On Random Weights and Unsupervised Feature Learning

Andrew M. Saxe, Pang Wei Koh, Zhenghao Chen,
Maneesh Bhand, Bipin Suresh, and Andrew Y. Ng
Stanford University
Stanford, CA 94305
{asaxe,pangwei,zhenghao,mbhand,bipins,ang}@cs.stanford.edu

Abstract

Recently two anomalous results in the literature have shown that certain feature learning architectures can perform very well on object recognition tasks, without training. 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. [1] 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. [2], who evaluated thousands of 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 and see [3]).

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, we find that the performance improvement can be modest and sometimes smaller than the performance differences due to architectural parameters. The key to good performance, then, lies not only in improving the learning algorithms but also in searching for the most suitable architectures [1]. This motivates a fast heuristic for architecture search 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

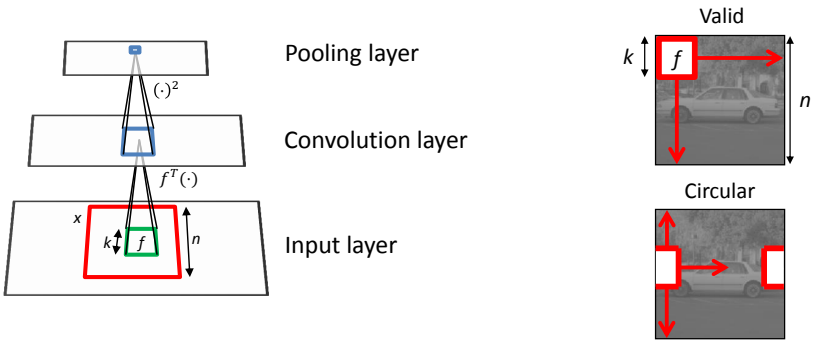

Figure 1: Left: Convolutional square pooling architecture. In our notation x is the portion of the input layer seen by a particular pooling unit (shown in red), and f is the convolutional filter applied in the first layer (shown in green). Right: Valid convolution applies the filter only at locations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

the time-consuming learning process by evaluating candidate architectures using random weights as a proxy for learned weights.

We conclude by showing that a surprising fraction of performance can be contributed by the architecture alone. In particular, we present a 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 learning system. 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 Jarrett et al. [1] found that only some architectures yielded 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. numerically computed the optimal input to each neuron using gradient descent in the input space, and found that the optimal inputs were often sinusoid and insensitive to translations. To better understand 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 envisaged 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 as follows: First, a filter $f \in \mathbb{R}^{k \times k}, k \leq n$ is convolved 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

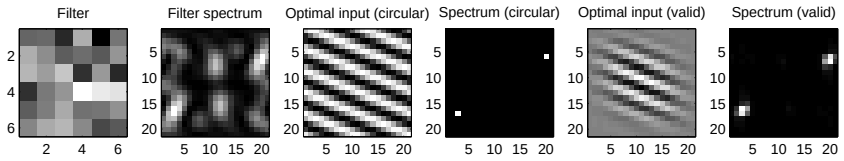

Figure 2: Left to right: Randomly sampled filter used in the convolution. Square of the magnitude of the Fourier coefficients of the filter. Input that would maximally activate a pooling unit in the case of circular convolution. Magnitude of the Fourier transform of the optimal input. Optimal input for valid convolution. Magnitude of the Fourier transform of the optimal input for valid convolution.

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 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, the optimal input will be near a sinusoid at the maximal frequency or frequencies 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. Recall that valid convolution applies the filter f only at locations where f lies entirely within x, yielding a result of size $n-k+1 \times n-k+1$. By contrast 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, 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 we will 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. Let $\tilde{f}$ be formed by zero-padding f to size $n \times n$, let $M = \{(v_1, h_1), (v_2, h_2), \ldots, (v_q, h_q)\}$ be the set of frequencies of maximal magnitude in the discrete Fourier transform of $\tilde{f}$, and suppose that the zero frequency component is not maximal.*[1] *Then the set of norm-one inputs that maximally activate a circular convolution square-pooling unit p_c is*

$$\left\{ x^{opt} \mid x^{opt}[m,s] = \frac{\sqrt{2}}{n} \sum_{j}^{q} a_j \cos\left(\frac{2\pi m v_j}{n} + \frac{2\pi s h_j}{n} + \phi_j \right), \|a\| = 1, a \in \mathbb{R}^q, \phi \in \mathbb{R}^q \right\}.$$

Proof The proof is given in Appendix A.

Although Theorem 2.1 allows for the case in which there are multiple maximal frequencies in f, if the coefficients of f are drawn independently from a continuous probability distribution, then with probability one M will have just one maximal frequency, $M = \{(v, h)\}$. In this case, the optimal input is simply a sinusoid at this frequency, of arbitrary phase:

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

This equation 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*.

[1] For notational simplicity we omit the case where the zero frequency component is maximal, which requires slightly different conditions on a and ϕ.

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. An example of this optimization for one random filter is shown in Fig. 2. In particular, note that the frequency of the optimal input (as shown in the plot of the spectrum) corresponds to the maximum in the spectrum of the random filter.

2.1 Optimal input for circular convolution is near-optimal for normal convolution

The above analysis computes the optimal input for the case of 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, we might expect that the optimal input for one would be quite good for the other. In Fig. 2, the optimal input for valid and circular convolution differs dramatically only near the border of the input region. We make this rigorous in the result below.

Theorem 2.2 *Let $p_v(x)$ denote the activation of a single pooling unit in a valid convolution, square-pooling architecture in response to an input x, and let x_v^{opt} and x_c^{opt} denote the optimal norm-one inputs for valid and circular convolution, respectively. Then if x_c^{opt} is composed of a single sinusoid,*

$$\lim_{n\to\infty} \left| p_v\left(x_v^{opt}\right) - p_v\left(x_c^{opt}\right) \right| = 0.$$

Proof The proof is supplied in Appendix B (deferred to the supplementary material due to space constraints).

An empirical example illustrating the convergence is given in Fig. 3. This result, in combination with Theorem 2.1, 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 these two properties contribute to the good performance reported by Jarrett et al. [1] and Pinto et al [2]. Frequency selectivity and translation invariance 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 features of their input.

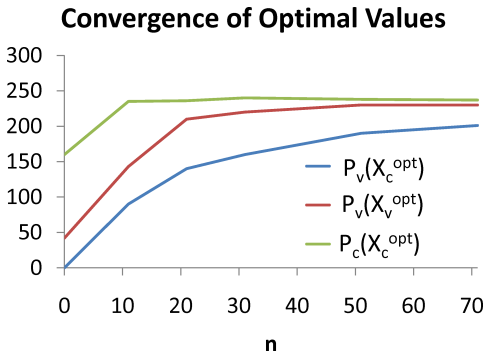

Figure 3: Empirical demonstration of the convergence.

2.2 Empirical evaluation of the effect of convolution

To show that convolutional networks indeed detect more salient features, we tested the classification performance of both convolutional and non-convolutional square-pooling networks across a variety of architectural parameters on variants of the NORB and CIFAR-10 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 different architectures with varying filter sizes {4x4, 8x8, 12x12, 16x16}, pooling sizes {3x3, 5x5, 9x9} and filter strides {1, 2}. On NORB-mono, we used 10 unique sets of convolutional and 10 unique sets of non-convolutional random weights 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 [4] for classification, with regularization parameter C determined by cross-validation over $\{10^{-3}, 10^{-1}, 10^{1}\}$. Algorithms were initially prototyped on the GPU with Accelereyes Jacket.

As expected, the convolutional random-weight networks outperformed the non-convolutional random-weight networks by an average of $3.8 \pm 0.29\%$ on NORB-mono (Fig.4) and $1.5 \pm 0.93\%$ on CIFAR-10-mono. 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 0.

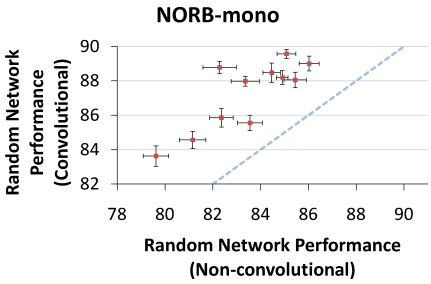

Figure 4: Classification performance of convolutional architectures vs non-convolutional architectures

Finally, we note that non-convolutional networks still performed significantly better than the raw-pixel baseline of 72.6% on NORB-mono and 28.2% on CIFAR-10-mono. Moreover, on CIFAR-10-mono, some networks performed equally well with or without convolution, even though convolutional networks did better on average. While we would expect an improvement over the raw-pixel baseline just from the fact that the neural networks work in a higher-dimensional space (e.g., see [5]), preliminary experiments indicate that the observed performance of non-convolutional random networks stem from other architectural factors such as filter locality and the form of the non-linearities applied to each hidden unit activation. This suggests that convolution is just one among many different architectural features that play a part in providing good classification performance.

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 can the right architecture compensate?

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. 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. [6] that was shown to achieve high performance on both NORB and CIFAR-10, and which operates on the same class of square-pooling architectures as our networks. We use the same experimental settings as before, with 11 architectures and 10 random initializations per architecture for the NORB dataset and 5 random initializations per architecture for the CIFAR-10 dataset. Pretraining these with TICA and then finetuning them gives us a total of 110 random-weight and 110 trained networks for NORB, and likewise 55 / 55 networks for CIFAR-10. Finetuning was carried out by backpropagating softmax error signals. For speed, we terminate after just 80 iterations of L-BFGS [7].

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, was finding that many learnt networks lose out in terms of performance to random-weight networks with different architectural parameters. For instance, the architecture with 8x8 filters, a convolutional stride size of 1 and a pooling region of 5x5 gets 89.6 ±0.3% with random weights on NORB, while the (8x8, 2, 3x3) architecture gets 86.5 ±0.5% with pretrained and finetuned weights. Likewise, on CIFAR-10, the (4x4, 1, 3x3) architecture gets 53.2 ±0.3% with random weights, while the (16x16, 1, 3x3) architecture gets 50.9 ±1.7% with trained weights.

We can hence conclude that to get good performance, one cannot solely focus on the learning algorithm and neglect a search over a range of architectural parameters, 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.

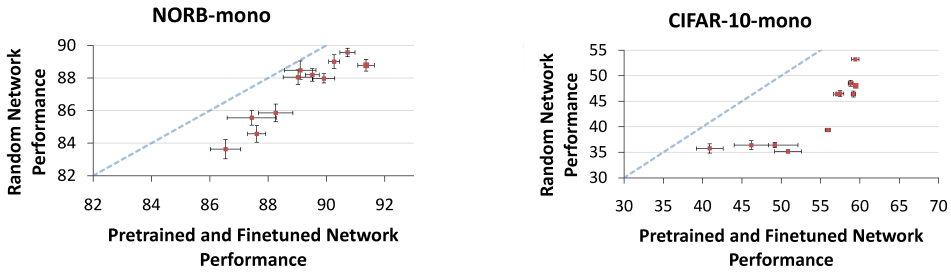

Figure 5: Classification performance of random-weight networks vs pretrained and finetuned networks. Left: NORB-mono. Right: CIFAR-10-mono (Error bars represent a 95% confidence interval about the mean)

4 Fast architecture selection

When we plot the classification performance of random-weight architectures against trained-weight architectures, a distinctive trend emerges: we see that architectures which perform well with random weights also tend to perform well with pretrained and finetuned weights, and vice versa (Fig. 5). Intuitively, our analysis in Section 2 suggests that random-weight performance is not truly random but should correlate with the corresponding trained-weight performance, as both are linked to intrinsic properties of the architecture. Indeed, this happens in practice.

This suggests that large-scale searches of the space of possible network architectures can be carried out by evaluating the mean performance of such architectures over several random initializations. This would allow us to determine suitable values for architecture specific parameters without the need for computationally expensive pretraining and finetuning. A subset of the best performing architectures could then be selected to be pretrained and finetuned.[2]

As classification is significantly faster than pretraining and finetuning, this process of heuristic architecture search provides a commensurate speedup over the normal approach of evaluating each pretrained and finetuned network, as shown in Fig. 1.[3]

	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(36x)**	**28.1h**	**1.0h(28x)**

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

As a concrete example, we see from Fig. 5 that if we had performed this random-weight architectural search and selected the top three random-weight architectures for pretraining and finetuning, we would have found the top-performing overall architecture for both NORB-mono and CIFAR-10-mono.

5 Distinguishing the contributions of architecture and learning

We round off the paper by presenting evidence that current state-of-the-art feature detection systems derive a surprising amount of performance just from their architecture alone. In particular, we focus once again on local receptive field TICA [6], 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

[2]We note that it is insufficient to evaluate an architecture based on a single random initialization because the performance difference between random initializations is not generally negligible. Also, while there was a significant correlation between the mean performance of an architecture using random weights and its performance after training, a particular high performing initialization did not generally perform well after pretraining and finetuning.

[3]Different architectures have different numbers of hidden units and consequently take different amounts of time to train, but the trend is similar across all architectures, so here we only report the speedup with one representative architecture.

square-pooling architecture can perform without any learning. It turns out that a convolutional and expanded version[4] of Le et al.'s top-performing architecture is sufficient to obtain highly competitive results on NORB, as shown in Table 2 below.

With such results, it becomes important to distinguish the contributions of architectures from those of learning systems by reporting the performance of random weights. At the least, recognizing these separate contributions will help us to hone in towards the 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 the learning algorithms.

Algorithm	Accuracy (NORB)
Random Weights	94.0%
Tiled Convolutional Neural Nets	96.1% [6]
Convolutional Neural Nets	94.1% [17], 94.4% [1]
3D DBNs	93.5% [16]
SVMs	88.4% [17]
DBMs	92.8% [18]

Table 2: Classification results for various methods on NORB

6 Conclusion

The performance of random-weight networks with convolutional pooling architectures demonstrates the important role architecture plays in feature representation for object recognition. We find that 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 major practical result from this study is a new method for fast architectural selection, as experimental results show that the performance of random-weight networks is significantly correlated with the performance of such architectures after pretraining and finetuning. While random weights are no substitute for pretraining and finetuning, we hope their use for architecture search will improve the performance of state-of-the-art systems.

Acknowledgments This work was supported by the DARPA Deep Learning program under contract number FA8650-10-C-7020. AMS is supported by NDSEG and Stanford Graduate fellowships. We warmly thank Quoc Le, Will Zou, Chenguang Zhu, and three anonymous reviewers for helpful comments.

A Optimal input for circular convolution

The appendices provide proofs of Theorems 2.1 and 2.2. They assume some familiarity with Fourier transforms and linear algebra. Because of space constraints, please refer to the supplementary material for Appendix B.

The activation of a single pooling unit is determined by first convolving a filter $\hat{f} \in \mathbb{R}^{n \times n}$ with a portion of the input image $\hat{x} \in \mathbb{R}^{n \times n}$, 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

$$\begin{aligned} \max_{x \in \Re^{N^2}} \quad & \|A_c x\|_2^2 \\ \text{subject to} \quad & ||x||_2 = 1. \end{aligned} \tag{1}$$

where the matrix A_c implements the two-dimensional circular convolution convolution of f and x, where f and x are formed by flattening $\hat{f}$ and $\hat{x}$, respectively, into a column vector in column-major order.

[4]The architectural parameters of this network are the same as that of the 96.2%-scoring architecture in [6], with the exception that ours is convolutional, and has 48 maps instead of 16, since without supervised finetuning, random-weight networks are less prone to overfitting.

The matrix A_c has special structure that we can exploit to find the optimal solution. In particular, A_c is block circulant, i.e., each row of blocks is a circular shift of the previous row,

$$A = \begin{bmatrix} A_1 & A_2 & \cdots & A_{n-1} & A_n \\ A_n & A_1 & \cdots & A_{n-2} & A_{n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_3 & A_4 & \cdots & A_1 & A_2 \\ A_2 & A_3 & \cdots & A_n & A_1 \end{bmatrix}.$$

Additionally each block A_n is itself circulant, i.e., each row is a circular shift of the previous row,

$$A_i = \begin{bmatrix} a_1 & a_2 & \cdots & a_{n-1} & a_n \\ a_n & a_1 & \cdots & a_{n-2} & a_{n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_3 & a_4 & \cdots & a_1 & a_2 \\ a_2 & a_3 & \cdots & a_n & a_1 \end{bmatrix},$$

where each row contains an appropriate subset of the filter coefficients used in the convolution. Hence the matrix A_c is *doubly block circulant*. Returning to (2), we obtain the optimization problem

$$\max_{x \in \mathbb{R}^{n^2}, x \neq 0} \frac{x^* A_c^* A_c x}{x^* x}. \tag{2}$$

This is a positive semidefinite quadratic form, for which the solution is the eigenvector associated with the maximal eigenvalue of $A_c^* A_c$. To obtain an analytical solution we will reparametrize the optimization to diagonalize $A_c^* A_c$ so that the eigenvalues and eigenvectors may be read off of the diagonal. We change variables to $z = Fx$ where F is the unitary 2D discrete Fourier transform matrix. The objective then becomes

$$\frac{x^* A_c^* A_c x}{x^* x} = \frac{z^* F A_c^* A_c F^* z}{z^* F^* F z} \tag{3}$$

$$= \frac{z^* F A_c^* F^* F A_c F^* z}{z^* z} \tag{4}$$

$$= \frac{z^* D^* D z}{z^* z} \tag{5}$$

where from (3) to (4) we have twice used the fact that the Fourier transform is unitary ($F^* F = I$), and from (4) to (5) we have used the fact that the Fourier transform diagonalizes block circulant matrices (which corresponds to the convolution theorem $\mathcal{F}\{f * x\} = \mathcal{F} f \mathcal{F} x$), that is, $F A_c F^* = D$. The matrix D is diagonal, with coefficients $D_{ii} = n(Ff)_i$, equal to the Fourier coefficients of the convolution filter f scaled by n. Hence we obtain the equivalent optimization problem

$$\max_{z \in \mathbb{C}^{n^2}, z \neq 0} \frac{z^* |D|^2 z}{z^* z}, \tag{6}$$

$$\text{subject to} \quad F^* z \in \mathbb{R} \tag{7}$$

where the matrix $|D|^2$ is diagonal with entries $|D|_{ii}^2 = n^2 |(Ff)_i|^2$, 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 $\lambda_i = n^2 |D|_{ii}^2$ 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

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

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 time domain proves the result. □

References

[1] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, 2009.

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

[3] N. Pinto and D.D. Cox. An evaluation of the invariance properties of a biologically-inspired system for unconstrained face recognition. In *BIONETICS*, 2010.

[4] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[5] A. Rahimi and B. Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization. In *NIPS*, 2008.

[6] Q.V. Le, J. Ngiam, Z. Chen, P. Koh, D. Chia, and A. Ng. Tiled convolutional neural networks. In *NIPS, in press*, 2010.

[7] M. Schmidt. minFunc, http://www.cs.ubc.ca/ schmidtm/Software/minFunc.html. 2005.

[8] Y. LeCun, F.J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *CVPR*, 2004.

[9] Y. Boureau, J. Ponce, and Y. LeCun. A Theoretical Analysis of Feature Pooling in Visual Recognition. In *ICML*, 2010.

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

[11] R. M. Gray. Toeplitz and Circulant Matrices: A review. *Foundations and Trends in Communications and Information Theory*, 2(3), 2005.

[12] N. K. Bose and K. J. Boo. Asymptotic Eigenvalue Distribution of Block-Toeplitz Matrices. *Signal Processing*, 44(2):858–861, 1998.

[13] A. Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, Technical report, Computer Science Department, University of Toronto, 2009.

[14] D. Erhan, Y. Bengio, A. Courville, P. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11:625–660, 2010.

[15] G. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[16] V. Nair and G. Hinton. 3D Object Recognition with Deep Belief Nets. In *NIPS*, 2009.

[17] Y. Bengio and Y. LeCun. Scaling learning algorithms towards A.I. In *Large-Scale Kernel Machines*. MIT Press, 2007.

[18] R. Salakhutdinov and H. Larochelle. Efficient learning of deep boltzmann machines. In *AISTATS*, 2010.