Applied Intelligence (2023) 53:25787–25801
https://doi.org/10.1007/s10489-023-04682-6

MSD-NAS: multi-scale dense neural architecture search for real-time pedestrian lane detection

Sui Paul Ang[1] · Son Lam Phung[1] · Soan T. M. Duong[2] · Abdesselam Bouzerdoum[1,3]

Accepted: 2 May 2023 / Published online: 11 August 2023

Abstract
Accurate detection of pedestrian lanes is a crucial criterion for vision-impaired people to navigate freely and safely. The current deep learning methods have achieved reasonable accuracy at this task. However, they lack practicality for real-time pedestrian lane detection due to non-optimal accuracy, speed, and model size trade-off. Hence, an optimized deep neural network (DNN) for pedestrian lane detection is required. Designing a DNN from scratch is a laborious task that requires significant experience and time. This paper proposes a novel neural architecture search (NAS) algorithm, named MSD-NAS, to automate this laborious task. The proposed method designs an optimized deep network with multi-scale input branches, allowing the derived network to utilize local and global contexts for predictions. The search is also performed in a large and generic space that includes many existing hand-designed network architectures as candidates. To further boost performance, we propose a Short-term Visual Memory mechanism to improve information facilitation within the derived networks. Evaluated on the PLVP3 dataset of 10,000 images, the DNN designed by MSD-NAS achieves state-of-the-art accuracy (0.9781) and mIoU (0.9542), while being 20.16 times faster and 2.56 times smaller than the current best deep learning model.

Keywords Pedestrian lane detection · Real-time video processing · Neural architecture search · Assistive navigation · Deep learning · Semantic segmentation

1 Introduction

Visual impairment is a disease that can affect the quality of life significantly. In 2015, around 36 million people globally suffered from blindness. This figure is estimated to reach 115 million by the year 2050 [1]. For vision-impaired people, accurately detecting the pedestrian lane is an essential criterion to navigate freely and safely. Currently, this task is performed using manual aids that are prone to errors, such as white canes and guide dogs [16]. Hence, there is a need for automatic pedestrian lane detection methods that are robust, accurate, and fast.

✉ Sui Paul Ang
academic@paul-ang.com

[1] School of Electrical, Computer and Telecommunications Engineering, University of Wollongong, Wollongong, Australia

[2] Computer Science Department, Le Quy Don Technical University, Hanoi, Vietnam

[3] College of Science and Engineering, Hamad Bin Khalifa University, Ar-Rayyan, Qatar

The early methods for automatic pedestrian lane detection use traditional image processing techniques. These methods are generally unsuitable for real-time systems because they are slow and ineffective. Several methods relied on white markers surrounding the pedestrian lanes, e.g., [20, 40]. This approach is ineffective because most pedestrian lanes are unmarked, and have arbitrary shapes and surfaces. Some methods relied on manually-extracted features or vanishing point estimation, e.g., [15, 30, 34]. These approaches are not robust because they are sensitive to scene variations.

The more recent automatic pedestrian lane detection methods use deep learning (DL) techniques, where the pedestrian lane detection task is cast as a two-class semantic segmentation problem (0: background, 1: pedestrian lane). A recent survey of pedestrian lane detection methods demonstrates that the DL-based segmentation methods can detect pedestrian lanes accurately [21]. However, they still lack practicality due to non-optimal accuracy, speed, and model size trade-off. Based on the survey, the most accurate DL method is a large DNN, named Multiscale HRNet [27, 38], and the fastest DL method is a DNN automatically designed by a neural architecture search method, called Fast-NAS [4].

The Multiscale HRNet is accurate (mean intersection over union of 0.9568), but it has a slow processing time (2.6 frames per second). In contrast, the DNN derived by Fast-NAS is fast (142.86 frames per second), but it is slightly inaccurate (mean intersection over union of 0.9229). Hence, a suitable DNN for real-time pedestrian lane detection is still needed.

Designing a suitable DNN manually for pedestrian lane detection is challenging. There are many design considerations, including the number of layers, the connection between layers, and the operation at every layer. A poorly designed network will lead to low performance. Additionally, the DNN has to be accurate, fast, and compact for practicality. To this end, NAS is a promising approach because it can find a suitable network automatically based on some criteria, thus reducing architecture engineering effort significantly.

This paper proposes a new NAS algorithm, named Multi-scale Dense NAS (MSD-NAS), that can automatically design a fast, accurate, and compact DNN for the pedestrian lane detection task. The search space of MSD-NAS is a large and dense search space that also contains many existing hand-crafted DNNs as candidates. Additionally, the network designed by MSD-NAS supports multi-scale inputs, allowing it to utilize both local and global contexts for predictions. To further improve the detection performance, we introduce a novel Short-term Visual Memory mechanism to improve information facilitation in the derived network.

The contributions of this paper can be highlighted as follows:

1. We propose a new neural architecture search algorithm, called MSD-NAS, to automatically find the optimum DNN with multi-scale input branches for pedestrian lane segmentation. The capability of MSD-NAS is demonstrated via extensive analysis and experiments.
2. We introduce a novel NAS search space that is generic and large. The search space is represented as a Generalized Segmentation Network (GSN). GSN has multi-scale input branches, allowing the search algorithm to select the best input scale. In fact, many state-of-the-art hand-crafted DNNs for image segmentation are special cases of the GSN.
3. We propose a new Short-term Visual Memory (STVM) mechanism for the derived network of MSD-NAS. It helps information-sharing within the derived network. Our experiments show that the STVM mechanism further improves the segmentation accuracy.

The remainder of this paper is organized as follows. Section 2 reviews the related work, Section 3 describes the proposed neural architecture search method, Section 4 presents the experimental results and analysis, and Section 5 concludes our work.

2 Related work

This section first reviews the existing unmarked lane detection algorithms (Section 2.1). To justify our NAS search space, we also discuss the state-of-the-art DL-based image segmentation methods (Section 2.2).

2.1 Unmarked lane detection

Several methods have been proposed for detecting unmarked lanes. In general, there are three main approaches: traditional-based lane segmentation, lane-border detection, and deep learning-based segmentation. *Traditional-based lane segmentation* involves using color models that have been pre-trained to classify each pixel as either the lane class or the background class [31, 35, 37]. The methods used for this approach vary based on the color space and classifier employed. For instance, Tan et al. [37] used color histograms in the RGB space, while Sotelo et al. [35] classified pixels using the hue-saturation-intensity space. Ramstrom and Christensen [31] used feature maps from both RGB and YUV color spaces to build Gaussian mixture models for classification. As these methods are trained offline, they have limitations when dealing with variations in lane surface, such as changes in color, texture, and shape.

To overcome the limitations of the previous methods, some methods directly model the lane pixels by selecting sample regions from the input image [3, 26, 29]. These methods vary in the way they choose the sample regions. For example, Miksik et al. [26] initialized the sample lane region as a trapezoid centered at the bottom of the image and then refined the region using the vanishing point. In contrast, Alvarez and Lopez [3] randomly selected small areas at the bottom of the input image, assuming that these areas are the road surface. However, these methods are sensitive to the sample regions' quality and may require domain expertise.

Lane-border detection identifies the lane boundaries by utilizing the vanishing point [19, 32] or templates of the lane boundaries [12]. Kong et al. [19] detected the lane borders by examining the edges pointing to the vanishing point, while Crisman and Thorpe [12] identified the lane boundaries from the edges of homogeneous color regions by matching the lane templates. These lane-border detection methods can be sensitive to background edges and the accuracy of vanishing point estimation. To tackle this issue, Chang et al. [6] combined the lane-border detection from the vanishing point and the lane segmentation by the color model approaches. Another method, proposed by Phung et al. [30], used the vanishing point to construct the sample lane region. The final lane region was determined by using a color model trained from the sample lane region, and the matching scores between the edges of the homogeneous color regions and the lane templates. Both the traditional-based lane segmentation and

lane-border detection approaches rely solely on image processing techniques, which are slow and not robust.

Deep learning-based segmentation detects the pedestrian lane from a scene by performing pixel-wise classification, i.e., semantic segmentation, using a deep neural network. Nguyen et al. [28] combined a Gaussian process classifier with a DNN. Thanh et al. [39] proposed a Gabor DNN, which uses variational Bayesian inference for semantic segmentation of pedestrian lanes. Both approaches perform uncertainty estimation to improve the reliability of the segmentation results, however, this comes at the cost of additional processing time. An alternative method proposed by Ang et al. [4] used neural architecture search to find a faster DNN for pedestrian lane segmentation. Still, the search space does not cover the most advanced segmentation networks, which will be discussed in the next section.

2.2 Deep learning-based semantic segmentation

The key difference among the different DL-based segmentation methods is the network architectural design. Many attributes of the existing architectures overlap, hence we group them based on their key novelty.

Fully convolutional network. Most of the current deep learning methods for image segmentation adopt a fully convolutional network (FCN) because of its efficiency [2, 36]. With a single forward-pass, FCN can generate the output segmentation map of the same size as the input image. The idea of using an FCN for image segmentation is first introduced by Long et al. [25]. They converted a standard convolutional neural network (CNN) into an FCN by replacing all fully-connected layers with convolutional layers.

Encoder-decoder framework. The recent FCN models follow a more systematic structure: the encoder-decoder framework. The encoder extracts salient features from the input images, and the decoder generates the output segmentation maps from the extracted features. For the encoder, many authors adopted a top-performing convolutional neural network [7, 22]. For the decoder, some authors designed their own decoder network [11], while other authors used a mirrored design of their encoder network [5].

Skip-connection. In the encoder-decoder framework, the feature maps transform from high-resolution to low-resolution (encoding stage), and then from low-resolution to high-resolution again (decoding stage). Hence, the fine-grained information may be lost while encoding and fail to be recovered while decoding. To overcome this problem, Ronneberger et al. [33] used skip-connections to transfer the high-resolution feature maps from the encoder to the decoder. Additionally, Long et al. [25] found that combining the feature maps of the earlier layers and the penultimate layer via skip-connections can improve segmentation performance.

Multi-scale processing. Both local and global contextual information is useful for accurate segmentation outputs. This can be exploited by processing feature maps at different scales. Lin et al. [23] proposed the feature pyramid network (FPN) for object detection and later extended it to image segmentation. Since different depths in the decoder process feature maps of different scales, FPN exploits this pyramidal characteristic by making predictions at every depth. Zhao et al. [43] developed the Pyramid Scene Parsing Network (PSPNet), which contains a pyramid pooling module that uses pooling operations of different sizes to downsample the input feature map into different scales. Wang et al. [41] proposed HRNet, which has multiple resolution streams that exchange feature maps in parallel.

Dilated convolution. Dilated convolution (also known as atrous convolution) manipulates the receptive field by using a sparse kernel. For example, a 3×3 convolution with a dilation rate of 2 can be visualized as a 5×5 convolution kernel with every second row and column emptied. Hence, the receptive field can be increased without incurring additional parameters.

A popular DL-based segmentation model that uses dilated convolution is the DeepLab family, namely DeepLabv1 [8], DeepLabv2 [9], DeepLabv3 [10], and DeepLabv3+ [11]. DeepLabv1 replaces the last few convolutional layers in an FCN with atrous convolutional layers to maintain higher resolution feature maps. Subsequently, DeepLabv2 introduces the atrous spatial pyramid pooling (ASPP) module. The ASPP module uses several atrous convolutional layers with different dilation rates in parallel, effectively processing the feature maps at multiple scales. DeepLabv3 comprises modules with atrous convolution that are placed in a cascade pattern. The latest version, DeepLabv3+, adopts the encoder-decoder framework.

As discussed above, there are many network architectures with different attributes, each with its own strengths. Inspired by these studies, we incorporate these design elements into our NAS search space. The network candidates in our search space are fully convolutional, have access to high-resolution feature maps, can utilize multi-contextual information, and can perform dilated convolution.

3 Methodology

We introduce a new neural architecture search algorithm to design the best DNN for pedestrian lane segmentation. The proposed method, named MSD-NAS, can design a network architecture with multiple input branches. Therefore, the derived network can utilize multi-scale information effectively. MSD-NAS finds the optimum architecture from a dense search space, called the Generalized Segmentation Network. Additionally, we propose a novel Short-term Visual

Memory mechanism to better facilitate information sharing within the derived network.

This section is organized as follows. Section 3.1 introduces the Generalized Segmentation Network. Section 3.2 describes the architectural parameters of the GSN. Section 3.3 explains the algorithm to optimize the GSN, and Section 3.4 shows the procedure to derive the optimum network from the optimized GSN. Lastly, Section 3.5 presents the Short-term Visual Memory mechanism.

3.1 Generalized segmentation network

The Generalized Segmentation Network is a large DNN that is represented by a group of nodes and edges. Each node performs an operation (e.g., a 3×3 convolution operation or an identity operation), and each edge represents the information flow between two nodes. The nodes are organized into a two-dimensional grid, where the horizontal axis represents the processing layer, and the vertical axis represents the scale, see Fig. 1.

The GSN has an input layer, an output layer, and L processing layers. Processing is done at multiple image scales: $0, 1, \ldots, S$. At image scale s, the input layer downsamples the input image by a factor of 2^s, before passing it to the processing nodes in Layer 1. Each processing node at Scales 1 to (S-1) will produce feature maps of three different scales, while, the nodes at Scales 0 and S will produce feature maps of two different scales. Let $H \times W$ be the size of the input image. At scale s, each node produces feature maps with a spatial size of $\frac{H}{2^{s-1}} \times \frac{W}{2^{s-1}}$, $\frac{H}{2^s} \times \frac{W}{2^s}$, and $\frac{H}{2^{s+1}} \times \frac{W}{2^{s+1}}$. For the receiving node at scale z, the channel size of the feature maps is $\mathcal{C} \times 2^z$, where $\mathcal{C}$ is a user-defined hyperparameter.

A processing node (l, s) at layer l and scale s can receive feature maps from nodes at layer $(l-1)$ and three adjacent scales $(s-1, s, s+1)$. This is in sharp contrast with many existing networks, where each node receives inputs from only one adjacent scale (except for the ad-hoc skip connections). At the output layer, a convolution along the third dimension of the feature map (i.e., 1×1 convolution) is performed to predict a segmentation map of size $\frac{H}{2^s} \times \frac{W}{2^s}$ pixels for scale s. To generate the final output, we convert the segmentation map at a selected scale to the same size as the input image via bilinear interpolation.

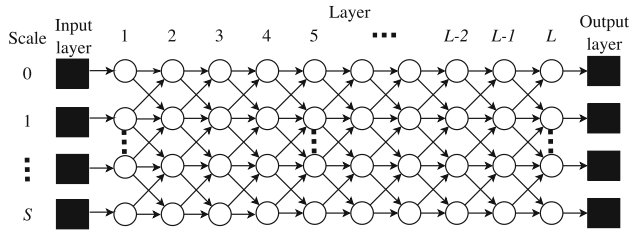


Fig. 1 The proposed Generalized Segmentation Network for image segmentation

The proposed GSN has a *generalized* architecture in that not all processing nodes and edges are activated, and typically only a few nodes in the input layer and the output layer are necessary. Fig. 2 shows that many high-performing DNN architectures for image segmentation can be considered as special cases of the GSN.

3.2 Architectural parameters of the GSN

GSN can be considered a parent model containing all possible candidate operations and paths. From the GSN, our goal is to find the optimum child network. To achieve this, we use the differentiable architecture search (DARTS) [24], which is explained next.

We incorporate three types of architectural parameters into the GSN that control the relative importance of i) the input image at each scale, ii) the operations at each node, and iii) the paths to nodes in the next layer. *First*, the GSN has S

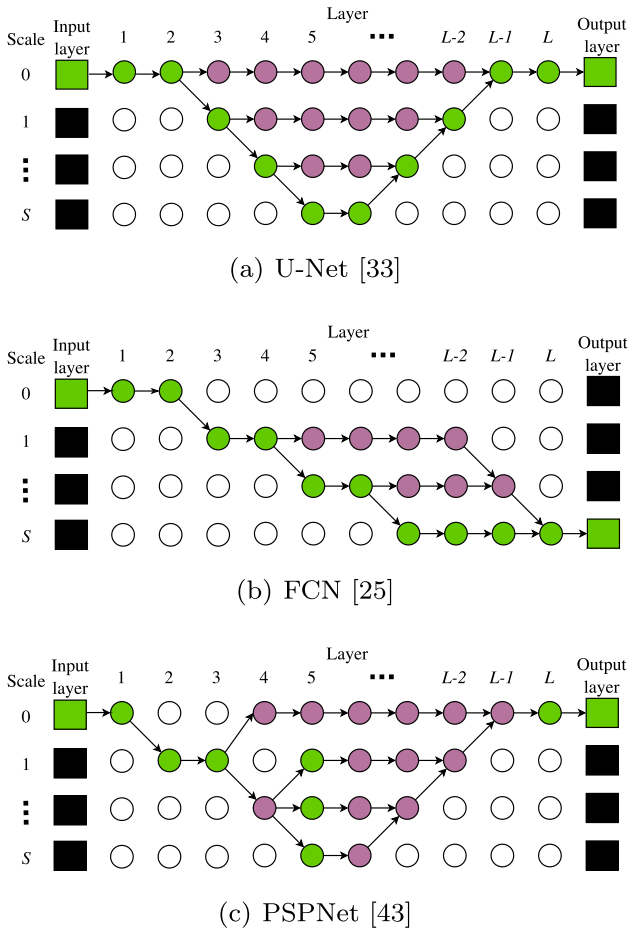


Fig. 2 Many hand-engineered state-of-the-art deep networks for image segmentation are special cases of the proposed GSN (best viewed in color). *Green node*: filtering operation. *Purple node*: identity operation

input nodes, one for each image scale. The input node $(0, s)$ at scale s will downsample the original input image size by a factor of $\frac{1}{2^s}$ before sending it to the next layer. Each input node $(0, s)$ is associated with an architectural parameter γ_s that determines the importance of the input at scale s. Hence, the input to node $(1, s)$ is defined as

$$I_{1,s} = \gamma_s \, f_s(X), \tag{1}$$

where X is the input image, and f_s is the downsampling operation by a factor of $\frac{1}{2^s}$. The parameters $\boldsymbol{\gamma} = \{\gamma_0, \gamma_1, \ldots, \gamma_S\}$ are normalized with *softmax* function to represent the importance probabilities.

Second, each node now computes a mixed operation, which is a weighted sum of multiple single operations. Let $I_{l,s}$ be the input of node (l, s). Here, $I_{l,s}$ is the sum of all the feature maps received from the connected nodes in the previous layer. Let $\mathcal{O} = \{O^1, O^2, \ldots\}$ be the candidate operations. At node (l, s), the *intermediate* feature map is computed as

$$\mathcal{F}_{l,s} = \sum_{j}^{|\mathcal{O}|} \alpha_{l,s}^{j} \, O^j(I_{l,s}). \tag{2}$$

Here, the architectural parameters $\boldsymbol{\alpha}_{l,s} = \{\alpha_{l,s}^1, \ldots, \alpha_{l,s}^{|\mathcal{O}|}\}$ denote the importance of each operation. The parameters $\boldsymbol{\alpha}_{l,s}$ are also normalized with *softmax* function to represent the importance probabilities.

Third, each node at Scales 1 to (S-1) will produce three feature maps of various scales. At node (l, s), the *output* feature maps are defined as

$$\begin{aligned} y_{l,s} &= \beta_{l,s} \, \mathcal{F}_{l,s}, \\ y_{l,s}^{+} &= \beta_{l,s}^{+} \, f^{+}(\mathcal{F}_{l,s}), \\ y_{l,s}^{-} &= \beta_{l,s}^{-} \, f^{-}(\mathcal{F}_{l,s}), \end{aligned} \tag{3}$$

where f^+ and f^- are the functions that upsample and downsample the feature map's size by 2, respectively. Note that the nodes at Scale 0 will not produce $y_{l,s}^{+}$ and the nodes at Scale S will not produce $y_{l,s}^{-}$. Here, the architectural parameters $\boldsymbol{\beta}_{l,s} = \{\beta_{l,s}, \beta_{l,s}^{+}, \beta_{l,s}^{-}\}$ denote the importance of each path. The parameters $\boldsymbol{\beta}_{l,s}$ are also normalized with *softmax* function to represent the importance probabilities.

The architectural parameters $\boldsymbol{\gamma}$, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ can be optimized using gradient descent since they are in a continuous space. However, the memory overhead for computing all the mixed operations is large as each node now consists of $|\mathcal{O}|$ candidate operations. To overcome this problem, we only compute the mixed operation using k input channels of the input feature map $I_{l,s}$, thereby reducing the memory overhead by k times. This method is known as the partially-connected DARTS [42]. Inspired by partially-connected DARTS, we modify Eq. (2) to

$$\mathcal{F}_{l,s} = \sum_{j}^{|\mathcal{O}|} \alpha_{l,s}^{j} \, O^j(B_{l,s} \times I_{l,s}) + (1 - B_{l,s}) \, I_{l,s}, \tag{4}$$

where $B_{l,s}$ is the sampled channel mask.

3.3 Optimizing the GSN

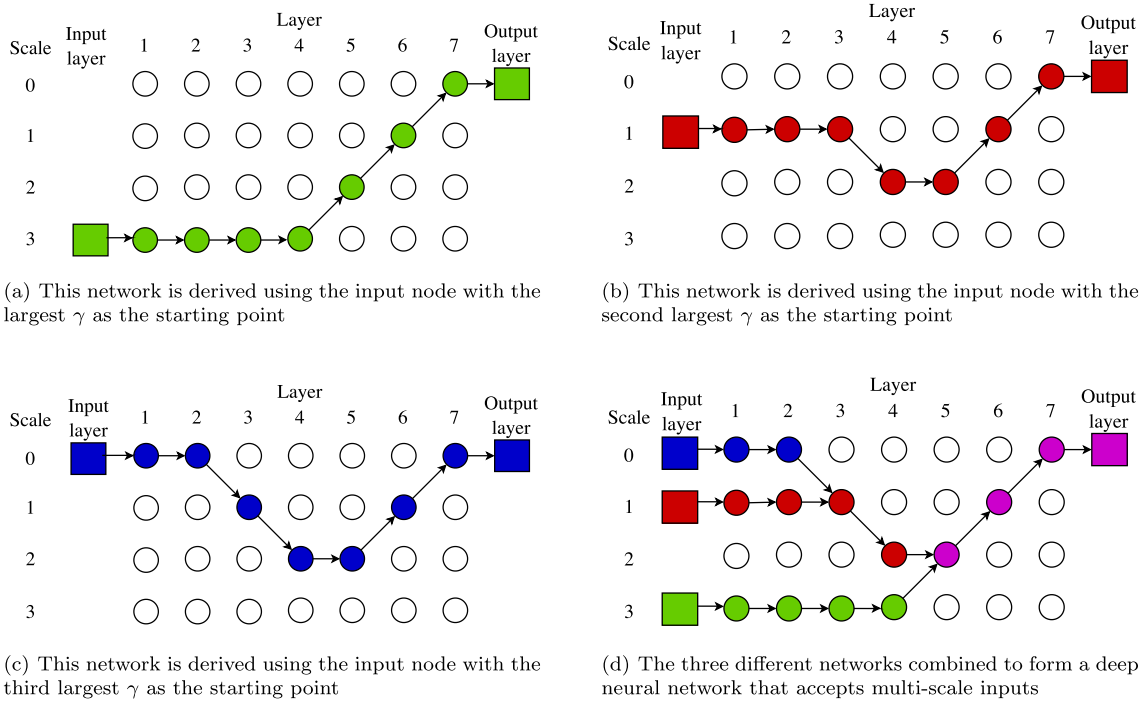
The final network is determined by the architectural parameters $\boldsymbol{\gamma}$, $\boldsymbol{\alpha}$, and $\boldsymbol{\beta}$. During the training phase of the GSN, the architectural parameters $\boldsymbol{\gamma}$, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and network weights $\boldsymbol{w}$ are optimized alternately using gradient descent. The optimization procedure is described as follows.

The training set is split into two equal subsets, A and B. For each training epoch, first, the network weights $\boldsymbol{w}$ are updated with the training loss $\mathcal{L}_{\mathrm{A}}$, which is computed on the training subset A. Then, the architectural parameters $\boldsymbol{\gamma}$, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ are updated with the training loss $\mathcal{L}_{\mathrm{B}}$, which is computed on the training subset B. The architectural parameters and network weights are optimized alternately and repeatedly until convergence. Note that the network weights $\boldsymbol{w}$ are pre-trained for n epochs before begin optimizing the architectural parameters $\boldsymbol{\gamma}$, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ to avoid local optima.

3.4 Deriving the final DNN from the optimized GSN

MSD-NAS can derive K unique networks ($K \leq S$), where each network processes the input image at a different scale. After the architectural parameters are optimized, we derive the networks as follows. We sort the $\boldsymbol{\gamma} = \{\gamma_0, \gamma_1, \ldots, \gamma_S\}$ in descending order. The input node with the largest γ determines the first selected node in Layer 1. At the first selected node, the output path with the largest β and the operation with the largest α are selected. We repeat this process for every active node until Layer L. For prediction, only the output node connected to the last active node is used. We repeat the above steps using the input node with the next largest γ until K unique networks are obtained, see Figs. 3a-c for some illustrations.

After the K unique networks are derived, we combine them into one final deep neural network with multiple input branches, with each branch handling the input image at a different scale. The networks will share the nodes if they have common segments, see Fig. 3d for an illustration. If the combined network has more than one output node, we only use the output node that predicts at the smallest scale (closest to the original image resolution). Note that we will train the final network from scratch using the full training set.



(a) This network is derived using the input node with the largest γ as the starting point

(b) This network is derived using the input node with the second largest γ as the starting point

(c) This network is derived using the input node with the third largest γ as the starting point

(d) The three different networks combined to form a deep neural network that accepts multi-scale inputs

Fig. 3 An illustration of the network derivation procedure when $K = 3$

3.5 Short-term visual memory mechanism

Several studies have shown that the skip-connection scheme has many benefits [14, 25, 33]. For segmentation neural networks, skip-connections can improve performance by transferring the high-resolution feature maps from the shallow to deep layers [25]. For very large networks, skip-connections can reduce the effects of vanishing gradients [14, 33].

There are three main challenges in using the skip-connection scheme with MSD-NAS. First, the skip-connection scheme is not efficient for MSD-NAS. The network derived by MSD-NAS can consist of multiple input branches; these input branches are processed sequentially, i.e., not in parallel. Hence, we need to store the intermediate feature maps if the skip-connections are between different input branches. Moreover, the processing time will be delayed if a node relies on the feature map from another input branch.

Second, in segmentation networks, skip-connections are primarily used to transfer high-resolution feature maps to aid the upsampling operations. Therefore, other types of nodes can not share information intra-branch and inter-branch. Third, in the skip-connection scheme, feature maps are unweighted, i.e., all pixels are treated as equally important. It is beneficial to let the network learn the importance of each pixel to the overall segmentation performance.

To overcome these problems, we propose the Short-term Visual Memory (STVM) mechanism for the nodes. The phrase *short-term* arises from the fact that the memory only contains information about the current input image. Note that we only apply the STVM mechanism on the derived network, i.e., after completing the search phase. Next, we describe the STVM mechanism in detail.

In the derived network, there are at most S STVM modules. The nodes at scale s share the same STVM module $\boldsymbol{m}_s$. Each STVM module $\boldsymbol{m}_s$ has the same dimensions as the output feature map of a node at scale s. Note that a feature map is a 3-D matrix, where each channel is the 2-D output of a convolution filter. Hence, the STVM module $\boldsymbol{m}_s$ can be represented as $[m_s^1, m_s^2, \ldots]$. Each node interacts with its STVM module through input and update gates. The input gate determines what information to use from the STVM module, and the update gate inserts new information into the STVM module.

Here, we describe the input gate in detail. At node (l, s), we perform a channel-wise 2-D convolution on the STVM module $\boldsymbol{m}_s$ to obtain the matrix $\boldsymbol{a}_{l,s} = [a_{l,s}^1, a_{l,s}^2, \ldots]$. Each element $a_{l,s}^j$ is computed as

$$a_{l,s}^j = \mathcal{I}_{l,s}^j * m_s^j, \tag{5}$$

where $*$ is the convolution operator, and $\mathcal{I}_{l,s}^j$ is a 2-D learnable weight. The input gate of node (l, s) is then defined as

$$\boldsymbol{i}_{l,s} = \sigma(\boldsymbol{a}_{l,s}), \tag{6}$$

where σ denotes the *sigmoid* activation function. Now, the input gate $\boldsymbol{i}_{l,s}$ represents the weight of every pixel in the STVM module $\boldsymbol{m}_s$. The input to node (l, s) is then given as

$$I_{l,s} = I_{l,s} + \boldsymbol{m}_s \times \boldsymbol{i}_{l,s}. \tag{7}$$

Next, we describe the update gate in detail. Let $\boldsymbol{\mathcal{F}}_{l,s} = [\mathcal{F}^1_{l,s}, \mathcal{F}^2_{l,s}, \ldots]$ be the output of node (l, s). We perform a channel-wise 2-D convolution on the output $\boldsymbol{\mathcal{F}}_{\boldsymbol{l,s}}$ to obtain the matrix $\boldsymbol{b}_{l,s} = [b^1_{l,s}, b^2_{l,s}, \ldots]$. The element $b^j_{l,s}$ is computed as

$$b^j_{l,s} = \mathcal{U}^j_{l,s} * \mathcal{F}^j_{l,s}, \tag{8}$$

where $\mathcal{U}^j_{l,s}$ is a 2-D learnable weight. The update gate of node (l, s) is then defined as

$$\boldsymbol{u}_{l,s} = \sigma(\boldsymbol{b}_{l,s}). \tag{9}$$

Now, the update gate $\boldsymbol{u}_{l,s}$ represents the weight of every pixel in the output $\boldsymbol{\mathcal{F}}_{\boldsymbol{l,s}}$. We update the STVM module as follows:

$$\boldsymbol{m}_s = \boldsymbol{u}_{l,s} \times \boldsymbol{\mathcal{F}}_{l,s} + (1 - \boldsymbol{u}_{l,s}) \times \boldsymbol{m}_s. \tag{10}$$

The interaction between a node and its STVM module is illustrated in Fig. 4.

4 Experiments and analysis

In this section, we present the experiments and analysis of MSD-NAS. Section 4.1 describes the pedestrian lane dataset. Section 4.2 presents the experimental steps, and Section 4.3 describes the search configurations. Section 4.4 analyzes the proposed MSD-NAS method, and Section 4.5 compares MSD-NAS with other hand-crafted deep learning models on the pedestrian lane segmentation task.

4.1 Pedestrian lane dataset

In this paper, we conducted the experiments using the Pedestrian Lane Detection and Vanishing Point Estimation Version 3.0 (PLVP3) dataset [30]. The PLVP3 dataset comprises 10,000 color images with their corresponding ground-truth annotations. The ground-truth masks were manually annotated, where every pixel is labeled as pedestrian-lane (1) or background (0) classes.

The images in PLVP3 were acquired from real indoor and outdoor scenes in various weather conditions and at different times of the day. The pedestrian paths in these images are diverse in shapes, colors, and textures. The cameras used to acquire these images are also different, resulting in images with varying widths and heights (ranging from 1224 to 1632 pixels). The overall statistics of this dataset are given in Table 1. Several images and their ground-truth masks from the PLVP3 dataset are shown in Fig. 5. The PLVP3 dataset can be downloaded from http://documents.uow.edu.au/~phung/plvp3.html.

4.2 Experimental steps

The pedestrian lane detection methods were evaluated using accuracy, mean intersection over union, and frames per second metrics. *Accuracy* is the percentage of image pixels that are correctly classified. *Mean intersection over union* (mIoU) is the average IoU score over each class. IoU is defined as the area of the intersection divided by the area of the union between the predicted output and the ground-truth mask: $\text{IoU} = \frac{\text{Area of intersection}}{\text{Area of union}}$. *Frames per second* (FPS) is the number of predictions that a given method can produce in a second. The FPS was measured using a system that has a 2.4 GHz Intel Xero Gold 5115 CPU and a 12 GB NVIDIA GeForce GTX Titan Xp GPU.

The experiments were conducted using 5-fold cross-validation. The PLVP3 dataset was divided into five partitions of equal sizes. For each fold, one partition was used as the test set, and the remaining partitions were used as the train-

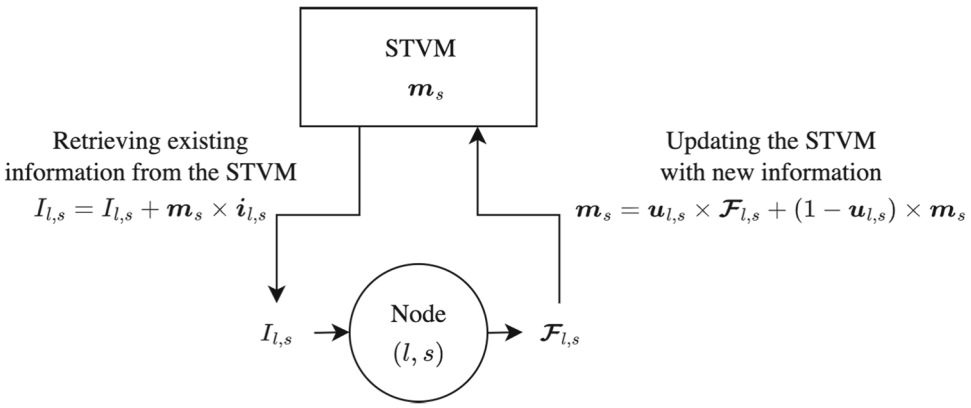


Fig. 4 The interaction between a node (l, s) and its STVM module $\boldsymbol{m}_s$

Table 1 Statistics of the PLVP3 dataset

Condition	Description	Number of images
Surfaces	Brick (outdoor)	2,917
	Concrete (outdoor)	4,860
	Pavement (outdoor)	1,164
	Indoor	734
	Other (mixed indoor/outdoor)	325
Lighting	Normal	7,845
	Shadows and extreme	2,155

ing set. This process was repeated five times for different choices of the test set. Note that each training set was further divided into 90% images for training and 10% images for validation. The images were resized to 320 × 320 pixels for the experiments.

4.3 Search settings

The search was conducted using a GSN with 14 layers ($L = 14$) and 6 scales ($S = 6$). We ran the search algorithm for 30 epochs. For each node, the candidate operations $\mathcal{O}$ consisted of:

1. Identity.
2. 3×3 convolution (conv3).
3. 3×3 convolution $\times 2$ (conv3t2).
4. 3×3 convolution with a dilation of 2 (conv3d2).
5. 5×5 convolution (conv5).
6. 3×3 depthwise separable convolution (dconv3).
7. 3×3 depthwise separable convolution $\times 2$ (dconv3t2).
8. 5×5 depthwise separable convolution (dconv5).
9. 3×3 convolution with residual connection (res3).
10. 3×3 convolution $\times 2$ with residual connection (res3t2).

The downsampling function f^- was implemented as a conv3 operation with a stride of 2. The upsampling function f^+ was implemented as a bilinear upsampling operation with a scale of 2, followed by a conv3 operation. The downsampling function f_s was implemented as a conv3 operation with a stride of 2^s. Every convolution operation was followed by a batch normalization operation and a ReLU activation function.

The search phase of the proposed NAS method was conducted using two optimizers. The Adam optimizer was used to update the architectural parameters $\boldsymbol{\gamma}, \boldsymbol{\alpha}, \boldsymbol{\beta}$ with the following settings: learning rate of 0.0003, weight decay of 0.0005, and exponential decay rates β_1 of 0.9 and β_2 of 0.999. The SGD optimizer with momentum was used to update the network weights $\boldsymbol{w}$ with the following settings: learning rate of 0.01, momentum of 0.9, and weight decay rate of 0.0005.

The optimization of the architectural parameters $\boldsymbol{\gamma}, \boldsymbol{\alpha}, \boldsymbol{\beta}$ only begun after the network weights $\boldsymbol{w}$ were trained for 20 epochs ($n = 20$). With these configurations, the search running on a 12GB NVIDIA GTX Titan Xp GPU took roughly 20 hours using a GSN with a base channel size of 8 ($\mathcal{C} = 8$), and roughly 37 hours using a GSN with a base channel size of 16 ($\mathcal{C} = 16$).

4.4 Ablation study

In this section, we performed an ablation study to:

- determine the optimum number of search epochs,
- find the best number of layers and scales for the GSN,
- analyze the effects of using different numbers of input branches K,
- determine the effectiveness of the STVM mechanism, and
- compare the different values of base channel size $\mathcal{C}$.

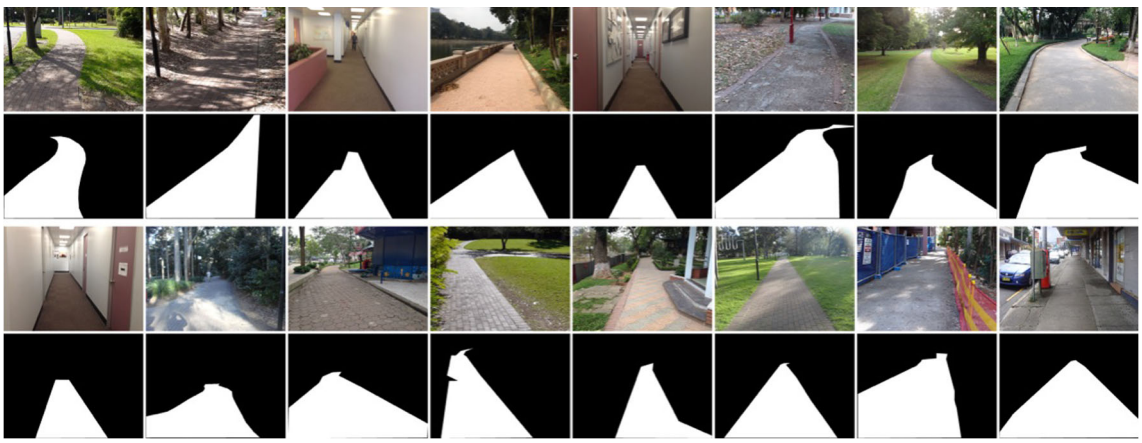


Fig. 5 Several examples from the PLVP3 dataset

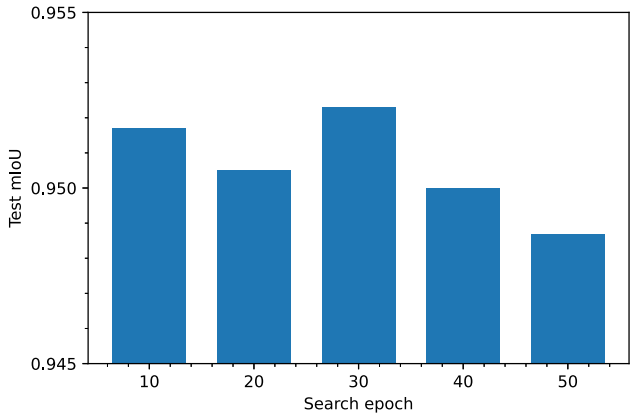


Fig. 6 The test mIoU of the networks found by searching for different numbers of epochs

This ablation study was conducted using fold-1 of the dataset.

Optimum number of search epochs. We ran the search five times, and each time with a different number of search epochs: i) 10 epochs; ii) 20 epochs; iii) 30 epochs; iv) 40 epochs; and v) 50 epochs. The higher the number of epochs, the longer the search will take. The derived network from each search was trained from scratch, and then the mIoU score was computed for the test set. Fig. 6 shows the results of this comparison. The network found by searching for 50 epochs achieved the lowest mIoU. We believe this is due to overfitting. The network found by searching for 30 epochs achieved the highest mIoU; the search time was also the average among all other configurations. Hence, we chose to search for 30 epochs in this paper.

Optimum number of layers and scales for the GSN. We performed a 3×3 grid search with the following choices: 12, 14, and 16 layers; and 4, 5, and 6 scales. Fig. 7 presents the results of the grid search. For the number of scales, the mIoU increased as the number of scales increased, except for

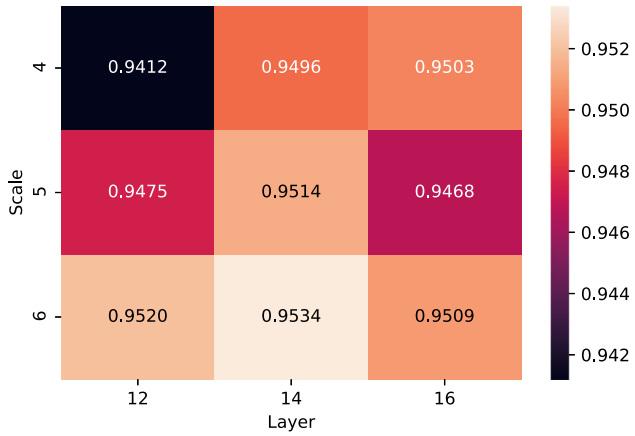


Fig. 7 Results of the grid search for finding the optimum number of GSN layers and scales. *Value inside the box*: test mIoU

Table 2 The effects of using different numbers of input branches

No. input branches	mIoU	FPS	Trainable params. (M)
One branch ($K = 1$)	0.9423	142.86	5.754
Three branches ($K = 3$)	0.9533	76.92	8.690
Six branches ($K = 6$)	0.9534	58.82	10.998

configurations with 16 layers. For the number of layers, the mIoU increased as the number of layers increased. However, it stopped improving after 14 layers, except on configurations with 4 scales. For the smallest GSN (12 layers and 4 scales), the derived network obtained the lowest mIoU. Among all the tested combinations, the network derived from the GSN with 14 layers and 6 scales achieved the highest mIoU. Therefore, we adopted 14 layers and 6 scales for the GSN in this paper.

Effects of using different numbers of input branches K. We tested three configurations on the same derived network: i) one input branch ($K = 1$); ii) three input branches ($K = 3$); and iii) six input branches ($K = 6$). Table 2 shows the results of this experiment. For all configurations, the network's mIoU, inference time, and trainable parameters increased as the number of input branches increased. The network with six input branches obtained the highest mIoU (0.9534), whereas the network with one input branch had the lowest mIoU (0.9423). We adopted the configuration with six input branches ($K = 6$) because it achieved the best performance, while still being able to support real-time performance.

Effectiveness of the STVM mechanism. We tested two settings on the same derived network: i) with STVM; and ii) without STVM. Table 3 presents the results of this study. The derived network with STVM (mIoU of 0.9534) outperformed the network without STVM (mIoU of 0.9462). This improvement of 0.0072 in mIoU by including STVM is significant because as seen in Table 5, the top and bottom of the 13 evaluated methods ($\leq$ 27M parameters) differ only 0.0004 in mIoU. However, the network with STVM had more parameters than the network without STVM (10.998M versus 10.922M). This slight increment of parameters is expected because the STVM mechanism requires additional computations in every node, i.e., the input and output gates.

Fig. 8 illustrates the STVM module at scales 1, 3, and 5. For simplicity, we only visualize the channel with the highest activation. The figure shows that the STVM mechanism

Table 3 The effects of using the STVM mechanism in the derived network

Setting	mIoU	Trainable parameters (M)
With STVM	0.9534	10.998
Without STVM	0.9462	10.922

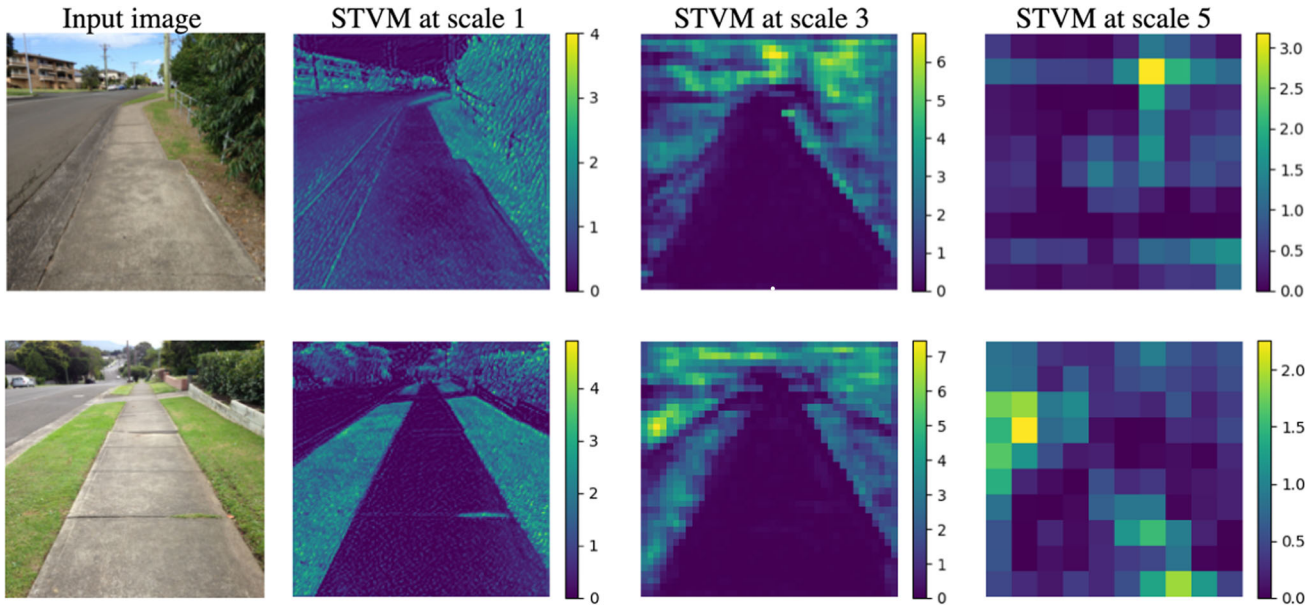


Fig. 8 A visualization of STVM at various scales (1, 3, and 5). At each scale, we only show the STVM's channel with the highest activation. This figure is best viewed in color

stores different salient information about the input image at different scales. At scale 1, high activation values were concentrated on the grass regions. At scale 3, high activation values included the sky too. At scale 5, the memory became very coarse to be understood. These results and illustrations justify the use of the STVM mechanism in the derived network.

Effects of using different values of base channel size $\mathcal{C}$. We searched on three different GSNs, each with a different base channel size: i) $\mathcal{C} = 4$; ii) $\mathcal{C} = 8$; and $\mathcal{C} = 16$. Table 4 presents the results of this comparison. The network derived with $\mathcal{C} = 16$ (mIoU of 0.9545) had the highest mIoU; it also had the most trainable parameters (29.961M). While the network derived with $\mathcal{C} = 4$ (mIoU of 0.9503) had the lowest mIoU, it also had the least trainable parameters (1.880M). Hence, selecting the value of $\mathcal{C}$ is a trade-off between mIoU and model size. A smaller value of $\mathcal{C}$ will reduce the model size and mIoU, and vice versa. In this paper, we used both $\mathcal{C} = 8$ and $\mathcal{C} = 16$ configurations as they yield networks of different sizes.

Table 4 The comparison between the different values of base channel size $\mathcal{C}$

Base channel $\mathcal{C}$	mIoU	Trainable parameters (M)
4	0.9503	1.880
8	0.9534	10.998
16	0.9545	29.961

4.5 Comparison with the existing pedestrian lane detection methods

In this section, we compared MSD-NAS with *22* existing pedestrian lane detection methods using 5-fold cross-validation, which included 2 traditional methods, 1 NAS method, and 19 hand-designed models. The evaluated models were grouped into two sizes: i) methods with $\leq$ 27M trainable parameters, and ii) methods with $>$ 27M trainable parameters.

For a robust evaluation, we ran the proposed search algorithm on every fold. This resulted in five unique deep networks designed by MSD-NAS. We then trained these networks from scratch and tested their performances on their respective test sets. The performance of the MSD-NAS is defined as the test results averaged from these five networks. To search for two network sizes that fit the two groups, we set the $\mathcal{C} = 8$ for a smaller network and $\mathcal{C} = 16$ for a larger network.

Table 5 presents the results of this analysis. In the group with $\leq$ 27M trainable parameters, MSD-NAS ($\mathcal{C} = 8$) achieved the highest accuracy (0.9769) and mIoU (0.9517). Compared with the fastest method, PSPNet (ResNet-34), MSD-NAS ($\mathcal{C} = 8$) was 4.25 times slower (58.82 versus 250.00). However, MSD-NAS ($\mathcal{C} = 8$) had 2.61 times fewer trainable parameters (8.228M versus 21.443M).

Compared with the smallest model, BGN, MSD-NAS ($\mathcal{C} = 8$) was 27.43 times larger (8.228M versus 0.300M). However, MSD-NAS ($\mathcal{C} = 8$) outperformed BGN significantly in terms of accuracy (0.9769 versus 0.9734) and mIoU (0.9517 versus 0.9492). Compared with the current

Table 5 The mean performance of different lane segmentation methods over five folds. The best metrics in each group of methods were given in bold

Network size	Method	Accuracy↑	mIoU↑	FPS↑	Trainable params. (M)↓
> 27M trainable parameters	Multiscale HRNet-OCR [27, 38]	**0.9792**	**0.9568**	2.61	72.100
	PSPNet (ResNet-101) [43]	0.9776	0.9540	30.30	68.059
	DeepLabV3+ (ResNet-101) [11]	0.9772*	0.9523*	47.62	59.339
	Bayesian DeepLabv3+ [13]	0.9737*	0.9486*	37.88	54.750
	DeepLabV3+ (Xception) [11]	0.9764*	0.9508*	41.67	54.700
	DeepLabV3 (ResNet-50) [10]	0.9780	0.9539	41.67	39.634
	Hybrid DL-GP [28]	0.9640*	0.9262*	1.01	29.469
	Bayesian SegNet [17]	0.9681*	0.9344*	16.13	29.443
	SegNet [5]	0.9645*	0.9258*	14.49	29.443
	Our MSD-NAS ($\mathcal{C} = 16$)	0.9781	0.9542	**52.63**	**28.175**
≤ 27M trainable parameters	Traditional combined method [30]	0.8964*	0.7569*	0.76	–
	Traditional border-based method [19]	0.7923*	0.6312*	0.01	–
	U-Net (ResNet-34) [33]	0.9754*	0.9488*	142.86	24.437
	FPN (ResNet-34) [18]	0.9758*	0.9497*	142.86	23.156
	DeepLabV3+ (ResNet-34) [11]	0.9765	0.9509	125.00	22.438
	LinkNet (ResNet-34) [7]	0.9760*	0.9499*	142.86	21.772
	PAN (ResNet-34) [22]	0.9767	0.9513	125.00	21.476
	PSPNet (ResNet-34) [43]	0.9768	0.9512	**250.00**	21.443
	FCN (VGG-16) [25]	0.9720*	0.9488*	27.78	14.720
	HRNetv2-W32 [41]	0.9744*	0.9467*	40.74	9.980
	UNet++ [44]	0.9757*	0.9495*	16.42	9.200
	Our MSD-NAS ($\mathcal{C} = 8$)	**0.9769**	**0.9517**	58.82	8.228
	Fast-NAS [4]	0.9700*	0.9229*	142.87	4.716
	BGN [39]	0.9734*	0.9492*	68.03	**0.300**

We also computed two-tailed tests between MSD-NAS and other methods in its group. The results are sorted based on the trainable parameters in descending order

*We reject the null hypothesis $\mathcal{H}_0 : \boldsymbol{m}_{\text{MSD-NAS}} = \boldsymbol{m}_{\text{other}}$ at a confidence level of 99.99%

That is, there is a *significant difference* compared to MSD-NAS

state-of-the-art NAS method for pedestrian lane detection, Fast-NAS, MSD-NAS ($\mathcal{C} = 8$) had significantly better accuracy (0.9769 versus 0.9700) and mIoU (0.9517 versus 0.9229). However, MSD-NAS ($\mathcal{C} = 8$) was 2.43 times slower (58.82 versus 142.87) and 1.74 times larger (8.228M versus 4.716M).

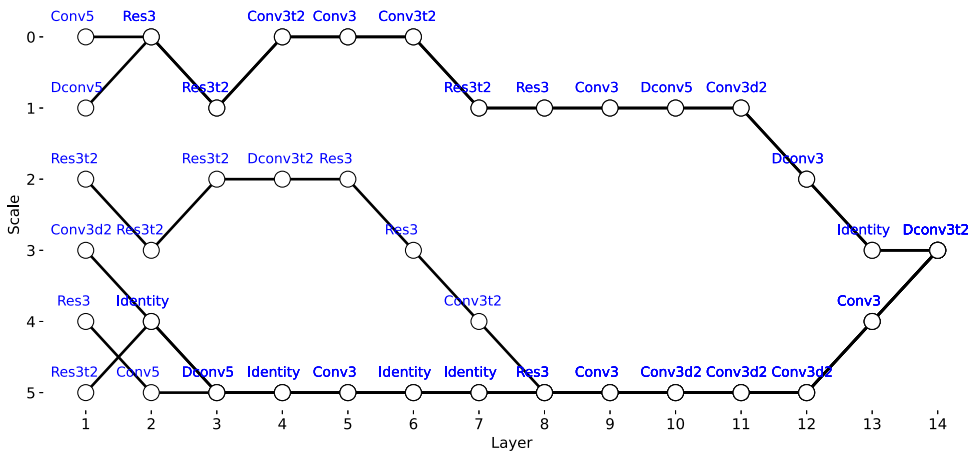


Fig. 9 The MSD-NAS ($\mathcal{C} = 16$) derived from fold-1 of the dataset. The input and output nodes are omitted for conciseness. *White circle*: node. *Blue text*: operation. *Solid black line*: data flow between nodes

In the group with > 27M trainable parameters, MSD-NAS ($\mathcal{C} = 16$) outperformed every method except Multiscale HRNet-OCR in terms of accuracy (0.9781), mIoU (0.9542), FPS (52.63), and model size (28.175). See Fig. 9 for the network architecture of MSD-NAS ($\mathcal{C} = 16$). Compared to the Multiscale HRNet-OCR, MSD-NAS ($\mathcal{C} = 16$) had slightly lower accuracy (0.9781 versus 0.9792) and mIoU (9.9542 versus 0.9568). However, the difference was statistically insignificant. In terms of processing speed and model size, MSD-NAS ($\mathcal{C} = 16$) was 20.16 times faster (52.63 versus 2.61) and 2.56 times smaller (28.175M versus 72.100M) than Multiscale HRNet-OCR.

Compared to the second fastest method in the group, DeepLabV3+ (ResNet-101), MSD-NAS ($\mathcal{C} = 16$) was 0.90 times faster (52.63 versus 47.62) and 2.11 times smaller (28.175M versus 59.339M). Furthermore, MSD-NAS ($\mathcal{C} = 16$) had a statistically significant performance advantage in terms of accuracy (0.9781 versus 0.9772) and mIoU (0.9542 versus 0.9523). Compared to the second smallest method, SegNet, MSD-NAS ($\mathcal{C} = 16$) was 1.05 times smaller (28.175M versus 29.443M) and 3.63 times faster (52.63 versus 14.49). Additionally, MSD-NAS ($\mathcal{C} = 16$) had significantly better accuracy (0.9781 versus 0.9645) and mIoU (0.9542 versus 0.9258).

Fig. 10 shows some visual results of MSD-NAS ($\mathcal{C} = 16$) and PSPNet (ResNet-101). We selected representative results that consist of different environments (indoor versus outdoor), lighting conditions (bright versus dark), and lane shapes (straight versus curve). Across these samples, we can see that the segmentation outputs of MSD-NAS were more precise than PSPNet (ResNet-101).

Several conclusions can be drawn from the results. First, MSD-NAS can automatically design DNNs that outperform the hand-designed network architectures in terms of accuracy, mIoU, speed, and model size. The MSD-NAS (58.82 FPS for $\mathcal{C} = 8$ and 52.63 FPS for $\mathcal{C} = 16$) also can support real-time pedestrian lane detection as most video cameras capture at 30 to 50 FPS.

Second, a DNN can achieve high efficiency by customizing the architecture according to the task. For example, MSD-NAS ($\mathcal{C} = 8$) outperformed U-Net (ResNet-34) despite having 2.97 times fewer parameters (8.228M versus 24.437M), and MSD-NAS ($\mathcal{C} = 16$) outperformed Multiscale HRNet-OCR despite having 2.56 times fewer parameters (28.175M versus 72.100M).

Third, the type of encoder used will affect the segmentation performance of the same architectural design. For example, PSPNet (ResNet101) outperformed PSPNet (ResNet34), and DeepLabV3+ (ResNet101) outperformed DeepLabV3+ (Xception). Hence, for every new problem, we need to consider different encoders for the same architectural design too.

Fig. 10 The visual results of MSD-NAS ($\mathcal{C} = 16$) and PSPNet (ResNet-101) on the test images (best viewed in color). The notable differences are marked with red arrows. *Column 1*: input image; *Column 2*: MSD-NAS ($\mathcal{C} = 16$); *Column 3*: PSPNet (ResNet-101)

5 Conclusion

This paper introduces a new neural architecture search algorithm, called MSD-NAS, for pedestrian lane detection. MSD-NAS can automatically design an optimized DNN with multi-scale input branches, allowing the derived network to utilize global and local contextual information for predictions. The search space of MSD-NAS is the GSN, which

is a large and dense space that also contains many existing hand-designed deep models as candidates. Hence, the search space of MSD-NAS is sufficiently large and generic. Additionally, we propose a novel Short-term Visual Memory mechanism to improve information facilitation among the nodes in the derived DNN. Our experiments demonstrate that MSD-NAS can automatically find compact and fast DNNs that outperform 22 existing methods in some or all performance metrics (accuracy, mIoU, processing speed, and model size). Notably, the DNN found by MSD-NAS is 20.16 times faster and 2.56 times smaller than the existing best deep learning model, Multiscale HRNet-OCR, while having a statistical insignificance difference in accuracy and mIoU. This paper demonstrates a concrete step toward a practical system for the assistive navigation of blind people.

Although MSD-NAS can achieve high performance in the pedestrian lane detection task, several research directions can be further explored in future work. The first research direction is optimizing the search time of MSD-NAS. Currently, the search process of MSD-NAS is split into two phases: optimizing the GSN and the derived network. The optimized GSN weights are discarded, and the derived network is trained from scratch. Future work can explore re-using the GSN weights or combining the two phases. The second research direction is incorporating more search criteria into the loss functions. At present, MSD-NAS only focuses on obtaining the best accuracy. MSD-NAS may find more efficient DNNs by directly considering the latency or model size in the loss function.

Acknowledgements This research work is supported by the Discovery Project DP190100607 "Assistive Micro-navigation for Vision Impaired People" from the Australian Research Council. The first author is also supported by a PhD scholarship from the University of Wollongong.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

References

1. Ackland P, Resnikoff S, Bourne R (2018) World blindness and visual impairment: despite many successes, the problem is growing. Community Eye Heal. 30(100):71–73
2. Ali R, Chuah JH, Talip MSA, Mokhtar N, Shoaib MA (2021) Automatic pixel-level crack segmentation in images using fully convolutional neural network based on residual blocks and pixel local weights. Eng Appl Artif Intell 104:104391
3. Alvarez JM, Lopez AM (2011) Road detection based on illuminant invariance. IEEE Trans Intell Transp Syst 12(1):184–193
4. Ang SP, Phung SL, Bouzerdoum A, Nguyen TNA, Duong STM, Schira MM: Real-time pedestrian lane detection for assistive navigation using neural architecture search. In: Proc Int Conf Pattern Recognit, pp 1–8 (2020)
5. Badrinarayanan V, Kendall A, Cipolla R (2017) SegNet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans Pattern Anal Mach Intell 39(12):2481–2495
6. Chang CK, Siagian C, Itti L: Mobile robot monocular vision navigation based on road region and boundary estimation. In: Proc IEEE/RSJ Int Conf Intell Robot Syst, pp 1043–1050 (2012)
7. Chaurasia A, Culurciello E: LinkNet: Exploiting encoder representations for efficient semantic segmentation. In: Proc IEEE Vis Commun Image Process, pp 1–5 (2017)
8. Chen LC, Papandreou G, Kokkinos I, Murphy K, Yuille AL: Semantic image segmentation with deep convolutional nets and fully connected CRFs. In: Proc Int Conf Learn Represent, pp 1–14 (2015)
9. Chen LC, Papandreou G, Kokkinos I, Murphy K, Yuille AL (2018) DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE Trans Pattern Anal Mach Intell 40(4):834–848
10. Chen LC, Papandreou G, Schroff F, Adam H: Rethinking atrous convolution for semantic image segmentation. In: Proc European Conf Computer Vision, pp 1–14 (2017)
11. Chen LC, Zhu Y, Papandreou G, Schroff F, Adam H: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proc Eur Conf Comput Vis, pp 1–18 (2018)
12. Crisman JD, Thorpe CE: UNSCARF - A color vision system for the detection of unstructured roads. In: Proc IEEE Int Conf Robot Autom, pp 2496–2501 (1991)
13. Gustafsson FK, Danelljan M, Schon TB: Evaluating scalable Bayesian deep learning methods for robust computer vision. In: Proc IEEE/CVF Conf Computer Vision and Pattern Recognition Workshops, pp 1289–1298 (2020). https://doi.org/10.1109/CVPRW50498.2020.00167
14. He K, Zhang X, Ren S, Sun J: Deep residual learning for image recognition. In: Proc Conf Comput Vis Pattern Recognit, pp 770–778 (2016)
15. Ivanchenko V, Coughlan J, Shen H: Detecting and locating crosswalks using a camera phone. In: Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit Work, pp 1–8 (2008)
16. Jackson AJ, Wolffsohn JS (2007) Low Vision Manual. Elsevier
17. Kendall A, Badrinarayanan V, Cipolla R: Bayesian SegNet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. In: Proc Br Mach Vis Conf, pp 1–11 (2017)
18. Kirillov A, Girshick R, He K, Dollar P: Panoptic feature pyramid networks. In: Proc Conf Comput Vis Pattern Recognit, pp 6399–6408 (2019)
19. Kong H, Audibert JY, Ponce J (2010) General road detection from a single image. IEEE Trans Image Process 19(8):2211–2220
20. Le MC, Phung SL, Bouzerdoum A: Pedestrian lane detection for assistive navigation of blind people. In: Proc Int Conf Pattern Recognit, pp 2594–2597 (2012)
21. Lei Y, Phung SL, Bouzerdoum A, Le Thanh H, Luu K (2022) Pedestrian lane detection for assistive navigation of vision-impaired people: Survey and experimental evaluation. IEEE Access 10:101071–101089
22. Li H, Xiong P, An J, Wang L: Pyramid attention network for semantic segmentation. In: Proc Br Mach Vis Conf, pp 1–13 (2018)

23. Lin TY, Dollar P, Girshick R, He K, Hariharan B, Belongie S: Feature pyramid networks for object detection. In: Proc Conf Comput Vis Pattern Recognit, pp 2117–2125 (2017)
24. Liu H, Simonyan K, Yang Y: DARTS: Differentiable architecture search. In: Proc Int Conf Learn Represent, pp 1–13 (2019)
25. Long J, Shelhamer E, Darrell T: Fully convolutional networks for semantic segmentation. In: Proc Conf Comput Vis Pattern Recognit, pp 3431–3440 (2015)
26. Miksik O, Petyovsky P, Zalud L, Jura P: Robust detection of shady and highlighted roads for monocular camera based navigation of UGV. In: Proc IEEE Int Conf Robot Autom, pp 64–71 (2011)
27. Minaee S, Boykov Y, Porikli F, Plaza A, Kehtarnavaz N, Terzopoulos D (2022) Image segmentation using deep learning: a survey. IEEE Transactions on Pattern Analysis and Machine Intelligence 44(7):3523–3542
28. Nguyen TNA, Phung SL, Bouzerdoum A (2020) Hybrid deep learning-gaussian process network for pedestrian lane detection in unstructured scenes. IEEE Trans Neural Networks Learn Syst 31(12):5324–5338
29. Oh C, Son J, Sohn K: Illumination robust road detection using geometric information. In: Proc IEEE Conf Intell Transp Syst, pp 1566–1571 (2012)
30. Phung SL, Le MC, Bouzerdoum A (2016) Pedestrian lane detection in unstructured scenes for assistive navigation. Comput Vis Image Underst 149:186–196
31. Ramstrom O, Christensen H: A method for following unmarked roads. In: Proc IEEE Symp Intell Veh, pp 650–655 (2005)
32. Rasmussen C: Texture-based vanishing point voting for road shape estimation. In: Proc Br Mach Vis Conf, pp 470–477 (2004)
33. Ronneberger O, Fischer P, Brox T: U-Net: Convolutional networks for biomedical image segmentation. In: Proc. Int Conf Med Image Comput Comput Assist Interv, pp 234–241 (2015)
34. Se S, Brady M (2003) Road feature detection and estimation. Mach Vis Appl 14(3):157–165
35. Sotelo MA, Rodriguez FJ, Magdalena L, Bergasa LM, Boquete L (2004) A color vision-based lane tracking system for autonomous driving on unmarked roads. Auton Robots 16(1): 95–116
36. Sultana F, Sufian A, Dutta P (2020) Evolution of image segmentation using deep convolutional neural network: A survey. Knowledge-Based Syst 201–202:1–25
37. Tan C, Hong T, Chang T, Shneier Michael: Color model-based real-time learning for road following. In: Proc IEEE Conf Intell Transp Syst, pp 939–944 (2006)
38. Tao A, Sapra K, Catanzaro B: Hierarchical multi-scale attention for semantic segmentation. ArXiv e-prints pp 1–11 (2020)
39. Le Thanh H, Phung SL, Bouzerdoum A (2022) Bayesian gabor network with uncertainty estimation for pedestrian lane detection in assistive navigation. IEEE Transactions on Circuits and Systems for Video Technology 32(8):5331–5345
40. Uddin MS, Shioyama T: Bipolarity and projective invariant-based zebra-crossing detection for the visually impaired. In: Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit Work, pp 22–30 (2005)
41. Wang J, Sun K, Cheng T, Jiang B, Deng C, Zhao Y, Liu D, Mu Y, Tan M, Wang X, Liu W, Xiao B: Deep high-resolution representation learning for visual recognition. IEEE Trans Pattern Anal Mach Intell pp 1–16 (2019)
42. Xu Y, Xie L, Zhang X, Chen X, Qi Gj, Tian Q, Xiong H: PC-DARTS: Partial channel connections for memory-efficient architecture search. In: Proc Int Conf Learn Represent, pp 1–13 (2020)
43. Zhao H, Shi J, Qi X, Wang X, Jia J: Pyramid scene parsing network. In: Proc Conf Comput Vis Pattern Recognit, pp 2881–2890 (2017)
44. Zhou Z, Siddiquee MMR, Tajbakhsh N, Liang J: Unet++: A nested U-Net architecture for medical image segmentation. In: Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support, pp 3–11 (2018)

Sui Paul Ang received the B. CompSc. in software engineering (2017), M.Phil. in computer engineering (2018), and Ph.D. in computer engineering (2023) from the University of Wollongong, Australia. His research interests include neural architecture search, deep learning, medical imaging, pattern recognition, and computer vision.

Son Lam Phung (Senior Member, IEEE) received the B.Eng. (Hons.) and Ph.D. degrees in computer engineering from Edith Cowan University, Australia, in 1999 and 2003, respectively. Dr Phung is currently a Professor at the University of Wollongong. He was also a Visiting Senior Research Scientist at VinAI in 2020-2021. He has published over 136 papers in journals and international conferences. His research interests include image and signal processing, neural networks, pattern recognition, and machine learning. Dr Phung has served as the Chief Investigator for over 18 research projects funded by government agencies (research, defense, intelligence, foreign affairs, and trade) and industry. He was awarded the University Medal in 2000. He is currently serving as an Associate Editor for IEEE Access and a Section Editor for Sensors.

Soan T. M. Duong received her B.Eng degree in information technology from Le Quy Don Technical University, Vietnam, in 2010; her M.Eng degree in computer science from Dongguk University, South Korea, in 2014; and her Ph.D. degree in computer engineering from the University of Wollongong, Australia, in 2020. She was awarded the Examiners' Commendation for Outstanding Thesis in 2020. Her research interests include image processing, medical image registration, medical image processing, machine learning, and neural architecture search.

Abdesselam Bouzerdoum (M'89 - SM'03) received the M.Sc. and Ph.D. degrees in electrical engineering from the University of Washington, Seattle, USA. He has extensive experience in teaching, research, and academic leadership. He is currently serving as Associate Provost for Academic Affairs at Hamad Bin Khalifa University (HBKU), Doha, Qatar. Most recently, he served as Head of the ICT Division, College of Science and Engineering, HBKU. In 2004, he was appointed Professor and Head of School of Electrical, Computer and Telecommunications Engineering at the University of Wollongong (UOW), Wollongong, Australia, where he also served as Associate Dean (Research) from 2007 to 2013. In 2015, he was promoted to Senior Professor of computer engineering at UOW. From 2009 to 2011, he was a member of the Australian Research Council College of Experts and served as Deputy Chair of the EMI panel. He was a Distinguished Visiting Professor at several international institutions in France, USA, Germany, China, and New Zealand.

Dr. Bouzerdoum is the recipient of the Eureka Prize for Outstanding Science in Support of Defence or National Security (2011), the Chester Sall Award of IEEE Trans. Consumer Electronics (2005), and a Distinguished Researcher Award (Chercheur de Haut Niveau) from the French Ministry (2001). He served as Associate Editor for 5 International journals, including *IEEE Transactions on Image Processing.* His main research interests include signal and image processing, radar imaging, vision, machine learning, and pattern recognition.