Adaptive Linear Span Network for Object Skeleton Detection

Chang Liu, Yunjie Tian, Zhiwen Chen, Jianbin Jiao, *Member, IEEE*,
and Qixiang Ye, *Senior Member, IEEE*

***Abstract*— Conventional networks for object skeleton detection are usually hand-crafted. Despite the effectiveness, hand-crafted network architectures lack the theoretical basis and require intensive prior knowledge to implement representation complementarity for objects/parts in different granularity. In this paper, we propose an adaptive linear span network (AdaLSN), driven by neural architecture search (NAS), to automatically configure and integrate scale-aware features for object skeleton detection. AdaLSN is formulated with the theory of linear span, which provides one of the earliest explanations for multi-scale deep feature fusion. AdaLSN is materialized by defining a mixed unit-pyramid search space, which goes beyond many existing search spaces using unit-level or pyramid-level features. Within the mixed space, we apply genetic architecture search to jointly optimize unit-level operations and pyramid-level connections for adaptive feature space expansion. AdaLSN substantiates its versatility by achieving significantly higher accuracy and latency trade-off compared with the state-of-the-arts. It also demonstrates general applicability to image-to-mask tasks such as edge detection and road extraction. Code is available at https://github.com/sunsmarterjie/SDL-Skeletongithub.com/sunsmarterjie/SDL-Skeleton.**

***Index Terms*— Skeleton detection, linear span network, neural architecture search, genetic algorithm.**

I. Introduction

SKELETON is a kind of representative visual descriptor, which contains rich information about object topology, constituting an explicated abstraction of object shape. Object skeletons can be converted to descriptive features and/or spatial constraints, promoting computer vision tasks such as human pose estimation [1], hand gesture recognition [2], text detection [3] and object localization [4], in an explainable fashion.

In the deep learning era, object skeleton detection using convolutional neural networks (CNNs) has made unprecedented progress. State-of-the-art approaches commonly utilize side-output architectures to integrate feature pyramid as a countermeasure for variations from object appearances, poses, and scales. This is based on the observation that low-level features focus on detailed structures while high-level features are rich in semantics [5].

As a pioneer work, the holistically-nested edge detection (HED) [6] used a deeply supervised strategy to take full use of the hierarchical multi-scale features in a parallel manner. Fusing scale-associated deep side-outputs (FSDS) [7] adopted a divided-and-conquer approach to supervise network side-outputs given scale-associated ground-truth. SRN [8] and RSRN [5] investigated the multi-layer association problem by utilizing side-output residual units to pursue the complementarity among multi-scale features in a deep-to-shallow fashion. HiFi [9] introduced a bilateral feature integration mechanism to incorporate the low-level details and high-level semantics.

Despite the encouraging progress, one limitation lies in that hand-crafted network architectures for skeleton detection lack the theoretical basis to maximize representation complementarity for objects/parts in different granularity. This hinders further performance optimization of object skeleton detection in complex scenes.

In this paper, we formulate the pixel-wise binary classification tasks as linear reconstruction problems within the linear span framework [10] and conclude that the key for network design is to perform sufficient feature space expansion. Accordingly, we propose an approach, termed adaptive linear span network (AdaLSN), to transform and integrate hierarchical scale-aware features for object skeleton detection, Fig. 1. AdaLSN is driven by neural architecture search (NAS), which optimizes the network operations and connections to push the features towards scale-aware configuration. This improves the feature versatility towards higher accuracy and latency trade-off compared to the state-of-the-arts.

Under the guidance of the linear span theory [11], we ascertain that not only should we expand feature subspace in each stage, but also adaptively reduce structure redundancy to compress their intersection space related to multiple stages. Specifically, we design the linear span unit (LSU) to transform the input features towards complementary with customized operations. Based on LSUs, we construct an explainable search space, termed the linear span pyramid (LSP), to perform subspace and sum-space expansion. LSP is a unit-pyramid mixed search space, which pursues feature complement spaces with a complementary learning strategy. At the unit level, the feature space is expanded by the LSU. At the pyramid level, we progressively pose intermediate supervision to expand feature subspaces to reduce their semantic overlap and expand the sum-space. With a genetic search algorithm, AdaLSN auto-

Manuscript received October 18, 2020; revised April 14, 2021; accepted April 28, 2021. Date of publication May 17, 2021; date of current version May 21, 2021. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 61836012 and Grant 61771447 and in part by the Strategic Priority Research Program of Chinese Academy of Sciences under Grant XDA27000000. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Christophoros Nikou. *Chang Liu and Yunjie Tian contribute equally to this work. (Corresponding author: Qixiang Ye.)*

Chang Liu, Yunjie Tian, Jianbin Jiao, and Qixiang Ye are with the School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: liuchang615@mails.ucas.ac.cn; tianyunjie19@mails.ucas.ac.cn; jiaojb@ucas.ac.cn; qxye@ucas.ac.cn).

Zhiwen Chen is with Alibaba Group, Hangzhou 311121, China (e-mail: zhiwen.czw@alibaba-inc.com).

Digital Object Identifier 10.1109/TIP.2021.3078079

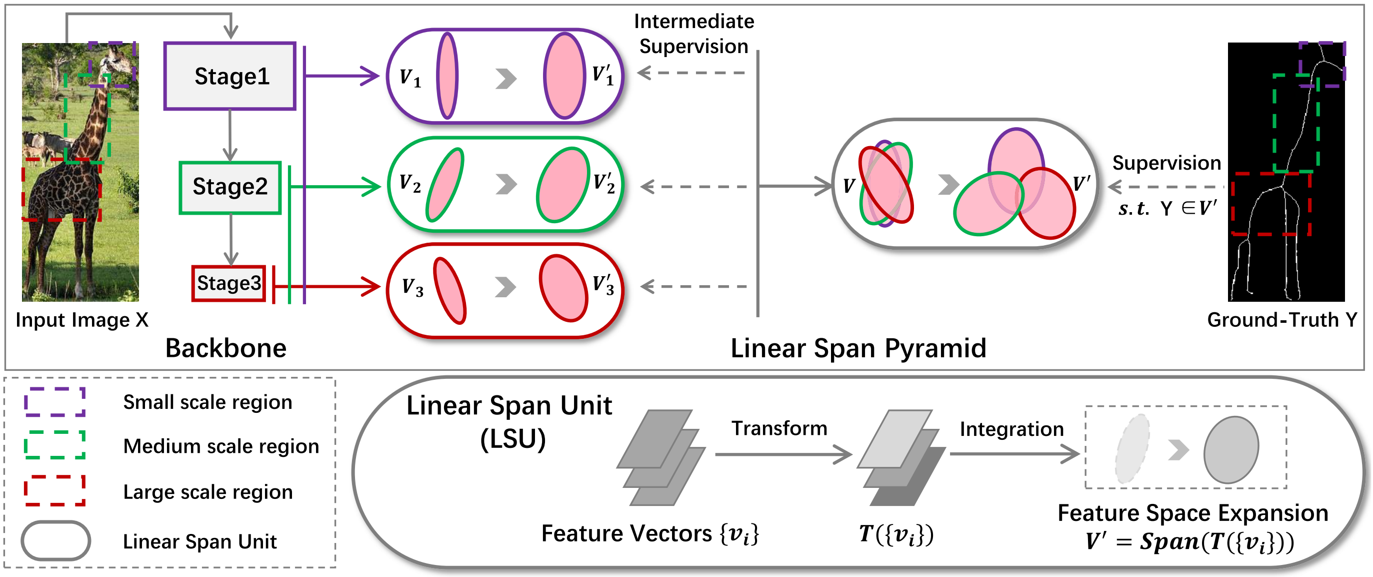


Fig. 1. The architecture search space designed for AdaLSN. AdaLSN is formulated with the theory of linear span, which aims to perform feature space expansion in both feature subspace (left three color LSUs) and sum-space (right gray LSU) levels. Driven by genetic search algorithm, AdaLSN adaptively configures network architectures and scale-aware features to represent objects in different scale granularity. (Best viewed in color).

matically optimizes the network operations and connections in each area of the search space, including multiple side-outputs, short connections, feature transform, and intermediate supervision, to push the features towards scale-aware configuration. In order to increase the complementary nature of the feature vectors we introduce this architecture optimization, which removes edges from the architecture, thereby encouraging a reduction in the intersection space between feature vectors.

LSN was proposed in our previous study [10], while is promoted by introducing a well-designed architecture search space and an adaptive search algorithm. The contributions of this paper are summarized as follows:

- We propose an adaptive linear span network (AdaLSN), opening up a promising direction to learn complementary scale-aware features within the framework of linear span.
- We design a unit-pyramid mixed search space to expand the feature subspace and the sum-space, where competitive network architectures evolve via genetic operations.
- AdaLSN significantly improves the state-of-the-arts of object skeleton detection and is extended to edge detection and road extraction, demonstrating the general applicability to image-to-mask tasks.

II. Related Work

Object skeleton detection has attracted much attention by the computer vision community for its significance: elaborating object skeleton facilitates image understanding. Existing works typically perform geometric modeling or multi-scale feature fusion to improve skeleton detection performance, however, to the best of our knowledge, NAS methods have not been considered in this area.

A. Hand-Crafted Method

Early skeleton detection methods were usually performed on binary images by geometric modeling, *e.g.*, morphological image operations. One approach is to treat object skeleton as line subsets which connect the center points of super-pixels. Such line subsets were explored from the super-pixels and extract skeleton paths using a sequence of disc models [12]. The smoothness of the skeleton can be enforced with spatial filters, *e.g.*, particle filters, which link local skeleton segments to continuous curves [13]. When applied on color images, an image segmentation procedure for contour extraction was first performed as pre-processing. The segmentation procedure tended to produce multi-scale super-pixels, which were converted to skeleton pixels using geometric models.

In the deep learning era, object skeleton detection has been formulated as pixel-wise binary classification problem with multi-scale feature integration. For clarity, the characters of several typical methods are summarized in Table I. The HED method [6] developed parallel multiple side-outputs to produce edges, which can be also used for skeleton detection. Fusing scale-associated deep side-outputs (FSDS) [7] learned skeleton representations with specific scales across multiple network stages given scale-associated ground-truth.

Side-output residual network (SRN) [8] leveraged the side-output residual units to build short connections between adjacent side-output branches for complementary feature utilization. It progressively expanded the feature space to fit the errors between the skeleton ground-truth and the multiple side-outputs. To take advantage of richer scale-aware features, RCF [14], RSRN [5], and HiFi [9] establish dense side-output branches for smooth multi-scale feature integration.

In the linear span view [10], the feature integration actually aims at spanning larger feature spaces for complementary feature extraction and fusion. To this end, DeepFlux [15] utilized the ASPP module [16] to enrich semantic levels and scale granularity of the feature space. DeepFlux also contributed a learnable strategy to predict a vector field, which corresponded to each image pixel with a skeleton/background pixel, in the fashion of flux-based skeletonization. This flux-based

TABLE I

CHARACTERISTICS OF STATE-OF-THE-ART APPROACHES FOR OBJECT SKELETON DETECTION

	Method	Year	Network Architecture				Multi-scale Annotation	Geometric Modeling
			Multiple Side-outputs	Short Connection	Feature Transform	Intermediate Supervision		
Manual Design	**HED** [6]	2015	✓	✗	✗	✓	✗	✗
	FSDS [7]	2016	✓	✓	✗	✓	✓	✗
	RCF [14]	2017	✓(dense)	✗	✗	✗	✗	✗
	SRN [8]	2017	✓	✓	✗	✓	✗	✗
	RSRN [5]	2017	✓(dense)	✓	✗	✓	✗	✗
	HiFi [9]	2018	✓(dense)	✓	✗	✗	✓	✗
	LSN [10]	2018	✓	✓(dense)	✗	✓	✗	✗
	Deepflux [15]	2019	✓	✗	✓	✗	✗	Skeleton context flux
	GeoSkelNet [17]	2019	✓	✗	✓	✗	✗	Hausdorff distance
Automatic Search	**AdaLSN (ours)**	2021	✓(adaptive)	✓(adaptive)	✓(adaptive)	✓(adaptive)	✗	✗

skeletonization explicitly encoded the skeleton pixel positions to context-aware entities. GeoSkelNet [17] also used the region-based vector field to model object parts at multiple scales. By designing Hausdorff distance-inspired objective function, both global and local contours were detected through an end-to-end network.

The conventional hand-designed skeleton networks, although incorporating rich prior knowledge, have obvious limitations. With elaborately designed feature integration mechanisms, they still experience difficulty when configuring representation for objects/parts in different scale granularity.

B. Neural Architecture Search

NAS targets at optimizing network architectures driven by training data without human intervene. Early NAS methods formulated network designs using reinforcement learning [18]–[21], evolutionary algorithms [22], [23], or random search approaches [24].

To reduce time cost, one-shot search methods employed weight-reusing [25] and weight-sharing [26], [27] strategies, which fuse network architecture search with network weight optimization. A special family of one-shot methods, which adjusted itself in a continuous space, formulated the search space as a super-network [28]. This leads to differentiate search methods where the network and architectural parameters are jointly optimized [29]–[31]. Recent EfficientNet [32] defined a scaling method which uniformly searched network depth, width, and resolution by introducing compound coefficients.

To automatically fuse multi-scale features, NAS-FPN [33] defined a search space where the hierarchical side-output features can be optimally combined based on reinforcement learning. NAS-Unet [34] adopted a similar idea to fuse hierarchical side-output features by defining primitive operation sets on the search space to automatically find cell architectures. For the image segmentation task, Auto-DeepLab [35] designed a search space covering popular hand-designed networks. Driven by a gradient-based searching algorithm, Auto-DeepLab found novel architectures which significantly improved the segmentation performance.

In this paper, we propose using NAS to define a general feature integration method. By introducing the linear span units upon side-output features, we provide the foundation to define operations and connections for feature space expansion. By using the linear span pyramid, we define a complete space for complementary feature extraction and architecture optimization in the linear span framework.

III. RETHINKING SKELETON NETWORK DESIGN

A. Problem Retrospect

With cascaded convolution and down-sampling operations, CNN backbones typically generate the feature pyramids where features in deeper layers have richer semantics and larger receptive fields, yet fewer fine-details and smaller spatial resolutions. In the pyramid, the feature semantic levels, scale granularity, and resolution variation tangle together, raising the following challenges to feature integration.

1) How to tackle the large variations of appearance, shape, pose, and scale of objects while depressing cluttered backgrounds with limited convolutional layers?
2) How to balance the benefit of complementary features and the degradation caused by up-sampling in the resolution alignment for feature integration?
3) How to explicitly enrich the semantic hierarchy and scale granularity of the convolutional features?
4) How to find out the hierarchical scale-aware features of the largest complementary?

To tackle these challenges, existing methods typically employed the techniques including multiple side-outputs, short connections, feature transform and intermediate supervisions, Table I, each of which can solve a challenge. However, the lack of systematic way to integrate these techniques hinders finding out optimal feature representation.

B. Linear Span View

In the deep learning era, object skeleton detection is typically performed with a pre-trained CNN backbone for feature extraction and 1 × 1 convolutional layers in the side-output branches for skeleton pixel detection.

Given an input image X, we denote the extracted features as $V_{C\times W\times H} = (v_{c,i,j})_{C\times W\times H}$ and the 1 × 1 convolutional layer as $\Lambda_{C\times 1} = (\lambda_c)_{C\times 1}$, where C, W, and H respectively denote the channel number, feature map width, and feature map height. Skeleton detection is formulated as a pixel-wise classification problem [6], as

$$\sum_{c=1}^{C} \lambda_c \cdot v_{c,i,j} = \hat{y}_{i,j} \approx y_{i,j}, \tag{1}$$

where $\hat{y}_{i,j}$ and $y_{i,j}$ respectively denote the pixel values of the output image $\hat{Y}$ and the ground-truth mask Y at location (i, j). Regarding each feature map as a vector v_c, where c indexes feature channels, Eq. 1 is rewritten from the perspective of linear reconstruction, as

$$\sum_{c=1}^{C} \lambda_c v_c = \hat{Y} \approx Y. \tag{2}$$

Based on Eq. 2, we introduce the linear span concepts and space decomposition theorems [11], which lead to the explainable search space and the implementation of AdaLSN.

Definition 1 (Linear Span Operator). *Given a feature vector set $V = \{v_1, v_2, \cdots, v_C\}$ over a field $\mathbb{R}$, the linear span operator is defined as*

$$\mathcal{V} = Span(V) = \{v | v = \sum_{c=1}^{C} \lambda_c v_c, v_c \in V,\ \ \lambda_c \in \mathbb{R}\},$$

where $\mathcal{V}$ constructs a linear space. V denotes the spanning set of $\mathcal{V}$. It is the basis of $\mathcal{V}$, if V is linearly independent.

With *Definition 1*, Eq. 2 is updated to

$$Span(\{v_1, v_2, \cdots, v_C\}) \ni \hat{Y} \approx Y. \tag{3}$$

Eq. 3 reveals that each side-output branch of the network can be approximated as a linear system, which is driven by the loss layer to fit the ground-truth.

In the procedure, the key is to optimize the spanning set and implement feature space expansion towards precise skeleton reconstruction. However, existing approaches, without customized modules to facilitate feature space expansion, are implicit and limited. This inspires utilizing proper convolutional operators (O) to explicitly transform the spanning set while expanding the spanned space, as

$$Span(O(\{v_1, v_2, \cdots, v_C\})) \ni \hat{Y} \approx Y. \tag{4}$$

With feature space expansion, we further propose to improve multi-stage feature integration based on the space decomposition and space dimension theorems.

Definition 2 (Sum-space). *Suppose $\mathcal{V}_1$ and $\mathcal{V}_2$ are subspaces in the linear space $\mathcal{V}$, the set $\mathcal{V}_{1,2}$, defined as*

$$\mathcal{V}_{1,2} = \mathcal{V}_1 + \mathcal{V}_2 = \{v_1 + v_2 | v_1 \in \mathcal{V}_1, v_2 \in \mathcal{V}_2\},$$

is also a linear space, termed the sum-space of $\mathcal{V}_1$ and $\mathcal{V}_2$.

Theorem 1 (Space Decomposition). *Any subspace $\mathcal{V}_1 \subset \mathcal{V}$ has a complement subspace $\mathcal{V}_1^C \in \mathcal{V}$ such that*

$$\mathcal{V} = \mathcal{V}_1 + \mathcal{V}_1^C,$$

and the union of the bases of $\mathcal{V}_1$ and $\mathcal{V}_2$ is the basis of $\mathcal{V}$.

Definition 2 reveals that short connections among side-output branches actually sum the corresponding feature subspaces. Theorem 1 substantiates that one can expand the sum-space of two subspaces by forcing one of them to fit the complement subspace of the other, which guides the complementary feature learning. In specific, we propose to progressively force the feature subspaces from the shallow stages to fit the complement subspace of the deep stages [8]. Numbering the S stages from shallow to deep, we denote the features extracted from the $s-th$ stage as V_s, $s = 1, 2, \cdots, S$, the spanned feature subspaces as $\mathcal{V}_s = Span(V_s)$, and the corresponding subspace after transform as $\mathcal{V}'_s$, such that $\mathcal{V}'_s = Span(O(V_s))$, Eq. 4. The complementary learning strategy is concluded as

$$\begin{cases} \mathcal{V}'_S \ni \hat{Y}_S \approx Y \\ \mathcal{V}'_S + \mathcal{V}'_{S-1} \ni \hat{Y}_{S-1} \approx Y \\ \cdots \\ (\mathcal{V}'_S + \mathcal{V}'_{S-1} + \cdots + \mathcal{V}'_2) + \mathcal{V}'_1 \ni \hat{Y}_1 \approx Y. \end{cases} \tag{5}$$

Theorem 2 (Space Dimension). *Supposing $\mathcal{V}$ is a finite dimensional linear space, $\mathcal{V}_1$ and $\mathcal{V}_2$ are two subspaces of $\mathcal{V}$ such that $\mathcal{V} = \mathcal{V}_1 + \mathcal{V}_2$, and $\mathcal{V}_{1,2}$ is the intersection of $\mathcal{V}_1$ and $\mathcal{V}_2$, i.e., $\mathcal{V}_{1\cap 2} = \mathcal{V}_1 \cap \mathcal{V}_2$. $\mathcal{V}_{1\cap 2}$ is a linear space, and*

$$\dim \mathcal{V} = \dim \mathcal{V}_1 + \dim \mathcal{V}_2 - \dim \mathcal{V}_{1\cap 2}.$$

Theorem 2 uncovers that smaller dimension of the intersection space implies larger dimension of the sum-space. This again supports performing feature space expansion at the subspace level while compressing their intersection space. Note that the intersection space of feature subspaces is actually the semantic overlap of multi-stage features, caused by architecture redundancy. We accordingly propose to utilize architecture encoding and NAS to solve these issues.

IV. Adaptive Linear Span Network

Based on the linear span theory [11], AdaLSN is materialized by defining a mixed unit-pyramid search space to perform both subspace and sum-space expansion, Fig. 2. To fulfill feature subspace expansion, we design the linear span unit (LSU) which transforms the input feature vectors towards complementary by adaptively selecting operators and connections. Based on the LSUs and the complementary learning strategy, we establish a pyramidal architecture search space, referred to as the linear span pyramid (LSP). On the pyramid, short connections are utilized to merge the spanning set of each feature subspace towards complementary and complete. To reduce the architecture redundancy, we encode the LSP to gene segments for genetic architecture search. The searched AdaLSNs comprehensively integrate the techniques about multiple side-outputs, short connections, feature transforms and intermediate supervisions.

A. Unit-Pyramid Space for Linear Span

Linear Span Unit (LSU). An LSU can be regarded as a directed acyclic graph, which consists of an input node, an output node, and P intermediate nodes, each of which represents a set of feature vectors, Fig. 2. An LSU encourages multiple paths for feature transform, while the last node in

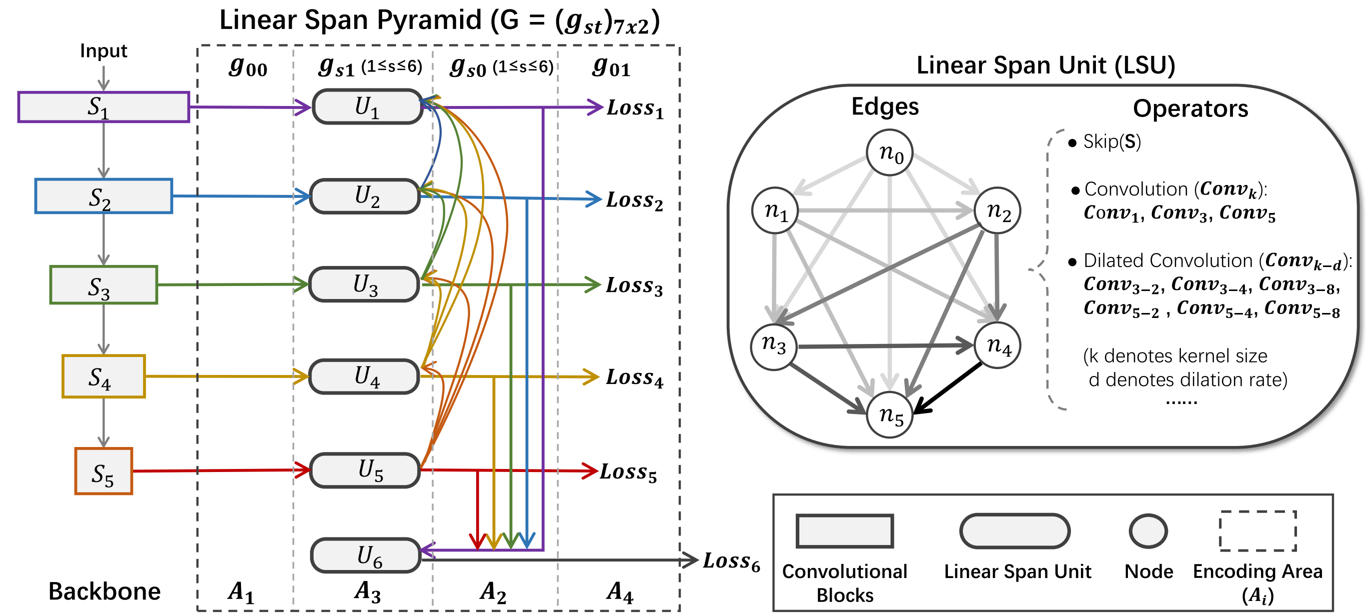


Fig. 2. Linear span pyramid (LSP) and linear span unit (LSU) constitute the unit-pyramid search space for architecture search and adaptive linear span.

the unit aggregates the transformed features together for space expansion. The nodes are denoted as $n_i, i = 0, 1, \cdots, P+1$. Each pair of nodes has an edge $e_{i_1 i_2}$ connecting the lower-numbered node n_{i_1} with the higher-numbered node n_{i_2}, which consists of an operator $O_{i_1 i_2}$ selected from a pre-defined operator set $\mathcal{O}$. The node n_{i_2} is calculated as $n_{i_2} = O_{i_1 i_2}(n_{i_1})$.

To reduce the searching complexity, we set each immediate node to keep only a single edge and the output node n_{P+1} to be the sum of all other $P+1$ nodes. Denote the path from the input node n_0 to the i-th intermediate node as $\mathcal{P}_i$ and the node index at the k-th step in the i-th path as $\mathcal{P}_i(k)$. Suppose $\mathcal{P}_i$ has K_i steps, such that $\mathcal{P}_i(0) = 0$ and $\mathcal{P}_i(K_i) = i$. We construct a side-output branch by attaching an LSU U_s to the last convolutional layer of the $s-th$ network stage, where the input node n_0^s and output node n_{P+1}^s are actually aforementioned V_s and V'_s in Eq. 5. Thereafter, the s-th LSU is formulated as

$$\begin{aligned} V'_s &= \sum_{i=0}^{P} \Big(\prod_{k=1}^{K_i} O_{\mathcal{P}_i^s(k-1),\mathcal{P}_i^s(k)}\Big)(V_s) \\ &= \sum_{i=0}^{P} O_{\mathcal{P}_i^s}(V_s) \in Span(O^s(V_s)), \end{aligned} \tag{6}$$

where $O_{\mathcal{P}_i^s} = \prod_{k=1}^{K_i} O_{\mathcal{P}_i^s(k-1),\mathcal{P}_i^s(k)}$ denotes the composite of operators in the path $\mathcal{P}_i^s$ and $O^s = \sum_{i=0}^{P} O_{\mathcal{P}_i^s}$ the transformation posed on the input node n_0, which is defined by the combination of operators at all paths.

According to Eq. 4, we conclude that by selecting proper operators and edges the unit can transform the input nodes to facilitate feature space expansion. Accommodating the third challenge defined in Sec. III-A, we add skip, convolutions and dilated convolutions with various kernel sizes and dilation coefficients in $\mathcal{O}$ to pertinently enrich the semantic hierarchy and the scale granularity of features.

Linear Span Pyramid (LSP). To tackle the challenges in multi-stage feature integration, we materialize the complementary learning strategy by stacking LSUs to construct the pyramid search space with intermediate supervisions.

In specific, according to Eq. 5, we add short connections among LSUs in a deep-to-shallow fashion to merge the spanning sets of the corresponding feature subspaces. Each LSU accepts and accumulates input features from the backbone and/or the output of LSUs from deeper stages, the channels and resolutions of which are respectively aligned to those of the current stage by 1 × 1 convolution and upsampling. To ease the noise caused by feature upsampling, we restrict the upsampling rate to 4× at most, and each LSU only accepts features from two adjacent deeper stages. With intermediate loss layers, LSUs are assigned with different supervision priorities to progressively force the feature subspaces to expand towards complementary with each other. We additionally use an LSU (U_{S+1}) to accept outputs from all other LSUs to further expand the feature sum-space, Fig. 2.

B. Adaptive Linear Span by Architecture Search

Regarding the LSP as a sup-network, Fig. 2, we instantiate an AdaLSN in the architecture search space to perform linear span. To fulfill this purpose, we propose to encode the LSP into a chromosome matrix and use the genetic algorithm for adaptive architecture search.

Architecture Encoding. LSP consists of a set of operators and connections, which are partitioned into four encoding areas, Fig. 2. These four encoding areas are respectively defined to handle the four challenges aforementioned in Sec. III-A.

The first encoding area corresponds to connections between the backbone network and the LSUs, which are denoted as a string $g_{00} = \delta_{S_1}\delta_{S_2}\cdots\delta_{S_S}$. δ is a 0-1 binary variable where $\delta_{S_s} = 1$ $(1 \le s \le S)$ mean that U_s is connected with the s-th stage, otherwise to be discarded. The shallow network stages output high resolution features, which require larger memory but have low representation capability, while deep stages output low resolution features with coarse scale granularity, which require smaller memory. Within the first encoding area, the training procedure requires to adaptively determine connections between the backbone network and the LSUs so that features of shallow and deep stages can be optimally selected.

The second encoding area corresponds to the connections among LSUs, which are represented as a string $g_{s0} = \delta^s_{s+1} \cdots \delta^s_{S+1}$ and $g_{(S+1)0} = \delta_{U_1}\delta_{U_2} \cdots \delta_{U_S}$. $\delta^s_{s+1} \cdots \delta^s_S$ $(1 \leq s \leq S)$ denotes the connection states of unit U_s with $U_{s+1}, \cdots, U_S$ and δ_{U_s} denotes the connection states of U_s with U_{S+1}. This encoding area aims to balance the advantage brought by feature complementarity and the degradation caused by the up-sampling operation during multi-stage feature integration.

The third encoding area contains operators and edges in each LSU to enrich the semantic hierarchy and scale granularity of the feature subspace, which should be adaptive to the characteristics of each stage. This encoding area is denoted as a string $g_{s1} = e^s_1 e^s_2 \cdots e^s_P o^s_1 o^s_2 \cdots o^s_P$, $1 \leq s \leq S+1$. e_p $(p = 1, 2, \cdots, P)$ denotes the index of the node connected to the p-th node and o_p the corresponding operator on it. e_p varies from 0 to $p-1$ and o_p varies from 0 to $|\mathcal{O}|$. Based on the above defined three encoding areas, LSUs are defined as $U_s = (\delta_{S_s} \quad g_{s0} \quad g_{s1})$ $(1 \leq s \leq S)$ and $U_{S+1} = (g_{(S+1)0} \quad g_{(S+1)1})$.

The fourth encoding area is about the intermediate supervisions, which are denoted as string $g_{01} = \delta_{L_1}\delta_{L_2} \cdots \delta_{L_S}$, where δ_{L_s} $(1 \leq s \leq S)$ indicates the connection state of the U_s with the s-th loss layer. The adaptive architecture search in this encoding area facilitates balancing advantages brought by linear span and the disadvantages (*e.g.*, error accumulation) of intermediate supervisions.

Combining the four encoding areas, we define the architecture search space for AdaLSN. The search space is formulated as a matrix $(G = (g_{st})_{(S+2)\times 2})$ with dimensionality $(S+2) \times 2$ where the elements in G are denoted as

$$\begin{cases} g_{00} = \delta_{S_1}\delta_{S_2} \cdots \delta_{S_S}, & \\ g_{01} = \delta_{L_1}\delta_{L_2} \cdots \delta_{L_S}, & \\ g_{s0} = \delta^s_{s+1} \cdots \delta^s_{S+1}, & 1 \leq s \leq S, \\ g_{s1} = e^s_1 e^s_2 \cdots e^s_P o^s_1 o^s_2 \cdots o^s_P, & 1 \leq s \leq S+1, \\ g_{(S+1)0} = \delta_{U_1}\delta_{U_2} \cdots \delta_{U_S}, & \end{cases} \tag{7}$$

where g_{00} corresponds to the multiple side-outputs, g_{01} the intermediate supervision, g_{s0} $(1 \leq s \leq S)$ the short connection, and g_{s1} the feature transform. The search space covers most side-output network architectures. For example, the SRN [8] containing manual design of multiple side-outputs, short connections between adjacent side-output branches, and intermediate supervisions, falls into this space. SRN is represented as

$$G_{SRN} = \begin{bmatrix} 11111 & 11111 \\ 01000 & - \\ 00100 & - \\ 00010 & - \\ 00001 & - \\ - & - \\ 11111 & - \end{bmatrix}.$$

Architecture Search. Architecture search aims to find out sub-structures in the search space for adaptive linear span. Considering the large dimensionality of the search space, the genetic algorithm is employed for architecture search. In the searching procedure, AdaLSN is defined as a chromosome G, the element g_{ij} in the character string matrix G as a gene segment, and each byte in the encoded strings as a gene, Eq. 7. A chromosome is an individual in the population. Genes and gene segments constitute the smallest units for mutation and crossover, respectively. The initial population size is set to 24. Half of the population is randomly selected from all possible chromosomes with equal probability. The other 12 individuals are initialized by mutating the ASPP-like module [36] once for LSUs and randomly sampling for the rest structures.

To select superior individuals for evolution, the prediction performance with sufficient training is accurate and reliable but too time-consuming. We resort to sort the loss at one thousand training iterations of the population in each generation and update the 8 individuals with the smallest loss value among all historical generations. The selected top-8 individuals survive to the next generation.

In specific, we apply the crossover and mutation operation upon the selected 8 individuals to generate another 8 individuals. To perform crossover, we randomly select a pair of chromosomes and randomly exchange one of their gene segments in the same position. In this way, competitive sub-structures could be preserved and evolve to optimal architectures. To avoid local optimum and increase the gene diversity of the population, we randomly change the value of every byte of all gene segments of the chromosomes in the current generation for mutation operation. Each gene in the chromosome has a chance to be changed and excellent gene combinations are compete to survive and passed to the next generation. In this way, the individuals evolve for feature space expansion.

All individuals in the current generation are trained for evaluation and the surviving (selected) individuals are encoded for the next generation search. After tens of generations, the search procedure stops and the individual with the smallest training loss outputs as the optimal network architecture.

V. Experiments

In this section, the experimental settings are first described and the modules of AdaLSN are analyzed with ablation studies. AdaLSN is then evaluated and compared with the state-of-the-arts. Finally, AdaLSN is applied on other image-to-mask tasks including edge detection and road extraction.

A. Experimental Setting

Datasets. Five commonly used skeleton datasets, including SK-LARGE [37], SK-SMALL [7], SYM-PASCAL [8], SYMMAX300 [38], and WH-SYMMAX [39], are used to evaluate AdaLSN. SK-LARGE involves skeletons from about 16 classes of objects, and contains 746/745 training and test images sampled from MS-COCO [40]. SK-SMALL (SK506) is sampled from MS-COCO but contains fewer images. SYM-PASCAL contains 648/787 training and test

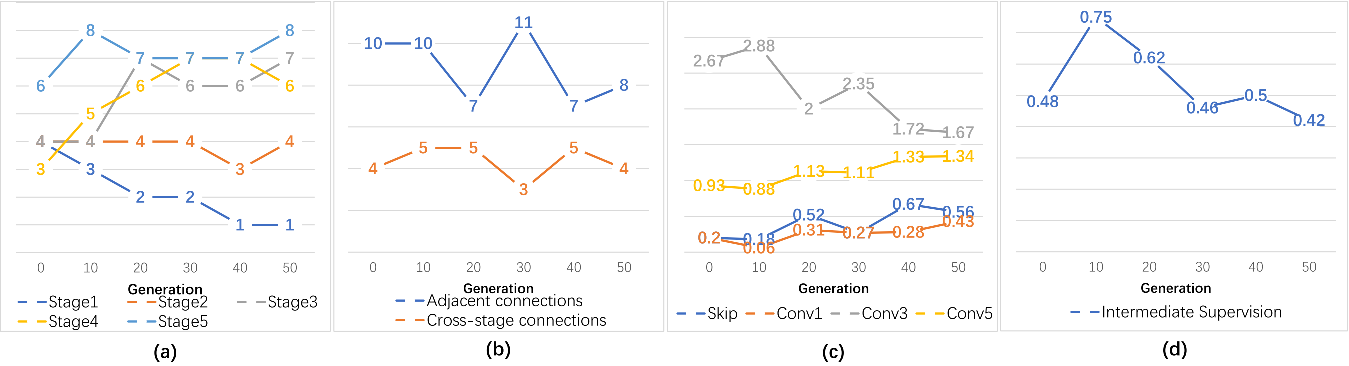


Fig. 3. Adaptability of architecture search. Each sub-figure shows a statistic of the searched (top-8) architectures. (a) Number of side-outputs attached to each network stage in A_1; (b) Numbers of short connections in A_2; (c) Operator distributions (average number of operators in an LSU) in A_3; (d) Ratio of intermediate supervision in A_4.

images annotated from the segmentation subset of PASCAL VOC 2011 [41]. SYMMAX300 has 200 training images and 100 test images, which are annotated on BSDS300 [42]. WH-SYMMAX is developed for skeleton detection with 228/100 training and test images.

Implementation details. AdaLSN is implemented using PyTorch and runs on NVIDIA TITAN RTX GPUs (with 24 GB of memory). In the training phase, the raw training set without pre-processing is used to search the architectures in each generation, and fifty valid images are used for evaluation. For training, we use the Adam optimizer [43] with the initial learning rate 1e-6, a momentum (0.9, 0.999), and a weight decay 0.0005. The batch size is set as 1 while network parameters are updated every 10 times of forward propagation. In the retaining phase, the searched architecture is optimized for 25 epochs. The learning rate is fixed during searching and reduced a magnitude after 20 epochs. We use multiple data augmentation techniques, including resizing to 3 scales (0.8x, 1.0x, and 1.2x), rotating for 4 directions (0°, 90°, 180°, and 270°), flipping in 2 orientations (left-to-right and up-to-down), and resolution normalization [17].

Evaluation protocol. Following the settings in [38], F-measure score (F-score) and Precision-Recall (PR) curves are used as evaluation metrics. While PR curves diagnose the binary classification results under different thresholds, F-score provides a single score weighting both precision and recall.

B. Ablation Study

Ablation studies are performed on SK-LARGE [37], which is the most used benchmark for object skeleton detection. In the search phase, by default, we set the index of side-outputs as $\{1, 2, 3, 4, 5\}$, the channel number of the nodes (Channel Number) as 32, the intermediate node number (Intermediate Node) as 4. The kernel size and dilation rate of the convolution layers in the alternative operator set (Operator) vary in $\{1, 3, 5\}$ and $\{1, 2, 4, 8\}$, respectively.

Architecture Search. In Table II, one can see that the searched AdaLSNs significantly outperform the randomly sampled architectures by $\sim$ 3% F-score. AdaLSNs also demonstrate robustness to initializations.

TABLE II
EFFECT OF ARCHITECTURE SEARCH

	Random Sampling	**Architecture Search**	
		Random Init.	**ASPP-like Init.**
F-score	0.715 ± 0.011	0.749	0.753

TABLE III
PERFORMANCE EVOLUTION DURING ARCHITECTURE SEARCH

Generation	**0**	**10**	**20**	**30**	**40**	**50**	**60**
F-score	0.727	0.744	0.751	0.751	0.750	**0.753**	0.752

TABLE IV
EVALUATION OF ADALSN ENCODING AREAS. "RAND". DENOTES RANDOM SAMPLING. ENCODING AREAS $\{A_1, A_2, A_3, A_4.\}$ ARE INTRODUCED IN FIG. 2

A_1	A_2	A_3	A_4	**F-score**
✓	✓	✓	✓	0.753
Rand.	✓	✓	✓	0.739
w/o	✓	✓	✓	0.727
✓	**Rand.**	✓	✓	0.749
✓	**w/o**	✓	✓	0.720
✓	✓	**Rand.**	✓	0.746
✓	✓	**w/o**	✓	0.738
✓	✓	✓	**Rand.**	0.750
✓	✓	✓	**w/o**	0.750

In Table III, we search fifty generations and re-train the individuals selected in each population iteration, and report the performance every ten generations. It can be seen that the top-1 individuals evolve quickly and the F-score increases significantly from 0.727 to 0.751, and slightly improves to 0.753 after 30 more generations.

In Table IV, we validate the effectiveness of the encoding areas including multiple side-outputs (A_1), short connection (A_2), feature transform (A_3), and intermediate supervision (A_4). Specifically, we compare the performance of partial search by screening an encoding area (w/o) and randomly sampling the corresponding area after complete search (Rand.). Experiments show that the encoding areas can

TABLE V

ABLATION STUDY OF THE PROPOSED ADALSN UNDER DIFFERENT SEARCH SPACE SETTINGS

Search Space			**Search Phase**		**Retraining Phase**		
			Memory(G)	**Search time(h)**	**Param(M)**	**Runtime(ms)**	**F-score**
Unit-level	**Channel Number**	2	$\leq$ 8.1	43.4	14.7	12.52	0.726
		8	$\leq$ 9.9	46.0	14.8	12.80	0.728
		16	$\leq$ 12.7	46.3	14.9	12.56	0.749
		32	$\leq$ 19.8	47.5	15.2	13.87	0.753
		64	$\approx$ 24.0	49.2	16.1	14.21	0.759
		128	$\geq$ 24.0	–	–	–	–
	Intermediate Node	0	$\leq$ 12.8	43.1	14.7	9.13	0.704
		1	$\leq$ 13.8	44.0	14.9	9.46	0.740
		2	$\leq$ 15.6	45.3	14.9	10.97	0.746
		3	$\leq$ 17.3	46.7	15.1	12.03	0.746
		4	$\leq$ 19.8	47.5	15.2	13.87	0.753
		5	$\leq$ 20.7	49.5	15.2	12.67	0.749
	Operator {kernel size}- {dilation rate}	{1} - {1}	$\leq$ 18.8	45.1	14.8	12.18	0.713
		{1,3,5} - {1}	$\leq$ 19.8	49.0	15.0	13.32	0.750
		{1,3,5} - {1,2,4,8}	$\leq$ 19.8	47.5	15.2	13.87	0.753
		{1,3,5} - {1,8,16,24}	$\leq$ 19.8	48.1	15.1	12.24	0.748
		{1,3,5} - {1,2,4,8,16,24}	$\leq$ 19.8	46.5	15.0	13.38	0.749
Pyramid-level	**Side-output**	{1,2,3,4,5}	$\leq$ 19.8	47.5	15.2	13.87	0.753
		{2,3,4,5}	$\leq$ 13.2	42.2	15.2	14.42	0.751
		{3,4,5}	$\leq$ 11.3	39.5	15.1	12.34	0.750
		{4,5}	$\leq$ 10.3	38.4	15.0	12.13	0.746
		{5}	$\leq$ 9.8	37.5	14.9	10.12	0.723

significantly boost the performance, validating that the search space is more complete.

Architecture Adaptability. In Fig. 3, we validate the adaptability of the four encoding areas using statistics figures. In A_1, Fig. 3a shows that the side-output numbers in deep stages are significantly larger than those in shallow stages. This is because the features from deep stages have small resolutions yet coarse-scale granularity and strong representation capability. More deep stage features imply higher accuracy. The adaptive configuration of multi-stage features greatly eases the scale variation problem. In A_2, we divide the short connections adjacent ones and cross-stage ones, Fig. 3b. Considering the degradation caused by feature upsampling, the adjacent connections are preferred.

In A_3, we compare LSU operator distributions, including skips and convolutions with different kernel sizes, Fig. 3c. For the moderate performance gain (0.750 vs 0.753) with dilated convolutions, we calculate the average number of different convolutions in an LSU according to their kernel size only. It can be seen that the $Conv_3$ and $Conv_5$ are more preferred than $Conv_1$. This is because they correspond to stronger representation capability and larger receptive fields to ease the imbalance of semantic levels and scale granularity. In the early search phrase, the number of $Conv_3$ is noticeably larger than that of $Conv_5$ while the gap reduces when the search goes on. The reason lies in that with fewer parameters, $Conv_3$ layers are easy to be optimized. When sufficient evolution takes place, $Conv_5$ with large a receptive field and representation capability is competitive. This validates AdaLSN's adaptability on network architecture configuration.

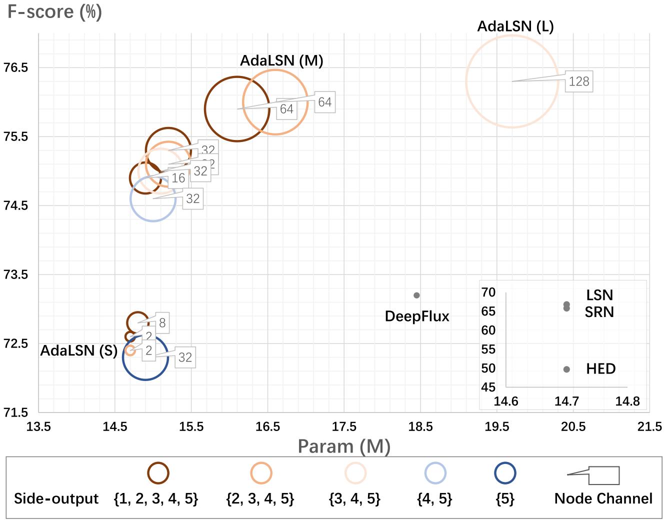


Fig. 4. Visualization of the parameter and F-score of AdaLSN exemplars with the VGG backbone. The circle size indicates the channel number. (Best view in color and zoom in).

In A_4, we calculate the ratio of intermediate supervision, Fig. 3d. It can be seen that in the early phase, the architecture is more dependent on intermediate supervision because intermediate supervision forces the feature subspaces to expand and fit the ground-truth. With sufficient training after tens of generations, architectures with less intermediate supervisions ease error accumulation and implement complementary learning for feature space expansion.

AdaLSN Exemplars. The computational cost and the prediction accuracy of AdaLSNs largely depend on the search space settings, Table V. Specifically, we explore the effect of "channel number", "intermediate node", and "operator"

TABLE VI

COMPARISON OF ADALSN EXEMPLARS WITH DIFFERENT SEARCH SPACE SETTINGS AND BACKBONE NETWORKS

		Search Space			Search Phase		Retraining Phase		
	Backbone	**Side-output**	**Channel Number**	**Intermediate Node**	**Memory(G)**	**Search time(h)**	**Param(M)**	**Runtime(ms)**	**F-score**
(S)	**VGG16**	{2,3,4,5}	2	1	$\leq$ 7.2	43.1	14.7	12.22	0.724
(M)	**VGG16**	{2,3,4,5}	64	4	$\leq$ 16.0	45.3	16.6	14.67	0.760
(L)	**VGG16**	{3,4,5}	128	4	$\leq$ 19.2	47.2	19.7	15.53	0.763
(L)	**ResNet50**	{3,4,5}	128	4	$\approx$ 24.0	138.7	30.9	43.12	0.764
(L)	**Res2Net**	{3,4,5}	128	4	$\approx$ 24.0	117.3	26.1	34.23	0.768
(L)	**InceptionV3**	{3,4,5}	128	4	$\approx$ 24.0	193.2	32.4	69.37	0.786

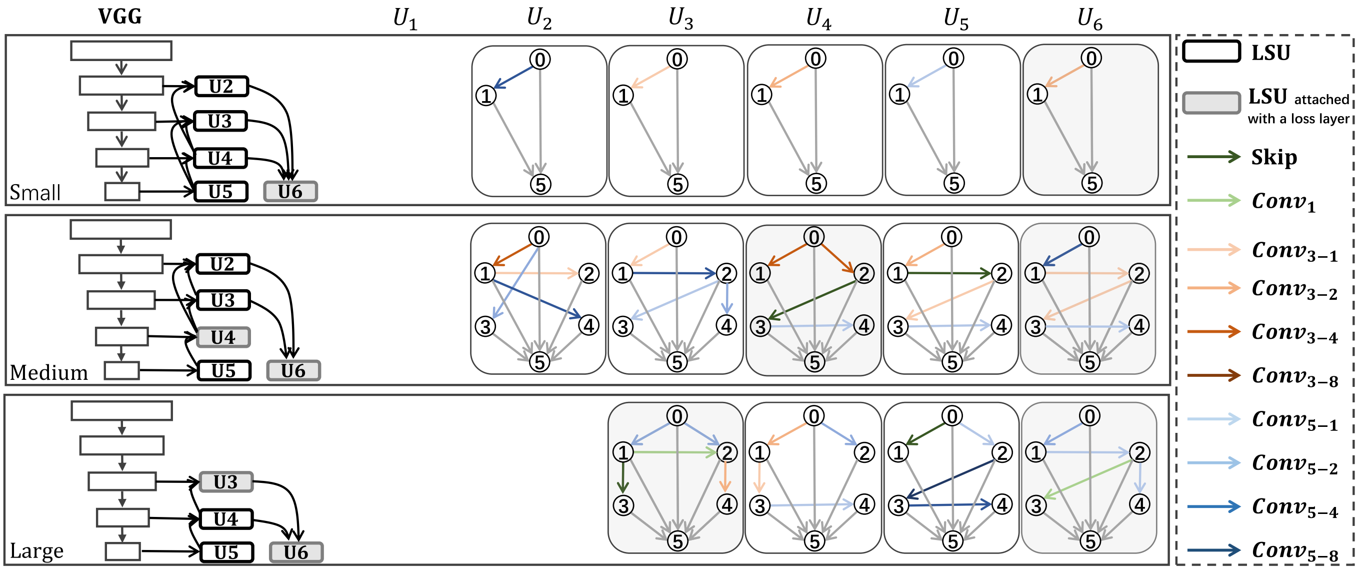


Fig. 5. The searched exemplar architectures of AdaLSN.

({kernel size}-{dilation rate}) at the unit level; and the effect of "side-output" at the pyramid level.

When the channel number increases from 2 to 64, the memory cost increases from 8.1G to 24.0G. The parameters and runtime of the searched network in the retraining phase moderately increases from (14.7M, 12.52ms) to (16.1M, 14.21ms), while the F-score significantly increases from 0.726 to 0.759, Table V. We conclude that for our approach the channel number is an important factor in balancing the computational cost and accuracy.

When increasing intermediate nodes in each LSU from 0 to 4, the computational cost in both phases increases moderately, while the F-score greatly increases from 0.704 to 0.753. It is worth noting that comparing LSUs without any intermediate node, the F-score significantly increases from 0.704 to 0.740 with only 1 intermediate node used in each LSU. This validates the importance of operators and their combinations in feature transform for feature space expansion. However, when the intermediate node number increases to 5, the F-score of AdaLSN falls to 0.749. The reason lies in that more intermediate nodes means higher search complexity, which significantly improves the difficulty of architecture optimization.

We evaluate the convolution operators with different kernel sizes and dilation rates. Table V shows that convolutions with kernel sizes larger than 1 have superiority for feature space expansion, $e.g.$, the F-score increases from 0.713 to 0.750 with negligible computational cost. When the dilation rate enlarges properly as $\{1, 2, 4, 8\}$, which enriches the scale granularity, the performance improves up to 0.753. Nevertheless, the F-score falls upon larger dilation rates, because convolution operators with large dilation tend to missing feature details. Accordingly, the kernel sizes and dilation rates are set as $\{1, 3, 5\} - \{1, 2, 4, 8\}$.

As features from shallow stages are less representative but have higher resolution, we gradually reduce the shallow side-outputs in the search phase to figure out the effect of each side-output. As shown in the bottom row of Table V, without S_1 and S_2, the memory cost significantly reduces from 19.8G to 11.3G, the search time decreases from 47.5h to 39.5h, and the F-score slightly reduces. In Fig. 4, we visualize the figures of ablation on channel number and side-output in Table V and Table VI using the VGG backbone. One can find that when reducing the shallow network stages $\{1, 2\}$ while increasing the channel number from 32 to 128, F-scores of the searched architectures significantly increase. When reducing the deep stages with medium channel numbers, the F-score significantly drops. For instance, with channel number 32, F-score drops from 0.746 to 0.723 when the side-outputs reduce from $\{4, 5\}$ to $\{5\}$.

Empirically, we set the side-output as $\{2, 3, 4, 5\}$ and change the channel number and intermediate node to (2, 1) for a

TABLE VII
PERFORMANCE COMPARISON OF STATE-OF-THE-ART APPROACHES ON COMMONLY USED SKELETON\SYMMETRY DETECTION DATASETS

Methods	**Datasets**				
	SK-LARGE [37]	**SK-SMALL** [38]	**WH-SYMMAX** [40]	**SYM-PASCAL** [8]	**SYMMAX300** [39]
MIL [47]	0.353	0.392	0.365	0.174	0.362
HED [6]	0.497	0.541	0.732	0.369	0.427
RCF [14]	0.626	0.613	0.751	0.392	-
FSDS [7]	0.633	0.623	0.769	0.418	0.467
SRN [8]	0.658	0.632	0.780	0.443	0.446
LSN [10]	0.668	0.633	0.797	0.425	0.480
Hi-Fi [9]	0.724	0.681	0.805	0.454	-
DeepFlux [15]	0.732	0.695	0.840	**0.502**	0.491
AdaLSN (ours)	**0.786**	**0.740**	**0.851**	0.497	**0.495**

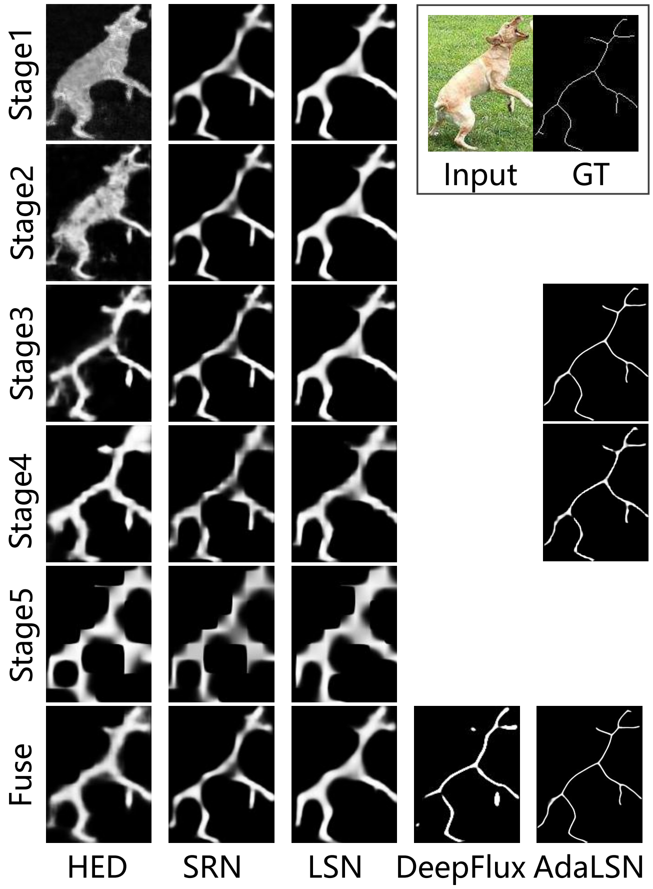


Fig. 6. Comparison of predicted masks, which reflect the feature space expansion in different feature stages.

small AdaLSN(S), while (64, 4) for a medium AdaLSN(M). By increasing the channel number to 128 while reducing the side-output to $\{3, 4, 5\}$, we have a large AdaLSN(L). The detailed structures of three versions of AdaLSN with VGG [44] are depicted in Fig. 5. Comparing with the state-of-the-art networks, such as HED [6], SRN [8], and LSN [10], AdaLSN(S) has significant accuracy improvement ($0.056 \sim 0.227$ F-score gain) with negligible parameter cost. Comparing with DeepFlux [15], AdaLSN(M) achieves better F-score (0.760 vs 0.724) with less params (16.6 M vs 18.3 M). In Table VI, with the InceptionV3 [45] backbone, AdaLSN(L) achieves 0.786, improving the state-of-the-art with a large margin.

Linear Span. In Fig. 6, we compare the skeleton predictions of the state-of-the-art approaches about a dog. It can be seen that with only parallel side-outputs for scale-aware feature utilization, the predictions of HED [6] suffer background noise in shallow network stages and mosaic effects in deep stages. By adding short connections among side-outputs for feature integration, SRN [8] purses the residual between adjacent stages to progressively improve the predictions in a deep-to-shallow fashion so that the predictions in shallow stages are improved. To learn more complementary features and span a larger feature space, LSN [10] builds dense short connections among side-outputs for feature space expansion. However, without explicit feature transform, LSN moderately improves the result comparing with SRN. Deepflux [15] generates a slim skeleton with manually designed ASPP modules [16] and the context flux constrain. AdaLSN incorporates these advantages in the linear span view and updates them with the adaptive architecture search algorithm. As the feature subspaces are fully expanded while forced to be complementary with each other AdaLSN architectures are adaptive to skeleton characteristics. This explains why the prediction results at all stages are precise, clear, and consecutive.

C. Performance and Comparison

AdaLSN outperforms the state-of-the-art methods in terms of F-score, Tab. VII. The results of AdaLSN (large) is reported by the architecture searched upon SKLARGE and retrained on each dataset.

For SKLARGE, Hi-Fi [9] with additional scale-associated ground-truth achieves the F-score of 72.4%; DeepFlux [15] with skeleton context flux reports the highest skeleton detection performance up to date of 73.2%. Without additional supervision information and explicitly geometric modeling, AdaLSN achieves a significantly higher F-score of 78.6%, which outperforms that of DeepFlux by 5.4%. When comparing to DeepFlux with transferring the architecture to other skeleton\symmetry datasets, AdaLSN respectively improves the F-score by 4.5%, 1.1% and 0.4% on SK-SMALL, WH-SYMMAX, and SYMMAX300, and achieves comparable performance on SYM-PASCAL.

The detected skeleton results are shown in Fig. 7, where AdaLSN produces skeleton maps of different granularity with better continuity and higher accuracy. In comparison, HED produces skeletons with significant noise. SRN and LSN

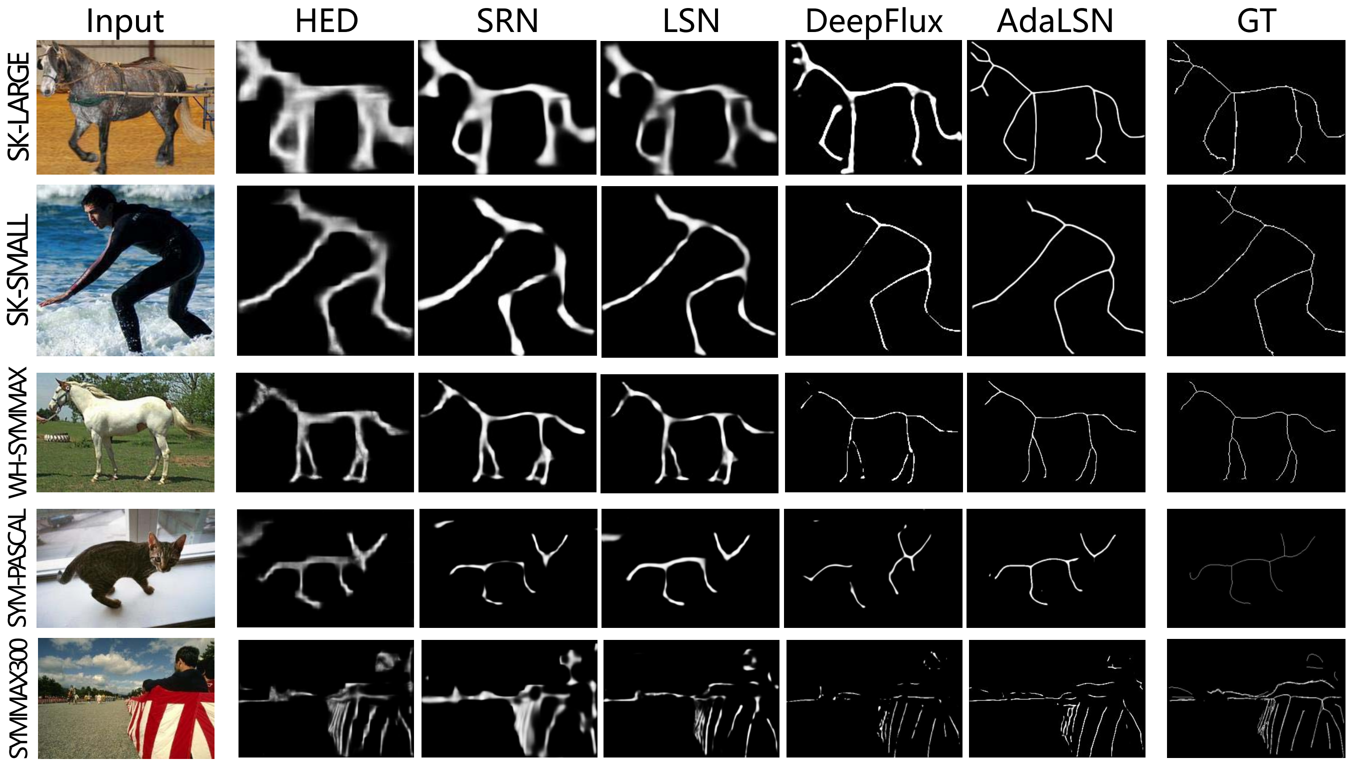


Fig. 7. Skeleton detection examples by state-of-the-art approaches on commonly used skeleton detection datasets including SK-LARGE [37], SK-SMALL [7], WH-SYMMAX [39], SYM-PASCAL [8], and SYMMAX300 [38].

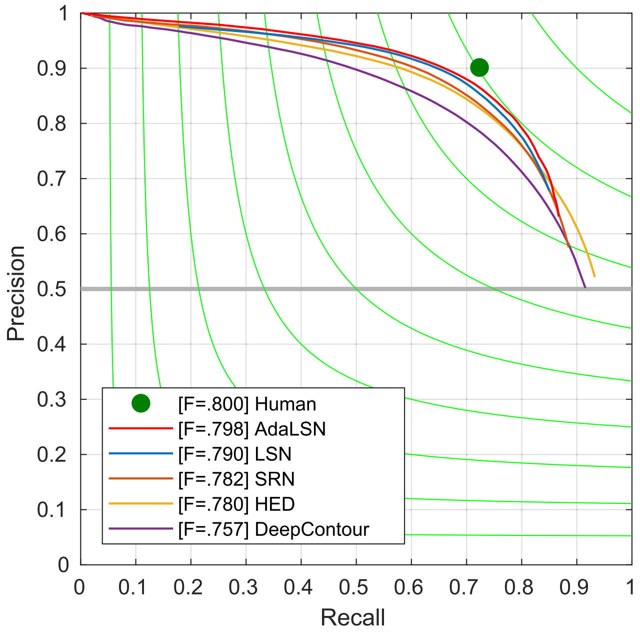


Fig. 8. The PR-curve on the BSDS500 edge detection dataset.

predict relatively clearer skeletons which however are not smooth. Deepflux reports clearer and slimmer results, which still have false positive points and dis-continual segments.

D. Other Image-to-Mask Tasks

AdaLSN is applied on other image-to-mask tasks including edge and road extraction. The good performance of AdaLSN demonstrates its general applicability to image-to-mask tasks.

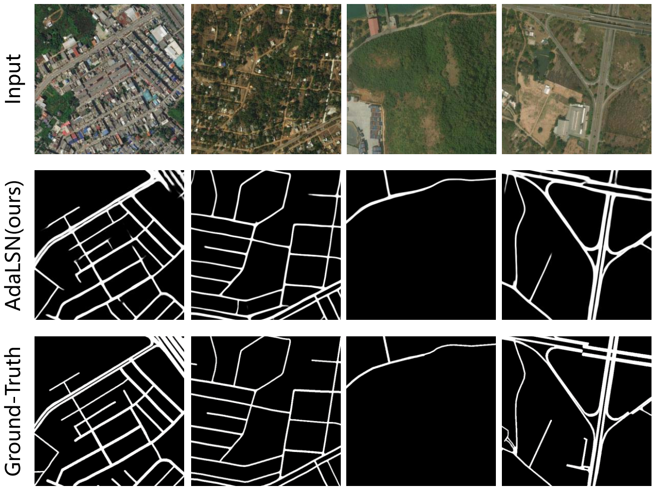


Fig. 9. Examples of road extraction results by AdaLSN.

Edge Detection. We directly apply the model searched on the skeleton dataset to edge detection on the BSDS500 dataset [46], which is composed of 200 training images, 100 validation images, and 200 testing images. F-score is used as the evaluation metric which is calculated by choosing an optimal scale for the entire dataset. As shown in Fig. 8, AdaLSN reports the highest F-score of 0.798, which has a very small gap (0.002) to human performance.

Road Extraction. AdaLSN is also applied for road extraction in aerial images, using the dataset from the DeepGlobe Road Extraction Challenge [47]. As the ground-truth annotations of the test dataset are not published, we randomly select 100 images from the training dataset as a test dataset and partitioned the rest images to 512 × 512 sub-images for training. Ada-LSN slightly outperforms DLinkNet [36] (0.6354 vs 0.6321 mAP), one state-of-the-art approach for road extraction. The detected road masks by AdaLSN are very close to the ground-truth masks, Fig. 9.

VI. Conclusion

Object skeleton detection is a representative image-to-mask task in computer vision yet remains challenged by objects in different granularity. In this paper, we proposed adaptive linear span network (AdaLSN), and automatically configured and integrated scale-aware features for object skeleton detection. Following the linear span theory and driven by neural architecture search (NAS), AdaLSN found complementary features for subspace and sum-space expansion and thereby optimized the multi-scale feature integration in a data-adaptive fashion. The significant higher accuracy compared with the state-of-the-arts on commonly used benchmarks and the general applicability to image-to-mask tasks, *e.g.*, edge detection and road extraction, demonstrated the effectiveness of AdaLSN. The NAS-driven feature space expansion provided a fresh insight for feature representation learning. So far, AdaLSN only models and optimizes the side-output branches with a given backbone network. To apply AdaLSN upon backbone network optimization for stronger feature representation could be exploited in the future.

References

[1] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4724–4732.

[2] C. L. Teo, C. Fermüller, and Y. Aloimonos, "Detection and segmentation of 2D curved reflection symmetric structures," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1644–1652.

[3] Z. Zhang, W. Shen, C. Yao, and X. Bai, "Symmetry-based text line detection in natural scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 2558–2567.

[4] T. Lee, S. Fidler, and S. Dickinson, "Learning to combine mid-level cues for object proposal generation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1680–1688.

[5] C. Liu, W. Ke, J. Jiao, and Q. Ye, "RSRN: Rich side-output residual network for medial axis detection," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 1739–1743.

[6] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1395–1403.

[7] W. Shen, K. Zhao, Y. Jiang, Y. Wang, Z. Zhang, and X. Bai, "Object skeleton extraction in natural images by fusing scale-associated deep side outputs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 222–230.

[8] W. Ke, J. Chen, J. Jiao, G. Zhao, and Q. Ye, "SRN: Side-output residual network for object symmetry detection in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 302–310.

[9] K. Zhao, W. Shen, S. Gao, D. Li, and M.-M. Cheng, "Hi-Fi: Hierarchical feature integration for skeleton detection," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 1191–1197.

[10] C. Liu, W. Ke, F. Qin, and Q. Ye, "Linear span network for object skeleton detection," in *Proc. ECCV*, Sep. 2018, pp. 133–148.

[11] P. Lax, *Linear Algebra and its Applications*, vol. 2. Hoboken, NJ, USA: Wiley, 2007.

[12] T. S. H. Lee, S. Fidler, and S. Dickinson, "Detecting curved symmetric parts using a deformable disc model," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 1753–1760.

[13] N. Widynski, A. Moevus, and M. Mignotte, "Local symmetry detection in natural images using a particle filtering approach," *IEEE Trans. Image Process.*, vol. 23, no. 12, pp. 5309–5322, Dec. 2014.

[14] Y. Liu *et al.*, "Richer convolutional features for edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 8, pp. 1939–1946, Aug. 2019.

[15] Y. Wang, Y. Xu, S. Tsogkas, X. Bai, S. Dickinson, and K. Siddiqi, "DeepFlux for skeletons in the wild," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5287–5296.

[16] L.-C. 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 Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.

[17] W. Xu, G. Parmar, and Z. Tu, "Geometry-aware end-to-end skeleton detection," in *Proc. BMVC*, 2019, p. 256.

[18] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. ICLR*, 2017, pp. 1–16.

[19] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8697–8710.

[20] C. Liu *et al.*, "Progressive neural architecture search," in *Proc. ECCV*, Sep. 2018, pp. 540–555.

[21] X. Du *et al.*, "SpineNet: Learning scale-permuted backbone for recognition and localization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11589–11598.

[22] E. Real *et al.*, "Large-scale evolution of image classifiers," in *Proc. ICML*, 2017, pp. 2902–2911.

[23] L. Xie and A. Yuille, "Genetic CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1388–1397.

[24] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, Jul. 2019, pp. 4780–4789.

[25] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2018, pp. 2787–2794.

[26] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," in *Proc. ICML*, Jul. 2018, pp. 4092–4101.

[27] X. Zheng, R. Ji, L. Tang, B. Zhang, J. Liu, and Q. Tian, "Multinomial distribution learning for effective neural architecture search," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1304–1313.

[28] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, "Neural architecture optimization," in *Proc. NIPS*, 2018, pp. 7827–7838.

[29] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. ICLR*, 2019, pp. 1–13.

[30] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1294–1303.

[31] Y. Xu *et al.*, "PC-DARTS: Partial channel connections for memory-efficient architecture search," in *Proc. ICLR*, 2020, pp. 1–13.

[32] T. Mingxing and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. ICML*, 2019, pp. 1–10.

[33] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "NAS-FPN: Learning scalable feature pyramid architecture for object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7036–7045.

[34] Y. Weng, T. Zhou, Y. Li, and X. Qiu, "NAS-UNet: Neural architecture search for medical image segmentation," *IEEE Access*, vol. 7, pp. 44247–44257, 2019.

[35] C. Liu *et al.*, "Auto-DeepLab: Hierarchical neural architecture search for semantic image segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 82–92.

[36] L. Zhou, C. Zhang, and M. Wu, "D-LinkNet: LinkNet with pretrained encoder and dilated convolution for high resolution satellite imagery road extraction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 182–186.

[37] W. Shen, K. Zhao, Y. Jiang, Y. Wang, X. Bai, and A. Yuille, "DeepSkeleton: Learning multi-task scale-associated deep side outputs for object skeleton extraction in natural images," *IEEE Trans. Image Process.*, vol. 26, no. 11, pp. 5298–5311, Nov. 2017.

[38] S. Tsogkas and I. Kokkinos, "Learning-based symmetry detection in natural images," in *Proc. ECCV*, 2012, pp. 41–54.

[39] W. Shen, X. Bai, Z. Hu, and Z. Zhang, "Multiple instance subspace learning via partial random projection tree for local reflection symmetry in natural images," *Pattern Recognit.*, vol. 52, pp. 306–316, Apr. 2016.

[40] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 664–676, Apr. 2017.

[41] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[42] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th IEEE Int. Conf. Comput. Vis. (ICCV)*, vol. 2, Jul. 2001, pp. 416–423.

[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015, pp. 1–15.

[44] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICCV*, 2015, pp. 1–14.

[45] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.

[46] P. Arbelaez, M. Maire, C. C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, May 2011.

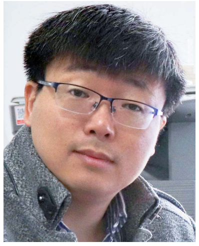
[47] I. Demir *et al.*, "DeepGlobe 2018: A challenge to parse the earth through satellite images," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 172–181.

Chang Liu received the B.S. degree from Jilin University, Jilin, China, in 2012. He is currently pursuing the Ph.D. degree with the School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing, China. He has published more than ten articles in referred conferences, including ECCV, IEEE ICCV, and IEEE CVPR. His research interests include self-supervised learning, neural architecture design, and visual object detection.

Yunjie Tian received the B.S. degree from Jilin University, Jilin, China, in 2019. He is currently pursuing the Ph.D. degree with the School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing, China. His research interests include computer vision and machine learning, specifically for neural architecture design and visual object detection.

Zhiwen Chen received the B.E. degree in computer science from Shanghai Jiao Tong University, China, in 2012, and the M.E. degree in computer science from the National University of Singapore, Singapore, in 2014. From 2014 to 2016, he was a Video Analytic Researcher with Trakomatic Pte. Ltd., Singapore. He joined Alibaba Group, China, in 2017, where he is currently a Senior Algorithm Engineer. His research interests include computer vision and machine learning.

Jianbin Jiao (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from the Harbin Institute of Technology (HIT), China, in 1989, 1992, and 1995, respectively. From 1997 to 2005, he was an Associate Professor with HIT. Since 2006, he has been a Professor with the University of the Chinese Academy of Sciences, Beijing, China. He has authored over 50 articles in refereed conferences and journals. His research interests include image processing and pattern recognition.



Qixiang Ye (Senior Member, IEEE) received the B.S. and M.S. degrees in mechanical and electrical engineering from the Harbin Institute of Technology, China, in 1999 and 2001, respectively, and the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, in 2006. He has been a Professor with the University of Chinese Academy of Sciences since 2009. He was a Visiting Assistant Professor with the Institute of Advanced Computer Studies (UMIACS), University of Maryland, College Park, MD, USA, until 2013. He has published more than 100 articles in refereed conferences and journals, including IEEE CVPR, ICCV, ECCV, and IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (PAMI). His research interests include image processing, object detection, and machine learning. He received the Sony Outstanding Paper Award.