In the Appendix we provide additional studies, implementation details, and qualitative results.

A. ImageNet Linear Evaluation

We show the results obtained on ImageNet with self-supervised pre-training, via linear probing as well as ImageNet foreground segmentation readout results in Table 12.

Linear probing. We follow SimSiam [14] setup. Specifically, given the pre-trained network, we train a supervised linear classifier on frozen features from the ResNet's global average pooling layer. When training the linear classifier we use a learning rate of $lr = 0.02$ with a cosine decay schedule for 90 epochs, weight decay = 0, momentum = 0.9, a batch size of 4096, and a LARS optimizer [75]. After training the linear classifier, we evaluate it on the centered 224x224 crop in the validation set.

Semantic segmentation readout. We also evaluate the pre-trained backbone's features on the dense prediction tasks on ImageNet. We use the dataset from BigDatasetGAN [40], which has object mask annotation on ImageNet 1k classes. On average, each class has 5 annotated images and in total, it has 6,571 training images and 1,813 testing images. We use the dataset to evaluate the pre-trained backbones on foreground/background semantic segmentation performance. We train an FCN [45] decoder with the frozen pre-trained backbone's features. The FCN readout network is trained using the SGD optimizer with $lr = 0.01$, momentum=0.9 and weight decay=0.0005. We use poly learning rate schedule with power=0.9 and train for 20k iterations. Both training and testing images are center cropped with 256x256 resolution. For all methods, we use ResNet-50 as the image backbone.

Results. Our method with feature distillation and the ADM generative model [20], referred to as *DT-feat.distil w/ ADM* achieves 63.9 Top1 accuracy with linear probing, outperforming all generative pre-trained baselines, and also outperforms competitive discriminative pre-train backbones SimCLR and denseCL. However, we do not outperform the best performing contrastive based method BYOL in this task. Potential explanation could be our pre-training focus on spatial features instead of global discriminative features, which is important for global classification tasks. Notably, our method achieves 79.3 mIoU on the ImageNet foreground/background segmentation task (readout), outperforming all baselines, including dense contrastive based method denseCL, showing the advantage of our method for downstream dense prediction tasks.

Pre-training (ResNet-50)	Eff. epoch	**ImageNet Cls.** *Linear Top1*	**ImageNet Seg.** *Readout mIoU*
Supervised	-	79.3	74.1
Discriminative Pretrain:			
SimCLR [12]	4000	62.5	73.9
denseCL [63]	1600	63.6	76.5
MocoV2 [13]	1600	67.5	76.3
SimSiam [14]	800	69.8	75.0
SwAV [7]	1200	70.4	76.1
BYOL [27]	1600	**71.7**	75.9
Generative Pretrain:			
BiGAN [22]	-	31.0	-
BigBiGAN [23]	-	56.6	-
SparK [60]	1600	54.1	75.6
DT-feat.distil w/ ADM [20]	*600	63.9	**79.3**

Table 12. **ImageNet linear classificiation and semantic segmentation performance.** Our method with ADM includes 400 epochs of generative model pre-training and 200 epochs of feature distillation onto the backbone.

B. BDD100K Pre-training

We show in-domain pre-training on BDD100K instance segmentation task in the main text. Here we show in-domain transfer learning results on two more tasks: semantic segmentation and panoptic segmentation on BDD100K. We further show transfer learning performance on Cityscapes.

In-domain transferring. Following the setup in the main text, we investigate in-domain pre-training in the driving dataset BDD100K. We pre-trained backbones of all methods from scratch using 70k unlabeled images in BDD100K training set. Following the recommendation of [24], we pre-trained contrastive based method BYOL and MIM based method SparK with longer pre-training epochs. We then fine-tune the pre-trained backbone on 7k labeled training dataset on semantic segmentation and panoptic segmentation tasks. In Table 13, we show DreamTeacher using StyleGAN2 and ADM outperform BYOL and SparK pre-training on semantic segmentation and panoptic segmentation task, achieving 60.3 mIoU and 21.8 PQ. However, it does not outperform ImageNet supervised pre-training. One reason can be the unlabelled driving dataset is small and lacks of rare objects and concepts comparing to well-curated object-centred dataset ImageNet.

Cityscapes transferring. Next, we evaluate pre-training on BDD100K [77], and transfer to the Cityscapes [19] semantic segmentation benchmark, which contains 2,975 training images. While both are autonomous driving datasets, they have a domain gap, being captured with different cameras and in different cities/continents. Table 14 shows that our method outperforms all self-supervised baselines significantly. Compared to the best-performing baseline, SparK, our performance gain increases when the number of available downstream labeled examples decreases. We outperform SparK by 1.8%, 2.7%, 4.0% and 4.7%, when fine-tuning on 2,975, 744, 374 and 100 labeled examples.

Pre-training (ResNet-50)	PT task	Eff. epoch	Seg. mIoU	Pan. PQ
Supervised [77]	-	-	61.1	22.4
BYOL [27]	CL	5000	58.4	20.9
SparK [58]	MIM	2500	56.9	20.2
DT-feat.distil. w/ StyleGAN2 [37]	GEN	*900	59.7	21.5
DT-feat.distil. w/ ADM [20]	GEN	*900	**60.3**	**21.8**

Table 13. **In-domain pre-training on BDD100k.** We follow the recommendation of [24] to pre-train contrastive and masking based self-supervised method with long schedule for small dataset like BDD100k with 70k train images. For semantic segmentation, we use UperNet [68] following official implementation from [77]. Reported number is mean IoU at single sacle. For panoptic segmentation, we use panoptic FPN [39], fine-tune for 36($3\times$) epochs, reported number is Panoptic Quality (PQ).

Pre-training (RN-50)	PT task	Eff. epoch	1(2,975)	1/4(744)	1/8(372)	1/30(100)
IN sup init.	-	-	78.7	71.3	65.4	54.4
from stratch	-	-	70.9	41.8	40.7	36.7
BYOL [27]	CL	5000	73.7	54.3	49.9	44.4
Spark [58]	MIM	2500	75.7	61.2	56.0	45.3
DT-feat.distil (ADM)	GEN	*900	**77.5**	**63.9**	**60.0**	**50.0**

Table 14. **Transfer learning: BDD100K to Cityscapes semantic segmentation task.** We pre-trained baselines including ours with unlabeled BDD100K images, and finetuned the backbone on Cityscapes at varying number of labels. Here, we show our method learned transferable features, and achieves the best results in all data portions. Reported numbers are mean IoU, note that effective epochs of our method includes 300 epochs generative model pre-training and 600 epochs feature distillation pre-training.

Pre-training (ResNet-50)	FID (IN1k)	Pre. Data	ADE20k(mIoU)
DT-feat.distil. w/ BigGAN	25.3	Synthetic	40.8
DT-feat.distil. w/ ICGAN	17.0	Synthetic	41.2
DT-feat.distil. w/ ADM	26.2	Real	42.5

Table 15. Ablation study with different unconditional generative models on ImageNet using DreamTeacher.

C. Generative Models Analysis

Here we include additional analysis on the effect of using different generative models with DreamTeacher. In Table 15, we show DreamTeacher with different generative models on ImageNet. For GAN-based models, ICGAN has better generative modelling performance compared to BigGAN in terms of FID (17.0 and 25.3). The backbone pre-trained with ICGAN also has better transfer learning performance in ADE20K (41.2 and 40.8). This observation is in line with the empirical results shown in BigBiGAN paper, which found that generative models with lower FID obtain higher ImageNet classification accuracy. For GANs, we use synthesized data to pre-train the image backbone. For diffusion models, we use encoded data (real images, see Sec. 3.1 main paper), which makes FID scores of generated data less informative. In fact, DreamTeacher with diffusion models performs better in transfer learning, despite ADM's lower generation FID.

D. Architecture

Here we provide additional implementation details about our method's architecture.

D.1. Implementation: Feature Regressors

Our Feature Regressor is implemented as a Feature Pyramid Network (FPN) [42] with additional Pyramid Pooling Module (PPM) [83]. We use pool scale at 1,2,3,6. We add batch normalization and ReLU activation for both the lateral connection and the bottom-up convolution blocks in FPN. We use feature channel size 256. We also add 1x1 conv layer at the end of each level to map the output feature channel to the generator's feature channel.

D.2. Implementation: Feature Interpreter

Feature Interpreter is implemented as a series of Feature Fuse layer to map and fuse generator's features into a logit map. Feature Fuse Layer is implemented by a block of 1x1 conv, bilinear upsampling, concatenation, and Depth-wise Separable Convolution [17]. We applied Group Norm [67] with group number 32 and Swish Activation [49] in-between blocks. The feature dimension is 256. We also apply dropout with rate 0.1 before the 1x1 conv mapping to the output logits.

E. ImageNet Benchmarks

E.1. Implementation: generative models

For GAN based generative models, we use pre-trained unconditional BigGAN [5] from this repository[1], ICGAN [9] from this repository[2]. For diffusion based generative model, we use ADM without classifier guidance, pre-trained with resolution 256x256 from this repository [3]. Please also refer to Table 16 for the hyperparameters used in training the diffusion model on ImageNet.

E.2. Implementation: pre-training

We pre-trained ResNet-50 backbone by diffusing images for 150 steps and running a single denoising step to extract the ADM's UNet decoder features at blocks 3,6,9, and 12. We use the LAMB [76] optimizer with batch size of 2048 and $lr = 4e-3$ with cosine decay learning rate schedule and pre-trained for 200 epochs on the ImageNet training set without class labels. We center crop images into resolution 256x256 and only apply horizontal flipping as the data augmentation. Please see Table 17 for other hyperparameters.

[1] https://github.com/lukemelas/pytorch-pretrained-gans
[2] https://github.com/facebookresearch/ic_gan
[3] https://github.com/openai/guided-diffusion

	LSUN	FFHQ	ImagetNet	BDD100K
resolution	256x256	256x256	256x256	128x256
diffusion steps	1000	1000	1000	1000
noise Schedule	linear	linear	linear	linear
channels	256	256	256	128
depth	2	2	2	2
channels multiple	1,1,2,2,4,4	1,1,2,2,4,4	1,1,2,2,4,4	1,1,2,2,3,4
heads Channels	64	64	64	32
attention resolution	32,16,8	32,16,8	32,16,8	16,8
dropout	0.1	0.1	0.0	0.0
batch size	256	256	256	256
learning rate	1e-4	1e-4	1e-4	1e-4

Table 16. **Hyperparameters for diffusion models used in the paper.**

E.3. Implementation: downstream tasks

ImageNet: classification fine-tuning. For ResNet50, we follow [58] to use the latest open-source ResNet A2 schedule from [65]. For ConvNeXt-B, we use the official implementation. We did not tune the hyper-parameters from [58]. Please refer to SparK official github repo[4] for the hyper-parameters.

ADE20K: semantic segmentation. We use UperNet implemented in MMSegmentation for training ADE20K semantic segmentation task. We use the default hyperparameters from MMSegmentation and train for 160K iterations. We use the same hyperparameters for both the baselines and our methods. After finetuning, we evaluate the performance on ADE20K validation set without using multi-scale flip augmentation.

MSCOCO: instance segmentation. For ResNet-50, we follow [58] to use Mask R-CNN with R50-FPN backbone implemented in Detectron2[5] for MSCOCO instance segmentation task. We use the same configuration as in SparK [58] to finetune the pre-trained backbone for 12 epochs ($1\times$ schedule) or 24 epochs ($2\times$) and evaluate performance on the MSCOCO 2017 validation set. For ConvNeXt-B, we follow the configuration of iBOT [85] to use Cascade Mask R-CNN, fine-tune for 12 epochs. Following the convention, we do not use multi-scale testing, large-scale jittering augmentation, or soft-NMS in all our COCO experiments.

BDD100K: instance segmentation. We use Mask R-CNN with R50-FPN backbone implemented in MMDetection for BDD100K instance segmentation task. We use the official data processing script from this repository[6] and finetuned pre-trained backbone with 36 epochs (3x schedule) and evaluate performance on BDD100K validation set.

[4]https://github.com/keyu-tian/SparK/
[5]https://github.com/facebookresearch/detectron2
[6]https://github.com/SysCV/bdd100k-models/tree/main/ins_seg

F. BDD100K Benchmarks

F.1. Implementation: Generative Models

StyleGAN2. We resize BDD100K images from the original resolution 720x1280 to 512x1024 when training the GAN. We use the default configuration from [37] without adaptive augmentation. We empirically found that turning off path length regularization and style mixing loss improves data sample quality on the BDD100k driving scenes. We train the network with batch size 32 and $gamma = 10.0$ until convergence.

Diffusion Models. We train our own diffusion model on BDD100K, and we summarize training hyperparameters in table 16. We resize BDD100K images from the original resolution 720x1280 to 128x256 resolution. We use diffusion model architecture builds on the U-Net by [20], and we use linear noise schedule from $1e-4$ to $2e-2$.

F.2. Implementation: Pre-training

When DreamTeacher pre-training using the StyleGAN2 generator, we use resolution 512x1024. We use resolution 128x256 for the ADM generator, same resolution as in the generative model training. We use AdamW [46] optimizer with learning rate $lr = 4e-3$ and cosine decay learning schedule to train 600 epochs. Note that we only use horizontal flip as the augmentation during pre-training. Please see Table 17 for information about the hyperparameters.

F.3. Implementation: Downstream Tasks

For all three tasks, we use the official data split and dataset configuration from the official BDD100K repository [7].

Semantic segmentation. We use UperNet [68] implemented in MMSegmentation [8] for evaluating the semantic segmentation task on BDD100K. We train UperNet with the pre-trained ResNet-50 backbone using SGD optimizer with $lr = 0.01$, momentum=0.9, weight decay=0.0005, and poly learning rate schedule with power=0.9 for 80k iterations. We

[7]https://github.com/SysCV/bdd100k-models
[8]https://github.com/open-mmlab/mmsegmentation

	LSUN	FFHQ	ImagetNet	BDD100K
resolution	256x256	256x256	256x256	128x256/512x1024
noise steps	50	50	150	50
optimizer	AdamW	AdamW	LAMB	AdamW
base learning rate	4e-3	4e-3	4e-3	4e-3
weight decay	0.05	0.05	0.05	0.05
optimizer momentum	$\beta_1, \beta_2 = 0.9, 0.95$	$\beta_1, \beta_2 = 0.9, 0.95$	$\beta_1, \beta_2 = 0.9, 0.95$	$\beta_1, \beta_2 = 0.9, 0.95$
batch size	256	256	2048	256
training epochs	100	100	200	600
learning rate schedule	cosine decay	cosine decay	cosine decay	cosine decay
warmup epochs	20	20	30	30
augmentation	horizontal flip	horizontal flip	horizontal flip	horizontal flip

Table 17. **Hyperparameters for pre-training the image backbones.**

use the same training hyperparameters for both the baseline models and our models. After finetuning, we evaluate the model on BDD100K validation set without multi-scale flip test-time augmentation.

Instance segmentation. We use Mask R-CNN [30] with R50-FPN backbone implemented in MMDetection[9] for evaluating the instance segmentation task on BDD100K. We only use the pre-trained weights of ResNet-50 during finetuning and the rest of the networks are randomly initialized. We use SGD optimizer with $lr = 0.02$, momentum=0.9, and weight decay=0.0001 to train Mask R-CNN for 36 epochs (3x schedule). We use the same training hyperparameters for both the baseline models and our models.

Panoptic segmentation. We use PanopticFPN [39] with R50-FPN backbone implemented in MMDetection for evaluating panoptic segmentation task on BDD100K. We only use the pre-trained weights of ResNet-50 during finetuning and the rest of the networks are randomly initialized. We use the SGD optimizer with $lr = 0.02$, momentum=0.9, and weight decay=0.0001 to train PanopticFPN for 36 epochs (3x schedule). We use the same training hyperparameters for both the baseline models and our models.

Cityscapes transfer learning. In this experiment, both the baselines and our method use ResNet-50 backbone pre-trained on 70k unlabeled images from the BDD100K training set. After pre-training, we finetuned the backbone with UPerNet [68] for semantic segmentation tasks in Cityscapes. We follow the public split [16] to split 2,975 training images into 1/4, 1/8, and 1/30 subsets. We use MMSegmentation to train UperNet with 80k iterations for the full set and 20k iterations for the subsets. We use the default hyperparameters for all the baselines and our method. After finetuning, the model is evaluated on the official validation set (5,000 images).

[9]https://github.com/open-mmlab/mmdetection

G. Label-Efficient Benchmarks

G.1. Implementation: generative models

For LSUN cat, horse and bedroom dataset, we use pre-trained ADM models from the guided-diffusion models' repository[10]. For FFHQ, we use pre-trained model from this repository [11], following DDPM-seg [32]. Please see Table 16 for the hyperparmeters used in training the diffusion models.

G.2. Implementation: Feature Interpreter

We train the feature interpreter by first diffusing the real images with 50 time steps, and then extract the ADM's features by running one step of denoising. We use ADM's UNet decoder features at block 3,6,9, and 12. We then train the feature interpreter branch with AdamW optimizer with $lr = 4e{-}3$, weight decay 0.05, $\beta_1, \beta_2 = 0.9, 0.95$, warm-up epochs 20 and train for 100 epochs. We only use horizontal flip when training the feature interpreter.

G.3. Implementation: pre-training

We use the AdamW optimizer with learning rate $lr = 4e{-}3$ and cosine decay learning schedule to train 100 epochs. Please see Table 17 for hyperparmeters for different datasets. Note that we only use horizontal flipping as data augmentation during pre-training.

G.4. Implementation: downstream tasks

We use UperNet implemented in MMSegmentation for semantic segmentation tasks. We use the default hyperparameters in MMSegmentation. For all four tasks, we train UperNet using 20k iteration schedule. For feature distillation, only the backbone is initialized from pre-trained weight, and the rest are randomly initialized. For mix-distillation, we initialize the backbone as well as the UperNet with our pre-trained weights.

[10]https://github.com/openai/guided-diffusion

[11]https://github.com/yandex-research/ddpm-segmentation

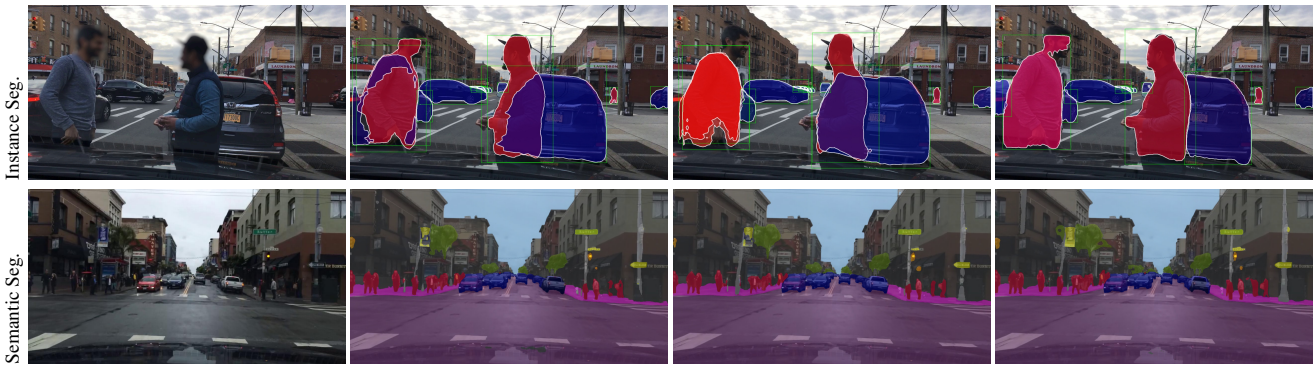

Figure 6. **Qualitative results on BDD100k Inst./Sem. Seg.** Compared with denseCL, our method pre-trained on ImagetNet predicts the correct box on pedestrians and occluded cars, and the mask boundaries are clearer. On semantic segmentation (second row), our prediction segments traffic signs and thin objects like poles. We blur pedestrian faces in the figure, while the methods make predictions on original images.

H. Visualization

We first provide qualitative comparison of instance/sementic segmentation in BDD100K with baseline and then show visualization of ADM denoising network features at different resolution of driving scenes in BDD100K and feature activation maps of a pre-trained ResNet50 backbone using DreamTeacher on ImageNet1k. We then show semantic segmentation results on FFHQ, LSUN-cat, horse and bedroom. The backbone is ConvNX-b pre-trained with our mix-distillation method. We also show semantic, instance and panoptic segmentation results of our method on BDD100K. Note that results of all three tasks are pre-trained using our DreamTeacher feature distillation method on ResNet-50 backbone. We show results on pre-training on BDD100K in-domain data solely and ImageNet-1k-1M as general domain data.

H.1. Qualitative Comparison

In Figure 6, we show qualitative results on BDD100k instance/semantic segmentation task. Comparing to denseCL, our DreamTeacher pre-training method performs better at small/thin objects like traffic signs and poles when fine-tuned on downstream tasks.

H.2. Feature Activation Maps

In Figure 7, we show feature activation maps at different resolution blocks and different noise steps from ADM [20] pre-trained on BDD100k without classifier guidance. We visualize multi-scale features at different resolution blocks, showing features in lower resolution focus on structures, and focus on parts in higher resolution. In Figure 8, we show feature activation maps on ResNet50 backbone pre-trained with DreamTeacher, the backbone learns coarse features at lower level layers and finer features at higher level layers.

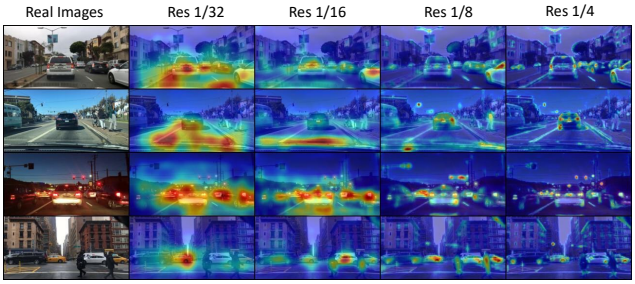

Figure 7. **ADM feature visualization (BDD100k)**. BDD100k image is first diffused by 50 steps and we run one denoising step of the ADM model to extract the feature. We see that features in the low resolution block focus on scene layouts and objects, and in higher resolution, they focus on parts like car wheels, and traffic lights.

Figure 8. **DreamTeacher pre-trained ResNet50 backbone feature activation maps on ImageNet images**. From left to right, we show the image and features at 1/32, 1/16, and 1/8 input resolution.

H.3. Label-Efficient Semantic Segmentation

In Figure 9, we show semantic segmentation results on FFHQ unlabeled images. In Figure 10, we show semantic segmentation results on LSUN-bedroom unlabeled images. In Figure 11, we show semantic segmentation results on LSUN-cat unlabeled images. In Figure 12, we show semantic segmentation results on LSUN-horse unlabeled images. From the visulization, our method is robust to cat in different pose, multi objects occurs in the same images (horse/cat).

H.4. BDD100K: semantic segmentation

In Figure 13, we show semantic segmentation results on BDD100K, pre-trained by DreamTeacher feature distillation on BDD100K unlabeled dataset. And in figure 14, we show results pre-trained on ImageNet unlabeled dataset with our method. Comparing to BDD100K pre-trained, ImageNet pre-trained method works better with rare and small objects like rider and traffic lights.

H.5. BDD100K: instance segmentation

In Figure 15, we show instance segmentation results on BDD100K, pre-trained by DreamTeacher feature distillation on BDD100K unlabeled dataset. and in figure 16, we show instance segmentation results on BDD100 on ImageNet unlabeled dataset. As a comparison, model pre-trained on ImageNet detect and segment small objects better.

H.6. BDD100K: panoptic segmentation

In Figure 17, we show panoptic segmentation results on BDD100K, pre-trained by DreamTeacher feature distillation on BDD100K unlabeled dataset. And in figure 18, we show panoptic segmentation results on BDD100K, pre-trained on ImageNet unlabeled dataset. Note that model pre-trained with BDD100K performs well on things class like road and tree etc, but model pre-trained on ImageNet gets clearer boundary, especially for small objects.

Figure 9. **Semantic segmentation: FFHQ with 34 classes.** Qualitative results of our ConvNX-B model pre-trained with DreamTeacher-feature distillation on FFHQ unlabelled images.

Figure 10. **Semantic segmentation: LSUN-bedroom with 28 classes.** Qualitative results of our ConvNX-B model pre-trained with DreamTeacher-feature distillation on LSUN-bedroom unlabelled images.

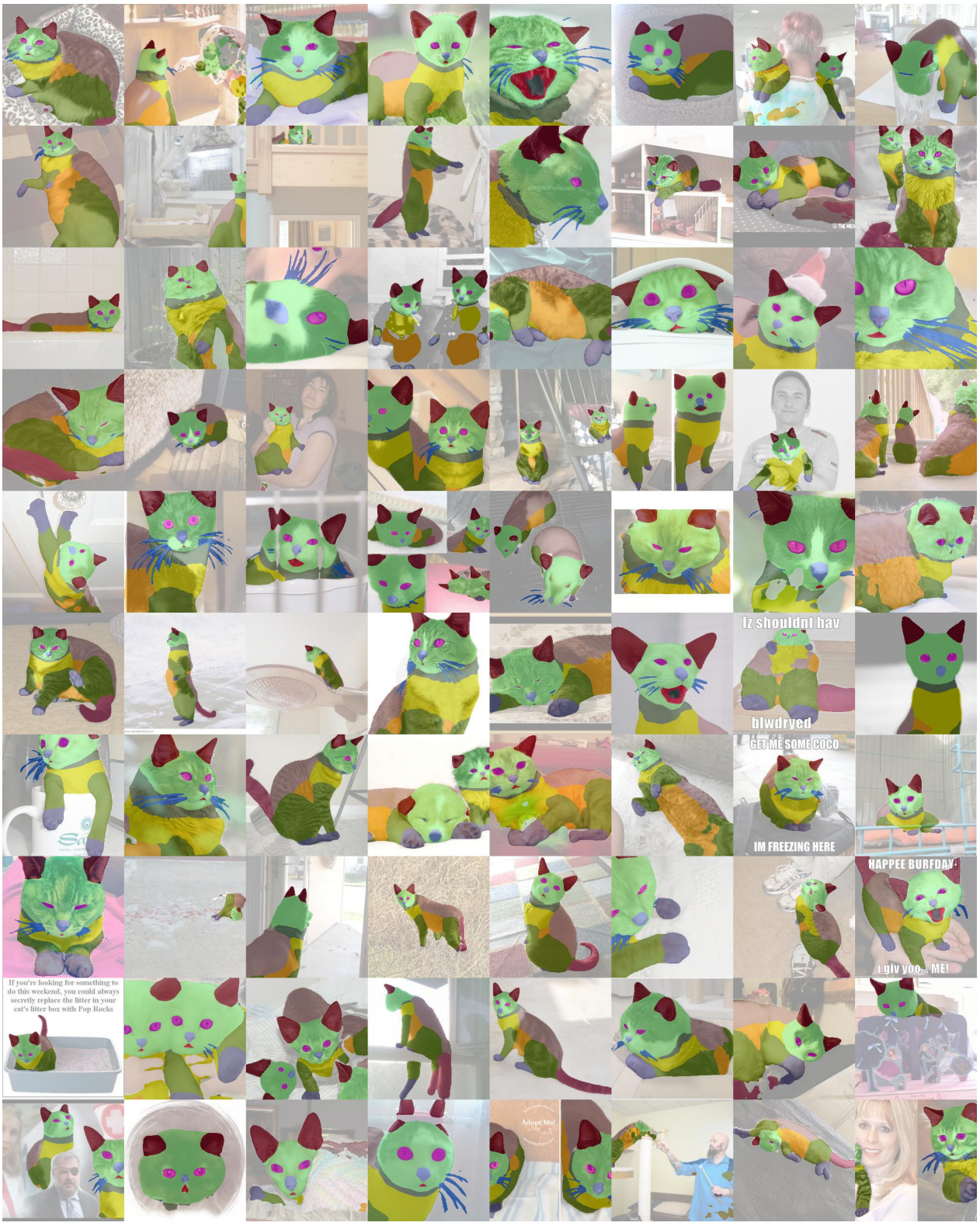

Figure 11. **Semantic segmentation: LSUN-cat with 15 classes.** Qualitative results of our ConvNX-B model pre-trained with DreamTeacher-feature distillation on LSUN-cat unlabelled images.

Figure 12. **Semantic segmentation: LSUN-horse with 21 classes.** Qualitative results of our ConvNX-B model pre-trained with DreamTeacher-feature distillation on LSUN-horse unlabelled images.

Figure 13. **BDD100K semantic segmentation visualization: pre-trained with DreamTeacher feature distillation on BDD100K.** The backbone is resnet-50, finetuned using UperNet.

Figure 14. **BDD100K semantic segmentation visualization: pre-trained with DreamTeacher feature distillation on IN1k-1M.** The backbone is resnet-50, finetuned using UperNet. Only the backbone weight is pre-trained, other part of the networks are randomly initialized.

Figure 15. **BDD100K instance segmentation visualization: pre-trained with DreamTeacher feature distillation on BDD100K.** The backbone is resnet-50, finetuned using Mask R-CNN. Only the backbone weight is pre-trained, other part of the networks are randomly initialized.

Figure 16. **BDD100K instance segmentation visualization: pre-trained with DreamTeacher feature distillation on IN1k-1M.** The backbone is resnet-50, finetuned using Mask R-CNN. Only the backbone weight is pre-trained, other part of the networks are randomly initialized.

Figure 17. **BDD100K panoptic segmentation visualization: pre-trained with DreamTeacher feature distillation on BDD100k.** The backbone is resnet-50, finetuned using PanopticFPN. Only the backbone weight is pre-trained, other part of the networks are randomly initialized.

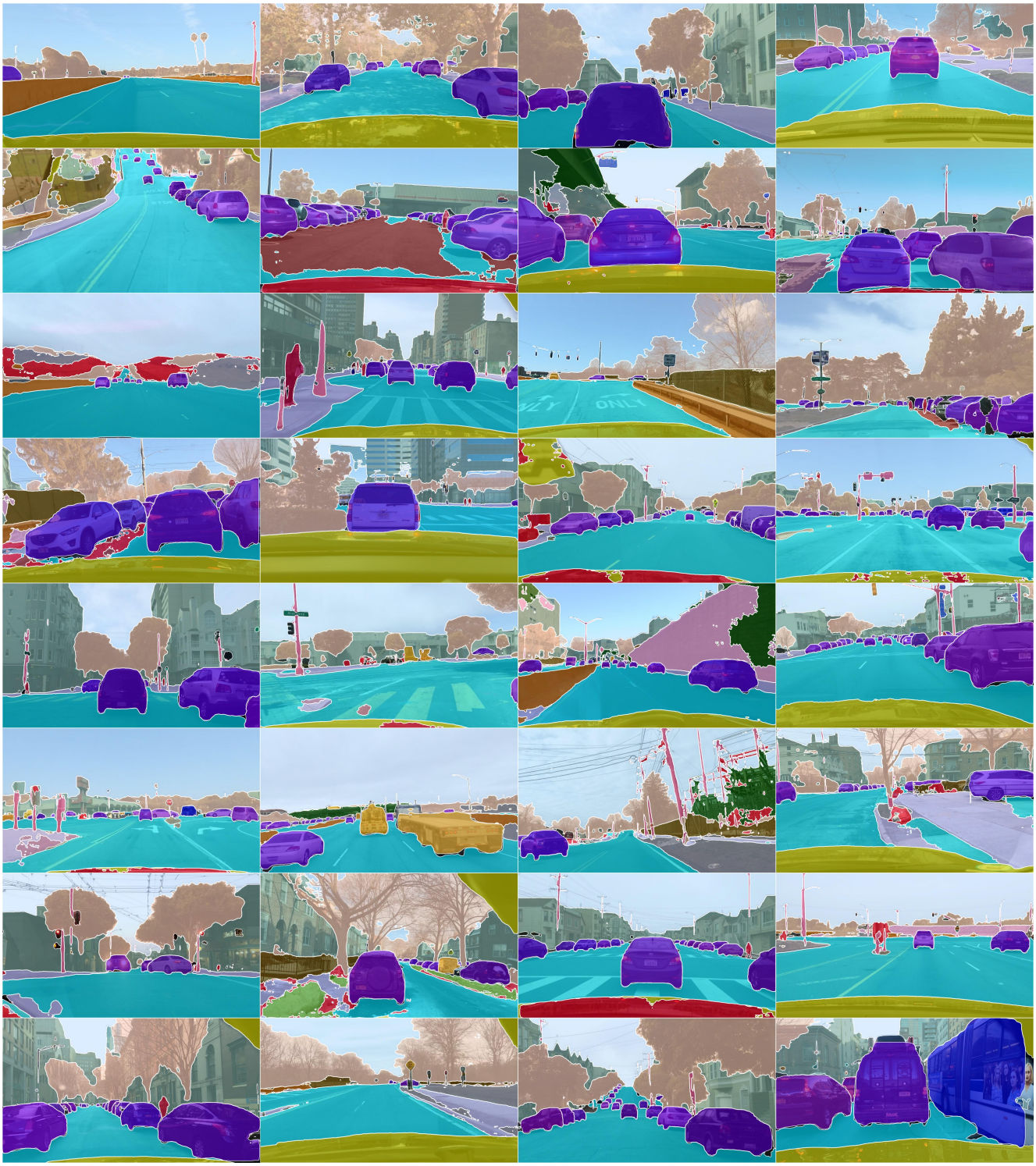

Figure 18. **BDD100K panoptic segmentation visualization: pre-trained with DreamTeacher feature distillation on IN1k-1M.** The backbone is resnet-50, finetuned using PanopticFPN. Only the backbone weight is pre-trained, other part of the networks are randomly initialized.