Multi-Exit Semantic Segmentation Networks

Alexandros Kouris[1], Stylianos I. Venieris*[,1], Stefanos Laskaridis*[,1], Nicholas Lane[1,2]

[1]Samsung AI Center, Cambridge [2]University of Cambridge *Indicates equal contribution.
{a.kouris,s.venieris,stefanos.l,nic.lane}@samsung.com

Abstract. Semantic segmentation arises as the backbone of many vision systems, spanning from self-driving cars and robot navigation to augmented reality and teleconferencing. Frequently operating under stringent latency constraints within a limited resource envelope, optimising for efficient execution becomes important. At the same time, the heterogeneous capabilities of the target platforms and the diverse constraints of different applications require the design and training of multiple target-specific segmentation models, leading to excessive maintenance costs. To this end, we propose a framework for converting state-of-the-art segmentation CNNs to Multi-Exit Semantic Segmentation (MESS) networks: specially trained models that employ parametrised early exits along their depth to *i)* dynamically save computation during inference on easier samples and *ii)* save training and maintenance cost by offering a post-training customisable speed-accuracy trade-off. Designing and training such networks naively can hurt performance. Thus, we propose a novel two-staged training scheme for multi-exit networks. Furthermore, the parametrisation of MESS enables co-optimising the number, placement and architecture of the attached segmentation heads along with the exit policy, upon deployment via exhaustive search in $<$1GPUh. This allows MESS to rapidly adapt to the device capabilities and application requirements for each target use-case, offering a train-once-deploy-everywhere solution. MESS variants achieve latency gains of up to 2.83$\times$ with the same accuracy, or 5.33 pp higher accuracy for the same computational budget, compared to the original backbone network. Lastly, MESS delivers orders of magnitude faster architectural customisation, compared to state-of-the-art techniques.

1 Introduction

Semantic segmentation constitutes a core machine vision task that has demonstrated tremendous advancement due to the emergence of deep learning [15]. By predicting dense (every-pixel) semantic labels for an image of arbitrary resolution, semantic segmentation forms one of the finest-grained visual scene understanding tasks, materialised as an enabling technology for myriad applications, including augmented reality [36,63], video conferencing [68,45], navigation [50,61], and semantic mapping [41].

This wide adoption of segmentation models in consumer applications has pushed their deployment away from the cloud, towards resource-constrained edge devices [22,63] such as smartphones and home robots. With quality-of-service (QoS) and safety being of utmost importance when deploying such real-time systems, efficient and accurate segmentation becomes a core problem to solve. Additionally, device heterogeneity in the consumer ecosystem (*e.g.* co-existence of top-tier and low-cost smartphones) and

the diverse constraints of different applications (*e.g. 30 fps* for AR/VR vs *1 fps* for photo effects), call for segmentation models with variable latency-accuracy characteristics to be designed, trained and distributed to end devices, leading to high maintenance costs.

State-of-the-art segmentation models, however, pose their own challenges to efficient deployment and adaptation, as their impressive accuracy comes at the cost of excessive computational and memory demands. Particularly, the every-pixel nature of the segmentation output calls for high-resolution feature maps to be preserved throughout the network (to avoid eradicating spatial information) [66], while also maintaining a large receptive field on the output (to incorporate context and extract robust semantics) [46], leading to inflated training and inference costs.

Aiming to alleviate this latency burden for on-device inference [1], recent work has focused on the design of lightweight segmentation models either manually [42,72] or through Neural Architecture Search [35,43]. However, such methods typically involve huge search spaces and disallow the re-use of ImageNet [9] pre-trained classification backbones. This leads to long and non-reusable training cycles per model, which often differ for each target device, aggravating prohibitively the training and adaptation time.

Orthogonally, advances in early-exit DNNs offer complementary efficiency gains by adjusting the computation path at run time in an input-dependent manner, while natively providing a tunable speed-accuracy trade-off. However, these solutions [20,24,74] have mainly aimed at image classification so far, leaving challenges in segmentation, such as the design of lightweight exit architectures and exit policies, largely unaddressed. In fact, naively applying early-exiting on segmentation CNNs may not lead to any latency gains due to the inherently heavyweight architecture of segmentation heads, aggravated by the large incoming feature volume. For example, adding a single extra head on DeepLabV3 [6] leads to an overhead of up to 40% of the original model's workload.

In this work, we introduce a novel methodology for deriving and training Multi-Exit Semantic Segmentation (MESS) networks starting from existing CNNs and aiming for efficient and versatile on-device segmentation tailored to the platform and task at hand. MESS brings together architecture customisation and early-exit networks, through a novel training scheme and a compact and highly re-usable search space that allows post-training adaptation through exhaustive search in abridged time frames.

Specifically, MESS uses a given segmentation CNN as a backbone model, pre-trains it in an *early-exit aware* manner (without loss of accuracy) and attaches numerous candidate early-exit architectures (*i.e.* segmentation heads) at different depths, offering predictions with varying workload-accuracy characteristics (Fig. 1). Importantly, through targeted design choices, the *number*, *placement* and *architecture* of exits remain configurable and can be co-optimised upon deployment, to adapt to different-capability devices and diverse application requirements, without the need of retraining, leading to a *train-once, deploy-everywhere* paradigm. The main contributions of this work are:

- The design of MESS networks, combining adaptive inference through early exiting with architecture customisation, to provide a fine-grain speed-accuracy trade-off, tailor-made for semantic segmentation. *This enables efficient and adaptive segmentation based on the use-case requirements and the target device capabilities.*
- A two-stage scheme for training MESS networks, starting with an end-to-end exit-aware pre-training of the backbone that employs a novel exit-dropout loss which

pushes the extraction of semantically strong features towards shallow layers of the network without compromising its final accuracy or committing to an exit configuration; followed by a frozen-backbone stage that jointly trains all candidate early-exit architectures through a novel selective distillation scheme. *This mechanism boosts the accuracy of multi-exit networks and decouples training from the deployed MESS configuration, thus enabling rapid post-training adaptation of the architecture.*

– An input-dependent inference pipeline for MESS networks, employing a novel method for estimating the prediction confidence at each exit, used as exit policy, tailored for every-pixel outputs. *This mechanism enables difficulty-based allocation of resources, by early-stopping for "easy" inputs with corresponding performance gains.*

2 Related Work

Efficient Segmentation. Semantic segmentation is rapidly evolving, since the emergence of the first CNN-based approaches [38,2,44,48]. Recent advances have focused on optimising accuracy through stronger backbone CNNs [17,21], dilated convolutions [66,5], multi-scale processing [65,73], multi-path refinement [32,14], knowledge distillation [37] and adversarial training [40]. To reduce the computational cost, the design of lightweight hand-crafted [42,57,72,64] and more recently NAS-crafted [43,35,69] architectures has been explored. MESS is *model-agnostic* and can follow the above advancements by being *applied on top of existent CNN backbones* to achieve complementary gains by exploiting the orthogonal dimension of input-dependent early-exiting.

Adaptive Inference. The key paradigm behind adaptive inference is to save computation on "easy" samples and reduce the overall computation with minimal accuracy degradation [3,12]. Existing methods can be taxonomised into: *1) Dynamic Routing networks* selecting a different sequence of operations to run in an input-dependent manner by skipping layers [53,55,58] or channels [33,19,11,13,56]; and *2) Multi-Exit Networks* forming a class of architectures with intermediate classifiers along their depth [52,20,27,59,60,67]. With earlier exits running faster and deeper ones being more accurate, such networks provide varying accuracy-cost trade-offs. Existing work has mainly focused on image classification, proposing hand-crafted [20,71], model-agnostic [52,24] and deployment-aware architectures [26,27]. Yet, adopting these techniques in segmentation poses additional, still unexplored, challenges.

Multi-Exit Network Training. So far, the training of multi-exit models for classification can be categorised into: *1) End-to-end* schemes jointly training the backbone and early exits [20,24,71], leading to increased accuracy in early exits, at the expense of often downgrading the accuracy deeper on or even causing divergence [20,29] due to early-exit "cross-talk"; and *2) Frozen-backbone* methods which firstly train the backbone until convergence and subsequently attach and train intermediate exits individually [24,27]. This decoupling of the backbone from the exits allows for faster training of the exits, at the expense of an accuracy penalty due to fewer degrees of freedom in parameter tuning. Orthogonally, self-distillation methods have been proposed in the literature [29,47,70,23,28] to further improve the accuracy of early exits by treating them as students of the last exit. In this work, we propose a fused two-stage training scheme, backed by self-distillation with information filtering, that enables exit-aware pre-training and full customisation potential without affecting the final exit's accuracy.

Adaptive Segmentation Networks. Recently, initial efforts on adaptive segmentation have emerged. Li *et al.* [31] combined NAS with a trainable dynamic routing mechanism that generates data-dependent processing paths at run time. NAS approaches, however, compose enormous search spaces with minimum re-use between instances, leading to soaring training times. Furthermore, by incorporating the computation cost to the loss function, this approach is unable to customise the model to meet varying speed-accuracy characteristics without retraining, leading also to inflated adaptation cost. Closer to our work, Layer Cascade (LC) [30] studies *early-stopping* for segmentation. LC treats segmentation as a vast group of independent classification tasks, where *each pixel* propagates to the next exit only if the latest prediction does not surpass a confidence threshold. Nonetheless, due to different *per-pixel* paths, this scheme leads to heavily unstructured computations, for which existing BLAS libraries cannot achieve realistic speedups [62]. LC also constitutes a manually-crafted model, tied to a specific backbone architecture, and non-customisable to the target device's capabilities.

MESS networks bring together benefits of all the above worlds. Our framework supports model customisation within a compact search space of early-exit architectures tailor-made for semantic segmentation, while preserving the ability to re-use pre-trained backbones cutting down training time. Additionally, MESS networks push the limits of efficient inference by incorporating *image-level* confidence-based early exiting, through a novel exit policy that addresses the unique challenges of dense segmentation predictions. Simultaneously, the proposed two-stage training scheme combines end-to-end and frozen-backbone training approaches, boosting the accuracy of shallow exits without compromising deeper ones. Finally, design choices allow us to decouple MESS training from the deployment configuration, enabling exhaustive search to be rapidly performed post-training, in order to customise the architectural configuration for different devices or application-specific requirements, without any parameter fine-tuning.

3 MESS Networks Overview

To enable efficient segmentation, the MESS *framework* employs a target-specific configuration search to obtain a *multi-exit* segmentation network optimised for the platform and task at hand. We call the resulting model a MESS network, with an example depicted in Fig. 1. Constructing a MESS network involves three stages: *i)* starting from a backbone segmentation CNN, we identify several candidate *exit points* along its depth (Sec. 4.1), and *attach* to each of them multiple *early exits* (*i.e.* segmentation heads) of varying architectural configurations (Sec. 4.2), leading to a newly defined *overprovisioned* network; *ii) training* all candidate exits together with the backbone through a novel two-stage scheme (Sec. 4.3); and *iii) tailoring* the overprovisioned network post-training to extract a MESS *instance*, comprising the backbone and a subset of the available exits, considering user-defined constraints and optimisation objectives (Sec. 5.1). Our framework supports various inference settings, ranging from extracting efficient target-specific submodels (meeting accuracy/speed constraints) to progressive refinement of the segmentation prediction and confidence-based exiting (Sec. 5.2). Across all settings, MESS networks save computation by circumventing deeper parts of the network. The next two sections follow the flow of the proposed framework.

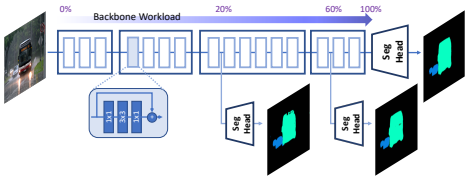

Fig. 1: Multi-Exit Semantic Segmentation network instance.

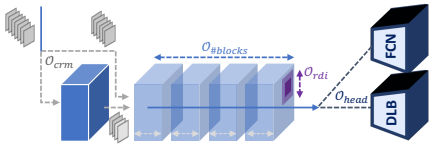

Fig. 2: Parametrisation of segmentation head architecture.

4 MESS Networks Design & Training

In this section, we go through the design choices that shape MESS networks, their early-exit architectural configuration and training process. This yields an overprovisioned network, ready to be customised for the target application and device at hand.

4.1 Backbone Initialisation & Exit Placement

Initially, a backbone segmentation CNN is provided. Typically, such models aim to preserve large receptive field on the output, while preventing loss of spatial information (*e.g.* by replacing traditional pooling operations with dilated convolutions [66]). As a result, and combined with the increased number of channels integrated, deeper layers demonstrate significantly larger feature volumes, leading to an unbalanced distribution of computational demands across the network (Fig. 1). This motivates the adoption of early-exiting during inference as a means of improving processing speed.

As a first step, the provided backbone is profiled in terms of per-layer workload (FLOPs). Based on the results of this analysis, N *candidate exit points* are identified following an approximately equidistant workload distribution (every $1/N$-th of the total backbone's FLOPs). For simplicity, exit points are restricted to be at the output of individual network blocks[1] b_k. Although some of these exit points may subsequently be dropped during MESS configuration search, this placement currently maximises the distance between subsequent exits, improving the efficiency of our search. An example of the described analysis on a DRN-50 backbone [66] is presented in Fig. 3.

4.2 Early-Exit Architecture Search Space Design

Early-exiting in segmentation CNNs faces the challenge of: *i) enlarged feature volumes* of segmentation models, leading to inflated computation cost for the early-exit heads, *ii) limited receptive field* and *iii) weak semantics* in shallow exits. MESS addresses these challenges in a two-fold manner: *i)* by pushing the extraction of semantically strong features to shallower layers of the backbone during training (Sec. 4.3) and *ii)* by introducing a configuration space tailored-made for segmentation head architectures:

[1] *e.g.* Dilated Residual Blocks for ResNet-based [17] backbones, Inverted Residual Blocks for MobileNet-based [49] backbones etc.

1) Channel Reduction Module (CRM): To reduce the computational overhead of each exit without compromising the spatial resolution of the feature volume that is particularly important for accuracy, we optionally include a 1×1 convolutional layer (CRM) that reduces the number of channels fed to the segmentation head by a tunable factor.
2) Extra Trainable Blocks: To address the weak semantics of shallow exits, while avoiding an unnecessary surge in the computational overhead of deeper exits, we allow incorporating a configurable number of additional convolutional blocks in each exit's architecture. These layers are tactically appended *after* the CRM to take advantage of the computational efficiency of its reduced feature-volume width.
3) Rapid Dilation Increase (RDI): To address the limited receptive field of shallow exits, apart from the addition of trainable blocks, we optionally allow the dilation rate employed in the exit layers to be rapidly increased, doubling in each block.
4) Head: MESS currently supports two types of output segmentation blocks from the literature, positioned at the end of each exit: *i)* Fully Convolutional Network-based Head (*FCN-Head*) [38] and *ii)* DeepLabV3-based Head (*DLB-Head*) [6,7].

Overall, the configuration space for each exit architecture (Fig. 2) is shaped as:

1. Channel Reduction Module: $\mathcal{O}_{\text{crm}} = \{/1, /2, /4, /8\} \mapsto \{0, 1, 2, 3\}$
2. Extra Trainable Blocks: $\mathcal{O}_{\text{\#blocks}} = \{0, 1, 2, 3\}$
3. Rapid Dilation Increase: $\mathcal{O}_{\text{rdi}} = \{\textit{False}, \textit{True}\} \mapsto \{0, 1\}$
4. Segmentation Head: $\mathcal{O}_{\text{head}} = \{\textit{FCN-Head}, \textit{DLB-Head}\} \mapsto \{0, 1\}$

Expecting that varying exit-point depths favour different architectural configurations (*e.g.* channel-rich for deeper exits and layer-multitudinous for shallower), MESS allows each early exit to adopt a tailored architecture based on its position in the backbone. Formally, we represent the configuration space for the i-th exit's architecture as:

$$\mathcal{S}^{i}_{\text{exit}} = \mathcal{O}_{\text{crm}} \times \mathcal{O}_{\text{\#blocks}} \times \mathcal{O}_{\text{rdi}} \times \mathcal{O}_{\text{head}} \tag{1}$$

where $i \in \{1, 2, ..., N\}$ and $\mathcal{O}_{\text{crm}}$, $\mathcal{O}_{\text{\#blocks}}$, $\mathcal{O}_{\text{rdi}}$ and $\mathcal{O}_{\text{head}}$, are the sets of available *options* for the CRM, number of trainable blocks, RDI and exit head, respectively.

4.3 Training Scheme

Two-Stage MESS Training. As aforementioned, early-exit networks are typically either trained *end-to-end* or in a *frozen-backbone* manner [25]. However, both can lead to suboptimal accuracy results in the final or the early exits. For this reason, we combine the best of both worlds by proposing a novel two-stage training scheme.
Stage 1 (end-to-end): In the exit-aware pre-training stage, we aim to fully train the backbone network that will be shared across all candidate exits, specially preparing it for early-exiting by pushing the extraction of semantically strong features at shallow layers, without committing to any particular exit configuration. To achieve this, vanilla FCN-Heads are attached to all candidate exit points, generating an intermediate multi-exit model. This network is trained end-to-end, updating the weights of the backbone and a *single* early exit at each iteration, with the remainder of the exits being dropped-out in a round-robin fashion (Eq. (2), referred to as *exit-dropout loss*). As a result, cross-talk between exits is minimised allowing the final head to reach its full potential, while

the backbone remains exposed to gradients from shallower exits. Formally, we denote the segmentation predictions after softmax for each early exit by $\boldsymbol{y}_i \in [0,1]^{R\times C\times M}$ where R and C are the output's number of rows and columns, respectively, and M the number of classes. Given the ground truth labels $\hat{\boldsymbol{y}} \in \{0,1,...,M\text{-}1\}^{R\times C}$, the loss function for the proposed exit-aware pre-training stage is formulated as:

$$\mathcal{L}_{\text{pretrain}}^{\text{batch}^{(j)}} = \sum_{i=1}^{N-1} \mathbb{1}(j \bmod i = 0) \cdot \mathcal{L}_{\text{CE}}(\boldsymbol{y}_i, \hat{\boldsymbol{y}}) + \mathcal{L}_{\text{CE}}(\boldsymbol{y}_N, \hat{\boldsymbol{y}}) \tag{2}$$

where $\mathbb{1}(\cdot)$ is the indicator function and $\mathcal{L}_{\text{CE}}$ the cross entropy. Although after this stage the early exits are not fully trained, their contribution to the loss *guides the backbone towards learning stronger representations throughout, consequently aiding early-exiting.*

Stage 2 (frozen-backbone): At this stage, the backbone and final exit are kept frozen (*i.e.* weights are not updated). The *MESS overprovisioned network* is formed by attaching *all* candidate early-exit architectures of the proposed configuration space $\mathcal{S}_{\text{exit}}^{i}$ (Sec. 4.2) across *all* candidate exit points $i \in \{1,2,...,N\}$ (Sec. 4.1) and training them jointly. Importantly, keeping the backbone unchanged during this stage allows different exit architectures to be: *i)* attached and trained *simultaneously* even to the same candidate exit point *without interfering* with each other, or with the backbone *ii)* trained at significantly reduced cost than the end-to-end approach, while taking advantage of the strong semantics extracted by the backbone due to its early-aware pre-training and *iii)* interchanged at deployment time on top of the shared backbone in a *plug-and-play* manner (without re-training), offering enormous flexibility for customisation (Sec. 5.1).

Positive Filtering Distillation (PFD). In the second stage of our training process, we also exploit the joint potential of knowledge distillation and early-exit networks.

In prior self-distillation works for multi-exit networks, the backbone's final output is used as the teacher for earlier classifiers [70], whose loss function typically combines ground-truth and distillation terms [47,39]. To further exploit what information is back-propagated to the shallow exits, given the pre-trained final exit and taking advantage of the multitude of information available in segmentation predictions due to their dense structure, we propose *Positive Filtering Distillation* (PFD). This technique selectively controls the flow of information of the high-entropy ground-truth reference to earlier exits using only signals from "easier pixels", *i.e.* pixels about which the last exit could yield a correct prediction, while filtering out gradients from more difficult or ambiguous pixels. Our hypothesis is that early-exit heads, having limited learning capacity, can become stronger by only incorporating signals of less ambiguous pixels from the last exit, avoiding noisy gradients and the confusion of trying to mimic contradicting references.

Formally, we express the i-th exit's tensor of predicted classes for each pixel $\boldsymbol{p} = (r,c)$ with $r\in[1,R]$ and $c\in[1,C]$ as $\hat{\boldsymbol{y}}_i\in\{0,1,...,M\text{-}1\}^{R\times C}$ where $(\hat{\boldsymbol{y}}_i)_{\boldsymbol{p}} = \arg\max(\boldsymbol{y}_i)_{\boldsymbol{p}}$ in $\{0,1,...,M\text{-}1\}$. Given the corresponding output of the final exit $\hat{\boldsymbol{y}}_N$, the ground-truth labels $\hat{\boldsymbol{y}}\in\{0,1,...,M\text{-}1\}^{R\times C}$ and a hyperparameter α, we employ the following loss during the frozen-backbone stage of our training scheme, where $\mathcal{L}_{\text{KL}}$ is KL-divergence:

$$\mathcal{L}_{\text{PFD}} = \sum_{i=1}^{N} \alpha \cdot \mathbb{1}(\hat{\boldsymbol{y}}_N = \hat{\boldsymbol{y}}) \mathcal{L}_{\text{CE}}(\boldsymbol{y}_i, \hat{\boldsymbol{y}}) + (1-\alpha) \cdot \mathcal{L}_{\text{KL}}(\boldsymbol{y}_i, \boldsymbol{y}_N) \tag{3}$$

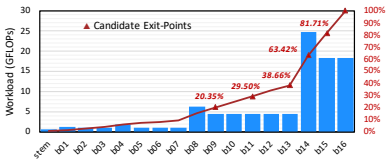

Fig. 3: Workload breakdown analysis and exit points identification on a DRN-50 [66] backbone (N=6).

Table 1: Cost functions for different inference settings. b_i is the i-th block in the backbone; K_n is the block ordinal of the n-th exit point; $\mathcal{S}^{*n}_{\text{exit}} \in \mathcal{S}^{n}_{\text{exit}}$ is the selected architecture for the n-th exit; p_n is the percentage of samples propagated to the n-th exit.

Inference	$cost(s)$
Final-Only	$cost(b_{1:K_N}) + cost(\mathcal{S}^{*N}_{\text{exit}})$
Budgeted	$cost(b_{1:K_n}) + cost(\mathcal{S}^{*n}_{\text{exit}})$, $n \leq N$
Anytime	$cost(b_{1:K_N}) + \sum_{n=1}^{N} cost(\mathcal{S}^{*n}_{\text{exit}})$
Input-Dep.	$\sum_{n=1}^{N} p_{n-1} \cdot (cost(b_{K_{n-1}:K_n}) + cost(\mathcal{S}^{*n}_{\text{exit}}))$, K_0=0 , p_0=1

5 MESS Networks Deployment & Inference

Having designed and trained the *overprovisioned* model, here we discuss its customisation to the task- and target-specific deployment for inference. This involves configuring the MESS *instance* architecture via post-training search and crafting the exit policy.

5.1 Deployment-time Parametrisation

Post-training of the overprovisioned network (comprising *all* candidate exit architectures), MESS instances (comprising a *subset* of the trained exits) can be derived, reflecting on the capabilities of the target device, the required accuracy or latency of the use-case and the intricacy of the inputs.

Inference Settings. To satisfy performance needs under each device and application-specific constraints, MESS networks support different inference settings:

***1) Budgeted Inference*:** in which workload-lighter static submodels, up to a (single) specific exit, are extracted to enable deployment on heterogeneous target platforms.

***2) Anytime Inference*:** in which every sample goes through multiple exits sequentially, initially providing a rapid approximation of the output and progressively refining it through a series of deeper exits until a deadline is met.

***3) Input-dependent Inference*:** where inputs also go through exits sequentially, but each sample dynamically adjusts its path (*i.e.* finalises its output at a different depth) according to its difficulty, as captured by the confidence of each exit's prediction.

Configuration Search. Our framework tailors MESS networks for each of the above settings considering the target use-case, by searching the configuration space post-training. Contrary to most works in multi-exit classification models [20,24,71,27], which employ a uniform architecture across all exits for the sake of simplicity, MESS favours flexibility allowing for per-exit architectural customisation. This is enabled by our overprovisioned training scheme (Sec. 4.3), allowing all trained exits to be interchangeably attached to the same backbone for inference, offering rapid validation of candidate choices that significantly accelerates the search for a tailored design.

The proposed method contemplates all trained exit architectures and exhaustively creates different configurations, trading for example a workload-heavier shallow exit with a more lightweight deeper exit. The search strategy considers the target *inference setting*, along with user-specified requirements in *workload*, *latency* and *accuracy*[2], which can be expressed as a combination of hard constraints and optimisation objectives. As a result, the *number* and *placement* of exits and the *architecture* of each

[2]Evaluated in a held-out *Calibration Set* during search (equally sized to the target *Test Set*).

individual exit of the resulting MESS instance are jointly optimised (along with the *exit policy*, discussed in Sec. 5.2, for the input-dependent inference case).

Given the exit-architecture search space $\mathcal{S}^i_{\text{exit}}$ (Eq. 1), we define the overall configuration space of a MESS network as:

$$\mathcal{S} = (\mathcal{S}^1_{\text{exit}} + 1) \times (\mathcal{S}^2_{\text{exit}} + 1) \times ... \times (\mathcal{S}^N_{\text{exit}} + 1) \tag{4}$$

where the extra term accounts for a "*None*" option for each of the exit positions. Under this formulation, the framework can minimise $workload/latency$, formally expressed as $cost$ for each setting in Table 1, given an $accuracy$ constraint th_{acc}:

$$s^\star = \arg\min_{s \in \mathcal{S}} \{\text{cost}(s) \mid \text{acc}(s) \geq th_{\text{acc}}\} \tag{5}$$

or optimise for $accuracy$, given a $cost$ constraint th_{cost}:

$$s^\star = \arg\max_{s \in \mathcal{S}} \{\text{acc}(s) \mid \text{cost}(s) \leq th_{\text{cost}}\} \tag{6}$$

Importantly, a combination of design choices render the *exhaustive exploration* of the search space not only computationally tractable, but extremely efficient. Conversely to heuristic alternatives, this guarantees *optimality* within the examined space. The main enabling factors include: *i)* the informed outlining of the search space (being compact and tailor-made for segmentation), *ii)* the proposed two-stage training scheme (allowing all exits architectures to exploit a shared backbone), *iii)* a vast pruning of configurations at search time (prioritising the less costly constraint verification on latency before evaluating accuracy), and *iv)* prediction memoisation (eliminating duplicate inference execution by storing and combining per-exit predictions on the calibration set). Finally, in contrast to NAS methods [4], MESS overprovisioned networks are fully trained and can be customised without the need of fine-tuning, offering rapid post-training adaptation.

5.2 Input-Dependent Exit Policy

During input-dependent inference, each input image goes through the selected early exits of the deployed MESS instance sequentially. After a prediction is produced from an exit, a mechanism to calculate its confidence is used to determine whether inference should continue to the next exit or not. In [30], *each pixel* in an image is treated as an independent classification task, exiting early if its prediction confidence in an exit is high, thus yielding irregular computation paths. In contrast, our approach treats the segmentation of *each image* as a single task, aiming to drive each sample through a uniform computation route. To this end, we fill a gap in literature by introducing a novel mechanism to quantify the overall confidence in semantic segmentation predictions.

Confidence Metric. Given the per-pixel confidence map, calculated from the probability distribution across classes of each pixel $\boldsymbol{c}^{\text{map}} = f_c(\boldsymbol{y}) \in [0,1]^{R \times C}$ (where f_c is usually $top1(\cdot)$ [20] or $entropy(\cdot)$ [8]), we introduce a mechanism to reduce these *every-pixel* confidence values to a single *per-image* confidence. The proposed metric considers the *percentage of pixels with high prediction confidence* (i.e. surpassing a tunable threshold th^{pix}_i) in the output of an exit $\boldsymbol{y}_i$:

$$c^{\text{img}}_i = \frac{1}{RC} \sum_{r=1}^{R} \sum_{c=1}^{C} \mathbb{1}(\boldsymbol{c}^{\text{map}}_{r,c}(\boldsymbol{y}_i) \geq th^{\text{pix}}_i) \tag{7}$$

Edge Confidence Enhancement. Moreover, it has been observed that due to the progressive downsampling of the feature volume in CNNs, some spatial information is unavoidably lost. As a result, semantic predictions near object edges are naturally underconfident [54]. Driven by this observation, we enhance our proposed metric to account for these expected low-confidence pixel-predictions, by introducing a pre-processing step for c_i^{img}. Initially, we conduct edge detection on the semantic masks, followed by an erosion filter with kernel equal to the output stride of the respective exit os_i, in order to compute a semantic-edge map (Eq.8). Thereafter, we apply a median-based smoothing on the confidence values of pixels lying on the semantic edges (Eq.9).

$$\mathcal{M} = \text{erode}(\text{cannyEdge}(\hat{\boldsymbol{y}}_i), s_i) \tag{8}$$

$$\widehat{\boldsymbol{c}_{r,c}^{\text{map}}}(y_i) = \begin{cases} \text{median}(\boldsymbol{c}_{w_r,w_c}^{\text{map}}(\boldsymbol{y}_i)) & \text{if } \mathcal{M}_{r,c} = 1 \\ \boldsymbol{c}_{r,c}^{\text{map}}(\boldsymbol{y}_i) & \text{otherwise} \end{cases} \tag{9}$$

where $w_l = \{l - 2 \cdot os_i, ..., l + 2 \cdot os_i\}$ is the window size of the filter. *This sets the pixels around semantic edges to inherit the confidence of their neighbouring pixel predictions*.
Exit Policy. At inference time, each sample is sequentially processed by the selected early exits. For each prediction $\boldsymbol{y}_i$, the proposed metric c_i^{img} is calculated, and a tunable confidence threshold (exposed to the search space) determines whether the sample will exit early ($c_i^{\text{img}} \geq th_i^{\text{img}}$) or be processed further by subsequent backbone layers/exits.

6 Evaluation

6.1 Experimental Setup

Models & Datasets. We apply our methodology on top of DRN-50 [66], DeepLabV3 [6] and SegMBNetV2 [49] segmentation CNNs, using ImageNet [9] pre-trained ResNet50 [17] and MobileNetV2 [49] backbones, representing *high-end* and *edge* use-cases, respectively. We train all backbones on MS COCO [34] and fine-tune early exits on MS COCO and PASCAL VOC [10] (augmented from [16]) independently.
Development & Deployment Setup. MESS networks are implemented on *PyTorch (v1.6.0)*. For inference, we deploy MESS instances on a *high-end* (Nvidia GTX1080Ti; 250W TDP) and an *edge* (Nvidia Jetson AGX Xavier; 30W TDP) compute platform.
Baselines. To compare our work against the following state-of-the-art baselines:
1) **DRN** [66]; *2)* **DLBV3** [66,7]; *3)* **segMBNetV2** [49]; *4)* **LC** [30]; *5)* **AutoDLB** [35]; *6)* **E2E** [20,52]; *7)* **Frozen** [24,27]; *8)* **KD** [18] and *9)* **SelfDistill** [71,47,70].

6.2 MESS End-to-End Evaluation

Comparison with Single-Exit Baselines. First, we apply our MESS framework on single-exit segmentation backbones from the literature, namely **DRN**, **DLBV3** and **segMBNetV2**. Table 2 lists the achieved results for MESS instances optimised for varying use-cases (framed as speed/accuracy constraints fed to our configuration search).

Table 2: End-to-end evaluation of MESS network designs

Method		Backbone*	Head	Search Targets		Results: MS COCO			Results: PASCAL VOC		
				Error	GFLOPs	mIoU	GFLOPs	Latency†	mIoU	GFLOPs	Latency†
DRN [66]	(i)	ResNet50	FCN	–Baseline–		59.02%	138.63	39.96ms	72.23%	138.63	39.93ms
Ours	(ii)	ResNet50	FCN	min	$\leq 1\times$	64.35%	113.65	37.53ms	79.09%	113.65	37.59ms
Ours	(iii)	ResNet50	FCN	$\leq 0.1\%$	min	58.91%	41.17	17.92ms	72.16%	44.81	18.63ms
Ours	(iv)	ResNet50	FCN	$\leq 1\%$	min	58.12%	34.53	15.11ms	71.29%	38.51	16.80ms
DLBV3 [6]	(v)	ResNet50	DLB	–Baseline–		64.94%	163.86	59.05ms	80.32%	163.86	59.06ms
Ours	(vi)	ResNet50	DLB	min	$\leq 1\times$	65.52%	124.10	43.29ms	82.32%	124.11	43.30ms
Ours	(vii)	ResNet50	DLB	$\leq 0.1\%$	min	64.86%	69.84	24.81ms	80.21%	65.29	24.14ms
Ours	(viii)	ResNet50	DLB	$\leq 1\%$	min	64.03%	57.01	20.83ms	79.30%	50.29	20.11ms
segMBNetV2 [49]	(ix)	MobileNetV2	FCN	–Baseline–		54.24%	8.78	67.04ms	69.68%	8.78	67.06ms
Ours	(x)	MobileNetV2	FCN	min	$\leq 1\times$	57.49%	8.10	56.05ms	74.22%	8.10	56.09ms
Ours	(xi)	MobileNetV2	FCN	$\leq 0.1\%$	min	54.18%	4.05	40.97ms	69.61%	3.92	32.79ms
Ours	(xii)	MobileNetV2	FCN	$\leq 1\%$	min	53.24%	3.48	38.83ms	68.80%	3.60	31.40ms

*Dilated network [66] based on backbone CNN. †Measured on: GTX for ResNet50 and AGX for MobileNetV2 backbone.

Table 3: Comparison with LC (speedup to backbone)

Head	Search Target	Exit Points	Exit Policy	
			LC[30]	Ours
FCN	Error $\leq$ 0.1%	*3-exit:*$\{\mathcal{E}_1, \mathcal{E}_3, \mathcal{E}_6\}$	1.13×	**3.36**×
FCN	Error $\leq$ 10.0%	*2-exit:*$\{\mathcal{E}_1, \mathcal{E}_6\}$	0.98×	**6.02**×

Table 4: Comparison with SOTA NAS approach

Method	Approach	ImgNet	Training*	Adaptation*		mIoU	GFLOPs
				search	re-training		
DLBV3 [6]	Baseline	✓	192	-Non-adaptive-		80.32%	163.86
AutoDLB [35]	NAS	-	12,248	72	12,176	79.78%	57.61
Ours	MESS	✓	2,580	<1	-	79.94%	51.59

*Initial-training and Adaptation times expressed in GPU-hours.

For a DRN-50 backbone on MS COCO, we observe that a latency-optimised MESS instance achieves a 3.36× workload reduction with no accuracy drop (row (iii)), translating to a latency speedup of 2.23× over the single-exit **DRN** (row (i)). This improvement is amplified to 4.01× in workload (2.65× in latency) for use-cases that can tolerate a controlled accuracy degradation of $\leq$1 pp (row (iv)). Additionally, a MESS instance optimised for accuracy under the same workload budget as **DRN**, can achieve an mIoU gain of 5.33 pp compared to **DRN**, with 1.22× fewer GFLOPs (row (ii)).

Similar results are obtained for **DLBV3**, as well as when targeting PASCAL VOC. Moreover, the gains are consistent on **segMBNetV2**, which forms an inherently efficient segmentation model, with 15.7× smaller workload than DRN-50. This demonstrates the model-agnostic nature of our framework, yielding complementary gains to efficient backbone design, by exploiting the orthogonal dimension of input-dependent inference.

Comparison with Multi-Exit Baselines. Next, we compare MESS networks against *Deep Layer Cascade* (**LC**) [30], the current SOTA in multi-exit segmentation, which proposes *per-pixel* early-exiting through multiple segmentation heads. Due to their unstructured computation, standard BLAS libraries cannot realise true latency benefits from this approach. However, we apply **LC**'s pixel-level exit policy on diverse MESS configurations, and compare with our image-level policy analytically (in GFLOPs), tuning both thresholds so as to meet varying accuracy requirements.

By using SOTA techniques for semantic segmentation, such as larger dilation rates or DeepLab's ASPP, the gains of **LC** rapidly fade away, as for each pixel that propagates deeper on, a substantial feature volume needs to be precomputed. Concretely, when employing **LC** on our designs, up to a substantial 45% of the feature volume at the output of the first exit falls within the receptive field of a *single pixel* in the final output for the case of *FCN-Head*, reaching 100% for *DLB-Head*. As a result, **LC**'s policy presents heavily dissipated to no reduction in workload against the corresponding single-exit baselines, being heavily reliant on the exit placement. In contrast, the respective MESS instances equipped with our proposed exit policy (Sec. 5.2) are able to achieve significant workload reduction, reported in Table 3.

Comparison with NAS Baselines. Finally, we position our work against NAS solutions for deriving efficient segmentation models. We employ Auto-DeepLab (**AutoDLB**) [35] as our strong baseline, due to its SOTA performance both in accuracy and search efficiency, and use our framework to generate a MESS instance matching its accuracy (staring from DeepLabV3 [6] backbone). Table 4 lists our findings on PASCAL VOC.

Remarkably, MESS achieves a better (but comparable) speed-accuracy trade-off than **AutoDLB** (3.17× vs 2.85× speedup over DeepLabV3), although the latter samples from a larger space during search (10^{19} points vs 10^6) and takes advantage of more degrees of freedom during training. Additionally, being able to exploit ImageNet pre-trained backbones, MESS demonstrates significant training time savings (4.7× faster) compared to NAS-crafted models like **AutoDLB**, that can only be trained from scratch.

Most importantly, due to our "train-once-deploy-everywhere" design, enabled by the two-stage training approach of MESS, after the initial training of the overprovisioned MESS network *all 10^6 possible MESS instances are ready-to-deploy* without any need for re-training. Alternatively, training end-to-end all 10^6 MESS instances would require >200 million GPU-hours. As a result, different MESS instances can be obtained with a minimal search cost (<1 GPU-hour). Overall, MESS offers up to five orders of magnitude faster adaptation time compared to NAS-based methodologies.

6.3 MESS Training Evaluation

Having shown the benefits of MESS networks against different state-of-the-art methods, we now move to the evaluation of specific components of our framework.

Exit-Aware Pre-training. Initially, we demonstrate the effectiveness of the proposed training scheme. We compare the accuracy of models with uniform exit configuration across all candidate exits points, trained using different strategies. Table 5 summarises the results of this comparison on a DRN-50 backbone with N=6, on MS COCO.

When multiple exits are attached to the backbone and jointly trained end-to-end, as in [20,52], the accuracy of the final exit can notably degrade (row (ii)) compared to a vanilla training of the backbone with solely the final exit attached (row(i)). This is attributed to contradicting gradient signals between the early and the late classifiers and to the larger losses of the early results, which dominate the loss function [3]. On the other hand, freezing the weights of the vanilla backbone of row (i) and independently training the same early exits, as in [24,27], leads to degraded accuracy in shallow exits (row (iv)). This is due to the limited degrees of freedom of this second training stage and the weaker semantics extracted by shallow layers of the frozen backbone.

Our (1st-stage) exit-aware pre-training pushes the extraction of semantically strong features towards shallow parts of the network, while yielding the highest accuracy on the final exit (row (iii)). Similar to observations from [51], we fathom that the extra signal midway through the model acts both as a regulariser and as an extra backpropagation source, reducing the effect of vanishing gradients.

Capitalising on this exit-aware pre-trained backbone, and without any harm of the final exit's accuracy, our subsequent frozen backbone training achieves consistently higher accuracy (up to 12.57pp) across all exits (row (v)) compared to a traditionally pre-trained segmentation network (**Frozen**), and up to 3.38 pp compared to an end-to-end trained model (**E2E**), which also suffers a 1.57 pp accuracy drop in the final exit.

Table 5: Evaluation of two-stage training scheme on DRN-50 (mIoU).

Method	Init.	Loss	$\mathcal{E}_1$	$\mathcal{E}_2$	$\mathcal{E}_3$	$\mathcal{E}_4$	$\mathcal{E}_5$	$\mathcal{E}_6$
(i) Baseline Init.	ImageNet	$\mathcal{L}_{\text{CE}}(\mathcal{E}_6)$	-	-	-	-	-	59.02%
(ii) **E2E** [20,52]	ImageNet	$\mathcal{L}_{\text{CE}}(\mathcal{E}_1)$+...+$\mathcal{L}_{\text{CE}}(\mathcal{E}_6)$	29.02%	40.67%	48.64%	51.69%	55.34%	58.33%
(iii) Exit-aware Init.	ImageNet	Eq. (2) (**Ours**)	28.21%	39.61%	47.13%	50.81%	56.11%	**59.90%**
(iv) **Frozen** [24,27]	(i)	$\mathcal{L}_{\text{CE}}(\mathcal{E}_1)$, ..., $\mathcal{L}_{\text{CE}}(\mathcal{E}_5)$	23.94%	31.50%	38.24%	44.73%	54.32%	59.02%
(v) **Ours** (§4.3)	(iii)	$\mathcal{L}_{\text{CE}}(\mathcal{E}_1)$, ..., $\mathcal{L}_{\text{CE}}(\mathcal{E}_5)$	**32.40%**	**43.34%**	**50.81%**	**53.73%**	**57.9%**	**59.90%**

* Experiments repeated 3 times. The sample stdev in mean IoU is at most ± 0.09 in all cases.

Table 6: Evaluation of Positive Filtering Distillation (mIoU)

Method	Loss	DRN-50			MobileNetV2		
		$\mathcal{E}_1$	$\mathcal{E}_2$	$\mathcal{E}_3$	$\mathcal{E}_1$	$\mathcal{E}_2$	$\mathcal{E}_3$
E2E [20,52]	CE	49.96%	55.40%	58.96%	31.56%	41.57%	51.59%
KD [18]	KD	50.33%	55.67%	59.08%	31.04%	41.93%	51.66%
SelfDistill [71,47,70]	CE+KD	50.66%	55.91%	58.84%	32.08%	41.96%	51.58%
Ours (§4.3)	PFD	**51.02%**	**56.21%**	**59.36%**	**33.36%**	**42.95%**	**52.20%**

CE=Cross-entropy, KD=Knowledge Distillation, PFD=Positive Filtering Distillation

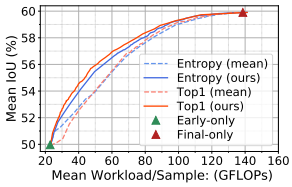

Fig. 4: Comparison of different early-exit policies on a DRN-50-based MESS instance with two exits.

Positive Filtering Distillation. Here, we quantify the benefits of Positive Filtering Distillation (PFD) for the second stage (frozen-backbone) of our training methodology. To this end, we compare against **E2E** utilising cross-entropy loss (CE), traditional knowledge distillation (**KD**), and **SelfDistill** approach commonly used in multi-exit classification. Table 6 summarises our results on a representative exit-architecture, on both DRN-50 and MobileNetV2, across MS COCO validation set.

Our proposed loss consistently yields higher accuracy across all cases, achieving up to 1.8, 2.32 and 1.28 pp accuracy gains over **E2E**, **KD** and **SelfDistill**, respectively. This accuracy boost is more salient on shallow exits, whereas a narrower improvement is obtained in deeper exits where the accuracy gap to the final exit is natively bridged.

6.4 MESS Deployment under Different Settings

In this section, we showcase the effectiveness and flexibility of the proposed *train-once, deploy-everywhere* approach for semantic segmentation. There are three inference settings in MESS networks: *i)* budgeted, *ii)* anytime and *iii)* input-dependent, for which we optimise separately *post-training* (Sec. 5.1). Here, we employ our search to find the best single early-exit architecture for each case, using a 50% mIoU requirement. The results are summarised in Table 7. Fig. 5 also depicts the underlying workload-accuracy relationship across the architectural configuration space for DRN-50 backbone. Different points represent different architectures, colour-coded by their placement in the network.
Budgeted Inference. In this setting, we search for a *single-exit submodel* that can execute within a given latency/memory/accuracy target. Our method is able to provide the most efficient MESS instance, tailored to the requirements of the underlying application and target device (Table 7; row (ii)). This optimality gets translated in Fig. 5a, showcasing the cost-accuracy trade-off from the input until the respective early exit of the network, in the presence of candidate design points along the Pareto front of the search space. In this setting, our search tends to favour designs with powerful exit architectures, consisting of multiple trainable layers, mounted earlier in the network (Fig. 6a).
Anytime Inference. In this setting, each sample is sequentially processed by multiple exits, progressively refining its prediction. When a deadline is met or a result is needed, the last available output of the multi-exit network is asynchronously returned. This paradigm creates an inherent trade-off: denser exits provide more frequent "checkpoints", whereas each added head adds computational overhead when not explicitly used. To control this trade-off, our method considers the additional computational cost

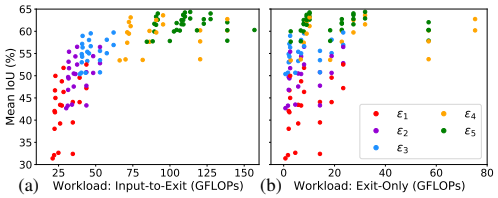

Fig. 5: Workload-accuracy trade-off between design points. Capturing: (a) input-to-exit workload, (b) the overhead of each exit.

Table 7: DRN-50 with one early exit optimised for different inference schemes (Requirement of 50% mIoU)

	Inference	Workload (GFLOPs) Overhead	$\mathcal{E}_{\text{early}}$	$\mathcal{E}_{\text{final}}$	mIoU $\mathcal{E}_{\text{early}}$	$\mathcal{E}_{\text{final}}$
(i)	Final-Only	-	-	138.63	-	59.90%
(ii)	Budgeted	8.01	28.34	-	51.76%	-
(iii)	Anytime	0.69	39.32	139.33	50.37%	59.90%
(iv)	Input-Dep.	2.54	($\mathcal{E}_{\text{sel}}$: 23.02)		($\mathcal{E}_{\text{sel}}$: 50.03%)	

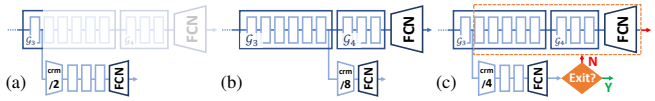

Fig. 6: Selected design points for: (a) budgeted, (b) anytime and (c) input-dependent inference, with the same accuracy target.

of each exit, when populating the MESS network architecture (Fig. 5b). Contrary to budgeted inference, in this setting our search produces heads with extremely lightweight architecture, sacrificing flexibility for reduced computational overhead, mounted deeper in the network (Fig. 6b). Table 7 showcases that, for anytime inference (row (iii)), our search yields an exit architecture with 11.6× less computational requirements compared to budgeted inference (row (ii)), under the same accuracy constraint.

Input-Dependent Inference. In this setting, each input sample propagates through the selected MESS instance until the model yields a confident-enough prediction ($\mathcal{E}_{\text{sel}}$). By selecting different threshold values for the confidence-based exit policy (Sec. 5.2), even the simplest (2-exit) configuration of input-dependent MESS network (Fig. 6c) provides a fine-grained trade-off between workload and accuracy. Exploiting this trade-off, we observe that input-dependent inference (Table 7; row (iv)) offers the highest computational efficiency under the same (50% mIoU) constraint.

Confidence Metric: To evaluate MESS exit-policy, we apply the proposed *image-level* confidence metric for segmentation, on top of both *top1* [20] and *entropy* [52]-based *pixel-level* confidence estimators, commonly used in multi-exit classification. Our experiments with various architectural configurations indicate that the proposed exit-policy offers a consistently better speed-accuracy trade-off compared to corresponding averaging counterparts (directly generalising from classification-based metrics by averaging per-pixel confidences for each image), with accuracy gains of up to 6.34 pp (1.17 pp on average). An example of this trade-off is illustrated in Fig. 4.

7 Conclusion

In this paper, we have presented the concept and realisation of multi-exit semantic segmentation. Applicable to state-of-the-art CNN approaches, MESS models perform efficient semantic segmentation, without sacrificing accuracy. This is achieved by introducing novel training and early-exiting techniques, tailored for MESS networks. Post-training, our framework can customise the MESS network by searching for the optimal multi-exit configuration (number, placement and architecture of exits) according to the target platform, pushing the limits of efficient deployment.

References

1. Mario Almeida, Stefanos Laskaridis, Ilias Leontiadis, Stylianos I. Venieris, and Nicholas D. Lane. EmBench: Quantifying Performance Variations of Deep Neural Networks Across Modern Commodity Devices. In *The 3rd International Workshop on Deep Learning for Mobile Systems and Applications (EMDL)*, 2019.
2. Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(12):2481–2495, 2017.
3. Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive Neural Networks for Efficient Inference. In *International Conference on Machine Learning (ICML)*, pages 527–536, 2017.
4. Liang-Chieh Chen, Maxwell Collins, Yukun Zhu, George Papandreou, Barret Zoph, Florian Schroff, Hartwig Adam, and Jon Shlens. Searching for efficient multi-scale architectures for dense image prediction. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8699–8710, 2018.
5. Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40(4):834–848, 2017.
6. Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv preprint arXiv:1706.05587*, 2017.
7. Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In *European Conference on Computer Vision (ECCV)*, pages 801–818, 2018.
8. Feiyang Cheng, Hong Zhang, Ding Yuan, and Mingui Sun. Leveraging semantic segmentation with learning-based confidence measure. *Neurocomputing*, 329:21–31, 2019.
9. Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.
10. Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision (IJCV)*, 88(2):303–338, 2010.
11. Biyi Fang, Xiao Zeng, and Mi Zhang. NestDNN: Resource-Aware Multi-Tenant On-Device Deep Learning for Continuous Mobile Vision. In *Annual International Conference on Mobile Computing and Networking (MobiCom)*, page 115–127, 2018.
12. Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially Adaptive Computation Time for Residual Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1039–1048, 2017.
13. Xitong Gao, Yiren Zhao, Łukasz Dudziak, Robert Mullins, and Cheng zhong Xu. Dynamic Channel Pruning: Feature Boosting and Suppression. In *International Conference on Learning Representations (ICLR)*, 2019.
14. Golnaz Ghiasi and Charless C Fowlkes. Laplacian pyramid reconstruction and refinement for semantic segmentation. In *European Conference on Computer Vision (ECCV)*, pages 519–534. Springer, 2016.
15. Swarnendu Ghosh, Nibaran Das, Ishita Das, and Ujjwal Maulik. Understanding Deep Learning Techniques for Image Segmentation. *ACM Computing Surveys (CSUR)*, 52(4):1–35, 2019.

16. Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic Contours from Inverse Detectors. In *International Conference on Computer Vision (ICCV)*, pages 991–998, 2011.
17. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
18. Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. In *NeurIPS 2014 Deep Learning Workshop*, 2014.
19. Weizhe Hua, Yuan Zhou, Christopher M De Sa, Zhiru Zhang, and G. Edward Suh. Channel Gating Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1886–1896, 2019.
20. Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Weinberger. Multi-Scale Dense Networks for Resource Efficient Image Classification. In *International Conference on Learning Representations (ICLR)*, 2018.
21. Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely Connected Convolutional Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4700–4708, 2017.
22. Andrey Ignatov, Radu Timofte, Andrei Kulik, Seungsoo Yang, Ke Wang, Felix Baum, Max Wu, Lirong Xu, and Luc Van Gool. AI Benchmark: All About Deep Learning on Smartphones in 2019. In *International Conference on Computer Vision (ICCV) Workshops*, 2019.
23. Junguang Jiang, Ximei Wang, Mingsheng Long, and Jianmin Wang. Resource Efficient Domain Adaptation. In *ACM International Conference on Multimedia (MM)*, 2020.
24. Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. Shallow-Deep Networks: Understanding and Mitigating Network Overthinking. In *International Conference on Machine Learning (ICML)*, 2019.
25. Stefanos Laskaridis, Alexandros Kouris, and Nicholas D. Lane. Adaptive Inference through Early-Exit Networks: Design, Challenges and Directions. In *Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning (EMDL)*, page 1–6, 2021.
26. Stefanos Laskaridis, Stylianos I. Venieris, Mario Almeida, Ilias Leontiadis, and Nicholas D. Lane. SPINN: Synergistic Progressive Inference of Neural Networks over Device and Cloud. In *Annual International Conference on Mobile Computing and Networking (MobiCom)*. ACM, 2020.
27. Stefanos Laskaridis, Stylianos I. Venieris, Hyeji Kim, and Nicholas D. Lane. HAPI: Hardware-Aware Progressive Inference. In *International Conference on Computer-Aided Design (ICCAD)*, 2020.
28. Ilias Leontiadis, Stefanos Laskaridis, Stylianos I. Venieris, and Nicholas D. Lane. It's Always Personal: Using Early Exits for Efficient On-Device CNN Personalisation. In *Proceedings of the 22nd International Workshop on Mobile Computing Systems and Applications (HotMobile)*, 2021.
29. Hao Li, Hong Zhang, Xiaojuan Qi, Ruigang Yang, and Gao Huang. Improved Techniques for Training Adaptive Deep Networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.
30. Xiaoxiao Li, Ziwei Liu, Ping Luo, Chen Change Loy, and Xiaoou Tang. Not All Pixels Are Equal: Difficulty-aware Semantic Segmentation via Deep Layer Cascade. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3193–3202, 2017.
31. Yanwei Li, Lin Song, Yukang Chen, Zeming Li, Xiangyu Zhang, Xingang Wang, and Jian Sun. Learning Dynamic Routing for Semantic Segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8553–8562, 2020.
32. Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-Path Refinement Networks for High-Resolution Semantic Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1925–1934, 2017.

33. Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime Neural Pruning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2181–2191, 2017.
34. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision (ECCV)*, pages 740–755, 2014.
35. Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-DeepLab: Hierarchical Neural Architecture Search for Semantic Image Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 82–92, 2019.
36. Luyang Liu, Hongyu Li, and Marco Gruteser. Edge Assisted Real-Time Object Detection for Mobile Augmented Reality. In *Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2019.
37. Yifan Liu, Ke Chen, Chris Liu, Zengchang Qin, Zhenbo Luo, and Jingdong Wang. Structured Knowledge Distillation for Semantic Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
38. Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.
39. Yunteng Luan, Hanyu Zhao, Zhi Yang, and Yafei Dai. MSD: Multi-Self-Distillation Learning via Multi-classifiers within Deep Neural Networks. *arXiv:1911.09418*, 2019.
40. Pauline Luc, Camille Couprie, Soumith Chintala, and Jakob Verbeek. Semantic Segmentation using Adversarial Networks. In *NIPSW on Adversarial Training*, 2016.
41. John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. SemanticFusion: Dense 3D Semantic Mapping with Convolutional Neural Networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4628–4635. IEEE, 2017.
42. Sachin Mehta, Mohammad Rastegari, Anat Caspi, Linda Shapiro, and Hannaneh Hajishirzi. ESPNet: Efficient Spatial Pyramid of Dilated Convolutions for Semantic Segmentation. In *European Conference on Computer Vision (ECCV)*, pages 552–568, 2018.
43. Vladimir Nekrasov, Hao Chen, Chunhua Shen, and Ian Reid. Fast Neural Architecture Search of Compact Semantic Segmentation Models via Auxiliary Cells. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9126–9135, 2019.
44. Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning Deconvolution Network for Semantic Segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1520–1528, 2015.
45. NVIDIA. NVIDIA Maxine - Cloud-AI Video-Streaming Platform. https://developer.nvidia.com/maxine, 2020. [Retrieved: July 20, 2022].
46. Chao Peng, Xiangyu Zhang, Gang Yu, Guiming Luo, and Jian Sun. Large Kernel Matters–Improve Semantic Segmentation by Global Convolutional Network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4353–4361, 2017.
47. Mary Phuong and Christoph H Lampert. Distillation-based Training for Multi-Exit Architectures. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1355–1364, 2019.
48. Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.
49. Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520, 2018.
50. Mennatullah Siam, Mostafa Gamal, Moemen Abdel-Razek, Senthil Yogamani, Martin Jagersand, and Hong Zhang. A Comparative Study of Real-Time Semantic Segmentation for Autonomous Driving. In *Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018.

51. Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
52. 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)*, pages 2464–2469. IEEE, 2016.
53. Andreas Veit and Serge Belongie. Convolutional Networks with Adaptive Inference Graphs. In *European Conference on Computer Vision (ECCV)*, pages 3–18, 2018.
54. Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Pérez. ADVENT: Adversarial Entropy Minimization for Domain Adaptation in Semantic Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2517–2526, 2019.
55. 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 (ECCV)*, pages 409–424, 2018.
56. Yulong Wang, Xiaolu Zhang, Xiaolin Hu, Bo Zhang, and Hang Su. Dynamic Network Pruning with Interpretable Layerwise Channel Selection. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 6299–6306, 2020.
57. Huikai Wu, Junge Zhang, Kaiqi Huang, Kongming Liang, and Yu Yizhou. FastFCN: Rethinking Dilated Convolution in the Backbone for Semantic Segmentation. In *arXiv preprint arXiv:1903.11816*, 2019.
58. Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. BlockDrop: Dynamic Inference Paths in Residual Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8817–8826, 2018.
59. Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. DeeBERT: Dynamic Early Exiting for Accelerating BERT Inference. In *58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2246–2251, 2020.
60. Qunliang Xing, Mai Xu, Tianyi Li, and Zhenyu Guan. Early Exit Or Not: Resource-Efficient Blind Quality Enhancement for Compressed Images. In *European Conference on Computer Vision (ECCV)*, 2020.
61. Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. End-to-end learning of driving models from large-scale video datasets. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2174–2182, 2017.
62. Zhuliang Yao, Shijie Cao, Wencong Xiao, Chen Zhang, and Lanshun Nie. Balanced Sparsity for Efficient DNN Inference on GPU. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 33, pages 5676–5683, 2019.
63. Juheon Yi and Youngki Lee. Heimdall: Mobile GPU Coordination Platform for Augmented Reality Applications. In *Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2020.
64. Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. BiSeNet: Bilateral Segmentation Network for Real-Time Semantic Segmentation. In *European Conference on Computer Vision (ECCV)*, pages 325–341, 2018.
65. Fisher Yu and Vladlen Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. In *International Conference on Learning Representations (ICLR)*, 2016.
66. Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated Residual Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 472–480, 2017.
67. Zhihang Yuan, Bingzhe Wu, Zheng Liang, Shiwan Zhao, Weichen Bi, and Guangyu Sun. S2DNAS: Transforming Static CNN Model for Dynamic Inference via Neural Architecture Search. In *European Conference on Computer Vision (ECCV)*, 2020.
68. E. Zakharov, Aleksei Ivakhnenko, Aliaksandra Shysheya, and V. Lempitsky. Fast Bi-layer Neural Synthesis of One-Shot Realistic Head Avatars. In *European Conference on Computer Vision (ECCV)*, 2020.

69. Dewen Zeng, Weiwen Jiang, Tianchen Wang, Xiaowei Xu, Haiyun Yuan, Meiping Huang, Jian Zhuang, Jingtong Hu, and Yiyu Shi. Towards Cardiac Intervention Assistance: Hardware-aware Neural Architecture Exploration for Real-Time 3D Cardiac Cine MRI Segmentation. In *ACM/IEEE International Conference on Computer-Aided Design (ICCAD)*, 2020.
70. Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your Own Teacher: Improve the Performance of Convolutional Neural Networks via Self Distillation. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.
71. Linfeng Zhang, Zhanhong Tan, Jiebo Song, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. SCAN: A Scalable Neural Networks Framework Towards Compact and Efficient Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
72. Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. ICNet for Real-Time Semantic Segmentation on High-Resolution Images. In *European Conference on Computer Vision (ECCV)*, pages 405–420, 2018.
73. Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid Scene Parsing Network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2881–2890, 2017.
74. Zhi Zhou, Xu Chen, En Li, Liekang Zeng, Ke Luo, and Junshan Zhang. Edge Intelligence: Paving the Last Mile of Artificial Intelligence with Edge Computing. *Proceedings of the IEEE*, 107(8):1738–1762, 2019.