

Supplementary Materials: PTSBench: A Comprehensive Post-Training Sparsity Benchmark Towards Algorithms and Models

Anonymous Authors

1 DETAILS OF POST-TRAINING SPARSITY ALGORITHMS

Model compression methods, including quantization[16, 21, 23, 29, 38, 46], distillation[13, 17], and sparsification[6, 7, 12, 14, 25–28, 32, 47], have been widely used in various deep learning tasks, especially in computer vision task[10, 11]. Model sparsification allows for efficient acceleration while maintaining model accuracy by simply pruning the weights. Consequently, it has increasingly garnered attention in recent years. Among all sparsification techniques, post-training sparsity (PTS) stands out for its less demand for data and no need for re-training, while other types often require training on a huge datasets[4, 7, 49].

1.1 Post-Training Sparsity

Most state-of-the-art PTS methods work by adopting a layer-wise reconstruction [8, 12, 20, 26]. In this setting, we can define the problem as follows. Mathematically, we model a layer ℓ as a function $f_\ell(x_\ell, w_\ell)$ acting on inputs x_ℓ with weights w_ℓ .

$$\begin{aligned} \arg \min_{\hat{w}_\ell} \mathcal{L}(M(f_\ell(x_\ell, w_\ell)), M(f_\ell(x_\ell, \hat{w}_\ell))), \\ \text{subject to } \|\hat{w}_\ell\|_0 \leq k. \end{aligned} \quad (1)$$

In practice, the expectation of the output activations is often used as the metric for assessment. Therefore, we can get the formula as follows.

$$\begin{aligned} \arg \min_{\hat{w}_\ell} \mathbb{E}_{x_\ell} \mathcal{L}(f_\ell(x_\ell, w_\ell), f_\ell(x_\ell, \hat{w}_\ell)), \\ \text{subject to } \|\hat{w}_\ell\|_0 \leq k. \end{aligned} \quad (2)$$

The expectation over the layer inputs x_ℓ is typically approximated by calculating the mean over a small set of N input samples. Furthermore, most previous work targets the sparsification of linear and convolutional layers, which can be represented as linear layers by unfolding them, as these types of layers are commonly utilized. In practice, the squared loss metric is employed to evaluate the error in approximation since it can be analyzed by a series of approximations, such as second-order information [37] and hessian matrix [29]. Moreover, the effectiveness of this approach has been demonstrated in many applications [9, 22, 29, 37].

By following these conventions, we can formally state the layer-wise reconstruction problem as below, where $W_\ell \in \mathbb{R}^{d_{row} \times d_{col}}$ and $X_\ell \in \mathbb{R}^{d_{col} \times N}$ are weights and activations matrices respectively.

$$\begin{aligned} \arg \min_{\hat{W}_\ell} \|\mathbf{W}_\ell \mathbf{X}_\ell - \hat{\mathbf{W}}_\ell \mathbf{X}_\ell\|_2^2, \\ \text{subject to } \|\hat{\mathbf{W}}_\ell\|_0 \leq k. \end{aligned} \quad (3)$$

In current studies, most of them followed this layer-wise paradigm, such as POT[26] and OBC [8]. Each of these methods designs

its own criteria for reconstruction based on the layer-wise reconstruction paradigm. For example, the POT is based on magnitude, while OBC relies on the Hessian matrix. However, most of these methods lack a more fine-grained exploration of the layer-wise paradigm itself. In practical application scenarios, we have identified three types of fine-grained pluggable components that can influence the effectiveness of sparsity.

1.2 Comparison with Other Model Compression Techniques

Current model compression studies mainly focus on reducing original models' size and computation complexity. Model quantization reduces the size of a model and enhances computational efficiency by lowering the precision of weights and activations within the network [3, 29, 46]. Common quantization approaches involve converting from 32-bit floating-point numbers to 8-bit integers or even lower bit depths. Model distillation involves using a pre-trained larger model (the teacher model) to guide the training of a smaller model (the student model)[13, 17]. This method leverages the knowledge from the large model to enhance the performance of the smaller model. Lightweight model design involves creating or modifying existing neural network architectures from scratch to reduce computational complexity and model size while maintaining performance[34?]. Low-rank decomposition reduces model parameters by decomposing weight matrices in neural networks into products of matrices with lower ranks[48]. Model pruning reduces the complexity of neural networks by removing some of the less important connections (weights)[12, 49].

1.3 Fine-grained PTS Algorithms

The main paper investigates several fine-grained PTS algorithms without a thorough introduction. In this section, we introduce each algorithm we benchmark in detail.

1.3.1 Sparsity Allocation. We choose 4 commonly used sparsity allocation strategies: Uniform, L2Norm, ERK, and FCPTS.

- (1) **Uniform** [36]: The sparsity s_ℓ of each independent layer is equal to the global sparsity constraint S .
- (2) **L2Norm**[26]: First, calculate the number of weights to be removed, n_s , based on the global sparsity rate S and the total number of parameters n .

$$n_s = S \cdot n \quad (4)$$

Sort all weights according to their L2 norm size. Mark the smallest n_s weights for removal. This process results in the specific sparsity rate for each layer of the network.

- (3) **ERK (Erdős-Rényi-Kernel)** [4]: This method is an improvement of origin Erdős-Rényi (ER) algorithm[35]. Origin ER algorithm modifies the sparsity s_ℓ of each layer obtained by

L2Norm as the following formula.

$$s'_\ell = s_\ell \cdot \left(1 - \frac{n_{\ell-1} + n_\ell}{n_{\ell-1} * n_\ell}\right), \quad (5)$$

where n_ℓ denotes the number of neurons at layer ℓ . Scaling the sparsity enables the number of weights in a sparse layer to scale with the sum of the number of input and output channels.

ERK enhances ER by including the kernel dimensions in the scaling factors.

$$s'_\ell = s_\ell \cdot \left(1 - \frac{n_{\ell-1} + n_\ell + w_\ell + h_\ell}{n_{\ell-1} * n_\ell * w_\ell * h_\ell}\right), \quad (6)$$

where w_ℓ and h_ℓ are the width and height of ℓ th convolutional layer. The sparsity of the fully connected layers is the same as in the ER. In a word, ERK allocates larger sparsity for layers that possess more parameters.

- (4) **FCPTS**[12]: This method allows us to learn the sparsity allocation accurately and rapidly by incorporating a differentiable bridge function and a controllable optimization object.

$$L = L_{rec} + L_c, \quad (7)$$

$$L_{rec} = D_{KL}(Y_d || Y_s), \quad (8)$$

$$L_c = \left| \frac{\sum_\ell s_\ell n_\ell}{\sum_\ell n_\ell} - s_0 \right|, \quad (9)$$

where r_ℓ denotes the sparsity rate and n_ℓ is element number of weights of the ℓ th layer. r_0 is the global sparsity rate target. $D_{KL}(\cdot)$ represents the Kullback-Leibler divergence function. Y_d and Y_s are the output of the dense and sparse models.

1.3.2 Reconstruction. After sparsifying the model according to the allocated sparsity rates, reconstruction techniques are usually necessary to restore the accuracy of the sparse model. We utilize a small set of unlabeled samples to reconstruct the sparse model's outputs with those of the original model. This process can be viewed as distilling the sparse model using the dense model as a teacher. In our implementation, we utilize the standard Mean Squared Error (MSE) as the loss function, as shown in 10:

$$L = \sum_{i \in batch} (Y_d^i - f(W_s \odot M_s, X_d^i) - b_s)^2, \quad (10)$$

$$\text{where } Y_d^i = f(W_d, X_d^i).$$

Here, $f(X, W)$ is the convolutional or matrix-multiplication operation conducted by the layer with weight W acting upon input X . X_d is the input of this layer from the dense model. M_s is the binary mask corresponding to the sparsity pattern, which is set to zero if the weight is removed. The sparse binary mask is kept fixed, and gradient descent based on the loss function defined above is used for each layer independently to update the parameters and determine the optimal sparse weights and biases W_s, b_s .

Based on a summary of previous work and practical experimental observations, we have identified three fine-grained algorithms crucial to the reconstruction's effectiveness. In this section, we will provide a detailed introduction to these three algorithms.

Error correction. After establishing the layer-wise sparsity rates, the weights are set to zero based on their specific criterion within each layer. This sparsification operation distorts the weight distribution, introducing biases and scale shifts. Error correction

is widely used in post-training quantization (PTQ) [16, 38, 39] to restore the distortion. However, current PTS methods [26] do not comprehensively and systematically evaluate this procedure. Therefore, we borrow this concept and validate whether this component can be effective for PTS.

To transfer error correction to PTS, we first perform weight correction according to 11:

$$\hat{W}_s = \lambda W_s + E(W_d) - E(\lambda W_s), \quad (11)$$

where $\lambda = \frac{\sigma(W_d)}{\sigma(W_s) + \epsilon}$.

\hat{W}_s is the weights after the weight correction operation, and W_s and W_d denote the weights of sparse and dense models, respectively. E and σ are the mean and standard deviation operators, ϵ is a small constant.

Then we conduct bias correction as 12:

$$\hat{b}_s = b_d + E(f(W_d, X_d)) - E(f(\hat{W}_s, X_d)). \quad (12)$$

where $f(W, X)$ represents the convolutional or matrix-multiplication operation performed by the layer on inputs X with weights W . b_d and X_d are bias and input activation in the dense model. After the weight and bias correction, we can partially correct the error caused by the distribution shift of weights and biases.

The input of reconstruction. During the reconstruction, we can either use the output of the previous reconstruction unit from the dense model as the input (i.e., X_d in 10) or opt for the output after the previous layers, which are sparsified. We find that the choice of reconstruction inputs also greatly impacts the final results. This issue has not been systematically investigated in previous work. Hence, we also explore this factor in our PTSBench.

Reconstruction granularity.

In the main paper, we provide descriptions of various reconstruction granularities. To have a better explanation, we offer a more intuitive visual illustration, as shown in Figure 1.

2 DETAILS OF EXPERIMENTAL SETTINGS

2.1 Details of Datasets

In this section, we give a detailed introduction to the datasets we include.

CIFAR-10/100[24]: The CIFAR-10 and CIFAR-100 are two widely used computer vision datasets created by the Canadian Institute for Advanced Research. The CIFAR-10 dataset consists of 60,000 color images, each sized at 32x32 pixels, divided into 10 classes with 6,000 images per class. In contrast, the CIFAR-100 dataset also comprises 60,000 images but is categorized into 100 classes, with each class containing 600 images. These datasets are commonly utilized for training machine learning and computer vision algorithms, particularly in image recognition and classification tasks. The evaluation metric of CIFAR-10/100 is accuracy, defined as :

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \quad (13)$$

where TP (True Positive) means cases correctly identified as positive, TN (True Negative) means cases correctly identified as negative, FP (False positive) means cases incorrectly identified as positive, and FN (False Negative) means cases incorrectly identified as

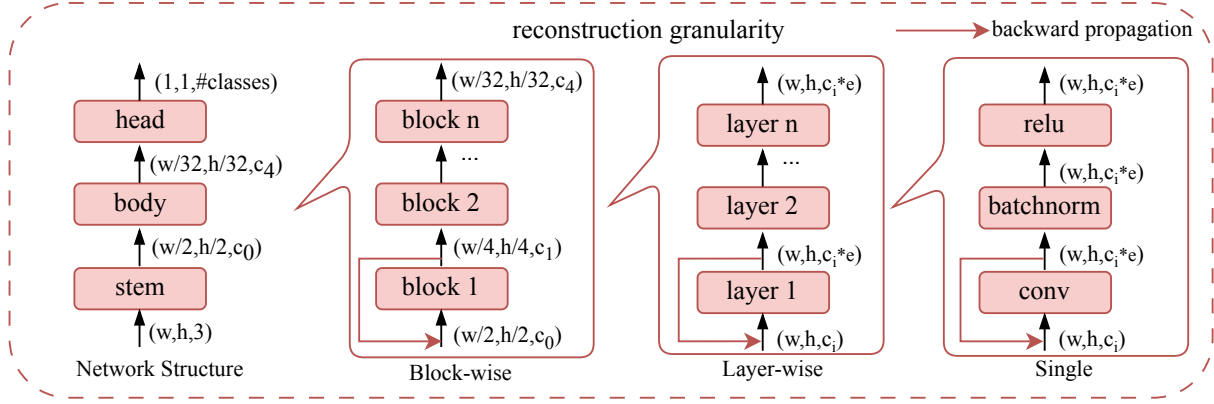


Figure 1: Visualization of different reconstruction granularities. We take a typical CNN structure model [40] as an example. A network consists of a stem layer (usually the first convolutional layer on the input), a body, and a head (usually a fully connected layer). A body is composed of several blocks, and a block contains several layers.

negative. We should calculate the proportion of TP and TN in all evaluated cases to estimate the accuracy.

ImageNet-1k:[1] ImageNet is a substantial dataset extensively used for visual object recognition, consisting of over 14 million images organized into roughly 20,000 categories based on the WordNet hierarchy. Specifically, it includes about 1.2 million images in the training set, 50,000 images in the validation set, and 150,000 images in the test set. These images are annotated and used to train, validate, and test machine learning models. ImageNet is well-known for the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), a competition that has significantly contributed to advancements in deep learning and computer vision by benchmarking algorithms across 1,000 different categories.

PASCAL VOC07[5]: The PASCAL VOC07 (PASCAL Visual Object Classes 2007) dataset is a key computer vision resource consisting of 9,963 images annotated across 20 specific categories. These categories are: Person, Bird, Cat, Cow, Dog, Horse, Sheep, Aeroplane, Bicycle, Boat, Bus, Car, Motorbike, Train, Bottle, Chair, Dining table, Potted plant, Sofa, and TV/Monitor. The dataset is divided into training, validation, and testing sets and is used for a variety of tasks such as object classification, detection, and segmentation, serving as a benchmark for evaluating the performance of advanced object detection algorithms. The PASCAL VOC07 uses mean average precision (mAP) to evaluate results, which is defined as:

$$mAP = \frac{1}{n} \sum_{k=1}^n AP_k, \quad (14)$$

where AP_k denotes the average precision of the k th category, which calculates that area under the precision-recall curve:

$$AP_k = \int_0^1 p_k(r) dr. \quad (15)$$

MSCOCO-2017[31]: The MS COCO (Microsoft Common Objects in Context) dataset is a large-scale object detection, segmentation, key-point detection, and captioning dataset. The dataset consists of 328K images. According to community feedback, in the 2017 release, the training/validation split was changed from

83K/41K to 118K/5K. And the images and annotations are the same. The 2017 test set is a subset of 41K images from the 2015 test set. Additionally, 123K images are included in the unannotated dataset. The COCO17 dataset also uses mean average precision (mAP) as defined above PASCAL VOC07 uses, which is defined as above.

LSUN-Churches/Bedroom: The LSUN (Large-scale Scene Understanding) dataset is a specialized collection aimed at scene understanding and includes specific categories of images such as bedrooms and churches. The LSUN dataset includes specific subsets like LSUN-bedroom and LSUN-churches, aimed at advancing scene understanding in computer vision. The LSUN-bedroom category provides a vast collection of bedroom images, extensively used for training algorithms in image generation and interior scene analysis tasks. The LSUN-churches subset contains a variety of church exterior images utilized primarily for architectural style classification and generative modeling. These subsets help in refining algorithms' capabilities in recognizing and generating images of complex indoor and architectural scenes, forming a crucial part of the larger LSUN initiative designed to improve machine understanding of diverse real-world environments. LSUN uses FID (Fréchet Inception Distance) as the evaluation metric, which is defined as follows:

$$FID = \|\mu_r - \mu_g\|^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2}), \quad (16)$$

where (μ_r, Σ_r) and (μ_g, Σ_g) denote the mean and covariance of the feature vector, which are extracted by a pre-trained Inception V3 model[44], of the real images and generated images, $Tr(\cdot)$ calculates the trace.

2.2 Details of Neural Architecture

This section provides a brief introduction to the benchmarked model architectures, especially the specific structure.

ResNet[15]: ResNet, or Residual Network, is a revolutionary convolutional neural network architecture introduced in 2015 that effectively addresses the vanishing gradient problem in deep networks. It features residual blocks with skip connections, allowing the training of much deeper networks by enabling layers to learn

identity functions, preventing performance degradation. This structure has led to significant improvements in various computer vision tasks, making ResNet a cornerstone in deep learning advancements.

RegNetX[40]: RegNetX is a family of network architectures that emerged from research on designing network architectures systematically to achieve optimal trade-offs between speed, accuracy, and model complexity. Introduced by Facebook AI, RegNetX models use a simple, regularized design that scales depth, width, and resolution in a predictable manner. They are built using blocks of convolutions with a consistent structure, which simplifies the scaling process across different computational budgets and performance needs. This approach allows for efficient and effective scaling of models, making RegNetX suitable for a range of applications, from mobile devices to high-end servers, while maintaining competitive performance in tasks like image classification.

MobileNet[18, 19, 42]: MobileNet is a class of efficient models for mobile and edge devices, introduced by Google. It is designed to provide lightweight, deep neural networks by using depth-wise separable convolutions, significantly reducing the number of parameters and computational cost compared to standard convolutions. MobileNetV2 and MobileNetV3 are both advancements of the original MobileNet model, optimized for mobile and edge devices. MobileNetV2 introduces the inverted residual structure and linear bottlenecks to improve efficiency, while MobileNetV3, developed using techniques like network architecture search, enhances the model further with features like optimized squeeze-and-excitation blocks and the hard-swish activation function.

ShuffleNet[34]: ShuffleNetV2 is an advanced neural network architecture designed to be highly efficient for mobile devices, improving upon its predecessor, ShuffleNet. It features an optimized structure that reduces computational complexity while maintaining high accuracy. Key improvements include the use of channel split and shuffle operations to facilitate better feature mixing and a streamlined architecture that minimizes memory access cost and power consumption. These design choices make ShuffleNetV2 particularly effective for applications in resource-constrained environments where processing power and memory are limited.

VGG[43]: VGG is a classical convolutional neural network architecture. It is proposed by an analysis of how to increase the depth of such networks. It is characterized by its simplicity: the network utilizes small 3×3 filters, and the only other components are pooling layers and a fully connected layer.

ViT[2]: ViT, or Vision Transformer, is a pioneering model that applies the transformer architecture, typically used in natural language processing, to computer vision tasks. Introduced by Google in 2020, ViT segments an image into fixed-size patches, processes these through multiple transformer layers, and uses self-attention mechanisms to capture complex image features at various scales. This approach allows ViT to achieve impressive results on image classification tasks, challenging traditional convolutional neural networks, particularly in scenarios where large-scale training datasets are available. ViT's performance demonstrates the potential of transformers to generalize across different domains beyond text processing.

DeiT[45]: DeiT, or Data-efficient Image Transformers, is a model that adapts the Vision Transformer (ViT) for more data-efficient performance in image classification tasks. Developed by Facebook

AI, DeiT incorporates distillation techniques, where the transformer learns from a pre-trained convolutional neural network acting as a teacher. This approach improves DeiT's training efficiency and effectiveness, making it suitable for scenarios with limited data or computational resources.

RetinaNet[30]: RetinaNet is a popular deep learning framework for object detection that effectively addresses the challenge of detecting objects across a range of scales and object sizes. Introduced by Facebook AI in 2017, RetinaNet is notable for its use of the Focal Loss function, which helps to solve the problem of class imbalance by focusing training on hard-to-classify examples. This model combines a feature pyramid network (FPN) with a ResNet backbone, allowing it to efficiently detect objects at multiple resolutions. The architecture is designed to be both fast and accurate, making it highly effective for real-time object detection applications.

SSD[33]: SSD, or Single Shot MultiBox Detector, is an efficient and powerful algorithm for object detection that performs detection tasks in a single pass through the network, making it faster than methods that require separate proposals and detection stages. Introduced in 2016, SSD divides the image into a grid and uses a series of convolutional layers to predict the presence of objects and their bounding boxes at multiple scales directly from the feature maps. This approach eliminates the need for a separate region proposal network, streamlining the detection process and enhancing speed. SSD is widely used for real-time applications due to its balance of speed and accuracy.

Stable Diffusion[41]: Stable Diffusion is a state-of-the-art text-to-image generation model developed by Stability AI and other collaborators. Released in 2022, it utilizes a latent diffusion model architecture to generate high-quality images based on textual descriptions. The model works by gradually refining an initial random noise image through a series of steps, using a deep learning network to guide the transformation toward an image that matches the textual input. Stable Diffusion is notable for its efficiency and the ability to produce detailed, creative images quickly, even on consumer-grade hardware. It has gained popularity for its open-source availability, allowing widespread use and customization in various applications ranging from art creation to educational tools.

3 MORE EVALUATION RESULTS

We provide more detailed experimental results in this section.

3.1 Results on More Sparsity Rates

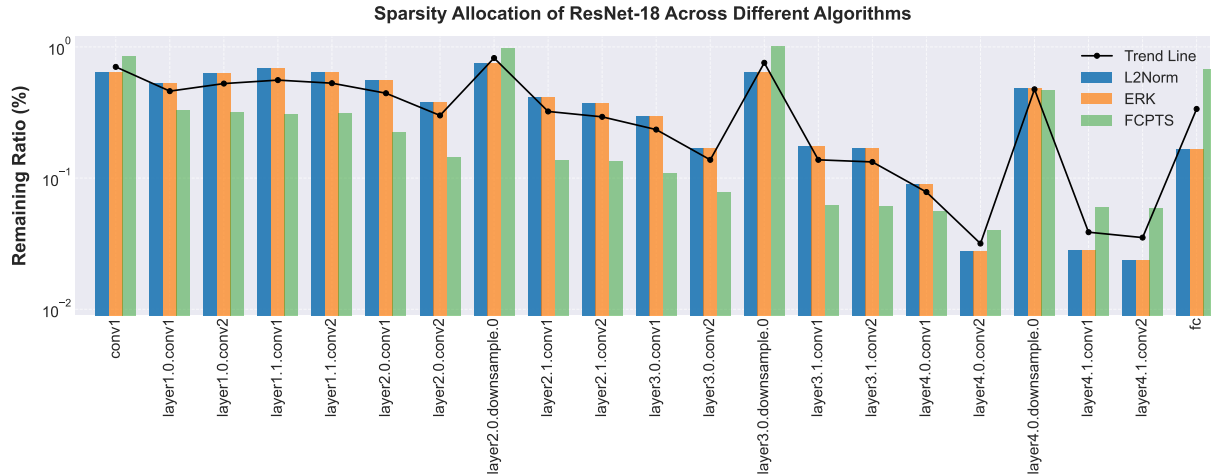
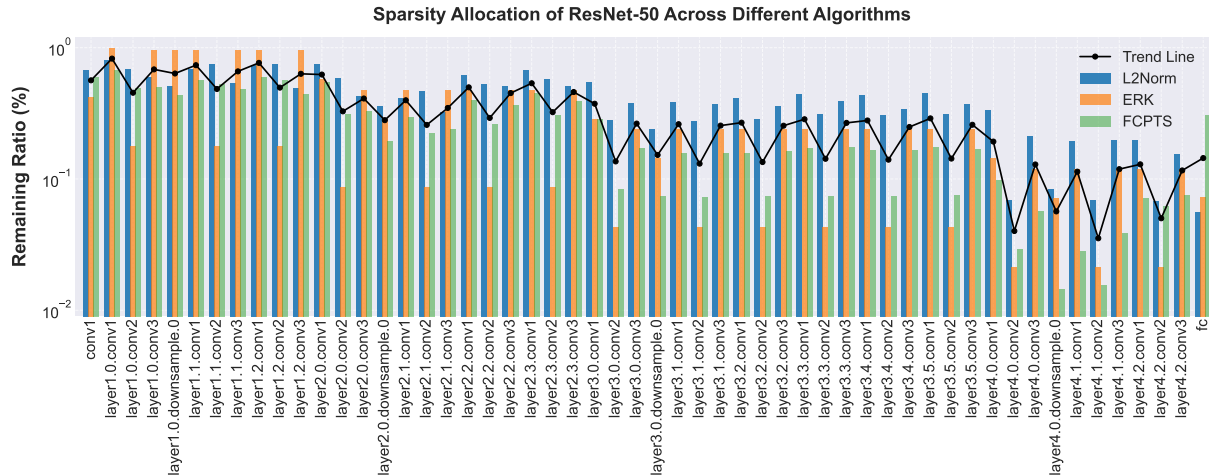
In the main paper, we mainly choose 4 sparsity rates (50%, 60%, 70%, and 80%) for evaluation. We report results under more sparsity rate in Table 1. We can observe almost all models have no performance decrease at a sparsity rate under 50%. In contrast, nearly all models face a collapse of precision at a sparsity rate higher than 80%. Therefore, to make the results more meaningful, we excluded sparsity rates below 40% and above 90% from the benchmark evaluation.

3.2 Results on Sparsity Allocation

In addition to the sparsity allocation of ResNet-32 that we visualize in the main paper, we provide more visualization results in this section, as shown in Figure 2, Figure 3, and Figure 4. From these figures, we can find similar observations as the main paper: 1)

Table 1: The Top@1/Top@5 results under more sparsity rate. Sparsity 0 represents the results of the dense model without sparsification.

Model	Sparsity (%)						
	0	40	50	60	70	80	90
ResNet-18	70.88/90.44	70.69/90.35	70.10/90.11	68.40/89.10	63.72/86.51	44.94/71.50	6.03/15.70
RegNetX-200M	68.41/89.11	66.84/88.22	64.68/87.07	59.98/84.39	48.36/75.85	24.47/49.90	2.02/7.41
MobileNetV2-x0.5	64.95/86.47	63.38/85.36	62.32/84.82	62.21/83.28	54.29/78.92	24.41/48.14	0.18/0.76
ShuffleNetV2-x0.5	61.25/83.34	60.63/83.16	59.75/82.79	58.59/82.22	56.42/80.59	48.00/75.94	24.03/52.61

**Figure 2: Visualization of the sparsity allocation of ResNet-18 on ImageNet. The name of each layer is listed at the bottom. The black line denotes the average remaining ratio of the three algorithms.****Figure 3: Visualization of the sparsity allocation of ResNet-50 on ImageNet. The name of each layer is listed at the bottom. The black line denotes the average remaining ratio of the three algorithms.**

the fully connected layer is preserved well, especially in good-performance sparsity allocation algorithms, and 2) the downsample layers usually have a high remaining ratio, while good-performance

algorithms tend to sparsify them more. Besides, we can draw a new interesting observation: the deeper the layers are, the more weights

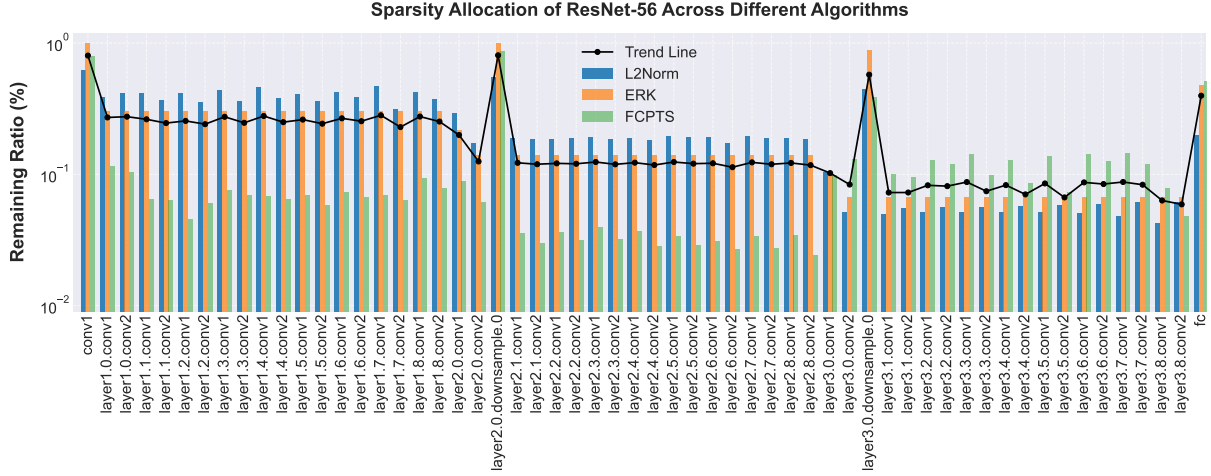


Figure 4: Visualization of the sparsity allocation of ResNet-56 on CIFAR-100. The name of each layer is listed at the bottom. The black line denotes the average remaining ratio of the three algorithms.

Table 2: Comparison of the time consumption of error correction. The best results of each model are marked bolder. The h denotes hours, and the m denotes minutes. 'w/' denotes 'with', and 'w/o' denotes 'without'.

Model	Error Correction	Sparsity Rate			
		50	60	70	80
ResNet-18	w/o	35m56s	35m53s	38m48s	41m53s
	w/	1h26m	1h38m	1h32m	1h32m
ResNet-50	w/o	1h32m	1h35m	1h25m	1h27m
	w/	2h6m	2h7m	2h7m	2h4m
RegNet-200M	w/o	31m44s	38m15s	33m44s	31m59s
	w/	45m19s	45m15s	41m35s	41m31s
RegNet-400M	w/o	46m50s	46m29s	47m19s	47m9s
	w/	1h8m	1h8m	1h8m	1h8m
MobileNetV2-x1.0	w/o	32m1s	35m0s	44m5s	44m4s
	w/	1h4m	1h4m	59m53s	59m51s

it would be sparsified in L2Norm and ERK, while FCPTS does not obey this manner.

3.3 Results on Time Speed

We also evaluate the time consumption. Since it is obvious that sparsity allocation and the input of reconstruction will not introduce extra time consumption, we only provide the time consumption results of error correction and different reconstruction granularities, as shown in Figure 2 and Figure 3.

From Figure 2, we can discover that although error correction may have positive effects on the performance of the sparse model in specific settings, it will bring un-negligible time costs. So, there is a trade-off between precision and time when adopting error correction.

Figure 3 presents the results of time consumption of different reconstruction granularities. The block-wise reconstruction consistently outperforms the time perspective, while reconstructing layer-wisely is better than performing it singly. This is due to the fact that using coarser granularity for reconstruction significantly reduces the need for loss calculations, which are often very time-consuming. Therefore, reconstruction in the block-wise pattern is always the preferred choice, considering both time and precision factors.

4 RAW RESULTS

This section provides the raw results of each track we benchmark in the main paper.

Table 3: Comparison of the time consumption of different reconstruction granularities. The best results of each model are marked bolder. The h denotes hours, and the m denotes minutes.

Model	Granularity	Sparsity Rate (%)			
		50	60	70	80
ResNet-18	Single	3h32m	3h34	3h31	3h28
	Layer-wise	2h2m	1h58m	2h11m	2h5m
	Block-wise	1h37m	1h40m	1h47m	1h41m
ResNet-50	Single	4h44m	4h40m	4h38m	4h31m
	Layer-wise	3h30m	3h21m	3h8m	3h22m
	Block-wise	2h7m	2h12m	2h12m	2h10m
RegNet-200M	Single	5h24m	5h30m	5h16m	5h22m
	Layer-wise	1h10m	1h4m	1h8m	1h15m
	Block-wise	31m55s	38m15s	40m22s	42m19s
RegNet-400M	Single	6h3m	6h2m	6h0m	6h0m
	Layer-wise	1h33m	1h33m	1h48m	1h48m
	Block-wise	1h1m	1h9m	1h19m	1h19m
MobileNetV2-x1.0	Single	5h42m	5h33m	5h29m	5h39m
	Layer-wise	42m43s	35m0s	59m42s	45m22s
	Block-wise	32m1s	27m43s	44m5s	44m4s

Table 4: The raw results of L2Norm sparsity allocation strategy on different models, datasets, and tasks. The sparsity 0 denotes the origin model without sparsification. Each dataset and model architecture is introduced in the previous. The CLS task uses accuracy as a metric, the DET task uses mAP as a metric, and the GEN task uses FID as a metric.

Task	Datasets	Model	Sparsity (%)				
			0	50	60	70	80
CLS	ImageNet	ResNet-18	70.88	69.83	68.25	62.85	35.08
		ResNet-50	77.67	77.30	75.97	68.86	2.17
		RegNetX-200M	68.41	64.68	59.68	48.36	24.47
		RegNetX-400M	70.05	67.53	65.67	56.62	28.55
		MobileNet-x1.0	72.84	70.58	66.19	46.10	0.39
		ViT	77.28	77.1	75.32	70.42	50.41
	CIFAR-10	ResNet-32	93.53	93.12	92.51	90.29	70.16
		ResNet-56	94.37	93.93	93.61	92.84	87.03
		VGG-19	93.91	93.86	93.90	93.8	93.78
	CIFAR-100	ResNet-32	70.16	67.95	64.09	48.50	11.59
		ResNet-56	72.63	72.35	71.02	64.63	39.10
		VGG-19	73.87	73.84	73.85	73.88	73.47
DET	PASCAL VOC07	MobileNetV2-SSDLite	68.70	68.54	67.85	64.45	33.56
		MobileNetV1-SSD	67.50	67.41	66.85	66.29	57.78
	MSCOCO17	RetinaNet-ResNet18	32.70	31.70	30.60	30.50	19.90
		RetinaNet-ResNet50	37.90	37.20	36.50	34.40	25.80
GEN	LSUN-Bedroom	Stable Diffusion	2.97	4.25	-	-	-
	LSUN-Churches	Stable Diffusion	4.55	5.29	-	-	-

Table 5: The raw results of ERK sparsity allocation strategy on different models, datasets, and tasks. The sparsity 0 denotes the origin model without sparsification. Each dataset and model architecture is introduced in the previous. The CLS task uses accuracy as a metric, the DET task uses mAP as a metric, and the GEN task uses FID as a metric.

Task	Datasets	Model	Sparsity (%)				
			0	50	60	70	80
CLS	ImageNet	ResNet-18	70.88	69.28	66.71	59.60	32.26
		ResNet-50	77.67	74.27	69.31	49.99	3.80
		RegNetX-200M	68.41	64.53	60.71	52.96	31.55
		RegNetX-400M	70.05	69.96	67.07	60.39	37.10
		MobileNet-x1.0	72.84	69.14	63.00	41.84	3.07
		ViT	77.28	77.19	75.98	71.09	59.82
	CIFAR-10	ResNet-32	93.53	92.71	92.03	89.61	76.83
		ResNet-56	94.37	93.93	93.61	92.84	87.03
		VGG-19	93.91	93.87	93.83	93.70	93.42
	CIFAR-100	ResNet-32	70.16	66.75	61.74	45.51	15.05
		ResNet-56	72.63	71.73	70.22	62.65	36.96
		VGG-19	73.87	73.89	73.88	73.40	72.23
	PASCAL VOC07	MobileNetV2-SSDLite	68.70	68.55	67.84	64.63	33.60
		MobileNetV1-SSD	67.50	67.36	66.86	66.31	57.79
	MSCOCO17	RetinaNet-ResNet18	32.70	32.10	31.60	30.50	27.30
		RetinaNet-ResNet50	37.90	37.60	37.10	36.00	33.20
GEN	LSUN-Bedroom	Stable Diffusion	2.97	8.49	-	-	-
	LSUN-Churches	Stable Diffusion	4.55	5.60	-	-	-

Table 6: The raw results of Uniform sparsity allocation strategy on different models, datasets, and tasks. The sparsity 0 denotes the origin model without sparsification. Each dataset and model architecture is introduced in the previous. The CLS task uses accuracy as a metric, the DET task uses mAP as a metric, and the GEN task uses FID as a metric.

Task	Datasets	Model	Sparsity (%)				
			0	50	60	70	80
CLS	ImageNet	ResNet-18	70.88	69.54	68.15	63.01	24.71
		ResNet-50	77.67	77.02	72.99	56.22	0.98
		RegNetX-200M	68.41	63.92	54.23	29.38	8.82
		RegNetX-400M	70.05	66.98	61.67	39.25	1.24
		MobileNet-x1.0	72.84	69.70	49.44	5.56	0.54
		ViT	77.28	73.81	18.73	0.17	0.11
	CIFAR-10	ResNet-32	93.53	92.19	90.24	85.23	44.00
		ResNet-56	94.37	93.11	91.22	87.22	71.09
		VGG-19	93.91	93.23	93.55	92.91	92.01
	CIFAR-100	ResNet-32	70.16	65.34	59.01	41.09	1.25
		ResNet-56	72.63	71.93	66.23	59.22	15.09
		VGG-19	73.87	73.29	73.09	73.01	71.02
	PASCAL VOC07	MobileNetV2-SSDLite	68.70	68.31	64.22	59.22	21.09
		MobileNetV1-SSD	67.50	66.09	64.90	62.21	41.29
	MSCOCO17	RetinaNet-ResNet18	32.70	31.50	29.30	19.20	9.10
		RetinaNet-ResNet50	37.90	36.90	33.20	27.50	10.90
GEN	LSUN-Bedroom	Stable Diffusion	2.97	49.03	-	-	-
	LSUN-Churches	Stable Diffusion	4.55	20.34	-	-	-

Table 7: The raw results of FCPTS initiated with ERK sparsity allocation strategy on different models, datasets, and tasks. The sparsity 0 denotes the origin model without sparsification. Each dataset and model architecture is introduced in the previous. The CLS task uses accuracy as a metric, the DET task uses mAP as a metric, and the GEN task uses FID as a metric.

Task	Datasets	Model	Sparsity (%)				
			0	50	60	70	80
CLS	ImageNet	ResNet-18	70.88	69.44	68.86	67.62	64.90
		ResNet-50	77.67	76.53	75.40	73.81	70.41
		RegNetX-200M	68.41	64.98	63.52	60.87	55.41
		RegNetX-400M	70.05	70.15	69.11	67.07	61.89
		MobileNet-x1.0	72.84	68.92	65.89	62.12	49.65
		ViT	77.28	77.10	75.32	70.42	50.41
	CIFAR-10	ResNet-32	93.53	86.23	89.14	81.07	89.48
		ResNet-56	94.37	91.29	92.13	89.18	89.16
		VGG-19	93.91	93.81	93.79	93.82	93.78
	CIFAR-100	ResNet-32	70.16	69.70	78.82	69.02	67.28
		ResNet-56	72.63	71.62	71.13	70.38	68.67
		VGG-19	73.87	73.84	73.86	73.85	73.61
DET	PASCAL VOC07	MobileNetV2-SSDLite	68.70	68.15	68.01	67.20	64.29
		MobileNetV1-SSD	67.50	67.41	67.12	66.64	66.01
	MSCOCO17	RetinaNet-ResNet18	32.70	32.30	32.10	31.50	28.80
		RetinaNet-ResNet50	37.90	37.50	37.20	36.30	33.50
GEN	LSUN-Bedroom	Stable Diffusion	2.97	3.23	-	-	-
	LSUN-Churches	Stable Diffusion	4.55	4.99	-	-	-

Table 8: The raw results of the sparse model without reconstruction on different tasks, datasets, and model architectures. Sparsity 0% denotes the origin model without sparsification.

Task	Datasets	Model	Sparsity (%)				
			0	50	60	70	80
CLS	ImageNet	ResNet-18	70.88	65.92	56.89	34.02	3.76
		ResNet-50	77.67	73.48	65.03	28.66	0.28
		RegNetX-200M	68.41	53.88	30.46	6.06	0.32
		RegNetX-400M	71.84	59.24	33.10	3.83	0.17
		MobileNet-x1.0	72.84	56.98	28.74	1.72	0.11
DET	MSCOCO17	RetinaNet-ResNet18	32.70	30.90	15.70	0	0
		RetinaNet-ResNet50	37.90	32.50	19.70	1.50	0
GEN	LSUN-Bedroom	Stable Diffusion	2.97	49.03	-	-	-
	LSUN-Churches	Stable Diffusion	4.55	29.34	-	-	-

4.1 Sparsity Allocation

Table 4, 5, 6, and 7 shows the results of the sparsity allocation track. From Table 4 and Table 5, we can find that L2Norm and ERK can maintain model performance at a relatively low sparsity rate but face challenges at a high sparsity rate. Table 6 shows that uniform sparsity allocation strategy may have difficulty even in a sparsity at 60%. Table 7 represents the raw results of FCPTS(ERK), which can reach high performance in a sparsity rate at 80%. From these tables, we can intuitively recognize the well-behaved algorithms.

4.2 Reconstruction

Table 8 shows the raw results of the sparse model without performing any reconstruction procedure. Table 9, Table 10, Table 11, Table 12, Table 13, Table 14, and Table 15 represents the detailed raw results of the reconstruction track. Note that FID is a negative indicator, so the smaller the REL, the better the performance. While it is the opposite for CLS and DET tasks.

Table 9: The raw results without error correction. Sparsity 0% denotes the origin model without sparsification. ABS denotes the absolute precision, and REL denotes the relative precision against the one without reconstruction.

Task	Datasets	Model	Sparsity (%)								
			0	50		60		70		80	
				ABS	REL	ABS	REL	ABS	REL	ABD	REL
CLS	ImageNet	ResNet-18	70.88	70.28	+4.35	69.02	+12.12	64.61	+30.58	46.66	+42.90
		ResNet-50	77.67	76.92	+3.43	74.65	+9.61	61.80	+33.13	1.21	+0.93
		RegNetX-200M	68.41	64.94	+11.05	58.73	+28.27	43.80	+37.73	20.26	+19.93
		RegNetX-400M	71.84	69.95	+10.71	66.36	+33.25	57.47	+53.64	28.08	+27.90
		MobileNet-x1.0	72.84	69.90	+12.91	60.42	+31.68	31.16	+29.44	0.87	+0.76
DET	MSCOCO17	RetinaNet-ResNet18	32.70	32.50	+1.60	32.20	+16.50	31.90	+31.90	30.10	+30.10
		RetinaNet-ResNet50	37.90	37.80	+5.30	37.60	+17.90	36.50	+35.00	35.10	+35.10
GEN	LSUN-Bedroom	Stable Diffusion	2.97	4.25	-44.78	-	-	-	-	-	-
	LSUN-Churches	Stable Diffusion	4.55	5.29	-24.05	-	-	-	-	-	-

Table 10: The raw results with error correction. Sparsity 0% denotes the origin model without sparsification. ABS denotes the absolute precision, and REL denotes the relative precision against the one without reconstruction.

Task	Datasets	Model	Sparsity (%)								
			0	50		60		70		80	
				ABS	REL	ABS	REL	ABS	REL	ABD	REL
CLS	ImageNet	ResNet-18	70.88	70.27	+4.34	69.10	+12.20	64.64	+30.61	47.70	+43.94
		ResNet-50	77.67	76.92	+3.43	75.22	+10.18	62.80	+34.13	1.53	+1.23
		RegNetX-200M	68.41	65.00	+11.05	61.28	+30.82	52.42	+46.35	29.98	+29.65
		RegNetX-400M	71.84	69.94	+10.70	67.92	+34.81	61.55	+57.71	41.02	+40.84
		MobileNet-x1.0	72.84	69.90	+12.92	66.23	+37.49	46.45	+44.73	0.63	+0.52
DET	MSCOCO17	RetinaNet-ResNet18	32.70	26.90	0	21.40	+5.70	1.20	+1.20	0	0
		RetinaNet-ResNet50	37.90	10.20	0	0	0	0	0	0	0
GEN	LSUN-Bedroom	Stable Diffusion	2.97	4.33	-44.70	-	-	-	-	-	-
	LSUN-Churches	Stable Diffusion	4.55	5.27	-24.07	-	-	-	-	-	-

Table 11: The raw results using the output of the sparse model as input. Sparsity 0% denotes the origin model without sparsification. ABS denotes the absolute precision, and REL denotes the relative precision against the one without reconstruction.

Task	Datasets	Model	Sparsity (%)								
			0	50		60		70		80	
				ABS	REL	ABS	REL	ABS	REL	ABD	REL
CLS	ImageNet	ResNet-18	70.88	69.83	+3.90	68.52	+11.62	63.28	+29.25	45.98	+42.22
		ResNet-50	77.67	77.03	+3.54	75.64	+10.60	65.09	+36.42	10.92	+10.64
		RegNetX-200M	68.41	64.90	+10.92	61.28	+30.82	52.42	+46.35	29.98	+29.65
		RegNetX-400M	71.84	70.03	+10.79	66.86	+33.76	59.90	+56.07	37.54	+37.36
		MobileNet-x1.0	72.84	70.25	+13.26	66.22	+37.48	46.40	+44.69	0.63	+0.52
DET	MSCOCO17	RetinaNet-ResNet18	32.70	32.40	+1.50	32.10	+16.40	31.80	+31.80	29.90	+29.90
		RetinaNet-ResNet50	37.90	37.80	+5.30	37.60	+17.90	36.50	+35.00	35.10	+35.10
GEN	LSUN-Bedroom	Stable Diffusion	2.97	4.25	-44.78	-	-	-	-	-	-
	LSUN-Churches	Stable Diffusion	4.55	5.29	-24.05	-	-	-	-	-	-

Table 12: The raw results using the output of the sparse model as input. Sparsity 0% denotes the origin model without sparsification. ABS denotes the absolute precision, and REL denotes the relative precision against the one without reconstruction.

Task	Datasets	Model	Sparsity (%)								
			0	50		60		70		80	
				ABS	REL	ABS	REL	ABS	REL	ABD	REL
CLS	ImageNet	ResNet-18	70.88	69.81	+3.88	68.16	+11.26	62.57	+28.54	34.15	+30.39
		ResNet-50	77.67	77.02	+3.54	75.22	+10.18	62.80	+34.13	1.53	+1.25
		RegNetX-200M	68.41	63.92	+10.04	60.11	+29.65	48.97	+42.90	18.63	+18.30
		RegNetX-400M	71.84	69.96	+10.72	66.60	+33.50	58.30	+54.47	32.75	+32.57
		MobileNet-x1.0	72.84	69.28	+12.30	65.59	+36.85	45.05	+43.33	0.42	+0.31
DET	MSCOCO17	RetinaNet-ResNet18	32.70	32.30	+1.40	31.90	+16.20	31.10	+31.10	29.00	+29.00
		RetinaNet-ResNet50	37.90	37.50	+5.00	36.90	+17.20	35.50	+34.00	33.40	+33.40
GEN	LSUN-Bedroom	Stable Diffusion	2.97	4.77	-44.26	-	-	-	-	-	-
	LSUN-Churches	Stable Diffusion	4.55	6.01	-23.33	-	-	-	-	-	-

Table 13: The raw results using single reconstruction. Sparsity 0% denotes the origin model without sparsification. ABS denotes the absolute precision, and REL denotes the relative precision against the one without reconstruction.

Task	Datasets	Model	Sparsity (%)								
			0	50		60		70		80	
				ABS	REL	ABS	REL	ABS	REL	ABD	REL
CLS	ImageNet	ResNet-18	70.88	70.31	+4.38	69.61	+12.71	67.59	+33.56	61.50	+57.74
		ResNet-50	77.67	73.44	-0.04	71.32	+6.28	59.20	+30.53	14.41	+14.14
		RegNetX-200M	68.41	64.01	+10.12	61.91	+31.45	54.95	+48.88	36.38	+36.05
		RegNetX-400M	71.84	69.20	+9.95	67.85	+34.74	61.11	+57.27	35.37	+35.19
		MobileNet-x1.0	72.84	65.90	+8.91	62.22	+33.48	46.04	+44.32	9.18	+9.07
DET	MSCOCO17	RetinaNet-ResNet18	32.70	30.30	-0.60	29.70	+14.00	28.10	+28.10	22.60	+22.60
		RetinaNet-ResNet50	37.90	36.10	+3.60	35.10	+15.40	32.20	+30.70	28.30	+28.30
GEN	LSUN-Bedroom	Stable Diffusion	2.97	5.51	-43.52	-	-	-	-	-	-
	LSUN-Churches	Stable Diffusion	4.55	5.66	-23.68	-	-	-	-	-	-

Table 14: The raw results using layer-wise reconstruction. Sparsity 0% denotes the origin model without sparsification. ABS denotes the absolute precision, and REL denotes the relative precision against the one without reconstruction.

Task	Datasets	Model	Sparsity (%)								
			0	50		60		70		80	
				ABS	REL	ABS	REL	ABS	REL	ABD	REL
CLS	ImageNet	ResNet-18	70.88	70.41	+4.48	69.76	+12.86	67.84	+33.81	63.25	+59.49
		ResNet-50	77.67	74.91	+1.42	73.24	+8.20	68.01	+39.34	52.40	+52.12
		RegNetX-200M	68.41	64.87	+10.98	62.40	+31.94	56.20	+50.13	41.40	+41.07
		RegNetX-400M	71.84	70.22	+10.97	68.57	+35.36	63.73	+59.89	49.33	+49.15
		MobileNet-x1.0	72.84	65.94	+8.94	62.76	+34.02	50.16	+48.44	21.80	+21.69
DET	MSCOCO17	RetinaNet-ResNet18	32.70	30.50	-0.40	30.30	+14.60	29.90	+29.90	26.30	+26.30
		RetinaNet-ResNet50	37.90	36.40	+3.90	35.60	+15.90	34.20	+32.70	31.90	+31.90
GEN	LSUN-Bedroom	Stable Diffusion	2.97	5.43	-43.60	-	-	-	-	-	-
	LSUN-Churches	Stable Diffusion	4.55	5.45	-23.89	-	-	-	-	-	-

Table 15: The raw results using block-wise reconstruction. Sparsity 0% denotes the origin model without sparsification. ABS denotes the absolute precision, and REL denotes the relative precision against the one without reconstruction

Task	Datasets	Model	Sparsity (%)								
			0	50		60		70		80	
				ABS	REL	ABS	REL	ABS	REL	ABD	REL
CLS	ImageNet	ResNet-18	70.88	70.44	+4.51	69.70	+12.80	68.11	+34.08	64.08	+60.32
		ResNet-50	77.67	75.08	+1.59	74.21	+9.17	71.44	+42.77	63.78	+63.50
		RegNetX-200M	68.41	65.02	+11.13	63.91	+33.45	57.89	+51.82	44.49	+44.16
		RegNetX-400M	71.84	70.98	+11.73	69.11	+36.00	65.53	+61.69	55.83	+55.65
		MobileNet-x1.0	72.84	66.01	+9.02	65.14	+36.40	53.39	+51.67	27.21	+27.10
DET	MSCOCO17	RetinaNet-ResNet18	32.70	32.30	+1.40	31.60	+15.90	30.50	+30.50	37.30	+27.30
		RetinaNet-ResNet50	37.90	37.50	+5.00	37.10	+17.40	36.00	+34.50	33.20	+33.20
GEN	LSUN-Bedroom	Stable Diffusion	2.97	4.25	-44.78	-	-	-	-	-	-
	LSUN-Churches	Stable Diffusion	4.55	5.29	-24.05	-	-	-	-	-	-

Table 16: Network architectures used in our PTSBench.

Model	Architecture
ResNet [15]	residual
RegNetX [40]	residual + group conv
MobileNet-V2 [42]	depth-wise conv
MobileNet-V3 [18]	squeeze-and-excitation block
ShuffleNet-V2 [34]	group conv
DeiT [45]	transformer
ViT [2]	transformer

4.3 Neural Architecture and Model Size Robustness

Table 17 shows different models' raw results under different sparsity rates. We also provide the intermediate mean relative accuracy loss for each model architecture. The results of neural architecture and model size robustness tracks are both calculated from it. We list these networks' unique structures in Table 16 to better demonstrate these networks' differences.

4.4 Different Tasks

Table 18 presents the raw results of different tasks track. The results of the GEN task collapse when the sparsity rate is higher than 50%, thus we do not take them into account in many tracks in the main paper.

5 MORE DISCUSSION

5.1 Discussion of Novelty and Significance

We emphasize that our PTSBench includes the following significant contributions: (1) the **first** systematic benchmark that enables a new view to quantitatively evaluate fine-grained PTS algorithms and the sparsification ability of models, (2) uncovering several useful insights and take-away conclusions, and (3) a well-organized open-source evaluation framework and codebase.

- (1) PTSBench is the first effort to facilitate comprehensive and systematic evaluation and comparisons between PTS algorithms and models. PTSBench deconstructs the common pipeline used by existing methods and, based on summarizing a universal paradigm for PTS, explores the fine-grained algorithms of PTS in detail. It also, for the first time, decouples the model's sparsity effects from the sparsity algorithms, linking and exploring sparsity abilities as an inherent aspect of the model itself. In existing works, the sparsity algorithms and model architectures are often asynchronously considered, which may result in misleading experiments and conclusions. Our PTSBench enables a new approach towards a fair comparison of different models by building a unified evaluation track for each model on neural architecture, model size robustness, and different tasks.
- (2) PTSBench reveals several valuable and useful insights and conclusions. Based on the systematic and quantitative evaluation, superior guidance can emerge, which is essential for pushing PTS algorithms to be accurate and efficient. For instance, we recommend using block-wise reconstruction for its superior accuracy and efficiency. We also provide several interesting hypotheses and valuable observations, such as the attention-based models, which are usually more suitable for sparsification for their unique mechanism. These unprecedented quantitative insights identify which techniques are effective and which models tend to be sparsification-friendly, which may provide convenience for further research and practical deployment.
- (3) PTSBench is an upcoming well-coded open-source framework. It will enable every individual to easily evaluate a model's sparsity ability. It also outperforms for its coding. We provide a comparison of PTSBench and POT[26], as shown in Table19. Our PTSBench outperforms in both accuracy and efficiency. In future work, we will also include more state-of-the-art methods. We hope our PTSBench can provide useful advice for future studies.

Table 17: The Top@1/Top@5 results of different models under different sparsity rates. MRAL denotes the mean relative accuracy loss, which is introduced in the main paper.

Model Family	Model Architecture	Dense	Sparsity Rate (%)				MRAL
			50	60	70	80	
ResNet [15]	ResNet-18	70.88/90.44	69.94/89.62	69.09/89.07	66.22/87.56	54.57/79.85	0.334/0.173
	ResNet-34	74.74/92.94	74.31/92.13	73.66/91.83	71.05/90.53	59.51/83.67	0.273/0.146
	ResNet-50	77.67/94.16	77.49/93.98	76.44/93.15	73.13/91.65	48.22/73.43	0.453/0.257
	ResNet-101	79.29/94.93	78.90/94.33	77.36/93.72	69.61/90.32	11.07/25.16	1.011/0.802
	ResNet-152	79.91/95.27	79.54/94.66	77.75/94.05	69.09/90.52	7.30/13.51	1.075/0.927
RegNetX [40]	RegNetX-200M	68.41/89.11	65.34/86.79	62.02/84.88	53.22/79.80	31.45/60.70	0.900/0.496
	RegNetX-400M	72.05/91.27	69.96/89.63	66.60/88.01	58.30/83.24	32.75/63.07	0.841/0.450
	RegNetX-600M	73.43/92.10	72.13/90.90	69.90/89.92	63.19/86.67	42.01/71.91	0.633/0.314
	RegNetX-800M	74.83/92.89	73.49/91.80	71.44/90.79	65.69/87.80	42.11/71.90	0.622/0.315
	RegNetX-1600M	76.78/93.85	75.35/92.86	72.35/91.63	61.49/86.49	24.55/48.66	0.955/0.594
	RegNetX-3200M	78.21/94.60	76.72/93.55	73.77/92.29	60.58/86.26	8.74/20.47	1.189/0.907
	RegNetX-4000M	78.86/94.69	77.93/93.98	75.44/93.11	66.01/89.23	23.12/41.45	0.925/0.644
	RegNetX-6400M	78.94/95.03	78.75/94.44	77.82/94.18	75.33/93.24	66.31/88.93	0.222/0.097
ShuffleNetV2 [34]	ShuffleNetV2-x0.5	61.25/83.34	59.31/81.44	58.06/80.73	55.55/79.35	47.94/74.38	0.393/0.209
	ShuffleNetV2-x1.0	69.75/89.46	67.22/87.56	64.20/86.28	56.44/82.09	32.78/61.59	0.836/0.450
	ShuffleNetV2-x1.5	72.78/91.19	69.68/89.27	65.97/87.42	55.56/81.37	27.58/55.02	0.993/0.566
	ShuffleNetV2-x2.0	74.33/92.06	72.14/90.45	69.60/89.13	62.36/85.06	39.39/66.20	0.724/0.406
MobileNetV2 [42]	MobileNetV2-x0.5	64.95/86.48	63.28/84.76	61.89/84.01	57.89/82.02	39.75/68.04	0.568/0.312
	MobileNetV2-x0.75	70.27/89.98	68.19/88.45	65.53/87.05	55.69/81.28	19.01/39.12	1.033/0.711
	MobileNetV2-x1.0	72.84/91.61	70.63/89.90	66.76/88.21	52.56/79.00	10.34/24.89	1.250/0.921
	MobileNetV2-x1.4	75.57/92.61	73.95/91.84	70.85/90.49	59.36/83.66	11.81/27.22	1.142/0.833
MobileNetV3 [18]	MobileNetV3-x0.35	50.57/74.53	49.71/73.88	48.82/73.23	47.24/71.12	42.81/68.78	0.271/0.135
	MobileNetV3-x0.5	57.68/80.50	56.66/79.78	55.58/79.14	53.25/77.60	44.51/71.49	0.358/0.173
	MobileNetV3-x0.75	63.05/84.41	61.68/83.60	60.19/82.75	55.95/80.28	39.20/67.48	0.557/0.278
	MobileNetV3-x1.0	66.93/87.05	65.26/86.24	63.13/85.04	56.39/81.21	32.41/59.72	0.754/0.413
	MobileNetV3-x1.4	71.31/89.84	69.76/89.19	67.78/88.32	62.01/85.04	38.65/66.65	0.659/0.335
ViT[2]	ViT-B/32	75.86/92.49	75.29/92.15	74.03/91.48	67.43/87.50	45.21/69.85	0.546/0.313
	ViT-L/16	77.28/94.77	77.10/93.91	75.32/93.05	70.42/90.78	50.41/77.56	0.464/0.250
DeiT[45]	DeiT-S	79.90/95.00	77.47/93.97	74.05/92.14	60.34/83.87	22.83/41.53	1.062/0.720
	DeiT-B	81.80/95.60	81.16/95.37	80.48/93.06	78.62/94.23	71.93/90.38	0.183/0.097

5.2 Discussion of the Rationale for Using PTS to Benchmark Model's Sparsity Ability

Using the Post-Training Sparsification (PTS) algorithm to evaluate a model's sparsity ability primarily involves two considerations:

- (1) **Efficiency evaluation:** Using non-PTS methods typically requires training models on large datasets, which can take many hours or even days to complete. This extensive time commitment can be impractical for benchmarking experiments, where numerous models need to be evaluated. In contrast, the PTS method can sparsify a model in just a few hours or even minutes. Therefore, from a time efficiency perspective, PTS is a superior choice.
- (2) **Avoiding the impact of training:** The PTS method does not require extensive training of the model, which better reflects the inherent sparsity capabilities of the model's structure. We also test a non-PTS method by concating a training

step after the PTS pipeline. The results are shown in Table 20. We can observe that even at a high sparsity rate, the models can still easily reach a high performance, which is not beneficial to the evaluation of models.

Therefore, we take PTS methods as the ruler of the model from the two aforementioned aspects.

5.3 Discussion of Error Correction

In the main paper, we find that the effectiveness of the error correction step varies across different tasks. In classification tasks, error correction can enhance the final performance of the model. However, in detection tasks, error correction tends to result in a low model performance. In this section, we delve deeper into investigating the reasons behind these differences. We aim to find whether the failure of error correction under the detection task is related to the sensitivity of the neck and head in detectors.

Table 18: The raw results of different tasks. The sparsity 0 denotes the origin model without sparsification. Each dataset and model architecture is introduced in the previous. The CLS task uses accuracy as a metric, the DET task uses mAP as a metric, and the GEN task uses FID as a metric.

Task	Datasets	Model	Sparsity (%)				
			0	50	60	70	80
CLS	ImageNet	ResNet-18	70.88	69.83	68.25	62.85	35.08
		ResNet-50	77.67	77.30	75.97	68.86	2.17
		RegNetX-200M	68.41	64.68	59.68	48.36	24.47
		RegNetX-400M	70.05	67.53	65.67	56.62	28.55
		MobileNet-x1.0	72.84	70.58	66.19	46.10	0.39
		ViT	77.28	77.1	75.32	70.42	50.41
	CIFAR-10	ResNet-32	93.53	93.12	92.51	90.29	70.16
		ResNet-56	94.37	93.93	93.61	92.84	87.03
		VGG-19	93.91	93.86	93.90	93.8	93.78
	CIFAR-100	ResNet-32	70.16	67.95	64.09	48.50	11.59
		ResNet-56	72.63	72.35	71.02	64.63	39.10
		VGG-19	73.87	73.84	73.85	73.88	73.47
	PASCAL VOC07	MobileNetV2-SSDLite	68.70	68.54	67.85	64.45	33.56
		MobileNetV1-SSD	67.50	67.41	66.85	66.29	57.78
	MSCOCO17	RetinaNet-ResNet18	32.70	31.70	30.60	30.50	19.90
		RetinaNet-ResNet50	37.90	37.20	36.50	34.40	25.80
GEN	LSUN-Bedroom	Stable Diffusion	2.97	4.25	57.88	1129.12	29803.50
	LSUN-Churches	Stable Diffusion	4.55	5.29	79.28	3922.09	87293.50

Table 19: Comparison of the accuracy and efficiency between POT and PTSBench. OpenVINO is the framework POT adopt.

Model	Framework	Accuracy	Time
ResNet-18	OpenVINO	70.10	4h15min
	PTSBench	70.27	1h26min
ResNet-50	OpenVINO	76.82	10h16min
	PTSBench	76.92	2h6min
RegNetX-200M	OpenVINO	64.68	3h1min
	PTSBench	64.94	43min19s
RegNetX-400M	OpenVino	67.53	6h21min
	PTSBench	69.29	1h8min
MobileNetV2-x1.0	OpenVino	65.51	5h40min
	PTSBench	69.90	1h4min

Table 20: The Top@1 accuracy results of a non-PTS method.

Model	Dense	Sparsity Rate (%)			
		50	60	70	80
ResNet-18	70.88	70.95	70.67	70.42	68.62
ResNet-50	77.67	77.49	77.46	77.29	76.26
RegNetX-200M	68.41	67.29	66.30	64.35	60.40
RegNetX-400M	72.05	71.01	70.93	69.54	66.98
MobileNetV2	72.84	71.26	70.16	67.53	63.43

Table 21: Comparing the results of applying error correction to different components of the detection model.

Sparsity Rate (%)	Method		
	no EC	all EC	backbone EC
50	32.1	30.2	31.1
60	31.6	21.4	22.8
70	30.5	1.2	2.7
80	27.3	0	0
90	15.5	0	0

We apply error correction only to the backbone part of the detection model while preserving the post-sparsity weight distribution in the neck and head parts. We aim to find whether the failure of error correction under the detection task is related to the sensitivity of the neck and head in detectors. The results are shown in Table 21. We can find that including the neck and head in the error correction process does not significantly impact the results (e.g., with 1% mAP increase at 50%, 60%, and 70% sparsity rates). This suggests that the failure of error correction under the detection task is not because of the neck and head. We will investigate the reason behind it in our future work.

As we also use other techniques like parameter reconstruction in previous experiments, it is still unclear whether the error correction fails because of itself instead of the combination of other techniques. To further investigate this, after performing error correction, we

Table 22: The results of directly measuring the model's accuracy without undergoing reconstruction.

Sparsity Rate (%)	Method	
	w/o EC	w/ EC
50	32.5	0.1
60	26.6	0
70	19.2	0
80	2.4	0
90	0	0

bypassed the reconstruction process and directly evaluated the accuracy of the sparse model. The results are presented in Table 22. We can see that there is a huge gap between the methods using EC and not using EC (e.g., at 50% sparsity rate, the method with EC occurs a collapse on mAP while the result of the method without EC is 32.5), which demonstrates the negative effect caused by the EC itself.

REFERENCES

- [1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.
- [2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiuhua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).
- [3] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. 2019. Learned step size quantization. *arXiv preprint arXiv:1902.08153* (2019).
- [4] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. 2020. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*. PMLR, 2943–2952.
- [5] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2010. The pascal visual object classes (voc) challenge. *International journal of computer vision* 88 (2010), 303–338.
- [6] Gongfan Fang, Xinyin Ma, and Xinchao Wang. 2023. Structural Pruning for Diffusion Models. *arXiv preprint arXiv:2305.10924* (2023).
- [7] Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635* (2018).
- [8] Elias Frantar and Dan Alistarh. 2022. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems* 35 (2022), 4475–4488.
- [9] Elias Frantar and Dan Alistarh. 2022. SPDY: Accurate pruning with speedup guarantees. In *International Conference on Machine Learning*. PMLR, 6726–6743.
- [10] Trevor Gale, Erich Elsen, and Sara Hooker. 2019. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574* (2019).
- [11] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. 2022. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*. Chapman and Hall/CRC, 291–326.
- [12] Ruihao Gong, Yang Yong, Zining Wang, Jinyang Guo, Xiuying Wei, Yuqing Ma, and Xianglong Liu. 2024. Fast and Controllable Post-training Sparsity: Learning Optimal Sparsity Allocation with Global Constraint in Minutes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 12190–12198.
- [13] Jinyang Guo, Jiaheng Liu, Zining Wang, Yuqing Ma, Ruihao Gong, Ke Xu, and Xianglong Liu. 2023. Adaptive Contrastive Knowledge Distillation for BERT Compression. In *Findings of the Association for Computational Linguistics: ACL 2023*. 8941–8953.
- [14] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems* 28 (2015).
- [15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
- [16] Yefei He, Luping Liu, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. 2023. PTQD: Accurate Post-Training Quantization for Diffusion Models. *arXiv preprint arXiv:2305.10657* (2023).
- [17] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).
- [18] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. 2019. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*. 1314–1324.
- [19] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).
- [20] Itay Hubara, Brian Chmiel, Moshe Island, Ron Banner, Joseph Naor, and Daniel Soudry. 2021. Accelerated sparse neural training: A provable and efficient method to find n: m transposable masks. *Advances in neural information processing systems* 34 (2021), 21099–21111.
- [21] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2017. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research* 18, 1 (2017), 6869–6898.
- [22] Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. 2021. Accurate post training quantization with small calibration sets. In *International Conference on Machine Learning*. PMLR, 4466–4475.
- [23] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and2 training of neural networks for efficient integer-arithmetic-only inference. In *IEEE Conf. Comput. Vis. Pattern Recog.* 2704–2713.
- [24] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).
- [25] Woosuk Kwon, Sehoon Kim, Michael W Mahoney, Joseph Hassoun, Kurt Keutzer, and Amir Gholami. 2022. A fast post-training pruning framework for transformers. *Advances in Neural Information Processing Systems* 35 (2022), 24101–24116.
- [26] Ivan Lazarevich, Alexander Kozlov, and Nikita Malinin. 2021. Post-training deep neural network pruning via layer-wise calibration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 798–805.
- [27] Jaeho Lee, Sejun Park, Sangwoo Mo, Sungsoo Ahn, and Jinwoo Shin. 2020. Layer-adaptive sparsity for the magnitude-based pruning. *arXiv preprint arXiv:2010.07611* (2020).
- [28] Yawei Li, Kamil Adamczewski, Wen Li, Shuhang Gu, Radu Timofte, and Luc Van Gool. 2022. Revisiting random channel pruning for neural network compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 191–201.
- [29] Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. 2021. Brecq: Pushing the limit of post-training quantization by block reconstruction. *arXiv preprint arXiv:2102.05426* (2021).
- [30] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*. 2980–2988.
- [31] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part V 13*. Springer, 740–755.
- [32] Shiwei Liu, Tianlong Chen, Xiaohan Chen, Li Shen, Decebal Constantin Mocanu, Zhangyang Wang, and Mykola Pechenizkiy. 2022. The unreasonable effectiveness of random pruning: Return of the most naive baseline for sparse training. *arXiv preprint arXiv:2202.02643* (2022).
- [33] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. Ssd: Single shot multibox detector. In *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 21–37.
- [34] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. 2018. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*. 116–131.
- [35] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. 2018. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications* 9, 1 (2018), 2383.
- [36] Hesham Mostafa and Xin Wang. 2019. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *International Conference on Machine Learning*. PMLR, 4646–4655.
- [37] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. 2020. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*. PMLR, 7197–7206.
- [38] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. 2019. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1325–1334.
- [39] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart Van Baalen, and Tijmen Blankevoort. 2021. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295* (2021).

- [40] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. 2020. Designing network design spaces. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10428–10436.
- [41] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10684–10695.
- [42] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4510–4520.
- [43] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
- [44] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2818–2826.
- [45] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2021. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*. PMLR, 10347–10357.
- [46] Xiuying Wei, Ruihao Gong, Yuhang Li, Xianglong Liu, and Fengwei Yu. 2022. Qdrop: Randomly dropping quantization for extremely low-bit post-training quantization. *arXiv preprint arXiv:2203.05740* (2022).
- [47] Kaixin Xu, Zhe Wang, Xue Geng, Min Wu, Xiaoli Li, and Weisi Lin. 2023. Efficient Joint Optimization of Layer-Adaptive Weight Pruning in Deep Neural Networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 17447–17457.
- [48] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. 2017. On compressing deep models by low rank and sparse decomposition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7370–7379.
- [49] Michael Zhu and Suyog Gupta. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878* (2017).