Appendix

A Proofs of Theorems

A.1 Proof of Theorem 1

Theorem 3 (The best low-rank approximation)**.** *Suppose W is decomposed via SVD and yield $W = \sum_{i=1}^{r} \sigma_i u_i v_i^T$ where singular values $\{\sigma_i\}$ are sorted in descending order. Given integer $k < r$, the best k-rank approximation of W, namely the k-rank matrix that has the smallest l_2 distance to W is*

$$\widetilde{W} = \sum_{i=1}^{k} \sigma_i u_i v_i^T.$$

This theorem is also the Eckart-Young-Mirsky Theorem for Frobenius norm.

Though the proof is easily accessible (*e.g.* from Wikipedia [2]), we will provide a sketch of proof here for reference.

Proof. Denote $W_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$ be the best k-rank approximation of W, $\sigma_i(W)$ be the i^{th} largest singular value of W. Then the low-rank approximation error could be reduced as follows:

$$\left\| W - \widetilde{W} \right\|_F^2 = \left\| \sum_{i=k+1}^{r} \sigma_i u_i v_i^T \right\|_F^2 = \sum_{i=k+1}^{r} \sigma_i^2. \tag{A.1}$$

Given $W = W' + W''$, according to the triangle inequality of spectral norm,

$$\sigma_1(W) = \sigma_1(W') + \sigma_1(W''). \tag{A.2}$$

Then for two arbitrary ranks $i, j \geq 1$, we have:

$$\begin{aligned} \sigma_i\left(W'\right) + \sigma_j\left(W''\right) &= \sigma_1\left(W' - W'_{i-1}\right) + \sigma_1\left(W'' - W''_{j-1}\right) \\ &\geq \sigma_1\left(W - W'_{i-1} - W''_{j-1}\right) \\ &\geq \sigma_1\left(W - W_{i+j-2}\right) \\ &= \sigma_{i+j-1}(W). \end{aligned} \tag{A.3}$$

Assume there is another k-rank approximation X, Then according to the above formula, for arbitrary $i \geq 1$,

$$\sigma_i(W - X) = \sigma_i(W - X) + \sigma_{k+1}(X) \geq \sigma_{k+i}(W) \tag{A.4}$$

Hence,

$$\|W - X\|_F^2 \geq \sum_{i=1}^{n} \sigma_i\left(W - X\right)^2 \geq \sum_{i=k+1}^{n} \sigma_i^2, \tag{A.5}$$

which means $\widetilde{W}$ is the best k-rank approximation. □

A.2 Proof of Theorem 2

Theorem 2 states the effectiveness of rank loss. Before the theorem, recall notations that we previously defined: $\overline{W} := \frac{W}{\|W\|}$ is the l_2 normalized weight matrix W; U, Σ, V are matrices reached from the SVD of $\overline{W}$, where $U = \{u_1, u_2, ...\}$ and $V = \{v_1, v_2, ...\}$ are orthonormal bases; Σ is a diagonal matrix where singular values $\{\sigma_1, \sigma_2, ...\}$ are sorted in descending order on the diagonal; operator $\mathrm{Trun}\left(U\Sigma V^T\right) = \sum_{i=1}^{k} \sigma_i u_i v_i^T$ stands for k-rank truncated SVD, or the k-rank best approximation of $\overline{W}$ according to Theorem 1.

[2] https://en.wikipedia.org/wiki/Low-rank_approximation

Theorem 4 (Effectiveness of the adversarial rank loss)**.** *Given the adversarial rank loss*

$$\mathcal{L}_{rank} = -\|\overline{W} - \text{Trun}\left(U\Sigma V^T\right)\|_F^2. \tag{A.6}$$

If we optimize W in rank loss via gradient descent, the rank of W will increase.

Proof. In gradient descent, the update from weight W to W' based on rank loss $\mathcal{L}_{rank}$ could be described as:

$$W' = W - \gamma \frac{\partial \mathcal{L}_{rank}}{\partial W}. \tag{A.7}$$

We first simplify rank loss. Since $U = \{u_1, u_2, ...\}$ and $V = \{v_1, v_2, ...\}$ are orthonormal bases, we could easily rewrite rank loss $\mathcal{L}_{rank}$ as the squared sum of small singular values:

$$\begin{aligned}
\mathcal{L}_{rank} &= -\|\frac{W}{\|W\|_F} - \sum_{i=1}^{k} \sigma_i u_i v_i^T\|_F^2 \\
&= -\|U\Sigma V^T - U\Sigma_{[1:k]}V^T\|_F^2 \\
&= -\|\Sigma - \Sigma_{[1:k]}\|_F^2 = -\sum_{i=k+1}^{r} \sigma_i^2.
\end{aligned} \tag{A.8}$$

The form Equation (A.8) allows us to apply chain rule to calculate the gradient of the normalized weight matrix $\frac{\partial \mathcal{L}_{rank}}{\partial \overline{W}}$:

$$\frac{\partial \mathcal{L}_{rank}}{\partial \overline{W}} = \sum_{i=k+1}^{r} \frac{\partial \mathcal{L}_{rank}}{\partial \sigma_i} \frac{\partial \sigma_i}{\partial \overline{W}} = -\sum_{i=k+1}^{r} 2\sigma_i u_i v_i^T. \tag{A.9}$$

Again, the chain rule is applied for the derivative of the weight matrix. For clarity, we show the gradient expression for a single weight parameter $W[m, n]$ (the weight value at position (m, n) in the reshaped weight matrix W):

$$\begin{aligned}
\frac{\partial \mathcal{L}_{rank}}{\partial W[m,n]} =& tr\left(\frac{\partial \mathcal{L}_{rank}}{\partial \overline{W}^T} \frac{\partial \overline{W}}{\partial W[m,n]}\right) \\
=& -\frac{(\sum_{i=k+1}^{r} 2\sigma_i u_i v_i^T)[m,n]}{\|W\|_F} \\
&+ \frac{W[m,n] \sum \left(W \odot \left(\sum_{i=k+1}^{r} 2\sigma_i u_i v_i^T\right)\right)}{\|W\|_F^3}.
\end{aligned} \tag{A.10}$$

Based on the gradient, one step of optimization under learning rate α could be expressed in a neat matrix multiplication format, decomposed by orthonormal bases $U = \{u_1, u_2, ...\}$ and $V = \{v_1, v_2, ...\}$.

Lemma 1 (Weight optimization based on rank loss)**.** *Denote* $\sum \left(W \odot \left(\sum_{i=k+1}^{r} 2\sigma_i u_i v_i^T\right)\right) := c$*, which is a scalar constant within each step, one step of weight W optimization under learning rate*

$\gamma > 0$ *and rank loss (as defined in Equation* (2.6)*) could be expressed as:*

$$\begin{aligned} W' =& W - \gamma \frac{\partial \mathcal{L}_{rank}}{\partial W} \\ =& W - \gamma \left(-\frac{\sum_{i=k+1}^{r} 2\sigma_i u_i v_i^T}{\|W\|_F} + \frac{cW}{\|W\|_F^3} \right) \\ =& U\Sigma V^T + \frac{2\gamma}{\|W\|_F} U\Sigma_{[k+1:r]} V^T - \frac{c\gamma}{\|W\|_F^3} U\Sigma V^T \\ =& U \left(\left(1 - \frac{c\gamma}{\|W\|_F^3} \right) \Sigma + \frac{2\gamma}{\|W\|_F} \Sigma_{[k+1:r]} \right) V^T. \end{aligned} \tag{A.11}$$

From the formula of the optimized weight W', we can reach the following conclusions on the optimized weight W': firstly, W' promises the same set of orthonormal bases U, V after SVD; secondly, comparing small singular values against others, all singular values are penalized by the same amount $\frac{c\gamma}{\|W\|_F^3}$; but the small singular values (ranking from $k+1$ to r) are awarded with increments $\frac{2\gamma}{\|W\|_F}$. Regardless of swapped singular values due to magnitude change (because of small learning rate γ), small singular values will make up more proportion in all the singular values after one step of update. Recall the definition for δ-rank, given fixed sum of squared singular values after l_2 normalization, the rank of W will increase. □

B Method Details

B.1 Details on Gradient-Grow.

We explain in details about the procedure of gradient grow evolved from RigL [14]. At each gradient grow step, first we calculate the Rank-based Pruning (RPG) objective $\mathcal{L}$. Then we back-propagate $\mathcal{L}$ for gradients of network weights. Finally, gradients of pruned network weights are sorted in magnitudes; weights with large gradients are re-activated.

The number of weights to be re-activated are determined by the number of remaining weights. The whole pruning framework is detailed in Algorithm 1. Grow fraction α is a function of training iterations that gradually decays for stability of training. Cosine Annealing is used for α, following [14].

B.2 Selection of Approximation Rank

Factor k of the truncation operator Trun controls the rank of the low-rank approximation in this adversarial process. However, controlling k for each layer is hardly practical, because layers are large in quantity and vary in shapes. We leverage the concept of δ-rank in Definition 1, and tend to control the approximation error (also the negative rank loss $-\mathcal{L}_{rank}$ for a layer) rather than control k. We set a constant $\tilde{\delta}$ between $(0, 1)$ as the target approximation error between the normalized weight matrix $\overline{W}$ and its low rank counterpart. Then we find the best k that has the closest low-rank approximation error as $\tilde{\delta}$ for each layer. Mathematically, this selection process could be written as:

$$\arg\min_{k} | - \mathcal{L}_{rank} - \tilde{\delta}|. \tag{B.1}$$

C Experiment Details

C.1 CIFAR Experiments

In the CIFAR experiments, we train VGG-19 and ResNet-32 for 300 epoch, which is identical to Zhou et al. [59]. We use the SGD optimizer with momentum 0.9, batchsize 128, learning rate 0.1, and weight decay 0.005.

Algorithm 1: Rank-based Pruning (RPG)

Input : A dense model W with n layers; Target density (a function of iteration) d; Grow fraction (a function of iteration) α; Mask update interval ΔT; Total training iteration $T = T_{prune} + T_{finetune}$

Output : A Sparse Model $W \odot M$

// Initialize a dense mask ;
$M \leftarrow \mathbb{1}$;
// Stage 1: prune and grow;
for $t \leftarrow 1$ ***to*** T_{prune} **do**
  Forward propagation for minibatch loss $\mathcal{L}_{task}$;
  if $t\%\Delta T = 0$ **then**
    // Update mask M;
    Calculate and sum layerwise rank loss;
    Keep top d_t proportion of weights in W globally to get layerwise density d_t^i;
    for $i \leftarrow 1$ ***to*** n **do**
      Prune to density $(1 - \alpha_t)d_t^i$ based on $|W|$;
      Grow to density d_t^i again based on $|\nabla \mathcal{L}|$;
    end for
  end if
  else
    Train sparse net with task loss;
  end if
end for
// Stage 2: keep masks fixed and finetune;
for $t \leftarrow T_{prune} + 1$ ***to*** T **do**
  Keep mask M static, and train weight W till converge;
end for
return *Weights of the Pruned Model* $W \odot M$

C.2 ResNet-50 on ImageNet

We mainly follow Goyal et al. [18] for ImageNet experiments. ImageNet experiments are run on 8 NVIDIA Tesla V100s. Our RPG method is applied on ResNet-50 for 100 epoch, and compared with 100-epoch baselines at various sparsities. We use the SGD optimizer with momentum 0.9, 1024 batchsize, a learning rate of 0.4, 0.0001 weight decay, 0.1 label smoothing, 5 epochs of learning rate warmup. We choose 1024 batchsize instead of the commonly-used 4096 batchsize [14, 45] due to GPU memory constraints. We did not use augmentation tricks like mixup or cutmix. Standard pretrained model from torchvision is used in ImageNet setting for fair comparison with top baselines. α is kept at 0.3; ΔT is set to 100; T_{prune} is set to 90 epochs.

C.3 Downstream Vision Tasks

In Mask R-CNN experiments, all methods are implemented by us on the identical settings for fair comparison. We follow the training setting of the original Mask R-CNN [21] on Detectron-2 [54]. The three methods are applied for 96000 iterations. Notably, certain components in Mask R-CNN ResNet-50 FPN are loaded from pretrained models and left untrained. These components are omitted from pruning and sparsity calculations for RPG and baseline experiments.

C.4 Vision Transformers

In vision transformers, we mainly follow the official setting of DeiT [52], but we extend training epochs from 300 to 600 for fair comparison with SViT-E [6]. All other training settings are identical to the official training setting of DeiT [52].

C.5 Replication of AC/DC

AC/DC [40] achieves State-of-the-Art performance at high sparsities. Therefore, we hope to compare our method extensively with AC/DC on various settings. Accordingly, the schedule of AC/DC need slight modifications based on the original setting. According to the original work [40], dense and sparse status are alternated every 5 epochs. We scale the period according to the ratio of the training epochs versus the standard training epochs in the paper [40]. For example, if the AC/DC method is applied for 300 epochs, since the original ImageNet AC/DC setting has only 100 epochs, we lengthen the period of AC/DC from the original 5 epochs to 15 epochs. For warming-up and finetuning periods, the similar scaling is performed.

D Additional Experiments

We also tried RPG pruning without loading pretrained models on ResNet-50 ImageNet experiments. Training settings are kept the same with other ResNet-50 pruning experiments. The results are provided in Table 8. The proposed RPG method could surpass all baselines even without loading pretrained models.

Algorithm	Sparsity	FLOPs	Accuracy
ResNet-50 [22]	0	1×	76.80
RPG from scratch	0.9	0.155×	75.35
RPG from scratch	0.95	0.093×	73.62

Table 8. RPG-Sparsified ResNet-50 from scratch on ImageNet.