Introducing utility-based Stochastic Gradient Descent with Adaptive Noise Injection

Constantin Pinkl   Laura Schulz   Tilman de Lanversin   Yumi Kim

Abstract

Online continual learning presents unique challenges in real-time adaptability to sequential data, particularly catastrophic forgetting and loss of plasticity. Utility-based Perturbed Gradient Descent (UPGD) addresses these issues but is limited by standard Gaussian noise injection and neuron-level utility evaluation. In this work, we propose enhancing UPGD with layer-wise noise scaling and kernel-level utility evaluation to better capture parameter variations and neuron interactions. Experiments show that our model, Utility-based Stochastic Gradient Descent with Adaptive Noise (`usdg`), outperforms UPGD, achieving higher average accuracy and plasticity while reducing forgetting. In contrast, kernel-based utility methods show limited effectiveness.

1. Introduction

Online continual learning adapts to sequential data in real-time, processing it in a single pass with minimal access to past data. This further complicates the two predominant challenges in continual learning: catastrophic forgetting (CF), where old knowledge is lost while learning new tasks, and the loss of plasticity (LoP), which reduces adaptability to new information. Utility-based Perturbed Gradient Descent (UPGD) has shown potential in addressing both CF and LoP in online continual learning, achieving a more balanced learning performance across tasks [4]. However, it applies standard Gaussian noise across layers and evaluates utility at the neuron level, neglecting parameter variation and neuron group behavior (i.e. CNN kernels).

This work is based on the UPGD method and addresses its challenges by making three key contributions, which we have made publicly available on GitHub.

1. Introduce layer-wise noise scaling to account for parameter heterogeneity, validated by measurable improvements in accuracy and plasticity.
2. Pioneer in kernel-level utility evaluation, a novel perspective on capturing collective neuron behavior in continual learning.
3. Provide a detailed experimental analysis to aid in refining these techniques for future research.

The results highlight the potential of the proposed approaches and the idea of injecting layer-wise adaptive noise. Especially Utility-based SGD with Adaptive Noise [1] outperforms UPGD, achieving higher average accuracy (55.38% vs. 55.29%) and average plasticity (45.52% vs. 41.86%). While exploratory, these methods show strong potential to address CF and LoP, paving the way for robust and adaptive solutions in dynamic learning environments.

2. Related Work

Continual learning has seen many advancements in recent years, but the online setting introduces unique challenges. While techniques like experience replay and regularization-based methods have addressed CF in broader continual learning [13, 15], they are often impractical in streaming scenarios due to memory and computational demands [6]. Online EWC [14] is an continuation of the EWC algorithm [7] for the online setting, but lacking focus on maintaining plasticity. Plasticity preservation receives less attention, but is another critical issue, especially in streaming contexts, where models must adapt to evolving data distributions. Recent methods like Adaptive Plasticity Improvement (API) [11] and continual backpropagation [3] have demonstrated the potential to address these challenges.

The literature proposes injecting noise into the gradient as a mechanism to help explore diverse solutions in the loss landscape. This helps in escaping local minima and improving plasticity, or robustness [3]. Approaches include Stochastic Gradient Descent (SGD) [2] which introduces inherent randomness, and Perturbed Gradient Descent (PGD) which utilizes perturbations to achieve these specific objectives [18]. Some approaches have investigated rescaling the noise for each layer based on characteristics such as parameter norms or gradient magnitudes, tailoring the perturbations to better suit each layer's role in the network [17].

The UPGD algorithm from [5] [2] introduces an innovative

[1]Denoted as `usgd` for brevity

[2]The UPGD algorithm from Elsayed et al. (2024) will be referenced as UPGD without repeated citations for brevity.

approach by perturbing gradients based on a utility function, with the aim of simultaneously addressing the challenges of CF and LoP in the continuous online learning setting [4].

Specifically, the gradient descent equation for UPGD is

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

where the weights $w_{l,i,j}$ are iteratively updated to minimize the loss function $\mathcal{L}$. The term $\frac{\partial \mathcal{L}}{\partial w_{l,i,j}}$ represents the gradient of the loss with respect to the weights, guiding the direction of the update, while the noise term, ξ, is sampled from a standard Gaussian distribution, $\xi \sim \mathcal{N}(0,1)$ for all layers. The utility $U_{i,j}$ of a neuron is calculated by approximating the change of loss when omitting this neuron. This utility function evaluates the importance of each neuron, perturbing and resetting neurons with low utility while preserving the activations of neurons with high utility.

3. Visualizations

Using a subset of the CIFAR-10 dataset, we performed a series of visualizations to gain insights into UPGD algorithm and the utility-based kernel dynamics.

The initial visualization mapped utility values across the kernel space, thereby unveiling patterns and opportunities for subarea averaging. These findings underscored the diverse roles of kernel regions in model functionality.

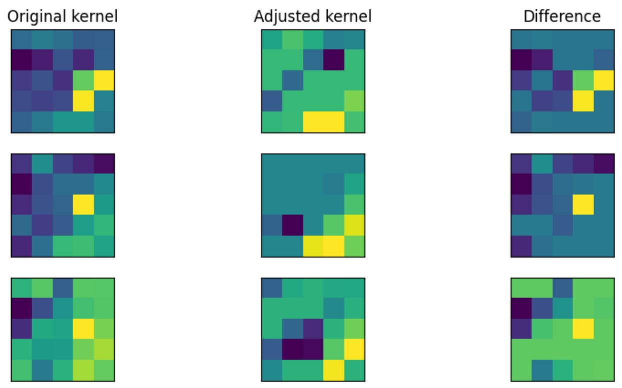


Figure 1. First convolutional kernel

Inspired by Wang et al. (2021), we explored the utility of convolutional kernels by analyzing their effect on input images. We conducted a forward pass through the first convolutional layer and compared the outputs to variations of the original layer to evaluate how utility values influence kernel behavior. Additionally, the kernels were scaled by their utility values to assess their importance within the layer. The outputs resembled the original images, albeit scaled down, thereby confirming that the utility mechanism preserves key contributions while suppressing irrelevant ones.High-utility kernels retained essential input features.

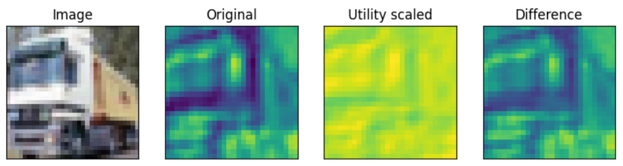


Figure 2. Applied convolutions on input image

The final visualization examined utility values by layer, showing a strong link between high-utility regions and significant feature activations. This highlights the importance of these layers in model performance and supports deeper integration of utility-based adjustments for improved feature extraction.

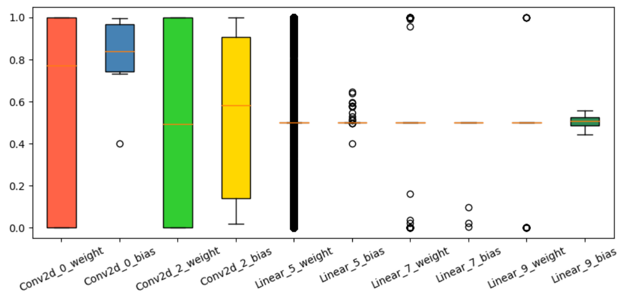


Figure 3. Raw utilite values of the model

4. Methodology

In this section, we describe the two key areas that we have identified for potentially improving the UPGD method: Enhancing noise injection by accounting for layer-wise parameter norms and refining the utility function by considering kernel-level importance in addition to individual neuron utilities. Additionally, inspired by the effectiveness of layer-wise noise scaling, we propose incorporating stochastic gradient descent for further exploration of parameter space.

4.1. Layer-wise Noise Scaling

To address variations in parameter norms between layers, we propose scaling noise injection explicitly at the layer level using different strategies for adaptive noise scaling. The gradient descent update is given by:

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

where $S_{l,i,j}$ represents the adaptive noise scaling factor, defined as follows for each strategy:

1. **Gradient Norm (`grad_norm`):** Scaling by the L_2 norm of the gradients $GN_{l,i,j}$

$$S_{l,i,j} = \left\| \frac{\partial \mathcal{L}}{\partial w_{l,i,j}} \right\|_2 .$$

   This approach ensures that the perturbation is influenced by the gradient magnitude, potentially aligning it with the optimization trajectory.

2. **Weight Norm (`weight_norm`)** $WN_{l,i,j}$: Scaling by the L_2 norm of the weights in each layer

$$S_{l,i,j} = \|w_{l,i,j}\|_2.$$

3. **Gradient-to-Weight Ratio (`ratio_norm`):** $RN_{l,i,j}$ Scaling by the ratio of the gradient norm to the weight norm:

$$S_{l,i,j} = \frac{\left\|\frac{\partial \mathcal{L}}{\partial w_{l,i,j}}\right\|_2}{\|w_{l,i,j}\|_2}$$

This approach draws inspiration from the Layer-wise Adaptive Rate Scaling (LARS) method [17], which employs a layer-wise learning rate based on the ratio of gradient norm to weight norm. By adopting a similar layer-wise strategy, we hope to enhance stability and adaptability in training dynamics.

4.2. Utility-based Stochastic Gradient Descent with Adaptive Noise (`usgd`)

Building on the success of layer-wise noise scaling, we propose replacing the perturbed gradient descent with stochastic gradient descent (SGD) together with `ratio_norm`:

$$w_{l,i,j} \leftarrow w_{l,i,j} - \alpha \left(\frac{\partial \mathcal{L}}{\partial w_{l,i,j}} \left(1 - \bar{U}_{l,i,j}\right) + \frac{\left\|\frac{\partial \mathcal{L}}{\partial w_{l,j}}\right\|_2}{\|w_{l,i,j}\|_2} \cdot \xi \right)$$

This is motivated by the stochastic nature of SGD, which may help the model escape local minima and explore a wider range of solutions in the loss landscape than PGD's fixed perturbations. This could be particularly beneficial in preserving the plasticity of the model.

4.3. Kernel utility

In UPGD, utility is evaluated at the neuron level, but in CNNs, features are often identified collectively by neurons within a kernel. To address this, we propose incorporating kernel-level importance into utility evaluation. Our investigation into kernel importance aligns with concepts in kernel pruning methods, which similarly emphasize evaluating the relevance of kernels within the network and removing those deemed less important [10]. We focus on three approaches as a starting point to determine the effectiveness in guiding the re-initialization of less relevant areas to capture meaningful features:

1. **Entire Kernel Evaluation (`entire_kernel`):** The kernel utility U_K is computed as the mean utility of all its parameters:

$$U_K = \frac{1}{m \cdot n} \sum_{i=1}^{m} \sum_{j=1}^{n} U_{i,j}, \tag{1}$$

where m and n are the kernel dimensions and utility matrix U. This utility metric is applied per neuron as proposed by the UPGD algorithm.

2. **Column-wise Kernel Evaluation (`column_kernel`):** In CNNs the primary responsibility of kernels is to detect edges. Therefore, we tailor our utility unification approach accordingly and instead of considering the utility of all parameters in the kernel. We focus specifically on columns and compute in a the same manner to `entire_kernel`.

5. Experimental Setup

To evaluate the efficacy of the proposed methods, we use the main UPGD algorithm, specifically Algorithm 1 from the original paper, as our baseline, and used the same experimental setup with a fixed seed. To extract the effectiveness of the algorithms on catastrophic forgetting and loss plasticity we conducted two experiments. First, we run a continuous learning experiment based on the CIFAR-10 dataset [9] for 400 tasks. The labels are permuted at regular intervals of k steps to assess the effectiveness our methods in preventing catastrophic forgetting.
The effect of loss of plasticity can be exacerbated in experiments through the introduction of non-stationarity in the dataset [12]. Therefore, we utilized an input permuted MNIST [1] dataset with 200 tasks for our second experiment, where we permute the input every k steps; each k-step window of a single permutation is referred to as an episode.

5.1. Metrics

Elsayed et al. [4] introduced metrics to evaluate catastrophic forgetting (CF) and plasticity in online learning. CF is measured as the difference between final and initial accuracy, with forgetting at each step computed as:

$$F_{k+1} = A_k - A_{k+1} \in [-1, 1],$$

where negative values indicate improvement and positive values denote forgetting.

Plasticity is assessed per training step as:

$$p(Z) = \max\left(1 - \frac{\mathcal{L}(W^\dagger, Z)}{\max(\mathcal{L}(W, Z), \epsilon)}, 0\right) \in [0, 1],$$

where $W^\dagger$ and W are the weights before and after the update, respectively. The plasticity of an entire episode $\bar{p}_k$ is calculated as the mean of all k-samples in the episode. To track the development of plasticity over time the metric compares to adjacent episodes $\Delta\bar{p}_{k+1} = \bar{p}_k - \bar{p}_{k+1}$, with the loss of plasticity metric being computed as the difference between the final and the initial plasticity.

Lastly, a sufficiently smooth loss function is strongly associated with improved convergence properties [8]. Therefore, we employ the Lipschitz constant as an additional metric to assess the smoothness of the loss function.

6. Experimental Results and Discussion

6.1. Results

While developing Utility-based Stochastic Descend with Adaptive Noise (`usgd`), we experimented with several approaches. Tables 1 and 2 summarizes the quantitative metrics and Figures 4 and 5 show the performance on an online continuous learning stream.

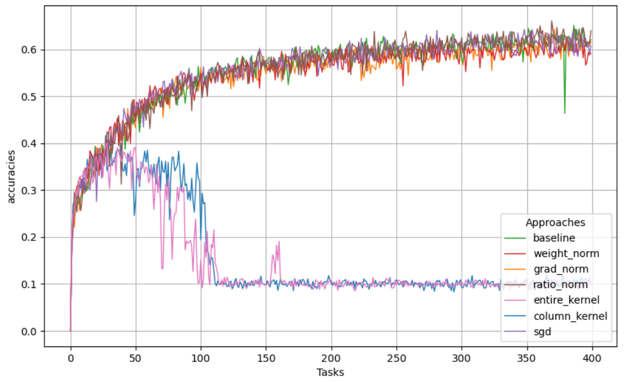


Figure 4. Accuracy of all approaches from CIFAR-10 experiment

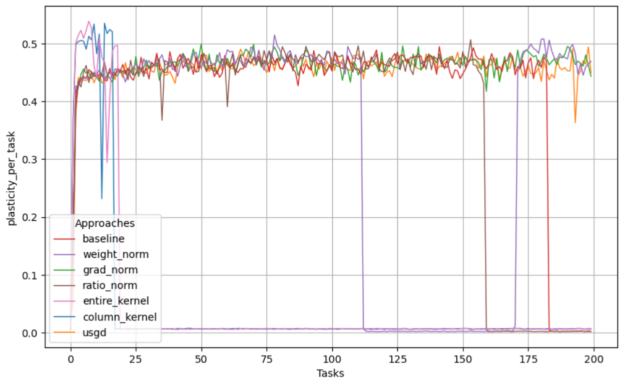


Figure 5. Plasticity of all approaches from MNIST experiment

The key observations are that the kernel-based approaches fail early in both accuracy and plasticity, while all adaptive noise-based approaches show strong results. In contrast, adaptive noise-based approaches consistently outperform the baseline, with `usgd` showing the most robust and consistent performance across metrics The results also highlight the trade-offs between catastrophic forgetting (CF) and loss of plasticity (LoP): While `ratio_norm` effectively prevents CF, it collapses in plasticity, whereas `grad_norm` prevents LoP well but underperforms in the forgetting experiment. Interestingly, `weight_norm` also collapses in the plasticity experiment but manages to recover, leaving open the possibility that other approaches could similarly recover given sufficient training on more tasks.

Approach	**Avg. Accuracy**	**CF**	**Lipschitz CF**
`baseline`	0.5529	-0.6204	0.4732
`weight_norm`	0.5462	-0.5920	**0.1991**
`grad_norm`	0.5427	-0.6160	0.2006
`ratio_norm`	0.5530	**-0.6400**	0.2515
`usgd`	**0.5538**	-0.6076	0.2820
`entire_kernel`	0.1529	-0.0996	0.2700
`column_kernel`	0.1614	-0.1000	0.2717

Table 1. Performance metrics related to catastrophic forgetting (CF)

Approach	**Avg. Plasticity**	**LoP**	**Lipschitz LoP**
`baseline`	0.4186	0.0045	1.2917
`weight_norm`	0.3277	**-0.4570**	1.4502
`grad_norm`	**0.4608**	-0.4393	**0.9892**
`ratio_norm`	0.3654	0.0047	1.3320
`usgd`	0.4552	-0.4374	1.2344
`entire_kernel`	0.0488	0.0036	1.1360
`column_kernel`	0.0451	0.0083	1.2494

Table 2. Performance metrics related to loss of plasticity (LoP)

6.2. Discussion

Our findings highlight the critical role of randomness and adaptive noise scaling in enhancing model performance. Increased randomness, as introduced by `usgd`, improves both accuracy and plasticity by enabling the model to escape local minima and explore a broader range of solutions. Howver, excessive randomness could destabilize optimization, remaining a key-area for further exploration.

The failure of kernel-based approaches indicates that utility evaluation at the kernel level may be too coarse, overlooking important interactions between neurons.

Overall, the trade-offs between mitigating CF and maintaining LoP are evident. `usgd` emerges as the most consistent and effective approach, but further experiments, such as testing with additional random seeds, are necessary to confirm robustness across varying conditions.

7. Conclusion

This study highlights the potential of adaptive noise injection techniques in addressing catastrophic forgetting and loss of plasticity in online continual learning. The proposed methods, particularly `usgd`, demonstrate notable improvements over the baseline UPGD algorithm, achieving higher accuracy and plasticity while maintaining stability. Although kernel-based utility evaluation showed limited success, the insights gained suggest promising directions for further refinement. Overall, these findings emphasize the importance of adaptive approaches in dynamic learning environments and provide a foundation for developing more robust and scalable continual learning solutions.

References

[1] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. doi: 10.1109/MSP.2012.2211477.

[2] Shibhansh Dohare, Richard S Sutton, and A Rupam Mahmood. Continual backprop: Stochastic gradient descent with persistent randomness. *arXiv preprint arXiv:2108.06325*, 2021.

[3] Shibhansh Dohare, J Fernando Hernandez-Garcia, Parash Rahman, A Rupam Mahmood, and Richard S Sutton. Maintaining plasticity in deep continual learning. *arXiv preprint arXiv:2306.13812*, 2023.

[4] Mohamed Elsayed and A Rupam Mahmood. Addressing loss of plasticity and catastrophic forgetting in continual learning. *arXiv preprint arXiv:2404.00781*, 2024.

[5] Mohamed Elsayed and A. Rupam Mahmood. Addressing Loss of Plasticity and Catastrophic Forgetting in Continual Learning, April 2024. URL http://arxiv.org/abs/2404.00781. arXiv:2404.00781 [cs].

[6] Tyler L Hayes, Nathan D Cahill, and Christopher Kanan. Memory efficient experience replay for streaming learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9769–9776. IEEE, 2019.

[7] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[8] Nikita Kiselev and Andrey Grabovoy. Unraveling the hessian: A key to smooth convergence in loss function landscapes, 2024. URL https://arxiv.org/abs/2409.11995.

[9] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[10] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets, 2017. URL https://arxiv.org/abs/1608.08710.

[11] Yan-Shuo Liang and Wu-Jun Li. Adaptive Plasticity Improvement for Continual Learning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7816–7825, Vancouver, BC, Canada, June 2023. IEEE. ISBN 9798350301298. doi: 10.1109/CVPR52729.2023.00755. URL https://ieeexplore.ieee.org/document/10204088/.

[12] Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks, 2023. URL https://arxiv.org/abs/2303.01486.

[13] Haoxuan Qu, Hossein Rahmani, Li Xu, Bryan Williams, and Jun Liu. Recent advances of continual learning in computer vision: An overview, 2024. URL https://arxiv.org/abs/2109.11369.

[14] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International conference on machine learning*, pages 4528–4537. PMLR, 2018.

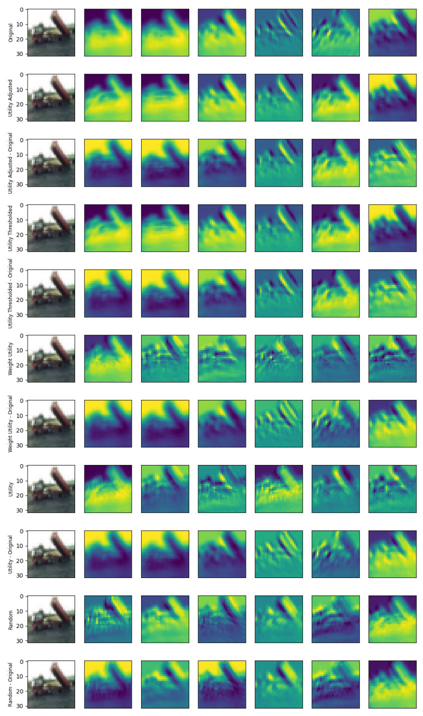
[15] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[16] Zijie J. Wang, Robert Turko, Omar Shaikh, Haekyu Park, Nilaksh Das, Fred Hohman, Minsuk Kahng, and Duen Horng Polo Chau. Cnn explainer: Learning convolutional neural networks with interactive visualization. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1396–1406, 2021. doi: 10.1109/TVCG.2020.3030418.

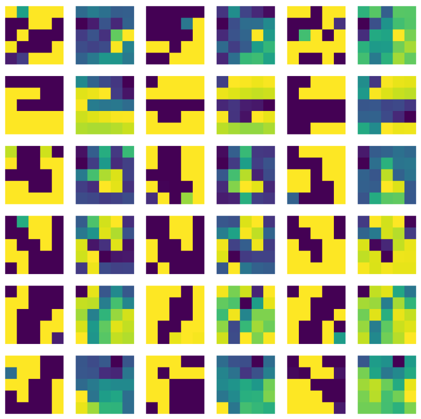
[17] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks, 2017. URL https://arxiv.org/abs/1708.03888.

[18] Mo Zhou, Tianyi Liu, Yan Li, Dachao Lin, Enlu Zhou, and Tuo Zhao. Towards understanding the importance of noise in training neural networks, 2019. URL https://arxiv.org/abs/1909.03172.

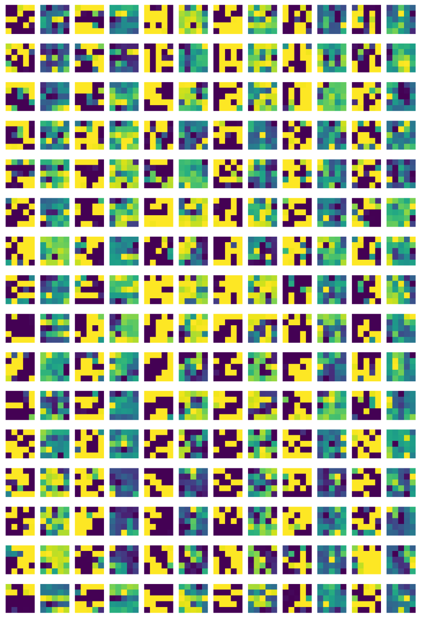
A. Further Visualizations



(a) Different alternatives of the first layer convolutional kernel



(b) Full visualization of utilities if the kernels of the first layer



(c) Full visualization of utilities of the kernels of the second layer

Figure 6. Side-by-side visualization of layer kernels and forward passes

In Figure 6a, the visualizations are displayed from top to bottom as follows:

1. The original layer.
2. The layer multiplied by the utility value.
3. The difference between the original layer and the utility-multiplied layer.
4. Kernel values thresholded by utility.
5. The difference between the original and the thresholded kernels.
6. Kernels squared to represent weight utility.
7. The difference between the original and squared kernels.
8. Kernels showing only the utility values.
9. The difference between the original and utility-only kernels.
10. Random multiplication of the original kernel.
11. The difference between the original kernel and the randomly multiplied kernel.

B. Further Experimental Results

To gain deeper insights into the performance of our approaches, we analyzed zoomed-in graphs of the last 200 tasks to see how they compare. Figures 7 illustrate this comparison for the `usgd` and `ratio_norm`, the best appraoches regarding accuracy and catastrophic forgetting. Notably, the original UPGD algorithm (`basecase`) experiences some sudden drops with recoveries in both accuracy and plasticity. In contrast, `ratio_norm` shows a steady upward trend in accuracy, indicating good potential for handling even longer task sequences.

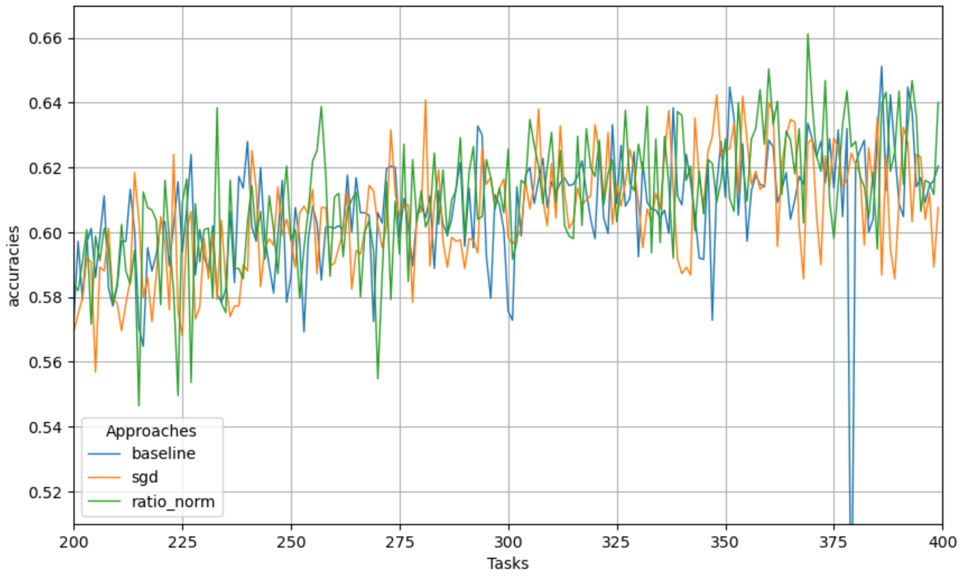
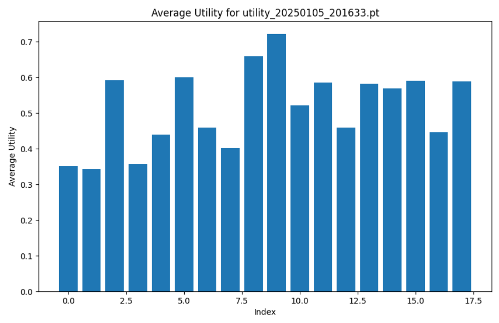


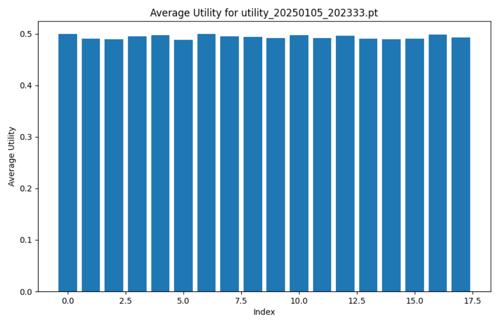
Figure 7. Zoomed in accuracy of two best approaches (`usgd` and `ratio_norm`) vs baseline on the last 200 tasks

C. Entire Kernel Learner

To explain the behavior of the Entire Kernel Learner we analyzed the development of the utility metric over the course of the continual learning experiment. The development of the plasticity metric 6.1 suggests that the model looses all ability to learn after k steps. The development of the utility of each kernel explains this behavior seen in figure 8. After a few episodes the utility of every kernel is averages at 0.5 with very little variance as the training continues. This hints where the collapse of the model originates from. Since we treat in every step every kernel now with the roughly same utility value, it hinders the model in the long run to continue learning.



(a) Kernel Utility of all kernels in one convolutional layer before starting to train.



(b) Kernel Utility of all kernels in one convolutional layer after a few episodes.

Figure 8. Utilities of the Kernels in our CIFAR-10 Label Permuted experiment.