Learning To Simulate

Nataniel Ruiz[1], Samuel Schulter[2], Manmohan Chandraker[2]

[1]Boston University [2]NEC-Labs

Motivation

- ❑ Sampling data randomly from a simulator can be very beneficial when data is scarce or annotation is costly [1,2].
- ❑ Previous work simulates large quantities of random scenes for tasks such as semantic segmentation or object detection in traffic scenes [3,4].

Approach

- ❑ Our objective is to learn to simulate better data, which, when trained on yields a model with improved performance.
- ❑ We propose a reinforcement learning-based method for automatically adjusting the parameters of any (non-differentiable) simulator.

Our Simulator

Synthetic images generated by our parameterized simulator. We simulate a straight portion of road with houses and five different types of cars with variable weather and length of road.

Our simulator is a heavily modified version of the CARLA [1] plugin in the Unreal Engine 4 development suite.

Method

➢ We want to solve the following bi-level optimization problem.

$$\boldsymbol{\psi}^* = \arg\min_{\boldsymbol{\psi}} \sum_{(\boldsymbol{x},\boldsymbol{y})\in \boldsymbol{D}_{\text{val}}} \mathcal{L}\left(y, h_{\boldsymbol{\theta}}(\boldsymbol{x}; \boldsymbol{\theta}^*(\boldsymbol{\psi}))\right)$$

— meta-learner that learns *how* to generate data by optimizing ψ

$$\text{s.t.} \quad \boldsymbol{\theta}^*(\boldsymbol{\psi}) = \arg\min_{\boldsymbol{\theta}} \sum_{(\boldsymbol{x},\boldsymbol{y})\in \boldsymbol{D}_{q(\mathbf{x},\mathbf{y}|\,\boldsymbol{\psi})}} \mathcal{L}\left(\boldsymbol{y}, h_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{\theta})\right),$$

— learn model parameters on the generated dataset, this is the main task model which learns to solve the actual task at hand

ψ are the simulator parameters, $h_{\boldsymbol{\theta}}$ is the model parametrized by θ , $\mathcal{L}$ is the loss, $\boldsymbol{D}_{\text{val}}$ is the validation set and $\boldsymbol{D}_{q(\mathbf{x},\mathbf{y}|\,\psi)}$ describes a dataset generated by the simulator distribution $q(\mathbf{x}, \mathbf{y}; \psi)$.

➢ We resort to reinforcement learning to solve this problem since the simulator is non-differentiable in the general case, among other reasons.

➢ We use the vanilla policy gradient method to optimize ψ .

for *iteration=1,2,...* **do**
Use policy π_ω to generate K model parameters ψ_k
Generate K datasets $\boldsymbol{D}_{q(\mathbf{x},\mathbf{y}|\,\psi_k)}$ of size M each
Train or fine-tune K main task models (MTM) for ξ epochs on data provided by $\mathcal{M}_k$
Obtain rewards $R(\psi_k)$, i.e., the accuracy of the trained MTMs on the validation set
Compute the advantage estimate $\hat{A}_k = R(\psi_k) - b$
Update the policy parameters $\omega \leftarrow \omega - \eta \frac{1}{K} \sum_{k=1}^{K} \nabla_\omega \log(\pi_\omega) \hat{A}_k$
end

Algorithm 1: Our approach for "learning to simulate" based on policy gradients.

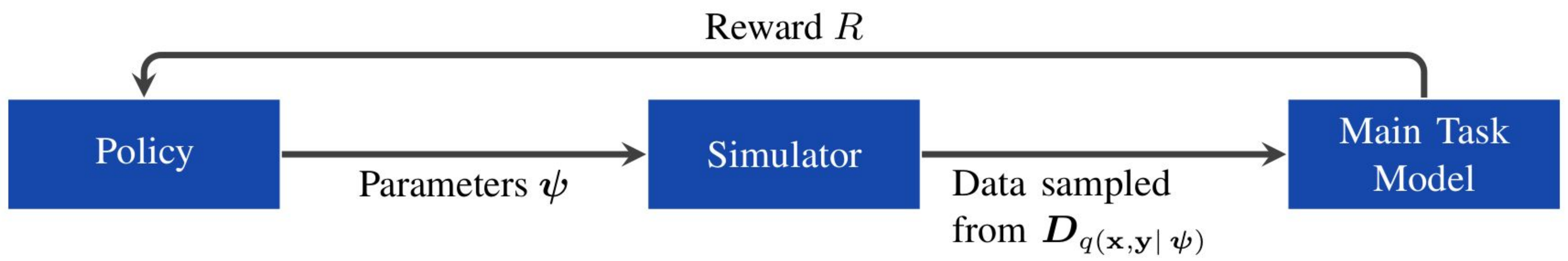


Figure 1: A high-level overview of our "learning to simulate" approach.

Results

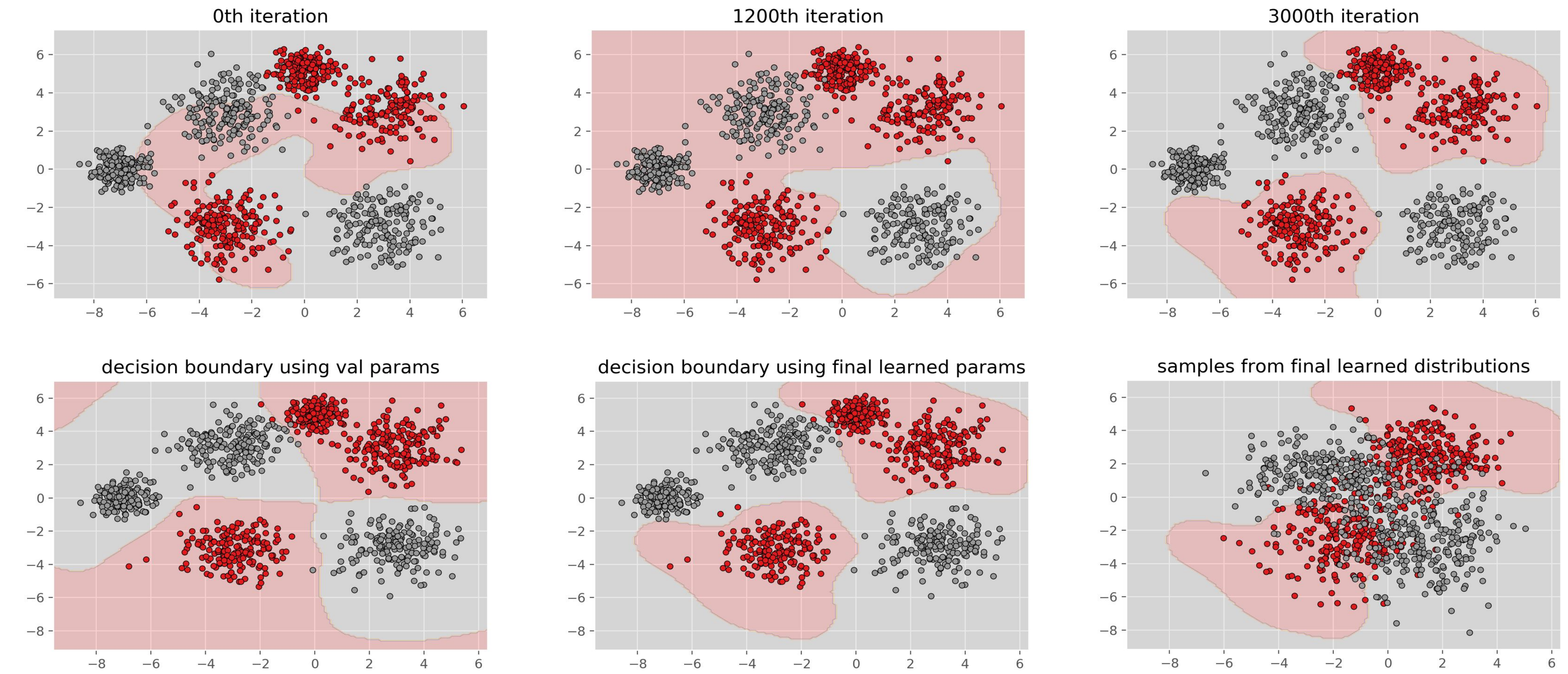
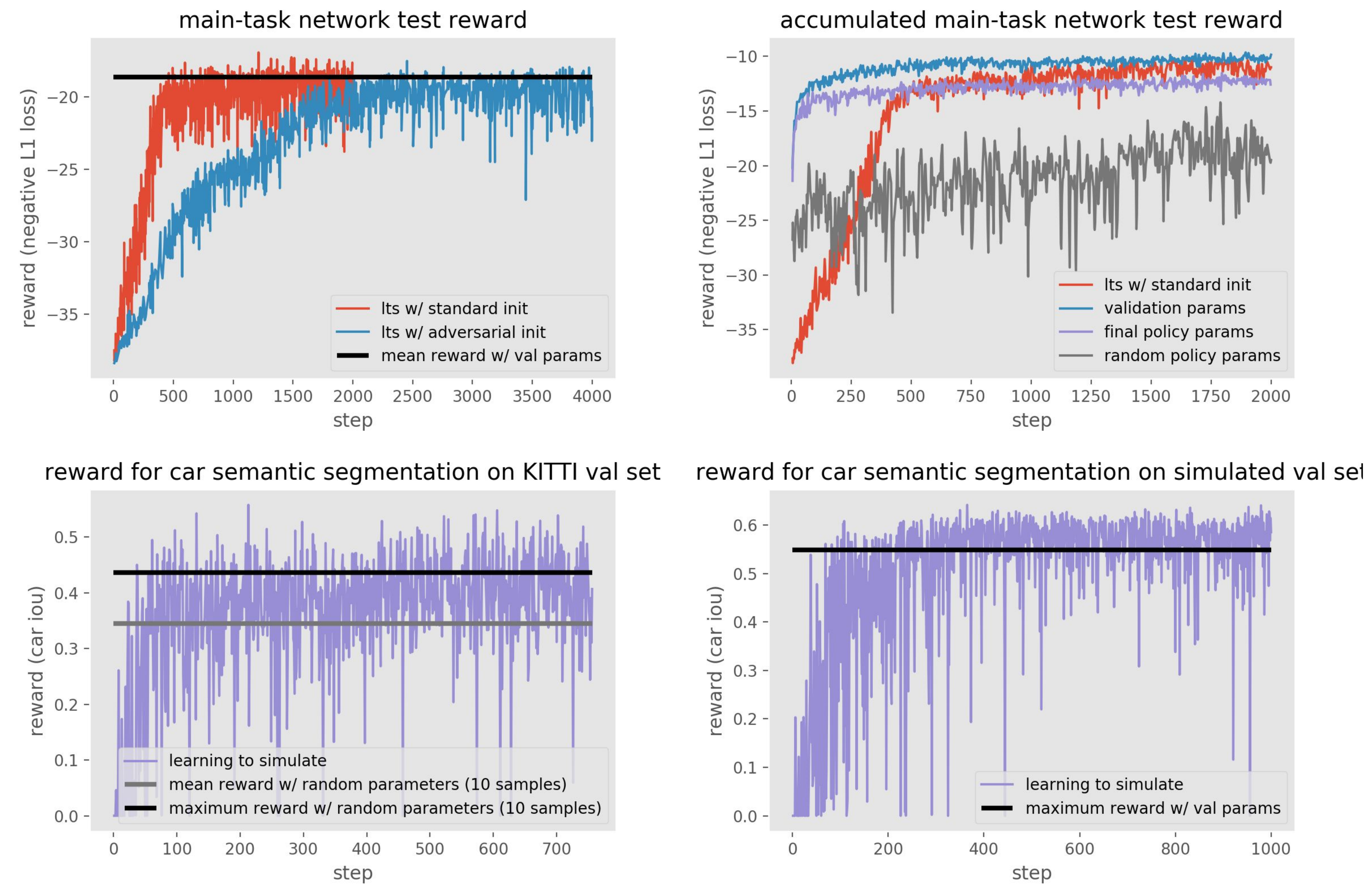


Figure 2: **Top row:** The decision boundaries (shaded areas) of a non-linear SVM trained on data generated by $q(\mathbf{x}, \mathbf{y}|\,\psi_i)$ for three different iterations i of our policy π_ω. The data points overlaid are the test set. **Bottom row:** Decision boundary when trained on data sampled from $p(\mathbf{x}, \mathbf{y}|\,\psi_{\text{real}})$ (left) and on the converged parameters ψ^* (middle); Data sampled from $q(\mathbf{x}, \mathbf{y}|\,\psi^*)$ (right).



➢ We work on two computer vision tasks using our traffic scenes simulator: the **car counting task** and **semantic segmentation.**

➢ For the **car counting task** we train a convolutional neural network to count all instances individually for five different types of cars in an image.

➢ We observe that we learn how to simulate datasets which achieve lower error than the mean error obtained using the validation set parameters, independent of the simulation parameter initialization.

➢ Our method approximates the upper bound set by generating data using the validation set parameters and also outperforms random parameters by a large margin.

➢ For **semantic segmentation**, our method outperforms random policy parameters on **real data** (both on the **KITTI validation set** and on the **KITTI test set**). Moreover it outperforms the validation parameters on a simulated dataset.

Generative Parameters	Car IoU
random params	0.260 ± 0.037
learned params	$\mathbf{0.334 \pm 0.019}$

Table 1: Mean value of Car IoU on the KITTI test set for models $h_{\boldsymbol{\theta}}$ trained from synthetic data generated by random or learned parameters.

References

[1] Dosovitskiy et al. CARLA: An Open Urban Driving Simulator. CORL 2017.
[2] Gaidon et al. Virtual worlds as proxy for multi-object tracking analysis. CVPR 2016.
[3] Richter et al. Playing for data: Ground truth from computer games. ECCV 2016.
[4] Tremblay, et al. Training Deep Networks With Synthetic Data: Bridging the Reality Gap by Domain Randomization. CVPR Workshop 2018