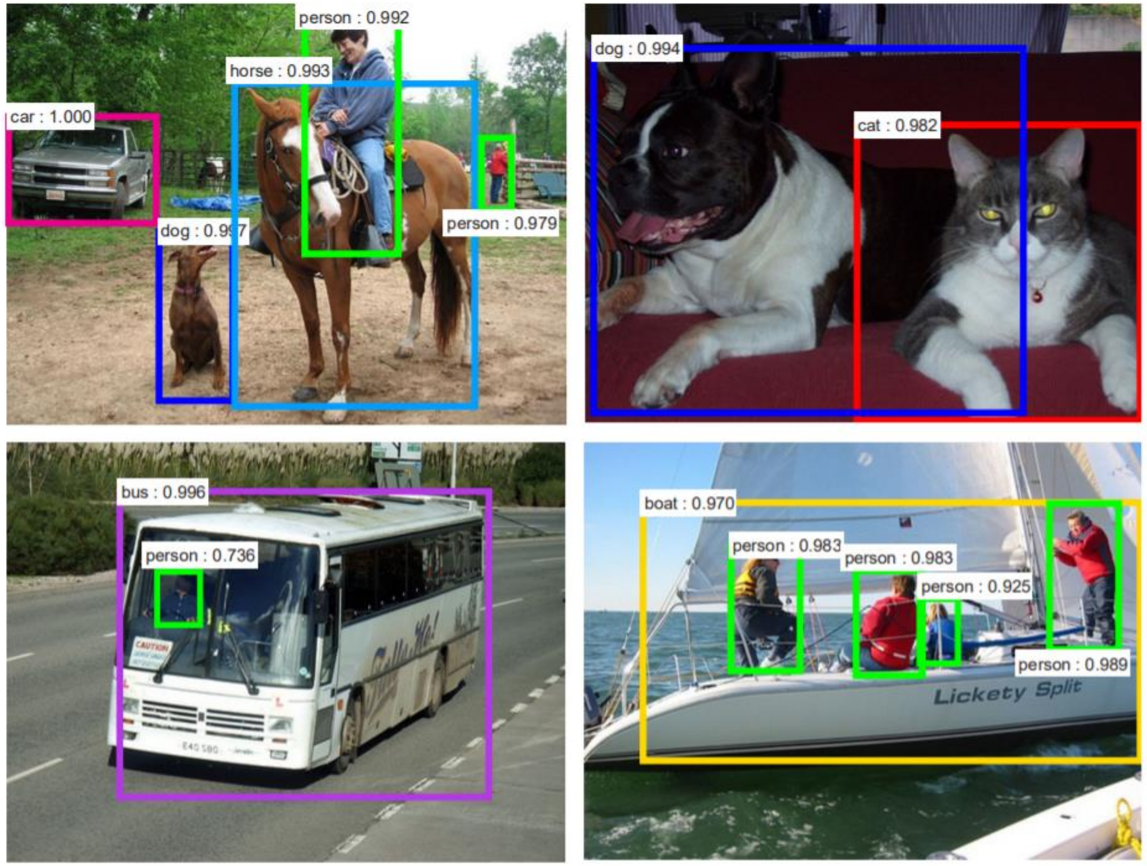
Deblur-YOLO: Real-Time Object Detection with Efficient Blind Motion Deblurring

Shen Zheng, YuXiong Wu, Shiyu Jiang, Changjie Lu and Gaurav Gupta

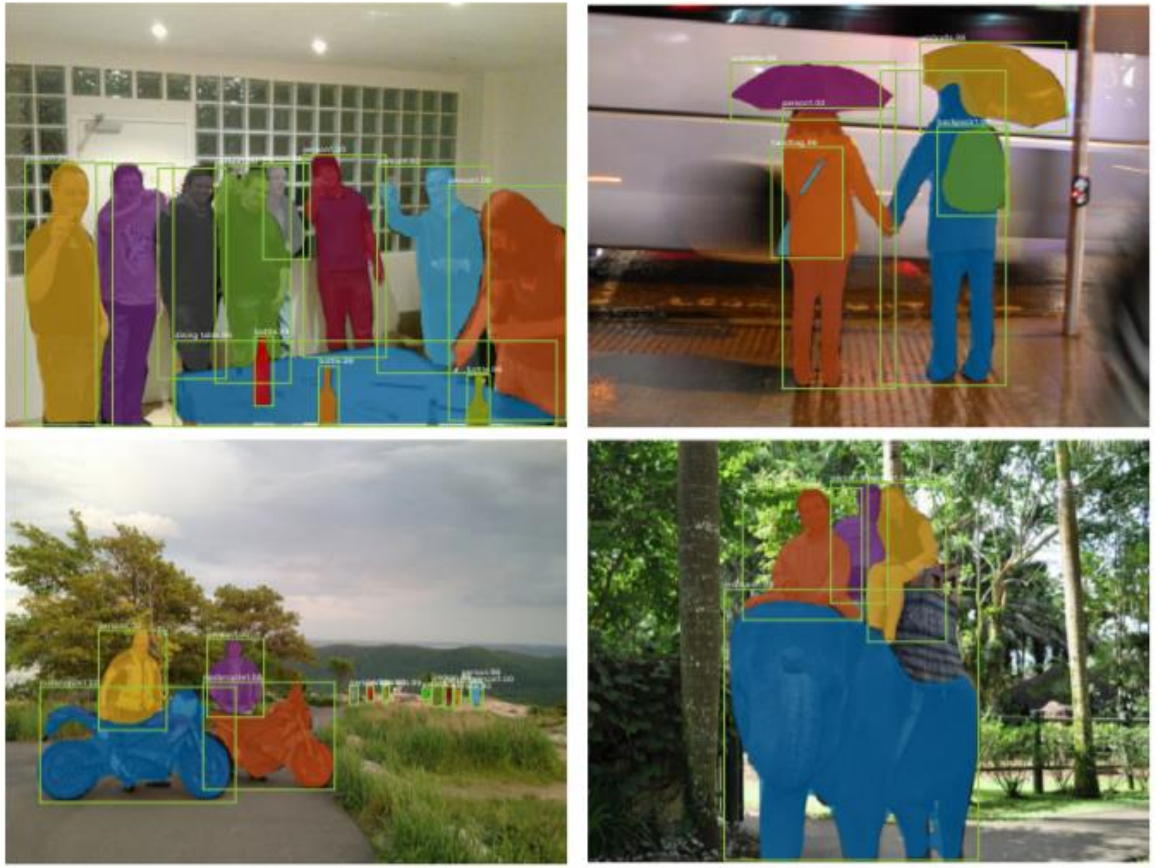
Wenzhou Kean University

Background

Object Detector are AWESOME!



Ren et.al. Faster R-CNN (2016)

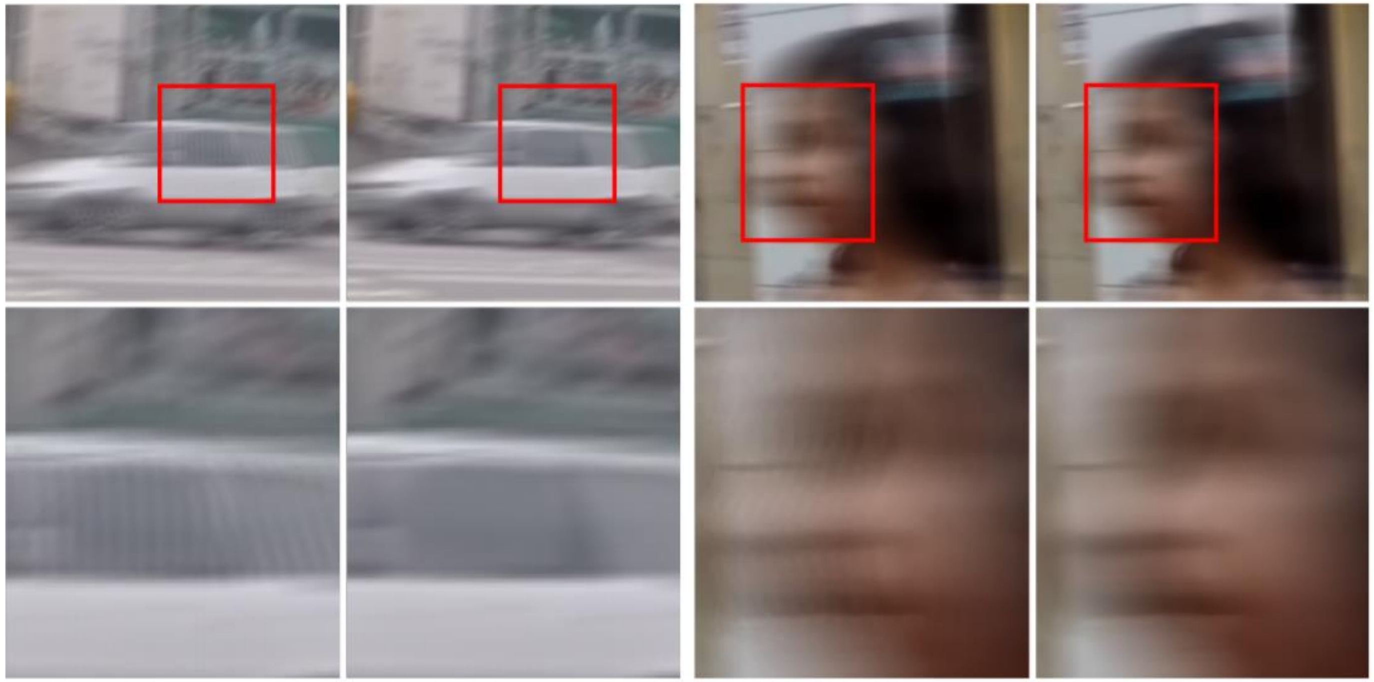


He et.al. Mask R-CNN (2018)

Problems

Real-World Situations ?

- Vehicle movement
- Camera Shake
- Poor Weather



Kupyn et.al. DeblurGANv2 (2019)

Problems

- “AWESOME” **ONLY** at Clean Images
- Suffer from Image Degradation

(a) Clean Image (b) Blurred Image (c) Deblur-YOLO

Fig. 1: **Sample Detection Result**. Deblur-YOLO makes blur robust object detecton at a densely populated image from COCO 2014. Left: Yolov3 at clean image. Middle: Yolov3 at Blurred Image. Right: Deblur-YOLO at Blurred Image

Existing Solutions

- Non-Blind Deblurring
  - Unnatural l0 sparse representation (Xu, 2013)
  - Edge-based kernel estimation + Patch priors (Sun, 2013)

- Blind Deblurring
  - Non-uniform motion blur kernel estimation (Sun, 2015)
  - Fourier coefficient of deconvolutional kernel (Chakrabarti, 2016)
  - DeepDeblur (Nah, 2017)
  - SRN-DeblurNet (Tao, 2018)
  - DeblurGANv1&v2 (Kupyn, 2018&2019)

Problems

- VERY **SLOW** => Unsuitable for real-world tasks

TABLE II: Deblurring Performance at COCO 2014

	Time	Params	PSNR	SPSNR	SSIM
Blur Image	None	None	21.02	116.51	0.701
DeepDeblur	1.5495	47.4	24.86	105.08	0.823
DynamicDeblur	1.5247	47.8	**27.19**	113.20	**0.873**
SRN	0.3790	86.9	24.61	99.92	0.815
DeblurGANv2(I-R)	0.1589	233.0	20.29	108.45	0.687
DeblurGANv2(M)	**0.0769**	**12.8**	20.34	124.94	0.687
Deblur-Yolo	0.0772	12.9	23.94	**131.39**	0.817

Our Solution

- Deblur-YOLO Work Flow

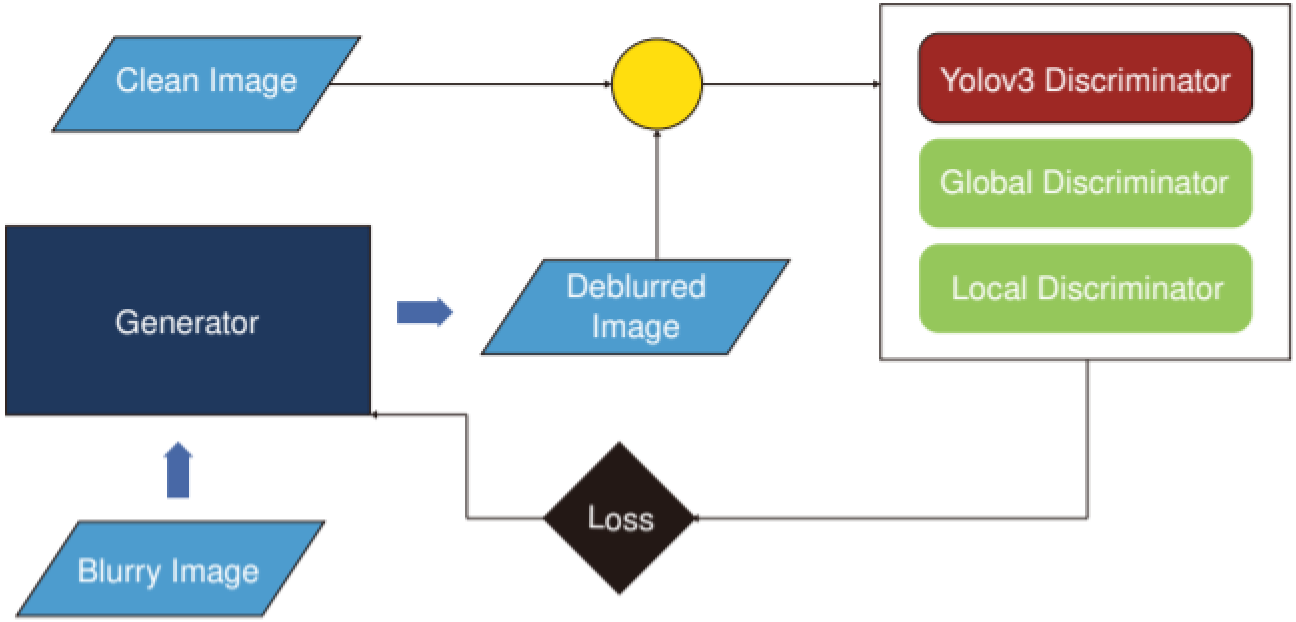


Fig. 2: **Model WorkFlow Design**. Deblur-YOLO is a Generative Adversarial Network (GAN) with one generator and a group of discriminators.

Our Solution

- Deblur-YOLO Generator Architecture

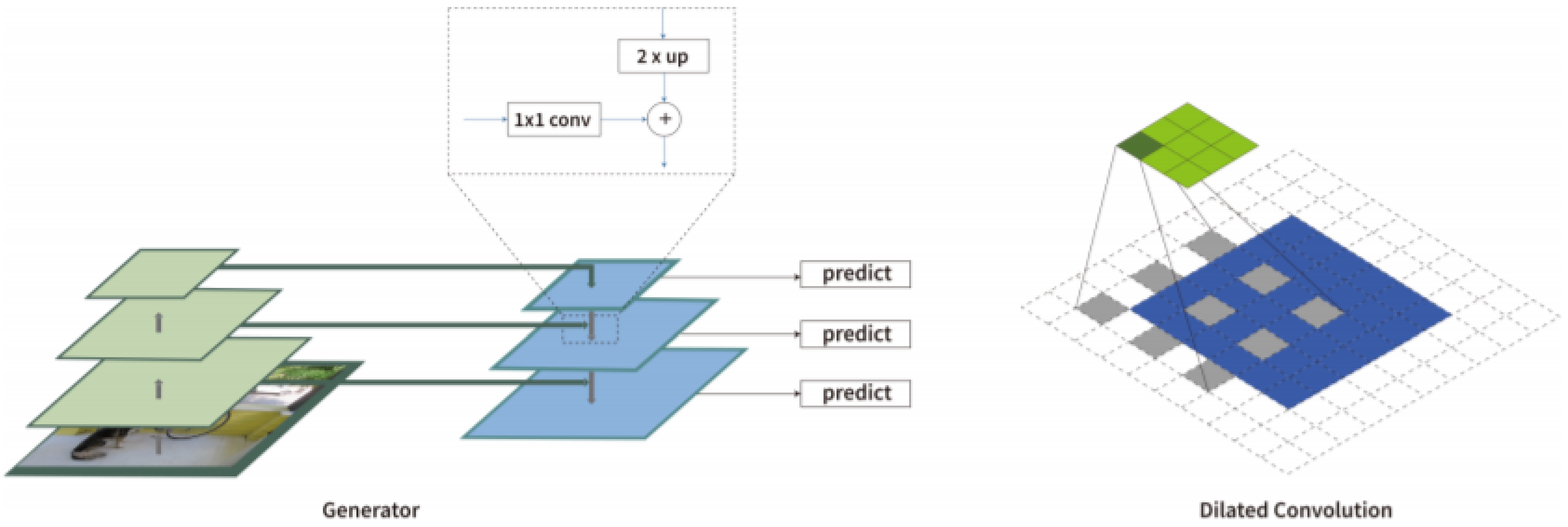


Fig. 3: **Generator Architecture**. Left: Generator building blocks with convolution and upsampling operations. Right: Dilated Convolution layers with stride 1, padding 2, dilation 2, kernel size 3 and filter number 128. We use blue, light green and dark green for dilated convolution blocks, vanilla convolution blocks and kernels, respectively. Each Convolution block consists of a convolution layer, a normalization layer and a ReLU [15] activation layer.

Our Solution

- Deblur-YOLO Discriminator Architecture

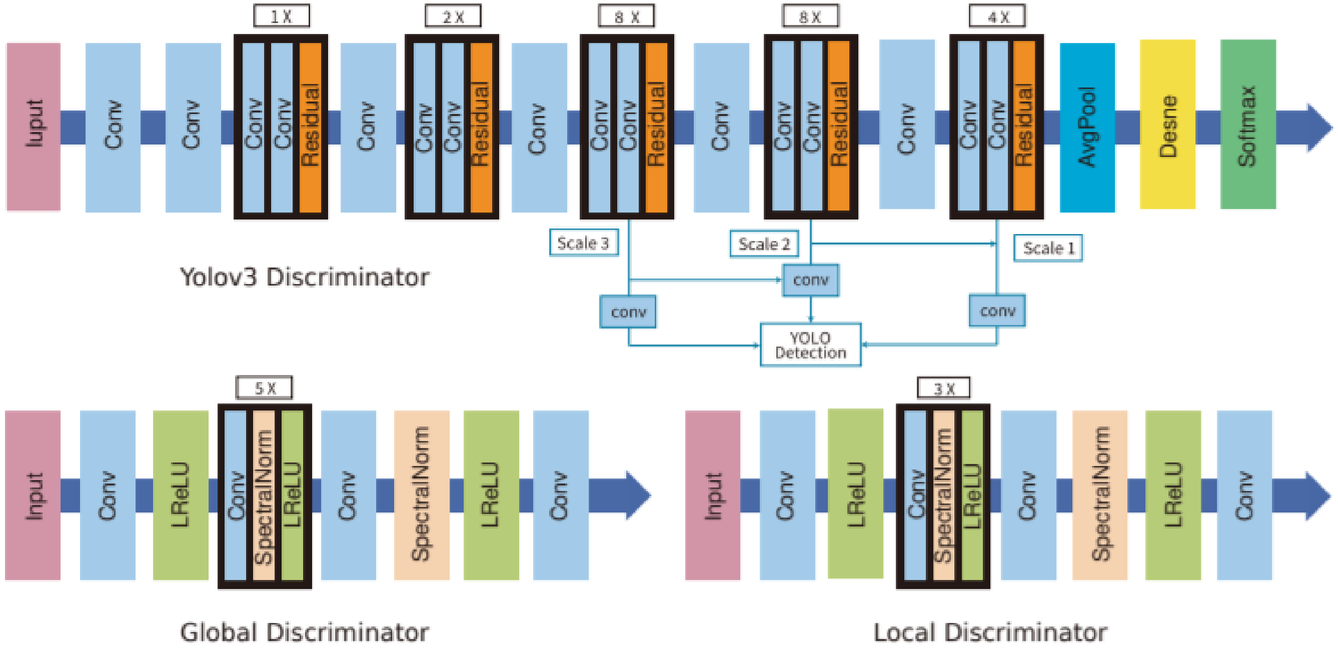


Fig. 4: **Discriminator Architecture**. Up: Detection Discriminator. Lower Left: Global-Scale Discriminator. Lower Right: Local-Scale Discriminator. For Yolov3 Discriminator, we use "Conv" for convolution blocks and "Residual" for residual blocks [29]. For Global and Local Discriminator, we use "Conv" for convolution layers and "LReLU" for Leaky ReLU [30] activation layers. Each convolution layers have stride 2, padding 2 and kernel size 4.

Our Solution

- Deblur-YOLO Loss Function

$$\mathrm{L_G} = 0.5 * \mathrm{L_C} + 0.006 * \mathrm{L_P} + 0.01 * \mathrm{L_{A_G}} + 0.1 * \mathrm{L_D}$$

$$\begin{aligned}\mathrm{L_{A_G}} = & \mathbb{E}_{z\sim p_z(z)}[(G(z) - \mathbb{E}_{x\sim p_{\text{data}}(x)}G(x) - 1)^2] \\ & + \mathbb{E}_{x\sim p_{\text{data}}(x)}[(G(x) - \mathbb{E}_{z\sim p_z(z)}G(z) + 1)^2]\end{aligned}$$

$$\begin{aligned}\mathrm{L_{A_D}} = & \mathbb{E}_{x\sim p_{\text{data}}(x)}[(D(x) - \mathbb{E}_{z\sim p_z(z)}D(G(z)) - 1)^2] \\ & + \mathbb{E}_{z\sim p_z(z)}[(D(G(z)) - \mathbb{E}_{x\sim p_{\text{data}}(x)}D(x) + 1)^2]\end{aligned}$$

Experiments

- Qualitative Result at Set5

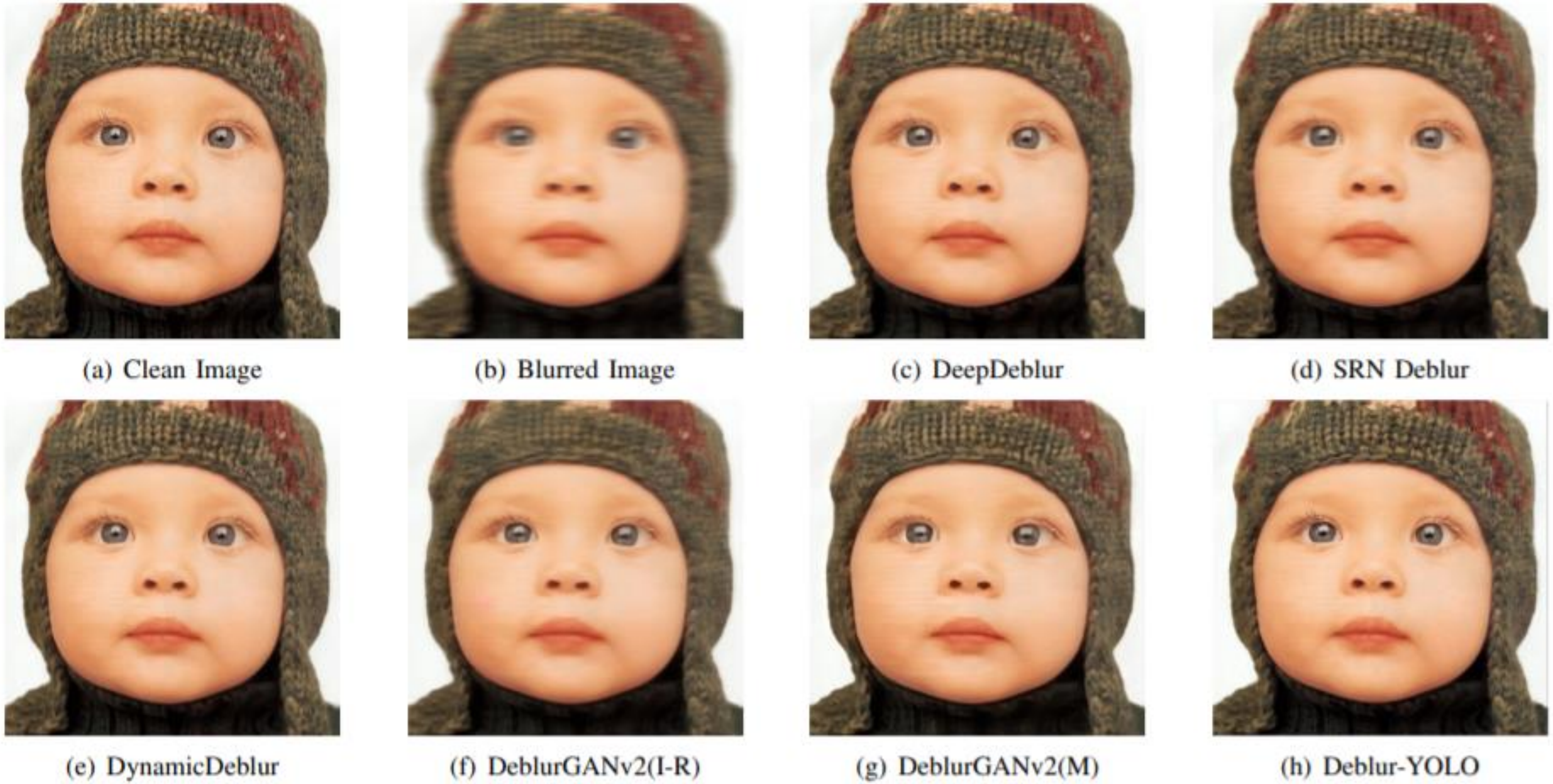


Fig. 5: **Deblurring Result Comparison at Baby Picture from Set 5**

Experiments

- Qualitative Result at Set14

Fig. 6: **Deblurring Result Comparison at Lenna Picture from Set14**

Experiments

- Qualitative Result at COCO 2014

(a) clean Image (b) Blurred Image (c) DeepDeblur (d) SRN Deblur (e) DynamicDeblur (f) DeblurGANv2(I-R) (g) DeblurGANv2(M) (h) Deblur-YOLO

Fig. 7: **Deblurring and Detection Result Comparison at COCO 2014**

Experiments

- Quantitative Results

TABLE I: mAP Score at COCO 2014

	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Clean Image	58.5	73.7	46.9	44.4	40.6	42.3	75.2	58.5	73.0	39.8	52.3	41.4	73.2	77.5	61.7	69.1	42.4	59.9	52.1	75.4	69.9
Blur Image	29.7	43.3	16.9	15.3	14.8	10.4	51.8	34.5	40.4	13.5	12.6	26.9	31.7	30.9	28.2	42.8	19.0	23.7	33.7	57.6	45.4
DeepDeblur	51.7	64.9	36.9	35.2	35.3	32.2	73.1	53.7	70.3	33.9	40.8	40.1	59.6	68.1	51.9	65.3	35.2	46.8	48.5	72.8	68.4
DynamicDeblur	**56.0**	**70.6**	**43.2**	**41.4**	**41.4**	**36.8**	**75.3**	**57.1**	**72.6**	**36.6**	**45.8**	40.0	**68.2**	**72.7**	**56.3**	**67.0**	**39.8**	**58.4**	49.9	**75.3**	**71.7**
SRN	52.3	70.1	38.2	35.8	35.8	31.8	71.9	53.4	69.2	33.1	39.4	39.8	63.4	66.6	53.5	64.1	35.1	51.7	48.0	**75.3**	69.0
DeblurGANv2(I-R)	42.0	55.0	28.6	26.6	30.2	24.9	61.4	44.9	53.5	27.5	35.4	32.4	47.4	53.7	39.6	51.8	24.8	41.2	39.2	65.2	55.9
DeblurGANv2(M)	40.8	52.2	27.4	25.0	28.9	24.3	61.0	44.3	53.7	25.9	31.7	30.5	45.2	49.4	39.2	50.8	25.0	38.6	40.6	66.0	56.8
Deblur-Yolo	47.5	55.5	33.8	30.0	37.7	29.7	67.7	51.1	62.6	31.2	39.5	**41.2**	51.4	54.7	44.9	56.1	33.6	53.9	**50.2**	72.8	52.2

TABLE III: Deblurring Performance at Set 5 & Set 14

		Blur Image	DeepDeblur	DynamicDeblur	SRN	DeblurGANv2(I-R)	DeblurGANv2(M)	Deblur-Yolo
Set 5	PSNR	24.20	28.36	29.10	28.07	26.64	27.06	**29.39**
	SPSNR	113.79	104.80	113.99	98.77	103.74	122.66	**128.40**
	SSIM	0.66	0.81	0.85	0.80	0.74	0.77	**0.88**
Set 14	PSNR	23.12	26.65	27.35	25.90	25.95	25.03	**27.85**
	SPSNR	119.26	111.00	115.09	111.58	116.26	**128.30**	121.70
	SSIM	0.55	0.69	0.73	0.67	0.68	0.65	**0.75**

Conclusion

- Deblur-YOLO
- Efficient, Detection-Driven, One-Stage
- Generator + Multi-Scale Discriminator + Detection Discriminator
- Blind motion deblurring + Object Detection
- Smooth Peak Signal-to-Noise Ratio (SPSNR)
- Promising Results on COCO2014, Set5 and Set14

Acknowledgement

- This work is supported by the research funding from Wenzhou-Kean University with project number SpF2021011.
- We are grateful for **Dr. Kennedy E. Ehimwenma** for his helpful comments on model annotation and result interpretation. We also thank **Yayun Chen** for her help on model design.