allows a compact neural network to denoise as well as a heavy network. Based on our noise parameterization, we apply the following mapping function to the noisy frame to normalize the effect of L, σ, γ

$$f(y) = Ly^\gamma + \sigma^2. \tag{4}$$

In this way, $f(y) \sim \mathcal{N}(Lx_l^* + \sigma^2, Lx_l^* + \sigma^2)$, which means $f(y) \sim \mathcal{N}(f(x^*), f(x^*))$. $f(y)$ is agnostic to light level, sensor additive noise, and gamma coefficient since $f(y)$ only depends on the normalized image $f(x^*)$. Neural networks that take $f(y)$ as input and predict $f(x^*)$ can be designed to be much more compact as the input follows a more constrained distribution.

4. Denoisers for uncalibrated webcams

By applying the noise normalization function, Wang *et al*. [19] trained lightweight deep neural networks to denoise for various cameras with known noise parameters. In our situation, even though the noise parameters are available in the training data, they are not in the testing data captured from uncalibrated webcams. The challenge for uncalibrated webcams is how to estimate their noise parameters during test time. In this section, we describe the lightweight denoiser training and test-time noise parameters estimation. Using the proposed framework, we are able to adapt lightweight denoisers to handle a large variety of webcams and under different light conditions.

4.1. Training lightweight denoisers

During training, we apply ground-truth L^*, σ^*, γ^* to noisy frames y and their ground-truth clean frames x^* according to Eq. 4, and train a lightweight network Θ with the normalized frames $\{f(y), f(x^*)\}$. The trained neural network $\hat{\Theta}$ is

$$\hat{\Theta} = \arg\min_{\Theta} \mathcal{L}(\Theta(f(y)), f(x^*)). \tag{5}$$

We utilize ℓ_1 as the loss function. And we train two popular lightweight backbones, ConvNeXt [11] and EfficientNet [18]. They are trained for 100,000 and 200,000 iterations correspondingly. The network is optimized by Adam [9], where $\beta_1 = 0.9$, $\beta_2 = 0.999$. We adopt an initial learning rate of 1×10^{-4} and a batch size of 32. The image crop size is 256×256.

4.2. Test-time noise adaptation

While ground-truth noise parameters are accessible during training, they are unknown in test time for uncalibrated webcams and unknown light conditions. We propose a one-time optimization of L, σ, γ for an unknown webcam in a single video session. By applying the estimated noise parameters, we are able to adapt the pre-trained lightweight

denoiser to restore videos from the same webcam in the same video session.

The basic idea is that if we can find noisy-clean image pairs from the test dataset, we can feed them to the denoising model while freezing its parameters and only optimizing the noise parameters. Even though the clean image corresponding to a given noisy image is not directly available during test time, we can create an estimated clean background image by accumulating samples from the stationary background. When the noisy parameters are optimized well enough, the restored clean background is the most similar to the estimated clean background.

Estimate one clean frame. In most video conferences, the webcams are stationary and the background scene behind the person stays static across frames. Since the noise parameters are shared across the background and the foreground, we can leverage the estimation from the background to denoise the entire frame and all subsequent frames in the entire video session. To estimate a clean background, we first segment out the background regions using off-the-shelf person segmenters, produce a background mask for each frame, and take the intersection of all background masks from the first N frames. We also remove saturated pixels with an intensity over 0.95 as saturation breaks the noise model in Equation 1. We denote the intersected mask from all frames as m. When computing the clean background, we first linearize all N noisy frames by inverse gamma correcting them with a coefficient of 2.2, average the linearized noisy frames, and then apply gamma correction to the averaged frame. Note that we assume the noise to be signal independent here, as done in [7], and show empirically that this does not affect the estimation of noise parameters. The resulting clean frame is denoted as $\hat{x}$ and is used as the supervision in noise parameter estimation. Note that our algorithm typically requires that the camera is steady for the first 50 frames, which is equivalent to $\sim$2 seconds for a 24 frame-per-second video.

Noise parameter optimization. As shown in Figure 1, during test time, we freeze the lightweight denoiser Θ and only optimize for the three noise parameters.

$$\widehat{L}_h, \widehat{\sigma}_h, \widehat{\gamma}_h = \arg\min_{L_h, \sigma_h, \gamma_h} \mathcal{L}(x_{\text{pred}} \circ m, \hat{x} \circ m), \tag{6}$$

$$\text{where} \quad x_{\text{pred}} = f^{-1}\{\hat{\Theta}[f(y; L, \sigma, \gamma)]; L, \sigma, \gamma\}$$
$$L = h_1(L_h), \sigma = h_2(\sigma_h), \gamma = h_3(\gamma_h).$$

We multiply both the restored image x_{pred} and clean image $\hat{x}$ with a mask m using Hadamard multiplication to remove the loss from the foreground and saturated region. The restored image x_{pred} is computed by sequentially applying normalization function $f(\cdot)$, the frozen denoising network $\hat{\Theta}(\cdot)$, and inverse normalization $f^{-1}(\cdot)$. In order to constrain light level L to be a non-negative value, we apply a softplus function to the variable, $L = h_1(L_h) = 10\log(1 + e^{0.1L_h})$. To match human perception, gamma correction usually has a coefficient larger than 1 [13]. We therefore constrain γ to be greater than 1 by applying another softplus function $\gamma = h_3(\gamma_h) = 1 + 10\log(1 + e^{0.1\gamma_h})$. We also map additive readout noise σ to a common range of $\sigma \in [0, 5]$ electrons using a sigmoid function, $\sigma = h_2(\sigma_h) = 5/(1 + e^{-0.1\sigma_h})$. We use a weighted sum of PSNR, SSIM, and ℓ_1 as our loss function, $\mathcal{L} = w_1\mathcal{L}_{\text{PSNR}} + w_2\mathcal{L}_{\text{SSIM}} + w_3\ell_1$, where we used $w_1 = -1, w_2 = 0, w_3 = 0$ for synthetic REDS datasets, and $w_1 = 0, w_2 = -0.425, w_3 = 0.15$ for real video conferencing dataset. We still use Adam [9] as the optimizer and optimize them for 2000 iterations.

Denoising. After the noise parameters are estimated, we map them to the constrained space $\widehat{L} = h_1(\widehat{L}_h), \widehat{\sigma} = h_2(\widehat{\sigma}_h), \widehat{\gamma} = h_3(\widehat{\gamma}_h)$, and we plug in $\widehat{L}, \widehat{\sigma}, \widehat{\gamma}$ to normalize all frames in the video and feed the normalized noisy frames to the pre-trained lightweight deep denoisers to restore clean frames.

5. Experiments

We describe our experimental settings, compare the proposed framework with baseline lightweight neural networks on synthetic and real-world data, and finally we show ablation studies.

5.1. Settings

Datasets. We use the high-quality video dataset REDS [15] and synthesize photorealistic noise on the videos. We also use a self-collected real video conferencing dataset.

- **REDS [15].** We use 240 and 20 videos for training and testing respectively. Each video is 24 fps and contains 100 frames. We use the original high-quality video frames as ground-truth clean and add photorealistic noise on the fly to emulate noisy frames.
- **Video Conferencing dataset.** We collect a video conferencing dataset consisting of 27 videos. The dataset is captured with webcams from three commonly seen on-the-market laptops with various qualities and noise profiles. We record videos of three subjects talking in video conferences. The videos are shot under dim, normal, and bright light conditions. Dim light condition corresponds to video conference at night with limited lights turned on, normal light condition is for indoors and sufficient light, and bright refers to outdoors with sunlight. Each video clip contains 120 frames.

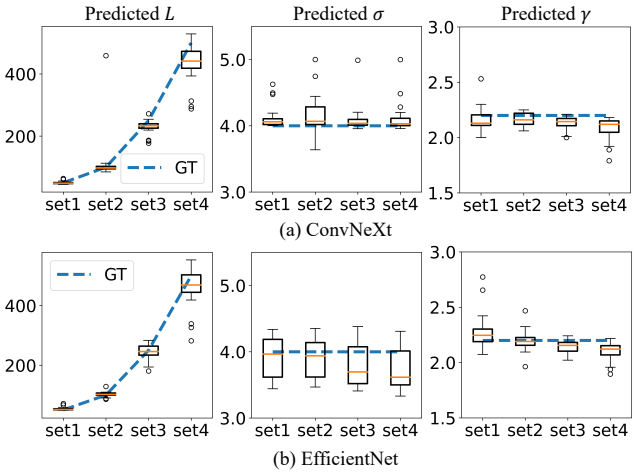


Figure 2. **Optimized noise parameters and their ground-truth.** The upper row shows estimated parameters by freezing pre-trained ConvNeXt and the lower by EfficientNet. From left to right are predictions of light level L, additive noise standard deviation σ, and gamma correction coefficient γ. Blue lines are ground-truth values, box plots are predicted noise parameters from 20 videos in REDS eval dataset, and the orange line inside each box is the average prediction.

Comparisons. We train on two off-the-shelf backbones, ConvNeXt [11] and EfficientNet [18] because of their efficient design and suitability for real-time applications. ConvNeXt modernizes ResNet [8] by introducing depth-wise convolution, inverted bottleneck design, and using a larger kernel size, GELU activation to achieve better efficiency and accuracy tradeoff. EfficientNet is a family of network architectures that are scaled up from a baseline architecture by compound model scaling. We adopt the baseline EfficientNet-B0. We use ConvNeXt and EfficientNet as backbone feature extractors and add a decoder with skip connections to restore images of the same resolution as the input images. For each backbone, we train it without and with the proposed noise normalization. We also compare with traditional denoiser BM3D [4]. We use the BM3D implementation from FFMPEG [1] with parameters `bm3d=sigma=<SIGMA>:block=4:bstep=2:group=1:estim=basic`, where the parameter `<SIGMA>` controls the denoising strength. We chose different `<SIGMA>` for datasets of different light levels to achieve the best denoising results. `<SIGMA>` equals to 150, 100, 60, and 30 for the datasets with light levels of 16dB, 20dB, 24dB, and 27dB, respectively.

Training. We use the high-quality frames from the REDS dataset as ground-truth clean frames and add noise on the fly according to Eq. 3. We fix the gamma correction coefficient to $\gamma^* = 2.2$. For each training sample, we randomly sample the light level L^* from $[50, 500]$ electrons, a typical range for a webcam sensor to capture an indoor environment with a fixed frame rate. We randomly sample the additive noise standard deviation σ^* from $[3, 5]$ electrons. For vanilla backbones, we train them with noisy and clean image pairs. For backbones with noise normalization, we first normalize the noisy and clean images with their L, σ^*, γ^* and then train them with the normalized pairs.

Testing. We evaluate our algorithm on both synthetic dataset and real video conferencing dataset. For synthetic dataset, we use the REDS evaluation set with four different noise levels. The noise levels are controlled by setting scene light levels $L^* = 50, 100, 250, 500$ electrons and fixing $\sigma^* = 4.0, \gamma^* = 2.2$, which produces images that roughly have a signal-to-noise-ratio of 16dB, 20dB, 24dB, and 27dB. For the proposed method, we utilize the first noisy frame and its ground-truth clean frame for noise parameter estimation, and apply them to the rest 99 frames in each video clip. For real-world videos, we test baseline ConNeXt and ours on self-collected videos. For the proposed method, we conduct a one-time noise parameter optimization using the first 50 frames of each video clip and apply them to normalize all noisy frames using the rest 70 frames in the same video.

5.2. Main experiments

We compare the estimated noise parameters at test-time with the ground-truth noise parameters. As shown in Figure 2, the upper row is results with ConvNeXt and the lower row is with EfficientNet. From left to right, we show estimation and ground-truth for L, σ, γ. Noisy images are synthesized with four noise settings with increasing light levels $L^* = \{50, 100, 250, 500\}$, fixed $\sigma^* = 4.0$, and fixed $\gamma^* = 2.2$. For each noise setting, we estimate L, σ, γ for 20 videos in the REDS eval set, and visualize them in box plots. Ground-truth values are shown in blue dashed lines. Both ConvNeXt and EfficientNet estimations are close to the ground truth, and ConvNeXt results are more concentrated. We notice that the mean of light level estimation is biased towards smaller values for ConvNeXt in set 4. Since a smaller light level indicates a smaller signal-to-noise-ratio, we hypothesize that ConvNeXt predicts a smaller light level to apply a larger denoising strength, which is preferable by the loss function.

In Table 1, we compare lightweight backbones trained in a vanilla style and with the proposed noise normalization. Both backbones have FLOPs at around 0.2G and are computationally efficient. For EfficientNet, ours is on par with the baseline on REDS-16dB and REDS-20dB and achieves better performance than the baseline when the noise level is low. On average, EfficientNet with normalization has higher PSNR and SSIM than the baseline EfficientNet. For

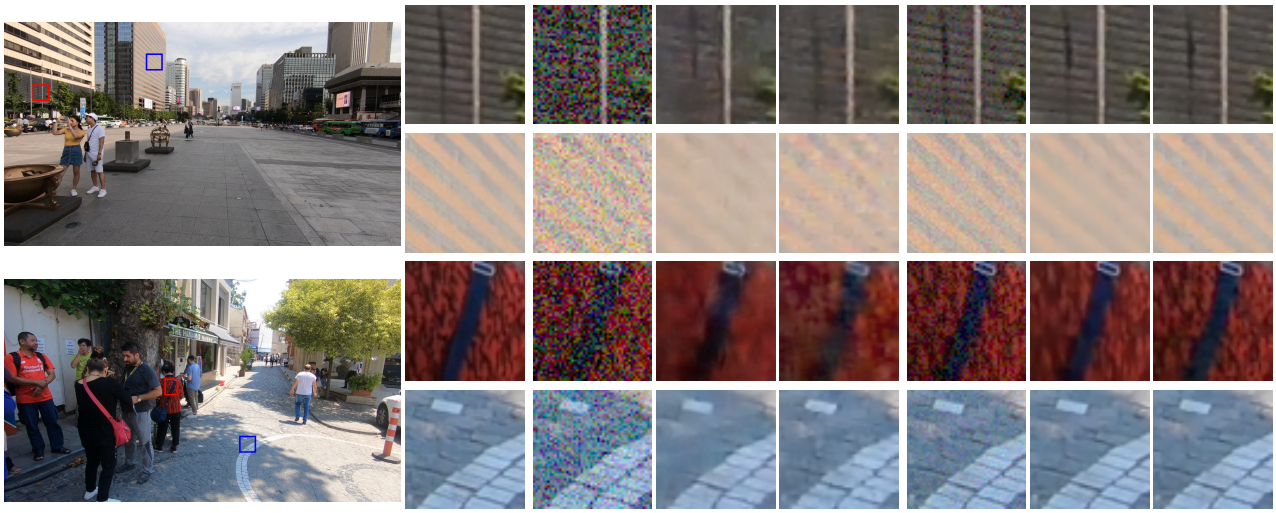


Figure 3. (b) shows two crops from an input image (a) from the REDS dataset. (c) and (f) show synthetically degraded inputs with different levels of noise, (d) and (g) show corresponding results from the baseline ConvNeXt model while (e) and (h) show results from our approach. Our approach is able to recover more details, e.g., the shadow of the pole and the horizontal slats in the top row, the striped pattern in the second row, the cloth pattern in the third row, and the brick pattern on the ground in the bottom row, while also removing noise.

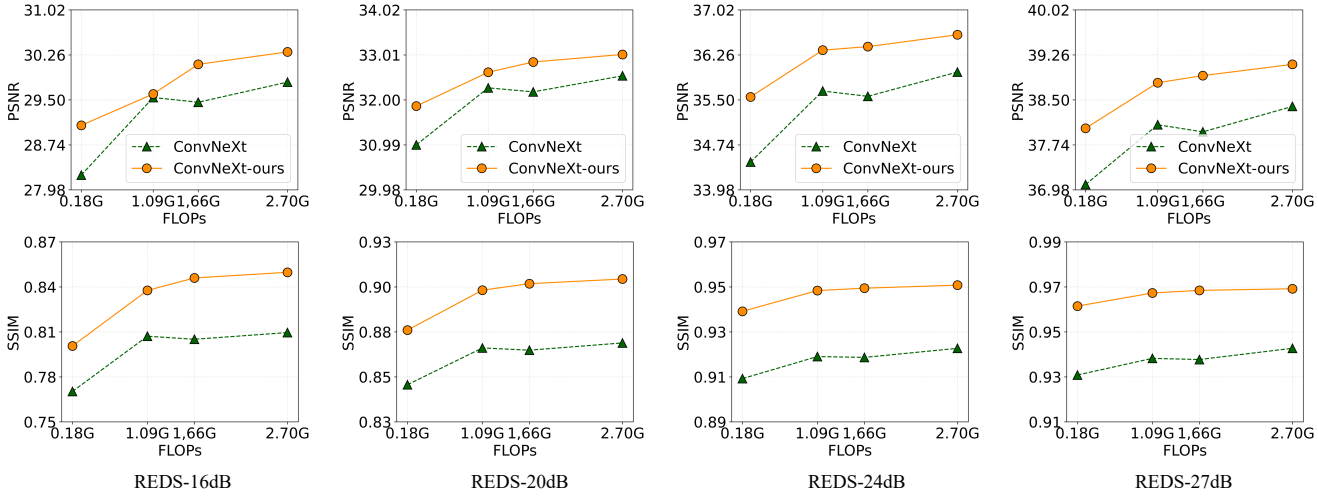


Figure 4. **Comparing ConvNeXt w/wo. normalization with an increasing network capacity.** The upper row shows PSNRs and the lower shows SSIM. From left to right, four columns show the REDS dataset with four different noise settings. The x-axis of each plot is the FLOPs of each ConNeXt model, and the y-axis is the metric. Higher values are better. Our approach improves the results at all capacities and for all input noise levels.

ConvNeXt, which is more compact than EfficientNet, ours are consistently better than baseline at all noise levels, and the average PSNR is higher than baseline by 0.94dB and SSIM by 0.03. Overall, the proposed framework is able to increase the performance of lightweight denoisers in diverse noise conditions.

We evaluate the proposed and baseline ConvNeXt on REDS with different gamma corrections at test time. We fix the light level and sensor additive noise and vary γ from 1.2 to 4.2, a common range for camera gamma encoding [6]. Note that both ConvNeXt models are trained with images encoded by $\gamma = 2.2$. As shown in Table 2, the performance of baseline ConNeXt is realatively good when $\gamma = 1.2, 2.2$ but starts to drop quickly as γ increases. The proposed method generalizes well and achieves better performance across all γ's.

Figure 3 shows qualitative results from REDS dataset. We show the same scene with two different noise levels at

	FLOPs	REDS-16dB		REDS-20dB		REDS-24dB		REDS-27dB		Average	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Input	—	17.08	0.2712	20.73	0.4139	25.60	0.6174	29.21	0.7501	23.16	0.5132
BM3D	—	21.16	0.4319	28.08	0.7790	31.24	0.8638	33.28	0.9017	28.44	0.7441
EfficientNet	0.23G	**28.70**	0.7828	**31.66**	0.8565	35.03	0.9106	37.15	0.9331	33.14	0.8708
EfficientNet + normalization	0.23G	28.52	**0.7865**	31.55	**0.8715**	**35.39**	**0.9358**	**37.97**	**0.9612**	**33.36**	**0.8888**
ConvNeXt	0.18G	28.23	0.7712	30.99	0.8483	34.45	0.9069	37.07	0.9322	32.69	0.8647
ConvNeXt + normalization	0.18G	**29.07**	**0.8011**	**31.86**	**0.8777**	**35.55**	**0.9380**	**38.02**	**0.9616**	**33.63**	**0.8946**

Table 1. **Comparing the proposed method with baseline lightweight denoisers.** We show quantitative results on REDS eval set with four different noise settings. From left to right, the noise level decreases. We evaluate the performance of all methods using PSNR and SSIM, both metrics are higher the better. Our noise normalization increases the performance of baseline ConvNeXt and EfficientNet while keeps FLOPs the same.

	REDS-20dB / $\gamma = 1.2$		REDS-20dB / $\gamma = 2.2$		REDS-20dB / $\gamma = 3.2$		REDS-20dB / $\gamma = 4.2$		Average	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
ConvNeXt	31.55	0.8576	30.99	0.8483	27.31	0.7644	23.87	0.6609	28.43	0.7610
ConvNeXt-Ours	32.98	0.9009	31.86	0.8777	29.28	0.8229	26.91	0.7651	30.26	0.8417

Table 2. **Comparing ConvNeXt on videos with various gamma correction functions.** Both methods are trained with images that are encoded by $\gamma = 2.2$. We test both methods on images with various gammas $\gamma = 1.2, 2.2, 3.2, 4.2$. The performance of baseline ConvNeXt decreases fast as γ increases, while ours generalize well to different γ's. And ours are consistently better than baseline.

16dB and 24dB. Compared to baseline ConvNeXt, ours are able to remove excessive noise while preserving details.

5.3. Ablation study

We evaluate the proposed noise normalization for ConvNeXt with an increasing capacity, including ConvNeXt-T, ConvNeXt-S, and ConvNeXt-B listed in Liu et al.'s work [11], and show PSNR and SSIM in Figure 4. The ConvNeXt capacity increases from 0.18 Gigabytes floating point operations (FLOPs) to 2.70 Gigabytes FLOPs for a 256×256 image. The results show that our normalization framework is able to improve the performance of ConvNeXt in various capacities.

	Average PSNR ↑	Average SSIM ↑
20 frames	32.05	0.8726
100 frames	32.10	0.8726
241 frames	32.13	0.8722
GT clean frame	32.16	0.8723

Table 3. **The effect of using different number of noisy frames to estimate clean a clean frame.** We compute the averaged PSNR and SSIM on REDS dataset of noise levels from 16dB to 24dB.

We examine how many frames should be averaged to get the desired clean frame. We gradually increase N from 20 to 241 and compare the estimated clean with that using ground-truth clean. When $N = 20$, the denoised PSNR is only 0.11dB smaller than that from the ground-truth. This indicates that a small number of frames is able to produce decent noise estimation and therefore denoising results.

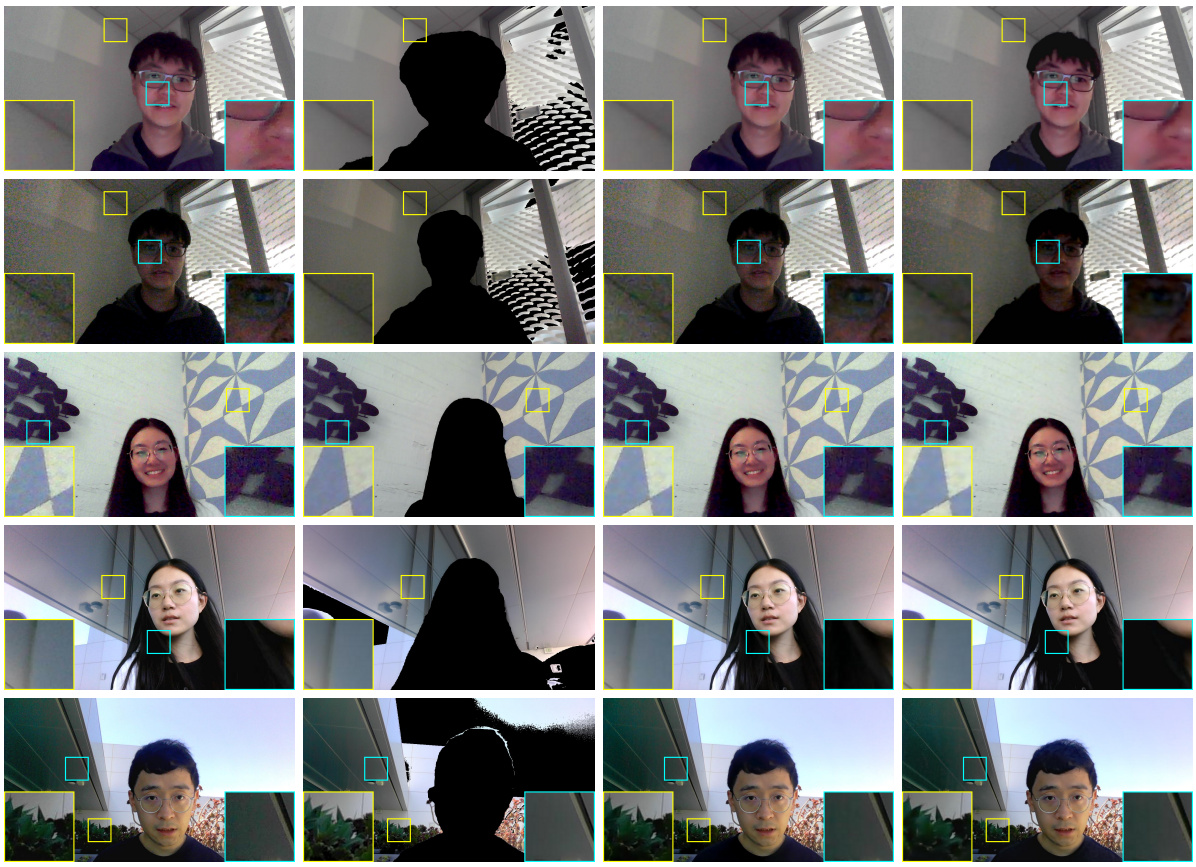
	Average PSNR ↑	Average SSIM ↑
No normalization	31.22	0.8391
Normalize L, σ	31.51	0.8654
Normalize L, σ, γ	32.16	0.8723

Table 4. **Comparing noise normalization w/wo. γ coefficient.** We compute the averaged PSNR and SSIM on REDS dataset of noise levels from 16dB to 24dB.

We ablate the effect of normalizing gamma correction. We trained ConvNeXt with a linear mapping function $f(y) = Ly + \sigma^2$ and only estimate L, σ in test time. As shown in Table 4, compared to no normalization, normalization with L, σ increases the PSNR and SSIM by 0.29dB and 0.0263. Adding γ to the normalization further achieves the best performance.

5.4. Results on real data

In Figure 5, we present qualitative results obtained from our video conferencing dataset. In general, we find our real video dataset is much more challenging than synthetic datasets. The quality difference is not as obvious as results on the synthetic dataset. In the uppermost row, our



(a) Input Image (b) Masked estimated GT (c) ConvNeXt (d) Ours

Figure 5. Results on real data. (a) shows input frame from a noisy video captured using a webcam, (b) shows the masked estimated ground truth computed from the first 50 frames of the video, (c) shows results from the ConvNeXt model while (d) shows results from the ConvNeXt method using our framework. Please zoom in to see details.

approach exhibits slightly better noise reduction capabilities compared to the baseline method, e.g., on the face. But sometimes our method tends to blur the high-frequency details from the input as seen in the bottom row. We also notice certain artifacts such as image darkening which might be due to the numerical stability of optimization which we leave as future work. Please see the supplementary material for more results.

The limitation of our method on real videos can also be attributed to the inherent simplicity of our image processing pipeline model. We hypothesize that the integration of additional steps of the imaging pipeline, such as auto white balance, demosaicing, and tone mapping, as in [3], could mitigate these artifacts.

6. Conclusion

We described an approach to adapt lightweight neural networks to uncalibrated webcams using a framework that uses normalization and test-time noise estimation. Our method does not add any additional compute except for a one-time optimization that can be done at the beginning of a video call. We also showed that our framework can incorporate steps from image processing pipeline, e.g., gamma correction, as long as they are differentiable and can be parameterized by low dimensional variables.

We did experiments on data with synthetically added noise and showed that our method can boost the performance of existing lightweight neural networks. We also collected and tested our method on a real-world dataset.

References

[1] Ffmpeg. https://ffmpeg.org/. 5

[2] Bryan Ackland and Alex Dickinson. Camera on a chip. In *1996 IEEE International Solid-State Circuits Conference. Digest of TEchnical Papers, ISSCC*, pages 22–25. IEEE, 1996. 3

[3] Tim Brooks, Ben Mildenhall, Tianfan Xue, Jiawen Chen, Dillon Sharlet, and Jonathan T. Barron. Unprocessing images for learned raw denoising. In *CVPR 2019*. 8

[4] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-

domain collaborative filtering. *IEEE Transactions on image processing*, 16(8):2080–2095, 2007. 5

[5] Alessandro Foi, Mejdi Trimeche, Vladimir Katkovnik, and Karen Egiazarian. Practical poissonian-gaussian noise modeling and fitting for single-image raw-data. *IEEE TIP*, 17(10):1737–1754, 2008. 3

[6] Michael D Grossberg and Shree K Nayar. Modeling the space of camera response functions. *IEEE transactions on pattern analysis and machine intelligence*, 26(10):1272–1282, 2004. 6

[7] Samuel W. Hasinoff, Dillon Sharlet, Ryan Geiss, Andrew Adams, Jonathan T. Barron, Florian Kainz, Jiawen Chen, and Marc Levoy. Burst photography for high dynamic range and low-light imaging on mobile cameras. In *SIGGRAPH 2016*. 4

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR 2016*. 5

[9] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, 2014. 3, 4

[10] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *ICCV 2021*. 1, 2

[11] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *CVPR 2022*. 3, 5, 7

[12] Matteo Maggioni, Yibin Huang, Cheng Li, Shuai Xiao, Zhongqian Fu, and Fenglong Song. Efficient multi-stage video denoising with recurrent spatio-temporal fusion. In *CVPR 2021*. 1, 2

[13] TOM McREYNOLDS and DAVID BLYTHE. Chapter 3 - color, shading, and lighting. In TOM McREYNOLDS and DAVID BLYTHE, editors, *Advanced Graphics Programming Using OpenGL*, The Morgan Kaufmann Series in Computer Graphics, pages 35–56. Morgan Kaufmann, San Francisco, 2005. 4

[14] Armin Mehri, Parichehr B Ardakani, and Angel D Sappa. Mprnet: Multi-path residual network for lightweight image super resolution. In *WACV 2021*. 1, 2

[15] Seungjun Nah, Sanghyun Son, Suyoung Lee, Radu Timofte, and Kyoung Mu Lee. Ntire 2021 challenge on image deblurring. In *CVPR Workshops*, pages 149–165, June 2021. 1, 4

[16] Chao Ren, Xiaohai He, Chuncheng Wang, and Zhibo Zhao. Adaptive consistency prior based deep network for image denoising. In *CVPR 2021*. 1, 2

[17] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015. 2

[18] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML 2019*. 3, 5

[19] Yuzhi Wang, Haibin Huang, Qin Xu, Jiaming Liu, Yiqun Liu, and Jue Wang. Practical deep raw image denoising on mobile devices. In *ECCV 2020*. 1, 2, 3

[20] Zhendong Wang, Xiaodong Cun, Jianmin Bao, Wengang Zhou, Jianzhuang Liu, and Houqiang Li. Uformer: A general u-shaped transformer for image restoration. In *CVPR 2022*. 1, 2

[21] Cheng Yang, Lijing Liang, and Zhixun Su. Real-world denoising via diffusion model. *arXiv preprint arXiv:2305.04457*, 2023. 1, 2

[22] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5728–5739, 2022. 1, 2

[23] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Learning enriched features for real image restoration and enhancement. In *ECCV 2020*. 1, 2