Learned Deformation Stability in Convolutional Neural Networks

Avraham Ruderman [1] **Neil Rabinowitz** [1] **Ari S. Morcos** [1] **Daniel Zoran** [1]

Abstract

Conventional wisdom holds that interleaved pooling layers in convolutional neural networks lead to stability to small translations and deformations. In this work, we investigate this claim empirically. We find that while pooling confers stability to deformation at initialization, the deformation stability at each layer changes significantly over the course of training and even decreases in some layers, suggesting that deformation stability is not unilaterally helpful. Surprisingly, after training, the pattern of deformation stability across layers is largely independent of whether or not pooling was present. We then show that a significant factor in determining deformation stability is filter smoothness. Moreover, filter smoothness and deformation stability are not simply a consequence of the distribution of input images, but depend crucially on the joint distribution of images and labels. This work demonstrates a way in which biases such as deformation stability can in fact be learned and provides an example of understanding how a simple property of learned network weights contributes to the overall network computation.

1. Introduction

In recent years, Convolutional Neural Networks (CNNs) have proven extremely successful at object recognition in computer vision (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; He et al., 2016; Russakovsky et al., 2015). However, understanding why these models are so successful remains unclear.

Until recently, one common explanation for the success of CNNs was that interleaved pooling layers make these models insensitive to small translations and deformations. Much of the variability in the visual domain comes from slight changes in view, object position, rotation, size and non-rigid deformations of, for example, organic objects. It would therefore seem plausible that representations which are less sensitive to such transformations would be useful. Further, has long been hypothesized that building this bias into the model itself with interleaved pooling layers would be of benefit (LeCun et al., 1990; Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; LeCun et al., 2015; Giusti et al., 2013). However this hypothesized explanation has not been thoroughly tested.

Figure 1. **Examples of deformed ImageNet images.** left: original images, right: deformed images. While the images have changed significantly, for example under the L^2 metric, they would likely be given the same label by a human.

Indeed, recent evidence suggests that the inductive bias provided by interleaved pooling layers is not actually necessary for good performance as recent successful architectures have dropped the interleaved pooling layers and yet have still achieved strong performance (Springenberg et al., 2014; He et al., 2016). This raises the following questions which we address in this paper:

1. Does pooling have an effect on the learned deformation stability?
2. Is deformation stability achieved in the absence of pooling?
3. If so, how can deformation stability be achieved in the absence of pooling?

[1]DeepMind, London, UK. Correspondence to: Avraham Ruderman <aruderman@google.com>.

This traditional line of reasoning regarding pooling assumes that invariance to nuisance variables is helpful in general. Here, we provide an intuition as to why this might be the case and further define the particular class of nuisance deformations which we will focus on in this work.

Invariance Invariance to transformations of an input X that do not affect the output Y can help in achieving good generalization. In particular, if

$$P(Y|\tau(X)) = P(Y|X) \ \ \forall \tau \in T$$

and if a model f is invariant to some deformation function τ,

$$f(\tau(X)) = f(X) \ \ \forall \tau \in T$$

then any good prediction on an example x will automatically lead to a good predictions on all examples $\{\tau(x) : \tau \in T\}$ possibly leading to a large reduction in the number of examples required for learning.

Deformation invariance In this work, we focus on invariance, or more precisely stability, to smooth deformations of image coordinates. We consider transformations of image coordinates $\tau : \mathbb{R}^2 \to \mathbb{R}^2$ such that $\|\nabla\tau\|_\infty < C$ for some C, similar to Mallat (2012). For examples of the effects of such transformations, see Figure 1.

Transformations such as global shear, rotation and scaling are also deformations but form only a small subset of all possible deformations. For instance, many simultaneous small rotations and translations of different object parts in different directions in a single image can be well approximated by a deformation, but not by a single rotation, translation or shear.

It seems plausible that invariance to such deformations would be useful since much of the variability in the visual domain comes from transformations which are well approximated by continuous deformations of the image coordinates. In addition, some empirical evidence exists that supports the claim that such invariances help with generalization. For example, Wong et al. (2016) showed that augmenting a small subset of the MNIST training set with deformed versions of the images greatly improves test performance.

In this work, we will discuss stability rather than invariance of representations. While invariance suggests that a representation does not change *at all* as the consequence of a change to the input, stability merely suggests that a representation does not change *much* as the consequence of a change to the input.

1.1. History of deformation invariance in machine learning

Deformation invariance has historically been built into machine learning models for computer vision (Lowe, 1999; Felzenszwalb et al., 2008). As we show in this work however, building in this bias into convolutional networks is not actually necessary as deformation stability can alternatively be learned and implemented through smooth filters.

Invariances in non-neural models In non-neural computer vision models, invariance to deformation has long been used. For example, SIFT features are local features descriptors constructed such that they are invariant to translation, scaling and rotation (Lowe, 1999). In addition, by using blurring, SIFT features become somewhat robust to deformations. Another example is deformable parts models which contain a single stage spring-like model of connections between pairs of object parts giving robustness to translation at a particular scale (Felzenszwalb et al., 2008).

Deformation invariance and pooling Important early work in neuroscience found that in the visual cortex of cats, there exist special complex-cells which are somewhat insensitive to the precise location of edges (Hubel & Wiesel, 1968). These findings inspired work on the neocongnitron which included components capturing a similar invariance which were inspired by complex cells (Fukushima & Miyake, 1982). In turn, the use of complex cells in the neocognitron inspired the use of pooling layers in CNNs (LeCun et al., 1990).

In early papers proposing CNNs, pooling is motivated as conferring invariance to translations and deformations:

> Each feature extraction in our network is followed by an additional layer which performs a local averaging and a sub-sampling, reducing the resolution of the feature map. This layer introduces a certain level of invariance to distortions and translations. (LeCun et al., 1990)

In fact, until recently, pooling was still seen as an essential ingredient in CNNs, allowing for invariance to small shifts and distortions (Simonyan & Zisserman, 2014; He et al., 2016; Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; LeCun et al., 2015; Giusti et al., 2013).

1.2. Previous analyses of invariances in CNNs

While it has long been reasoned informally that CNNs with pooling are invariant to deformation, empirical evidence for this has been lacking. More recently however, significant work has been done to investigate the stabilities of CNNs both theoretically and empirically.

Theoretical analyses A significant body of theoretical work shows formally that scattering networks, which share some architectural components with CNNs, are stable to deformations (Mallat, 2012; Sifre & Mallat, 2013; Bruna & Mallat, 2013; Mallat, 2016). However this work does not apply

to widely used CNN architectures for two reasons. First, the architectures differ significantly, including differences in connectivity, pooling, and non-linearities. Second, and perhaps more importantly, this line of work assumes that the filters are fixed wavelets that do not change during training.

More recent theoretical work applies to architectures more similar to CNNs used in practice (Bietti & Mairal, 2017). However, this work requires that the network has interleaved pooling layers between the convolutional layers. Therefore, these results cannot not explain the success of more recent architectures that do not have interleaved pooling layers.

Empirical investigations Previous empirical investigations of invariances in CNNs have focused on a more limited set of invariances such as global affine transformations (Lenc & Vedaldi, 2015). More recent work has looked at robustness to adversarial geometric transformations. In particular, these studies looked at worst-case sensitivity of the output to such transformations and found that indeed CNNs are often quite sensitive to geometric transformation and that augmenting a the training sets with such transformations can lead to increased robustness (Fawzi & Frossard, 2015; Kanbak et al., 2017). However this line of work does not address how deformation invariance is achieved and how it changes over the course of training. In addition, these investigations have considered only a limited class of deformations, whereas we consider a much larger class of transformations.

1.3. Our contributions

The main contributions of this work are

- **Learned stability:** We show that networks without pooling are sensitive to deformation at initialization but over the course of training learn representations that are stable to deformation. We also show that the pattern of deformation stability changes significantly over the course of training even in networks with pooling. Moreover, deformation stability sometimes decreases over the course of training, suggesting that this stability is not unilaterally helpful (Subsection 3.2).
- **Convergent stability:** We show that the pattern of deformation stability across layers for trained networks with and without pooling, converges to a similar structure (Subsection 3.3).
- **Implementation of stability:** We show that networks both with and without pooling implement and modulate deformation stability through the smoothness of filters (Section 4).

Additionally, from the perspective of understanding how the weights and layers of a neural network give rise to overall network behavior, this work provides a potentially important example of how it is possible to understand simple properties of the weights in each layer and how these properties contribute to the overall computation carried out by the network.

From the perspective of designing neural network models, this work provides insight into what was long considered an important inductive bias that has the guided design of neural networks for over 20 years. It has long been assumed that pooling was important in achieving deformation invariance and that pooling was a major contributor to the success of CNNs. This work demonstrates that our intuitions as to why neural networks work can often be inaccurate, no matter how reasonable they may seem, and can be sharpened through empirical and theoretical validation.

2. Methods

2.1. Deformation sensitivity

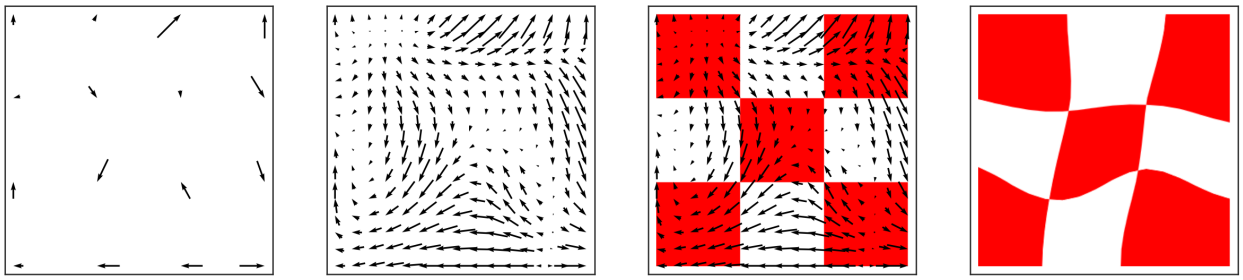
Throughout this work we will measure deformation stability by: i) sampling random deformations as described below; ii) applying these deformations to input images; iii) measuring the effect of this image deformation on the representations throughout the various layers of the CNN.

Generating deformed images To generate deformed images for ImageNet we use a grid of 5x5 evenly spaced control points on the image and then choose a destination for each control point uniformly at random with a maximum distance of 10 in each direction. For CIFAR10 images, we used 3x3 evenly spaced control points and a maximum distance of 2 pixels in each direction. The resulting deformation map was then smoothed using thin plate spline interpolation and finally the deformation was applied with bilinear interpolation (Duchon, 1977). The entire process for generating deformed images is illustrated in Figure 2. Figure 1 shows some examples of deformed images using this method.

Measuring sensitivity to deformation For a representation r mapping from an input image (e.g., 224x224x3) to some layer of a CNN (e.g., a tensor of size 7x7x512), we measure sensitivity of the representation r to a deformation τ using the quantity

$$\frac{\mathrm{dcos}(r(x), r(\tau(x)))}{\mathrm{median}(\mathrm{dcos}(r(x), r(y)))}$$

where dcos is the cosine distance. That is, we normalize distances by the median distance in representation space between randomly selected images from the original dataset. We will refer to this quantity as the *Normalized Cosine Distance* and will use this repeatedly in graphs in this work. Note that while we use cosine distance here, we found that using the Euclidean distance instead gave qualitatively similar results. For our results, we average this quantity over 128 images and compute the median using all pairwise



(a) Coarse displacement map (b) Smooth displacement map (c) Overlayed on image (d) Final result

Figure 2. **Generating deformed images**: To randomly deform an image we: (a) Start with a fixed evenly spaced grid of control points (here 4x4 control points) and then choose a random source for each control point within a neighborhood of the point; (b) we then smooth the resulting vector field using thin plate interpolation; (c) vector field overlayed on original image: the value in the final result at the tip of an arrow is computed using bilinear interpolation of values in a neighbourhood around the tail of the arrow in the original image; (d) the final result.

distances between the representations of the 128 images.

Our work is of course limited to the distribution of deformations we have chosen. Indeed in future work we would like to explore sensitivity to deformation where the deformation τ is found via optimization rather than through sampling as in Fawzi & Frossard (2015) and Kanbak et al. (2017).

2.2. Model architectures and training

All networks we trained for our experiments are based on a modified version of the VGG network (Simonyan & Zisserman, 2014). The networks consist of multiple blocks as follows:

- **Conv block:** A block consists of multiple layers of convolutional filters followed by batch-norm and then a ReLU non-linearity, we will denote the structure of a block by the number of filters in each conv layer and the number of layers, for example, 2x64 will mean a block with 2 layers with 64 filters in the convolutional layers. All filters have a spatial dimension of 3x3.

- **Downsampling:** Each block is followed by a downsampling layer where the spatial resolution is decreased by a factor of 2 in both height and width dimensions.

- **Global average pooling:** we replace the fully connected layers of VGG with global average pooling and a single linear layer as is now commonly done (He et al. (2016) following Springenberg et al. (2014)).

For the ImageNet experiments, we used networks with block structure 2x64, 2x128, 3x256, 3x512, 3x512. For the CIFAR10 experiments, we used networks with block structure 2x32, 2x64, 2x128, 2x256.

We compared networks with the following downsampling layers in our CIFAR10 experiments:

- **Subsample:** Keep top left corner of each 2x2 block.
- **Max-pool:** Standard max-pooling layer.
- **Average-pool:** Standard average-pooling layer.
- **Strided:** we replace the max pooling layer with a convolutional layer with kernels of size 2x2 and stride 2x2.
- **Strided-ReLU:** we replace the max pooling layer with a convolutional layer with kernels of size 2x2 and stride 2x2. The convolutional layer is followed by batch-norm and ReLU nonlinearity.

For our ImageNet experiments, we compared only Max-pool and Strided-ReLU due to computational considerations.

To rule out variability due to random factors in the experiment (initial random weights, order in which data is presented), we repeated all experiments 5 times for each setting. The error bands in the plots correspond to 2 standard deviations estimated across these 5 experiments.

3. Learned deformation stability is similar in networks with and without pooling

It is a commonly held belief that pooling leads to invariance to small translations and deformations. But to what extent

- is pooling *sufficient* for achieving the correct amount of stability to deformation?
- is pooling *necessary* for achieving stability?

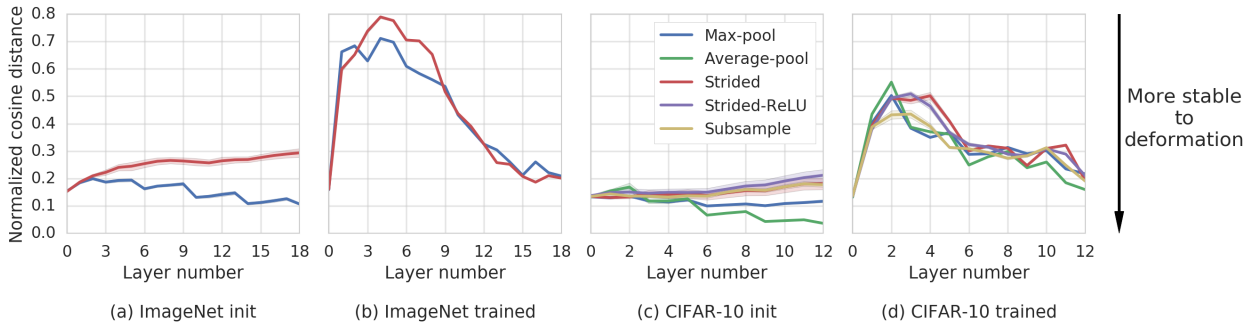


Figure 3. **Pooling confers stability to deformation at initialization but the stability changes significantly over the course of training and converges to a similar stability regardless of whether pooling is used.** (a) At initialization, networks with max-pooling are less sensitive to deformation. (b) After training, networks with and without max-pooling have very similar patterns of sensitivity to deformation throughout the layers. Similar patterns emerge for CIFAR10: (c) At initialization, pooling has significant impact on sensitivity to deformation but (d) after training, the choice of downsampling layers has little effect on deformation stability throughout the layers. Layer number 0 corresponds to the input image; The layers include the downsampling layers; The final layer corresponds to the final downsampling layer. For CIFAR10 we therefore have 1 input layer, 8 convolutional layers and 4 pooling layers for a total of 13 layers.

In this section we explore the above questions.

3.1. Pooling influences deformation stability

To test whether pooling on its own is sufficient for achieving any significant change in deformation stability, we measured the sensitivity to deformation of networks with pooling at initialization. As can be seen in Figure 3a, we find that indeed pooling leads to representations that are more stable to deformation at initialization than representations in networks without pooling. This result also provides us with a basic sanity check that our experimental setup is reasonable.

3.2. Pooling does not determine the final pattern of stability across layers

Next we ask to what extent does pooling confer the final pattern of deformation stability? Also, if pooling leads to a suboptimal pattern of deformation stability for the task, then to what extent can learning correct for this? To examine this, we measured the pattern of sensitivity to deformation of networks before and after training. Perhaps surprisingly, we found that the sensitivity to deformation of networks with pooling actually changes significantly over the course of training as can be seen in Figure 3b. Moreover while deformation sensitivity sometimes decreases with training, it often increases, especially in the early layers of the network. This result suggests that while deformation stability might be helpful in some cases, it is not always helpful.

3.3. The final pattern of deformation stability across layers is similar for networks with and without pooling

The above results demonstrate that pooling does not determine the final pattern of deformation stability across the layers of the network. This raises the question of whether pooling has any influence on the final deformation sensitivity at all. Recently, it has been observed that CNNs without pooling can achieve high accuracy on image classification tasks (Springenberg et al., 2014). If CNNs with and without pooling can perform equally well, do CNNs perform this task without any deformation stability at all? Or alternatively, can networks without pooling learn deformation stability? And if so, do networks without pooling learn similar patterns of deformation stability to networks with pooling? To test this, we measured the pattern of sensitivity to deformation of across layers for networks with and without pooling. As can be seen in Figure 3, sensitivity to deformation for networks with and without pooling trained on ImageNet are very similar throughout the layers, suggesting that pooling has almost no effect on the final deformation stability.

To rule out the dependence of this result on the particular dataset we trained on and the particular choice of downsampling layer, we repeated the experiments on the CIFAR10 dataset and with a variety of different downsampling layers. As can be seen in Figure 3, the results in this case are qualitatively similar: despite the choice of downsampling having a significant effect on sensitivity to deformation at initialization, by the conclusion of training, these differences have vanished, and all networks have converged to a similar pattern.

4. Filter smoothness contributes to deformation stability

In the previous section we demonstrated that pooling is not necessary for achieving deformation stability and that the final pattern of deformation stability after training is not significantly affected by the presence of pooling layers. This raises a critical question: if pooling is not the major determinant of deformation stability, then what is? In this section, we show that the smoothness of filters in the convolutional layers strongly impacts deformation stability.

Why might smooth filters lead to deformation stability? Informally, one could reason that a smooth filter can be decomposed into a less smooth filter of similar norm to the original filter followed by a smoothing operation similar to average pooling or smoothing with a Gaussian kernel (Bietti & Mairal, 2017). Therefore a smooth filter might function similarly to the combination of a pooling layer followed by a convolutional layer. If this is the case, then the approach taken in work on CNNs with pooling may lead to similar conclusions for CNNs with smooth filters (Bietti & Mairal, 2017).

Here we demonstrate empirically that filter smoothness is critical in determining deformation stability. First, in Subsection 4.1 we define a measure of filter smoothness. Second, in Subsection 4.2 we show that forcing filter smoothness at initialization leads to deformation stability. Third, in Subsection 4.3 we show that in a series of synthetic tasks requiring increasing stability to deformations, CNNs learn progressively smoother filters. Finally, in Subsection 4.4 we demonstrate on ImageNet and CIFAR10 that filter smoothness increases as a result of training, even for networks with pooling.

4.1. Measuring filter smoothness

As a measure of filter smoothness, or rather the lack of smoothness in a given layer, we will use the normalized total variation of the filters. In particular, for a 4D (height x width x input filters x output filters) tensor W representing convolutional weights we use the *normalized total variation*:

$$\frac{\mathrm{TV}(W)}{\|W\|_1}$$

where

$$\mathrm{TV}(W) = \sum_{i,j} \|W_{i,j,\cdot,\cdot} - W_{i+1,j,\cdot,\cdot}\|_1 + \|W_{i,j,\cdot,\cdot} - W_{i,j+1,\cdot,\cdot}\|_1$$

where i and j are the indices for the spatial dimensions of the filter.

For filters that are constant over the spatial dimension, the smoothest possible filters by definition, the normalized total variation would be zero. At initialization for a given layer, the expected smoothness is identical across network architectures[1]. Given that this value has an approximately fixed mean and small standard deviation across layers and architectures, we plot this as a single dotted black line in Figure 6 and Figure 5 for simplicity.

4.2. Forcing filter smoothness at initialization leads to deformation stability

To test whether smooth filters lead to deformation stability, we initialize networks with different amounts of filter smoothness and asked whether the amount of smoothness at initialization leads to greater deformation stability. To initialize filters with different amounts of smoothness, we used our usual random initialization[2], but then smoothed the filters by convolving them with Gaussian filters of varying smoothness. Indeed we found that networks initialized with smoother random filters are more stable to deformation (Figure 4), suggesting that filter smoothness contributes to deformation stability.

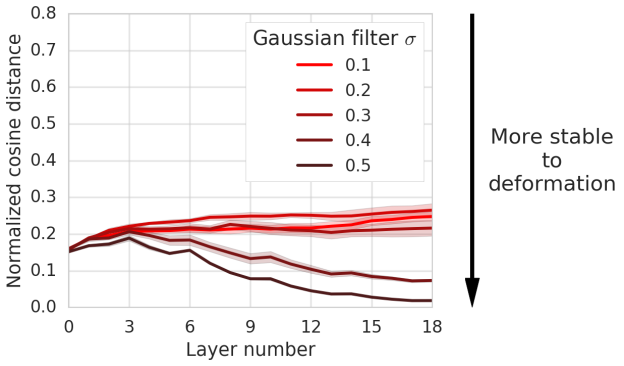


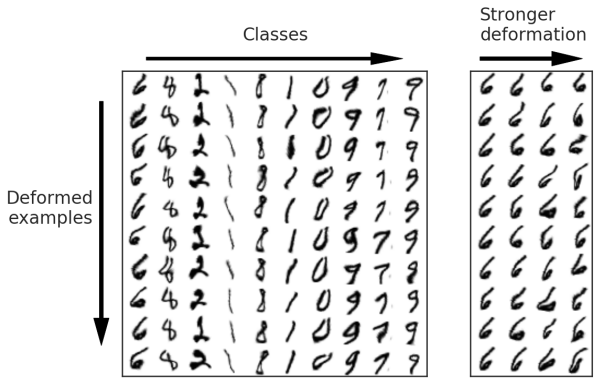
Figure 4. **Initialization with smoother random filters lead to deformation stability.** We smooth filters by convolving with a Gaussian filter with standard-deviation σ and then measure the sensitivity to deformation. As we increase the smoothness of the filters by increasing σ, the representations become less sensitive to deformation. Darker lines are for smoother random filters.

4.3. Requiring increasing stability to deformation in synthetic tasks leads to progressively smoother filters

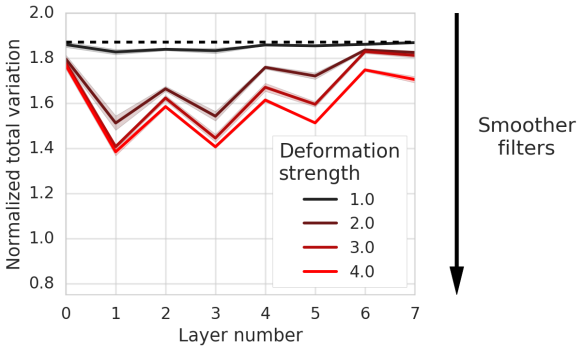
The above result demonstrates that smooth randomly initialized filters lead to greater deformation stability. However

[1]We estimated this average smoothness empirically by resampling filters 10,000 times and found an average smoothness of approximately 1.87 (this differed between layers only in the fourth decimal place) with a standard deviation that depended on the size of the layer, ranging from 0.035 in the first layer to 0.012 for the larger layers.

[2]Truncated normal with standard deviation $1/\sqrt{n_{in}}$, where n_{in} is the number of inputs.



(a) Deformed templates classes for synthetic tasks



(b) Smoothness of learned filters

Figure 5. **Tasks requiring more deformation invariance lead to smoother filters.** (a) We generate synthetic tasks where each class is based on a single MNIST image and within each class examples are generated by applying a random deformation of strength C to this class image. The image on the left is generated using deformations of strength 3. The columns for the image on the right are generated using deformations of strengths 1, 2, 3, 4 respectively. (b) After training, filters from networks trained on tasks where stronger deformations are used are smoother. Dotted black line indicates average value at initialization.

the distribution of learned filters may differ significantly from that of random filters. Therefore we asked whether smoother filters are learned in tasks requiring stability to stronger deformation. To test this, we constructed a set of synthetics classification tasks. In these tasks, each class consisted of deformed versions of a single image as depicted in Figure 5a. The tasks varied in the strength of the deformation used to generate the examples in each class. We then measured the effect of increased intra-class deformation on the smoothness of the filters learned in each task. Consistent with our previous result, we observed that stronger deformations led to smoother filters after training as shown in Figure 5.

4.4. Filter smoothness increases as a result of training on real datasets

Finally, we asked whether filter smoothness increases over the course of training in more realistic datasets. To test this we examined the filter smoothness across layers for a variety of network architectures trained on both ImageNet and CIFAR10. We found that filter smoothness increases as a result of training across all architectures for both ImageNet (Figure 6a) and CIFAR10 (Figure 6b). Interestingly for ImageNet, filters in the upper layers are particularly smooth, suggesting that perhaps in the upper layers precise relative position is less important.

Taken together these results demonstrate a clear relationship between filter smoothness and deformation stability.

5. Filter smoothness depends on the supervised task

In the previous section we demonstrated that smooth filters contribute to deformation stability of CNN representations. However it remains unclear what aspects of the training lead to this filter smoothness and deformation stability. It is possible that smooth filters emerge as a consequence of the distribution $P(X)$ of the input images X. Alternatively, it may be the case that an important factor in determining filter smoothness is the nature of the task itself, i.e. the conditional distribution $P(Y|X)$ of the labels Y given the inputs X.

To test what role $P(X)$ and $P(Y|X)$ play in the smoothness of the learned filters, we trained networks on a modified versions of the CIFAR10 dataset as in Zhang et al. (2016) in which we replace the labels with uniform random labels which are consistent over training epochs. The representations learned under such tasks have been studied before, but not in the context of deformation stability or filter smoothness (Morcos et al., 2018; Achille & Soatto, 2017). Therefore, we analyzed the patterns of deformation stability and filter smoothness of networks trained on random images (modifying $P(X)$) and random labels (modifying $P(Y|X)$ but holding $P(Y|X)$ fixed).

In contrast to networks trained on unmodified datasets, we found that networks with different architectures trained on random labels converged to different patterns of deformation stability across layers (Figure 7). Interestingly, unlike with real labels, this depends on architecture significantly, but does not vary much as a function of the random seed. This suggests that even with random labels, there is still sig-

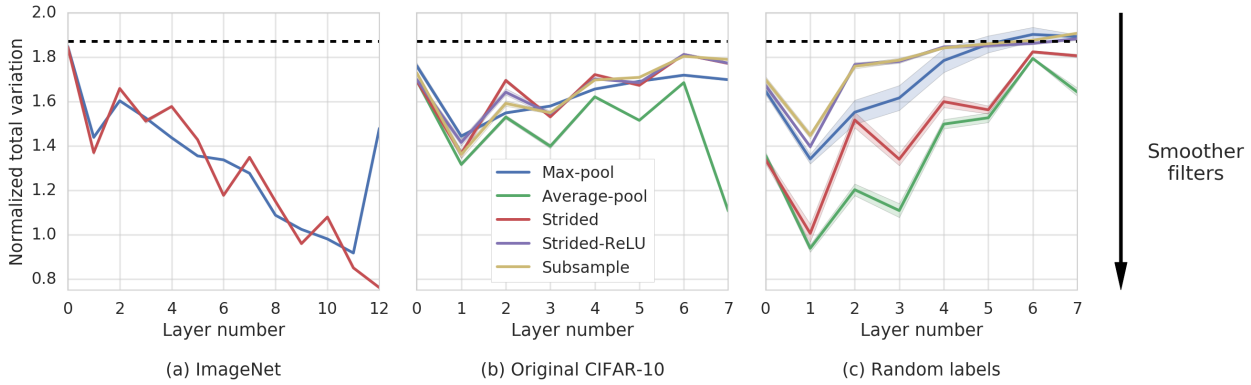


Figure 6. **Training leads to smoother filters.** (a) and (b) After training, the filters are significantly smoother and different architectures converge to similar levels of filter smoothness. (c) When training on random labels the smoothness of filters depends largely on the chosen downsampling layer. Interestingly, the smoothness of filters when training on ImageNet (a) increases from layer to layer, whereas for CIFAR10 (b) the smoothness decreases from layer to layer. Dotted black lines indicate average value at initialization.

nificant structure to the problem so that different networks in a population converge to representations with similar properties. The filters for these networks also converge to different levels of smoothness and this correlates qualitatively with their level of deformation stability as can be seen Figure 6b.

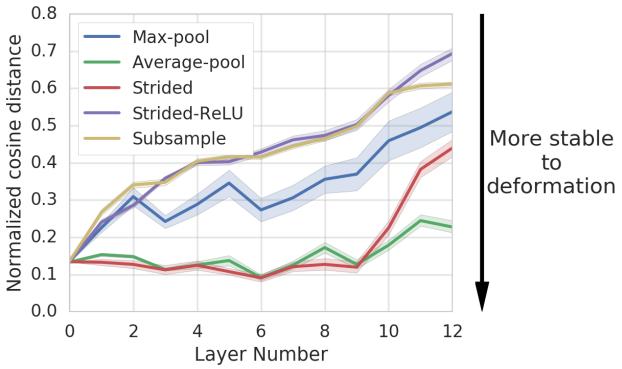


Figure 7. **Deformation stability is architecture dependent when training with random labels.**

6. Discussion

In this work, we have empirically tested a variety of properties associated with deformation stability. We demonstrated that while pooling confers deformation stability at initialization, the pattern of deformation stability across layers learned after training differs significantly from that at initialization and is consistent across network architectures, both with and without pooling. Moreover, the amount of deformation stability often decreases, suggesting that deformation stability is not always helpful. We also found that filter smoothness contributes significantly to achieving deformation stability in CNNs and that the joint distribution of inputs and outputs is important in determining the level of learned deformation stability. Together, these results provide new insights into the necessity and origins of deformation stability and provide an instructive example of how simple properties of learned weights can be investigated to shed light on the inner workings of deep neural networks.

One caveat of this work is that we only focused on deformations sampled from a particular distribution and looked at average sensitivity over these deformations. In future work, it will be interesting to explore similar questions but with the worst case deformations found via maximization of the deformation sensitivity similar to the approach taken in Fawzi & Frossard (2015) and Kanbak et al. (2017).

Another interesting direction would be to understand exactly under what conditions smooth filters lead to deformation stability. While we have presented empirical evidence and an informal argument as to how smooth filters could lead to deformation stability, there is still significant work to be done in evaluating whether this line of argument can be formalized into a mathematical proof. It will also be necessary to find the sufficient conditions under which smooth filters lead to deformation stability.

Finally, our work compares only two points in time, the beginning and the end of training. In future work, it will be interesting to see how the observations we have made change over the course of training. For example, when do filters become smooth? Does this differ across layers and architectures? Is the trajectory toward smooth filters and deformation stability monotone, or are there periods of training where filters become smoother and then periods when the filter smoothness decreases?

References

Achille, Alessandro and Soatto, Stefano. On the emergence of invariance and disentangling in deep representations. *CoRR*, abs/1706.01350, 2017. URL http://arxiv.org/abs/1706.01350.

Bietti, Alberto and Mairal, Julien. Group invariance and stability to deformations of deep convolutional representations. *arXiv preprint arXiv:1706.03078*, 2017.

Bruna, Joan and Mallat, Stéphane. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013.

Duchon, Jean. Splines minimizing rotation-invariant seminorms in sobolev spaces. In *Constructive theory of functions of several variables*, pp. 85–100. Springer, 1977.

Fawzi, Alhussein and Frossard, Pascal. Manitest: Are classifiers really invariant? *arXiv preprint arXiv:1507.06535*, 2015.

Felzenszwalb, Pedro, McAllester, David, and Ramanan, Deva. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8. IEEE, 2008.

Fukushima, Kunihiko and Miyake, Sei. 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.

Giusti, Alessandro, Ciresan, Dan C, Masci, Jonathan, Gambardella, Luca M, and Schmidhuber, Jurgen. Fast image scanning with deep max-pooling convolutional neural networks. In *Image Processing (ICIP), 2013 20th IEEE International Conference on*, pp. 4034–4038. IEEE, 2013.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hubel, David H and Wiesel, Torsten N. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

Kanbak, Can, Moosavi-Dezfooli, Seyed-Mohsen, and Frossard, Pascal. Geometric robustness of deep networks: analysis and improvement. *arXiv preprint arXiv:1711.09115*, 2017.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

LeCun, Yann, Boser, Bernhard E, Denker, John S, Henderson, Donnie, Howard, Richard E, Hubbard, Wayne E, and Jackel, Lawrence D. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pp. 396–404, 1990.

LeCun, Yann, Bengio, Yoshua, and Hinton, Geoffrey. Deep learning. *Nature*, 521(7553):436–444, 2015.

Lenc, Karel and Vedaldi, Andrea. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 991–999, 2015.

Lowe, David G. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pp. 1150–1157. Ieee, 1999.

Mallat, Stéphane. Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65(10):1331–1398, 2012.

Mallat, Stéphane. Understanding deep convolutional networks. *Phil. Trans. R. Soc. A*, 374(2065):20150203, 2016.

Morcos, Ari, Barrett, David G.T., Rabinowitz, Neil C., and Matthew, Botvinick. On the importance of single directions for generalization. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1iuQjxCZ. accepted as poster.

Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, Berg, Alexander C., and Fei-Fei, Li. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Sifre, Laurent and Mallat, Stéphane. Rotation, scaling and deformation invariant scattering for texture discrimination. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pp. 1233–1240. IEEE, 2013.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Springenberg, Jost Tobias, Dosovitskiy, Alexey, Brox, Thomas, and Riedmiller, Martin. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

Wong, Sebastien C, Gatt, Adam, Stamatescu, Victor, and McDonnell, Mark D. Understanding data augmentation

for classification: when to warp? In *Digital Image Computing: Techniques and Applications (DICTA), 2016 International Conference on*, pp. 1–6. IEEE, 2016.

Zhang, Chiyuan, Bengio, Samy, Hardt, Moritz, Recht, Benjamin, and Vinyals, Oriol. Understanding deep learning requires rethinking generalization. *CoRR*, abs/1611.03530, 2016. URL http://arxiv.org/abs/1611.03530.