Per-Pixel Classification is Not All You Need for Semantic Segmentation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Modern approaches typically formulate semantic segmentation as a *per-pixel classification* task, while instance-level segmentation is handled with an alternative *mask classification*. Our key insight: mask classification is sufficiently general to solve both semantic- and instance-level segmentation tasks in a unified manner using the exact same model, loss, and training procedure. Following this observation, we propose **MaskFormer**, a simple mask classification model which predicts a set of binary masks, each associated with a *single* global class label prediction. Overall, the proposed mask classification-based method simplifies the landscape of effective approaches to semantic and panoptic segmentation tasks and shows excellent empirical results. In particular, we observe that MaskFormer outperforms per-pixel classification baselines when the number of classes is large. Our mask classification-based method outperforms the current state-of-the-art semantic segmentation model by 2.1 mIoU on ADE20K, achieving 55.6 mIoU.

1 Introduction

The goal of semantic segmentation is to partition an image into regions with different semantic categories. Starting from Fully Convolutional Networks (FCNs) work of Long *et al*. [30], most *deep learning-based* semantic segmentation approaches formulate semantic segmentation as *per-pixel classification* (Figure 1 left), applying a classification loss to each output pixel [8, 47]. Per-pixel predictions in this formulation naturally partition an image into regions of different classes.

Mask classification is an alternative paradigm that disentangles the image partitioning and classification aspects of segmentation. Instead of classifying each pixel, mask classification-based methods predict a set of binary masks, each associated with a *single* class prediction (Figure 1 right). The more flexible mask classification dominates the field of instance-level segmentation, since per-pixel classification assumes a static number of outputs and cannot return a variable number of predicted regions/segments, which is required for instance-level tasks. For example, Mask R-CNN [20] and DETR [3] yield a single class prediction per segment for instance and panoptic segmentation.

Our key observation: mask classification is sufficiently general to solve both semantic- and instance-level segmentation tasks. In fact, before FCN [30], the best performing semantic segmentation methods like O2P [4] and SDS [19] used mask classification. Given this perspective, a natural question emerges: *can a single mask classification model simplify the landscape of effective approaches to semantic- and instance-level segmentation tasks?* And can such a model be competitive with per-pixel classification methods for semantic segmentation?

To address both questions we propose a simple **MaskFormer** module that seamlessly converts any existing per-pixel classification model into a mask classification method. Using the set prediction mechanism proposed in DETR [3], MaskFormer employs a Transformer decoder [38] to compute a

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

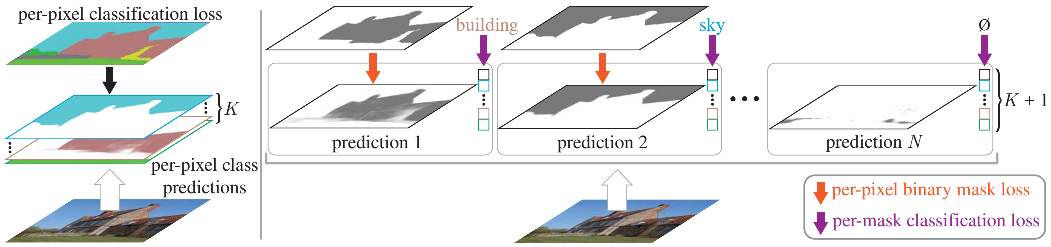


Figure 1: **Per-pixel classification (left) *vs*. mask classification. (left)** Semantic segmentation with per-pixel classification applies the same classification loss to each location. **(right)** Mask classification predicts a set of binary masks and assigns a single class to each mask. Each prediction is supervised with a per-pixel binary mask loss and a classification loss. Matching between the set of predictions and ground truth segments can be done either via *bipartite matching* similarly to DETR [3] or by *fixed matching* via direct indexing if the number of predictions and classes match, *i.e*., if $N = K$.

set of pairs, each consisting of a class prediction and a mask embedding vector. The mask embedding vector is used to get the binary mask prediction via a dot product with the per-pixel embedding obtained from an underlying fully-convolutional network. The new model solves both semantic- and instance-level segmentation tasks in a unified manner: no changes to the model, losses and training procedure are required. Specifically, for semantic and panoptic segmentation tasks alike, MaskFormer is supervised with the same per-pixel binary mask loss and a single classification loss per mask. We also design a simple probabilistic inference to blend the outputs of MaskFormer into a final prediction, which is more efficient than existing heuristics for mask classification [23, 3].

We evaluate MaskFormer on four semantic segmentation datasets with various numbers of categories: Cityscapes [14] (19 classes), ADE20K [50] (150 classes), COCO-Stuff-10K [2] (171 classes), ADE20K-Full [50] (847 classes). While MaskFormer performs on par with per-pixel classification models for Cityscapes, which has a few diverse classes, the new model demonstrates superior performance for datasets with larger vocabulary. We hypothesize that mask classification uses global context which is required for more efficient fine-grained recognition. We observe that MaskFormer achieves the new state-of-the-art on ADE20K (55.6 mIoU) with Swin-Transformer [29] backbone, outperforming the best per-pixel classification model with the same backbone by 2.1 mIoU, while being more efficient (10% reduction in parameters and 40% reduction in FLOPs).

Finally, we study MaskFormer's ability to solve instance-level tasks using two panoptic segmentation datasets: COCO [28, 23] and ADE20K [50]. The new model performs on par with the more complex DETR model [3], highlighting its ability to unify instance- and semantic-level segmentation.

2 Related Works

Both per-pixel classification and mask classification have been extensively studied for semantic segmentation. In early work, Konishi and Yuille [24] apply per-pixel Bayesian classifiers based on local image statistic. Then, inspired by early works on non-semantic groupings [12, 34], mask classification-based methods became popular demonstrating the best performance in PASCAL VOC challenges [17]. For example, methods like O2P [4] and CFM [15] have achieved state-of-the-art results by classifying mask proposals [5, 37, 1]. In 2015, FCN [30] extended the idea of per-pixel classification to deep nets, significantly outperforming all prior methods on mIoU (a per-pixel evaluation metric which particularly suits the per-pixel classification formulation of segmentation).

Per-pixel classification became the dominant way for *deep-net-based* semantic segmentation since the seminal work of Fully Convolutional Networks (FCNs) [30]. Modern semantic segmentation models focus on aggregating long-range context in the final feature map: ASPP [6, 7] uses atrous convolutions with different atrous rates; PPM [47] uses pooling operators with different kernel sizes; DANet [18], OCNet [46] and CCNet [22] use different variants of non-local blocks [40]. Recently, SETR [48] and Segmenter [35] replace traditional convolutional backbones with Vision Transformers (ViT) [16] that capture long-range context starting from the very first input layer. However, these concurrent Transformer-based [38] semantic segmentation approaches still use per-pixel classification. Note, that our MaskFormer module can convert any per-pixel classification model to the mask classification setting, allowing seamless adoption of advances in per-pixel classification.

Mask classification is commonly used for instance-level segmentation [19, 23] these days. These tasks require a dynamic number of predictions, making application of per-pixel classification challenging as it assumes a static number of outputs. Omnipresent Mask R-CNN [20] uses a global classifier to classify mask proposals for instance segmentation. DETR [3] further incorporates a Transformer [38] design to handle thing and stuff segmentation simultaneously for panoptic segmentation [23]. However, these mask classification methods require predictions of bounding boxes, which may limit their usage in semantic segmentation. Max-DeepLab [39] removes the dependence on box predictions for panoptic segmentation with conditional convolutions [36, 41]. However, in addition to the main mask classification losses it requires three auxiliary losses (*i.e.*, instance discrimination loss, mask-ID cross entropy loss and per-pixel classification loss).

3 From Per-Pixel to Mask Classification

In this section, we first describe how semantic segmentation can be formulated as either a per-pixel classification or a mask classification problem. Then, we introduce our instantiation of the mask classification model with the help of a Transformer decoder [38]. Finally, we propose a probabilistic inference strategy to take full advantage of the mask classification formulation.

3.1 Per-pixel classification formulation

For per-pixel classification, a segmentation model aims to predict the probability distribution over all possible K categories for every pixel of an $H \times W$ image: $y = \{p_i | p_i \in \Delta^K\}_{i=1}^{H\cdot W}$. Here Δ^K is the K-dimensional probability simplex. Training a per-pixel classification model is straight-forward: given ground truth category labels $y^{\text{gt}} = \{y_i^{\text{gt}} | y_i^{\text{gt}} \in \{1,\dots,K\}\}_{i=1}^{H\cdot W}$ for every pixel, a per-pixel cross-entropy (negative log-likelihood) loss is usually applied, *i.e.*, $\mathcal{L}_{\text{pixel-cls}}(y, y^{\text{gt}}) = \sum_{i=1}^{H\cdot W} -\log p_i(y_i^{\text{gt}})$.

3.2 Mask classification formulation

Mask classification splits the segmentation task into 1) partitioning/grouping the image into N regions, represented with binary masks $\{m_i | m_i \in [0,1]^{H\times W}\}_{i=1}^{N}$; and 2) associating each region as a whole with some distribution over K categories. To jointly group and classify a segment, *i.e.*, to perform mask classification, we define the desired output z as a set of N probability-mask pairs, *i.e.*, $z = \{(p_i, m_i)\}_{i=1}^{N}$. In contrast to per-pixel class probability prediction, for mask classification the probability distribution $p_i \in \Delta^{K+1}$ contains an auxiliary "no object" label ($\varnothing$) in addition to the K category labels. The $\varnothing$ label is predicted for masks that do not correspond to any of the K categories. Note, mask classification allows multiple mask predictions with the same associated class, making it applicable to both semantic- and instance-level segmentation.

To train a mask classification model, a matching σ between the set of predictions z and the set of N^{gt} ground truth segments $z^{\text{gt}} = \{(c_i^{\text{gt}}, m_i^{\text{gt}}) | c_i^{\text{gt}} \in \{1,\dots,K\}, m_i^{\text{gt}} \in \{0,1\}^{H\times W}\}_{i=1}^{N^{\text{gt}}}$ is required.[1] Here c_i^{gt} is the ground truth class of the i^{th} ground truth segment. Since the size of prediction set $|z| = N$ and ground truth set $|z^{\text{gt}}| = N^{\text{gt}}$ generally differ, we assume $N \geq N^{\text{gt}}$ and pad the set of ground truth labels with "no object" tokens $\varnothing$ to allow one-to-one matching.

For semantic segmentation, a trivial *fixed matching* is possible if the number of predictions N matches the number of category labels K. In this case, the i^{th} prediction is matched to a ground truth region with class label i and to $\varnothing$ if class label i is not present in the ground truth. In our experiments, we found that a *bipartite matching*-based assignment demonstrates better results than the fixed matching. Unlike DETR [3] that uses bounding boxes to compute the assignment costs between prediction z_i and ground truth z_j^{gt} for the Hungarian algorithm [25], we directly use class and mask predictions, *i.e.*, $-p_i(c_j^{\text{gt}}) + \mathcal{L}_{\text{mask}}(m_i, m_j^{\text{gt}})$, where $\mathcal{L}_{\text{mask}}$ is a binary mask loss.

Given a matching, the main mask classification loss $\mathcal{L}_{\text{mask-cls}}$ is composed of a cross-entropy classification loss and a binary mask loss $\mathcal{L}_{\text{mask}}$ for each predicted segment:

$$\mathcal{L}_{\text{mask-cls}}(z, z^{\text{gt}}) = \sum_{j=1}^{N} \left[-\log p_{\sigma(j)}(c_j^{\text{gt}}) + \mathbb{1}_{c_j^{\text{gt}} \neq \varnothing} \mathcal{L}_{\text{mask}}(m_{\sigma(j)}, m_j^{\text{gt}}) \right]. \tag{1}$$

[1] Different mask classification methods utilize various matching rules. For instance, Mask R-CNN [20] uses a heuristic procedure based on anchor boxes and DETR [3] optimizes a bipartite matching between z and z^{gt}.

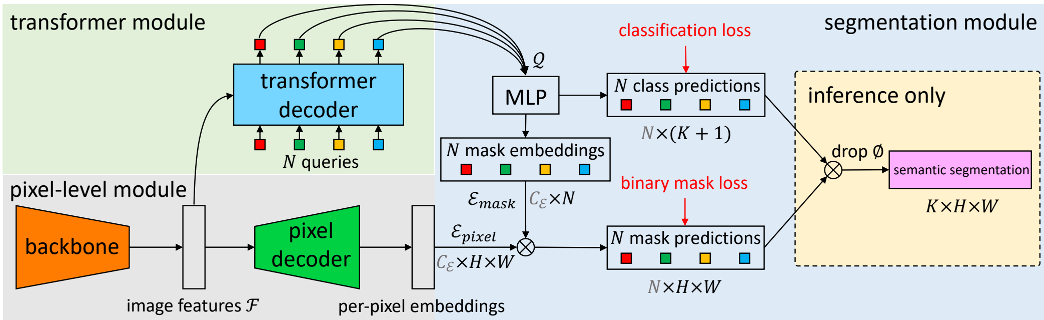


Figure 2: **MaskFormer overview.** We use a backbone to extract image features $\mathcal{F}$. A pixel decoder gradually upsamples image features to extract per-pixel embeddings $\mathcal{E}_{\text{pixel}}$. A transformer decoder attends to image features and produces N per-segment embeddings $\mathcal{Q}$. The embeddings independently generate N class predictions with N corresponding mask embeddings $\mathcal{E}_{\text{mask}}$. Then, the model predicts N possibly overlapping binary mask predictions via a dot product between pixel embeddings $\mathcal{E}_{\text{pixel}}$ and mask embeddings $\mathcal{E}_{\text{mask}}$ followed by a sigmoid activation. Finally, we get semantic segmentations by combining N binary masks with their class predictions using a simple matrix multiplication (see Section 3.4). Note, the dimensions used to perform multiplication $\bigotimes$ are shown in gray.

Note, that most existing mask classification models use auxiliary losses (*e.g.*, a bounding box loss [20, 3] or an instance discrimination loss [39]) in addition to $\mathcal{L}_{\text{mask-cls}}$. In the next section we present a simple mask classification module that allows end-to-end training with $\mathcal{L}_{\text{mask-cls}}$ alone.

3.3 MaskFormer

We now introduce MaskFormer, the new mask classification model, which computes N probability-mask pairs $z = \{(p_i, m_i)\}_{i=1}^N$. The model contains three modules (see Fig. 2): 1) a pixel-level module that extracts per-pixel embeddings used to generate binary mask predictions; 2) a transformer module, where a stack of Transformer decoder layers [38] computes N per-segment embeddings; and 3) a segmentation module, which generates predictions $\{(p_i, m_i)\}_{i=1}^N$ from these embeddings. During inference, discussed in Sec. 3.4, p_i and m_i are assembled into the final prediction.

Pixel-level module takes an image of size $H \times W$ as input. A backbone generates a (typically) low-resolution image feature map $\mathcal{F} \in \mathbb{R}^{C_\mathcal{F} \times \frac{H}{S} \times \frac{W}{S}}$, where $C_\mathcal{F}$ is the number of channels and S is the stride of the feature map ($C_\mathcal{F}$ depends on the specific backbone and we use $S = 32$ in this work). Then, a pixel decoder gradually upsamples the features to generate per-pixel embeddings $\mathcal{E}_{\text{pixel}} \in \mathbb{R}^{C_\mathcal{E} \times H \times W}$, where $C_\mathcal{E}$ is the embedding dimension. Note, that any per-pixel classification-based segmentation model fits the pixel-level module design including recent Transformer-based models [35, 48, 29]. MaskFormer seamlessly converts such a model to mask classification.

Transformer module uses the standard Transformer decoder [38] to compute from image features $\mathcal{F}$ and N learnable positional embeddings (*i.e.*, queries) its output, *i.e.*, N per-segment embeddings $\mathcal{Q} \in \mathbb{R}^{C_\mathcal{Q} \times N}$ of dimension $C_\mathcal{Q}$ that encode global information about each segment MaskFormer predicts. Similarly to [3], the decoder yields all predictions in parallel.

Segmentation module applies a linear classifier, followed by a softmax activation, on top of the per-segment embeddings $\mathcal{Q}$ to yield class probability predictions $\{p_i \in \Delta^{K+1}\}_{i=1}^N$ for each segment. Note, that the classifier predicts an additional "no object" category ($\varnothing$) in case the embedding does not correspond to any region. For mask prediction, a Multi-Layer Perceptron (MLP) with 2 hidden layers converts the per-segment embeddings $\mathcal{Q}$ to N mask embeddings $\mathcal{E}_{\text{mask}} \in \mathbb{R}^{C_\mathcal{E} \times N}$ of dimension $C_\mathcal{E}$. Finally, we obtain each binary mask prediction $m_i \in [0,1]^{H \times W}$ via a dot product between the i^{th} mask embedding and per-pixel embeddings $\mathcal{E}_{\text{pixel}}$ computed by the pixel-level module. The dot product is followed by a sigmoid activation, *i.e.*, $m_i[h, w] = \text{sigmoid}(\mathcal{E}_{\text{mask}}[:, i]^{\text{T}} \cdot \mathcal{E}_{\text{pixel}}[:, h, w])$.

Note, we empirically find it is beneficial to *not* enforce mask predictions to be mutually exclusive to each other by using a softmax activation. During training, the $\mathcal{L}_{\text{mask-cls}}$ loss combines a cross entropy classification loss and a binary mask loss $\mathcal{L}_{\text{mask}}$ for each predicted segment. For simplicity we use the same $\mathcal{L}_{\text{mask}}$ as DETR [3], *i.e.*, a linear combination of a focal loss [27] and a dice loss [32] multiplied by hyper-parameters λ_{focal} and λ_{dice} respectively.

3.4 Mask-classification inference

First, we present a simple *general inference* procedure that converts mask classification outputs $\{(p_i, m_i)\}_{i=1}^N$ to either panoptic or semantic segmentation output formats. Then, we describe a *probabilistic inference* procedure specifically designed for semantic segmentation.

General inference partitions an image into segments by assigning each pixel $[h, w]$ to one of the N predicted probability-mask pairs via $\arg\max_{i:c_i \neq \varnothing} p_i(c_i) \cdot m_i[h, w]$. Here c_i is the most likely class label $c_i = \arg\max_{c \in \{1,\dots,K,\varnothing\}} p_i(c)$ for each probability-mask pair i. Intuitively, this procedure assigns a pixel at location $[h, w]$ to probability-mask pair i only if both the *most likely* class probability $p_i(c_i)$ and the mask prediction probability $m_i[h, w]$ are high. Pixels assigned to the same probability-mask pair i form a segment where each pixel is labelled with c_i. For semantic segmentation, segments sharing the same category label are merged; whereas for instance-level segmentation tasks, the index i of the probability-mask pair helps to distinguish different instances of the same class.

Probabilistic inference is designed specifically for semantic segmentation and is done via a simple matrix multiplication. We empirically find that marginalization over probability-mask pairs, *i.e.*, $\arg\max_{c \in \{1,\dots,K\}} \sum_{i=1}^N p_i(c) \cdot m_i[h, w]$, yields better results than the hard assignment of each pixel to a probability-mask pair i used in general inference strategy. The argmax does not include the "no object" category ($\varnothing$) as standard semantic segmentation requires each output pixel to take a label. Note, that probabilistic inference returns a per-pixel class probability $\sum_{i=1}^N p_i(c) \cdot m_i[h, w]$ similarly to per-pixel classification. However, we empirically observe directly maximizing per-pixel class likelihood for MaskFormer leads to poor performance. We hypothesize, that in this case gradients are evenly distributed to every query, which complicates training.

4 Experiments

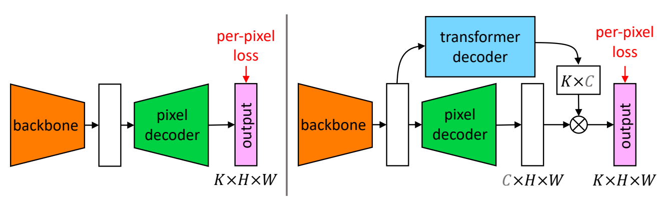
First, we compare mask classification-based MaskFormer with state-of-the-art methods on multiple semantic segmentation datasets. Then, we show that the same model achieves competitive performance on panoptic segmentation. Finally, we ablate the MaskFormer design confirming that observed improvements indeed stem from the shift from per-pixel classification to mask classification.

Datasets. We study MaskFormer using three widely used semantic segmentation datasets: ADE20K [50] (150 classes) from the SceneParse150 challenge [49], COCO-stuff-10K [2] (171 classes), and Cityscapes [14] (19 classes). In addition, we use the ADE20K-Full [50] dataset annotated in an open vocabulary setting (874 classes are present in both train and validation sets).

For panotic segmenation evaluation we use COCO [28, 2, 23] (80 "things" and 53 "stuff" categories) and ADE20K-Panoptic [50, 23] (100 "things" and 50 "stuff" categories). Please see the supplementary material for detailed descriptions of all used semantic and panoptic segmentation datasets.

Evaluation metrics. For semantic segmentation the standard metric is **mIoU** (mean Intersection-over-Union) [17], a per-pixel metric that directly corresponds to the per-pixel classification formulation. To better illustrate the difference between segmentation approaches, in our ablations we supplement mIoU with **PQ$^{\text{St}}$** (PQ stuff) [23], a per-region metric that treats all classes as "stuff" and evaluates each segment equally, irrespective of its size. We report the median of 3 runs for all datasets, except for Cityscapes where we report the median of 5 runs. For panoptic segmentation, we use the standard **PQ** (panoptic quality) metric [23] and report single run results due to prohibitive training costs.

Baseline models. On the right we sketch the used per-pixel classification baselines. The **PerPixelBaseline** uses the pixel-level module of MaskFormer and directly outputs per-pixel class scores. For a fair comparison, we design **PerPixelBaseline+** which adds the transformer module and mask embedding MLP to the PerPixelBaseline. Thus, PerPixelBaseline+ and MaskFormer differ only in the formulation: per-pixel *vs*. mask classification. Note that, these baselines are for ablation and we compare MaskFormer with state-of-the-art per-pixel classification models as well.



(a) PerPixelBaseline (b) PerPixelBaseline+

4.1 Implementation details

Backbone. MaskFormer is compatible with any backbone architecture. In our work we use the standard convolution-based ResNet [21] backbones (R50 and R101 with 50 and 101 layers respectively) and recently proposed Transformer-based Swin-Transformer [29] backbones. In addition, we use the R101c model [6] which replaces the first 7×7 convolution layer of R101 with 3 consecutive 3×3 convolutions and which is popular in the semantic segmentation community.

Pixel decoder. The pixel decoder in Figure 2 can be implemented using any semantic segmentation decoder (*e.g.*, [8–10]). Many per-pixel classification methods use modules like ASPP [6] or PSP [47] to collect and distribute context across locations. In our experiments, we observe that such modules do not improve MaskFormer. The Transformer module attends to all image features, collecting global information to generate class predictions. This setup reduces the need of the per-pixel module for heavy context aggregation that distributes global information to each pixel. Therefore, for MaskFormer, we design a light-weight pixel decoder based on the popular FPN [26] architecture.

Following FPN, we $2\times$ upsample the low-resolution feature map in the decoder and sum it with the projected feature map of corresponding resolution from the backbone; Projection is done to match channel dimensions of the feature maps with a 1×1 convolution layer followed by GroupNorm (GN) [42]. Next, we fuse the summed features with an additional 3×3 convolution layer followed by GN and ReLU activation. We repeat this process starting with the stride 32 feature map until we obtain a final feature map of stride 4. Finally, we apply a single 1×1 convolution layer to get the per-pixel embeddings. All feature maps in the pixel decoder have a dimension of 256 channels.

Transformer decoder. We use the same Transformer decoder design as DETR [3]. The N query embeddings are initialized as zero vectors, and we associate each query with a learnable positional encoding. We use 6 Transformer decoder layers with 100 queries by default, and, following DETR, we apply the same loss after each decoder. In our experiments we observe that MaskFormer is competitive for semantic segmentation with a single decoder layer too, whereas for instance-level segmentation multiple layers are necessary to remove duplicates from the final predictions.

Segmentation module. The multi-layer perceptron (MLP) in Figure 2 has 2 hidden layers of 256 channels to predict the mask embeddings $\mathcal{E}_{\text{mask}}$, analogously to the box head in DETR. Both per-pixel $\mathcal{E}_{\text{pixel}}$ and mask $\mathcal{E}_{\text{mask}}$ embeddings have 256 channels.

Loss weights. We use focal loss [27] and dice loss [32] for our mask loss: $\mathcal{L}_{\text{mask}}(m, m^{\text{gt}}) = \lambda_{\text{focal}}\mathcal{L}_{\text{focal}}(m, m^{\text{gt}}) + \lambda_{\text{dice}}\mathcal{L}_{\text{dice}}(m, m^{\text{gt}})$, and set the hyper-parameters to $\lambda_{\text{focal}} = 20.0$ and $\lambda_{\text{dice}} = 1.0$. Following DETR [3], the weight for the "no object" ($\varnothing$) in the classification loss is set to 0.1.

4.2 Training settings

Semantic segmentation. We use Detectron2 [43] and follow the commonly used training settings for each dataset. We use AdamW [31] and the *poly* [6] learning rate schedule with an initial learning rate of 10^{-4} and a weight decay of 10^{-4} for ResNet [21] backbones, and an initial learning rate of $6 \cdot 10^{-5}$ and a weight decay of 10^{-2} for Swin-Transformer [29] backbones. Backbones are pre-trained on ImageNet-1K [33] if not stated otherwise. A learning rate multiplier of 0.1 is applied to CNN backbones and 1.0 is applied to Transformer backbones. The standard random scale jittering between 0.5 and 2.0, random horizontal flipping, random cropping as well as random color jittering are used as data augmentation. For the ADE20K dataset, if not stated otherwise, we use a crop size of 512×512, a batch size of 16 and train all models for 160k iterations. For the ADE20K-Full dataset, we use the same setting as ADE20K except that we train all models for 200k iterations. For the COCO-stuff-10k dataset, we use a crop size of 640×640, a batch size of 32 and train all models for 60k iterations. All models are trained with 8 V100 GPUs. We report both performance of single scale (s.s.) inference and multi-scale (m.s.) inference with horizontal flip and scales of 0.5, 0.75, 1.0, 1.25, 1.5, 1.75. By default, we use the probabilistic inference strategy discussed in Section 3.4.

Panoptic segmentation. We follow exactly the same architecture, loss and training procedure as we use for semantic segmentation. The only difference is supervision: *i.e.*, category region masks in semantic segmentation *vs*. object instance masks in panoptic segmentation. We strictly follow the DETR [3] setting to train our model on the COCO panoptic segmentation dataset [23] for a fair comparison. On the ADE20K panoptic segmentation dataset, we follow the semantic segmentation setting but train for longer (720k iterations) and use a larger crop size (640×640). COCO models

Table 1: **Semantic segmentation on ADE20K `val` with 150 categories.** Mask classification-based MaskFormer outperforms the best per-pixel classification approaches while using fewer parameters and less computation. We report both single-scale (s.s.) and multi-scale (m.s.) inference results with $\pm$*std*. FLOPs are computed for the given crop size. Frame-per-second (fps) is measured on a V100 GPU with a batch size of 1.[2] Backbones pre-trained on ImageNet-22K are marked with †.

	method	backbone	crop size	mIoU (s.s.)	mIoU (m.s.)	#params.	FLOPs	fps
CNN backbones	OCRNet [45]	R101c	520×520	-	45.3	-	-	-
	DeepLabV3+ [8]	R50c	512×512	44.0	44.9	44M	177G	21.0
		R101c	512×512	45.5	46.4	63M	255G	14.2
	PerPixelBaseline	R50	512×512	39.2 $\pm$0.2	40.9 $\pm$0.1	31M	48G	45.5
	PerPixelBaseline+	R50	512×512	41.9 $\pm$0.2	42.9 $\pm$0.3	41M	50G	30.3
	MaskFormer (ours)	R50	512×512	44.4 $\pm$0.5	46.6 $\pm$0.6	41M	53G	24.5
		R101	512×512	45.1 $\pm$0.5	47.1 $\pm$0.2	60M	73G	19.5
		R101c	512×512	**45.9** $\pm$0.1	**48.0** $\pm$0.2	60M	80G	19.0
Transformer backbones	SETR [48]	ViT-L†	512×512	-	50.3	308M	-	-
	Swin-UperNet [29, 44]	Swin-T	512×512	-	46.1	60M	236G	18.5
		Swin-S	512×512	-	49.3	81M	259G	15.2
		Swin-B†	640×640	-	51.6	121M	471G	8.7
		Swin-L†	640×640	-	53.5	234M	647G	6.2
	MaskFormer (ours)	Swin-T	512×512	46.7 $\pm$0.7	48.8 $\pm$0.6	42M	55G	22.1
		Swin-S	512×512	49.8 $\pm$0.4	51.0 $\pm$0.4	63M	79G	19.6
		Swin-B†	640×640	52.7 $\pm$0.4	54.0 $\pm$0.2	102M	195G	12.6
		Swin-L†	640×640	**54.1** $\pm$0.2	**55.6** $\pm$0.1	212M	375G	7.9

are trained using 64 V100 GPUs and ADE20K experiments are trained with 8 V100 GPUs. During inference, we use the general inference strategy discussed in Section 3.4 with the following parameters: 1) we filter masks with class confidence below 0.7; 2) we set masks whose contribution to the final panoptic segmentation is less than 80% of its mask area to VOID. We only report performance of single scale inference for panoptic segmentation.

4.3 Main results

Semantic segmentation. In Table 1, we compare MaskFormer with our baselines and state-of-the-art per-pixel classification models for semantic segmentation on the ADE20K dataset. With the same standard CNN backbones (*e.g.*, ResNet [21]), MaskFormer outperforms DeepLabV3+ [8]. MaskFormer is also compatible with recent Vision Transformer [16] backbones (*e.g.*, the Swin Transformer [29]), achieving a new state-of-the-art of 55.6 mIoU, which is 2.1 mIoU better than the prior state-of-the-art. Observe that MaskFormer outperforms the best per-pixel classification-based models while having fewer parameters and faster inference time. This result suggests that the mask classification formulation has significant potential for semantic segmentation.

Beyond ADE20K, we compare MaskFormer with existing per-pixel classification models and our baselines on COCO-stuff-10K and ADE20K-Full. The new mask classification model achieves competitive results on both datasets. Using ADE20K-Full which has 847 categories we observe that MaskFormer is more memory efficient than per-pixel classification baselines that make 847 class predictions for each pixel. In contrast, mask classification-based MaskFormer makes such class predictions for each mask only (100 by default in our experiments). We refer to the supplementary material for a detailed description of our experiments on these two datasets.

In Table 2a, we report MaskFormer performance on Cityscapes, the standard testbed for modern semantic segmentation methods. The dataset has only 19 categories and therefore, the recognition aspect of the dataset is less challenging than in other considered datasets. We observe that MaskFormer performs on par with the best per-pixel classification methods. To better analyze MaskFormer, in Table 2b, we further report PQ^{St} by treating every category as "stuff". Unlike mIoU, this metric treats all segments equally, irrespective of their size, and allows to evaluate recognition quality. MaskFormer performs better in terms of recognition quality (RQ^{St}) while lagging in per-pixel segmentation quality (SQ^{St}). This observation suggests that on datasets, where recognition is relatively easy to solve, the main challenge for mask classification-based approaches is pixel-level accuracy.

[2] It isn't recommended to compare fps from different papers: speed is measured in different environments. DeepLabV3+ fps are from MMSegmentation [13], and Swin-UperNet fps are from the original paper [29].

Table 2: **Semantic segmentation on Cityscapes** `val` **with 19 categories.** 2a: MaskFormer is on-par with state-of-the-art methods on Cityscapes which has fewer categories than other considered datasets. We report multi-scale (m.s.) inference results with $\pm std$ for a fair comparison across methods. 2b: We analyze MaskFormer with a complimentary PQ^{St} metric, by treating all categories as "stuff." The breakdown of PQ^{St} suggests mask classification-based MaskFormer is better at recognizing regions (RQ^{St}) while slightly lagging in generation of high-quality masks (SQ^{St}).

(a) Cityscapes standard **mIoU** metric.

method	backbone	mIoU (m.s.)
Panoptic-DeepLab [10]	X71 [11]	81.5
OCRNet [45]	R101c	**82.0**
MaskFormer (ours)	R101	80.5 $\pm$0.1
	R101c	81.4 $\pm$0.2

(b) Cityscapes analysis with $\mathbf{PQ^{St}}$ metric suit.

PQ^{St} (m.s.)	SQ^{St} (m.s.)	RQ^{St} (m.s.)
66.6	**82.9**	79.4
66.1	82.6	79.1
65.9	81.5	79.7
66.9	82.0	**80.5**

Table 3: **Panoptic segmentation on COCO panoptic** `val` **with 133 categories.** MaskFormer seamlessly unifies semantic- and instance-level segmentation without modifying the model architecture or loss. Our model, which achieves better results, can be regarded as a box-free simplification of DETR [3]. The major improvement comes from "stuff" classes (PQ^{St}) which are ambiguous to represent with bounding boxes. However, our model performs slightly worse than DETR for "thing" classes (PQ^{Th}). We hypothesize that matching instances with boxes is more reliable than masks, which suggests there is room for improvement. Note, for a fair comparison with DETR, we add 6 additional Transformer encoders (6 Enc) after the ResNet [21] backbones.

method	backbone	PQ	PQ^{Th}	PQ^{St}	SQ	RQ
DETR [3]	R50 + 6 Enc	43.4	**48.2**	36.3	79.3	53.8
MaskFormer (ours)		**44.3**	48.0 (-0.2)	**38.7** (+2.4)	**80.3**	**54.1**
DETR [3]	R101 + 6 Enc	45.1	**50.5**	37.0	79.9	55.5
MaskFormer (ours)		**45.7**	49.7 (-0.8)	**39.8** (+2.8)	**80.6**	**55.6**

Panoptic segmentation. In Table 3, we compare the same exact MaskFormer model with DETR [3] on the COCO panoptic dataset. For a fair comparison, we add 6 additional Transformer encoder layers after the CNN backbone. Unlike DETR, our model does not predict bounding boxes but instead predicts masks directly. The overall performance of our model is similar to DETR. Interestingly, we observe a large improvement in PQ^{St} compared to DETR. This suggests that detecting "stuff" with bounding boxes is suboptimal, and therefore, box-based segmentation models (*e.g.*, Mask R-CNN [20]) do no suit semantic segmentation. In contrast, we find that MaskFormer slightly lags behind DETR in terms of PQ^{Th}. This observation indicates that instance-level segmentation benefits from either predicting boxes or using them during matching, suggesting a room for improvement in the matching process or the mask loss used in MaskFormer.

We further evaluate our model on the panoptic segmentation version of ADE20K dataset. Our model is competitive to state-of-the-art methods and we refer to the supplementary for detailed results.

4.4 Ablation studies

We perform a series of ablation studies of MaskFormer for semantic segmentation using a single ResNet-50 backbone [21]. As the standard mIoU metric is computed per-pixel at a dataset level, it neither rewards a model for correctly recognizing small segments nor penalizes it for small false positive predictions. Thus, in addition to mIoU, we compute a complementary PQ^{St} metric by treating every category as "stuff." This metric is computed per-segment and treats all segments equally, irrespective of their size, allowing to evaluate recognition quality of semantic segmentation methods.

Per-pixel *vs*. mask classification. In Table 4, we verify that the gains demonstrated by MaskFromer come from shifting the paradigm to mask classification. We start by comparing PerPixelBaseline+ and MaskFormer. The models are very similar and there are only 3 differences: 1) per-pixel *vs*. mask classification used by the models, 2) MaskFormer uses bipartite matching, and 3) the new model uses a combination of focal and dice losses as a mask loss, whereas PerPixelBaseline+ utilizes per-pixel cross entropy loss. First, we rule out the influence of loss differences by training PerPixelBaseline+ with exactly the same losses and observing no improvement. Next, in Table 4a, we compare PerPixelBaseline+ with MaskFormer trained using a fixed matching (MaskFormer-fixed), *i.e.*, matching mask prediction to ground truth segments by category index identically to per-pixel

Table 4: **Per-pixel *vs*. mask classification for semantic segmentation.** All models use 150 queries for a fair comparison. We evaluate the models on ADE20K `val` with 150 categories. 4a: PerPixel-Baseline+ and MaskFormer-fixed use similar fixed matching (*i.e*., matching by category index), this result confirms that the shift from per-pixel to masks classification is the key. 4a: bipartite matching is not only more flexible (can make less prediction than total class count) but also gives better results.

(a) Per-pixel *vs*. mask classification.

	mIoU	PQ^{St}
PerPixelBaseline+	41.9	28.3
MaskFormer-fixed	**43.7** (+1.8)	**30.3** (+2.0)

(b) Fixed *vs*. bipartite matching assignment.

	mIoU	PQ^{St}
MaskFormer-fixed	43.7	30.3
MaskFormer-bipartite (ours)	**44.2** (+0.5)	**33.4** (+3.1)

Table 5: **Inference strategies for semantic segmentation.** *general:* general inference (Section 3.4) which first filters low-confidence masks (using a threshold of 0.3) and assigns labels to remaining ones. *probabilistic:* the proposed probabilistic inference (Section 3.4). + *iterative:* our iterative probabilistic inference removes masks whose contribution to the final segmentation is less than 30% of its mask area. Probabilistic inference has a clear advantage over general inference in terms of mIoU. However, general inference has higher PQ due to better recognition quality (RQ). Iterative inference reduces the number of false positives by removing overlapping masks from the final prediction.

	ADE20K (150 classes)				COCO-Stuff (171 classes)				ADE20K-Full (847 classes)			
inference	mIoU	PQ^{St}	SQ^{St}	RQ^{St}	mIoU	PQ^{St}	SQ^{St}	RQ^{St}	mIoU	PQ^{St}	SQ^{St}	RQ^{St}
PerPixelBaseline+	41.9	28.3	71.9	36.2	34.2	24.6	62.6	31.2	13.9	9.0	24.5	12.0
general	42.4	34.2	74.4	43.5	35.5	29.7	66.3	37.0	15.1	11.6	28.3	15.3
probabilistic	44.4	33.4	**75.4**	42.4	37.1	28.9	66.3	35.9	**16.0**	11.9	28.6	15.7
+ iterative	**44.7**	**36.6**	75.3	**46.5**	**37.3**	**31.3**	**66.5**	**38.9**	15.8	**13.0**	**28.7**	**17.0**

cross entropy loss. We observe that MaskFormer-fixed is 1.8 mIoU better than the baseline, suggesting that shifting from per-pixel classification to mask classification is indeed the main reason for the gains of MaskFormer. In Table 4b, we further compare MaskFormer-fixed with MaskFormer trained with bipartite matching (MaskFormer-bipartite) and find bipartite matching is not only more flexible (allowing to predict less masks than the total number of categories) but also gives better results.

Number of queries. In the supplementary material we report the impact of the number of predictions for mask classification. We observe the model that predicts $N = 100$ masks consistently performs the best across datasets with different numbers of classes, suggesting that N is a stable hyper-parameter.

Inference strategies for semantic segmentation. In Table 5, we ablate inference strategies for mask classification-based models performing semantic segmentation (discussed in Section 3.4). We start with the general inference strategy which first filters out low-confidence masks (a threshold of 0.3 is used) and assigns the class labels to remaining masks. We observe 1) general inference is only slightly better than the PerPixelBaseline+ in terms of the mIoU metric, and 2) on multiple datasets the general inference strategy performs worse in terms of the mIoU metric than our proposed probabilistic inference. However, the general inference has higher PQ^{St}, due to better recognition quality (RQ^{St}). We hypothesize that the filtering step removes false positives which increases the RQ^{St}. Motivated by this observation, we further propose an iterative probabilistic inference which combines both advantages of general and probabilistic inference. Instead of removing masks by confidence scores, our iterative inference strategy removes a mask if its contribution to the final semantic segmentation output is less than 30% of its mask area. The iterative probabilistic inference improves both mIoU and PQ^{St}. However, it slows down inference due to its iterative nature. Thus, by default in this paper, we use probabilistic inference that can be done via a simple matrix multiplication.

5 Conclusion

The paradigm discrepancy between semantic and instance-level segmentation results in entirely different models for each task, hindering development of image segmentation as a whole. We show that a simple mask classification model can outperform state-of-the-art per-pixel classification models, especially in the presence of large number of categories. Our model also remains competitive for panoptic segmentation, without a need to change model architecture, losses or training procedure. We hope this unification spurs a joint effort across semantic- and instance-level segmentation.

References

[1] Pablo Arbeláez, Jordi Pont-Tuset, Jonathan T Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping. In *CVPR*, 2014. 2

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

[3] 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, 3, 4, 6, 8

[4] Joao Carreira, Rui Caseiro, Jorge Batista, and Cristian Sminchisescu. Semantic segmentation with second-order pooling. In *ECCV*, 2012. 1, 2

[5] Joao Carreira and Cristian Sminchisescu. CPMC: Automatic object segmentation using constrained parametric min-cuts. *PAMI*, 2011. 2

[6] 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. *PAMI*, 2018. 2, 6

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

[8] 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. 1, 6, 7

[9] Bowen Cheng, Liang-Chieh Chen, Yunchao Wei, Yukun Zhu, Zilong Huang, Jinjun Xiong, Thomas S Huang, Wen-Mei Hwu, and Honghui Shi. SPGNet: Semantic prediction guidance for scene parsing. In *ICCV*, 2019.

[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. 6, 8

[11] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017. 8

[12] Dorin Comaniciu and Peter Meer. Robust Analysis of Feature Spaces: Color Image Segmentation. In *CVPR*, 1997. 2

[13] MMSegmentation Contributors. MMSegmentation: OpenMMLab semantic segmentation toolbox and benchmark. `https://github.com/open-mmlab/mmsegmentation`, 2020. 7

[14] 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

[15] Jifeng Dai, Kaiming He, and Jian Sun. Convolutional feature masking for joint object and stuff segmentation. In *CVPR*, 2015. 2

[16] 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, 7

[17] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The PASCAL visual object classes challenge: A retrospective. *IJCV*, 2015. 2, 5

[18] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *CVPR*, 2019. 2

[19] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *ECCV*, 2014. 1, 3

[20] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 1, 3, 4, 8

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6, 7, 8

[22] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. CCNet: Criss-cross attention for semantic segmentation. In *ICCV*, 2019. 2

[23] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *CVPR*, 2019. 2, 3, 5, 6

[24] Scott Konishi and Alan Yuille. Statistical Cues for Domain Specific Image Segmentation with Performance Analysis. In *CVPR*, 2000. 2

[25] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 1955. 3

[26] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 6

[27] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 4, 6

[28] 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

[29] 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. 2, 4, 6, 7

[30] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1, 2

[31] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 6

[32] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-Net: Fully convolutional neural networks for volumetric medical image segmentation. In *3DV*, 2016. 4, 6

[33] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015. 6

[34] Jianbo Shi and Jitendra Malik. Normalized Cuts and Image Segmentation. *PAMI*, 2000. 2

[35] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. *arXiv preprint arXiv:2105.05633*, 2021. 2, 4

[36] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. In *ECCV*, 2020. 3

[37] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *IJCV*, 2013. 2

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 1, 2, 3, 4

[39] 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. 3, 4

[40] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. 2

[41] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. SOLOv2: Dynamic and fast instance segmentation. *NeurIPS*, 2020. 3

[42] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018. 6

[43] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 6

[44] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, 2018. 7

[45] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. In *ECCV*, 2020. 7, 8

[46] Yuhui Yuan, Lang Huang, Jianyuan Guo, Chao Zhang, Xilin Chen, and Jingdong Wang. OCNet: Object context network for scene parsing. *arXiv preprint arXiv:1809.00916*, 2018. 2

[47] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. 1, 2, 6

[48] 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, 4, 7

[49] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing challenge 2016. http://sceneparsing.csail.mit.edu/index_challenge.html, 2016. 5

[50] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *CVPR*, 2017. 2, 5

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Section 3 describes per-pixel and mask classification formally and introduces a new mask classification model MaskFormer. Experimental evaluation in Section 4 supports the claims described in the abstract.
   (b) Did you describe the limitations of your work? [Yes] See Section 4, we show that for a datasets with a small number of classes per-pixel classification methods perform on par with the proposed model.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We study a classical task.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We did not include our code in the supplemental material because we did not have time to remove author identity from our code before the submission deadline. But we will release our code in the future.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4.2.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We run all experiments at least 3 times and report median, see Section 4, *Evaluation metrics*.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 4.2.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cite creators of ADE20K, Cityscapes, COCO-Stuff-10K, and COCO.
   (b) Did you mention the license of the assets? [No] The dataset are standard in our community, their licence information can be obtained via cited references.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No] We are using existing datasets. We did not include our code in the supplemental material because we did not have time to remove author identity from our code before the submission deadline. But we will release our code on in the future.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] The dataset we use are standard in our community. Their are known to have issues in this regard yet they are currently indespensible for publishing in this research area.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] The dataset we use are standard in our community. Their are known to have issues in this regard yet they are currently indespensible for publishing in this research area.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]