Efficient Pyramid Context Encoding and Feature Embedding for Semantic Segmentation

DOI:
10.1016/j.imavis.2021.104195

Document Version
Accepted author manuscript

Citation for published version (APA):
Liu, M., & Yin, H. (2021). Efficient Pyramid Context Encoding and Feature Embedding for Semantic Segmentation. *Image and Vision Computing*, *111*, [104195]. https://doi.org/10.1016/j.imavis.2021.104195

Published in:
Image and Vision Computing

Efficient Pyramid Context Encoding and Feature Embedding for Semantic Segmentation

Mengyu Liu, Hujun Yin*

Department of Electrical and Electronic Engineering, The University of Manchester, Manchester, UK

Abstract

For reality applications of semantic segmentation, inference speed and memory usage are two important factors. To address these challenges, we propose a lightweight feature pyramid encoding network (FPENet) for semantic segmentation with a good trade-off between accuracy and speed. We use a series of feature pyramid encoding (FPE) blocks to encode context at multiple scales in the encoder. Each FPE block consists of different depthwise dilated convolutions that perform as a spatial pyramid to extract features and reduce computational costs. During training, a one-shot neural architecture search algorithm is adopted to find the optimal structure for each FPE block from a large search space with a small search cost. After the search for the encoder, a mutual embedding upsample module is introduced in the decoder, consisting of two attention blocks. The encoder-decoder attention mechanism is used to help aggregate efficiently high-level semantic features and low-level spatial details. The proposed network outperforms the existing real-time methods with fewer parameters and improved inference speed on the Cityscapes and CamVid benchmark datasets. Specifically, it achieved 72.3% mean IoU on the Cityscapes test set with only 0.4M parameters and 192.6 FPS speed on an Nvidia Titan V100 GPU, and 73.4% mean IoU with 116.2 FPS when running on higher resolution images.

Keywords: Semantic segmentation, Convolutional neural networks, Pyramid context encoding, Real-time processing

1. Introduction

Semantic segmentation is a fundamental topic in computer vision and is becoming increasingly important in various applications such as autonomous systems, medical image diagnostics and video content annotation. It is a process of assigning a category label to each pixel for a given image with consideration of image context. With the success of convolutional neural networks (CNNs) in image classification [1, 2, 3, 4], many methods have achieved impressive performances in semantic segmentation by utilizing deep CNNs as end-to-end dense classifiers [5, 6, 7, 8]. Some methods further employ post-processing methods to model pairwise distribution of pixel labels [9, 10] or refine segmentation boundaries [11].

Most recent methods are based on the fully convolutional network (FCN) [5]. Some [8, 9, 12] utilize different dilated convolutions or pooling operations to encode multi-scale features at the end of the base models, and then multi-scale context information is aggregated by combining these extracted features. Others [7, 13] adopt a self-attention mechanism to capture long-range feature dependencies in spatial and channel dimensions, respectively. Besides, many semantic segmentation models adopt the U-shape structure [14, 15, 16, 17] with an encoder to extract features and a decoder to fuse high-level features with low-level features for final pixel-level classification. Methods in [18, 19] adopted symmetric encoder-decoder structure to preserve spatial information and refine boundaries.

Most high performing segmentation methods use deep and wide architectures (e.g. ResNet101 [1], DenseNet [3]) and time consuming post-processing (e.g. CRFs, multi-scale testing), while some [17, 18] employ complicated decoders. These models often have millions of parameters and hence are computational expensive and require huge memory storage. For instance, PSPNet [8] has 65.7 million parameters and DeepLabV3+ [6] contains 54.6 million parameters. These methods take a long time to process an image even on a high-end GPU-based platform, impossible for real-time processing. For semantic segmentation running on resource-constrained devices like autonomous vehicles, there is a demand for low latency and low memory footprint without scarifying much in accuracy.

To address these challenge, several strategies, summarized as follows, have been developed to trade off between accuracy and speed for designing CNN architectures. (1) Knowledge distillation [20] method uses output information of a teacher network to teach a student network. A compact network is trained using knowledge extracted from a pre-trained cumbersome network [21, 22]. (2) Quantization of weights is another strategy used to train networks with low precision weights [23, 24, 25, 26]. (3) Manually designed lightweight architectures to improve effi-

*Corresponding author

Email addresses: mengyu.liu@manchester.ac.uk (Mengyu Liu), hujun.yin@manchester.ac.uk (Hujun Yin)

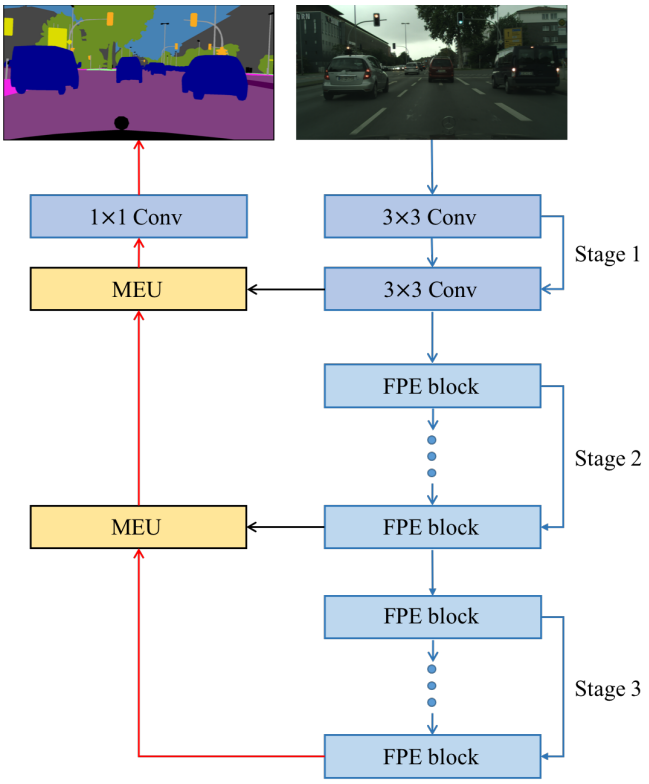


Figure 1: Architecture of the proposed network with an example input image and its corresponding output.

ciency. Some methods take downsampled inputs and fuse features at different levels to reduce computational complexity [27, 28], while others prune redundant channels to reduce parameters [29, 30]. These lightweight networks have achieved faster inference at a cost of lower accuracy on benchmarks [31, 32]. However, features extracted from downsampled images lack spatial details, and pruned shallow networks are weak in encoding contextual information with small receptive fields. Some U-shape structured models [33, 29] utilize light decoders, consisting of few convolutional layers and bilinear upsampling to recover resolution. These simple decoders reduce number of parameters and increase speed, but fine information is lost, leading to coarse segmentation at boundaries.

More recently, with the development of neural architecture search (NAS) algorithms, automatically identified networks have exceeded human-designed networks on image recognition task [34, 35]. Researchers have explored NAS for semantic segmentation and achieved state-of-the-art results. Auto-DeepLab [36] jointly searches the architectures at network and cell levels. CAS [37] searches for computational cells and multi-scale cells with constraints for lightweight networks.

Based on these observations, we propose a lightweight feature pyramid encoding network (FPENet) consisting of an encoder and a decoder for real-time semantic segmentation. Its architecture is shown in Figure 1. Unlike other methods that place a spatial pyramid module at end of the network, we build a spatial pyramid in each block to capture spatial dependencies and to learn representations from feature maps at various levels. In the encoder, a sequence of feature pyramid encoding (FPE) blocks are stacked, and in each FPE block, groups of depthwise dilated convolutions of different rates are employed to perform as a spatial pyramid and to reduce computational complexity. The specific structures (i.e. number of filters, filter sizes and dilation rates) of each FPE block are determined by NAS. For the decoder, in order to aggregate features of different levels efficiently, we propose a mutual embedding upsample (MEU) module based on the encoder-decoder attention mechanism. It uses global contextual concepts from high-level features to guide low-level features and embeds local spatial information from low-level features into high-level features simultaneously.

This paper extends our preliminary work [38] in four aspects: (1) redesign of a deeper but faster architecture to further reduce inference latency, (2) extensive schemes of FPE blocks explored by varying filter size and expansion rate, together with additional analysis of FPE blocks, (3) one-shot NAS algorithm adopted and threshoulding regularizations proposed to search for optimal structure of FPE blocks to improve performance; and (4) various attention approaches compared and analysed in the MEU modules with more experiments.

Main contributions of the work are summarized as,

(1) An efficient feature pyramid encoding scheme is proposed to encode multi-scale features and to reduce computational complexity with groups of depthwise dilated convolutions.

(2) A mutual embedding upsample module is introduced to aggregate high-level and low-level features.

(3) Significant improvements over the existing segmentation methods are obtained on Cityscapes [31] and CamVid [39] benchmarks, with much fewer parameters but faster inference speed.

2. Related Work

We review recent developments in semantic segmentation especially those for real-time processing and various studies exploring impact of encoding multi-level contextual features. Methods that adopt NAS to explore network architecture for specific tasks are also reviewed, followed by summarizing recent research on feature aggregation.

Real-time Segmentation Algorithms. These methods often require to make trade-off between accuracy and speed to be lightweight. One method is to use a lightweight backbone with efficient context encoding operations [40]. In SegNet [30], a light-weight encoder-decoder structure is used to reduce computational complexity. While in ENet [29], the last stage of the network is discarded and the number of downsampling times is reduced to shrink the model. Mehta *et al.* proposed ESPNet [41], to use efficient

pyramid modules to extract multi-scale features. Another strategy is to use multi-path architecture. In ICNet [27] and ContextNet [28], multi-scale images are employed as inputs of cascaded networks to extract features. Downsampled images are applied to deep branches, while large images are applied to shallow branches in these two models to reduce computation. BiSeNet [42] extracts high-level semantic features and low-level spatial information independently with two paths.

Multi-level Contextual Features. Encoding contextual features at multiple levels helps achieve good results in semantic segmentation due to multiple scales of objects and spatial dependency. Zhao *et al.* showed the benefit of global contextual features and proposed PSPNet [8] with a multi-scale spatial pooling module at the end of the model to exploit multi-level contextual features. In [9], an atrous spatial pyramid pooling (ASPP) module was proposed to model semantic contextual information. ASPP contains several parallel atrous (dilated) convolutions of different rates, so multi-level contextual features are encoded simultaneously. In the pyramid attention network (PAN) [12], spatial pyramid pooling is adopted to extract different scale information and generate precise pixel-level attention for high-level contextual features. In Res2Net [43], the 3×3 convolution filters in residual block are replaced with smaller groups of filters to extract contextual information simultaneously.

Neural Architecture Search (NAS). NAS is an algorithm to automatically find the best architecture for various deep learning tasks within a search space. Most work on NAS is based on reinforcement learning [44, 45, 46] to sample candidate networks or evolutionary algorithms [34, 47, 48] with a population of models repeatedly trained and mutated until the best is found. Although these two kinds of algorithms can achieve remarkable results, they are very time-consuming and not suitable for reality applications because thousands of models are required to be trained from scratch during the search. To address this problem, some one-shot NAS algorithms have been proposed to speed up the search process. In [49] a one-shot model was trained with all the operations in the search space and then child models were sampled by zeroing out other operations for evaluation. DARTS [50] introduces a continuous relaxation for architecture distribution, leading to a differentiable search space. Each operation in DARTS is assigned a weight to be optimized; the operation with largest weights is selected after training. In ProxylessNAS [51], an over-parametrized model containing all candidate blocks was trained, each block was assigned an architectural weight and a binary gate to make the architecture differentiable. In FBNet [52], a super net containing all candidate blocks was trained. During inference, only one candidate block was sampled by the Gumbel softmax function and the corresponding architecture was updated. For semantic segmentation task, Auto-DeepLab [36] was used to jointly search the downsampling strategy and cell level architectures using the DARTS search scheme. CAS [37] searches for computational cells and multi-scale cells separately with latency constraints. FasterSeg [53] combines NAS with knowledge distillation to search a teacher and a student networks simultaneously. In [54], a jagged path pruning strategy was proposed to ignore invalid paths to reduce the search cost.

Feature Aggregation. Because of repeated downsampling layers in CNNs, directly upsampling the final score map to the original resolution would lead to coarse results and loss of fine details. FCN adopts skip connections which combine coarse and fine predictions to reconstruct dense feature maps. Ronneberger *et al.* proposed a U-shape network [14] composed of an encoder and a symmetric decoder, and long skip connections were introduced to link these two parts. Peng *et al.* [16] utilized boundary refinement modules in the decoder to enhance feature aggregation ability. Li *et al.* [12] proposed a global attention upsampling module in the decoder to extract global context of high-level features to guide low-level feature information.

3. Methods

In this section, we first describe how to build the spatial pyramid in the proposed feature pyramid encoding (FPE) blocks and how to leverage NAS to search for specific structure of each block. Then we introduce the mutual embedding upsample (MEU) module in detail. Finally, the complete network architecture is presented.

3.1. Feature Pyramid Encoding

Encoding multi-scale features with different sizes of receptive field has been shown helpful for semantic segmentation. Many approaches [9, 8, 12] encode and aggregate multi-scale features with a pyramid of different filters or pooling modules at end of the model. While others [41, 55, 33] adopt parallel dilated convolutions with different rates in each block to combine local information with surrounding context.

In semantic segmentation, dilated convolution is commonly used to enlarge receptive field by inserting zeros between weights of convolutional kernels without increasing parameters. However, using dilated convolution introduces the "gridding" artifacts [56], as shown in Figure 2. For a dilated convolution with kernel size $k \times k$ and dilation rate r, the convolution region size is $[r(k-1)+1] \times [r(k-1)+1]$, but the actual number of pixels participate in the computation from the convolution region is only $k \times k$. Figure 2(a) shows that, if $k = 3$ and $r = 2$, only 9 out of 25 pixels are used for convolution, and most information (72%) is lost. When r becomes larger, the problem become severer, indicating that although dilated convolution with large dilation rate can enlarge convolution kernel

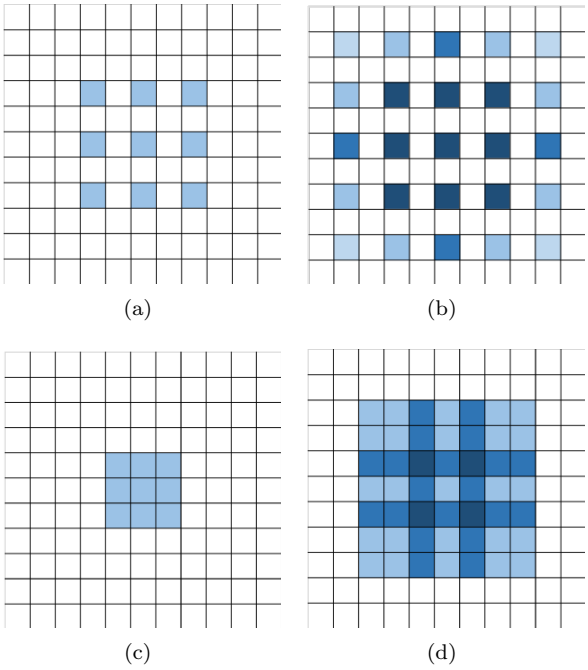


Figure 2: An example of gridding artifact problem, pixels participate in convolution are shown in blue. (a) A 3×3 dilated convolution with rate 2. (b) Another 3 × 3 dilated convolution with rate 2 stacked on top of (a). (c) A 3 × 3 dilated convolution with rate 1. (d) Another 3 × 3 dilated convolution with rate 2 stacked on top of (c).

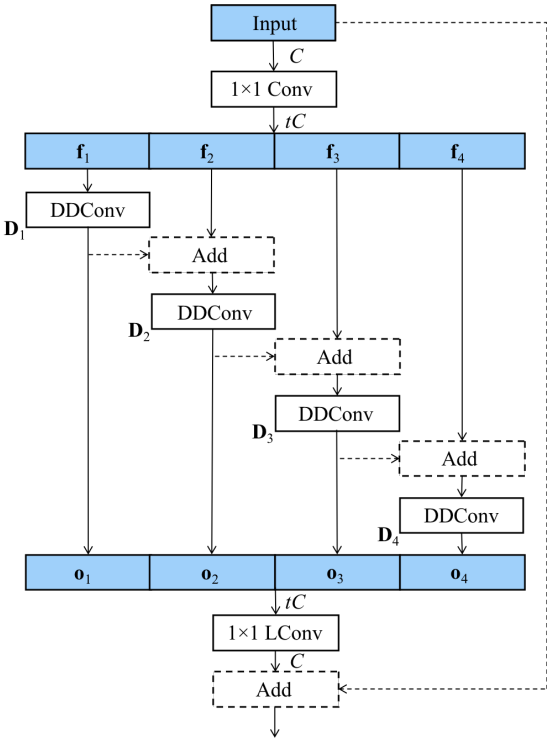


Figure 3: Structure of FPE block (stride=1). Dashed lines and blocks do not exist when stride=2. Expansion ratio is t, and dilation rate of branch $\mathbf{D}_i$ is 2^{i-1}. DDConv: depthwise dilated convolution. LConv: linear convolution. C: the number of input channels.

to encode more contextual information, most local details are lost. To address this problem, cascaded dilated convolutions with different rates are used to generate feature maps. As shown in Figure 2(c) and Figure 2(d), pixels are convoluted with two cascaded dilated convolution layers, the second layer can access a broader region due to the first layer. The receptive field is unchanged but all pixels participate in the computation.

Figure 3 shows the structure of the proposed FPE block, based on the mobile inverted bottleneck convolution (MBConv) [57] and the Res2Net module [43]. It is composed of a 1×1 expansion convolution, a group of depthwise dilated convolutions (DDConvs) and a 1×1 linear squeeze convolution, and residual connection is employed where the number of input channels is equal to the number of output channels.

For an input feature map of size $H \times W \times C$ where H, W are the spatial height and width of the feature map, respectively, and C is the number of channels, the FPE block first expands the number of channels from C to tC using 1×1 convolution. The output feature map is split into t subsets of C channels, denoted by $\mathbf{f}_i$, $i \in \{1, ..., t\}$, and each subset is processed by a group of depthwise dilated filters D_i in parallel branches. The output of D_i is added to the following subset $\mathbf{f}_{i+1}$, and then processed by D_{i+1}. The outputs of these parallel branches are concatenated and then fused by the final 1×1 linear convolution to reduce to C channels.

Different from the Res2Net module, the input of parallel branches are different subsets of the output of preceding 1×1 convolution, and batch normalization and ReLU non-linearity are only applied after the concatenation operation. The pyramid encoding mechanism is performed by these t parallel depthwise dilated convolutions, and the dilation rate of D_i is 2^{i-1}. For branch i, the output $\mathbf{o}_i$ can be written as

$$\mathbf{o}_i = \begin{cases} D_1 * \mathbf{f}_1 & i = 1; \\ D_i * (\mathbf{o}_{i-1} + \mathbf{f}_i) & 1 < i \leq t, \end{cases} \tag{1}$$

where $*$ refers to convolution operation. In branch i, the DDConv filter $\mathbf{D}_i$ processes all the outputs from the previous branches. Because of the distributive nature of convolution operations, Eqn. 1 can be rewritten as

$$\begin{aligned} \mathbf{o}_i &= D_i \left(D_{i-1} \left(\dots \right) + \mathbf{f}_i \right) \\ &= D_i \dots D_1 \mathbf{f}_1 + \dots + D_i \mathbf{f}_i \\ &= \prod_{j=1}^{i} D_j \mathbf{f}_1 + \dots + \prod_{j=i}^{i} D_j \mathbf{f}_i \\ &= \sum_{k=1}^{i} \prod_{j=k}^{i} D_j \mathbf{f}_k, \end{aligned} \tag{2}$$

where the convolution symbol is omitted for brevity.

Hence, in each branch i, according to Eqn. 2, the output $\mathbf{o}_i$ can be considered as result of $\mathbf{f}_i$ being convolved with cascaded DDConv filters. The dilation rate of D_i increases with the number of branches, and the number of

pixels participating in computation increases with the dilation rate. In the early branches, a few convolutional filters with small receptive fields are adopted to learn local features. Meanwhile, in the late branches, more convolutional filters with larger receptive fields are employed to capture global context information and to address the "gridding" artifacts caused by single dilated convolutional filter. Besides, branch D_i reuses all the features extracted from the previous branches, and this can enhance the information flow.

The FPE block can be considered as a spatial pyramid encoding modules with t branches, where the dilation rate increases one by one and contextual features are encoded efficiently with just four scales. The final output of FPE block is a feature map generated by multi-scale features, carrying local and surrounding contextual information.

3.2. Neural Architecture Search

Previous work [44, 45, 34] on NAS sampled architectures from a search space and trained them separately from scratch. This is time-consuming and requires enormous resources. One-shot approaches [49, 50, 58] addressed this problem by weight sharing. In the one-shot approach, given a super net $\mathcal{A}$, which contains all the candidates in the architecture search space, and the weight set $W_{\mathcal{A}}$ of $\mathcal{A}$, the training cycle of one-shot approaches can be decoupled into two steps:

Firstly, the weights of the super net are optimized as a normal network by minimizing the training loss:

$$W_{\mathcal{A}}^{*} = \arg\min_{W_{\mathcal{A}}} \mathcal{L}_{train}\left(\mathcal{N}(\mathcal{A}, W_{\mathcal{A}})\right). \tag{3}$$

Secondly, the architecture is updated to minimize the validation loss based on the trained super net:

$$\mathcal{A}^{*} = \arg\min_{\mathcal{A}} \mathcal{L}_{val}\left(\mathcal{N}(\mathcal{A}, W_{\mathcal{A}}^{*})\right). \tag{4}$$

After each training cycle, the updated $W_{\mathcal{A}}^{*}$ and $\mathcal{A}^{*}$ are inherited directly as initialization for next training cycle. This bilevel optimization cycle is repeated until convergence, and the final optimal network architecture $a \in \mathcal{A}$ with weights $w_a \in W_{\mathcal{A}}$ is found to achieve the minimal loss $\mathcal{L}_{val}\left(\mathcal{N}(a, w_a)\right)$. Hence, the super net $\mathcal{A}$ is only trained once, and all individual architectures in the super net share the same weights $W_{\mathcal{A}}$ to reduce the search cost.

In the current work, we construct a layer-level search space, and each layer is chosen from a set of candidate FPE blocks $\mathcal{O}$. The architecture of the super net is described in Table 1. The outputs of different stages are fused to generate the final segmentation results following the FCN model. The first stage and the last layers are fixed, while the remaining layers need to be searched. The super net consists of three stages, and the number of channels in each stage is 16, 32, 64, respectively, which were selected empirically. In stages 2 and 3, we employ 3 and 9 layers respectively, and the strides of DDConvs are set to 2 in the first layers of each stage to downsample the feature maps.

Table 1: Architecture details of the super net. Input sizes is $3 \times 1536 \times 768$. t is the expansion ratio of FPE block. C is the number of classes. "n" denotes the the number of repeated blocks and "s" denotes the stride of DDConv in FPE block.

Stage	Input shape	Block	Channel	n	s
stage1	1536×768	3×3 Conv	16	1	2
	1024×512	3×3 Conv	16	1	2
	512×256	3×3 Conv	16	1	1
stage2	512×256	FPE	32	1	2
	256×128	FPE	32	2	1
stage3	256×128	FPE	64	1	2
	128×64	FPE	64	8	1
	128×64	1×1 Conv	C	1	1

Table 2: Configurations of candidate FPE blocks in the search space.

Block name	Kernel size	Expansion rate
k3_e2	3	2
k3_e3	3	3
k3_e4	3	4
k5_e2	5	2
k5_e3	5	3
k5_e4	5	4

For each searchable layer in the super nets, we search for the proper expansion ratio (i.e. number of branches) and kernel size for the DDConv of the FPE block. Hence, the search space contains 6 candidate blocks, and their configurations are shown in Table 2. As the super net contains 12 searchable layers, and each layer can be chosen from 6 candidate blocks, the total number of architectures that can be selected is 6^{12}.

In order to find the optimal architecture efficiently and reduce search cost, we adopt a single path one-shot search algorithm. Instead of optimizing all the weights of the super net in Eqn. 3, only one candidate block is sampled and updated at each layer with sampling probability,

$$P_{l,i}(b_l = o_i) = \frac{\exp(\alpha_{l,i})}{\sum_{j=1}^{6} \exp(\alpha_{l,j})}, \tag{5}$$

where $\alpha \in \mathbb{R}^{12\times 6}$ is the architecture parameters, $\alpha_{l,i}$ represents the architecture parameter of candidate block i at layer l, b_l denotes the sampled block at layer l, and $o_i \in \mathcal{O}$ denotes candidate block i. Hence, Eqn. 3 can be rewritten as

$$w_a^{*} = \arg\min_{w_a} \mathcal{L}_{train}\left(\mathcal{N}(a, w_a)\right), \tag{6}$$

where architecture a is composed of sampled blocks b. This sampling strategy can save a large amount of memory when optimizing the weights of the super net as only one path is activated.

In the meantime, instead of searching in the architecture space $\mathcal{A}$ to find the optimal architecture, we relax

the problem to optimizing the architecture parameters α by minimizing the validation loss. Hence, Eqn. 4 can be rewritten as

$$\alpha^* = \arg\min_{\alpha} \mathcal{L}_{val}\left(\mathcal{N}(\mathcal{A}, w_a^*)\right). \tag{7}$$

However, the loss in Eqn. 7 is not directly differentiable to the architecture parameters α, because the sampling probabilities generated by α are not directly involved in the computation and cannot be optimized by the stochastic gradient descent (SGD). To address this problem, we adopt the binary gating method proposed in [51] to make gradient w.r.t. sampling probabilities involved in the computation graph. We select the candidate block with the largest sampling probability at each layer to compose the final searched architecture.

Although one-shot approach greatly improves searching efficiency, there is still an issue to address. The weights of different candidate blocks in the super net are deeply coupled, one candidate block is optimized with different combinations of other candidate blocks in different training cycles, and the weights are inherited after each cycle. This may introduce bias in weights and make search difficult. In experiments, we observed that some layers converged very early with high largest sampling probabilities (close to 0.9), while others converged slowly with low largest sampling probabilities (lower than 0.3). To address this issue, we formulate two thresholding regularizations for training: (1) When the sampling probability of one candidate block is larger than a upper threshold, this block is considered as a fixed layer in the future training and other blocks in the same layer are discarded. (2) When the sampling probability of one candidate block is smaller than a lower threshold, this block is removed from the search space. We empirically set the upper and lower thresholds to 0.85 and 0.05 respectively.

3.3. MEU Module

In the U-shape models for segmentation, the decoder is usually designed to aggregate features extracted at different levels and to recover the resolution. Many methods [6, 8] use bilinear upsampling or several simple convolution layers as a naive decoder. These naive decoders only consider semantic concepts from higher layers and ignore low-level spatial details leading to coarse segmentation. While other approaches [16, 17, 15] adopt complicated structures to build decoders to aggregate features from different stages and utilize low-level features to refine boundaries. However, these well-designed decoders are time-consuming.

Attention Mechanism. Instead of adopting self-attention mechanism at the top of backbone, we consider using the encoder-decoder attention mechanism where attention key and query are from two different sets of features in the decoder of the U-shape model. In deep neural networks, high-level features at higher layers contain contextual information while low-level features are rich in spatial details. This makes feature aggregation difficult. We consider that low-level and high-level features can be used as key features to provide spatial and contextual information to each other mutually.

We propose an encoder-decoder attention framework for aggregating features in the decoder of the U-shape model. This attention mechanism at position i of $\mathbf{z}$ can be defined as:

$$\mathbf{y}_i = F(\mathbf{z}_i, W_i \sum_{j \in \Omega_i} \theta_j \mathbf{x}_j), \tag{8}$$

where $\sum_{j\in\Omega_i} \theta_j \mathbf{x}_j$ denotes the attention map generation within key region Ω_i for query $\mathbf{z}_i$, which aggregates features of all positions using weight θ to generate the global context feature, then W_i transforms it to attention map and obtain the global relationships of features, and $F(\cdot)$ denotes the feature aggregation function to embed the attention weights to $\mathbf{z}$ at each position to obtain attention features $\mathbf{y}$.

As shown in Figure 4, the MEU module consists of two attention blocks, each being considered as an instance of the encoder-decoder attention framework. In the MEU module, first, two 1×1 convolutions with batch normalization and ReLU are performed on the high-level and low-level features to generate $\mathbf{x}_H$ and $\mathbf{x}_L$, respectively.

Channel attention block. We generate a channel attention map to exploit the global context information from the high-level feature map $\mathbf{x}_H$. Because each channel in a feature map contains a specific semantic concepts [59], to capture these global concepts in each channel, we adopt a global average pooling operation F_{avgpool} and a 1×1 convolution δ like the squeeze-excitation (SE) block proposed in [60] to squeeze and transform the high-level feature map. The obtained 1×1 vector is considered as the channel attention map containing the global contextual information of each channel. Next, this vector is fused with the low-level feature map $\mathbf{x}_L$ using broadcast element-wise multiplication to obtain the attention feature $\mathbf{y}_L$. The detailed structure of channel attention block is depicted in Figure 4(b) and can be formulated as

$$\mathbf{y}_L = \mathbf{x}_L \times \delta\left(F_{\text{avgpool}}\left(\mathbf{x}_H\right)\right). \tag{9}$$

where F_{avgpool} groups features at all positions together to exploit the inter-channel relationships. δ transforms the global context feature to the channel attention map. Feature aggregation function is performed as an element-wise multiplication.

Spatial attention block. We create a spatial attention map to exploit the spatial information from low-level feature map $\mathbf{x}_L$. Different from high-level features, low level features contain more spatial details. To preserve such spatial information at each position, we first squeeze $\mathbf{x}_L$ using an average pooling operation F'_{avgpool} along the channel axis.

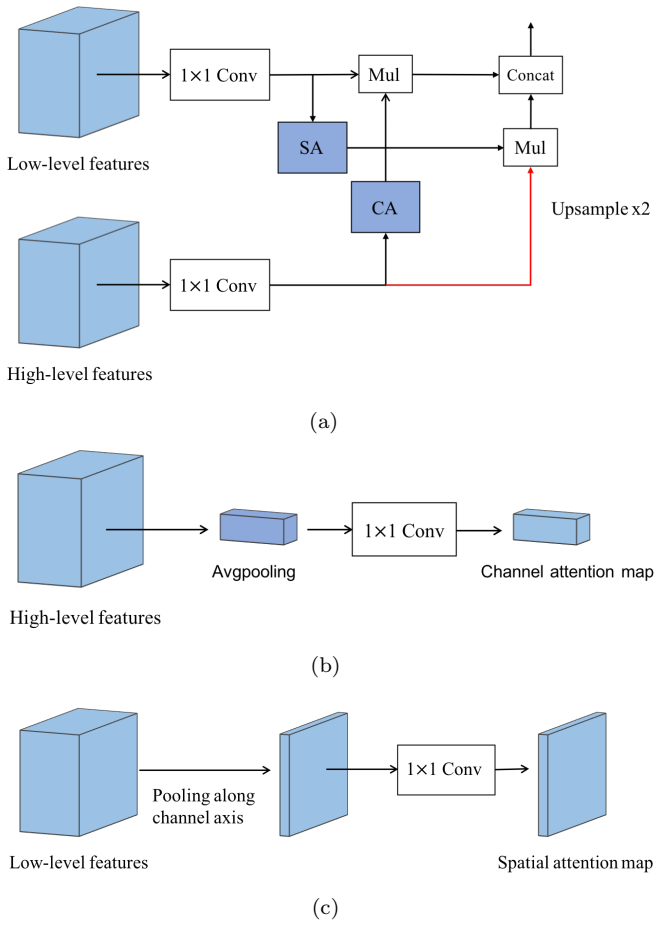


Figure 4: (a) Structure of MEU module. SA: spatial attention block. CA: channel attention block. (b) Spatial attention block. (c) Channel attention block.

Then, a single channel 1×1 convolution δ' is applied to generate a single channel spatial attention map that contains local spatial information at each position. Finally, this attention map is applied to fuse with high-level feature map $\mathbf{x}_H$ using broadcast element-wise multiplication to obtain attention feature $\mathbf{y}_H$. The detailed structure of spatial attention block is depicted in Figure 4(c) and can be formulated as

$$\mathbf{y}_H = \mathbf{x}_H \times \delta' \left(F'_{\text{avgpool}} \left(\mathbf{x}_L \right) \right). \tag{10}$$

where, F'_{avgpool} aggregates features at all channels to exploit the inter-spatial relationships. δ' is a transformation function to create the spatial attention map, and features are finally aggregated by element-wise multiplication.

After attention blocks, these two attention features $\mathbf{y}_L$ and $\mathbf{y}_H$ are fused by concatenation. The spatial attention map generated from the low-level features corresponds to the importance of each pixel. It focuses on localizing the objects and refining the boundaries with spatial details. While the squeezed channel attention map generated from the high-level features reflects the importance of each channel and focuses on the global context to provide content information. The MEU module extracts these two kinds of attention map and efficiently embeds semantic concepts and spatial details into low-level and high-level features.

Table 3: Architecture details of our network. Input size is $3 \times 1536 \times 768$, k is the expansion ratio of FPE block, and C is the number of classes.

Name	Operator	Channel	Output size
stage1	3×3 Conv	16	1024×512
	3×3 Conv	16	512×256
	3×3 Conv	16	512×256
stage2	FPE×3	32	256×128
stage3	FPE×9	64	128×64
decoder2	MEU	64	256×128
decoder1	MEU	32	512×256
final	1×1 Conv	C	512×256

3.4. Network Architecture

The entire network architecture is shown in Figure 1. Based on the above discussion, we have designed this lightweight encoder-decoder model with FPE blocks and MEU modules. In order to preserve spatial information and reduce number of parameters, the total downsampling rate is 16. Structural details of the proposed model are shown in Table 3.

We employ FPE blocks in the encoder except for the stage 1, which is consists of three 3×3 convolutional layers, and the numbers of channels in each stage are 16, 32, 64, respectively. In stages 2 and 3, we employ 3 and 9 FPE blocks respectively, and stride of the depthwise dilated convolutions is set to 2 in the first blocks to downsample the feature maps. The specific structure of each FPE block is searched by NAS. We add long skip connections in stages 2 and 3, where the inputs of these two stages are combined from the output of the first and last blocks of their preceding stages. These skip connections encourage signal propagation and perform as an implicit deep supervision. Earlier layers can connect to the deepest layer directly and receive supervision from different stages of the decoder. For the decoder, two MEU modules are used to aggregate features from each stage and recover the resolution step by step. Finally, a 1×1 convolutional layer is applied as the pixel-level classifier.

4. Experiments

4.1. Implementation Protocol

All the experiments were conducted using PyTorch [61]. Stochastic gradient descent algorithm with batch size 8 and weight decay 0.0001 were used to train the networks from scratch without any pre-training on any dataset. The "poly" learning rate policy [9] was employed:

$$lr = init\ lr \times (1 - \frac{iter}{max\_iter})^{power}, \tag{11}$$

where *iter* is the current iteration, *power* is 0.9, and initial learning rate was 0.05. We employed the zero-mean normalization, random horizontal flip, random scaling between 0.5 and 1.75 for data augmentation. In the search procedure, we used the same optimizer to train weight parameters, and adopted another Adam optimizer with fixed learning rate of 0.001 to update the architecture parameters. We searched for 200 epochs on the Cityscapes dataset. Then the generated network was combined with MEU modules and trained for 600 epochs on the Cityscapes and 400 epochs on the CamVid to evaluate performance. Accuracy was measured using the mean Intersection-over-Union (mIoU) metric. The mean of cross-entropy error over all pixels was applied as the loss.

4.2. Ablation Studies

The Cityscapes is an urban street scene dataset for semantic understanding. It contains 5000 fine annotated images, divided into three sets, 2975 for training, 500 for validation and 1525 for test. Furthermore, 20000 coarsely annotated images are also provided for training. All images are of resolution, 2048×1024, and all pixels are annotated to 19 classes. In our experiments, only the fine annotated images were used for training.

In ablation studies, we evaluated only the encoder part on the validation set of Cityscapes to investigate the effect of each component and help hand-select configuration, and the kernel size of DDConv in each FPE block was set to 3×3. Naive bilinear upsampling was employed as the decoder in these networks. This section can be considered as a preliminary study before searching for the optimal architecture.

Ablation on pyramid encoding structure. We adopted four schemes to evaluate the effect of pyramid encoding structure by setting the number of branches in the FPE block to 1, 2, 4 or 6. Note, when the number is 1, the FPE block is the same as the MBConv block. The expansion ratios were the same in these schemes to keep number of parameters same, and the numbers of blocks in stages 2 and 3 were set to 3 and 9, respectively. Results are shown in Table 4, indicating that the pyramid encoding structure gave better results and three schemes improved the segmentation quality by 4.1%, 7.0% and 8.7% compared with MBConv block. These statistically significant improvements verify that the pyramid encoding structure is beneficial for segmentation as multi-scale contextual features are encoded efficiently without introducing new parameters. Although the 6-branch FPE block achieved the best result, the branches in the encoder are cascaded connected and cannot be computed in parallel, resulting higher computational complexity. Therefore we selected 4-branch FPE block as the basic block as a trade-off between speed and performance gain.

Ablation on dilation rates. We designed two kinds of 4-branch FPE block with different combinations of dilation rates and used them to build the encoder, one with dilation rates of 1, 2, 3, 4, while the other with 1, 2, 4, 8. The kernel sizes of DDConvs were all 3×3. As shown in Table 4, the models with larger dilation rates in FPE block achieved better results. The range of receptive field of the former FPE block was from 3×3 to 9×9, while the latter 3×3 to 17×17. We also tested the performance with all the dilation rates set to 1, which was similar to the original Res2Net module, and the result was worse than FPE blocks with larger dilation rates. Larger receptive field can encode more surrounding features and learn better multi-scale representations.

Ablation on addition between branches. In the FPE blocks, we added output of one branch to the input of following branch. As shown in Table 5, the addition operations between adjacent branches improved the accuracy from 66.5% to 66.9%. This improvement came from the addition operations that turned independent branches to a cascaded pyramid module, so larger dilated convolutions performed on the features extracted by smaller dilated convolutions to address the "gridding" artifacts. The number of pixels convoluted by large kernels is also increased, sim-

Table 4: Results of FPE encoder with different number of branches and dilation rates.

#Branches	Dilation rates	mIoU (%)
1 (MBConv block)	1	59.5
2	1, 2	63.6
4	1, 1, 1, 1	64.3
4	1, 2, 3, 4	65.6
4	1, 2, 4, 8	66.5
6	1, 2, 3, 4, 5, 6	68.2

Table 5: Results of FPE encoder with different settings.

Addition	Long skip	mIoU (%)
		66.5
√		66.9
√	√	68.1

Table 6: Results of FPENet with different depths, number of parameters and FLOPS are estimated on 1536×768 input.

p	q	#Params	FLOPs	mIoU (%)
3	5	233K	3.77G	61.5
3	7	305K	4.37G	66.7
5	7	325K	5.04G	66.9
3	9	382K	3.22G	68.1
5	9	398K	5.64G	68.2
3	11	450K	5.58G	68.5

ilar effect to the DenseASPP module in [62].

Ablation on long skip connection. Long skip connections were employed in stages 2 and 3 in FPENet to combine outputs of the first and final blocks. Accuracy was improved by 1.2% as shown in Table 5. Intuitively, long skip connections apply implicit supervision to earlier layers and increase flow of information.

Ablation on encoder depth. We varied numbers of blocks in stages 2 and 3 to change the depth of the encoder, denoted as p and q, respectively. The numbers of parameters, FLOPs and accuracies of different configurations are shown in Table 6. It can be seen that the value of q had more impact on accuracy than p, indicating that stacking more FPE blocks in stage 3 increases receptive field and performance. However, raising q from 9 to 11, the improvement became minor, this may be due to that large receptive field in stage 3 was beyond the size of feature maps, and efficient features could not be extracted. Therefore, for a trade-off between, we set p to 3 and q to 9 in the final architecture.

4.3. Searching for Optimal Architecture

We conducted the search experiments on the Cityscapes dataset. We randomly sampled 1000 images from the training set as the validation set for architecture search, and the remaining 1975 images were used for training with batch size 4, and the thresholding regularizations introduced in section 3.2 was applied. The search experiment was run on one GPU, and from the experiments, training one epoch took around 9 minutes, so the total search cost was about $9/60 \times 200 = 30$ GPU hours.

The architecture of our searched network is shown in Figure 5, and Table 7 shows the results evaluated on the Cityscapes validation set. For comparison, we also set up a series of baselines: (a) select certain candidate blocks for all layers; (b) randomly select several architectures from the search space and evaluate the average result; (c) search without regularization. The results in Table 7 show that our search algorithm can find better architecture with comparable parameters and FLOPs, and the proposed thresholding regularizations can further boost the performance. This searched architecture was employed as the encoder part of our network.

Besides, in the searched architecture blocks with smaller filters and less branches were selected in stage 2, while blocks with larger filters and more branches were selected in stage 3. This indicates that smaller receptive field is required in low layers to preserve better spatial details while larger receptive field is needed in high layers to encode more context information.

4.4. MEU Modules

MEU modules are used to aggregate the features extracted by the PFE blocks to provide dense pixel-level prediction. We first evaluated the MEU module with only channel attention block, then we used channel and spatial attention together in the MEU module to test the performance. As shown in Table 8, the channel and spatial attention blocks both improved the accuracy, indicating that embedding semantic concepts into low-level features and spatial details into high-level features with the MEU module can lead to better results.

Then we conducted experiments to explore different attention map transformations and aggregation approaches in the attention block. We tested linear (i.e. convolution only) and non-linear (i.e. convolution+ReLU, convolution+Sigmoid) transformations combined with different aggregation approaches (i.e. element-wise addition, element-wise multiplication). Results are shown in Table 9, indicating that multiplication aggregation performed better than addition and non-linear transformation was more efficient than linear transformation.

4.5. Cityscapes

Based on the ablation studies, we combined the FPE blocks and MEU modules to build the complete network and experimented it on the Cityscapes dataset. First, we conducted experiments to estimate the inference speed at

Table 7: Results of networks with different depths, number of parameters and FLOPS are estimated on 1536×768 input. 'Regs' represents the thresholding regularizations.

Model	#Params	FLOPs	mIoU (%)
all k3_e3	293K	2.58G	66.4
all k3_e4	382K	3.22G	68.1
all k5_e3	326K	2.79G	67.2
all k5_e4	425K	3.50G	68.8
5 random select	365K	3.34G	67.3
searched w/o regs	413K	3.23G	69.1
searched w regs	411K	3.29G	69.9

Table 8: Results of MEU module with different components.

MEU	CA	SA	mIoU (%)
w/o	—	—	69.9
w	√		71.1
w	√	√	72.5

Table 9: Results of MEU module with different transformation and aggregation methods.

Method	mIoU (%)
Conv+add	70.9
Conv+Sigmoid+mul	71.2
Conv+ReLU+mul	72.5

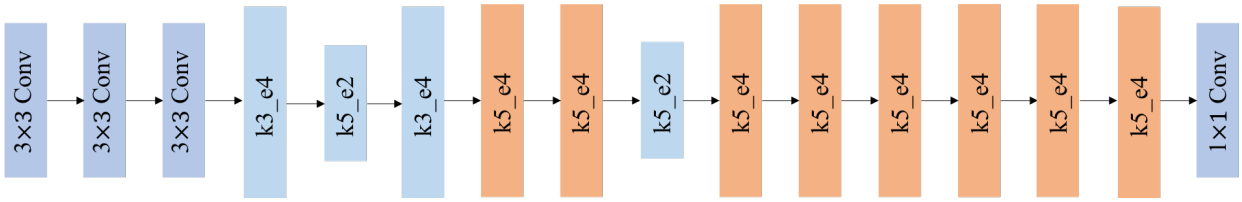


Figure 5: Architecture of searched network. Note kx_ey means a kernel size of x for its depthwise dilated convolution layer and an expansion of y for its expansion layer.

Table 10: Speed and accuracy comparison of FPENet on Cityscapes test set.

Method	Resolution	#Params	FPS	mIoU (%)
ENet [29]	1024 × 512	0.4M	78.4	58.3
ESPNet [41]	1024 × 512	0.4M	195.6	60.3
ESPNetv2 [55]	1024 × 512	0.7M	140.6	62.1
BiSeNet1 [42]	1536 × 768	5.8M	159.4	68.4
ICNet [27]	2048 × 1024	7.8M	69.6	69.5
FasterSeg [53]	2048 × 1024	4.4M	160.3	71.5
DeepLab [63]	1024 × 512	262.1M	0.3	63.1
PSPNet [8]	713 × 713	250.8M	0.8	78.4
HRNet [64]	2048 × 1024	70.3M	6.7	82.5
FPENet	1536 × 768	0.4M	192.6	72.3
FPENet	2048 × 1024	0.4M	116.2	73.4

different resolution. For fair comparison with other methods, we tested all the models on an Nvidia Titan V100 GPU, using PyTorch framework, the inference time is estimated by running the model for six seconds following [53]. The results and corresponding input sizes are shown in Table 10. Next, we trained our network with downsampled fine annotated images of Cityscapes and evaluated accuracies on the test set, as shown in Table 10. Note that we did not employ multi-scale or multi-crop test.

From Table 10, the number of parameters of the proposed FPENet is close to that of the ESPNet and the ENet, but it achieves higher performance. While the model size of FPENet is 19 and 11 times smaller than the ICNet and FasterSeg, but mIoU is 2.8% and 0.8% higher, respectively. Besides, FPENet achieves 192.6 FPS speed at 1536 × 768 input resolution, which significantly outperforms all of the existing real-time methods with higher accuracy. When the input resolution is 2048 × 1024, the accuracy performance can be further improved, and the network still maintains competitive speed. The results demonstrate the effectiveness of the proposed search algorithm and the encoder-decoder attention mechanism. Examples of the segmentation results of FPENet are presented in Figure 6.

4.6. CamVid

The CamVid road scenes dataset has fully labelled images for semantic segmentation: 367 for training, 101 for validation and 233 for test. Each image is of 960 × 720 pixels, labelled with 11 semantic classes. We used the training and validation set to train our network searched on Cityscapes, and then tested on the test set. Results and number of parameters are shown in Table 11. The proposed FPENet outperforms all other deep models even with much fewer parameters in most cases. These results show the robustness as well as good generalization ability of FPENet.

Table 11: Results on CamVid test set.

Method	#Params	mIoU (%)
ENet [29]	0.4M	51.3
FCN8 [5]	134.5M	52.0
Bayesian SegNet [65]	29.5M	63.1
BiSeNet1 [42]	5.8M	65.6
ICNet [27]	7.8M	67.1
FPENet	0.4M	69.4

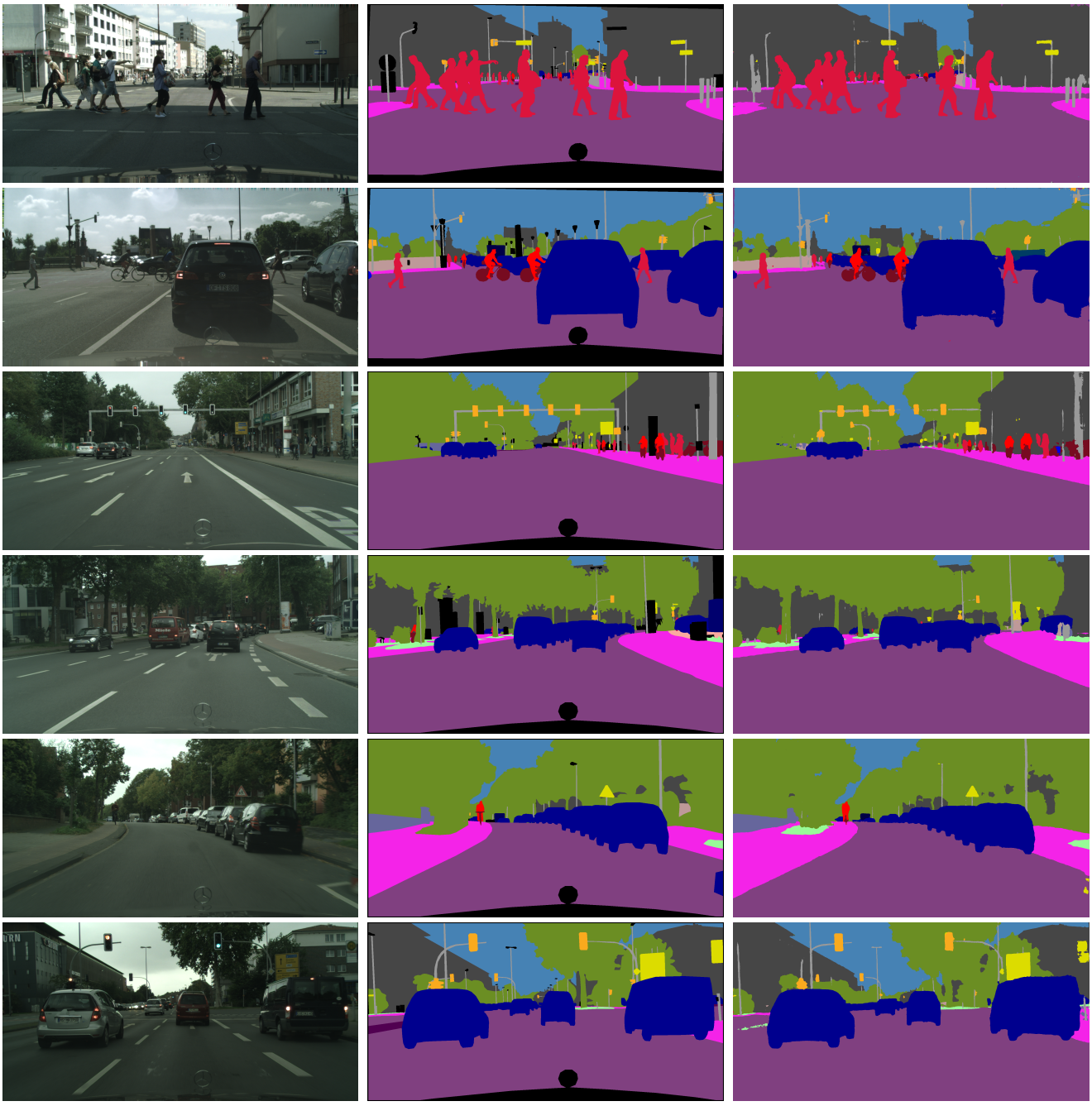
5. Conclusions

This paper presents a lightweight architecture for real-time semantic segmentation. A feature pyramid encoding (FPE) block is proposed and adopted in every stage of the proposed network to encode multi-scale features using a spatial pyramid of depthwise dilated convolutions. A one-shot neural architecture search algorithm is used to search optimal structure of each FPE block. Mutual embedding upsample (MEU) modules are employed in the decoder to aggregate features from different stages based on encoder-decoder attention mechanism. The ablation experiments show that FPE blocks significantly improve accuracy due to their expanded receptive field and enhanced information flow, and the MEU modules aggregate deep contextual features and shallow spatial features efficiently. Experimental results on the Cityscapes and CamVid datasets demonstrate marked improvements by the purposed FPENet over other real-time methods even with fewer parameters and faster inference speeds.

References

[1] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[2] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, in: Proceedings of the



(a) Image (b) Groundtruth (c) Prediction

Figure 6: Visualization results on the Cityscapes validation dataset.

IEEE conference on computer vision and pattern recognition, 2017, pp. 1492–1500.

[3] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.

[4] C. Szegedy, S. Ioffe, V. Vanhoucke, A. A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning, in: Thirty-First AAAI Conference on Artificial Intelligence, 2017.

[5] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3431–3440.

[6] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, H. Adam, Encoder-decoder with atrous separable convolution for semantic image segmentation, in: Proceedings of the European Conference on Computer Vision, 2018, pp. 801–818.

[7] J. Fu, J. Liu, H. Tian, Z. Fang, H. Lu, Dual attention network for scene segmentation, arXiv:1809.02983.

[8] H. Zhao, J. Shi, X. Qi, X. Wang, J. Jia, Pyramid scene parsing network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2881–2890.

[9] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A. L.

Yuille, Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, IEEE Transactions on Pattern Analysis and Machine Intelligence 40 (4) (2018) 834–848.

[10] G. Lin, C. Shen, A. Van Den Hengel, I. Reid, Exploring context with deep structured models for semantic segmentation, IEEE transactions on pattern analysis and machine intelligence 40 (6) (2017) 1352–1366.

[11] Z. Dong, J. Li, T. Fang, X. Shao, Lightweight boundary refinement module based on point supervision for semantic segmentation, Image and Vision Computing (2021) 104169.

[12] H. Li, P. Xiong, J. An, L. Wang, Pyramid attention network for semantic segmentation, arXiv preprint arXiv:1805.10180.

[13] Y. Yuan, J. Wang, Ocnet: Object context network for scene parsing, arXiv preprint arXiv:1809.00916.

[14] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: International Conference on Medical Image Computing and Computer Assisted Intervention, 2015, pp. 234–241.

[15] G. Lin, A. Milan, C. Shen, I. Reid, Refinenet: Multi-path refinement networks for high-resolution semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1925–1934.

[16] C. Peng, X. Zhang, G. Yu, G. Luo, J. Sun, Large kernel matters–improve semantic segmentation by global convolutional network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4353–4361.

[17] Z. Zhang, X. Zhang, C. Peng, X. Xue, J. Sun, Exfuse: Enhancing feature fusion for semantic segmentation, in: Proceedings of the European Conference on Computer Vision, 2018, pp. 269–284.

[18] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, Y. Bengio, The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017, pp. 11–19.

[19] A. Chaurasia, E. Culurciello, Linknet: Exploiting encoder representations for efficient semantic segmentation, in: 2017 IEEE Visual Communications and Image Processing (VCIP), IEEE, 2017, pp. 1–4.

[20] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, arXiv preprint arXiv:1503.02531.

[21] Y. Liu, K. Chen, C. Liu, Z. Qin, Z. Luo, J. Wang, Structured knowledge distillation for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2604–2613.

[22] T. He, C. Shen, Z. Tian, D. Gong, C. Sun, Y. Yan, Knowledge adaptation for efficient semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 578–587.

[23] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, D. Kalenichenko, Quantization and training of neural networks for efficient integer-arithmetic-only inference, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2704–2713.

[24] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, Y. Bengio, Quantized neural networks: Training neural networks with low precision weights and activations, The Journal of Machine Learning Research 18 (1) (2017) 6869–6898.

[25] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, Xnor-net: Imagenet classification using binary convolutional neural networks, in: European Conference on Computer Vision, Springer, 2016, pp. 525–542.

[26] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, Y. Zou, Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients, arXiv preprint arXiv:1606.06160.

[27] H. Zhao, X. Qi, X. Shen, J. Shi, J. Jia, Icnet for real-time semantic segmentation on high-resolution images, in: Proceedings of the European Conference on Computer Vision, 2018, pp. 405–420.

[28] R. P. Poudel, U. Bonde, S. Liwicki, C. Zach, Contextnet: Exploring context and detail for semantic segmentation in real-time, arXiv preprint arXiv:1805.04554.

[29] A. Paszke, A. Chaurasia, S. Kim, E. Culurciello, Enet: A deep neural network architecture for real-time semantic segmentation, arXiv preprint arXiv:1606.02147.

[30] V. Badrinarayanan, A. Kendall, R. Cipolla, Segnet: A deep convolutional encoder-decoder architecture for image segmentation, arXiv:1511.00561.

[31] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, The cityscapes dataset for semantic urban scene understanding, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 3213–3223.

[32] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, A. Zisserman, The pascal visual object classes (voc) challenge, International Journal of Computer Vision 88 (2) (2010) 303–338.

[33] T. Wu, S. Tang, R. Zhang, Y. Zhang, Cgnet: A light-weight context guided network for semantic segmentation, arXiv preprint arXiv:1811.08201.

[34] E. Real, A. Aggarwal, Y. Huang, Q. V. Le, Regularized evolution for image classifier architecture search, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 4780–4789.

[35] M. Tan, Q. V. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, arXiv preprint arXiv:1905.11946.

[36] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, L. Fei-Fei, Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 82–92.

[37] Y. Zhang, Z. Qiu, J. Liu, T. Yao, D. Liu, T. Mei, Customizable architecture search for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 11641–11650.

[38] M. Liu, H. Yin, Feature pyramid encoding network for real-time semantic segmentation, in: British Machine Vision Conference, 2019.

[39] G. J. Brostow, J. Shotton, J. Fauqueur, R. Cipolla, Segmentation and recognition using structure from motion point clouds, in: Proceedings of the European Conference on Computer Vision, 2008, pp. 44–57.

[40] H. Li, P. Xiong, H. Fan, J. Sun, Dfanet: Deep feature aggregation for real-time semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 9522–9531.

[41] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, H. Hajishirzi, Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation, in: Proceedings of the European Conference on Computer Vision, 2018, pp. 552–568.

[42] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, N. Sang, Bisenet: Bilateral segmentation network for real-time semantic segmentation, in: Proceedings of the European Conference on Computer Vision, 2018, pp. 325–341.

[43] S.-H. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, P. Torr, Res2net: A new multi-scale backbone architecture, arXiv preprint arXiv:1904.01169.

[44] B. Zoph, Q. V. Le, Neural architecture search with reinforcement learning, arXiv preprint arXiv:1611.01578.

[45] B. Zoph, V. Vasudevan, J. Shlens, Q. V. Le, Learning transferable architectures for scalable image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8697–8710.

[46] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, J. Dean, Efficient neural architecture search via parameter sharing, arXiv preprint arXiv:1802.03268.

[47] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, A. Kurakin, Large-scale evolution of image classifiers, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 2902–2911.

[48] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, K. Kavukcuoglu,

Hierarchical representations for efficient architecture search, arXiv preprint arXiv:1711.00436.
[49] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, Q. Le, Understanding and simplifying one-shot architecture search, in: International Conference on Machine Learning, 2018, pp. 549–558.
[50] H. Liu, K. Simonyan, Y. Yang, Darts: Differentiable architecture search, arXiv preprint arXiv:1806.09055.
[51] H. Cai, L. Zhu, S. Han, Proxylessnas: Direct neural architecture search on target task and hardware, arXiv preprint arXiv:1812.00332.
[52] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, K. Keutzer, Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 10734–10742.
[53] W. Chen, X. Gong, X. Liu, Q. Zhang, Y. Li, Z. Wang, Fasterseg: Searching for faster real-time semantic segmentation, arXiv preprint arXiv:1912.10917.
[54] W. Jing, J. Lin, H. Wang, Building nas: Automatic designation of efficient neural architectures for building extraction in high-resolution aerial images, Image and Vision Computing 103 (2020) 104025.
[55] S. Mehta, M. Rastegari, L. Shapiro, H. Hajishirzi, Espnetv2: A light-weight, power efficient, and general purpose convolutional neural network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 9190–9200.
[56] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, G. Cottrell, Understanding convolution for semantic segmentation, in: 2018 IEEE winter conference on applications of computer vision (WACV), IEEE, 2018, pp. 1451–1460.
[57] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520.
[58] X. Chen, L. Xie, J. Wu, Q. Tian, Progressive differentiable architecture search: Bridging the depth gap between search and evaluation, arXiv preprint arXiv:1904.12760.
[59] M. D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: European conference on computer vision, Springer, 2014, pp. 818–833.
[60] J. Hu, L. Shen, G. Sun, Squeeze-and-excitation networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 7132–7141.
[61] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch, in: Advances in Neural Information Processing Systems Workshops, 2017.
[62] M. Yang, K. Yu, C. Zhang, Z. Li, K. Yang, Denseaspp for semantic segmentation in street scenes, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3684–3692.
[63] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A. Yuille, Semantic image segmentation with deep convolutional nets and fully connected crfs, in: ICLR, 2015.
[64] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, et al., Deep high-resolution representation learning for visual recognition, IEEE transactions on pattern analysis and machine intelligence.
[65] A. Kendall, V. Badrinarayanan, R. Cipolla, Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding, arXiv preprint arXiv:1511.02680.