Combining Additive Input Noise Annealing and Pattern Transformations for Improved Handwritten Character Recognition

J.M. Alonso-Weber jmaw@ia.uc3m.es

M.P. Sesmero msesmero@inf.uc3m.es

A. Sanchis masm@inf.uc3m.es

Department of Informatics

Polytechnic School of Engineering

Universidad Carlos III de Madrid

Avenida de la Universidad 30

Leganés 28911, Madrid (Spain)

ABSTRACT Two problems that burden the learning process of Artificial Neural Networks with Back Propagation are the need of building a full and representative learning data set, and the avoidance of stalling in local minima. Both problems seem to be closely related when working with the handwritten digits contained in the MNIST dataset. Using a modest sized ANN, the proposed combination of input data transformations enables the achievement of a test error as low as 0.43%, which is up to standard compared to other more complex neural architectures like Convolutional or Deep Neural Networks.

Keywords Artificial Neural Networks - Back Propagation – MNIST – Handwritten Text Recognition

1 Introduction

Handwritten text recognition is a demanding problem for which ANNs are well suited learning models, as is shown by ongoing research. There are multiple technological applications that require a more or less robust ability to perform handwritten text recognition, for example the validation of signatures in banks, the entry of text in mobile devices or for large-scale digitizing and archival of manuscripts (Plamondon & Srihari, 2000).

The complexity of the text recognition varies greatly depending on the type of the context or the required application. The recognition of handwritten text always demands a more sophisticated approach that in the case of machine written texts. There is a greater difficulty in recognizing writer independent unconstrained continuous handwritten text than for texts written with printed isolated letters.

Some applications such as the identification of signatures in banks use a different approach as they rely on touch-pads that allow obtaining the data describing the sequence followed during the writing process, in what is called on-line recognition. In this case there is information available in real time on the strokes that make up the text, i.e., the direction, angle, speed and pressure.

The biggest challenge in text recognition arises when addressing the off-line variant, which operates on digitized raster images and lacks information about the strokes, sequencing and timing of the writing process.

The digitized image will be contained in a plane, where each point represents a pixel with a variable intensity between the white paper and the dark ink strokes. Such collection of pixels represents a high dimensional and complex set of data with regard to the recognition of characters. This is the reason for which the off-line recognition techniques have to resort to a series of previous processes that can transform the images into a more usable set of data. These processes involve several tasks as binarization, contrast and brightness adjustments, skeletonization, and noise removals. A crucial and complex task prior to the recognition is to segment each of the paragraphs, lines and characters contained in the image (Alonso-Weber, Galván, & Sanchis, 2003; Alonso-Weber & Sanchis, 2011; Lacerda & Mello, 2013), which may require some feedback from the recognition stage (Fernández-Caballero, López, & Castillo, 2012).

Mainstream research deals with the problem of extracting and using explicit information from the high dimensional image data. The purpose of this Feature Extraction is two-fold. At first hand, the interest is to extract structured and relevant information from the high volume of data available in an image, and to dismiss redundant or useless information. The second purpose is to reduce the dimensionality of the data, which can help to find better solutions and decreases the computational costs associated to the learning and test processes.

A different approach deals with implicit feature extraction, and the recognition is performed directly on the original, unprocessed image data. A machine learning method is applied and expected to extract the relevant information from the raw data. Here, several ANN models have shown to be particularly efficient, achieving rather low error rates such as a 0.4% and 0.39% for Convolutional Neural Networks (Ranzato, Poultney, Chopra, & LeCun, 2006; Simard, Steinkraus, & Platt, 2003), 0.35% for Deep Neural Networks (Ciresan, Meier, Gambardella, & Schmidhuber, 2010), or even 0.27% and 0.23% for Committees of Convolutional Neural Networks (Ciresan, Meier, Gambardella, & Schmidhuber, 2011; Ciresan, Meier, & Schmidhuber, 2012).

Our proposal follows the raw recognition approach, and is based on a modestly sized ANN with two hidden layers containing 300x200 cells. This is considerably smaller than most of the ANN architectures shown in the literature. The training is performed with the standard Back Propagation Algorithm. We show that good results can be achieved boosting the training process with a combination of several input pattern transformations. These transformations comprise a line-up of affine transformations including a trapezoidal deformation, a dimensionality reduction through image downsizing, and an annealed input noise addition schema. Previous work of the authors involved handwritten recognition through skeletal structures and had a limited accuracy result due to the usual problems of premature stalling of the Back Propagation algorithm (Alonso-Weber & Sanchis, 2011).

The performance of our proposal is validated against the MNIST dataset (LeCun, Bottou, Bengio, & Haffner, 1998; LeCun & Cortes, 1998), that has been used thoroughly over the time to develop increasingly refined recognition algorithms.

The next section contains a review of the work related to our proposal, which is presented in section three, whereas the experimental setup and evaluation are shown in section four.

2 Related Work

As it has been already mentioned, there are several ANN models that have shown to be particularly efficient performing the handwritten text recognition starting from the raw images. Due to the complexity of the task, ANNs with multiple layers are used, because it is expected that the first layers are able to detect the most basic features, while the subsequent layers construct higher level feature detectors based on the detected features from the precedent layers.

Several proposals have been made, including Convolutional Neural Networks (CNN) and Deep Neural Networks (DNN), which are now described. We also depict here several issues that are related with the proposal of this paper, such as pattern deformations, input noise addition and dimensionality reduction.

2.1 Convolutional and Deep Neural Networks

Convolutional Neural Networks (LeCun et al., 1998) are biologically inspired models that can be traced back (Fernandes, Cavalcanti, & Ren, 2013) to the work of (Hubel &

Wiesel, 1962) and Fukushima"s Neocognitron (Fukushima, 1980). In a CNN each cell in the first layer has a set of inputs (receptive fields) coming from a limited region of the input space (retinotopic region). These cells are arranged in a tile structure covering the whole input space. The function of these cells is to perform a simple filtering i.e., a feature detection based on the high correlation between neighbouring input pixels, which is called convolutional mapping. The cells in the subsequent layers replicate the same structure, with their inputs covering a small region of the preceding layer. The stacking of these retinotopic mappings builds progressively more complex feature detectors that cover a greater input region.

Layers with a different functionality can be alternated: a convolutional layer can be cascaded with a so called *max-pooling* layer that performs a subsampling of the detected features. For the final output a conventional fully-connected layer can be used.

A specific constraint joint to the max-pooling layers allows these detectors to develop invariance to the position of the features. This is accomplished using neighbouring cells that share the same weights, i.e. they perform the same feature detection with a slight displacement in the input space.

This constraint and the sparse connectivity of the model have associated a reduced parameter space, where the search for an optimal weight configuration can be performed in a more efficient way than in a conventional ANN with a fully connected architecture.

Some applications of CNNs on the MNIST dataset have reported very low error rates: such as a 0.4% for (Simard et al., 2003) and 0.39% for (Ranzato et al., 2006).

Training ANNs with Back Propagation has a serious inconvenience when several hidden layers are used. The errors fade (Hochreiter & Munchen, 1998) when they are propagated back through multiple layers, and this hinders a proper learning. Several authors proposed alternative algorithms to overcome this inconvenience. Hinton (Hinton, 2007) and Bengio (Bengio & Lamblin, 2007) proposed unsupervised methods for training independently each layer of a Deep Belief Neural Network, with a subsequent fine tuning with a supervised learning algorithm. Based on these concepts (Ciresan et al., 2010) proposed a Deep Neural Network with error rates as low as 0.35%.

There are other posterior works that show even lower error rates as 0.27% (Ciresan et al., 2011) and 0.23% (Ciresan et al., 2012) are based on combining several CNNs into additive ensembles or committees. These ensembles can achieve an improvement in the accuracy in an additional 0.1% (at these extreme performance ratios) with respect to the underlying neural model.

In practice, the multilayer models are extremely profuse in the number of layers, cells and weights, even in spite of the sparse connections and the weight sharing of the CNNs. This leads to one of their major inconveniences, which is the need of a very high computational power in order to achieve reasonable training times. In the case of (Ciresan et al., 2010) the DNN with the best performance has 7500 cells distributed in five hidden layers that are fully connected, and about 12.000.000 connections. The use of Graphical Processing Units (GPU) allows a considerable boosting in the processing times (Ciresan et al., 2010) with a reported speedup of 40x. These GPUs have allowed exploring the possibilities of the DNNs, but require more elaborate programs for the simulations.

2.2 Pattern Deformations

Although the use of DNNs and CNNs may help to find good solutions, another problem that they do not avoid is the need of using a very large set of training samples. Even datasets like the MNIST that contains 60000 training instances fall short for a satisfactory training.

A usual strategy is extending the training set performing some kind of changes on the images. These changes are mostly affine transformations (Yaeger, Lyon, & Webb, 1996) like displacements, rotations and linear deformations. Simard (Simard et al., 2003) proposes a distortion process based on elastic deformations which tries to reproduce the natural uncontrolled oscillations of the writer"s hand. Two parameters allow adjusting these distortions.

The deformation helps the learning process to develop a recognition ability that is more invariant to changes in the input data of the test, and reduces the error rate.

2.3 Input Noise Addition, Weight Decay and Regularization

Another problem that weights down the performance of Back Propagation is the fact that convergence usually stalls in local minima, often far away from a reasonable working point.

Several perturbation techniques have been used in the past in order to improve the convergence and the generalization abilities of ANN: for example noise injection adds small random values of noise to weights (Abunawass & Owen, 1993), to node activations, including output perturbations (Wang & Principe, 1999).

Injecting additive input noise (Matsuoka, 1992; Sietsma & Dow, 1991) is a particular case which has been related (An, 1996; Grandvalet, Canu, & Boucheron, 1997) to other well known learning techniques as weight decay and kernel regression estimation or Tikhonov regularization (Bishop & Avenue, 1995; Holmstrom & Koistinen, 1992; Koistinen & Holmstrom, 1991).

Regularization tries to accomplish smoother network mappings with a modified error function. A penalty function is added to the sum squared error. This modifies the weight update function. A special case is weight decay where the penalty function is the sum of the squared network weights. This allows a smoother learning because it leads to a more moderate growth of the weights.

Weight decay and regularization are interesting options for avoiding local minima. They are easier to study and analyze than noise injection. A drawback for both is that they have an impact on the computational cost of the Back Propagation algorithm, as they imply the computation of some additional terms in each weight update. For computationally intensive problems such as image processing, this fact cannot be neglected.

Input noise injection is used in different applications (Piotrowski, Rowinski, & Napiorkowski, 2012; Unsworth & Coghill, 2006), being necessary to adjust the parameters of the noise distribution according to the requirements of each input variable. Regarding the computational cost, input noise injection is more efficient than regularization or weight decay. The overhead is the addition of random values to each input value, which are much more reduced in number than the total weight updates for each pattern presentation.

2.4 Dimensionality reduction, subsampling

Dimensionality reduction is a crucial issue in machine learning research. There are many situations where a reduction of the data to be processed is needed (Peteiro-Barral, Bolón-Canedo, Alonso-Betanzos, Guijarro-Berdiñas, & Sánchez-Maroño, 2013). This is the case of the off-line handwritten text recognition in which the raw data from the scanned images has a high dimensionality. It was already mentioned that most research in this domain involves the extraction of relevant features, discarding redundant or useless information, which is a particular case of dimensionality reduction. But also the approach of working directly on the raw data can benefit from a dimensionality reduction.

In a scanned image neighbouring pixels usually have a high correlation between them. If the resolution of the image is high, a convenient approach is subsampling the pixels. The reduced image may retain sufficient information for a correct recognition of its content.

This dimensionality reduction has already been applied to the MNIST problem (LeCun et al., 1998), with an interpolation of the input images from the original 28x28 size down to 16x16. Some layers of the Convolutional NN model perform a sub-sampling of the detected features in the previous layer. But the usual approaches rely more on explicit feature extraction, or on feature selection, rather than on subsampling based on interpolation.

3 Proposed model

Bengio (Bengio & Lamblin, 2007) mentions the long-standing belief that ANNs with multiple hidden layers should not perform better than single layered networks. This belief might be originated by the difficulties of training deep multi-layer neural networks. With the development of newer learning strategies these difficulties disappear, and this has spurred the use of increasingly complex and *deep neural networks*.

Our motivation for the present research is to explore if a relatively small ANN (with two hidden layers with 300x200 cells) is able of achieving similar results to those reported for Deep and Convolutional Neural Networks. Therefore we enhance the learning process with a set of transformations that are applied onto the input instances, and create this way a virtually infinite training data set:

a) We design a new trapezoidal deformation that is combined with the usual linear transformations (rotation, displacement, scaling). This deformation differs from both the affine deformation (Yaeger et al., 1996) and the elastic distortions (Simard et al., 2003) used in a great part of the published research. The experimentation suggests that the proposed deformation is rather effective.
b) The input images are sub sampled with an interpolation method. This was previously used by LeCun (LeCun et al., 1998), and is part of the some layers of the Convolutional Neural Networks. But it has not been used as a regular strategy.
c) An annealed input noise addition schema to help to avoid local minima during the training phase is fundamental for overcoming stalling. Much literature has been devoted to the input noise addition, but the usual approach is the use of a static noise, mostly following a Gaussian distribution. Our proposal is based on combining an annealing scheme with a uniform noise distribution, which shows to be rather more effective than static Gaussian noise. An open question is how this annealed noise scheme relates with weight decay.

We now describe with detail the mentioned processes. At last, the task of finding adequate parameter values is addressed.

3.1 Dimensionality Reduction through subsampling

The MNIST Dataset is a collection of 70000 numerical handwritten symbols which are almost evenly distributed in ten classes („0" – „9"). The dataset is divided into the training set with 60000 images and the test set with 10000 images. Each one of the digitized numerals are represented with a 28x28 sized, 256 graylevel image.

Each image is coded as a pattern with 784 input values and 10 output values. Each input value encodes a pixel value, where 1 represents a black pixel and 0 a white one. The output layer has ten nodes where each node represents one of the ten possible classes.

Different image sizes were tried out with appropriate sized neural networks. The original images where downsized to 14x14 and 20x20 pixels. Differences in the performance are not highly significant (see **Table 1**), but show that there might be an issue with the input data dimensionality. Loss of detail in the downsized digits is compensated with the benefits of lower dimensionality up to a certain performance level.

Another advantage of using downsized images is the lower computational cost. The learning process for the 14x14 sized database lasts typically about 10 hours, whereas the original 28x28 sized images need about 41 hours.

3.2 Image Transformation

Image transformation is performed applying different affine transformations: rotation, translation and scaling, followed by a new trapezoidal deformation

Rotation, translation and scaling:

Given a pixel defined by its lattice coordinates *(i, j)*, the mapping of this pixel in the transformed image is given by $(i, j) \rightarrow (x_{ij}, y_{ij})$:

$$x_{ij} = - j \cdot \sin(-\alpha) / s + i \cdot \cos(-\alpha) / s + \Delta x_{ij} \quad (1)$$

$$y_{ij} = j \cdot \cos(-\alpha) / s + i \cdot \sin(-\alpha) / s + \Delta y_{ij} \quad (2)$$

Parameter α represents the rotation angle selected randomly in the [-*a*, +*a*] interval with a uniform distribution, where *a* is a parameter which limits the maximal rotation. The scale factor is represented with *s*.

Translation is defined by Δx_{ij} and Δy_{ij} :

$$\Delta x_{ij} = R_1(d_x) = \text{sgn}\,(r_{xij})\cdot\text{int}\,(|r_{xij}|^{\gamma} \cdot d_x) \tag{3}$$

$$\Delta y_{ij} = R_1(d_y) = \text{sgn}\,(r_{yij})\cdot\text{int}\,(|r_{yij}|^{\gamma} \cdot d_y) \tag{4}$$

where d_x and d_y are parameters that limit the maximal displacement, and r_{xij} , r_{yij} are uniform distributed random numbers in the [-1, +1] interval.

The purpose of function R_1 is to reduce the incidence of extreme random displacements. For $\gamma = 1$ the function R_1 returns a random value which is uniformly distributed in the $[-d_x, +d_x]$ or $[-d_y, +d_y]$ interval. For values of γ higher than 1, the distribution density increases around the zero values, with fewer extreme values. This transformation works best when the displacements are applied as integer values, therefore the int() function is used.

Trapezoidal Deformation:

The trapezoidal deformation operates on each of the four corner vertex of the lattice containing the original digit, displacing them by some random amount and in a random direction, and stretching or compressing the rest of the lattice in a proportional quantity. The inverse map in Figure 1 shows an example of a deformed lattice.

For each corner vertex $W_h \in \{ (x_A, y_A), (x_B, y_B), (x_C, y_C), (x_D, y_D) \}$ of the nxn sized lattice, which represents each of the coordinates A:(0, 0), B:(n-1, 0), C:(0, n-1), D:(n-1, n-1)}, a displacement is applied such that $(x_h, y_h) \rightarrow (u_h , v_h)$,

$$u_h = x_h + R_2(b_x) \tag{5}$$

$$v_h = y_h + R_2(b_y) \tag{6}$$

$$R_2(b_x) = \text{sgn}(b_x) \cdot |r_{xh}|^{\beta} \cdot b_x \tag{7}$$

$$R_2(b_y) = \text{sgn}(b_y) \cdot |r_{yh}|^{\beta} \cdot b_y \tag{8}$$

where b_x and b_y are parameters that limit the displacement of each lattice corner, and r_{xh} , r_{yh} are uniform distributed random numbers in the [-1, +1] interval.

The purpose of function R_2 is to reduce the incidence of extreme random deformations. For $\beta=1$ the function R_2 returns a random value which is uniformly distributed in the [-b, +b] interval. For values of β higher than 1, the distribution accumulates around the zero values, with fewer extreme values.

Given the displaced corners, the final position of each pixel in the image lattice can be computed following the displacement scheme $(x_{ij}, y_{ij}) \rightarrow (u_{ij}, v_{ij})$:

$u_{ij} = Dx_{ij} \cdot j + x_A$ (9)

$v_{ij} = Dy_{ij} \cdot j + y_A$ (10)

where Dx_{ij} , Dy_{ij} are the differences that define the slope of the i-th horizontal lattice-line:

$Dx_{ij} = (x_{BDi} – x_{ACi}) / (n-1)$ (11)

$Dy_{ij} = (y_{BDi} – y_{ACi}) / (n-1)$ (12)

where x_{ACi} , y_{ACi} define the i-th point on the lattice line between vertex A-B:

$x_{ACi} = Dx_{AC} \cdot i + x_A$ (13)

$y_{ACi} = Dy_{AC} \cdot i + y_A$ (14)

and x_{BDi} , y_{BDi} define the i-th point on the lattice line between vertex B-D:

$x_{BDi} = Dx_{BD} \cdot i + x_B$ (15)

$y_{BDi} = Dy_{BD} \cdot i + y_B$ (16)

finally, the slope differences Dx_{AC}, Dy_{AC}, and Dx_{BD}, Dy_{BD} for the lattice-lines A-C and B-D are defined:

$Dx_{AC} = (x_C – x_A) / (n-1)$ (17)

$Dy_{AC} = (y_C – y_A) / (n-1)$ (18)

$Dx_{BD} = (x_D – x_B) / (n-1)$ (19)

$Dy_{BD} = (y_D – y_B) / (n-1)$ (20)

Sampling

Applying the sequence of transformations gives a collection of coordinate pairs *(u_{ij}, v_{ij})* which are not integer valued. This means that the corresponding pixel values need to be computed by interpolation. Bilinear sampling is an easy way to undertake this task. Since each one of mentioned transformations degrades the image quality, the operation sequence should be done in a single shot, rather than applying them independently on the image.

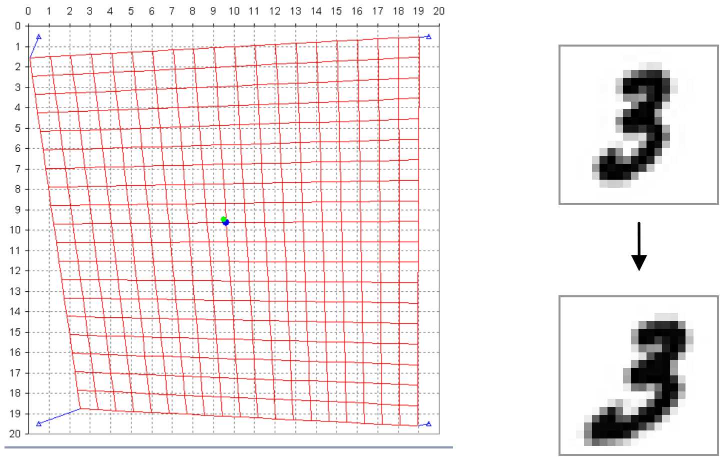

Figure 1 Inverse mapping (left) for a rotation and deformation. When applied to the original digit (right above) gives the deformed version (right below).

3.3 Additive Input Noise Annealing

Input noise is fed into the input layer at the time each pattern is presented to the network.

For each pixel value V_{ij} :

$$V''_{ij} = V_{ij} + \varepsilon_{ij} \cdot Q(g, T, t) \tag{21}$$

$$Q(g, T, t) = g - t \cdot T \tag{22}$$

where ε_{ij} is a uniform random number in the [0, +1] interval, and $Q(g, T, t)$ defines an annealing schema that subdues the noise influence with increasing simulation time t. The parameters g and T define the profile of the annealing schema. A typical value set for a $t_{max} = 1000$ iteration learning is $g = 1$, $T = 1/ t_{max}$, which implies a rather hefty noise injection for the initial iterations that gets dimmed towards the end of the learning process.

3.4 Parameter values

Finding the adequate parameter values requires some prospecting experimentation. Applying each transformation individually has a limited influence on the ANNs performance. In such cases, finding a good working point requires higher parameter values than in combination with other transformations. In order to shortcut an undesirable lengthy search in the parameter space, a good approach relies on exploring at first the transformation with greatest influence and gradually introducing the rest (following this order: deformation, translation, rotation, and scaling).

For the combined transformations a good maximal rotation value is *a = 0.15*, which is valid for all the digit sizes. For 20x20 sized digits good translation parameter values are $d_x = d_y = 3.2$, $\gamma = 2$, and the deformation process requires: $b_x = b_y = 3.5$, $\beta = 1$. For the 28x28 sized images the parameters that need adaptation are $d_x = d_y = 4.5$, and $b_x = b_y = 5$.

The scale factor should be limited to random values between [0.99, 1.1], or can be fixed to 1 since the deformation process has a strong influence on the image scale.

The parameter exploration was not performed in an exhaustive way, so some room for further improvement can be expected.

Figure 2 shows a selection of transformations (rotation, translation and deformation) applied on the same input digit.

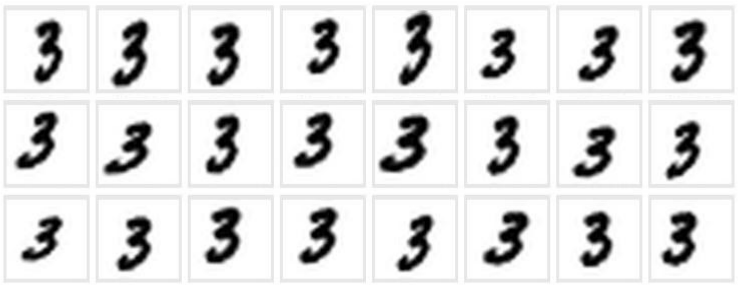

Figure 2 A sample containing different transformations of the same original digit (rotation, translation, deformation).

4 Experimental Evaluation

Initial experimentation served for determining some generic parameter values needed for the experimental evaluation. Throughout the extensive experimentation of this work these parameter values showed to be rather stable, which means that no circumstances entailed a change of these values, and no dependencies between parameters were stated.

A Neural Network with two hidden layers and size 784x300x200x10 was determined to be a good option. In the course of the experimentation, it was stated that changing the input data size (14x14, 20x20 and 28x28) had no influence on the needed network layers and hidden cells. ANNs with a single layer were discarded because they were prone to premature stalling with an increasing size of the hidden layer.

The weight values were initialized with uniform distributed random values in the [-

0.3, +0.3] interval. Best results were obtained with a Learning Rate η valued at *0.03*.

Training of the ANN follows the usual cycle sequence until some convergence criterion is reached. At each cycle all the training patterns are transformed with our deformation and noise processing, and presented to the ANN in a randomized order. This extends the input data set into a virtually infinite training set.

The experimental evaluation arises some issues which are how to evaluate the performance and when to stop the learning process. In order to evaluate the generalization capability the test set should not be used directly for this purpose. The first option is using a third validation set whose aim is determining the optimal working point of the ANN during the learning process. Therefore the original training set is split into a new training set with 50000 patterns, and a validation set with the remaining 10000. Another option is assuming that the optimal configuration is achieved in the last training cycle.

Another issue is the training length. Conventional Back Propagation training without noise annealing shows a faster convergence but gets stalled in local minima. Extending the learning process beyond 500 cycles (for example) improves slightly but with a very low cost-effectiveness ratio. Training with input noise annealing requires extending the learning process until the input noise is near zero. Beyond this point the improvement rate is near zero. As the selected noise annealing scheme requires 1000 cycles for experimental coherence we have chosen all the simulations to last the same cycle number (1000 cycles).

All performance values are computed as the mean value over series of ten experiments.

A first comparison is done with the object of determining the performance of plain Back Propagation using the MNIST dataset without any transformations. Results are shown in **Table 1**.

Input image size			
28x28	20x20	14x14	Test Error [%] based on:
1.79	1.63	**1.50**	Last Iteration
1.82	1.66	**1.51**	Validation

Table 1 Errors on the MNIST test set without any transformations.

Here, best results are obtained for the 14x14 sized images. Differences for the distinct image sizes are not exceedingly high, but are an indication about the difficulty of finding good solutions when the input data has a high dimensionality. Test errors based on the validation stopping criterion are higher than those at the last iteration. This behaviour was already stated in (Simard et al., 2003). This can be due to the fact that extending the learning process of the ANN indefinitely never worsens the results on the test set. The stopping point determined with the validation set occurs usually about 200 cycles before the last iteration. So, some loss of the accuracy can be expected. Therefore we evaluate the accuracy based only on the errors at the last iteration.

The next step involves comparing the performance when the input data are transformed with annealed noise or affine transformations. The corresponding results are shown in Table 2.

Individual Transformations	Input image size			
	28x28	20x20	14x14	Test Error [%] on:
Input Noise Annealing	1.03	**0.96**	**0.96**	Last iteration
Rotation+Translation+Deformation	0.55	**0.53**	0.66	
Input Noise Annealing	1.05	0.98	**0.97**	Validation
Rotation+Translation+Deformation	**0.58**	0.60	0.66	

Table 2 Errors on the MNIST test set applying individual transformations.

Applying the deformation combined with the usual rotation and translation has a high influence on the performance reducing the error rate in more than a 1% in relation to the standard Back Propagation learning (Table 1). The test errors for 28x28 sized images (0.55%) are even lower than those published by Simard for elastic distortions combined with 2 layer MLPs (0.7%-0.9%) (Simard et al., 2003). Input Noise Annealing has also a respectable influence, reducing the test error in about 0.5% - 0.7% depending on the image size.

The next step involves comparing the performance when the input data are processed with input noise combined with the transformations. Table 3 shows a better performance for the 20x20 sized images. The 14x14 images have the worst results. This might be due to an insufficient resolution, which is more patent for the few remaining conflictive patterns.

Combined Transformations	Input image size			
	28x28	20x20	14x14	Test Error [%] on:
Input Noise Annealing + Rotation + Translation + Deformation	0.51	**0.43**	0.56	Last iteration
	0.53	**0.45**	0.59	Validation

Table 3 Errors on the MNIST test set applying all transformations.

Adding the input noise annealing to the transformation sequence reduces the error less than 0.1% (about 9 mismatches) for the 20x20 sized digits. Considering it the other way round, the addition of the transformation line-up to the noise annealing reduces the error rate in about 0.45% - 0.55% depending on the image size. Contribution of the input noise annealing seems low in the total test error. We conjecture is that the action of the input noise annealing and the image transformation do overlap.

Figure 3 shows the mismatched digits for a minimal configuration found during the extensive experimentation (for an experiment with 20x20 sized images).

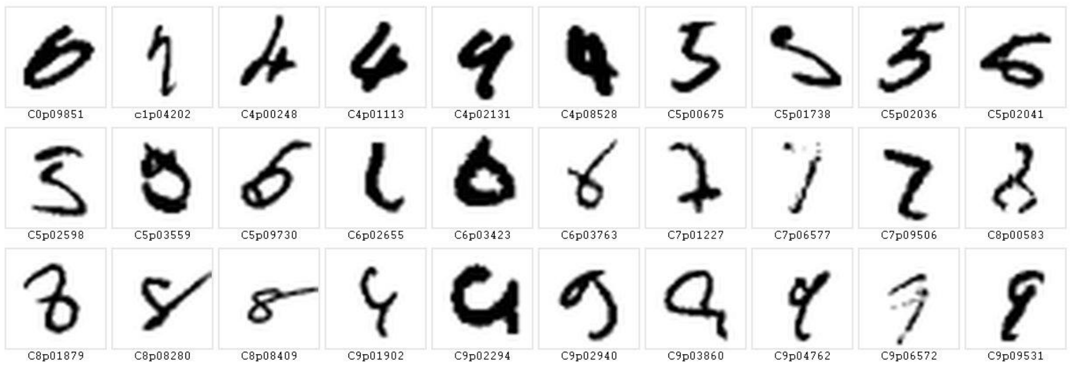

Figure 3 Mismatched digits for the best configuration found (30 failures)

A last comparison involves testing the invariance of the trained ANN to noise in the test patterns. Since the proposed training is based on a noise injection, the conjecture is that the resulting learned ANN should be rather robust.

For this purpose the test set with the 20x20 sized digits is transformed in two different ways. At first, several new test sets are generated which contain wiped out pixels (set to white value) up to a certain proportion (at 10%, 20%, up to 50%). A second collection contains different test sets where a certain proportion of pixels are changed by random noise (in the [0, 1] interval). A sample is shown in **Figure 4**.

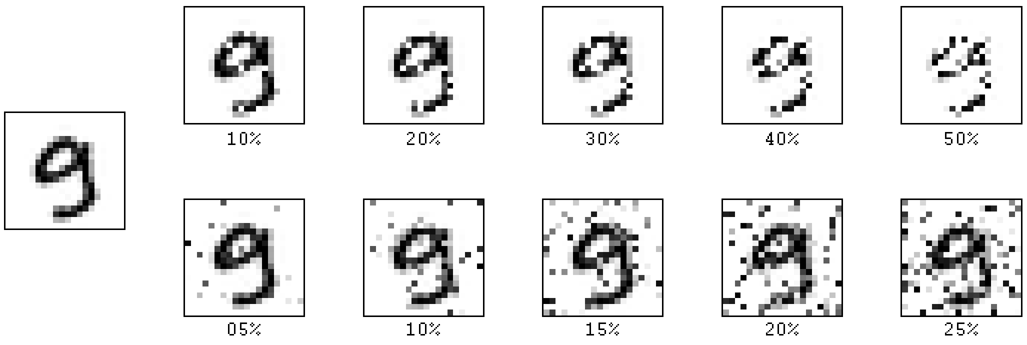

Figure 4 Noise Tests. Original test image (left) and successive versions with wiped out pixels (above) at 10%-50% proportion, and with randomized pixels at 5%-25% rates (below).

All the ANNs were trained with the usual Pattern Transformations and confronted with and without Noise Annealing.

The results for the wiped out pixels are shown in **Table 4**. ANNs trained with noise injection show a respectable resistance in the test set. Test errors degrade to a 20% in the last iteration when half of the pixels are wiped out. Training without noise annealing, leads to a degraded performance, increasing the error rate by a factor between 1.5 and 1.7.

Wiped out pixels	*0%*	10%	20%	30%	40%	50%	Test Error [%] on:
With Noise Annealing	*0.43*	0.99	2.32	5.23	11.10	19.82	Last Iteration
Without Noise Annealing	*0.53*	1.47	3.82	8.84	17.74	29.66	
With Noise Annealing	*0.45*	1.10	2.92	6.79	14.32	24.33	Validation
Without Noise Annealing	*0.60*	1.47	3.95	9.01	17.63	28.91	

Table 4 Errors on the MNIST test set with wiped out pixels. The ANNs were trained with the usual Pattern Transformations, and with or without the Noise Annealing on the 20x20 sized dataset. Cursive values match those obtained in Table 2 and Table 3 and are included for reference.

The results for the randomized pixels are shown in **Table 5**. Test errors degrade to a 16% in the last iteration when a 25% of the pixels are randomized. Training without noise annealing, leads to a considerable degraded performance, increasing the error rate by a factor between 3 and 6.

Randomized pixels	*0%*	5%	10%	15%	20%	25%	Test Error [%] on:
With Noise Annealing	*0.43*	0.76	1.71	4.22	11.25	15.73	Last Iteration
Without Noise Annealing	*0.53*	2.25	9.97	20.64	34.65	45.56	
With Noise Annealing	*0.45*	0.89	2.61	6.74	15.70	24.00	Validation
Without Noise Annealing	*0.60*	2.31	9.94	22.40	35.66	46.61	

Table 5 Errors on the MNIST test set with randomized pixels. The ANNs were trained with the usual Pattern Transformations, and with or without the Noise Annealing on the 20x20 sized dataset. Cursive values match those obtained in Table 2 and Table 3 and are included for reference.

In order to show how well our method matches against previous proposals, we list in Table 6 the results for different works based on ANNs. The proposal of this paper compares very favourable in terms of accuracy and network size (considering the number of connections) against the others. All the proposed versions, even without the Input Noise Annealing, outperform the rest.

Classifier Type	**(Inputs) Hid - … - Hid - Out**	**Deform.**	**Additional Processing**	**%Test**	**Reference**
ANN	(28x28) 1000-10	Yes	None	3.80	LeCun et al. 1998
ANN	(28x28) 300-10	Yes	None	3.60	LeCun et al. 1998
ANN	(28x28) 300-100-10	Yes	None	2.50	LeCun et al. 1998
ANN	(28x28) 500-150-10	Yes	None	2.45	LeCun et al. 1998
ANN	(29x29) 800-10	None	None	1.60	Simard et al. 2003
ANN	(29x29) 800-10	Affine	None	1.10	Simard et al. 2003
ANN	(28x28) 500-500-2000-30	Elastic	None	1.00	Salakhutdinov&Hinton 2007
ANN	(29x29) 800-10	Elastic	None	0.70	Simard et al. 2003
ANN	(14x14) 300-200-10	Trapez.	None	**0.66**	**[this paper]**
ANN	(14x14) 300-200-10	Trapez.	Input Noise annealing	**0.56**	**[this paper]**
ANN	(28x28) 300-200-10	Trapez.	None	**0.55**	**[this paper]**
ANN	(20x20) 300-200-10	Trapez.	None	**0.53**	**[this paper]**
ANN	(28x28) 300-200-10	Trapez.	Input Noise annealing	**0.51**	**[this paper]**
ANN	(20x20) 300-200-10	Trapez.	Input Noise annealing	**0.43**	**[this paper]**

Table 6 Comparison of different proposals based on ANNs. Sorted by decreasing error on the MNIST test set. Source: http://yann.lecun.com/exdb/mnist/

Table **7** summarizes the results for different works based on CNNs and DNNs. The proposal of this paper performs better than a good part of the works. More recent CNNs and DNNs have better accuracies. Ensembles are able to deliver about 10 errors fewer (a 0.1%) than their base members. This fact should be taken into account when comparing the different proposals.

Classifier Type	(Inputs) Hid … x Hid x Out [connections (free)]	Deform.	Committee	Additional Processing	%Test	Referente
Conv. NN	(16x16) [~100000 {~2600}]	None		Subsampling to 16x16 pixels	1.70	LeCun 1998
Conv. NN	(28x28) [~260000 {~17000}]	None		None	0.95	LeCun 1998
Conv. NN	(28x28) [~260000 {~17000}]	Huge		None	0.85	LeCun 1998
Conv. NN	(28x28) [~260000 {~17000}]	Yes		None	0.70	LeCun 1998
Conv. NN	(28x28) 50-50-50-50-200-10	None		None	0.60	Ranzato 2006
Conv. NN	(29x29)	Affine		None	0.60	Simard 2003
ANN	(14x14) 300-200-10 [121310]	Trapez.		Input Noise annealing	**0.56**	**[this paper]**
Conv. NN	(28x28)	None		None	0.53	Jarrett 2009
ANN	(28x28) 300-200-10 [297710]	Trapez.		Input Noise annealing	**0.51**	**[this paper]**
Deep NN	(28x28) 1000-500-10 [~1340000]	Elastic		None	0.49	Ciresan 2010
Deep NN	(28x28) 1500-1000-500-10 [~3260000]	Elastic		None	0.46	Ciresan 2010
ANN	(20x20) 300-200-10 [182510]	Trapez.		Input Noise annealing	**0.43**	**[this paper]**
Deep NN	(28x28) 2000-1500-1000-500-10 [~6690000]	Elastic		None	0.41	Ciresan 2010
Conv. NN	(29x29)	Elastic		None	0.40	Simard 2003
ANN	*(var) 800-10*	*Elastic*	***Yes, 25x***	*Width normalization deslanting*	*0.39*	*Meier 2011*
Conv. NN	(28x28) 50-50-50-50-200-10	Elastic		None	0.39	Ranzato 2006
Deep NN	(28x28) 2500-2000-1500-1000-500-10 [~12110000]	Elastic		None	0.35	Ciresan 2010
Conv. NN	(var) 1-20-30-40-60-80-100-120-120-10	Elastic		None	0.35	Ciresan 2011
Conv. NN	*(var) 1-20-P-40-P-150-10*	*Elastic*	***Yes, 7x***	*width normalization*	*0.27*	*Ciresan 2011*
Conv. NN	*(var) 1-20-P-40-P-150-10*	*Elastic*	***Yes, 35x***	*width normalization*	*0.23*	*Ciresan 2012*

Table 7 Comparison of the proposed model with DNNs and CNNs. Sorted by decreasing error on the MNIST test set. Source: http://yann.lecun.com/exdb/mnist/

Simulations were performed on several workstations with dedicated Intel Xeon E5520 CPU at 2.27 GHz, having each several core-processors which allow to process up to 8 concurrent simulations. A specific Back Propagation simulator with vector oriented data structures for a better vector processing optimization was designed. Each simulation lasted about 10 hours for the downsized MNIST dataset (14x14), about 20 hours for the 20x20 sized images, and up to 41 hours for the original dataset. Image pre-processing was performed using *xnview*.

5 Conclusions and Future Work

Achieving successive improvements in the recognition rate of handwritten text has a two-fold purpose: on the one hand, very low error rates are needed for standalone applications for automated recognition, because even error rates of 1% or lower imply the need of human supervision. On the other hand, this constitutes an excellent benchmark for pushing on step further the learning capabilities of a particular model.

Our research shows that a relatively small ANN is able of reaching a similar performance to those reported for more complex architectures. A crucial strategy for this achievement is an appropriate set of input transformations.

A more general conclusion is that the “raw recognition” method is a valid approach. No prior knowledge about the images and the expected features has to be explicitly stated for achieving good recognition rates.

The Input Noise Annealing technique reveals to be useful in order to avoid the frequent stalling in local minima which is typical for the Back Propagation algorithm. On a standalone basis it reduces the test error down to a 1%. Another key strength appears with noisy test patterns, to which this model is more tolerant.

The possible equivalence of the Noise Annealing with other techniques such as weight decay or regularization has to be checked, but the first has an advantage for a start, because it implies a much lower computational cost. Adding a random value to 28x28 input values has a lower impact in the processing time rather than computing an extra term for each weight update.

A drawback of this Noise Annealing is the extension in time of the learning process. The required 1000 cycles are compensated in part through the low dimensions of the proposed neural network. Once the learning process has finished, the working of the ANN is rather fast, moreover as it has a relatively small architecture.

The proposed deformation of the input patterns combined with the usual affine transformations (rotation and translation) increases the robustness of the training set. Without input noise annealing a test error of 0.55% is achieved, which is lower than the 0.7%-0.9% stated in (Simard et al., 2003) for the now widely used elastic distortion.

Further improvement is achieved through Dimensionality Reduction. The subsampling the input space from 28x28 to 20x20 pixels with an interpolation technique reduces the computational cost and leaves the test error rate at 0.43%. This value compares very favourably against other similar methods, and does not lie to far away from those stated for Convolutional (0.39%) and Deep Neural Networks (0.35%).

We consider that the original MNIST dataset has a low instance to input dimension rate. Extending the training set through pattern deformations is effective, but to a certain limit. Reducing the input dimensionality with an interpolation method decreases the size of the parameter space, which in turn helps the gradient descent to be more effective. Here the limit is the need to preserve a minimal image resolution for a correct recognition.

Applying these input data transformations is relatively easy: no alternative neural architectures or learning procedures are required. Incidentally, other models may benefit from the proposed transformations.

The proposed deformation of the input patterns is different from the known affine deformations and the elastic distortions. The latter was designed to simulate the natural uncontrolled oscillations of the hand of the writer. Our proposal lacks the ability of inducing several local deformations in the image, so we intend to enhance the deformation in this aspect.

Other pending questions are how the Input Noise Annealing is related with other techniques such as weight decay or ridge regression, and to check whether some type of regularization is able to further improve the learning and generalization capacities of classic ANNs.

Acknowledgements

This research reported has been supported by the Spanish MICINN under projects TRA2010-20225-C03-01, TRA 2011-29454-C03-02, and TRA 2011-29454-C03-03.

6 References

Abunawass, A. M., & Owen, C. B. (1993). Statistical Analysis of the Effect of Noise Injection During Neural Network Training. In *Proc. SPIE 1966, Science of Artificial Neural Networks II* (pp. 362–371). doi:10.1117/12.152635

Alonso-Weber, J. M., Galván, I. M., & Sanchis, A. (2003). Modified Self-Organizing Maps for Line Extraction in Digitized Text Documents. In *The 21st IASTED International Multi-Conference on Applied Informatics (AI 2003)* (pp. 281–286). Innsbruck.

Alonso-Weber, J. M., & Sanchis, A. (2011). A Skeletonizing Reconfigurable Self-Organizing Model: Validation Through Text Recognition. *Neural Processing Letters*, *34*(1), 39–58. doi:10.1007/s11063-011-9182-0

An, G. (1996). The Effects of Adding Noise During Backpropagation Training on a Generalization Performance. *Neural Computation*, *674*, 643–674.

Bengio, Y., & Lamblin, P. (2007). Greedy Layer-Wise Training of Deep Networks. In B. Schölkopf, J. Platt, & T. Hoffman (Eds.), *Advances in Neural Information Processing Systems 19* (Vol. 19, pp. 153–160). MIT Press.

Bishop, C. M., & Avenue, J. J. T. (1995). Training with Noise is Equivalent to Tikhonov Regularization. *Neural Computation*, *7*(1), 108–116.

Ciresan, D. C., Meier, U., Gambardella, L. M., & Schmidhuber, J. (2010). Deep, Big, Simple Neural Nets for Handwritten. *Neural Computation*, *22*(12), 3207–3220.

Ciresan, D. C., Meier, U., Gambardella, L. M., & Schmidhuber, J. (2011). Convolutional Neural Network Committees for Handwritten Character Classification. In *2011 International Conference on Document Analysis and Recognition* (Vol. 10, pp. 1135–1139). Ieee. doi:10.1109/ICDAR.2011.229

Ciresan, D. C., Meier, U., & Schmidhuber, J. (2012). Multi-column Deep Neural Networks for Image Classification. In *IEEE Conf. on Computer Vision and Pattern Recognition CVPR 2012* (pp. 3642–3649).

Fernandes, B. J. T., Cavalcanti, G. D. C., & Ren, T. I. (2013). AutoAssociative Pyramidal Neural Network for one class pattern classification with implicit feature extraction. *Expert Systems with Applications*, *40*(18), 7258–7266. doi:10.1016/j.eswa.2013.06.080

Fernández-Caballero, A., López, M. T., & Castillo, J. C. (2012). Display text segmentation after learning best-fitted OCR binarization parameters. *Expert Systems with Applications*, *39*(4), 4032–4043. doi:10.1016/j.eswa.2011.09.162

Fukushima, K. (1980). Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics*, *36*(4), 193–202. doi:10.1007/BF00344251

Grandvalet, Y., Canu, S., & Boucheron, S. (1997). Noise Injection: Theoretical Prospects. *Neural Computation*, *9*(5), 1093–1108. doi:10.1162/neco.1997.9.5.1093

Hinton, G. (2007). Learning multiple layers of representation. *Trends in Cognitive Sciences*, *11*(10), 428–34. doi:10.1016/j.tics.2007.09.004

Hochreiter, S., & Munchen, T. U. (1998). THE VANISHING GRADIENT PROBLEM DURING LEARNING. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *2*, 107–116. Retrieved from http://www.bioinf.jku.at/publications/older/2304.pdf

Holmstrom, L., & Koistinen, P. (1992). Using Additive Noise in Back-Propagation Training. *IEEE Transactions on Neural Networks*, *3*(1), 24–38. doi:10.1109/72.105415

Hubel, B. Y. D. H., & Wiesel, A. D. T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat"s visual cortex. *Journal of Physiology*, *160*, 106–154.

Jarrett, K., Kavukcuoglu, K., Ranzato, M. A., & LeCun, Y. (2009). What is the best multi-stage architecture for object recognition? *2009 IEEE 12th International Conference on Computer Vision*, 2146–2153. doi:10.1109/ICCV.2009.5459469

Koistinen, P., & Holmstrom, L. (1991). Kernel Regression and Backpropagation Training with Noise. In *International Joint Conference on Neural Networks* (Vol. 1, pp. 367–372). doi:10.1109/IJCNN.1991.170429

Lacerda, E. B., & Mello, C. a. B. (2013). Segmentation of connected handwritten digits using Self-Organizing Maps. *Expert Systems with Applications*, *40*(15), 5867–5877. doi:10.1016/j.eswa.2013.05.006

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. doi:10.1109/5.726791

LeCun, Y., & Cortes, C. (1998). THE MNIST DATABASE of handwritten digits. Retrieved from http://yann.lecun.com/exdb/mnist/

Matsuoka, K. (1992). Noise Injection into Inputs in Back-Propagation Learning. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETIC*, *22*(3), 436–440.

Meier, U. U., Ciresan, D. C., Gambardella, L. M., & Schmidhuber, J. (2011). Better Digit Recognition with a Committee of Simple Neural Nets. *2011 International Conference on Document Analysis and Recognition*, *1*, 1250–1254. doi:10.1109/ICDAR.2011.252

Peteiro-Barral, D., Bolón-Canedo, V., Alonso-Betanzos, a., Guijarro-Berdiñas, B., & Sánchez-Maroño, N. (2013). Toward the scalability of neural networks through feature selection. *Expert Systems with Applications*, *40*(8), 2807–2816. doi:10.1016/j.eswa.2012.11.016

Piotrowski, A. P., Rowinski, P. M., & Napiorkowski, J. J. (2012). Comparison of evolutionary computation techniques for noise injected neural network training to estimate longitudinal dispersion coefficients in rivers. *Expert Systems with Applications*, *39*(1), 1354–1361. doi:10.1016/j.eswa.2011.08.016

Plamondon, R., & Srihari, S. N. (2000). Online and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(1), 63–84. doi:10.1109/34.824821

Ranzato, A. M., Poultney, C., Chopra, S., & LeCun, Y. (2006). Efficient Learning of Sparse Representations with an Energy-Based Model. In *Advances in Neural Information Processing Systems 19 (NIPS 2006).*

Sietsma, J., & Dow, R. J. F. (1991). Creating artificial neural networks that generalize. *Neural Networks*, *4*(1), 67–79. doi:10.1016/0893-6080(91)90033-2

Simard, P. Y., Steinkraus, D., & Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.* (Vol. 1, pp. 958–963). IEEE Comput. Soc. doi:10.1109/ICDAR.2003.1227801

Unsworth, C. P., & Coghill, G. (2006). Excessive noise injection training of neural networks for markerless tracking in obscured and segmented environments. *Neural Computation*, *18*(9), 2122–45. doi:10.1162/neco.2006.18.9.2122

Wang, C., & Principe, J. (1999). Training Neural Networks with Additive Noise in the Desired Signal. *IEEE Transactions on Neural Networks*, *10*(6), 1511–1517. doi:10.1109/72.809097

Yaeger, L., Lyon, R., & Webb, B. (1996). Effective Training of a Neural Network Character Classifier for Word Recognition. *NIPS*, *9*, 807–813.