Surrogate-assisted Multiobjective Neural Architecture Search for Real-time Semantic Segmentation

Zhichao Lu, *Member, IEEE*, Ran Cheng, *Senior Member, IEEE*, Shihua Huang, Haoming Zhang, Changxiao Qiu, and Fan Yang

***Abstract*—The architectural advancements in deep neural networks have led to remarkable leap-forwards across a broad array of computer vision tasks. Instead of relying on human expertise, neural architecture search (NAS) has emerged as a promising avenue towards automating the design of architectures. While recent achievements on image classification have suggested opportunities, the promises of NAS have yet to be thoroughly assessed on more challenging tasks of semantic segmentation. The main challenges of applying NAS to semantic segmentation arise from two aspects: i) high-resolution images to be processed; ii) additional requirement of real-time inference speed (i.e. real-time semantic segmentation) for applications such as autonomous driving. To meet such challenges, we propose a surrogate-assisted multi-objective method in this paper. Through a series of customized prediction models, our method effectively transforms the original NAS task to an ordinary multi-objective optimization problem. Followed by a hierarchical pre-screening criterion for in-fill selection, our method progressively achieves a set of efficient architectures trading-off between segmentation accuracy and inference speed. Empirical evaluations on three benchmark datasets together with an application using Huawei Atlas 200 DK suggest that our method can identify architectures significantly outperforming existing state-of-the-art architectures designed both manually by human experts and automatically by other NAS methods. Code is available from here.**

***Impact Statement*—In the recent past, the emerging research in neural architecture search has promoted increasing applications in the industry to automate the designs and deployments of deep learning models. However, the heavy computational burdens and the lack of ability to handle multiple competing objectives (e.g. prediction accuracy, inference speed, etc.) are the key obstacles restraining its outreach to small business and beyond simple image classification tasks. To meet these challenges, we have proposed a surrogate-assisted multi-objective neural architecture search method. On the challenging task of semantic segmentation (one of the key underlining tasks for autonomous driving, medical imaging, etc.), we have demonstrated that a set of representative neural network models trading-off between segmentation accuracy and inference speed can be efficiently obtained as an automated end-to-end process by general users with limited computation resources. Our work creates potential opportunities to democratize the applications of deep learning technologies.**

***Index Terms*—Neural Architecture Search, Semantic Segmentation, Multi-objective Optimization, Evolutionary Algorithm.**

Z. Lu and S. Huang were with the Guangdong Key Laboratory of Brain-Inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China. E-mail: {luzhichaocn, shihuahuang95}@gmail.com.

R. Cheng and H. Zhang are with the Guangdong Key Laboratory of Brain-Inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China. E-mail: {ranchengcn, small.min.zhm}@gmail.com (*Corresponding author: Ran Cheng*).

C. Qiu and F. Yang are with Hisilicon Research Department, Huawei Technologies Co., Ltd., Shenzhen 518055, China. E-mail: {qiuchangxiao, yangfan74}@huawei.com

This work was supported by the National Natural Science Foundation of China (No. 62106097, 61906081), China Postdoctoral Science Foundation (No. 2021M691424), and the Guangdong Provincial Key Laboratory (No. 2020B121201001).

I. Introduction

DEEP Convolutional Neural Networks (CNNs) have been proven effective across a wide-range of machine learning tasks, including object recognition [1], speech recognition [2], language processing [3], decision making [4], etc. While improvements in computing hardware and training techniques have certainly contributed, architectural advancements have been undeniably the core driving forces behind these successes. Early progress primarily relies on skilled practitioners and elaborated designs. In computer vision, this manual process has led to designs such as VGG [5], ResNet [6], DenseNet [7] for object classification, and Faster R-CNN [8], DeepLab [9] for object detection and segmentation.

More recently, there has been a surge of interest in automating the design of network architectures. It is well-recognized now that the manual design process is a computationally impractical endeavor with respect to the increasing application scenarios of CNNs and many other types of deep neural networks [10]. Notably, neural architecture search (NAS) has emerged as a trending research topic for both academia and industries, as automatically generated network models exceeding the performance of manually designed ones on large-scale image classification problems [11], [12]. While the recent algorithmic development of NAS on object classification has established a promising starting point for computer vision tasks, the potential of NAS has yet to be fully assessed on more challenging and demanding tasks, such as semantic segmentation [13].

The task of semantic segmentation performs classification at pixel-level–i.e., assigning a class label to every pixel in an input image. Thereby, in addition to the higher-level contextual information which is solely sufficient for object classification, semantic segmentation also requires rich spatial details to group pixels based on their semantic categories. Modern deep learning based methods typically follow the process shown in Fig. 1. A CNN encoder (also referred as *backbone*) is adopted to first extract features at multiple spatial resolutions,

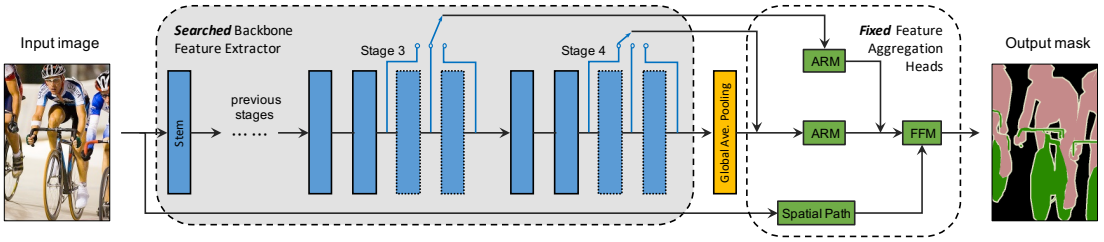

Fig. 1: Search Space Overview: The design of our search space follows the encoder-decoder convention. Each searchable architecture comprises a backbone CNN, as the encoder, for extracting features at multiple scales; and a decoder module to aggregate the extracted multi-scale features. Towards making the search tractable, we fix the decoder to the efficient segmentation heads from BiSeNet [14] and focus on searching the architectural settings of the backbone CNN, including the input image scale, the No. of layers and output channels for each stage, and the No. of intermediate channels for each layer in the stage.

followed by another CNN decoder to aggregate these multi-scale features and output a prediction mask. The need of multi-scale representation and the necessity to operate on high resolution imagery result in steep computational requirement in evaluating the performance of an architecture for segmentation, hindering the utility of conventional simulation-based optimization means. Particularly, many important real-world scenarios of semantic segmentation impose rigorous speed requirements on CNN models to be practically useful, leading to the need of considering multiple objectives simultaneously. This special category is commonly referred as *real-time* semantic segmentation (i.e., CNN models with inference speed of more than 30 images per second are typically considered to be real-time [14], [15]), with applications ranging from autonomous driving in navigation [16] to precision farming in agriculture [17].

Existing NAS methods for semantic segmentation either (i) opt for problem reformulation via continuous relaxation, which allows architecture parameters to be jointly optimized with the weights, to search the entire architecture all at once [18], [19]; or (ii) only search the decoder architecture while inheriting a generic backbone CNN from object classification [20]. Despite the efficiency provided from the continuous relaxation, the reformulated problem tends to bias architectures with faster convergence, leading to potentially sub-optimal architectures, as faster converging architectures need not necessarily be the ones that also generalize better [21]. Moreover, incorporating additional application-related objectives in continuous relaxation based methods, which rely on gradients to update architectural parameters, is not straightforward.

Concurrently, evolutionary algorithms (EAs) have earned a plethora of attention in NAS [22], [23], [24], [25], [26]. In an EA, a set of solutions are processed in parallel to approximate the optimal architectures, where the selection is carried out on the relative differences among solutions, obviating the need of gradient estimation from continuous relaxation. Despite that the population-based nature of EAs enables a flexible extension to handle multiple objectives, direct porting ideas from the current literature of EAs would not suffice the requirements of NAS for real-time semantic segmentation – the computational cost caused by the performance evaluations of architectures obtained in every generation can be prohibitively expensive.

In this paper, we propose *MoSegNAS* to breach this steep computational barrier and present the first attempt of tailoring evolutionary multi-objective optimization based NAS for real-time semantic segmentation, with the assistance of surrogate modeling. Through a series of customized prediction models, MoSegNAS effectively transforms the NAS task to an ordinary multi-objective optimization problem, such that existing multi-objective evolutionary algorithms can be applied to obtaining solutions (i.e. architectures) trading-off between segmentation accuracy and inference speed. A partial set from the obtained solutions is selected via a hierarchical pre-screening criterion, and then used to refine the learned prediction models. These two steps are alternated in iterations until a computation budget is exhausted. The key contributions are summarized below:

- We introduce MoSegNAS as an alternative to existing NAS methods for semantic segmentation. Instead of replying on estimated gradients from continuous relaxation, we advocate for a surrogate-assisted evolutionary multi-objective framework.
- MoSegNAS adopts an online surrogate model for predicting the segmentation accuracy. With a sequence of modifications introduced to a multi-layer perceptron, we demonstrate that an indicative predictor can be efficiently learned with few hundreds of samples.
- MoSegNAS adopts an offline surrogate model, in the form of a look-up table, for predicting inference latency. Considering the fact that high-fidelity evaluation of latency is an-order-of-magnitude cheaper than segmentation accuracy in terms of simulation time, MoSegNAS is further equipped with a customized hierarchical pre-screening criterion for in-fill selection.
- We demonstrate that MoSegNAS leads to state-of-the-art performance on real-time semantic segmentation, achieving higher accuracy and inference speed on Cityscapes dataset [16] than architectures designed both manually and by other NAS methods. Under transfer learning setup, we demonstrate that the obtained models from MoSegNAS also lead to state-of-the-art performance on COCO-Stuff-10K [27] and PASCAL VOC 2012 [13] datasets.
- We demonstrate the practical utility of MoSegNAS in designing hardware-dependent models on an application using Huawei Atlas 200 DK, where the obtained models consistently outperform existing models across a spectrum of inference latency.

The remainder of the paper is organized as follows. Section II summarizes the related literature. The proposed method

is elaborated in Section III. In Section IV, we describe the experimental setup to validate our method along with discussions of the results, followed by ablative experiments and an application study in Sections V. Section VI presents the results of applying our method to Huawei Atlas 200 Developer Kit. Finally, we conclude with a brief summary of the proposed method and our findings in Section VII.

II. Related Work

NAS has been overwhelmingly successful in the recent past for automatically customizing network models that are tailored to the task at hand. The early development of NAS primarily concentrated on the fundamental task in computer vision – i.e., image classification. Relying on reinforcement learning (RL), continuous relaxation (i.e. gradient), or evolutionary algorithms (EAs), impressive empirical results surpassing human-level performance have been reported on various image classification benchmarks [11]. Moving beyond classification, many recent attempts aiming to adapt NAS methods to higher-level tasks have been reported. Chen el at. [28] presented the first effort of applying the RL-based NAS method proposed in [12] to the domain of dense image prediction. Similarly, NAS-FPN [29] learned a recurrent neural network (RNN) as a controller to automatically design the architectures of feature pyramid networks for object detection. DetNAS [30] used NAS to demonstrate the importance of backbone architecture design for object detection. Auto-DeepLab proposed a hierarchical search space that allows both macro and micro structures of a network to be jointly optimized, achieving state-of-the-art results on semantic segmentation [18]. In the remainder of this section, we provide a brief overview of methods closely related to the technical aspects of this paper.

Surrogate Modelling: The main computation bottleneck of NAS resides in the performance evaluations of architectures. Many surrogate-based methods have been proposed to expedite the evaluation of architectures for image classification. In general, existing methods can be broadly classified into two categories. The first category focuses on mitigating the computation cost of a single high-fidelity evaluation by reducing the number of training epochs that are required before the performance of architecture can be assessed. Common methods include i) weight sharing that allows offspring architectures to inherit weights from parents or a super network as a warm-start [31], [32], [33]; ii) early stop that terminates the training process at an earlier phase by either heuristics [12], [34] or extrapolation [35]; iii) partial training samples that lead to a reduction in epochs required for training to converge [25].

The second category focuses on reducing the number of high-fidelity evaluations required to drive their algorithms towards optimal architectures. Methods in this group mostly resort to learning accuracy predictors [36]. E2EPP [37] adopted an off-line random forest-based surrogate model to directly predict performance from architectural encoding. OnceForAll [38] used an MLP to predict accuracy, where the surrogate model is also learned in an offline manner, thus requiring a large number of training samples. ChamNet [39] improved the sample efficiency of offline surrogate modeling by selectively sampling architectures with diverse complexity (FLOPs, latency, energy, etc.) to construct the accuracy predictor. Further improvement introduced in NSGANetV2 [33] adopted online surrogate modeling that used architecture search to guide the construction of the accuracy predictor, thus substantially reducing the number of training samples.

Despite the impressive advancement, the surrogate models adopted by most existing methods are generic or even simplistic. In contrast, MoSegNAS applied a series of enhancements (sparse encoding, ranking loss, synthetic data, etc.) to a conventional MLP model, demonstrating the first attempt to predict the segmentation performance of neural architectures using surrogate models.

Multi-objective NAS: A plethora of hardware-dependent NAS works has been introduced lately for image classification [40], [34], [41]. These methods sought to improve models' computational efficiency (i.e. inference latency, power consumption, memory footprint) on specific hardware while trading-off classification accuracy to a small extent. A common theme behind these methods is to adopt a scalarized objective function or an additional constraint to encourage high accuracy and penalize compute inefficiency at the same time. CAS [19] studied various resource-related constraints and demonstrated the utility of NAS on real-time semantic segmentation. It applied continuous relaxation to an extended cell-based search space [32], allowing the architectural parameters to be optimized jointly with weights via stochastic gradient descent. As a follow-up work, FasterSeg [42] further reduced the run-time latency of the searched models by incorporating prior wisdom (e.g. multi-scale branches [43] and dilated convolution [44]) to the search space.

Conceptually, the search of architectures (in the aforementioned works) is still guided by a single objective and thus only one architecture is obtained per search. Empirically, multiple runs with a different weighting of the objectives are required to find an architecture with the desired trade-off, or multiple architectures with different complexities. Concurrently, another streamlining of multi-objective NAS works for image classification emerged, aiming to approximate the entire Pareto-optimal front simultaneously in a single run [45], [46], [47], [48], [49]. These methods utilize heuristics and relative differences among individuals in a population to efficiently navigate through the search space, allowing practitioners to visualize and choose a suitable model *a posteriori* for the search. Our proposed MoSegNAS, belonging to this category, is the first tailored multi-objective NAS method for real-time semantic segmentation.

III. Proposed Method

As summarized in Algorithm 1, MoSegNAS employs a series of surrogate modeling techniques to expedite the search of architectures. First, it learns a surrogate model, in the form of a hypernetwork, to approximate the reaction set mapping, through which the inner optimization loop over the network weights is obviated. Then, in each iteration of the search, an accuracy predictor is learned from previously evaluated architectures to rank newly generated offspring. Jointly with

Algorithm 1: General framework of MoSegNAS

Input : Training data $\mathcal{D}_{trn}$, validation data $\mathcal{D}_{vld}$, initial population size N, max. # of generations T, # of high-fidelity eval. per gen. K.

$t \leftarrow 0$ // initialize an iteration counter.
$\mathcal{A} \leftarrow \emptyset$ // initialize an archive to store all evaluated architectures.
$\mathcal{H} \leftarrow$ train a surrogate model, *hypernetwork*, to generate weights for candidate architectures (see Sec. III-C).
$LuT \leftarrow$ construct a surrogate model, *look-up table*, to predict latency (see Sec. III-D).
// Initialize the parent population (see Sec. in the supp. materials).
$P, F_p, \mathcal{A} \leftarrow Initialization(N, \mathcal{H}, \mathcal{D}_{vld}, \mathcal{A})$
while $t < T$ **do**
  // Construct a surrogate model to predict accuracy.
  $\tilde{f} \leftarrow ConsAccPred(\mathcal{A})$ ◁ Algo. 3
  // Generate offspring population using a MOEA (e.g., NSGA-II [50], MOEA/D [51]) with surrogate objectives.
  $Q \leftarrow$ MOEA($\tilde{f}$, LuT)
  // Select top-K offspring for high-fidelity evaluation
  $Q^* \leftarrow PreScreen(Q, P, K)$ ◁ Algo 4
  // Evaluate the selected offspring.
  $F_q \leftarrow Evaluate(Q^*, \mathcal{H}, \mathcal{D}_{vld})$ ◁ Algo 2
  $\mathcal{A} \leftarrow \mathcal{A} \cup (Q^*, F_p)$ // archive the evaluated offspring.
  $t \leftarrow t + 1$ // repeat above steps for T generations.
end
$P^* \leftarrow$ choose top-ranked architectures based on trade-offs (see Sec. 1 in supp. materials).
Return P^*

the latency look-up table (i.e. a learned surrogate model to predict the inference speed of a network), an auxiliary problem is constructed. Afterward, the optimization outcomes from a standard multi-objective evolutionary algorithm (MOEA) become the candidate architectures, from which a subset is selected for (high-fidelity) evaluation and archived. At the end of this evolutionary process, architectures with the best trade-offs will be selected as outputs. In the remainder of this section, we first formally define the problem and introduce the encoding strategy in Section III-A and III-B respectively, followed by detailed descriptions on each component of MoSegNAS in Sections III-C–III-E.

A. Problem Formulation

In this work, we approach the task of real-time semantic segmentation from a multi-objective perspective and mathematically formulate the problem as a bilevel optimization problem to simultaneously maximize segmentation accuracy and inference speed. For a targeted dataset $\mathcal{D} = \{\mathcal{D}_{trn}, \mathcal{D}_{vld}, \mathcal{D}_{tst}\}$, the problem is defined as:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{Maximize}} \quad & \Big\{ f_1\big(\boldsymbol{x}; \boldsymbol{w}^*\big),\ f_2\big(\boldsymbol{x}\big) \Big\}, \\ \text{subject to} \quad & \boldsymbol{w}^* \in \arg\min_{\boldsymbol{w}}\ \mathcal{L}oss\big(\boldsymbol{w}; \boldsymbol{x}\big), \\ & \boldsymbol{x} \in \boldsymbol{\Omega}_x, \quad \boldsymbol{w} \in \boldsymbol{\Omega}_w. \end{aligned} \tag{1}$$

where $\boldsymbol{\Omega}_x = \Pi_{i=1}^{n}[a_i, b_i] \subseteq \mathbb{Z}^n$ is the architecture decision space; a_i, b_i are the lower and upper bounds, respectively. The variables $\boldsymbol{x} = (x_1, \ldots, x_n)^T \in \boldsymbol{\Omega}_x$ define a candidate architecture, and the variables $\boldsymbol{w} \in \boldsymbol{\Omega}_w \subseteq \mathbb{R}^m$ denote its associated weights. We use $f_1(\cdot)$, $f_2(\cdot)$ to denote the segmentation accuracy on the validation data $\mathcal{D}_{vld}$ and the inference speed on a specific hardware, respectively. $\mathcal{L}oss(\boldsymbol{w}; \boldsymbol{x})$ is the training loss of an architecture $\boldsymbol{x}$ on the training data $\mathcal{D}_{trn}$. We reserve $\mathcal{D}_{tst}$ for comparison with other methods.

The hierarchical nature of the problem presented in (1) necessitates a nested loop of optimizations, i.e., an inner optimization over the network weights $\boldsymbol{w}$ for a given architecture $\boldsymbol{x}$ and an outer optimization over the network architectural variables $\boldsymbol{x}$ themselves. It is worth noting that the bilevel optimization, on its own, is a challenge research topic [52], [53], and it is beyond the main scope of this paper.

B. Search Space and Encoding

In line with prior wisdom, we follow the encoder-decoder architectural framework and leverage the BiSeNet [14] heads as the decoder, and focus on designing the encoder. A pictorial illustration is provided in Fig. 1. We start with the premise of constructing a search space that may express most of the well-established backbone CNNs for segmentation tasks [8]. The overall structure of our encoder network consists of a stem, four stages and a tail of global averaging pooling. The *stem* and *stages* are searched, while the *tail* is handcrafted and common to all networks. Each *stage* comprises of multiple layers, and each *layer* follows the residual bottleneck structure, i.e., a sequence of a 1×1 convolution for compression, a 3×3 convolution, and another 1×1 convolution for expansion [6].

MoSegNAS then searches over four important dimensions constituting the design of an encoder network, including the input image scale, the width (in terms of # of channels), the depth (in terms of # of layers), and the locations for outputting features. We encode these choices in an integer-valued string, and pad zeros to avoid variable-length representations (as the depth of each stage is also searched). With one digit for input scale, two digits for stem, six digits for each stage (except stage 3 where it uses eight digits), the total length of the encoding is 29. The resulting search space contains approximately 1×10^{14} unique architecture designs. It is worth mentioning that we use one-shot encoding to sparsify the integer string representation for learning the accuracy surrogate model, which is elaborated in the following subsections.

Readers can refer to the supplementary materials for a more detailed break-down of our search space in Table 1 and an example of our encoding in Fig. 1.

C. Surrogate Modeling for Segmentation Accuracy

Single-level Reduction. The main computational bottleneck of solving the problem in (1) stems from the fact that every evaluation of segmentation accuracy (f_2) invokes another optimization to be performed for learning the optimal weights ($\boldsymbol{w}^*$). Hence, as the first step to improve the efficiency of our approach, we aim to remove the inner loop of weights optimization via surrogate modelling. More specifically, let us consider an equivalent formulation of the problem, stated in terms of the reaction set mapping $\Psi : \mathbb{Z}^n \rightrightarrows \mathbb{R}^m$, as follows [52]:

$$\Psi(\boldsymbol{x}) = \underset{\boldsymbol{w} \in \boldsymbol{\Omega}_w}{\arg\min}\, \mathcal{L}oss\big(\boldsymbol{w}; \boldsymbol{x}\big), \tag{2}$$

which represents the constraint defined by the inner optimization problem, i.e., $\Psi(\boldsymbol{x}) \subset \boldsymbol{\Omega}_w$ for every $\boldsymbol{x} \in \boldsymbol{\Omega}_x$. The nested loop of optimizations in (1) can then be re-formulated as a constrained single-level optimization problem, as below:

$$\begin{aligned} \underset{\boldsymbol{x}\in\boldsymbol{\Omega}_x,\ \boldsymbol{w}\in\boldsymbol{\Omega}_w}{\text{Maximize}} \quad & \Big\{f_1\big(\boldsymbol{x};\boldsymbol{w}^*\big),\ f_2\big(\boldsymbol{x}\big)\Big\}, \\ \text{subject to} \quad & \boldsymbol{w}^* \in \Psi(\boldsymbol{x}). \end{aligned} \tag{3}$$

where Ψ can be interpreted as a parameterized constraint for the variables $\boldsymbol{w}$ of the original inner optimization in (1).

If the Ψ-mapping can somehow be determined, it effectively reduces a hierarchical problem to a regular single-level optimization, thereby, the costly iterations of the inner optimization are avoided. However, in the context of neural architecture search, the Ψ-mapping between architectures and their optimal weights is seldom defined analytically due to the non-linear nature of modern CNNs (e.g., non-linear activation and pooling functions). Thereby, we opt for the route of surrogate modelling the reaction set mapping in this work.

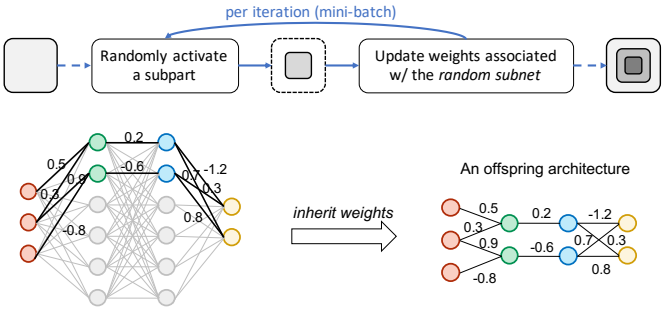

Fig. 2: Top: For learning a hypernetwork, a sub-part of the hypernetwork is randomly sampled during each iteration and only the associated parameters are updated via stochastic gradient descent. **Bottom:** With a trained hypernetwork, the segmentation accuracy of a candidate architecture is evaluated with the weights inherited from the hypernetwork.

Following recent advances in one-shot and weight sharing NAS works [31], we model the Ψ-mapping in the form of a hypernetwork $\mathcal{H}(\theta;\boldsymbol{x})$, which is trained (by optimizing its parameters θ) to generate weights conditioned on a given architecture [54]. In MoSegNAS, the $\mathcal{H}(\cdot)$ corresponds to the largest architecture[1] encoded in the search space, such that all searchable architectures become sub-parts. For learning the $\mathcal{H}(\cdot)$, a random sub-part ($\boldsymbol{x}_i$) is activated for forward passes during each iteration of training, and only the parameters associated with the activated sub-part ($\theta(\boldsymbol{x}_i)$) will be updated. The trained hypernetwork then becomes an approximation of the actual Ψ-mapping, from which the optimal weights ($\boldsymbol{w}^*$) of an architecture can be directly obtained. A pictorial illustration is provided in Fig 2. And the evaluation of segmentation accuracy becomes an inference on the validation data, as shown in Algorithm 2.

Segmentation Accuracy Prediction. Recall that the task of semantic segmentation assigns a class label to every pixel in an input image, which requires intermediate features to be processed at high-resolution in a CNN [43]. As a result, the cost of evaluating a network for semantic segmentation by forward inference is still considerably high due to a low batch size, as opposed to object classification (see Fig 3 for a visual comparison). Therefore, extensive inference by probing the approximated Ψ-mapping can still render the entire search computationally prohibitive. To mitigate this computational burden, we further develop a surrogate model to directly predict the segmentation accuracy of an architecture from its architectural parameters. By learning a functional relation between the architectural representations and the corresponding accuracy, our surrogate predictor disentangles the evaluation of an architecture from data-processing. Consequently, the cost of a single evaluation reduces from minutes to less than a second.

[1]The largest architecture is defined by setting the searched architectural parameters to their upper bounds.

Algorithm 2: Performance evaluation of architectures

Input : Individuals X, hypernetwork $\mathcal{H}$, validation data $\mathcal{D}_{vld}$.
$F \leftarrow \emptyset$ // initialize a list to store objective vectors.
for x *in* X **do**
  $net \leftarrow$ decode architecture x to a neural network.
  $w \leftarrow \mathcal{H}(x)$ // inherit weights from the trained hypernetwork.
  $t_o \leftarrow$ time() // initialize a latency timer.
  $acc \leftarrow net(w, \mathcal{D}_{vld})$ // evaluate segmentation accuracy on validation data using inherited weights directly.
  $lat \leftarrow$ (time() - t_o) / $length(\mathcal{D}_{vld})$ // measure latency.
  $F \leftarrow F \cup (acc, lat)$
end
Return F

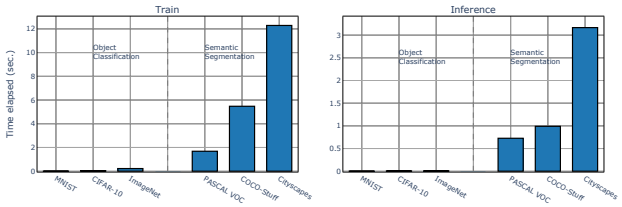

Fig. 3: Wall-clock time complexity comparison between object classification and semantic segmentation tasks. We record the time to process a batch of 64 images during training (*Left*) and inference (*Right*).

In this work, we adopt a multi-layer perceptron (MLP). For the purpose of regression, a MLP is conventionally trained with the mean square error (MSE) loss that minimizes the Euclidean distances between ground truth and network output values, as below:

$$\mathcal{L}_{mse} = \frac{1}{N}\sum_i \ \|\hat{f}(\boldsymbol{x}_i;\boldsymbol{\Theta}) - y_i\|_2^2, \tag{4}$$

where $\hat{y}_i = \hat{f}(\boldsymbol{x}_i;\boldsymbol{\Theta}) \in \mathbb{R}$ is the output of a neural network on the ith training sample $\boldsymbol{x}_i$, and $y_i \in \mathbb{R}$ is the ground truth label. N is the number of training samples. However, for an evolutionary algorithm (as in our case) where the selections are carried out on the relative differences among solutions, a low MSE is desirable but *not* necessary. In line with this derivation, we instead use a ranking loss to explicitly align the training of the MLP with the goal of maximizing the rank-order correlation in predictions, as follows:

$$\begin{aligned} \mathcal{L}_{rank} &= \frac{1}{2N}\sum_{i,j} max\big(0, \gamma - \delta(\hat{y}_i,\ \hat{y}_j)(y_i - y_j)\big), \\ \delta(\hat{y}_i, \hat{y}_j) &= \begin{cases} 1, & \text{if } \hat{y}_i > \hat{y}_j, \\ -1, & \text{otherwise.} \end{cases} \end{aligned} \tag{5}$$

where $\hat{y}_i = \hat{f}(\boldsymbol{x}_i; \boldsymbol{\Theta})$, $\hat{y}_j = \hat{f}(\boldsymbol{x}_j; \boldsymbol{\Theta})$ are the outputs of the MLP on the ith, jth training samples, respectively. While $y_i \in [0, 1]$, $y_j \in [0, 1]$ are the normalized ground truth labels for the ith, jth training samples, respectively. To be consistent, we also map $\hat{y}_i$ and $\hat{y}_j$ to $[0, 1]$ via a sigmoid function. $\delta(\hat{y}_i,\ \hat{y}_j)$ is an utility function that indicates the relative ranking of the two inputs. And γ is a hyperparameter controlling the margin. A similar concept can be found in the step of maximum-margin classification in a support vector machine. With the added ℓ_2-norm of the weights to prevent over-fitting, the loss that we use to backpropagate for training the MLP is defined as:

$$\mathcal{L}oss = \mathcal{L}_{rank} + \beta \|\boldsymbol{w}\|_2^2, \tag{6}$$

where β is the regularization parameter, which is set at a small value, e.g. 2.5×10^{-4}.

In addition to high rank-order correlation, another desired property of a surrogate model is *sample efficiency*. It is well-known in the literature that neural networks generalize well only when there is sufficient amount of data. To overcome this barrier without excessive high-fidelity evaluations, we focus on generating and using synthetic data to complement the training of the MLP under limited genuine data. Inspired by the teacher-student knowledge distillation [55], we first train a set of widely-used surrogate models (e.g., Radial Basis Function (RBF), Decision Tree (DT), Gradient Boosting (GB) [56], etc.) as teachers prior to the MLP training. In each iteration of the subsequent training, we randomly sample a small set of extra data with labels predicted by these teachers. We then augment the original data with these synthetic data and proceed to the stochastic gradient descent. For reference, we name the proposed accuracy predictor as *RankNet* and the pseudocode outlining the process of building RankNet is provided in Algorithm 3.

Algorithm 3: Segmentation accuracy predictor (RankNet)

Input : Archive containing all past evaluated solutions $\mathcal{A}$, # of epochs T, weight decay λ, initial learning rate η_o.

$\hat{X}$, $\hat{Y} \leftarrow$ use solutions from $\mathcal{A}$ as the true training data and labels.
$\hat{X} \leftarrow$ one-hot encodes the training data $\hat{X}$.
// Prepare a set of surrogate models for synthetic data generation.
$\{rbf,\ dt,\ gb\} \leftarrow$ fit each model from $\hat{X}$ and $\hat{Y}$.
$\tilde{f} \leftarrow$ initialize a neural network with random weights w.
$t \leftarrow 0$ // initialize an epoch counter.
while $t < T$ **do**
  $\eta \leftarrow \frac{1}{2}\eta_o\left(1 + \cos\left(\frac{t}{T}\pi\right)\right)$ // anneal the learning rate.
  // Enrich training set by data with synthetic labels.
  $\tilde{X} \leftarrow$ randomly sample a small set of data.
  $\tilde{Y} \leftarrow$ use predictions from $\{rbf,\ dt,\ gb\}$ as labels for $\tilde{X}$.
  $X \leftarrow \hat{X} \cup \tilde{X}$; $Y \leftarrow \hat{Y} \cup \tilde{Y}$
  // Train $\tilde{f}(w)$ with the proposed ranking loss.
  $\mathcal{L} \leftarrow$ calculate the loss on $(X,\ Y)$ following Eq (6).
  $\nabla w \leftarrow$ Compute the gradient by $\partial\mathcal{L}/\partial w$.
  $w \leftarrow (1 - \lambda)w - \eta\nabla w$ // stochastic gradient descent.
  $t \leftarrow t + 1$
end
Return Trained accuracy predictor $\tilde{f}(w)$.

D. Surrogate Modeling for Latency

To design architectures with real-time performance, we need to include the run-time latency as an additional objective to be minimized. However, getting a stable measurement of the latency is not straightforward, as it critically depends on the operating conditions of the hardware, e.g., current loading, ambient temperature, etc. To circumvent the issue of inconsistent latency readings and avoid replicating the exact computing environment for every node that one may wish to parallelize the evaluations on, we adopt the look-up table approach [40].

Recall that the architectures encoded in our search space follow feedforward sequential connections at the layer level, i.e., no connections skipping over intermediate layers or an input feeding into multiple layers in parallel. The expected latency of an architecture can be approximated as:

$$\mathbb{E}[\text{latency}] = \sum_{l \in L} \underbrace{lat(m_l, e_l) + lat_s(d_0, m_0)}_{\substack{\text{encoder, stem} \\ \text{(searched)}}} + \underbrace{lat_t + lat_d}_{\substack{\text{tail, decoder} \\ \text{(common to all)}}}, \tag{7}$$

where L is the set of active layers specified by the *depth* encoding; $lat(m_l, e_l)$ is the latency of the lth layer of the searched encoder network, depending on the output size m_l and the hidden size e_l. Since the tail, and decoder networks are commonly shared among all searchable architectures, the latency induced by these components act as constant factors to the latency calculation.

The latency of each layer is measured by enumerating over all combinations of m_l and e_l settings (nine options in total assuming independence among layers). Then the measured latency is stored in a *look-up table*, through which the latency of a layer under any setting can be queried via a designated key. This approach essentially decouples hardware inference from latency computation, allowing the search of optimal architectures to be distributed across multiple nodes without sophisticated control of compute environment. A performance overview of the constructed look-up table is provided in Fig. 4.

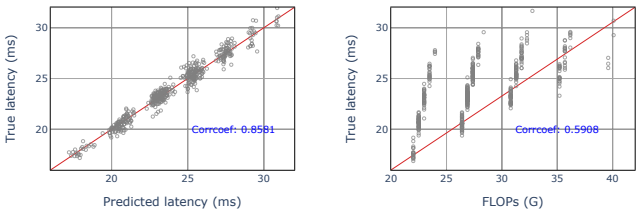

Fig. 4: Left: Performance of our latency look-up table. The Kendall rank correlation coefficient τ is provided along with the fitted linear model (red line). Correlation from the number of floating-point operations, FLOPs (G), is also provided for comparison (*Right*).

E. Pre-screening

With the constructed surrogate accuracy predictor RankNet (Section III-C) and the latency look-up table (Section III-D), the problem presented in Eq (1) essentially becomes a numerical multi-objective optimization problem with discrete variables, which can be exhaustively optimized by conventional multi-objective evolutionary algorithms (MOEAs) in minutes.

Algorithm 4: Hierarchical pre-screening procedure

Input : Candidate solutions Q, parent population P, # of solutions to be selected K.

// First select solutions based on (high-fidelity evaluated) latency.
$P^* \leftarrow$ get non-dominated (ND) solutions from P via ND sorting.
$Lat_{P^*} \leftarrow$ get latency of ND parent solutions.
$Lat_Q \leftarrow$ (high-fidelity) evaluate the latency of candidate solutions.
$Q^* \leftarrow$ select a subset from Q by solving the problem in Eq 8.
// Then fill the remaining slots based on (predicted) accuracy.
if $|Q^*| < K$ **then**
  $\tilde{Q} \leftarrow$ sort Q in descending order by predicted accuracy.
  for q *in* $\tilde{Q}$ **do**
    if $q \notin Q^*$ **then** $Q^* \leftarrow Q^* \cup q$;
    if $|Q^*| \geq K$ **then** break;
  end
end
Return A selected subset for high-fidelity evaluations Q^*.

In this work, we adopt NSGA-II [50] to simultaneously optimize the predicted segmentation accuracy and the predicted latency (line 11 in Algo 1). At the conclusion of the NSGA-II search, a set of non-dominated architectures is returned. With the search efficiency in mind, a small subset of the returned candidate arechitectures can be evaluated (with high-fidelity). The procedure to optimally select the subset is widely studied in the surrogate modeling literature. However, most existing pre-screening (also referred as infill criteria) techniques are assuming equal evaluation-cost of both objectives. While in our case, (high-fidelity) evaluation of one of the objectives (i.e. latency) is significantly cheaper than the other (i.e., segmentation accuracy) in terms of simulation time. Thereby, a hierarchical pre-screening procedure is proposed with an inductive bias towards the latency objective.

As shown in Algorithm 4, the proposed pre-screening routine first evaluates the latency of all the candidate solutions. Then the solutions are selected solely based on the latency, to encourage uniformity along the latency objective axis. To quantitatively measure the uniformity, we use the Kolmogorov–Smirnov statistic (denoted as f_{k-s}), which computes a distance between the empirical cumulative distribution of samples and a reference distribution (i.e., a uniform distribution in our case). We formulate this problem as a subset selection problem, as follows:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{Minimize}} \quad & f_{k\text{-}s}\big(F(L \cdot \boldsymbol{x}), F_{\mathcal{U}}\big), \\ \text{subject to} \quad & |\boldsymbol{x}| \leq K; \quad x_i \in \{0, 1\}. \end{aligned} \tag{8}$$

where L represents the list of candidate solutions; $\boldsymbol{x}$ are binary decision variables where one indicates a solution is selected; $F(L \cdot \boldsymbol{x})$ is the empirical cumulative distribution function estimated from the selected solutions; $F_{\mathcal{U}}$ is the cumulative distribution function of the uniform distribution; K is a hyperparameter indicating the maximum # of solutions to be selected.

To efficiently solve this subset selection problem, we adopt a binary genetic algorithm with customized crossover and mutation operators to ensure that feasibility is always satisfied during solution generation steps[2]. In case the optimized number of selected candidate solutions is smaller than the pre-specification of K, the rest of the slots are filled according to the predicted segmentation accuracy.

[2]More details and pseudocodes are provided in the supplementary materials under Section 1.

F. Post-search Decision Making

The proposed multi-objective algorithm is expected to produce a set of non-dominated solutions trading-off between segmentation accuracy and inference speed. In absence of explicit user-preferences, we resort to *trade-off*, as an unsupervised metric, to select architectures. Assuming a minimization problem of M objectives, the trade-off of the i-th solution is calculated by the following steps: (1) we determine the nearest neighbors $B(i)$ of the i-th solution based on the Euclidean distance in the normalized objective space; (2) we calculate the loss and gain averaged over all neighbor solutions in $B(i)$; (3) then the trade-off is calculated as the ratio of the average loss to the average gain. This process is mathematically defined as follows:

$$\text{Trade-off}(i) = \max_{j \in B(i)} \frac{\mathbb{E}\big(Loss(i,j)\big)}{\mathbb{E}\big(Gain(i,j)\big)}, \tag{9}$$

where

$$\begin{aligned} \mathbb{E}\big(Loss(i,j)\big) &= \frac{\sum_{k=1}^{M} \max\big(0, f_k(j) - f_k(i)\big)}{\sum_{k=1}^{M} \big\{1 | f_k(j) > f_k(j)\big\}}, \\ \mathbb{E}\big(Gain(i,j)\big) &= \frac{\sum_{k=1}^{M} \max\big(0, f_k(i) - f_k(j)\big)}{\sum_{k=1}^{M} \big\{1 | f_k(i) > f_k(j)\big\}}. \end{aligned} \tag{10}$$

It is worth noting that the average loss is normalized with respect to the average gain. Accordingly, given a solution, a higher trade-off value indicates that it causes larger average loss in some objectives to make one (unit/normalized) average gain in other objectives to choose any of its nearest neighbors. Thereby, we select solutions with trade-off values that are significantly larger (e.g., over two standard deviations; from a statistical point of view) than others.

IV. Experiments

In this section, we first introduce the benchmark datasets and baselines studied in this work, followed by the implementation details. We then present the empirical results to evaluate the efficacy of MoSegNAS. Further analysis of the transferability of the obtained models is also studied.

A. Datasets

We consider three widely-studied benchmark datasets to evaluate our method, including Cityscapes [16], COCO-Stuff-10K [27], and PASCAL VOC 2012 [13]. These three datasets cover a diverse variety of images in terms of objects, scenes, and scales for semantic segmentation. Examples from these datasets are provided in Fig. 6 and Fig. 7 for visualization.

Cityscapes [16] is a large-scale dataset for the semantic understanding of urban street scenes. It is officially split into a training set of 2,975 images, a validation set of 500 images, and a (privately hosted) testing set of 1,525 images. The provided annotations (i.e., ground truth labels) include 30 classes, 19 of which are used for semantic segmentation.

The images in Cityscapes are of higher (and unified) spatial resolution of 1024×2048 pixels, which make the dataset a challenging task for real-time semantic segmentation. In this paper, we only use images with fine annotations[3] to train and validate our proposed method.

COCO-Stuff-10K [27] is a subset of 10K images from the original MS COCO dataset [57] with additional dense stuff annotations. COCO-Stuff-10K is also a challenging task for semantic segmentation as it contains 182 different semantic categories—91 categories each for thing and stuff classes, respectively. In this paper, we follow the official split of 9,000 images for training and 1,000 images for testing.

PASCAL VOC 2012 [13] is a relatively small-scale dataset containing 20 foreground object categories and one background class. To be consistent with prior works [18], we augment the original training set with the extra annotations provided by [58], resulting in 10,582 images (*train_aug*) in total for training.

B. Baselines

Representative semantic segmentation methods and models are selected from the literature to evaluate the effectiveness of the proposed method. The chosen peer competitors can be broadly classified into two categories depending on if the architectures are manually designed by skilled practitioners or automatically generated by algorithms. The first group of human engineered models includes BiSeNet [14], ESPNetv2 [59], and HRNet [43], etc. The second category consists of models that are optimized both by reinforcement learning (RL) based methods (e.g., AuxCell [20]) and continuous relaxation (gradient) based methods (e.g., Auto-DeepLab [18], CAS [19]).

For real-time semantic segmentation, the effectiveness of each model is evaluated by both the segmentation accuracy and the inference speed (i.e., reciprocal of latency). We adopt the metric of mean Intersection-over-Union (mIoU) to evaluate the segmentation accuracy. It computes the IoU for each semantic class, which is then averaged over classes. And the IoU metric measures the number of pixels common between the ground truth and predicted masks divided by the total number of pixels present across both masks, mathematically as below:

$$\text{mIoU} = \frac{1}{K+1} \sum_{i=0}^{K} \frac{p_{ii}}{\sum_{j=0}^{K} p_{ij} + \sum_{j=0}^{K} p_{ji} - p_{ii}}$$

where K denotes the total number of foreground classes; p_{ij} counts the number of pixels whose ground-truth labels are i, but are predicted as j.

C. Implementation Details

We pretrain the hypernetwork on ImageNet-1K [11] for 1,200 epochs, followed by 300 epochs of fine-tuning on Cityscapes [16]. To stabilize the hypernetwork training, the searched architectural options (e.g., # of layers, # of channels, etc.) are gradually activated during the training process [38].

[3]There are roughly 20,000 additional coarsely annotated training images available.

TABLE I: Summary of hyperparameter settings. Notations in the parentheses are in correspondence with the notations in Algorithm 1.

Category	Parameter	Setting
Global	population size (N)	300
	# of generations (T)	20
	# of high-fidelity eval. per gen. (K)	8
Accuracy predictor (RankNet)	# of training epochs	500
	# of layers / neurons	3 / 300
	weight decay	1×10^{-5}
	init. learning rate	8×10^{-4}
	margin γ	0.05
NSGA-II	population size	100
	# of generations	1,000
	crossover probability	0.9
	mutation probability	0.05

This procedure takes about a week on a server with eight Titan RTX GPUs. It's worth mentioning that the majority of the computation expense is from ImageNet-1K pretraining, which is a one-time cost that can be amortized over many search runs. For practicality consideration, the search is carried out on Cityscapes. A subset of 500 images is randomly separated from the training set to guide the evolution. We run MoSegNAS with two objectives: maximize the mIoU and inference speed. We use NSGA-II [50] as the MOEA to generate the offspring population (line 11 in Algorithm 1), and its hyperparameters are provided in the third section of Table I. This search itself takes less than a day on a single Titan RTX GPU card and is repeated five times to capture the variance in performance induced by the stochastic nature of the algorithm. We select and report the performance of the median run as measured by the hypervolume (HV) calculated from the reference point at the origin. The remaining hyperparameter settings for the overall algorithm are provided in the first section of Table I. And the hyperparameter settings for the proposed surrogate model are provided in the second section of Table I.

D. Results on Cityscapes

From the set of non-dominated solutions returned at the conclusion of the evolution, we choose five architectures —based on their trade-offs (following the procedures outlined in supplementary materials under Section 1). For reference, our obtained models are referred to as MoSegNets (Visualizations of the architectures are provided in Fig. 3 from the supplementary materials). For comparison with other peer methods, we re-train the weights of each model thoroughly from scratch. Our post-search training largely follows [14]: SGD optimizer with momentum 0.9 and weight decay 1×10^{-5}; batch normalization and size of eight images per GPU card. The learning rate decays following a "poly" schedule (i.e., $0.01 \times (1 - \frac{iter}{maxIter})^{0.9}$) from 0.01 to zero in $40K$ iterations[4]. The data augmentation settings include horizontal flip, scale, and crop to 1536×768. The scaling ratio is randomly selected from $\{0.75, 1.0, 1.25, 1.5, 1.75, 2.0\}$. We note that our reported inference speed is measured on a single Titan RTX GPU card without optimized inference acceleration

[4]Each iteration refers to each mini-batch.

TABLE II: Comparison with state-of-the-art models on Cityscapes [16]. Models are grouped into sections and the best result in each section is in bold. "RL" stands for reinforcement learning. * denotes the use of a high-performance inference framework *TensorRT* to measure speed. ‡ is an ensemble of multiple input scales.

	Model	Method	FLOPs ↓ (G)	Speed ↑ (imgs / sec)	mIoU ↑ (%)
real-time models	ESPNetV2 [59]	manual	-	-	66.4
	BiSeNet [14]		72	65.5	74.8
	SwiftNet [60]		104	39.9	75.4
	BiSeNetV2 [61]		78	47.3	75.8
	CAS [19]	gradient	-	**108***	71.6
	FNA [62]		**24**	40.8	76.6
	AuxCell [20]	RL	-	44.2	77.1
	MoSegNet$_{small}$	EA	35	73.2	75.9
	MoSegNet$_{large}$		42	50.1	**78.2**
large models	PSPNet [63]	manual	412	0.8	78.4
	DeepLabv3+ [44]		-	-	79.6
	HRNetV2 [43]		696	6.3	81.1
	Auto-DeepLab-S [18]	gradient	333	-	79.7
	MoSegNet$_{xlarge}$	EA	**128**	**29.2**	79.5
	MoSegNet$^{‡}_{xlarge}$		-	0.5	**81.3**

implementations (e.g., TensorRT[5]); and our reported mIoU is computed without test-time augmentation techniques (e.g., multi-crop, multi-scale) unless otherwise specified.

Table II presents the experimental results in terms of model efficiency (FLOPs and inference speed) and segmentation accuracy (mIoU) of the compared models. Specifically, Table II is divided into two sections for comparison between real-time and large models. The compared models are further grouped based on methods, and are in ascending mIoU order. The symbol "-" is used to denote the information is not available publicly. In general, results indicate that our models perform favorably against a wide range of models that were designed manually or generated by algorithms. In particular, with the same decoder, MoSegNet$_{small}$ improves BiSeNet [14] in all aspects, i.e., 1.1 points more accurate in mIoU, 10% faster per second in inference, and 2× more efficient in FLOPs. While MoSegNet$_{large}$ is on average 1.3 points more accurate and 18% faster than models obtained by RL-based methods, i.e., FNA [62] and AuxCell [20]. When compared to non real-time models, MoSegNet$_{xlarge}$ is 36× faster than PSPNet [63] while being more accurate in mIoU; it is also 3× more efficient in FLOPs than Auto-DeepLab-S [18] while being comparable in mIoU; and it achieves state-of-the-art performance in Cityscapes mIoU with multi-scale inference, surpassing HRNetV2 [43]. Moreover, Figs. 5 and 6 provide visual comparisons on speed-accuracy trade-off curve and sample outputs, respectively.

E. Transferability to other Datasets

In line with the practice adopted in most previous NAS methods [18], [19], [20], [62], we evaluate the transferability of the obtained architectures by inheriting the topology optimized for one dataset with weights retrained for a new dataset. In this work, we apply the evolved MoSegNets on Cityscapes dataset [16] to two other datasets—i.e., COCO-Stuff-10K [27] and PASCAL VOC 2012 [13]. And the retraining procedure

[5]https://developer.nvidia.com/tensorrt

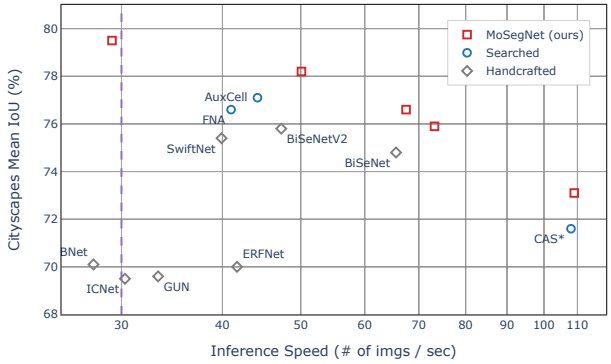

Fig. 5: *Speed-Accuracy* trade-off comparison with other real-time models on Cityscapes. Models with an inference speed of more than 30 images per second (purple line) are considered to be real-time. * uses a high-performance inference framework *TensorRT* to measure speed. See Table II for more comparisons to large models (non-real-time).

for these two datasets is very similar to the one used for Cityscapes. The cropped image size is modified to 640×640 for COCO-Stuff and 512×512 for PASCAL VOC. The number of training iterations is halved for COCO-stuff and increased by 20K for PASCAL VOC.

TABLE III: Performance comparison on COCO-Stuff-10K [27]. Models are evaluated on network complexity (FLOPs), inference speed, and accuracy (mIoU). The best result in each section is in bold.

Model	FLOPs ↓ (G)	Speed ↑ (imgs / sec)	mIoU ↑ (%)
FCN [64]	-	5.9	22.7
DeepLab	-	8.1	26.9
BiSeNet [14]	25.1	-	28.1
BiSeNetV2 [61]	27.1	42.5	28.7
ICNet [65]	-	35.7	29.1
PSPNet50 [63]	-	6.6	32.6
MoSegNet$_{small}$	**12.0**	**92.1**	30.5
MoSegNet$_{large}$	14.3	78.3	**34.2**

TABLE IV: Performance comparison on PASCAL VOC 2012 [13]. Models are evaluated on network complexity (FLOPs) and accuracy (mIoU). "COCO" indicates additional pretraining on MS COCO dataset [57]. The best result in each section is in bold.

Model	Method	COCO	FLOPs ↓ (G)	mIoU ↑ (%)
BiSeNet [14]	manual		16	71.0
DeepLabv3+ [44]			101	79.0
Res2Net [66]			101	80.2
PSPNet [63]			-	82.4
RefineNet [67]	gradient	✓	261	**82.9**
Auto-DeepLab-S [18]			43	71.7
Auto-DeepLab-L [18]		✓	87	80.8
MoSegNet$_{small}$	EA		**7.8**	75.8
MoSegNet$_{large}$		✓	9.3	82.2

Table III and IV compare the performance in terms of model complexity (inference speed or FLOPs) and segmentation accuracy (mIoU) on COCO-Stuff and PASCAL VOC, respectively. In general, our models are consistently more accurate than peer competitors while being an order of magnitude more efficient in FLOPs or inference speed, further validating the effectiveness of MoSegNAS under transfer learning setup. More specifically, MoSegNet$_{small}$ is 1.8 - 4.8 points more accurate in mIoU than BiSeNet [14] on COCO-stuff and

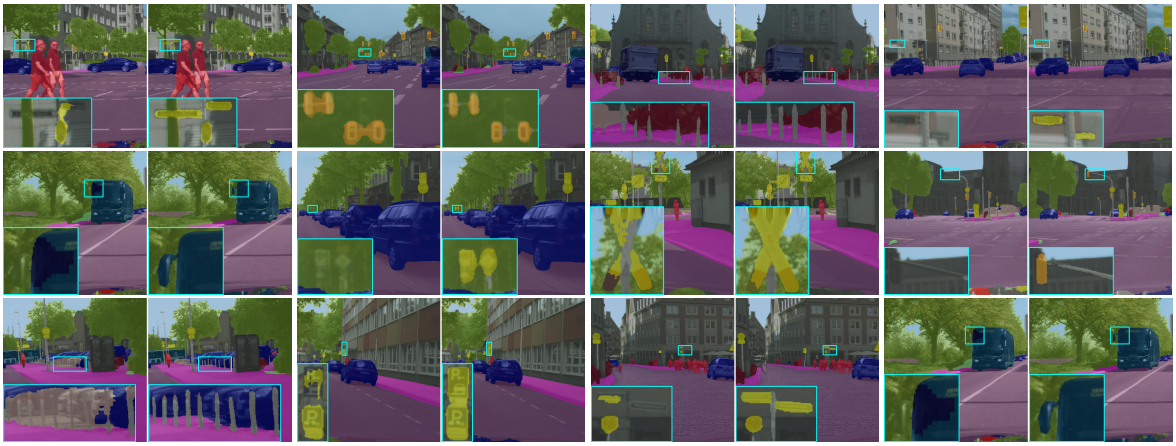

Fig. 6: Qualitative comparison on Cityscapes [16]. Example result pairs visualizing the segmentation performance of BiSeNet [14] (*left* image) vs. our proposed MoSegNet (*right* image). Note how MoSegNet predicts masks with substantially finer detail on small objects.

Fig. 7: Visual Comparison on COCO-Stuff [27]. Example pairs visualizing the input image (*left*), the ground truth (*middle*), and the predictions from the proposed MoSegNet (*right*).

PASCAL VOC, while being 2× more efficient in FLOPs and inference speed. Without pretraining on extra MS COCO data [57], MoSegNet$_{large}$ achieves 2.2 points higher mIoU on PASCAL VOC than peer NAS method Auto-DeepLab-L [18] while using an order of magnitude less FLOPs. Example outputs are provided in Fig. 7 for visualization.

V. Ablation Studies

This section aims to disentangle the individual contribution of each main component in the proposed method, followed by the hyperparameter analysis.

A. Analysis of Surrogate Modeling

To assess the effectiveness of the adopted surrogate modeling pipeline, we collect a number of well-established surrogate models from the literature. The considered surrogate models range from low-complexity methods (e.g., RBF, SVR), to more sophisticated ensemble-based methods (e.g., Gradient Boosting [56]), and dedicated DNN performance predictors proposed by previous NAS methods (e.g., E2EPP [37]). We uniformly sample from the search space to generate a pool of 1,100 architectures, where 1000 and 100 architectures are selected as the the training set and test set respectively. For architectures in the testing set, we train them with SGD optimizer on Cityscapes for $40K$ iterations and use the segmentation accuracy (mIoU) computed at the end of the training as the ground truth targets. For architectures in the training set, we use segmentation accuracy computed with the inherited weights from the hypernetwork as the training targets.

Recall that the selection of architectures in our proposed evolutionary routine compares relative performance differences among architectures. Hence, we adopt coefficients of correlation to quantitatively compare different surrogate models, as oppose to root mean square error or coefficient of determination. Specifically, we report all three indicators for measurement of correlation, including Pearson coefficient (r), Spearman's Rho (ρ), and Kendall's Tau (τ). The values of these three indicators range between [-1, 1] with a higher value indicating a better prediction performance. Table V presents the experimental results comparing our surrogate model with the selected peer competitors. Evidently, our method significantly outperforms other methods, achieving the best correlation coefficient in Pearson r, Spearman ρ and Kendall τ.

TABLE V: Coefficients of correlation comparison. Each method is trained with 1,000 samples and evaluated on a held-out test set of 100 samples. The results are averaged over 31 runs with standard deviation shown in the parentheses. "Time" denotes the training time measured in seconds. The best result in each section is in bold.

Method	Pearson r	Spearman ρ	Kendall τ	Time (sec.)
RBF	0.6199 (0.12)	0.5688 (0.10)	0.4692 (0.05)	0.02
Kriging	0.1161 (0.18)	0.0211 (0.15)	0.0107 (0.10)	4.52
SVR	0.5721 (0.11)	0.5597 (0.10)	0.4386 (0.05)	0.11
DT	0.6621 (0.15)	0.5986 (0.10)	0.5377 (0.05)	**0.01**
GB [56]	0.7409 (0.10)	0.7079 (0.08)	0.6259 (0.04)	0.61
E2EPP [37]	0.7746 (0.10)	0.7302 (0.07)	0.6162 (0.04)	1.01
RankNet (ours)	**0.7988 (0.08)**	**0.7658 (0.07)**	**0.6873 (0.03)**	3.82

In addition to being indicative in prediction, another desired property of a surrogate model is sample efficiency. To validate

the scalability of our method with respect to the training size, we repeat the previous experiment with gradually reduced # of training samples. The experimental results are presented in Fig. 8. In general, we observe that all methods exhibit a significant degradation in performance, i.e., lower mean and higher variance in Kendall τ as the # of available training samples reduces. However, our method remains consistently better than compared methods across all training size regimes, particularly in the high sample-efficient regime.

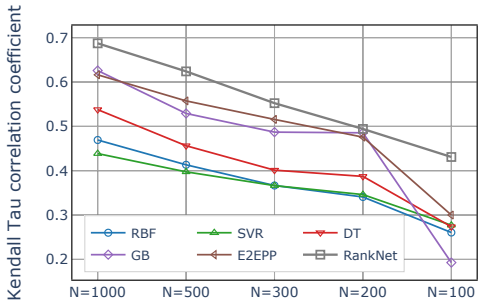

Fig. 8: Kendall rank-order correlation comparison under different # of training samples N. For each method, we repeat for 31 runs and report the mean performance.

Towards facilitating a better understanding on the observed effectiveness and efficiency, we visualize the development progress of our proposed surrogate model in Table VI, showing the impacts of the main components introduced in RankNet. We quantitatively measure the impacts by the Kendall τ correlation coefficient and the time complexity. A method with higher τ and lower time complexity is preferred.

TABLE VI: Development of RankNet. We disentangle the impact of adding individual components to a multi-layer perceptron (MLP; baseline), leading to our proposed surrogate model, i.e. RankNet. "GPU" indicates the use of GPU (CUDA) for training acceleration. All configurations are with 100 training samples.

Method	One-hot encoding	Ranking loss	Synthetic data	GPU	Kendall $\tau \uparrow$	Time (sec.) $\downarrow$
baseline					0.0036	2.4
	✓				0.3157	2.9
		✓			0.2170	3.8
			✓		0.0770	5.4
	✓	✓			0.4180	6.7
	✓	✓	✓		0.4309	6.9
RankNet	✓	✓	✓	✓	0.4309	3.7

B. Analysis of Pre-screening

Towards quantifying the effectiveness of the proposed hierarchical pre-screening criterion, the following ablative experiments are performed. Specifically, we consider two other conventional pre-screening methods along with a variant of the proposed method, as follows:

1) *Hierarchical*: the proposed hierarchical pre-screening procedure described in Algorithm 4.
2) *Survival*: the candidate solutions are selected based on their ranks and crowding distances, computed by applying NSGA-II environmental survival operator to the merged population of parents and all candidate solutions.
3) *HV contribution*: the candidate solutions are selected based on their individual contributions to the HV.
4) *Latency*: the candidate solutions are selected based on the uniformity along the second objective of latency.

On Cityscapes dataset [16], we run each setting to maximize both segmentation accuracy and inference speed (reciprocal of latency) for five times and report the median performance (measured in HV) as a function of the number of generations in Fig. 9. Evidently, the consistently higher HV suggests that the proposed hierarchical pre-screening criterion is more effective in selecting candidate solutions for high-fidelity evaluation than all other compared methods.

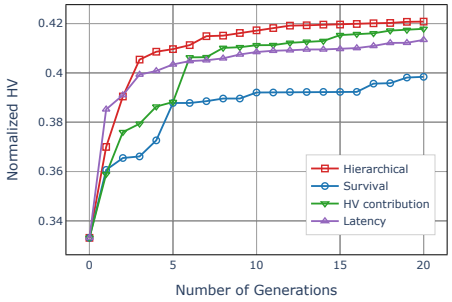

Fig. 9: Ablation study on the proposed hierarchical pre-screening criterion. The median generation-wise HV over five runs are reported.

C. Search vs. Transfer from ImageNet

Directly using an efficient encoder architecture designed for ImageNet-1K classification (e.g., ResNet [6]) has been the defacto method for dense image prediction tasks, including semantic segmentation. However, we argue that this method is conceptually sub-optimal given that classification mainly requires architectures to extract higher-level contextual information, while segmentation requires rich spatial details to be simultaneously maintained. In line with this derivation, we aim to evaluate the effectiveness of searching directly for segmentation (our method) by comparing the obtained models with efficient models designed for classification by other NAS methods [40], [68], [41].

We collect three state-of-the-art mobile models of relatively the same complexity (in terms of inference speed) as MoSegNet$_{small}$, and evaluate all four models for both classification on ImageNet and segmentation performance under the same training hyperparameter settings. Specifically, the training on ImageNet follows [40]: RMSProp optimizer with decay 0.9 and momentum 0.9; weight decay 1×10^{-5}; batch size 512. The learning rate decays from 0.012 to zero following the cosine annealing schedule. The data augmentation includes horizontal flip, crop and dropout ratio 0.2.

The experimental results presented in Table VI indicate that the classification and segmentation performance can be in conflict situation, suggesting the necessity of searching directly for the targeted task instead of transferring from ImageNet. In particular, despite the deficiency in ImageNet top-1 accuracy, MoSegNet$_{small}$ significantly outperforms other three state-of-the-art ImageNet models on all three semantic segmentation datasets.

TABLE VII: Semantic segmentation performance comparison on backbone (encoder) models optimized for classification (*Cls.*) and segmentation (*Seg.*). The compared backbone models are of similar complexity in terms of inference speed. The averaged relative differences are shown in parentheses.

Backbone	Optimized for	ImageNet Top-1	Semantic Segmentation mIoU (%)		
			Cityscapes	VOC 2012	COCO-Stuff
ProxylessNAS [40]	Cls.	74.6	73.4	74.2	28.2
MobileNetV3 [68]	Cls.	75.2	75.2	73.8	28.5
FBNetV2 [41]	Cls.	75.5	72.6	73.6	28.5
MoSegNet$_{small}$	Seg.	(-1.7) 73.4	(+2.9) 75.9	(+1.9) 75.8	(+2.1) 30.5

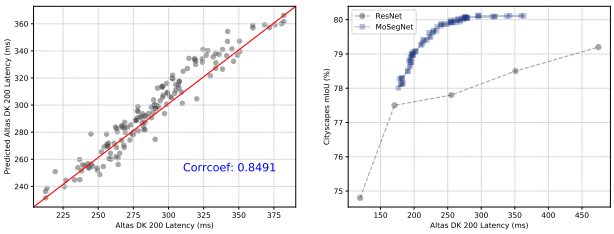

Fig. 10: (*Left*) Performance of the constructed latency look-up table of the Huawei Atlas 200 DK. The Kendall correlation coefficient τ is provided along with the fitted linear model (red line). **(*Right*) Latency-Accuracy trade-off comparison** between the obtained models and ResNets. Models with lower latency and higher accuracy are preferred (top-left corner).

VI. Application to Huawei Atlas 200 DK

A portfolio of prior works have demonstrated the necessity of incorporating hardware feedback in the loop of NAS for designing hardware-dependent architectures. However, most existing NAS works that target hardware performance were carried out on a simpler task of image classification, and mainly for mobile devices [40], [38]. In this work, we target a more challenging and demanding task of semantic segmentation. Using the Huawei Atlas 200 Developer Kit (DK) as an example, we demonstrate the practical utility of the proposed algorithm, MoSegNAS, for designing hardware-dependent models. Powered by the Ascend 310 processor, the Huawei Atlas 200 DK delivers 11 TFLOPs@FP16 with merely eight watts of power consumption, making it capable of handling challenging tasks in resource-constrained deployment environments.

The experimental setup is identical to the previous experiment described in Section IV-C, where we use the segmentation accuracy on Cityscapes dataset and the inference speed on the Huawei Atlas 200 DK as the twin objectives. We uniformly sample 2K architectures from our search space to construct a look-up table for predicting latency on the Atlas board. The performance of the fitted look-up table is visualized in Fig. 10 (*Left*). The returned models at the end of the evolution are re-trained thoroughly from-scratch. Fig. 10 (*Right*) depicts the experimental results, where we observe that the obtained MoSegNets achieve a substantially better speed-accuracy trade-off compared to the ResNets [6]. More details and visualizations of the obtained architectures are available in the supplementary materials.

To quantify the algorithmic contribution of MoSegNAS, the following ablative experiment has been performed. We impose an artificial constraint of mIoU being greater than 79% on Cityscapes dataset, and then search for architectures that are efficient for FLOPs, GPU and Atlas, respectively. As can be observed from the results in Table VIII, in general, different computational metrics or hardware require specialized models in order to be efficient. It suggests the competing nature of different efficiency metrics and computational environments, leading to the need of considering them in the optimization process, i.e., multi-objective NAS. Furthermore, the experimental results also validate that our proposed algorithm can be a viable mean to design task-specific neural architectures for the challenging task of semantic segmentation.

TABLE VIII: Computational efficiency comparison among MoSegNets optimized for FLOPs and latency on RTX 2080Ti and Huawei Atlas 200 DK.

Optimized for	Cityscapes mIoU (%)	Params (M)	FLOPs (G)	Latency (ms)	
				GPU	Atlas
FLOPs	79.02	35.3	**103.1**	28.7	267.4
GPU (2080Ti)	79.04	30.6	106.2	**26.3**	261.4
Atlas 200 DK	79.01	30.1	106.4	29.2	**196.7**

VII. Conclusion

This paper considered the problem of automating the designs of efficient neural network architectures for semantic segmentation tasks. For this purpose, we introduced *MoSegNAS*, a method harnessing the concept of surrogate modelling within an evolutionary multi-objective framework. With a sequence of modifications made to a multi-layer perceptron, we demonstrated that an indicative predictor can be efficiently learned online with few hundreds of samples. In the meantime, another surrogate model, in the form of a look-up table, was learned offline and specific to the search space for predicting inference latency. Utilizing the fact that high-fidelity evaluation of latency is significantly cheaper than segmentation accuracy, a customized hierarchical pre-screening criterion was proposed for in-fill selection. Experimental evaluation on three semantic segmentation datasets showed that models obtained by MoSegNAS outperform a wide range of state-of-the-art models in terms of both accuracy and inference speed. Finally, we demonstrated the practical utility of MoSegNAS in designing hardware-specific models on an application to the Huawei Atlas 200 DK. To the best of our knowledge, MoSegNAS presents the first realization of evolutionary multi-objective optimization for automatically designing efficient and custom neural network models for the challenging task of real-time semantic segmentation.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

[2] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[3] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[4] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[5] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-scale Image Recognition," in *International Conference on Learning Representations (ICLR)*, 2015.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[7] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[9] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2018.

[10] E. Galván and P. Mooney, "Neuroevolution in deep neural networks: Current trends and future challenges," *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 6, pp. 476–493, 2021.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[12] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[13] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International journal of computer vision*, vol. 111, no. 1, pp. 98–136, 2015.

[14] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "BiSeNet: Bilateral segmentation network for real-time semantic segmentation," in *European Conference on Computer Vision (ECCV)*, 2018.

[15] M. Fan, S. Lai, J. Huang, X. Wei, Z. Chai, J. Luo, and X. Wei, "Rethinking bisenet for real-time semantic segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 9716–9725.

[16] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[17] A. Milioto, P. Lottes, and C. Stachniss, "Real-time semantic segmentation of crop and weed for precision agriculture robots leveraging background knowledge in cnns," in *IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018.

[18] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei, "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[19] Y. Zhang, Z. Qiu, J. Liu, T. Yao, D. Liu, and T. Mei, "Customizable architecture search for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[20] V. Nekrasov, H. Chen, C. Shen, and I. Reid, "Fast neural architecture search of compact semantic segmentation models via auxiliary cells," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[21] Y. Shu, W. Wang, and S. Cai, "Understanding architectures learnt by cell-based neural architecture search," in *International Conference on Learning Representations (ICLR)*, 2020.

[22] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *AAAI Conference on Artificial Intelligence*, 2019.

[23] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Completely automated cnn architecture design based on blocks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 4, pp. 1242–1254, 2020.

[24] Z. Lu, I. Whalen, Y. Dhebar, K. Deb, E. Goodman, W. Banzhaf, and V. N. Boddeti, "Multiobjective evolutionary design of deep convolutional neural networks for image classification," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 2, pp. 277–291, 2021.

[25] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Evolving deep convolutional neural networks for image classification," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 394–407, 2020.

[26] T. Zhang, C. Lei, Z. Zhang, X. B. Meng, and C. L. P. Chen, "As-nas: Adaptive scalable neural architecture search with reinforced evolutionary algorithm for deep learning," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2021.

[27] H. Caesar, J. Uijlings, and V. Ferrari, "Coco-stuff: Thing and stuff classes in context," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[28] L.-C. Chen, M. Collins, Y. Zhu, G. Papandreou, B. Zoph, F. Schroff, H. Adam, and J. Shlens, "Searching for efficient multi-scale architectures for dense image prediction," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[29] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "NAS-FPN: Learning scalable feature pyramid architecture for object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[30] Y. Chen, T. Yang, X. Zhang, G. Meng, X. Xiao, and J. Sun, "Detnas: Backbone search for object detection," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[31] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, "Understanding and simplifying one-shot architecture search," in *International Conference on Machine Learning (ICML)*, 2018.

[32] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *International Conference on Learning Representations (ICLR)*, 2019.

[33] Z. Lu, K. Deb, E. Goodman, W. Banzhaf, and V. N. Boddeti, "NSGANetV2: Evolutionary multi-objective surrogate-assisted neural architecture search," in *European Conference on Computer Vision (ECCV)*, 2020.

[34] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[35] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *International Conference on Learning Representations (ICLR)*, 2017.

[36] Y. Sun, X. Sun, Y. Fang, G. G. Yen, and Y. Liu, "A novel training protocol for performance predictors of evolutionary neural architecture search algorithms," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2021.

[37] Y. Sun, H. Wang, B. Xue, Y. Jin, G. G. Yen, and M. Zhang, "Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 350–364, 2020.

[38] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once for all: Train one network and specialize it for efficient deployment," in *International Conference on Learning Representations (ICLR)*, 2020.

[39] X. Dai, P. Zhang, B. Wu, H. Yin, F. Sun, Y. Wang, M. Dukhan, Y. Hu, Y. Wu, Y. Jia *et al.*, "Chamnet: Towards efficient network design through platform-aware model adaptation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[40] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," in *International Conference on Learning Representations (ICLR)*, 2019.

[41] A. Wan, X. Dai, P. Zhang, Z. He, Y. Tian, S. Xie, B. Wu, M. Yu, T. Xu, K. Chen *et al.*, "FBNetv2: Differentiable neural architecture search for spatial and channel dimensions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[42] W. Chen, X. Gong, X. Liu, Q. Zhang, Y. Li, and Z. Wang, "Fasterseg: Searching for faster real-time semantic segmentation," in *International Conference on Learning Representations (ICLR)*, 2020.

[43] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao, "Deep high-resolution representation learning for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020.

[44] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *European Conference on Computer Vision (ECCV)*, 2018.

[45] Y. Jin and B. Sendhoff, "Pareto-based multiobjective machine learning: An overview and case studies," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 3, pp. 397–415, 2008.

[46] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, "NSGA-Net: Neural architecture search using multi-objective genetic algorithm," in *Genetic and Evolutionary Computation Conference (GECCO)*, 2019.

[47] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution," in *International Conference on Learning Representations (ICLR)*, 2019.

[48] H. Zhang, Y. Jin, R. Cheng, and K. Hao, "Efficient evolutionary search of attention convolutional networks via sampled training and node inheritance," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 2, pp. 371–385, 2021.

[49] Z. Lu, G. Sreekumar, E. Goodman, W. Banzhaf, K. Deb, and V. N. Boddeti, "Neural architecture transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 9, pp. 2971–2989, 2021.
[50] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.
[51] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.
[52] A. Sinha, P. Malo, and K. Deb, "A review on bilevel optimization: From classical to evolutionary approaches and applications," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 276–295, 2017.
[53] K. Deb, Z. Lu, I. Kropp, J. S. Hernandez-Suarez, R. Hussein, S. Miller, and A. P. Nejadhashemi, "Minimizing expected deviation in upper-level outcomes due to lower-level decision-making in hierarchical multi-objective problems," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2022.
[54] A. Brock, T. Lim, J. Ritchie, and N. Weston, "SMASH: One-shot model architecture search through hypernetworks," in *International Conference on Learning Representations (ICLR)*, 2018.
[55] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *Advances in Neural Information Processing Systems (NeurIPS) Deep Learning Workshop*, 2014.
[56] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
[57] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision (ECCV)*, 2014.
[58] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," in *International Conference on Computer Vision (ICCV)*, 2011.
[59] S. Mehta, M. Rastegari, L. Shapiro, and H. Hajishirzi, "ESPNetv2: A light-weight, power efficient, and general purpose convolutional neural network," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
[60] M. Orsic, I. Kreso, P. Bevandic, and S. Segvic, "In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
[61] C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang, "Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation," *International Journal of Computer Vision*, vol. 129, no. 11, pp. 3051–3068, 2021.
[62] J. Fang*, Y. Sun*, K. Peng*, Q. Zhang, Y. Li, W. Liu, and X. Wang, "Fast neural network adaptation via parameter remapping and architecture search," in *International Conference on Learning Representations (ICLR)*, 2020.
[63] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
[64] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
[65] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "ICNet for real-time semantic segmentation on high-resolution images," in *European Conference on Computer Vision (ECCV)*, 2018.
[66] S. Gao, M. M. Cheng, K. Zhao, X. Y. Zhang, M. H. Yang, and P. H. S. Torr, "Res2net: A new multi-scale backbone architecture," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2019.
[67] G. Lin, F. Liu, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks for dense prediction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 5, pp. 1228–1242, 2020.
[68] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," in *International Conference on Computer Vision (ICCV)*, 2019.

Zhichao Lu received his Ph.D. degree in Electrical and Computer Engineering from Michigan State University, USA, in 2020. He is currently an Assistant Professor in School of Software Engineering at Sun Yat-sen University, China. His current research focuses on the intersections of evolutionary computation, learning, and optimization, notably on developing efficient, reliable, and automated machine learning algorithms and systems. He received the GECCO-2019 best paper award.

Ran Cheng (M'16-SM'22) received the B.Sc. degree from the Northeastern University, Shenyang, China, in 2010, and the Ph.D. degree from the University of Surrey, Guildford, U.K., in 2016. He is currently an Associate Professor with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China. He is a recipient of the 2018 and 2021 IEEE Transactions on Evolutionary Computation Outstanding Paper Award, the 2019 IEEE Computational Intelligence Society (CIS) Outstanding Ph.D. Dissertation Award, and the 2020 IEEE Computational Intelligence Magazine Outstanding Paper Award. He is the founding Chair of the IEEE CIS Shenzhen Chapter. He is an Associate Editor of the IEEE TEVC, the IEEE TAI, and the IEEE TCDS.

Shihua Huang received the B.Eng. degree from Northeastern University, Shenyang, China, in 2018. He was a research assistant with the Department of Computer Science and Engineering, Southern University of Science and Technology (SUSTech), from 2018 to 2021. He is currently a Ph.D. student with the Department of Computer Science and Engineering at Michigan State University, East Lansing, USA. His current research interests include representation learning and evolutionary algorithm.

Haoming Zhang received the B.Eng. degree in computer science and technology from the Southern University of Science and Technology, China, in 2021. He is currently pursuing M.Eng. degree with Department of Computer Science and Engineering, Southern University of Science and Technology, China. His current research interests include computer vision, neural architecture search and evolutionary algorithms.

Changxiao Qiu received the B.S. degree in 2012, and the M.S. degree in 2015 from the University of Electronic Science and Technology of China(UESTC), Chengdu, China. He is currently a Senior Engineer with the Hisilicon Research Department, Huawei Technologies Co., Ltd., Shenzhen, China. His research interests include deep learning and computer vision.

Fan Yang received the electronic bachelor and master degree from the Paris-Sud University (University of Paris XI), 91400 Orsay, France, and the Ph.D. degree in informatics from the Paris-Saclay University, 91400 Orsay, France, in 2015. He is currently a Principal Engineer and Project Manager with the Hisilicon Research Department, Huawei Technologies Co., Ltd., Shenzhen, China. His current research interests include neural network compression and acceleration.

Surrogate-assisted Multiobjective Neural Architecture Search for Real-time Semantic Segmentation

(Supplementary Materials)

Zhichao Lu, Ran Cheng, Shihua Huang, Haoming Zhang, Changxiao Qiu, and Fan Yang

In this supplementary materials, we include: more details on the search space in Section 1, more details on the proposed method in Section 2; visualizations of the obtained architectures in Figure 3; and a further analysis on the Huawei Atlas 200 DK in Section 3.

1 Search Space and Encoding Continued

Designing a search space that is expressive but also tractable has been a key element to the success of NAS algorithms [1], [2]. The generality of the chosen search space has a major influence on the kinds of architectures that are even possible. While the tractability ensures good models can be identified by a principled algorithm within a reasonable time frame. In this work, we follow the encoder-decoder architectural framework, which has been proven successful for the task of semantic segmentation [3].

The encoder aims to extract higher-level contextual information at multiple scales via gradually downsampled feature maps, which is very similar to the backbone[1] part of a CNN for object classification. The decoder aims to aggregate the extracted multi-scale features (from the encoder) to derive lower-level spatial information for locating objects. In this work, we leverage the BiSeNet [4] heads as the decoder, and focus on designing the encoder. A pictorial illustration is provided in Fig. 1 in the main paper.

TABLE 1: Architecture Schema: The searched encoder architecture is based on ResNet [5], including the input image scale (r), depth (d_i), and the width (m_i, $\vec{e}_i$). The detailed setting is presented using $\big[k \times k$, #Ch$\big]$, where k denotes the convolution filter size, #Ch denotes the number of output channels, and $\big[\cdot\big]$ indicates residual connection. †indicates the locations for outputting features. The ResNet-50 architecture is also provided for reference.

	output scale	ResNet-50 [5]	MoSegNAS
Stem	$\frac{r}{2} \times \frac{r}{2}$	7×7, 64, stride 2	3×3, 32, stride 2 $\big[3 \times 3, 32\big] \times d_0$ $3 \times 3, 64 \times m_0$
		3×3 max pool, stride 2	
Stage 1	$\frac{r}{4} \times \frac{r}{4}$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \times \vec{e}_1 \\ 3 \times 3, 64 \times \vec{e}_1 \\ 1 \times 1, 256 \times m_1 \end{bmatrix} \times d_1$
Stage 2	$\frac{r}{8} \times \frac{r}{8}$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 128 \times \vec{e}_2 \\ 3 \times 3, 128 \times \vec{e}_2 \\ 1 \times 1, 512 \times m_2 \end{bmatrix} \times d_2$
Stage 3†	$\frac{r}{16} \times \frac{r}{16}$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 256 \times \vec{e}_3 \\ 3 \times 3, 256 \times \vec{e}_3 \\ 1 \times 1, 1024 \times m_3 \end{bmatrix} \times d_3$
Stage 4†	$\frac{r}{32} \times \frac{r}{32}$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \times \vec{e}_4 \\ 3 \times 3, 512 \times \vec{e}_4 \\ 1 \times 1, 2048 \times m_4 \end{bmatrix} \times d_4$
Tail†	1×1	global average pool	

For each stage, we allow the # of layers to vary from two to four (four to six for stage 3), and apply stride of two to the first layer to half the feature map size (except the first stage where a max pooling with stride 2 is used). Additionally, we allow the output # of channels (m_i) to vary from $0.6\times$ to $1\times$ of the default values. For each layer, we allow the

1. all layers prior to the last linear classification layer.

output # of channels of the first 1×1 convolution ($\vec{e}_i$) to vary from $0.8\times$ to $1.4\times$ of the default values. The choices of the feature-outputting locations are implicitly controlled by the depth settings as we always output the features of the last layers from the final two stages and tail. Table 1 summaries our architectural decision space. And example of our encoding is provided in Fig. 1.

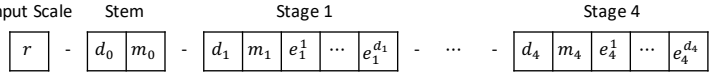

Fig. 1: Encoding: An integer-valued string is used to encode searchable architectures comprising a stem and four stages. The leading two digits encode the # of layers (d_i) and the output # of channels (m_i). For stages, the remaining digits encode the intermediate # of channels ($\vec{e}_i$) for each layer in the stage. Zeros are padded to the stage strings to avoid variable-length representations. See Table 1 for more details.

2 PROPOSED METHOD CONTINUED

– Initialization: Recall that our method is an evolutionary process in which initial solutions are made gradually better as a group. Instead of starting from a completely random set of solutions (i.e., architectures), we follow a customized initialization procedure shown in Algorithm 1. It ensures that the extreme solutions (i.e., most and least complex architectures) are always parts of the initial population, in the spirit of enhancing diversity. Empirically, we observe that this initialization method leads to better surrogate modeling performance, which in turn improves the performance of the subsequent evolutionary search for optimal architectures.

Algorithm 1: Initialization

Input : Population size N, hypernetwork $\mathcal{H}$, validation data $\mathcal{D}_{vld}$, archive $\mathcal{A}$.

$i \leftarrow 0$ // initialize an individual counter.
$P, F_p \leftarrow \emptyset$ // initialize parent population.
while $i < N$ **do**
  if $i == 0$ **then**
    // Set individual to the lower bound of the search space.
    $p \leftarrow lower\ bound$
  else if $i == 1$ **then**
    // Set individual to the upper bound of the search space.
    $p \leftarrow upper\ bound$
  else
    $p \leftarrow$ randomly sample an individual from the search space.
  end
  $F \leftarrow Evaluate(p, \mathcal{H}, \mathcal{D}_{vld})$ ◁ Algo 2 (main paper)
  $P, F_p \leftarrow P \cup p, F_p \cup F$
end
$\mathcal{A} \leftarrow \mathcal{A} \cup (P, F_p)$
Return $P, F_p, \mathcal{A}$

– Segmentation accuracy prediction: most selection operators in existing EAs (such as binary tournament selection, non-dominated sorting, etc.) merely require relative relationships among individuals instead of specific fitness values. For example, given two candidate solutions A and B, it is sufficient for selection operators to proceed by knowing either $f_A > f_B$ or $f_A < f_B$, while the absolute difference between the fitness of them (i.e., $\|f_A - f_B\|$) is not required. Hence, we argue that "a low MSE is desirable but not necessary".

Considering such a context, our motivation is to design a computationally efficient surrogate model to substitute the time-costly evaluation via the supernet, where the fundamental requirement is that the rank correlation between the predicted values and the actual values should be sufficiently high. Therefore, we replace the standard loss of MSE (Eq. (4)) with a ranking-based loss (Eq. (5)) for learning the surrogate model. A pictorial illustration is provided in Figure 2.

– Pre-screening: Recall that the pre-screening criterion adopted in our method for in-fill selection inherits an inductive bias towards the objective that is (relatively speaking) computationally friendly to (high-fidelity) evaluate, i.e., inference latency. To realize this goal, we formulate the problem as a subset selection problem and solve it using a customized binary

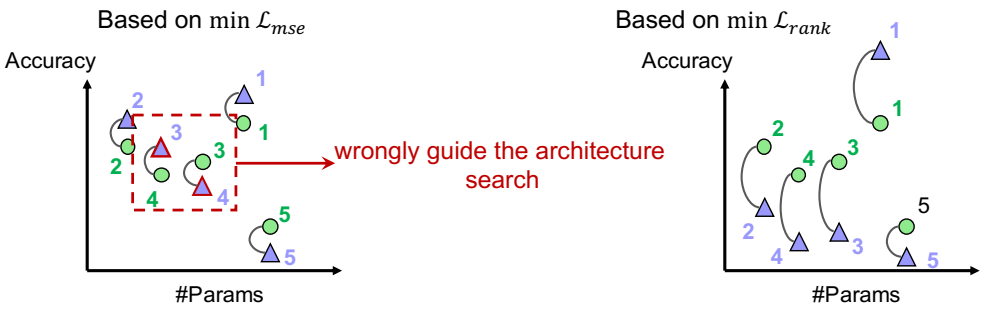

Fig. 2: Considering five pairs of the true (**green** circles) and predicted (**purple** triangles) values. An MSE loss aims to minimize the Euclidean distance between the true and predicted values (i.e., left sub-figure). While a ranking-based loss aims to maintain the same ranking between sorting based on the true and predicted values (right sub-figure). For the purpose of differentiating architectures during an EA process, a high-rank correlation is required, while a low MSE is desirable but not required.

genetic algorithm (GA). The modifications introduced to a conventional GA consider the feasibility in solution creation steps (i.e., initialization, crossover, and mutation) to improve the efficiency of the algorithm. The detailed procedures are outlined in Algorithm 2.

Algorithm 2: Customized Binary GA

Input : Population size N, max. # of generations T,
total # of candidate solutions L,
max. # of solutions to be selected K.

// initialization
$i \leftarrow 0$, $P \leftarrow \emptyset$
while $||P|| < N$ **do**
  $p \leftarrow zeros(L)$
  $k \leftarrow$ create a random integer less than K.
  $idx \leftarrow$ create a set of permutation indices of length L.
  $p[idx[:k]] \leftarrow 1$
  $P \leftarrow P \cup p$
end
while $t < T$ **do**
  $Q \leftarrow \emptyset$
  while $||Q|| < N$ **do**
    $p_1, p_2 \leftarrow$ binary tournament selection (P)
    // crossover
    $q \leftarrow zeros(L)$
    $idx_1 \leftarrow$ AND(p_1, p_2) // find indices where both $p1$ & $p2$ take value "1".
    $q[idx_1] \leftarrow 1$
    $idx_2 \leftarrow$ XOR(p_1, p_2) // find indices where at least one parent takes value "1".
    $k \leftarrow$ create a random integer less than $K - ||idx_1||$.
    $q[idx_2[:k]] \leftarrow 1$
    // mutation
    $q \leftarrow$ randomly flip a "0" to "1"
    $q \leftarrow$ randomly flip a "1" to "0"
  end
  $P \leftarrow$ Survive$(P \cup Q, N)$ // survive the top-K ranked solutions.
end
Return P

3 Application to Huawei Atlas 200 DK Continued

Based on the experimental results shown in Section VI of the main paper, we observe a weak correlation between inference latency on the Huawei Atlas 200 DK and on GPUs, such as RTX 2080 Ti. Towards facilitating a better understanding on

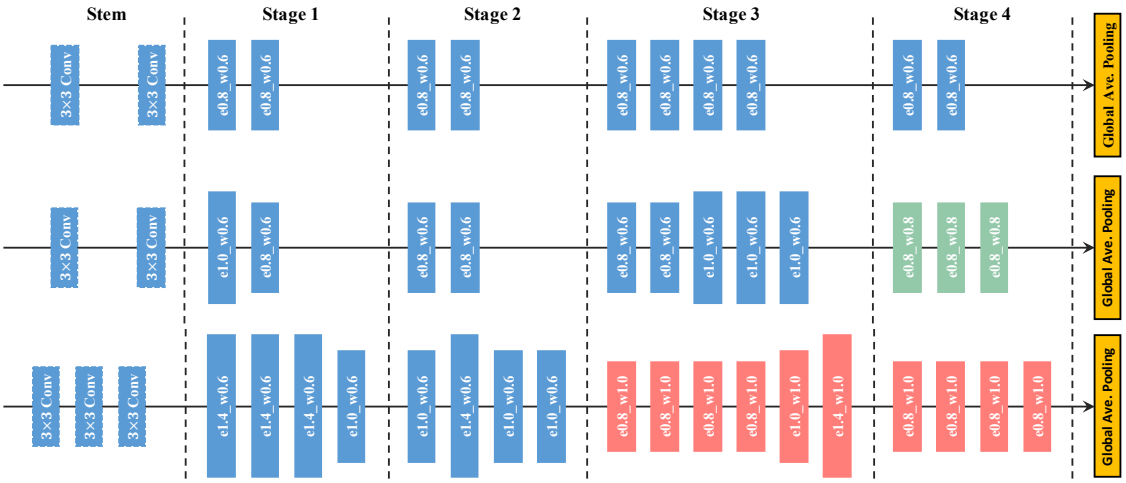

Fig. 3: MoSegNets-{small, median, large} architectures from Table II in the main paper. Architectures are arranged in descending inference-speed order from top to bottom.

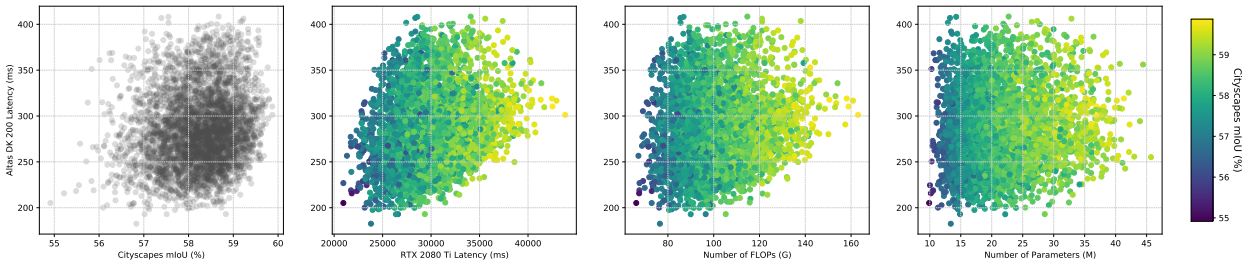

Fig. 4: Visualization of the trade-off between the inference latency on the Huawei Atlas 200 DK and {Cityscapes mIoU, RTX 2080 Ti latency, #FLOPs, #Params}, from left to right respectively.

the characteristics of the run-time latency on Huawei Atlas 200 DK, the following experiments have been performed. We sample a large number of 4K architectures uniformly from our search space. For each model, we compute the semantic segmentation accuracy (mIoU) on Cityscapes dataset using the weights inherited from the hypernetwork (Section III-C in the main paper), with an additional 2K iterations of fine-tuning. In the meantime, we also measure #Params, #FLOPs and the run-time latency on both Atlas and 2080Ti for each model. Fig. 4 depicts the experimental results. In general, we observe that the latency on Atlas 200 DK is relatively more correlated with latency on 2080 Ti, as oppose to #FLOPs, #Params. And it's surprising that there exists high-performing architectures across all levels of latency (on Atlas 200 DK).

REFERENCES

[1] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
[2] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
[3] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *European Conference on Computer Vision (ECCV)*, 2018.
[4] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "BiSeNet: Bilateral segmentation network for real-time semantic segmentation," in *European Conference on Computer Vision (ECCV)*, 2018.
[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.