Efficiency of Convolutional Neural Networks

Peter Racioppo & Jie He

June 15, 2020

1 Abstract

A convolutional neural network (CNN) is a neural network in which convolutional filters are applied layer-wise to extract hierarchical patterns in input data. CNNs have sparse connectivity, with local neurons acting on local subsets of the input. Neurons in a convolutional layer have shared weights (which define the filter for that layer). Sparsity and parameter sharing reduce the network's memory and computation time requirements and prevent overfitting. Like fully-connected neural networks, CCNs are known to be universal approximators. Their expressiveness and spatial-invariance properties have made them a state-of-the-art algorithm in image recognition and classification, and their efficiency in practice is well-established. However, general theoretical results demonstrating the superior performance and efficiency of CNNs, particularly as compared to dense nets, are as of yet, lacking. In this paper, we review some important theoretical results on the expressiveness and optimization of convolutional networks. We then discuss some results which explore how CNNs take advantage of the properties of typical real-world data. In order to find other promising directions of research, we investigate the connection between deep learning and renormalization group (RG) theory in physics. Finally, we discuss the connections between Toeplitz and circulant matrices, CNNs, and sparse graphs, and suggest a direction for future research.

2 Introduction

The convolution of two real-valued, discrete-time functions f and g is defined as $(f * g)(n) = \sum_{-\infty}^{\infty} f(m)g(n-m)$. However, CNNs typically use the cross-correlation operation, defined as $(f \star g)(n) = \sum_{-\infty}^{\infty} f(m)g(n+m)$. We will follow the convention in referring to this operation as convolution. The convolution operation can be computed using banded Toeplitz matrices, (A Toeplitz matrix has constant diagonals; a banded matrix has its nonzero entries within a band including the main diagonal). For example, given input vectors $w = (w_0...w_s)$ and $v = (v_0...v_D)$ and a filter of length 2, the corresponding Toeplitz matrix T would be such that the diagonal containing the (1,1) element would take values w_0 and the subsequent descending diagonals would take the subsequent values of w. The output of the jth layer can then be represented as $h^{(j)}(x) = \sigma(T^{(j)}h^{(j-1)}(x) - b^{(j)})$, $j \in 1, ..., J$, where σ is an activation function (often the ReLU activation $\sigma(x) = \max(0, x)$). In practice, inputs are often "padded" with zeros to modify the dimension of the output. Another variable that can be controlled is the "stride," which defines how much the convolutional filter moves, i.e. whether elements in the sum are skipped. The "pooling" operation, in which an operation such as $\max(.)$ (max-pooling) is applied to a subset of the input, is commonly performed to introduce small spatial invariances. Pooling is a downsampling operation and results in a lower-dimensional output.

3 Network Design

Convolutional networks have been the state of the art in image recognition since AlexNet (Krizhevsky et al, 2012) won the ImageNet image recognition competition in 2012 and helped start a surge of interest in CNNs. AlexNet, a variant of the CNN architecture designed in (Yann LeCun et al, 1989), used eight layers (deep for the time) with ReLU activations and max pooling. ZFNet (Zeiler and Fergus, 2013) improved this design the following year, using smaller filters applied with smaller strides. VGGNet (Simonyan and Zisserman, 2014) further improved performance by focusing on smaller filters and increasing the network's depth. However, increasing network depth comes at the cost of increasing the number of parameters and tending to induce overfitting. Several approaches to reducing "effective depth" have thus been pursued. GoogLeNet (Szegedy et al, 2014) decreased network parameters by adding "inception layers," mini-networks within the larger network which concatenate their outputs into a single output vector. This design also allows the network to adaptively select the filter sizes in each layer. ResNet (He et al, 2015) employed residual layers, which include a connection directly to the output, to learn a residual value to add to the input. FractalNet (Larsson et al, 2017) employed a method for building deep CNNs by repeated application of an expansion rule. In this publication, the authors argued that the key to the success of deep CNNs "may be the ability to transition, during training, from effectively shallow to deep" architectures. In (Sabour, Frosst, and Hinton, 2017) the authors argued that pooling operations are "a big mistake" as data representation in a CNN "does not take into account important spatial hierarchies between simple and complex objects." The authors introduced an alternative architecture called capsule networks (CapsNets) in which small groups of neurons called capsules are organized in a tree structure. The authors claim that CapsNets significantly improve viewpoint invariance, use fewer parameters, and improve generalization.

4 Expressiveness of ConvNets

Universal approximation results for shallow neural networks were obtained in the late 1980s and early 1990s ((Cybenko, 1989), (Hornik et al, 1989), (Barron, 1993)). These results showed that any continuous function f on a compact subset of R^d can be approximated to arbitrary accuracy by a shallow, fully-connected neural network (FNN) with a sufficiently large number of neurons and sigmoidal activations. These results were extended in (Leshno, et al, 1993) and (Pinkus, 1999), which show that these networks are universal approximators if and only if they have non-polynomial activations. These results were also extended to deep networks ((Hornik et al, 1989), (Chui et al, 1989), (Mhaskar, 1993)). Results in (Barron, 1993) and (Mhaskar, 1993) gave rates of convergence. Later work proved that deep but narrow neural networks and deep belief networks are also universal approximators, and require no more parameters than shallow networks to achieve universal approximation (resp. (Rojas, 2003), (Sutskever and Hinton, 2008)). Work in the late 2010s by e.g. (Telgarsky, 2016), (Eldan and Shamir, 2016), (Yarotsky, 2017), (Shaham, et al, 2018) explored the representational power of deep neural networks and the importance of network depth. (Bolcskei, et al, 2018) and (Petersen and Voigtlaender, 2018) showed that, for some function classes, sparsely connected neural networks can achieve approximation error bounds of the same order as those obtained by fully-connected networks. However, these works do not give any conditions on the patterns of sparse connections.

Universal approximation results were extended to CNNs in the late 2010s (Zhou, 2018), (Yarotsky, 2018), (Petersen and Voigtlaender, 2018). In (Zhou, 2018), the author shows that an arbitrary sequence can be factorized into convolutions of a finite sequence of filter masks. Zhou uses this result to construct the sparsely connected weight and bias vectors and, using a result from (Klusowski and Barron, 2018) on approximation by ramp ridge functions, to construct bounds on the approximation error. These results indicate that

the bounds on the number of free parameters in a CNN are significantly more favorable than the corresponding bounds for a dense network. As a corollary of the error bound result, Zhou proves a universality result for deep CNNs. In particular, any continuous function on a compact subset of R^d can be approximated by a sufficiently deep CNN. Formally, given the hypothesis space $H_J^{w,b} = \{\sum_{k=1}^{d_J} c_k h_k^{(J)}(x) : c \in R^{d_J}\}$ of CNNs, for any $\Omega \subseteq R^d$, $f \in C(\Omega)$, $\exists w, b$ and $\exists f_J^{w,b} \in H_J^{w,b}$ s.t. $\lim_{J\to\infty} ||f - f_J^{w,b}||_{C(\Omega)} = 0$.

(Yarotsky, 2018) introduces a "charge-conserving Convnet" model, and shows that it is a universal approximator for continuous equivariant transformations in SE(2), the group of isometries in a two-dimensional Euclidean space. In (Petersen and Voigtlaender, 2018), the authors prove an equivalence result for FNNs and CNNs. Namely, for any FNN approximating a function $f : R^n \mapsto R$, there exists a CNN with the same number of parameters, up to a constant factor, which approximates a translation equivariant function $g : R^n \mapsto R^n$, with f equal to the first coordinate of g. A converse result also holds.

5 Optimization

Important recent work on optimization of neural networks includes (Lee , et al, 2016), in which the Stable Manifold Theorem from Dynamical Systems Theory is used to show that gradient descent always converges to a minimizer, provided some mild conditions are met by the objective function. (Du, et al, 2019) applied this result to show that, again under mild assumptions on the objective function, activation function, and gradient descent step size, gradient descent applied to a sufficiently large FNN always converges to a global minimum during training. An earlier result in (Brutzkus and Globerson, 2017) shows that, for a convolutional neural net with a single hidden layer, with no overlap of filters and a ReLU activation, and with a Gaussian input distribution, gradient descent is guaranteed to converge to a global minimum in polynomial time during training.

6 Are CNNs better than Fully Connected Networks?

It appears that to quantify the efficiency and invariance properties of CNNs, it may be necessary to consider models of data generation which reflect the invariant or hierarchical properties of real-world data. Some recent work in this direction includes the 2018 paper *Deep Learning and Hierarchical Generative Models* (Mossel, 2018), which shows that, for a particular class of hierarchical generative models, deep learning is both efficient and necessary. In particular, consider the class of Markov models on a directed tree $T =< V, E >$ of degree d. Let each vertex v have a set of labels $L(v)$ and a representation function $R : V \mapsto [q]$, (where $[q]$ is some alphabet of symbols). A transition matrix determines the transition probabilities along the directed edges between any two representations $R(u)$ and $R(v)$. The classification problem in this case is to infer the labels of T from the representation functions and some small subset of data with known labels. Data is generated from the root of the tree, cascading down to the leaves. In the simplest model, we assume each letter of $R(w)$ is copied with probability λ from its parent and is otherwise chosen at random, and that each letter is sampled independently. The authors also consider a more sophisticated version of this model which dispenses with the assumption of independence. In particular, "the wiring between different features at different levels is given by some known permutation that is level dependent." This level-dependent sparsity pattern is qualitatively similar to a convolutional network. The authors show that:
1.) There exists an efficient deep learning algorithm that reconstructs the tree T, and
2.) The probability that a shallow algorithm labels a random leaf in T correctly is no better than random.

Another contribution to the study of hierarchical generative models appeared in 2018 in *A Provably Correct Algorithm for Deep Learning that Actually Works* (Malach and Shalev-Shwartz, 2018), which describes a method for training deep CNNs layer by layer. In this paper, the authors propose an algorithm based on a hierarchical generative model for natural images which applies gradient training on a two-layer network followed by a clustering step. The algorithm consists of the following steps:
(1) Run a clustering algorithm on the input image patches.
(2) Run a convolution operation on the cluster centroids, followed by a ReLU activation and pooling.
(3) Run a 1×1 convolution, followed by a linear layer, for classification. Train each such two-layer network using gradient descent.
(4) Throw away the linear layer. Use the Conv-Pool-ReLU-Conv net to generate input to the next layer.
(5) After repeating this process k times, feed the output of the (Conv-Pool-ReLU-Conv) $\times k$ network to a classifier.
The clustering step of the algorithm identifies geometrically similar regions, using e.g. the Euclidean distance between two patches. The CNN identifies "semantic" similarity between patches: the similarity between the distribution of occurrences of each patch across the image. The gradient of the output of the (first) convolutional layer depends only on the "mean image" of the semantic classes. The authors show that it then follows that the two-layer ConvNet learns an embedding of the observed patches into a space where patches from the same semantic class are close together, while patches from different classes are far apart, that is, "The gradient descent algorithm maps semantically similar patches to geometrically similar vectors, allowing the clustering of the next iteration to perform clustering based again on the simple geometrical distance." The authors prove convergence of their algorithm for data produced by their generative model, and show that applying the algorithm on real-world data produces results comparable to standard CNNs.

Another class of results concerns wavelet networks, which are inspired by translation-invariant representations based on Fourier transforms. Image classification algorithms must be invariant to translation and rotation, and continuous to deformation. In other words, a deformed image should have a small distance from the original in the representation space. Fourier transforms can be used to construct algorithms which are invariant to translations, but Fourier transforms are unstable to deformations. Wavelets are localized waveforms, which are stable to deformation, but they lack the translation invariance of Fourier transforms. Building invariant representations from wavelet coefficients requires introducing non-linear operators, which leads to a convolution-like network architecture. In *Group Invariant Scattering* (Mallat, 2012), the authors construct a windowed scattering transform, a local integration of products of wavelet moduli. These wavelet-based filters are both translation-invariant and Lipschitz-continuous to diffeomorphisms. Wavelet Scattering Networks, introduced in (Zhang, Benveniste, 1992), construct a network from wavelet generators, which can be thought of as taking the place of neurons as the basic computational unit in the network. These networks achieve translation-invariance by an average pooling of the wavelet modulus coefficients. Here, modulus nonlinearity is chosen for stability. Scattering operators are localized operations, so the values (and possibly sparsity) of network parameters are related to the geometry of the input image. In *Invariant Scattering Convolution Networks* (Bruna, Mallat, 2012), the authors show that the modulus of the scattering transform is nearly zero everywhere but a subset of paths through the network. Reducing computations to these paths defines a convolutional network with much fewer internal and output coefficients. Thus, the mathematical properties of scattering operators given rise to a sparsity pattern analogous to that which emerges in convolutional networks.

In *Dynamic Routing Between Capsules* (Sabour, Frosst, and Hinton, 2017), the authors claim that CNNs "translate knowledge about good weight values acquired at one position in an image to other

positions" but do not account for "important spatial hierarchies between simple and complex objects." The authors thus suggest dispensing with pooling operations and replacing the scalar-output feature detectors of CNNs with vector-outputs by groups of neurons termed "capsules." The length of the activity vector is used to represent the probability of the existence of a certain object, edge, etc. in the image, while its orientation represents certain properties of the object. The use of vectors allows for the use of a dynamic routing algorithm. Max-pooling is replaced with routing-by-agreement, in which each capsule sends its output to capsules in the next layer using a softmax-like algorithm, which is input a log-likelihood proportional to the dot product of the capsule's activation vectors. A non-linear "squashing" function is used to shrink short vectors to near zero length, in order to encourage sparsity. The authors claim that CapsNets significantly improve viewpoint invariance, use fewer parameters, and improve generalization. In the paper, a CapsNet trained on MNIST achieves state-of-the-art performance and outperforms CNNs at recognizing highly overlapping digits. Studying how sparsity arises in architectures of this type (i.e. with dynamic routing and vector representation of inputs) may shed light on sparsity patterns in traditional CNNs.

7 Stability of CNNs

Despite the recent success of CNN and its related architectures, the theoretical analysis of such network is lacking. Many advancements, such as ResNet(He, 2016) and densenet(Huang, 2017), are empirical by nature. There is a growing gap between theory and practice, and hence it is crucial to understand the nature of CNN in order to further utilize its power, instead of using engineering tricks. Recently, the work of Bietti and Mallat(Bietti, 2017; Bietti, 2019) has shed some lights into the complex structures of the CNN by the kernel method. Using the kernel method, the authors obtained important theoretical confirmations of the stability of CNN.

Kernel method can be a useful tool because with a positive definite kernel, people can study a network structure in the Reproducing Kernel Hilbert Space(RKHS) associated with the kernel. Another benefit of the kernel method is that the CNN can be studied independently of the data. Previous works such as (Cho, 09) attempted to construct hierarchical kernels for multi-layer fully-connected networks. However, such construction didn't utilize the power of hierarchical structure. The convolutional kernel network (CKN) constructed by Bietti, on the other hand, is a hierarchical kernel that can exploit the local structures of the data. The CKN has three main components, the patch extraction operator, P, the kernel mapping operator M, and the pooling operator A. Suppose the input signal has dimension $\mathbf{x}_{i,j} \in R^p$, then P will gather all input signals of a set S around coordinate i, j such that $P\mathbf{x}_{i,j} = (\mathbf{x}_{i+k,j+l})_{(k,l)\in S}$. After the signals are mapped into a higher dimension $R^{p|S|}$, the kernel mapping operator M maps the data into the RKHS associated with a positive definite kernel κ. Such operation can be realized by applying a feature map φ related to the kernel κ: $MP\mathbf{x}_{i,j} = \varphi(P\mathbf{x}_{i,j}) \in \mathcal{H}$. Once the data are in the RKHS, the pooling operator A can be applied. The pooling operator allows the network to obtain some shift-invariance by globally or locally pooling the signal. Consequently, one convolutional layer can be formulated as $AMP\mathbf{x}$, and multilayer convolutional structures can be obtained by stacking operators in a sequence, namely $\mathbf{x}^n = A_n M_n P_n ... A_1 M_1 P_1 \mathbf{x}^0$.

A final result regarding the stability of CKN (Theorem 2.7 of Bietti, 2019): Consider C^1 diffeomorphism: a function $\tau : R^d \to R^d$ such that $L_\tau x(u) = x(u - \tau(u))$. Then, assume the Jacobian $||\nabla\tau||_\infty \leq 1/2$

$$||\Phi_n(L_\tau x) - \Phi_n(x)|| \leq (C_1(1+n)||\nabla\tau||_\infty + \frac{C_2}{\sigma_n}||\tau||_\infty)||x||$$

where n is number of layers and σ_n is the scale parameter that usually scales exponentially with network depth. Based on the theorem, in order to improve the stability of a CKN in pratice, it's recommended to (1) use smallest patches at each layer, (2) perform pooling and downsampling at a

factor smaller than the patch size, and (3) increase the depth to a desired level of translation invariance.

It should be noted that the CKN constructed by the three kernel operators does not function exactly like a CNN, but the author showed that the results can be generalized to a larger class of CNN with smooth activation functions. It's possible to include CNN with non-smooth activation functions, but there are more restrictions to be applied. Although the stability theorem is limited in application, it's one of the first theoretical confirmations of properties relating to neural networks with deep convolutional structures.

8 Linking CNNs to Physics

While many strive to establish the theoretical foundations of the CNN, Metha and Schwab viewed such neural network from a different point of view. In their work (Metha, 2014), the authors mentioned an exact mapping between the variational renormalization group (RG) and deep learning.

Formulated by Kadanoff (Kadanoff, 1976), the RG theory describes an iterative coarse-graining procedure of solving large-scale physical systems. If the data follows the 2d Ising lattice model, the RG attempts to represent the state of spins by fewer "hidden spins" by integrating out local features, a procedure similar to the patch extraction and downsampling operators of a CNN.

While the authors proved their point in the original paper by using a deep Boltzmann machine, we believe a simple fully-connected neural network may also learn to capture local features of the data. We prepared 10,000 training samples generated from a 10 by 10 2D Ising lattice using Gibbs sampling with inverse temperature 0.4 and an additional 2,000 testing samples. The network is a two-layer, fully-connected autoencoder built by Tensorflow. It contains 32 neurons in the hidden layer. After 200 iterations, the validation loss stabilized to approximately 0.385. As shown in Fig. 1, the network can reconstruct the rough shape of the original image. The weights of the network (Fig. 2) tend to concentrate on local areas of the image, demonstrating a behavior similar to the filters of a CNN. Indeed, an image can be viewed as a lattice of pixels, and a hidden configuration of the lattice network defines the meaning of an image to humans. The successes of CNNs suggest that there may be a deeper relationship between human vision and physics.

9 Some facts about Toeplitz and Circulant Matrices

Recall that a Toeplitz matrix is a matrix with constant diagonals. A circulant matrix is a Toeplitz matrix in which every row (column) is copy of the preceding row (column), shifted by one element. The adjacency matrix of a 2D lattice with periodic boundary conditions is a circulant matrix.
The following facts, from (R. Gray, 2005) may be useful:
(1) The product of circulant matrices is circulant.
(2) The inverse of a circulant matrix is circulant.
(3) The product of Toeplitz matrices is not Toeplitz. However, the product of properly-defined n-dimensional Toeplitz matrices converges to a Toeplitz matrix in the limit that n goes to infinity.
(4) A sequence of properly-defined n-dimensional Toeplitz matrices converges to a sequence of properly-defined circulant matrices in the limit that n goes to infinity.

10 Convergence of Linear Networks

Consider a CNN whose output at the jth layer is $h^{(j)}(x) = \sigma(T^{(j)}h^{(j-1)}(x) - b^{(j)})$, $j \in 1, ..., J$, where σ is an activation function and T is a Toeplitz matrix. The output of an m-layer network is then $\hat{y} = \sigma(T_m\sigma(T_{m-1}(...\sigma(T_1x - b_1)) - b_{m-1}) - b_m)$, where $x \in R^n$ is the input at the first layer. In

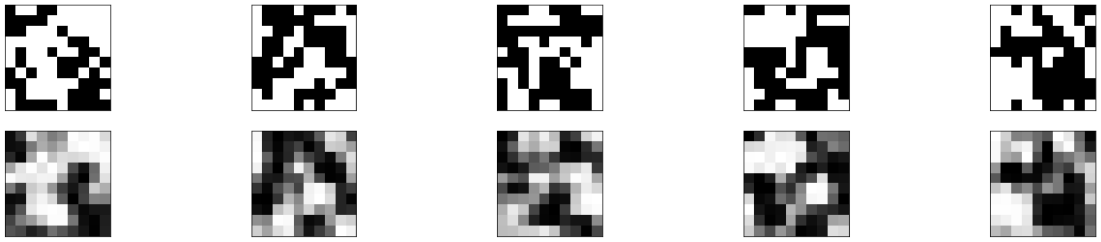

Figure 1: Reconstructed data. Top: original. Bottom: reconstructed

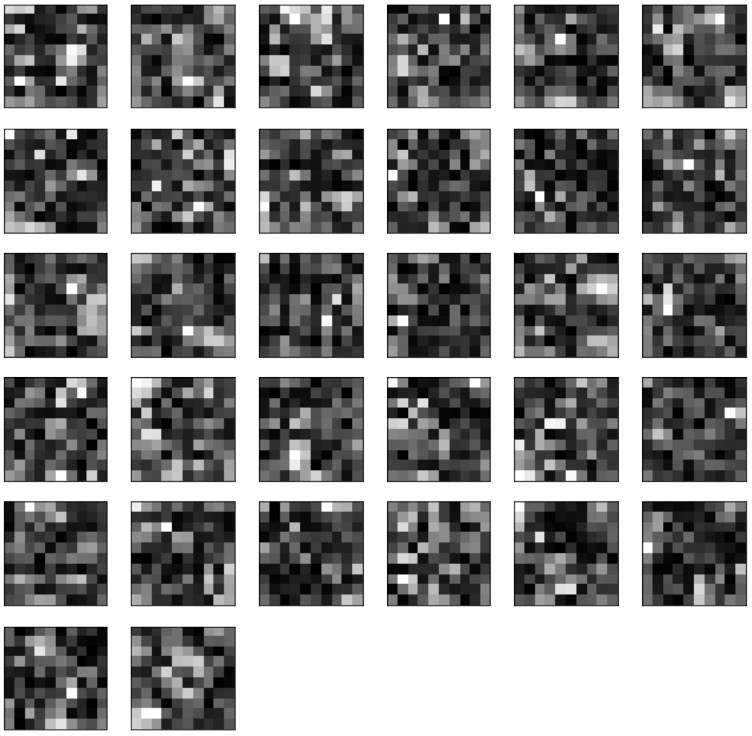

Figure 2: Normalized weights of neurons. White pixel indicates high weight value.

a linear network, that is a network in which $\sigma = I$, we have that $\hat{y} = Tx - Q$, where $T = \prod_{i=1}^{m} T_{m+1-i}$ and $Q = -\sum_{j=1}^{m} \prod_{i=1}^{m-j} T_{m+1-i} b_j$. Letting $X = [x^T, -I]^T$ and $W_i = [T_i, Q_i]$, we can rewrite the network output as $\hat{y} = WX$.

For classification, we will generally need to perform an additional operation, say $\Psi(\hat{y}) = \text{sign}(\text{softmax}(a^T \hat{y} b) - c)$, where a,b, and c are weighting vectors, needed to decrease the dimension of $\hat{y}$. (Typically, $\hat{y}$ is passed through a fully-connected layer before application of the softmax function.) Let us instead consider the related problem of minimizing the distance between the network output $\hat{y}$ and a target image y. Let us suppose we are testing the network on l input images $x_i \in \{x_1, ..., x_l\}$. Let $Z \in R^{n \times l}$ be a matrix whose ith column is X_i and let $Y \in R^{n \times l}$ be a matrix of l target images. (We're here considering the case that the output of the network is the same dimension as the input.) We can define a least-squares loss $||Y - WZ||_F$, where $||.||_F$ is the Frobenius norm $||A||_F = \sqrt{\text{tr}(A^* A)}$. The unique W that minimizes this loss is $\hat{W} = YZ^\dagger = Y(Z^T Z)^{-1} Z^T$. Suppose that Z is comprised of n shifted copies of the same input image and Y is comprised of n shifted copies of the same target image. Then Z and Y are circulant. Any square circulant matrix is invertible, so $Z^\dagger = Z^{-1}$ is circulant. Finally, $\hat{W}$ is circulant (since the product of two circulant matrices is circulant), so T is circulant (and hence Toeplitz). We can then reconstruct T with a product of circulant matrices T_i. Thus, in this special circumstance, the global optimum of the fully connected linear neural network is a CNN.

11 Inference on Graphs

The matrix T in the foregoing discussion can be seen as the stochastic matrix which represents the transition probabilities of a Markov chain over the pixels in a given image. We hypothesize that learning T is like learning the stochastic matrix which describes the generative model of the training data. We can recover this stochastic matrix by averaging the correlation of individual pixels over many training images. In other words, for each training image, we loop through all pairs of vertices. If the vertices are in the same state, we increase the weight of the edge between them in the matrix. The resulting matrix will converge to a sparse matrix provided the correlations between pixels are not all equal. For example, a matrix of this kind constructed from training data generated by an Ising model on a 2D lattice will tend to converge to the circulant matrix corresponding to the adjacency matrix of a 2D lattice (Fig. 3).

We repeated this process for 200 images of the digit "5" from the MNIST dataset (Fig 4.) To test that the correct graph structure was learned, we normalized the matrix to form a stochastic matrix, and added a small, nonzero probability of transitioning between states not connected by edges (as in the PageRank algorithm). The stationary distribution can be thought of as the percent of time spent at each vertex in an infinitely long random walk on the stochastic matrix. Plotting the result produced an image visually similar to the average of the 200 images (Fig. 4). We can induce sparsity in the stochastic matrix by removing edges whose weights fall below some threshold. Figure 5 shows the result of removing edges with weights below 10% and 35% of the maximum, which resulted, respectively, in matrices with 4.74% and 0.46% of the edges in a fully-connected graph. Even with only 1 in 200 of the possible edges included in the graph, the image is still clearly a "5."

Now, if we can interpret T as a Markov chain, then Tx represents the the distribution of the pixel intensities after the input image makes a single step in the Markov chain. In particular, if x is nearly equal to the stationary distribution, then Tx will be nearly equal to x. If the target image y is equal to the stationary distribution of the Markov chain for the generative model, then the set of input images with low loss is the set of images which are nearly equal to the stationary distribution.

This suggests the following method to study

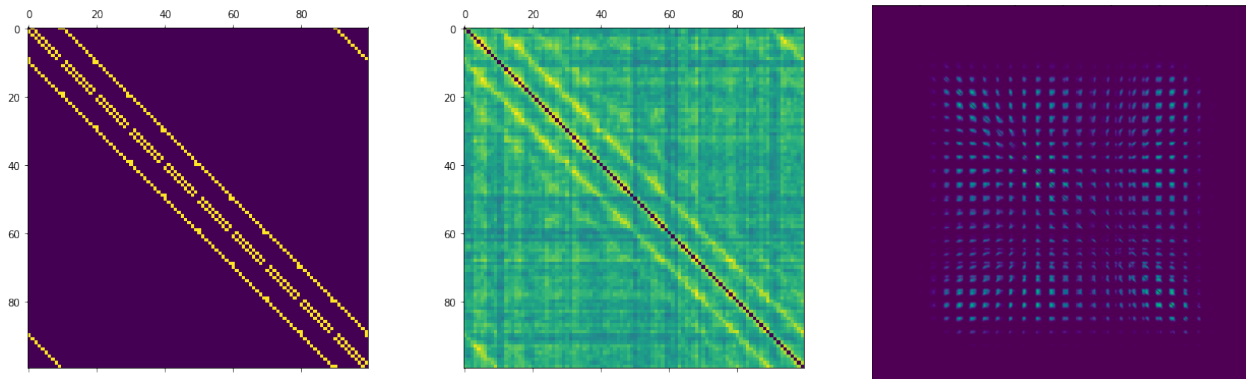

Figure 3: **Left**: The (circulant) adjacency matrix of a 2D lattice; **Middle**: The adjacency matrix obtained by averaging over 200 samples from an Ising model with inverse temperature 0.5. **Right**: The adjacency matrix obtained by averaging over 200 samples of MNIST ″5″ digits.

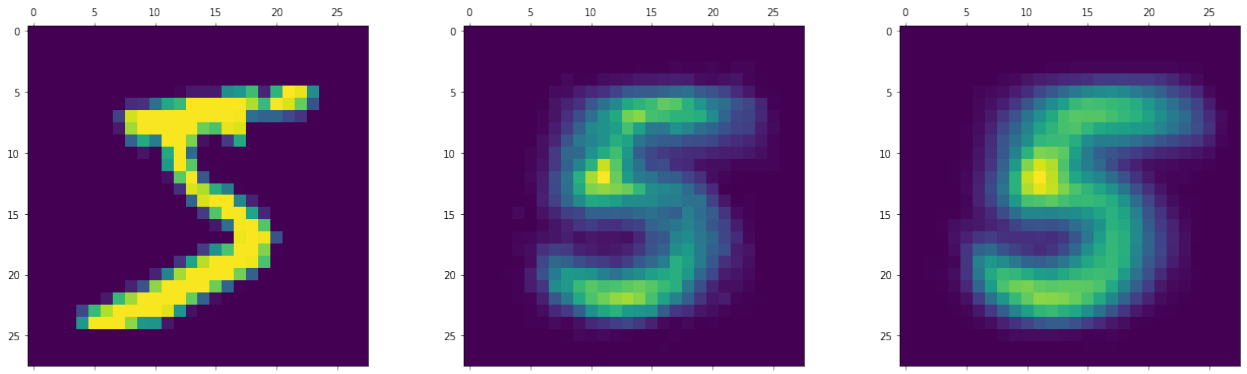

Figure 4: **Left**: an input MNIST image of the digit ″5″; **Middle**: Stationary distribution of the Markov chain corresponding to the stochastic matrix produced by averaging the pixel correlations over 200 training images; **Right**: Average of 200 training images.

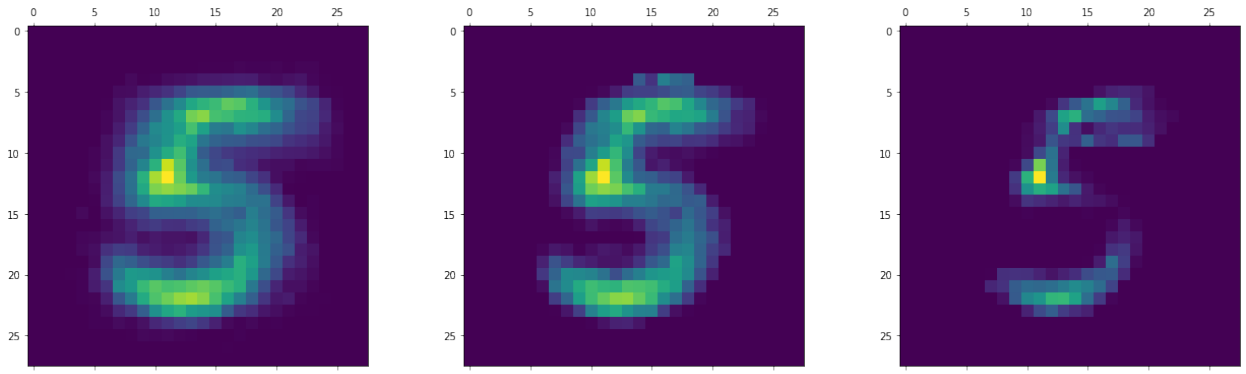

Figure 5: Stationary distributions of average adjacency matrices of increasing sparsity. **Left**: Using the original adjacency matrix, which has 30.01% active edges; **Middle**: After removing all edges with weights less than 10% of the maximum, which produces an adjacency matrix with 4.74 % active edges; **Right**: After removing all edges with weights less than 35% of the maximum, which produces an adjacency matrix with 0.46 % active edges.

the sparse structure of CNNs:
(1) Construct a CNN whose expected testing loss after an infinite number of training steps has the same global minimum as the expectation of $||y - \hat{y}||$, where y is the stationary distribution of the Markov chain obtained by averaging the adjacency matrices of an infinite number of training images and $\hat{y}$ is the output of the last network layer.
(2) Show that the optimal W obtained by training a neural network with this loss is equal to the stochastic matrix obtained by averaging the adjacency matrices of an infinite number of training images.
(3) Determine the sparsity pattern of the W obtained by averaging the adjacency matrices of an infinite number of training images.

12 References

A. Krizhevsky, I. Sutskever, G. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." ICLR 2015.

Y. LeCun, B. Boser, J. Denker, D. Henderson, E. Howard, W. Hubbard, L. Jackel. "Backpropagation Applied to Handwritten Zip Code Recognition." Neural Computation. MIT Press - Journals. 1989.

Y. LeCun, L. Bottou, Y. Bengio, P. Haffner. "Gradient-based learning applied to document recognition." 1998. Proceedings of the IEEE. 86 (11): 2278–2324.

Y. LeCun and Y. Bengio. "Convolutional networks for images, speech, and time series." The handbook of brain theory and neural networks, 3361(10), 1995.

Y. LeCun, Y. Bengio, G. Hinton. Deep learning. Nature, 521(7553):436–444, May 2015.

M. Zeiler, R. Fergus. "Visualizing and Understanding Convolutional Networks." 2013. arXiv:1311.2901.

K. Simonyan, A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." 4 Sept. 2014. arXiv:1409.1556.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich. "Going Deeper with Convolutions" 17 Sept. 2014. arXiv:1409.4842.

K. He, X. Zhang, S. Ren, J. Sun. "Deep Residual Learning for Image Recognition." 10 Dec. 2015. arXiv:1512.03385.

G. Larsson, M. Maire, G. Shakhnarovich. "FractalNet: Ultra-Deep Neural Networks without Residuals." arXiv:1605.07648.

S. Sabour, N. Frosst, G. Hinton. "Dynamic Routing Between Capsules." 26 Oct. 2017. arXiv:1710.09829.

G. Cybenko. "Approximations by superpositions of sigmoidal func- tions." 1989. Mathematics of Control, Signals, and Systems 2: 303–314.

K. Hornik, M. Stinchcombe, H. White. "Multilayer feedforward net- works are universal approximators." 1989. Neural networks 2: 359-366.

A. Barron. "Universal approximation bounds for superpositions of a sigmoidal function." 1993. IEEE Transactions on Information Theory 39: 930- 945.

M. Leshno, Y. Lin, A. Pinkus, S. Schocken. "Multilayer feedforward networks with a non-polynomial activation function can approximate any function." 1993. Neural Networks 6: 861–867.

A. Pinkus. "Approximation theory of the MLP model in neural networks." 1999. Acta Numerica 8: 143–195.

H. Mhaskar. "Approximation properties of a multilayered feedfor- ward artificial neural network." 1993. Advances in Computational Mathematics 1: 61–80.

C. Chui, X. Li, H. Mhaskar. "Limitations of the approximation capabilities of neural networks with one hidden layer." 1996. Advances in Computational Mathematics 5: 233–243.

R. Rojas. "Networks of width one are universal classifiers." 2003. In International Joint Conference on Neural Networks, volume 4, pages 3124–3127.

I. Sutskever, G. Hinton. "Deep, narrow sigmoid belief networks are universal approximators." 2008. Neural Computation, 20(11), 2629–2636.

M. Telgarsky. "Benefits of depth in neural networks." 2016. 29th Annual Conference on Learning Theory PMLR 49:1517–1539.

R. Eldan, O. Shamir. "The power of depth for feedforward neural networks." 2016. COLT: 907-940.

D. Yarotsky. "Error bounds for approximations with deep ReLU networks." 2017. Neural Networks 94: 103–114.

U. Shaham, A. Cloninger, R. Coifman. "Provable approximation properties for deep neural networks." 2018. Applied and Computational Har- monic Analysis 44: 537–557.

H. Bolcskei, P. Grohs, G. Kutyniok, P. Petersen. "Optimal approximation with sparsely connected deep neural networks." 2018. arXiv:1705.01714v4.

P. Petersen, V. Voigtlaender. "Optimal approximation of piecewise smooth functions using deep ReLU neural networks." 2018. arXiv:1709.05289v4.

D.X. Zhou. "Universality of Deep Convolutional Neural Networks." arXiv:1805.10769. 28 May 2018.

D. Yarotsky. "Universal approximations of invariant maps by neural networks." arXiv:1804.10306. 26 April 2018.

P. Petersen and F. Voigtlaender. "Equivalence of approximation by convolutional neural networks and fully-connected networks." arXiv:1809.00973. 4 Sept. 2018.

J. Klusowski , A. Barron. "Uniform approximation by neural networks activated by first and second order ridge splines." 2018. arXiv:1607.07819v2.

J. Lee , M. Simchowitz , M. Jordan, B. Recht. "Gradient Descent Only Converges to Minimizers." 2016. JMLR: Workshop and Conference Proceedings vol 49:1–12, 2016.

S. Du, J. Lee, H. Li, L. Wang, X. Zhai. "Gradient Descent Finds Global Minima of Deep Neural Networks." 2019. arXiv:1811.03804.

A. Brutzkus, A. Globerson. "Globally Optimal Gradient Descent for a ConvNet with Gaussian Inputs." arXiv:1702.07966. 26 Feb 2017.

Q. Nguyen, M. Hein. "Optimization Landscape and Expressivity of Deep CNNs." 30 Oct 2017. arXiv:1710.10928.

E. Malach, S. Shalev-Shwartz. "A Provably Correct Algorithm for Deep Learning that Actually Works." School of Computer Science, The Hebrew University, Israel. arXiv:1803.09522v2 [cs.LG]. 24 June 2018.

E. Mossel. "Deep Learning and Hierarchal Generative Models." 2016. arXiv:1612.09057.

S. Mallat. "Group Invariant Scattering." 2011. arXiv:1101.2286.

Q. Zhang, A. Benveniste. "Wavelet Networks," IEEE Transactions on Neural Networks, Vol. 3, No. 6, 1992, pp. 889-898. doi:10.1109/72.165591.

J. Bruna, S. Mallat. "Invariant Scattering Convolution Networks." 2012. arXiv:1203.1513v2.

A. Bietti, J. Mairal. "Group Invariance, Stability to Deformations, and Complexity of Deep Convolutional Representations." arXiv:1706.03078. 2017.

He, Kaiming Zhang, Xiangyu Ren, Shaoqing Sun, Jian. (2016). Deep Residual Learning for Image Recognition. 770-778. 10.1109/CVPR.2016.90.

Huang, Gao Liu, Zhuang van der Maaten, Laurens Weinberger, Kilian. (2017). Densely Connected Convolutional Networks. 10.1109/CVPR.2017.243.

Alberto Bietti. Foundations of deep convolutional models through kernel methods. Signal and Image Processing. Université Grenoble Alpes, 2019. English.

Y. Cho and L. K. Saul. Kernel methods for deep learning. In Advances in Neural Information Processing Systems (NIPS), 2009.

P. Mehta, D. J. Schwab, An exact mapping between the variational renormalization group and deep learning, arXiv preprint arXiv:1410.3831 (2014).

L. P. Kadanoff, A. Houghton, M. C. Yalabik, Variational approximations for renormalization group transformations, Journal of Statistical Physics 14 (1976) 171–203.

R. Gray. "Toeplitz and Circulant Matrices: A review." Stanford University. 2005. Commun. Inf. Theory.