i-TAP meeting

홍익대학교
노승문
2021. 5. 13

Demosaicing

Deep Joint Demosaicking and Denoising

Michaël Gharbi
MIT CSAIL

Gaurav Chaurasia
MIT CSAIL

Sylvain Paris
Adobe

Frédo Durand
MIT CSAIL

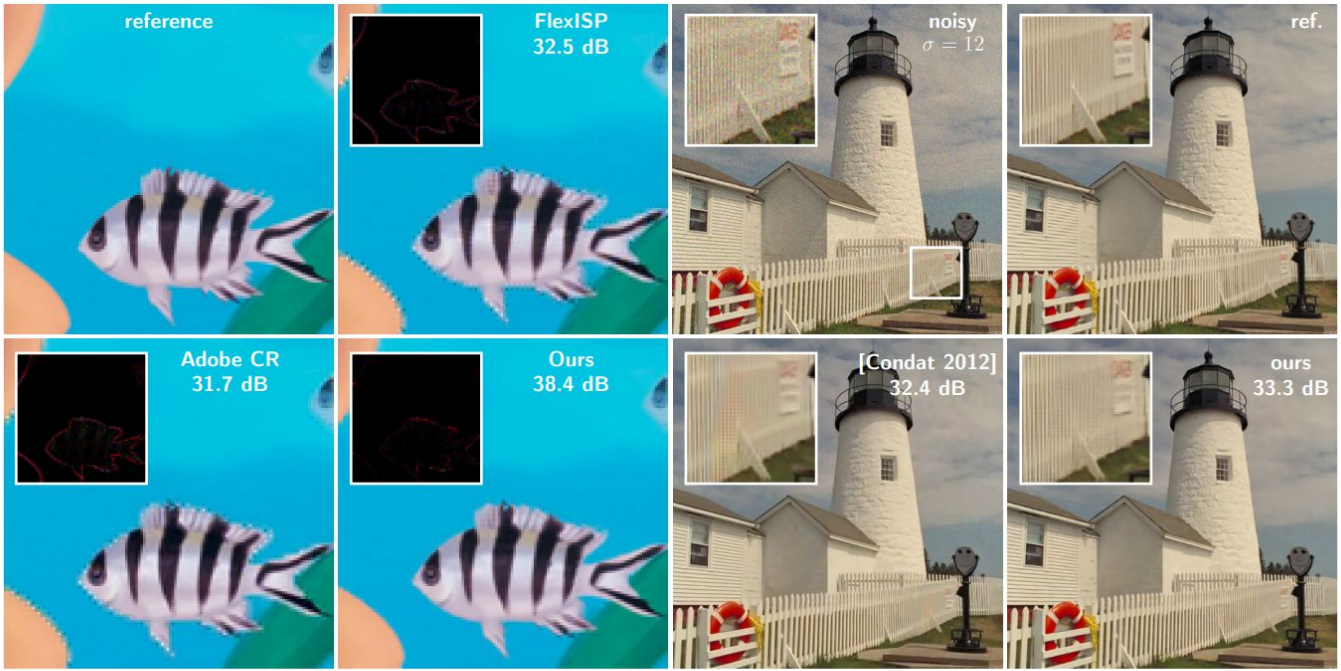


Figure 1: *We propose a data-driven approach for jointly solving denoising and demosaicking. By carefully designing a dataset made of rare but challenging image features, we train a neural network that outperforms both the state-of-the-art and commercial solutions on demosaicking alone (group of images on the left, insets show error maps), and on joint denoising–demosaicking (on the right, insets show close-ups). The benefit of our method is most noticeable on difficult image structures that lead to* moiré *or* zippering *of the edges.*

ACM
Transactions on Graphs
2016

Deep Image Demosaicking using a Cascade of Convolutional Residual Denoising Networks

Filippos Kokkinos and Stamatios Lefkimmiatis
{filippos.kokkinos, s.lefkimmiatis}@skoltech.ru

Skolkovo Institute of Science and Technology (Skoltech), Moscow, Russia

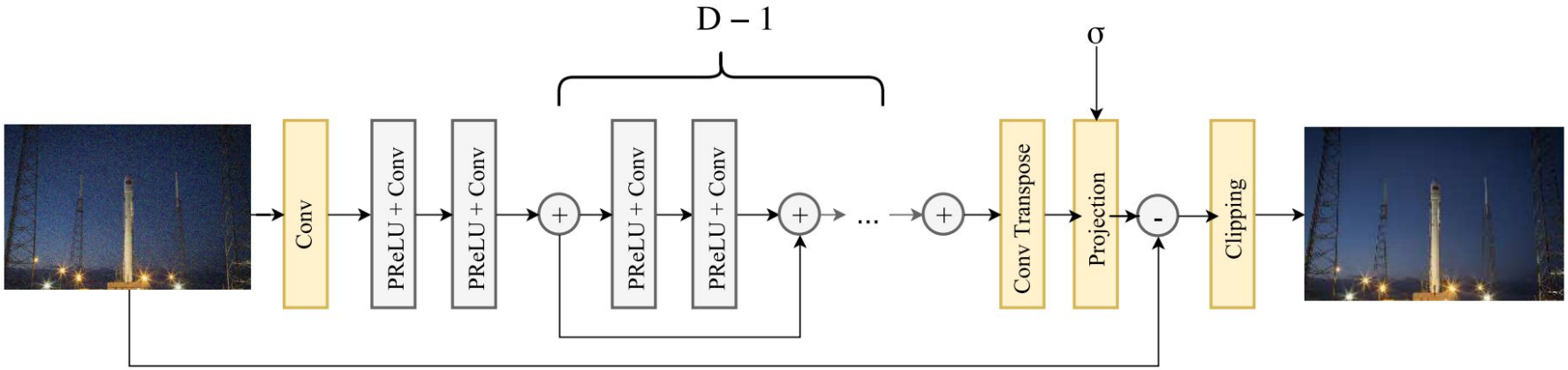


Fig. 1: The architecture of the proposed ResDNet denoising network, which serves as the back-bone of our overall system.

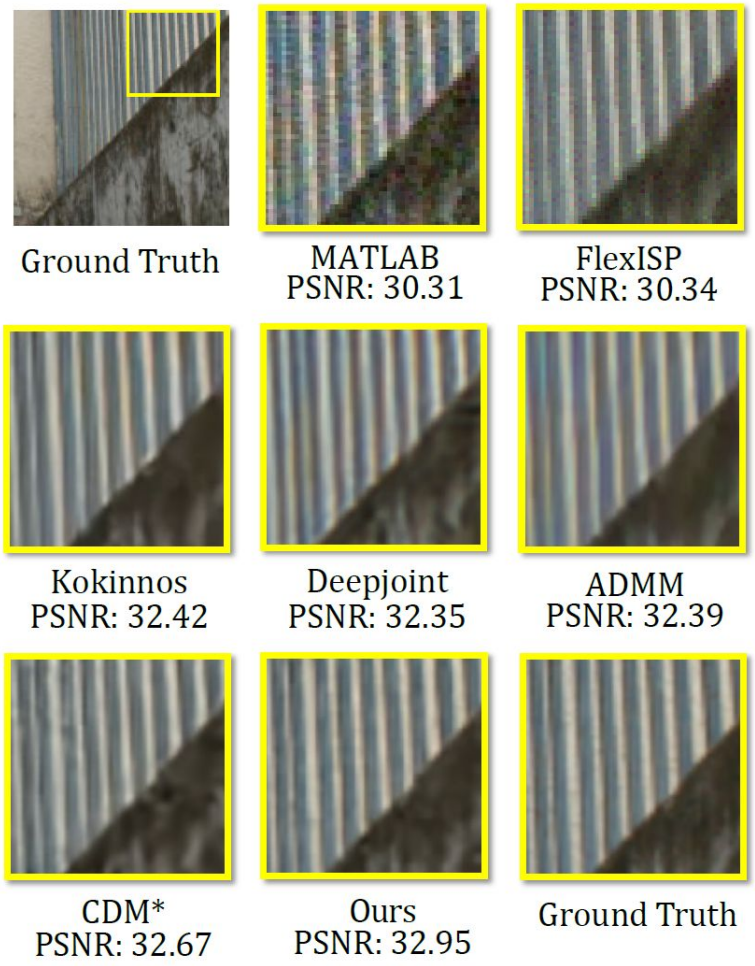
Joint Demosaicing and Denoising with Self Guidance

Lin Liu[1,2] Xu Jia[2*] Jianzhuang Liu[2] Qi Tian[2]

[1]CAS Key Laboratory of GIPAS, University of Science and Technology of China
[2]Noah's Ark Lab, Huawei Technologies

Abstract

Usually located at the very early stages of the computational photography pipeline, demosaicing and denoising play important parts in the modern camera image processing. Recently, some neural networks have shown the effectiveness in joint demosaicing and denoising (JDD). Most of them first decompose a Bayer raw image into a four-channel RGGB image and then feed it into a neural network. This practice ignores the fact that the green channels are sampled at a double rate compared to the red and the blue channels. In this paper, we propose a self-guidance network (SGNet), where the green channels are initially estimated and then works as a guidance to recover all missing values in the input image. In addition, as regions of different frequencies suffer different levels of degradation in image restoration. We propose a density-map guidance to help the model deal with a wide range of frequencies. Our model outperforms state-of-the-art joint demosaicing and denoising methods on four public datasets, including two real and two synthetic data sets. Finally, we also verify that our method obtains best results in joint demosaicing , denoising and super-resolution.



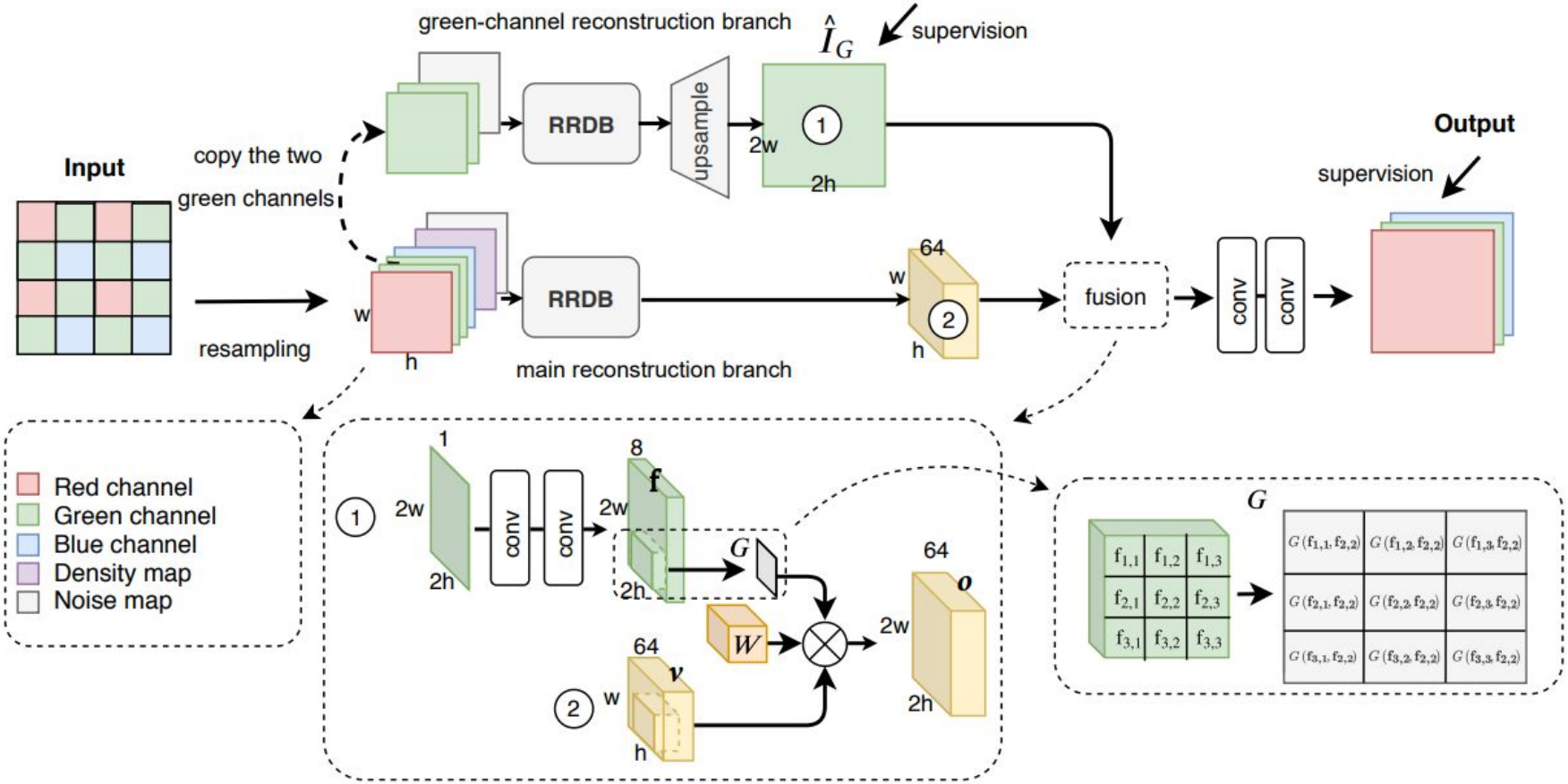


Figure 2: The architecture of the proposed self-guidance network.

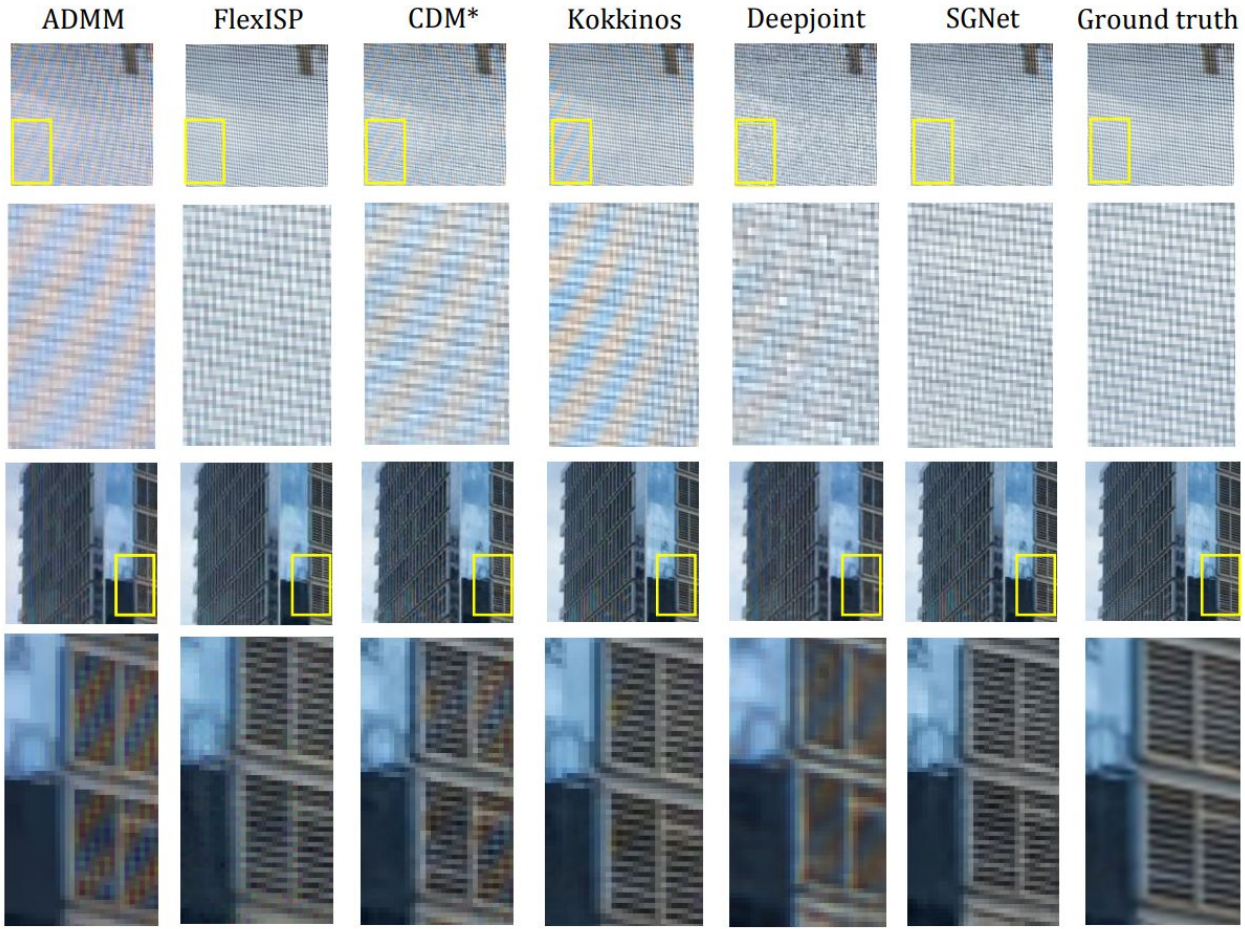


Figure 5: Visual comparison between state-of-the-arts and our method for joint demosaicing and denoising.

Replacing Mobile Camera ISP with a Single Deep Learning Model

Andrey Ignatov
andrey@vision.ee.ethz.ch

Luc Van Gool
vangool@vision.ee.ethz.ch

Radu Timofte
timofter@vision.ee.ethz.ch

ETH Zurich, Switzerland

CVPR 2020

Abstract

As the popularity of mobile photography is growing constantly, lots of efforts are being invested now into building complex hand-crafted camera ISP solutions. In this work, we demonstrate that even the most sophisticated ISP pipelines can be replaced with a single end-to-end deep learning model trained without any prior knowledge about the sensor and optics used in a particular device. For this, we present PyNET, a novel pyramidal CNN architecture designed for fine-grained image restoration that implicitly learns to perform all ISP steps such as image demosaicing, denoising, white balancing, color and contrast correction, demoireing, etc. *The model is trained to convert RAW Bayer data obtained directly from mobile camera sensor into photos captured with a professional high-end DSLR camera, making the solution independent of any particular mobile ISP implementation. To validate the proposed approach on the real data, we collected a large-scale dataset consisting of 10 thousand full-resolution RAW–RGB image pairs captured in the wild with the Huawei P20 cameraphone (12.3 MP Sony Exmor IMX380 sensor) and Canon 5D Mark IV DSLR. The experiments demonstrate that the proposed solution can easily get to the level of the embedded P20's ISP pipeline that, unlike our approach, is combining the data from two (RGB + B/W) camera sensors. The dataset, pre-trained models and codes used in this paper are available on the project website:* https://people.ee.ethz.ch/~ihnatova/pynet.html

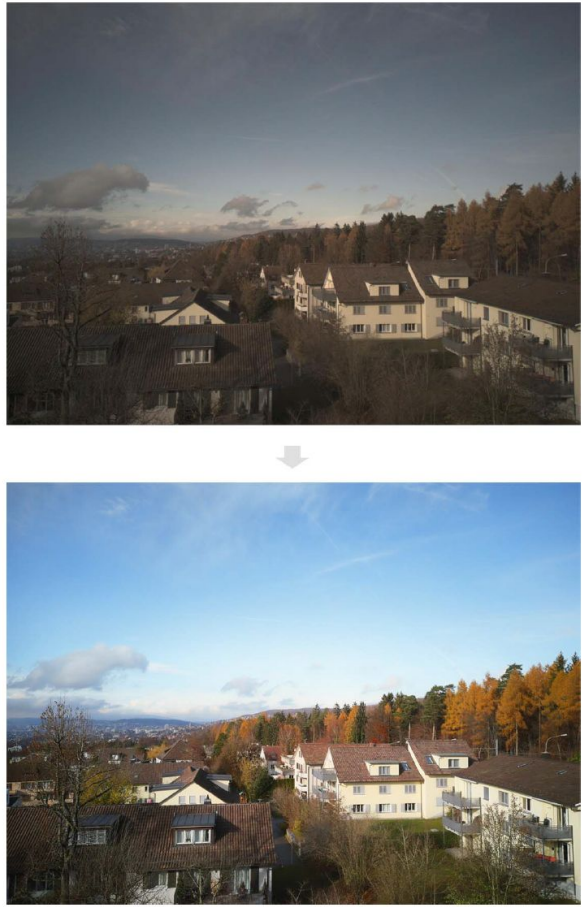
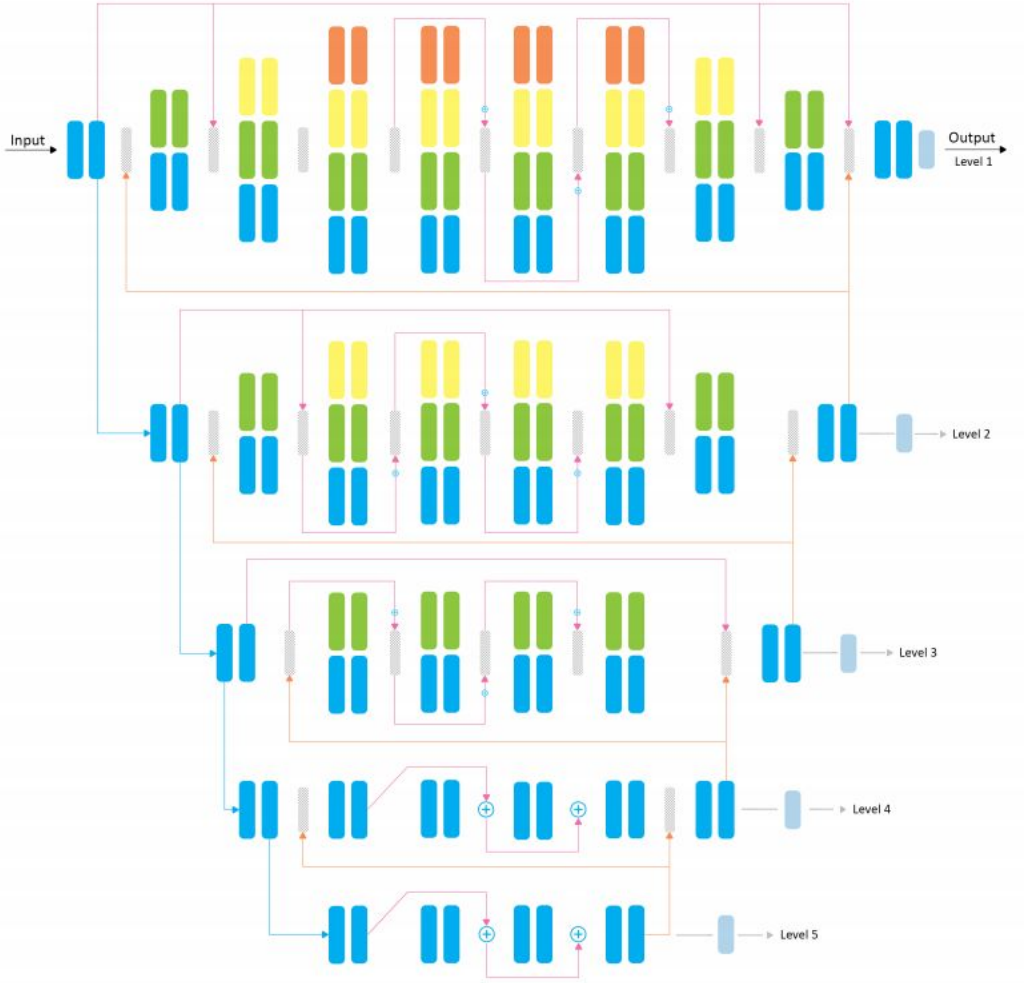
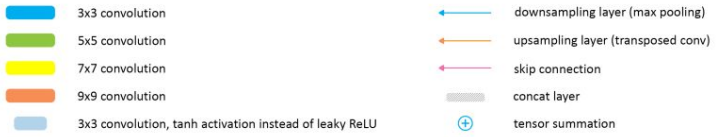


Figure 1. Huawei P20 RAW photo (visualized) and the corresponding image reconstructed with our method.

Figure 6. Visual results obtained with 7 different architectures. From left to right, top to bottom: visualized RAW photo, SRCNN [10], VDSR [27], SRGAN [31], Pix2Pix [25], U-Net [45], DPED [19], our PyNET architecture, Huawei ISP image and the target Canon photo.

Pynet


Input
Output
Level 1
Level 2
Level 3
Level 4
Level 5


3x3 convolution
5x5 convolution
7x7 convolution
9x9 convolution
3x3 convolution, tanh activation instead of leaky ReLU
downsampling layer (max pooling)
upsampling layer (transposed conv)
skip connection
concat layer
tensor summation

PyNET-CA: Enhanced PyNET with Channel Attention for End-to-end Mobile Image Signal Processing

Byung-Hoon Kim[1], Joonyoung Song[1], Jong Chul Ye[1], and JaeHyun Baek[2]

[1] Korea Advanced Institute of Science and Technology, Daejeon, South Korea
{egyptdj,songjy18,jong.ye}@kaist.ac.kr
[2] Amazon Web Services, Seoul, South Korea
jakemraz100@gmail.com

Arxiv 2021.4

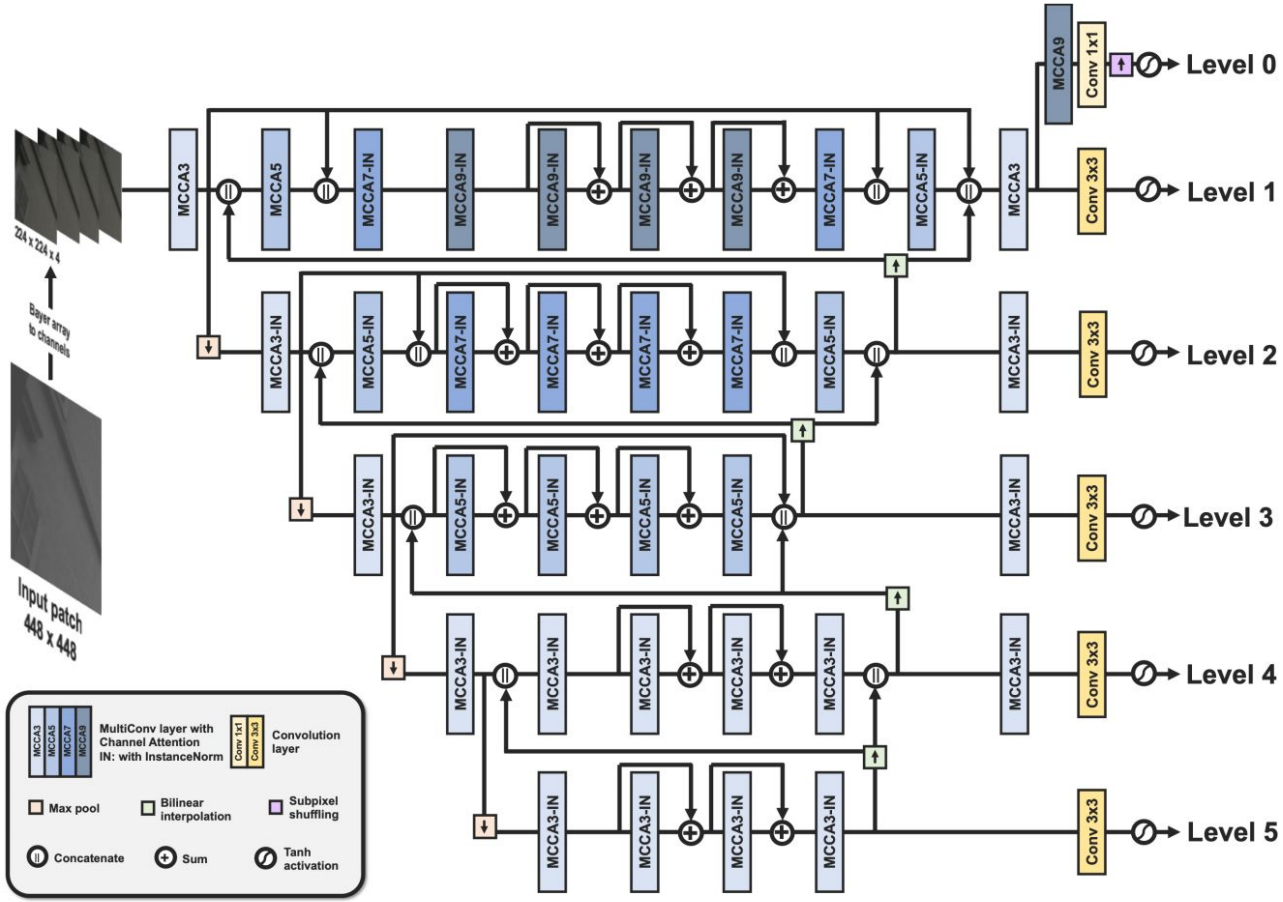


Fig. 3. Schematic illustration of the PyNET-CA model.

A Review of an Old Dilemma: Demosaicking First, or Denoising First?

Qiyu Jin
School of mathematical science,
Inner Mongolia University
qyjin2015@aliyun.com

Gabriele Facciolo
Centre Borelli,
ENS Paris-Saclay, CNRS
facciolo@cmla.ens-cachan.fr

Jean-Michel Morel
Centre Borelli,
ENS Paris-Saclay, CNRS
moreljeanmichel@gmail.com

못읽음

CycleISP: Real Image Restoration via Improved Data Synthesis

Syed Waqas Zamir[1] Aditya Arora[1] Salman Khan[1] Munawar Hayat[1]
Fahad Shahbaz Khan[1] Ming-Hsuan Yang[2,3] Ling Shao[1]
[1]Inception Institute of Artificial Intelligence, UAE
[2]University of California, Merced [3]Google Research

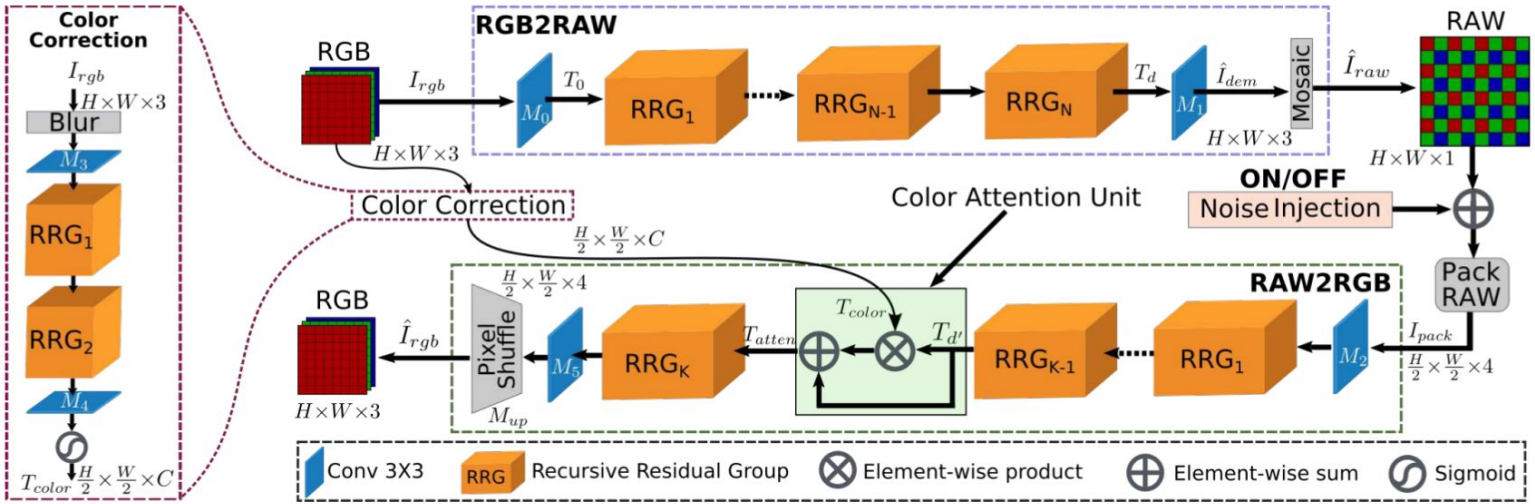


Figure 2: Our CycleISP models the camera imaging pipeline in both directions. It comprises two main branches: RGB2RAW and RAW2RGB. The RGB2RAW branch converts sRGB images to RAW measurements, whereas the RAW2RGB branch transforms RAW data to sRGB images. The auxiliary color correction branch provides explicit color attention to RAW2RGB network. The noise injection module is switched OFF while training the CycleISP (Section 3), and switched ON when synthesizing noise data (Section 4).

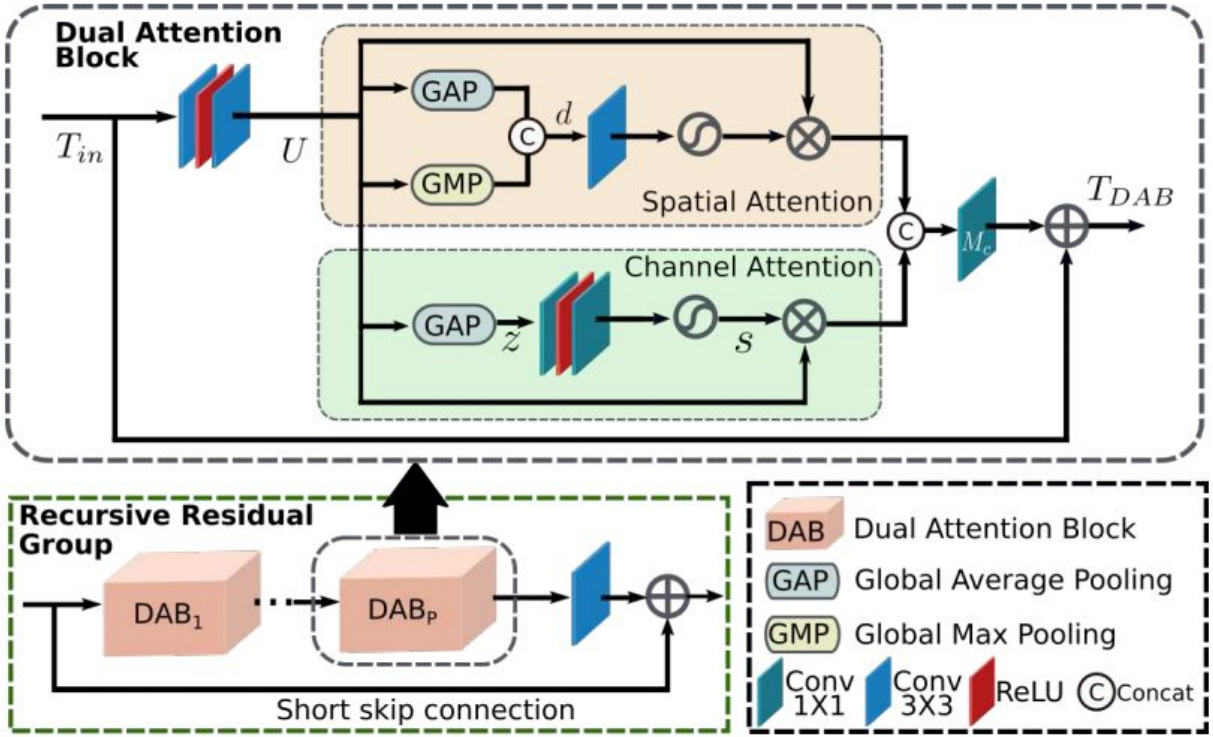


Figure 3: Recursive residual group (RRG) contains multiple dual attention blocks (DAB). Each DAB contains spatial attention and channel attention modules.

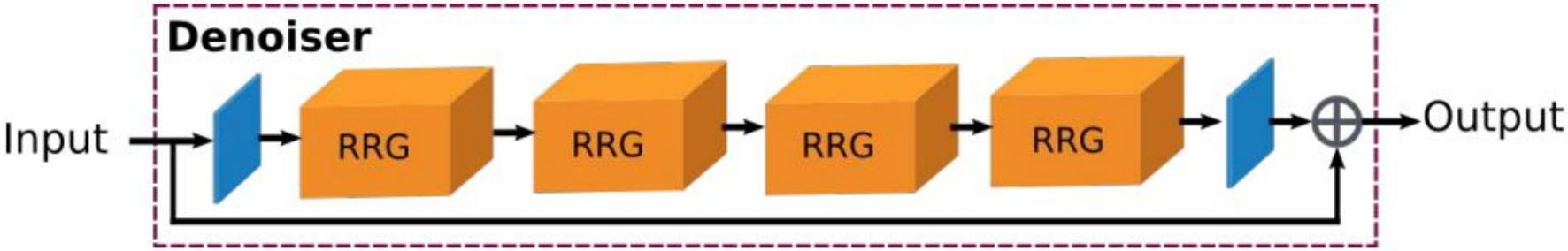
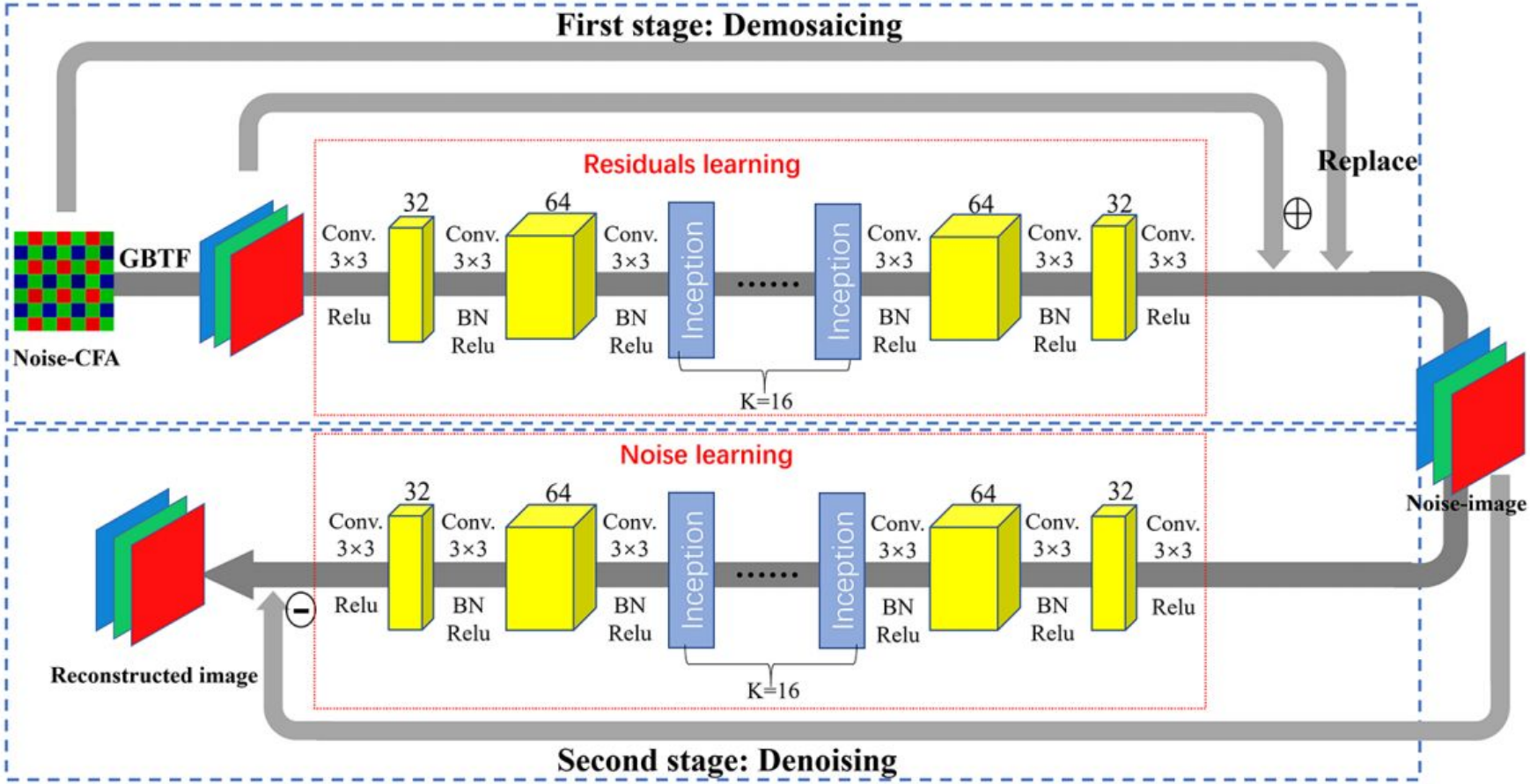


Figure 5: Proposed denoising network. It has the same network structure for denoising both RAW images and sRGB images, except in the handling of input and output.

Residual Learning for Effective joint Demosaicking-Denoising

Yu Guo [1] Qiyu Jin [1] Gabriele Facciolo [2] Tieyong Zeng [3] Jean-Michel Morel [2]
[1] School of Mathematical Science, Inner Mongolia University, China
[2] Centre Borelli, ENS Paris-Saclay, CNRS, France
[3] Department of Mathematics, The Chinese University of Hong Kong, Satin, Hong Kong
{yuguomath, qyjin2015}@aliyun.com, gabriele.facciolo@ens-paris-saclay.fr,
zeng@math.cuhk.edu.hk, moreljeanmichel@gmail.com

Arxiv 2020.11


First stage: Demosaicing
Residuals learning
Replace
GBTF
Noise-CFA
Conv. 3×3
Relu
32
64
BN
Relu
Inception
K=16
Noise-image
Noise learning
Reconstructed image
Second stage: Denoising

CameraNet: A Two-Stage Framework for Effective Camera ISP Learning

Zhetong Liang, Jianrui Cai, Zisheng Cao, and Lei Zhang, *Fellow, IEEE*

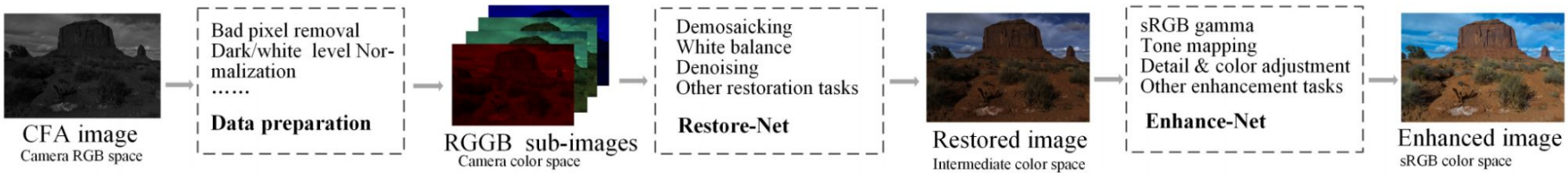


Fig. 3. The proposed CameraNet system for ISP learning.

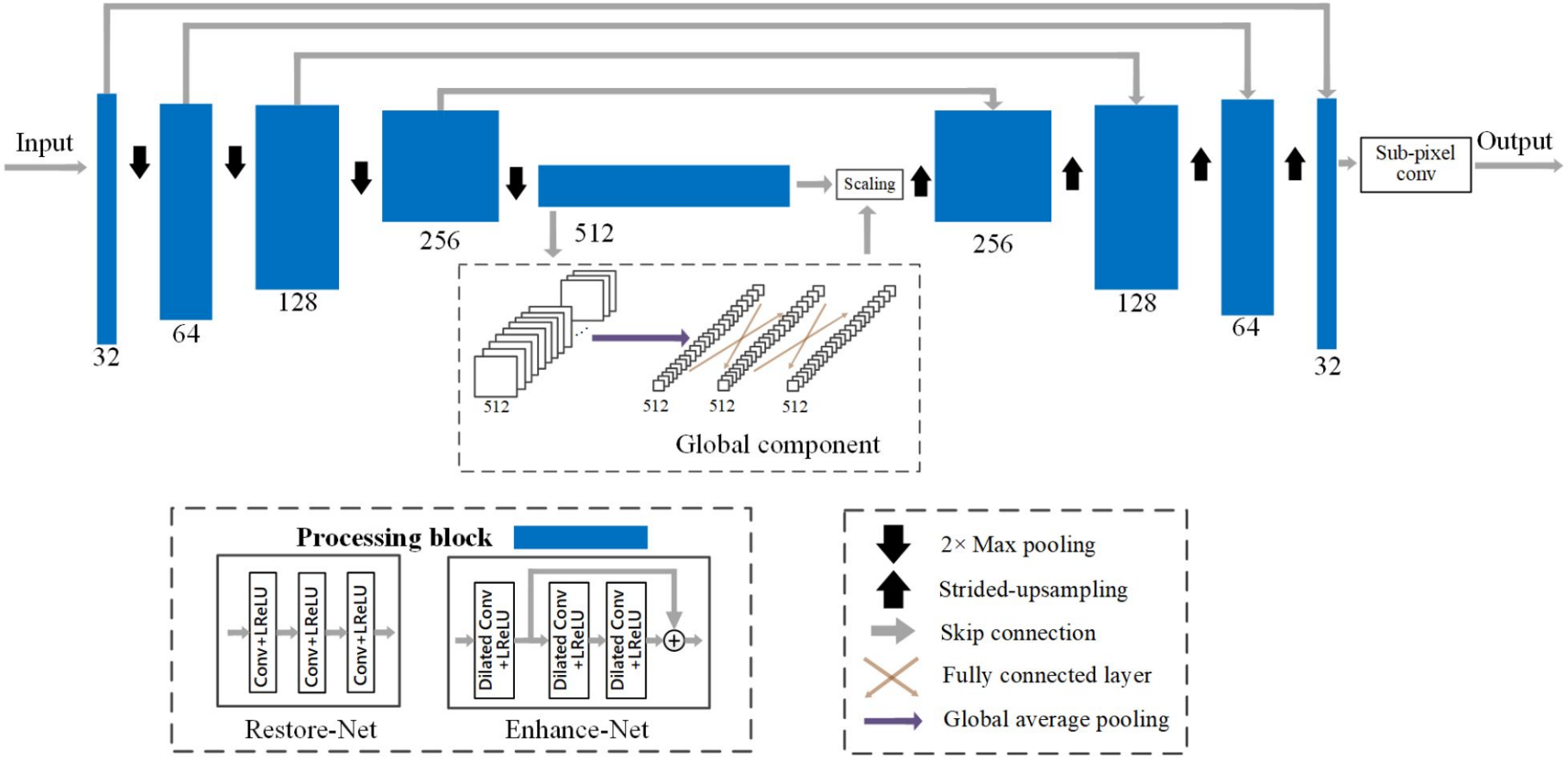


Fig. 4. The structure of UNet-like Restore-Net and Enhance-Net modules in the proposed CameraNet system.

	GFLOPS	Running time (sec.)	Number of parameters (mill.)
CameraNet	**3306.69**	**0.892**	26.53
DeepISP-Net [18]	12869.79	2.12	0.629
DeepCamera [19]	4460.35	1.62	**0.467**

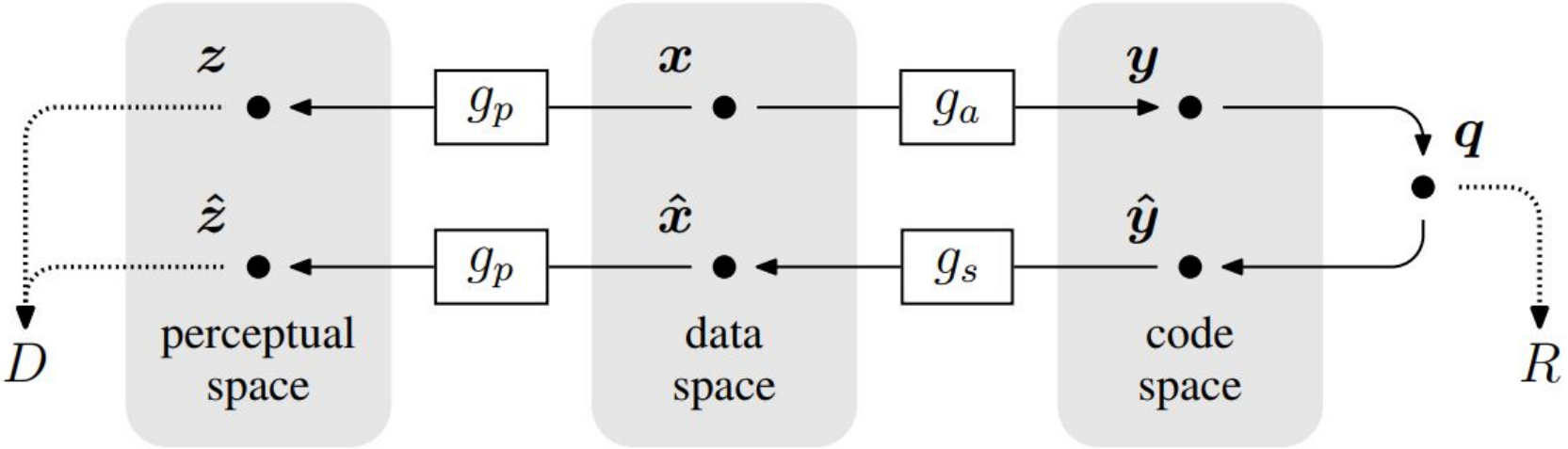
Image Reconstruction
Image Compression

END-TO-END OPTIMIZED IMAGE COMPRESSION

Johannes Ballé*
Center for Neural Science
New York University
New York, NY 10003, USA
johannes.balle@nyu.edu

Valero Laparra
Image Processing Laboratory
Universitat de València
46980 Paterna, Spain
valero.laparra@uv.es

Eero P. Simoncelli*
Center for Neural Science and Courant Institute of Mathematical Sciences
New York University
New York, NY 10003, USA
eero.simoncelli@nyu.edu


$\boldsymbol{z}$
$\boldsymbol{x}$
$\boldsymbol{y}$
$\boldsymbol{q}$
g_p
g_a
$\hat{\boldsymbol{z}}$
$\hat{\boldsymbol{x}}$
$\hat{\boldsymbol{y}}$
g_p
g_s
D
perceptual space
data space
code space
R

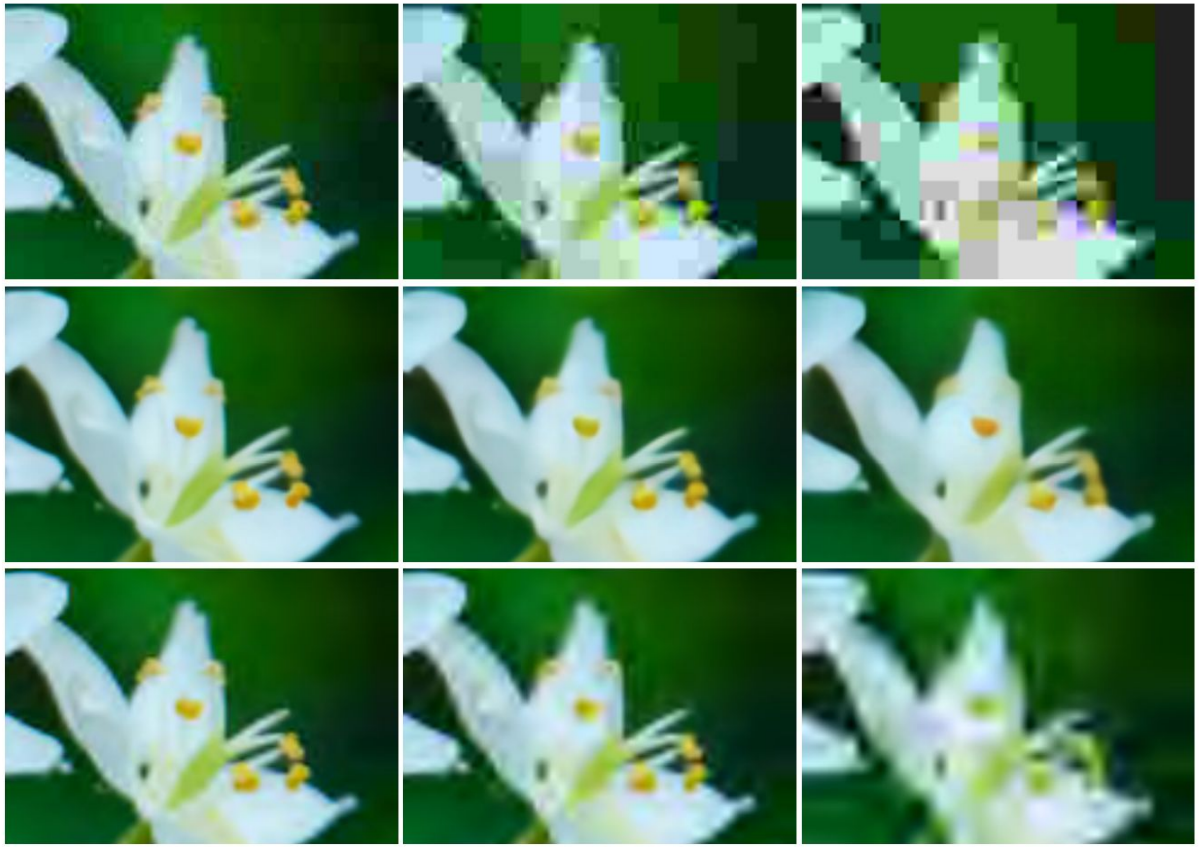


Figure 6: Cropped portion of an image compressed at three different bit rates. Middle row: the proposed method, at three different settings of λ. Top row: JPEG, with three different quality settings. Bottom row: JPEG 2000, with three different rate settings. Bit rates within each column are matched.

Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network

Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, Wenzhe Shi
Twitter

{cledig,ltheis,fhuszar,jcaballero,aacostadiaz,aaitken,atejani,jtotz,zehanw,wshi}@twitter.com

CVPR 2017

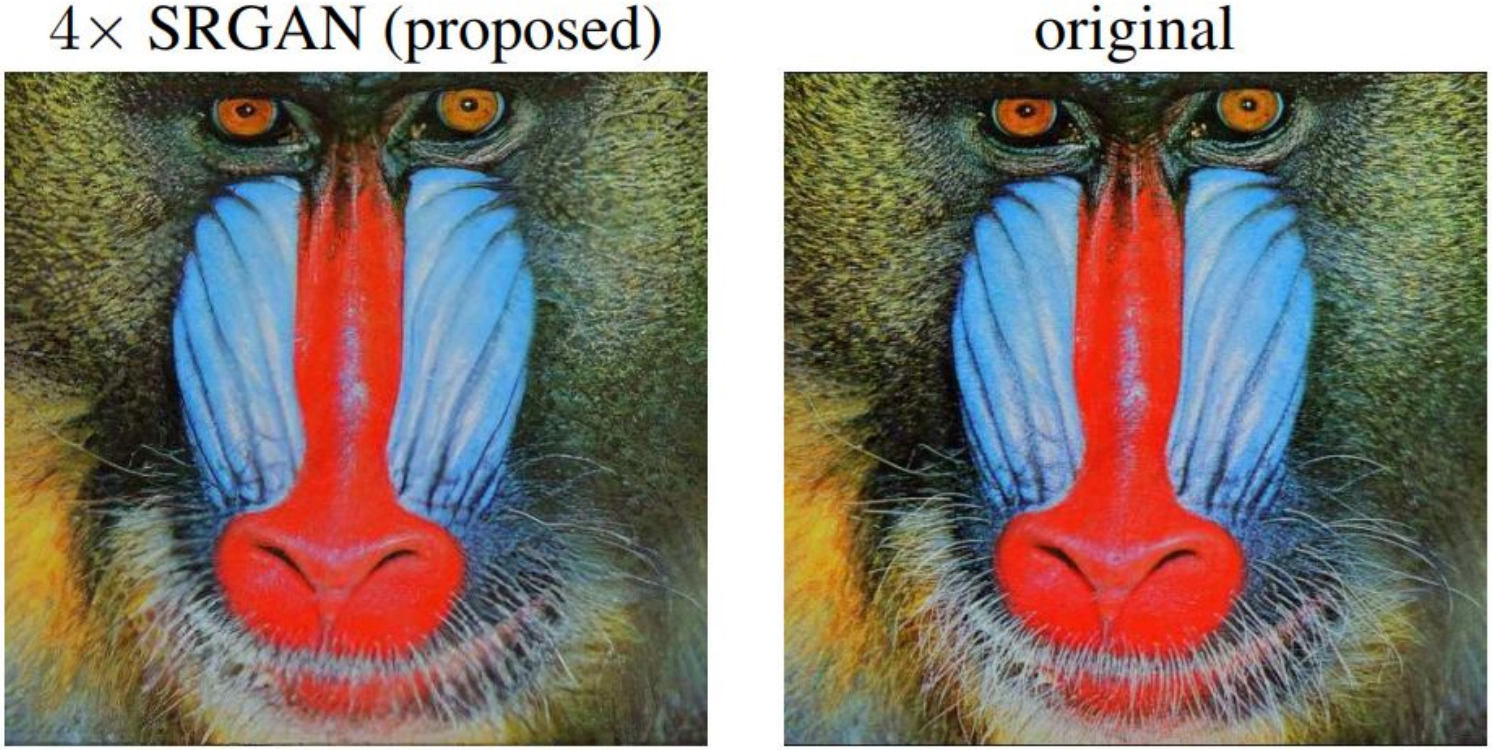


Figure 1: Super-resolved image (left) is almost indistinguishable from original (right). [4× upscaling]

Real-Time Adaptive Image Compression

Oren Rippel [*1] **Lubomir Bourdev** [*1]

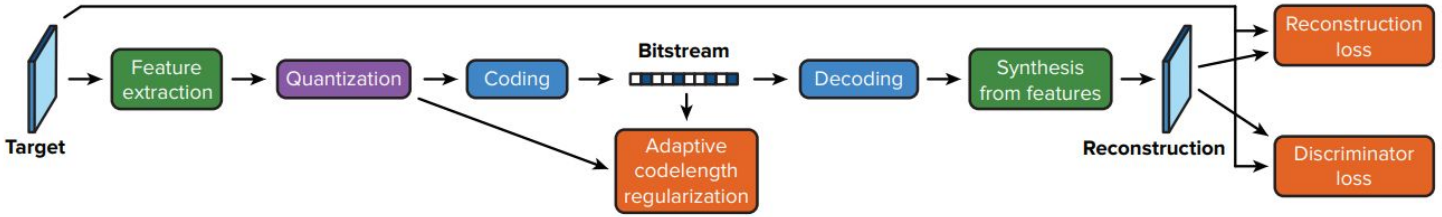


Figure 2. Our overall model architecture. The feature extractor, described in Section 3.1, discovers structure and reduces redundancy via the pyramidal decomposition and interscale alignment modules. The lossless coding scheme, described in Section 3.2, further compresses the quantized tensor via bitplane decomposition and adaptive arithmetic coding. The adaptive codelength regularization then modulates the expected code length to a prescribed target bitrate. Distortions between the target and its reconstruction are penalized by the reconstruction loss. The discriminator loss, described in Section 4, encourages visually pleasing reconstructions by penalizing discrepancies between their distributions and the targets'.

ICML 2017

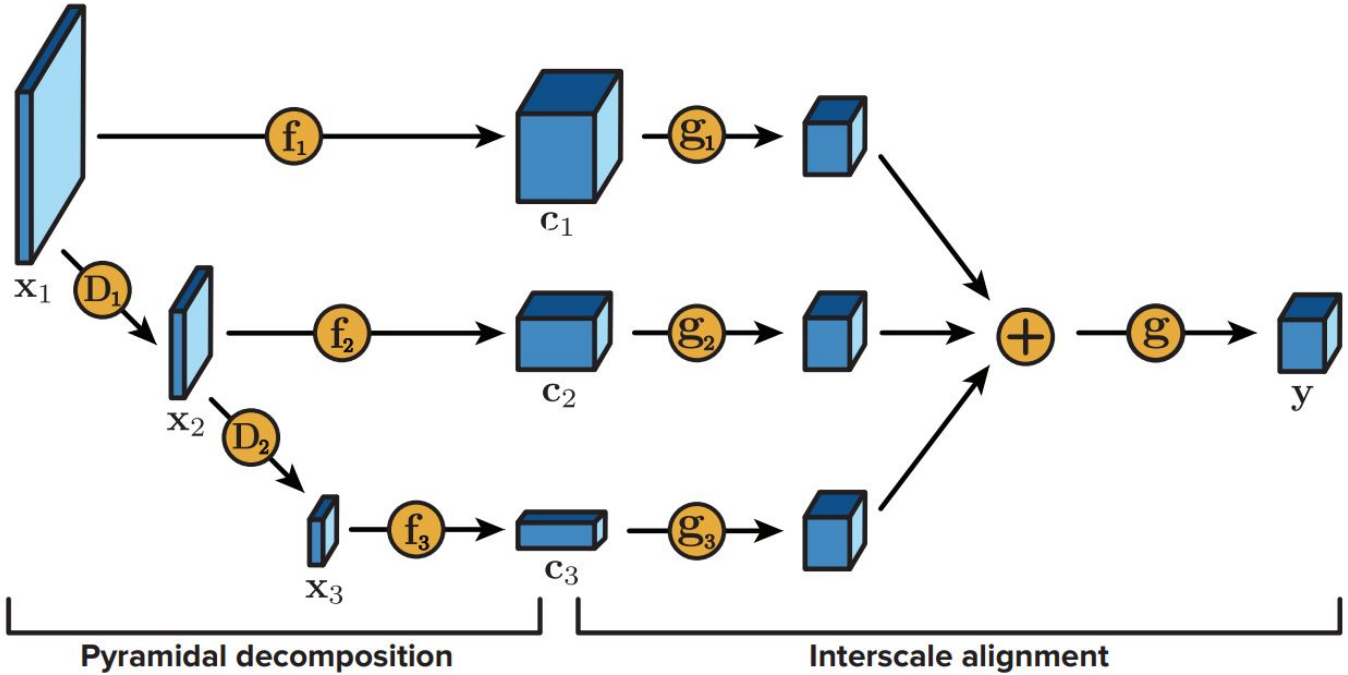


Figure 3. The coefficient extraction pipeline, illustrated for 3 scales. The pyramidal decomposition module discovers structure within individual scales. The extracted coefficient maps are then aligned to discover joint structure across the different scales.

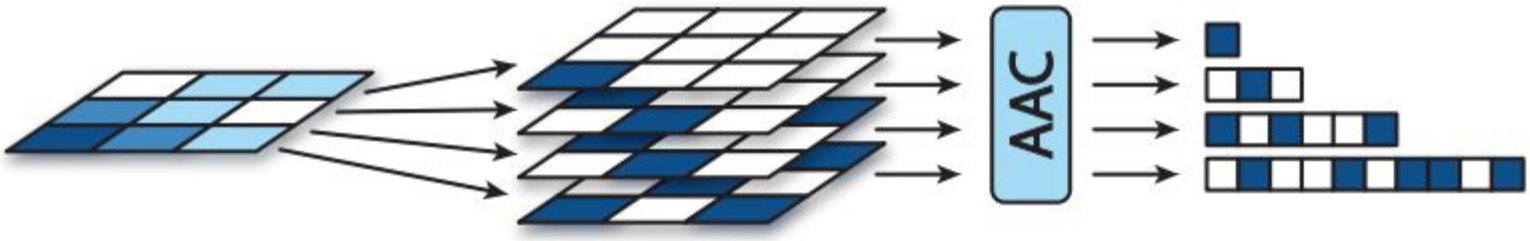


Figure 4. Each of the C spatial maps $\hat{\mathbf{y}}_c \in \mathbb{R}^{H \times W}$ of $\hat{\mathbf{y}}$ is decomposed into B bitplanes as each element $\hat{y}_{chw}$ is expressed in its binary expansion. Each set of bitplanes is then fed to the adaptive arithmetic coder for variable-length encoding. The adaptive codelength regularization enables more compact codes for higher bitplanes by encouraging them to feature higher sparsity.

VARIATIONAL IMAGE COMPRESSION WITH A SCALE HYPERPRIOR

Johannes Ballé* jballe@google.com
David Minnen* dminnen@google.com
Saurabh Singh* saurabhsingh@google.com
Sung Jin Hwang* sjhwang@google.com
Nick Johnston* nickj@google.com
*Google
Mountain View, CA 94043, USA

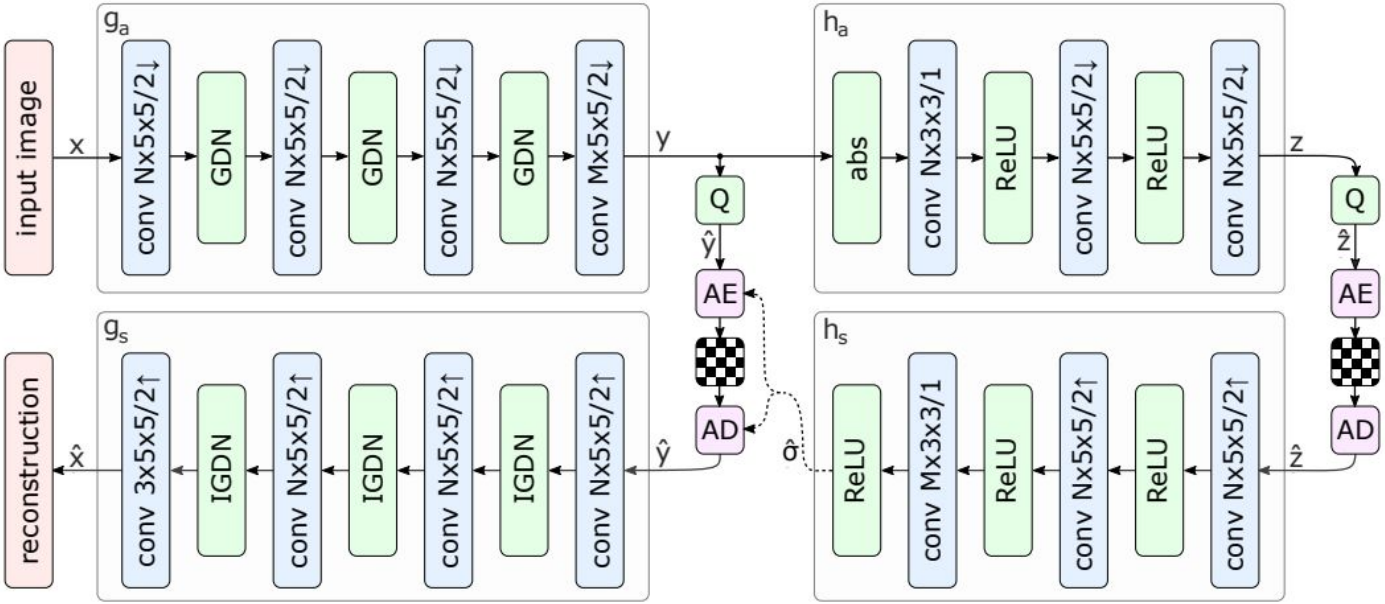


Figure 4: Network architecture of the hyperprior model. The left side shows an image autoencoder architecture, the right side corresponds to the autoencoder implementing the hyperprior. The factorized-prior model uses the identical architecture for the analysis and synthesis transforms g_a and g_s. Q represents quantization, and AE, AD represent arithmetic encoder and arithmetic decoder, respectively. Convolution parameters are denoted as: number of filters $\times$ kernel support height $\times$ kernel support width / down- or upsampling stride, where $\uparrow$ indicates upsampling and $\downarrow$ downsampling. N and M were chosen dependent on λ, with $N = 128$ and $M = 192$ for the 5 lower values, and $N = 192$ and $M = 320$ for the 3 higher values.

Learned Image Compression with Discretized Gaussian Mixture Likelihoods and Attention Modules

Zhengxue Cheng[1], Heming Sun[2,3], Masaru Takeuchi[2], Jiro Katto[1]

[1] Department of Computer Science and Communication Engineering, Waseda University, Tokyo, Japan

[2] Waseda Research Institute for Science and Engineering, Tokyo, Japan [3] JST, PRESTO, 4-1-8 Honcho, Kawaguchi, Saitama, Japan

{zxcheng@asagi., hemingsun@aoni., masaru-t@aoni., katto@}waseda.jp

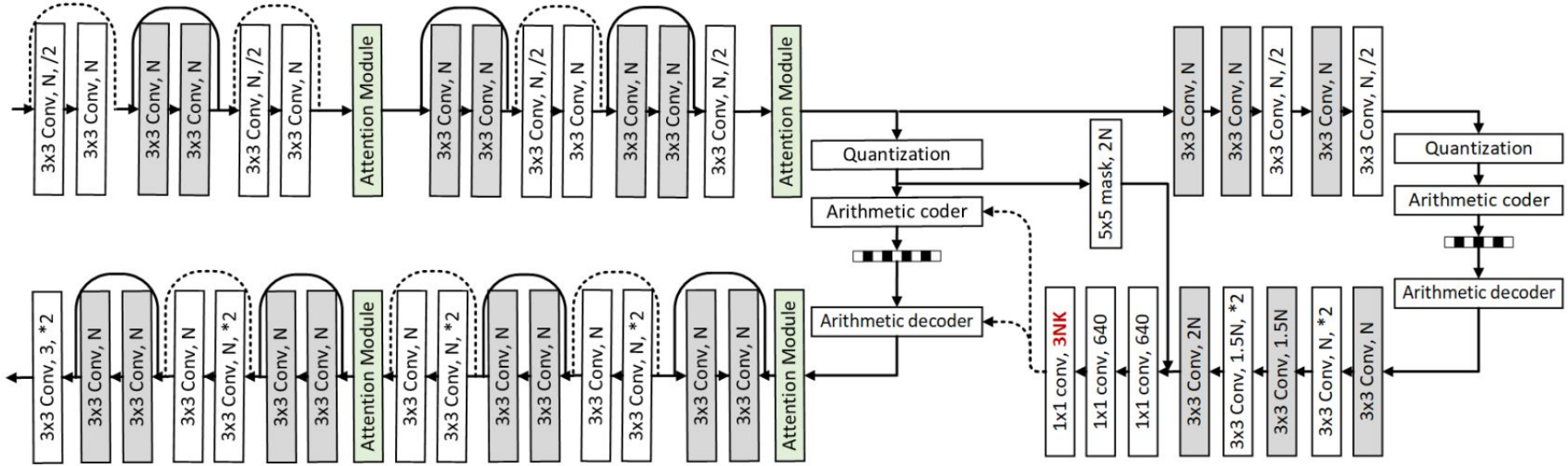


Figure 4: Network architecture.

Compression

GAN Compression: Efficient Architectures for Interactive Conditional GANs

Muyang Li[1,3] Ji Lin[1] Yaoyao Ding[1,3] Zhijian Liu[1] Jun-Yan Zhu[2] Song Han[1]

lmxyy@mit.edu jilin@mit.edu yyding@mit.edu zhijian@mit.edu junzhu@adobe.com songhan@mit.edu

[1]Massachusetts Institute of Technology [2]Adobe Research [3]Shanghai Jiao Tong University

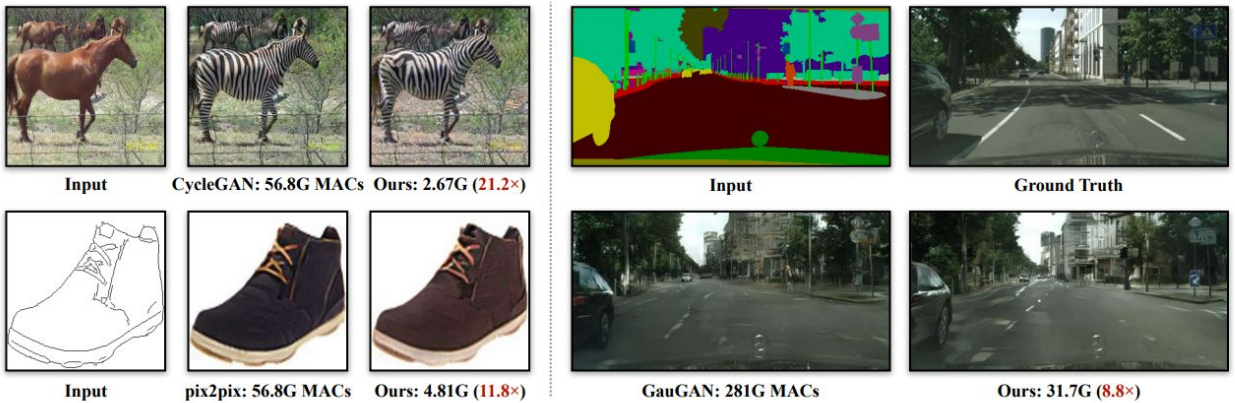


Figure 1: We introduce *GAN Compression*, a general-purpose method for compressing conditional GANs. Our method reduces the computation of widely-used conditional GAN models including pix2pix, CycleGAN, and GauGAN by 9-21× while preserving the visual fidelity. Our method is effective for a wide range of generator architectures, learning objectives, and both paired and unpaired settings.

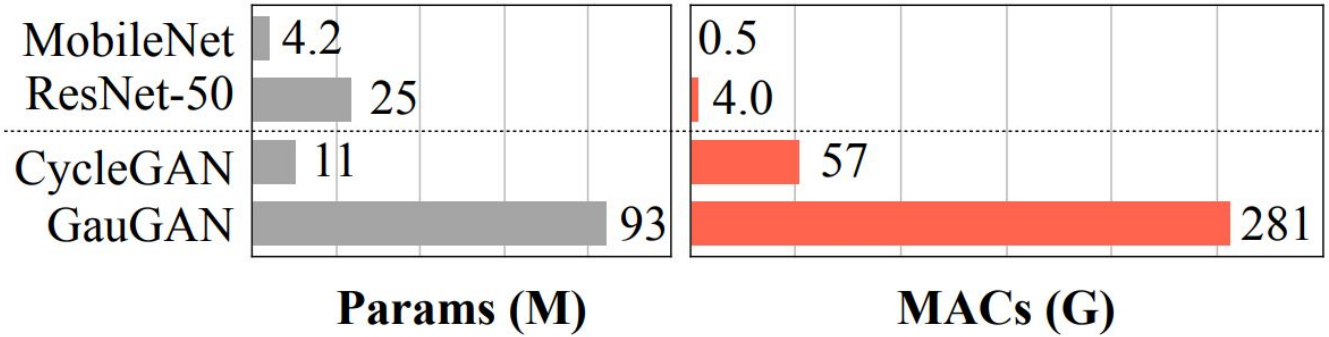


Figure 2: Conditional GANs require two orders of magnitude more computation than image classification CNNs, making it prohibitive to be deployed on edge devices.

*We use the number of Multiply-Accumulate Operations (MAC) to quantify the computation cost. Modern computer architectures use fused multiply–add (FMA) instructions for tensor operations. These instructions compute $a = a + b \times c$ as one operation. 1 MAC=2 FLOPs.

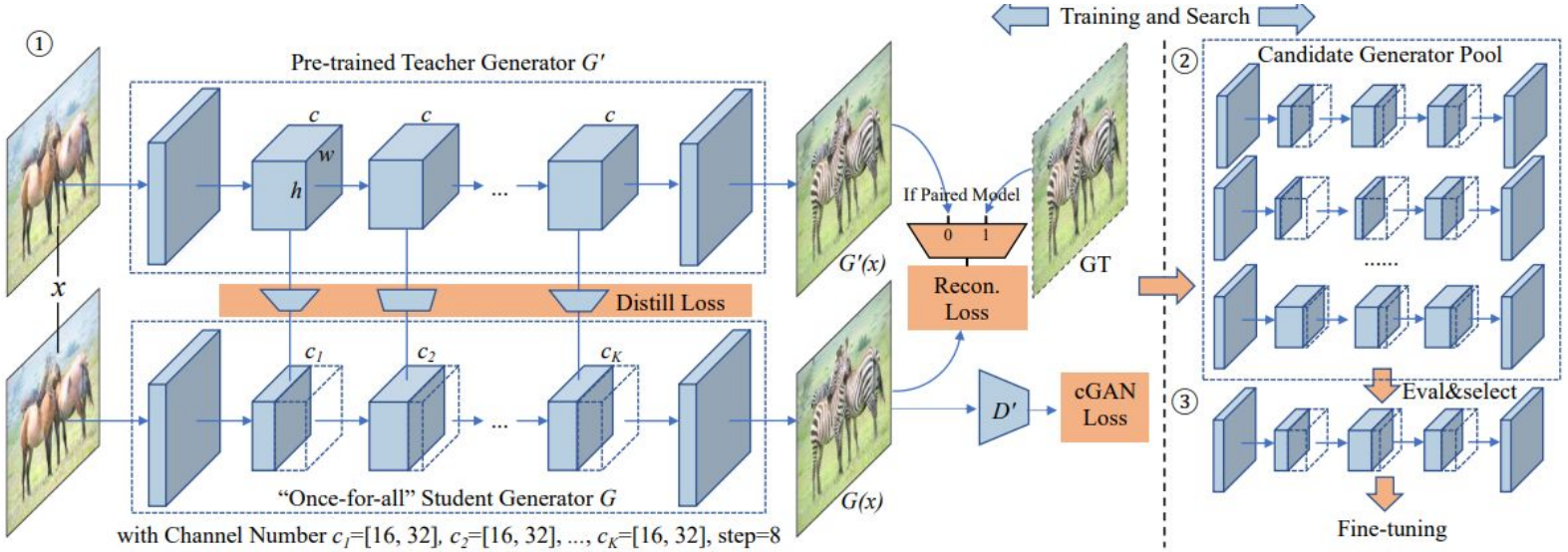


Figure 3: GAN Compression: ① Given a pre-trained teacher generator G', we distill a smaller "*once-for-all*" [6] student generator G that contains all possible channel numbers through weight sharing. We sample different channel numbers $\{c_k\}_{k=1}^{K}$ for G at each training step so that *one* generator can support *all* channel numbers. ② We then extract many sub-generators with different channel numbers from the "once-for-all" generator and evaluate their performance. No retraining is needed, which is the advantage of the "once-for-all" generator. ③ Finally, we choose the best sub-generator given the compression ratio target and performance target (FID or mAP), perform fine-tuning, and obtain the final compressed model.

Model	Dataset	Method	#Parameters		MACs		Metric			
							FID (↓)		mAP (↑)	
CycleGAN	horse→zebra	Original	11.3M	–	56.8G	–	61.53	–	–	
		Shu *et al.* [52]	–	–	13.4G	(4.2×)	96.15	(34.6 ☹)	–	
		Ours (w/o fine-tuning)	0.34M	(**33.3**×)	2.67G	(**21.2**×)	64.95	(**3.42** ☹)	–	
		Ours	0.34M	(33.3×)	2.67G	(21.2×)	71.81	(10.3 ☹)	–	
Pix2pix	edges→shoes	Original	11.3M	–	56.8G	–	24.18	–	–	
		Ours (w/o fine-tuning)	0.70M	(16.3×)	4.81G	(11.8×)	31.30	(7.12 ☹)	–	
		Ours	0.70M	(**16.3**×)	4.81G	(**11.8**×)	26.60	(**2.42** ☹)	–	
	cityscapes	Original	11.3M	–	56.8G	–	–		35.62	–
		Ours (w/o fine-tuning)	0.71M	(16.0×)	5.66G	(10×)	–		29.27	(6.35 ☹)
		Ours	0.71M	(**16.0**×)	5.66G	(**10.0**×)	–		34.34	(**1.28** ☹)
	map→arial photo	Original	11.3M	–	56.8G	–	47.76	–	–	
		Ours (w/o fine-tuning)	0.75M	(15.1×)	4.68G	(11.4×)	71.82	(24.1 ☹)	–	
		Ours	0.75M	(**15.1**×)	4.68G	(**11.4**×)	48.02	(**0.26** ☹)	–	
GauGAN	cityscapes	Original	93.0M	–	281G	–	–		58.89	–
		Ours (w/o fine-tuning)	20.4M	(4.6×)	31.7G	(8.8×)	–		56.75	(2.14 ☹)
		Ours	20.4M	(**4.6**×)	31.7G	(**8.8**×)	–		58.41	(**0.48** ☹)

Table 1: Quantitative evaluation of GAN Compression: We use the mAP metric (the higher the better) for the Cityscapes dataset and FID (the lower the better) for other datasets. Our method can compress state-of-the-art conditional GANs by **9-21**× in MACs and **5-33**× in model size, with only minor performance degradation. For CycleGAN compression, our systematic approach outperforms previous CycleGAN-specific Co-Evolution method [52] by a large margin.

Fréchet inception distance

Mean Average Precision


Input
Ground
-truth
Original
Model
(GauGAN)
mAP: 58.9
GAN
Compression
(8.8×)
mAP: 58.4

ECCV 2020

GAN Slimming: All-in-One GAN Compression by a Unified Optimization Framework

Haotao Wang[1], Shupeng Gui[2], Haichuan Yang[2], Ji Liu[3], and Zhangyang Wang[1(✉)]

[1] University of Texas at Austin, Austin, TX 78712, USA
{htwang,atlaswang}@utexas.edu
[2] University of Rochester, Rochester, NY 14627, USA
{sgui2,hyang36}@ur.rochester.edu
[3] AI Platform, Ytech Seattle AI Lab, FeDA Lab, Kwai Inc., Seattle, WA 98004, USA
ji.liu.uwisc@gmail.com

Abstract. Generative adversarial networks (GANs) have gained increasing popularity in various computer vision applications, and recently start to be deployed to resource-constrained mobile devices. Similar to other deep models, state-of-the-art GANs suffer from high parameter complexities. That has recently motivated the exploration of compressing GANs (usually generators). Compared to the vast literature and prevailing success in compressing deep classifiers, the study of GAN compression remains in its infancy, so far leveraging individual compression techniques instead of more sophisticated combinations. We observe that due to the notorious instability of training GANs, heuristically stacking different compression techniques will result in unsatisfactory results. To this end, we propose the first unified optimization framework combining multiple compression means for GAN compression, dubbed **GAN Slimming** (GS). GS seamlessly integrates three mainstream compression techniques: model distillation, channel pruning and quantization, together with the GAN minimax objective, into one unified optimization form, that can be efficiently optimized from end to end. Without bells and whistles, GS largely outperforms existing options in compressing image-to-image translation GANs. Specifically, we apply GS to compress CartoonGAN, a state-of-the-art style transfer network, by up to **47**× times, with minimal visual quality degradation. Codes and pre-trained models can be found at `https://github.com/TAMU-VITA/GAN-Slimming`.

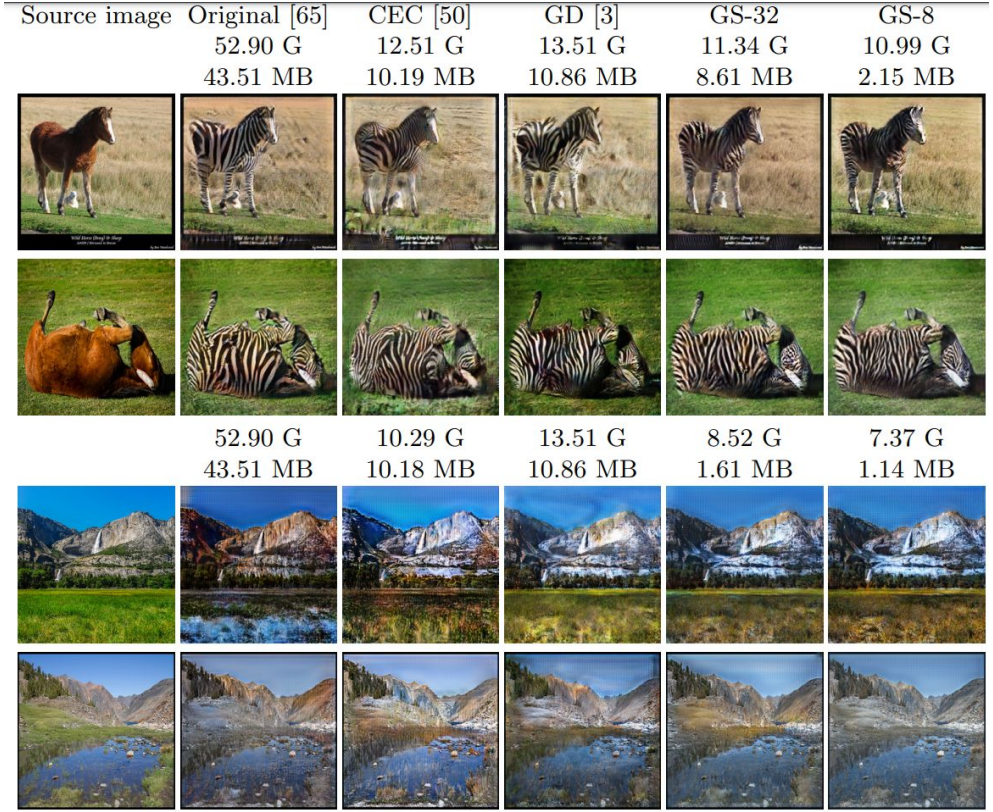


Fig. 2: CycleGAN compression results. Top two rows: horse-to-zebra task. Bottom two rows: summer-to-winter task. Six columns from left to right: source image, style transfer results by original CycleGAN, CEC, GD, GS-32 and GS-8 respectively. FLOPs (in G) and model size (in MB) of each method on each task are annotated above the images.

Self-Supervised Generative Adversarial Compression

Chong Yu
NVIDIA
chongy@nvidia.com

Jeff Pool
NVIDIA
jpool@nvidia.com

NIPS 2020

Table 1: GAN compression algorithms comparison

Technique	Generator(s)		Discriminator		Loss Terms				Results	
	Compressed	Init Scheme	Init Scheme	Fixed	L-Gc	L-Dc	L-Go	L-Do	Qualitative	FID Score
(a) No Compression [10]	Dense	Random	Dense,Random	No	-	-	Yes	Yes	Good	6.113
(b) Self-Supervised MIS **(ours)**	Dense,Sparse	From Dense	Dense,Pretrained	No	Yes	Yes	Yes	Yes	Good	6.929
(c) Small & Dense Network	Dense	Random	Dense,Random	No	-	-	Yes	Yes	Mode collapse	72.821
(d) One-shot Pruning & Fine-Tuning [11]	Sparse	From Dense	Dense,Pretrained	No	Yes	Yes	-	-	Facial artifacts	24.404
(e) Gradual Pruning & Fine-Tuning [12]	Sparse	From Dense	Dense,Random	No	Yes	Yes	-	-	Facial artifacts	35.677
(f) Gradual Pruning during Training [12]	Sparse	Random	Dense,Random	No	Yes	Yes	-	-	No faces	84.941
(g) One-shot Pruning & Distillation [2]	Dense,Sparse	From Dense	-	-	Yes	-	Yes	-	Mode collapse	45.461
(h) (d) & Distillation [2]	Dense,Sparse	From Dense	Dense,Pretrained	No	Yes	Yes	Yes	-	Color artifacts	38.985
(i) (g) & Fix Original Loss	Dense,Sparse	From Dense	Dense,Pretrained	Yes	Yes	Yes	-	-	Facial artifacts	15.182
(j) Adversarial Learning [27]	Dense,Sparse	Random	Dense,Random	No	Yes	Yes	Yes	Yes	Mode collapse	92.721
(k) Knowledge Distillation [28]	Dense,Sparse	From Dense	Dense,Random	No	Yes	-	Yes	Yes	Mode collapse	103.094
(l) Distill Intermediate (LIT) [29]	Dense,Sparse	From Dense	Dense,Pretrained	Yes	-	-	-	-	Mode collapse	61.150
(m) E-M Pruning [30]	Dense,Sparse	From Dense	Sparse,Pretrained	No	Yes	Yes	Yes	-	Color artifacts	159.767
(n) G & D Both Pruning [31]	Dense,Sparse	From Dense	Sparse,Pretrained	No	Yes	Yes	Yes	-	Mode collapse	46.453

(a)

(b)

(c)

(d)

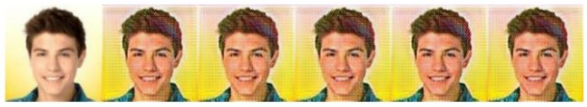
(e)

(f)

(g)

(h)

(i)



(j)

(k)

(l)

(m)

(n)

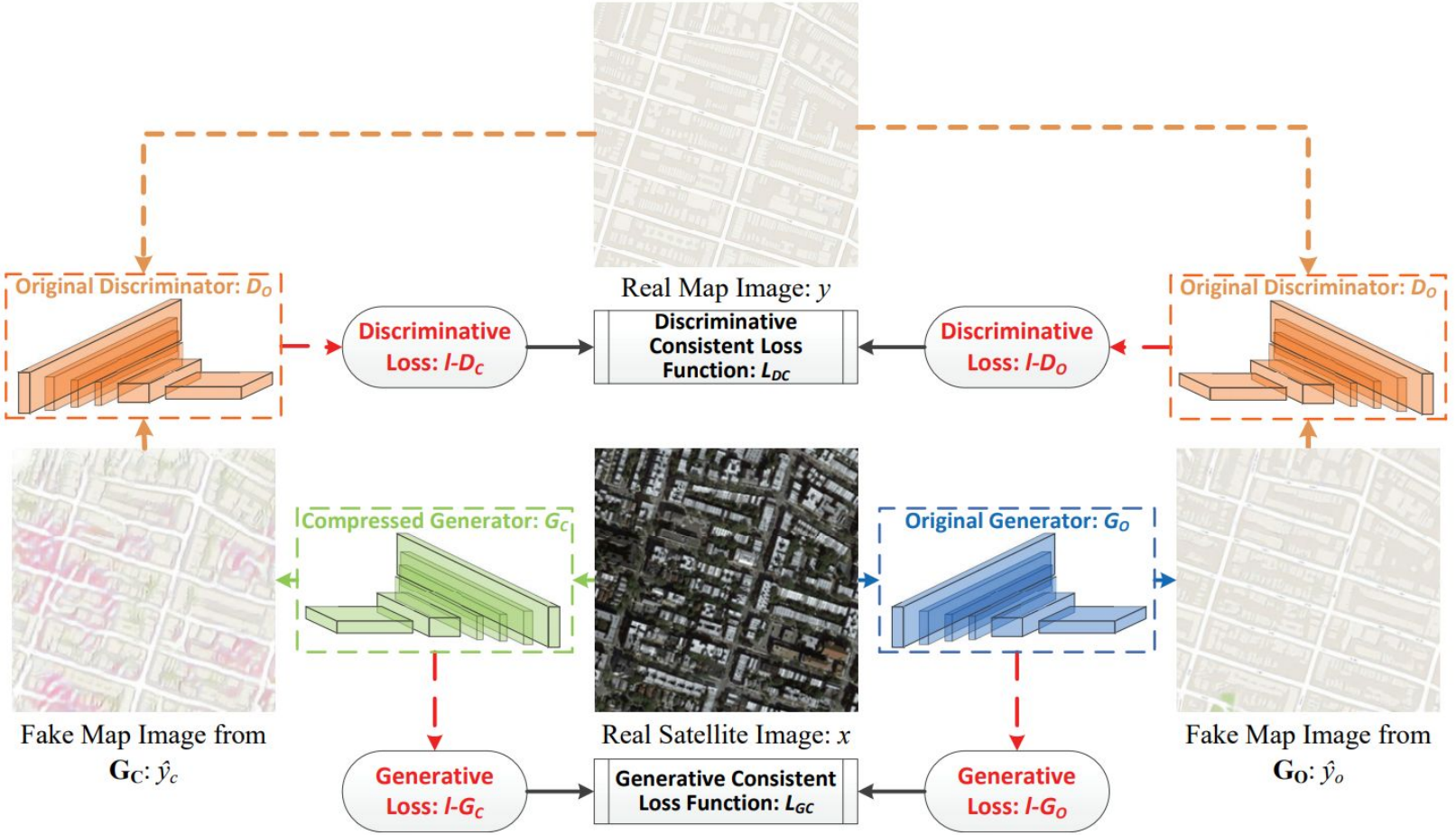


Figure 2: Workflow of self-supervised (by the original discriminator) GAN compression.

Image Synthesis. We apply the proposed compression method to DCGAN [5][3], a network that learns to synthesize novel images from some distribution. We task DCGAN with generating images that could belong to the MNIST data set, with results shown in Figure 3.

Domain Translation. We apply the proposed compression method to pix2pix [39][4], an approach to learn the mapping between paired training examples by applying conditional adversarial networks. In our experiment, the task is synthesizing fake satellite images from label maps and vice-versa. Representative results of this bidirectional task are shown in Figure 4.

Style Transfer. We apply the proposed compression method to CycleGAN [9], used to exchange the style of images from a source domain to a target domain in the absence of paired training examples. In our experiment, the task is to transfer the style of real photos with that of Monet's paintings. Representative results of this bidirectional task are shown in Figure 5: photographs are given the style of Monet's paintings and vice-versa.

Image-to-image Translation. In addition to the StarGAN results above (Section 3, Figure 1), we apply the proposed compression method to CycleGAN [9] performing bidirectional translation between images of zebras and horses. Results are shown in Figure 6.

Super Resolution. We apply self-supervised compression to SRGAN [40][5], which uses a discriminator network trained to differentiate between upscaled and the original high-resolution images. We trained SRGAN on the DIV2K data set [41], and use the DIV2K validation images, as well as Set5 [42] and Set14 [43] to report deployment quality. In this task, quality is often evaluated by two

Table 3: *PSNR* (dB), *SSIM* and *FID* indicators for validation datasets

Dataset	Original Generator			50% Filter-Compressed G			50% Element-Compressed G			90% Element-Compressed G		
	PSNR	SSIM	FID	PSNR	SSIM	FID	PSNR	SSIM	FID	PSNR	SSIM	FID
Set5	31.063	0.853	30.762	30.234	0.860	39.514	30.484	0.862	36.824	30.301	0.861	37.475
Set14	27.643	0.716	55.457	27.315	0.745	82.118	27.417	0.744	70.126	27.369	0.743	80.684
DIV2K	29.206	0.778	14.653	28.876	0.801	18.500	28.975	0.801	16.609	28.868	0.798	18.263

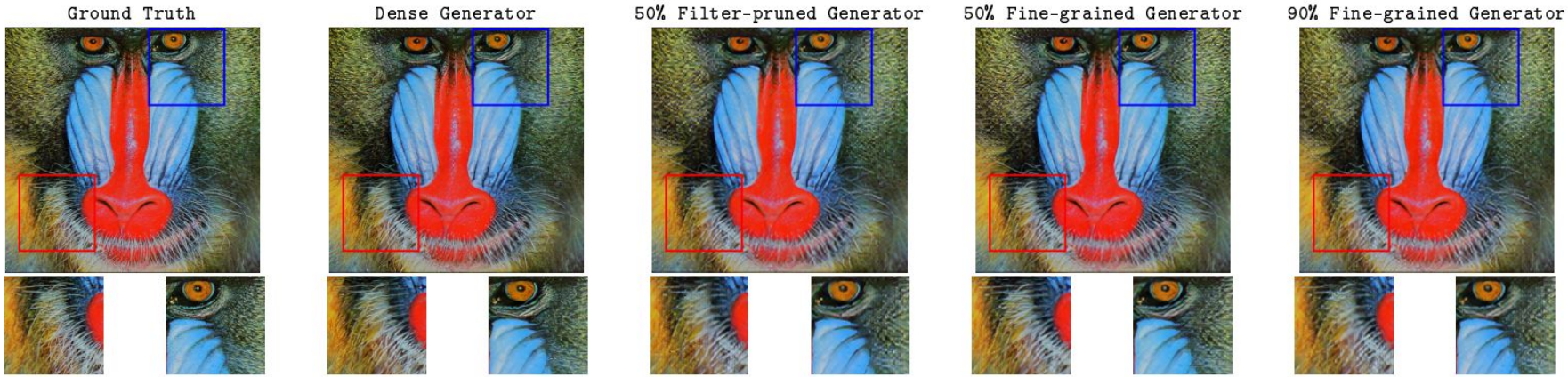


Figure 7: Representative super resolution results: SRGAN (with enlargements of boxed areas).

GANs Can Play Lottery Tickets Too

Xuxi Chen[1*], Zhenyu Zhang[1*], Yongduo Sui[1], Tianlong Chen[2]
[1]University of Science and Technology of China, [2]University of Texas at Austin
{chanyh,zzy19969,syd2019}@mail.ustc.edu.cn, tianlong.chen@utexas.edu

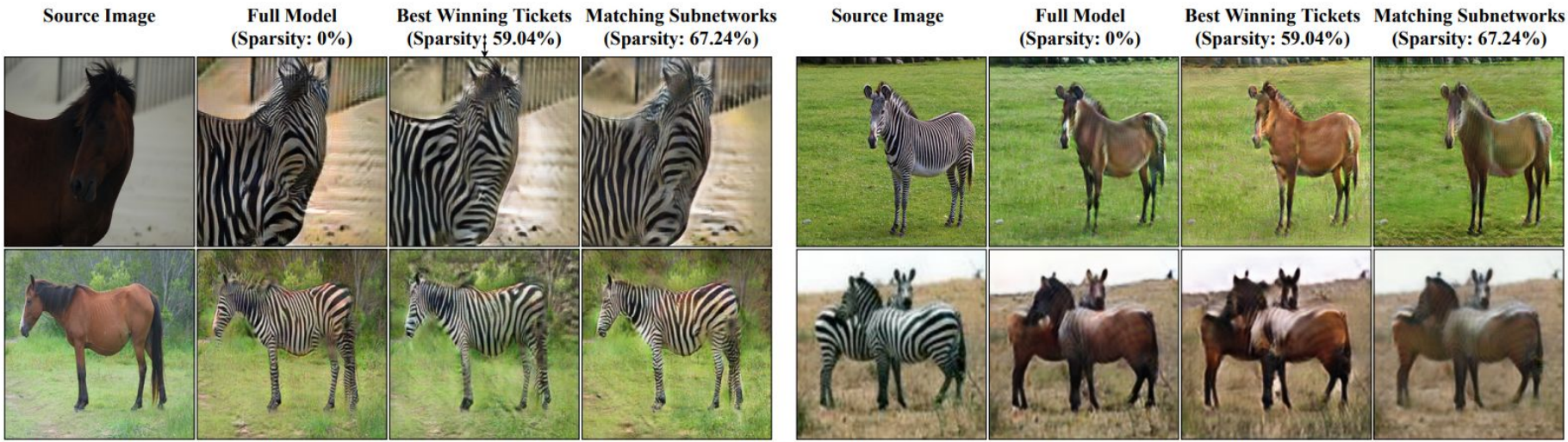


Figure 3: Visualization of CycleGAN Winning Tickets found by IMP. Sparsity of best winning tickets : 59.04%. Extreme sparsity of matching subnetworks: 67.24%. Left: visualization results on horse2zebra. Right: visualization results on zebra2horse.

Model	Benchmark	FID_{Full} (Sparsity)	FID_{Best} (Sparsity)	$\text{S}_{\text{Extreme}}$ (Sparsity)
DCGAN (Radford et al., 2016)	CIFAR-10	57.39 (0%)	49.31 (20.0%)	54.48 (67.2%)
WGAN-GP (Gulrajani et al., 2017)	CIFAR-10	19.23 (0%)	16.77 (36.0%)	17.28 (73.8%)
ACGAN (Odena et al., 2017)	CIFAR-10	39.26 (0%)	31.45 (36.0%)	38.95 (79.0%)
GGAN (Lim & Ye, 2017)	CIFAR-10	38.50 (0%)	33.42 (20.0%)	36.67 (48.8%)
ProjGAN (Miyato & Koyama, 2018)	CIFAR-10	31.47 (0%)	28.19 (20.0%)	31.31 (67.2%)
SAGAN (Zhang et al., 2019)	CIFAR-10	14.73 (0%)	13.57 (20.0%)	14.68 (48.8%)
AutoGAN(A) (Gong et al., 2019)	CIFAR-10	14.38 (0%)	14.04 (36.0%)	14.04 (36.0%)
AutoGAN(B) (Gong et al., 2019)	CIFAR-10	14.62 (0%)	13.16 (20.0%)	14.20 (36.0%)
AutoGAN(C) (Gong et al., 2019)	CIFAR-10	13.61 (0%)	13.41 (48.8%)	13.41 (48.8%)
DiffAugGAN (Zhao et al., 2020b)	CIFAR-10	8.23 (0%)	8.05 (48.8%)	8.05 (48.8%)

Table 6: Results on other GAN models on Tiny ImageNet. FID_{Full}: FID score of the full model. FID_{Best}: The minimal FID score of all subnetworks. $\text{FID}_{\text{Extreme}}$: The FID score of matching networks at extreme sparsity level.

Model	Benchmark	FID_{Full} (Sparsity)	FID_{Best} (Sparsity)	$\text{S}_{\text{Extreme}}$ (Sparsity)
DCGAN (Radford et al., 2016)	Tiny ImageNet	121.35 (0%)	78.51 (36.0%)	114.00 (67.2%)
WGAN-GP (Gulrajani et al., 2017)	Tiny ImageNet	211.77 (0%)	194.72 (48.8%)	200.22 (67.2%)

Pruning is beneficial

The State of Sparsity in Deep Neural Networks

Trevor Gale [*1 †] **Erich Elsen** [*2] **Sara Hooker** [1 †]

Sparse GPU Kernels for Deep Learning

Trevor Gale
Stanford University
United States of America
tgale@cs.stanford.edu

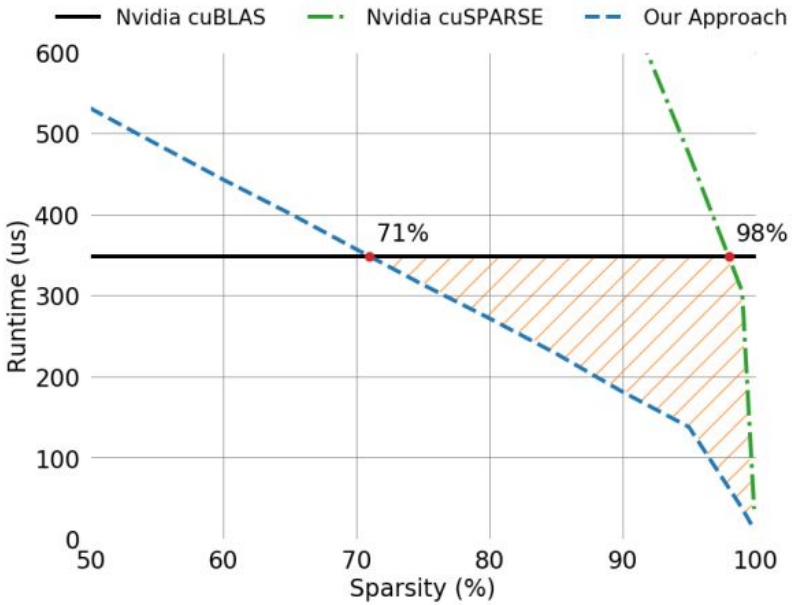
Matei Zaharia
Stanford University
United States of America
matei@cs.stanford.edu

Cliff Young
Google Brain
United States of America
cliffy@google.com

Erich Elsen
DeepMind
United Kingdom
eriche@google.com

***Abstract*—Scientific workloads have traditionally exploited high levels of sparsity to accelerate computation and reduce memory requirements. While deep neural networks can be made sparse, achieving practical speedups on GPUs is difficult because these applications have relatively moderate levels of sparsity that are not sufficient for existing sparse kernels to outperform their dense counterparts. In this work, we study sparse matrices from deep learning applications and identify favorable properties that can be exploited to accelerate computation. Based on these insights, we develop high-performance GPU kernels for two sparse matrix operations widely applicable in neural networks: sparse matrix–dense matrix multiplication and sampled dense–dense matrix multiplication. Our kernels reach 27% of single-precision peak on Nvidia V100 GPUs. Using our kernels, we demonstrate sparse Transformer and MobileNet models that achieve 1.2–2.1× speedups and up to 12.8× memory savings without sacrificing accuracy.**

***Index Terms*—Neural networks, sparse matrices, graphics processing units**



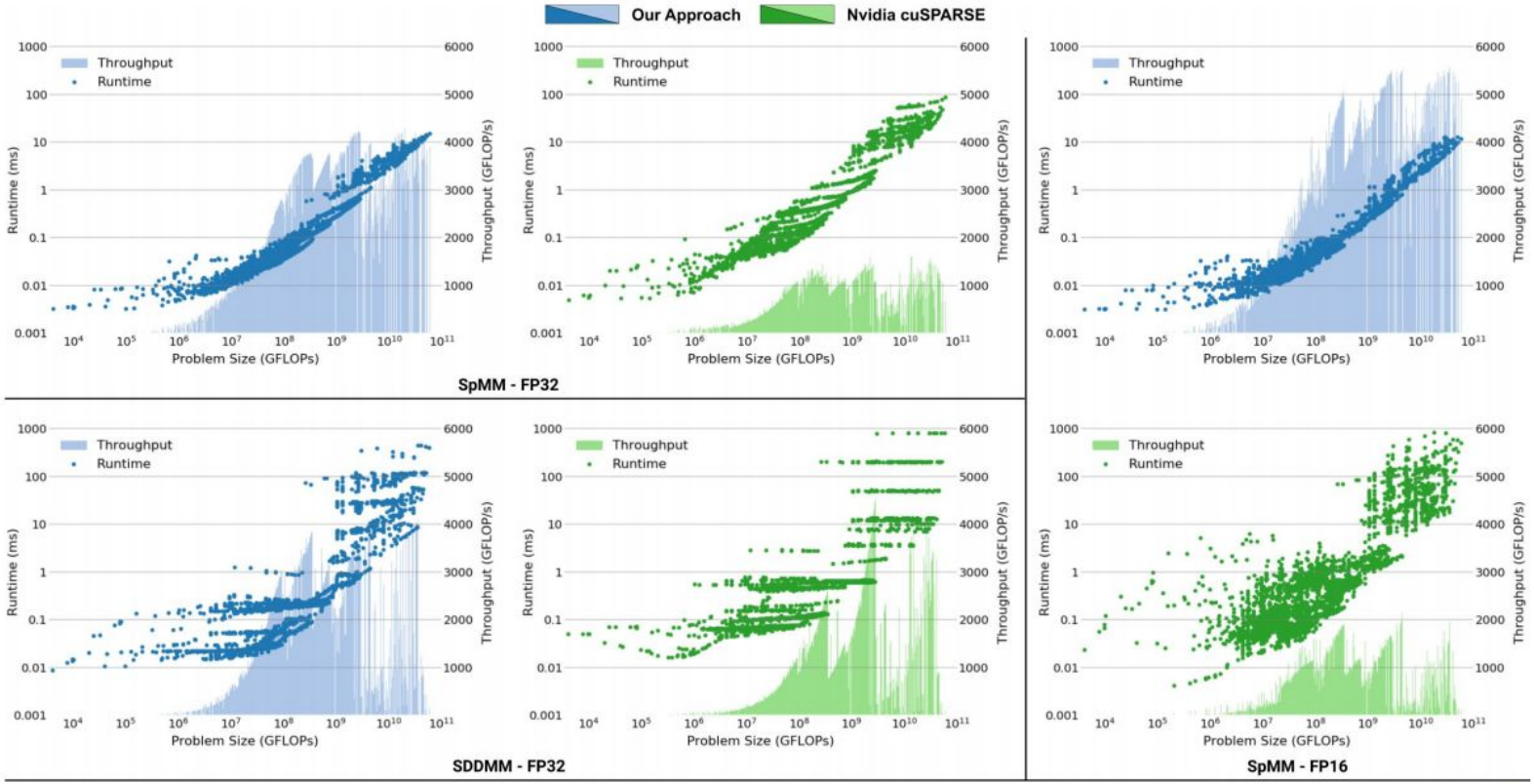


Fig. 9. **Benchmarks on our dataset of sparse matrices from deep neural networks.** Runtime (left y-axis) and throughput (right y-axis) plotted with increasing problem size for each kernel and precision. Benchmarked on an Nvidia V100 GPU. **Top Left:** SpMM benchmarks in single-precision. Across all problems, our approach achieves a geometric mean speedup of 3.58× and a peak speedup of 14.2× over Nvidia cuSPARSE. **Bottom Left:** SDDMM benchmarks in single-precision. Across all problems, our approach achieves a geometric mean speedup of 2.19× and a peak speedup of 6.58× over Nvidia cuSPARSE. **Right:** SpMM benchmarks in mixed precision with 16-bit data and 32-bit computation. Across all problems, our approach achieves a geometric mean speedup of 5.97× and a peak speedup of 297.5× over Nvidia cuSPARSE.

Questions?