Towards incremental deep learning: multi-level change detection in a hierarchical recognition architecture

Thomas Hecht, Alexander Gepperth *

U2IS, ENSTA ParisTech
Inria, Université Paris-Saclay
828 bd des Maréchaux, 91762 Palaiseau Cedex
France

Abstract. We present a trainable hierarchical architecture capable of detecting newness (or outliers) at all hierarchical levels. This contribution paves the way for deep neural architectures that are able to learn in an incremental fashion, for which the ability to detect newness is an indispensable prerequisite. We verify the ability to detect newness by conducting experiments on the MNIST database, where we introduce either localized changes, by adding noise to a small patch of the input, or global changes, by changing the global arrangement of local patterns which is not detectable at the local level.

1 Introduction

This study is conducted in the context of hierarchical incremental learning algorithms. It is a proof-of-concept type study, showing the capacity to detect newness (or outliers) in a minimal hierarchical architecture operating on a semi-realistic but widely known and accepted benchmark database in machine learning, the MNIST dataset [1].

Incremental learning algorithms are, roughly speaking, algorithms that can take examples one by one, without knowing their number in advance, and which can deal with changes in the input statistics without requiring a complete retraining with all data. An important prerequisite for incremental learning is the detection of newness or change: if an algorithm is supposed to track changing data statistics, it first of all needs to be able to detect change[2]. To our knowledge, all existing incremental learning algorithms possess this property (see [3] for a review): some algorithms achieve it by an adaptive subdivision of the input space, e.g., [4, 5, 6, 7], where local models are learned in each sub-volume, other achieve this by using prototypes to directly approximate the data distribution in input space[8, 9].

On the other hand, while hierarchical learning algorithms are currently showing very promising perspectives, all existing incremental learning algorithms are, to our knowledge, "flat" architectures: Although the input space is decomposed into sub-volumes, the input vectors themselves are not, such as convolutional

*Dr. Alexander Gepperth is with Inria Flowers. Thomas Hecht gratefully acknowledges funding support by the "Direction Générale de l'Armement" (DGA) and École Polytechnique.

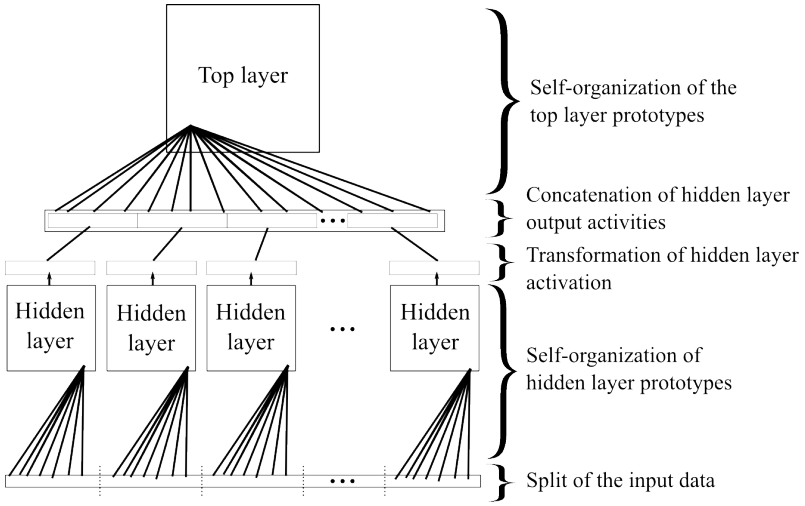

Fig. 1: The minimalist hierarchical system used in this study, composed of input layer, hidden layer and top layer.

neural networks do when they analyze their inputs by local receptive fields. Especially prototype-based incremental learning methods[9, 8] could profit greatly from hierarchies namely by strongly reduced memory requirements for storing prototypes. In this case, change detection would have to work on all levels of a hierarchical incremental architecture, apart from the issue of learning rules that would *react* to such changes which will not be treated here. The goal of this contribution is finally to demonstrate multi-level change detection in a minimalist hierarchical extension of the prototype-based incremental architecture proposed in [9]. This architecture, for which excellent performance on real-world benchmarks has been demonstrated [10, 9], is closely related to other prototype-based methods such as GMLVQ[11] and notably ioLVQ[8].

The architecture contains three layers as depicted in Fig. 1, where separate hidden-layer *maps* try to represent local receptive fields in the input, and the top-layer map aims at representing the union of all hidden-layer map activities[1].

Change detection should be activated in different layers depending on the type of change that occurs: if it exclusively occurs inside a local receptive field, the hidden layer activities responsible for this receptive field should signal change, whereas the top layer should react in case of global changes. Such global changes may not even be detectable in the hidden layer because they can leave hidden-layer statistics completely unaffected.

[1]Normally, the top layer would be connected to a readout mechanism inferring class attributes as in [9], but, as classification performance on the MNIST database has already been demonstrated in [9] for the "flat" system, we omit it here and focus on change detection.

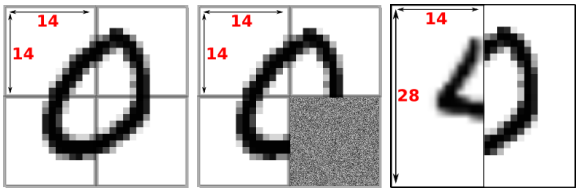

Fig. 2: Illustration of the experimental protocol. Left: partitioning of a 28x28 MNIST sample into 2x2 receptive fields (RFs) of 14x14 pixels each. Middle: application of a local change, replacing a single RF by white noise. Right: application of global change, by mixing the left and right halves of different MNIST samples. The local patterns inside the RFs are not, statistically speaking, affected by this operation.

2 Methods

For representing both hidden and top layer inputs, we use a prototype-based learning algorithm which is loosely based on the self-organizing map model, see [9]. Inputs are represented by graded neural activities arranged in *maps* of $n \times n$ units, organized on a two-dimensional grid lattice. Each unit's associated weight vector (its *prototype*) is updated using the learning rule and good practices proposed by Kohonen[12] which decreases learning rate $\epsilon(t)$ and Gaussian neighborhood radius $\sigma(t)$ from initially large values to their asymptotic values ϵ_∞, σ_∞. Since the hidden-layer map activities are inputs to the top layer, they need to be fully converged before top-layer selectivities are learned, which needs to be ensured by a proper temporal organization of the training phase, see Sec. 3.

For any map X, the *map activity* $z_{ij}^X(t)$ at position (i, j) is derived from the Euclidean distance between the unit prototype $w_{ij}^X(t)$ and the current input $x(t)$:

$$z_{ij}^X(t) = \mathrm{TF}\left(g_\kappa\left(||w_{ij}^Z(t) - x(t)||\right)\right) \tag{1}$$

where, as described in [9], $g_\kappa(\cdot)$ is a Gaussian function with an adaptive parameter κ that converts distances into the $[0, 1]$ interval, and $\mathrm{TF}(\cdot)$ is a *transfer function*. Differently from [9], we chose a transfer function that just keeps the activity of the *best-matching unit* at coordinates (i^*, j^*):

$$\mathrm{TF}(x, i, j) = \begin{cases} x \text{ if } (i, j) = (i^*, j^*) \\ 0 \text{ otherwise} \end{cases} \tag{2}$$

3 Experiments

We fix the sizes of all hidden-layer maps to $n \times n = 6 \times 6$; whereas the top-layer map has size 10×10. As we apply our architecture to the MNIST database [1], the original sample size is 28×28 pixels. We use non-overlapping receptive fields of 14×14 pixels (see Fig. 2), which leads to 2x2=4 hidden-layer maps. The asymptotic values of learning rate and neighbourhood radius are $\epsilon_\infty = 0.001$

and $\sigma_\infty = 0.1$ for all maps in the system. The principal quantity investigated in this study is the activity of the best-matching unit for a map X, denoted $z^{*X}(t)$, which is inversely related to the Euclidean distance between sample and best-matching prototype. A significant deviation of $z^{*X}(t)$ from its long-term average value is interpreted as an indication of change.

3.1 Protocol

An experiment consists of $T = 200000$ random pattern presentations and is conducted in three phases:

Hidden layer training From step 0 to 45.000, each hidden-layer map is trained with while top-layer learning and adaptive distance κ updating, see [9], is disabled.

Top layer training From step 45.000 to 140.000, hidden-layer learning and adaptive distance κ updating, see [9], is disabled and the top-layer map is trained using the concatenation of hidden-layer map activations.

Testing From step 140.000 to the end, the data distribution is artificially changed, without any prototype or adaptive distance adaptation.

Changes in data distribution can be either at the *local level* or the *global level*, see Fig. 2. The former corresponds to a localized change within an RF which should be detectable mainly by the hidden-layer maps. The latter should affect the top-level map alone as only the global arrangement of patterns within each RF changes w.r.t. the learning phase but not the local patterns themselves. Local changes are produced by adding uniform noise, $\mathcal{U}(0, 1)$, to a single 14×14 pixels RF content. Global changes are obtained by concatenating the left and right half of two randomly chosen input samples, giving again a 28×28 sample.

3.2 Results

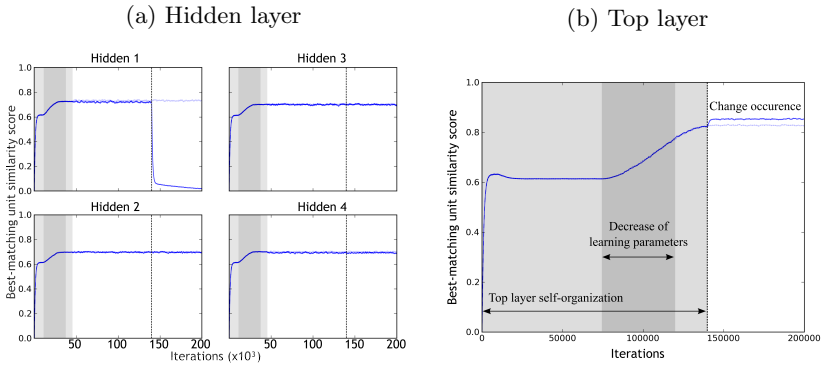

Fig. 3: Detection of low-level change. Shown are best-matching unit activations over time for all four hidden-layer maps (left) and top-layer map (right). Time series have been post-processed by exponential smoothing with a parameter of $\alpha = 0.005$ for better visualization. Dotted lines indicate the case of no change.

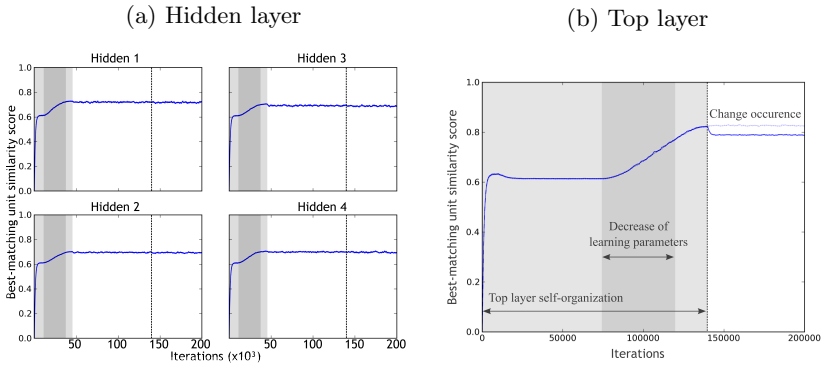

Fig. 4: Detection of high-level change. Shown are best-matching unit activations over time for all four hidden-layer maps (left) and top-layer map (right). Time series have been post-processed by exponential smoothing with a parameter of $\alpha = 0.005$ for better visualization. Dotted lines indicate the case of no change.

Whether the neural architecture is fed with locally or globally modified inputs starting at $t = 140.000$, the architecture clearly detects that a change occurred in the data distribution, see Figs. 4, 3. Furthermore, the hierarchical organization of the architecture allows to more accurately identify the type of change: Fig. 4 shows that a high-level change (see Fig 2) can be detected in the top-level map while hidden-layer activations stay unchanged. For local changes (see Fig 2), the reverse is the case: best-matching unit activations in the hidden-layer map associated with the noised signal are severely affected. We observe a slight change in average top-level layer activity as well, which is understandable as top-level activations directly depend hidden-layer activations, so a change in the former affects the latter. With a finer subdivision into receptive fields, a local change in one RF would not have such a significant impact on top-level activations.

4 Discussion, conclusion, perspectives

This study has, first of all, outlined how a prototype-based incremental learning architecture that is "flat" could be generalized to a hierarchical architecture. In this generalized architecture, we showed that it is possible to detect changes in input statistics at all levels of the hierarchy. As change detection is an important prerequisite for incremental learning, see Sec. 1, we remove an important conceptual obstacle to using hierarchical prototype-based architectures in an incremental fashion. This in turn is a desirable thing as prototype-based methods are particularly suited for incremental learning[8, 9].

Hierarchical prototype-based methods might also remove another major obstacle to wide-spread use: the curse of dimensionality. For difficult problems in high-dimensional spaces, it stands to reason that a very high number of prototypes might be required to accurately map data distributions. If however a problem is partly factorisable, i.e., can be simplified by subdividing the input

into receptive fields whose content is approximately independent (as it is often the case in visual processing), the size of the prototypes to be stored reduces dramatically. If we denote the number of top-level units in the presented architecture by $T = 100$, the number of hidden layer maps (each composed of h units) by $H = 4$, and the input dimensionality by $I = 784$, we obtain for the total number of weights $W = h(I + TH) = 50.400$. For a comparable "flat" system without hidden layer we would need $\tilde{W} = IT = 78400$ connections already. This difference between these two quantities can be increased by varying the free parameter h: with maps of size $h = 4 \times 4 = 16$ in the hidden layer, we obtain $W = 22400$. If recognition performance is maintained, this offers a promising way around the curse of dimensionality for prototype-based methods in general.

In future work, will wish to explore precisely these questions: can the presented (non-incremental) architecture obtain comparable classification results (for MNIST, this seems to be the case but tests on other problems are needed)? How far can we reduce h without sacrificing performance? And, most importantly, we will investigate learning rules that can in a hierarchical incremental learning architecture, with the goal of pushing the capacities of incremental learning architectures beyond the current state of the art.

References

[1] Yann LeCun and Corinna Cortes. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2010.

[2] 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.

[3] Olivier Sigaud, Camille Salaün, and Vincent Padois. On-line regression algorithms for learning mechanical models of robots: a survey. *Robotics and Autonomous Systems*, 59(12):1115–1129, 2011.

[4] Sethu Vijayakumar, Aaron D'souza, and Stefan Schaal. Incremental online learning in high dimensions. *Neural computation*, 17(12):2602–2634, 2005.

[5] 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.

[6] T. Cederborg, M. Li, A. Baranes, and P.-Y. Oudeyer. Incremental local online gaussian mixture regression for imitation learning of multiple tasks. 2010.

[7] M. Butz, D. Goldberg, and P. Lanzi. Computational complexity of the xcs classifier system. *Foundations of Learning Classifier Systems*, 51, 2005.

[8] L. Fischer, Hammer B., and H. Wersing. Certainty-based prototype insertion/deletion for classification with metric adaptation. In *Processings of the European Symposium on Artificial Neural Networks (ESANN)*, 2015.

[9] A Gepperth and C Karaoguz. A bio-inspired incremental learning architecture for applied perceptual problems. *Cognitive Computation*, 2015. accepted.

[10] Thomas Hecht, Alexander Gepperth, and Mandar Gogate. A generative learning approach to sensor fusion and change detection. *Cognitive Computation*, 2015.

[11] P. Schneider, M.Biehl, and B. Hammer. Adaptive relevance matrices in learning vector quantization. *Neural Computation*, 21(12), 2009.

[12] T. Kohonen. *Self-Organizing Maps*. Springer, third, extended edition edition, 2001.