Neural Computing and Applications

Incremental learning with a homeostatic self-organizing neural architecture

--Manuscript Draft--

Manuscript Number:	NCAA-D-18-00539R1
Full Title:	Incremental learning with a homeostatic self-organizing neural architecture
Article Type:	S.I. : WSOM 2017
Keywords:	incremental learning; Pattern recognition; Self-Organizing Maps; Concept drift
Corresponding Author:	Alexander Gepperth, PhD Hochschule Fulda Fulda, GERMANY
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	Hochschule Fulda
Corresponding Author's Secondary Institution:	
First Author:	Alexander Gepperth, PhD
First Author Secondary Information:	
Order of Authors:	Alexander Gepperth, PhD
Order of Authors Secondary Information:	
Funding Information:	
Abstract:	We present a neural architecture for incremental learning applied to high-dimensional visual problems. Key building block is a new self-organizing neural model that we term ReST (Resilient Self-organizing Tissue), which takes over topological prototype organization from self-organizing maps, but is based on an energy function and offers a clear probabilistic interpretation of neural activities, which are constrained to be log-normal. Incremental learning in the presented architecture is made possible by the localized prototype update behavior of the ReST model which avoids catastrophic forgetting when faced with concept drift. We present experiments on three visual classification problems that show hat incremental learning is feasible, describe key mechanisms implemented in the architecture and validate them by experiments on the same problems.
Response to Reviewers:	Reviewer # 1: Thank you for the time you took to review the paper! We tried to address all of your comments, we hope in a satisfactory fashion, in order to have an improved paper... 1. I suggest the authors to discuss how the proposed method is better/different from the state-of-the-art techniques. → OK, added another section to the introduction 2. The authors should add the comparison of the proposed method with state-of-the-art. → very hard to do that because code for most of related work is not available, or would have to be reimplemented with the risk of getting it wrong. What we did: we added a section to the discussion. This includes a comparison to EWC, HAT, LWTA and IMM. We do no conduct additional experiments bu compare incremental learning performance from the experiments section to two of our articles where we perform a systematic comparison of incremental learning on MNIST. 3. The authors should add the description of figures 3 and 4. → OK, done 4. It would be better to add the details about table 1 in the text rather than adding in

	caption. The caption should be short. → OK 5. On what basis, the Rest parameters are selected, it should be explained in detail. → we added a subsection detailing the choice of parameters to the beginning of section 3. 6. It would be better if the method described in section 2 can de explained with the help of diagrammatic representaion or flowchart. → adapted captions to Figs. 3 and 4, especially 4, to serve as a flowchart for PROPRE20 7. The captions for tables 4 and 5 should be above the tables. → OK 8. The caption of table 5 can be reduced by adding the details in text. → OK 9. Again, the captions of figures 5,6,7,8,9 need to be reduced by adding their details in text. → OK for Figs 6,7,8,9 → not OK for Fig.5, the whole caption describes what is shown in the figure, which cannot go to the text. For the other figures, we could move assessments and judgements to the text, which makes the captions shorter. 10. The results should include the difference between original PROPRE and extended PROPRE. → we did the exact opposite: it is actually not necessary to refer to the original PROPRE algorithm, this only confuses a reader (I suppose you also found that confusing). So, while citing the original article, we presented this version in a self-contained form that does not rely on other articles. A comparison of results would have been possible but rather meaningless as PROPRE2.0 is intended to replace PROPRE, and should rather be benchmarked against current DNN-based incrental approaches (which we do). 11. The authors should add more results by comparing their scheme with existing techniques to validate their scheme. → done, see point 2 Reviewer #2: I have the strong feeling that this review was not about my paper but about another. Could you please verify this, and (if true), inform the editors? Thank you!

Noname manuscript No.
(will be inserted by the editor)

Incremental learning with a homeostatic self-organizing neural model

Alexander Gepperth

Received: date / Accepted: date

Abstract We present a new self-organized neural model that we term ReST (Resilient Self-organizing Tissue), which can be run as a convolutional neural network (CNN), possesses a c^{∞} energy function as well as a probabilistic interpretation of neural activities. The latter arises from the constraint of log-normal activity distribution over time that is enforced during ReST learning. The principal message of this article is that self-organized models in general are, due to their localized learning rule that updates only those units close to the best-matching unit (BMU), ideal representation learners for incremental learning architectures. We present such an architecture that uses ReST layers as a building block, benchmark its performance w.r.t. incremental learning in three real-world visual classification problems, and justify the mechanisms implemented in the architecture by dedicated experiments.

Keywords Incremental learning · pattern recognition · neural networks · self-organizing maps

1 Introduction

This article is in the context of research on incremental learning algorithms, and elucidates how self-organizing neural models can be an integral and necessary part of such algorithms.

Incremental learning [1] is a form of learning that is quite different from the more well-established algorithms like neural networks (including deep learning), support vector machines or bagged decision trees, as it does not impose a separation of training and test phases. Incremental models

A.Gepperth
University of Applied Sciences Fulda
Tel.: +49-(0)661-9640-3485
E-mail: alexander.gepperth@cs.hs-fulda.de

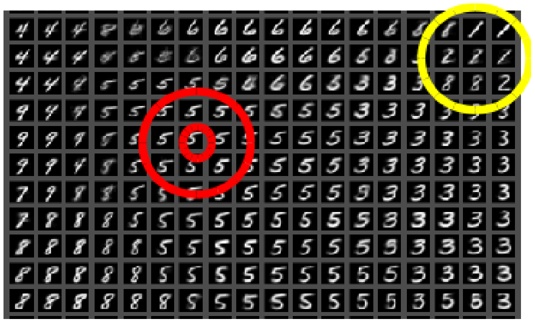

Fig. 1 Illustration of the local update behavior of self-organized models: a sample representing the number 5 will trigger updates only in the vicinity of neurons already "tuned" to this visual class (region marked in red). Under no circumstances will selectivities of neurons far away from this location (e.g., the region marked in yellow) be adapted. In the case of concept drift in the form of an added visual class, only the selectivities of neurons that best repond to this class already are adapted, but no others. This prevents (total) catastrophic forgetting although some accuracy loss is probable as the local updates still "overwrite" previously learned information.

can be updated with new samples at any time, and thus retrained as often as desired which is not (or not fully) possible with the aforementioned models.

The precise definition of incremental learning is not a subject of universal agreement. Although attempts at formal definition ([2, 1]) conflict in minor points, they seem to agree that incremental learning algorithms

- take their training samples one by one, a capacity which shall be termed here *online learning*)
- do not know the total number of samples in advance
- can deal with changes in the statistics underlying sample generation

When one thinks about it, the last two properties are really equivalent: *any* finite dataset that is split into two parts will almost surely exhibit different statistics in each part (see [3] on how this fact affects cross-validation methods in machine

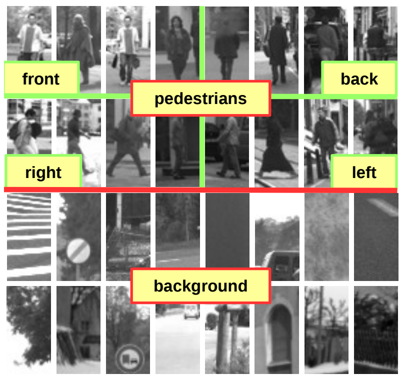

Fig. 2 Representative samples from the real-world pedestrian detection and classification datasets used in this study, in addition to MNIST. There are two classification problems: pose classification into the four classes left/right/front/back (boxes with green borders), and pedestrian detection, i.e. classification into a pedestrian and a background class (boxes with red borders).

learning). Since data in incremental learning scenarios can always be divided into past (already seen) and future (as yet unseen) samples, statistics in those two parts will in general be different. For algorithms not suited for incremental learning, this leads to the so-called *catastrophic forgetting* effect [4,5,6,7,8] which was mainly coined for supervised learning with neural networks but exists for other models as well.Thus, although incremental learning is not a priori specific to supervised or unsupervised learning, the term is mostly used in the context of supervised learning, and we will treat supervised incremental learning in this article.

Self-organized neural models are unsupervised models of neural learning, of which the best-known one is the SOM algorithm [9]. Put simply, the relevance of self-organized models for incremental learning is due to their local update behavior: only the weights in a local neighbourhood around the neuron that best represents the current input are updated. In an incremental learning setting, this means that local concept drift will lead to parameter updates only in a subset of neurons. This simple idea is visualized in Fig. 1 and represents the conceptual foundation of our work on incremental learning so far[10,11], including this article.

Generally, changes in data statistics as encountered in incremental learning are denoted somewhat generally as *concept drift* [12,13], which can be gradual or abrupt. In the latter case one often uses the term *concept shift*. When data statistics do not change globally but only in a specific region of data space, sometimes the term *local concept drift* is used [13]. A prominent example is the addition of a new, visually dissimilar class to a classification problem, which is the kind of concept drift mainly treated in this article. Local concept drift in particular is an important use case for incremental learning algorithms as there is, a priori, no reason why new statistics in localized regions of data space should disrupt learned models elsewhere. Another and much more problematic case is local concept drift/shift with conflict (also treated here), for example when a new but similar class appears in the data: this will in any event have an impact on classification performance until the model can be locally re-adapted to separate the old from the new class. Recognizing concept drift at application time constitutes a challenging task, see [12,13] and references therein. Coarsely speaking, an algorithm needs to decide whether a deviation is just due to noise, or due to a real change in data statistics. This implies a model of the data, in the simplest form a time scale on which "real" concept drift can occur. If concept drift can be reliably detected, it is possible to adapt to it, although this adaptation raises another difficulty: when old and new data statistics are in conflict, how quickly should models be updated? They can be updated quickly but in this case, old information will be forgotten equally quickly. On the other hand, adaptation can be performed slowly, in which case old information is retained longer: it really depends on the application one has in mind to correctly set these parameters. This complexity of online learning might seem intimidating w.r.t. batch learning algorithms. However, the conceptual simplicity of the latter stems from the extreme simplification that is made in discarding all temporal information in a set of data samples.

1.1 Focus and contributions of this article

This article proposes a new self-organized learning model termed ReST (**Re**silient **S**elf-organizing **T**issue), and incorporates it as a building block into an architecture for incremental learning, PROPRE2.0. Since self-organized models (in this case ReST) are not, on their own, capable of supervised incremental learning, PROPRE2.0 implements several additional mechanisms which are described together with an in-depth analysis of their performance and their justification in a typical, applied incremental learning scenario, namely the abrupt addition/presentation of a new, previously unseen visual class (see Figs. 1,2 for a visual impression of the learning problems). The key mechanisms which will be discussed and validated in this article are:

– a supervised read-out mechanism for ReST layers
– a ReST-based concept drift detection mechanism
– protection of the ReST layers against sampling bias by selectively controlling prototype updates
– adaptive suppression of sub-leading ReST activity to protect read-out layer against concept drift
– adaptive control of ReST parameters to maintain topological ordering in the face of concept drift

The ReST model itself presents, while retaining all relevant properties of a self-organizing map, several truly new features that enhance its usefulness for incremental learning problems:

– derivation of the learning rule from an infinitely often continuously differentiable energy function
– clear probabilistic interpretation by enforced log-normal distribution of neural activities
– use of activities with problem-independent log-normal distribution instead of problem-dependent input-prototype distances
– convolutional architecture analogous to a DNN layer

Taking a broader view, the article aims at promoting the fundamental suitability of self-organizing algorithms for incremental learning, which is due to the following favorable properties:

– online learning capacity: traditionally, self-organized models are fed their training samples one by one (in ReST, this is obtained as stochastic gradient descent on an energy function).
– robustness against changing statistics: all SOM-like models update prototypes exclusively in the vicinity of the best-matching unit (BMU). Due to the topological organization of prototypes, concept drift in one part of data space will not affect prototypes for another, distant part of data space, thus avoiding catastrophic forgetting (see Sec. 3.3).
– constant model complexity: except for growing neural gas models, the model complexity of self-organizing is fixed by the number of units. For incremental learning, this is an advantage as it is often conducted under real-world/real-time conditions where guaranteed time and memory complexity are of high importance (e.g., for treating streaming data or in robotics applications). Comparable incremental learning algorithms es listed in Sec. 1.2 often exhibit variable model complexity which can effectively render them unusable depending on the chosen application.

1.2 Related work

There are a number of approaches for incremental learning with support vector machines (see [14] for an overview), but in the light of the given definitions, these approaches are closer to online learning and will run into trouble under concept drift. Furthermore, there are ensemble learning algorithms [15,14] that achieve incremental learning simply by training new classifiers for new batches of data, and combining all existing classifiers for decision making. While this indeed achieves incremental learning under some conditions, it makes the implicit hypothesis that concept drift coincides with new data batches, whereas a *detection* of concept drift is not addressed at all. As the problem of catastrophic forgetting was first remarked for multilayer perceptron (MLP) models [4,5], it is hardly surprising that there is significant work on the subject of how catastrophic forgetting could be avoided[16,17,18,19,20,21,22] although none of these proposals is completely free of problems and of limitations.

1.2.1 Incremental deep learning methods

The field of incremental learning is large, e.g., [23] and [24]. Recent systematic comparisons between different DNN approaches to avoid CF are performed in, e.g., [25] or [26] and [27]. Principal recent approaches to avoid CF include ensemble methods [28,29], dual-memory systems [30,31,32, 10] and regularization approaches [33,34,35,36,20]. The work presented in [20] advocates the popular ”dropout” method as a means to reduce or eliminate catastrophic forgetting, In [37], a new kind of competitive transfer function is presented which is termed LWTA (”local winner takes all”). This article also remarks that most forgetting happens in the readout layers of a DNN, and that maintaining separate readout layers for each sub-task can alleviate catastrophic forgetting. In [35] the authors advocate determining the hidden layer weights that are most ”relevant”, and punishing the change of those weights more heavily during re-training. A related approach is pursued by the Incremental Moment Matching technique (IMM) (see [36]), where weights from DNNs trained on a current and a past sub-tasks are “merged” using the Fisher information matrix. In [38], the concept of Deep Adaptation is proposed, the basic idea being that newly trained filters are constrained to be linear combinations of existing ones, thus guaranteeing unimpaired performance on the original problem. In the context of object detection architectures, [39] proposed to limit catastrophic forgetting by modifying the loss function, including a so-called distillation loss term that minimizes the discrepancy between responses for old classes from the original and the updated network. The iCaRL model proposed in [40] addresses class-incremental learning in an essentially prototype-based architecture, with a focus on managing/updating the class-specific prototypes in an incremental fashion. Other regularization-oriented approaches are proposed in [33,34] and [41] which focus on enforcing sparsity of neural activities by lateral interactions within a layer.

1.2.2 Explicitly incremental learning algorithms

To perform explicit incremental learning as defined earlier in this section, most modern approaches perform an explicit local partitioning of the input space and train a separate classification/regression model for each partition [42,43,44,45, 46]. The manner of performing this partitioning is very diverse, ranging from kd-trees [46] to genetic algorithms [45] and adaptive Gaussian receptive fields [42]. Equally, the choice of local models varies between linear models [42], Gaussian mixture regression [46] or Gaussian Processes [43]. Since this article is concerned with high-dimensional perceptual problems, it can be stated for all cited approaches that it is really the partitioning of the input space that is costly in terms of memory. Most notably, covariance matrices used in [42] are quadratic in the number of input dimensions which makes their use prohibitive for high data dimensionalities.

1.2.3 Neural self-adaptation

Self-adaptation has been observed in biological systems [47] where it is usually termed homeostatic plasticity in order to distinguish it from synaptic plasticity mechanisms. The most influential modeling approach based on these findings is the intrinsic plasticity mechanism described, e.g., in [48] and references therein. Intrinsic plasticity aims at matching the lower-order moments of a neuron's activity to match an exponential distribution which is very similar to what we propose here, although we propose to match a log-normal distribution which is observed abundantly in neural circuits [49] and our self-adaptation is performed in an online learning scenario which was not the case in [48]. In the context of deep neural networks, the concept of batch normalization[50] is a very similar notion: here, neural activities are matched to fit a normal distribution, although only for the current mini-batch. We observe, just as stated in [50], that training convergence is strongly accelerated by this mechanism and that the choice of learning rates becomes much less problem-dependent.

1.2.4 Energy-based self-organizing models

There has been significant research (see [51] and references therein) on finding an energy function for the original SOM model [9], but is was finally shown that such an energy function does not exist. However, a minimal modification of the winner selection mechanism in the original SOM model was shown to allow the construction of a simple and intuitive energy function, see [51]. It is upon this model we build our work on the ReST model, extending this "Heskes model" to include self-adaptation and thereby a conversion to problem-independent activities (whereas the Heskes model works with input-prototype distances whose values will always depend on the problem at hand).

1.3 Overview of the PROPRE2.0 incremental learning architecture

The PROPRE architecture for incremental supervised learning combines generative, self-organized learning of an internal representation with discriminative read-out, see Fig. 3.

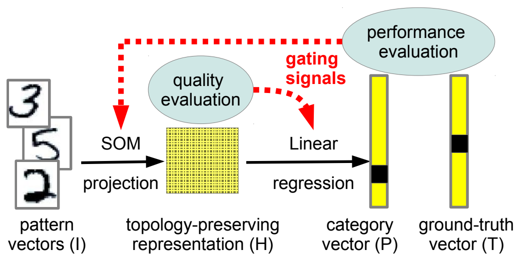

Fig. 3 Block diagram of the PROPRE2.0 architecture for incremental learning. Main building blocks are one or more self-organized ReST layers, a linear regression module based on the last ReST layer, and a module that evaluates prediction performance and ambiguity for gating ReST layer learning.

As a typical self-organized model, the internal ReST representation is topologically organized, and prototype adaptation modifies weights only locally. It is above all this property that allows for incremental learning of prototypes: adaptation of a single prototype changes just its neighbours, which are close in data spacea as ensured by the topological organization of selectivities.

From the classification estimate, a task-related error signal is derived which adapts the internal representation in case of mismatch or classification ambiguity. This ensures that prototype density increases in regions of the input space that are difficult to classify, or in which concept drift is occurring. Prototype adaptation is stably self-terminating when no more errors are made, or when concept drift subsides.

PROPRE2.0 includes several mechanisms that make the internal ReST layer suitable for stable supervised and incremental learning, namely self-adaptation, supervised read-out, protection of the hidden layer against sampling bias, protection of the readout layer against concept drift effects and preservation of topological organization under concept drift. After a demonstration of the incremental learning capabilities of PROPRE2.0, these mechanisms are discussed in-depth and justified by experiments.

A block diagram of the PROPRE2.0 architecture and a simplified version of its reaction to concept drift during incremental learning, is given in Figs. 3,4.

1.4 Differences of PROPRE2.0 to incremental deep learning methods

PROPRE2.0 differs in several ways from recently proposed DNN appraoches to incremental learning. First of all there are differences about assumptions and scaling behavior: PROPRE2.0 does not assume any knowledge about data it is going to be retrained with later, such as it is done in [52]. It also requires no knowledge about which sub-task the model is currently being tested on, as it is assumed in [25]. Furthermore, in contrast to methods based on generative re-

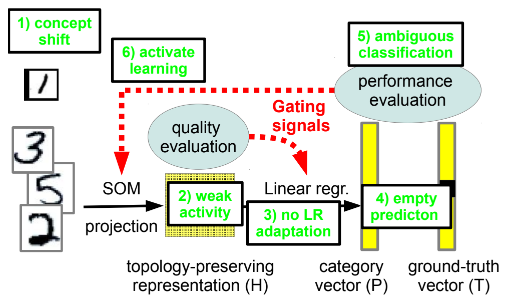

Fig. 4 Sequence of events as a reaction to concept shift in the form of a new visual class, to be regarded together with Fig. 3. A complete PROPRE2.0 treatment of a single sample is always composed of the following steps, which are here linked to the depicted events: **a)** feed-forward pass and linear classification (events 2,4) **b)** decision about adapting LR based on ReST activity (event 3) **c)** measurement of prediction ambiguity (event 5) **d)** decision about adapting ReST layer(s) based on prediction ambiguity (event 6).

play, PROPRE2.0 has constant update complexity as samples from previously trained classes do not need to be retrained again. Then, there are fundamental differences in the energy functions used for DNN training: while most DNN models use a standard cross-entropy loss (at least for classification problems), at best complemented by additive regularization terms [33,34,35,36], PROPRE2.0 uses a separate loss function (here termed energy functions) for each ReST layer and a standard cross-entropy loss for the readout layer. The ReST energy function is such that locality in data space is automatically enforced, even in the absence of supervision (indeed, ReST layers have an essentially unsupervised energy function). ReST layers therefore do not require regularization to "protect" important weights, since new data samples automatically modify only those weights that they are similar to, and leave the others unchanged. Of course, this comes at the price of a potentially lower classification accuracy since the only form of supervision is a mechanism that trains ReST layers only when the current sample was misclassified.

2 Methods

2.1 Datasets

Table 1 Important properties of the three datasets used in this article, see text for details.

Dataset	#train	# test	dim.sions	preproc.	# classes
MNIST	60.000	10.000	28x28=784	raw	10
poses	12.684	2.516	756	HOG	4
peddet	10.000	19.148	756	HOG	2

In this article, we use three different classification benchmarks[1] in order to investigate incremental learning with self-organizing models. First of all, we make use of the MNIST dataset of hand-written digits [53] which is a standard benchmark in machine learning. In addition, we use two datasets obtained from a visual pedestrian classification benchmark, the Daimler Pedestrian Detection Benchmark[54]. The first dataset if about pedestrian detection, i.e., the distinction of pedestrian images from background/non-pedestrian samples, whereas the second task is about pedestrian pose classification, assigning one of the four classes "front/back/left/right" to pedestrian samples (no background samples in this task) according to their visually perceived orientation. Please see Fig. 2 for a visualization of the pedestrian-related tasks, whereas we refer to [53] for an in-depth description of the MNIST benchmark.

The MNIST dataset is a relatively "clean" problem (very little noise and overall variance) where excellent recognition rates can be achieved even on the raw image data. In contrast, the two pedestrian datasets are difficult real-world problems which require a more sophisticated preprocessing in order to be solved with any degree of precision. The applied preprocessing method is termed HOG (histogram of oriented gradients, see [55]) and represents a standard method in real-world visual object recognition. Using the terms of [55], the parameters of the HOG transform applied to cropped images downsampled to a size of 32×64 are: block size 16×16, cell size 8×8, 2×2 cells per block, 9 orientations bins and normalization enabled.

For the MNIST benchmark, each of the 10 classes contributes approximately 10% of the total data samples in training and test data. The other two datasets are unbalanced: for the "Pedestrian Pose" Problem, the distribution of classes is 19% (front), 37% (back), 22% (left) and 22% (right) for the training data and 19%(front), 11% (back), 35% (left) and 35% (right) for the test data. For the "Pedestrian Detection" problem, the distribution is 50% (pedestrians) and 50% (background) for the training data, as well as 56%(pedestrians) and 44%(background) for test data.

Important global facts about all three datasets are given in Tab. 1.

2.2 The ReST model: single-layer, non-convolutional version

ReST is used as a building block for incremental learning within the PROPRE2.0 architecture that will be detailed below. For pedagogical reasons, we introduce ReST first in a non-convolutional version since the introduction of the latter makes the notation much more complex (without changing the essentials). The generalization to a convolutional model

[1] Available under www.gepperth.net/alexander/data

and its integration into a multi-layered architecture will be outlined in Sec. 2.3.

For now, we assume a dataset (or a mini-batch) of input vectors $\mathbf{x}_n \in \mathbb{R}^k$ and a two-dimensional set of $K \times K$ neurons with non-negative activities $a_i \geq 0, i = 1 \ldots, K^2$. It is convenient to express activities computed for an input $\mathbf{x}_n$ as a one-dimensional vector $\mathbf{a}_n \in \mathbb{R}^{K^2}$. A neuron with index i and coordinates $x_i,\ y_i$ has an associated prototype $\mathbf{p}_i \in \mathbb{R}^k, i = 1, \ldots, K^2$, as well as an $K \times K$ neighbourhood matrix $\mathbf{g}_i$ that we write as a one-dimensional vector $\mathbf{g}_i \in \mathbb{R}^{K^2}$ in analogy to the vector of activities. Differing from the SOM model, each neuron furthermore possesses two internal variables o_i and s_i that play a role in enforcing log-normal statistics for the activities $\mathbf{a}_n$ that are computed as follows:

$$d_{ni} = \sqrt{(\mathbf{p}_i - \mathbf{x}_n)^2} \tag{1}$$
$$\tilde{a}_{ni} = o_i - s_i d_{ni} \tag{2}$$
$$a_{ni} = \exp(\tilde{a}_{ni}) \tag{3}$$

The adaptation of the prototypes $\mathbf{p}_i$ is now achieved by minimizing the following energy function:

$$c_{ni} = \langle \mathbf{g}_i, \log \mathbf{a}_n \rangle = \langle \mathbf{g}_i, \tilde{\mathbf{a}}_n \rangle \tag{4}$$
$$\mathcal{E} = \frac{1}{N} \sum_n \langle \mathbf{c}_n, \mathbf{S}(\mathbf{c}_n) \rangle \tag{5}$$

where the logarithm and the vector-valued softmax function $\mathbf{S}(\mathbf{v})$ are applied in a component-wise fashion as

$$e_i = \exp(\beta v_i) \tag{6}$$
$$\mathbf{S}(\mathbf{v})_i = \frac{e_i}{\sum_i e_i} \equiv S_i, \tag{7}$$

β being a parameter that controls the selectivity of the softmax: for higher β values, the output $S(\mathbf{v})$ will tend to be more strongly peaked, the maximal value closer to 1.0 and the rest to 0.0. For lower β values, this relationship is inversed. The minimization of the energy function is performed as a constrained optimization problem, the constraint being that the activities $\mathbf{a}_n$ obey a log-normal distribution over time, with the parameters μ and σ. This implies that $\log \mathbf{a}_n$ (with logarithm applied component-wise!) is normally distributed, with the empirical mean and standard deviation $\hat{\mu}$, $\hat{\sigma}$ coinciding with μ, σ:

$$\hat{\mu} \equiv \frac{1}{N} \sum_n \log a_{ni} = \frac{1}{N} \sum_n \tilde{a}_{ni} \overset{!}{=} \mu \tag{8}$$
$$\hat{\sigma} \equiv \sqrt{\frac{1}{N} \sum_n (\log a_i - \hat{\mu})^2} = \sqrt{\frac{1}{N} \sum_n (\tilde{a}_i - \hat{\mu})^2} \overset{!}{=} \sigma \tag{9}$$

From these requirements, the per-neuron parameters o_i and s_i can be determined unambiguously from the first two moments of the input-prototype distances

$$s_i = \sqrt{\frac{\sigma^2}{\overline{d_i^2} - \overline{d_i}^2}} \tag{10}$$
$$o_i = \mu + s_i \overline{d_i}, \tag{11}$$

which can be computed empirically over a dataset of N samples:

$$\overline{d_i} = \frac{1}{N} \sum_n d_{ni} \tag{12}$$
$$\overline{d_i^2} = \frac{1}{N} \sum_n d_{ni}^2 \tag{13}$$

In a mini-batch setting, we instead take averages over the current mini-batch of N samples (the extreme case being fully online learning where $N = 1$). If we wish to compute the averages $\overline{d_i}$ and $\overline{d_i^2}$ over periods longer than the mini-batch size N, we replace eqns.(12) by a composite method that makes use of a moving average of mini-batches averages:

$$\overline{d_i}(\nu) = (1 - \alpha_d N)\overline{d_i}(\nu - 1) + \alpha_d \sum_n d_{ni} \tag{14}$$
$$\overline{d_i^2}(\nu) = (1 - \alpha_d N)\overline{d_i^2}(\nu - 1) + \alpha_d \sum_n d_{ni}^2 \tag{15}$$

where variable ν expresses the number of the current mini-batch. We scale the adaptation rate $\alpha_d < 1$ with the mini-batch size N since a larger N implies that more samples are used per step in eqn.(14), and thus adaptation can proceed more quickly. Please note that by setting $\alpha_d = \frac{1}{N}$ we can turn off the moving average mechanism, and in this case only the current mini-batch is considered, as it is the case in eqn.(12).

2.2.1 ReST learning rule

For performing gradient descent for the energy function of eqn.(4), we take its derivative w.r.t. to the k-th element of prototype i:

$$\frac{\partial E}{\partial p_{ik}} = \frac{\partial}{\partial p_{ik}} \frac{1}{N} \sum_{nj} c_{nj} S(\mathbf{c})_{nj} = \tag{16}$$
$$= \frac{1}{N} \sum_{nj} \left(S(\mathbf{c}_n)_j \frac{\partial c_{nj}}{\partial p_{ik}} + \frac{\partial S(\mathbf{c})_{nj}}{\partial p_{ik}} c_{nj} \right) = \tag{17}$$
$$= \frac{1}{N} \sum_{nj} \left(S(\mathbf{c}_n)_j \frac{\partial c_{nj}}{\partial p_{ik}} + \beta S(\mathbf{c}_n)_i \left(\delta_{ij} - S(\mathbf{c}_n)_j\right) \frac{\partial c_{nj}}{\partial p_{ik}} \right) \tag{18}$$

where we have used the expression $\partial_j S_i = \beta S_i(\delta_{ij} - S_j)$ for the derivative of the softmax function. If we assume that the softmax function is parameterized such that it puts 1.0 at the position of the maximal value (whose index is expressed by $*$), and 0 everywhere else, we obtain:

$$\frac{\partial E}{\partial p_{ik}} \approx \frac{1}{N}\sum_n \frac{\partial c_{n*}}{\partial p_{ik}} \tag{19}$$

and arrive at the update rule

$$\mathbf{p}_i \leftarrow \mathbf{p}_i - \frac{\epsilon s_i}{2N}\sum_n g_{*i}\frac{\mathbf{p}_i - \mathbf{x}_n}{||\mathbf{p}_i - \mathbf{x}_n||} \tag{20}$$

where we have one more time designed the index of the best-matching unit (BMU) by a star:

$$* = \text{arg max}_i c_i \tag{21}$$

If we had omitted the square root in the definition of input-prototype distances in eqn. (1), we would have arrived at the equivalent rule

$$\mathbf{p}_i \leftarrow \mathbf{p}_i - \frac{\epsilon s_i}{N}\sum_n g_{*i}(\mathbf{p}_i - \mathbf{x}_n) \tag{22}$$

which differs (for the online case of $N = 1$) from the energy-based SOM model proposed in [51] only by a factor of s_i for each neuron. Regarding the original SOM model [9], the additional difference is that the best-matching unit is not determined from input-prototype distances but from the convolution $\mathbf{c}$ of activities with the neighbourhood matrix as given in eqn. (4). We can therefore see that the learning rules (20,22) scale each neuron's prototype adaptation by a factor that is, by eqn. (10), inversely proportional to the variance of activities of that neuron. Thus, neurons whose prototypes are either too unspecific or too generic (resulting in uniformly low or high activations with low variance) receive a competitive advantage. We also note that this mechanism is self-limiting: increased prototype adaptation usually increases the variance of a neuron's activities, thus eventually annulling the competitive advantage and leading to stable competitive learning dynamics.

2.2.2 *Implementation of constrained optimization*

Minimizing the energy function (4) is performed by performing repeated gradient descent steps using learning rule (20) on the whole available training data set or mini-batch, each step followed by an explicit enforcement of the constraints by applying eqn. (10), this again being followed by an update of the averages using eqn.(12).

For speeding up convergence, the neighbourhood matrix $\mathbf{g}_i$ of neuron i is modelled as a Gaussian whose standard deviation $S(\nu)$ is decayed exponentially over time, as it is usual with SOMs:

$$g_{ij} = \exp\left(-\frac{(x_j - x_i)^2 + (y_j - y_i)^2}{2S(\nu)^2}\right) \tag{23}$$

```
Algorithm: Constrained ReST optimization
Parameters :
  – nr of iterations $T$
  – mini-batch size $N$
  – initial and final neigh. radius $S_0$, $S_\infty$
  – learning rate $\alpha$
  – self-adaptation rate $\alpha_d$
  – time parameters $t_A$, $t_0$ and $t_\infty$
  – target values $\sigma$, $\mu$ for self-adaptation
Result: trained prototypes $\mathbf{p}_i$
begin
  Initialize all prototypes $\mathbf{p}_i$ to small random values ;
  Initialize moving averages $\overline{d_i}(0) = 0$ and $\overline{d_i^2} = 0$ ;
  Initialize per-neuron parameters $s_i = 0.5, o_i = 0$ ;
  Compute decay time constant $\lambda = -\frac{\log(-S_\infty/S_0)}{t_\infty - t_0}$ ;
  for mini-batch $\nu < T$ do
    compute nb.radius $S(\nu)$ and learning rate $\alpha(\nu)$:
     begin
       if $\nu < t_A$ then $\alpha(\nu) = 0, S(\tau) = S_0$;
       else if $\nu < t_0$ then $\alpha(\nu) = \alpha, S(\nu) = S_0$;
       else if $\nu < t_\infty$ then
         $\alpha(\nu) = \alpha, S(\nu) = S_0 e^{-\lambda\nu}$;
       else $\alpha(\nu) = \alpha, S(\nu) = S_\infty$ ;
     end
    recompute nb. matr. $\mathbf{g}_i$ based on $S(\nu)$ ;
    select a random mini-batch $\mathbf{x}_n$, $0 < n < N$ ;
    update prototypes $\mathbf{p}_i$ according to eqn. (20) ;
    enforce constraint using eqn. (10) ;
    adapt averages $\overline{d_i}(\nu)$ and $\overline{d_i^2}(\nu)$ using eqn. (14) ;
  end
  return $\mathbf{p}_i$
end
```

Algorithm 1: Mini-batch based learning with the ReST model.

In contrast to normal SOM learning we do not decay the ReST learning rate α over time, since this complicates advanced gradient descent strategies and introduces unnecessary parameters. Additionally, we impose an initial period without prototype adaptation where only the neural statistics are adapted. This ensures that any deviation from the desired statistics introduced by prototype adaptation are small, leading only to small corrections to the o_i and s_i, which avoid potentially problematic feedback loops between the two adaptation processes. The detailed training procedure, as well as the relevant parameters, are detailed in Alg. 1

2.2.3 *Choice of ReST parameters*

The self-adaptation process is governed by the parameters μ and σ of the log-normal distribution that the activities a_i are required to obey, which raises the question of what their intrinsic significance could be, especially within the context of self-organizing maps and incremental learning. First of all, from the properties of log-normal distributions we know that the quantity e^μ represents both the geometric mean and at the same time the median of a log-normally distributed variable, so essentially we could just fix a median value M

and compute $\mu = \log M$ from it. The median for this distribution is smaller but usually close to the arithmetic mean as well so we can also see M as a rough indicator for the arithmetic time average of a neuron's activity. The quantity e^σ is sometimes termed the geometric standard deviation and can be expressed as

$$e^\sigma = \exp\left(\sqrt{\frac{1}{N}\sum_n \left(\log\frac{a_{ni}}{e^{\hat{\mu}}}\right)^2}\right) = \tag{24}$$

$$= \sqrt[N]{\mathit{\Pi}_n \exp\left(\left(\log\frac{a_{ni}}{e^{\hat{\mu}}}\right)^2\right)} = E^g_n\sqrt{\exp\left(\left(\log\frac{a_{ni}}{e^{\hat{\mu}}}\right)^2\right)} \tag{25}$$

and is thus related to the geometric mean of the expression $\sqrt{\exp\left(\left(\log\frac{a_{ni}}{e^{\hat{\mu}}}\right)^2\right)}$. This expresses the multiplicative spread of values around their empirical geometric mean $e^{\hat{\mu}}$, regardless of the direction. We can thus expect that higher values of e^σ will push the activities further away from their geometric mean, forcing them to be more specific, either close to 0 or far away from it. We can thus think of σ as a parameter controlling the sparsity of neural responses, which previous studies on transfer functions for self-organized maps [56] found to be an important factor for performing classification based on SOM activities.

In order to guarantee identical functioning of the WTM mechanism for variable map sizes, the softmax function needs to be parameterized correctly, and more specifically as a function of the number of neurons in the SOM. We therefore need to set the parameter β such that qualitatively identical behavior ensues for any map size. We measure identical behavior by demanding that the the maximal response of the softmax function be ξ when given a vector $\mathbf{x} \in \mathbb{R}^n$ that consists of $n-1$ times value B and 1 time value λB. Solving this for β gives us the expression

$$\beta = \frac{\ln(\xi^{-1}-1) - \ln(n-1)}{B(1-\lambda)} \tag{26}$$

The softmax function is a very useful tool for obtaining a "hard" yet differentiable winner selection, in addition to allowing a steady transition between "hard" and "soft" winner selection. In some cases, problems can occur: first of all, sensible choices for B and λ may be hard to obtain because they depend on the learning dynamics. Furthermore, when $\beta > 700$, numerical issues arise due to the exponentials involved. Fortunately there is a simple rule-of-thumb solution for both problems that consists of applying a softmax function with "best guess" parameters *several times* in eqn. (4). This complicates the gradient, but as long as the final softmax function gives a sufficiently hard winner assignment, the learning rule (20) remains valid. Software frameworks like TensorFlow can compute the gradient symbolically, so even the exact gradient can be used regardless of how often softmax was applied. We found that a three-fold application was always sufficient to guarantee a unique winner selection.

The parameter S_0 is usually made to depend on the map size. A rule of thumb that always worked well is to choose it proportional to the diagonal of the quadratic $K \times K$ map, i.e., $S_0 = \frac{K}{4}$. In contrast, classification experiments always give best results the smaller S_∞ is, so this is always fixed at small values like $S_\infty = 0.01$. The values of t_0, t_A and t_∞ can be determined empirically be requiring that i) self-adaptation has occurred before t_A ii) the energy function has converged to a stable value before t_0 and iii) that the energy function is as low as possible while still satisfying all constraints at t_∞. Here, we see the value of an energy function as it can be used to determine convergence, so these parameters which for SOMs have to be obtained by visual inspection, can be determined by cross-validation. By a similar reasoning, a good value for the learning rate can be obtained, where smaller values are always acceptable but lead to increased training time. The mini-batch size is generally assumed to be $N = 100$ in this article. The self-adaptation rate, $\alpha_d N$, be chosen such that the constraints are approximately upheld during prototype adaptation, meaning it will depend on the choice of α and is thus not a free parameter but can be indirectly obtained by cross-validation.

2.3 ReST: Multilayer convolutional formulation

As the ReST model is to be used for incremental learning in a potentially multi-layered and convolutional neural architecture, we keep the basic concepts from Sec. 2.2 but adapt the notation and explain the generalization to convolutional ReST layers. This is pretty simple: instead of having a single input vector treated by a single set of prototypes, we now subdivide the input into rectangular and regularly spaced patches ("receptive fields", RF), each of which is nevertheless still analyzed by a single set of prototypes and regulatory parameters o, s. For learning and statistics adaptation, activities in each of the "little ReST models" are averaged, resulting in a unique update law for prototypes and regulatory parameters. In this section, we describe just the parts of the model that are changed when passing to a convolutional formulation, meaning that all other mechanisms remain as outlined in the previous sections.

We adopt here the usual NHWC convention from deep learning, modelling activities in each layer X as a four-dimensional tensor $a^{(X)} \in \mathbb{R}_X$ using the short-hand notation $\mathbb{R}^{(X)} \equiv \mathbb{R}^{N\times W^X \times H^X \times C^X}$. This tensor is indexed by mini-batch index $0 \le n < N$, RF y and x indices $0 \le h < H$ and $0 \le w < W$, as well as the channel index $0 \le i < C$, which corresponds to the linear index into the self-organizing ReST map in analogy to the activity vector $\mathbf{a}$ in Sec. 2.2. For a ReST map of size $K \times K$, it therefore follows that $C =$

K^2. In the manner of a convolutional network, inputs to layer X, which are nothing but the activities of the preceding layer $X-1$ denoted as $a^{(X-1)} \in \mathbb{R}^{(X-1)}$, are re-arranged into a new input vector $\mathbf{x}^{(X)}_{nwh} \in \mathbb{R}^{N \times H^{(X)} \times W^{(X)} \times \tilde{C}^{(X)}}$ according to RF sizes $F^{(X)} = (f^{(X)}_x f^{(X)}_y)$ and step sizes $(\Delta^{(X)}_x, \Delta^{(X)}_y)$, where $H^{(X)} \equiv 1 + \frac{h^I - f^I}{f^I - \delta^I}$, $W^{(X)}$ analogously and $\tilde{C}^{(X)} \equiv f^{(X)}_x f^{(X)}_y$. The prototype vector associated with each "neuron", i.e., each element indexed by the channel index i, is therefore written $\mathbf{p}^{(X)}_i \in \mathbb{R}^{F^{(X)}}$. Analogously, the neighbourhood matrix of neuron i is now denoted $\mathbf{g}^{(X)}_i$, and the per-neuron parameters for regulating output statistics are written $o^{(X)}_i$ and $s^{(X)}_i$. As the superscript indicating the layer makes equations very cumbersome, we will drop it wherever this is possible without creating ambiguities.

In analogy to eqn. (1), we formulate the computation of activities $a^{(X)} \in \mathbb{R}_X$ in layer X as follows, dropping the layer superscript everywhere:

$$d_{nwhi} = \sqrt{(\mathbf{p}_i - \mathbf{x}_{nwh})^2} \tag{27}$$

$$\tilde{a}_{nwhi} = o_i - s_i d_{nwhi} \tag{28}$$

$$a_{nwhi} = \exp\left(\tilde{a}_{nwhi}\right) \tag{29}$$

The adaptation of the prototypes $\mathbf{p}^{(X)}_i$ is now achieved by minimizing the following generalized energy function which is analogous to eqn. (4) except that we additionally average over the W and H dimensions, i.e., over each "little ResT model".

$$c_{nwhi} = \langle \mathbf{g}_i, \log \mathbf{a}_{nwh} \rangle \tag{30}$$

$$\mathcal{E}^{(X)} = \frac{1}{NWH} \sum_{nwh} \langle \mathbf{c}_{nwh}, \mathbf{S}(\mathbf{c}_{nwh}) \rangle \tag{31}$$

The updated values of the regulatory parameters $o^{(X)}_i$, $s^{(X)}_i$ are now obtained as

$$s_i = \sqrt{\frac{\sigma^2}{\overline{d^2_i} - \overline{d_i}^2}} \tag{32}$$

$$o_i = \mu + s_i \overline{d_i} \tag{33}$$

where the first two moments of the input-prototype distances are obtained as

$$\overline{d_i} = \frac{1}{NWH} \sum_{nwh} d_{nwhi} \tag{34}$$

$$\overline{d^2_i} = \frac{1}{NWH} \sum_{nwh} d^2_{nwhi} \tag{35}$$

As the convolutional ReST model is explicitly based on mini-batches, the computation of the long-term averages of the quantities d_{nwhi} and d^2_{nwhi} must use, from the very beginning, a form analogous to eqn. (14), where again superscripts are dropped and averages are taken over the NWH dimensions:

$$\overline{d}_i(\tau) = (1 - \alpha_d N)\overline{d}^{(\tau-1)}_i + \frac{\alpha_d}{WH} \sum_{nwh} d_{nwhi} \tag{36}$$

$$\overline{d^2_i}(\tau) = (1 - \alpha_d N)\overline{d^2_i}(\tau - 1) + \frac{\alpha_d}{WH} \sum_{nwh} d^2_{nwhi} \tag{37}$$

2.4 Convolutional ReST model with independent filters

An interesting alternative to a fully convolutional model where only a single set of prototypes and regulatory parameters exists that is replicated over the WH dimensions, we propose here a similar possibility but with independent prototypes/parameters for every (w, h) value. This only changes the initial computation of activities to

$$d_{nwhi} = \sqrt{(\mathbf{p}_{whi} - \mathbf{x}_{nwh})^2} \tag{38}$$

$$\tilde{a}_{nwhi} = o_{whi} - s_{whi} d_{nwhi} \tag{39}$$

$$a_{nwhi} = \exp\left(\tilde{a}_{nwhi}\right). \tag{40}$$

The energy function (30) remains valid, and the only real difference to the convolutional model is that the regulatory parameters o_{whi} and s_{whi} are now adapted independently from each other at different spatial locations, which requires a separate computation of moving averages in the spirit of eqn.(36) as well. Obviously, the existence of a greater number of free parameters might allow a better representation of input statistics and thereby a higher classification accuracy. This will be tested in Sec. 3.1.

2.5 PROPRE2.0

We introduce a model for incremental learning as proposed in [10], which used ReST layers (conceptually similar to self-organizing maps) as the building blocks for incremental learning. The presented model is similar in spirit to [10], yet is aimed at performing incremental learning in a deep network structure that is analogous to a CNN. By using ReST layers, the number of independent free parameters is reduced strongly w.r.t. [10]. The following model overview is self-contained and uses a terminology more adapted to describe convolutional deep networks, as this is the direction in which we wish to develop this model. A TensorFlow/Python [57] implementation is made publicly available.

A PROPRE 2.0 neural network can technically comprise many layers, although in this investigation we consider a shallow architecture of three layers: input layer (I), hidden (H) modeled as a ReST layer, and an output layer (O). PROPRE2.0 is a supervised model which requires a vector-valued target $\mathbf{t}$ for every input $\mathbf{x}$, where $\mathbf{t}$ usually encodes class

Table 2 Significance of various value ranges of the confidence measure m_n.

range	meaning
$\in [-1, -0.5]$	certain, incorrect
$\in [-0.5, -0.1]$	uncertain, incorrect
$\in [0.1, 0.5]$	uncertain, correct
$\in [0.5, 1]$	certain, correct

membership in a one-hot fashion. The output layer O performs simple linear regression, transforming hidden layer activities $a^{(H)}$ to output layer activities $a^{(O)}$ to match the target values $\mathbf{t}$. A particular point are the *modulatory* influences within the architecture, which control and restrict learning in hidden and output layers. The reaction of PROPRE2.0 to the kind of concept drift/shift considered in this article is shown in Fig. 4.

2.5.1 Forward pass and decision making

Following the presentation of a new mini-batch $\{\mathbf{x}_n\}$, activities $a^{(H)}$ in the hidden ReST layer H are generated as detailed in Sec. 2.2. Linear regression is applied to a flattened (over the single-sample indices h, w, i) and thresholded version of $a^{(H)}_{nwhi}$ to generate output layer activities $a^{(O)}$:

$$t^{(H)}_{nwhi} = \begin{cases} a^{(H)}_{nwhi} & \text{if} \quad a^{(H)}_{nwhi} \geq \theta \\ 0 & \text{else} \end{cases} \tag{41}$$

$$\mathbf{f}^{(H)}_n = \text{flatten}(t^{(H)}_{nwhi}) \tag{42}$$

$$\mathbf{a}^{(O)}_n = W^{(O)}\mathbf{f}^{(H)}_n + \mathbf{b}^{(O)} \tag{43}$$

The classification decision χ_n is then computed as the simple argmax of output layer activities, together with a confidence measure m_n:

$$\chi_n = \text{arg max}_i a^{(O)}_{ni} \tag{44}$$

$$u_n = \text{max}_i a^{(O)}_i - \text{max2}_i a^{(O)}_i \tag{45}$$

$$m_n = \begin{cases} u_n & \text{if arg max}_j t_{nj} = \text{arg max}_j a^{(O)}_{nj} \\ -u_n & \text{otherwise} \end{cases} \tag{46}$$

where max2 indicates the second-highest value. Please see Tab. 2 for a clarification of the significance of the confidence measure m_n.

2.5.2 Learning and modulation

Training prototypes in layer H is performed by minimizing the ReST energy function by gradient descent as detailed in the previous section. Training the linear regression weights in the output layer O is done by minimizing the cross-entropy loss function based on a softmax-processed version of the activities $\mathbf{a}^{(O)}$:

$$\mathcal{E}^{(O)} = \frac{1}{N}\sum_{ni} t_{ni} \log S(\mathbf{a}^{(O)}_n)_i + + (1 - t_{ni}) \log \left(1 - S(\mathbf{a}^{(O)}_n)_i\right) \tag{47}$$

This requires the setting of a learning rate α_{LR} in the update rule, which is a free parameter of the PROPRE2.0 architecture.

However, hidden layer and output layer neurons must not adapt their weights all the time, only when it can be done safely. Here, there are two distinct cases to be distinguished: first, when the hidden layer has low overall activity, maybe because the input belongs to a newly added class, then linear regression should not adapt its weights because hidden layer activity is probably not meaningful and it will impair performance for already represented classes. This is achieved automatically by thresholding hidden layer activities in eqn. (41), leading to zero activity if activity is too low and thereby effectively switching off linear regression learning. Secondly, hidden layer prototypes will only be updated when the current estimate of class membership, i.e. the output layer activities $\mathbf{a}^{(O)}_n$, is either uncertain or wrong. To measure uncertainty, we first define an uncertainty measure based on the output layer activities, whose basic idea is that a certain estimate of class membership has a clear activity maximum, so a good measure is just to use the bounded difference between first and second maximum, finally leading to the confidence measure m_n defined in eqn. (44). In contrast to the updating of LR weights, which is regulated automatically by the thresholding operation in eqn. (41), prototype adaptation in the ReST layer needs to be controlled explicitly. For this purpose, we define a *modulation measure* λ_n depending on confidence measure m_n, which is defined as:

$$\lambda_n = \begin{cases} 0 & \text{if } m_n > \theta_m \\ 1 & \text{otherwise} \end{cases} \tag{48}$$

Here, the threshold $-1 < \theta_m < 1$ is another free parameter which governs the willingness of the architecture to update itself as a reaction to the classification confidence m_n (see Tab. 2 for an interpretation of this measure). By thresholding m_n we thus allow hidden layer weights to be trained only when the current class estimate is of less-than-perfect quality, either outright incorrect (for $\theta_m \leq 0$) or correct but of significant uncertainty (for $\theta_m > 0$).

2.6 Evaluation measures

In the experiments presented in Sec. 3, we use several different measures to evaluate and visualize results which shall briefly be explained here. They are standard measures for

Table 3 Parameters used in PROPRE2.0 experiments. Please refer to Sec. 2 for an explanation of the symbols.

K	10	θ	0.5	t_A	3000
α_d	0.00001	α	0.005	S_0	$0.25K$
t_0	9000	$t_{\inf}$	19000	t_{incr}	24000
S_∞	0.01	α_{LR}	0.001	θ_m	0.0
e^σ, e^μ	0.1, 3	Δt_{incr}	6000	N	100
S_{incr}	0.2				

SOM quality and supervised learning that are frequently used in the literature. For comparing the topological ordering in a given ReST layer, we resort to the so-called topographic error e_{top} [58]

$$e_n \equiv \begin{cases} 1 & \text{if } \mathbf{y}_n^* \text{ is adjacent to } \mathbf{y}_n^{*2} \\ 0 & \text{else} \end{cases} \tag{49}$$

$$e_{\text{top}} \equiv 1 - \left\langle e_n \right\rangle_n, \tag{50}$$

where $\mathbf{y}_n^{*2}$ is the position of the unit with the second-highest activation after the BMU which resides at $\mathbf{y}^*$.

Lastly, we consider the classification accuracy E in order to assess the success of supervised learning in assigning the proper classes to input vectors. Given the ground-truth vector $\mathbf{t}_n$ and the a network output $\mathbf{a}p(O)_n$ in response to a sample with dataset or mini-batch index n, the classification accuracy is defined as

$$E_n \equiv \begin{cases} 1 & \text{if } \arg\max_i a_{ni}^{(O)} = \arg\max_i t_{ni} \\ 0 & \text{else} \end{cases} \tag{51}$$

$$E = \left\langle E_n \right\rangle_n \tag{52}$$

3 Experiments

All PROPRE2.0 experiments use, by default, a single non-convolutional hidden ReST layer of size $K = 10$. Each experiment begins with a convergence phase where only ReST self-adaptation is performed in order to have a well-defined starting point for learning. Subsequently, full ReST learning plus self-adaptation are conducted with neighbourhood radius S_0 until t_0, whereupon it is exponentially decreased to reach its asymptotic value S_∞ at t_∞. Self-adaptation and training with asymptotic parameters is conducted until t_{incr}, whereupon normal PROPRE2.0 training is conducted with modulatory influences turned on until $T_{\text{incr}} + \Delta T_{\text{incr}}$. The incremental learning step, presenting exclusively examples of a previously unseen class, requires an additional procedure in order to maintain a topological prototype organization, which is ensured by increasing the neighbourhood radius to S_{incr} at $t = t_{\text{incr}}$, and exponentially reducing it to its asymptotic value S_∞ at $t = t_{\text{incr}} + \Delta t_{\text{incr}}$.

The adaptable parameters of the PROPRE algorithm are given in Tab. 3. Parameters like N, K, T_{inf}, T_{A} and t_{inf} are chosen such that a single learning run can be performed in under 10 minutes on off-the-shelf hardware. The ReST learning rate α is chosen empirically, whereas the self-adaptation rate α_d is chosen such that self-adaptation is performed at a slower timescale than protoype adaptation. The target values for geometric mean and standard deviation are in principle arbitrary except that a e^σ must be high enough to allow discrimination by the linear regression readout (the learning rate of which is chosen as one would choose it for a DNN, in relation to the batch size N). The PROPRE2.0 parameters θ are chosen such that sufficient learning can happen during the incremental learning period. All experiments will use these values unless explicitly stated otherwise.

Table 4 Classification accuracy E as a function of hidden layer size K. As may be expected, increasing K increases performance.

Task	E_{10x10}	E_{30x30}	E_{50x50}
MNIST	91.3%	96.7%	98.2%
peddet	95.1%	97.0 %	99.2%
poses	85.2%	89.8%	92.1%

3.1 Non-incremental and incremental learning performance

This experiment has the goal of establishing that PROPRE2.0 can achieve acceptable non-incremental performance on the datasets in this article, as well as showing that incremental learning is possible and feasible. It is therefore conducted in two variants: the non-incremental variant trains a PROPRE2.0 architecture with various choices for the hidden ReST layer ("flat" in the sense of Sec. 2.2, convolutional as detailed in Sec. 2.3 or independent as in Sec. 2.4) until t_{incr}, that is, without incremental learning. No comparison to state-of-the-art deep learning results on MNIST or other datasets is intended, although the classification rates obtained here (see Tab. 4) are not extremely far from state-of-the-art results using DNNs. We find that S_∞ has a strong impact on classification accuracy and should be chosen very small. We also find that keeping $\theta = 0$ during training increases classification accuracy moderately. The learning rates α_{LR} and α have an impact on final classification accuracy, but thanks to self-adaptation we find that they do not need to be varied across tasks.

The incremental variant, one the other hand, first splits each dataset into class 0 (0 digits for MNIST, "left" for pose classification and "background" for pedestrian detection) versus the remaining classes. These two datasets are termed D_1 and D_2, respectively, and are used to measure incremental learning capacity in a simple way. Testing various convolutional configurations of the ReST layer (at least on MNIST since its samples are images), initial training and ReST convergence are performed on D_1 until T_{conv}, retaining the last test set performance on D_1. Subsequently, re-

Table 5 Incremental learning accuracies (see text) on all three datasets used in this study (but mainly on MNIST). For conv(olutional) and ind(ependent) filter networks, the numbers in the first table column indicate the filter sizes in X and Y direction, as well as the filter steps in the X and Y direction. All results are classification accuracies before and after training on D_2:

model / dataset	poses	peddet	MNIST
flat28/28/x/x	x	x	91/83
conv21/21/7/7	x	x	85/78
conv14/14/14/14 x	x		81/75
conv14/14/7/7	x	x	93/87
conv7/7/7/7	x	x	65/61
ind21/21/7/7	x	x	87/82
ind14/14/14/14 x	x		83/73
ind14/14/7/7	x	x	96/92
ind7/7/7/7	x	x	88/85
flat21/36/x/x	85/77	95/85	x

training on D_2 is conducted until an accuracy of 90% is achieved on the D_2 test set, whereupon classification accuracy on the D_1 test set is measured, which is taken as a measure of incremental learning capacity on that particular task. Clearly, the higher this accuracy is, the less was forgotten about D_1. Results for all three datasets and various convolutional ReST configurations are displayed in Tab. 5. The show that some forgetting is occurring, which is logical as prototypes get reallocated to the new class, but that forgetting is gradual and, above all, limited. In a previous study [59] we performed very similar experiments using DNNs and found catastrophic forgetting behavior (D_1 accuracy at chance level) when applied to MNIST, so the results presented here, while not perfect, represent a strong improvement over the capabilities of current DNN models. For MNIST, we tested various possibilities for convolutional ReST layers, both with fully-connected models as well as independent and convolutional prototypes/parameters.

In the rest of this section, we will conduct experiments that shed lights on the key mechanisms in PROPRE2.0 and outline their necessity and their merit for incremental learning.

3.2 Self-adaptation of neural activities

In this section we will demonstrate the beneficial effects of neural self-adaptation guaranteed by the ReST model. To this end, we will conduct simulations with all three datasets described in Sec. 2.1 and a single "flat" (non-convolutional) ReST layer. After ReST convergence at t_∞, statistics is collected for 5000 iterations until t_{incr} and subsequently evaluated. Histograms of all neural activities during these 5000 iterations are computed and compared to a theoretical log-normal distribution. As can be seen in Fig. 5, the fit between theoretical and measured distribution is generally acceptable for all datasets, although of course a perfect fit is not to be expected. This is because we only fit the first two moments of the log activities to defined values. For a better fit, at least the third moment of the log activities should be controlled, which would however result in a more complex constrained optimization scheme. For the activity histograms of all neurons in the ReST layer for all three datasets, please see Appendix A.

3.3 Basic feasibility of ReST for incremental learning

In this experiment, we wish to demonstrate the basic feature of the ReST model that makes it an excellent choice for incremental learning: its purely local update behavior. Using the parameter settings from Sec. 3 and a hidden layer size of $K = 10$ and an asymptotic neighbourhood radius of $S_\infty = 1.0$, we train the PROPRE architecture on the MNIST dataset where the class "0" has been removed from the training data. MNIST is well suited for this experiment as its samples can be visually interpreted, making drift effects in the prototypes clearly discernible.

At $t = t_{\text{incr}}$, we introduce strong concept drift by presenting only the hitherto excluded class 0 and the evolution of ReST prototypes is observed at various times $t > t_{\text{incr}}$, here 500, 1500 and 5000. In order to obtain a purely unsupervised behavior without any control of updating, we set $\theta_m = 1.1$ which results in a guaranteed ReST update for every sample, see eqn. (48). The results can be observed in Fig. 6. We observe that, as t increases, the insertion of the new class affects more and more of the ReST prototypes but the change is gradual and strictly local.

3.4 Protecting the ReST layer against sampling bias

As the experiments of the last section (see Fig. 6) show, ReST prototype adaptation in the hidden layer H is strongly impacted by the time a new class is presented for $t > t_{\text{incr}}$. The longer it is presented, the more prototypes lean towards the new class, and consequently prototypes describing old classes are forgotten. This is a variation of the sampling bias problem often encountered in machine learning: the frequency of classes during training is not correlated at all to their frequency during application. In our case, this is a highly undesirable effect as it is not at all the time of presenting a new class that should control forgetting, but classification performance! PROPRE2.0 therefore adapts prototypes only as long as classification performance for the new class, as measured by the confidence measure of eqn. (44), is unsatisfactory. This effectively protects the ReST layer and makes prototype adaptation totally independent of the time of presentation. To demonstrate this, we repeat the experiments of the previous section but this time using a parameter of $\theta_m = 0.7$, meaning that once the confidence measure for

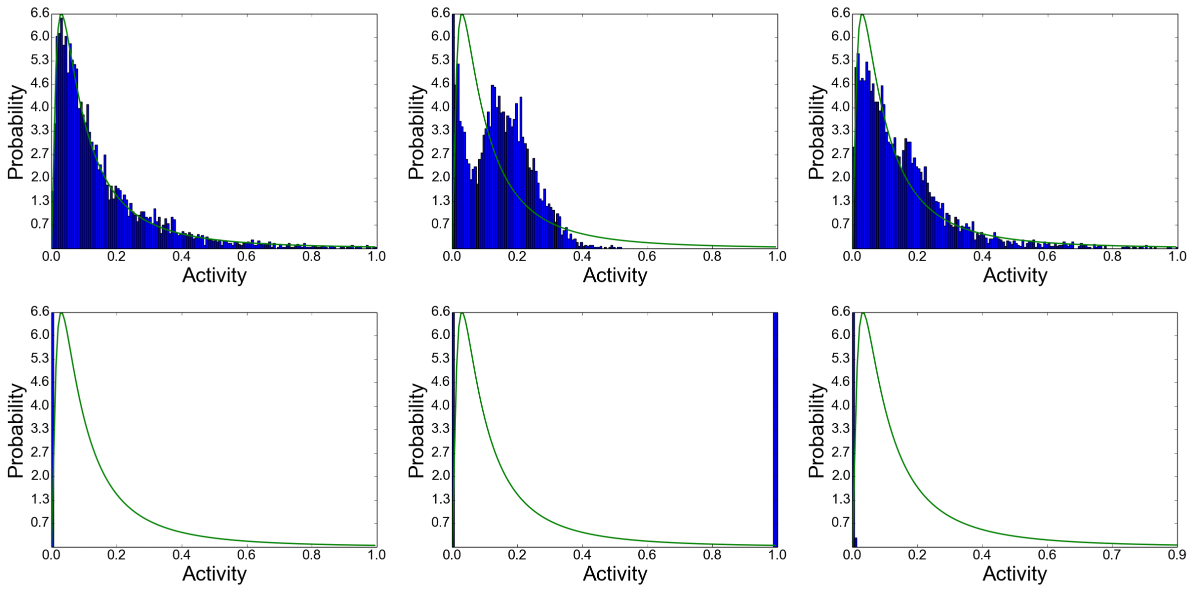

Fig. 5 Activity histograms for neuron $(4, 4)$ in a ReST layer trained on the pose classification (left), pedestrian detection (middle) and MNIST (right) problem, both for the case of enabled (upper row) and disabled (lower row) self-adaptation. The theoretical log-normal density is superimposed onto the histogram as a solid green line, showing a generally good match for all three tasks.

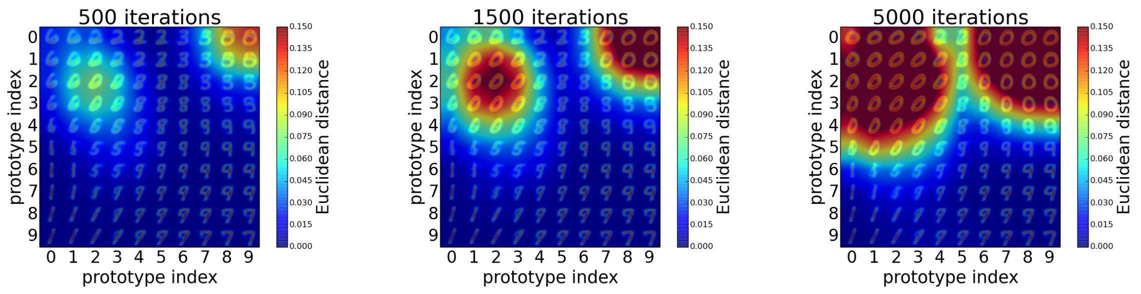

Fig. 6 Demonstration of the purely local update behavior of the ReST model when learning incrementally, shown exemplarily for the MNIST handwritten digit dataset. Indicated by color coding in each diagram is the euclidean distance between prototypes at times current time t and t_{incr}, for each prototype with index x/y. For better understanding, the color-coded distance map has been augmented by linear interpolation, and a visualization of prototypes at time $t > t_{\text{incr}}$ is overlaid over each distance image.

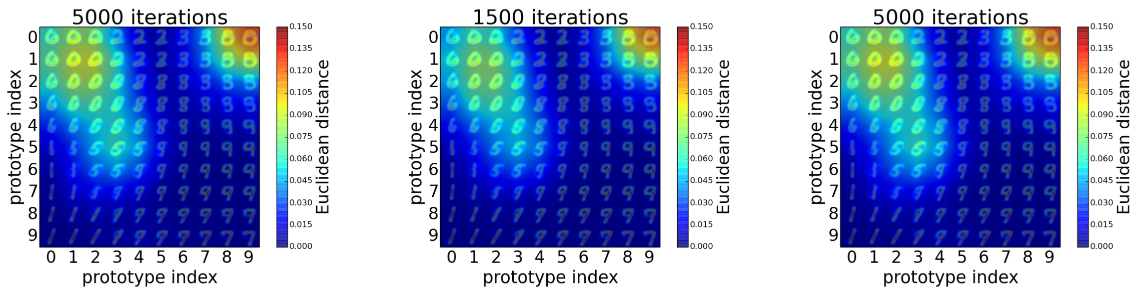

Fig. 7 Protecting the ReST layer against sampling bias, shown exemplarily for the MNIST handwritten digit dataset. Color coding in each diagram indicates the euclidean distance between prototypes at times t and t_{incr}, for each prototype with index x/y. For better understanding, the color-coded distance map has been augmented by linear interpolation, and a visualization of prototypes at time $t > t_{\text{incr}}$ is overlaid over each distance image.

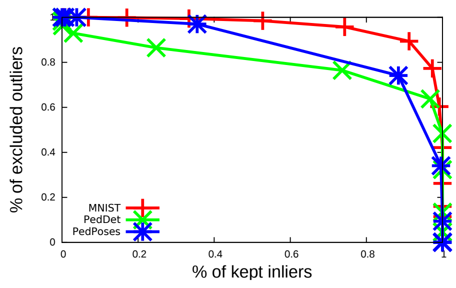

Fig. 8 Elimination of outliers/concept drift using activities in the ReST layer only, shown for the three datasets used in this study, see text for details.

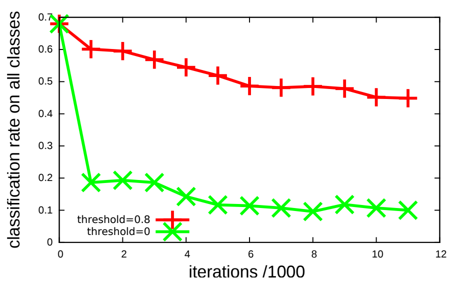

Fig. 9 Effect of the threshold θ during incremental learning. Shown is classification performance on the test set of the pose classification task during incremental learning of the left-out "left" class with $\theta = 0.8$ (red curve) and $\theta = 0$ (green curve).

a sample exceeds 0.7, ReST adaptation is not carried out any longer. By varying this parameter one can effectively control forgetting in the ReST layer H. The results are shown in Fig. 7. We observe that, as t increases, ReST prototypes are updated and subsequently protected against unnecessary adaptation as the new class is sufficiently well recognized by the read-out mechanism. This figure should be compared to Fig. 6 where ReST protection is disabled.

3.5 Concept drift detection

For this experiment, we ask the question whether the ReST layer is able to detect the presence of a new class in the training data, or, more precisely, the percentage of samples from this class that can be recognized as "outliers". Using standard parameters and a hidden layer size of $K = 10$, we train a PROPRE2.0 architecture on all three datasets described in Sec. 2.1 until $t = t_{\text{incr}}$. For each dataset, class 0 (digit 0 for MNIST, the "background" class for pedestrian detection and the "left" class for pose classification) is excluded from training. At $t > t_{\text{incr}}$, the previously excluded class is exclusively presented whereas adaptation is entirely disabled by setting $\alpha = 0$, $\alpha_{\text{LR}} = 0$ and $\alpha_d = 0$. It is assumed that outlier samples generate less BMU activity than inliers in the ReST layer, therefore a threshold θ_{out} is applied to the hidden layer $a^{(H)}_{ni}$, and a sample n is labelled as an outlier if $\max_i a^{(H)}_{ni} < \theta_{\text{out}}$. By varying θ_{out} in a $[0, 10]$ interval, graphs reminiscent of ROCs can be obtained that indicate how effective this strategy is at detecting outliers while letting pass inliers. These results are depicted, for the three datasets, in Fig. 8. Please note that no supervision information has been used to obtain these results which are essentially classification results, only the intrinsic similarity grouping detected by the hidden ReST layer. As can be expected, MNIST as the cleanest dataset performs best here but performance on the pedestrian datasets is acceptable as well.

3.6 Concept drift detection for controlling readout

An interesting finding is that the readout layer can be strongly "damaged" by the advent of concept drift of the hidden ReST layer H is not managed appropriately. The addition of a class the ReST layer was not trained on will, in general, cause diffuse, low-level activity. The readout mechanism will however try to map this essentially random activity to the new class label, therefore destroying readout weights all over the internal ReST layer. This is the reason behind the introduction of the threshold θ in eqn.(41): it will suppress small "random" activities until such time as the ReST layer has partially converged on the new class, which is indicated by super-threshold ReST activity. We show this effect exemplarily for the pedestrian pose classification task (the behavior is exactly the same for all three datasets) in Fig. 9. Using standard settings and a hidden layer size of $K = 10$, we train the PROPRE2.0 architecture on all classes but the "left" class until t_{incr}. Subsequently, we add the "left" class, one time with $\theta = 0.8$ and another time using $\theta = 0$. In parallel, we monitor the classification accuracy over time on the test (in which all classes are present) set every 1000 iterations. Results can be observed in Fig. 9 and show markedly inferior performance when no threshold is used. We can therefore conclude that thresholding does what it was meant to do: it "protects" the readout layer against concept drift effects.

3.7 ReST topology protection

As the focus of this article is on incremental learning scenarios where a new class is abruptly introduced, it is important to ensure the continued topological organization of the internal ReST layer H. As explained in Sec. 2.2, the ReST neighbourhood radius is reduced from an initially large value S_0 to an asymptotic value S_∞ in order to guarantee topologically ordered prototypes. As seen in Sec. 3.1, S_∞ needs to be small in order to ensure good classification performance of the whole architecture. Therefore, a dilemma presents it-

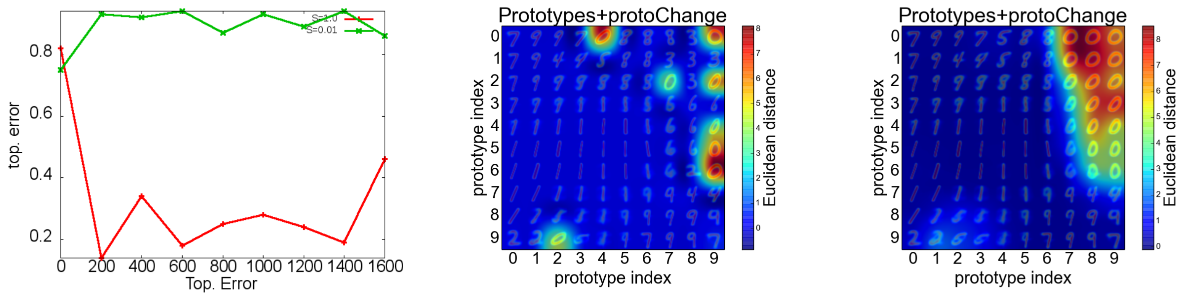

Fig. 10 Left: topographic error e_{top} over time when adding a new class, depending on whether ReST topology is "protected" or not. Middle: prototype changes at $t = t_{\text{incr}} + 1000$ w.r.t. $t = t_{\text{incr}}$, no topology preservation. Right: same diagram only this time with topology preservation. We observe that transition in the latter case is smoother and no structural defects are introduced, which is further corroborated by the consistently lower e_{top}. The price is of course the "overwriting" of a larger area of the original ReST layer.

self: adding a new class using S_∞ will presumably be good for immediate classification accuracy but may conceivably introduce strong topological defects, breaking the assumption that close-by prototypes are close in data space as well. As this is the fundamental principle on which we build incremental learning on top of the SOM model, the formation of topological defects must be avoided by appropriate mechanism for ensuring long-term incremental learning capacity. A simple way is to moderately increase the neighbourhood radius to an in-between value of S_{incr} when a new class is presented, and smoothly reducing it to S_∞ over time. In order to test whether this approach is necessary and, if so, feasible, we conduct a simple experiment using the MNIST benchmark: training is conducted normally (see Sec. 3) with a hidden ReST layer size of $K = 10$ using all classes except the class 0. In contrast to Sec. 3, we use parameter values of $\theta_m = 0.1$ to accelerate incremental learning. At $t = t_{\text{incr}}$, class 0 is presented exclusively for 6.000 iterations. The experiment is conducted twice, one time with $S_{\text{incr}} = 0.2$ and one time with $S_{\text{incr}} = S_\infty$ throughout the remaining experiment. The development of the topographic error e_{top} and a visualization of the prototype modifications are given in Fig. 10. They show that topological ordering is preserved better when S_{incr} is higher (which of course incurs a lower classification accuracy as more prototypes are overwritten).

4 Discussion and conclusion

The basic message of this article is that self-organized models like ReST are an extremely useful tool for performing incremental learning if appropriately managed. We presented the PROPRE2.0 architecture for incremental learning in very high dimensions which draws its incremental learning capacity from its internal ReST layer. The internal ReST layer exhibits localized and gradual update behavior as demonstrated in Sec. 3.3 which avoids catastrophic forgetting, but can also detect concept drift by BMU analysis (see Sec. 3.5. Based on this ability, we presented PROPRE's mechanisms of controlling its ReST layer in order maintain topological organization (see Sec. 3.7) and stability in the face of sampling bias (see Sec. 3.4), and verified that they are necessary ingredients when performing incremental learning with SOM-like models.

The basic benefit of the new ReST model is its self-adaptation capability which allows architecture parameters like learning rates to be set in a task-independent way. It furthermore permits a task-independent, probabilistic interpretation of neural activities which is important in the context of generic outlier or drift detection. Another technical benefit is the existence of a C^∞ energy function which ensures convergence and allows the use of automatic gradient computation and advanced gradient descent algorithms that are offered by modern machine learning frameworks.

4.1 Performance comparison to other methods

Although the main purpose of this article was to present points that make self-organized neural layers ideal candidates for incremental learning, we can compare the results presented in Sec. 3.1 to results presented in the literature, at least for MNIST which is commonly used to benchmark incremental learning. In particular, the works presented in [27] and [60] use (among others) an experimental setup identical to the one presented here (termed "D9-1" task), meaning that initial training is conducted on all but one class, and retraining on the remaining class. This is evaluated for a variety of models, including EWC, LWTA, IMM and Dropout (see Sec. 1.2). Furthermore, models like HAT[25] can be excluded from this comparison since HAT assumes that it is known whether a samples comes from initially-trained or re-trained classes for selecting the proper readout weights from the multi-head output layer.

Consulting Tab. 5 and [27], we conclude that PROPRE2.0 has a performance competitive to EWC, as evaluated in, and outperforms all other methods which show dramatic catastrophic forgetting behavior.

4.2 Outlook

It is clear that incremental learning as performed here (first train with $P-1$ classes, then train with remaining class only) would need to be complemented by a re-training phase where all P classes are presented together in order to compensate for ReST model units that now respond to the newly added class, and indeed it has been shown in [10] that this is a perfectly viable strategy. Other incremental learning methods such as LWPR who have adaptive model complexity can avoid such steps as the addition of a new class does not impair the representation of the "old" ones as long as there is little overlap. Since we wish, for reasons of real-world processing capacity, to have constant model complexity guaranteeing a fixed upper bound on computational resources, zero-loss incremental learning seems difficult, and it will be interesting to devise efficient retraining strategies to counter this.

Immediate next steps will, first of all, consist of creating a deep PROPRE2.0 architecture, with many hidden ReST layers, because currently PROPRE2.0 gives only mediocre results on complex problems like COIL100 or SVHN, which arguably require deep architecture for being well solved. Technically, deep PROPRE2.0 could be realized by complementing the ReST energy function by a supervised term. To keep optimization feasible, this could again be cast as a constrained optimization problem, where the supervised gradient is forced to be parallel to the ReST gradient, or the other way round. Finally, it will have to be investigated how incremental learning can work in a deep architecture that performs online learning, as layers can now converge at very different times.

Most importantly, PROPRE2.0 and potential extension will have to benchmarked in a thorough way on a large set of difficult problems in order to ensure its applicability in a convincing way.

References

1. A Gepperth and B Hammer. Incremental learning algorithms and applications. 2016. 1
2. Sethu Vijayakumar, Aaron D'souza, and Stefan Schaal. Incremental online learning in high dimensions. *Neural computation*, 17(12):2602–2634, 2005. 1
3. R.J. May, H.R. Maier, and G.C. Dandy. Data splitting for artificial neural networks using som-based stratified sampling. *Neural Networks*, 23(2):283 – 294, 2010. 1
4. M. McCloskey and N. Cohen. Catastrophic interference in connectionist networks: the sequential learning problem. In G. H. Bower, editor, *The psychology of learning and motivation*, volume 24. 1989. 2, 3
5. R. Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological Review*, 97, 1990. 2, 3
6. RM French. Semi-distributed representations and catastrophic forgetting in connectionist networks. *Connect. Sci.*, 4, 1992. 2
7. RM French. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychol Rev.*, 97(2), 1990. 2
8. McCloskey M and Cohen NJ. Catastrophic interference in connectionist networks: the sequential learning problem. *Psychol. Learn. Motiv.*, 24, 1989. 2
9. T Kohonen. Self-organized formation of topologically correct feature maps. *Biol. Cybernet.*, 43:59–69, 1982. 2, 4, 7
10. A Gepperth and C Karaoguz. A bio-inspired incremental learning architecture for applied perceptual problems. *Cognitive Computation*, 2015. accepted. 2, 3, 9, 16
11. A Gepperth and M Lefort. Biologically inspired incremental learning for high-dimensional spaces. In *IEEE International Conference on Development and Learning (ICDL)*, 2015. 2
12. Pallavi Kulkarni and Roshani Ade. Incremental learning from unbalanced data with concept class, concept drift and missing features: a review. *International Journal of Data Mining and Knowledge Management Process*, 4(6), 2014. 2
13. Alexey Tsymbal. The problem of concept drift: definitions and related work. Technical report, Computer Science Department, Trinity College Dublin, 2004. 2
14. Y. M. Wen and B. L. Lu. Incremental learning of support vector machines by classifier combining. In *Proc. of 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2007)*, volume 4426 of *LNCS*, 2007. 3
15. Robi Polikar, Lalita Upda, Satish S Upda, and Vasant Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 31(4):497–508, 2001. 3
16. N. Sharkey and A. Sharkey. An analysis of catastrophic interference. *Connection Science*, 7(3-4), 1995. 3
17. R. M. French. Dynamically constraining connectionist networks to produce distributed, orthogonal representations to reduce catastrophic interference. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*. 1994. 3
18. J. Murre. The effects of pattern presentation on interference in backpropagation networks. In *Proceedings of the 14th Annual Conference of the Cognitive Science Society*. 1992. 3
19. C. Kortge. Episodic memory in connectionist networks. In *Proceedings of the 12th Annual Conference of the Cognitive Science Society*. 1990. 3
20. Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013. 3
21. J. Krushke. ALCOVE: An exemplar-based model of category learning. *Psychological Review*, 99, 1992. 3
22. S. Sloman and D. Rumelhart. Reducing interference in distributed memories through episodic gating. In A. Healy and S. Kosslynand R. Shiffrin, editors, *Essays in Honor of W. K. Estes*. 1992. 3
23. German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *arXiv preprint arXiv:1802.07569*, 2018. 3
24. Alexander Gepperth and Barbara Hammer. Incremental learning algorithms and applications. *European Symposium on Artificial Neural Networks (ESANN)*, (April):357–368, 2016. 3
25. Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4548–4557. PMLR, 2018. 3, 4, 15
26. Ronald Kemker, Marc McClure, Angelina Abitino, Tyler L Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 3
27. B Pfülb and A Gepperth. A comprehensive, application-oriented study of catastrophic forgetting in dnns. In *International Conference on Learning Representations (ICLR)*, 2019. accepted. 3, 15

28. Boya Ren, Hongzhi Wang, Jianzhong Li, and Hong Gao. Lifelong learning based on dynamic combination model. *Applied Soft Computing Journal*, 56:398–404, 2017. 3
29. Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017. 3
30. Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999, 2017. 3
31. Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. *arXiv preprint arXiv:1711.10563*, 2017. 3
32. Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. iCARL: Incremental classifier and representation learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5533–5542. IEEE, 2017. 3
33. Rahaf Aljundi, Marcus Rohrbach, and Tinne Tuytelaars. Selfless sequential learning. *arXiv preprint arXiv:1806.05421*, 2018. 3, 5
34. Rupesh Kumar Srivastava, Jonathan Masci, Sohrob Kazerounian, Faustino Gomez, and Jürgen Schmidhuber. Compete to Compute. In *Advances in Neural Information Processing Systems*, pages 2310–2318, 2013. 3, 5
35. James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, page 201611835, 2017. 3, 5
36. Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *Advances in Neural Information Processing Systems*, pages 4652–4662, 2017. 3, 5
37. Rupesh K Srivastava, Jonathan Masci, Sohrob Kazerounian, Faustino Gomez, and Jürgen Schmidhuber. Compete to compute. In *Advances in neural information processing systems*, pages 2310–2318, 2013. 3
38. Amir Rosenfeld and John K Tsotsos. Incremental learning through deep adaptation. *arXiv preprint arXiv:1705.04228*, 2017. 3
39. Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. Incremental learning of object detectors without catastrophic forgetting. *arXiv preprint arXiv:1708.06977*, 2017. 3
40. Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proc. CVPR*, 2017. 3
41. Hyo-Eun Kim, Seungwook Kim, and Jaehwan Lee. Keep and learn: Continual learning by constraining the latent space for knowledge preservation in neural networks. *arXiv preprint arXiv:1805.10784*, 2018. 3
42. S. Vijayakumar and S. Schaal. Locally weighted projection regression: An o(n) algorithm for incremental real time learning in high-dimensional spaces. In *International Conference on Machine Learning*, 2000. 3, 4
43. D. Nguyen-Tuong and J. Peters. Local gaussian processes regression for real-time model-based robot control. In *IEEE/RSJ International Conference on Intelligent Robot Systems*, 2008. 3
44. O. Sigaud, C. Sagaün, and V. Padois. On-line regression algorithms for learning mechanical models of robots: A survey. *Robotics and Autonomous Systems*, 2011. 3
45. M. Butz, D. Goldberg, and P. Lanzi. Computational complexity of the xcs classifier system. *Foundations of Learning Classifier Systems*, 51, 2005. 3
46. T. Cederborg, M. Li, A. Baranes, and P.-Y. Oudeyer. Incremental local online gaussian mixture regression for imitation learning of multiple tasks. 2010. 3
47. Gina G Turrigiano and Sacha B Nelson. Homeostatic plasticity in the developing nervous system. *Nature Reviews Neuroscience*, 5(2):97, 2004. 4
48. Nicholas J. Butko and Jochen Triesch. Learning sensory representations with intrinsic plasticity. *Neurocomputing*, 70(7):1130 – 1138, 2007. Advances in Computational Intelligence and Learning. 4
49. György Buzsáki and Kenji Mizuseki. The log-dynamic brain: how skewed distributions affect network operations. *Nature Reviews Neuroscience*, 15(4):264, 2014. 4
50. Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456, 2015. 4
51. Tom M Heskes and Bert Kappen. Error potentials for self-organization. In *Neural Networks, 1993., IEEE International Conference on*, pages 1219–1223. IEEE, 1993. 4, 7
52. Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013. 4
53. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*, pages 306–351. IEEE Press, 2001. 5
54. M. Enzweiler and D.M. Gavrila. Monocular pedestrian detection: Survey and experiments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(12):2179–2195, 2009. 5
55. N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005. 5
56. M Lefort, T Hecht, and A Gepperth. Using self-organizing maps for regression: the importance of the output function. In *European Symposium On Artificial Neural Networks (ESANN)*, 2015. 8
57. Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016. 9
58. Daniel Polani. Measures for the organization of self-organizing maps. In *Self-Organizing neural networks*, pages 13–44. Springer, 2002. 11
59. A Gepperth. Catastrophic forgetting: still a problem for deep neural networks. In *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2018. 12
60. B Pfülb, A Gepperth, S Abdullah, and A Krawczyk. Catastrophic forgetting: still a problem for dnns. In *International Conference on Artificial Neural Networks (ICANN)*. 15

Appendix A

Figs. 11, 12, 13 give a detailed comparison of all neural activities in a ReST layer with and without self-adaptation. This is a complement to the experiments in Sec. 3.2 which was shifted to the appendix for reasons of readability.

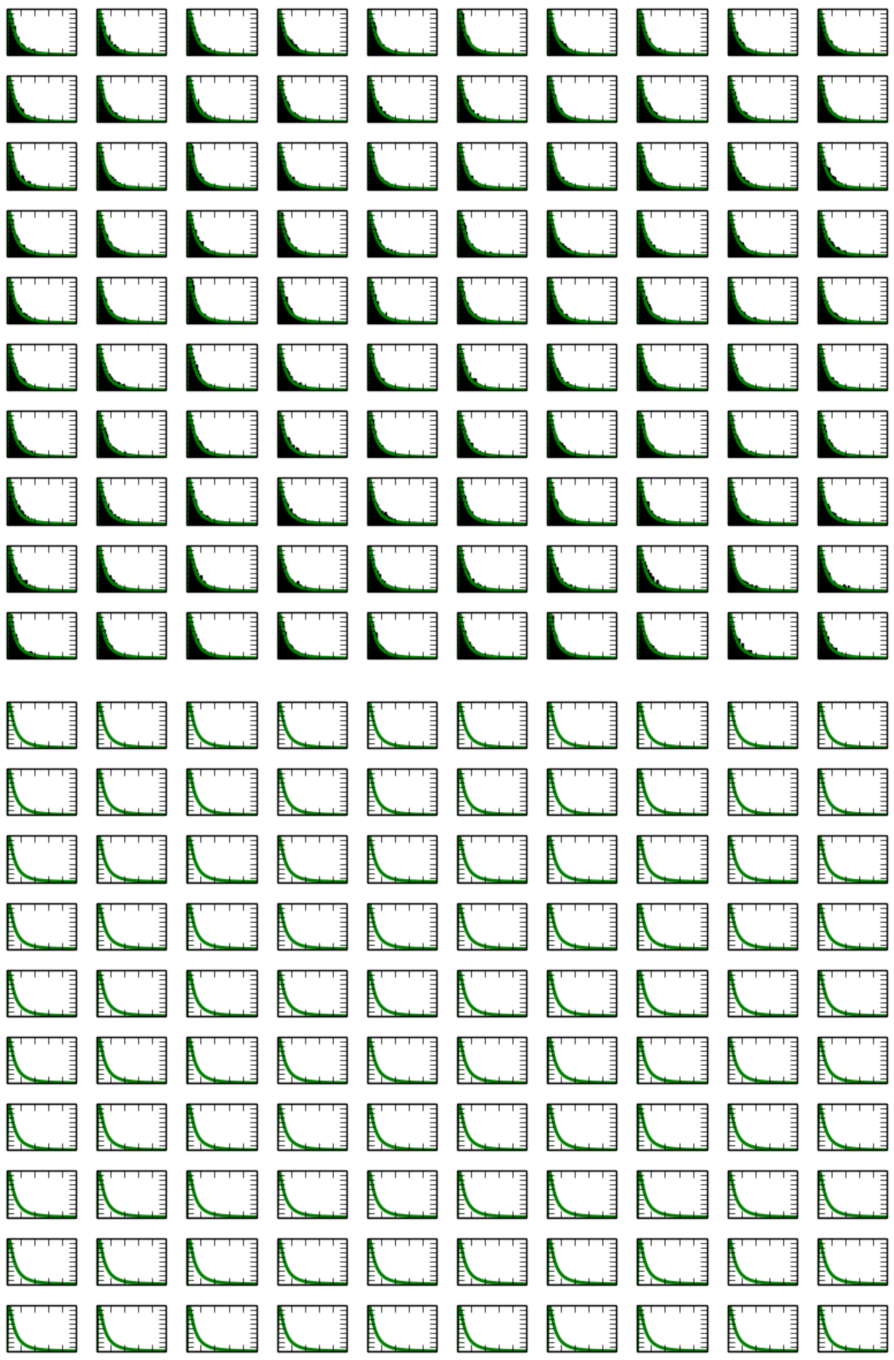

Fig. 11 Effects of self-adaptation on hidden layer ReST activities for the pose classification task ("poses"). Self-adaptation is enabled in the upper and disabled in the lower diagram. Each diagram includes an activity histogram for every ReST neuron, giving 10×10=100 histograms. The interpretation of individual is identical to the one in Fig. 5. In particular, the targeted log-normal distribution is superimposed as a solid green line.

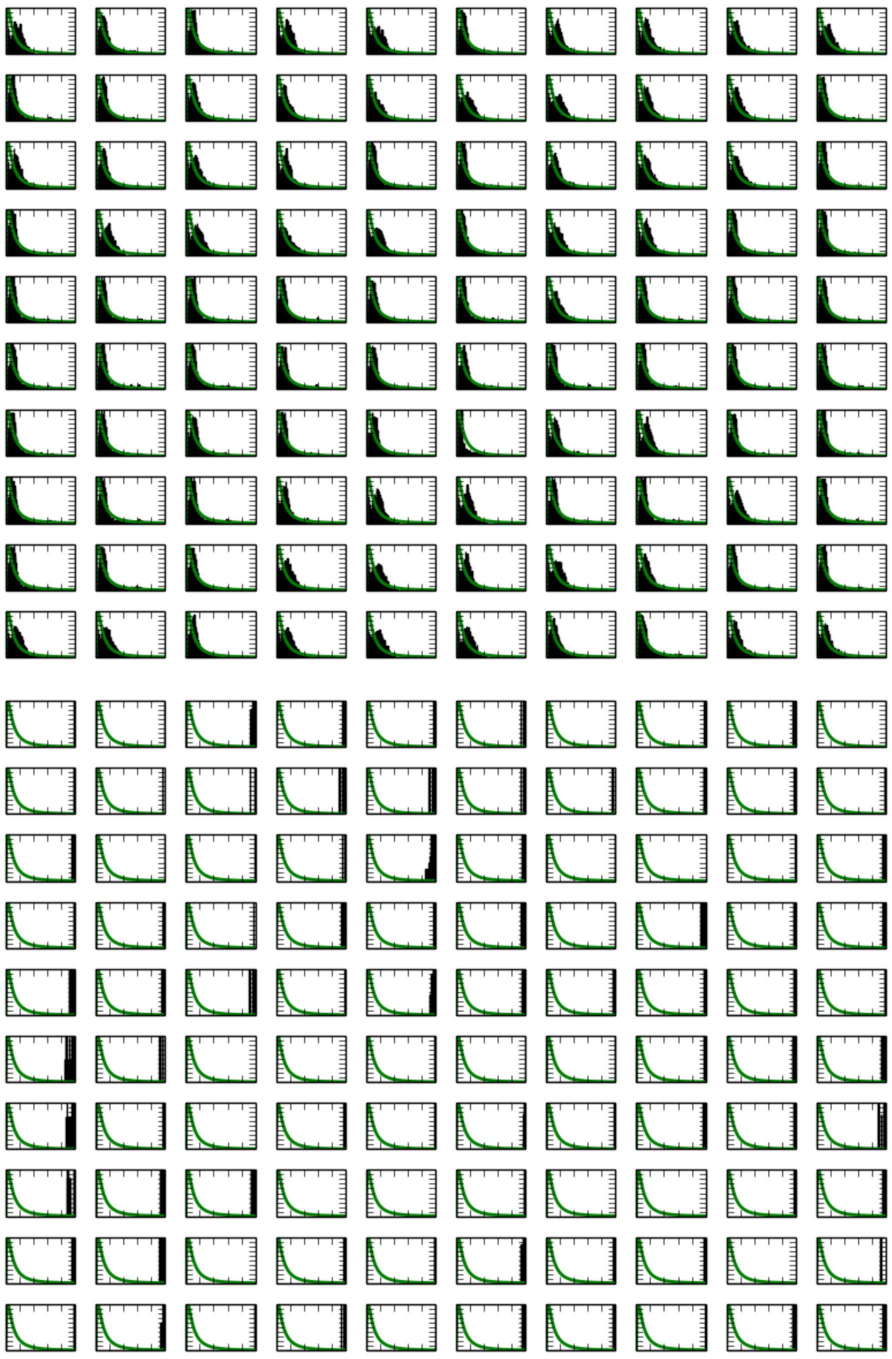

Fig. 12 Effects of self-adaptation on hidden layer ReST activities for the pedestrian detection task (”peddet”). Self-adaptation is enabled in the upper and disabled in the lower diagram. Each diagram includes an activity histogram for every ReST neuron, giving 10×10=100 histograms. The interpretation of individual is identical to the one in Fig. 5. In particular, the targeted log-normal distribution is superimposed as a solid green line.

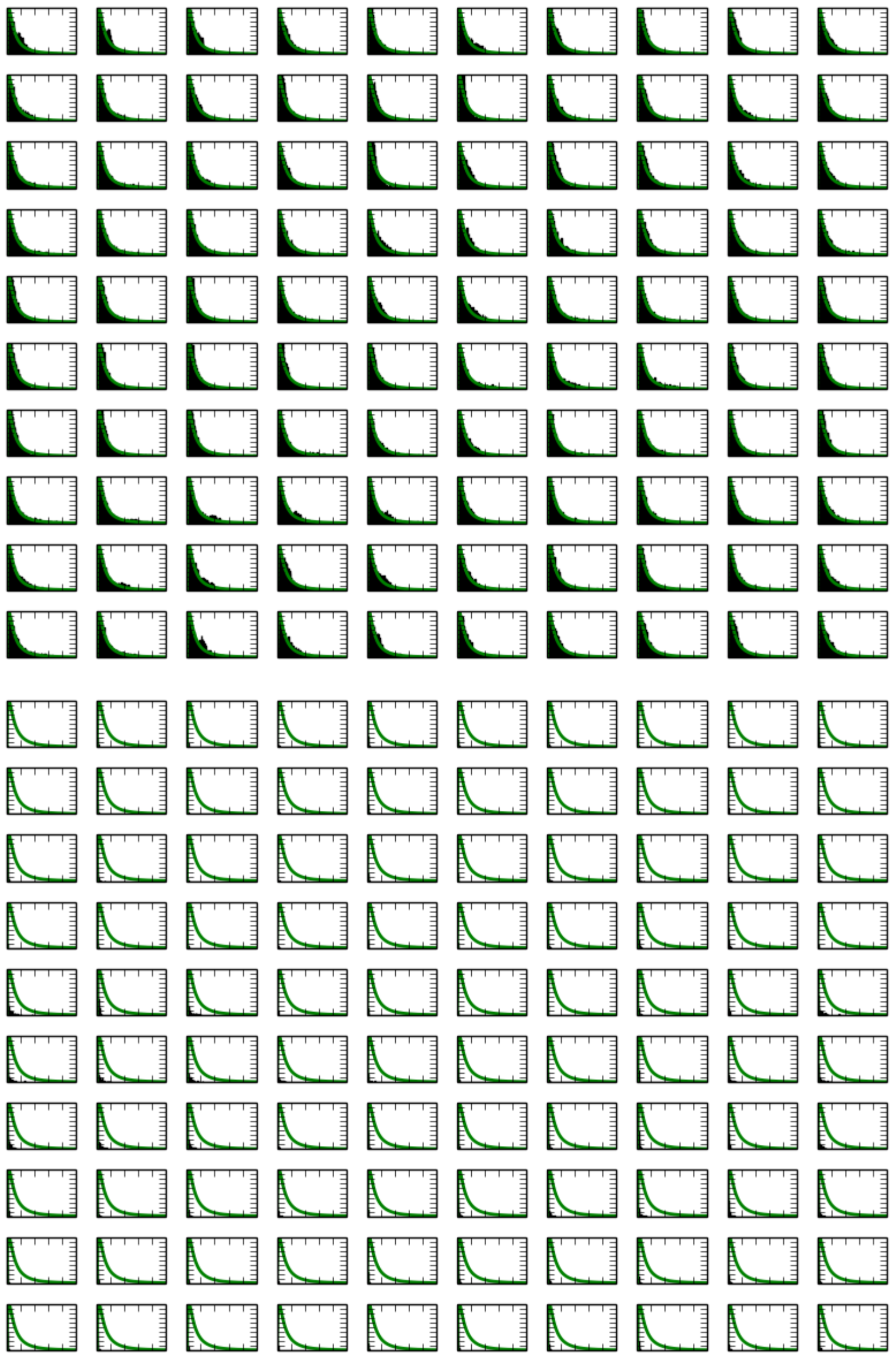
Fig. 13 Effects of self-adaptation on hidden layer ReST activities for MNIST. Self-adaptation is enabled in the upper and disabled in the lower diagram. Each diagram includes an activity histogram for every ReST neuron, giving 10×10=100 histograms. The interpretation of individual is identical to the one in Fig. 5. In particular, the targeted log-normal distribution is superimposed as a solid green line.