4

Deep Autoencoders — Unsupervised Learning

This section and the next two will each select one prominent example deep network for each of the three categories outlined in Section 3. Here we begin with the category of the deep models designed mainly for unsupervised learning.

4.1 Introduction

The deep autoencoder is a special type of the DNN (with no class labels), whose output vectors have the same dimensionality as the input vectors. It is often used for learning a representation or effective encoding of the original data, in the form of input vectors, at hidden layers. Note that the autoencoder is a nonlinear feature extraction method without using class labels. As such, the features extracted aim at conserving and better representing information instead of performing classification tasks, although sometimes these two goals are correlated.

An autoencoder typically has an input layer which represents the original data or input feature vectors (e.g., pixels in image or spectra in speech), one or more hidden layers that represent the transformed feature, and an output layer which matches the input layer for reconstruction. When the number of hidden layers is greater than one, the autoencoder is considered to be deep. The dimension of the hidden layers can be either smaller (when the goal is feature compression) or larger (when the goal is mapping the feature to a higher-dimensional space) than the input dimension.

An autoencoder is often trained using one of the many back-propagation variants, typically the stochastic gradient descent method. Though often reasonably effective, there are fundamental problems when using back-propagation to train networks with many hidden layers. Once the errors get back-propagated to the first few layers, they become minuscule, and training becomes quite ineffective. Though more advanced back-propagation methods help with this problem to some degree, it still results in slow learning and poor solutions, especially with limited amounts of training data. As mentioned in the previous chapters, the problem can be alleviated by pre-training each layer as a simple autoencoder [28, 163]. This strategy has been applied to construct a deep autoencoder to map images to short binary code for fast, content-based image retrieval, to encode documents (called semantic hashing), and to encode spectrogram-like speech features which we review below.

4.2 Use of deep autoencoders to extract speech features

Here we review a set of work, some of which was published in [100], in developing an autoencoder for extracting binary speech codes from the raw speech spectrogram data in an unsupervised manner (i.e., no speech class labels). The discrete representations in terms of a binary code extracted by this model can be used in speech information retrieval or as bottleneck features for speech recognition.

A deep generative model of patches of spectrograms that contain 256 frequency bins and 1, 3, 9, or 13 frames is illustrated in Figure 4.1. An undirected graphical model called a Gaussian-Bernoulli RBM is built that has one visible layer of linear variables with Gaussian noise and one hidden layer of 500 to 3000 binary latent variables. After learning the Gaussian-Bernoulli RBM, the activation

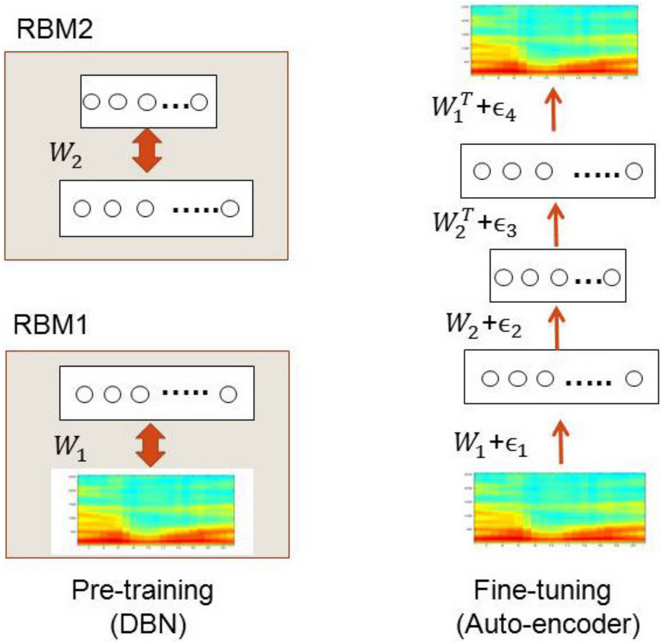

Figure 4.1: The architecture of the deep autoencoder used in [100] for extracting binary speech codes from high-resolution spectrograms. [after [100], @Elsevier].

probabilities of its hidden units are treated as the data for training another Bernoulli-Bernoulli RBM. These two RBM's can then be composed to form a deep belief net (DBN) in which it is easy to infer the states of the second layer of binary hidden units from the input in a single forward pass. The DBN used in this work is illustrated on the left side of Figure 4.1, where the two RBMs are shown in separate boxes. (See more detailed discussions on the RBM and DBN in Section 5).

The deep autoencoder with three hidden layers is formed by "unrolling" the DBN using its weight matrices. The lower layers of this deep autoencoder use the matrices to encode the input and the upper layers use the matrices in reverse order to decode the input. This deep autoencoder is then fine-tuned using error back-propagation to minimize the reconstruction error, as shown on the right side of Figure 4.1. After learning is complete, any variable-length spectrogram can be encoded and reconstructed as follows. First, N consecutive overlapping frames of 256-point log power spectra are each normalized to zero-mean and unit-variance across samples per feature to provide the input to the deep autoencoder. The first hidden layer then uses the logistic function to compute real-valued activations. These real values are fed to the next, coding layer to compute "codes." The real-valued activations of hidden units in the coding layer are quantized to be either zero or one with 0.5 as the threshold. These binary codes are then used to reconstruct the original spectrogram, where individual fixed-frame patches are reconstructed first using the two upper layers of network weights. Finally, the standard overlap-and-add technique in signal processing is used to reconstruct the full-length speech spectrogram from the outputs produced by applying the deep autoencoder to every possible window of N consecutive frames. We show some illustrative encoding and reconstruction examples below.

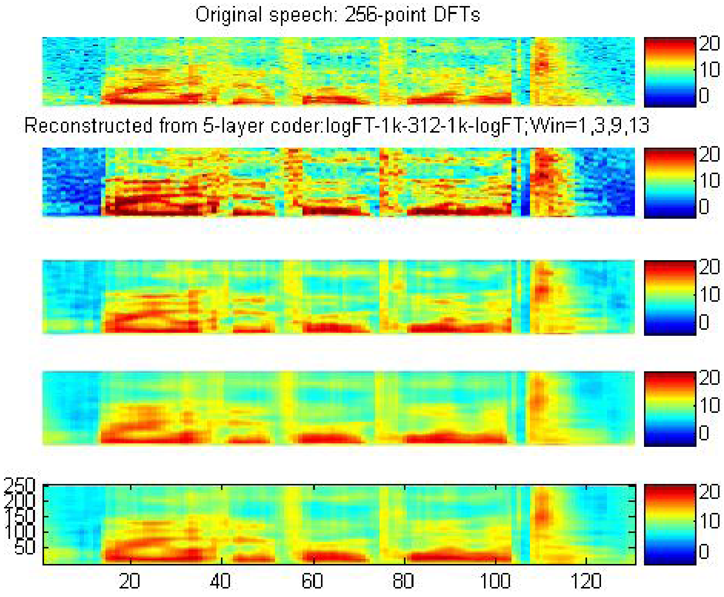

Figure 4.2: Top to Bottom: The ordinal spectrogram; reconstructions using input window sized of $N = 1, 3, 9,$ and 13 while forcing the coding units to take values of zero one (i.e., a binary code) . [after [100], @Elsevier].

At the top of Figure 4.2 is the original, un-coded speech, followed by the speech utterances reconstructed from the binary codes (zero or one) at the 312 unit bottleneck code layer with encoding window lengths of $N = 1, 3, 9$, and 13, respectively. The lower reconstruction errors for $N = 9$ and $N = 13$ are clearly seen.

Encoding error of the deep autoencoder is qualitatively examined in comparison with the more traditional codes via vector quantization (VQ). Figure 4.3 shows various aspects of the encoding errors. At the top is the original speech utterance's spectrogram. The next two spectrograms are the blurry reconstruction from the 312-bit VQ and the much more faithful reconstruction from the 312-bit deep autoencoder. Coding errors from both coders, plotted as a function of time, are

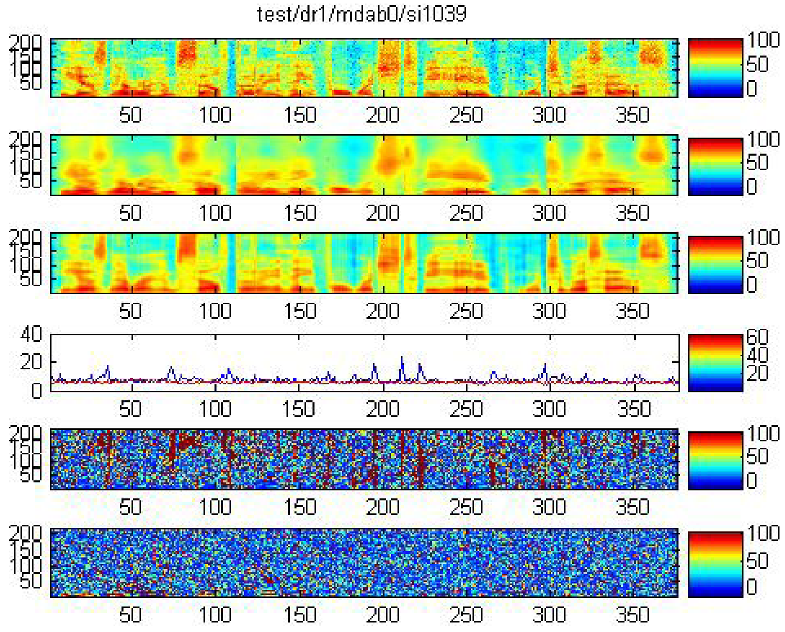

Figure 4.3: Top to bottom: The original spectrogram from the test set; reconstruction from the 312-bit VQ coder; reconstruction from the 312-bit autoencoder; coding errors as a function of time for the VQ coder (blue) and autoencoder (red); spectrogram of the VQ coder residual; spectrogram of the deep autoencoder's residual. [after [100], @ Elsevier].

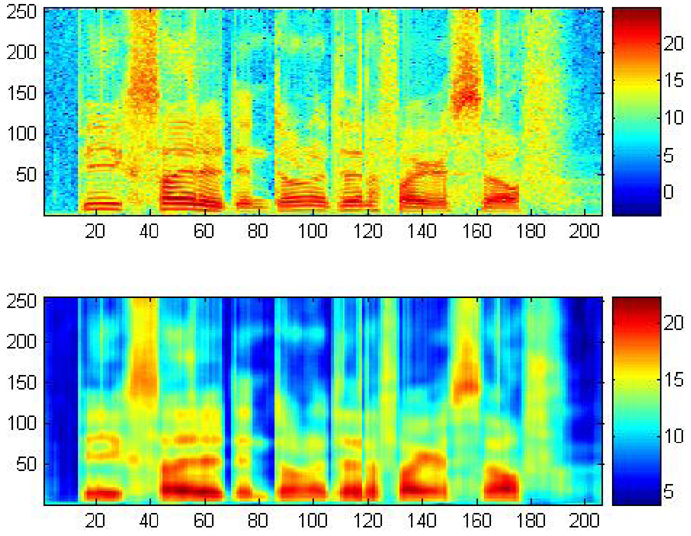

Figure 4.4: The original speech spectrogram and the reconstructed counterpart. A total of 312 binary codes are with one for each single frame.

shown below the spectrograms, demonstrating that the autoencoder (red curve) is producing lower errors than the VQ coder (blue curve) throughout the entire span of the utterance. The final two spectrograms show detailed coding error distributions over both time and frequency bins.

Figures 4.4 to 4.10 show additional examples (unpublished) for the original un-coded speech spectrograms and their reconstructions using the deep autoencoder. They give a diverse number of binary codes for either a single or three consecutive frames in the spectrogram samples.

4.3 Stacked denoising autoencoders

In early years of autoencoder research, the encoding layer had smaller dimensions than the input layer. However, in some applications, it is desirable that the encoding layer is wider than the input layer, in which case techniques are needed to prevent the neural network from learning the trivial identity mapping function. One of the reasons for using a

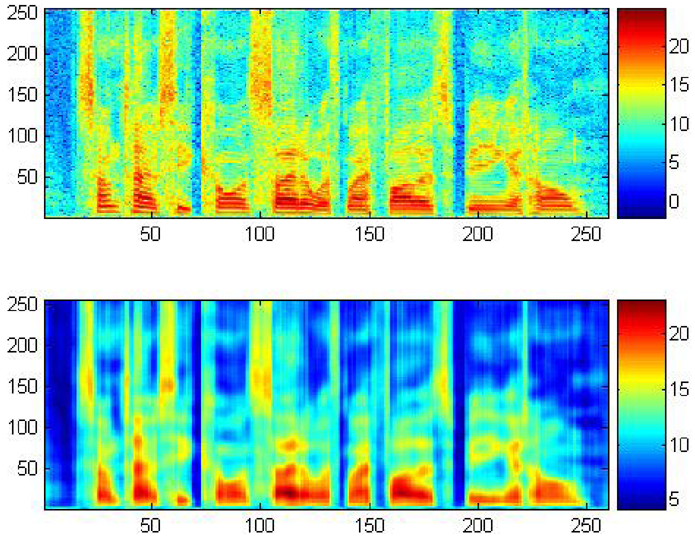

Figure 4.5: Same as Figure 4.4 but with a different TIMIT speech utterance.

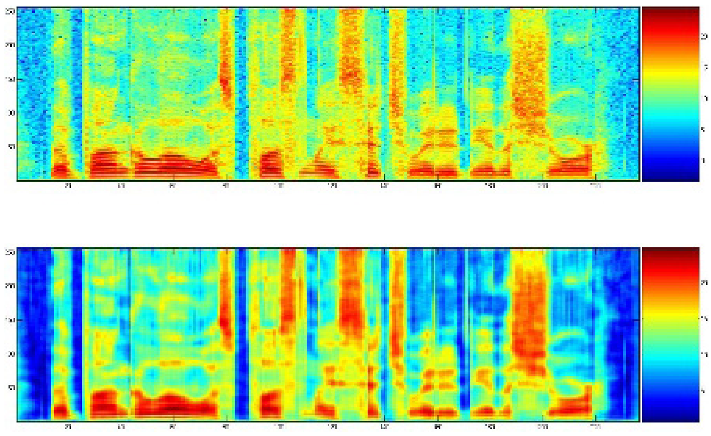

Figure 4.6: The original speech spectrogram and the reconstructed counterpart. A total of 936 binary codes are used for three adjacent frames.

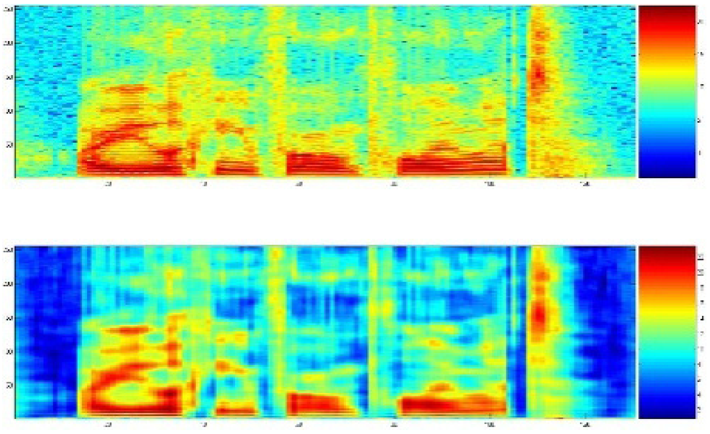

Figure 4.7: Same as Figure 4.6 but with a different TIMIT speech utterance.

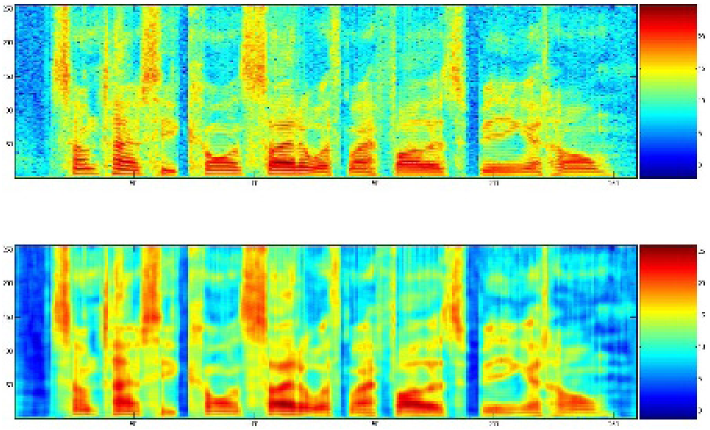

Figure 4.8: Same as Figure 4.6 but with yet another TIMIT speech utterance.

higher dimension in the hidden or encoding layers than the input layer is that it allows the autoencoder to capture a rich input distribution.

The trivial mapping problem discussed above can be prevented by methods such as using sparseness constraints, or using the "dropout" trick by randomly forcing certain values to be zero and thus introducing distortions at the input data [376, 375] or at the hidden layers [166]. For

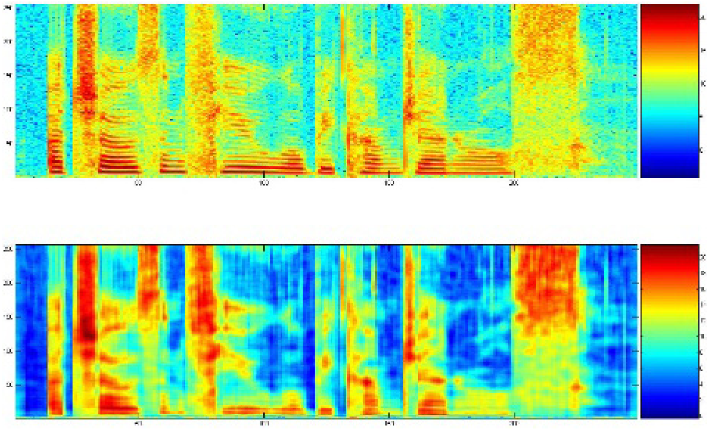

Figure 4.9: The original speech spectrogram and the reconstructed counterpart. A total of 2000 binary codes with one for each single frame.

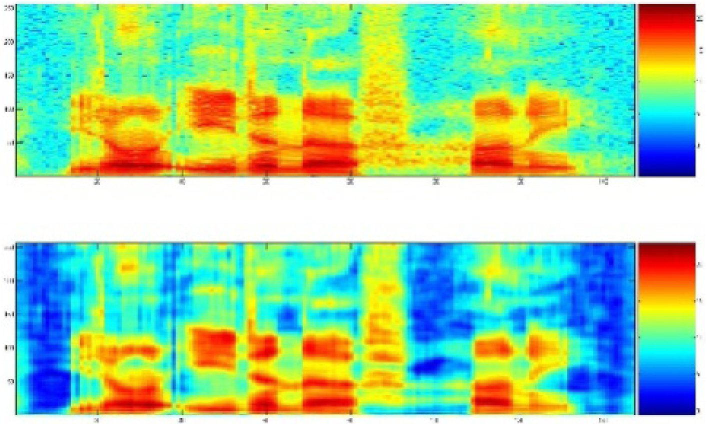

Figure 4.10: Same as Figure 4.9 but with a different TIMIT speech utterance.

example, in the stacked denoising autoencoder detailed in [376], random noises are added to the input data. This serves several purposes. First, by forcing the output to match the original undistorted input data the model can avoid learning the trivial identity solution. Second, since the noises are added randomly, the model learned would be robust to the same kind of distortions in the test data. Third, since each distorted input sample is different, it greatly increases the training set size and thus can alleviate the overfitting problem.

It is interesting to note that when the encoding and decoding weights are forced to be the transpose of each other, such denoising autoencoder with a single sigmoidal hidden layer is strictly equivalent to a particular Gaussian RBM, but instead of training it by the technique of contrastive divergence (CD) or persistent CD, it is trained by a score matching principle, where the score is defined as the derivative of the log-density with respect to the input [375]. Furthermore, Alain and Bengio [5] generalized this result to any parameterization of the encoder and decoder with squared reconstruction error and Gaussian corruption noise. They show that as the amount of noise approaches zero, such models estimate the true score of the underlying data generating distribution. Finally, Bengio et al. [30] show that any denoising autoencoder is a consistent estimator of the underlying data generating distribution within some family of distributions. This is true for any parameterization of the autoencoder, for any type of information-destroying corruption process with no constraint on the noise level except being positive, and for any reconstruction loss expressed as a conditional log-likelihood. The consistency of the estimator is achieved by associating the denoising autoencoder with a Markov chain whose stationary distribution is the distribution estimated by the model, and this Markov chain can be used to sample from the denoising autoencoder.

4.4 Transforming autoencoders

The deep autoencoder described above can extract faithful codes for feature vectors due to many layers of nonlinear processing. However, the code extracted in this way is transformation-variant. In other words, the extracted code would change in ways chosen by the learner when the input feature vector is transformed. Sometimes, it is desirable to have the code change predictably to reflect the underlying transformation-invariant property of the perceived content. This is the goal of the transforming autoencoder proposed in [162] for image recognition.

The building block of the transforming autoencoder is a “capsule,” which is an independent sub-network that extracts a single parameterized feature representing a single entity, be it visual or audio. A transforming autoencoder receives both an input vector and a target output vector, which is transformed from the input vector through a simple global transformation mechanism; e.g., translation of an image and frequency shift of speech (the latter due to the vocal tract length difference). An explicit representation of the global transformation is assumed known. The coding layer of the transforming autoencoder consists of the outputs of several capsules.

During the training phase, the different capsules learn to extract different entities in order to minimize the error between the final output and the target.

In addition to the deep autoencoder architectures described here, there are many other types of generative architectures in the literature, all characterized by the use of data alone (i.e., free of classification labels) to automatically derive higher-level features.