FRANDI: DATA-FREE NEURAL NETWORK COMPRESSION VIA FEATURE REGRESSION AND DEEP INVERSION

Konstantin Sobolev[1,2], **Dmitry Ermilov**[1], **Nikolay Kozyrskiy**[1],
Anh-Huy Phan[1], **Andrzej Cichocki**
[1]Skolkovo Institute of Science and Technology, *Moscow, Russia*
[2]AIRI, *Moscow, Russia*
konstantin.sobolev@skoltech.ru

ABSTRACT

Contemporary post-training neural network compression methods make a model lighter and faster without a significant drop in performance. However, these methods heavily depend on the model's training data which might be unavailable in practical scenarios. In this work, we present *FRanDI*, a novel framework to enable post-training neural networks compression without data. Our method leverages the DeepInversion-based approach to generate synthetic data from the pre-trained model. We propose a compressed network degradation teacher-student based recovery scheme called *Feature Regression*. In addition, we present a new proxy metric that correlates with the original model's target metric to evaluate model compression policies called *Output Discrepancy*. Our algorithm does not depend on the neural network's target task compared to other data-free methods. We evaluate our framework on three different neural network compression approaches: low-rank weight approximation, unstructured pruning, and quantization.

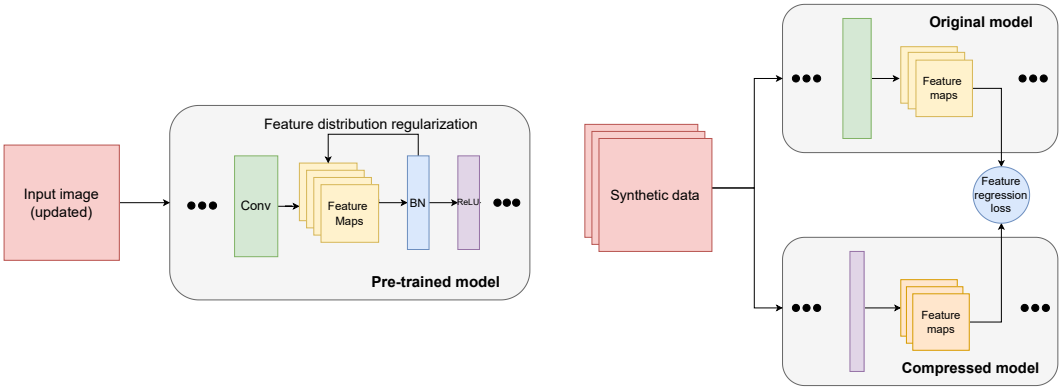
1 INTRODUCTION

Neural Network (NN) compression is a widely used technique aimed at reducing resource consumption and accelerating heavy pre-trained models, all while maintaining a negligible loss in model quality. Common methods for NN compression include: structured pruning Molchanov et al. (2019), unstructured pruning Han et al. (2015a); Figurnov et al. (2016); Molchanov et al. (2017), quantization Nagel et al. (2019) and weight low-rank approximation Lebedev et al. (2015); Phan et al. (2020).

A typical NN compression pipeline is heavily reliant on the original model's training data, which is crucial for selecting the compression ratio for different layers and for recovering from performance degradation through fine-tuning and statistical calibration. However, in practical situations, this reliance can pose challenges, especially when the dataset is unavailable due to privacy, security, or transmission constraints.

To this end, recent studies have aimed to enable the acceleration and compression of neural networks without relying on data. One of the pioneering works Lopes et al. (2017) proposed using aggregated activations from the training dataset for knowledge distillation. In another study Yoo et al. (2019), the authors introduced two auxiliary architectures, KEGNET, to extract knowledge from a pre-trained model. Chen et al. (2019) utilized a generator to train a student network in an adversarial manner. Additionally, two contemporary works Haroush et al. (2020); Yin et al. (2020) synthesized input images to produce internal statistics or high output responses for selected classes. While the aforementioned methods effectively compress neural networks without requiring data, they remain linked to a particular neural network target task, often necessitating custom loss functions to extract knowledge from the pre-trained network. As demonstrated by Chawla et al. (2021), extending these methods to other tasks can require significant effort.

In this work, we propose *FRanDI* method for data-free neural network compression (Figure 1) that is not tied to a specific neural network architecture. Our data generation method is based on Deep



(a) Synthetic data generation overview: We calculate the distance in feature distribution between original and synthetic data across multiple feature maps. This loss is used to optimize the input image, aligning its feature statistics with those of the original data.

(b) Feature Regression overview: We propagate generated data through the original and compressed models to compute Feature Discrepancy between their feature maps, and this loss is used to optimize the compressed model's parameters.

Figure 1: Visualization of proposed Data-Free Neural Network Compression method.

Inversion Yin et al. (2020) and uses the original model's statistics and distance between the original and compressed model to generate input data (Figure 1a). We propose the Feature Regression method, which exploits the teacher-student training approach on synthetized data (Figure 1b). Feature Regression minimizes the distance between corresponding layers in the original and compressed model to reduce the degradation of the compressed model. We further show that feature distance between networks' outputs can be used as a proxy to the model's target metric in the compression ratio selection task. To demonstrate the performance of our method, we apply it to neural networks for image classification and semantic segmentation combined with different model compression methods like low-rank weight approximation, unstructured pruning and quantization.

2 Related Works

Neural Network Compression and Acceleration. Extensive research has focused on accelerating neural networks through compression methods that reduce redundancy within their structures. Weight sparsification or unstructured pruning compresses neural networks by eliminating individual weights deemed to be of low importance Han et al. (2015a), enabling a size reduction of up to 20× without significant loss in performance. Structural pruning Molchanov et al. (2019) extends this approach by targeting entire structural elements of the network, such as filters or channels. This technique facilitates simultaneous speedup and compression of neural networks without the need for specialized software. Methods based on low-rank approximation take advantage of the fact that the weights of neural networks often reside in a low-rank linear space Denil et al. (2013). These methods approximate layer weights through low-rank matrix or tensor decomposition, replacing the original layers with lightweight factorized layers Lebedev et al. (2015); Kim et al. (2016); Sobolev et al. (2022); Phan et al. (2024), thereby enhancing both speed and efficiency. Quantization decrease the model's latency and memory footprint by reducing redundancy in the representations of weights and activations by approximating them with lower-precision numbers Nagel et al. (2019).

Knowledge Distillation. An alternative to model compression is the approach of using the outputs of a large neural *teacher network* to facilitate the training of a smaller *student network* initially proposed in Hinton et al. (2015). In this work, the authors employed KL-divergence between the outputs of the teacher and student networks, with an increased temperature applied to the SoftMax function. To enhance the utilization of the information embedded in the teacher network, several methods have been introduced that focus on feature distillation rather than output distillation. FitNets Romero et al. (2015) proposed using L_2 loss between the features of the teacher and student networks to guide the training of the student. Zagoruyko & Komodakis (2017) suggested transferring activation attention, which has been shown to improve the distillation process. Furthermore, Heo et al. (2019) explored

various aspects of the feature distillation process, including teacher and student transformations, the position of distillation features, and distance functions, ultimately proposing a novel feature distillation loss that significantly enhances performance.

Data-Free Neural Network Compression Methods. use synthetic data to perform fine-tuning and compression ratio selection for the original neural network. A series of works apply model-based, which uses the auxiliary network for data generation. Chen et al. (2019) proposed to train student using the adversarial approach, where a generator is trained to produce samples that maximize the distance between the outputs of the teacher and student networks, while the student is trained to minimize this distance. Similarly, Micaelli & Storkey (2019) adopted this setup, adding an L_2 loss between the intermediate spatial attention maps of the teacher and student networks, as suggested in Zagoruyko & Komodakis (2017). Additionally, Zhang et al. (2021) combined an adversarial approach with progressive growing and reconstruction loss to compress a super-resolution model without relying on data. Meanwhile, Yoo et al. (2019) introduced an alternative framework called KEGNET, consisting of two networks—a generator and a decoder—that learn to sample data based on dataset labels. Another approach involves network inversion techniques. Two concurrent studies Haroush et al. (2020) and Yin et al. (2020) proposed a method that updates a noise image to maximize predefined class probabilities while minimizing the distance between intermediate activation statistics and those of the training set, preserved in Batch Normalization (BN) layers. Furthermore, Yin et al. (2020) incorporated adversarial loss between the teacher and student networks. Both methods successfully generated realistic datasets for image recognition without using actual data, which can then be leveraged to distill knowledge into a smaller or compressed model. To extend DeepInversion Yin et al. (2020) to object detection tasks, Chawla et al. (2021) applied a comprehensive set of differentiable augmentations along with a novel automated scheme for bounding box and category sampling.

3 METHOD

In this paper, we focus on post-training model compression. A typical pipeline for this process consists of the following steps: (i) selecting a compression scheme (i.e., per-layer compression ratio); (ii) compressing the model (through pruning, quantization, or low-rank weight approximation); and (iii) recovering the model's performance degradation by fine-tuning the compressed model on the training dataset. This process can be conducted iteratively.

Most neural network compression methods extensively depend on the original training dataset throughout various stages of the pipeline: for evaluating the compression scheme, during the compression process, and, most importantly, during the fine-tuning phase. Our proposed method, *FRanDI*, addresses this limitation. In Section 3.1, we detail our synthetic data generation approach. Section 3.2 introduces a degradation recovery scheme utilizing the generated synthetic data as a substitute for fine-tuning. Lastly, in Section 3.3, we present a pipeline for evaluating the compression scheme with synthetic data.

3.1 SYNTHETIC DATA GENERATION

To resolve the problem of missing data, we apply a *DeepInversion*-based scheme for synthetic data generation Yin et al. (2020); Haroush et al. (2020). Given the original neural network F_W with weights W, we aim to generate a synthetic dataset. Overall, the image generation scheme can be described as an optimization problem: $\min_{\hat{x}} \mathcal{L}(\hat{x})$,, where $\hat{x} \in \mathcal{R}^{H \times W \times C}$ is a randomly initialized input of the neural network, H, W, C are height, width and number of channels respectively (Figure 1a). $\mathcal{L}(\cdot)$ encompasses a composite loss that includes input data regularizers and a discrepancy loss that measures the difference between the outputs of the compressed and original models. In Section 4.1, we evaluate various combinations of these loss components, demonstrating that utilizing Batch-Norm statistics for feature loss yields the best results.

BN-Statistics Loss. Following the approaches in Haroush et al. (2020) and Yin et al. (2020), our method utilizes feature regularization $\mathcal{R}_{BN}$ to ensure that the deep feature statistics of the synthetic image $\hat{x} \in \hat{X}$ closely resemble those of the original dataset. These statistics are captured in the Batch Normalization (BN) layers of the original model, enabling the generation of synthetic samples $\hat{x} \in \hat{X}$ that are as similar as possible to the actual data.

Formally, we calculate the statistics $\hat{\mu} = \mu(\hat{x})$ and $\hat{\sigma} = \sigma(\hat{x})$ for input features of $\hat{x}$ within the BN layer. Concurrently, we maintain running estimates of the original dataset's input feature statistics, denoted as μ^* and σ^*. Assuming that the intermediate features in the neural network follow a Normal distribution, we can assess the distance between the feature distributions of the original and synthetic data using Kullback-Leibler Divergence (KLD). Our experiments demonstrate that regularizing the distribution of intermediate features is sufficient for effective synthetic data generation in our pipeline. Consequently, the image generation loss can be expressed as:

$$\begin{aligned}\mathcal{L}(\hat{x}) = \mathcal{R}_{BN}(\hat{x}) &= \sum_{l=1}^{L} KL\left(\mathcal{N}\left(\hat{\mu}_l, \hat{\sigma}_l^2\right) \| \mathcal{N}\left(\mu_l^*, {\sigma_l^*}^2\right)\right) \\ &= \sum_{l=1}^{L}\sum_{i=1}^{N_l}\left(\log\frac{\hat{\sigma}_{l,i}}{\sigma_{l,i}^*} - \frac{1}{2}\left(1 - \frac{{\sigma_{l,i}^*}^2 + (\mu_{l,i}^* - \hat{\mu}_{l,i})^2}{\hat{\sigma}_{l,i}^2}\right)\right),\end{aligned} \tag{1}$$

where L is number of BN layers in the neural network, N_l is l-th channel in l-th layer.

3.2 Degradation Recovery Scheme

We consider individual layer compression as a representation of layer weights in a compact format:

$$w_{original} = w_{compact} + w_{redundant}, \tag{2}$$

where $w_{compact}$ is sparse, quantized, or factorized representation of weights tensor, $w_{redundant}$ is an unimportant weight component that is removed during compression procedure (e.g., pruned weights, high-bit part of weights, or approximation error).

Since $w_{redundant}$ is non-zero, replacement of $w_{original}$ by $w_{compact}$ results in compressed network's feature-map distortion:

$$f^{w_{compact}}(x) = f^{w_{original}}(x) - f^{w_{redundant}}(x) = f^{w_{original}}(x) + E, \tag{3}$$

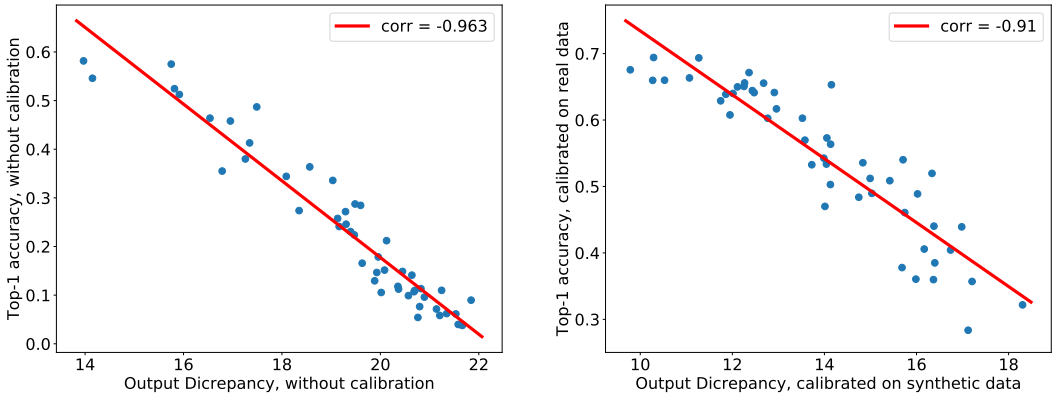
where x and f^w are input data and operation performed by layer with weight w, respectively.

Assume, compressed model $\tilde{F}_{\tilde{W}}$ is obtained by compressing N layers of original model F_W. In multi-layer compression, feature map distortions tend to increase with depth due to error accumulation from earlier layers to deeper ones. These distortions can lead to significant degradation in model performance. To mitigate this performance loss, conventional neural network compression processes typically involve a fine-tuning step. This step involves retraining the compressed model on the original dataset using a low learning rate. However, in a data-free scenario, fine-tuning becomes challenging due to the unavailability of data and labels.

Feature Regression. We propose to address the model degradation by minimizing the feature distortion between the original and compressed models. To achieve this, we formulate the recovery process as a feature-based teacher-student training framework (Figure 1b). In this framework, the student (compressed) model aims to approximate the intermediate features of the teacher (original) model:

$$\min_{\tilde{W}} L_{FR}(F_W(\hat{x}), F^*(\tilde{W})(\hat{x})) = \min_{\tilde{W}} \sum_{i=1}^{N} L_{FD}(f_i^{w_i}(\hat{x}), \tilde{f}_i^{\tilde{w}_i}(\hat{x})), \tag{4}$$

where $f_i^{w_i}(\hat{x})$ and $\tilde{f}_i^{\tilde{w}_i}(\hat{x})$ are outputs of i-th uncompressed and compressed layers given generated image $\hat{x}$, $F_W(\hat{x})$ and $\tilde{F}_{\tilde{W}}(\hat{x})$ are outputs of uncompressed and compressed models, L_{FD} is a loss function that defines distance between feature maps. In our work, we use Frobenius norm of distance between normalized feature-maps, we call it *Feature Discrepancy* (**FD**):



(a) Without BN calibration

(b) With BN calibration

Figure 2: Compressed model accuracy vs **OD** proxy metric: (a) without calibration, (b) accuracy measured after calibration on real data, proxy metric measured after calibration on synthetic data. The plot is based on the 50 different compressed using Spatial-SVD (with different rank sets and fixed 4x FLOPs compression ratio) *Resnet-18* networks, *Cifar-100* dataset. Compressed model accuracy correlates with **OD** in both cases.

$$L_{FD}(f_i^{w_i}(\hat{x}), \tilde{f}_i^{\tilde{w}_i}(\hat{x})) = \left\| \frac{f_i^{w_i}(\hat{x})}{\|f_i^{w_i}(\hat{x})\|_2} - \frac{\tilde{f}_i^{\tilde{w}_i}(\hat{x})}{\left\|\tilde{f}_i^{\tilde{w}_i}(\hat{x})\right\|_2} \right\|_2 . \tag{5}$$

Using feature regression instead of regular fine-tuning allows us to avoid generating labeled data and make our method independent from the task of the original model.

3.3 Model Selection Approach

Neural network compression methods typically rely on target metrics (e.g., accuracy) to select hyperparameters that balance model size and performance, such as sparsity, pruning ratio, and decomposition rank. However, since our method does not generate labels for the synthetic dataset, we are unable to reproduce model metrics on this data. To address this limitation, we use *Output Discrepancy* (**OD**) between the compressed and uncompressed networks as an alternative proxy metric for evaluating model performance:

$$\mathbf{OD}(F_W(\hat{x}), \tilde{F}_{\tilde{W}}(\hat{x})) = \left\| \frac{F_W(\hat{x})}{\|F_W(\hat{x})\|_2} - \frac{\tilde{F}_{\tilde{W}}(\hat{x})}{\|\tilde{F}_{\tilde{W}}(\hat{x})\|_2} \right\|_2 , \tag{6}$$

where $\hat{x}$, $F_W(\hat{x})$ and $F^*(\hat{W}^*)(\hat{x})$ represent the outputs of the uncompressed and compressed models for a batch of synthesized images $\hat{x}$. Our experimental results demonstrate that this function correlates with the performance metrics of the original model and can serve as a proxy metric for selecting hyperparameters in model compression (see Figure 2a). A recent work Li et al. (2020) has shown that evaluating a compressed model after calibrating BatchNorm statistics (adaptive batch normalization) significantly enhances its correlation with final accuracy post-fine-tuning. In our experiments, we find that the proposed **OD** metric, after calibrating the statistics, displays a strong correlation with model accuracy following BatchNorm calibration (see Figure 2b).

We aim to find an optimal vector $\tilde{R} = (\tilde{r_1}, ..., \tilde{r_L})$, in which r_i denotes compression ratio of i-th layer in L-layer neural network given a global reduction of operations (FLOPs) or parameters equal to α. With **OD** proxy metric, we formulate compression ratio search procedure as the following problem:

Problem 1 (Optimal Compression Ratio Search)

$$\begin{aligned} &\underset{R=(r_1,\ldots,r_L)}{\textit{minimize}} && \textbf{\textit{OD}}(F_W(\hat{x}), F^*(\tilde{F}_{\tilde{W}_R}(\hat{x})) \\ &\textit{subject to} && (\alpha - \Delta\alpha)C_{orig} \leq C(R) \leq (\alpha + \Delta\alpha)C_{orig} \end{aligned} \tag{7}$$

where $C(R)$ is a complexity of the model compressed with parameters $R = (r_1, ..., r_L)$, C_{orig} is complexity of the original model, $\Delta\alpha$ is allowed deviation from compression complexity constraint α. In our model selection pipeline, we use a simple random sampling approach to generate candidates for model compression. Concretely, it randomly samples L real numbers from a given range $[r_{min}; r_{max}]$ to form a compression strategy that satisfies model complexity constraint used in the Problem 1. After sampling N candidates, we evaluate them and select one with the lowest **OD** score. Noticeably, other more sophisticated search procedures can be applied to find the best set of compression ratios, such as reinforcement learning, bayesian search, evolutionary algorithm, etc.

4 Ablation Studies

In this section, we evaluate how different components of our method affect the overall results. For this purpose, we compress *ResNet-18* trained on *CIFAR-100* dataset (classification, 100 classes, 32×32 image resolution) with *Spatial-SVD* weight low-rank approximation scheme Kuzmin et al. (2019). Original model has 77.1% top-1 accuracy and its FLOPs compression ratio is fixed to 5 (decomposition ranks are provided in supplementary materials). In all experiments Feature Regression was performed by Stochastic Gradient Descent with 10^{-4} learning rate, 0.9 momentum, and 10^{-4} weight decay. Synthetic data generation is performed by 500 iterations of Adam optimizer with 0.1 learning rate and $(\beta_1, \beta_2) = (0.5, 0.9)$.

4.1 Synthetic Image Generation

Image Regularization. Our data generation approach is a simplification of *DeepIversion*-based methods proposed in Haroush et al. (2020) and Yin et al. (2020). We consciously decided to abandon using image classes for data generation to make the method task-independent in our version. In addition, Haroush et al. (2020) uses image regularization loss term (total variance, l_2 norm of image) to improve image optimization process. We evaluate these regularizations in combination with BN-Statistics loss. Results in Table 1 show that the image regularizations do not provide any statistically significant improvement in the target metric.

Table 1: Evaluation of image regularizations *ResNet-18* on *CIFAR-100* dataset. We run each experiment 5 times with different random seeds to estimate the measurement error.

TV	L_2	Top1 Accuracy
✗	✗	**73.74 ± 0.14**
✗	✓	73.61 ± 0.15
✓	✗	73.53 ± 0.18
✓	✓	73.48 ± 0.16

Generated Dataset Size. We evaluate how the size of generated dataset impacts the convergence process of the model. Our experiments show that one batch of 256 images is sufficient for the optimal Feature Regression procedure (See Figure 3). In addition, it is worth noting that convergence happens in approximately 1000 of gradient steps, which is comparable to 4 epochs of fine-tuning on *CIFAR-100* dataset.

4.2 Feature Regression

Feature Loss Function. We evaluate different feature-based knowledge distillation losses (See Table 2). FitNets Romero et al. (2015) and OFD Heo et al. (2019) computes L_2 distance and Partial L_2 distance between teacher and student networks. In the case of Feature Regression, first gradient updates might be unstable due to the high distance between intermediate features of original and compressed models. This leads to a failure of Feature Regression process, as can be seen in Table 2. AT Zagoruyko & Komodakis (2017) loss shows more stable performance during degradation recovery since it computes the distance between normalized spatial attention maps. In **FD** loss we also use normalization to stabilize training; we combine its Frobenius norm of the difference between feature maps which provides natural normalization of gradients.

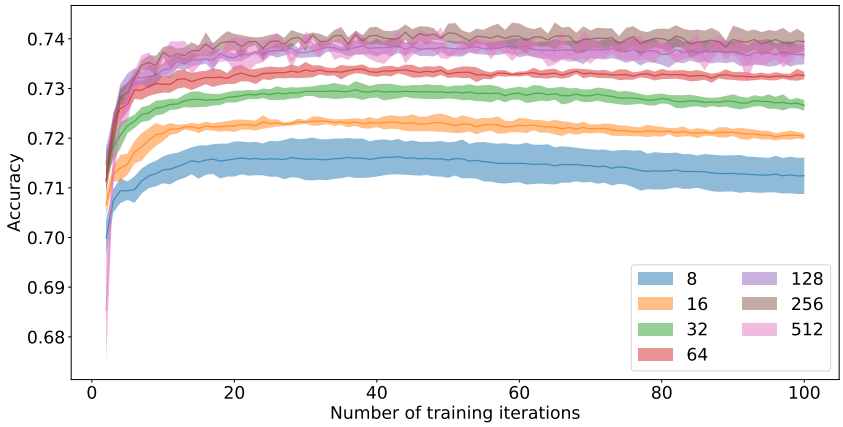


Figure 3: Visualization of Feature Regression process with different generated dataset size. One training iteration is equal to 10 gradient steps. Batch size is during feature regression is equal to dataset size. We run each experiment 5 times with different random seeds to estimate the measurement error.

Table 2: Comparison of different Feature Regression losses.

Loss type	Feature Transform	Distance	Position	Accuracy, %
FitNets	None	L_2	Conv	1 (failed)
FitNets	None	L_2	BN	1 (failed)
AT	Attention	L_2	Conv	72.04
AT	Attention	L_2	BN	72.23
OFD	Margin ReLU	Partial L_2	BN	1 (failed)
FD (proposed)	Normalization	Frobenius	Conv	**73.66**

Model Convergence. We compare fine-tuning of compressed model on real data with our data-free Feature Regression approach. Figure 4 evidences that for *CIFAR-100* dataset Feature Regression has comparable efficiency as ordinary fine-tuning with real training data. Training curve of Feature Regression converges faster and has more stable character.

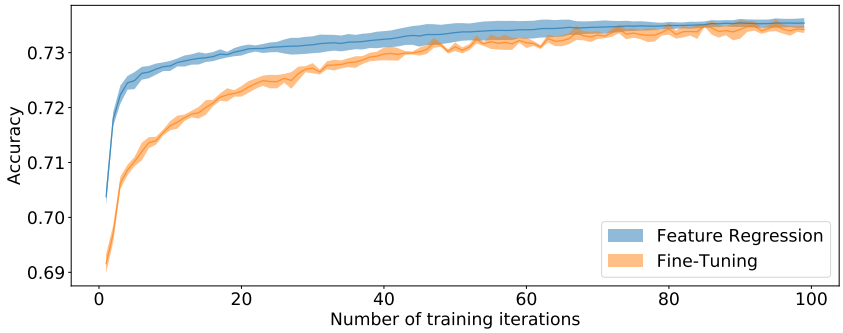


Figure 4: Visualization of model degradation recovery after compression with different generated dataset size. One training iteration is equal to 10 gradient steps. We run each experiment 5 times with different random seeds to estimate the measurement error.

5 Experiments

We demonstrate our data-free model compression approach on various datasets and models with different sizes and complexity. Firtsly, we evaluate combination of different model compression approaches (low-rank weight approximation, unstructured pruning, structured pruning and quanti-

zation) combined with our data-free regime on the complex *ImageNet* dataset (classification, 1000 classes, 224×224 image resolution) and compare our results with other compression methods. Secondly, we test our approach on non-classification task - semantic segmentation. We used a pre-trained *ImageNet* model shipped with *Torchvision*: *ResNet-18* with 69.8 and 89.1 top-1 and top-5 accuracies, *VGG-16* with *BN* – 73.4 and 91.5 top-1 and top-5 accuracies and textitResNet-50 – 76.1 and 92.9 top-1 and top-5 accuracies. For *Cifar-100* experiments, we use a pre-trained *ResNet-18* model with 77.1% top-1 accuracy.

5.1 Classification Results

Quantization Experiments. We compare our method against various data-free quantization techniques for 4-bit quantization of model weights and activations (w4a4) in ResNet-18 on the CIFAR-100 dataset. This includes ZeroQ Cai et al. (2020), GDFQ Shoukai et al. (2020), DFQ Nagel et al. (2019), ACIQ Banner et al. (2019), and ZAQ Liu et al. (2021). Table 3 demonstrates that our method achieves a higher top-1 accuracy within this benchmark compared to prior methods. Section A.3 in appendix shows results for other quantization setups.

Table 3: Comparison with state of the art data-free quantization methods. *ResNet-18* model, *Cifar100* dataset.

Method	Settings	Top-1 acc.
ZeroQ	w4a4	70.25
GDFQ		71.53
DFQ		40.35
ACIQ		54.73
ZAQ		72.67
FRanDI (Ours)		**75.90**

Unstructured Pruning. For experiments with unstructured pruning, we use a magnitude-based approach Han et al. (2015b) in a one-shot fashion. Prune rates for each layer are obtained using random sampling with a constraint on a number of non-zero model parameters. To select the best prune model, *Output Discrepancy* between candidates and the original model is evaluated. To recover prune model performance, Feature Regression was applied for 400 iterations while fine-tuning on an original dataset for 200 iterations. Results are presented in Table 4.

Table 4: Results for unstructured pruning with magnitude-based approach. CR - compression ratio, ratio of non-zero parameters in the model. *Original* denotes accuracy of original model, *Fine-tuned* - accuracy after pruning and fine-tuning on original dataset, *Recovered* - accuracy after pruning and Feature Regression.

Model	Dataset	CR	BatchSize	Top-1 Accuracy, %		
				Original	Fine-tuned	Recovered
ResNet-18	Cifar-100	0.5	256	77.10	76.12	76.62
ResNet-18	ImageNet	0.8	256	69.76	69.16	69.20
ResNet-50	ImageNet	0.5	128	76.13	72.23	72.81

Low-Rank Weight Approximation. We evaluate our data-free pipeline alongside low-rank neural network compression using *Spatial-SVD* convolutional layer weight factorization Kuzmin et al. (2019). For each model, we generate synthetic data individually. Our experiments show that a single batch is sufficient for model convergence during Feature Regression. Specifically, we generate 128 images for *VGG-16* and *ResNet-50*, while *ResNet-18* utilizes 256 images. To determine the optimal rank set for each model, we sample 500 candidates at a specified compression ratio and select the one with the minimal **OD** metric, as outlined in Subsection 3.3. In our experiments, the

Table 5: Results for low-rank neural network compression using *Spatial-SVD* on *ImageNet* dataset. *Compressed* denotes accuracy after model compression, *Calibrated* - accuracy after BatchNorm calibration on synthetic data, *Recovered* - accuracy after Feature Regression.

Model	↓FLOPs	*Compressed*		*Calibrated*		*Recovered*	
		Top 1 acc.	Top 5 acc.	Top 1 acc.	Top 5 acc.	Top 1 acc.	Top 5 acc.
VGG-16	4	27.2	51.0	50.2	75.5	65.1	86.7
ResNet-18	2.1	39.1	64.1	57.1	80.6	65.4	86.6
ResNet-50	2	32.9	56.9	61.0	83.8	70.2	89.8

recovery of performance degradation in compressed models converged within just 500 iterations of Feature Regression using the SGD optimizer, with an initial learning rate of 10^{-3} and weight decay set to 10^{-4}. Table 5 presents the results of our *ImageNet* model compression, demonstrating that our method successfully recovers between 25% to 37% of the top-1 accuracy drop following model compression.

Mixed Mode Experiments. In certain cases of neural network compression, a portion of the training data may be accessible. In this context, we can integrate our synthetic pipeline with the original data. Table 6 illustrates how fine-tuning on a subset of the data can impact the performance of a model previously reconstructed using Feature Regression (each experiment repeated 5 times to estimate measurement error.). Starting with just 5% of the original dataset, post-Feature Regression fine-tuning enhances the accuracy of the compressed model. Prior to fine-tuning, the model recovered via Feature Regression achieved a Top-1 Accuracy of 73.81% on the *Cifar-100* dataset. This model was compressed using Spatial-SVD, resulting in a fivefold reduction in FLOPs, while the original model had a Top-1 Accuracy of 77.1%.

Table 6: Accuracy of the fine-tuned model on a subset of the original dataset after Feature Regression.

Dataset, %	Top 1 accuracy, %
1	73.55 ± 0.03
2	73.64 ± 0.03
5	73.94 ± 0.07
10	74.00 ± 0.08
20	74.17 ± 0.06
33	74.32 ± 0.07
50	74.30 ± 0.12
100	74.41 ± 0.09

5.2 Semantic Segmentation Results

For semantic segmentation, we use model DeepLabV3+, which is trained on PASCAL VOC 2012 (21 class, 513×513 image resolution). For this model, we generate a batch with only 16 synthetic images. The original model has 75.88 mIoU. As we can see from Table 7, after compression, model's performance drops severely. Our method allows recovery of up to 55 mIoU.

Table 7: Results for low-rank neural network compression using *Spatial-SVD* on *PASCAL VOC 2012* dataset. *Compressed* denotes accuracy after model compression, *Calibrated* - accuracy after BatchNorm calibration on synthetic data, *Recovered* - accuracy after Feature Regression.

Model	FLOPs	Compressed	Calibrated	Recovered
		mIoU	mIoU	mIoU
DeepLabV3+	2.5	15.25	54.83	65.10
	3	5.79	37.98	58.25

6 Conclusion

In this paper, we propose a novel framework for Data-Free model compression. It enables all components of an ordinary post-training neural network compression pipeline without training data: selection of compression parameters, model calibration, and degradation recovery. To this end, we propose a novel *Output Discrepancy* metric for compressed models ranking and teacher-student based training *Feature Regression* approach for compressed model degradation recovery. For synthetic data generation, we use a simplified *DeepInversion* method. Our approach allows performing model compression using only one batch of synthetic data with convergence in just 500 iterations of SGD. Since our method does not use data labels, it can be applied to an arbitrary architecture and task.

Acknowledgments

This work was partially supported by the joint project Artificial Intelligence for Life (AIfoL) between the University of Sharjah and the Skolkovo Institute of Science and Technology.

REFERENCES

Ron Banner, Yury Nahshan, Elad Hoffer, and Daniel Soudry. ACIQ: Analytical clipping for integer quantization of neural networks, 2019. URL https://openreview.net/forum?id=B1x33sC9KQ.

Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Zeroq: A novel zero shot quantization framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13169–13178, 2020.

Akshay Chawla, Hongxu Yin, Pavlo Molchanov, and Jose Alvarez. Data-free knowledge distillation for object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 3289–3298, January 2021.

Hanting Chen, Yunhe Wang, Chang Xu, Zhaohui Yang, Chuanjian Liu, Boxin Shi, Chunjing Xu, Chao Xu, and Qi Tian. Dafl: Data-free learning of student networks. In *ICCV*, 2019.

Misha Denil, Babak Shakibi, Laurent Dinh, Marc' Aurelio Ranzato, and Nando de Freitas. Predicting parameters in deep learning. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper/2013/file/7fec306d1e665bc9c748b5d2b99a6e97-Paper.pdf.

Mikhail Figurnov, Aizhan Ibraimova, Dmitry P Vetrov, and Pushmeet Kohli. PerforatedCNNs: Acceleration through elimination of redundant convolutions. In *Advances in Neural Information Processing Systems*, pp. 947–955, 2016.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 28*, pp. 1135–1143. 2015a.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1135–1143, 2015b.

Matan Haroush, Itay Hubara, Elad Hoffer, and Daniel Soudry. The knowledge within: Methods for data-free model compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Byeongho Heo, Jeesoo Kim, Sangdoo Yun, Hyojin Park, Nojun Kwak, and Jin Young Choi. A comprehensive overhaul of feature distillation. In *International Conference on Computer Vision (ICCV)*, 2019.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. URL http://arxiv.org/abs/1503.02531.

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 Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.06530.

Andrey Kuzmin, Markus Nagel, Saurabh Pitre, Sandeep Pendyam, Tijmen Blankevoort, and Max Welling. Taxonomy and evaluation of structured compression of convolutional neural networks. *CoRR*, abs/1912.09802, 2019. URL http://arxiv.org/abs/1912.09802.

Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned CP-decomposition. *International Conference on Learning Representations*, 2015.

Bailin Li, Bowen Wu, Jiang Su, and Guangrun Wang. Eagleeye: Fast sub-net evaluation for efficient neural network pruning. In *European conference on computer vision*, pp. 639–654. Springer, 2020.

Yuang Liu, Wei Zhang, and Jun Wang. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021.

Raphael Gontijo Lopes, Stefano Fenu, and Thad Starner. Data-free knowledge distillation for deep neural networks. *CoRR*, abs/1710.07535, 2017. URL `http://arxiv.org/abs/1710.07535`.

Paul Micaelli and Amos J Storkey. Zero-shot knowledge transfer via adversarial belief matching. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/fe663a72b27bdc613873fbbb512f6f67-Paper.pdf`.

Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2498–2507. JMLR. org, 2017.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Anh-Huy Phan, Konstantin Sobolev, Konstantin Sozykin, Dmitry Ermilov, Julia Gusak, Petr Tichavskỳ, Valeriy Glukhov, Ivan Oseledets, and Andrzej Cichocki. Stable low-rank tensor decomposition for compression of convolutional neural network. In *European Conference on Computer Vision*, pp. 522–539. Springer, 2020.

Anh-Huy Phan, Dmitri Ermilov, Nikolay Kozyrskiy, Igor Vorona, Konstantin Sobolev, and Andrzej Cichocki. How to train your unstable looped tensor network. *IEEE Journal of Selected Topics in Signal Processing*, pp. 1–10, 2024. doi: 10.1109/JSTSP.2024.3463480.

Adriana Romero, Samira Ebrahimi Kahou, Polytechnique Montréal, Y. Bengio, Université De Montréal, Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *in International Conference on Learning Representations (ICLR*, 2015.

Xu Shoukai, Li Haokun, Zhuang Bohan, Liu Jing, Cao Jiezhang, Liang Chuangrun, and Tan Mingkui. Generative low-bitwidth data free quantization. In *The European Conference on Computer Vision*, 2020.

Konstantin Sobolev, Dmitry Ermilov, Anh-Huy Phan, and Andrzej Cichocki. Pars: Proxy-based automatic rank selection for neural network compression via low-rank weight approximation. *Mathematics*, 10(20):3801, 2022.

Hongxu Yin, Pavlo Molchanov, Jose M. Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K. Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Jaemin Yoo, Minyong Cho, Taebum Kim, and U Kang. Knowledge extraction with no observable data. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/596f713f9a7376fe90a62abaaedecc2d-Paper.pdf`.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *ICLR*, 2017. URL `https://arxiv.org/abs/1612.03928`.

Yiman Zhang, Hanting Chen, Xinghao Chen, Yiping Deng, Chunjing Xu, and Yunhe Wang. Data-free knowledge distillation for image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7852–7861, June 2021.

A APPENDIX

A.1 SYNTHETIZED DATA VISUALIZATION

In Figure 5 one can see the visualization of several images from a synthetic dataset generated by Resnet50 for the ImageNet classification task. Despite the images looking difficult to perceive, they produce the statistically close feature maps in Resnet50 for Imagenet thus applicable for the quality restoration procedure.

Figure 5: Visualization of synthetic dataset generated by Resnet50 for ImageNet classification task.

A.2 SPATIAL-SVD DETAILS

A.2.1 DECOMPOSITION DESCRIPTION

Spatial-SVD is a single-rank decomposition method that replaces initial layer by 2 layers (Figure 6): $D \times 1$ convolution that performs convolution in vertical direction projects I_{ch} input channels to R channels and $1 \times D$ convolution that performs convolution in horizontal direction projects R input channels to O_{ch} channels. The parameter and computation reduction rate is $(I_{ch}D + O_{ch}D)R/I_{ch}O_{ch}D^2$.

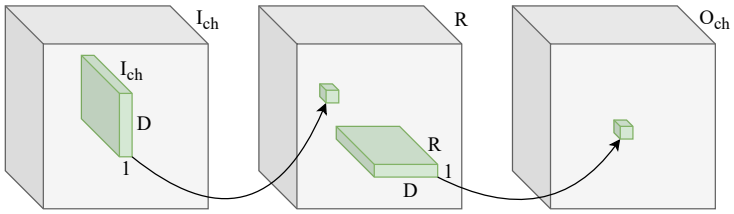


Figure 6: Visualization of Spatial-SVD layer. I_{ch} and O_{ch} denote number of input and output number of channels respectively, D denotes size of the convolutional layer, R denotes the decomposition ranks.

A.2.2 SPATIAL-SVD RANKS FOR ABLATION STUDIES MODEL

We provide details for compression of *ResNet-18* model used in ablation studies in Table 8.

Table 8: Decomposition rank for *ResNet-18* model trained on *CIFAR-100* dataset which was used in ablation studies. Original model has 77.1% top-1 accuracy. Compression ratio of decomposed model is equal to 5.

Layer	**Spatial-SVD Rank**
layer1.0.conv1	18
layer1.0.conv2	18
layer1.1.conv1	12
layer1.1.conv2	11
layer2.0.conv1	26
layer2.0.conv2	46
layer2.1.conv1	28
layer2.1.conv2	23
layer3.0.conv1	56
layer3.0.conv2	114
layer3.1.conv1	56
layer3.1.conv2	101
layer4.0.conv1	172
layer4.0.conv2	100
layer4.1.conv1	107
layer4.1.conv2	94

A.3 Quantization Results

We also provide neural network quantization experiment results with other quantization setups and recover degradation using feature regression with synthetic and real data. Results are presented in Table 9.

Table 9: Accuracy for quantized model recovery using Feature Regression: *ResNet18* for *CIFAR-10* and *CIFAR-100* datasets. FP32 *CIFAR-10* model has 95.1 % accuracy, FP32 *CIFAR-100* model has 77.1 % accuracy.

Dataset	Settings	Top-1 Accuracy, %		
		Before fine-tuning	Real Data	Synth. Data
Cifar-10	w3a3	88.4	93.1	92.4
	w4a4	94.3	94.8	94.7
	w4a8	94.7	94.9	94.9
	w8a4	94.5	94.9	94.8
Cifar-100	w3a3	63.8	72.1	70.6
	w4a4	74.4	76.2	75.9
	w4a8	75.8	76.6	76.4
	w8a4	75.8	76.4	76.3