CNN Representations Are Lower Dimensional Than You Might Think

Alex Neckar
Stanford University
Stanford, CA
aneckar@stanford.edu

Abstract

Convolutional Neural Networks (CNNs) do cool stuff but need tons of parameters to do it. We also don't know how they really work. Did you know that you can get rid of 95% of the parameters from the FC layer in an AlexNet without doing much to its classification accuracy? By forming a low-rank approximation of the FC layer, I show how to both save memory in implementing prediction and expose a low-dimensional feature-space that the CNN encodes input images in. By extracting these features and doing ICA, I find some directions in the space which seem to encode a set of high-level features that are not associated with any one particular class.

1. Introduction

Convolutional Neural Networks (CNNs) achieve state-of-the art performance in a number of real-world tasks, but the large numbers of parameters required by the highest-performing networks poses a challenge for embedded devices. To achieve the greatest energy and performance benefits, all parameters must reside on the same chip that does the forward computations. Even a relatively large custom chip, implemented in a modern process, which devoted a large portion of its area to SRAM arrays would only have tens of MBs of memory. However, for a large, deep CNN, such as [7], there might be over 100 MB of parameters. Almost all of a deep CNN's parameters are used by the final fully-connected layers. In the first section of this report I investigate the effect on classification performance of two simple parameter-reduction techniques on the second-to-last fully-connected layer in the AlexNet [6]. Starting from the pre-trained version of this network, I found that I was able to reduce the total number of parameters in this layer by about 95% with only a minimal loss of classification by replacing the original weight matrix with a either a sparse or low-rank approximation. I found similar results when sparsifying the original matrix by setting small values to zeros.

Since the low-rank factorization of the matrix does not reduce the classification performance of the system, this implies that there is effectively a low-dimensional space in which all input images are represented before being classified. Understanding how deep networks encode different image features could do a great deal to demystify how they function. In the second section of this report I attempt to gain some insight into the properties of this low-dimensional space by doing feature extraction on it and analyzing the resulting data. Unfortunately, the 100-dimensional space I investigated was still big enough to get lost in. I observed qualitatively that the independent components of the 100-dimensional feature data appear to have some interesting properties: they seem to represent a set of higher level qualities associated with no one particular class, and that arithmetic of independent components can produce intuitive results. This hints at a hypothesis that the CNN encodes images as combinations of these high-level qualities. Coming up with a quantitative metric to evaluate the strength of this hypothesis proved challenging, however.

2. Background

Although unknown to the author until partway through the project, the phenomenon that the fully-connected layers of a deep network may be factorized without harming performance has been previously studied, and appears to be an area of active research. Most research has focused on doing the factorization at training time, to reap the maximum benefits of the performance increase, but a naive approach where the matrix is initialized in factorized form does not seem to work well in most cases [1]. One counter-example to this observation is found in [8], where Sainath was able to train a deep (not convolutional) network for audio processing with the final FC layer replaced with a factorized matrix. In this case, factorization during training caused no substantial loss in performance. In [1], Denil blames the extra degrees of freedom in the factorized representation for the difficulties in training: if $W = AB^T$, then I could also write $W = (AD^{-1})(BD)^T$ for any invertible D. Denil suggests the problem may be solved if either A or B is fixed

before training and goes on to suggest ways of generating appropriate initialization values using autoencoders. Denil also observes that there are additional forms of sparsity that may be exploited in CNNs due the smoothness of the convolution filters both in X-Y space and depth: different from this report's focuse, Denil applies the low-rank factorization to the convolutional layers as well as the FC ones. The performance consequences of this optimization for networks implemented in caffe running on a CPU is explored in [4]. Unfortunately for those interested in embedded hardware, memory size is not directly studied. With respect to weight matrix factorization, this is the primary contribution of this report. I also consider sparsifying the individual entries of the matrix, forming a sparse-plus-low-rank approximation of the original matrix.

The fact that deep networks can serve as low-dimensional feature extractors is hardly unknown either. Hinton designed a deep (not convolutional) network with a special structure for this purpose in [2]. Although this network was initialized using RBMs to perform unsupervised learning on training images and did not use convolution, its key observations hold true for CNNs as well.

That a low-rank factorization of the FC matrices is possible and does not harm performance very much is interesting in its own right from an implementation perspective, but the fact that such factorizations create a low-dimensional bottleneck seems underappreciated.

3. Approach and Methods

3.1. Parameter Reduction Approach

In the first set of experiments, I investigated the effect of reducing the number of parameters in the fully-connected layer on network classification performance. I started with the pre-trained version of the AlexNet [6] available through Caffe [5], modifying the "fc7" layer (the second-to-last fully-connected layer). The network structure, is shown in Figure 1. If W is the original 4096×4096 matrix of fc7, I made the following approximation:

$$\hat{W} = W_{sp} + W_{LR}$$

W_{sp} is constructed by simple thresholding of the weights by their absolute values. Weights whose absolute values fall below the threshold T are set to zero.

$$W_{sp_{ij}} = \begin{cases} 0 & |W_{ij}| < T \\ W_{ij} & \text{otherwise} \end{cases}$$

W_{LR} is constructed by first forming the Singular Value Decomposition (SVD) of W and setting the singular values corresponding to ranks above R to zero. Intuitively, the SVD describes how any matrix-vector multiply may be thought of as rotation of the input vector, followed by a scaling, followed by another rotation. By knocking out all but the R largest singular values, I am essentially enlarging the nullspace of the matrix: all projections of the input data except those on the R left singular vectors are crushed. When these singular values are small to begin with, the matrix is already effectively low-rank and the approximation will function very similarly to the original matrix.

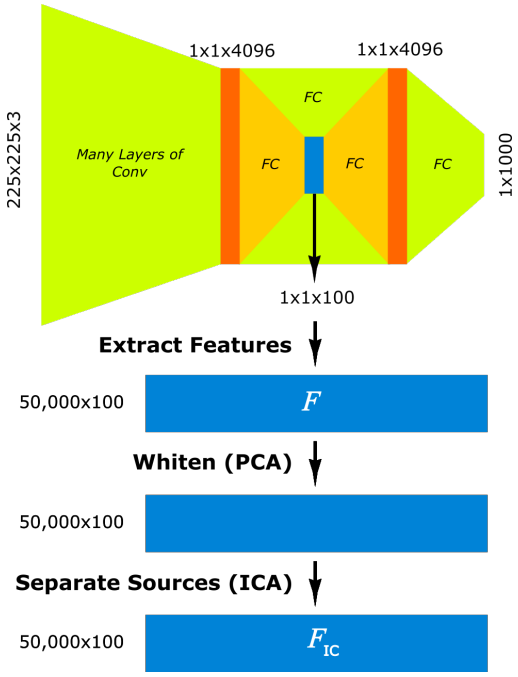

Figure 1. Network structure used in all experiments, showing the structure with W_{LR} superimposed over the original structure with W. W_{sp} is not represented in the diagram. In the parameter reduction experiments, the length R of the central feature was varied. The feature extraction pipeline is diagrammed, showing the whitening and ICA steps following the initial feature extraction.

$$\begin{aligned} W &= U\Sigma V^T \\ W_{LR} &= U\Sigma_{LR}V^T \\ \Sigma_{LR_{ii}} &= \begin{cases} 0 & i > R \\ \Sigma_{ii} & \text{otherwise} \end{cases} \end{aligned}$$

The number of parameters required to describe $\hat{W}$ is equal to the number of nonzero entries in W_{sp} plus $2 \cdot 4096 \cdot R$ for the reduced-rank versions of the left and right singular vectors. In a real implementation, storing the position of each sparse entry also needs to be considered.

3.2. Feature Extraction Approach

In the second set of experiments, I performed feature extraction on the set of 100-dimensional features that results

Figure 2. KNNs in F_{IC} of several ICs. Each row corresponds to a single IC (of the 91 total extracted). Note how most rows contains objects of various classes but has a consistent visual theme.

from a rank-100 approximation of W and tried to gain some idea of the structure of how the data is encoded to this 100-dimensional space. My data processing pipeline is shown in Figure 1. After making the rank-100 approximation of W, $\hat{W} = W_{LR}$ (no W_{sp} component), I fed in the 50,000-image validation set from the ILSVRC2012 challenge and extracted the 100D features, producing F, a 50,000$\times$100 data matrix.

Any single entry of the 100D feature vector does not necessarily have any particular meaning in terms of how information is encoded. Many different methods of encoding information in this 100D feature space could rely on having meaningful vectors could be mixed together in some arbitrary way. Attempting to isolate these 'interesting' directions in the 100D feature space is an instance of the blind-source separation problem. To perform blind-source separation, I first mean-centered and performed PCA on the data as a precursor to whitening it. I then performed ICA on the whitened data using the FastICA [3] implementation available in Python's 'scikit-learn' package. FastICA attempts to find projections of the data that have kurtosis most different from that of a Gaussian distribution. This leverages the Central Limit Theorem: sums of independent random variables tend towards a Gaussian distribution. By minimizing one measure of the Gaussian-ness of a given projection, FastICA thus attempts to pull out projections along with the data is not independent. The algorithm is not deterministic: in order to ensure that the independent components that I produced were robust, I ran FastICA with different random initializations and threw out the ICs that did not appear multiple times, using the absolute dot of the normalized mixing vectors, $|m_i^T m_j|$, as a measure of similarity (FastICA does not yield information about the sign or magnitude of the components it produces, only their directions). In practice, most of the ICs returned by any single run of FastICA appeared in the set of repeated directions. This technique produced a 50,000$\times C$ data matrix, F_{IC}, where C is the number of repeated directions. F_{IC} is F projected into the basis defined by the IC's mixing vectors, $M = [m_0, m_1, ..., m_C]^T$. Depending on whether C is greater or less than 100, different pseudoinverses may be applied to solve $F^T = MF_{IC}^T \Rightarrow F_{IC}^T = F_T M^\dagger$. In my case, C was

slightly less than 100, so I performed Least Squares with $\ell 2$ regularization: $M^{\dagger} = (M^T M + \lambda^2 I)^{-1} M^T$, minimizing $||MF_{IC}^T - F^T||_2^2 + \lambda ||F_{IC}||_2^2$.

A totally unguided exploration of this (still relatively large) feature space is challenging. Unless the data has some obvious structure, learning how information is encoded feels like a blind search. Fixing a hypothesis (even a naive one) can at least suggest a more principled set of experiments. A binary feature vector with K entries can encode 2^K different values. If the ICs represent a sufficiently rich set of qualities that images might possess, it is possible to imagine that the 1000 different classes known to the AlexNet are encoded in a similar fashion. Under this hypothesis, each image's representation in F_{IC} describes which IC qualities are mixed in what proportions to describe the image. The network has all the machinery necessary to implement this type of encoding, so I set out to design some experiments to collect evidence for or against my simple hypothesis.

A necessary intuitive aspect of the hypothesis is that the vectors encode some recognizable concept along each IC. To get a sense for what each IC might be encoding, I performed K-Nearest Neighbors (KNN) in F_{IC}, displaying the images whose encodings in had the smallest $\ell 2$ distance to each IC where I characterized the ICs each as being one row of $C \times C$ identity matrix, scaled by the mean norm of the rows of $F_I C$ (my ICs have the average feature vector length), as well as the the opposite of that vector. I then manually assessed whether each row had a coherent theme on either direction.

Since the number of the number of classes is very small compared to the total number of possible binary codes, it might be reasonable to expect that F_{IC} would be relatively sparse. In general, projecting the data onto the ICA should result in a more sparse encoding of the space than if the signals are initially randomly mixed: if F contains independent signals mixed in some arbitrary way F_{IC} should certainly be more sparse than F. To assess whether this is the case, I took the absolute values of both the whitened version of F and F_{IC}, sorted each row, then took the mean of all columns. Plotting these (rescaled) component decay profiles allows a comparison of the relative sparsity of each encoding.

Assuming that each image class requires the addition of several IC vectors to describe it completely, the hypothesis suggests that the ICs would not share the same direction as a collection of feature vectors that correspond to any particular class. To evaluate this, I tried to compare the ICs with the cluster centroids output by K-means, but K-means did not appear to be very successful: no metric of the clustering quality indicated a very good clustering. I also tried another approach relying on computing dot products between feature vectors. Assuming that each feature vector is surrounded by a cluster of features of the sames class, I compared the counts of large positive dot products between a randomly chosen feature vector and all other features. To keep the scales comparable, I normalized each feature to unit length. This approach hence discards all information about the length of feature vectors, implicitly assuming that all information is encoded directionally. Doing this several times might give you a relative idea how clustered one data set is compared to another. I used the dataset of points uniformly randomly distributed on the unit-hypersphere as a baseline reference. This approach gives very hard-to-interpret results when the data is not very white, however. Arbitrarily whitening the data also does not seem correct. Given the large number of assumptions inherent to this approach (clustering, directional encoding, requirement that data be whitened), I similarly did not draw anything conclusive from this line of experimentation.

Finally, for the hypothesis to be correct, it should be possible to take random images' encodings in F_{IC} and visualize both the image and the KNNs of the top ICs that make up the F_{IC} encoding. Conversely, it should be possible to add ICs together to synthesize image codes, visualizing the KNN of the resulting sum. This line of experiments is strictly qualitative, unfortunately, as it relies completely on human judegement: both correct interpretation of the ICs' associated image qualities and the degree to which those qualities are present in the reference image/set of results from KNN.

4. Results

4.1. Feature Extraction Results

4.2. Parameter Reduction Results

n the first set of experiments, I studied the combination of two ways of sparsifying the weights in the fc7 layer of my AlexNet, starting from pre-trained weights. I approximated the original W as $\hat{W} = W_{sp} + W_{LR}$. Figure 3 shows the singular value spectrum and the weight histogram of the original W. Figure 4 shows the effect on classification performance and number of parameters when the number of sparse entries in W_{sp} and the rank of $W_L R$ are varied independently. From the singular value and weight profile alone, it is surprising that sparsification and rank reduction can be applied so aggressively without damaging performance. The matrix does not appear to be effectively low rank (although adding IID Gaussians the entries of a low rank matrix can probably produce SV profiles like the one seen here), similarly the shape of the distribution of weights does not immediately suggest any significant threshold for weight sizes. With rank reduction alone, it was possible to reduce the numbers of parameters by about 95% while only sacrificing about 3% of top-5 classification accuracy. Sparsification appreared even more effective, allowing a simi-

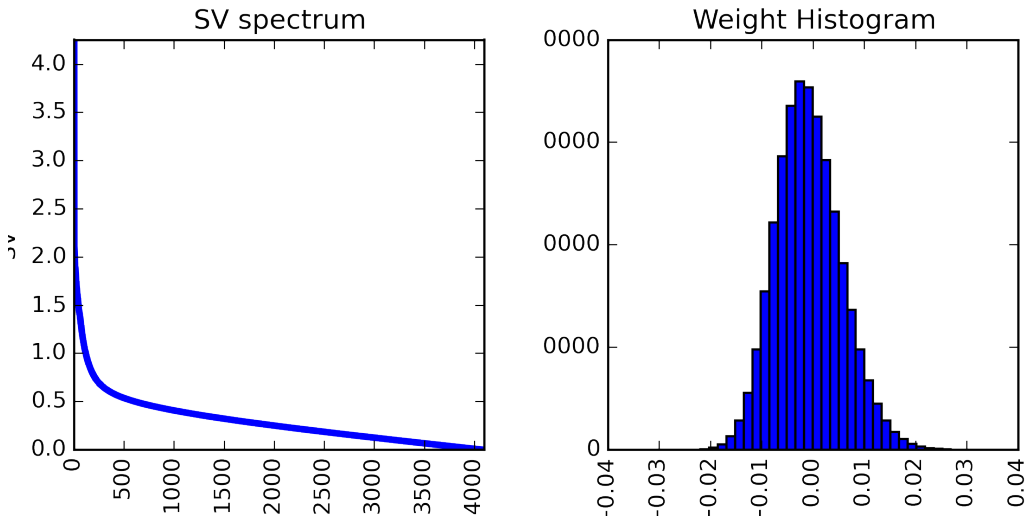

Figure 3. Left: singular value spectrum of original W weight matrix taken from fc7. Right: histogram of weight values. At first glance, the matrix doesn't seem to be effectively very low rank, and doesn't seem to have any obvious thresholds in the weight magnitudes.

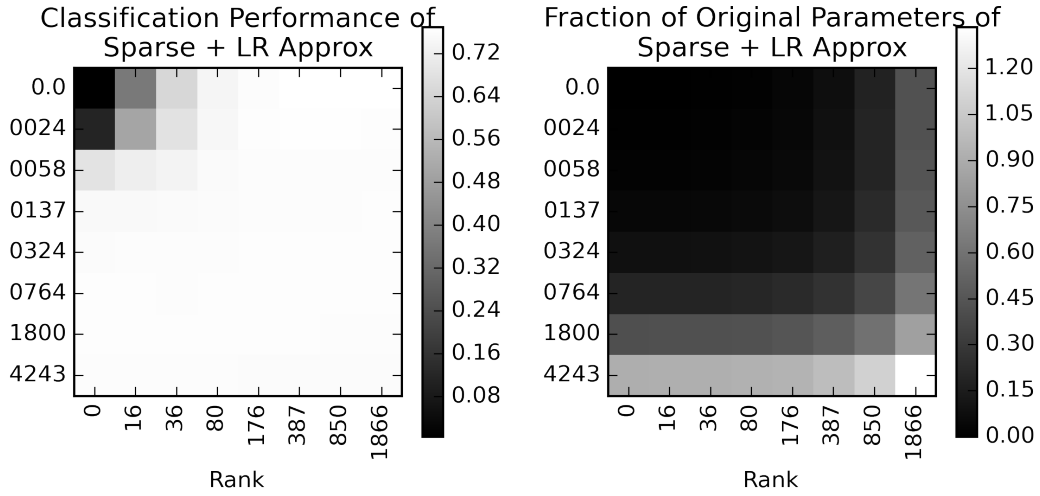

Figure 4. Left: top-5 classification accuracy as a function of number of parameters used in the W_{sp} and the rank of W_{LR}. Lighter squares indicate higher performance. Note the log-scale used on either axis. Right: for the same network configurations, the fraction of original parameters that remain in the network. Darker squares indicate a lower fraction of parameters remaining. For not-very-sparse or low-rank combinations, the approximation can actually require more parameters than the original matrix, hence the range of the scale. Parameter reduction works surprisingly well.

lar reduction in parameters while incurring an even smaller loss of performance, but it is important to note that the real-world memory footprint is likely to be higher than the parameter count suggests (because of the need to store the indices of the sparse entries). Most computer hardware is also not nearly as amenable to working with a sparse matrix representation as it is with a dense one. The low-rank factorization avoids these problems as it simply consists of two dense matrices.

In these experiments, I focused on studying the consequences of the low-rank matrix factorization studied in the first experiment. While both forms of parameter reduction are effective, rank reduction clearly exposes a low-dimensional feature space that the network operates in: all of the classification power of the network must be flow through the R-dimensional bottleneck between the left and right singular vectors.

I first performed whitening on the extracted features

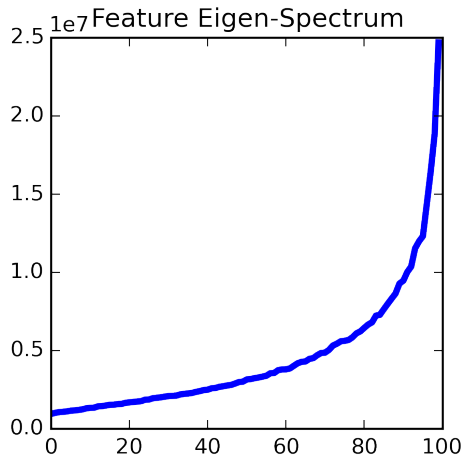

Figure 5. Eigenvalue spectrum of mean-centered F. There is a single large eigenvalue which happens to be in the same direction as the mean.

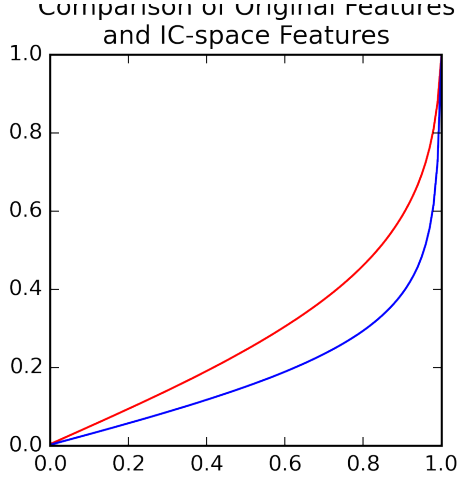

Figure 6. Scaled mean-sorted-absolute weight profiles for whitened F and for F_{IC}. ICA seems to have increased the sparsity of the encoding.

F. The eigenspectrum of a mean-centered version of F is shown in Figure 5. There is one very strong eigenvalue which also aligns with the mean of the data. I am not sure what to make of this, but I include the observation for completeness. It could be that there is simply a set of outliers very far from the rest of the data. Walking along this eigenvector and visualizing the KNN images did not result in any epiphanies.

After performing the concensus of FastICAs approach, I visualized $C = 91$ independent components obtained. For almost all of the ICs, the nearest neighors of one of the directions I_i or $-I_i$ had clear common properties while the other direction's nearest neighbors seemed to have nothing in common. I manually decided which direction was correct (and picked some of my favorite ICs) to create Figure 2. The complete visualization of all ICs is included in the supplementary material. The visualizations in the supplemental differ slightly from those in Figure 2 because the supplemental comes from "concensus of FastICAs" while Figure 2 comes from a single run (the figure was generated before I switched to the new approach). I also used a different length for the IC vectors for each figure when computing KNN images. Fortunately, the visualized image types are fairly robust to each of these effects: you should be able to match each image in Figure 2 with a row in the supplemental visualization. The supplemental visualizations are also annotated with the [(network's classification)(correct class)] for each image. Interestingly, the common properties of a given IC's closest images aren't associated with a single class: images from many different classes (both according to the network's predictions and the actual labels) are present in each set of KNN images.

The sparsity of F_{IC} as compared to that of F by the process described in methods is shown in Figure 6. As expected, the ICs form a basis that leads to a more sparse reprentation, suggesting that some unmixing must have been performed.

Some examples of synthesizing images from their IC components is shown in Figure 7 The reader is encouraged to run the code provided in the supplemental (it's fun!). In the case of synthesis, remember that the visualization is based on KNN of items that are in the dataset. Images corresponding to some given combination of ICs may simply not exist in the dataset. When decomposing images, the two largest independent components usually made a lot of sense with respect to the image. For example, a necktie with a very fine regular pattern, partially curled, was composed mostly of the "snake/squiggle" IC as well as the "rough texture" IC.

5. Conclusion

In this report, I began by studying the effect on classification performance of reducing the number of parameters in one of the final FC layers. Both the low-rank and sparse approximations of the weight matrix (or a combination of those two approaches) proved very effective, sacrificing little classification performance while allowing for a great deal of memory savings. The low-rank approximation has the interesting corollary of exposing a relatively low-dimensional feature space that must encodes the network's representation of each image. My focus then turned towards trying to understand this encoding. Without any particular hypotheses about how the information is encoded in the

Figure 7. Synthesis of images with IC vectors. Top: the "black and white" IC is added to the "human faces" IC to produce a vector whose KNNs are all black and white portraits. Bottom: the same "black and white" vector is added to one of the "birds" vectors to produce a vector whose first nearest neighbor is a black and white picture of a bird.

space, PCA and ICA offer two ways of getting some handle on the structure of the extracted features. Visualizing the independent components yielded directions in the feature space that seemed to encode high level qualities that images of several different classes may possess. At this point, I made a simple hypothesis about how the encoding might be done in something resembling a binary fashion in order to guide my experiments. Unfortunately, I don't think I was able to prove anything too concrete. The most solid finding to come out of this second line of investigation is probably simply that the ICs of the feature space are significant in some way.

There are a number of things that I would have liked to have spent more time on: specifically, I would have liked to have produced more quantitative metrics to explain the low-dimensional feature space.. Understanding how ICA was transforming the feature space is probably one of the higher priorities. The ICs produced by FastICA are not always the most intuitive (spending some time with ICA in 2- or 3-dimensions on sythetic data can lead to head-scratching). While the "F_{CI} is more sparse than F" result does something to suggest that ICA might be working as expected, it's far from conclusive. The "IC vectors vs cluster directions analysis" that I attempted was supposed to put some quantitative metric on how ICA might be working in one particular respect, but both versions of the analysis rest on shaky assumptions and I didn't trust any of the results in the end. Another good test for the "binary code" hypothesis would have been to see to what extent F_{IC} occupies a single hyper-quadrant. Under the hypothesis, a negative IC vector is not supposed to have any particular meaning (although if the set if ICs was an overcomplete basis for the space–not the case in my experiments–it could probably be useful).

References

[1] M. Denil, B. Shakibi, L. Dinh, and N. D. Freitas. Predicting Parameters in Deep Learning. *Advances in Neural ...*, pages 1–9, 2013.

[2] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science (New York, N.Y.)*, 313:504–507, 2006.

[3] A. Hyvärinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10:626–634, 1999.

[4] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up Convolutional Neural Networks with Low Rank Expansions. *arXiv preprint arXiv:1405.3866*, 2014.

[5] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, pages 1–9, 2012.

[7] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. Technical report, 2015.

[8] I. B. M. T. J. Watson and Y. Heights. LOW-RANK MATRIX FACTORIZATION FOR DEEP NEURAL NETWORK TRAINING WITH HIGH-DIMENSIONAL OUTPUT TARGETS Tara N . Sainath , Brian Kingsbury , Vikas Sindhwani , Ebru Arisoy , Bhuvana Ramabhadran { tsainath , bedk , vsindhw , earisoy , bhuvana } @ us . ibm . com. pages 6655–6659, 2013.