Deep Learning using Robust Interdependent Codes

Hugo Larochelle, Dumitru Erhan and Pascal Vincent
Dept. IRO, Université de Montréal
P.O. Box 6128, Succ. Centre-Ville, Montreal, H3C 3J7, Qc, Canada
{larocheh,erhandum,vincentp}@iro.umontreal.ca

Abstract

We investigate a simple yet effective method to introduce inhibitory and excitatory interactions between units in the layers of a deep neural network classifier. The method is based on the greedy layer-wise procedure of deep learning algorithms and extends the denoising autoencoder (Vincent et al., 2008) by adding asymmetric lateral connections between its hidden coding units, in a manner that is much simpler and computationally more efficient than previously proposed approaches. We present experiments on two character recognition problems which show for the first time that lateral connections can significantly improve the classification performance of deep networks.

1 INTRODUCTION

Recently, an increasing amount of work in the machine learning literature has been addressing the difficult issue of training neural networks with many layers of hidden neurons. The motivation behind introducing several intermediate layers between the input of a neural network and its output is that hard AI-related learning problems, such as those addressing vision and language, require discovering complex high-level abstractions, which can be represented more efficiently by models with a deep architecture (Bengio & LeCun, 2007). While deep networks are not novel, the discovery of techniques able to train them successfully and deliver superior generalization performance is recent. This new class of algorithms, *deep learning* algorithms, have proved successful at leveraging the power of deep networks in several contexts such as image classification (Larochelle et al., 2007), object recognition (Ranzato et al., 2007), regression (Salakhutdinov & Hinton, 2008), dimensionality reduction (Hinton & Salakhutdinov, 2006) and document retrieval (Salakhutdinov & Hinton, 2007).

Current deep learning algorithms are based on a greedy layer-wise training procedure (Hinton et al., 2006; Bengio et al., 2007) which decouples the algorithm in two phases. The *pre-training phase* initializes a deep network with a set of *greedy modules* by training them sequentially in an unsupervised manner. Each is trained on the representation produced by the greedy module below, with the goal to discover a higher-level representation of it, so that the representations become more abstract as we move up the network. This is followed by a *fine-tuning phase* which aims at globally adjusting all the parameters of the network according to some (often supervised) criterion related to the ultimate task of interest.

Most recent research has been focusing on the development of good greedy modules, which play a decisive role in the quality of the representations learned by deep networks. A variety of greedy modules have been proposed: Restricted Boltzmann Machines (RBMs) (Hinton et al., 2006), autoassociators or autoencoders (Bengio et al., 2007), sparse autoencoders (Ranzato et al., 2008), denoising autoencoders (Vincent et al., 2008) and non-linear embedding algorithms (Weston et al., 2008). These greedy modules leverage unlabeled data to discover meaningful representations and their training objectives span a vast variety of motivations and properties of representations.

All these previous greedy modules however share one characteristic about the way they transform their input into a new representation: given an input pattern, all elements of the representation are computed independently and cannot interact in an inhibitory or

Appearing in Proceedings of the 12^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2009, Clearwater Beach, Florida, USA. Volume 5 of JMLR: W&CP 5.

excitatory fashion. However, there is a growing body of work on introducing pairwise interactions between the hidden units of models with latent representations (Garrigues & Olshausen, 2008; Hyvärinen et al., 2001; Osindero et al., 2006; Hinton et al., 2005), which show that they can be beneficial in modeling data such as patches of natural images.

In this paper, we extend the basic denoising autoencoder (Vincent et al., 2008) by introducing lateral connections between coding elements, which permit simple yet useful interactions between codes. We show experimentally that the lateral connections learn to implement inhibitory and excitatory interactions which allow discrimination between visually overlapping patterns. We also demonstrate that such a **denoising autoencoder with interdependent codes (DAIC)** outperforms the basic denoising autoencoder as well as RBMs in training deep neural network classifiers on two character recognition problems. Finally, we show that interdependent codes tend to extract a richer set of features which are less likely to be linearly predictable from each other (i.e. less correlated), leaving it to upper layers to account for the remaining non-linear dependencies between these features.

2 DENOISING AUTOENCODER

The present work builds on the **denoising autoencoder** (Vincent et al., 2008) as a greedy module for deep learning. Denoising autoencoders are motivated by the idea that a good representation $\mathbf{enc}(\mathbf{x})$ for some input vector $\mathbf{x}$ should be informative of $\mathbf{x}$ and invariant to induction of noise in the input. Given a *corrupted* version $\widetilde{\mathbf{x}}$ of the input, such a *robust* representation should make it possible to recover $\mathbf{x}$ from $\mathbf{enc}(\widetilde{\mathbf{x}})$, through a decoding function $\mathbf{dec}(\cdot)$.

A denoising autoencoder thus requires the following:

- $\mathbf{enc}(\cdot)$: an encoder function which computes a new representation for its input. This function's parameters should be adjustable given an error gradient.
- $\mathbf{dec}(\cdot)$: a decoder function which decodes a representation and gives a prediction for the original input. This function's parameters should also be adjustable.
- $p(\widetilde{\mathbf{x}}|\mathbf{x})$: a conditional distribution used to generate corrupted versions $\widetilde{\mathbf{x}}$ of an input $\mathbf{x}$.
- $C(\cdot,\cdot)$: a differentiable cost function that computes the dissimilarity between two vectors or representations.

The corruption process $p(\widetilde{\mathbf{x}}|\mathbf{x})$ used originally (Vincent et al., 2008) sets to zero (i.e. destroys all information from) a random subset of the elements of $\mathbf{x}$, corresponding to a fraction α of all elements. This means that the autoencoder must learn to compute a representation that is informative of the original input even when some of its elements are missing. This technique was inspired by the ability of humans to have an appropriate understanding of their environment even in situations where the available information is incomplete (e.g. when looking at an object that is partly occluded).

Training a denoising autoencoder is as simple as training a standard autoencoder through backpropagation, with the additional step of corrupting the input. Given a training input pattern $\mathbf{x}_t$, first we generate a noisy version $\widetilde{\mathbf{x}}_t$, compute its representation $\mathbf{enc}(\widetilde{\mathbf{x}}_t)$, compute a reconstruction $\mathbf{dec}(\mathbf{enc}(\widetilde{\mathbf{x}}_t))$ and compare it to the **original input** $\mathbf{x}_t$ using the cost function $C(\mathbf{x}_t, \mathbf{dec}(\mathbf{enc}(\widetilde{\mathbf{x}}_t))$. Then we compute the error gradient $\frac{\partial}{\partial\theta_k}C(\mathbf{x}_t, \mathbf{dec}(\mathbf{enc}(\widetilde{\mathbf{x}}_t))$ for all parameters θ_k of the encoder and decoder functions, and update all parameters using stochastic gradient descent.

We consider the same corruption process $p(\widetilde{\mathbf{x}}|\mathbf{x})$ and encoder/decoder pair as proposed originally:

$$\mathbf{enc}(\tilde{\mathbf{x}}) = \mathrm{sigm}(\mathbf{b} + \mathbf{W}\tilde{\mathbf{x}}) \quad (1)$$

$$\mathbf{dec}(\mathbf{enc}(\tilde{\mathbf{x}})) = \mathrm{sigm}(\mathbf{c} + \mathbf{W}^T\mathbf{enc}(\tilde{\mathbf{x}})) \quad (2)$$

and use the same cross-entropy cost function: $C(\mathbf{x},\mathbf{y}) = -\sum_i (x_i \log y_i + (1-x_i)\log(1-y_i))$, where the elements of $\mathbf{x}$ and $\mathbf{y}$ are assumed to be in $[0,1]$.

We wish to use denoising autoencoders to train a deep neural network classifier. In a network with l hidden layers, we compute the activity $\mathbf{h}^i(\mathbf{x})$ of the ith hidden layer given some input $\mathbf{x}$ as follows:
$\mathbf{h}^i(\mathbf{x}) = \mathrm{sigm}(\mathbf{b}^i + \mathbf{W}^i\mathbf{h}^{i-1}(\mathbf{x}))\ \forall i \in \{1,\ldots,l\}$, with $\mathbf{h}^0(\mathbf{x}) = \mathbf{x}$.
Class assignments probabilities are computed at the output layer as follows:
$\mathbf{o}(\mathbf{x}) = \mathrm{softmax}(\mathbf{b}^{l+1} + \mathbf{W}^{l+1}\mathbf{h}^l(\mathbf{x}))$ with

$$\mathrm{softmax}(\mathbf{a}) = \left(\frac{\exp(a_i)}{\sum_k \exp(a_k)}\right)_{i=1}^{m}$$

To use denoising autoencoders for deep learning, we follow the general greedy layer-wise procedure (Hinton et al., 2006; Bengio et al., 2007) and pre-train each layer of a deep neural network as a denoising autoencoder. The procedure is depicted in Fig. 1. During the greedy pre-training phase, when training the ith layer, each input is mapped to its hidden representation $\mathbf{h}^{i-1}(\mathbf{x})$ and is used as a training sample to a denoising autoencoder with biases $\mathbf{b} = \mathbf{b}^i$, $\mathbf{c} = \mathbf{b}^{i-1}$ and weights $\mathbf{W} = \mathbf{W}^i$. Note that this requires the corruption of $\mathbf{h}^{i-1}(\mathbf{x})$ into $\widetilde{\mathbf{h}^{i-1}(\mathbf{x})}$. A layer is pre-trained

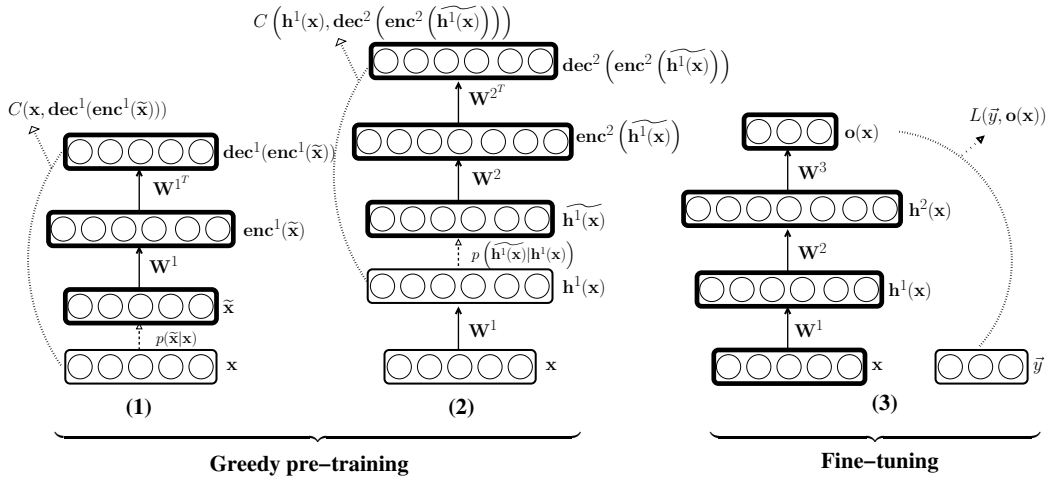

Figure 1: Illustration of the greedy layer-wise procedure for training a 2 hidden layer neural network with denoising autoencoders. To avoid clutter, biases $\mathbf{b}^i$ and $\mathbf{c}^i$ are not represented in the figures.

for a fixed number of updates, after which the new representation it learned is stored to be used as input for the next layer. Greedy pre-training then moves on to the next hidden layer. Once all layers have thus been initialized, the whole network is fine-tuned[1] by stochastic gradient descent using backpropagation and the class assignment negative log-likelihood cost $L(\vec{y}, \mathbf{o}(\mathbf{x})) = -\sum_k \vec{y}_k \log o(\mathbf{x})_k$ where $\vec{y} = (1_{k=y})_{k=1}^m$.

3 DENOISING AUTOENCODER WITH INTERDEPENDENT CODES (DA-IC)

As mentioned earlier, a denoising autoencoder is one example of a deep network greedy module among others in the literature where the elements of the hidden representations (or codes) are computed independently. By this, we mean that the activation of a hidden layer neuron is a simple direct function of its input pattern only, and is not influenced by what other neurons in its layer do. They are therefore unable to implement interactions between these codes, such as inhibitory and excitatory interactions. Lateral connections between elements of hidden representations have been used successfully to model natural images in sparse coding (Garrigues & Olshausen, 2008), ICA (Hyvärinen et al., 2001) and energy-based (Osindero et al., 2006) models.

In this work, we investigate whether such interactions can also be useful in learning a deep neural network classifier. One approach to introduce interactions between the units of a layer is to express their effect in a recursive equation (Shriki et al., 2001; Osindero & Hinton, 2008):

$$\mathbf{enc}(\widetilde{\mathbf{x}})_j = \text{sigm}\left(b_j + \sum_k W_{jk}\tilde{x}_k + \sum_{k \neq j} V_{jk}\mathbf{enc}(\widetilde{\mathbf{x}})_k\right) \tag{3}$$

where each V_{jk} induces an interaction between hidden neuron j and k, if $V_{jk} \neq 0$.

To compute an encoding, its elements are updated recursively according to Equation 3 for a number of iterations or until convergence. There are two disadvantages to this approach. First, computing the encoding becomes expensive for large layers or number of iterations. Second, optimizing this encoding through gradient descent is also expensive and hard. For these reasons, we decided to take a different approach which, while being much simpler conceptually and computationally, is able to implement the type of lateral interactions that are expected from Equation 3. We simply view the inhibitory and excitatory lateral connections as performing an extra non-linear processing step on the regular encoding, and model this step by a standard linear+sigmoid layer. Thus our approach is akin to simply adding a hidden layer to the encoding function, ensuring that all computations will be fast. The presence of simple constraints on the autoencoder, specifically the encoding/decoding functions sharing the same (transposed) weights, ensures that the role of the additional set of weights $\mathbf{V}$ can be interpreted as that of lateral connections, just like in Equation 3.

We extend the denoising autoencoder model by taking into account such lateral connections *in the encoder function only*, and propose to study their effect, and verify that they indeed behave according to what we expect from lateral connections. Introducing such richer interactions only in the encoder function can

[1] without any data corruption

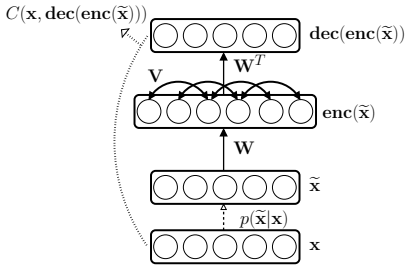

Figure 2: Illustration of the denoising autoencoder with interdependent codes.

be motivated by the view of the decoder function as a generative model for which the encoder performs a crude variational "inference" (Vincent et al., 2008). It is well known that even very simple generative models can yield a complicated posterior over the hidden representation, due to "explaining away" effects. From this perspective, the mapping from visible to hidden is often more complex than the mapping from hidden to visible. So it makes sense to have a higher capacity encoder, with the ability to learn a more complex non-linear mapping, than the decoder.

Formally, the denoising autoencoder is modified by adding asymmetric lateral connections, whose strengths are stored in a square matrix $\mathbf{V}$, as follows: given a pre-encoding of a corrupted input

$$\widehat{\mathbf{enc}(\tilde{\mathbf{x}})} = \text{sigm}(\mathbf{b} + \mathbf{W}\tilde{\mathbf{x}})$$

a final encoding is computed by using the following interaction between hidden units:

$$\mathbf{enc}(\tilde{\mathbf{x}})_j = \text{sigm}\left(d_j + V_{jj}\widehat{\mathbf{enc}(\tilde{\mathbf{x}})}_j + \sum_{k \neq j} V_{jk}\widehat{\mathbf{enc}(\tilde{\mathbf{x}})}_k\right)$$

where $V_{jj} > 0$. The same decoding function of Equation 2 is used. Though the constraint of a positive diagonal for $\mathbf{V}$ could have required special attention, using the same weight matrix $\mathbf{W}$ in the pre-encoding and decoding implicitly favors this situation, a fact that was observed to hold empirically. We also find the diagonal elements of $\mathbf{V}$ to be usually larger than other elements on the same column or row. This DA-IC architecture is illustrated in Fig. 2.

To perform deep learning, we use a greedy layer-wise procedure to pre-train all layers. In this case, each layer $\mathbf{h}^i(\mathbf{x})$ also has lateral connections $\mathbf{V}^i$ as well as the additional set of biases $\mathbf{d}^i$:

$$\mathbf{h}^i(\mathbf{x}) = \text{sigm}\left(\mathbf{d}^i + \mathbf{V}^i\text{sigm}(\mathbf{b}^i + \mathbf{W}^i\tilde{\mathbf{x}})\right)$$

Thus, for each layer, pre-training is using the previous layer representations $\mathbf{h}^{i-1}(\mathbf{x})$ as training samples to a DA-IC with biases $\mathbf{b} = \mathbf{b}^i$, $\mathbf{c} = \mathbf{b}^{i-1}$, $\mathbf{d} = \mathbf{d}^i$ and weights $\mathbf{W} = \mathbf{W}^i$, $\mathbf{V} = \mathbf{V}^i$.

4 RELATED WORK

The idea of introducing pairwise connections between elements in a hidden representation for unsupervised learning is not new. They have been used in an information maximization framework to allow overcomplete representations (Shriki et al., 2001). One important difference in our approach is that the computation of the elements of the representation requires only one quick pass through the lateral connections instead of several recursive passes; the latter would render their use in a deep network much more computationally expensive.

Lateral connections have also been used previously in models with several layers of hidden representation (Hinton et al., 2005; Osindero & Hinton, 2008). However, these connections are only used in the top-down generative process of the model and approximate bottom-up inference is done independently for each element of a hidden layer given the previous one. Interpreting the decoding function as the deterministic equivalent of a top-down generative process, the DA-IC takes the inverse perspective, where inference is complicated and generation (reconstruction) is simple.

Several models of the primary visual cortex have also integrated the concept of pairwise interactions, including sparse coding (Garrigues & Olshausen, 2008), ICA (Hyvärinen et al., 2001) and energy-based models (Osindero et al., 2006). One motivation often cited for using such connections is that they permit to better capture higher-order dependencies that would not be modeled otherwise.

Our work is aimed at leveraging the use of lateral connections in multi-layer neural networks for building competitive classifiers, in contrast to modeling the distribution of images. To our knowledge, none of the previously published approaches on introducing lateral connections in *deep networks* has studied if they did indeed yield a performance gain when used to build a classifier. The discriminative power of sparse codes (whose inference exhibit inhibitory interactions, though without explicit lateral connections) has been investigated previously (Raina et al., 2007), however they are not applicable directly to deep learning, since fine-tuning such representations according to a global task presents a technical challenge. Moreover, though the Sparse Encoding Symmetric Machine (Ranzato et al., 2008) approach to sparse coding is appropriate for deep learning, as mentioned earlier, the encoding function in that case still computes the codes independently given an input, a situation we try to improve on here. Our simple approach for introducing interdependent codes in denoising autoencoders could however easily be adapted to that framework.

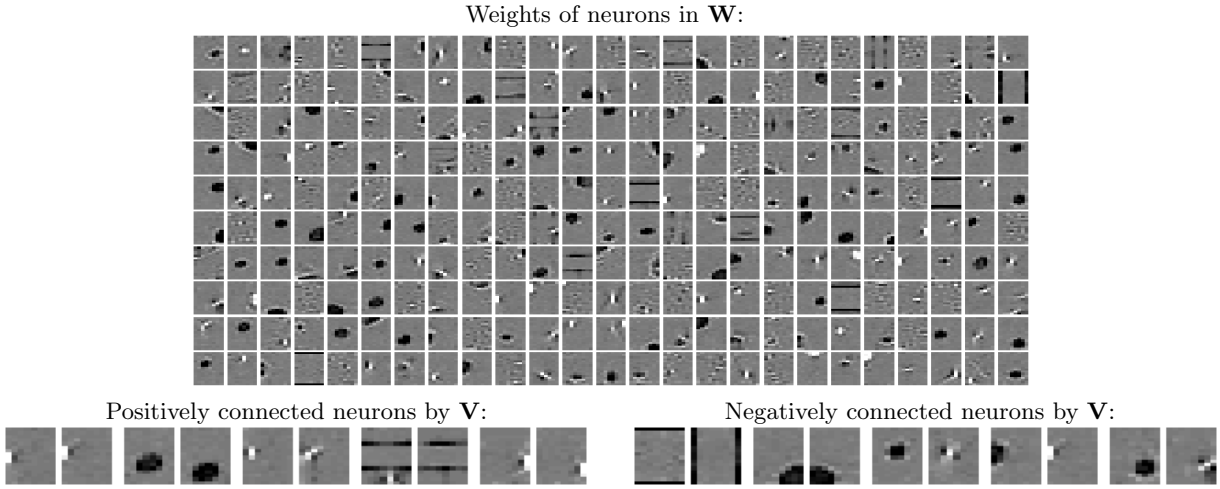

Figure 3: **Top:** visualization of the input weights of the hidden units, corresponding to the rows of **W**. A variety of filters were learned, including small pen strokes and empty background detectors. **Bottom:** visualization of a subset of excitatory and inhibitory connections in **V**. Positively connected neurons have overlapping filters, often shifted by few pixels. Negatively connected neurons detect aspects of the input which are mutually exclusive, such as empty background versus pen strokes.

Table 1: Classification performance of deep networks and gaussian kernel SVMs for two character recognition problems. The deep networks with interdependent codes statistically significantly outperform other models on both problems. We report the results on each fold of the *OCR-letters* experiment to show that the improvement in performance of interdependent codes is consistent.

Dataset	**SVM**$_{rbf}$	**DBN-3**	**SDA-3**	**SDA-6**	**SDAIC-3**
MNIST-rot	11.11	9.01	9.53	9.68	**8.07**
OCR-letters (fold 1)	9.70	9.68	9.69	10.15	**9.60**
OCR-letters (fold 2)	9.36	9.68	9.92	9.92	**9.31**
OCR-letters (fold 3)	9.94	10.07	10.29	10.32	**9.46**
OCR-letters (fold 4)	10.32	10.46	10.42	10.51	**9.92**
OCR-letters (fold 5)	10.19	10.58	9.93	10.58	**9.50**
OCR-letters (all)	9.90	10.09	10.05	10.30	**9.56**

5 EXPERIMENTS

We performed experiments on two character recognition problems where the input patterns from different classes are likely to be overlapping visually. This is a setting where lateral connections ought to be useful by using inhibitory connections to discern similar but mutually exclusive features of the input. The first problem, noted *MNIST-rot* (Larochelle et al., 2007), consists in classifying images of rotated digits[2]. The second classification dataset, noted *OCR-letters* [3] corresponds to an English character recognition problem where 16×8 binary pixel images must be classified into 26 classes, corresponding to the 26 letters of the English alphabet (see Fig. 4).

5.1 COMPARISON OF CLASSIFICATION PERFORMANCE

We evaluated the performance of the DA-IC as a greedy module for deep learning by comparing it with two other greedy modules: basic denoising autoencoders and RBMs. For each type of greedy module, deep neural network classifiers were initialized by

[2]This dataset has been regenerated since its publication and can be downloaded here: http://www.iro.umontreal.ca/∼lisa/icml2007. The set of 28×28 pixel images was generated using random rotations of digit images taken from the MNIST dataset, and was divided into training, validation and test splits of 10000, 2000 and 50000 examples each.

[3]This dataset is publicly available at http://ai.stanford.edu/∼btaskar/ocr/. For our experiments, we took the original dataset and generated 5 folds with mutually exclusive test sets of 10000 examples each.

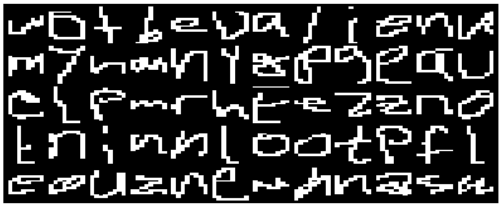

Figure 4: Input samples from the *OCR-letters* dataset of binary character images.

stacking three such greedy modules before fine-tuning the whole network by stochastic gradient descent[4]. The deep networks initialized with DA-ICs had 1000 hidden units in each layer. For fairness, since RBMs and basic denoising autoencoders have fewer parameters (hence less capacity) for the same size of hidden layer, we also considered deep networks with larger layers of up to 2000 hidden units in model selection. We chose networks with the same number of hidden units at each layer, as we found this topology to work well. Another fair comparison with a network with similar number of parameters, is to stack **6** layers of either RBMs or denoising autoencoders: both achieved about the same performance, so we report results on denoising autoencoders only. We denote by **DBN-**l, **SDA-**l and **SDAIC-**l deep networks initialized by stacking l modules of RBMs, denoising autoencoders, and DA-IC, respectively. As a general baseline, we also report the performance of a kernel SVM with Gaussian kernel (noted $\mathbf{SVM}_{rbf}$), which often achieves state-of-the-art performance.

The results, reported in Table 1, confirm that the **interdependent codes are able to improve the discriminative performance of a deep network classifier.** The addition of lateral connections also enables deep networks to outperform an SVM classifier. The fact that SDAIC-3 outperforms SDA-6 shows that it is not simply the additional capacity of SDAIC-3 with respect to SDA-3 and DBN-3 that explains these performance differences. We also tried to add a phase of global *unsupervised* fine-tuning[5] before the supervised fine-tuning of SDA-6, but it at best improved only slightly its performance, not reaching the performance of SDAIC-3. This confirms the primary importance of pre-training with a DA-IC greedy module.

5.2 QUALITATIVE ANALYSIS OF LEARNT PARAMETERS

To get a better idea of the type of interactions the lateral connections are able to capture, we display in Fig. 3 the values of the weights or filters learned for each neuron, as well as the weights for pairs of neurons which have strong positive or negative lateral connections. Black, mid-gray and white pixels in the filters correspond to weights of -3, 0, and 3 respectively, with intermediate values corresponding to intermediate shades. The DA-IC was trained for 2.5 million updates on samples from the *OCR-letters* dataset, with a learning rate of 0.005, $\alpha = 0.25$ and a small L1 weight decay of 0.0001. The learned filters detect various aspects of the input, such as small pen strokes, which have localized positive weights and negative biases[6] (thus will be active only if a pen stroke is present), and regions of empty backgrounds, which have localized negative weights and positive biases (thus will only be active if no pen stroke is present). There are also filters that can determine whether the width and height of a character is smaller than a certain number of pixels (see filters with wide horizontal or vertical bars).

The lateral connections also model interesting interactions between these filters. Pairs of neurons that are positively connected often have visually similar filters. Also, pairs of neurons that are negatively connected are sensitive to mutually exclusive patterns in the input. For instance, pairs of pen-stroke and empty background detectors in the same region of the image usually inhibit each other. Another example is two filters that detect whether the sides or the top and bottom of the image are empty (see the first negatively connected pair in Fig. 3), two events that cannot be true simultaneously since all characters touch at least one border of the image.

Next, we wanted to examine more closely the effect of $\mathbf{V}$. We presented a number of input patterns to a DA-IC trained on OCR-letters and considered pairs of neurons in the hidden layer with inhibitory lateral connections between them (corresponding to a negative weight in $\mathbf{V}$). We measured the activity of these neurons before applying $\mathbf{V}$ and after. Fig. 5 shows two examples, together with the filters associated to the considered neurons. A typical inhibitory behaviour can be observed: after applying $\mathbf{V}$ and a nonlinearity, a clear winner emerges within pairs of negatively connected neurons that have equally strong activities

[4]Model selection, based on the classification error obtained on the validation set, was done over the number of iterations of greedy pre-training as well as the value of the learning rates for greedy pre-training and fine-tuning. For denoising autoencoders, the fraction of masked or destroyed inputs α also had to be chosen by model selection; we compared $\alpha = 0.1$ and 0.25. Early-stopping based on the validation set error determined the number of fine-tuning iterations.

[5]Global unsupervised fine-tuning consists in optimizing reconstruction error after a full up and down pass through all the layers.

[6]To simplify the visualization, the value of the biases are not shown in Fig. 3

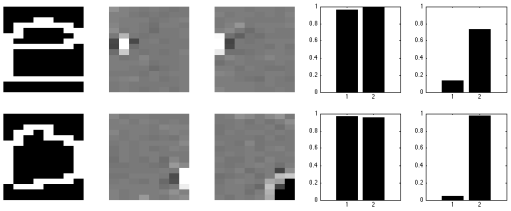

Figure 5: Illustration of inhibitory behaviour. Two examples are shown: **e** and **o**. In each, from left to right: the input pattern, the filters for two neurons of the first hidden layer, the values taken by these neurons before taking into account lateral connection weights $\mathbf{V}$, and their values after applying $\mathbf{V}$ and a sigmoid. As can be seen, lateral connections allow to disambiguate situations in which we have equally strong initial responses from the two neurons.

before applying $\mathbf{V}$. In the **e** example, the competition is between detecting a vertical segments on the left edge, or detecting it one pixel to the right. These are unlikely to occur together. In the **o** example, the choice is between detecting an empty spot in the lower right corner or seeing a vertical segment on the right edge that continues nearly to the bottom of the corner. Again, the two are contradictory. In both cases, inhibitory connections appear crucial in choosing the feature that better describes the input pattern. This disambiguation between two conflicting aspects in the input would not be possible with a simple layer that does not correct for interdependencies.

5.3 COMPARISON WITH ALTERNATIVE TECHNIQUES FOR LEARNING LATERAL INTERACTIONS

Next, we wanted to see how our simple method for learning lateral interactions (DA-IC) compared to alternatives based on iterating a recursive equation, as previously proposed. Due to these alternatives being very time consuming, we focused on unsupervised training of a single layer (greedy module) to learn a representation (code)[7]. We then measured the classification performance obtained by a linear least squares classifier that uses that learned code as input. We specifically considered the following greedy modules:

- **RBM:** Restricted Boltzmann Machine with no lateral connections.
- **DA:** Ordinary Denoising Autoencoder, no lateral connections.

[7]We tested using both 10 and 30 iterations through Equation 3. Notice that computing $\mathbf{enc}(\widetilde{\mathbf{x}})$ with these alternative models requires **10 and 30 times** (respectively) as many multiply-add operations involving the $H^2 - H$ lateral connections V_{jk}, where H is the number of hidden units (the diagonal of $\mathbf{V}$ is not used in Equation 3).

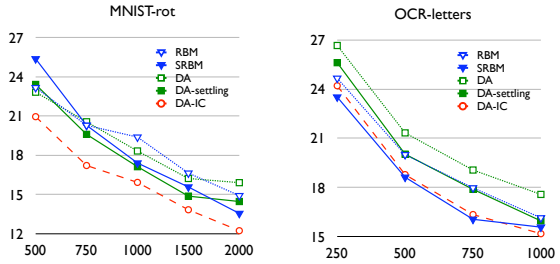

Figure 6: Test classification error (%) of a linear classifier using the codes learned by different types of greedy modules, for increasing size of hidden layer.

- **SRBM:** Semi-Restricted Boltzmann Machines (Osindero & Hinton, 2008), but with lateral connections between hidden units, instead of visible units as originally proposed.
- **DA-settling:** Denoising Autoencoder with "settling" lateral connections in the encoder: i.e. we iterate several times through Eq. 3.
- **DA-IC:** Our proposed Denoising Autoencoder with Interdependent Codes.

Fig. 6 gives the resulting classification performance as a function of the size of the code (the number of hidden units). We emphasize that the codes were learned in an entirely unsupervised fashion[8]. We observe that DA-IC systematically outperforms both RBM and DA (differences are statistically significant, except for 250 units on OCR-letters). When compared to the alternative techniques for introducing lateral interactions, DA-IC outperforms them on *MNIST-rot* (differences are statistically significant), and is also best (statistically equivalent to SRBM) on OCR-letters. We want to emphasize here that, contrary to the alternative techniques involving iterating a recursive equation, DA-IC is very simple and computationally very cheap (no iteration involved).

5.4 ANALYSIS OF CORRELATION

Finally, we provide a possible explanation as to why DA-ICs are better suited for deep learning. The performance of deep networks with 1, 2 and 3 stacked DA-ICs is 10.33%, 8.91% and 8.07% respectively on the *MNIST-rot* dataset, which confirms that the DA-IC can leverage the addition of layers. Intuitively, a necessary condition for a greedy module to be appropriate for deep learning is that it should compute representations which, while being informative of the input, are not too linearly correlated. Otherwise, some of the coding elements would be easily predictable by

[8]Only the number of unsupervised training iterations and the learning rate were selected based on classification performance on the validation set

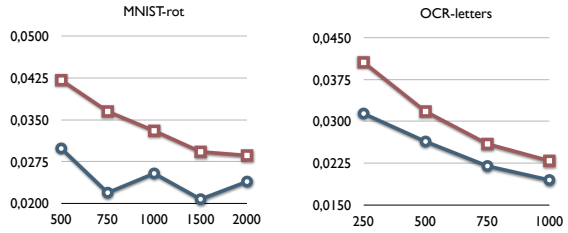

Figure 7: Mean pairwise absolute correlation between the coding elements of a basic denoising autoencoder (squares) and a denoising autoencoder with interdependent codes (circles), for different layer sizes.

others and therefore essentially useless. Since denoising autoencoders use a log-linear decoder function, training implicitly discourages highly correlated hidden units, which would waste some of the capacity of the encoder. However, as the size of the hidden layer grows, it is likely that adding uncorrelated units requires more non-linear computations from the encoder. So, by adding lateral connections to the encoder function, we would expect the encoder to be better able to reduce the correlation in its code units. To verify this claim, we computed the mean of pairwise absolute correlations between the activities of the hidden units of a denoising autoencoder and of a DA-IC for several large sizes of hidden layers, on the *MNIST-rot* dataset. Model selection was performed based on the mean absolute correlations obtained on the validation set. The result, reported in Fig. 7, confirms that interdependent codes exhibit less correlation between their elements.

6 CONCLUSION

We presented a simple extension of denoising autoencoders which allows learning inhibitory and excitatory interactions between the hidden code units and demonstrated their usefulness as greedy modules for deep learning. Experiments on two character recognition problems showed that using Denoising Autoencoder with Interdependent Codes (DA-IC) **outperforms state-of-the-art learning algorithms for deep networks classifiers and kernel SVMs**. While the technique we use for taking into account lateral interactions is both **simpler and computationally much more efficient than previously proposed alternative techniques** (based on a recursive update equation) we showed it does learn codes that yield **equivalent or better classification performance** than these more cumbersome alternatives.

Acknowledgements

The authors thank Yoshua Bengio for constructive discussions. This research was supported by MITACS.

References

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. *Advances in NIPS 19*.

Bengio, Y., & LeCun, Y. (2007). Scaling learning algorithms towards AI. In L. Bottou, O. Chapelle, D. DeCoste and J. Weston (Eds.), *Large scale kernel machines*. MIT Press.

Garrigues, P., & Olshausen, B. (2008). Learning horizontal connections in a sparse coding model of natural images. In *Nips'20*.

Hinton, G., Osindero, S., & Bao, K. (2005). Learning causally linked markov random fields. *AISTATS'05*.

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*, 1527–1554.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*, 504–507.

Hyvärinen, A., Hoyer, P. O., & Inki, M. O. (2001). Topographic independent component analysis. *Neural Computation*, *13*, 1527–1558.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., & Bengio, Y. (2007). An empirical evaluation of deep architectures on problems with many factors of variation. *ICML*.

Osindero, S., & Hinton, G. (2008). Modeling image patches with a directed hierarchy of markov random field. *NIPS 20*.

Osindero, S., Welling, M., & Hinton, G. E. (2006). Topographic product models applied to natural scene statistics. *Neural Computation*, *18*, 381–414.

Raina, R., Battle, A., Lee, H., Packer, B., & Ng, A. Y. (2007). Self-taught learning: transfer learning from unlabeled data. *ICML 2007*.

Ranzato, M., Boureau, Y., & LeCun, Y. (2008). Sparse feature learning for deep belief networks. In *Nips 20*.

Ranzato, M., Huang, F., Boureau, Y., & LeCun, Y. (2007). Unsupervised learning of invariant feature hierarchies with applications to object recognition. *CVPR'07*.

Salakhutdinov, R., & Hinton, G. E. (2007). Semantic hashing. *SIGIR*.

Salakhutdinov, R., & Hinton, G. E. (2008). Using deep belief nets to learn covariance kernels for gaussian processes. In *Nips 20*.

Shriki, O., Sompolinsky, H., & Lee, D. D. (2001). An information maximization approach to overcomplete and recurrent representations. *NIPS 13*.

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. *Proceedings of ICML'2008* (pp. 1096–1103).

Weston, J., Ratle, F., & Collobert, R. (2008). Deep learning via semi-supervised embedding. *ICML 2008*.