Contrastive Feature Loss for Image Prediction Supplemental Materials

Alex Andonian[1,3] Taesung Park[2,3] Bryan Russell[3] Phillip Isola[1] Jun-Yan Zhu[3,4] Richard Zhang[3]
[1]MIT [2]UC Berkeley [3]Adobe Research [4]CMU

1. Additional network architectures and training/evaluation details

1.1. Feature spaces

As described in the main paper, our main experiments focus on two primary feature spaces (1) Pixel space and (2) the VGG19 [3] feature space.

Pixel Space. In order to apply our NCE loss to raw pixel space, we begin by cropping the ground truth and generated images into patches of varying sizes: 4x4, 8x8, 16x16 and 32x32 crops with these patch scales separated by strides of 2, 4, 8, 16 pixels respectively. 1024 randomly chosen pixel patches are then "flattened" into 1D vectors before passing through a learned linear linear layer to produce the 256 dimensional representation that participates in the NCE loss.

VGG Feature Space. Here, we use the pretrained VGG19 network available for download using PyTorch's `torchvision` utility library containing common pretrained networks. In keeping with the VGG loss present in SPADE [2], features are extracted from the output of the ReLUs found at layers 2, 7, 12, 21, and 30. Then, 1024 depth columns/fibres (corresponding to spatial patch locations) are randomly selected from the feature tensors and pass through a simple (learned) linear layer to obtain low dimensional embeddings with size 256.

In both the pixel and VGG cases, these embeddings are L2 normalized to lie on the unit hypersphere before participating in the NCE loss. As with previous works, our NCE loss uses a softmax temperature of 0.07.

1.2. NCE Patch Selection.

Number of negative patches. We tested our method with the number of negative patches ranging from as few as 256 to as many 4096 and found it fairly robust to this choice. Since the computational cost of more negative patches appeared to outpaced the corresponding improvements in downstream performance, we opted to run experiments with 1024 patches to strike a balance between performance and efficiency.

Negative mining from same image We experimented with taking negative patches from within the same image vs. negative patches from other images in the same batch and did not observe significant qualitative or quantitative differences. This finding suggests that with enough randomly chosen patches, a sufficient number of proper negatives emerge and allow learning to proceed.

1.3. Optimization details

The generator and discriminator architecture of all our methods and baselines follow those of SPADE [2] for fair comparison. Our models are trained anywhere from 400-1000 epochs using hyperparameter settings similar to [2], with batch sizes ranging from 12-64 examples, Adam optimizer with learning rate 0.0002, and flip-equivariance data augmentation. All experiments were run on 4-8 GPU nodes and took on the order of a few days depending on the particular experimental configuration.

1.4. Evaluation metrics

FID justification for depth experiments. For consistency with the other experiments, we continue to report FID metrics for the depth datasets as the protocol is an emerging standard comparing image distributions. The rationale is that while the pretrained InceptionV3 domain gap may shift the absolute ranges of scores, relative differences in scores should still give some insights into the performance of different models. Therefore, we supplement FID scores with standard depth-specific metrics for our experiments.

1.5. Code

Code is currently available upon request: please contact `andonian.mit.edu` for further details.

2. Results using different encoder backbone architectures

We wish to show that our method's effectiveness does not depend on a particular choice of backbone architecture or feature space. To this end, we evaluate our method's performance using two additional architectures: the popular ResNet50 architecture and a generic 6-layer

Method					Performance		
Backbone	NCE variant	GAN	Pretrained F	Frozen F	FID ↓	mAP ↑	Acc ↑
ImageNet-pretrained ResNet50	Standard	✓	✓	✓	57.3	**64.3**	**82.2**
	Standard	✓	✗	✗	**51.9**	58.8	81.5
	Bidirectional	✓	✓	✓	63.3	62.0	82.1
	Bidirectional	✗	✓	✓	80.3	60.3	81.6
6 Layer CUT ConvEncoder	Standard	✓	✗	✗	60.6	58.7	81.2
	Standard	✗	✗	✗	117.0	48.8	78.0
	Bidirectional	✓	✗	✗	76.6	62.0	81.9
	Bidirectional	✗	✗	✗	86.4	61.5	81.9

Table 1. Performance on Cityscapes dataset using different backbone feature encoders. Surprisingly, the ImageNet-pretrained ResNet50 backbone is capable of achieving the best FID score of all the backbones, variants and baselines investigated in this work. The lower half of the table shows that feature encoders without pretraining can learn useful representations over the course of joint training with the generator.

CNN. For ResNet50, we extract features from the output of the first maxpool layer, and `layers1...layer4` as defined by the standard ResNet50 implementation in `torchvision.models`.

The second feature extraction network for our contrastive loss uses the same architecture as CUT [1], which consists of a simple 6-layer convolutional network followed by a linear projection and L2-normalization. Inputs are first resized to 256x256 pixel resolution via bilinear interpolation. Convolutional layers are followed by a spectral instance normalization layers. Features are extracted from the first 5 layers as well as the raw RGB input pixels. Here, we only adopt the specific architecture of the convolutional encoder network present in CUT [1], while all other details pertaining to the NCE loss formulation remain the same as the main paper.

The results of these experiments and ablations shown in Table 1 indicate that our method works effectively with additional backbone networks. In particular, an ImageNet-pretrained ResNet50 encoder can, in fact, be used to achieve an FID score (51.9) that **outperforms** both the best of our method using a VGG backbone and the SPADE [2] method. Similarly, the ResNet50 backbone also achieves segmentation metrics that rival the VGG based approach.

Given the architecture of the 6 layer CUT encoder, pretrained ImageNet weights are not available and so these experiments are performed with an unfrozen encoder that is trained jointly with the generator. The lower half of Table 1 shows that strong segmentation metrics (mAP and pixel Acc.) can be achieved *even with a lightweight encoder without the need for any pretraining on large datasets*. Consistent with previous experiments, the small FID performance penalty incurred by forgoing expensive pretraining can be significantly reduced by employing the bidirectional NCE loss variant over the standard NCE loss and further reduced by including a GAN discriminator loss.

3. Additional qualitative examples

We show more qualitative comparisons to Figure 3 and 4 of the main paper, in Figure 1, 2, 3, (this supplemental).

4. Comparing the PatchNCE loss to GANs

Our PatchNCE loss aims to be a replacement for the L1 regression loss, while complementary to the GAN loss. Still, the output images produced by the PatchNCE loss bear similar characteristics to the GAN-generated images, such as color saturation and sharpness (Figure 1, 2, and 3). Also, in quantitative metrics (Table 1 of the main paper), using the PatchNCE loss alone without GAN often achieves higher segmentation scores than the GAN + L1 in the VGG feature space, at moderate sacrifice in FIDs. Therefore, the PatchNCE loss may be used without the GAN loss and still produce useful outputs for downstream tasks like domain adaptation. Here we discuss the benefits of using the PatchNCE loss without the GAN loss in image synthesis tasks.

Training stability. Training GANs can be notoriously difficult. Adversarial training is a highly dynamic characterized by instability, particularly in the discriminator network. Models may never converge and mode collapses are common. A growing body of research has focused of improvements to the GAN training procedure to impart stability, which include techniques such as "progressive growing" of the outputs, modifications to the GAN objective through gradient penalties/clipping, and particular normalization layers. Still, debugging GANs can be difficult because objective is inherently adversarial and loss trajectories tend not to be very informative, as shown in figure 4 (left). In contrast, our method trains the generator and feature loss in cooperative fashion, which is reflected in a smooth and interpretable loss trajectory (4, right). As a consequence, we have empirically found that our method can sustain learning rates approximately 5 times larger for a given batch size thus providing an opportunity to increase convergence speed and reduce training time.

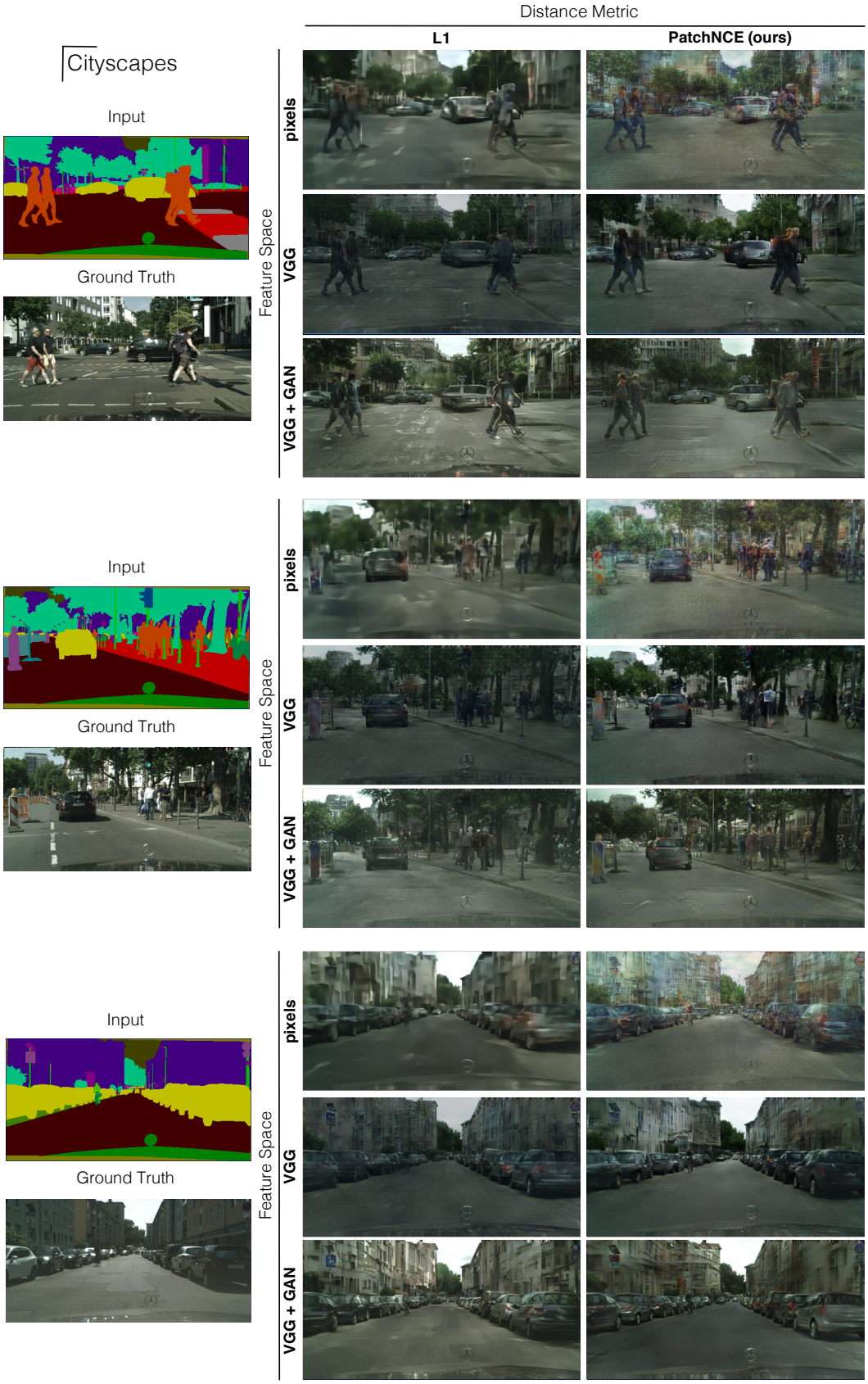

Figure 1. We show additional qualitative comparison results, extending the Cityscapes examples of Figure 3 and 4 of the main paper.

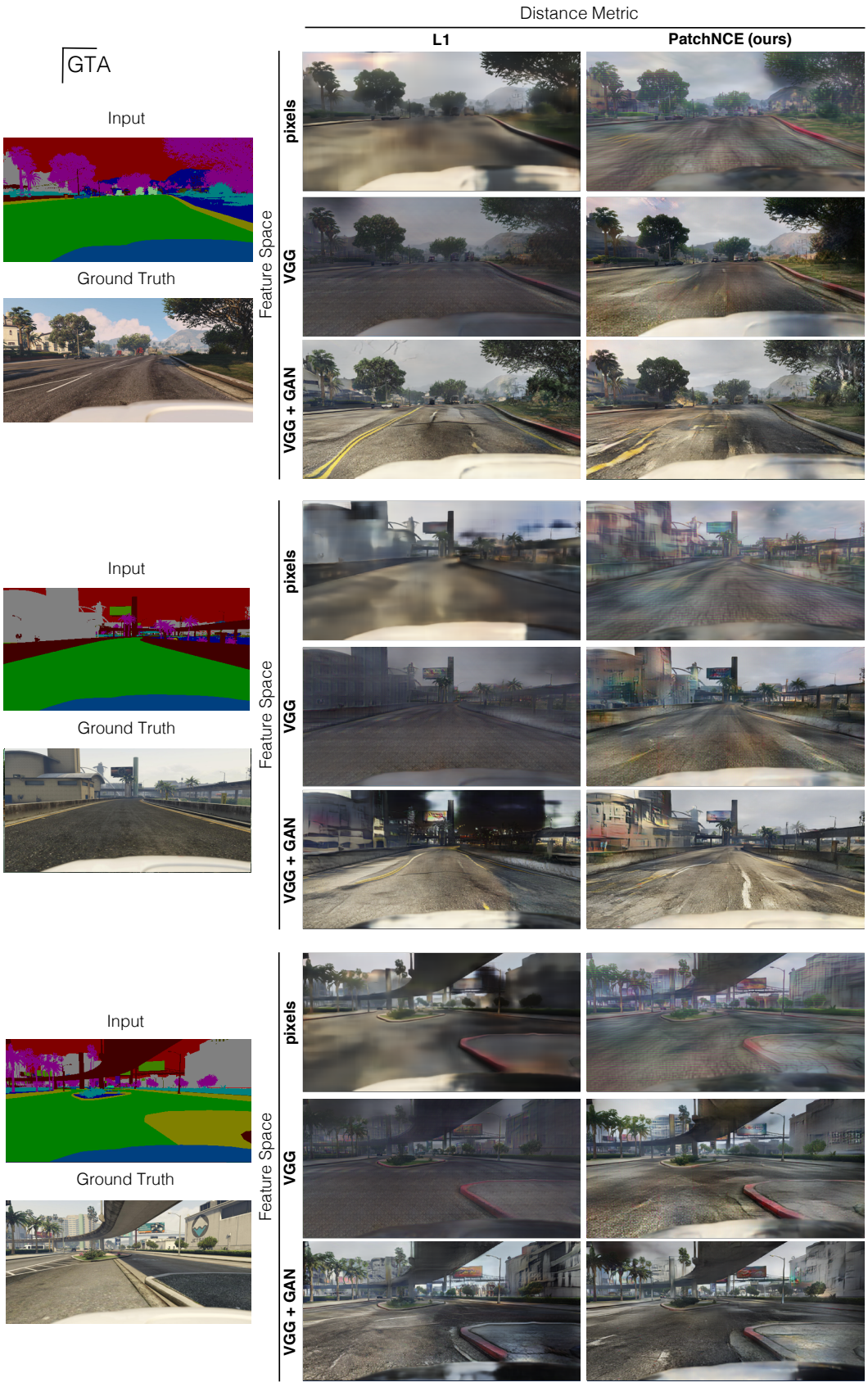

Figure 2. We show additional qualitative comparison results, extending the GTA examples of Figure 3 and 4 of the main paper.

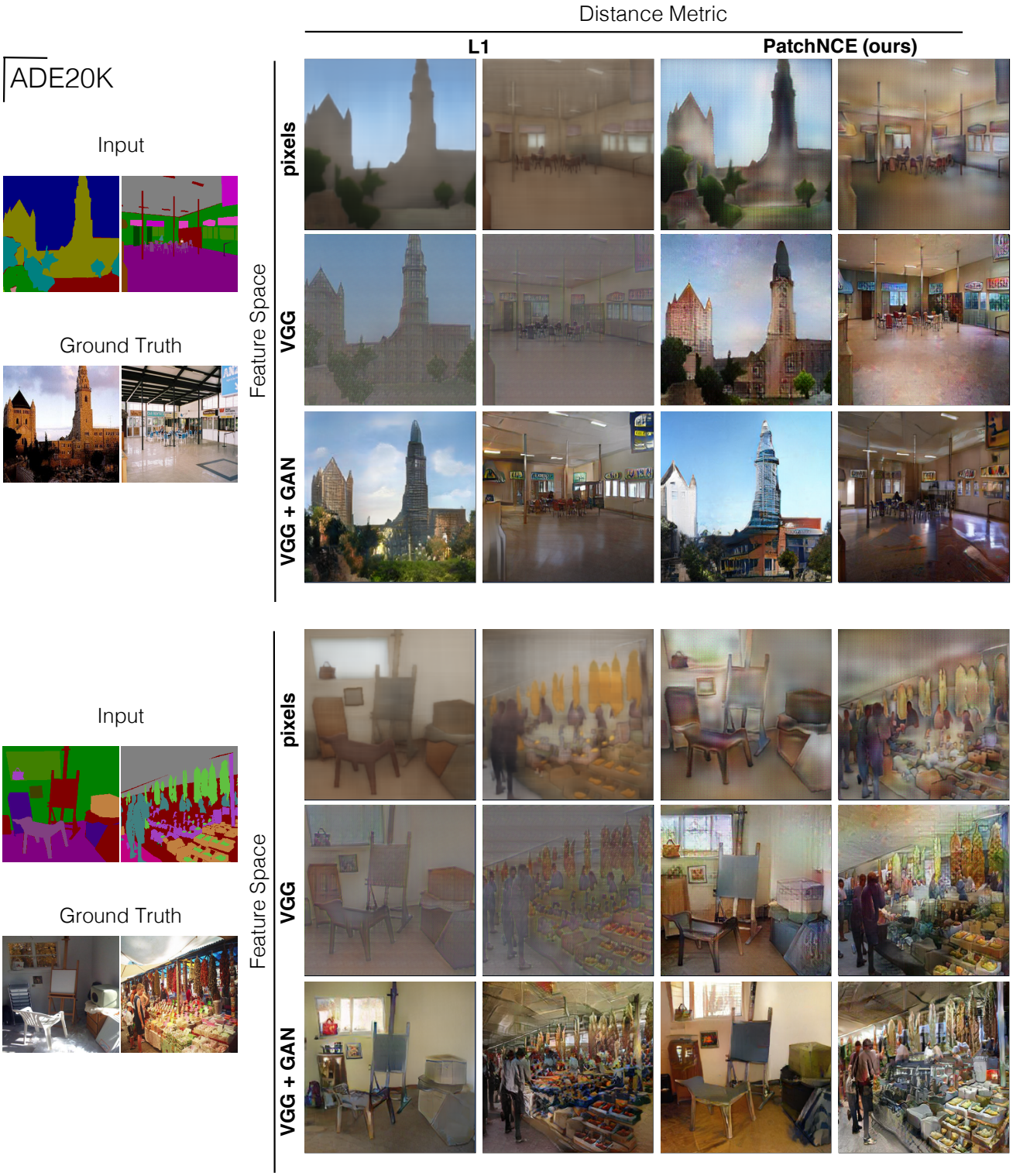

Figure 3. We show additional qualitative comparison results, extending the ADE20K examples of Figure 3 and 4 of the main paper.

Simpler implementation. In modern GAN implementations, training steps alternate between updating the generator and updating the discriminator, requiring two separate optimizers with potentially two different learning rates and additional choices such as updating one network more frequently than the other as with TTUR as seen in [4]. Our

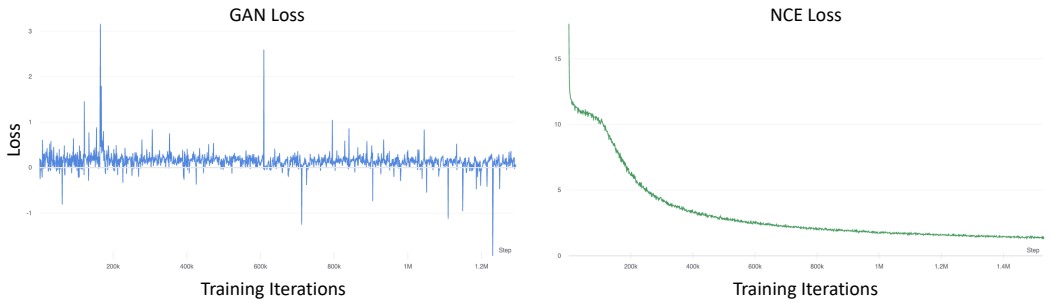

Figure 4. Representative loss trajectories over the course of training for GAN loss (left) and PatchNCE loss (right). GAN losses tend to be noisy with no clear indication of training progress. In contrast, the PatchNCE loss exhibits a clear downward trajectory under normal training circumstances, thus potentially serving as a strong diagnostic/debugging signal for practitioners.

method's training loop is conceptually simpler, and we find that both the generator and encoder can share the same optimizer and hyperparameters while making updates to both networks in a single gradient update. In this way, we eliminate the two step procedure for updating the generator and discriminator separately. Our loss can be incorporated into a full objective nearly as easily as it would be to incorporate an L1 loss term.

Speed and memory comparisons. The results in section 2 demonstrate that our method is robust to choice of encoder. As such, in addition to removing the discriminator network, one can choose a lighter, more modern and efficient networks to serve as the contrastive encoder in our formulation to accommodate the compute and memory constraints of a given circumstance. For example, replacing the pretrained VGG network with ResNet50 reduced the number of weight parameters by over 6x (143.6M vs 23M) with very marginal performance penalties. Further memory reductions are achieved with the 6 layer CUT encoder (9.1M parameters) for a total of nearly 16x few parameters, matching segmentation metric scores and only mild FID score degradation.

Performance comparisons. Our method provides clear advantages in achieving better segmentation metrics such as pixel accuracy and mAP, while still maintaining competitive FID scores.

References

[1] Taesung Park, Alexei A. Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for conditional image synthesis. In *European Conference on Computer Vision (ECCV)*, 2020. 2

[2] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2

[3] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 1

[4] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019. 5