CONDITIONAL COMPLEMENTARY FEATURE LEARNING FOR EARLY-EXIT NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Early-exit networks enable adaptive inference on resource-constrained devices by allowing easy inputs to terminate early while harder inputs receive additional computation. Conventional depth-wise designs attach classifiers to a shared backbone, introducing two limitations: *architectural constraints*, which tie early-exit capacity to backbone position, and *early-late interference*, where shared features must support both immediate classification and continued processing, potentially degrading early-exit accuracy under joint training. Our approach is motivated by the intuition that shared-prefix computation may contribute to these limitations. Our key insight is to separate architectural coupling from sequential execution. Rather than evaluating a single network to increasing depth, we organize inference as a sequence of complete, exit-specific subnetworks. Since earlier representations are already available when a later exit is reached, later computation can focus on learning features that complement them. Guided by this principle of *conditional complementary feature learning*, we introduce an early-exit framework in which each subnetwork processes the original image through a full-depth feature hierarchy. This permits narrow, full-depth early predictors and relaxes the requirement that earlier features preserve all information needed for later computation. Each later subnetwork conditions feature extraction on the *stage-end features* of all preceding subnetworks, while its classifier combines the current and earlier final embeddings. Earlier representations thus guide subsequent extraction while remaining directly available for prediction. We instantiate the framework with CNN backbones and evaluate it on ImageNet and embedded CPU and GPU platforms. Our models improve the accuracy-efficiency trade-off, requiring up to 2.2$\times$ fewer FLOPs than the compared early-exit baselines at matched accuracy. Comparisons with Dynamic Perceiver show gains of up to 10.2 and 14.0 percentage points in top-1 accuracy at matched FLOPs and latency, respectively, and up to 2.5$\times$ lower latency at matched accuracy. Our work offers a new perspective on early-exit design, in which additional computation is conditioned on, and complementary to, what has already been computed.

1 INTRODUCTION

Deploying deep networks on resource-constrained platforms, such as mobile phones and embedded devices, requires accurate predictions under tight and often varying latency and energy budgets. Early-exit networks address this setting by providing multiple opportunities to terminate inference (Teerapittayanon et al., 2016; Huang et al., 2018; Han et al., 2021): in budgeted classification, easy inputs exit early, while harder inputs receive additional computation. The quality of an early-exit network is therefore measured by its accuracy across the entire range of budgets. An effective design must provide strong predictions at small budgets *and* use additional computation productively when inference continues.

The dominant approach attaches intermediate classifiers to a single shared backbone (Teerapittayanon et al., 2016; Kaya et al., 2019; Huang et al., 2018), so that the predictor at exit k is the *prefix* of the shared backbone up to that exit. Despite its simplicity, this organization has two key limitations. First, it creates *architectural constraints*. Early exits are restricted to shallow prefixes, limiting the depth available for learning discriminative features (Huang et al., 2018) Consequently, the capacity of a low-budget predictor is tied to its position within a larger network rather than cho-

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.

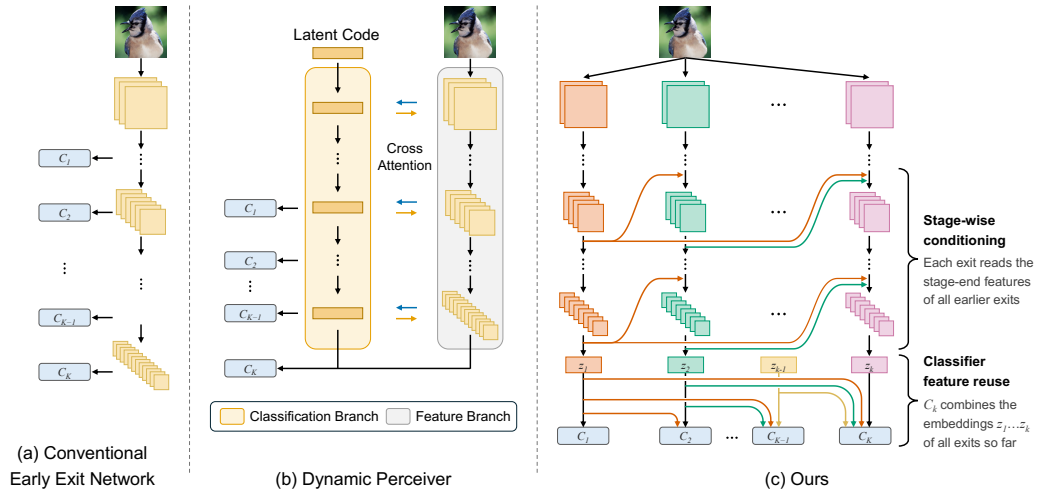


Figure 1: Overview of existing early-exit approaches vs. our proposed approach. **(a)** A conventional early-exit network attaches its classifiers $C_1, \ldots, C_K$ to intermediate layers of one shared backbone, so each early predictor is only a shallow front part of the later ones. **(b)** Dynamic Perceiver (Han et al., 2023) splits the network into a feature branch and a classification branch that exchange information through cross-attention, but all exits still draw on the same feature stream. **(c)** Our framework gives each exit its own full-depth network, run one after another. With *stage-wise conditioning*, a later network also takes in the stage-end features that earlier networks produced at each resolution. With *classifier feature reuse*, classifier C_k predicts from the final embeddings $z_1, \ldots, z_k$ of all exits so far. Earlier results are only read, never recomputed.

sen for its budget. Second, it causes *early-late interference*. A shared feature must be immediately discriminative for the current classifier while also preserving the information needed by all later layers, and these competing demands can degrade performance when exits are trained jointly (Huang et al., 2018; Li et al., 2019). Prior work mitigates this interference by rebalancing gradients (Li et al., 2019; Sun et al., 2022; Gong et al., 2024) or by separating classification and feature extraction into interacting branches (Han et al., 2023). However, since all exits still share one feature stream, the underlying coupling remains. For example, the first exit of Dynamic Perceiver (Han et al., 2023) still loses 3.05 points of accuracy when it is trained jointly with later exits (Section 4.4).

Both limitations arise because every early predictor is *part of* every later predictor. A natural alternative is to give each exit a complete, exit-specific subnetwork, which allows low-budget predictors to use moderately narrow, full-depth networks. If every subnetwork also processes the original input, later subnetworks can extract additional information directly, rather than relying on earlier features to preserve everything needed for subsequent processing. Earlier representations can thus serve as conditioning signals without being the sole source of input information for later computation. However, simply executing independent networks in sequence does not make direct use of earlier representations when forming later predictions. The challenge is to let later subnetworks build on earlier computation while retaining their ability to extract additional information from the input.

Our key insight is that later subnetworks can focus on extracting features that improve prediction *given the representations already available from earlier exits*. Sequential execution makes those representations available whenever inference continues: they can guide subsequent feature extraction and remain directly available for prediction. Later subnetworks therefore need not reproduce all earlier predictive content in their own representations. We call this design principle *conditional complementary feature learning*. This perspective separates sequential execution from the requirement that predictions come from increasingly deep prefixes of one network, and instead organizes inference around the accumulation of complementary features across complete subnetworks.

We realize this principle through three coordinated design choices (Figure 1). *(i)* Each exit owns a complete, exit-specific subnetwork that processes the original input through a full-depth feature hierarchy. This provides a complete feature-extraction path at every exit while preserving direct input access for later computation. *(ii)* Each later subnetwork conditions its feature extraction on the *stage-end features* of all preceding subnetworks at corresponding stages. We select the final repre-

sentation produced at each reused stage as context summarizing the processing already performed by an earlier subnetwork. *(iii)* Each classifier combines the current subnetwork's *final embedding* with those of all preceding subnetworks. Retained and newly extracted features therefore contribute jointly to prediction, allowing later features to supplement earlier ones.

We evaluate CNN instantiations of our framework on ImageNet and two embedded hardware platforms. Our models improve the accuracy-efficiency trade-off, with the largest gains at low and intermediate budgets. Against the broader set of early-exit baselines, our MobileNetV3 model family requires up to $2.2\times$ fewer FLOPs at matched accuracy. Comparisons with Dynamic Perceiver show gains of up to 10.2 and 14.0 percentage points in top-1 accuracy at matched FLOPs and latency, respectively, and up to $2.5\times$ lower latency at matched accuracy.

Contributions. Our main contributions are as follows. *(i)* We propose *conditional complementary feature learning* as a design principle for early-exit networks. *(ii)* We propose an early-exit framework that combines exit-specific subnetworks with causal reuse of stage-end features and final embeddings. *(iii)* Extensive experiments on ImageNet and two embedded hardware platforms show consistent improvements over existing early-exit methods, and our analysis confirms that the gains arise from stronger low-budget predictors, reduced interference under joint training, and more complementary later features.

2 Related Work

Early-exit and dynamic networks. Dynamic neural networks adapt their computation to the input or to resource constraints (Han et al., 2021), for example by skipping layers (Wu et al., 2018; Wang et al., 2018), pruning tokens (Rao et al., 2021), attending to image regions (Wang et al., 2020), or adjusting width (Yu et al., 2019; Li et al., 2021). Conventional depth-wise early-exit networks provide predictions at increasing budgets by attaching intermediate classifiers to a shared backbone. BranchyNet (Teerapittayanon et al., 2016) and SDN (Kaya et al., 2019) add exits to standard networks, MSDNet (Huang et al., 2018) maintains multi-scale features so that early classifiers receive coarse-level information, and RANet (Yang et al., 2020) further adapts input resolution across exits. Dynamic Perceiver (Han et al., 2023) decouples feature extraction from classification with interacting branches. Another line of work improves exit policies and prediction reliability (Regol et al., 2024; Bajpai & Hanawal, 2025; Jazbec et al., 2023; 2024). Most of these architectures obtain every exit from a single shared feature stream, so low-budget predictors inherit only its shallow computation. In contrast, our framework gives every exit its own complete, full-depth subnetwork and is orthogonal to the choice of exit policy.

Training multi-exit networks. Jointly optimizing multiple exits has motivated distillation from later to earlier exits (Phuong & Lampert, 2019; Dong et al., 2026), gradient equilibrium in IMTA (Li et al., 2019), learned gradient fusion in Meta-GF (Sun et al., 2022), exit-specific sample weighting in L2W (Han et al., 2022), and systematic studies of joint and disjoint training (Kubaty et al., 2024). Deep Feature Surgery (DFS) (Gong et al., 2024) partitions features into shared and exit-specific parts and blocks later-exit gradients through the protected parts. These methods mitigate early-late interference by changing *how* a shared backbone is optimized. In contrast, our framework uses complete, exit-specific subnetworks with direct image access, so earlier features are no longer the sole source of information for later computation. We train our models with a plain sum of classification losses, without distillation or gradient manipulation.

Progressive predictors and feature reuse. Several works build predictors that are not shallow prefixes of a larger network. Vertical slicing (Lee & Shin, 2018) constructs thin, full-depth subnetworks, and its incremental variant reuses depth-aligned features layer by layer. Hierarchical ensembles (Ruiz & Verbeek, 2021) obtain anytime predictions from a hierarchy of distilled networks. In vision Transformers, DVT (Wang et al., 2021) cascades Transformers with increasing token budgets and reuses upstream features and attention, while CF-ViT (Chen et al., 2023), MG-ViT (Zhang et al., 2023), ThinkingViT (Hojjat et al., 2026b), RAViT (Guidez et al., 2026), and ProgResViT (Hojjat et al., 2026a) progressively increase resolution, granularity, or width. At the prediction level, Zero Time Waste (Wołczyk et al., 2021) and BoostNet (Yu et al., 2023) reuse or boost earlier predictions rather than conditioning later feature extraction. These methods reuse either intermediate features at matched depths or only final predictions. In contrast, our framework conditions each later feature extractor on the *stage-end* features of all earlier exits and reuses their final embeddings

in the classifier. Together, these two forms of reuse are designed for conditional complementary feature learning rather than for feature reuse alone, and our ablations show that both are necessary (Section 4.5).

3 METHOD

We propose an early-exit framework organized around *conditional complementary feature learning*: each later subnetwork learns features that improve prediction given the representations already computed by earlier subnetworks. Our framework comprises a sequence of K complete, exit-specific subnetworks. Each processes the original image, conditions its feature extraction on earlier representations, and combines its own final embedding with earlier embeddings for prediction (Figure 1). This organization provides a complete feature hierarchy at every exit while allowing additional computation to build on retained features. In this work, we formulate and evaluate the framework for image classification with stage-structured CNN backbones.

3.1 CONDITIONAL COMPLEMENTARY FEATURE LEARNING

From shared prefixes to complete subnetworks. A conventional depth-wise early-exit network divides a backbone into segments $g_1, \ldots, g_K$ and attaches a classifier $\mathcal{C}_k$ after each segment:

$$s_k = g_k(s_{k-1}), \qquad s_0 = x, \qquad \ell_k = \mathcal{C}_k(s_k), \tag{1}$$

where x is the input image and ℓ_k denotes the class logits at exit k. The feature extractor available to an early exit is a prefix of the backbone. Its depth is therefore tied to the exit's position, and its output must serve both immediate classification and continued processing. These roles can place conflicting demands on the shared representation.

Giving each exit a complete subnetwork allows even an early exit to use a full feature hierarchy. Direct access to the image also gives later subnetworks a source of information beyond the representations retained by earlier ones. This relaxes the requirement that early features preserve everything needed for later inference. However, if each subnetwork predicts independently, earlier features do not contribute to later predictions, and successive subnetworks may duplicate feature extraction.

Learning given earlier representations. When inference reaches exit k, subnetworks $1, \ldots, k-1$ have already executed. Their features can therefore inform both *what the next subnetwork extracts* and *how its prediction is formed*. We use earlier stage features to condition feature extraction, and retain earlier final embeddings as direct inputs to the later classifier. The later subnetwork can then use its access to the image to extract features that are useful in combination with the retained representations. We call this *conditional complementary feature learning*. Here, complementarity means incremental predictive usefulness; it does not require statistically independent features or an explicit decomposition of image information. The following architectural choices implement this learning intuition.

3.2 COMPLETE EXIT-SPECIFIC SUBNETWORKS

Each exit has its own stem, backbone, embedding transformation, and classifier, with parameters distinct from those of other exits. Every subnetwork starts from the original image x and follows a complete feature-extraction hierarchy. Subnetwork depth is independent of exit order, so even an early exit can use a narrow, full-depth network. Later subnetworks augment their local computation with features from all preceding exits. Thus, *complete* describes the subnetwork's path from the image to its output; later predictions also depend on the reused representations.

In our CNN instantiations, feature extraction proceeds through stages at decreasing spatial resolutions. Within a model, subnetworks follow a common backbone family, stage organization, and block-depth schedule. Exit-specific width multipliers control their channel capacity, while corresponding spatial resolutions provide locations for feature reuse. The backbone choices and model configurations used in our experiments are specified in Section 4.1 and Appendix A.1.

Dependencies run only from earlier to later exits. The first subnetwork has no incoming reuse connections and has the forward computation of a standalone predictor. Each later subnetwork

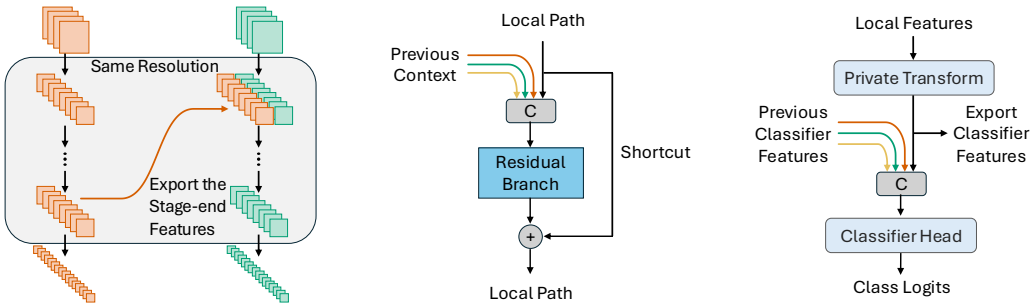


Figure 2: **Components of conditional complementary feature learning. (a)** Each earlier subnetwork exports its stage-end feature, the last feature map at each resolution, to later subnetworks at the same resolution. **(b)** A fusion block concatenates these features with its local input; the residual branch computes an update from this fused context, while the shortcut keeps the local state. **(c)** Each classifier combines the current subnetwork's embedding z_k with those of all earlier subnetworks.

receives the original image and cached features from every preceding subnetwork. No prediction requires computation from a future exit.

3.3 Stage-Wise Conditioning

Stage-end features as context. To inform later extraction with the results of earlier processing, each subnetwork exports a *stage-end feature* at each reused spatial resolution. Let $m_{j,r}$ denote the last representation produced by subnetwork j at resolution r, before downsampling or backbone termination. Here, stage-end refers to completed processing within that resolution. Because the earlier subnetwork has finished executing, this representation is available even at the beginning of the later subnetwork's processing at the same resolution (Figure 2a). We use it as a summary of the earlier subnetwork's processing at that scale. Exit k receives $m_{1,r}, \ldots, m_{k-1,r}$ from all preceding exits.

A conditioned update to the local state. For $k > 1$, fusion occurs once per reused resolution, at the first block that receives an input at that resolution. Let $u_{k,r}$ be the local input to this block. We concatenate the local state with the earlier stage-end features along the channel dimension and project the result back to the local width:

$$\widetilde{u}_{k,r} = \mathcal{P}_{k,r}\left(\mathrm{Concat}\left(u_{k,r}, m_{1,r}, \ldots, m_{k-1,r}\right)\right), \tag{2}$$

where $\mathcal{P}_{k,r}$ is a learned channel projection. The shape-preserving residual block at the fusion location then computes

$$u^{+}_{k,r} = u_{k,r} + \mathcal{T}_{k,r}\left(\widetilde{u}_{k,r}\right), \tag{3}$$

where $\mathcal{T}_{k,r}$ is the block transformation without its native residual addition. The fused features enter the residual branch, while the shortcut carries the unfused local state (Figure 2b). Earlier context therefore conditions an update to the local representation. Preserving the local shortcut is a deliberate design choice; Section 4.5 compares it with using the fused features in both paths.

Subsequent blocks at the same resolution continue processing without additional fusion. Earlier cached features are read without being overwritten. Fusing once per resolution and projecting to the local width limit the cost of reuse: the subsequent backbone blocks operate at the subnetwork's own width even as the accumulated context grows.

3.4 Classifier Feature Reuse

Stage-wise conditioning informs feature extraction, while classifier feature reuse makes earlier representations directly available for prediction. Let h_k denote the final backbone representation of subnetwork k. An exit-specific embedding function $\mathcal{E}_k$, comprising spatial pooling and a learned

transformation, produces the final embedding $z_k = \mathcal{E}_k(h_k)$. The classifier combines all embeddings computed so far:

$$a_k = \text{Concat}\left(z_1, \ldots, z_k\right), \qquad \ell_k = \mathcal{C}_k(a_k), \tag{4}$$

where $\mathcal{C}_k$ refines the combined embedding and produces class logits (Figure 2c). Each subnetwork exports z_k before aggregation, so each embedding appears once in a_k.

The classifier can use earlier embeddings directly, so the current subnetwork need not reproduce their predictive content in its own embedding. Conditioned on earlier stage features, it can extract additional features from the image that improve prediction when combined with the retained embeddings. We examine the resulting feature complementarity in Section 4.6.

3.5 Joint Training and Sequential Inference

Joint training. All exits are trained jointly from initialization. For an image x with label y, we minimize an unweighted sum of cross-entropy losses:

$$\mathcal{L}(x, y) = \sum_{k=1}^{K} \text{CE}\left(\ell_k(x), y\right). \tag{5}$$

We use neither distillation nor an auxiliary complementarity objective. Gradients propagate through both stage-wise conditioning and classifier feature reuse, allowing later losses to update earlier subnetworks. Thus, private parameters and local states do not imply gradient isolation. Direct image access relaxes the dependence of later computation on earlier representations, but joint optimization can still create interactions between exits.

Sequential inference. Subnetworks execute in exit order. After exit k produces logits ℓ_k, a stopping policy may return its prediction or continue to exit $k + 1$. If inference continues, the next subnetwork processes the image with cached stage-end features and final embeddings from all earlier subnetworks; previously computed features are not recomputed. The cumulative cost c_k includes all computation through exit k, including preceding subnetworks, classifiers, and reuse operations. The framework provides a prediction at every exit and does not prescribe a stopping policy. Our experiments use confidence thresholds calibrated for budgeted batch classification, as described in Section 4.1.

4 Experiments

We evaluate our framework on ImageNet against existing early-exit methods, measuring accuracy across computational budgets and on real hardware (Sections 4.2 and 4.3). We then return to the motivation in Section 1 and verify that the gains arise from stronger low-budget predictors with less interference (Section 4.4), from stage-wise conditioning (Section 4.5), and from more complementary later features (Section 4.6).

4.1 Experimental Setup

Models. We instantiate our framework with MobileNetV3 (MBV3) (Howard et al., 2019) and EfficientNetV2 (EffV2) (Tan & Le, 2021). A family of MobileNetV3 models spans the compute range of the main comparison. For controlled comparisons with Dynamic Perceiver (DP) (Han et al., 2023), we additionally construct one MobileNetV3 model and one EfficientNetV2 model whose first- and final-exit computation approximately match selected DP configurations. All models are trained on ImageNet and evaluated on its validation set. Architectures, training recipes, and preprocessing are detailed in the Appendix A.

Budgeted Batch classification. Following MSDNet (Huang et al., 2018) and DP (Han et al., 2023), we calibrate confidence thresholds for each target budget on 50,000 images randomly sampled from the ImageNet training set, and then evaluate the resulting exit policy on the validation set. An image terminates at the first exit whose confidence threshold is met. Thresholds are calibrated separately for each model.

Compute accounting. The cost of an exit includes all operations required to reach it, including preceding predictors, classifiers, and feature-reuse operations. We count one multiply-add as one

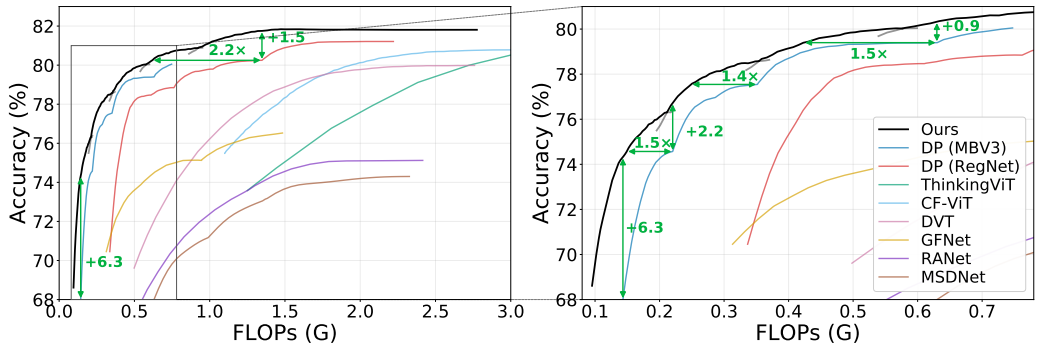


Figure 3: Budgeted batch classification against early-exit baselines. The black curve shows the envelope of our MobileNetV3-based models. Gray segments show the underlying individual model curves near points where the envelope switches between configurations. The right panel enlarges the boxed low-compute region in the left panel. Vertical annotations show accuracy gains at matched FLOPs, and horizontal annotations show FLOP reductions at matched accuracy. Comparisons are restricted to overlapping measured compute ranges.

FLOP, consistent with the compared baselines, and report average FLOPs per validation image. For the model family, we display the upper accuracy-compute envelope across trained configurations and restrict comparisons to overlapping measured ranges without extrapolation.

Baselines. We compare with DP (Han et al., 2023), MSDNet (Huang et al., 2018), RANet (Yang et al., 2020), GFNet (Wang et al., 2020), DVT (Wang et al., 2021), CF-ViT (Chen et al., 2023), and ThinkingViT (Hojjat et al., 2026b). DP is our principal controlled comparator because it explicitly targets the tension between early classification and continued feature extraction. We reproduce its MobileNetV3- and RegNet-based models under our setup, while other baseline curves are digitized from their original papers.

Hardware. We measure cumulative exit latency on a Raspberry Pi 4 CPU with XNNPACK and an NVIDIA Jetson AGX Xavier GPU with TensorRT, using batch size one and FP32 inference. Accuracy-latency curves weight these measured latencies by validation-set exit frequencies. Device settings and timing procedures are provided in the Appendix A.3.

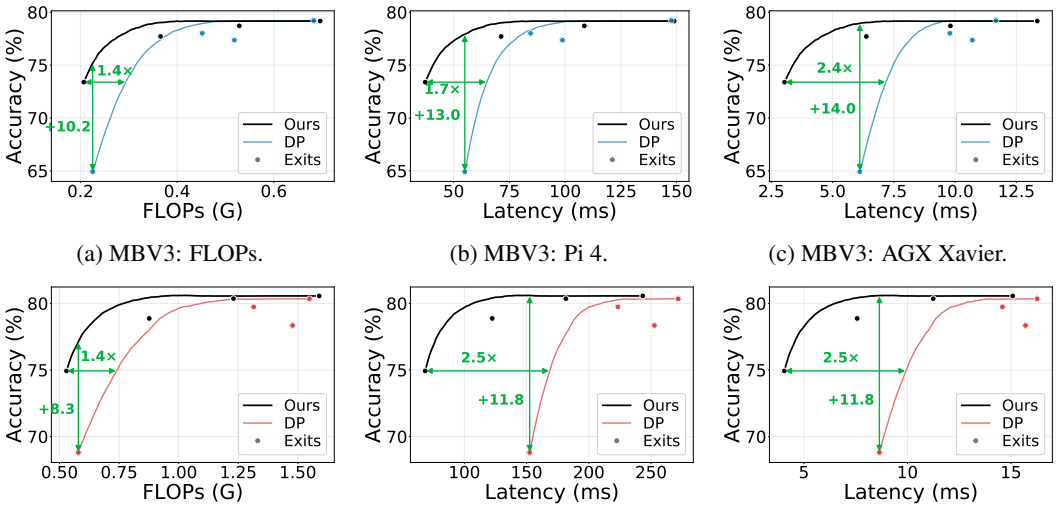
4.2 Budgeted Batch Classification Across Compute Ranges

Figure 3 compares our MobileNetV3-based models against existing early-exit methods over their overlapping compute ranges. Our models form a stronger accuracy-compute frontier, with the largest gains at low and intermediate compute. The right panel enlarges the boxed low-compute region for a more detailed comparison. At matched FLOPs, the improvement reaches 6.3 percentage points, while at matched accuracy, our models require up to $2.2\times$ fewer FLOPs. These gains are largest where shared-backbone methods rely on shallow early exits, consistent with the limitation identified in Section 1. The gap narrows at higher compute as competing methods progress to deeper exits.

4.3 Matched Comparisons and Hardware Efficiency

We compare our MobileNetV3- and EfficientNetV2-based models with reproduced DP models based on MobileNetV3 and RegNet, respectively. In each comparison, our model and the corresponding DP model approximately match cumulative computation at the first and final exits. Since intermediate exit costs differ, we compare adaptive inference at matched average budgets rather than by exit index. Figure 4 reports accuracy against FLOPs and measured latency. Fixed-exit results are reported in the Appendix A.5.

Our models achieve substantially higher accuracy at low and intermediate compute budgets while reaching comparable accuracy at the final exit. Figures 4(a) and (d) show this pattern for MobileNetV3 and EfficientNetV2/RegNet pair, respectively. For MobileNetV3, the improvement reaches 10.2 percentage points at matched FLOPs, and our model requires up to $1.4\times$ fewer FLOPs



(a) MBV3: FLOPs. (b) MBV3: Pi 4. (c) MBV3: AGX Xavier.

(d) EffV2/RegNet: FLOPs. (e) EffV2/RegNet: Pi 4. (f) EffV2/RegNet: AGX Xavier.

Figure 4: Controlled budgeted comparisons with Dynamic Perceiver (DP). Top row: our MobileNetV3 model versus MobileNetV3-based DP. Bottom row: our EfficientNetV2 model versus RegNet-based DP. Columns show accuracy against FLOPs, Raspberry Pi 4 latency, and NVIDIA AGX Xavier latency. Vertical and horizontal annotations show gains at matched budget and matched accuracy, respectively.

Table 1: Effect of joint multi-exit training on the first-exit accuracy. Joint-training loss is the accuracy decrease from standalone to joint training, $\mathrm{Acc}_{\mathrm{standalone}} - \mathrm{Acc}_{\mathrm{joint}}$, lower is better.

Method	Joint	Standalone	Loss
DP	64.92	67.97	3.05
Ours	73.39	74.55	**1.16**

Table 2: Feature-reuse ablation on the matched MobileNetV3 model: ImageNet top-1 accuracy (%) at every exit.

Reuse	Exit 1	Exit 2	Exit 3	Exit 4
Classifier-only	73.21	76.16	76.34	76.55
Matched-depth	73.99	76.30	77.41	77.37
Fused-shortcut	**74.03**	76.71	77.81	78.10
Stage-end (ours)	73.39	**77.70**	**78.70**	**79.17**

at matched accuracy. The EfficientNetV2/RegNet comparison shows a similar pattern, with an improvement of 8.3 points and up to $1.4\times$ fewer FLOPs.

The low- and intermediate-budget advantage persists when efficiency is measured by hardware latency rather than FLOPs. Figures 4(b), (c), (e), and (f) show the results on the Raspberry Pi 4 and NVIDIA AGX Xavier. At matched accuracy, our models require up to $2.5\times$ less latency, while the accuracy improvement at matched latency reaches 14.0 percentage points. At the final exit, latency is similar for MobileNetV3 on the Raspberry Pi 4 and slightly lower for DP on the AGX Xavier, whereas our EfficientNetV2 model has lower final-exit latency on both platforms.

4.4 Stronger Low-Budget Predictors with Less Interference

We evaluate whether exit-specific subnetworks address the two limitations identified in Section 1: weak low-budget predictors and interference from joint training. Table 1 compares the first exit of each matched MobileNetV3 model with the same predictor trained independently. Our standalone predictor improves first-exit accuracy by 6.58 percentage points over DP, while our first exit incurs a $2.6\times$ smaller accuracy loss under joint training. This smaller degradation occurs even though DP uses self-distillation (Han et al., 2023), whereas our model is trained using only classification losses. These results show that exit-specific subnetworks provide a stronger low-budget predictor and better preserve its accuracy under joint training.

Table 3: Complementarity of exit-1 and exit-2 representations. ImageNet top-1 validation accuracy (%). Gain is $\mathrm{Acc}([z_1, z_2]) - \max\{\mathrm{Acc}(z_1), \mathrm{Acc}(z_2)\}$. Results are averaged over two seeds.

Reuse scheme	Probe	z_1	z_2	Joint	Gain
Classifier-only	Linear	72.3	54.5	74.6	2.3
	MLP	72.0	54.6	74.0	2.1
Matched-depth	Linear	**73.1**	60.7	74.9	1.8
	MLP	**72.8**	60.7	74.7	1.9
Stage-end (ours)	Linear	72.0	**71.8**	**76.2**	**4.2**
	MLP	71.7	**72.0**	**75.8**	**3.8**

4.5 Ablation of Cross-Exit Feature Reuse

We evaluate how the source and integration of reused features affect later-exit accuracy. Table 2 compares three variants of our reuse mechanism on the matched MobileNetV3 model. *Classifier-only* removes stage-wise conditioning and retains only classifier feature reuse. *Matched-depth* replaces each stage-end feature with the feature at the corresponding depth of the earlier predictor. At the first block of each resolution, this is the downsampled feature entering that resolution. *Fused-shortcut* feeds the fused feature to both the residual and shortcut branches, instead of preserving the local feature on the shortcut path. Our full design consistently performs best once cross-exit reuse begins. Table 2 shows that it achieves the highest accuracy at Exits 2–4, with the advantage becoming more pronounced as reuse accumulates. This supports the use of stage-wise conditioning, stage-end features, and preservation of the local feature during fusion.

4.6 Complementarity of Cross-Exit Representations

We evaluate whether conditioning on stage-end features encourages the second-exit representation to complement information already available at the first exit. Table 3 reports linear and MLP probe accuracy using the exit-1 embedding z_1, the exit-2 embedding z_2, and their concatenation. We define complementarity gain as the improvement of the joint representation over the better individual representation. Probe capacity is held fixed across inputs, and all model features are frozen.

Conditioning on stage-end features produces the largest complementarity gain for both probe families. It also yields the strongest z_2 representation, showing that greater complementarity does not come at the expense of its individual predictive quality. Together, these results indicate that conditioning on stage-end features encourages later features to remain individually informative while contributing additional predictive information beyond earlier representations.

5 Conclusion

We traced two known limitations of early-exit networks built on a shared backbone, weak low-budget predictors and early-late interference, to a common structural cause: every early predictor is part of every later one. To address this, we proposed conditional complementary feature learning and realized it by giving each budget a complete, exit-specific subnetwork that is conditioned on the stage-end features and final embeddings of earlier exits. On ImageNet, our framework establishes a stronger accuracy-compute frontier than existing early-exit methods, with the largest gains at low and intermediate budgets, and the gains persist under measured latency on embedded CPU and GPU platforms. Our analysis attributes these gains to stronger and better-preserved low-budget predictors and to more complementary later features.

Limitations and future work. First, stage-wise conditioning relies on exits sharing a stage template with corresponding resolutions, so we have only instantiated the framework for stage-structured CNNs; extending it to vision Transformers is a natural next step. Second, the reused context grows with the number of exits, and although earlier features are projected to the local width, projection cost and feature caching grow accordingly. Third, our evaluation is limited to ImageNet classification, and our controlled comparison with Dynamic Perceiver does not fully isolate architecture from training procedure. Applying conditional complementary feature learning to dense prediction tasks and to other exit policies is an interesting direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools for three tasks: *(i)* editing the manuscript, including polishing language and restructuring the presentation of the paper; *(ii)* writing and debugging code for training, evaluation, and hardware measurement; and *(iii)* searching for and summarizing related literature. We have not used generative AI tools for research ideation, method design, experimental design, or the generation or analysis of experimental results; all reported numbers come directly from our own experiments. We have reviewed all AI-assisted work. The authors read and revised all AI-edited text and checked every number and claim against our experimental records. Every citation was verified manually against the original publication. All AI-assisted code was reviewed and tested by the authors before being used to produce results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work proposes an architecture for efficient early-exit image classification and does not involve human subjects, user studies, or the collection of new data. All experiments use the publicly available ImageNet dataset under its standard terms of use. ImageNet is known to contain images of people and to reflect demographic and geographic biases, which models trained on it, including ours, may inherit; we do not claim that our models are fair or robust across subpopulations, and practitioners should evaluate them for their target application before deployment. Our method aims to reduce the computation, latency, and energy required for inference on embedded devices, which can lower the environmental cost of deployment and make accurate models accessible on inexpensive hardware. As with other efficient vision models, the same efficiency could also make on-device recognition easier to deploy in applications with privacy or surveillance concerns; our work does not target such applications, and we encourage its responsible use.

REPRODUCIBILITY STATEMENT

We have made efforts to ensure that our results can be reproduced. Section 3 specifies the complete architecture, including exit-specific subnetworks, the extraction and fusion of stage-end features (Equations 2 and 3), classifier feature reuse (Equation 4), and the training objective (Equation 5). Section 4.1 describes the budgeted-classification protocol, including threshold calibration on 50,000 randomly sampled ImageNet training images, FLOP accounting, the reproduced Dynamic Perceiver baselines, and the hardware setup (Raspberry Pi 4 with XNNPACK and NVIDIA Jetson AGX Xavier with TensorRT, batch size one, FP32). Section 4.6 describes the probing protocol. Model configurations, training recipes, preprocessing, device settings and timing procedures, and probe details are provided in the appendix. All experiments use the publicly available ImageNet dataset. We will release our code and model configurations upon publication of this paper.

REFERENCES

Divya Jyoti Bajpai and Manjesh Kumar Hanawal. Beem: Boosting performance of early exit dnns using multi-exit classifiers as experts. In *International Conference on Learning Representations*, volume 2025, pp. 62520–62535, 2025.

Mengzhao Chen, Mingbao Lin, Ke Li, Yunhang Shen, Yongjian Wu, Fei Chao, and Rongrong Ji. Cf-vit: A general coarse-to-fine method for vision transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 2023.

Yonghao Dong, Qiang He, Penghong Rui, Zhenzhe Zheng, Zhao Li, Feifei Chen, Hai Jin, and Yun Yang. Envit: Enhancing the performance of early-exit vision transformers via exit-aware structured dropout-enabled self-distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 20852–20860, 2026.

Cheng Gong, Yao Chen, Qiuyang Luo, Ye Lu, Tao Li, Yuzhi Zhang, Yufei Sun, and Le Zhang. Deep feature surgery: Towards accurate and efficient multi-exit networks. In *European Conference on Computer Vision*, pp. 435–451. Springer, 2024.

Martial Guidez, Stefan Duffner, and Christophe Garcia. Ravit: Resolution-adaptive vision transformer. *arXiv preprint arXiv:2602.24159*, 2026.

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, 2021.

Yizeng Han, Yifan Pu, Zihang Lai, Chaofei Wang, Shiji Song, Junfeng Cao, Wenhui Huang, Chao Deng, and Gao Huang. Learning to weight samples for dynamic early-exiting networks. In *European conference on computer vision*, pp. 362–378. Springer, 2022.

Yizeng Han, Dongchen Han, Zeyu Liu, Yulin Wang, Xuran Pan, Yifan Pu, Chao Deng, Junlan Feng, Shiji Song, and Gao Huang. Dynamic perceiver for efficient visual recognition. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5969–5979. IEEE, 2023.

Ali Hojjat, Janek Haberer, and Olaf Landsiedel. Progresvit: Progressive resolution and width for adaptive vision transformers. *arXiv preprint arXiv:2609.03216*, 2026a.

Ali Hojjat, Janek Haberer, Soren Pirk, and Olaf Landsiedel. Thinkingvit: Matryoshka thinking vision transformer for elastic inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026b.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1314–1324, 2019.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens Van Der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations (ICLR)*, 2018.

Metod Jazbec, James Allingham, Dan Zhang, and Eric Nalisnick. Towards anytime classification in early-exit architectures by enforcing conditional monotonicity. *Advances in Neural Information Processing Systems*, 36:56138–56168, 2023.

Metod Jazbec, Alexander Timans, Tin H Veljković, Kaspar Sakmann, Dan Zhang, Christian A Naesseth, and Eric Nalisnick. Fast yet safe: Early-exiting with risk control. *Advances in Neural Information Processing Systems*, 37:129825–129854, 2024.

Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. Shallow-deep networks: Understanding and mitigating network overthinking. In *International conference on machine learning*, pp. 3301–3310. PMLR, 2019.

Piotr Kubaty, Bartosz WĂłjcik, BartĹ Krzepkowski, Monika Michaluk, Jary Pomponi, Kamil Adamczewski, et al. How to train your multi-exit model? analyzing the impact of training strategies. *arXiv preprint arXiv:2407.14320*, 2024.

Hankook Lee and Jinwoo Shin. Anytime neural prediction via slicing networks vertically. *arXiv preprint arXiv:1807.02609*, 2018.

Changlin Li, Guangrun Wang, Bing Wang, Xiaodan Liang, Zhihui Li, and Xiaojun Chang. Dynamic slimmable network. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8603–8613. IEEE, 2021.

Hao Li, Hong Zhang, Xiaojuan Qi, Ruigang Yang, and Gao Huang. Improved techniques for training adaptive deep networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1891–1900, 2019.

Mary Phuong and Christoph H Lampert. Distillation-based training for multi-exit architectures. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1355–1364, 2019.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems*, 34:13937–13949, 2021.

Florence Regol, Joud Chataoui, and Mark Coates. Jointly-learned exit and inference for a dynamic neural network. In *International Conference on Learning Representations*, volume 2024, pp. 39518–39546, 2024.

Adria Ruiz and Jakob Verbeek. Anytime inference with distilled hierarchical neural ensembles. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 9463–9471, 2021.

Yi Sun, Jian Li, and Xin Xu. Meta-gf: Training dynamic-depth neural networks harmoniously. In *European Conference on Computer Vision*, pp. 691–708. Springer, 2022.

Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *International conference on machine learning*, pp. 10096–10106. PMLR, 2021.

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)*, pp. 2464–2469. IEEE, 2016.

Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *European Conference on Computer Vision*, pp. 420–436. Springer, 2018.

Yulin Wang, Kangchen Lv, Rui Huang, Shiji Song, Le Yang, and Gao Huang. Glance and focus: a dynamic approach to reducing spatial redundancy in image classification. *Advances in neural information processing systems*, 33:2432–2444, 2020.

Yulin Wang, Rui Huang, Shiji Song, Zeyi Huang, and Gao Huang. Not all images are worth 16x16 words: Dynamic transformers for efficient image recognition. *Advances in neural information processing systems*, 34:11960–11973, 2021.

Maciej Wołczyk, Bartosz Wójcik, Klaudia Bałazy, Igor T Podolak, Jacek Tabor, Marek Śmieja, and Tomasz Trzcinski. Zero time waste: Recycling predictions in early exit neural networks. *Advances in Neural Information Processing Systems*, 34:2516–2528, 2021.

Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8817–8826. IEEE, 2018.

Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. Resolution adaptive networks for efficient inference. In *2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 2366–2375. IEEE, 2020.

Haichao Yu, Haoxiang Li, Gang Hua, Gao Huang, and Humphrey Shi. Boosted dynamic neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 10989–10997, 2023.

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

Yu Zhang, Yepeng Liu, Duoqian Miao, Qi Zhang, Yiwei Shi, and Liang Hu. Mg-vit: a multi-granularity method for compact and efficient vision transformers. *Advances in Neural Information Processing Systems*, 36:69328–69347, 2023.

A Experimental Details

We provide model configurations and training settings, the hardware measurement protocol, details of the analysis experiments, and per-exit results for the matched comparisons.

A.1 Model Configurations

Table 4 lists the MobileNetV3-based models used for the budgeted batch classification comparison. Table 5 gives the configurations used in the matched comparisons: our MobileNetV3 model is paired with MobileNetV3-based Dynamic Perceiver (DP) model 4, and our EfficientNetV2 model with RegNet-based DP model 4.

Within each model, subnetworks share a backbone family, stage organization, and block-depth schedule, while exit-specific width factors control the channel capacity of each complete subnetwork. Repeat counts are listed for the six operator groups at output resolutions $[112, 56, 28, 14_a, 14_b, 7]$. Fusion resolutions specify where later subnetworks use stage-wise conditioning.

Model 4* uses exactly the same trained checkpoint as Model 4, without retraining or fine-tuning. For evaluation, we resize the image's shorter side to 329 pixels using bicubic interpolation, center-crop to 288×288, convert to a tensor, and apply ImageNet mean/std normalization. The resize size preserves the standard crop ratio of 0.875 ($288/0.875 \approx 329$), analogous to resizing to 256 and center-cropping to 224. Evaluation uses a single center crop, without test-time augmentation or post-crop upsampling.

Table 4: Configurations of the MobileNetV3-Large model family used for the budgeted batch classification curves. $a/b\times5$ denotes width a at exit 1 and width b at exits 2–6.

Configuration	Model 1	Model 2	Model 3	Model 4	Model 4*
Backbone	MobileNetV3-Large				
Number of exits	6				
Channel divisor	4				
MBConv repeats	$[1, 2, 2, 2, 2, 3]$	$[1, 2, 3, 4, 2, 3]$	$[1, 2, 3, 4, 3, 4]$	$[1, 2, 3, 4, 4, 5]$	$[1, 2, 3, 4, 4, 5]$
Exit width factors	$.65/.45\times5$	$.75/.60\times5$	$.90/.70\times5$	$1.10/.85\times5$	$1.10/.85\times5$
Fusion resolutions	$[56, 28, 14, 7]$				
Classifier export width	960				
Classifier hidden width	1280				
Classifier dropout	0.2				
Training size	224				
Test size	224	224	224	224	288

Table 5: Configurations used in the matched-compute comparisons. $a/b\times3$ denotes width a at exit 1 and width b at exits 2–4.

Configuration	Ours–MBV3	Ours–EffV2
Backbone	MBV3-L	EffV2-B0
Number of exits	4	
Block repeats	$[1, 2, 3, 4, 3, 4]$	$[1, 2, 2, 3, 4, 5]$
Exit width factors	$.86/.73\times3$	$.93/.70\times3$
Channel divisor	4	
Fusion resolutions	$[56, 28, 14, 7]$	
Classifier export width	960	1280
Classifier hidden width	1280	
Classifier dropout	0.2	
Reuse gradient coefficient	1.0	
Training size	224	
Test size	224	

A.2 Training Settings

Our MobileNetV3 and EfficientNetV2 models use the settings in Table 6. All exits are trained jointly from initialization with an unweighted sum of cross-entropy losses, without distillation or an auxiliary complementarity objective. For the standalone first-exit comparison, our MobileNetV3 first-exit predictor uses exactly the same recipe, with only the later subnetworks and their losses removed.

The reproduced DP models follow the original DP training recipe (Han et al., 2023), including self-distillation during joint multi-exit training. The standalone DP first-exit predictor follows that recipe without self-distillation: it has no later exits and no teacher.

Table 6: ImageNet-1K training settings shared by the MobileNetV3 and EfficientNetV2 models.

Training configuration	Value
Optimizer	SGD
Base learning rate	0.1
Momentum	0.9
Weight decay	1×10^{-5}
Weight-decay exclusions	BN and bias
Global batch size	256
Training epochs	300
Learning-rate schedule	cosine decay
Warmup	5 epochs, linear from 0.001
Input resolution	224^2
Train crop/interpolation	random resized/bicubic
Random horizontal flip	yes
Test resize/crop	$256/224$
Automatic augmentation	TrivialAugment Wide
Random erasing probability	0.1
Label smoothing	0.1
Mixup α	0.2
CutMix α	1.0
Mixup/CutMix probability	1.0
Mixup-to-CutMix switch prob.	0.5
Exit-loss weights	all 1.0
Classifier dropout	0.2
Gradient clipping	none
Exponential moving average	0.99998
EMA update interval	32 steps
Mixed precision / memory format	yes / channels-last
EfficientNetV2 stochastic depth	0.2[†]

[†]MobileNetV3 uses no stochastic depth. For EfficientNetV2, the nominal maximum DropPath probability is 0.2 and increases linearly with block depth; it is set to zero in feature-fusion blocks.

A.3 Hardware Measurement

We evaluate a Raspberry Pi 4 CPU using ExecuTorch/XNNPACK and an NVIDIA Jetson AGX Xavier GPU using TensorRT. All measurements use batch size one and FP32 inference. Each exit is compiled separately as a marginal computation, with prior-exit state cached.

Raspberry Pi 4. We use four CPU threads. For each marginal exit, we perform five untimed warm-up executions followed by 20 wall-clock timed executions using `perf_counter_ns`, repeated in three fresh processes. A run is rejected if current thermal throttling is detected. We average the 20 timings within each process, then report the arithmetic mean of the three process means; variability is their sample standard deviation.

Jetson AGX Xavier. Each marginal exit has its own TensorRT engine. We perform three fresh `trtexec` runs, reloading the engine for each run, warming up for 10 seconds, and measuring continuously for 15 seconds with spin-wait enabled. We report the arithmetic mean of TensorRT's

three mean GPU-compute latencies, with variability measured by their sample standard deviation. These measurements report GPU-compute time, not host or end-to-end latency.

Cumulative exit latency. We sum the marginal means within corresponding measurement rounds, then average across the three rounds. If $\mu_i^{(q)}$ is the mean marginal latency of exit i in round q, the reported cumulative latency is

$$t_k = \frac{1}{3}\sum_{q=1}^{3}\left(\sum_{i=1}^{k}\mu_i^{(q)}\right). \tag{6}$$

Accuracy–latency curves weight these cumulative latencies by validation-set exit frequencies: $\sum_k \pi_k t_k$, where π_k is the fraction of validation images terminating at exit k. Thus, the curves are constructed from measured marginal computations rather than directly timing complete adaptive execution.

A.4 Analysis Experiments

Feature-reuse ablations. The variants in Section 4.5 use the matched MobileNetV3 model and the same training procedure:

- **Classifier-only** removes stage-wise conditioning while retaining classifier feature reuse.
- **Matched-depth** replaces earlier subnetworks' stage-end features with features at the corresponding depth, keeping the fusion locations and mechanism unchanged. At the first block receiving an input at a reused resolution, this is the earlier subnetwork's feature after downsampling into that resolution.
- **Fused-shortcut** uses the fused feature in both the residual and shortcut branches, computing $u_{k,r}^{+} = \widetilde{u}_{k,r} + \mathcal{T}_{k,r}(\widetilde{u}_{k,r})$. The proposed design instead preserves the unfused local state on the shortcut, as in Equation 3.

Probe features and data. For the classifier-only, matched-depth, and proposed models, we extract the 960-dimensional final embeddings z_1 and z_2 before classifier feature reuse. Each embedding is the output of its exit's sequence

$$\mathrm{Conv}_{1\times1} \;\to\; \mathrm{BatchNorm} \;\to\; \mathrm{Hardswish} \;\to\; \mathrm{GAP},$$

where GAP denotes global average pooling. We use the full ImageNet-1K training split (1,281,167 images) for probe training and the full validation split (50,000 images) for evaluation, without holding out or subsampling either split. Both splits use identical deterministic preprocessing: resize the shorter side to 256 pixels with bicubic interpolation, center-crop to 224×224, convert to a tensor, and normalize with mean $[0.485, 0.456, 0.406]$ and standard deviation $[0.229, 0.224, 0.225]$. No data augmentation is used, including during probe training.

Features are extracted once with the source network in evaluation mode and gradient recording disabled. BatchNorm uses the checkpoint's fixed running statistics, and dropout and stochastic depth are disabled during extraction. Both probe seeds and all six probe heads reuse the same FP32 feature caches; the source network is not executed or updated during probe training. Training features are shuffled each epoch.

Probe architectures and training. Each probe receives $[z_1, 0]$, $[0, z_2]$, or $[z_1, z_2]$, representing exit 1 alone, exit 2 alone, or their combination. Zero padding keeps the input dimension at 1920 and the probe architecture and parameter count identical across conditions. Table 7 specifies the two probe families. The MLP matches the shape of the native classifier on the concatenated pooled representations. Neither probe applies feature standardization or additional normalization. Both output logits, without an output softmax.

Table 7: Complementarity-probe architectures. All linear layers include biases.

Probe	Architecture
Linear	Linear$(1920, 1000)$
MLP	Linear$(1920, 1280)$ $\to$ Hardswish $\to$ Dropout(0.2) $\to$ Linear$(1280, 1000)$

Linear-layer weights are initialized from $\mathcal{N}(0, 0.01^2)$ and biases to zero. Within each probe family, all three input conditions start from exactly identical weights for a given seed. Both families are trained in FP32 for 100 epochs using SGD with momentum 0.9, no Nesterov acceleration, and batch size 256. We use cross-entropy with label smoothing 0.1 and weight decay 10^{-5} on weights but not biases. The learning rate increases linearly from 0.01 to 0.1 over epochs 1–5; cosine decay starts at 0.1 in epoch 6 and reaches zero at epoch 100. All six heads are trained concurrently on the same batches. Results are averaged over two probe-training seeds.

A.5 Per-Exit Results of the Matched Comparisons

Table 8 reports per-exit FLOPs, cumulative latencies on both hardware platforms, and ImageNet top-1 accuracies. The MobileNetV3 pair uses MobileNetV3-based models for both methods; the second pair compares our EfficientNetV2 model with RegNet-based DP.

Table 8: Per-exit compute, latency, and ImageNet accuracy for the approximately endpoint-matched comparisons. F denotes FLOPs (M), P denotes Raspberry Pi 4 CPU latency (ms), X denotes Xavier GPU latency (ms), and A denotes top-1 accuracy (%). Latencies are cumulative through the selected exit.

		Exit 1				Exit 2				Exit 3				Exit 4			
Pair	Model	F	P	X	A	F	P	X	A	F	P	X	A	F	P	X	A
MBV3	Ours	207.1	37.3	3.0	73.4	365.8	71.3	6.4	77.7	528.7	108.6	9.8	78.7	695.9	148.9	13.4	79.2
	DP	225.3	55.0	6.1	64.9	452.1	84.5	9.8	78.0	518.4	98.8	10.7	77.4	682.5	147.7	11.7	79.2
EffV2/RegNet	Ours	529.1	68.5	4.1	74.9	876.4	122.4	7.6	78.9	1229.9	181.6	11.2	80.4	1589.7	243.3	15.1	80.6
	DP	579.0	152.5	8.7	68.8	1315.0	223.4	14.6	79.7	1478.0	252.7	15.7	78.3	1549.0	271.9	16.3	80.4