Neurogenesis-Inspired Dictionary Learning: Online Model Adaption in a Changing World

Sahil Garg*, **Irina Rish**, **Guillermo Cecchi**, and **Aurelie Lozano**
IBM Thomas J. Watson Research Center
sahilgar@usc.edu, {rish, gcecchi, aclozano}@us.ibm.com

Abstract

We address the problem of online model adaptation when learning representations from non-stationary data streams. Specifically, we focus here on online dictionary learning (i.e. sparse linear autoencoder), and propose a simple but effective online model-selection approach involving "birth" (addition) and "death" (removal) of hidden units representing dictionary elements, in response to changing inputs; we draw inspiration from the *adult neurogenesis* phenomenon in the dentate gyrus of the hippocampus, known to be associated with better adaptation to new environments. Empirical evaluation on real-life datasets (images and text), as well as on synthetic data, demonstrates that the proposed approach can considerably outperform the state-of-art non-adaptive online sparse coding of [Mairal *et al.*, 2009] in the presence of non-stationary data. Moreover, we identify certain data- and model properties associated with such improvements.

1 Introduction

[1] Adaptation to a changing environment is essential for successful functioning of both natural and artificial intelligent systems. In human brains, adaptation is achieved via neuroplasticity, which includes synaptic plasticity, i.e. change in strength of neuronal connections, and (adult) neurogenesis [Kempermann, 2006], i.e. the birth and maturation of new neurons, accompanied by some neuronal death; hippocampal neurogenesis, in particular, is a "candidate mechanism for the specific dynamic and flexible aspects of learning" [Stuchlik, 2014].

In representation learning, which always involves a hidden-variable model such as dictionaries, autoencoders or feed-forward neural nets, probabilistic hidden-factor models, etc., synaptic plasticity is analogous to parameter learning (e.g., neural net weight training), while neurogenesis can be modeled as an online architecture adapation via adding/deleting hidden units (neurons).

However, optimal model selection in large-scale hidden-variable models, e.g., choosing the number of layers, hidden units, and their connectivity, can be intractable due to enormous search space size. An online approach to dynamically expanding and contracting model's architecture can serve as a potentially more effective alternative to the standard off-line model selection (e.g., MDL-based off-line sparse coding [Ramirez and Sapiro, 2012]), as well as to the currently popular network compression (distillation) approaches [Hinton *et al.*, 2015; Mariet and Sra, 2015; Srivastava *et al.*, 2014; Ba and Caruana, 2014; Bucilu *et al.*, 2006], which involve learning a large-scale model first, and compressing it to a smaller one later.

In this paper, we focus on dictionary learning, a.k.a. sparse coding [Olshausen and Field, 1997; Kreutz-Delgado *et al.*, 2003; Aharon *et al.*, 2006; Lee *et al.*, 2006] – a representation learning approach which finds a set of basis vectors (atoms, or dictionary elements) and representations (encodings) of the input samples as sparse linear combinations of those elements[2]. More specifically, our approach builds upon the computationally efficient *online dictionary-learning* method of [Mairal *et al.*, 2009], where the data samples are processed sequentially, one at a time (or in small batches). Online approaches are particularly important in large-scale applications with millions of potential training samples, where off-line learning can be infeasible; furthermore, online approaches are a natural choice for building systems capable of continual, lifelong learning.

We propose a novel online model-selection approach for dictionary learning, inspired by the neurogenesis process, involving addition and deletion of the elements, in response to the dynamically changing properties of input data. Namely, at each iteration corresponding to a new batch of data, a number of random dictionary elements is added (neuronal birth); higher representation error, indicating mismatch between the current dictionary and the new samples, triggers more neurogenesis. The neuronal death involves removing "useless" dic-

*The author contributed to this work as a research intern, who is currently a PhD student at the University of Southern California.

[1]The extended version of this paper is available at arxiv.org/abs/1701.06106 .

[2]This corresponds to a sparse, singe-hidden-layer linear autoencoder, where the hidden units correspond to dictionary elements, each element represented by a weight vector associated with the unit's outgoing links in the output layer, and the sparse vector of hidden unit activations correspond to the encoding of an input.

tionary elements, and is implemented by l_1/l_2 group-sparsity regularization; this step is essential in neurogenesis-inspired learning, since it reduces a potentially uncontrolled growth of the dictionary, and helps to avoid overfitting. Note that in natural adult neurogenesis, neuronal death is also an important factor balancing neuronal birth [Kempermann, 2006]).

Moreover, besides selecting the model complexity (i.e. the number of hidden units) in online sparse coding, our framework allows for online connectivity adaptation by imposing sparsity on dictionary elements, rather than just on codes; this is a more biologically plausible assumption than the full connectivity, i.e. dense dictionary elements, which also results into superior empirical performance.

Note that, although the group-sparsity constraint enforcing deletion of some dictionary elements was introduced earlier in the group-sparse coding method of [Bengio *et al.*, 2009], it was only implemented and tested in the off-line rather than online setting, and, most importantly, it was not accompanied by neurogenesis. On the other hand, while some prior works considered online node addition in hidden-variable models, and specifically, in neural networks, from cascade correlations [Fahlman and Lebiere, 1989] to the recent work by [Draelos *et al.*, 2016; Rusu *et al.*, 2016], no model pruning was incorporated in those approaches to balance the model expansion.

We demonstrate on simulated data and on two real-life datasets, i.e. natural images and language processing task, that our approach significantly outperform non-adaptive, fixed-dictionary-size online method of [Mairal *et al.*, 2009] when the input is non-stationary. Moreover, we identify certain data properties and parameter settings associated with such improvements. In summary, our work appears to be the first one to propose and evaluate, both empirically and theoretically, online model selection in dictionary learning under non-stationary inputs.

2 Background on Dictionary Learning

Traditional off-line *dictionary learning* [Olshausen and Field, 1997; Aharon *et al.*, 2006; Lee *et al.*, 2006], also known as *sparse coding*, aims at finding a *dictionary* $\boldsymbol{D} \in \mathbb{R}^{m\times k}$, which allows for an accurate encoding of each sample in the training data set $\boldsymbol{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n \in \mathbb{R}^m\}$ by a linear combinations of a *relatively small* (thus, sparse encoding) subset of *dictionary elements* $\{\boldsymbol{d}_1, \cdots, \boldsymbol{d}_k \in \mathbb{R}^m\}$. This is achieved by minimizing

$$f_n(D) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}||\boldsymbol{x}_i - \boldsymbol{D}\boldsymbol{\alpha}_i||_2^2 + \lambda_c||\boldsymbol{\alpha}_i||_1 \quad (1)$$

where the first term is the representation error, and the second term is the l_1-regularization which enforces the *codes* $\boldsymbol{\alpha}_i$ to be sparse. The joint minimization of $f_n(D)$ with respect to the dictionary and codes is non-convex, and commonly used approach is to use an alternating minimization over the codes and the dictionary.

However, the classical dictionary learning does not scale to very large datasets; moreover, it is not immediately applicable to online learning from a continuous stream of data. The *online dictionary learning (ODL)* method proposed by [Mairal *et al.*, 2009] overcomes both of these limitations, and serves as a basis for our proposed approach, presented in Alg. 1 in the next section. While the highlighted lines in Alg. 1 represent our extension of ODL , the non-highlighted ones are common to both approaches, and are discussed first. The algorithms start with some dictionary $\boldsymbol{D}^0$, e.g. a randomly initialized one (other approaches include using some of the inputs as dictionary elements [Mairal *et al.*, 2009; Bengio *et al.*, 2009]). At each iteration t, both online approaches consider the next input sample $\boldsymbol{x}_t$ (or a batch of samples), in step 3 and compute its sparse code $\boldsymbol{\alpha}_t$ by solving the LASSO [Tibshirani, 1996] problem (step 4), with respect to the current dictionary. Next, ODL computes the dictionary update, $\boldsymbol{D}^{(t)}$ (denoted simply as $\boldsymbol{D}$ in the algorithm), by optimizing the *surrogate* objective function $\hat{f}_t(D)$, similar to the original one in eq. (1), for $n = t$, except for an important change: in eq. (1), each code $\boldsymbol{\alpha}_i$ is computed using the *same* dictionary $\boldsymbol{D}$, while the surrogate function keeps the codes $\boldsymbol{\alpha}_i$ from the *previous* iterations, computed using the corresponding previous dictionaries $\boldsymbol{D}^{(i)}$, i.e. *does not recompute* the codes of previously seen samples after each dictionary update. This speeds up the learning without worsening the (asymptotic) performance, since the surrogate objective converges to the original one in (1), under certain assumptions, including data stationarity [Mairal *et al.*, 2009]. Note that, in order to prevent the dictionary entries from growing arbitrarily large, [Mairal *et al.*, 2009] impose the norm constraint, i.e. keep the columns of $\boldsymbol{D}$ within the convex set $\mathcal{C} = \{\boldsymbol{D} \in \mathbb{R}^{m\times k} \;\; s.t. \; \forall j \; \boldsymbol{d}_j^T\boldsymbol{u}_j \leq 1\}$. Then the dictionary update step computes $\boldsymbol{D}^{(t)} = \arg\min_{\boldsymbol{D}\in\mathcal{C}} \hat{f}_t(D)$, ignoring l_1-regularizer over the code which is fixed at this step, i.e. it finds $\min_{\boldsymbol{D}\in\mathcal{C}} \frac{1}{t}\sum_{i=1}^{t}\frac{1}{2}||\boldsymbol{x}_i - \boldsymbol{D}\boldsymbol{\alpha}_i||_2^2$, which is equivalent to finding

$$\min_{\boldsymbol{D}\in\mathcal{C}} \frac{1}{2}Tr(\boldsymbol{D}^T\boldsymbol{D}\boldsymbol{A}) - Tr(\boldsymbol{D}^T\boldsymbol{B}), \quad (2)$$

where $\boldsymbol{A} = \sum_{i=1}^{t}\boldsymbol{\alpha}_i\boldsymbol{\alpha}_i^T$ and $\boldsymbol{B} = \sum_{i=1}^{t}\boldsymbol{x}_i\boldsymbol{\alpha}_i^T$ are the "bookkeeping" matrices (we also call them "memories" of the model), compactly representing the input samples and encoding history. At each iteration, once the new input sample $\boldsymbol{x}_i$ is encoded, the matrices are updated as $\boldsymbol{A} \leftarrow \boldsymbol{A} + \boldsymbol{\alpha}_t\boldsymbol{\alpha}_t^T$ and $\boldsymbol{B} \leftarrow \boldsymbol{B} + \boldsymbol{x}_t\boldsymbol{\alpha}_t^T$ (see the step 11 of Alg. 1). In [Mairal *et al.*, 2009], a *block coordinate descent* is used to optimize the convex objective in eq. 2; it iterates over the dictionary elements in a fixed sequence, until convergence, optimizing each while keeping the others fixed as shown in eq. (3) (steps 14 and 17 in Alg. 1, except that, in our approach, $\boldsymbol{u}_j$ is transformed into $\boldsymbol{w}_j$ to impose an additional regularizer before computing step 17) [3].

$$\boldsymbol{u}_j \leftarrow \frac{\boldsymbol{b}_j - \sum_{k\neq j}\boldsymbol{d}_k a_{jk}}{a_{jj}}; \quad \boldsymbol{d}_j \leftarrow \frac{\boldsymbol{u}_j}{\max(1, ||\boldsymbol{u}_j||_2)} \quad (3)$$

[3] Note that when the off-diagonal entries $\boldsymbol{a}_{jk}$ in $\boldsymbol{A}$ are as large as the diagonal a_{jj}, the dictionary elements get "tied" to each other, playing complementary roles in the dictionary, thereby constraining the updates of each other - an insight used later to explain empirical performance of the proposed approach.

It this work, we will also impose sparsity on dictionary elements (step 15 in Alg. 1), i.e. replace the objective in eq. (2) with

$$\min_{\boldsymbol{D}\in\mathcal{C}} \frac{1}{t}\sum_{i=1}^{t}\frac{1}{2}||\boldsymbol{x}_i - \boldsymbol{D}\boldsymbol{\alpha}_i||_2^2 + \sum_j \lambda_j||\boldsymbol{d}_j||_1. \qquad (4)$$

From now on, ODL will refer to the above extended version of the fixed-size method of [Mairal *et al.*, 2009] wherever we have sparsity in dictionary elements.

3 Neurogenic Online Dictionary Learning

Our objective is to extend the state-of-art online dictionary learning, designed for stationary input distributions, to a more adaptive framework capable of handling nonstationary data effectively, and learning to represent new types of data without forgetting how to represent the old ones. Towards this end, we propose a novel algorithm, called Neurogenetic Online Dictionary Learning (see Alg. 1), which can flexibly extend and reduce a dictionary in response to the changes in an input distribution, and possibly to the inherent representation complexity of the data. The main changes, as compared to the non-adaptive, fixed-dictionary-size algorithm of [Mairal *et al.*, 2009], are highlighted in Alg. 1; the two parts involve (1) neurogenesis, i.e. the addition of dictionary elements (hidden units, or "neurons") and (2) the death of old and/or new elements which are "less useful" than other elements for the task of data reconstruction.

At each iteration in Alg. 1, the next batch of samples is received and the corresponding codes, in the dictionary, are computed; next, we add k_n new dictionary elements sampled at random from $\mathbb{R}^m$ (i.e., k_n random linear projections of the input sample). The choice of the parameter k_n is important; one approach is to tune it (e.g., by cross-validation), while another is to adjust it dynamically, based on the dictionary performance: e.g., if the environment is changing, the old dictionary may not be able to represent the new input well, leading to decline in the representation accuracy, which triggers neurogenesis. Herein, we use as the performance measure the Pearson correlation between a new sample and its representation in the current dictionary $r(\boldsymbol{x}_t, \boldsymbol{D}^{(t-1)}\boldsymbol{\alpha}_t)$, i.e. denoted as $p_c(\boldsymbol{x}_t, \boldsymbol{D}^{(t-1)}, \boldsymbol{\alpha}_t)$ (for a batch of data, the average over $p_c(.)$ is taken). If it drops below a certain pre-specified threshold γ (where $0 \ll \gamma \leq 1$), the neurogenesis is triggered (the step 5). The number k_n of new dictionary elements is proportional to the error $1 - p_c(\cdot)$, so that worse performance will trigger more neurogenesis, and vice versa; the maximum number of new elements is bounded by c_k (the step 6). We refer to this approach as *conditional neurogenesis* as it involves the *conditional birth* of new elements. Next, k_n random elements are generated and added to the current dictionary (the step 7), and the memory matrices $\boldsymbol{A}, \boldsymbol{B}$ are updated, respectively, to account for larger dictionary (the step 8). Finally, the sparse code is recomputed for $\boldsymbol{x}_t$ (or, all the samples in the current batch) with respect to the extended dictionary (the step 9).

The next step is the dictionary update, which uses block-coordinate descent, with the following objective function:

$$\min_{\boldsymbol{D}\in\mathcal{C}} \frac{1}{t}\sum_{i=1}^{t}\frac{1}{2}||\boldsymbol{x}_i - \boldsymbol{D}\boldsymbol{\alpha}_i||_2^2 + \lambda_g\sum_j||\boldsymbol{d}_j||_2 + \sum_j \lambda_j||\boldsymbol{d}_j||_1. \qquad (5)$$

Algorithm 1 Neurogenetic Online Dictionary Learning (NODL)

```
Require: Data stream $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n \in \mathbb{R}^m$; initial dictionary $\boldsymbol{D} \in \mathbb{R}^{m\times k}$; conditional neurogenesis threshold, $\gamma$; max number of new elements added per data batch, $c_k$; group sparsity regularization parameter, $\lambda_g$; number of non-zeros in a dictionary element, $\beta_d$; number of non-zeros in a code, $\beta_c$.
 1: Initialize: $\boldsymbol{A} \leftarrow 0, \boldsymbol{B} \leftarrow 0$  %  reset the ``memory''
    % single data in a batch, for simpler exposition
 2: for $t = 1$ to $n$ do
 3:     Input $\boldsymbol{x}_t$ % representing the $t_{th}$ batch of data
        % Sparse coding of data:
 4:     $\boldsymbol{\alpha}_t = \arg_{\boldsymbol{\alpha}\in\mathbb{R}^k} \min \frac{1}{2}||\boldsymbol{x}_t - \boldsymbol{D}\boldsymbol{\alpha}||_2^2 + \lambda_c||\boldsymbol{\alpha}||_1$ % $\lambda_c$ tuned to have $\beta_c$ non-zeros in $\alpha_t$
        % Conditional neurogenesis:  if accuracy below threshold, add more elements
 5:     if $p_c(\boldsymbol{x}_t, \boldsymbol{D}, \boldsymbol{\alpha}_t) \leq \gamma$ then
 6:         $k_n = (1 - p_c(\boldsymbol{x}_t, \boldsymbol{D}, \boldsymbol{\alpha}_t))c_k$ %  the count of the births
 7:         $\boldsymbol{D}_n \leftarrow initializeRand(k_n)$,
            $\boldsymbol{D} \leftarrow [\boldsymbol{D} \quad \boldsymbol{D}_n]$
 8:         $\boldsymbol{A} \leftarrow \begin{bmatrix}\boldsymbol{A} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix}$, $\boldsymbol{B} \leftarrow [\boldsymbol{B} \quad \boldsymbol{0}], k \leftarrow k + k_n$
            %  Repeat sparse coding, with the new elements
 9:         $\boldsymbol{\alpha}_t = \arg_{\boldsymbol{\alpha}\in\mathbb{R}^k} \min \frac{1}{2}||\boldsymbol{x}_t - \boldsymbol{D}\boldsymbol{\alpha}||_2^2 + \lambda_c||\boldsymbol{\alpha}||_1$
10:     end if % End of neurogenesis
        %  ``Memory'' update:
11:     $\boldsymbol{A} \leftarrow \boldsymbol{A} + \boldsymbol{\alpha}_t\boldsymbol{\alpha}_t^T$, $\boldsymbol{B} \leftarrow \boldsymbol{B} + \boldsymbol{x}_t\boldsymbol{\alpha}_t^T$
        % Dictionary update by block-coordinate descent
12:     repeat
13:         for $j = 1$ to $k$ do
14:             $\boldsymbol{u}_j \leftarrow \frac{\boldsymbol{b}_j - \sum_{k\neq j}\boldsymbol{d}_k a_{jk}}{a_{jj}}$
                % Sparsifying elements (optional):
15:             $\boldsymbol{v}_j \leftarrow Prox_{\lambda_j||\cdot||_1}(\boldsymbol{u}_j) = sgn(\boldsymbol{u}_j)(|\boldsymbol{u}_j| - \lambda_j)_+$, % $\lambda_j$ tuned to get $\beta_d$ non-zeros in $\boldsymbol{v}_j$
                % Killing useless elements with $l_1/l_2$ group sparsity
16:             $\boldsymbol{w}_j \leftarrow \boldsymbol{v}_j\left(1 - \frac{\lambda_g}{||\boldsymbol{v}_j||_2}\right)_+$
17:             $\boldsymbol{d}_j \leftarrow \frac{\boldsymbol{w}_j}{\max(1, ||\boldsymbol{w}_j||_2)}$
18:         end for
19:     until convergence
20: end for
21: return $\boldsymbol{D}$
```

The first term is the standard reconstruction error, as before. The second term, l_1/l_2-regularization, promotes group sparsity over the dictionary entries, where each group corresponds to a column, i.e. a dictionary element. The group-sparsity [Yuan and Lin, 2006] regularizer causes some columns in $\boldsymbol{D}$ to be set to zero (i.e. the columns less useful for accurate data representation), thus effectively eliminating the corresponding dictionary elements from the dictionary ("killing" the corresponding hidden units). As mentioned previously, [Bengio *et al.*, 2009] used the l_1/l_2-regularizer in dictionary learning, though not in online setting, and without neurogenesis.

Finally, the third term imposes l_1-regularization on dictionary elements thus promoting sparse dictionary, besides the sparse coding. Introducing sparsity in dictionary elements, corresponding to the sparse connectivity of hidden units in the neural net representation of a dictionary, is motivated by both their biological plausibility (neuronal connectivity tends to be rather sparse in multiple brain networks), and by the computational advantages this extra regularization can provide, as we observe later in experiments section (Sec. 4).

As in the original algorithm of [Mairal *et al.*, 2009], the above objective is optimized by the block-coordinate descent, where each block of variables corresponds to a dictionary element, i.e., a column in $\boldsymbol{D}$; the loop in steps 12-19 of the Alg. 1 iterates until convergence, defined by the magnitude of change between the two successive versions of the dictionary falling below some threshold. For each column update, the first and the last steps (the steps 14 and 17) are the same as in the original method of [Mairal *et al.*, 2009], while the two intermediate steps (the steps 15 and 16) are implementing additional regularization. Both steps 15 and 16 (sparsity and group sparsity regularization) are implemented using the standard proximal operators as described in [Jenatton *et al.*, 2011]. Note that we actually use as input the desired number of non-zeros, and determine the corresponding sparsity parameter λ_c and λ_j using a binary search procedure. Overall, *the key features of our algorithm is the interplay of both the (conditional) birth and (group-sparsity) death of dictionary elements in an online setting.* [4]

4 Experiments

Our empirical evaluation compares the proposed NODL approach with the standard ODL. We also investigate separately the effects of adding or deleting dictionary elements, by evaluating the following restricted versions of our method: NODL+ involves only addition but no deletion (equivalent to NODL with no group-sparsity, i.e. $\lambda_g = 0$), and NODL- which, vice versa, involves deletion only but no addition (equivalent to NODL with the number of new elements $c_k = 0$). We use a simple non-stationary setting, where a sequence of training samples from one environment is followed by another sequence from a different environment.

4.1 Real-life Images

Our first domain includes the images of Oxford buildings, i.e. urban environment [5], while the second uses a combination of images from Flowers [6] and Animals [7] image databases (natural environment). We converted the original color images into black&white format and compressed them to smaller sizes, 32x32 and 100x100; also, here we use as the input samples full images rather than image patches.

Parameter settings. We selected 5700 images for training and another 5700 for testing; each subset contained 1900 images of each type (i.e., Oxford, Flowers, Animals). In the training phase, the algorithms receive a sequence of 1900 samples from the first domain (Oxford), and then a sequence of 3800 samples from the second domain (1900 Flowers and 1900 Animals, permuted randomly); the batch size is 200 images ([Mairal *et al.*, 2009] used a batch of size 256, though image patches rather than full images). We use Pearson correlation threshold $\gamma = 0.9$, group sparsity parameter $\lambda_g = 0.03$ and $\lambda_g = 0.07$, for 32x32 and 100x100 images, respectively; $c_k = 50$ is the upper bound on the number of new dictionary elements at each iteration. We observed that, overall, the results were quite robust to the changes in parameter values.

Evaluation. Once the training phase is completed, the resulting dictionary is evaluated on test images from both the first (urban) and the second (natural) domains; for the second domain, separate evaluation is performed for flowers and animals. First, we evaluate the *reconstruction* ability of the resulting dictionary $\boldsymbol{D}$, comparing the actual inputs $\boldsymbol{x}$ versus approximations $\boldsymbol{x}^* = \boldsymbol{D}\boldsymbol{\alpha}$, using the mean square error (MSE), Spearman correlation, and Pearson correlation; we only plot the latter since all metrics showed consistent results. Next, we used the codes as new features and evaluated the corresponding image classification accuracy for flowers vs animals. *The evaluation results differed dramatically for sparse vs dense dictionaries, as discussed below.*

In Fig. 1(a), 1(b) and 1(c), we present the results for *sparse dictionaries*, where each column (an element in the dictionary) has 5 nonzeros out of the 1024 dimensions; the codes are relatively dense, with at most 200 nonzeros out of k (the number of dictionary elements), and k ranging from 5 to 1000 (i.e. the codes are not sparse for $k \leq 200$). In Fig. 1(a), we compare the dictionary size for different methods: the final dictionary size after completing the training phase (y-axis) is plotted against the initial dictionary size (x-axis). Obviously, the baseline (fixed-size) ODL method (magenta plot) keeps the size constant, deletion-only NODL- approach reduces the initial size (red plot), and addition-only NODL+ increases the size (light-blue plot). However, the interplay between the addition and deletion in our NODL method (dark-blue) produces a more interesting behavior: it tends to adjust the representation complexity towards certain balanced range, i.e. very small initial dictionaries are expanded, while very large ones are, vice versa, reduced.

NODL advantages. Our main results demonstrating the advantages of the proposed NODL method are shown next in Fig. 1(b) and Fig. 1(c), for the "old" (Oxford) and "new" (Flowers) environment (domain), respectively. (Very similar result are obtained for the Animals dataset). The x-axis shows the final dictionary size, and the y-axis is the reconstruction accuracy achieved by the trained dictionary on the test samples, measured by Pearson correlation between the actual and reconstructed data. NODL clearly outperforms the fixed-size ODL, especially on smaller dictionary sizes; remarkably, this happens on both domains, i.e. besides improved adaptation to the new data, NODL is also better at preserving the "memories" of the old data, *without increasing the representation complexity, i.e. for the same dictionary size*.

Furthermore, the addition/deletion trade-off is indeed important, since the performance of deletion-only NODL- is inferior to NODL, while the addition-only, or NODL+, method matches NODL accuracy but at the cost of unnecessarily large dictionary. NODL achieves the best trade-off here, attaining superior performance while keeping the dictionary size under control, in a narrower range (400 to 650 elements).

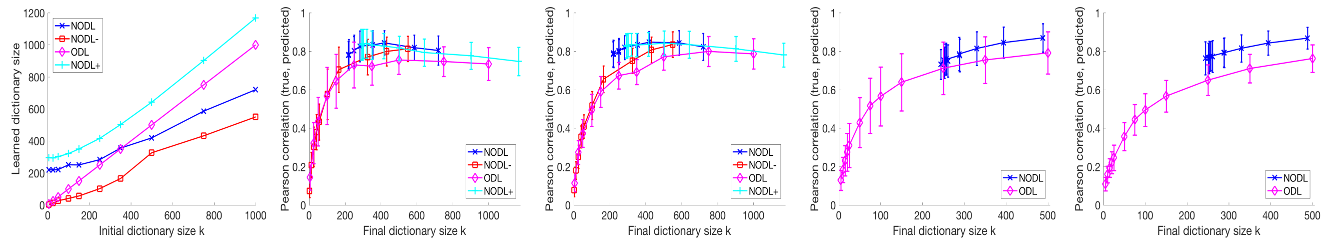
Larger-scale setting. We now focus on comparing the two main methods, the baseline ODL and the proposed NODL. The advantages of our approach become even more pronounced on larger input sizes, 100x100 images, in similar sparse-dictionary, dense-code settings (keeping the dictionary

[4]Note that the computational cost analysis, as well as the convergence analysis for our proposed algorithm can be more complicated than in the standard dictionary learning setting, since the size of the dictionary continues to change, in a non-stationary environment.

[5] www.robots.ox.ac.uk/ vgg/data/oxbuildings/

[6] www.robots.ox.ac.uk/ vgg/data/flowers/102/

[7] www.robots.ox.ac.uk/ vgg/data/pets/



(a) Learned Dictionary Size (b) 1st domain (Oxford) (c) 2nd domain (Flowers) (d) 1st domain (Oxford) (e) 2nd domain (Flowers)

Figure 1: Reconstruction accuracy (Pearson correlation) of NODL and ODL with sparse dictionaries learned on images

elements at the same sparsity rate, 50 nonzeros out of 10k dimensions, and just use completely non-sparse codes). In Fig. 1(d) and Fig. 1(e), we see that NODL considerably outperforms ODL on both the first (Oxford) and the part of the second domain (Flowers); the results for Animals are similar.

We also performed a variety of experiments, not included due to the space restrictions, which demonstrated robustness of our results to the parameter perturbations across a wide range of γ, λ_g, c_k, β_c, β_d, and batch-size parameters, as well as to various orders of the input datasets. Moreover, we explored a version of ODL mentioned in [Mairal *et al.*, 2009] where occasional zero-column ("dead" elements) are reinitialized, but did not observe significant differences in ODL performance, since without the explicit group-sparse regularization of NODL, the number of "dead" elements in ODL was very small.

Dense vs. sparse dictionary elements. When experimenting with more traditional dense dictionaries, commonly used in sparse coding literature, we observed that both adaptive NODL and non-adaptive ODL approaches behaved very similarly; being adaptive never hurt the performance, but also did not result into significant improvements observed in case of sparse dictionaries. Furthermore, we also evaluated both types of dictionaries, sparse vs dense, in both adaptive and non-adaptive approaches, for the purpose of *classification* rather than just *reconstruction* of the inputs. Namely, we use the codes (i.e., feature vectors) computed on the test data from the second domain (Animals and Flowers) and evaluate multiple classifiers learned on those features in order to discriminate between the two classes. We find that the overall classification errors, for both ODL and NODL, are much higher in dense dictionary setting (0.30, 0.40 error with dense and sparse codes respectively) than in the sparse-dictionary setting (0.24 error with dense codes). This again demonstrates the advantages of sparse dictionaries over the dense ones.

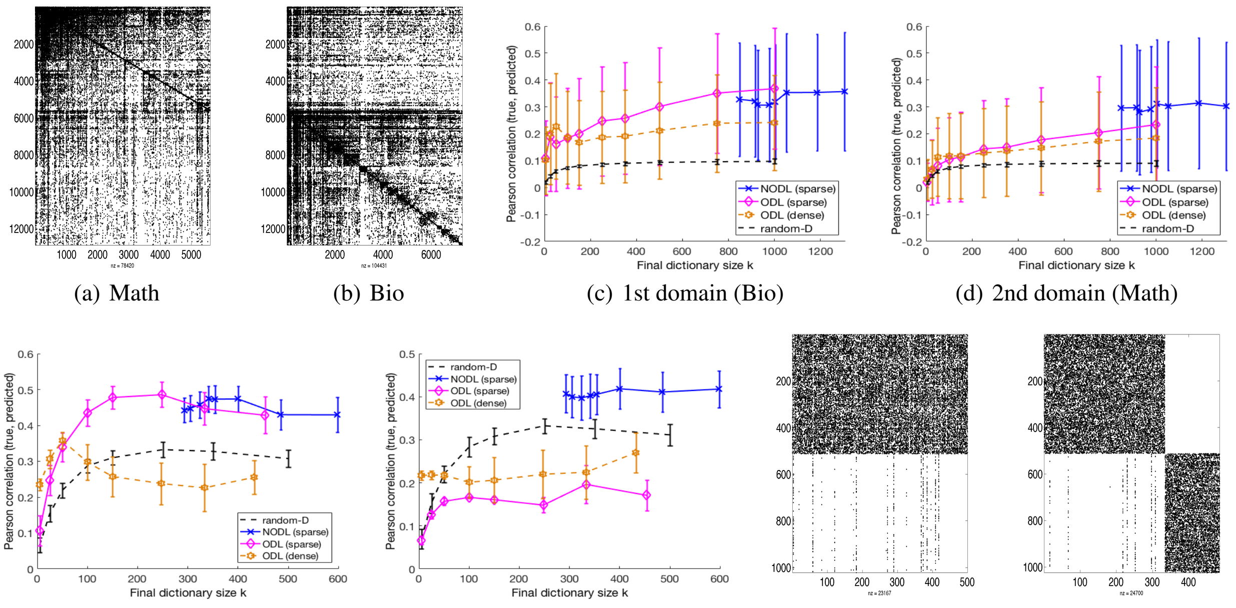
In summary, *(1) our adaptive NODL approach significantly outperforms nonadaptive ODL, on both the new data (adaptation) and the old ones (memory), when using sparse dictionaries; (2) the results are robust to parameter and data set order variations; (3) for dense dictionaries, adaptive approach performs as good as the nonadaptive baseline.*

4.2 Sparse Orthogonal Inputs: Natural Language Processing and Synthetic Data

Our next question was: which specific *data properties* (rather than *method properties*) make neurogenetic adaptation most beneficial ? We noticed that the standard ODL approach has difficulties adapting to a new domain when: (1) the data in both domains are sparse and (2) across the domains, the supports (subsets of non-zero coordinates) are almost non-overlapping, which makes the old and new datasets nearly orthogonal. This phenomenon was observed in a Natural Language Processing (NLP) task presented below, further investigated on simulated data, and formalized later in Lemma 1.

Sparse Natural Language Processing Problem. We consider a very sparse word co-occurrence matrix (avg. 14 nnz per column) using the text from two different domains, biology and mathematics, with the total vocabulary size of approximately 12,883 words. The full matrix was split in two with math terms correspond to the first block of columns (see Fig. 2(a)) and the biology terms correspond to the second one (see Fig. 2(b)). We use the sparse columns (or rows) in the matrix, indexed by the vocabulary words, as our input data to learn the dictionary of sparse elements (25 nonzeros) with sparse codes (or word embeddings with 38 nonzeros) [Yogatama *et al.*, 2015]. Herein, we evaluate our NODL method (i.e. NODL (sparse) in the plots) versus baseline ODL dictionary learning approach (i.e. ODL (sparse)) in the settings where the biology domain is processed first and then the mathematics domain (2750 samples for training and test from each domain). The evaluation results are shown in Fig. 2(c) and 2(d). For the first domain (biology), both methods perform very similarly (i.e., remember the old data equally well), while for the second, more recent domain, our NODL algorithm is clearly outperforming its competitor. Also, note that dense dictionaries do not work well on these sparse data – see inferior performance of both random dense dictionaries (random-D) and the dense dictionaries learned with ODL (i.e. ODL (dense)).

Synthetic Sparse Data. The word co-occurrence matrix from different domains such as mathematics and biology tends to have approximately block-diagonal structure. So, herein, we studied the simulated sparse dataset wherein the data matrix is purely block-diagonal with each column of 1024 dimension with 50 nonzeros (100 samples from each of the two domains, for training and testing). In Fig. 2(e) and 2(f), we see reconstruction accuracy, for the first and second domain data. For the first domain, the baseline ODL method (i.e. ODL (sparse) in the plots) and our NODL (i.e. NODL (sparse)) perform equally well. However, for the second domain, the nonadaptive ODL performs much worse than adaptive NODL, being unable to adapt to the new data coming from a subspace orthogonal (due to nonoverlapping support) to the first dataset.



(a) Math (b) Bio (c) 1st domain (Bio) (d) 2nd domain (Math)

(e) Synthetic: 1st domain (f) Synthetic: 2nd domain (g) D- ODL (h) D- NODL

Figure 2: Reconstruction accuracy (Pearson correlation) for the sparse NLP as well as synthetic data.

The dictionaries learned by each method are shown in Fig. 2(g) and Fig. 2(h): nonadaptive ODL maintains the sparsity structure learned from the first domain, and cannot adapt when domains change, while NODL learns elements representing the sparsity structure of the second domain, due to its ability to add new randomly initialized dense dictionary elements, which adapt to the new sparsity pattern.

5 Theoretical Analysis and Discussion

Sparse data modeling with sparse dictionaries. Our empirical observations for the sparse data are formally supported by the Lemma 1 below which proves that the baseline ODL method cannot adapt to a new domain if there is no overlap with the old domain's support [8].

Lemma 1. *Let $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{t-1} \in \mathbb{R}^m$ be a set of samples from the first domain, with non-zeros (support) in the set of dimensions $P \subset M = \{1, \cdots, m\}$, and let $\boldsymbol{x}_t, \boldsymbol{x}_{t+1}, \cdots, \boldsymbol{x}_n \in \mathbb{R}^m$ be a set of samples from the second domain, with non-zeros (support) in dimensions $Q \subset M$, such that $P \cap Q = \emptyset$, $|P| = |Q| = l$. Let us denote as $\boldsymbol{d}_1, \boldsymbol{d}_2, \cdots, \boldsymbol{d}_k \in \mathbb{R}^m$ dictionary elements learned by ODL algorithm, with the sparsity constraint of at most l nonzeros in each element (l corresponds to β_d in Alg. 1), on the data from the first domain, $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{t-1}$. Then (1) those elements have non-zero support in P only, and (2) after learning from the second domain data, the support (nonzero dimensions) of the corresponding updated dictionary elements will remain in P.*

Proof Sketch. When processing the data from the first domain, at the first iteration, a sample $\boldsymbol{x}_1$ is received, its code $\boldsymbol{\alpha}_1$ is computed, and the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are updated, as shown in Alg. 1 (non-highlighted part); next, the dictionary update step is performed, which optimizes

$$\boldsymbol{D}^{(1)} = \arg\min_{\boldsymbol{D}\in\mathcal{C}} \frac{1}{2} Tr(\boldsymbol{D}^T\boldsymbol{D}\boldsymbol{A}) - Tr(\boldsymbol{D}^T\boldsymbol{B}) + \sum_j \lambda_j ||\boldsymbol{d}_j||_1.$$

Since the support of $\boldsymbol{x}_1$ is limited to P, we can show that optimal dictionary $\boldsymbol{D}^*$ must also have all columns/elements with support in P. Indeed, assuming the contrary, let $\mathbf{d_j}(i) \neq 0$ for some dictionary element/column j, where $i \notin P$. But then it is easy to see that setting $\mathbf{d_j}(i)$ to zero reduces the sum-squared error and the l_1-norm in (1), yielding another dictionary that achieves a lower overall objective; this contradicts our assumption that $\mathbf{D}^*$ was optimal. Thus, a dictionary obtained after the dictionary update step must produce a dictionary where all columns have their support in P. By induction, this statement will also be true for the dictionary obtained after processing all samples from the first domain. Next, we start receiving the samples from the second domain which belongs to a different subspace, spanning the dimensions within the support set Q not intersecting with P. Thus, using the current dictionary, the encoding $\boldsymbol{\alpha}_t$ of first sample $\boldsymbol{x}_t$ from the second domain (i.e. the solution of the LASSO problem in step 4 of the Alg. 1) will be a zero vector. Therefore, the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ remains unchanged during the update in step 11, and thus the support of each $\boldsymbol{b}_j$, and, consequently, $\boldsymbol{u}_j$ and the updated dictionary elements $\boldsymbol{d}_j$ will remain in P. By induction, every dictionary update in response to a new sample from the second domain will preserve the support of the dictionary elements, and thus the final dictionary elements will also have their support only in P. □

Non-sparse data modeling with sparse dictionaries. When learning sparse dictionaries on non-sparse data such as natural images, we observed that many dictionary elements have non-overlapping supports with respect to each other.

[8]The minor adaptation, i.e., a few nonzeros, observed in our results in Fig. 2(g) occurs only due to implementation details involving normalization of sparse dictionary elements when computing codes in the dictionary – the normalization introduces non-zeros of small magnitude in all dimensions.

The non-overlapping support of dictionary elements results into a specific structure of the matrix $\boldsymbol{A}$. For ODL approach, the resulting matrix $\boldsymbol{A}$ includes many off-diagonal nonzero elements of large absolute values (along with high values on the diagonal). Note that in the dictionary update expression $\boldsymbol{u}_j \leftarrow \frac{\boldsymbol{b}_j - \sum_{k \neq j} \boldsymbol{d}_k a_{jk}}{a_{jj}}$ in (3), when the values a_{jk}/a_{jj} are large for multiple k, the j_{th} dictionary element becomes tightly coupled with other dictionary elements, which reduces its adaptability to new, non-stationary data. In our algorithm, the values a_{jk}/a_{jj} remain high if both elements j and k have similar "age"; however, those values are much lower if one of the elements is introduced much more recently than the other one. This allows for an adaptation to a new domain without forgetting the old one.

6 Conclusions

We proposed a novel algorithm, Neurogenetic Online Dictionary Learning (NODL), for learning representations in non-stationary environments. Our online algorithm builds a dictionary while also adapting the dictionary structure (the number of elements/hidden units and their connectivity; connectivity adaptation is possibly due to dictionary sparsity) via continuous birth (addition) and death (deletion) of dictionary elements, inspired by the adult neurogenesis phenomenon.

Our extensive empirical evaluation on both real world and synthetic data demonstrated that the interplay between the birth and death of dictionary elements learns a more adaptive, better-performing dictionary, in non-stationary environments as compared to both of its counterparts, such as the fixed-size online method of [Mairal *et al.*, 2009] (no addition and no deletion), and the online version of the group-sparse coding method by [Bengio *et al.*, 2009] (deletion only). Furthermore, we evaluated, both empirically and theoretically, some conditions on the method's parameters and on the data which yield superior performance of the proposed method over the baseline. Overall, neurogenetic dictionary learning performs at least as good as, and often much better than its competitors. Future work includes extending our approach to deep and nonlinear autoencoders, as well as to supervised setting.

Acknowledgements

It is our pleasure to acknowledge discussions with Djallel Bouneffouf, Amit Dhurandhar, Changhoan Kim, Tim Klinger, Matthew Riemer, Mattia Rigotti, Vidyashankar Sivakumar, Gerald Tesauro, and other colleagues at the IBM Thomas J. Watson Research Center. We are also grateful to the anonymous reviewers for their valuable feedback.

References

[Aharon *et al.*, 2006] Michal Aharon, Michael Elad, and Alfred Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *Signal Processing, IEEE Transactions on*, 2006.

[Ba and Caruana, 2014] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, 2014.

[Bengio *et al.*, 2009] Samy Bengio, Fernando Pereira, Yoram Singer, and Dennis Strelow. Group sparse coding. In *Advances in Neural Information Processing Systems 22*. 2009.

[Bucilu *et al.*, 2006] Cristian Bucilu, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006.

[Draelos *et al.*, 2016] Timothy J Draelos, Nadine E Miner, Christopher C Lamb, Craig M Vineyard, Kristofor D Carlson, Conrad D James, and James B Aimone. Neurogenesis deep learning. *arXiv preprint arXiv:1612.03770*, 2016.

[Fahlman and Lebiere, 1989] Scott E Fahlman and Christian Lebiere. The cascade-correlation learning architecture. 1989.

[Hinton *et al.*, 2015] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[Jenatton *et al.*, 2011] Rodolphe Jenatton, Julien Mairal, Guillaume Obozinski, and Francis Bach. Proximal methods for hierarchical sparse coding. *Journal of Machine Learning Research*, 2011.

[Kempermann, 2006] Gerd Kempermann. *Adult neurogenesis: stem cells and neuronal development in the adult brain*. 2006.

[Kreutz-Delgado *et al.*, 2003] Kenneth Kreutz-Delgado, Joseph F Murray, Bhaskar D Rao, Kjersti Engan, Te-Won Lee, and Terrence J Sejnowski. Dictionary learning algorithms for sparse representation. *Neural computation*, 2003.

[Lee *et al.*, 2006] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y Ng. Efficient sparse coding algorithms. In *Advances in neural information processing systems*, 2006.

[Mairal *et al.*, 2009] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*, 2009.

[Mariet and Sra, 2015] Zelda Mariet and Suvrit Sra. Diversity networks. *arXiv preprint arXiv:1511.05077*, 2015.

[Olshausen and Field, 1997] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 1997.

[Ramirez and Sapiro, 2012] Ignacio Ramirez and Guillermo Sapiro. An mdl framework for sparse coding and dictionary learning. *IEEE Transactions on Signal Processing*, 60(6):2913–2927, 2012.

[Rusu *et al.*, 2016] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[Srivastava *et al.*, 2014] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 2014.

[Stuchlik, 2014] Ales Stuchlik. Dynamic learning and memory, synaptic plasticity and neurogenesis: an update. *Frontiers in behavioral neuroscience*, 2014.

[Tibshirani, 1996] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 1996.

[Yogatama *et al.*, 2015] Dani Yogatama, Manaal Faruqui, Chris Dyer, and Noah A Smith. Learning word representations with hierarchical sparse coding. In *Proc. of ICML*, 2015.

[Yuan and Lin, 2006] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2006.