Bridging the Sim-to-Real Gap: Unsupervised Learning of Scene Structure for Synthetic Data Generation

Supplementary Material

Jeevan Devaranjan*[1,3], Amlan Kar*[1,2,4], and Sanja Fidler[1,2,4]

[1]NVIDIA [2]University of Toronto [3]University of Waterloo [4]Vector Institute

In the Supplementary Material, we present all experimental details in Sec. 1 and show additional qualitative results in Sec. 2.

1 Experimental Details

1.1 Loss gradient scale

We estimate the log ratio $\ln \frac{q_f}{p_f}$ using kernel density estimators. The magnitude of this value can be large and lead to unstable training. We scale this value by 10^{-2}. This was chosen empirically in order to match the magnitude of the MMD. Since we have $\lambda = 10^{-2}$ in our original loss and $r_{\text{reject}}(F)$ is independent of F and dependent only on the structure generation model we rewrite the loss gradient as

$$\nabla_\theta \mathcal{L} \approx 10^{-2}\left(\frac{1}{M}\sum_{j=1}^{m}(\ln \tilde{q}_f(\varphi(v'_j)) - \ln \tilde{p}_f(\varphi(v'_j)))\nabla_\theta \log q_I(v'_j) + \mathbf{1}_{(1,\infty)}(r_{\text{reject}})\right)$$

where r_{reject} is the rejection rate (rejections per successful sample) of the model when sampling the m images in the batch.

1.2 Multi-MNIST

Feature Network Architecture: We use a ResNet architecture consisting of 6 residual blocks and 3 fully connected layers. We use the standard residual block consisting of two 3x3 convolutions with batch normalization and leaky relu activations. Average pooling is performed before passing the output to the fully connected layers. The final fully-connected layer outputs a vector of dimension 10 which corresponding to the logits for detecting whether a given digit class is in the scene. Since the Multi-MNIST images are grayscale, we duplicate them across channels to create a 3-channel image. The Multi-MNIST images are 256x256. The features used for training our structure generation model come from the average pool layer and the first two fully connected layers.

Structure Generation Model Architecture: The logits used for sampling are generated in an autoregressive fashion using an LSTM. The input of the LSTM is a one-hot

* authors contributed equally, work done during JD's internship at NVIDIA

encoding of the previous rule index (70 dimensional, in the Multi-MNIST case). We used a hidden dimension of 50 for the LSTM. The embedding of the one-hot encodings are computing using a fully connected layer. The output of the LSTM is produced by a fully connected layer which takes the hidden state of the LSTM as input and produces the logits used for rule sampling.

Feature Network Training: We first sample 5000 prior structures and their corresponding images, then train our feature network until we reach 75% accuracy on digit detections (around 10 epochs). We use the sigmoid cross entropy loss, and use the ADAM optimizer with learning rate $\epsilon = 10^{-3}$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ with a batch size of 50.

Structure Generation Model Pre-Training: Using the 5000 prior structures generated for training the feature network, we train our structure generation model to minimize the negative log likelihood of the generated structures. This is done for 10 epochs using the ADAM optimizer with learning rate $\epsilon = 10^{-3}$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ with a batch size of 100.

Structure Generation Model Training: We train using 5000 target images. We use a batch size of $m = 500$ and $l = 500$ for both the generated and real images. We train for 20 epochs. We use the ADAM optimizer with a learning rate of $\epsilon = 10^{-4}$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We also use a moving average baseline with $\alpha = 0.05$ in order to reduce the variance of the REINFORCE estimator.

1.3 Aerial 2D

Feature Network Architecture: We use the same feature network architecture as the Multi-MNIST with the final fully connected layer producing a vector of dimension 100. This vector is then split into 4 vectors of dimension 25 which correspond to the logits for one hot encodings. The one hot encodings represent (ranging from 0 to 24) represent the total number of objects of that class (cars, roads, trees and houses) in the scene. Aerial images are 256x256. The features used for training our structure generation model come from the average pool layer and the first two fully connected layers.

Feature Network Training: We first sample 5000 prior structures and their corresponding images, then train our feature network until we reach 75% accuracy on car counting and road counting while ignoring the accuracy figures for trees and houses. Our loss function is the average of the cross entropy loss for each scene element. We use the ADAM optimizer with learning rate $\epsilon = 10^{-3}$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

Structure Generation Model Pre-Training: Using the 5000 prior structures generated for training the feature network, we train our structure generation model to minimize the negative log likelihood of the generated structures. This is done for 10 epochs using the ADAM optimizer with learning rate $\epsilon = 10^{-3}$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ with a batch size of 100.

Structure Generation Model Training: We train using 5000 target images only (no structure ground truth). We use a batch size of $m = 500$ and $l = 500$ for both the generated and real images. We train for 20 epochs. We use the ADAM optimizer with

a learning rate of $\epsilon = 10^{-4}$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We also use a moving average baseline with $\alpha = 0.05$ in order to reduce the variance of the REINFORCE estimator.

1.4 3D Driving Scenes

Structure Generation Model Architecture: We use the same architecture as the previous experiments except we use 3 models in conjunction with three scenarios, as defined by [4]. Each scenario has a slightly different grammar, as described below.

Scenario Grammars: Different grammars are used for each scenario following SDR. The most general is the city scenario grammar

$$
\begin{aligned}
\text{S}_{\text{city}} &\rightarrow \text{Street}\ \ \text{Outer}_L\ \ \text{Outer}_R \\
\text{Street} &\rightarrow \text{Median Cars} \mid \text{Median Cars Cars} \mid \text{Cars Median Cars} \mid \\
&\quad \text{Cars Median Cars Cars} \mid \text{Cars Cars Median Cars} \mid \\
&\quad \text{Cars Cars Median Cars Cars} \\
\text{Outer}_L &\rightarrow \text{Sidewalk Buildings Buildings Foliage} \\
\text{Outer}_R &\rightarrow \text{Sidewalk Buildings Buildings Foliage} \\
\text{Cars} &\rightarrow \text{car} \mid \epsilon \mid \text{Cars} \\
\text{Foliage} &\rightarrow \text{tree} \mid \epsilon \mid \text{Foliage} \\
\text{Buildings} &\rightarrow \text{building} \mid \epsilon \mid \text{Buildings} \\
\text{Sidewalk} &\rightarrow \text{pole} \mid \text{sign} \mid \text{pedestrian} \mid \text{object} \mid \text{bike} \mid \epsilon \mid \text{Sidewalk} \\
\text{Median} &\rightarrow\ \text{bush} \mid \text{object} \mid \text{tree} \mid \epsilon \mid M
\end{aligned}
$$

The suburban grammar is a restriction of the city grammar, specifically having one collection of buildings in the Outer_L and Outer_R non-terminals.

$$
\begin{aligned}
\text{S}_{\text{suburban}} &\rightarrow \text{Street}\ \ \text{Outer}_L\ \ \text{Outer}_R \\
\text{Street} &\rightarrow \text{Median Cars} \mid \text{Median Cars Cars} \mid \text{Cars Median Cars} \mid \\
&\quad \text{Cars Median Cars Cars} \mid \text{Cars Cars Median Cars} \mid \\
&\quad \text{Cars Cars Median Cars Cars} \\
\text{Outer}_L &\rightarrow \text{Sidewalk Buildings Foliage} \\
\text{Outer}_R &\rightarrow \text{Sidewalk Buildings Foliage} \\
\text{Cars} &\rightarrow \text{car} \mid \epsilon \mid \text{Cars} \\
\text{Foliage} &\rightarrow \text{tree} \mid \epsilon \mid \text{Foliage} \\
\text{Buildings} &\rightarrow \text{building} \mid \epsilon \mid \text{Buildings} \\
\text{Sidewalk} &\rightarrow \text{pole} \mid \text{sign} \mid \text{pedestrian} \mid \text{object} \mid \text{bike} \mid \epsilon \mid \text{Sidewalk} \\
\text{Median} &\rightarrow\ \text{bush} \mid \text{object} \mid \text{tree} \mid \epsilon \mid M
\end{aligned}
$$

The rural grammar is a further restriction with the addition of a shoulder instead of a sidewalk

$$
\begin{aligned}
\text{S}_{\text{rural}} &\rightarrow \text{Street}\ \ \text{Outer}_L\ \ \text{Outer}_R \\
\text{Street} &\rightarrow \text{Cars Median Cars} \mid \text{Cars Median Cars Cars} \mid \\
&\quad\ \ \text{Cars Cars Median Cars} \mid \text{Cars Cars Median Cars Cars} \\
\text{Outer}_L &\rightarrow \text{Shoulder Foliage} \\
\text{Outer}_R &\rightarrow \text{Shoulder Foliage} \\
\text{Cars} &\rightarrow \text{car} \mid \epsilon \mid \text{Cars} \\
\text{Foliage} &\rightarrow \text{tree} \mid \epsilon \mid \text{Foliage} \\
\text{Median} &\rightarrow \epsilon
\end{aligned}
$$

The Median non-terminal represents a grass or stone median that divides the left and right portions of the street. Note that the order of non-terminals in a production rule does not always reflection position on the scene but rather the order in sampling. The position of a child object within its parent is decided by its parameters, that come from either a prior or are learnt, using the method of [3].

Feature Network Architecture: We use the pool-3 layer of the inception-v3 network pre-trained on imagenet as our feature network.

Structure Generation Model Pre-Training: We sample 2000 prior structures for each scenario. We train the structure generation model of each scenario to minimize the negative log likelihood of the generated structures for the given scenario. This is done for 10 epochs using the default ADAM optimizer with a batch size of 100. We have two different priors for sampling structures. The prior from [3] and a simple prior. Both priors have the same sampling procedure but differ in terms of the distribution. In general sampling is done by sampling a random number of objects to be placed onto each container (the containers being Street, Outer, Shoulder, Sidewalk and Median). The sampling is done using a uniform distribution bounded by n_{min} and n_{max} which determine the minimum and maximum number of objects in the container. In both priors the objects (i.e pole, sign, bike) are sampled with equal probability with replacement from the set of allowable objects in that specific container. An additional detail is determining the number of lanes on the road and whether or not the scene has a Median container and where should it be placed in the rural and urban scenarios. In both priors the number of lanes is sampled uniformly from 1 to 4 and the probability of a road having a median is 0.5. This can be interpreted as selecting a random production rule for the Street non terminal. Where the priors differ is in their choices of $(n_{\text{min}}, n_{\text{max}})$ for each container. In the prior from [3] the bounds are chosen to match the given scenario. For example in the rural setting there is a large bound on the number of foliage while in the urban setting it is smaller. Thus each container has a unique $(n_{\text{min}} n_{\text{max}})$ that is determined by the scenario. In the simple prior we restrict $n_{\text{min}} = 0$ and $n_{\text{max}} = 2$ for all containers in all scenarios.

Structure Generation Model Training: We use a batch size of $l = 300$ for the real images $V = \{v_1, \ldots, v_l\}$. The real images are taken from the train split of the

KITTI dataset. To produce our generated batch we sample $m = 100$ images $B^s = \{v_1^s, \ldots, v_m^s\}$ for each scenario $s \in \{\text{city}, \text{suburban}, \text{rural}\}$. We then get KDE estimates

$$\tilde{p}_f(F) = \frac{1}{3m} \sum_s \sum_{j=1}^{m} K_H(F - \varphi(v_j^s)) \qquad \tilde{q}_f(F) = \frac{1}{l} \sum_{j=1}^{l} K_H(F - \varphi(v_j))$$

The gradient of the loss for a single structure generation model is given by

$$\nabla_\theta \mathcal{L}^s \approx 10^{-2} \left(\frac{1}{M} \sum_{j=1}^{m} (\ln \tilde{q}_f(\varphi(v_j^s)) - \ln \tilde{p}_f(\varphi(v_j^s))) \nabla_\theta \log q_I(v_j^s) + \mathbf{1}_{(1,\infty)}(r^s_{\text{reject}}) \right)$$

so the densities are calculated using the whole generated set but only use images generated by that scenario for the REINFORCE sample. We use the ADAM optimizer with learning rate $\epsilon = 10^{-4}$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We train for 5 epochs and use a moving average baseline with $\alpha = 0.05$ in order to reduce the variance of the REINFORCE estimator.

Mask-RCNN Training: We train a Mask-RCNN [2][1] network on a generated training set and then test on the KITTI validation set [1]. In both the original and simple prior experiments we produce the generated image set by generating 10K samples for each experiment setting, with the scenario being chosen uniformly. For training, we render images with random saturation and contrast, which we observed to work better. Both the good prior and simple prior have 3 experiment settings which correspond to the rows of the Tab. 1 and 2. The training procedure for both experiments is the same.

First, we train on 10K samples from the probabilistic grammar. We use the ADAM optimizer with learning rate $\epsilon = 10^{-3}$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ with a batch size of 2. We finetune every row of Tab. 1 and Tab. 2 from the previous row. For the next two rounds (corresponding to the second and third rows of Tab. 1 and 2), we use learning rates of $5 * 10^{-4}$ and 10^{-4}.

2 Additional Qualitative Results

We show additonal generated images for the simple prior experiment in Fig. 1. Notice how the prior images (left) are mostly empty, while our learnt generated images have a structure that matches the real images much better. We notice that our method is restricted by the quality of the grammar, and therefore cannot generate structures such as parking side lanes (Row 5 KITTI Image) or intersections (Row 7.11 KITTI image) etc. With more work on writing a grammar, we hope to learn much better structures to cover more scenarios. Training for data generation is still very important after the effort on making the grammar, since for different downstream domains (such as driving in different countries), the target dataset can change drastically and appropriate synthetic data must be generated, where learning can help mitigate long cycles of careful tuning of scene parameters by humans.

[1] Using code from: https://github.com/facebookresearch/maskrcnn-benchmark

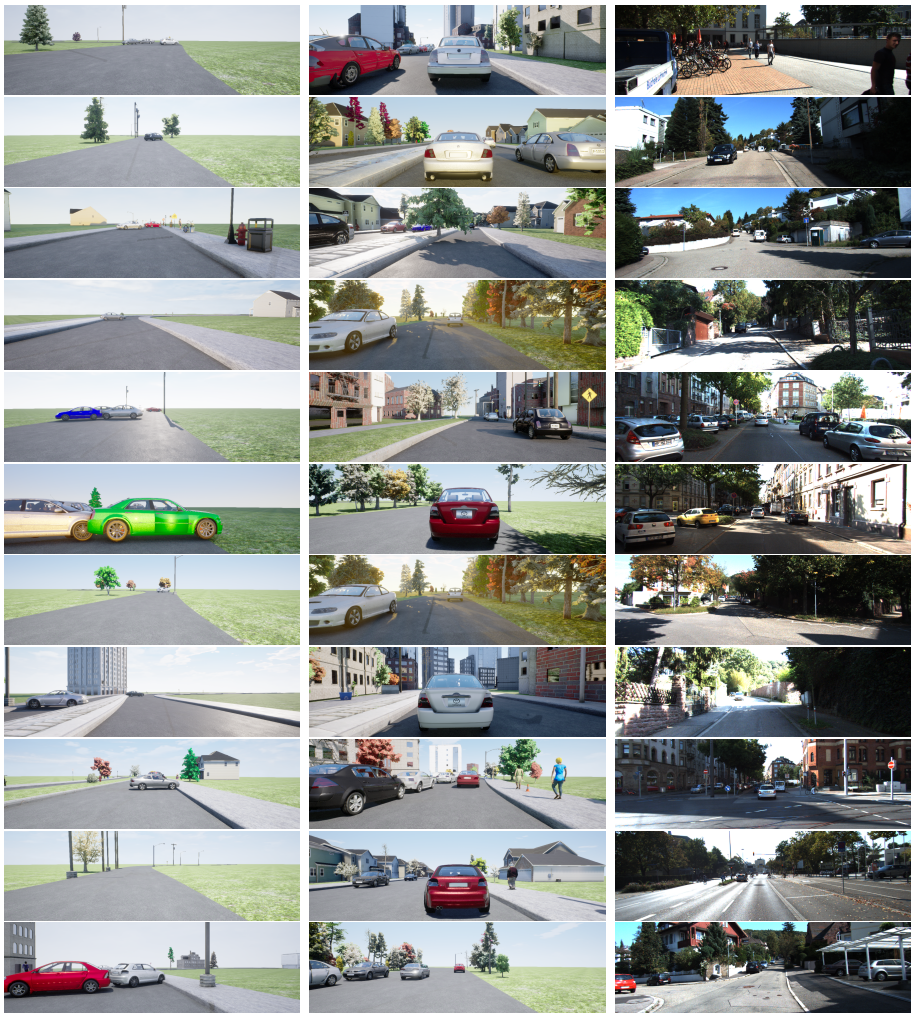

Fig. 1. Random generated samples from the simple prior experiment. (Left) Using both the structure and parameter prior, (Middle) Using our learnt structure and parameters and (Right) random KITTI images Note: images in the same row are not correlated

References

1. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In: CVPR (2012) 5
2. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017) 5
3. Kar, A., Prakash, A., Liu, M.Y., Cameracci, E., Yuan, J., Rusiniak, M., Acuna, D., Torralba, A., Fidler, S.: Meta-sim: Learning to generate synthetic datasets. In: ICCV (2019) 4
4. Prakash, A., Boochoon, S., Brophy, M., Acuna, D., Cameracci, E., State, G., Shapira, O., Birchfield, S.: Structured domain randomization: Bridging the reality gap by context-aware synthetic data. In: arXiv:1810.10093 (2018) 3