Segmentation with diffusion

A Generalist Framework for Panoptic Segmentation of Images and Videos

Ting Chen, Lala Li, Saurabh Saxena, Geoffrey Hinton[†], David J. Fleet[†]
Google Research, Brain Team
{iamtingchen,lala,srbs,geoffhinton,davidfleet}@google.com

Diffusion with discrete data

Original DDPM: only generate continuous data
Sampling:

```
def p_sample(self, xt, t):
    # xt: H×W
    eps_theta = self.eps_model(xt, t)    # ϵ_θ(x_t, t)
    alpha_bar = gather(self.alpha_bar, t) #gather ᾱ_t
    alpha = gather(self.alpha, t) #gather α_t
    eps_coef = (1 - alpha) / (1 - alpha_bar) ** .5
    mean = 1 / (alpha ** 0.5) * (xt - eps_coef * eps_theta) #μ_θ can be further derived from ϵ_θ
    var = gather(self.sigma2, t)
    eps = torch.randn(xt.shape, device=xt.device)
    return mean + (var ** .5) * eps #sample from 𝒩(μ_θ, σ)
```

Diffusion with discrete data

Bit Diffusion:

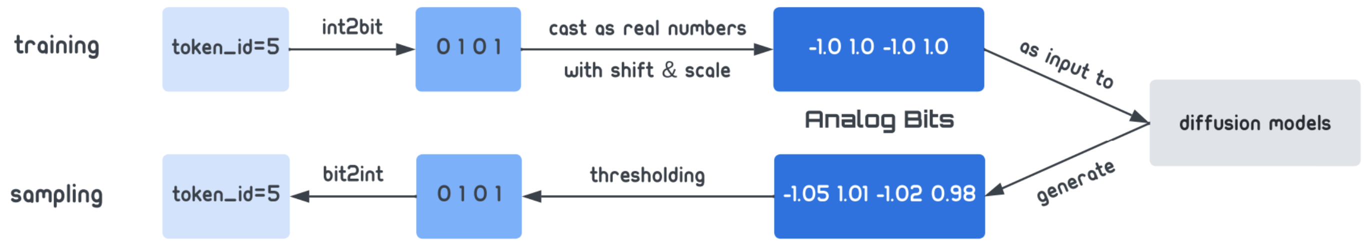


Figure 1: Bit Diffusion: modeling discrete data using continuous diffusion models with analog bits.

Different discrete encoding: (1) UINT8 (2) GRAY CODE (3) UINT8 RAND

Table 2: Comparison of FIDs on class-conditional IMAGENET 64×64. The corresponding samples can be found in Figure 4 and 11.

DDPM (our repo.) on continuous pixels	Bit Diffusion on UINT8	Bit Diffusion on GRAY CODE	Bit Diffusion on UINT8 (RAND)
3.43	4.84	5.14	8.76

Segmentation with diffusion

Image encoder: ResNet + Transformer Encoder + FPN
Mask decoder: TransUNet

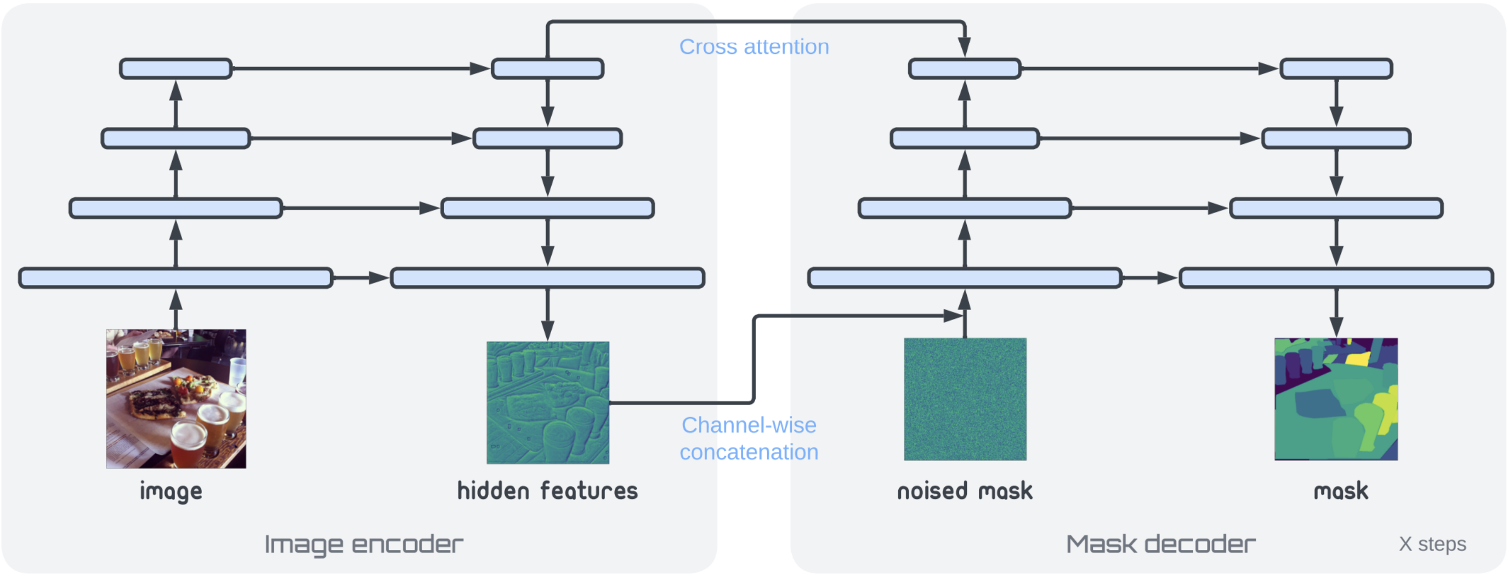
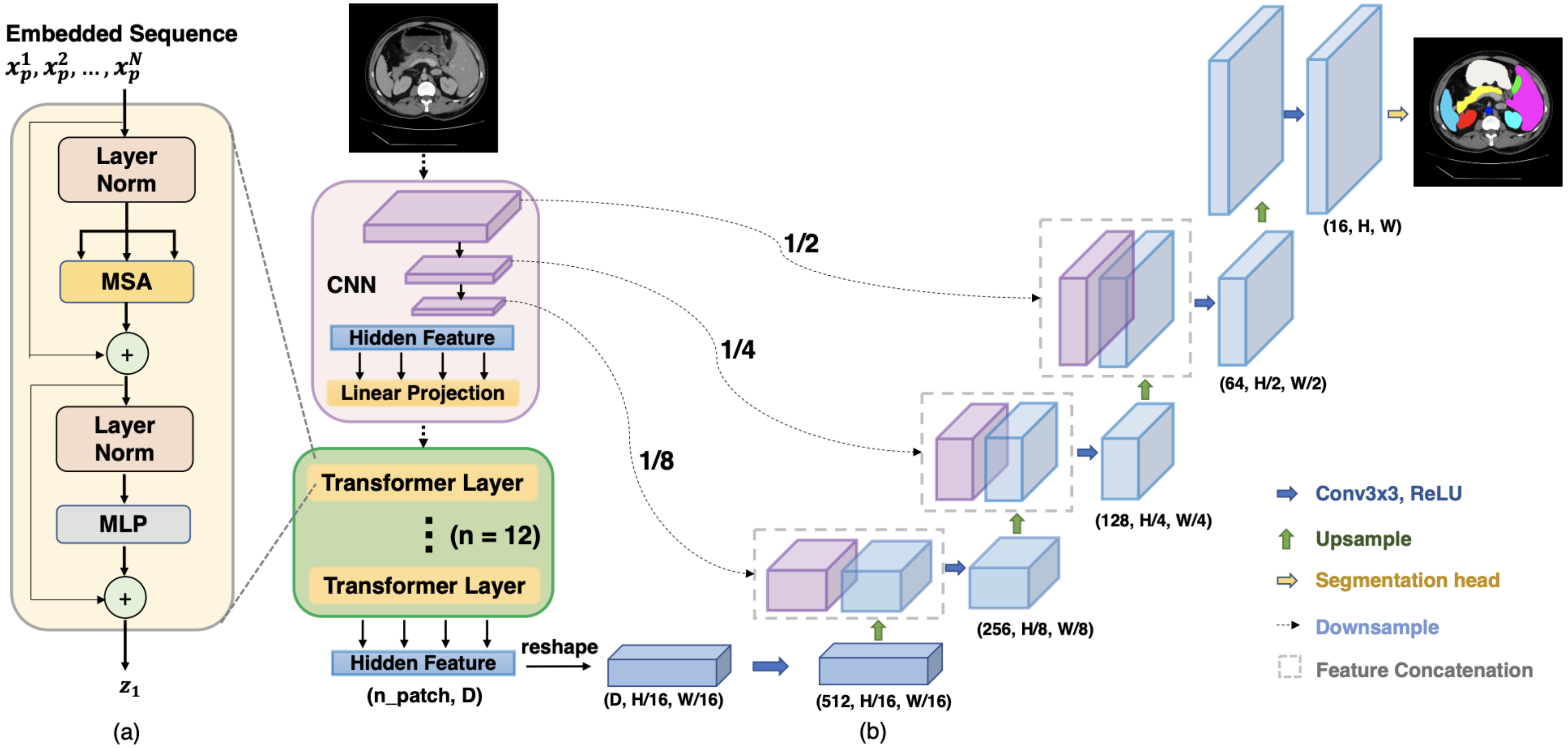


Figure 2. The architecture for our panoptic mask generation framework. We separate the model into image encoder and mask decoder so that the iterative inference at test time only involves multiple passes over the decoder.

Segmentation with diffusion

Mask decoder: TransUNet



Segmentation with diffusion

Training Algorithm

Algorithm 1 Pix2Seq-$\mathcal{D}$ training algorithm.

```
def train_loss(images, masks):
  """images: [b, h, w, 3], masks: [b, h', w', 2]."""

  # Encode image features.
  h = pixel_encoder(images)

  # Discrete masks to analog bits.
  m_bits = int2bit(masks).astype(float)
  m_bits = (m_bits * 2 - 1) * scale

  # Corrupt analog bits.
  t = uniform(0, 1)               # scalar.
  eps = normal(mean=0, std=1) # same shape as m_bits.
  m_crpt = sqrt(    gamma(t)) * m_bits +
           sqrt(1 - gamma(t)) * eps

  # Predict and compute loss.
  m_logits, _ = mask_decoder(m_crpt, h, t)
  loss = cross_entropy(m_logits, masks)

  return loss.mean()
```

Algorithm 2 Pix2Seq-$\mathcal{D}$ inference algorithm.

```
def infer(images, steps=10, td=1.0):
  """images: [b, h, w, 3]."""

  # Encode image features.
  h = pixel_encoder(images)

  m_t = normal(mean=0, std=1) # same shape as m_bits.
  for step in range(steps):
    # Get time for current and next states.
    t_now = 1 - step / steps
    t_next = max(1 - (step + 1 + td) / steps, 0)

    # Predict analog bits m_0 from m_t.
    _, m_pred = mask_decoder(m_t, h, t_now)

    # Estimate m at t_next.
    m_t = ddim_step(m_t, m_pred, t_now, t_next)

  # Analog bits to masks.
  masks = bit2int(m_pred > 0)
  return masks
```

Segmentation with diffusion

Input Scaling:

Analog Bits: scale the data into {-b, b}, where b is set to be 1

Ours: A smaller b will lead to samaller SNR

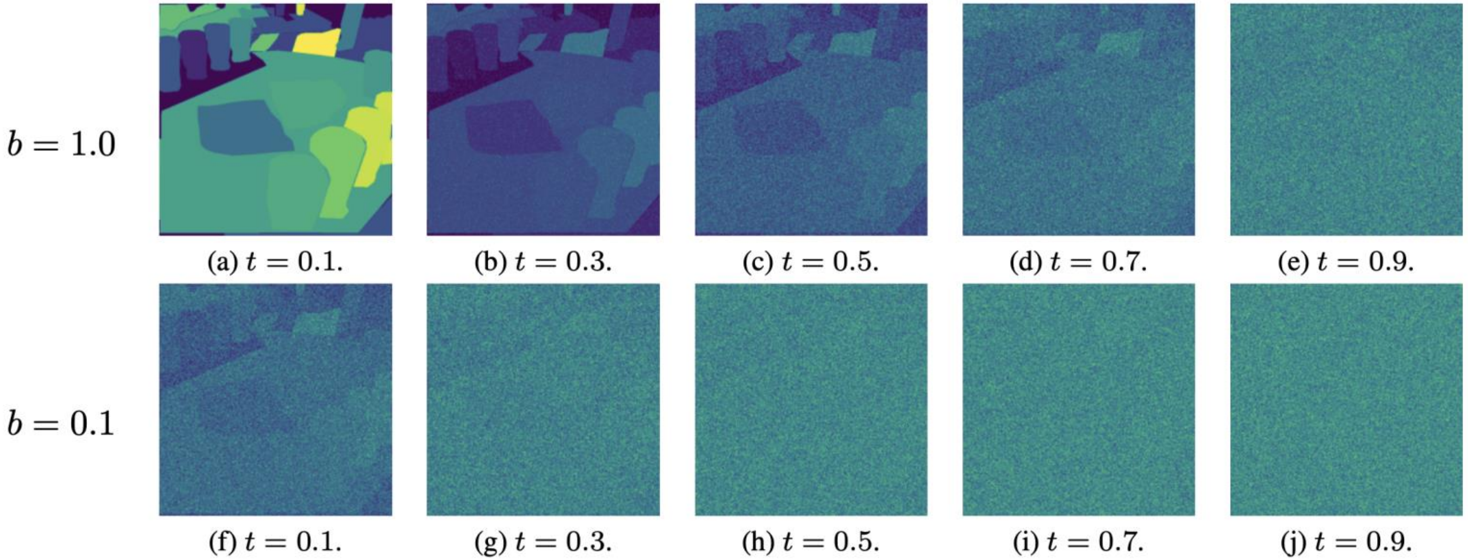


Figure 3. Noisy masks at different time steps under two input scaling factors, $b = 1.0$ (top row) and $b = 0.1$ (bottom row). Decreasing the input scaling factor leads to smaller signal-to-noise ratio (at the same time step), which gives higher weights to harder cases.

Input scaling	0.03	0.1	0.3	1.0
PQ	40.8	43.9	38.7	21.3

Table 3. Ablation on input scaling

Segmentation with diffusion

Another perspective of SNR in diffsuion model

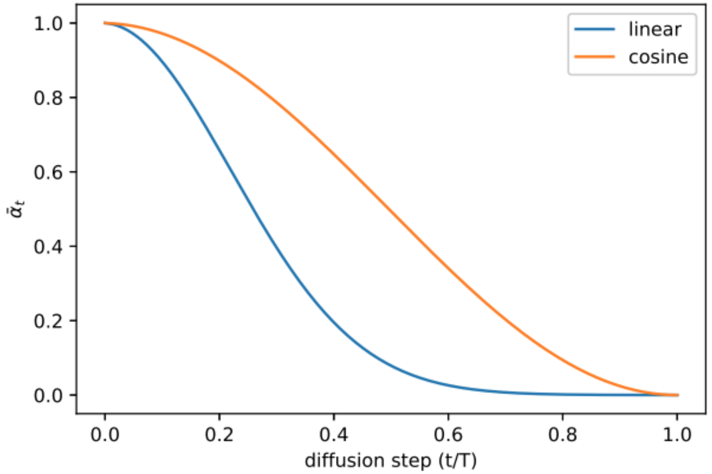


Figure 5. $\bar{\alpha}_t$ throughout diffusion in the linear schedule and our proposed cosine schedule.

Table 1. Ablating schedule and objective on ImageNet 64×64.

Iters	T	Schedule	Objective	NLL	FID
200K	1K	linear	L_{simple}	3.99	32.5
200K	4K	linear	L_{simple}	3.77	31.3
200K	4K	linear	L_{hybrid}	3.66	32.2
200K	4K	cosine	L_{simple}	3.68	**27.0**
200K	4K	cosine	L_{hybrid}	3.62	28.0
200K	4K	cosine	L_{vlb}	**3.57**	56.7
1.5M	4K	cosine	L_{hybrid}	3.57	**19.2**
1.5M	4K	cosine	L_{vlb}	**3.53**	40.1

Table 2. Ablating schedule and objective on CIFAR-10.

Iters	T	Schedule	Objective	NLL	FID
500K	1K	linear	L_{simple}	3.73	3.29
500K	4K	linear	L_{simple}	3.37	**2.90**
500K	4K	linear	L_{hybrid}	3.26	3.07
500K	4K	cosine	L_{simple}	3.26	3.05
500K	4K	cosine	L_{hybrid}	3.17	3.19
500K	4K	cosine	L_{vlb}	**2.94**	11.47

Segmentation with diffusion

Softmax Cross Entropy Loss

Original DDPM:

$$L_{t-1} = D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))$$

$$= \mathbb{E}_{x0,\epsilon}[C||\epsilon - \epsilon_\theta||^2]$$

Ours:

$$‘01’ = w_0‘00’ + w_1‘01’ + w_2‘10’ + w_3‘11’$$

$$\mathcal{L} = \sum_{i,j,k} \boldsymbol{y}_{ijk} \log \mathrm{softmax}(\tilde{\boldsymbol{y}}_{ijk})$$

Loss Weighting:

Pixels in small instance are multipied with a bigger weight

$$w_{ij} = 1/c_{ij}^p \text{ , and } w'_{ij} = H * W * w_{ij} / \sum_{ij} w_{ij}$$

Loss function	ℓ_2 Regression	Cross Entropy
PQ	41.9	43.9

Table 4. Ablation on loss function.

Loss weight p	0	0.2	0.4	0.6
PQ	40.4	43.9	43.7	41.3

Table 5. Ablation on loss weighting.

Segmentation with diffusion

Extension to Videos:

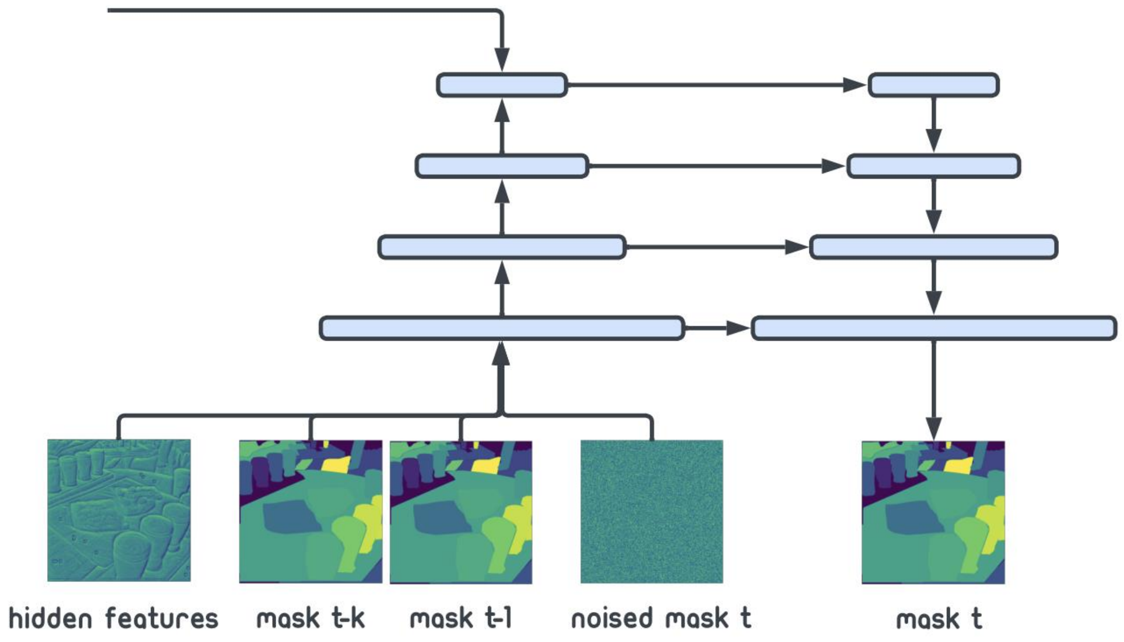


Figure 5. Mask decoder extended for video settings. The image conditional signal to the mask decoder is concatenated with mask predictions from previous frames of the video.

Segmentation with diffusion

Main results:

Method	Backbone	# of Params	PQ	PQ^{thing}	PQ^{stuff}
Specialist approaches:					
MaskFormer [17]	ResNet-50	45M	46.5	51.0	39.8
K-Net [69]	ResNet-50	-	47.1	51.7	40.3
CMT-DeepLab [67]	ResNet-50	-	48.5	-	-
Panoptic SegFormer [35]	ResNet-50	51M	49.6	54.4	42.4
Mask2Former [16]	ResNet-50	44M	51.9	57.7	43.0
kMaX-DeepLab [68]	ResNet-50	57M	53.0	58.3	44.9
DETR [7]	ResNet-101	61.8M	45.1	50.5	37.0
Mask2Former [13]	Swin-L	216M	57.8	-	-
kMaX-DeepLab [68]	ConvNeXt-L	232M	58.1	64.3	48.8
MasK DINO [33]	Swin-L	223M	59.4	-	-
Generalist approaches:					
UViM [31]	ViT	939M	45.8	-	-
Pix2Seq-$\mathcal{D}$ (steps=5)	ResNet-50	94.5M	47.5	52.2	40.3
Pix2Seq-$\mathcal{D}$ (steps=10)	ResNet-50	94.5M	49.4	54.4	41.9
Pix2Seq-$\mathcal{D}$ (steps=20)	ResNet-50	94.5M	50.3	55.3	42.9
Pix2Seq-$\mathcal{D}$ (steps=50)	ResNet-50	94.5M	50.2	55.1	42.8

Table 1. Results on MS-COCO. Pix2Seq-$\mathcal{D}$ achieves competitive results to state-of-the-art specialist models with ResNet-50 backbone.

Method	Backbone	$\mathcal{J}$&$\mathcal{F}$	$\mathcal{J}$-Mean	$\mathcal{J}$-Recall	$\mathcal{F}$-mean	$\mathcal{F}$-Recall
Specialist approaches:						
RVOS [56]	ResNet-101	41.2	36.8	40.2	45.7	46.4
STEm-Seg [3]	ResNet-101	64.7	61.5	70.4	67.8	75.5
MAST [32]	ResNet-18	65.5	63.3	73.2	67.6	77.7
UnOVOST [44]	ResNet-101	67.9	66.4	76.4	69.3	76.9
Propose-Reduce [37]	ResNeXt-101	70.4	67.0	-	73.8	-
Generalist approaches:						
Pix2Seq-$\mathcal{D}$ (ours)	ResNet-50	68.4	65.1	70.6	71.7	77.1

Table 2. Results of unsupervised video object segmentation on DAVIS 2017 validation set.

Segmentation with diffusion

Visualization:

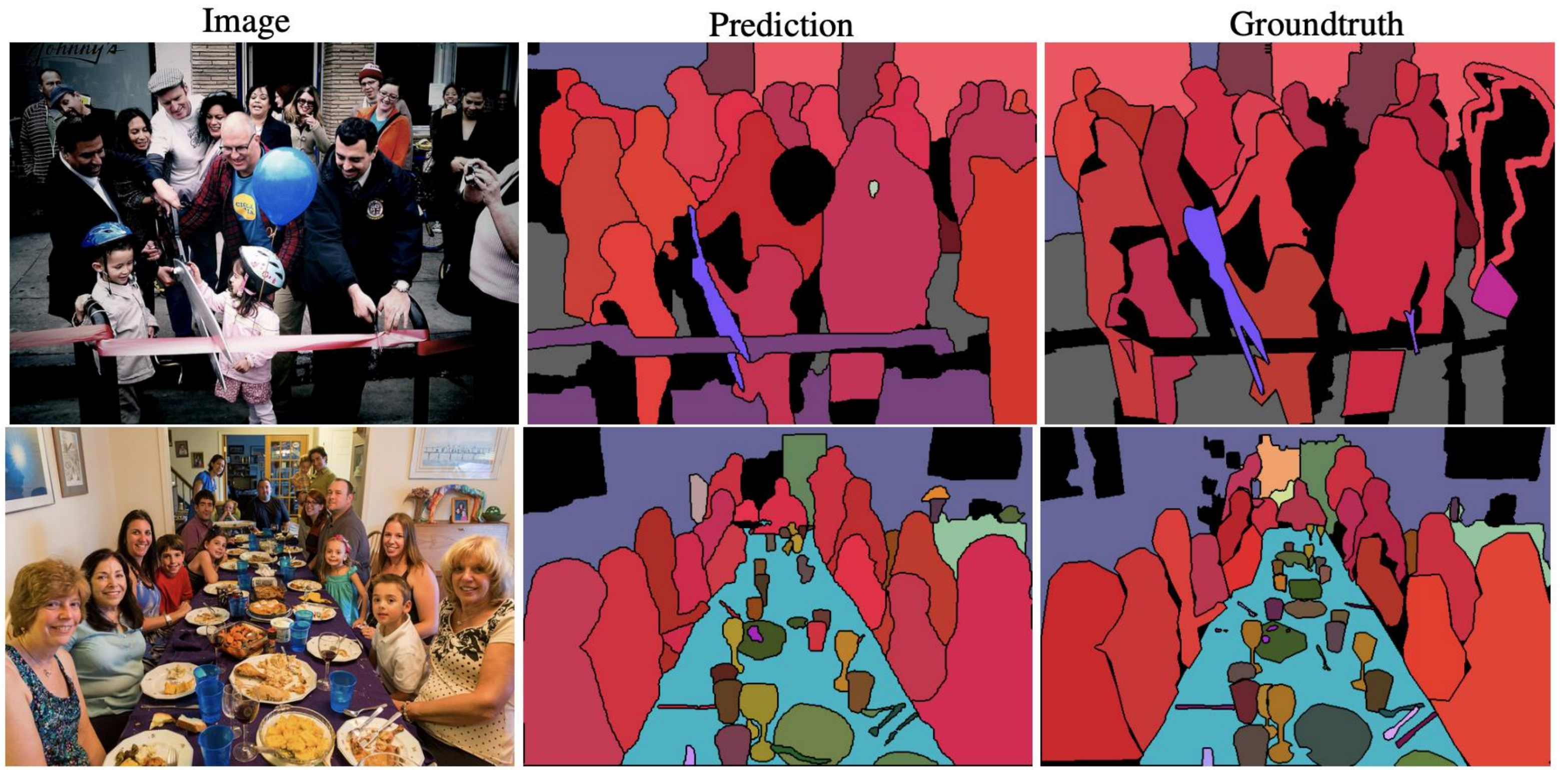


Figure 8. Predictions on MS-COCO *val* set.

Segmentation with diffusion

Visualization:

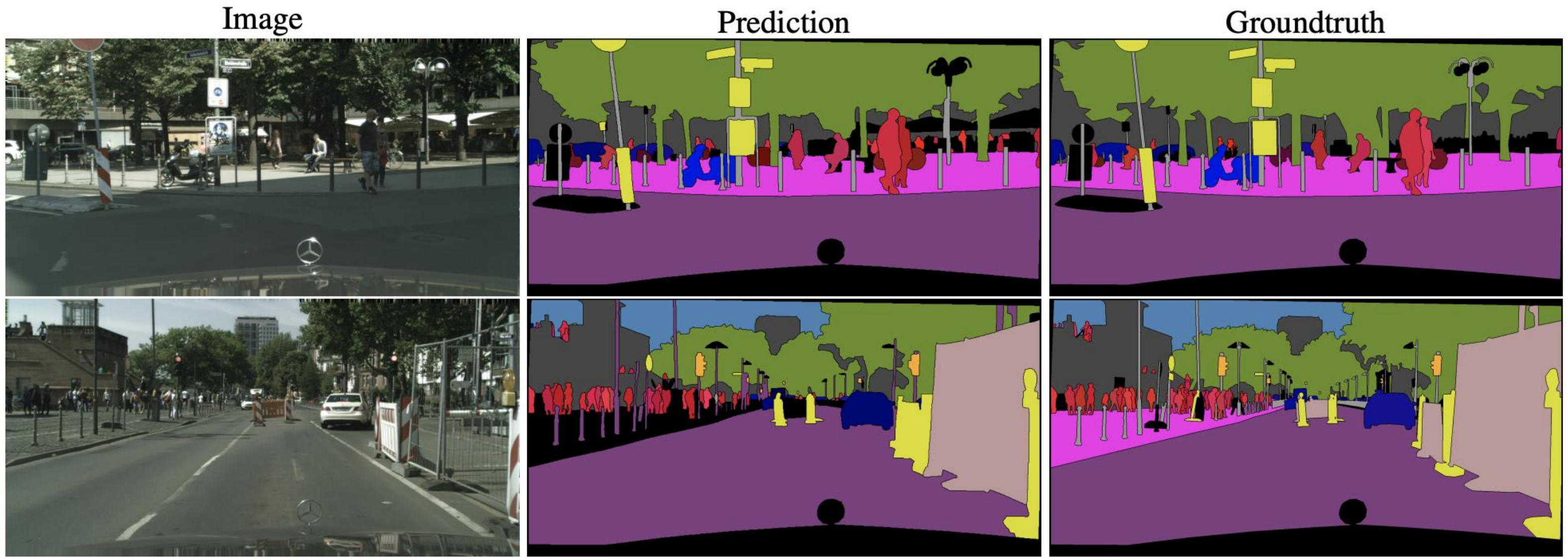


Figure 9. Predictions on Cityscapes *val* set.

Figure 10. Predictions on DAVIS *val* set.