Effect of Optimizer, Initializer, and Architecture of Hypernetworks on Continual Learning from Demonstration

Sayantan Auddy[1], Sebastian Bergner[1], and Justus Piater[1,2]

[1] Department of Computer Science, University of Innsbruck, Innsbruck, Austria
{sayantan.auddy, justus.piater}@uibk.ac.at
sebastian.bergner@student.uibk.ac.at
[2] Digital Science Center, University of Innsbruck, Innsbruck, Austria

Abstract. In *continual learning from demonstration* (CLfD), a robot learns a sequence of real-world motion skills *continually* from human demonstrations. Recently, hypernetworks have been successful in solving this problem. In this paper, we perform an exploratory study of the effects of different optimizers, initializers, and network architectures on the continual learning performance of hypernetworks for CLfD. Our results show that adaptive learning rate optimizers work well, but initializers specially designed for hypernetworks offer no advantages for CLfD. We also show that hypernetworks that are capable of *stable* trajectory predictions are robust to different network architectures. Our open-source code is available at `https://github.com/sebastianbergner/ExploringCLFD`.

Keywords: Learning from Demonstration, Continual Learning

1 Introduction

Learning from Demonstrations (LfD) [1] is an intuitive way for humans to train robots without explicit programming. While the majority of research on LfD addresses single skill acquisition, some recent methods investigate continual learning from demonstration (CLfD) [2, 3], i.e. learning multiple LfD motion skills sequentially in an open-ended way. Auddy et al. [2] propose a system of hypernetwork-generated neural ordinary differential equation solvers (NODEs) for continually learning a sequence of real-world 6-DoF trajectory learning tasks from human demonstrations. More recent work [3], shows that by enforcing stable trajectory predictions through hypernetwork-generated *stable* NODEs [4], the continual learning performance is greatly enhanced in addition to the expected guarantee of non-divergent and safe trajectory predictions. Hypernetworks have also been utilized for continual reinforcement learning with robots [5]. The popularity of hypernetworks for robotic continual learning is mainly due to desirable features such as not having to store and retrain on data of past tasks, negligible parameter growth with additional tasks, and low catastrophic forgetting [6].

In any deep learning system, many decisions related to the architecture and training need to be taken. In previous works on CLfD [2, 3], the effect of different deep learning components on the continual learning performance of hypernetworks remains unexplored. These past works have either followed accepted best practices (e.g. Adam is the optimizer) or followed the defaults from prior work (e.g. hypernetwork architecture). In this paper, we conduct an exploratory study in which we evaluate the effect of three key deep learning factors on the performance of hypernetworks for CLfD: (i) *optimization algorithms*, (ii) *initialization schemes*, and (iii) *hypernetwork and target network architectures*. We adopt the RoboTasks9 dataset of real-world LfD tasks [3] as a benchmark, and train hypernetworks and chunked hypernetworks continually on the 9 tasks of this dataset. Additionally, we evaluate two kinds of target networks (generated by the hypernetworks): NODE [2], and stable NODE (sNODE) [3].

Our results show that adaptive learning rate optimizers exhibit the best empirical performance, but an initializer designed for hypernetworks (Principled Weight Initialization [7]) does not outperform a good default choice (Kaiming [8]) for CLfD. We also show that when stable NODEs (sNODEs) are used as the target network (i.e. the LfD trajectory predictions are non-divergent), the continual learning performance is mostly independent of the network architecture.

2 Background

Continual Learning from Demonstration: Learning from Demonstration (LfD) [1] is a robot training paradigm where a robot learns motion skills from human demonstrations. LfD can be performed via kinesthetic teaching where a human physically guides a robot and shows it how to perform a particular motion task. The trajectories demonstrated by the human are recorded and used to learn a vector field [4] which can then be used by the robot to perform a similar motion as the demonstration. While typical LfD approaches focus on learning a single motion skill, the objective of *continual* LfD is to learn and remember a sequence of different motion skills, one at a time, in an open-ended manner with a single model and without storing training data of past demonstrations. In the past, this has been achieved by generating parameterized dynamical systems called *Neural Ordinary Differential Equation solvers* (NODE) with Hypernetworks [2]. More recently, it has been shown that hypernetwork-generated *stable* NODEs (NODEs augmented with a stabilizing Lyapunov function) [4] produce stable, non-divergent trajectories and are more effective at continually learning sequences of real-world and high-dimensional LfD tasks [3].

Hypernetworks: A hypernetwork is a neural network that generates the parameters of another neural network called the *target network* [9]. A hypernetwork $\mathbf{f}$ with parameters $\mathbf{h}$, takes as input a trainable task embedding vector $\mathbf{e}^m$ and generates the target network parameters $\mathbf{f}(\mathbf{e}^m, \mathbf{h}) = \theta^m$ for the m^{th} task. A two-stage optimization process (see [2, 3, 9]) is employed to optimize $\mathbf{h}$ and the task embedding vector $\mathbf{e}^m$. Once the m^{th} task is learned, the task embedding $\mathbf{e}^m$ is frozen and stored. For learning the $m+1^{\text{th}}$ task, a new task embedding $\mathbf{e}^{m+1}$ is initialized and the same two-step learning process is repeated. A regular hypernetwork generates all the parameters θ^m of the target network from the final layer, which can result in a large parameter size. Alternatively, *chunked* hypernetworks [9] generate the target network parameters in smaller segments called chunks, and consequently have a smaller size. See [2, 3, 9] for details.

3 Experiments and Results

We train hypernetworks (HN) and chunked hypernetworks (CHN), each with either a NODE or an sNODE as the target network, resulting in 4 kinds of hypernetworks (HN→NODE, CHN→NODE, HN→sNODE, CHN→sNODE). We compare the performance of 3 different optimizers: Stochastic Gradient Descent (SGD), RMSProp [10], Adam [11] and 3 different initializers: Kaiming [8], Principled Weight Initialization (PWI) [7], Xavier [12] when used to train the 4 kinds of hypernetworks. We evaluate 16 different architectures for each hypernetwork. Each model is trained continually on the 9 LfD tasks of RoboTasks9 [3]. We report the widely used *Dynamic Time Warping* (DTW) error metric [2, 3]. Due to the large number of possible combinations, we perform our experiments in 3 stages to keep the number of runs manageable. To aid reproducabilty and further research, our code and experiment hyperparameters are available at `https://github.com/sebastianbergner/ExploringCLFD`.

Experiment 1 (Optimizers): We train each of the 4 kinds of hypernetworks with the 3 different optimization algorithms (SGD, RMSProp, Adam). We use a fixed architecture (same as [3]) and initializer (Kaiming) for all hypernetworks. After each task is learned during the continual learning process, we evaluate each model on the currently learned task and all previous tasks and repeat each run 5 times with independent seeds. Fig. 1 (top row) shows the overall DTW errors during this evaluation. For all optimizers, HN→sNODE and CHN→sNODE outperform HN→NODE and CHN→NODE. The overall performance of SGD is much worse than Adam and RMSProp, both of which achieve similarly good DTW errors.

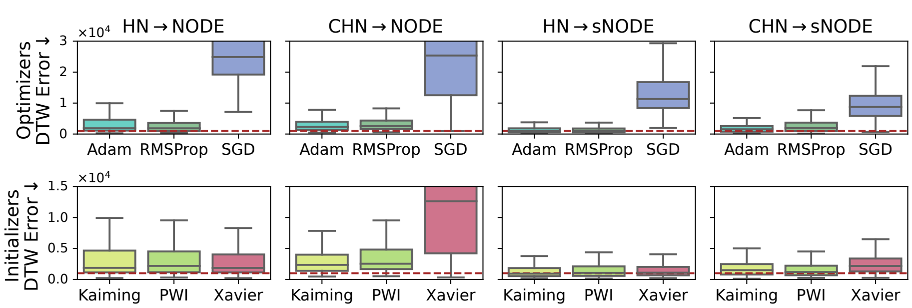

Fig. 1. DTW errors (lower is better) of different *optimizers* (top), and *initializers* (bottom) for 5 independent runs. For reference, the dotted brown line shows the best possible median DTW score from [3] (when each task is learned with a separate model).

Experiment 2 (Initializers): We use Adam as the optimizer since it is slightly better than RMSProp for the well-performing models (with sNODE) in the previous experiment. We compare 3 initializers (Kaiming, PWI, and Xavier) while training the 4 kinds of hypernetworks with fixed architectures (same as [3]). We follow the same training and evaluation steps as in experiment 1 and repeat each run 5 times. We report the DTW errors for these evaluations in Fig. 1 (bottom row). All initializers perform similarly, except for CHN→NODE, where the Xavier initialization fails completely (very high DTW errors). For the other hypernetworks, all the initializers achieve similar results. All our hypernetworks use ReLU activations, and while Xavier is designed for *tanh*/*sigmoid*, we still include it in our experiments since it is used in a similar comparison in a prior work on hypernetwork initialization [7]. Additionally, though PWI is specially designed for hypernetworks, it does not outperform Kaiming for CLfD.

Experiment 3 (Architecture): In our final experiment, we fix Adam as the optimizer and Kaiming as the initializer, as they achieve marginally better median DTW scores than their respective alternatives. In this experiment, we evaluate 4 different network depths (2, 3, 4, or 8 layers) for both the hypernetwork and its generated target network, resulting in 16 different architectures for each of the 4 kinds of hypernetworks. We also modify the number of units in each layer such that the overall parameter size of the networks is roughly similar and comparable to the network sizes of the previous experiments. We follow the same training procedure as the previous two experiments and repeat each run 5 times with independent seeds. The median DTW results are shown in Fig. 2. For HN→NODE, the depth of the target network affects the overall performance much more than the depth of the hypernetwork, while for HN→sNODE, almost all architectures achieve similar results irrespective of the depth of either network. Chunked hypernetworks (CHN→NODE and CHN→sNODE) on the other hand, perform best when the hypernetwork is 3-4 layers deep and the target network is 4-8 layers deep. However, similar to HN→sNODE, CHN→sNODE also performs similarly for almost all architectures. In summary, hypernetworks with sNODE

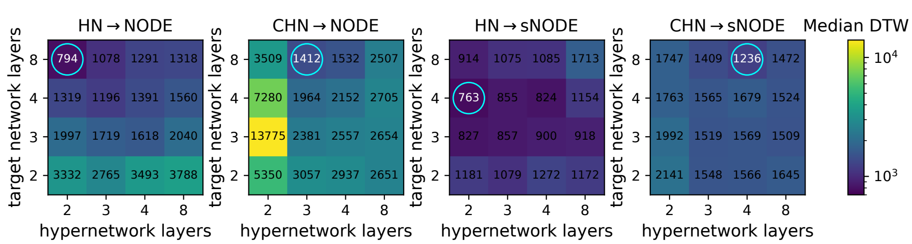

Fig. 2. Effect of hypernetwork depth (y-axis) and target network depth (x-axis) on continual learning from demonstration. Each heatmap corresponds to a different hypernetwork type. Circled numbers show the best DTW for each hypernetwork. Colors are scaled logarithmically. Median values over 5 independent runs are shown.

as the target network perform well and are much less sensitive to the network architecture than hypernetworks with NODE.

4 Conclusion

We demonstrated the effects of the optimizer, initializer, and network architecture on hypernetwork-based continual learning from demonstration. Our findings show that adaptive learning rate optimizers (Adam, RMSProp) are a good choice for CLfD. Kaiming is a good default initializer that performs as well as PWI which is specially designed for hypernetworks. We also showed that hypernetworks with sNODE as the target network are mostly independent of the network architecture. Our findings can help in making informed design decisions while developing hypernetwork-based methods for CLfD in the future.

References

1. A. Billard, S. Calinon, and R. Dillmann, "Learning from humans," *Springer Handbook of Robotics, 2nd Ed.*, 2016.
2. S. Auddy, J. Hollenstein, M. Saveriano, A. Rodríguez-Sánchez, and J. Piater, "Continual learning from demonstration of robotics skills," *Robotics and Autonomous Systems*, vol. 165, p. 104427, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889023000660
3. S. Auddy, J. Hollenstein, M. Saveriano, A. Rodríguez-Sánchez, and J. Piater, "Scalable and efficient continual learning from demonstration via hypernetwork-generated stable dynamics model," *arXiv preprint arXiv:2311.03600*, 2023.
4. J. Z. Kolter and G. Manek, "Learning stable deep dynamics models," *Advances in Neural Information Processing Systems*, vol. 32, pp. 11 128–11 136, 2019.
5. P. Schöpf, S. Auddy, J. Hollenstein, and A. Rodriguez-Sanchez, "Hypernetwork-ppo for continual reinforcement learning," in *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.
6. G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019.
7. O. Chang, L. Flokas, and H. Lipson, "Principled weight initialization for hypernetworks," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020.* OpenReview.net, 2020.
8. K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.
9. J. von Oswald, C. Henning, J. Sacramento, and B. F. Grewe, "Continual learning with hypernetworks," in *International Conference on Learning Representations (ICLR)*, 2019.
10. T. Tieleman and G. Hinton, "6.5-rmsprop coursera: Neural networks for machine learning university of toronto," *Tech. Rep.*, 2012.
11. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
12. X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.