3-25-2020

Learning in the Machine: To Share or Not to Share?

Jordan Ott

Erik Linstead

Nicholas LaHaye

Pierre Baldi

Learning in the Machine: To Share or Not to Share?

Neural Networks 126 (2020) 235–249

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Learning in the machine: To share or not to share?

Jordan Ott [a,b], Erik Linstead [a,*], Nicholas LaHaye [a], Pierre Baldi [b]

[a] Fowler School of Engineering, Chapman University, United States of America
[b] Department of Computer Science, Bren School of Information and Computer Sciences, University of California, Irvine, United States of America

ARTICLE INFO

Article history:
Received 24 April 2019
Received in revised form 15 March 2020
Accepted 16 March 2020
Available online 25 March 2020

Keywords:
Deep learning
Convolutional neural networks
Weight-sharing
Biologically plausible architectures

ABSTRACT

Weight-sharing is one of the pillars behind Convolutional Neural Networks and their successes. However, in physical neural systems such as the brain, weight-sharing is implausible. This discrepancy raises the fundamental question of whether weight-sharing is necessary. If so, to which degree of precision? If not, what are the alternatives? The goal of this study is to investigate these questions, primarily through simulations where the weight-sharing assumption is relaxed. Taking inspiration from neural circuitry, we explore the use of Free Convolutional Networks and neurons with variable connection patterns. Using Free Convolutional Networks, we show that while weight-sharing is a pragmatic optimization approach, it is not a necessity in computer vision applications. Furthermore, Free Convolutional Networks match the performance observed in standard architectures when trained using properly translated data (akin to video). Under the assumption of translationally augmented data, Free Convolutional Networks learn translationally invariant representations that yield an approximate form of weight-sharing.

1. Introduction

Digital simulations of neural networks are successful in many applications but rely on a fantasy where neurons and synaptic weights are objects stored in digital computer memories. This fantasy often obfuscates some fundamental principles of computing in native neural systems. To remedy this obfuscation, learning in the machine refers to a general approach for studying neural computations. In this approach, the physical constraints of physical neural systems, such as brains or neuromorphic chips, are taken into consideration. When applied to single neurons, learning in the machine can lead, for instance, to the discovery of dropout (Baldi & Sadowski, 2014; Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). When applied to synapses, learning in the machine can lead, for instance, to the discovery of local learning (Baldi & Sadowski, 2016) and random backpropagation (Baldi, Lu, & Sadowski, 2017, 2018; Lillicrap, Cownden, Tweed, & Akerman, 2016). Moreover, when applied to layers of neurons, as we do in this paper, learning in the machine leads one to question the fundamental assumption of weight-sharing behind convolutional neural networks (CNNs).

The technique of weight-sharing, whereby different synaptic connections share the same strength, is a widely used and successful technique in neural networks and deep learning. This paradigm is particularly true in computer vision where weight-sharing is one of the pillars behind convolutional neural networks and their successes. In any physical neural system, for instance, carbon- or silicon-based, exact sharing of connections strengths over spatial distances is difficult to realize, especially on a massive 3D scale. In physical systems, not only is it difficult to create identical weights at a given time point, but it is also challenging to maintain the identity over time. During phases of development and learning the weights may be changing rapidly. During more mature stages weights must retain their integrity against the microscopic, entropic forces surrounding any physical synapse. Furthermore, given the exquisitely complex geometry of neuronal dendritic trees and axon arborizations, it is implausible to form large arrays of neurons with identically translated connection patterns. In short, not only is it challenging to share weights exactly, but it is also difficult to exactly share the same connection patterns.

While weight-sharing has proven to be very useful in computer vision and other applications, it is extremely implausible in biological and other physical systems. This discrepancy raises the fundamental question of whether weight-sharing is a strict prerequisite for convolution-based deep learning, or if similar levels of learning are possible without it. In particular, we consider the following research questions:

1. Is weight-sharing necessary to prevent overfitting?
2. Is weight-sharing necessary to ensure translational invariant recognition?

* Corresponding author.
E-mail addresses: jott1@uci.edu (J. Ott), linstead@chapman.edu (E. Linstead), lahay100@mail.chapman.edu (N. LaHaye), pfbaldi@ics.uci.edu (P. Baldi).

https://doi.org/10.1016/j.neunet.2020.03.016

3. Can acceptable classification performance be achieved without weight-sharing?
4. Does approximate or exact weight-sharing emerge in a natural way[1]?

In formulating the research questions above, we considered the most common reasons practitioners give for employing weight sharing in convolutional network architectures. The purpose is to challenge these common points based on the intuitive principle that weight sharing is implausible in biological systems. In total, answers to these questions provide new insight into whether weight sharing is a strict prerequisite for the effective training of convolutional architectures, and what happens if the requirement for weight sharing is relaxed. The goal of this study is to investigate these research questions, primarily through simulations where the weight-sharing assumption is relaxed. Some of these questions have been previously considered in the literature in other contexts. We are not the first to utilize locally connected architectures or assess their performance on computer vision tasks (Bartunov et al., 2018), for example. However, in aggregate, the answer to these questions provide novel insight into whether weight-sharing is vital for convolutional architectures and whether it can emerge in other ways when connections are not shared.

2. Origins and functions of weight-sharing

Before addressing the question of its necessity, it is useful to review the origins and functions of weight-sharing. Hubel and Wiesel's neurophysiological work (Hubel & Wiesel, 1962) on the cat visual cortex was the inception of weight-sharing. These experiments suggested the existence of entire arrays of neurons dedicated to implementing simple operations, such as edge detection and other Gabor filters, at all possible image locations. Fukushima systematically used the ideas proposed by Hubel and Wiesel to create the neocognitron (Fukushima, 1980) in computer vision. Primarily a convolutional neural network architecture with Hebbian learning. However, Hebbian Learning alone applied to a feedforward CNN cannot solve vision tasks (Baldi & Sadowski, 2016). Solving vision tasks requires feedback channels and learning algorithms for transmitting target information to the deep synapses. A job that is precisely achieved by backpropagation, or stochastic gradient descent. Successful CNNs for vision problems, trained via backpropagation, were developed in the late 80s and 90s (Baldi & Chauvin, 1993; Cun et al., 1990; Schmidhuber, 2015).

Substantial improvements in the size of the training sets and available computing power have led to a new wave of successful implementations in recent years, (He, Zhang, Ren, & Sun, 2015; Krizhevsky, Sutskever, & Hinton, 2012; Srivastava, Greff, & Schmidhuber, 2015; Szegedy et al., 2015), as well as applications to a variety of specific domains, ranging from biomedical images (Cireş, Giusti, Gambardella, & Schmidhuber, 2012; Esteva et al., 2017; Gulshan et al., 2016; Urban et al., 2018; Wang, Ding, et al., 2017; Wang, Fang, et al., 2017) to particle physics (Aurisano et al., 2016; Baldi, Bauer, Eng, Sadowski, & Whiteson, 2016; Sadowski, Radics, Ananya, Yamazaki, & Baldi, 2017) and video analysis (Ott, Atchison, Harnack, Bergh, & Linstead, 2018; Ott, Atchison, Harnack, Best, et al., 2018; Tompson, Stein, Lecun, & Perlin, 2014; Tran, Bourdev, Fergus, Torresani, & Paluri, 2015). Older (Zipser & Andersen, 1988) as well as more recent work (Cadieu et al., 2014; Yamins et al., 2014) has also shown that not only do convolutional neural networks rival the object category recognition accuracy of the primate cortex but also seem to provide the best match to biological neural responses, at least at some coarse level of analysis.

It is worth pointing out that weight-sharing is sometimes used in other settings, for instance when Siamese Networks are used to process and compare objects (Baldi & Chauvin, 1993; Bromley et al., 1993; Kayala & Baldi, 2012), which also includes Siamese CNNs for images. Siamese Networks consist of two or more subnetworks that are used to train a contrastive loss function in order to learn the differences between pairs of inputs. In the case of Siamese CNN's, the typical weight sharing paradigm is used between each subnetwork architecture. In addition, the weights of each subnetwork are identical with each other, hence achieving another "level" of weight sharing. Finally, a different kind of weight-sharing that will not concern us here, when a recurrent network is unfolded in time. In this case, weight-sharing occurs over time and not over space (Cho et al., 2014; Hochreiter & Schmidhuber, 1997).

Biologically motivated architectures have become an important focal point of deep learning as of late (Baldi & Sadowski, 2018; Baldi, Sadowski, & Lu, 2017, 2018; Bartunov et al., 2018; Lillicrap et al., 2016; Ott, 2019; Samadi, Lillicrap, & Tweed, 2017). Most recently Bartunov et al. (2018) investigated how biologically motivated deep learning algorithms scale to more massive datasets. Locally connected networks were included in their simulations, where each weight kernel interacts with *local* features, not the entire input. However, the focus of the paper is on how learning algorithms such as feedback alignment (Lillicrap et al., 2016) and target propagation (Lee, Zhang, Fischer, & Bengio, 2015), scale to more significant image recognition problems. Not addressed in the paper is the problem of learning translationally invariant representations, overfitting, variable connection patterns, and the emergence of approximate weight-sharing. Nor was there in-depth analysis comparing networks with weight-sharing to those without.

Two primary but different purposes are typically associated with weight-sharing. The first is to reduce the number of free parameters that need to be stored or updated during learning. This requirement can be important in applications where storage space is limited (i.e., cell phones), or where training data is limited, and overfitting is a danger. It is important to remember, however, that in convolutional architectures the local connectivity of neurons contributes at least as much to parameter reduction as weight-sharing. In other words, weight-sharing is not the only way to shrink the parameter space. The second function of weight-sharing is to apply the same operation at different locations of the input data, to process the data uniformly and provide a basis for invariance, typically translation invariant recognition in CNN architectures.

3. Free convolutional networks

The research questions proposed in Section 1 deal with the necessity of weight sharing and the result of its absence. As a result, it is natural to consider simulations where weights are not shared, but instead are maintained for specific regions of the input space. Relaxing the weight-sharing assumption in CNNs yields a Free Convolutional Network (FCN). In FCNs, the weights of a filter at a specific location are not tied to the weights of the same filter at a different location (see Fig. 1). This naturally means that FCNs have far more parameters than the corresponding CNN and are slower to train on a digital machine. However, this is not a concern here, as our primary goal is not to improve the efficiency of CNNs deployed on digital machines, but rather to understand the consequences of relaxing the weight-sharing assumption inside a native neural machine.

[1] Without explicit constraints in the architecture or imposed constraints in the form of losses.

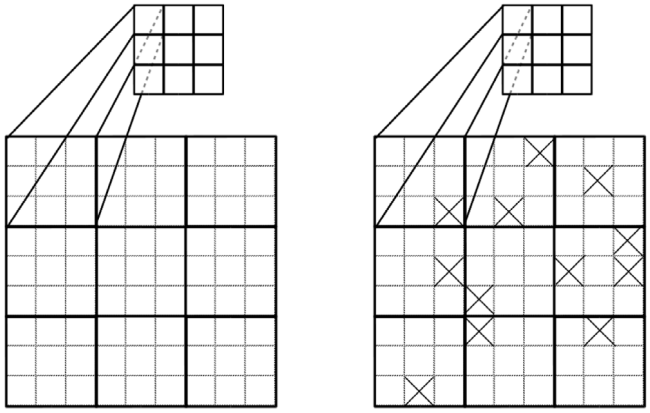

Fig. 1. Free convolutional layers maintain a separate kernel at each location, unlike typical convolutional layers, that apply the same filter across all possible locations. The above figures are examples of FCN layers on a 9×9 input space. Each 3×3 subregion of the input is covered with a distinct kernel (weight matrix), as shown in the diagram on the left. The top square represents the output obtained from applying the filter to the corresponding input region. The diagram on the right depicts free convolutional layers with variable connection patterns, where the x's represent absent connections. In this example, 12 out of the 91 connections are missing, creating a variable connection probability of roughly 0.15.

Furthermore, it is highly implausible that a given neuron will share the same dendritic tree with a neighboring neuron (Jan & Jan, 2010), thus in addition to neighboring neurons of the same layer not having the same weights, we would like to consider also the possibility of them not having the same receptive field pattern. Thus, in addition to plain FCNs, FCNs with variable connection patterns are implemented in the simulations below. The implementation of variable connection patterns has many options. Here, for simplicity, a random percentage of connections is severed (Fig. 1) [Note: this is very different from dropout where different sets of weights get randomly set to 0 at each presentation of a training example]. Similar to in methodology to DropConnect (Wan, Zeiler, Zhang, Le Cun, & Fergus, 2013), however, the same weights remain 0 for all of training and testing. The x's in the right image of Fig. 1 correspond to missing synaptic connections between neurons that are set to zero and never trained.

By running simulations comparing CNNs and FCNs (with and without variable connection patterns), we seek to answer the research questions laid out in the introduction. Appendix C contains complete details regarding simulations with variable connection patterns. To our knowledge, this is the first systematic study, through large-scale computational simulations, that explores the necessity of weight-sharing in deep convolutional architectures as it pertains to overfitting and performance in the broader context of biological plausibility. Further novelty is found in our assessment of the emergence of approximate weight-sharing in free convolutional architectures with and without variable connection patterns.

4. Data and methods

In the simulations, we focus exclusively on computer vision tasks. We evaluate various free and shared weight networks on two well-known benchmark datasets: the handwritten digit dataset, MNIST (LeCun & Cortes, 2010), as well as the CIFAR-10 object dataset (Krizhevsky, Nair, & Hinton, 2009).

In the case of free weights, we consider using data augmentation by translating images horizontally and vertically to potentially compensate for the lack of translation, inherent in the architecture. Due to the local receptivity of free weight networks, individual filters learn features solely within their receptive field. Translationally invariant data will allow filters to learn more or less the same features across the input space. Conversely, in CNNs, weights are shared across space. By applying the same operation across the input space, translational invariance is naturally embedded in the model. This property of CNNs gives them an inherent advantage in vision-based tasks. The purpose of these experiments is not to demonstrate the superiority of one network but to understand the consequences of weight-sharing and properties that arise around it.

Eleven settings of translation were tested in experiments (0% to 99% by increments of 10%). Translational augmentation involves shifting images horizontally and vertically by varying degrees of the width and height respectively. Points outside the boundaries of the input get filled according to the nearest pixels. Fig. 2 shows examples of translational augmentation results on MNIST. During training, each image presented to the network is translated left–right as well as up-down by random amounts. 0% to the augmentation setting for that trial. e.g., at 90% augmentation, an image may be translated any amount 0%–90% up-down as well as 0%–90% left–right.

All simulations, each augmentation setting, were completed using five-fold cross-validation. Simulations were implemented in Keras (Chollet et al., 2015) with a Tensorflow (Abadi et al., 2015) backend using NVIDIA GeForce GTX Titan X GPUs with 12 GB memory. For purposes of reproducibility, the code has been made publicly available.[2]

4.1. Networks

Conducting a grid search yielded appropriate hyperparameter configuration for all networks. See Appendix A.2 for complete details. The search produced network architectures composed of three convolutional/free convolutional layers, followed by a single hidden layer, and one output layer for classification. A full description of all networks, including filter size, number of filters, stride, and learning rate can be found in Table A.2. MNIST and CIFAR require different architectures as the input sizes differ. For every setting of data augmentation (0–99%, increments of 10%) a CNN and FCN were trained via five-fold cross-validation.

For simplicity, the architecture of CNNs and FCNs are equivalent, except that free convolutional layers replace convolutional layers. The activation functions, softmax layer, as well as the number of filters per layer, remain the same across all networks. Table A.2 lists hyperparameter settings of the networks used in this paper. Additionally, it is essential to note that there is no architectural difference between the networks trained with data augmentation and those trained without.

4.2. Variable connection patterns

Also implemented in this study are neurons with variable connection patterns in FCNs. At the start of training, a chosen percentage of weights is randomly set to 0, representing the absence of a dendritic connection. These missing weights do not contribute to the output of the layer, and their values are never updated during backpropagation. The resulting connection patterns are maintained throughout training and testing. There are multiple options for implementing neurons with variable connection patterns. For computational simplicity, the implementation used in this paper is to turn off connections within each square filter given some probability (depicted in Fig. 1). In simulations, we vary the drop probability from 0 to 99% by increments of 10%. The results from these simulations are reported in Figs. C.1 and C.2 of Appendix.

[2] https://github.com/Learning-In-The-Machine/Weight-Sharing

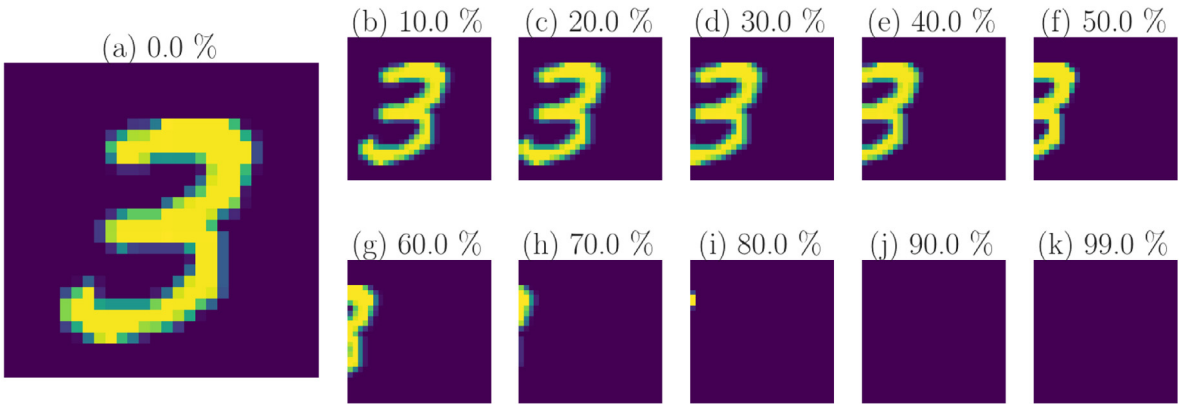

Fig. 2. Examples of translational augmentation on MNIST. In training images are translated left–right as well as up-down. The above only display the result of continually translating an image leftwards, for a visual example. (a) 0% translation augmentation, equivalent to a un-altered MNIST image. (b–k) Gradually increasing the percentage of augmentation by 10% each time.

Table 1
MNIST results. Median accuracies of training, un-augmented validation, and augmented validation set. The left most column denotes the amount of translational augmentation used during training. When performing translational augmentation on the validation set, 25% augmentation was used throughout the experiments. Bold values indicate the highest performing model in that accuracy metric for CNN and FCN, respectively.

Aug %	Training accuracy		Validation accuracy		Translation accuracy	
	CNN	FCN	CNN	FCN	CNN	FCN
0.00	**0.999964**	**0.999175**	0.990536	0.988071	0.366536	0.372287
0.10	0.997793	0.995665	**0.993071**	**0.991857**	0.837963	0.813368
0.20	0.996719	0.991730	0.993000	0.990357	0.986220	0.977033
0.30	0.992555	0.983998	0.992143	0.987643	**0.990741**	**0.985062**
0.40	0.976966	0.965521	0.991571	0.984500	0.989439	0.982422
0.50	0.935973	0.923703	0.990071	0.981571	0.987594	0.979456
0.60	0.863680	0.847367	0.988286	0.978500	0.986111	0.976273
0.70	0.752630	0.731917	0.987000	0.974286	0.984484	0.971933
0.80	0.626956	0.606541	0.985500	0.970786	0.982820	0.968171
0.90	0.520896	0.503312	0.984714	0.967821	0.982096	0.965133
0.99	0.448491	0.431218	0.984000	0.965071	0.981120	0.961372

Table 2
CIFAR results. Median accuracies of training, un-augmented validation, and augmented validation set. The left most column denotes the amount of translational augmentation used during training. When performing translational augmentation on the validation set, 25% augmentation was used throughout the experiments. Bold values indicate the highest performing model in that accuracy metric for CNN and FCN, respectively.

Aug %	Training accuracy		Validation accuracy		Translation accuracy	
	CNN	FCN	CNN	FCN	CNN	FCN
0.00	**1.000000**	**1.000000**	0.671083	0.647333	0.444556	0.433468
0.10	0.982156	0.973354	0.749333	0.724333	0.598538	0.561324
0.20	0.876542	0.858094	0.773250	**0.731500**	0.705225	0.653478
0.30	0.834646	0.781885	**0.774083**	0.720167	0.739163	**0.683972**
0.40	0.775188	0.725198	0.768750	0.704958	**0.742608**	0.680444
0.50	0.712146	0.657438	0.751667	0.682250	0.729419	0.661458
0.60	0.650719	0.598344	0.734042	0.659958	0.714130	0.642557
0.70	0.592385	0.545354	0.717250	0.641542	0.699555	0.624286
0.80	0.535958	0.496677	0.699667	0.626250	0.681452	0.607695
0.90	0.483635	0.449490	0.684833	0.611667	0.667801	0.593834
0.99	0.442260	0.413802	0.668667	0.599458	0.653436	0.583375

5. Results

We report accuracy metrics on translationally augmented training (0%–99%, increments of 10%), un-augmented validation, and translationally augmented validation set (with 25% translation) throughout training. Results for MNIST and CIFAR are shown in Figs. 3 and 4, respectively. The legend denotes the amount of translation used during training (0%–99%, increments of 10%).

Tables 1 and 2 show the median accuracy of the five-fold cross validation experiments for the MNIST and CIFAR-10 datasets respectively. The tables display results for each corresponding setting of translation augmentation (0%–99%, increments of 10%). Bold values indicate the highest performing model in that accuracy metric for CNN and FCN, respectively. The Appendix contains additional experiments regarding the use of other augmentation methods, additional training metrics, as well as simulations regarding neurons with variable connection patterns.

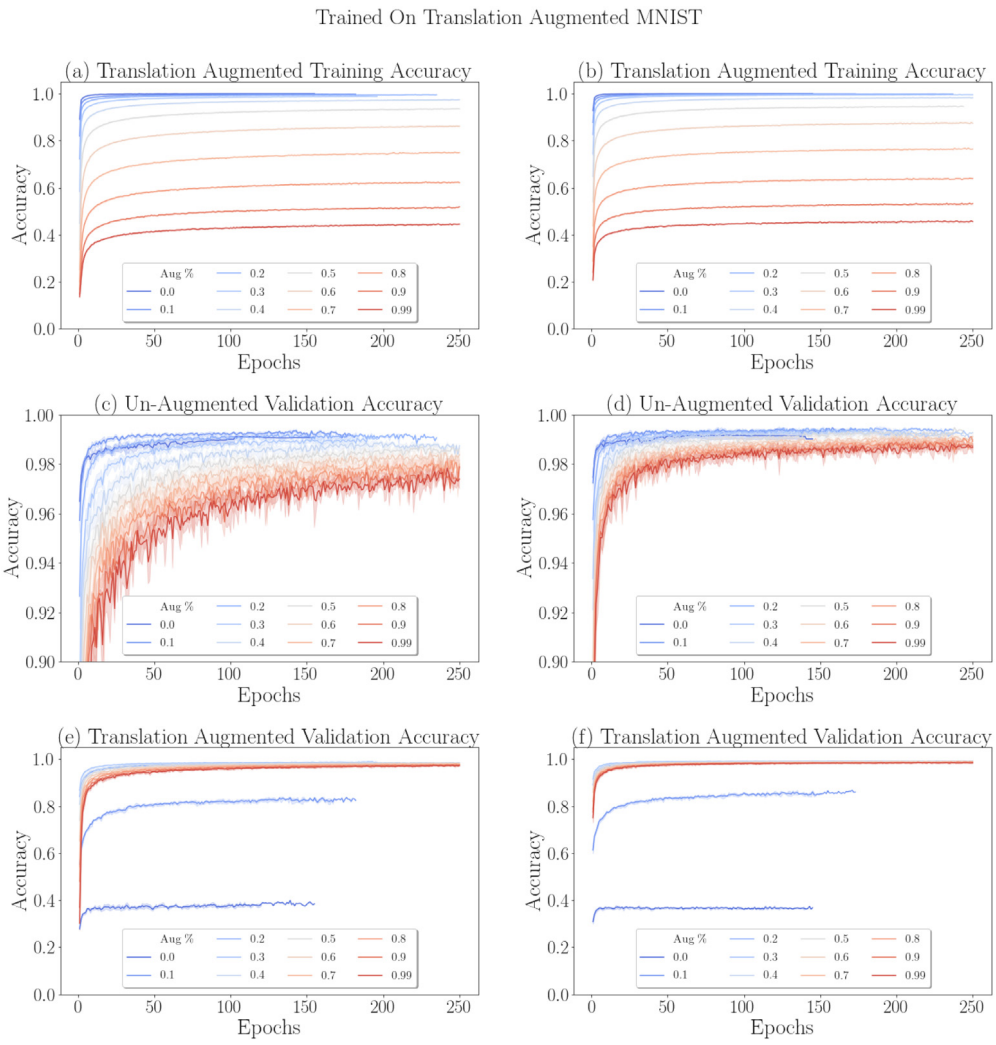

Fig. 3. MNIST results. Shown above are FCN (left column) and CNN (right column) results trained with varying degrees of translational augmentation, indicated by the legend. Top row: training accuracy over time. Middle row: validation accuracy over time. Bottom row: accuracy on the translationally augmented validation set.

5.1. Is weight-sharing necessary to prevent overfitting?

Both FCNs and CNNs are shown to overfit the training set, given a sufficient number of epochs. This result is evident from a divergence in training accuracy vs. validation accuracy over time (MNIST and CIFAR, Figs. 3 and 4 respectively). In situations, without overfitting, one would expect training and validation scores to be close to one another. However, as the training accuracy continues to increase towards 100%, in many cases the validation accuracy declines, the telltale indication of overfitting. In the remaining cases the validation accuracy plateaus while training accuracy continues to grow. The large gap between training and validation performance in these cases suggests that the model's ability to generalize to new data is suffering, and that overfitting is likely taking place. In experiments performed on MNIST and CIFAR overfitting was observed for CNNs and FCNs when translation was not used, Figs. 3 and 4 respectively. This effect manifests in CIFAR (Table 2) where there is more than a 30% gap between training accuracy and validation accuracy for both CNN and FCN, with no translational augmentation (i.e., 0% translation).

In the case of MNIST, we see from Table 1 that FCNs, trained on augmented data (10% translation), achieve a median accuracy of 99.1857% on the validation set. Only slightly less than 99.3071% median accuracy is achieved by standard CNNs. On the more complicated CIFAR dataset, with 10% translation, FCNs can come within 2% of CNN accuracy on the validation set (Table 2). Thus, we can infer that data augmentation alone is sufficient to prevent overfitting, and weight-sharing need not be leveraged to achieve this.

Using augmentation during training (Figs. 3 and 4) not only reduces overfitting but also leads to an increase in performance on validation sets. For FCNs on CIFAR specifically, training with translational amounts of 10, 20, and 30% increases FCN validation performance while dramatically reducing overfitting, compared to 0% translation. In human learning, translational augmentation comes as a byproduct of interacting with a changing world, which inherently provides the brain with samples of the same object translated at different positions — for example, a car driving down the street. Because FCNs cannot rely on weight-sharing to simulate the effect of translation, they must, therefore, rely on manual augmentation.

Large, translationally invariant datasets, like those produced through data augmentation, are essential for free weight networks to achieve excellent performance and avoid overfitting. If this constraint can be met, overfitting can be mitigated without weight-sharing.

5.2. Is weight-sharing necessary to ensure translational invariant recognition?

Learning translationally invariant representations is tested by using translational augmentation on the validation set. Fig. 3e and

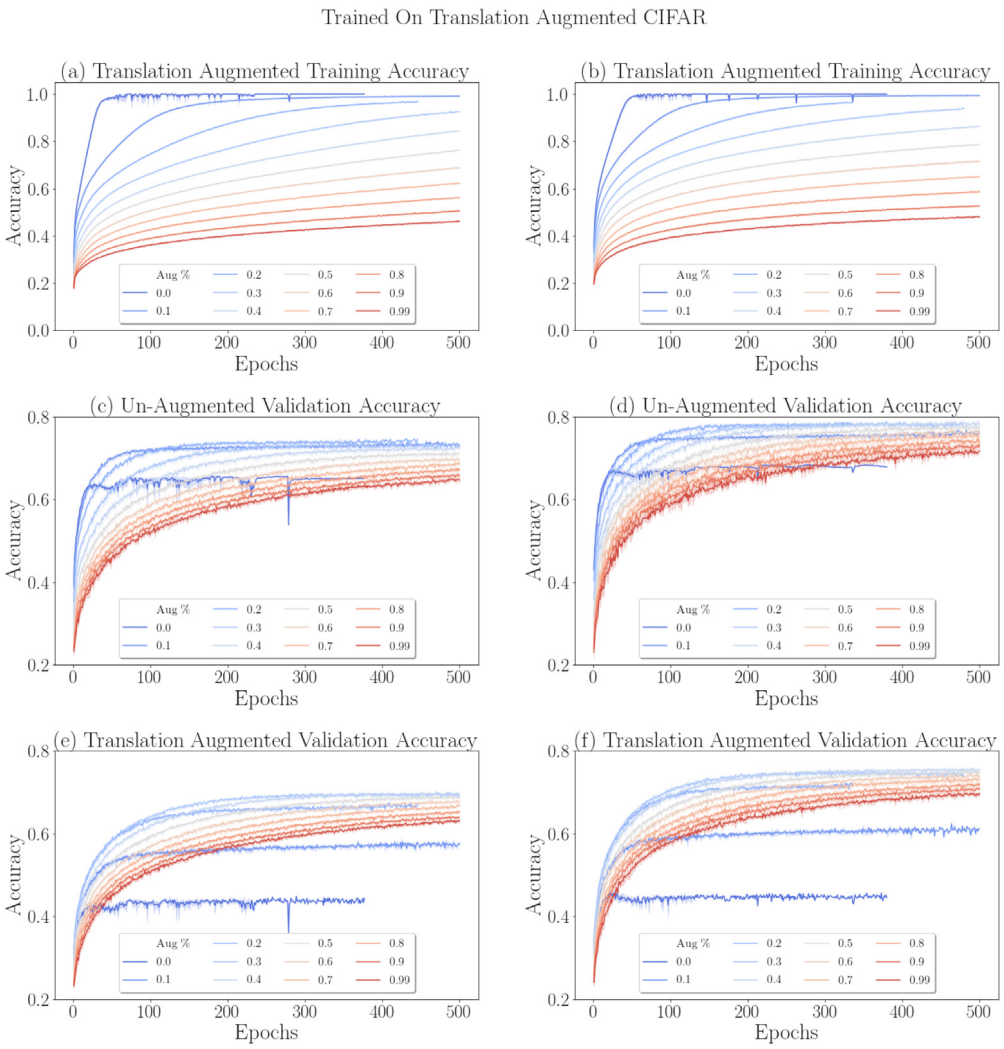

Fig. 4. CIFAR results. Shown above are FCN (left column) and CNN (right column) results trained with varying degrees of translational augmentation, indicated by the legend. Top row: training accuracy over time. Middle row: validation accuracy over time. Bottom row: accuracy on the translationally augmented validation set.

f for MNIST and Fig. 4e and f for CIFAR show results of translation on the augmented validation set.

Significant disparity between validation accuracy and translation augmented validation accuracy would signal that the network is not capable of translationally invariant recognition. For example, the FCN trained on MNIST data without augmentation saw more than a sixty percent gap between validation set accuracy and translation augmented validation set accuracy (Table 1). Similarly, on CIFAR, the FCN trained without translation augmentation performed 20% worse on the translation augmented validation set than the standard one (Table 2). Meaning these networks trained with 0% augmentation are incapable of learning translation invariant representations.

On both MNIST and CIFAR, FCNs achieve comparable results on the validation set and translation augmented validation set. This result is evident on MNIST where an FCN trained with 30% translation achieves a 0.2% difference between validation and translation augmented validation performance. Likewise, on CIFAR, the FCN trained with 40% translation comes within 2% of its validation performance on the translation augmented validation set.

The results show that as translational augmentation is used during training, the gap between the validation accuracy and translation augmented validation accuracy decreases. This confirms that FCNs are capable of learning translationally invariant representations provided sufficient, translationally augmented, data.

5.3. Can acceptable performance be achieved without weight-sharing?

FCNs can achieve high-performance scores on two benchmark computer vision datasets, which agrees with what has been noted previously in the literature (Bartunov et al., 2018). In the absence of data augmentation (0% translation), CNNs outperform FCNs on MNIST and CIFAR. Referencing Table 1, CNNs achieve 99.05% validation accuracy whereas FCNs are slightly worse with 98.81% validation accuracy for MNIST. On CIFAR, Table 2, CNNs outperform FCNs with 67.11% and 64.73%, respectively. When FCNs are exposed to translational data, they can achieve 99.19%, 10% augmentation, and 73.15%, 20% augmentation, on MNIST and CIFAR validation sets, respectively.

The validation set accuracy observed in our simulations shows that as translational augmentation is increased, FCNs can match the performance of CNNs. Thus, there is not a fundamental necessity of weight-sharing to attain satisfactory performance.

5.4. Does approximate or exact weight-sharing emerge in a natural way?

To test for the emergence of approximate weight-sharing, we calculate the average euclidean distance between each filter of the FCN layer, at a specified radius. Experiments in this paper

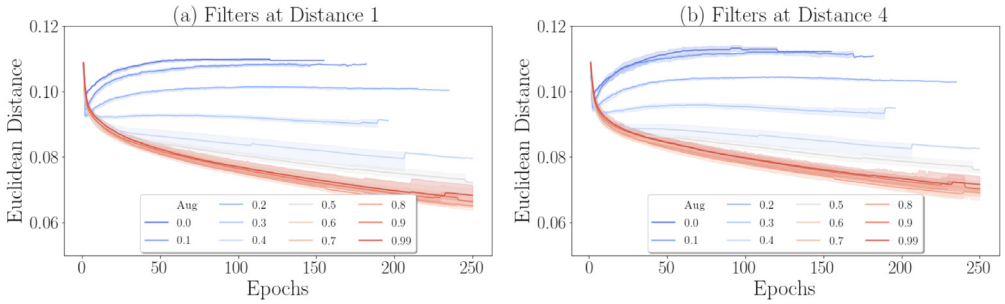

Fig. 5. Shown above is the average Euclidean distance between weight kernels in FCN layer overtime at the specified radius away. E.g., a radius of four indicates comparing a given weight kernel to all kernels four units away. As translational augmentation is increased weights become more similar to one another. As expected, weights closer in proximity (radius 4; left figure) have a smaller average distance than weights farther away (radius 7; right figure).

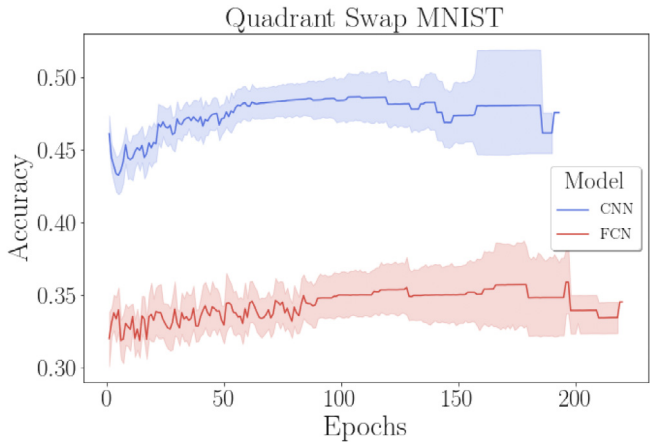

Fig. 6. Results for CNN and FCN models trained on MNIST and tested on images where quadrant I is replaced with quadrant III and vice versa.

use a radius of one and four units to test the similarity of filters at different distances. Fig. B.5 in the Appendix, gives a visual example. This depiction means that each filter in the FCN layer is compared to the filters one and four units, respectively, away from it. Fig. 5 reports the euclidean distance between filters at radius one and radius four.

If weight-sharing did emerge naturally, one would expect to see a lower average euclidean distance between filters within an FCN layer throughout training. One would only expect to see this convergence in weight values if a similar stimulus is present across the input space (i.e., translationally augmented data, akin to video). Simulations conducted on the MNIST dataset confirms this. As the amount of translation increases (indicated by the legend in Fig. 5), the parameters of the FCN layer converge to similar values; the euclidean distance between filters decreases. Furthermore, a higher degree of translation yields more similar the weight values. Conversely, when translation is not used during training, the weights diverge and become less similar from each other over time (i.e., the distance between them increases).

One would also expect filters near one another to be more similar than those farther away (i.e., filters at radius 1 should be more similar than at radius 4). Again, this is confirmed in simulations where Fig. 5a shows lower average Euclidean distance values, per augmentation setting, compared to Fig. 5b.

To ascertain the cause of approximate weight-sharing, FCNs were trained with noise and rotation augmentation while measuring the distance between their filters. Figs. B.6 and B.7 display euclidean distance results for these experiments. These metrics confirm that only translation augmentation is sufficient to endow FCNs with approximate weight-sharing. Additional explanations are provided in Appendix B.5.

While not exact, the distance between FCN filters shows how approximate weight-sharing can emerge in a natural way with translationally augmented data.

5.5. For what learning tasks are free convolutional networks most applicable?

In CNNs, the same filter is applied to all locations in the input space. One may hypothesize that, as a result, CNNs will be focused less on the overall structure of features in spatial relation to each other, and more on identifying the features themselves. This is in contrast to FCNs, which can only operate locally within their receptive field, and so must inherently learn the global structure of features in relation to one another.

To test this hypothesis, images presented to the networks for testing need to be manipulated in such a way so as to compromise the overall global structure of the image while at the same time preserving individual features appearing within an image. Appendix B.3 details the quadrant swap task, in which quadrants I and III of images are interchanged with each other in the validation set. See Fig. B.3 for a visual example. In this task, lower classification accuracy would indicate a higher importance placed on global structure during training. That is, it is not sufficient for learned features just to be present in an image, but the spatial relationship between them must also be preserved to some degree.

The results from the quadrant swap task are shown in Fig. 6. One might expect severe degradation of performance after rearranging parts on an image. However, the CNN performs at nearly 50% accuracy. This performance indicates that the CNN is still able to recognize features of the altered images when making its prediction. Conversely, the FCN performs nearly 20% worse. Higher accuracy bolsters the notion that the overall global structure of an image is less important for a CNN as opposed to the FCN. *In domains where global structure is essential such as face recognition or biomedical imaging, FCNs may find substantial applicability.*

5.6. Is weight-sharing necessary?

The necessity of weight-sharing arises as a means of parameter reduction. Reducing the number of parameters leads to networks that are faster to train and smaller to store. Due to the limitations of modern hardware, practical applications of free weight networks are not currently viable, especially in embedded environments. Thus in situations where space is a constraint, and translationally invariant datasets are not available, weight-sharing becomes a necessity.

In terms of accuracy, FCNs have shown comparable results are possible without weight-sharing. In this regard, weight-sharing is not a necessity.

Table A.1
Hyperparameter space for the conducted grid search.

Name	Options	Parameter type
Learning rate	$10^{-i}, i \in [1, 6]$	Choice
Number of layers	[2, 3]	Discrete

5.7. If weight-sharing is not necessary, are translational invariant training sets necessary?

To examine the necessity of translational datasets, additional experiments were conducted using noise and rotational augmentation during training. Intuition would suggest that only translational data is sufficient to endow FCNs with translational invariant recognition. The full results, including accuracy performance on the noise and rotation, augmented training, un-augmented validation, rotation augmented validation, noise augmented validation, and translation augmented validation set can be found in Appendix B.

Training using other augmentation methods that do not produce translational invariant datasets yield poor results when testing on the translationally augmented validation set. Referencing Appendix B (Tables B.1, B.2, and B.3), the results show that FCNs trained on non-translational data are consistently not capable of learning translationally invariant representations. Low performance indicates that only translationally augmented training data allows FCNs to learn translationally invariant representations. *Using translationally invariant datasets yields better results in validation set and augmented validation set accuracy, specifically in FCNs.*

6. Conclusion

The use of weight-sharing arises as a solution to parameter reduction and translationally invariant recognition in neural networks. Though weight-sharing is implausible in any biological or physical setting, it is instrumental in computer vision tasks. We have examined alternatives to weight-sharing, such as free convolutional networks, where the weight-sharing assumption is relaxed. FCNs trained with augmented datasets have been shown to match and even surpass standard CNNs in validation set accuracy. Data augmentation, specifically datasets augmented via translation, is a necessity as a means to avoid overfitting and train FCNs capable of translationally invariant recognition. Thus, in environments where data is plentiful and computational resources can cope with the large number of parameters that result from abandoning weight-sharing, FCNs provide an alternative to CNNs that can achieve potentially superior performance and higher fidelity to physical systems.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The work of JO and PB partially supported by NSF grant 1839429 and NIH grant GM123558 to PB. The work of EL partially supported by a gift from Experian Consumer Services and a hardware grant from Nvidia.

Appendix A. Experimental settings

A.1. Hyperparameter search

We conducted a hyperparameter search to explore the space of possible architectures. CNN and FCNs trained on MNIST and CIFAR, using a grid search to find the optimal setting. The search was executed using SHERPA (Hertel, Collado, Sadowski, & Baldi, 2018), a Python library for hyperparameter tuning. We detail the hyperparameters of interest in Table A.1, as well as the range of available options during the search.

During the hyperparameter search, MNIST trained for 100 epochs with a patience of 25 monitoring the validation accuracy. CIFAR trained for 200 epochs with a patience of 50 monitoring the validation accuracy. We show the hyperparameters of the best-performing networks in Table A.2. All networks achieved the best performance with three layers. The ultimate difference between MNIST and CIFAR architectures is the learning rate, 10^{-3} and 10^{-4}, respectively.

A.2. Network architectures

Table A.2 describes the architectures used in this paper. MNIST networks have three convolutional or free convolutional layers respective of the architecture. All weight kernels are of size 3×3 with 32, 64, and 128 filters for the three layers. This output feeds

Table A.2
Architecture specification resulting from the grid search. The format for convolutional layers is (kernel size, number of output channels, stride). All layers, except the output, use ReLU activations.

	CNN	FCN
MNIST	Convolutional (3×3, 32, 2)	Free Convolutional (3×3, 32, 2)
	Convolutional (3×3, 64, 2)	Free Convolutional (3×3, 64, 2)
	Convolutional (3×3, 128, 1)	Free Convolutional (3×3, 128, 1)
	Fully connected (1024)	Fully connected (1024)
	Softmax (10)	Softmax (10)
	Learning rate: 10^{-3}	Learning rate: 10^{-3}
CIFAR	Convolutional (5×5,64,2)	Free Convolutional (5×5,64,2)
	Convolutional (5×5, 128, 2)	Free Convolutional (5×5, 128, 2)
	Convolutional (3×3, 256, 1)	Free Convolutional (3×3, 256, 1)
	Fully connected (1024)	Fully connected (1024)
	Softmax (10)	Softmax (10)
	Learning rate: 10^{-4}	Learning rate: 10^{-4}

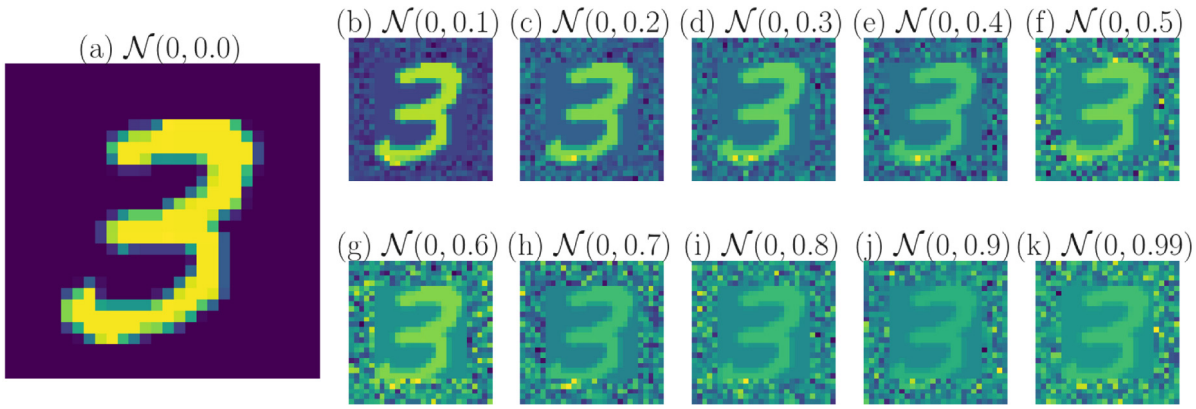

Fig. B.1. Examples of edge noise augmentation on MNIST. (a) 0 variance noise, equivalent to a un-altered MNIST image. (b–k) Gradually increasing the variance of noise augmentation by .1 each time.

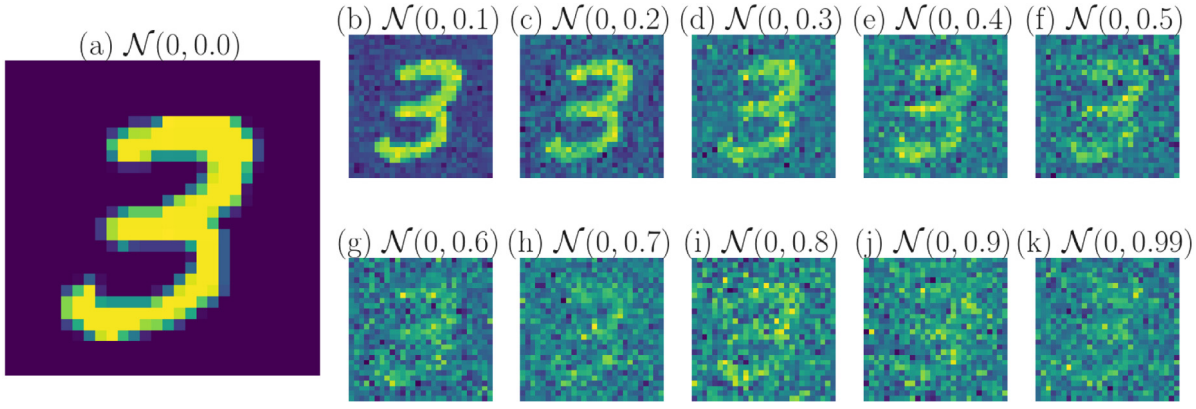

Fig. B.2. Examples of noise augmentation on MNIST. (a) 0 variance noise, equivalent to a un-altered MNIST image. (b–k) Gradually increasing the variance of noise augmentation by .1 each time.

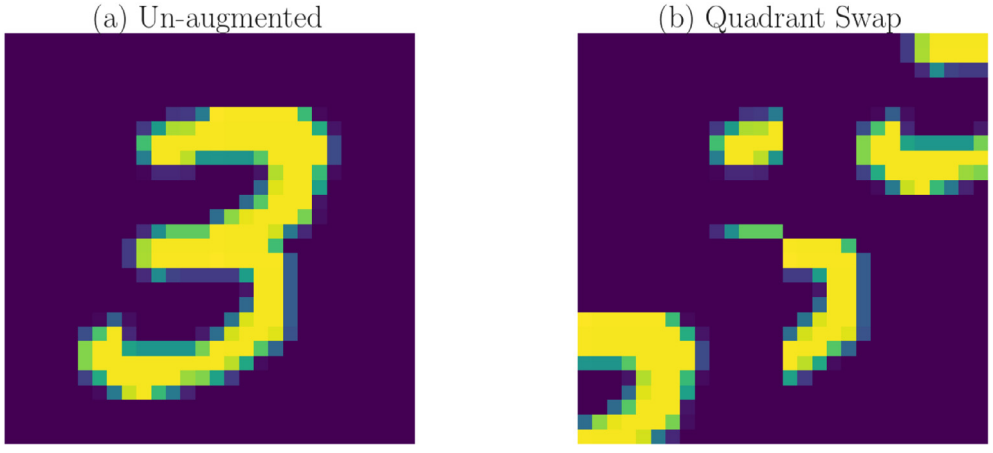

Fig. B.3. Quadrant swap on MNIST. (a) Un-augmented image. (b) Quadrant swap augmentation. Where quadrant I is replaced with quadrant III and III with I.

into a fully connected layer of 1024 nodes, followed by a softmax layer for classification.

CIFAR networks have three convolutional or free convolutional layers respective of the architecture. Layer one has a 5×5 weight kernel, 64 filters, and a stride of 2. Layer two has a 5×5 weight kernels, 128 filters, and a stride of 2. Layer three has 3×3 weight kernels, 256 filters, and a stride of 1. Following these layers are a fully connected layer of 1024 nodes, followed by a softmax layer for classification.

A.3. Implementation details

Random seeds were set to zero for the Tensorflow backend and Numpy. Batch sizes were 256 and 128 for MNIST and CIFAR, respectively. Weights for all layers were initialized using

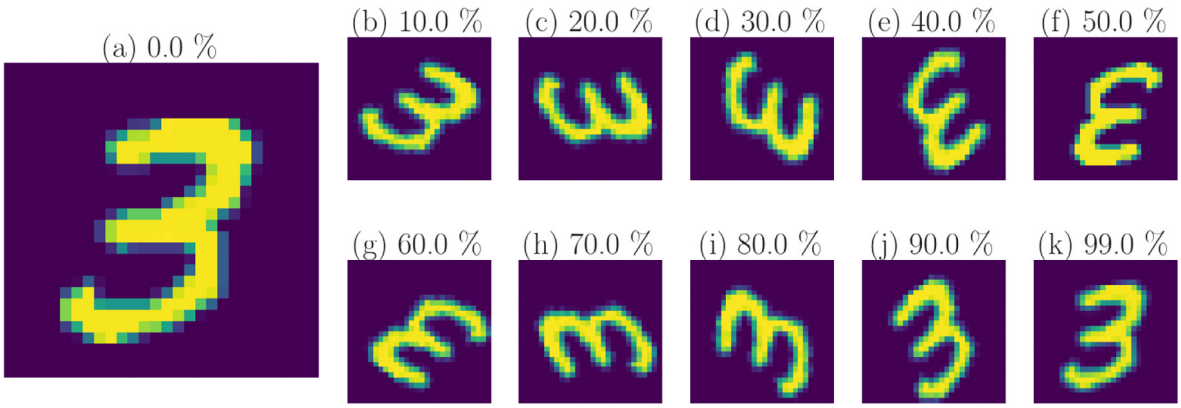

Fig. B.4. Examples of rotation augmentation on MNIST. (a) 0% rotation, equivalent to a un-altered MNIST image. (b–k) Gradually increasing the degree rotation by 10% each time.

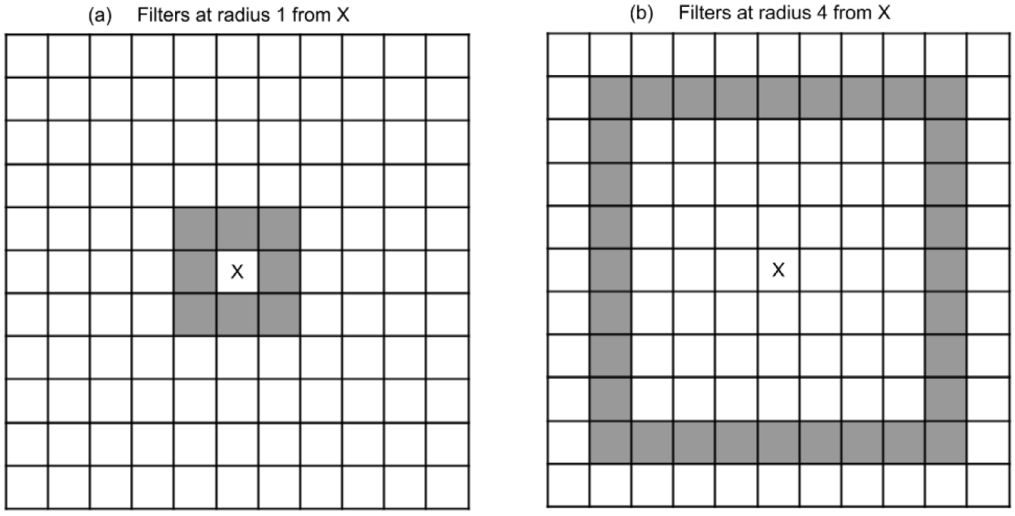

Fig. B.5. Filters at a specified radius. (a) Filters at radius 1 from the desired filter, marked by an X. (b) Filters at radius 4 from the desired filter, marked by an X.

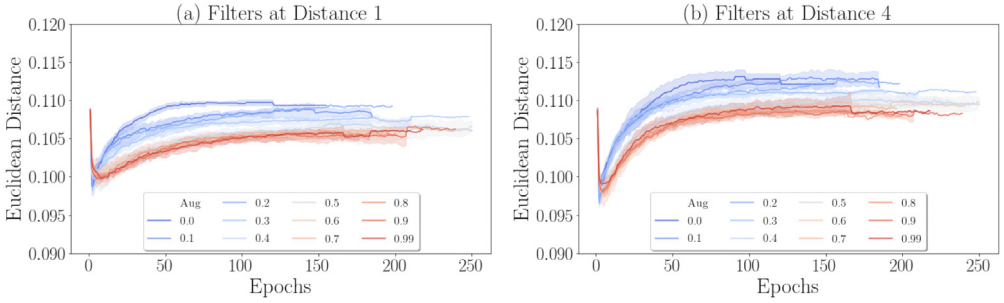

Fig. B.6. Euclidean distance between filters of an FCN layer, trained on MNIST with rotation augmentation. (a) Average euclidean distance between filters one unit away. i.e. all adjacent filters in the layer. (b) Average euclidean distance between filters four units away.

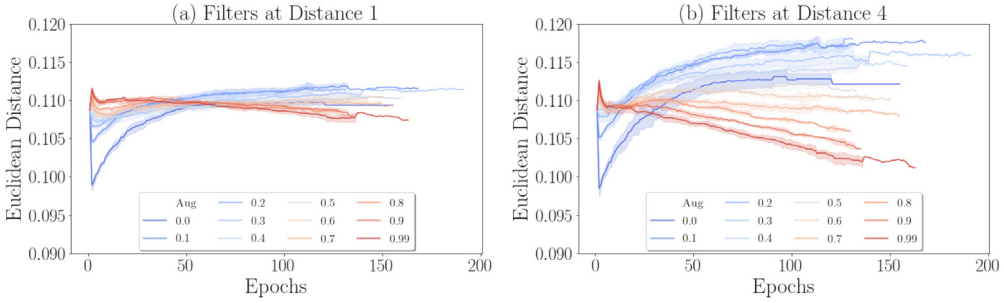

Fig. B.7. Euclidean distance between filters of an FCN layer, trained on MNIST with noise augmentation. (a) Average euclidean distance between filters one unit away. i.e. all adjacent filters in the layer. (b) Average euclidean distance between filters four units away.

Table B.1
Edge noise results. Median accuracy of un-augmented validation and translation augmented validation set. The left most column denotes the amount variance of noise used during training. When performing translational augmentation on the validation set, 25% augmentation was used throughout the experiments. Bold values indicate the highest performing model in that accuracy metric for CNN and FCN, respectively.

Aug %	MNIST				CIFAR			
	Validation Acc		Translation Acc		Validation Acc		Translation Acc	
	CNN	FCN	CNN	FCN	CNN	FCN	CNN	FCN
0.00	0.99054	**0.98807**	**0.36654**	0.37229	**0.67108**	**0.64733**	**0.44456**	**0.43347**
0.10	**0.99100**	0.98786	0.36263	**0.37666**	0.64475	0.62633	0.42742	0.42179
0.20	0.98921	0.98714	0.35503	0.37587	0.63567	0.62375	0.42288	0.42011
0.30	0.98943	0.98714	0.35113	0.36777	0.62992	0.62350	0.41944	0.41986
0.40	0.98950	0.98750	0.34368	0.36053	0.61225	0.62342	0.40835	0.41952
0.50	0.98864	0.98664	0.34107	0.35236	0.60917	0.62167	0.40348	0.41734
0.60	0.98893	0.98750	0.33550	0.34968	0.59083	0.61617	0.39365	0.41541
0.70	0.98843	0.98707	0.33565	0.34650	0.59308	0.61558	0.39415	0.41473
0.80	0.98821	0.98671	0.33200	0.34165	0.58392	0.61375	0.38794	0.41322
0.90	0.98800	0.98679	0.33377	0.33970	0.58058	0.60925	0.38458	0.41112
0.99	0.98829	0.98650	0.32986	0.33767	0.57383	0.60996	0.37912	0.41053

Table B.2
Noise results. Median accuracy of un-augmented validation and translation augmented validation set. The left most column denotes the amount variance of noise used during training. When performing translational augmentation on the validation set, 25% augmentation was used throughout the experiments. Bold values indicate the highest performing model in that accuracy metric for CNN and FCN, respectively.

Aug %	MNIST				CIFAR			
	Validation Acc		Translation Acc		Validation Acc		Translation Acc	
	CNN	FCN	CNN	FCN	CNN	FCN	CNN	FCN
0.00	**0.99054**	0.98807	**0.36654**	0.37229	**0.67108**	**0.64733**	**0.44456**	**0.43347**
0.10	0.99000	0.98750	0.35757	0.37004	0.63092	0.63425	0.41507	0.41919
0.20	0.98957	0.98779	0.35880	**0.37410**	0.59258	0.61825	0.39037	0.40381
0.30	0.99007	0.98829	0.36328	0.36957	0.56258	0.59404	0.37433	0.38899
0.40	0.98979	**0.98879**	0.35793	0.36769	0.53521	0.56983	0.35652	0.37576
0.50	0.98964	0.98857	0.35084	0.34990	0.51142	0.55075	0.34232	0.36794
0.60	0.98900	0.98836	0.33623	0.33261	0.48083	0.53192	0.32451	0.35685
0.70	0.98807	0.98750	0.31854	0.31782	0.44858	0.50850	0.30612	0.34341
0.80	0.98614	0.98586	0.30433	0.30136	0.42417	0.47683	0.29255	0.32502
0.90	0.98407	0.98350	0.29080	0.28516	0.41179	0.44933	0.28642	0.30855
0.99	0.98164	0.98086	0.27590	0.27235	0.38250	0.40042	0.27050	0.28154

the Xavier Uniform Initialization. The source code for all experiments has been made publicly available at: https://github.com/Learning-In-The-Machine/Weight-Sharing

Appendix B. Other augmentation methods

To examine the necessity of translational datasets, additional experiments were conducted using noise and rotational augmentation during training. The hypothesis being that only translational data is sufficient to endow FCNs with translational invariant recognition. The full results for edge noise, noise, and rotation augmentation can be found in Tables B.1, B.2, and B.3, respectively.

Training using other augmentation methods that do not produce translational invariant datasets yield poor results when testing on the translationally augmented validation set. This result indicates that only translationally augmented training data allows FCNs to learn translationally invariant representations. *Using translationally invariant datasets yields better results in validation set and augmented validation set accuracy, specifically in FCNs.*

B.1. Edge noise

Augmentation with edge noise represents adding Gaussian noise to the periphery of an image. For experiments in this paper, the periphery is defined as the 5 pixels bordering an image (for both CIFAR and MNIST). This type of augmentation serves to test the network's resiliency to noise on the fringe. For datasets like MNIST and CIFAR that contain the object of interest in center focus, this noise is unlikely to corrupt the object. Fig. B.1 shows the effect of edge noise on the periphery, starting from zero noise, Fig. B.1a, up to a standard deviation of 0.99, Fig. B.1k.

B.2. Noise

Augmentation with noise represents adding Gaussian noise to the image. This type of augmentation serves to test the network's resiliency to noise corruption. Fig. B.2 shows the effect of edge noise on the periphery, starting from zero noise, Fig. B.2a, up to a standard deviation of 0.99, Fig. B.2k.

B.3. Quadrant swap

Using the convention from Cartesian geometry, quadrants I and III of images are swapped. This type of image deformation serves to test the networks reliance on features as opposed to global structure. The hypothesis being that because of CNNs shared weight paradigm, the location of features matters less opposed to the presence of the feature itself. Conversely, FCN filters can only operate locally within their receptive field, rendering the global structure of the image essential.

Table B.3
Rotation results. Median accuracies of un-augmented validation and translation augmented validation set. The left most column denotes the percentage of rotation used during training. When performing translational augmentation on the validation set, 25% augmentation was used throughout the experiments. Bold values indicate the highest performing model in that accuracy metric for CNN and FCN, respectively.

Aug %	MNIST				CIFAR			
	Validation Acc		Translation Acc		Validation Acc		Translation Acc	
	CNN	FCN	CNN	FCN	CNN	FCN	CNN	FCN
0.00	0.99054	0.98807	0.36654	0.37229	0.67108	0.64733	0.44456	0.43347
0.10	**0.99079**	**0.98836**	**0.38527**	**0.37753**	**0.68233**	**0.65500**	**0.48660**	**0.46211**
0.20	0.98907	0.98571	0.36372	0.35055	0.65746	0.62608	0.47450	0.44766
0.30	0.98779	0.98436	0.33587	0.30107	0.64708	0.61000	0.47106	0.43364
0.40	0.98636	0.98329	0.32147	0.28877	0.63625	0.59767	0.46455	0.42981
0.50	0.98439	0.98114	0.32183	0.29492	0.62242	0.58550	0.46337	0.42465
0.60	0.98400	0.97979	0.32154	0.29854	0.61533	0.57458	0.45682	0.41919
0.70	0.98336	0.97864	0.32060	0.29337	0.60717	0.57100	0.44750	0.41465
0.80	0.98236	0.97786	0.31547	0.29015	0.60875	0.56567	0.44758	0.40961
0.90	0.98071	0.97686	0.31040	0.28566	0.59958	0.55767	0.44414	0.40692
0.99	0.98350	0.97975	0.32429	0.29239	0.61842	0.58108	0.46102	0.42221

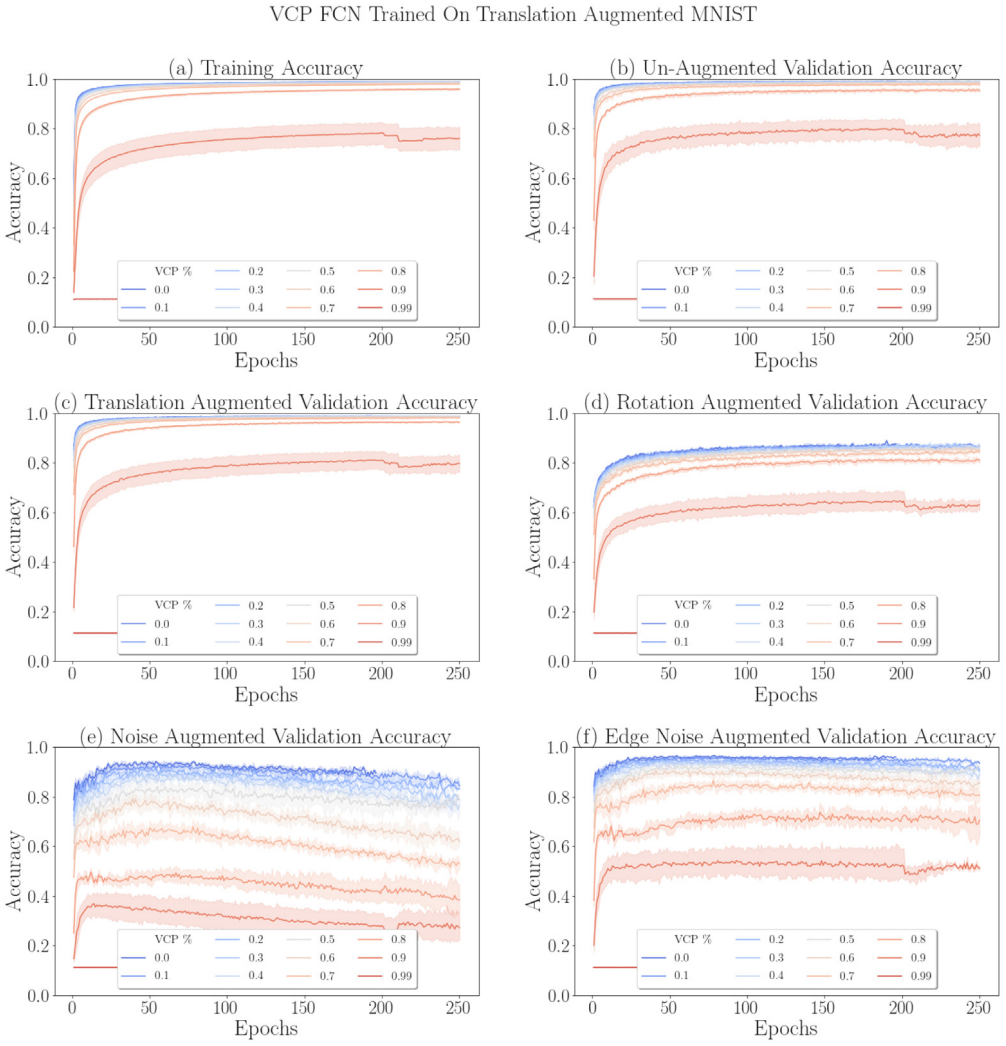

Fig. C.1. FCNs trained with variable connection patterns on MNIST. The legends indicate the probability of absent dentritic connections in a FCN filter. (a) Accuracy on translation augmented training set, 30% translations. (b) Accuracy on the un-augmented validation set. (c) Accuracy on the translation augmented validation set, using 25% translations. (d) Accuracy on the rotation augmented validation set. (e) Accuracy on the noise augmented validation set. (f) Accuracy on the edge noise augmented validation set.

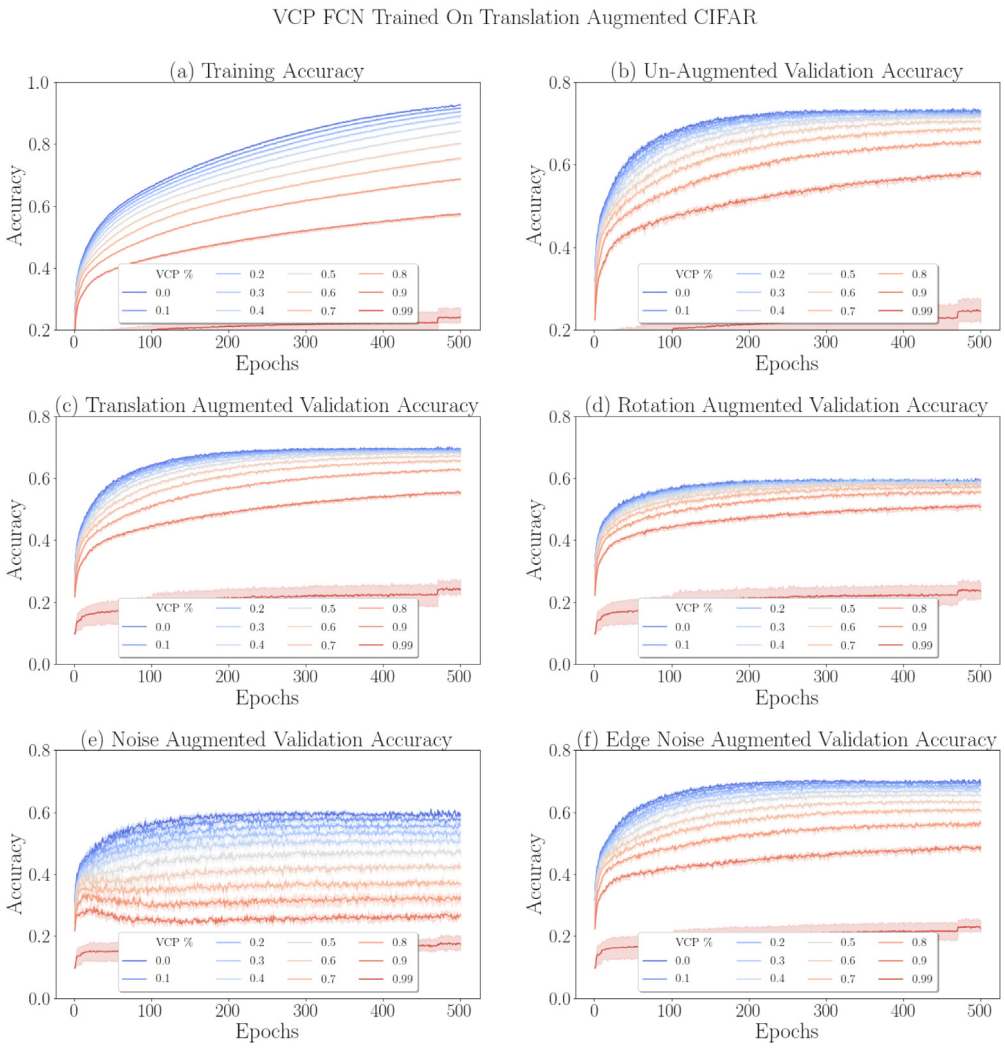

Fig. C.2. FCNs trained with variable connection patterns on CIFAR. The legends indicate the probability of absent dentritic connections in a FCN filter. (a) Accuracy on translation augmented training set, 30% translations. (b) Accuracy on the un-augmented validation set. (c) Accuracy on the translation augmented validation set, using 25% translations. (d) Accuracy on the rotation augmented validation set. (e) Accuracy on the noise augmented validation set. (f) Accuracy on the edge noise augmented validation set.

To test this hypothesis, features of images need to be manipulated in such a way to compromise the overall structure of the image but preserve individual features. Fig. B.3 shows a visual example of the swap procedure.

The results from this task, shown in Fig. 6, indicate CNNs score a higher accuracy on Quadrant swapped images compared to FCNs. This indicates that CNNs are still able to recognize features of the altered images when making their prediction. This bolsters the notion that the overall structure of an image is less important for a CNN as opposed to the FCN, that performs nearly 20% worse. In this task, a lower score indicates higher importance on the global structure.

B.4. Rotation

Rotation augmentation is accomplished via rotating images clockwise about the center point. Fig. B.4 shows the effect of rotating an MNIST image from 0%, Fig. B.4a, up through 99%, Fig. B.4k. Table B.3 displays the results of training networks with rotation augmentation.

B.5. Approximate weight-sharing

The distance between weight kernels within an FCN layer was measured during training with other types of augmentation. This experiment tests if translation augmentation is the cause of approximate weight-sharing in FCNs. Figs. B.6 and B.7 show the results of training with rotation and noise augmentation, respectively.

The results show that for all augmentation settings of rotation, the distance between filters increases over time. Additionally, in noise augmented training, the average Euclidean distance increases in all cases except very high values of noise (0.8, 0.9, 0.99). Indicating this increase in filter similarity is due to the high noise levels across the image. This is confirmed visually by examining Fig. B.2i, j, and k. Also, the decrease in euclidean distance for noise is not as dramatic as observed for translation, Fig. 5.

Appendix C. Variable connection patterns

Also implemented in this study are neurons with variable connection patterns in FCNs. At the start of training, a chosen percentage of weights is randomly set to 0, representing the

absence of a dendritic connection. These missing weights do not contribute to the output of the layer, and their values are not updated during backpropagation. The resulting connection patterns are maintained throughout training and testing. There are multiple options for implementing neurons with variable connection patterns. For computational simplicity, the implementation used in this paper is to turn off connections within each square filter given some probability. In simulations, we vary the probability from 0 to 99% by increments of 10%. The results from these simulations are reported in Figs. C.1 and C.2 for MNIST and CIFAR, respectively. Both datasets use 30% translational augmentation for these experiments.

The variable connection probability is varied from 0% to 99% as indicated by the legend (VCP %). The results shown in Figs. C.1 and C.2 were trained using 20% translation augmentation. Accuracy on the validation set indicates FCNs are robust to even large amounts of missing connections. Even when 80% of connections are absent, the FCN is able to perform comparably well on both MNIST and CIFAR. FCNs are shown to be robust to a high degree of missing connections. Performance degrades rapidly beyond 90%.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems, software available from tensorflow.org. URL http://tensorflow.org/.

Aurisano, A., Radovic, A., Rocco, D., Himmel, A., Messier, M., Niner, E., et al. (2016). A convolutional neural network neutrino event classifier. *Journal of Instrumentation*, *11*(09), P09001, URL http://stacks.iop.org/1748-0221/11/i=09/a=P09001.

Baldi, P., Bauer, K., Eng, C., Sadowski, P., & Whiteson, D. (2016). Jet substructure classification in high-energy physics with deep neural networks. *Physical Review D*, *93*(9), 094034.

Baldi, P., & Chauvin, Y. (1993). Neural networks for fingerprint recognition. *Neural Computation*, *5*(3), 402–418.

Baldi, P., Lu, Z., & Sadowski, P. (2017). Learning in the machine: The symmetries of the deep learning channel. *Neural Networks*, *95*, 110–133.

Baldi, P., Lu, Z., & Sadowski, P. (2018). Learning in the machine: Random backpropagation and the deep learning channel. *Artificial Intelligence*, *260*, 1–35, Also: arXiv:1612.02734.

Baldi, P., & Sadowski, P. (2014). The dropout learning algorithm. *Artificial Intelligence*, *210C*, 78–122.

Baldi, P., & Sadowski, P. (2016). A theory of local learning, the learning channel, and the optimality of backpropagation. *Neural Networks 83*, 51–74.

Baldi, P., & Sadowski, P. (2018). Learning in the machine: Recirculation is random backpropagation. *Neural Networks*, *108*, 479–494.

Baldi, P., Sadowski, P., & Lu, Z. (2017). Learning in the machine: The symmetries of the deep learning channel. *Neural Networks*, *95*, 110–133.

Baldi, P., Sadowski, P., & Lu, Z. (2018). Learning in the machine: Random backpropagation and the deep learning channel. *Artificial Intelligence*, *260*, 1–35.

Bartunov, S., Santoro, A., Richards, B., Marris, L., Hinton, G. E., & Lillicrap, T. (2018). Assessing the scalability of biologically-motivated deep learning algorithms and architectures. In *Advances in neural information processing systems* (pp. 9368–9378).

Bromley, J., Bentz, J. W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., et al. (1993). Signature verification using a siamese time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, *7*(4).

Cadieu, C. F., Hong, H., Yamins, D. L. K., Pinto, N., Ardila, D., Solomon, E. A., et al. (2014). Deep neural networks rival the representation of primate IT cortex for core visual object recognition. *PLoS Computational Biology*, URL https://doi.org/10.1371/journal.pcbi.1003963.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., et al. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

Chollet, F., et al. (2015). *Keras*. GitHub, https://github.com/fchollet/keras.

Cireş, D., Giusti, A., Gambardella, L. M., & Schmidhuber, J. (2012). Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in neural information processing systems* (pp. 2843–2851).

Cun, Y. L., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., et al. (1990). Handwritten digit recognition with a back-propagation network. In D. Touretzky (Ed.), *Advances in neural information processing systems* (pp. 396–404). San Mateo, CA: Morgan Kaufmann.

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., et al. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, *542*(7639), 115–118.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, *36*(4), 193–202.

Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., et al. (2016). Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Jama*, *316*(22), 2402–2410.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385.

Hertel, L., Collado, J., Sadowski, P., & Baldi, P. (2018). Sherpa: Hyperparameter optimization for machine learning models. URL https://github.com/sherpa-ai/sherpa.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

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

Jan, Y.-N., & Jan, L. Y. (2010). Branching out: Mechanisms of dendritic arborization. *Nature Reviews Neuroscience*, *11*(5), 316–328, URL http://dx.doi.org/10.1038/nrn2836.

Kayala, M., & Baldi, P. (2012). ReactionPredictor: Prediction of complex chemical reactions at the mechanistic level using machine learning. *Journal of Chemical Information and Modeling*, *52*(10), 2526–2540.

Krizhevsky, A., Nair, V., & Hinton, G. (2009). CIFAR-10 (Canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

LeCun, Y., & Cortes, C. (2010). MNIST handwritten digit database [cited 2016-01-14 14:24:11]. URL http://yann.lecun.com/exdb/mnist/.

Lee, D.-H., Zhang, S., Fischer, A., & Bengio, Y. (2015). Difference target propagation. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 498–515). Springer.

Lillicrap, T. P., Cownden, D., Tweed, D. B., & Akerman, C. J. (2016). Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, *7*, 13276.

Ott, J. (2019). Questions to guide the future of artificial intelligence research. arXiv preprint arXiv:1912.10305.

Ott, J., Atchison, A., Harnack, P., Bergh, A., & Linstead, E. (2018). A deep learning approach to identifying source code in images and video. In *2018 IEEE/ACM 15th international conference on mining software repositories* (pp. 376–386). IEEE.

Ott, J., Atchison, A., Harnack, P., Best, N., Anderson, H., Firmani, C., et al. (2018). Learning lexical features of programming languages from imagery using convolutional neural networks. In *Proceedings of the 26th conference on program comprehension* (pp. 336–339). ACM.

Sadowski, P., Radics, B., Ananya, Yamazaki, Y., & Baldi, P. (2017). Efficient antihydrogen detection in antimatter physics by deep learning. *Journal of Physics Communications*, *1*(2), 025001, URL http://stacks.iop.org/2399-6528/1/i=2/a=025001.

Samadi, A., Lillicrap, T. P., & Tweed, D. B. (2017). Deep learning with dynamic spiking neurons and fixed feedback weights. *Neural Computation*, *29*(3), 578–602.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85–117.

Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). Training very deep networks. In *Advances in neural information processing systems* (pp. 2368–2376).

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, *15*(1), 1929–1958.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1–9).

Tompson, J., Stein, M., Lecun, Y., & Perlin, K. (2014). Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics*, *33*(5), 169.

Tran, D., Bourdev, L., Fergus, R., Torresani, L., & Paluri, M. (2015). Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 4489–4497).

Urban, G., Tripathi, P., Alkayali, T., Mittal, M., Jalali, F., Karnes, W., et al. (2018). Deep learning achieves near human-level polyp detection in screening colonoscopy. *Gastroenterology 155*(4), 1069–1078.

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., & Fergus, R. (2013). Regularization of neural networks using dropconnect. In *International conference on machine learning* (pp. 1058–1066).

Wang, J., Ding, H., Azamian, F., Zhou, B., Iribarren, C., Molloi, S., et al. (2017). Detecting cardiovascular disease from mammograms with deep learning. *IEEE Transactions on Medical Imaging*, *36*(5), 1172–1181.

Wang, J., Fang, Z., Lang, N., Yuan, H., Su, M.-Y., & Baldi, P. (2017). A multi-resolution approach for spinal metastasis detection using deep siamese neural networks. *Computers in Biology and Medicine*, *84*, 137–146.

Yamins, D. L. K., Hong, H., Cadieu, C. F., Solomon, E. A., Seibert, D., & DiCarlo, J. J. (2014). Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences*, *111*(23), 8619–8624. http://dx.doi.org/10.1073/pnas.1403112111, arXiv:http://www.pnas.org/content/111/23/8619.full.pdf, URL http://www.pnas.org/content/111/23/8619.abstract.

Zipser, D., & Andersen, R. A. (1988). A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. *Nature*, *331*(6158), 679–684.