Fine-Pruning: Joint Fine-Tuning and Compression of a Convolutional Network with Bayesian Optimization

Frederick Tung
ftung@sfu.ca
Srikanth Muralidharan
smuralid@sfu.ca
Greg Mori
mori@cs.sfu.ca

School of Computing Science
Simon Fraser University
Burnaby, BC, Canada

Abstract

When approaching a novel visual recognition problem in a specialized image domain, a common strategy is to start with a pre-trained deep neural network and fine-tune it to the specialized domain. If the target domain covers a smaller visual space than the source domain used for pre-training (e.g. ImageNet), the fine-tuned network is likely to be over-parameterized. However, applying network pruning as a post-processing step to reduce the memory requirements has drawbacks: fine-tuning and pruning are performed independently; pruning parameters are set once and cannot adapt over time; and the highly parameterized nature of state-of-the-art pruning methods make it prohibitive to manually search the pruning parameter space for deep networks, leading to coarse approximations. We propose a principled method for jointly fine-tuning and compressing a pre-trained convolutional network that overcomes these limitations. Experiments on two specialized image domains (remote sensing images and describable textures) demonstrate the validity of the proposed approach.

1 Introduction

Convolutional neural networks (CNNs) have been widely adopted for visual analysis tasks such as image classification [14, 26], object detection [17, 22], action recognition [25, 29], place recognition [1, 35], 3D shape classification [12, 19], image colorization [34], and camera pose estimation [13]. CNNs learn rich image and video representations that have been shown to generalize well across vision tasks.

When faced with a recognition task in a novel domain or application, a common strategy is to start with a CNN pre-trained on a large dataset, such as ImageNet [24], and fine-tune the network to the new task (Fig. 1a). Fine-tuning involves adapting the structure of the existing network to enable the new task, while re-using the pre-trained weights for the unmodified layers of the network. For example, a common and simple form of fine-tuning involves replacing the final fully-connected layer of the pre-trained CNN, which has an output dimensionality based on the pre-training dataset (e.g. 1000 dimensions for ImageNet), with a new fully-connected layer with a dimensionality that matches the target dataset.

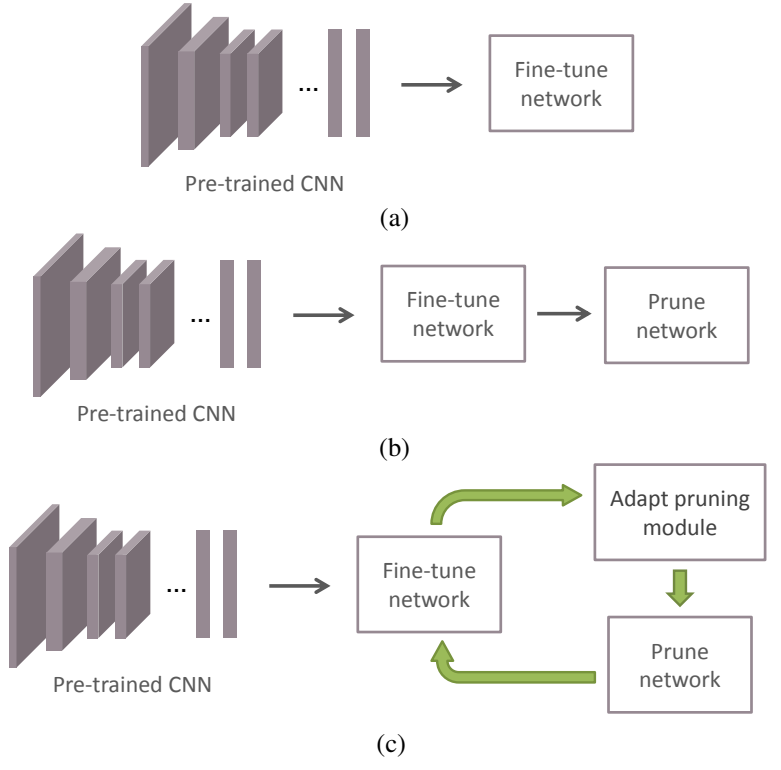


Figure 1: Consider the task of training a deep convolutional neural network on a specialized image domain (e.g. remote sensing images). (a) The conventional approach starts with a network pre-trained on a large, generic dataset (e.g. ImageNet) and fine-tunes it to the specialized domain. (b) Since the specialized domain spans a narrower visual space, the fine-tuned network is likely to be over-parameterized and can be compressed. A natural way to achieve this is to perform network pruning after fine-tuning, however this approach has limitations (see discussion in Section 1). (c) Fine-pruning addresses these limitations by jointly fine-tuning and compressing the pre-trained network in an iterative process. Each iteration consists of network fine-tuning, pruning module adaptation, and network pruning.

Fine-tuning allows powerful learned representations to be transferred to novel domains. Typically, we fine-tune complex network architectures that have been pre-trained on large databases containing millions of images. For example, we may fine-tune AlexNet [14] pre-trained on ImageNet's 1.2 million images (61 million parameters). In this way, we adapt these complex architectures to smaller and more specialized domains, such as remote sensing images. However, the specialized domain may not span the full space of natural images on which the original network was pre-trained. This suggests that the network architecture may be over-parameterized, and therefore inefficient in terms of memory and power consumption, with respect to the more constrained novel domain, in which a much more lightweight network would suffice for good performance. In applications with tight constraints on memory and power, such as mobile devices or robots, a more lightweight network with comparable classification accuracy may be valuable.

Given a fine-tuned network, a straightforward way to obtain a more lightweight network is to perform network pruning [6, 8, 28] (Fig. 1b). However, this strategy has drawbacks: (1) the fine-tuning and pruning operations are performed independently; (2) the pruning parameters are set once and cannot adapt after training has started; and (3) since state-of-the-art pruning methods are highly parameterized, manually searching for good pruning hyperparameters is often prohibitive for deep networks, leading to coarse pruning strategies (e.g. pruning convolutional and fully connected layers separately [6]).

We propose a novel process called *fine-pruning* (Fig. 1c) that addresses these limitations:

1. Fine-pruning obtains a lightweight network specialized to a target domain by jointly fine-tuning and compressing the pre-trained network. The compatibility between the target domain and the pre-training domain is not normally known in advance (e.g. how similar are remote sensing images to ImageNet?), making it difficult to determine a priori how effective knowledge transfer will be, how aggressively compression can be applied, and where compression efforts should be focused. The knowledge transfer and network compression processes are linked and inform each other in fine-pruning.

2. Fine-pruning applies a principled adaptive network pruning strategy guided by Bayesian optimization, which automatically adapts the layer-wise pruning parameters over time as the network changes. For example, the Bayesian optimization controller might learn and execute a gradual pruning strategy in which network pruning is performed conservatively and fine-tuning restores the original accuracy in each iteration; or the controller might learn to prune aggressively at the outset and reduce the compression in later iterations (e.g. by splicing connections [6]) to recover accuracy.

3. Bayesian optimization enables efficient exploration of the pruning hyperparameter space, allowing all layers in the network to be considered together when making pruning decisions.

2 Related Work

Network pruning. Network pruning refers to the process of reducing the number of weights (connections) in a pre-trained neural network. The motivation behind this process is to make neural networks more compact and energy efficient for operation on resource constrained devices such as mobile phones. Network pruning can also improve network generalization by reducing overfitting. The earliest methods [9, 16] prune weights based on the second-order derivatives of the network loss. Data-free parameter pruning [28] provides a data-independent method for discovering and removing entire neurons from the network. Deep compression [8] integrates the complementary techniques of weight pruning, scalar quantization to encode the remaining weights with fewer bits, and Huffman coding. Dynamic network surgergy [6] iteratively prunes and splices network weights. The novel splicing operation allows previously pruned weights to be reintroduced. Weights are pruned or spliced based on thresholding their absolute value. All weights, including pruned ones, are updated during backpropagation.

Other network compression strategies. Network pruning is one way to approach neural network compression. Other effective strategies include weight binarization [9, 21], architectural improvements [11], weight quantization [8], sparsity constraints [15, 36], guided knowledge distillation [10, 23], and replacement of fully connected layers with structured

projections [7, 18, 33]. Many of these network compression methods can train compact neural networks from scratch, or compress pre-trained networks for testing in the same domain. However, since they assume particular types of weights, mimic networks trained in the same domain, or modify the network structure, most of these methods are not easily extended to the task of fine-tuning a pre-trained network to a specialized domain.

In this paper, we consider joint fine-tuning and network pruning in the context of transferring the knowledge of a pre-trained network to a smaller and more specialized visual recognition task. Previous approaches for compressing pre-trained neural networks aim to produce a compact network that performs as well as the original network *on the dataset on which the network was originally trained*. In contrast, our focus is on the fine-tuning or *transfer learning* problem of producing a compact network for a small, specialized target dataset, given a network pre-trained on a large, generic dataset such as ImageNet. Our approach does not require the source dataset (e.g. ImageNet) on which the original network was trained.

3 Method

Each fine-pruning iteration comprises three steps: fine-tuning, adaptation of the pruning module, and network pruning (Fig. 1c). Fine-pruning can accommodate any parameterized network pruning module. In our experiments, we use the state-of-the-art dynamic network surgery method [6] for the network pruning module, but fine-pruning does not assume a particular pruning method. Pruning module adaptation is guided by a Bayesian optimization [5, 27] controller, which enables an efficient search of the joint pruning parameter space, learning from the outcomes of previous exploration. This controller allows the pruning behaviour to change over time as connections are removed or formed.

Bayesian optimization is a general framework for solving global minimization problems involving blackbox objective functions:

$$\min_{\mathbf{x}} \ell(\mathbf{x}), \tag{1}$$

where ℓ is a blackbox objective function that is typically expensive to evaluate, non-convex, may not be expressed in closed form, and may not be easily differentiable [30]. Eq. 1 is minimized by constructing a probabilistic model for ℓ to determine the most promising candidate $\mathbf{x}^*$ to evaluate next. Each iteration of Bayesian optimization involves selecting the most promising candidate $\mathbf{x}^*$, evaluating $\ell(\mathbf{x}^*)$, and using the data pair $(\mathbf{x}^*, \ell(\mathbf{x}^*))$ to update the probabilistic model for ℓ.

In our case, $\mathbf{x}$ is a set of pruning parameters. For example, if the network pruning module is deep compression [8], $\mathbf{x}$ consists of the magnitude thresholds used to remove weights; if the network pruning module is dynamic network surgery [6], $\mathbf{x}$ consists of magnitude thresholds as well as cooling function hyperparameters that control how often the pruning mask is updated. We define ℓ by

$$\ell(\mathbf{x}) = \varepsilon(\mathbf{x}) - \lambda \cdot s(\mathbf{x}), \tag{2}$$

where $\varepsilon(\mathbf{x})$ is the top-1 error on the held-out validation set obtained by pruning the network according to the parameters $\mathbf{x}$ and then fine-tuning; $s(\mathbf{x})$ is the sparsity (proportion of pruned connections) of the pruned network obtained using the parameters $\mathbf{x}$; and λ is an importance weight that balances accuracy and sparsity, which is set by held-out validation (we set λ

Algorithm 1 Fine-Pruning

Require: Pre-trained convolutional network, importance weight λ

```
Fine-tune network {▷ Fig. 1a}
repeat
    repeat {▷ Bayesian optimization controller}
        Select next candidate parameters to evaluate as x* = arg max_x̂ EI(x̂)
        Evaluate ℓ(x*)
        Update Gaussian process model using (x*, ℓ(x*))
    until converged or maximum iterations of Bayesian optimization reached
    Prune network using best x* found
    Fine-tune network
until converged or maximum iterations of fine-pruning reached
```

to maximize the achieved compression rate while maintaining the held-out validation error within a tolerance percentage, e.g. 2%).

We model the objective function as a Gaussian process [20]. A Gaussian process is an uncountable set of random variables, any finite subset of which is jointly Gaussian. Let $\ell \sim \mathcal{GP}(\mu(\cdot), k(\cdot,\cdot))$, where $\mu(\cdot)$ is a mean function and $k(\cdot,\cdot)$ is a covariance kernel such that

$$\begin{aligned} \mu(\mathbf{x}) &= \mathbb{E}\left[\ell(\mathbf{x})\right], \\ k(\mathbf{x},\mathbf{x}') &= \mathbb{E}\left[(\ell(\mathbf{x})-\mu(\mathbf{x}))(\ell(\mathbf{x}')-\mu(\mathbf{x}'))\right]. \end{aligned} \tag{3}$$

Given inputs $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ and function evaluations $\ell(\mathbf{X}) = \{\ell(\mathbf{x}_1), \ell(\mathbf{x}_2), ..., \ell(\mathbf{x}_n)\}$, the posterior belief of ℓ at a novel candidate $\hat{\mathbf{x}}$ can be computed in closed form. In particular,

$$\tilde{\ell}(\hat{\mathbf{x}}) \sim \mathcal{N}\left(\tilde{\mu}_\ell(\hat{\mathbf{x}}), \tilde{\Sigma}^2_\ell(\hat{\mathbf{x}})\right), \tag{4}$$

where

$$\begin{aligned} \tilde{\mu}_\ell(\hat{\mathbf{x}}) &= \mu(\hat{\mathbf{x}}) + k(\hat{\mathbf{x}},\mathbf{X})k(\mathbf{X},\mathbf{X})^{-1}(\ell(\mathbf{X})-\mu(\mathbf{X})), \\ \tilde{\Sigma}^2_\ell(\hat{\mathbf{x}}) &= k(\hat{\mathbf{x}},\hat{\mathbf{x}}) - k(\hat{\mathbf{x}},\mathbf{X})k(\mathbf{X},\mathbf{X})^{-1}k(\mathbf{X},\hat{\mathbf{x}}). \end{aligned} \tag{5}$$

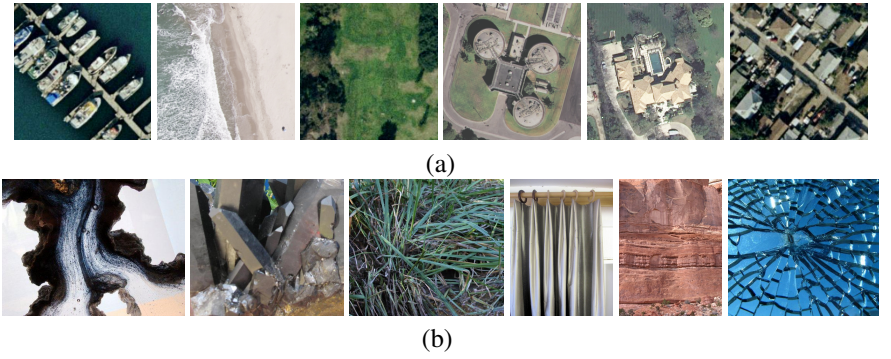
The implication of the closed form solution is that, given a collection of parameters and the objective function evaluated at those parameters, we can efficiently predict the posterior at unevaluated parameters.

To select the most promising candidate to evaluate next, we use the expected improvement criterion. Let $\mathbf{x}^+$ denote the best candidate evaluated so far. The expected improvement of a candidate $\hat{\mathbf{x}}$ is defined as

$$\mathrm{EI}(\hat{\mathbf{x}}) = \mathbb{E}\left[\max\left\{0, \ell(\mathbf{x}^+) - \tilde{\ell}(\hat{\mathbf{x}})\right\}\right], \tag{6}$$

For a Gaussian process, the expected improvement of a candidate can also be efficiently computed in closed form. Specifically,

$$\begin{aligned} \mathrm{EI}(\hat{\mathbf{x}}) &= \tilde{\Sigma}_\ell(\hat{\mathbf{x}})(Z\Phi(Z) + \phi(Z)), \\ Z &= \frac{\tilde{\mu}_\ell(\hat{\mathbf{x}}) - \ell(\mathbf{x}^+)}{\tilde{\Sigma}_\ell(\hat{\mathbf{x}})}, \end{aligned} \tag{7}$$



(a)

(b)

Figure 2: Sample images from the two specialized domain datasets used in our experiments: (a) Remote sensing images from the UCMerced Land Use Dataset [32]; (b) Texture images from the Describable Textures Dataset [3].

where Φ is the standard normal cumulative distribution function and ϕ is the standard normal probability density function. For a more detailed discussion on Gaussian processes and Bayesian optimization, we refer the interested reader to [5], [20], and [27]. We use the publicly available code of [5] and [20] in our implementation.

The complete fine-pruning process is summarized in Algorithm 1.

4 Experiments

Datasets. We performed experiments on two specialized image domains:

- Remote sensing images: The UCMerced Land Use Dataset [32] is composed of public domain aerial orthoimagery from the United States Geological Survey. The dataset covers 21 land-use classes, such as agricultural, dense residential, golf course, and harbor. Each land-use class is represented by 100 images. We randomly split the images into 50% for training, 25% for held-out validation, and 25% for testing.

- Describable textures: The Describable Textures Dataset [3] was introduced as part of a study in estimating human-describable texture attributes from images, which can then be used to improve tasks such as material recognition and description. The dataset consists of 5,640 images covering 47 human-describable texture attributes, such as blotchy, cracked, crystalline, fibrous, and pleated. We use the ten provided training, held-out validation, and testing splits.

Fig. 2 shows examples of images from the two datasets.

Baselines. We compare fine-pruning with a fine-tuning only baseline (Fig. 1a) as well as independent fine-tuning followed by pruning (Fig. 1b), which for brevity we will refer to as the independent baseline. All experiments start from an ImageNet-pretrained AlexNet [14]. For a controlled comparison, we run the same state-of-the-art pruning method, dynamic network surgery [6], in both the independent baseline and fine-pruning. In the original dynamic network surgery paper [6], the authors prune convolutional and fully connected layers separately due to the prohibitive complexity of manually searching for layer-wise pruning parameters.

	Accuracy (Val.)	Accuracy (Test)	Parameters	Compression Rate
UCMerced Land Use Dataset [32]				
Fine-tuning only (Fig. 1a)	94.7%	94.3%	57.0 M	–
Independent fine-tuning and pruning (Fig. 1b)	92.7±0.7%	93.8±0.7%	1.78±0.41 M	31.9 ×
Fine-pruning (Fig. 1c)	92.5±0.9%	94.1±0.6%	1.17±0.39 M	**48.8** ×
Describable Textures Dataset [3]				
Fine-tuning only (Fig. 1a)	53.5±0.8%	53.7±0.9%	57.1 M	–
Independent fine-tuning and pruning (Fig. 1b)	52.8±1.2%	53.4±1.5%	3.62±0.54 M	15.8 ×
Fine-pruning (Fig. 1c)	53.0±0.9%	52.8±0.8%	2.41±0.68 M	**23.7** ×

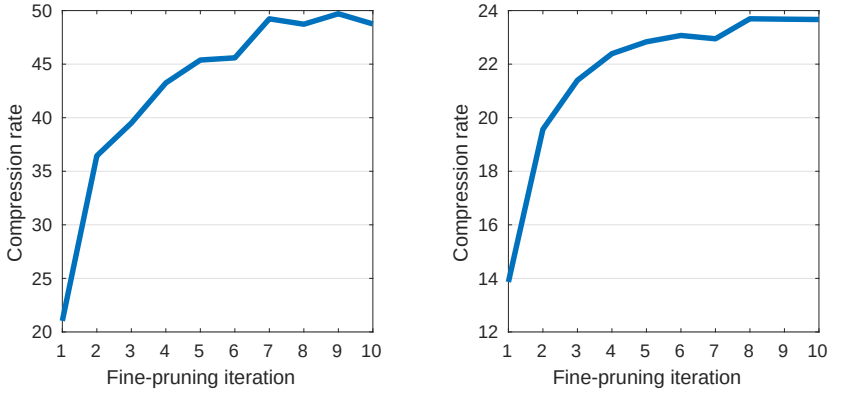
Table 1: Experimental results on two specialized image domains: remote sensing images and describable textures. All experiments start with ImageNet-pretrained AlexNet [14] and use the state-of-the-art dynamic network surgery method [6] for network pruning. For a fair comparison, the pruning parameters in the independent fine-tuning and pruning baseline are also tuned by Bayesian optimization. We average results over ten runs on the remote sensing dataset and the ten provided splits on the describable textures dataset.

To more fairly illustrate the benefit of fine-pruning, we set layer-wise pruning parameters for dynamic network surgery in the independent baseline using Bayesian optimization as well.

Implementation details. We set all parameters by held-out validation on the two datasets. The importance weight λ is set to 1 on both datasets. We warm-start both the independent baseline and fine-pruning with identical parameters obtained by random search. Fine-pruning is run to convergence or to a maximum of 10 iterations. In each fine-pruning iteration, Bayesian optimization considers up to 50 candidates and network fine-tuning is performed with a fixed learning rate of 0.001 (the same learning policy used to obtain the initial fine-tuned network) to 10 epochs.

Results. Table 1 summarizes our experimental comparison of fine-tuning, independent fine-tuning and pruning, and fine-pruning, on the UCMerced Land Use and Describable Textures datasets. On UCMerced Land Use, the independent baseline produces sparse networks with 1.78 million parameters on average over ten runs, representing a reduction in the number of weights by 31.9-fold, while maintaining the test accuracy within 1% of the dense fine-tuned network. Fine-pruning achieves further improvements in memory efficiency, producing sparse networks of 1.17 million parameters on average, or a 48.8-fold reduction in the number of weights, while maintaining the test accuracy within 1% of the dense fine-tuned network. On Describable Textures, we average the results over the ten provided splits. Similar improvements are obtained on this harder dataset. The independent baseline reduces the number of weights by 15.8-fold while maintaining the test accuracy within 1% of the dense fine-tuned network. Fine-pruning lifts the compression rate to 23.7-fold while maintaining the test accuracy within 1% of the dense fine-tuned network.

Fig. 3 shows how the compression rate varies with the fine-pruning iteration. We observe



(a) (b)

Figure 3: Compression as a function of fine-pruning iteration. On both the (a) UCMerced Land Use Dataset and (b) Describable Textures Dataset, the pruning module adaptation, guided by Bayesian optimization, learns a policy of starting with a strong initial prune and tapering off in later iterations.

that, on both datasets, the pruning module adaptation learns to start with a strong initial prune and then gradually increase pruning aggressiveness in later iterations until the network converges. This behavior can also be observed by examining the pruning parameters $\mathbf{x}^*$ selected by Bayesian optimization.

Table 2 illustrates the average number of weights layer by layer after fine-pruning for both datasets. We observe that the original fine-tuned networks in both cases are highly over-parameterized, and a significant reduction in memory can be obtained by fine-pruning. A large proportion of the original network parameters reside in the fully connected layers fc6 and fc7. Provided that the underlying network pruning module allows for pruning parameters to be set on an individual layer basis, our Bayesian optimization controller automatically learns to prioritize the compression of these layers because they have the largest influence on $s(\mathbf{x})$ in the objective function (Eq. 2).

5 Conclusion

In this paper we have presented a joint process for network fine-tuning and compression that produces a memory-efficient network tailored to a specialized image domain. Our process is guided by a Bayesian optimization controller that allows pruning parameters to adapt over time to the characteristics of the changing network. Fine-pruning is general and can accommodate any parameterized network pruning algorithm. In future we plan to study whether our technique can be applied to provide better time efficiency as well. For example, structured sparsity may result in more significant time savings on a GPU than unstructured sparsity [30]. Specialized hardware engines [7] can also accelerate networks with unstructured sparsity while reducing energy consumption.

Acknowledgements. This work was supported by the Natural Sciences and Engineering Research Council of Canada.

	Parameters: Before	Parameters: After	Percentage Pruned
UCMerced Land Use Dataset [32]			
conv1	35 K	26 K	26.1%
conv2	307 K	92 K	70.2%
conv3	885 K	261 K	70.5%
conv4	664 K	218 K	67.2%
conv5	443 K	181 K	59.1%
fc6	37.8 M	313 K	99.2%
fc7	16.8 M	60 K	99.6%
fc8	86 K	17 K	80.3%
total	57.0 M	1.17 M	98.0%
Describable Textures Dataset [3]			
conv1	35 K	32 K	8.3%
conv2	307 K	245 K	20.4%
conv3	885 K	343 K	61.3%
conv4	664 K	442 K	33.4%
conv5	443 K	216 K	51.2%
fc6	37.8 M	401 K	98.9%
fc7	16.8 M	661 K	96.1%
fc8	193 K	72 K	62.4%
total	57.1 M	2.41 M	95.8%

Table 2: Layer-wise compression results. Our Bayesian optimization controller automatically learns to prioritize the compression of the fc6 and fc7 layers, which have the most parameters.

References

[1] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[2] Y. Cheng, F. X. Yu, R. S. Feris, S. Kumar, A. Choudhary, and S.-F. Chang. An exploration of parameter redundancy in deep networks with circulant projections. In *IEEE International Conference on Computer Vision*, 2015.

[3] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[4] M. Courbariaux, Y. Bengio, and J.-P. David. BinaryConnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*, 2015.

[5] J. R. Gardner, M. J. Kusner, Z. Xu, K. Q. Weinberger, and J. P. Cunningham. Bayesian optimization with inequality constraints. In *International Conference on Machine Learning*, 2014.

[6] Y. Guo, A. Yao, and Y. Chen. Dynamic network surgery for efficient DNNs. In *Advances in Neural Information Processing Systems*, 2016.

[7] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally. EIE: Efficient inference engine on compressed deep neural network. In *ACM/IEEE International Symposium on Computer Architecture*, 2016.

[8] S. Han, H. Mao, and W. J. Dally. Deep Compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In *International Conference on Learning Representations*, 2016.

[9] B. Hassibi and D. G. Stork. Second order derivatives for network pruning: optimal brain surgeon. In *Advances in Neural Information Processing Systems*, 1992.

[10] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. arXiv:1503.02531, 2015.

[11] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. arXiv:1602.07360, 2016.

[12] E. Johns, S. Leutenegger, and A. J. Davison. Pairwise decomposition of image sequences for active multi-view recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[13] A. Kendall, M. Grimes, and R. Cipolla. PoseNet: A convolutional network for real-time 6-DOF camera relocalization. In *IEEE International Conference on Computer Vision*, 2015.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

[15] V. Lebedev and V. Lempitsky. Fast ConvNets using group-wise brain damage. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[16] Y. LeCun, J. S. Denker, and S. A. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*, 1990.

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *European Conference on Computer Vision*, 2016.

[18] M. Moczulski, M. Denil, J. Appleyard, and N. de Freitas. ACDC: A structured efficient linear layer. In *International Conference on Learning Representations*, 2016.

[19] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view CNNs for object classification on 3D data. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[20] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[21] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. XNOR-Net: ImageNet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, 2016.

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[23] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. FitNets: hints for thin deep nets. In *International Conference on Learning Representations*, 2015.

[24] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. arXiv:1409.0575, 2014.

[25] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, 2014.

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[27] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, 2012.

[28] S. Srinivas and R. V. Babu. Data-free parameter pruning for deep neural networks. In *British Machine Vision Conference*, 2015.

[29] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. In *IEEE International Conference on Computer Vision*, 2015.

[30] Z. Wang, M. Zoghi, F. Hutter, D. Matheson, and N. de Freitas. Bayesian optimization in high dimensions via random embeddings. In *International Joint Conference on Artificial Intelligence*, 2013.

[31] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, 2016.

[32] Y. Yang and S. Newsam. Bag-of-visual-words and spatial extensions for land-use classification. In *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2010.

[33] Z. Yang, M. Moczulski, M. Denil, N. de Freitas, A. Smola, L. Song, and Z. Wang. Deep fried convnets. In *IEEE International Conference on Computer Vision*, 2015.

[34] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *European Conference on Computer Vision*, 2016.

[35] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems*, 2014.

[36] H. Zhou, J. M. Alvarez, and F. Porikli. Less is more: towards compact CNNs. In *European Conference on Computer Vision*, 2016.