DANCE: DAta-Network Co-optimization for Efficient Segmentation Model Training and Inference

CHAOJIAN LI, Rice University, TX, 77005
WUYANG CHEN, University of Texas at Austin, TX, 78712
YUCHEN GU, Rice University, TX, 77005
TIANLONG CHEN, University of Texas at Austin, TX, 78712
YONGGAN FU, Rice University, TX, 77005
ZHANGYANG WANG, University of Texas at Austin, TX, 78712
YINGYAN LIN, Rice University, TX, 77005

Semantic segmentation for scene understanding is nowadays widely demanded, raising significant challenges for the algorithm efficiency, especially its applications on resource-limited platforms. Current segmentation models are trained and evaluated on massive high-resolution scene images ("data-level") and suffer from the expensive computation arising from the required multi-scale aggregation ("network level"). In both folds, the computational and energy costs in training and inference are notable due to the often desired large input resolutions and heavy computational burden of segmentation models. To this end, we propose DANCE, general automated **DA**ta-**N**etwork **C**o-optimization for **E**fficient segmentation model **training and inference**. Distinct from existing efficient segmentation approaches that focus merely on light-weight network design, DANCE distinguishes itself as an automated **simultaneous** data-network **co-optimization** via both input data manipulation and network architecture slimming. Specifically, DANCE integrates automated data slimming which adaptively downsamples/drops input images and controls their corresponding contribution to the training loss guided by the images' spatial complexity. Such a downsampling operation, in addition to slimming down the cost associated with the input size directly, also shrinks the dynamic range of input object and context scales, therefore motivating us to also adaptively slim the network to match the downsampled data. Extensive experiments and ablating studies (on four SOTA segmentation models with three popular segmentation datasets under two training settings) demonstrate that DANCE can achieve **"all-win"** towards efficient segmentation (reduced training cost, less expensive inference, and better mean Intersection-over-Union (mIoU)). Specifically, DANCE can reduce ↓25%–↓77% energy consumption in training, ↓31%–↓56% in inference, while boosting the mIoU by ↓0.71%–↑ 13.34%.

CCS Concepts: • **Computing methodologies** → **Image segmentation**; *Neural networks*; • **Hardware** → *Platform power issues;*

Additional Key Words and Phrases: Efficient training and inference methods, semantic segmentation

We would like to acknowledge the funding support from the NSF RTML program (Award IDs: 1937592 and 2053279) for this project.

Authors' addresses: C. Li, Y. Gu, Y. Fu, and Y. Lin, Rice University, 6100 Main ST, Houston, TX, 77005; emails: {cl114, yg50, yf22, yingyan.lin}@rice.edu; W. Chen, T. Chen, and Z. Wang, University of Texas at Austin, 110 Inner Campus DR, Austin, TX, 78712; emails: {wuyang.chen, tianlong.chen, atlaswang}@utexas.edu.

1084-4309/2022/09-ART50 $15.00
https://doi.org/10.1145/3510835

ACM Reference format:
Chaojian Li, Wuyang Chen, Yuchen Gu, Tianlong Chen, Yonggan Fu, Zhangyang Wang, and Yingyan Lin. 2022. DANCE: DAta-Network Co-optimization for Efficient Segmentation Model Training and Inference. *ACM Trans. Des. Autom. Electron. Syst.* 27, 5, Article 50 (September 2022), 20 pages.
https://doi.org/10.1145/3510835

1 INTRODUCTION

The recent record-breaking performance of semantic segmentation using deep networks motivates an ever-growing application demand. However, those segmentation models typically bear a heavy computational cost to run (i.e., **inference**), making them extremely challenging to be deployed into resource-constrained platforms, ranging from mobile phones to wearable glasses, drones, and autonomous vehicles. Particularly, while existing works on improving inference efficiency are traditionally focused on classification, **state-of-the-art** (**SOTA**) segmentation models are even much more costly. For example, a ResNet50 [18] costs 4 GFLOPs for inference with an input size of 224×224. In comparison, for a DeepLabv3+ [3] with the ResNet50 backbone and the same 224×224 input (associated with an output stride of 16), the inference cost jumps up to 13.3 GFLOPs; the cost could further soar to 435 GFLOPs if we operate on a higher input resolution of $2,048 \times 1,024$. A similar trend can be expected in terms of the required energy costs. These highly required resource costs prohibit segmentation models from edge device deployments or at least degrade the quality of user experience. Specifically, such expensiveness of segmentation models arises from two aspects:

— *High input resolution and its proportional costs:* segmentation, as a dense prediction task, typically relies on fully convolutional networks whose inference FLOPs are proportional to the input size. Meanwhile, unlike classification, segmentation is known to be more resolution-sensitive due to its much finer prediction granularity [5]. Therefore, high-resolution inputs are preferred for improving algorithmic performance, yet contradicting the resource-saving needs.
— *Multi-scale aggregation:* segmentation is well-known for its strong dependency on multiple scale features [3, 45, 46, 51, 52] for contextual reasoning in combination with full-resolution outputs. Such a desired feature is often achieved by fusing a multi-resolution stream or aggregating paralleled filters with different sizes. Both the fusion and aggregation modules can incur heavy resource costs.

The expensiveness of segmentation is further amplified when we come to consider its **training** (e.g., continuous learning and adaptation) in resource-constrained settings. Many applications, such as autonomous vehicles and robots, require real-time and in-situ learning and continuous adaptation to new data to be considered truly intelligent. As compared to cloud-based (re)training, local (re)training helps avoid transferring data back and forth between data centers and local platforms, reducing communication loads, and enhancing privacy. Besides, the increasingly prohibitive energy, financial and environmental costs of training ML algorithms have become a growing concern even for training in the cloud [37]. However, resource-constrained training has not been explored much until a few recent efforts on classification [22, 39, 44].

Our contributions. This work aims at pushing forward the training and inference efficiency of SOTA segmentation models to a new level, from the current practice of merely focusing on lightweight network design, towards **a novel data-network co-optimization perspective**. Its core driving motivation can be summarized in **two points**: (1) *not all input samples are born equal* [13, 22]; and (2) *eliminating input variances reduces the model's learning workload* [12].

More specifically, we propose DANCE, an efficient training and inference framework that can be applied to any existing segmentation model. First, DANCE adopts an input adaptive automated data slimming technique. We propose a spatial complexity indicator to adapt the input images' spatial resolution, training sampling frequency, and weighted coefficients in the loss function. Thus DANCE makes the models focus more on the complicated samples during training, while during testing the input images' spatial resolution will be similarly reduced (i.e., downsampled).

Meanwhile, adaptively reducing input resolution has direct (proportional) impacts on the training and inference energy costs (i.e., both computation and memory movement costs). The indirect, yet also the important consequence is that the downsampled inputs become more "normalized" in terms of object and feature scales. Current segmentation models strongly rely on built-in multi-scale aggregation modules to balance between contextual reasoning and fine-detail preservation [5, 46]. Interestingly, with spatial-complexity-adaptive downsampled inputs, slimming those cost-dominant multi-scale aggregation building blocks save both training and inference costs without hampering the algorithmic performance; that is our proposed automated network slimming in DANCE.

Below we outline the contributions of the proposed DANCE framework:

- DANCE, **the first** data-network co-optimization framework, boosts the efficiency of both training and inference for segmentation models while mostly improving the accuracy. Furthermore, DANCE is general and thus can be applied to any existing segmentation backbone.
- DANCE, as shown in this article, simultaneously integrates automated data and network slimming to manipulate input images and their contribution to the model while slimming the network architecture in a co-optimization manner. Interestingly, the former can emulate the effect of multi-scale aggregation, thus enabling more aggressive slimming of their corresponding cost-dominant building blocks.
- Extensive experiments and ablation studies demonstrate that DANCE can achieve **"all-win"** (i.e., reduced training and inference costs, and improved model accuracy) towards efficient segmentation, when benchmarking on four SOTA segmentation models and three popular segmentation benchmark datasets. As shown in Figure 1, DANCE establishes **a new record tradeoff** between segmentation models' accuracy and training&inference efficiency.

2 RELATED WORKS

2.1 Efficient CNN Inference and Training

Extensive works have been proposed to improve the efficiency of CNN inference, most of them focus on the classification tasks. Network compression has been widely studied to speed up CNN inference, e.g., by pruning unimportant network weights [16, 20], quantizing the network into low bitwidths [21], or distilling lighter-weight networks from teachers [34]. For example, a representative automated pruning method (Network Slimming [25]) imposes L_1-sparsity making use of the scaling factor from the batch normalization; later progressive pruning methods (i.e., gradually increase the pruning ratio) are developed to improve the resulting models' accuracy [43]. Another stream of approaches involves designing compact models, such as MobileNet [36] and ShuffleNet [50]. Energy cost was leveraged in [42] to guide the pruning towards the goal of energy-efficient inference.

Resource-efficient training is different from and more complicated than its inference counterpart. However, many insights gained from the latter can be lent to the former. For example, the recent work [27] showed that performing active channel pruning during training can accelerate the empirical convergence. Later, Wang et al. [39] proposed one of the first comprehensive energy-efficient training frameworks, consisting of stochastic data dropping, selective layer updating, and low-precision back-propagation. They demonstrated its success in training several classification

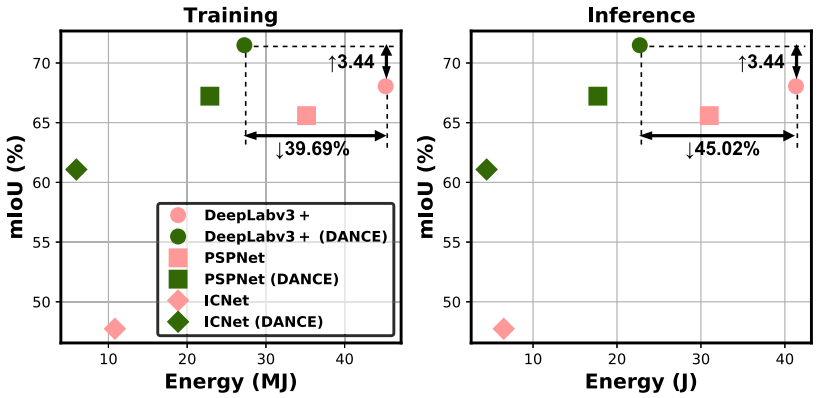

Fig. 1. The achieved mIoU vs. the required energy cost (Left: the total training energy cost; Right: the averaged inference energy cost per image) on the Cityscapes [8] test dataset. For the three segmentation models evaluated, DANCE achieves "all-win": reduced training cost, less expensive inference, and improved mIoU.

models with over 80% energy savings. Reference [22] accelerated training by skipping samples that may lead to low loss values (considered as less informative) at each iteration.

2.2 Semantic Segmentation

Multi-scale aggregation in segmentation. The multi-scale aggregation has been proven to be powerful for semantic segmentation [3, 45, 51, 52], via integrating multi-scale modules and high-/low-level features to capture patterns of different granularities. Pyramid Pooling and **Atrous Spatial Pyramid Pooling** (**ASPP**) modules were introduced in [52] and [3] to aggregate features learned in different sizes of receptive fields, adapting the models to objects with different semantic sizes. Parallel branches of different downsampling rates were proposed by [45, 51] to cover different resolutions. Although multi-scale aggregation contributes to segmentation accuracy improvement, it and its associated header introduce extra overhead during both training and inference (e.g., 52.98% inference FLOPs of Deeplabv3+ with a ResNet50 backbone and output stride of 16). That motivates us to slim such modules in DANCE.

Efficient segmentation models. A handful of efficient semantic segmentation models have been developed: ENet [32] used an asymmetric encoder-decoder structure together with early downsampling; ICNet [51] cascaded feature maps from multi-resolution branches under proper label guidance, together with network compression; and BiSeNet [45] fused a context path with a fast downsampling scheme and a spatial path with smaller filter strides.

Remaining challenges. However, the models above were neither *customized for* nor *evaluated on* ultra-high resolution images, and our experiments show that they did not achieve a sufficiently satisfactory tradeoff in such cases. A knowledge distillation method was also leveraged to boost the performance of a computationally light-weight segmentation model from a teacher network [19]. Despite their progress, none of them touches the training efficiency, nor any discussion related to *co-optimization* with the input data. Besides, the FLOPs number has a correlation to, but is not a faithful indicator of the actual energy cost, as pointed out by many prior works [42].

3 THE PROPOSED DANCE FRAMEWORK

This section presents our proposed DANCE framework. We will first provide an overview of DANCE in Section 3.1, and then introduce DANCE's automated *data* slimming and automated *network* slimming design in Sections 3.2 and 3.3, respectively.

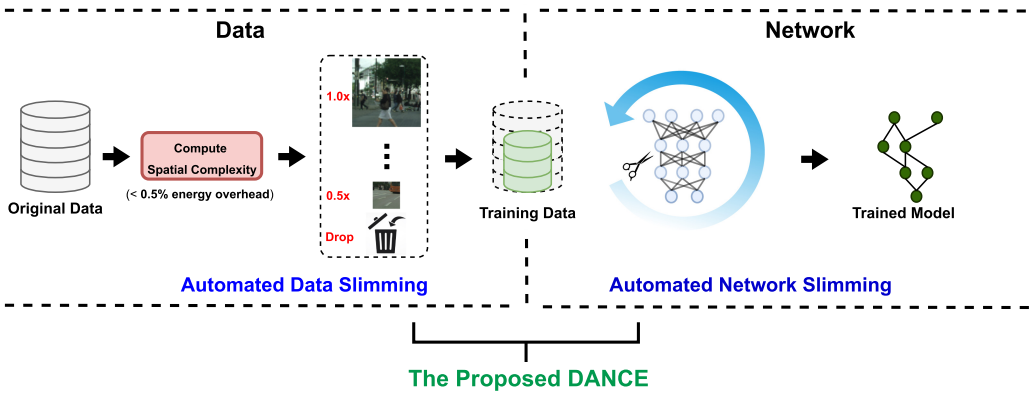

Fig. 2. An overview of the data-network co-optimization pipeline in the proposed DANCE framework.

3.1 DANCE overview

The driving hypothesis of DANCE is that matching the data and network can potentially boost both the model performance and hardware efficiency by removing redundancy associated with both the data and network. As such, DANCE aims at reducing the computational and energy costs of segmentation tasks during both training and inference via **a joint effort** from the **data-level** and **network-level**. Specifically, as shown in Figure 2, DANCE integrates both automated data and network slimming, where the former automatically performs *complexity-driven* data downsampling/dropping before applying the data to a network while the latter automatically and progressively prunes the network to **match the slimmed data**. A bonus benefit of DANCE is that the resulting data-network pipeline after training (i.e., inference) is also naturally cost-efficient.

3.2 DANCE: Automated Data Slimming

DANCE's automated *data* slimming strives to automatically downsample or drop input images and controls their corresponding contribution to the training loss, **adapting to** the images' spatial complexity which is estimated using a spatial complexity indicator.

Spatial complexity indicator. Spatial complexity has been commonly used as the basis for estimating image complexity [15, 30, 41], such as the one proposed in [48]:

$$SC_{mean} = \frac{1}{M} \sum \sqrt{s_h^2 + s_v^2}, \tag{1}$$

where s_h and s_v denote gray-scale images filtered with horizontal and vertical Sobel kernels, respectively, and M denotes the number of pixels. Developed by [48] to predict the image complexity for imaging compression/coding purpose, SC_{mean} reflects the pixel-level variances and is extremely efficient to calculate, e.g., account for only 0.15% FLOPs and <0.5% energy (on-device measurement when including both computations and data movements) of the DeepLabv3+ model (ResNet50 as the backbone with an output stride of 16) on one RGB image patch of size 224×224. The one proposed in [48] (1) has a higher correlation coefficient with the compression ratio of JPEG [38], which is regarded as the length of the shortest binary computer program as described in [48], as compared to the other image complexity estimators based on the entropy of images [23, 49, 53]; and (2) is computationally more efficient as compared to the estimators based on deep neural networks [7, 40].

In DANCE, we first compute all training samples' SC_{mean} and fit the corresponding **cumulative distribution function** (**CDF**) using a Maxwell–Boltzmann distribution [29], which turns out to

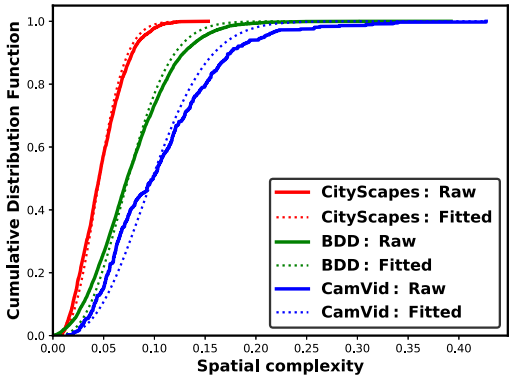

Fig. 3. The image complexity distribution of the training sets in the three considered urban scene understanding datasets.

be well-matched in all considered datasets as shown in Figure 3. Statistical analysis of SC_{mean} for a specific dataset is an interesting question, which we leave for future works.

Thanks to the fitted CDF, given an input image, we can project its $SC_{mean} \in [0, inf]$ to a variable $p \in [0, 1]$ via probability integral transform [11]. The resulting p is then directly used as the corresponding input image's downsampling ratio, stochastic dropping probability, and weighted coefficient in the training loss.

Complexity-adaptive downsampling. The proposed complexity-adaptive downsampling draws inspiration from recent findings, which show that *not all input samples are born equal* [13, 22], and is motivated by the fact that downsampling input image sizes can most straightforwardly reduce the training/inference energy costs, as well as directly benefit the memory throughput. Meanwhile, a few recent works learn to adjust resolution or respective fields [10, 28]. Their promising results further motivate our complexity-adaptive downsampling.

As prior works show that the minimal acceptable downsampling ratio is 0.5 for most segmentation models [3, 45], we make use of the spatial complexity indicator SC_{mean} to downsample the input images with a ratio of $(0.5p + 0.5) \in [0, 0.5]$, where p is the aforementioned projected value corresponding to the images' SC_{mean}. In contrast to the learning-based approaches in prior works [10, 28] that incur extra training workloads, we seek a reliable indicator that is mostly "training free" and inexpensive to compute, based on which we can estimate a proper downsampling rate per image adaptively. In particular, the energy overhead of our complexity-adaptive downsampling is <0.02% when estimated using real-device measurements in all our considered datasets.

Complexity-adaptive stochastic dropping. Recent pioneering CNN efficient works [22, 39] proposed that dropping a portion of training samples/mini-batches, either randomly or using some loss-based deterministic rules, can reduce the total training costs without notably sacrificing or even improving the algorithmic accuracy. Inspired by the stochastic dropping idea of [39], we incorporate the readily available spatial complexity indicator in Equation (1) to calibrate the dropping probability. Specifically, [39] proposes to randomly skip incoming data (in mini-batch) with a default probability of 50% (i.e., 50% of the data is discarded without being fed into the models). The authors demonstrated this naively simple idea (with zero overhead) to be highly effective for efficient training without hurting and even improving the achieved accuracy. We further hypothesize that the images with larger spatial complexity are more informative and likely to favor the achieved accuracy if being more frequently trained than the ones with smaller spatial complexity.

Therefore, instead of adopting a uniformly dropping probability for all images, we propose a simple yet effective heuristic to enable complexity-adaptive stochastic dropping by assigning a

smaller dropping probability to input images with larger spatial complexity. In particular, we assign $(1 - p)$ as the dropping probability, where p is the aforementioned projected value of the images' spatial complexity indicator (SC_{mean}).

Complexity-adaptive loss. Similarly, the losses produced by images with different complexities have been observed to contribute differently to the training loss [15] or convergence in training [22]. We thus prioritize the updates generated by samples with larger spatial complexity, and adopt an adaptive weighted loss as below:

$$\mathcal{L} = \frac{\sum w_i \cdot l_i}{\sum w_i} = \frac{\sum p_i \cdot l_i}{\sum p_i}, i = 1, 2, \ldots N, \tag{2}$$

where w_i is a scalar weighted coefficient, and l_i is the cross-entropy loss of samples, corresponding to the ith image of the current mini-batch with N images. Similar to the dropping probability assignment in complexity-adaptive stochastic dropping, an input image with larger spatial complexity will be assigned a larger weighted coefficient than the one with smaller spatial complexity. As such, we adopt weighted coefficients equal to the aforementioned projected value p of the images' spatial complexity indicator (SC_{mean}), i.e., $w_i = p_i, i = 1, 2, \ldots N$.

The above three techniques have a different effect on DANCE's achieved accuracy vs. efficiency tradeoffs, as shown in Section 4.3.5. Specifically, complexity-adaptive downsampling leads to the most significant cost (i.e., FLOPs, on-device energy, and latency) reduction, while obviously hurting the **mean Intersection-over-Union (mIoU)** (e.g., ↓57% and ↓61% less FLOPs in training and inference, respectively, with a ↓2.54% lower mIoU, when being applied on top of DeepLabv3+ [3] on the Cityscapes [9] dataset); complexity-adaptive dropping can boost the mIoU while slightly reducing the cost (e.g., a ↑3.33% higher mIoU with ↓14% and ↓25% less FLOPs in training and inference, respectively, when being applied on top of DeepLabv3+ [3] on the Cityscapes [9] dataset); and the complexity-adaptive loss can improve the mIoU (e.g., a ↑1.91% higher mIoU when being applied on top of DeepLabv3+ [3] on the Cityscapes [9] dataset together with complexity-adaptive downsampling) but have no influence on the cost of training or inference.

3.3 DANCE: Automated Network Slimming

Various ways to aggregate multi-scale features [3, 45, 51, 52] have been proved to improve segmentation accuracy at a cost of extra parameters and computations, leading to a higher training/inference energy burden. Thanks to the developed complexity-adaptive downsampling in DANCE's automated *data* slimming (see Section 3.2), the resulting inputs have been re-scaled according to their spatial complexity. We conjecture that such downsampled inputs naturally have more "normalized" object feature scales, i.e., complexity-adaptive downsampling can emulate the effect of multi-scale aggregation, and thus can potentially rely less on multi-scale aggregation modules for improving the segmentation accuracy. We thus expect that the network will appear to be more redundant when handling our automated *data* slimming's resulting downsampled inputs as the cost-dominant building blocks of multi-scale aggregation now become less important.

Progressive pruning during training. Motivated by the above conjecture and targeting reduced costs for both the training and inference (e.g., post-training pruning merely reduces inference costs), we propose an automated network slimming with a progressive pruning schedule during the training trajectory to prune the header (defined as the network added right after the backbone network for classification tasks, following [36]) of the networks for segmentation, which includes the aforementioned multi-scale feature modules and also often dominates both the training and inference costs, e.g., accounts for 52.98% FLOPs in DeepLabv3+ (with a ResNet50 backbone and an output stride of 16). Note that DANCE's effectiveness and insights are extended when other network pruning methods are considered. Here we consider progressive pruning without loss of generalization.

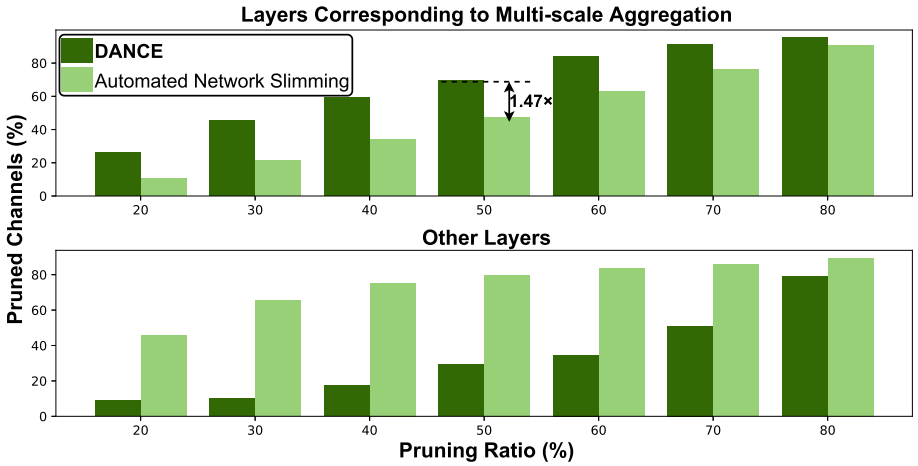

Fig. 4. The percentage of pruned channels for different kinds of layers under various pruning ratios on a DeepLabv3+ model (with a ResNet50 backbone and an output stride of 16) with the Cityscapes dataset. Specifically, the Pruning Ratio represents the target pruning ratio (i.e., the ratio of the total pruned channels as compared to the total channels of the unpruned models) of the whole model, which includes both the layers corresponding to multi-scale aggregation and other layers, and the Pruned Channels refers to the number of pruned channels in the layers corresponding to multi-scale aggregation (the top) or other layers (the bottom).

ALGORITHM 1: The algorithm for automated network slimming

1: Initialize the backbone weights $\boldsymbol{W_b}$ from ImageNet pre-trained models, the header weights $\boldsymbol{W_h}$ from random initialization, target pruning ratio $\boldsymbol{(1-r)}$, and pruning stages $\boldsymbol{S}$.
2: **while** t (epoch) $< t_{max}$ **do**
3: Update $\boldsymbol{W_b}$ and $\boldsymbol{W_h}$ using SGD training
4: **if** $\boldsymbol{t}$ **mod** $\frac{t_{max}}{S}$ **== 0 then**
5: Perform channel-wise pruning (based on [25]) on W_h with the target pruning ratio as $\boldsymbol{(1-r^{\frac{1}{S}})}$
6: **end if**
7: **end while**

To design the progressive pruning schedule, we develop a straightforward heuristic design, following the commonly used schedule in most pruning works [17, 26, 35]. Specifically, as illustrated in Algorithm 1, we first divide the whole training/adaptation process into several stages w.r.t the total number of iterations, and then perform channel-wise pruning (based on [25]) at the end of each stage. Please note that the optimization applied to the multi-scale feature modules are the same as the optimization used for other layers.

Co-optimization affects pruning patterns. To validate the aforementioned conjecture, we visualize the percentage of pruned channels in layers corresponding to multi-scale aggregation and other layers under different pruning ratios in Figure 4, when the models are trained with DANCE or merely DANCE's automated network slimming.

We can see that training with both automated data and network slimming, i.e., DANCE, always prunes more channels in layers corresponding to multi-scale aggregation (e.g., the ASPP module in DeepLabv3+) and fewer channels on other layers, under all the considered seven pruning ratios

between 20% and 80%, while merely automated network slimming does the opposite. Specifically, as compared to training using merely automated data slimming under the same pruning ratio of 50%, the model trained with both automated data and network slimming, i.e., models trained using DANCE, prunes 1.47× more channels in layers associated with multi-scale aggregation, where the corresponding accuracy is also higher (e.g., a 5.33% higher mIoU on the Cityscapes validation dataset together with a 54.8% lower inference energy with images of 592×592).

The experiment in Figure 4 together with those in the experiment section verifies our conjecture that (1) matching the data with the network can potentially improve the accuracy (thanks to the match between slimmed data and unpruned channels' distribution) and remove redundant costs associated with both the data and network, thus achieving "all-win": reducing both the training and inference costs while improving the achieved model accuracy (mIOU); and (2) DANCE's automated data slimming can (partially) emulate the effect of multi-scale aggregation in segmentation models, enabling a higher pruning ratio on the corresponding multi-scale aggregation modules. The observations are consistent when other pruning methods and different pruning hyperparameters are used in DANCE's automated network slimming (more details in Section 4.3.4), again verifying that the above conclusion (i.e., "co-optimization affects the optimal pruning patterns") holds for DANCE regardless of the adopted pruning designs.

3.4 DANCE: Theoretical Cost Saving

To better illustrate why DANCE can reduce both training and inference costs, we perform a theoretical analysis below. Taking the FLOPs cost as an example, the cost reduction ratio in inference (R_{infer}) and training (R_{train}) can be formulated as

$$R_{infer} = \frac{1}{I_{infer}} \sum_{i=1}^{I_{infer}} [1 - (0.5p_i + 0.5)^2 r])], \tag{3}$$

$$R_{train} = \frac{1}{SI_{train}} \sum_{s=1}^{S} \sum_{i=1}^{I_{train}} p_i [1 - (0.5p_i + 0.5)^2 r^{\frac{s}{S}}], \tag{4}$$

where I_{infer} and I_{train} are the numbers of images in the dataset for inference and training, respectively, S is the number of total stages during progressive pruning with a target pruning ratio of $(1 - r)$, and $p_i \in [0, 1]$ is the indicator output for the ith image. Specifically, $\boldsymbol{p_i}$ in Equation (4) represents the cost reduction ratio caused by the **complexity-adaptive stochastic dropping** (**CASD**); $\mathbf{1 - (0.5}\boldsymbol{p_i}\mathbf{ + 0.5)^2}$ in Equations (3) and (4) represents the cost reduction ratio caused by the **complexity-adaptive downsampling** (**CAD**); $\boldsymbol{r}$ in Equation (3) and $\boldsymbol{r}^{\frac{s}{S}}$ in Equation (4) represents the cost reduction ratio caused by the **automated network slimming** (**ANS**) during inference and training, respectively. Thus, the cost reduction resulting from DANCE is **quadratic** in terms of $p_i \in [0, 1]$ **for inference** and **cubic for training**.

4 EXPERIMENTS

In this section, we evaluate DANCE on four segmentation models and three popular urban scene understanding datasets in terms of mIoU and the total training/inference FLOPs and energy cost, where the energy cost is measured when training/inference the corresponding models in a SOTA edge device (JETSON TX2 [31]). We consider both the computational and energy costs because the former is commonly adopted and thus helps to benchmark against prior works while the latter better captures the real hardware cost.

Table 1. The FLOPs, Energy Cost, and mIoU of DANCE on Top of the Four Models on the **Cityscapes** Test Dataset

Model	FLOPs		Energy		mIoU
	Train. (P)	Infer. (G)	Train. (MJ)	Infer. (J)	(%)
DeepLabv3+	198.31	743.64	45.21	41.32	68.05
DANCE Improv.	**−35.75%**	**−53.67%**	**−39.69%**	**−45.02%**	**+3.44**
PSPNet	153.61	582.54	35.16	30.99	65.59
DANCE Improv.	**−39.28%**	**−50.23%**	**−34.92%**	**−42.81%**	**+1.93**
ICNet	39.33	45.20	10.82	6.51	47.74
DANCE Improv.	**−49.77%**	**−55.67%**	**−45.27%**	**−47.69%**	**+13.34**
BiSeNet	73.64	157.41	18.40	9.93	71.69
DANCE Improv.	**−32.77%**	**−39.29%**	**−25.66%**	**−31.27%**	**−0.71**

4.1 Experiment Setting

Considered models and datasets. Our evaluation of DANCE considers four SOTA segmentation models (two complicated models: DeepLabv3+ [3], PSPNet [52], and two compact models: ICNet [51], and BiSeNet [45]) and three commonly used urban scene understanding datasets (Cityscapes [9], CamVid [1], and BDD [47]) in many efficient segmentation models [4, 45, 51].

Experimental platforms and training details. All experiments (except the energy measurements) are performed on a workstation with NVIDIA 2080Ti GPU cards using the PyTorch framework [33] for a fair comparison. We use an SGD optimizer with a learning rate of 1×10^{-3} for training all models except ICNet, which adopts a learning rate of 1×10^{-2} due to the unavailability of the corresponding ImageNet pre-trained model; and a minibatch size of (1) 8 for the DeepLabv3+ and PSPNet models and (2) 16 for the BiSeNet and ICNet models. Specifically, the pruning stage S and target pruning ratio $(1 - r)$ in Algorithm 1 are set to 6 and 0.5, respectively, in all our experiments except for their ablation studies (e.g., Section 4.3.4).

4.2 Performance on Various Datasets/Models

In this subsection, we apply DANCE to the four segmentation models and three datasets and compare the resulting segmentation accuracies and inference/training costs with those of the base models.

4.2.1 DANCE on the Cityscapes Dataset. Table 1 compares the segmentation accuracy, and computational and energy costs of DANCE on the four models, i.e., DeepLabv3+ [3], PSPNet [52], ICNet [51], and BiSeNet [45], when evaluated on the Cityscapes dataset. We can see that (1) DANCE saves about 36%–39% and 35%–40% computational and energy costs in training (a similar trend in inference), while boosting the mIoU in the cases of DeepLabv3+ [3] and PSPNet [52] by 3.44% and 1.93%, respectively; (2) In the case of ICNet, DANCE achieves a 13.34% higher mIoU with up to 45% energy savings than those of the base model, where the lower mIoU of the base model might be due to the lack of a corresponding ImageNet pre-trained model; and (3) Though DANCE does not boost the mIoU on the compact model of BiSeNet, it does save in training energy cost and win bigger (saving up to 31% energy) in inference.

4.2.2 DANCE on the CamVid Dataset. Under smaller images (720×960) in CamVid (vs. $1{,}048 \times 2{,}048$ in Cityscapes), we can still observe similar trends as those in Cityscapes (see Table 1). Specifically, our DANCE can still save 32%–49% energy cost, as shown in Table 2, while achieving improved mIoU (over 1.4%). For the compact model BiSeNet, with a comparable mIoU, our DANCE still stably brings 32% and 33% energy savings in training and inference, respectively.

Table 2. The FLOPs, Energy Cost, and mIoU of DANCE on Top of the Four Models on the CamVid Test Set

Model	FLOPs		Energy		mIoU
	Train. (P)	Infer. (G)	Train. (MJ)	Infer. (J)	(%)
DeepLabv3+	37.19	254.62	7.30	20.19	69.15
DANCE Improv.	**−32.76%**	**−47.6%**	**−31.76%**	**−43.65%**	**+1.51**
PSPNet	27.27	208.77	4.71	14.64	65.28
DANCE Improv.	**−39.69%**	**−46.47%**	**−32.22%**	**−41.22%**	**+2.82**
ICNet	6.01	16.33	1.78	4.21	53.29
DANCE Improv.	**−45.32%**	**−52.64%**	**−49.21%**	**−56.21%**	**+1.40**
BiSeNet	6.46	54.17	2.72	6.77	68.6
DANCE Improv.	**−38.09%**	**−41.10%**	**−32.45%**	**−33.76%**	**−0.27**

Table 3. The FLOPs, Energy Cost, and mIoU of DANCE on Top of the Four Models on the **BDD** Test Set on Adaptation

Model	FLOPs		Energy		mIoU
	Train. (P)	Infer. (G)	Train. (MJ)	Infer. (J)	(%)
DeepLabv3+	97.01	339.46	17.74	27.67	52.66
DANCE Improv.	**−79.31%**	**−51.86%**	**−77.15%**	**−43.5%**	**+0.12**
PSPNet	72.56	290.57	11.82	19.37	39.54
DANCE Improv.	**−37.47%**	**−49.65%**	**−25.18%**	**−40.29%**	**+5.51**
ICNet	58.27	21.68	15.73	5.17	39.53
DANCE Improv.	**−58.65%**	**−51.51%**	**−61.47%**	**−37.47%**	**+0.47**
BiSeNet	45.32	72.27	7.48	7.94	56.20
DANCE Improv.	**−34.08%**	**−44.47%**	**−27.92%**	**−38.46%**	**+0.27**

Additionally, our experiments of applying DANCE on top of DeepLabv3 [2] on the PASCAL VOC 2012 [14] dataset, where the second last feature map of Mobilenetv2 [36] is used for DeepLabv3's [2] heads, validate DANCE's benefits for even compact models, since it stably brings 34% and 37% FLOPs savings in training and inference, respectively, while achieving a comparable mIoU (↓0.04%).

4.2.3 DANCE on the BDD Dataset for Adaptation. As Section 1 stated, for most on-device learning applications, training from scratch is not necessary and the ability to adapt to new data can be more interesting for some applications, especially for autonomous vehicles and robots. Here, we choose the BDD [47] for the adaptation experiments. We use pre-trained models on Cityscapes to adapt to unseen images in BBD. For a fair comparison, we choose the same checkpoints as the pre-trained model for each model in the experiments. The adaptation performance is summarized in Table 3, which shows that while being similar to the performance on Cityscapes, DANCE saves up to 77% energy cost while achieving a slightly better (+0.12%) mIoU over the baseline, or boosts the mIoU by 5.51% when requiring even a 25% lower energy cost than the baseline.

The extensive results in Tables 1–3 show that **DANCE can achieve "all-win" on all the three datasets when applied to both DeepLabv3+ and PSPNet**: lower training cost (energy savings: 77%–25%), more efficient inference (energy savings: 40%–45%), and improved mIOU (0.12%–5.51%), demonstrating **the consistent superiority of DANCE on complicated models**. As for the performance on compact models, DANCE can improve the efficiency of both trainings (energy savings: 25%–61%) and inference (energy savings: 31%–56%) with a slightly dropped or even better mIoU

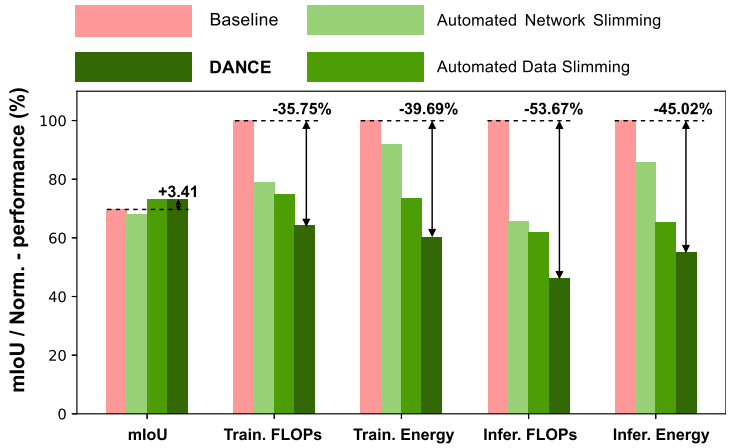

Fig. 5. Ablation studies of data-network co-optimization in DANCE on the DeepLabv3+ model (with a ResNet50 backbone and an output stride of 16) and Cityscapes validation dataset, where FLOPs and energy are normalized to those of the baseline.

Table 4. The FLOPs and mIoU of **Co-optimization** and **Separate Optimization** on top of DeepLabv3+@CityScapes

Method	FLOPs		mIoU
	Train. (P)	Infer. (G)	(%)
Baseline	198.31	743.64	69.71
Optimize network After Data	+0.85%	−61.10%	+0.66
Optimize Data After network	+12.99%	−55.88%	+3.08
Co-Optimize (DANCE)	**−35.75%**	**−56.57%**	**+3.41**

(−0.71%−13.34%) on all three datasets, indicating that **DANCE can benefit energy efficiency of even compact models**.

4.3 Ablation Studies of DANCE

In this subsection, we perform ablation studies of DANCE for evaluating the effectiveness of its data-network co-optimization, p indicator, and automated data slimming.

4.3.1 Ablation Study on the Effectiveness of DANCE's Data-network Co-optimization. **DANCE vs. only automated network/data slimming.** As shown in Figure 5, combining both automated data and network slimming (i.e., DANCE) achieves (1) better performance (in terms of the training cost, inference cost, and mIoU) than the standalone implementation of either of these two techniques integrated into DANCE (i.e., automated data and network slimming); and (2) a much higher mIoU than the baseline (+3.41%) while requiring 39% and 45% less energy in training and inference, respectively. This set of experiments indicates the advantage of jointly matching the data and network for co-optimization.

DANCE vs. optimizing the network and data separately. To demonstrate how DANCE integrates co-optimization rather than a naive combination of automated network and data slimming, we compare **co-optimization** (DANCE) with **separate optimization** (optimizing the network/data and then the data/network sequentially) in Table 4. In the co-optimization setting, both the automated network and data slimming will happen in each iteration of the training process.

Table 5. The Inference mIoU of w/ DANCE and w/o DANCE on Top of DeepLabv3+ for CityScapes's Large, Medium, and Small Scale of Objects (Manually Picked **Static** Scales), where DANCE Can Further Provide **Dynamic** Scales

Method	w/o DANCE			**w/ DANCE**
Image Scales	368 × 368	496 × 496	592 × 592	**Dynamic**
IoU of Wall (%)	**50.56**	48.68	46.40	**52.69**
IoU of Motorcycle (%)	53.96	**57.51**	55.09	**58.23**
IoU of Traffic Sign (%)	70.17	72.41	**72.94**	**73.58**

Table 6. The mIoU of Using the **Proposed** p (in Section 3.2), **Random** p, or **Inverse** p Indicator in DANCE on Top of DeepLabv3+@CityScapes under Same Training Cost Budget

Method	**Train. FLOPs (P)**	**mIoU(%)**
Proposed p indicator	**127.41**	**73.12**
Random p indicator	+0.06%	−4.08
Inverse p indicator, i.e., 1-p	+0.00%	−11.03

However, in the separate optimization setting, the automated network slimming will only happen in the first/second half of the whole training process and the automated data slimming will only happen in the second/first half of the whole training process. Table 4 illustrates that network and data need to be jointly co-optimized to achieve the best mIoU-cost tradeoff, while optimizing (i.e., slimming) the network and data sequentially will cause a 0.33%–2.75% mIoU drop on DeepLabv3+@Cityscapes at an even higher computational cost (e.g., +48.74%) than DANCE.

4.3.2 Ablation Study of DANCE on Objects with Different Scales. Here we compare the inference mIoU when turning off and on our DANCE applied on top of DeepLabv3+, when testing representative large, medium, and small scales (i.e., wall, motorcycle, and traffic sign) of objects in Cityscapes. As shown in Table 5, we can see that (1) small/large scales of objects favor/degrade the achieved inference mIoU of applying DeepLabv3+ to the selected objects of different scales; and (2) DANCE, which inherently incorporates **dynamic** scales to its applied data, consistently outperforms its baselines even for the manually selected objects which have **static** scales by design, indicating the advantage of DANCE's automated choices of adaptive scales of data, validating DANCE's inherent advantages in handling datasets/tasks in which the objects have different scales, which is common for semantic segmentation datasets (e.g., Cityscapes [9], CamVid [1], and BBD [47]). This is also the reason why our proposed DANCE can even improve the accuracy in addition to reducing the training and inference cost.

4.3.3 Ablation Study of the p Indicator' Effectiveness. The spatial complexity indicator presented in Section 3.2 is to provide a variable $p \in [0, 1]$ for estimating a given image's complexity, which will be directly used to guide the slimming direction (e.g., image's downsampling ratio). As shown in Table 6, we apply **inverse** p or **random** p to replace the **proposed** p indicator in DANCE, and find that its resulting mIoU drops 11.03% or 4.08% under the same training cost budget, respectively, validating the advantageous effectiveness of our proposed p indicator. Additionally, Figure 6 visualizes 24 image samples randomly selected from the image groups with the **largest** 33%, **medium** 33%, and **smallest** 33% spatial complexity in the Cityscapes [9] training dataset. Interestingly, we can see, as expected, that the image complexity identified by the adopted indicator is consistent with that seen by human eyes, e.g., images with spatial complexity falling within the smallest 33% of the dataset have a simpler background and include fewer objectives.

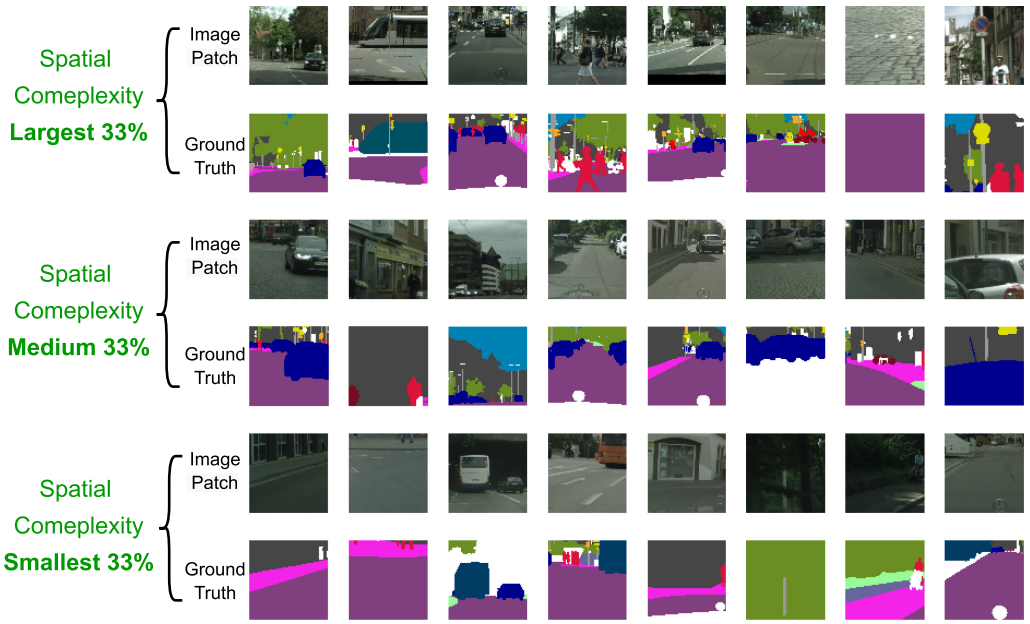

Fig. 6. Visualizing the images randomly selected from image groups with the **largest** 33%, **medium** 33%, and **smallest** 33% spatial complexity in the Cityscapes [9] training dataset.

Table 7. The Number of Pruned Weights for Different Kinds of Layers Under Various Pruning Ratios on a DeepLabv3+ Model (with a ResNet50 Backbone and an Output Stride of 16) with the Cityscapes Dataset

Pruning	**#pruned weights (DANCE's - ANS's)**	
Ratio	ASPP Module Layers	Other Layers
20%	61,759	−61,759
30%	96,455	−96,455
40%	102,535	−102,535
50%	96,674	−96,674
60%	76,767	−76,767
70%	51,177	−51,177
80%	24,986	−24,987

4.3.4 Ablation Study of DANCE's Effectiveness Regardless of the Adopted Pruning Methods. We consistently find that DANCE's advantages in enabling data model co-optimization is effective regardless of the adopted pruning methods. For example, Table 7 summarizes the pruning results when turning on and off DANCE's automated data slimming during pruning, where we adopt the unstructured pruning in [16]. Again, similar observations can be made as those in [25] when using channel-wise pruning.

By comparing the number of pruned weights in the ASPP module layers and other layers when performing our proposed DANCE and ANS as shown in Table 7, we can observe that (1) DANCE prunes more weights of the ASPP module layers than ANS, i.e., the number of DANCE pruned weights is more than that of ANS pruned weights for the ASPP module layers, while it is opposite

Table 8. Ablation Studies on the Component Techniques of DANCE's Automated Data Slimming on the DeepLabv3+ Model (with a ResNet50 Backbone and an Output Stride of 16) and the Cityscapes Validation Dataset, where † (i.e., No.7) is our DANCE Setting

No.	ANS	CAD	CAL	CASD	RD	Train. FLOPs	Train. Energy	Infer. FLOPs	Infer. Energy	mIoU
1						198.3 (P)	45.21 (MJ)	743.6 (G)	41.32 (J)	69.71(%)
2	✓					−21.16%	−8.08%	−34.34%	−14.36%	−1.52
3	✓	✓				−57.05%	−51.12%	−60.96%	−46.52%	−2.54
4	✓	✓	✓			−56.98%	−51.45%	−61.15%	−46.96%	−0.63
5	✓				✓	−13.73%	−15.90%	−25.07%	−14.21%	+1.96
6	✓			✓		−13.92%	−14.10%	−25.21%	−15.71%	+3.33
7†	✓	✓	✓	✓		**−35.75%**	**−39.69%**	**−53.67%**	**−45.02%**	**+3.41**

[a] ANS: Automated network slimming.
[b] CAD: Complexity-adaptive downsampling.
[c] CASD: Complexity-adaptive stochastic dropping.
[d] CAL: Complexity-adaptive loss.
[e] RD: Randomly drop 50% [39].

in other layers and (2) under the same pruning ratio, the more weights of ASPP module layers are pruned by DANCE than by ANS (e.g., +96, 674 under 50% pruning ratio), the fewer weights of other layers would be pruned by DANCE than by ANS (e.g., −96, 674 under 50% pruning ratio), leading to a symmetric distribution of the difference between the number of pruned weights in the ASPP module layers and other layers when using DANCE and ANS. Specifically, training with both automated data and network slimming, i.e., DANCE, always prunes more weights in layers corresponding to multi-scale aggregation (e.g., the ASPP module in DeepLabv3+) and fewer weights on other layers, under all the considered seven pruning ratios between 20% and 80%, whereas merely using DANCE's ANS does the opposite. This set of experiment results further confirms that (1) DANCE's automated data slimming can (partially) emulate the effect of multi-scale aggregation in segmentation models, and thus enable a higher pruning ratio on the corresponding multi-scale aggregation modules, and (2) matching the data with a model can potentially improve the model accuracy and remove redundant costs associated with both the data and model, thus achieving "all-win", which is consistent with the results in Figure 1.

4.3.5 Ablation Study of DANCE's Automated Data Slimming. As described in Section 3.2, DANCE's *automated data slimming* integrates three techniques, including CAD, CASD, and **complexity-adaptive loss** (**CAL**), which are guided by the adopted spatial complexity indicator. In this subsection, we evaluate the efficacy of these techniques and their different combinations on top of DANCE's ANS (see Section 3.3), in terms of the resulting task accuracy (mIoU), and computational and energy savings of both inference and training, as summarized in Table 8. Note that all the task accuracy and computational and energy savings are normalized to those of the standard DeepLabv3+ [3] model and Cityscapes dataset (See row No. 1 of Table 8). We next discuss the observations in terms of the **"all-win"** goal (i.e., reducing both the training and inference costs while improving the achieved model accuracy (mIoU)):

1. Complexity-adaptive downsampling (CAD): Comparing the results in Rows No. 2 and No. 3 shows that CAD+ANS (see Row No. 3, i.e., applying CAD, which has the advantage of "training free", on top of DANCE's ANS), can save 42.92% and 32.16% energy cost in training and inference, respectively, while decreasing the mIoU by 1.02% (i.e., −1.52% vs. −2.54%), as compared to merely performing ANS (see Row No. 2), indicating that CAD offers a new tradeoff between the achieved energy efficiency and mIoU.

2. Complexity-adaptive loss (CAL): Comparing the results in Rows No. 3, and No. 4 shows that CAL+CAD+ANS (see Row No. 4, i.e., applying CAL on top of ANS and CAD) can boost the mIoU by 1.91% as compared to merely combining CAD and ANS (i.e., CAD+ANS in Row No. 3), while still

Table 9. Ablation Study of DANCE's Hyperparameters: DANCE with Different Ranges of (1) Weighted Coefficients in CAL and (2) Dropping Probability in CASD on DeepLabv3+ with Cityscapes

Settings		Train. FLOPs	Train. Energy	Infer. FLOPs	Infer. Energy	mIoU
Range of the Weighted Coefficients in CAL	2.0–1.0	124.66 (P)	26.43 (MJ)	358.39 (G)	23.57 (J)	72.30%
	4.0–1.0	−1.76%	−2.65%	+0.78%	+0.52%	+0.64%
	1.0–0.0	+2.21%	+3.18%	−3.87%	−3.60%	+0.82%
Range of the Dropping Probability in CASD	60%–40%	128.05 (P)	27.55 (MJ)	345.21 (G)	22.74 (J)	71.13%
	75%–25%	−0.88%	−0.57%	−1.37%	−1.22%	+0.27%
	100%–0%	−0.50%	−1.02%	−0.20%	−0.07%	+1.99%

reducing the energy cost in training and inference by 51.45% and 46.96%, respectively, as compared to the DeepLabv3+ baseline (Row No. 1), indicating that adding CAL on top of CAD and ANS can further boost the model accuracy while maintaining the achieved energy efficiency.

3. Complexity-adaptive stochastic dropping (CASD): First, comparing the results in Rows No. 5 and No. 6 shows that the proposed CASD (Row No. 6) can achieve a 1.37% higher mIoU than the random dropping technique in [39] (Row No. 5) under the same energy cost of both training and inference, indicating the advantage of *complexity-adaptive* stochastic dropping over *random* dropping in [39]. Second, comparing the results in Rows No. 4 and No. 7 shows that applying CASD on top of CAL+CAD+ANS (Row No. 4) can boost the mIoU by 4.04% as compared to merely combining ANS, CAD, and CAL (Row No. 4), and by 3.41% as compared to the DeepLabv3+ baseline (Row No. 1), while obtaining 39.69% and 45.02% energy savings in training and inference, respectively, as compared to the DeepLabv3+ baseline (Row No. 1).

This set of comparisons indicates the effectiveness of the proposed DANCE's automated data slimming, i.e., integrating all three component techniques of DANCE's automated data slimming can achieve the most favorable data-network co-optimization benefits as it achieves the **"all-win"** goal as shown in Figure 1.

4.3.6 Ablation Study of DANCE's Hyperparameters. In this subsection, we perform experiments for evaluating DANCE with different hyperparameters by changing the ranges of (1) the weighted coefficients in CAL and (2) dropping probability in complexity-adaptive stochastic dropping (CAL) (as described in Section 3.2), and summarize the results in Table 9. To better study the effect of each of the aforementioned hyperparameters, we fix others with the default ones (as described in Section 3.2) when tuning one of them.

Note that the larger the *ratio of endpoints in the dynamic range* of both the weighted coefficients and dropping probabilities are, the more (less) frequent images with a higher (lower) spatial complexity would be used. And the largest ratio is $1.0/0.0 = \infty$ and $100\%/0\% = \infty$ for the weighted coefficients and dropping probabilities, respectively, which is also the default setting as mentioned in Section 3.2.

The results in Table 9 show that **increasing the frequency of training images with a higher spatial complexity** (defined in Equation (1)), by increasing the *ratio of endpoints in the dynamic range* of the weighted coefficient or dropping probability, **favors the segmentation accuracy (i.e., a higher mIoU)**. This observation is consistent with that of [15, 22]. Specifically, changing the dropping probability range from 60%–40% to 100%–0% boosts the achieved mIoU by 1.99%, while changing the weighted coefficient range from 2.0–1.0 to 1.0–0.0 leads to an improved mIoU of 0.82%, while the training and inference costs of both cases mostly stay the same.

4.3.7 Comparing DANCE with other Data Preprocessing and Network Slimming Methods. To better illustrate the advantage of the proposed DANCE framework, we conduct a comparison between DANCE and both (1) random dropping, a data pre-processing method for efficient training proposed in [39], and (2) MTP [6], a dedicated network pruning method for semantic

Table 10. Comparing the Proposed DANCE, Random Dropping [39], MTP [6] in Terms of mIoU, Inference FLOPs, and Training FLOPs, when Applying Them on top of BiSeNet [45] at the Cityscapes [9] Dataset

Method	mIoU (%)	Infer. FLOPs	Train. FLOPs
Random dropping [39] Improv.	69.56	↓0%	↓50%
MTP [6] Improv.	70.22	↓43%	↑>0%
DANCE Improv.	**70.98**	**↓39%**	**↓33%**

Table 11. Adding DANCE on Top of DeepLabv3+ [3] at Cityscapes [9] Dataset with Costly Recipes

Method	mIoU (%)	Infer. FLOPs	Train. FLOPs
DeepLabv3+ (costly recipes)	82.10	100%	100%
DANCE Improv.	**↑0.26%**	**↓51%**	**↓33%**

Table 12. Runtime Breakdown of DeepLabv3+ [3] at Cityscapes [9] Dataset w/ and w/o DANCE

Layers	Runtime w/o DANCE	Runtime w/ DANCE
ResNet50 backbone	100%	64%
Header - Multi-scale aggregation layers	100%	41%
Header - Other layers	100%	52%
All layers	100%	55%

segmentation networks. As summarized in Table 10, when being applied on top of BiSeNet [45] with the Cityscapes [9] dataset, we can see that (1) as compared to random dropping [39], DANCE achieves a ↑1.42% higher mIoU while requiring ↓39% less FLOPs during inference, and (2) as compared to MTP [6], DANCE achieves a ↑0.76% higher mIoU under a similar inference FLOPs while trimming down the training FLOPs by at least ↓33%. This set of experiments further validates DANCE's advantage as a data-network co-optimization framework for both training and inference, as compared to prior works that merely focus on data preprocessing or network slimming.

4.3.8 Effectiveness of DANCE with More Costly Recipes. The results in Section 4.2 are reported when using light recipes to better align with our goals of efficient edge training/inference. Specifically, we use a relatively small **output stride** (**OS**) of 1/16, as compared to 1/8 in [3], and do not use the extra 20 K Cityscapes coarse data [9] or COCO pretraining [24]. To verify the effectiveness of the proposed DANCE under more costly recipes, we further apply DANCE on top of DeepLabv3+ [3] and the Cityscapes [9] dataset, where we adopt ResNet101 as the backbone, an OS of 1/8, and extra 20 K Cityscapes coarse data [9]. As shown in Table 11, DANCE still achieves "all-win" under such a more costly recipe, i.e., achieving a ↑0.26% higher mIoU, while requiring ↓51% and ↓33% less FLOPs in inference and training, respectively.

4.3.9 Runtime Breakdown w/ and w/o DANCE. To better understand how the proposed DANCE reduces the hardware cost (e.g., FLOPs, on-device energy, and latency) of the networks, we break the runtime of the whole network into different parts when conducting inference of DeepLabv3+ [3] with the Cityscapes [9] dataset w/ and w/o DANCE. As shown in Table 12, the multi-scale aggregation layers in the header benefit the most from our DANCE technique, requiring only 41% of the runtime as compared to that of w/o DANCE). This is because (1) the cost of the multi-scale aggregation layers can be reduced by both DANCE's automated data and network slimming and (2) DANCE tends to prune more weights in the multi-scale aggregation layers than in other layers, as discussed in Section 3.3.

5 CONCLUSIONS

We proposed DANCE for boosting segmentation efficiency during both training and inference, leveraging the hypothesis that maximum model accuracy and efficiency should be achieved when the data and model are optimally matched. On the "data-level", DANCE's automated data slimming not only halves the computational and energy costs, but also boosts the segmentation accuracy. Interestingly, DANCE's automated data slimming can emulate the effect of multi-scale feature extraction yet at a much lower cost. This further motivates DANCE's automated network slimming on the "model-level" that advocates automatically pruning the model adapting to the resulting data slimmed by DANCE's automated data slimming and leads to more pruning in the cost-dominant building blocks for multi-scale feature extraction, validating our hypothesis and further reducing both training and inference costs. Extensive experiments and ablation studies validate DANCE's effectiveness and superiority, which resides in its capability to automatically match the data and network via automated co-optimization.

REFERENCES

[1] Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. 2008. Segmentation and recognition using structure from motion point clouds. In *Proceedings of the European Conference on Computer Vision*. Springer, 44–57.

[2] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. 2017. Rethinking atrous convolution for semantic image segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17)*.

[3] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. 2018. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision*. 801–818.

[4] Wuyang Chen, Xinyu Gong, Xianming Liu, Qian Zhang, Yuan Li, and Zhangyang Wang. 2019. FasterSeg: Searching for faster real-time semantic segmentation. In *International Conference on Learning Representations*.

[5] Wuyang Chen, Ziyu Jiang, Zhangyang Wang, Kexin Cui, and Xiaoning Qian. 2019. Collaborative global-local networks for memory-efficient segmentation of ultra-high resolution images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8924–8933.

[6] Xinghao Chen, Yunhe Wang, Yiman Zhang, Peng Du, Chunjing Xu, and Chang Xu. 2020. Multi-task pruning for semantic segmentation networks. In *2022 IEEE International Conference on Multimedia and Expo (ICME)*. 1–6.

[7] Ting-Wu Chin, Ruizhou Ding, and Diana Marculescu. 2019. Adascale: Towards real-time video object detection using adaptive scaling. *Proceedings of Machine Learning and Systems* 1 (2019), 431–441.

[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. 2016. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3213–3223.

[9] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. 2016. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[10] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. 2017. Deformable convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*. 764–773.

[11] Yadolah Dodge and Daniel Commenges. 2006. *The Oxford Dictionary of Statistical Terms*. Oxford University Press on Demand.

[12] Andries P. Engelbrecht, L. Fletcher, and Ian Cloete. 1999. Variance analysis of sensitivity information for pruning multilayer feedforward neural networks. In *Proceedings of the International Joint Conference on Neural Networks. Proceedings*. IEEE, 1829–1833.

[13] Li et.al. 2017. Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017). DOI: https://doi.org/10.1109/cvpr.2017.684

[14] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2010. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision* 88, 2 (2010), 303–338.

[15] Alex Gain and Hava Siegelmann. 2019. Relating information complexity and training in deep neural networks. In *Proceedings of the Micro-and Nanotechnology Sensors, Systems, and Applications XI*. International Society for Optics and Photonics, 109822H.

[16] Song Han, Huizi Mao, and William J. Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations (ICLR)*.

[17] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. In *Proceedings of the Advances in Neural Information Processing Systems*. 1135–1143.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[19] Tong He, Chunhua Shen, Zhi Tian, Dong Gong, Changming Sun, and Youliang Yan. 2019. Knowledge adaptation for efficient semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 578–587.

[20] Yihui He, Xiangyu Zhang, and Jian Sun. 2017. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*. 1389–1397.

[21] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2017. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research* 18, 1 (2017), 6869–6898.

[22] Angela H. Jiang, Daniel L. K. Wong, Giulio Zhou, David G. Andersen, Jeffrey Dean, Gregory R. Ganger, Gauri Joshi, Michael Kaminksy, Michael Kozuch, Zachary C. Lipton, and Padmanabhan Pillai. 2019. Accelerating deep learning by focusing on the biggest losers.

[23] Pouria Khanzadi, Babak Majidi, and Ehsan Akhtarkavan. 2017. A novel metric for digital image quality assessment using entropy-based image complexity. In *Proceedings of the 2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation*. IEEE, 0440–0445.

[24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Proceedings of the European Conference on Computer Vision*. Springer, 740–755.

[25] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. 2017. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*. 2736–2744.

[26] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. 2017. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE International Conference on Computer Vision*. 5058–5066.

[27] Sangkug Lym, Esha Choukse, Siavash Zangeneh, Wei Wen, Sujay Sanghavi, and Mattan Erez. 2019. PruneTrain: Fast neural network training by dynamic sparse model reconfiguration. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–13.

[28] Dmitrii Marin, Zijian He, Peter Vajda, Priyam Chatterjee, Sam Tsai, Fei Yang, and Yuri Boykov. 2019. Efficient segmentation: Learning downsampling near semantic boundaries. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2131–2141.

[29] James Clerk Maxwell. 1860. V. Illustrations of the dynamical theory of gases.-Part I. On the motions and collisions of perfectly elastic spheres. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 19, 124 (1860), 19–32.

[30] Suraj Mishra, Peixian Liang, Adam Czajka, Danny Z. Chen, and X. Sharon Hu. 2019. CC-NET: Image complexity guided network compression for biomedical image segmentation. In *Proceedings of the 2019 IEEE 16th International Symposium on Biomedical Imaging*. IEEE, 57–60.

[31] NVIDIA Inc. [n.d.]. NVIDIA Jetson TX2. Retrieved from https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-tx2/, accessed 2019-09-01.

[32] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. 2016. Enet: A deep neural network architecture for real-time semantic segmentation. arXiv:1606.02147. Retrieved from https://arxiv.org/abs/1606.02147.

[33] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *Proceedings of the NIPS-W*.

[34] Antonio Polino, Razvan Pascanu, and Dan Alistarh. 2018. Model compression via distillation and quantization. In *International Conference on Learning Representations*.

[35] Alex Renda, Jonathan Frankle, and Michael Carbin. 2020. Comparing rewinding and fine-tuning in neural network pruning. In *Proceedings of the International Conference on Learning Representations*.

[36] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4510–4520.

[37] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2020. Energy and Policy Considerations for Modern Deep Learning Research. *Proceedings of the AAAI Conference on Artificial Intelligence* 34 (Apr. 2020), 13693–13696. https://doi.org/10.1609/aaai.v34i09.7123

[38] Gregory K. Wallace. 1992. The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics* 38, 1 (1992), xviii–xxxiv.

[39] Yue Wang, Ziyu Jiang, Xiaohan Chen, Pengfei Xu, Yang Zhao, Yingyan Lin, and Zhangyang Wang. 2019. E2-train: Training state-of-the-art CNNs with over 80% less energy. In *Proceedings of the Advances in Neural Information Processing Systems*.

[40] Zuxuan Wu, Caiming Xiong, Chih-Yao Ma, Richard Socher, and Larry S. Davis. 2019. Adaframe: Adaptive frame selection for fast video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1278–1287.

[41] Ran Xu, Rakesh Kumar, Pengcheng Wang, Peter Bai, Ganga Meghanath, Somali Chaterji, Subrata Mitra, and Saurabh Bagchi. 2021. ApproxNet: Content and Contention-Aware Video Object Classification System for Embedded Clients. *ACM Trans. Sen. Netw.* 18, 1 (2021).

[42] Tien-Ju Yang, Yu-Hsin Chen, and Vivienne Sze. 2017. Designing energy-efficient convolutional neural networks using energy-aware pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5687–5695.

[43] Shaokai Ye, Tianyun Zhang, Kaiqi Zhang, Jiayu Li, Kaidi Xu, Yunfei Yang, Fuxun Yu, Jian Tang, Makan Fardad, Sijia Liu, et al. 2018. Progressive weight pruning of deep neural networks using ADMM. arXiv:1810.07378. Retrieved from https://arxiv.org/abs/1810.07378.

[44] Haoran You, Chaojian Li, Pengfei Xu, Yonggan Fu, Yue Wang, Xiaohan Chen, Yingyan Lin, Zhangyang Wang, and Richard G. Baraniuk. 2019. Drawing early-bird tickets: Towards more efficient training of deep networks. In *International Conference on Learning Representations*.

[45] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. 2018. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European Conference on Computer Vision*. 325–341.

[46] Fisher Yu and Vladlen Koltun. 2016. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations (ICLR)*.

[47] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. 2018. Bdd100k: A diverse driving video database with scalable annotation tooling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'20)*.

[48] H. Yu and S. Winkler. 2013. Image complexity and spatial information. In *Proceedings of the 2013 5th International Workshop on Quality of Multimedia Experience*. 12–17. DOI: https://doi.org/10.1109/QoMEX.2013.6603194

[49] Hong Zhang, Jiongwei Dong, Shasha He, and Shengze Lv. 2020. Research on image complexity description method based on approximate entropy. In *Proceedings of the 2020 International Conference on Intelligent Transportation, Big Data & Smart City*. IEEE, 918–920.

[50] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. 2018. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6848–6856.

[51] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. 2018. Icnet for real-time semantic segmentation on high-resolution images. In *Proceedings of the European Conference on Computer Vision*. 405–420.

[52] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. 2017. Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2881–2890.

[53] Elizabeth Y. Zhou, Claudia Damiano, John Wilder, and Dirk B. Walther. 2019. Measuring complexity of images using multiscale entropy. *Journal of Vision* 19, 10 (2019), 96a–96a.

Received 15 July 2021; revised 23 December 2021; accepted 8 January 2022