Meta Compression: Learning to Compress Pre-trained Deep Neural Networks

Anonymous authors
Paper under double-blind review

Abstract

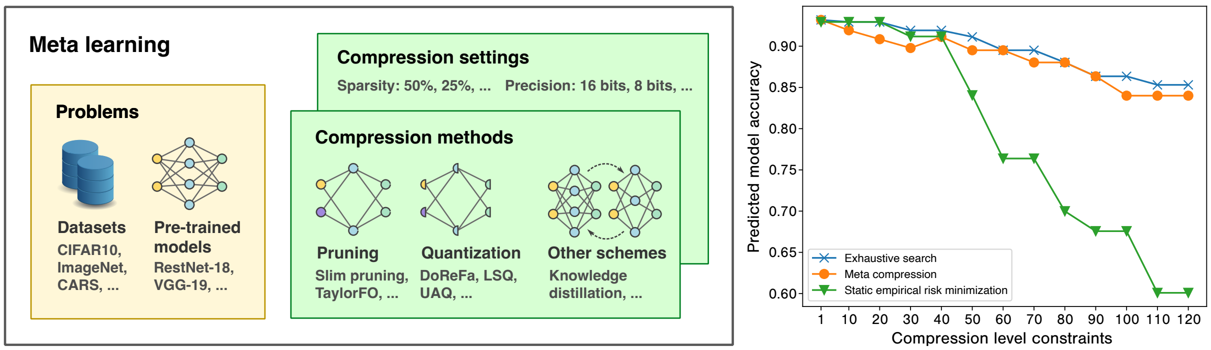
State-of-the-art deep neural networks (DNN) have achieved outstanding results in a variety of tasks. Unfortunately, these DNN so large that cannot fit into the limited resources of edge servers or end devices such as smartphones and IoT sensors. Several approaches have been proposed to design compact yet efficient DNNs, however, the performance of the compressed model can be only characterized a posteriori. This work addresses this issue by introducing meta compression, a novel approach based on meta learning to simplify a pre-trained DNN into one that fulfills given constraints on size or accuracy. We leverage diffusion-based generative models to improve generalization performance of meta learning and extensively evaluate meta compression on an image classification task with popular pre-trained DNNs. The obtained results show that meta compression achieves a 92% top-5 recommendation accuracy and that the top-1 recommendation is only 1% far from the optimal compression method in terms of average accuracy loss.

1 Introduction

Deep neural networks (DNNs) are being increasingly adopted in diverse domains including computer vision (Redmon et al., 2016), natural language processing (Vaswani et al., 2017), and speech recognition (Amodei et al., 2015). The size of state-of-the-art DNNs has exponentially grown in recent years (Desislavov et al., 2023), allowing them to achieve or even exceed human-level performance for a variety of tasks (Alzubaidi et al., 2021; Silver et al., 2018). Despite this unquestionable benefit, these DNNs have become so large that in some cases they cannot run on a single GPU. Smaller models cannot still fit the limited resources of edge servers or end devices such as mobile phones and smart objects in the Internet of things. This work specifically addresses such a challenge to enable efficient computation, closer to end users.

Several approaches have been proposed to design compact yet efficient DNNs – in terms of both accuracy and performance – by explicitly targeting resource-constrained devices, for instance. On the one hand, model compression takes an existing DNN and applies techniques such as pruning and (or) quantization to obtain a simplified model (Gale et al., 2019; Gholami et al., 2022). The obtained model is smaller, therefore, also faster to execute for inference. This approach has been shown to obtain a significant reduction in model size with close to negligible loss in accuracy (Han et al., 2015). Despite its potential, the performance of the compressed model can be only characterized a posteriori: a source DNN has to be first compressed and only then evaluated. As a consequence, tailoring a model to fulfill certain requirements results in a trial-and-error process that depends on appropriately setting the parameters of the specific compression method such as target sparsity level, quantization precision, fine-tuning schedule, and so on.

On the other hand, network architecture search explores a design space to find a model that fulfills application-specific requirements in an automated way (Chitty-Venkata & Somani, 2022). This approach effectively enables tailoring the characteristics of a DNN architecture, for instance, based on hardware-specific constraints. However, it entails carrying out an extremely large amount of computation to explore a combinatorially large design space (Wu et al., 2019; Stamoulis et al., 2019). Even worse, the process generally needs to be repeated from scratch for each target configuration. Recently, more advanced techniques have been introduced to train a so-called once-for-all network (Cai et al., 2019) that can be specialized for diverse



(a) (b)

Figure 1: (a)Given several datasets and models pre-trained on them, meta compression predicts the accuracy of a source model when compressed with different methods and settings, *without* running it. (b) Meta compression outperforms a recommendation strategy based on empirical risk minimization for high compression levels (at least 50 times), with a close-to-optimal prediction accuracy.

target configurations without re-training. However, these schemes significantly constrain the design space and still require substantial computation (Section 2).

This article bridges the gap between these two extremes by introducing **meta compression**, a novel approach based on meta learning to simplify a source DNN into one that fulfills application- and device-specific constraints. Our approach aims at answering the following questions: Which is the simplest DNN that achieves a certain accuracy? Which is the most accurate DNN that can run on a given device? It does so by predicting the compressed model accuracy of given pre-trained DNN models. This is achieved by learning across a collection of problems, consisting of several reference DNN models, compression methods, and their settings (Figure 1).

Employing pre-trained DNN models is a key design choice to reduce the computational overhead of meta learning. However, such a choice entails several challenges related to the partitioning of the source dataset the DNN was trained on. In fact, meta learning requires access to sufficient evaluation data to estimate generalization performance of compressed models. Evaluation data should also be separate from the test data used to evaluate the performance of the meta learning. In a practical deployment scenario, the training / testing split may not be known for a given pre-trained model, or the data not used for training may be not enough for meta learning. For this reason, we leverage diffusion models to improve the generalization performance of meta learning and also achieve a high prediction accuracy. We extensively evaluate our proposed framework by using popular DNNs, achieving a 92% top-5 recommendation accuracy compared to the 15% that is obtained with the compression algorithm that works the best on average. Moreover, our top recommendation is only 1% far from the optimal compression method in terms of average accuracy loss.

In summary, our work establishes the following contributions.

- We introduce a meta compression framework that takes a collection of pre-trained models and a set of compression methods so as to predict the accuracy of a compressed model without the need for evaluating it. Our framework obtains the most accurate option that fulfills given resource constraints. We also provide theoretical guarantees on the effectiveness of meta learning across multiple problems and compression methods (Section 3).
- We extensively evaluate meta compression on an image classification task with models pre-trained on the CIFAR10[1] dataset. The obtained results show that the learned meta predictor is indeed accurate: it reliably estimates the performance of new compressed architectures after compression by leveraging features specific to the DNNs and to the compression methods, in addition to evaluation data generated by diffusion models (Section 4).

[1]An evaluation on the ImageNet dataset is also provided in Appendix A.

2 Related Work

Model compression. DNN compression broadly encompasses techniques aiming at simplifying a source model into a smaller one (Hoefler et al., 2021). Among them, the major approaches are quantization, to reduce the precision of operands, and pruning, to sparsify a network by removing "unimportant" weights or activation values (Gholami et al., 2021; Liang et al., 2021). Both benefit from fine-tuning to restore accuracy (Sanh et al., 2020) and can jointly be applied for further reduction in model size (Hu et al., 2021; Yang et al., 2020; Frantar & Alistarh, 2022). However, characterizing the performance of specific compression methods have been elusive. (Kuzmin et al., 2023) introduce an analytical framework to compare pruning and quantization. However, their results are limited to magnitude pruning and uniform quantization as considered separately. In contrast, our solution allows to predict the accuracy of diverse compression methods, including those leveraging fine-tuning, even when jointly applied.

Network architecture search. Network architecture search aims at automatically selecting a model that can satisfy certain properties (Elsken et al., 2019). In this broad context, AutoNBA (Fu et al., 2021) devises a search strategy over networks, bitwidths, and features for hardware acceleration. To reduce computation, OFA (Cai et al., 2019) takes a reference DNN as a backbone to build a so-called once-for-all network. Such a network is trained in such a way that it can be tailored for hardware-specific constraints at a later stage, without re-training – the latter phase achieves pruning as a side effect. The same approach is extended in APQ (Wang et al., 2020) to also consider quantization. However, APQ only supports channel pruning and uniform quantization. Instead, our approach allows to plug-in arbitrary pruning and quantization schemes as compression methods.

Meta learning. Meta learning involves iterating over ML tasks while an outer training loop optimizes meta parameters (Hospedales et al., 2022) and has been applied to different scenarios (Garg & Kalai, 2018; Verma et al., 2020). Metapruning (Liu et al., 2019) applies meta learning to predict weights of a channel-pruned DNN, which is in turn employed to derive a specific channel pruning strategy. Instead, our approach applies meta learning to predict the performance of a compressed model and is not limited to pruning channels. In a different context, a few solutions have recently leveraged diffusion models in the context of meta learning. Among them, Meta-DM (Hu et al., 2023) performs data augmentation to improve the performance of few-shot learning, whereas the work in (Nava et al., 2023) carries out zero-shot task adaptation with both classifier and classifier-free guidance. In contrast, we introduce a framework for meta compression where diffusion models generate evaluation data to improve prediction performance.

3 Meta Compression

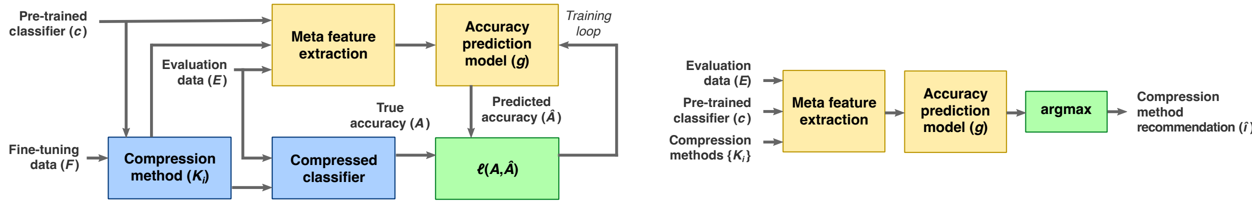
We now introduce the system model and describe our meta compression scheme (Figure 1). For clarity, we first address a single dataset and compression method, then extend our analysis to different configurations.

3.1 Overview and system model

A source *dataset* $D = \{T, F, E, V\}$ consists of different, disjoint sets: T for training, F for fine-tuning, E for evaluation, and V for testing. A *classifier*[2] c is pre-trained on T. Given these, a *problem* is defined as $p = (c, F, E)$. A *compression method* K uses the pre-trained classifier c and the fine-tuning dataset F to obtain a compressed classifier $c' = K(c, F)$. The loss of the compressed classifier is assessed over the evaluation set E to obtain the true accuracy A. An *accuracy prediction model* (meta predictor) g takes meta features (i.e., those derived by the meta compression) as input to derive the predicted accuracy $\tilde{A}$.

Accordingly, the training of the accuracy prediction model follows the process in Figure 2a. Meta feature extraction involves computing problem features, compression method features, and compressed classifier performance (see Section B.1). A given classifier c compressed by using a configuration i with K_i, then the performance of the compressed classifier c'_i is assessed by using the evaluation dataset E. In summary,

[2]For simplicity, here we consider classification problems, even though our formulation is general, thus, applicable to different domains. To show this, Appendix C applies meta compression to a regression task.



(a) (b)

Figure 2: (a) The inner loop of the meta learning process involves training a model to predict the performance of a classifier once compressed with a certain method and related settings. (b) Once learned, the accuracy prediction model can be leveraged to recommend the best compression method for a given problem.

training g involves learning a mapping from problem-specific and compression method-specific features to the compressed classifier performance.

We then proceed to the recommendation once the accuracy prediction model is trained. The first type of recommendation involves selecting the most accurate compression model that satisfies device-specific constraints. To do so, we first prepare a shortlist of compression methods along with their settings. We then refine the candidate options by only keeping those that satisfy the given constraints, for instance, in terms of model size or target compression ratio. The best compression method can be selected by a simple $\arg\max$ function over the predicted accuracy values (Figure 2b).

The second type of recommendation entails selecting the method that obtains the highest compression while satisfying an application-specific accuracy target. For this purpose, we consider all available compression methods as predictions and prepare a shortlist of those fulfilling the accuracy constraint. Similar to the previous case, we use the $\arg\max$ function to obtain the method that achieves the highest predicted compression.

We have just outlined of how meta compression allows to answer the research questions introduced in Section 1. The next step is to establish a rigorous theoretical foundation by leveraging the Probably Approximately Correct (PAC) learning framework to obtain theoretical guarantees for making meta learning effective across multiple problems and compression methods.

3.2 Theoretical foundation

So far, we have limited our attention to a single problem p and a given compression method K, specialized into K_i according to different configurations. We now extend our consideration to a collection of problems and compression methods, and precisely state our objective.

In classification problems, D is a labeled dataset $(\mathcal{X}, \mathcal{Y})$ and $c = L(T)$ is a classifier obtained by learner $L : (\mathcal{X}, \mathcal{Y})^* \to \mathcal{Y}^{\mathcal{X}}$ (namely, the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$) based on the training set $T \sim \mu^n$. A function $l : \mathcal{Y} \times \mathcal{Y} \to [0, 1]$ is such that $l(y, \hat{y})$ is the loss when the prediction is $\hat{y}$ and the true label is y. $l_\mu(c) = \mathbb{E}_{(x,y)\sim\mu}\left[l(y, c(x)\right]$ characterizes the expected loss of a classifier over some data distribution μ.

Now, consider a compression method $K : \mathcal{Y}^{\mathcal{X}} \times (\mathcal{X} \times \mathcal{Y})^* \to \mathcal{Y}^{\mathcal{X}}$ such that $c' = K(c, F)$, wherein the index i is dropped from K_i for convenience. The loss of the compressed classifier can be evaluated by using the evaluation dataset E. Let us consider a distribution over problems $p \sim \nu$ to evaluate the loss of a compression method K. Each problem specifies a pre-trained classifier $c \in \mathcal{Y}^{\mathcal{X}}$, and retraining and evaluation dataset $F, E \in (\mathcal{X} \times \mathcal{Y})^*$. Then, the expected loss of a compression method can be defined as:

$$l_\nu(K) = \mathbb{E}_{p\sim\nu}\left[l(y, K(c, F)(x))\right]$$

Let us consider the task of selecting the best performing compression method across several problems, given a family of compression methods $\mathcal{K}$ that satisfy certain constraints on the compression level. A practical objective is to select the compression method with the lowest empirical error across multiple problems, namely, to carry out an empirical risk minimization (ERM). Accordingly, let us consider a collection P of problems, containing problems $(c_\tau, F_\tau, V_\tau) = p_\tau \in P$. Then,

$$\mathrm{ERM}_{\mathcal{K}}(P) \in \arg\min_{K \in \mathcal{K}} \sum_{p_\tau \in P} \sum_{(x_t, y_t) \in E_\tau} l(y_t, K(c_\tau, F_\tau)(x_t))$$

Lemma 3.1. *For any finite family $\mathcal{K}$ of compression algorithms, any distribution ν over problems p_τ, and any $n \geq 1$, $\delta > 0$,*

$$\Pr_{P \sim \nu^n} \left[l_\nu \left(\mathrm{ERM}_{\mathcal{K}}(P) \right) \leq \min_{K \in \mathcal{K}} l_\nu(K) + \sqrt{\frac{2}{n} \log \frac{|\mathcal{K}|}{\delta}} \right] \geq 1 - \delta \tag{1}$$

Proof. Let

$$\epsilon = \sqrt{\frac{2}{n} \log \frac{|\mathcal{K}|}{\delta}}.$$

Consider an empirical estimate of $l_\nu(K)$ using some dataset of problems $P \sim \nu^n$, $(c_\tau, F, E) = p_\tau \in P$, given by

$$l_P(K) = \frac{1}{n} \sum_{p_\tau \in P} \sum_{(x_t, y_t) \in E} l(y_t, K(c_\tau, F_\tau)(x_t)).$$

Denote the optimal compression method $K_0 \in \arg\min_{K \in \mathcal{K}} l_\nu(K)$. Writing the Chernoff bound for $l_P(K_0)$, we have

$$\Pr_{P \sim \nu^n} \left[l_P(K_0) \geq l_\nu(K_0) + \epsilon/2 \right] \leq e^{-2n(\epsilon/2)^2}.$$

Now, consider the set B of bad compression methods that are more than ϵ far away from $l_\nu(K_0)$. More formally, $B = \{K \in \mathcal{K} \mid l_\nu(K) \geq l_\nu(K_0) + \epsilon\}$. By writing the Chernoff bound for each $K \in B$, we have

$$\Pr_{P \sim \nu^n} \left[l_P(K) \leq l_\nu(K) - \epsilon/2 \right] \leq e^{-2n(\epsilon/2)^2}.$$

Clearly, $l_\nu(\mathrm{ERM}_{\mathcal{K}}) \geq l_\nu(K_0) + \epsilon$ holds only if $l_D(K) \leq l_D(K_0)$ for some $K \in B$. Specifically, either $l_P(K) \leq l_\nu(K) - \epsilon/2$ for some $K \in B$, or $l_D(K_0) \geq l_\nu(K_0) + \epsilon/2$. By applying the union bound, the result holds with probability at most $|\mathcal{K}| e^{-2n(\epsilon/2)^2} = \delta$. ☐

The lemma implies that the best compression method can be found across *all problems* by using $\mathcal{O}(log(|\mathcal{K}|))$ training problems. It is also possible to select the compression method that is best suited to a specific problem. Hence, let us consider the task of choosing the best compression method among candidates $\{K_1, \ldots, K_m\}$ given the problem specification p_τ. For this purpose, let us consider problem features $\phi(c_\tau, F_\tau, E_\tau) \in \Phi$ and compression method features $\gamma(K) \in \Gamma$. Moreover, let us consider a family $\mathcal{F}$ of functions $f : \Phi \times \Gamma^m \to \{1, \ldots, m\}$ that apply compression method among m based on extracted features. Finally, the objective of finding the function that selects the best compression method based on ERM can be defined as:

$$\arg\min_{f \in \mathcal{F}} \sum_{p_\tau \in P} \sum_{(x_t, y_t) \in E_\tau} l(y_t, K_{f(\phi(p_\tau), \gamma(K_1), \ldots, \gamma(K_m))}(c_\tau, F_\tau)(x_t)) \tag{2}$$

Solving $\mathrm{ERM}_{\mathcal{F}}$ entails a multi class classification problem based on problem features $\phi(c_\tau, F_\tau, E_\tau)$ and compression method features $\gamma(K)$, while $\mathrm{ERM}_{\mathcal{K}}$ selects the best single class which we call the *static ERM strategy.*

Theorem 3.2. *The class $\{K_i | i \in [1, m]\}$ of m compression methods can be agnostically learned. In particular, a polynomial-time algorithm achieves an error of at most*

$$\min l_\nu(K_i) + \sqrt{2/n(\log m/\delta)}$$

with probability $\geq 1 - \delta$ over n training problems.

Proof. Lemma 3.1 implies that, given n training problems and $|K_i| \leq m$, the error of ERM is within $\sqrt{2/n \log m/\delta}$ of $\min_i l_\nu(K_i)$. Let us compute the mean loss of K_i across the training problems, for each compression method $i \in [1, m]$, to find the best compression method in polynomial time (of the input size $|T|$). As K_i runs in polynomial time, the loss can be computed in polynomial time too, and the number of compression methods is also bounded. Thus, the entire procedure takes polynomial time. $\square$

The theorem establishes the learnability of the compression method recommendation function f. We further simplify the design of f by employing a loss prediction function $g : \Phi \times \Gamma \to \mathbb{R}_{\geq 0}$ that predicts the loss of a compressed classifier c' for a given problem p_τ and compression method K_i:

$$g(\phi(p_\tau), \gamma(K_i)) \approx l_\mu(K_i(c_\tau, F_\tau))$$

Accordingly, the ERM objective for learning g is

$$\arg\min_{g \in \mathcal{G}} \sum_{p_\tau \in P} \sum_i \left[g(\phi(p_\tau), \gamma(K_i)) - l_{\mu^e}(K_i(c_\tau, F_\tau))\right]^2 \tag{3}$$

Once g has been learned by using sufficient data, f can be described in terms of g as:

$$f(\phi(p_\tau), \gamma(K_1), \ldots, \gamma(K_m)) = \arg\min_{i=\{1,\ldots,m\}} g(\phi(p_\tau), \gamma(K_i)) \tag{4}$$

This approach finds the best compression method for any problem d_τ. Consequently, it also finds the best compression methods across all problems, which was the ERM objective of learning f. Such a formulation of f has an additional benefit, as it allows us to solve this performance-constrained compression maximization problem too with the same loss prediction function g. We choose classification accuracy as the negative of loss to be predicted using g, and refer to g as the accuracy predictor in the rest of the paper.

3.3 Meta features and accuracy predictor

We employ gradient boosted decision trees (Chen & Guestrin, 2016) as the accuracy prediction model according to (White et al., 2021), which showed that they are remarkable in predicting the performance DNNs for NAS without substantial computation. We have also empirically confirmed that employing a feed-forward DNN did not improve the results. Appendix B.1 details the meta features in the decision tree and their importance.

Obtaining a compressed classifier requires applying a compression method K to a pre-trained classifier c and possibly retraining using tuning dataset F. Thus, making predictions about the compressed classifier performance requires problem features $\phi(p_\tau)$ and compression method features $\gamma(K_i)$. We also need an evaluation dataset E that is separate from the tuning dataset to accurately evaluate the behavior of the compressed classifier. We describe these features in more detail next.

Problem features ($\phi(p_\tau) \in \Phi$). A problem $p_\tau = (c_\tau, F_\tau, E_\tau)$ consists of a pre-trained classifier, and tuning dataset, and an evaluation dataset. We extract two set of features to describe the pre-trained classifier: architecture features and solution features. Architecture features encode information about the modules used as building blocks of the DNN architecture. Instead, solution features describe the particular solution learned from training the model. These include norms of weights, gradients, in addition to loss and accuracy of c evaluated through E.

Compression method features ($\gamma(K_i) \in \Gamma$). The compression process typically consists of iterative pruning and retraining, followed by quantization and retraining. We consider several popular pruning and quantization methods in our experiments. Each pruning method is encoded via a pruning identifier and the target sparsity level. Similarly, each quantization method is encoded with a unique quantization identifier and the target level of bits.

Compressed classifier performance. We evaluate the loss and accuracy of the compressed classifier on the evaluation dataset E considered as the ground truth for predictions.

The accuracy prediction model g takes problem features and compression method features as input to predict the compressed classifier performance.

4 Experimental evaluation

We conduct extensive experiments to evaluate the effectiveness of the proposed meta compression framework in estimating the performance of common pruning and quantization methods. We also characterize the impact evaluation data selection and the generalization performance of the meta predictor to new evaluation data, new architectures, and new compression methods. The performance of the recommended compression method is always evaluated using the test set V, which is left untouched until this step. The rest of the section presents the obtained results for image classification models trained on the CIFAR10 (Krizhevsky, 2009) dataset. The appendix further provides an evaluation for the the same task on the ImageNet (Deng et al., 2009) dataset (Appendix A), additional results on the compute costs (Appendix B.2), and an application to a regression task (Appendix C). The code to reproduce the experiments is available at `https://anonymous.4open.science/r/DeeperCompression-4EED/`.

4.1 Experimental setup

4.1.1 Architectures

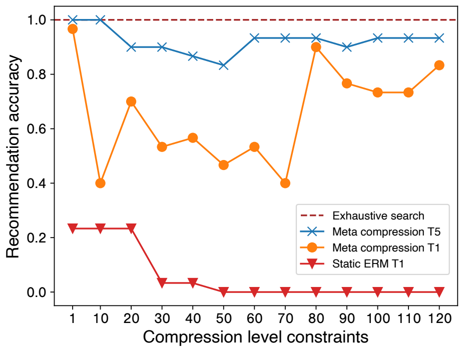
Our setup include popular DNN models trained on the CIFAR10 dataset, namely, VGG19 (Simonyan & Zisserman, 2014), ResNet18 (He et al., 2015b), GoogLeNet (Szegedy et al., 2014), DenseNet121 (Huang et al., 2016), ResNeXt29-2x64d (Xie et al., 2016), MobileNet (Howard et al., 2017), MobileNetV2 (Sandler et al., 2018), DPN92 (Chen et al., 2017), SENet18 (Hu et al., 2017), ShuffleNetV2 (Ma et al., 2018), RegNetX-200MF (Radosavovic et al., 2020), SimpleDLA (Yu et al., 2017), through their implementation in (Kuang-Liu, 2017). Unless stated otherwise, F consists of 20% randomly sampled images from the CIFAR10 training dataset, E consists of 10k images generated using a diffusion model trained on CIFAR10 train dataset, and V comprises the complete CIFAR10 test dataset. The latency bottleneck of the training loop is obtaining the true accuracy of the compressed classifier, which takes around 40 minutes for a single pruning / quantization level (40 retraining epochs) on a Tesla A100 GPU. The loop iterates over 10 different sparsity and 4 different quantization levels, a total of 40 combinations. The total time required to run this configuration is approximately 26 hours (12 GPUs for 1.0833 days, i.e., 13 GPU days) considering parallel execution on 12 Tesla A100 GPUs for 12 different architectures. Once learned, obtaining recommendations through the accuracy prediction model takes only a few minutes.

4.1.2 Compression methods

We selected a broad set of compression methods that span complementary design paradigms and are applicable across diverse DNN architectures. A large number of pruning and quantization techniques exist, however, integrating them – especially together with retraining – poses engineering challenges because the individual stages can be mutually incompatible (Qu et al., 2025). For this reason, we evaluated a large number of options and eventually selected the methods discussed next, as they were able to consistently and reliably compress the pre-trained models considered in our study.

Pruning. We consider approaches entailing both unstructured and structured pruning. Specifically, we selected the following unstructured pruning methods: *magnitude pruning*, consisting of a simple weight pruning scheme; and *pruning+tuning*, as the former followed by retraining for 40 epochs. In addition, we consider the following structured pruning techniques: *L1 norm pruning* (Li et al., 2016), which removes weights with low L1 norms; *network slimming* (Liu et al., 2017), which masks scaling factors in later batch normalization layers to prune channels in convolution layers; and *TaylorFO* (Molchanov et al., 2019), which prunes convolutional layers based on the first-order Taylor expansion on the weights.

Quantization. We considered the following quantization methods: *uniform affine quantizer* (Krishnamoorthi, 2018), which maps continuous values to discrete levels by scaling and rounding within a given quantization range; *Dorefa-net* (Zhou et al., 2016), which represents the weights and activations through a ternary code to save storage, then employs a mix of fixed-point quantization, scaling, and rounding to reduce quantization error; *learned step-size quantization* (Esser et al., 2020), which employs a learnable scaling factor and a fixed quantization step size.



(a)

Metric	Meta Compression	Static ERM
T5 Accuracy	**0.92**	0.15
T1 Accuracy	**0.66**	0.06
T1 Error	**0.01**	0.25
MAE	**0.10**	0.12
Kendall's Tau (τ)	**0.65**	–

(b)

Figure 3: (a) Recommendation accuracy of meta compression against the optimal static ERM and exhaustive search for different compression levels. (b) Recommendation performance summary metrics comparing meta compression with the static ERM approach (the single best pruning and quantization methods used across all constraint levels).

Compression Algorithms	T5 Accuracy	T1 Accuracy	T1 Error	MAE
Prune + Quant	0.97	0.92	0.02	0.14
Level-prune + Dorefa-quant	0.97	0.65	0.04	0.07
L1-prune + Dorefa-quant	0.93	0.45	0.04	0.08
TaylorFO-prune + Dorefa-quant	0.85	0.38	0.06	0.08
Slim-prune + Dorefa-quant	0.99	0.70	0.03	0.18
Level-prune + LSQ-quant	0.98	0.78	0.01	0.02
L1-prune + LSQ-quant	0.99	0.64	0.02	0.06
TaylorFO-prune + LSQ-quant	1.00	0.64	0.03	0.07
Slim-prune + LSQ-quant	0.98	0.72	0.02	0.17

Table 1: MAE of the predictor g trained for different compression algorithms.

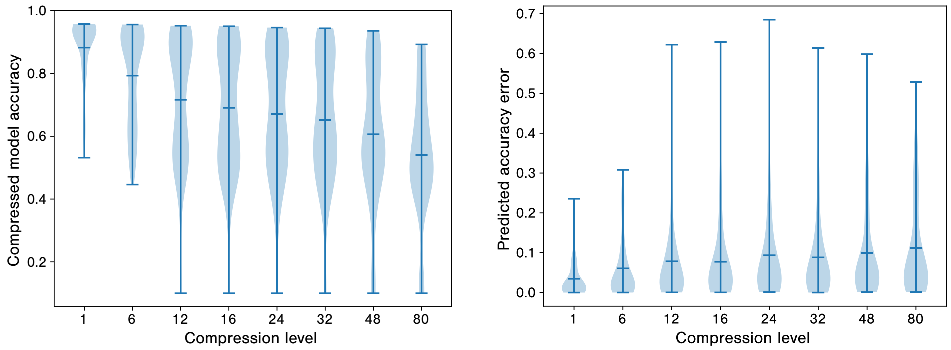
4.2 Recommendation performance

We evaluate the recommendation performance of the proposed meta compression algorithm by considering whether any of the top-k recommendations for a given constraint performs at most within a factor of ϵ with respect to the optimal recommendation. We primarily refer to the task of maximizing accuracy subject to compression constraints, for which the analytical results in Section 3.2 hold. We also present some results for the task of maximizing compression given an accuracy constraint at the end of the section.

We compute top-k recommendation accuracy by averaging it across several constraints. We also compute top-1 error, which characterizes how far off is the top recommendation from the optimal one in terms of compressed model accuracy (on average). For comparison purposes, we also consider selecting the single best pruning and quantization method on a dataset of problems and recommending the same for all architectures and compression level constraints (i.e., the static recommendation with ERM). We find that Slim pruning (Liu et al., 2017) and learned step size quantization (Esser et al., 2020) perform the best in such a setting.

We start by splitting the set of architectures into train set and test set. The meta prediction model is trained for architectures in the train split, and used to predict performance of architectures in the test split. We perform ten such splits to remove bias towards a specific choice. We set the tolerable drop in accuracy ϵ to 0.01 and vary the minimum compression constraint between 1x to 120x, with a step size of 10. We then compare the predictions made using our meta prediction model against the optimal choice found through exhaustive search for each constraint. The corresponding results are reported in Figure 3a, which clearly shows how the proposed meta compression approach outperforms fixed recommendation with ERM.

Figure 3b reports different aggregate metrics across all compression constraints. The results show that the top-5 performance is 92% and the average error is only 0.01 compared to 0.25 for static recommendation with ERM. In other words, one the top-5 recommendations of meta compression is close to the optimal



(a)

(b)

Figure 4: Accuracy of the (a) compressed model and of the (b) prediction for different compression levels.

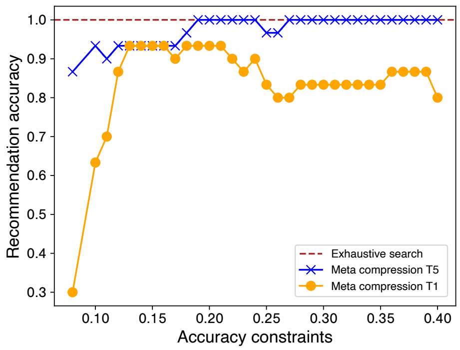
compression algorithm 92% of times and the performance of the recommended compression method has only a 1% difference in accuracy on average.

We also report the Kendall's Tau (τ), which measures the rank correlation between the ordering induced by our predicted accuracy and the that of the observed accuracy across compression options under a given constraint (averaged over constraints). Higher values of τ indicate that the predictor preserves the relative orderings that drive recommendation quality: values close to 1 mean near-perfect agreement, while 0 implies no association. In our experiments, τ is high and comparable to the numbers reported by White et al. (2021) in a study on NAS. This is remarkable, as our study entails more complex settings that requires ranking compression methods applied to previously unseen architectures, compared to the simpler task of ranking unseen architectures in that work.

We also evaluate how the accuracy prediction model g performs across different compression methods. We divide the test set by method under the same setup employed for Figure 3, then assess accuracy / prediction and recommendation performance on each subset. Table 1 reports a MAE ranging between 2 and 18%. Even though a few methods exhibit a rather high MAE, the top-5 recommendation remains high, indicating that g preserves the within-method ranking of pruning / quantization levels even when absolute predictions are biased. Moreover, one-shot Prune + Quant has a higher MAE than most methods with retraining, possibly because retraining shifts and concentrates accuracy toward higher values, thereby making the prediction task easier. Overall, the recommendation quality remains strong across different methods.

Figure 4 characterizes how the compressed model accuracy and the prediction error vary with the compression levels (grouped into equally-sized bins). Figure 4a shows that accuracy are higher and more concentrated at very low compression levels, thereby reducing the prediction error. The mean prediction error remains roughly stable as compression increases, but its variance grows. Pushing compression beyond the range shown in the figure substantially reduces accuracy (below 50%), so raw accuracy values are not informative and are therefore omitted. Furthermore, Figure 4b shows the absolute error in the prediction accuracy across compression levels. The error remains consistently low, even at 48× and 80× compression. The variance is more pronounced in the higher range of the absolute error, although most of the values are concentrated below the mean. The high variance of the absolute error is notably correlated with that of the compressed model accuracy, as one would expect.

Finally, Figure 5 shows the accuracy in predicting the method that achieves the highest compression given an acceptable level of loss of accuracy. Specifically, Figure 5a show that the learned accuracy predictor performs similar to the previously-considered task, with the best top-5 recommendation performing close to the optimal solution across all constraint levels. Table 5b presents aggregate metrics, highlighting that meta compression achieves over 90% top-5 recommendation accuracy on both accuracy-constrained and resource-constrained tasks.



(a)

Metric	Meta Compression Accuracy-constrained	Meta Compression Resource-constrained
T5 Accuracy	0.98	0.92
T1 Accuracy	0.84	0.66
T1 Error	0.10	0.01
MAE	0.07	0.10

(b)

Figure 5: (a) Recommendation accuracy of meta compression against exhaustive search for different accuracy constraints and (b) summary metrics comparing recommendations for the accuracy-constrained compression maximization task with those for the compression-constrained accuracy maximization task.

4.3 Design choice evaluation

This section evaluates the impact of data selection on performance and characterizes the generalization capability of the predictor.

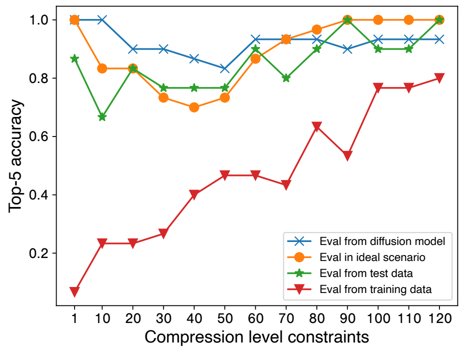
4.3.1 Evaluation data

The primary metric under consideration when choosing the evaluation data E is the generalization performance to test data. For satisfactory outcomes, it is desirable that the evaluation data contains samples not seen during the pre-training or fine-tuning phase. In the *ideal* scenario, this could be achieved by reserving some of the training data to be only used as evaluation data. However, this is often impossible as the full training data is generally used by pre-trained models, and pre-training the models again on a limited subset of train would be very expensive. Another option is to sample the evaluation data from the test data to avoid retraining from scratch. Unfortunately, this approach has several disadvantages. First, it risks leaking information from test data, possibly compromising the integrity of the final test evaluation. Second, sampling from test data in practice results in a dataset with a very limited size. Instead, we propose using diffusion models to generate evaluation data from the source dataset to overcome the above-stated limitations.

Figure 6a shows the top-5 recommendation accuracy as a function of the compression level. As expected, using evaluation samples drawn from the training set yields the worst performance across all compression levels. Sampling from the test set performs better but is typically inferior to the remaining two strategies. The ideal scenario (i.e., restricting evaluation to held-out samples from the original dataset) achieves strong results and sometimes outperforms using samples from the generative model at certain compression levels. However, evaluation data generated by a diffusion model trained on the training data delivers the best performance. Table 6b summarizes aggregate metrics for recommendation quality and MAE of the accuracy predictor, where using a diffusion model performs slightly better than the ideal scenario across all metrics. We attribute this improvement to the ability to generate a sufficiently large evaluation set without reducing the size of the training set, which would otherwise degrade the performance of the pre-trained models.

4.3.2 Generalization performance

We train the meta predictor across multiple problems and compression algorithms. Therefore, it is very important to characterize how well it generalizes to scenarios involving unseen architectures, unseen data, or even novel compression algorithms. Such an analysis also reveals insights into what can be learned and what cannot, and is key to validating the hypothesis that performance of compression methods applied to novel problems can be predicted in practice (as stated by Theorem 3.2). We carried out several experiments to answer these questions, which are summarized in Table 2.



(a)

Eval data selection	T5 Accuracy	T1 Accuracy	T1 Error	MAE
From training data	0.48	0.14	0.13	0.24
From test data	0.85	0.55	0.06	0.12
Ideal (From original dataset)	0.89	0.58	0.03	0.12
Diffusion model samples	**0.92**	**0.66**	**0.01**	**0.10**

(b)

Figure 6: (a) Evaluation data selection choices and (b) Prediction performance for different data selection strategies.

New data	New architectures	New compr. methods	T5 Accuracy	T1 Accuracy	T1 Error	MAE
No	Yes	No	0.92	0.66	0.01	0.10
Yes	Yes	No	0.91	0.66	0.02	0.11
No	No	Yes	0.86	0.34	0.11	0.13
Yes	No	Yes	0.85	0.34	0.12	0.15

Table 2: Generalization performance of g to new data, architectures, and compression methods.

New architectures. The first question is how well a meta predictor trained on one set of architectures perform at predicting performance of new architectures. This resembles a typical use case scenario where the same evaluation data used for training the meta predictor is available, and the best compression method from an already defined set of compression methods needs to be estimated.

To evaluate this, we split the set of 12 pre-trained DNNs in a 3:1 train-test ratio, and report the average the performance across ten random splits. On average, the accuracy of the predicted compression method shows only a 1% drop in accuracy (T1 error) compared to the optimal compression method, and a top 5 recommendation accuracy of 92%. This validates the hypothesis that compression performance of novel architectures can effectively be predicted by learning across a sufficiently large dataset of DNN architectures.

New data. This setting evaluates the case where the evaluation dataset used during meta-training is not available at test time. The evaluation set is distinct from the data used to train the base DNNs and is used only to compute post-compression accuracy while training the meta-predictor; at deployment it may differ or could be unavailable. To assess sensitivity to such a dataset shift, we hold out the original evaluation set and evaluate on a disjoint dataset. Under this shift, the predicted top-1 selection is within 2% of the optimal method on average (T1 error), and the top-5 recommendation accuracy is 91%. The impact of not sharing the same evaluation data is therefore modest, indicating appropriate generalization to new data.

New compression methods. This is an atypical scenario, which considers generalization to new compression methods while giving recommendations for pre-trained models already seen during the meta-training phase. In practice, a known set of compression methods is applied to novel problems, and not vice-versa. Still, these experiments describe to what extent similar patterns exist in the compression characteristics of different compression methods, and how well these patterns generalize to unseen compression methods.

To evaluate this, we partition compression specifications by method: we train the meta-predictor on a subset of compression methods and evaluate on held-out methods, by keeping the same set of pre-trained classifiers. The last two rows of Table 2 correspond to this setup. We see that top-1 selection performance substantially reduces (e.g., with a higher T1 error or lower top-1 accuracy), whereas the top-5 recommendation accuracy decreases only modestly. This is expected: different compression methods exhibit distinct trade-offs, yet there is enough shared structure for the correct method to appear among the top-5 recommendations in a majority of the cases. The exact magnitudes depend on the particular set of methods considered in this work and the shared structures present in their compression-accuracy trade-offs.

5 Conclusion

This work has addressed the problem of compressing large deep neural networks (DNNs) so that they can be deployed onto edge servers and resource-constrained devices. Specifically, we learned how to compress DNNs with a novel meta-learning approach – the first, to the best of our knowledge. Accordingly, we train a regression model to predict the accuracy of a pre-trained DNN after being compressed with a combination of techniques – including pruning and quantization – without having to evaluate it. We take a flexible yet rigorous approach to achieve provably approximate correct meta learning. We also carry out an extensive evaluation using common compression schemes. The obtained results demonstrate that meta compression is effective, and that diffusion models are instrumental to improve the generalization capabilities and also the accuracy of the predictor. As a consequence, meta compression can be applied even to scenarios in which evaluation data is scarce, thereby extending its applicability in practice.

Our study has focused on classification tasks in the visual domain. Extending meta compression to different scenarios (such as natural language processing) and more complex architectures (including graph neural networks) would be an interesting direction for future work.

References

Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8:1–74, 2021.

Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Hannun, Billy Jun, Patrick LeGresley, Libby Lin, Sharan Narang, Andrew Ng, Sherjil Ozair, Ryan Prenger, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Yi Wang, Zhiqian Wang, Chong Wang, Bo Xiao, Dani Yogatama, Jun Zhan, and Zhenyao Zhu. Deep speech 2: End-to-end speech recognition in english and mandarin, 2015.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 785–794, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785. URL `http://doi.acm.org/10.1145/2939672.2939785`.

Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual path networks, 2017. URL `https://arxiv.org/abs/1707.01629`.

Krishna Teja Chitty-Venkata and Arun K Somani. Neural architecture search survey: A hardware perspective. *ACM Computing Surveys*, 55(4):1–36, 2022.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Radosvet Desislavov, Fernando Martínez-Plumed, and José Hernández-Orallo. Trends in ai inference energy consumption: Beyond the performance-vs-parameter laws of deep learning. *Sustainable Computing: Informatics and Systems*, 38:100857, 2023.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. In *International Conference on Learning Representations*, 2020.

Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35:4475–4488, 2022.

Yonggan Fu, Yongan Zhang, Yang Zhang, David Cox, and Yingyan Lin. Auto-nba: Efficient and effective search over the joint space of networks, bitwidths, and accelerators. In *International Conference on Machine Learning*, pp. 3505–3517. PMLR, 2021.

Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

Vikas Garg and Adam T Kalai. Supervising unsupervised learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pp. 291–326. Chapman and Hall/CRC, 2022.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34:18932–18943, 2021.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015b. URL https://arxiv.org/abs/1512.03385.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *The Journal of Machine Learning Research*, 22(1):10882–11005, 2021.

T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 44(09):5149–5169, sep 2022. ISSN 1939-3539. doi: 10.1109/TPAMI.2021.3079209.

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017. URL https://arxiv.org/abs/1704.04861.

Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks, 2017. URL https://arxiv.org/abs/1709.01507.

Peng Hu, Xi Peng, Hongyuan Zhu, Mohamed M Sabry Aly, and Jie Lin. Opq: Compressing deep neural networks with one-shot pruning-quantization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7780–7788, 2021.

Wentao Hu, Xiurong Jiang, Jiarun Liu, Yuqi Yang, and Hui Tian. Meta-dm: Applications of diffusion models on few-shot learning, 2023.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2016. URL https://arxiv.org/abs/1608.06993.

Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size, 2016.

Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/cifar.html.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Liu Kuang-Liu. Pytorch-cifar. https://github.com/kuangliu/pytorch-cifar, 2017. [Online; accessed 2023-04-16].

Andrey Kuzmin, Markus Nagel, Mart van Baalen, Arash Behboodi, and Tijmen Blankevoort. Pruning vs quantization: Which is better? *arXiv preprint arXiv:2307.02973*, 2023.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403, 2021.

Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3296–3305, 2019.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pp. 2736–2744, 2017.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design, 2018. URL https://arxiv.org/abs/1807.11164.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11264–11272, 2019.

Elvis Nava, Seijin Kobayashi, Yifei Yin, Robert K. Katzschmann, and Benjamin F. Grewe. Meta-learning via classifier(-free) diffusion guidance, 2023.

R. Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics and Probability Letters*, 33: 291–297, 1997.

Xiaoyi Qu, David Aponte, Colby Banbury, Daniel P Robinson, Tianyu Ding, Kazuhito Koishida, Ilya Zharkov, and Tianyi Chen. Automatic joint structured pruning and quantization for efficient neural network training and compression. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 15234–15244, 2025.

Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces, 2020. URL https://arxiv.org/abs/2003.13678.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Victor Sanh, Thomas Wolf, and Alexander Rush. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in Neural Information Processing Systems*, 33:20378–20389, 2020.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014. URL https://arxiv.org/abs/1409.1556.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

Dimitrios Stamoulis, Ruizhou Ding, Di Wang, Dimitrios Lymberopoulos, Bodhi Priyantha, Jie Liu, and Diana Marculescu. Single-path nas: Designing hardware-efficient convnets in less than 4 hours. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 481–497. Springer, 2019.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014. URL https://arxiv.org/abs/1409.4842.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

Vinay Kumar Verma, Dhanajit Brahma, and Piyush Rai. Meta-learning for generalized zero-shot learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 6062–6069, 2020.

Tianzhe Wang, Kuan Wang, Han Cai, Ji Lin, Zhijian Liu, Hanrui Wang, Yujun Lin, and Song Han. Apq: Joint search for network architecture, pruning and quantization policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2078–2087, 2020.

Colin White, Arber Zela, Robin Ru, Yang Liu, and Frank Hutter. How powerful are performance predictors in neural architecture search? *Advances in Neural Information Processing Systems*, 34:28454–28469, 2021.

Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10734–10742, 2019.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks, 2016. URL https://arxiv.org/abs/1611.05431.

Haichuan Yang, Shupeng Gui, Yuhao Zhu, and Ji Liu. Automatic neural network compression by sparsity-quantization joint learning: A constrained optimization-based approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2178–2188, 2020.

Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation, 2017. URL https://arxiv.org/abs/1707.06484.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices, 2017.

Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

A Evaluation on ImageNet dataset

We also carried out experiments on the ImageNet dataset Deng et al. (2009). In this case, we considered the following architectures: VGG11 (Simonyan & Zisserman, 2015), Squeezenet (Iandola et al., 2016), Densenet121 (Huang et al., 2016), Alexnet (Krizhevsky et al., 2012), Resnet (He et al., 2015a) and Shufflenet (Zhang et al., 2017). Here F consists of 30k labelled images taken from the ImageNet training set, V (30k images) and E (20k images) comprise images sampled from the ImageNet validation set (50k images) in the ratio of 3:2, respectively. The rest of the section characterizes the recommendation and generalization performance of meta compression for the case of maximizing accuracy subject to compression constraints.

A.1 Recommendation performance

We conducted ImageNet recommendation experiments for a smaller set of compression levels due to significantly higher compute cost for fine tuning ImageNet models. The maximum sparsity level is set to 0.94, the considered quantization precision is limited to 16 and 32 bits, and the maximum compression level is set to 24. We have considered 10 different sparsity levels ranging from 0 to 0.94. We evaluate the recommendation performance of the proposed meta compression algorithm by considering whether any of the top-k recommendations for a given constraint performs at most ϵ worse than the optimal recommendation. We find that the best static strategy for this case involves quantizing to 32 bits with LSQ and adapting Slim pruning rate to match the compression level provided in the constraint. We set ϵ to 0.01, same as the CIFAR10 setup, and vary the minimum compression constraint between 1x to 24x, with a step size of 1.

The top-1 recommendation performance and MAE of learned accuracy prediction function g are reported in Table 3. The top-1 error reveals an 8% drop in accuracy on average, compared to the 44% drop when using the static ERM approach. The same observation can also be made from Figure 7, where meta compression significantly outperforms the static recommendation strategy at higher compression levels.

Table 4 shows the recommendation performance for different compression methods, while Figure 8 shows the distribution of compressed model accuracy and absolute error in accuracy prediction at several compression levels. Compared to the CIFAR10 setup, the accuracy of compressed models drops rapidly as compression level increases. This can be improved with more fine-tuning using more data and retraining epochs, at the cost of additional computation. Once again, the variance of compressed model accuracy increases with the mean absolute error in accuracy prediction at different compression level ranges. The absolute error values are concentrated below the mean, and the mean always stays below 0.12.

Eval data selection. For the ImageNet setup, we conducted experiments with two data selection choices, a) E sampled from train data, and b) E sampled from test data. Table 5 tabulates the results. The behavior is consistent with the observation made for the CIFAR setup, that sampling from test data (and using a reduced test set for final evaluation) performs significantly better than sampling from train data as the feature evaluations done during the meta-training phase generalize well to testing/recommendation phase.

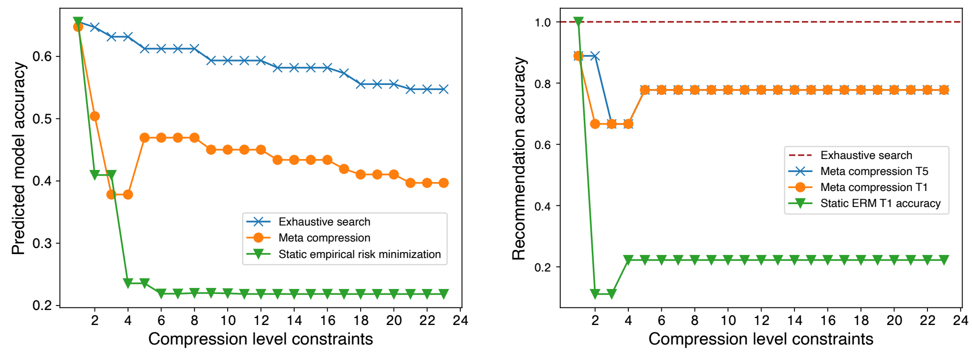
A.2 Generalization performance

We now discuss the results on generalization for the ImageNet setup presented in Table 6.

New architectures. This is the default setup. It assumes using the same evaluation data and compression methods that have been used during meta training to be also used at the recommendation phase. This resembles a typical deployment scenario where we apply previously known compression methods to new

Metric	Meta Compression	Static ERM
T1 Accuracy	**0.77**	0.25
T1 Error	**0.08**	0.44
MAE	**0.12**	0.13

Table 3: Recommendation performance for the ImageNet setup.



(a) (b)

Figure 7: Accuracy of the (a) predicted model and of the (b) predictor across several compression level constraints.

Compression Algorithms	T5 Accuracy	T1 Accuracy	T1 Error	MAE
Prune + Quant	0.98	0.80	0.03	0.13
Level-prune + Dorefa-quant	0.69	0.49	0.16	0.25
L1-prune + Dorefa-quant	0.81	0.55	0.03	0.08
TaylorFO-prune + Dorefa-quant	0.73	0.66	0.02	0.08
Slim-prune + Dorefa-quant	0.90	0.79	0.04	0.19
Level-prune + LSQ-quant	1.0	1.0	0.01	0.10
L1-prune + LSQ-quant	0.95	0.79	0.02	0.06
TaylorFO-prune + LSQ-quant	0.97	0.75	0.02	0.06
Slim-prune + LSQ-quant	0.93	0.85	0.01	0.13

Table 4: MAE of the predictor g trained for different compression algorithms for the ImageNet setup.

pre-trained models. To evaluate this, we split the 6 models for ImageNet in a 2:1 train-test ratio. We report the average the performance across multiple random splits. The in the ImageNet setup do not match those for the CIFAR10 dataset because it employs E sampled from test set instead of using a diffusion model. Moreover, it lacks sufficient training data due to the more constrained train-test split, fewer architectures, and fewer compression levels. Nevertheless, the ImageNet setup obtains an 8% top-1 error when eval data are sampled from test data, which is only slightly worse than the 6% top-1 error obtained for the CIFAR10 setup.

New data. This case occurs when the evaluation data used for meta-feature computation during training is not available at the testing phase. We are still giving recommendations for new architectures as in the previous case. The obtained results for ImageNet setup show only slight drop in performance in this case, similar to the results presented earlier for the CIFAR10 setup.

New compression methods. This considers generalization to new compression methods while giving recommendations for pre-trained models already seen during the meta-training phase. To evaluate this, we split the set of compression specifications into train set and test set based on the compression method used, and keep the same set of pre-trained classifiers in both the train set and the test set. For both setups, Figure 6 reveals considerable drop in top-1 error and accuracy but a marginal drop in top-5 accuracy compared to the case of new architectures, same as the trend also observed observed in the CIFAR10 setup.

B Additional results

This section presents additional results that were obtained by considering both the CIFAR10 and the ImageNet datasets.

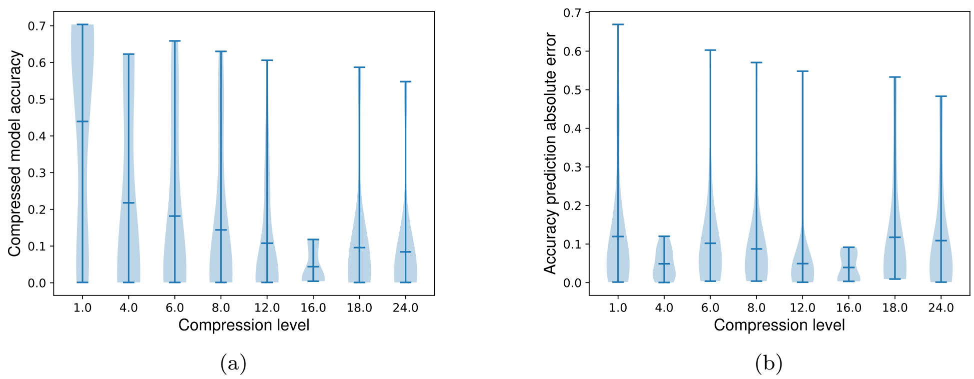


Figure 8: (a) Compressed model accuracy and (b) absolute error in accuracy prediction for different compression levels for the ImageNet setup.

Eval data selection	T5 Accuracy	T1 Accuracy	T1 Error	MAE
From test data	0.78	0.77	0.08	0.12
From training data	0.35	0.14	0.30	0.17

Table 5: Recommendation performance for different data selection strategies with the ImageNet setup.

B.1 Meta Prediction Features

The learned accuracy prediction model g can offers valuable insights about the specific meta features that contribute the most towards accurate compressed model accuracy predictions. Specifically, decision trees allow us to compute feature importance by using the Breiman equation (Hastie et al., 2009), which are presented later in this section. First of all, we provide additional details about the configuration of the decision tree model and of the considered features.

XGBoost meta prediction. The XGBoost model is configured with 100 estimators (i.e., the number of trees) and a maximum depth of 10 for each tree. The features are preprocessed as follows before being fed to the XGBoost model.

Categorical Feature Encoding. The raw data needs to be converted into a format that can be effectively used by the XGBoost model. To achieve this, we encoded the categorical features – including dataset identifier, compression method identifier – through one-hot encoding.

Numerical Features. Numerical features including loss and accuracy of the pre-trained classifier, gradient norms were incorporated after scaling them between 0 and 1. The number of architecture parameters was divided into 10 bins and linearly mapped onto the [0,1] interval.

Feature importance results. Figure 9 illustrates the feature importance. The results show that the prevalent features include the tags of a few compression algorithms, followed by the loss and accuracy of the pre-trained classifier. The target sparsity level is a stronger predictor than target quantization level, which is related to using more sparsity levels and fewer quantization levels in the considered configurations. Interestingly, the aggregate gradient metrics such as the L_0 and L_1 norm of gradients are more important than the number of parameters in the pre-trained model. This supports the observation that the slope of the solution learned after pre-training offers meaningful insights about the compression performance of the model. We also experimented with using more descriptive architecture features as well as the largest eigenvalues of Hessian to extract second-order derivative information; however, we observed no significant improvement in the results.

New data	New architectures	New compr. methods	T5 Accuracy	T1 Accuracy	T1 Error	MAE
No	Yes	No	0.78	0.77	0.08	0.12
Yes	Yes	No	0.78	0.77	0.09	0.13
No	No	Yes	0.74	0.51	0.14	0.14
Yes	No	Yes	0.74	0.51	0.15	0.14

Table 6: Generalization performance of g to new data, architectures, and compression methods for ImageNet setup.

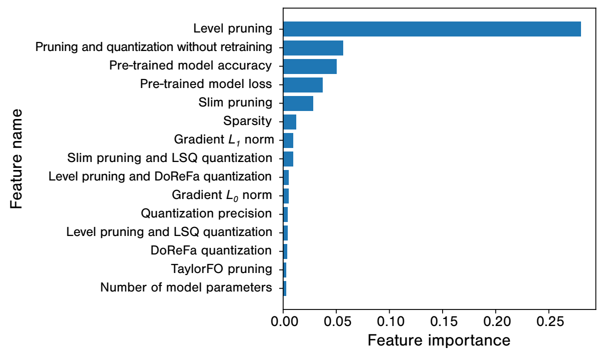


Figure 9: Feature importance for the meta predictor in the combined CIFAR10 and ImageNet setup.

Metric	T1 Accuracy	T1 Error	MAE	Kendall Tau
Meta Compression	1.00	0.00	0.13	0.51

Table 7: Meta recommendation and meta prediction performance for the California housing regression experiment.

B.2 Compute cost

We divide the compute cost into the two main steps entailed by meta compression, namely, meta training and meta recommendation.

Meta training cost. One iteration of the training loop for a single pruning and quantization level (40 retraining epochs) on the ImageNet takes approximately 300 minutes using an AMD MI250x GPU. This process is performed for 10 different sparsity levels and 2 different quantization levels, resulting in a total of 20 combinations and taking 100 hours. Learning across 6 architectures with 6 AMD MI250x GPUs running in parallel takes 4 days (equivalent to 24 GPU days).

Meta recommendation cost. Obtaining meta recommendation given a set of pre-trained models and compression constraints involves extracting meta features such as pre-trained model accuracy, which requires running inference on test data using the pre-trained models taking up to a few minutes (5 minutes for CIFAR10 setup and 20 minutes for ImageNet setup). Predicting compressed model accuracy through the extracted meta features only takes 0.002 seconds per compression level and compression method using the XGBoost model. In contrast, exhaustive search costs several GPU days.

C Application to regression tasks

We finally evaluate applying meta compression to regression tasks to further demonstrate its applicability to diverse scenarios. For this purpose, we set up a smaller experiment with four benchmark architectures from (Gorishniy et al., 2021), namely, DCNv2, SNN, ResNet, and MLP. We employ the California housing regression dataset (Pace & Barry, 1997), a ratio of 3:1 between train and test data, 12 different compression levels, and consider the compressed regressor RMSE prediction instead of the compressed classifier accuracy prediction. Separate data splits were used for training, evaluation, and testing in the commonly used 64:16:20 ratio. RMSE values were normalized for convenience, as they are less than 0.51 for all pre-trained models.

Table 7 reports the obtained results. Meta compression achieves 100% top-1 recommendation accuracy when predicting the regression performance of a new architecture, clearly demonstrating the applicability of meta compression to the tabular datasets as well.