MAKING CONVOLUTIONAL NETWORKS
SHIFT-INVARIANT AGAIN

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern convolutional networks are not shift-invariant, despite their convolutional nature: small shifts in the input can cause drastic changes in the internal feature maps and output. In this paper, we isolate the cause – the downsampling operation in convolutional and pooling layers – and apply the appropriate signal processing fix – low-pass filtering before downsampling. This simple architectural modification boosts the shift-equivariance of the internal representations and consequently, shift-invariance of the output. Importantly, this is achieved while maintaining downstream classification performance. In addition, incorporating the inductive bias of shift-invariance largely removes the need for shift-based data augmentation. Lastly, we observe that the modification induces spatially-smoother learned convolutional kernels. Our results suggest that this classical signal processing technique has a place in modern deep networks.

1 INTRODUCTION

Deep convolutional neural networks (CNNs) are designed to perform high-level tasks and be robust to low-level nuisance factors. For example, small shifts in the input should simply shift the internal feature maps (shift-equivariance), and leave the output relatively unaffected (shift-invariance). This property has been explicitly engineered through convolutional and pooling layers, where the same function is applied on a local region across the image in a sliding window fashion. However, recent work (Engstrom et al., 2017; Azulay & Weiss, 2018) has found that small shifts can drastically change the output of a classification network. Why is this the case?

Shift-invariance is lost when spatial resolution is lost, for example, from pooling layers. Our insight is that conventional strided-pooling, as shown in Fig. 1 (top), is inherently composed of two operations: (1) evaluating the pooling operator densely (without striding), and (2) downsampling. Naive downsampling loses shift-equivariance, as high-frequency components of the signal alias into low-frequencies. This phenomenon is commonly illustrated in movies, where wheels appear to spin backwards, due to the frame rate not meeting the Nyquist sampling criterion (known as the Stroboscopic effect). Separating out these operations is important, as it allows us to keep the advantages of the pooling operation, which preserves shift-equivariance, while applying the appropriate fix to the downsampling operation.

We propose to add the signal processing tool of low-pass filtering before downsampling, as shown in Fig. 1 (bottom). By low-pass filtering, the high-frequency components of the signal are reduced, reducing aliasing and better preserving shift-equivariance. This ultimately cascades into better shift-invariance in the output. We show example classification instabilities in Fig. 2.

A potential concern is that over-aggressive low-pass filtering can result in heavy loss of information. However, we find that with a reasonable selection of low-pass filter weights, we can maintain classification performance while increasing shift-invariance. Furthermore, we show that without shift-based data augmentation, incorporating this inductive bias actually improves performance.

We find that the learned filters also naturally become smoother after adding the blurring layer. These results indicate that incorporating this small modification not only induces shift-invariance, but causes the network to learn a smoother feature extractor.

In summary, our contributions are as follows:
Figure 1: (Top) Pooling does not preserve shift-equivariance. It is functionally equivalent to densely evaluated pooling (which preserves shift-equivariance), followed by naive downsampling. The latter operation ignores the Nyquist sampling theorem and loses shift-equivariance. (Bottom) We propose to low-pass filter between the pooling and downsampling operations. This modification allows us to keep the original pooling operation of choice untouched, while applying antialiasing to the appropriate signal before the downsampling operation. This equivalent analysis and modification can be applied to any type of strided layer, such as convolution.

- We isolate the cause for loss of shift-invariance – downsampling. Separating the down-sampling and pooling operations enables us to keep the desired pooling function, while fixing the loss of shift-equivariance. We propose to low-pass filter before downsampling, a common signal processing technique.
- We validate on a classification task, and demonstrate increased shift-equivariance in the features and shift-invariance in the output.
- In addition, we observe large improvements in classification performance when training without shift-augmentation, indicating more efficient usage of data.

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, 2014), ResNet (He et al., 2016), and DenseNet (Huang et al., 2017). In biological systems, local connectivity was famously discovered observed in a cat’s visual system by Hubel & Wiesel (1962). Recent work has strived to build in additional types of invariances, such as rotation, reflection, and scaling (Sifre & Mallat, 2013; Bruna & Mallat, 2013; Esteves et al., 2017; Kanazawa et al., 2014; Worrall et al., 2017; Cohen & Welling, 2016). Our work focusses on the elusive goal of shift-invariance.

Though properties such as shift-equivariance have been engineered into networks, what factors and invariances does an emergent representation actually learn? Analysis of deep networks have included qualitative approaches, such as showing patches which activate hidden units (Girshick et al., 2014; Zhou et al., 2014), actively maximizing hidden units (Mordvintsev et al., 2015), and mapping features back into pixel space (Dosovitskiy & Brox, 2016; Mahendran & Vedaldi, 2015; Zeiler & Fergus, 2014; Nguyen et al., 2017; Henaff & Simoncelli, 2015). Our analysis is focussed on a specific, low-level property and is complementary to these qualitative 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 (Ruderman et al., 2018), Fawzi & Prossard (2015), and CG renderings with various shape, poses, and colors (Aubry & Russell, 2015). A related line of work is in adversarial examples, where directed perturbations in the input can result in large changes in the output. These perturbations can be directly on pixels (Goodfellow et al., 2014a,b), a single pixel (Su et al., 2017), small deformations (Xiao et al., 2018), or even affine transformations (Engstrom et al., 2017). We aim make the network robust to the simplest of these types of attacks and perturbations: shifts. Both Hénaff & Simoncelli (2015) and Azulay & Weiss (2018) identify that modern deep networks ignore the Nyquist sampling criterion when downsampling. In our work, we propose and empirically validate an easily adoptable fix which minimally perturbs the existing network architecture.

Classic hand-engineered computer vision and image processing representations, such as SIFT (Lowe, 1999), wavelets, and image pyramids (Burt & Adelson, 1987; Adelson et al., 1984) 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 with 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, 2010). Deep networks can also be evaluated densely, by removing striding and making appropriate changes to subsequent layers by using à trous dilated convolutions (Chen et al., 2014, 2018; Yu & Koltun, 2015). This comes at great computation and memory cost. Our work investigates achieving shift-equivariance with minimal additional computation.

3 METHODS

3.1 Preliminaries

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

Shift-equivariance and shift-invariance A representation $\tilde{\mathcal{F}}$ is shift-equivariant if shifting the input produces a shifted feature map, meaning that shifting and feature extraction are commutable.

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

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

$$\tilde{\mathcal{F}}(X) = \tilde{\mathcal{F}}(\text{Shift}_{\Delta h, \Delta w}(X)) \quad \forall (\Delta h, \Delta w)$$
Figure 3: Example of sensitivity to shifts when downsampling. We show a simple example of max-pooling, to illustrate how downsampling affects shift-equivariance. (Top-Left) We show a toy signal in light gray. Max-pooling this signal with (kernel 2, stride 2) is shown in blue. (Top-Right) Simply shifting the input signal provides a completely different max-pooled result, shown in red. (Bot-Left) Max-pooling densely (kernel 2, stride 1) is shown in thick black. The blue and red points are inherently sampled from this intermediate signal. The red and blue are far from each other, and shift-equivariance is lost. (Bot-Right) In the proposed method, we sample from the low-passed version of the signal (low-pass kernel of [0.25, 0.5, 0.25]), shown in green and magenta. Shift-equivariance is better preserved (though not perfectly).

For modern classifiers, layer $l = 0$ is the raw pixels, and final layer $L$ is a probability distribution over $D$ classes, $\mathcal{F}_L \in \Delta^{1 \times 1 \times D}$. The network typically reduces spatial resolution throughout the network, until all spatial resolution is lost and features are of shape $\mathbb{R}^{1 \times 1 \times C}$. A common technique, such as used in (He et al., 2016; Lin et al., 2013), is to average across the entire feature map spatially, and use fully-connected layers in all subsequent layers, which can be expressed as $1 \times 1$ convolutions (Long et al., 2015). In such a setting, as proven by Azulay & Weiss (2018), shift-invariance on the output will necessarily emerge from shift-equivariance in the convolutional features.

**Modulo-$N$ shift-equivariance/invariance** In some cases, the definitions in Equations 1, 2 may hold only when shifts $(\Delta h, \Delta w)$ are integer multiples of $N$. We refer to these scenarios as modulo-$N$ shift-equivariance or invariance. For example, modulo-2 shift-invariance means that even-pixel shifts of the input result in an identical representation, but odd-pixel shifts may not.

### 3.2 Conventional Pooling vs Proposed Pool-Blur-Downsample

**Conventional strided pooling breaks shift-equivariance** In Fig. 3, we show an example 1-D signal $[0, 0, 1, 1, 0, 0, 1, 1]$. Max-pooling (with kernel size 2, stride 2) will result in $[0, 1, 0, 1]$. Simply shifting the input by one index results a dramatically different answer, $[1, 1, 1, 1]$. Shift-equivariance is lost. As seen in the middle-left, both of these results are inherently downsampling from an intermediate signal – a dense max-pooling (kernel size 2, stride 1) of the original signal. We can write a max-pooling layer as a composition of two functions, max-pooling densely evaluated, followed by naive downsampling: MaxPool$_{k,s}(X) = \text{Downsample}_s(\text{MaxPool}_{k,1}(X))$.

In the illustrative example, there is significant energy in the high-frequency in this intermediate signal. Max-pooling preserves shift-equivariance (when evaluated densely), but naive downsampling does not.

**Blurring before downsampling better preserves shift-equivariance** We propose to low-pass filter the intermediate signal before downsampling, as shown in Fig. 3 (mid-right). We define our MaxPoolBlurDownsample operator below.

$$\text{MaxPoolBlurDownsample}_{k,s}(X) = \text{Downsample}_s(\text{Blur}_{k,s,\sigma}(\text{MaxPool}_{k,1}(X)))$$  \hspace{1cm} (3)

Sampling from the low-pass filtered signal gives $[.5, .1, .5, 1]$ and $[.75, .75, .75, .75]$ (Fig. 3 mid-right). These are closer to each other and better representations of the intermediate signal.

The method allows for a choice of blur kernel. In image processing, small kernels are often used across applications such as edge detection (Canny 1986) and image pyramids (Adelson et al. 1984).
Figure 4: Shift equivariance throughout the network. We show how close each layer is to satisfying shift-equivariance condition by commuting the shift and feature extraction operations, and computing their distance in feature space across the test set. Each point in each heatmap is a value of $(\Delta h, \Delta w)$. Resolution of the layer written in [brackets]. The last three layers have no spatial information, and shift-equivariance is equivalent to shift-invariance. Layers pix-pool1(dense) have perfect equivariance (distance 0 at all shifts, shown by blue). Red is half mean distance between two random different images. (a) On the baseline VGG13 net, shift-equivariance is reduced each time downsampling takes place. Modulo-N shift-equivariance holds with N doubling with each downsampling. (b) With our proposed change, shift-equivariance is better maintained throughout the network, and the resulting classifications (softmax) layer is more shift-invariant.

We try a number of kernels, ranging from size $2 \times 2$ to $7 \times 7$. As the blur kernels are separable, it can be implemented as a series of two convolutions (vertical blur followed by horizontal), and added computation scales linearly with $k_{blur}$, rather than quadratically.

4 Experiments

4.1 Experimental Setup

Data, architecture, training schedule We test on CIFAR10 classification (Krizhevsky & Hinton, 2009), which consists of 50k training and 10k testing images at resolution $32 \times 32$. We use the VGG13 architecture (Simonyan & Zisserman, 2014). Each block consists of 2 Conv-BatchNorm-ReLU chunks, followed by a MaxPool. Each block doubles the feature channels and reduces spatial resolution by a factor of 2, until all spatial resolution is lost. A final softmax layer predicts a probability vector. We use stochastic gradient descent (SGD) with momentum 0.9 and batch size 128. We train for 100 epochs at the initial learning rate 0.1 and 50 additional epochs at 0.01 and 0.001. We use the PyTorch framework (Paszke et al., 2017) and will make code available.

Low-pass filter kernels We try a number of standard low-pass filters, shown in Table 1, ranging from size $2 \times 2$ to $7 \times 7$. All filters allow the DC signal pass and suppress (or completely kill) the highest frequency. Variations in filters correspond to tradeoffs between location of the cutoff frequency, slope of the cutoff, and variation of lobes in the passband and stopband. These properties are well-studied in the context of finite impulse response (FIR) filter design. However, it is unclear which types of filters are best suited for deep networks, so we empirically investigate their effects.

Circular convolution and shifting Edge artifacts are an important consideration. When an image is shifted, information is necessarily lost on one side, and has to be filled in on the other. In all our experiments, we use circular shifting and convolution. When the convolutional kernel hits the edge, it wraps to the other side. When shifting, pixels are “rolled” off the edge to the other side.

$$[\text{Shift}_{\Delta h, \Delta w}(X)]_{h,w,c} = X_{(h-\Delta h)\%H,(w-\Delta w)\%W,c}, \text{ where } \% \text{ is the modulus function} \quad (4)$$
We use the following metrics to characterize shift-equivariance and invariance.

1. Feature distance (lower is better) We test how close shift-equivariance and invariance are to being fulfilled by computing $d(\Delta h, \Delta w, \bar{F}(X)), \bar{F}(\Delta h, \Delta w, X))$ and $d(\bar{F}(X), \bar{F}(\Delta h, \Delta w, X))$ (left & right-hand sides of Eq. 1, 2), respectively. We use cosine distance, which is commonly used for deep features (Kiros et al., 2015; Zhang et al., 2018).
2. Classification consistency (higher is better) The metric above can be used to measure shift-invariance of the output classification. Perhaps of greater interest is the actual decision the classifier makes. We can measure its consistency by checking how often the network outputs the same classification, given the same image with two different shifts: \[ E[X, h, w_1, w_2, \Delta h, \Delta w] I \{ \arg \max P(\text{Shift}_{h, w_1}(X)) = \arg \max P(\text{Shift}_{h, w_2}(X)) \}. \]

3. Classification variation (lower is better) Similar to Azulay & Weiss (2018), we trace the variation in probability of correct classification, given different shifts. We can capture the variation across all possible shifts: \[ \sqrt{\text{Var}_{h, w}(\{ P_{\text{correct class}}(\text{Shift}_{h, w}(X)) \})}. \]

Table 1 shows results across a number of different low-pass filters, training with and without data augmentation. We dissect the results below.

How shift-equivariant are deep features? In Fig. 4 (top), we compute distance from shift-equivariance, as a function of all possible shift-offsets \((\Delta h, \Delta w)\) and layers. MaxPool layers are broken into two components – before and after downsampling. Pixels are trivially shift-equivariant, as are all layers before the first downsampling. Once downsampling occurs in pool1(ds), shift-equivariance is lost. However, modulo-N shift-equivariance still holds, and each subsequent downsampling doubles the factor.

Additionally, we observe that before the downsampling operation, the pooling layer first increases shift-equivariance (e.g., conv3_2 to pool13(dense)). This is consistent with the long-held intuition that pooling build invariances inside the network (LeCun et al., 1990) and isolates the downsampling operation as the culprit behind loss of shift-equivariance.

Does blurring before downsampling achieve better shift-equivariance? In Fig. 4 (bottom), we add a blurring filter to the MaxPool layers, as proposed in Section 3 and again plot shift-equivariance maps for each layer. 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 final softmax layer.

Some selected examples are in Fig. 2. Our method stabilizes the classifications. In Fig. 6 we show the distribution of classification variations, before and after adding in the low-pass filter. Even a small \(2 \times 2\) filter, immediately variation. As the filter size is increased, the output classification variation decreases. This has a larger effect when training without data augmentation, but is still observable when training with data augmentation.

Does shift-invariance degrade performance? Our method produces more shift-equivariant feature maps and consequently, more shift-invariant outputs. However, does this come at a cost?

We study the output classification consistency versus classification accuracy. In Fig. 5 (left), we show results, trained without shift-based data augmentation. Training with the baseline MaxPooling gives accuracy 91.6% and consistency 88.1%. Our proposed change – with a \(5 \times 5\) triangle filter
improves accuracy to 93.3% and consistency to 98.2%. This indicates that low-pass filtering does not destroy the signal, or make learning harder. On the contrary, preserving shift-equivariance serves as "built-in" augmentation, indicating more efficient data usage.

In principle, networks can learn to be shift-invariant from data. Does adding shift-based data augmentation remove the benefit from method? Shift-based data augmentation with the baseline network results in consistency of 96.6%, lower than our method trained without data augmentation. In addition, as seen in Fig. 5 (right), applying out method with data augmentation provides an immediate jump in classification consistency, while maintaining accuracy. From there, a clear tradeoff appears — higher amounts of shift-invariance can be achieved at the cost of decreased accuracy. For example, very large rectangular filters over-aggressively smooth the signal. Downstream applications may favor one factor over another, and the choice of filter allows one to explore this space.

Fig. 5 investigates the distribution of classification variations. Training with data augmentation with the baseline network reduces variation (black distributions on left and right plots). Adding our method reduces variation in both scenarios. More aggressive filtering further decreases variation.

How do the learned convolutional filters change with the proposed modification? We measure spatial smoothness using the normalized Total Variation (TV) metric proposed in [23]. Our proposed change smooths the internal feature maps for purposes of downsampling. As shown in Fig. 6, this induces smoother learned filters throughout the network. Adding in more aggressive blur kernels further decreases the TV (increasing smoothness). This indicates that our modification actually induces a smoother feature extractor overall.

How does the proposed method affect timing? In Tab. 2, we show the added time each element of the proposed method takes: evaluating the MaxPool layer at stride 1 instead of stride 2, and running a blurring filter. Since the blurring filters are separable, time increases linearly with filter size. The largest filter adds 12.3% per forward pass. This is significantly cheaper than evaluating multiple forward passes in an ensembling approach. These timings are on our VGG13 network setup. With deeper networks, the relative added computation decreases.

5 Conclusions and Discussion

In summary, we show that shift-invariance is lost through a deep network as downsampling in pooling layers do not meet the Nyquist criteria. We propose a simple architectural modification, following signal processing principles, to improve shift-equivariance. This change allows the network architecture designer to keep their pooling layer of choice untouched.

We achieve higher consistency while maintaining classification performance. In addition, we show large improvements in both performance and consistency when training without data augmentation. This is potentially applicable to online learning scenarios, where the data distribution is changing. 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. Another possible future direction is learning the downsampling kernels. Overall, our experiments indicate that this classical signal processing technique has a place in modern deep networks.
REFERENCES


