# AdaptaNet: Towards Modular and Dynamic Slimmable Neural Networks for Adaptive Edge Computing

Ian Harshbarger
*Computer Science Dept.*
*University of California Irvine, USA*
*iharshba@uci.edu*

Marco Levorato
*Computer Science Dept.*
*University of California Irvine, USA*
*levorato@uci.edu*

***Abstract*—Resource-efficient deployment of deep neural networks often pits accuracy against computing effort and latency, especially when a single model must operate across system parameters at run time. In this context, task offloading becomes instrumental to achieve acceptable performance tradeoffs. However, even edge servers have limited computing capacity, and it is then critical to design neural models whose complexity can gracefully scale in response to input characteristics and load. This paper introduces *AdaptaNet*, a modular slimmable methodology that unifies width and depth adaptivity on a shared backbone. The approach centers on *anchor layers* that align intermediate *student* features to a fixed *teacher*, enabling slot-stable interfaces; *depth-slimmed ($\delta$) substitutes* that reduce serial depth without re-wiring; and lightweight, *decoupled Var Gates* that read anchor features to emit confidence signals for cost-aware dynamic control. Experiments show: (i) accuracy at large widths comparable to baselines while supporting a larger subnetwork set; (ii) under runtime-random selection, stable means/variances that exceed AdaptaNet's slimmest fixed-width entries; (iii) device-agnostic latency gains from $\delta$ substitutes consistent with depth-reduction principles; and (iv) exploitable confidence/accuracy trends from Var Gates, with simple binning policies matching uniform mixtures at similar computing load. These results demonstrate a practical bridge from slimmable capacity control to modular, deployment-oriented design with interchangeable width/depth components as insertable modules.***

## I. Introduction

The efficient deployment of deep neural networks (DNNs) in resource-constrained edge devices such as mobile devices, unmanned aerial vehicles, and robotic platforms is increasingly challenging as neural models continue to scale in complexity and computational demands - directly impacting battery life, thermal management, and overall system reliability. In this context, task offloading, where compute-capable servers positioned at the network edge take over the execution of the neural models, is instrumental in achieving acceptable performance. However, excessive load on edge servers may degrade their ability to serve their workload within application parameters.

To mitigate this issue, the research community has explored several avenues. While specialized accelerators and systems-on-chip designs continue to reduce inference cost [4], architectural and algorithmic strategies - such as quantization, pruning, and low-rank compression [9, 15], as well as Neural Architecture Search (NAS) [7, 35, 32, 5] - contribute greatly to the efficient deployment of DNNs.

One of the main limitations of the above techniques is that the neural models are designed to have a fixed architecture, so that the computing effort needs to match the worst-case sample complexity. Dynamic neural networks, including early-exit [33], slimmable [16] and multi-branched architectures [27], offer adaptation mechanisms to tune complexity to sample distributions or even individual samples. At the core of our design, slimmable neural networks (SNNs) train a single shared-weight backbone that exposes multiple width/depth instantiations at inference time [37, 16]. SNNs are therefore not inherently a NAS problem, until a *dynamic* configuration policy is introduced to select a subnetwork per input or budget, at which point search/selection becomes part of the deployment pipeline. Modular neural networks (MNNs) offer a complementary axis by structuring models into reusable components with explicit interfaces [3, 1, 27, 11], and recent incubation-style modularization suggests broader generality beyond narrow endpoints [29].

In this paper, we introduce *AdaptaNet*: a modular, slimmable approach that unifies width and depth control behind stable stage interfaces. *The design we propose in this paper bridges SNN capacity control with such modularity by treating stage boundaries as teacher-aligned anchors* inserting independent depth slimmed modules to reduce serial depth, and adding decoupled variance gates that emit confidence signals for cost-aware configuration. Concretely, AdaptaNet is *adaptive* as it exposes multiple valid execution paths behind fixed stage interfaces: computational complexity (width) can be varied across stages, latency (depth) can be reduced via slot-compatible $\delta$ substitutes, and selection can be driven by lightweight confidence signals emitted from Var Gates. This enables the same trained backbone to adjust computing effort in response to time-varying load and per-sample difficulty, without requiring re-wiring or re-training the full model when swapping or tuning auxiliary modules. This framing treats efficient deployment as a multi-faceted, architecture-first problem that remains compatible with, yet not dependent on, hardware specialization and makes execution adaptive to load and sample-level characteristics. Our contributions include:

This paper has been partially supported by the US National Science Foundation under grant CCF 2140154.

**(1) AdaptaNet**, A new MNN methodology emphasizing inter-module feature alignment allowing seamless integration of, conceptually, numerous auxiliary components such as our demonstrated *depth-slimmed modules*
**(2) Training study.** A comparative analysis of normalization and subnetwork sampling across USNet/DSNet and AdaptaNet, linking stability/variance patterns to shared statistics and depth control; results are reported jointly with latency and energy observations.
**(3) Var Gates.** Decoupled, uncertainty-aware gates that read anchor features to emit confidence scores for cost-aware selection; we document dataset-specific confidence–accuracy trends and show simple heuristics match uniform mixtures at similar computing effort.

AdaptaNet matches baseline accuracy at larger widths while supporting a broader set of subnetworks, and it remains stable under runtime-random configuration selection (mean and variance exceeding the slimmest fixed-width setting). Moreover, the slot-compatible $\delta$ depth substitutes deliver consistent, device-agnostic latency reductions Our approach fundamentally advances the flexibility and efficiency of modular and slimmable neural network designs, facilitating deployment in time varying and dynamic resource-constrained scenarios.

## II. Related Work

**Slimmable Neural Networks** are depth/width–adaptive networks that trade accuracy for computing effort at run time. [17] surveys dynamic architectures and gating families as a unifying lens for adaptivity. Early SNN models train a single backbone to operate at multiple widths via shared parameters [39, 38]. From these two techniques remain foundational: (i) *switchable batch normalization* to stabilize statistics across widths, and (ii) the *sandwich rule* with in-place knowledge distillation (KD) [14] from the largest subnetwork to smaller ones. Beyond these, recent variants explore structural sparsity and deployability, including *Slimmable Pruned Networks* [23] and parallel/distributable training and inference with *PARADIS* [30], as well as application-driven designs [18]. Compound scaling baselines (e.g., EfficientNet) provide a non-adaptive point of comparison [31].
**Dynamic Neural Network and Dynamic Slimmable variants** extend fixed width and functional parameter sets with input-conditioned selection via lightweight gates [26, 25], improving adaptivity but tying decision logic closely to the backbone. Orthogonally, systems that combine slimming with early exits co-optimize depth/width and latency via auxiliary heads and simple controllers [24, 18]. Related dynamic-routing and early-exit approaches such as SkipNet [34] and ShallowDeep [19] likewise reduce compute by conditional execution. Confidence-aware or uncertainty-guided criteria can further stabilize decisions [13, 36], though most methods supervise only final logits rather than internal features. Our approach is similar in goals but differs in mechanism: we supervise *intermediate* features at anchor layers and keep selection modules structurally decoupled from the backbone. This decoupling is key to *adaptivity*: it allows the runtime selection logic to evolve (e.g., new heuristics, different confidence calibrations, or system-driven policies) without re-training or re-wiring the core backbone. In contrast to gates that entangle routing with backbone internals, AdaptaNet treats selection as a swappable deployment component operating on a stable interface.
**Modular Neural Networks** pursue compositionality and reuse through explicit submodules or routes, from early NMNs [3] to modern modular/meta-learning and hierarchical formulations [2, 1, 10, 20, 28, 8]. These works emphasize modular *endpoints*, while SNNs emphasize modular *capacity* within a shared backbone. Closest to our core idea is [29], which progressively trains blocks while aligning *intermediate* representations using a strong teacher. In essence, our anchor-layer supervision instantiates an incubation-style alignment of internal features, develops additional solutions to enable multi-width/multi-depth regimes, and supports additional task modules such as depth-slimmed modules and decoupled gates for per-sample selection.

## III. AdaptaNet Methodology

### A. Slimming Definitions

Any DNN $M$ can be decomposed into learnable layers $l$ and static operations $o$. In this work we focus on $l$ and define model complexity in terms of two metrics: multiply-accumulate operations (MAdds) and structural layer complexity. We explicitly define the two complementary slimming forms to control these complexity terms as *Width Slimming* and *Depth Slimming*. We clarify the use of notation for these going forward: In Width Slimming, for a given $s$ subset of layers with the parameter sets $p$ as $l_s^p \subseteq M$, a subset $l_s^{p'} \subseteq l_s^p$ is selected to reduce computation. This subset still includes all existing layers but with reduced parameters sets of those layers, i.e., $p' \subseteq p$. In contrast, in depth slimming, entire layers or sequences from an $m$-th positional layer to an $n$-th as $l_{m:n} \subseteq M$ are bypassed or replaced. However, some composite blocks are optimized for dense execution, making naive removal counterproductive without extensive work and/or re-optimization as demonstrated in [34]. In such cases, depth slimming uses reduced versions of $l_{m:n}$ that maintain compatibility while lowering complexity in favor of attempting to bypass existing layers.

### B. Anchor Layers

To address the training complexity of slimmable models and to introduce modularity, we propose the concept of *anchor layers*. Let $M$ be a network with configurable width $\rho$ and depth $\delta$. Training all sub-networks $m \in M$ across the full range of $\rho$ and $\delta$ values is computationally expensive. Prior works restrict subnetwork sets $M_m \subseteq \{m_0, \ldots, m_n\}$ to uniformly slimmed variants, typically bounded below by a minimum width. However, when focusing training on uniform slimming configurations, various other non-uniform configurations of the network remain undertrained or underperforming without extensive supervision as seen in Table II. This is partially due to the combinatorial explosion of network configurations when non-uniform configurations are present. We mitigate this by drawing from DSNets and partitioning $M$ into a sequence of

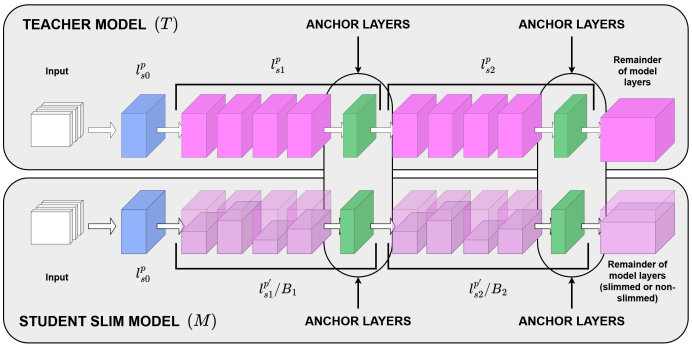


Fig. 1. Visualization of our AdaptaNet method with anchor layer implementation.

*blocks* $B$, where each block comprises a distinct set of layers. Each layer of the overall subset $s$ of layers mentioned above is uniquely assigned to a block $B_n$, such that $B_n$ spans the set of $l^p_s$ layer parameters. These blocks are structured to allow consistent width slimming, with blockwise depth slimmed variants being constructed as needed. Within each block $B$, we designate, at least, the final layer as an our proposed *anchor layer* whose output has a fixed dimensionality. This output is aligned with a corresponding intermediate output in a pretrained teacher model $T$ (i.e., the base non-slimmed network a slimmable network derives from). The concept of the anchor layer implementation is illustrated in Fig. 1.

The Training of the blocks proceeds sequentially by individual $B_n$, aligning the output of each anchor layer to its counterpart in the teacher network using a knowledge distillation objective. We refer to this blockwise training as the *Cascade Method*, which facilitates modular learning while leveraging the representational power of $T$ and grounding all subnetworks in a shared feature space. This echos Deep Incubation Models [29], but altered to facilitate multi-subnetwork training: we treat anchors as fixed interfaces and additionally require them to reconstruct downstream teacher representations so that alignment remains stable across many width–depth combinations without re-entangling later student blocks. This yields an anchor-centered loss: for each $B_n$ with width–depth configurations $(\rho, \delta)$, we compare the slimmable student output $\hat{y}_n$ with the teacher output $y_n$, primarily via mean squared error (MSE), and define the primary loss for each anchor layer as:

$$L_n(\hat{y}_n, y_n) = \text{MSE}(\hat{y}_n, y_n) + \sum_{i=n+1}^{N} \text{MSE}(\hat{y}'_i, y'_i). \quad (1)$$

To acquire $\hat{y}'_i$ for $i > n$, we forward the target anchors output through the corresponding teacher blocks:

$$\text{s.t.} \quad \hat{y}'_i = \begin{cases} T_{B_i}(\hat{y}_n) & \text{if } i = n+1 \\ T_{B_i}(\hat{y}'_{i-1}) & \text{otherwise.} \end{cases} \quad (2)$$

This formulation encourages the intermediate student features not only to match the teacher at a specific anchor, but also to reconstruct downstream teacher representations, preserving semantic consistency across the anchor training process.

To stabilize performance across the diverse configurations we introduce the *Multi-Dimensional Sandwich Rule* (MDSR), extending the sandwich rule proposed in [38]. For each optimization step when training an anchor layer or full module output, we compute the loss for the largest possible subnetwork, the losses for the largest and smallest subnetworks for each dimensionality of additional modules and/or depth slimming factors, and the loss for at least one fully randomized subnetwork configuration. The number of sampled subnetworks under the MDSR is at least $2^{|D|} + 1$ configurations, where $D$ is the set of configurable variables/dimensions to a single anchor layer or output (e.g., $\rho$ and $\delta$). The final per-block loss becomes Eq. (3), where we denote the largest subnetwork output of dimensionality with the most parameters with $\hat{y}_{nL}$ and any sampled networks of the selected configurations set $\eta$ as $\hat{y}_{ni}$. The $\upsilon$ term then balances supervision between the teacher and the largest subnetwork. We note that the largest and smallest subnetwork outputs for each dimension must be present in $\eta$, excluding $\hat{y}_{nL}$.

$$\overline{L}_n(\eta) = \frac{\sum_{i \in \eta} \left[ \upsilon \, L_n\big(\hat{y}_{ni}, \, y_{ni}\big) + (1 - \upsilon) \, L_n\big(\hat{y}_{ni}, \, \hat{y}_{nL}\big) \right].}{|\eta| - 1}$$

$$L_B(\hat{y}_n, y_n) = L_n\big(\hat{y}_{nL}, \, y_n\big) + \overline{L}_n(\eta). \quad (3)$$

### C. Var Gates

Existing gating structures, such as those in [26], enable effective dynamic subnetwork selection, but incur considerable training overhead and are tightly coupled to the model. In AdaptaNet, anchor supervision decouples gating from backbone features: selection modules read stable anchor representations and can be replaced or tuned without re-training.

As demonstrated in our results (see Table II), the training of an AdaptaNet model produces a gradient of performance based on slimming (Madd) complexity. From this we assess Anchor Layer quality using compact, dual-headed early-exit gates inspired by the works of [13, 36]. Each gate contains a detection head approximating the output of the largest subnetwork. However, early exits suffer from inherently lower accuracy yields, especially in earlier layers. We then introduce a secondary confidence head, the variance head, that predicts output variance for each detection element with respect to a ground detection as a mean. Specifically we model two distributions: a pseudo-Dirac ground truth $G(\mu_{gt}, \lambda)$ centered on the largest subnetworks, $m_L$'s, output with narrow variance $\lambda \approx 0.0001$, and a predicted normal $P(\mu_p, \sigma_p)$ using the predicted detection and variance head outputs. With the detection head optionally frozen if pretrained, we minimize their KL divergence $D_{KL}(P||G)$ enabling the model to estimate output uncertainty as guide value for heuristic based dynamic slimming.

Given the above KL Divergence loss for training, we implement the full loss of the Var Gates as in Eq. (4). Here we train over both heads of gate with the $L_{KD}$ loss being any

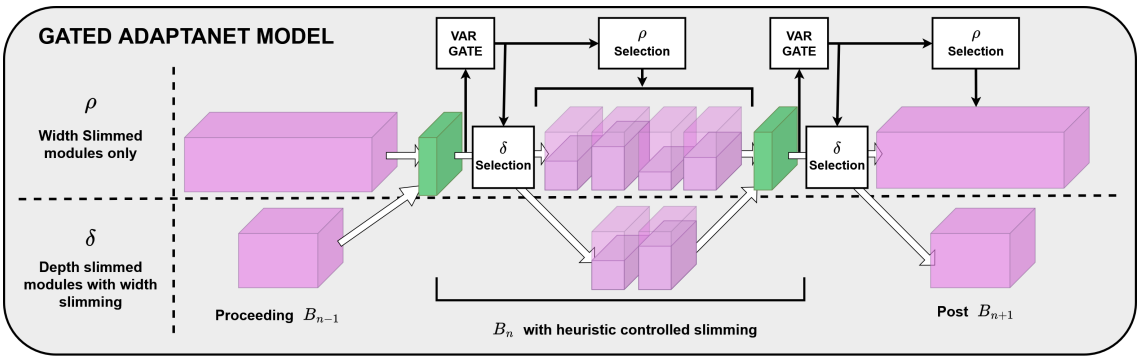


Fig. 2. Visualization of the AdaptaNet's gated sub-network selection process of a $\rho$ and $\delta$ heuristic selection from Var Gate outputs.

method of knowledge distillation sufficient for a particular task on the detection head. In this work we focus on a classification task and utilize the soft-cross entropy loss:

$$L_{VG} = |D_{KL}(P||G)| + L_{KD}. \tag{4}$$

## IV. Implementation and Training

We now detail the specific implementation of the concepts presented in the previous section. Our results are structured evaluating the AdaptaNet methodology as a bridge of *slimmable width/depth* ideas with *modular neural networks* designs. All comparisons are made against USNet and DSNet, the canonical baselines for slimmable training. Newer variants (e.g., DSNet++, PARADIS, NaviSlim) add auxiliary losses and architecture-specific couplings.

### A. Adapta-Res-Net Framework

**Base Network:** We instantiate AdaptaNet on a ResNet-50 backbone [12]. The four-stage layout (3/4/6/3 bottlenecks) provides stable anchor placements and consistent width-slimming granularity across stages chosen to be $\rho \in \{1.0, 0.75, 0.5, 0.25\}$. All models are trained and evaluated on ImageNet [6] and CIFAR-100 [22] with input sizes $224\times224$ and $32\times32$, respectively.

**Depth Slimming:** Resnet-50 bottleneck blocks apply a $1\times1$, $3\times3$, $1\times1$ and kernel convolutional transforms with channel reduction and expansion. Plus a residual $1\times1$ path when down-sampling resolution. Width slimming for this condensed layer module is implemented directly by adjusting the channel weight parameters of blocks. To implement depth slimming, instead of removing layers, we introduce our *depth-slimmed modules* as plug-in surrogates that expose the same input/output interface at the anchor boundary. Concretely, each module retains only the central $3\times3$ kernel convolutional layer transform, omitting the first/third $1\times1$ and the residual branch. This makes the substitute *slot-compatible* with the original bottleneck block, enabling per-stage interchangeability. Either the full bottleneck or its depth slimmed module can be selected at run time while, the last bottleneck block layer of each stage remains intact and serves as the stage anchor for teacher alignment. This allows width *and* depth to vary against fixed anchor points. The practical implication is that adaptivity is achieved through *interchangeability*: at inference time, the system can select either the full bottleneck path or the $\delta$ substitute per stage, and can do so independently of the width choice. Because the anchor boundary is preserved, these swaps do not require re-optimization of downstream stages, which keeps configuration changes lightweight and deployment-friendly.

**Var Gates:** To support runtime adaptivity, we attach Var Gates after each stage anchor (excluding the final one). Each gate comprises two $2\times2$ kernel convolutions with BN and ReLU (stride 1 or 2 depending on target feature resolution), followed by average pooling to standardize the feature map. For ImageNet the gate emits $[\mathrm{B}\times32\times7\times7]$; for CIFAR-100, $[\mathrm{B}\times32\times4\times4]$. Flattened features feed to the two parallel heads mentioned in Sec.V-C. Gates read anchor features rather than internal backbone states, so selection remains a modularized component. This keeps the interface comparable to DSNet-style gating while avoiding tight coupling between gate parameters and the backbone.

### B. USNets and DSNets

For the sake of fair comparison, it is necessary to re-implement the USNet and DSNet models with slight alterations. Moreover, the official per-subnetwork metrics and weights were unavailable at time of writing, except for the recorded uniform slimmings configurations of USNets [38]. Our re-implementations follow each paper's original hyper-parameters and training protocols (USNet sandwich rule; DSNet parallel EMA model), with slimming factors $\rho \in \{1.0, 0.75, 0.5, 0.25\}$. We also train a variant that samples any three subnetworks between and including the largest and slimmest configurations at each training step (instead of fixed uniform picks). DSNet's gating heads are omitted to avoid overhead.

For the variants used in Sec. V-A, we evaluate USNet with BN (without post-training BN calibration) and with GN. In Sec. V-B we use USNet with BN and re-enable post-training BN calibration. For DSNet, we report a BN variant in Sec. V-A and the original GN variant used elsewhere. Final USNet uniform-width accuracies in our runs are within $\approx 1\%$ of the reported numbers; for DSNet, the lack of a

reported supernet baseline prevents a precise comparison, but our training mirrors the published method.

## V. Experiments and Results

We proceed in this section with three blocks of evaluation. First, we validate the practical runtime levers inferenced by prior works [24, 16] in their application to our methodology. Namely, that *depth reduction* yields consistent latency gains, while we contribute with clarifying normalization choices that affect shared-width stability (Sec.V-A). Second, using this setup, we evaluate AdaptaNet's accuracy trends against recreated USNet and DSNet models under uniform, mixed, and runtime-random configuration schemes (Sec.V-B). Third, we assess the *Var Gates* as *modular, decoupled*, confidence probes to inform cost-aware selection without coupling the gates to the backbone structure (Sec.V-C). Experiments are performed on a Linux workstation with a NVIDIA RTX 4090; we additionally report latency on an NVIDIA Orin Nano to illustrate hardware sensitivity.

### A. An Analysis on SNN Architecture

A central, but lightly explored, topic for slimmable training is which supervision and normalization choices preserve *all* subnetworks, not just those seen most often. USNets and DSNets provide strong methods, but to our knowledge the fine-grained behavior across many subnetwork/sample schemes remains sparsely explored in favor of fixed and/or learnable model configurations. We round out this topic here for later experimental protocols with controlled ResNet-50 variants (Sec.IV-B).

In width-shared training, activations presented to a layer come from a *mixture* of widths. With BN, running means/variances integrate statistics across this mixture; rarely sampled widths inherit biased normalization at test time, shifting feature scales and gradients on those paths. Switchable BN [38] tracks per-width statistics, yet the optimizer still updates *shared* weights under heterogeneous feature scales, amplifying interference between widths. GN removes batch dependence and is less sensitive to sampling frequency, but its per-group rescaling slightly alters the channel-wise inductive bias. LayerNorm (LN) is memory-bound on devices, yielding high latency (see Table I). Empirically, skipping BN re-normalization exposes a lower bound for under-supervised widths; the resulting accuracy drift is most apparent with BN under non-uniform sampling Fig.3.

Normalization choice alone yields a gradient of latency changes on hardware devices (BN $<$ GN $<$ LN in Table I). In contrast, *reducing depth*, even with simple interface-compatible substitutes, cuts end-to-end time substantially: AdaptaNet-$\delta$ (GN) improves 4.38$\rightarrow$2.38 ms on the NVIDIA 4090RTX and 17.21$\rightarrow$12.71 ms on the NVIDIA Orin. This mirrors prior observations [24]: when kernels are already highly optimized, width changes often leave the number of sequential stages and memory traffic largely intact, whereas depth slimming shortens the serial path and removes residual merges, producing the most consistent, device-agnostic speedups.

These characterizations fix our evaluation protocol primarily to GN with AdaptaNet Models, and frame the later results in latency/energy and accuracy trade-offs around the depth slimming modular substitutes.

### B. Adapta-SuperNet

*1) Model Setup and Training:* For the AdaptaNet model, Anchors are supervised by two dataset specific teachers to check sensitivity to target quality: an ImageNet-pretrained ResNet-50 (80.85% top-1) and an in-house teacher trained on CIFAR-100 (78.56% top-1). Preprocessing is normalization only with training batch sizes being 64 (CIFAR-100) and 100 (ImageNet) for both teacher and AdaptaNet Models.

We train AdaptaNet using the Cascade method with MDSR (three random subnetworks per step when numerically feasible) and the anchor loss in Eq. (1). On CIFAR-100, we use Adam optimization [21] with learning rate $\mathrm{lr}{=}10^{-3}$ and no weight decay, training each stage until loss convergence, which in practice corresponds to 3/4/5/6 epochs for the stem+$B_1$, $B_2$, $B_3$, and $B_4$+classifier modules, respectively, followed by a 7-epoch *locking-in* phase at $\mathrm{lr}{=}10^{-4}$. We then insert the modular depth-slimmed substitutes, freeze all layers except the depth-slimmed modules, and run a second cascade+locking-in schedule that updates only depth-slimmed weights with the same optimizer and epoch counts. On ImageNet, we apply the same procedure with stage lengths of 3/5/7/8 epochs and an 8-epoch lock-in, again mirrored for the depth-slimmed modules. In the ImageNet+GN setting, MSE-only anchors tend to converge prematurely to weaker feature reconstructions, so at the last epoch of each cascaded phase we augment the objective with a soft cross-entropy term (as in [39]), averaged with the MDSR loss across sampled subnetworks (excluding $m_L$). Because ImageNet with mixed widths and practical batch sizes accentuates normalization drift for rarely sampled configurations (Sec. V-A), we additionally report an AdaptaNet-LN variant as a stability-oriented reference, which remains feasible despite LN's added latency due to AdaptaNet's comparatively shorter training schedule.

*2) Result Evaluations:* Table II reports the task performance of the AdaptaNet method against the foundational slimmable baselines (USNet and DSNet), elaborated in Sec.IV-B. Explicitly they are expressed in: (i) *uniform* widths ($\times1.0, \times0.75, \times0.5, \times0.25$), (ii) *mixed* 4 stage-wise permutations geared toward unconventional and sporadic slimming configurations ($\{\times0.25, \times1.0, \times1.0, \times0.25\}$, etc.), and (iii) a *runtime-random* protocol: for each validation sample, draw one configuration uniformly at random and average accuracy over 10 passes of the validation set. At the end of each full pass of the validation set we record the accuracy over all validation samples. After the 10 passes we generate the mean, and one standard deviation, of the 10 collected accuracy values as the metric to inform *supernet stability* without a learned policy.

Across *uniform widths*, AdaptaNet-GN achieves similar performance to the USNets and DSNets at larger slimmings,

| | Exec. Time (ms) | | Power Draw (W) | | Energy Consumption (mJ \| W×ms) | |
|---|---|---|---|---|---|---|
| Norm. Layer | Nvidia 4090RTX | Jetson Orin Nano | Nvidia 4090RTX | Jetson Orin Nano | Nvidia 4090RTX | Jetson Orin Nano |
| BatchNorm | 4.77 | 18.78 | 72.47 | 15.00 | 345.68 | 281.70 |
| LayerNorm | 22.22 | 91.57 | 83.52 | 15.00 | 1855.81 | 1373.55 |
| GroupNorm | 5.85 | 21.54 | 74.21 | 15.00 | 434.12 | 323.10 |
| GroupNorm (AdaptaNet Teacher) | 4.38 | 17.21 | 76.42 | 15.00 | 334.72 | 258.15 |
| GroupNorm (AdaptaNet $\delta$) | 2.38 | 12.71 | 72.16 | 15.00 | 171.74 | 190.65 |

TABLE I

AVERAGE EXECUTION TIMES, INSTANTANEOUS POWER DRAW, AND ENERGY CONSUMPTION OF RESNET-50 MODELS WITH NORMALIZATION LAYERS (IMAGENET). THE "ADAPTANET TEACHER" ROW IS THE FULL (NON-SLIMMED) MODEL USED FOR ANCHOR SUPERVISION. "ADAPTANET $\delta$" APPLIES THE *modular depth-slimmed modules*

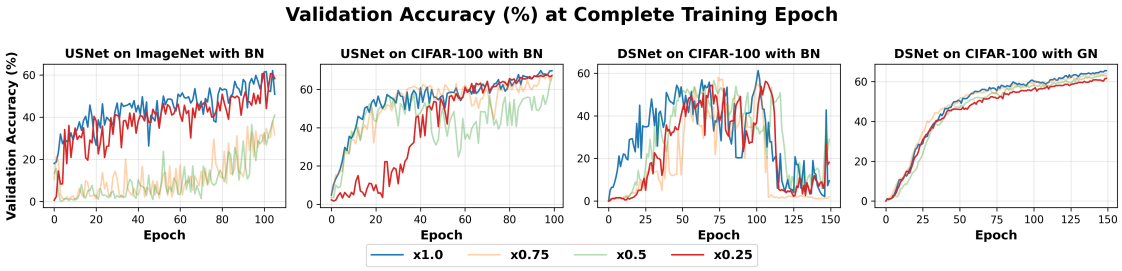


Fig. 3. Validation accuracy trajectories with 3 randomly sampled subnetworks (of 256) per step.

but operates at a reduced accuracy (slimmer configurations) on both datasets. The depth-slimmed $\delta$ variants follow the same shape with uniformly lower accuracy while the ImageNet LN variant exhibits a narrower spread across widths at a cost to maxium attainable accuracy of the GN variant.

Under *mixed stage-wise schedules*, AdaptaNet-GN remains close to DSNet and typically above USNet, indicating that anchor-aligned intermediate representations remain usable when widths vary within the network. The ImageNet LN variant exhibits a narrower spread across widths with slightly lower top-width accuracy, consistent with batch-agnostic normalization effects.

With a *runtime-random* policy, AdaptaNet (GN/LN) sits above its own $\times 0.25$ fixed-width entries while keeping deviations modest. USNet* shows higher variance (especially on ImageNet), whereas DSNet* is steadier but tends to operate at a reduced performance to AdaptaNet.

Taken together, the measurements indicate stable accuracy when capacity is chosen on-the-fly, with $\delta$ modules preserving the stability pattern while exhibiting the expected accuracy reductions alongside the measured latency gains shown Table I.

### C. AdaptaNet Var Gates

In this section, we assess the implementation of the modular Var Gates as confidence scoring outputs uncoupled from dependency on the entire network structure - that is, they do not, by themselves, implement a full dynamic routing policy. After the *locking-in* phase of the AdaptaNet models from the previous subsection, we train each Var Gate individually for the GN variants of the AdaptaNets. During the Var Gate training all other module weights are frozen, each gate receives the corresponding anchor-layer outputs as input, and is trained for one epoch using the loss in Eq. (4) with Adam (learning rate $10^{-4}$, no weight decay). In practice, this loss converges quickly, with earlier-stage gates experiencing more samples before convergence. While the Var Gates themselves only produce uncertainty scores, to examine the reliability of these scores we design two simple selection heuristics based on their observed characteristics. Specifically, we sample 10% of the training data, evaluate each input under all subnetwork configurations, record prediction correctness, and for each configuration compute the mean of the top-10 confidence scores aligned with the top-10 detection outputs. We then bin these confidence values into eight intervals for and per each width–depth configuration. The resulting average accuracy per confidence bin is shown in Fig.4.

We notice from our results in Fig.4 Two consistent phenomena appear. First, On CIFAR-100, higher confidence associates with higher accuracy across gates and widths while on ImageNet, the trend inverts (higher confidence aligns with lower accuracy). We do not claim a definitive cause; in our setup, the direction seems *dataset-specific* and plausibly reflects confidence calibration under different label granularity and distributional complexity. As we do not learn normalized mean and variance scores opting to train with the models produced real values. Secondly, Moving from Gate 1 to Gate 4, curves become smoother and more separated. This is expected as deeper anchors expose features that are both more linearly separable and lower dimensional, making confidence more regular even when the monotonic direction differs across datasets.

Guided by these distributions, we test two simple heuristic policies: *pull-up* (map lower-confidence bins to the empirically higher-accuracy subnetworks for that gate) and *pull-down*

| Slimming | CIFAR-100 | | | | | |
|---|---|---|---|---|---|---|
| | Ours-(GN) | Ours-GN($\delta$) | USNet | DSNet | USNet* | DSNet* |
| Uniform-x1.0 | 72.2 | 66.9 | 71.9 | 72.8 | 69.8 | 66.4 |
| Uniform-x0.75 | 70.75 | 65.9 | 71.6 | 72.5 | 67.6 | 65.3 |
| Uniform-x0.5 | 67.8 | 62.1 | 70.63 | 72.0 | 66.3 | 64.4 |
| Uniform-x0.25 | 61.8 | 55.21 | 68.6 | 70.5 | 67.2 | 62.6 |
| {x0.25, x1.0, x1.0, x0.25} | 64.2 | 58.8 | 66.7 | 68.3 | 63.0 | 62.5 |
| {x1.0, x0.25, x0.25, x1.0} | 64.0 | 58.3 | 66.9 | 68.8 | 63.6 | 63.6 |
| {x0.25, x1.0, x0.25, x1.0} | 63.9 | 59.5 | 65.6 | 59.3 | 64.3 | 62.4 |
| {x1.0, x0.25, x1.0, x0.25} | 64.1 | 59.6 | 65.9 | 63.5 | 64.8 | 64.2 |
| Runtime-random (mean±std) | **67.9** ± 1.8 | 61.2 ± 2.7 | 66.6 ± 2.6 | 67.5 ± 4.8 | 65.8 ± 2.3 | 63.9 ± **1.4** |

| Slimming | ImageNet | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Ours-GN | Ours-GN($\delta$) | Ours-LN | Ours-LN($\delta$) | USNet | DSNet | USNet* | DSNet* |
| Uniform-x1.0 | 76.2 | 61.4 | 71.3 | 69.9 | 75.6 | 73.6 | 62.0 | 70.1 |
| Uniform-x0.75 | 74.1 | 58.2 | 69.2 | 65.3 | 74.0 | 71.5 | 38.1 | 69.3 |
| Uniform-x0.5 | 64.1 | 53.8 | 66.6 | 60.4 | 71.2 | 68.3 | 37.9 | 65.1 |
| Uniform-x0.25 | 57.4 | 48.7 | 63.7 | 55.9 | 64.2 | 66.5 | 60.6 | 63.9 |
| {x0.25, x1.0, x1.0, x0.25} | 62.2 | 52.6 | 65.1 | 58.9 | 56.6 | 62.8 | 33.9 | 61.3 |
| {x1.0, x0.25, x0.25, x1.0} | 61.3 | 51.2 | 64.8 | 59.1 | 50.2 | 61.1 | 49.1 | 64.1 |
| {x0.25, x1.0, x0.25, x1.0} | 61.6 | 51.6 | 64.9 | 58.9 | 49.6 | 61.6 | 51.4 | 62.0 |
| {x1.0, x0.25, x1.0, x0.25} | 62.1 | 51.9 | 65.4 | 59.2 | 52.8 | 59.8 | 20.2 | 64.3 |
| Runtime-random (mean±std) | 63.8 ± 3.6 | 53.2 ± 3.8 | **65.1** ± **2.6** | 61.0 ± 4.5 | 63.8 ± 10.9 | 61.9 ± 5.1 | 48.1 ± 14.2 | 63.2 ± 3.2 |

TABLE II

SLIMMING PERFORMANCE OVERVIEW ON: ADAPTANET (OURS) WITH GROUPNORM (GN) AND LAYERNORM (LN), AND WITH DEPTH-SLIMMED MODULES (OURS($\delta$)). OUR UNIFORMLY TRAINED USNETS/DSNETS AND THEIR 3-RANDOMLY-SAMPLED VARIANTS (*). THE RANDOM RUNTIME EVALUATION ARE OVER 10 RUNS OF THE VALIDATION SETS, SAMPLE-BY-SAMPLE. EACH SAMPLE IS THEN EVALUATED ON A RANDOM SUB-NETWORK CONFIGURATION.

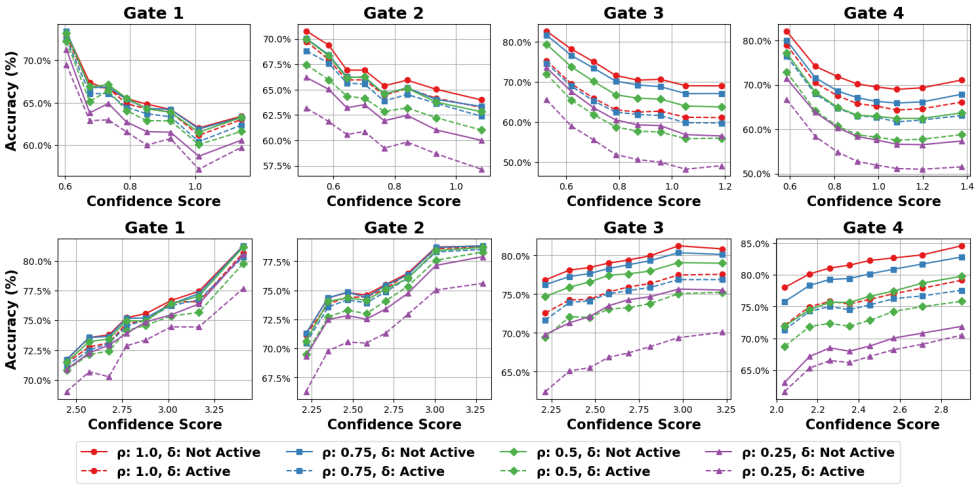


Fig. 4. Accuracy vs. confidence for AdaptaNet across gates and widths. Top: ImageNet. Bottom: CIFAR-100.

(inverse mapping). We evaluate three scopes; joint $(\rho, \delta)$, $\rho$-only, and $\delta$-only, and report accuracy, estimated MAdds, and overall validation set completion time in Table III. For reference we specify the Madd complexity of our AdaptaNet models at a $\times 0.5$ slimming to be 1383.3 and 414.9 for the $\rho$ and $\delta$ configurations respectively. Likewise the complexity at .75 is 2536.1 and 791.3. In aggregate, neither policy substantially outperforms uniform slimming at matched complexity though they do preform better than the average runtime-random subnetwork configurations. These results align with observations observed from the previous subsections. First, AdaptaNet exhibits relatively *stable* gradient accuracy across computational width complexity (Table II), so bin-to-choice mappings naturally average out. Second, our bins are equal-frequency and the validation set distributions mirror the training sets; under such stratification, a bin-wise policy effectively samples a near-uniform mixture of subnetworks.

These results imply the the Var Gates confidences scores are strong indicators of inter-feature quality, exploitable by heuristic choices.

## VI. CONCLUSIONS

In this paper we presented AdaptaNet, a modular slimmable methodology that unifies width and depth capacity control behind stable stage anchors. We instantiated it on a ResNet-50 backbone to study accuracy–cost trade-offs on CIFAR-100 and ImageNet compared to the foundational USNet and DSNet models. Our design combines anchor-layer supervision

| | ImageNet Validation Set | | | | | |
|---|---|---|---|---|---|---|
| | $\rho$ & $\delta$ | | $\rho$ only | | $\delta$ only | |
| | pull-up | pull-down | pull-up | pull-down | pull-up | pull-down |
| Accuracy(%) | 59.1 | 57.8 | 68.2 | 69.4 | 55.8 | 56.9 |
| Complexity(Madds) | 1623.2 | 1584.7 | 2208.3 | 2168.9 | 1076.5 | 994.3 |
| Execution Time (s) | 195.8±5.4 | 202.5±6.2 | 229.3±3.1 | 226.3±4.3 | 145.6±0.5 | 147.9±1.1 |

| | CIFAR-100 Validation Set | | | | | |
|---|---|---|---|---|---|---|
| | $\rho$ & $\delta$ | | $\rho$ only | | $\delta$ only | |
| | pull-up | pull-down | pull-up | pull-down | pull-up | pull-down |
| Accuracy(%) | 63.3 | 65.0 | 69.8 | 68.2 | 63.6 | 61.5 |
| Complexity(Madds) | 445.1 | 595.1 | 663.6 | 605.3 | 331.8 | 288.7 |
| Execution Time (s) | 49.9±1.9 | 51.3±2.3 | 64.2±1.2 | 64.9±1.4 | 35.4±0.6 | 36.6±0.7 |

TABLE III
PERFORMANCE EVALUATIONS OF THE PULL-UP AND PULL-DOWN STRATEGIES ON ADAPTANET INCLUDING VAR GATES, WITH OVERALL COMPUTATIONAL COMPLEXITY AND TOTAL TIME TO COMPLETE THE EVALUATION OF VALIDATION SETS.

trained via our cascade method with the MDSR, depth-slimmed modules that reduce serial depth while preserving anchor paths, and architecturally decoupled Var Gates that read anchor features to produce confidence scores for cost-aware subnetwork selection. The accompanying training study links normalization choices and subnetwork sampling strategies across USNet/DSNet and AdaptaNet, highlighting when shared-width statistics and depth slimming remain stable under mixed-width operation, and showing that simple confidence-bin heuristics can match uniform mixtures at comparable compute while improving over random selection. From our implementation of AdaptaNet, modularity suggests a path toward richer dynamic module implementations, structured modules for detection and segmentation backbones, and end-to-end learned routing and calibration schemes that co-design gates, substitutes, and system-level schedulers for resource-constrained deployment.

## REFERENCES


[1] Ferran Alet, Tomas Lozano-Perez, and Leslie Pack Kaelbling. "Modular meta-learning". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018, pp. 200–210.

[2] Mohamed Amer, Thomas Maul, and Vignesh Pandi. "A review of modular neural networks: applications, challenges, and perspectives". In: *AI Open* 1 (2020), pp. 5–22.

[3] Jacob Andreas et al. "Neural Module Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 39–48.

[4] Hadjer Benmeziane et al. "Hardware-Aware Neural Architecture Search: Survey and Taxonomy". In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)* (2021), pp. 4322–4329.

[5] Han Cai, Ligeng Zhu, and Song Han. "ProxylessNAS: Direct neural architecture search on target task and hardware". In: *International Conference on Learning Representations*. 2019.

[6] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255.

[7] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. "Neural architecture search: A survey". In: *Journal of Machine Learning Research* 20.55 (2019), pp. 1–21.

[8] Anirudh Goyal et al. "Recurrent independent mechanisms". In: *International Conference on Learning Representations (ICLR)* (2021).

[9] Song Han, Huizi Mao, and William J. Dally. "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding". In: *Proceedings of the International Conference on Learning Representations (ICLR)* (2016).

[10] Wei Han et al. "Modular Neural Network via Exploring Category Hierarchy". In: *Information Sciences* 569 (2021), pp. 496–507.

[11] Ian Harshbarger, Colin Zejda, and Marco Levorato. "Condar: Context-Aware Distributed Dynamic Object Detection on Radar Data". In: *MILCOM 2024 - 2024 IEEE Military Communications Conference (MILCOM)*. 2024, pp. 875–880.

[12] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[13] Yihui He et al. "Bounding Box Regression With Uncertainty for Accurate Object Detection". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 2883–2892.

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. Presented at the NIPS 2014 Deep Learning Workshop. https://arxiv.org/abs/1503.02531. 2014.

[15] Itay Hubara et al. "Quantized neural networks: Training neural networks with low precision weights and activations". In: *Journal of Machine Learning Research* 18.1 (2017), pp. 6869–6898.

[16] Timothy K Johnsen, Ian Harshbarger, and Marco Levorato. "An Overview of Adaptive Dynamic Deep Neural Networks via Slimmable and Gated Architectures". In: *2024 15th International Conference on Information and Communication Technology Convergence (ICTC)*. 2024, pp. 252–256.

[17] Timothy K Johnsen, Ian Harshbarger, and Marco Levorato. "An Overview of Adaptive Dynamic Deep Neural Networks via Slimmable and Gated Architectures". In: *2024 15th International Conference on Information and Communication Technology Convergence (ICTC)*. 2024, pp. 252–256.

[18] Timothy K. Johnsen and Marco Levorato. "NaviSlim: Adaptive Context-Aware Navigation and Sensing via Dynamic Slimmable Networks". In: *Proceedings of the 2024 IEEE/ACM Ninth International Conference on Internet-of-Things Design and Implementation (IoTDI)*. Hong Kong, China: IEEE, 2024, pp. 110–121.

[19] Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. "Shallow-Deep Networks: Understanding and Mitigating Network Overthinking". In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 3301–3310. URL: https://proceedings.mlr.press/v97/kaya19a.html.

[20] Hiroaki Kingetsu, Kenichi Kobayashi, and Taiji Suzuki. "Neural Network Module Decomposition and Recomposition". In: *CoRR* abs/2112.13208 (2021). URL: https://arxiv.org/abs/2112.13208.

[21] Diederik Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations* (Dec. 2015).

[22] Alex Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. Tech. rep. Technical report. University of Toronto, 2009. URL: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[23] Hideaki Kuratsu and Atsuyoshi Nakamura. "Slimmable Pruned Neural Networks". In: *CoRR* abs/2212.03415 (2022). URL: https://arxiv.org/abs/2212.03415.

[24] Basar Kutukcu, Sabur Baidya, and Sujit Dey. "SLEXNet: Adaptive Inference Using Slimmable Early Exit Neural Networks". In: *ACM Trans. Embed. Comput. Syst.* 23.6 (Sept. 2024). ISSN: 1539-9087.

[25] Changlin Li et al. "DS-Net++: Dynamic Weight Slicing for Efficient Inference in CNNs and Vision Transformers". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.4 (2023), pp. 4430–4446. DOI: 10.1109/TPAMI.2022.3194044.

[26] Changlin Li et al. "Dynamic Slimmable Network". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 8603–8613.

[27] Arnav Vaibhav Malawade, Trier Mortlock, and Mohammad Abdullah Al Faruque. "Hydrafusion: Context-aware selective sensor fusion for robust and efficient autonomous vehicle perception". In: *2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE. 2022, pp. 68–79.

[28] Sarthak Mittal, Yoshua Bengio, and Guillaume Lajoie. "Is a Modular Architecture Enough?" In: *Advances in Neural Information Processing Systems (NeurIPS)* 35 (2022), pp. 28747–28760. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/b8d1d741f137d9b6ac4f3c1683791e4a-Paper-Conference.pdf.

[29] Zanlin Ni et al. "Deep Incubation: Training Large Models by Divide-and-Conquering". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, pp. 17289–17299.

[30] Alexey Ozerov, Anne Lambert, and Suresh Kirthi Kumaraswamy. "ParaDiS: Parallelly Distributable Slimmable Neural Networks". In: *CoRR* abs/2110.02724 (2021). URL: https://arxiv.org/abs/2110.02724.

[31] Mingxing Tan and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 6105–6114. URL: https://proceedings.mlr.press/v97/tan19a.html.

[32] Mingxing Tan et al. "MnasNet: Platform-aware neural architecture search for mobile". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2820–2828.

[33] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. "Branchynet: Fast inference via early exiting from deep neural networks". In: *2016 23rd international conference on pattern recognition (ICPR)*. IEEE. 2016, pp. 2464–2469.

[34] Xin Wang et al. "SkipNet: Learning Dynamic Routing in Convolutional Networks". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 409–424.

[35] Bichen Wu et al. "FBNet: Hardware-aware efficient convnet design via differentiable neural architecture search". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 10734–10742.

[36] Le Yang et al. "AdaDet: An Adaptive Object Detection System Based on Early-Exit Neural Networks". In: *IEEE Transactions on Cognitive and Developmental Systems* 16.1 (2024), pp. 332–345.

[37] Jiahui Yu and Thomas Huang. "Universally slimmable networks and improved training techniques". In: *ICCV*. 2019, pp. 1803–1811.

[38] Jiahui Yu and Thomas S. Huang. "Universally Slimmable Networks and Improved Training Techniques". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019.

[39] Jiahui Yu et al. "Slimmable Neural Networks". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2019.