Measuring (in)variances in Convolutional Networks

Facundo Quiroga[1] ✉, Jordina Torrents-Barrena[2], Laura Lanzarini[1], and Domenec Puig[2]

[1] Instituto de Investigación en Informática LIDI, Facultad de Informática, Universidad Nacional de La Plata, Argentina
fquiroga@lidi.info.unlp.edu.ar,
Website: www.lidi.info.unlp.edu.ar
[2] Intelligent Robotics and Computer Vision Group
Universitat Rovira i Virgili, Spain
Website: http://deim.urv.cat/rivi

Abstract. Convolutional neural networks (CNN) offer state-of-the-art performance in various computer vision tasks such as activity recognition, face detection, medical image analysis, among others. Many of those tasks need invariance to image transformations (*i.e.*, rotations, translations or scaling).
This work proposes a versatile, straightforward and interpretable measure to quantify the (in)variance of CNN activations with respect to transformations of the input. Intermediate output values of feature maps and fully connected layers are also analyzed with respect to different input transformations. The technique is applicable to any type of neural network and/or transformation. Our technique is validated on rotation transformations and compared with the relative (in)variance of several networks. More specifically, ResNet, AllConvolutional and VGG architectures were trained on CIFAR10 and MNIST databases with and without rotational data augmentation. Experiments reveal that rotation (in)variance of CNN outputs is class conditional. A distribution analysis also shows that lower layers are the most invariant, which seems to go against previous guidelines that recommend placing invariances near the network output and equivariances near the input.

Keywords: transformation invariance,rotation invariance, neural networks, variance measure, MNIST dataset, CIFAR10 dataset, Residual Network, VGG Network, AllConvolutional Network.

1 Introduction

Convolutional neural networks (CNNs) provide outstanding results for several computer vision applications [4]. Nevertheless, CNNs have difficulty learning good representations when objects appear rotated in many domains such as textures or galaxy classification, among other domains [4].

Dealing with rotations, or other transformations, requires the network f to be invariant or equivariant to the corresponding transformations Ts. Thus, f is invariant to T if altering the input x with T does not change the network output. In other words, $f(T(x)) = f(x)\ \forall\ x$. Alternatively, a network is equivariant to T if its output changes predictably when x is transformed by T. Formally, it is equivariant if there exists a function T' such that $\forall\ x$, we have $f(T(x)) = T'(f(x))$ [4]. Invariance is a special case of equivariance in which T' is simply the identity transformation. Analysing if f is equivariant to a T that operates on x requires finding a corresponding T' that operates on outputs [15]. Since CNNs are approximately invertible (sufficient condition) [7], the existence of T' is very likely. However, characterizing T' requires assuming its functional form and estimating its parameters [15].

Typical CNNs rely on feed-forward architectures with a series of convolutional layers followed by one or two dense layers. These models, commonly trained with stochastic gradient descent and without data augmentation, cannot learn invariances or equivariances to rotations [16,1]. Feed-forward networks exclusively composed of dense layers can approximate smooth functions given enough parameters. They can even learn arbitrary invariance and equivariance properties with heavy data augmentation[16]. Besides, some CNN models avoid dense layers [18] due to their questionable efficiency at dealing with certain inputs (*e.g.*, images). On the contrary, convolutional layers, by definition, are translation equivariant and much more efficient, but they are not invariant nor equivariant to other transformations [4]. Since they have a lower representational power than dense layers, they cannot become so even with data augmentation.

Recently, models such as Transformation-Invariant Pooling [13], Deep Symmetry Networks [6], Steerable CNNs [3] were proposed to provide convolutional layers with rotation invariance or equivariance. Most schemes were based on modifying the filters so that they were invariant, or employing a set of equivariant filters which were subsequently pooled to supply invariance [3]. Other approaches made multiple predictions with rotated input versions to subsequently combine them [6]. For instance, Spatial Transformer Networks learned a canonical representation of the input [10].

Alternatively, data augmentation is also used to achieve partial invariance to geometric transformations of the input and improve generalization accuracy. While applied transformations are often mild, full rotation invariance is possible by including all transformation angles or deformations. This approach was studied for Deep Restricted Boltzmann Machines [14], HOGs and CNNs [15,16,19]. Although employed architectures are simpler, they generally require more training epochs to explore the wide space of rotated inputs. Thus, given sufficient computational budget, typical CNNs with data augmentation could learn the same set of filters that other models include by design [19].

In both cases, the network mechanisms to learn equivariant or invariant representations are not well-understood. It is still unclear whether the model or the data augmentation provides the invariance [3,13,10]. Many proposed layers are individually invariant or equivariant, but no analysis was reported to under-

stand how networks as a whole encode such properties. Several authors studied which methods work best to achieve invariance [16,19], but no guiding principle in their analysis was employed except comparing the output accuracy. To the best of our knowledge, no works measure the internal invariance or equivariance of the network.

In this work, we shed some light on how and where CNNs represent and learn invariances. Notice that invariance can be readily estimated by measuring the changes of the network's outputs through the traditional *variance*. Following this principle, we define V to quantify the variance of a neural network with respect to a set of transformations. Our measure V can be computed not only in the output layers but in the internal layers and activations. This allows to visualize and quantify how invariant a network is as a whole and by layers or individual activations, thus providing insights about how invariance is encoded inside current CNNs. The method is applicable to any neural network, irrespective of its design or architecture, and any set of transformations.

2 Related work

There are few works in the literature that measure invariance or equivariance in neural networks. Nevertheless, previous attempts to quantify invariance were mainly performed on translation and rotation transformations.

The work in [15] evaluated the equivariance of the internal convolutional representations with respect to a transformation T of the input. The proposed method assumed that the corresponding transformation T' acting on outputs was affine, and used empirical risk minimization to obtain the T' parameters once the network was trained. A different transformation T' was then estimated for each layer using the total network error as loss. A particular distance [15] from A_T to the identity matrix was utilized as an (in)variance measure of the layer's representation. Although this approach measures the equivariance, it *i)* only deals with affine types, which limits its applicability to convolutional layers as a spatial correspondence for the affine map is needed, *ii)* requires an arduous optimization process, and *iii)* is not simple to interpret. Other works [2] modified the loss function to improve equivariance and invariance capabilities. However, the authors only estimated the loss impact with the technique of [15] in the last network layer.

Measuring invariance to transformations was also tackled from an adversarial perspective [5], confirming that simple rotations or translations can have a big impact on performance. In [16,11,19], the effect of using different data augmentation schemes and CNNs architectures was measured and compared. Specifically, the translation sensitivity map developed in [11] related the classifier accuracy with the center position of the object in the image. Equivalent 1D plots were employed in [16,11] to evaluate the rotation and other transformation invariances. Moreover, [1] studied the lack of equivariance in some CNNs by relating the Shannon sampling theorem to strided convolutions.

With the sole exception of [15], all aforementioned methods were focused on measuring how the network performance varies according to the learning algorithm, architecture or data augmentation scheme, disregarding the internal representation.

3 Proposed V variance measure

Invariance in neural networks is often measured just in the output layer by reporting the final performance (accuracy). Instead, we are interested in analyzing the invariance of the intermediate values or activations. Therefore, we propose a measure named V to quantify the (in)variance of the individual network activations with respect to a transform of the input. We assume that an activation is invariant when $V \sim 0$. By analyzing the activation's (in)variance, we can characterize the network's distribution of invariances in a fine-grained fashion. Variance at higher levels (*i.e.*, feature maps, spatial regions, layer types) can be calculated by aggregating the individual variances.

We denote the activation value as $a(x)$, where x is the network input. Note that a is not just an activation function such as $ReLU$ or $tansig$, but an intermediate value or activation. Note that a may also be a resulting element of a matrix multiplication, convolutional filter, $ReLU$ activation function, etc. To evaluate $V(a)$, we assume a finite set of samples X and transformations T. For example, X may be a set of images and T a set of rotations around the center of those images. The same definition works for other transformation and inputs.

We define the auxiliary set $a(X,T) = \{a(t(x)) \mid t \in T, x \in X\}$, which contains the activation values of a for all combinations of samples and transformations in X and T. Armed with this definition, we define the following (naive) way to calculate the variance (see Equation 1):

$$V_{naive}(a,X,T) = Var(a(X,T)), \tag{1}$$

where $Var(X) = \frac{\sum_{x \in X}(x - Mean(X))^2}{n-1}$ and $Mean(X) = \frac{\sum_{x \in X} x}{n}$ are the standard sample variance and mean operators.

The main problem with this definition is that it mixes the variance generated by the randomness of the samples with the variance of the transformation. Instead, we calculate the variance generated by the transformation of a single sample, and then average the variance over the set of all samples (see Equation 2):

$$V_{transformation}(a,X,T) = Mean(\{Var(a(x,T) \mid x \in X\}) \tag{2}$$

While $V_{transformation}$ captures the variance we are interested in, comparing its value between different layers or even activations in the same layer is still hard, since values may differ significantly in scale. However, we can calculate a normalizing factor based on the variance computed over the samples (not transformations) (see Equation 3):

$$V_{sample}(a,X,T) = Mean(\{Var(a(X,t) \mid t \in T\}) \tag{3}$$

Equation 4 formulates the normalized transformation variance V:

$$V(a, X, T) = \frac{V_{transformation}(a, X, T)}{V_{sample}(a, X, T)} \tag{4}$$

Dividing by V_{sample} makes the magnitude order of V adimensional and comparable between layers. This expression is valid whenever $V_{sample}(a, X, T) > 0$ or $V_{sample}(a, X, T) = V_{transformation}(a, X, T) = 0$. Given a large enough X and T, in practice it is very hard to find cases for which both $V_{sample}(a, X, T) = 0$ and $V_{sample}(a, X, T) > 0$ hold. In those cases, however, we set $V(a, X, T) = +\infty$.

3.1 Stratified V

For categorization problems with a set of classes C, it is useful to measure the per class variance $V(a, X_c, T)$, where X_c is the set of samples belonging to class c. This shows if the invariance distribution is class specific.

We can also hypothesize that activation variances are different between classes, thus an alternative measure named $V_{stratified}$ can be defined (see Equation 5):

$$V_{stratified}(a, X, T) = Mean(\{V(a, X_c, T) \mid c \in C\}) \tag{5}$$

Both $V_{stratified}(a, X, T)$ and $V(a, X_c, T)$ can assess if the invariance structure of a network is dependent on the class.

3.2 V for convolutional feature maps

A convolutional layer outputs fm feature maps of size (w, h). The number of individual activations is $fm \times w \times h$, which can be too large. In those cases, we measure the variance of each feature map by aggregating the variance of the spatial dimensions. Given a feature map Fm such that $Fm(i, j)$ is the activation in the i, j spatial coordinates, we define $V_{fm}(Fm, X, T)$ as (see Equation 6):

$$\begin{aligned} V_{f,transformation}(F, X, T) &= Sum(\{V_{transformation}(F(i, j), X_c, T) \mid \forall i, j\}) \\ V_{f,sample}(F, X, T) &= Sum(\{V_{sample}(F(i, j), X_c, T) \mid \forall i, j\}) \\ V_f(F, X, T) &= \frac{V_{f,transformation}(a, X, T)}{V_{f,sample}(a, X, T)} \end{aligned} \tag{6}$$

We aggregate the variances of the feature map with a Sum function, so that V represents the total variance. Aggregation is performed before normalization to remove the dependence on the size of the feature map with the division. Since feature maps are generally sparse, given that filters may be active only in certain spatial regions, aggregating activations with a $Mean$ function instead of Sum would significantly underestimate the variance of the feature map. Other aggregation functions such as Max suffer from similar problems.

3.3 Visualization of the distribution of invariances across the network via V

Calculating V on each activation independently enables the visualization of the distribution of invariances across the network at a glance. This performs qualitative analysis and guide research.

Activation variance can be visualized in various manners. For instance, Figure 1 displays a heatmap of the activations' variance in a simple CNN trained on MNIST.

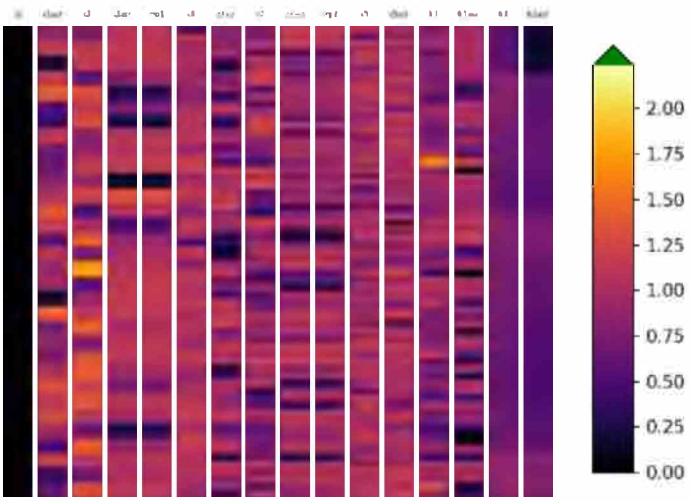


Fig. 1: Sample heatmap showing V for each activation of a CNN. Columns correspond to layers, and each column cell corresponds to a different layer activation. Greater values indicate more variance.

4 Experiments and Discussion

This section shows the experiments performed to validate the designed measure. The code to repeat the experimentation [3] and the library to calculate V [4] are publicly available online.

4.1 Methodology

The experiments were conducted on both CIFAR10 and MNIST datasets to provide an easy interpretation of the results. The following CNN architectures were selected to test the proposed approach: *i)* a simple CNN as baseline *ii)*

[3] https://github.com/facundoq/rotational_variance
[4] https://github.com/facundoq/inmeasure

VGG16 [17], *iii)* AllConvolutional (without dense layers) [18], and *iv)* ResNet (with residual connections and bottlenecks) [8]. We ignored the activations of the Batch Normalization layers [9] since these are linear transformations of the previous activations and therefore the variance is strongly correlated.

We assessed V on each model/dataset combination. For each combination, two networks were trained: i a rotated network with rotational data augmentation derived from random and uniform angles within the range $0^\circ - 360^\circ$, and ii an unrotated network with no data augmentation. Both were trained with the ADAM [12] optimizer using a learning rate of 10^{-3}. Rotated networks were trained for approximately twice the number of epochs than unrotated. Note that only test samples were employed to calculate V.

Rotated and unrotated models trained on MNIST obtained an accuracy of $\sim 99\%$. CIFAR10 unrotated models achieved accuracies between 65% and 75%, whereas rotated versions performed slightly worse (6% $\downarrow$).

4.2 Experiment 1: Validation of V

To validate if V correlates with the invariance of the model, we calculated the global average variance of both rotated and unrotated models.

Table 1 shows the obtained results for each pair. In all cases, the variance of unrotated models is greater, which confirms the viability of V as a transformational variance measure. For VGG and ResNet models trained on MNIST, the difference in V is small, which may be due to bigger models have a more complex representation and may capture finer detail. Indeed, both VGG and ResNet may be overly complex for the MNIST dataset.

Table 1: Comparison of global average variances computed with V for each pair of model and dataset. Greater values indicate less invariance.

Dataset	Version	AllConvolutional	SimpleConv	ResNet	VGG
MNIST	rotated	0.47	0.59	0.70	0.68
MNIST	unrotated	0.68	0.80	0.74	0.77
CIFAR10	rotated	0.44	0.34	0.54	0.50
CIFAR10	unrotated	0.73	0.65	0.82	0.66

Table 2 shows the obtained results with $V_{stratified}$. In general, variance values are significantly higher for the stratified version. This is because V_{sample} is lower, since by calculating the variance for each class independently, the inter-class variance of the representations is ignored. Therefore, networks learn different invariant representations for each class (*i.e.*, the invariance is class-specific).

4.3 Experiment 2: Variance distribution across layers

Deep neural networks build increasingly higher level and more complex representations as the depth increases. It has been argued that there is a similar pattern

Table 2: Comparison of global average variances computed with $V_{stratified}$ for each pair of model - dataset. Greater values indicate less invariance.

Dataset	Version	AllConvolutional	SimpleConv	ResNet	VGG
MNIST	rotated	0.77	0.92	0.97	0.97
MNIST	unrotated	0.96	1.1	1.0	1.0
CIFAR10	rotated	0.51	0.37	0.59	0.54
CIFAR10	unrotated	0.83	0.71	0.89	0.72

with respect to equivariance and invariance for classification problems [4]; lower layers should be equivariant to preserve the multiplicity of representations, and the last layers should be invariant to perform the final classification.

We test this hyphotesis by calculating the average value of V for each of the layers of the models. We included all activations from all layers, except for BatchNormalization layers. Figure 2 shows such values for both stratified and normal variants of V, as well as rotated and unrotated datasets (blue and red, respectively).

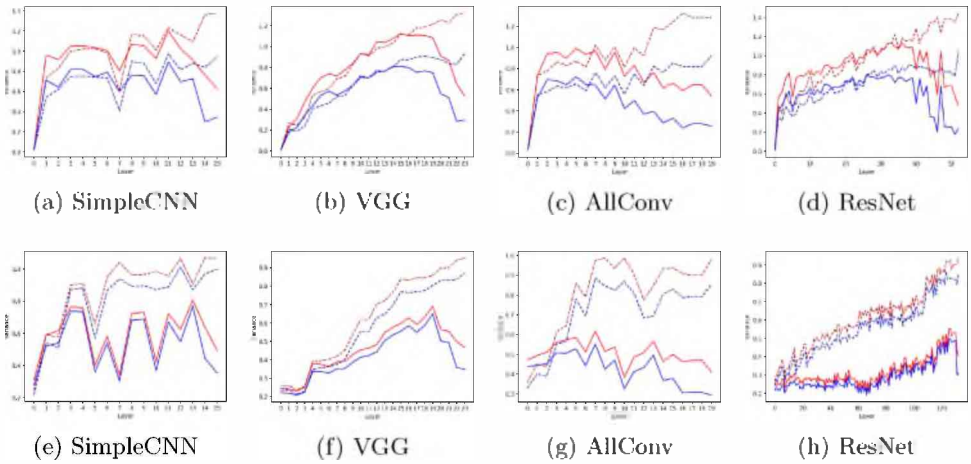
For models trained with unrotated data, the variance increases quasi monotonically, which suggests that higher level activations are more susceptible to rotations of the input. Indeed, since these networks where not trained with rotated data, their resulting dynamics when tested on rotated data seems essentially random, with cumulative errors in representation building up through the layers.

For MNIST (figure 2(a) to (d)), the distribution of invariances is not significantly different between rotated and unrotated models, except for the last layers. These final layers must then code the invariance, as suggested by [4]. Nonetheless, this trend decreases as the model increases in complexity and depth (left to right). Indeed, ResNet models (figure 2 (d)) show a significant decrease in variance beyond layer 35 for the rotated models.

For CIFAR10, however, models trained with rotated data show a general pattern of invariant lower layers, variant middle layers, and invariant final layers. This would suggest that equivariance in lower layers is not necessary or a pattern that emerges naturally. The middle layers are possibly coding equivariant representations of the objects, based on the more invariant features generated by lower layers.

In the case of the simple CNN (figure 2 (a) and (e)), it clearly seen that the variance of an activation layer is always higher than that of the previous convolutional layer. This trend is still visible but less noticeably on the other models, in both datasets.

By comparing the values of the stratified and normal versions of V, the figures also confirm the relationship between both variants mentioned in subsection 4.2. Furthermore, we can see that $V_{stratified}$ is lower than V for all layers, not just globally.



(a) SimpleCNN (b) VGG (c) AllConv (d) ResNet

(e) SimpleCNN (f) VGG (g) AllConv (h) ResNet

Fig. 2: Average V for different layers for MNIST (top row), CIFAR10 (bottom row), and all models. Dashed lines show results for models trained on unrotated datasets, while solid lines correspond to rotated datasets. Blue lines show results for the stratified version of V, while red lines indicate the normal version.

4.4 Experiment 3: Class conditional invariance

We analysed $V(a, X, T)$ for the softmax output layer of the networks, but for each class separately. Figure 3 shows heatmaps of this variance for MNIST and Figure 4 shows the same for CIFAR10. We used the normal, non-stratified variant of V, but the stratified version shows similar results.

In each heatmap, each column corresponds to the variance of the softmax, for the samples of different classes. That is, each column c shows $V(a_1, X_c, T)$, $\ldots, V(a_C, X_c, T)$, where X_c is the set of samples of class c, and $a_1, \ldots, a_C$ are the softmax outputs of the network. Therefore, the entry in row r and column c is the variance of the r element of the softmax when evaluated with samples of class c.

The first row of figure 3 shows that, for all models trained on rotated MNIST, the heatmap has a diagonal structure. This suggests that the softmax specific for a class is more invariant with respect to that class than all others. In other words, the invariance is class conditional; the networks are not learning to be invariant in all their outputs, just in those corresponding to the expected class. On the other hand, the output for class 1 results in a high variance for all classes, which indicates that the network has difficulties representing the digit 1 in rotated positions, not just recognizing it.

For unrotated MNIST (figure 3, row 2), this situation does not arise; the distribution of invariances shows that networks learn to be slighly invariant to class 0 because it has a natural rotational symmetry. The variance heatmaps for unrotated CIFAR10 (figure 4) show a similar lack of structure.

In the case of rotated CIFAR10, the heatmaps do not show a well defined diagonal structure. This could be due to the fact that models trained on CIFAR10

only obtained about 70% accuracy, while on MNIST this was always approximately 99%. On the other hand, this phenomena could be due the networks being overfit on MNIST.

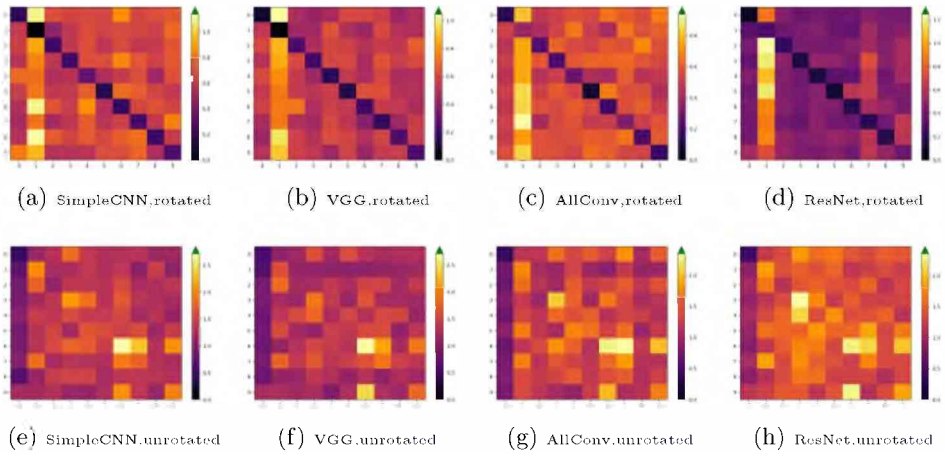


Fig. 3: Comparison of V for the output layer, by class, for MNIST (rotated/unrotated) and all models.

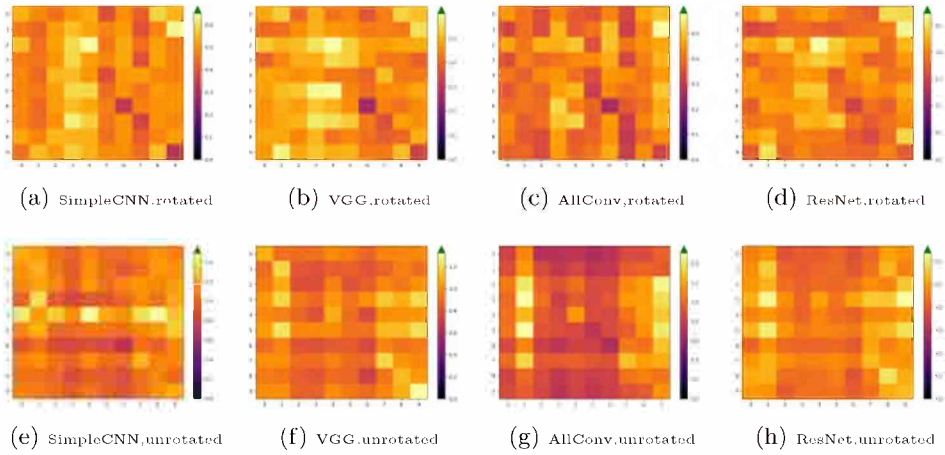
5 Conclusion

In this work, we present a flexible, simple and understandable measure to quantify the (in)variance of individual CNN activations. Our measure is completely adaptable in terms of network architectures, layer types, arbitrary inputs and transformations. We also propose a modified version of the variance measure to analyze convolutional feature maps.

We evaluated the proposed measure V on well-known models (*e.g.*, ResNet, VGG and AllConvolutional) and datasets (*e.g.*, CIFAR10 and MNIST) for rotation transformations. We also validated the correlation of the measure with the invariances learned by the networks. Our main findings suggest that: *i)* networks learn class specific invariances, and *ii)* the invariance distribution of networks trained with / without data augmentation is similar. The invariance increases according to the network's depth, except for the final layers where augmented models have a sharp increase in invariance.

Many modified layers can generate equivariant feature maps. However, if some of them are found to be approximately invariant, it would suggest that equivariance is not needed. Thus, we believe that V will guide the development of new and more robust CNN models.

Future work will qualitatively assess our measure through the representation (*i.e.*, feature map activations and filters) of what networks learn. In addition,



(a) SimpleCNN,rotated (b) VGG,rotated (c) AllConv,rotated (d) ResNet,rotated

(e) SimpleCNN,unrotated (f) VGG,unrotated (g) AllConv,unrotated (h) ResNet,unrotated

Fig. 4: Comparison of V for the output layer, by class, for CIFAR10 (rotated/unrotated) and all models.

we will experiment with more complex datasets and affine transformations, specialized models with equivariant or invariant layers, and other learning problems (*e.g.,* segmentation). Finally, we will extend the experimentation to successfully describe the network invariance before training (random networks), and after training and fine-tuning.

6 Acknowledgments

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU used in this research.

References

1. Azulay, A., Weiss, Y.: Why do deep convolutional networks generalize so poorly to small image transformations? CoRR abs/1805.12177 (2018), http://arxiv.org/abs/1805.12177
2. Christopher Tensmeyer, T.M.: Improving invariance and equivariance properties of convolutional neural networks. International Conference on Learning Representations (2017), https://openreview.net/forum?id=SyBPtQfAZ
3. Cohen, T.S., Welling, M.: Steerable CNNs. arXiv:1612.08498 [cs, stat] (Dec 2016), http://arxiv.org/abs/1612.08498, arXiv: 1612.08498
4. Dieleman, S., De Fauw, J., Kavukcuoglu, K.: Exploiting Cyclic Symmetry in Convolutional Neural Networks. arXiv:1602.02660 [cs] (Feb 2016), http://arxiv.org/abs/1602.02660, arXiv: 1602.02660
5. Engstrom, L., Tsipras, D., Schmidt, L., Madry, A.: A rotation and a translation suffice: Fooling cnns with simple transformations. CoRR abs/1712.02779 (2017)

6. Gens, R., Domingos, P.M.: Deep Symmetry Networks. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 27, pp. 2537–2545. Curran Associates, Inc. (2014), http://papers.nips.cc/paper/5424-deep-symmetry-networks.pdf
7. Gilbert, A.C., Zhang, Y., Lee, K., Zhang, Y., Lee, H.: Towards understanding the invertibility of convolutional neural networks. Proceeding of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI) (2017)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
9. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR abs/1502.03167 (2015), http://arxiv.org/abs/1502.03167
10. Jaderberg, M., Simonyan, K., Zisserman, A., Kavukcuoglu, K.: Spatial Transformer Networks. arXiv:1506.02025 [cs] (Jun 2015), http://arxiv.org/abs/1506.02025, arXiv: 1506.02025
11. Kauderer-Abrams, E.: Quantifying translation-invariance in convolutional neural networks. CoRR abs/1801.01450 (2018), http://arxiv.org/abs/1801.01450
12. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1412.6980
13. Laptev, D., Savinov, N., Buhmann, J.M., Pollefeys, M.: TI-POOLING: transformation-invariant pooling for feature learning in convolutional neural networks. CoRR abs/1604.06318 (2016), http://arxiv.org/abs/1604.06318
14. Larochelle, H., Erhan, D., Courville, A., Bergstra, J., Bengio, Y.: An Empirical Evaluation of Deep Architectures on Problems with Many Factors of Variation. In: Proceedings of the 24th International Conference on Machine Learning. pp. 473–480. ICML '07, ACM, New York, NY, USA (2007), http://doi.acm.org/10.1145/1273496.1273556
15. Lenc, K., Vedaldi, A.: Understanding image representations by measuring their equivariance and equivalence. arXiv:1411.5908 [cs] (Nov 2014), http://arxiv.org/abs/1411.5908, arXiv: 1411.5908
16. Quiroga, F., Ronchetti, F., Lanzarini, L., Bariviera, A.F.: Revisiting data augmentation for rotational invariance in convolutional neural networks. In: International Conference on Modelling and Simulation in Management Sciences. pp. 127–141. Springer (2018)
17. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv e-prints arXiv:1409.1556 (Sep 2014)
18. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.A.: Striving for simplicity: The all convolutional net. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings (2015), http://arxiv.org/abs/1412.6806
19. Srivastava, M., Grill-Spector, K.: The effect of learning strategy versus inherent architecture properties on the ability of convolutional neural networks to develop transformation invariance. CoRR abs/1810.13128 (2018), http://arxiv.org/abs/1810.13128