TuneNet: One-Shot Residual Tuning for System Identification and Sim-to-Real Robot Task Planning

Author information omitted for review

I. Introduction

Recent research has validated simulators as a learning testbed for real-world robot behaviors and control policies. Several studies have shown the ability to adapt simulation-learned policies to the real world by performing additional learning based on observations from the real world [1, 2, 3, 4, 5, 6, 7], but these techniques require extensive data collection. Even before collecting data in the real world, however, substantial improvements in accuracy can be obtained by selecting simulator parameter values that cause the simulator to behave realistically. This is often done using gradient-free optimization techniques [2, 8, 9, 10] which require evaluating many simulation samples. Other approaches train over a large set of possible parameter values to create a robust policy [11, 12, 13, 14, 2], but we are interested in the problem of creating a single "canonical" simulation that behaves as realistically as possible, which can then be used for robust physics prediction or faster policy learning.

Taking cues from recent work in *real-to-sim* transfer [15, 16] we propose TuneNet, a model that tunes one physical model to approximate another by using a neural *residual tuning* approach, estimating the difference in physical parameters between the two models. Because TuneNet is pretrained over simulated data, it is much faster than standard gradient-free optimization techniques. At the same time, estimating parameter differences (rather than raw values) allows it to generalize well and iteratively tune parameters based on a single observation. The resulting tuned simulator can be used for a variety of tasks, such as physics prediction and robot skill learning. We perform experiments to validate that TuneNet can tune simulations to match real-world environments, even when the physical parameters are outside the range seen in training. We show that the tuned simulations can be used for sim-to-real task learning by teaching a robot to complete a task that requires accurate dynamics prediction.

II. Residual Tuning with TuneNet

Our *residual tuning* approach estimates the *difference* in parameters, $\Delta\zeta = \zeta_P - \zeta_T$, between a single proposed model $f_{\zeta_P}(s_t, a_t) = s_{t+1}$, and a single target model, $g_{\zeta_T}(s_t, a_t) = a_{t+1}$, where ζ_P and ζ_T are physics parameters of those models.

By learning to estimate $\Delta\zeta$ instead of ζ, residual tuning naturally becomes better at estimating small deltas, because the difference, $\Delta\zeta$ between two variables uniformly distributed over $[d, u]$ is distributed according to $P(\Delta\zeta) = -\frac{2}{(u-d)^2}x + \frac{2}{u-d}$, making small delta values more common in the training data. As a parameter estimate becomes closer and closer to the true value, the network is able to leverage more and more training data, allowing the estimate to iteratively improve by resampling the proposed model using the current best parameter estimate.

Algorithm 1 TuneNet's parameter tuning procedure.

procedure TUNE(o_T, simulator f_ζ, initial guess ζ_{P_0}, Network h_θ, K)
  for $k = 1 \dots K$ **do**
    $o_P \leftarrow z_P(f_{\zeta_{P_{k-1}}}(s_t, a_t)), t = 1 \dots L$
    $o_T \leftarrow \text{RESAMPLE}(o_T, L)$
    $\zeta_{P_k} \leftarrow \zeta_{P_{k-1}} + h_\theta(o_P, o_T)$
  return ζ_{P_K}

Our model, TuneNet, estimates the parameter residual by taking observations o_P and o_T generated from each model and approximating a function h, which is parameterized by some θ (in this work, the weights of a neural network): $h_\theta(o_P, o_T) = \Delta\zeta \approx \zeta'_T - \zeta_P$.

Our final optimization target for training (over a dataset of length n) is given in Eq. (1), where λ is a weight regularization constant.

$$\arg\min_\theta \sum_{n=1}^{N} \left\| (\zeta_{P_n} + h_\theta(o_{P_n}, o_{T_n})) - \zeta'_{T_n} \right\| + \lambda \|\theta\|^2 \quad (1)$$

TuneNet iteratively updates the proposed model parameters accordingly to match those of the target model (Algorithm 1). Importantly, it is able to do this without collecting any additional observations from the target model, and only sampling the proposed model once per iteration. Iterative updates also allow TuneNet converge to parameter values that lie outside the training range without rescaling the network output. After tuning, the tuned simulator f_{ζ_P} is ready to be used for other tasks such as object motion prediction and robot task learning.

This iterative optimization method can be thought of as *gradient descent in parameter space*, using a neural network to approximate the gradient. In this work we show that TuneNet performs well without even with simple gradient descent updates.

III. Experiments

We implement TuneNet in PyTorch. Our network consists of one independent fully connected layer for each observation, the outputs of which are concatenated and passed to additional connected layers which estimate the parameter residual. We found that concatenating the two intermediate outputs

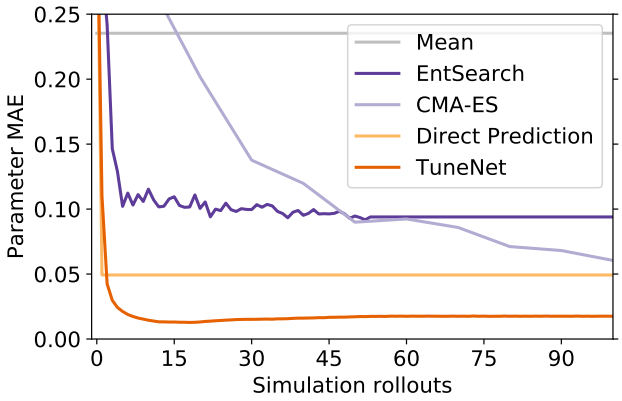

Figure 1: Tuning performance compared to number of simulation samples utilized for the simulated ball-bouncing test set for various estimation techniques.

performed better than other combination methods. In this work, we use a PyBullet physics simulations for the proposed dynamics model f_ζ, and the target model g_ζ is either another simulator or the real world.

To test TuneNet's ability to tune parameters, we used it to tune the coefficient of restitution (COR) of a simulated bouncing ball. Our proposed and target dynamics models were simulations in which balls were dropped having randomly sampled drop heights and COR values. The observation from each model is the height of the ball at each simulation step. TuneNet is trained using pairs of simulations with COR values sampled from the range $[0.3, 0.7]$, and tested on pairs of simulations where the *target* COR is sampled from the range $[0, 1]$. This tests both the residual tuning approach and the ability to tune to values outside the training range.

As seen in Fig. 1, TuneNet outperforms *CMA-ES* and Greedy Entropy Search (*EntSearch*), two gradient-free optimization techniques, and achieves extremely low error over the test set in just a few iterations. The algorithm's ability to iteratively tune parameters also allows it to outperform a neural network that is directly trained to predict parameter values in one step.

We also validated TuneNet for sim-to-real learning by estimating the COR of a bouncing ball in the real world and learning to complete a "bounce shot" off an inclined plane and into a hoop. The task is difficult to complete for the first time using a previously unseen ball, even for a human. For this task, we trained TuneNet over a modified version of the dataset in the previous experiment, where the input from the target model was replaced with visual tracking results from rendered simulation videos. The input to TuneNet at each iteration was one ground truth state and one visual tracking result. At test time, our robot picked up a ball and dropped it once on a flat plane, recorded a video of the ball being dropped, and used the object tracking results from the video to tune a simulator to match the real-world COR. This COR was then used in a second simulation to determine the correct drop height to use for the robot to complete the bounce shot. After discounting

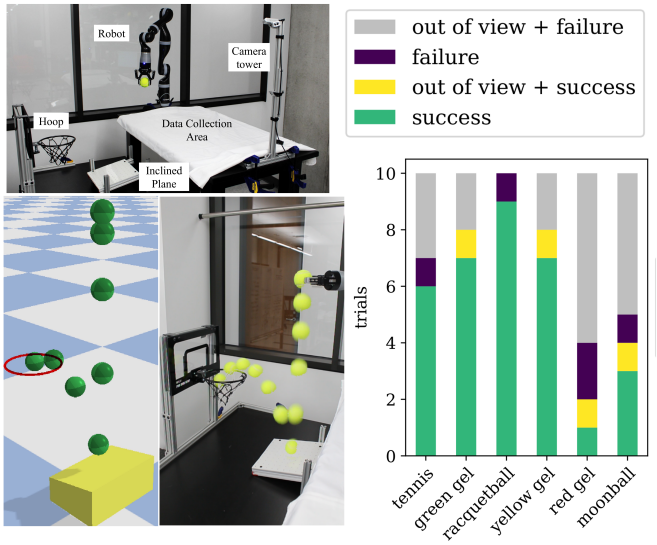

Figure 2: Making a bounce shot in the tuned simulator and real world (left). Right: task success rates for various ball types.

error cases where the ball left the field of view during the data collection step, our system was 87% successful at completing the bounce shot with six different types of balls (see Fig. 2).

IV. Discussion and Future Work

Residual tuning represents a new method for closing the reality gap, and demonstrates that by learning to estimate the differences in parameters and training on artificial datasets, we can produce more accurate simulators from a single real-world example and minimal simulation sampling. Our TuneNet model trains quickly (7 minutes on an NVIDIA GTX 1080Ti), and also estimates parameters more quickly than existing gradient-free optimization techniques or newer tuning techniques that alternate between evaluation in simulation and the real world [8, 2].

We plan to continue to evaluate TuneNet's ability to rapidly tune simulators, including comparisons in other simulated environments and tuning multiple parameters at once. We also are developing comparisons with state-of-the-art object prediction approaches to further validate the accuracy of our approach. Ultimately, we plan to develop models for predicting outcomes and learning to manipulate objects without requiring extensive real-world practice.

References

[1] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual Reinforcement Learning for Robot Control," *CoRR*, vol. abs/1812.0, 2018.

[2] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience," *CoRR*, 10 2018.

[3] J. P. Hanna and P. Stone, "Grounded Action Transformation for Robot Learning in Simulation," in *Proceedings*

of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17), no. February, 2 2017, pp. 3834–3840.

[4] F. Golemo, A. A. Taïga, P.-Y. Oudeyer, and A. Courville, "Sim-to-Real Transfer with Neural-Augmented Robot Simulation," in *2nd Conference on Robot Learning (CoRL18)*, ser. Proceedings of Machine Learning Research, vol. 87. PMLR, 2018, pp. 817–828.

[5] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. A. Funkhouser, "TossingBot: Learning to Throw Arbitrary Objects with Residual Physics," *CoRR*, vol. abs/1903.1, 2019.

[6] T. Silver, K. Allen, J. Tenenbaum, and L. P. Kaelbling, "Residual Policy Learning," *CoRR*, vol. abs/1812.0, 2018.

[7] A. A. Rusu, M. Večerík, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-Real Robot Learning from Pixels with Progressive Nets," in *Conference on Robot Learning*, 2017, pp. 262–270.

[8] S. Zhu, A. Kimmel, K. E. Bekris, and A. Boularias, "Fast Model Identification via Physics Engines for Data-efficient Policy Search," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, ser. IJCAI'18. AAAI Press, 2018, pp. 3249–3256.

[9] J. Tan, Z. Xie, B. Boots, and C. K. Liu, "Simulation-based design of dynamic controllers for humanoid balancing," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2016-Novem. IEEE, 10 2016, pp. 2729–2736.

[10] S. Zhu, D. Surovik, K. E. Bekris, and A. Boularias, "Closing the Reality Gap of Robotic Simulators through Task-oriented Bayesian Optimization," *Journal of Machine Learning Research*, 2019.

[11] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2017-Septe. IEEE, 9 2017, pp. 23–30.

[12] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-Real: Learning Agile Locomotion For Quadruped Robots," in *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation, 6 2018.

[13] A. Rajeswaran, S. Ghotra, B. Ravindran, and S. Levine, "EPOpt: Learning Robust Neural Network Policies Using Model Ensembles," *CoRR*, 2016.

[14] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric Actor Critic for Image-Based Robot Learning," in *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation, 6 2017.

[15] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis, "Sim-to-Real via Sim-to-Sim: Data-efficient Robotic Grasping via Randomized-to-Canonical Adaptation Networks," in *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[16] J. Zhang, L. Tai, P. Yun, Y. Xiong, M. Liu, J. Boedecker, and W. Burgard, "Vr-goggles for robots: Real-to-sim domain adaptation for visual control," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1148–1155, 2019.