WHEN THE LABEL SPACE SHRINKS: TASK-ADAPTIVE DEPTH COMPRESSION VIA RECOMPUTED GRADIENT–RESIDUAL SENSITIVITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern neural networks are often trained to support a broad range of functionalities, while a particular deployment may require only a small subset of them. This mismatch raises a natural compression opportunity: if the required functionality becomes narrower, can the model itself become smaller? We study label-space reduction as a concrete instance of this setting, where a reference classifier is trained on a broad source label space but deployment focuses only on a smaller target subset. Rather than shrinking the classifier head, we ask whether reduced target requirements expose removable computation within the network backbone. We propose Task-ReGRIP, a training-free method for task-adaptive depth compression. For each removable residual block, Task-ReGRIP combines the representation change introduced by the block with the target-task loss gradient to estimate the first-order perturbation caused by bypassing that block. It then iteratively removes the least sensitive block and recomputes the sensitivities of all remaining candidates after every structural change. No recovery training or parameter updates are performed during compression. We evaluate Task-ReGRIP on CIFAR-100 using growth-trained ResNet and MobileNetV2 models across multiple target label-space sizes. Across 12 compression paths, the accuracy on the evaluation half of the selected classes' official test images at the maximum deletion budget ranges from -1.2 to $+1.0$ percentage points relative to the corresponding uncompressed reference model, while the ResNet models remove approximately one third of their blocks. These results show substantial depth removal with limited target-task accuracy change in the evaluated settings, without recovery training.

1 INTRODUCTION

Modern neural networks are often trained to support a broad range of classes and deployment scenarios within a single model. This generality is convenient, but it also creates a mismatch between what a model is trained to support and what a particular deployment actually needs. A model may be sized for a large set of potential functionalities, while only a small subset is relevant in a given environment. Nevertheless, deploying that model still incurs the cost of executing the full network. This raises a simple question: if the required functionality becomes narrower, can the model itself become smaller? Many model-compression methods do not explicitly target this setting. They typically seek to reduce computation while preserving performance on the original task as a whole. When deployment requirements have changed, however, preserving all of the original functionality may be unnecessarily restrictive. A deployment may only require a subset of the capabilities supported by the source model, while degradation on unused functionality is acceptable. **Rather than asking a smaller model to reproduce the entire capability set of the original model, we ask how much computation can be removed while preserving only the functionality that remains relevant to the target deployment.**

We study label-space reduction as a concrete instance of this problem. A reference classifier is trained on a broad source label space, while deployment is concerned with only a smaller subset of those classes. Importantly, our goal is not to obtain savings by trivially shrinking the classifier head. We retain the original output space and prediction rule, and instead ask whether the reduced deployment requirement makes computation inside the network backbone unnecessary. This setting

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.

exposes a task-dependent form of redundancy: a block that matters for the full source task need not be equally important for a restricted target task.

Identifying such redundancy is challenging for two reasons. First, block importance should be measured with respect to the target task rather than the source task as a whole. Second, block importance is not static during compression. Removing one block changes the activations seen by downstream blocks and the gradients propagated to earlier blocks, so a ranking computed once on the original network may become stale after structural changes. We therefore seek a criterion that is both target-task specific and recomputed as the network changes.

To this end, we introduce Task-ReGRIP (Task-Adaptive Recomputed Gradient–Residual Iterative Pruning), a training-free method for structured depth compression. Task-ReGRIP operates on residual blocks that can be replaced by the identity map without changing tensor shapes. For each candidate block, it combines two quantities: the representation change introduced by the block, captured by its effective residual, and the sensitivity of the target-task loss to that representation, captured by the activation gradient. The negative of their inner product gives a local first-order approximation to the signed per-example loss change induced by bypassing the block. We take the absolute value of this inner product for each example and then average across examples to obtain Gradient–Residual Sensitivity (GRS) and iteratively remove the least sensitive block. After each deletion, all remaining candidates are rescored in the modified network. No recovery training is performed, and the parameters of retained blocks remain fixed throughout compression.

We evaluate Task-ReGRIP on CIFAR-100 using growth-trained ResNet and MobileNetV2 models and target tasks with different numbers of classes. Across the three ResNet architectures, the maximum deletion budget removes approximately one third of the original blocks. Across 12 target-task compression paths, the change in accuracy on the evaluation half of the selected classes' official test images at the maximum deletion budget ranges from -1.2 to $+1.0$ percentage points relative to the corresponding uncompressed reference model. On the ten-class target task, training-free Task-ReGRIP also achieves accuracy comparable to the evaluated SLEB-style deletion baseline at the last shared deletion budget for each architecture. These results show that, for the evaluated target tasks, substantial depth can be removed without updating the retained parameters while keeping accuracy near the corresponding reference.

Our contributions are: 1. We study model compression under reduced deployment requirements, using label-space reduction as a concrete setting in which only part of the original model functionality must remain relevant at deployment. 2. We introduce Task-ReGRIP, a training-free structured depth-compression method that estimates the target-task effect of removing a residual block through Gradient–Residual Sensitivity and recomputes this sensitivity after every deletion. 3. We show that substantial depth can be removed while preserving target-task performance across multiple CNN architectures and target label-space sizes, without recovery training or parameter updates during structure selection.

2 Related Work

Network pruning and structured compression. Network pruning reduces model cost by removing redundant parameters or structures. Early work estimates parameter importance using second-order information (LeCun et al., 1989; Hassibi & Stork, 1992), while later methods demonstrate substantial redundancy through magnitude-based sparsification and deep compression (Han et al., 2015b;a). Structured pruning removes regular units such as filters and channels using weight magnitude, reconstruction error, learned sparsity, propagated importance, or resource-aware objectives (Li et al., 2017; Luo et al., 2017; He et al., 2017; Liu et al., 2017; Yu et al., 2018; Gordon et al., 2018; He et al., 2018). DepGraph further generalizes structural pruning by explicitly modeling dependencies among coupled network components (Fang et al., 2023). These methods primarily compress a model while preserving its original task, whereas we study whether reduced deployment requirements expose additional redundancy along network depth.

Sensitivity-based pruning. Another line of work estimates the effect of pruning through loss sensitivity. Taylor pruning approximates the change in the objective caused by removing parameters or structured units (Molchanov et al., 2016; 2019), while SNIP, GraSP, and SynFlow develop gradient- or sensitivity-based criteria for identifying important connections before or during training (Lee

et al., 2018; Wang et al., 2020; Tanaka et al., 2020). Task-ReGRIP is related to these methods in using first-order information, but its perturbation is tied directly to replacing an identity-compatible residual block by the identity map. It couples the target-task activation gradient with the block's effective input–output residual and recomputes this deletion-specific sensitivity after every structural change.

Depth and block-level compression. Depth compression removes or replaces entire layers or blocks. LayerDrop trains Transformer models that support subnetworks of different depths (Fan et al., 2019), while BERT-of-Theseus learns compact successor modules through progressive replacement (Xu et al., 2020). More recent methods directly exploit block redundancy: SLEB greedily removes redundant Transformer blocks (Song et al., 2024), and ShortGPT uses block input–output similarity to identify removable layers (Men et al., 2025). Task-ReGRIP similarly operates at block granularity, but conditions deletion on a restricted target task, keeps surviving parameters fixed, and repeatedly rescales block importance as the network structure changes.

Class-aware and task-specific specialization. Several works specialize pretrained classifiers when only a subset of classes is relevant at deployment. CAP'NN constructs personalized subnetworks using class-dependent neuron activity (Hemmat et al., 2020; 2022), while OCAP performs class-aware channel pruning for edge deployment followed by lightweight fine-tuning (Ma et al., 2023). Other methods incorporate target-class information into filter or channel pruning, including class-aware structured pruning (Jiang et al., 2024), task-aware adaptive pruning (Wang et al., 2024a), task-level customized pruning with class-specific gates (Wang et al., 2024b), and task-specific pruning based on target-class feature importance (Wang et al., 2025; 2026). These works establish that the computation required by a model can depend on deployment-specific class composition. Our work studies a complementary form of specialization: instead of pruning filters, channels, or neurons, Task-ReGRIP removes complete residual blocks from a fixed pretrained model using target-task gradient–residual sensitivity, without recovery training during structure selection.

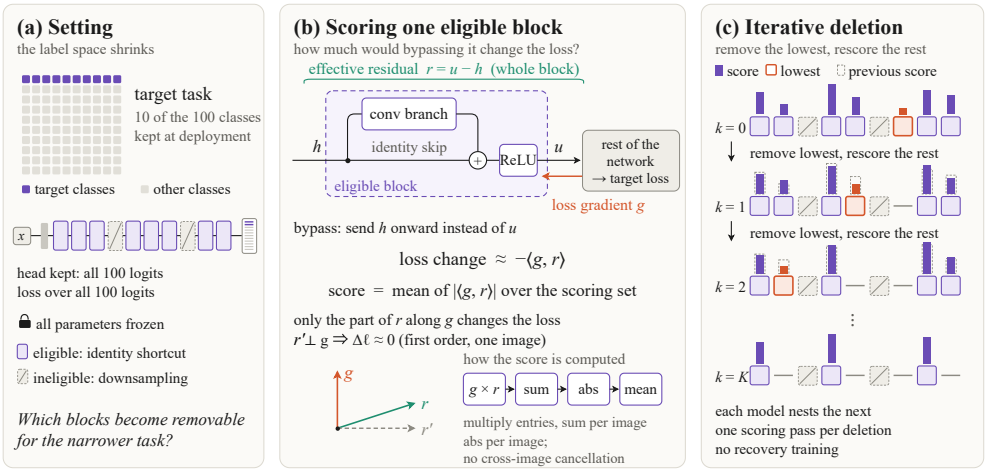


Figure 1: Overview of Task-ReGRIP. Gradient–residual sensitivity on target-class examples guides iterative block bypass and rescoring, while the original 100-way head remains unchanged. The ten-class task and ResNet-style block are examples; score bars are schematic.

3 Method: Task-ReGRIP

Task-ReGRIP (Task-Adaptive Recomputed Gradient–Residual Iterative Pruning) starts from a reference model trained on a source task and uses examples from a subset of target classes to determine the order in which entire blocks are removed. It first restricts deletion to blocks that can be replaced by the identity map, then estimates each block's effect on the target-task loss. After each deletion, it

recomputes the scores of the remaining candidates in the resulting model. The output is a sequence of models spanning deletion budgets from zero to a prescribed maximum; the parameters of retained blocks remain fixed throughout. Figure 1 summarizes the task setting, block scoring, and iterative deletion procedure.

3.1 Inputs and eligible blocks

Let $\mathcal{Y}_s$ be the source-task label set and $\mathcal{Y}_t \subsetneq \mathcal{Y}_s$ the target-task label set at deployment. The inputs are a reference model M_0 trained on the source task, a labeled target-task scoring set D_p, and a maximum deletion budget K. The set D_p determines the deletion order but is not used to update model parameters. The classifier continues to produce $|\mathcal{Y}_s|$ logits, and target labels retain their source-task indices. Thus, for a target example (x_i, y_i) and a logit vector z, cross-entropy is computed over the full source label space:

$$\ell_i(z) = -\log \frac{\exp z_{y_i}}{\sum_{c\in\mathcal{Y}_s} \exp z_c}, \quad y_i \in \mathcal{Y}_t, \mathcal{L}_i(M) \quad = \ell_i(M(x_i)). \tag{1}$$

Retaining the original output space keeps the loss and prediction rule consistent before and after compression.

Let $\mathcal{C}(M)$ denote the set of blocks in the current model M that have not yet been removed, have a strict identity skip connection, and can be replaced in their entirety by the identity map without changing the shape of any downstream tensor. Write $\mathcal{C}_0 = \mathcal{C}(M_0)$. An eligible block must have inputs and outputs of the same shape; blocks with projection shortcuts, downsampling, or channel changes are excluded. For $S \subseteq \mathcal{C}_0$, let M_S denote the model in which every block in S has been replaced by the identity map. Given $K \leq |\mathcal{C}_0|$, we formalize structure selection at a fixed deletion count $k \leq K$ by minimizing cross-entropy on the scoring set:

$$\widehat{R}_p(M) = \frac{1}{|D_p|} \sum_{(x_i,y_i)\in D_p} \mathcal{L}_i(M), S_k^\star \quad \in \underset{\substack{S\subseteq\mathcal{C}_0\\|S|=k}}{\arg\min}\, \widehat{R}_p(M_S). \tag{2}$$

This fixed-budget objective entails a combinatorial search over subsets of eligible blocks. Task-ReGRIP neither enumerates these subsets nor claims to find a global optimum. Instead, it constructs a nested sequence $M_0, \ldots, M_K$ one deletion at a time using the local loss-stability criterion developed below.

3.2 AG×Residual: a target-task deletion criterion

Consider an eligible block B_l in the current model. For example i, let $h_{l,i}$ be its input and $u_{l,i} = B_l(h_{l,i})$ its output. Let G_l denote the network downstream of the block, and define $f_{l,i}(v) = \ell_i(G_l(v))$. We use two tensors directly associated with replacing the block by the identity map:

$$r_{l,i} = u_{l,i} - h_{l,i}, g_{l,i} = \nabla f_{l,i}(u_{l,i}). \tag{3}$$

Here $r_{l,i}$ is the change from block input to output, which we call the *effective residual*, and $g_{l,i}$ is the activation gradient (AG) of the target-task loss with respect to the block output. Taking the input–output difference also accounts for any activation at the end of the block; consequently, $r_{l,i}$ need not equal the raw residual-branch output before addition. Replacing B_l by the identity map changes the input to the downstream network from $u_{l,i}$ to $h_{l,i} = u_{l,i} - r_{l,i}$. Holding the other blocks of the current model fixed, if $f_{l,i}$ is differentiable at $u_{l,i}$ and a local first-order approximation is appropriate, the per-example loss change is

$$\Delta\mathcal{L}_{l,i} = f_{l,i}(h_{l,i}) - f_{l,i}(u_{l,i}) \approx -\langle g_{l,i}, r_{l,i}\rangle . \tag{4}$$

The norm of AG alone does not reflect the size of the deletion-induced perturbation, while the magnitude of the effective residual alone does not capture loss sensitivity along that perturbation. Equation (4) shows that their inner product determines the first-order term in the per-example effect of deletion. Over the scoring set, the actual change in mean cross-entropy for a candidate block is

$$\Delta_p(l; M) = \widehat{R}_p(M[B_l \leftarrow I]) - \widehat{R}_p(M) \approx -\frac{1}{|D_p|} \sum_{(x_i,y_i)\in D_p} \langle g_{l,i}, r_{l,i}\rangle . \tag{5}$$

Positive and negative contributions can cancel in the signed mean on the right. To account for the effect of a single deletion on individual examples, we instead consider the mean absolute per-example loss change:

$$J_p(l; M) = \frac{1}{|D_p|} \sum_{(x_i,y_i)\in D_p} |\mathcal{L}_i(M[B_l \leftarrow I]) - \mathcal{L}_i(M)| . \tag{6}$$

Algorithm 1 Task-ReGRIP: iterative block deletion with recomputed AG×Residual scores

Input: reference model M_0; target-task scoring set D_p; budget K
Output: model sequence $(M_0, \dots, M_K)$; deletion order $(l_1, \dots, l_K)$
Assign each candidate block a fixed unique identifier $\mathrm{id}(l)$
for $k = 1, \dots, K$ **do**
  $\mathcal{C}_k \leftarrow \mathcal{C}(M_{k-1})$
  Traverse D_p with M_{k-1} to obtain $g_{l,i}$ and $r_{l,i}$ for all $l \in \mathcal{C}_k$
  $s_l \leftarrow |D_p|^{-1} \sum_{(x_i,y_i)\in D_p} |\langle g_{l,i}, r_{l,i}\rangle|$ for all $l \in \mathcal{C}_k$
  $l_k \leftarrow \arg\min_{l\in\mathcal{C}_k}(s_l, \mathrm{id}(l))$
  $M_k \leftarrow M_{k-1}[B_{l_k} \leftarrow I]$
end for
return $(M_0, \dots, M_K)$, $(l_1, \dots, l_K)$

Applying the local first-order approximation in Equation (4) to each loss difference in Equation (6) yields gradient–residual sensitivity (GRS) on the target task:

$$\mathrm{GRS}_l(M, D_p) = \frac{1}{|D_p|} \sum_{(x_i,y_i)\in D_p} |\langle g_{l,i}, r_{l,i}\rangle| = \frac{1}{|D_p|} \sum_{(x_i,y_i)\in D_p} \left| \sum_j g_{l,i,j} r_{l,i,j} \right|. \tag{7}$$

The index j ranges over flattened feature coordinates. AG×Residual specifically means *multiplying the activation gradient and effective residual elementwise, summing within each example, taking the absolute value, and then averaging across examples*. It is neither the product of two tensor norms nor the absolute value of an average inner product. When the local first-order approximation is appropriate, GRS approximates the one-step loss-stability objective in Equation (6); it does not directly solve the global structure-selection objective in Equation (2). By the triangle inequality, GRS is at least the absolute value of the first-order term on the right of Equation (5). It is neither an upper bound on the actual loss change nor a measure of its sign. A low GRS therefore indicates a small first-order influence for the block in the current model on D_p, without guaranteeing that its removal will not increase the loss.

3.3 Iterative deletion and rescoring

Starting from M_0, at iteration k we recompute Equation (7) for every remaining eligible block in the current model M_{k-1} and remove the block with the lowest score:

$$l_k = \underset{l\in\mathcal{C}(M_{k-1})}{\arg\min}\ \mathrm{GRS}_l(M_{k-1}, D_p), \qquad M_k = M_{k-1}[B_{l_k} \leftarrow I]. \tag{8}$$

A deletion changes the activations received by downstream blocks and the backpropagated gradients at earlier blocks. The previous GRS ranking may therefore cease to hold even when the parameters of all remaining blocks are unchanged. Algorithm 1 rescores the candidates after every structural change. With $S_k = \{l_1, \dots, l_k\}$, we have $S_{k-1} \subset S_k$, and the kth output model has exactly k blocks removed. Ties are broken by fixed block identifiers.

During scoring, the model is in evaluation mode. For each batch, we record the inputs and outputs of the candidate blocks and backpropagate the *sum* of per-example cross-entropies. For a model that processes examples independently, the resulting gradient at each example's block output corresponds to that example's own loss; summing avoids the additional scaling introduced by a batch mean. A single traversal of D_p scores all remaining candidates simultaneously, so a complete K-step path requires K scoring rounds rather than a separate trial deletion and loss evaluation for every candidate. No recovery training is performed during deletion.

4 Experiments

4.1 Experimental setup

Reference models and deletion budgets. We use ResNet110, ResNet398, ResNet542, and a CIFAR variant of MobileNetV2 trained through network growth on the full CIFAR-100 task as reference models. For the ResNets, only residual blocks with identity shortcuts, unit stride, and matching

Table 1: Architectures and block-deletion budgets of the four reference models

Model	Total blocks	Eligible blocks	Max. deletions K
ResNet110	54	52	18
ResNet398	198	196	66
ResNet542	270	268	90
CIFAR MobileNetV2	17	10	5

input and output shapes are eligible for deletion; for MobileNetV2, only inverted residual blocks satisfying `use_res_connect=True` are eligible. Stage-transition, downsampling, and channel-changing blocks are excluded. The maximum budget K is the floor of one third of the reference model's initial number of blocks (Table 1).

Data split and target tasks. We stratify the $50{,}000$ official CIFAR-100 training images by class into $45{,}000$ source-task training images and $5{,}000$ validation images, using split seed 2027. A fixed checkpoint for each growth-trained reference model is selected on the validation split. The ten-class target task uses the original class indices $\{0, \ldots, 9\}$. Under a fixed partition of the official test images from these classes, 50 images per class form the block-selection set D_p and the other 50 per class form the independent evaluation set D_e, giving 500 images in each set. The source-task training split contains $4{,}500$ images from the target classes; training-set curves are evaluated on these images with deterministic preprocessing. No method uses D_e for block selection, parameter optimization, or checkpoint selection. Cross-entropy and Top-1 predictions use the original 100-class output space, with labels retaining their original indices.

Training-free baselines. Reference denotes the undeleted model at $k = 0$. Task-ReGRIP Direct iteratively selects eligible residual blocks by GRS without updating parameters after deletion. We adapt SLEB-style greedy deletion to visual classification by evaluating each candidate's temporary bypass using whole-network cross-entropy on a fixed set of 128 ten-class training images, permanently removing the candidate with the lowest loss. At each round, this implementation searches only the interior of the eligible candidate sequence, protecting its first and last blocks. Neither method performs recovery training, and both are evaluated on the same D_e.

Baselines with target-task training. Ship of Theseus constructs and trains a compact successor model with the same number of retained blocks per stage as Task-ReGRIP Direct at each deletion budget. Task-ReGRIP + fine-tuning follows the Direct deletion order, retains the surviving reference-model parameters, and optimizes them on the ten-class training images. For each method and budget, its training-set and D_e metrics are computed from the same saved checkpoint.

Training from scratch on ten classes. Scratch (10 class) is independently initialized and trained at each reported deletion budget using only ten-class training images. Checkpoints are selected using target-class images from the source-task validation split. Each Scratch model matches the corresponding SLEB-compressed model in its number of blocks per stage, but its exact block positions need not match those of the Direct path; equal deletion counts therefore do not imply identical architectures. The initial learning rate is 0.1 except for ResNet398 at $k = 10$ and ResNet542 at $k = 15$, where it is 0.01.

4.2 Training-free block deletion

Figure 2 shows the full Task-ReGRIP Direct deletion path and the evaluated SLEB-style budgets, with Reference as a horizontal baseline. We compare the two deletion methods numerically only at shared budgets. Each panel uses its own vertical-axis range and should be read against its labeled ticks.

At the last shared deletion budget, Direct's test accuracy changes relative to the corresponding Reference by -1.2, -0.2, $+1.0$, and 0.0 percentage points for ResNet110, ResNet398, ResNet542, and MobileNetV2, respectively. Its absolute difference from SLEB-style deletion is at most 0.6 percentage points in each case. For ResNet398, the last shared budget is $k = 60$, although the Direct path

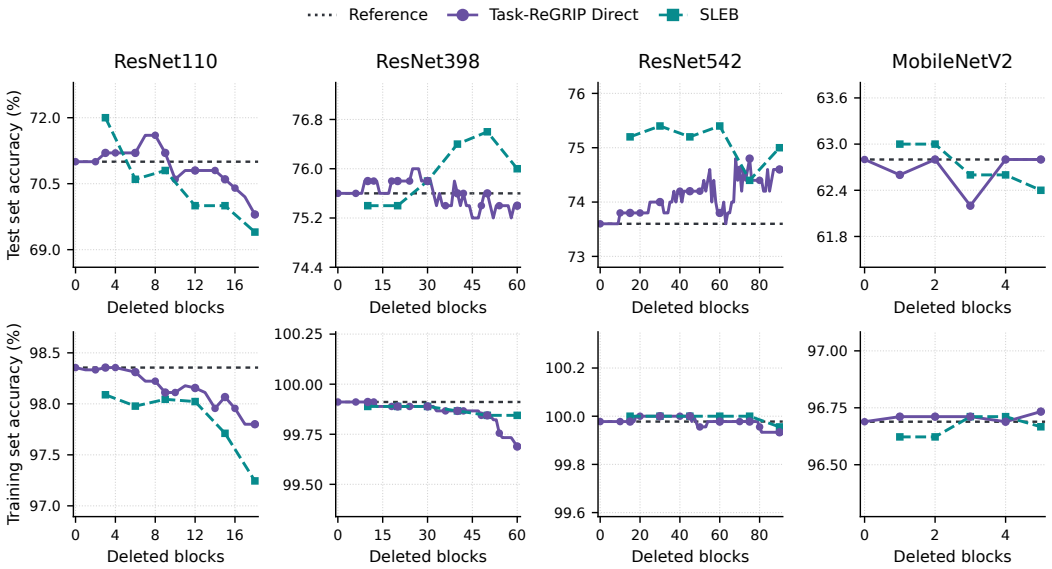


Figure 2: Training-free block deletion on the ten-class target task. Columns show growth-trained ResNet110, ResNet398, ResNet542, and MobileNetV2, respectively; the top row shows the independent test split and the bottom row shows target-class training images.

extends to $k = 66$. Thus, selecting blocks with target-task examples removes multiple blocks across four architectures without updating reference-model parameters while keeping test accuracy close to the undeleted model. Direct and SLEB-style deletion achieve similar accuracy at these shared budgets.

4.3 Comparisons with target-task training

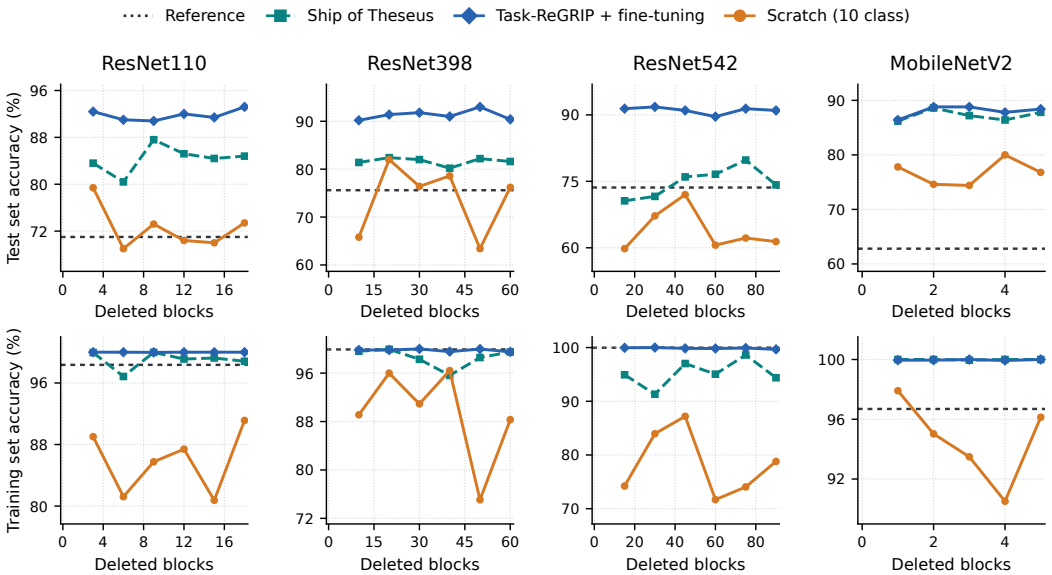


Figure 3: Baselines with target-task training on ten classes. The top row shows the independent test split and the bottom row shows target-class training images.

Figure 3 shows the results of methods with target-task training alongside the training-free Direct path. At the last shared deletion budgets for ResNet110, ResNet398, ResNet542, and MobileNetV2, respectively, Direct obtains test accuracies of 69.8%, 75.4%, 74.6%, and 62.8%. Task-ReGRIP + fine-tuning reaches 93.2%, 90.4%, 91.0%, and 88.4%, while Ship of Theseus reaches 84.8%, 81.6%, 74.2%, and 87.8%. Fine-tuning exceeds Direct for all four models. Ship of Theseus exceeds Direct for three models, but is 0.4 percentage points lower for ResNet542. Both methods optimize parameters on the target task, whereas Direct does not. Scratch provides a from-scratch baseline; because its stagewise block counts follow the SLEB models, an equal k does not imply the same architecture as Direct.

4.4 Effect of target-class count

To examine the effect of target-class count, we construct tasks with $C \in \{10, 30, 50\}$ and label sets $\mathcal{Y}_C = \{0, \ldots, C-1\}$ for each of the four growth-trained reference models, yielding 12 complete paths. The three tasks for a given model start from the same fixed checkpoint and use the data partition described above, but select their deletion orders independently on their respective D_p. No parameters are updated during deletion.

Figures 4 and 5 show the independent-evaluation and training-set curves, respectively, for the four architectures and three target-class counts. Because both the task and evaluation examples change with C, we compare each path with its own $k = 0$ reference. At the maximum deletion budget, the change in test accuracy across all 12 paths ranges from -1.2 to $+1.0$ percentage points. The three ResNets have respectively removed 18 of 54, 66 of 198, and 90 of 270 blocks, while MobileNetV2 has removed 5 of 17. Independently selected paths remain close to their own reference accuracies at each of the three target-class counts studied. The cross-entropy curves also show changes in predictive loss beyond those captured by accuracy.

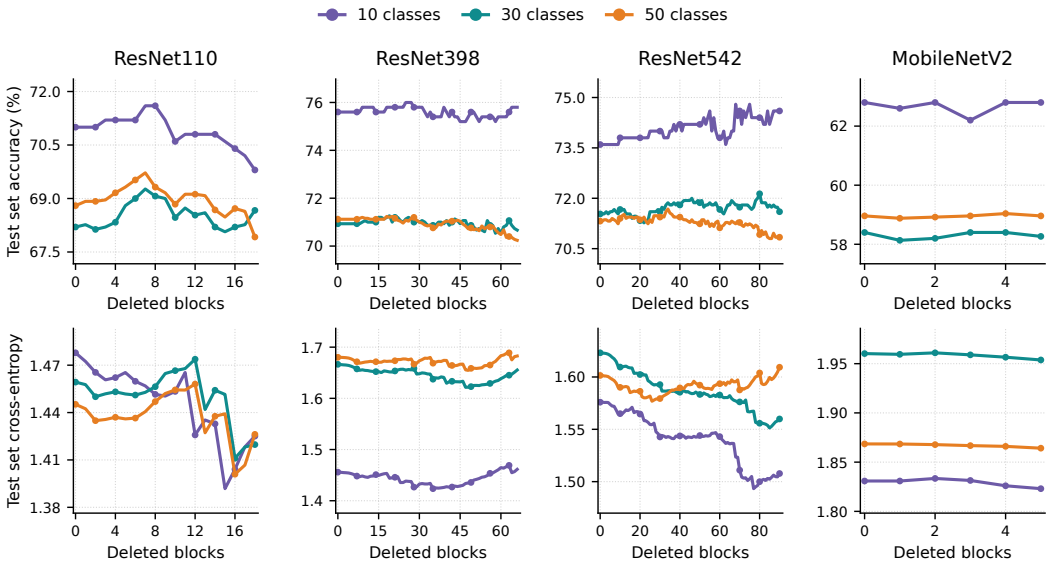


Figure 4: Task-ReGRIP Direct deletion paths on the independent evaluation split across target-class counts.

4.5 Compute cost at selected budgets

Table 2 reports recorded computation through the last deletion budget shared by the two training-free methods for each model. The Direct column sums GRS scoring time, which includes iteration over batches from D_p and the associated forward and backward computation, but excludes evaluation after each deletion. The SLEB-style column sums candidate evaluation and block selection time on 128 training images cached before the timed search; cache construction and post-deletion evaluation

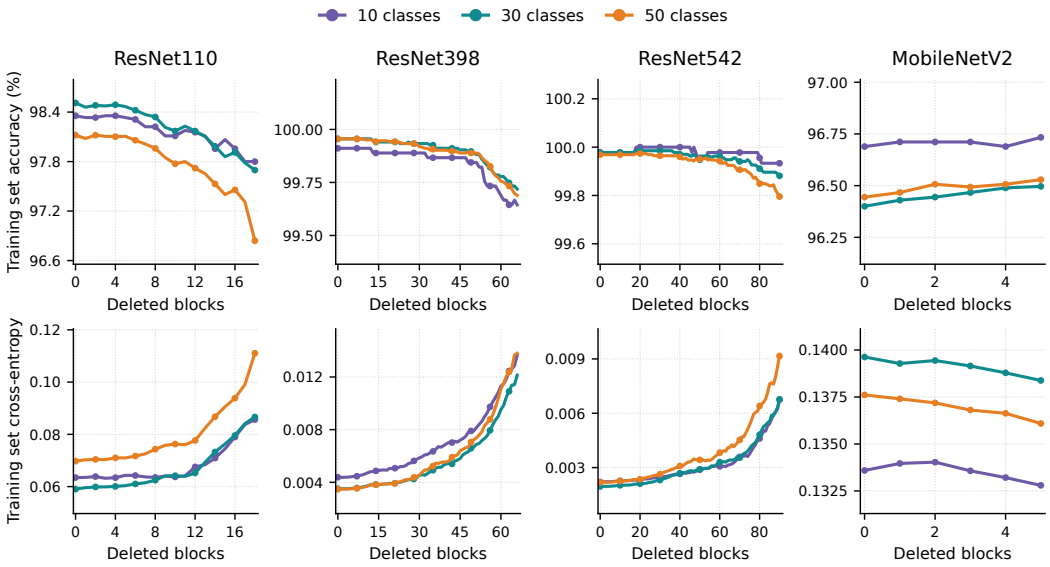


Figure 5: Training-set metrics along the Task-ReGRIP Direct deletion paths across target-class counts.

Table 2: Path-search and scratch-training times at selected budgets on the ten-class target task

Model	Shared k	Direct (s)	SLEB (s)	Scratch (h)
ResNet110	18	5.3	13.3	0.72
ResNet398	60	51.5	502.3	2.67
ResNet542	90	355.1	1311.6	3.56
MobileNetV2	5	1.2	6.6	0.62

are excluded. The Scratch column sums the full training and evaluation time of its independently trained models at the reported budgets. Direct and SLEB-style deletion do not optimize model parameters, whereas Scratch does. Training of the shared growth-trained reference models is not included.

At the last shared deletion budgets, Direct's test accuracy differs from SLEB-style deletion by at most 0.6 percentage points, while its recorded block-selection time is lower for all four models, by factors of approximately 2.5, 9.8, 3.7, and 5.4, respectively. These times reflect the evaluated implementations, which use different calibration sets and candidate-evaluation procedures.

5 Conclusion

Task-ReGRIP reduces the depth of a source-trained classifier for a target-class subset without recovery training. It removes the lowest-GRS eligible block and rescores the remainder after each deletion, preserving the original 100-way head and surviving weights. Across four growth-trained CIFAR-100 models and 10-, 30-, and 50-class targets, the 12 paths changed evaluation accuracy by -1.2 to $+1.0$ percentage points relative to their references at maximum budgets; each ResNet shed one third of its blocks. On ten classes, Direct remained within 0.6 percentage points of SLEB-style deletion at shared budgets while recording lower selection times in the evaluated implementations.

AI use statement

We used generative AI tools to assist with manuscript translation and language editing, plotting-code development, and checks of the manuscript and figures against experimental logs. These tools also provided feedback on the presentation of the methodology and experimental comparisons and

helped summarize and interpret selected results. We take full responsibility for the final content of this paper.

REFERENCES

Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*, 2019.

Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. Depgraph: Towards any structural pruning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16091–16101. IEEE, 2023.

Ariel Gordon, Elad Eban, Ofir Nachum, Bo Chen, Hao Wu, Tien-Ju Yang, and Edward Choi. Morphnet: Fast & simple resource-constrained structure learning of deep networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1586–1595. IEEE, 2018.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015a.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015b.

Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5, 1992.

Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*, 2018.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *2017 IEEE international conference on computer vision (ICCV)*, pp. 1398–1406. IEEE, 2017.

Maedeh Hemmat, Joshua San Miguel, and Azadeh Davoodi. Cap'nn: Class-aware personalized neural network inference. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6. IEEE, 2020.

Maedeh Hemmat, Joshua San Miguel, and Azadeh Davoodi. Cap'nn: A class-aware framework for personalized neural network inference. *ACM Transactions on Embedded Computing Systems*, 21 (5):1–24, 2022.

Mengnan Jiang, Jingcun Wang, Amro Eldebiky, Xunzhao Yin, Cheng Zhuo, Chao Lin, and Grace Li Zhang. Class-aware pruning for efficient neural networks. In *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6. IEEE, 2024.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, 2017.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *2017 IEEE international conference on computer vision (ICCV)*, pp. 2755–2763. Ieee, 2017.

Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *2017 IEEE international conference on computer vision (ICCV)*, pp. 5068–5076. IEEE, 2017.

Ye-Da Ma, Zhi-Chao Zhao, Di Liu, Zhenli He, and Wei Zhou. Ocap: On-device class-aware pruning for personalized edge dnn models. *Journal of Systems Architecture*, 142:102956, 2023.

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20192–20204, 2025.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 11256–11264. IEEE, 2019.

Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. Sleb: Streamlining llms through redundancy verification and elimination of transformer blocks. *arXiv preprint arXiv:2402.09025*, 2024.

Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in neural information processing systems*, 33:6377–6389, 2020.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*, 2020.

Yanting Wang, Feng Li, and Han Zhang. Ta2p: Task-aware adaptive pruning method for image classification on edge devices. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2580–2584. IEEE, 2024a.

Yanting Wang, Feng Li, Han Zhang, and Bojie Shi. Task-level customized pruning for image classification on edge devices. *Electronics*, 13(20):4029, 2024b.

Yanting Wang, Bojie Shi, and Han Zhang. Tsp: Task-specific pruning for personalized image classification on edge devices. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2025.

Yanting Wang, Bojie Shi, Han Zhang, Zhipu Xie, and Jinchao Huang. Model pruning method for differentiated image classification tasks on edge devices. *Journal of Systems Architecture*, pp. 103782, 2026.

Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. Bert-of-theseus: Compressing bert by progressive module replacing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7859–7869, 2020.

Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 9194–9203. Ieee, 2018.