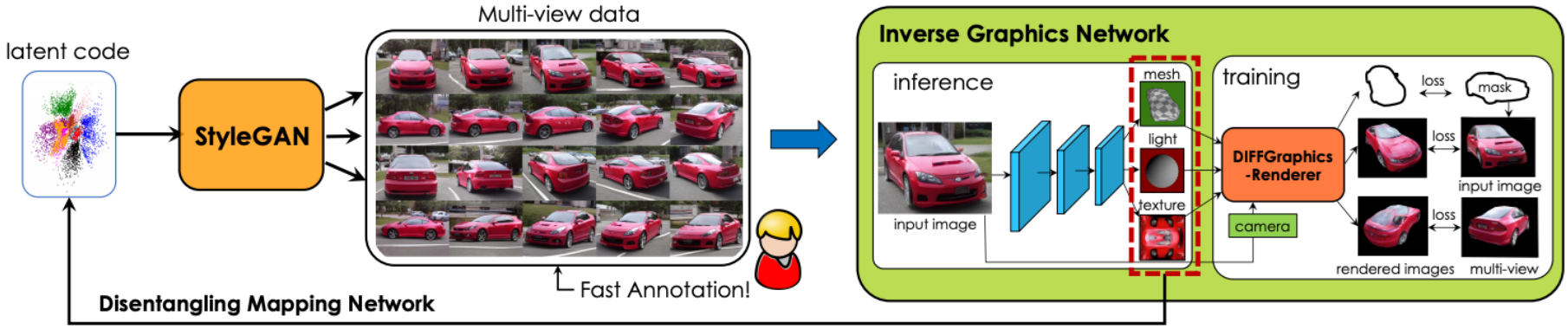


Image GANs Meet Differentiable Rendering For Inverse Graphics And Interpretable 3D Neural Rendering (Ganverse3D)

Zhang, Chen, Ling, Gao, Zhang, Torralba, Fidler

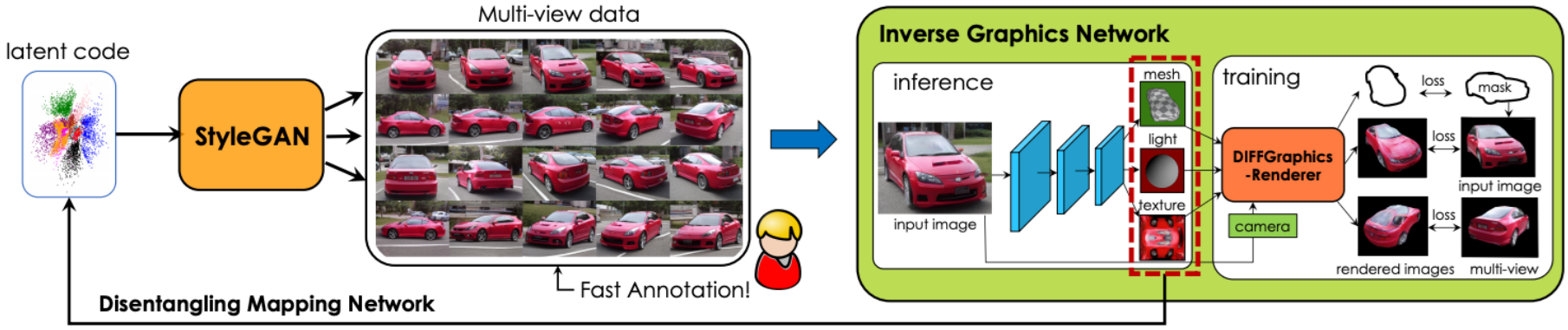
ICLR 2021 oral

Motivation

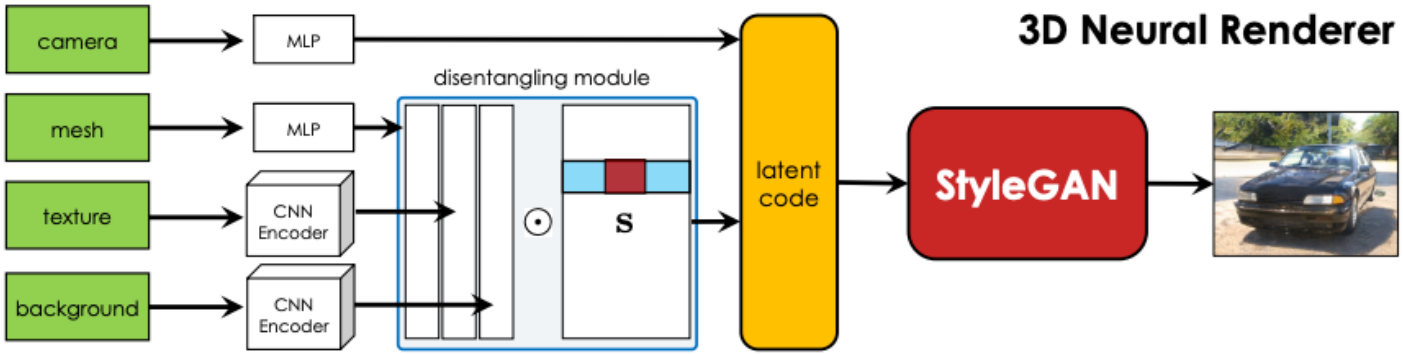
- Inferring 3D properties e.g. geometry, texture, material, etc. from images (Inverse graphics) requires 3D groundtruth (often from synthetic dataset) or multi-view or some keypoint annotation (differentiable renderer).
- Generative models learn 3D information implicitly where manipulating the latent code can produce images with different 3D properties, but the latent code representation is usually not fully disentangled (cannot manipulate shape and material independently).
- This work aims to train differentiable-renderer-based inverse graphics model with multi-view images generated by GANs while making the latent code of the GAN disentangled and interpretable.

Contribution

- Demonstrate the possibility of training 3D reconstruction using GAN generated images (considered easier to obtain than 3D GT or multi-view images or keypoints)
- Making the generative model more interpretable and disentangled.
  - Act like a ‘real’ graphics renderer (takes object shape, texture and viewpoint as input to generate images)
- Technical contributions to make the approach work



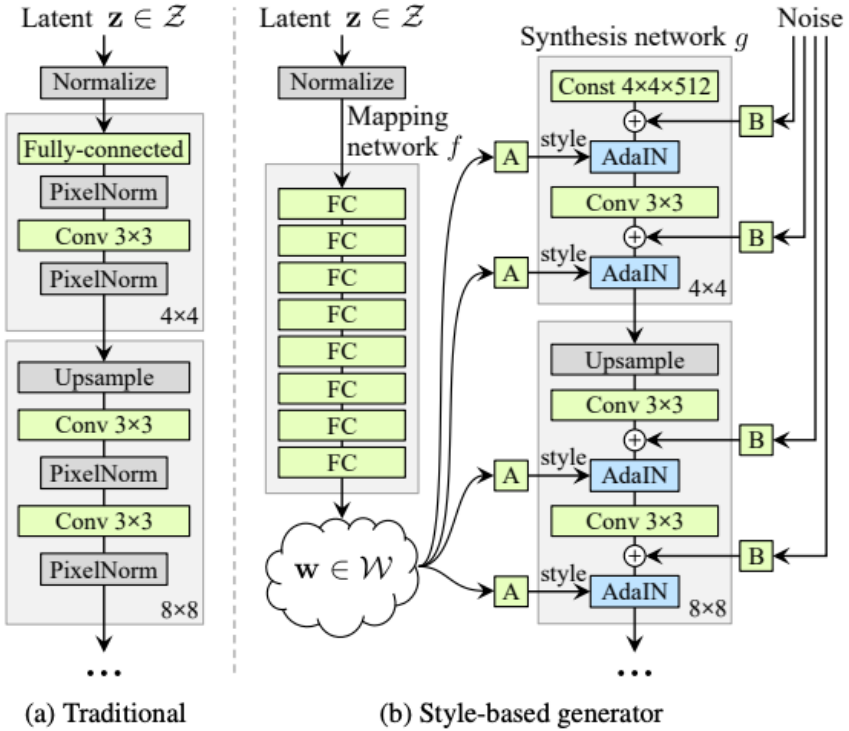
Overview



Mapping network

StyleGAN

- Latent code is fed to the generator through AdaIN at each conv layer
- Observation: styles in early layers adjust the camera viewpoint, styles in the intermediate and higher layers influence shape, texture and background.



StyleGAN as Multi-View Synthetic Data Generator

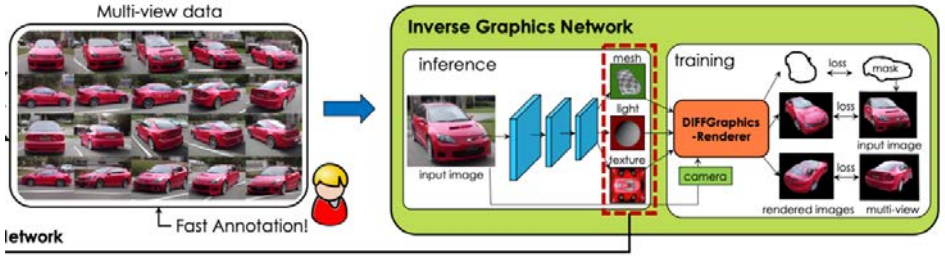
Figure 2: We show examples of cars (first two rows) synthesized in chosen viewpoints (columns). To get these, we fix the latent code w_v^* that controls the viewpoint (one code per column) and randomly sample the remaining dimensions of (Style)GAN's latent code (to get rows). Notice how well aligned the two cars are in each column. In the third row we show the same approach applied to horse and bird StyleGAN.

StyleGAN Multi-view Dataset

- Randomly select several views covering all the common viewpoints of an object ranging from 0-360 in azimuth and roughly 0-30 in elevation
- Manually annotate each viewpoint code by classify it into one of 12 azimuth angles. Elevation is set to 0 and camera distance is fixed. (only takes 1 minute)
- For each viewpoint, sample a large number of content codes to synthesize different objects in the viewpoint
- Filter out bad images. Apply MaskRCNN to get instance segmentation then filter out images with multiple objects or small masks.

Training An Inverse Graphics Neural Network

- {Shape,Texture} = f(I_V)
- Optimize V and f jointly



$$L(I,S,T,V;\theta) = \lambda_{\text{col}} L_{\text{col}}(I,I') + \lambda_{\text{percpt}} L_{\text{pecept}}(I,I') + L_{\text{IOU}}(M,M') + \lambda_{\text{sm}} L_{\text{sm}}(S) + \lambda_{\text{lap}} L_{\text{lap}}(S) + \lambda_{\text{mov}} L_{\text{mov}}(S) \quad (1)$$

Loss function

$$\mathcal{L}_k(\theta) = \sum_{i,j,i\neq j} \left(L(I_{V_i^k}, S_k, T_k, V_i^k; \theta) + L(I_{V_j^k}, S_k, T_k, V_j^k; \theta) \right) \quad (2)$$

$$\text{where } \{S_k, T_k, L_k\} = f_\theta(I_{V_i^k})$$

Multi-view consistency for object k

Disentangling Stylegan With The Inverse Graphics Model

- Mapping Network: mapping 3D property to latent code (w_v, w_mtb)

$$\mathbf{z}^{\text{view}} = g_v(V;\theta_v),\ \mathbf{z}^{\text{shape}} = g_s(S;\theta_s), \mathbf{z}^{\text{txt}} = g_t(T;\theta_t),\ \mathbf{z}^{\text{bck}} = g_b(B;\theta_b),$$

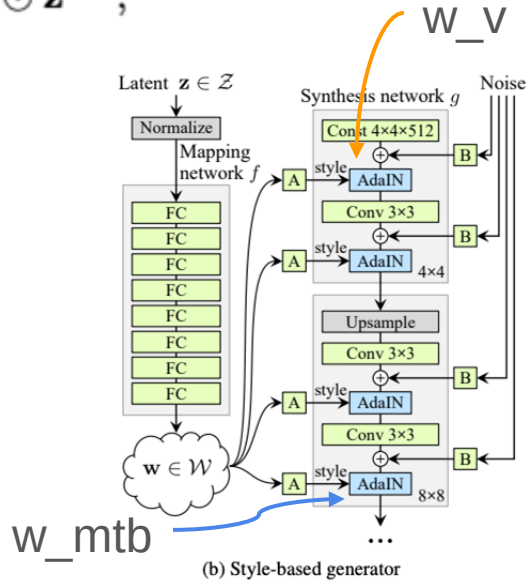
$$\tilde{w}^{mtb} = \mathbf{s}^{\text{m}} \odot \mathbf{z}^{\text{shape}} + \mathbf{s}^{\text{t}} \odot \mathbf{z}^{\text{txt}} + \mathbf{s}^{\text{b}} \odot \mathbf{z}^{\text{bck}},$$

- Disentanglement
  - Each dim of [s^m, s^t, s^b] is normalized using softmax
  - and the entropy is penalized

$$L_{\text{mapnet}}(\theta_v, \theta_s, \theta_t, \theta_v) = ||\tilde{w} - w^*||_2 - \sum_i \sum_{k\in\{m,t,b\}} \mathbf{s}_i^k \log(\mathbf{s}_i^k).$$

  - Finetune StyleGAN using a cycle consistency loss. Randomly sample S, T, B to generate I', then obtain \bar{S}, \bar{T}, \bar{B} from inverse graphics network.

$$L_{\text{stylegan}}(\theta_{\text{gan}}) = ||S - \bar{S}||_2 + ||T - \bar{T}||_2 + ||g_b(B) - g_b(\bar{B})||_2 + ||g_b(\bar{B}_1) - g_b(\bar{B}_2)||_2$$



(b) Style-based generator

Experiments

- 3D reconstruction
- Dual renderer
- 3D image manipulation
  - Controlling Viewpoints
  - Controlling Shape, Texture and Background
  - Real Image Editing

Limitations

- Fail to predict lighting
- Partly succeed at disentangling the background
- Out-of-distribution shape

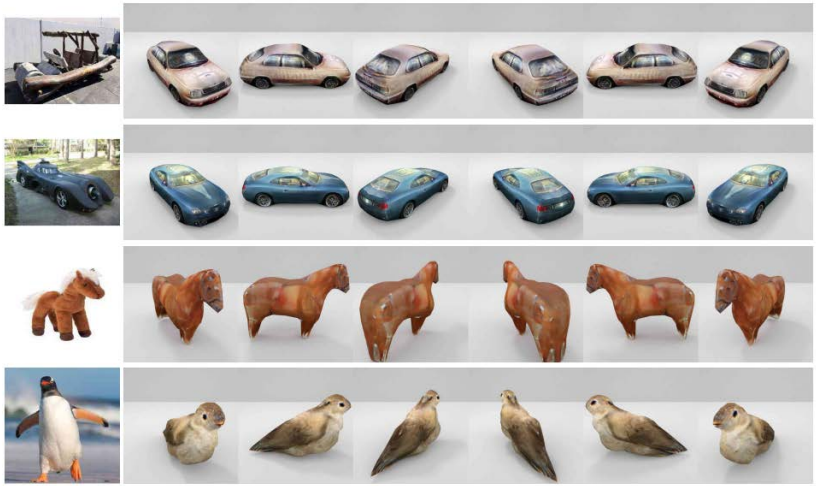
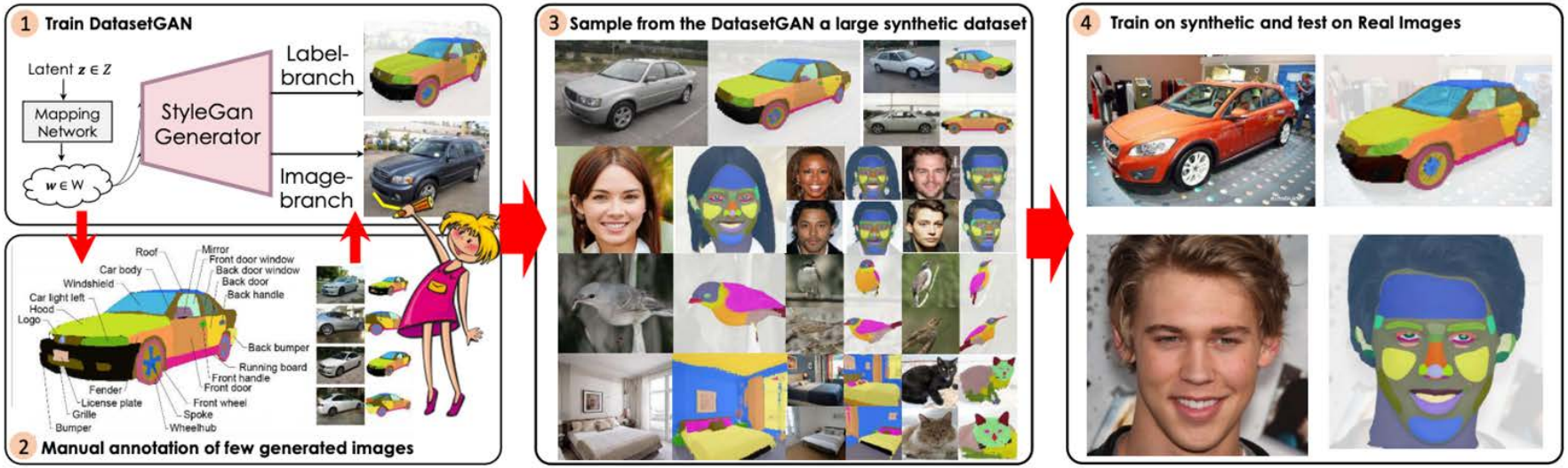


Figure M: **3D Reconstruction Failure Cases**: We show examples of failure cases for car, bird and horse. Our method tends to fail to produce relevant shapes for objects with out-of-distribution shapes (or textures).

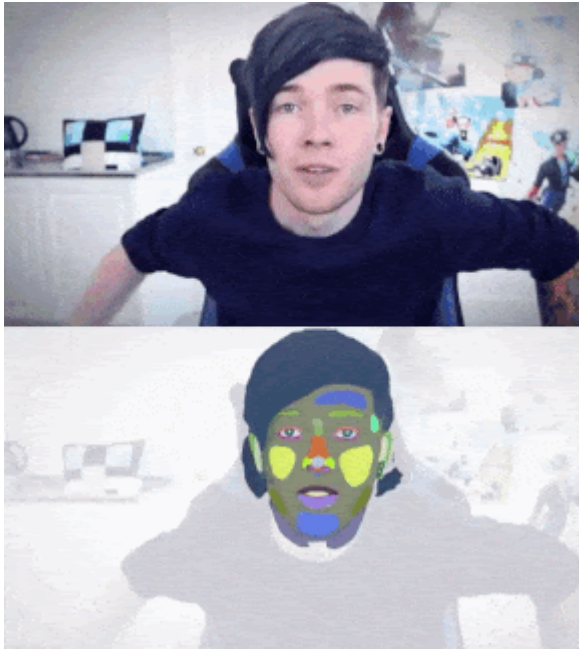


DatasetGAN: Efficient Labeled Data Factory with Minimal Human Effort

Zhang, Ling, Gao, Yin, Lafleche, Barriuso, Torralba, Fidler

CVPR 2021 oral

Left: The video showcases our detailed part segmentation in reconstructing animatable 3D objects from monocular images.



Right: The video shows the result of running a DeepLab trained with a dataset generated from just 16 annotated images.

Motivation

- If generative models can behave like graphics renderers, we may also be able to extract rich semantic knowledge from the intermediate states during rendering just as we do when using a graphics renderer (we annotate 3D model once and generate numerous images with semantic segmentation labels).
- As only a few images need to be manually annotated, we could annotate images in extreme detail

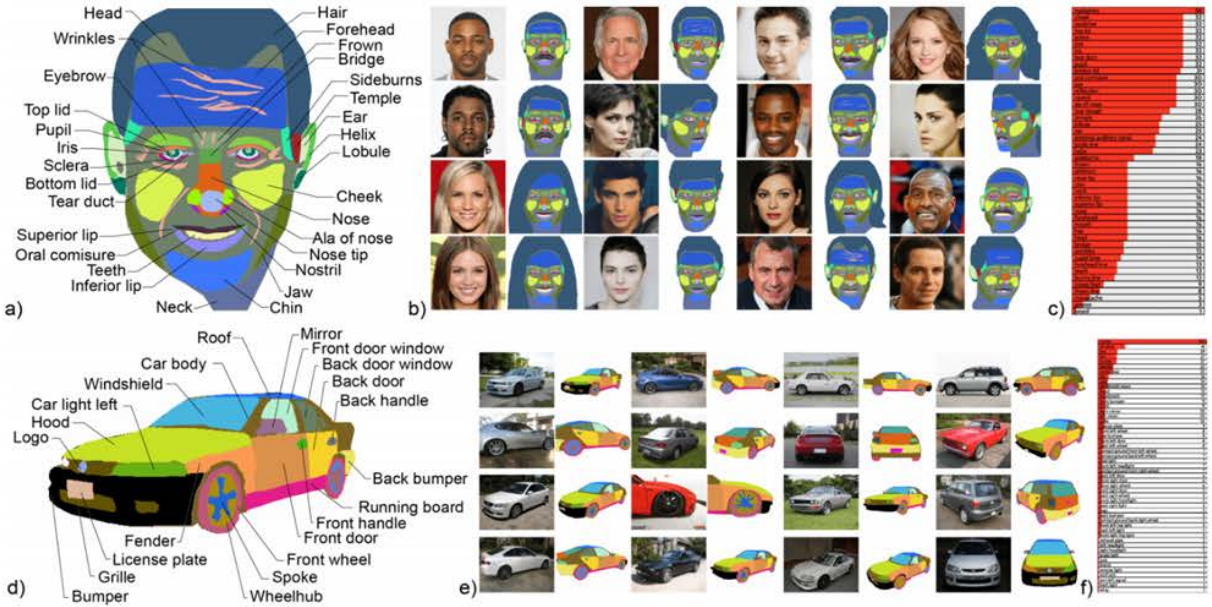


Figure 3: **Small human-annotated face and car datasets.** Most datasets for semantic segmentation (MS-COCO [34], ADE [57], Cityscapes [11]) are too large for a user to be able to check every single training image. In this figure, we show all labeled training examples for face (a-c) and car (d-f) segmentation. a) shows an example of segmentation mask and associated labels, b) shows the full collection of training images (GAN samples), and c) shows the list of annotated parts and the number of instances in the dataset. As a fun fact, note that there are more labels in a single image than there are images in the dataset.

Figure 4: Examples of synthesized images and labels from our DATASETGAN for faces and cars. StyleGAN backbone was trained on CelebA-HQ (faces) on 1024×1024 resolution images, and on LSUN CAR (cars) on 512×384 resolution images. DATASETGAN was trained on 16 annotated examples.

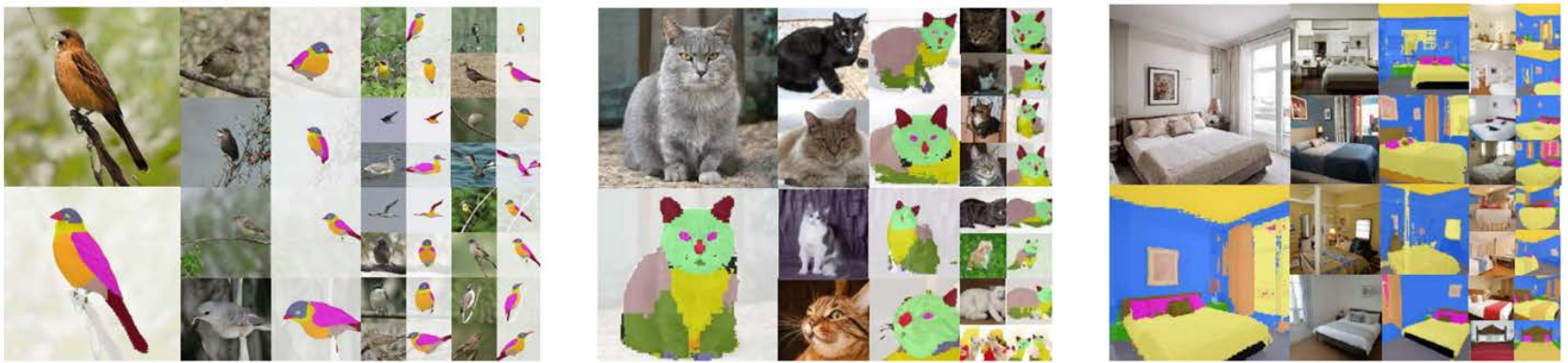


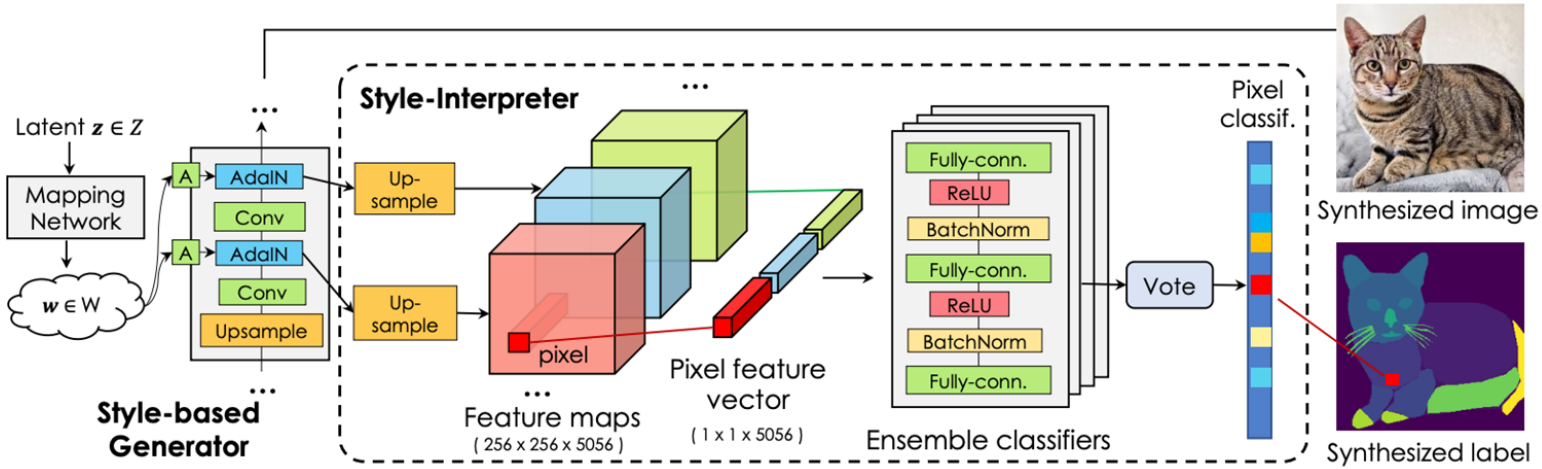
Figure 5: Examples of synthesized images and labels from our DATASETGAN for birds, cats, bedrooms. StyleGAN was trained on NABirds (1024×1024 images), LSUN CAT (256×256), and LSUN Bedroom (256×256). DATASETGAN was trained on 30 annotated bird examples, 30 cats, and 40 bedrooms.

DatasetGAN: annotate a few examples and generalize to the whole latent space

1. Synthesize a small number of images (e.g. 40) using StyleGAN and label them manually
2. Train an ensemble of MLPs (called Style Interpreter) on top of the StyleGAN's pixel-wise feature vectors to match the target human-provided labeling
3. Generate more images with labels using the trained style interpreter
4. Use the ensemble of classifiers to measure the uncertainty of a synthesized example.
    a. Compute JS divergence per pixel.
    b. Image uncertainty is averaged over all pixels.
    c. Filter out the top 10% most uncertain images.

Style Interpreter

- Predict pixel labels from feature vectors of the generator
- Train an ensemble of N=10 classifier and use majority voting at test time for label generation



Experiments

- Part segmentation
- Keypoint detection
- 3D application: single-image 3D asset creation

Part Segmentation: Compare to semi-supervised learning baseline

*denotes In-domain experiment where training and testing are conducted on the same dataset but a different split. Otherwise, training is conducted on the DatasetGAN-generated images (10k by default).

Testing Dataset	ADE-Car-12	ADE-Car-5	Car-20	CelebA-Mask-8 (Face)	Face-34	Bird-11	Cat-16	Bedroom-19
Num of Training Images	16	16	16	16	16	30	30	40
Num of Classes	12	5	20	8	34	11	16	19
Transfer-Learning	24.85	44.92	33.91 ± 0.57	62.83	45.77 ± 1.51	21.33 ± 1.32	21.58 ± 0.61	22.52 ± 1.57
Transfer-Learning (*)	29.71	47.22	✗	64.41	✗	✗	✗	✗
Semi-Supervised [41]	28.68	45.07	44.51 ± 0.94	63.36	48.17 ± 0.66	25.04 ± 0.29	24.85 ± 0.35	30.15 ± 0.52
Semi-Supervised [41] (*)	34.82	48.76	✗	65.53	✗	✗	✗	✗
Ours	**45.64**	**57.77**	**62.33 ± 0.55**	**70.01**	**53.46 ± 1.21**	**36.76 ± 2.11**	**31.26 ± 0.71**	**36.83 ± 0.54**

✗ means that the method does not apply to this setting due to missing labeled data in the domain.

Part Segmentation: Compare to fully supervised baseline

- On par when test in domain
- Surpass when test out-of-domain

Testing Dataset	ADE-Car-5	PASCAL-Car-5
Num of Classes	5	5
Deeplab-V3 [6] (2600 labels)	59.41 (*)	54.31
Ours (25 labels)	57.71	55.65

Table 5: **Comparisons to fully supervised methods for Part Segmentation.** (*) denotes In domain experiments. Deeplab-V3 is trained on ADE-CAR and our model is trained on our generated dataset.

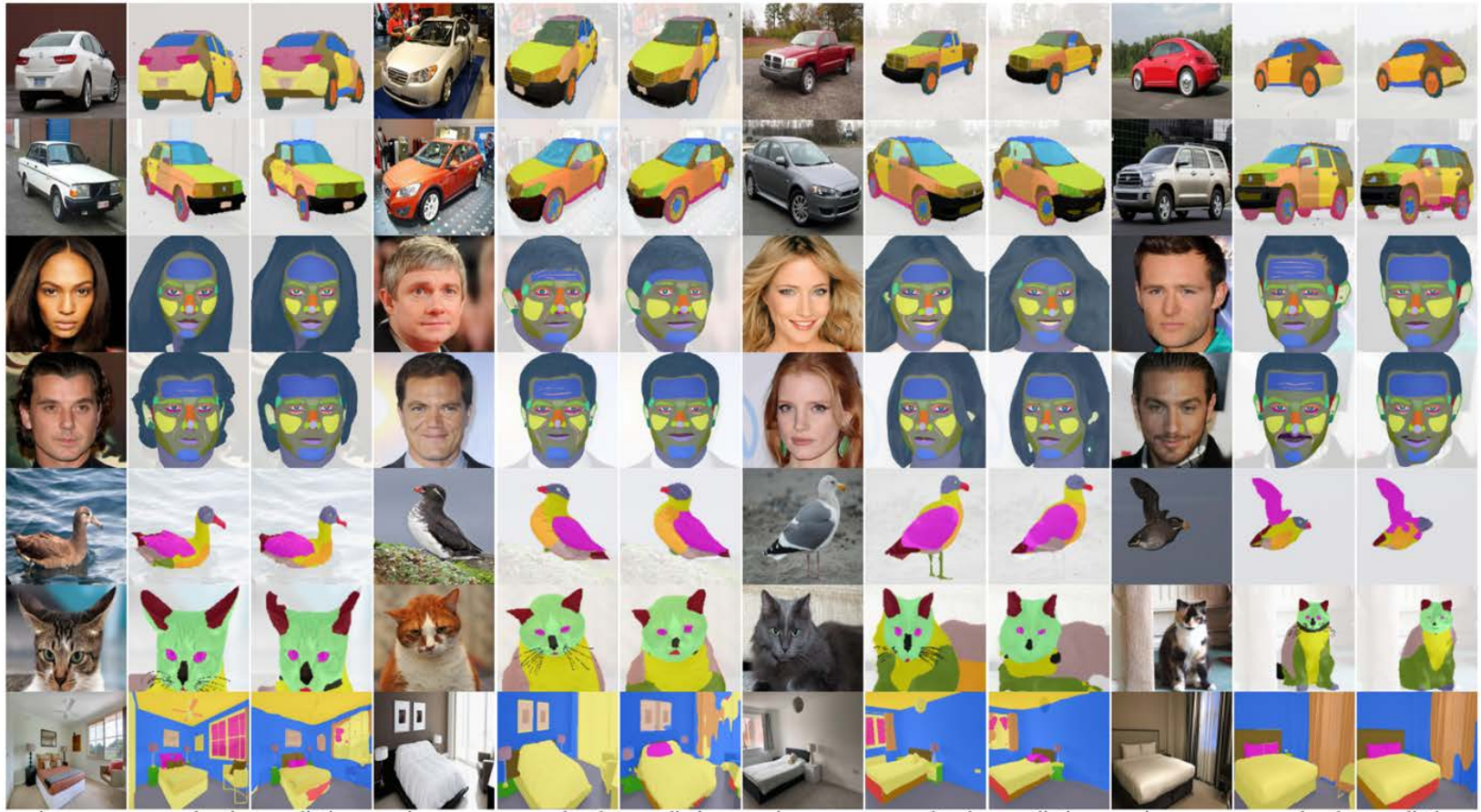


Figure 7: **Qualitative Results:** We visualize predictions of DeepLab trained on DATASETGAN's datasets, compared to ground-truth annotations. Typical failure cases include parts that do not have clear visual boundaries (neck of the cat), or thin structures (facial wrinkles, bird legs, cat whiskers).

Keypoint Detection

Testing Dataset	Car-20				CUB-Bird			
Metric	L2 Loss ↓	PCK th-15 ↑	PCK th-10 ↑	PCK th-5 ↑	L2 Loss↓	PCK th-25 ↑	PCK th-15 ↑	PCK th-10 ↑
Transfer Learn.	4.4×10^{-4}	43.54	36.66	18.53	5.3×10^{-4}	23.17	18.21	12.74
Ours	2.4×10^{-4}	79.91	67.14	35.17	4.3×10^{-4}	60.61	46.36	32.00
Fully Sup.	✗	✗	✗	✗	3.2×10^{-4}	77.54	65.00	53.73

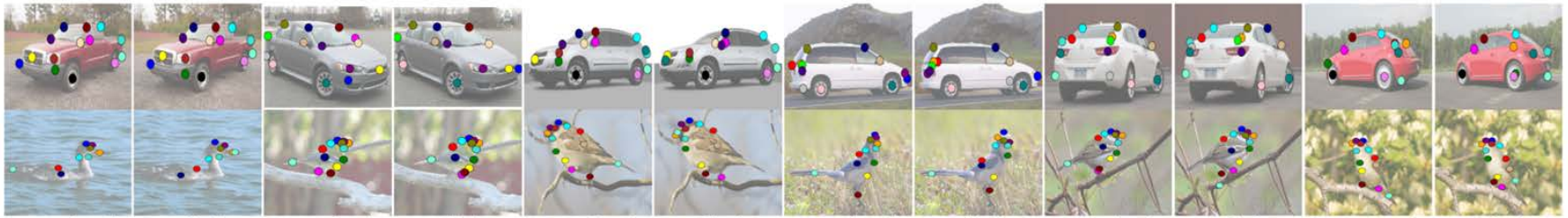


Figure 8: **Qualitative results for Keypoint Detection**. **First row:** Model trained on the generated dataset using 30 human-provided annotations. Results shown are on CUB-Bird test set. **Second row:** Here 16 human-provided annotations are used. Results shown are on Car-20 test set.

3D Application: single-image 3D asset creation

- Utilize the predicted keypoints to improve monocular 3D shape estimation
- Map part segmentation to the estimated 3D model for post-processing
  - 1) placing correct materials for each part such as transparent windshields,
  - 2) creating emissive lighting and
  - 3) repalcing wheels with rigged wheels to enable the estimated 3D cars to drive realistically

Figure 9: **3D Application:** We showcase our detailed part segmentation and keypoint detection in reconstructing animatable 3D objects from monocular images. We follow [55] for training the inverse graphics network, but augment it with 3D part segmentation and 3D keypoint branches, which we supervise with 2D losses. Top left corner shows the input image, keypoint prediction is in the bottom, followed by a rendering of the predicted textured & segmented 3D model into several views. We show an animated scene with lit front and back lights in the last column, which is made possible due to our 3D part segmentation. Cars have physics, rigged wheels, and can be driven virtually. See Supplementary for a video.

3D Application: single-image 3D asset creation

- The pipeline is similar to Ganverse3D, which also predict 3D parts and 3D keypoints
  a. Generate multi-view images with part and keypoint labels using SyleGAN and Style Interpreter
  b. Train an inverse graphics network that accepts an images as input and predicts 3D shape, texture, 3D part labeling and 3D keypoints by utilizing differentiable rendering.
  c. 3D parts are treated like texture. 3D keypoints are represented as a probability distribution over all vertices in the deformed shape.
  d. Use all loss from the Ganverse3D besides an L2 loss on the projected keypoints and Cross Entropy loss on the projected part segmentation.

Discussion

- Adam: use the DatasetGAN to generate adversarial or out-of-distribution images