MaskConver: Revisiting Pure Convolution Model for Panoptic Segmentation

Abdullah Rashwan
Google
arashwan@google.com

Jiageng Zhang
Google
jiageng@google.com

Ali Taalimi
Google
taalimi@google.com

Fan Yang
Google
fyangf@google.com

Xingyi Zhou
Google
zhouxy@google.com

Chaochao Yan
Google
allenyan@google.com

Liang-Chieh Chen
ByteDance *

Yeqing Li
Google
yeqing@google.com

Abstract

In recent years, transformer-based models have dominated panoptic segmentation, thanks to their strong modeling capabilities and their unified representation for both semantic and instance classes as global binary masks. In this paper, we revisit pure convolution model and propose a novel panoptic architecture named MaskConver. MaskConver proposes to fully unify things and stuff representation by predicting their centers. To that extent, it creates a lightweight class embedding module that can break the ties when multiple centers co-exist in the same location. Furthermore, our study shows that the decoder design is critical in ensuring that the model has sufficient context for accurate detection and segmentation. We introduce a powerful ConvNeXt-UNet decoder that closes the performance gap between convolution- and transformer-based models. With ResNet50 backbone, our MaskConver achieves 53.6% PQ on the COCO panoptic val set, outperforming the modern convolution-based model, Panoptic FCN, by 9.3% as well as transformer-based models such as Mask2Former (+1.7% PQ) and kMaX-DeepLab (+0.6% PQ). Additionally, MaskConver with a MobileNet backbone reaches 37.2% PQ, improving over Panoptic-DeepLab by +6.4% under the same FLOPs/latency constraints. A further optimized version of MaskConver achieves 29.7% PQ, while running in real-time on mobile devices. The code and model weights will be publicly available. [1]

1. Introduction

Panoptic segmentation [38] aims to unify instance [23] and semantic segmentation [25] in the same framework. Existing works propose to merge instance and semantic segmentation outputs using post-processing layers [37, 51, 78, 81]. These architectures however rely on many customized components like non-maximum suppression (NMS), and thing-stuff merging heuristics to produce panoptic outputs. Recent works [11, 12, 47, 72, 82, 83] unify both segmentation tasks by producing binary masks and class scores for both things and stuff classes. Such universal architectures result in a simpler post-processing logic and make the loss closely correlated with the panoptic quality (PQ) metric. As a result, they have achieved significantly higher PQ numbers compared to traditional architectures.

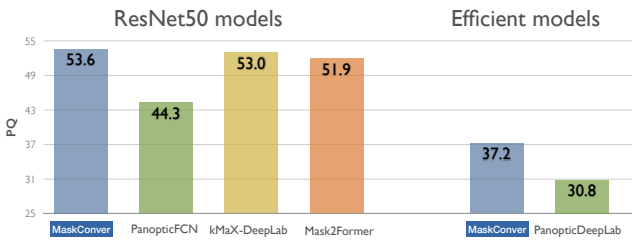

Figure 1. **MaskConver versus existing panoptic models.** MaskConver successfully bridges the gap between the modern convolution-based method, Panoptic FCN, and the transformer-based methods, Mask2Former and kMaX-DeepLab. In the efficient model setting, MaskConver outperforms Panoptic-DeepLab under the same FLOPs/latency constraints.

Among these unified panoptic segmentation models, transformers [71] have played a critical role due to their ability to learn instance-level embeddings via a transformer decoder. DETR [6] was introduced for object detection by learning object embeddings, each of which predicts an object class and a bounding box. The idea of instance-level embeddings influenced many of the transformer-based panoptic models. Following DETR, MaX-DeepLab [72] uses a transformer decoder to learn mask embeddings [33, 67, 76] to predict a set of binary masks. The binary masks

*Work done while at Google.

[1] https://github.com/tensorflow/models/tree/master/official/projects/maskconver.

are then merged using a simple post-processing layer [77] to filter out duplicates, similar to NMS. Other architectures [11, 12, 47, 82, 83] follow a similar paradigm, but further improved the performance by developing modern transformer decoders [91]. The commonality between these methods is the employment of transformer blocks to learn a set of binary masks and their corresponding classes (hence the panoptic masks). On the other hand, the convolution-based methods [10, 46] lag behind in performance. It is yet unclear if using transformers justifies the quality gap compared to convolution-based models.

In this work, we revisit the pure convolution panoptic models [10, 46] and propose a novel architecture for panoptic segmentation (Fig. 2), named MaskConver, which produces segmentation masks for thing and stuff classes in a unified way. The meta architecture of MaskConver contains four main components: backbone, pixel decoder, prediction heads, and mask embedding generator. The backbone is a typical ImageNet [61] pretrained convolutional neural network (ConvNet) [40], such as ResNet [24]. We design a novel pixel-decoder, ConvNeXt-UNet, which deploys ConvNeXt blocks [54] in a manner similar to UNet decoder [60] but in an asymmetric way. Particularly, in the decoder, we discover that it is critical to stack more ConvNeXt blocks at the highest level (*i.e.*, level 5 with stride 32), which benefits the model to effectively learn context information. The prediction heads include three predictions: Center Heatmap Prediction, Center Embedding Prediction, and Mask Feature Prediction. The Center Heatmap prediction predicts the center point heatmaps [90] for *both* things and stuff. We utilize the mask centers instead of box centers to represent both things and stuff. The Center Embedding Head generates the embeddings for the center points, while the Mask Feature Head produces the mask features. Finally, the Mask Embedding Generator aims to generate high-quality mask embeddings by taking into account the "instance collision", where the center points of neighboring instances may collide, yielding the same (indistinguishable) center embeddings. To alleviate the issue, it first produces the class embeddings (via Class Embedding Lookup Table) by taking the predicted semantic classes of the center points. The output mask embeddings are then obtained by modulating the center embeddings with the class embeddings (via addition and MLP) to a different space conditioned on the semantic class of the instance. Finally, the mask embeddings are multiplied with the mask features to produce segmentation masks for things and stuff in a unified way.

We evaluate the quality of MaskConver in several settings on COCO panoptic segmentation dataset [50]. On the COCO validation set, our proposed ConvNeXt-UNet pixel decoder improves the PQ of the solid pixel decoder baseline BiFPN [66] by +3.1%, while being 18% more efficient on FLOPs. When using the ResNet50 backbone [24], MaskConver achieves 53.6% and runs at 19.6 FPS on a V100 GPU. Our model demonstrates significant gains (+9.3%) compared to the modern convolution-based method, Panoptic FCN [46], while also outperforming the transformer-based models like Mask2Former [11] (+1.7%) and kMaX-DeepLab [83] (+0.6%). When using the MobileNet backbone [29], MaskConver achieves 37.2% PQ, which is 6.4% better than Panoptic-DeepLab [10]. In addition, after further optimization via quantization, MaskConver runs at 30 FPS on Pixel 6 GPU, while achieving 29.7% PQ.

2. Related Work

Since the proposal of panoptic segmentation in [38], numerous efforts have emerged in this domain. Initiatives began with adaptations to existing networks, adding either a semantic [37] or an instance branch [10] to state-of-the-art models These methods established a baseline using hand-crafted post-processing layers [37, 51, 78, 81] for final panoptic predictions. Following these works, researchers start to think about architectures that can solve the task in a more unified way. MaX-DeepLab [72] learns mask embeddings [33, 67, 76] in the transformer framework for both thing and stuff classes. These mask embeddings predict a set of binary masks, and a post-processing layer [77] is used to predict the final panoptic outputs. MaskFormer [12] proposes a similar paradigm, and shows how to use the same model for both semantic and panoptic segmentation by only modifying the post-processing logic. Panoptic-Segformer [47] extends Deformable DETR [91] for the panoptic segmentation task. CMT-DeepLab [82] reformulates the transformer cross-attention from the clustering perspective. Mask2Former [11] proposes masked-attention to significantly outperform MaskFormer architecture on smaller objects by masking unrelated parts of the image. kMaX-DeepLab [83] improves MaX-DeepLab by reformulating the cross attention layers to mimic k-means algorithm. Similarly, our MaskConver learns mask embeddings for both thing and stuff classes, but only uses fully convolutional layers [55] without any transformer blocks.

Transformers have surpassed ConvNets on several vision problems beyond panoptic segmentation [32, 42] including classification [16, 18, 20, 52, 53, 75, 80], detection [6, 21, 41, 44, 84], and segmentation [17, 63, 73, 79, 88]. Recently, ConvNeXt [54], a pure convolution-based backbone, provides competitive performance compared to transformer architectures, while preserving the simplicity and efficiency of ConvNets. ConvNeXt block adopts depthwise convolutions with a large kernel size 7×7, layer norm [2], and layer scale [69]. In this work, we adopt the ConvNeXt block as a building block for our pixel decoder. The design is further improved by using Squeeze-and-Excitation [31] layer that improves the performance with little impact on

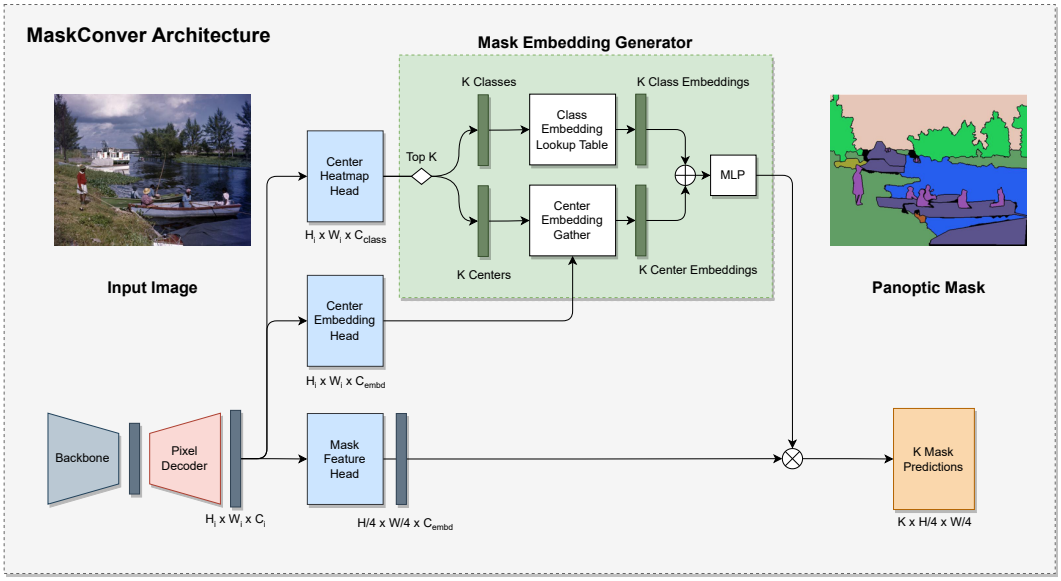

Figure 2. **Illustration of MaskConver architecture.** The meta architecture of MaskConver contains four components: backbone (gray), pixel decoder (pink), prediction heads (light blue), and mask embedding generator (green). The backbone is any commonly deployed neural network, *e.g.*, ResNet50. We propose a novel ConvNeXt-UNet for the pixel decoder, which effectively captures long-range context and high-level semantics by stacking many ConvNeXt blocks at the highest level of backbone. We propose three prediction heads: Center Heatmap Head (for predicting center point heatmaps), Center Embedding Head (for predicting the embeddings for center points), and Mask Feature Head (for generating mask features). The Mask Embedding Generator first produces the class embeddings via a lookup table (Class Embedding Lookup Table module) by taking the predicted semantic classes from the top-K center points. The output mask embeddings are obtained by modulating the class embeddings with the center embeddings (via addition and MLP) to mitigate the center point collision between instances of different classes. In the end, the mask features are multiplied with the mask embeddings to generate the final binary masks. Unlike transformer-based methods, MaskConver only exploits convolutions without any self- or cross-attentions.

the model latency and FLOPs.

Panoptic FCN [46] employs the fully convolutional architecture [55] that predicts things and stuff classes in a unified way, through the convolutional kernel generator to predict things and stuff kernels. These kernels are used to predict binary masks. Although Panopic FCN unifies things and stuff classes towards the post-processing, the kernel generator still treats things and stuff differently (particularly, they represent things as centers, but stuff as regions). Panoptic FCN's quality is lagging behind the recent transformer-based panoptic models. To bridge the gap, MaskConver proposes to fully unify the architecture by only relying on things and stuff centers. It creates a lightweight class embedding module that can break the ties when multiple centers co-exist in the same location. MaskConver also introduces a novel pixel decoder (ConvNeXt-UNet), which provides the model with sufficient context to produce high quality centers and mask predictions.

Efficient Panoptic Segmentation models are less explored, since most architectures focus more on pushing the panoptic quality instead of having more efficient architectures. Panoptic-DeepLab [8, 10] reports quality and latency numbers on a V100 GPU when using MobileNetV3 backbone [29] on an image size of 640×640. Hou *et al*. [28] propose a single-shot panoptic segmentation model that runs in real-time on a V100 GPU. In this work, we tailor MaskConver architecture for mobile devices (Pixel 6) through a set of architectural design choices.

Various lightweight model backbones, including MobileNet [29, 30, 62], EfficientNet [64, 65], and ShuffleNet [57, 86], have been proposed. They are designed for low computational power devices, like mobile CPU and GPU. A multi-hardware MobileNet (MobileNet-MH) [14], discovered by neural architecture search [4, 92] and optimized for multiple hardware [5], achieves state-of-the-art latency and accuracy trade-off on a variety of mobile devices, and has been adopted as the backbone for the semantic segmentation task [74]. To evaluate MaskConver for mobile use cases, we use MobileNet-MH since it delivers similar accuracy as MobileNetV3 [29], while being more compatible with different mobile devices.

Center Point Representation is used in tasks like 2D detection, tracking, action recognition, instance segmentation and panoptic segmentation [34, 45, 81, 89, 90]. In this work, we propose to utilize the center point to model both things and stuff. Additionally, we propose to utilize the mask cen-

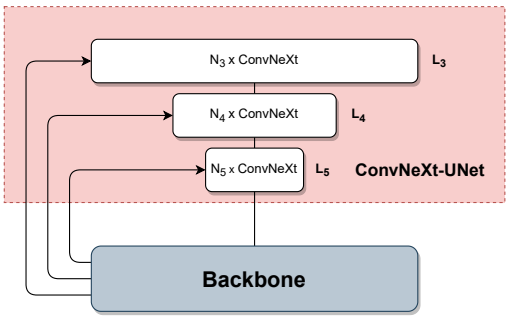

Figure 3. **Pixel decoder with ConvNeXt blocks**. We deploy the modified ConvNeXt blocks in a manner similar to UNet, but employ more blocks in the high level (*i.e.*, level L_5 with stride 32).

ters instead of box centers.

3. Method

Fig. 2 shows the meta architecture of MaskConver, containing four main components: backbone, pixel decoder, prediction heads, and mask embedding generator.

Overview. The backbone is a typical convolutional neural network, such as ResNet [24] and MobileNet [30]. A novel pixel decoder ConvNeXt-UNet is proposed to generate the image features, on top of which prediction heads are appended. We propose three prediction heads: (1) Center Heatmap Head that predicts center point heatmaps [90] for *both* things and stuff, (2) Center Embedding Head that predicts embeddings for the center points, and (3) Mask Feature Head that yields mask features. The Mask Embedding Generator generates the mask embeddings by taking as input both the top-K confident predicted centers (their semantic classes and coordinates) and the center embeddings. In the end, a set of binary masks are obtained by multiplying the mask features with the mask embeddings [72]. We will first explain our design motivations before detailing the proposed modules in the following subsections.

Motivations. Convolution-based models [10, 46] lag in performance, compared to transformer-based models [11, 83]. We carefully look into the mask transformer frameworks [11, 72, 83], and discover that the advantages of employing transformer blocks [71] are twofold: the attention mechanism [3] effectively enriches the pixel decoder with long-range information (thus generates high quality masks) and the object queries produce thing and stuff segmentation masks in a unified way. The observation motivates us to revisit the existing convolution-based methods [10, 46] by designing a better pixel decoder (Sec. 3.1), prediction heads (Sec. 3.2), and mask embedding generator (Sec. 3.3).

3.1. Pixel Decoder: ConvNeXt-UNet

To bridge the gap with transformer-based methods, we first design a novel pixel decoder ConvNeXt-UNet, as shown in Fig. 3, consisting of the modern ConvNeXt blocks [54] deployed in a manner similar to UNet [60] to generate image features. Notably, ConvNeXt-UNet deploys more ConvNeXt blocks at the highest level L_5 of backbone (stride 32). Thanks to the large kernel design, stacking more ConvNeXt blocks at level L_5 effectively captures long-range context information and high-level semantics. Specifically, the decoder architecture is defined by two hyper-parameters: number of repeats, $N = [N_5, N_4, N_3]$, and channel sizes, $D = [D_5, D_4, D_3]$, determining the UNet structure from high level L_5 (stride 32) to low level L_3 (stride 8). For example, setting $N_5 = 18$ and $D_5 = 384$ means the deployment of 18 ConvNeXt blocks with 384 channels at the level L_5. Additionally, we empirically find it effective to add another Squeeze-and-Excitation [31] layer in the ConvNeXt block (called ConvNeXt-SE), as shown in Fig. 4, which improves the model capacity at the cost of extra marginal parameters and negligible FLOPs.

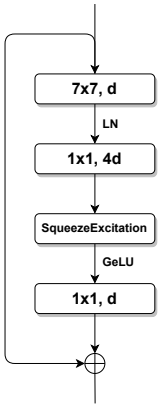

Figure 4. **Modified ConvNeXt block (ConvNeXt-SE)**. On top of the original ConvNeXt block, we additionally include the squeeze-and-excitation operation in-between the 1×1 convolutions.

3.2. Prediction Heads

On top of the image features generated by the proposed pixel decoder, we build three prediction heads for center heatmaps, class embeddings, and mask features. Below, we first explain the structure of our prediction heads.

Light Structure of Head. Unlike existing methods [7, 49, 90] that commonly employ 3×3 convolutions in the prediction heads and introduce heavy computations on low level features (*i.e.*, stride 8 or even stride 4 features), MaskConver, following the design principle of ConvNeXt [54], adopts depthwise convolutions with a large kernel size 7×7 (along with layer normalization [2] and GeLU activation function [26]). This design reduces the FLOPs significantly with no degradation in the panoptic quality.

Center Heatmap Head. Extending object detection methods [68, 90], we propose to use center point representation for *both* things and stuff. We empirically discover that mask center is a better representation than bounding box center. The Center Heatmap Head produces a feature map with shape $H_i \times W_i \times C_{class}$, where H_i and W_i are the height and width of i-th level feature map in the feature pyramid [48], and C_{class} is the number of semantic classes. We will feed the top-K most confident predicted center points (their predicted semantic classes and coordinates) to the Mask Embedding Generator.

Center Embedding Head. The Center Embedding Head generates the embeddings for center points with shape $H_i \times W_i \times C_{embd}$, where C_{embd} is the channel size of embeddings. Its output is fed into the Mask Embedding Generator to gather K center embeddings for the top-K most confident predicted center points (based on their coordinates).

Mask Feature Head. The Mask Feature Head combines the decoder features from L_5 to L_3 to create the mask features. This is done by resizing all the decoder features to the same size (stride 4) and summing them together, before feeding to the light prediction head. The resulting mask features have shape $H/4 \times W/4 \times C_{embd}$, where H and W are the height and width of input image, respectively. The mask features, multiplied with the mask embeddings (from the Mask Embedding Generator, detailed in Sec. 3.3), generate the final output: a set of K binary masks.

3.3. Mask Embedding Generator

The Mask Embedding Generator is one of the crucial designs in MaskConver, aiming to generate high quality mask embeddings. It takes as input both the top-K most confident predicted centers from the Center Heatmap Head (their semantic classes and coordinates) and center embeddings from the Center Embedding Head.

An naïve design would be the simple gathering of the K center embeddings based on the top-K center coordinates (*i.e.*, directly use the K center embeddings as mask embeddings). However, it results in an inferior performance, as we observe the confusion caused by neighboring instances, especially when their centers collide, leading to exactly the same embedding vector being gathered from the output of Center Embedding Head.

Therefore, we propose to additionally exploit the *class embeddings*, which learn to embed each semantic class to a vector of size C_{embd}. The class embeddings are used to modulate (via addition and a MLP) the center embeddings, mitigating the center collisions caused by instances of different semantic classes. Specifically, we design a "Class Embedding Lookup Table" module, which stores the learned embeddings for semantic classes. For the top-K centers, we infer their most likely semantic classes, and obtain their corresponding class embeddings from the module.

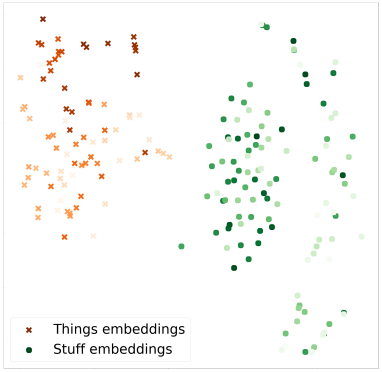

Figure 5. **tSNE [70] plot of learned class embeddings split by things and stuff.** Orange crosses are thing classes and green dots are stuff classes. Our learned class embedding separates things and stuff classes automatically.

We then add the obtained class embeddings and the center embeddings, and pass them to a MLP module (two fully-connected layers) to generate the final mask embeddings. We note that exploiting the class embeddings is critical to the quality of the predicted mask embeddings. It ensures that each instance will have a unique embedding vector, avoiding the problem of instance center collision.

Fig. 5 visualizes the learned class embeddings for things and stuff, using tSNE [70]. As shown in the figure, there are two well-separated clusters: one for things and the other for stuff. As a result, the center collision between things and stuff is alleviated by adding the class embeddings and center embeddings, forming better mask embeddings to generate high quality masks. We will show in experiments that class embeddings provide a decent performance improvements.

4. Experimental Results

We evaluate the effectiveness of the proposed MaskConver on the challenging COCO dataset [50]. COCO has 118,287 training images and 5,000 validation images. It has 80 thing categories and 53 stuff categories. The code and model weights will be publicly available.

4.1. Implementation Details

Training Strategy. The Center Heatmap Head is supervised by the Focal loss [49], while the final mask prediction is supervised by binary cross entropy loss and Dice loss [58]. The total loss is thus defined as: $\text{Loss}_{\text{total}} = \lambda_{\text{centers}}\text{Loss}_{\text{centers}} + \lambda_{\text{bce}}\text{Loss}_{\text{bce}} + \lambda_{\text{dice}}\text{Loss}_{\text{dice}}$. Across all experiments and unless stated otherwise, we fix the weighting factors as $\lambda_{\text{centers}} = 1$, $\lambda_{\text{bce}} = 10$, and $\lambda_{\text{dice}} = 10$. We use Adam weight decay optimizer (AdamW) [36, 56] with learning rate of 0.001, and weight decay of 0.05. Exponential Moving Average (EMA) [59] optimizer is used with average decay of 0.99996. We train the models for 270k it-

Architecture	Backbone	Params	FLOPs	PQ	PQ^{thing}	PQ^{stuff}	FPS
Transformer Based							
DETR [6]	ResNet50 [24]	-	-	43.4	-	-	-
MaskFormer [12]	ResNet50 [24]	45M	181B	46.5	51.0	39.8	17.6
K-Net [85]	ResNet50 [24]	-	-	47.1	51.7	40.3	-
CMT-DeepLab [82]	ResNet50 [24]	-	-	48.5	-	-	-
Panoptic SegFormer [47]	ResNet50 [24]	51M	214B	49.6	54.4	42.4	7.8
Mask2Former [11]	ResNet50 [24]	44M	226B	51.9	57.7	43.0	8.6
kMaX-DeepLab [83]	ResNet50 [24]	57M	168B	53.0	58.3	44.9	22.8
Convolutional Based							
Real-time Panoptic [10]†	ResNet50 [28]	-	-	37.1	41.0	31.3	16
Panoptic FPN [37]	ResNet50 [24]	-	-	39.0	45.9	28.7	17.5
Panoptic-DeepLab [10]	Xception-71 [13]	-	279B	39.7	43.9	33.2	7.5
SOLO-V2 [76]	ResNet50 [24]	-	-	42.1	49.6	30.7	-
Unifying [43]	ResNet50 [24]	-	-	43.3	48.6	35.5	-
Panoptic FCN [46]	ResNet50 [24]	-	-	44.3	50.0	35.6	9.2
MaskConver (ours)	ResNet50 [24]	57M	171B	**53.6**	**58.9**	**45.6**	19.6
Efficient Models							
Panoptic-DeepLab [10]	MobileNetV3L [29]	-	12.2B	30.0	-	-	-
Panoptic-DeepLab [10]†	MobileNet-MH [14]	3.9M	13.9B	30.8	-	-	74
MaskConver (ours)	MobileNet-MH [14]	3.4M	9.5B	**37.2**	39.8	33.1	105
MaskConver-256‡ (ours)	MobileNet-MH [14]	3.4M	1.5B	29.7	30.0	29.2	**375**

Table 1. **Comparison with existing models on COCO panoptic validation set.** The FLOPs and latency are measured on a V100 GPU, using input size 800×1200 and 640×640 for ResNet50 and MobileNet backbones, respectively, following kMaX-DeepLab [83] and Panoptic-DeepLab [10]. †: Reimplemented, using the same MobileNet-MH backbone. ‡: Method evaluated with input size 256×256.

erations and batch size of 128. The input images are resized and padded to 1280×1280 or 640×640 for ResNet50 [24] or MobileNet [14, 29], respectively, following [10, 83]. We use stronger scale jittering [19, 22] with random scale of 0.05 to 2.9 (we observed overfitting if using (0.1 to 1.9)). Additionally, following [83], panoptic Copy-Paste augmentation [22, 35] is used. We note that our model is built on top of TF Vision Garden [27] in TensorFlow [1], which does not support advanced dynamic mechanisms used in modern panoptic segmentation models [11], such as deformable attention [91] and uncertainty-based point supervision [39].

Inference Strategy. To predict K centers, we employ a simple NMS (non-maximum suppression) layer by applying a 3×3 max-pooling to the Center Heatmap Head's output. The locations and corresponding semantic classes from these top-K centers are subsequently used to obtain the center embeddings and class embeddings, which are combined to generate the mask embeddings and produce the final K binary masks. We follow the post-processing logic of [12] to generate panoptic segmentation outputs. We use a score threshold of 0.2, and overlapping threshold of 0.75.

MaskConver w/ ResNet50 Backbone. We provide more details for MaskConver architecture, when using the ResNet50 backbone [24]. For the proposed ConvNeXt-UNet pixel decoder, we set $N = [18, 1, 1]$ and $D = [384, 384, 384]$, *i.e.*, stacking 18 ConvNeXt-SE blocks with 384 channels at the L_5 level, and only one ConvNeXt-SE block at L_4 and L_3 levels. The Center Heatmap Head and Center Embedding Head are attached from levels L_3 to L_7, where extra strided 7×7 depthwise convolutions are applied to the backbone to get L_6 and L_7.

MaskConver w/ Efficient Backbone. We make some changes, when deloying MaskConver with efficient backbones [14, 29]. The Center Heatmap Head and Center Embedding Head are only appended to a single scale, *i.e.*, L_3 feature map in the feature pyramid. We experiment with MobileNetV3-Large [29] and multi-hardware MobileNet (MobileNet-MH) [14] as the backbone. The ConvNeXt-UNet pixel decoder is replaced with the simpler DeepLabv3+ decoder [9]. For efficiency, we use hard-sigmoid [15], since sigmoid is costly on mobile devices. The model is converted to TFLite models and benchmarked on mobile devices to obtain the latency. We also apply weight-only post-training quantization to further speed up the model by $2\times$.

4.2. Main Results

In Tab. 1, we compare the proposed MaskConver with other methods in three categories: convolution-based, transformer-based, and efficient models.

Convolution-Based Models. In the category of convolution-based Models (*i.e.*, middle group in Tab. 1), MaskConver consistently outperforms all the other convolution-based methods in terms of both performance (PQ) and speed (FPS). Particularly, when comparing with the state-of-the-art Panoptic FCN [46], MaskConver is +9.3% PQ better and running 2.13 times faster.

Transformer-Based Models. When compared with transformer-based models (*i.e.*, top group in Tab. 1), MaskConver achieves better PQ when using similar FLOPs/parameters. In particular, MaskConver is +1.7% better than Mask2Former [11], while also being faster on a V100 GPU. MaskConver is also +0.6% better than kMaX-DeepLab [83] with a slightly higher number of flops. These results suggest that with a better designed pixel decoder, prediction heads, and mask embedding generator, MaskConver can successfully bridge the gap between transformer- and convolution-based models.

Efficient Models. For efficient models [14, 29] (*i.e.*, bottom group in Tab. 1), we compared MaskConver with Panoptic-DeepLab [10]. We employ Panoptic-DeepLab with the same MobileNet-MH backbone [14] and input size 640×640 to have a fair comparison. Our model with 640 input image achieves +6.4% better PQ compared to Panoptic-DeepLab, while also being $1.42\times$ times faster on a V100 GPU. Furthermore, if we change the input size to 256×256, our MaskConver-256 achieves a similar PQ to Panoptic-DeepLab (29.7% *vs*. 30.8% PQ), while running $5.07\times$ times faster. Our MaskConver-256 runs real-time on Pixel 6 GPU with 33 FPS.

4.3. Ablation Studies

We conduct the systematic ablation studies on the MaskConver architecture, using the ResNet50 backbone.

Pixel Decoder. In Tab. 2, we ablate on the design choices of pixel decoder. We start with the popular pixel decoder choice: FPN (feature pyramid network) [48] as our baseline, which attains the performance of 43.1% PQ. We then use the more advanced feature pyramid architecture, BiFPN (bi-directional feature pyramid network) [66], which improves the performance to 47.7% PQ with a reasonable increase in both FLOPs and model parameters, serving as another solid baseline. After setting the solid baselines, we explore the structure of the proposed ConvNeXt-UNet with two hyper-parameters: number of repeats of ConvNeXt blocks, $N = [N_5, N_4, N_3]$, and channel sizes, $D = [D_5, D_4, D_3]$, which determine the UNet structure from high level L_5 (stride 32) to low level L_3 (stride 8). We first set $N = [3, 9, 3]$ and $D = [768, 384, 192]$, which corresponds to the inverting ConvNeXt-Tiny [54] structure. This simple design surprisingly improves over the strong baseline BiFPN by +1.4% PQ, while also being slightly more efficient in FLOPs. Motivated by the prior works [7, 87] that employ the multi-scale context module at the highest level (*i.e.*, stride 32) of backbone, we explore stacking more ConvNeXt blocks to level 5 to capture more long-range information and high-level semantics. Hence, we move most of the blocks to level 5 by setting $N = [11, 1, 1]$ and $D = [512, 383, 384]$, in order to keep the similar number of parameters. This structure further improves the performance by +1% PQ. To further push the envelope, we explore stacking more blocks at level 5 by using $N = [18, 1, 1]$ and $D = [384, 384, 384]$, which yields additional +0.3% improvement. Finally, employing the proposed ConvNeXt-SE block (*i.e.*, adding another Squeeze-and-Excitation [31] layer in ConvNeXt block) improves the PQ by +0.4% with minor effect on the model FLOPs. Overall, we observe a +3.1% improvement in PQ compared to the solid baseline BiFPN [66] with 19% lower FLOPs, and +7.7% improvement in PQ compared to FPN.

Light Structure of Head. In this study, we evaluate the effectiveness of the adopted light structure of prediction heads (in Sec. 3.2). As shown in Tab. 3, the light prediction head using the 7×7 depthwise convolution slightly improves over the regular 3×3 convolution by +0.3% PQ with a significant reduction of 75% in FLOPs. This reduction is mainly because the convolution layers that process low-level features (*e.g.*, stride 8 features) are very expensive. Replacing these regular convolution layers with depthwise convolution is more efficient for both accuracy and FLOPs.

Class Embeddings. The proposed class embeddings (generated by Class Embedding Lookup Table in Sec. 3.3) modulates the center embeddings (via addition and a MLP) to mitigate the instance collision. In this study, we evaluate its significance, using both ResNet50 and MobileNet-MH backbone, in Tab. 4. As shown in the table, using the class embedding (see column 'Cls-Embd') shows +1.5% and +2.3% improvement for ResNet50 and MobileNet-MH, respectively. These results suggest that using class embeddings helps break the tie, when two centers of different classes are present at the same location. Additionally, the improvement for MobileNet-MH is more significant, since MaskConver with MobileNet-MH uses only a single output scale (L_3) to predict the centers (for the purpose of efficiency), in which case we expect more center collisions and hence the class embeddings become more important.

Mask Centers *vs*. Box Centers. Unlike CenterNet [90], MaskConver uses mask centers, instead of bounding box centers. We ablate this design choice in Tab. 5, which shows +0.6% PQ improvement, when using the mask centers. As a result, the mask generates a better center representation than the bounding box.

Effect of Training Strategy. In this study, we evaluate the effect of several training techniques that are used in our framework. As shown in Tab. 6, training the model longer for 270k iterations gives 0.3% improvement over 150k it-

Pixel Decoder	Design	FLOPs	Params	PQ
FPN [48]	L_3-L_7	155B	31M	43.1
BiFPN [66]	6 BiFPN layers, L_3-L_7	210B	55M	47.7
ConvNeXt-UNet	N = [3, 9, 3], D = [768, 384, 192]	195B	54M	49.1
ConvNeXt-UNet	N = [11, 1, 1], D = [512, 384, 384]	173B	53M	50.1
ConvNeXt-UNet	N = [18, 1, 1], D = [384, 384, 384]	171B	51M	50.4
ConvNeXt-UNet	N = [18, 1, 1], D = [384, 384, 384] + SE [31]	171B	57M	50.8

Table 2. **Effect of pixel decoder designs on COCO val set.** Our final design of ConvNeXt-UNet pixel decoder (last row) stacks many ConvNeXt blocks at the level 5 (stride 32), effectively capturing long-range information and high-level semantics.

Prediction Head Structure	FLOPs	Params	PQ
3×3 Convs	696B	68M	53.3
7×7 Depthwise Convs	**171B**	**57M**	**53.6**

Table 3. **Effect of using efficient heads on COCO val set.** In the prediction heads, using 7×7 depthwise convolution is more efficient than 3×3 convolution in both accuracy and FLOPs.

Backbone	Cls-Embd	PQ	PQ^{thing}	PQ^{stuff}
ResNet-50	✗	52.1	57.2	44.1
ResNet-50	✓	**53.6**	**58.9**	**45.6**
MobileNet-MH	✗	34.9	37.5	31.0
MobileNet-MH	✓	**37.2**	**39.8**	**33.1**

Table 4. **Significance of class embeddings on COCO val set.** Using class embeddings (Cls-Embd) improves both things classes (PQ^{thing}) and stuff classes (PQ^{stuff}) for both ResNet50 and MobileNet-MH.

Centers	FLOPs	Params	PQ
Box Centers	171B	57M	53.0
Mask Centers	171B	57M	**53.6**

Table 5. **Effect of mask centers on COCO panoptic val set.** Using mask centers improves the PQ compared to box centers.

erations (used in [83]). We did not observe any improvement if we train the model even longer. Using stronger scale jittering with scale [0.05, 2.9] further improves the performance by +0.8%. Finally, the panoptic Copy-Paste [35, 83] shows a significant improvement with +1.7% PQ. We note again that our model is built in TensorFlow [1], which does not support advanced dynamic mechanisms commonly used in modern panoptic segmentation models [11, 47], such as deformable attention [91] and uncertainty-based point supervision [39].

Stuff Center vs Stuff Region. We contrast Region-based and Center-based methods for representing stuff classes in MaskConver with a ResNet-50 backbone in Tab. 7 on COCO val set. The center-based approach demonstrates a significant 21% drop in latency (FPS) — a crucial advantage for real-time computer vision applications — with only a marginal dip in PQ^{stuff}.

Longer Training	Strong Aug.	Copy-Paste	PQ
			50.8
✓			51.1
✓	✓		51.9
✓	✓	✓	53.6

Table 6. **Effect of training techniques on COCO val set.** Our final setting employs longer training iterations (270k), strong scale augmentation ([0.05, 2.9]), and panoptic copy-paste.

Stuff	Backbone	PQ^{stuff}	FPS
Center	Resnet-50	45.6	19.6
Region [46]	Resnet-50	45.8	15.4

Table 7. **Region-based vs. Center-based stuff representations using ResNet-50.** The table highlights the efficiency gains of the center-based approach in terms of latency (FPS), while showcasing a minimal compromise in PQ^{stuff}.

5. Conclusion

In this work, we have presented MaskConver, revisiting pure convolution for panoptic segmentation. MaskConver simplifies the convolution-based panoptic models by unifying things and stuff modeling. Specifically, MaskConver uses centers to represent both thing and stuff regions, and employs the light class embedding module to predict unique embedding vectors for multiple instances that are present at the same locations. MaskConver also adopted the ConvNeXt-UNet pixel decoder that provides the prediction heads with long-range context and high-level semantics. With simplified architecture and the ConvNeXt-UNet, MaskConver closes the gap with the transformer-based models on COCO dataset. Finally, MaskConver excelled in the mobile domain, thanks to the simplicity and efficiency of convolutions.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467*, 2016. 6, 8

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv:1607.06450*, 2016. 2, 4

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015. 4

[4] Gabriel Bender, Hanxiao Liu, Bo Chen, Grace Chu, Shuyang Cheng, Pieter-Jan Kindermans, and Quoc V Le. Can weight sharing outperform random architecture search? an investigation with tunas. In *CVPR*, 2020. 3

[5] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *ICLR*, 2020. 3

[6] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 1, 2, 6

[7] 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. 4, 7

[8] Liang-Chieh Chen, Huiyu Wang, and Siyuan Qiao. Scaling wide residual networks for panoptic segmentation. *arXiv:2011.11675*, 2020. 3

[9] 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. 6

[10] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *CVPR*, 2020. 2, 3, 4, 6, 7

[11] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *CVPR*, 2022. 1, 2, 4, 6, 7, 8

[12] Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. *NeurIPS*, 2021. 1, 2, 6

[13] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017. 6

[14] Grace Chu, Okan Arikan, Gabriel Bender, Weijun Wang, Achille Brighton, Pieter-Jan Kindermans, Hanxiao Liu, Berkin Akin, Suyog Gupta, and Andrew Howard. Discovering multi-hardware mobile models via architecture search. In *CVPR*, 2021. 3, 6, 7

[15] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. *NeurIPS*, 2015. 6

[16] Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. *NeurIPS*, 34:3965–3977, 2021. 2

[17] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. Cswin transformer: A general vision transformer backbone with cross-shaped windows. In *CVPR*, 2022. 2

[18] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 2

[19] Xianzhi Du, Barret Zoph, Wei-Chih Hung, and Tsung-Yi Lin. Simple training strategies and model scaling for object detection. *arXiv:2107.00057*, 2021. 6

[20] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *ICCV*, 2021. 2

[21] Peng Gao, Minghang Zheng, Xiaogang Wang, Jifeng Dai, and Hongsheng Li. Fast convergence of detr with spatially modulated co-attention. In *ICCV*, 2021. 2

[22] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D Cubuk, Quoc V Le, and Barret Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation. In *CVPR*, 2021. 6

[23] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *ECCV*, 2014. 1

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 4, 6

[25] Xuming He, Richard S Zemel, and Miguel Á Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *CVPR*, 2004. 1

[26] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv:1606.08415*, 2016. 4

[27] Xianzhi Du Yeqing Li Abdullah Rashwan Le Hou Pengchong Jin Fan Yang Frederick Liu Jaeyoun Kim Hongkun Yu, Chen Chen and Jing Li. TensorFlow Model Garden. https://github.com/tensorflow/models, 2020. 6

[28] Rui Hou, Jie Li, Arjun Bhargava, Allan Raventos, Vitor Guizilini, Chao Fang, Jerome Lynch, and Adrien Gaidon. Real-time panoptic segmentation from dense detections. In *CVPR*, 2020. 3, 6

[29] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *ICCV*, 2019. 2, 3, 6, 7

[30] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv:1704.04861*, 2017. 3, 4

[31] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018. 2, 4, 7, 8

[32] Jitesh Jain, Jiachen Li, MangTik Chiu, Ali Hassani, Nikita Orlov, and Humphrey Shi. Oneformer: One transformer to rule universal image segmentation. In *CVPR*, 2023. 2

[33] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *NeurIPS*, 2016. 1, 2

[34] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, 2018. 3

[35] Dahun Kim, Jun Xie, Huiyu Wang, Siyuan Qiao, Qihang Yu, Hong-Seok Kim, Hartwig Adam, In So Kweon, and Liang-Chieh Chen. Tubeformer-deeplab: Video mask transformer. In *CVPR*, 2022. 6, 8

[36] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[37] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *CVPR*, 2019. 1, 2, 6

[38] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *CVPR*, 2019. 1, 2

[39] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *CVPR*, 2020. 6, 8

[40] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2

[41] Feng Li, Hao Zhang, Shilong Liu, Jian Guo, Lionel M Ni, and Lei Zhang. Dn-detr: Accelerate detr training by introducing query denoising. In *CVPR*, 2022. 2

[42] Feng Li, Hao Zhang, Shilong Liu, Lei Zhang, Lionel M Ni, Heung-Yeung Shum, et al. Mask dino: Towards a unified transformer-based framework for object detection and segmentation. *arXiv:2206.02777*, 2022. 2

[43] Qizhu Li, Xiaojuan Qi, and Philip HS Torr. Unifying training and inference for panoptic segmentation. In *CVPR*, 2020. 6

[44] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. In *ECCV*, 2022. 2

[45] Yixuan Li, Zixu Wang, Limin Wang, and Gangshan Wu. Actions as moving points. In *ECCV*, 2020. 3

[46] 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, 3, 4, 6, 7, 8

[47] Zhiqi Li, Wenhai Wang, Enze Xie, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, Ping Luo, and Tong Lu. Panoptic segformer: Delving deeper into panoptic segmentation with transformers. In *CVPR*, 2022. 1, 2, 6, 8

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

[49] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 4, 5

[50] 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

[51] Huanyu Liu, Chao Peng, Changqian Yu, Jingbo Wang, Xu Liu, Gang Yu, and Wei Jiang. An end-to-end network for panoptic segmentation. In *CVPR*, 2019. 1, 2

[52] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *CVPR*, 2022. 2

[53] 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. In *ICCV*, 2021. 2

[54] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *CVPR*, 2022. 2, 4, 7

[55] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 2, 3

[56] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 5

[57] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, 2018. 3

[58] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *3DV*, 2016. 5

[59] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992. 5

[60] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 2, 4

[61] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *IJCV*, 115:211–252, 2015. 2

[62] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 3

[63] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *ICCV*, 2021. 2

[64] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*. PMLR, 2019. 3

[65] Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *ICML*, 2021. 3

[66] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *CVPR*, 2020. 2, 7, 8

[67] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. In *ECCV*, 2020. 1, 2

[68] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *ICCV*, 2019. 5

[69] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *CVPR*, 2021. 2

[70] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 2008. 5

[71] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 1, 4

[72] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Max-deeplab: End-to-end panoptic segmentation with mask transformers. In *CVPR*, 2021. 1, 2, 4

[73] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-DeepLab: Stand-Alone Axial-Attention for Panoptic Segmentation. In *ECCV*, 2020. 2

[74] Weijun Wang and Andrew Howard. Mosaic: Mobile segmentation via decoding aggregated information and encoded context. *arXiv:2112.11623*, 2021. 3

[75] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*, 2021. 2

[76] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic and fast instance segmentation. *NeurIPS*, 33:17721–17732, 2020. 1, 2, 6

[77] Mark Weber, Huiyu Wang, Siyuan Qiao, Jun Xie, Maxwell D. Collins, Yukun Zhu, Liangzhe Yuan, Dahun Kim, Qihang Yu, Daniel Cremers, Laura Leal-Taixe, Alan L. Yuille, Florian Schroff, Hartwig Adam, and Liang-Chieh Chen. DeepLab2: A TensorFlow Library for Deep Labeling. *arXiv: 2106.09748*, 2021. 2

[78] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. Upsnet: A unified panoptic segmentation network. In *CVPR*, 2019. 1, 2

[79] Jiarui Xu, Shalini De Mello, Sifei Liu, Wonmin Byeon, Thomas Breuel, Jan Kautz, and Xiaolong Wang. Groupvit: Semantic segmentation emerges from text supervision. In *CVPR*, 2022. 2

[80] Chenglin Yang, Siyuan Qiao, Qihang Yu, Xiaoding Yuan, Yukun Zhu, Alan Yuille, Hartwig Adam, and Liang-Chieh Chen. Moat: Alternating mobile convolution and attention brings strong vision models. In *ICLR*, 2023. 2

[81] Tien-Ju Yang, Maxwell D Collins, Yukun Zhu, Jyh-Jing Hwang, Ting Liu, Xiao Zhang, Vivienne Sze, George Papandreou, and Liang-Chieh Chen. Deeperlab: Single-shot image parser. *arXiv:1902.05093*, 2019. 1, 2, 3

[82] Qihang Yu, Huiyu Wang, Dahun Kim, Siyuan Qiao, Maxwell Collins, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Cmt-deeplab: Clustering mask transformers for panoptic segmentation. In *CVPR*, 2022. 1, 2, 6

[83] Qihang Yu, Huiyu Wang, Siyuan Qiao, Maxwell Collins, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. k-means mask transformer. In *ECCV*, 2022. 1, 2, 4, 6, 7, 8

[84] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel Ni, and Heung-Yeung Shum. DINO: DETR with improved denoising anchor boxes for end-to-end object detection. In *ICLR*, 2023. 2

[85] Wenwei Zhang, Jiangmiao Pang, Kai Chen, and Chen Change Loy. K-net: Towards unified image segmentation. *NeurIPS*, 34:10326–10338, 2021. 6

[86] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018. 3

[87] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. 7

[88] 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. 2

[89] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *ECCV*, 2020. 3

[90] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv:1904.07850*, 2019. 2, 3, 4, 5, 7

[91] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR*, 2021. 2, 6, 8

[92] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017. 3