Maintaining Plasticity for Deep Continual Learning

Activation Function-Adapted Parameter Resetting Approaches

Author:
Victor Purice
v.purice@student.tudelft.nl

Responsible Professor:
Dr.Wendelin Böhmer
j.w.bohmer@tudelft.nl

Supervisor:
Laurens Engwegen
l.r.engwegen@tudelft.nl

1.Introduction

Continual Learning (CL) - environment where a model learns sequentially from a stream of data, updating its knowledge while aiming to retain previously learned information.

Plasticity Loss - the phenomenon where a models' ability to learn and adapt to new information decreases over time, especially in CL scenarios.

2.Background

- Plasticity loss in **deep learning** can stem from a plethora of different factors, such as: the objective landscape sharpness [1], shifts in the effective rank dynamics [2], issues like vanishing gradients [3], and a high number of dormant/dead neurons [4].
- Standard deep learning utensils, in particular feedforward artificial neural networks and the backpropagation algorithm, fail to adapt to CL scenarios [5].
- A category of strategies that address the loss of plasticity in CL selectively **reinitializes the weights** of some neurons during the training process. For example, the Continual Backpropagation [6] algorithm reinitalizes neurons based on a heuristic measure of utility, assessing how valuable a neuron is to the network, and restricting the reinitialization to the less-used neurons.
- We design reinitialization approaches that leverage the intrinsic properties of the underlying **activation function** of the network. Most families of activation functions suffer from a few common **problematic regions** (see Figure 1). The lowest and highest activation values are typically detrimental to the learning performance, particularly in CL contexts.

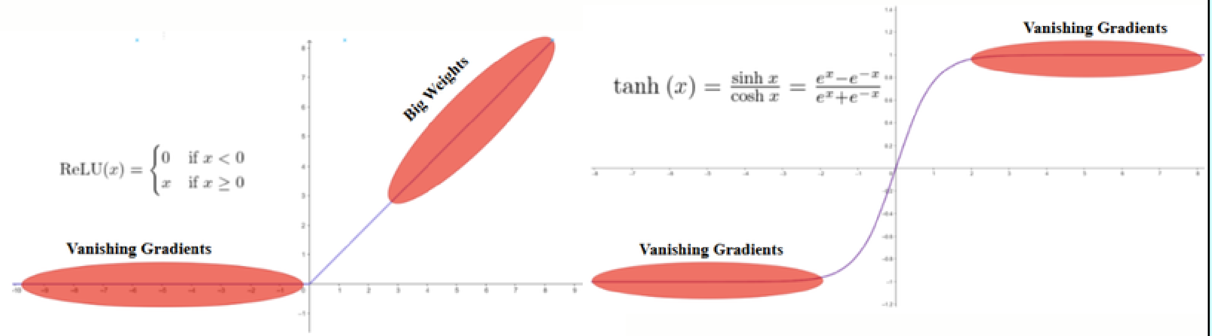

Figure 1: Illustration of some of the problems faced by deep learning activation functions.

3.Problem Description

- Supervised classification problem based on the **MNIST** dataset [1].
- The training process is divided into multiple **tasks** that are defined by the way the pixels within the context of the original image's shape are permuted (see Figure 2).
- **Base model:** artificial neural network with 3 hidden layers and 100 neurons per layer.
- The network is trained on a single pass through the data. For **each data point**, the network estimates the probabilities of each of the 10 classes and performs stochastic gradient descent on the cross-entropy loss.

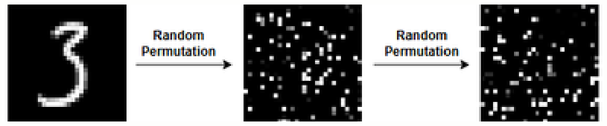

Figure 2: The process of generating different input configurations for each training task. Each particular image of the dataset is altered according to a unique random permutation.

4.Algorithm Design

The proposed algorithm reinitializes the neurons that yield the highest and lowest activation scores by rescaling their weights by some constant factor. To estimate a neuron's activation value, the following measurement metric is introduced:

$$u_l[i] \leftarrow \eta \cdot u_l[i] + (1 - \eta) \cdot a_{l,i,t}$$

where $u_l[i]$ is the utility score of the i-th neuron of layer l, $a_{l,i,t}$ is the the activation output of the i-th hidden unit in layer l at time t, and η represents a decay factor.

Let us denote by c_{low} and c_{high} the coefficients that will be used for rescaling the weights of the low and high utility score neurons. Additionally, let ρ_{low} and ρ_{high} denote the rates that will define how often the neurons will be reinitialized. The table $age_{l\,l=0}^{L-1}$ measures the number of steps since the last time that a neuron's weights have been rescaled, hence protecting the neurons from being rescaled too often. Additionally, two counters are introduced, which count the number of neurons that should be reinitialized at each timestep.

Algorithm 1 Proposed Strategy

Input: low/high replacement rates $\rho_{\text{low}}, \rho_{\text{high}}$; low/high coefficients $c_{\text{low}}, c_{\text{high}}$; decay rate η; maturity threshold m
Initialise weights $\{w_\ell\}_{\ell=0}^{L-1}$ with $w_\ell \sim d_\ell$
Initialise utilities $\{u_\ell\}_{\ell=0}^{L-1} \leftarrow 0$
Initialise ages $\{age_\ell\}_{\ell=0}^{L-1} \leftarrow 0$
Initialise counters $cnt_{\text{low},\ell\,\ell=0}^{L-1}, cnt_{high,\ell\,\ell=0}^{L-1} \leftarrow 0$
for each input x_t **do**
    $\hat{y}_t \leftarrow f(x_t; w)$ ▷ forward pass
    $\mathcal{L} \leftarrow \ell(y_t, \hat{y}_t)$ ▷ evaluate
    Update w with SGD (or variant) ▷ back-prop
    for $\ell = 1$ **to** $L - 1$ **do**
        $age_\ell \leftarrow age_\ell + 1$ ▷ age each unit
        Update u_ℓ using (2)
        $n_{\text{eligible}} \leftarrow \#\{i \mid age_{\ell,i} > m\}$
        $cnt_{\text{low},\ell} \mathrel{+}= \rho_{\text{low}}\, n_{\text{eligible}}$
        if $cnt_{\text{low},\ell} \geq 1$ **then**
            $r \leftarrow \arg\min_i u_{\ell,i}$ ▷ least-useful unit
            $w_{\ell-1}[:, r] \leftarrow c_{\text{low}} \cdot w_{\ell-1}[:, r]$ ▷ reset inputs
            $age_{\ell,r} \leftarrow 0$
            $cnt_{\text{low},\ell} \mathrel{-}= 1$
        end if
        $cnt_{\text{high},\ell} \mathrel{+}= \rho_{\text{high}}\, n_{\text{eligible}}$
        if $cnt_{\text{high},\ell} \geq 1$ **then**
            $r \leftarrow \arg\max_i u_{\ell,i}$ ▷ most-useful unit
            $w_{\ell-1}[:, r] \leftarrow c_{\text{high}} \cdot w_{\ell-1}[:, r]$ ▷ reset inputs
            $age_{\ell,r} \leftarrow 0$
            $cnt_{\text{high},\ell} \mathrel{-}= 1$
        end if
    end for
end for

The formalised pseudo-code of the proposed algorithm is presented in Algorithm 1.

A hyperparameter search was performed for the coefficients of a **ReLU-based strategy** corresponding to the model yielding the highest accuracy on a ReLU-activated network. Similarly, the parameters selected for the **tanh-based strategy** were chosen based on their optimal performance on a tanh-activated network.

Strategy	ρ_{high}	ρ_{low}	c_{big}	c_{small}
ReLU	10^{-2}, 10^{-3}, 10^{-4}, $\mathbf{10^{-5}}$, 10^{-6}, 10^{-7}	10^{-2}, 10^{-3}, $\mathbf{10^{-4}}$, 10^{-5}, 10^{-6}, 10^{-7}	-2, -1, -0.5, -0.2, 0, 0.2, $\mathbf{0.5}$, 1, 2	-2, -1, $\mathbf{-0.5}$, -0.2, 0, 0.2, 0.5, 1, 2
tanh	10^{-2}, 10^{-3}, 10^{-4}, $\mathbf{10^{-5}}$, 10^{-6}, 10^{-7}	10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, $\mathbf{10^{-6}}$, 10^{-7}	-2, -1, -0.5, -0.2, 0, $\mathbf{0.2}$, 0.5, 1, 2	-2, -1, -0.5, -0.2, 0, 0.2, $\mathbf{0.5}$, 1, 2

Table 1: Values used for the grid searches to find the best set of hyperparameters for the proposed algorithm tested on the permuted MNIST. The best-performing set of values for each activation strategy is in **bold**.

5. Results

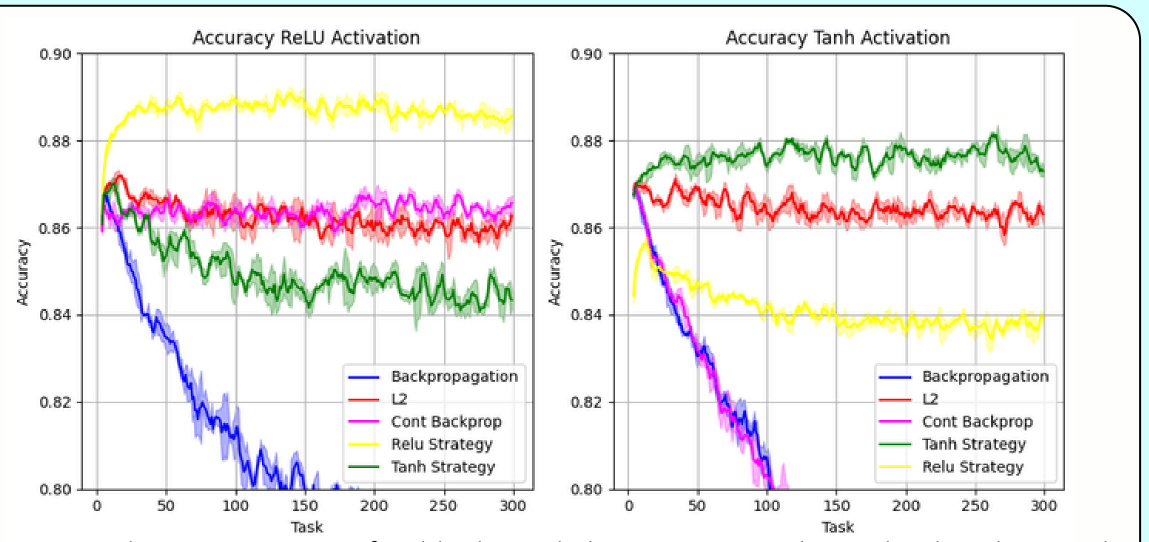

Figure 3: Online MNIST accuracy of models when applied to a ReLU-activated network and a tanh-activated network. The results represent the average and standard deviation taken over 3 seeds for a running window of size (n = 5).

- The accuracy of the simple backpropagation model decreases with each subsequent task, which showcases its lack of plasticity
- The L2 framework maintains its plasticity in both scenarios, thus serving as a baseline for preserving plasticity in CL.
- The Continual Backpropagation algorithm maintains the plasticity of the network when the hidden activation is ReLU, but loses plasticity when the hidden activation is tanh.
- The designed ReLU strategy outperforms the other strategies, achieving a performance of (88.65 ± 0.4%) (mean and std) across all tasks, compared to (86.23 ± 0.4%) of L2 regularisation and (84.98 ± 0.7%) of tanh strategy when used on a ReLU-activated network.
- The tanh strategy outperforms the other strategies, achieving a performance of (87.60 ± 0.32%) compared to (86.45 ± 0.37%) of L2 regularisation and (84.12 ± 0.5%) of ReLU when used on a tanh-activated network.
- Even though the performance of the strategies decreases when used on the opposite activation function scenario, the algorithms still preserve some plasticity of the model, as they perform better than simple backpropagation.

6. Conclusions

- This research introduced a framework which aims to maintain plasticity in CL, that **selectively** reinitializes neurons that yield the highest and the lowest activation values of the network, groups that are usually problematic in deep learning.
- A custom utility measure that estimates the activation value of each neuron was proposed.
- The strategy **successfully mitigates** loss of plasticity in simple supervised learning scenarios when used on a ReLU and tanh-activated network.

7. Future Work

- Does the proposed strategy mitigate plasticity loss in more general scenarios, including various **supervised learning** and deep **reinforcement learning** popular benchmarks?
- Is the model able to generalize to various activation functions, such as SeLU, GeLU, and others?
- What is the rate at which the model **forgets** previously acquired knowledge when exposed to new data distributions?

References

[1]Clare Lyle et al. Understanding plasticity in neural networks. 2023. arXiv: 2303 . 01486 [cs.LG]. URL: https://arxiv.org/abs/2303.01486.
[2]Aviral Kumar et al. Implicit Under-Parameterization Inhibits Data-Efficient Deep Reinforcement Learning. 2021. arXiv: 2010.14498 [cs.LG]. URL: https://arxiv. org/abs/2010.14498.
[3]Zaheer Abbas et al. Loss of Plasticity in Continual Deep Reinforcement Learning. 2023. arXiv: 2303 . 07507 [cs.LG]. URL: https : / / arxiv. org / abs / 2303 . 07507.
[4]Ghada Sokar et al. The Dormant Neuron Phenomenon in Deep Reinforcement Learning. 2023. arXiv: 2302. 12902 [cs.LG]. URL: https : / / arxiv. org / abs / 2302 . 12902.
[5] Shibhansh Dohare et al. "Loss of Plasticity in Deep Continual Learning". In: Nature 632 (2024), pp. 768– 774.
[6]Liyuan Wang et al. A Comprehensive Survey of Continual Learning: Theory, Method and Application. 2024. arXiv: 2302.00487 [cs.LG]. URL: https://arxiv.org/ abs/2302.00487.