SVD-NAS: Coupling Low-Rank Approximation and Neural Architecture Search (Supplementary Material)

S1. Implementation Details

S1.1. Identify the Optimal Design Point

With regards to which layers to compress, for ResNet-18, we target every 3×3 convolutional layer except the first layer. For MobileNetV2 and EfficientNet-B0, we compress all the point-wise convolutions.

To reduce the searching cost, the design space is pruned by FLOPs for all three models, where the step size of the grid search is 5%. When pruning by accuracy, 500 images are randomly selected from the validation set for the proxy task, and these images are fixed throughout the experiment. This technique is only used on MobileNetV2 and EfficientNet-B0, and the accuracy degradation tolerance of the proxy task τ_{proxy} is set as 5pp of top-1 accuracy.

During NAS, the sampling parameters $\boldsymbol{\theta_i}$ are learned using Adam with lr=0.01, $betas$=(0.9,0.999), $weight\ decay$=0.0005. The learning process takes 100 epochs and 50 epochs respectively for searching the first and the second branch. The Gumbel-Softmax temperature is initialised as 5, and decays by 0.965 for every epoch.

S1.2. Iterative Searching Method

The proposal of the iterative searching method aims to reduce the searching cost by focusing on one branch of the building block at each time. Since this method identifies the optimal configuration of each branch in a greedy way, it is at the risk of not finding the global optimal solution. Empirically, we found that relaxing the configuration identified for the first branch can help us to reach better design points.

As Algorithm S.1 shows, for the first branch, we choose to relax its configuration from r_i^0 to $r_i^{0,*}$ by removing a proportion of ranks Δr_i. The intuition behind is that the relaxation will give the second iteration of NAS more space to explore, probably reducing the chance of being stuck at local optimum. In our experiment, we choose the value of Δr_i so that the corresponding FLOPs difference between using r_i^0 and $r_i^{0,*}$ is equal to 20% FLOPs of the *original layer*.

Algorithm S.1 Iterative Searching (Relaxed Configuration)

```
1: $\boldsymbol{E_{i,0}} = \boldsymbol{W_i}$
2: for $b \in \{0, 1\}$ do
3:     identify optimal $F_b(\hat{\boldsymbol{W}}_{\boldsymbol{i}}^{\boldsymbol{1,b}}, \hat{\boldsymbol{W}}_{\boldsymbol{i}}^{\boldsymbol{0,b}})$ to approximate $\boldsymbol{E_{i,b}}$, with the rank of $r_i^b$
4:     if b==0 then
5:         relax the rank $r_i^{b,*} = r_i^b - \Delta r_i$, and update the low-rank weight tensors accordingly
6:     end if
7:     $\boldsymbol{E_{i,b+1}} = \boldsymbol{E_{i,b}} - F_b(\hat{\boldsymbol{W}}_{\boldsymbol{i}}^{\boldsymbol{1,b}}, \hat{\boldsymbol{W}}_{\boldsymbol{i}}^{\boldsymbol{0,b}})$
8: end for
```

S1.3. Generate the Synthetic Dataset

Each random initialised image is optimised with Adam for 500 iterations. The learning rate is initialised at 0.5, 0.25 and 0.5 for ResNet-18, MobileNetV2 and EfficientNet-B0 respectively, and it is scheduled to decay by 0.1 as long as the loss stopped improving for 100 iterations. α in l_{bn} is set to 1 for ResNet-18 and MobileNetV2, and 100 for EfficientNet-B0. The scaling factor $\frac{1}{f_i}$ is introduced on MobileNetV2 and EfficientNet-B0 only, but not ResNet-18. The batch size is set to 32 for all three models.

Compared with ZeroQ, Our objective function l_{bn} differs in that we averaged the loss for each batch normalisation layer by scaling it with the number of channels f_i. We found taking this average is important for those compact models which use MBBlock, such as MobileNetV2 and EfficientNet to produce high-quality synthetic images. The intuition behind this change is that the number of channels in these compact models varies a lot between layers because of the inverted residual structure. Therefore, averaging by f_i helps balancing the contribution of each batch normalisation layer towards the total loss.

In terms of the storage of the synthetic dataset, the floating point format leads to better knowledge distillation results the quantised RGB format. As such, a synthetic dataset containing 25k images requires about 14 GB disk space.

S1.4. Fine-tune the Low-rank Model

For all the experiment set-ups: post-training, few-sample training and full training, our framework uses SGD for fine-tuning, with momentum and weight decay set to 0.9 and 0.0001 respectively. The learning rate is initialised to 0.001 and set to decay by 0.1, as long as validation accuracy has not been improved for the last 10 epochs. The fine-tuning stops once the learning rate is below 0.0001.

S1.5. Measure Performance

S1.5.1 FLOPs and Parameters Measurement

The number of FLOPs and parameters in a given model is calculated using the open-source library thop[1], and the number of FLOPs is defined as the twice of Multiply-Accumulate (MAC).

[1]https://github.com/Lyken17/pytorch-OpCounter

S1.5.2 Latency Measurement

The reported CPU latency is measured on a Pixel 4 phone using the Tensorflow Lite 2.1 native benchmark binary [2]. The CNN model runs on a single thread using one of the Cortex-A76 core, and the reported result is the average of 50 runs.

S2. Reproduce Previous Work

When comparing our SVD-NAS with existing works, for few-sample training and full training set-ups, we presented the data reported in those papers. However, for the post-training set-up, those relevant works did not disclose the precise data. Therfore, we reproduce the following works to generate the numbers in the table of post-training comparison. The code for reproducing these works has also been released.

S2.1. ALDS

The authors of ALDS demonstrated the post-training results, referred to as "Compress-only" in their paper, in figures without providing the actual performance numbers. Therefore, we use their official code [3] to rerun the post-training tests. For fair comparison, we altered their way of FLOPs and parameters measurement to align with ours, as their official code only counts the FLOPs in the convolutional and fully-connected layers rather than the entire model, which would overestimate the compression ratio.

S2.2. LR-S2

The authors of LR-S2 proposed a single decomposition scheme and a rank selection method, which is:

$$\min_{r_k}(\sum_{k=1}^{K}(\lambda C_k(r_k) + \frac{\mu}{2}\sum_{i=r_k+1}^{R_k} s_{k,i}^2)). \tag{S.1}$$

K denotes the number of layers in the model, and r_k denotes the decomposition rank in the k^{th} layer. In addition, $C_k(r_k)$ denotes the per-layer computation cost in FLOPs and $s_{k,i}$ denotes the i^{th} largest singular value in the k^{th} layer. λ and μ are two hyperparameters.

However, since the authors of LR-S2 did not consider the data-limited setting, they chose to use (S.1) as a regularisation term and solve it by gradient descent. Instead, we solved the above optimisation problem without the need of data by identifying the minimal rank r_k which makes the following inequality hold true for each layer.

$$\lambda C_k(r_k+1) - \lambda C_k(r_k) - \frac{\mu}{2}s_{k,r_k+1}^2 \geq 0 \tag{S.2}$$

[2]https://www.tensorflow.org/lite/performance/measurement

[3]https://github.com/lucaslie/torchprune

S2.3. F-Group

F-Group is a work which manually picked the decomposition scheme as well as the decomposition rank. We reproduced the exact same decomposition configuration shown in the paper and collected the model accuracy without any fine-tuning. We observed this hard-crafted method heavily degrades the model accuracy to almost zero top-1 accuracy.

S3. Additional Results

S3.1. Searching Time

In our framework, the procedure of NAS is completed on a single RTX 2080 Ti or a GTX 1080 Ti GPU. Table S1 summarises the averaged searching time which mainly includes deriving the weights of SVD building blocks, pruning the design space, and the training loop of the sampling parameters.

Table S1: Averaged execution time of NAS.

Model	first iteration		second iteration	
	GPU	hours	GPU	hours
ResNet-18	2080	8.66	1080	4.66
MobileNetV2	1080	10.03	1080	3.07
EfficientNet-B0	1080	14.67	1080	3.07

S3.2. Overfitting in the Search

Due to the data-limited problem setting that the access to training data is difficult, the sampling parameters $\boldsymbol{\theta_i}$ are learned on the validation set. To understand the effect of overfitting, Table S2 compares the performance of the identified design point when searching with the whole validation set and 1% of it. The results show that, under the similar compression ratio, the accuracy difference is no more than 1.07pp.

Table S2: Train the sampling parameters $\boldsymbol{\theta_i}$ using the whole (50k samples) and 1% (500 samples) of ImageNet validation set. Experiment set-up is B1-SD25k.

Model	Val Size	Δ FLOPs (%)	Δ Params (%)	Δ Top-1 (pp)	Δ Top-5 (pp)
ResNet-18	50k	-59.17	-66.77	-5.83	-3.39
	500	-58.96	-67.57	-6.90	-3.95
MobileNetV2	50k	-12.54	-9.00	-9.99	-6.11
	500	-12.37	-8.73	-10.79	-6.53
EfficientNet-B0	50k	-26.53	-17.69	-15.35	-8.99
	500	-26.28	-17.53	-13.53	-7.64

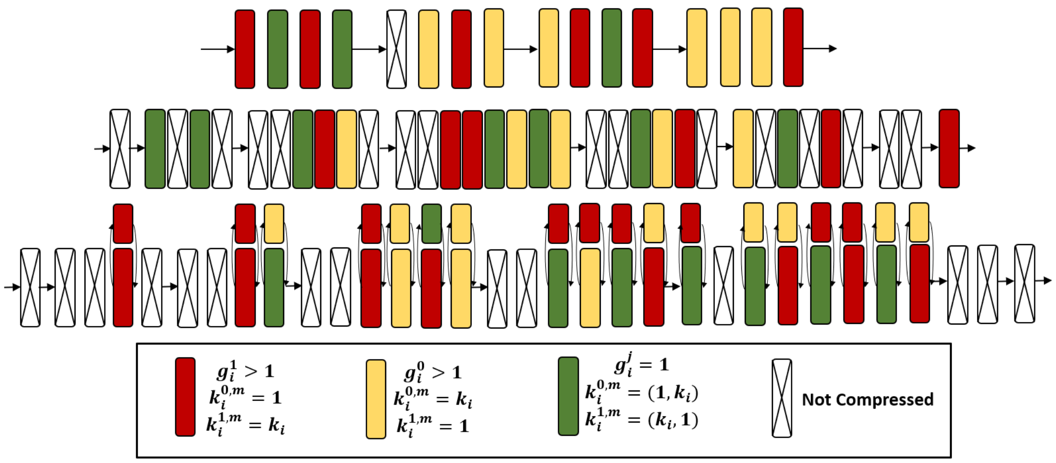

Figure S1: Schematics of the obtained low-rank models. The coloured rectangles represent SVD building blocks and they are clustered into three types (red, yellow, green) based on the value of their design parameters. **Top:** ResNet-18, single-branch design, obtained by the setting that β=48, γ=[0.3,0.7]. **Middle:** MobileNetV2, single-branch design, the setting is β=88, γ=[0.6,0.95], τ_{proxy}=5. **Bottom:** EfficientNet-B0, two-branch design, the setting is β=48, γ=[0.5,0.95], τ_{proxy}=5.

S3.3. Visualise Low-rank Models

In order to visualise the optimal design parameters identified by the gradient-descent NAS, we sketched several examples of the obtained low-rank models in Fig. S1. As our framework targets the compression of every 3×3 convolutions except the first layer in ResNet-18, and every point-wise convolutions in MobileNetV2 and EfficientNet-B0, only these targeted layers are demonstrated while others are hidden from the schematics. These visualised results demonstrate the advantage of our framework over the previous work that we have a larger design space and we can also efficiently explore that space.

S3.4. Complete Post-training Results

Table S3 provides the complete post-training results of SVD-NAS, including metrics of interest and the corresponding hyperparameters to obtain these designs.

Table S3: Post-training results of SVD-NAS. Δ F/P denote the FLOPs and parameters reduction in percentage (%), while Δ Top1/5 denote the accuracy degradation in percentage point (pp). β is the hyperparameter which balances the cross-entropy term and the computation cost term in the loss function of gradient descent NAS. γ and τ_{proxy} control the level of design space pruning. When pruning by FLOPs, γ denotes the values of the grid search, with a default step size at 0.05; for example, γ=[0.3,0.7] means "0.3, 0.35, 0.4 ... 0.7". When pruning by accuracy, τ_{proxy} denotes the tolerance of top-1 accuracy degradation in the units of pp.

ResNet-18	β=48, γ=[0.3,0.7]		β=48, γ=[0.4,0.8]		β=32, γ=[0.4,0.8]	
	Δ F/P	Δ Top1/5	Δ F/P	Δ Top1/5	Δ F/P	Δ Top1/5
B1		-17.58/-12.79		-10.55/-7.22		-6.75/-4.41
B1-SD1k	-59.17/-66.77	-6.72/-4.14	-53.64/-59.79	-4.50/-2.48	-50.39/-58.79	-3.75/-2.10
B1-SD25k		-5.83/-3.39		-3.68/-2.03		-3.05/-1.62
B2	-62.53/-72.61	-20.73/-18.88	-60.74/-69.21	-16.20/-11.12	-58.60/-68.05	-13.35/-9.14
B2-SD25k		-8.98/-5.28		-6.98/-4.07		-5.85/-3.34
ResNet-18	β=16, γ=[0.5,0.9]		β=4, γ=[0.5,0.9]		β=2, γ=[0.5,0.9]	
	Δ F/P	Δ Top1/5	Δ F/P	Δ Top1/5	Δ F/P	Δ Top1/5
B1		-3.32/-2.5		-1.12/-0.84		-0.64/-0.24
B1-SD1k	-38.91/-43.71	-1.68/-0.85	-25.50/-36.70	-0.83/-0.36	-17.30/-27.19	-0.43/-0.21
B1-SD25k		-1.16/-0.61		-0.62/-0.27		-0.44/-0.09
B2	-45.16/-53.64	-4.20/-2.52	-28.86/-42.19	-1.38/-0.75	-13.98/-26.94	-0.67/-0.30
B2-SD25k		-2.13/-0.95		-1.00/-0.38		-0.59/-0.18
MobileNetV2	β=88, γ=[0.6,0.95], τ_{proxy}=5		β=80, γ=[0.6,0.95], τ_{proxy}=5		β=64, γ=[0.6,0.95], τ_{proxy}=5	
	Δ F/P	Δ Top1/5	Δ F/P	Δ Top1/5	Δ F/P	Δ Top1/5
B1		-19.82/-13.15		-15.09/-7.79		-7.66/-4.46
B1-SD1k	-14.17/-10.66	-14.41/-9.00	-12.54/-9.00	-11.63/-7.17	-7.73/-6.70	-5.51/-3.10
B1-SD25k		-13.15/-8.34		-9.99/-6.11		-4.91/-2.88
B2	-16.13/-13.24	-23.16/-15.75	-13.64/-9.70	-17.59/-11.39	-8.09/-6.60	-8.06/-4.70
B2-SD25k		-17.78/-11.35		-11.60/-6.92		-5.36/-2.97
MobileNetV2	β=48, γ=[0.6,0.95], τ_{proxy}=5		β=32, γ=[0.6,0.95], τ_{proxy}=5			
	Δ F/P	Δ Top1/5	Δ F/P	Δ Top1/5		
B1		-5.26/-3.08		-1.90/-0.86		
B1-SD1k	-6.53/-5.82	-4.41/-2.52	-1.92/-2.74	-1.89/-0.84		
B1-SD25k		-4.22/-2.35		-1.76/-0.73		
B2	-6.60/-5.71	-4.69/-2.72	-2.20/-2.34	-1.62/-0.84		
B2-SD25k		-4.10/-2.21		-1.48/-0.78		
EfficientNet-B0	β=88, γ=[0.5,0.95], τ_{proxy}=5		β=80, γ=[0.5,0.95], τ_{proxy}=5		β=64, γ=[0.5,0.95], τ_{proxy}=5	
	Δ F/P	Δ Top1/5	Δ F/P	Δ Top1/5	Δ F/P	Δ Top1/5
B1		-24.65/-15.92		-13.15/-7.45		-10.77/-5.95
B1-SD1k	-26.53/-17.69	-16.36/-9.62	-22.85/-16.06	-9.68/-5.18	-21.04/-15.60	-7.48/-3.90
B1-SD25k		-15.35/-8.99		-9.45/-5.08		-7.40/-3.78
B2	-29.83/-20.60	-23.47/-15.02	-28.52/-20.38	-18.65/-11.20	-25.72/-18.86	-13.96/-8.10
B2-SD25k		-16.70/-10.01		-13.87/-7.87		-10.45/-5.69
EfficientNet-B0	β=48, γ=[0.5,0.95], τ_{proxy}=5		β=32, γ=[0.5,0.95], τ_{proxy}=5		β=16, γ=[0.5,0.95], τ_{proxy}=5	
	Δ F/P	Δ Top1/5	Δ F/P	Δ Top1/5	Δ F/P	Δ Top1/5
B1		-6.99/-3.79		-5.98/-3.05		-2.98/-1.52
B1-SD1k	-18.10/-13.23	-5.60/-2.76	-15.32/-12.08	-4.65/-2.36	-8.97/-6.11	-2.09/-0.98
B1-SD25k		-5.42/-2.79		-4.56/-2.37		-1.99/-0.87
B2	-22.17/-16.41	-10.11/-5.49	-15.95/-12.85	-6.03/-3.16	-8.85/-6.19	-2.91/-1.54
B2-SD25k		-7.67/-4.06		-4.46/-2.32		-1.86/-0.91