Attentive GSM Network for Defocus Deblurring

Anonymous Author(s)
Affiliation
Address
email

Abstract

Defocus blur is one often-seen effect in digital images, and correcting defocus blur is a challenging task due to the spatial variation of defocus blur degree. This paper develops an end-to-end deep learning approach for removing defocus blur from an image, so as to have an all-in-focus image for consequent vision tasks. Based on a pixel-wise Gaussian scale mixture (GSM) model for modeling defocus blur effects, this paper proposes a deep neural network (DNN) with a strong motivation from the unrolling of a fixed-point iteration for deblurring. The proposed DNN is built on a lightweight scale-recurrent architecture with an attention module for performing pixel-wise GSM-based deblurring. Extensive experiments show that the proposed DNN not only outperforms existing defocus deblurring methods noticeably, but also has its advantages in terms of model complexity and computational efficiency.

1 Introduction

The appearance sharpness of an object in an image taken by a camera is determined by the scene distance of the object to the focal plane of the camera. An object will have the sharpest appearance when it is on the focal plane, *i.e.*, the object is in focus. The area around the focal plane where objects appear to be in focus is called the depth of field (DoF). When an object is away from DoF, it appears blurry. The further is an object away from the DoF, the more blurry it appears. Such a phenomenon is called defocus or out-of-focus. The defocus blurring effect will be prominent in an image with a shallow DoF, *e.g.* the one captured with a large aperture. This paper concerns the problem of single-image defocus deblurring (SIDD) which is about reconstructing an all-in-focus image from an image with defocused regions. SIDD is of practical values to many applications in machine vision, *e.g.* photo refocusing, camera systems for robotics, and many others [1].

Consider a defocused image $\boldsymbol{y}$, which relates to its in-focus counterpart $\boldsymbol{x}$ by

$$\boldsymbol{y} = \boldsymbol{B} \circ \boldsymbol{x} + \boldsymbol{\epsilon}, \tag{1}$$

where $\boldsymbol{\epsilon}$ denotes the measurement noise. The operator $\boldsymbol{B}$ is a linear operator defined by

$$(\boldsymbol{B} \circ \boldsymbol{x})[m, n] := \sum_i \sum_j \boldsymbol{b}_{m,n}[i, j]\boldsymbol{x}[m - i, n - j]. \tag{2}$$

Each pixel indexed with (m, n) is associated with a point spread function (PSF) $\boldsymbol{b}_{m,n}$, which is determined by the distance to the focal plane. Often, these pixel-wise PSFs can be well approximated by Gaussian kernels [2, 3, 4, 5] or disk kernels [6, 7]. Without supplementary information on the scene depth, these pixel-wise PSFs are unknown. Thus, SIDD is a challenging non-linear inverse problem which needs to estimate both $\boldsymbol{B}$ and $\boldsymbol{x}$.

1.1 Discussion on Existing Work

Most existing methods (*e.g.* [2, 6, 3, 7, 4, 5]) take a two-stage approach which *(i)* estimates a dense defocus map to derive the operator $\boldsymbol{B}$ and then *(ii)* recovers the image $\boldsymbol{x}$ by using the estimated $\boldsymbol{B}$ to

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

solve (1). Generally, such a two-stage approach has a long pipe-line with many modules, and the estimation error in one module will be magnified in consequent modules. For instance, the defocus degree only can be estimated on a subset of image pixels such as edge points. Then a dense defocus map needs to be constructed by propagating these few estimations to all pixels. It can be seen that any error in the sparse defocus map will spread out to many pixels, which then will lead to erroneous PSFs that result in severe damages to the deconvoluted images [8]. As a result, the two-stage approach does not perform well in practice. Also, the computational cost in the second stage is high. Note that different from uniform blurring, $\boldsymbol{B}$ is not a convolution operator and thus its operations cannot be efficiently computed using fast Fourier transform. Since existing powerful deconvolution methods are usually iterative with many calls on the operations of $\boldsymbol{B}$, they can be slow for SIDD.

Deep learning has become one prominent tool for solving a wide range of image restoration problems. In comparison to the rapid progress of DNNs for spatially-varying motion deblurring (*e.g.* [9, 10, 11, 12, 13, 14]), there have been few works on studying DNN-based solutions to defocus deblurring. One might directly adapt an existing motion deblurring DNN for SIDD. However, the PSFs of defocus blur are very different from that of motion blur, *e.g.* roughly isotropic support vs highly curvy support. Also, the spatial variation in defocus blur differs much from that in motion blur, *e.g.* transparency effect of a moving object in motion blurring. As a result, the direct adoption is not optimal for SIDD.

Another straightforward implementation of introducing deep learning to SIDD is to replace the estimator of the defocus map in the traditional two-stage approach by a DNN-based method (*e.g.* [5]). Such an implementation still suffers from the issues existing in traditional methods, *i.e.* inaccurate estimation of $\boldsymbol{B}$ from a non-perfect defocus map and high computational cost for deblurring with a spatially-varying blurring operator. To fully exploit the potential of deep learning for SIDD, one needs to specifically design an end-to-end DNN that directly predicts an all-in-focus image from a defocused one. Very recently, Abuolaim and Brown [15] developed a DNN-based solution for constructing an all-in-focus image from dual-pixel image data *i.e.*, the DNN predicts an all-in-focus image from two sub-aperture views of the same scene. They also tried the DNN designed for dual-pixel-based defocus deblurring on SIDD, but saw a significant performance decrease.

1.2 Main Idea

This paper aims at developing an end-to-end DNN for SIDD with better performance than existing work, which is based on the following two derivations.

GSM-based model for defocus blurring Since the PSFs of defocus blurring show strong isotropy and smoothness, we propose to model the PSFs $\{\boldsymbol{b}_{m,n}\}$ by Gaussian scale mixture (GSM):

$$\boldsymbol{b}_{m,n} = \sum_{k=1}^{K} \boldsymbol{\beta}_k[m,n]\boldsymbol{g}(\sigma_k), \tag{3}$$

where $\boldsymbol{g}(\sigma)$ denotes the 2D Gaussian kernel of zero mean and variance σ^2, and $\boldsymbol{\beta}_k$ denotes the coefficient matrix for each kernel. As the GSM can fit well most isotropic distributions, (3) is a more accurate model for real defocus PSFs than often-used single Gaussian (or disk) form. Rewrite (1) as

$$\boldsymbol{y} = \sum_{m,n} \boldsymbol{\delta}_{m,n} \odot (\boldsymbol{b}_{m,n} \otimes \boldsymbol{x}) = \sum_{m,n}\sum_{k} \boldsymbol{\delta}_{m,n} \odot ((\boldsymbol{\beta}_k[m,n]\boldsymbol{g}(\sigma_k)) \otimes \boldsymbol{x}) = \sum_{k} \boldsymbol{\beta}_k \odot (\boldsymbol{g}(\sigma_k) \otimes \boldsymbol{x}),$$

where $\boldsymbol{\delta}_{m,n}$ denotes the Dirac delta centered at $[m,n]$, and $\otimes$ and $\odot$ denote 2D convolution and entry-wise multiplication respectively. We have then the GSM-based model for defocus blurring:

$$\boldsymbol{B} : \boldsymbol{x} \to \sum_{k} \boldsymbol{\beta}_k \odot (\boldsymbol{g}(\sigma_k) \otimes \boldsymbol{x}). \tag{4}$$

Unrolling fixed-point iteration As image blurring operator $\boldsymbol{B}$ is about keeping low-frequency components and attenuating high-frequency components of an image. Let $\boldsymbol{I}$ denotes the identity map, the mapping $\boldsymbol{I} - \boldsymbol{B}$ is then about attenuating low-frequency components and keeping high-frequency components. Rewriting (1) by

$$\boldsymbol{x} = \boldsymbol{y} + (\boldsymbol{I} - \boldsymbol{B}) \circ \boldsymbol{x} + \boldsymbol{n}, \tag{5}$$

we have then a fixed-point iteration for solving (1) given by

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{f}(\boldsymbol{x}^{(t)}) = \boldsymbol{y} + \boldsymbol{x}^{(t)} - \boldsymbol{B} \circ \boldsymbol{x}^{(t)} = \boldsymbol{y} + \boldsymbol{x}^{(t)} - \sum_{k=1}^{K} \boldsymbol{\beta}_k \odot (\boldsymbol{g}(\sigma_k) \otimes \boldsymbol{x}^{(t)}), \text{ for } t = 1, 2, \ldots. \tag{6}$$

Define $\sigma_1 = 0$ such that $\boldsymbol{g}(\sigma_1) = \boldsymbol{\delta}$. Let $\boldsymbol{\gamma}_1 = 1 - \boldsymbol{\beta}_1$ and $\boldsymbol{\gamma}_k = -\boldsymbol{\beta}_k$ for $k > 1$. The iteration (6) can be expressed as

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{y} + \sum_{k=1}^{K} \boldsymbol{\gamma}_k \odot (\boldsymbol{g}(\sigma_k) \otimes \boldsymbol{x}^{(t)}). \tag{7}$$

In short, based on a GSM-based model of the PSFs for defocus blurring, we can unroll a fixed-point iteration to solve (1) with learnable coefficient matrices $\boldsymbol{\gamma}_1, \cdots, \boldsymbol{\gamma}_K$. These coefficient matrices can be interpreted as the attention maps associated to different Gaussian kernels. Thus, we construct an attentive DNN with a long skip connection that utilizes (7) for SIDD. In addition, we take a multi-scale scheme to implement the unrolling: at each iteration the DNN predicts the all-in-focus image at current scale and up-samples it for the calculation of the next iteration (scale). This leads to a scale-recurrent attentive DNN with a lightweight implementation. Since the attention maps serves as the parameters of defocus PSFs, the recurrent attention in our DNN fits well the coarse-to-fine progressive estimation scheme extensively used in existing motion deblurring techniques.

1.3 Main Contributions

In comparison to existing two-stage methods and dual-pixel-based methods, this paper is among the first to present an end-to-end DNN for SIDD. See below for the summary of technical contributions

- A new GSM-based parametric model for defocus PSFs, which fits real-world data better than existing models and thus leads to better performance in SIDD.
- A new formulation of deblurring process derived from a fixed-point iteration so as to have a simple parameterization of defocus deblurring, which inspires an attentive DNN for SIDD with low model complexity and high computational efficiency.
- A scale-recurrent attention mechanism which combines coarse-to-fine progressive estimation and unrolled deblurring process for better performance.

The experiments show that the proposed DNN brings noticeable improvement over existing solutions to SIDD, in terms of recovery quality, model complexity and computational efficiency.

2 Related Work

Two-stage SIDD Most studies of two-stage methods for SIDD are concentrated on the first stage, *i.e.* defocus map estimation, while the second stage is often done by calling existing non-blind deconvolution methods (*e.g.* [16, 17]). Defocus map estimation itself is a challenging task. There are several non-learning-based methods (*e.g.* [2, 18, 4, 7]) available for sparse defocus map estimation on edge points or regions. The dense defocus map is then constructed via some propagation method (*e.g.* Poisson matting [19]). Recently, deep learning has been extensively studied for defocus map estimation; see *e.g.* [6, 3, 5, 20]. As discussed in Section 1.1, the two-stage methods suffer from sensitivity to estimation error and high computational cost. In comparison, the proposed end-to-end DNN does not involve the estimation of defocus map and thus does not suffer from these issues.

End-to-end learning for SIDD There are few works on learning an end-to-end DNN for defocus deblurring. Abuolaim and Brown [15] proposed to train an end-to-end U-Net for predicting an all-in-focus image from two view images captured by a dual-pixel sensor, and contributed a dataset containing 500 quadruples: a defocus blurred image, its all-in-focus counterpart, and two dual-pixel view images. Such a dual-pixel network showed impressive performance. However, its performance significantly decreases when being used for SIDD, where only a single image is available for input. The prerequisite on dual-view inputs limits the wider applicability of this method. In contrast, the proposed method only assumes single image as input, and thus is applicable to commodity cameras.

DNNs for spatially-varying motion deblurring There have been many studies on deep learning for spatially-varying motion deblurring, especially deblurring dynamic scenes with moving objects; see *e.g.* [9, 11, 10, 13, 12, 21, 22, 23, 24, 25, 14]. The methods using optical flow (*e.g.* [24] or temporal cues of moving objects [25] for training are not applicable to SIDD. Some of these methods also adopt multi-scale structure [10] or attention [14], but with generic design which cannot effectively exploit the inherent characteristics of defocus blurring, *e.g.* approximate isotropy and high correlation

of the shapes of pixel-wise defocus PSFs. In comparison, our DNN is specifically optimized for defocus blurring in terms of inherent characteristics of pixel-wise defocus PSFs.

Unrolling-based deep learning for image deblurring Unrolling-based DNNs have been extensively studied for non-blind image deblurring where the PSF is given as an input; see *e.g.* [26, 27, 28]. There are also some works [29, 30, 31, 32] on unrolling-based blind image deblurring where the PSF is unknown. However, these methods are restricted to handling uniform blurring. The proposed unrolling-based DNN is the first one that can handle spatially-varying defocus blurring without given the PSFs, thanks to the proposed GSM-based defocus blur model.

3 Proposed Method

3.1 Network architecture

The attentive network we construct for SIDD, named as AGSMNet (Attentive GSM Network), is based on the fix-point iteration (7). See Figure 1 for the outline of work flow, which shows how AGSMNet follows the flow of (7) in a multi-scale recurrent fashion.

Given an input image $\boldsymbol{y}$, we generate its multi-scale versions $\boldsymbol{y}_1, \cdots, \boldsymbol{y}_T$ via bi-linear downsampling with factor $2^{T-1}, \cdots, 2^0$ respectively. There are totally T recurrent blocks in AGSMNet. Let $\boldsymbol{x}_1 = \boldsymbol{y}_1$ and let $\uparrow_2$ denote the upsampling operation by factor 2. The t-th block takes $\boldsymbol{y}_t$ and $\boldsymbol{x}_t\uparrow_2$ as input and outputs $\boldsymbol{x}_{t+1}$ of the same size as $\boldsymbol{y}_t$. Each block in AGSMNet contains two modules: Gaussian convolution module (GCM) and scale-recurrent attention module (SRAM). The GCM simulates the Gaussian filtering in (7) which provides K feature maps, denoted by $\{\boldsymbol{z}_{t,1}, \ldots, \boldsymbol{z}_{t,K}\}$, for the t-th block. Accordingly, the SRAM generates K attention maps from $\boldsymbol{y}_t$, denoted by $\{\boldsymbol{\gamma}_{t,1}, \ldots, \boldsymbol{\gamma}_{t,K}\}$, which act as the coefficient maps in (7) at the t-th scale. Then the output of the tth block is defined as

$$\boldsymbol{x}_t = \boldsymbol{y} + \sum_{k=1}^{K} \omega_k(\boldsymbol{\gamma}_{t,k} \odot \boldsymbol{z}_{t,k}), \tag{8}$$

where the weighted summation with $\{\omega_k\}_k$ is implemented by 1×1 convolution. The output of the T-th block, *i.e.* $\boldsymbol{x}_T$, is used as the final output. The AGSMNet relates the scales not only by passing the output from one scale to the next, but also by the recurrence mechanism built in the SRAM. Let $\boldsymbol{x}_t^{\text{gt}}$ denotes the downsampled ground truth of the same size as $\boldsymbol{x}_t$. The training loss is defined by

$$\mathcal{L} := \sum_{t=1}^{T} \mathcal{C}(\boldsymbol{x}_t, \boldsymbol{x}_t^{\text{gt}}), \text{ for some cost function } \mathcal{C}(\cdot,\cdot). \tag{9}$$

In other words, the supervision is given at each scale. For simplicity, we assign equal weights to the loss at each scale.

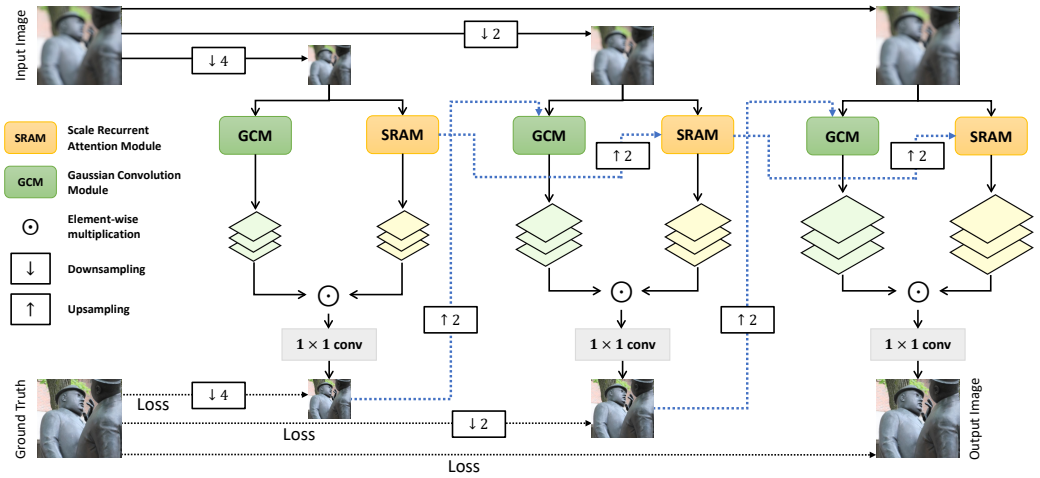

Figure 1: Diagram of proposed AGSMNet for SIDD.

3.2 Gaussian Convolution Module

The GCM is a simple group convolutional layer defined by a series of pre-designed 2D Gaussian kernels applied to R/G/B channels respectively. The kernel sizes are set to $1\times1, 3\times3, 5\times5, \cdots, M\times M$ respectively. For a kernel size $m \times m$ with $m > 1$, we generate two zero-mean Gaussian kernels with σ set to $\frac{1}{4}(m-2)$ and $\frac{1}{4}(m-1)$ respectively. Note that the Gaussian kernel with size 1×1 is the Dirac delta kernel. As a result, there are totally M different kernels. In addition to $\boldsymbol{x}_t$, we also input $\boldsymbol{y}_t$ in parallel for feeding more information to the next stage. It is note that convolution kernels in GCM is not learned for two reasons. One is that it reduces model complexity and the other is that learnable convolution kernel does not benefit the performance as empirically observed.

3.3 Scale-recurrent Attention Module

The SRAM maps the input image $\boldsymbol{y}_t \in \mathbb{R}^{P_s \times Q_s}$ to coefficient maps $\{\boldsymbol{\gamma}_{t,k} \in \mathbb{R}^{P_t \times Q_t}\}_{k=1}^{K}$. Note that $\boldsymbol{\gamma}_{t,k}[i,j]$ indicates the coefficient for mixing the pixel value at (i,j) reblurred by the k-th Gaussian kernel. When recovering the (i,j)th pixel, $\{\boldsymbol{\gamma}_{t,k}\}_{t,k=1}^{T,K}$ can be viewed as the spatial-channel attention maps. As illustrated in Figure 2, the SRAM has a lightweight structure which mainly consists of *(i)* a U-shape backbone [33] for feature extraction; and *(ii)* an attention prediction unit (APU) based on Conv-LSTM (convolutional long short-term memory) [34] for generating coefficient maps.

Backbone. The U-shape backbone is an encoder-decoder network with attention. It sequentially connects a convolutional layer, two encoder blocks, and two decoder blocks. Each encoder/decoder block contains one convolutional layer with downsampling/upsampling, two convolutional layers with residual connections, and a triplet attention block [35]. The triplet attention is used for improving the NN's adaptivity to test data. It can be viewed as an attention-in-attention mechanism in the SRAM, as SRAM also outputs attention maps. Given a feature tensor $\mathcal{F} \in \mathbb{R}^{C\times H\times W}$ defined in the channel-height-width space, the triplet attention block generates three parallel attention maps $\boldsymbol{a}_{\mathrm{W}} \in \mathbb{R}^{C\times H}$, $\boldsymbol{a}_{\mathrm{H}} \in \mathbb{R}^{C\times W}$, $\boldsymbol{a}_{\mathrm{C}} \in \mathbb{R}^{H\times W}$ from the channel-height, channel-width and height-width slices of $\mathcal{X}$ respectively, and then applies them for re-calibrating $\mathcal{X}$. This enables the network to encode spatial-channel dependencies into the generated features.

Attention prediction. The APU contains two paths. The first path mainly contains a Conv-LSTM and applies downsampling/upsampling before/after it for inducing local smoothness on the predicted attention maps. The second path contains two convolutional layers, but without downsampling/upsampling for preserving detailed information for the prediction. Let $\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2$ denote the predictions from these two paths respectively. Then the final attention map is predicted by

$$\boldsymbol{\gamma} = \tanh(\boldsymbol{\gamma}_1 \odot \boldsymbol{\gamma}_2 + \boldsymbol{\gamma}_1). \tag{10}$$

Such a design is motivated by the idea of the Squeeze-and-Attention [36] designed for segmentation. Indeed, the coefficient (attention) maps produced by SRAM are correlated to the map of pixel-wise defocus blur degrees. Such a map enables one to segment the objects from their cardinal depths as pixels with close depths will have similar blurring degrees.

The Conv-LSTM at the first path in the APU is used for capturing the dependencies among the blur degrees at different scales. The hidden state in the Conv-LSTM can capture useful information from different scales and benefit the restoration across scales. This enables the network to progressively improve the estimation on the coefficient maps.

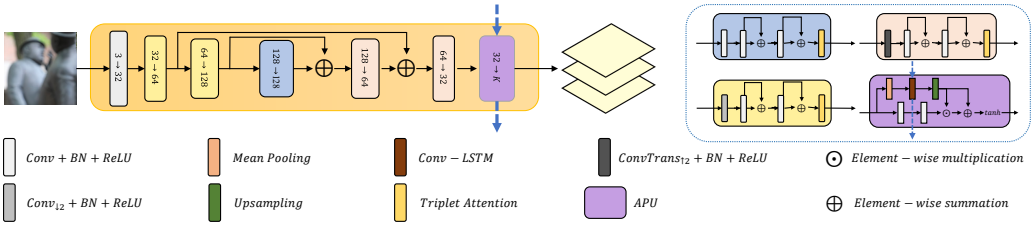

Figure 2: Diagram of SRAM. All convolutional kernels are of size 3×3.

4 Experiments

4.1 Protocol and Setting

There are few datasets available for benchmarking defocus deblurring. The most well-known one is the DPD dataset [15]. It provides 500 pairs of images with defocus blur and their corresponding all-in-focus images, as well as the two associated dual-pixel sub-aperture views, all with 16-bit color. The training/validation/test splits in the dataset consist of $350/74/76$ pairs of images, respectively. Our AGSMNet is trained and tested on the DPD dataset without using the dual-pixel views. In addition, we test AGSMNet on the RTF test set [6] as well as the images from CUHK-BD dataset [37].

Through all experiments, the maximum size of Gaussian kernels in GCM is set to $M = 21$. The number of scales is set to $T = 3$. The learnable network weights in SRAM are initialized by Xavier. The Adam optimizer [38] is used for training with 3000 epochs are and batch size of 4. The learning rate is fixed at $1e^{-4}$ in the first 2000 epochs and decayed to $1e^{-5}$ in the last 1000 epochs. The loss function is set to squared ℓ_2 loss in the first 1000 iterations and alternatingly set to ℓ_2 loss and SSIM loss every 500 epochs afterward. Data augmentation is implemented by random cropping to 256×256 pixels. We implemented AGSMNet with Pytorch 1.6. The code will be made publicly available upon paper's acceptance.

Three image quality metrics are used for quantitative evaluation, including two standard metrics: PSNR (Peak Signal to Noise Ratio) and SSIM (Structural Similarity Index Measure) [39], and the LPIPS (Learned Perceptual Image Patch Similarity) [40] for perceptual quality (also used in [15]).

4.2 Comparison against Existing Methods

4.2.1 Evaluation on DPD dataset

The methods for comparison include JNB [2], EBDB [4], DMENet [5] and DPDNet [15]. The DPDNet [15] has two versions: DPDNet-D taking two dual-pixel images as input and DPDNet-S only taking a single image as input. Both versions are included for comparison whose results are reproduced by the pre-trained models from the authors. The JNB, EDBD and DMENet are all two-stage methods focusing on defocus map estimation, with results quoted from [15]. In addition, we also include two DNNs of deblurring dynamic scene: SRN [10] and AttNet [14]. These two DNNs also adopt a multi-scale architecture similar to our SRAM. Both DNNs are retrained using the same data as ours for fairness.

See Table 1 for the quantitative comparison. It can be seen that AGSMNet outperforms all other compared methods. The performance of EBDB, DMENet and JNB is much worse than ours. This is probably due to that the error in their defocus map is magnified in the consequent deconvolution process. This indicates the advantage of our end-to-end DNN which does not estimate defocus map. Our AGSMNet also outperforms DPDNet-S by a large margin. It is not surprising as DPDNet-S is originally designed for defocus deblurring on two dual-pixel images, not the single one. Also unsurprisingly, AGSMNet outperforms SRN and AttNet, two DNNs designed for deblurring dynamic scenes not for defocus deblurring. Surprisingly, our AGSMNet with single-image input even outperforms DPDNet-D which takes two dual-pixel images as input. Such an experiment clearly demonstrated the effectiveness of the proposed AGSMNet on SIDD.

Table 1: Quantitative comparison of different methods on DPD dataset.

Model	PSNR (dB)	SSIM	LPIPS	Parameters (Million)	FLOPs (Billion)	Model Size (MegaBytes)	Time (Seconds) CPU	Time (Seconds) GPU
JNB [2]	23.84	0.715	0.315	-	-	-	843.1	-
EBDB [4]	23.45	0.683	0.336	-	-	-	929.7	-
DMENet [5]	23.41	0.714	0.349	26.94	-	-	613.7	-
DPDNet-S [15]	24.34	0.747	0.277	32.25	4042.5	355.3	56.6	0.41
DPDNet-D [15]	25.12	0.786	0.223	32.25	4048.6	355.3	56.6	0.41
SRN [10]	24.61	0.612	0.265	10.25	3117.6	32.0	58.3	**0.032**
AttNet [14]	25.22	0.781	0.219	6.91	3450.4	39.5	62.8	0.041
ASGMNet [Ours]	**25.47**	**0.789**	**0.219**	**1.41**	**603.5**	**5.7**	**43.9**	0.040

Comparison on complexity and efficiency The AGSMNet is also compared to other DNNs in terms of model complexity and computational efficiency. The model complexity is measured by three metrics: number of parameters, number of FLOPs (floating-point operations per second), and model size. The computational efficiency time is measured by the average test time on an image of 1680×1120 pixels, using an Intel i5-9600KF CPU and a single NVIDIA GTX 1080 GPU respectively. For non-DNN-based methods, only the test time on CPU is reported.

See Table 1 for the comparison. In terms of all three metrics, AGSMNet's complexity is much lower than that of other deep models. For instance, around $1/23$ of DPDNet and $1/7$ of SRN for number of parameters, and around $1/62$ of DPDNet for model size. Such low complexity comes from the compact architecture of AGSMNet. Regarding the running time, AGSMNet is also much faster than the JNB, EBDB and DEMNet, as these methods need to call an iterative deconvolution post-process which is slow. AGSMNet is about ten time faster than DPDNet, and its speed is comparable to SRN and AttNet. It is noted that AGSMNet has a much faster training time due to its low complexity. Its training on the DPD dataset takes less than 32 hours, while SRN and AttNet take nearly three days.

Visualization of attention maps Figure 3 visualizes the attention maps at different scales which correspond to the Gaussian kernel of size 15×15. It can be seen that the attention maps are closely related to the estimated defocus map that indicates the point-wise blurring degree. Also, the attention maps across scales are mostly consistent with some fine adjustments.

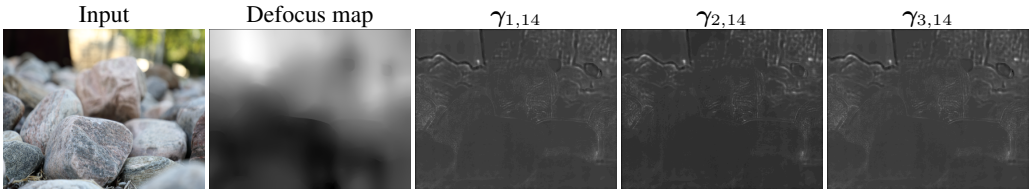

Figure 3: Visualization of attention maps associated to the 14th Gaussian kernels. The defocus map is estimated by [4]. Darker color indicates lower value.

Evaluation on noise robustness The robustness of AGSMNet to image noise is evaluated on the DPD dataset with additional corruption by additive Gaussian white noise with s.t.d. $\hat{\sigma} = 1, 3, 5$ respectively. The methods: DPDNet-D, SRN, AttNet, are included for comparison. Both the models trained on original DPD training set, as well as the models retrained on the noisier version of DPD training set are included. The later ones are marked with a "+". See Table 2 for the results. In the presence of noise, our AGSMNet still outperforms other models in both versions of training dataset.

Table 2: Quantitative comparison of different methods on Noisier DPD test set.

Noise	DPDNet-D			SRN			AttNet			AGSMNet		
$\hat{\sigma}$	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
1	25.08	0.788	0.220	24.58	0.692	0.315	25.18	0.783	0.221	**25.43**	**0.784**	**0.225**
3	24.73	0.746	0.248	24.48	0.668	0.366	24.77	0.749	0.261	**25.02**	**0.758**	**0.246**
5	24.07	0.665	0.441	24.30	0.628	0.411	24.36	0.660	0.347	**24.61**	**0.712**	**0.299**
Noise	DPDNet-D+			SRN+			AttNet+			AGSMNet+		
$\hat{\sigma}$	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
1	25.09	0.788	0.220	24.61	0.697	0.316	25.18	0.783	0.221	**25.48**	**0.789**	**0.221**
3	24.76	0.748	0.248	24.53	0.672	0.360	24.85	0.753	0.255	**25.12**	**0.760**	**0.247**
5	24.37	0.710	0.379	24.34	0.632	0.399	24.48	0.681	0.332	**24.92**	**0.742**	**0.273**

4.2.2 Evaluation on RTF dataset and CUHK-BD sample images

The RTF dataset [6] contains 22 pairs of defocused and all-in-focus images. Following [6], in addition to original images, two noisier versions are generated by adding Gaussian white noise with s.t.d. $\hat{\sigma} = 1$ and 2.55 respectively. The methods: DPDNet-S, SRN and AttNet are included for comparison. Note that the training data of RTF is not available. Thus, we directly apply the models trained on original and noisier DPD datasets respectively. See Table 3 for the results. The AGSMNet again outperforms other DNN models.

Table 3: Quantitative comparison of different methods on RTF test set.

Noise	DPDNet-S			SRN			AttNet			AGSMNet		
$\hat{\sigma}$	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
0	23.61	0.597	0.296	23.71	0.617	0.324	25.45	0.802	0.219	**25.72**	**0.811**	**0.211**
1	23.58	0.592	0.332	23.68	0.611	0.311	24.99	0.797	0.243	**25.56**	**0.798**	**0.207**
2.55	23.44	0.576	0.344	23.50	0.591	0.276	23.83	0.727	0.257	**25.01**	**0.761**	**0.205**

Noise	DPDNet-S+			SRN+			AttNet+			AGSMNet+		
$\hat{\sigma}$	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
0	23.65	0.605	0.290	23.70	0.617	0.325	25.40	0.792	0.225	**25.69**	**0.808**	**0.212**
1	23.59	0.594	0.329	23.65	0.605	0.313	25.06	0.800	0.236	**25.60**	**0.802**	**0.206**
2.55	23.49	0.581	0.341	23.54	0.597	0.281	24.33	0.749	0.250	**25.21**	**0.787**	**0.210**

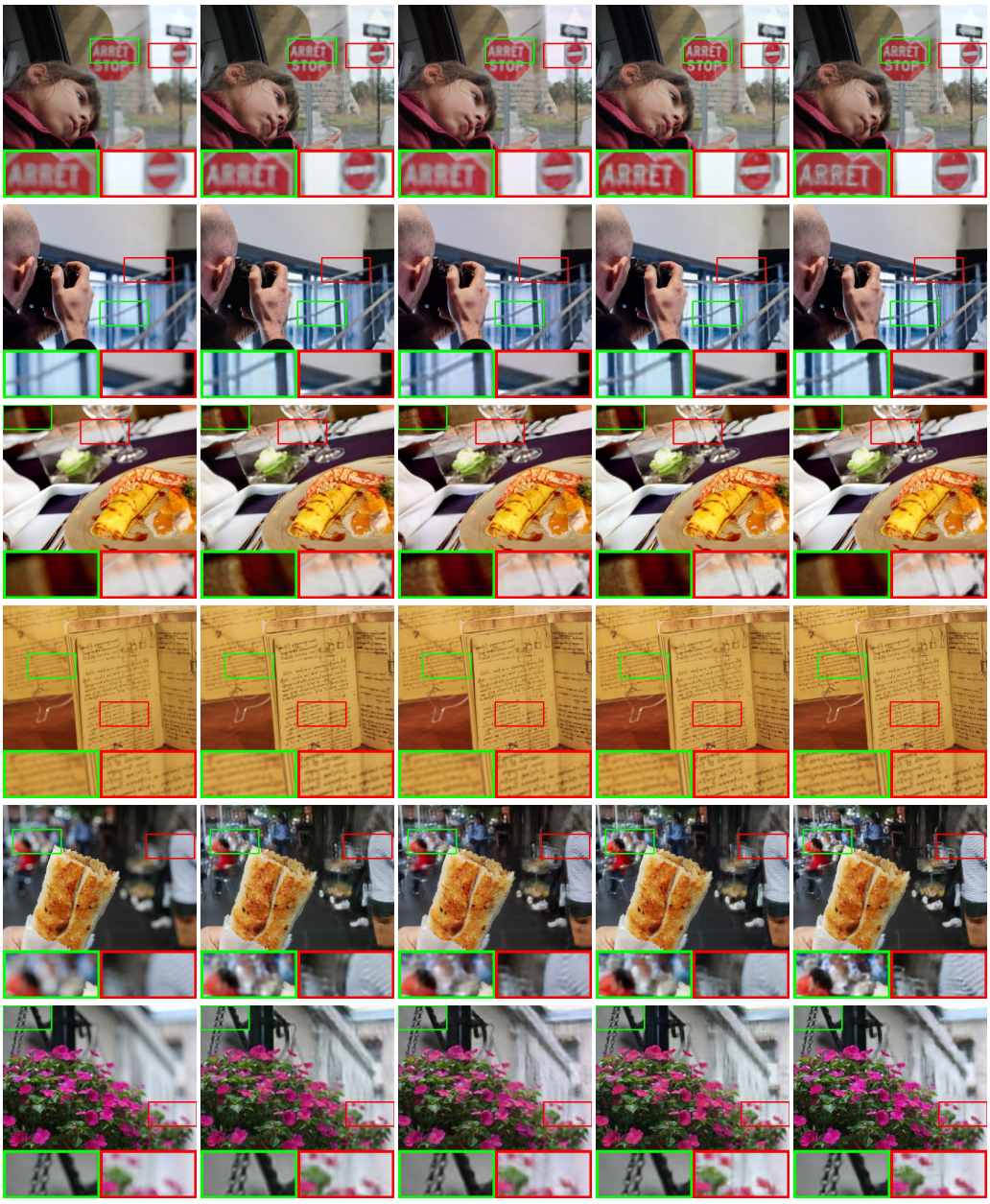

Figure 4: Visual comparison of SIDD results of different methods on CUHK-BD sample images.

The CUHK-BD dataset [37] contains 704 defocused images without ground truths. We select some of its images for test. See Figure 4 for the visual inspection on the results. Our AGSMNet can successfully restore fine details with less visual artifacts, in comparison to DPDNet-S, SRN and AttNet. More visual comparison can be found in supplementary materials.

4.3 Ablation Study

To evaluate the performance impact of each component in AGSMNet, we implement the following baselines from AGSMNet for comparison. (a) 'ASGMNet(1)': Use only original image scale in AGSMNet with $T = 1$. (b) 'w/o Conv-LSTM': Replace the Conv-LSTM block with a convolutional layer with the same kernel size from the APU in SRAM, and its weights are shared across scales for relating different scales; (c) 'SRAM*': Replace the last activation of SRAM by an 1×1 convolutional layer with Sigmoid activation, and then train the resulting model to directly predict a clear image from the input; (d) 'Learned GCM (GSM)': The kernels in GCM are initialized to GSM kernels and learned with the DNN. (e) 'Learned GCM (Rand)': The kernels in GCM are randomly initialized by Xavier and learned with the DNN. Note that sharing the weights of GCM across scales will lead to better performance in (d) and (e). Thus, it is adopted in (d) and (e). The models of these baselines are trained in the same way as original ASGMNet.

See Table 4 for the ablation study, from which we have the following observations. (a) The multi-scale scheme in SRAM leads to significant improvement over the single-scale one, with only a few additional model parameters. (b) The improvement using Conv-LSTM over weight sharing across scales is significant, which implies the Conv-LSTM can effectively exploit the features learned from different scales to guide the deblurring process. (c) Directly using the SRAM structure for end-to-end SIDD leads much worse results, which justifies the effectiveness of our specific DNN design. (d) Learning with GSM initialization leads to a very minor improvement over fixed GSM-based kernels in GCM. (e) Learning GCM with random initialization even yields slight worse results than using fixed GSM kernels. Both (d) and (e) indicate the effectiveness of our GSM model for defocus kernels. See also Figure 3 for the kernels w/o and w/ learning in GCM. It can be seen that the learned kernels based on GSM initialization are very close to original GSM kernels.

Table 4: Results in ablation study.

Model	PSNR (dB)	SSIM	LPIPS	Parameters (Million)	FLOPs (Billion)	Model Size (MegaBytes)	Time (Seconds)
SRAM*	21.98	0.724	0.251	1.41	606.1	5.30	0.040
ASGMNet(1)	23.63	0.723	0.248	1.41	459.7	5.70	**0.036**
w/o Conv-LSTM	25.00	0.774	0.231	**1.11**	**418.7**	**4.33**	0.036
Learned GCM (Rand)	25.35	0.787	0.223	1.41	603.5	5.70	0.040
Learned GCM (GSM)	**25.49**	**0.790**	**0.217**	1.41	603.5	5.70	0.040
ASGMNet	25.46	0.789	0.219	1.41	603.5	5.70	0.040

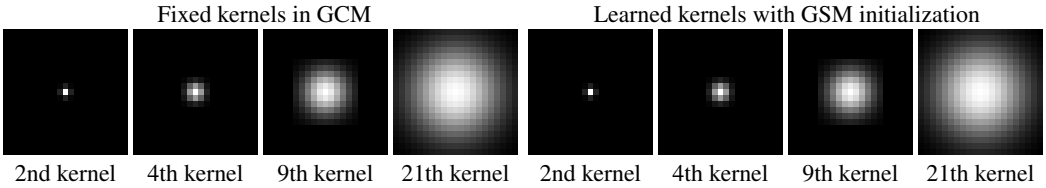

Figure 5: Visualization of fixed/learned kernels in GCM.

5 Conclusion

Defocus blur is an often-seen blur effects on images and has its own characteristics from motion blur. This paper proposed an attentive network for SIDD with strong motivation from unrolling a fixed-point iteration derived from a GSM-based model of defocus PSFs. Together with a a scale-recurrent implementation, we developed a lightweight DNN with SoTA performance. In future, we would extend our method to handling other image deblurring problems such as dynamic scene deblurring.

References

[1] Anat Levin, Rob Fergus, Frédo Durand, and William T Freeman. Image and depth from a conventional camera with a coded aperture. In *Proc. SIGGRAPH*, 2007.

[2] Jianping Shi, Li Xu, and Jiaya Jia. Just noticeable defocus blur detection and estimation. In *Proc. CVPR*, pages 657–665, 2015.

[3] Jinsun Park, Yu-Wing Tai, Donghyeon Cho, and In So Kweon. A unified approach of multi-scale deep and hand-crafted features for defocus estimation. In *Proc. CVPR*, pages 2760–2769, 2017.

[4] Ali Karaali and Claudio Jung. Edge-based defocus blur estimation with adaptive scale selection. *IEEE Trans. Image Process.*, 27:1126–1137, 2018.

[5] Junyong Lee, Sungkil Lee, Sunghyun Cho, and Seungyong Lee. Deep defocus map estimation using domain adaptation. In *Proc. CVPR*, pages 12214–12222, 2019.

[6] Laurent D'Andrès, Jordi Salvador, Axel Kochale, and Sabine Süsstrunk. Non-parametric blur map regression for depth of field extension. *IEEE Trans. Image Process.*, 25:1660–1673, 2016.

[7] Sunghyun Cho and Seungyong Lee. Convergence analysis of map based blur kernel estimation. In *Proc. ICCV*, pages 4818–4826, 2017.

[8] Hui Ji and Kang Wang. Robust image deblurring with an inaccurate blur kernel. *IEEE Trans. Image Process.*, 21(4):1624–1634, 2011.

[9] Seungjun Nah, Tae Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *Proc. CVPR*, pages 257–265, 2017.

[10] Xin Tao, Hongyun Gao, Yi Wang, Xiaoyong Shen, Jue Wang, and Jiaya Jia. Scale-recurrent network for deep image deblurring. In *Proc. CVPR*, pages 8174–8182, 2018.

[11] Jiawei Zhang, Jinshan Pan, Jimmy Ren, Yibing Song, Linchao Bao, Rynson WH Lau, and Ming-Hsuan Yang. Dynamic scene deblurring using spatially variant recurrent neural networks. In *Proc. CVPR*, pages 2521–2529, 2018.

[12] Hongyun Gao, Xin Tao, Xiaoyong Shen, and Jiaya Jia. Dynamic scene deblurring with parameter selective sharing and nested skip connections. In *Proc. CVPR*, pages 3848–3856, 2019.

[13] Hongguang Zhang, Yuchao Dai, Hongdong Li, and Piotr Koniusz. Deep stacked hierarchical multi-patch network for image deblurring. In *Proc. CVPR*, pages 5978–5986, 2019.

[14] Yong Xu, Ye Zhu, Yuhui Quan, and Hui Ji. Attentive deep network for blind motion deblurring on dynamic scenes. *Computer Vision and Image Understanding*, 205:103169, 2021.

[15] Abdullah Abuolaim and Michael S. Brown. Defocus deblurring using dual-pixel data. In *Proc. ECCV*, 2020.

[16] D. A. Fish, A. M. Brinicombe, E.R. Pike, and J. G. Walker. Blind deconvolution by means of the richardson-lucy algorithm. *Journal of The Optical Society of America A-optics Image Science and Vision*, 12:58–65, 1995.

[17] Dilip Krishnan and Rob Fergus. Fast image deconvolution using hyper-laplacian priors. In *Proc. NIPS*, 2009.

[18] Guodong Xu, Yuhui Quan, and Hui Ji. Estimating defocus blur via rank of local patches. In *Proc. ICCV*, pages 5371–5379, 2017.

[19] Jian Sun, Jiaya Jia, Chi-Keung Tang, and Heung-Yeung Shum. Poisson matting. In *ACM SIGGRAPH*, pages 315–321. 2004.

[20] Saeed Anwar, Zeeshan Hayder, and Fatih Porikli. Deblur and deep depth from single defocus image. *Machine Vision and Applications*, 32:1–13, 2021.

[21] Kuldeep Purohit and AN Rajagopalan. Region-adaptive dense network for efficient motion deblurring. In *Proc. AAAI*, volume 34, pages 11882–11889, 2020.

[22] Maitreya Suin, Kuldeep Purohit, and AN Rajagopalan. Spatially-attentive patch-hierarchical network for adaptive motion deblurring. In *Proc. CVPR*, pages 3606–3615, 2020.

[23] Kaihao Zhang, Wenhan Luo, Yiran Zhong, Lin Ma, Bjorn Stenger, Wei Liu, and Hongdong Li. Deblurring by realistic blurring. In *Proc. CVPR*, pages 2737–2746, 2020.

[24] Yuan Yuan, Wei Su, and Dandan Ma. Efficient dynamic scene deblurring using spatially variant deconvolution network with optical flow guided training. In *Proc. CVPR*, pages 3555–3564, 2020.

[25] Dongwon Park, Dong Un Kang, Jisoo Kim, and Se Chun. Multi-temporal recurrent neural networks for progressive non-uniform single image deblurring with incremental temporal training. In *Proc. ECCV*, 2020.

[26] Yuesong Nan and Hui Ji. Deep learning for handling kernel/model uncertainty in image deconvolution. In *Proc. CVPR*, pages 2388–2397, 2020.

[27] Dong Gong, Zhen Zhang, Qinfeng Shi, Anton van den Hengel, Chunhua Shen, and Yanning Zhang. Learning deep gradient descent optimization for image deconvolution. *IEEE Trans. Neural Netw. Learning Syst.*, 2020.

[28] Weisheng Dong, Peiyao Wang, Wotao Yin, and Guangming Shi. Denoising prior driven deep neural ketwork for image restoration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(10):2305–2318, 2019.

[29] Christian J Schuler, Michael Hirsch, Stefan Harmeling, and Bernhard Schölkopf. Learning to deblur. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(7):1439–1451, 2015.

[30] Wangmeng Zuo, Dongwei Ren, David Zhang, Shuhang Gu, and Lei Zhang. Learning iteration-wise generalized shrinkage–thresholding operators for blind deconvolution. *IEEE Trans. Image Process.*, 25(4):1751–1764, 2016.

[31] Yuelong Li, Mohammad Tofighi, Vishal Monga, and Yonina C Eldar. An algorithm unrolling approach to deep image deblurring. In *Proc. ICASSP*, pages 7675–7679. IEEE, 2019.

[32] Yuelong Li, Mohammad Tofighi, Junyi Geng, Vishal Monga, and Yonina C Eldar. Efficient and interpretable deep blind image deblurring via algorithm unrolling. *IEEE Trans. Comput. Imaging*, 6:666–681, 2020.

[33] Xiao-Jiao Mao, Chunhua Shen, and Yubin Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *Proc. NIPS*, 2016.

[34] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Proc. NIPS*, 2015.

[35] Diganta Misra, Trikay Nalamada, Ajay Uppili Arasanipalai, and Qibin Hou. Rotate to attend: Convolutional triplet attention module. In *Proc. IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3139–3148, 2021.

[36] Zilong Zhong, Zhong Qiu Lin, Rene Bidart, Xiaodan Hu, Ibrahim Ben Daya, Jonathan Li, and Alexander Wong. Squeeze-and-attention networks for semantic segmentation. In *Proc. CVPR*, pages 13062–13071, 2020.

[37] Jianping Shi, Li Xu, and Jiaya Jia. Discriminative blur detection features. In *Proc. CVPR*, pages 2965–2972, 2014.

[38] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zach DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[39] Zhou Wang, Alan Bovik, Hamid Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.*, 13:600–612, 2004.

[40] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proc. CVPR*, pages 586–595, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Supplementary Materials.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] However, as stated in Section 4.1, the code will be made public upon paper's acceptance.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]