FaPN: Feature-aligned Pyramid Network for Dense Image Prediction

Shihua Huang Zhichao Lu Ran Cheng* Cheng He
Southern University of Science and Technology†
{shihuahuang95, luzhichaocn, ranchengcn, chenghehust}@gmail.com

Abstract

Recent advancements in deep neural networks have made remarkable leap-forwards in dense image prediction. However, the issue of feature alignment remains as neglected by most existing approaches for simplicity. Direct pixel addition between upsampled and local features leads to feature maps with misaligned contexts that, in turn, translate to mis-classifications in prediction, especially on object boundaries. In this paper, we propose a feature alignment module that learns transformation offsets of pixels to contextually align upsampled higher-level features; and another feature selection module to emphasize the lower-level features with rich spatial details. We then integrate these two modules in a top-down pyramidal architecture and present the Feature-aligned Pyramid Network (FaPN). Extensive experimental evaluations on four dense prediction tasks and four datasets have demonstrated the efficacy of FaPN, yielding an overall improvement of 1.2 - 2.6 points in AP / mIoU over FPN when paired with Faster / Mask R-CNN. In particular, our FaPN achieves the state-of-the-art of 56.7% mIoU on ADE20K when integrated within MaskFormer. The code is available from https://github.com/EMI-Group/FaPN.

1. Introduction

Dense prediction is a collection of computer vision tasks that aim at labeling every pixel in an image with a predefined class. It plays a fundamental role in scene understanding and is of great importance to real-world applications, such as autonomous driving [7], medical imaging [44], augmented reality [1], *etc*. The modern solutions for these tasks are built upon Convolutional Neural Networks (CNNs). With the recent advancements in CNN architectures, a steady stream of promising empirical leap-forwards was reported across a wide range of dense prediction tasks, including object detection [26, 39, 40], semantic segmentation [4, 28], instance segmentation [13, 25], and panoptic segmentation [18, 19], to name a few.

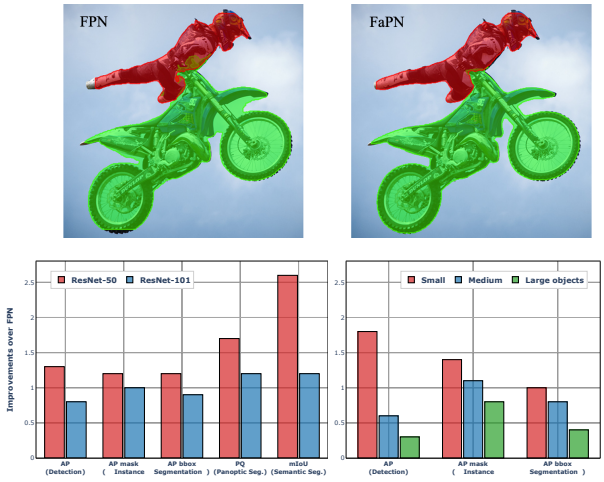

Figure 1: **Comparisons between FPN and FaPN**: (*Top row*) Qualitatively, FaPN significantly improves the performance on object boundaries as opposed to its counterpart, *i.e*. FPN [23]. (*Bottom row*) Quantitatively, FaPN's improvements over FPN are consistent across different tasks, backbones, and object scales. Best view in color.

Dense prediction requires both rich spatial details for object location and strong semantics for object classification, which most likely reside at different resolution / scale levels [28]. How to effectively generate a hierarchy of features at different scales becomes one of the key barriers to overcome in handling dense prediction tasks [23]. Broadly speaking, there are two common practices to address this issue. The first kind uses atrous convolutions with different atrous rates to effectively capture long-range information (*i.e*. semantic context) without reducing spatial resolution [4]. The other kind builds a top-down feature pyramid based on the default bottom-top pathway of a ConvNet [2]. More specifically, the (higher-level) spatially coarser feature maps are upsampled before merging with the corresponding feature maps from the bottom-up path-way. However, there are inaccurate correspondences (*i.e. feature misalignment*) between the bottom-up and upsampled features owing to the non-learnable nature of the commonly-used upsampling operations (*e.g*. nearest neighbor) and the re-

*Corresponding author.
†Authors are with Department of Computer Science and Engineering.

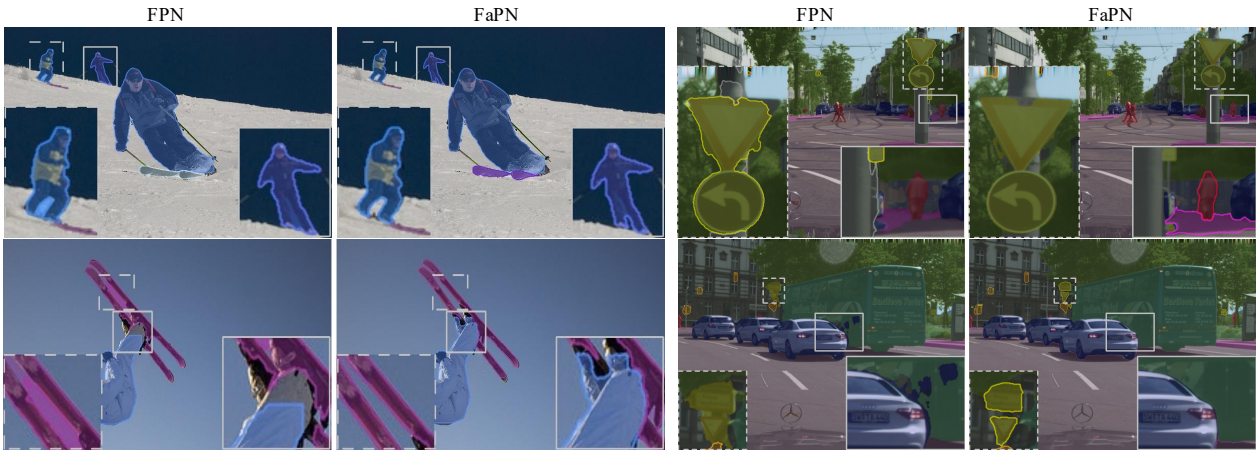

Figure 2: **Example pairs of results from FPN [23] and our FaPN**. Both methods are implemented in Mask R-CNN [13] with ResNet50 [14] being the backbone and PointRend [20] as the mask head. Qualitatively, FaPN significantly improves the performance on object boundaries. Images are randomly chosen from [24] and [9] for instance (*left*) and semantic (*right*) segmentation, respectively. More visualization examples are available in the supplementary materials.

peated applications of downsampling and upsampling. The misaligned features, in turn, adversely affects the learning in the subsequent layers, resulting in mis-classifications in the final predictions, especially around the object boundaries. To address the aforementioned issue, we propose a feature alignment module that learns to align the upsampled feature maps to a set of reference feature maps by adjusting each sampling location in a convolutional kernel with a learned offset. We further propose a feature selection module to adaptively emphasize the bottom-up feature maps containing excessive spatial details for accurate locating. We then integrate these two modules in a top-down pyramidal architecture and propose the *Feature-aligned Pyramid Network (FaPN)*.

Conceptually, FaPN can be easily incorporated to existing bottom-up ConvNet backbones [14, 29, 31, 33] to generate a pyramid of features at multiple scales [23]. We implement FaPN in modern dense prediction frameworks (Faster R-CNN [40], Mask R-CNN [13], PointRend [20], MaskFormer [8], PanopticFPN [18], and PanopticFCN [22]), and demonstrate its efficacy on object detection, semantic, instance and panoptic segmentation. Extensive evaluations on multiple challenging datasets suggest that FaPN leads to a significant improvement in dense prediction performance, especially for small objects and on object boundaries. Moreover, FaPN can also be easily extended to real-time semantic segmentation by pairing it with a lightweight bottom-up backbone [14, 30, 32]. Without bells and whistles, FaPN achieves favorable performance against existing dedicated real-time methods. Our key contributions are:

– We first develop (i) a feature alignment module that learns transformation offsets of pixels to contextually align upsampled (higher-level) features; and (ii) another feature selection module to emphasize (lower-level) features with rich spatial details.

– With the integration of these two contributions, we present, *Feature-aligned Pyramid Network (FaPN)*, an enhanced drop-in replacement of FPN [23], for generating multi-scale features.

– We present a thorough experimental evaluation demonstrating the efficacy and value of each component of FaPN across *four* dense prediction tasks, including object detection, semantic, instance, and panoptic segmentation on three benchmark datasets, including MS COCO [24], Cityscapes [9], COCO-Stuff-10K [3].

– Empirically, we demonstrate that our FaPN leads to a significant improvement of **1.2% - 2.6%** in performance (AP / mIoU) over the original FPN [23]. Furthermore, our FaPN achieves the state-of-the-art of 56.7% mIoU on ADE20K when integrated within MaskFormer [8].

2. Related Work

Feature Pyramid Network Backbone: The existing dense image prediction methods can be broadly divided into two groups. The first group utilizes atrous convolutions to enlarge the receptive field of convolutional filters for capturing long-range information without reducing resolutions spatially. DeepLab [4] is one of the earliest method that adopt atrous convolution for semantic segmentation. It introduced an Atrous Spatial Pyramid Pooling module (ASPP) comprised of atrous convolutions with different atrous rates to aggregate multi-scale context from high-resolution feature maps. Building upon ASPP, a family of methods [4–6] were

developed. However, the lack of the ability to generate feature maps at multiple scales restricts the application of this type of methods to other dense prediction tasks beyond semantic segmentation. The second group of methods focuses on building an encoder-decoder network, *i.e*. bottom-up and top-down pathways. The top-down pathway is used to back-propagate the high-level semantic context into the low-level features via a step-by-step upsampling. There is a plethora of encoder-decoder methods [12, 13, 18, 23, 37, 43, 45] proposed for different dense image prediction tasks. DeconvNet [37] is one of the earliest works that proposed to use upsample operations with learnable parameters, *i.e*. deconvolution. DSSD [12] and FPN [23] are the extensions of SSD [26] and Faster R-CNN [40] respectively for object detection. Mask R-CNN [13] and SOLOs [43, 45] are used for real-time instance segmentation. Moreover, Kirillov *et al*. propose the Panoptic FPN [18] for panoptic segmentation.

Feature Alignment: In case of the increasing loss of boundary detail with the step-by-step downsampling, SegNet [2] stores the max-pooling indices in its encoder and upsamples feature maps in the decoder with the corresponding stored max-pooling indices. Instead of memorizing the spatial information in the encoder previously as SegNet, GUN [34] tries to learn the guidance offsets before upsampling in the decoder and then upsamples feature maps following those offsets. To solve the misalignment between extracted features and the RoI caused by the quantizations in RoIPool, RoIAlign [13] avoids any quantizations and computes the values for each RoI with linear interpolation. To establish accurate correspondences among multiple frames given a large motion for video restoration, TDAN [41] and EDVR [42] achieve implicit motion compensation with by deformable convolution [10] at the feature level. AlignSeg [17] and SFNet [21] are two concurrent works that share a similar motivation as ours and both are flow-based alignment methods. In particular, AlignSeg proposes a two-branched bottom-up network and uses two types of alignment modules to alleviate the feature misalignment before feature aggregation. In contrast, we propose to construct a top-down pathway based on the bottom-up network and align features from the coarsest resolution (top) to the finest resolution (bottom) in a progressive way. Specifically, we only align $2\times$ upsampled features to their corresponding bottom-up features, while AlignSeg tries to align diversely scaled features (*i.e*. upsampled from $1/4$, $1/8$, and even $1/16$) directly which are difficult and may not always be feasible.

3. Feature-aligned Pyramid Network

In this section, we present the general framework of our method, comprised of a Feature Selection Module (FSM) and a Feature Alignment Module (FAM), as shown in Figure 3 (*right*). Specifically, we define the output of the i-th stage of the bottom-up network as $\mathbf{C}_i$, which has stride of 2^i pixels with respect to the input image, *i.e*. $\mathbf{C}_i \in \mathbb{R}^{\frac{H}{2^i}\times\frac{W}{2^i}}$, where $H \times W$ is the size of the input image. And we denote $(\frac{H}{2^i}, \frac{W}{2^i})$ by (H_i, W_i) for brevity. We use $\hat{\mathbf{C}}_i$ to denote the output of a FSM layer given the input of $\mathbf{C}_i$. Also, the output after the i-th feature fusion in the top-down pathway is defined as $\mathbf{P}_i$, and its upsampled and aligned features to $\mathbf{C}_{i-1}$ as $\mathbf{P}_i^u$ and $\hat{\mathbf{P}}_i^u$, respectively.

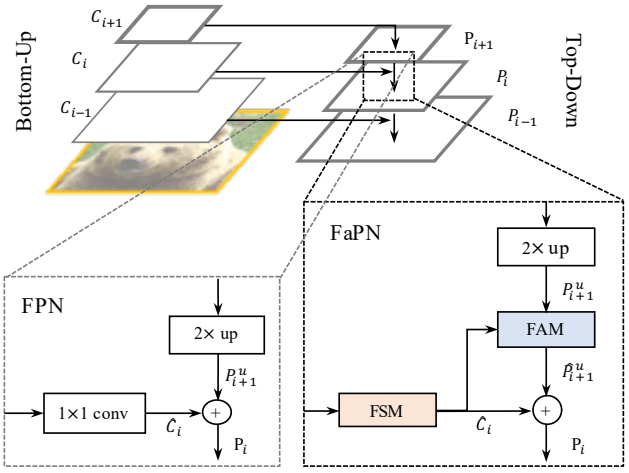

Figure 3: **Overview comparison between FPN and FaPN**. Details of the FAM and FSM components are provided in Figure 4 and Figure 5, respectively.

3.1. Feature Alignment Module

Due to the recursive use of downsampling operations, there are foreseeable spatial misalignment between the upsampled feature maps $\mathbf{P}_i^u$ and the corresponding bottom-up feature maps $\mathbf{C}_{i-1}$. Thus, the feature fusion by either element-wise addition or channel-wise concatenation would harm the prediction around object boundaries. Prior to feature aggregation, aligning $\mathbf{P}_i^u$ to its reference $\hat{\mathbf{C}}_{i-1}$ is essential, *i.e*. adjusting $\mathbf{P}_i^u$ accordingly to the spatial location information provided by the $\hat{\mathbf{C}}_{i-1}$. In this work, the spatial location information is presented by 2D feature maps, where each offset value can be viewed as the shifted distances in 2D space between each point in $\mathbf{P}_i^u$ and its corresponding point in $\hat{\mathbf{C}}_{i-1}$. As illustrated by Figure 4, the feature alignment can be mathematically formulated as:

$$\begin{aligned} \hat{\mathbf{P}}_i^u &= f_a\big(\mathbf{P}_i^u, \mathbf{\Delta}_i\big), \\ \mathbf{\Delta}_i &= f_o\big([\hat{\mathbf{C}}_{i-1}, \mathbf{P}_i^u]\big), \end{aligned} \tag{1}$$

where $[\hat{\mathbf{C}}_{i-1}, \mathbf{P}_i^u]$ is the concatenation of $\hat{\mathbf{C}}_{i-1}$ and $\mathbf{P}_i^u$ which provides spatial difference between the upsampled and corresponding bottom-up features. $f_o(\cdot)$ and $f_a(\cdot)$ denote the functions for learning offsets ($\mathbf{\Delta_i}$) from the spatial

differences and aligning feature with the learned offsets, respectively. In this work, $f_a(\cdot)$ and $f_o(\cdot)$ are implemented using deformable convolutions [10, 52], followed by activation and standard convolutions of the same kernel size.

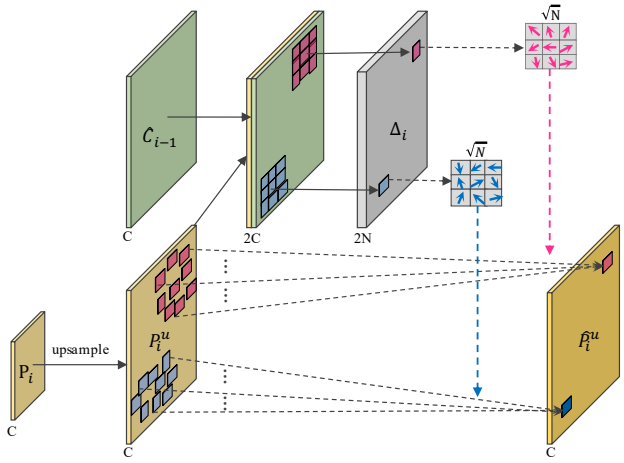

Figure 4: **Feature Alignment Module**. The offset fields have the same spatial resolution with the input and $2N$ channels corresponding to N 2D offsets. Specifically, N denotes a convolutional kernel of N sample locations, *e.g.* N is equal to 9 for a 3×3 conv, and each value in the n-th offset filed is the horizontal or vertical offset for the n-th sample point.

Here, we briefly review the deformable convolution [10], and then explain why it can be used as our feature alignment function and provide some important implementation details. We first define an input feature map $\mathbf{c}_i \in \mathbb{R}^{H_i \times W_i}$ and a $k \times k$ conv layer. Then, the output feature at any position $\hat{x}_{\mathbf{p}}$* after the convolutional kernel can be obtained by

$$\hat{x}_{\mathbf{p}} = \sum_{n=1}^{N} w_n \cdot x_{\mathbf{p}+\mathbf{p}_n}, \tag{2}$$

where N is the size of the $k \times k$ convolutional layer (*i.e.* $N = k \times k$), w_n and $\mathbf{p}_n \in \{(-\lfloor \frac{k}{2} \rfloor, -\lfloor \frac{k}{2} \rfloor), (-\lfloor \frac{k}{2} \rfloor, 0), \ldots, (\lfloor \frac{k}{2} \rfloor, \lfloor \frac{k}{2} \rfloor)\}$ refer to the weight and the pre-specified offset for the n-th convolutional sample location, respectively. In addition to the pre-specified offsets, the deformable convolution tries to learn additional offsets $\{\Delta\mathbf{p}_1, \Delta\mathbf{p}_2, ..., \Delta\mathbf{p}_N\}$ adaptively for different sample locations, and Equation (2) can be reformulated as

$$\hat{x}_{\mathbf{p}} = \sum_{n=1}^{N} w_n \cdot x_{\mathbf{p}+\mathbf{p}_n+\Delta\mathbf{p}_n}, \tag{3}$$

where each $\Delta\mathbf{p}_n$ is a tuple (h, w), with $h \in (-H_i, H_i)$ and $w \in (-W_i, W_i)$.

*where $\mathbf{p} \in \{(0,0), (1,0), (0,1), \ldots, (H_i - 1, W_i - 1)\}$

When we apply the deformable convolution over the $\mathbf{P}_i^u$ and take the concatenation of $\hat{\mathbf{C}}_{i-1}$ and $\mathbf{P}_i^u$ as the reference (*i.e.* offset fields $\mathbf{\Delta}_i = f_o([\hat{\mathbf{C}}_{i-1}, \mathbf{P}_i^u])$), the deformable convolution can adjust its convolutional sample locations following the offsets following Equation (1)†, *i.e.* aligning $\mathbf{P}_i^u$ according to the spatial distance between $\hat{\mathbf{C}}_{i-1}$ and $\mathbf{P}_i^u$.

3.2. Feature Selection Module

Prior to channel reduction for detailed features, it is vital to emphasize the important feature maps that contain excessive spatial details for accurate allocations while suppressing redundant feature maps. Instead of simply using a 1×1 convolution [23], we propose a feature selection module (FSM) to explicitly model the importance of feature maps and re-calibrate them accordingly.

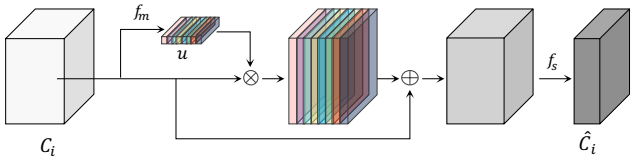

Figure 5: **Feature Selection Module**. $\mathbf{C}_i = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_D]$ and $\hat{\mathbf{C}}_i = [\hat{\mathbf{c}}_1, \hat{\mathbf{c}}_2, \ldots, \hat{\mathbf{c}}_{D'}]$ refer to the input and output feature maps respectively, where $\mathbf{c_d}$ and $\hat{\mathbf{c}}_{\mathbf{d'}} \in \mathbb{R}^{H_i \times W_i}$, D and D' denote the input and output channels, respectively. $\mathbf{u} = [u_1, u_2, \ldots, u_D]$ is the feature importance vector, where u_d represents the importance of the d-th input feature map. f_m and f_s represent the feature importance modeling and feature selection layer, respectively. See text for details.

The general dataflow of the proposed FSM is presented in Figure 5. To begin with, the global information $\mathbf{z}_i$ of each input feature map $\mathbf{c}_i$ is extracted by a global average pooling operation, while a feature importance modeling layer $f_m(\cdot)$ (*i.e.* a 1×1 conv layer followed by a sigmoid activation function) learns to use such information for modeling the importance of each feature map and outputs an importance vector $\mathbf{u}$. Next, the original input feature maps are scaled with the importance vector, and then the scaled feature maps are added to the original feature maps, referred as rescaled feature maps. Finally, a feature selection layer $f_s(\cdot)$ (*i.e.* a 1×1 conv layer for efficiency) is introduced over the rescaled feature maps, which is used to selectively maintain important feature maps and drop useless feature maps for channel reduction. Overall, the process of FSM can be formulated as

$$\begin{aligned} \hat{\mathbf{C}}_i &= f_s(\mathbf{C}_i + \mathbf{u} * \mathbf{C}_i), \\ \mathbf{u} &= f_m(\mathbf{z}), \end{aligned} \tag{4}$$

†Following the convention of the deformable convolution, this study adopts 3×3 as the kernal size for $f_a(\cdot)$ and $f_o(\cdot)$.

where $\mathbf{z} = [z_1, z_2, \ldots, z_D]$ and is calculated by

$$z_d = \frac{1}{H_i \times W_i} \sum_{h=1}^{H_i} \sum_{w=1}^{W_i} c_d(h, w). \quad (5)$$

It is worth mentioning that the design of our FSM is motivated by the squeeze-and-excitation (SE) [16]. The main difference lies in the additional skip connection introduced between the input and scaled feature maps (Figure 5). Empirically, we find that lower bounding the scaled feature (through the skip connection) is essential, which avoids any particular channel responses to be over-amplified or -suppressed. Conceptually, both of these two modules learn to adaptively re-calibrate channel-wise responses by channel attention. However, SE is conventionally used in the backbone for enhancing feature extraction, while FSM is used in the neck (*i.e.* top-down pathway) for enhancing multi-scale feature aggregation. Additionally, the selected/scaled features from FSM are also supplied as references to FAM for learning alignment offsets.

4. Experiments

In this section, we first briefly introduce the benchmark datasets studied in this work, followed by the implementation and training details. We then evaluate the performance of the proposed FaPN on four dense image prediction tasks, including object detection, semantic, instance and panoptic segmentation. Ablation studies demonstrating the effectiveness of each component in FaPN are also provided. Moreover, we incorporate our proposed FaPN with lightweight backbones and evaluate its efficacy under real-time settings.

Datasets: We consider four widely-used benchmark datasets to evaluate our method, including MS COCO [24] for object detection, instance and panoptic segmentation; Cityscapes [9], COCO-Stuff-10K [3] and ADE20K [51] for semantic segmentation.

MS COCO consists of more than 100K images containing diverse objects and annotations, including both bounding boxes and segmentation masks. We use the *train2017* set (around 118K images) for training and report results on the *val2017* set (5K images) for comparison. For both object detection and instance segmentation tasks, there are 80 categories; and for panoptic segmentation task, there are 80 things and 53 stuff classes annotated.

Cityscapes is a large-scale dataset for semantic understanding of urban street scenes. It is split into training, validation and test sets, with 2975, 500 and 1525 images, respectively. The annotation includes 30 classes, 19 of which are used for semantic segmentation task. The images in this dataset have a higher and unified resolution of 1024×2048, which poses stiff challenges to the task of real-time semantic segmentation. For the experiments shown in this part, we only use images with fine annotations to train and validate our proposed method.

COCO-Stuff-10K contains a subset of 10K images from the COCO dataset [24] with dense stuff annotations. It is a challenging dataset for semantic segmentation as it has 182 categories (91 thing classes plus 91 stuff classes). In this work, we follow the official split – 9K images for training and 1K images for test.

ADE20K is a challenging scene parsing dataset that contains 20k images for training and 2k images for validation. Images in the dataset are densely labeled as hundreds of classes. In this work, only 150 semantic categories are selected to be included in the evaluation.

Implementation details: Following the original work of FPN [23], we use ResNets [15] pre-trained on ImageNet [11] as the backbone ConvNets for the bottom-up pathway. We then replace the FPN with our proposed FaPN as the top-down pathway network. Next, we connect the feature pyramid with the Faster R-CNN detector [40] for object detection, and Mask R-CNN (with PointRend masking head [20]) for segmentation tasks.

For performance evaluation, the Average Precision (AP) is used as the primary metric for both object detection and instance segmentation. We evaluate AP on small, medium and large objects, *i.e.* AP_s, AP_m, and AP_l. Note that AP^{bb} and AP^{mask} denote AP for bounding box and segmentation mask, respectively. The mean Intersection-over-Union (mIoU) and the Panoptic Quality (PQ) are two primary metrics used for semantic and panoptic segmentation, respectively. Additionally, we also use PQ^{St} and PQ^{Th} metrics to evaluate stuff and thing performances separately for panoptic segmentation.

4.1. Ablation Study

We first breakdown the individual impacts of the two components introduced in FaPN, *i.e.* the feature alignment and selection modules. Using ResNet50 as the bottom-up backbone, we evaluate on Cityscapes for semantic segmentation. Table 1 shows the improvement in accuracy along with the complexity overheads measured in #Params.

Evidently, with marginal increments in model size, our proposed feature alignment module alone significantly boosts the performance of the original FPN [23], yielding an improvement of **2.3 points** in mIoU. In particular, our method (*80.0@33.1M*) is significantly more effective than naively expanding either i) the #Params of FPN by extra 3×3 conv. (*77.5@33.4M*) or ii) the capacity of the backbone from R50 to R101 (*78.9@47.6M*). Empirically, we observe that a naive application of SE [16] (for feature selection) adversely affects the performance, while our proposed FSM provides a further boost in mIoU.

Recall that the *misalignment* in this work refers to the spatial misalignment of features induced during the ag-

Table 1: **Ablative Analysis:** Comparing the performance of our FaPN with other variants on Cityscapes for semantic segmentation. † denotes placing FAM after feature fusion. "deconv" refers to the deconvolution which is a learnable upsample operation. The relative improvements/overheads are shown in parenthesis.

method	backbone	#Params (M)	mIoU (%)
FPN	R50	28.6 (+4.5)	77.4 (+2.6)
FPN + extra 3×3 conv.	R50	33.4 (-0.3)	77.5 (+2.5)
FPN	R101	47.6 (-14.5)	78.9 (+1.1)
FPN + FAM	R50	31.7 (+1.4)	79.7 (+0.3)
FPN + FAM + SE	R50	33.1 (+0.0)	78.8 (+1.2)
FPN + FAM + FSM (FaPN)	R50	33.1 (+0.0)	**80.0** (+0.0)
FPN + deconv + FSM	R50	32.7 (+0.4)	76.7 (+3.3)
FPN + FAM† + FSM	R50	32.7 (+0.4)	79.3 (+0.7)

gregation of multi-resolution feature maps (i.e., top-down pathway in FPN), particularly around object boundaries. One plausible cause relates to the non-learnable nature of commonly-used upsampling operations (e.g., bilinear). However, simply swapping it to a learnable operation (e.g., deconvolution) is insufficient, suggesting the need of better engineered methods. This reinforces the motivation of this work. Instead of performing the feature alignment before feature fusion, we place our FAM after feature fusion, in which our FAM learns the offsets from the fused features instead. Although this variation performs better than all other variants, it is still substantially worse than the proposed FaPN, which reiterate the necessity of feature alignment before fusion.

4.2. Boundary Prediction Analysis

We provide the mIoU over the boundary pixels[‡] in Table 2. Evidently, our method achieves a substantially better segmentation performance than FPN on boundaries. Moreover, we visualize the input (upsampled features P_2^u) to and the output (aligned features $\hat{P}_2^u$) from the last feature alignment module in FaPN-R50 (Figure 6) to perceive the alignment corrections made by our FAM. In contrast to the raw upsampled features (before FAM) which are noisy and fluctuating, the aligned features are smooth and containing more precise object boundaries. Both the quantitative evaluation and qualitative observation are consistent and suggest that FaPN leads to better predictions on the boundaries. More visualizations are provided in Figure 2.

4.3. Main Results

In this section, we present the detailed empirical comparisons to FPN [23] on four dense prediction tasks, including object detection, semantic, instance and panoptic segmentation in Table 3 - 6, respectively.

[‡]we consider n pixels around the outline of each object to be boundary pixels, where n can be one of [3, 5, 8, 12].

Table 2: **Segmentation Performance around Boundaries:** Comparing the performance of our FaPN with the original FPN [23] in terms of mIoU over boundary pixels on Cityscape *val* with different thresholds on boundary pixels.

method	backbone	3px	5px	8px	12px	mean
FPN	PointRend [20] R50	46.9	53.6	59.3	63.8	55.9
FaPN		49.2	56.2	62.0	66.4	58.5
improvement		***(+2.3)***	***(+2.6)***	***(+2.7)***	***(+2.6)***	***(+2.6)***
FPN	PointRend [20] R101	47.8	54.6	60.5	64.9	57.0
FaPN		50.1	57.1	62.9	67.2	59.3
improvement		***(+2.3)***	***(+2.5)***	***(+2.4)***	***(+2.3)***	***(+2.3)***

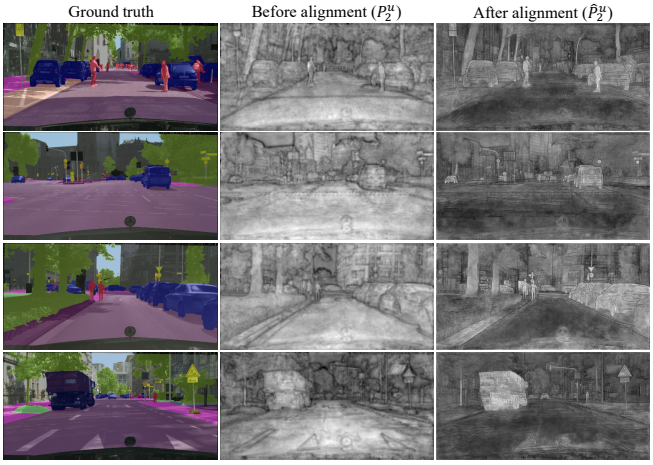

Figure 6: Visualization of the input (upsampled features) to and the output (aligned features) from our FAM. Zoom in for better details.

In general, FaPN substantially outperforms FPN on all scenarios of tasks and datasets. There are several detailed observations. First, FaPN improves the primary evaluation metrics by **1.2 - 2.6 points** over FPN on all four tasks with ResNet50 [15] as the bottom-up backbone. Second, the improvements brought by FaPN hold for stronger bottom-top backbones (e.g. ResNet101 [15]) with a longer training schedule of 270K iterations. Third, the improvement from FaPN extends to more sophisticated mask heads, *e.g.* PointRend [20], on instance segmentation, as shown in Table 5 (bottom section).

Table 3: **Object Detection:** Performance comparisons on MS COCO *val* set between FPN and FaPN.

method	backbone	AP^{bb}	AP^{bb}_s	AP^{bb}_m	AP^{bb}_l
FPN	Faster R-CNN [40] R50	37.9	22.4	41.1	49.1
FaPN (ours)		39.2	24.5	43.3	49.1
improvement		***(+1.3)***	***(+2.1)***	***(+2.2)***	*(+0.0)*
FPN	Faster R-CNN [40] R101	42.0	25.2	45.6	54.6
FaPN (ours)		42.8	27.0	46.2	54.9
improvement		***(+0.8)***	***(+1.8)***	***(+0.6)***	***(+0.3)***

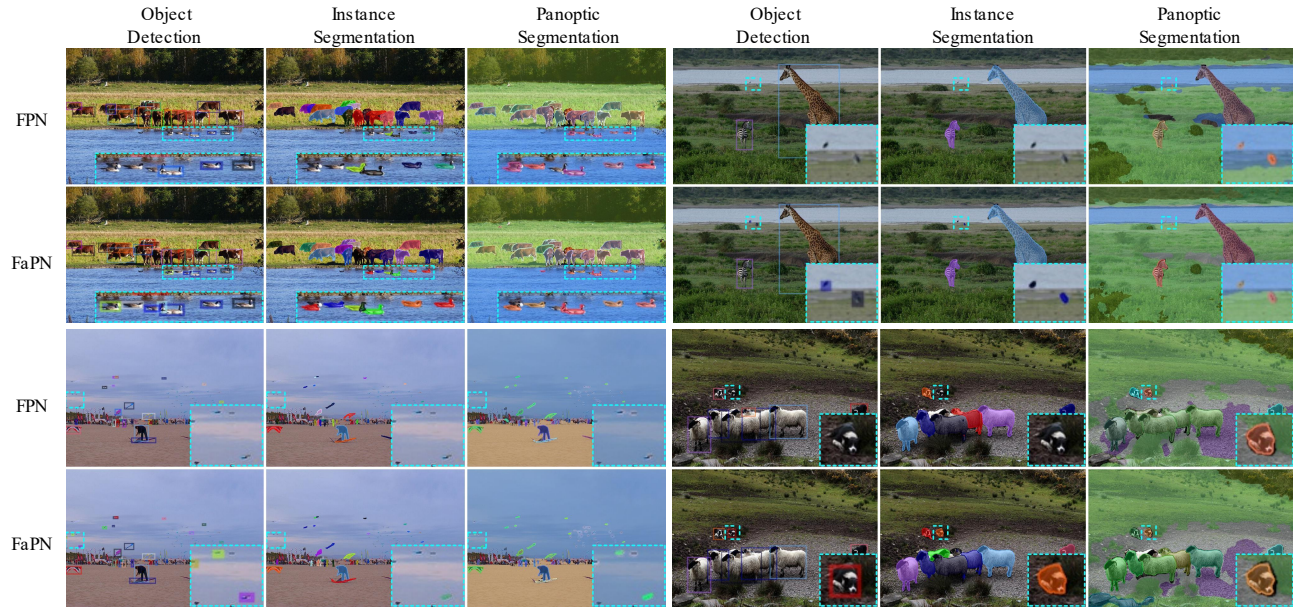

Figure 7: **Example pairs of results comparing FPN [23] and our FaPN**. Images are randomly chosen from [24]. Best view in color and zoom in for details.

Table 4: **Semantic Segmentation:** Performance comparisons on Cityscapes *val* set between FPN and FaPN.

method	backbone	mIoU	iIoU	IoU_sup	iIoU_sup
FPN	PointRend [20] R50	77.4	58.5	89.9	76.9
FaPN (ours)		80.0	61.3	90.6	78.5
improvement		***(+2.6)***	***(+2.8)***	***(+0.7)***	***(+1.6)***
FPN	PointRend [20] R101	78.9	59.9	90.4	77.8
FaPN (ours)		80.1	62.2	90.8	78.6
improvement		***(+1.2)***	***(+2.3)***	***(+0.4)***	***(+0.8)***

Table 5: **Instance Segmentation:** Performance comparisons on MS COCO *val* set between FPN and FaPN.

method	backbone	AP^{mask}	AP^{mask}_s	AP^{bb}	AP^{bb}_s
FPN	Mask R-CNN [13] R50	35.2	17.1	38.6	22.5
FaPN (ours)		36.4	18.1	39.8	24.3
improvement		***(+1.2)***	***(+1.0)***	***(+1.2)***	***(+1.8)***
FPN	Mask R-CNN [13] R101	38.6	19.5	42.9	26.4
FaPN (ours)		39.6	20.9	43.8	27.4
improvement		***(+1.0)***	***(+1.4)***	***(+0.9)***	***(+1.0)***
FPN	PointRend [20] R50	36.2	17.1	38.3	22.3
FaPN + PR (ours)		37.6	18.6	39.4	24.2
improvement		***(+1.4)***	***(+1.5)***	***(+1.1)***	***(+1.9)***

In particular, we notice that the improvement is larger on small objects (e.g. AP^{bb}_s, AP^{mask}_s). For instance, FaPN improves the bounding box AP on small objects by **2.1 points** and **1.8 points** over FPN on MS COCO object detection and instance segmentation, respectively. Conceptually, small objects occupy fewer pixels in an image, and most of pixels are distributed along the object boundaries.

Table 6: **Panoptic Segmentation:** Performance comparisons on MS COCO *val* set between FPN and FaPN.

method	backbone	PQ	mIoU	PQ^{St}	AP^{bb}	PQ^{Th}
FPN	PanopticFPN [18] R50	39.4	41.2	29.5	37.6	45.9
FaPN (ours)		41.1	43.4	32.5	38.7	46.9
improvement		***(+1.7)***	***(+2.2)***	***(+3.0)***	***(+0.9)***	***(+1.0)***
FPN	PanopticFPN [18] R101	43.0	44.5	32.9	42.4	49.7
FaPN (ours)		44.2	45.7	35.0	43.0	50.3
improvement		***(+1.2)***	***(+1.2)***	***(+2.1)***	***(+0.6)***	***(+0.6)***
FPN	PanopticFCN [22] R50	41.1	79.8	49.9	30.2	41.4
FaPN (ours)		41.8	80.2	50.5	30.8	42.0
improvement		***(+0.7)***	***(+0.4)***	***(+0.6)***	***(+0.6)***	***(+0.6)***
FPN	PanopticFCN [22] R50-600	42.7	80.8	51.4	31.6	43.9
FaPN (ours)		43.5	81.3	52.1	32.3	53.5
improvement		***(+0.8)***	***(+0.5)***	***(+0.7)***	***(+0.7)***	***(+0.6)***

Hence, it is vital to be able to correctly classify the boundaries for small objects. However, as features traverse the top-bottom pathway through heuristics-based upsampling operations (*e.g.* FPN uses nearest neighbor upsampling), shifts in pixels (*i.e.* misalignment) are foreseeable and the amount of shifts will accumulate as the number of upsampling steps increases. Hereby, the severity of the misalignment will reach its maximum at the finest feature maps in the top-down pathway pyramid, which are typically used for detecting or segmenting small objects, resulting in a significant degradation in performance. On the other hand, FaPN performs feature alignment progressively which in turn alleviates the misalignment at the finest level step by step, and thus achieves significant improvements on small

objects compared to the FPN [23]. Qualitative improvements are also evidenced in Figure 7. Finally, we incorporate FaPN into MaskFormer [8], and demonstrate that FaPN leads to state-of-the-art performance on two complex semantic segmentation tasks, *i.e*. ADE20K and COCO-Stuff-10K, as shown in Table 7.

Table 7: **Comparison to SOTA** on (a) ADE20K `val` and (b) COCO-Stuff-10K `test`. We report both single-scale (s.s.) and multi-scale (m.s.) semantic segmentation performance. Backbones pre-trained on ImageNet-22K are marked with †. Our results are highlighted in shade.

(a) ADE20K `val`

method	backbone	crop size	mIoU (s.s.)	mIoU (m.s.)
OCRNet [48]	R101	520 × 520	-	45.3
AlignSeg [17]	R101	512 × 512	-	46.0
SETR [50]	ViT-L†	512 × 512	-	50.3
Swin-UperNet [27]	Swin-L†	640 × 640	-	53.5
MaskFormer [8]	Swin-L†	640 × 640	54.1	55.6
MaskFormer + FaPN	Swin-L†	640 × 640	**55.2**	**56.7**

(b) COCO-Stuff-10K `test`

method	backbone	crop size	mIoU (s.s.)	mIoU (m.s.)
OCRNet [48]		520 × 520	-	39.5
MaskFormer [8]	R101	640 × 640	38.1	39.8
MaskFormer + FaPN		640 × 640	**39.6**	**40.6**

Overall, extensive comparisons on scenarios comprised of different tasks and datasets have confirmed the effectiveness of our proposed FaPN for dense image prediction. A straightforward replacement of FPN with FaPN achieves substantial improvements without bells and whistles. The generality and flexibility to different bottom-up backbones or mask heads have further strengthened the practical utilities of FaPN.

4.4. Real-time Performance

Driven by real-world applications (*e.g*., autonomous driving), there are growing interests in real-time dense prediction, which requires the generation of high-quality predictions with minimal latency. In this section, we aim to investigate the effectiveness of our proposed FaPN under real-time settings, *i.e*. inference speed $\geq$ 30 FPS. The full details are provided in the supplementary materials.

We compare our FaPN with state-of-the-art real-time semantic segmentation methods on Cityscapes and COCO-Stuff-10K in Table 8, in terms of accuracy (mIoU) and inference speed (FPS). In general, we observe that a straightforward replacement of FPN with the proposed FaPN results in a competitive baseline against other dedicated real-time semantic segmentation methods.

In particular, on Cityspaces, FaPN-R18 runs 2× faster than SwiftNet [38], while maintaining a similar mIoU performance. In addition, with a larger backbone and input size, FaPN-R34 achieves a competitive mIoU of 78.1 points on the *test* split, in the same time outputting 30 FPS. On the more challenging COCO-Stuff-10K, our FaPN also outperforms other existing methods by a substantial margin. Specifically, FaPN-R34 outperforms BiSeNetV2 [46] in both segmentation accuracy measured in mIoU and inference speed.

Table 8: **Real-time semantic segmentation** on (a) Cityscapes and (b) COCO-Stuff-10K. † denotes a method with a customized backbone. Our results are highlighted in shade.

(a) Cityscapes

method	backbone	crop size	FPS	mIoU (*val*)	mIoU (*test*)
ESPNet [35]	†	512 × 1024	113	-	60.3
ESPNetV2 [36]	†	512 × 1024	-	66.4	66.2
FaPN	R18	512 × 1024	**142**	**69.2**	**68.8**
BiSeNet [47]	R18	768 × 1536	65.6	74.8	74.7
FaPN	R18	768 × 1536	**78.1**	**75.6**	**75.0**
SwiftNet [38]	R18	1024 × 2048	**39.9**	75.4	75.5
ICNet [49]	R50	1024 × 2048	30.3	-	69.5
FAPN	R34	1024 × 2048	30.2	**78.5**	**78.1**

(b) COCO-Stuff-10K

method	backbone	crop size	FPS	mIoU (*val*)
BiSeNet [47]	R18		-	28.1
BiSeNetV2 [46]	†		42.5	28.7
ICNet [49]	R50	640 × 640	35.7	29.1
FaPN	R18		**154**	28.4
FaPN	R34		110	**30.3**

5. Conclusion

This paper introduced *Feature-aligned Pyramid Network (FaPN)*, a simple yet effective top-down pyramidal architecture to generate multi-scale features for dense image prediction. It is comprised of a feature alignment module that learns transformation offsets of pixels to contextually align upsampled higher-level features; and a feature selection module to emphasize the lower-level features with rich spatial details. Empirically, FaPN leads to substantial and consistent improvements over the original FPN on four dense prediction tasks and three datasets. Furthermore, FaPN improves the state-of-the-art segmentation performance when integrated in strong baselines. Additionally, FaPN can be easily extended to real-time segmentation tasks by pairing it with lightweight backbones, where we demonstrate that FaPN performs favorably against dedicated real-time methods. In short, given the promising performance on top of a simple implementation, we believe that FaPN can serve as the new baseline/module for dense image prediction.

Acknowledgments: This work was supported by the National Natural Science Foundation of China (Grant No. 61903178, 61906081, and U20A20306) and the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (Grant No. 2017ZT07X386).

References

[1] Hassan Abu Alhaija, Siva Karthik Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *IJCV*, 2018. 1

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *TPAMI*, 2017. 1, 3

[3] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. COCO-Stuff: Thing and stuff classes in context. In *CVPR*, 2018. 2, 5

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2017. 1, 2

[5] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv:1706.05587*, 2017. 2

[6] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 2

[7] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *CVPR*, 2016. 1

[8] Bowen Cheng, Alexander G Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. *arXiv preprint arXiv:2107.06278*, 2021. 2, 8

[9] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 2, 5

[10] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017. 3, 4

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 5

[12] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C Berg. Dssd: Deconvolutional single shot detector. *arXiv:1701.06659*, 2017. 3

[13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 1, 2, 3, 7

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5, 6

[16] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018. 5

[17] Zilong Huang, Yunchao Wei, Xinggang Wang, Humphrey Shi, Wenyu Liu, and Thomas S Huang. Alignseg: Feature-aligned segmentation networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 3, 8

[18] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *CVPR*, 2019. 1, 2, 3, 7

[19] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *CVPR*, 2019. 1

[20] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. PointRend: Image segmentation as rendering. In *CVPR*, 2020. 2, 5, 6, 7

[21] Xiangtai Li, Ansheng You, Zhen Zhu, Houlong Zhao, Maoke Yang, Kuiyuan Yang, Shaohua Tan, and Yunhai Tong. Semantic flow for fast and accurate scene parsing. In *ECCV*, 2020. 3

[22] Yanwei Li, Hengshuang Zhao, Xiaojuan Qi, Liwei Wang, Zeming Li, Jian Sun, and Jiaya Jia. Fully convolutional networks for panoptic segmentation. In *CVPR*, 2021. 2, 7

[23] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 1, 2, 3, 4, 5, 6, 7, 8

[24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 2, 5, 7

[25] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *CVPR*, 2018. 1

[26] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 1, 3

[27] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021. 8

[28] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1

[29] Zhichao Lu, Kalyanmoy Deb, and Vishnu N. Boddeti. MUXConv: Information multiplexing in convolutional neural networks. In *CVPR*, 2020. 2

[30] Zhichao Lu, Kalyanmoy Deb, Erik Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. NSGANetV2: Evolutionary multi-objective surrogate-assisted neural architecture search. In *ECCV*, 2020. 2

[31] Zhichao Lu, Gautam Sreekumar, Erik Goodman, Wolfgang Banzhaf, Kalyanmoy Deb, and Vishnu N. Boddeti. Neural architecture transfer. *TPAMI*, 2021. 2

[32] Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, and Wolfgang Banzhaf. NSGA-Net: Neural architecture search using multi-objective genetic algorithm. In *GECCO*, 2019. 2

[33] Zhichao Lu, Ian Whalen, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, Wolfgang Banzhaf, and Vishnu N. Boddeti. Multiobjective evolutionary design of deep convolutional neural networks for image classification. *TEVC*, 2021. 2

[34] Davide Mazzini. Guided upsampling network for real-time semantic segmentation. *arXiv:1807.07466*, 2018. 3

[35] Sachin Mehta, Mohammad Rastegari, Anat Caspi, Linda Shapiro, and Hannaneh Hajishirzi. Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In *ECCV*, 2018. 8

[36] Sachin Mehta, Mohammad Rastegari, Linda Shapiro, and Hannaneh Hajishirzi. Espnetv2: A light-weight, power efficient, and general purpose convolutional neural network. In *CVPR*, 2019. 8

[37] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015. 3

[38] Marin Orsic, Ivan Kreso, Petra Bevandic, and Sinisa Segvic. In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images. In *CVPR*, 2019. 8

[39] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 1

[40] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 1, 2, 3, 5, 6

[41] Yapeng Tian, Yulun Zhang, Yun Fu, and Chenliang Xu. Tdan: Temporally-deformable alignment network for video super-resolution. In *CVPR*, 2020. 3

[42] Xintao Wang, Kelvin CK Chan, Ke Yu, Chao Dong, and Chen Change Loy. Edvr: Video restoration with enhanced deformable convolutional networks. In *CVPR Workshops*, 2019. 3

[43] Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li. SOLO: Segmenting objects by locations. In *ECCV*, 2020. 3

[44] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *CVPR*, 2017. 1

[45] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic and fast instance segmentation. *NeurIPS*, 2020. 3

[46] Changqian Yu, Changxin Gao, Jingbo Wang, Gang Yu, Chunhua Shen, and Nong Sang. Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation. *arXiv:2004.02147*, 2020. 8

[47] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *ECCV*, 2018. 8

[48] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. In *ECCV*, 2020. 8

[49] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images. In *ECCV*, 2018. 8

[50] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *CVPR*, 2021. 8

[51] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017. 5

[52] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *CVPR*, 2019. 4