A Self-Organizing Incremental Neural Network for Continual Supervised Learning

Chayut Wiwatcharakoses and Daniel Berrar

Data Science Laboratory, Tokyo Institute of Technology
2-12-1-S3-70 Ookayama, Meguro-ku, Tokyo 152-8550, Japan
Email: wiwatcharakoses.c.aa@m.titech.ac.jp
daniel.berrar@ict.e.titech.ac.jp

Abstract

Continual learning algorithms can adapt to changes of data distributions, new classes, and even completely new tasks without catastrophically forgetting previously acquired knowledge. Here, we present a novel self-organizing incremental neural network, GSOINN+, for continual supervised learning. GSOINN+ learns a topological mapping of the input data to an undirected network and uses a weighted nearest-neighbor rule with fractional distance for classification. GSOINN+ learns incrementally—new classification tasks do not need to be specified a priori, and no rehearsal of previously learned tasks with stored training sets is required. In a series of sequential learning experiments, we show that GSOINN+ can mitigate catastrophic forgetting, even when completely new tasks are to be learned.

Keywords: catastrophic forgetting; concept drift; continual learning; incremental learning; supervised learning

1. Introduction

A fundamental characteristic of how humans learn is that we acquire knowledge *incrementally* (Flesch et al., 2018; Rios & Itti, 2019). By contrast, the dominant machine learning paradigm is characterized by a distinct training phase and an application phase, which has been referred to as *isolated learning* (Chen & Guestrin, 2016). Once the training phase is completed, the model is ready for deployment in the specified application domain. For example, a conventional multi-layer perceptron (MLP) that was trained to classify images of cats and dogs can be used only for this specific purpose. The number of predefined classes (in this example, two) determines the structure of the output layer. But let us suppose now that the requirements change, and the model should be used to classify images of cats, dogs, *and* birds. It would be necessary to change the structure of the output layer and retrain the network from scratch in order to accommodate the new, third class. However, a complete retraining using old data is inefficient and often not even feasible (Li & Hoiem, 2016). In addition, machine learning models are generally designed for stationary environments, in which the process that generates the data is assumed to be stable over time. In many real-world scenarios, however, the data are generated by processes that evolve over time, so the underlying probability distribution may change and give rise to a phenomenon known as concept drift.

Continual learning (or *lifelong learning*) denotes the machine learning paradigm that considers algorithms that learn incrementally, so-called *continual learners*, which are capable of adapting to new classes and even new tasks. Continual learners can assimilate new input without compromising previously acquired knowledge, that is, without *catastrophic forgetting* (McCloseky & Cohen, 1989; Kirkpatrick et al., 2017). Continual learning is an inherently incremental process, without a sharp distinction between a training phase and an application phase. A further characteristic is *forward and backward transfer learning*: ideally, the model can leverage previously acquired knowledge to solve related new tasks; and conversely, new knowledge might improve the performance on old tasks (Parisi et al., 2019; Ruvolo & Eaton, 2013). Often, constraints are imposed on the learner's capacity to process and store knowledge; for example, if the model is based on a neural network, then the maximum number of nodes may be fixed to prevent boundless system growth.

In this study, we consider classification problems in which a continual learner has to solve two tasks sequentially, T_1 and T_2, from different domains. Each task is defined by learning a mapping $f_T : \mathcal{X}_T \rightarrow \mathcal{Y}_T$ from an instance space $\mathcal{X}_T \subseteq \mathbb{R}^m$ of m-dimensional instances (or cases) $\mathbf{x}_T$ to a set of j class labels $\mathcal{Y}_T = \{y_{T,1}, y_{T,2}, ..., y_{T,j}\}$. Importantly, the final model that solves the first classification task must be able to continue to learn and solve the second task without forgetting what it has learned before. Standard supervised learning algorithms, such as MLPs, are not suitable for this challenge, as the final, learned models can be used only for that task for which they were initially trained.

Our primary research goal is to develop a continual supervised learner that is able to solve a sequence of classification tasks from different domains without forgetting previously learned tasks. Here, we present a novel self-organizing incremental neural network, GSOINN+. In sequential learning experiments, we demonstrate that GSOINN+ can solve a classification task T_1 and then learn a completely different task, T_2, without catastrophically forgetting what it has learned before. GSOINN+ builds upon, and extends, our recently developed neural network, SOINN+ (Wiwatcharakoses & Berrar, 2019, 2020). SOINN+ was designed as an unsupervised method for learning a network topology from evolving data streams. Here, we introduce the concept of *ghost nodes* (lending the "G" in the new method's name). These virtual nodes play a pivotal role in the extension to supervised learning. Ghost nodes are created when real nodes die during training. They carry information about the class membership of deceased nodes and can be "summoned" when required. However, they have no influence on the evolution of the network during training. The novel contributions of this article can be summarized as follows.

1. We present a new neural network for continual supervised learning, GSOINN+. This network is able to solve learning tasks from different domains without detrimentally altering previously learned structures. GSOINN+ extends the unsupervised SOINN+ method to supervised learning. We also improve the learning algorithm by using the concept of fractional distance, which is particularly beneficial for high-dimensional feature spaces. In sequential benchmark experiments, GSOINN+ outperformed related continual learners, the *Associative Grow When Required* (AGWR) network (Parisi et al., 2015) and the *Associative Gamma-GWR* (AG-GWR) network (Parisi et al., 2017).

2. We introduce the concept of *ghost nodes*, which are virtual nodes carrying information about the class membership of previously deleted nodes. However, as virtual nodes, they have no role to play in the network evolution. Matlab source code is provided as supplemental information at the project website `https://osf.io/gqxya/`.

This article is organized as follows. After reviewing some related works, we focus on the algorithmic details of GSOINN+ and how it extends SOINN+. In Section 4, we describe our benchmark data sets and the learning experiments. Section 5 presents our results. We conclude the paper with a discussion and outlook at future work.

2. Related Work

Emulating human-like continual learning capabilities remains one of the grand challenges for AI. One step towards this goal is the mitigation of catastrophic forgetting when new tasks are being learned (Rios & Itti, 2019). Early approaches relied on memory-based systems that keep old training data for rehearsal; however, this approach is not efficient, and it might even be impossible when storing old data is not feasible (Lomonaco & Maltoni, 2017). Various alternatives have been proposed, which can be categorized as (*i*) regularization methods; (*ii*) complementary learning systems; and (*iii*) dynamic architectures (Parisi et al., 2019).

Regularization methods prevent drastic changes of those model parameters that are essential for maintaining a high performance on the old tasks when new tasks need to be learned. Aljundi et al. (2019) proposed a regularizer for active neurons, so that some capacity is reserved for future tasks. Penalizing weight updates in *elastic weight consolidation* (EWC) (Kirkpatrick et al., 2017) is another regularization technique. When the network faces a new learning task, EWC prevents the weight update for those nodes that are crucial for solving the old task. Complementary learning systems, such as the dual-memory self-organizing architecture (Parisi et al., 2018), are inspired by the episodic and semantic memory functions of the brain. In dynamic architectures, new system components are generated when new tasks need to be learned; for example, *progressive networks* grow new subnetworks sequentially (Rusu et al., 2016).

Another example of dynamic architectures are *self-organizing incremental neural networks* (SOINN) (Shen & Hasegawa, 2006; Shen et al., 2007; Shen & Hasegawa, 2008; Nakamura & Hasegawa, 2017). Briefly, a SOINN finds a topological mapping of the input data to a network of nodes by competitive learning. Learning means that nodes can move, merge with other nodes or remain as singletons, be deleted, and edges between nodes can be created or deleted. A node can be thought of as a microcluster of input cases that are close to each other because they share common traits, for example, they may belong to the same class. Nodes are connected by (undirected) edges if they are considered related, for example, if they belong to the same macrocluster. Conceptually, SOINN belong to the broad family of topology learners, which include *self-organizing maps* (Kohonen, 1982), *growing cell structures* (Fritzke, 1994a), *dynamic cell structures* (Bruske & Sommer, 1995), and *growing neural gas* (Fritzke, 1994b).

All previous SOINN variants have in common that nodes and edges are deleted at user-defined, fixed intervals. In our recently developed SOINN+, however, "forgetting" is an intrinsic part of the learning process (Wiwatcharakoses & Berrar, 2020). The deletion of edges and nodes no longer depends on arbitrary user-defined thresholds but on the current state of the network. Thereby, "forgetting" is learned. SOINN+ can detect clusters of arbitrary shapes in noisy data streams with concept drift; however, as an unsupervised method, it cannot be used for classification. To our knowledge, the methods that are conceptually most similar to SOINN+ are the *Grow When Required* (GWR) network (Marsland et al., 2002) and the *Gamma-GWR* network (Parisi et al., 2017). Like SOINN+, both of these networks are unsupervised learners. They delete edges that exceed a user-defined maximum age, and the nodes that thereby become unconnected are then removed. Parisi et al. extended GWR and Gamma-GWR to supervised methods, called *Associative GWR* (AGWR) (Parisi et al., 2015) and *Associative Gamma-GWR* (AG-GWR) (Parisi et al., 2017), respectively, by updating an associative matrix of class membership values for the nodes. The maximum number of nodes in AGWR and AG-GWR has to be specified by the user, whereas SOINN+ can grow indefinitely in theory.

Learning a SOINN+ network works as follows. The network is initialized with three randomly selected training cases. An m-dimensional input $\mathbf{x} = (x_1, x_2, ..., x_m)$, where x_i is the i^{th} feature value, is represented as a single node (singleton) in an undirected graph. New training cases are processed sequentially. After processing one training case, one iteration[1] is complete. Training cases are either added as new nodes, or they are merged with existing nodes, depending on a similarity threshold τ: if a new case is quite different from existing nodes, then it is considered as representing new knowledge and added as an unconnected node; if, on the other hand, a new case is close to an already existing node, then it is merged (Wiwatcharakoses & Berrar, 2020). A node can be linked to other nodes, or it can remain unconnected. Edges can only be created between first and second winner nodes after a new node has been merged with its first winner node. Whether an edge for a node $\mathbf{a}$ is created or not depends on the *trustworthiness* of that node: the more often $\mathbf{a}$ has been selected as first winner, the more trustworthy it is. Hence, trustworthiness is a function of a node's winning time (Wiwatcharakoses & Berrar, 2020). An edge is deleted if its lifetime exceeds an adaptive threshold λ_{edge}. This threshold depends on the lifetimes of previously deleted edges. Unconnected nodes are candidates for deletion. A node is deleted based on an *unutility* measure, which is calculated as the ratio of its idle time to its winning time. A high value of unutility means that the corresponding node was relatively rarely selected as first winner; hence, it is considered less important for the network and has a higher chance of being pruned as learning continues (Wiwatcharakoses & Berrar, 2020).

[1]The processing of one single training case causes an update of the network state. We refer to this update cycle as one *iteration* and not epoch because the latter term usually implies that all training cases have been processed.

3. GSOINN+: concepts and algorithms

GSOINN+ uses the same algorithms for adding, merging, and deleting nodes and edges as SOINN+. The main differences are the ghost nodes and the weighted nearest-neighbor rule based on the fractional distance for classification.

3.1. Fractional "distance"

How to measure closeness in high-dimensional space was identified as an open problem in SOINN+ (Wiwatcharakoses & Berrar, 2020). Here, we consider the Minkowski-like *fractional distance* as a possible solution. The fractional "distance" is an ℓ_p quasinorm for two m-dimensional vectors $\mathbf{a} = (a_1, a_2, ..., a_m)$ and $\mathbf{b} = (b_1, b_2, ..., b_m)$, when $0 < p < 1$,

$$\text{dist}_p(\mathbf{a}, \mathbf{b}) = \left(\sum_{i=1}^{m} |a_i - b_i|^p \right)^{\frac{1}{p}} \tag{1}$$

Strictly speaking, the fractional "distance" is no longer a proper distance metric because the triangle inequality is violated. However, it was shown to be more meaningful than the Euclidean metric to measure closeness in high-dimensional space (Aggarwal et al., 2001; François et al., 2007).

3.2. Ghost nodes

The basic notation used in the following equations and algorithms is as follows.

t	iteration of network update. $t \in \mathbb{N}$
$\mathbf{a}_{i,t}$	coordinates of node $\mathbf{a}_i$ at iteration t
$y_{\mathbf{a}}$	class label of node $\mathbf{a}$
$\mathbf{n}_1$	first nearest neighbor (= first winner)
$\mathbf{n}_2$	second nearest neighbor (= second winner)
$\mathbf{g}$	ghost node
$k_{\max}$	maximum number of nearest neighbors
h_k	number of ghost nodes of the k^{th} nearest neighbor
$\delta(\alpha, \beta)$	Kronecker delta; 1 if $\alpha = \beta$ and 0 otherwise
$\mathcal{G}_{\mathbf{a}}$	set of ghost nodes of node $\mathbf{a}$
$\mathcal{Y}^g_{\mathbf{a}}$	set of class labels of ghost nodes of node $\mathbf{a}$
$\mathcal{N}_1$	set of nodes with an edge to first winner $\mathbf{n}_1$
$WT(\mathbf{a})$	winning time of node $\mathbf{a}$. $WT \in \mathbb{N}$ (i.e., how often was $\mathbf{a}$ selected as first winner?)
$IT(\mathbf{a})$	idle time of node $\mathbf{a}$. $IT \in \mathbb{N}$ (i.e., for how many cycles was $\mathbf{a}$ *not* selected as the first winner?)
η	pull factor during node merging

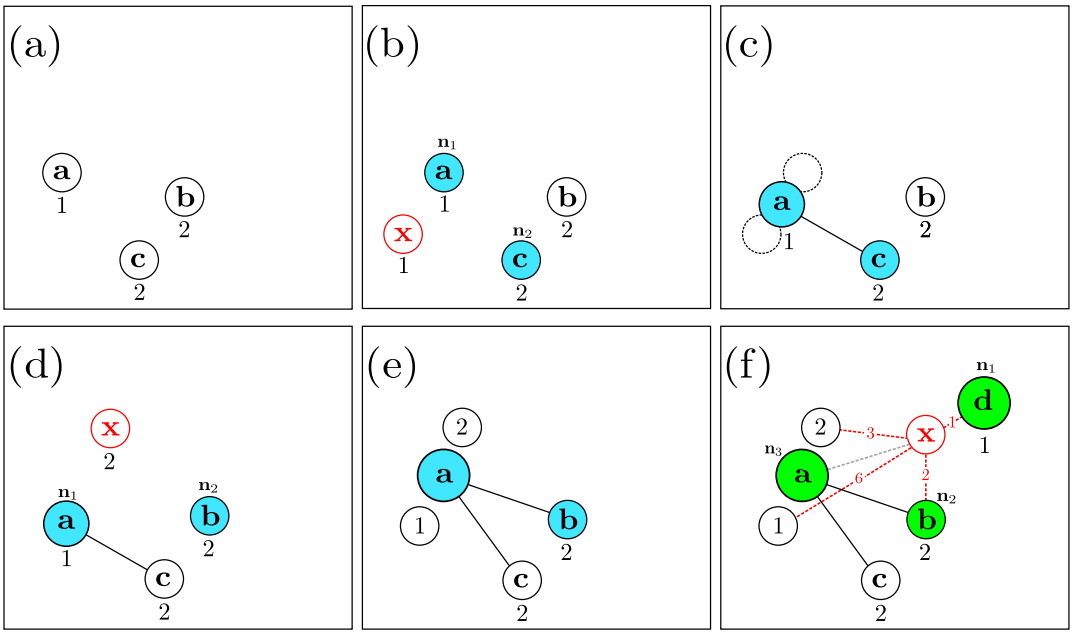

Figure 1: (a) A network of three nodes **a**, **b**, and **c** and associated class labels 1 and 2. (b) A new training sample, **x**, is presented to the network and the two nearest neighbors (here, **a** and **c**) are retrieved. (c) The new sample **x** is merged with its nearest neighbor, **a**. Each merging increases the inertia of a node (represented by larger circles). An edge between the two nearest neighbors is created. (d) Another new training case, **x**, is processed and its first and second nearest neighbors are retrieved (**a** and **b**, respectively). (e) **x** is merged with **a**. As **x** and **a** have different class labels (2 and 1, respectively), the class label of the merged node is unknown (marked by ?) and the original nodes **x** and **a** remain as *ghost nodes*. (f) This panel illustrates the classification of a test case **x** based on a k-nearest neighbor search, with $k_{\max} = 3$ (cf. Algorithm 2). As **a** has no class label, the labels of its ghost nodes are used, and $k_{\max} = 3 - 1 + 2 = 4$. The classification score for each class is calculated according to Eq. 4. The distances between **x** and the nodes are indicated by the dashed red lines. In this example, $S(y = 1|\mathbf{x}) = (\frac{1}{1} + 0 + 0 + \frac{1}{6})/(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{6}) = 0.583$ and $S(y = 2|\mathbf{x}) = (0 + \frac{1}{2} + \frac{1}{3} + 0/(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{6}) = 0.417$.

Ghost nodes are the central idea underpinning the extension to supervised learning. Although ghost nodes do not appear as real "living" nodes in the network and thus do not influence the network evolution, they solve the problem of keeping track of class labels when nodes of different classes are being merged. This idea is illustrated in Figure 1. The first panel shows a simple network of three, still unconnected nodes **a**, **b**, and **c** and their respective class labels 1 and 2. We consider now only two classes for illustration, but the idea generalizes to multiple classes.

When a new training case, **x**, is processed (Figure 1b), GSOINN+ retrieves its first and second nearest neighbors, which are **a** and **c** in this example. Nodes **a** and **c** are called "winner nodes" and labeled as $\mathbf{n}_1$ and $\mathbf{n}_2$, respectively. Based on a similarity threshold, **x** is considered close to **a** and therefore both nodes are merged. The similarity threshold $\tau(\mathbf{n}_i)$ for a winner node $\mathbf{n}_i$, $i = 1, 2$, is defined as follows (Wiwatcharakoses & Berrar, 2020):

$$\tau(\mathbf{n}_i) = \begin{cases} \max\limits_{\mathbf{a}_j \in \mathcal{N}_i} \parallel \mathbf{n}_i - \mathbf{a}_j \parallel & \text{if } \mathcal{N}_i \neq \emptyset \\ \parallel \mathbf{n}_i - \mathbf{m} \parallel & \text{otherwise} \end{cases} \tag{2}$$

where $\| \cdot \|$ denotes a distance; $\mathcal{N}_i$ is the set of nodes that are directly linked to node $\mathbf{n}_i$, $\mathbf{a}_j$ is a node in the network, and $\mathbf{m}$ is the second nearest neighbor of $\mathbf{n}_i$.

Furthermore, an edge between the merged node and the second nearest neighbor, $\mathbf{c}$, is created in Figure 1c. Merging implies that the original, to-be-merged nodes "die" and are replaced by a new (merged) node. In Figure 1c, the merged node is half-way between $\mathbf{n}_1$ and $\mathbf{x}$. But each time that a node $\mathbf{n}_1$ merges with another node, the inertia of $\mathbf{n}_1$ increases (represented by the slightly bigger node $\mathbf{a}$ in Figure 1c). The inertia of a node is quantified by its winning time. When an already merged node is the first winner and when it merges again with another training case, the position of the newly merged node will, therefore, be closer to the already merged node. In addition, all nodes that are linked to the first winner will be pulled slightly towards the new position (cf. lines 17-19 in Algorithm 1).

At iteration t, the coordinates of the first nearest neighbor are $\mathbf{n}_{1,t} = (n_{11,t}, n_{12,t}, ..., n_{1m,t})$. Like in SOINN+, the updated coordinates of this node after merging with node $\mathbf{x}$ are calculated as

$$\mathbf{n}_{1,t+1} = \mathbf{n}_{1,t} + \frac{\mathbf{x} - \mathbf{n}_{1,t}}{WT(\mathbf{n}_{1,t}) + 1} \tag{3}$$

where $WT(\mathbf{n}_1)$ denotes the winning time of node $\mathbf{n}_1$, that is, how often this node has been selected as the first nearest neighbor. Since both $\mathbf{a}$ and $\mathbf{x}$ have the same class label (here, 1 in Figure 1c), the same label is also assigned to the merged node.

However, there is a problem when two nodes are merged with *different* class labels. This scenario is shown in Figure 1d. Here, the new training case $\mathbf{x}$ is a member of class 2. The case is closest to node $\mathbf{a}$ and therefore merged (Figure 1e). But which class label should we assign to the newly merged node? The training case $\mathbf{x}$ has class label 2, whereas $\mathbf{a}$ has class label 1. Therefore, the class label of the newly merged node is uncertain, which is represented by the question mark in Figure 1e.

To solve this problem, two ghost nodes are created (represented by the white, unnamed circles in Figure 1e). The coordinates of these ghosts are the same as the coordinates of the nodes *before* merging. Loosely speaking, merging of two nodes implies that the original nodes cease to exist, but here the ghosts of the original nodes remain. Being ghosts, however, they have no further role to play in the network evolution; for example, they cannot be linked to other nodes or influence their inertia. However, they remember the original class labels and can be "summoned" when required. This is illustrated in Figure 1f where $\mathbf{x}$ represents a test case that is to be classified. The network consists of nodes $\mathbf{a}$ (with two ghost nodes), $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{d}$ (with two ghost nodes).

The classification is based on a weighted k-nearest neighbor rule. In Figure 1f, we consider the Euclidean distance and use $k = 3$ only for the purpose of illustration. The first three nearest neighbors of $\mathbf{x}$ are $\mathbf{a}$, $\mathbf{d}$,

and **b**. The class label of **b** is 2, but the labels of **a** and **d** are unknown. However, the nearest ghost node of **a** has the label 2, and the nearest ghost node of **d** has the label 1. Thus, a classification based on the inverse distances to the ghosts (shown as red dotted lines in Figure 1f) becomes possible (Algorithm 1).

Algorithm 1: GSOINN+: merging nodes and creating ghost nodes.

$\mathbf{x} \leftarrow$ new training case
$y_{\mathbf{x}} \leftarrow$ class label of $\mathbf{x}$
$\mathbf{n}_1 \leftarrow$ first winner of $\mathbf{x}$
$WT(\mathbf{n}_1) \leftarrow WT(\mathbf{n}_1) + 1$ // Increment winning time of first winner node.
if class label of $\mathbf{n}_1 \neq y_{\mathbf{x}}$ **then**
  if set of ghost nodes of $\mathbf{n}_1$ is empty, $\mathcal{G}_{\mathbf{n}_1} = \emptyset$, **then**
    add $\mathbf{n}_1$ to set of ghost nodes: $\mathcal{G}_{\mathbf{n}_1} \cup \{\mathbf{n}_1\}$
    add class label of $\mathbf{n}_1$ to set ghost labels: $\mathcal{Y}^g_{\mathbf{n}_1} \cup \{y_{\mathbf{n}_1}\}$
  add $\mathbf{x}$ to set of ghost nodes: $\mathcal{G}_{\mathbf{n}_1} \cup \{\mathbf{x}\}$
  add class label of $\mathbf{x}$ to set of ghost labels: $\mathcal{Y}^g_{\mathbf{n}_1} \cup \{y_{\mathbf{x}}\}$
$\mathbf{n}_1 \leftarrow \mathbf{n}_1 + \frac{\mathbf{x}-\mathbf{n}_1}{WT(\mathbf{n}_1)}$ // Merge test case and first winner node.
$\mathcal{N}_1 \leftarrow$ set of nodes with an edge to $\mathbf{n}_1$
$\eta \leftarrow 100$ // Pull factor.
for all nodes in $\mathcal{N}_1$ **do**
  $\mathbf{a}_i \leftarrow i^{th}$ node in $\mathcal{N}_1$
  $\mathbf{a}_i \leftarrow \mathbf{a}_i + \frac{\mathbf{x}-\mathbf{a}_i}{\eta WT(\mathbf{a}_i)}$ // Pull connected nodes.
$IT(\mathbf{n}_1) \leftarrow 0$ // Reset idle time.

Given a test case $\mathbf{x}$, GSOINN+ calculates the class membership score $S(y|\mathbf{x})$ for a class y as

$$S(y|\mathbf{x}) = \frac{\sum_{k=1}^{k_{\max}} \left[[\mathrm{dist}(\mathbf{x}, \mathbf{n}_k)]^{-1} \ \delta(h_k, 0) \ \delta(y_k, y) + \sum_{i=1}^{h_k} [\mathrm{dist}(\mathbf{x}, \mathbf{g}_{ik})]^{-1} \ \delta(y_{\mathbf{g}_{ik}}, y) \right]}{\sum_{k=1}^{k_{\max}} \left[[\mathrm{dist}(\mathbf{x}, \mathbf{n}_k)]^{-1} \ \delta(h_k, 0) + \sum_{i=1}^{h_k} [\mathrm{dist}(\mathbf{x}, \mathbf{g}_{ik})]^{-1} \ \delta(y_{\mathbf{g}_{ik}}, y) \right]} \tag{4}$$

where $k_{\max}$ denotes the user-defined maximum number of nearest neighbors; $\mathrm{dist}(\mathbf{x}, \mathbf{n}_k)$ is the distance between $\mathbf{x}$ and the k^{th} nearest neighbor, and y_k is its class label; $\delta(\alpha, \beta) = 1$ if $\alpha = \beta$ and 0 otherwise; h_k is the number of ghost nodes of the k^{th} nearest neighbor; $\mathbf{g}_{ik}$ is the i^{th} ghost node of the k^{th} nearest neighbor; and $y_{\mathbf{g}_{ik}}$ is its class label. Note that the second sum in the numerator (and denominator) is empty for all nearest neighbors that carry a class label (i.e., have no ghost nodes). The score $S(y|\mathbf{x})$ is normalized between $[0, 1]$, but it is not a posterior class probability, since it is based on the inverse distances to the nearest neighbors. Algorithm 2 is the pseudocode for calculating the classification score in GSOINN+.

Our primary research goal is to develop a continual supervised learner that learns to solve classification tasks from different domains without catastrophic forgetting. More specifically, we are interested in the following questions:

1. What is the effect of the distance metric used to measure closeness between nodes when the network

Algorithm 2: Classification score.

$k_{\max} \leftarrow$ maximum number of nearest neighbors
$\delta(\alpha, \beta) \leftarrow 1$ if $\alpha = \beta$ and 0 otherwise
$\text{dist}(\mathbf{a}, \mathbf{b}) \leftarrow$ Eq. 1 (distance metric)
$sum_k \leftarrow 0$
$sum_m \leftarrow 0$
$y \leftarrow$ class label for which the score is to be computed
for k from 1 to $k_{\max}$ **do**
    $\mathbf{n}_k \leftarrow k^{th}$ nearest neighbor of $\mathbf{x}$
    $h_k \leftarrow$ number of ghost nodes of $\mathbf{n}_k$
    if $h_k > 0$ **then**
        for i from 1 to h_k **do**
            $\mathbf{g}_{ik} \leftarrow i^{th}$ ghost node of $\mathbf{n}_k$
            $y_{\mathbf{g}_{ik}} \leftarrow$ class label of $\mathbf{g}_{ik}$
            $sum_k \leftarrow sum_k + [\text{dist}(\mathbf{x}, \mathbf{g}_{ik})]^{-1} \cdot \delta(y, y_{\mathbf{g}_{ik}})$
            $sum_m \leftarrow sum_m + [\text{dist}(\mathbf{x}, \mathbf{g}_{ik})]^{-1}$
    else
        $y_k \leftarrow$ class label of $\mathbf{n}_k$
        $sum_k \leftarrow sum_k + [\text{dist}(\mathbf{x}, \mathbf{n}_k)]^{-1} \cdot \delta(y, y_k)$
        $sum_m \leftarrow sum_m + [\text{dist}(\mathbf{x}, \mathbf{n}_k)]^{-1}$
$S(y|\mathbf{x}) \leftarrow sum_k \cdot (sum_m)^{-1}$

is learned? Are there performance differences, depending on the distance metric? What is the effect of k, the number of nearest neighbors used for classification?

2. Can GSOINN+ mitigate catastrophic forgetting, that is, can the model solve an entirely new classification task and still perform well on the previously learned task?
3. How does GSOINN+ perform against similar methods, AGWR (Parisi et al., 2015) and AG-GWR (Parisi et al., 2017)?

4. Materials and Methods

4.1. Data sets

We used three benchmark data sets: the MNIST and Balanced data sets (Cohen et al., 2017) and the Fashion-MNIST data set (Xiao et al., 2017). The entire MNIST data set contains 70 000 images of handwritten numbers 0 to 9. The training set, MNIST-train, contains 60 000 images (6000 images per class) and the test set, MNIST-test, contains 10 000 images (1000 images per class). The Balanced data set consists of images of handwritten numbers (0 to 9) and the upper- and lowercase letters A to Z. Lowercase letters that look identical to their uppercase letters were removed; the discarded characters are c, i, j, k, l, m, o, p, s, u, v, w, x, y, and z. The training set, Balanced-train, has 112 800 images (2400 images per class) and the test set, Balanced-test, has 18 800 images (400 images per class). The Fashion-MNIST data set contains 70 000 images of various garments from 10 classes, with 7000 images per class (Xiao et al., 2017). The training set, Fashion-train, has 60 000 images (6000 per class) and the test set, Fashion-test, has 10 000 images (1000 per class). In all data sets, each image is represented as a 28×28 matrix of gray-scale pixels, which we pre-processed into a 1-dimensional feature vector of length 784 by concatenating the rows.

4.2. Experiments

We designed two different types of experiments to evaluate the performance of GSOINN+ and its ability to mitigate catastrophic forgetting. The goal of the *single-task experiment* was to address our first question, that is, to assess the classification performance as a function of the new distance metric (Equation 3.1, for different fractional distances $p = 0.1, 0.3, 0.5, 0.7, 0.9$, and $p = 1$ for the Manhattan distance, and $p = 2$ for the Euclidean distance) and the number of nearest neighbors ($k = 1, 2, 3, 4, 5$) to be used for the classification score (Equation 4). Figure 2 shows a schematic overview of the single-task learning experiments.

To address the second question and test our hypothesis that GSOINN+ can mitigate catastrophic forgetting, we designed a series of *sequential learning experiments* (Figure 3). First, we trained GSOINN+ on the training set A_{train} from domain A. After processing all training cases only once, we immediately continued with processing the training set B_{train} from domain B. Then, we applied the resulting model to the corresponding test sets: first, A_{test} and second, B_{test}. Then, we reversed the training sequence and applied the resulting model again to the test sets. Hence, for each pair $(A_{\text{train}}, B_{\text{train}})$, there are four sequences:

1. $A_{\text{train}} \rightarrow B_{\text{train}} \rightarrow A_{\text{test}}$

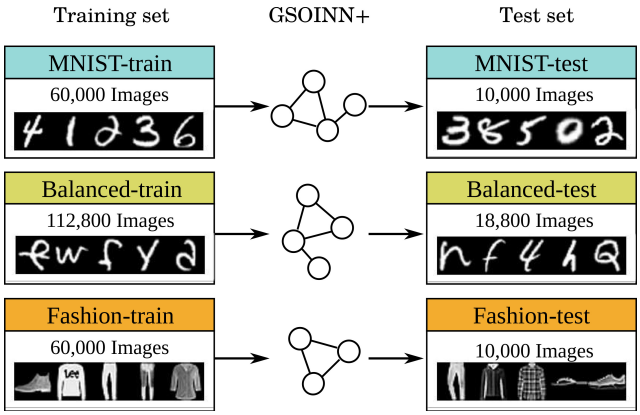

Figure 2: Single-task learning experiments. For each data set, five randomly selected images are shown as examples.

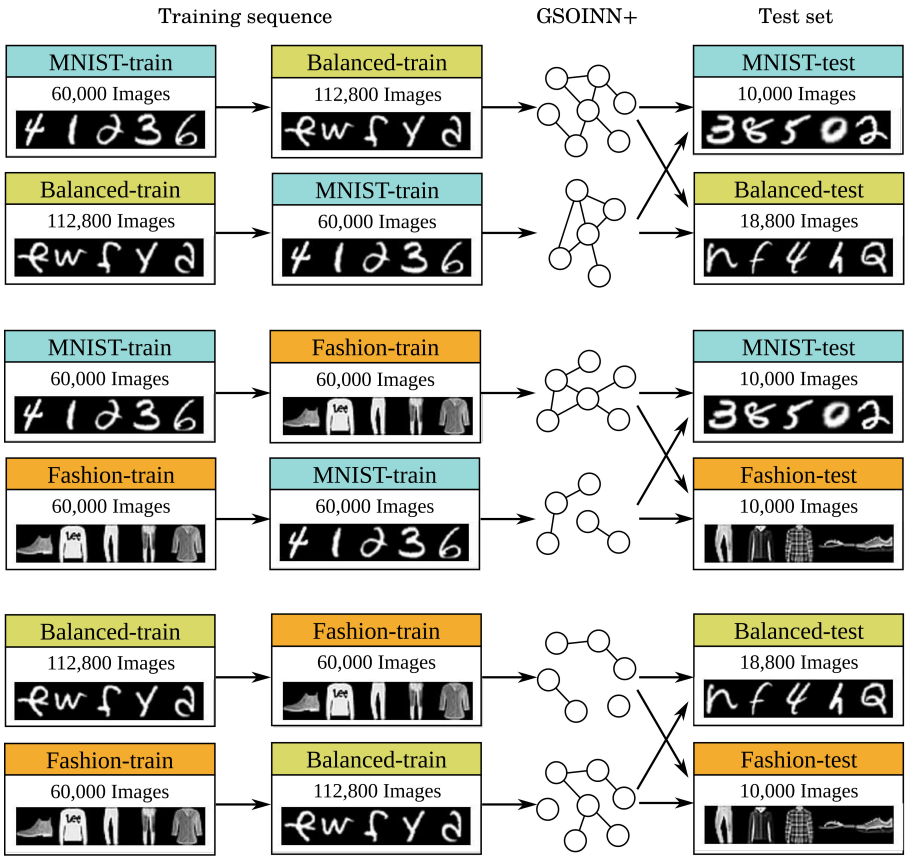

Figure 3: Sequential tasks learning experiments. For each data set, five randomly selected images are shown as examples.

2. $A_{\text{train}} \rightarrow B_{\text{train}} \rightarrow B_{\text{test}}$
3. $B_{\text{train}} \rightarrow A_{\text{train}} \rightarrow A_{\text{test}}$
4. $B_{\text{train}} \rightarrow A_{\text{train}} \rightarrow B_{\text{test}}$

Here, the notation "$A_{\text{train}} \rightarrow B_{\text{train}}$" means that the network is first built by processing all training cases from A_{train}. The resulting network then immediately faces training cases from B_{train}. The notation "$\rightarrow A_{\text{test}}$" means that the final model is then applied to classify the test cases from A_{test}.

The rationale for these sequences is the following. If our hypothesis is false (i.e., if the model forgets catastrophically), then the performance on A_{test} in sequence #1 should be substantially worse than that in sequence #3. Under the assumption of catastrophic forgetting, learning a new task would severely affect or even erase the knowledge that was learned before. Similarly, the performance in sequence #4 should be substantially worse than that in sequence #2. By contrast, if our hypothesis is true (i.e., the model does *not* forget catastrophically), then the performance in sequences #1 and #3 and in sequences #2 and #4, respectively, should be similar. In other words, the order of learning tasks should not have a strong influence on the performance. We considered all combinations of sequences for MNIST-train, Balanced-train, and Fashion-train, leading to six different experiments (Figure 3). We used exactly the same experiments for the comparison with AGWR (Parisi et al., 2015) and AG-GWR (Parisi et al., 2017).

5. Results

5.1. Single task experiments

Figure 4 shows the average classification accuracies for the single-task learning experiments. In all three test sets, the performance was lowest for the Euclidean distance ($p = 2$). On MNIST-test and Balanced-test, the best performances were achieved for fractional distances $p = 0.5$; on Fashion-test, the performance was marginally better for $p = 0.3$. In all experiments, the lowest performance was observed for $k = 1$, with only minute differences for $k = 3, 4, 5$. The highest accuracy on MNIST-test was 0.95 (for $p = 0.3, 0.5$ and $k = 2, 3, 4, 5$). The highest accuracy on Balanced-test was 0.71 (for $p = 0.5$ and $k = 3, 4, 5$). The highest accuracy on Fashion-test was 0.81 (for $p = 0.3$ and $k = 3, 4, 5$).

For AGWR and AG-GWR, we used the default parameter settings for the maximum age of edges ($\mu_{\max} = 600$) and the maximum number of nodes ($n_{\max} = 10000$) (Parisi, 2020). Table 1 shows the point estimates for the accuracies on the test sets, together with their 95% Bayesian posterior highest density intervals (HDI) (assuming flat priors) for the out-of-sample prediction accuracy. Table 1 also shows the results for GSOINN+ with fractional distance $p = 0.5$ and $k = 3$ nearest neighbors.

It is meaningful to define a *region of practical equivalence* (ROPE) around the null value of the parameter of interest, i.e., the difference in performance, $\theta = 0$ (Benavoli et al., 2017; Kruschke & Liddell, 2018; Kruschke, 2015). If the posterior HDI of the parameter lies completely inside the ROPE, then all values with the highest credibility are practically equivalent with the null value, and the null value is therefore

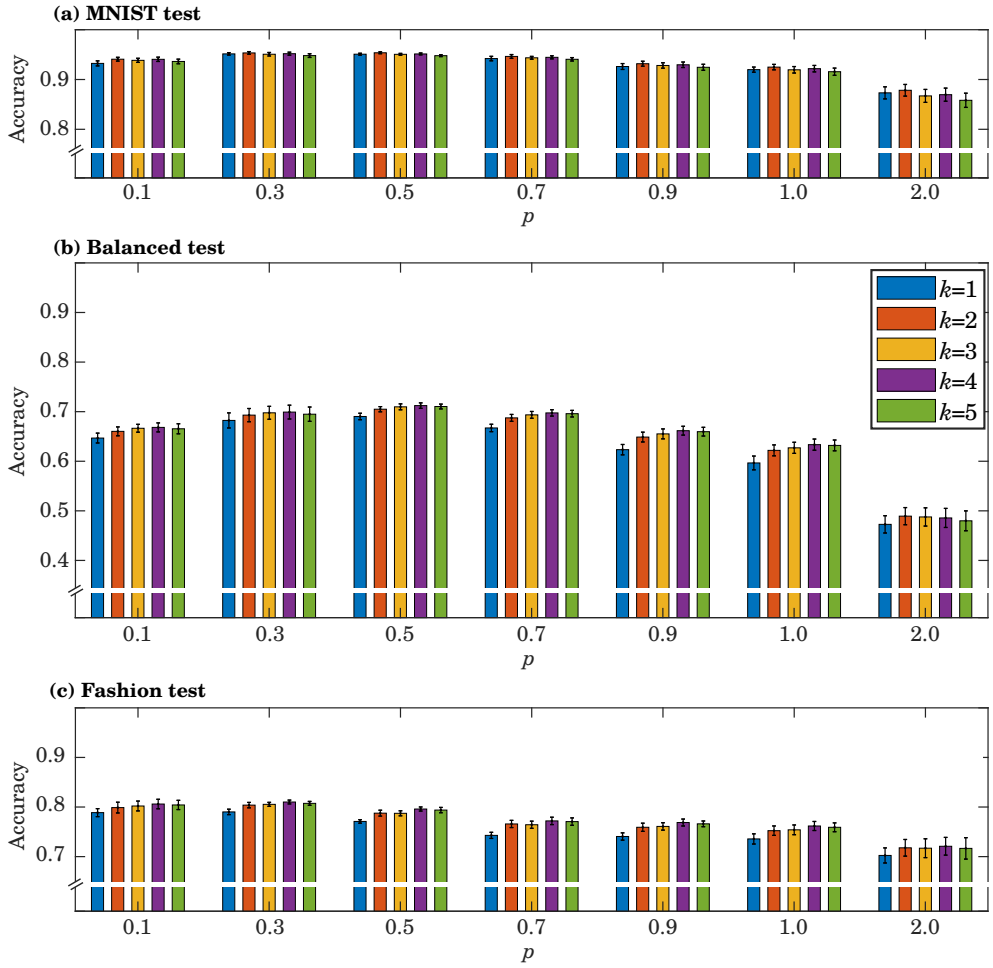

Figure 4: Average classification accuracy for single-task learning experiments. The experiments were repeated ten times for different random initializations of GSOINN+. Fractional distances are $p = 0.1, 0.3, 0.5, 0.7, 0.9$, Manhattan distance ($p = 1.0$) and Euclidean distance ($p = 2$). The number of nearest neighbors is $k = 1, 2, 3, 4, 5$.

accepted. If, on the other hand, the HDI falls completely outside the ROPE, then the null value is rejected, since all practically equivalent values have low credibility. If the ROPE and HDI only partially overlap, then no decision is made. The width of the ROPE should be specified for the application at hand. For the comparison of classifiers based on accuracy, Benavoli et al. (2017) recommend a ROPE of $[-0.01, 0.01]$. Thus,
260 if the HDI for the difference in performance is outside this interval, then we conclude that the performances are not practically equivalent. Based on this criterion, GSOINN+ performs better than AGWR and AG-GWR on all three test sets. On Fashion-test, the accuracies of AGWR and GSOINN are most similar, with 0.775 (95%-HDI $[0.767, 0.783]$) and 0.787 (95%-HDI $[0.779, 0.795]$), respectively. The HDI for the difference in accuracy is $[0.0104, 0.0147]$, which is just outside the ROPE of $[-0.01, 0.01]$.

Table 1: Classification accuracy (with 95% Bayesian posterior highest density interval (HDI) for the true out-of-sample accuracy) on the test sets for AGWR and AG-GWR ($\mu_{\max} = 600$, $n_{\max} = 10000$) and GSOINN+ ($p = 0.5$, $k = 3$) in the single-task learning experiments.

Training set	**Test set**	**AGWR**	**AG-GWR**	**GSOINN+**
MNIST-train	MNIST-test	0.908 [0.902, 0.914]	0.893 [0.887, 0.899]	0.951 [0.946, 0.955]
Balanced-train	Balanced-test	0.534 [0.527, 0.541]	0.514 [0.506, 0.521]	0.710 [0.703, 0.716]
Fashion-train	Fashion-test	0.775 [0.767, 0.783]	0.744 [0.735, 0.753]	0.787 [0.779, 0.795]

5.2. Sequential learning experiments

For the sequential learning experiments, we used the fractional distance metric with $p = 0.5$. For the number of nearest neighbors, we considered $k = 1, 2, 3, 4, 5$. Table 2 shows the classification accuracy on the test sets as described in Figure 3b.

Table 2: Classification accuracy of GSOINN+ ($p = 0.5$) on the test sets for the sequential learning experiments.

Training sequence	**Test set**	***k*=1**	***k*=2**	***k*=3**	***k*=4**	***k*=5**
MNIST-train→Balanced-train	MNIST-test	0.85	0.86	0.88	0.88	0.89
	Balanced-test	0.70	0.70	0.70	0.71	0.70
Balanced-train→MNIST-train	MNIST-test	0.86	0.88	0.90	0.90	0.91
	Balanced-test	0.68	0.69	0.69	0.69	0.69
MNIST-train→Fashion-train	MNIST-Test	0.95	0.95	0.95	0.95	0.95
	Fashion-test	0.80	0.81	0.81	0.82	0.81
Fashion-train→MNIST-train	MNIST-Test	0.95	0.95	0.94	0.94	0.94
	Fashion-test	0.77	0.79	0.79	0.80	0.79
Balanced-train→Fashion-train	Balanced-test	0.68	0.69	0.70	0.70	0.70
	Fashion-test	0.80	0.81	0.81	0.81	0.81
Fashion-train→Balanced-train	Balanced-test	0.68	0.69	0.70	0.70	0.70
	Fashion-test	0.77	0.79	0.79	0.80	0.79

Let us focus on $k = 3$ (Table 2, highlighted) and the first training sequence, MNIST-train→Balanced-train. The accuracy on the MNIST-test set is 0.88, which is lower than that from the single-task experiments (0.95, Figure 4a). Apparently, the additional task of learning to classify Balanced-train (after MNIST-train) has led to a decreased performance on MNIST-test, but clearly *not* to a catastrophic forgetting. The performance on Balanced-test is 0.70 and on par with that from the single-task experiment (0.71, Figure 4a).

Recall that MNIST-train does not contain any characters, only numbers. When we investigated the classification confusion matrix (Figure 5) for the final model (i.e., after processing both MNIST-train and Balanced-train), we found that many of the errors occurred for letters and numbers that look similar; for example, there are 1000 zeroes ("0") in MNIST-test, and 119 of these were misclassified as the letter "O". Ninty-five times, "1" was misclassified as "I", and 67 times, "5" was misclassified as "S". It is plausible

that the model makes these errors after learning the second task, given that these particular characters and numbers are indeed barely distinguishable.

For the reversed training sequence, Balanced-train→MNIST-train, we observe that GSOINN+ performs slightly better on MNIST-test, with an accuracy of 0.90; conversely, the performance on Balanced-test has slightly dropped to 0.69. The differences, however, are only marginal and indicate that the order of the learning tasks does indeed not entail a complete forgetting of the first task.

Consider now the third training sequence, MNIST-train→Fashion-train. The performance on MNIST-test did not deteriorate after the model was trained on Fashion-train, since the accuracy of 0.95 is the same as in Figure 4a. So even after learning a task from a completely different domain (classification of images of garments), the knowledge for solving the earlier task (classification of images of handwritten numbers) is still available. Also, the performance on Fashion-test is comparable to that in Figure 4c.

For the reversed training sequence, Fashion-train→MNIST-train, the performance on MNIST-test is comparable to that in sequence MNIST-train→Fashion-train; however, the performance on Fashion-test has dropped slightly. A similar effect is observed for the last two sequences. For Balanced-train→Fashion-train, the performance on Balanced-test is the same as in the single-task learning experiments (Figure 4a), which means that learning the new task of Fashion-train did not overwrite the knowledge gleaned from Balanced-train. The performance on Fashion-test is also the same as before (Figure 4c); again, this is not unexpected, given that Fashion-train was learned the last. For the reversed sequence, Fashion-train→Balanced-train, the performance on Balanced-test did not change; however, we observe again a drop in accuracy between 1% and 3% for Fashion-test. For all sequences, the number of nearest neighbors plays a negligible role, as the accuracies on the test sets are comparable.

Taken together, these results corroborate our hypothesis: learning a new task does not lead to a catastrophic forgetting of previously acquired knowledge in GSOINN+.

Next, we carried out the same experiments with AGWR and AG-GWR. For both methods, the important hyperparameter is the user-defined maximum age of edges. First, we used the default value of $\mu_{\max} = 600$ (Parisi, 2020), which means that edges whose age exceeds this value are deleted. Nodes that

Predicted Class \ MNIST-test Real Class	0	1	2	3	4	5	6	7	8	9
1	832	7	17	5	5	2	8	5	6	9
2	7	799	2		10	1		8	4	6
3		2	880	4	3			2	2	1
4		1	2	959		12			25	2
5	1		1		850	2		4	5	8
6				4		884	2		18	3
7	4			1	2	3	936		2	
8			18	4	3			943	3	21
9			5	3		1			852	1
A				3	22	1		9	10	848
B					4					
C				1					22	
D	5		1		1	4	9		1	
E	18		2			1				1
F			1			5			3	
G					2	1			3	
H			1			1	9			
I					3					1
J	1	95	2		3		1	2	2	
K		2	5	4	1	7				
L										
M	1	91	4		3			2	1	2
N										1
O			1		1					
P	119						1			
Q									6	2
R	1		2						1	3
S			3						1	
T				11		67			2	2
U					1	2		10		1
V	4				15		1			2
W	1		2		7				5	
X							1			
Y									1	1
Z					20			2	2	3
a			42						1	
b	3		2		4				1	2
d			1			2	27		3	
e		1	4	1	4	1	1			
f	1		1						2	
g					2			2		
h			1		2				5	24
n					5		4		1	
q	1				2			2		3
r					3			1	1	53
t	1	1			2	3		1	9	
		1			20			7		

Figure 5: Confusion matrix of GSOINN+ for MNIST-test from the sequence MNIST-train→Balanced-train. Most errors occurred due to "0" being mistaken as "O", "1" being mistaken as "I", and "5" being mistaken as "S".

thereby become fully unconnected are deleted as well. Like for GSOINN+, we presented each training case only once to AGWR and AG-GWR. Both models achieved test set performances that are comparable to those observed in the single-task learning experiment (Table 1), but only if the corresponding training sets appeared *last* in the training sequence. For example, in the sequence Balanced-train→Fashion-train, the accuracy on Fashion-test is 0.78 for AGWR and 0.74 for AG-GWR (Table 3). However, the accuracy on Balanced-test dropped to 0%, which means that the knowledge from Balanced-train has become completely overwritten. This effect can be explained by the fact that the age of *all* edges is incremented after each training cycle. In the network that was learned for the first task, the age of the edges are being incremented while new structures are learned for the second task. Eventually, all those edges that exceed the maximum age are deleted, which leads to a complete forgetting of the first task.

We then repeated the experiments, forcing the networks not to forget by setting $\mu_{\max}$ to $+\infty$; thereby, no edges (and consequently no nodes) could be deleted anymore. The maximum number of nodes was set to be equal to the number of nodes (including ghosts) in GSOINN+ for the respective training sequences. This "enforced remembering" came at a high computational cost: on a standard PC, one sequence involving MNIST-train and Fashion-train took around one day, and one sequence involving Balanced-train took around five days (implementation in Python). Interestingly, for $\mu_{\max} = +\infty$, AGWR and AG-GWR do not necessarily remember the last task the best. For example, consider the sequence MNIST-train→Fashion-train (Table 3). Both models achieved an accuracy of 0.95 on MNIST-test. However, when the training sequence was reversed, the performances dropped to 0.82 and 0.79, respectively. Like for GSOINN+, we had expected that the last training set would be remembered better than the first one, but that was not the case: a better performance was achieved for the test set of the *first* task. As a possible explanation, the models might have used up the allocated maximum number of nodes during the first learning task, so that insufficient resources were available for learning the second task.

An important factor that determines performance, specifically for high-dimensional data sets, is the choice of the distance metric. To classify a test case, GSOINN+ uses the fractional distance (Equation 3.1), whereas AGWR and AG-GWR use the Euclidean metric. It is tempting to speculate that the performances of AGWR and AG-GWR could be improved by using the fractional distance.

6. Discussion and Conclusions

We developed a novel self-organizing incremental neural network, GSOINN+, for continual supervised learning. GSOINN+ is based on our recently proposed unsupervised SOINN+ (Wiwatcharakoses & Berrar, 2020), which learns a topological mapping of the training data to a network structure. GSOINN+ extends SOINN+ to supervised learning via the new concept of ghost nodes. As virtual nodes, they play no role in the evolution of the network structure during training; however, they enable the network to retrieve those class labels that otherwise would be lost due to the merging of nodes with different labels.

Table 3: Classification accuracy on the test sets for AGWR and AG-GWR in the sequential learning experiments, for the default maximum age of edges ($\mu_{\max} = 600$) and for "enforced remembering" ($\mu_{\max} = +\infty$).

Training sequence	**Test set**	**AGWR** $\mu_{\max}$=default	**AG-GWR** $\mu_{\max}$=default	**AGWR** $\mu_{\max}$=$+\infty$	**AG-GWR** $\mu_{\max}$=$+\infty$
MNIST-train→Balanced-train	MNIST-test	0.57	0.54	0.81	0.79
	Balanced-test	0.56	0.51	0.72	0.69
Balanced-train→MNIST-train	MNIST-test	0.90	0.88	0.89	0.86
	Balanced-test	0.21	0.22	0.71	0.68
MNIST-train→Fashion-train	MNIST-Test	0.01	0.01	0.95	0.95
	Fashion-test	0.69	0.69	0.67	0.64
Fashion-train→MNIST-train	MNIST-Test	0.86	0.89	0.82	0.79
	Fashion-test	0.01	0.03	0.82	0.80
Balanced-train→Fashion-train	Balanced-test	0.00	0.00	0.72	0.70
	Fashion-test	0.78	0.74	0.69	0.65
Fashion-train→Balanced-train	Balanced-test	0.55	0.51	0.71	0.67
	Fashion-test	0.03	0.05	0.83	0.79

We evaluated the performance of GSOINN+ in two different experimental settings, (*i*) single task experiments and (*ii*) sequential learning experiments. The single task experiments were standard classification experiments using the benchmark data sets MNIST, Balanced, and Fashion-MNIST (Cohen et al., 2017; Xiao et al., 2017). Here, no learning across disparate domains was required. We compared the classification accuracy of GSOINN+ with that of two related methods, AGWR and AG-GWR. Based on a region of practical equivalence of ±1% for the difference in performance, GSOINN+ performed better than its competitors in these experiments.

Our primary research goal was to develop a continual learner that is able to mitigate catastrophic forgetting when entirely new tasks are to be learned. We therefore carried out a series of sequential learning experiments where we reversed the order of the training sets. The challenge was to maintain the performance on the previously learned task after the new task has been learned. Such incremental learning is useful when a rehearsal with old training data is not possible, for example, when such data are no longer accessible, or when a rehearsal would be inefficient, or when completely new classes are to be learned. Our experiments showed that GSOINN+ can indeed mitigate catastrophic forgetting. For example, in the training sequence MNIST-train→Fashion-train, GSOINN+ achieved an accuracy of 95% on MNIST-test and 80% on Fashion-test. Remarkably, the knowledge from MNIST-train was not erased, although Fashion-train is a training set from an entirely different domain.

GSOINN+ has several limitations, though. In our current model, there is no limit for the maximum number of nodes, which means that the network can grow indefinitely, at least theoretically. Ideally, however, a continual learner should have a bounded system size (Rebuffi et al., 2017), so that resources are used most efficiently. In our future work, we will address this problem. Another scope for our future work is forward-

and backward transfer learning, which is another hallmark of a continual learner.

Note

Supplemental material including Matlab source code is available at the project website at `https://osf.io/gqxya/`.

Acknowledgments

CW was supported by a scholarship from the Japanese Ministry of Education, Culture, Sports, Science and Technology.

References

Aggarwal, C., Hinneburg, A., & Keim, D. (2001). On the surprising behavior of distance metrics in high dimensional spaces. In V. den Bussche J., & V. Vianu (Eds.), *Proceedings of the 8th International Conference on Database Theory* Lecture Notes in Computer Science (pp. 420–434).

Aljundi, R., Rohrbach, M., & Tuytelaars, T. (2019). Selfless sequential learning. In *International Conference on Learning Representations* (pp. 1–17).

Benavoli, A., Corani, G., Demšar, J., & Zaffalon, M. (2017). Time for a change: a tutorial for comparing multiple classifiers through Bayesian analysis. *Journal of Machine Learning Research*, *18*, 1–36.

Bruske, J., & Sommer, G. (1995). Dynamic cell structure learns perfectly topology preserving map. *Neural Computation*, *7*, 845–865.

Chen, T., & Guestrin, C. (2016). XGBoost: Reliable large-scale tree boosting system. In M. Shah, A. Smola, C. Aggarwal, D. Shen, & R. Rastogi (Eds.), *Proceedings of the 22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA* (pp. 785–794).

Cohen, G., Afshar, S., Tapson, J., & van Schaik, A. (2017). EMNIST: an extension of MNIST to handwritten letters. *CoRR*, *abs/1702.05373*.

Flesch, T., Balaguer, J., Dekker, R., Nili, H., & Summerfield, C. (2018). Comparing continual task learning in minds and machines. *Proceedings of the National Academy of Sciences*, *115*, E10313–E10322.

François, D., Wertz, V., & Verleysen, M. (2007). The concentration of fractional distances. *IEEE Transactions on Knowledge and Data Engineering*, *19*, 873–886.

Fritzke, B. (1994a). Growing cell structures–a self-organizing network for unsupervised and supervised learning. *Neural Networks*, *7*, 1441–1460.

Fritzke, B. (1994b). A growing neural gas network learns topologies. In *Proceedings of the 7th International Conference on Neural Information Processing Systems* (pp. 625–632). Cambridge, MA, USA: MIT Press.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., & Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, *114*, 3521–3526.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, *43*, 59–69.

Kruschke, J. (2015). *Doing Bayesian Data Analysis*. (2nd ed.). Elsevier Academic Press, Amsterdam/-Boston/Heidelberg.

Kruschke, J., & Liddell, T. (2018). The Bayesian New Statistics: Hypothesis testing, estimation, meta-analysis, and power analysis from a Bayesian perspective. *Psychonomic Bulletin & Review*, *25*, 178–206.

Li, Z., & Hoiem, D. (2016). Learning without forgetting. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Computer Vision–ECCV 2016* (pp. 614–629). Springer volume 9908 of *Lecture Notes in Computer Science*.

Lomonaco, V., & Maltoni, D. (2017). CORe50: a new dataset and benchmark for continuous object recognition. In S. Levine, V. Vanhoucke, & K. Goldberg (Eds.), *Proceedings of the 1st Annual Conference on Robot Learning* (pp. 17–26). PMLR volume 78 of *Proceedings of Machine Learning Research*.

Marsland, S., Shapiro, J., & Nehmzow, U. (2002). A self-organizing network that grows when required. *Neural Networks*, *15*, 1041–1058.

McCloseky, M., & Cohen, N. (1989). Catastrophic interference in connectionist networks: the sequential learning problem. *The Psychology of Learning and Motivation*, *24*, 104–169.

Nakamura, Y., & Hasegawa, O. (2017). Nonparametric density estimation based on self-organizing incremental neural network for large noisy data. *IEEE Transactions on Neural Networks and Learning Systems*, *28*, 8–17.

Parisi, G. (2020). `https://github.com/giparisi/gwr-tb`, accessed 30 March 2021.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., & Wermter, S. (2019). Continual lifelong learning with neural networks: A review. *Neural Networks*, *113*, 54–71.

Parisi, G. I., Tani, J., Weber, C., & Wermter, S. (2017). Lifelong learning of human actions with deep neural network self-organization. *Neural Networks*, *96*, 137–149.

Parisi, G. I., Tani, J., Weber, C., & Wermter, S. (2018). Lifelong learning of spatiotemporal representations with dual-memory recurrent self-organization. *CoRR*, *abs/1805.10966*.

Parisi, G. I., Weber, C., & Wermter, S. (2015). Self-organizing neural integration of pose-motion features for human action recognition. *Frontiers in Neurorobotics*, *9*, 1–14.

Rebuffi, S., Kolesnikov, A., Sperl, G., & Lampert, C. (2017). iCaRL: Incremental classifier and representation learning. In *Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 2001–2010).

Rios, A., & Itti, L. (2019). Closed-loop memory GAN for continual learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI2019)* (pp. 3332–3338).

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., & Hadsell, R. (2016). Progressive neural networks. *CoRR*, *abs/1606.04671*.

Ruvolo, P., & Eaton, E. (2013). ELLA: An efficient lifelong learning algorithm. In *Proceedings of the 30th International Conference on International Conference on Machine Learning* (pp. 507–515).

Shen, F., & Hasegawa, O. (2006). An incremental network for on-line unsupervised classification and topology learning. *Neural Networks*, *19*, 90–106.

Shen, F., & Hasegawa, O. (2008). A fast nearest neighbor classifier based on self-organizing incremental neural network. *Neural Networks*, *21*, 1537–1547.

Shen, F., Ogura, T., & Hasegawa, O. (2007). An enhanced self-organizing incremental neural network for online unsupervised learning. *Neural Networks*, *20*, 893–903.

Wiwatcharakoses, C., & Berrar, D. (2019). Self-organizing incremental neural networks for continual learning. In S. Kraus (Ed.), *Proc.* 28^{th} *International Joint Conference on Artificial Intelligence (IJCAI 2019)* (pp. 6476–6477).

Wiwatcharakoses, C., & Berrar, D. (2020). SOINN+, a self-organizing incremental neural network for unsupervised learning from noisy data streams. *Expert Systems with Applications*, *143*, 113069.

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, *abs/1708.07747*.