Improving Feature Stability during Upsampling – Spectral Artifacts and the Importance of Spatial Context

Supplementary Material

In the following, we present results and figures to support our statements in the main paper and provide additional information. The following has been covered in the appendix:

– Appendix A: **Detailed experimental setup** for all downstream tasks.

- Appendix A.1: Image Restoration experimental setup
- Appendix A.2: Semantic Segmentation experimental setup
- Appendix A.3: Disparity Estimation experimental setup
- Appendix A.4: Detailed setup of Adversarial attacks for all downstream tasks.
- Appendix A.5: Detailed setup of adversarial training for semantic segmentation and image restoration.

– Appendix B: **Semantic Segmentation**: Additional Experiments and Ablations. In detail:

- Appendix B.1: Detailed results from Sec. 5.2 and Sec. 5.2.
- Appendix B.1: Discussion on saturation of kernel size for upsampling.
- Appendix B.2: An ablation on the impact of the capacity of the encoder block for standard options such as ResNet or ConvNeXt blocks.
- Appendix B.3: Ablation about including or excluding a small parallel kernel during upsampling using transposed convolution.
- Appendix B.4: Short study on drawbacks of using interpolation for pixel-wise upsampling.
- Appendix B.5: A comparison to different kinds of upsampling Operations on Segmentation Models.
- Appendix B.6: A comparison of the performance of different sized kernels in the transposed convolution operations of UNet-like models adversarially trained using FGSM attack and 3-step PGD attack on 50% of the mini-batches during training.

- Appendix C: **Image Restoration** : Additional Results:
  - Appendix C.1: Here we report the number of parameters and latency study of LCTC.
  - Appendix C.2: Adversarial training evaluation for Restormer and NAFNet for Image deblurring task.
  - Appendix C.3: Qualitative results for image reconstruction models using Restormer and NAFNet and evaluated on clean data, PDG and CosPGD attack with varying numbers of attack iterations.
  - Appendix C.4: **Visualizing Kernel Weights**: Here we visualize kernel weights from a random channel for models from Figure 5 to show the how different kernels handle uneven contributions of pixels that leads to spectral artifacts.
  - Appendix C.5: Out-Of-Distribution and Real World Generalization.
- Appendix D: **Disparity Estimation** : We provide additional results for Section 5.3: including performance against adversarial attacks.
  - Appendix D.1 Additional discussion on the results and importance of a parallel 3×3 kernel with large kernels for transposed convolution operation.
- Appendix E: **Nomenclature- What are "Large Context Transposed Convolutions?"**: We discuss the nomenclature used in this work and describe what comprises a LCTC.
- Appendix F: **Additional visualizations of Upsampling Artifacts and their Frequency Spectra**: Here we extend Figure 1 with more examples showing failure of upsampling operations used in prior work and superiority of LCTC both in the spatial and frequency domain.
- Appendix G: **Limitations**: Here we discuss the limitations of our work in detail.

A Experimental Setup

All the experiments were done using NVIDIA V100 16GB GPUs or NVIDIA Tesla A100 40GB GPUs. For image restoration, models were trained on 1 NVIDIA Tesla A100 40GB GPU. For the semantic segmentation downstream task, UNet [70] was trained using 1 GPU. For the disparity estimation task, STTR-light [50] was trained using 4 NVIDIA V100 GPUs in parallel.

A.1 Image Restoration

Architectures. We consider the recently proposed state-of-the-art transformer-based Image Restoration architectures Restormer [89] and NAFNet [16]. Both architectures as proposed use Pixel Shuffle [77] to upsample feature maps. We use these as our baseline models. We replace this pixel shuffle operation with a transposed convolution operation.
Dataset. For the Image Restoration task, we focus on Image Deblurring. For this, we use the GoPro image deblurring dataset [63]. This dataset consists of 3214 real-world images with realistic blur and their corresponding ground truth (deblurred images) captured using a high-speed camera. The dataset is split into 2103 training images and 1111 test images.
Training Regime. For Restormer we follow the same training regime of progressive training as that used by [89]. Similarly, for NAFNet we use the same training regime as that used by [16].
Evaluation Metrics. Following common practice [1, 16, 89], We report the PSNR and SSIM scores of the reconstructed images w.r.t. to the ground truth images, averaged over all images. PSNR stands for Peak Signal-to-Noise ratio, a higher PSNR indicates a better quality image or an image closer to the image to which it is being compared. SSIM stands for Structural similarity [86]. A higher SSIM score corresponds to better higher similarity between the reconstruction and the ground-truth image.

A.2 Semantic Segmentation

Here we describe the experimental setup for the segmentation task, the architectures considered, the dataset considered and the training regime.
Architectures. We considered UNet [70] with encoder layers from ConvNeXt [54]. For the decoder, the baseline comparison is done with 2×2 kernels in the transposed convolution layers and the commonly used ResNet [37] BasicBlock style layers for the convolution layers in the decoder building blocks. In our experiments, we used larger sized kernels, e.g. 7×7 and 11×11 in the transposed convolution while keeping the rest of the architecture, including the convolution blocks in the decoder identical to Sec. 5.2. When using kernels larger than 7×7 for transposed convolution we follow the work of [17,51] and additionally include a parallel 3×3 kernel to keep the local context. Usage of this parallel kernel is denoted by "+3×3" Further, we analyze the behavior of a different block of convolution layers in the decoder, as explained in Sec. 4 and replace the ResNet-style layers with ConvNeXt-style layers in Sec. 5.2.
Dataset. We considered the PASCAL VOC 2012 dataset [24] for the semantic segmentation task. We follow the implementation of [92–94] and augment the training examples with semantic contours from [36] as instructed by [75].
Training Regime. We follow a similar training regime as [92, 93], and train for 50 epochs, with an AdamW optimizer [57] and the learning rate was scheduled using Cosine-Annealing [56]. In the implementation of [93], the authors slide over the images using a window of size 473×473, however for computation reasons and for symmetry we use a window of size 256×256. We use a starting learning rate of 10^{-4} and a weight decay of 5×10^{-2}.

Evaluation Metrics. We report the mean Intersection over Union (mIoU) of the predicted and the ground truth segmentation mask, the mean accuracy over all pixels (mAcc) and the mean accuracy over all classes (allAcc).

A.3 Disparity Estimation

Following, we describe the experimental setup for disparity estimation and occlusion detection tasks.
Architectures. We consider the STTR-light [50] architecture for our work. To analyze the influence of implementing larger kernels in transposed convolution as described in Section 4 we alter the kernel sizes in the transposed convolution layers used for pixel-wise upsampling in the "feature extractor" module of the architecture. We consider the STTR-light architecture as proposed by [50] with 3×3 kernels in the transposed convolution layers as our baseline.
Dataset. Similar to [50] we train and test our models on FlyingThings3D dataset [60].
Training Regime. We follow the training regime as implemented in [50].
Evaluation Metrics. We report the end-point-error (epe) and the 3-pixel error (3px) for the disparity estimation w.r.t. the ground truth.

A.4 Adversarial Attacks

We consider the commonly used [34,59,67,88] FGSM attack [28] and a new segmentation-specific SegPGD attack [34] for testing the robustness of the models against adversarial attacks. For the **semantic segmentation downstream task**, each crop of the input was perturbed with FGSM and SegPGD, while for the disparity estimation downstream task, each of the left and right inputs were perturbed using FGSM.
For FGSM, we test our model against epsilons $\epsilon \in \{\frac{1}{255}, \frac{8}{255}\}$. Where, we follow common practice and use $\frac{1}{255}$ $\approx$0.004 and $\frac{8}{255}$ $\approx$0.03 .

For SegPGD we follow the testing parameters as originally proposed in [34], with $\epsilon \approx \frac{8}{255}$, α=0.01 and number of iterations $\in \{3, 5, 10, 20, 40, 100\}$. We use the same scheduling for loss balancing term λ as suggested by the authors. We use SegPGD for the semantic segmentation task as it is a stronger attack specifically designed for segmentation. Thus providing more accurate insights into the models' performance and giving a better evaluation of the architectural design choices made.

For the **Image Restoration task**, we follow the evaluation method of [1], and evaluate against CosPGD [3] and PGD [49] adversarial attacks. For both attacks, we use $\epsilon \approx \frac{8}{255}$, α=0.01 and test for number of attack iterations $\in \{5, 10, 20\}$.

For the **Depth Estimation task**, we use the PGD attack with $\epsilon \approx \frac{8}{255}$, α=0.01 and test for number of attack iterations $\in \{5, 10, 20\}$.

A.5 Adversarial Training

Following, we describe the adversarial training setup employed in this work for adversarially training models for semantic segmentation and image restoration.

Semantic Segmentation. We follow the commonly used [34] procedure and split the batch into two 50%-50% mini-batches. One mini-batch is used to generate adversarial examples using FGSM attack with $\epsilon \approx \frac{8}{255}$ and PGD attack with 3 attack iterations and with $\epsilon \approx \frac{8}{255}$ and α=0.01 during training.

Image Restoration. We follow the training procedure used by [1]. We split each training batch into two equal 50%-50% mini-batches. We use one of the mini-batches to generate adversarial samples using FGSM attack with $\epsilon \approx \frac{8}{255}$.

A.6 Frequency spectrum analysis

To analyze the images in the frequency domain, we use the Fast Fourier Transform [9] (FFT) $X_c = FFT(x_c)$ for all channels c of feature maps x and aggregate a 2D representation over frequencies w. We compute the mean over C channels of the FFT of the difference between the prediction and the ground truth.

$$\text{2D Frequency Spectra} = \frac{1}{C}\sum_{c \in C} FFT(x_c^{pred} - x_c^{gt}) \tag{4}$$

Here, x^{pred} are the predictions from the model, x^{gt} is the ground truth, and in Fig. 1 and Fig. 14 C=3 for the RGB channels. For better visualization, we plot the $\log$ of the magnitude of the Discrete Fourier Transform.

Next, we describe, from the literature, the process of performing a Discrete Fourier Transform.

Fast Fourier Transform (FFT) [9]. The discrete Fourier transform has been used in this work to convert the images from the spatial domain to the frequency domain.

> " DFT is a linear operator (i.e. a matrix) that maps the data points in f to the frequency domain $\hat{f}$ " [10]

Equation 2.26 in [10] shows the formula to perform DFT is:

$$\hat{f}_k = \sum_{j=0}^{n-1} f_j \epsilon^{-i2\pi jk/n} \tag{5}$$

where $\hat{f}_k$ from each sample n contains the amplitude and phase (of the sine and cosine components) information at frequency k. These are integer multiples of $\epsilon^{-2\pi j/n}$, the fundamental frequency, short-handed as ω_n [10]. Equation 2.29 in [10] shows the Discrete Fourier transform matrix (in terms of ω_n) that when multiplied by the samples in f, converts the information in those samples to frequency domain (a basis transformation). FFT is an algorithm by [9] to perform Discrete Fourier transform in an efficient manner. In Eq. (4), we use these frequencies w (referred to as k in Eq. (5)) from sample x_c obtained using an FFT( ) function that uses the FFT algorithm.

B Additional Experiments and Ablation

Here we provide detailed results from Sec. 5 and Sec. 5.2 and additional results as mentioned in the main paper.

B.1 Semantic Segmentation

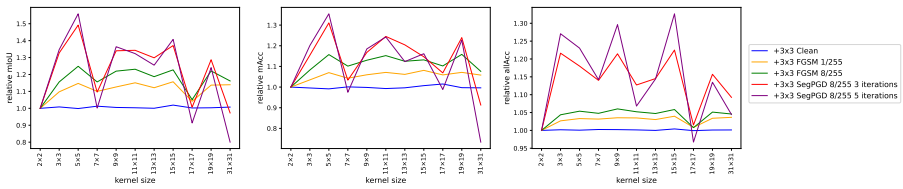


Fig. 8: Comparison of performance of different sizes of transpose convolutions from standard sizes like 2×2 as well as very large 31×31 kernels with ConvNeXt style 11×11 + 3×3 style in the decoder building blocks. All have a parallel 3×3 kernel, as shown in Figure 4 (bottom left).

Table 7 and Table 8 provide all the results of empirical performance (across the considered upsampling blocks) on clean inputs images and input images perturbed by varying intensities of FGSM and SegPGD attacks respectively.

Limit of large kernels for Upsampling As discussed in Sec. 5.2, the performance of large kernels begins to saturate at a point. We report results from Figure 7 in tabular form in Table 9. In Table 9, we find that 13×13 appears to be the saturation point for this setting and 31×31 kernels are beyond this saturation point. While 31×31 performs worse or on-par with 17×17, it still performs significantly better than the baseline of 2×2. In Section 5.2 we explain the kernel size limit and that larger kernels are difficult to train. **We also find that these results further strengthen our Hypothesis 2.** For ease of understanding, we visualize the trends from Table 9 in Figure 8.

B.2 Choice of encoder

Following we aim to understand the importance of the encoder and its influence on the quality of representations later decoded during the upsampling. Consequently, we justify our choice of using ConvNeXt tiny encoder for the majority of our studies.

In Table 11 we compare different encoders: ResNet50, ConvNeXt tiny, and SLaK [51] while fixing the decoder to the baseline implementation. All encoders are pre-trained on the ImageNet-1k training dataset.

We observe that using ConvNeXt tiny and SLaK as the encoder backbone gives us significantly better performance than using ResNet50 as the encoder. This observation holds true for both clean and adversarially perturbed samples. We additionally observe that SLaK gives us marginally better performance than ConvNeXt. As shown by [51], SLaK is a significantly better encoder than ConvNeXt tiny as it provides significantly more context than ConvNeXt by using kernel sizes up to 51×51 in the convolution layers during encoding. This proves that better encoding can be harnessed during decoding which can lead to better upsampling.

However, in this work, we used the ConvNeXt tiny encoder since the SLaK encoder takes significantly longer to train for only a marginal gain in performance. We report

the performance results in Table 12. We observe that given our computation budget and the wall-clock time limit of 24 hours, we are unable to even compute the performance of the model with the SLaK encoder at 100 attack iterations.

B.3 Ablation over small parallel kernel

Following we ablate over the use of a small (3×3) kernel in parallel to a large ($\geq7\times7$) kernel for Large Context Transposed Convolutions. This concept is inspired by [17,51] who use a small kernel in parallel with the large kernels to preserve local context when downsampling. Similar behavior is observed while upsampling. Table 7 compares the usage of this small parallel kernel. We observe, that while not using the small kernel results in marginal better performance on clean images (for a fixed backbone style), it lacks context and thus performs poorly (when compared to using a small parallel kernel) against adversarial attacks.

This is further highlighted inTab. 8 when the performance is compared against strong adversarial attacks. Moreover, we observe that from medium-sized kernels i.e. , the upsampling seems to lose local context, and adding a kernel in parallel helps the model in getting this additional context. This effect can also be observed in the adversarial performances of the respective models.

B.4 Drawbacks of interpolation

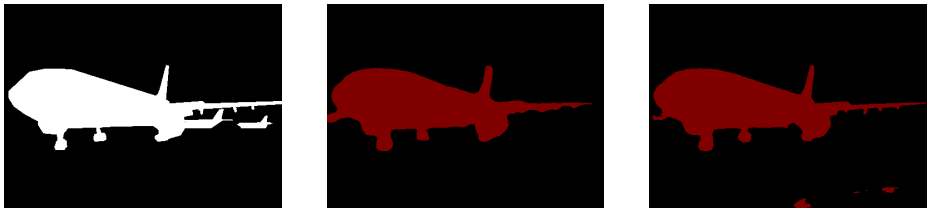
As discussed in Section 3, architecture designs that use interpolation for pixel-wise upsampling suffer with over-smoothening of feature maps. This can be seen in the final predictions, as shown in Fig. 9b compared to the ground truth segmentation mask in Fig. 9a and prediction from a model with 11×11 + 3×3 transposed convolution kernel in Fig. 9c.

In their work, [34] showed that PSPNet has considerably lower performance against adversarial attacks, similar to the analysis made in Section 5.2. This is explained by H2.

B.5 Different Upsampling Methods

Following we compare different upsampling techniques thus justifying our advocacy for using Transposed Convolution instead of other upsampling techniques like interpolation and pixel shuffle.

We report the comparison in Table 5 and observe that both Pixel shuffle and Nearest Neighbor interpolation perform better than the usually used Transposed Convolution with a 2×2 kernel size. However, as we increase the kernel size for Transposed Convolution to 11×11 with a 3×3 small kernel in parallel, we observe that Large Context Transposed Convolutions are strictly outperforming pixel shuffle, on both clean unperturbed images and under adversarial attacks, across all metrics used. Transposed Convolution with a large kernel is either outperforming or performing at par with Nearest Neighbor interpolation as well. Thus we demonstrate the superior clean and adversarial performance of large kernel-sized Transposed Convolution operation over other commonly used upsampling techniques.



(a) Ground truth segmentation mask of the third image in the test set.

(b) Prediction from PSPNet with ResNet 50 backbone as implemented by the authors.

(c) Prediction when using LCTC (11×11 + 3×3) and 3×3 convolution kernels in the decoder building blocks of UNet.

Fig. 9: A comparison of differences in the sharpness of final predictions due to different upsampling techniques. Fig. 9a is the ground truth segmentation mask with sharp and thin edges in the rear fin and wing with protrusions in the wing of the aircraft. We observe that PSPNet with a ResNet50 backbone as implemented by [93] is not able to accurately predict the thin edges and the protrusions, and is simply smoothening them out. This is due to the interpolation operation used in upsampling. However in comparison, as shown in Fig. 9c, when a transposed convolution operation is used for pixel-wise upsampling, the thin edges are sharper and protrusions are more accurately predicted.

There might be speculation if other downsampling techniques can utilize larger convolution kernels in the decoder building blocks better than transposed convolution. Thus, we additionally experiment using a ConvNeXt-like 7×7+3×3 kernel in the Convolution operations in the decoder building blocks that follow the upsampling operation. We report these results in Table 13 and observe that similar to transposed convolution, other upsampling methods also do not benefit from an increase in the kernel size in the decoder building blocks.

B.6 Adversarial Training

Following, we present the results from adversarial training for semantic segmentation. In Table 10, we report the performance of different transposed convolution kernel-sized adversarially trained UNet on clean input and adversarially perturbed inputs. The observed performance improvement when increasing the transposed convolution kernel size during normal training also extends to adversarial training.

C Additional Results on Image Restoration

Following we provide additional results for the Image deblurring tasks, like the performance of models after adversarial training and some visual results of the deblurring for a better understanding of the impact of increased spatial context against different adversarial attack methods and strengths.

C.1 Latency Study

As PixelShuffle, when downsampling with a factor of 2, reduces the channel dims by a factor of 4, works [16, 89] use a 1×1 convolution layer before the PixelShuffle to increase the number of channels by a factor of 4. This added complexity is not needed for Transposed Convolution. Thus, in Table 14 we report the number of parameters in the models from Figure 5 and report latencies (mean over 1000 runs) of the upsampling operations, and show that these are comparable. In practice, these differences are negligible as other unchanged operations are more costly.

C.2 Adversarial Training

In Table 15 we provide additional results for adversarially training image restoration network NAFNet using FGSM attack on 50% of the training minibatch of the GoPro dataset each iteration. The state-of-the-art Image Restoration models are significantly larger w.r.t. the number of parameters, compared to the models considered for semantic segmentation. Thus, they are significantly more difficult to train adversarially. They require more training iterations. Due to the limited computing budget, we have only trained them for the same iterations as clean (non-adversarial) training iterations. We already observe the advantages of using a larger kernel for transposed convolution over pixel-shuffle in these experiments.

C.3 Visual Results

Figure 10 shows reconstruction under PGD attack for Restormer [89] and NAFNet [16].
Figure 11 shows reconstruction under CosPGD attack for Restormer [89] and NAFNet [16].

C.4 Visualizing Kernel Weights

An increase in kernel size leads to an increase in context and since the context is increased, the effect of uneven contributions of pixels is negated leading to reduced spectral artifacts. This can be seen in Figure 12. Here we observe that the weights for 3×3 are high at the edges, causing the described grid effect, whereas for 11×11 kernels there is a smooth fading towards the border of kernels, negating this effect.

C.5 Real World and Out-Of-Distribution (ODD) Generalization

Since LCTC leads to improved sampling that provides stability to feature maps learned by the network (not merely defense), inspired by observations from [29], we hypothesize that the trends on adversarial attacks should translate to Real-World noise. We show this in Table 16 by applying 2D common corruptions (CC) (severity=3) from [39] on images from the GoPro dataset and using NAFNet models from Figure 5. Since the task is deblurring, we consider all common corruptions but additional blurring and weather corruptions, as these would have to be captured before blurring.

D Additional Results Disparity Estimation

Following we report additional results for Disparity Estimation using STTR-light. In Table 17 we report the performance of STTR-light architecture on clean test images and under PGD attack. Whereas in Figure 13, we present a visual comparison of depth estimation predictions by a vanilla STTR-light as proposed by [50] and our proposed modification of increasing the kernel size of the transposed convolution operation in the "feature extractor" module of the architecture from 3×3 to Large Context Transposed Convolutions with kernel sizes 7×7+3×3 and 11×11+3×3.

D.1 Disparity Estimation Discussion

In Figure 13 as shown by the region in the red circle, both vanilla architecture and the architecture with our proposed change perform well compared to the ground truth on clean images. However, under a 10 iteration PGD adversarial attack, we observe small protrusion's depth(shown by the red arrow) is incorrectly estimated by the vanilla architecture. The architecture with 7×7+3×3 and 11×11+3×3 transposed convolution kernels preserves the prediction of the disparity.

Additionally from Table 17, we observe the significance of the parallel 3×3 small kernel with the large 7×7 and 11×11 kernels. The stability of the performance of the large kernels without the small parallel kernel compared to the baseline is better. However, the stability of performance when only using larger kernels compared to larger kernels with small parallel kernels is marginally worse.

E Nomenclature: What are *Large Context Transposed Convolutions*?

In Section 4 we introduce the term **"Large Context Transposed Convolutions (LCTC)"**. In this work, we use this to describe the Transposed Convolution layers in the decoder with large kernel sizes and thus a large spatial context. However, terms like "large" are subjective, this in the following we discuss our interpretation of a "large" kernel size.

Most previous works use kernel sizes of 2×2 or 3×3 for any convolution operation, be it for downsampling [37,52] or be it for upsampling [70]. [54] introduced performing downsampling using convolution operations with a large kernel size which in their case was 7×7. This "larger" kernel size for downsampling was further extended by other works like [17, 35] to 31×31 and even up to 51×51.

In Section 3, we show how increasing context during upsampling can reduce spectral artifacts from a theoretical perspective. Theoretically, we would want an infinite-sized kernel when performing upsampling. However, this is not practical, thus we used Transposed Convolution with kernel sizes sufficiently large to give a good trade-off between theorized context and practical trainability and compute requirements.

Thus, inspired by encoding literature [17,35,54] we use kernel sizes for upsampling that are larger than those used by previous works. Given that previous works used kernel sizes like 2×2 or 3×3, anything bigger than this already provides more spatial context.

Thus, even a kernel size of 5×5 would be an interesting exploration and thus we explore this as well in Tab. 7 and Tab. 8.

However, given the theoretically ideal kernel size is infinity, a kernel size of 5×5 does not provide enough spatial context and thus we start calling transposed convolution operations as Large Context Transposed Convolution only when their kernel sizes are 7×7 or larger.

F Additional visualizations of Upsampling Artifacts and their Frequency Spectra

Following, we extend the example from Figure 1 to Figure 14 showing similar upsampling artifacts but on different input images to demonstrate that our findings are not limited to one example.

G Limitations

Current metrics for measuring performance do not completely account for spectral artifacts. Spectral artifacts begin affecting these metrics only when they become pronounced such as under adversarial attacks, and here Large Context Transposed Convolutions consistently perform better across tasks and architectures. Ideally, we would want infinitely large kernels, however, with increasing kernel size and task complexity, training extremely large kernels can be challenging. Thus, in this work, while having ablated over kernels as large as 31×31, we propose using kernels only as large as 7×7 to 11×11 for good practical trade-offs. Further improvements might be possible when jointly optimizing the encoder *and* decoder of architectures.

In this work, we are focused on the reduction of spectral artifacts in upsampled images and features introduced due to the theoretical limitations of upsampling operations. However, there might exist other factors that contribute to the introduction and existence of spectral artifacts such as spatial bias. This might also present an interesting avenue to explore.

Table 7: Complete comparison of performances against FGSM attack, of UNet with ConvNeXt encoder and decoder with architectures along with different sized kernels in transposed convolution and different convolution blocks in the decoder for upscaling the feature maps.

Transposed Convolution Kernels	Backbone Style	Test Accuracy			FGSM attack epsilon					
					$\epsilon=\frac{1}{255}$			$\epsilon=\frac{8}{255}$		
		mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc
2×2	ResNet Style 3×3	78.34	86.89	95.15	53.54	70.96	86.08	47.02	65.41	82.78
	ConvNeXt style 7×7	77.17	86.86	94.81	77.42	86.24	94.94	42.04	64.86	79.08
	ConvNeXt style 7×7 + 3×3	77.24	86.03	94.84	51.09	70.53	85.29	43.52	63.74	81.18
	ConvNeXt style 11×11	77.68	86.42	94.97	50.73	69.78	84.88	42.33	61.80	80.36
	ConvNeXt style 11×11 + 3×3	77.17	86.86	94.81	47.34	67.72	83.34	37.91	57.79	78.21
3×3	ResNet Style 3×3	78.45	86.66	95.20	53.76	70.62	86.32	47.33	64.58	83.16
	ConvNeXt style 7×7	77.70	86.89	94.99	52.30	71.56	85.73	44.80	65.38	81.99
	ConvNeXt style 7×7 + 3×3	77.33	87.53	94.79	50.90	72.77	83.78	44.40	67.08	79.11
	ConvNeXt style 11×11	77.86	86.75	94.99	51.30	70.39	85.33	42.78	62.76	81.08
	ConvNeXt style 11×11 + 3×3	77.81	86.48	94.98	51.95	70.08	85.57	43.82	62.56	81.63
5×5 (Ours)	ResNet Style 3×3	79.19	87.62	95.36	55.57	73.51	86.65	48.96	67.97	83.41
	ConvNeXt style 7×7	76.94	86.92	94.75	51.32	72.37	84.96	44.19	66.56	81.13
	ConvNeXt style 7×7 + 3×3	78.52	87.39	95.13	54.4	72.48	86.29	46.33	65.65	82.0
	ConvNeXt style 11×11	77.83	86.99	94.91	53.76	72.8	85.96	45.32	65.82	81.82
	ConvNeXt style 11×11 + 3×3	77.92	86.92	95.02	48.67	68.11	83.96	38.88	58.13	78.96
5×5 + 3×3 (Ours)	ResNet Style 3×3	78.83	87.56	95.28	56.11	73.97	86.91	49.84	69.26	83.44
	ConvNeXt style 7×7	78.11	86.90	95.01	53.17	71.55	86.0	45.98	66.05	82.18
	ConvNeXt style 7×7 + 3×3	78.73	87.81	95.24	53.86	73.12	85.86	45.93	66.83	81.51
	ConvNeXt style 11×11	77.83	86.57	95.07	52.12	70.29	85.79	44.05	63.11	81.63
	ConvNeXt style 11×11 + 3×3	77.07	86.11	94.87	54.31	72.45	86.1	47.33	66.88	82.42
LCTC: 7×7 (Ours)	ResNet Style 3×3	78.92	88.06	95.23	56.02	74.13	86.45	49.24	68.89	82.87
	ConvNeXt style 7×7	77.57	87.04	94.92	52.93	72.18	85.51	44.89	65.71	80.74
	ConvNeXt style 7×7 + 3×3	77.88	87.0	95.05	51.63	70.74	85.37	43.15	62.74	80.83
	ConvNeXt style 11×11	77.9	87.35	94.94	53.47	72.61	85.79	45.49	67.04	81.36
	ConvNeXt style 11×11 + 3×3	77.99	87.86	94.96	51.61	73.01	84.85	43.93	66.22	80.73
LCTC: 7×7 + 3×3 (Ours)	ResNet Style 3×3	78.5	87.57	95.13	53.85	72.75	85.87	47.1	67.57	82.04
	ConvNeXt style 7×7	78.09	87.14	95.04	52.42	71.88	85.59	43.43	65.39	80.88
	ConvNeXt style 7×7 + 3×3	78.37	88.11	95.07	52.15	72.31	84.95	42.77	63.69	79.78
	ConvNeXt style 11×11	77.71	87.22	94.97	52.47	73.22	85.55	44.07	65.84	81.31
	ConvNeXt style 11×11 + 3×3	78.14	86.94	95.05	52.08	70.63	85.98	43.82	63.65	81.95
LCTC: 9×9 (Ours)	ResNet Style 3×3	78.36	86.88	95.18	55.62	72.62	86.9	49.5	67.03	83.9
	ConvNeXt style 7×7	77.17	86.74	94.84	52.76	72.31	85.56	44.23	64.98	81.39
	ConvNeXt style 7×7 + 3×3	77.93	86.97	95.04	51.01	70.59	84.87	41.93	61.63	80.18
	ConvNeXt style 11×11	77.80	86.80	94.99	52.42	72.22	85.39	44.14	65.56	81.16
	ConvNeXt style 11×11 + 3×3	78.25	86.71	95.07	54.59	72.04	86.48	46.88	65.56	82.73
LCTC: 9×9 + 3×3 (Ours)	ResNet Style 3×3	78.77	87.77	95.24	55.94	73.79	86.67	48.82	69.2	82.76
	ConvNeXt style 7×7	77.79	86.65	94.92	52.6	70.51	85.75	43.3	62.16	80.89
	ConvNeXt style 7×7 + 3×3	77.96	87.24	94.98	51.21	70.01	85.24	41.75	61.16	80.64
	ConvNeXt style 11×11	77.92	86.82	95.03	52.71	71.17	86.02	44.33	63.26	82.2
	ConvNeXt style 11×11 + 3×3	77.57	86.71	95.02	53.32	71.75	86.29	46.24	65.3	82.92
LCTC: 11×11 (Ours)	ResNet Style 3×3	79.11	87.06	95.36	56.18	72.11	87.27	49.51	66.15	84.12
	ConvNeXt style 7×7	77.87	86.98	95.06	54.32	72.59	86.42	47.14	67.05	82.71
	ConvNeXt style 7×7 + 3×3	78.34	87.06	95.07	51.93	71.19	85.54	41.77	62.31	80.8
	ConvNeXt style 11×11	77.42	86.68	94.94	53.11	71.43	86.03	44.55	63.45	81.75
	ConvNeXt style 11×11 + 3×3	77.75	86.83	95.01	52.88	71.47	85.93	43.55	62.75	81.4
LCTC: 11×11 + 3×3 (Ours)	ResNet Style 3×3	79.33	87.81	95.41	58.04	74.93	87.8	51.25	69.31	84.64
	ConvNeXt style 7×7	78.32	86.98	95.09	53.31	72.45	86.16	44.89	65.18	82.03
	ConvNeXt style 7×7 + 3×3	78.64	86.78	95.17	54.32	71.27	86.63	45.48	63.62	82.32
	ConvNeXt style 11×11	77.15	85.93	94.87	51.19	69.72	85.45	42.02	61.09	81.1
	ConvNeXt style 11×11 + 3×3	77.42	86.24	94.94	54.48	72.53	86.25	46.67	66.59	82.29
LCTC: 13×13 (Ours)	ResNet Style 3×3	79.41	88.18	95.36	56.89	74.71	87.36	51.06	70.39	84.48
	ConvNeXt style 7×7	77.99	87.11	95.06	54.96	73.32	86.69	47.39	67.2	82.73
	ConvNeXt style 7×7 + 3×3	78.44	87.22	95.13	54.21	72.18	86.34	47.27	65.72	82.95
	ConvNeXt style 11×11	77.57	85.99	95.00	53.51	70.31	86.67	45.63	63.59	83.11
	ConvNeXt style 11×11 + 3×3	77.40	86.53	94.89	53.16	71.62	86.12	45.09	64.23	82.39
LCTC: 13×13 + 3×3 (Ours)	ResNet Style 3×3	79.17	87.96	95.38	57.17	75.08	87.44	50.8	70.67	84.06
	ConvNeXt style 7×7	78.05	86.73	95.02	53.41	71.62	86.12	45.07	65.04	81.76
	ConvNeXt style 7×7 + 3×3	77.76	86.14	95.06	54.09	72.11	86.29	45.69	65.15	82.2
	ConvNeXt style 11×11	77.81	87.43	95.01	51.71	71.77	85.25	41.97	62.61	80.66
	ConvNeXt style 11×11 + 3×3	77.20	86.55	94.81	53.1	71.88	85.87	45.0	65.01	81.91
LCTC: 15×15 (Ours)	ResNet Style 3×3	79.17	87.68	95.28	58.08	73.56	87.58	51.11	67.94	84.36
	ConvNeXt style 7×7	78.34	87.14	95.03	53.86	72.77	86.11	45.12	65.22	81.65
	ConvNeXt style 7×7 + 3×3	77.39	86.40	94.95	51.2	69.42	85.27	42.65	60.88	81.24
	ConvNeXt style 11×11	77.14	86.36	94.82	50.14	69.32	84.49	40.97	60.11	79.81
	ConvNeXt style 11×11 + 3×3	77.67	86.78	94.90	54.44	72.74	86.54	46.37	66.24	82.29
LCTC: 15×15 + 3×3 (Ours)	ResNet Style 3×3	78.72	87.50	95.25	56.28	73.97	87.15	49.5	68.69	83.53
	ConvNeXt style 7×7	77.56	87.01	94.93	53.28	72.15	85.78	45.51	64.84	81.57
	ConvNeXt style 7×7 + 3×3	77.09	86.27	94.76	52.25	70.01	85.41	44.01	62.49	81.16
	ConvNeXt style 11×11	77.40	86.39	94.92	53.59	71.49	86.21	45.48	64.37	82.28
	ConvNeXt style 11×11 + 3×3	78.64	87.46	95.20	54.77	73.2	86.65	46.53	65.4	82.78
LCTC: 17×17 (Ours)	ResNet Style 3×3	79.22	87.77	95.37	56.5	73.3	87.27	50.1	68.23	84.11
	ConvNeXt style 7×7	77.36	87.64	94.89	54.06	73.88	85.84	47.25	68.3	82.19
	ConvNeXt style 7×7 + 3×3	78.03	87.56	95.01	52.75	72.0	85.65	44.32	64.16	81.54
	ConvNeXt style 11×11	77.82	87.40	94.92	51.43	70.57	85.22	42.53	62.68	80.79
	ConvNeXt style 11×11 + 3×3	77.74	86.69	94.99	51.31	69.71	85.53	41.58	60.43	80.83
LCTC: 17×17 + 3×3 (Ours)	ResNet Style 3×3	78.41	86.84	95.26	56.03	73.28	87.16	49.65	67.95	83.74
	ConvNeXt style 7×7	78.14	86.99	94.98	53.44	72.34	86.01	45.02	65.35	81.85
	ConvNeXt style 7×7 + 3×3	78.62	87.64	95.14	55.54	73.87	86.85	47.86	67.22	83.18
	ConvNeXt style 11×11	77.59	87.73	94.84	52.84	74.14	84.63	44.1	67.34	79.57
	ConvNeXt style 11×11 + 3×3	77.33	88.15	94.75	49.29	71.71	84.04	39.85	63.7	78.81
LCTC: 19×19 (Ours)	ResNet Style 3×3	78.54	87.64	95.12	56.63	74.09	87.25	50.02	68.73	83.99
	ConvNeXt style 7×7	78.74	87.66	95.15	56.28	73.79	87.11	49.44	68.74	83.84
	ConvNeXt style 7×7 + 3×3	77.05	86.33	94.89	54.47	72.38	86.78	45.63	64.94	82.81
	ConvNeXt style 11×11	77.66	86.61	95.00	51.58	71.51	84.83	42.48	63.44	79.58
	ConvNeXt style 11×11 + 3×3	77.61	86.59	94.93	50.34	69.39	84.54	41.82	61.29	79.75
LCTC: 19×19 + 3×3 (Ours)	ResNet Style 3×3	78.78	87.34	95.28	56.53	74.59	86.97	50.6	69.95	83.98
	ConvNeXt style 7×7	77.44	86.70	94.91	54.05	72.52	86.09	45.52	65.29	81.52
	ConvNeXt style 7×7 + 3×3	78.14	87.14	95.02	55.82	74.54	86.96	48.97	69.98	83.3
	ConvNeXt style 11×11	78.03	86.64	95.08	53.5	71.21	86.26	45.79	64.16	82.42
	ConvNeXt style 11×11 + 3×3	77.42	86.61	94.91	53.83	72.54	86.17	46.29	66.94	82.22
LCTC: 31×31 (Ours)	ResNet Style 3×3	78.69	86.98	95.30	56.61	73.22	87.08	49.49	66.69	83.68
	ConvNeXt style 7×7	77.54	87.30	94.84	52.36	72.27	85.14	43.56	65.14	8 .
	ConvNeXt style 7×7 + 3×3	76.96	86.38	94.77	53.59	72.14	86.05	45.22	65.22	81.84
	ConvNeXt style 11×11	76.84	86.72	94.71	50.74	70.53	84.61	41.62	61.96	79.96
	ConvNeXt style 11×11 + 3×3	76.77	85.60	94.71	51.42	69.17	85.2	42.12	60.32	80.77
LCTC: 31×31 + 3×3 (Ours)	ResNet Style 3×3	78.47	87.26	95.16	56.27	73.39	87.22	49.66	68.81	83.92
	ConvNeXt style 7×7	77.43	86.56	94.93	53.45	72.74	86.17	45.84	66.41	82.16
	ConvNeXt style 7×7 + 3×3	78.43	87.07	95.17	56.72	73.65	87.6	49.56	68.15	84.22
	ConvNeXt style 11×11	78.00	87.04	94.94	50.66	70.23	84.83	40.71	61.31	79.94
	ConvNeXt style 11×11 + 3×3	77.73	86.54	94.93	53.94	71.65	86.39	44.04	62.19	81.8

Table 8: Comparison of performances against SegPGD attack, of UNet with ConvNeXt encoder and decoder with architectures along with different sized kernels in transposed convolution and different convolution blocks in the decoder for upscaling the feature maps.

Transposed Convolution Kernels	Backbone Style	SegPGD attack iterations: 3			5			10			20			40			100		
		mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc
2×2	ResNet Style 3×3	23.06	46.51	60.04	14.43	35.50	45.30	08.12	24.67	29.88	05.54	18.79	23.72	04.39	14.98	23.70	03.50	11.61	27.93
	ConvNeXt style 7×7	17.94	0.4481	47.96	10.64	33.63	30.64	05.47	21.74	15.8	03.2	14.73	09.81	02.04	0.1047	0.0641	01.35	07.57	04.3
	ConvNeXt style 7×7 + 3×3	17.59	42.55	0.5168	09.88	30.41	0.3233	04.75	16.83	0.1431	02.65	09.46	0.0668	01.68	05.64	0.034	01.0	0.0316	01.94
	ConvNeXt style 11×11	16.39	0.4013	0.485	09.37	28.66	29.63	03.97	14.16	11.41	01.56	06.11	03.56	00.59	02.61	01.31	00.23	00.99	00.51
	ConvNeXt style 11×11 + 3×3	13.97	35.82	45.68	07.61	25.07	28.33	03.4	14.38	12.04	02.21	10.75	05.29	01.57	08.02	03.01	01.07	05.75	01.85
3×3	ResNet Style 3×3	23.37	46.33	60.78	15.26	38.0	46.51	09.26	29.64	31.9	06.78	24.18	26.95	05.71	20.39	28.69	05.02	16.11	33.12
	ConvNeXt style 7×7	18.48	43.81	54.97	09.51	29.92	34.86	03.63	15.1	13.03	01.64	08.23	04.51	01.0	05.12	02.13	00.59	02.84	00.89
	ConvNeXt style 7×7 + 3×3	19.08	46.97	47.74	11.15	34.6	29.9	05.96	22.62	15.67	03.61	15.04	09.33	02.17	09.18	05.86	01.29	06.02	03.55
	ConvNeXt style 11×11	16.2	39.11	50.93	09.52	29.32	32.61	04.93	20.31	14.82	02.86	13.94	06.46	02.05	10.94	03.58	01.4	08.23	02.21
	ConvNeXt style 11×11 + 3×3	18.54	41.34	55.56	10.25	30.11	36.0	04.8	19.25	13.94	02.41	11.87	04.56	01.59	07.78	02.11	01.11	04.21	01.09
5×5 (Ours)	ResNet Style 3×3	24.23	51.8	57.82	16.16	42.98	43.29	10.11	32.79	30.3	07.32	25.16	27.42	06.02	19.04	31.25	05.16	14.03	37.36
	ConvNeXt style 7×7	17.59	43.57	51.41	09.9	30.84	33.14	04.74	18.3	14.55	02.23	09.47	05.21	01.47	06.03	02.32	00.97	03.64	01.28
	ConvNeXt style 7×7 + 3×3	18.7	43.18	52.74	10.56	31.41	33.32	04.87	18.5	14.78	02.49	10.84	05.59	01.39	05.61	02.69	00.91	03.38	01.46
	ConvNeXt style 11×11	18.96	44.79	53.09	09.85	29.77	32.88	03.89	14.94	12.6	01.94	08.29	04.58	01.03	04.72	01.86	00.48	02.63	00.75
	ConvNeXt style 11×11 + 3×3	13.38	33.61	45.0	06.84	20.94	25.99	02.51	08.85	08.5	01.18	04.39	02.62	00.71	02.48	01.07	00.48	01.52	00.53
5×5 + 3×3 (Ours)	ResNet Style 3×3	25.03	53.96	58.89	16.61	45.8	42.18	10.79	37.16	27.34	08.0	29.62	21.71	06.16	21.69	22.25	04.83	13.87	28.97
	ConvNeXt style 7×7	17.65	44.79	48.41	09.79	31.78	28.51	04.62	18.37	11.12	02.58	10.89	04.61	01.52	06.59	02.3	01.0	04.04	01.33
	ConvNeXt style 7×7 + 3×3	18.31	42.75	49.26	09.89	28.58	30.02	03.78	12.49	11.08	01.34	04.76	03.54	00.48	02.13	01.45	00.19	00.88	00.76
	ConvNeXt style 11×11	17.87	40.62	52.77	09.74	27.94	34.21	04.65	14.98	14.34	02.0	05.95	04.77	01.07	02.81	01.71	00.32	00.98	00.63
	ConvNeXt style 11×11 + 3×3	20.84	46.95	53.91	11.86	33.96	34.86	05.65	19.8	16.66	02.83	10.73	08.2	01.59	06.21	04.68	01.11	04.2	02.62
LCTC: 7×7 (Ours)	ResNet Style 3×3	26.53	53.05	61.16	17.75	43.31	46.99	10.26	30.92	32.62	07.17	23.05	27.52	05.69	17.24	29.48	04.37	11.29	35.16
	ConvNeXt style 7×7	17.64	43.32	47.8	09.95	30.43	28.02	04.21	15.08	10.07	01.86	07.18	03.55	00.99	03.52	01.42	00.68	01.89	00.74
	ConvNeXt style 7×7 + 3×3	16.64	40.11	50.56	09.75	29.72	32.23	04.95	19.4	14.47	02.87	13.23	06.4	02.06	09.55	03.45	01.59	07.24	02.04
	ConvNeXt style 11×11	17.37	45.07	47.32	08.86	30.03	26.48	03.47	14.22	07.94	01.53	06.55	02.45	00.93	03.9	01.2	00.61	02.3	00.64
	ConvNeXt style 11×11 + 3×3	17.07	42.3	48.78	09.31	28.04	28.88	03.82	13.79	09.54	01.8	07.11	03.04	01.03	04.3	01.45	00.53	02.61	00.77
LCTC: 7×7 + 3×3 (Ours)	ResNet Style 3×3	24.03	52.08	57.43	16.21	43.38	43.01	09.99	32.77	30.22	07.38	26.16	26.11	06.31	22.42	28.32	05.35	17.41	33.09
	ConvNeXt style 7×7	16.19	43.4	48.59	09.02	32.38	29.17	04.23	19.63	10.47	02.46	12.18	03.99	01.53	06.85	01.97	00.91	03.94	01.1
	ConvNeXt style 7×7 + 3×3	16.04	39.67	48.16	08.94	27.45	30.33	03.81	14.69	12.79	01.91	09.17	04.63	01.2	06.16	01.95	00.84	03.96	00.95
	ConvNeXt style 11×11	18.08	46.24	50.64	10.18	33.17	31.35	04.49	18.33	12.04	02.01	07.98	04.55	01.04	03.91	02.17	00.45	01.7	01.2
	ConvNeXt style 11×11 + 3×3	15.31	37.02	52.08	07.62	24.44	32.35	03.3	15.04	12.35	01.92	10.24	05.12	01.32	07.37	02.63	00.91	05.07	01.39
LCTC: 9×9 (Ours)	ResNet Style 3×3	25.26	50.75	60.85	16.88	41.02	47.16	09.44	28.03	33.87	06.23	20.76	28.91	04.71	16.45	29.14	03.69	12.63	31.93
	ConvNeXt style 7×7	18.11	44.53	50.69	10.46	31.69	32.26	04.92	18.52	14.48	02.86	12.13	06.36	02.1	09.3	03.51	01.5	06.59	01.9
	ConvNeXt style 7×7 + 3×3	16.2	39.55	50.82	09.0	28.53	33.31	04.07	17.03	15.6	02.14	10.13	07.12	01.38	05.91	03.74	00.71	02.56	01.82
	ConvNeXt style 11×11	17.02	43.01	48.45	08.92	28.35	28.13	03.64	14.36	10.06	01.17	06.28	03.11	00.55	04.04	01.35	00.32	02.76	00.77
	ConvNeXt style 11×11 + 3×3	19.34	43.6	54.41	10.71	31.22	33.98	04.6	15.76	12.75	01.98	07.78	04.04	00.95	03.95	01.69	00.51	01.96	00.78
LCTC: 9×9 + 3×3 (Ours)	ResNet Style 3×3	24.87	55.04	57.35	17.0	46.34	42.08	10.88	36.04	28.55	07.91	28.17	22.86	06.02	21.13	22.87	04.63	14.45	27.39
	ConvNeXt style 7×7	16.56	36.5	53.58	08.74	23.95	35.67	04.01	13.92	16.64	02.13	08.87	06.34	01.38	06.37	02.27	01.01	04.8	01.11
	ConvNeXt style 7×7 + 3×3	16.03	36.92	51.5	08.8	25.53	33.15	03.64	13.95	12.25	01.61	06.02	04.08	00.83	02.72	01.83	00.37	01.13	00.87
	ConvNeXt style 11×11	16.42	39.19	51.71	08.32	26.64	31.11	03.66	15.7	11.61	01.94	10.11	04.4	01.19	06.75	02.23	00.83	04.83	01.36
	ConvNeXt style 11×11 + 3×3	18.72	41.83	55.48	10.38	29.7	36.72	04.74	18.16	17.44	02.49	11.2	07.18	01.69	08.21	03.56	01.24	06.07	01.93
LCTC: 11×11 (Ours)	ResNet Style 3×3	26.02	48.81	63.76	16.8	39.62	49.72	09.62	29.4	34.22	06.85	24.07	27.66	05.63	20.38	26.45	04.56	15.64	28.86
	ConvNeXt style 7×7	19.04	45.39	52.63	10.17	32.3	32.46	04.58	20.16	13.36	02.44	13.63	05.33	01.74	10.13	03.04	01.21	07.07	01.7
	ConvNeXt style 7×7 + 3×3	16.08	39.09	53.1	08.86	28.27	35.06	03.94	16.77	15.75	02.25	11.87	06.31	01.32	07.98	02.72	00.82	05.14	01.28
	ConvNeXt style 11×11	18.09	40.72	53.7	09.93	29.6	34.68	04.55	18.22	14.17	02.21	10.51	05.2	01.38	06.35	02.34	00.96	03.84	01.28
	ConvNeXt style 11×11 + 3×3	15.29	37.2	50.71	07.6	25.19	30.65	03.17	15.06	09.58	01.78	10.21	03.07	01.3	07.74	01.39	01.0	05.6	00.88
LCTC: 11×11 + 3×3 (Ours)	ResNet Style 3×3	27.49	53.08	64.13	18.15	43.51	49.36	10.29	31.12	33.17	07.08	23.3	26.82	05.14	16.14	27.32	03.77	09.6	31.61
	ConvNeXt style 7×7	16.14	40.65	50.39	08.08	27.2	31.4	03.34	15.36	12.29	01.93	09.35	03.9	01.36	05.77	01.76	00.92	03.51	00.83
	ConvNeXt style 7×7 + 3×3	17.7	39.71	54.64	09.71	26.92	35.8	04.32	13.93	15.8	02.37	08.49	06.7	01.59	05.85	03.43	01.09	03.87	01.83
	ConvNeXt style 11×11	14.62	34.73	49.37	07.26	22.21	29.37	02.76	12.24	10.69	01.23	07.06	04.16	00.71	04.71	01.96	00.63	03.65	00.96
	ConvNeXt style 11×11 + 3×3	18.76	44.6	51.49	10.07	31.15	30.26	04.4	17.02	10.56	02.31	08.7	03.5	01.34	04.85	01.66	00.73	02.56	00.81
LCTC: 13×13 (Ours)	ResNet Style 3×3	28.51	57.18	63.94	19.71	48.99	50.08	11.99	37.69	33.26	08.31	28.29	26.23	06.17	21.38	25.65	04.83	15.34	29.52
	ConvNeXt style 7×7	20.9	46.62	55.13	12.32	34.21	35.91	06.14	21.39	16.39	03.15	13.44	07.51	02.16	10.21	04.3	01.41	06.61	02.54
	ConvNeXt style 7×7 + 3×3	20.13	42.92	57.7	11.38	29.96	39.57	04.85	15.81	19.37	02.54	09.48	09.47	01.65	06.45	05.61	00.86	03.83	03.0
	ConvNeXt style 11×11	18.65	39.48	56.4	10.02	27.46	38.02	04.69	17.27	19.03	02.47	11.35	08.76	01.39	07.95	04.12	00.9	06.02	02.11
	ConvNeXt style 11×11 + 3×3	18.95	42.88	55.82	10.68	31.21	35.69	04.92	18.29	12.63	02.35	09.29	03.78	01.26	05.02	01.6	00.79	02.56	00.72
LCTC: 13×13 + 3×3 (Ours)	ResNet Style 3×3	28.08	58.22	63.4	19.4	50.01	48.89	12.04	39.2	32.11	08.77	31.09	24.9	06.46	22.51	23.98	04.34	13.59	28.41
	ConvNeXt style 7×7	18.42	43.52	51.26	10.23	30.56	30.5	04.37	16.41	11.29	02.08	09.09	04.39	01.35	06.65	02.49	00.86	04.28	01.37
	ConvNeXt style 7×7 + 3×3	16.7	41.09	50.56	08.54	26.94	30.39	03.31	13.44	11.22	01.53	06.72	04.13	01.02	04.01	02.01	00.56	02.07	01.03
	ConvNeXt style 11×11	14.1	36.4	47.79	07.01	23.32	27.54	02.87	12.05	10.26	01.51	06.85	04.56	01.13	05.21	03.2	01.13	05.21	03.2
	ConvNeXt style 11×11 + 3×3	18.14	43.14	52.31	09.55	28.16	32.41	03.54	12.57	11.22	01.52	06.19	03.35	00.97	03.66	01.52	00.65	02.11	00.84
LCTC: 15×15 (Ours)	ResNet Style 3×3	29.41	51.54	66.7	19.96	41.26	55.14	11.51	29.26	41.04	07.17	20.8	31.7	05.13	15.79	28.53	03.9	11.59	29.37
	ConvNeXt style 7×7	18.62	44.42	51.51	10.55	32.47	32.54	04.69	18.66	12.4	02.64	11.81	04.44	01.67	07.93	02.02	01.29	05.26	01.21
	ConvNeXt style 7×7 + 3×3	17.63	37.55	55.52	09.13	23.28	37.06	03.46	09.05	15.8	01.41	03.5	06.96	00.77	01.62	04.02	00.51	01.04	02.48
	ConvNeXt style 11×11	15.24	36.62	49.45	08.05	24.89	31.93	03.68	14.62	13.68	02.03	07.68	05.66	01.26	04.48	03.25	00.61	02.51	01.87
	ConvNeXt style 11×11 + 3×3	19.01	44.78	52.74	10.35	31.98	32.35	04.38	19.15	12.42	02.31	12.1	04.84	01.53	07.72	02.24	01.06	04.86	01.17
LCTC: 15×15 + 3×3 (Ours)	ResNet Style 3×3	26.38	53.79	61.59	17.9	45.03	47.36	10.79	33.9	31.75	07.14	25.02	25.01	05.43	18.97	25.39	04.18	13.25	30.47
	ConvNeXt style 7×7	19.81	42.18	53.73	11.14	28.25	37.03	04.72	13.45	17.41	01.92	05.27	07.64	01.06	03.07	04.37	00.7	01.97	02.61
	ConvNeXt style 7×7 + 3×3	17.53	39.4	54.39	09.51	26.67	35.52	03.73	12.39	13.27	00.93	04.2	03.22	00.44	02.14	01.01	00.16	00.94	00.36
	ConvNeXt style 11×11	16.69	39.29	52.18	08.78	27.55	32.8	03.72	17.08	12.37	02.06	11.89	03.94	01.38	08.59	02.0	00.99	06.41	01.24
	ConvNeXt style 11×11 + 3×3	19.15	41.08	55.96	10.71	29.12	37.58	05.28	19.35	18.54	02.88	13.69	08.43	02.0	11.2	04.47	01.41	09.0	02.39
LCTC: 17×17 (Ours)	ResNet Style 3×3	27.74	53.24	64.48	18.51	43.47	51.02	10.72	32.21	36.08	07.43	25.5	28.78	05.85	20.69	28.85	04.93	16.94	32.03
	ConvNeXt style 7×7	19.82	46.01	54.48	11.09	32.71	36.79	05.35	19.1	17.49	02.67	10.25	07.1	01.87	07.1	02.98	01.25	04.64	01.33
	ConvNeXt style 7×7 + 3×3	16.96	38.94	54.19	09.23	26.92	36.22	04.47	16.8	16.9	02.45	11.16	07.09	01.61	08.12	03.53	01.03	05.47	01.95
	ConvNeXt style 11×11	13.72	34.03	48.09	06.57	20.94	26.93	02.32	09.22	07.84	01.08	04.61	02.28	00.63	02.55	01.0	00.35	01.33	00.47
	ConvNeXt style 11×11 + 3×3	14.47	33.55	52.16	07.33	20.91	32.91	02.82	09.75	13.11	01.28	04.95	04.96	00.79	03.01	02.52	00.47	01.55	01.32
LCTC: 17×17 + 3×3 (Ours)	ResNet Style 3×3	26.97	54.13	62.04	18.41	45.5	47.66	11.05	34.55	32.01	07.43	25.65	24.78	05.07	17.38	24.12	03.51	10.48	27.4
	ConvNeXt style 7×7	17.96	41.81	54.93	09.08	27.73	35.7	03.85	15.38	15.51	01.95	09.2	05.76	01.17	05.74	02.35	00.86	03.99	01.25
	ConvNeXt style 7×7 + 3×3	19.86	42.55	56.89	10.29	28.26	37.83	04.43	15.77	16.52	01.93	08.45	06.36	01.0	05.27	02.61	00.68	03.7	01.31
	ConvNeXt style 11×11	16.84	44.91	45.35	09.41	31.25	26.04	03.8	14.79	09.08	01.4	05.56	02.83	00.47	02.05	01.03	00.17	01.1	00.54
	ConvNeXt style 11×11 + 3×3	14.06	38.28	46.37	06.95	24.78	27.4	02.92	14.65	10.36	01.55	09.22	03.9	00.96	06.21	02.04	00.68	04.61	01.29
LCTC: 19×19 (Ours)	ResNet Style 3×3	27.64	52.62	64.53	18.46	42.79	51.19	10.49	30.27	36.37	06.92	22.02	28.21	05.17	17.07	26.09	03.99	12.16	27.92
	ConvNeXt style 7×7	20.28	46.96	56.75	10.06	30.29	36.2	03.5	13.07	14.56	01.51	06.13	05.64	00.72	03.24	02.4	00.64	02.2	01.4
	ConvNeXt style 7×7 + 3×3	18.14	39.86	56.98	09.34	26.13	37.16	03.5	13.58	14.24	01.52	07.79	04.4	00.83	05.76	01.83	00.54	03.82	00.93
	ConvNeXt style 11×11	15.85	40.55	46.13	08.33	28.21	25.47	03.27	15.94	07.52	01.8	10.44	02.95	01.3	07.65	01.81	00.97	05.34	01.22
	ConvNeXt style 11×11 + 3×3	16.17	37.68	50.29	08.47	25.22	31.86	03.84	14.74	13.88	01.93	08.84	06.27	01.2	05.41	03.28	00.79	03.68	02.02
LCTC: 19×19 + 3×3 (Ours)	ResNet Style 3×3	28.62	56.15	63.93	19.19	47.17	48.9	10.96	34.41	31.54	07.15	25.67	23.6	05.25	19.32	21.84	04.24	15.12	24.17
	ConvNeXt style 7×7	17.45	40.44	51.24	09.13	27.29	31.41	03.56	13.76	10.77	01.61	06.83	03.35	00.94	04.0	01.37	00.56	02.22	00.54
	ConvNeXt style 7×7 + 3×3	20.9	48.61	54.54	11.59	35.05	33.3	04.6	18.94	11.57	02.05	10.9	03.51	01.42	07.83	01.63	00.9	05.28	00.88
	ConvNeXt style 11×11	19.01	41.41	55.17	10.56	28.9	37.54	05.23	19.35	17.13	02.92	12.99	07.27	02.05	09.7	03.82	01.42	07.14	02.18
	ConvNeXt style 11×11 + 3×3	17.98	44.39	52.86	09.44	30.77	32.15	03.36	14.64	09.76	01.14	05.22	02.06	00.48	02.49	00.6	00.17	01.16	00.26
LCTC: 31×31 (Ours)	ResNet Style 3×3	26.44	50.1	63.1	17.8	39.96	50.81	10.44	29.22	36.6	06.67	21.09	28.13	04.91	16.07	23.92	03.65	10.93	23.02
	ConvNeXt style 7×7	17.75	41.69	51.94	09.26	27.63	32.32	03.52	12.48	11.73	01.37	04.95	04.18	00.62	02.57	01.93	00.37	01.68	01.02
	ConvNeXt style 7×7 + 3×3	16.53	40.82	50.9	08.0	25.9	30.59	02.79	11.15	10.24	01.24	05.19	03.11	00.56	02.42	01.12	00.34	01.4	00.53
	ConvNeXt style 11×11	13.08	31.95	45.87	05.85	17.71	25.83	02.06	07.35	08.65	00.88	03.15	02.74	00.39	01.55	01.05	00.26	01.15	00.55
	ConvNeXt style 11×11 + 3×3	15.42	35.92	51.53	07.44	21.84	31.72	02.43	09.33	10.18	00.85	03.85	02.59	00.41	01.79	01.04	00.22	00.99	00.56
LCTC: 31×31 + 3×3 (Ours)	ResNet Style 3×3	27.41	54.28	64.15	18.27	44.66	49.97	11.02	33.64	34.65	07.24	25.06	26.54	05.39	18.81	22.82	04.3	14.03	22.46
	ConvNeXt style 7×7	18.76	40.98	55.63	10.33	28.32	38.72	04.95	18.11	19.82	02.74	12.53	08.41	01.69	08.6	03.73	01.03	05.94	01.75
	ConvNeXt style 7×7 + 3×3	20.55	44.15	58.99	10.65	30.05	40.0	04.69	17.49	18.07	02.65	11.74	07.33	01.6	08.05	03.37	01.09	05.88	01.79
	ConvNeXt style 11×11	14.47	36.3	49.11	07.12	22.87	29.65	02.57	11.14	10.67	01.24	06.52	03.69	00.9	05.06	01.68	00.63	03.65	00.96
	ConvNeXt style 11×11 + 3×3	13.59	32.71	49.91	06.09	18.39	29.59	01.96	06.7	08.76	00.79	02.53	02.11	00.4	01.47	00.89	00.12	00.65	00.45

Table 9: Comparison of performance of Large Context Transposed Convolutions (LCTC) with very large: 31×31 kernels in transposed convolution to large (7×7 to 17×17) kernels. All have a parallel 3×3 kernel, as shown in Figure 4 (bottom left). Here we observe the saturation of performance for very large kernels for upsampling. This comparison is for the same encoder (ConvNeXt) and same ResNet-like building blocks in the decoder (our baseline). The complete table is provided in Appendix B.1.

Transposed Convolution Kernels	Test Accuracy			FGSM attack epsilon						SegPGD attack iterations		
				$\frac{1}{255}$			$\frac{8}{255}$			20		
	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc
7×7	78.50	87.57	95.13	53.85	72.75	85.87	47.10	67.57	82.04	7.38	**26.16**	26.11
11×11	**79.33**	**87.81**	**95.41**	**58.04**	**74.93**	**87.80**	**51.25**	69.31	**84.64**	7.08	23.30	**26.82**
15×15	78.72	87.50	95.25	56.28	73.97	87.15	49.50	68.69	83.53	7.14	25.02	25.01
17×17	78.41	86.84	95.26	56.03	73.28	87.16	49.65	67.95	83.74	**7.43**	25.65	24.78
19×19	78.78	87.34	95.28	56.53	74.59	86.97	50.60	**69.95**	83.98	7.15	25.67	23.60
31×31	78.47	87.26	95.16	56.27	73.39	87.22	49.66	68.81	83.92	7.24	25.06	26.54

Table 10: Adversarially trained models using FGSM and PGD from Table 2 tested against adversarial attacks on UNet with ConvNeXt encoder and decoder with different sized kernels in the transposed convolution for upscaling, while keeping rest of the architecture identical.

Transposed Convolution Kernels	Clean Test Accuracy			FGSM attack epsilon						SegPGD attack iterations					
				$\frac{1}{255}$			$\frac{8}{255}$			3			20		
	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc
FGSM training															
2×2 (baseline)	78.57	86.68	95.23	54.28	70.80	86.91	52.45	68.38	**86.26**	26.59	48.99	67.71	7.6	24.06	31.37
LCTC: 7×7 (Ours)	78.41	86.22	95.20	56.87	72.92	87.70	51.31	68.4	85.17	28.11	53.39	66.30	8.36	28.54	28.13
LCTC: 11×11 + 3×3 (Ours)	**79.57**	**88.1**	**95.3**	**57.90**	**74.64**	87.61	52.15	**70.23**	84.96	**30.37**	**55.54**	**68.3**	**9.4**	**29.79**	**32.37**
PGD training with 3 attack iterations															
2×2 (baseline)	75.33	84.66	94.39	53.87	72.17	86.58	58.57	73.93	89.01	29.38	57.82	66.67	9.39	33.15	28.11
LCTC: 7×7 (Ours)	75.79	84.89	94.38	54.82	72.31	**86.80**	61.29	74.33	89.96	31.12	58.36	68.58	10.24	**33.99**	31.14
LCTC: 11×11 + 3×3 (Ours)	**75.90**	**86.60**	94.30	**56.27**	**75.66**	86.68	**63.02**	**76.17**	**90.42**	**33.50**	58.34	**71.50**	**10.77**	32.23	**37.36**

Table 11: Comparison of performances of different encoders in the UNet-like architecture. All architectures here have the baseline 2×2 transposed convolution kernel for upsampling followed by 3×3 convolution kernels in the decoder blocks. For more results please refer to Table 12.

Encoder	Test Accuracy			FGSM attack epsilon						SegPGD attack iterations		
				$\frac{1}{255}$			$\frac{8}{255}$			20		
	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc
ResNet50	67.69	79.04	92.80	36.78	58.41	78.16	32.60	52.63	74.56	4.98	19.28	21.07
ConvNeXt tiny	78.45	86.66	95.20	53.76	70.62	86.32	47.33	64.58	83.16	5.54	18.79	23.72
SLaK tiny	78.82	87.01	95.17	55.22	71.72	86.97	48.69	66.45	83.57	8.45	25.42	32.37

Table 12: Comparison of performances of different encoders in the UNet-like architecture. All architectures here have the baseline 2×2 transposed convolution kernel followed by 3×3 convolution kernels in the decoder block.

Encoder	Test Accuracy			FGSM attack epsilon						SegPGD attack iterations													
				$\frac{1}{255}$			$\frac{8}{255}$			3			5			10			20			40			100		
	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc
ResNet50	67.69	79.04	92.80	36.78	58.41	78.16	32.60	52.63	74.56	16.18	37.46	50.04	11.32	30.59	38.98	7.21	23.76	27.58	4.98	19.28	21.07	3.95	16.49	18.35	3.09	13.87	15.87
ConvNeXt tiny	78.45	86.66	95.20	53.76	70.62	86.32	47.33	64.58	83.16	23.06	46.51	60.04	14.43	35.50	45.30	8.12	24.67	29.88	5.54	18.79	23.72	4.39	14.98	23.70	3.50	11.61	27.93
SLaK tiny	78.82	87.01	95.17	55.22	71.72	86.97	48.69	66.45	83.57	26.71	50.92	64.04	19.28	43.51	52.88	12.24	33.65	39.78	8.45	25.42	32.37	6.22	19.58	29.06	-	-	-

Table 13: Comparison of performances of different upsampling methods in the UNet-like architecture. All architectures here have the baseline i.e. ConvNeXt encoder and a ResNet style 3×3 or ConvNext style 7 ×7+3×3 convolution kernels in the decoder block.

Upsampling Method	Convolution Kernel in Decoder blocks	Test Accuracy			FGSM attack epsilon $\frac{1}{255}$			FGSM attack epsilon $\frac{3}{255}$			SegPGD attack iterations 3			SegPGD attack iterations 5			SegPGD attack iterations 10			SegPGD attack iterations 20			SegPGD attack iterations 40			SegPGD attack iterations 100		
		mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc
Pixel Shuffle	ResNet Style 3×3	78.54	87.32	95.18	53.82	71.58	85.88	46.67	65.03	81.71	23.08	48.18	56.54	15.06	38.85	41.71	9.17	29.43	28.17	6.69	23.43	24.05	5.69	19.61	25.71	4.80	15.53	32.10
	ConvNeXt Style 7×7+3×3	77.10	85.90	94.88	51.78	69.68	85.44	43.80	62.24	81.06	17.52	40.16	50.31	9.43	27.37	30.37	3.53	12.25	10.93	1.41	5.42	3.74	0.78	3.04	1.55	0.52	1.96	0.93
Nearest Neighbour Interpolation	ResNet Style 3×3	78.40	88.16	95.09	52.68	73.51	84.55	46.08	67.96	80.22	22.82	53.16	51.75	15.34	44.53	36.21	10.02	34.83	23.84	7.65	27.89	20.48	6.43	23.23	21.48	5.40	17.34	28.05
	ConvNeXt Style 7×7+3×3	77.86	86.92	94.97	50.71	71.21	84.45	41.97	64.92	78.89	15.77	44.36	42.09	8.56	30.25	23.74	2.96	12.56	7.19	1.27	5.70	2.10	0.52	2.08	0.75	0.17	0.85	0.35
Transposed Convolution 2×2	ResNet Style 3×3	78.45	86.66	95.20	53.76	70.62	86.32	47.33	64.58	83.16	23.06	46.51	60.04	14.43	35.50	45.30	8.12	24.67	29.88	5.54	18.79	23.72	4.39	14.98	23.70	3.50	11.61	27.93
	ConvNeXt Style 7×7+3×3	77.24	86.03	94.84	51.09	70.53	85.29	43.52	63.74	81.18	17.59	42.55	51.68	9.88	30.41	32.33	4.75	16.83	14.31	2.65	9.46	6.68	1.68	5.64	3.4	1.0	3.16	1.94
LCTC: 11×11+3×3 (Ours)	ResNet Style 3×3	79.33	87.81	95.41	58.04	74.93	87.8	51.25	69.31	84.64	27.49	53.08	64.13	18.15	43.51	49.36	10.29	31.12	33.17	7.08	23.3	26.82	5.14	16.14	27.32	3.77	9.6	31.61
	ConvNeXt Style 7×7+3×3	78.64	86.78	95.17	54.32	71.27	86.63	45.48	63.62	82.32	17.7	39.71	54.64	9.71	26.92	35.8	4.32	13.93	15.8	2.37	8.49	6.7	1.59	5.85	3.43	1.09	3.87	1.83

Table 14: Comparing latency and number of parameters for models from Figure 5.

Upsampling Method	Latency (ms)	No. of Params
Pixel Shuffle	0.26	17.11 M
Trans. Conv. 3×3	0.27	16.43 M
LCTC 11×11+3×3	0.38	16.54 M

Table 15: Comparison of performances of adversarially trained *SotA* Image Restoration Networks. The considered architectures use Pixel Shuffle for Upsampling, we propose replacing the Pixel Shuffle with Transposed Convolution operations using the large filter. Testing for image deblurring on GoPro dataset.

Network	Upsampling Method	Test Accuracy		PGD attack iterations 5		10		20	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
NAFNet + ADV	Pixel Shuffle	29.91	0.9291	15.76	0.5228	13.91	0.4445	12.73	0.3859
	Transposed Conv 3×3	31.26	0.9448	15.89	**0.5390**	13.43	**0.4627**	11.62	**0.4098**
	LCTC: 7×7 + 3 ×3 (Ours)	31.21	0.9446	**16.46**	0.5061	**14.55**	0.4211	**13.31**	0.3688
	LCTC: 11×11 + 3×3 (Ours)	30.70	0.9390	13.68	0.4857	11.91	0.4085	10.92	0.3604

Table 16: Performance of different upsampling methods in NAFNet in real-world ODD setting by applying 2D common corruption [39] (severity=3) on GoPro dataset. We use all common corruptions from [39] GitHub repository except weather conditions (ideally these should happen before the motion blurring) and blurring (since the images are already motion blurred). Here "Mean" is performance over all the considered corruptions:

Common Corruption	Upsampling Method: Pixel Shuffle		Trans. Conv. 3×3		**LCTC 11×11+3×3**	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Gaussian Noise	4.8501	0.0104	8.7346	0.1014	13.6475	0.1523
Shot Noise	4.8616	0.0127	8.9524	0.0984	13.2464	0.1564
Impulse Noise	5.0154	0.0214	9.2451	0.1065	14.8425	0.187
Brightness	32.3199	0.9576	30.676	0.9394	30.4098	0.9361
Contrast	26.5941	0.7759	25.9743	0.7561	25.8733	0.7525
Elastic Transform	17.944	0.6392	19.7686	0.703	19.7672	0.702
Pixelate	4.4977	0.246	4.4999	0.246	4.4958	0.246
JPEG Compression	25.2767	0.8095	25.1014	0.8032	25.3788	0.8104
Speckle Noise	4.8287	0.0158	9.2336	0.1044	14.6622	0.2473
Saturate	32.1969	0.958	30.5904	0.9399	30.3005	0.9365
Mean	15.8385	0.4447	17.2776	0.4798	**19.262**	**0.5127**

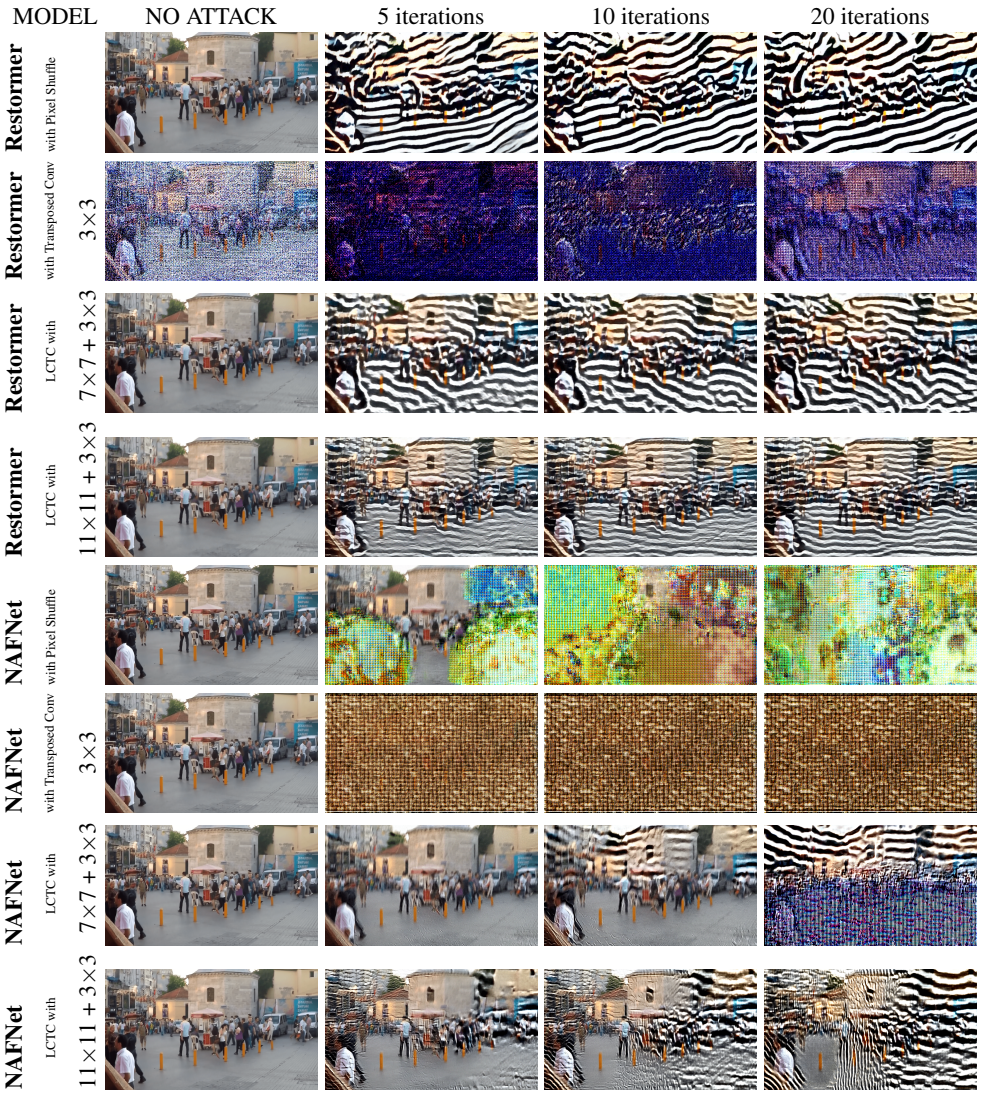


Fig. 10: Comparing images reconstructed by all models after **PGD attack** on variants of Upsampling.

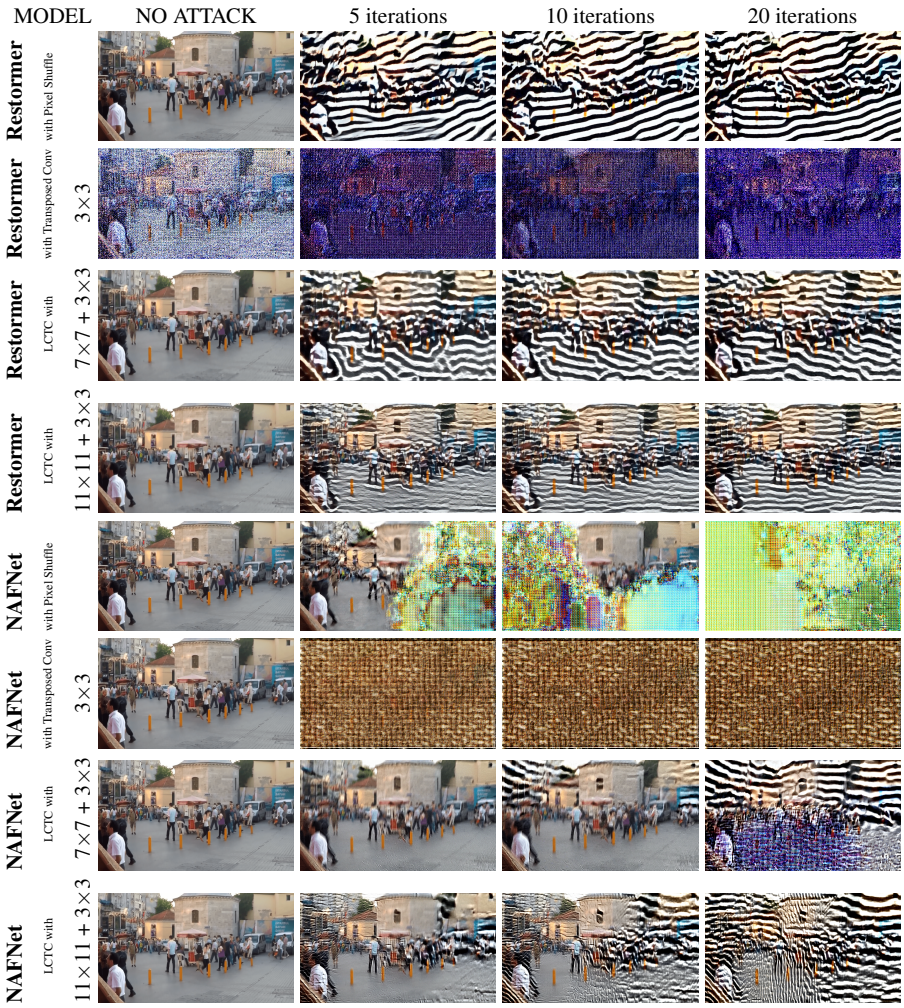


Fig. 11: Comparing images reconstructed by the considered variants of the SotA models after CosPGD attack [3]. We observe that the originally proposed Restormer and NAFNet architectures that use Pixel Shuffle for upsampling perform considerably well under no adversarial attack but even a small perturbation of $\epsilon=\frac{8}{255}$ causes ringing and other spectral artifacts to occur in the deblurred images to the extent that the images are unrecognizable. However, on replacing the Pixel Shuffle operation in these architectures with a Transposed Convolution operation with a large kernel ($11\times11+3\times3$), we observe a significant reduction in the spectral artifacts in the images restored under adversarial attack while the image restored under no attack are very comparable to those restored by the original architectures.

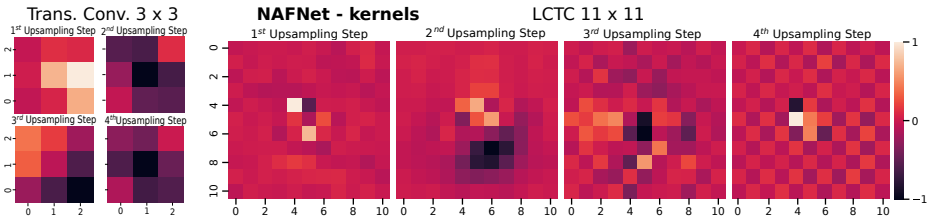


Fig. 12: Normalized kernel weights from a random channel each for the models from Figure 5.

Table 17: Comparison of performance of STTR-light architecture with different sized kernels in transposed convolution for upscaling the feature maps in the feature extractor.

Transposed Convolution Kernels	**Test Accuracy**		PGD Attack					
			3 Iterations		5 Iterations		10 Iterations	
	epe↓	3px error↓	epe↓	3px error↓	epe↓	3px error↓	epe↓	3px error↓
STTR-light [50] reported	0.5	1.54						
3×3 [50] reproduced	0.4927	1.54	4.05	18.46	4.07	18.59	4.08	18.6
LCTC: 7×7 (Ours)	0.487	1.52	4.26	19.09	4.289	19.21	4.294	19.23
LCTC: 7×7 + 3×3 (Ours)	**0.4788**	**1.50**	4.02	18.3	4.0474	18.43	4.05	18.45
LCTC: 9×9 (Ours)	0.4983	**1.50**	4.36	**18.02**	4.386	**18.14**	4.39	**18.16**
LCTC: 11×11 +3×3 (Ours)	0.5124	1.57	**4.004**	18.29	**4.028**	18.42	**4.032**	18.44

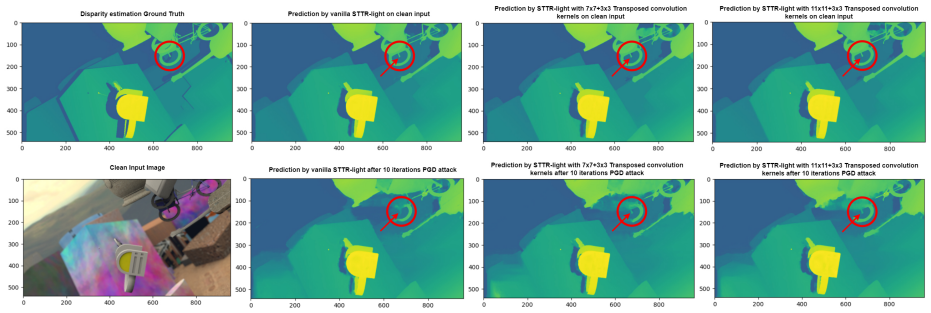


Fig. 13: Visual comparison of Disparity Estimation predictions by a vanilla STTR-light as proposed by [50] and our proposed modification of increasing the kernel size of the transposed convolution operation in the "feature extractor" module of the architecture from 3×3 to LCTC with 7×7+3×3 and 11×11+3×3 sized kernels. As shown by the region in the red circle, both vanilla architecture and the architecture with our proposed change perform well compared to the ground truth on clean images. However, under 10 iteration PGD adversarial attack, we observe small protrusion's depth(shown by the red arrow) is incorrectly estimated by the vanilla architecture, however, the architectures with LCTC preserve the prediction of the disparity.

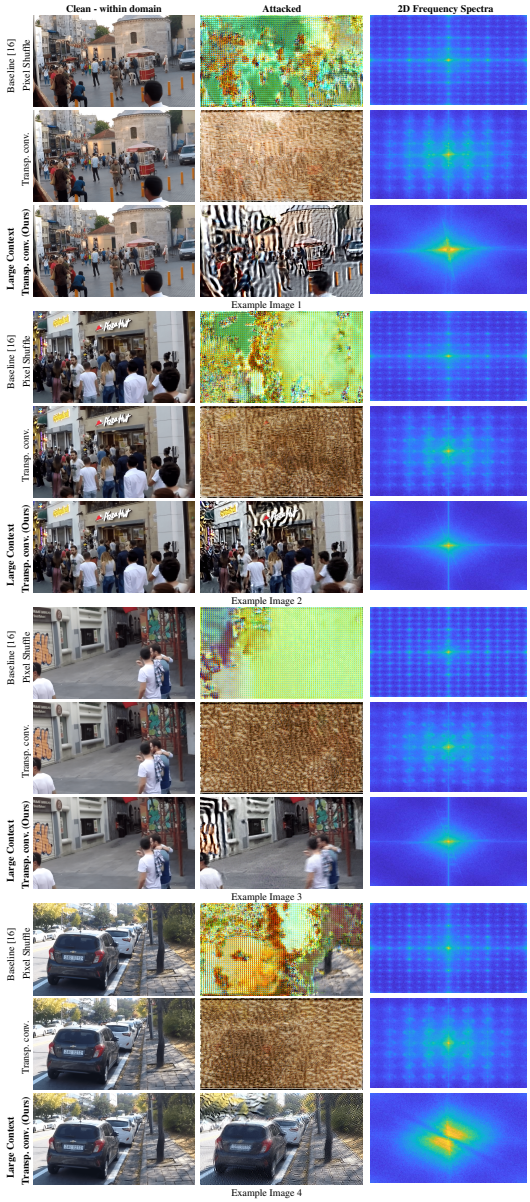


Fig. 14: This is extension to Fig. 1, here we observe the same artifacts both in the spatial and frequency domain as that observed in Fig. 1. Here we perform Image restoration using NAFNet [16] variants on GoPro [63]. Normal Transposed Convolution uses 3×3 sized kernels. Large Context Transposed Convolution uses kernels of size 7×7+3×3 for upsampling. LCTC significantly increases the model's stability during upsampling, observable in the restored image under attack and the frequency spectrum. The procedure for obtaining the 2D Frequency Spectra has been explained in Appendix A.6.

Acknowledgements.

Margret Keuper acknowledges funding by the DFG Research Unit 5336 - Learning to Sense. The OMNI cluster of the University of Siegen was used for some of the initial computations. Additionally, Shashank Agnihotri would like to thank Dr. Bin Zhao for his help in translating [75].

References

1. Agnihotri, S., Gandikota, K.V., Grabinski, J., Chandramouli, P., Keuper, M.: On the unreasonable vulnerability of transformers for image restoration-and an easy fix. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3707–3717 (2023)
2. Agnihotri, S., Grabinski, J., Keuper, J., Keuper, M.: Beware of aliases–signal preservation is crucial for robust image restoration. arXiv preprint arXiv:2406.07435 (2024)
3. Agnihotri, S., Jung, S., Keuper, M.: CosPGD: A unified white-box adversarial attack for pixel-wise prediction tasks. In: International Conference on Machine Learning (2024)
4. Aitken, A., Ledig, C., Theis, L., Caballero, J., Wang, Z., Shi, W.: Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize. arXiv preprint arXiv:1707.02937 (2017)
5. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
6. Badki, A., Troccoli, A., Kim, K., Kautz, J., Sen, P., Gallo, O.: Bi3D: Stereo depth estimation via binary classifications. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
7. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE transactions on pattern analysis and machine intelligence **39**(12), 2481–2495 (2017)
8. Baranchuk, D., Rubachev, I., Voynov, A., Khrulkov, V., Babenko, A.: Label-efficient semantic segmentation with diffusion models (2021)
9. Brigham, E.O., Morrow, R.: The fast fourier transform. IEEE spectrum **4**(12), 63–70 (1967)
10. Brunton, S.L., Kutz, J.N.: Data-driven science and engineering: Machine learning, dynamical systems, and control. Cambridge University Press (2022)
11. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European conference on computer vision. pp. 213–229. Springer (2020)
12. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 ieee symposium on security and privacy (sp). pp. 39–57. IEEE (2017)
13. Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., Joulin, A.: Unsupervised learning of visual features by contrasting cluster assignments. Advances in neural information processing systems **33**, 9912–9924 (2020)
14. Chandrasegaran, K., Tran, N.T., Cheung, N.M.: A closer look at fourier spectrum discrepancies for cnn-generated images detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7200–7209 (June 2021)
15. Chang, J.R., Chen, Y.S.: Pyramid stereo matching network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5410–5418 (2018)
16. Chen, L., Chu, X., Zhang, X., Sun, J.: Simple baselines for image restoration. In: European Conference on Computer Vision. pp. 17–33. Springer (2022)
17. Ding, X., Zhang, X., Han, J., Ding, G.: Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11963–11975 (2022)

18. Dosovitskiy, A., Brox, T.: Generating images with perceptual similarity metrics based on deep networks. Advances in neural information processing systems **29** (2016)
19. Dosovitskiy, A., Brox, T.: Inverting visual representations with convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4829–4837 (2016)
20. Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., Brox, T.: Flownet: Learning optical flow with convolutional networks. In: Proceedings of the IEEE international conference on computer vision. pp. 2758–2766 (2015)
21. Dosovitskiy, A., Springenberg, J.T., Tatarchenko, M., Brox, T.: Learning to generate chairs, tables and cars with convolutional networks. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**(4), 692–705 (2017)
22. Dumoulin, V., Visin, F.: A guide to convolution arithmetic for deep learning. arXiv preprint arXiv:1603.07285 (2016)
23. Durall, R., Keuper, M., Keuper, J.: Watch your up-convolution: Cnn based generative deep neural networks are failing to reproduce spectral distributions. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 7890–7899 (2020)
24. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html (2012)
25. Forsyth, D.A., Ponce, J.: Computer vision: a modern approach. Prentice Hall professional technical reference (2002)
26. Gal, R., Hochberg, D.C., Bermano, A., Cohen-Or, D.: SWAGAN: A style-based wavelet-driven generative model. ACM Transactions on Graphics (TOG) **40**(4), 1–11 (2021)
27. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks. Communications of the ACM **63**(11), 139–144 (2020)
28. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
29. Grabinski, J., Jung, S., Keuper, J., Keuper, M.: Frequencylowcut pooling-plug and play against catastrophic overfitting. In: European Conference on Computer Vision. pp. 36–57. Springer (2022)
30. Grabinski, J., Keuper, J., Keuper, M.: Aliasing and adversarial robust generalization of cnns. Machine Learning pp. 1–27 (2022)
31. Grabinski, J., Keuper, J., Keuper, M.: Aliasing coincides with cnns vulnerability towards adversarial attacks. In: The AAAI-22 Workshop on Adversarial Machine Learning and Beyond. pp. 1–5 (2022)
32. Grabinski, J., Keuper, J., Keuper, M.: Fix your downsampling asap! be natively more robust via aliasing and spectral artifact free pooling (2023)
33. Grabinski, J., Keuper, J., Keuper, M.: As large as it gets – studying infinitely large convolutions via neural implicit frequency filters. Transactions on Machine Learning Research (2024), https://openreview.net/forum?id=xRy1YRcHWj, featured Certification
34. Gu, J., Zhao, H., Tresp, V., Torr, P.H.: Segpgd: An effective and efficient adversarial attack for evaluating and boosting segmentation robustness. In: European Conference on Computer Vision. pp. 308–325. Springer (2022)
35. Guo, M.H., Lu, C.Z., Hou, Q., Liu, Z., Cheng, M.M., Hu, S.M.: Segnext: Rethinking convolutional attention design for semantic segmentation. Advances in Neural Information Processing Systems **35**, 1140–1156 (2022)
36. Hariharan, B., Arbeláez, P., Bourdev, L., Maji, S., Malik, J.: Semantic contours from inverse detectors. In: 2011 international conference on computer vision. pp. 991–998. IEEE (2011)

37. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
38. He, Y., Yu, N., Keuper, M., Fritz, M.: Beyond the spectrum: Detecting deepfakes via resynthesis. In: Zhou, Z.H. (ed.) Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21. pp. 2534–2541. International Joint Conferences on Artificial Intelligence Organization (8 2021), main Track
39. Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. Proceedings of the International Conference on Learning Representations (2019)
40. Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415 (2016)
41. Hoffmann, J., Agnihotri, S., Saikia, T., Brox, T.: Towards improving robustness of compressed cnns. In: ICML Workshop on Uncertainty and Robustness in Deep Learning (UDL) (2021)
42. Hossain, M.T., Teng, S.W., Lu, G., Rahman, M.A., Sohel, F.: Anti-aliasing deep image classifiers using novel depth adaptive blurring and activation function. Neurocomputing **536**, 164–174 (2023)
43. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: Flownet 2.0: Evolution of optical flow estimation with deep networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2462–2470 (2017)
44. Jung, S., Keuper, M.: Spectral distribution aware image generation. In: Proceedings of the AAAI conference on artificial intelligence. vol. 35, pp. 1734–1742 (2021)
45. Jung, S., Lukasik, J., Keuper, M.: Neural architecture design and robustness: A dataset. In: Eleventh International Conference on Learning Representations. OpenReview.net (2023)
46. Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., Aila, T.: Alias-free generative adversarial networks. Advances in Neural Information Processing Systems **34**, 852–863 (2021)
47. Khayatkhoei, M., Elgammal, A.: Spatial frequency bias in convolutional generative adversarial networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 7152–7159 (2022)
48. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25** (2012)
49. Kurakin, A., Goodfellow, I.J., Bengio, S.: Adversarial machine learning at scale. In: International Conference on Learning Representations (2017), https://openreview.net/forum?id=BJm4T4Kgx
50. Li, Z., Liu, X., Drenkow, N., Ding, A., Creighton, F.X., Taylor, R.H., Unberath, M.: Revisiting stereo depth estimation from a sequence-to-sequence perspective with transformers. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 6197–6206 (2021)
51. Liu, S., Chen, T., Chen, X., Chen, X., Xiao, Q., Wu, B., Kärkkäinen, T., Pechenizkiy, M., Mocanu, D., Wang, Z.: More convnets in the 2020s: Scaling up kernels beyond 51x51 using sparsity. arXiv preprint arXiv:2207.03620 (2022)
52. Liu, S., Deng, W.: Very deep convolutional neural network based image classification using small training sample size. In: 2015 3rd IAPR Asian conference on pattern recognition (ACPR). pp. 730–734. IEEE (2015)
53. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2021)

54. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11976–11986 (2022)
55. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3431–3440 (2015)
56. Loshchilov, I., Hutter, F.: SGDR: Stochastic gradient descent with warm restarts. In: International Conference on Learning Representations (2017), https://openreview.net/forum?id=Skq89Scxx
57. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=Bkg6RiCqY7
58. Maiya, S.R., Ehrlich, M., Agarwal, V., Lim, S.N., Goldstein, T., Shrivastava, A.: A frequency perspective of adversarial robustness (2021)
59. Mathew, A., Patra, A., Mathew, J.: Monocular depth estimators: Vulnerabilities and attacks. ArXiv **abs/2005.14302** (2020)
60. Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., Brox, T.: A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4040–4048 (2016)
61. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2574–2582 (2016)
62. Mosleh, A., Langlois, J.M.P., Green, P.: Image deconvolution ringing artifact detection and removal via psf frequency analysis. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) Computer Vision – ECCV 2014. pp. 247–262. Springer International Publishing, Cham (2014)
63. Nah, S., Kim, T.H., Lee, K.M.: Deep multi-scale convolutional neural network for dynamic scene deblurring. In: CVPR (July 2017)
64. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: Proceedings of the IEEE international conference on computer vision. pp. 1520–1528 (2015)
65. Odena, A., Dumoulin, V., Olah, C.: Deconvolution and checkerboard artifacts. Distill **1**(10), e3 (2016)
66. Peng, C., Zhang, X., Yu, G., Luo, G., Sun, J.: Large kernel matters – improve semantic segmentation by global convolutional network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017)
67. Pervin, M., Tao, L., Huq, A., He, Z., Huo, L., et al.: Adversarial attack driven data augmentation for accurate and robust medical image segmentation. arXiv preprint arXiv:2105.12106 (2021)
68. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
69. Radosavovic, I., Kosaraju, R.P., Girshick, R., He, K., Dollar, P.: Designing network design spaces. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)
70. Ronneberger, O., Fischer, P., Brox, T.: 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. pp. 234–241. Springer (2015)
71. Scheurer, E., Schmalfuss, J., Lis, A., Bruhn, A.: Detection defenses: An empty promise against adversarial patch attacks on optical flow. arXiv preprint arXiv:2310.17403 (2023)

72. Schmalfuss, J., Mehl, L., Bruhn, A.: Attacking motion estimation with adversarial snow. arXiv preprint arXiv:2210.11242 (2022)
73. Schmalfuss, J., Mehl, L., Bruhn, A.: Distracting downpour: Adversarial weather attacks for motion estimation (2023)
74. Schmalfuss, J., Scholze, P., Bruhn, A.: A perturbation-constrained adversarial attack for evaluating the robustness of optical flow (2022)
75. segcv: segcv/pspnet. https://github.com/segcv/PSPNet/blob/master/Train.md (2021)
76. Shannon, C.E.: Communication in the presence of noise. Proceedings of the IRE **37**(1), 10–21 (1949)
77. Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D., Wang, Z.: Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network (2016)
78. Si-Yao, L., Ren, D., Yin, Q.: Understanding kernel size in blind deconvolution. In: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 2068–2076. IEEE (2019)
79. Smith III, J.O.: Physical audio signal processing: For virtual musical instruments and audio effects. (No Title) (2010)
80. Sommerhoff, H., Agnihotri, S., Saleh, M., Moeller, M., Keuper, M., Kolb, A.: Differentiable sensor layouts for end-to-end learning of task-specific camera parameters. arXiv preprint arXiv:2304.14736 (2023)
81. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
82. Tan, M., Le, Q.: Efficientnetv2: Smaller models and faster training. In: International conference on machine learning. pp. 10096–10106. PMLR (2021)
83. Teed, Z., Deng, J.: Raft: Recurrent all-pairs field transforms for optical flow. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16. pp. 402–419. Springer (2020)
84. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: International conference on machine learning. pp. 10347–10357. PMLR (2021)
85. Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., Madry, A.: Robustness may be at odds with accuracy. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=SyxAb30cY7
86. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing **13**(4), 600–612 (2004)
87. Xu, L., Ren, J.S., Liu, C., Jia, J.: Deep convolutional neural network for image deconvolution. Advances in neural information processing systems **27** (2014)
88. Yamanaka, K., Matsumoto, R., Takahashi, K., Fujii, T.: Adversarial patch attacks on monocular depth estimation networks. IEEE Access **8**, 179094–179104 (2020)
89. Zamir, S.W., Arora, A., Khan, S., Hayat, M., Khan, F.S., Yang, M.H.: Restormer: Efficient transformer for high-resolution image restoration. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 5728–5739 (2022)
90. Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., Jordan, M.: Theoretically principled trade-off between robustness and accuracy. In: ICML (2019)
91. Zhang, R.: Making convolutional networks shift-invariant again. In: ICML (2019)
92. Zhao, H.: semseg. https://github.com/hszhao/semseg (2019)
93. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2881–2890 (2017)

94. Zhao, H., Zhang, Y., Liu, S., Shi, J., Loy, C.C., Lin, D., Jia, J.: PSANet: Point-wise spatial attention network for scene parsing. In: ECCV (2018)
95. Zou, X., Xiao, F., Yu, Z., Lee, Y.J.: Delving deeper into anti-aliasing in convnets. In: BMVC (2020)