IMPROVING PERFORMANCE ON THE MANISKILL CHALLENGE VIA SUPER-CONVERGENCE AND MULTI-TASK LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

We present key aspects of our approach to the ManiSkill Challenge, where we used Imitation Learning on the provided demonstration dataset to let a robot learn how to manipulate interactive objects. We present what is to our knowledge the first application of super-convergence via learning rate scheduling to Imitation Learning and robotics, enabling better policy performance with a training time reduced by almost an order of magnitude. We also present how we used Multi-task Learning to reach a top score on unseen object of 1 task of the challenge. It shows that the strategy can unlock generalization performance on some tasks, corroborating other work in the field. We also show that simple data augmentation strategies can help push the model performance further.

1 OVERVIEW

The ManiSkill Challenge (Mu et al., 2021) consists of object manipulation tasks to be performed by a mobile robot arm in the SAPIEN simulation environment (Xiang et al., 2020). 4 tasks are available in the challenge: OpenCabinetDrawer, OpenCabinetDoor, PushChair and MoveBucket. For each task, a given robot must interact with an object of a specific class (e.g. a cabinet) to make it reach a desired state (e.g. the upper drawer is opened). In our work, we focus on the two first tasks, where a single robotic arm mounted on a mobile base must open either a drawer or a door of a cabinet.

We worked on the *No Interaction* track of the challenge, in which the agent policy is to be trained via offline Imitation Learning (Hussein et al., 2017), relying only on a provided dataset of successful demonstrations, and without additional interaction with the simulated environment. The dataset (containing 300 trajectories per environment) provides sequences of robot joint states and 3D point clouds from the robot's sensors along the execution of each task. A segmentation of the point cloud, provided as input, specifies the target door/drawer to be opened. The resulting policy is evaluated by measuring the success rate (ratio of successfully opened doors/drawers) in the simulator on both demonstration cabinet and unseen cabinets.

1.1 CHALLENGES HIGHLIGHTED IN THE CHALLENGE BENCHMARK IMPLEMENTATION

The original study achieves up to 76% success rate on a single environment, and the performance drops to 37% when trained on all the training environments (Mu et al., 2021). This suggests limitations in either the learning capacity of the architecture, or the training process.
Object level generalization appears challenging. Success rates show a significant performance drop on test objects, for example from 30% to 11% on the OpenCabinetDoor task. This suggests opportunities for improvements in terms of generalization.

1.2 BASE ARCHITECTURE & KEY ASPECTS OF SUBMISSION

We use Behavior Cloning (Pomerleau, 1988) as an Imitation Learning strategy, coupled with a PointNet (Qi et al., 2017)+Transformer (Vaswani et al.) architecture, which showed the best generalization performance in the challenge benchmark implementation (Mu et al., 2021). Behavior Cloning enables a model to train under an end-to-end supervised learning regime, from perception to control.

We present below 2 key aspects of our submission: The first is learning rate scheduling for training efficiency. The second is the use of multi-task learning and network modifications for improving generalization.

2 LEARNING SCHEDULE EFFECT ON TRAINING SPEED

The impact of learning schedule is remarkable on policy-based architectures, including policy performance, data requirements and learning speed (Schrittwieser et al., 2020). In this experiment, we try to induce super-convergence (Smith & Topin, 2017), "extremely" fast learning, using a single cycle learning rate schedule [1] coupled with the AdamW optimizer(which has been shown to generally offer better generalization performance than Adam (Loshchilov & Hutter, 2017)).

We train the baseline model on the OpenCabinetDrawer task, using the data split in Table 2, and compare the results of using no scheduler with 150k gradient steps (as in the original implementation, which is roughly equivalent to 12 epochs in this setting) to the OneCycle scheduler on different cycle length. Results are compiled in table 1.

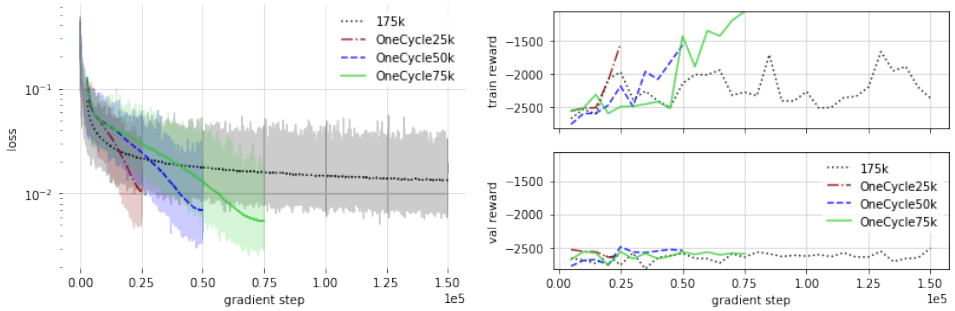

Figure 1: (Left) L2 loss on predicted vs demonstration action vector; (Right) mean train and validation reward, as defined in Mu et al. (2021). Data from 40 trials on OpenCabinetDrawer.

We note that since the learned policy influences the distribution of inputs on which it will be tested, the Behavior Cloning setting violates the i.i.d hypothesis of supervised learning, which means we cannot estimate the policy performance from loss alone (Ross & Bagnell, 2010). Typically, the agent will show an inability to exhibit recovery behaviors which are unlikely to have been encountered during training. Consequently, we track additional performance metrics by performing regular online evaluations during training and measuring reward and success rate provided in the context of the Challenge. We rely on only 40 trials for those evaluations, as they are time-consuming. Reward proves to be the most relevant metric as a proxy for performance during training, being more granular and thus providing more signal than the success rate (which is noisy and may remains at 0 for validation). Our experiments do not show over-fitting in the range of training length tested, however the validation reward does appear to stagnate, as observable on the bottom right graph on figure 1.

Table 1: Average success rate on training and test environments, with 10 trials per environment, for the OpenCabinetDrawer task (34 environments used for training).

Learning mode	Adam 150k	OneCycle 25k	OneCycle 50k	OneCycle 75k
training	0.26 ± 0.21	0.47 ± 0.25	0.62 ± 0.18	$\mathbf{0.64} \pm 0.20$
validation	0.10 ± 0.13	0.14 ± 0.15	0.16 ± 0.17	$\mathbf{0.19} \pm 0.15$

In the final evaluations (Table 1), we observe diminishing returns with a lot less improvement between 50k and 75k steps, as opposed to the difference between 25k and 50k steps. There may not be margin for much further improvement with longer training in the OneCycle case. Compared to

[1] Using the OneCyleLR implementation in PyTorch

the baseline, the OneCycle schedule with 25k steps yields more than 50% improvement in success rate in 1/6th of the training duration. OneCycle with 75k steps roughly doubles the success rate in half the training duration. We reuse the OneCycle learning rate scheduling with 75k gradient steps in the rest of the paper.

3 MULTI-TASK LEARNING

We next investigate the impact of Multi-task Learning by training a single model on the (OpenCabinetDoor, OpenCabinetDrawer) pair of tasks. We expect that doing so will allow the single model to benefit from more training data and will improve generalization by learning higher level representations (Caruana, 1997).
The action domain space being the same for the 2 tasks, no adaptation is required on the output side of the model. The model input for the 2 tasks is also directly compatible, with semantically equivalent segmentation (target part being a Door/Drawer depending on the task) and does not require any change. Moreover, since the 2 tasks deal with the same class of objects (cabinets), we expect the learned internal representations to be highly compatible.

3.1 TASK ENCODING

While we could let the model infer the task from input only, we choose to help the model by adding an additional input to do task conditioning, similarly to related work (Ebert et al., 2021), especially as some doors could be confused as drawers. In practice this is done by appending a one hot encoded identifier to the state vector. In our experiments the identifier $\boldsymbol{t}$ can take only 2 vector values: $\boldsymbol{t} = [0, 1]$ or $\boldsymbol{t} = [1, 0]$. The growth of model parameters due to the additional input is negligible compared to the overall model complexity (see details A.2).

3.2 DATA AND EXPERIMENTAL SETUP

The experimental setups splits the environments as listed in table 2, so that both single task and multi-task models get the same amount of input data, and get evaluated on the same validation environments. This setup guarantees a fair comparison between single and multi-task approaches.

Table 2: Number of demonstration environments for each task.

split	Training1 (single and multi task)	Training2 (single task only)	Evaluation	Total
OpenCabinetDrawer	17	17	8	42
OpenCabinetDoor	17	17	22	56

Cabinets having demonstration data available for both tasks are included in the *Training1* split, used for multi-task learning. This prevents data leak when evaluating the performance of the multi-task solution (making sure that the cabinet are unseen through the other task). We also expect that the use of common cabinets between tasks will improve generalization of the multi-task model by preventing it learning of associations between cabinet model and task.

3.3 ON THE FLY DATA AUGMENTATION

As data augmentation usually contributes to reducing risks of over-fitting and improving generalization, we introduce "on the fly" point cloud randomization during training on input point clouds, similarly to related work Chen et al. (2020). Jittering is applied on XYZ coordinates for each point individually, constrained to modify the coordinates within the estimated precision requirements for gripping tasks. Since object colors should have only limited impact for scene understanding, the input point clouds being pre-segmented, we similarly apply random color jittering and RGB channel swapping (details in A.4).

3.4 MODEL CAPACITY INCREASE

To evaluate how the multi-task model (with augmentation) scales, we then increase the number of environments used for training to the the 68 training environments of OpenCabinetDoor and OpenCabinetDrawer combined. Since we notice that the performance improves only marginally on the OpenCabinetDrawer (see Table 1), we choose a larger model to evaluate whether extra capacity can address this limitation. The new model, labelled *Multi-task++* in Table 1, keeps the same architecture, with increased layer sizes, number of attention heads and embedding dimension (see A.2.1).

3.5 ANALYSIS

A summary of the results is presented in Table 3. Generalization on unseen environments is highly variable, as seen in the high standard deviation of the results across environments (additional details in A.5). It could be interesting to further analyse which properties of the cabinets influence the generalisation performance (e.g. apperance, mechanical properties). Moreover, while we manage to obtain satisfying generalization results on the OpenCabinetDrawer task, the OpenCabinetDoor task does not result in good generalization. We think that this may be due the more complex coordination required to pivot an object vs pulling it. The various handle to hinge distance of doors may increase the variety of trajectories required, and more data may help resolving the generalization gap.

Table 3: Mean and standard deviation of average success rates on training and validation environments of each task over 10 different runs.

Task	OpenCabinetDrawer		OpenCabinetDoor	
Split	Training	Validation	Training	Validation
Drawer Only (17+17)	**0.64** $\pm$ 0.20	**0.19** $\pm$ 0.15	N/A	N/A
Door Only(17+17)	N/A	N/A	**0.55** $\pm$ 0.13	0.03 $\pm$ 0.08
Multi-tasks w/o augm	0.63 $\pm$ 0.19	0.13 $\pm$ 0.14	0.52 $\pm$ 0.25	**0.05** $\pm$ 0.21
Multi-tasks w/ augm	**0.64** $\pm$ 0.13	**0.19** $\pm$ 0.19	0.54 $\pm$ 0.19	**0.05** $\pm$ 0.14
Multi-tasks w/ augm (68)	0.65 $\pm$ 0.17	0.21 $\pm$ 0.17	0.54 $\pm$ 0.26	**0.05** $\pm$ 0.15
Multi-tasks ++ (68)	**0.67** $\pm$ 0.17	**0.33** $\pm$ 0.33	**0.58** $\pm$ 0.18	0.04 $\pm$ 0.15

While the multi-task model trained on 34 environments does not generally outperform the single-task model, it also does not exhibit significant performance drop, except for validation on OpenCabinetDrawer. Once randomization is added, we observe a small overall performance improvement on both training and validation, obtaining the best results of all models, with the exception of the OpenCabinetDoor validation. Additional testing would be required to show the significance of the result. We think that the diversity of cabinet and environments provides enough input variability, and data augmentation provides only marginal additional diversity.

Finally, we observe that the larger model does yield better overall results, in particular a significant jump in generalization on the Drawer validation.

4 CONCLUSION & FURTHER EXPLORATION

Regarding learning optimization, we showed that OneCycle learning rate scheduling can dramatically decrease training duration. We think exploring further learning strategies, like Curriculum Learning (Bengio et al., 2009), can be very valuable. This would mean organising the presentation of new object instances and tasks to the model in a way that optimizes learning.

Having noticed potential limitations in the model, as well as the stability issue inherent to naive behavior cloning, we think it would be useful to enforce more temporal coherence. This could be done by replacing PointNet with a model capable of integrating a sensory sequence like Fan et al. (2021), or by learning to predict multiple actions ahead of time (Jang et al., 2021).

Regarding the multi-task approach, we think it would be valuable to explore multi-task learning on the (PushChair, MoveBucket) pair of tasks, which are similar in nature, and also investigate how to apply multi-task learning when the body is different (different number of arms), by learning on the 4 tasks of the challenge.

REFERENCES

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, pp. 1–8, Montreal, Quebec, Canada, 2009. ACM Press. ISBN 978-1-60558-516-1. doi: 10.1145/1553374.1553380.

Rich Caruana. Multitask Learning. *Machine Learning*, 28(1):41–75, July 1997. ISSN 1573-0565. doi: 10.1023/A:1007379606734.

Yunlu Chen, Vincent Tao Hu, Efstratios Gavves, Thomas Mensink, Pascal Mettes, Pengwan Yang, and Cees G. M. Snoek. PointMixup: Augmentation for Point Clouds. August 2020.

Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge Data: Boosting Generalization of Robotic Skills with Cross-Domain Datasets. *arXiv:2109.13396 [cs]*, September 2021.

Hehe Fan, Xin Yu, Yuhang Ding, Yi Yang, and Mohan Kankanhalli. PSTNET: Pont Spatio-Temporal Convolution On Point Cloud Sequences. pp. 23, 2021.

Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation Learning: A Survey of Learning Methods. *ACM Computing Surveys*, 50(2):21:1–21:35, April 2017. ISSN 0360-0300. doi: 10.1145/3054912.

Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. BC-Z: Zero-Shot Task Generalization with Robotic Imitation Learning. *5th Annual Conference on Robot Learning*, pp. 12, 2021.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. November 2017.

Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. ManiSkill: Generalizable Manipulation Skill Benchmark with Large-Scale Demonstrations. *arXiv:2107.14483 [cs]*, November 2021.

Dean A. Pomerleau. ALVINN: An autonomous land vehicle in a neural network. In *Proceedings of the 1st International Conference on Neural Information Processing Systems*, NIPS'88, pp. 305–313, Cambridge, MA, USA, January 1988. MIT Press.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *arXiv:1612.00593 [cs]*, April 2017.

Stephane Ross and J Andrew Bagnell. Efficient Reductions for Imitation Learning. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 8, 2010.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. *Nature*, 588(7839):604–609, December 2020. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-020-03051-4.

Leslie N. Smith and Nicholay Topin. Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates. *arXiv:1708.07120 [cs, stat]*, August 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. pp. 11.

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A SimulAted Part-Based Interactive ENvironment. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11094–11104, Seattle, WA, USA, June 2020. IEEE. ISBN 978-1-72817-168-5. doi: 10.1109/CVPR42600.2020.01111.

A APPENDIX

A.1 DATA

The demonstration dataset contains:

- OpenCabinetDoor: 42 unique cabinets (56 cabinet/door pairs)
- OpenCabinetDrawer: 25 cabinets (42 cabinet/drawer pairs)

10 cabinets are present in demonstrations for both OpenCabinetDoor and OpenCabinetDrawer. 300 demonstrations are provided for each environment, an environment being a Cabinet/Door or Cabinet/Drawer pair (Cabinets can have multiple doors and drawers as seen in the figure below).

Figure 2: Robot arm attempting to open a drawer.

A.2 MODEL ARCHITECTURE

We reuse the architecture from Mu et al. (2021), as shown in figure A.2, where our modifications are highlighted.

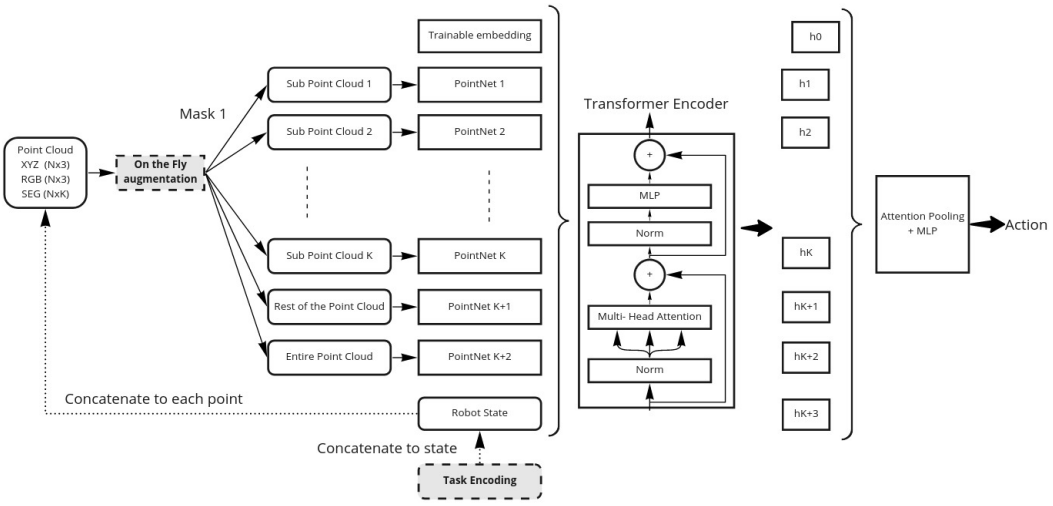

Figure 3: Architecture diagram.

A.2.1 LARGE MODEL

For the large model (*MultiTasks ++*), we switch from 4 to 8 attention heads, and double the size of all the non-constrained layers, resulting in an increase parameter count as seen in Table 4.

A.3 OBSERVED TRAINING DURATIONS

Training duration for the experiments on a Ryzen3700X with a RTX2060Super GPU are shown below.

Table 4: Parameter count in each model

model	parameter count	relative to baseline
baseline	5,234,855	-
MultiTask	5,238,439	+0.07%
MultiTask++	15,115,303	+189%

Table 5: Training Duration for each learning setting, including evaluation times. The evaluation times tends to be shorter on success.

Learning mode	Adam 150k	OneCycle 25k	OneCycle 50k	OneCycle 75k
duration	6h47m	1h8m	2h16m	3h24m

A.4 Details of on the fly data augmentation

We perform 3 on the fly input randomization:

- RGB color channel swapping, applied to the whole point cloud (In order to preserve global coherence)
- color jittering with uniformly distributed noise in the range +/- 0.025 of the RGB channels unit scale, applied to the whole point cloud (In order to preserve global coherence)
- XYZ jittering with uniformly distributed noise in the range +/- 1mm, applied to each point individually.

A.5 Per cabinet model results

The tables 6 and 7 show the individual success rates for 10 trials on each of the environments.

While most of the validation environments don't yield any successes, some manage to be repeatedly successful.

Table 6: Results on validation environment for OpenCabinetDrawer

env	success
OpenCabinetDrawer_1013_link_0-v0	0.0
OpenCabinetDrawer_1040_link_0-v0	0.6
OpenCabinetDrawer_1032_link_0-v0	0.2
OpenCabinetDrawer_1066_link_0-v0	0.0
OpenCabinetDrawer_1033_link_2-v0	0.0
OpenCabinetDrawer_1005_link_3-v0	0.8
OpenCabinetDrawer_1021_link_0-v0	0.3
OpenCabinetDrawer_1082_link_2-v0	0.7

Table 7: Results on training environment for OpenCabinetDrawer

env	success
OpenCabinetDrawer_1045_link_0-v0	0.6
OpenCabinetDrawer_1067_link_0-v0	0.9
OpenCabinetDrawer_1000_link_0-v0	0.9
OpenCabinetDrawer_1052_link_0-v0	0.8
OpenCabinetDrawer_1027_link_0-v0	0.6
OpenCabinetDrawer_1044_link_1-v0	0.7
OpenCabinetDrawer_1016_link_1-v0	0.5
OpenCabinetDrawer_1054_link_0-v0	0.7
OpenCabinetDrawer_1061_link_0-v0	0.6
OpenCabinetDrawer_1044_link_0-v0	0.2
OpenCabinetDrawer_1016_link_0-v0	0.7
OpenCabinetDrawer_1000_link_1-v0	0.7
OpenCabinetDrawer_1016_link_3-v0	0.7
OpenCabinetDrawer_1063_link_0-v0	0.7
OpenCabinetDrawer_1016_link_2-v0	0.6
OpenCabinetDrawer_1027_link_1-v0	0.6
OpenCabinetDrawer_1038_link_0-v0	0.9