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

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

[a]*Schmid College of Science and Technology*
*Chapman University*
[b]*Department of Computer Science*
*University of California, Irvine*

**Abstract**

Weight-sharing is one of the pillars behind Convolutional Neural Networks and their successes. However, in physical neural systems such as the brain, exact weight-sharing is extremely implausible. This raises the fundamental question of whether weight-sharing is really 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 are able to show that while weight-sharing is a pragmatic optimization approach, it not a necessity in computer vision applications. Furthermore, Free Convolutional Networks are able to match and at times surpass the performance observed in standard convolutional architectures when trained using properly translated data (akin to video). In simulations on the CIFAR-10 dataset FCNs are able to achieve a validation set accuracy of 77% vs 70% from the CNN.

*Keywords:* neural networks, computer vision, convolution, weight sharing, overfitting, neural development

## 1. Introduction

Digital simulations of neural network 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 where 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 [1, 2]. When applied to synapses, learning in the machine can lead for instance to the discovery

*Corresponding author
*Email addresses:* ott109@mail.com (Jordan Ott), linstead@chapman.edu (Erik Linstead), lahay100@mail.chapman.edu (Nicholas LaHaye), pfbaldi@ics.uci.edu (Pierre Baldi)

of local learning [3] and random backpropagation [4, 5, 6]. And when applied to layers of neurons, as we do in this short paper, learning in the machine leads one to question the fundamental assumption of weight sharing behind convolutional neural networks.

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 is particularly true in computer vision where weight-sharing is one of the pillars behind convolutional neural networks (CNNs) and their successes. Yet 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 it is difficult to create identical weights at a given time point, but it is also very difficult to maintain the identity over time, both during phases of development and learning when the weights may be changing rapidly, or during more mature phases when weights must retain their integrity against the microscopic entropic forces surrounding any physical synapse. Furthermore, at least in the case of biology, given the exquisitely complex geometry of neuronal dendritic trees and axon arborizations, it is also implausible that they could form large arrays of neurons with identically translated connection patterns. In short, not only is it difficult to exactly share weights, but it is also difficult to exactly share the same connection patterns.

Thus, paradoxically, 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 raises the fundamental question of whether weight-sharing is really 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.

## 2. Origins and Functions of Weight-Sharing

Before addressing the question of its necessity, it is useful to briefly review the origins and functions of weight-sharing. The concept of weight-sharing can be traced back to the neurophysiological work of Hubel and Wiesel [7] on the cat visual cortex, suggesting 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. The ideas proposed by Hubel and Wiesel were systematically used by Fukushima who proposed the neocognitron architecture for computer vision, essentially a convolutional neural network architecture with Hebbian learning. However, Hebbian learning alone applied to a feedforward CNN cannot solve vision tasks [3]. Solving vision tasks requires feedback channels and learning algorithms for transmitting target information to the deep synapses, and this is precisely what is achieved by backpropagation, or stochastic gradient descent. Successful CNNS for vision problem trained by backpropagation were developed already in the late 80s and 90s [8, 9, 10].

Substantial improvements in the size of the training sets and the available computing power, have led to a new wave of successful implementations in recent years, [11, 12, 13, 14], as well as applications to a variety of specific domains, ranging from biomedical images

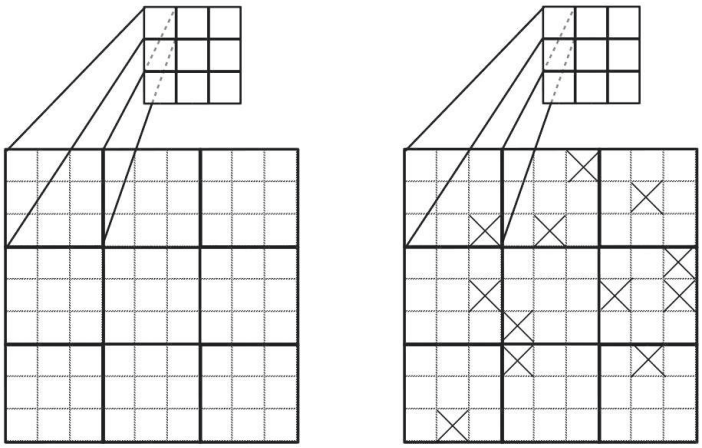

Figure 1: Unlike typical convolutional layers, where the same filter is applied across all possible locations, free convolutional layers maintain a separate filter at each location. The above figures are examples of FCN layers on a 9x9 input space. Each 3x3 subregion of the input is covered with a distinct filter, 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 connections that are absent. In this example 12, out of the 91 connections are missing, creating a variable connection probability of roughly 0.15, whereas in the figure on the left there are no absent connections.

[15, 16, 17, 18, 19, 20] to particle physics [21, 22, 23]. Older [24] as well as more recent work [25, 26] has also shown that not only 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 [27, 9, 28], which also includes Siamese CNNs for images. Finally, a different kind of weight-sharing that will not concern us here, is obtained when a recurrent network is unfolded in time. In this case, weight-sharing occurs over time and not over space.

In terms of functions, weight-sharing is typically associated with two main but different purposes. The first is to reduce the number of free parameters that need to be stored or updated during learning. This 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. The second function is to apply the exact 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

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 Figure 1). Thus naturally FCNs have far more parameters than the corresponding CNN and slower to train on a digital machine. However

Figure 2: MNIST examples of data augmentation (left). CIFAR-10 examples of data augmentation (right).

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.

Furthermore, it is highly implausible that a given neuron will share the exact same dendritic tree with a neighboring neuron [29], thus in addition to neighboring neurons of the same layer not having exactly the same weights, we would like to consider also the possibility of them not having the exact same receptive field pattern. Thus, in addition to plain FCNs, FCNs with variable connection patterns are implemented in the simulations below. Variable connection patterns can be achieved in many ways. Here, for simplicity, a random percentage of connections are set to zero once for all (Figure 1) [Note: this is very different from dropout where different sets of weights are randomly set to 0 at each presentation of a training example]. The x's in the right image of Figure 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 following questions in regards to weight-sharing: Is weight-sharing necessary? Is weight-sharing necessary to prevent overfitting? Is weight-sharing necessary to ensure translational invariant recognition? Can good performance be achieved without weight-sharing? If weight-sharing is not necessary, are translational invariant training sets necessary? Does approximate or exact weight-sharing emerge in a natural way? Can even better performance be achieved without weight-sharing?

## 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 dataset: the handwritten digit dataset, MNIST [30], as well as the CIFAR-10 object dataset [31].

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. Due to

| Method | MNIST | | CIFAR-10 | |
|---|---|---|---|---|
| | Parameters | Data set size | Parameters | Data set size |
| CNN | 1,199,882 | 58,333 | 6,447,562 | 50,000 |
| FCN* | 12,051,082 | 3,733,312 | 21,735,434 | 4,050,000 |
| FCN | 12,051,082 | 58,333 | 21,735,434 | 50,000 |
| FCN*† | 11,982,551 | 3,733,312 | 21,647,846 | 4,050,000 |
| FCN | 11,982,551 | 58,33 | 21,647,846 | 50,000 |

Table 1: Number of parameters in each network as well as the amount of training examples available to the network at each fold of training. The data set size can be calculated by the amount of vertical and horizontal translation performed on the original training data (i.e. horizontally translating one image by zero to ten pixels creates ten additional training samples). * Data augmentation was used during training. † Variable Connection patterns with ten percent probability.

the local receptivity of free weight networks, individual filters learn features solely within their receptive field. More or less translationally invariant data should allow filters to learn more or less the same features. In practice, during training, images were shifted horizontally and vertically by a random amount (0-25%) of the width and height respectively. Shifting images by more than 25% often causes the object of interest to partially leave the image frame. Points outside the boundaries of the input are filled according to the nearest pixels. Figure 2 shows examples of augmentation results on MNIST and CIFAR-10. All simulations were completed using six-fold cross validation, which allowed for roughly 10,000 images in every fold validation set. Simulations were implemented in Keras [32] with a Tensorflow [33] backend using NVIDIA GeForce GTX Titan X GPUs with 12 GB memory.

### *4.1. Networks*

Five networks were trained on each dataset, ten in total. A CNN without data augmentation, a FCN with and without data augmentation, and a variable connection pattern FCN with and without data augmentation. Table 1 provides the size of each data set as well as the amount of parameters each network contains.

For simplicity, the architecture of these networks are very similar, except that convolutional layers are replaced by free convolutional layers in both FCNs. The activation functions, pooling layers, softmax layer, as well as the number of filters in each layer remain the same across all networks. Visualizations of these networks can be seen in Figures 3 and 4. It is important to note that there is no architectural difference between the networks that are trained with data augmentation and those trained without.

For weight initialization we use the Xavier initialization [34]. The weights of each filter are drawn from a uniform distribution (Equation 1), where $x_{in}$ represents the number of incoming connections to the filter and $x_{out}$ is the number of outgoing connections. The value, $x_{in} + x_{out}$, is the same for all filters in corresponding layers of FCNs and LCNs. This allows CNN and FCN filters to be drawn from the same initial distributions.

$$U\Big[-\frac{\sqrt{6}}{\sqrt{x_{in}+x_{out}}}, \frac{\sqrt{6}}{\sqrt{x_{in}+x_{out}}}\Big] \tag{1}$$

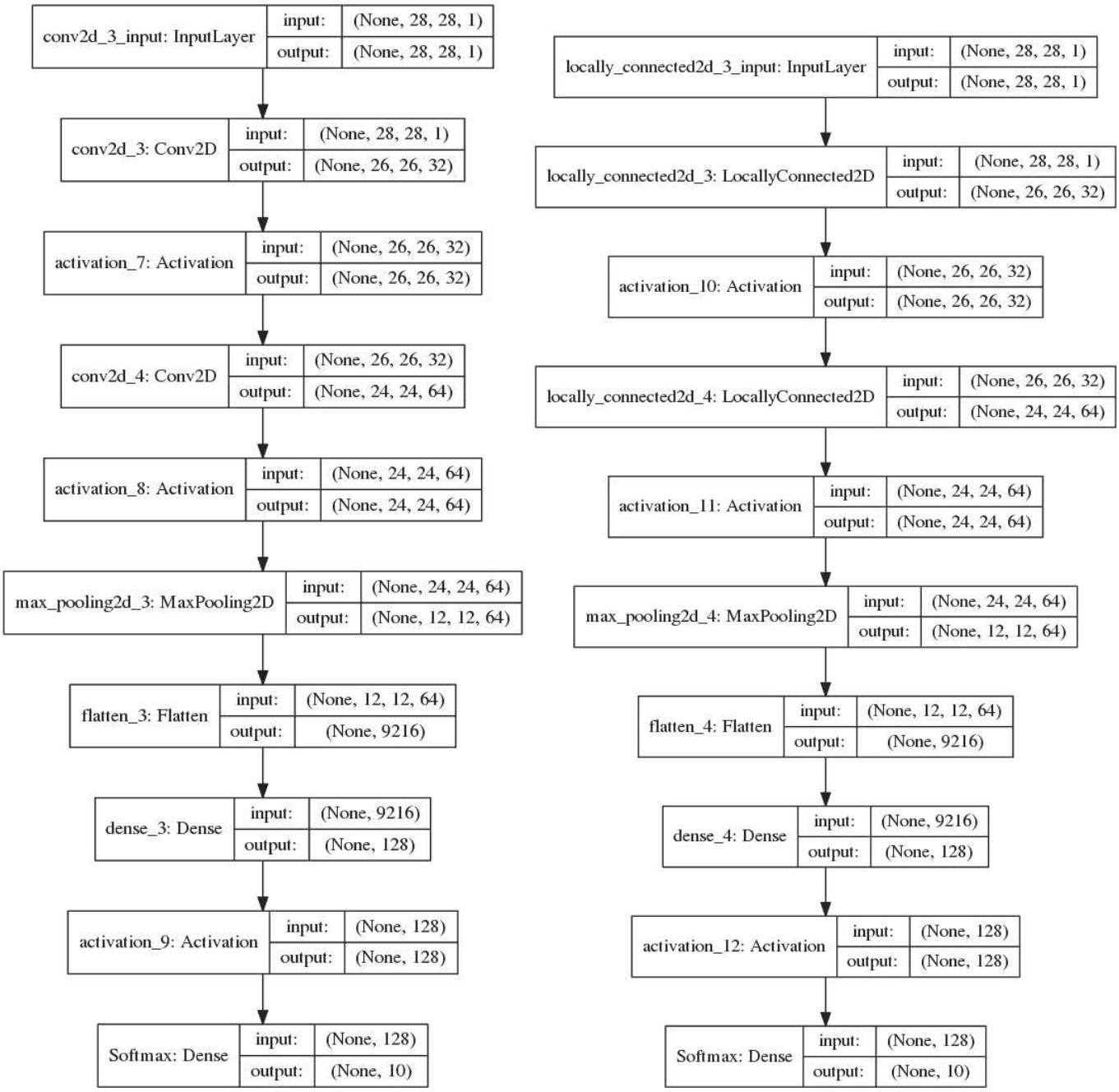


Figure 3: Networks for MNIST dataset. Shared weight network(left) and free weight network (right).

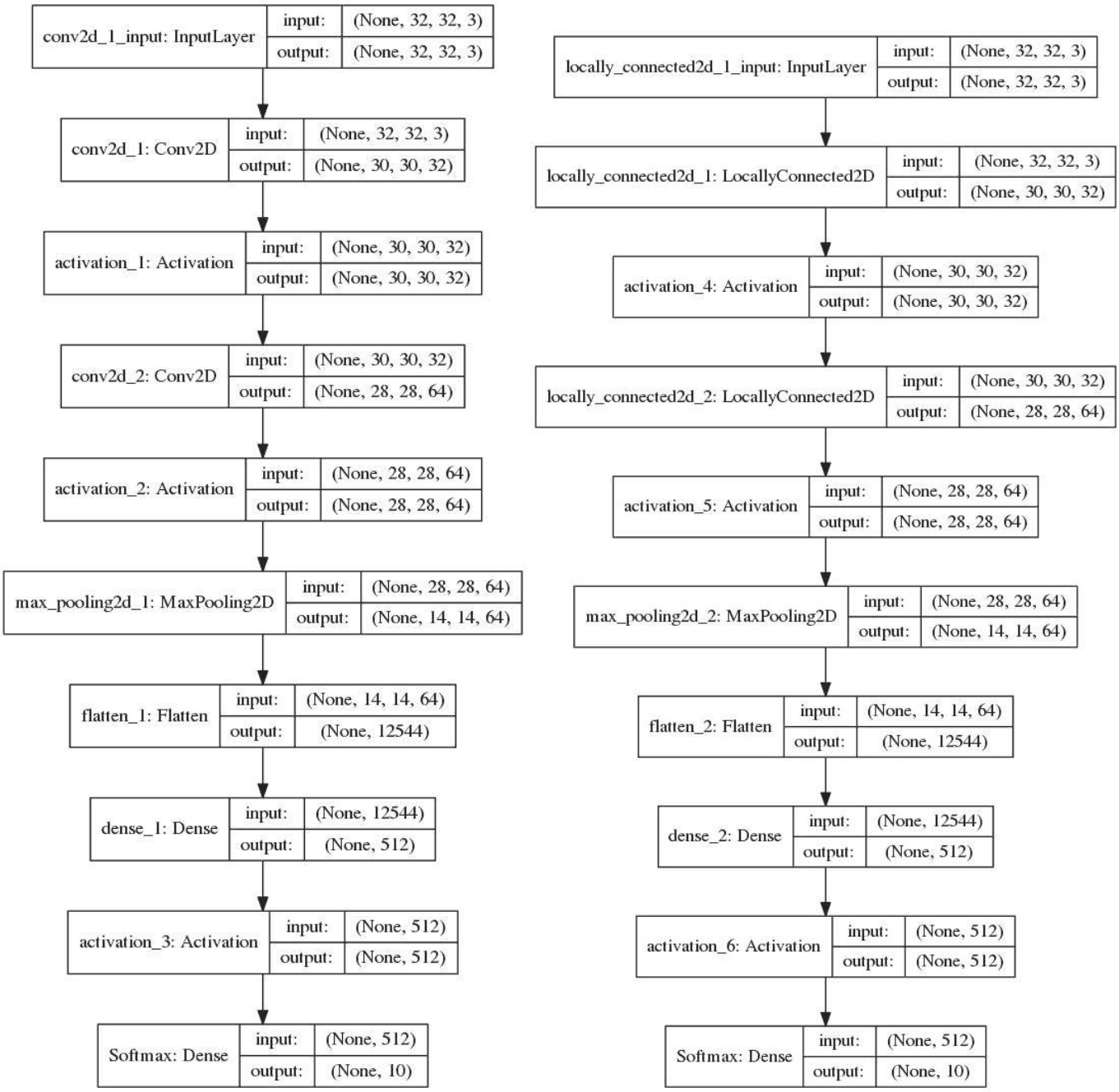


Figure 4: Networks for CIFAR-10 dataset. Shared weight network(left) and free weight network (right).

| Method | MNIST | | | | CIFAR-10 | | | |
|---|---|---|---|---|---|---|---|---|
| | Mean Test Acc | Median Test Acc | Mean Aug Test | Median Aug Test | Mean Test Acc | Median Test Acc | Mean Aug Test | Median Aug Test |
| CNN | **98.984** | **99.031** | 64.882 | 65.008 | 70.33 | 70.28 | 60.149 | 60.238 |
| FCN$^{*}$ | 98.654 | 98.706 | **98.592** | **98.582** | 76.85 | 76.595 | **76.123** | 75.973 |
| FCN | 95.503 | 98.564 | 59.901 | 60.875 | 67.818 | 67.77 | 58.068 | 58.098 |
| FCN$^{\dagger *}$ | 98.633 | 98.616 | 98.553 | 98.562 | **77.015** | **76.960** | 76.043 | **75.997** |
| FCN$^{\dagger}$ | 97.038 | 98.641 | 61.434 | 62.203 | 67.663 | 67.675 | 57.913 | 57.948 |

Table 2: Results of the six-fold cross validation experiments. FCN weights were initialized with the Xavier initialization per individual filter, so that they would match CNN weights. Columns for each data set correspond to: Mean accuracy on non-augmented test set; Median accuracy on non-augmented test set; Mean accuracy on augmented test set; Median accuracy on non-augmented test set. Each filter was initialized using the Xavier Uniform initialization. CNN and FCN weights were drawn from the same uniform distribution. $^{*}$Data augmentation was used during training. $^{\dagger}$Variable Connection patterns with ten percent probability.

### *4.2. Variable Connection Patterns*

We also implemented variable connection patterns in FCNs. At the start of training a chosen percentage of weights are randomly set to 0. These missing weights do not contribute to the output of the layer and their values are never updated during backpropagation training. 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 use a missing connection probability of ten percent.

Using neurons with variable connections results in fewer total network parameters. As a result we add additional filters to free convolutional layers to compensate for this (Table 1). This allows FCNs with variable connection patterns to have roughly the same number of parameters as standard FCNs. Neurons with variable connection patterns are only implemented in free weight simulations.

## 5. Results

We report the percent error on the normal and augmented validation sets of each fold in Figures 5 and 6. Table 2 shows the mean and median accuracy of the six-fold cross validation experiments for both the MNIST and CIFAR-10 datasets. Over the course of training, the highest validation set accuracy was recorded and averaged with the highest validation set accuracy of the other five folds. Refer to the supplementary material for graphs on training accuracy, validation set accuracy, augmented validation set accuracy and validation set loss. Supplementary material can be found on the web at: `https://github.com/mlat/weightsharing`

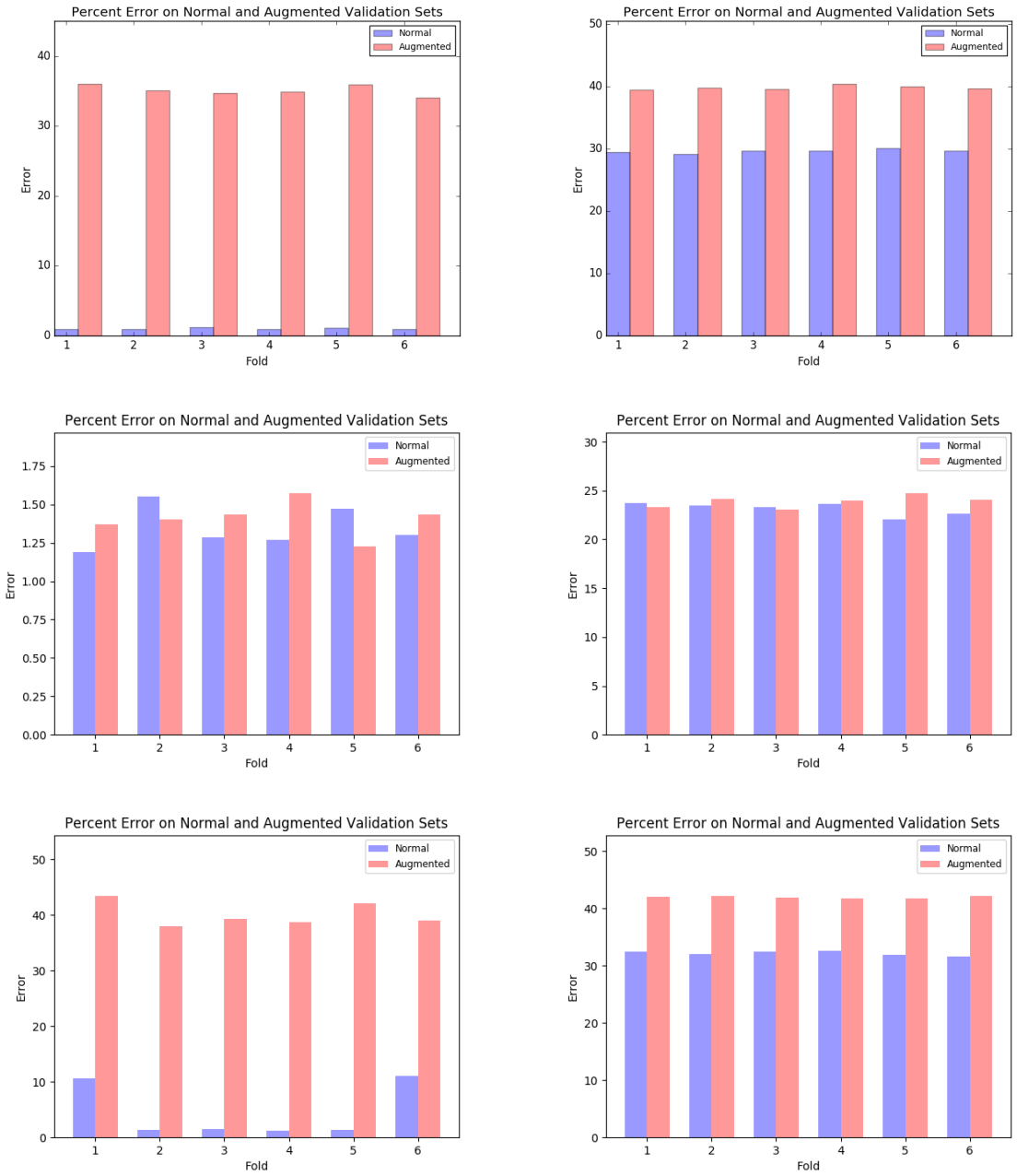


Figure 5: Percent error on the normal and augmented validation set of each fold. The left column shows MNIST results, from top down: CNN trained without augmentation, FCN trained with augmentation, and FCN trained without augmentation. The right column shows CIFAR-10 results, from top down: CNN, FCN trained with augmentation, and FCN trained without augmentation.

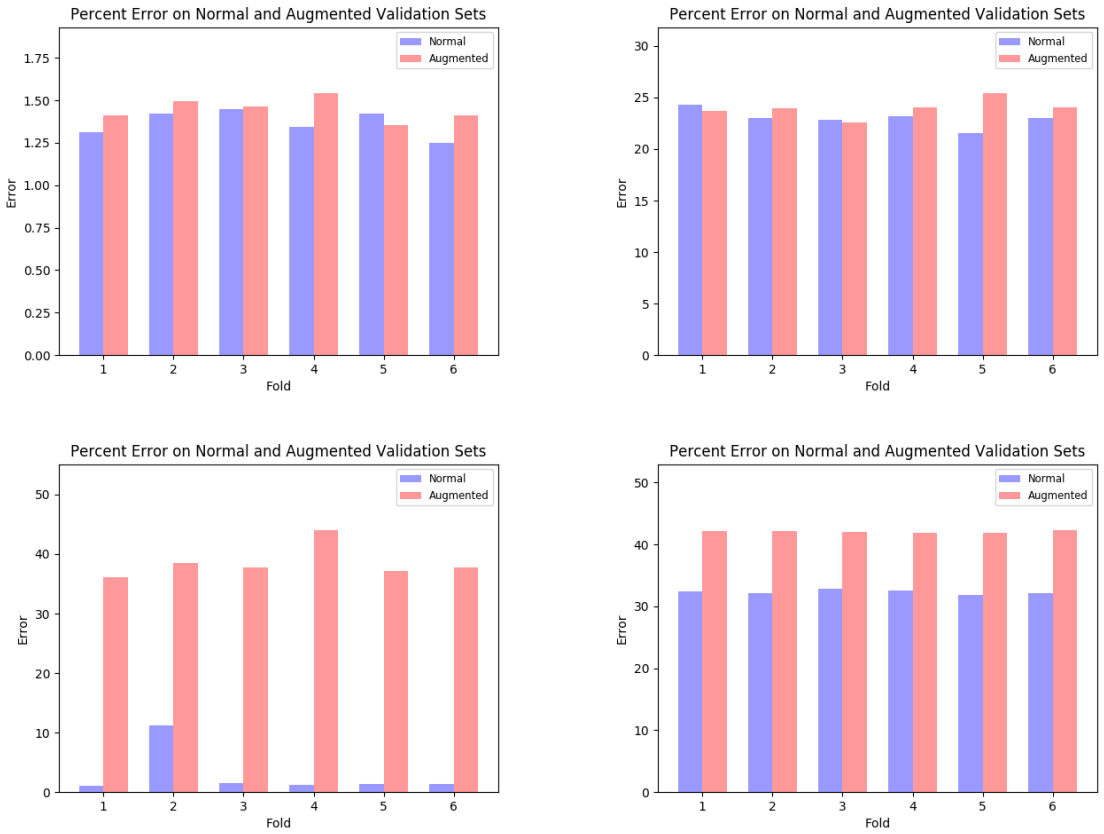


Figure 6: Percent error on the normal and augmented validation set of each fold, for networks with variable connection patterns. The left column shows MNIST results, from top down: FCN trained with augmentation, and FCN trained without augmentation. The right column shows CIFAR-10 results, from top down: FCN trained with augmentation, and FCN trained without augmentation. Neurons with variable connection patterns are only implemented in free weight simulations.

*5.1. Is weight-sharing necessary?*

Reducing the number of parameters leads to networks that are faster to train and smaller to store. Thus exact weight sharing can be necessary in three situations: (1) when space is a constraint in a digital simulation; when learning time is a constraint in a digital simulation; or (2) when large translated (akin to video) training sets are not available. *However, weight sharing is not necessary in a physical neural system that has access to translated training data. In particular FCNs, while not necessarily practical for digital simulations, achieve accuracies comparable to CNNs when trained with augmented data or video.*

*5.2. Is weight-sharing necessary to prevent overfitting?*

Weight sharing can help mitigate overfitting in the regime of small training sets. Both CNNs and FCNs will overfit on small, non-augmented data sets, but FCNs more so. Continuously increasing the number of samples in the dataset using augmentation prevents models from overfitting training sets. *Large, translationally invariant datasets, like those produced here through data augmentation, are essential for free weight networks to achieve good performance and avoid overfitting. If this constraint can be met, overfitting can be mitigated without weight-sharing.*

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

Learning translationally invariant representations is tested by using augmentation on the validation set. Table 2 shows the performance of different models on the augmented test set. Error bars for each of the six-fold cross-validation experiments are displayed in Figure 5 and 6. Great disparity between validation accuracy and 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 has nearly a forty percent gap between validation set accuracy and augmented validation set accuracy. We are able to show that neurons with variable connection patterns are capable of learning translationally invariant representations. Referencing the results in Table 2 and Figure 6 we can see the proximity between the accuracy on the non augmented test set and the augmented one. *For standard FCNs, the results show there is less than a 2% gap between the validation accuracy and augmented validation accuracy of a FCN trained with data augmentation, proving that FCNs can learn translationally invariant representations when provided with sufficient data. Thus exact weight-sharing is not necessary to ensure translational-invariant recognition. As can be expected, approximate weight sharing emerges naturally when training FCNs with properly augmented data.* (see below)

*5.4. Can good performance be achieved without weight-sharing?*

FCNs are able to achieve high accuracy scores on two standard computer vision benchmark datasets. In the case of MNIST, we see from Table 2 that FCNs, trained on augmented data, achieve a mean accuracy of 98.59%, only slightly less than the 98.98% mean accuracy achieved by standard CNNs. On the more complicated CIFAR-10 dataset, FCNs are able to outperform CNNs by a margin of 6-7% depending on the utilization of augmented test sets and variable connection patterns. *The validation set accuracy observed in our simulations*

*show that a FCN is able to meet or exceed the accuracy of a standard CNN on both benchmark datasets when able to take advantage of augmented data for training. Thus, again there is not a fundamental necessity of weight-sharing to attain satisfactory performance.*

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

Analysis of the weights (not shown) shows that exact weight-sharing does not emerges even with translationally augmented data. However, as can be expected, with translationally augmented data two similar units do see roughly the same training data and therefore, except for initial differences and other random fluctuations such as the order in which samples are presented, do converge to roughly the same weights. *In short, approximate but not exact weight-sharing emerges in a natural way with translationally augmented training data.*

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

Weight sharing is not a necessity in computer vision tasks, as FCNs have demonstrated high classification accuracy on two benchmark datasets. As alluded to previously, translationally invariant datasets are necessary to combat overfitting and achieve translationally invariant recognition. The consequences of non-translationally invariant data sets is shown in Figures 5 and 6. The classification accuracy, on augmented test sets, of FCNs with and without neurons with variable connection patterns drops significantly when a translationally invariant training set is not provided. However, when translationally invariant data sets are available the accuracy of FCNs on normal and augmented test sets is within 2%. *Using translationally invariant datasets yields better results in validation set and augmented validation set accuracy, specifically in FCNs.*

## 6. Conclusion

The use of exact weight sharing was somewhat paradoxically inspired by biology and the work of Hubel and Wiesel. In digital simulations of neural networks, weight-sharing provides an efficient solution for both parameter reduction and translational-invariant recognition in neural networks. In digital simulations, CNNs are more compact, faster to train, and more robust against overfitting than FCNs.

However, exact weight sharing is implausible in biological and other physical neural systems. We have examined alternatives to weight-sharing, such as free convolutional networks, where the weight-sharing assumption is relaxed. FCNs trained with translationally augmented datasets have been shown to match and possibly even surpass standard CNNs in validation set accuracy. Data augmentation has been proven to be 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 are able to 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. During development and throughout life, primate brains have access to plenty of translated visual data, due to objects moving in the visual field, or to apparent motion induced by head or eye movements. Furthermore, computational issues associated

with having to train an FCN rather than a CNN vanish in a physical system where synapses learn in parallel. Finally, fine tuning of the synaptic weights of each neuron in the same filter family, possibly throughout one's life time, is likely to be necessary to compensate for small anatomical, physiological, or other fluctuations and lead to better performance.

## References


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

[2] P. Baldi, P. Sadowski, The dropout learning algorithm, Artificial Intelligence 210C (2014) 78–122.

[3] P. Baldi, P. Sadowski, A theory of local learning, the learning channel, and the optimality of backpropagation, Neural NetworksTo appear.

[4] T. P. Lillicrap, D. Cownden, D. B. Tweed, C. J. Akerman, Random synaptic feedback weights support error backpropagation for deep learning, Nature Communications 7.

[5] P. Baldi, Z. Lu, P. Sadowski, Learning in the machine: Random backpropagation and the deep learning channel, Artificial IntelligenceIn press. Also: arXiv:1612.02734.

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

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

[8] Y. L. Cun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, L. Jackel, Handwritten digit recognition with a back-propagation network, in: D. Touretzky (Ed.), Advances in Neural Information Processing Systems, Morgan Kaufmann, San Mateo, CA, 1990, pp. 396–404.

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

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

[11] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.

[13] R. K. Srivastava, K. Greff, J. Schmidhuber, Training very deep networks, in: Advances in Neural Information Processing Systems, 2015, pp. 2368–2376.

[14] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, arXiv preprint arXiv:1512.03385.

[15] D. Cireş, A. Giusti, L. M. Gambardella, J. Schmidhuber, Deep neural networks segment neuronal membranes in electron microscopy images, in: Advances in neural information processing systems, 2012, pp. 2843–2851.

[16] V. Gulshan, L. Peng, M. Coram, M. C. Stumpe, D. Wu, A. Narayanaswamy, S. Venugopalan, K. Widner, T. Madams, J. Cuadros, et al., Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs, Jama 316 (22) (2016) 2402–2410.

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

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

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

[20] G. Urban, P. Tripathi, T. Alkayali, M. Mittal, F. Jalali, W. Karnes, P. Baldi, Deep Learning Achieves near Human-level Polyp Detection in Screening Colonoscopy, GastroenterologyIn revision.

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

[22] A. Aurisano, A. Radovic, D. Rocco, A. Himmel, M. Messier, E. Niner, G. Pawloski, F. Psihas, A. Sousa, P. Vahle, A convolutional neural network neutrino event classifier, Journal of Instrumentation 11 (09) (2016) P09001.
URL http://stacks.iop.org/1748-0221/11/i=09/a=P09001

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

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

[25] C. F. Cadieu, H. Hong, D. L. K. Yamins, N. Pinto, D. Ardila, E. A. Solomon, N. J. Majaj, J. J. DiCarlo, 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

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

[27] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Sackinger, R. Shah, Signature verification using a siamese time delay neural network, International Journal of Pattern Recognition and Artificial Intelligence 7 (4).

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

[29] Y.-N. Jan, L. Y. Jan, Branching out: mechanisms of dendritic arborization, Nat Rev Neurosci 11 (5) (2010) 316–328.
URL http://dx.doi.org/10.1038/nrn2836

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

[31] A. Krizhevsky, V. Nair, G. Hinton, Cifar-10 (canadian institute for advanced research).
URL http://www.cs.toronto.edu/~kriz/cifar.html

[32] F. Chollet, et al., Keras, https://github.com/fchollet/keras (2015).

[33] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, software available from tensorflow.org (2015).
URL http://tensorflow.org/

[34] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Y. W. Teh, M. Titterington (Eds.), Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Vol. 9 of Proceedings of Machine Learning Research, PMLR, Chia Laguna Resort, Sardinia, Italy, 2010, pp. 249–256.
URL http://proceedings.mlr.press/v9/glorot10a.html