Factorized Diffusion Architectures for Unsupervised Image Generation and Segmentation

Xin Yuan[†‡*]
[†]Google
yuanxzzz@google.com

Michael Maire[‡]
[‡]University of Chicago
mmaire@uchicago.edu

Abstract

We develop a neural network architecture which, trained in an unsupervised manner as a denoising diffusion model, simultaneously learns to both generate and segment images. Learning is driven entirely by the denoising diffusion objective, without any annotation or prior knowledge about regions during training. A computational bottleneck, built into the neural architecture, encourages the denoising network to partition an input into regions, denoise them in parallel, and combine the results. Our trained model generates both synthetic images and, by simple examination of its internal predicted partitions, semantic segmentations of those images. Without fine-tuning, we directly apply our unsupervised model to the downstream task of segmenting real images via noising and subsequently denoising them. Experiments demonstrate that our model achieves accurate unsupervised image segmentation and high-quality synthetic image generation across multiple datasets.

1 Introduction

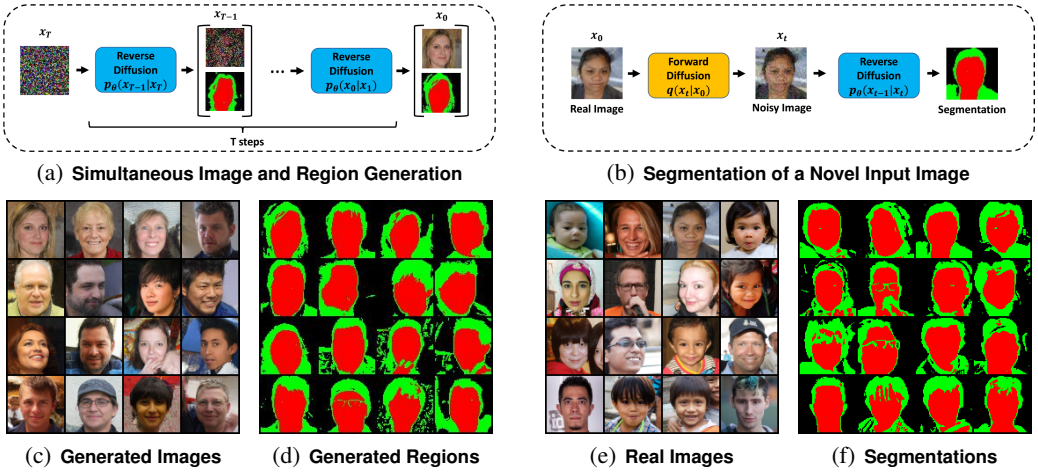
Supervised deep learning yields powerful discriminative representations, and has fundamentally advanced many computer vision tasks, including image classification [13, 58, 21, 28], object detection [18, 49, 41], and semantic and instance segmentation [42, 22, 33]. Yet, annotation efforts [13], especially those involving fine-grained labeling for tasks such as segmentation [39], can become prohibitively expensive to scale with increasing dataset size. This motivates an ongoing revolution in self-supervised methods for visual representation learning, which do not require any annotated data during a large-scale pre-training phase [7, 15, 67, 35, 23, 10, 12]. However, many of these approaches, including those in the particularly successful contrastive learning paradigm [23, 10, 12], still require supervised fine-tuning (*e.g.,* linear probing) on labeled data to adapt networks to downstream tasks such as classification [23, 10] or segmentation [8, 69].

In parallel with the development of self-supervised deep learning, rapid progress on a variety of frameworks for deep generative models [32, 19, 65, 66, 60, 37, 27, 59, 50] has lead to new systems for high-quality image synthesis. This progress inspires efforts to explore representation learning within generative models, with recent results suggesting that image generation can serve as a good proxy task for capturing high-level semantic information, while also enabling realistic image synthesis.

Building upon generative adversarial networks (GANs) [19] or variational autoencoders (VAEs) [32], InfoGAN [11] and Deep InfoMax [26] demonstrate that generative models can perform image classification without any supervision. PerturbGAN [4] focuses on a more complex task, unsupervised image segmentation, by forcing an encoder to map an image to the input of a pre-trained generator so that it synthesizes a composite image that matches the original input image. However, here training is conducted in two stages and mask generation relies on knowledge of predefined object classes.

[*]This work was completed while Xin Yuan was a PhD student at the University of Chicago.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).



(a) **Simultaneous Image and Region Generation** (b) **Segmentation of a Novel Input Image**

(c) **Generated Images** (d) **Generated Regions** (e) **Real Images** (f) **Segmentations**

Figure 1: **Unifying image generation and segmentation.** (a) We design a denoising diffusion model with a specific architecture that couples region prediction with spatially-masked diffusion over predicted regions, thereby generating both simultaneously. (b) An additional byproduct of running our trained denoising model on an arbitrary input image is a segmentation of that image. Using a model trained on FFHQ [31], we achieve both high quality synthesis of images and corresponding semantic segmentations (c-d), as well as the ability to accurately segment images of real faces (e-f). Segmenting a real image is fast, requiring only one forward pass (one denoising step).

Denoising diffusion probabilistic models (DDPMs) [27] also achieve impressive performance in generating realistic images. DatasetDDPM [2] investigates the intermediate activations from the pre-trained U-Net [51] network that approximates the Markov step of the reverse diffusion process in DDPM, and proposes a simple semantic segmentation pipeline fine-tuned on a few labeled images. In spite of this usage of labels, DatasetDDPM demonstrates that high-level semantic information, which is valuable for downstream vision tasks, can be extracted from pre-trained DDPM U-Net. Diff-AE [48] and PADE [70] are recently proposed methods for representation learning by reconstructing images in the DDPM framework. However, their learned representations are in the form of a latent vector containing information applicable for image classification.

In contrast to all of these methods, we demonstrate a fundamentally new paradigm for unsupervised visual representation learning with generative models: constrain the architecture of the model with a structured bottleneck that provides an interpretable view of the generation process, and from which one can simply read off desired latent information. This structured bottleneck does not exist in isolation, but rather is co-designed alongside the network architecture preceding and following it. The computational layout of these pieces must work together in a manner that forces the network, when trained from scratch for generation alone, to populate the bottleneck data structure with an interpretable visual representation.

We demonstrate this concept in the scenario of a DDPM for image generation and semantic segmentation as the interpretable representation to be read from the bottleneck. Thus, we frame unsupervised image segmentation and generation in a unified system. Moreover, experiments demonstrate that domain-specific bottleneck design not only allows us to accomplish an end task (segmentation) for free, but also boosts the quality of generated samples. This challenges the assumption that generic architectures (*e.g.,* Transformers [61]) alone suffice; we find synergy by organizing such generic building blocks into a factorized architecture which generates different image regions in parallel.

Figure 1 provides an overview of our setting alongside example results, while Figure 2 illustrates the details of our DDPM architecture which are fully presented in Section 3. This architecture constrains the computational resources available for denoising in a manner that encourages learning of a factorized model of the data. Specifically, each step of the DDPM has the ability to utilize additional inference passes through multiple copies of a subnetwork if it is willing to decompose the denoising task into parallel subproblems. The specific decomposition strategy itself must be learned, but, by design, is structured in a manner that reveals the solution to our target task of image segmentation. We summarize our contributions as three-fold:

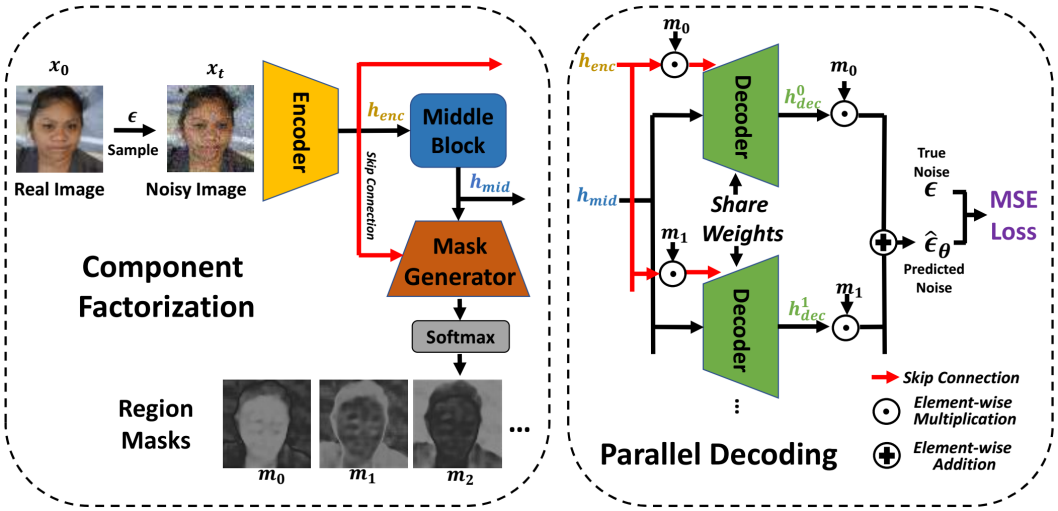


Figure 2: **Factorized diffusion architecture.** Our framework restructures the architecture of the neural network within a DDPM [27] so as to decompose the image denoising task into parallel subtasks. All modules are end-to-end trainable and optimized according to the same denoising objective as DDPM. ***Left: Component factorization.*** An *Encoder*, equivalent to the first half of a standard DDPM U-Net architecture, extracts features $\boldsymbol{h}_{enc}$. A common *Middle Block* processes *Encoder* output into shared latent features $\boldsymbol{h}_{mid}$. Note that *Middle Block* and $\boldsymbol{h}_{mid}$ exist in the standard denoising DDPM U-Net by default. We draw it as a standalone module for a better illustration of the detailed architectural design. A *Mask Generator*, structured as the second half of a standard U-Net receives $\boldsymbol{h}_{mid}$ as input, alongside all encoder features $\boldsymbol{h}_{enc}$ injected via skip connections to layers of corresponding resolution. This later network produces a soft classification of every pixel into one of K region masks, $\boldsymbol{m_0}, \boldsymbol{m_1}, ..., \boldsymbol{m_K}$. ***Right: Parallel decoding.*** A *Decoder*, also structured as the second half of a standard U-Net, runs separately for each region. Each instance of the *Decoder* receives shared features $\boldsymbol{h}_{mid}$ and a masked view of encoder features $\boldsymbol{h}_{enc} \odot \boldsymbol{m_i}$ injected via skip connections to corresponding layers. Decoder outputs are masked prior to combination. Though not pictured, we inject timestep embedding $\boldsymbol{t}$ into the *Encoder*, *Mask Generator*, and *Decoder*.

- **Unified learning of generation and segmentation.** We train our new DDPM architecture once, obtaining a model directly applicable to two different tasks with zero modification or fine-tuning: image generation and image segmentation. Segmenting a novel input image is fast, comparable in speed to any system using a single forward pass of a U-Net [51] like architecture.
- **Unsupervised segmentation for free.** Our method automatically learns meaningful regions (*e.g.,* foreground and background), guided only by the DDPM denoising objective; no extra regularization terms, no use of labels.
- **Higher quality image synthesis.** Our model generates higher-quality images than the baseline DDPM, as well as their corresponding segmentations simultaneously. We achieve excellent quantitative and qualitative results under common evaluation protocols (Section 4).

Beyond improvements to image generation and segmentation, our work is a case study of a new paradigm for using generation as a learning objective, in combination with model architecture as a constraint. Rather than viewing a pre-trained generative model as a source from which to extract and repurpose features for downstream tasks, design the model architecture in the first place so that, as a byproduct of training from scratch to generate, it also learns to perform the desired task.

2 Related Work

Image Segmentation. Generic segmentation, which seeks to partition an image into meaningful regions without prior knowledge about object categories present in the scene, is a longstanding challenge for computer vision. Early methods rely on combinations of hand-crafted features based on intensity, color, and texture cues [6, 44], clustering algorithms [57], and a duality between closed

contours and the regions they bound [1]. Deep learning modernized the feature representations used in these pipelines, yielding systems which, trained with supervision from annotated regions [43], reach near human-level accuracy on predicting and localizing region boundaries [3, 56, 64, 34].

Semantic segmentation, which assigns a category label to each pixel location in image, has been similarly revolutionized by deep learning. Here, the development of specific architectures [42, 51, 20] enabled porting of approaches for image classification to the task of semantic segmentation.

Recent research has refocused on the challenge of learning to segment without reliance on detailed annotation for training. Hwang et al. [29] combine two sequential clustering modules for both pixel-level and segment-level to perform this task. Ji et al. [30] and Ouali et al. [46] follow the concept of mutual information maximization to partition pixels into two segments. Savarese et al. [54] further propose a learning-free adversarial method from the information theoretic perspective, with the goal of minimizing predictability among different pixel subsets. Note that even completely unsupervised foreground/background segmentation is a non-trivial task. Liu et al. [40], a recent advance in this regime, produces similar region mask output, yet depends entirely upon motion cues from video for training. We achieve such unsupervised learning from static images alone.

Learning Segmentation in Generative Models. Previous generative model-based approaches learn semantic segmentation by perturbing [4] or redrawing [9] generated foreground and background masks. Despite good performance, these methods apply only to two-class partitions and require extra loss terms based upon object priors in training datasets.

Denoising diffusion probabilistic models (DDPMs) [27] achieve state-of-the-art performance in generating realistic images. Their noise schedule in training may offer advantages for scaling up models in a stable manner. Recent works [2, 48, 70] explore representation learning capability in DDPMs. DatasetDDPM [2] examines few-shot segmentation with pre-trained diffusion models, but requires human labels to train a linear classifier. With the default U-Net architecture [51], it loses the efficiency and flexibility of generating image and masks in a single-stage manner. Diff-AE [48] and PADE [70] perform representation learning driven by a reconstruction objective in the DDPM framework. Unfortunately, their learned latent vectors are not applicable to more challenging segmentation tasks and they require a pre-trained interpreter to perform downstream image classification.

DiffuMask [63] takes a pre-trained Stable Diffusion model [50], which is built using large-scale text-to-image datasets (and thus solves a far less challenging problem), and conducts a post-hoc investigation on how to extract segmentation from its attention maps. Neither our system, nor the baseline DDPM to which we compare, makes use of such additional information. Furthermore, DiffuMask does not directly output segmentation; it is basically a dataset generator, which produces generated images and pseudo labels, which are subsequently used to train a separate segmentation model. Our method, in contrast, is both completely unsupervised and provides an end-to-end solution by specifying an architectural design in which training to generate reveals segmentations as a bonus.

MAGE [38] shares with us a similar motivation of framing generation and representation learning in a unified framework. However, our approach is distinct in terms of both (1) task: we tackle a more complex unsupervised segmentation task (without fine-tuning) instead of image classification (with downstream fine-tuning), and (2) design: 'masks' play a fundamentally different role in our system. MAGE adopts an MAE [24]-like masking scheme on input data, in order to provide a proxy reconstruction objective for self-supervised representation learning. Our use of region masks serves a different purpose, as they are integral components of the model being learned and facilitate factorization of the image generation process into parallel synthesis of different segments.

BlobGAN [16] is a generative model for creating images with fine-grained control over the spatial arrangement of content. It leverages blob-like components instead of accurate region masks as basic building blocks for the synthesis process, allowing for intuitive content manipulation. In the generative modeling space, BlobGAN serves a different purpose than our method: BlobGAN excels in scenarios requiring explicit spatial control and interactive editing, while our factorized diffusion approach provides a framework for learning high-quality image generation and segmentation.

3 Factorized Diffusion Models

Figure 2 illustrates the overall architecture of our system, which partitions the denoising network within a diffusion model into an unsupervised region mask generator and parallel per-region decoders.

3.1 Unsupervised Region Factorization

To simultaneously learn representations for both image generation and unsupervised segmentation, we first design the region mask generator based on the first half (encoder) of a standard DDPM U-Net. We obtain input $\boldsymbol{x}_t$, a noised version of $\boldsymbol{x}_0$, via forward diffusion:

$$\begin{aligned} q(\boldsymbol{x}_t|\boldsymbol{x}_0) &:= \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)I), \\ \boldsymbol{x}_t &= \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(0,1), \end{aligned} \tag{1}$$

where $\alpha_t = 1 - \beta_t, \bar{\alpha}_t = \prod_{s=1}^{t} \alpha_t$.

In addition to the encoder half of the U-Net, we instantiate a middle block consisting of layers operating on lower spatial resolution features. Parameterizing these subnetworks as θ_{enc} and θ_{mid}, we extract latent representations:

$$\boldsymbol{h}_{enc} = \theta_{enc}(\boldsymbol{x}_t, t), \tag{2}$$

$$\boldsymbol{h}_{mid} = \theta_{mid}(\boldsymbol{h}_{enc}, t) \tag{3}$$

where $\boldsymbol{h}_{enc}$ encapsulates features at all internal layers of θ_{enc}, for subsequent use as inputs, via skip connections, to corresponding layers of decoder-style networks (second half of a standard U-Net).

We instantiate a mask generator, θ_{mask}, as one such decoder-style subnetwork. A softmax layer produces an output tensor with K channels, representing K different regions in image $\boldsymbol{x}_0$:

$$\boldsymbol{m}_k = \theta_{mask}(\boldsymbol{h}_{mid}, \boldsymbol{h}_{enc}, t) \tag{4}$$

Following a U-Net architecture, $\boldsymbol{h}_{enc}$ feeds into θ_{mask} through skip-connections.

3.2 Parallel Decoding Through Weight Sharing

We aim to extend a standard DDPM U-Net decoder θ_{dec} to consider region structure during generation. One simple design is to condition on $\boldsymbol{m} = \{\boldsymbol{m_0}, \boldsymbol{m_1}, ...\}$ by concatenating it with input $\boldsymbol{h}_{mid}$ and $\boldsymbol{h}_{enc}$ along the channel dimension:

$$\hat{\boldsymbol{\epsilon}} = \theta_{dec}(\text{concat}[\boldsymbol{h}_{mid}, \boldsymbol{m}], \text{concat}[\boldsymbol{h}_{enc}, \boldsymbol{m}], t), \tag{5}$$

where $\boldsymbol{h}_{mid}$ and $\boldsymbol{h}_{enc}$ are generated from Eq. 2 and Eq. 3. We downsample $\boldsymbol{m}$ accordingly to the same resolution as $\boldsymbol{h}_{mid}$ and $\boldsymbol{h}_{enc}$ at different stages. However, such a design significantly modifies (*e.g.,* channel sizes) the original U-Net decoder architecture. Moreover, conditioning with the whole mask representation may also result in a trivial solution that simply ignores region masks.

To address these issues, we separate the decoding scheme into multiple parallel branches of weight-shared U-Net decoders, each masked by a single segment. Noise prediction for k-th branch is:

$$\hat{\boldsymbol{\epsilon}}_k = \theta_{dec}(\boldsymbol{h}_{mid}, \boldsymbol{h}_{enc} \odot \boldsymbol{m}_k, t) \tag{6}$$

and the output is a sum of region-masked predictions:

$$\hat{\boldsymbol{\epsilon}} = \sum_{k=0}^{K-1} \hat{\boldsymbol{\epsilon}}_k \odot \boldsymbol{m}_k \tag{7}$$

3.3 Optimization with Denoising Objective

We train our model in an end-to-end manner, driven by the simple DDPM denoising objective. Model weights $\theta = \{\theta_{enc}, \theta_{mid}, \theta_{dec}, \theta_{mask}\}$ are optimized by minimizing the noise prediction loss:

$$L = \mathbb{E}||\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}||_2^2 \tag{8}$$

Unlike previous work, our method does not require a mask regularization loss term [54, 4, 9], which predefines mask priors (*e.g.,* object size). Algorithm 1 summarizes training.

3.4 Segmentation via Reverse Diffusion

Once trained, our model can both segment novel input images and synthesize images from noise.

Real Image Segmentation. Given clean input image $\boldsymbol{x}_0$, we first sample a noisy version $\boldsymbol{x}_t$ through forward diffusion in Eq. 1. We then perform one-step denoising by passing $\boldsymbol{x}_t$ to the model. We collect the predicted region masks as the segmentation for $\boldsymbol{x}_0$ using Eq. 4.

Image and Mask Generation. Using reverse diffusion, our model can generate realistic images and their corresponding segmentation masks, starting from a pure noise input $\boldsymbol{x}_T \sim \mathcal{N}(0, 1)$. Reverse diffusion predicts $\boldsymbol{x}_{t-1}$ from $\boldsymbol{x}_t$:

$$\boldsymbol{x}_{t-1} = 1/\sqrt{\alpha_t}(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\theta(\boldsymbol{x}_t, t)) + \sigma_t \boldsymbol{z}, \tag{9}$$

$$\boldsymbol{z} \sim \mathcal{N}(0, 1) \quad \text{if} \quad t > 1 \quad \text{else} \quad \boldsymbol{z} = 0. \tag{10}$$

where σ_t is empirically set according to the DDPM noise scheduler. We perform T steps of reverse diffusion to generate an image. We also collect its corresponding masks using Eq. 4 when $t = 1$. Algorithm 2 summarizes this process.

Algorithm 1
Training Masked Diffusion

Input: Data $\boldsymbol{x}_0$
Output: Trained model θ
Initialize:
  Model weights θ, Timesteps T
for iter $= 1$ **to** Iter$_{total}$ **do**
  Sample $t \in [1, T]$
  Sample $\boldsymbol{x}_t$ using Eq. 1
  Calculate $\hat{\boldsymbol{\epsilon}}$ using Eq. 7
  Backprop with Eq. 8
  Update θ
end for
return θ

Algorithm 2
Image and Mask Generation

Input: Noise $\boldsymbol{x}_T$, trained model θ
Output:
  Image $\hat{\boldsymbol{x}}_0$ and segmentation $\hat{\boldsymbol{m}}_0$
Initialize: $\boldsymbol{x}_T \sim \mathcal{N}(0, 1)$
for t $= T$ **to** 1 **do**
  Sample $\boldsymbol{z}$ using Eq. 10
  Perform reverse diffusion using Eq. 9
  if $t = 1$ **then**
    Collect $\hat{\boldsymbol{m}}_0$ using Eq. 4
    Return $\hat{\boldsymbol{x}}_0$ and $\hat{\boldsymbol{m}}_0$
  end if
end for

4 Experiments

We evaluate on: (1) real image segmentation, (2) image and region mask generation, using Flower [45], CUB [62], FFHQ [31], CelebAMask-HQ [36], and ImageNet [53]. In addition to the design of flat set of K regions, we also conduct a preliminary investigation into reorganizing our architectural design to support hierarchical segmentations; see Section A.1.

Evaluation Metrics. For unsupervised segmentation on Flower and CUB, we follow the data splitting in IEM [54] and evaluate predicted mask quality under three commonly used metrics, denoted as Acc., IOU and DICE score [54, 9]. Acc. is the (per-pixel) mean accuracy of the foreground prediction. IOU is the predicted foreground region's intersection over union with the ground-truth foreground region. DICE score is defined as $2\frac{\hat{F} \cap F}{|\hat{F}|}$ [14]. On ImageNet, we evaluate our method on Pixel-ImageNet [68], which provides human-labeled segmentation masks for 0.485M images covering 946 object classes. We report Acc., IOU and DICE score on a randomly sampled subset, each class containing at most 20 images. For face datasets, we train our model on FFHQ and only report per-pixel accuracy on the CelebAMask test set, using provided ground-truth.

For image and mask generation, we use Fréchet Inception Distance (FID) [25] for generation quality assessment. Since we can not obtain the ground-truth for generated masks, we apply a supervised U-Net segmentation model, pre-trained on respective datasets, to the generated images and measure the consistency between masks in terms of per-pixel accuracy. In addition to quantitative comparisons, we show extensive qualitative results.

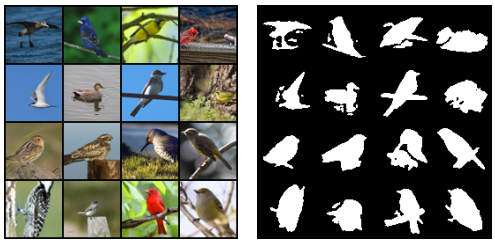
Implementation Details. We train Flower, CUB and Face models at both 64×64 and 128×128 resolution. We also train class-conditioned ImageNet models with 64×64 resolution. For all

(a) **Real Images** (b) **Segmentation**

Figure 3: Segmentation on Flower.

Methods	Acc.	IOU	DICE
GrabCut [52]	82.0	69.2	79.1
ReDO [9]	87.9	76.4	-
IEM [54]	88.3	76.8	84.6
IEM+SegNet [54]	89.6	78.9	86.0
Ours	**90.1**	**79.7**	**87.2**

Table 1: Comparisons on Flower.



(a) **Real Images** (b) **Segmentation**

Figure 4: Segmentation on CUB.

Methods	Acc.	IOU	DICE
GrabCut [52]	72.3	36.0	48.7
PerturbGAN [4]	-	38.0	-
ReDO [9]	84.5	42.6	-
IEM [54]	88.6	52.2	66.0
IEM+SegNet [54]	89.3	55.1	68.7
Ours	**89.6**	**56.1**	**69.4**

Table 2: Comparisons on CUB.

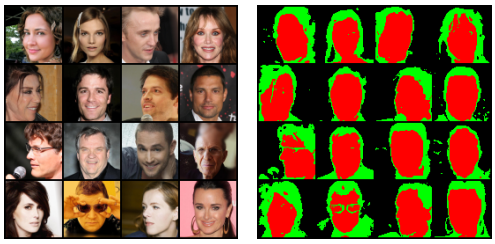
experiments, we use the U-Net [51] encoder-middle-decoder architecture similar to [27]. We use the decoder architecture as our mask generator and set the number of factorized masks K as 3. We note that K is the maximum number of regions the model may use. It could learn fewer components during training. For binary segmentation, we found setting $K = 3$ rather than $K = 2$ to assist training, with learned regions emerging as foreground, background, and a contour or transition between the two. For segmentation evaluation, we simply select the mask channel that emerges as foreground and apply standard benchmarks. For 64×64 the architecture is as follows: The downsampling stack performs four steps of downsampling, each with 3 residual blocks. The upsampling stack is setup as a mirror image of the downsampling stack. From highest to lowest resolution, U-Net stages use $[C, 2C, 3C, 4C]$ channels, respectively. For 128×128 architecture, the down/up sampling block is 5-step with $[C, C, 2C, 3C, 4C]$ channels, each with two residual blocks, respectively. We set $C = 128$ for all models.

We use Adam to train all the models with a learning rate of 10^{-4} and an exponential moving average (EMA) over model parameters with rate 0.9999. For all datasets except ImageNet, we train 64×64 and 128×128 models on 8 and 32 Nvidia V100 32GB GPUs, respectively. For Flower, CUB and FFHQ, we train the models for 50K, 50K, 500K iterations with batch size of 128, respectively. For ImageNet, we train 500K iterations on 32 Nvidia V100 GPUs with batch size 512. We adopt the linear noise scheduler as in Ho *et al.* [27] with $T = 1000$ timesteps.

4.1 Image Segmentation

To evaluate our method on real image segmentation, we set t as 30 for the forward diffusion process. We also investigate the segmentation results with different noise levels in Figure 18. For Flower and CUB, Figures 3 and 4 show test images and predicted segmentations. Tables 1 and 2 provide quantitative comparison with representative unsupervised image segmentation methods: GrabCut [52], ReDO [9] and IEM [54]. As shown in Table 1 and Table 2, our method outperforms all competitors.

We also visualize the predicted face parsing results on FFHQ and CelebAMask datasets in Figure 1(c)(d) and Figure 5. Our model learns to accurately predict three segments corresponding to semantic components: skin, hair, and background. This particular semantic partitioning emerges from our unsupervised learning objective, without any additional prior. With ground-truth provided on CelebAMask-HQ, we also compare the pixel accuracy and mean IOU with a supervised U-Net and DatasetDDPM [2]. For the former, we train a supervised segmentation model with 3-class cross-entropy loss. For the unsupervised setting, we perform K-means (K=3) on the pre-trained DDPM, denoted as DatasetDDPM-unsup. Table 3 shows that we outperform DatasetDDPM by a large margin and achieve a relatively small performance gap with a supervised U-Net.

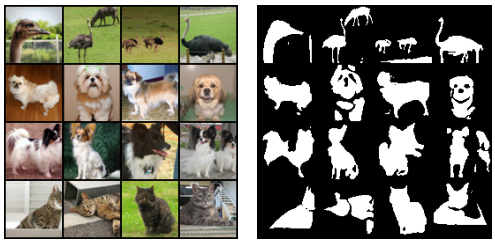


(a) **Real Images** (b) **Segmentation**

Figure 5: Segmentation on CelebA.

Methods	Acc.	mIOU
Supervised UNet	95.7	90.2
DatasetDDPM-unsup. [2]	78.5	69.3
Ours	87.9	80.3

Table 3: Seg. comparisons on CelebA.



(a) **Real Images** (b) **Segmentation**

Figure 6: Segmentation on ImageNet.

Methods	Acc.	mIOU
Supervised UNet	85.7	74.1
DatasetDDPM-unsup. [2]	74.1	60.4
Ours	80.7	67.7

Table 4: Seg. comparisons on ImageNet.

Figure 6 shows the accurate segmentation results for ImageNet classes: ostrich, pekinese, papillon, and tabby. We compare with supervised U-Net and DatasetDDPM-unsup in Table 4. We show more visualizations in the Appendix.

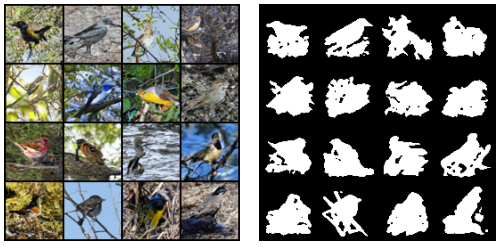
4.2 Image and Mask Generation

We evaluate our method on image and mask generation. As shown in Figure 7, 8, 1(c)(d) and 9, our method is able to generate realistic images. In the upper row of Table 5, we see a consistent quality improvement over the original DDPM. This suggests that our method is a better generation architecture than the standard U-Net; separate computational paths for denoising individual image regions is a beneficial prior to impose when learning to model the image distribution. Additionally, our method produces accurate corresponding masks, closely aligned with the semantic partitions in the generated image.

We also evaluate the quality of these segmentations. Since there is no ground-truth mask provided for generated images, we apply the U-Net segmentation models (pre-trained on respective labeled training sets) to the generated images to produce reference masks. We measure the consistency between the reference and the predicted parsing results in terms of pixel-wise accuracy. We compare our method with a pre-trained DDPM baseline, in which we first perform image generation, then pass them to DatasetDDPM-unsup to get masks. As shown in Table 5 (bottom), our method consistently achieves better segmentation on generated images than the DDPM baseline. Note that, different from the two-stage baseline, our method performs the computation in a single stage, generating image and mask simultaneously. The Appendix shows more visualizations.

(a) **Generated Images** (b) **Generated Masks**

Figure 7: Generation on Flower.

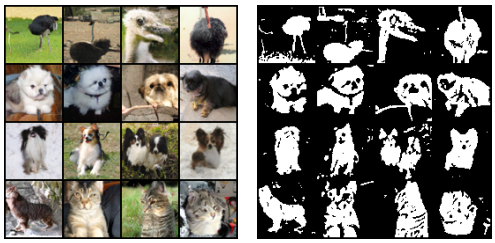


(a) **Generated Images** (b) **Generated Masks**

Figure 8: Generation on CUB.

Table 5: Image and mask generation comparison on all datasets (top: FID(↓) bottom: Acc. (↑)).

Models	Flower-64	Flower-128	CUB-64	CUB-128	FFHQ-64	FFHQ-128	ImageNet-64
DDPM	15.81	14.62	14.45	14.01	13.72	13.35	7.02
Ours	**13.33**	**11.50**	**10.91**	**10.28**	**12.02**	**10.79**	**6.54**
DDPM	80.5	82.9	84.2	83.7	84.2	84.2	71.2
Ours	**92.3**	**92.7**	**91.4**	**91.2**	**90.3**	**90.7**	**84.1**



(a) **Generated Images** (b) **Generated Masks**

Figure 9: Conditional generation on ImageNet.

Methods	IOU.(↑)	FID (↓)
Concat	20.7	14.21
Masking $\boldsymbol{h}_{mid}$	20.2	14.33
w/o weight sharing	50.5	17.21
Ours	**56.1**	**10.28**

Table 6: Ablations of decoding scheme on CUB.

4.3 Ablation Study and Analysis

Multi-branch Decoders with Weight Sharing. Separating computation in multi-branch decoders with weight sharing is an essential design in our method. We show the effectiveness of this design by varying how to apply factorized masks in our decoding scheme: (1) concat: we use single branch to take concatenation of $\boldsymbol{h}$ and $\boldsymbol{m}$. (2) masking $\boldsymbol{h}_{mid}$: we use $\boldsymbol{m}$ to mask $\boldsymbol{h}_{mid}$ instead of $\boldsymbol{h}_{enc}$. (3) w/o weight sharing: we train decoders separately in our design. Table 6 shows separate design consistently yields better visual features than other designs for CUB. This suggests that our design benefits from fully utilizing mask information in the end-to-end denoising task and avoids a trivial solution where masks are simply ignored.

Investigation on Mask Factorization. Our architecture is able to generate factorized representations, each representing a particular segment of the input image. We show this by visualizing the individual channels from softmax layer output in our mask generator. As shown in Figure 10, skin, hair, and background are separated in different channels.

Mask Refinement along Diffusion Process. In the DDPM Markov process, the model implicitly formulates a mapping between noise and data distributions. We validate that this occurs for both images and latent region masks by visualizing image and mask generation along the sequential reverse diffusion process in Figure 11. We observe gradual refinement as denoising steps approach $t = 0$.

Zero-shot Object Segmentation. We evaluate zero-shot object segmentation on both PASCAL VOC 2012 [17] and DAVIS-2017 videos [47]. Baseline DDPM generation is not solved for these datasets when training from scratch without external large-scale datasets (*e.g.,* LAION [55] used in Stable Diffusion [50]). We directly adopt zero-shot transfer from our pre-trained ImageNet model by applying the conditional label mapping to VOC. Table 7 details the mapping rule. Figure 13 shows the accurate segmentation results for images of classes: aeroplane, monitor, person, and sofa from VOC. Since our method does not require any pixel labels, we evaluate the performance of

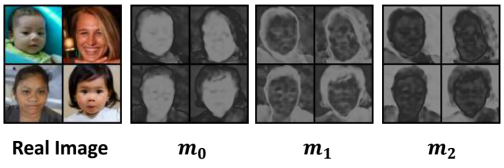


Figure 10: Mask factorization (3 parts) on FFHQ.

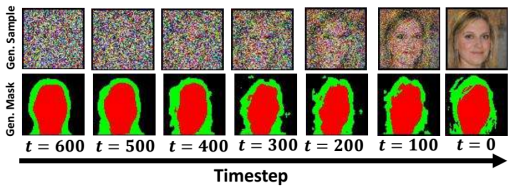


Figure 11: Gen. refinement along diffusion.

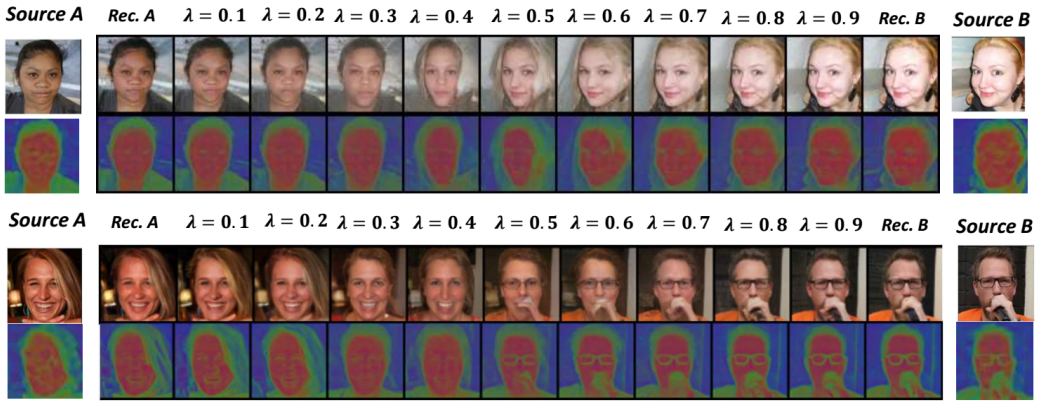
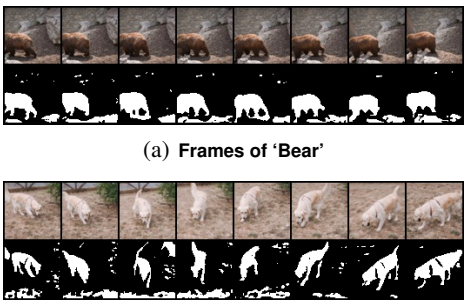


Figure 12: Interpolations on FFHQ with 250 timesteps of diffusion.

(a) **Real Images** (b) **Segmentation**

Figure 13: Segmentation on VOC-2012.



(a) **Frames of 'Bear'**

(b) **Frames of 'Dog'**

Figure 14: Segmentation on DAVIS-17.

each object class individually. We report pixel accuracy and mIOU of each class in VOC in Table 7, which demonstrates that our method can achieve reasonably high performance. Our method achieves an accuracy of **0.78** and mIOU of **0.54** when averaging over all 20 classes. We also show video segmentation on DAVIS-2017 in Figure 14 and the Appendix, without any labeled video pre-training.

Face Interpolation. We also investigate face interpolation on FFHQ. Similar to standard DDPM [27], we perform the interpolation in the denoising latent space with 250 timesteps of diffusion. Figure 12 shows good reconstruction in both pixels and region masks, yielding smoothly varying interpolations across face attributes such as pose, skin, hair, expression, and background.

5 Conclusion

We propose a factorized architecture for diffusion models that is able to perform unsupervised image segmentation and generation simultaneously, while being trained once, from scratch, for image generation via denoising alone. Using model architecture as a constraint, via carefully designed component factorization and parallel decoding schemes, our method effectively and efficiently bridges these two challenging tasks in a unified framework, without the need for fine-tuning or altering the original DDPM training objective. Our work is the first example of engineering an architectural bottleneck so that learning a desired end task becomes a necessary byproduct of training to generate.

Our work is at the stage of a new architectural design for diffusion-based segmentation and generation, with 2- or 3-class segmentation results demonstrating improvements across multiple datasets, scaling up to ImageNet. Our initial investigation into hierarchical extensions suggests a promising future path towards handling complex scenes.

References

[1] Pablo Arbeláez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *PAMI*, 2011.

[2] Dmitry Baranchuk, Andrey Voynov, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Label-efficient semantic segmentation with diffusion models. In *ICLR*, 2022.

[3] Gedas Bertasius, Jianbo Shi, and Lorenzo Torresani. DeepEdge: A multi-scale bifurcated deep network for top-down contour detection. In *CVPR*, 2015.

[4] Adam Bielski and Paolo Favaro. Emergence of object segmentation in perturbed generative models. In *NeurIPS*, 2019.

[5] Chris Burgess and Hyunjik Kim. 3D shapes dataset. https://github.com/google-deepmind/3d-shapes, 2018.

[6] John Canny. A computational approach to edge detection. *PAMI*, 1986.

[7] Mathilde Caron, Piotr Bojanowski, Julien Mairal, and Armand Joulin. Unsupervised pre-training of image features on non-curated data. In *ICCV*, 2019.

[8] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

[9] Mickaël Chen, Thierry Artières, and Ludovic Denoyer. Unsupervised object segmentation by redrawing. In *NeurIPS*, 2019.

[10] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

[11] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *NeurIPS*, 2016.

[12] Xinlei Chen, Haoqi Fan, Ross B. Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv:2003.04297*, 2020.

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

[14] Lee Raymond Dice. Measures of the amount of ecologic association between species. *Ecology*, 1945.

[15] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015.

[16] Dave Epstein, Taesung Park, Richard Zhang, Eli Shechtman, and Alexei A. Efros. BlobGAN: Spatially disentangled scene representations. In *ECCV*, 2022.

[17] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012). http://host.robots.ox.ac.uk/pascal/VOC/voc2012.

[18] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[19] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

[20] Bharath Hariharan, Pablo Andrés Arbeláez, Ross B. Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015.

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

[22] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *ICCV*, 2017.

[23] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[24] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

[25] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *NeurIPS*, 2017.

[26] R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Philip Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *ICLR*, 2019.

[27] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

[28] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[29] Jyh-Jing Hwang, Stella X. Yu, Jianbo Shi, Maxwell D. Collins, Tien-Ju Yang, Xiao Zhang, and Liang-Chieh Chen. SegSort: Segmentation by discriminative sorting of segments. In *ICCV*, 2019.

[30] Xu Ji, Andrea Vedaldi, and João F. Henriques. Invariant information clustering for unsupervised image classification and segmentation. In *ICCV*, 2019.

[31] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

[32] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[33] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollar, and Ross Girshick. Segment anything. In *ICCV*, 2023.

[34] Iasonas Kokkinos. Pushing the boundaries of boundary detection using deep learning. In *ICLR*, 2016.

[35] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Colorization as a proxy task for visual understanding. In *CVPR*, 2017.

[36] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation. In *CVPR*, 2020.

[37] Ke Li and Jitendra Malik. Implicit maximum likelihood estimation. *arXiv:1809.09087*, 2018.

[38] Tianhong Li, Huiwen Chang, Shlok Mishra, Han Zhang, Dina Katabi, and Dilip Krishnan. MAGE: Masked generative encoder to unify representation learning and image synthesis. In *CVPR*, 2023.

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

[40] Runtao Liu, Zhirong Wu, Stella X. Yu, and Stephen Lin. The emergence of objectness: Learning zero-shot segmentation from videos. In *NeurIPS*, 2021.

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

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

[43] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, 2001.

[44] David Martin, Charless Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color and texture cues. *PAMI*, 2004.

[45] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *ICVGIP*, 2008.

[46] Yassine Ouali, Céline Hudelot, and Myriam Tami. Autoregressive unsupervised image segmentation. In *ECCV*, 2020.

[47] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 DAVIS challenge on video object segmentation. *arXiv:1704.00675*, 2017.

[48] Konpat Preechakul, Nattanat Chatthee, Suttisak Wizadwongsa, and Supasorn Suwajanakorn. Diffusion autoencoders: Toward a meaningful and decodable representation. In *CVPR*, 2022.

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

[50] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

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

[52] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "GrabCut": Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 2004.

[53] 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.

[54] Pedro Savarese, Sunnie S. Y. Kim, Michael Maire, Greg Shakhnarovich, and David McAllester. Information-theoretic segmentation by inpainting error maximization. In *CVPR*, 2021.

[55] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5B: An open large-scale dataset for training next generation image-text models. In *NeurIPS*, 2022.

[56] Wei Shen, Xinggang Wang, Yan Wang, Xiang Bai, and Zhijiang Zhang. DeepContour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *CVPR*, 2015.

[57] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *PAMI*, 2000.

[58] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[59] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021.

[60] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with PixelCNN decoders. In *NeurIPS*, 2016.

[61] 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.

[62] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[63] Weijia Wu, Yuzhong Zhao, Mike Zheng Shou, Hong Zhou, and Chunhua Shen. DiffuMask: Synthesizing images with pixel-level annotations for semantic segmentation using diffusion models. In *ICCV*, 2023.

[64] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *ICCV*, 2015.

[65] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. AttnGAN: Fine-grained text to image generation with attentional generative adversarial networks. In *CVPR*, 2018.

[66] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, 2017.

[67] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. In *ECCV*, 2016.

[68] Shiyin Zhang, Jun Hao Liew, Yunchao Wei, Shikui Wei, and Yao Zhao. Interactive object segmentation with inside-outside guidance. In *CVPR*, 2020.

[69] Xiao Zhang and Michael Maire. Self-supervised visual representation learning from hierarchical grouping. In *NeurIPS*, 2020.

[70] Zijian Zhang, Zhou Zhao, and Zhijie Lin. Unsupervised representation learning from pre-trained diffusion probabilistic models. In *NeurIPS*, 2022.

A Appendix

A.1 Hierarchical Factorized Diffusion

We conduct a further investigation to reorganize our architectural design to support hierarchical mask factorization in place of a flat set of K regions. We formulate a hierarchical factorized diffusion architecture to progressively refine segmentation results from a coarse initial prediction to a fine, detailed final segmentation. This approach helps in capturing both global context and fine details in the segmentation task. As shown in Figure 15, the first level replicates the factorized diffusion architecture depicted in Figure 2 to generate initial region masks $m_0^0, m_1^0, ...$, each applied on the noisy input for the next level factorized diffusion process. Each branch of the second level architecture generates finer representations of region masks $m_0^1, m_1^1, ...$, constructing the final denoising output as $\sum_i m_i^0 \frac{(h_i^0 + \sum_j h_j^1 m_j^1)}{2}$. This nested architectural design can be instantiated as repeated levels of factorized diffusion, which is a promising way to handle multiscale scenes. As a proof of concept, we experiment on the shape 3D dataset [5] with a 2-level hierarchy. We first visualize each level's region mask in Figure 16. We observe that the first level generates a coarse segmentation, based on which, second-level factorized diffusion generates fine segmentations of 3D shapes. Figure 17 provides a more direct visualization of partitions at each level through a 3-class mapping.

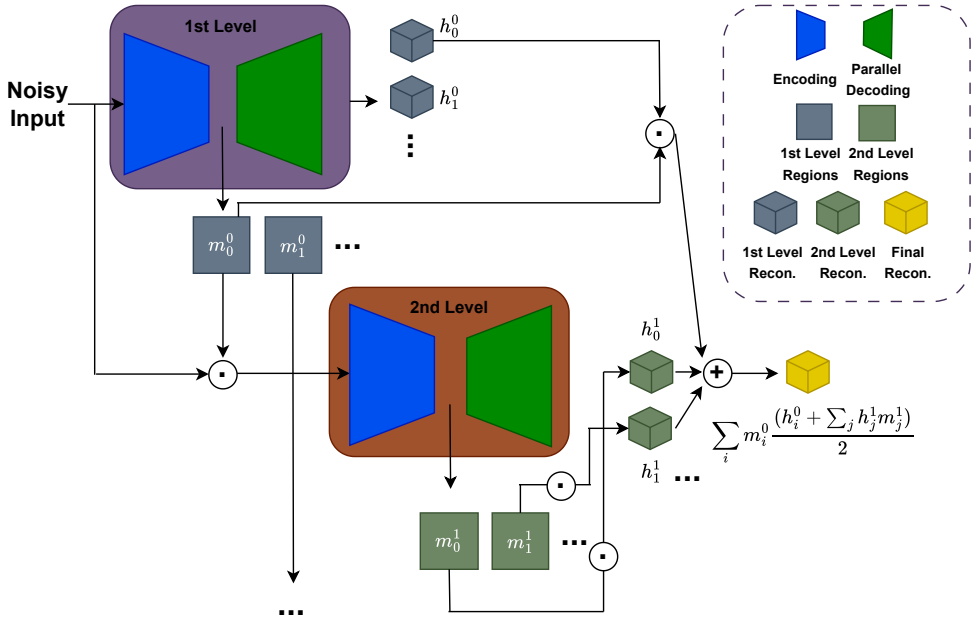


Figure 15: **Hierarchical factorized diffusion architecture.**

A.2 Additional Segmentation Results

We show more segmentation results for Flower, CUB, FFHQ, CelebA and ImageNet. As shown in Figures 19, 20, 21, 22, and 23, our method consistently predicts accurate segmentations for real image inputs.

A.3 Additional Generation Results

We show more generation results for Flower, CUB, FFHQ, and ImageNet (classes: flamingo, water buffalo, garbage truck, and sports car). As shown in Figures 24, 25, 26, and 27, our method consistently produces images and masks with high quality.

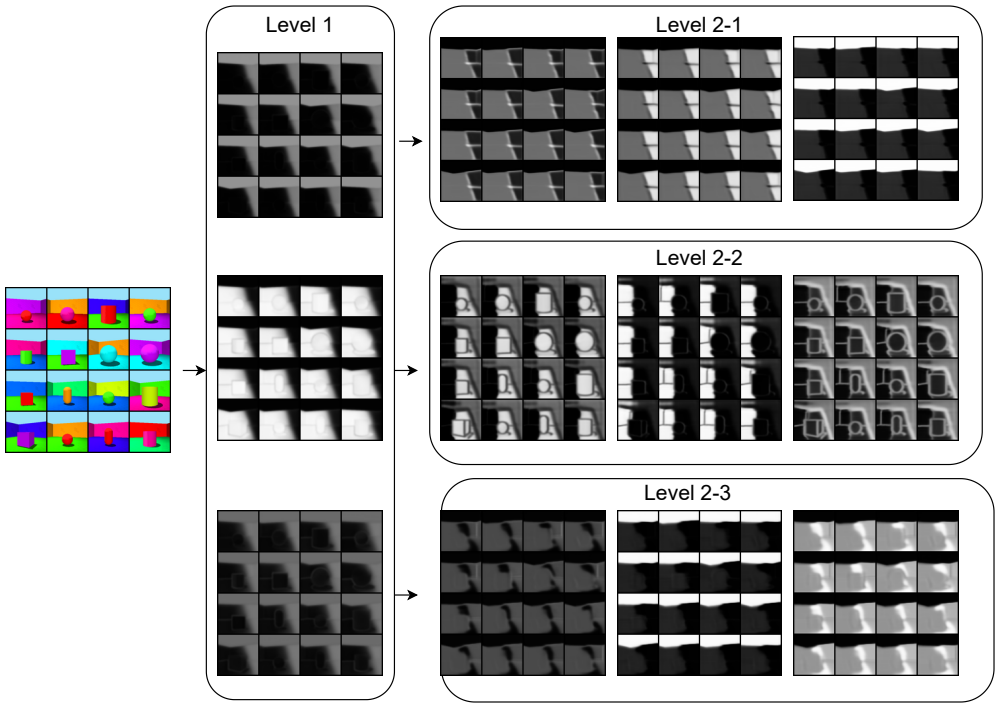


Figure 16: **Mask factorization for each level.** ***Level 1:*** visualization of each mask channel at the first level. ***Level 2-1, 2-2, 2-3:*** visualization of each mask channel per branch at the second level.

A.4 Additional Zero-shot Results on VOC

We provide more segmentation results of 'bicycle', 'chair', 'potted plant' and 'train' in Figure 28.

A.5 Additional Zero-shot Results on DAVIS

We provide more DAVIS-2017 video segmentation results of 'classic-car', 'dance-jump' in Figure 29.

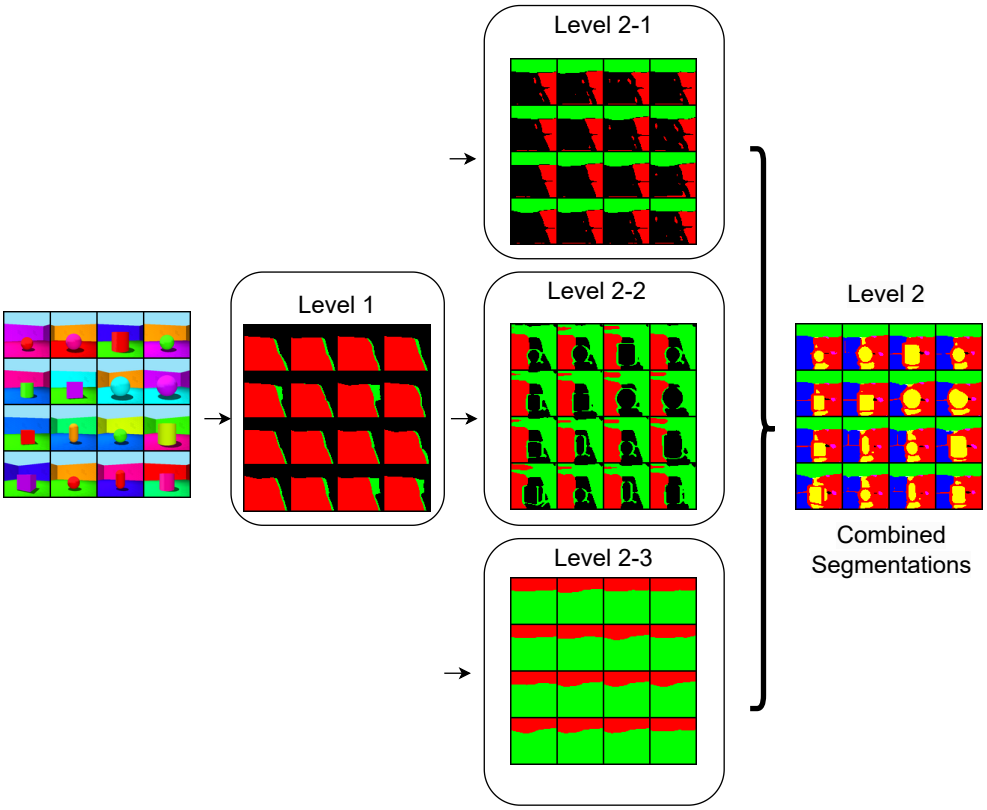
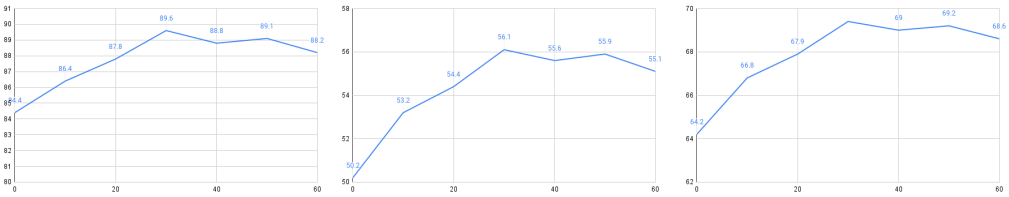


Figure 17: **Segmentations for each level.** ***Level 1:*** 3-color-coded region assignments at the first level. ***Level 2-1, 2-2, 2-3:*** 3-color-coded region assignments per branch at the second level. ***Level 2 combined segmentations:*** 9-color-coded region assignments at the second level.



(a) **Acc.** (b) **IOU.** (c) **DICE**

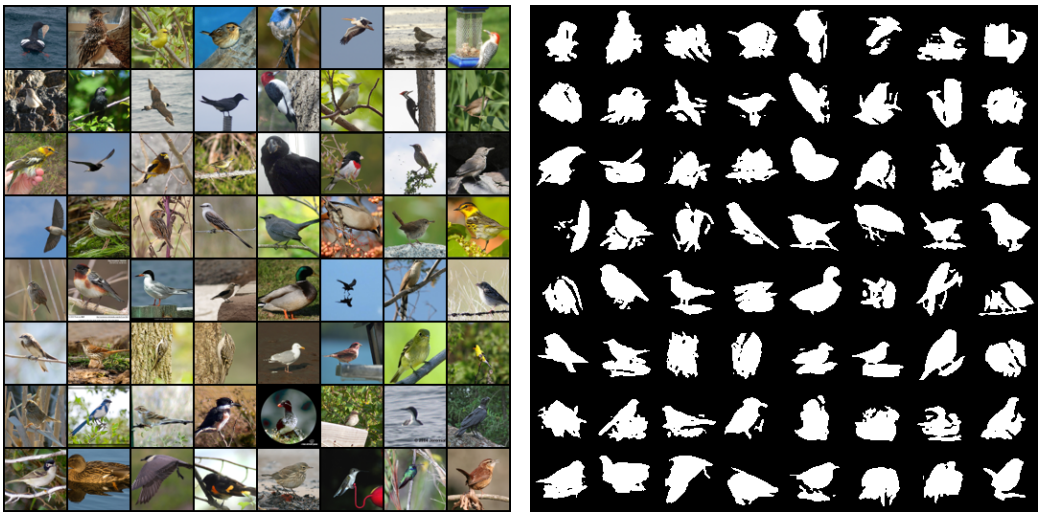
Figure 18: **Segmentation results on CUB with** $t \in \{0, 10, 20, 30, 40, 50, 60\}$**.**

Table 7: We perform class label mapping from ImageNet to VOC, and report zero-shot transfer Accuracy and mIOU per class on VOC validation dataset.

VOC Class.	ImageNet Class.	Num. of VOC-val Image	Accuracy	mIOU
1:aeroplane	895:warplane	136	0.82	0.57
2:bicycle	671:mountain-bike	108	0.79	0.47
3:bird	94:hummingbird	168	0.83	0.58
4:boat	814:speedboat	115	0.81	0.51
5:bottle	907:wine-bottle	133	0.76	0.47
6:bus	779:school-bus	114	0.73	0.54
7:car	817:sports-car	191	0.74	0.48
8:cat	281:tabby	206	0.82	0.66
9:chair	765:rocking-chair	175	0.75	0.64
10:cow	346:water-buffalo	102	0.82	0.45
11:diningtable	532:dining-table	89	0.69	0.62
12:dog	153:maltese-dog	204	0.82	0.67
13:horse	603:horsecart	104	0.84	0.53
14:motorbike	670:motorscooter	117	0.76	0.52
15:person	981:ballplayer	584	0.77	0.46
16:potted plant	883:vase	116	0.74	0.46
17:sheep	348:ram	89	0.84	0.64
18:sofa	831:studio-couch	109	0.73	0.51
19:train	466:bullet-train	126	0.76	0.56
20:tv/monitor	664:monitor	106	0.73	0.47
Average	-	-	0.78	0.54

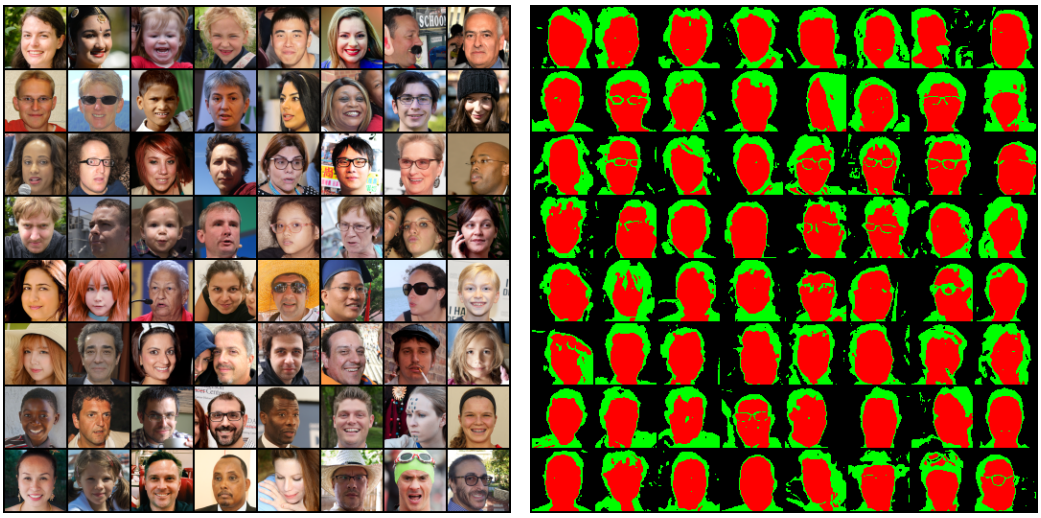
(a) **Real Images** (b) **Segmentation**

Figure 19: Segmentation on Flower.



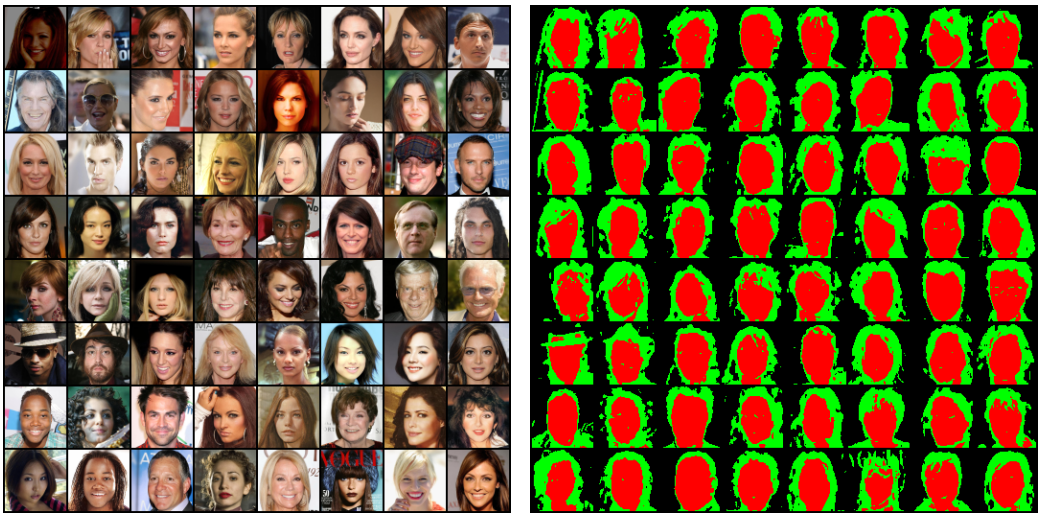
(a) **Real Images** (b) **Segmentation**

Figure 20: Segmentation on CUB.



(a) **Real Images** (b) **Segmentation**

Figure 21: Segmentation on FFHQ.



(a) **Real Images** (b) **Segmentation**

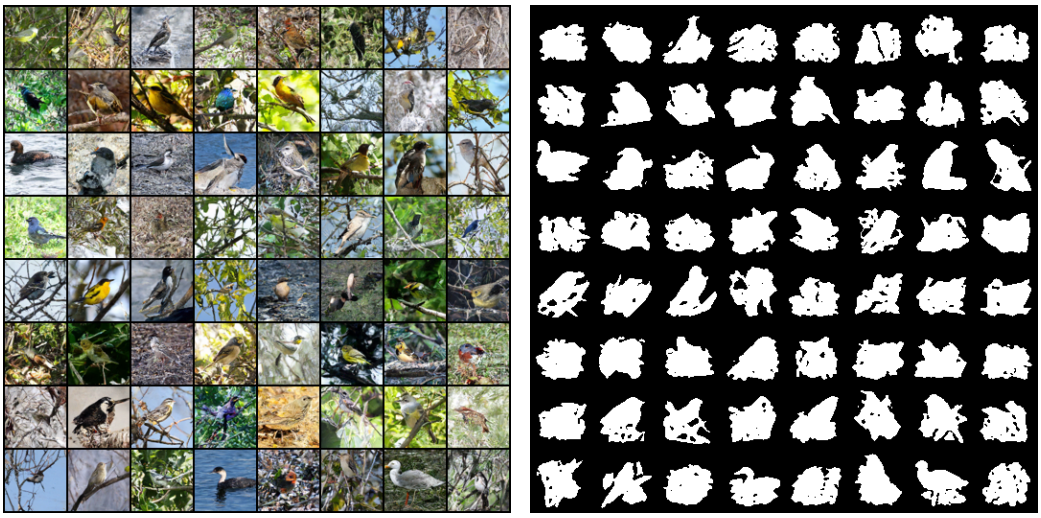
Figure 22: Segmentation on CelebA.

(a) **Real Images** (b) **Segmentation**

Figure 23: Segmentation on ImageNet.

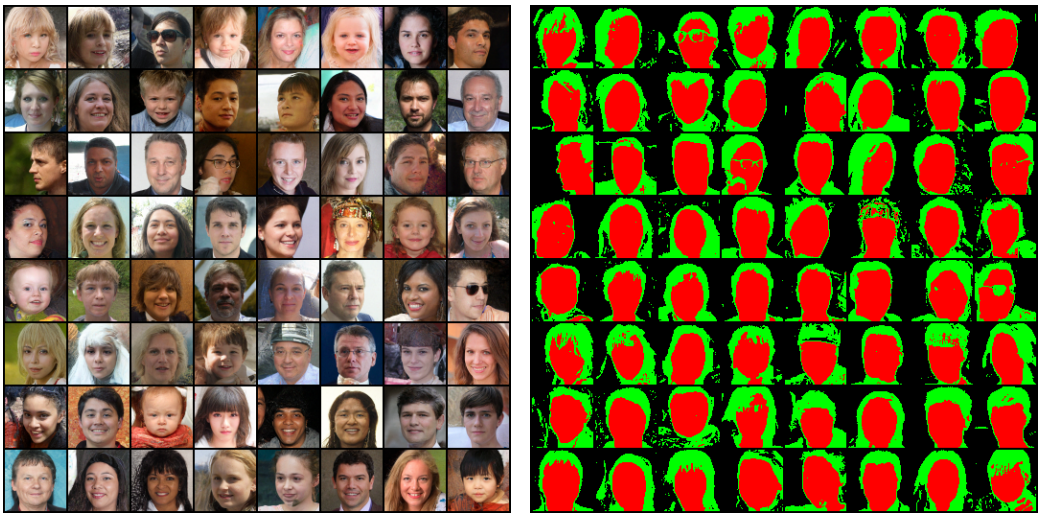
(a) **Generated Images** (b) **Generated Masks**

Figure 24: Generation on Flower.



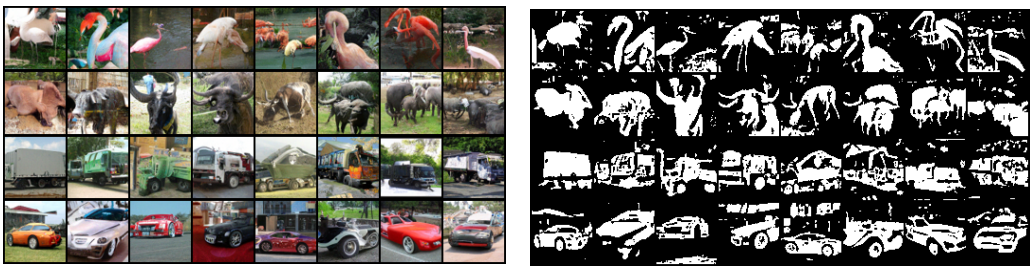
(a) **Generated Images** (b) **Generated Masks**

Figure 25: Generation on CUB.



(a) **Generated Images** (b) **Generated Masks**

Figure 26: Generation on FFHQ.



(a) **Generated Images** (b) **Generated Masks**

Figure 27: Conditional ImageNet generation.

(a) **Real Images** (b) **Segmentation**

Figure 28: Segmentation on VOC-2012.

(a) **Frames of 'Classic-car'**

(b) **Frames of 'Dance-jump'**

Figure 29: Segmentation on DAVIS-2017.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: [NA]

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: [NA]

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: [NA]

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We will release the code upon acceptance.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: [NA]

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [NA]

   Justification: [NA]

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: [NA]

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [NA]

   Justification: [NA]

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.