Marginalized Denoising Auto-encoders for Nonlinear Representations

Minmin Chen M.CHEN@CRITEO.COM
Criteo
Kilian Weinberger KILIAN@WUSTL.EDU
Washington University in St. Louis
Fei Sha FEISHA@USC.EDU
University of Southern California
Yoshua Bengio
Université de Montréal, Canadian Institute for Advanced Research

Abstract

Denoising auto-encoders (DAEs) have been successfully used to learn new representations for a wide range of machine learning tasks. During training, DAEs make many passes over the training dataset and reconstruct it from partial corruption generated from a pre-specified corrupting distribution. This process learns robust representation, though at the expense of requiring many training epochs, in which the data is explicitly corrupted. In this paper we present the *marginalized Denoising Auto-encoder* (mDAE), which (approximately) *marginalizes out* the corruption during training. Effectively, the mDAE takes into account *infinitely many* corrupted copies of the training data in every epoch, and therefore is able to match or outperform the DAE with much fewer training epochs. We analyze our proposed algorithm and show that it can be understood as a classic auto-encoder with a special form of regularization. In empirical evaluations we show that it attains 1-2 order-of-magnitude speedup in training time over other competing approaches.

1. Introduction

Learning with artificially corrupted data, which are training samples with manually injected noise, has long been a well-known trick of the trade. For example, images of objects or handwritten digits should be label-invariant with respect to small distortions (e.g, translation, rotation, or scaling) applied to the images. This prior knowledge has been exploited to generate additional training samples for SVM classifiers or neural networks to improve generalization to unseen samples (Bishop, 1995; Burges & Schölkopf, 1997; Herbrich & Graepel, 2004; Ciresan et al., 2012).

Learning with corruption also has benefits in scenarios where no such prior knowledge is available. Denoising auto-encoder (DAE), one of the few building blocks for deep learning architectures, learns useful representations of data by denoising, *i.e.,* reconstructing input data from artificial corruption (Vincent et al., 2008; Maillet et al., 2009; Vincent et al., 2010; Mesnil et al., 2011; Glorot et al., 2011). Moreover, dropout regularization — randomly deleting hidden units during the training of deep neural networks — has been shown to be highly effective at preventing deep architectures from overfitting (Hinton et al., 2012; Krizhevsky et al., 2012; Srivastava, 2013).

However, these advantages come at a price. Explicitly corrupting the training data (or hidden units) effectively increases the training set size, which results in much longer training time and increased computational demands. For example in the case of DAEs, each data sample must be corrupted many times and passed through the learner. This may present a serious challenge for high-dimensional inputs. In the case of dropout regularization, each random deletion gives rise to a different deep learning architecture, all sharing subsets of parameters, and the need to average over many such subsets increases training time too.

In this paper, we propose a novel auto-encoder that takes advantage of learning from many corrupted samples, yet elegantly circumvents any additional computational cost. Instead of *explicitly* corrupting samples, we propose to *implicitly* marginalize out the reconstruction error over all possible data corruptions from a pre-specified corrupting

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32. Copyright 2014 by the author(s).

distribution. We refer to our algorithm as *marginalized Denoising Auto-encoder* (mDAE).

While in spirit similar to several recent works, our approach stands in stark contrast to them. Although Chen et al. (2012) also marginalizes out corruption in auto-encoders, their work is restricted to *linear* auto-encoders, whereas our proposed model directly marginalizes over *nonlinear* encoding and decoding. In contrast to several fast algorithms for log-linear models, our approach learns *hidden* representations while the formers do not (van der Maaten et al., 2013; Wang & Manning, 2013; Wager et al., 2013). Nonetheless, our approach generalizes many of those works when nonlinearity and latent representations are stripped away.

We evaluate the efficacy of mDAE on several popular benchmark problems in deep learning. Empirical studies show that mDAE attains up to 1-2 order-of-magnitude speedup in training time over denoising auto-encoders and their variants. Furthermore, in most cases, mDAE learns *better* representation of the data, evidenced by significantly improved classification accuracies than those competing methods. This can attributed to the fact that mDAE are effectively trained on *infinitely* many training samples.

The rest of the paper is organized as follows. We start by describing our approach in section 2. We discuss related work in section 3 and contrast with our approach. We report experimental results in section 4, followed by conclusion in section 5.

2. Marginalized Denoising Auto-encoder

In what follows, we describe our approach. The key idea is to marginalize out the noise of the corrupted inputs in the denoising auto-encoders. We start by describing the conventional denoising auto-encoders and introducing necessary notations. Afterwards, we present the detailed derivations of our approach. Our approach is general and flexible to handle various types of noise and loss functions for denoising. A few concrete examples with popular choices of noise and loss functions are included for illustration. We then analyze the properties of the proposed approach while drawing connections to existing works.

2.1. Denoising Auto-encoder (DAE)

The Denoising Auto-Encoder (DAE) is typically implemented as a one-hidden-layer neural network which is trained to reconstruct a data point $\mathbf{x} \in \mathcal{R}^D$ from its (partially) corrupted version $\tilde{\mathbf{x}}$ (Vincent et al., 2008). The corrupted input $\tilde{\mathbf{x}}$ is typically drawn from a conditional distribution $p(\tilde{\mathbf{x}}|\mathbf{x})$ — common corruption choices are additive Gaussian noise or multiplicative mask-out noise (where values are set to 0 with some probability q and kept unchanged with probability of $1-q$).

The corrupted input $\tilde{\mathbf{x}}$ is first mapped to a latent representation through the *encoder* (*i.e.,* the nonlinear transformation between the input layer and the hidden layer). Let $\mathbf{z}=h_\theta(\tilde{\mathbf{x}}) \in \mathcal{R}^{D_h}$ denote the D_h-dimensional latent representation, collected at the outputs of the hidden layer.

The code $\mathbf{z}$ is then decoded into the network output $\mathbf{y} = g_\theta(\mathbf{z}) \in \mathcal{R}^D$ by the nonlinear mapping from the hidden layer to the output layer. Note that we follow the custom to have both mappings share the same parameter θ.

For denoising, we desire $\mathbf{y} = g \circ h(\tilde{\mathbf{x}}) = f_\theta(\tilde{\mathbf{x}})$ to be as close as possible to the clean data $\mathbf{x}$. To this end, we use a loss function $\ell(\mathbf{x}, \mathbf{y})$ to measure the reconstruction error. Given a dataset $\mathcal{D} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$, we optimize the parameter θ by corrupting each $\mathbf{x}_i$ m-times, yielding $\tilde{\mathbf{x}}_i^1, \ldots, \tilde{\mathbf{x}}_i^m$, and minimize the averaged reconstruction loss

$$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{m}\sum_{j=1}^{m}\ell\left(\mathbf{x}_i, f_\theta(\tilde{\mathbf{x}}_i^j)\right). \tag{1}$$

Typical choices for the loss ℓ are the *squared loss* for real-valued inputs, or the *cross-entropy loss* for binary inputs.

2.2. Infinite and Implicit Denoising via Marginalization

The disadvantage of *explicitly* corrupting $\mathbf{x}$ and using its multiple copies $\tilde{\mathbf{x}}^1, \ldots, \tilde{\mathbf{x}}^m$ is that the optimization algorithm has to cope with an m-fold larger training dataset. When m is large, this increase directly translates into increased computational cost and training time.

Can we avoid explicitly increasing the dataset size yet still reap the benefits of training with corrupted inputs? Our key idea seems counterintuitive at the first glance: we will use as many copies of corrupted as possible, even infinite!

The trick is to recognize that the empirical average in eq. (1) becomes the *expected* averaged loss under the corruption distribution $p(\tilde{\mathbf{x}}|\mathbf{x})$, as $m \to \infty$. In other words, we will attempt to minimize the following objective function

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{p(\tilde{\mathbf{x}}_i|\mathbf{x}_i)}[\ell(\mathbf{x}_i, f_\theta(\tilde{\mathbf{x}}_i))]. \tag{2}$$

While conceptually appealing, the expectation is not analytically tractable in the most general case due to the nonlinearity of the mappings and the loss function. We overcome this challenge with two approximations. These approximations depend only on the first-order and second-order statistics of the corruption distribution $p(\tilde{\mathbf{x}}|\mathbf{x})$ and can be computed efficiently.

Second-order expansion and approximation. We approximate the loss function $\ell(\cdot)$ by its Taylor expansion

with respect to $\tilde{\mathbf{x}}$ up to the second-order. Concretely, we choose to expand at the mean of the corruption $\mu_{\mathbf{x}} = \mathbb{E}_{p(\tilde{\mathbf{x}}|\mathbf{x})}[\tilde{\mathbf{x}}]$:

$$\begin{aligned} \ell(\mathbf{x}, f_\theta(\tilde{\mathbf{x}})) &\approx \ell(\mathbf{x}, f_\theta(\mu_{\mathbf{x}})) \\ &+ (\tilde{\mathbf{x}} - \mu_{\mathbf{x}})^\top \boldsymbol{\nabla}_{\tilde{\mathbf{x}}} \ell \\ &+ \frac{1}{2}(\tilde{\mathbf{x}} - \mu_{\mathbf{x}})^\top \boldsymbol{\nabla}^2_{\tilde{\mathbf{x}}} \ell \, (\tilde{\mathbf{x}} - \mu_{\mathbf{x}}) \end{aligned} \tag{3}$$

where $\boldsymbol{\nabla}_{\tilde{\mathbf{x}}} \ell$ and $\boldsymbol{\nabla}^2_{\tilde{\mathbf{x}}} \ell$ are the first-order derivative (i.e. gradient) and second-order derivate (*i.e.,* Hessian) of $\ell(\cdot)$ with respect to $\tilde{\mathbf{x}}$ respectively.

The expansion at the mean $\mu_{\mathbf{x}}$ is crucial as the next step shows, where we take the expectation with respect to the corrupted $\tilde{\mathbf{x}}$,

$$\begin{aligned} \mathbb{E}[\ell(\mathbf{x}, f_\theta(\tilde{\mathbf{x}}))] &\approx \ell(\mathbf{x}, f_\theta(\mu_{\mathbf{x}})) \\ &+ \frac{1}{2}\mathrm{tr}\left(\mathbb{E}[(\tilde{\mathbf{x}} - \mu_{\mathbf{x}})(\tilde{\mathbf{x}} - \mu_{\mathbf{x}})^\top] \boldsymbol{\nabla}^2_{\tilde{\mathbf{x}}} \ell\right). \end{aligned}$$

Here, the linear term in eq. (3) vanishes as $\mathbb{E}[\tilde{\mathbf{x}}] = \mu_{\mathbf{x}}$. We substitute in the matrix $\boldsymbol{\Sigma}_{\mathbf{x}} = \mathbb{E}[(\tilde{\mathbf{x}} - \mu_{\mathbf{x}})(\tilde{\mathbf{x}} - \mu_{\mathbf{x}})^\top]$ for the variance of the corrupting distribution, and obtain

$$\mathbb{E}[\ell(\mathbf{x}, f_\theta(\tilde{\mathbf{x}}))] \approx \ell(\mathbf{x}, f_\theta(\mu_{\mathbf{x}})) + \frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Sigma}_{\mathbf{x}} \boldsymbol{\nabla}^2_{\tilde{\mathbf{x}}} \ell\right). \tag{4}$$

Note that the formulation in eq. (4) only requires the first and the second-order statistics of the corrupted data. While this approximation could in principle be used to formulate our new learning algorithm, we make a few more computationally convenient simplifications.

Scaling up. We typically assume the corruption is applied to each dimension of $\mathbf{x}$ independently. This immediately simplifies $\boldsymbol{\Sigma}_{\mathbf{x}}$ to a diagonal matrix. Further, it also implies that we only need to compute the diagonal terms of the Hessian $\boldsymbol{\nabla}^2_{\tilde{\mathbf{x}}} \ell$. This constitutes significant savings in practice, especially for high-dimensional data. The full Hessian matrix scales quadratic with respect to the data dimensionality, while its diagonal scales only linearly.

The d^{th} dimension of the Hessian's diagonal is given by

$$\frac{\partial^2 \ell}{\partial \tilde{x}_d^2} = \left(\frac{\partial \mathbf{z}}{\partial \tilde{x}_d}\right)^\top \frac{\partial^2 \ell}{\partial \mathbf{z}^2} \frac{\partial \mathbf{z}}{\partial \tilde{x}_d} + \left(\frac{\partial \ell}{\partial \mathbf{z}}\right)^\top \frac{\partial^2 \mathbf{z}}{\partial \tilde{x}_d^2}, \tag{5}$$

through a straight-forward application of the chain-rule and the derivatives are backpropagted through the latent representation $\mathbf{z}$. We follow the suggestion by LeCun et al. (1998) and drop the last term in (5). The remaining first term is in a quadratic form. Note that the matrix $\boldsymbol{\nabla}^2_{\mathbf{z}} \ell = \partial^2 \ell / \partial \mathbf{z}^2$ is the Hessian of ℓ with respect to $\mathbf{z}$, and is often positive definite. For instance, for a classification task where the output layer is a softmax-multinomial, the Hessian is that of multinomial logistic regression and therefore positive definite. We exploit the positive definiteness by further reducing the matrix to its non-negative diagonal terms, which gives rise to our final approximation

Table 1. Corrupting distributions with mean and variance

Type	$\mu_{\mathbf{x}}$	$\sigma^2_{\mathbf{x}d}$
Additive Gaussian	$\mathbf{x}$	σ^2_d
Unbiased Mask-out/drop-out	$\mathbf{x}$	$x_d^2 q/(1-q)$

$$\frac{\partial^2 \ell}{\partial \tilde{x}_d^2} \approx \sum_{h=1}^{D_h} \frac{\partial^2 \ell}{\partial z_h^2} \left(\frac{\partial z_h}{\partial \tilde{x}_d}\right)^2 \tag{6}$$

Note that this approximation also brings up significant computational saving as most modern deep learning architectures have a large number of hidden units — the Hessian $\boldsymbol{\nabla}^2_{\mathbf{z}} \ell$ would also have been expensive to compute and store without this approximation.

Learning objective. Combining our results so far, we minimize the following objective function (using one training example for notation simplicity)

$$\ell(\mathbf{x}, f_\theta(\mu_{\mathbf{x}})) + \underbrace{\frac{1}{2} \sum_{d=1}^{D} \sigma^2_{\mathbf{x}d} \sum_{h=1}^{D_h} \frac{\partial^2 \ell}{\partial z_h^2} \left(\frac{\partial z_h}{\partial \tilde{x}_d}\right)^2}_{R_\theta(\mu_{\mathbf{x}})} \tag{7}$$

where $\sigma^2_{\mathbf{x}d}$ is the corruption variance of the d^{th} input dimension, i.e., the d^{th} element of $\boldsymbol{\Sigma}_{\mathbf{x}}$'s diagonal.

It is straightforward to identify that the first term in (7) represents the loss due to the feedforward "mean" (of the corrupted data). We postpone to later sections a detailed discussion and analysis of the intuition behind the second term. In short, the term $R_\theta(\mu_{\mathbf{x}})$ functions as a form of regularization, reminiscent of those used in the contractive auto-encoder (Rifai et al., 2011b) and the reconstruction contractive auto-encoder (Alain & Bengio, 2013) – details in section 3.

2.3. Examples

We exemplify our approach with a few concrete examples of the corrupting distributions and loss functions.

Corrupting distributions. Table 1 summarizes two types of noise models and their corresponding statistics.

In the case of *additive Gaussian noise*, we have $p(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\mathbf{x}, \boldsymbol{\Sigma})$ where the covariance matrix is independent of $\mathbf{x}$. Additive Gaussian noise is arguably the most common data corruption used to model data impurities in practical applications (Bergmans, 1974).

For *mask-out/drop-out* corruption, we overwrite each of the dimensions of $\mathbf{x}$ randomly with 0 at a probability of q. To

Table 2. Reconstruction loss functions and the relevant derivatives.

Name	$\ell(\mathbf{x}, \mathbf{y})$	$\frac{\partial^2 \ell}{\partial z_h^2}$	$\frac{\partial z_h}{\partial \tilde{x}_d}$
Cross-entropy loss	$-\mathbf{x}^\top \log(\mathbf{y}) - (1-\mathbf{x})^\top \log(1-\mathbf{y})$	$\sum_d y_d(1-y_d)w_{hd}^2$	$z_h(1-z_h)w_{hd}$
Squared loss	$\|\mathbf{x}-\mathbf{y}\|^2$	$2\sum_d w_{hd}^2$	$z_h(1-z_h)w_{hd}$

make the corruption unbiased, we set uncorrupted dimensions to $1/(1-q)$ times its original value. That is,

$$P(\tilde{x}_d = 0) = q, \text{ and } P(\tilde{x}_d = 1/(1-q)x_d) = 1-q. \quad (8)$$

While the noise is unbiased, the variance is now a function of $\mathbf{x}$, as shown in Table 1. This type of corruption has been shown to be highly effective for bag-of-words document vectors (Glorot et al., 2011; Chen et al., 2012), simulating the loss of some features due to *e.g.* other word choices by the document's authors, and recently has become known as "drop-out" in the context of neural network regularization (Hinton et al., 2012).

Loss. Table 2 highlights two loss functions and the corresponding derivatives in eq. (7). The *cross-entropy loss* is best suited for binary inputs and the *squared loss* a typical choice for regression.

We assume that in both cases the hidden representation is computed as

$$\mathbf{z} = \sigma(\mathbf{W}\tilde{\mathbf{x}} + \mathbf{b}), \quad (9)$$

where $\sigma()$ is the sigmoid function, $\mathbf{W} \in \mathcal{R}^{D_h \times D}$ is the connection weight matrix between the input and the hidden layers and $\mathbf{b}$ the bias term. For the binary inputs scenario, the outputs are computed as $\mathbf{y} = \sigma(\mathbf{W}^\top \mathbf{z} + \mathbf{b}')$ and we use the cross-entropy loss to measure the reconstruction. For regression, the outputs are $\mathbf{y} = \mathbf{W}^\top \mathbf{z} + \mathbf{b}'$ and we use the squared loss. Table 2 summarizes the relevant derivatives with different reconstruction loss function. We leave the detailed derivation to the Supplementary Material.

2.4. Analysis of the Regularizer

We gain further insight by examining the regularizer $R_\theta(\mu_\mathbf{x})$ in eq. (7) under specific combinations of corruption distributions and reconstruction loss functions. For example, under the mask-out noise and the cross-entropy loss, we have

$$\mathcal{R}_\theta(\mu_\mathbf{x}) \propto \sum_h z_h^2(1-z_h)^2 \sum_{d,d'} x_d^2 y_{d'}(1-y_{d'}) w_{hd}^2 w_{hd'}^2.$$

This form reveals several interesting aspects of the regularizer.

Our first observation is that the regularizer favors a binary hidden representation and penalizes if the hidden output z_h is ambiguous — the most extreme case being $z_h = 1/2$.

Secondly, the regularizer is *adaptive* to both the inputs and the outputs. For active values x_d and $y_{d'}$ it penalizes all paths $w_{hd}, w_{hd'}$ that use x_d for the reconstruction of $x_{d'}$. This observation is analogous to the adaptive regularization effect previously observed on the logistic regression (Wager et al., 2013).

Thirdly, in contrast to typical measuring model parameters with L_2 norms, our regularizer captures higher-order interactions. When $d = d'$, we see a penalty term of w_{hd}^4, which grows faster than w_{hd}^2. Furthermore, there is a **mutual competition and suppression** for weights belonging to the same hidden unit. The regularizer prefers all w_{hd} for the same h to different inputs (or outputs units) to be as **orthogonal** as possible:

$$w_{hd}^2 w_{hd'}^2 \approx 0$$

As our experiments will show later, this preference leads to a **group** of sparser weights (cf. fig. 2). When interpreting those weights as filters, we obtain sharply contrasted filters. It is worth pointing out that this type of orthogonality regularization has been used in other settings of learning models with disjoint sets of features (Hwang et al., 2011; Zhou et al., 2011; Chen et al., 2011).

3. Related work

Various forms of auto-encoders have been studied in the literature (Rumelhart et al., 1986; Baldi & Hornik, 1989; Kavukcuoglu et al., 2009; Lee et al., 2009; Vincent et al., 2008; Rifai et al., 2011b). While originally intended as a technique for dimensionality reduction (Rumelhart et al., 1986), auto-encoders have been repurposed to learn sparse and distributed representation in the over-complete settings, where the learned representation has higher dimensions than the input space. To avoid learning an identity mapping (thus uninteresting features) under this setting, it is crucial to have regularization in those models. The simplest form is to use weight decay (Bengio & LeCun, 2007), which favors small weights. The sparse auto-encoders proposed by (Lee et al., 2007; Ranzato et al., 2007) encourage sparse activation of the hidden representation. Our work generalizes those ideas by suggesting more complex forms of regularization, for example, being adaptive to inputs when using mask-out noise.

Connection to DAE and its variants. Denoising auto-encoders (DAE) (Vincent et al., 2008) incorporate a new form of regularization to force the mapping between the inputs and the outputs to deviate from an identity mapping.

That is achieved by corrupting the inputs (for instance, randomly setting a subset of input dimensions to zero) while demanding the corrupted dimensions be reconstructed at the outputs.

Rifai et al. (2011b) asks the more direct question: what kind of representations we desire and thus what regularizers do we need for a regular auto-encoder? Their contractive auto-encoder (CAE) thus explicitly encourages learning latent representation to be robust to small perturbation to the inputs. To this end, CAE penalizes the magnitude of the Jacobian matrix of the hidden units at the training examples:

$$\|J_h(\mathbf{x})\|_F^2 = \left\| \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right\|_F^2 = \sum_{h=1}^{D_h} \sum_{d=1}^{D} \left(\frac{\partial z_h}{\partial \tilde{x}_d} \right)^2$$

In contrast, our regularizer in eq. (7) takes also into consideration the curvature of the reconstruction loss function by weighting the Jacobian with $\frac{\partial^2 \ell}{\partial z_h^2}$. Moreover, in contrast to CAE, our regularizer is able to adapt to the inputs explicitly by scaling with input-dependent noise variance.

Alain & Bengio (2013) aims to understand the regularization property of the DAE by marginalizing the (Gaussian) noise. They arrive at a reconstruction contractive auto-encoder (RCAE) whose regularization term is the Jacobian of the *reconstruction function*. While RCAE cannot be seen as a direct replacement of CAE, it is interesting to note that our mDAE has the flavor of both RCAE and CAE — mDAE's regularization encodes jointly the properties of the loss (thus indirectly the regression function) and the hidden representations.

Connection to other marginalized models. Wager et al. (2013) analyze the effect of dropout/mask-out on learning logistic regression models. In particular, they analyze the expected loss function of the learning algorithm with respect to the corruption distribution. An approximation to the expected loss is derived under small noise condition. They discover that the effect of marginalizing out the noise is equivalent to adding an adaptive regularization term to the loss function formulated with the original training samples. A similar effect is also observed in our analysis, cf. section 2.4.

While sharing in spirit with that line of work, our focus is also inspired by (Chen et al., 2012; van der Maaten et al., 2013) which see marginalization as a vehicle to arrive at fast and efficient computational alternative to explicitly constructing corrupted learning samples. Because the hidden layers in our auto-encoders are no longer linear, our analysis extends existing work in interesting directions, revealing novel aspects of adaptive regularization due to the need of learning latent representations and compounded (sigmoidal) nonlinearity.

4. Experimental Results

We evaluate mDAE on a variety of popular benchmark datasets for representation learning and contrast its performance to several competitive state-of-the-art algorithms. We start by describing the experimental setup, followed by reporting results.

4.1. Setup

Datasets. Our datasets consist of the original MNIST dataset (*MNIST*) for recognizing images of handwritten digits, for the sake of comparison with prior work a subsampled version (*basic*) and its several variants (Larochelle et al., 2007; Vincent et al., 2010; Rifai et al., 2011b). The variants consist of five more challenging modification to the MNIST dataset, including images of rotated digits (*rot*), images superimposed onto random (*bg-rand*) or image background (*bg-img*) and the combination of rotated digits with image background (*bg-img-rot*). We also experimented on three shape classification tasks (*convex, rect, rect-img*). Each dataset is split into three subsets: a training set for pre-training and fine-tuning the parameters, a validation set for choosing the hyper-parameters and a testing set on which the results are reported. More details can be found in (Vincent et al., 2010).

Methods. We compare to the original denoising auto-encoder (DAE) (Vincent et al., 2010), the contractive auto-encoder (CAE) (Rifai et al., 2011b) and the marginalized linear auto-encoder (mLDAE) (Chen et al., 2012). The performance of these algorithm before and after fine-tuning the learned representation are both included. Our baseline is a linear SVM on the raw image pixels.

We used cross-entropy loss and additive isotropic gaussian noise for DAE and mDAE throughout these experiments (similar trends was observed with mask-out noise). The hyper-parameters for these different algorithms are chosen on the validation set. These include the learning rate for pre-training and fine-tuning (candidate set $[0.01, 0.05, 0.1, 0.2]$), noise levels in mLDAE, DAE and our method mDAE (candidate set $[0.05, 0.1, 0.3, 0.5, 0.7, 0.9, 1.1, 1.3]$)), and the regularization coefficient in CAE (candidate set $[0.01, 0.05, 0.1, 0.3, 0.5, 0.7, 0.9]$). Except mLDAE which has closed-form solutions for learning representations, all other methods use stochastic gradient descent for parameter learning.

4.2. Results

Training speed. Figure 1 displays the testing error on all benchmark data sets as a function of the training epochs. The best results based on the validation set are highlighted

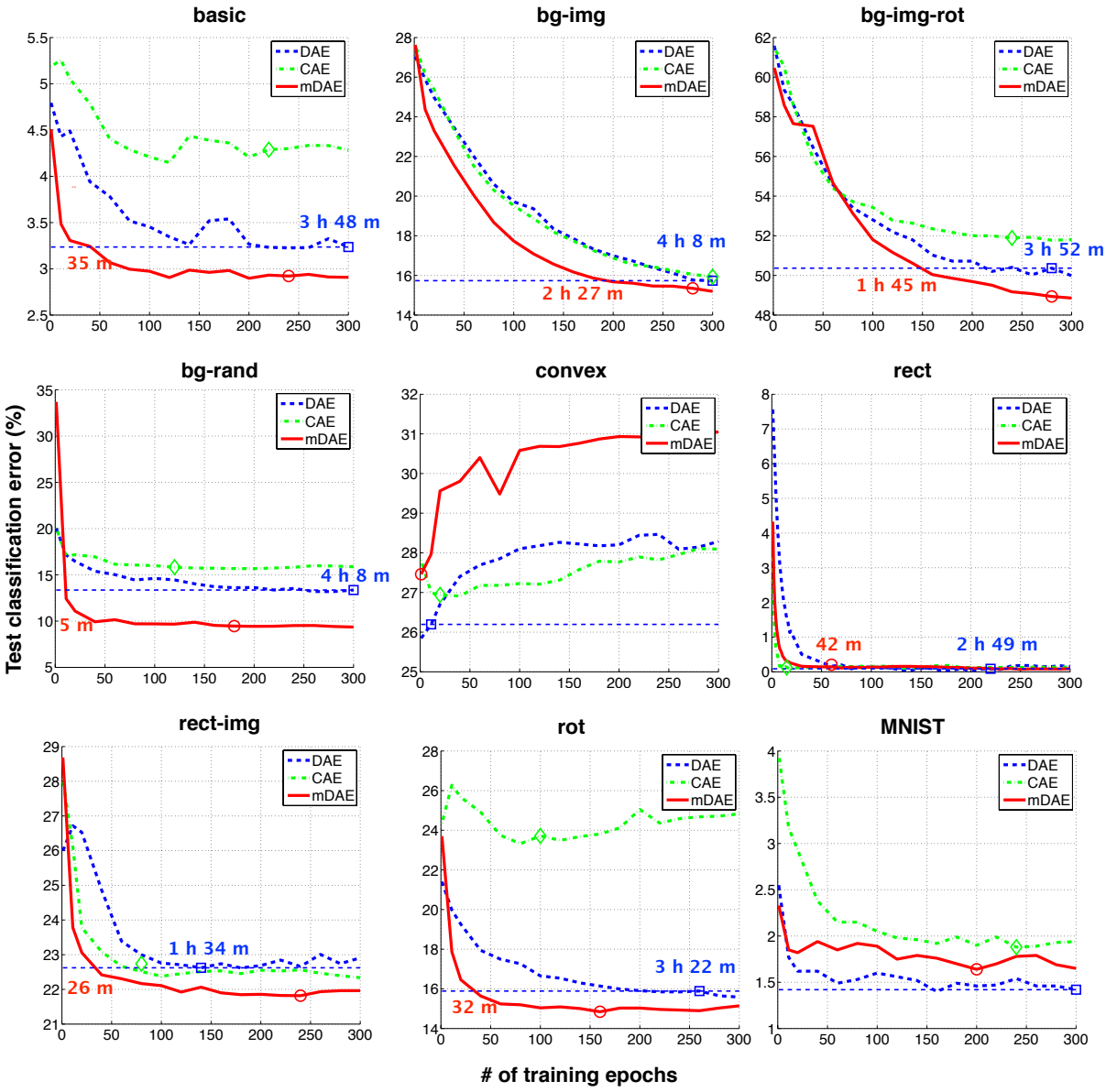

Figure 1. Test error rates (in %) on the nine benchmark datasets obtained by DAE, CAE and our mDAE at different training epochs.

with small markers. We can see that mDAE is able to match the performance of the DAE or CAE often with much fewer training epochs, thus significantly reducing the training time. In the most prominent case, *bg-rand*, it requires less than five training epochs (after 5 minutes of training time) to reach the same error as the DAE, which requires over 4 hours to finish training. Similar trends are observed on most datasets, with the exceptions of *MNIST* (where mDAE performs slightly worse than DAE).

Better representations. If allowed to progress until the lowest error on the validation set is reached, mDAE is also able to yield better representations than DAE in 7 out of 9 data sets. Figure 1 shows that the features learned with mDAE quickly yield lower classification errors in these cases. Table 3 summarizes the classification errors of the linear SVMs (Fan et al., 2008) using representations learned (before fine-tuning) by all algorithms, as well as the errors after fine-tuning the learned representations using discriminative labels. The test errors obtained with the raw pixel inputs are record in the *baseline* column. When trained with one hidden layer, mDAE often outperforms other approaches by significant margins. The table also shows the results of two hidden layers, learned through stacking (Vincent et al., 2010). With two layers, the beneficial effects of mDAE decrease slightly, however training

Table 3. Test error rates (in %) of a baseline linear SVM on raw input and mLDAE learned representation, as well as the error rates produced by DAE, CAE and mDAE before (*upper row*) and after (*lower row*) fine-tuning. Best results with one layer and two layers are indicated in **bold** (before fine-tuning) and **bold**(after fine-tuning), respectively.

Dataset	*baseline*	mLDAE_1	one layer			two layers		
			DAE_1	CAE_1	mDAE_1	DAE_2	CAE_2	mDAE_2
MNIST	8.31	7.17	**1.42**	1.88	1.64	1.38	-	1.60
			1.37	1.49	**1.37**	**1.29**	-	1.43
basic	10.15	8.02	3.24	4.29	**2.92**	2.79	3.98	**2.61**
			3.13	4.01	3.17	2.75	3.34	**2.66**
rot	49.34	25.31	15.89	23.49	**14.84**	**15.50**	20.09	16.61
			12.61	14.58	**12.05**	11.94	13.62	**10.36**
bg-rand	20.83	21.31	13.35	15.82	**9.46**	11.67	13.23	**8.15**
			13.85	15.05	**13.07**	11.56	14.84	**11.04**
bg-img	28.17	29.76	15.74	15.94	**15.20**	**17.59**	18.12	18.09
			18.62	17.87	**17.18**	17.30	**16.75**	17.38
bg-img-rot	65.97	66.07	49.63	51.89	**48.78**	51.45	51.91	**49.62**
			48.70	49.02	**47.27**	**44.92**	48.25	46.12
rect	24.66	12.50	0.26	0.29	**0.12**	0.06	0.22	**0.05**
			0.20	0.10	**0.07**	**0.04**	0.19	0.07
rect-img	49.80	25.31	22.63	22.39	**21.82**	22.42	24.33	**21.97**
			22.04	**21.66**	22.01	22.19	23.42	**22.05**
convex	46.27	29.96	**26.20**	26.94	27.46	22.10	26.25	**21.52**
			21.35	21.01	**20.53**	18.44	19.30	**18.10**

is still significantly faster in most cases. Note that without fine-tuning two stacked layers often do not improve the feature quality across all approaches. A single-layer mDAE is able to outperform stacking two layers of DAE or CAE on several datasets, such as *bg-rand* and *bg-img-rot*. Representation learned by mDAE without fine-tuning is able to outperform DAE or CAE with fine-tuning on several datasets, such as *basic* and *bg-rand*.

Analysis of the model parameters. The connection weights between neural network layers are often interpreted as filters that transform lower-level inputs. Thus, it is often instructive to study the properties of those filters to understand the process of the learning.

Figure 2 shows 100 randomly selected filters, from a total of 1000, learned by mDAE on three datasets. Exemplary inputs for the various data sets are shown on the very left. mDAE is able to discover interesting (visibly non-random and clearly structured) filters. On the *basic* (left) dataset, it is able to learn specialized feature extractors, detecting for example ink blobs, local oriented strokes and digit parts such as loops. On both *bg-rand* (middle) and *bg-img* (right) datasets, the model learns filters which are more sensitive to foreground digits as well as filters which capture the backgrounds.

Figure 3 compares the filters learned by four different auto-encoder variants: an auto-encoder without denoising or regularization (AE), DAE, CAE and our mDAE. As shown in the figure, AE filters largely look random and fail to learn any interesting features, confirming the importance of applying regularization to such models. Some of CAE's filters capture interesting patterns such as edges and blobs. Both DAE and mDAE seem to have highly specialized and well-structured feature detectors. In particular, mDAE seems to have sharply contrasted filters. The filters from mDAE have the tendency to be specialized towards smaller image regions. This may be an artifact of the regularization term, which penalizes reconstruction paths across different input dimensions. The strongest reconstruction signal is usually the pixel itself and its neighboring pixels (which are highly correlated) and the mDAE filters tend to focus on exactly those. Note that these filters with local activation regions tend to have less overlap and are more likely to be orthogonal. Both observations are in tune with the analysis in section 2.4.

5. Conclusion

Regularized auto-encoders are important building blocks for learning deep and rich representations of data. The standard approach of denoising auto-encoder incorporates regularization via learning reconstruction from partially corrupted samples. While effective, this is often a computa-

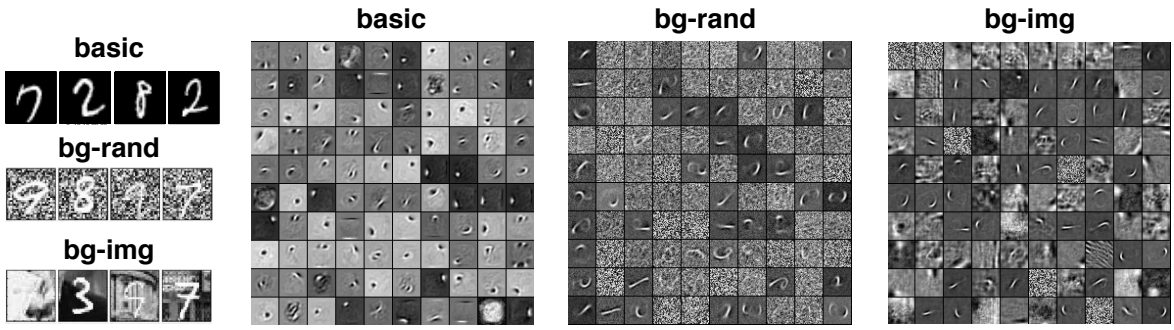

Figure 2. 100 filters (randomly selected from 1,000) learned by mDAE on the *basic* (left), *bg-rand* (middle), *bg-img* (right) datasets. Exemplary input images are shown in the very left column. It is interesting to observe that for the *bg-rand* and *bg-img* data sets mDAE learns different specialized filters for foreground and background attributes.

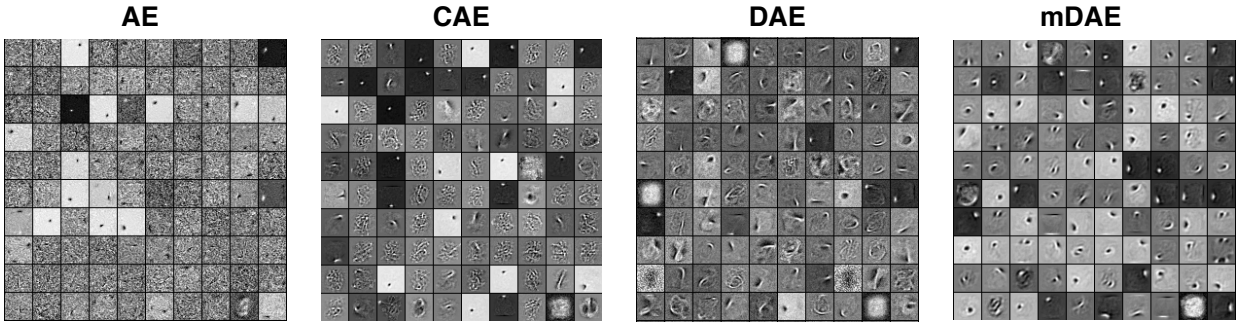

Figure 3. 100 filters (randomly selected from 1,000) learnt by a regular auto-encoder without regularization (AE), CAE, DAE and our mDAE on the *basic* dataset. Additive isotropic gaussian noise is used in DAE and mDAE.

tionally intensive and lengthy process. Our mDAE overcomes the limitation by marginalizing the corruption process, effectively learning from infinitely many corrupted samples. At the core of our approach is to approximate the expected loss function with its Taylor expansion. Our analysis yields a regularization term that takes into consideration both the reconstruction function's sensitivity to the hidden representations and the hidden representation's sensitivity to the inputs. Algebraically, those sensitivities are measured by the norms of the corresponding Jacobians.

The idea of employing Jacobians to form regularizations has been studied before and has since resulted in several interesting models, including ones for regularizing auto-encoders (Rifai et al., 2011b). We plan to advance further in this direction by exploring high-order effects of corrupt. For instance, inspiring thoughts include injecting noise into a Jacobian-based regularizer itself (Rifai et al., 2011a) as well as approximating with higher-order expansions.

In summary, this paper contributes to the deeper understanding of feature learning with DAE and also proposes a novel practical algorithm. The modular structure of mDAE allows many different corruption distributions as well as reconstruction loss functions to be readily used in a plug-and-play manner, providing interesting directions for future research and analysis.

Acknowledgement

KQW were supported by NSF IIS-1149882 and IIS-1137211. FS is partially supported by the IARPA via DoD/ARL contract # W911NF-12-C-0012. YB was supported by NSERC, the Canada Research Chairs and CIFAR.

References

Alain, G and Bengio, Y. What regularized auto-encoders learn from the data generating distribution. In *ICLR*, 2013.

Baldi, P. and Hornik, K. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.

Bengio, Y. and LeCun, Y. Scaling learning algorithms towards AI. *Large-Scale Kernel Machines*, 34, 2007.

Bergmans, P. A simple converse for broadcast channels

with additive white gaussian noise. *Information Theory, IEEE Transactions on*, 20(2):279–280, 1974.

Bishop, C. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116, 1995.

Burges, C.J.C. and Schölkopf, B. Improving the accuracy and speed of support vector machines. *NIPS*, 9:375–381, 1997.

Chen, M., Weinberger, K.Q., and Chen, Y. Automatic Feature Decomposition for Single View Co-training. In *ICML*, 2011.

Chen, M, Xu, Z, Weinberger, K, and Sha, F. Marginalized denoising autoencoders for domain adaptation. In *ICML*, 2012.

Ciresan, D, Meier, U, and Schmidhuber, J. Multi-column deep neural networks for image classification. In *CVPR*, 2012.

Fan, R, Chang, K, Hsieh, C, Wang, X, and Lin, C. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

Glorot, X., Bordes, A., and Bengio, Y. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*, 2011.

Herbrich, R. and Graepel, T. Invariant pattern recognition by semidefinite programming machines. In *NIPS*, 2004.

Hinton, Geoffrey E, Srivastava, Nitish, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Hwang, Sungju, Grauman, Kristen, and Sha, Fei. Learning a tree of metrics with disjoint visual features. In *NIPS*, Granada, Spain, 2011.

Kavukcuoglu, K., Ranzato, M.A., Fergus, R., and Le-Cun, Y. Learning invariant features through topographic filter maps. In *CVPR*, 2009.

Krizhevsky, A, Sutskever, I, and Hinton, G. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

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

LeCun, Y, Bottou, L, Orr, G B, and Müller, K. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–50. Springer, 1998.

Lee, H, Largman, Y, Pham, P, and Ng, A Y. Unsupervised Feature Learning for Audio Classification using Convolutional Deep Belief Networks. In *NIPS*. 2009.

Lee, Honglak, Ekanadham, Chaitanya, and Ng, Andrew. Sparse deep belief net model for visual area v2. In *NIPS*, 2007.

Maillet, F., Eck, D., Desjardins, G., and Lamere, P. Steerable playlist generation by learning song similarity from radio station playlists. In *ISMIR*, pp. 345–350, 2009.

Mesnil, G., Dauphin, Y., Glorot, X., Rifai, S., Bengio, Y., Goodfellow, I., Lavoie, E., Muller, X., Desjardins, G., Warde-Farley, D., Vincent, P., Courville, A., and Bergstra, J. Unsupervised and transfer learning challenge: a deep learning approach. *JMLR: Workshop and Conference Proceedings*, 7:1–15, 2011.

Ranzato, M., Boureau, L., and LeCun, Y. Sparse feature learning for deep belief networks. *NIPS*, 2007.

Rifai, S, Glorot, X, Bengio, Y, and Vincent, P. Adding noise to the input of a model trained with a regularized objective. *arXiv:1104.3250*, 2011a.

Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. Contractive auto-encoders: Explicit invariance during feature extraction. In *ICML*, 2011b.

Rumelhart, D E, Hintont, G E, and Williams, R J. Learning representations by back-propagating errors. *Nature*, 323 (6088):533–536, 1986.

Srivastava, Nitish. Improving neural networks with dropout. Technical report, 2013.

van der Maaten, L.J.P., Chen, M., Tyree, S., and Weinberger, K.Q. Learning with marginalizing corrupted features. In *ICML*, 2013.

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

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

Wager, S, Wang, S, and Liang, P. Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems*, pp. 351–359, 2013.

Wang, S and Manning, C. Fast dropout training. In *ICML*, pp. 118–126, 2013.

Zhou, D., Xiao, L., and Wu, M. Hierarchical classification via orthogonal transfer. In *Proc. of ICML*, 2011.