Backprop-Free Auto-Encoders

Dong-Hyun Lee
Université de Montréal
Montréal, QC H3C 3J7

Yoshua Bengio
Université de Montréal
Montréal, QC H3C 3J7
CIFAR Fellow

Abstract

Back-propagation is typically used to train multi-layer neural networks, in particular auto-encoders, which have at least one hidden layer containing the learned representation. Can auto-encoders be trained without back-propagation? This is an interesting question because auto-encoders are building blocks for other unsupervised learning procedures, and being able to train good auto-encoders without using back-propagation could be useful for hardware implementations, as a model for computational neuroscience, or to handle non-differentiable activation functions. Following up on the much earlier idea of recirculation (dating to the years when auto-encoders were invented), we propose an algorithm for training Backprop-Free Auto-Encoders (BFAE). It exploits the fact that when the encoder sees an input x and produces an output h, the (h, x) pair can be used as an (input,target) pair for the decoder, and similarly in the other direction. The proposed algorithm exploits this symmetry and is actually minimizing the two-way reconstruction error (to auto-encode x's as well as auto-encoding h's) and does it as well as or better than regular denoising auto-encoders trained with back-propagation.

1 Introduction

Auto-encoders are interesting building blocks for learning representations, and in particular deep representations (Bengio *et al.*, 2007, 2013b). An interesting question raised by Bengio (2014) is whether one can succesfully train auto-encoders without requiring back-propagation to obtain parameter updates. One motivation is biological plausibility, since backpropagation appears at least on the surface to require non-local computation that is not neural-like (Crick, 1989; Zipser and Andersen, 1988). Another motivation, introduced by Bengio (2014) is that one might be able to generalize backpropagation-based updates to networks that are very non-linear or even perform discrete (non-differentiable) computation. Using the same hardware for computing neuron outputs and for propagating training signals, as well as the ability to deal with high-nonlinear, or discrete units (e.g. binary units, as in spiking networks), could also be useful for low-power hardware implementations of neural networks. Backprop-free auto-encoders are also building blocks towards replacing back-propagation by target propagation, according to the approach proposed by Bengio (2014).

Fortunately, there is a very basic reason why one might hope that it should be possible to train auto-encoders (and even deep ones) without actually requiring any back-propagation. If the encoder computes an output h for a given input x, then the (h, x) pair should constitute a good (input,target) training example for the decoder. Similarly, if the decoder computes an output $\hat{x}$ for a given input $\hat{h}$, then the $(\hat{x}, \hat{h})$ should constitute a good (input,target) training example for the encoder. Hence the encoder and decoder can provide supervised input/output examples to each other. Although this idea is simple, one might wonder about getting stuck in poor solutions where the auto-encoder does not learn anything useful.

This paper studies the most basic scenario for training an auto-encoder, i.e., a shallow encoder f coupled with a shallow decoder g, with the usual composition of affine and sigmoidal non-linear

activation:

$$f(x) = \sigma(b + Wx)$$
$$g(h) = \sigma(c + Vh) \tag{1}$$

This proposes a novel algorithm for training auto-encoders without backpropagation, called *Backprop-Free Auto-Encoder* (BFAE). The experiments are performed in the context of the Denoising Auto-Encoder (DAE) (Vincent *et al.*, 2008), because it works well as a feature extractor (Vincent *et al.*, 2010) for deep networks and because it is now better understood mathematically: it can be shown to implicitly estimate the data generating distribution (Alain and Bengio, 2013; Bengio *et al.*, 2013a, 2014), and one can sample from the estimated distribution by iterated applications of encoding, decoding and noise injection, similarly to the Gibbs Markov chain for sampling from RBMs (Hinton *et al.*, 2006).

1.1 Recirculation

The proposed algorithm is a direct descendant of and closely related to the much older algorithm by Hinton and McClelland (1988) called *recirculation*. The basic recirculation algorithm is similar to the vanilla BFAE illustrated in Figure 1, with the delta-rule for updating weights, linear reconstruction (with squared error), and no noise injection. Hinton and McClelland (1988) also consider a variant called *recirculation with regression* in which the new value of the visible (input) units after the reconstructions are computed are obtained by a linear combination of the original value of x and the reconstruction $g(h)$, with a weight λ:

$$\hat{x} = \lambda x + (1 - \lambda) g(h) \tag{2}$$

where $\lambda = 0.75$ in their experiments. The paper shows that with sufficiently small λ and the above architecture, the delta-rule approximates gradient descent on the squared error reconstruction. They call this variant the recirculation with regression and call λ the regression coefficient.

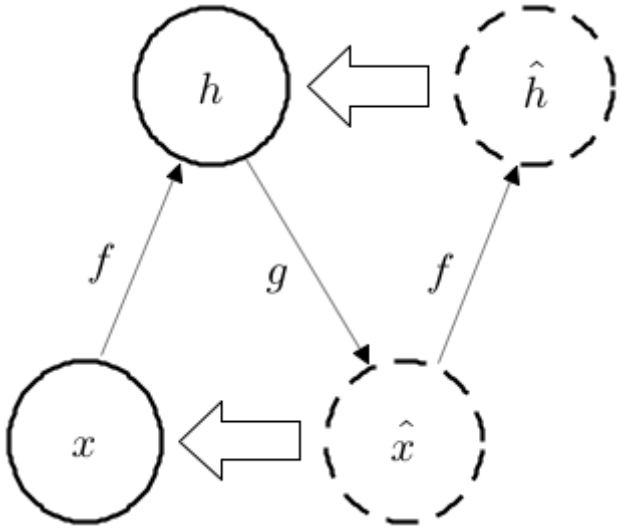


Figure 1: Vanilla Backprop-Free Auto-Encoder, very similar to the recirculation procedure (Hinton and McClelland, 1988). The decoder parameters (in g) are updated to make $g(f(x))$ close to x while the encoder parameters (in f) are updated to make $f(g(h))$ close to h. Noise is injected to make both $g \circ f$ and $f \circ g$ good denoising auto-encoders.

2 Backprop-Free Auto-Encoders

The basic Backprop-Free Auto-Encoder (BFAE) is very similar to the basic version of the recirculation procedure (Hinton and McClelland, 1988), but with the possibility of having non-linear

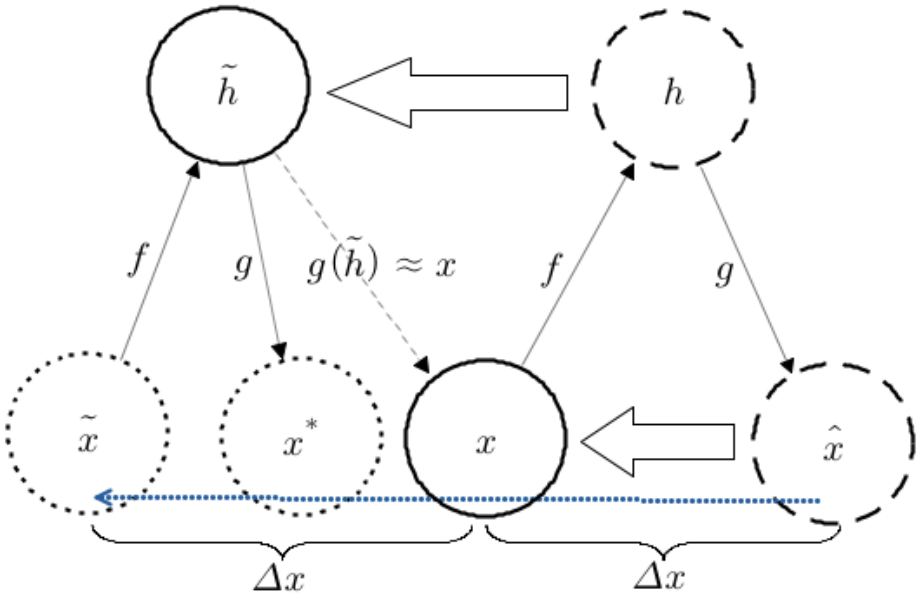


Figure 2: Back-step Backprop-Free Auto-Encoder. The motivation for this variant is that we want the encoder to be best at inverting the decoder where the decoder generates data points. Thus we feed the encoder with an input that will be such that applying the decoder on the result will be close to x.

encoders and decoders, arbitrary reconstruction loss functions, and noise injected, as in the denoising auto-encoder. Thus, the basic recirculation algorithm is a special case and the main inspiration. We compute

$$h = f(x) \tag{3}$$

$$\hat{x} = g(corrupt(h)) \tag{4}$$

$$\hat{h} = f(corrupt(\hat{x})) \tag{5}$$

and optimize the following two-way loss functions together:

$$L = \text{loss}(g(corrupt(h)), x) + \text{loss}(f(corrupt(\hat{x})), h) \tag{6}$$

where loss(output,target) is the chosen reconstruction loss function and $corrupt(a)$ injects noise, e.g. in our experiments zero-mean Gaussian noise is added to a. In other words, the input x is a target for the reconstruction $\hat{x}$ and the representation h is a target for the second-step encoder output $\hat{h}$. A target provides the desired value for the corresponding unit, so we update only the encoder f with respect to $\text{loss}(f(corrupt(\hat{x})), h)$ and we update only the decoder g with respect to $\text{loss}(g(corrupt(h)), x)$. x, h and $\hat{x}$ are thus treated as a constants in the optimization, with no gradient back-propagated through them. So we do not need to use back-propagation: error signals do not need to cross over hidden layers. The advantage of the added corruption, like in the denoising auto-encoder variants (Vincent *et al.*, 2008; Bengio *et al.*, 2014), is that we get contractive mappings for both $f(x)$ and $g(h)$, yielding better representations that are more robust in the vicinity of the training examples and allow to capture the data distribution (Alain and Bengio, 2013; Bengio *et al.*, 2013a).

2.1 Back-step BFAE

The input denoising performance of the vanilla BFAE can be degraded because the target it uses for improving $h = f(x)$ is not the optimal one. *The optimal target value for $h = f(x)$ should be an h^* that would make the decoder g recover x when the decoder is applied to it*, i.e., we would like to use as target for h a vector h^* such that

$$g(h^*) = x.$$

Finding that h^* is difficult and would require an iterative optimization. Instead, in order to improve on the vanilla BFAE model, we suggest the following Back-Step BFAE variant. To approximate h^*, we assume a linear approximation and obtain the proposed target $\tilde{h}$. If

$$g(f(x)) = \hat{x} \approx x + \Delta x$$

is an additive approximation of $g \circ f$, i.e., locally $g \circ f$ adds Δx to its argument, then in order to get the output of $g \circ f$ to produce x, this approximation suggests to use

$$\tilde{x} = x - \Delta x = 2x - \hat{x}$$

so that

$$g(f(\tilde{x})) = x^* \approx x.$$

In order to estimate x^* by $\tilde{x}$, we thus first compute a *clean version* of the reconstruction:

$$h = f(x) \tag{7}$$
$$\hat{x} = g(h) \tag{8}$$
$$\tilde{x} = 2x - \hat{x} \tag{9}$$
$$\tilde{h} = f(\tilde{x}) \tag{10}$$
$$x^* = g(\tilde{h}) \tag{11}$$

and optimize the following loss functions together:

$$L = loss(g(corrupt(h)), x) + loss(f(corrupt(x)), \tilde{h}) + loss(f(corrupt(x^*)), \tilde{h}). \tag{12}$$

If $loss(x^*, x) < loss(x, \hat{x})$ holds, we can think that this approximation works well and eventually $x^* = g(\tilde{h})$ can approach x as $loss(g(corrupt(h)), x)$ decreases. Actually this relation always holds in our experiments. This model shows better reconstruction error on both x and h than the vanilla BFAE. The last loss function is added to get more robust results and make sure that f also makes $f \circ g$ being a good denoising auto-encoder at $\tilde{h}$.

Note that the updates on the parameters of f and g are obtained by computing the gradient of the above losses on the weights and biases, which ends up being (like for recirculation) the delta-rule, of the form (target - output) times input.

3 Experiments

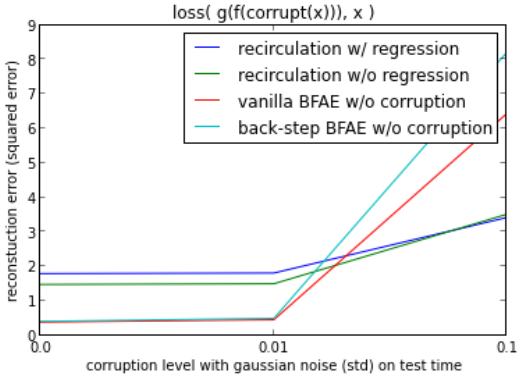
3.1 Dataset and Performance Monitoring

In order to test our models, we used the MNIST handwritten digit recognition dataset. This dataset is the most famous one in deep learning literature and has been used a lot for unsupervised learning. Most of the papers estimate classification error or the generative log-likelihood on test set. However, we estimate only the denoising reconstruction error on the validation set in order to know whether our model can train (denoising) auto-encoding function without back-propagation, and we compare with the denoising reconstruction loss obtained with a regular DAE trained using back-prop to compute true gradients.

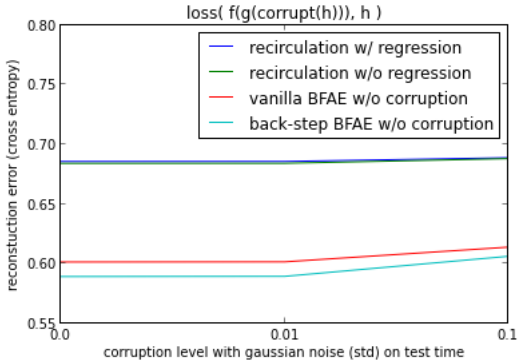
Although we do not expect the BFAE to optimize denoising reconstruction error better than the backprop-based DAE, we note that the two models do not really minimize the same objective. The BFAE is minimizing a reconstruction error not only on the $g \circ f$ auto-encoder but also on the $f \circ g$ auto-encoder.

For that reason, the experiments monitor both reconstruction errors, $\mathrm{loss}(g(f(corrupt(x))), x)$ and $\mathrm{loss}(f(g(corrupt(h))), h)$ and their sum. Moreover we also test the two-way DAE which optimizes the two-way loss (the sum of the above losses) using back-propagation as a strong baseline.

We use averaged binary cross entropy as a loss function and salt and pepper noise for corruption. For testing, we split the original 60,000 training set into 50,000 train / 10,000 valid sets and used the validation set for hyper-parameter search.

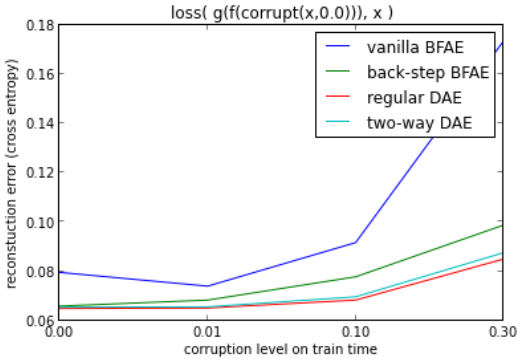


(a) Denoising reconstruction error on input x

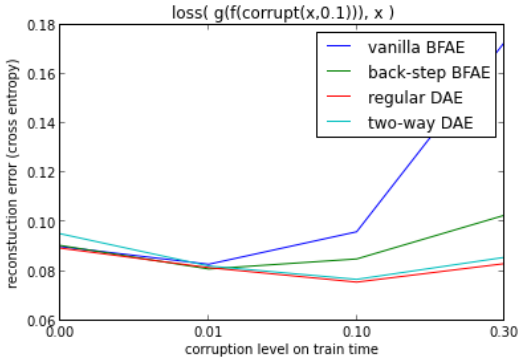


(b) Denoising reconstruction error on hidden representation h

Figure 3: Denoising reconstruction error on input x (a) and hidden representation h (b) for various levels of corruption at testing time to compare BFAEs to the recirculation algorithm. BFAEs performed better than the recirculation except on input x with $std = 0.1$, due to choosing learning rates according to $loss(g(f(x)), x)$ and without using corruption. The experiment setup is the same as the recirculation paper, with squared error reconstruction on x (linear units), but cross-entropy on h (sigmoid units).



(a) test-time corruption level p = 0.0



(b) test-time corruption level p = 0.1

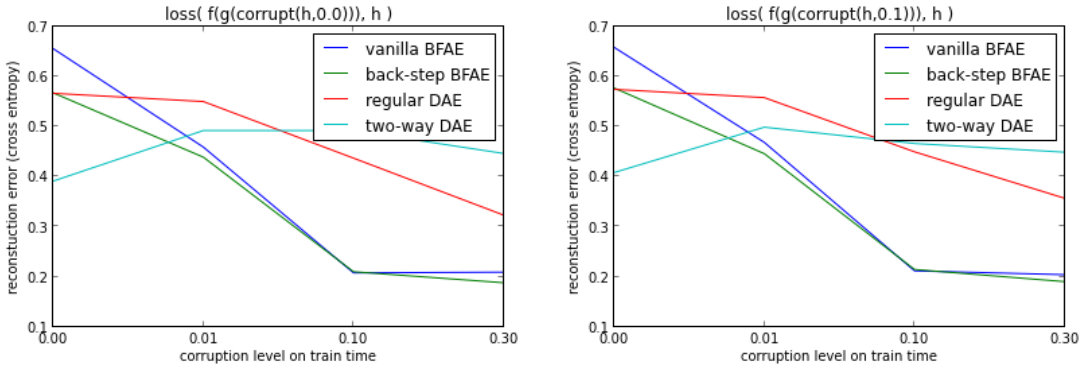
Figure 4: Denoising reconstruction error on input x for various levels of corruption at training time (horizontal axis) and at test time (0 noise in (a), p=0.1 in (b)). The back-step BFAE is slightly worse than the DAEs on x reconstruction, with or without test noise, across training noise conditions. The back-step BFAE is better than the vanilla BFAE which is also the regular recirculation algorithm when the training noise level is 0.

3.2 Experimental Setup

To show that our Backprop-Free Auto-Encoders act like the recirculation, At first we compare 4 ways of training the same model:

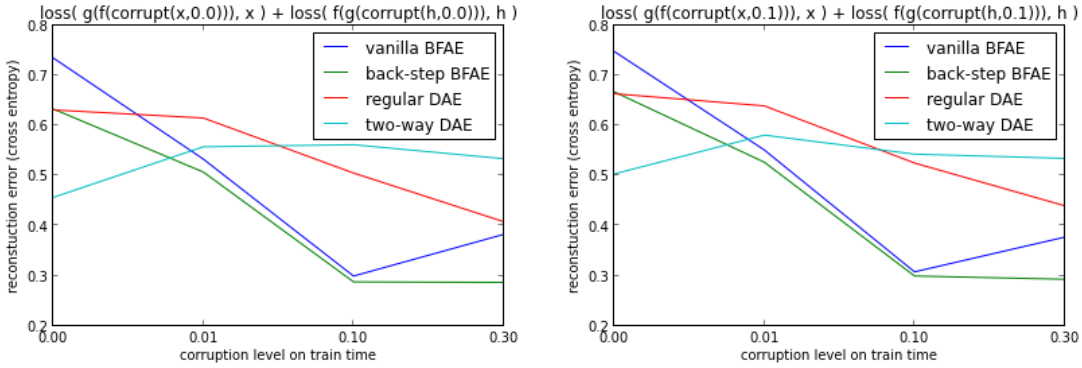
- the recirculation model with regression ($\lambda = 0.75$),
- the recirculation model without regression (i.e., $\lambda = 0$),
- vanilla backprop-free auto-encoder (vanilla BFAE) and
- back-step backprop-free auto-encoder (back-step BFAE).

We choose the learning rate among $\{0.005, 0.01, 0.1, 1\}$ according to validation set reconstruction error (squared error) in each experiment. We use 1000 hidden units and train over 1000 epochs without corruption. According to the recirculation paper, we use sigmoid encoding units and linear



(a) test-time corruption level p = 0.0 (b) test-time corruption level p = 0.1

Figure 5: Denoising reconstruction error on representation h for various levels of corruption at training time (horizontal axis) and at test time (0 noise in (a), p=0.1 in (b)). The back-step BFAE is much better than the regular DAE on h reconstruction, and slightly better than the two-way DAE, in both noise conditions and across training noise conditions. or without test noise.



(a) test-time corruption level p = 0.0 (b) test-time corruption level p = 0.1

Figure 6: Denoising reconstruction error on both representation h and input x, for various levels of corruption at training time (horizontal axis) and at test time (0 noise in (a), p=0.1 in (b)). The back-step BFAE is much better than the regular DAE on h reconstruction, and slightly better than the two-way DAE, in both noise conditions and across training noise conditions. or without test noise. It is also slightly better than the regular BFAE (which is basically the regular recirculation algorithm, in the 0 training noise condition).

decoding units. We use squared error loss on x (linear decoding units), cross entropy on h (sigmoid encoding units). In figure 3, we estimate reconstruction error on x and h with test-time corruption (gaussian noise) $std \in \{0.0, 0.01, 0.1\}$.

Then, in order to compare our model to well-known DAE, we compare 4 ways of training the same model:

- as a regular denoising auto-encoder (DAE) trained with backprop,
- as a DAE with the two-way losses (two-way DAE) trained with backprop,
- vanilla backprop-free auto-encoder (vanilla BFAE), and
- back-step backprop-free auto-encoder (back-step BFAE).

We choose the learning rate among $\{1, 10, 100\}$ according to the validation set reconstruction error in each experiment. [1] We use 1000 hidden units and train over 1000 epochs. We use salt-and-pepper corruption with probability of corrupting a bit of $p \in \{0.0, 0.01, 0.1, 0.3\}$ at training time, and $p \in \{0.0, 0.1\}$ at test time.

3.3 Experimental Results

Figure 3 shows that the BFAEs perform generally better (except at noise level 0.1 for x reconstruction) compared to the recirculation variants. At first, we estimate reconstruction error on x with test-time corruption $p \in \{0.0, 0.1\}$. In Figure 4, we see that DAEs are better than BFAEs in terms of x-reconstruction, but in Figure 5 we see that regular DAEs are worse than BFAEs in terms of h-reconstruction. However, the back-step BFAE is only slightly worse than DAEs in terms of x-reconstruction but much better in terms of h-reconstruction, while the back-step BFAE is almost equivalent to the DAEs for x-reconstruction but clearly stronger in terms of h-reconstruction. Putting these two losses together in Figure 6, we see that in terms of the total 2-way reconstruction error, BFAEs are much better than regular DAEs and 2-way DAEs, even though we train the two-way DAE with the two-way loss on x and h using back-propagation.

4 Conclusion

We investigate Backprop-free Auto-Encoder, which can be trained without back-propagation. In our experiment, we showed that our model inspired by the recirculation can train (denoising) auto-encoding functions successfully. The denoising performance outperforms the original recirculation, Moreover, it is competitive with well-known Denoising Auto-Encoder. Meanwhile the denoising performance on (h, x) is better than DAE and DAE with the two-way loss. This model could be useful for hardware implementations, as a model for computational neuroscience, or to handle non-differentiable activation functions. In addition, this is also building blocks towards replacing back-propagation by target propagation, according to the approach proposed by Bengio (2014).

Acknowledgments

The author would like to acknowledge the support of the following agencies for research funding and computing support: NSERC, Calcul Québec, Compute Canada, the Canada Research Chairs and CIFAR.

References

Alain, G. and Bengio, Y. (2013). What regularized auto-encoders learn from the data generating distribution. In *International Conference on Learning Representations (ICLR'2013)*.

Bengio, Y. (2014). How auto-encoders could provide credit assignment in deep networks via target propagation. Technical report, arXiv preprint arXiv:1407.7906.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy layer-wise training of deep networks. In *NIPS'2006*.

Bengio, Y., Yao, L., Alain, G., and Vincent, P. (2013a). Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems 26 (NIPS'13)*. NIPS Foundation (http://books.nips.cc).

Bengio, Y., Courville, A., and Vincent, P. (2013b). Unsupervised feature learning and deep learning: A review and new perspectives. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*.

Bengio, Y., Thibodeau-Laufer, E., Alain, G., and Yosinski, J. (2014). Deep generative stochastic networks trainable by backprop. In *Proceedings of the 30th International Conference on Machine Learning (ICML'14)*.

Crick, F. (1989). The recent excitement about neural networks. *Nature*, **337**, 129–132.

[1] the learning rates are large because we use averaged cross entropy per unit. Otherwise, these values would be approximately $\{0.001, 0.01, 0.1\}$.

Hinton, G. E. and McClelland, J. L. (1988). Learning representations by recirculation. In *NIPS'1987*, pages 358–366.

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

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

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Machine Learning Res.*, **11**.

Zipser, D. and Andersen, R. (1988). A back propagation programmed network that simulates response properties of a subset of posterior parietal neurons. *Nature*, **331**, 679–684.