Supplementary Material for MaX-DeepLab: End-to-End Panoptic Segmentation with Mask Transformers

Huiyu Wang[1*] Yukun Zhu[2] Hartwig Adam[2] Alan Yuille[1] Liang-Chieh Chen[2]
[1]Johns Hopkins University [2]Google Research

In this supplementary material, we mainly provide more visualizations, runtimes, and analyses on COCO [6] validation set. In addition, we include more technical details of our architectures and training settings for the experiments in our main paper.

1. Panoptic Segmentation Results

Similar to the case study in Fig. 2 of the main paper, we provide more panoptic segmentation results of our MaX-DeepLab-L and compare them to the state-of-the-art *box-free* method, Axial-DeepLab [9], the state-of-the-art *box-based* method, DetectoRS [7], and the first Detection Transformer, DETR [1] in Fig. 1 and Fig. 2. MaX-DeepLab demonstrates robustness to the challenging cases of similar object bounding boxes and nearby objects with close centers, while other methods make systematic mistakes because of their individual surrogate sub-task design. MaX-DeepLab also shows exceptional mask quality, and performs well in the cases of many small objects. Similar to DETR [1], MaX-DeepLab fails typically when there are too many object masks.

2. Runtime

In Tab. 1, we report the end-to-end runtime (i.e., inference time from an input image to final panoptic segmentation) of MaX-DeepLab on a V100 GPU. All results are obtained by (1) a single-scale input without flipping, and (2) built-in TensorFlow library without extra inference optimization. In the fast regime, MaX-DeepLab-S takes 67 ms with a typical 641×641 input. This runtime includes 5 ms of postprocessing and 15 ms of batch normalization that can be easily optimized. This fast MaX-DeepLab-S does not only outperform DETR-R101 [1], but is also around 2x faster. In the slow regime, the standard MaX-DeepLab-S takes 131 ms with a 1025×1025 input, similar to Panoptic-DeepLab-X71 [2]. This runtime is also similar to our run of the official DETR-R101 which takes 128 ms on a V100, including 63 ms for box detection and 65 ms for the heavy mask decoding.

3. Mask Output Slot Analysis

In this section, we analyze the statistics of all $N = 128$ mask prediction slots using MaX-DeepLab-L. In Fig. 3, we visualize the joint distribution of mask slot firings and the classes they predict. We observe that the mask slots have imbalanced numbers of predictions and they specialize on 'thing' classes and 'stuff' classes. Similar to this Mask-Class joint distribution, we visualize the Mask-Pixel joint distribution by extracting an average mask for each mask slot, as shown in Fig. 4. Specifically, we resize all COCO [6] validation set panoptic segmentation results to a unit square and take an average of masks that are predicted by each mask slot. We split all mask slots into three categories according to their total firings and visualize mask slots in each category. We observe that besides the class-level specialization, our mask slots also specialize on certain regions of an input image. This observation is similar to DETR [1], but we do not see the pattern that almost all slots have a mode of predicting large image-wide masks.

4. Mask Head Visualization

In Fig. 5 of the main paper, we visualize how the mask head works by training a MaX-DeepLab with only $D = 3$ decoder feature channels (for visualization purpose only). Although this extreme setting degrades the performance from 45.7% PQ to 37.8% PQ, it enables us to directly visualize the decoder features as RGB colors. Here in Fig. 5 we show more examples using this model, together with the corresponding panoptic sementation results. We see a similar clustering effect of instance colors, which enables our simple mask extraction with just a matrix multiplication (a.k.a. dynamic convolution [8, 10, 4, 11]).

5. Transformer Attention Visualization

We also visualize the *M2P* attention that connects the transformer to the CNN. Specifically, given an input image from COCO validation set, we first select four output

*Work done while an intern at Google.
[1]https://github.com/facebookresearch/detr

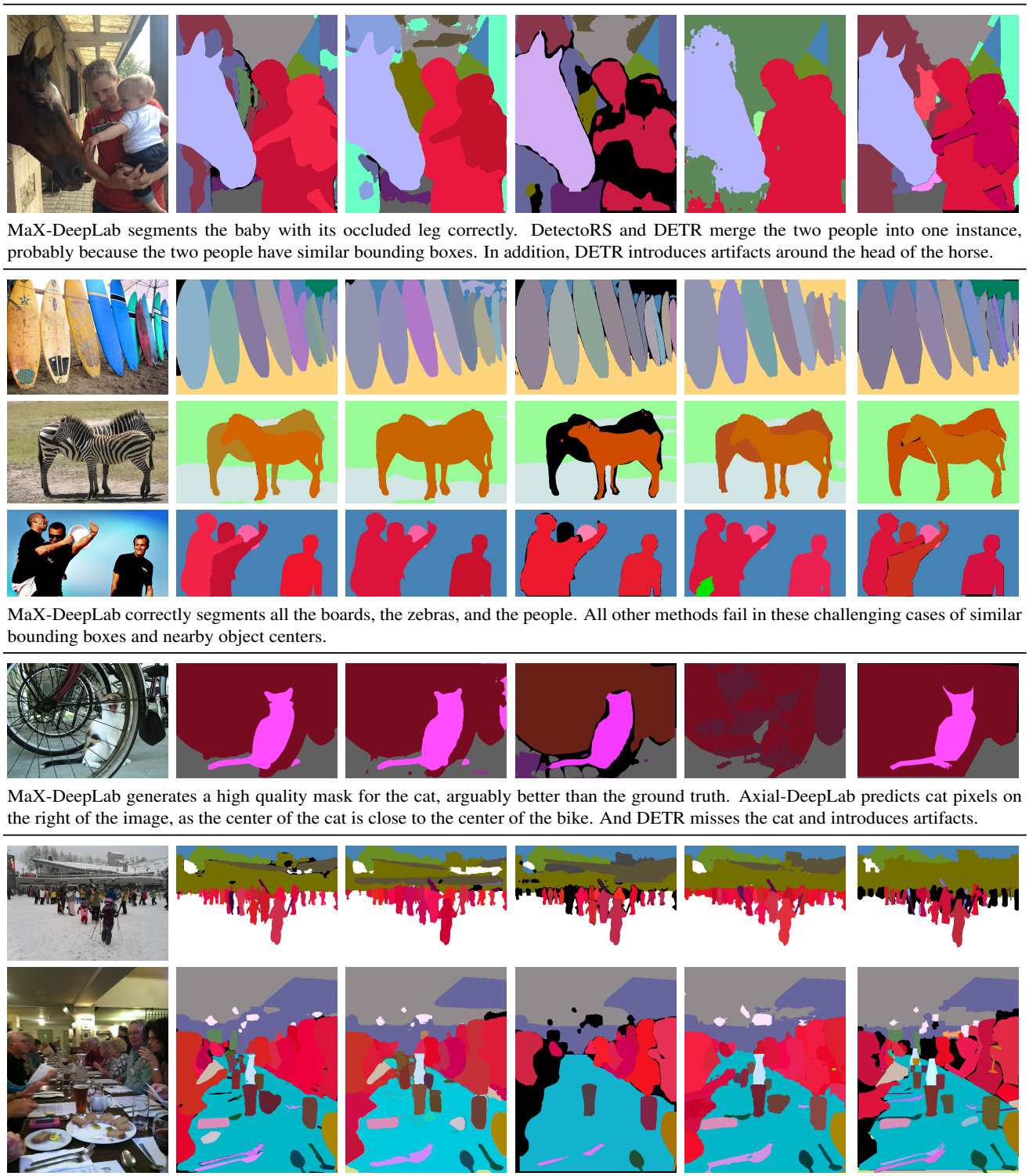


Figure 1. Comparing MaX-DeepLab with other representative methods on the COCO *val* set. (Colors modified for better visualization).

Method	Backbone	Input Size	Runtime (ms)	PQ [val]	PQ [test]
Fast Regime					
Panoptic-DeepLab [2]	X-71 [3]	641×641	74	38.9	38.8
MaX-DeepLab-S	MaX-S	641×641	67	46.4	46.7
Slow Regime					
DETR [1]	RN-101	1333×800	128[1]	45.1	46.0
Panoptic-DeepLab [2]	X-71 [3]	1025×1025	132	39.7	39.6
MaX-DeepLab-S	MaX-S	1025×1025	131	48.4	49.0

Table 1. End-to-end runtime. **PQ [val]:** PQ (%) on COCO val set. **PQ [test]:** PQ (%) on COCO test-dev set.

masks of interest from the MaX-DeepLab-L panoptic prediction. Then, we probe the attention weights between the four masks and all the pixels, in the last dual-path transformer block. Finally, we colorize the four attention maps with four colors and visualize them in one figure. This process is repeated for two images and all eight attention heads as shown in Fig. 6. We omit our results for the first transformer block since it is mostly flat. This is expected because the memory feature in the first transformer block is unaware of the pixel-path input image at all. Unlike DETR [1] which focuses on object extreme points for detecting bounding boxes, our MaX-DeepLab attends to individual object (or stuff) masks. This mask-attending property makes MaX-DeepLab relatively robust to nearby objects with similar bounding boxes or close mass centers.

6. More Technical Details

In Fig. 7, Fig. 8, and Fig. 9, we include more details of our MaX-DeepLab architectures. As marked in the figure, we pretrain our model on ImageNet [5]. The pretraining model uses only *P2P* attention (could be a convolutional residual block or an axial-attention block), without the other three types of attention, the feed-forward network (FFN), or the memory. We directly pretrain with an average pooling followed by a linear layer. This pretrained model is used as a backbone for panoptic segmentation, and it uses the backbone learning rate multiplier we mentioned in Sec. 4 of the main paper. After pretraining the CNN path, we apply (with random initialization) our proposed memory path, including the memory, the three types of attention, the FFNs, the decoding layers, and the output heads for panoptic segmentation. In addition, we employ multi-head attention with 8 heads for all attention operations. In MaX-DeepLab-L, we use shortcuts in the stacked decoder. Specifically, each decoding stage (resolution) is connected to the nearest two previous decoding stage outputs of the same resolution.

References

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

[2] 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.

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

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

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

[6] 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.

[7] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. *arXiv:2006.02334*, 2020.

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

[9] 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.

[10] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. SOLOv2: Dynamic and fast instance segmentation. In *NeurIPS*, 2020.

[11] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. In *NeurIPS*, 2019.

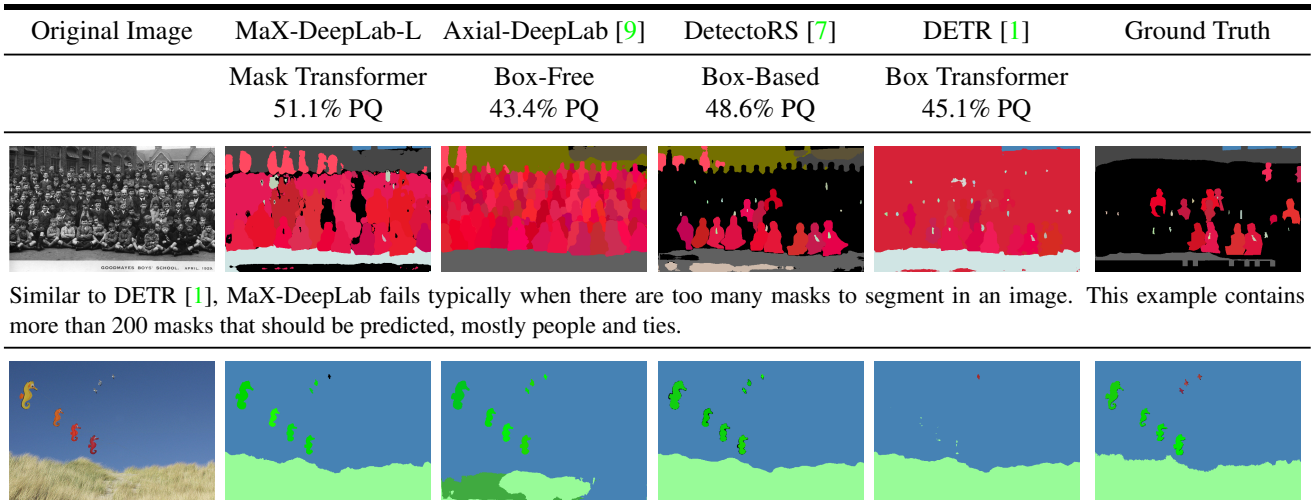


Figure 2. Failure cases of MaX-DeepLab on the COCO *val* set.

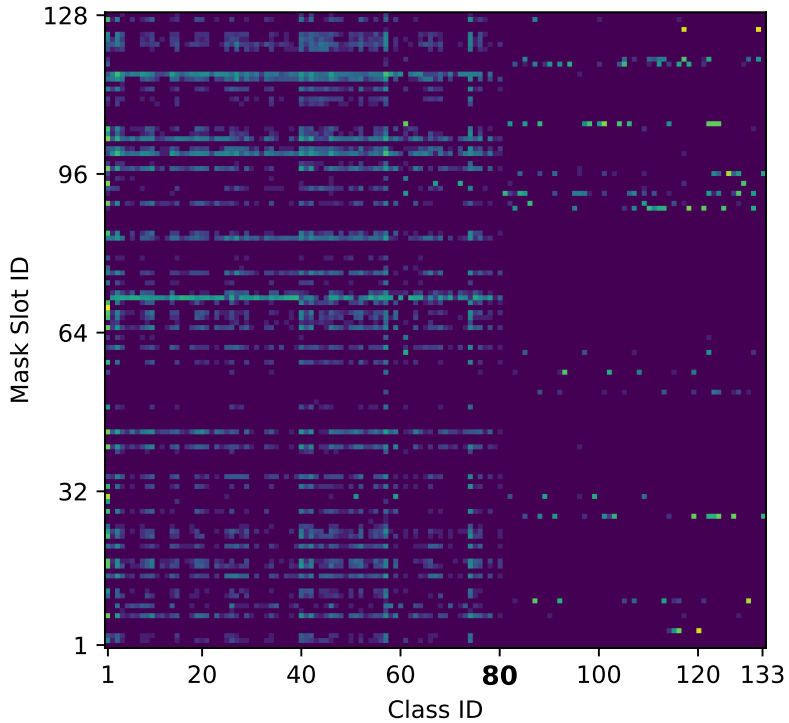


Figure 3. The joint distribution for our $N = 128$ mask slots and 133 classes with 80 'thing' classes on the left and 53 'stuff' classes on the right. We observe that a few mask slots predict a lot of the masks. Some mask slots are used less frequently, probably only when there are a lot of objects in one image. Some other slots do not fire at all. In addition, we see automatic functional segregation between 'thing' mask slots and 'stuff' mask slots, with a few exceptions that can predict both thing and stuff masks.

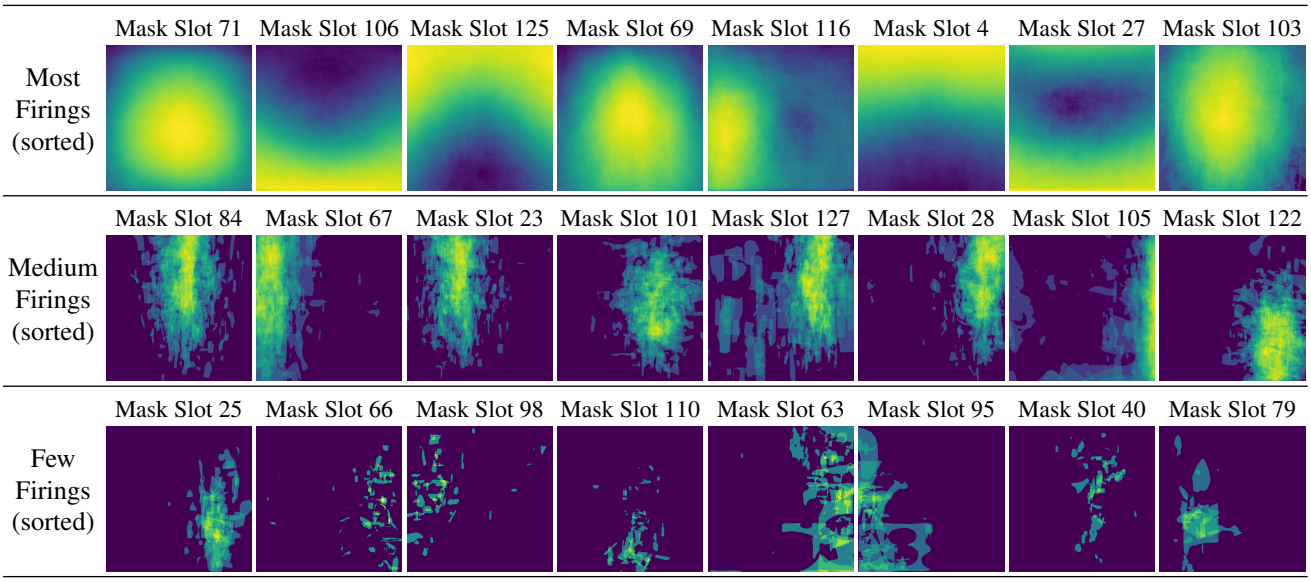
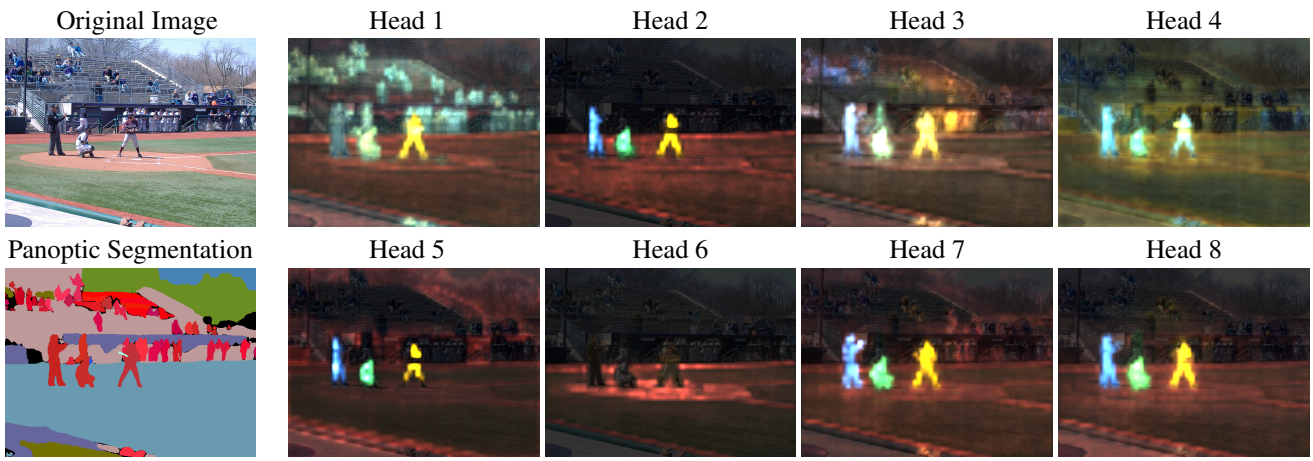
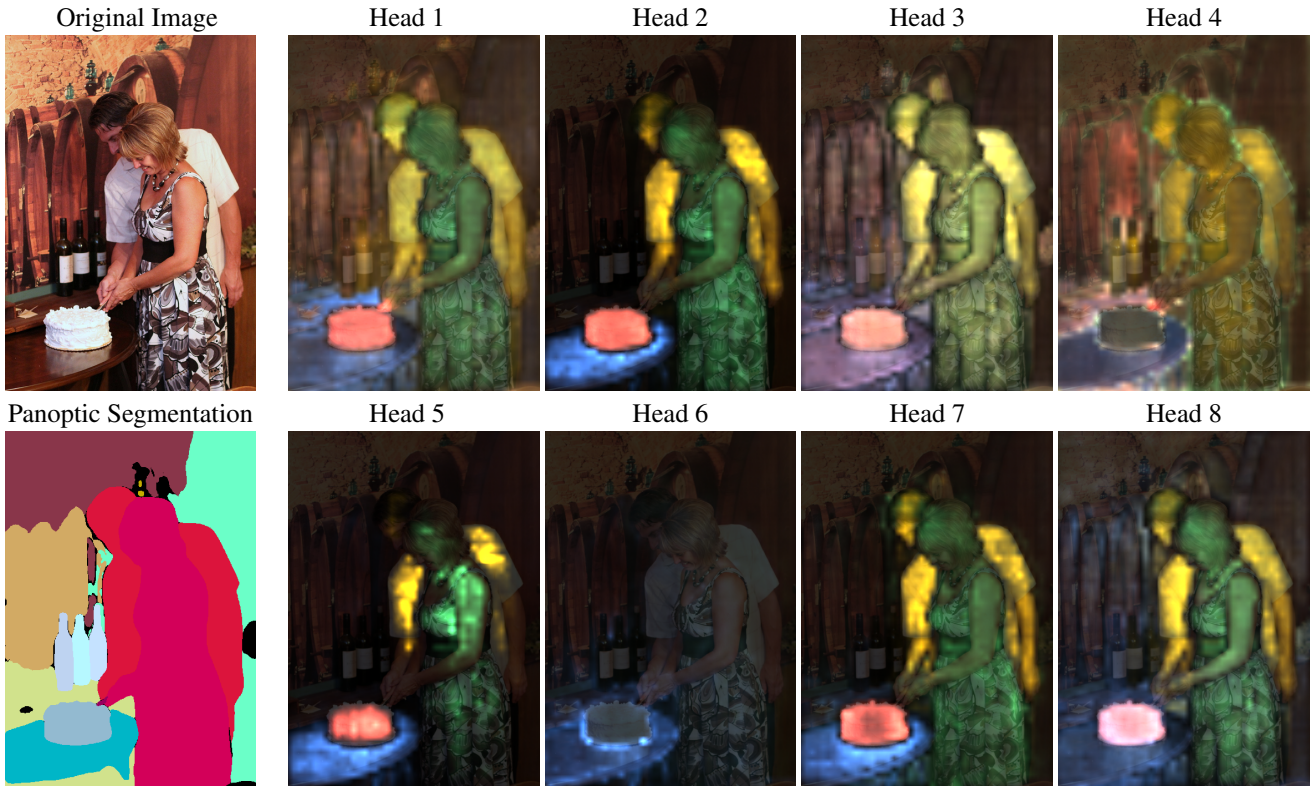


Figure 4. The average masks that each mask slot predicts, normalized by image shape. Mask slots are categorized by their total number of firings and sorted from most firings to few firings. We observe spatial clustered patterns, meaning that the mask slots specialize on certain regions of an input image. For example, the most firing mask slot 71, focusing on the center of an image, predicts almost all 80 'thing' classes but ignores 'stuff' classes (Fig. 3). The top three categories are tennis rackets, cats, and dogs. The second firing mask slot 106 segments 14 classes of masks on the bottom of an image, such as road, floor, or dining-tables. The third firing mask slot 125 concentrates 99.9% on walls or trees that are usually on the top of an image. The fourth firing mask slot 69 focuses entirely on the person class and predicts 2663 people in the 5000 validation images.

Figure 5. More visualizations of the decoder feature g with $D = 3$. Similar to Fig. 5 of the main paper, we observe a clustering effect of instance colors, *i.e.*, pixels of the same instance have similar colors (features) while pixels of different instances have distinct colors. Note that in this extreme case of $D = 3$ (that achieves 37.8% PQ), there are not enough colors for all masks, which causes missing objects or artifacts at object boundaries, but these artifacts do not present in our normal setting of $D = 128$ (that achieves 45.7% PQ).



Attention maps for three people (**left**, **middle**, **right**) on a **playing field**.



Attention maps for two people (**woman**, **man**) cutting a **cake** on a **table**.

Figure 6. Visualizing the transformer *M2P* attention maps for selected predicted masks. We observe that head 2, together with head 5, 7, and 8, mainly attends to the output mask regions. Head 1, 3, and 4 gather more context from broader regions, such as semantically-similar instances (scene 1 head 1) or mask boundaries (scene 2 head 4). In addition, we see that head 6 does not pay much attention to the pixel-path, except for some minor firings on the playing field and on the table. Instead, it focuses more on *M2M* self-attention which shares the same softmax with *M2P* attention (Equ. (14) of the main paper).

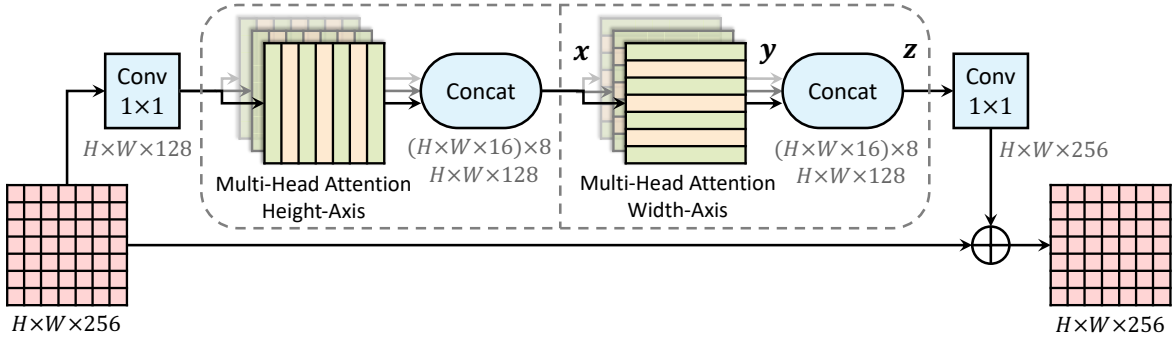
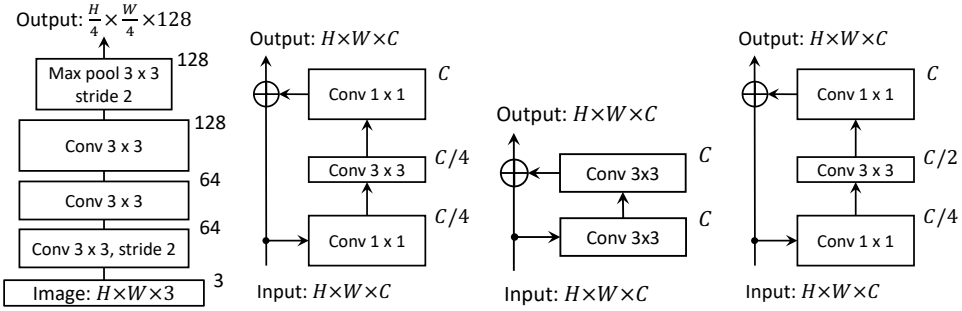
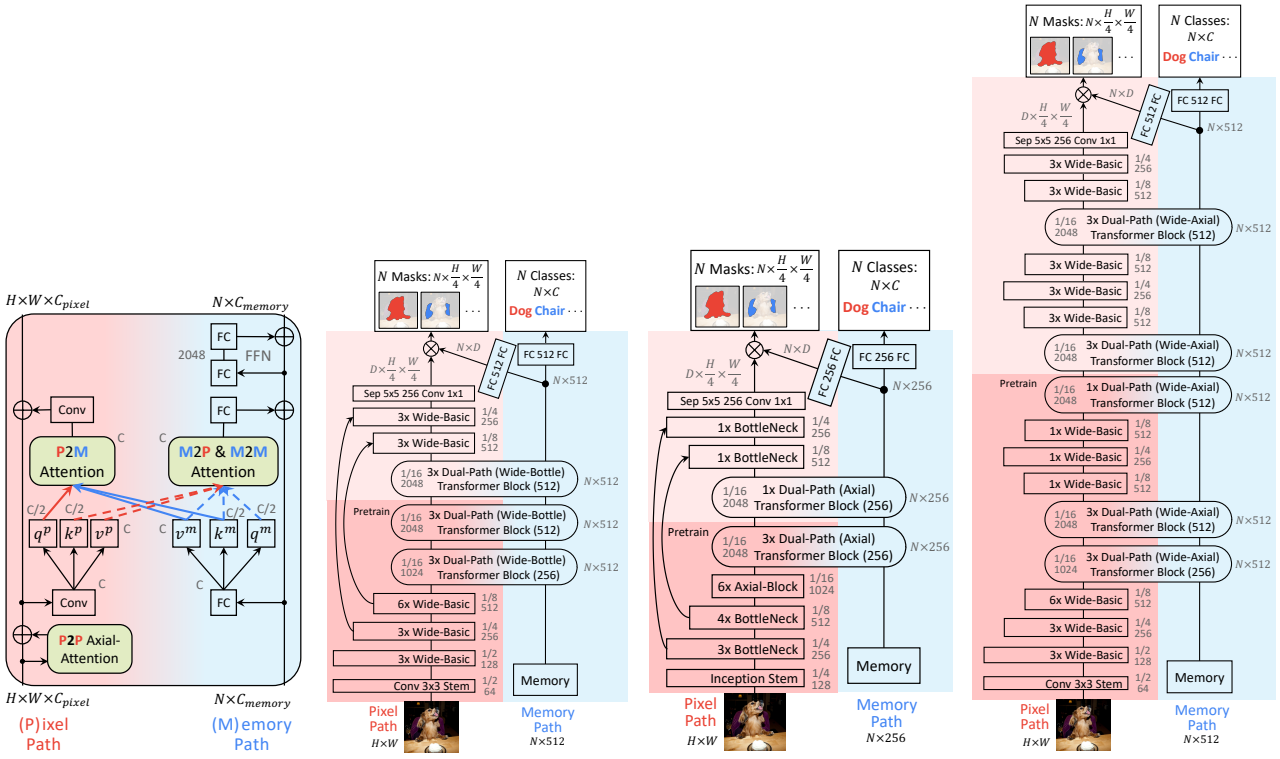


Figure 7. An example Axial-Block from Axial-DeepLab [9]. This axial-attention bottleneck block consists of two axial-attention layers operating along height- and width-axis sequentially.



(a) Inception Stem (b) Bottleneck block (c) Wide-Basic block (d) Wide-Bottle block

Figure 8. Building blocks for our MaX-DeepLab architectures.



(a) Dual-Path Transformer Block (b) MaX-DeepLab-Ablation (c) MaX-DeepLab-S (d) MaX-DeepLab-L

Figure 9. More detailed MaX-DeepLab architectures. **Pretrain** labels where we use a classification head to pretrain our models on ImageNet [5]. (a) A dual-path transformer block with C intermediate bottleneck channels. (b) The baseline architecture for our ablation studies in Sec. 4.2 of the main paper. (c) MaX-DeepLab-S that matches the number of parameters and M-Adds of DETR-R101-Panoptic [1]. **Axial-Block** (Fig. 7) is an axial-attention bottleneck block borrowed from Axial-DeepLab-L [9]. (d) MaX-DeepLab-L that achieves the state-of-the-art performance on COCO [6]. **Wide-Axial** is a wide version of Axial-Block with doubled intermediate bottleneck channels, similar to the one used in Axial-DeepLab-XL [9]. (The residual connections are dropped for neatness).