

Autoencoders

Walter Hugo Lopez Pinaya^{1,2}, Sandra Vieira¹,
Rafael Garcia-Dias¹, Andrea Mechelli¹

¹ Department of Psychosis Studies, Institute of Psychiatry, Psychology & Neuroscience, King's College London, London, United Kingdom; ² Centre of Mathematics, Computation, and Cognition, Universidade Federal do ABC, Santo André, São Paulo, Brazil

11.1 Introduction

In the last decade, machine learning has led to impressive advances in medical data analysis (He et al., 2019; Topol, 2019; Yu, Beam, & Kohane, 2018). Despite its notable successes, the application of machine learning algorithms in real-world clinical practice is still limited. Most applications of machine learning to medical data still rely on the handcrafted extraction of features from the original data. Once a good feature representation is provided, a machine learning algorithm can perform well. The process of extracting a feature representation from the data, known as *feature engineering*, requires specialized expertise and is slow and laborious. It is also specific to each type of data, which means that there are no standard approaches that can be generalized to different data modalities. In light of the above shortcomings, there is much interest in the development of algorithms that can automatically learn useful features from the data, thereby simplifying the process of feature engineering.

The autoencoder is a neural network that is part of a larger family of representation learning methods capable of learning features from unlabeled data automatically. These methods are designed to map the input data to an internal latent representation, which is then used to produce an output similar to the input data (i.e., it attempts to copy its input to its output). Through unsupervised learning (see Chapter 1), the autoencoder automatically uncovers the underlying structure of the data by disentangling sources of variation in the input data. The resulting latent representation can then be used to retrieve relevant information when

building classifiers or other predictors. Therefore, the use of appropriate training data to develop a latent representation replaces the need of extensive feature engineering. The features generated via this approach tend to be comparable to those extracted via best handcrafted engineering in terms of algorithm performance (Mehrotra & Musolesi, 2018; Miotto, Li, Kidd, & Dudley, 2016).

The autoencoder was developed in the late 1980s and has been an important part of the historical landscape of neural networks since then (Ballard, 1987; Bourlard & Kamp, 1988). The autoencoders were first developed for the purpose of nonlinear dimensionality reduction. It was mainly a nonlinear extension of the standard linear principal component analysis (PCA) (Kramer, 1991) (see Chapter 12). In the 2000s, the autoencoder was also used to pretrain neural networks (Bengio, Lamblin, Popovici, & Larochelle, 2007). Together with the introduction of restricted Boltzmann machines (Hinton, Osindero, & Teh, 2006), the unsupervised pretraining of neural networks was the breakthrough that led to what is today known as “deep learning.” In this pretraining, deep neural networks (DNNs) had their parameters initialized using weights values learned by autoencoders instead of random values. These pretrained weights initialize the neural network with good intermediary representations of the input data, resulting in a better performance (Vincent, Larochelle, Lajoie, Bengio, & Manzagol, 2010). Recent progress in latent variable models has also brought autoencoders to the forefront of generative modeling (Kingma & Welling, 2013; Makhzani, Shlens, Jaitly, Goodfellow, & Frey, 2015). All these advances allowed autoencoders being used to address noise in images (image denoising) (Creswell & Bharath, 2018; Im, Ahn, Memisevic, & Bengio, 2017), to reconstruct lost or deteriorated parts of images (inpainting) (Yeh et al., 2017), and to enhance the resolution of an image (super-resolution) (Zeng, Yu, Wang, Li, & Tao, 2017).

In this chapter, we present the fundamental concepts of autoencoders. Autoencoders are a particular case of feedforward neural networks and as such can be trained using the same techniques, including gradient descent and backpropagation (see Chapter 9). In the first part of the chapter, we focus on the main characteristics of some of the most commonly used variants of the autoencoder. In the second section of the chapter, we discuss three exemplary applications to brain disorders.

11.2 Method description

An autoencoder is a neural network consisting of two parts: an *encoder* and a *decoder* (Fig. 11.1), each with their own set of learnable parameters. The function of the encoder is to take in an *input* \mathbf{x} (this can be any type of data, such as 2D or 3D images, audio, video, or text) and map it into a

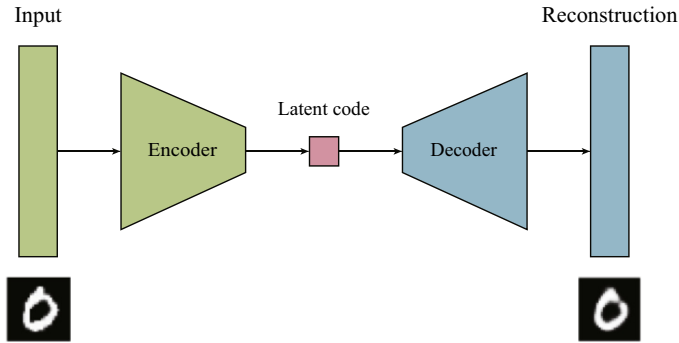


FIGURE 11.1 The structure of the autoencoder. This structure comprises two neural networks, the encoder and the decoder. The input passes through the encoder to produce a latent code; this latent code is then used by the decoder to generate the output. In most autoencoders, the architecture of the decoder is the mirror image of the architecture of the encoder. The goal of the network is to generate an output that is identical to the input. In this example, the autoencoder is given an image of the digit “0” and outputs the reconstructed image from the learned latent code.

latent encoding space, creating a *latent code* \mathbf{h} . For example, the encoder can transform a magnetic resonance imaging (MRI) slice of size 256×256 voxels into a latent code vector of size 50×1 (or any other chosen size). On the other hand, the decoder recovers the input data from the code. In the same example, the decoder takes the encoder’s output \mathbf{h} (vector size 50×1) and tries to create a *reconstruction* \mathbf{x}' of the original 256×256 MRI slice.

The encoder and decoder are neural networks that can be implemented using custom architectures: for example, they can have a shallow design involving just input and output layers or a deeper structure involving a higher number of layers and neurons per layer and a larger code size. By increasing the architecture complexity, the autoencoder will be able to learn more complex latent coding. The encoder and decoder can be fully connected neural networks (see Chapter 9) or, if the data come in a grid format such as image, they can have convolutional and pooling layers to be more efficient (see Chapter 10).

Regardless of the architecture, an autoencoder has one primary objective: *reconstruct its input as accurately as possible*. This goal is achieved using a specific loss function during the training of the model. This loss function, known as *reconstruction loss*, is usually the mean squared error between the output and the input. During training, the loss function penalizes the network for generating outputs different from the input.

$$\text{Loss function} = \|\mathbf{x} - \mathbf{x}'\|^2$$

Attempting to generate a copy of the input may sound useless. However, in representation learning, we are not typically interested in the

output of the decoder. Instead, we hope that by training the autoencoder to generate a copy of the input data, the result will be a latent code with useful properties. Ideally, the resulting code \mathbf{h} will have less correlated/redundant features and/or lower dimensionality than the input data, both essential characteristics to avoid the waste of processing time and the overfitting of machine learning models.

These useful properties can be obtained by applying constraints on the copying task. In the following sections, we present some approaches that have been widely used to enforce such constraints: the undercomplete autoencoder, the denoising autoencoder, the sparse autoencoder, and the adversarial autoencoder.

11.2.1 Undercomplete autoencoder

The simplest way to obtain useful features from the autoencoder is constraining its architecture, by restraining the latent code \mathbf{h} to have a smaller dimension than the input data \mathbf{x} . This type of autoencoder is called *undercomplete*. As the latent code has far fewer neurons than the input, we force the encoder to compress the data. If the input features were all independent from one another, this dimensionality reduction and subsequent reconstruction would be a tough task. However, if there is some underlying structure in the data (i.e., correlations between input features), this can be learned and consequently leveraged when forcing the input through the latent code.

In the undercomplete autoencoder, the encoder essentially projects the data into a lower-dimensional space (performing the task of dimensionality reduction). Indeed, if an autoencoder uses the linear activation function in combination with the mean squared error loss function, the resulting model tends to perform equally to the PCA algorithm. As neural networks are capable of learning nonlinear relationships, this type of autoencoder can be thought of as a more powerful (nonlinear) generalization of PCA.

11.2.2 Denoising autoencoder

The development of the denoising autoencoder was motivated by the human ability to recognize an object even if it is partially occluded or corrupted (Vincent, Larochelle, Bengio, & Manzagol, 2008). During the training of this type of autoencoder, the input is partially corrupted by masking or adding noise to some values of the input vector in a random manner (Fig. 11.2). Then, the model is trained to recover the original noise-free data. Therefore, while the architecture of the denoising autoencoder can be the same as that of the undercomplete variant, the two types of networks differ concerning the training process.

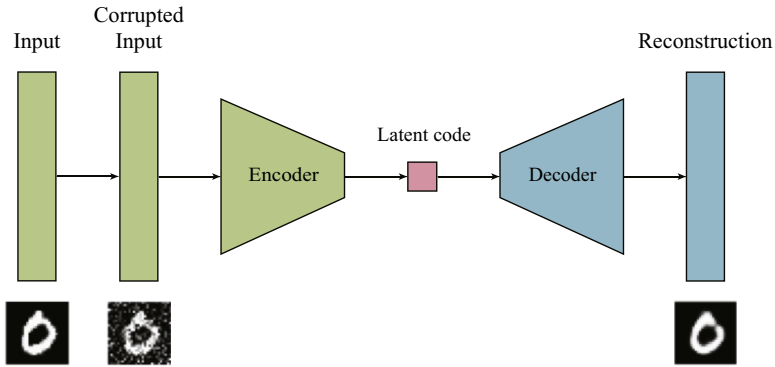


FIGURE 11.2 The structure of the denoising autoencoder. Here, the input to the autoencoder is the noisy data, whereas the expected target is the original noise-free data. The network learns to recognize and remove the noise in the input data before generating the output. The autoencoder does not see the original, noise-free image at all, it only sees the image that has undergone some corruption. This corruption process is flexible; it can be based on masking noise, Gaussian noise, salt-and-pepper noise (shown in the example), or even prior knowledge.

To “repair” the partially destroyed input, the denoising autoencoder has to discover and capture the relationship between input variables and infer the missing pieces. For high dimensional data with high redundancy, such as images, the model is likely to rely on evidence gathered from a combination of many input variables rather than a single input variable. This process represents a good foundation for learning robust latent representation that captures stable dependencies and regularities within the unknown distribution of the input.

11.2.3 Sparse autoencoder

While undercomplete autoencoders discover useful structures by having a small number of hidden neurons in the latent code \mathbf{h} , a different class of autoencoders known as the sparse autoencoders can use a large number of neurons in the latent code and still learn useful representations. The sparse autoencoder achieves this by introducing the so-called “*sparsity constraint*.” This constraint limits the model to have a small number of neurons being activated at the same time (a neuron is activated when its output value is close to 1 and inactivate with a value close to 0). This method works well even if the code \mathbf{h} is large because the autoencoder is still forced to represent each input as a combination of a small number of neurons, enabling the model to discover interesting structures in the data.

There are many approaches to impose sparsity on the latent code. One strategy is to add a term in the loss function that penalizes the absolute

value of the activations (L1-norm). Note that this is a different approach to the regularization strategies used to address the issue of overfitting in which we regularize the weights of the network rather than its activations. A different approach is to add a Kullback–Leibler divergence (KL-divergence) term in the loss function that measures the difference between two probability distributions. Here, the KL-divergence calculates the difference between the average activation of the neurons and a distribution function which takes the value 1 with a small probability p and the value 0 with a probability $1-p$ (e.g., a Bernoulli distribution). If the neurons' activation is not close to the desired sparsity rate (defined by p), the difference between the distributions will be high, producing a loss function with a high value which penalizes this network's solution.

11.2.4 Adversarial autoencoder

A limitation of the previous autoencoders is that they do not generate a well-structured latent space. This is because, during training, different observations from the dataset end up being encoded randomly scattered across the latent space. As a result, the positioning of these latent representations sometimes produces empty spaces in the latent space (Fig. 11.3). These empty spaces contain codes \mathbf{h} that the decoder was never trained to reconstruct, and for this reason the process of generating new data from a randomly chosen latent code \mathbf{h} can be challenging. Indeed, the reconstruction of samples from these empty spaces tends to be flawed, as

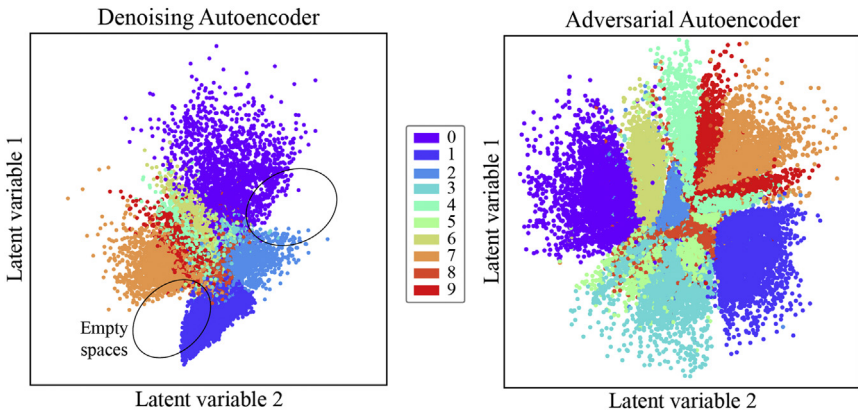


FIGURE 11.3 Representations in the latent space. In this example, images of digits from 0 to 9 are used to train a denoising autoencoder and an adversarial autoencoder. In the two maps, the training data are represented using two latent variables. Because we do not apply any constraint to the latent space of the denoising autoencoder, its latent space tends to create empty spaces. On the other hand, the adversarial autoencoder constrains the representation to be similar to a prior distribution (in this case, a two-dimensional Gaussian distribution), resulting in no empty spaces.

the autoencoder does not interpolate well along the groups of seen latent codes. This flawed reconstruction limits the application of the previous autoencoders as generative models.

Thankfully, recent advances have led to new methods that allow us to control the structure of the latent space. One such approach is the *adversarial autoencoder*. This variant of autoencoder is a blend of the general autoencoder framework with the notion of adversarial training which is typical of the generative adversarial networks (Goodfellow et al., 2014).

In brief, generative adversarial networks are models comprised of two neural networks that compete against each other to improve themselves (Fig. 11.4). The first one is the *generator*; it receives random values as inputs and tries to generate synthetic data as similar as possible to the true dataset. The second one is the *discriminator*; it decides if its input comes from the generator or the true dataset. A common analogy is to think of the generator as an art forger and the discriminator as an art expert. The forger creates forgeries, with the aim of making realistic images. The expert receives both imitations and authentic paintings and aims to tell them apart. Both networks are trained simultaneously, and the generator has no direct access to real images—the only way to improve itself is through its interaction with the discriminator. Based on the feedback from the discriminator, the generator will try to adjust its network accordingly, to produce outputs that are more difficult for the discriminator to identify as a forgery, i.e., that are more similar to the true data.

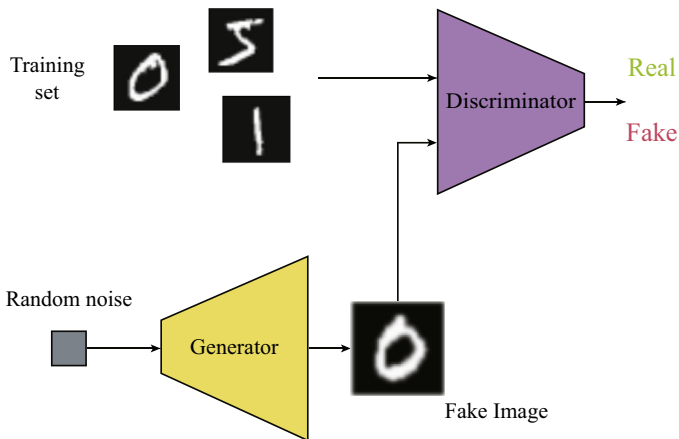


FIGURE 11.4 The structure of the generative adversarial network. To train the model, the generator and the discriminator compete with each other to improve their performance. In this competition, the generator tries to produce an input that will be misinterpreted as real by the discriminator, whereas the discriminator has to decide if each input comes from the generator or the true dataset and in doing so guides the generator to produce more realistic input.

Adversarial autoencoders use this adversarial training to shape the distribution of the encoded input data to look as similar as possible to a predefined prior distribution. This is achieved by adding the discriminator network into the autoencoder framework (Fig. 11.5).

In adversarial autoencoders, the discriminator receives two types of inputs: values sampled from the desired distribution (e.g., random values sampled from a Gaussian distribution) and latent code \mathbf{h} from observations of the training set. Importantly, both have the same dimension. Here, the desired distribution can be thought of as a prior distribution. During the training process, the discriminator will make a classification decision regarding whether its input data are sampled from the prior distribution or the latent code. Once we include this discriminator, the encoder is forced to do two things simultaneously: (i) it has to provide a latent code space that the decoder can work with to create the reconstruction of the input; (ii) it has to produce a latent code space that can fool the discriminator into thinking that the encoded samples (the forgeries) are just another sample from the prior distribution (the authentic paintings). In other words, the encoder is also the generator.

To implement this, the training of the adversarial autoencoder has three steps per epoch:

1. The autoencoder updates the encoder and the decoder parameters based on the standard reconstruction loss.

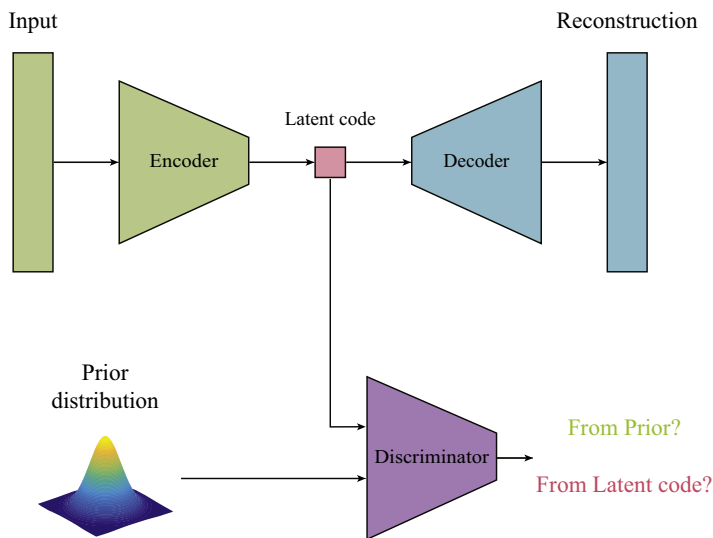


FIGURE 11.5 The structure of the adversarial autoencoder. The discriminator network is added to the autoencoder to force the encoder to produce latent codes similar to the prior distribution.

2. The model updates the discriminative network parameters to tell apart the true samples (generated using the prior distribution) from the generated samples (the latent codes created by the encoder).
3. The encoder uses the feedback error backpropagated through the discriminator to update its parameters. This feedback enables the encoder to improve the generation of latent codes that fool the discriminator network.

If training succeeds, the encoder learns to convert the input data distribution to the prior distribution, the discriminator would always be unsure of whether its inputs are real or not, and the decoder learns a generative model that maps the imposed prior distribution to the data distribution. Finally, it is possible to generate new data by sampling from the well-mapped prior distribution, feeding it to your decoder, and letting it produce the output.

11.3 Applications to brain disorders

Historically, autoencoders have been extensively used to pretrain neural networks. The process of pretraining is aimed at providing the neural network parameters with a better initialization point, to improve the performance of the neural network training process. Pretraining was introduced by [Hinton et al. \(2006\)](#) as a crucial step to train DNNs. Nowadays, it tends to be employed as an additional technique to help avoid overfitting irrespective of the depth of the neural network; this can be particularly useful in the case of small sample sizes—a common issue in the medical field including psychiatry and neurology. It is therefore unsurprising that an increasing number of clinical neuroimaging studies are using autoencoders for pretraining machine learning models. Two other important applications of the autoencoder include the automatic acquisition of useful features from the input data as well as dimensionality reduction. These applications have great relevance to brain disorders research due to the high dimensionality of genetic and neuroimaging data and the complexity involved in the creation of handcrafted features from such data. In this section, we discuss some examples in which autoencoders have been applied to the investigation of brain disorders.

11.3.1 Predicting prognosis from electronic health records

The use of electronic health records (EHRs) has been growing exponentially in the last decade. The massive quantity of data created via EHRs offers great promise for the development of precision medicine tools. However, dealing with this type of data can be very challenging. EHRs include a wide range of data, such as demographic information,

diagnoses, laboratory tests and results, prescriptions, radiological images, and natural language free text such as progress notes or discharge summaries. Each type of data contains potentially useful information which needs to be extracted before it can be inputted into the machine learning model. The performance of a model based on EHRs will therefore be largely dependent on the steps of feature engineering.

In Miotto et al. (2016), the authors presented a framework, which they named “deep patient,” designed to learn a deep representation from patients’ data. With this framework, first EHRs were preprocessed to identify and normalize clinically relevant phenotypes. Then, the authors grouped the data resulting in the creation of vectors named “raw representation.” Finally, the collection of vectors obtained from all the patients was used as input for the feature learning algorithm to discover a set of high-level descriptors. For this step, Miotto and colleagues used a DNN composed by stacked denoising autoencoders (Vincent et al., 2008, 2010), which comprised of three layers, where each layer was independently trained (Fig. 11.6). At the beginning of training, the first denoising autoencoder was trained as usual in the input data. Here, the autoencoder learned to capture stable structures and regular patterns in the input data. After training, the parameters of the encoder were used to create a first-level representation of the input data. Then, a second denoising autoencoder was trained on this first-level representation. As a result, this second autoencoder learned representations of a higher order. Again, after training, the encoder was used to propagate the first-level representation to a deeper representation. Finally, the same process was

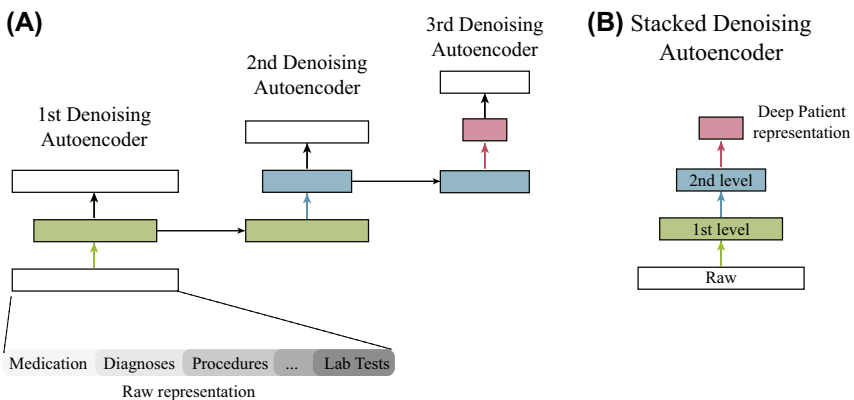


FIGURE 11.6 Overview of the deep patient model used in Miotto et al. (2016). (A) The training process for each denoising autoencoder. Here, the first autoencoder uses raw representation as input; the latent code of these raw representations is used to train the second denoising autoencoder, and the latent code generated by the second denoising autoencoder is used to train the third autoencoder. (B) The encoder parameters of all autoencoders are combined to create the stacked denoising autoencoder.

performed with a third denoising autoencoder. After combining the encoders from the three denoising autoencoders, the authors obtained the stacked denoising autoencoder. In this final model, every level creates a representation of the input pattern that was more abstract than the previous level.

Finally, the authors evaluated their feature learning algorithm by predicting a patient's future disease. Using a dataset of about 1.2 million patients, with an average of 88.9 records per patient, it was possible to classify the raw representation and the deep patient deepest representation into 78 different diseases, including diseases of the nervous system, as well as mental illness (e.g., Parkinson's disease, attention-deficit disorder, schizophrenia). The performance of the deep patient representations was superior to the one obtained using the raw representations (15% improvement in the AUC-ROC). Specifically, for brain disorders, deep patient representations achieved an AUC-ROC of 0.75 for Parkinson's disease, 0.77 for attention-deficit disorders, and 0.86 for schizophrenia. This indicates that the use of the autoencoder allowed the automatic extraction of the information that was critical for predicting future disease. In addition to prognostic classification, this approach could be useful for other clinically relevant tasks such as personalized prescriptions, treatment recommendations, and recruitment for clinical trials.

11.3.2 Diagnosis of autism spectrum disorder

Autism spectrum disorder (ASD) is a complex neurodevelopmental disorder characterized by repetitive and restricted behaviors, as well as impaired social and communication skills. Currently, the diagnosis of ASD is primarily based on clinical assessment. However, the combination of clinical heterogeneity and individual variation in the timing of developmental changes means that the diagnosis and definition of ASD is still a clinical challenge. Several neuroimaging studies have employed machine learning methods in an attempt to identify objective biomarkers that could aid in the decision-making process involved in the diagnosis and treatment of ASD. For example, in [Heinsfeld, Franco, Craddock, Buchweitz, and Meneguzzi \(2018\)](#), the authors used DNNs to distinguish between healthy controls and ASD, as well as to investigate the neural patterns associated with the disorder. In this chapter, the authors combined the traditional supervised learning of neural networks with the unsupervised pretraining step using autoencoders.

As discussed in Chapter 9, the training of a DNN involves changing the weights of the network to reflect the importance of the input features and how these interact to produce the output. DNNs are typically initialized with random weights, i.e., the initial weights are set to an arbitrary value; this means the network knows nothing about the problem at hand before the training step. However, DNNs can also initialize its parameters using

pretrained ones, i.e., the network has already learned something useful about the data and therefore is not starting the learning process from scratch. This means that, during training, the network does not have to worry about learning the basic properties of the data and will fine-tune the features that are relevant to the task.

In their study, [Heinsfeld et al. \(2018\)](#) trained a DNN using a large population sample of brain imaging data, the Autism Imaging Data Exchange I (ABIDE I) ([Di Martino et al., 2014](#)). The dataset comprised of 505 ASD individuals and 530 matched controls recruited at 17 different sites. Using resting-state functional MRI data, the authors obtained a functional connectivity matrix for each participant based on an atlas with 200 different brain regions, resulting in 19,900 variables.

To pretrain the DNN, [Heinsfeld et al. \(2018\)](#) used a two-layer stacked denoising autoencoder ([Vincent et al., 2010](#)) trained on one layer at a time ([Fig. 11.7](#)). Once trained, the parameters of the encoder were used to initialize the DNN, which was subsequently fine-tuned to perform the classification of the subjects into patients and control categories.

The resulting DNN achieved a mean classification accuracy of 70% in the cross-validation process. Compared to shallow models, the pretrained DNN showed improvements of 5% and 7% in mean accuracy relative to Support Vector Machine and random forests, respectively. To evaluate classifier performance across sites, [Heinsfeld et al. \(2018\)](#) also performed a

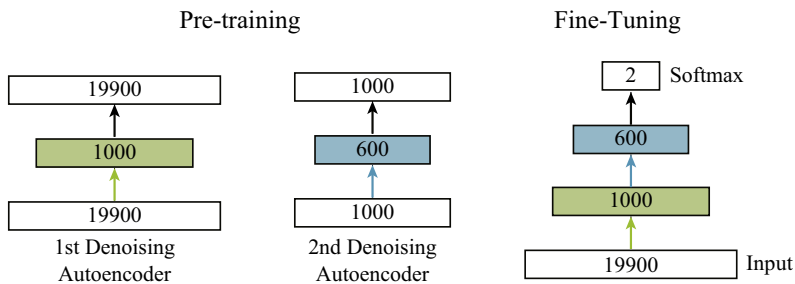


FIGURE 11.7 Pretraining and fine-tuning process using stacked denoising autoencoder performed in [Heinsfeld et al. \(2018\)](#). In the pretraining process, the first denoising autoencoder maps the input data (with 19,900 neurons) to a latent code (with 1000 neurons). This latent representation is then used as input to the second autoencoder that generates a second-level representation (with 600 neurons). In the fine-tuning process, the encoders' parameters are combined in a single structure, the stacked denoising autoencoder. Then, an output layer is added at the top of the stacked denoising autoencoder. This output layer is responsible for classifying the latent representations of the input data based on their probability of belonging to each group (in this case, the groups are represented by two output neurons with the softmax activation). Finally, the neural network is trained to perform classification using the gradient descent algorithm (Chapter 9).

leave-one-site-out cross-validation process, with accuracies ranging from 63% to 68%. Taken collectively, these results suggest that deep learning methods pretrained using autoencoders may reliably classify big multi-site datasets.

11.3.3 Mobility features for predicting depressive states

Mobile phones have changed from merely communication tools to an indispensable part of daily life. These devices host an array of sensors, such as accelerometers, GPS receiver, cameras, and microphones, to name a few. As their owners carry these devices at all time, mobile phones hold the promise of revolutionizing modern healthcare. Recent studies have revealed the potential of using mobile sensing technologies for mental health monitoring (Ben-Zeev, Scherer, Wang, Xie, & Campbell, 2015; Mahmood, Ul-Haq, Batool, Rana, & Tariq, 2016; Torous, Onnela, & Keshavan, 2017). Also, there is much interest in the application of machine learning models to this new kind of data to support the development of predictive models of patient's health and behavior (Mehrotra & Musolesi, 2018; Mikelsons, Smith, Mehrotra, & Musolesi, 2017; Mohr, Zhang, & Schueller, 2017). Mobility data, also called geospatial activity, is a type of information that can be collected via mobiles phones. However, feature engineering for characterizing human mobility remains a heuristic process that relies on expert knowledge. As mentioned at the begin of this chapter, feature engineering can be a complex and time-consuming task, and this is especially the case for mobility features that are still investigated and poorly understood compared to other types of data. For example, there is no guarantee that the crafted features would adequately capture the mobility behavior of the participants that are most relevant to the task. Therefore, approaches that can automatically learn useful features from the data may be particularly useful with this type of data.

In a recent study, Mehrotra and Musolesi (2018) used deep autoencoders to learn mobility features from raw movement data (i.e., GPS trace) passively collected through mobile phones to infer people's future depressive states. Daily GPS data from 28 different subjects from the trajectories of depression dataset (Canzian & Musolesi, 2015) were used to measure the participants' mobility. The Patient Health Questionnaire depression scale (PHQ-8; Kroenke et al., 2009) was used to assess and monitor depression severity every day. Based on the daily score of the PHQ-8, participants were divided into two groups with and without depressed mood for that given day. Every subject had records from a minimum of 71 days. A deep autoencoder was trained on the normalized mobility trajectories to uncover useful features. The resulting latent representation was then presented to three different classifiers (Support

Vector Machine, random forest, and XGBoost), each one trained to predict the future presence or absence of depressed mood. For comparison, the same models were also trained using handcrafted features. These features include total distance covered, the maximum distance between two locations, the radius of gyration, the standard deviation of the displacements, the maximum distance from home, number of different places visited, and others.

The models trained using the autoencoder-based features achieved 90% specificity and 75% sensitivity; this was a significant improvement (i.e., around 8.5% specificity and 10.5% sensitivity) on the performance of the models trained with the handcrafted features. The results of this study—one of the first studies applying autoencoders in complex human mobility behavior—indicate that the use of automatically extracted features not only reduces the burden of the feature engineering step but also can help improve the performance of the prediction model.

11.4 Conclusion

In this chapter, we explored the main principles involved in autoencoders. In the first part, we presented some of its most popular variants, starting from the simplest one—the undercomplete autoencoder—to the more complex adversarial autoencoder that combines the general autoencoder framework with elements from other types of networks. In the second part, we presented examples of how autoencoders have been applied to different types of data in the existing literature.

Autoencoders have been evolving since its creation. The model, initially designed to perform dimensionality reduction, has led to a step change in the field of representational learning by enabling the implementation of the first deep learning models, and is now one of the most promising generative models together with generative adversarial networks. In the coming years, autoencoders are likely to become more popular in brain disorders research for two main reasons. First, they are promising approaches for overcoming the limitations of feature engineering. Second, the potential usefulness of purely supervised learning is likely to be limited because human and animal learning is mostly unsupervised: we discover the structure of the world by observing it, not by being told the category of every object. Therefore, we expect that more research efforts will focus on the type of learning enabled by the autoencoder, i.e., unsupervised learning. This type of learning remains vastly underexplored, in general, and in brain disorders research, in particular, and thus still has significant potential.

11.5 Key points

- Autoencoders are neural networks that can automatically learn useful features from unlabeled data.
- Autoencoder is an artificial neural network designed to output a reconstruction of its input.
- The autoencoder creates a latent code that can represent useful features by adding constraints on its copying task.
- There are several variants of the autoencoder including, for example, the undercomplete autoencoder, the denoising autoencoder, the sparse autoencoder, and the adversarial autoencoder.
- In brain disorders research, autoencoders have been mainly used to reduce the complexity of feature engineering step and to pretrain deep models.

References

- Ballard, D. H. (1987). Modular learning in neural networks. In *Aaai* (pp. 279–284).
- Ben-Zeev, D., Scherer, E. A., Wang, R., Xie, H., & Campbell, A. T. (2015). Next-generation psychiatric assessment: Using smartphone sensors to monitor behavior and mental health. *Psychiatric Rehabilitation Journal*, 38(3), 218.
- Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. In *Advances in neural information processing systems* (pp. 153–160).
- Bourlard, H., & Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59(4–5), 291–294.
- Canzian, L., & Musolesi, M. (2015). Trajectories of depression: Unobtrusive monitoring of depressive states by means of smartphone mobility traces analysis. In *Proceedings of the 2015 ACM international joint conference on pervasive and ubiquitous computing* (pp. 1293–1304). ACM.
- Creswell, A., & Bharath, A. A. (2018). Denoising adversarial autoencoders. *IEEE Transactions on Neural Networks and Learning Systems*, (99), 1–17.
- Di Martino, A., Yan, C.-G., Li, Q., Denio, E., Castellanos, F. X., Alaerts, K., et al. (2014). The autism brain imaging data exchange: Towards a large-scale evaluation of the intrinsic brain architecture in autism. *Molecular Psychiatry*, 19(6), 659–667.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672–2680).
- He, J., Baxter, S. L., Xu, J., Xu, J., Zhou, X., & Zhang, K. (2019). The practical implementation of artificial intelligence technologies in medicine. *Nature Medicine*, 25(1), 30.
- Heinsfeld, A. S., Franco, A. R., Craddock, R. C., Buchweitz, A., & Meneguzzi, F. (2018). Identification of autism spectrum disorder using deep learning and the ABIDE dataset. *NeuroImage: Clinical*, 17, 16–23.
- Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7), 1527–1554. <https://doi.org/10.1162/neco.2006.18.7.1527>.
- Im, D. J., Ahn, S., Memisevic, R., & Bengio, Y. (2017). Denoising criterion for variational auto-encoding framework. In *Aaai* (pp. 2059–2065).

- Kingma, D. P., & Welling, M. (2013). *Auto-encoding variational bayes*. ArXiv Preprint ArXiv: 1312.6114.
- Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2), 233–243.
- Kroenke, K., Strine, T. W., Spitzer, R. L., Williams, J. B. W., Berry, J. T., & Mokdad, A. H. (2009). The PHQ-8 as a measure of current depression in the general population. *Journal of Affective Disorders*, 114(1–3), 163–173.
- Mahmood, K., Ul-Haq, Z., Batool, S. A., Rana, A. D., & Tariq, S. (2016). Application of temporal GIS to track areas of significant concern regarding groundwater contamination. *Environmental Earth Sciences*, 75(1). <https://doi.org/10.1007/s12665-015-4844-2>.
- Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., & Frey, B. (2015). *Adversarial autoencoders*. ArXiv Preprint ArXiv:1511.05644.
- Mehrotra, A., & Musolesi, M. (2018). Using autoencoders to automatically extract mobility features for predicting depressive states. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(3), 127.
- Mikelsons, G., Smith, M., Mehrotra, A., & Musolesi, M. (2017). *Towards deep learning models for psychological state prediction using smartphone data: Challenges and opportunities*. ArXiv Preprint ArXiv:1711.06350.
- Miotto, R., Li, L., Kidd, B. A., & Dudley, J. T. (2016). Deep patient: An unsupervised representation to predict the future of patients from the electronic health records. *Scientific Reports*, 6, 26094.
- Mohr, D. C., Zhang, M., & Schueller, S. M. (2017). Personal sensing: Understanding mental health using ubiquitous sensors and machine learning. *Annual Review of Clinical Psychology*, 13, 23–47.
- Topol, E. J. (2019). High-performance medicine: The convergence of human and artificial intelligence. *Nature Medicine*, 25(1), 44.
- Torous, J., Onnela, J. P., & Keshavan, M. (2017). New dimensions and new tools to realize the potential of RDoC: Digital phenotyping via smartphones and connected devices. *Translational Psychiatry*, 7(3), e1053.
- Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. *Proceedings of the 25th International Conference on Machine Learning*, 1096–1103. <https://doi.org/10.1145/1390156.1390294>.
- Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec), 3371–3408.
- Yeh, R. A., Chen, C., Lim, T.-Y., Schwing, A. G., Hasegawa-Johnson, M., & Do, M. N. (2017). Semantic image inpainting with deep generative models. In *Cvpr* (Vol. 2, p. 4).
- Yu, K.-H., Beam, A. L., & Kohane, I. S. (2018). Artificial intelligence in healthcare. *Nature Biomedical Engineering*, 2(10), 719.
- Zeng, K., Yu, J., Wang, R., Li, C., & Tao, D. (2017). Coupled deep autoencoder for single image super-resolution. *IEEE Transactions on Cybernetics*, 47(1), 27–37.