Making Convolutional Networks Shift-Invariant Again

Richard Zhang [1]

Abstract

Modern convolutional networks are not shift-invariant, as small input shifts or translations can cause drastic changes in the output. Commonly used downsampling methods, such as max-pooling, strided-convolution, and average-pooling, ignore the sampling theorem. The well-known signal processing fix is anti-aliasing by low-pass filtering before downsampling. However, simply inserting this module into deep networks leads to performance degradation; as a result, it is seldomly used today. We show that when integrated correctly, it is compatible with existing architectural components, such as max-pooling. The technique is general and can be incorporated across layer types and applications, such as image classification and conditional image generation. In addition to increased shift-invariance, we also observe, surprisingly, that anti-aliasing boosts accuracy in ImageNet classification, across several commonly-used architectures. This indicates that anti-aliasing serves as effective regularization. Our results demonstrate that this classical signal processing technique has been undeservingly overlooked in modern deep networks.

1. Introduction

When downsampling a signal, such an image, the textbook solution is to anti-alias by low-pass filtering the signal (Oppenheim et al., 1999; Gonzalez & Woods, 1992). Without it, high-frequency components of the signal alias into lower-frequencies. This phenomenon is commonly illustrated in movies, where wheels appear to spin backwards, known as the Stroboscopic effect, due to the frame rate not meeting the classical sampling criterion (Nyquist, 1928). Interestingly, most modern convolutional networks do not worry about anti-aliasing.

Early networks did employ a form of blurred-downsampling – average pooling (LeCun et al., 1990). However, ample empirical evidence suggests max-pooling provides stronger task performance (Scherer et al., 2010), leading to its widespread adoption. Unfortunately, max-pooling does not provide the same anti-aliasing capability, and a curious, recently uncovered phenomenon emerges – small shifts in the input can drastically change the output (Engstrom et al., 2019; Azulay & Weiss, 2018). As illustrated in Fig. 1, network outputs can oscillate depending on the input position.

Blurred-downsampling and max-pooling are commonly viewed as competing downsampling strategies (Scherer et al., 2010). However, we show that they are compatible. Our simple observation is that max-pooling is inherently composed of two operations: (1) evaluating the max operator densely and (2) naive subsampling. We propose to low-pass filter between them as a means of anti-aliasing. This viewpoint enables low-pass filtering to augment, rather than replace max-pooling. As a result, shifts in the input leave the output relatively unaffected (shift-invariance) and more closely shift the internal feature maps (shift-equivariance).

Furthermore, this enables proper placement of the low-pass filter, directly before subsampling. With this methodology, practical anti-aliasing can be achieved with any existing strided layer, such as strided-convolution.

A potential concern is that overaggressive filtering can result in heavy loss of information, degrading performance. However, with a reasonable selection of low-pass filter weights, we actually observe *increased* absolute performance in ImageNet classification (Russakovsky et al., 2015), across architectures. We also test on an image-to-image translation task, where generating high-frequency content is critical for high-quality results. In summary, our contributions are:

- We integrate classic anti-aliasing filtering to improve shift-equivariance/invariance of deep networks. Critically, the method is compatible with any existing downsampling strategy, such as max-pooling.
- We validate on common downsampling strategies – max-pooling, average-pooling, strided-convolution – in different architectures. We test across multiple tasks – image classification and image-to-image translation.
- For ImageNet classification, we find, surprisingly, that accuracy increases, indicating effective regularization.

[1] Adobe Research, San Francisco, CA. Correspondence to: Richard Zhang <rizhang@adobe.com>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

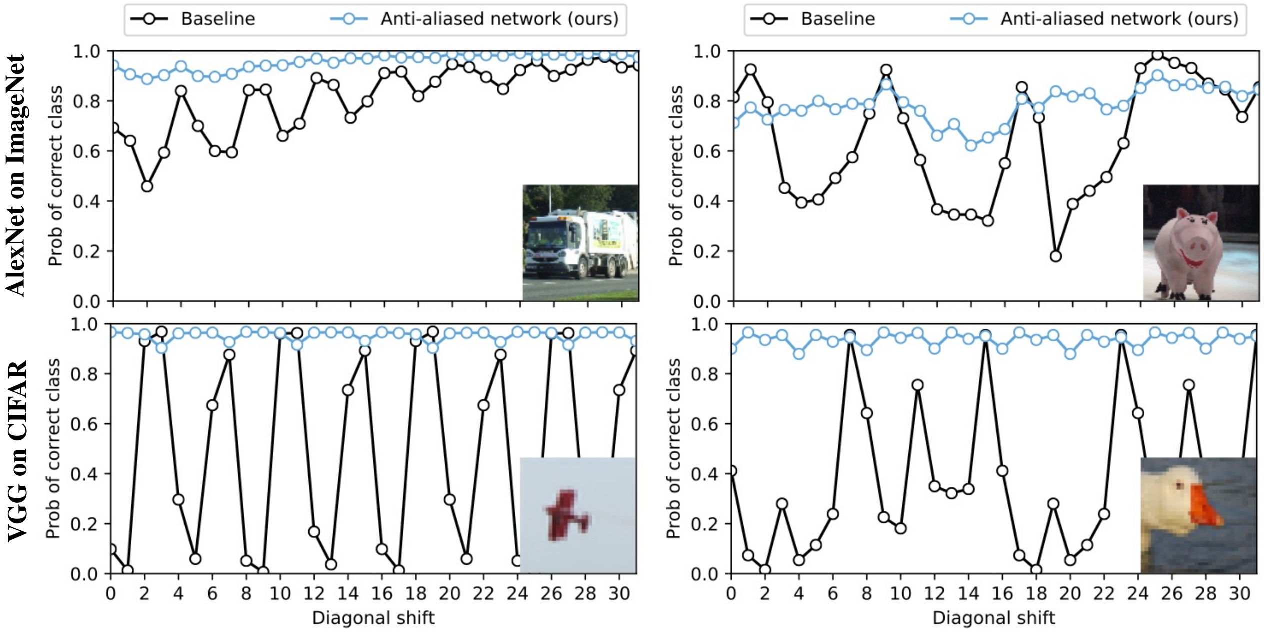


Figure 1. **Classification stability for selected images.** Predicted probability of the correct class changes when shifting the image. The baseline (black) exhibits chaotic behavior, which is stabilized by our method (blue). We find this behavior across networks and datasets. Here, we show selected examples using an AlexNet architecture on ImageNet **(top)** and a VGG architecture on CIFAR10 **(bottom)**. Code and anti-aliased versions of popular networks are available at `https://richzhang.github.io/antialiased-cnns/`.

2. Related Work

Local connectivity and weight sharing have been a central tenet of neural networks, including the Neocognitron (Fukushima & Miyake, 1982), LeNet (LeCun et al., 1998) and modern networks such as Alexnet (Krizhevsky et al., 2012), VGG (Simonyan & Zisserman, 2015), ResNet (He et al., 2016), and DenseNet (Huang et al., 2017). In biological systems, local connectivity was famously discovered in a cat's visual system (Hubel & Wiesel, 1962). Recent work has strived to add additional invariances, such as rotation, reflection, and scaling (Sifre & Mallat, 2013; Bruna & Mallat, 2013; Kanazawa et al., 2014; Cohen & Welling, 2016; Worrall et al., 2017; Esteves et al., 2018). We focus on shift-invariance, which is often taken for granted.

Though different properties have been engineered into networks, what factors and invariances does an emergent representation actually learn? Qualitative analysis of deep networks have included showing patches which activate hidden units (Girshick et al., 2014; Zhou et al., 2015), actively maximizing hidden units (Mordvintsev et al., 2015), and mapping features back into pixel space (Zeiler & Fergus, 2014; Hénaff & Simoncelli, 2016; Mahendran & Vedaldi, 2015; Dosovitskiy & Brox, 2016a;b; Nguyen et al., 2017). Our analysis is focused on a specific, low-level property and is complementary to these approaches.

A more quantitative approach for analyzing networks is measuring representation or output changes (or robustness to changes) in response to manually generated perturbations to the input, such as image transformations (Goodfellow et al., 2009; Lenc & Vedaldi, 2015; Azulay & Weiss, 2018), geometric transforms (Fawzi & Frossard, 2015; Ruderman et al., 2018), and CG renderings with various shape, poses, and colors (Aubry & Russell, 2015). A related line of work is adversarial examples, where input perturbations are purposely directed to produce large changes in the output. These perturbations can be on pixels (Goodfellow et al., 2014a;b), a single pixel (Su et al., 2019), small deformations (Xiao et al., 2018), or even affine transformations (Engstrom et al., 2019). We aim to make the network robust to the simplest of these types of attacks and perturbations: shifts.

Classic hand-engineered computer vision and image processing representations, such as SIFT (Lowe, 1999), wavelets, and image pyramids (Adelson et al., 1984; Burt & Adelson, 1987) also extract features in a sliding window manner, often with some subsampling factor. As discussed in Simoncelli et al. (1992), literal shift-equivariance cannot hold when subsampling. Shift-equivariance can be recovered if features are extracted densely, for example textons (Leung & Malik, 2001), the Stationary Wavelet Transform (Fowler, 2005), and DenseSIFT (Vedaldi & Fulkerson, 2008). Deep networks can also be evaluated densely, by removing striding and making appropriate changes to subsequent layers by using *á trous*/dilated convolutions (Chen et al., 2015; 2018; Yu & Koltun, 2016; Yu et al., 2017). This comes at great computation and memory cost. Our work

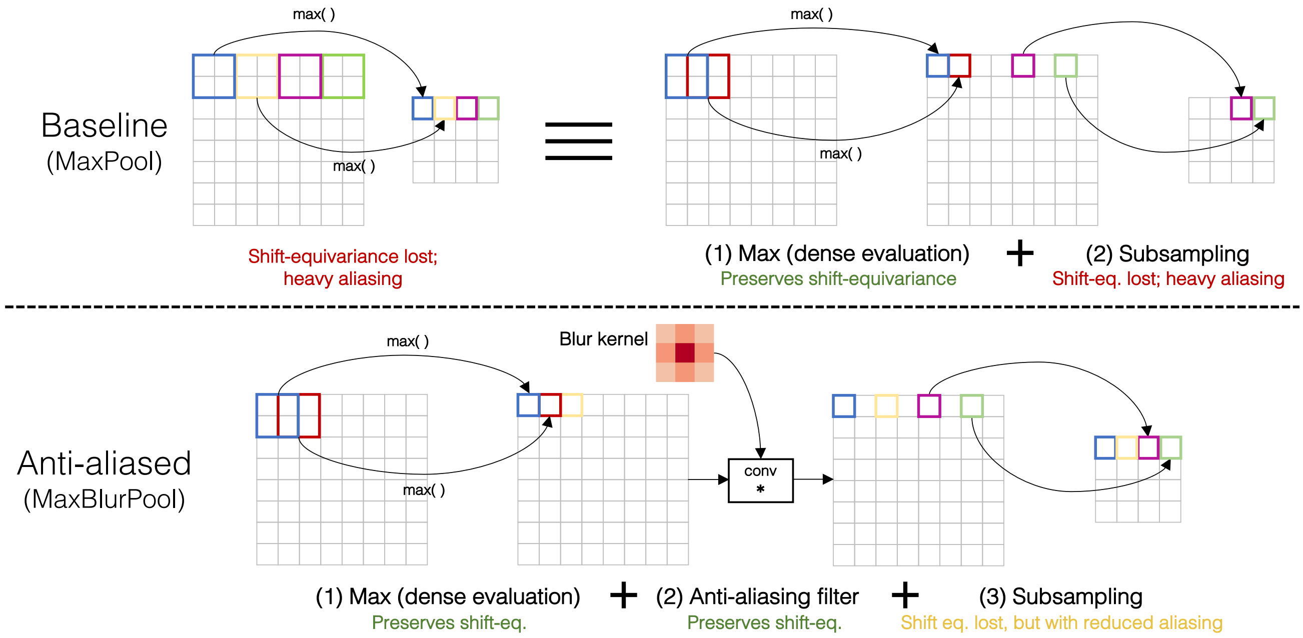


Figure 2. **Anti-aliasing convolutional networks. (Top)** Pooling does not preserve shift-equivariance. It is functionally equivalent to densely evaluated pooling followed by subsampling. The latter ignores the Nyquist sampling theorem and loses shift-equivariance. **(Bottom)** We low-pass filter between the operations. This keeps the first operation, while anti-aliasing the appropriate signal. This equivalent analysis and modification can be applied to any strided layer – we experiment with **MaxPool→MaxBlurPool** (shown above), **StridedConv→ConvBlurPool**, and **AvgPool→BlurPool**.

investigates improving shift-equivariance with minimal additional computation, by blurring before subsampling.

Early networks employed average pooling (LeCun et al., 1990), which is equivalent to blurred-downsampling with a box filter. However, work (Scherer et al., 2010) has found max-pooling to be more effective, which has consequently become the predominant method for downsampling. While previous work (Scherer et al., 2010; Hénaff & Simoncelli, 2016; Azulay & Weiss, 2018) acknowledges the drawbacks of max-pooling and benefits of blurred-downsampling, they are viewed as separate, discrete choices, preventing their combination. Interestingly, Lee et al. (2016) does not explore low-pass filters, but does propose to softly gate between max and average pooling. However, this does not fully utilize the anti-aliasing capability of average pooling.

Mairal et al. (2014) derive a network architecture, motivated by translation invariance, named Convolutional Kernel Networks. While theoretically interesting (Bietti & Mairal, 2017), CKNs perform at lower accuracy than contemporaries, resulting in limited usage. Interestingly, a byproduct of the derivation is a standard Gaussian filter; however, no guidance is provided on its proper integration with existing network components. Instead, we demonstrate practical integration with any strided layer, and empirically show performance increases on a challenging benchmark – ImageNet classification – on widely-used networks.

3. Methods

3.1. Preliminaries

Deep convolutional networks as feature extractors Let an image with resolution $H \times W$ be represented by $X \in \mathbb{R}^{H \times W \times 3}$. An L-layer CNN can be expressed as a feature extractor $\mathcal{F}_l(X) \in \mathbb{R}^{H_l \times W_l \times C_l}$, with layer $l \in \{0, 1, ..., L\}$, spatial resolution $H_l \times W_l$ and C_l channels. Each feature map can also be upsampled to original resolution, $\widetilde{\mathcal{F}}_l(X) \in \mathbb{R}^{H \times W \times C_l}$.

Shift-equivariance and invariance A function $\widetilde{\mathcal{F}}$ is shift-equivariant if shifting the input equally shifts the output, meaning shifting and feature extraction are commutable.

$$\text{Shift}_{\Delta h, \Delta w}(\widetilde{\mathcal{F}}(X)) = \widetilde{\mathcal{F}}(\text{Shift}_{\Delta h, \Delta w}(X)) \quad \forall\, (\Delta h, \Delta w) \tag{1}$$

A representation is shift-invariant if shifting the input results in an *identical* representation.

$$\widetilde{\mathcal{F}}(X) = \widetilde{\mathcal{F}}(\text{Shift}_{\Delta h, \Delta w}(X)) \quad \forall\, (\Delta h, \Delta w) \tag{2}$$

Periodic-N shift-equivariance/invariance In some cases, the definitions in Eqns. 1, 2 may hold only when shifts $(\Delta h, \Delta w)$ are integer multiples of N. We refer to such scenarios as periodic shift-equivariance/invariance. For example, periodic-2 shift-invariance means that even-pixel shifts produce an identical output, but odd-pixel shifts may not.

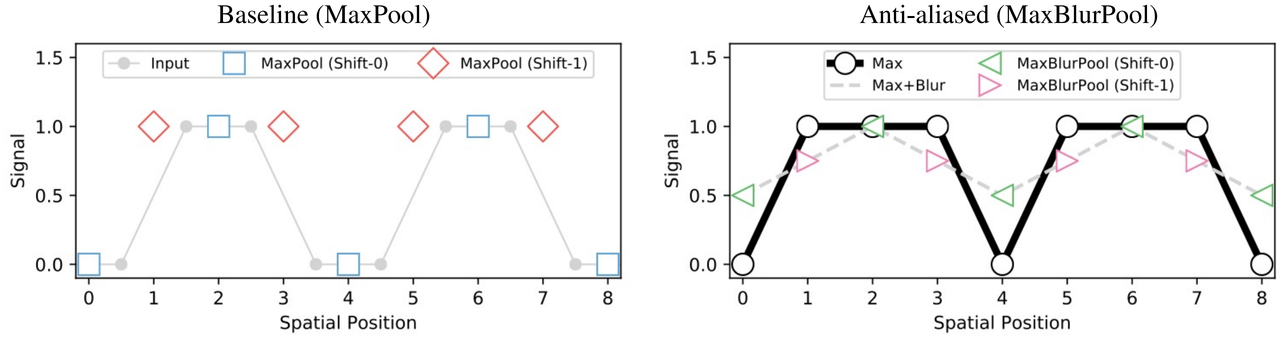


Figure 3. **Illustrative 1-D example of sensitivity to shifts.** We illustrate how downsampling affects shift-equivariance with a toy example. **(Left)** An input signal is in light gray line. Max-pooled ($k = 2$, $s = 2$) signal is in blue squares. Simply shifting the input and then max-pooling provides a completely different answer (red diamonds). **(Right)** The blue and red points are subsampled from a densely max-pooled ($k = 2$, $s = 1$) intermediate signal (**thick black line**). We low-pass filter this intermediate signal and then subsample from it, shown with green and magenta triangles, better preserving shift-equivariance.

Circular convolution and shifting Edge artifacts are an important consideration. When shifting, information is lost on one side and has to be filled in on the other.

In our CIFAR10 classification experiments, we use circular shifting and convolution. When the convolutional kernel hits the edge, it "rolls" to the other side. Similarly, when shifting, pixels are rolled off one edge to the other.

$$[\text{Shift}_{\Delta h,\Delta w}(X)]_{h,w,c} = X_{(h-\Delta h)\%H,(w-\Delta w)\%W,c}\ , \quad \text{where } \% \text{ is the modulus function} \tag{3}$$

The modification minorly affects performance and could be potentially mitigated by additional padding, at the expense of memory and computation. But importantly, this affords us a clean testbed. Any loss in shift-equivariance is purely due to characteristics of the feature extractor.

An alternative is to take a shifted crop from a larger image. We use this approach for ImageNet experiments, as it more closely matches standard train and test procedures.

3.2. Anti-aliasing to improve shift-equivariance

Conventional methods for reducing spatial resolution – max-pooling, average pooling, and strided convolution – all break shift-equivariance. We start by analyzing max-pooling, the predominant downsampling method in deep networks.

MaxPool→MaxBlurPool Consider the example $[0, 0, 1, 1, 0, 0, 1, 1]$ signal in Fig. 3 (left). Max-pooling (kernel k=2, stride s=2) will result in $[0, 1, 0, 1]$. Simply shifting the input results in a dramatically different answer of $[1, 1, 1, 1]$. Shift-equivariance is lost. These results are subsampling from an intermediate signal – the input densely max-pooled (stride-1), which we simply refer to as "max". As illustrated in Fig. 2 (top), we can write max-pooling as a composition of two functions: $\text{MaxPool}_{k,s} = \text{Subsample}_s \circ \text{Max}_k$.

The Max operation preserves shift-equivariance, as it is densely evaluated in a sliding window fashion, but subsequent subsampling does not. We simply propose to add an anti-aliasing filter with kernel $m \times m$, denoted as Blur_m as shown in Fig. 3 (right). During implementation, blurring and subsampling are combined, as commonplace in image processing. We call this function $\text{BlurPool}_{m,s}$.

$$\begin{aligned} \text{MaxPool}_{k,s} &\rightarrow \text{Subsample}_s \circ \text{Blur}_m \circ \text{Max}_k \\ &= \text{BlurPool}_{m,s} \circ \text{Max}_k \end{aligned} \tag{4}$$

Sampling after low-pass filtering gives $[.5, 1, .5, 1]$ and $[.75, .75, .75, .75]$. These are closer to each other and better representations of the intermediate signal.

StridedConv→ConvBlurPool Strided-convolutions suffer from the same issue, and the same method applies.

$$\text{Relu} \circ \text{Conv}_{k,s} \rightarrow \text{BlurPool}_{m,s} \circ \text{Relu} \circ \text{Conv}_{k,1} \tag{5}$$

Importantly, this analogous modification applies conceptually to any strided layer, meaning the network designer can keep their original operation of choice.

AveragePool→BlurPool Blurred downsampling with a box filter is the same as average pooling. Replacing it with a stronger filter provides better shift-equivariance. We examine such filters next.

$$\text{AvgPool}_{k,s} \rightarrow \text{BlurPool}_{m,s} \tag{6}$$

Anti-aliasing filter selection The method allows for a choice of blur kernel. We test filters ranging from size 2×2 to 7×7, with increasing smoothing. Weights are normalized to sum to 1.

- ***Dirac Delta*** (baseline) [1]: equivalent to subsampling and leaves the input unchanged
- ***Rectangle-2*** [1, 1]: often referred to as a moving average or box filter
- ***Triangle-3*** [1, 2, 1]: two box filters convolved together

Method	Filter	VGG13-bn			
		Train w/o aug		Train w/ aug	
		Acc	Con	Acc	Con
Baseline	**Delta-1**	91.6	88.1	**93.8**	96.6
Anti-Aliased	**Rect-2**	92.8	90.5	93.7	97.6
	Tri-3	93.1	93.9	93.6	98.0
	Bin-4	93.0	93.2	93.2	98.1
	Bin-5	**93.2**	96.3	93.2	98.4
	Bin-6	93.0	96.9	93.4	98.6
	Bin-7	93.0	**98.1**	93.2	**98.8**

Table 1. **CIFAR Classification.** We evaluate accuracy (Acc) and consistency (Con), using progressively larger anti-aliasing filters. We evaluate the network, training both without and with shift-based data augmentation. Results are plotted in Fig. 4.

- ***Binomial-4, 5, 6, 7***: the box filter convolved with itself repeatedly. For example, *Bin-5* [1, 4, 6, 4, 1] is the standard filter used in Laplacian pyramids (Burt & Adelson, 1987).

4. Experiments

4.1. Testbeds

CIFAR Classification To begin, we test classification of low-resolution 32 × 32 images. The dataset contains 50k training and 10k validation images, classified into one of 10 categories. We dissect the VGG architecture (Simonyan & Zisserman, 2015), showing that shift-equivariance is a signal-processing property, progressively lost in each downsampling layer.

ImageNet Classification We then test on large-scale classification on 224 × 224 resolution images. The dataset contains 1.2M training and 50k validation images, classified into one of 1000 categories. We test across 4 different architectures – AlexNet (Krizhevsky & Hinton, 2009), VGG16 (Simonyan & Zisserman, 2015), ResNet50 (He et al., 2016), and DenseNet121 (Huang et al., 2017) – with different downsampling strategies, as described in Tab. 2.

Conditional Image Generation Finally, we show that the same aliasing issues in classification networks are also present in conditional image generation networks. We test on the Labels→Facades (Tyleček & Šára, 2013; Isola et al., 2017) dataset, where a network is tasked to generated a 256×256 photorealistic image from a label map. There are 400 training and 100 validation images.

4.2. Shift-Invariance/Equivariance Metrics

Ideally, a shift in the input would result in equally shifted feature maps internally:

Internal feature distance. We examine internal feature maps with $d(\text{Shift}_{\Delta h,\Delta w}(\widetilde{\mathcal{F}}(X)), \widetilde{\mathcal{F}}(\text{Shift}_{\Delta h,\Delta w}(X)))$ (left & right-hand sides of Eqn. 1). We use cosine distance, as common for deep features (Kiros et al., 2015; Zhang et al., 2018).

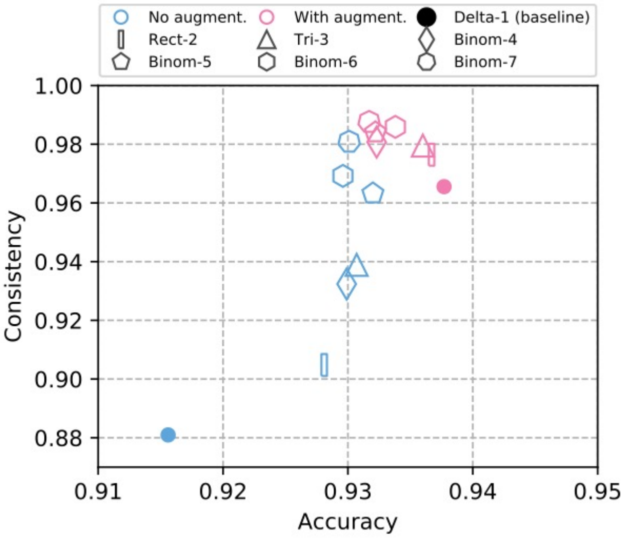


Figure 4. **CIFAR classification consistency vs accuracy.** Up (more consistent) and to the right (more accurate) is better. Number of sides corresponds to number of filter taps used (e.g., diamond for 4-tap filter); colors correspond to filters trained without (blue) and with (pink) shift-based data augmentation, using various filters. We show accuracy for no shift when training without shifts, and a random shift when training with shifts.

We can also measure the stability of the output:

Classification consistency. For classification, we check how often the network outputs the same classification, given the same image with two different shifts: $\mathbb{E}_{X,h_1,w_1,h_2,w_2}\mathbb{1}\{\arg\max P(\text{Shift}_{h_1,w_1}(X)) = \arg\max P(\text{Shift}_{h_2,w_2}(X))\}$.

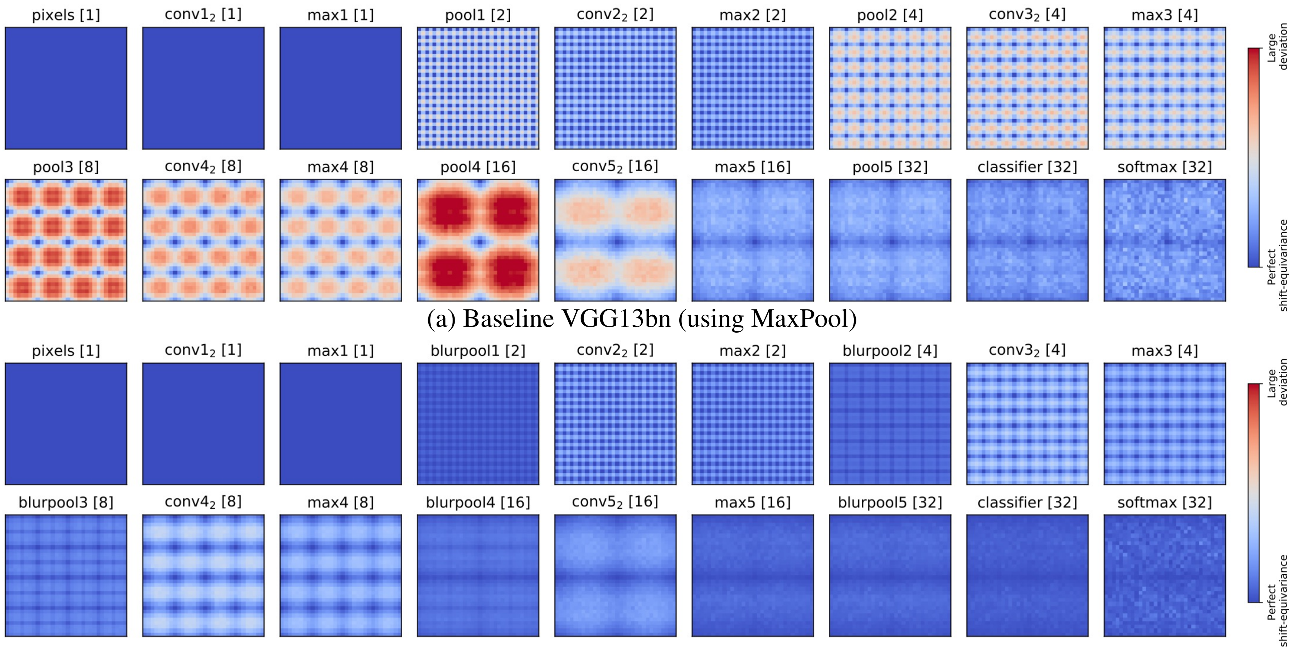
Generation stability. For image translation, we test if a shift in the input image generates a correspondingly shifted output. For simplicity, we test horizontal shifts. $\mathbb{E}_{X,\Delta w}\text{PSNR}\big(\text{Shift}_{0,\Delta w}(\mathcal{F}(X))), \mathcal{F}(\text{Shift}_{0,\Delta w}(X))\big)$.

4.3. Classification

4.3.1. CIFAR WITH VGG

We first test on the CIFAR dataset using the VGG13-bn (Simonyan & Zisserman, 2015) architecture. We train both without and with shift-based data augmentation. We evaluate on classification accuracy and consistency. The results are shown in Tab. 1 and Fig. 4.

Training without data augmentation Without the benefit of seeing shifts at training time, the baseline network produces inconsistent classifications – random shifts of the same image only agree 88.1% of the time. Our anti-aliased network, with the MaxBlurPool operator, increases consis-



(a) Baseline VGG13bn (using MaxPool)

(b) Anti-aliased VGG13bn (using MaxBlurPool, *Bin-5*)

Figure 5. **Deviation from perfect shift-equivariance, throughout VGG.** Feature distance between left & right-hand sides of the shift-equivariance condition (Eqn 1). Each pixel in each heatmap is a shift $(\Delta h, \Delta w)$. Blue indicates perfect shift-equivariance; red indicates large deviation. Note that the dynamic ranges of distances are different per layer. For visualization, we calibrate by calculating the mean distance between two different images, and mapping red to half the value. Accumulated downsampling factor is in [brackets]; in layers `pool5`, `classifier`, and `softmax`, shift-equivariance and shift-invariance are equivalent, as features have no spatial extent. Layers up to `max1` have perfect equivariance, as no downsampling yet occurs. **(a)** On the **baseline network**, shift-equivariance is reduced each time downsampling takes place. Periodic-N shift-equivariance holds, with N doubling with each downsampling. **(b)** With our **antialiased network**, shift-equivariance is better maintained, and the resulting output is more shift-invariant.

tency. The larger the filter, the more consistent the output classifications. This result agrees with our expectation and theory – improving shift-equivariance throughout the network should result in more consistent classifications across shifts, even when such shifts are not seen at training.

In this regime, accuracy clearly increases with consistency, as seen with the blue images in Fig. 4. Filtering does not destroy the signal or make learning harder. On the contrary, shift-equivariance serves as "built-in" augmentation, indicating more efficient data usage.

Training with data augmentation In principle, networks can *learn* to be shift-invariant from data. Is data augmentation all that is needed to achieve shift-invariance? By applying the *Rect-2* filter, a large increase in consistency, 96.6 → 97.6, can be had at a small decrease in accuracy 93.8 → 93.7. Even when seeing shifts at training, antialiasing increases consistency. From there, stronger filters can increase consistency, at the expense of accuracy.

Internal shift-equivariance We further dissect the progressive loss of shift-equivariance by investigating the VGG architecture internally. The network contains 5 blocks of convolutions, each followed by max-pooling (with stride 2), followed by a linear classifier. For purposes of our understanding, MaxPool layers are broken into two components – before and after subsampling, e.g., `max1` and `pool1`, respectively. In Fig. 5 (top), we show internal feature distance, as a function of all possible shift-offsets $(\Delta h, \Delta w)$ and layers. All layers before the first downsampling, `max1`, are shift-equivariant. Once downsampling occurs in `pool1`, shift-equivariance is lost. However, periodic-N shift-equivariance still holds, as indicated by the stippling pattern in `pool1`, and each subsequent subsampling doubles the factor N.

In Fig. 5 (bottom), we plot shift-equivariance maps with our anti-aliased network, using MaxBlurPool. Shift-equivariance is clearly better preserved. In particular, the severe drop-offs in downsampling layers do not occur. Improved shift-equivariance throughout the network cascades into more consistent classifications in the output, as seen by some selected examples are in Fig. 1 (bottom). The plot is made with a *Bin-5* filter, trained without data augmentation. The same trends hold for other filters and when training with augmentation.

Additional analysis We present additional analysis in the supplementary material – using DenseNet (Huang et al., 2017), timing analysis, investigating how learned convolutional layers change in an anti-aliased net, swapping Max

	ImageNet Classification				Generation
	Alex-Net	VGG-16	Res-Net50	Dense-Net121	U-Net
StridedConv	1$^\diamond$	–	4‡	1‡	8
MaxPool	3	5	1	1	–
AvgPool	–	–	–	3	–

Table 2. **Testbeds.** We test across tasks (ImageNet classification and Labels→Facades) and network architectures. Each architecture employs different downsampling strategies. We list how often each is used here. We can antialias each variant. $^\diamond$This convolution uses stride 4 (all others use 2). We only apply the antialiasing at stride 2. Evaluating the convolution at stride 1 would require large computation at full-resolution. ‡For the same reason, we do not antialias the first strided-convolution in these networks.

Filter	ImageNet Classification							
	AlexNet		VGG16		ResNet50		DenseNet121	
	Acc	Con	Acc	Con	Acc	Con	Acc	Con
Baseline	56.5	78.2	71.6	88.5	76.2	89.2	74.4	88.8
Rect-2	**57.2**	81.3	72.2	89.2	76.8	90.0	75.0	89.5
Tri-3	56.9	82.2	72.2	89.6	76.8	90.9	**75.1**	89.8
Bin-5	56.6	**82.5**	**72.3**	**90.2**	**77.0**	**91.3**	75.0	**90.4**

Table 3. **Imagenet Classification.** We show 1000-way classification accuracy and consistency (higher is better), across 4 architectures, with our anti-aliasing filtering added. We test 3 filters, in addition to the off-the-shelf models. As designed, classification consistency is improved across all methods. Interestingly, accuracy is *also improved*. We recommend *Tri-3* or *Bin-5* for general use.

and Blur operations, combining Max and Blur in parallel rather than in series, a negative result on learning the blur filter, and robustness against a shift-based adversarial attacker.

4.3.2. Large-scale ImageNet classification

We next test on large-scale image classification of ImageNet (Russakovsky et al., 2015). In Tab. 3, we show classification accuracy and consistency, across several architectures – AlexNet, VGG16, Resnet50, and DenseNet121. The off-the-shelf networks are labeled as "baseline", and we use standard training schedules from the publicly available PyTorch (Paszke et al., 2017) repository for our anti-aliased networks. Each architecture has a different downsampling strategy, shown in Tab. 2. For example, VGG exclusively uses MaxPooling, whereas DenseNet uses a StridedConv, a MaxPool, and 3 AvgPools. The consistency and accuracy of baseline and anti-aliased networks are visualized in Fig. 6.

Improved shift-invariance We apply progressively stronger filters – *Rect-2*, *Tri-3*, *Bin-5* – to the 4 architectures. Using a small *Rect-2* filter increases consistency by +3.15, +0.72, +0.76, and +0.72%, across AlexNet, VGG, ResNet, and DenseNet, respectively. The *Bin-5* filter increases by +4.33, +1.67, +2.11, and +1.58%. As expected, the larger the filter, the more shift-invariant the output.

Classification performance Filtering improves the shift-invariance. How does it affect absolute classification performance? We find that across the board, *performance actually modestly increases*. A *Rect-2* filter increases AlexNet by +0.69%. *Bin-5* increases VGG16 and Resnet to +0.74% and +0.88%, respectively. *Tri-3* increases DenseNet by +0.71%. While certain filters work better for certain networks, of the 12 combinations of filters and networks we examined, none reduce accuracy and 10 actually improve accuracy by at least +0.5% (all improve consistency). This is a surprising, unexpected result, as low-pass filtering removes information, and could be expected to reduce performance. On the contrary, we find that it serves as effective regularization, and these widely-used methods improve with simple anti-aliasing. As ImageNet-trained nets often serve as the backbone for downstream tuning, this potential improvement may be observed across other applications as well.

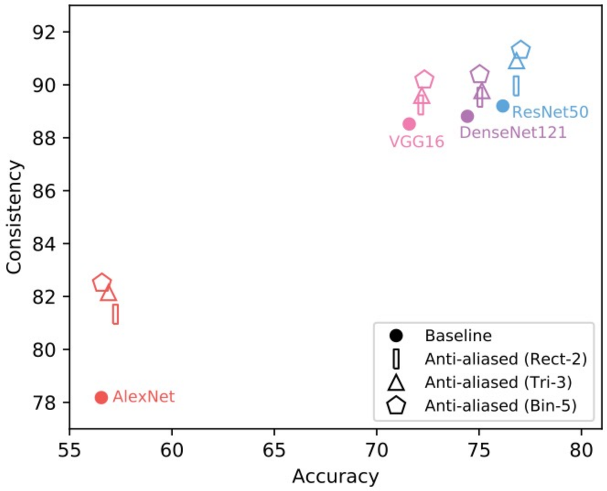


Figure 6. **ImageNet Classification consistency vs. accuracy.** Up (more consistent) and to the right (more accurate) is better. Different shapes correspond to the baseline (circle) or variants of our anti-aliased networks (bar, triangle, pentagon for length 2, 3, 5 filters, respectively). We test across network architectures. As expected, low-pass filtering helps shift-invariance. Surprisingly, classification accuracy is also improved.

We recommend using the *Tri-3* or *Bin-5* filter. If shift-invariance is especially desired, stronger filters can be used.

4.4. Conditional image generation (Label→Facades)

We test on image generation, outputting an image of a facade given its semantic label map (Tyleček & Šára, 2013).

Baseline We use the pix2pix method (Isola et al., 2017). The method uses U-Net (Ronneberger et al., 2015), which contains 8 downsampling and 8 upsampling layers, with skip connections to preserve local information. No anti-aliasing filtering is applied in down or upsampling layers in the baseline. In Fig. 7, we show a qualitative example, focusing in on a specific window. In the baseline (top), as

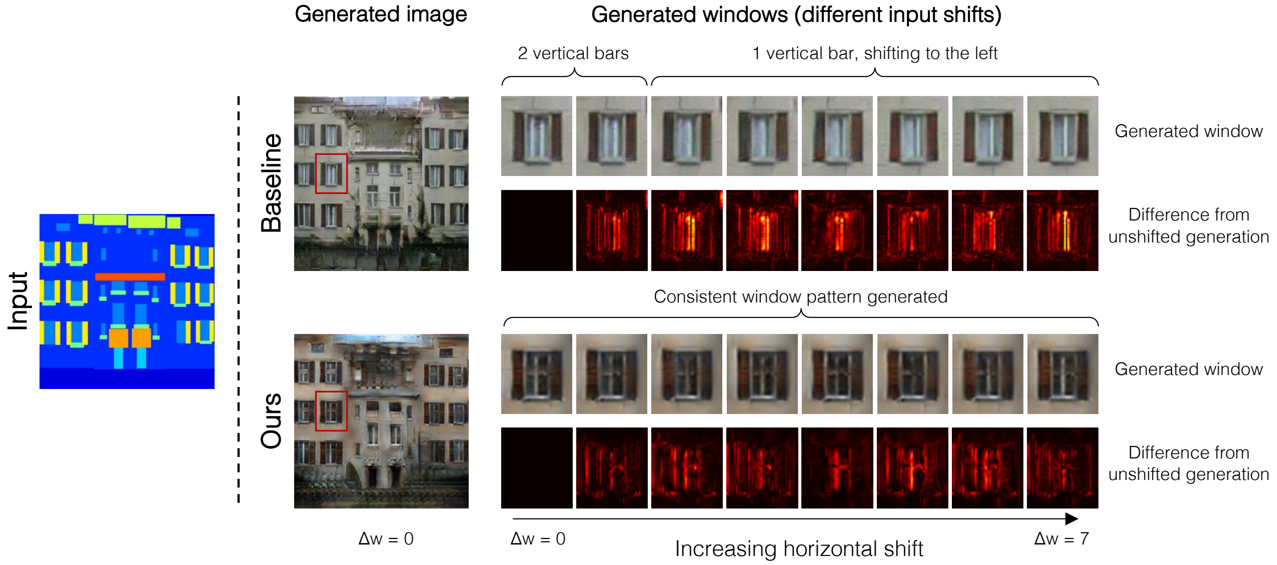

Generated image
Generated windows (different input shifts)
2 vertical bars
1 vertical bar, shifting to the left
Input
Baseline
Generated window
Difference from unshifted generation
Consistent window pattern generated
Ours
Generated window
Difference from unshifted generation
Δw = 0
Δw = 0
Increasing horizontal shift
Δw = 7

Figure 7. **Selected example of generation instability.** The left two images are generated facades from label maps. For the baseline method (top), input shifts cause different window patterns to emerge, due to naive downsampling and upsampling. Our method (bottom) stabilizes the output, generating the same window pattern, regardless the input shift.

	Delta-1	**Rect-2**	**Tri-3**	**Bin-4**	**Bin-5**
Stability [dB]	29.0	30.1	30.8	31.2	34.4
TV Norm ×100	7.48	7.07	6.25	5.84	6.28

Table 4. **Generation stability** PSNR (higher is better) between generated facades, given two horizontally shifted inputs. More aggressive filtering in the down and upsampling layers leads to a more shift-equivariant generator. **Total variation (TV) of generated images** (closer to ground truth images 7.80 is better). Increased filtering decreases the frequency content of generated images.

the input X shifts horizontally by Δw, the vertical bars on the generated window also shift. The generations start with two bars, to a single bar, and eventually oscillates back to two bars. A shift-equivariant network would provide the same resulting facade, no matter the shift.

Applying anti-aliasing filtering We augment the strided-convolution downsampling by blurring. The U-Net also uses upsampling layers, without any smoothing. Similar to the subsampling case, this leads to aliasing, in the form of grid artifacts (Odena et al., 2016). We mirror the downsampling by applying the same filter after upsampling. Note that applying the *Rect-2* and *Tri-3* filters while upsampling correspond to “nearest” and “bilinear” upsampling, respectively. By using the *Tri-3* filter, the same window pattern is generated, regardless of input shift, as seen in Fig. 7 (bot).

We measure similarity using peak signal-to-noise ratio between generated facades with shifted and non-shifted inputs: $\mathbb{E}_{X,\Delta w}\text{PSNR}(\text{Shift}_{0,\Delta w}(F(X)), F(\text{Shift}_{0,\Delta w}(X))))$. In Tab. 4, we show that the smoother the filter, the more shift-equivariant the output.

A concern with adding low-pass filtering is the loss of ability to generate high-frequency content, which is critical for generating high-quality imagery. Quantitatively, in Tab. 4, we compute the total variation (TV) norm of the generated images. Qualitatively, we observe that generation quality typically holds with the *Tri-3* filter and subsequently degrades. In the supplemental material, we show examples of applying increasingly aggressive filters. We observe a boost in shift-equivariance while maintaining generation quality, and then a tradeoff between the two factors.

These experiments demonstrate that the technique can make a drastically different architecture (U-Net) for a different task (generating pixels) more shift-equivariant.

5. Conclusions and Discussion

Shift-equivariance is lost in modern deep networks, as commonly used pooling layers ignore Nyquist sampling and alias. We integrate low-pass filtering to anti-alias, a common signal processing technique. Our method is a simple architectural modification and compatible with commonly-used pooling layers. We achieve higher consistency in both image classification and conditional image generation, across architectures and downsampling techniques.

In addition to more consistent classifications, in ImageNet classification, we observe a modest boost in accuracy across architectures. Future directions include exploring the potential benefit to downstream applications, such as nearest-neighbor retrieval, improving temporal consistency in video models, robustness to adversarial examples, and high-level vision tasks such as detection. Adding the inductive bias of shift-invariance serves as “built-in” shift-based data augmentation. This is potentially applicable to online learning scenarios, where the data distribution is changing.

ACKNOWLEDGMENTS

I am especially grateful to Eli Shechtman for helpful discussion and guidance. I thank Michaël Gharbi, Andrew Owens, and anonymous reviewers for providing beneficial feedback on earlier drafts. I thank labmates and mentors, past and present – Sylvain Paris, Oliver Wang, Alexei A. Efros, Angjoo Kanazawa, Taesung Park, Phillip Isola – for their helpful comments and encouragement.

References

Adelson, E. H., Anderson, C. H., Bergen, J. R., Burt, P. J., and Ogden, J. M. Pyramid methods in image processing. *RCA engineer*, 29(6):33–41, 1984.

Aubry, M. and Russell, B. C. Understanding deep features with computer-generated imagery. In *ICCV*, 2015.

Azulay, A. and Weiss, Y. Why do deep convolutional networks generalize so poorly to small image transformations? In *arXiv*, 2018.

Bietti, A. and Mairal, J. Invariance and stability of deep convolutional representations. In *NIPS*, 2017.

Bruna, J. and Mallat, S. Invariant scattering convolution networks. *TPAMI*, 2013.

Burt, P. J. and Adelson, E. H. The laplacian pyramid as a compact image code. In *Readings in Computer Vision*, pp. 671–679. Elsevier, 1987.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2018.

Cohen, T. and Welling, M. Group equivariant convolutional networks. In *ICML*, 2016.

Dosovitskiy, A. and Brox, T. Generating images with perceptual similarity metrics based on deep networks. In *NIPS*, 2016a.

Dosovitskiy, A. and Brox, T. Inverting visual representations with convolutional networks. In *CVPR*, 2016b.

Engstrom, L., Tsipras, D., Schmidt, L., and Madry, A. A rotation and a translation suffice: Fooling cnns with simple transformations. In *ICML*, 2019.

Esteves, C., Allen-Blanchette, C., Zhou, X., and Daniilidis, K. Polar transformer networks. In *ICLR*, 2018.

Fawzi, A. and Frossard, P. Manitest: Are classifiers really invariant? In *BMVC*, 2015.

Fowler, J. E. The redundant discrete wavelet transform and additive noise. *IEEE Signal Processing Letters*, 12(9): 629–632, 2005.

Fukushima, K. and Miyake, S. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pp. 267–285. Springer, 1982.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

Gonzalez, R. C. and Woods, R. E. *Digital Image Processing*. Pearson, 2nd edition, 1992.

Goodfellow, I., Lee, H., Le, Q. V., Saxe, A., and Ng, A. Y. Measuring invariances in deep networks. In *NIPS*, 2009.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *NIPS*, 2014a.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *ICLR*, 2014b.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Hénaff, O. J. and Simoncelli, E. P. Geodesics of learned representations. In *ICLR*, 2016.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *CVPR*, 2017.

Hubel, D. H. and Wiesel, T. N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154, 1962.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

Kanazawa, A., Sharma, A., and Jacobs, D. Locally scale-invariant convolutional neural networks. In *NIPS Workshop*, 2014.

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. Skip-thought vectors. In *NIPS*, 2015.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., and Jackel, L. D. Handwritten digit recognition with a back-propagation network. In *NIPS*, 1990.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lee, C.-Y., Gallagher, P. W., and Tu, Z. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *AISTATS*, 2016.

Lenc, K. and Vedaldi, A. Understanding image representations by measuring their equivariance and equivalence. In *CVPR*, 2015.

Leung, T. and Malik, J. Representing and recognizing the visual appearance of materials using three-dimensional textons. *IJCV*, 2001.

Lowe, D. G. Object recognition from local scale-invariant features. In *ICCV*, 1999.

Mahendran, A. and Vedaldi, A. Understanding deep image representations by inverting them. In *CVPR*, 2015.

Mairal, J., Koniusz, P., Harchaoui, Z., and Schmid, C. Convolutional kernel networks. In *NIPS*, 2014.

Mordvintsev, A., Olah, C., and Tyka, M. Deepdream-a code example for visualizing neural networks. *Google Research*, 2:5, 2015.

Nguyen, A., Clune, J., Bengio, Y., Dosovitskiy, A., and Yosinski, J. Plug & play generative networks: Conditional iterative generation of images in latent space. In *CVPR*, 2017.

Nyquist, H. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, pp. 617–644, 1928.

Odena, A., Dumoulin, V., and Olah, C. Deconvolution and checkerboard artifacts. *Distill*, 2016. doi: 10.23915/distill.00003. URL http://distill.pub/2016/deconv-checkerboard.

Oppenheim, A. V., Schafer, R. W., and Buck, J. R. *Discrete-Time Signal Processing*. Pearson, 2nd edition, 1999.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

Ruderman, A., Rabinowitz, N. C., Morcos, A. S., and Zoran, D. Pooling is neither necessary nor sufficient for appropriate deformation stability in cnns. In *arXiv*, 2018.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

Scherer, D., Muller, A., and Behnke, S. Evaluation of pooling operations in convolutional architectures for object recognition. In *ICANN*. 2010.

Sifre, L. and Mallat, S. Rotation, scaling and deformation invariant scattering for texture discrimination. In *CVPR*, 2013.

Simoncelli, E. P., Freeman, W. T., Adelson, E. H., and Heeger, D. J. Shiftable multiscale transforms. *IEEE transactions on Information Theory*, 38(2):587–607, 1992.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Su, J., Vargas, D. V., and Sakurai, K. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 2019.

Tyleček, R. and Šára, R. Spatial pattern templates for recognition of objects with regular structure. In *German Conference on Pattern Recognition*, pp. 364–374. Springer, 2013.

Vedaldi, A. and Fulkerson, B. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/, 2008.

Worrall, D. E., Garbin, S. J., Turmukhambetov, D., and Brostow, G. J. Harmonic networks: Deep translation and rotation equivariance. In *CVPR*, 2017.

Xiao, C., Zhu, J.-Y., Li, B., He, W., Liu, M., and Song, D. Spatially transformed adversarial examples. *ICLR*, 2018.

Yu, F. and Koltun, V. Multi-scale context aggregation by dilated convolutions. *ICLR*, 2016.

Yu, F., Koltun, V., and Funkhouser, T. Dilated residual networks. In *CVPR*, 2017.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. Object detectors emerge in deep scene cnns. In *ICLR*, 2015.