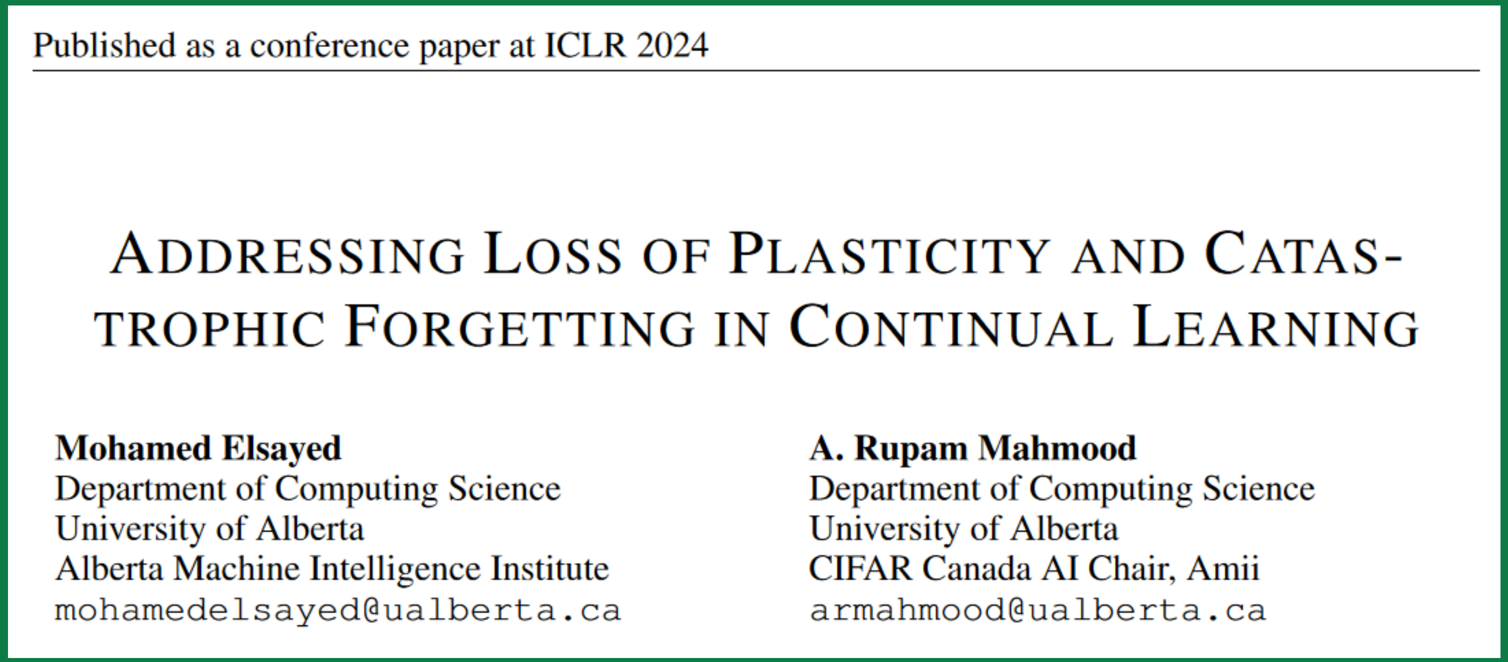
Addressing Loss of Plasticity and Catastrophic Forgetting in Continual Learning

GIST AIGS/EECS

INFONET, Sumin Kim

smkim6927@gm.gist.ac.kr



Published as a conference paper at ICLR 2024

ADDRESSING LOSS OF PLASTICITY AND CATASTROPHIC FORGETTING IN CONTINUAL LEARNING

Mohamed Elsayed
Department of Computing Science
University of Alberta
Alberta Machine Intelligence Institute
mohamedelsayed@ualberta.ca

A. Rupam Mahmood
Department of Computing Science
University of Alberta
CIFAR Canada AI Chair, Amii
armahmood@ualberta.ca

The reason for choosing this paper and the Abstract of this paper

My research topic is Continual Pre-training.

1. I chose this paper because my research topic is how to prevent catastrophic forgetting even when learning multiple domains with continual pre-training in LM. And Continual Learning is a more big topic than continual pre-training.
2. In the research area of artificial intelligence, catastrophic forgetting in neural networks is widely recognized as a major challenge. Another pernicious challenge of continual learning is the loss of plasticity, where the learner's ability to learn new things diminishes.
3. This paper handles a catastrophic forgetting problem and loss of plasticity. These topics are the reason we study continual learning.
4. In this paper, they 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.
    - Make smaller updates to the more important units to prevent them from being forgotten.
    - Making larger updates to less useful units to restore their plasticity, allowing the model to continue learning effectively.

INDEX

Challenges and Related Works

1. Addressing Catastrophic Forgetting
2. Addressing Loss of Plasticity
3. Addressing Both Issues
4. Importance Measure in Neural Network Pruning

Method

1. Scalable Approximation of the True Utility
2. Utility-Based Perturbed Gradient Descent (UPGD)
3. Forgetting and Plasticity Evaluation Metrics

Experiments

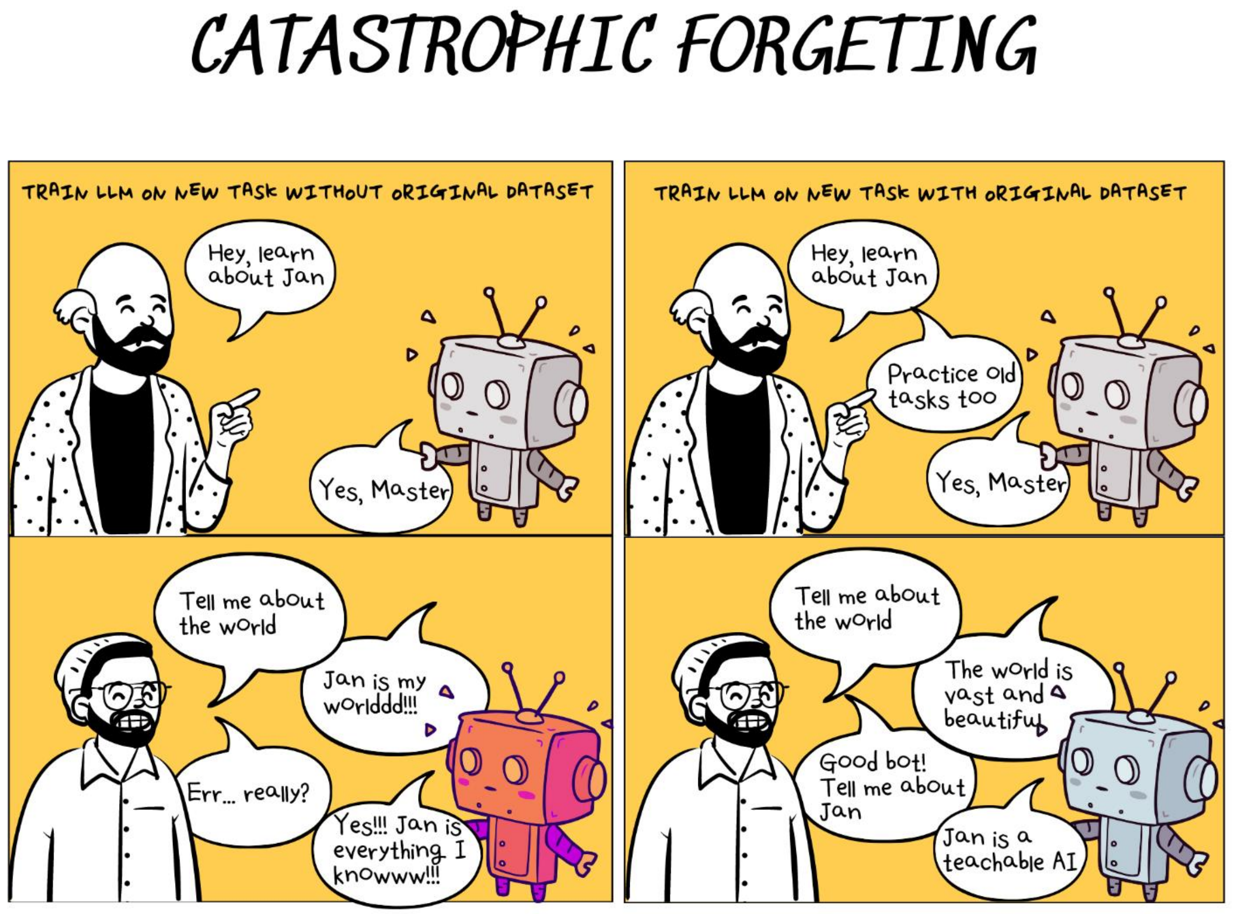
1. Quality of the Approximated Utilities
2. UPGD against Loss of Plasticity
3. UPGD against Catastrophic Forgetting
4. UPGD against Loss of Plasticity and Catastrophic Forgetting

Conclusion

Challenges and Related Works

Challenges and Related Works (1)

Addressing Catastrophic Forgetting



Challenges

1. Catastrophic forgetting is widely recognized as a major challenge of continual learning (de Lange et al. 2021). The phenomenon is clear and evident 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).
2. This issue also raises **a concern for reusing large practical models**, where finetuning them for new tasks causes significant forgetting of pre-trained models (Chen et al. 2020, He et al. 2021).

Addressing ways

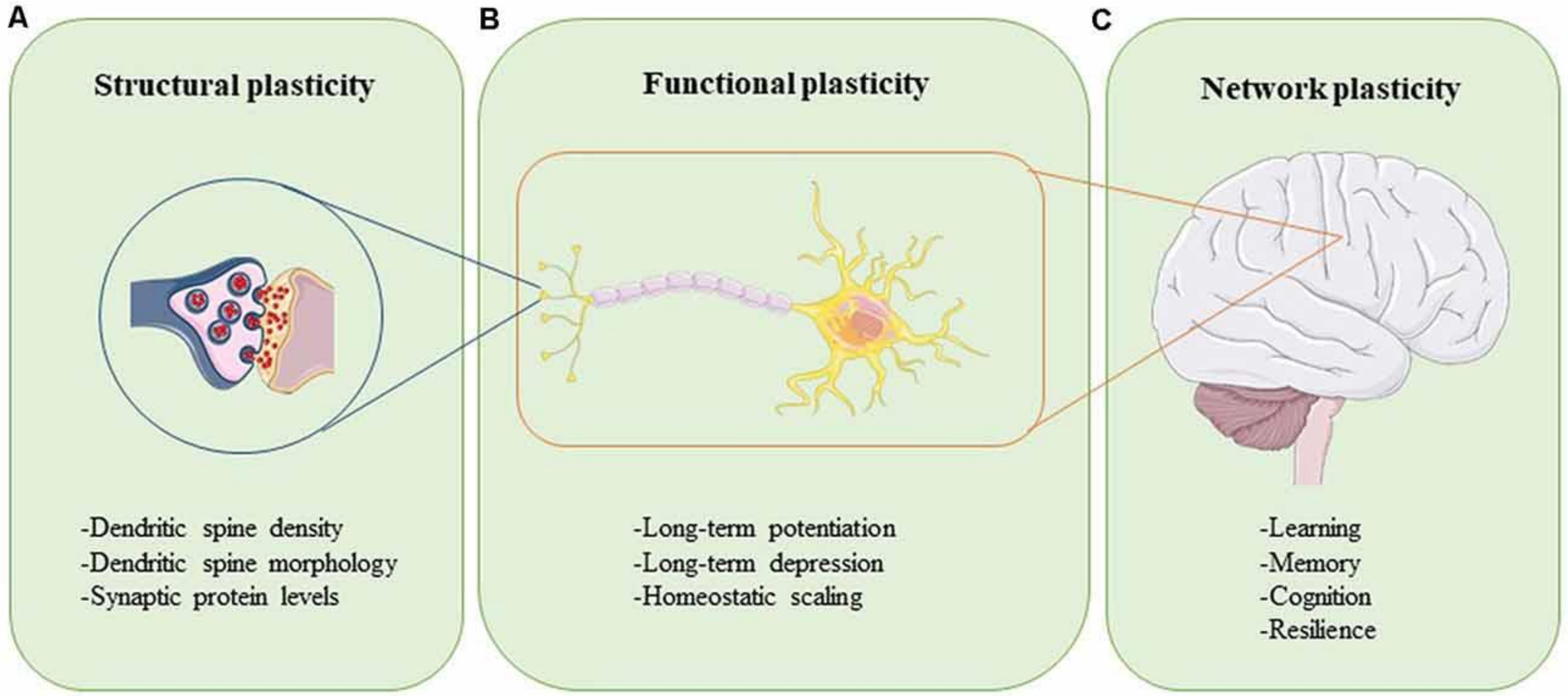
1. Replay-based methods (e.g., Chaudhry et al. 2019, Isele & Cosgun 2018, Rolnick et al. 2019) address forgetting by using a replay buffer to store incoming non-i.i.d. data and then sample from the buffer i.i.d. samples.
2. Parameter isolation methods (e.g., Rusu et al. 2016, Schwarz et al. 2018, Lee et al. 2019, Wortsman et al. 2020, Ge et al. 2023) that can expand to accommodate new information without significantly affecting previously learned knowledge.
3. 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.
4. 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.

Challenges and Related Works (2)

Addressing Loss of Plasticity

Challenges

1. Yet another pernicious challenge of continual learning is loss of plasticity, where the learner's ability to learn new things diminishes.
2. 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).



Addressing ways

1. Dohare et al. (2023) introduced **a generate-and-test method** (cf., Mahmood & Sutton 2013) 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.
2. Later, several methods were presented to retain plasticity. For example, Nikishin et al. (2023) proposed dynamically **expanding the network**, Abbas et al. (2023) recommended **adapted activation functions**, and Kumar et al. (2023) **proposed regularization** toward initial weights.

Challenges and Related Works (3)

Addressing Both Issues

Addressing ways

1. The trade-off between plasticity and forgetting, known as the stability-plasticity dilemma, was first described by Carpenter & Grossberg in 1987. This dilemma involves balancing the need to maintain performance on previously learned experiences while also adapting to new ones. Historically, the continual learning community has primarily focused on improving stability by mitigating forgetting.

2. Recently, however, there has been a shift towards methods that address both plasticity and forgetting simultaneously. For example, Chaudhry et al. (2018) introduced **RWalk**, which employs **a regularization-based approach** that adapts quickly to changes in weight importance, balancing the emphasis on past and present tasks. Jung et al. (2022) proposed techniques such as **structure-wise distillation loss and pretraining** to achieve a similar balance. Gurbuz et al. (2022) explored connection rewiring to enhance plasticity in sparse neural networks.

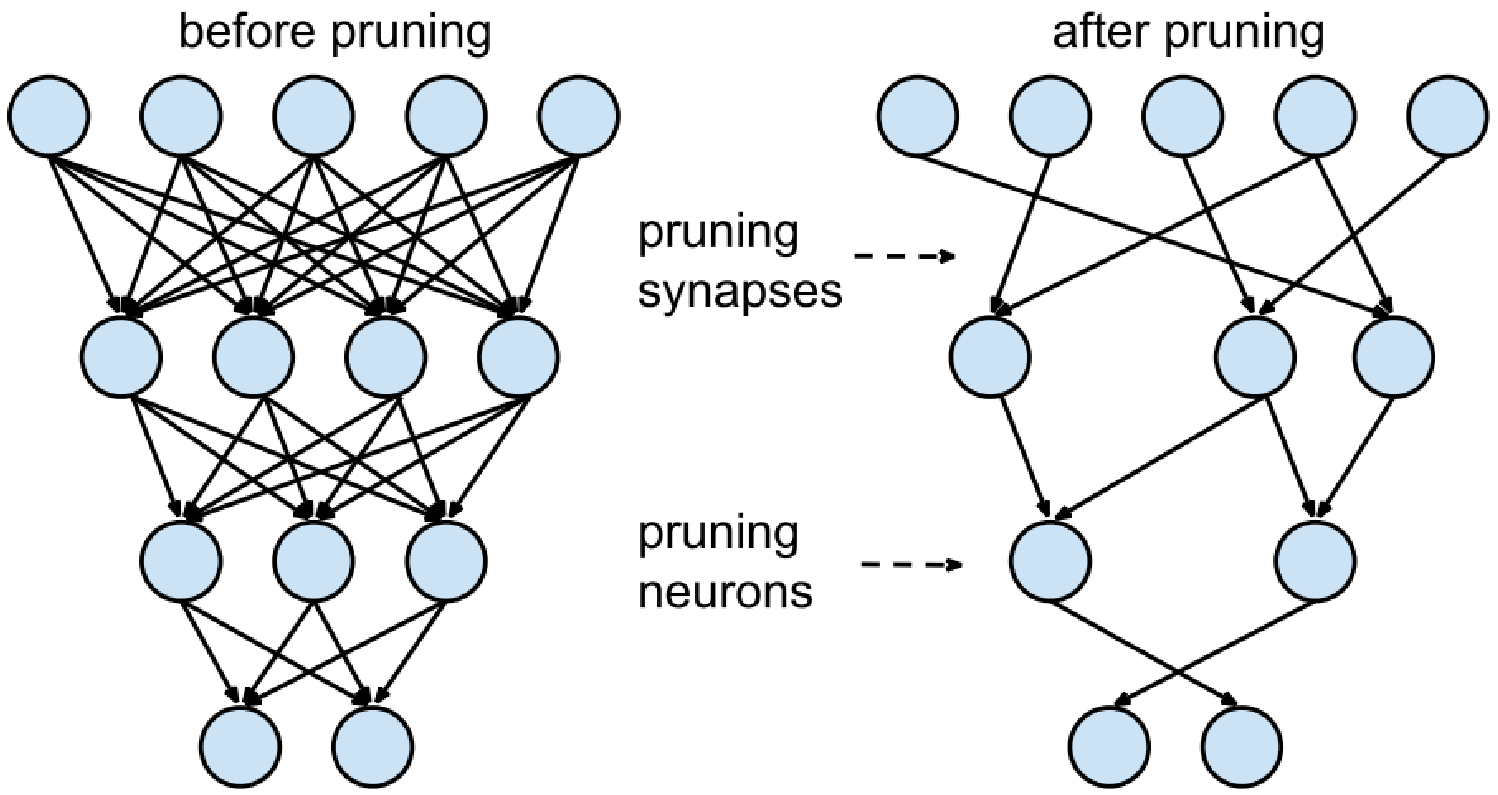
3. Additionally, Kim et al. (2023) suggested using a separate network for learning new tasks, which can later be **consolidated** into a second network for previous tasks. Despite these advancements, most existing methods still struggle in streaming learning contexts, as they often require knowledge of task boundaries, replay buffers, or pretraining processes.

Challenges and Related Works (3)

Importance Measure in Neural Network Pruning

Addressing ways

1. The Pruning in neural networks requires an important metric to determine which weights to remove.
2. Typically, the network is pruned using different measures such as the weight magnitude (e.g., Han et al. 2015, Park et al. 2020), first-order information (e.g., Mozer & Smolensky 1988, Hassibi & Stork 1992, Molchanov et al. 2016), second-order information (e.g., LeCun et al. 1989, Dong et al. 2017), or both (e.g., Tresp et al. 1996, Molchanov et al. 2019).
3. Similar to pruning, regularization-based methods that address catastrophic forgetting use weight-importance measures such as the Fisher information diagonals to weigh their penalties.



Method

Method (0)

Scalable Approximation of the True Utility

Weight utility can be defined as the change in loss when setting the weight to zero, essentially removing its connection (Mozer & Smolensky 1988, Karnin 1990).

Notation

1. $L(W,\ Z)$: The loss function, where Z represents the sample input-output pair, and W is the set of weights.
2. $W_{\neg\{l,i,j\}}$: The weight set where the specific weight $W_{\{l,i,j\}}$ Utility is defined as the change in the loss function when a particular weight is set to zero(0). Each layer's activation function σ, weight matrix W, and the forward propagation process contribute to computing the output.
3. a_{l} : Activation input to layer l before applying the activation function.
4. $h_l = \sigma(a_l)$: Activation output at layer l, where σ is the activation function (e.g., ReLU, softmax).
5. $U_{\{l,i,j\}}(Z)$: Utility of weight $W_{\{l,i,j\}}$ at layer l for sample Z, defined as the difference between the loss with and without the weight.
6. $L\left(W_{\neg\{l,i,j\}}, Z\right)$ is a counterfactual loss where $W_{\neg\{l,i,j\}}$ is the same as W except the weight $W_{\{l,i,j\}}$ is set to 0.

$$U_{\{l,i,j\}}(Z) \doteq L\left(W_{\neg\{l,i,j\}}, Z\right) - L(W, Z)$$

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.

Method (1)

Utility–Based Perturbed Gradient Descent (UPGD)

$$\begin{aligned} U_{l,i,j}(Z) &= \mathcal{L}(\mathcal{W}_{\neg[l,i,j]}, Z) - \mathcal{L}(\mathcal{W}, Z) \\ &\approx \mathcal{L}(\mathcal{W}, Z) + \frac{\partial \mathcal{L}(\mathcal{W}, Z)}{\partial W_{l,i,j}}(0 - W_{l,i,j}) + \frac{1}{2}\frac{\partial^2 \mathcal{L}}{\partial W_{l,ij}^2}(0 - W_{l,i,j})^2 - \mathcal{L}(\mathcal{W}, Z) \\ &= -\frac{\partial \mathcal{L}(\mathcal{W}, Z)}{\partial W_{l,i,j}} W_{l,i,j} + \frac{1}{2}\frac{\partial^2 \mathcal{L}(\mathcal{W}, Z)}{\partial W_{l,i,j}^2} W_{l,i,j}^2. \end{aligned} \qquad (2)$$

The above is the quadratic approximation of $U_{\{l,i,j\}}(Z)$.

- The computation of the true utility is very complicated, so instead of directly calculating it, they use an approximation method to make things easier.
- They don’t want to run the model again with different weights (no extra forward passes), so they use a mathematical shortcut called a second-order Taylor approximation. This method looks at how the loss function changes near the current weight and estimates what the loss would be if the weight were set to zero, without actually setting it to zero and recalculating everything. 이 방법은 손실 함수가 현재 가중치 근처에서 어떻게 변하는지 살펴보고 실제로 0으로 설정하고 모든 것을 다시 계산하지 않고 가중치를 0으로 설정했을 경우 손실이 얼마인지 추정합니다.
- They use the approximation by Elsayed and Mahmood (2022) that provides a Hessian diagonal approximation in linear complexity. This makes the computation of both of our utility approximations linear in complexity and therefore scalable.
- This gives a **first-order approximation** of how the loss changes when the weight is removed. The **third term** adds the second derivative (curvature) of the loss with respect to $W_{\{l,i,j\}}$, capturing how the rate of change itself changes. This is the **second-order approximation**, which provides a more accurate estimate.

∴ Overall, this equation approximates how much a specific weight $W_{\{l,i,j\}}$ contributes to the loss function, helping us estimate the utility of keeping or removing that weight.

Method (2)

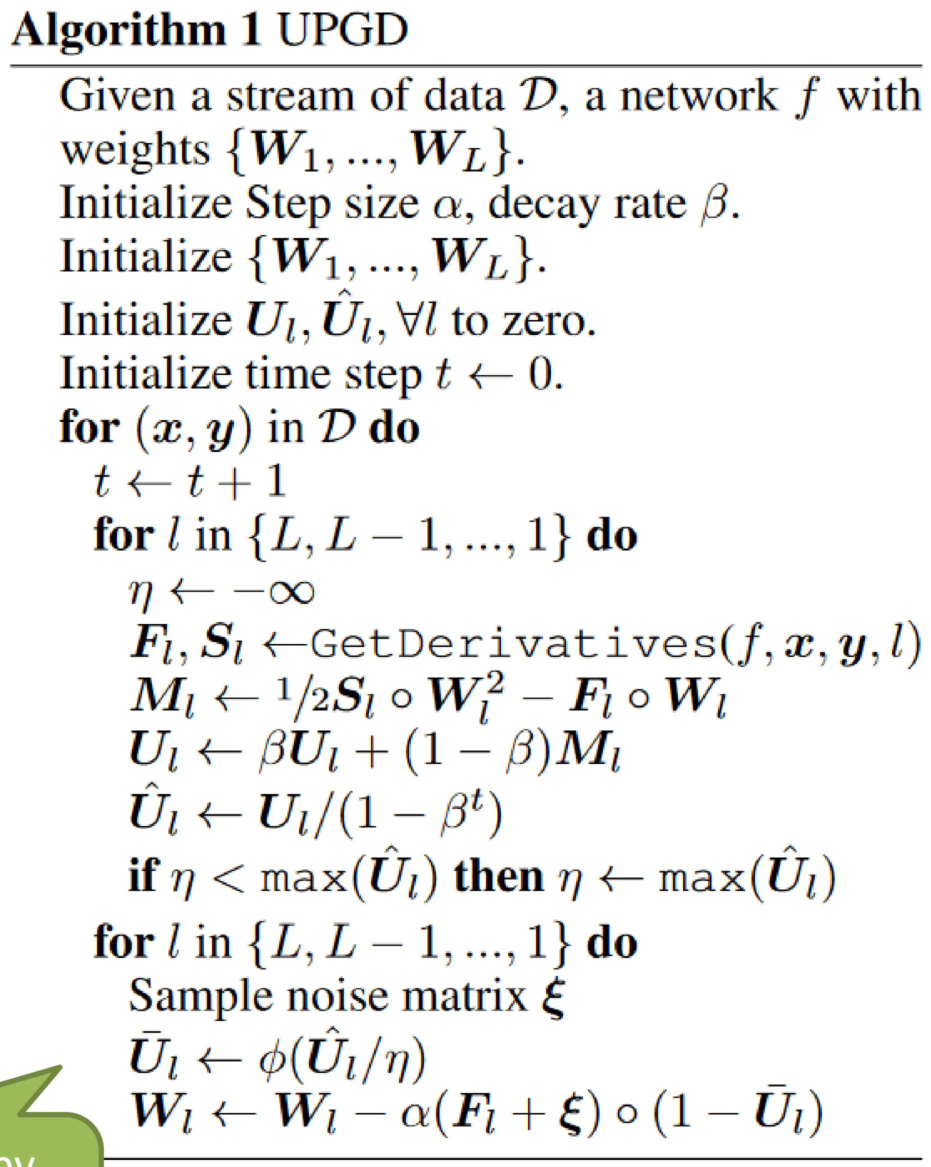
Utility–Based Perturbed Gradient Descent (UPGD)

The utility information is used as a gate, referred to as utility gating, 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 (see Dohare et al. 2023).

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

where $\xi \sim \mathcal{N}(0,1)$ is noise, α is the step-size parameter, 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.

- ξ~N(0,1): Gaussian noise applied to perturb the weights for maintaining plasticity.
- $U_{\{l,i,j\}}$∈[0,1]: Scaled utility value for weight, where 1 indicates high utility (important weight) and 0 indicates low utility.
- F_l. contains first derivatives and S_l contains second-derivative approximations



Algorithm 1 UPGD

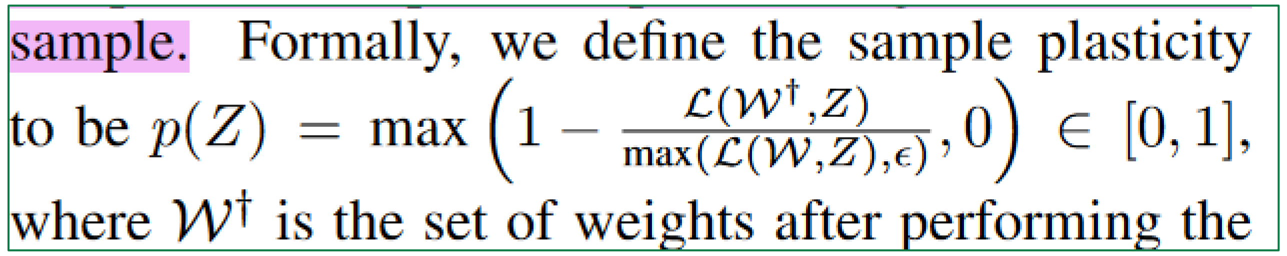
```
Given a stream of data $\mathcal{D}$, a network $f$ with weights $\{\boldsymbol{W}_1, ..., \boldsymbol{W}_L\}$.
Initialize Step size $\alpha$, decay rate $\beta$.
Initialize $\{\boldsymbol{W}_1, ..., \boldsymbol{W}_L\}$.
Initialize $\boldsymbol{U}_l, \hat{\boldsymbol{U}}_l, \forall l$ to zero.
Initialize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

Scale the utility by **the maximum utility of the weights**

Method (3)

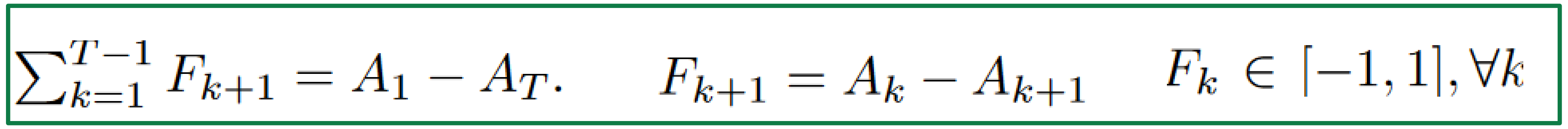
Forgetting and Plasticity Evaluation Metrics

This metric measures plasticity directly, especially since most measures that are introduced (e.g., weight norm) to show loss of plasticity do not often correlate with plasticity (see Lyle et al. 2023).



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

This formula calculates the **difference in accuracy** between two consecutive evaluation windows, measuring how much accuracy is lost (or gained) as the model learns new tasks.
Forgetting is measured by how much accuracy drops between two consecutive windows.If the performance drops over time, it means the model is forgetting what it learned in previous tasks.If the performance improves, it means the model is adapting or improving its previous knowledge.



$$\sum_{k=1}^{T-1} F_{k+1} = A_1 - A_T. \qquad F_{k+1} = A_k - A_{k+1} \qquad F_k \in [-1, 1], \forall k$$

Experiments

Experiments (1)

Quality of the Approximated Utilities

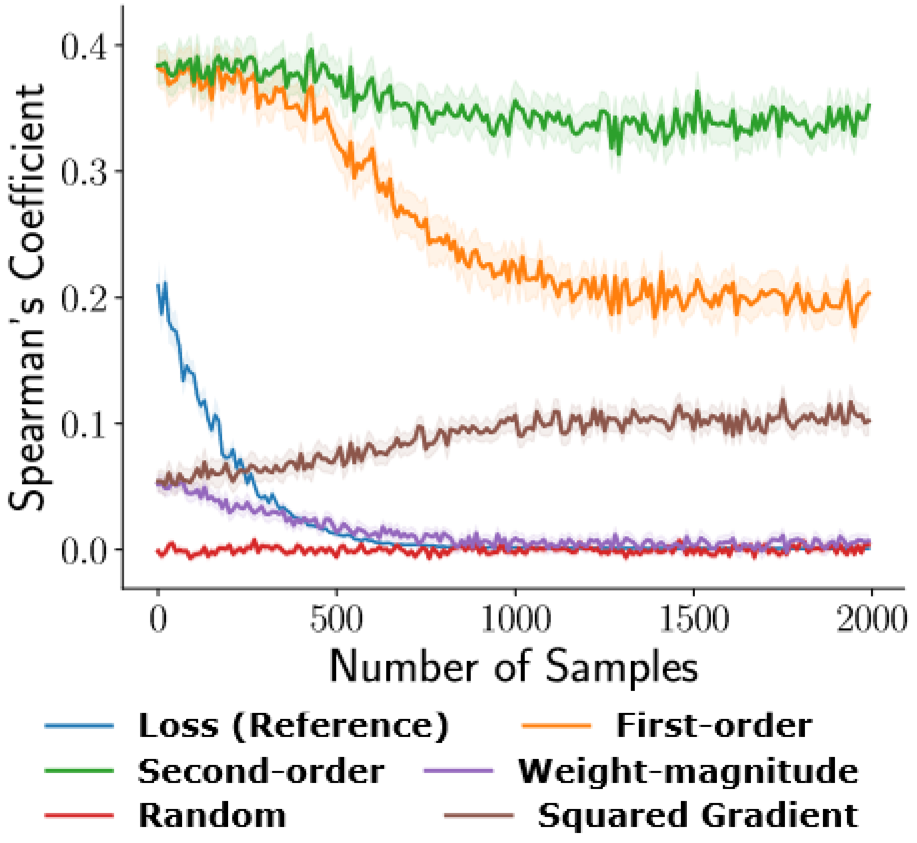
Setting

1. UPGD is tested on popular datasets like **MNIST**, **EMNIST**, **CIFAR-10**, and **ImageNet**, using different neural network architectures, such as **MLPs**, **CNNs**, and **ResNets**.
2. Importantly, the experiments are conducted without common aids like **replay**, **batches**, or predefined **task boundaries**, making the setting more challenging and realistic for continual learning.
3. Evaluation Method assesses **Average online accuracy, 20 independent runs, hyperparameter search**

Analysis

To assess this, the **Spearman correlation** is used, which measures how well the approximated utility aligns with the **true utility**.

1. An **SGD learner** with a small neural network and **ReLU activations** is trained to minimize **online squared error**.
2. Spearman's correlation is calculated for different utility measures
   - **Second-order utility** correlates with the true utility throughout learning, making it the most accurate approximation.
   - **First-order utility** starts well but loses correlation as learning plateaus, possibly due to fluctuations in gradients near the solution.
   - **Random ordering** (baseline) shows zero correlation.
   - **Squared-gradient utility** improves over time but remains below first-order utility.
   - **Weight-magnitude utility** shows a weak and decreasing correlation.



Experiments (2)

UPGD against Loss of Plasticity

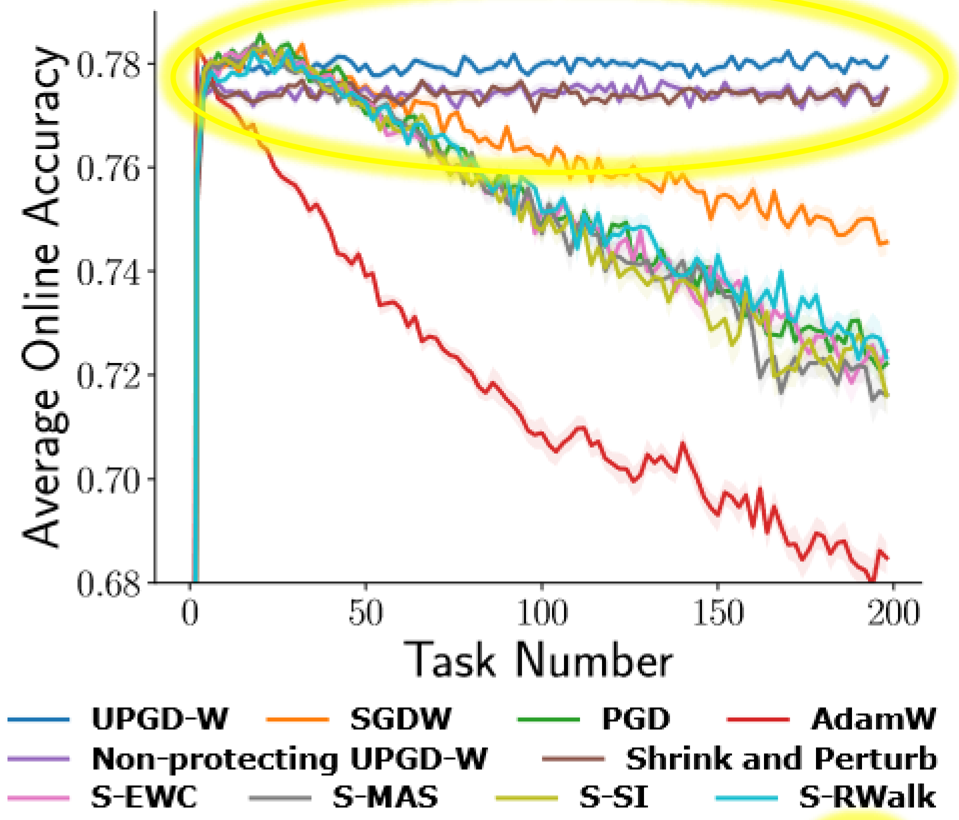
Setting

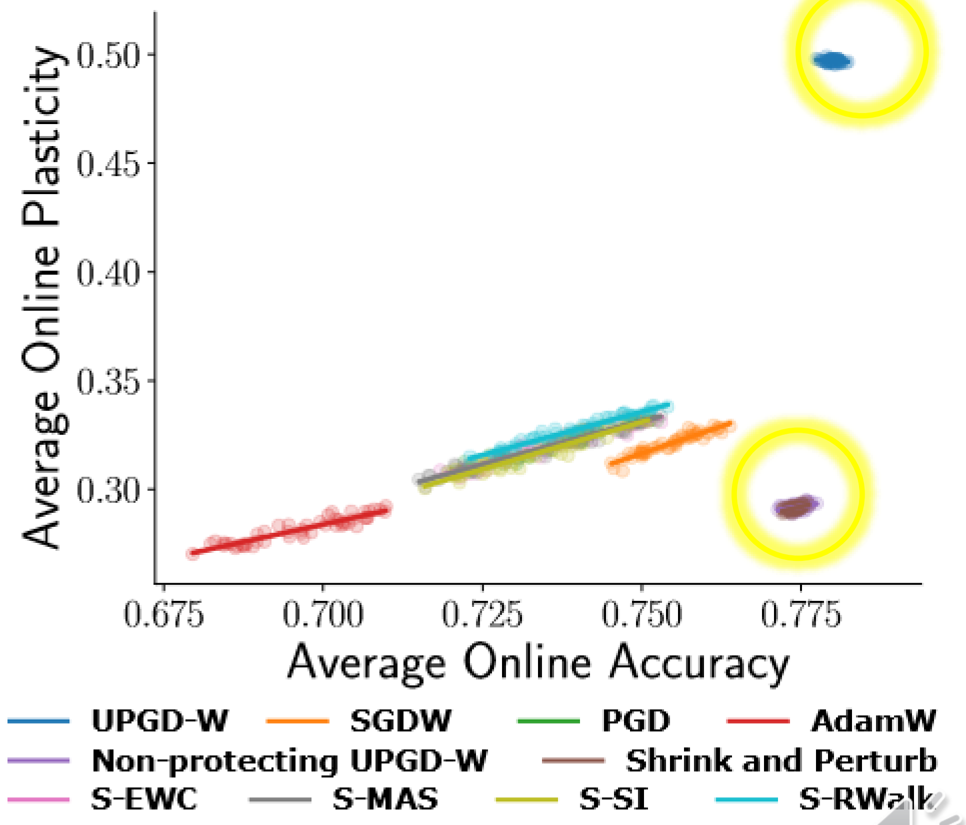
How do **UPGD** and other methods perform in addressing plasticity loss?

1. In this task, inputs are permuted every 5000 steps to create new tasks, forcing the learner to **overwrite previously learned features**. This setup isolates the problem of **plasticity loss**.
2. Various methods, including **SGDW**, **PGD**, **AdamW**, **Shrink and Perturb (S&P)**, **S-EWC**, **S-MAS**, **S-SI**, and **S-RWalk**, were compared to UPGD.

Analysis

1. **UPGD** performs well in maintaining both **plasticity and accuracy**, while methods focusing only on catastrophic forgetting, such as **S-EWC**, show decaying performance over time.
2. When plasticity is the only issue, performance decreases as plasticity is lost, confirming that in this case, **performance reflects plasticity**.
3. **UPGD-W** and **S&P** preserve plasticity effectively, while methods like **S-RWalk** show poorer results in this context.



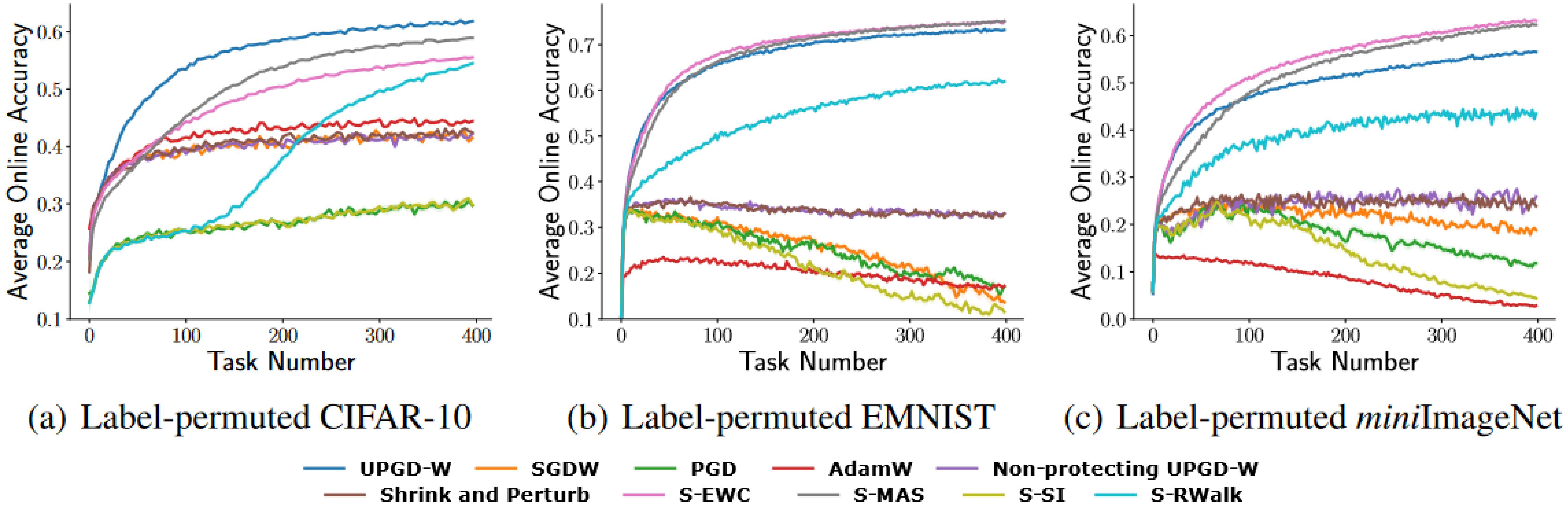


Experiments (3)

UPGD against Catastrophic Forgetting

Analysis

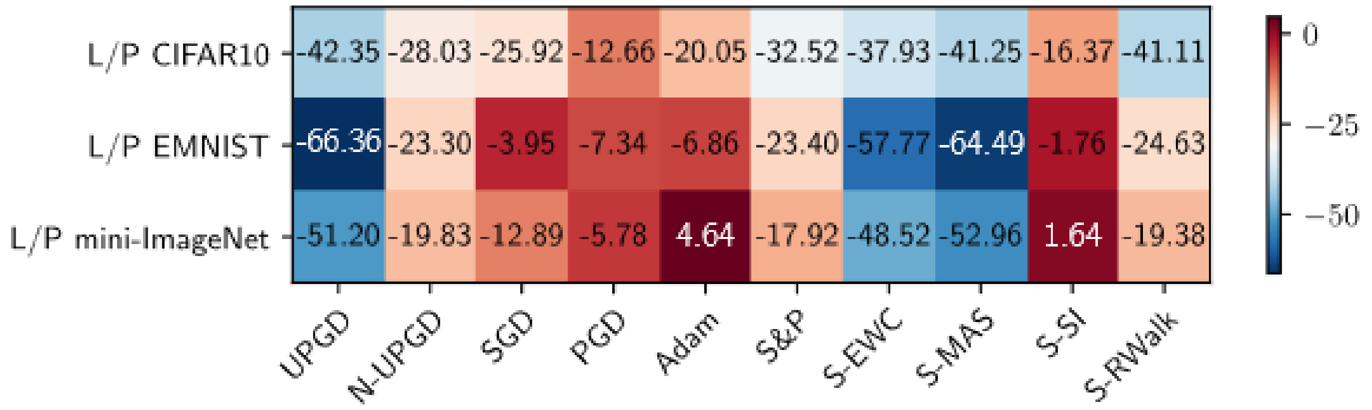
1. **Fig. 6** shows that methods addressing catastrophic forgetting, like UPGD, continually improve their **online accuracy**, while methods that don’t explicitly handle forgetting (e.g., **AdamW**) plateau at around 40% accuracy.
2. **Fig. 7(a)** uses a **forgetting metric**, where positive values indicate forgetting. Learners like UPGD, which address catastrophic forgetting, exhibit less forgetting and better performance compared to those that don’t.
3. **Fig. 7(b)** demonstrates that **loss of plasticity** is not a major issue in this problem, as most methods show negative values for plasticity loss, confirming that the main challenge is catastrophic forgetting.



(a) Label-permuted CIFAR-10 (b) Label-permuted EMNIST (c) Label-permuted *mini*ImageNet

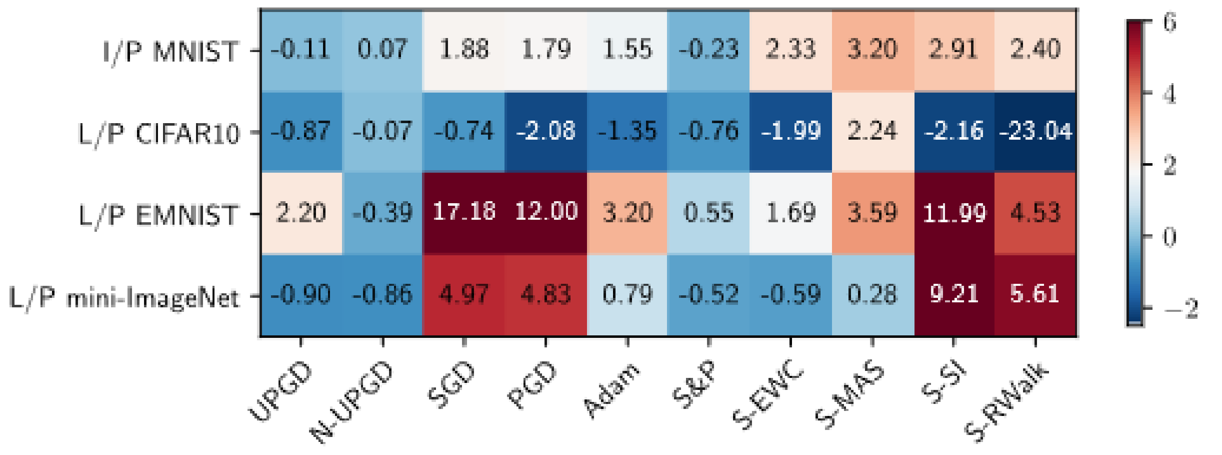
Experiments (4)

UPGD against Loss of Plasticity and Catastrophic Forgetting



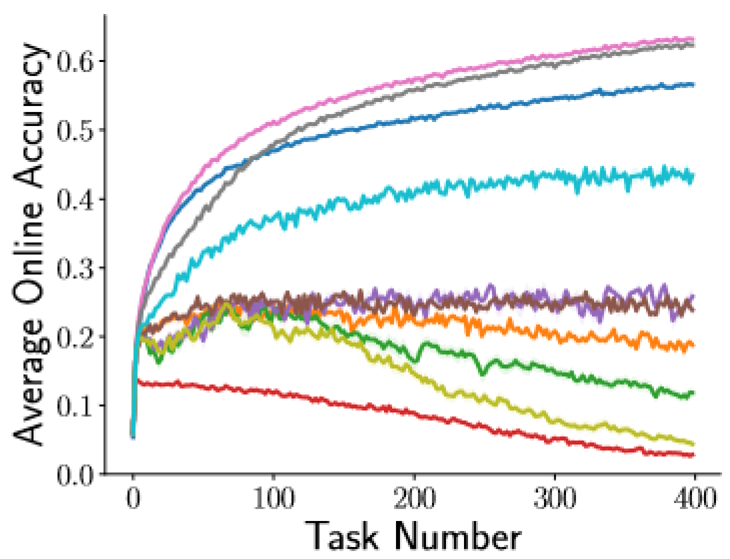
	UPGD	N-UPGD	SGD	PGD	Adam	S&P	S-EWC	S-MAS	S-SI	S-RWalk
L/P CIFAR10	-42.35	-28.03	-25.92	-12.66	-20.05	-32.52	-37.93	-41.25	-16.37	-41.11
L/P EMNIST	-66.36	-23.30	-3.95	-7.34	-6.86	-23.40	-57.77	-64.49	-1.76	-24.63
L/P mini-ImageNet	-51.20	-19.83	-12.89	-5.78	4.64	-17.92	-48.52	-52.96	1.64	-19.38

(a) Forgetting (×100)



	UPGD	N-UPGD	SGD	PGD	Adam	S&P	S-EWC	S-MAS	S-SI	S-RWalk
I/P MNIST	-0.11	0.07	1.88	1.79	1.55	-0.23	2.33	3.20	2.91	2.40
L/P CIFAR10	-0.87	-0.07	-0.74	-2.08	-1.35	-0.76	-1.99	2.24	-2.16	-23.04
L/P EMNIST	2.20	-0.39	17.18	12.00	3.20	0.55	1.69	3.59	11.99	4.53
L/P mini-ImageNet	-0.90	-0.86	4.97	4.83	0.79	-0.52	-0.59	0.28	9.21	5.61

(b) Loss of Plasticity (×100)



(c) Label-permuted *mini*ImageNet

Analysis

The experiment uses **Label-permuted EMNIST**, which includes **47 classes** (digits and letters). This introduces more **non-stationarity** than previous tasks, increasing the likelihood of both plasticity loss and forgetting.

1. To explore both **catastrophic forgetting** and **loss of plasticity**, the experiment uses **Label-permuted EMNIST**, which includes **47 classes** (digits and letters). This introduces more **non-stationarity** than previous tasks, increasing the likelihood of both plasticity loss and forgetting.
2. **Fig. 6(b)** shows that methods like **UPGD-W**, which address both catastrophic forgetting and plasticity loss, continually improve their performance, outperforming methods that focus only on plasticity (e.g., **S&P**).
3. **UPGD-W** continues to show the best results in both **forgetting** and **plasticity metrics** across all tasks, as shown in **Fig. 7(a)** and **Fig. 7(b)**, proving its strength in handling both issues simultaneously.

Conclusion

Conclusion

Limitations of this paper and Future Directions

1. One limitation of UPGD is its assumption that weight utility is measured independently of the changes in other weights. Future work could investigate utility measures that consider interactions between weights.

2. Retaining important weights is a key idea in continual learning, and I aim to incorporate this concept into my research.

3. While UPGD requires minimal **hyperparameter tuning**, finding the best set of hyperparameters in a **lifelong learning** scenario is still a challenge. A potential solution is a **generate-and-test** method for continuous hyperparameter adaptation.

Thank you for your Attention