UNVEILING THE MASK OF POSITION-INFORMATION PATTERN THROUGH THE MIST OF IMAGE FEATURES

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent studies have shown that paddings in convolutional neural networks encode absolute position information which can negatively affect the model performance for certain tasks. However, existing metrics for quantifying the strength of positional information remain unreliable and frequently lead to erroneous results. To address this issue, we propose novel metrics for measuring and visualizing the encoded positional information. We formally define the encoded information as Position-information Pattern from Padding (PPP) and conduct a series of experiments to study its properties as well as its formation. The proposed metrics measure the presence of positional information more reliably than the existing metrics based on PosENet and tests in F-Conv. We also demonstrate that for any extant (and proposed) padding schemes, PPP is primarily a learning artifact and is less dependent on the characteristics of the underlying padding schemes.

1 INTRODUCTION

Padding, one of the most fundamental components in neural network architectures, has received much less attention than other modules in the literature. In convolutional neural networks (CNNs), zero padding is frequently used perhaps due to its simplicity and low computational costs. This design preference remains almost unchanged in the past decade. Recent studies (Islam* et al., 2020; Islam et al., 2021b; Kayhan & Gemert, 2020; Innamorati et al., 2020) show that padding can implicitly provide a network model with positional information. Such positional information can cause unwanted side-effects by interfering and affecting other sources of position-sensitive cues (e.g., explicit coordinate inputs (Lin et al., 2022; Alsallakh et al., 2021a; Xu et al., 2021; Ntavelis et al., 2022; Choi et al., 2021), embeddings (Ge et al., 2022), or boundary conditions of the model (Innamorati et al., 2020; Alguacil et al., 2021; Islam et al., 2021a)). Furthermore, padding may lead to several unintended behaviors (Lin et al., 2022; Xu et al., 2021; Ntavelis et al., 2022; Choi et al., 2021), degrade model performance (Ge et al., 2022; Alguacil et al., 2021; Islam et al., 2021a), or sometimes create blind spots (Alsallakh et al., 2021a). Meanwhile, simply ignoring the padding pixels (known as no-padding or valid-padding) leads to the foveal effect (Alsallakh et al., 2021b; Luo et al., 2016) that causes a model to become less attentive to the features on the image border. These observations motivate us to thoroughly analyze the phenomenon of positional encoding including the effect of commonly used padding schemes.

Conducting such a study requires reliable metrics to detect the presence of positional information introduced by padding, and more importantly, quantify its strength consistently. We observe that the existing methods for detecting and quantifying the strength of positional information yield inconsistent results. In Section 3, we revisit two closely related evaluation methods, PosENet (Islam* et al., 2020) and F-Conv (Kayhan & Gemert, 2020). Our extensive experiments demonstrate that (a) metrics based on PosENet are unreliable with an unacceptably high variance, and (b) the Border Handling Variants (BHV) test in F-Conv suffers from unaware confounding variables in its design, leading to unreliable test results.

In addition, we observe all commonly-used padding schemes actually encode consistent patterns underneath the highly dynamic model features. However, such a pattern is rather obscure, noisy, and visually imperceptible for most paddings (except zeros-padding), which makes recognizing and analyzing it difficult. Fortunately, we show that such patterns can be consistently revealed with a sufficient number of samples by defining an optimal padding scheme (see Section 2.1 and Figure 1).

The source codes and data collection scripts will be made publicly available.

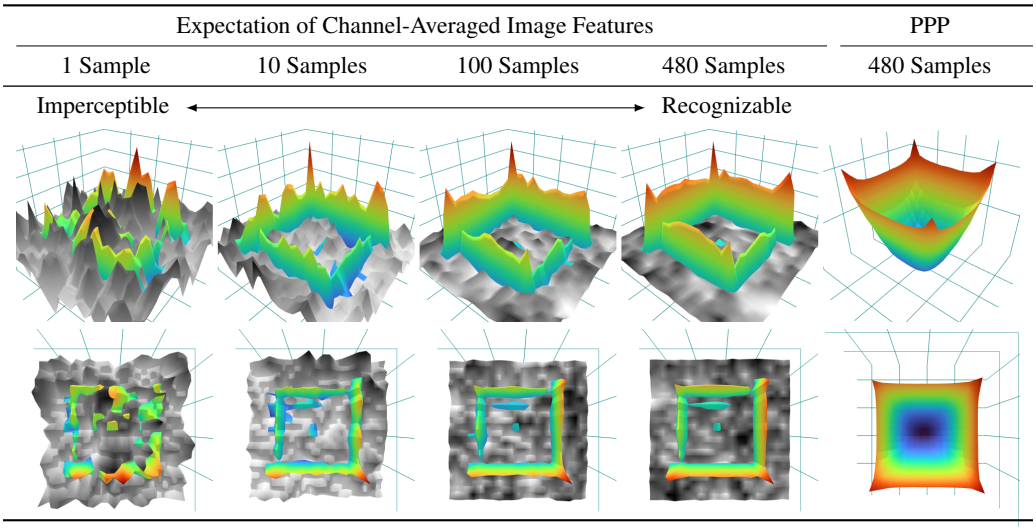

Figure 1: **Position-information Pattern from Padding (PPP).** We propose a method that can consistently and effectively extract PPPs through the distributional difference between optimally-padded (gray-scale surfaces) and algorithmically-padded features (colored surfaces). The results show that the two distributions become distinguishable as the number of sample increases. Following the procedure in Section 2.2, we extract a clear view of PPP with the expectation of the pair-wise differences between optimally-padded and algorithmically-padded features. We render each visualization in tilted view (first row) and top view (second row). The colors represent the magnitude (blue/cold/weak to green/warm/strong) at each pixel. The features are extracted at the 3rd layer of interest (Appendix A) from a randn-padded (Section 2.4) ResNet50 pretrained on ImageNet.

We accordingly propose a new evaluation paradigm and develop a method to consistently detect the presence of the Position-information Pattern from Padding (PPP), which is a persistent pattern embedded in the model features to retain positional information. We present two metrics to measure the response of PPP from the signal-to-noise perspective and demonstrate its robustness and low deviation among different settings, each with multiple trials of training.

To weaken the effect of PPP, in Section 2.4, we design a padding scheme with built-in stochasticity, making it difficult for the model to consistently construct such biases. However, our experiments show that the models can still circumvent the stochasticity and end up consistently constructing PPPs. These results suggest that a model likely constructs PPPs purposely to facilitate its training, rather than falsely or accidentally learning some filters that respond to padding features.

With reliable PPP metrics, we conduct a series of experiments to analyze the characteristics of PPP in Section 4.1. Specifically, we analyze the formation of PPP throughout each model training process in Section 4.3. The results show PPPs are formed expeditiously at the early stage of model training, slowly but steadily strengthen through time, and eventually shaped in clear and complete patterns. These results show that a model intentionally develops and reinforces PPPs to facilitate its learning process. Moreover, we observe the PPPs of all pretrained networks are significantly stronger than those in their initial states. This indicates an unbiased training procedure is of great importance in resolving the critical failures caused by PPP in numerous vision tasks (Alsallakh et al., 2021a; Xu et al., 2021; Ge et al., 2022; Alguacil et al., 2021).

2 Observations and Methodology

In this section, we first define symbols for expressing the functionality of paddings and define the optimal-padding scheme. We then give a formal definition of Position-information Pattern from Padding (PPP) and utilize the optimal-padding scheme to develop propose a method to capture PPP and measure its response with two metrics.

2.1 Optimal Padding

The process of capturing an image from the real world can be simplified into two steps: (a) 3D information of the environment is first projected onto an infinitely large 2D plane, and then (b) the camera determines resolution as well as field-of-view to form a digital image from such infinitely large

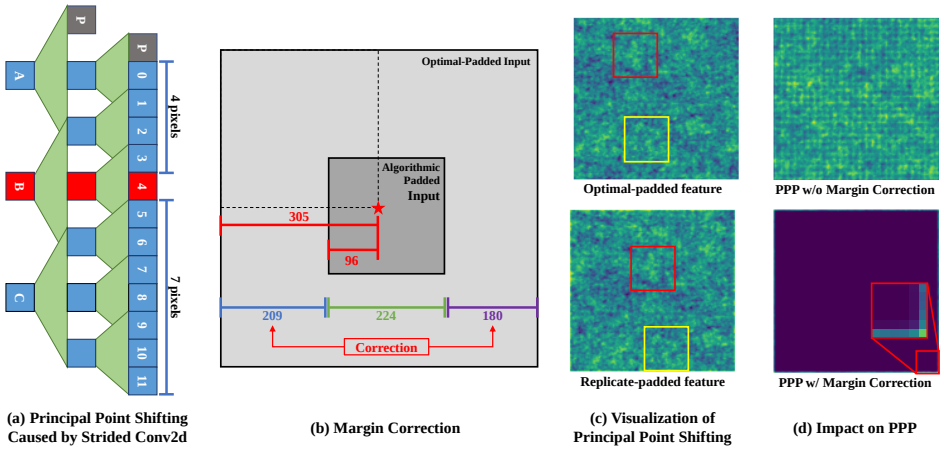

Figure 2: **Principal point shift.** (a) The stride-2 Conv2d only pads on one side, causing the principal point shift (red squares) in earlier layers. (b) Such a shift requires careful margin correction while aligning algorithmically-padded and optimally-padded features (we describe the details of point shift in Appendix A). (c) The shift is visible in the feature space (marked with red and yellow boxes). (d) It is crucial to correct the principal point shift while measuring PPP. The PPP calculation involves pixel-wise distance functions, which are not robust to spatial shifts (Zhang et al., 2018).

and continuous 2D signals (Liu et al., 2019; Ravi et al., 2020). Let $S^* = \{s_n^*\}_{n=1}^N$ be a collection of such infinitely large and continuous 2D signals, and the collection of 2D images captured by cameras at a spatial size (h_n, w_n) be $S' = \{s'_n\}_{n=1}^N$. A padding scheme can be used to generate a set of *algorithmically-padded* images $\hat{S} = \{\hat{s}_n\}_{n=1}^N$ by a padding function ρ:

$$\hat{s}_n[i,j] = \begin{cases} s'_n[i,j] = s^*[i,j] & \text{if } 0 < i < h_n \text{ and } 0 < j < w_n\,, \\ \rho(s'_n, i, j) & \text{otherwise,} \end{cases} \tag{1}$$

where i and j are indexes of a pixel in the spatial dimension. We define a theoretical *optimally-padded* collection $S^\dagger = \{s_n^\dagger\}_{n=1}^N$ with an optimal-padding function $\rho^\dagger$ by:

$$s_n^\dagger[i,j] = \begin{cases} s'_n[i,j] & = s^*[i,j] & \text{if } 0 < i < h_n \text{ and } 0 < j < w_n\,, \\ \rho^\dagger(s'_n, i, j) & = s^*[i,j] & \text{otherwise.} \end{cases} \tag{2}$$

In practice, without curated data, the *optimal*-padding scheme described in Eq. 2 is difficult to achieve. We describe how we relax this constraint in Section 2.3

2.2 Positional-information Pattern from Padding

Despite the previous literature discovered the existence of positional information caused by the model paddings, there is still no clear definition for such information, and lacks effective metrics to detect or quantify it. Ideally, an effective metric for such positional information should have two properties. First, it is a spatial pattern, it contributes distinctive information to different spatial locations. Its shape enables the network to develop and exploit the absolute positional information of each pixel, eventually leading to the unattended and undesirable effects in certain tasks (Lin et al., 2022; Alsallakh et al., 2021a; Xu et al., 2021; Ntavelis et al., 2022; Choi et al., 2021; Ge et al., 2022; Alguacil et al., 2021). Second, as it represents the positional information purely contributed by the padding, it is a constant pattern irrelevant to the image contents. We accordingly name it the Positional-information Pattern from Padding (PPP).

Unfortunately, such a pattern shares space with image features, where the image features typically have very diverse appearances and high dimensionality. When these two signals *interfere* with each other, the appearance of PPP becomes extremely obscure and imperceptible in most cases (except zeros padding). Figure 1 shows if we visualize features sample-by-sample, there are no obvious differences between optimally-padded features (gray-scale surface) and algorithmically-padded features (colored surface). To address the issue, we show that, by assuming the interferences between PPP and image features to be random, its expectation over a large set of images will saturate to a constant bias and no longer hinder us from capturing PPP.

Based on these observations and assumptions, we define PPP as the constant component independent of model inputs, and its presence is completely contributed by the existence of a padding scheme ρ. Given $\hat{S}$ and a model $F(\hat{s}; \theta, \rho)$, which θ is the model parameters and ρ is a padding scheme applied to F. Let the model feature extracted at k-th layer be $f_{n,k} = F_k(\hat{s}_n; \theta, \rho)$, where F_k is the model from the first layer to the k-th layer. The PPP at k-th layer (PPP_k) can be formulated by:

$$\text{PPP}_k \;=\; \mathop{\mathbb{E}}_{n} \Big[\; d\left(\; F_k(s_n^{\dagger}; \theta, \rho^{\dagger}) \;,\; F_k(\hat{s}_n; \theta, \rho) \;\right) \;\Big] \;, \tag{3}$$

where $d(\cdot, \cdot)$ can be any distance function. We use ℓ_1 distance in this work, and accordingly name the metric PPP-MAE.

Pitfalls: feature misalignment. It is important to note that, some CNN components can cause serious feature misalignment while computing PPP and leads to erroneous results. A typical example is *principal point shift*, where the uneven padding in stride-2 convolution causes the center of features slightly drifted, as shown in Figure 2. Since the measurement of PPP requires perfect alignment, such a drift should be carefully considered while integrating PPP into new architectures. We discuss the issue along with other pitfalls in Appendix A and provide three detailed examples of correcting the principal point shifting.

2.3 Simulated Optimal Padding

In practice, it is impossible to gain access to S^* for calculating the optimal padding $S^{\dagger}$ described in Eq. 2. But fortunately, given our goal in Eq. 3 is to analyze the model features within the (h_n, w_n) region, S^* is an overshoot of the data we actually required. Given a vision model $F(\hat{s}; \theta, \rho)$ trained at a field-of-view (h_n, w_n) pixels, the receptive field of such vision model is (h_m, w_m) pixels (we show the computation in Appendix A), where $h_m \gg h_n$ and $w_m \gg w_n$. Let an alternative image collection $S^{\odot} = \{s_n^{\odot}\}_{n=1}^{N}$ at (h_m, w_m) pixels, the definition of receptive field implies $F_k(s_n^{\dagger}; \theta, \rho)$ equals to $F_k(s_n^{\odot}; \theta, \rho^{\dagger})$ for all k.

In other words, in terms of computing Eq. 3, $S^{\odot}$ is equivalent to S^* within the finite (h_n, w_n) region for a given model architecture. Therefore, we can simulate the procedure described in Eq. 1 and Eq. 2 using $S^{\odot}$ instead of $S^{\dagger}$, as long as $\forall s_n^{\odot} \in S^{\odot}$ the spatial size of $s_n^{\odot}$ is strictly larger than (h_m, w_m).

2.4 Randn Padding

Most of the existing padding schemes (e.g., zeros, reflect, replicate, circular) exhibit certain consistent patterns that can be easily detected by some designed convolutional kernels. One may argue that the nature of easy detectability can be a root cause of encouraging the models to learn to rely on these obvious patterns. This motivates us to design an additional sampling-based padding scheme without any consistent patterns, namely randn (i.e., random normal) padding, which produces dynamical values from a normal distribution while following the local statistics. We first determine the maximal and minimal values of a sliding window (which can be easily achieved with max-pooling), use the average of them as a proxy mean μ_p, and use the difference between the mean and the maximal value as a proxy standard deviation σ_p. For each padding location, we sample the padding value according to a normal distribution $\mathcal{N}(\mu_p, \sigma_p^2)$ from the nearest sliding window. We include more implementation details in Appendix A.

Aside from creating a pattern-less padding scheme with sampling, the design of randn padding is based on several factors. The sampled padding pixels are allowed to occasionally exceed the min/max bound of the sliding window. Without breaking the min/max bound can introduce detectable patterns in certain extreme cases, such as a gradient-like feature that has its maximal intensity at the top-left corner and minimal intensity at the bottom-right corner. We also design the padding scheme to follow the local distribution. The padding exhibits high entropy when the local variation is high, while degenerates to value repetition with imperceptible perturbations while padding a flat area. As such, not only do the padding pixels exhibit less pattern, but it also prevents the padding pixels from breaking the features in the border region. We later show that a model still deliberately and incredibly built up PPP over time even with such a sophisticated padding scheme.

3 Revisiting Prior Work

In this section, we first reproduce two experiments from the prior art, which aim to assess positional information from paddings. We show several critical design issues in these experiments and discuss how these problems affect the drawn conclusions. Finally, we propose two additional experiments to quantify the amount of positional information embedded in the paddings.

Table 1: **Background color as a critical confounding variable in BHV test.** We show that using a grey background similar to Figure 3 leads to discrepant results. The standard deviations are reported among 10 individual trials. We mark the best performance in green, and the worst two in red.

Padding	F-Conv?	Black Background				Grey Background			
		Similar (%)	Dissimilar (%)	Diff (%)	Inconsistency (%)	Similar (%)	Dissimilar (%)	Diff (%)	Inconsistency (%)
Zeros	N	$99.83_{\pm 0.00}$	$3.21_{\pm 8.35}$	-87.68	$95.81_{\pm 2.07}$	$100.00_{\pm 0.00}$	$4.96_{\pm 5.93}$	-95.04	$97.85_{\pm 4.55}$
	Y	$89.24_{\pm 0.98}$	$89.24_{\pm 0.98}$	0.00	$18.02_{\pm 8.08}$	$100.00_{\pm 0.00}$	$4.77_{\pm 6.52}$	-95.23	$96.79_{\pm 7.13}$
Circular	N	$80.31_{\pm 3.23}$	$80.31_{\pm 3.23}$	0.00	$34.25_{\pm 8.32}$	$72.75_{\pm 0.96}$	$72.75_{\pm 0.96}$	0.00	$26.30_{\pm 5.55}$
	Y	$99.20_{\pm 0.23}$	$93.14_{\pm 2.88}$	-6.06	$18.48_{\pm 3.55}$	$98.26_{\pm 0.50}$	$92.40_{\pm 4.23}$	-5.87	$28.67_{\pm 6.18}$
Reflect	N	$100.00_{\pm 0.00}$	$15.67_{\pm 12.72}$	-84.33	$91.18_{\pm 13.19}$	$100.00_{\pm 0.00}$	$19.96_{\pm 13.54}$	-80.04	$90.33_{\pm 11.95}$
	Y	$100.00_{\pm 0.00}$	$11.70_{\pm 15.38}$	-88.30	$97.33_{\pm 6.16}$	$100.00_{\pm 0.00}$	$17.16_{\pm 12.19}$	-82.84	$98.13_{\pm 3.44}$
Replicate	N	$100.00_{\pm 0.00}$	$43.39_{\pm 11.42}$	-56.61	$75.32_{\pm 8.20}$	$100.00_{\pm 0.00}$	$33.16_{\pm 6.42}$	-66.83	$84.09_{\pm 6.47}$
	Y	$98.32_{\pm 0.39}$	$93.65_{\pm 1.36}$	-4.67	$32.60_{\pm 4.97}$	$97.17_{\pm 0.48}$	$94.99_{\pm 1.20}$	-2.18	$32.15_{\pm 5.11}$
Randn	N	$100.00_{\pm 0.00}$	$10.31_{\pm 12.56}$	-89.70	$94.88_{\pm 5.55}$	$99.97_{\pm 0.13}$	$35.47_{\pm 10.82}$	-64.50	$83.59_{\pm 8.48}$
	Y	$100.00_{\pm 0.00}$	$20.80_{\pm 14.15}$	-79.20	$92.54_{\pm 8.37}$	$77.28_{\pm 16.13}$	$66.70_{\pm 11.58}$	-10.59	$45.70_{\pm 20.62}$
No-pad	-	$100.00_{\pm 0.00}$	$3.21_{\pm 8.35}$	-96.79	$95.81_{\pm 2.07}$	$100.00_{\pm 0.00}$	$30.07_{\pm 4.06}$	-69.93	$81.30_{\pm 2.44}$

3.1 POSENET

Islam *et al*. show zeros-padding provides CNN models positional information cues, and propose PosENet (Islam* et al., 2020) to quantify the amount of positional information encoded within CNN features. A PosENet experiment involves several components: a pretrained CNN model F, a shallow CNN E_{pem} (i.e., position encoding module), an image dataset $X = \{x_i\}_{i=1}^{N}$ to examine, and a constant target pattern y (e.g., 2D Gaussian pattern). PosENet first extracts intermediate features at k-th layer with $f_{(i,k)} = F_k(x_i)$ using the pretrained CNN, and then optimizes E_{pem} to minimize $\mathbb{E}_{i,k}[||E_{pem}(f_{(i,k)}) - y||_2]$. Finally, the amount of positional information is quantified by the average Spearman's correlation (SPC) and Mean Absolute Error (MAE) overall $E_{pem}(f_{(i,k)})$ toward y.

A critical issue with PosENet is the use of an optimization-based metric. It is sensitive to hyperparameters with large variation. As shown in Table 2, for all the PosENet results, the standard deviation over five trials significantly dominates the differences between different types of paddings, and thus no definitive conclusions can be drawn. We also observed that PosENet can report NaN results in certain setups. Furthermore, PosENet quantifies the amount of positional information by the faithfulness of the final reconstruction. However, a better reconstruction does not have a clear relationship to *measuring* the strength and significance of positional information. For instance, PosENet sometimes shows responses to no-padding models, demonstrating it is a metric with an indefinite bias pending on the memorization ability of E_{pem}. Moreover, optimizing for pattern reconstruction is highly dependent on the underlying data distribution, simply changing the evaluation data distribution without changing the model weights can drastically change the PosENet numerical magnitudes and the conclusions of which model embeds the strongest positional information.

Another issue is that the no-padding scheme used in the E_{pem} module in PosENet is known to have the foveal effect (Alsallakh et al., 2021b; Luo et al., 2016), where a model pays less attention to the information on the edge of inputs. Using such a padding scheme for detecting positional information from paddings, which is mostly concentrated on the edge of the feature maps, is less effective. This is an inevitable dilemma as PosENet aims to identify positional information from the padding of the pretrained F, while applying any padding scheme to E_{pem} introduces intractable effects between the paddings of the two models.

3.2 F-CONV

Kayhan *et al*. propose a full-padding scheme (F-Conv) (Kayhan & Gemert, 2020) and demonstrate it is more translational invariant than the alternatives. One of the critical results is on "border handling variants" (Exp 2 of (Kayhan & Gemert, 2020)), which we call it BHV test. The BHV test creates a toy dataset, where each image has a black background with a green square and a red square in the foreground. The task is to predict if the red square is on the left of the green square (class 1), or vice versa (class 2). In addition, Kayhan *et al*. intentionally adds a *location bias* such that both squares are located in the upper half of the image for class 1, and located in the lower half of the image for class 2. During testing, a "similar test" inherits the same bias, while a "dissimilar test" exchanges the bias (i.e., both squares are in the lower half of the image for class 1). As a truly translation-invariant CNN model should not be affected by the location bias, it should focus on the relation between

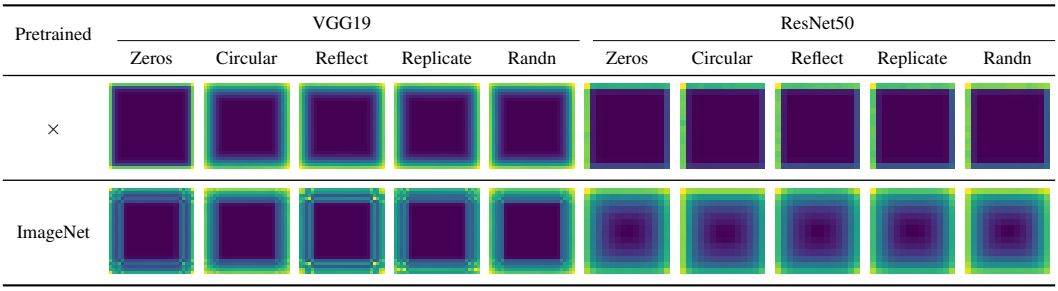

Figure 4: **Visualization of Position-Information Pattern from Padding (PPP).** The visualizations are calculated based on Eq. 3 over 480 GMap samples extracted at the 3rd layer-of-interest (Appendix A). The results show that the pretrained model significantly reinforces PPP compared to randomly initialized networks. Note that each image is normalized to $[0, 1]$ separately, therefore the colors between images are not comparable. More visualizations are presented in Appendix B.

the red and green squares and perform similarly on both tests. Since the experimental results show that F-Conv performs best on the dissimilar test, it is concluded that F-Conv is less sensitive to the location bias. The authors also conclude the circular padding performs worse due to the behavior of wrapping the pixels to the other side of the image, which leads to confusion between two classes.

However, as shown in Figure 3, we find the experimental design does not consider a crucial confounding variable: the black background has a zero intensity, making zeros padding the optimal padding that perfectly follows the background distribution. In Table 1, we show that the dissimilar test is no longer in favor of F-Conv zeros after changing the background color to grey. We also show that F-Conv replicate and F-Conv circular perform best on the dissimilar test, which is different from the original observation.

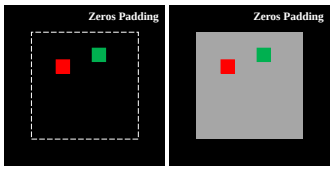

Figure 3: The BHV test trains a binary classifier to predict the relative position of the two colored squares. It hypothesizes if the padding provides no positional information, the classifier will only focus on the relative position of the two squares. (Left) The black background is a confounding variable. (Right) Zeros padding no-longer pads optimum values after changing the background color.

Finally, we report an additional inconsistency rate to show that the CNN architecture used in the BHV test actually has access to the absolute position of the squares. Given a random sample in class 1, we create a *trajectory* of samples by simultaneously moving the two squares to the bottom of the canvas and recording the CNN-model prediction in all intermediate states. We label a trajectory to be *inconsistent* if the prediction of the CNN-model switches classes at any step of the trajectory. A CNN model with no access to the absolute-position information should have all trajectories maintaining consistent predictions, with 0% inconsistency. Table 1 shows the inconsistent ratio over 228 uniformly sampled trajectories, where all models maintain high inconsistency rates, even with a no-padding architecture. These results show that the CNN model used in the BHV test is not translation invariant. This can be attributed to that a CNN model has a large receptive field covering the whole experiment canvas, therefore capable of gradually constructing absolute coordinates for each input pixel. Note that we only show the design of the BHV test is not suitable for quantifying the amount of positional information exhibited in a CNN model. Such a conclusion does not imply that F-Conv cannot potentially improve the translation-invariant property of CNNs.

4 Experiments and Analysis

Datasets Since most vision models are trained on tasks for recognizing objects, an image collection containing a diverse object appearance is more suitable for the task. As mentioned in Section 2.3, evaluating PPP requires images at a large field-of-view, in practice, we collect images at $2{,}048 \times 2{,}048$ pixels, which is larger than the receptive field of all the models we tested. Due to the constraint of large field-of-view, we compute PPP on three datasets (all at $2{,}048 \times 2{,}048$ pixels): (a) 480 satellite images crawled from Google Map, (b) 1,024 images synthesized by InfinityGAN (Lin et al., 2022) trained with Flickr-Landscape dataset, and (c) 1,024 images synthesized by InfinityGAN trained with LSUN-Tower (Yu et al., 2015) dataset. We crop the images depending on the requested input image sizes and principal point shifts from each model (see Appendix A for details). We will release the script for collecting and composing these large images.

Table 2: **Comparing PosENet and our proposed PPP metrics. Most of the PosENet results are not distinguishable due to the high standard deviations.** The standard deviation is computed by five different pretrained models for each test. The performance shows the accuracy (for classification) or weighted F-measure score (for saliency object detection). We use 2D Gaussian as PosENet reconstruction pattern, and PPP-MAE is measured at the 4th layer of interest. Here, (↑) indicates a higher value corresponds to stronger positional information or better performance on the task (vice versa for (↓)). For each group of pretrained models, we label the strongest positional information response with red, and the experiments within its standard deviation range with orange.

Model	Padding	Eval Dataset	PosENet		PPP-MAE(ours) (↑)	Performance (↑)
			SPC (↑)	MAE (↓)		
VGG-19	Zeros	GMap	$0.107_{\pm 0.128}$	$0.196_{\pm 0.006}$	$0.0176_{\pm 0.0005}$	$74.0972_{\pm 0.0870}$
		InfinityGAN-flickr	$0.368_{\pm 0.116}$	$0.183_{\pm 0.007}$	$0.0163_{\pm 0.0006}$	
		InfinityGAN-tower	$0.492_{\pm 0.106}$	$0.173_{\pm 0.010}$	$0.0179_{\pm 0.0001}$	
	Circular	GMap	$0.098_{\pm 0.139}$	$0.197_{\pm 0.007}$	$0.0158_{\pm 0.0006}$	$74.4716_{\pm 0.0863}$
		InfinityGAN-flickr	$0.323_{\pm 0.147}$	$0.185_{\pm 0.009}$	$0.0137_{\pm 0.0004}$	
		InfinityGAN-tower	$0.460_{\pm 0.102}$	$0.176_{\pm 0.009}$	$0.0184_{\pm 0.0005}$	
	Reflect	GMap	$0.109_{\pm 0.139}$	$0.196_{\pm 0.007}$	$0.0158_{\pm 0.0002}$	$74.0516_{\pm 0.0621}$
		InfinityGAN-flickr	$0.343_{\pm 0.132}$	$0.185_{\pm 0.008}$	$0.0146_{\pm 0.0008}$	
		InfinityGAN-tower	$0.460_{\pm 0.113}$	$0.177_{\pm 0.009}$	$0.0168_{\pm 0.0005}$	
	Replicate	GMap	$0.084_{\pm 0.137}$	$0.197_{\pm 0.006}$	$0.0144_{\pm 0.0009}$	$73.9964_{\pm 0.1079}$
		InfinityGAN-flickr	$0.356_{\pm 0.111}$	$0.184_{\pm 0.007}$	$0.0128_{\pm 0.0012}$	
		InfinityGAN-tower	$0.498_{\pm 0.111}$	$0.173_{\pm 0.010}$	$0.0156_{\pm 0.0006}$	
	Randn	GMap	$0.125_{\pm 0.154}$	$0.195_{\pm 0.006}$	$0.0182_{\pm 0.0012}$	$73.7716_{\pm 0.0758}$
		InfinityGAN-flickr	$0.374_{\pm 0.137}$	$0.185_{\pm 0.007}$	$0.0167_{\pm 0.0008}$	
		InfinityGAN-tower	$0.421_{\pm 0.161}$	$0.181_{\pm 0.010}$	$0.0186_{\pm 0.0012}$	
	NoPad	GMap	$0.001_{\pm 0.239}$	$0.204_{\pm 0.013}$	$0.0000_{\pm 0.0000}$	$62.0396_{\pm 0.0830}$
		InfinityGAN-flickr	$0.303_{\pm 0.192}$	$0.187_{\pm 0.012}$	$0.0000_{\pm 0.0000}$	
		InfinityGAN-tower	$0.516_{\pm 0.139}$	$0.172_{\pm 0.014}$	$0.0000_{\pm 0.0000}$	
ResNet50	Zeros	GMap	$0.191_{\pm 0.188}$	$0.193_{\pm 0.008}$	$0.0162_{\pm 0.0012}$	$75.6856_{\pm 0.0924}$
		InfinityGAN-flickr	$0.682_{\pm 0.107}$	$0.152_{\pm 0.019}$	$0.0137_{\pm 0.0004}$	
		InfinityGAN-tower	$0.721_{\pm 0.077}$	$0.144_{\pm 0.017}$	$0.0153_{\pm 0.0013}$	
	Circular	GMap	$0.398_{\pm 0.115}$	$0.197_{\pm 0.007}$	$0.0188_{\pm 0.0016}$	$76.1432_{\pm 0.1026}$
		InfinityGAN-flickr	$0.628_{\pm 0.084}$	$0.159_{\pm 0.013}$	$0.0178_{\pm 0.0005}$	
		InfinityGAN-tower	$0.585_{\pm 0.105}$	$0.165_{\pm 0.014}$	$0.0189_{\pm 0.0012}$	
	Reflect	GMap	$0.197_{\pm 0.185}$	$0.192_{\pm 0.008}$	$0.0150_{\pm 0.0004}$	$75.5068_{\pm 0.1213}$
		InfinityGAN-flickr	$0.594_{\pm 0.096}$	$0.169_{\pm 0.012}$	$0.0134_{\pm 0.0009}$	
		InfinityGAN-tower	$0.667_{\pm 0.087}$	$0.153_{\pm 0.016}$	$0.0157_{\pm 0.0002}$	
	Replicate	GMap	$0.249_{\pm 0.192}$	$0.189_{\pm 0.009}$	$0.0138_{\pm 0.0003}$	$75.6122_{\pm 0.0911}$
		InfinityGAN-flickr	$0.700_{\pm 0.095}$	$0.147_{\pm 0.018}$	$0.0114_{\pm 0.0003}$	
		InfinityGAN-tower	$0.726_{\pm 0.069}$	$0.142_{\pm 0.016}$	$0.0142_{\pm 0.0007}$	
	Randn	GMap	$0.210_{\pm 0.192}$	$0.191_{\pm 0.009}$	$0.0147_{\pm 0.0007}$	$75.3076_{\pm 0.1016}$
		InfinityGAN-flickr	$0.566_{\pm 0.100}$	$0.171_{\pm 0.011}$	$0.0122_{\pm 0.0011}$	
		InfinityGAN-tower	$0.714_{\pm 0.068}$	$0.142_{\pm 0.015}$	$0.0153_{\pm 0.0004}$	
SOD (PiCANet)	Zeros	GMap	$0.156_{\pm 0.212}$	$0.201_{\pm 0.017}$	$0.0049_{\pm 0.0001}$	$0.6269_{\pm 0.0015}$
		InfinityGAN-flickr	$0.365_{\pm 0.140}$	$0.184_{\pm 0.012}$	$0.0036_{\pm 0.0001}$	
		InfinityGAN-tower	$0.449_{\pm 0.120}$	$0.179_{\pm 0.013}$	$0.0032_{\pm 0.0001}$	
	Circular	GMap	$0.011_{\pm 0.209}$	$0.207_{\pm 0.014}$	$0.0062_{\pm 0.0001}$	$0.6260_{\pm 0.0009}$
		InfinityGAN-flickr	$0.329_{\pm 0.133}$	$0.187_{\pm 0.012}$	$0.0068_{\pm 0.0001}$	
		InfinityGAN-tower	$0.398_{\pm 0.115}$	$0.182_{\pm 0.011}$	$0.0050_{\pm 0.0002}$	
	Reflect	GMap	$0.062_{\pm 0.210}$	$0.205_{\pm 0.016}$	$0.0053_{\pm 0.0001}$	$0.6243_{\pm 0.0022}$
		InfinityGAN-flickr	$0.322_{\pm 0.133}$	$0.188_{\pm 0.013}$	$0.0030_{\pm 0.0001}$	
		InfinityGAN-tower	$0.396_{\pm 0.125}$	$0.183_{\pm 0.013}$	$0.0039_{\pm 0.0001}$	
	Replicate	GMap	$0.071_{\pm 0.215}$	$0.204_{\pm 0.016}$	$0.0043_{\pm 0.0002}$	$0.6255_{\pm 0.0013}$
		InfinityGAN-flickr	$0.335_{\pm 0.139}$	$0.186_{\pm 0.012}$	$0.0023_{\pm 0.0001}$	
		InfinityGAN-tower	$0.409_{\pm 0.120}$	$0.182_{\pm 0.012}$	$0.0032_{\pm 0.0001}$	
	Randn	GMap	$0.002_{\pm 0.244}$	$0.202_{\pm 0.009}$	$0.0001_{\pm 0.0000}$	$0.2570_{\pm 0.0022}$
		InfinityGAN-flickr	$0.228_{\pm 0.173}$	$0.197_{\pm 0.010}$	$0.0001_{\pm 0.0000}$	
		InfinityGAN-tower	$0.212_{\pm 0.148}$	$0.200_{\pm 0.011}$	$0.0001_{\pm 0.0000}$	
	NoPad	GMap	$0.000_{\pm 0.264}$	$0.211_{\pm 0.019}$	$0.0000_{\pm 0.0000}$	$0.4759_{\pm 0.0013}$
		InfinityGAN-flickr	$0.454_{\pm 0.194}$	$0.178_{\pm 0.021}$	$0.0000_{\pm 0.0000}$	
		InfinityGAN-tower	$0.520_{\pm 0.167}$	$0.172_{\pm 0.020}$	$0.0000_{\pm 0.0000}$	

Table 3: **Significant PPP gain from model training.** We measure PPP-MAE on GMap with randomly initialized and fully trained models. The results show a consistent and significant increment of PPP is developed through the model training.

Model	Pretrained	Padding				
		Zeros	Circular	Reflect	Replicate	Randn
VGG-19	×	$0.0132_{\pm 0.0006}$	$0.0000_{\pm 0.0000}$	$0.0000_{\pm 0.0000}$	$0.0000_{\pm 0.0000}$	$0.0000_{\pm 0.0000}$
	ImageNet	$0.0176_{\pm 0.0005}$	$0.0158_{\pm 0.0006}$	$0.0158_{\pm 0.0002}$	$0.0144_{\pm 0.0009}$	$0.0182_{\pm 0.0012}$
ResNet50	×	$0.0052_{\pm 0.0004}$	$0.0032_{\pm 0.0004}$	$0.0018_{\pm 0.0001}$	$0.0015_{\pm 0.0001}$	$0.0020_{\pm 0.0002}$
	ImageNet	$0.0162_{\pm 0.0012}$	$0.0188_{\pm 0.0016}$	$0.0150_{\pm 0.0004}$	$0.0150_{\pm 0.0004}$	$0.0147_{\pm 0.0007}$

4.1 Visualizing Position-information Pattern from Padding (PPP)

We start with visualizing PPP in Figure 4. All the visualizations are conducted at the 3rd layer of interest as detailed in Appendix A. We compute PPP using Eq. 3 and ℓ_1 norm as the distance metric, then average the resulting PPP in the channel dimension to generate a gray-scale image. Since the quantities are small and difficult to perceive, we normalize the gray-scale image to $[0, 1]$ range, and thus the colors between images are not directly comparable.

In all scenarios, PPP noticeably spreads out after being pretrained on ImageNet. In Table 3, the PPP-MAE of the VGG19 and ResNet50 also reflects that the response of PPP is significantly strengthened after model training. That is, the model training has substantial effects on the construction of PPP. Although the formation of padding pattern is suggested to be mainly caused by the distributional difference between features and paddings (Alsallakh et al., 2021a), our results show that it only increases the response slightly, compared to the considerable PPP-MAE gain through training.

Another intriguing observation is that, despite some variations in the detailed patterns, the overall structure of PPP remains similar. Regardless of padding minimum values with zero-padding (consider the features are processed with ReLU activation), randn-padding that can sometimes produce large quantities by chance, or the unbalanced initial state of ResNet50 caused by strided convolution (the first row of ResNet50 in Figure 4), all models tend to have the maximal PPP response in the corner of the features after fully trained. While the underlying mechanism causing such consistent preferences remains unknown, such preferences may be an important factor to consider in future model design.

4.2 Quantifying PPP and Comparing with PosENet

Table 2 shows the measurements of PPP and PosENet on various architectures and padding schemes. We train five models for each setup and measure the standard deviation of these models. Our PPP-MAE has significantly lower standard deviations compared to PosENet, where the standard deviation of PosENet dominates the differences between padding variants, and thus the quantities from PosENet cannot provide sufficient information for any analysis. Evaluating the true mean of PosENet requires an even larger number of pretrained models, each requiring full training on the target dataset (e.g., ImageNet), which is impractical in reality. The main reason that PosENet has such a large variation is due to its optimization-based formulation, and thus the final quantities highly depend on the convergence of the PosENet training. In fact, we also observe a similar level of standard deviation even when the PosENet is measured on the same model for multiple trials. On the other hand, PPP is based on a closed-form formulation, and thus the variations are only introduced by the differences among the parameters of the pretrained models. Furthermore, PosENet often reports positive SPC responses from no-padding models, as shown in its large standard deviation. In contrast, PPP has zero response to no-padding models by definition, and therefore is less biased for measuring the positional information from padding.

Although certain paddings seem to have slightly lower PPP-MAE than other paddings, in Table 3, we find the differences are not significant when comparing the extremely low PPP-MAE from most of the randomly initialized networks. In most cases, the network can effectively construct its PPP, even with the highly stochastic randn padding. The only exception seems to be the case of randn padding in the salient object detection (SOD) task, where the network fails to achieve a compatible performance with other paddings[1]. The results show that the model training plays an important role

[1]We use the same setting as PosENet to evaluates PiCANet (Liu et al., 2018) on the SOD task. PiCANet is initialized by a model pretrained on ImageNet (with zero padding). The discrepancy in the padding scheme can be the major cause of failure while training the network on SOD task with randn padding.

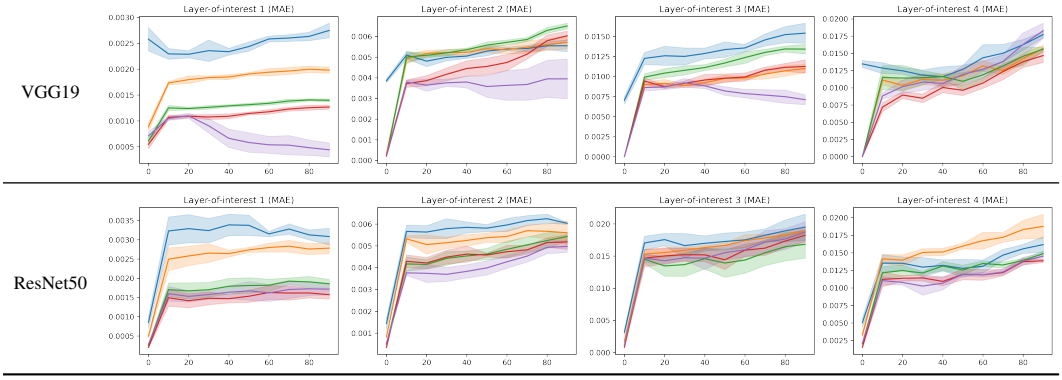

Figure 5: **Chronological PPP.** We quantify PPP every 10 epochs and plot its development in four different layer of depth (the rightmost layer is the one closest to model output). All curves consistently show a sudden surge at the early stage, and all the later layers are slowly but steadily gaining stronger PPP until the end of training. The shadow region represents standard deviations among 5 individual training episodes. The colors represent zeros, circular, reflect, replicate, and randn paddings.

in the formation of PPP, and perhaps its contribution is much larger than which underlying padding scheme is being used. This motivates us to further analyze the PPP formulation during model training.

4.3 Chronological PPP

To understand the formulation of PPP through time, we snapshot checkpoints every 10 epochs for all training episodes. By measuring the PPP-MAE at all the checkpoints, we plot a chronological curve and monitor the progress of PPP. We train 5 individual models for each pair of model-padding setting and report the standard deviations, which demonstrates the significance of the trend.

Figure 5 shows all models achieve a significant gain of PPP within the first 10 epochs in all intermediate layers. Most models continuously increase their PPP as training proceeds, especially in the fourth layer of interest, which is the last output from the convolutional layers before the final linear projection. Another interesting observation is that our randn padding, which is designed to be less easily detectable with built-in stochasticity, indeed shows less PPP built-up at the intermediate stages in certain layers. However, the network still adjusts the behavior and ends up forming complete PPPs at the fourth layer of interest in all scenarios. All these evidences show that the network builds PPP purposely as a favorable representation to assist its learning.

5 Conclusion and Limitations

In this paper, we develop a reliable method for measuring PPP and conduct a series of analyses toward understanding the formation and properties of PPP. Through a large-scale study, we demonstrate that PPP is a representation that the network favorably develops as a part of its learning process, and its formation has weak connections to the underlying padding algorithm. We show that reliable PPP metrics are important steps for understanding the effects of PPPs in different tasks, and useful for measuring the effectiveness of future methods in debiasing PPP.

However, an unfortunate and inevitable limitation of the PPP metrics is that their measure is biased by the model architecture and parameters. Since the PPP metrics are based on the distributional differences between the paired model outputs (i.e., optimal padding to algorithmic padding), different architecture and layers of depth exhibit different and intractable biases due to different interactions between PPP and model parameters. Such a bias makes PPP metrics less comparable while dissecting models with different architectures or parameter distributions (e.g., weight decay and weight normalization), which is important for studying the effect of architectural changes. However, this limitation is inevitable for any (and all existing) metric that attempts to measure PPP using the outputs of a model. We note future studies in measuring PPP without model inferences[2] will be an important step toward tackling and understanding the property of PPP under different architectural choices.

[2]A related analogy of the contradictory problem can be found in neural architecture search literature (Mellor et al., 2021), which aims to assess the final performance of architecture without training the model.

REFERENCES

Antonio Alguacil, Wagner Gonçalves Pinto, Michael Bauerheim, Marc C Jacob, and Stéphane Moreau. Effects of boundary conditions in fully convolutional networks for learning spatio-temporal dynamics. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2021. 1, 2, 3

Bilal Alsallakh, Narine Kokhlikyan, Vivek Miglani, Jun Yuan, and Orion Reblitz-Richardson. Mind the pad – {cnn}s can develop blind spots. In *International Conference on Learning Representations*, 2021a. 1, 2, 3, 8

Bilal Alsallakh, Vivek Miglani, Narine Kokhlikyan, David Adkins, and Orion Reblitz-Richardson. Are convolutional networks inherently foveated? In *SVRHM 2021 Workshop at NeurIPS*, 2021b. 1, 5

Jooyoung Choi, Jungbeom Lee, Yonghyun Jeong, and Sungroh Yoon. Toward spatially unbiased generative models. In *IEEE International Conference on Computer Vision*, 2021. 1, 3

Songwei Ge, Thomas Hayes, Harry Yang, Xi Yin, Guan Pang, David Jacobs, Jia-Bin Huang, and Devi Parikh. Long video generation with time-agnostic vqgan and time-sensitive transformer. *arXiv preprint arXiv:2204.03638*, 2022. 1, 2, 3

Carlo Innamorati, Tobias Ritschel, Tim Weyrich, and Niloy J Mitra. Learning on the edge: Investigating boundary filters in cnns. *International Journal of Computer Vision*, 2020. 1

Md Amirul Islam*, Sen Jia*, and Neil D. B. Bruce. How much position information do convolutional neural networks encode? In *International Conference on Learning Representations*, 2020. 1, 5

Md Amirul Islam, Matthew Kowal, Sen Jia, Konstantinos G. Derpanis, and Neil Bruce. Boundary effects in {cnn}s: Feature or bug? https://openreview.net/forum?id=M4qXqdw3xC, 2021a. 1

Md Amirul Islam, Matthew Kowal, Sen Jia, Konstantinos G Derpanis, and Neil DB Bruce. Position, padding and predictions: A deeper look at position information in cnns. *arXiv preprint arXiv:2101.12322*, 2021b. 1

Osman Semih Kayhan and Jan C van Gemert. On translation invariance in cnns: Convolutional layers can exploit absolute spatial location. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 1, 5

Kuangliu. pytorch-cifar. https://github.com/kuangliu/pytorch-cifar, 2017. 12

Chieh Hubert Lin, Yen-Chi Cheng, Hsin-Ying Lee, Sergey Tulyakov, and Ming-Hsuan Yang. InfinityGAN: Towards infinite-pixel image synthesis. In *International Conference on Learning Representations*, 2022. 1, 3, 6

Nian Liu, Junwei Han, and Ming-Hsuan Yang. Picanet: Learning pixel-wise contextual attention for saliency detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 8

Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *IEEE International Conference on Computer Vision*, 2019. 3

Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. In *Neural Information Processing Systems*, 2016. 1, 5

Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. In *International Conference on Machine Learning*, 2021. 9

Evangelos Ntavelis, Mohamad Shahbazi, Iason Kastanis, Radu Timofte, Martin Danelljan, and Luc Van Gool. Arbitrary-scale image synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 1, 3

Oskyhn. Cnns-without-borders. https://github.com/oskyhn/CNNs-Without-Borders, 2019. 12

Pytorch. vision. https://github.com/pytorch/vision, 2016. 12

Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501*, 2020. 3

Rui Xu, Xintao Wang, Kai Chen, Bolei Zhou, and Chen Change Loy. Positional encoding as spatial inductive bias in gans. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 1, 2, 3

Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 6

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 3

Supplementary Material

APPENDIX A IMPLEMENTATION DETAILS

A.1 ARCHITECTURE AND FEATURE ALIGNMENTS

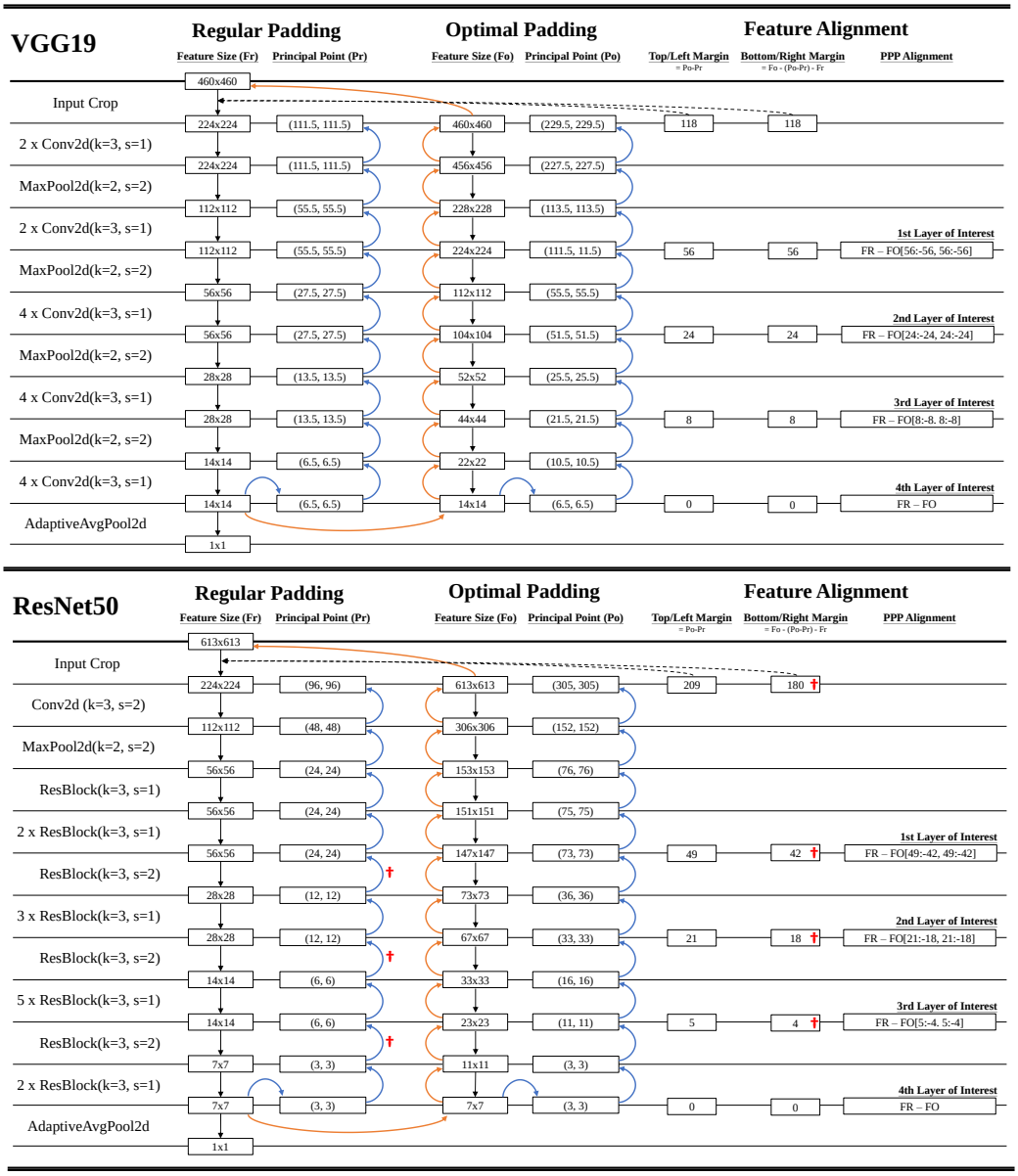

Figure 6: **The architecture for VGG19 and ResNet50 used in the paper.** We mark the calculation of optimal padding in orange arrows and principal point in blue arrows. We label the layers of interest that are used in the paper. The red † indicates where a principal point shift is identified.

A.2 PPP FEATURE MISALIGNMENT

There are several pitfalls in visualizing and quantifying PPP. We identify two critical pitfalls from the architectures we implemented. However, these may not be sufficient to cover all potential issues while integrated into other architectures. Therefore one must be alerted to any unusual behavior (e.g., Figure 2(d) in the main paper) throughout their implementation.

Principal point shifting. Conv2d has a hidden behavior that few people are aware of, the operation is one-pixel skewed while applying a stride-two Conv2d on even-shaped features. To understand how the one-pixel shift happens, we first define the principal point of a feature map. We first define the principal point of the last feature map as the center pixel (note that we define it as the middle-point between the center-two pixels in case the last feature size is even). Then, we recursively define the principal point of the $(N-1)$-th layer as the pixel that positions at the center of the Conv2d receptive field that mainly forms the principal point of the N-th layer. In the case of optimally-padded features, the principal points in every layer are the center of the feature map. But, as shown in Figure 2(a), the principal point of algorithmically-padded features will have a one-pixel shift when a stride-2 convolution is applied to even-shaped features, which can be further amplified as more layers stack up. Such a skew causes the principal points of algorithmically-padded features shift several pixels away from the principal points of optimally-padded features. As PPP metrics use pixel-wise subtraction to distinguish the image content from PPP, the misalignment becomes a critical issue, since the image contents are no longer aligned and subtractable.

In Figure 6, we show the procedure of calculating the principal point in blue arrows and marking the values impacted by principal point shift with red †. For the ResNet50 architecture, the principal point shift accumulates to $16 (= 224/2 - 96)$ pixels in the early layers.

Fortunately, such a displacement can be fixed by adding corrections to how we calculate the feature margins. As shown in Figure 2(b), the concept of the margin correction is to make the two principal points overlapping each other after adding the margin. In the example, the left-right margins are corrected to $(209, 180)$ (instead of the more intuitive choice of $(195, 194)$ or $(194.5, 194.6)$).

We also show how the principal point shift visually looking like in Figure 2(c), notice the patterns have right-bottom shifted 16 pixels. As shown in Figure 2(d), failing to identify the principal point shift will result in checkerboard artifacts while calculating PPP, and adding correction eliminates the artifacts.

Maxpooling misalignment. This is a hypothetical condition that may potentially happen but has not been observed in the three architectures we tested. Consider a case of a Maxpooling layer of window size 2 and stride 2, the sliding windows of each pooling operation have no overlap, therefore the initial index of the first sliding window solely determines the spatial location of all sliding windows. Accordingly, there is a chance that the initial condition of the optimally-padded features causes all of its sliding windows to be one-pixel misaligned to the algorithmically-padded features. Fortunately, the condition can be easily determined by calculating the top and left margins of the feature alignment (similar to the aforementioned principal point shift calculation). For the case of a Maxpooling layer of window size 2 and stride 2, the misalignment will not happen if the top and left margins are even numbers, and that is exactly the case for VGG19 and ResNet50, as shown in Figure 6.

A.3 RANDN PADDING

A critical implementation detail is that such a padding scheme must be applied before activation functions. Since the paddings are based on the distribution within sliding windows, activation functions such as ReLU, which clamps all negative values, can discard a significant amount of information beforehand. Instead of the traditional use of padding-convolution-normalization-activation, we modify the order to convolution-normalization-padding-activation. Note that such a change of order does not affect the behavior or results of other padding schemes.

A.4 ACKNOWLEDGING OPEN-SOURCE CONTRIBUTORS

Our implementation reuses codes from several open-source codebases, which greatly supports our development. The repositories used in the paper are F-Conv (Oskyhn, 2019), torchvision (Pytorch, 2016) and Pytorch-cifar (Kuangliu, 2017).

APPENDIX B MORE PPP VISUALIZATIONS

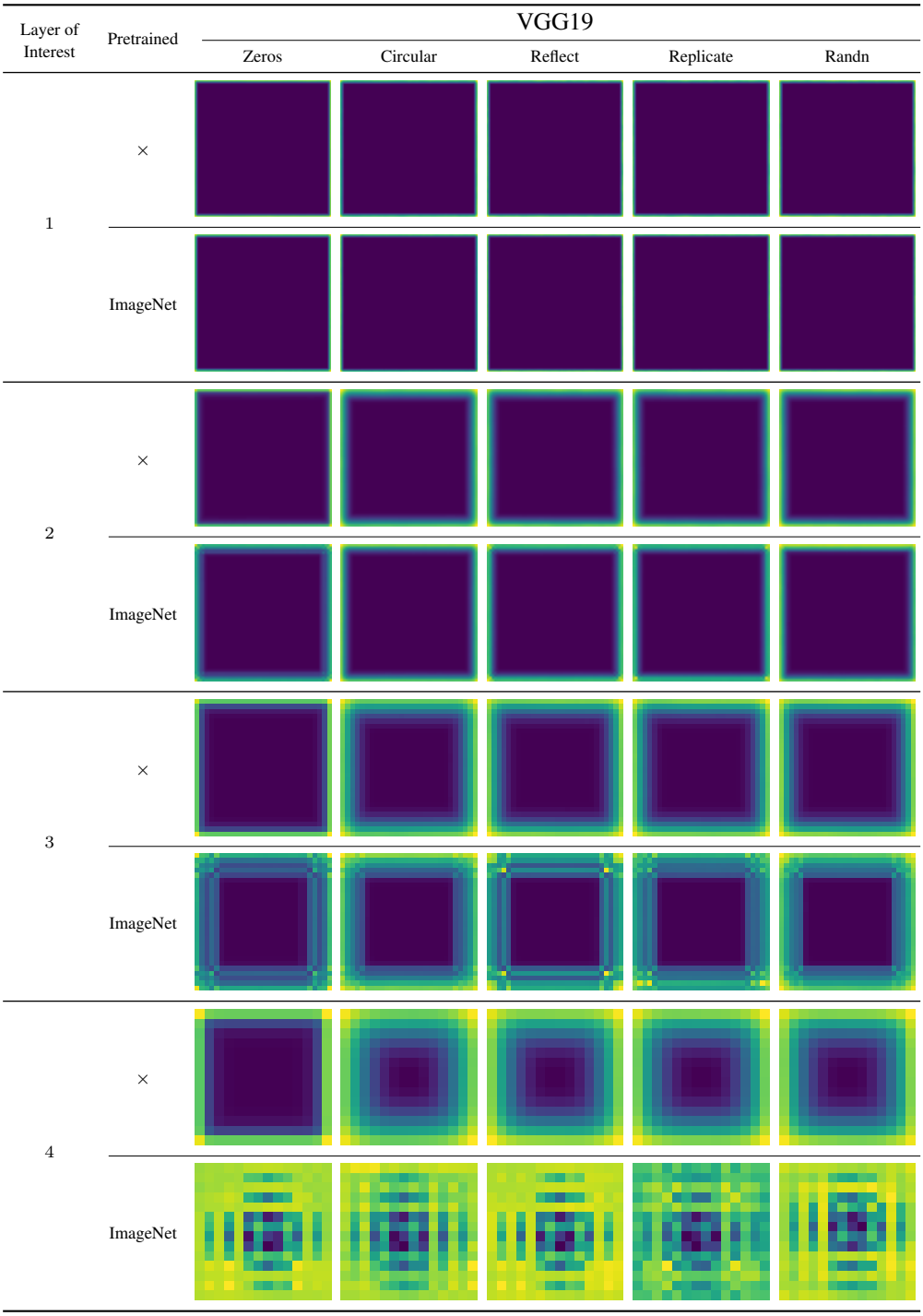

Figure 7: **Visualization of Position-Information Pattern from Padding (PPP).** The visualizations are calculated based on Eq. 3 over 480 GMap samples. The results show that the pretrained model significantly reinforces PPP compared to randomly initialized networks. Note that each image is normalized to $[0, 1]$ separately, therefore the colors between images are not comparable.

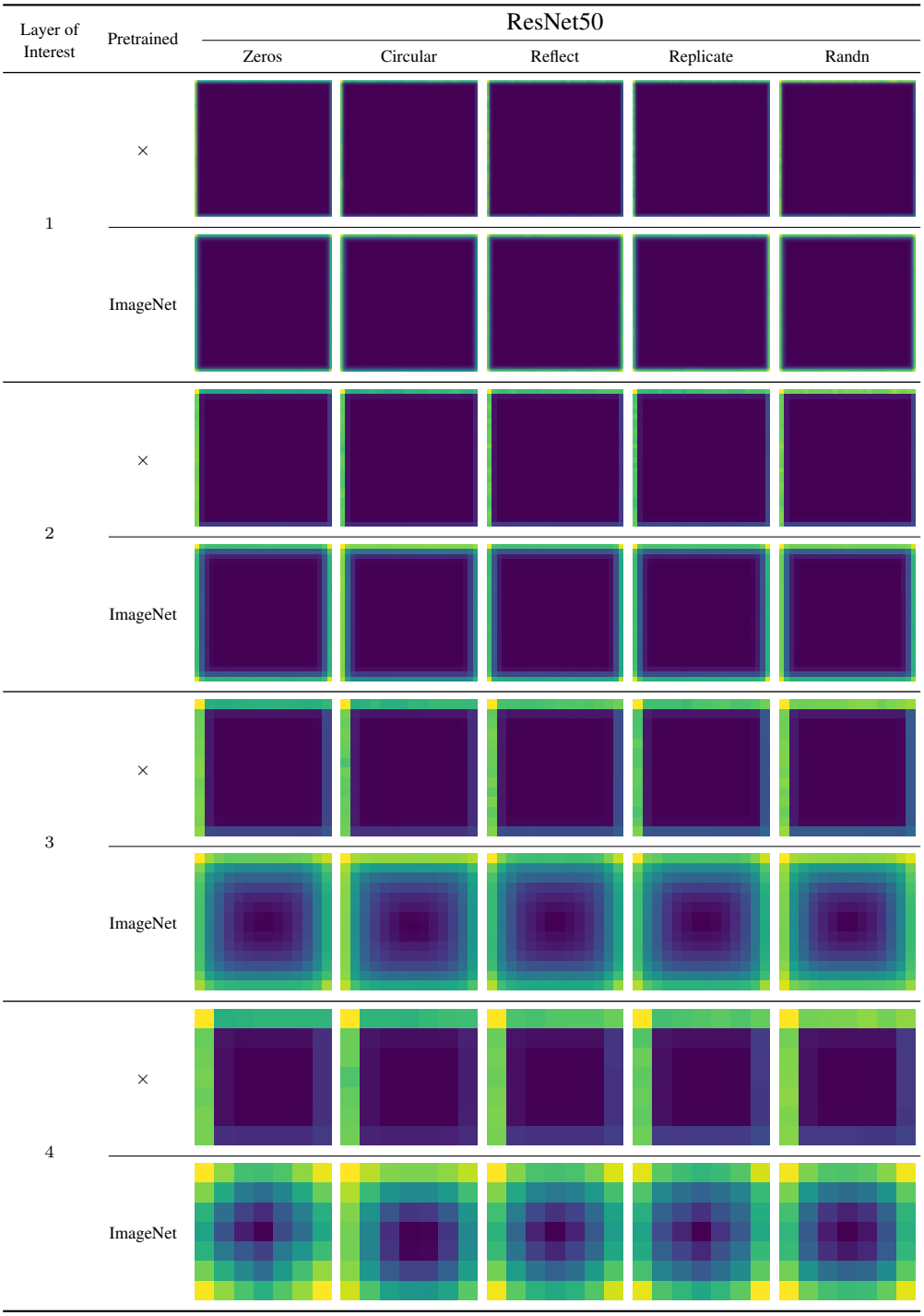

Figure 8: **Visualization of Position-Information Pattern from Padding (PPP).** The visualizations are calculated based on Eq. 3 over 480 GMap samples. The results show that the pretrained model significantly reinforces PPP compared to randomly initialized networks. Note that each image is normalized to $[0, 1]$ separately, therefore the colors between images are not comparable.

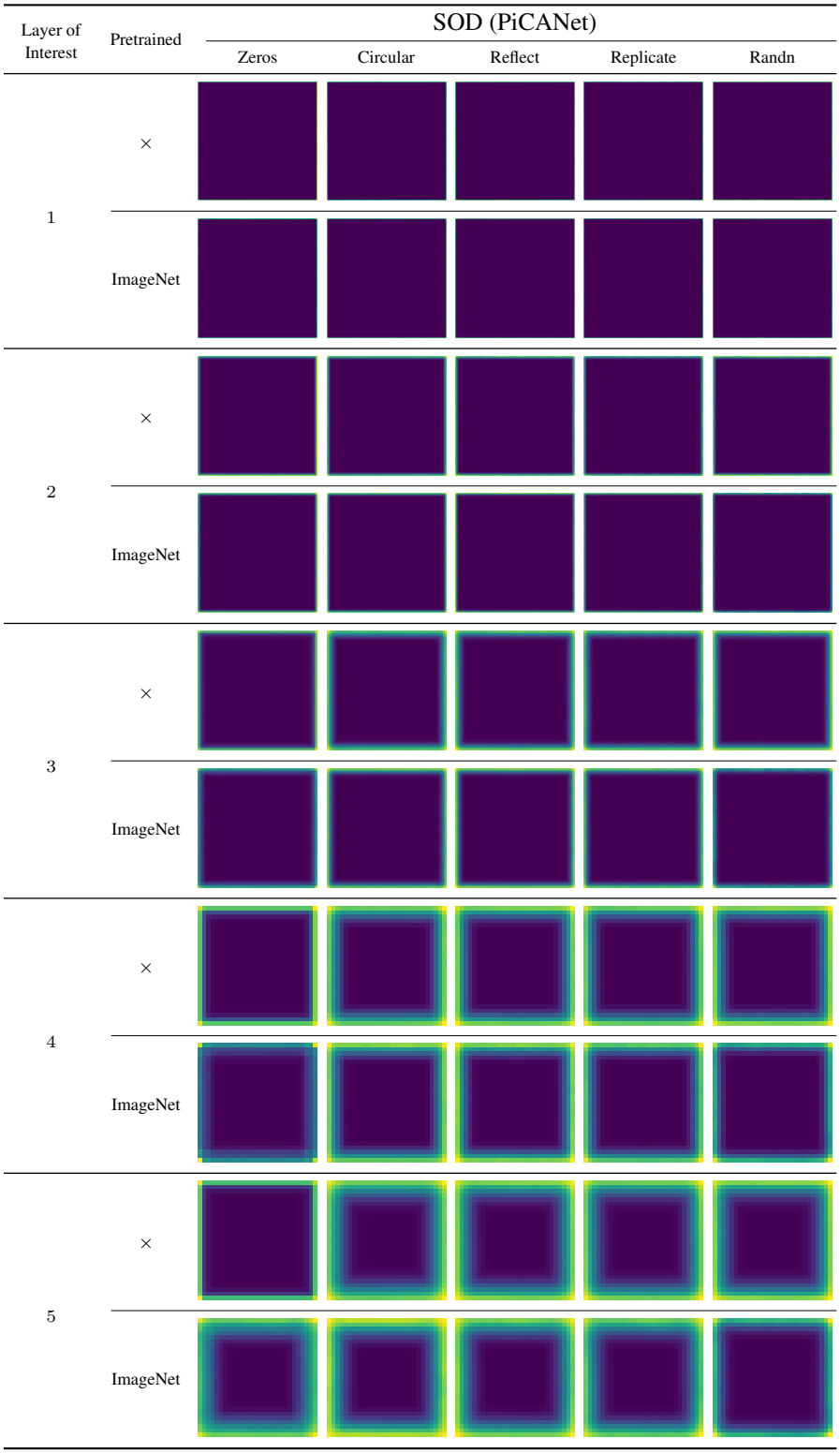

Figure 9: **Visualization of Position-Information Pattern from Padding (PPP).** The visualizations are calculated based on Eq. 3 over 480 GMap samples. The results show that the pretrained model significantly reinforces PPP compared to randomly initialized networks. Note that each image is normalized to $[0, 1]$ separately, therefore the colors between images are not comparable.