Partial Convolution for Padding, Inpainting, and Image Synthesis

Guilin Liu*, Aysegul Dundar*, Kevin J. Shih, Ting-Chun Wang, Fitsum A. Reda, Karan Sapra, Zhiding Yu, Xiaodong Yang, Andrew Tao, Bryan Catanzaro

Abstract—Partial convolution weights convolutions with binary masks and renormalizes on valid pixels. It was originally proposed for image inpainting task because a corrupted image processed by a standard convolutional often leads to artifacts. Therefore, binary masks are constructed that define the valid and corrupted pixels, so that partial convolution results are only calculated based on valid pixels. It has been also used for conditional image synthesis task, so that when a scene is generated, convolution results of an instance depend only on the feature values that belong to the same instance. One of the unexplored applications for partial convolution is padding which is a critical component of modern convolutional networks. Common padding schemes make strong assumptions about how the padded data should be extrapolated. We show that these padding schemes impair model accuracy, whereas partial convolution based padding provides consistent improvements across a range of tasks. In this paper, we review partial convolution applications under one framework. We conduct a comprehensive study of the partial convolution based padding on a variety of computer vision tasks, including image classification, 3D-convolution-based action recognition, and semantic segmentation. Our results suggest that partial convolution-based padding shows promising improvements over strong baselines.

Index Terms—Partial Convolution, Padding, Image Inpainting, Image Synthesis, Object Classification, Semantic Segmentation.

1 Introduction

Convolutional Neural Networks (CNNs) have achieved excellent results on various computer vision tasks such as object recognition [22], [56], detection [38], [51], segmentation [6], [82], [94], video action recognition [19], [79], video interpolation [29], [49], prediction [35], [54], image and video inpainting [47], [77], and synthesis [3], [13], [31], [41], [69], to name a few. These networks rely on convolutions to extract useful features from images that can be used for downstream tasks. However, there are many scenarios when parts of the input data are missing, invalid, or even corrupted. The following work demonstrates various scenarios in which this is the case, and how an adaptive re-weighting of the filter can drastically improve results.

An obvious case in which filters convolve over missing pixel values is the image inpainting task, the task of filling holes in an image with plausible imagery. It has various applications such as inferring occluded image content in novel-view synthesis, and image editing tasks such as context-aware content removal. CNNs have been used for recent image inpainting efforts, employing convolutional filters on images where the removed content has been replaced with a fixed value (typically the mean value). The placeholder hole values are then treated no differently from truly valid image content, likely confusing the neural networks and leading to artifacts such as color discrepancy or blurriness. Post-processing is often required to ameliorate these issues [24], [84].

Invalid pixels also naturally occur in cases of data padding. Parts of convolutional filters often need to extend beyond the spatial dimensions of their inputs, such as when we want the output of a 3×3 or larger filter to maintain the original input dimensions. In order for the output of a convolution to be defined when parts of the filters extend beyond the original input dimensions, we pad the borders of our input data with some assumed reasonable values. For small input dimensions, padding has a surprisingly large effect. For instance, given a 6×6 input feature map with 1 pixel padding on each side, $(8 \times 8 - 6 \times 6)/(8 \times 8) = 43.75\%$ of the data will be from padding. Several commonly used padding approaches introduce assumptions about the padded regions, leading to unwanted biases in trained models. For example, the most common padding scheme is zero-padding, in which the input data is extended with zeros along the edges. While zero padding is easy to implement and computationally efficient, it can introduce sharp spikes or drop in filter response values at the edges of resulting feature maps, thereby affecting the model's final prediction. Other padding approaches such as reflection padding and repetition padding differ from zero-padding in that they attempt to extrapolate the missing data beyond the input dimensions via simple heuristics. Reflection padding reflects pixel values along the border axis whereas replication padding simply extends the border-most pixel. While these heuristics result in a smoother transition from input data to padded data than zero padding would, reflecting or repeating the border regions for many layers also tend to introduce unrealistic data patterns. In the following work, we argue that explicitly differentiating the valid input pixels from the corrupted/invalid input pixels in convolution operation can

*Joint first authors, contributed equally.

- G. Liu, KJ. Shih, TC. Wang, K. Sapra, Z. Yu, X. Yang, A. Tao, B. Catanzaro are with NVIDIA, CA, USA. E-mail: {gliu, kshih, tingchunw, freda, ksapra, zhidingy, xiaodongy, atao, bcatanzaro}@nvidia.com
- A. Dundar is with Department of Computer Science, Bilkent University, Ankara, Turkey. (e-mail: adundar@cs.bilkent.edu.tr)
- FA. Reda is with Google, CA, USA.

help improve the feature extraction quality and robustness of the models.

Lastly, traditional convolution operation may have difficulty generating realistic images in conditional image synthesis applications. Image synthesis refers to the task of generating photo-realistic images, where a prevalent subcategory known as conditional image synthesis outputs images that are conditioned on various input data. The input to the image synthesis network may be a semantic segmentation or instance segmentation map, and synthesized images are expected to have high fidelity to the underlying input information. Convolutional operations independent of class and instance boundaries will result in an undesirable blending of conditional information across instance boundaries, generating synthesized results with blurred boundaries.

Partial convolutions, where the convolution is masked and renormalized to be conditioned on only valid pixels can improve the results in different applications of convolutional neural networks. In all aforementioned applications, we see instances where input pixels may be corrupted, invalid, and uncorrelated. The following work argues that a convolutional operation that is capable of explicitly differentiating between valid and invalid input pixels will perform more robustly on the incomplete input data.

In this work, we provide a comprehensive study of partial convolution applications in padding, image inpainting, and image synthesis under one framework. Partial convolutions for image inpainting and synthesis have been studied before in [37] and [14], respectively. The following work provides more discussions and results of how the application of partial convolution differ for these two tasks. Furthermore, little attention has been given to padding in the literature. We conduct a thorough empirical comparison of partial convolution-based padding, comparing with other padding techniques. We show that partial convolution padding performs favorably compared to existing padding techniques with no additional parameters on a variety of diverse single label and dense prediction tasks, including image classification, video action recognition, and semantic segmentation.

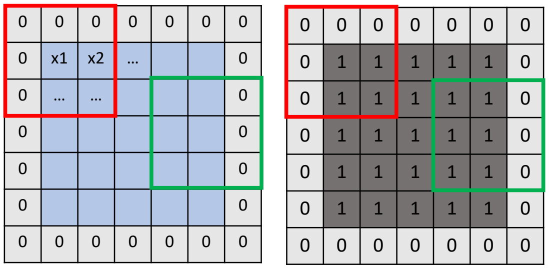
2 Related Work

Reweighted Convolution. Reweighted convolution has been explored to capture more powerful representations by weighting up the properties of an image that are prominent for a given task, while diminishing the affect of the other parts in the image. There have been many works that learn to weight the convolution activations based on attention mechanisms [20], [70], [80], [90]. These mechanisms operate on feature maps to capture the spatial locations that are related to each other while making a decision. Harley et al. [20] use soft attention masks to reweight the convolution results for semantic segmentation. Uhrig et al. [64] propose sparsity invariant CNNs with reweighted convolution and max pooling based mask updating mechanism for depth completion. Inpainting methods like [50], [64] take the hole mask into consideration for reweighting the convolution results. PixelCNN [66] designs the reweighted convolution such that the next pixel generation only depends on previous generated pixels. Partial convolution is a type of reweighted convolution where binary masks defined for each task are used to weight the convolutions and renormalize on only valid pixels.

Padding. Researchers have improved the accuracy of CNN models through careful research of almost all aspects of the model and optimization process, including different variants of SGD [34], [63], non-linearities [21], [44], normalization layers [26], etc. However, little attention has been paid to improving padding beyond the standard zero padding, reflection padding, and repetition padding schemes. One proposed improvement came from Innamorati et al. [25], which augment networks by introducing four separate filters dedicated to corners and another four to extract features in the borders. The resulting models from this scheme are not directly comparable to traditionally padded models, because they require dedicated filters and increase the number of parameters by eight times. Cheng et al. [9] propose a special image projection to handle undefined boundaries resulting from projecting a 360° view image onto a 2D image. Images are first projected onto a cube and a final image is formed by concatenating cube faces. As such, large undefined borders can be eliminated. Our proposed partial convolution based padding scheme is orthogonal to all of these ideas. Furthermore, it does not increase the parameter count; instead, it uses a single filter with a simple yet effective re-weighting mechanism at boundary pixels where padded regions are interpreted as missing data.

Image Inpainting. Recent deep learning based methods for image inpainting and outpainting [24], [36], [47], [62], [78] initialize the holes (regions with missing data) with constant placeholder values, and treat the newly initialized hole region and original non-hole region equally. Then a CNN is used to re-generate a new image, conditioned on this initialized image. The results usually suffer from dependence on the initial hole values, with lack of texture in hole regions, obvious color contrasts or artificial edge responses. Many of them require post-processing, like a refinement network [57], [84], Poisson blending [24] or other post-optimization techniques [78]. Some other methods [64], [65], [81] ignore the hole placeholder values in the image inpainting or data completion tasks; none of them explicitly handle the missing data. Partial convolution is previously proposed by [37] and become very popular in this domain. Later, Yu et al. [85] extended partial convolution to gated convolution by changing the binary mask to fractional mask with values in [0, 1] and gradually updating the mask values. Xie et al. [75] extended the forward mask updating in partial convolution to bi-directional mask updating. In this paper, we review partial convolution in image inpainting to show partial convolution can achieve padding, inpainting, and synthesis under one framework.

Conditional Image Synthesis can vary based on different type of inputs to be conditioned upon. For example, natural and synthetic images, keypoints, skeletons, and pose [2], [15], [23], [40], [93]. Recently, CNNs powered with GAN trainings [17] have achieved synthesizing realistic images based on semantic maps [27], [69]. In addition to semantic maps, boundary maps are also input to CNNs to synthesize instances with clear boundaries [46], [69]. In conditional image synthesis tasks, the results are improved by research



(a) $\mathbf{X}^{p0}$ (b) mask ($\mathbf{M}$)

Fig. 1: Visualization of $\mathbf{M}$ (mask) generation for partial-convolution based padding. $\mathbf{X}^{p0}$ is zero-padded input, $\mathbf{X}$. $\mathbf{M}_{(i,j)}$ can be constructed by first starting with a single channel binary $\mathbf{1}_{(i,j)}$ which denotes a 2D matrix having same height and width as $\mathbf{X}_{(i,j)}$ (grey region), and then zero-padding $\mathbf{1}_{(i,j)}$.

on normalization layers [43], [46], multi-scale discriminators [69], and architectural changes in the generator [22]. Attention mechanisms [86], [90] and adaptive convolutional kernels that are aware of the distinct semantic labels [39] have been proposed to weight the convolutions in image synthesis task. In this paper, we will review how partial convolution can improve image synthesis task conditioned on panoptic maps which combine semantic and instance information.

Partial Convolution was proposed to handle holes in image inpainting task [37]. Since then it has been used in other image inpainting works [68], [88], video inpainting [12], face inpainting [42], depth inpainting [18] and speech inpainting [33]. Furthermore, it has been used for new view synthesis [45], [55], depth denoising [58], medical data processing for disease diagnosis [5], [48], [67], [72], [76], precipitation prediction [28], remote sensing data cleaning [7], [60], recovering historical artworks [8], [87] etc. In this paper, we review the image inpainting and synthesis applications of partial convolution and provide a comprehensive study of partial convolution based padding. We expect partial convolution to be used in various tasks when input data has missing points.

3 Partial Convolution and its Applications

In this section, we review partial convolution and how it applies to different applications such as partial padding, image inpainting and image synthesis conditioned on panoptic maps. All applications share the same fundamental that output of convolution operation should not be affected by invalid data. The invalid data may come from padding pixels, missing data (holes) for image inpainting, or in the case of image synthesis, pixels that belong to separate instances. The invalid pixels are defined by masks which are used in partial convolution setting. First, we define partial convolution, then show how we reconstruct the masks for each application.

Partial Convolution. Partial convolution is proposed to handle invalid input data, such as images with holes. Let $\mathbf{X}_{(i,j)}$ be the feature values (pixel values) for the current convolution (sliding) window at the position (i,j) and $\mathbf{M}_{(i,j)}$ be the corresponding binary mask. We will define how the definition of $\mathbf{M}_{(i,j)}$ differs for each application, but for simplicity, lets take image inpainting example. $\mathbf{M}_{(i,j)}$ is constructed with the hole region being 0 and non-hole region being 1. The partial convolution at every location is defined as:

$$x'_{(i,j)} = \begin{cases} \mathbf{W}^T(\mathbf{X}_{(i,j)} \odot \mathbf{M}_{(i,j)}) r_{(i,j)} + b, & ||\mathbf{M}_{(i,j)}||_1 > 0 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where

$$r_{(i,j)} = \frac{||\mathbf{1}_{(i,j)}||_1}{||\mathbf{M}_{(i,j)}||_1}, \tag{2}$$

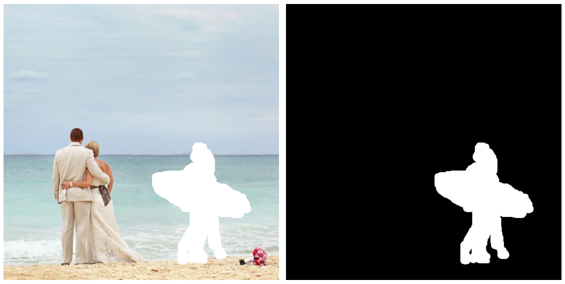
$\odot$ denotes element-wise multiplication, $\mathbf{1}_{(i,j)}$ is the all-one vector with the same shape as $\mathbf{M}_{(i,j)}$ and $\mathbf{W}$ is the filter weight matrix. We compute $x'_{(i,j)} = x'_{(i,j)} + b$ to account for an additional bias term (when $||\mathbf{M}_{(i,j)}||_1 > 0$). As can be seen, output values depend only on the unmasked inputs. The scaling factor $||\mathbf{1}_{(i,j)}||_1/||(\mathbf{M}_{(i,j)}||_1$ applies appropriate scaling to adjust for the varying amount of valid (unmasked) inputs.

Partial Convolution-based Padding. Partial convolution can be extended to the to-be-padded regions by treating them as yet another zero-filled hole in the input image. $\mathbf{M}_{(i,j)}$ can be constructed by first starting with a single channel binary $\mathbf{1}_{(i,j)}$ which denotes a 2D matrix having same height and width as $\mathbf{X}_{(i,j)}$, and then zero-padding $\mathbf{1}_{(i,j)}$. The visualization of the mask construction can be found in Figure 1. With the zero-padded mask, $\mathbf{M}_{(i,j)}$, and zero-padded $\mathbf{X}_{(i,j)}$, convolution is applied as given in Eq. 1

Based on Eq. 1, we can see that partial convolution-based padding is corresponding to the widely used zero padding with the scaling factor given in Eq. 2. This is an adaptive scaling based on the number of invalid pixels to adjust the varying amount of valid inputs. Note that it is not a constant scaling factor which would have no effect since network weights are learnable and can learn to compensate for the constant scaling. The change may seem subtle, but in the experiment section, we will show how partial-convolution based padding scheme improves network results on multiple applications such as object recognition, segmentation, and action recognition.

Furthermore, in some cases, we may require padding larger than the kernel size. This occurs when padding is used to resize images such that images of differing sizes can fit in the same batch tensor. Re-sizing by padding is typically preferred because image scaling may introduce visual distortions. Convolutions at such borders may not have valid data because the kernel size could be smaller than the padding width. For such cases, we include the mask updating step. Specifically, for the very first convolutional layer, the input mask $\mathbf{M}_{1st-layer}$ will be the padded result of $\mathbf{1}$ ($\mathbf{M}_{1st-layer} = \mathbf{1}^{p0}$). In the next layers, if the convolution was able to condition its output on at least one valid input value in the padding region, then we mark that location to be valid. This is expressed as:

$$m'_{(i,j)} = \begin{cases} 1, & \text{if } ||\mathbf{M}_{(i,j)}||_1 > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$



(a) input image (b) mask ($\mathbf{M}$)

Fig. 2: Visualization of $\mathbf{M}$ (mask) generation for partial-convolution inpainting. Breaking with tradition, the black pixels correspond to a binary mask value of 1 (valid), while the to-be-ignored hole values 0 are visualized in white.

This will produce an output mask $\mathbf{M}'_{1st-layer}$ using the rule in Eq. 3. The mask input for the next layer will be the padded result of $\mathbf{M}'_{1st-layer}$, namely $\mathbf{M}_{2nd-layer} = \mathbf{M}'^{\,p0}_{1st-layer}$, and so on.

Partial Convolution-based Image Inpainting. Image painting is the task of filling the missing data in images. The missing data appears as holes, and conditioning the output on the hole values ultimately results in various types of visual artifacts. In this task, $\mathbf{M}_{(i,j)}$ is constructed by setting holes as 0, and the other pixels as 1 as demonstrated in Figure 2.

The mask update given in Eq. 3 is especially crucial for this task, as holes are usually significantly greater than convolution kernels. Therefore, after each partial convolution operation, the mask is updated and with sufficient successive applications of the partial convolution layer, any mask will eventually be all ones, if the input contained any valid pixels.

Partial Convolution-based Image Synthesis. Conditional image synthesis algorithms can be guided by semantic and panoptic maps. A panoptic map combines instance and segmentation information, an example can be seen in Figure 3. We use partial convolutions to re-weight convolutions based on these panoptic maps. This way, convolution results of an instance or a semantic class depend only on the feature values that belong to the same instance or class. We follow the same convolution equation as given in Eq. 1, but we reconstruct the mask, $\mathbf{M}$, in a different way than in other applications. Let $\mathbf{P}$ be the panoptic map values for the current convolution (sliding) window, and $\mathbf{M}$ is the corresponding binary mask. $\mathbf{M}$ defines which pixels will contribute to the output of the convolution operation based on the panoptic maps. The pixel coordinates which share the same identity with the center pixel in the panoptic map are assigned 1 in the mask, while the others are assigned 0. This is expressed as:

$$m_{(i,j)} = \begin{cases} 1, & \text{if } \mathbf{P}_{(\mathbf{i},\mathbf{j})} = \mathbf{P}_{(\mathbf{center},\mathbf{center})} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

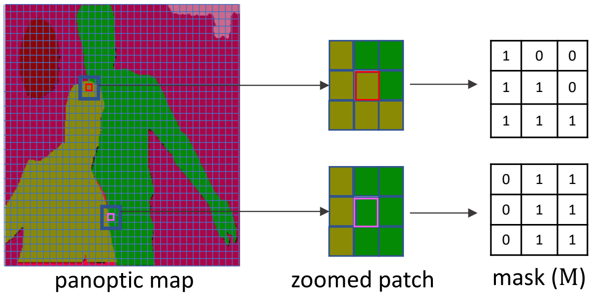


Fig. 3: Visualization of $\mathbf{M}$ (mask) generation for partial-convolution based image synthesis. The mask is constructed based on panoptic maps (colorized for visualization). It operates in a sliding-window fashion to generate a binary mask value at the center of each window. The pixels that share the same identity with that of the center of the window are assigned 1, otherwise 0.

This can be implemented by first subtracting the center pixel from the patch and clipping the absolute value to [0, 1], then subtracting the clipped output from 1 to invert the zeros and ones. Figure 3 depicts the construction of the mask $\mathbf{M}$.

4 Experiments

In this section, we present our experiments of partial convolution in three main subsections, partial convolution-based padding, partial convolution-based image inpainting, and partial convolution-based image synthesis. Convolution-based image inpainting and image synthesis are previously explored in [37] and [14], respectively. We provide additional qualitative results for these two applications. On the other hand, little attention has been given to padding in the literature. In this paper, we provide an extensive study on the effects of different padding schemes on various tasks starting with simple task of learning a Gaussian filtering, to image classification, segmentation, and video classification. Note that, we do not aim at achieving state-of-the-art results for each task but instead we show improvements of partial convolution in controlled studies with strong baselines.

4.1 Partial Convolution-based Padding

Implementation Details. We implement the partial convolution based padding in PyTorch extension of existing convolution modules. We implement the mask of ones ($\mathbf{1}$) as a single-channel feature with the same batch size, height and width as the input tensor $\mathbf{X}$. The 1-padded version of $\mathbf{1}$ ($||\mathbf{1}_{p1}||_1$) is directly set to be $k_h \times k_w$, where k_h and k_w are the height and width of the kernel. $||\mathbf{1}_{p0}||_1$ is implemented by calling the convolution operation once with all the weights being 1, bias being 0 and original target padding size. The result of $\frac{||\mathbf{1}_{p1}||_1}{||\mathbf{1}_{p0}||_1}$ only needs to be computed at the first time and the result can then be cached for future iterations as long as the input size does not change. Thus, the execution time starting from the second iteration will be lower than the first iteration. In Table 1, we show the comparison of GPU execution time between networks with various padding and partial convolution-based padding powered networks for both the first iteration and after iterations. It can be seen

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2022.3209702

	runtime (ms)	relative
VGG16BN_zero	3.46	100%
VGG16BN_ref	3.65	105%
VGG16BN_rep	3.64	105%
VGG16BN_explicit	31.23	903%
VGG16BN_partial (1st)	4.42	128%
VGG16BN_partial (2nd - nth)	4.18	121%
RN50_zero	3.43	100%
RN50_ref	3.55	103%
RN50_rep	3.53	103%
RN50_explicit	16.85	491%
RN50_partial (1st)	5.84	170%
RN50_partial (2nd - nth)	4.29	125%

TABLE 1: Inference time (ms) comparison between zero and partial convolution-based padding networks using a 224×224 image as input, measured on a single NVIDIA V100 GPU. *zero, *ref and *rep, and *partial indicate the corresponding models with zero, reflection, repetition, explicit [25] and partial convolution-based paddings. This is based on the raw PyTorch implementation with no custom CUDA code. 1st, 2nd and nth represent the first, second, and nth iterations respectively.

that starting from the second iteration partial convolution-based padding powered VGG16 (batch normalization layer version) and ResNet50 (RN50) cost about 25% more time to do the inference on a 224×224 input image with a single NVIDIA V100 GPU. Note that this implementation is written in pure PyTorch and does not require custom CUDA code. If implemented in CUDA, the extra cost would be negligible as we would only need to re-weight the convolution results at the border. In Table 1, we also measure explicit padding [25] method which augments networks by introducing four separate filters dedicated to corners and another four dedicated to borders of images and feature maps. Due to its additional parameters, the runtime increases up to 9 times of zero padding for VGG16 and 5 times for RN50.

4.1.1 Learning Gaussian Filtering

Following [25], we first run experiments to observe partial convolution-based padding's effect on a simplified task of learning how to perform a Gaussian blur of a fixed size. The neural network is asked to learn the effect of a Gaussian Filter of size 13×13 from ImageNet dataset. Following [25], we generate the ground truths by applying the Gaussian Filter over $(128 + 12) \times (128 + 12)$ images which we later crop to size of 128×128 as targets and the crops of 128×128 original images as inputs. We train variant of U-NET [52] architecture on this task with various padding schemes. The U-NET consists of 4 blocks of convolution and leaky relu layers where each convolution has 5×5 kernels applied with a stride of 2 and padding size of 2. Number of filters starts with 32 and increases by 2× at each block. It has a symmetric decoder block with decreasing number of filters. There are skip connections between 1st, 2nd, and 3rd blocks. We compare methods by means of the MSE metric, which is also used as the loss function during training. This task is a very simple one with the only challenge of network not seeing the overall image of $(128+12) \times (128+12)$ but only seeing the crop of it 128×128.

Table 2 presents the quantitative comparison on this task. Explicit padding [25] achieves the best result and partial convolution-based padding obtains the second best result, achieving the $\%94.7$ relative improvement of explicit padding. Explicit padding has more flexibility than partial convolution-based padding as it learns four separate filters for corners and another four for borders of images and feature maps. Therefore, not only it has the flexibility to learn the same behaviour of partial convolution-based padding but also it has more expressive power to also learn different weights for corners and borders. It also comes with additional parameters and slower training and inference speed. Therefore, we find that it is quite of a remarkable achievement the partial convolution-based padding obtains relative to the other padding schemes without introducing any additional parameters and no slow down if the kernels are implemented in CUDA. Furthermore, in the next section, we compare these five methods in image classification task and observe that explicit padding achieves the worst results due to its increased number of parameters. These results suggest that partial convolution-based padding is a good candidate for various tasks.

	MSE metric	relative percentage over best
U-NET_zero	0.000485	%81.8
U-NET_ref	0.000600	%66.1
U-NET_rep	0.000616	%64.4
U-NET_explicit	0.000397	%100
U-NET_partial	0.000419	%94.7

TABLE 2: Quantitative results on Gaussian filtering task. *zero, *ref and *rep, and *partial indicate the corresponding models with zero, reflection, repetition, explicit [25] and partial convolution-based paddings.

4.1.2 Image Classification

We conduct experiments with the VGG{16,19} [56], and ResNet{50,101,152} [22] models on the ImageNet classification task. We use the official training and validation set provided by the ImageNet dataset. We train VGG models with Batch Normalization layers, as they are known to be less sensitive to learning rates and initializations. We use the training scripts and model definitions from the corresponding official PyTorch examples.

For each model, we replace all the zero padding with partial convolution-based padding while keeping all other parameters the same. While the default PyTorch training script trains for 90 epochs, we use 100 in our experiments as in [73]. All models are trained with 8 NVIDIA V100 GPUs on DGX-1 workstations. The initial learning rate is set to be 0.1 and decreased by a factor of 10 at every 30 epochs. We train each network 5 times to account for model variance. To have a full comparison with all the existing padding schemes, we also run experiments for explicit padding [25], reflection padding, and replication padding with the same settings. For each training run, we select the checkpoint with the highest validation set accuracy to represent that run.

We evaluate the models using the top-1 classification accuracy based on the single center crop evaluation. Figure 4 shows the mean results of each run's "best" scenario among 5 runs for each model and each padding scheme. Note that the top-1 accuracy for all the baseline models from our training runs, closely matched those reported in official Py-

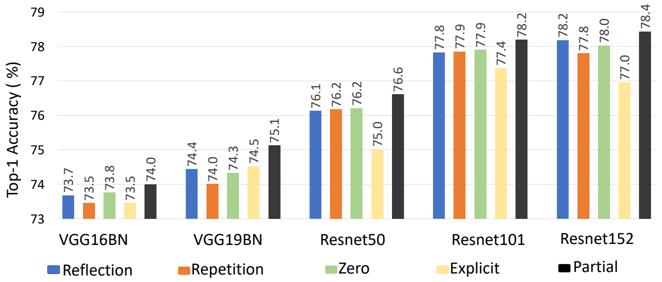


Fig. 4: Comparison of the ImageNet classification top-1 accuracy with center crop among zero padding (zero), reflection padding (reflection), repetition padding (repetition), the padding proposed in [25] (explicit) and our partial convolution-based padding (partial) on VGG [56] and ResNet [22] networks. VGG16BN and VGG19BN represent the VGG16 network and VGG19 network with batch normalization layers.

Torch documentation[1]. partial convolution-based padding (*_partial) provides better validation accuracy than other padding schemes in all architectures. VGG19 with partial convolution-based padding has the largest improvement with a 0.68% accuracy boost. In the ResNet family, ResNet50 model has the largest improvement (0.478%) compared with ResNet101 (0.36%) and ResNet152 (0.248%). We observe that the improvements are consistent across strong baselines. Furthermore, we find that reflection padding and replication padding lead to similar or worse accuracies compared with zero padding as shown in Figure 4. Overall, the padding in [25], which uses additional eight filters for borders and corners, has the worst accuracies. Weight-sharing property of CNNs is known to be an important regularizer and an important ingredient in the success of discriminative networks. Explicit padding partially removes this behaviour and that may be the reason of its degraded performance in image classification task. On the other hand, in image synthesis task of learning Gaussian Filtering, explicit padding achieves the best results. This is in-line with other works where adaptive filters are used for successful image generation [39]. Due to explicit padding's unsuccessful results on image classification task and its requirement for longer training (5× longer training time), and memory (8× more parameters), we omit this method in our other experiments which are already more computationally demanding. In segmentation and video classification tasks, we only compare zero, reflective, repetitive, and partial convolution-based paddings.

4.1.3 Semantic Segmentation

We further test partial convolution-based padding scheme in semantic segmentation task. The semantic segmentation task is to classify each pixel of an image with a semantic category. Most recent semantic segmentation networks use an encoder-decoder architecture [1], [52] to ensure the output dimensions match those of the input. It is common to employ an ImageNet pretrained network such as ResNet50 as the backbone for the encoder part, followed by a series of deconvolutions or upsampling layers for the decoder. Padding is essential in guaranteeing same input-output dimensions as the center of each filter would not be able to reach the edge pixels otherwise. Some networks also use dilated convolutions to achieve larger receptive fields without downsampling or increasing the number of filter parameters. Dilated convolutions require large number of padding pixels which increases the influence of padded values in the encoded features. We use a strong baseline as our segmentation model [94] and use DeepLabV3+ [6], one of the state-of-the-art semantic segmentation networks, which utilizes dilated convolutionsl. It uses a pretrained ImageNet classifier like ResNet as the encoder backbone. The encoder further includes a dilated-convolution-based *atrous spatial pyramid pooling* module (ASPP) to encode feature from multiple scales. Next, a decoder with skip links to the encoder features and a final upsampling is used to output the final predictions.

We run our segmentation experiments on the Cityscapes dataset [10] which contains 5000 images with pixel-level annotations. The default splits include 2975 images for the training set, 500 images for the validation set and 1525 images for the test set. It also contains $20,000$ coarsely annotated images. To keep our experiments simple and focused on the differences between regular and partial convolution-based padding, we do not employ external datasets for pre-training as is done to achieve state-of-the-art performance in works such as [4]. The Cityscapes dataset contains training data from 21 different cities. The default train-val split creates an 18/3 train/val split by cities. We create an additional second 18/3 split to experiment on as well. Motivated by Mapillary [4], we evaluate partial convolution-based padding using WideResNet38 [74] and ResNet50 encoder networks. We also use a data sampling strategy similar to [4] and use the 20k coarsely annotated images along with the finely annotated images. We train the segmentation network for 31K iterations with SGD with an initial learning rate of 0.03 and 0.1 for ResNet50 and WideResNet38, respectively and with a polynomial learning rate decay of 1.0. Our momentum and weight decay are set to 0.9 and 0.0001, respectively. Similar to other semantic segmentation works [6], [89], [91], we use the following augmentations during training: random scaling, horizontal flipping, gaussian blur, and color jitter. Our crop size is set to 896 for ResNet50 and 736 for WideResNet38. Lastly, to due to the large crop size, we use a batch size of 2 with synchronized batch normalization (for distributed training), similar to PSPNet [91].

Results. We compare and evaluate the segmentation models using mean intersection-over-union (mIoU) (%) across 19 classes defined by the Cityscapes dataset. The ASPP module includes a spatial pooling step that outputs a single 1-D feature vector. During training, this pooling procedure operates on square crops (1:1 aspect ratio). However, during full-image inference, we must account for the fact that the full images are of size 1024×2048 with an aspect ratio of 1:2. This means that the pooling module, trained to pool over 1:1 aspect ratios, must now pool over 1:2 aspect ratio input at test time. We resolve this by breaking down the full image into overlapping square tiles to feed into our model. We report two types of inference results:

[1] https://pytorch.org/docs/stable/torchvision/models.html

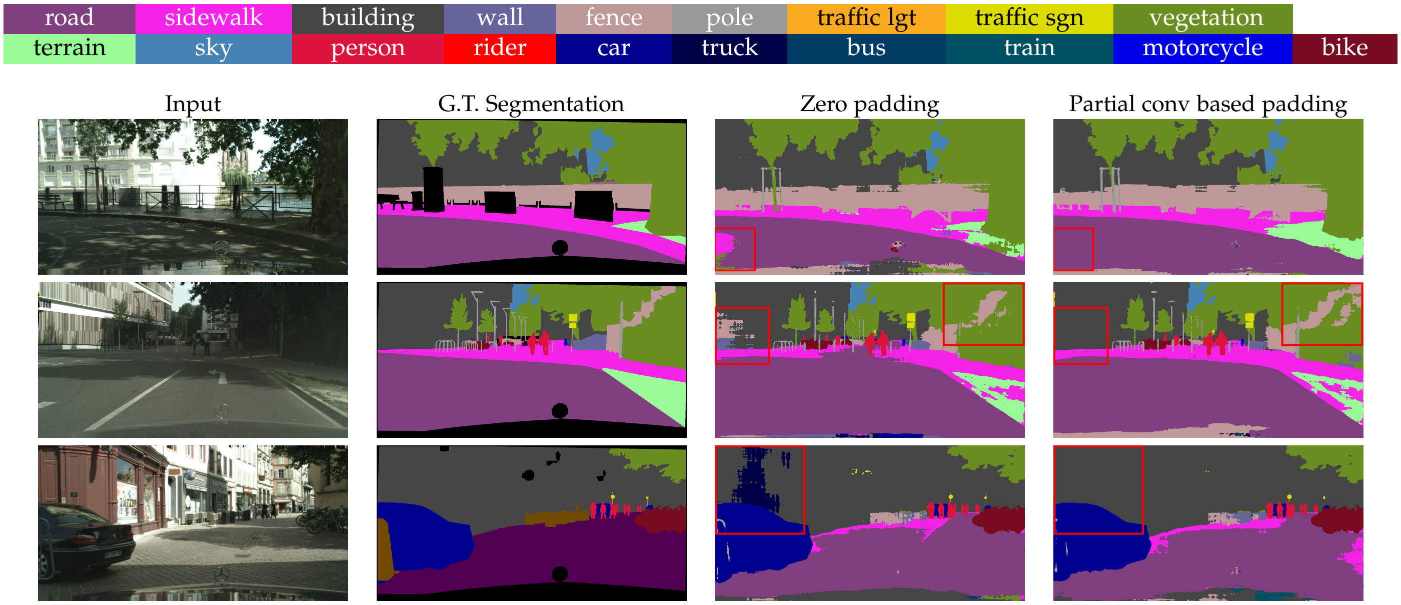


Fig. 5: Semantic segmentation results on Cityscapes dataset. From left to right: Input image, Ground truth segmentation, Zero padding prediction, Partial conv based padding prediction. We demonstrate that partial convolution-based padding method can remove border artifacts, thus resulting in a better prediction. Only zero and partial convolution-based padding results are visualized since these are the best two performing methods.

	default split			additional split		
	mean	diff	stdev	mean	diff	stdev
RN50_zero	78.08	-	0.06	76.58	-	0.42
RN50_ref	77.75	-0.33	0.38	76.63	0.04	0.31
RN50_rep	77.31	-0.78	0.13	76.20	-0.38	0.44
RN50_partial	**78.20**	0.12	0.19	**76.91**	0.33	0.15
WN38_zero	80.20	-	0.18	78.84	-	0.20
WN38_ref	79.37	-0.83	0.12	78.18	-0.66	0.06
WN38_rep	79.62	-0.59	0.20	78.15	-0.69	0.08
WN38_partial	**80.31**	0.11	0.19	**79.00**	0.16	0.11

TABLE 3: DeepLabV3+ mIoU(%) evaluation on Cityscapes dataset. *_zero, *_ref and *_rep, and *_partial indicate the corresponding model with zero, reflection, repetition, and partial convolution-based paddings. Models are trained from the scratch on the training set and evaluated on the validation set.

1) **regular:** directly feeding the 1024×2048 image into the network regardless of the aspect ratio issue.
2) **tile:** feeding patches after dividing the images into square tiles of size 1024×1024. A straightforward tile based evaluation divides the images into non-overlapping regions. There is also the overlapping version of tile based evaluation, but we do not discuss it in this paper for brevity. Tile based evaluation has been used by popular segmentation works [59], [91], [94] to obtain better evaluation performance.

In Table 3, we show results with regular inference in which our segmentation models using partial convolution-based padding with ResNet50 and WideResNet38 encoder backbones achieve better mIoU. ResNet50 encoder based segmentation model trained using partial convolution-based padding achieves 0.12% and 0.329% higher mIoU on the default and additional splits, respectively. We also observe similar performance gains with WideResNet38+, partial convolution-based padding outperforming its counterpart by 0.112% and 0.164% in the default and additional splits.

In Table 4 - first column, we report results with tile-based evaluation on the additional split. We show that the tile based evaluation achieves higher mIoU than the regular evaluation, and when partial convolution-based padding is used, the improvements are even more significant. One major concern for tile-based evaluation is that by subdividing the image, we significantly increase the number of image border pixels. This leads to an increase in the proportion of padded values that the network sees, as well as more cases of truncated image context. We observe that the importance of partial convolution-based padding even becomes more significant in this scenario. The model with partial convolution-based padding is around 0.37% better than the model with zero padding, the second best performing method. This is because our proposed model is more robust to the boundary issue, and thus is much less affected by the increase in border pixels. In Figure 5, we show qualitative comparisons between partial convolution-based padding and zero padding for WideResNet38 for the tile based evaluation mode. We only compare visual results between zero padding and partial convolution-based padding results since they are the best two performing methods. It can be seen that partial convolution-based padding leads to better segmentation results on border regions.

Focused Evaluation on Border Regions. To better understand the advantage of partial convolution-based padding in handling the regions close to the boundaries, we perform some additional experiments where we only evaluates the mIoUs on the border regions. Specifically, we set the target labels to “don’t care” for the center region of the image at varying proportions, as illustrated in Figure 6. These evaluations use non-overlapping tiling method. Table 4 shows the corresponding evaluation results by leaving out

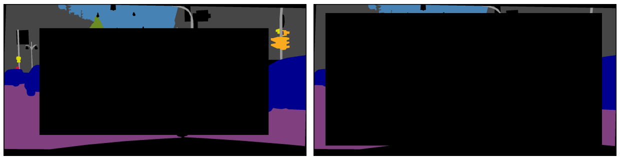


Fig. 6: Samples with different proportions of leaving out center regions. Specifically, we set the target labels to dont care for the black region of the image at varying proportions. This way we can demonstrate how the partial convolution-based padding scheme improves prediction accuracy near the image boundaries, where it is more effective.

	Proportion of image center left out				
Padding	0	$\frac{1}{3} \times \frac{1}{3}$	$\frac{2}{3} \times \frac{2}{3}$	$\frac{3}{4} \times \frac{3}{4}$	$\frac{7}{8} \times \frac{7}{8}$
WN38_zero	79.530	80.472	81.197	80.684	78.981
WN38_ref	78.509	79.583	80.084	79.370	77.218
WN38_rep	78.303	79.132	79.354	78.616	76.892
WN38_partial	**79.907**	**81.051**	**82.031**	**81.600**	**80.230**

TABLE 4: Evaluation of segmentation results with tile based evaluation method and when leaving out different proportions of the center region as shown in Figure 6 and explained in main text in Section 4.1.3.

different proportions of the center regions. It can be seen that as we leave out more proportions of the center region, the evaluation difference between zero padding and partial convolution-based padding becomes larger. For example, if we leave out $\frac{1}{3} \times \frac{1}{3}$ center regions, the partial convolution-based padding outperforms the zero padding with 0.385% mIoU. However, when we leave out $\frac{7}{8} \times \frac{7}{8}$ center regions, the difference becomes 1.249%. This empirically demonstrates that the partial convolution-based padding scheme significantly improves prediction accuracy near the image boundaries.

4.1.4 *Video Action Recognition*

Video action recognition models analyze multiple frames along the temporal axis, typically using 3D convolutions to achieve this. Due to the additional temporal dimension, padding must be performed in three dimensions instead of two, resulting in an increased proportion of padded values in the padded tensor. Thus, padding choices are potentially more important for 3D convolutional models.

We compare partial convolution-based padding against zero padding in the 3D convolution scenario using the 3D ResNet50 models available on Github[2] released by [19]. The models are trained and tested with the default settings provided on the Kinetics-400 dataset [32], taking video clips of 16 frames of 112×112 pixels as input and predicting one of 400 possible action categories.

As shown in Table 5, compared with zero padding, partial convolution-based padding obtains 1.5% top-1 accuracy improvement. Note that the dataset provides only YouTube links, and some videos are no longer accessible. We end up downloading 237,337 videos out of the original 246,535 videos for training and 19,535 videos out of the original 19,907 videos for validation. The unavailability in data may partially explain the discrepancies between our zero padding results and those originally reported. An additional difference is that the original results were based on Lua-Torch whereas, we are basing our experiment on their PyTorch re-implementation. Nonetheless, the partial convolution result outperforms our zero-padding internal baseline and the padding result in [19] as well as reflection and repetition padding baselines. We find the improvements obtained from partial convolution-based padding to be more significant in this task than the object classification and segmentation tasks. We believe this result to be the outcome of increased importance of the padding choice when 3D convolutional models are used since now padding must be performed in three dimensions. These results show that partial convolution-based padding may be an essential component when 3D data is processed such as in videos, or medical datasets.

Padding	Round1	Round2	mean	[19]
3D RN50_zero	60.3	60.6	60.5	$61.3^{\#}$
3D RN50_ref	60.3	60.1	60.2	-
3D RN50_rep	60.6	59.1	59.8	-
3D RN50_partial	61.9	62.1	**62.0**	-

TABLE 5: Accuracies on Kinetics validation set. *_zero, *_ref and *_rep, and *_partial indicate the corresponding model with zero, reflection, repetition, and partial convolution-based paddings. Note that for the baseline zero padding result, there is some difference between our experiments (round 1 & 2) and the experiment in [19]. Nonetheless, the partial convolution result outperforms the zero padding result in [19].

4.2 Partial Convolution-based Image Inpainting

We provide additional qualitative results with partial convolution in image inpainting task which was previously explored in [37]. The experiment set-up uses a UNet-like architecture [52] similar to the one used in [27] to fill the hole in the image with plausible imagery. All convolutional layers in the network are replaced with partial convolutional layers. The network is trained with various reconstruction losses: per-pixel loss, perceptual loss [30], style loss [16], and total variation loss.

To validate the effectiveness of partial convolution for image inpainting, we train the network with both traditional convolution layer and partial convolution layer. We use the images from Places2 dataset [92] and ImageNet [11]. We generated 55,116 random masks using the occlusion/disocclusion mask estimation method described in [61] for training and 12,000 masks with different hole-to-image area ratios from 0.01 to 0.6 for testing. We follow the previous image inpainting works [78], [84] by reporting ℓ_1 error, PSNR, SSIM [71], and the inception score [53]. ℓ_1 error, PSNR and SSIM are reported on Places2, whereas the Inception score (IS) is reported on ImageNet as it uses a pretrained image object classification network.

Table 6 and Figure 7 show quantitative and qualitative results of comparing outputs of networks using traditional convolution layers (Conv) and of networks using partial convolution layers (Partial Conv). As can be seen in Table 6 and Figure 7, partial convolution can robustly handle holes of any shape, size location, or distance from the image borders compared to traditional convolution operation. Furthermore, the performance of network empowered

[2] https://github.com/kenshohara/3D-ResNets-PyTorch

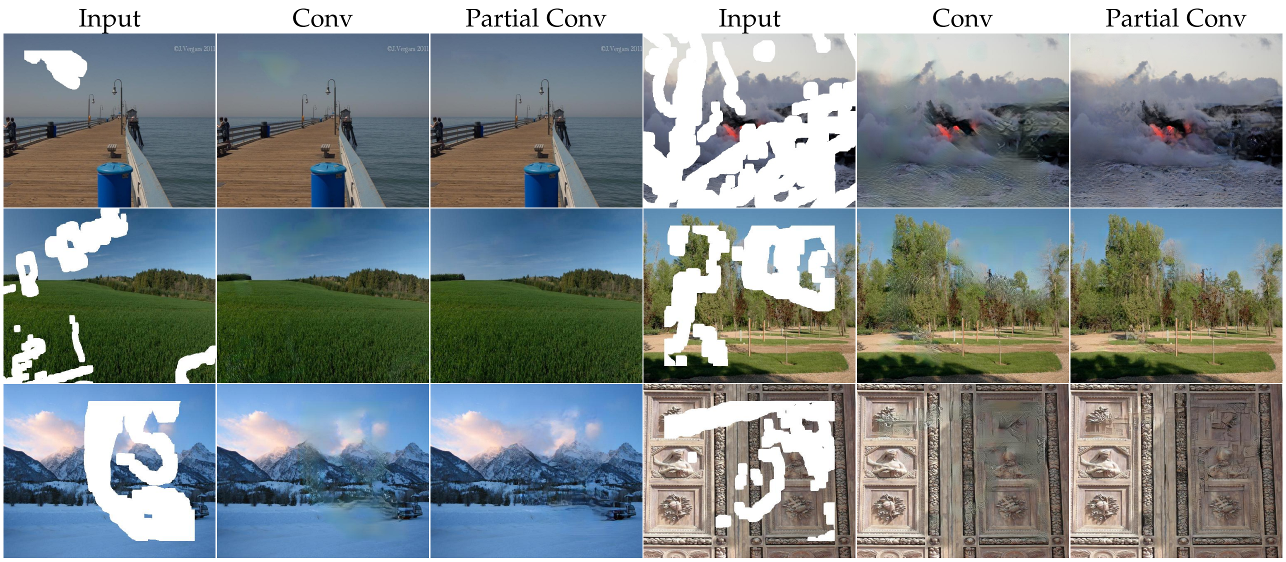


Fig. 7: Comparison between typical convolution layer based results and partial convolution layer based results. Convolution layers do not handle the invalid pixels, resulting in lack of texture in hole regions and obvious color discrepancies. On the other hand, partial convolution can robustly handle holes of any shape, size, and location.

Hole Ratio	[0.01, 0.2]	(0.2, 0.4]	(0.4, 0.6]
ℓ_1(Conv)(%)	0.863	2.653	5.338
ℓ_1(PartialConv)(%)	**0.808**	**2.495**	**5.098**
IS(Conv)	0.122	0.741	2.486
IS(PartialConv)	**0.088**	**0.559**	**1.978**
SSIM(Conv)	0.903	0.724	0.526
SSIM(PartialConv)	**0.907**	**0.731**	**0.533**
PSNR(Conv)	30.548	23.380	19.627
PSNR(PartialConv)	**31.030**	**23.673**	**19.743**

TABLE 6: Comparisons between a typical convolution layer and a partial convolution layer for image inpainting. Lower is better for ℓ_1(%) and IS; higher is better for SSIM and PSNR.

with partial convolutions does not deteriorate catastrophically as holes increase in size as can be seen in Figure 7. Looking at the metrics in Table 6, partial convolution based models consistently outperform the networks using traditional convolutional layers for both small and large hole ratios. We are also interested in analyzing if partial convolution-based image inpainting improves more with larger inpainting ratios. This is difficult to measure with ℓ_1 error, PSNR and SSIM metrics since they compare output image with ground-truth and expects a match between them. However, when the inpainting ratio increases, there are more uncertainities even for the color of the pixels and not matching the ground-truth may not necessarily be a bad thing. IS is an important metric to look at for those cases since it looks at the overall image quality. Looking at IS, as the ratio gets larger, the partial convolution method improves the results with a larger margin. For example, baseline versus partial convolutions are 0.741 versus 0.559 for hole ratios between (0.2, 0.4] (a margin of 0.182) and 2.486 versus 1.978 for (0.4, 0.6] ratios (a margin of 0.508).

Note that in this task, we compare traditional convolution and partial convolution and not the padding schemes. This task has the similarities with padding formulation as both require handling missing pixels. Whereas in padding schemes, reflection, and repetition paddings are proposed, they are not used in inpainting since the holes are irregular.

4.3 Partial Convolution-based Image Synthesis

In this section, we provide more qualitative and quantitative results for partial convolution-based image synthesis, previously explored in [14]. We use the same network architecture and training parameters given in [46] and run image synthesis experiments on the Cityscapes dataset [10]. Synthesized images are of size 256 × 512. The baseline network is conditioned on semantic maps concatenated with boundary maps. Boundary maps are generated from instance maps as such: a pixel in the boundary map is set to 1 if its object identity is different from any of its 4 neighbors, and 0 otherwise. This conditioning is done via spatially adaptive normalization layer, SPADE [46]. SPADE layers process concatenated semantic and boundary maps via convolution layers and output scale and shift parameters. Other than that, this baseline architecture uses regular convolution layers before each normalization layer. In our experiments, we replace those convolution layers with partial-based convolution layers. The image generator with partial convolution additionally uses panoptic maps. Masks that are used in partial convolution are constructed based on panoptic maps as described in Section 3. This way, convolutional output conditioned on an instance or a semantic class depends only on the feature values that belong to the same instance or class.

We expect the partial convolution-based generator to synthesize images with higher fidelity to the underlying segmentation and instance information, especially when objects are small and instances are partially occluded. To measure that, we evaluate the generated images with state-of-the-art segmentation and object detection networks. Specifically, DRN-D-105 [83] is used to measure segmentation accuracy, and Faster-RCNN [51] with a ResNet-50 backbone is used for measuring detection accuracy. State-of-the-art

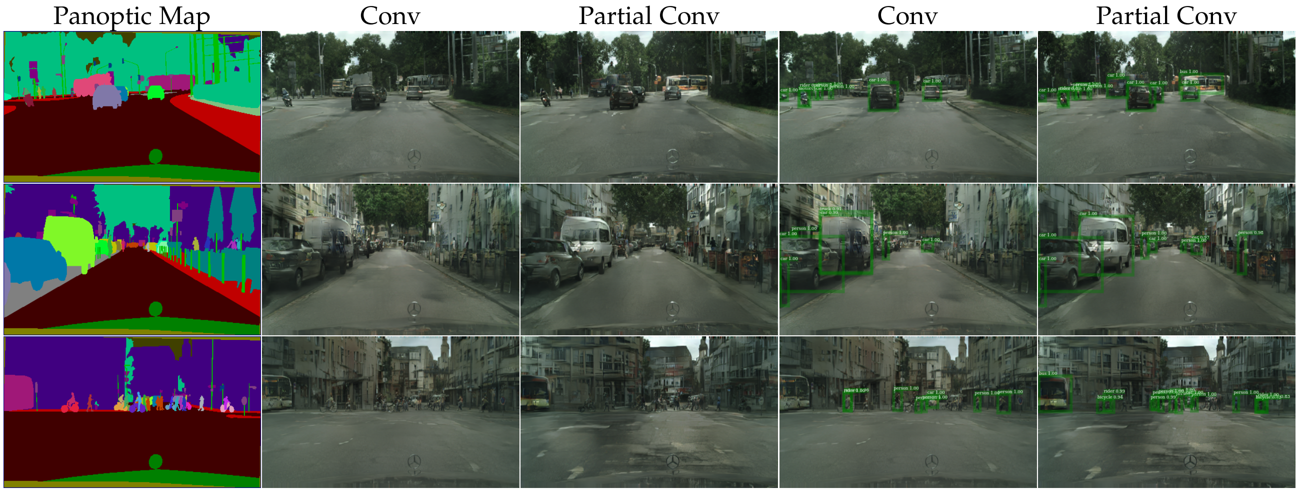


Fig. 8: Visual comparison of image synthesis results on Cityscapes dataset. Columns show in the order of panoptic maps, images synthesized with convolution and partial convolution based generators, and bounding box detections from Faster-RCNN on those syntehsized images. The partial convolution based generator outputs images with instances that can be detected by Faster-RCNN with higher accuracy.

Method	mIoU	detAP
Image generator with convolution	60.00	10.97
Image generator with partial convolution	**61.24**	**11.50**

TABLE 7: Image synthesis evaluation results on Cityscapes dataset. Results are averaged over 3 runs.

	Inference			
Trained	zero/diff	ref/diff	rep/diff	partial/diff
zero	**76.13**/0	72.79/-3.33	73.52/-2.60	59.77/-16.35
ref	74.09/-2.08	**76.17**/0	74.14/-2.03	73.46/-2.71
rep	73.85/-2.36	73.08/-3.13	**76.21**/0	73.15/-3.06
partial	75.88/-0.73	73.31/-3.30	74.00/-2.61	**76.61**/0

TABLE 8: Cross testing of using different padding schemes. Each row represent a model trained with the corresponding padding methods. Each column indicates use of different padding methods during inference. All models are ResNet50 and averaged over 5 runs.

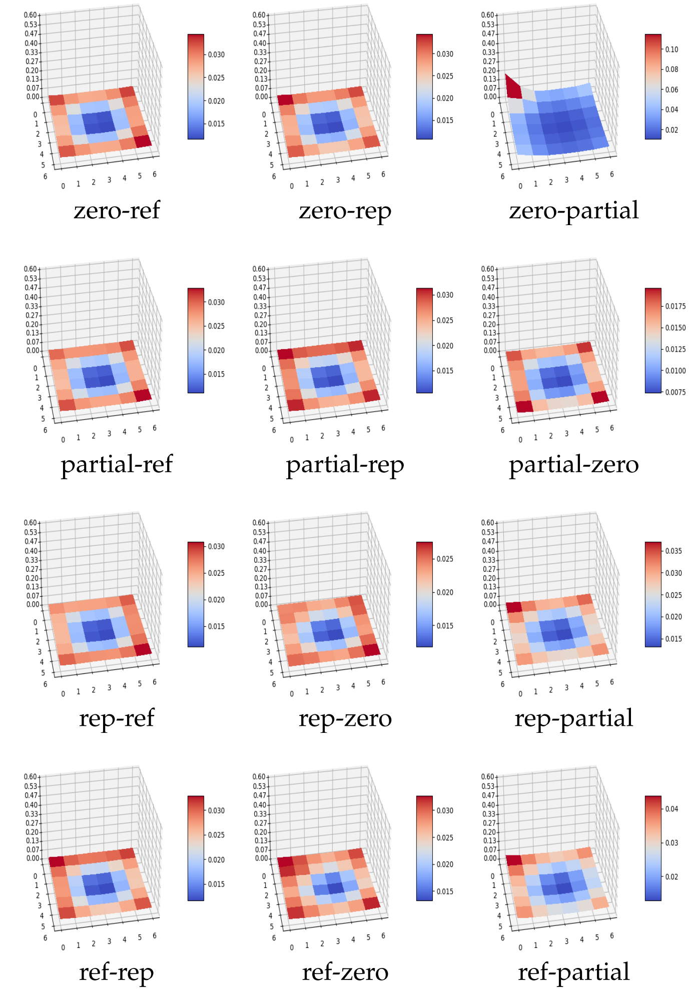
segmentation and object detection networks we use are quite sensitive to small objects and instance boundaries and can detect such improvements quantitatively. As can be seen in Table 7, the scores improve with the incorporation of partial convolutions. In the qualitative results shown in Figure 8, we see that the partial convolution-based generator outputs images with instances that can be detected by Faster-RCNN with higher accuracy. Specifically, we find that the partial convolution-based generator generates distinct cars even when they are occluding each other, and can generate detectable people even when they are far away as shown in Figure 8. As can be seen in Figure 8, we find that the convolution based architecture may blend the pattern and texture of objects among neighboring instances, whereas our method clearly separates them.

5 Analysis and Discussion

In this section, we provide more in-depth discussion on the performance gains with partial convolution based padding. In our experiments, we see that partial convolution-based padding achieves superior performance over other padding schemes in image classification, segmentation, and especially in video action recognition tasks. Our experiments cover various tasks and also various networks such as 2D convolutions in popular networks of VGG and ResNet as well as dilated convolutions for segmentation tasks in ResNet and WideResNet, and 3D convolutions for video recognition tasks. We showcase that with different architectures and convolution operations, partial convolution-based padding is always the winner. Furthermore, the improvements come with zero cost, if implemented in CUDA, since it only requires a re-weight of the convolution results at the border. We also observe that zero padding achieves better accuracies when compared to reflection and repetitive padding. Reflection and repetitive padding introduce unrealistic borders, this may cause CNNs to have a more difficult job when handling boundaries. Padding these borders with zeros also generates unrealistic borders but at least it is consistent among all images and can be handled by CNNs better. Another interesting finding we observe is that explicit padding [25] which require eight additional filters dedicated to borders and corners improve results in simple task of learning Gaussian filtering and deteriorate the accuracies in image classification task. It also comes with additional parameters and around 5× slowdown during training and inference.

Robustness to Cross Testing. We argue that the improvements of partial convolution-based padding is provided because the network learns more robust features. We experiment that with cross testing set-up between zero, reflection, repetition and partial convolution-based padding networks. We set this experiment, for example, by training a zero padding network and during inference time replacing the zero padding with partial convolution-based padding. We do the same for all these different padding based networks and with all combinations.

Results are presented in Table 8. Each row presents



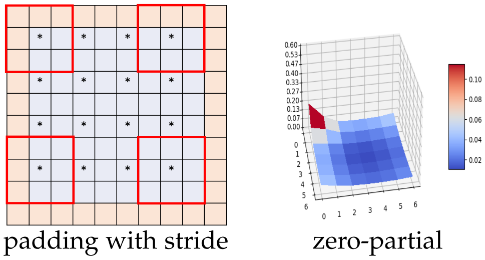
zero-ref zero-rep zero-partial

partial-ref partial-rep partial-zero

rep-ref rep-zero rep-partial

ref-rep ref-zero ref-partial

Fig. 9: Feature distance visualization of cross testing experiments. In the first row, model trained with zero padding is used for inference with different padding methods. In this set-up, we run inference with two models, zero-padding versus another, and plot the differences between the feature values. We calculate the mean absolute differences of feature values from the last layer of the network which have a spatial dimension of 7×7. Distances are averaged for validation set. With these plots, we are able to see how much feature values change when trained with zero-padding and inferred with a different padding scheme. The other rows show the same set-up with partial, repetition, and reflection paddings. The largest differences are observed when model is trained with zero padding and inferred with partial padding.



padding with stride zero-partial

Fig. 10: [Left] Visualization of 3x3 convolution operation with stride 2x2 and padding 1x1. As it shows the left convolution windows utilize more padded entries than right convolution windows. Top-left corner utilizes more padded entries than any other corners. [Right] Feature distance visualization of zero versus partial padding.

the results of the network trained with the corresponding padding scheme (e.g. first row trained with zero padding). Each column corresponds to the padding method used during inference. The results are averaged over 5 runs. Diagonal results are the best results since same padding is used for both training and inference. In other cells, we also share the difference by subtracting the performance from its diagonal correspondence. As expected, our results show that when the padding scheme is changed during inference, the results degrade. However, we also observe networks that are powered with partial padding during training are more robust to padding switches and in average their performances degrade less. They provide the second best results two times out of three when different paddings are used during inference. Interestingly, we observe in Table 8 that a model trained with zero padding suffers a significant drop in test-time accuracy if switched to the partial convolution-based padding for inference. On the other hand, the performance drop when applying a model trained with partial convolution padding to zero-padded data is nominal (16% versus 0.7%). We also see that reflection and repetition padding methods are more robust to padding switches than zero padding since they do not need to over-specialize to handle borders as much as zero-padding requires.

We extend our cross testing analysis in Figure 9 where instead of looking at the accuracy of the models, we plot how much feature values deviate when padding scheme is switched. We look at the features from the last convolutional layer of ResNet50 which has a spatial dimension of 7×7. In the first row, we load a model that is trained with zero-padding and observe how much feature values change when we run the model with another padding versus zero padding. Other rows present the same settings for partial, repetition, and reflection paddings. Since these features are towards the end of the network, all feature values are affected but as expected the corners and borders are affected more than the center pixels. We see that models trained with zero padding and inferred with partial padding results in the most difference between feature values overall and especially a large spike on the left corner and around the edges. The reason for the larger spike on the left corner is because the ResNet architecture employs many convolutional layers with kernel size 3 or 7, stride 2, and padding 1 or 3 to decrease the spatial dimension. As shown in Figure 10[Left], when convolutional layer is applied with kernel size 3, stride 2, and padding 1, there is more padding effect on the top-left corner which is consistent with our feature map distance visualization as also given in 10[Right]. When kernel size 7, stride 2 and padding 3 is used, the difference of how many padded entries are used between left corners and right corners becomes even larger. For the other layers in ResNet where kernel sizes are 3, strides are 1, and paddings are 1, all corners are affected the same way.

Based on Eq. 1, we see that partial convolution-based padding is corresponding to the widely used zero padding

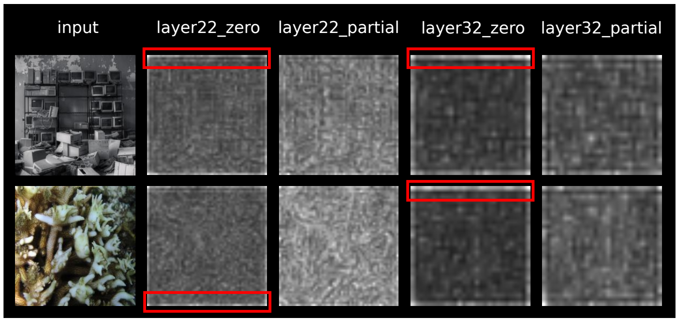


Fig. 11: Activation Map at 22nd layer and 32nd layer of VGG19BN network with zero padding and VGG19BN network with partial padding. These two layers are ReLU layers, and we sum up the activation along channels and resize the summation for visualization. Red rectangles show the strong activation regions from VGG19BN network with zero padding whereas response magnitudes are relatively uniform for the partial convolution.

with the scaling factor given in Eq. 2. Even though it may seem that the change between zero and partial padding is subtle, computationally they output very different results. In fact, it may seem that zero and partial padding are the most similar ones to each other, however, based on our analysis that is not true. There may be cases when the border values of feature maps are close to zero which results in repetition and reflection paddings outputting values similar to the zero padding results. On the other hand, partial padding has a scaling factor which can change the results more significantly. The obtained improvements and the behaviour of zero and partial convolution-based padding in the cross testing provide further evidence of unwanted model specialization occurring in models trained on zero-padded data to deal with image borders. On the other hand, partial convolution-based padding does not suffer from such specialization to handle boundaries and show more robustness. Reflection and repetition paddings also do not suffer from such specialization but they perform worse than partial padding.

Activation Map Visualization. We further investigate the behaviour of zero and partial convolution-based paddings when handling boundaries. They are the top 2 performing methods in our experiments and have subtle differences where partial convolution-based padding introduces a scaling factor among the borders. However, they behave very differently in cross testing experiments, partial convolution-based padding improves the accuracies significantly over zero-padding and show more roboustness. To understand the behaviour of these two methods, we visualize the activations of various convolutional layers of partial padding and zero padding based networks for image classification task. Figure 11 shows the activations for two examples where zero-padding fails but partial convolution-based padding succeeds to correctly classify them. By visualizing the activation maps, we notice, for models using zero padding, the features at the border have strong activation responses compared to the other spatial locations. On the other hand, the response magnitudes are relatively uniform throughout for the partial padded model. We conjecture that the higher response magnitudes along the edges for the zero-padded models means that the model is trying to overcome effects from padded zero values. However, it sometimes fails, and produces less accurate results compared to the counterpart network with partial convolution-based padding. These visuals are also consistent with the semantic segmentation experiments from Section 4.1.3. In segmentation experiments, we observe that the largest improvements were obtained near the image boundaries since partial convolution-based padding help the network output robust features across the edges.

6 Conclusion

In this paper, we have reviewed various applications of partial convolution in CNNs. We demonstrated qualitative improvements provided by partial convolution in image inpainting and synthesis tasks. We have evaluated the effectiveness of partial convolution-based padding compared to existing padding methods. We demonstrate that it outperforms the widely adopted zero padding through extensive experiments on image classification, semantic segmentation and video action recognition tasks. The improvements are consistent, and improve on strong baselines.

References

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015. 6

[2] G. Balakrishnan, A. Zhao, A. V. Dalca, F. Durand, and J. Guttag. Synthesizing images of humans in unseen poses. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8340–8348, 2018. 2

[3] A. Bhattad, A. Dundar, G. Liu, A. Tao, and B. Catanzaro. View generalization for single image textured 3d models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6081–6090, 2021. 1

[4] S. R. Bulò, L. Porzi, and P. Kontschieder. In-place activated batchnorm for memory-optimized training of dnns. *CoRR, abs/1712.02616, December*, 5, 2017. 6

[5] R. Cao, X. Zhong, F. Scalzo, S. Raman, and K. Sung. Prostate cancer inference via weakly-supervised learning using a large collection of negative mri. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019. 3

[6] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *arXiv preprint arXiv:1802.02611*, 2018. 1, 6

[7] M. Chen, B. H. Newell, Z. Sun, C. A. Corr, and W. Gao. Reconstruct missing pixels of landsat land surface temperature product using a cnn with partial convolution. In *Applications of Machine Learning*, volume 11139, page 111390E. International Society for Optics and Photonics, 2019. 3

[8] M. Chen, X. Zhao, and D. Xu. Image inpainting for digital dunhuang murals using partial convolutions and sliding window method. In *Journal of Physics: Conference Series*, volume 1302, page 032040. IOP Publishing, 2019. 3

[9] H.-T. Cheng, C.-H. Chao, J.-D. Dong, H.-K. Wen, T.-L. Liu, and M. Sun. Cube padding for weakly-supervised saliency prediction in 360 videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1420–1429, 2018. 2

[10] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 6, 9

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 8

[12] Y. Ding, C. Wang, H. Huang, J. Liu, J. Wang, and L. Wang. Frame-recurrent video inpainting by robust optical flow inference. *arXiv preprint arXiv:1905.02882*, 2019. 3

[13] A. Dundar, J. Gao, A. Tao, and B. Catanzaro. Fine detailed texture learning for 3d meshes with generative models. *arXiv preprint arXiv:2203.09362*, 2022. 1

[14] A. Dundar, K. Sapra, G. Liu, A. Tao, and B. Catanzaro. Panoptic-based image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8070–8079, 2020. 2, 4, 9

[15] A. Dundar, K. Shih, A. Garg, R. Pottorff, A. Tao, and B. Catanzaro. Unsupervised disentanglement of pose, appearance and background from images and videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 2

[16] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015. 8

[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 2

[18] X. Han, Z. Zhang, D. Du, M. Yang, J. Yu, P. Pan, X. Yang, L. Liu, Z. Xiong, and S. Cui. Deep reinforcement learning of volume-guided progressive view inpainting for 3d point scene completion from a single depth image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 234–243, 2019. 3

[19] K. Hara, H. Kataoka, and Y. Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6546–6555, 2018. 1, 8

[20] A. W. Harley, K. G. Derpanis, and I. Kokkinos. Segmentation-aware convolutional networks using local attention masks. In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, page 7, 2017. 2

[21] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 2

[22] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 3, 5, 6

[23] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz. Multimodal unsupervised image-to-image translation. *European Conference on Computer Vision (ECCV)*, 2018. 2

[24] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (TOG)*, 36(4):107, 2017. 1, 2

[25] C. Innamorati, T. Ritschel, T. Weyrich, and N. J. Mitra. Learning on the edge: Explicit boundary handling in cnns. *arXiv preprint arXiv:1805.03106*, 2018. 2, 5, 6, 10

[26] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 2

[27] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 8

[28] V. Ivashkin and V. Lebedev. Spatiotemporal data fusion for precipitation nowcasting. *arXiv preprint arXiv:1812.10915*, 2018. 3

[29] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9000–9008, 2018. 1

[30] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016. 8

[31] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1

[32] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 8

[33] M. Kegler, P. Beckmann, and M. Cernak. Deep speech inpainting of time-frequency masks. *arXiv preprint arXiv:1910.09058*, 2019. 3

[34] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[35] A. X. Lee, R. Zhang, F. Ebert, P. Abbeel, C. Finn, and S. Levine. Stochastic adversarial video prediction. *arXiv preprint arXiv:1804.01523*, 2018. 1

[36] Y. Li, S. Liu, J. Yang, and M.-H. Yang. Generative face completion. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 3, 2017. 2

[37] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro. Image inpainting for irregular holes using partial convolutions. *arXiv preprint arXiv:1804.07723*, 2018. 2, 3, 4, 8

[38] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, 2016. 1

[39] X. Liu, G. Yin, J. Shao, X. Wang, et al. Learning to predict layout-to-image conditional convolutions for semantic image synthesis. In *Advances in Neural Information Processing Systems*, pages 568–578, 2019. 3, 6

[40] D. Lorenz, L. Bereska, T. Milbich, and B. Ommer. Unsupervised part-based disentangling of object shape and appearance. In *CVPR*, 2019. 2

[41] M. Mardani, G. Liu, A. Dundar, S. Liu, A. Tao, and B. Catanzaro. Neural ffts for universal texture image synthesis. *Advances in Neural Information Processing Systems*, 33, 2020. 1

[42] J. Mathai, I. Masi, and W. AbdAlmageed. Does generative face completion help face recognition? *arXiv preprint arXiv:1906.02858*, 2019. 3

[43] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018. 3

[44] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010. 2

[45] D. Novotny, B. Graham, and J. Reizenstein. Perspectivenet: A scene-consistent image generator for new view synthesis in real indoor environments. In *Advances in Neural Information Processing Systems*, pages 7599–7610, 2019. 3

[46] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019. 2, 3, 9

[47] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016. 1, 2

[48] A. Pimkin, A. Samoylenko, N. Antipina, A. Ovechkina, A. Golanov, A. Dalechina, and M. Belyaev. Multi-domain ct metal artifacts reduction using partial convolution based inpainting. *arXiv preprint arXiv:1911.05530*, 2019. 3

[49] F. A. Reda, D. Sun, A. Dundar, M. Shoeybi, G. Liu, K. J. Shih, A. Tao, J. Kautz, and B. Catanzaro. Unsupervised video interpolation using cycle consistency. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 892–900, 2019. 1

[50] J. S. Ren, L. Xu, Q. Yan, and W. Sun. Shepard convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 901–909, 2015. 2

[51] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1, 9

[52] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 5, 6, 8

[53] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016. 8

[54] K. J. Shih, A. Dundar, A. Garg, R. Pottorf, A. Tao, and B. Catanzaro. Video interpolation and prediction with unsupervised landmarks. *arXiv preprint arXiv:1909.02749*, 2019. 1

[55] M.-L. Shih, S.-Y. Su, J. Kopf, and J.-B. Huang. 3d photography using context-aware layered depth inpainting. *arXiv preprint arXiv:2004.04727*, 2020. 3

[56] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 5, 6

[57] Y. Song, C. Yang, Z. Lin, H. Li, Q. Huang, and C.-C. J. Kuo. Image inpainting using multi-scale feature image translation. *arXiv preprint arXiv:1711.08590*, 2017. 2

[58] V. Sterzentsenko, L. Saroglou, A. Chatzitofis, S. Thermos, N. Zioulis, A. Doumanoglou, D. Zarpalas, and P. Daras. Self-supervised deep depth denoising. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1242–1251, 2019. 3

[59] K. Sun, Y. Zhao, B. Jiang, T. Cheng, B. Xiao, D. Liu, Y. Mu, X. Wang, W. Liu, and J. Wang. High-resolution representations for labeling pixels and regions. *arXiv preprint arXiv:1904.04514*, 2019. 7

[60] L. Sun, Y. Zhang, X. Chang, Y. Wang, and J. Xu. Cloud-aware generative network: Removing cloud from optical remote sensing images. *IEEE Geoscience and Remote Sensing Letters*, 2019. 3

[61] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *European conference on computer vision*, pages 438–451. Springer, 2010. 8

[62] P. Teterwak, A. Sarna, D. Krishnan, A. Maschinot, D. Belanger, C. Liu, and W. T. Freeman. Boundless: Generative adversarial networks for image extension. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10521–10530, 2019. 2

[63] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012. 2

[64] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger. Sparsity invariant cnns. *arXiv preprint arXiv:1708.06500*, 2017. 2

[65] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Deep image prior. *arXiv preprint arXiv:1711.10925*, 2017. 2

[66] A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016. 2

[67] F. Wan, Ö. Smedby, and C. Wang. Simultaneous mr knee image segmentation and bias field correction using deep learning and partial convolution. In *Medical Imaging 2019: Image Processing*, volume 10949, page 1094909. International Society for Optics and Photonics, 2019. 3

[68] J. Wang, K. Chen, R. Xu, Z. Liu, C. C. Loy, and D. Lin. Carafe: Content-aware reassembly of features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3007–3016, 2019. 3

[69] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018. 1, 2, 3

[70] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018. 2

[71] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 8

[72] D. Wei, L. Zhang, Z. Wu, X. Cao, G. Li, D. Shen, and Q. Wang. Deep morphological simplification network (ms-net) for guided registration of brain magnetic resonance images. *Pattern Recognition*, 100:107171, 2020. 3

[73] Y. Wu and K. He. Group normalization. *arXiv preprint arXiv:1803.08494*, 2018. 5

[74] Z. Wu, C. Shen, and A. v. d. Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *arXiv preprint arXiv:1611.10080*, 2016. 6

[75] C. Xie, S. Liu, C. Li, M.-M. Cheng, W. Zuo, X. Liu, S. Wen, and E. Ding. Image inpainting with learnable bidirectional attention maps. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8858–8867, 2019. 2

[76] H. Xiong, C. Wang, D. Tao, M. Barnett, and C. Wang. Multiple sclerosis lesion inpainting using non-local partial convolutions. *arXiv preprint arXiv:1901.00055*, 2018. 3

[77] R. Xu, X. Li, B. Zhou, and C. C. Loy. Deep flow-guided video inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3723–3732, 2019. 1

[78] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li. High-resolution image inpainting using multi-scale neural patch synthesis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 3, 2017. 2, 8

[79] X. Yang, X. Yang, M.-Y. Liu, F. Xiao, L. Davis, and J. Kautz. STEP: Spatio-temporal progressive learning for video action detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1

[80] Z. Yang, X. He, J. Gao, L. Deng, and A. Smola. Stacked attention networks for image question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 21–29, 2016. 2

[81] R. Yeh, C. Chen, T. Y. Lim, M. Hasegawa-Johnson, and M. N. Do. Semantic image inpainting with perceptual and contextual losses. *arXiv preprint arXiv:1607.07539*, 2016. 2

[82] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations (ICLR)*, 2016. 1

[83] F. Yu, V. Koltun, and T. Funkhouser. Dilated residual networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 9

[84] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Generative image inpainting with contextual attention. *arXiv preprint arXiv:1801.07892*, 2018. 1, 2, 8

[85] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Free-form image inpainting with gated convolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4471–4480, 2019. 2

[86] N. Yu, G. Liu, A. Dundar, A. Tao, B. Catanzaro, L. S. Davis, and M. Fritz. Dual contrastive loss and attention for gans. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6731–6742, 2021. 3

[87] T. Yu, C. Lin, S. Zhang, X. Ding, J. Wu, J. Zhang, et al. End-to-end partial convolutions neural networks for dunhuang grottoes wall-painting restoration. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019. 3

[88] X. Zhan, X. Pan, B. Dai, Z. Liu, D. Lin, and C. C. Loy. Self-supervised scene de-occlusion. *arXiv preprint arXiv:2004.02788*, 2020. 3

[89] H. Zhang, K. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal. Context encoding for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 6

[90] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018. 2, 3

[91] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2881–2890, 2017. 6, 7

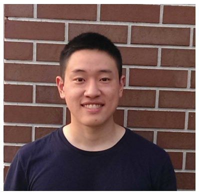
[92] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2017. 8

[93] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 2

[94] Y. Zhu, K. Sapra, F. A. Reda, K. J. Shih, S. Newsam, A. Tao, and B. Catanzaro. Improving semantic segmentation via video propagation and label relaxation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8856–8865, 2019. 1, 6, 7

Guilin Liu is a senior research scientist at NVIDIA. He obtained his Ph.D. in Computer Science from George Mason University in 2017. He received his B.E. in Spatial Informatics and Digitalized Technology together with a minor degree in Finance from Wuhan University in 2012. His research has attracted many attentions from some media outlets like Forbes, MIT Technology Review, Yahoo Finance etc. His research interests include deep learning for image/video editing and creation, machine learning for graphics and geometric data processing.

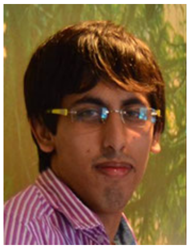
Aysegul Dundar is an Assistant Professor of Computer Science at Bilkent University, Turkey and a Sr. Research Scientist at NVIDIA. She received her Ph.D. degree at Purdue University in 2016, under supervision of Professor Eugenio Culurciello. She received a B.Sc. degree in Electrical and Electronics Engineering from Bogazici University in Turkey, in 2011. Her current research focuses are on domain adaptation, image segmentation, and generative models for image synthesis and manipulation.



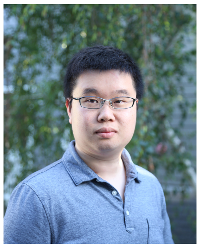
Kevin J. Shih is a research scientist in the Applied Deep Learning Research team at NVIDIA. He obtained his Ph.D. in Computer Science from University of Illinois at Urbana–Champaign under the supervision of Professor Derek Hoiem. Prior to that, he received his B.S.E from the University of Michigan. His research interests include object localization, pose estimation, attention mechanisms, and models that handle multiple modalities.

Ting-Chun Wang is a senior research scientist at NVIDIA in Santa Clara, US. He obtained his Ph.D. from University of California, Berkeley, department of EECS, advised by Professor Ravi Ramamoorthi and Alexei A. Efros. He received his B.E from National Taiwan University. He is a recipient of the Berkeley Fellowship. His research interests include computer vision, machine learning and computer graphics, particularly the intersections of all three. His recent research focus is on using generative adversarial models to synthesize realistic images and videos, with applications to rendering, visual manipulations and beyond.

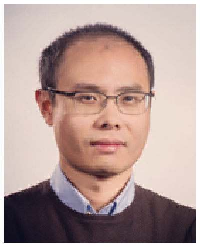
Fitsum A. Reda received the B.S. degree in electrical engineering from Mekelle University, Ethiopia, in 2006, the joint Erasmus Mundus M.S. degree in computer vision and robotics from Heriot-Watt University, Edinburgh, UK; Girona University, Girona, Spain; University of Burgundy, Le Creusot, France, all in 2009, and the PhD degree in electrical engineering from Vanderbilt University, Nashville, TN in 2014. From 2014 to 2016, he was a senior scientist at Siemens Healthcare, where he focused on learning techniques for medical image understanding and parsing. He was a senior research scientist at NVIDIA, where he focuses on video understanding for graphics and real-life vision. His primary research interests are in deep learning, computer vision, and medical imaging.



Karan Sapra is a Senior Research Scientist in the Applied Deep Learning Research team at NVIDIA. He graduated with his Ph.D. in Computer Engineering from Clemson University. His research interests include Deep Learning in Computer Vision, Graph Theory, and genomic networks. He has also previously worked in Peer-to-peer networks, cloud computing, and High-Performance Computing.



Zhiding Yu joined NVIDIA Research as a Research Scientist in 2018. he received Ph.D. in ECE from Carnegie Mellon University in 2017, and M.Phil. in ECE from The Hong Kong University of Science and Technology in 2012. His research interests mainly focus on deep representation learning, weakly/self-supervised learning, transfer learning and deep structured prediction, with their applications to vision and robotics problems. He is a winner of the Domain Adaptation for Semantic Segmentation Track, WAD Challenge at CVPR 2018. He is a co-author of the best student paper at ISCSLP 2014, and a winner of the best paper award at WACV 2015. His intern work on deep facial expression recognition at Microsoft Research won first runner-up at the EmotiW-SFEW Challenge 2015 and was integrated into the Microsoft Emotion Recognition API under the Microsoft Azure Cognitive Services.



Xiaodong Yang is a Principal Scientist at QCraft. Before joining QCraft in 2019, he was a Senior Research Scientist at NVIDIA Research. His research interests are in the areas of computer vision and machine learning. He has been working on perception and prediction for autonomous driving, image and video understanding, human activity and hand gesture recognition, dynamic facial analytics, target re-identification, deep generative modeling, multimedia search, assistive technology, etc. He received the B.S. degree from Huazhong University of Science and Technology, China, in 2009, and the Ph.D. degree from City University of New York, USA, in 2015. He is a recipient of the best paper award from Journal of Visual Communication and Image Representation in 2015 for his work on action recognition. His collaborators and he win first place in the optical flow competition of Robust Vision Challenge at CVPR 2018. He is recognized as AI2000 Most Influential Scholar Honorable Mention in 2020. He regularly serves on program committees and reviews papers for major computer vision and machine learning conferences. He co-organized tutorials and workshops at GTC 2019, CVPR 2019, and CVPR 2020.

Andrew Tao is a Distinguished Engineer and Manager of the Computer Vision side of the Applied Deep Learning Research group at Nvidia. He received his Masters in Electrical Engineering from Stanford University in 1992 with an emphasis on Computer Architecture. He has worked as a CPU hardware engineer, as GPU hardware engineer and architect, as the Director of Applied Architecture at Nvidia, and has led a number of Computer Vision teams in the Automotive sector.

Bryan Catanzaro is Vice President of Applied Deep Learning Research at NVIDIA. After receiving his Ph.D. from UC Berkeley in 2011, he worked as a research scientist at NVIDIA on programming models and applications for GPUs, focusing on libraries for neural networks, which led to the creation of the CUDNN library. He worked at Baidu Silicon Valley AI Lab from 2014-2016, contributing to the DeepSpeech project. In 2016, he returned to NVIDIA to build a lab applying deep learning to problems in computer vision, graphics, speech, language, and chip design.