I Fought the Low

Decreasing Stability Gap with Neuronal Decay

Kirill Zhankov[1]

Supervisor(s): Tom Viering[1]**, Gido van de Ven**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 22, 2025

Name of the student: Kirill Zhankov
Final project course: CSE3000 Research Project
Thesis committee: Tom Viering, Gido van de Ven, Alan Hanjalic

Abstract

Task-based continual learning setups suffer from temporary dips in performance shortly after switching to new tasks, a phenomenon referred to as stability gap. State-of-the-art methods that considerably mitigate catastrophic forgetting do not necessarily decrease the stability gap well. One notable continual learning regularization approach, neuronal decay, attempts to encourage learning solutions that have small activations in the hidden layers. It previously showed improvement in terms of catastrophic forgetting but was not assessed in the context of stability gap. In this study, we compare neuronal decay with a baseline model to see if it can reduce the stability gap. Qualitative analysis with plots and quantitative analysis with metrics, such as gap depth, time-to-recover and average accuracy, both give strong evidence that this simple regularization method can reduce the stability gap with no substantial sacrifice of performance or training time. The source code is available at https://github.com/zkkv/neuronal-decay.

1 Introduction

Deep artificial neural networks are sometimes required to learn from a stream of incrementally changing non-stationary training data [1], [2]. This process is commonly referred to as continual learning (CL). Such a setup might be used, for instance, in real-time systems or when dealing with sensitive data [3] that needs to be discarded immediately after being processed. In task-based continual learning (TBCL) scenarios, the model deals with a discrete set of tasks that are presented to that model one after another [4].

Many models often experience sudden drops in accuracy (just after a task-switch) for old tasks, that is then followed by a recovery period [5]. A schematic representation of the phenomenon can be seen in Figure 1. This *stability gap* becomes apparent if the evaluation is done throughout training, rather than solely at the end of each task. Surprisingly, it does not disappear, even when using full replay (also called joint incremental training), that involves exposing the model to the entire set of old tasks while learning new ones [6]. It is important to consider stability gap as it serves as an indication of the model's performance in the worst case [7], which may be more significant than the average performance in certain contexts.

Different methods of dealing with stability gap have been proposed and tested, each showing various success but none being perfect [4]. One notable approach described in [8] assessed the impact of neuronal decay regularization on overall accuracy. The authors argued that neuronal decay encourages simpler solutions, retaining inactive neurons for future tasks. Yet, they did not address how stability gap is influenced when the outlined method is applied. In this work, we quantitatively and qualitatively explore this approach and evaluate its impact specifically on stability gap.

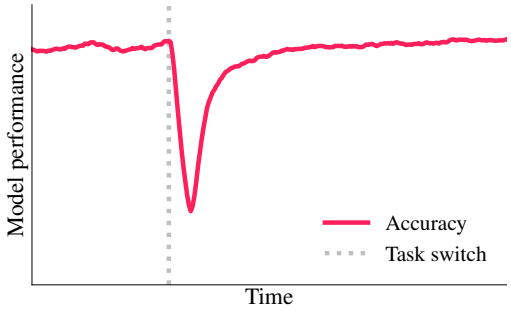


Figure 1: Stability gap. A continual learning model suffers from a temporary dip in performance shortly after a task switch, that is observed only if the evaluation is done continuously.

Multiple experiments with and without full replay were set up to then compare the performance of the model before and after introducing neuronal decay. This allowed us to answer the following research questions:

Q1 (*Main question*) Does inclusion of neuronal decay reduce the stability gap, compared to the baseline that uses replay but not decay?

Q2 Can neuronal decay on its own (with no replay) outperform the baseline that uses replay but not decay?

Q3 Is there a significant computational overhead associated with using neuronal decay?

The results that we have obtained indicate that neuronal decay indeed has the potential to reduce the stability gap, compared to the baseline. It was confirmed that the observed difference was present for various learning rates, model sizes and neuronal decay coefficient values. Notably, neuronal decay alone was not strong enough to decrease the stability gap. Nonetheless, combined with existing state-of-the-art methods, neuronal decay may complement them and help a model maintain acceptable accuracy throughout its lifetime. Importantly, this beneficial effect comes at little cost in terms of computational complexity.

In the remainder of the paper we cover the CL background and importance of attenuating stability gap (section 2); details of neuronal decay approach, used dataset and architecture, metrics and hypotheses (section 3); experimental setup and results (section 4). We conclude with a discussion and mention potential improvements that can be done (section 5).

2 Related Work

In this section, we will cover the most important background concepts that are required to understand the problem. Specifically, we go over the unique challenges CL is dealing with in subsection 2.1; the well-known CL problem of catastrophic forgetting in subsection 2.2; and the stability gap phenomenon in subsection 2.3.

2.1 Continual Learning

Continual learning is a subset of deep learning where the input distribution may change over time (such data is called

non-stationary) [1], [2]. In contrast to the classical deep learning, this requires the model to constantly adapt in order to maintain acceptable accuracy for both old and new data, having to minimize resource consumption, be it time, number of operations or anything else.

While, in general, the data may change gradually, in research and applications, the focus is usually put on task-based CL [4]. It is concerned with a set of discrete tasks that switch at certain time points. These tasks can range a lot in terms of their similarity to each other.

Orthogonal to that lies the separation of CL into three distinct scenarios [4], [9]. In task-incremental learning, the specific task is provided as input or somehow known, and the output does not change over time. The benefit of knowing the task is that one can simply use a part of the network suited best for that particular task. In class-incremental learning the task is not given, and the model's output space grows with each task, thus making this scenario drastically more difficult. The last scenario, which can be considered a middle ground, is domain-incremental learning. Here, the model knows nothing about what task it is dealing with, yet the output format remains constant. Put simply, in the case of classification, each new task expands the domain that we have to accommodate, but we ask the model to classify among the same number of possible labels. In this study, we focus solely on this scenario.

2.2 Catastrophic Forgetting

In numerous previous studies focusing on TBCL, it has been observed that the performance of the model in the initial tasks often deteriorates when new tasks are introduced, which is known as "catastrophic forgetting" [10], [11]. In its most extreme form, the network completely discards the previous set of parameters in order to achieve high accuracy (or, equivalently, low loss) for the active task. Crucially, the apparent decrease in performance is not necessarily caused by lack of model capacity, demonstrated by interleaved simultaneous training on multiple tasks at once [4]. This knowledge led researchers to believe that catastrophic forgetting in TBCL setting can be avoided, or at the very least minimized.

The goal of any TBCL method, both in artificial networks or otherwise, involves carefully balancing between stability and plasticity [12]. This ideally leads to a point in the parameter-space for which the model maintains acceptable accuracy in all previously seen tasks [13]. Generally, this is done in two non-exclusive ways [4], [6]: changing *what* exactly is optimized (such as with experience replay [14], Elastic Weight Consolidation [15]) and changing *how* the parameter space is traversed and a point of minimum is reached (Orthogonal Gradient Descent [16], hard attention mechanism [17]).

Neuronal decay, that we present in the next section, has properties that make it fit both categories. While the loss function is modified, the motivation behind that is to promote learning simpler solutions, thus preserving capacity for future tasks [8].

2.3 Stability Gap

The most common state-of-the-art approaches described in the literature put emphasis on avoiding catastrophic forgetting [5]. However, a less obvious and less studied issue still persists. Even though the model might ultimately reach the desired level of performance in every task, these approaches still suffer from a sudden drop in accuracy, a stability gap, occurring at task switches. To observe the stability gap, it is necessary to evaluate the model performance during its training, ideally – after every batch.

Studying the stability gap phenomenon is important for several reasons. For one, its presence could be an indication that existing CL approaches are suboptimal [6]. Discovering generalizable methods that exhibit no stability gap can potentially make deep neural networks spend fewer resources on achieving the same result. Higher efficiency can result in both more powerful and less expensive models.

Another intriguing aspect of stability gap is that it is seemingly not present in the human brain or other biological systems that perform continual learning [5], [18]. In fact, humans tend to do worse when tasks are introduced in an interleaved way, compared to cases where each task is learned sequentially [18]. This too could mean that models which were initially inspired by the brain network, lack some fundamental aspects that enable the brain to be so efficient at learning. Closing that gap could lead both to an improvement in deep learning methods and a better understanding of cognitive mechanisms.

Finally, as mentioned before, a large stability gap is associated with poor worst-case performance. This may be acceptable most of the time but pose a problem when dealing with safety-critical applications [7]. Any decrease in performance, even a momentary one, can theoretically be exploited by malicious actors [5].

3 Methodology

In this section, a high-level description of the applied method is outlined. In subsection 3.1 we examine the regularization method that is compared against a baseline model; in subsection 3.2 a description of the network architecture, dataset and tasks are given; subsection 3.3 covers the metrics; and subsection 3.4 lists the hypotheses and the justification behind them.

3.1 Neuronal Decay

Regularization refers to a broad set of machine learning techniques that try to keep the model more generalizable [19]. Traditionally, machine learning regularization involves modifying the loss function to discourage some particular behavior during the training step. In classical deep learning, such a penalty typically leads to a simpler model.

In the context of CL, however, the use of the term regularization is slightly different. Regularization-based methods usually try to either discourage big changes of the network's parameters or function values at specific points [4].

In the framework outlined in [8], the loss function comprises two terms: the usual cross-entropy loss $\mathcal{L}_{CE}$ and the additional neuronal decay term $\mathcal{L}_{ND}$, with some coefficient λ:

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda \mathcal{L}_{ND} \tag{1}$$

The decay term $\mathcal{L}_{ND}$ is defined by the authors to be either of:

$$\mathcal{L}_1 = \sum_{x \in B} \sum_{m}^{M} \sum_{i}^{N^{(m)}} \left| f^{(m)}(x)_i \right| \tag{2}$$

$$\mathcal{L}_2 = \sum_{x \in B} \sum_{m}^{M} \sum_{i}^{N^{(m)}} \left(f^{(m)}(x)_i \right)^2 \tag{3}$$

where the activation function value $f^{(m)}(x)_i$ of each neuron i of each hidden layer m is evaluated for every sample x of the batch B. Here, M indicates the number of hidden layers and $N^{(m)}$ is the number of neurons in a particular layer m. The absolute value or square is taken, which is then summed across all samples and hidden neurons to produce a single scalar value. This definition makes neuronal decay more closely related to the classical weight decay regularization, rather than the usual CL regularization approaches.

The inclusion of the regularization term encourages the model to learn compact solutions and preserve the rest of the network for subsequent tasks [8]. Between any two given solutions, both having high accuracy, the one that uses fewer resources becomes more favorable. Figure 2 demonstrates this concept. It shows two networks during inference, producing roughly the same result for a particular sample. The difference is that the network on the left learned sparse activations, indicated by few active neurons, while the other network learned dense activations. The first network is preferred due to its presumed remaining capacity for future tasks.

The original work on neuronal decay focused on mitigation of catastrophic forgetting. The authors showed considerable improvement in that aspect compared to the baseline experiment. However, since the authors did not provide a discussion of the effect of neuronal decay on stability gap, a more thorough investigation is conducted by us to assess it from that angle.

3.2 Network and Data

To answer the aforementioned research questions, a multilayer perceptron (MLP) neural network is set up. Due to its simple structure, we can focus our attention on the important differences. The baseline experiment uses full replay but without any neuronal decay (λ is effectively zero). In the experimental model, $\mathcal{L}_2$ neuronal decay, defined in (3), is computed for all hidden layers of the MLP. We stick to only $\mathcal{L}_2$ because we expect the effect of introducing $\mathcal{L}_1$ to be similar. The choice is made in favor of $\mathcal{L}_2$ due to its reported wider range of suitable values [8]. Neuronal decay value is calculated in each forward pass and added to the loss. All other parameters of the experiments, such as the learning rate, optimizer, batch size, model size, etc., are left exactly the same between the baseline and experimental models.

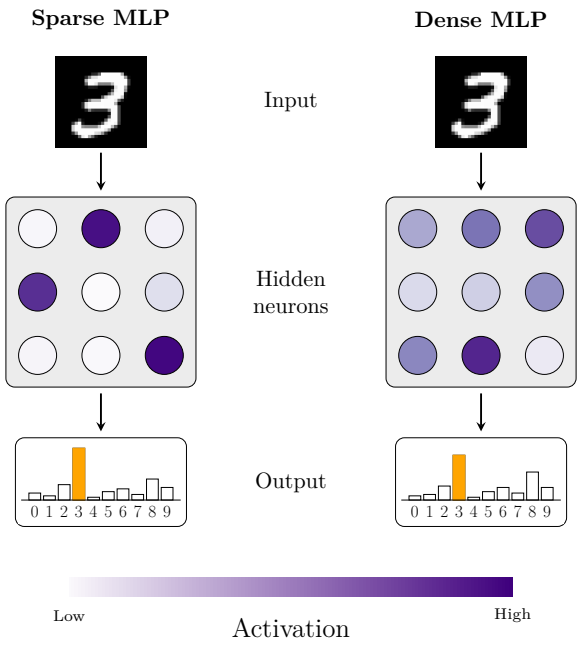


Figure 2: A schematic representation of a sparse network with few active neurons and a dense network with many active neurons. **Both networks achieve comparable performance, yet neuronal decay method prefers the sparser network.**

The dataset of choice is MNIST [20], containing low-resolution images of handwritten digits. It works well with an MLP and allows achieving high accuracy quickly. Reaching high accuracy is important for observing a stability gap in the first place. We apply rotations to the digits from the dataset to generate new tasks. Thus, in Task 1 the network has to learn to classify digits with no rotation applied; in the next task it has to classify digits both with and without rotation, and so on. An illustration of this setup can be seen in Figure 3.

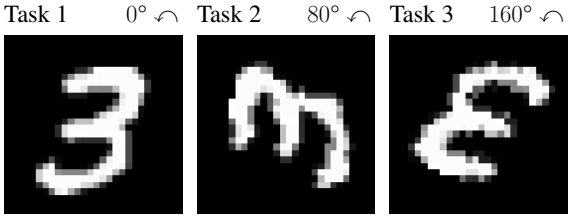


Figure 3: A training sample showing the three tasks used in the study. Each task corresponds to a rotation: $0°, 80°$ and $160°$.

To assess the model's performance continuously, each forward pass is followed by an evaluation step. The performance in that step is measured as the portion of the samples classified correctly. This value is computed for each task from the moment the task is first introduced onwards.

3.3 Metrics

We assess the stability gap both qualitatively, with the help of plots, and quantitatively, by simple metrics that we define here. Figure 4 gives an intuition behind how the metrics are computed. The gap depth GD for each task, starting from the second one, is measured as the decrease in percentage points (p.p.) from the accuracy in the very last batch before the task switch to the lowest accuracy throughout that same task. Another metric of interest is time-to-recover TTR. We measure, in percentage relative to the number of batches in the task, how long it took for the model to get from the lowest point within the task back to the level of accuracy before the task switch. Finally, we also consider accuracy ACC either for a single task or averaged out across all current and past tasks at the moment of the task switch. This lets us confirm that the applied method does not sacrifice plasticity.

For all three metrics, the subscript i, j denotes the performance of which task i is considered and at what interval j it is considered. For instance, the gap depth for performance of Task 1 when the Task 3 is being learned is $\mathtt{GD}_{1,3}$.

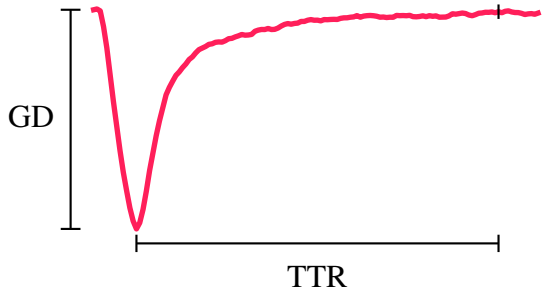


Figure 4: An example of how gap depth GD (in p.p.) and time-to-recover TTR (expressed as the percentage of the total number of batches in the task) metrics are computed for the accuracy curve.

In order to answer **Q3**, we use a profiler to better grasp the impact the proposed method may impose. To get a hardware-independent measure of how complex it is to train the model, we also calculate the number of multiply-accumulate operations (MACs) per training batch analytically. One MAC is defined as one multiplication and one addition, thus making it equal to two floating-point operations (FLOPs) [21].

3.4 Hypotheses

Based on the research questions, three hypotheses are formulated:

H1 Addition of neuronal decay reduces the stability gap with little impact to the average accuracy.

H2 Neuronal decay alone is not enough to outperform replay.

H3 Neuronal decay is marginally worse in terms of computational complexity than the baseline (during training).

The first hypothesis stems from the observation that sparser activations improve the network's ability to learn multiple tasks [8]. Thus, while higher average accuracy does not necessarily imply a smaller stability gap, it's reasonable to assume that newer tasks could be learned with less interference within the model. We motivate the next hypothesis by reflecting on the idea that full replay is an idealistic scenario, since it practically makes the model see the old data in full again. Therefore, it sets up a too high of bar to beat with a simple regularization term like this one. The final hypothesis is driven by preliminary analysis of the regularization term. Since it does not involve complex computations, it is expected to introduce very little overhead during training. At the same time, we point out that regularization, in general, doesn't affect inference runtime at all.

4 Experimental Setup and Results

Experiments were conducted using the PyTorch v2.7.0 framework for Python v3.12.8 and largely based on the code from [1]. The following hyperparameters and configuration variables stayed the same throughout all experiments, unless specified otherwise. Batch size was chosen to be 512 with 500 batches per task, which, for MNIST, equates to roughly 4 epochs per task. This number of batches per task was enough for accuracy to stabilize and avoid overfitting. Note that we simulated an environment where we had access to a stream of data, therefore, the number of epochs is not an integer amount, but it is still reported for completeness. After each complete epoch the training set got reshuffled. We set the test size to 2000 images to get a reliable estimation of accuracy after each batch. The test set was reshuffled with every batch. The learning rate was fixed at 10^{-3} and the chosen optimizer was Adam with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, which are the defaults of PyTorch. The MLP had a single hidden layer with 2048 neurons. Finally, we had three tasks corresponding to three rotations: $0°, 80°$ and $160°$.

To achieve reproducible results, fifteen random seeds were chosen, with which we ran the exact same set of experiments. The seeds can be found in the repository for this paper.

Visualized in Figure 5 is the performance of the model throughout its lifetime in Task 1, averaged at each time point over all executions. In red is the performance of the baseline model, and in blue is the performance of the experimental model (ND). The latter model showed a smaller gap depth and approximately the same time-to-recover after both task switches. The accuracy plateaued at a slightly lower value, which is most apparent in the third interval.

The average accuracy across all current and past tasks is given in the Figure 6. It appears to be similar for baseline and ND. To gain even more insight, it is useful to compare the accuracy for all tasks. Figure 7a shows how accuracy changed for every task in the baseline model, while Figure 7b does so for the ND model. The difference was mostly in favor of the second one: the gap depth was smaller, and the curve was sharper. These important observations imply that neuronal decay did not reduce the stability gap in exchange for worse accuracy for other tasks.

The computed metrics, presented in Table 1 and Table 2, sup-

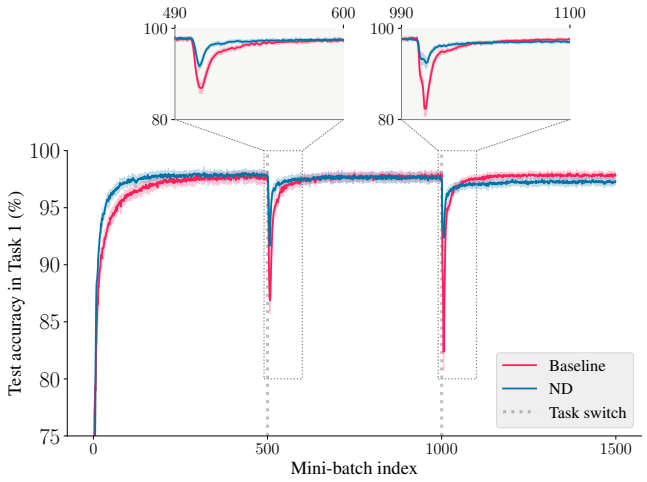


Figure 5: Test accuracy (%) for the baseline and neuronal decay (ND) model in Task 1. The shaded area represents the standard error computed over all seeds. **The gap depth in the ND model is smaller, but so is the final accuracy.**

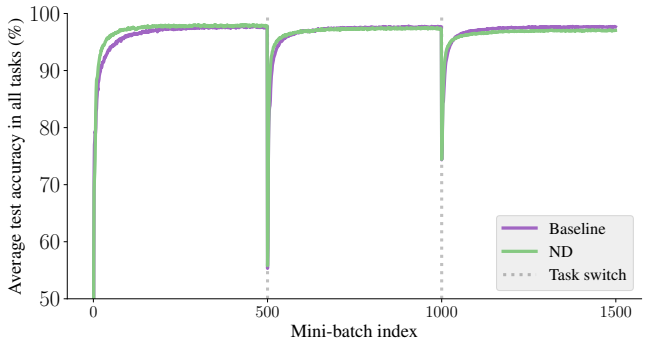


Figure 6: Average test accuracy (%) for the baseline and neuronal decay (ND) model, computed over all present and past tasks. The last interval considers all the tasks. **There is no significant difference between the two, apart from slightly sharper curves and lower final accuracy for the ND model.**

port the qualitative analysis. The values of $\texttt{GD}_{1,2}$ and $\texttt{GD}_{1,3}$ were smaller by 5.3 p.p. and 10.9 p.p. respectively. ACC was higher for the baseline model, but the difference was not drastic. It is likely that tweaking the hyperparameters could reduce this difference while preserving the positive effects. TTR had quite a large variance which makes it harder to conclusively state that either method should be preferred.

It should be noted that these metrics were computed individually for each seed, and the reported standard error was computed across the aggregated metrics. The figures, however, present the average performance at every time point.

To show the robustness of these results, we ran experiments with some hyperparameters tweaked in the baseline and the ND model. Figure 8 illustrates how λ affected the performance. A lower value of $1 \cdot 10^{-7}$ led to the regularization term being less significant, thus the gap becoming more similar to that of the baseline. On the contrary, a higher value of $5 \cdot 10^{-5}$ could provide an even better improvement in terms of gap depth but also sacrificing the (initial) performance in Task 1. We report how the model performed in Task 1 in the last interval and on average for different values of λ in Table 3.

Table 1: Task 1 gap depth GD, Task 1 time-to-recover TTR and average accuracy ACC, computed in the Task 2 interval for baseline and neuronal decay (ND) models. The average accuracy only includes Task 1 and Task 2, but not Task 3. **The ND model shows a decrease in GD while sacrificing some accuracy; TTR varies greatly in both models.**

Model	$\texttt{GD}_{1,2}$ (p.p.) ↓	$\texttt{TTR}_{1,2}$ (%) ↓	$\texttt{ACC}_{\text{AVG},2}$ (%) ↑
Baseline	11.7 ± 1.1	15.8 ± 7.0	$\mathbf{97.64 \pm 0.26}$
ND	$\mathbf{6.4 \pm 0.7}$	$\mathbf{14.8 \pm 11.0}$	97.23 ± 0.23

Table 2: Task 1 gap depth GD, Task 1 time-to-recover TTR and average accuracy ACC, computed in the Task 3 interval for baseline and neuronal decay (ND) models. The average accuracy includes all three tasks. **The ND model shows a decrease in GD while sacrificing some accuracy; TTR varies greatly in both models.**

Model	$\texttt{GD}_{1,3}$ (p.p.) ↓	$\texttt{TTR}_{1,3}$ (%) ↓	$\texttt{ACC}_{\text{AVG},3}$ (%) ↑
Baseline	16.4 ± 1.4	$\mathbf{15.0 \pm 6.4}$	$\mathbf{97.67 \pm 0.17}$
ND	$\mathbf{5.5 \pm 1.0}$	16.1 ± 9.6	97.04 ± 0.16

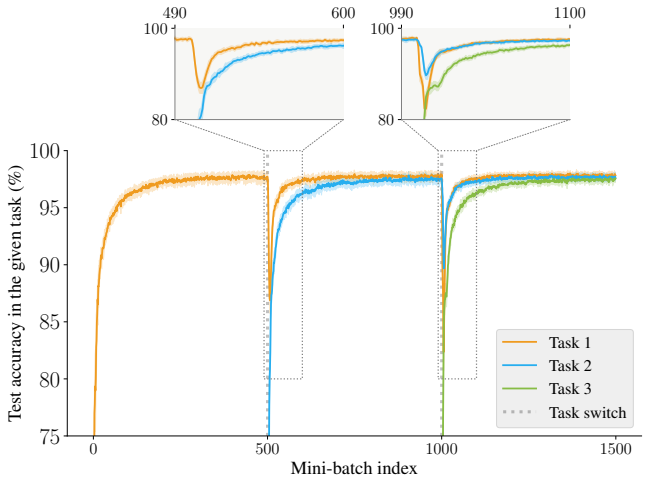
Table 3: Gap depth GD, time-to-recover TTR and accuracy ACC, computed for Task 1 in the Task 3 interval for three different values of the neuronal decay coefficient λ. At most three seeds are excluded from each TTR computation due the model not recovering at all. **Larger value of λ leads to a decrease in GD and TTR but also lower ACC.**

λ	$\texttt{GD}_{1,3}$ (p.p.) ↓	$\texttt{TTR}_{1,3}$ (%) ↓	$\texttt{ACC}_{1,3}$ (%) ↑
$1 \cdot 10^{-7}$	11.0 ± 1.3	21.3 ± 13.8	$\mathbf{98.20 \pm 0.18}$
$1 \cdot 10^{-5}$	5.5 ± 1.0	16.1 ± 9.6	97.33 ± 0.25
$5 \cdot 10^{-5}$	$\mathbf{4.2 \pm 0.9}$	$\mathbf{13.2 \pm 10.6}$	96.32 ± 0.34

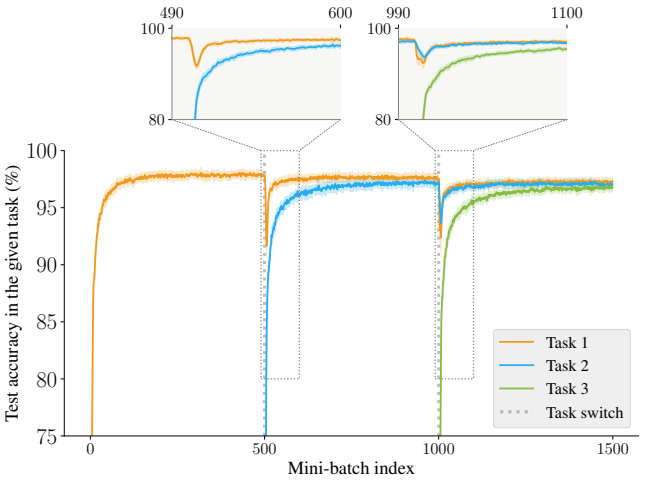
The effects of changing the learning rate of the optimizer and the number of neurons in the hidden layer are displayed in Figure 9 and Figure 10, respectively. The learning rate affected the grand picture to a high degree: lowering the learning rate led to better base performance of the ND model, but the absolute decrease in the gap was present across different learning rate settings. The difference in performance between an MLP model with 1024 neurons and 4096 appears marginal. However, the results may be different for models that are several magnitudes larger.

Next, we take a look at the performance of the models without replay (for the standard set of hyperparameters), which is plotted in Figure 11. The baseline model with replay performed best by a large margin. However, notably, neuronal decay *did* make the overall forgetting a bit less severe, compared to the baseline that did not use replay at all.

Lastly, we report the number of MACs in the baseline and



(a) Test accuracy (%) of all tasks for the baseline model.



(b) Test accuracy (%) of all tasks for the neuronal decay model.

Figure 7: Test accuracy of all tasks for the (a) baseline and (b) neuronal decay model. The shaded area represents the standard error computed over all seeds. **Accuracy curves of Task 2 and Task 3 follow a similar path between the two models.**

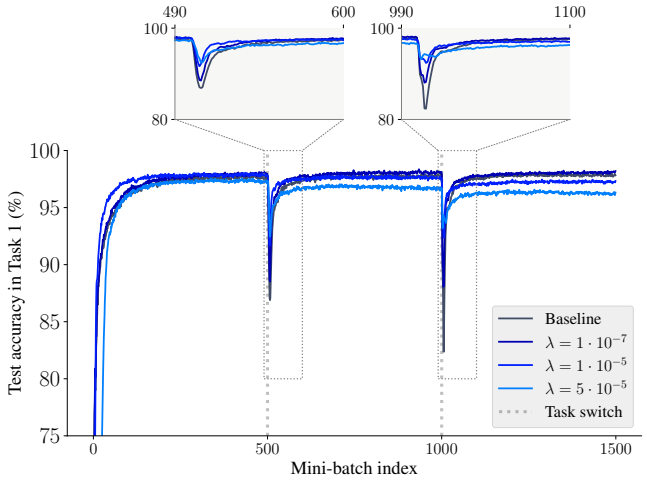


Figure 8: Test accuracy (%) for the baseline model and three neuronal decay models with different values of coefficient λ. **Lower values of λ provide less benefit in terms of stability gap but preserve a higher level of accuracy.**

ND models and the profiling results. The baseline model performs 5 820 416 operations in a single forward pass for a single image. The ND model adds an overhead of just 4096 MACs to compute the $\mathcal{L}_2$ decay. The setup for the profiling purpose was restricted to only Task 1. Time per batch was computed as the total time spent in the task, divided by the number of the batches in the task, then averaged over 60 seeds. The results are reported in Table 4. Both CPU and CUDA time was slightly higher when using neuronal decay.

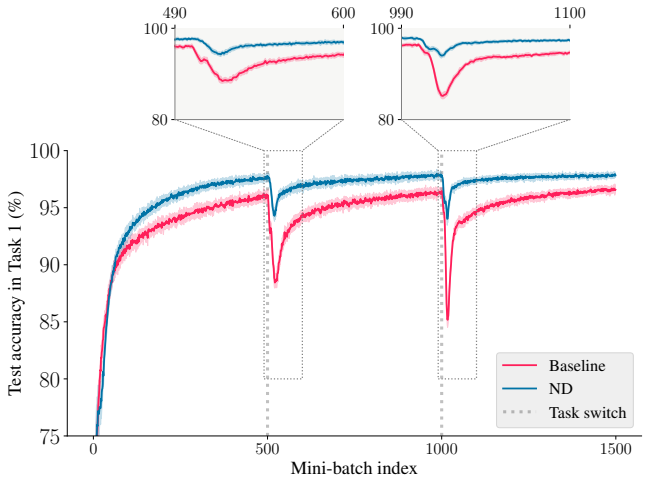
5 Discussion and Future Work

Analysis. The results showed a definite pattern: neuronal decay reduced the gap depth. The effect on accuracy in the same task was less severe and mostly offset by considering all tasks together. Time-to-recover remains the most problematic with its high variance: it needs to be studied further to see if more consistent results can be obtained. While neuronal decay on its own may help with maintaining some stability, it is clear that it should be used in conjunction with other methods. Neuronal decay's increase in the number of MACs is marginal: 0.007%. Importantly, the number of MACs for extra operations scales linearly with the number of neurons. This characteristic can be very favorable in CL scenarios. The increase in absolute time per batch was 6.66% for CPU and 8.56% for CUDA (these are not necessarily independent). The absolute time can be considered more tangible and useful for those who train the models, however it is, arguably, a lot less reliable of a metric.

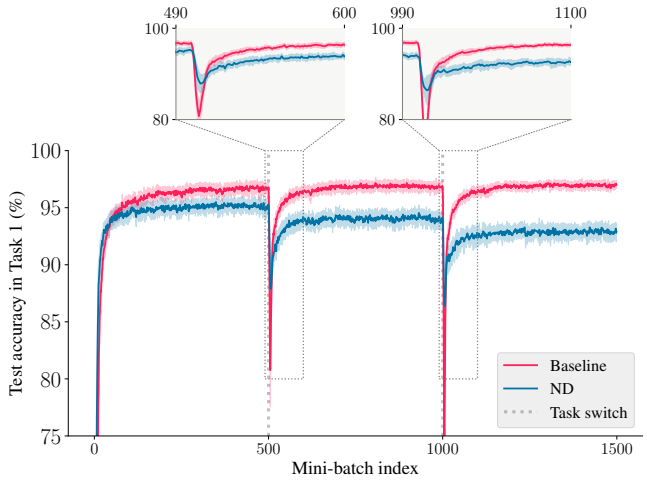
Table 4: Average training time per batch (ms) for the baseline and neuronal decay (ND) models in Task 1. The reported values are computed as the total time spent in the task divided by the number of batches in the task; then the mean and standard error are computed over 60 seeds. The CPU and CUDA events are not necessarily independent. **The ND model, on average, spends slightly more time per batch for both the CPU and CUDA events.**

Model	CPU time (ms) ↓	CUDA time (ms) ↓
Baseline	$\mathbf{18.01 \pm 0.61}$	$\mathbf{14.60 \pm 0.16}$
ND	19.21 ± 1.20	15.85 ± 0.48

Experiments with modified hyperparameters give a solid reason to think that models with neuronal decay exhibit such behavior in a wide range of setups, at least for this type of model and this dataset. Still, we optimized the hyperparameters to a limited extent, so further work can address this aspect. This could also explain the fact that we did not see an improvement in average accuracy, observed in [8]. In this light, we emphasize that a proper choice of λ is crucial for balancing

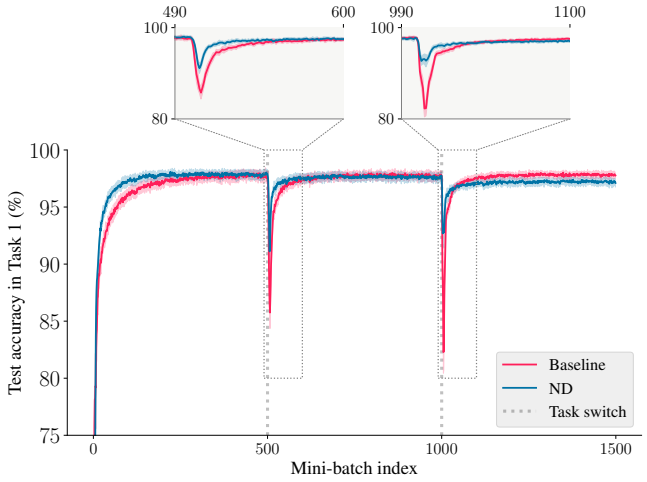


(a) Test accuracy (%) with the learning rate of 10^{-4}.

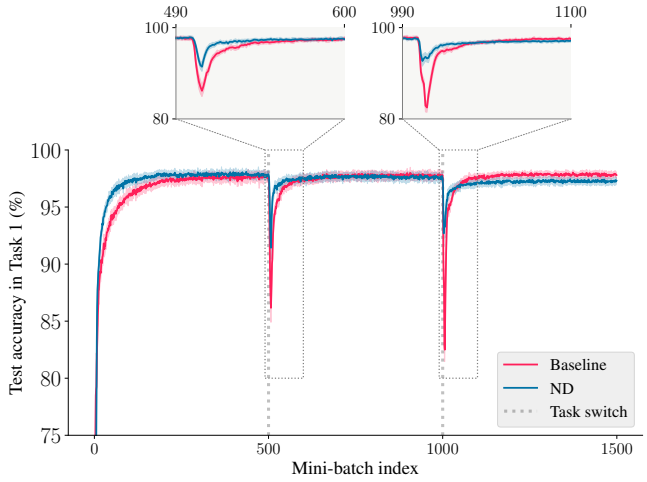


(b) Test accuracy (%) with the learning rate of 10^{-2}.

Figure 9: Test accuracy with (a) lower and (b) higher learning rates. The shaded area represents the standard error computed over all seeds. **The general pattern of a smaller gap depth is present in both cases, however lower learning rate demonstrates higher base performance, and vice versa.**



(a) Test accuracy (%) for a network with 1024 hidden neurons.



(b) Test accuracy (%) for a network with 4096 hidden neurons.

Figure 10: Test accuracy with (a) smaller and (b) larger multi-layer perceptron networks. The shaded area represents the standard error computed over all seeds. **The two plots show little difference.**

between keeping adequate performance and benefiting from the decrease in stability gap. Given how learning rate may affect the performance curve, a further investigation into its interaction with λ is essential.

Discussion. The negative effects on the accuracy were likely due to the loss function representing the true goal less precisely. This is the case with practically any regularization method. The decrease in stability gap, we think, can be caused by simplicity of the solutions, reflected in the sparsity of the neural network, which allows for better generalization. This explanation is in line with the results obtained in [8]. Future studies may focus more on assessing the model's complexity in a quantitative fashion.

Limitations. As mentioned above, the setup with an MLP is rather simple. Neuronal decay may behave completely differently, even detrimentally, when applied to different architectures. The authors of [8] state that the behavior of neuronal decay in convolutional neural networks and transformers is still unknown, and we too think that it might be the next step. We appreciate the fact that even though rotations were limited to 160°, some digits, like `0` or `8`, may still change too little under rotations. We think it is unlikely that this factor affected the results, however.

What we did not examine closely is how neuronal decay may act in the long term, meaning, when dealing with dozens or hundreds of tasks. Whether its effect becomes less apparent

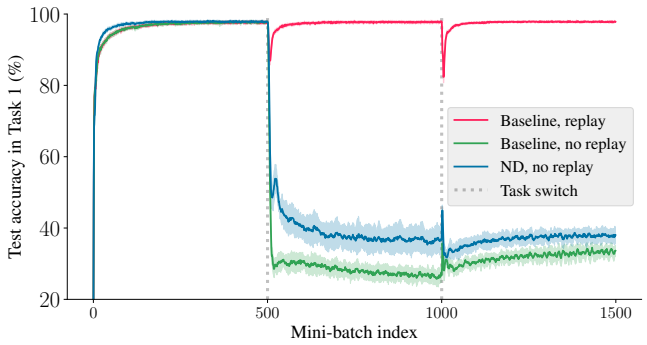


Figure 11: Test accuracy (%) for the baseline model with and without full replay and a neuronal decay (ND) model without replay. The shaded area represents the standard error computed over all seeds. **The baseline model with replay performs best by a large margin, but neuronal decay does better, compared to the baseline with no replay.**

or not remains an open question.

6 Conclusion

In this work we looked into a regularization technique called neuronal decay and its effect on stability gap. Its usage was motivated by trying to achieve sparser activations, thus leaving capacity for future tasks to be learned. Concretely, we compared the baseline model against the neuronal decay model, with and without full replay, and investigated the potential impact using such a method may impose.

The acquired results strongly suggest that neuronal decay is a solid way to reduce the stability gap. It was not powerful enough to mitigate the stability gap on its own, however it could complement other methods, while introducing little computational overhead. Its slight negative effect on the average accuracy could be considered acceptable in those setups where the worst-case performance is more important. We discovered that the choice of the neuronal decay coefficient λ affected the stability-plasticity balance greatly. Therefore, it should be paid close attention to when deploying models in practice.

7 Responsible Research

Licensing. We made sure that this work adheres to the principles of responsible research. All external code – such as the one contained in the libraries and frameworks – and the MNIST dataset are covered by permissive open-source licenses. The code developed for this work is itself released under the MIT license to promote collaboration and transparency.

Reproducibility. Considerable effort was made to ensure the work is reproducible. To facilitate that, we fixed and documented all seeds with which we ran the experiments. On top of that, we documented the exact versions of Python and all dependencies such that an identical environment could be easily recreated. Finally, we tried to make it as easy as possible to read and understand the code, as well as run the experiments, for anyone seeing the codebase for the first time.

Ethics and societal impact. This study focuses on fundamental aspects of deep learning and uses no sensitive or identifying human subject data. That said, we still appreciate the fact that applied machine learning, in general, is accompanied by societal and ethical concerns. Taking that into account, we strongly oppose the use of continual learning algorithms in ways that could lead to discrimination or pose risks to human well-being or safety.

References

[1] G. M. van de Ven, T. Tuytelaars, and A. S. Tolias, "Three types of incremental learning," *Nature Machine Intelligence*, vol. 4, no. 12, pp. 1185–1197, 2022.

[2] R. Hadsell, D. Rao, A. A. Rusu, and R. Pascanu, "Embracing change: Continual learning in deep neural networks," *Trends in Cognitive Sciences*, vol. 24, no. 12, pp. 1028–1040, 2020.

[3] S. Farquhar and Y. Gal, "A Unifying Bayesian View of Continual Learning," *Bayesian Deep Learning Workshop at NeurIPS*, 2018.

[4] G. M. van de Ven, N. Soures, and D. Kudithipudi, "Continual learning and catastrophic forgetting," 2024, *arXiv:2403.05175*.

[5] M. D. Lange, G. M. van de Ven, and T. Tuytelaars, "Continual evaluation for lifelong learning: Identifying the stability gap," in *The Eleventh International Conference on Learning Representations*, 2023.

[6] T. Hess, T. Tuytelaars, and G. M. van de Ven, "Two complementary perspectives to continual learning: Ask not only what to optimize, but also how," in *Proceedings of the 1st ContinualAI Unconference*, S. Swaroop, M. Mundt, R. Aljundi, and M. E. Khan, Eds., ser. Proceedings of Machine Learning Research, vol. 249, PMLR, 2024, pp. 37–61.

[7] S. Kamath, A. Soutif-Cormerais, J. Van De Weijer, and B. Raducanu, "The expanding scope of the stability gap: Unveiling its presence in joint incremental learning of homogeneous tasks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2024, pp. 4182–4186.

[8] R. O. Malashin and M. A. Mikhalkova, "Avoiding catastrophic forgetting via neuronal decay," in *2024 Wave Electronics and its Application in Information and Telecommunication Systems*, 2024, pp. 1–6.

[9] G. M. van de Ven and A. S. Tolias, "Three scenarios for continual learning," 2019, *arXiv:1904.07734*.

[10] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in Cognitive Sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[11] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in ser. Psychology of Learning and Motivation, G. H. Bower, Ed., vol. 24, Academic Press, 1989, pp. 109–165.

[12] W. C. Abraham and A. Robins, "Memory retention – the synaptic stability versus plasticity dilemma," *Trends in Neurosciences*, vol. 28, no. 2, pp. 73–78, 2005.

[13] S. Sodhani, M. Faramarzi, S. V. Mehta, *et al.*, "An introduction to lifelong supervised learning," 2022, *arXiv:2207.04354*.

[14] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, "Experience replay for continual learning," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32, Curran Associates, Inc., 2019.

[15] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[16] M. Farajtabar, N. Azizan, A. Mott, and A. Li, "Orthogonal gradient descent for continual learning," in *The 23rd International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 108, PMLR, 2020, pp. 3762–3773.

[17] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, PMLR, 2018, pp. 4548–4557.

[18] T. Flesch, J. Balaguer, R. Dekker, H. Nili, and C. Summerfield, "Comparing continual task learning in minds and machines," *Proceedings of the National Academy of Sciences*, vol. 115, no. 44, E10313–E10322, 2018.

[19] Y. Tian and Y. Zhang, "A comprehensive survey on regularization strategies in machine learning," *Information Fusion*, vol. 80, pp. 146–166, 2022.

[20] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[21] C. Meo, K. Sycheva, A. Goyal, and J. Dauwels, "Bayesian-LoRA: LoRA based Parameter Efficient Fine-Tuning using Optimal Quantization levels and Rank Values trough Differentiable Bayesian Gates," in *2nd Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization*, 2024.