NORMALIZED FACTORIZED WEIGHT GENERATION FOR NESTED ELASTIC CONVOLUTIONAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

How should a weight generator be parameterised so that one stored representation supports nested neural networks without making shared weights depend on the selected width or depth? We propose **Spectrum-Shared Generation (SSG)**, a normalized factorized parameterisation for elastic convolutional networks. Each residual kernel is a CP map with layer-specific directions and network-wide component magnitudes. Normalising complete factor columns before channel selection makes every narrower generated kernel an exact prefix sub-tensor of wider ones; fixed layer identities similarly preserve the generated tensor assigned to a retained block under depth selection. The parameterisation is invariant to radial factor rescaling and admits explicit local partial-Jacobian bounds, while configuration-specific normalisers handle activation statistics outside the generator. In controlled comparisons, SSG reduces persistent learnable storage by 5.0–$6.1\times$ across CIFAR-10, CIFAR-100, and Tiny-ImageNet, with mean accuracy differences of 0.36–1.03 points from dense elastic supernets. On CIFAR-100, one 2.084M model serves eight configurations spanning $34\times$ in compute and reaches 76.67% at full capacity, 1.14 points above an OFA baseline retrained under our protocol while using $5.4\times$ less persistent learnable storage.

1 INTRODUCTION

Deep networks are trained once but deployed on a wide range of hardware, from mobile and embedded platforms to servers, whose compute budgets differ by more than an order of magnitude (Howard et al., 2017; Sandler et al., 2018). A single application therefore needs several operating points, and the conventional answer is to train and store one model per point, so what has to be shipped and maintained grows linearly with the number of points supported. Reducing that stored footprint, rather than the inference cost of any single operating point, is the axis this paper addresses.

Elastic networks remove part of this cost. A single supernet is trained so that many of its substructures are directly usable at test time, with slimmable networks and their successors making width elastic (Yu et al., 2019; Yu & Huang, 2019; Yang et al., 2020) and Once-for-All and BigNAS additionally making depth and kernel size elastic (Cai et al., 2020; Yu et al., 2020). This replaces "store $|\mathcal{S}|$ models" with "store one model", and yet it leaves what one model costs unchanged, because the supernet is still a collection of tensors held element by element and every sub-network is a slice of them. Storage is thus pinned to the largest configuration however small the cheapest one is, and sharing stays implicit, in that the entries a narrow sub-network relies on are the ones a wider update overwrites, at a scale nothing controls.

A second and orthogonal redundancy is left untouched. Low-rank factorisation and pruning show that trained kernels are far from full rank (Denton et al., 2014; Jaderberg et al., 2014; Lebedev et al., 2015; Li et al., 2017) and that much of a network is predictable from a small subset of its parameters (Denil et al., 2013). Redundancy can also run across layers: normalised singular-value profiles of the generated stages of a trained dense ResNet-18 form a narrow empirical band (Appendix H). This observation motivates, but does not identify with, the shared CP component magnitudes used below.

This suggests parameterising an elastic network so that it stores a generator rather than a set of weights, keeping ϕ and producing the weight of layer l through a deterministic map $W_l = G(\phi; l)$ evaluated in the forward pass. Within a layer a rank-R CP structure (Kolda & Bader, 2009) writes

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

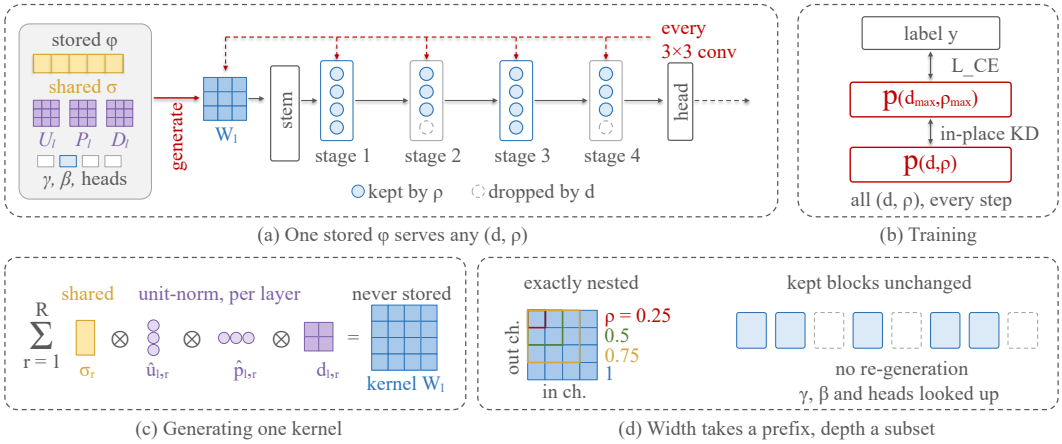


Figure 1: Overview of SSG. In (a) the stored ϕ holds a network-wide spectrum σ, per-layer direction factors (U_l, P_l, D_l), and normalisers and heads indexed by the configuration pair (d, ρ); the generator emits full-width weights once per step and feeds every 3×3 convolution, and a request then keeps a channel prefix and a subset of blocks. Panel (b) is the objective: every configuration receives cross entropy, and all non-teacher configurations additionally receive in-place distillation. Panel (c) expands one kernel as a rank-R sum of a shared magnitude times three unit-norm factors owned by the layer. Panel (d) is why a change of (d, ρ) triggers no re-generation, width being a prefix by Proposition 1 and depth a block subset by Proposition 2.

the kernel as a sum of R atoms, each an outer product of an output, an input and a spatial direction. Across layers the R atom magnitudes $\sigma \in \mathbb{R}^R$, which we call the *spectrum*, are shared network-wide while every layer keeps only unit-norm direction factors, so at R=256 the shared component is 256 scalars in total. We call this parameterisation **Spectrum-Shared Generation (SSG)**.

Compression alone would not make a generator an elastic-inference mechanism. Two decisions do: every layer owns discretely indexed factors, and those factors are normalised over complete columns rather than over the channel prefix a sub-network uses. Proposition 1 then makes each narrower generated kernel an exact prefix sub-tensor of every wider one. Proposition 2 says that, for fixed parameters and absolute layer identities, changing the active block set does not change the tensor assigned to a retained layer. Proposition 4 characterises radial scale invariance and local sensitivity; it does not remove directional or shared-spectrum coupling between configurations. A simplified moment model further motivates indexing normalisers by both depth and width, which our ablation supports empirically.

Our contributions are the following.

- We introduce a factorised generator that breaks the equality between an elastic CNN's persistent learnable storage and its largest dense sub-network.
- We establish exact prefix nesting of generated convolutional tensors, invariance of retained layers' generated-weight identities to the active block set, and conditional local sensitivity bounds.
- We separate the roles of low-rank factorisation, spectrum tying, column normalisation, and configuration-conditioned normalisation through controlled ablations.
- One 2.084M generator serves eight configurations spanning $34\times$ in compute, reaches 76.67% on CIFAR-100, and exceeds OFA retrained under our protocol by 1.14 points while storing $5.4\times$ fewer learnable parameters.

2 Related Work

2.1 Elastic and Dynamic Networks

Yu et al. (2019) showed that a single network can operate at several discrete widths provided each width owns its batch-normalisation statistics; Yu & Huang (2019) extended this to arbitrary widths and introduced sandwich sampling with in-place distillation, and MutualNet (Yang et al., 2020) coupled width with input resolution. DS-Net (Li et al., 2021) makes width input-dependent through learned gates. Once-for-All (Cai et al., 2020) and BigNAS (Yu et al., 2020) extend elasticity to depth and kernel size so that a single training run supports large-scale sub-network search. A parallel line makes computation rather than architecture adaptive through early exits, auxiliary classifiers, learned layer skipping, or skippable self-distilling paths; see Han et al. (2022) for a survey. Since they cut cost by dropping blocks from a full-width backbone, their cheapest configuration stays a large fraction of the full network. In all of the above a sub-network's weights are a subset of the supernet's dense weights, so supernet storage equals largest sub-network storage. Two recent methods loosen that identification. Integral Neural Networks (Solodskikh et al., 2023) treat filters as discretisations of continuous functions so that width becomes a sampling rate, and Neural Metamorphosis (Yang & Wang, 2024) fits an implicit network that emits weights for a queried shape. Both keep storage comparable to the dense backbone, and NeuMeta applies its generator to the final stage only. SSG retains prefix-subtensor semantics (Prop. 1) while replacing bilinear dense channel storage by factors whose size grows linearly in the layer widths at fixed rank cap. This breaks the usual equality between elastic-supernet storage and the largest dense sub-network without making storage independent of the backbone.

2.2 Low-Rank and Tensor Decompositions

CP, Tucker and Tensor-Train decompositions and SVD-type approximations have long been used to compress trained networks or to train structured ones directly (Kolda & Bader, 2009; Denton et al., 2014; Lebedev et al., 2015). Our objective differs from such work, which targets one fixed architecture, in that a single factorised generator parameterises an entire nested family of sub-networks. The multilinear structure is what makes taking a prefix along a channel mode coincide with taking a sub-network, a correspondence element-wise stored weights do not have.

2.3 Weight Generation and Parameter Sharing

HyperNetworks (Ha et al., 2017) generate a main network's weights with a smaller network, while MetaPruning (Liu et al., 2019) and DHP (Li et al., 2020) generate convolutional parameters for channel configurations. Later work also predicts parameters for unseen architectures (Knyazev et al., 2021). SSG belongs to this family, but its generator is an analytic multilinear map rather than a black-box network, which is what allows nestedness, positional stability and Jacobian scaling to be established directly. Cross-layer sharing has also been studied on its own; for example, ALBERT (Lan et al., 2020) ties weights across layers completely. Full tying forces every layer to realise the same transform, whereas SSG shares only the low-dimensional magnitude spectrum and keeps the high-dimensional directions per layer, which obtains the regularisation and gradient-averaging benefits of sharing at almost no cost in layer diversity.

3 Method

Figure 1 sketches SSG, and this section develops it in the order in which its pieces constrain one another. We state what an elastic model must provide, define the generator that produces every 3×3 kernel from a shared spectrum and per-layer directions, and derive both elastic axes from it, width as a slicing rule and depth as an indexing rule. Normalisation, the one component no generator can absorb, is treated separately before the objective.

3.1 Problem Setup

Given a backbone and a target configuration set

$$\mathcal{S} = \{(d, \rho) : d \in \mathcal{D},\ \rho \in \mathcal{R}\}, \tag{1}$$

where d specifies how many residual blocks each stage keeps and $\rho \in (0, 1]$ the retained channel fraction, we want to train one parameter set ϕ from which every $(d, \rho) \in \mathcal{S}$ yields a directly usable sub-network, subject to

$$|\phi| \ll \sum\nolimits_{(d,\rho)\in\mathcal{S}} |W_{d,\rho}| \quad \text{and} \quad |\phi| \ll |W_{\max}|. \tag{2}$$

The second inequality is what separates SSG from existing elastic methods, which satisfy $|\phi| = |W_{\max}|$ by construction.

3.2 Spectrum-Shared CP Generator

Let the l-th convolution have weight tensor $W_l \in \mathbb{R}^{C^l_{\text{out}} \times C^l_{\text{in}} \times k \times k}$, and flatten its spatial modes into a single index of size k^2. Given a global rank cap R, its effective rank is

$$R_l = \min\big(R, C^l_{\text{in}}, C^l_{\text{out}}\big)\,. \tag{3}$$

The generator stores

$$\phi_{\text{gen}} = \big\{\,\sigma\,\big\} \cup \big\{\,U_l,\ P_l,\ D_l\,\big\}_{l=1}^{L}, \tag{4}$$

with $\sigma \in \mathbb{R}^R$ shared by all layers, channel factors $U_l \in \mathbb{R}^{C^l_{\text{out}} \times R_l}$ and $P_l \in \mathbb{R}^{C^l_{\text{in}} \times R_l}$, and a spatial bank $D_l \in \mathbb{R}^{k^2 \times R}$ whose first R_l columns are used by layer l. Writing $\hat{u}_{l,r} = U_l[:, r]/\|U_l[:, r]\|_2$ for the column-normalised factors (Salimans & Kingma, 2016) and likewise for $\hat{p}_{l,r}$, $\hat{d}_{l,r}$, the generation rule is

$$W_l \;=\; G(\phi; l) \;=\; \sum_{r=1}^{R_l} \sigma_r\ \hat{u}_{l,r} \otimes \hat{p}_{l,r} \otimes \hat{d}_{l,r}, \tag{5}$$

that is, $W_l[o, i, :] = \sum_r \sigma_r\, \hat{u}_{l,r}[o]\, \hat{p}_{l,r}[i]\, \hat{d}_{l,r}$.

Equation equation 5 splits each layer's weight into two semantically distinct parts, where σ decides how much energy each of the R outer-product atoms contributes and $\hat{u}, \hat{p}, \hat{d}$ decide where those atoms point. Tying σ across layers imposes a common envelope on CP component magnitudes; because the directions remain free, different layers still realise different linear maps. It saves only $(L-1)R$ scalars relative to per-layer spectra, so low-rank factorisation—not tying itself—produces nearly all of the storage reduction. Its role is to make component magnitudes commensurate across layers by construction, a distinction tested in Section 5.3. All 3×3 convolutions are generated by Eq. equation 5. The stem and the 1×1 downsampling convolutions total only 0.17M and are structurally different, so they are kept dense.

3.3 Elastic Width by Prefix Nesting

For a width ratio ρ, let $c_{\text{o}} = \lceil \rho\, C^l_{\text{out}} \rceil$ and $c_{\text{i}} = \lceil \rho\, C^l_{\text{in}} \rceil$. The sub-network simply reads a prefix sub-block of the generated tensor,

$$W_l^{(\rho)} \;=\; W_l\big[:c_{\text{o}},\ :c_{\text{i}},\ :,\ :\big]. \tag{6}$$

Proposition 1 shows this is exactly nested and that one evaluation of Eq. equation 5 serves every width. Two consequences matter in practice. First, widths relate by containment rather than by recomputation, which matches the subset semantics that in-place distillation relies on. Second, had we normalised after slicing, different widths would receive mutually rescaled weights and nestedness would break, whereas normalising over full columns avoids this.

3.4 Elastic Depth by Positional Stability

A depth configuration $d = (d_1, \ldots, d_4)$ keeps the first d_s residual blocks of stage s. By Proposition 2, $\{W_l\}$ and the set of active blocks are decoupled, so which weights a layer receives is determined by its identity rather than by its ordinal position within the current sub-network. For contrast, suppose the layer index were mapped to a continuous coordinate $t_l = l/(L-1)$ and $W_l = G(\sigma, t_l)$, as an implicit-representation view would suggest. Deleting an intermediate block would then shift the coordinates of all subsequent blocks and change their weights, so one physical layer would be asked to realise different transforms under different depth configurations. The discrete per-layer indexing of Eq. equation 5 makes depth elasticity a pure subset selection instead.

Algorithm 1 One SSG training step

Require: batch (x, y); parameters ϕ; configuration set $\mathcal{S}$

1: $\forall l$: $W_l \leftarrow \sum_r \sigma_r \, \hat{u}_{l,r} \otimes \hat{p}_{l,r} \otimes \hat{d}_{l,r}$ {once, Eq. equation 5}
2: $p_{\max} \leftarrow$ forward with $(d_{\max}, \rho_{\max})$; $t \leftarrow \mathrm{sg}[p_{\max}]$
3: **for** each $(d, \rho) \in \mathcal{S}$ **do**
4: $\quad \forall l$: $W_l^{(\rho)} \leftarrow W_l[:c_o, :c_i]$ {slice only, Eq. equation 6}
5: $\quad$ run the first d_s blocks of each stage s
6: $\quad$ use BN and head indexed by (d, ρ)
7: $\quad$ accumulate $\mathcal{L}_{\mathrm{CE}} + \lambda \, \mathrm{KL}(t \,\|\, p_{d,\rho})$
8: **end for**
9: update ϕ by SGD on the accumulated loss

3.5 Configuration-Conditioned Normalisation

That normalisation statistics follow the active configuration is already standard for width elasticity (Yu et al., 2019). For intuition when depth is elastic as well, consider the simplified population-moment model

$$s(d, \rho) \;=\; \rho \, \big(v_0 + q(d) \, v\big), \tag{7}$$

where $q(d)$ is the number of active residual blocks. This approximation assumes comparable, weakly correlated residual contributions (He et al., 2016b; De & Smith, 2020) and width-independent weight-entry second moments; it is a motivation rather than an architectural identity. We therefore give every $(d, \rho) \in \mathcal{S}$ its own BN affine parameters and running statistics (Ioffe & Szegedy, 2015) and its own classifier head. These parameters bypass the generator and are indexed directly, and at $|\mathcal{S}| = 8$ they total 0.31M, or 14.6% of stored parameters as Appendix C details. Section 5.3 measures the benefit of this design.

3.6 Training Objective

Every step forwards all configurations in $\mathcal{S}$, a full-coverage variant of sandwich sampling, and distils in place (Hinton et al., 2015) from the largest sub-network,

$$\mathcal{L} = \sum_{(d,\rho)\in\mathcal{S}} \Big[\mathcal{L}_{\mathrm{CE}}\big(p_{d,\rho}, y\big) + \lambda \, \mathrm{KL}\Big(\mathrm{sg}\big[p_{d_{\max},\rho_{\max}}\big] \,\Big\|\, p_{d,\rho} \Big) \Big], \tag{8}$$

where $\mathrm{sg}[\cdot]$ stops gradients and λ weights distillation. By Proposition 1, Eq. equation 5 is evaluated once per step and its output is shared by all widths. This makes the *weight-generation* overhead independent of $|\mathcal{R}|$; elastic training still performs one forward path and loss evaluation per configuration. Algorithm 1 summarises one step.

After materialisation, inference has no generator arithmetic on the critical path. Because Eq. equation 5 does not depend on the input, W_l can be materialised once offline and cached, after which running a selected configuration is arithmetically identical to running a dense network of the same shape. The persistent learnable package, BN running-statistic buffers, optional dense cache, and peak training memory are therefore distinct quantities; all parameter counts below refer to the first unless stated otherwise.

4 Theoretical Analysis

We state the four properties that shape SSG; full proofs are in Appendix A.

Proposition 1 (Exact prefix nesting of generated kernels)**.** *Assume every factor column used by layer l is nonzero. For any $\rho_1 < \rho_2$, $W_l^{(\rho_1)} = W_l^{(\rho_2)}\big[:c_o(\rho_1), :c_i(\rho_1)\big]$. Hence all width-specific generated kernels come from one full-width generation followed by slicing. This statement does not include configuration-specific normalisers or classifier heads.*

The reason is that Eq. equation 5 acts element-wise on the channel indices, and the normalising denominators $\|U_l[:, r]\|$ are computed over full columns, hence do not depend on ρ. Restricting

the index ranges therefore returns the sub-block of Eq. equation 6 with no ρ-dependent rescaling anywhere.

Proposition 2 (Generated-weight identity under depth selection)**.** *For fixed parameters and fixed absolute layer labels, W_l depends only on (σ, U_l, P_l, D_l) and not on the depth configuration d. Consequently, removing block $l' \neq l$ leaves W_l unchanged. This does not imply invariance of the retained layer's input, activation, or the network output.*

Proposition 3 (Mismatch under a simplified moment model)**.** *Under Eq. equation 7, if normalisation statistics are indexed by ρ alone and set to $\bar{s}(\rho) = \mathbb{E}_d[s(d,\rho)]$, then the residual mismatch satisfies*

$$\max_{d\in\mathcal{D}} \big|s(d,\rho) - \bar{s}(\rho)\big| \;=\; \rho\,|v|\, \max_{d} \big|q(d) - \mathbb{E}_d q(d)\big|. \tag{9}$$

For two equally weighted depth counts this is $\frac{1}{2}\rho|v|(q_{\max} - q_{\min})$. Thus one width-only population statistic cannot exactly match both depths under this model, whereas (d,ρ) indexing can represent a separate statistic for each trained configuration.

Proposition 4 (Scale invariance and local partial-Jacobian bounds)**.** *Assume all factor columns are nonzero. Equation equation 5 is invariant to positive rescaling of any factor column, $U_l[:,r] \mapsto c\,U_l[:,r]$ with $c > 0$, and*

$$\left\| \frac{\partial W_l}{\partial \sigma_r} \right\|_F = 1, \qquad \left\| \frac{\partial W_l}{\partial U_l[:,r]} \right\|_{\mathrm{op}} \;\le\; \frac{|\sigma_r|}{\|U_l[:,r]\|_2}, \tag{10}$$

with analogous bounds for $P_l[:,r]$ and $D_l[:,r]$. The derivatives are therefore bounded on parameter regions where factor norms are bounded away from zero and $|\sigma_r|$ is bounded.

For the unnormalised CP form $W_l = \sum_r u_r \otimes p_r \otimes d_r$ the same derivative is $\|p_r\|\|d_r\|$, which multiplies the other factors' scales together and may drift during training. Column normalisation removes radial scale sensitivity from the generated tensor and separates an atom's magnitude, carried by σ, from its direction. It does not remove coupling through updates to the directions or the shared spectrum.

With the implementation's rank cap in Eq. equation 3, the persistent generator package is

$$|\phi_{\mathrm{gen}}| = R + \sum_l \left[R_l (C^l_{\mathrm{out}} + C^l_{\mathrm{in}}) + k^2 R \right], \tag{11}$$

against $\sum_l C^l_{\mathrm{out}} C^l_{\mathrm{in}} k^2$ for dense storage. The $k^2 R$ term reflects the implementation's fixed-width per-layer spatial bank, of which each layer uses the first R_l columns. The generator itself does not depend on $|\mathcal{S}|$, although each additional trained configuration adds a normaliser and classifier slot.

5 Experiments

5.1 Protocol

We evaluate on CIFAR-100 and CIFAR-10 (Krizhevsky, 2009) with a CIFAR ResNet-18 (He et al., 2016a). The trained set combines depths $(1,1,1,1)$ and $(2,2,2,2)$ with widths $\{0.25, 0.5, 0.75, 1.0\}$, giving eight configurations over 16–555 MFLOPs. We train for 200 epochs using SGD (momentum 0.9, weight decay 5×10^{-4}), batch size 256, initial learning rate 0.2 with cosine decay, label smoothing 0.1, and $\lambda = 0.5$. Every configuration receives cross entropy and, except for the largest teacher itself, in-place distillation. The dense elastic supernet and all generator ablations use the same implementation, configuration set, objective, and schedule. We additionally reimplement and train OFA (Cai et al., 2020) on the same ResNet-18 backbone, configuration set, and 200-epoch recipe. Other literature-reported elastic methods remain contextual references only (Appendix B).

Counting convention. We report persistent learnable parameters. For CIFAR-100, Eq. equation 11 gives 1,605,888 generator parameters; dense stem and shortcuts add 173,760, BN affine parameters 47,680, and eight heads 256,800, for 2,084,128 total. BN running means and variances add 47,680 non-learnable scalars on disk. The generated dense-weight cache is optional and is not counted in the persistent package.

Data	Parameterisation	Store	Mean	Largest
C100	SSG, paper run	2.084	–	**76.67**
	OFA, retrained	11.200	–	75.53
C100	SSG, R=256	2.084	72.40	75.55
	Stored, full width	11.501	72.76	76.84
	Stored, matched size	2.124	65.03	72.70
C10	SSG, R=256	1.853	92.90	94.30
	Stored, full width	11.270	93.93	95.38
	Stored, matched size	2.026	90.09	93.80

Table 1: Elastic-supernet comparisons. OFA is our reimplementation trained under the paper protocol. The remaining blocks are stricter same-code controls, where "Stored" changes only the 3×3 parameterisation. Store is learnable parameters in M; Mean averages all eight configurations. Full-width rows have equal per-configuration FLOPs; matched-size rows use width multiplier 0.42.

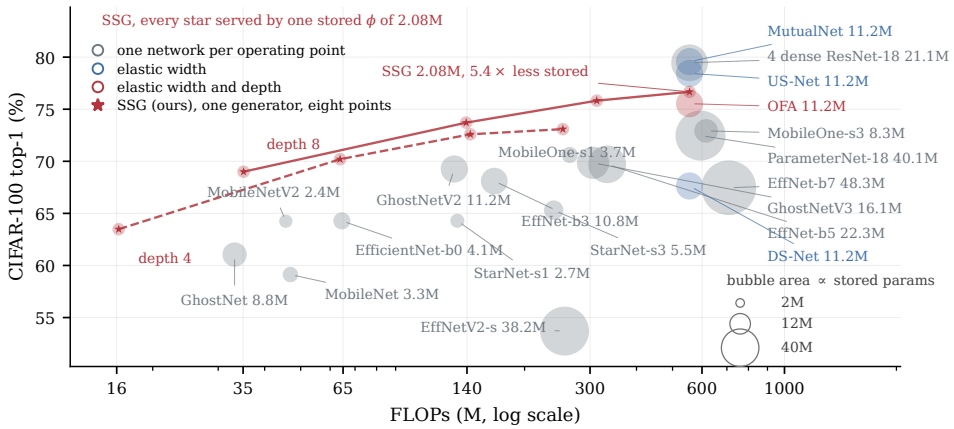


Figure 2: CIFAR-100 accuracy against inference compute, with bubble area proportional to persistent learnable parameters. All points are from our training runs. Elastic baselines use the same ResNet-18 backbone and training length; compact single-configuration architectures use different backbones and are included as contextual rather than architecture-controlled comparisons. The eight red stars are served by one 2.084M SSG model.

5.2 Controlled Storage–Accuracy Comparison

The direct baseline replaces every generated 3×3 kernel by a stored dense tensor while retaining the same width slicing, depth set, normalisers, heads, losses, and training code. Table 1 compares both equal-compute and approximately equal-storage regimes. In the headline elastic-method comparison, SSG reaches 76.67% at its largest configuration against 75.53% for our OFA reimplementation, a 1.14-point improvement with 2.084M rather than 11.200M learnable parameters (5.4× less storage). At equal compute on CIFAR-100, generation reduces learnable storage from 11.501M to 2.084M (5.5×) and changes mean accuracy from 72.76% to 72.40%, a 0.36-point difference. Three complete SSG runs give means 72.40, 72.23, and 72.81 (standard deviation 0.30), so this difference is comparable to observed run-to-run variation. At approximately equal storage, the dense supernet must reduce its base width to 0.42 and falls by 7.37 mean points. CIFAR-10 shows the same trade-off: 6.1× lower storage at equal compute for a 1.03-point mean difference, and a 2.81-point advantage at approximately equal storage.

5.3 What Each Design Decision Contributes

Table 2 separates spectrum tying from factorisation and column normalisation. A spectrum per layer adds only 3,840 parameters and changes mean accuracy by +0.09, below the three-seed standard deviation; tying is therefore a cross-layer magnitude constraint, not the source of compression or

Variant	Store	Mean	Atom spread
Tied spectrum, normalised	2.084	72.40	0.05
Spectrum per layer	2.088	72.49	0.23
No column normalisation	2.084	72.06	3.61
Generated projections	1.955	72.18	0.06

Table 2: Controlled CIFAR-100 ablations. Store is M learnable parameters; atom spread is the standard deviation in decades of realised rank-one component magnitudes.

a measurable accuracy gain. Without column normalisation, mean accuracy changes by -0.34, while the standard deviation of realised atom magnitudes grows from 0.05 to 3.61 decades. This directly supports the claimed control of radial scale, without implying a globally well-conditioned optimisation problem. Generating the 1×1 projections at rank 32 further reduces the package to 1.955M with a -0.22 mean change, showing that the dense carve-out is an implementation choice rather than a structural limitation.

Configuration-conditioned normalisation. At rank 128, replacing width-only normalisers and heads by (d, ρ)-indexed slots improves the eight-configuration mean by 0.78 points. The shallow configurations gain 1.13 points on average versus 0.43 for the deep configurations; narrow configurations also gain more empirically. We report these as observations consistent with depth-dependent moments, not as validation of a channel-count law for BN estimator variance. Full per-configuration results are in Appendix D.

5.4 Beyond the CIFAR Training Setting

A larger dataset. Tiny-ImageNet has 200 classes and 100K training images at 64×64. We stride the stem by two so that stage resolutions and per-configuration FLOPs remain within 1% of the CIFAR setting, then train SSG and the same-code stored supernet for 200 epochs. The generator retains the ordering of all eight operating points and reaches a 56.03% mean against 56.87%, a 0.84-point difference at $5.0\times$ less learnable storage. Together with Table 1, the controlled mean difference remains between 0.36 and 1.03 points across datasets with 10, 100, and 200 classes; Appendix P reports all eight Tiny-ImageNet configurations.

Configurations not present during training. The structural properties expose weights for additional depth patterns and per-stage width assignments, but their BN statistics are stale. Forward-only recalibration, using no labels or gradients, expands the CIFAR-100 Pareto front from six trained points to twelve when all 2^4 depth patterns are considered, and to 26 over 756 genuinely mixed per-stage width assignments. In contrast, globally unseen uniform widths remain ineffective because they lack matching affine parameters and classifier heads. Thus generated-kernel elasticity extends beyond the trained set only as far as the directly indexed parameters permit; Appendix E gives the complete sweeps and controls.

Distribution shift. On CIFAR-10-C and CIFAR-100-C, generated models exhibit clean-to-corrupted drops comparable to independently stored dense models. At similar FLOPs, deeper-and-narrower configurations outperform wider-and-shallower ones in all six comparisons, by 1.36–3.80 mCA-15 points on CIFAR-100-C and 0.68–1.95 on CIFAR-10-C. This result is not an additional compression claim; it shows why exposing both elastic axes can matter after deployment. Appendix F reports all corruptions and configurations.

Training cost. Training evaluates all eight configurations per batch, so it is substantially more expensive than one fixed network even though each layer's weights are generated once per step. A 200-epoch CIFAR-100 run takes approximately two hours on one RTX 4090 in our implementation. After offline materialisation, configuration inference uses the same convolution arithmetic as the corresponding dense sub-network.

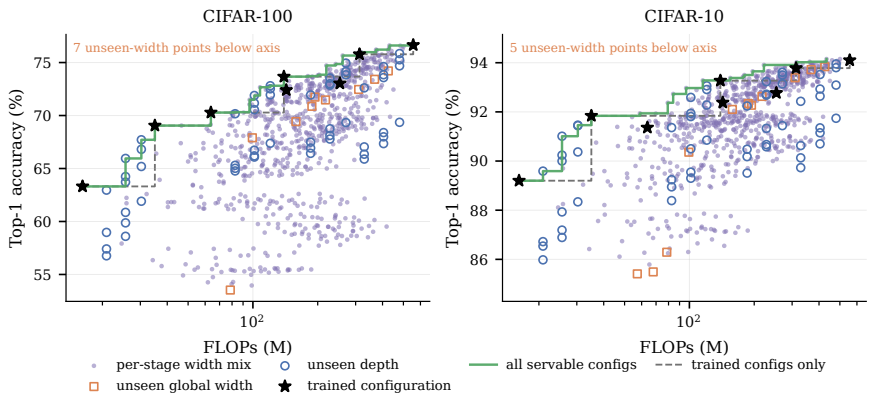


Figure 3: Configurations served after forward-only BN recalibration. Depth combinations and per-stage width mixes densify the Pareto front on both datasets, whereas globally unseen uniform widths remain below it; the worst points fall outside the plotted range. Black stars denote the eight trained configurations.

6 Conclusion

We presented SSG, which keeps a rank-R generator in place of the dense weights of an elastic network's largest sub-network. Every residual 3×3 kernel is a CP product of per-layer directions and network-wide component magnitudes. Full-column normalisation makes generated widths exact prefix sub-tensors, while fixed layer identities preserve the generated tensor assigned to each retained block. On CIFAR-100, a 2.084M SSG model serves eight configurations over a $34\times$ compute range and reaches 76.67%, 1.14 points above OFA retrained under our protocol while using $5.4\times$ less learnable storage. Against a stricter same-code dense elastic supernet, SSG uses $5.5\times$ less storage with a 0.36-point mean-accuracy difference comparable to observed seed variation. The gain is architecture-dependent: low-rank factorisation supplies nearly all storage reduction, while normalisers, heads, and any ungenerated projections remain dense. Extending the controlled analysis to larger convolutional backbones, including their pointwise projections, is the main next step.

Reproducibility Statement

The architecture, effective-rank rule, configuration set, optimisation recipe, parameter-counting convention, and recalibration protocol are specified in the paper and appendix. The accompanying implementation exposes switches for the dense-supernet, per-layer-spectrum, unnormalised, and generated-projection controls used in the ablations.

AI Use Statement

We used generative AI tools for language editing, manuscript restructuring, and consistency checking. We did not use them to create experimental measurements or bibliographic sources. The authors checked all AI-assisted text, mathematical derivations, citations, code references, and reported numbers and take responsibility for the final content.

References

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations (ICLR)*. OpenReview.net, 2020.

Soham De and Samuel L. Smith. Batch normalization biases residual blocks towards the identity function in deep networks. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, 2020.

Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando de Freitas. Predicting parameters in deep learning. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, 2013.

Emily Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, 2014.

David Ha, Andrew Dai, and Quoc V. Le. Hypernetworks. In *International Conference on Learning Representations (ICLR)*. OpenReview.net, 2017.

Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11): 7436–7456, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2016b.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2015.

Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, 2014.

Boris Knyazev, Michal Drozdzal, Graham W. Taylor, and Adriana Romero-Soriano. Parameter prediction for unseen deep architectures. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, 2021.

Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51 (3):455–500, 2009.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations (ICLR)*. OpenReview.net, 2020.

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

Changlin Li, Guangrun Wang, Bing Wang, Xiaodan Liang, Zhihui Li, and Xiaojun Chang. Dynamic slimmable network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2021.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient ConvNets. In *International Conference on Learning Representations (ICLR)*. OpenReview.net, 2017.

Yawei Li, Shuhang Gu, Kai Zhang, Luc Van Gool, and Radu Timofte. DHP: Differentiable meta pruning via hypernetworks. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2020.

Zhuang Liu, Haichao Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. MetaPruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2019.

Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, 2016.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018.

Kirill Solodskikh, Azim Kurbanov, Ruslan Aydarkhanov, Irina Zhelavskaya, Yury Parfenov, Dehua Song, and Stamatios Lefkimmiatis. Integral neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2023.

Taojiannan Yang, Sijie Zhu, Chen Chen, Shen Yan, Mi Zhang, and Andrew Willis. MutualNet: Adaptive convnet via mutual learning from network width and resolution. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2020.

Xingyi Yang and Xinchao Wang. Neural metamorphosis. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2024.

Jiahui Yu and Thomas Huang. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2019.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *International Conference on Learning Representations (ICLR)*. OpenReview.net, 2019.

Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. BigNAS: Scaling up neural architecture search with big single-stage models. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2020.