Deep Learners Benefit More from Out-of-Distribution Examples

Yoshua Bengio and **Frédéric Bastien** and **Arnaud Bergeron** and **Nicolas Boulanger-Lewandowski** and **Thomas Breuel** and **Youssouf Chherawala** and **Moustapha Cisse** and **Myriam Côté** and **Dumitru Erhan** and **Jeremy Eustache** and **Xavier Glorot** and **Xavier Muller** and **Sylvain Pannetier Lebeuf** and **Razvan Pascanu** and **Salah Rifai** and **Francois Savard** and **Guillaume Sicard**

Dept. IRO, U. Montreal, P.O. Box 6128, Centre-Ville branch, H3C 3J7, Montreal (Qc), Canada

Abstract

Recent theoretical and empirical work in statistical machine learning has demonstrated the potential of learning algorithms for deep architectures, i.e., function classes obtained by composing multiple levels of representation. The hypothesis evaluated here is that intermediate levels of representation, because they can be shared across tasks and examples from different but related distributions, can yield even more benefits. Comparative experiments were performed on a large-scale handwritten character recognition setting with 62 classes (upper case, lower case, digits), using both a multi-task setting and perturbed examples in order to obtain out-of-distribution examples. The results agree with the hypothesis, and show that a deep learner did *beat previously published results and reached human-level performance.*

1 Introduction

Deep Learning has emerged as a promising new area of research in statistical machine learning [Hinton et al., 2006, Ranzato et al., 2007, Bengio et al., 2007, Vincent et al., 2008, Ranzato et al., 2008, Taylor and Hinton, 2009, Larochelle et al., 2009, Salakhutdinov and Hinton, 2009, Lee et al., 2009a,b, Jarrett et al., 2009, Taylor et al., 2010]. See Bengio [2009] for a review. Learning algorithms for deep architectures are centered on the learning of useful representations of data, which are better suited to the task at hand, and are organized in a hierarchy with multiple levels. This is in part inspired by observations of the mammalian visual cortex, which consists of a chain of processing elements, each of which is associated with a different representation of the raw visual input. In fact, it was found recently that the features learnt in deep architectures resemble those observed in the first two of these stages (in areas V1 and V2 of visual cortex) [Lee et al., 2008], and that they become more and more invariant to factors of variation (such as camera movement) in higher layers [Goodfellow et al., 2009]. It has been hypothesized that learning a hierarchy of features increases the ease and practicality of developing representations that are at once tailored to specific tasks, yet are able to borrow statistical strength from other related tasks (e.g., modeling different kinds of objects). Finally, learning the feature representation can lead to higher-level (more abstract, more general) features that are more robust to unanticipated sources of variance extant in real data.

Whereas a deep architecture can in principle be more powerful than a shallow one in terms of representation, depth appears to render the training problem more difficult in terms of optimization and local minima. It is also only recently that successful algorithms were proposed to overcome some of these difficulties. All are based on unsupervised learning, often in an greedy layer-wise "unsupervised pre-training" stage [Bengio, 2009]. The principle is that each layer starting from the bottom is trained to represent its input (the output of the previous layer). After this unsupervised initialization, the stack of layers can be converted into a deep supervised feedforward neural network and fine-tuned by stochastic gradient descent. One of these layer initialization techniques, applied here, is the Denoising Auto-encoder (DA) [Vincent et al., 2008] (see Figure 2), which performed similarly or better [Vincent et al., 2008] than previously proposed Restricted Boltzmann Machines (RBM) [Hinton et al., 2006] in terms of unsupervised extraction of a hierarchy of features useful for classification. Each layer is trained to denoise its input, creating a layer of features that can be

Appearing in Proceedings of the 14^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2011, Fort Lauderdale, FL, USA. Volume 15 of JMLR: W&CP 15.

used as input for the next layer, forming a Stacked Denoising Auto-encoder (SDA). Note that training a Denoising Auto-encoder can actually been seen as training a particular RBM by an inductive principle different from maximum likelihood [Vincent, 2010], namely by Score Matching [Hyvärinen, 2005, Hyvärinen, 2008].

Previous comparative experimental results with stacking of RBMs and DAs to build deep supervised predictors had shown that they could outperform shallow architectures in a variety of settings, especially when the data involves complex interactions between many factors of variation [Larochelle et al., 2007, Bengio, 2009]. Other experiments have suggested that the unsupervised layer-wise pre-training acted as a useful prior [Erhan et al., 2010] that allows one to initialize a deep neural network in a relatively much smaller region of parameter space, corresponding to better generalization.

To further the understanding of the reasons for the good performance observed with deep learners, we focus here on the following *hypothesis*: intermediate levels of representation, especially when there are more such levels, can be exploited to **share statistical strength across different but related types of examples**, such as examples coming from other tasks than the task of interest (the multi-task setting [Caruana, 1997]), or examples coming from an overlapping but different distribution (images with different kinds of perturbations and noises, here). This is consistent with the hypotheses discussed in Bengio [2009] regarding the potential advantage of deep learning and the idea that more levels of representation can give rise to more abstract, more general features of the raw input.

This hypothesis is related to a learning setting called **self-taught learning** [Raina et al., 2007], which combines principles of semi-supervised and multi-task learning: in addition to the labeled examples from the target distribution, the learner can exploit examples that are unlabeled and possibly come from a distribution different from the target distribution, e.g., from other classes than those of interest. It has already been shown that deep learners can clearly take advantage of unsupervised learning and unlabeled examples [Bengio, 2009, Weston et al., 2008] in order to improve performance on a supervised task, but more needed to be done to explore the impact of *out-of-distribution* examples and of the *multi-task* setting (two exceptions are Collobert and Weston [2008], which shares and uses unsupervised pre-training only with the first layer, and Mobahi et al. [2009] in the case of video data). In particular the *relative advantage of deep learning* for these settings has not been evaluated.

The **main claim** of this paper is that deep learners (with several levels of representation) can **benefit more from out-of-distribution examples than shallow learners** (with a single level), both in the context of the multi-task setting and from perturbed examples. Because we are able to improve on state-of-the-art performance and reach human-level performance on a large-scale task, we consider that this paper is also a contribution to advance the application of machine learning to handwritten character recognition. More precisely, we ask and answer the following questions:

• Do the good results previously obtained with deep architectures on the MNIST digit images generalize to the setting of a similar but much larger and richer dataset, the NIST special database 19, with 62 classes and around 800k examples?

• To what extent does the perturbation of input images (e.g. adding noise, affine transformations, background images) make the resulting classifiers better not only on similarly perturbed images but also on the *original clean examples*? We study this question in the context of the 62-class and 10-class tasks of the NIST special database 19.

• Do deep architectures *benefit* **more** *from such out-of-distribution* examples, in particular do they benefit more from examples that are perturbed versions of the examples from the task of interest?

• Similarly, does the feature learning step in deep learning algorithms benefit **more** from training with moderately *different classes* (i.e. a multi-task learning scenario) than a corresponding shallow and purely supervised architecture? We train on 62 classes and test on 10 (digits) or 26 (upper case or lower case) to answer this question.

Our experimental results provide positive evidence towards all of these questions, as well as **classifiers that reach human-level performance on 62-class isolated character recognition and beat previously published results on the NIST dataset (special database 19)**. To achieve these results, we introduce in the next section a sophisticated system for stochastically transforming character images and then explain the methodology, which is based on training with or without these transformed images and testing on clean ones. Code for generating these transformations as well as for the deep learning algorithms are made available at `http://hg.assembla.com/ift6266`.

2 Perturbed and Transformed Character Images

Figure 1 shows the different transformations we used to stochastically transform 32×32 source images (such as the one in Fig.1(a)) in order to obtain data from a larger distribution which covers a domain substantially larger than the clean characters distribution from which we start. Although character transformations

have been used before to improve character recognizers, this effort is on a large scale both in number of classes and in the complexity of the transformations, hence in the complexity of the learning task. The code for these transformations (mostly Python) is available at `http://hg.assembla.com/ift6266`. All the modules in the pipeline (Figure 1) share a global control parameter ($0 \leq \textit{complexity} \leq 1$) that allows one to modulate the amount of deformation or noise introduced. There are two main parts in the pipeline. The first one, from thickness to pinch, performs transformations. The second part, from blur to contrast, adds different kinds of noise. More details can be found in Bastien et al. [2010].

3 Experimental Setup

Much previous work on deep learning had been performed on the MNIST digits task [Hinton et al., 2006, Ranzato et al., 2007, Bengio et al., 2007, Salakhutdinov and Hinton, 2009], with 60,000 examples, and variants involving 10,000 examples [Larochelle et al., 2009, Vincent et al., 2008][1] The focus here is on much larger training sets, from 10 times to to 1000 times larger, and 62 classes.

The first step in constructing the larger datasets (called NISTP and P07) is to sample from a *data source*: **NIST** (NIST database 19), **Fonts**, **Captchas**, and **OCR data** (scanned machine printed characters). See more in Section 3.1 below. Once a character is sampled from one of these sources (chosen randomly), the second step is to apply a pipeline of transformations and/or noise processes outlined in section 2.

To provide a baseline of error rate comparison we also estimate human performance on both the 62-class task and the 10-class digits task. We compare the best Multi-Layer Perceptrons (MLP) against the best Stacked Denoising Auto-encoders (SDA), when both models' hyper-parameters are selected to minimize the validation set error. We also provide a comparison against a precise estimate of human performance obtained via Amazon's Mechanical Turk (AMT) service (`http://mturk.com`). AMT users are paid small amounts of money to perform tasks for which human intelligence is required. Mechanical Turk has been used extensively in natural language processing and vision. AMT users were presented with 10 character images (from a test set) on a screen and asked to label them. They were forced to choose a single character class (either among the 62 or 10 character classes) for each image. 80 subjects classified 2500 images per (dataset,task) pair. Different humans labelers sometimes provided a different label for the same example, and we were able to estimate the error variance due to this effect because each image was classified by 3 different persons. The average error of humans on the 62-class task NIST test set is 18.2%, with a standard error of 0.1%. We controlled noise in the labelling process by (1) requiring AMT workers with a higher than normal average of accepted responses (>95%) on other tasks (2) discarding responses that were not complete (10 predictions) (3) discarding responses for which for which the time to predict was smaller than 3 seconds for NIST (the mean response time was 20 seconds) and 6 seconds seconds for NISTP (average response time of 45 seconds) (4) discarding responses which were obviously wrong (10 identical ones, or "12345..."). Overall, after such filtering, we kept approximately 95% of the AMT workers' responses.

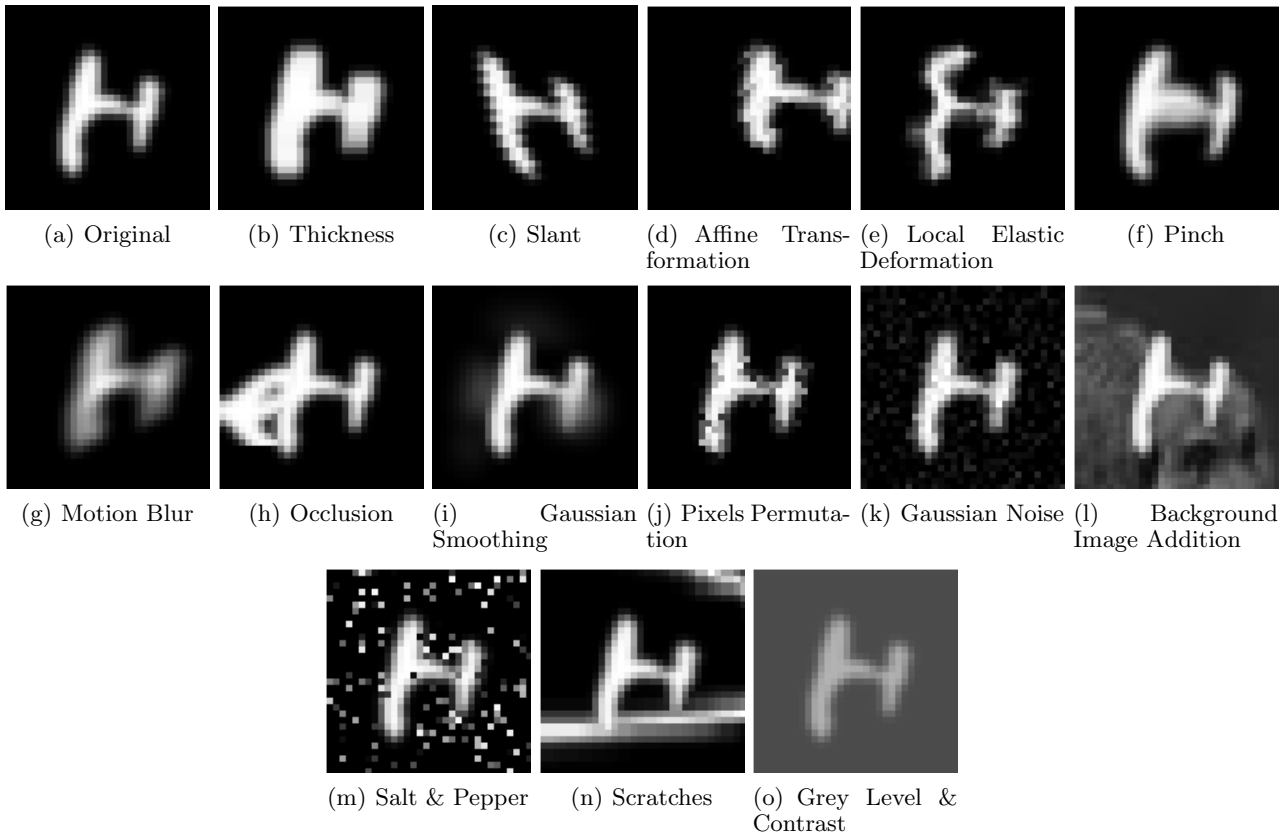
[1] Fortunately, there are more and more exceptions of course, such as Raina et al. [2009] using a million examples.

3.1 Data Sources

NIST. Our main source of characters is the NIST Special Database 19 [Grother, 1995], widely used for training and testing character recognition systems [Granger et al., 2007, Pérez-Cortes et al., 2000, Oliveira et al., 2002, Milgram et al., 2005]. The dataset is composed of 814255 digits and characters (upper and lower cases), with hand checked classifications, extracted from handwritten sample forms of 3600 writers. The characters are labelled by one of the 62 classes corresponding to "0"-"9","A"-"Z" and "a"-"z". The dataset contains 8 parts (partitions) of varying complexity. The fourth partition (called hsf_4, 82,587 examples), experimentally recognized to be the most difficult one, is the one recommended by NIST as a testing set and is used in our work as well as some previous work [Granger et al., 2007, Pérez-Cortes et al., 2000, Oliveira et al., 2002, Milgram et al., 2005] for that purpose. We randomly split the remainder (731,668 examples) into a training set and a validation set for model selection. The performances reported by previous work on that dataset mostly use only the digits. Here we use all the classes both in the training and testing phase. This is especially useful to estimate the effect of a multi-task setting. The distribution of the classes in the NIST training and test sets differs substantially, with relatively many more digits in the test set, and a more uniform distribution of letters in the test set (whereas in the training set they are distributed more like in natural text).

Fonts. In order to have a good variety of sources we downloaded an important number of free fonts from: `http://cg.scs.carleton.ca/~luc/freefonts.html`. Including an operating system's (Windows 7) fonts, there we uniformly chose from 9817 different fonts. The chosen `ttf` file is either used as input of the Captcha generator (see next item) or, by producing a corresponding image, directly as input to our models.

Captchas. The Captcha data source is an adaptation of the *pycaptcha* library (a Python-based captcha generator library) for generating characters of the same format as the NIST dataset. This software is based on a random



(a) Original (b) Thickness (c) Slant (d) Affine Transformation (e) Local Elastic Deformation (f) Pinch

(g) Motion Blur (h) Occlusion (i) Gaussian Smoothing (j) Pixels Permutation (k) Gaussian Noise (l) Background Image Addition

(m) Salt & Pepper (n) Scratches (o) Grey Level & Contrast

Figure 1: Top left (a): example original image. Others (b-o): examples of the effect of each transformation module taken separately. Actual perturbed examples are obtained by a pipeline of these, with random choices about which module to apply and how much perturbation to apply.

character class generator and various kinds of transformations similar to those described in the previous sections. In order to increase the variability of the data generated, many different fonts are used for generating the characters. Transformations (slant, distortions, rotation, translation) are applied to each randomly generated character with a complexity depending on the value of the complexity parameter provided by the user of the data source.

OCR data. A large set (2 million) of scanned, OCRed and manually verified machine-printed characters where included as an additional source. This set is part of a larger corpus being collected by the Image Understanding Pattern Recognition Research group led by Thomas Breuel at University of Kaiserslautern (`http://www.iupr.com`).

3.2 Data Sets

All data sets contain 32×32 grey-level images (values in $[0, 1]$) associated with one of 62 character labels.

NIST. This is the raw NIST special database 19 [Grother, 1995]. It has {651,668 / 80,000 / 82,587} {training / validation / test} examples.

P07. This dataset is obtained by taking raw characters from the above 4 sources and sending them through the transformation pipeline described in section 2. For each generated example, a data source is selected with probability 10% from the fonts, 25% from the captchas, 25% from the OCR data and 40% from NIST. The transformations are applied in the order given above, and for each of them we sample uniformly a *complexity* in the range $[0, 0.7]$. It has {81,920,000 / 80,000 / 20,000} {training / validation / test} examples obtained from the corresponding NIST sets plus other sources.

NISTP. This one is equivalent to P07 (complexity parameter of 0.7 with the same proportions of data sources) except that we only apply transformations from slant to pinch (see Fig.1(b-f)). Therefore, the character is transformed but without added noise, yielding images closer to the NIST dataset. It has {81,920,000 / 80,000 / 20,000} {training / validation / test} examples obtained from the corresponding NIST sets plus other sources.

3.3 Models and their Hyper-parameters

The experiments are performed using MLPs (with a single hidden layer) and deep SDAs. *Hyper-parameters are*

selected based on the **NISTP** *validation set error.*

Multi-Layer Perceptrons (MLP). The MLP output estimates the class-conditional probabilities

$$P(\text{class}|\text{input} = x) = \text{softmax}(b_2 + W_2 \tanh(b_1 + W_1 x)),$$

i.e., two layers, where $p = \text{softmax}(a)$ means that $p_i(x) = \exp(a_i)/\sum_j \exp(a_j)$ representing the probability for class i, tanh is the element-wise hyperbolic tangent, b_i are parameter vectors, and W_i are parameter matrices (one per layer). The number of rows of W_1 is called the number of hidden units (of the single hidden layer, here), and is one way to control capacity (the main other ways to control capacity are the number of training iterations and optionally a regularization penalty on the parameters, not used here because it did not help). Whereas previous work had compared deep architectures to both shallow MLPs and SVMs, we only compared to MLPs here because of the very large datasets used (making the use of SVMs computationally challenging because of their quadratic scaling behavior). Preliminary experiments on training SVMs (libSVM) with subsets of the training set allowing the program to fit in memory yielded substantially worse results than those obtained with MLPs[2] For training on nearly a hundred million examples (with the perturbed data), the MLPs and SDA are much more convenient than classifiers based on kernel methods. The MLP has a single hidden layer with tanh activation functions, and softmax (normalized exponentials) on the output layer for estimating $P(\text{class}|\text{input})$. The number of hidden units is taken in $\{300, 500, 800, 1000, 1500\}$. Training examples are presented in minibatches of size 20, i.e., the parameters are iteratively updated in the direction of the mean gradient of the next 20 examples. A constant learning rate was chosen among $\{0.001, 0.01, 0.025, 0.075, 0.1, 0.5\}$.

Stacked Denoising Auto-encoders (SDA). Various auto-encoder variants and Restricted Boltzmann Machines (RBMs) can be used to initialize the weights of each layer of a deep MLP (with many hidden layers) [Hinton et al., 2006, Ranzato et al., 2007, Bengio et al., 2007], apparently setting parameters in the basin of attraction of supervised gradient descent yielding better generalization [Erhan et al., 2010]. This initial *unsupervised pre-training phase* does not use the training labels. Each layer is trained in turn to produce a new representation of its input (starting from the raw pixels). It is hypothesized that the advantage brought by this procedure stems from a better prior, on the one hand taking advantage of the link between the input distribution $P(x)$ and the conditional distribution of interest $P(y|x)$ (like in semi-supervised learning), and on the other hand taking advantage of the expressive power and bias implicit in the deep architecture (whereby complex concepts are expressed as compositions of simpler ones through a deep hierarchy).

Here we chose to use the Denoising Auto-encoder [Vincent et al., 2008] as the building block for these deep hierarchies of features, as it is simple to train and explain (see Figure 2, as well as tutorial and code there: `http://deeplearning.net/tutorial`), provides efficient inference, and yielded results comparable or better than RBMs in series of experiments [Vincent et al., 2008]. Some denoising auto-encoders correspond to a Gaussian RBM trained by a Score Matching criterion Vincent [2010]. During its unsupervised training, a Denoising Auto-encoder is presented with a stochastically corrupted version $\tilde{x}$ of the input x and trained to reconstruct to produce a reconstruction z of the uncorrupted input x. Because the network has to denoise, it is forcing the hidden units y to represent the leading regularities in the data. In a slight departure from Vincent et al. [2008], the hidden units output y is obtained through the tanh-affine encoder $y = \tanh(c+Vx)$ and the reconstruction is obtained through the transposed transformation $z = \tanh(d+V'y)$. The training set average of the cross-entropy reconstruction loss (after mapping back numbers in (-1,1) into (0,1))

$$L_H(x, z) = -\sum_i \frac{(z_i + 1)}{2} \log \frac{(x_i + 1)}{2} + \frac{z_i}{2} \log \frac{x_i}{2}$$

is minimized. Here we use the random binary masking corruption (which in $\tilde{x}$ sets to 0 a random subset of the elements of x, and copies the rest). Once the first denoising auto-encoder is trained, its parameters can be used to set the first layer of the deep MLP. The original data are then processed through that first layer, and the output of the hidden units form a new representation that can be used as input data for training a second denoising auto-encoder, still in a purely unsupervised way. This is repeated for the desired number of hidden layers. After this unsupervised pre-training stage, the parameters are used to initialize a deep MLP (similar to the above, but with more layers), which is fine-tuned by the same standard procedure (stochastic gradient descent) used to train MLPs in general (see above). The top layer parameters of the deep MLP (the one which outputs the class probabilities and takes the top hidden layer as input) can be initialized at 0. The SDA hyper-parameters are the same as for the MLP, with the addition of the amount of corruption noise (we used the masking noise process, whereby a fixed proportion of the input values, randomly selected, are zeroed), and a separate learning rate for the unsupervised pre-training stage (selected from the same above set). The fraction of inputs corrupted

[2]RBF SVMs trained with a subset of NISTP or NIST, 100k examples, to fit in memory, yielded 64% test error or worse; online linear SVMs trained on the whole of NIST or 800k from NISTP yielded no better than 42% error; slightly better results were obtained by sparsifying the pixel intensities and projecting to a second-order polynomial (a very sparse vector), still 41% error. We expect that better results could be obtained with a better implementation allowing for training with more examples and a higher-order non-linear projection.

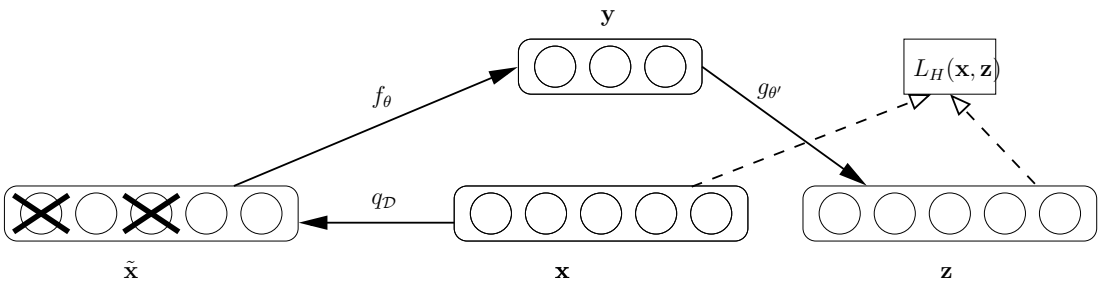


Figure 2: Illustration of the computations and training criterion for the denoising auto-encoder used to pre-train each layer of the deep architecture. Input x of the layer (i.e. raw input or output of previous layer) s corrupted into $\tilde{x}$ and encoded into code y by the encoder $f_\theta(\cdot)$. The decoder $g_{\theta'}(\cdot)$ maps y to reconstruction z, which is compared to the uncorrupted input x through the loss function $L_H(x, z)$, whose expected value is approximately minimized during training by tuning θ and θ'.

was selected among $\{10\%, 20\%, 50\%\}$. Another hyper-parameter is the number of hidden layers but it was fixed to 3 for our experiments, based on previous work with SDAs on MNIST [Vincent et al., 2008]. We also compared against 1 and against 2 hidden layers, to disantangle the effect of depth from that of unsupervised pre-training. The size of each hidden layer was kept constant across hidden layers, and the best results were obtained with the largest values that we tried (1000 hidden units).

4 Experimental Results

The models are either trained on NIST (MLP0 and SDA0), NISTP (MLP1 and SDA1), or P07 (MLP2 and SDA2), and tested on either NIST, NISTP or P07 (regardless of the data set used for training), either on the 62-class task or on the 10-digits task. Training time (including about half for unsupervised pre-training, for DAs) on the larger datasets is around one day on a GPU (GTX 285). Figure 3 summarizes the results obtained, comparing humans, the three MLPs (MLP0, MLP1, MLP2) and the three SDAs (SDA0, SDA1, SDA2), along with the previous results on the digits NIST special database 19 test set from the literature, respectively based on ARTMAP neural networks [Granger et al., 2007], fast nearest-neighbor search [Pérez-Cortes et al., 2000], MLPs [Oliveira et al., 2002], and SVMs [Milgram et al., 2005].The deep learner not only outperformed the shallow ones and previously published performance (in a statistically and qualitatively significant way) but when trained with perturbed data reaches human performance on both the 62-class task and the 10-class (digits) task. 17% error (SDA1) or 18% error (humans) may seem large but a large majority of the errors from humans and from SDA1 are from out-of-context confusions (e.g. a vertical bar can be a "1", an "l" or an "L", and a "c" and a "C" are often indistinguishible). Regarding shallower networks pre-trained with unsupervised denoising auto-encders, we find that the NIST test error is 21% with one hidden layer and 20% with two hidden layers (vs 17% in the same conditions with 3 hidden layers). Compare this with the 23% error achieved by the MLP, i.e. a single hidden layer and no unsupervised pre-training. As found in previous work Erhan et al. [2010], Larochelle et al. [2009], these results show that both depth and unsupervised pre-training need to be combined in order to achieve the best results.

In addition, as shown in the left of Figure 4, the relative improvement in error rate brought by out-of-distribution examples is greater for the deep SDA, and these differences with the shallow MLP are statistically and qualitatively significant. The left side of the figure shows the improvement to the clean NIST test set error brought by the use of out-of-distribution examples (i.e. the perturbed examples examples from NISTP or P07), over the models trained exclusively on NIST (respectively SDA0 and MLP0). Relative percent change is measured by taking 100%× (original model's error / perturbed-data model's error - 1). The right side of Figure 4 shows the relative improvement brought by the use of a multi-task setting, in which the same model is trained for more classes than the target classes of interest (i.e. training with all 62 classes when the target classes are respectively the digits, lower-case, or upper-case characters). Again, whereas the gain from the multi-task setting is marginal or negative for the MLP, it is substantial for the SDA. Note that to simplify these multi-task experiments, only the original NIST dataset is used. For example, the MLP-digits bar shows the relative percent improvement in MLP error rate on the NIST digits test set as 100%× (single-task model's error / multi-task model's error - 1). The single-task model is trained with only 10 outputs (one per digit), seeing only digit examples, whereas the multi-task model is trained with 62 outputs, with all 62 character classes as examples. Hence the hidden units are shared across all tasks. For the multi-task model, the digit error rate is measured by comparing the correct digit class with the output class associated with the maximum conditional probability among only the digit classes outputs. The

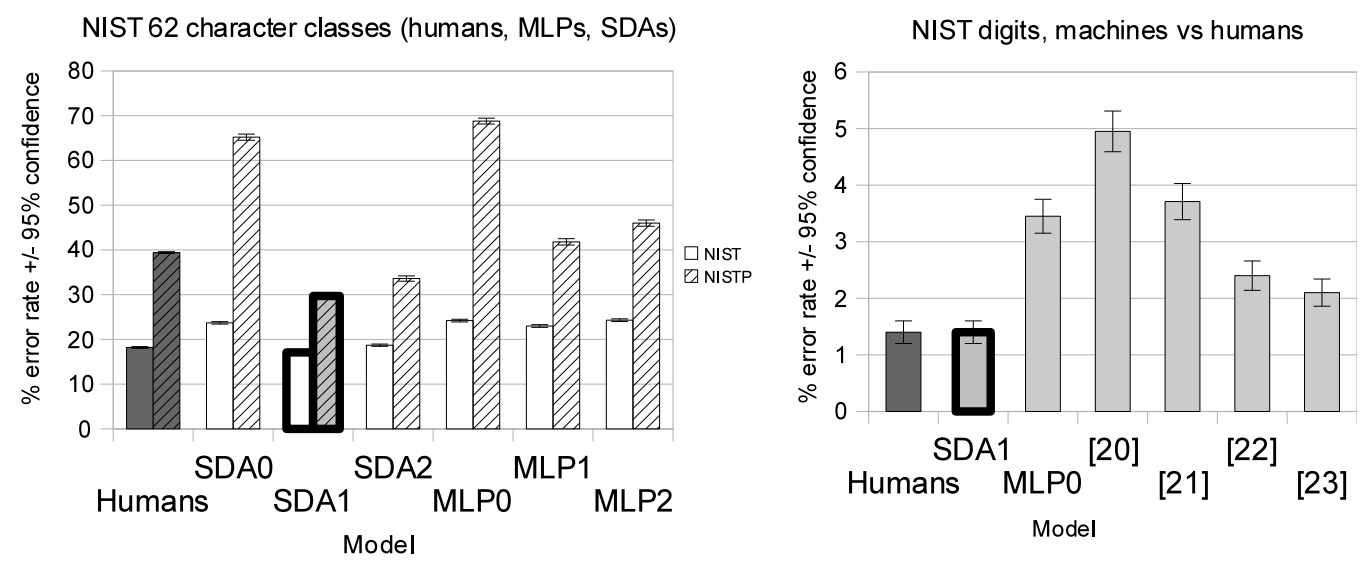


Figure 3: SDAx are the **deep** models. Error bars indicate a 95% confidence interval. 0 indicates that the model was trained on NIST, 1 on NISTP, and 2 on P07. Left: overall results of all models, on NIST and NISTP test sets. Right: error rates on NIST test digits only, along with the previous results from literature [Granger et al., 2007, Pérez-Cortes et al., 2000, Oliveira et al., 2002, Milgram et al., 2005] respectively based on ART, nearest neighbors, MLPs, and SVMs.

setting is similar for the other two target classes (lower case characters and upper case characters). Note however that some types of perturbations (NISTP) help more than others (P07) when testing on the clean images.

5 Conclusions and Discussion

We have found that out-of-distribution examples (multi-task learning and perturbed examples) are more beneficial to a deep learner than to a traditional shallow and purely supervised learner. More precisely, the answers are positive for all the questions asked in the introduction.

• **Do the good results previously obtained with deep architectures on the MNIST digits generalize to a much larger and richer (but similar) dataset, the NIST special database 19, with 62 classes and around 800k examples**? Yes, the SDA *systematically outperformed the MLP and all the previously published results on this dataset* (the ones that we are aware of), *in fact reaching human-level performance* at around 17% error on the 62-class task and 1.4% on the digits, and beating previously published results on the same data.

• **To what extent do out-of-distribution examples help deep learners, and do they help them more than shallow supervised ones**? We found that distorted training examples not only made the resulting classifier better on similarly perturbed images but also on the *original clean examples*, and more importantly and more novel, that deep architectures benefit more from such *out-of-distribution* examples. Shallow MLPs were helped by perturbed training examples when tested on perturbed input images (65% relative improvement on NISTP) but only marginally helped (5% relative improvement on all classes) or even hurt (10% relative loss on digits) with respect to clean examples. On the other hand, the deep SDAs were significantly boosted by these out-of-distribution examples. Similarly, whereas the improvement due to the multi-task setting was marginal or negative for the MLP (from +5.6% to -3.6% relative change), it was quite significant for the SDA (from +13% to +27% relative change), which may be explained by the arguments below. Since out-of-distribution data (perturbed or from other related classes) is very common, this conclusion is of practical importance.

In the original self-taught learning framework [Raina et al., 2007], the out-of-sample examples were used as a source of unsupervised data, and experiments showed its positive effects in a *limited labeled data* scenario. However, many of the results by Raina et al. [2007] (who used a shallow, sparse coding approach) suggest that the *relative gain of self-taught learning vs ordinary supervised learning* diminishes as the number of labeled examples increases. We note instead that, for deep architectures, our experiments show that such a positive effect is accomplished even in a scenario with a *large number of labeled examples*, i.e., here, the relative gain of self-taught learning and out-of-distribution examples is probably preserved in the asymptotic regime. However, note that in our perturbation experiments (but not in our multi-task experiments), even the out-of-distribution examples are

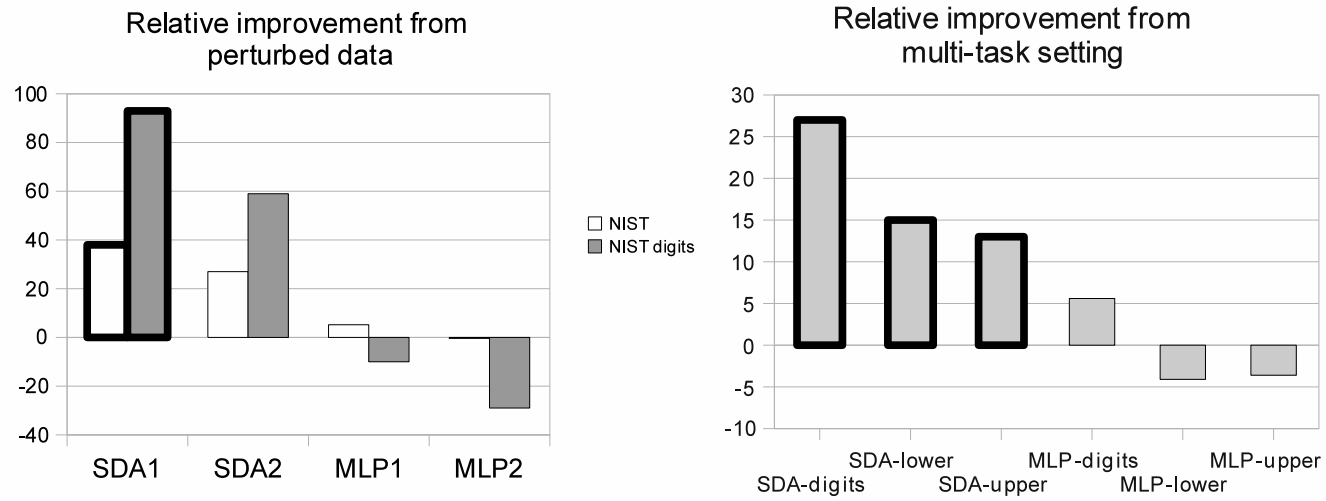


Figure 4: Relative improvement in error rate due to out-of-distribution examples. Left: Improvement (or loss, when negative) induced by out-of-distribution examples (perturbed data). Right: Improvement (or loss, when negative) induced by multi-task learning (training on all classes and testing only on either digits, upper case, or lower-case). The deep learner (SDA) benefits more from out-of-distribution examples, compared to the shallow MLP.

labeled, unlike in the earlier self-taught learning experiments [Raina et al., 2007].

Why would deep learners benefit more from the self-taught learning framework and out-of-distribution examples? The key idea is that the lower layers of the predictor compute a hierarchy of features that can be shared across tasks or across variants of the input distribution. A theoretical analysis of generalization improvements due to sharing of intermediate features across tasks already points towards that explanation [Baxter, 1995]. Intermediate features that can be used in different contexts can be estimated in a way that allows to share statistical strength. Features extracted through many levels are more likely to be more abstract and more invariant to some of the factors of variation in the underlying distribution (as the experiments in Goodfellow et al. [2009] suggest), increasing the likelihood that they would be useful for a larger array of tasks and input conditions. Therefore, we hypothesize that both depth and unsupervised pre-training play a part in explaining the advantages observed here, and future experiments could attempt at teasing apart these factors. And why would deep learners benefit from the self-taught learning scenarios even when the number of labeled examples is very large? We hypothesize that this is related to the hypotheses studied in Erhan et al. [2010]. In Erhan et al. [2010] it was found that online learning on a huge dataset did not make the advantage of the deep learning bias vanish, and a similar phenomenon may be happening here. We hypothesize that unsupervised pre-training of a deep hierarchy with out-of-distribution examples initializes the model in the basin of attraction of supervised gradient descent that corresponds to better generalization. Furthermore, such good basins of attraction are not discovered by pure supervised learning (with or without out-of-distribution examples) from random initialization, and more labeled examples does not allow the shallow or purely supervised models to discover the kind of better basins associated with deep learning and out-of-distribution examples.

A Java demo of the recognizer (where both the MLP and the SDA can be compared) can be executed on-line at `http://deep.host22.com`.

References

Frédéric Bastien, Yoshua Bengio, Arnaud Bergeron, Nicolas Boulanger-Lewandowski, Thomas Breuel, Youssouf Chherawala, Moustapha Cisse, Myriam Côté, Dumitru Erhan, Jeremy Eustache, Xavier Glorot, Xavier Muller, Sylvain Pannetier Lebeuf, Razvan Pascanu, Salah Rifai, François Savard, and Guillaume Sicard. Deep self-taught learning for handwritten character recognition. Technical Report 1353, University of Montréal, 2010.

Jonathan Baxter. Learning internal representations. In *Proceedings of the 8th International Conference on Computational Learning Theory (COLT'95)*, pages 311–320, Santa Cruz, California, 1995. ACM Press.

Yoshua Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009. Also published as a book. Now Publishers, 2009.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *NIPS 19*, pages 153–160. MIT Press, 2007.

Rich Caruana. Multitask learning. *Machine Learning*, 28 (1):41–75, 1997.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML 2008*, pages 160–167, 2008.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *JMLR*, 11:625–660, 2010.

Ian Goodfellow, Quoc Le, Andrew Saxe, and Andrew Ng. Measuring invariances in deep networks. In *NIPS'09*, pages 646–654. 2009.

Eric Granger, Robert Sabourin, Luiz S. Oliveira, and Catolica Parana. Supervised learning of fuzzy artmap neural networks through particle swarm optimization. *JPRR*, 2(1):27–60, 2007.

P.J. Grother. Handprinted forms and character database, NIST special database 19. In *National Institute of Standards and Technology (NIST) Intelligent Systems Division (NISTIR)*, 1995.

Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

Aapo Hyvärinen. Estimation of non-normalized statistical models using score matching. *JMLR*, 6:695–709, 2005.

Aapo Hyvärinen. Optimal approximation of signal priors. *Neural Computation*, 20(12):3087–3110, 2008.

Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *ICCV'09*. IEEE, 2009.

Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *ICML'07*, pages 473–480. ACM, 2007.

Hugo Larochelle, Yoshua Bengio, Jerome Louradour, and Pascal Lamblin. Exploring strategies for training deep neural networks. *JMLR*, 10:1–40, 2009.

Honglak Lee, Chaitanya Ekanadham, and Andrew Ng. Sparse deep belief net model for visual area V2. In *NIPS'07*, pages 873–880. MIT Press, Cambridge, MA, 2008.

Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML 2009*. Montreal (Qc), Canada, 2009a.

Honglak Lee, Peter Pham, Yan Largman, and Andrew Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *NIPS'09*, pages 1096–1104. 2009b.

J. Milgram, M. Cheriet, and R. Sabourin. Estimating accurate multi-class probabilities with support vector machines. In *Int. Joint Conf. on Neural Networks*, pages 906–1911, 2005.

Hossein Mobahi, Ronan Collobert, and Jason Weston. Deep learning from temporal coherence in video. In *ICML 2009*, pages 737–744, Montreal, June 2009. Omnipress.

L.S. Oliveira, R. Sabourin, F. Bortolozzi, and C.Y. Suen. Automatic recognition of handwritten numerical strings: a recognition and verification strategy. *IEEE Trans. Pattern Analysis and Mach. Intelli.*, 24 (11):1438–1454, 2002.

Juan Carlos Pérez-Cortes, Rafael Llobet, and Joaquim Arlandis. Fast and accurate handwritten character recognition using approximate nearest neighbours search on large databases. In *IAPR*, pages 767–776, London, UK, 2000. Springer-Verlag.

Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. Self-taught learning: transfer learning from unlabeled data. In *ICML 2007*, pages 759–766, 2007.

Rajat Raina, Anand Madhavan, and Andrew Y. Ng. Large-scale deep unsupervised learning using graphics processors. In *ICML 2009*, pages 873–880, New York, NY, USA, 2009. ISBN 978-1-60558-516-1.

M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun. Efficient learning of sparse representations with an energy-based model. In *NIPS'06*, 2007.

Marc'Aurelio Ranzato, Y-Lan Boureau, and Yann LeCun. Sparse feature learning for deep belief networks. In *NIPS'07*, pages 1185–1192, Cambridge, MA, 2008. MIT Press.

Ruslan Salakhutdinov and Geoffrey E. Hinton. Deep Boltzmann machines. In *AISTATS'2009*, volume 5, pages 448–455, 2009.

Graham Taylor and Geoffrey Hinton. Factored conditional restricted Boltzmann machines for modeling motion style. In *ICML 2009*, pages 1025–1032, Montreal, June 2009. Omnipress.

Graham Taylor, Leonid Sigal, David Fleet, and Geoffrey Hinton. Dynamic binary latent variable models for 3D pose tracking. In *Proc. CVPR'10*, 2010.

P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML 2008*, 2008.

Pascal Vincent. A connection between Score Matching and Denoising Autoencoders. Technical Report 1359, Universite de Montreal, 2010.

J. Weston, F. Ratle, and R. Collobert. Deep learning via semi-supervised embedding. In *ICML 2008*, 2008.