Meta Continual Learning

Risto Vuorio, Subin Yi, Dong-Yeon Cho, Daejoong Kim, and Jiwon Kim
SK T-Brain
{vuoristo, yisubin, dycho24, djkim, jk}@sktbrain.com

Abstract

Using neural networks in practical settings would benefit from the ability of the networks to learn new tasks throughout their lifetimes without forgetting the previous tasks. This ability is limited in the current deep neural networks by a problem called catastrophic forgetting, where training on new tasks tends to severely degrade performance on previous tasks. One way to lessen the impact of the forgetting problem is to constrain parameters that are important to previous tasks to stay close to the optimal parameters. Recently, multiple competitive approaches for computing the importance of the parameters with respect to the previous tasks have been presented. In this paper, we propose a learning to optimize algorithm for mitigating catastrophic forgetting. Instead of trying to formulate a new constraint function ourselves, we propose to train another neural network to predict parameter updates that respect the importance of parameters to the previous tasks. In the proposed meta-training scheme, the update predictor is trained to minimize loss on a combination of current and past tasks. We show experimentally that the proposed approach works in the continual learning setting.

1 Introduction

While we do not know exactly how humans learn throughout their lifetime, it is obvious that humans can continually accumulate information obtained by study or experience and efficiently develop new skills by using acquired knowledge. Many artificial neural network models have taken their inspiration from biological neural networks, however, they struggle to adeptly deal with solving multiple tasks sequentially. The primary source of this weakness has long been known as catastrophic forgetting or interference [18, 21, 7] where learning a new task may alter some important connection weights for old tasks and, in turn, make networks lose their ability to do previous tasks. Many attempts have been made to alleviate this intricate problem resorting to (i) increase in network capacity [23] that handles new tasks without affecting learned networks, (ii) extra-memory [16] or generative models [25] that sustainedly provide the data for past tasks, and (iii) consideration of a regularization term [13, 31, 3] that restrains drastic changes of current model.

Most of the highest performing algorithms in continual learning belong to the third category. At a high-level, the regularization terms employed by these approaches work by limiting the per-parameter updates to reduce the negative impact of learning a new task on the previously learned tasks. While these approaches often achieve high performance on continual learning tasks, they are limited by the accuracy of the assumptions made by the algorithm designers. For example, approximating the sensitivity of the performance on previous tasks to changes in each individual parameter is a challenging problem and improvements in the third category have come from improving this approximation [31, 3]. In order to design a truly general continual learning algorithm, it seems unlikely that hand-crafted approximations will be enough. To take a step into the direction of more general continual learning algorithms, inspired by the recent successes in meta-learning and learning to learn research [4, 6], we explore automatically learning a learning rule for continual learning.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

The purpose of this work is to show the feasibility of learning to learn without forgetting. We propose a meta-learning approach to continual learning where a model for adjusting per-parameter update step is trained to mitigate forgetting of past tasks. The aim of this update step prediction model is to assign small updates for parameters that should have low flexibility to maintain performance on the previous tasks and large updates for parameters that can be changed freely for the current task.

2 Our approach

One of the most typical deep learning scenarios is supervised learning, where we want to find parameters θ of the input-output mapping function f for the given task $\mathcal{T}$. During training, these parameters are updated iteratively, $\theta^{new} = \theta^{old} + \Delta\theta_t$. Gradient based optimization algorithms search the effective update step $\Delta\theta$ by considering the gradient $g = \nabla_\theta \mathcal{L}(f_\theta(\mathcal{T}))$ of a loss function $\mathcal{L}$ that measures distance between the output of the function f and that of given task. In SGD, for example, $\Delta\theta$ is set to $-\alpha g$, where α is the learning rate which can be regarded as a shared hyperparameter across all parameters in the model. As in [4], the optimizers learned in an automatic way also used this gradient as an input to predict the optimal update step. Although the learned optimizers achieve faster convergence in a limited test setting than hand-crafted optimizers, there is little chance to avoid catastrophic forgetting without thoughtful design of a method to carefully control the updates.

Instead of relying on human experts for developing superior intelligent systems, meta-learning algorithms try to make AI systems learn how to learn. Under the general meta-learning umbrella, many different algorithms have been proposed in various areas of deep learning. These algorithms include, for example, good initialization of parameters for fast adaptation to new data [6, 2] and improved optimization of gradient descent techniques [4]. In this paper, we present a meta-learning approach to continual learning, where we consider an update step prediction model h represented by a multilayer perceptron or an RNN (Figure 1). The model is trained to limit the updates for parameters of the mapping function f that are important for performance on the previous tasks and allow large updates for parameters used to learn the current task. Our prediction model is similar to the learnable optimizer in [4] with the distinction that our optimizer is specialized into continual learning through its inputs and the meta-training setting.

In the ideal case, the update step predictor would be able to consider all of the mapping function parameters simultaneously. However, such approach is not feasible as even very simple deep neural networks often have tens of thousands of parameters. Instead, and again similarly to [4], our update step predictor operates on the mapping function parameters element-wise.

When a new task is given to the mapping function f for continual learning, the function's parameters θ are updated using the outputs of the update step prediction model as follows:

$$\begin{aligned} g^*_{j-1} &= \nabla_{\theta^*} \mathcal{L}(f_{\theta^*}(\mathcal{T}_{j-1})), \\ g_j &= \nabla_\theta \mathcal{L}(f_\theta(\mathcal{T}_j)), \\ \Delta\theta &= h_\phi(g^*_{j-1}, g_j, \mathcal{I}), \\ \theta' &= \theta - \eta\Delta\theta, \end{aligned} \tag{1}$$

where j indexes the tasks in the continual learning sequence, θ^* are the (locally) optimal parameters for the previous task, g^*_{j-1} are the average squared gradients of the previous task computed at θ^*, h_ϕ is the update step prediction function parameterized by ϕ, the scaling factor η is a hyperparameter and $\mathcal{I}$ denotes the inputs to predict update step per parameter. In order to enable the update step predictor to output suitable updates considering the current and the previous tasks, it should be able to compute the importance of each mapping function parameter for the previous task compared to the other parameters. To make this possible, the update step predictor is fed the previous model parameters, the average squared gradients of the previous task and the current model parameters as inputs in addition to the current gradient.

The update step predictor parameters ϕ can be trained using a conventional optimization algorithm such as Adam [12]. The loss for the update predictor is computed over the combination of previous and current tasks. Computing this loss requires backpropagation through the last step of the update prediction model parameter as follows:

$$\phi \leftarrow \texttt{Adam}(\nabla_\phi \mathcal{L}(f_\theta(\mathcal{T}_{j-1} \cup \mathcal{T}_j))), \tag{2}$$

where we optimize over the union of the current and previous meta-training tasks. The motivation for this formulation is that, after the prediction model parameter update, we want to (i) achieve high performance on the current task (ii) retain high performance on the previously encountered tasks. Minimizing the loss on both previous and current tasks makes the update step predictor to assign small outputs to parameters that are central to the previous tasks and large outputs to parameters that are flexible to the current task. In this work, we focus on training of our update step prediction model as well as demonstrating how the prediction model can be used to keep important parameters for previous tasks unchanged in sequential classification problems. However, many interesting variations of our method are possible. For example, we can have the model prepare for fast transfer learning on future tasks considering the next task $\mathcal{T}_{0,j+1}$ during the meta continual learning.

The outline of the meta continual learning is presented in Algorithm 1. For training our update step prediction model, we consider an independent meta-training dataset $\mathcal{T}_0$, which contains subtasks similar to the target continual learning tasks. Assuming that these subtasks are given sequentially, h_ϕ is iteratively optimized over two consecutive subtasks as explained above. During this meta-training, we expect that h_ϕ gains the ability to control per-parameter update step for preventing catastrophic forgetting. Note that in our algorithm, we train the mapping function on the first task of each task sequence using a standard optimizer such as Adam because in the beginning of the sequence there is nothing to preserve. After the completion of meta-training, we start continual learning for the given tasks by using the trained h_{ϕ^*} function (Algorithm 2). Similarly as in our meta-training setting, we first train a deep neural network f_θ over $\mathcal{T}_1$ by Adam optimizer. Then, f_θ is continually trained over the following tasks not by using a conventional optimizer for deep neural networks but by using the learned update prediction model h_{ϕ^*} from Algorithm 1.

Following the recent learning to optimize approaches, a natural extension of our idea would be to use recurrent neural networks to parameterize the update step predictor. The meta-optimization process for the recurrent update step predictor would be the same as for the feed-forward predictor for the most part, except backpropagation through time would be used to optimize the recurrent model. We conducted preliminary experiments using an LSTM based update step predictor, but the results were inconclusive and are omitted from the paper.

3 Experiments

In recent research [9, 16, 25], continual learning has been most commonly applied to classification problems such as variations of images from the MNIST handwritten digit database. Using a variation of the MNIST dataset, we carried out experiments to validate the proposed algorithm as a feasible approach for mitigating catastrophic forgetting. In continual learning, a model is trained over a sequence of tasks. In meta-learning, a distribution over tasks is used to meta-train a model, which is expected to generalize to other tasks from the same distribution. Due to the nature of both continual learning and meta-learning, we need access to a distribution of tasks. To generate a suitable distribution over tasks, we follow a problem setting previously considered in continual learning [27, 8], where the tasks are pixel permutations of the MNIST dataset. In this setting, each task, both for the meta-training and continual learning, is generated by shuffling the pixels of the original dataset by a fixed random permutation. The setting gives us access to numerous datasets of equal size and difficulty, where a model trained on single task is expected to not be able solve the other tasks without finetuning.

In our experimental setting we use a fully-connected neural network as the classification model. The model is trained on the sequence of pixel shuffled tasks. The classification network has two hidden layers of 800 ReLU units. The update step predictor is another fully connected neural network with two layers of ten neurons that have the ReLU activation function.

3.1 Disjoint MNIST

In this section, we show the feasibility of the proposed algorithm by looking into the behavior of the learned update step prediction model on two sequential tasks. We use the Disjoint MNIST problem setting from [27]. In the Disjoint MNIST problem setting, MNIST classes are split into two disjoint sets, one consisting of images from $\{0, 1, 2, 3, 4\}$ and the other from $\{5, 6, 7, 8, 9\}$. Similarly to [14] we consider the problem of ten class joint classification. For the meta-training phase, we use another MNIST permutation, which has been split into two tasks similarly to the target task. Results from the

Method	disjoint MNIST	shuffled MNIST
SGD (untuned)	47.72 ± 0.11	89.15 ± 2.34
SGD (tuned)	71.32 ± 1.54	~ 95.5
EWC	52.72 ± 1.36	~ 98.2
IMM (best)	94.12 ± 0.27	98.3 ± 0.08
ours (MLP)	82.3 ± 0.92	95.5 ± 0.58

Table 1: Averaged test accuracy (%) $\pm$ standard deviation over 10 runs. Numbers for other methods are excerpts from [14].

experiment are presented in Table 1. The results demonstrate that our model is capable of significantly reducing the catastrophic forgetting compared to the baseline of training the prediction model on one task and then the other using SGD. This indicates that the trained update step prediction model has an ability to deal with current task while preserving knowledge about the previous task. We also confirmed this by checking the test accuracy of each task after finishing continual learning (Figure 2).

3.2 Multi-task setting

We conduct experiments on sequences of three shuffled MNIST tasks [8, 27]. In this setting, the continual learning tasks are three pixel permutations of MNIST. Similarly to the Disjoint MNIST setting, we train the update predictor using another set of shuffled datasets. In this experimental setting, the state-of-the-art models achieve performance close to the joint training over all tasks [14]. While not being competitive with the other models in terms of performance, our model demonstrates that it has learned to preserve the knowledge of the previous task as shown in Table 1.

4 Discussion

In this paper, we proposed a learning to optimize approach to continual learning. Through the proposed approach, the update step prediction model successfully learned how to guide the optimization of neural network parameters for continual learning. We showed the feasibility of our meta-learning algorithm for continual learning by applying it to sequential image classification problems.

Currently our meta-training and continual training both use pixel permutations of the MNIST dataset. The different permutations still have the same number of samples from each class and the input distributions are closely related, even though they are not the same. It is likely that a well tuned update predictor would be able to exploit similarities in the meta-training and meta-testing datasets. We acknowledge this limitation in our experimental setting and leave exploration of the generalization ability of our approach for future work.

In our experiments we limit our learning setting to a maximum of three sequential tasks. In general continual learning, we would naturally be interested in learning on much longer tasks sequences. Our current update step predictor is designed to only consider the previous task in the sequence. Information about further back tasks could be incorporated in the optimization process by for example chaining multiple update predictors, each with their own previous task to consider or summarizing previous task information with an RNN. Handling longer sequences would be a promising avenue of future work. Note that our meta continual learning setting would trivially generalize to longer task sequences through sampling meta-training samples from further back tasks.

Many recent advances in machine learning have come from the move from hand designed features to learned ones. The optimizers we use to learn the features have themselves been subject to parameterization and learning. Learning to optimize, however, has proven challenging. In [17, 29] some central challenges of learning to optimize are presented and methods for improving the learned optimizer performance are presented. In this work, we worked in the intersection of two challenging problems: continual learning and learning to optimize. We found learning to optimize challenging in its own right, but we note that the advances presented in [17, 29] are largely orthogonal to our approach, and therefore could be applied to improve our update step predictor.

References

[1] M. Al-Shedivat, T. Bansal, Y. Burda, I. Sutskever, I. Mordatch, and P. Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. In *International Conference on Learning Representations*, 2018.

[2] J. S. Alex Nichol, Joshua Achiam. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999, 2018.

[3] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars. Memory aware synapses: Learning what (not) to forget. *CoRR*, abs/1711.09601, 2017.

[4] M. Andrychowicz, M. Denil, S. Gòmez, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas. Learning to learn by gradient descent by gradient descent. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3981–3989. Curran Associates, Inc., 2016.

[5] G. Berseth, C. Xie, P. Cernek, and M. van de Panne. Progressive reinforcement learning with distillation for multi-skilled motion control. In *International Conference on Learning Representations*, 2018.

[6] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, pages 1126–1135, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[7] R. M. French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[8] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. In *International Conference on Learning Representations*, 2014.

[9] X. He and H. Jaeger. Overcoming catastrophic interference using conceptor-aided backpropagation. In *International Conference on Learning Representations*, 2018.

[10] H. Jaeger. Controlling recurrent neural networks by conceptors. *CoRR*, abs/1403.3369, 2014.

[11] R. Kemker and C. Kanan. Fearnet: Brain-inspired model for incremental learning. In *International Conference on Learning Representations*, 2018.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[13] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[14] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang. Overcoming catastrophic forgetting by incremental moment matching. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4652–4662. Curran Associates, Inc., 2017.

[15] K. Li and J. Malik. Learning to optimize. In *International Conference on Learning Representations*, 2017.

[16] D. Lopez-Paz and M. A. Ranzato. Gradient episodic memory for continual learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6467–6476. Curran Associates, Inc., 2017.

[17] K. Lv, S. Jiang, and J. Li. Learning gradient descent: Better generalization and longer horizons. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, pages 2247–2255, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[18] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165, 1989.

[19] F. Meier, D. Kappler, and S. Schaal. Online learning of a memory for learning rates. In *IEEE International Conference on Robotics and Automation*, 2018.

[20] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner. Variational continual learning. In *International Conference on Learning Representations*, 2018.

[21] R. Ratcliff. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychological Review*, 97(2):285–308, 1990.

[22] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. iCaRL: Incremental classifier and representation learning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 5533–5542, Honolulu, HI, USA, 21–26 Jul 2017. IEEE.

[23] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *CoRR*, abs/1606.04671, 2016.

[24] J. Schmidhuber. POWERPLAY: training an increasingly general problem solver by continually searching for the simplest still unsolvable problem. *CoRR*, abs/1112.5309, 2011.

[25] H. Shin, J. K. Lee, J. Kim, and J. Kim. Continual learning with deep generative replay. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2990–2999. Curran Associates, Inc., 2017.

[26] P. Sprechmann, S. Jayakumar, J. Rae, A. Pritzel, A. P. Badia, B. Uria, O. Vinyals, D. Hassabis, R. Pascanu, and C. Blundell. Memory-based parameter adaptation. In *International Conference on Learning Representations*, 2018.

[27] R. K. Srivastava, J. Masci, S. Kazerounian, F. Gomez, and J. Schmidhuber. Compete to compute. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2310–2318. Curran Associates, Inc., 2013.

[28] A. R. Triki, R. Aljundi, M. B. Blaschko, and T. Tuytelaars. Encoder based lifelong learning. *CoRR*, abs/1704.01920, 2017.

[29] O. Wichrowska, N. Maheswaranathan, M. W. Hoffman, S. G. Colmenarejo, M. Denil, N. de Freitas, and J. Sohl-Dickstein. Learned optimizers that scale and generalize. *CoRR*, abs/1703.04813, 2017.

[30] J. Yoon, E. Yang, J. Lee, and S. J. Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018.

[31] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, pages 3987–3995, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Appendix

Related Work

One of the simplest ways to prevent from catastrophic forgetting of learned knowledge and achieve fast adaptation to new tasks in continual learning is to consider the expansion of network structure

[23, 30] such as adding neurons, layers or networks whenever a new task is given. The main drawback of this approach is that it requires the boundless increase in network capacity as the number of tasks continuously grows even though the capacity can be expanded only when necessary as in [30].

Many continual learning approaches [22, 16] employed a certain form of memory to store data from previous tasks. Rebuffi *et al.* [22] trained their model with the augmented training set which consists of new data and stored examples by means of distillation to prevent catastrophic forgetting. In [16], regarding the losses of the previous data in the memory as inequality constraints, the model parameter was updated for the current task through quadratic programming. This allows not only to alleviate forgetting of the past tasks but also to beneficially transfer the knowledges from the current task to previous ones. Instead of explicit memory, a deep generative model to mimic past data is trained in the generative adversarial networks (GANs) framework [25]. In [28], informative features rather than the data of the previous tasks was captured by an autoencoder. This framework was able to ensure the preservation of important features when facing a new task. Kemker and Kanan [11] proposed FearNet architecture that consists of three brain-inspired neural networks for recent storage, long-term storage, and determining which one should be used for a particular example, respectively. Sprechmann *et al.* [26] employed an embedding network as well as output network to generate keys for training data. For a given query, keys and their output values stored in a memory were retrieved in a context-based way at a test time. Then, they were used to locally adapt the parameters of the output network.

Researchers also tried to alleviate catastrophic forgetting by estimating how important each weight is for the previous tasks. After the pioneering work by [13], where a regularizer to protect important parameters was computed as some coefficient times the squared difference of the current weight values and the last weight values from training on the previous task, some variations for computing the coefficient were proposed. Originally, the coefficient was an approximation of the Fisher information [13], which is computed from the gradients of the loss function. In [31], the coefficient was computed based on the contribution to the total reduction in loss and changes in parameter space over the entire trajectory of training. In [3], the coefficient was computed from the gradients of the trained model output on unlabeled as well as training data points. From the Bayesian inference point of view, Nguyen *et al.* [20] elaborated these approaches by combining online variational inference with Monte Carlo variational inference for neural networks.

Considering multiple models rather than a single model for sequentially given tasks, incremental moment matching (IMM) [14] was proposed to combine the models which had been individually trained on each task so that the final model had the ability to carry out multiple tasks. They tried to make the combined model balance the old and new tasks by approximating a mode of the mixture of two Gaussian posteriors with the assistance of a drop-out based transfer technique.

Using conceptors originally proposed in [10], He and Jaeger [9] adjusted the gradients given by backpropagation so that learning a new task would minimally interfere with previously learned tasks. Whereas this variant of the backpropagation was formulated by human experts, some learning-to-learn or meta-learning approaches successfully designed optimizers for update rules in an automatic way [4, 15, 17]. A problem setting similar to continual learning, but without concern for the forgetting problem has been explored in two meta-learning papers. In [19], an alternative to constraining the weight updates was explored by learning a memory for learning rates to facilitate fast adaptation on similar tasks encountered subsequently. Fast adaptation through meta-learning was also explored in [1], where an RL agent faces a sequence of opponents with increasing difficulty. To incrementally expand motion control skills without forgetting, progressive RL with distillation was proposed in [5]. This is in some way analogous to the Powerplay method [24], but they did not consider the meta learning framework.

Further analysis

To analyze the characteristics of the update step prediction model, we collected the output values of the model at different training stages. As shown in Figure 3, all output values were very closed to zero immediately after starting the meta-training. However, we had a clear observation of tri-modal distributions as the meta-training iterations went by. One possible elucidation on this change is as follows. At early stages of training, our update prediction model had little knowledge of the importance of parameters. The lack of knowledge makes the prediction model have very small output values. While training goes on, our model acquires the knowledge so it is able to supply the appropriate values for adjusting the parameters which are not crucial to the success of previous tasks.

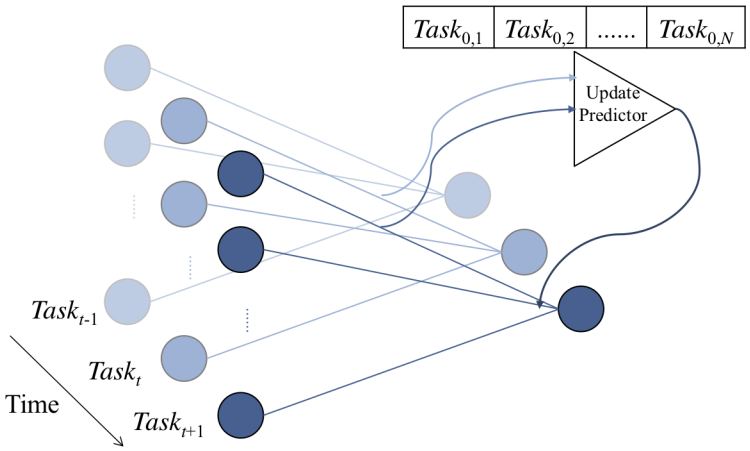

Figure 1: During meta-learning step, the proposed update prediction model was trained on the given task $\mathcal{T}_0$ which consists of some subtasks. Using this model, we successfully learned a deep neural network over the sequentially given tasks without catastrophic forgetting.

Algorithm 1 Meta continual learning for training h_ϕ

procedure META-CONTINUAL-LEARNING($f_\theta, h_\phi, \mathcal{T}_0$)
  for all $\mathcal{T}_{0,j>1}$ in $\mathcal{T}_0$ **do**
    for Epochs 1,2,3, ... **do**
      $\theta^*_{0,j-1} \leftarrow$ train f_θ for one epoch on $\mathcal{T}_{0,j-1}$ using `Adam`
      $\theta \leftarrow \theta^*_{0,j-1}$
      $g_{j-1} \leftarrow \nabla_\theta \mathcal{L}(f_\theta(\mathcal{T}_{0,j-1}))$
      for Epochs 1,2,3, ... **do**
        $g \leftarrow \nabla_\theta \mathcal{L}(f_\theta(\mathcal{T}_{0,j}))$
        $\theta \leftarrow \theta - \eta h_\phi(g_{j-1}, g, \theta^*_{0,j-1}, \mathcal{I})$
        $\phi \leftarrow$ `Adam` $(\nabla_\phi \mathcal{L}(f_\theta(\mathcal{T}_{0,j-1} \cup \mathcal{T}_{0,j})))$
      end for
    end for
  end for
end procedure

Algorithm 2 Continual learning using the trained h_{ϕ^*}

procedure CONTINUAL-LEARNING($f_\theta, h_{\phi^*}, \{\mathcal{T}_1, \mathcal{T}_2, \cdots\}$)
  for all $\mathcal{T}_i$ **do**
    if $i = 1$ **then**
      $\theta^*_1 \leftarrow$ Train f_{θ_1} on $\mathcal{T}_1$ using `Adam`
    else
      $\theta \leftarrow \theta^*_{i-1}$
      $g_{i-1} \leftarrow \nabla_\theta \mathcal{L}(f_\theta(\mathcal{T}_{i-1}))$
      for Epochs 1,2,3, ... **do**
        $g \leftarrow \nabla_\theta \mathcal{L}(f_\theta(\mathcal{T}_i))$
        $\theta \leftarrow \theta - \eta h_{\phi^*}(g_{i-1}, g, \theta^*_{i-1}, \mathcal{I})$
      end for
      $\theta^*_i \leftarrow \theta$
    end if
  end for
end procedure

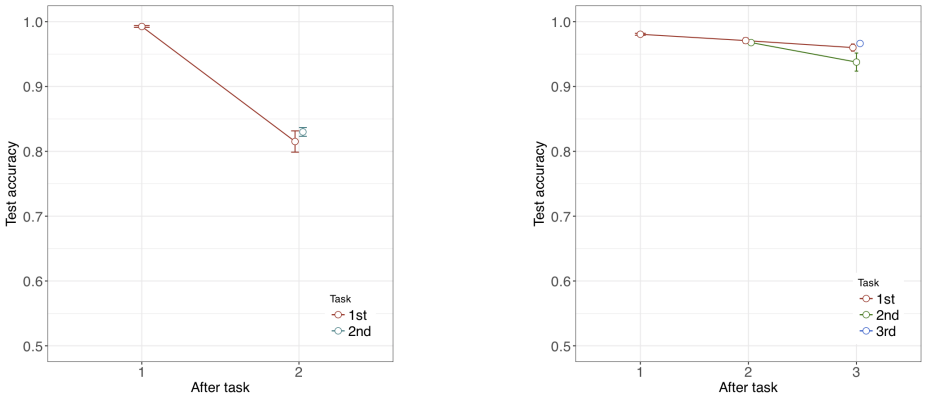

Figure 2: (Left) Results for disjoint MNIST (Right) three tasks from shuffled MNIST with our MLP predictor

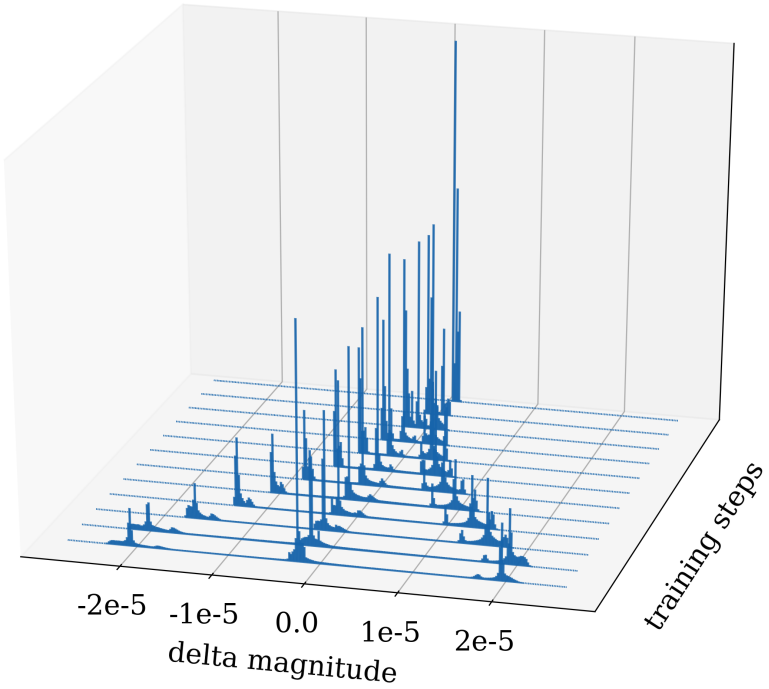

Figure 3: Evolution of the update predictor output (multiplied by the scaling factor η) of our update step prediction model