# NEURAL NETWORKS PRESERVE INVERTIBILITY ACROSS ITERATIONS: A POSSIBLE SOURCE OF IMPLICIT DATA AUGMENTATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Determining what kind of representations neural networks learn, and how this may relate to generalization, remains a challenging problem. Previous work has utilized a rich set of methods to invert layer representations of neural networks, i.e. given some reference activation $\Phi_0$ and a layer function $r_\ell$, find $x$ which minimizes $||\Phi_0 - r_\ell(x)||^2$ . We show that neural networks can preserve invertibility across several iterations. That is, it is possible to interpret activations produced in some later iteration in the context of the layer function of the current iteration. For convolutional and fully connected networks, the lower layers maintain such a consistent representation for several iterations, while in the higher layers invertibility holds for fewer iterations. Adding skip connections such as those found in Resnet allows even higher layers to preserve invertibility across several iterations. We believe the fact that higher layers may interpret weight changes made by lower layers as changes to the data may produce implicit data augmentation. This implicit data augmentation may eventually yield some insight into why neural networks can generalize even with so many parameters.

## 1 INTRODUCTION

Determining what kind of representations neural networks learn, and how this may relate to generalization, remains a challenging problem. Previous work has utilized a rich set of methods to compare representations learned by different networks . Lenc & Vedaldi (2015) compared properties of representations in terms of their equivariance, equivalence, and invariance. Dosovitskiy & Brox (2016) reconstruct images by training neural networks to 'invert' feature representations. Mahendran & Vedaldi (2015) also invert such representations, but they use GD on a loss function with a prior to do so. Raghu et al. (2017) find the most important directions in representation space, which allows them to compare representations produced by different networks. Kornblith et al. (2019) suggest using a similarity index equivalent to central kernel alignment to compare neural network representations.

The works described above yielded fruitful ways to determine if two networks were learning equivalent representations, to determine when in training network representations converge, and to determine the function of higher versus lower layers in a deep network.

We utilize some of the approaches above to explore whether the invertibility of image representations could shed light on neural network generalization.

Consider a neural network. Suppose $r_\ell^t(x) : \mathbb{R}^d \to \mathbb{R}^{m_\ell}$ is the $\ell$th layer's layer activation function, which takes in some input $x_i \in \mathbb{R}^d$ and the time $t$ weights, and applies a forward pass to compute a vector of length $m_\ell$ where $m_\ell$ is the number of neurons in the $\ell$th layer. Suppose layer $\ell$ updates its weights according to a GD update. We analyze the following: is it plausible that layer $\ell + 1$ could believe that $r_\ell$ has remained fixed, and interpret the change in layer $\ell$ activations from $r_\ell^t(x_i)$ to $r_\ell^{t+1}(x_i)$ as a perturbation on the input $x_i$, even when $r_\ell$ has changed? If so, this could increase the *effective* size of the training set for layer $\ell + 1$, and may tie to generalization.

We primarily use the inversion method of Mahendran & Vedaldi (2015). We invert some reference activation, $\Phi_0$ produced by a layer function $r_\ell^{t+p}$ at a time $t + p$, but minimize the L2 norm

$||\Phi_0 - r_\ell^t(x)||$, where $x$ is the variable we are optimizing and we apply the time $t$ layer function to it.

We interpret this optimization as a way to determine if the time $t + p$ activations can be interpreted as a perturbed image produced by the time $t$ layer $\ell$ function. Crucial to our intuition is the fact that layer $\ell + 1$ is trained on a stream of layer $\ell$'s activations, but does not need to know the specifics of the layer function $r_\ell$. This type of 'encapsulation' may allow layer $\ell + 1$ to interpret the change in the composite value $r_\ell(x_i)$ (at least partially) as a change in $x_i$, even if $x_i$ has not changed. If such an interpretation is possible, we reason that neural networks may experience a larger effective set of training images than the size of the training set itself. Since generalization bounds tend to decay with increased training set size, this may give us a hint as to why neural networks are able to generalize well, even given overparametrization and a lack of explicit regularization.

Our findings are as follows:

- We show that neural network can preserve invertibility across several iterations. That is, it is possible to interpret activations produced in some later iteration in the context of the function of the current iteration
- For convolutional and fully connected networks, the lower layers maintain such a consistent representation for several iterations, while in the higher layers invertibility holds for fewer iterations
- Adding skip connections such as those found in Resnet allows even higher layers to preserve invertibility across several iterations, and seems to affect the quality of the data augmentation.
- We investigate the role of capacity in a simple setting.

## 2 Methods and Definitions

### 2.1 Data Augmentation

Many techniques have been proposed to augment data during training including interpolating across deep feature space Upchurch et al. (2017), GAN based methods Denton et al. (2015) Ratner et al. (2017) Bowles et al. (2018). Whereas these methods intend to modify training to produce data augmentation, we seek to determine what intrinsic properties of training neural networks may already lead to data augmentation. We also seek to determine whether modifications to the architecture can influence this implicit data augmentation.

### 2.2 Layer Representations

Layer representations have a variety of interpretations and intended uses. Some previous work has found that the second to last layer of a neural network can be used as a representation for downstream tasks. Other work has sought to determine whether two neural networks learn essentially the same representations Li et al. (2016). Liao et al. (2016) introduce a regularizer that encourages layer representations to cluster.

Instead of focusing on these aspects of representation, our work more closely follows Mahendran & Vedaldi (2015) Cohen et al. (2020). We consider representations throughout training of a single neural network, and seek to understand how these representations may influence the learning of that single neural network during training. Let $m_\ell$ be the number of neurons in the $\ell$th layer in a neural network. Let $r_\ell^t : \mathbb{R}^d \to \mathbb{R}^{m_\ell}$ be the function that takes an input $x \in \mathbb{R}^d$ and maps it to the sequence of activations produced by the $\ell$th layer of a neural network using the weights at time $t$ during training, $\mathbf{w}_t$. We assume that data points $x$ lie in some training set $\mathcal{T}$ that will be iterated over many times during training. We consider batched training, where the training data are divided into batches $B_1, ..., B_M$. The $\ell + 1$th layer in the neural network observes the following sequence of activations from the layer below:

$$\{r_\ell^0(B_{i_0}), r_\ell^1(B_{i_1}), r_\ell^2(B_{i_2}), ...\}$$

where $B_{i_0}$ represents the batch stochastically drawn at time 0. Notice that when training layer $\ell+1$, it observes the composition of $r_\ell^j$ and $B_{i_j}$, i.e. it observes the $\ell$ layer activations, but may not know the exact structure of the function $r_\ell^j$. We seek to answer the following question: is it plausible that layer $\ell+1$ learns the distribution of layer $\ell$ in a lagged way? That is, is it plausible, from the perspective of layer $\ell+1$ that some $\ell$ layer activation $r_\ell^t(B_{i_t})$ was produced by function $r_\ell^q$ in some previous iteration $q < t$? Is it possible to invert $r_\ell^t(B_{i_t})$ to some reasonable (set of) input image(s) *assuming it was produced by some earlier function* $r_\ell^q$?

This last assumption is primarily where our work differs from previous approaches.

### 2.3 Gradients of neural networks

Let $w_{k,j}^{\ell+1}$ denote the $k$th neuron in the $\ell+1$th layer that is connected to the $j$th neuron in layer $\ell$. We can write the gradient of the $k$th neuron in the $\ell+1$th layer as

$$\frac{\partial l}{\partial w_k^{\ell+1}} \propto \sum_{i=1}^{P} \frac{\partial \ell}{\partial f_p} \frac{\partial f_p}{\partial r_{\ell+1}(x_i)} \frac{\partial r_{\ell+1}(x_i)}{w_k^{\ell+1}} \tag{1}$$

where $f_p$ is the $p$th coordinate of the output layer. If the network is a ReLU network, the last partial $\frac{\partial r_{\ell+1}(x_i)}{w_k^{\ell+1}}$ can be simplified to the $p$th coordinate of $r_\ell(x_i)$. In other words, we can see using the calculation above that is the overall activations produced by layer $\ell$ that inform the training of layer $\ell+1$. Layer $\ell+1$ does not need to know the intricate details of $r_\ell$.

The gradient of the neural network is proportional to the activations of the underlying layer, but does not reflect the underlying structure which produced those activations.

### 2.4 Slowly changing distributions

Early work Bartlett et al. (2000) analyzed learning from a distribution that is changing slowly. Later work Sugiyama et al. (2007) analyzed covariate shift, which is the case that $p(y|x)$ for training and test follow the same distribution, but $p(x)$ for training and test follow a different distribution. The setting they consider is different from our setting, but we are also concerned with whether the sequence $\{r_\ell^0(B_{i_0}), r_\ell^1(B_{i_1}), r_\ell^2(B_{i_2}), ...\}$ is changing quickly. Since layer $\ell+1$ must learn from the layer $\ell$ activations, we are interested in seeing if layer $\ell$ represents some given image, $x_0$, in a consistent way across time. Traditional definitions of slowly changing distributions looked at the difference in probability mass assigned to a particular input. Analogously, we look at the distance between the representation vector for $x_0$ at time t versus now. However, we will find that taking simple norms of the representations themselves does not directly correspond to invertibility of the image.

### 2.5 Image representation inversion

We give a brief overview of the method in Mahendran & Vedaldi (2015). For an optimization variable of the same shape as the input images, $x$, the inversion procedure takes in some reference activation, $\Phi_0$ produced e.g. by a layer function applied to an image, and a function, $f$. In their case, $f$ was the function used to produce $\Phi_0$. Their loss function consists of three parts. The first is the L2 difference between $||\Phi_0 - f(x)||^2$. The second is the norm of $x$. The third is the total variation of the optimization variable $x$. They use gradient descent with momentum to minimize this objective. The inversion of a layer representation is not unique, but when optimizing across different iterations, we always hold the initialization fixed so that this is not the source of variability in the images.

### 2.6 Resnets

Resnets He et al. (2016) are a subset of HighWay networks that utilize skip connections.

One traditional understanding of the success of these architectures is that they alleviate the vanishing and exploding gradient problem discussed in Bengio et al. (1994). Huang et al. (2018) view Resnets through the lens of boosting. Balduzzi et al. (2017) find that skip connections help because gradients. across different data points in higher layers begin to resemble white noise. Orhan & Pitkow (2018)

found that skip connections help eliminate singularities caused by linear dependence of the nodes and the permutation symmetry of nodes in a layer. Veit et al. (2016) find that much of the gradient comes from short paths in the neural network. Zaeemzadeh et al. (2020) find that skip connections preserve the norm of the gradient.

Many of the previous approaches to understand Resnets focused on the quality of the gradients, in terms of the gradient norm, or the distribution of the gradient. We look at this problem in terms of the representations produced by each layer. We find that Resnets allow activations from later iterations to be inverted according to the current $r_\ell$ function even for higher layers. This property does not seem to hold for fully connected or convolutional neural networks without skip connections.

A chief feature of data augmentation is that it should alter the image without accidentally altering its true label. (For the implicit data augmentation we describe, an image's label is constant throughout training, so that each image must be perturbed to a different image in the same class. Therefore we do not consider the case of label-mixup.) For example, mixing cat features into a car image would likely harm generalization instead of improving it. In addition to preserving invertibility across iterations, we will see in Figure 4 that inverted representations of a car image for Resnet very closely resemble a car. By contrast, those produced by Alexnet are harder to decipher, and may even be mixing with different classes. We therefore believe the skip connection may allow Resnet to perform 'smart' data augmentation, which alters the input image without unintentionally mixing in information from other classes.

## 3 EXPERIMENTS

For our experiments, we attempt to assess whether activations produced by a function $r_\ell^{t+p}(x_i)$ can be interpreted as a perturbed image being passed through $r_\ell^t$. We do so by solving the optimization problem

$$||r_\ell^{t+p}(x_i) - r_\ell^t(x)||^2 \tag{2}$$

where $x$ is the variable we are optimizing over and $x_i$ is an image (e.g. of a car). We use the same TV prior as Mahendran & Vedaldi (2015). We will use the notation $j\_k$ on the x axes of the graph to indicate that the reference activations (corresponding to those produced by $r_\ell^{t+p}$ in the optimization above) were produced using the weights at epoch $j$ iteration $k$.

### 3.1 MNIST

We will investigate networks that have convolutional layers throughout this paper. However, to ensure that our results still hold for fully connected layers, we run experiments on a fully connected network with architecture 2000 nodes followed by 1000 nodes followed by 100 nodes followed by the output layer of size 10. Figure 1 shows our results using the weights at epoch 0 iteration 10 as our function. We find that the bottom row which corresponds to the bottom layer of the network preserves invertibility across more iterations than the row above it.

### 3.2 ALEXNET ON IMAGENET

We run the analogue of the MNIST experiment on Imagenet data with Alexnet. We use an initial learning rate of .01 and train for 90 epochs decaying the learning rate by a factor of 10 every 30 epochs. Our results are shown in Figure 2. Again we find that for lower layers, the image may still be inverted several iterations later. However, as we can see in the bottom row of Figure 2, it cannot be inverted after an epoch for the fifth convolutional layer.

### 3.3 RESNETS

We train a Resnet 18 model on Imagenet where we remove the batch normalization layers and add biases to the convolutional layers to maintain expressivity. We use a starting learning rate of .01, train for 90 epochs, and drop the learning rate by a factor of 10 every 30 epochs. Our results are shown in Figure 3. The bottom row of Figure 3 labeled conv 2 corresponds to the fifth convolutional

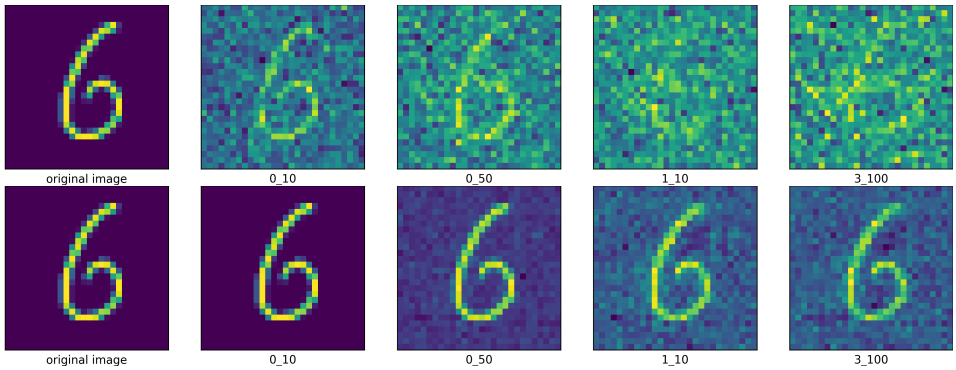


Figure 1: We find that in lower layers, even an epoch later, the image can still be inverted. However, even for the second layer, the later images are not invertible

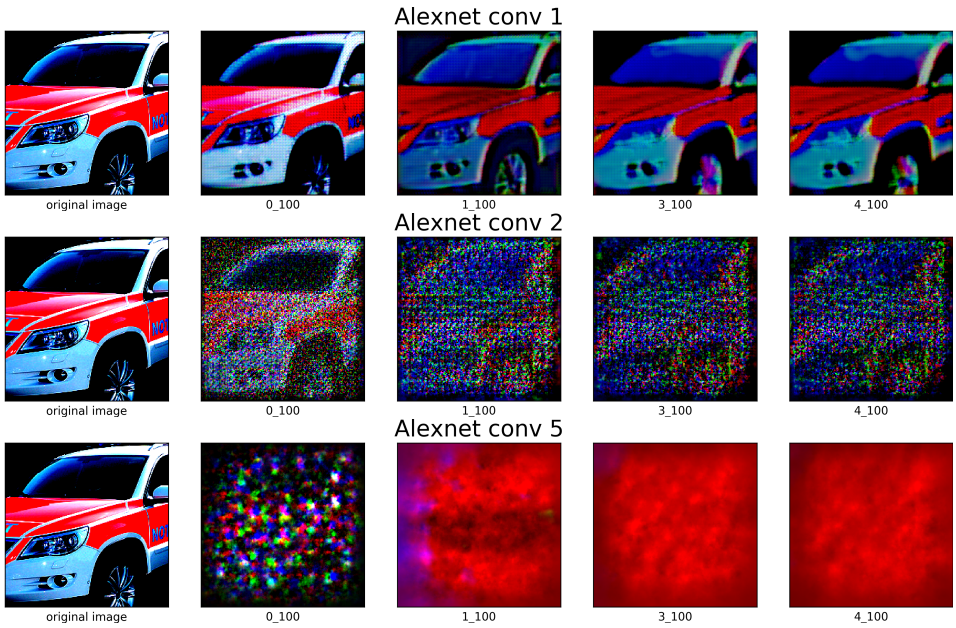


Figure 2: Layer inversions for Alexnet trained on Imagenet data. Original image shown in left hand column. Row 1 corresponds to Alexnet conv 1 layer, Row 2 to Alexnet conv2 layer, and Row 3 to the last Alexnet conv layer. Inversions are taken with the input function to the optimization being the epoch 0 iteration 100 layer function. We use the notation j_k on the x axes to denote that the activations used were from epoch $j$ and iteration $k$. We find that in lower layers, even an epoch later, the image can still be inverted. We find that for layer 5, the inversion fails after one epoch.

layer in the Resnet 18 architecture found at the end of Block 2. In comparison to the bottom row of Figure 2 which corresponds to the fifth convolutional layer in Alexnet, we find that Resnet preserves invertibility across more iterations than AlexNet.

In order to compare the inversions on the Resnet versus Alexnet, we invert the fifth convolutional layer for both Resnet and Alexnet in the top row of Figure 4. For the sake of visualization for the top row we increase the contrast on the images. We use the $4\_100$ layer function and the $4\_200$ activations for both networks. An important aspect of data augmentation is that it modifies an image of a particular class in a way that does not change the correct label of the resulting image. For example, augmenting a dog with a cat would likely harm generalization error. We find that Resnet preserves the underlying structure of the car, whereas Alexnet obfuscates it, and could even potentially be mixing in features from different classes. We find that in addition to preserving

the structure of the car, Resnet is also able to make interesting modifications to the image in the bottom row of Figure 4, like altering the shape of the wheel arch. We reason that perhaps the skip connections in Resnet improve generalization performance because they allow Resnet to augment the data in a way that preserves class membership.

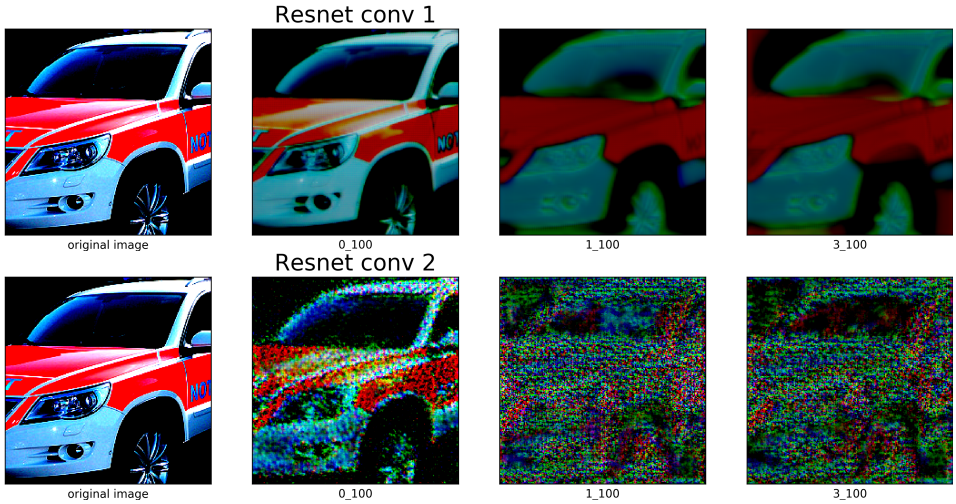


Figure 3: Layer inversions for Resnet18 trained on Imagenet data. Original image shown in left hand column. Row 1 corresponds to Resnet18 conv 1 layer, Row 2 to the last conv layer in Block 2 of Resnet18. This is the fifth convolutional layer. Inversions are taken with the input function to the optimization being the epoch 0 iteration 100 layer function. We use the notation j_k on the x axes to denote that the activations used were from epoch $j$ and iteration $k$. Comparing Row 2 of this figure with Row 3 of Figure 2 we find that Resnet preserves invertibility for longer.

## 4 Measuring the difference in activation norms

Previous work defined the distance between two probability measure, $P_1$ and $P_2$ as

$$\sup_E |P_1(E) - P_2(E)| \tag{3}$$

where $E$ are events. We might try to analogously check the difference in the representation produced by layer $\ell$ at two different times, for example writing

$$|r_\ell^t(x_i) - r_\ell^{t+p}(x_i)| \tag{4}$$

We check whether invertibility corresponds merely to having a smaller difference in the L2 norm of the current representation of a real input image $x_i$ and the later representation of a real input image $x_i$. In Figure 6 we plot the histograms of the norms of $||r_\ell^t(x_i) - r_\ell^t(x_j)||$ where $x_j$ is from a different class than $x_i$ as well as the scatter plot containing $||r_\ell^{t+p}(x_i) - r_\ell^t(x_i)||$ . We find that amount that the representation of $x_i$ moves is actually more than $||r_\ell^t(x_i) - r_\ell^t(x_j)||$ for the nearest $x_j$. Therefore, having a small norm difference in activations does not directly correlate to invertibility, because if $r_\ell^{t+p}(x_i)$ had been equal to $r_\ell^{t+p}(x_j)$, then the representation would have moved a relatively shorter distance, but its inversion would have yielded an image from a different class. This is possibly because there are two notions that the norm cannot capture. Firstly, it takes into account the magnitude of the difference without taking into account direction. For example, suppose a node has current activation of 5. Moving down to activation 3 and up to action 7 both correspond to the same change in norm, but may represent different things in input space. Secondly, it does not take into account the fact that some representations may never be picked by the gradient. Previous work Szegedy et al. (2014) has also found that information about images may be contained across many neurons, so the norm may not take into account subtleties in these activation directions.

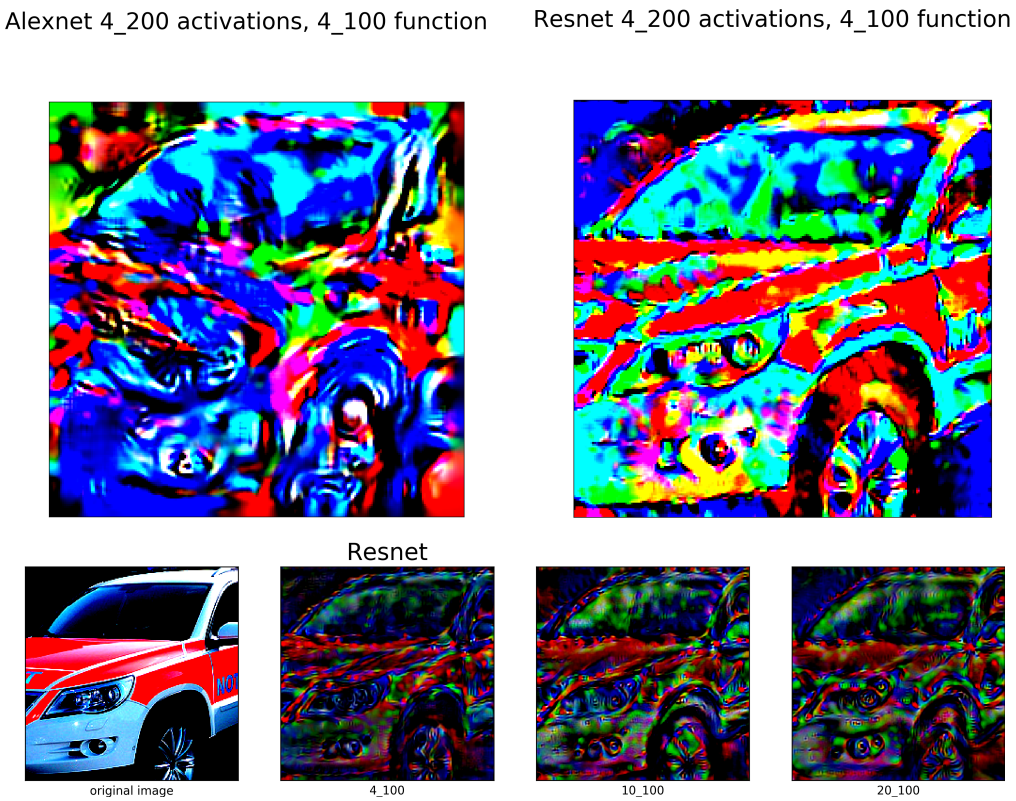


Figure 4: Top Row: On the left, we have the representation inversion for the fifth convolutional layer of AlexNet using the epoch 4 iteration 100 function and the epoch 4 iteration 200 activations. On the right, we have the analogous Resnet18 representation inversion. For the sake of visualization for this figure we increase the contrast on the images. We find that while Resnet preserves a clear image of the car, AlexNet does not. Bottom Row: Using the epoch 4 iteration 100 function. We find that Resnet 18 is still able to make interesting modifications to the car, like altering its wheel shape.

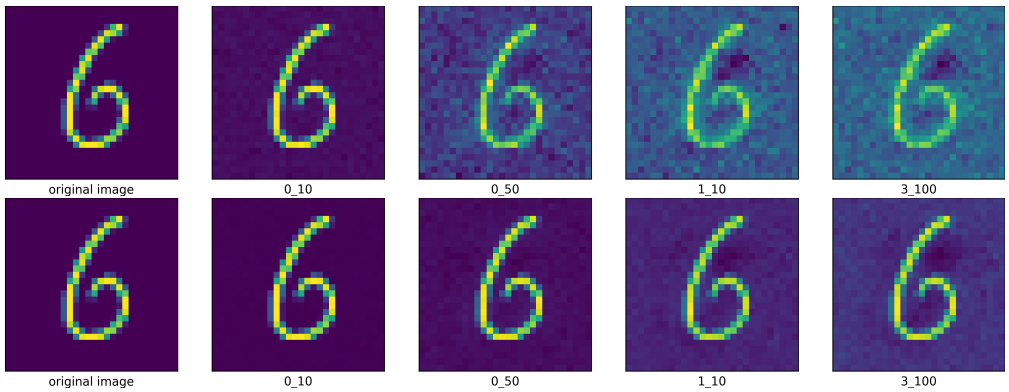


Figure 5: Image inversions for a larger MNIST network with 8000 nodes per fully connected layer. We notice that the ability to invert the network representations persists for longer than for the smaller MNIST network with 2000 nodes per fully connected layer.

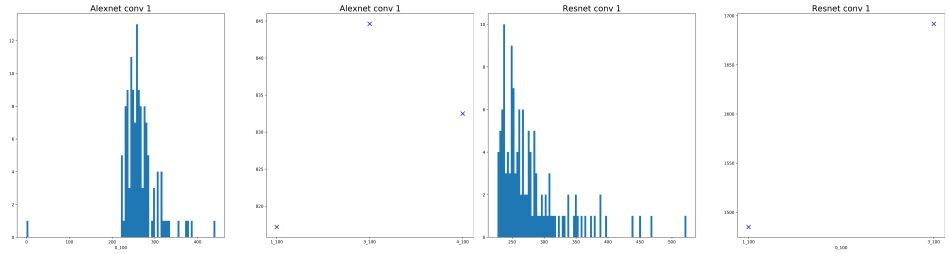


Figure 6: The histograms show the distance $||r_\ell^t(x_i) - r_\ell^t(x_j)||_2$ while the scatter plots show $||r_\ell^t(x_i) - r_\ell^{t+p}(x_i)||$ across iterations for which the representation of $x_i$ is invertible. We find that the latter distance is generally larger than the former, indicating that small norm of $||r_\ell^t(x_i) - r_\ell^{t+p}(x_i)||$ is not a good predictor of invertibility.

## 5 Discussion

We have found that the capacity and architecture of the neural network is important to the invertibility of the image representation across iterations, and the quality of this inversion.

**Capacity:** We run a simple experiment on MNIST where we increase the width of the smaller MNIST network so that it has 8000 nodes per layer. Our results can be found in Figure 5. We see that by comparison to Figure 1, that increasing the width of the network helps to preserve invertibility across more iterations. Traditional learning theory dictates that over-parametrization should increase generalization error. Neyshabur et al. (2015) showed that weight norms are not the main form of capacity control in neural networks, and a subsequent work Zhang et al. (2017) showed that generalization error continues to fall even when model capacity exceeds that required to memorize the training set. Since then, numerous works have provided explanations for this puzzling finding. Brutzkus & Globerson (2019) found that larger models may perform better because they are more efficient at exploring feature space and are subject to a clustering effect. We believe the finding that increasing width preserves invertibility has an interesting connection to the work of Park et al. (2019) who find that increasing capacity may help generalization because wider networks have an increased optimal noise scale. In future work it would be interesting to examine if increased noise scale corresponds to more varied data augmentation.

**Other architecture changes:** We found e.g. in Figure 4 that adding skip connections may improve the quality of the implicit data augmentation for Resnets by perturbing images in a way that does not accidentally alter their class. We believe in future work that it would be interesting to examine the intersection of this idea with other techniques used during training such as batch normalization. Batch normalization Ioffe & Szegedy (2015) was originally proposed to combat internal covariate shift, and has proved extremely effective in allowing the training of deep networks like Resnet. Subsequent work found that batch normalization may not affect covariate shift, but aids in training because it smooths the training loss as a function of the weight parameter $w$. Examining whether batch normalization improves either invertibility across time or the quality of the implicit data augmentation is an interesting problem for future study.

## 6 Conclusion

Traditional methods for data augmentation seek to modify training to produce an augmented set of images for the neural network. We have found that neural networks experience an implicit form of data augmentation. We have shown that activations produced by layer $\ell$ at a later timestep are still interpretable in terms of the representation function of layer $\ell$ at the current time. We find that this effect is more consistent across time for Resnets than for CNN architectures, and that adding skip connections may encourage a form of data augmentation that does not unintentionally alter the class label.

## REFERENCES

David Balduzzi, Marcus Frean, Lennox Leary, JP Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. The shattered gradients problem: If resnets are the answer, then what is the question? In *International Conference on Machine Learning*, pp. 342–350, 2017.

Peter L Bartlett, Shai Ben-David, and Sanjeev R Kulkarni. Learning changing concepts by exploiting the structure of change. *Machine Learning*, 41(2):153–174, 2000.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Christopher Bowles, Roger Gunn, Alexander Hammers, and Daniel Rueckert. Gansfer learning: Combining labelled and unlabelled data for gan based data augmentation. *arXiv*, pp. arXiv–1811, 2018.

Alon Brutzkus and Amir Globerson. Why do larger models generalize better? a theoretical perspective via the xor problem. In *International Conference on Machine Learning*, pp. 822–830. PMLR, 2019.

Uri Cohen, SueYeon Chung, Daniel D Lee, and Haim Sompolinsky. Separability and geometry of object manifolds in deep neural networks. *Nature communications*, 11(1):1–13, 2020.

Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pp. 1486–1494, 2015.

Alexey Dosovitskiy and Thomas Brox. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4829–4837, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016.

Furong Huang, Jordan Ash, John Langford, and Robert Schapire. Learning deep resnet blocks sequentially using boosting theory. In *International Conference on Machine Learning*, pp. 2058–2067, 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. *ICML*, 2019.

Karel Lenc and Andrea Vedaldi. 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.

Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John E Hopcroft. Convergent learning: Do different neural networks learn the same representations? In *ICLR*, 2016.

Renjie Liao, Alex Schwing, Richard Zemel, and Raquel Urtasun. Learning deep parsimonious representations. In *Advances in Neural Information Processing Systems*, pp. 5076–5084, 2016.

Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5188–5196, 2015.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pp. 1376–1401, 2015.

Emin Orhan and Xaq Pitkow. Skip connections eliminate singularities. In *International Conference on Learning Representations*, 2018.

Daniel Park, Jascha Sohl-Dickstein, Quoc Le, and Samuel Smith. The effect of network width on stochastic gradient descent and generalization: an empirical study. In *International Conference on Machine Learning*, pp. 5042–5051, 2019.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Advances in Neural Information Processing Systems*, pp. 6076–6085, 2017.

Alexander J Ratner, Henry Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. Learning to compose domain-specific transformations for data augmentation. In *Advances in neural information processing systems*, pp. 3236–3246, 2017.

Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert MÃžller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(May):985–1005, 2007.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014. URL `http://arxiv.org/abs/1312.6199`.

Paul Upchurch, Jacob Gardner, Geoff Pleiss, Robert Pless, Noah Snavely, Kavita Bala, and Kilian Weinberger. Deep feature interpolation for image content changes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7064–7073, 2017.

Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Advances in neural information processing systems*, pp. 550–558, 2016.

Alireza Zaeemzadeh, Nazanin Rahnavard, and Mubarak Shah. Norm-preservation: Why residual networks can become extremely deep? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *ICLR*, 2017.

# A Appendix

You may include other additional sections here.