Learning To Simulate

BU

NEC Laboratories America
Relentless passion for innovation

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

[1]Boston University [2]NEC Laboratories America

Motivation

- ❑ Simulating data can be beneficial when data is scarce or annotation is costly [1,2].
- ❑ Previous work simulates large quantities of random scenes, e.g., [3,4].
- ❑ Can we **automatically learn to simulate better data** for a particular task?
- ❑ Can we find a better **trade-off between diversity and volume** of the data?
- ❑ Are the true data generating parameters the **best for training prediction models**?
- ❑ We explore these questions with a **reinforcement-learning based approach to automatically adjust simulation parameters**

Our Simulator

Synthetic images generated by our **parameterized traffic scene simulator.**

We simulate:
- Road and intersection
- Houses on the side
- Cars
- Weather

Heavily modified version of the CARLA [1] plugin in Unreal Engine 4.

Method

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

Loss of main task model trained in simulation and evaluated on real data

Simulation parameters

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

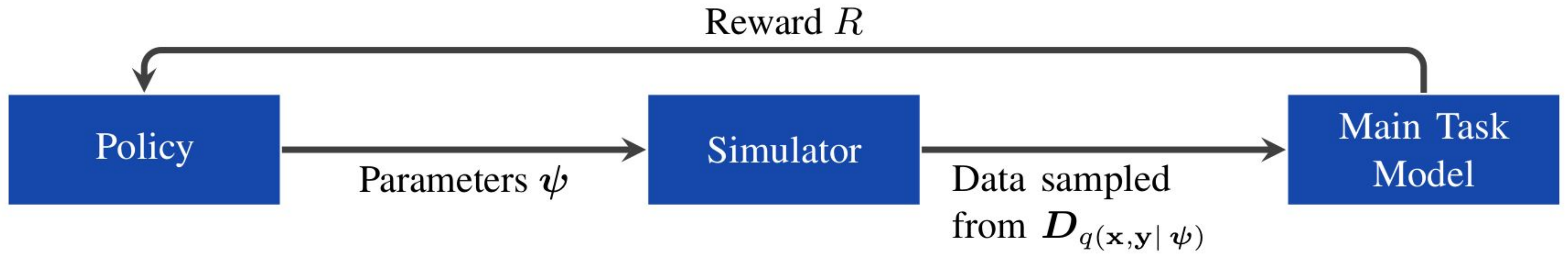
$$\text{s.t.} \quad \boxed{\boldsymbol{\theta}^*(\boldsymbol{\psi})} = \arg\min_{\boldsymbol{\theta}} \sum_{(\boldsymbol{x},\boldsymbol{y})\in \boxed{\boldsymbol{D}_{q(\mathbf{x},\mathbf{y}|\,\boldsymbol{\psi})}}} \mathcal{L}\left(\boldsymbol{y}, h_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{\theta})\right),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

main task model $h_{\boldsymbol{\theta}}$ trained on simulated data

Main task model parameters

Dataset generated by simulator

➢ **Approximation via policy gradients**



Reward R

Policy

Parameters $\boldsymbol{\psi}$

Simulator

Data sampled from $\boldsymbol{D}_{q(\mathbf{x},\mathbf{y}|\,\boldsymbol{\psi})}$

Main Task Model

➢ Simulator and sampling are non-differentiable

➢ Resort to reinforcement learning

➢ Vanilla policy gradients for optimization

```
for iteration=1,2,... do
    Use policy $\pi_\omega$ to generate $K$ model parameters $\boldsymbol{\psi}_k$
    Generate $K$ datasets $\boldsymbol{D}_{q(\mathbf{x},\mathbf{y}|\,\boldsymbol{\psi}_k)}$ of size $M$ each
    Train or fine-tune $K$ main task models (MTM) for $\xi$ epochs on data provided by $\mathcal{M}_k$
    Obtain rewards $R(\boldsymbol{\psi}_k)$, i.e., the accuracy of the trained MTMs on the validation set
    Compute the advantage estimate $\hat{A}_k = R(\boldsymbol{\psi}_k) - b$
    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.

Results

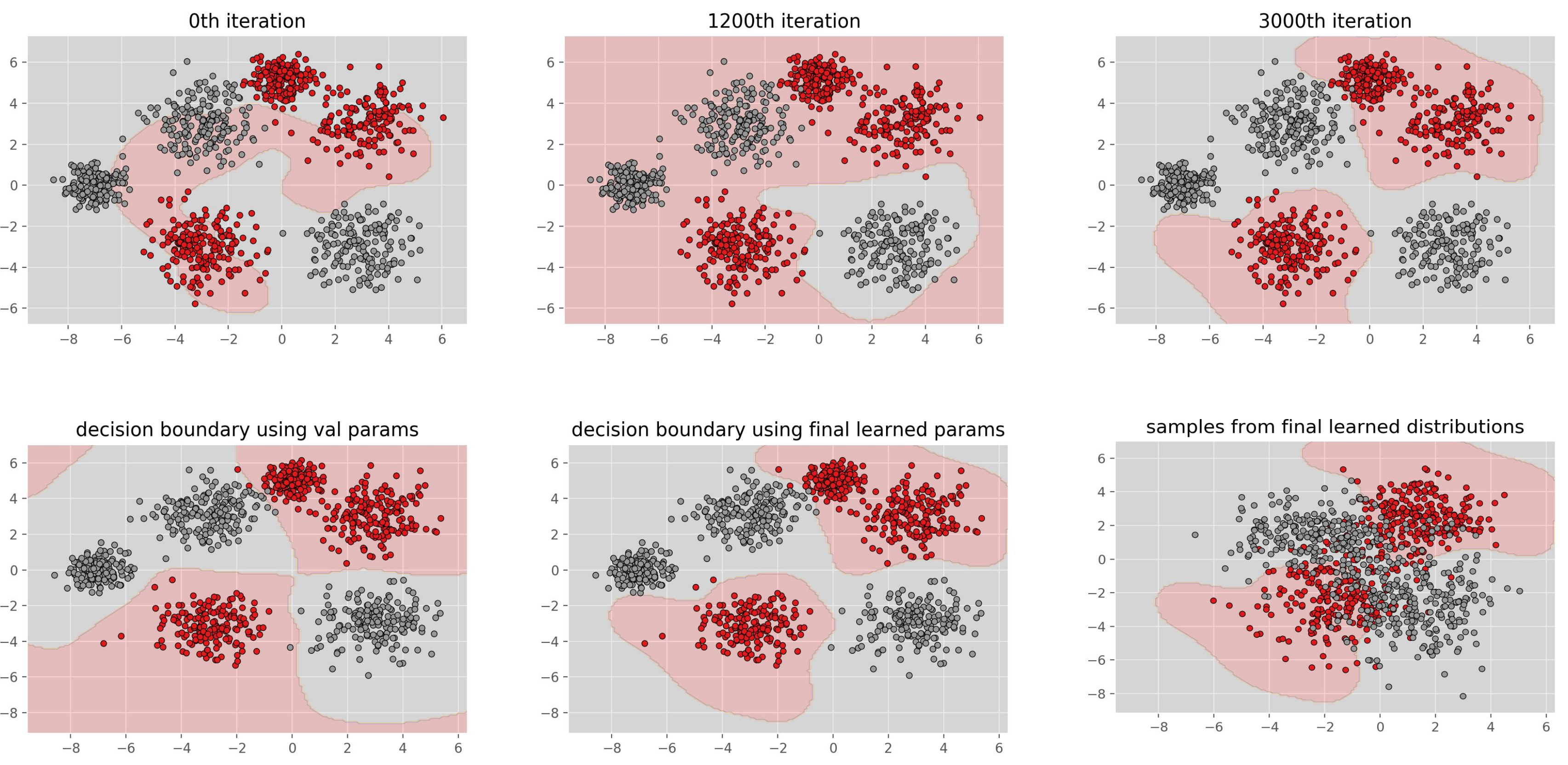


Figure 2: **Top row:** The decision boundaries (shaded areas) of a non-linear SVM trained on data generated by $q(\mathbf{x}, \mathbf{y}|\,\boldsymbol{\psi}_i)$ for three different iterations i of our policy π_ω. The data points overlaid are the test set. **Bottom row:** Decision boundary when trained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

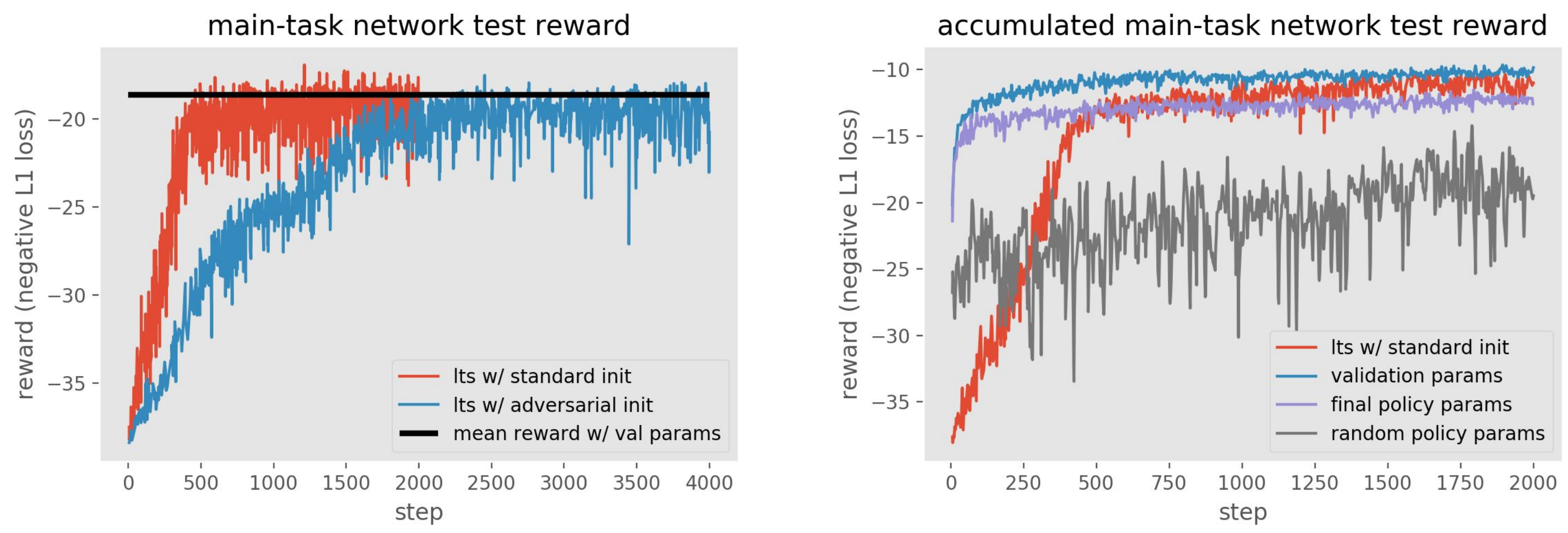
Toy experiment to illustrate the concept:

➢ Two GMMs with three components each

➢ RBF-SVM as main task model to separate the two classes

➢ Simulator: two GMMs with only two components each

➢ Our approach is still able get accurate decision boundaries



Car-counting on simulated data:

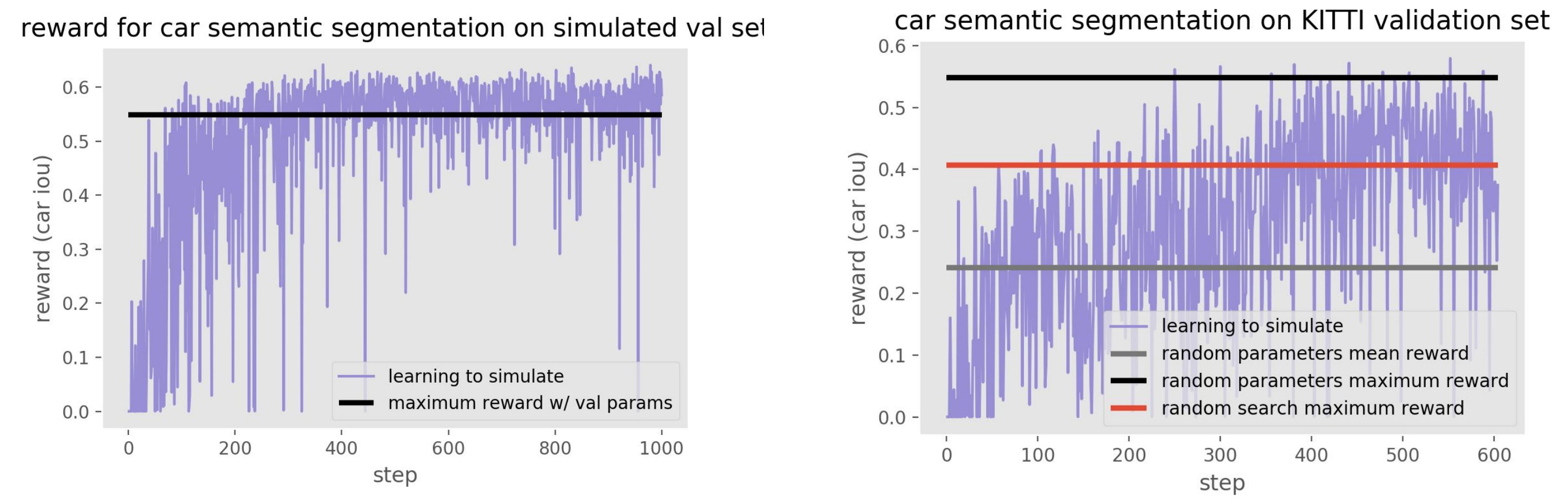
➢ Ground truth data-generating parameters are given.

➢ Train CNN to count all instances of 5 different types of cars

➢ Model achieves lower error on the unseen test set than the mean error obtained using the ground truth simulation parameters

➢ Robust to simulation parameter initialization

➢ Model approximates "upper bound" and outperforms random parameters



Training data	random params	random search	LTS	KITTI train set
Car IoU	0.480	0.407	**0.579**	**0.778**

Table 1: Segmentation Car IoU on the unseen KITTI test set for a ResNet-50 segmentation network trained using synthetic data generated by random parameters or learned parameters using random search or learning to simulate (LTS) for 600 epochs of each. We test the epoch with highest validation reward on the KITTI test set. We also report the maximum car IoU obtained by training on 982 annotated real KITTI training images.

Semantic segmentation on real data:

➢ Experiments on KITTI

➢ Model outperforms random policy parameters and random search on real data

➢ Model outperforms validation set parameters on simulated data

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-W 2018