On Random Weights and Unsupervised Feature Learning

Andrew M. Saxe ASAXE@CS.STANFORD.EDU
Pang Wei Koh PANGWEI@CS.STANFORD.EDU
Zhenghao Chen ZHENGHAO@CS.STANFORD.EDU
Maneesh Bhand MBHAND@CS.STANFORD.EDU
Bipin Suresh BIPINS@CS.STANFORD.EDU
Andrew Y. Ng ANG@CS.STANFORD.EDU
Computer Science Department, Stanford University, Stanford, CA 94305, USA

Abstract

Recently two anomalous results in the literature have shown that certain feature learning architectures can yield useful features for object recognition tasks even with untrained, random weights. In this paper we pose the question: why do random weights sometimes do so well? Our answer is that certain convolutional pooling architectures can be inherently frequency selective and translation invariant, even with random weights. Based on this we demonstrate the viability of extremely fast architecture search by using random weights to evaluate candidate architectures, thereby sidestepping the time-consuming learning process. We then show that a surprising fraction of the performance of certain state-of-the-art methods can be attributed to the architecture alone.

1. Introduction

Recently two anomalous results in the literature have shown that certain feature learning architectures with random, untrained weights can do very well on object recognition tasks. In particular, Jarrett et al. (2009) found that features from a one-layer convolutional pooling architecture with completely random filters, when passed to a linear classifier, could achieve an average recognition rate of 53% on Caltech101, while unsupervised pretraining and discriminative finetuning of the filters improved performance only modestly to 54.2%. This surprising finding has also been noted by Pinto et al. (2009), who evaluated thousands of convolutional pooling architectures on a number of object recognition tasks and again found that random weights performed only slightly worse than pretrained weights (James DiCarlo, personal communication; also see Pinto & Cox, 2011).

This leads us to two questions: (1) Why do random weights sometimes do so well? and (2) Given the remarkable performance of architectures with random weights, what is the contribution of unsupervised pretraining and discriminative finetuning?

We start by studying the basis of the good performance of these convolutional pooling object recognition systems. Section 2 gives two theorems which show that convolutional pooling architectures can be inherently frequency selective and translation invariant, even when initialized with random weights. We argue that these properties underlie their performance.

In answer to the second question, we show in Section 3 that, for a fixed architecture, unsupervised pretraining and discriminative finetuning improve classification performance relative to untrained random weights. However, the performance improvement can be modest and sometimes smaller than the performance differences due to architectural parameters. These results highlight the importance of searching for the most suitable architecture for a given classification task, an approach taken in, for example, (Jarrett et al., 2009). Our work thus motivates a fast heuristic for architecture search given in Section 4, based on our observed empirical correlation between the random-weight performance and the pretrained/finetuned performance of any given network architecture. This method allows us to sidestep the time-consuming learning process by evaluating candidate architectures using random weights as a proxy for learned weights, yielding an order-of-magnitude speed-up.

Appearing in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011. Copyright 2011 by the author(s)/owner(s).

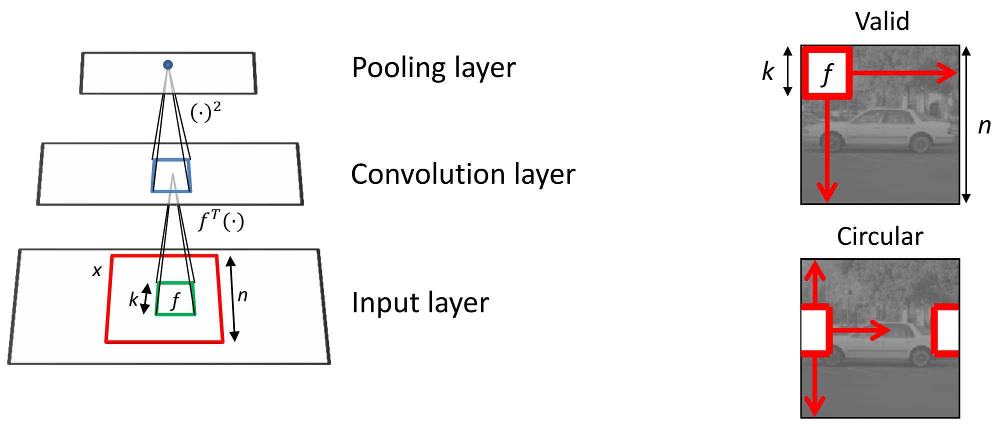


Figure 1. Left: Convolutional square pooling architecture. In our notation x is the portion of the input layer seen by a particular pooling unit (shown in red), and f is the convolutional filter applied in the first layer (shown in green). Right: Valid convolution applies the filter only at locations where f fits entirely; circular convolution applies f at every position, and permits f to wrap around.

We conclude by showing that a surprising fraction of performance can be contributed by the architecture alone. In particular, we present a single-layer convolutional square-pooling architecture with random weights that achieves competitive results on the NORB dataset, without any feature learning. This demonstrates that a sizeable component of a system's performance can come from the intrinsic properties of the architecture, and not from the unsupervised learning algorithm. We suggest distinguishing the contributions of architectures from those of learning systems by reporting random weight performance.

2. Analytical Characterization of the Optimal Input

Why might random weights perform so well? As only some architectures yield good performance with random weights, a reasonable hypothesis is that particular architectures can naturally compute features well-suited to object recognition tasks. Indeed, Jarrett et al. (2009) numerically computed the optimal input to each neuron using gradient descent in the input space, and found that the optimal inputs were often sinusoidal and insensitive to translations. To formalize this important intuition about what features of the input these random-weight architectures might compute, we analytically characterize the optimal input to each neuron for the case of convolutional square-pooling architectures. The convolutional square-pooling architecture can be envisioned as a two layer neural network (Fig. 1). In the first "convolution" layer, a bank of filters is applied at each position in the input image. In the second "pooling" layer, neighboring filter responses are combined together by squaring and then summing them. Intuitively, this architecture incorporates both selectivity for a specific feature of the input due to the convolution stage, and robustness to small translations of the input due to the pooling stage.

In response to a single image $I \in \mathbb{R}^{l \times l}$, the convolutional square-pooling architecture will generate many pooling unit outputs. Here we consider a single pooling unit which views a restricted subregion $x \in \mathbb{R}^{n \times n}, n \leq l$ of the original input image, as shown in Fig. 1. The activation of this pooling unit $p_v(x)$ is calculated by convolving a filter $f \in \mathbb{R}^{k \times k}, k \leq n$ with x using "valid" convolution. Valid convolution means that f is applied only at each position inside x such that f lies entirely within x (Fig. 1, top right). This produces a convolution layer of size $n-k+1 \times n-k+1$ which feeds into the pooling layer. The final activation of the pooling unit is the sum of the squares of the elements in the convolution layer. This transformation from the input x to the activation can be written as $p_v(x) = \sum_{i=1}^{n-k+1} \sum_{j=1}^{n-k+1} (f *_v x)_{ij}^2$ where $*_v$ denotes the "valid" convolution operation.

To understand the sort of input features preferred by this architecture, it would be useful to know the set of inputs that maximally activate the pooling unit. For example, intuitively, this architecture should exhibit some translation invariance due to the pooling operation. This would reveal itself as a family of optimal inputs, each differing only by a translation. While the translation invariance of this architecture is simple enough to grasp intuitively, one might also expect that the *selectivity* of the architecture will be approximately similar to that of the filter f used in the convolution. That is, if the filter f is highly frequency selective, we might expect that the optimal input would be close to

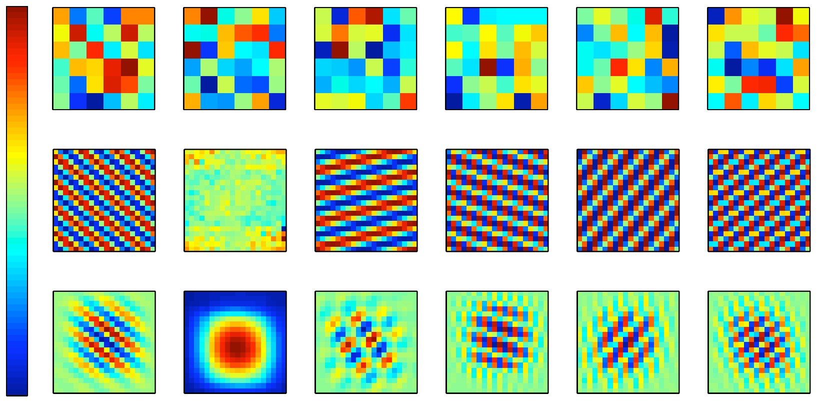


Figure 2. Six random filters and associated optimal inputs. Top: Randomly sampled 6×6 filter used in the convolution. Middle: Optimal 20×20 input for circular convolution. Bottom: Optimal 20×20 input for valid convolution.

a sinusoid at the maximal frequency in f, and if the filter f were diffuse or random, we might think that the optimal input would be diffuse or random. Our analysis shows that this latter intuition is in fact false; regardless of the filter f, a near optimal input will be a sinusoid at the maximal frequency present in the filter.

To achieve this result, we first make one modification to the convolutional square-pooling architecture to permit an analytical solution: rather than treating the case of "valid" convolution, we will instead consider "circular" convolution. Circular convolution applies the filter at every position in x, and allows the filter to "wrap around" in cases where it does not lie entirely within x, as depicted in Fig. 1, bottom right. Both valid and circular convolution produce identical responses in the interior of the input region but differ at the edges, where circular convolution is clearly less natural; however with it we will be able to compute the optimal input exactly, and subsequently show that the optimal input for the case of circular convolution is near-optimal for the case of valid convolution.

Theorem 2.1 *Let $f \in \mathbb{R}^{k \times k}$ and an input dimension $n \geq k$ be given, and assume the technical condition that the zero frequency (DC) component of f is not the maximal frequency in f.*[1] *Then there exists vertical and horizontal frequencies v and h such that a norm-one input that maximally activates a circular convolution square-pooling unit p_c is a simple sinusoid,*

$$x^{opt}[m,s] = \frac{\sqrt{2}}{n} \cos\left(\frac{2\pi mv}{n} + \frac{2\pi sh}{n} + \phi\right), \qquad (1)$$

where ϕ is an unspecified phase. In particular, let $\tilde{f}$ be formed by zero-padding f to size $n \times n$. Then (v,h) is the frequency pair with amplitude equal to the maximum amplitude of any frequency in $\tilde{f}$.

[1]For notational simplicity we omit the case where the zero frequency component is maximal, for which the result holds with a slightly different leading factor. In this case the optimal input has $v = h = 0$ which yields a constant input across space.

This result reveals two key features of the circular convolution, square-pooling architecture.

1. The frequency of the optimal input is the frequency of maximum magnitude in the filter f. Hence the architecture is *frequency selective.*

2. The phase ϕ is unspecified, and hence the architecture is *translation invariant.*

Even if the filter f contains a number of frequencies of moderate magnitude, such as might occur in a random filter, the best input will still come from the maximum magnitude frequency. Example filters and associated optimal inputs are shown in Fig. 2.

Informally, Theorem 2.1 is proved as follows. The maximization problem is converted to the Fourier domain, where the convolution operation becomes multiplication with the spectrum of the filter. Squaring the spectrum of the filter boosts higher amplitude frequencies more than lower amplitude frequencies, and hence the optimal input places all of its energy in the largest amplitude frequency present in the original filter. Inverting this optimal input amplitude from the Fourier domain back to the spatial domain yields a single sinusoid at this maximal frequency of unspecified phase. Full proof details are given in Appendix A.

2.1. Sinusoids are Near-Optimal for Valid Convolution

The above analysis computes the optimal input for circular convolution; object recognition systems, however, typically use valid convolution. Because circular convolution and valid convolution differ only near the edge of the input region (see Fig. 2), we might expect that the optimal input for one would be quite good for the other. We make this rigorous below.

Theorem 2.2 *Let $p_v(x)$ be the activation of a single pooling unit in a valid convolution square-pooling architecture, in response to an input $x \in \mathbb{R}^{n\times n}$, and suppose that the zero frequency (DC) component of f is not maximal. Then there exists a sinusoidal input $x^{\sin}$ that is near optimal for $p_v(x)$. More formally, there exist v, h, and ϕ such that if*

$$x^{\sin}[m,s] = \frac{\sqrt{2}}{n}\cos\left(\frac{2\pi m v}{n} + \frac{2\pi s h}{n} + \phi\right), \qquad (2)$$

then for all x ($||x|| = 1$), we have that

$$p_v\left(x^{\sin}\right) \geq p_v\left(x\right) - Kn^{-1}, \qquad (3)$$

where $K = 4k^3||f||^2$ does not depend on n.

Theorem 2.2 is proved by showing that the circular and valid convolutions are good approximations for each other. Thus, an optimal input for the circular convolution, specifically a single sinusoid (from Theorem 2.1), is also near-optimal for the valid convolution. The full proof is given in Appendix B.

This result shows that valid convolutional square-pooling architectures will respond near-optimally to sinusoids at the maximum frequency present in the filter. These results hold for arbitrary filters, and hence they hold for random filters. We suggest that this frequency selectivity and translation invariance contributes to the good performance reported by Jarrett et al. (2009) and Pinto et al. (2009), as these two properties are ingredients well-known to produce high-performing object recognition systems. Although many systems attain frequency selectivity via oriented, band-pass convolutional filters such as Gabor filters, this is not necessary for convolutional square-pooling architectures; even with random weights, these architectures are sensitive to Gabor-like input features.

2.2. Evaluation of the Effect of Convolution

To empirically test the link between the preceding analysis and performance, we disrupt a key feature of the architecture and evaluate the resulting impact on classification accuracy. In particular, the convolution operation features centrally in the analysis, and we therefore tested the classification performance of square-pooling architectures with and without convolution. The non-convolutional networks were identical

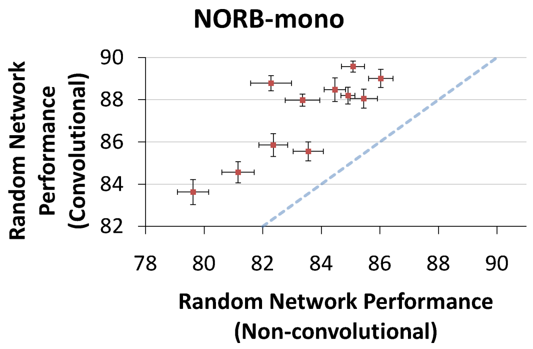


Figure 3. Classification performance of convolutional square-pooling architectures vs non-convolutional square-pooling architectures. Red squares indicate mean performance of a single architecture averaged across random seeds. Error bars represent a 95% confidence interval.

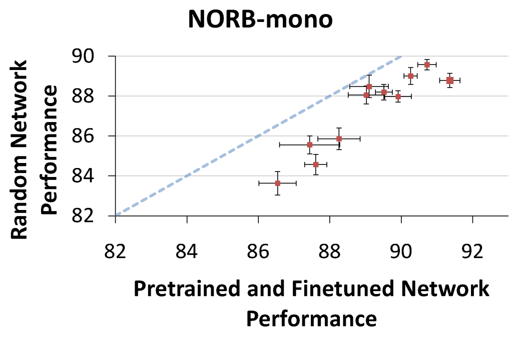
to their convolutional counterparts except that a different random filter was applied at each location in the input. We tested the classification performance of these networks across a variety of architectural parameters on variants of the NORB (LeCun et al., 2004) and CIFAR-10 (Krizhevsky, 2009) datasets. For NORB, we took the left side of the stereo images in the normalized-uniform set and downsized them to 32x32, and for CIFAR-10, we converted each 32x32 image from RGB to grayscale. We term these modified datasets NORB-mono and CIFAR-10-mono.

Classification experiments were run on 11 randomly chosen architectures with filter sizes in {4x4, 8x8, 12x12, 16x16}, pooling sizes in {3x3, 5x5, 9x9} and filter strides in {1, 2}. Only 11 of the 24 parameter combinations were evaluated due to computational expense.[2] On NORB-mono, we used 10 unique sets each of convolutional and non-convolutional random weights[3] for each architecture, giving a total of 220 networks, while we used 5/5 sets on CIFAR-10-mono, for a total of 110 networks. The classification accuracies of the sets of random weights were averaged to give a final score for the convolutional and non-convolutional versions of each architecture. We used a linear SVM (liblinear) for classification, with regularization parameter C determined by cross-validation over $\{10^{-3}, 10^{-1}, 10^{1}\}$.

As expected, the convolutional random-weight networks outperformed the non-convolutional random-

[2]Exact parameters used and code for reproducing the experiments are available at `http://www.stanford.edu/~asaxe/random_weights.html`.

[3]We found that the distribution the random weights were drawn from (e.g. uniform, Laplacian, Gaussian) did not affect their classification performance, so long as the distribution was centered about zero.



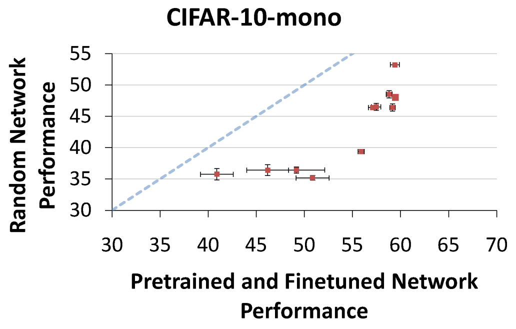


Figure 4. Classification performance of convolutional square-pooling networks with random weights vs with pretrained and finetuned weights. Left: NORB-mono. Right: CIFAR-10-mono. (Error bars represent a 95% confidence interval)

weight networks by an average of 3.8 ± 0.29% on NORB-mono (Fig. 3) and 1.5 ± 0.93% on CIFAR-10-mono. We note, however, that non-convolutional networks still performed significantly better than an SVM learned on raw pixels, which gives 72.6% on NORB-mono and 28.2% on CIFAR-10-mono.

3. What is the Contribution of Pretraining and Fine tuning?

The unexpectedly high performance of random-weight networks on NORB-mono leads us to a central question: is unsupervised pretraining and discriminative finetuning necessary, or do random weights in the right architecture perform just as well?

We investigate this by comparing the performance of pretrained and finetuned convolutional networks with random-weight convolutional networks on the NORB and CIFAR-10 datasets for the same 11 architectures evaluated in Section 2.2. Our convolutional networks were pretrained in an unsupervised fashion through local receptive field Topographic Independent Components Analysis (TICA), a feature-learning algorithm introduced by Le et al. (2010) that was shown to achieve high performance on both NORB and CIFAR-10. Importantly, this algorithm operates on the same class of square-pooling architectures as our networks. Finetuning was carried out by backpropagating softmax error signals. For speed, we terminate after just 80 iterations of L-BFGS (Schmidt, 2005).

The results are summarized in Fig. 4. As one would expect, the top-performing networks had trained weights, and pretraining and finetuning invariably increased the performance of a given architecture. This was especially true for CIFAR-10, where the top architecture using trained weights achieved an accuracy of 59.5 ±0.3%, while the top architecture using random weights only achieved 53.2 ±0.3%. Within our range of parameters, at least, pretraining and finetuning was necessary to attain near-top classification performance on CIFAR-10.

More surprising, however, is the finding that many trained networks lose out in terms of performance to random-weight networks with different architectural parameters. We conclude, similarly to Jarrett et al. (2009), that a search over a range of architectural parameters may be as beneficial to performance as pretraining and fine-tuning, especially since we often only have a broad architectural prior to work with. An important consideration, then, is to be able to perform this search efficiently. In the next section, we present and justify a heuristic for doing so.

4. Fast Architecture Selection

A computationally expensive component of feature learning systems is architecture selection, which typically involves large grid searches through the space of architecture parameters. Each architecture to be evaluated requires substantial time to pretrain due to the large amounts of data used and the complexity of the learning algorithms. These computational challenges, already considerable for single-layer architectures, are exacerbated in deeper networks due to the exponential increase in the number of possible architectures.

Evaluating architectures with random weights provides one means of greatly speeding this architecture selection process. Crucially, we note that Fig. 4 shows a robust correlation between random-weight performance and the corresponding trained-weight performance. Hence large-scale searches of the space of network architectures can be carried out by using random-weight performance as a proxy for trained-weight per-

Table 1. Time comparison between normal architecture search and random weight architecture search.

	NORB-mono		CIFAR-10-mono	
	TICA	Random Weights	TICA	Random Weights
Pretraining	2.9h	-	26.1h	-
Finetuning	0.6h	-	1.0h	-
Classification	0.1h	0.1h	1.0h	1.0h
Total	3.6h	0.1h (36×)	28.1h	1.0h (28×)

Table 2. Classification results for various methods on NORB

Algorithm	Accuracy (NORB)	
Random Weights	95.2%	
K-means + Soft Activation Fn.	97.2%	(Coates et al., 2011)
Tiled Convolutional Neural Nets	96.1%	(Le et al., 2010)
Convolutional Neural Nets	94.4%	(Jarrett et al., 2009)
3D DBNs	93.5%	(Nair & Hinton, 2009)
DBMs	92.8%	(Salakhutdinov & Larochelle, 2010)
SVMs	88.4%	(Bengio & LeCun, 2007)

formance.[4]

This removes the need to pretrain and finetune candidate architectures, providing an approximately order-of-magnitude speedup over the normal approach of evaluating each pretrained and finetuned network, as shown in Table 1.[5]

As a concrete example, we see from Fig. 4 that if we had performed this random-weight architectural search and selected the top three architectures according to their random-weight performance, we would have found the top-performing overall trained-weight architecture for both NORB-mono and CIFAR-10-mono.

To demonstrate the effectiveness of our method in a more realistic setting, we consider the "stacked" convolutional-square poooling architecture (Le et al., 2010), in which the pooling unit activities from one convolutional square-pooling architecture are provided as inputs to a second convolutional square-pooling architecture. Because each layer's architectural parameters can vary independently, the space of architectures is very large and evaluating every architectural combination, even with random weights search, is expensive. We therefore used random weights to search through the first layer's parameters on the CIFAR-10 dataset, holding the parameters of the second layer fixed. Top random-weights architectures were pretrained and finetuned, and evaluated on a hold out validation set. This search improved results from 70.6% with our initial parameters (hand-chosen based on prior experience), to 73.1% (as reported in Le et al., 2010). By doing random-weights search, we were able to evaluate a much broader range of parameters than would have been feasible with training. We note that even though random-weights performance was far below pretrained, finetuned performance, the correlation between the two was strong enough to enable this search.

[4]The mean random-weight performance over several random initializations should be used to evaluate an architecture because performance differences between random initializations can be large (note error bars in Fig. 4).

[5]Different architectures take different amounts of time to train. Here we report the speedup with one representative architecture.

5. Distinguishing the Contributions of Architecture and Learning

Finally, we provide evidence that current state-of-the-art feature detection systems may derive a surprising amount of performance from their architecture alone. In particular, we focus once again on local receptive field TICA (Le et al., 2010), which has achieved classification performance superior to many other methods reported in the literature. As TICA involves a square-pooling architecture with sparsity-maximizing pretraining, we investigate how well a simple convolutional square-pooling architecture can perform without any learning. We find that a convolutional version[6] of Le et al.'s top-performing architecture with 48 filters still obtains highly competitive results on NORB, as shown in Table 2.

With such results, it may be useful to distinguish the contributions of architectures from those of learning algorithms by reporting the performance of random weights. Recognizing these distinct contributions to performance may help identify good learning algorithms, which might not always give the best reported performances if they are paired with bad architectures; similarly, we will be better able to sift out the good architectures independently from particular learning algorithms.

6. Conclusion

The performance of convolutional square-pooling architectures with random weights demonstrates the important role architecture plays in feature learning systems for object recognition. We find that features generated by random-weight networks reflect intrinsic properties of their architecture; for instance, convolutional pooling architectures enable even random-weight networks to be frequency selective, and we prove this in the case of square pooling.

One practical result from this study is a new method for fast architecture selection. Our experimental results show that the performance of single layer convolutional square pooling networks with random weights is significantly correlated with the performance of such architectures after pretraining and finetuning. However the degree to which this correlation still persists in deeper networks with more varied nonlinearities remains to be seen. If it is strong, we hope the use of random weights for architecture search will improve the performance of state-of-the-art systems.

Acknowledgments We give warm thanks to Quoc Le and Jiquan Ngiam for code and ideas. We thank Accelereyes for help with their Jacket computing platform. This work is supported by the DARPA Deep Learning program under contract number FA8650-10-C-7020. Andrew Saxe is supported by a NDSEG and Stanford Graduate Fellowship.

A. Proof of Theorem 2.1

The activation of a single pooling unit is determined by first convolving a filter $\tilde{f} \in \mathbb{R}^{n\times n}$ with a portion of the input image $x \in \mathbb{R}^{n\times n}$, and then computing the sum of the squares of the convolution coefficients. Since convolution is a linear operator, we can recast the above problem as a matrix norm problem $\max_{x\in\Re^{N^2}} ||C\hat{x}||_2^2$ subject to $||\hat{x}||_2 = 1$. where the matrix C implements the two-dimensional circular convolution convolution of $\hat{f}$ and $\hat{x}$, where $\hat{f}$ and $\hat{x}$ are formed by flattening $\tilde{f}$ and x, respectively, into a column vector.

[6]The architectural parameters of this network are the same as that of the 96.1%-scoring architecture in (Le et al., 2010), with the exception that ours is convolutional and has 48 maps instead of 16.

The matrix C is *doubly block circulant* (see Gray, 2006; Tee, 2007, for overviews), i.e., each row of blocks is a circular shift of the previous row,

$$C = \begin{bmatrix} C_1 & C_2 & \cdots & C_{n-1} & C_n \\ C_n & C_1 & \cdots & C_{n-2} & C_{n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ C_2 & C_3 & \cdots & C_n & C_1 \end{bmatrix},$$

and each block C_n is itself circulant, such that each row (which contains a subset of the filter coefficients used in the convolution) is a circular shift of the previous row. Hence we obtain the optimization problem $\max_{x\in\mathbb{R}^{n^2},x\neq 0} \frac{x^*C^*Cx}{x^*x}$. This is a positive semidefinite quadratic form, for which the solution is the eigenvector associated with the maximal eigenvalue of C^*C. To obtain an analytical solution we reparametrize the optimization to diagonalize C^*C so that the eigenvalues and eigenvectors may be read off the diagonal. We change variables to $z = Fx$ where F is the unitary 2D discrete Fourier transform matrix. The objective then becomes

$$\frac{x^*C^*Cx}{x^*x} = \frac{z^*FC^*CF^*z}{z^*FF^*z} \quad (4)$$

$$= \frac{z^*FC^*F^*FCF^*z}{z^*z} = \frac{z^*\Lambda^*\Lambda z}{z^*z} \quad (5)$$

where from (4) to (5) we have twice used the fact that the Fourier transform is unitary ($F^*F = I$), and we have used the fact that the 2D Fourier transform diagonalizes doubly block circulant matrices (which corresponds to the convolution theorem $\mathcal{F}\{f * x\} = \mathcal{F}f\mathcal{F}x)$, that is, $FCF^* = \Lambda$. The matrix Λ is diagonal, with coefficients $\Lambda_{ii} = n(Ff)_i$, equal to the Fourier coefficients of the convolution filter f scaled by n. Hence we obtain the equivalent optimization problem $\max_{z\in\mathbb{C}^{n^2},z\neq 0} \frac{z^*|\Lambda|^2z}{z^*z}$ subject to $F^*z \in \mathbb{R}$ where the matrix $|\Lambda|^2$ is diagonal with entries $|\Lambda|^2_{ii} = n^2|(Ff)_i|^2$, the square of the magnitude of the Fourier coefficients of f scaled by n^2. The constraint $F^*z \in \mathbb{R}$ ensures that x is real, despite z being complex. Because the coefficient matrix is diagonal, we can read off the eigenvalues as $\lambda_i = n^2|\Lambda|^2_{ii}$ with corresponding eigenvector e_i, the i^{th} unit vector. The global solution to the optimization problem, setting aside the reality condition, is any mixture of eigenvectors corresponding to maximal eigenvalues, scaled to have unit norm.

To establish the optimal input taking account of the reality constraint, we must ensure that Fourier coefficients corresponding to negative frequencies are their complex conjugate, that is, the solution must satisfy the reality condition $z_{-i} = \overline{z_i}$. One choice for satisfying the reality condition is to take

$$z_j = \begin{cases} \dfrac{a_{|j|}}{\sqrt{2}} e^{i\text{sgn}(j)\phi_{|j|}} & \lambda_j \in \max \lambda \\ 0 & \text{otherwise} \end{cases}$$

where $a, \phi \in \mathbb{R}^q$ are vectors of arbitrary coefficients, one for each maximal frequency in Ff, and $||a|| = 1$. Then, for nonzero z_j, $z_{-j} = \frac{a_{|j|}}{\sqrt{2}} e^{i\text{sgn}(-j)\phi_{|j|}} = \overline{\frac{a_{|j|}}{\sqrt{2}} e^{i\text{sgn}(j)\phi_{|j|}}} = \overline{z_j}$ so the reality condition holds. Since this attains the maximum for the problem without the reality constraint, it must also be the maximum for the complete problem. Converting z back to the spatial domain proves the result. □

B. Proof of Theorem 2.2

Let V_n and C_n denote matrices performing valid and circular convolution, respectively, of a filter $f \in \mathbb{R}^{k \times k}$ with an input of size $n \times n$. Hence we have $p_v(x) = \|V_n \hat{x}\|^2$ and $p_c(x) = \|C_n \hat{x}\|^2$ where $\hat{x} \in \mathbb{R}^{n^2}$ is formed by flattening x into a column vector. We first note that both circular and valid convolution compute the same $n-k+1 \times n-k+1$ filter responses in the interior of the input. Hence we can order the rows of C_n so that the first $(n-k+1)^2$ are exactly V_n, and the remaining $n^2 - (n-k+1)^2 = 2(k-1)n - (k-1)^2 \equiv m$ differ. Forming the rows that differ into the matrix $D_n \in \mathbb{R}^{m \times n^2}$,

$$\begin{aligned} p_c(x) - p_v(x) &= \|C_n \hat{x}\|^2 - \|V_n \hat{x}\|^2, & (6) \\ &= \|D_n \hat{x}\|^2 \geq 0, & (7) \end{aligned}$$

and so $p_v(x) \leq p_c(x)$ for any x. Therefore,

$$\begin{aligned} p_v(x_v) &\leq p_c(x_v) \leq p_c(x^{\sin}), & (8) \\ &= p_v(x^{\sin}) + \left\|D_n \hat{x}^{\sin}\right\|^2. & (9) \end{aligned}$$

Finally we bound $\left\|D_n \hat{x}^{\sin}\right\|^2$. Because $x^{\sin}$ is a sinusoid that spans the entire input and the total norm is constrained, the individual elements diminish with n like $\left|x^{\sin}[j,l]\right| \leq \frac{\sqrt{2}}{n}$ (see Eqn. 2). Next we note that $\left\|D_n \hat{x}^{\sin}\right\|^2 = \sum_{i=1}^m (d_i^T \hat{x}^{\sin})^2$, where d_i^T is the i^{th} row of D_n, and $m = 2(k-1)n - (k-1)^2$ is the number of rows in D_n. The vector d_i^T contains the filter coefficients f and is otherwise zero; hence it has only k^2 nonzero entries. We can therefore form the vector $\tilde{x}_i \in \mathbb{R}^{k^2}$ from just those elements of $\hat{x}^{\sin}$ which will be involved in computing the dot product, such that $d_i^T \hat{x}^{\sin} = f^T \tilde{x}_i$. Then we have

$$\begin{aligned} \sum_{i=1}^m \left(d_i^T x\right)^2 &= \sum_{i=1}^m \left(f^T \tilde{x}_i\right)^2 \leq ||f||^2 \sum_{i=1}^m ||\tilde{x}_i||^2, & (10) \\ &\leq ||f||^2 \sum_{i=1}^m \frac{2k^2}{n^2} \leq 4k^3 ||f||^2 n^{-1}. & (11) \end{aligned}$$

Hence returning to (9) and inserting (11),

$$p_v(x_v) - p_v\left(x^{\sin}\right) \leq 4k^3 ||f||^2 n^{-1}$$

which proves the result. □

References

Bengio, Y. and LeCun, Y. Scaling learning algorithms towards A.I. In Bottou, L., Chapelle, O., and Weston, J. (eds.), *Large-Scale Kernel Machines*, pp. 321–360. MIT Press, 2007.

Coates, A., Lee, H., and Ng, A. Y. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, 2011.

Gray, R. M. Toeplitz and Circulant Matrices: A review. *Foundations and Trends in Communications and Information Theory*, 2(3):155–239, 2006.

Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y. What is the best multi-stage architecture for object recognition? In *IEEE 12th International Conference on Computer Vision*, pp. 2146–2153, 2009.

Krizhevsky, A. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, 2009.

Le, Q.V., Ngiam, J., Chen, Z., Koh, P., Chia, D., and Ng, A. Tiled convolutional neural networks. In *Advances in Neural Information Processing Systems 23*, 2010.

LeCun, Y., Huang, F.J., and Bottou, L. Learning methods for generic object recognition with invariance to pose and lighting. In *Conference on Computer Vision and Pattern Recognition*, 2004.

Nair, V. and Hinton, G. 3D Object Recognition with Deep Belief Nets. In *Advances in Neural Information Processing Systems 22*, 2009.

Pinto, N. and Cox, D.D. Beyond simple features: A large-scale feature search approach to unconstrained face recognition. In *IEEE International Conference on Automatic Face and Gesture Recognition*, 2011.

Pinto, N., Doukhan, D., DiCarlo, J. J., and Cox, D. D. A high-throughput screening approach to discovering good forms of biologically inspired visual representation. *PLoS Computational Biology*, 5(11), 2009.

Salakhutdinov, R. and Larochelle, H. Efficient learning of deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, 2010.

Schmidt, M. minfunc. 2005. URL `http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html`.

Tee, G. J. Eigenvectors of block circulant and alternating circulant matrices. *New Zealand Journal of Mathematics*, 36:195–211, 2007.