A Bayesian Optimization Framework for Neural Network Compression

Xingchen Ma*[a], Amal Rannen Triki * †[a], Maxim Berman[a], Christos Sagonas[b], Jacques Cali ‡[c], and Matthew B. Blaschko[a]

[a]KU Leuven
[b]Onfido
[c]Blue Prism

Abstract

Neural network compression is an important step for deploying neural networks where speed is of high importance, or on devices with limited memory. It is necessary to tune compression parameters in order to achieve the desired trade-off between size and performance. This is often done by optimizing the loss on a validation set of data, which should be large enough to approximate the true risk and therefore yield sufficient generalization ability. However, using a full validation set can be computationally expensive. In this work, we develop a general Bayesian optimization framework for optimizing functions that are computed based on U-statistics. We propagate Gaussian uncertainties from the statistics through the Bayesian optimization framework yielding a method that gives a probabilistic approximation certificate of the result. We then apply this to parameter selection in neural network compression. Compression objectives that can be written as U-statistics are typically based on empirical risk and knowledge distillation for deep discriminative models. We demonstrate our method on VGG and ResNet models, and the resulting system can find optimal compression parameters for relatively high-dimensional parametrizations in a matter of minutes on a standard desktop machine, orders of magnitude faster than competing methods.

1. Introduction

Neural networks have had an explosion in practical applications in the past six years. Often, the larger the network, the higher the performance. This means that during development, neural networks are often constructed to fill available computational resources in a high performance computing setting. However, at the time of deployment, lower power usage, or cheaper computational components mean there is an imperative to achieve the same performance at reduced computational cost. This has led to the growing field of neural network compression [15, 10].

In this work, we present a Bayesian optimization (BO) framework [2] for neural network compression with several novel contributions that assist in the speed and accuracy of the compression process:
1. Neural network compression with BO using novel objectives that allows for fast approximate evaluation of the quality criterion. This enables a faster search through the parameter space of a compression algorithm. Based on U-statistics, we can show that finite sample estimators of these quality measures have Gaussian distribution, making it compatible to propagate their uncertainty through the Gaussian Process (GP) model used in BO.
2. A novel acquisition function for BO ([2, 44]) that directly optimizes a probabilistic approximation criterion incorporating our measurement of uncertainty resulting from efficient, subsampled quality measures. We show that this acquisition function improves computational performance, has natural convergence parameters that are easy to set, and is a compelling drop-in replacement for the main acquisition functions that are currently employed in BO.

We demonstrate our framework in two settings: (i) Supervised compression minimizing risk; and (ii) a knowledge distillation framework where a compressed network is trained to produce similar outputs to the uncompressed output. In each setting, we demonstrate that our framework can compute probably (under the probabilistic model used in BO) approximately optimal compression in a matter of minutes on real-world neural networks including ResNet18, ResNet50, and VGG-16 (Section 4).

*Authors with equal contribution
†This author is currently affiliated with Deepmind.
‡Contribution to this research project was entirely made while this co-author was at Onfido, UK.

2. Related work

Network compression has gained increased interest in recent years with the growth of the need for on-device computation. Applications for compression are numerous, such as IoT, self driving cars, and edge computing. Multiple approaches to compression have been developed, but they can generally be divided into two groups: *parameter number reduction* and *storage size reduction* methods.

In the first category of techniques, the goal is to reduce the number of non-zero parameters in the network with minimum loss of performance, such as via weight pruning. In [11], the authors propose to first train the network, remove the weights that are smaller than a given threshold, and then retrain the reduced network. [27] considers a Bayesian sparsity enforcing prior on the weights. Then, the variational parameters of a parametric approximate posterior distribution on the weights are optimized according to the evidence-lower-bound. The approximate posterior is subsequently used to prune the neurons that are necessary for training. In [53], the weights to prune are selected according to their impact on the second to last layer before classification, measured using feature ranking techniques.

Low-rank decomposition is another efficient method to reduce the number of parameters of a deep neural network. Compression is achieved by replacing the parameters of the convolutional and fully connected layers with their low-rank approximations. A straightforward decomposition-based way of compressing a layer (e.g., fully connected) is by applying Singular Value Decomposition (SVD) to the weight matrix [6]. By building on the idea that a filter can be approximated as a linear combination of a smaller number of separable ones [39], [17] propose to decompose the full rank filters as combination of rank-1 filter basis. Similarly, in [24] a convolution is approximated as a composition of four convolutions with small kernels produced by employing a Canonical Polyadic (CP) decomposition. A shortcoming of the aforementioned method is that finding the best low-rank approximation is an ill-posed problem. To overcome this, [46] propose a low-rank decomposition that always exists and admits a closed form solution. Although the previous methods achieved good compression rates without sacrificing accuracy on different target problems, they were applied only in shallow networks, without taking into account the statistics of layer activations. One of the first attempts of applying low-rank decomposition in deeper networks trained on large datasets is presented in [55]. To this end, a new asymmetric decomposition method that incorporates feature map reconstruction is introduced. Recently, [54], inspired by Robust Principal Components Analysis [3], propose to use the feature map reconstruction error to approximate the weight matrix as a linear combination of a low-rank and sparse error matrices. Likewise, [28] address the problem of domain adaptive compression by proposing a rank-constrained regression problem which admits a closed-form solution. [35] proposes to use low-rank decomposition on groups of filters in order to reduce the size and computation complexity of the network, and apply it to deep models. Finally, some works combine both pruning or sparsity methods and low rank decomposition. In addition to [54], in a more recent work, [7] compress convolutional networks by first pruning then selecting a coreset of components to retrain in a data-based fashion, reducing the amount of needed retraining.

Size reduction methods quantize the weights and reduce their precision in order to reduce memory usage. The goal is to find the lowest precision level at which the performance of the network is preserved. This operation can also allow weight sharing, as qunatization can have the effect of increasing the number of equal weights. [10] combines quantization with pruning and Huffman coding for more efficiency. The authors also use weight sharing, but fine-tune the shared weights to counter the reduced capacity. [50] uses a version of soft-weight sharing introduced in [33] to achieve both quantization and pruning in a single retraining procedure. [27] also tackles quantization along with pruning by injecting noise to the model, pruning the required precision from the posterior. More recently, some works propose to use an adaptive quantization. [20] introduces a layer-wise precision selection by minimizing the change in the loss function, while [36] proposes to train directly a smaller model using distillation loss, where the precision of the weights is limited to a given set of levels.

Model selection in most of the cited works is done by grid search using a held out validation set. While it is easy to implement, such a search is very costly in time and resources. Moreover, this limit the search to a a very small number parameters for the whole model, as having a threshold, a target rank or a bit-width per layer would make this search unfeasible for most of the recent architectures. More recently, the problem of efficient model selection has been considered. [48] introduced the use of BO for the problem of model compression. The authors define the objective function to minimize a weighted difference between the error on a validation set and the weight sparsity of the network. The expected improvement is used as acquisition function. As the computation of the validation error is expensive, this method can fail to scale efficiently to larger networks. More recently, [4] uses BO with an application aware objective and expected improvement as acquisition function. While the definition of the constraints from the application reduces the number of hyperparameters to set, this methods still relies on a previously fixed number of iterations for BO and lacks an adaptive stopping criterion.

Bayesian optimization (BO) is an optimization framework based on continually updating a probabilistic model with measurements of a function to be optimized. Given a

set of parameters to be optimized, BO makes black-box calls to the objective, updates the probabilistic model with the new information, and selects the next point to evaluate using an acquisition function that combines information about the expectation and uncertainty of a function value under the probabilistic model [2, 8, 23]. The employed model is usually a GP [38] due to its favorable statistical and computational characteristics, and currently defines the state of the art for black box optimization. Although GPs can model Gaussian observation noise in a relatively straightforward manner, most available BO packages do not implement the ability to model varying noise in the measurement of the objective function [47, 29, 44, 5, 49]. As a result, it is not straightforward to use noisy approximations to the objective function while modeling a process that converges to the optimum of a noiseless procedure. The computational bottleneck of using a full estimate of the validation error remains.

This paper is based on the observation that natural objectives for neural network compression each can be expressed as a quantity whose minimum-variance unbiased estimator is a U-statistic [16, 41]. This includes subsampled estimates of validation error and knowledge distillation based training (Sec. 3.1) [15]. Results from mathematical statistics guarantee an asymptotic Gaussian distribution of a finite sample estimate of these quantities, with known variance [41]. This variance term can in turn be included in a GP model used in BO for model selection. In doing so, we enable highly efficient [1] selection of probably approximately optimal (under the probabilistic model used in BO) compression parameters in much higher dimensions than is computationally feasible using grid search or BO with a full validation set.

3. Neural network compression with Bayesian optimization

Let us consider the problem of neural network compression. Given a neural network f mapping an input space $\mathcal{X}$ to an output space $\mathcal{Y}$, a compression procedure is a functional that transforms f to $\tilde{f}_\theta$ that has a smaller size or smaller number number of parameters. The hyperparameter vector θ typically determines how small the resulting network is. For example, it can be a threshold when the compression is done by a pruning method, or a rank when it is done by an SVD or other low-rank-decomposition based method.

The training of the original network usually aims to minimize the risk $\mathcal{R}(f) = \mathbb{E}_{(x,y)\sim P}\left[\ell(f(x), y)\right]$. As the true risk is inaccessible, it is approximated by the empirical risk

$$\hat{\mathcal{R}}_n(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i), y_i) \tag{1}$$

[1] We emphasize that the increase of speed we are interested in in this work is for parameter selection. Other works consider inference speed [9], and are implemented in an orthogonal manner to our approach.

on a training set $\{(x_i, y_i)\}_{i=1\dots n} \subset \mathcal{X} \times \mathcal{Y}$, an i.i.d. sample drawn from the distribution $P(x, y)$. We suppose that the network to be compressed f^* generalizes well, and that we are interested in finding a compressed version $\tilde{f}_\theta$ that performs similarly to f^*.

The problem that we consider in this work is to select the compression hyperparameters θ. BO has been applied to hyperparameter search in machine learning [18, 44], but to our knowledge is only beginning to be explored for parameter search for neural network compression, e.g. [48]. In order to be able to apply the BO framework, we need first to define the objective that we want to maximize.

In a compression procedure, two quantities are of interest: (i) **The quality of the compressed network**: ultimately, one would aim to obtain a network that generalizes as well as the original one. As the true risk is inaccessible, we need a function that measures the quality of the compressed network $\tilde{f}_\theta$ either in terms of performance for the task of interest, or in terms of fidelity of the compressed network's outputs to the original network's outputs. In the rest of the paper, we refer to these measures as quality functions, denoted either $\mathcal{Q}(\tilde{f}_\theta)$ for a measure of performance, and $\mathcal{L}(\tilde{f}_\theta, f^*)$ for a measure of fidelity. (ii) **The size of the obtained network**: compression aims at minimizing the size of the compressed network relative to the original one. We consider here two measures: the compression ratio, $R(\tilde{f}_\theta, f^*)$, denoting the ratio of the number of parameters or the size of the compressed network divided by the number of parameters or the size of the original network, or its inverse, the compression rate that we will simply denote $R(\tilde{f}_\theta, f^*)^{-1}$. The hyperparameter selection problem can be formalized in two optimization problems. Depending on the target application, one would want either to maximize the quality given a target compression rate, or to maximize the compression rate given a target quality. Using Lagrange multipliers γ or κ, and without loss of generality, these constrained optimization problems can be written as follows

$$\arg\max_\theta \underbrace{\gamma\mathcal{Q}(\tilde{f}_\theta) + R(\tilde{f}_\theta, f^*)^{-1}}_{J_\mathcal{Q}(\theta)} \tag{2}$$

$$\text{or, } \arg\min_\theta \underbrace{\kappa\mathcal{L}(\tilde{f}_\theta, f^*) + R(\tilde{f}_\theta, f^*)}_{J_\mathcal{L}(\theta)}. \tag{3}$$

In the following paragraphs, we will first discuss the choice of the quality or fidelity measurement, and then introduce a new acquisition function for BO in order to find a probably approximately optimal solution to the considered objective.

3.1. Fidelity measures

In this section, we consider two variants of fidelity measures: (i) risk, and (ii) a student-teacher (or knowledge distillation) strategy [15]. These objectives enable compression in settings that where (i) a fully supervised training set is

available, and (ii) a set of unlabeled images (or a generative model) is available.

In supervised learning, a natural measure of quality is the empirical risk (Eq. (1)), which presumes a sufficiently large labeled set of data to closely approximate the true risk. Even if such a set of data is available, use of a full empirical risk estimate in the inner loop of an optimization procedure is typically computationally infeasible. This underlies the popularity of stochastic gradient descent based methods in neural network optimization. In the case of neural network compression, the compression algorithm itself may involve non-continuous, non-differentiable operations (e.g. thresholding weights to zero), meaning gradient based methods are inapplicable. Nevertheless, within the BO framework, we may consider multiple subsampled estimates $\hat{\mathcal{R}}_m(f)$ where $n \gg m$, and n is the total number of available samples in the training set. Even in the special case that $m = 1$, this forms an unbiased (albeit high-variance) estimator of the true (non-empirical) risk $\mathcal{R}(f)$. Furthermore, from the theory of U-statistics, finite sample estimates $\hat{\mathcal{R}}_m(f)$ have asymptotic Gaussian distribution whose standard deviation can be well estimated by the empirical standard deviation of $\{\ell(f(x_i, y_i))\}_{i=1\ldots m}$ divided by $\sqrt{m}$ [16, 41].

A student-teacher strategy [15] is perhaps favorable to risk in neural network compression as labeled data are not required. In principle, only samples from the marginal distribution $P(x)$ are required. This could be achieved by drawing a sufficiently large finite sample of data, replacing the true marginal distribution by a bootstrap approximation, approximating the marginal distribution by a GAN or VAE, etc. In practice, the latter strategies enable access to an unlimited number of independent samples from an approximation of the marginal distribution. The knowledge distillation framework then minimizes

$$\mathbb{E}_{x\sim P}[\ell(\tilde{f}_\theta(x), f^*(x))] \tag{4}$$

for some loss function ℓ. If we consider the special case that ℓ is the squared Euclidean loss, (4) is the square of a weighted L_2 function norm [12, 37]

$$\mathcal{L}(\tilde{f}_\theta, f^*) := \mathbb{E}_{x\sim P}(\|\tilde{f}_\theta(x) - f^*(x)\|_2^2) = \|f^* - \tilde{f}_\theta\|_{2,P}^2 \tag{5}$$

and it follows $\mathcal{L}(\tilde{f}_\theta, f^*)$ defines the canonical metric for this function norm. For Lipschitz continuous losses (most of the commonly used losses in neural network training, including cross-entropy loss), this immediately yields a bound on the generalization error of $\tilde{f}_\theta$ as a function of the generalization error of f^* monotonic in $\mathcal{L}(\tilde{f}_\theta, f^*)$. [2] Thus, minimizing $\mathcal{L}(\tilde{f}_\theta, f^*)$ also controls the generalization error of $\tilde{f}_\theta$. As in the case of risk, the result is a first order U-statistic and finite sample estimates have Gaussian distribution, which can be calculated using the same formula as developed for empirical risk above.

In this section, we have developed two fidelity measures. These are all natural to consider in the context of function minimization. As BO is frequently expressed in the context of maximization, we can consider $\mathcal{Q}(\tilde{f}_\theta) = -\mathcal{L}(\tilde{f}_\theta, f^*)$, and we will analyze the optimization of Eq. (2) in the sequel.

3.2. A novel acquisition function for Bayesian optimization with uncertain observations

We now consider the general problem of maximizing an objective $J(\theta)$ using BO when $J(\theta)$ can only be observed approximately with Gaussian noise, as in the case that $J(\theta)$ can be estimated by the sum of an analytic expression and a U-statistic as considered in the previous section. We develop here a novel acquisition function for BO, as well as a convergence criterion that terminates optimization when the probability of being within a certain distance to the global optimum exceeds a threshold (see Eq. (6)).

BO is a principled framework for black-box optimization that applies a surrogate function with quantified uncertainty to estimate regions of a search space that should be explored. The framework then continually evaluates a more expensive function (Eq. (2)) to be optimized at these points. The most popular variants employ a GP to approximate the true function, which gives uncertainty estimation in closed form, and currently defines the state of the art for black-box optimization without gradient information [2, 8].

Currently, BO uses several acquisition functions, such as the probability of improvement (PI), expected improvement (EI), lower- and upper- confidence bounds, entropy search, and knowledge gradient [2, 8]. Probability of improvement, proposed by in [23], is the most classically studied acquisition function and computes $\arg\max_\theta \frac{\mu_\theta - J(\hat{\theta}) - \xi}{\sigma_\theta}$ for an exploration-exploitation trade-off parameter $\xi > 0$, where μ_θ and σ_θ are the mean and standard deviation of θ under the GP, and $\hat{\theta}$ is the parameter with the highest function value observed so far [2, Sec. 2.3.1]. Several strategies for modifying the acquisition function in the case of noisy observation of $J(\theta)$ have been proposed including $\arg\max_\theta \frac{\mu_\theta - \mu_{\hat{\theta}} - \xi}{\sqrt{c\sigma_{\hat{\theta}}^2 + \sigma_{\hat{\theta}}^2}}$ for some parameter $c > 0$ [23, Eq. (18)] or imputing the acquisition function derived in the noiseless case [8, Sec. 5]. In the sequel, we derive a more natural solution that incorporates noise uncertainty, and also lends an alternate interpretation to the ξ parameter introduced in PI and EI.

We propose to optimize our objective to find a solution $\hat{\theta}$ with the semantics

$$p(J(\theta^*) - J(\hat{\theta}) < \delta) \geq 1 - \varepsilon \tag{6}$$

for user supplied δ and ε,[3] where θ^* is the global optimizer of J. This criterion closely mirrors the PAC semantics in-

[2] See the supplementary material for an explicit derivation of this bound.

[3] In practice, ε can be fixed to some small value, e.g. 10^{-3}, while δ can

troduced by Valiant [51], but in an optimization setting and where the notion of probability is the Bayesian posterior probability under the Gaussian Process model: p is defined by the most recent estimate of the GP surrogate.

As a GP model is employed in BO, $J(\theta^*) - J(\hat{\theta}) - \delta$ is a Gaussian distributed random variable. Furthermore, we may derive (cf. [1, Equation (10)]) that the probability of this variable being negative (i.e. $\hat{\theta}$ is approximately correct) is equal to

$$p(J(\theta^*) - J(\hat{\theta}) < \delta) = 1 - \mathbf{\Phi}\left(\frac{\mu_{\theta^*} - \mu_{\hat{\theta}} - \delta}{\sqrt{\sigma^2_{\theta^*} + \sigma^2_{\hat{\theta}} - 2\sigma_{\hat{\theta}\theta^*}}}\right), \tag{7}$$

where $\mathbf{\Phi}$ is the CDF of a standard normal variable, μ_θ is the mean of the GP model evaluated at θ, σ^2_θ is the variance of the GP model at θ, and $\sigma_{\hat{\theta}\theta^*}$ is the covariance of the GP between points $\hat{\theta}$ and θ^*. The means and (co-)variances conditioned on the observations are (cf. [38, Equations (2.22)-(2.24)]):

$$\mu_\theta = K(\theta, \Theta)[K(\Theta, \Theta) + \Sigma_n]^{-1}\mathbf{y}, \tag{8}$$

$$\sigma^2_\theta = k(\theta, \theta) - K(\theta, \Theta)[K(\Theta, \Theta) + \Sigma_n]^{-1}K(\Theta, \theta), \tag{9}$$

$$\sigma_{\hat{\theta}\theta^*} = k(\hat{\theta}, \theta^*) - K(\hat{\theta}, \Theta)[K(\Theta, \Theta) + \Sigma_n]^{-1}K(\Theta, \theta^*). \tag{10}$$

where K is the covariance matrix, Θ is the set of previously observed points, Σ_n is the noise matrix,[4] $\mathbf{y}$ is the vector of noisy targets and k is the kernel used in the GP.

We take $\hat{\theta}$ to be the value with the highest mean under the GP of a point we have visited so far. Our acquisition function will fix this value, and search for an estimate of θ^* that minimizes (7). The next point is given by

$$A_{\text{PAC}}(\text{GP}, \hat{\theta}) = \arg\max_{\tilde{\theta}} \frac{\mu_{\tilde{\theta}} - \mu_{\hat{\theta}} - \delta}{\sqrt{\sigma^2_{\hat{\theta}} + \sigma^2_{\tilde{\theta}} - 2\sigma_{\hat{\theta}\tilde{\theta}}}}. \tag{11}$$

Each of these quantities can be determined with low computational cost under the GP model (Eqs. (8)-(10)). Once $\theta_{i+1} := A_{\text{PAC}}(\text{GP}, \hat{\theta})$ has been computed, we either have found a new point θ_{i+1} to (approximately) evaluate or we can compute a probabilistic approximation certificate of $\hat{\theta}$ by showing that

$$1 - \varepsilon \leq 1 - \mathbf{\Phi}\left(\frac{\mu_{\theta_{i+1}} - \mu_{\hat{\theta}} - \delta}{\sqrt{\sigma^2_{\theta_{i+1}} + \sigma^2_{\hat{\theta}} - 2\sigma_{\hat{\theta}\theta_{i+1}}}}\right). \tag{12}$$

be fixed to a constant or set at each iteration to a small percentage of the current best candidate $\mu_{\hat{\theta}}$, probabilistically guaranteeing that the returned value is optimal to some percentage of the global optimum.

[4] In the case that $J(\theta)$ is estimated with independent samples, this will be a diagonal matrix with variances determined by the U-statistic estimator.

The right-hand-side of (12) is bounded above by (7) implying that (6) holds whenever (12) is true. This convergence criterion is nicely interpretable, in contrast to other proposed criteria, such as a fixed number of iterations or $\|\theta_i - \theta_{i+1}\| \leq \varepsilon$ [26]. If we specialize our acquisition function to the case that $\sigma^2_{\hat{\theta}} = \sigma_{\hat{\theta}\tilde{\theta}} = 0$, we recover the probability of improvement (PI). In the special case of PI, the analogous stopping criterion has been analyzed from the perspective of regret minimization showing favorable properties [32]. We also note that Eq. (11) can naturally be plugged in as the normal distribution used in the construction of the expected improvement acquisition function [31, 19]. Even if a different acquisition function is preferred, Eq. (11) can be employed periodically to test for convergence to an optimum satisfying the semantics described in Equation (6).

3.3. Parameter encoding

State of the art compression algorithms typically combine low-rank decompositions [6, 28, 46] and sparsity [11, 27]. Each of these parameters naturally have a range from zero to a maximal rank of the matrix/tensor in the case of low-rank decomposition, or zero sparsity to some maximal value encoding a complete sparsification of the network (i.e. setting all values to zero). In each case, the parameter range of the ith parameter can be encoded in some known range $[0, i_{\max}]$. It is useful to re-normalize each of these coordinates, and the BO procedure will optimize over the domain $\theta \in [0, 1]^d$ for a d-dimensional parametrization. As our framework scales favorably to higher-dimensional parametrizations of compression algorithms, this enables e.g. choosing a different rank parameter per layer of the network (cf. Table 1).

The rank parameters used in SVD and tensor decomposition methods are integers, in our experiments, we use a scaling scheme. First, in order to make the compression really work, we should constrain the rank parameter used in the compression algorithm. For SVD, we need that the compressed layer has smaller number of weights compared with the original layer. The original number of weights in a layer is HW, and the number in the compressed layer is $HK + KW = (H + W)K$, so this maximum rank is $\frac{HW}{H+W}$, where H and W are height and width for the matrix we are compressing. The number of parameters can be reduced further to $HK + KW - \binom{K}{2}$, by the fact that there exists an equivalent decomposition in which the second matrix is upper triangular [52, Eq. (5.4.1)]. For tensor decomposition methods, this maximum rank is $\frac{kC_iC_o}{C_i+C_o}$, where k is the convolutional kernel height (we assume the width is the same as its height), C_i and C_o are input and output channels respectively. Then, before compressing the model, we transform the parameters to lie in $[0, 1]$ using the maximum ranks per layer computed in the first step.

Table 1. Number of compression parameters in each model.

Model	FC3	VGG-16	ResNet18	ResNet50
Parameters	3	16	9	16

4. Experiments

In this section, we demonstrate the speed of our method compared with state-of-the-art optimization in Section 4.1. We then demonstrate that knowledge distillation provides computational advantages with no decrease in accuracy (Tables 3 and 4, and Sec. 4.2). In Sec. 4.3 we demonstrate that our approach matches state-of-the-art methods in the literature in terms of compression performance, and in Sec. 4.4 we show that the optimality properties of our optimization strategy can lead to enlightening meta-analysis.

We evaluate our proposed method on several representative networks, including a 3-layer fully-connected network (FC3), ResNet18, ResNet50 [13] and VGG-16 [42]. The FC3 has a network structure of *784-1000-1000-10* and we train it using the MNIST dataset [25] from scratch. The pre-trained ResNet18, ResNet50 and VGG-16 are downloaded from the model zoo in PyTorch [34]. In the case of compressing FC3, we apply SVD to the fully connected layers as proposed in [6]. In the case of compressing ResNet18, ResNet50, we apply the tensor decomposition proposed in [46] on the convolutional layers. In the case of compressing VGG-16, we apply tensor decomposition and SVD on the convolutional layers and fully-connected layers, respectively. For ResNets, we only compress convolutional layers with a kernel size 3×3. The number of parameters in each compression approach is listed in Table 1. For all experiments, we set the BO convergence parameters to $\delta = 0.05$ and $\varepsilon = 0.005$, and the GP used a Matern 5/2 kernel [30] with parameters set by maximum likelihood. To obtain the function norm, we consistently use 50 as our sampling size, as it balances the computation cost and the requirements of a GP model, details could be found in the supplementary material.

In all following tables, the columns show the compression ratio, top-1 and top-5 accuracy rates, and the time needed before our algorithm converges. We report the mean of 10 different runs for a specific γ. The top-1 and top-5 accuracies are computed using the full validation set of the corresponding dataset: for FC3, we use 10000 testing images from MNIST, for ResNet and VGG, we use 50000 validation images from ILSVRC2012.

4.1. Comparison of different model selection methods on Resnet18

For this experiment, we consider compression Resnet18. We compare our approach with the "Fabolas" method [22], which compares favorably (time and performance) with techniques such as Entropy Search [14], EI [31], and multi-task BO [45]. We also compare to random search, which can be competitive in low dimensional search spaces. Table. 2 shows the results. Not only we are much faster than Fabolas, Fig. 1 shows we Pareto-dominate the other methods w.r.t. top-1 accuracy on the full validation set. Under the null hypothesis that the performance of of our method is worse than Fabolas, a Wilcoxon signed-rank test gives a p-value equals to 0.0008, with a significance level of $\alpha = 0.05$, thus we reject the null hypothesis.

Table 2. Compression of ResNet18 using different methods

γ	Random			Fabolas			Ours		
	ratio (%)	top1 (%)	time (s)	ratio (%)	top1 (%)	time (s)	ratio (%)	top1 (%)	time (s)
0.8	32.1	56.8	1264	30.2	60.4	39316	31.2	60.2	1120
0.9	33.9	59.8	1351	30.7	61.6	38249	34.2	62.2	1277
1.0	34.9	58.2	1126	32.7	61.8	46876	34.3	62.9	1378
1.1	30.8	55.5	1329	32.1	61.1	35159	35.9	63.5	1660
1.2	41.7	61.2	1017	34.0	63.0	42296	35.2	63.4	1420

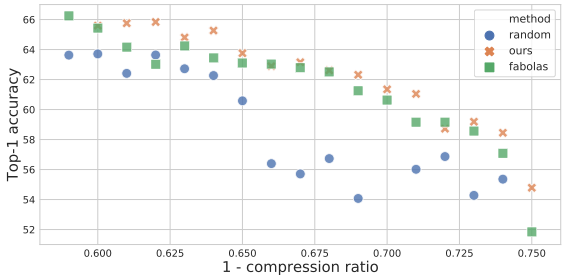


Figure 1. Ratio vs top1 accuracy for different methods

4.2. Knowledge distillation as a proxy for risk

A natural question is whether the knowledge distillation objective with L_2 loss (Eq. (5)) is a good proxy for risk in network compression. Figs. 2 and 3 show the relationship between the estimated norm and top-1 error rate. The top-1 error rate is obtained using 5000 random validation images from ILSVRC2012 [40] and the norm is estimated using 1000 random samples from the corresponding training set. We also estimate the norm in different layers, and show these results in different columns. For instance, in the first column of Fig. 2, if the estimated norm is larger than 50, the top-1 error rate is nearly 1.0, which means after compression, the information in the original model is completely lost. We see that the relationship between the top-1 error and function norm is monotonic, and even close to linear under this threshold. This justifies our use of the function norm as a fidelity criterion for compression of deep neural networks.

For further validation, we also compared the compression results with the knowledge distillation objective (Table 3) and with the risk objective(Table 4) on FC3. The obtained results show that using the function norm has comparable performance to the case of using the top-1 error rate as the fidelity term, with advantages of taking less time and requir-

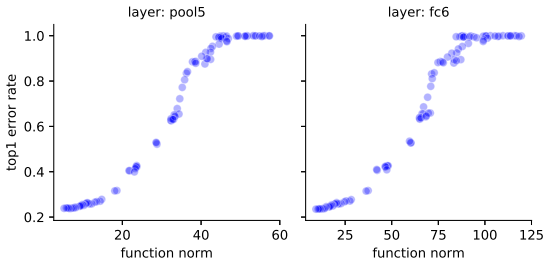


Figure 2. Estimated norm vs. Top-1 error in ResNet50.

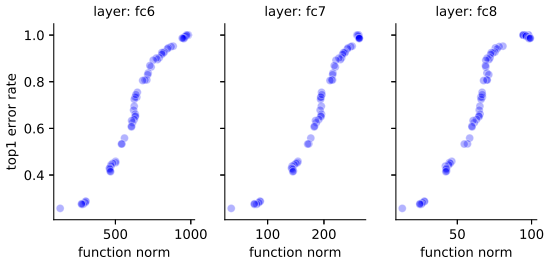


Figure 3. Estimated norm vs. Top-1 error in VGG-16.

ing no labeled data. This further improves our confidence in using the function norm to compress a pre-trained model.

Table 3. Compression on FC3 using SVD and the knowledge distillation fidelity term. *#W* is the number of parameters in the compressed layers.

γ	#W	ratio (%)	top-1 (%)	time (s)
0.001	0.05M	2.62	67.97	10.33
0.002	0.05M	2.62	67.99	9.25
0.005	0.06M	3.09	76.22	13.73
0.010	0.21M	11.96	97.80	71.19
0.020	0.36M	20.12	97.78	79.54
original	1.79M	100	98.20	

Table 4. Compression on FC3 using SVD and the top-1 error. *#W* is the number of parameters in the compressed layers.

γ	#W	ratio (%)	top-1 (%)	time (s)
0.9	0.16M	8.69	97.74	181.17
1.0	0.17M	9.23	97.73	378.65
1.1	0.20M	11.20	97.73	365.53
original	1.79M	100	98.20	

4.3. Compression of VGG-16

In this section, we demonstrate that our method finds compression parameters that compare favorably to state-of-the-art compression results reported on VGG-16 [10]. We first apply our method to compress convolutional layers of VGG-16 using tensor decomposition, which has 13 parameters. After that, we fine-tune the compressed model for 5 epochs, using Stochastic Gradient Descent (SGD) with momentum 0.9 and learning rate 1e-4, decreased by a factor of 10 every epoch. Second, we apply another pass of our algorithm to compress the fully-connected layers of the fine-tuned model using SVD, which has 3 parameters. A single optimization takes approximately 10 minutes. Again, after the compression, we fine-tune the compressed model, and we use SGD with momentum as our optimizer. We use cyclical learning rates [43] as our learning rate policy, with a learning rate range from 1e-7 to 1e-4. Finally, we apply our method to this fine-tuned model to apply one last pruning (without fine-tuning).

An optimal γ for a target compression ratio is selected according to the dual objective function (details in the supplementary material). Top-1 and top-5 accuracy for different compression stages are given in Table 5. Compared with the pruned model in [10], the evaluation of our SVD compressed model is is more than $2\times$ faster for fc6, comparable for fc7 and more than $20\times$ faster for fc8.

Table 5. Top-1 and top-5 accuracies (%) of compressed VGG-16, and compression ratio (%). See Sec. 4.3 for details.

Network	top-1	top-5	ratio
original	68.50	88.68	100
tensor decomposition (td)	69.11	88.69	-
td + svd	68.69	88.41	9.31
td + svd + pruning	68.16	88.15	7.4
Han et al. [10]	68.66	89.12	7.5

4.4. Analysis of the γ parameter

In this section we empirically analyze the role of the compression-vs.-accuracy parameter γ in Eq. 2 and how this parameter influences the compression ratio, the total time, and the top-1 and top-5 accuracy rates. Tables 6 and 7 show compression results for ResNet18 and ResNet50, respectively. It is clear from these tables that the overall compression ratio monotonically increases with γ. When compressing ResNet18 and ResNet50, if γ is larger than 0.0005, the top-1 and top-5 accuracy rate of the compressed models are quite similar to the case that $\gamma = 0.0005$, except the total time needed is longer. If γ is too small, the top-1 and top-5 accuracy rates are too low to be acceptable for any applications, so we do not report results for $\gamma < 0.00006$. This phenomenon also exists in compressing FC3 (cf. Table 3). From these tables, we conclude there exists a reasonable range for γ to compress a specific model, with expected compression-accuracy trade-off behavior.

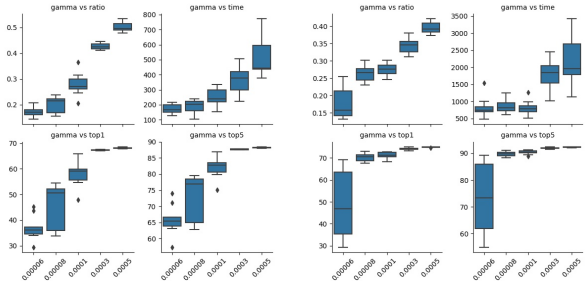
Tables 6 and 7 only show the performance for the compressed model before fine-tuning. After several epochs of fine-tuning (constraining the rank to be equal to the compressed model and therefore not increasing the size of the model), the performance can increase substantially. For example, in Sec. 4.3, after the first compression without

Table 6. Compression on ResNet18 using low-rank decomposition. *#W* is the number of parameters in the compressed layers.

γ	#W	ratio	top-1	top-5	time
0.00006	1.08M	17.19	36.81	65.72	172.0
0.00008	1.26M	20.13	45.49	72.68	188.9
0.0001	1.75M	27.90	58.18	82.16	248.1
0.0003	2.67M	42.65	67.33	87.61	365.6
0.0005	3.15M	50.21	68.16	88.15	515.0
original	6.27M	100	69.76	89.08	

Table 7. Compression on ResNet50 using low-rank decomposition. *#W* is the number of parameters in the compressed layers.

γ	#W	ratio	top-1	top-5	time
0.00006	2.03M	17.96	48.93	73.51	815.4
0.00008	2.99M	26.45	70.16	89.73	855.3
0.0001	3.12M	27.60	71.20	90.33	804.6
0.0003	3.89M	34.34	74.16	91.96	1802.6
0.0005	4.48M	39.55	74.83	92.28	2204.9
original	11.32M	100	76.13	92.86	



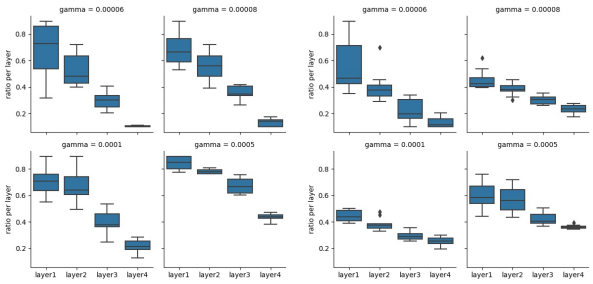
(a) ResNet18 (b) ResNet50

Figure 4. Compression ratio, total time, top-1 accuracy rate, top-5 accuracy rate for different γ and different models.

fine-tuning, the top-1 and top-5 accuracies dropped 8.64% and 5.67% compared to the original performance, and after 5 epoches of fine-tuning, the accuracies increased from 59.86% to 69.11% for top-1, and increased from 83.01% to 88.69% for top-5. Similarity, after the second fine-tuning, the top-1 accuracy increased from 65.41% to 68.69% and the top-5 accuracy increased from 87.17% to 88.41%.

Incorporating fine-tuning into the compression framework increases overall accuracy at the expense of significant additional computation. This can be done either as a post-processing step on the optimal model (faster), as in Sec. 4.3, or as part of the compression function to generate $\tilde{f}_\theta$ in the inner loop of the BO. We leave further study of these strategies to future work.

In Fig. 4, we show a graphical comparison of the performance with different settings of γ. For instance, in Fig. 4(b), we observe that when γ is small, there is a greater variability and more outliers for the compression ratio, and top-1 and top-5 accuracy rates. We can interpret this result as follows:



(a) ResNet18 (b) ResNet50

Figure 5. Per-block compression for different γ.

The objective function in Eq. 2 is a trade-off between the compression ratio and the fidelity term. When γ is small, the importance of the fidelity term is small, thus allowing it to vary greatly without significantly affecting the objective function. The time needed for BO to converge under the PAC criterion grows with γ; this is expected, as when γ is large, the optimization will work harder to find good parameters to make the norm term small.

We analyze the compression statistics for different blocks in ResNet. The box plots in Figs. 5(a) and 5(b) show the per-block compression ratios for ResNet18 and ResNet50, respectively. Each figure has 4 different γs and every box is calculated using 10 different runs. It appears that the compression ratio decreases with the depth of the block, which indicates deeper blocks have more redundant information. This is consistent with reported results on compressing convolutional layers in [21, 6, 11]. A second observation is that deeper blocks have smaller variability in the compression ratio than shallow blocks, potentially indicating stable patterns encoded by each of the different optimization runs.

5. Conclusion

In this work, we have developed a principled, fast, and flexible framework for optimizing neural network compression parameters. We have demonstrated its utility on a range of state-of-the-art neural network models, including multiple ResNet architectures, and in two different settings with variable amounts of supervision during the compression step: (i) a fully supervised optimization using empirical risk and (ii) a setting in with only unlabeled samples based on the knowledge distillation framework. In all settings, the framework achieves optimal solutions in minutes, orders of magnitude faster than competing methods. Software will be released at the time of publication.

Acknowledgements

X.M. and M.B.B. receive support from Onfido. A.R.T., M.B., and M.B.B. acknowledge support from FWO (grant G0A2716N), an Amazon Research Award, an NVIDIA GPU grant, and the Facebook AI Research Partnership.

References

[1] Wacha Bounliphone, Eugene Belilovsky, Matthew B. Blaschko, Ioannis Antonoglou, and Arthur Gretton. A test of relative similarity for model selection in generative models. In *Proceedings of the International Conference on Learning Representations*, 2016. 5

[2] Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *CoRR*, abs/1012.2599, 2010. 1, 3, 4

[3] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011. 2

[4] Changan Chen, Frederick Tung, Naveen Vedula, and Greg Mori. Constraint-aware deep neural network compression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 400–415, 2018. 2

[5] Antoine Cully, Konstantinos Chatzilygeroudis, Federico Allocati, and Jean-Baptiste Mouret. Limbo: A Flexible High-performance Library for Gaussian Processes modeling and Data-Efficient Optimization. *The Journal of Open Source Software*, 3(26):545, 2018. 3

[6] Emily Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. *Advances in Neural Information Processing Systems*, 2014. 2, 5, 6, 8

[7] Abhimanyu Dubey, Moitreya Chatterjee, and Narendra Ahuja. Coreset-based neural network compression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 454–470, 2018. 2

[8] Peter I. Frazier. A tutorial on Bayesian optimization. *CoRR*, abs/1807.02811, 2018. 3, 4

[9] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. Eie: efficient inference engine on compressed deep neural network. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 243–254. IEEE, 2016. 3

[10] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In *Proceedings of the International Conference on Learning Representations*, 2016. 1, 2, 7

[11] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. *Advances in Neural Information Processing Systems*, June 2015. 2, 5, 8

[12] Michiel Hazewinkel. Weighted space. In *Encyclopaedia of Mathematics*. Springer, 1987. 4

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 6

[14] Philipp Hennig and Christian J Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13(Jun):1809–1837, 2012. 6

[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *Advances in Neural Information Processing Systems, Deep Learning Workshop*, 2015. 1, 3, 4

[16] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. 3, 4

[17] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *British Machine Vision Conference*, 2014. 2

[18] Rodolphe Jenatton, Cedric Archambeau, Javier González, and Matthias Seeger. Bayesian optimization with tree-structured dependencies. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1655–1664, 2017. 3

[19] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998. 5

[20] Soroosh Khoram and Jing Li. Adaptive quantization of neural networks. 2018. 2

[21] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. In *Proceedings of the International Conference on Learning Representations*, 2016. 8

[22] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets. In *Artificial Intelligence and Statistics*, pages 528–536. 6

[23] Harold J Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106, 1964. 3, 4

[24] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned CP-decomposition. *Proceedings of the International Conference on Learning Representations*, 2014. 2

[25] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998. 6

[26] Romy Lorenz, Ricardo P Monti, Ines R Violante, Aldo A Faisal, Christoforos Anagnostopoulos, Robert Leech, and Giovanni Montana. Stopping criteria for boosting automatic experimental design using real-time fMRI with Bayesian optimization. *arXiv preprint arXiv:1511.07827*, 2015. 5

[27] Christos Louizos, Karen Ullrich, and Max Welling. Bayesian compression for deep learning. *Advances in Neural Information Processing Systems*, 2017. 2, 5

[28] Marc Masana, Joost van de Weijer, Luis Herranz, Andrew D Bagdanov, and Jose M Alvarez. Domain-adaptive deep network compression. In *Proceedings of the IEEE Conference on Computer Vision*, volume 16, pages 7370–7379, 2017. 2, 5

[29] MathWorks. MATLAB R2018b bayesopt function, 2018. Natick, MA, USA. 3

[30] Budiman Minasny and Alex. B. McBratney. The Matérn function as a general model for soil variograms. *Geoderma*, 128(3):192–207, 2005. 6

[31] Jonas Mockus. On Bayesian methods for seeking the extremum. In *Proceedings of the IFIP Technical Conference*, pages 400–404. Springer, 1974. 5, 6

[32] Vu Nguyen, Sunil Gupta, Santu Rana, Cheng Li, and Svetha Venkatesh. Regret for expected improvement over the best-observed value and stopping condition. In Min-Ling Zhang and Yung-Kyun Noh, editors, *Proceedings of the Ninth Asian Conference on Machine Learning*, volume 77 of *Proceedings of Machine Learning Research*, pages 279–294. PMLR, 15–17 Nov 2017. 5

[33] Steven J Nowlan and Geoffrey E Hinton. Simplifying neural networks by soft weight-sharing. *Neural computation*, 4(4):473–493, 1992. 2

[34] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *Proceedings of the International Conference on Learning Representations - Workshops*, 2017. 6

[35] Bo Peng, Wenming Tan, Zheyang Li, Shun Zhang, Di Xie, and Shiliang Pu. Extreme network compression via filter group approximation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 300–316, 2018. 2

[36] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018. 2

[37] Amal Rannen Triki and Matthew B. Blaschko. Function norms and regularization in deep networks. *arXiv:1605.09085v2*, 2016. 4

[38] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2005. 3, 5

[39] Roberto Rigamonti, Amos Sironi, Vincent Lepetit, and Pascal Fua. Learning separable filters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2754–2761, 2013. 2

[40] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 6

[41] Robert J. Serfling. *Approximation theorems of mathematical statistics*. Wiley, 1980. 3, 4

[42] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Proceedings of the International Conference on Learning Representations*, abs/1409.1556, 2014. 6

[43] Leslie N. Smith. Cyclical Learning Rates for Training Neural Networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472. 7

[44] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25*, pages 2951–2959, 2012. 1, 3

[45] Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. In *Advances in neural information processing systems*, pages 2004–2012, 2013. 6

[46] Cheng Tai, Tong Xiao, Xiaogang Wang, and E Weinan. Convolutional neural networks with low-rank regularization. *CoRR*, abs/1511.06067, 2016. 2, 5, 6

[47] The GPyOpt authors. GPyOpt: A Bayesian optimization framework in python. http://github.com/SheffieldML/GPyOpt, 2016. 3

[48] Frederick Tung, Srikanth Muralidharan, and Greg Mori. Fine-pruning: Joint fine-tuning and compression of a convolutional network with Bayesian optimization. In *British Machine Vision Conference*, 2017. 2, 3

[49] Tsuyoshi Ueno, Trevor David Rhone, Zhufeng Hou, Teruyasu Mizoguchi, and Koji Tsuda. COMBO: An efficient Bayesian optimization library for materials science. *Materials Discovery*, 4:18–21, 2016. 3

[50] Karen Ullrich, Edward Meeds, and Max Welling. Soft Weight-Sharing for Neural Network Compression. *Proceedings of the International Conference on Learning Representations*, Feb. 2017. 2

[51] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, Nov. 1984. 5

[52] Charles F Van Loan and Gene H Golub. *Matrix computations*. Johns Hopkins University Press, 3rd edition, 1996. 5

[53] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9194–9203, 2018. 2

[54] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7370–7379, 2017. 2

[55] Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun. Accelerating very deep convolutional networks for classification and detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):1943–1955, 2016. 2