OPT2023: 15th Annual Workshop on Optimization for Machine Learning

Utility-based Perturbed Gradient Descent: An Optimizer for Continual Learning

Mohamed Elsayed[†] MOHAMEDELSAYED@UALBERTA.CA
A. Rupam Mahmood[†‡] ARMAHMOOD@UALBERTA.CA
[†]*Department of Computing Science, University of Alberta*
[‡] *Alberta Machine Intelligence Institute (Amii)*

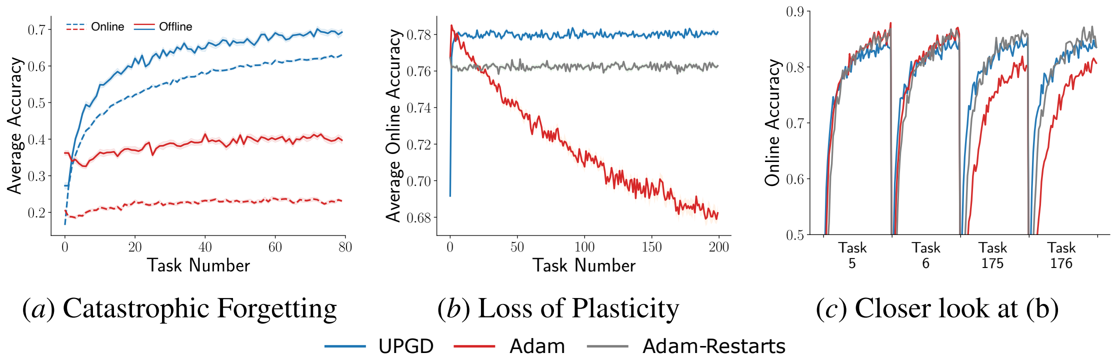
Abstract

Deep representation learning methods struggle with continual learning, suffering from both catastrophic forgetting of useful units and loss of plasticity, often due to rigid and unuseful units. While many methods address these two issues separately, only a few currently deal with both simultaneously. In this paper, we introduce Utility-based Perturbed Gradient Descent (UPGD) as a novel approach for the continual learning of representations. UPGD combines gradient updates with perturbations, where it applies smaller modifications to more useful units, protecting them from forgetting, and larger modifications to less useful units, rejuvenating their plasticity. We adopt the challenging setup of streaming learning as the testing ground and design continual learning problems with hundreds of non-stationarities and unknown task boundaries. We show that many existing methods suffer from at least one of the issues, predominantly manifested by their decreasing accuracy over tasks. On the other hand, UPGD continues to improve performance and surpasses all methods in all problems, being demonstrably capable of addressing both issues.

1. Introduction

While AI has seen notable successes in natural language processing (Radford et al. 2018, Devlin et al. 2019), games (Mnih et al. 2015, Silver et al. 2016), and computer vision (Krizhevsky et al. 2017), we are yet to see effective continual learning agents. *Catastrophic forgetting* (McCloskey & Cohen 1989, Hetherington & Seidenberg 1989) in neural networks is widely recognized as a major challenge of continual learning (de Lange et al. 2021). The phenomenon manifests as the failure of gradient-based methods like SGD or Adam to retain or leverage past knowledge due to forgetting or overwriting previously learned units (Kirkpatrick et al. 2017). In continual learning, these learners often relearn recurring tasks, offering little gain over learning from scratch (Kemker et al. 2018). This issue also raises a concern for reusing large practical models, where finetuning them for new tasks causes significant forgetting of pretrained models (Chen et al. 2020, He et al. 2021).

Methods for mitigating catastrophic forgetting are primarily designed for specific settings. These include settings with independently and identically distributed (i.i.d.) samples, tasks fully contained within a batch or dataset, growing memory requirements, known task boundaries, storing past samples, and offline evaluation. Such setups are often impractical in situations where continual learning is paramount, such as on-device learning. For example, retaining samples may not be possible due to the limitation of computational resources (Hayes et al. 2019, Hayes et al. 2020, Hayes & Kannan 2022, Wang et al. 2023) or concerns over data privacy (Van de Ven et al. 2020).



(*a*) Catastrophic Forgetting (*b*) Loss of Plasticity (*c*) Closer look at (b)

Figure 1: (a) Adam suffers from catastrophic forgetting and hence hardly improves performance. Adam loses plasticity as newer tasks are presented and performs much worse than Adam with restarts later. On the other hand, our proposed method, UPGD, quickly learns to perform better than Adam with restarts and maintains plasticity throughout learning.

In the challenging and practical setting of *streaming learning*, catastrophic forgetting is more severe and remains largely unaddressed (Hayes et al. 2019). In streaming learning, samples are presented to the learner as they arise, which is non-i.i.d. in most practical problems. The learner cannot retain the sample and is thus expected to learn from it immediately. Moreover, evaluation happens online on the most recently presented sample. This setup mirrors animal learning (Hayes et al. 2021, also c.f., list-learning, Ratcliff 1990) and is practical for many applications, such as robotics or autonomous on-device learning. In this work, we consider streaming learning with unknown task boundaries which further amplifies the issue of catastrophic forgetting.

Streaming learning in this work provides the learner with a stream of samples $(\boldsymbol{x}_t, \boldsymbol{y}_t)$ generated using a non-stationary *target function* f_t such that $\boldsymbol{y}_t = f_t(\boldsymbol{x}_t)$. The learner observes the input $\boldsymbol{x}_t \in \mathbb{R}^d$, outputs the prediction $\hat{\boldsymbol{y}}_t \in \mathbb{R}^m$, and then observes the true output $\boldsymbol{y}_t \in \mathbb{R}^m$, strictly in this order. The learner is then evaluated immediately based on the online metric $E(\boldsymbol{y}_t, \hat{\boldsymbol{y}}_t)$, for example, accuracy in classification or squared error in regression. The learner uses a neural network for prediction and E or a related loss to learn the network parameters immediately without storing the sample. The target function f_t is locally stationary in time, where changes to f_t occur occasionally, creating a nonstationary continual learning problem composed of a sequence of stationary tasks.

Figure 1(*a*)subfigure illustrates catastrophic forgetting with Adam in the streaming learning setting. Here, a sequence of tasks based on *Label-Permuted EMNIST* is presented to the learner. The tasks are designed to be highly coherent, where the features learned in one task are fully reusable in the other. Full details of the problem are described in Section 3. If the learner can remember and leverage prior learning, it should continue to improve performance as more tasks are presented. However, Figure 1(*a*)subfigure reveals that Adam can hardly improve its performance, which remains at a low level of accuracy, indicating forgetting. Although catastrophic forgetting is commonly studied under offline evaluation (solid lines), the issue also manifests in online evaluation (dashed lines). This result indicates that current representation learning methods are unable to leverage previously learned useful features but instead forget and relearn them in subsequent tasks.

Yet another pernicious challenge of continual learning is *loss of plasticity*, where the learner's ability to learn new things diminishes. Recent studies reveal that SGD or Adam continues to lose plasticity with more tasks, primarily due to features becoming difficult to modify (Dohare et al. 2021, Lyle et al. 2023). Several methods exist to maintain plasticity, but they generally do not address catastrophic forgetting. Figures 1(b)subfigure and 1(c)subfigure illustrate loss of plasticity, where Adam is presented with a sequence of new tasks based on *Input-Permuted MNIST*. Adam's performance degrades with more tasks and becomes worse than *Adam-Restarts*, which learns from scratch on each task. The stable performance of Adam-Restarts indicates that the tasks are of similar difficulty. Yet, Adam becomes slower to learn over time, demonstrating loss of plasticity.

A method that preserves useful units, such as features or weights, while leaving the other units adaptable would potentially address both catastrophic forgetting and loss of plasticity. Although a few methods address both issues simultaneously, such methods expect known task boundaries, maintain a replay buffer, or require pretraining, which does not fit streaming learning. In this paper, we intend to fill this gap and present a continual learning method that addresses both catastrophic forgetting and loss of plasticity in streaming learning without such limitations.

2. Method

To retain useful units while modifying the rest, we need a metric to assess their utility or usefulness. Here, we first introduce a measure for weight utility and outline an efficient method for computing it. Then, we devise an update rule that modifies weights based on their utility. Weight utility can be defined as the change in loss when setting the weight to zero, essentially removing its connection (Mozer & Smolensky 1988). Removing an important weight should result in increased loss. Ideally, both immediate and future losses matter, but we can only assess immediate loss at the current step.

To define utility precisely, let us consider that the learner produces the predicted output $\hat{\boldsymbol{y}}$ using a neural network with L layers, parametrized by the set of weights $\mathcal{W} = \{\boldsymbol{W}_1, ..., \boldsymbol{W}_L\}$. Here $\boldsymbol{W}_l$ is the weight matrix at the l-th layer, and its element at the i-th row and the j-th column is denoted by $W_{l,i,j}$. At each layer l, we get the activation output $\boldsymbol{h}_l$ of the features by applying the activation function $\boldsymbol{\sigma}$ to the activation input $\boldsymbol{a}_l$: $\boldsymbol{h}_l = \boldsymbol{\sigma}(\boldsymbol{a}_l)$. We simplify notations by defining $\boldsymbol{h}_0 \doteq \boldsymbol{x}$. The activation output $\boldsymbol{h}_l$ is then multiplied by the weight matrix $\boldsymbol{W}_{l+1}$ of layer $l+1$ to produce the next activation input: $a_{l+1,i} = \sum_{j=1}^{d_l} W_{l+1,i,j} h_{l,j}, \forall i$, where $\boldsymbol{h}_l \in \mathbb{R}^{d_l}$. Here, $\boldsymbol{\sigma}$ applies activation element-wise for all layers except for the final layer, which becomes the softmax function.

The utility $U_{l,i,j}(Z)$ of the weight i, j at layer l and sample Z is defined as

$$U_{l,i,j}(Z) \doteq \mathcal{L}(\mathcal{W}_{\neg[l,i,j]}, Z) - \mathcal{L}(\mathcal{W}, Z), \tag{1}$$

where $\mathcal{L}(\mathcal{W}, Z)$ is the sample loss given $\mathcal{W}$, and $\mathcal{L}(\mathcal{W}_{\neg[l,i,j]}, Z)$ is a counterfactual loss where $\mathcal{W}_{\neg[l,i,j]}$ is the same as $\mathcal{W}$ except the weight $W_{l,i,j}$ is set to 0. We refer to it as the *true utility* to distinguish it from its approximations, which are referred to as either *approximated utilities* or simply utilities. Note that this utility is a global measure, and it provides a total ordering for weights according to their importance. However, computing it is prohibitive since it requires additional N_w forward passes, where N_w is the total number of weights.

2.1. Scalable Approximation of the True Utility

Since the computation of the true utility is prohibitive, we aim to approximate it such that no additional forward passes are needed. To that end, we estimate the true utility by a first-order Taylor approximation. We expand the counterfactual loss $\mathcal{L}(\mathcal{W}_{\neg[l,i,j]}, Z)$ around the current weight $W_{l,i,j}$

and evaluate at weight zero. Hence, the approximation of $U_{l,i,j}(Z)$ can be written as

$$\begin{aligned} U_{l,i,j}(Z) &= \mathcal{L}(\mathcal{W}_{\neg[l,i,j]}, Z) - \mathcal{L}(\mathcal{W}, Z) \\ &\approx \mathcal{L}(\mathcal{W}, Z) + \frac{\partial \mathcal{L}(\mathcal{W}, Z)}{\partial W_{l,i,j}}(0 - W_{l,i,j}) - \mathcal{L}(\mathcal{W}, Z) = -\frac{\partial \mathcal{L}(\mathcal{W}, Z)}{\partial W_{l,i,j}} W_{l,i,j}. \end{aligned} \tag{2}$$

2.2. Utility-based Perturbed Gradient Descent

Now, we devise a new method called *Utility-based Perturbed Gradient Descent* (UPGD) that performs gradient-based learning guided by utility-based information. The utility information is used as a gate for the gradients to prevent large updates to already useful weights, addressing forgetting. On the other hand, the utility information helps maintain plasticity by perturbing unuseful weights which become difficult to change through gradients. The update rule of UPGD is given by

$$w_{l,i,j} \leftarrow w_{l,i,j} - \alpha \left(\frac{\partial \mathcal{L}}{\partial w_{l,i,j}} + \xi \right) \left(1 - \bar{U}_{l,i,j} \right), \tag{3}$$

where $\xi \sim \mathcal{N}(0,1)$ is the noise sample, α is the step size, and $\bar{U}_{l,i,j} \in [0,1]$ is a scaled utility. For important weights with utility $\bar{U}_{l,i,j} = 1$, the weight remains unaltered even by gradient descent, whereas unimportant weights with $\bar{U}_{l,i,j} = 0$ get updated by both perturbation and gradient descent. Utility scaling is important for the UPGD update rule. We present here a global scaling. The global scaled utility requires the maximum utility of all weights (e.g., instantaneous or trace) at every time step, which is given by $\bar{U}_{l,i,j} = \phi(U_{l,i,j}/\eta)$. Here η is the maximum utility of the weights and ϕ is the scaling function, for which we use sigmoid.

Another variation of UPGD, which we call *non-protecting UPGD*, is to add the utility-based perturbation to the gradient as $w_{l,i,j} \leftarrow w_{l,i,j} - \alpha[{\partial \mathcal{L}}/{\partial w_{l,i,j}} + \xi(1 - \bar{U}_{l,i,j})]$. However, such an update rule can only help against loss of plasticity, not catastrophic forgetting, as useful weights are not protected from change by gradients. We include non-protecting UPGD in our experiments to validate that using the utility information as a gate for both the perturbation and the gradient update is necessary to mitigate catastrophic forgetting. We provide convergence analysis for both UPGD and Non-protecting UPGD on non-convex stationary problems in Appendix A.

3. Experiments

Here, we compare the effectiveness of UPGD in mitigating continual learning issues against suitable baselines for our streaming learning setup, that is, without replay, batches, or task boundaries. Unless stated otherwise, we averaged the performance of each method over 20 independent runs.

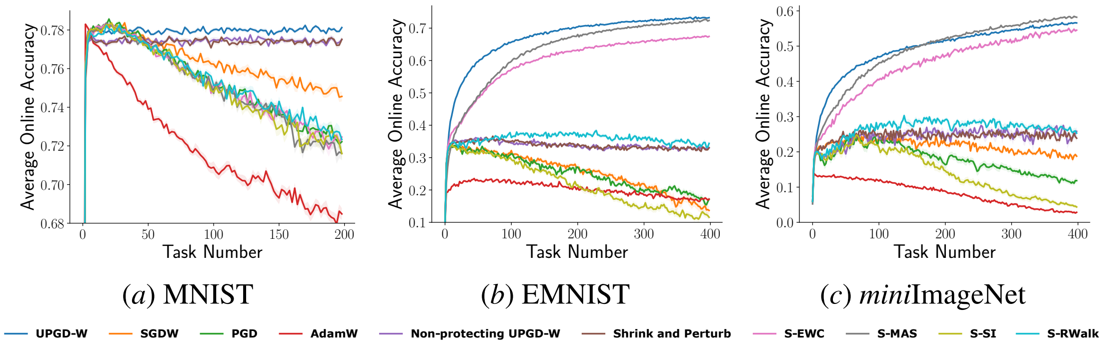
First, we use *Input-Permuted MNIST*, a problem where only loss of plasticity is present, and investigate how UPGD and the other continual learning methods perform on this problem. In Input-permuted MNIST, we permute the inputs every 5000 steps where the time step at each permutation marks the beginning of a new task. After each permutation, the learned representations become irrelevant to the new task, so the learner is expected to overwrite prior-learned representations as soon as possible. Thus, the input-permuted MNIST is a suitable problem to study loss of plasticity.

We compare SGD with weight decay (SGD-W), PGD (Zhou et al. 2019), S&P (Ash & Adams 2020), which addresses loss of plasticity, AdamW (Loshchilov & Hutter 2019), UPGD-W, and

Non-protecting UPGD-W. We also introduce and compare against *Streaming Elastic Weight Consolidation* (S-EWC), *Streaming Synaptic Intelligence* (S-SI), and *Streaming Memory-Aware Intelligence* (S-MAS). These methods can be viewed as a natural extension of EWC (Kirkpatrick et al. 2017), SI (Zenke et al. 2017), and MAS (Aljundi et al. 2018), respectively, which are regularization-based methods for mitigating forgetting to the streaming learning setting. Finally, we introduce and compare against *Streaming RWalk* (S-RWalk). This can be seen as a natural extension of RWalk (Chaudhry et al. 2018), a method that addresses both issues, adapted for streaming learning.

Fig. 2(*a*)subfigure shows that methods that only address catastrophic forgetting (e.g., S-EWC) continue to decay in performance whereas methods that address loss of plasticity alone (e.g., S&P) or together with catastrophic forgetting (e.g., UPGD), except S-RWalk, maintain their performance level. We plot the average online accuracy against the number of tasks. The average online accuracy is the percentage of correct predictions within each task, where the sample online accuracy is 1 for correct prediction and 0 otherwise. The prediction of the learner is given by argmax over its output probabilities. The learners are presented with a total of 1 million examples, one example per time step, and use multi-layer (300 $\times$ 150) networks with ReLU units.

Now, we study how UPGD and other continual learning address forgetting and loss of plasticity using *Label-permuted EMNIST*, and *Label-permuted mini-ImageNet*. The labels are permuted every 2500 time step. Each learner is trained for 1M samples, one sample each time step. Such permutations should not make the learner change its learned representations since it can simply change the weights of the last layer to adapt to that change. This makes the Label-permuted problems suitable for studying catastrophic forgetting. We also might have loss of plasticity present in these problems.



(*a*) MNIST (*b*) EMNIST (*c*) *mini*ImageNet

Figure 2: Performance of methods on Input-permuted MNIST, Label-permuted EMNIST, and Label-permuted mini-ImageNet. A utility trace is used for UPGD/Non-protecting UPGD.

Fig. 2(*b*)subfigure shows that methods addressing catastrophic forgetting, including UPGD-W but except S-RWalk and S-SI, keep improving their performance and outperform methods that only address loss of plasticity. Notably, we observe that S-RWalk, which can address forgetting, struggles in this problem, likely due to the additional loss of plasticity. On the other hand, the performance of S-SI and the rest of the methods keeps deteriorating over time. Fig. 2(*c*)subfigure exhibits the same trends manifested in the previous problem, where methods addressing catastrophic forgetting (e.g., S-EWC) performed the best, whereas methods addressing loss of plasticity (e.g., S&P) only maintained their performance at a lower level.

4. Related Works

Addressing Catastrophic Forgetting. Different approaches have been proposed to mitigate catastrophic forgetting. For example, replay-based methods (e.g., Chaudhry et al. 2019, Isele & Cosgun 2018, Rolnick et al. 2019) address forgetting by using a replay buffer to store incoming non-i.i.d. data and then sample from the buffer i.i.d. samples. Catastrophic forgetting is also addressed by parameter isolation methods (e.g., Rusu et al. 2016, Schwarz et al. 2018, Lee et al. 2019, Wortsman et al. 2020, Ge et al. 2023) that can expand to accommodate new information without significantly affecting previously learned knowledge. There are also sparsity-inducing methods (e.g., Liu et al. 2019, Pan et al. 2021) that work by maintaining sparse connections so that the weight updates can be localized and not affect many prior useful weights. Finally, regularization-based methods (e.g., Kirkpatrick et al. 2017, Aljundi et al. 2018, Aljundi et al. 2019) use a quadratic penalty that discourages the learner from moving too far from the previously learned weights. The penalty amount is usually a function of the weight importance based on its contribution to previous tasks.

Addressing Loss of Plasticity. Dohare et al. (2023) introduced a method that maintains plasticity by continually replacing less useful features and pointed out that methods with continual injection of noise (e.g., Ash & Adams 2020) also maintain plasticity. Later, several methods were presented to retain plasticity. For example, Nikishin et al. (2023) proposed dynamically expanding the network, Abbas et al. (2023) recommended using more adaptive activation functions, and Kumar et al. (2023) proposed a regularization term in the loss function towards the initial parameters.

Addressing Both Issues. The trade-off between plasticity and forgetting has been outlined early by Carpenter & Grossberg (1987) as the stability-plasticity dilemma, a trade-off between maintaining performance on previous experiences and adapting to newer ones. The continual learning community focused more on improving the stability aspect by overcoming forgetting. Recently, however, there has been a new trend of methods that address both issues simultaneously. For example, Chaudhry et al. (RWalk, 2018) utilized a regularization-based approach with a fast-moving average that quickly adapts to the changes in the weight importance, emphasizing the present and the past equally. Despite the recent remarkable advancement in addressing the two issues of continual learning, most existing methods do not fit the streaming learning setting since they require knowledge of task boundaries, replay buffers, or pretraining.

Acknowledgement

We gratefully acknowledge funding from the Canada CIFAR AI Chairs program, the Reinforcement Learning and Artificial Intelligence (RLAI) laboratory, the Alberta Machine Intelligence Institute (Amii), and the Natural Sciences and Engineering Research Council (NSERC) of Canada. We especially thank Shibhansh Dohare for the useful discussions that helped improve this paper.

5. Conclusion

In this paper, we introduced a novel approach to mitigating loss of plasticity and catastrophic forgetting in neural networks. We devised learning rules that protect useful weights and perturb less useful ones, maintaining plasticity and reducing forgetting. We performed a series of online continual learning experiments with many non-stationarities, which is a challenging setting for continual learning. Our experiments showed that UPGD maintains network plasticity and reuses previously learned useful features, being among the only few methods that can address both issues effectively.

References

Abbas, Z., Zhao, R., Modayil, J., White, A., & Machado, M. C. (2023). Loss of Plasticity in Continual Deep Reinforcement Learning. *arXiv preprint arXiv:2303.07507*.

Ash, J., & Adams, R. P. (2020). On warm-starting neural network training. *Advances in Neural Information Processing Systems*, *33*, 3884-3894.

Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., & Tuytelaars, T. (2018). Memory aware synapses: Learning what (not) to forget. European Conference on Computer Vision (pp. 139-154).

Aljundi, R., Kelchtermans, K., & Tuytelaars, T. (2019). Task-free continual learning. *Conference on Computer Vision and Pattern Recognition* (pp. 11254-11263).

Becker, S., & LeCun, Y. (1989). Improving the convergence of backpropagation learning with second-order methods. Proceedings of the 1988 Connectionist Models Summer School (pp. 29-37).

Carpenter, G. A., & Grossberg, S. (1987). ART 2: Self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, *26*(23), 4919-4930.

Chaudhry, A., Dokania, P. K., Ajanthan, T., & Torr, P. H. (2018). Riemannian walk for incremental learning: Understanding forgetting and intransigence. *European Conference on Computer Vision* (pp. 532-547).

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H., & Ranzato, M. A. (2019). On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*.

Chen, S., Hou, Y., Cui, Y., Che, W., Liu, T., & Yu, X. (2020). Recall and Learn: Fine-tuning deep pretrained language models with less forgetting. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*.

De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., ... & Tuytelaars, T. (2021). A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence, 44*(7), 3366–3385.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of the North American Chapter of the Association deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference for Computational Linguistics: Human Language Technologies*.

Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition* (pp.

248-255).

Dohare, S., Hernandez-Garcia, J. F., Rahman, P., Sutton, R. S., & Mahmood, A. R. (2023). Maintaining Plasticity in Deep Continual Learning. *arXiv preprint arXiv:2306.13812*.

Dohare, S., Sutton, R. S., & Mahmood, A. R. (2021). Continual backprop: Stochastic gradient descent with persistent randomness. *arXiv preprint arXiv:2108.06325*.

Dong, X., Chen, S., & Pan, S. (2017). Learning to prune deep neural networks via layer-wise optimal brain surgeon. *Advances in Neural Information Processing Systems*, *30*.

Ge, Y., Li, Y., Wu, D., Xu, A., Jones, A. M., Rios, A. S., ... & Itti, L. (2023). Lightweight Learner for Shared Knowledge Lifelong Learning. *Transactions on Machine Learning Research*.

Gurbuz, M. B., & Dovrolis, C. (2022). NISPA: Neuro-inspired stability-plasticity adaptation for continual learning in sparse networks. *International Conference on Machine Learning* (pp. 8157-8174).

Hetherington, P. A., & Seidenberg, M. S. (1989). Is there 'catastrophic interference' in connectionist networks? *Conference of the Cognitive Science Society* (pp. 26-33).

Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28.

He, T., Liu, J., Cho, K., Ott, M., Liu, B., Glass, J., & Peng, F. (2021). Analyzing the forgetting problem in pretrain-finetuning of open-domain dialogue response models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Conference on Computer Vision and Pattern Recognition* (pp. 770-778).

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *IEEE International Conference on Computer Vision* (pp. 1026-1034).

Hassibi, B., & Stork, D. (1992). Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, *5*.

Hayes, T. L., Cahill, N. D., & Kanan, C. (2019). Memory efficient experience replay for streaming learning. *International Conference on Robotics and Automation* (pp. 9769-9776).

Hayes, T. L., Kafle, K., Shrestha, R., Acharya, M., & Kanan, C. (2020). Remind your neural network to prevent catastrophic forgetting. *European Conference on Computer Vision* (pp. 466-483).

Hayes, T. L., Krishnan, G. P., Bazhenov, M., Siegelmann, H. T., Sejnowski, T. J., & Kanan, C. (2021). Replay in deep learning: Current approaches and missing biological elements. *Neural Computation*, *33*(11), 2908-2950.

Hayes, T. L., Kanan, C. (2022). Online continual learning for embedded devices. In *Conference on Lifelong Learning Agents, PMLR 199*:744–766.

Isele, D., & Cosgun, A. (2018). Selective experience replay for lifelong learning. *AAAI Conference on Artificial Intelligence* (pp. 3302-3309).

Jung, D., Lee, D., Hong, S., Jang, H., Bae, H., & Yoon, S. (2022). New insights for the stability-plasticity dilemma in online continual learning. *International Conference on Learning Representations*.

Karnin, E. D. (1990). A simple procedure for pruning back-propagation trained neural networks. *IEEE transactions on neural networks*, *1*(2), 239-242.

Kemker, R., McClure, M., Abitino, A., Hayes, T., & Kanan, C. (2018). Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI conference on artificial intelligence 32*:1.

Kingma, D. P. & Ba, J. (2015). Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., ... & Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *National Academy of Sciences*, *114*(13), 3521-3526.

Kim, S., Noci, L., Orvieto, A., & Hofmann, T. (2023). Achieving a Better Stability-Plasticity Trade-off via Auxiliary Networks in Continual Learning. *Conference on Computer Vision and Pattern Recognition* (pp. 11930-11939).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM, 60*(6), 84–90.

Krizhevsky, A. (2009) *Learning Multiple Layers of Features from Tiny Images*. Ph.D. dissertation, University of Toronto.

Konorski J. (1948). Conditioned Reflexes and Neuron Organization. *Cambridge University Press*.

Kumar, S., Marklund, H., & Van Roy, B. (2023). Maintaining Plasticity via Regenerative Regularization. *arXiv preprint arXiv:2308.11958*.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

LeCun, Y., Denker, J., & Solla, S. (1989). Optimal brain damage. *Advances in neural information processing systems*, *2*.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

Lesort, T., Ostapenko, O., Rodriguez, P., Arefin, M. R., Misra, D., Charlin, L., & Rish, I. (2023). Challenging Common Assumptions about Catastrophic Forgetting and Knowledge Accumulation. *Conference on Lifelong Learning Agents*.

Liu, V., Kumaraswamy, R., Le, L., & White, M. (2019). The utility of sparse representations for control in reinforcement learning. *AAAI Conference on Artificial Intelligence* (pp. 4384-4391).

Loshchilov, I., & Hutter, F. (2019). Decoupled weight decay regularization. *International Conference on Learning Representations*.

Lyle, C., Zheng, Z., Nikishin, E., Pires, B. A., Pascanu, R., & Dabney, W. (2023). Understanding plasticity in neural networks. *International Conference on Machine Learning* (pp. 23190-23211).

Mahmood, A. R., & Sutton, R. S. (2013). Representation search through generate and test. *AAAI Conference on Learning Rich Representations from Low-Level Sensors* (pp. 16-21).

Mozer, M. C., & Smolensky, P. (1988). Skeletonization: A technique for trimming the fat from a network via relevance assessment. Advances in neural information processing systems, 1.

Molchanov, P., Mallya, A., Tyree, S., Frosio, I., & Kautz, J. (2019). Importance estimation for neural network pruning. *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 11264-11272).

Molchanov, P., Tyree, S., Karras, T., Aila, T., & Kautz, J. (2016). Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *nature, 518*(7540), 529–533.

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

Nikishin, E., Oh, J., Ostrovski, G., Lyle, C., Pascanu, R., Dabney, W., & Barreto, A. (2023). Deep reinforcement learning with plasticity injection. In *Workshop on Reincarnating Reinforcement Learning at ICLR*.

Park, S., Lee, J., Mo, S., & Shin, J. (2020). Lookahead: a far-sighted alternative of magnitude-based pruning. *arXiv preprint arXiv:2002.04809*.

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-Training. *OpenAI blog*.

Ratcliff, R. (1990). Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review, 97*(2), 285–308.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., ... & Hadsell, R. (2016). Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., & Wayne, G. (2019). Experience replay for continual learning. *Advances in Neural Information Processing Systems*, *32*.

Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., & Hadsell, R. (2018). Progress & compress: A scalable framework for continual learning. *International Conference on Machine Learning* (pp. 4528-4537).

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *nature, 529*(7587), 484–489.

Tresp, V., Neuneier, R., & Zimmermann, H. G. (1996). Early brain damage. *Advances in Neural Information Processing Systems*, 9.

Van de Ven, G. M., Siegelmann, H. T., & Tolias, A. S. (2020). Brain-inspired replay for continual learning with artificial neural networks. *Nature communications, 11*(1), 4069.

Wang, Y.*, Vasan, G.*, & Mahmood, A. R. (2023). Real-time reinforcement learning for vision-based robotics utilizing local and remote computers. In *Proceedings of the 2023 International Conference on Robotics and Automation*.

Wortsman, M., Ramanujan, V., Liu, R., Kembhavi, A., Rastegari, M., Yosinski, J., & Farhadi, A. (2020). Supermasks in superposition. *Advances in Neural Information Processing Systems*, *33*, 15173-15184.

Zaheer, M., Reddi, S., Sachan, D., Kale, S., & Kumar, S. (2018). Adaptive methods for nonconvex optimization. *Advances in Neural Information Processing systems*, *31*.

Zenke, F., Poole, B., & Ganguli, S. (2017). Continual learning through synaptic intelligence. *International Conference on Machine Learning* (pp. 3987-3995).

Zhou, M., Liu, T., Li, Y., Lin, D., Zhou, E., & Zhao, T. (2019). Toward understanding the importance of noise in training neural networks. *International Conference on Machine Learning* (pp. 7594-7602).

Appendix A. Convergence Analysis for UPGD and Non-protecting UPGD

In this section, we provide convergence analysis for UPGD and Non-protecting UPGD in nonconvex stationary problems. We focus on the stochastic version of these two algorithms since we are interested in continual learners performing updates at every time step. The following proof shows the convergence to a stationary point up to the statistical limit of the variance of the gradients, where $\|\nabla f(\boldsymbol{\theta})\|^2 \leq \delta$ represent a δ-accurate solution and is used to measure the stationarity of $\boldsymbol{\theta}$. Nonconvex optimization problems can be written as:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} f(\boldsymbol{\theta}) \doteq \mathbb{E}_{S\sim P}\left[\mathcal{L}(\boldsymbol{\theta}, S)\right],$$

where f is the expected loss, $\mathcal{L}$ is the sample loss, S is a random variable for samples and $\boldsymbol{\theta}$ is a vector of weights parametrizing $\mathcal{L}$. We assume that $\mathcal{L}$ is L-smooth, meaning that there exist a constant L that satisfy

$$\|\nabla\mathcal{L}(\boldsymbol{\theta}_1, s) - \nabla\mathcal{L}(\boldsymbol{\theta}_2, s)\| \leq L\|\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1\|, \ \forall\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^d, s \in \mathcal{S}. \tag{4}$$

We further assume that $\mathcal{L}$ has bounded variance in the gradients $\mathbb{E}[\|\nabla\mathcal{L}(\boldsymbol{\theta}, S) - \nabla f(\boldsymbol{\theta})\|^2] \leq \sigma^2, \forall\boldsymbol{\theta} \in \mathbb{R}^d$. We assume that the perturbation noise has bounded variance $\mathbb{E}[\|\boldsymbol{\xi}\|^2] \leq \sigma_n^2$. Note that the assumption of L-smoothness on the sample loss result in L-smooth expected loss too, which is given by $\|\nabla f(\boldsymbol{\theta}_1) - \nabla f(\boldsymbol{\theta}_2)\| \leq L\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|$. For the simplicity of this proof, we use the true instantaneous weight utility, not an approximated one. We assume that any connection in the network has an average utility $\bar{u}$.

A.1. Non-protecting Utility-based Perturbed Gradient Descent

Remember that the update equation of the Non-protecting UPGD can be written as follows when the parameters are stacked in a single vector $\boldsymbol{\theta}$:

$$\theta_{t+1,i} = \theta_{t,i} - \alpha(g_{t,i} + \xi_{t,i}\rho_{t,i}).$$

where α is the step size, $\boldsymbol{g}_t$ is the sample gradient vector at time t, $\boldsymbol{\rho}_t = (1 - \boldsymbol{u}_t)$ is the opposite utility vector, and $\boldsymbol{\xi}_t$ is the noise perturbation. Since the function f is L-smooth, we can write the following:

$$f(\boldsymbol{\theta}_{t+1}) \leq f(\boldsymbol{\theta}_t) + (\nabla f(\boldsymbol{\theta}_t))^\top(\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t) + \frac{L}{2}\|\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t\|_2^2 \tag{5}$$

$$= f(\boldsymbol{\theta}_t) - \alpha\sum_{i=1}^{d}(\nabla[f(\boldsymbol{\theta}_t)]_i(g_{t,i} + \xi_{t,i}\rho_{t,i})) + \frac{L\alpha^2}{2}\sum_{i=1}^{d}(g_{t,i} + \xi_{t,i}\rho_{t,i})^2. \tag{6}$$

Next, we take the conditional expectation of $f(\boldsymbol{\theta}_{t+1})$ as follows:

$$\begin{aligned}
\mathbb{E}_t[f(\boldsymbol{\theta}_{t+1})|\boldsymbol{\theta}_t] &\leq f(\boldsymbol{\theta}_t) - \alpha \sum_{i=1}^{d} \nabla[f(\boldsymbol{\theta}_t)]_i \mathbb{E}_t[(g_{t,i} + \xi_{t,i}\rho_{t,i})] + \frac{L\alpha^2}{2} \sum_{i=1}^{d} \mathbb{E}_t[(g_{t,i} + \xi_{t,i}\rho_{t,i})^2] \\
&= f(\boldsymbol{\theta}_t) - \alpha \sum_{i=1}^{d} \nabla[f(\boldsymbol{\theta}_t)]_i \mathbb{E}_t[g_{t,i}] + \frac{L\alpha^2}{2} \sum_{i=1}^{d} \left(\mathbb{E}_t[g_{t,i}^2] + \mathbb{E}_t[(\xi_{t,i}\rho_{t,i})^2]\right) \\
&= f(\boldsymbol{\theta}_t) - \alpha \sum_{i=1}^{d} \nabla[f(\boldsymbol{\theta}_t)]_i^2 + \frac{L\alpha^2}{2} \sum_{i=1}^{d} \left(\mathbb{E}_t[g_{t,i}^2] + \mathbb{E}_t[(\xi_{t,i}\rho_{t,i})^2]\right) \\
&= f(\boldsymbol{\theta}_t) - \alpha \|\nabla f(\boldsymbol{\theta}_t)\|^2 + \frac{L\alpha^2}{2} \sum_{i=1}^{d} \left(\mathbb{E}_t[g_{t,i}^2] + \mathbb{E}_t[(\xi_{t,i}\rho_{t,i})^2]\right) \\
&\leq f(\boldsymbol{\theta}_t) - \alpha \|\nabla f(\boldsymbol{\theta}_t)\|^2 + \frac{L\alpha^2}{2} \sum_{i=1}^{d} \mathbb{E}_t[g_{t,i}^2] + \frac{L\alpha^2}{2}\sigma_n^2.
\end{aligned}$$

Note that $\mathbb{E}_t[g_{t,i}] = [\nabla f(\boldsymbol{\theta}_t)]_i$, $\mathbb{E}_t[\xi_{t,i}] = 0$, and $\mathbb{E}[(\xi_{t,i}\rho_{t,i})^2] \leq \mathbb{E}[\xi_{t,i}^2]$, since $0 \leq \rho_{t,i} \leq 1 \ \forall t, i$. From the bounded variance assumption, we know that the $\mathbb{E}[\|\boldsymbol{g}_t\|^2]$ is bounded as follows:

$$\mathbb{E}[\|\boldsymbol{g}_t\|^2] \leq \frac{\sigma^2}{b_t} + \|\nabla f(\boldsymbol{\theta}_t)\|^2,$$

where b_t is the batch size at time step t. We can now bound $\mathbb{E}_t[f(\boldsymbol{\theta}_{t+1})|\boldsymbol{\theta}_t]$ as follows:

$$\begin{aligned}
\mathbb{E}_t[f(\boldsymbol{\theta}_{t+1})|\boldsymbol{\theta}_t] &\leq f(\boldsymbol{\theta}_t) - \alpha \|\nabla f(\boldsymbol{\theta}_t)\|^2 + \frac{L\alpha^2}{2}\left(\sigma_n^2 + \frac{\sigma^2}{b_t} + \|\nabla f(\boldsymbol{\theta}_t)\|^2\right) \\
&= f(\boldsymbol{\theta}_t) - \frac{2\alpha - L\alpha^2}{2}\|\nabla f(\boldsymbol{\theta}_t)\|^2 + \frac{L\alpha^2}{2}\left(\sigma_n^2 + \frac{\sigma^2}{b_t}\right).
\end{aligned}$$

Rearranging the inequality and using the telescopic sum, we can write:

$$\frac{2\alpha - L\alpha^2}{2} \sum_{t=1}^{T} \|\nabla f(\boldsymbol{\theta}_t)\|^2 \leq f(\boldsymbol{\theta}_1) - \mathbb{E}_{T+1}[f(\boldsymbol{\theta}_{T+1})] + \frac{L\alpha^2 T}{2}\left(\sigma_n^2 + \frac{\sigma^2}{b_t}\right).$$

Multiplying both sides by $\frac{2}{T(2\alpha - L\alpha^2)}$ and using the fact that f is the lowest at the global minimum $\boldsymbol{\theta}^*$: $f(\boldsymbol{\theta}_{T+1}) \geq f(\boldsymbol{\theta}^*)$, we can write the following:

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\|\nabla f(\boldsymbol{\theta}_t)\|^2 \leq 2\frac{f(\boldsymbol{\theta}_1) - f(\boldsymbol{\theta}^*)}{T(2\alpha - L\alpha^2)} + \frac{L\alpha^2 2(\sigma_n^2 b_t + \sigma^2)}{b_t(2\alpha - L\alpha^2)}.$$

Therefore, the algorithm converges to a stationary point. However, in the limit $T \to \infty$, the algorithm has to have an increasing batch size or a decreasing step size to converge, which is the same requirement for convergence of other stochastic gradient-based methods at the limit (see Zaheer et al. 2018).

A.2. Utility-based Perturbed Gradient Descent

Remember that the update equation of UPGD can be written as follows when the parameters are stacked in a single vector $\boldsymbol{\theta}$:

$$\theta_{t+1,i} = \theta_{t,i} - \alpha(g_{t,i} + \xi_{t,i})\rho_{t,i}.$$

where α is the step size, $\boldsymbol{g}_t$ is the sample gradient vector at time t, $\boldsymbol{\rho}_t = (1 - \boldsymbol{u}_t)$ is the opposite utility vector, and $\boldsymbol{\xi}_t$ is the noise perturbation. Since the function f is L-smooth, we can write the following:

$$f(\boldsymbol{\theta}_{t+1}) \leq f(\boldsymbol{\theta}_t) + (\nabla f(\boldsymbol{\theta}_t))^\top(\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t) + \frac{L}{2}\|\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t\|_2^2 \tag{7}$$

$$= f(\boldsymbol{\theta}_t) - \alpha\sum_{i=1}^{d}\left(\nabla[f(\boldsymbol{\theta}_t)]_i\rho_{t,i}(g_{t,i} + \xi_{t,i})\right) + \frac{L\alpha^2}{2}\sum_{i=1}^{d}(g_{t,i} + \xi_{t,i})^2\rho_{t,i}^2. \tag{8}$$

Next, we take the conditional expectation of $f(\boldsymbol{\theta}_{t+1})$ as follows:

$$\begin{aligned}
\mathbb{E}_t[f(\boldsymbol{\theta}_{t+1})|\boldsymbol{\theta}_t] &\leq f(\boldsymbol{\theta}_t) - \alpha\sum_{i=1}^{d}\nabla[f(\boldsymbol{\theta}_t)]_i\mathbb{E}_t[\rho_{t,i}(g_{t,i} + \xi_{t,i})] + \frac{L\alpha^2}{2}\sum_{i=1}^{d}\mathbb{E}_t[(g_{t,i} + \xi_{t,i})^2\rho_{t,i}^2] \\
&= f(\boldsymbol{\theta}_t) - \alpha\sum_{i=1}^{d}\nabla[f(\boldsymbol{\theta}_t)]_i^2\mathbb{E}_t[\rho_{t,i}] + \frac{L\alpha^2}{2}\sum_{i=1}^{d}\mathbb{E}_t[g_{t,i}^2]\mathbb{E}_t[\rho_{t,i}^2] + \mathbb{E}_t[(\xi_{t,i}\rho_{t,i})^2] \\
&\leq f(\boldsymbol{\theta}_t) - \alpha\bar{\rho}\sum_{i=1}^{d}\nabla[f(\boldsymbol{\theta}_t)]_i^2 + \frac{L\alpha^2}{2}\left(\sigma_n^2 + \frac{\sigma^2}{b_t} + \|\nabla f(\boldsymbol{\theta}_t)\|^2\right) \\
&= f(\boldsymbol{\theta}_t) - \left(\alpha\bar{\rho} - \frac{L\alpha^2}{2}\right)\|\nabla f(\boldsymbol{\theta}_t)\|^2 + \frac{L\alpha^2}{2}\left(\sigma_n^2 + \frac{\sigma^2}{b_t}\right).
\end{aligned}$$

Note that $\bar{\rho} = 1 - \bar{u}$, $\mathbb{E}_t[g_{t,i}] = [\nabla f(\boldsymbol{\theta}_t)]_i$, $\mathbb{E}_t[\xi_{t,i}] = 0$, and $\mathbb{E}[(\xi_{t,i}\rho_{t,i})^2] \leq \mathbb{E}[\xi_{t,i}^2]$, since $0 \leq \rho_{t,i} \leq 1\ \forall t, i$.

Rearranging the inequality and using the telescopic sum, we can write:

$$\frac{2\alpha\bar{\rho} - L\alpha^2}{2}\sum_{t=1}^{T}\|\nabla f(\boldsymbol{\theta}_t)\|^2 \leq f(\boldsymbol{\theta}_1) - \mathbb{E}_{T+1}[f(\boldsymbol{\theta}_{T+1})] + \frac{L\alpha^2 T}{2}\left(\sigma_n^2 + \frac{\sigma^2}{b_t}\right).$$

Multiplying both sides by $\frac{2}{T(2\alpha\bar{\rho} - L\alpha^2)}$ and using the fact that f is the lowest at the global minimum $\boldsymbol{\theta}^*$: $f(\boldsymbol{\theta}_{T+1}) \geq f(\boldsymbol{\theta}^*)$, we can write the following:

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla f(\boldsymbol{\theta}_t)\|^2 \leq 2\frac{f(\boldsymbol{\theta}_1) - f(\boldsymbol{\theta}^*)}{T(2\alpha\bar{\rho} - L\alpha^2)} + \frac{L\alpha^2(\sigma_n^2 b_t + \sigma^2)}{b_t(2\alpha\bar{\rho} - L\alpha^2)}.$$

Therefore, the algorithm converges to a stationary point. However, in the limit $T \to \infty$, the algorithm has to have an increasing batch size or a decreasing step size to converge, which is the same requirement for convergence of other stochastic gradient-based methods at the limit (see Zaheer et al. 2018).

Appendix B. Utility Propagation

The instantaneous utility measure can be used in a recursive formulation, allowing for backward propagation. We can get a recursive formula for the utility equation for connections in a neural network. This property is a result of Theorem 1.

Theorem 1 *If the second-order off-diagonal terms in all layers in a neural network except for the last one are zero and all higher-order derivatives are zero, the true weight utility for the weight ij at the layer l can be propagated using the following recursive formulation:*

$$U_{l,i,j}(Z) \doteq f_{l,i,j} + s_{l,i,j}$$

where

$$\begin{aligned} f_{l,i,j} &\doteq \frac{\sigma'(a_{l,i})}{h_{l,i}} h_{l-1,j} W_{l,i,j} \sum_{k=1}^{|\boldsymbol{a}_{l+1}|} f_{l+1,k,i}, \\ s_{l,i,j} &\doteq \frac{1}{2} h_{l-1,j}^2 W_{l,i,j}^2 \sum_{k=1}^{|\boldsymbol{a}_{l+1}|} \left(2 s_{l+1,k,i} \frac{\sigma'(a_{l,i})^2}{h_{l,i}^2} - \frac{\sigma''(a_{l,i})}{h_{l,i}} f_{l+1,k,i} \right). \end{aligned}$$

Proof

First, we start by writing the partial derivative of the loss with respect to each weight in terms of earlier partial derivatives in the next layers as follows:

$$\frac{\partial \mathcal{L}}{\partial a_{l,i}} = \sum_{k=1}^{|\boldsymbol{a}_{l+1}|} \frac{\partial \mathcal{L}}{\partial a_{l+1,k}} \frac{\partial a_{l+1,k}}{\partial h_{l,i}} \frac{\partial h_{l,i}}{\partial a_{l,i}} = \sigma'(a_{l,i}) \sum_{k=1}^{|\boldsymbol{a}_{l+1}|} \frac{\partial \mathcal{L}}{\partial a_{l+1,k}} W_{l+1,k,i}, \tag{9}$$

$$\frac{\partial \mathcal{L}}{\partial W_{l,i,j}} = \frac{\partial \mathcal{L}}{\partial a_{l,i}} \frac{\partial a_{l,i}}{\partial W_{l,i,j}} = \frac{\partial \mathcal{L}}{\partial a_{l,i}} h_{l-1,j}. \tag{10}$$

Next, we do the same with second-order partial derivatives as follows:

$$\widehat{\frac{\partial^2 \mathcal{L}}{\partial a_{l,i}^2}} \doteq \sum_{k=1}^{|\boldsymbol{a}_{l+1}|} \left[\widehat{\frac{\partial^2 \mathcal{L}}{\partial a_{l+1,k}^2}} W_{l+1,k,i}^2 \sigma'(a_{l,i})^2 + \frac{\partial \mathcal{L}}{\partial a_{l+1,k}} W_{l+1,k,i} \sigma''(a_{l,i}) \right], \tag{11}$$

$$\widehat{\frac{\partial^2 \mathcal{L}}{\partial W_{l,i,j}^2}} \doteq \widehat{\frac{\partial^2 \mathcal{L}}{\partial a_{l,i}^2}} h_{l-1,j}^2. \tag{12}$$

Now, we derive the utility propagation formulation as the sum of two recursive quantities, $f_{l,ij}$ and $s_{l,ij}$. These two quantities represent the first and second-order terms in the Taylor approximation. Using Eq. 9, Eq. 10, Eq. 11, and Eq. 12, we can derive the recursive formulation as follows:

$$\begin{aligned}
U_{l,i,j}(Z) &\doteq -\frac{\partial \mathcal{L}(\mathcal{W},Z)}{\partial W_{l,i,j}} W_{l,i,j} + \frac{1}{2}\frac{\partial^2 \mathcal{L}(\mathcal{W},Z)}{\partial W_{l,ij}^2} W_{l,ij}^2 \\
&\approx -\frac{\partial \mathcal{L}(\mathcal{W},Z)}{\partial W_{l,i,j}} W_{l,i,j} + \frac{1}{2}\frac{\partial^2 \widehat{\mathcal{L}(\mathcal{W},Z)}}{\partial W_{l,ij}^2} W_{l,ij}^2 \\
&= -\frac{\partial \mathcal{L}}{\partial a_{l,i}} h_{l-1,j} W_{l,i,j} + \frac{1}{2}\frac{\partial^2 \widehat{\mathcal{L}(\mathcal{W},Z)}}{\partial a_{l,i,j}^2} h_{l-1,j}^2 W_{l,ij}^2 \\
&= f_{l,i,j} + s_{l,i,j}. \qquad (13)
\end{aligned}$$

From here, we can write the first-order part $f_{l,i,j}$ and the second-order part $s_{l,i,j}$ as follows:

$$f_{l,i,j} = -\sigma'(a_{l,i}) h_{l-1,j} W_{l,i,j} \sum_{k=1}^{|\boldsymbol{a}_{l+1}|} \left(\frac{\partial \mathcal{L}}{\partial a_{l+1,k}} W_{l+1,k,i} \right) \qquad (14)$$

$$s_{l,i,j} = \frac{1}{2} h_{l-1,j}^2 W_{l,i,j}^2 \sum_{k=1}^{|\boldsymbol{a}_{l+1}|} \left(\widehat{\frac{\partial^2 \mathcal{L}}{\partial a_{l+1,k}^2}} W_{l+1,k,i}^2 \sigma'(a_{l,i})^2 + \frac{\partial \mathcal{L}}{\partial a_{l+1,k}} W_{l+1,k,i} \sigma''(a_{l,i}) \right) \qquad (15)$$

Using Eq. 14 and Eq. 15, we can write the recursive formulation for $f_{l,ij}$ and $s_{l,ij}$ as follows:

$$\begin{aligned}
f_{l,ij} &= -\sigma'(a_{l,i}) h_{l-1,j} W_{l,i,j} \sum_{k=1}^{|\boldsymbol{a}_{l+1}|} \left(\frac{\partial \mathcal{L}}{\partial a_{l+1,k}} W_{l+1,k,i} \right) \\
&= \frac{\sigma'(a_{l,i})}{h_{l,i}} h_{l-1,j} W_{l,i,j} \sum_{k=1}^{|\boldsymbol{a}_{l+1}|} \left(-\frac{\partial \mathcal{L}}{\partial a_{l+1,k}} h_{l,i} W_{l+1,k,i} \right) \\
&= \frac{\sigma'(a_{l,i})}{h_{l,i}} h_{l-1,j} W_{l,i,j} \sum_{k=1}^{|\boldsymbol{a}_{l+1}|} f_{l+1,k,i} \qquad (16)
\end{aligned}$$

$$\begin{aligned}
s_{l,i,j} &= \frac{1}{2} h_{l-1,j}^2 W_{l,i,j}^2 \sum_{k=1}^{|\boldsymbol{a}_{l+1}|} \left(\widehat{\frac{\partial^2 \mathcal{L}}{\partial a_{l+1,k}^2}} W_{l+1,k,i}^2 \sigma'(a_{l,i})^2 + \frac{\partial \mathcal{L}}{\partial a_{l+1,k}} W_{l+1,k,i} \sigma''(a_{l,i}) \right) \qquad (17) \\
&= \frac{1}{2} h_{l-1,j}^2 W_{l,i,j}^2 \sum_{k=1}^{|\boldsymbol{a}_{l+1}|} \left(\widehat{\frac{\partial^2 \mathcal{L}}{\partial a_{l+1,k}^2}} h_{l,i}^2 W_{l+1,k,i}^2 \frac{\sigma'(a_{l,i})^2}{h_{l,i}^2} - \frac{\sigma''(a_{l,i})}{h_{l,i}} f_{l+1,k,i} \right) \qquad (18) \\
&= \frac{1}{2} h_{l-1,j}^2 W_{l,i,j}^2 \sum_{k=1}^{|\boldsymbol{a}_{l+1}|} \left(2 s_{l+1,k,i} \frac{\sigma'(a_{l,i})^2}{h_{l,i}^2} - \frac{\sigma''(a_{l,i})}{h_{l,i}} f_{l+1,k,i} \right) \qquad (19)
\end{aligned}$$

■

Appendix C. Utility-baesd Perturbed Gradient Descent Algorithm

Algorithm 1: UPGD

```
Data: Given a stream of data $\mathcal{D}$, a network $f$ with weights $\{\boldsymbol{W}_1, ..., \boldsymbol{W}_L\}$.
Initialize Step size $\alpha$, decay rate $\beta$.
Initialize $\{\boldsymbol{W}_1, ..., \boldsymbol{W}_L\}$.
Initialize $\boldsymbol{U}_l, \hat{\boldsymbol{U}}_l, \forall l$ to zero.
Initialize time step $t \leftarrow 0$.
for $(\boldsymbol{x}, \boldsymbol{y})$ in $\mathcal{D}$ do
    $t \leftarrow t + 1$;
    for $l$ in $\{L, L-1, ..., 1\}$ do
        $\eta \leftarrow -\infty$
        $\boldsymbol{F}_l, \boldsymbol{S}_l \leftarrow \texttt{GetDerivatives}(f, \boldsymbol{x}, \boldsymbol{y}, l)$;
        $\boldsymbol{M}_l \leftarrow {}^1\!/_2 \boldsymbol{S}_l \circ \boldsymbol{W}_l^2 - \boldsymbol{F}_l \circ \boldsymbol{W}_l$
        $\boldsymbol{U}_l \leftarrow \beta \boldsymbol{U}_l + (1-\beta)\boldsymbol{M}_l$
        $\hat{\boldsymbol{U}}_l \leftarrow \boldsymbol{U}_l / (1 - \beta^t)$
    end
    if $\eta < \mathtt{max}(\hat{\boldsymbol{U}}_l)$ then
        $\eta \leftarrow \mathtt{max}(\hat{\boldsymbol{U}}_l)$
    end
    for $l$ in $\{L, L-1, ..., 1\}$ do
        Sample noise matrix $\boldsymbol{\xi}$
        $\bar{\boldsymbol{U}}_l \leftarrow \phi(\hat{\boldsymbol{U}}_l / \eta)$
        $\boldsymbol{W}_l \leftarrow \boldsymbol{W}_l - \alpha(\boldsymbol{F}_l + \boldsymbol{\xi}) \circ (1 - \bar{\boldsymbol{U}}_l)$
    end
end
```

Appendix D. UPGD on Stationary MNIST

We use the MNIST dataset to assess the performance of UPGD under stationarity. A desirable property of continual learning systems is that they should not asymptotically impose any extra performance reduction, which can be studied in a stationary task such as MNIST. We report the results in Fig. 3. We notice that UPGD improves performance over SGD. Each point in the stationary MNIST figures represents an average accuracy over a non-overlapping window of 10000 samples. The learners use a network of 300×150 units with ReLU activations. The utility traces are computed using exponential moving averages given by $\tilde{U}_t = \beta_u \tilde{U}_{t-1} + (1-\beta_u)U_t$, where $\tilde{U}_t$ is the utility trace at time t and U_t is the instantaneous utility at time t. Each learner is trained for 1 million time steps. The results are averaged over 20 independent runs.

Appendix E. Quality of the Approximated Utilities

A high-quality approximation of utility should give a similar ordering of weights to the true utility. We use the ordinal correlation measure of Spearman to quantify the quality of our utility approximations. An SGD learner with a small neural network with ReLU activations is used on a simple problem to minimize the online squared error over a total of 2000 samples.

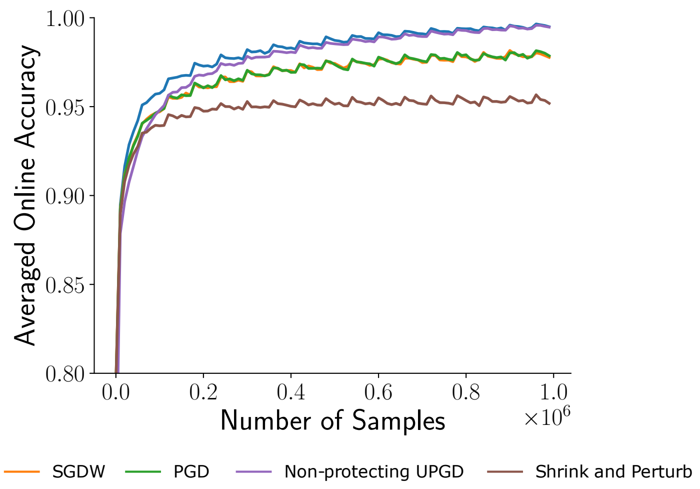


Figure 3: Performance of Utility-based Perturbed Gradient Descent with first-order approximated utilities on stationary MNIST.

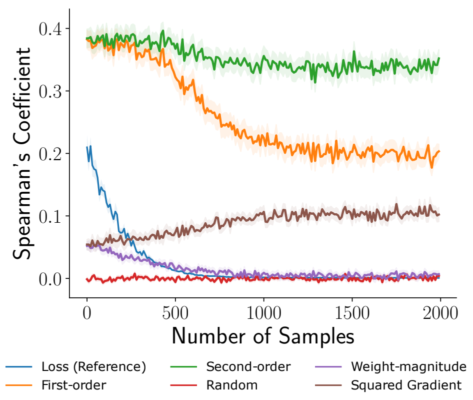


Figure 4: Rank correlation between the true utility and approximated utility.

At each time step, Spearman's correlation is calculated for first- and second-order global utility against the random ordering, the squared-gradient utility, and the weight-magnitude utility. We report the correlations between the true utility and approximated global weight utilities in Fig. 4. The correlation is the highest for the second-order utility throughout learning. On the other hand, the first-order utility becomes less correlated when the learner plateaus, likely due to zigzagging gradient elements near the solution. The weight-magnitude utility shows a small correlation to the true utility that gets smaller. The correlation of the squared-gradient utility increases with time steps but remains smaller than that of the first-order utility. We use random ordering as a baseline, which maintains zero correlation with the true utility, as expected.

Appendix F. More Diagnostic Statistics Characterizing Solution Methods

Here, we provide more diagnostic statistics for our methods Fig. 5, Fig. 6, and Fig. 7. We define the plasticity of a learner given a sample as the ability to change its prediction to match the target. The learner achieves plasticity of 1 given a sample if it can exactly match the target and achieves plasticity of 0 if it achieves zero or negative progress toward the target compared to its previous prediction given the same sample. Formally, we define the sample plasticity to be $p(Z) = \max\left(1 - \frac{\mathcal{L}(\mathcal{W}^\dagger, Z)}{\max(\mathcal{L}(\mathcal{W}, Z), \epsilon)}, 0\right) \in [0, 1]$, where $\mathcal{W}^\dagger$ is the set of weights after performing the update and ϵ is a small number to maintain numerical stability. Note that the term $\left(1 - \frac{\mathcal{L}(\mathcal{W}^\dagger, Z)}{\max(\mathcal{L}(\mathcal{W}, Z), \epsilon)}\right) \in (-\infty, 1]$ has an upper bound of 1, since $\mathcal{L}(\mathcal{W}, Z) \in [0, \infty), \forall \mathcal{W}, Z$ for cross-entropy and squared-error losses.

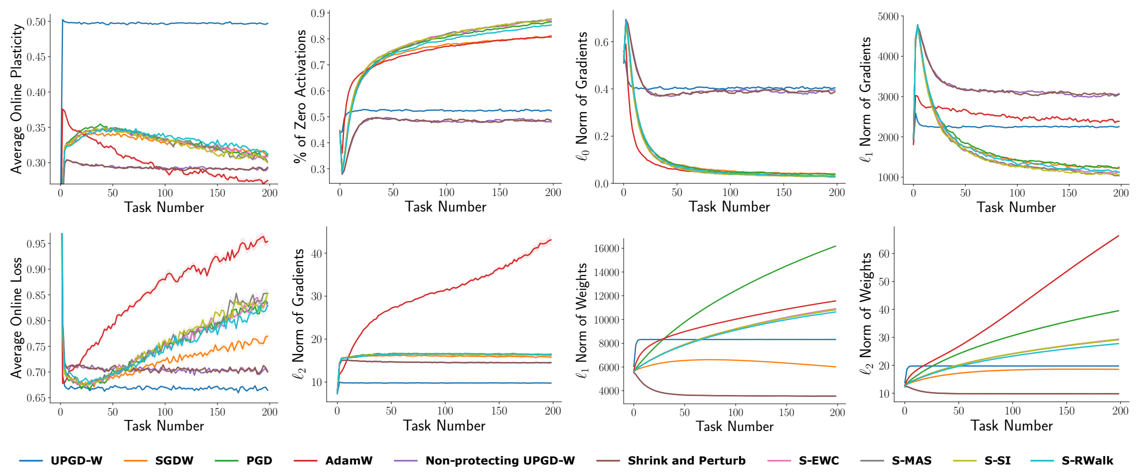


Figure 5: Diagnostic statistics of methods on Input-permuted MNIST. The shaded area represents the standard error.

Appendix G. Ablation on Components of UPGD

We conducted a short ablation study in Fig. 8 on the components of UPGD: weight decay, weight perturbation, and utility gating. Starting from SGD, we add each component step by step until we reach UPGD. Fig. 8 shows the performance of learners on Input-permuted MNIST, Label-permuted EMNIST, and Label-permuted mini-ImageNet.

We notice that both weight perturbation and weight decay improve SGD performance. Still, the role of weight decay seems to be more important in Input-permuted MNIST and Label-permuted mini-ImageNet. Notably, the combination of weight decay and weight perturbation makes the learner maintain its performance. When utility gating is added on top of weight decay and weight perturbation, the learner can improve its performance continually in all label-permuted problems and slightly improve its performance on input-permuted.

We also conduct an additional ablation in Fig. 9 where we start from UPGD-W and remove each component individually. This ablation bolsters the contribution of utility gating more. Using utility gating on top of SGD makes SGD maintain its performance instead of dropping on input-permuted

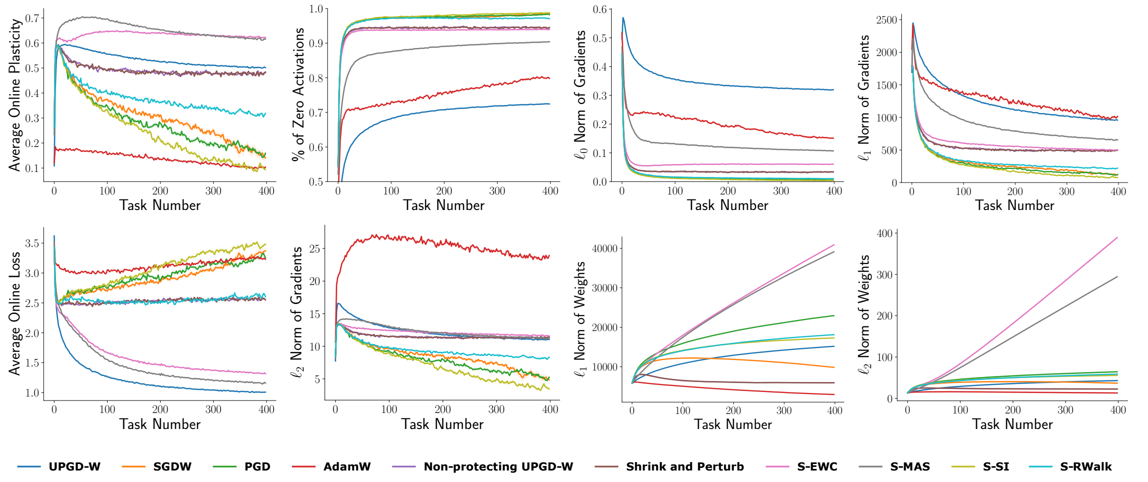


Figure 6: Diagnostic statistics of methods on Label-permuted EMNIST. The shaded area represents the standard error.

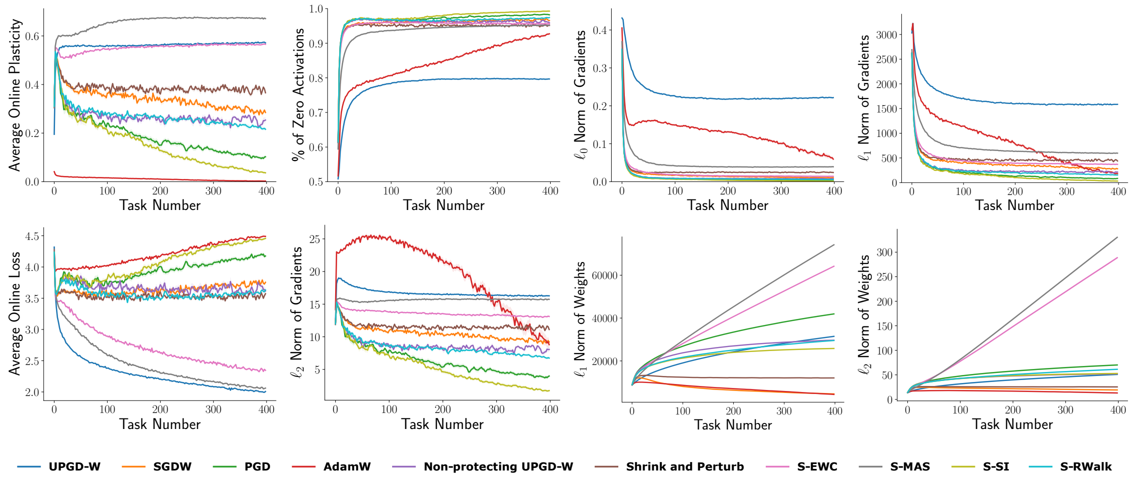


Figure 7: Diagnostic Statistics of methods on Label-permuted *mini*-ImageNet. The shaded area represents the standard error.

MNIST and makes SGD improve its performance continually on label-permuted problems. The role of weight decay and weight perturbation is not significant in label-permuted problems, but including both with utility gating improves performance and plasticity on input-permuted MNIST.

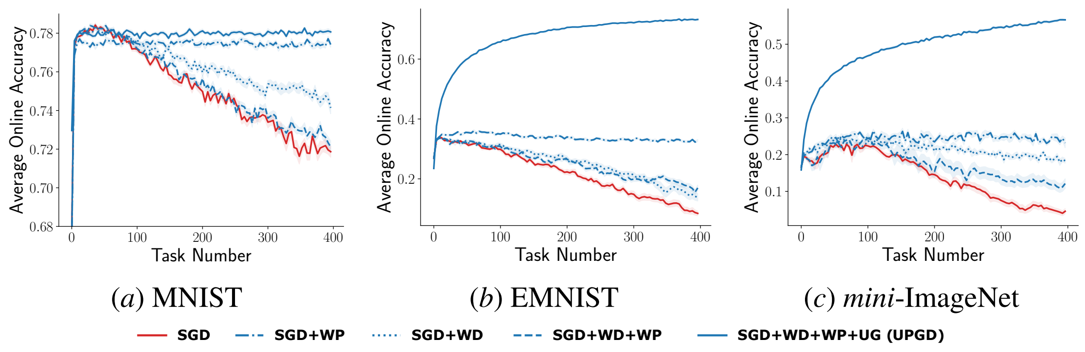


Figure 8: Ablation on the components of UPGD: Weight Decay (WD), Weight Perturbation (WP), and Utility Gating (UG) shown on Input-permuted MNIST, Label-permuted EMNIST, and Label-permuted *mini*-ImageNet.

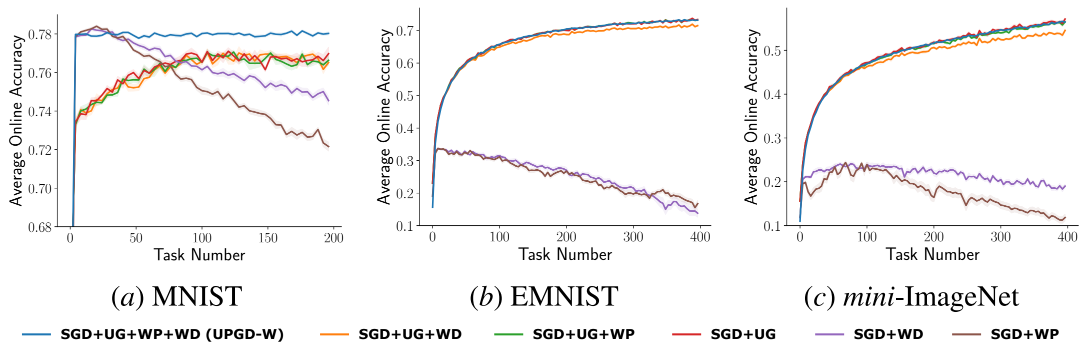


Figure 9: Ablation on the components of UPGD: Weight Decay (WD), Weight Perturbation (WP), and Utility Gating (UG) shown on Input-permuted MNIST, Label-permuted EMNIST, and Label-permuted *mini*-ImageNet, starting from UPGD-W and removing each component individually. SGD+WD and SGD+WP are added as baselines that do not use utility gating. A global first-order utility trace is used. Results are averaged over 10 runs. The shaded area represents the standard error.