Variational autoencoder to generate images and music

Fiammetta Strazzera Perniciani
Department of Social Sciences
Radboud University
6525 HP Nijmegen
f.strazzeraperniciani@student.ru.nl

Stefano Gentili
Department of Social Sciences
Radboud University
6525 HP Nijmegen
s.gentili@student.ru.nl

Abstract

Most of the data that could be used to train machine learning models is unlabeled and traditionally difficult to train with neural networks. In the last years, unsupervised learning, that is, algorithms capable of training with unlabeled data, has shown to be one of the most original areas of research, with generative adversarial networks[1] and variational autoencoders[2] being amongst the most well known algorithms recently developed. This paper will present the algorithm for a variational autoencoder neural network, show some results proving their capacity to learn from unlabeled data, and discuss the neuroscientific plausibility of such models.

1 Introduction

Deep Learning is a branch of machine learning[3] that applies learning algorithms to artificial neural networks with several hidden layers. Unlike older machine learning methods, deep learning models are capable of representation learning[4]: they allow any raw data as input and automatically discover the features needed to train the model, so they find ample use in several different fields. In the last few years, deep learning has become the new gold standard in artificial intelligence applications; having achieved state of the art performances in several tasks like pattern recognition, game playing, medical diagnosis, etc.

Most of the success of deep learning has been due to the ability of neural networks to train on labeled data to be able to correctly classify its inputs. But what can we do when we have data with no labels? Is it possible to learn from it? and how? The goal in case of unlabeled data would be to find an underlying hidden structure of the given input in order to learn a useful feature representation of the available data. Many machine learning algorithm approaches have been developed to learn from this typology of datasets, like clustering methods (such as k-clustering or mixture models) or methods learning latent variables of the data (such as principal component analysis or singular value decomposition). Yet recently also neural networks have shown their utility in unsupervised learning, with algorithms such as generative adversarial networks and deep belief networks.

Another recent class of unsupervised learning neural networks called autoencoders can be used to learn a lower dimensional feature representation from unlabelled training data and generate other instances of similar data. An autoencoder consists of two main neural layers (Fig. 1): an encoder, that sequentially deconstructs input data into hidden representations in the latent space, capturing the most important features of the data, and a decoder, that uses those representations to sequentially reconstruct outputs that resembles the original input data. Thus the data goes through a bottleneck: the latent space of the autoencoder captures the most meaningful features of the input data in a lower dimension. This dimensionality reduction can also be seen as a data compression algorithm: the decoder learns to efficiently store a data point in a smaller encoded format, and the decoder is an algorithm that reconstructs this information. Of course they are trained with specific data, so they become proficient in encoding only data of the same typology. In addition, this type of compression

will inevitably lose information of the input data and reconstruct a considerably degraded form of the original images.

Yet this type of algorithm holds many useful characteristics that make it one of the best known unsupervised learning frameworks available. First, it can be used to generate data that detains some specific feature learned from the training (like generating a particular digit). Unlike generative adversarial networks, that generate images from arbitrary noise, autoencoders (specifically variational autoencoders) can generate a specific class of picture that has been encoded in a specific position of the latent space. Indeed the latent feature representations of the original data can be later used to distinguish class labels, with the aid, for example, of a classifier. This can lead to many interesting connections with supervised an semi-supervised learning[5].

Another important area of use of autoencoders is data denoising. The decoder network is fed incomplete data, which however contains only the most important features of the input data and therefore low noise, and is able to reconstruct an image similar to the original. Finally an unique aspect of autoencoders is their ability to generate particular type of data based on the characteristics of the data it was trained on: if it was trained on, say, images of people with glasses, it could modify a complete image of a face without glasses to have glasses, etc.

In this paper we will be focusing on variational autoencoders (VAE), a variant inspired by the Bayesian framework, in which our training data is assumed to be generated from some underlying unobserved latent representation z. The input, the hidden representations and the outputs are treated as probabilistic random variables. The algorithm will be described now in general terms, and more in detail in later sections.

A VAE network can be seen as a general autoencoder with an encoding component that "saves" the data in input in a latent representation, and a decoder that reconstructs it. Yet an important ability of a generative network is its ability to create actual new data instances, learned from the dataset it has been training on, and not just "memorizing and recalling" specific data points. The elegant solution found by VAE networks is to add a specific constraint to the basic autoencoder network: that is, to force it to generate latent vectors that roughly follow a unit gaussian distribution. To generate a data point now means to sample a latent vector from the unit gaussian and input it into the decoder.

Thus VAEs can be considered as a deep learning framework that learns through probabilistic inference, specifically using ideas from the bayesian framework: the input x, hidden representation z and reconstructed output x' are probabilistic random variables, the encoding can be seen as a reconstruction of the z given the x (or, specifically, a probability $q(z|x)$) and the decoding as a reconstruction of the input given z (or a probability $p(x|z)$).

This process can have a simple algorithmic description: an encoding network maps the input into a hidden layer, that is then converted (again with a fully connected layer) into two parameters that define a latent space (a z_μ and a $z_{variance}$), the decoder will then take in input a z value sampled from the distribution defined by these two values, to get to an output through a final neural network layer. The reconstructed image will then be compared to the original to get a first loss value for the reconstruction ability of the VAE. Yet an additional term for the loss is also added: we want to keep the representation of the encoded data as efficient as possible (this means that also the various diverse features of the data will be encoded in different latent spaces). This will mean keeping our encoded vector as close as possible to a gaussian unit shape (this will be measured with KL divergence, as explained later).

Finally, as deep learning tradition dictates, this is all trained by backpropagation. An interesting question arising is what makes possible calculating gradients with respect to aleatory variables: this is often called the reparametrization trick, and will be discussed later in more detail, but a simple explanation is that the model can actually be rewritten deterministically. That is the function of the two layers z_μ and a $z_{variance}$: z depends on these two parameters deterministically, with the randomness injected independently from these two parameters (when they are combined by a gaussian function), making derivation with respect of this two, and thus backpropagation, possible.

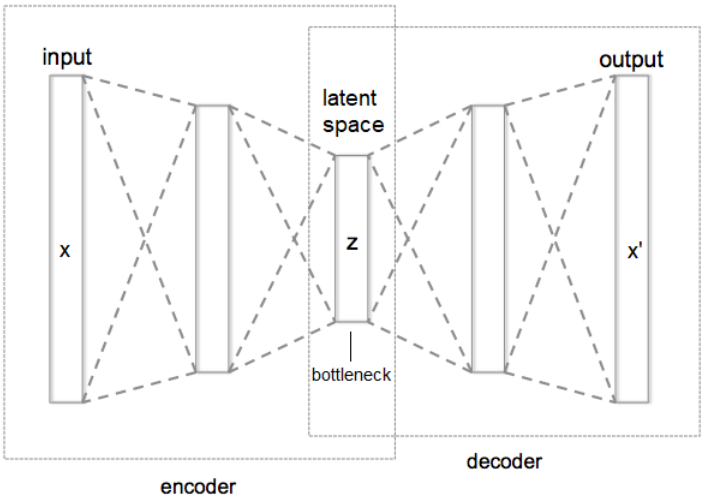

Figure 1: General structure of an autoencoder

2 Methods

2.1 Variational Autoencoders

In this section the components that constitute the autoencoder are analyzed in more details. The code implemented in Python using the chainer framework can be found in: `https://github.com/FiammettaS/Computational-Neuroscience-project/tree/master`

In the following, the input data, that can be for instance images or music, is denoted with x and the corresponding variable in the feature, or latent space, with z.

The intuition is that z could represent different kinds of attributes, for instance shape, position, colour, etc., over which there is a prior distribution $p(z)$. For simplicity, this prior is assumed to be Gaussian. In order to generate data, for each latent factor the values are sampled from this distribution: $z \sim p(z)$ and the values of x are sampled from the conditional distribution $p(x|z)$. A decoder neural network is specified such that it outputs the distribution $p(x|z)$. In order to infer latent variables from the observed data, the distribution $p(z|x)$ is needed. Using Bayes' rule it can be written as:

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}. \tag{1}$$

The denominator $p(x)$ is the evidence of the input data, which is defined taking the expectation over all possible values of z:

$$p(x) = \int p(x|z)p(z)dz \tag{2}$$

In order to estimate the parameters of the distribution $p(x)$, the network is trained by maximizing the likelihood of the training data.

The distribution in (2) is intractable; it is thus impossible to directly optimize the likelihood of the training data: an encoder is needed to approximate the posterior distribution with a given distribution $q(z|x)$ that is as close as possible to $p(z|x)$. This is equivalent to minimizing the dissimilarity of the two distributions, which can be expressed by means of their Kullback-Leibler (KL) divergence. The KL divergence between two distributions $p(x)$ and $q(x')$ is defined as:

$$D_{KL}(p(x)||q(x')) = -\Sigma q(x')log(\frac{p(x)}{q(x')}) \tag{3}$$

Since $p(x)$ does not depend on z and therefore on the expectation with respect to z sampled from $q(z|x)$, taking the log of the likelihood and using Bayes' rule, the equation becomes:

$$\begin{aligned}\log p(x) = E_{z\sim q(z|x)}\big[\log p(x)\big] = E_{z\sim q(z|x)}\left[\log \frac{p(x|z)p(z)}{p(z|x)}\right] = \\ E_{z\sim q(z|x)}\left[\log \frac{p(x|z)p(z)}{p(z|x)}\frac{q(z|x)}{q(z|x)}\right] = \\ E_{z\sim q(z|x)}\left[\log p(x|z)\right] - E_{z\sim q(z|x)}\left[\log \frac{q(z|x}{p(z)}\right] + E_{z\sim q(z|x)}\left[\log \frac{q(z|x}{p(z|x)}\right] = \\ E_{z\sim q(z|x)}\left[\log p(x|z)\right] - D_{KL}(q(z|x)||p(z)) + D_{KL}(q(z|x)||p(z|x)),\end{aligned} \tag{4}$$

where the last derivation is obtained from the definition of KL-divergence in equation (3).

The third term of equation (4) is impossible to compute; however, since it represents a KL divergence, it is always positive.

Therefore, if $\mathcal{L}(x, p, q)$ (also known as ELBO term) denotes the first two terms of the equation:

$$\mathcal{L}(x, p, q) = E_{z\sim q(z|x)}\left[\log p(x|z)\right] - D_{KL}(q(z|x)||p(z)), \tag{5}$$

we have that $\mathcal{L}(x, p, q)$ is a lower bound on the left hand side of equation (4), that is $\mathcal{L} \leq \log p(x)$. As explained before, the first term is given by the decoder and the second term by the encoder.

2.1.1 Encoder

The latent variable z represents the most important features in x that capture meaningful variation in the data. The encoder is a mapping from the input x to the feature z. This function can take many different forms, here artificial neural networks are used.

Different architectures with different numbers of linear, convolutional or LSTM layers were implemented. Here only the best performing networks are presented.

The encoder outputs the two parameters in the latent space of the Gaussian distribution $q(z|x)$: the mean z_μ and the standard deviation $z_{variance}$.

2.1.2 Decoder

The features in the latent space can be used to reconstruct the original data with a second network: the decoder, which is also modelled by an artificial neural network.

Some information from the original data is loss in the reconstruction process, and we measure it with the loss function.

2.1.3 Loss function

The loss is defined as a measure of the difference between the output and input layer, therefore the network is trained in order to make the reconstruction of the data as close as possible to the original data.

In order to maximize the log likelihood of the training data, we can in turn maximize the variational lower bound $\mathcal{L}(x, p, q)$, which is tractable.

This is equivalent to minimizing the loss function defined as:

$$l(x, p, q) = -\mathcal{L}(x, p, q) = -E_{z\sim q(z|x)}\left[\log p(x|z)\right] + D_{KL}\big(q(z|x)||p(z)\big), \tag{6}$$

in which the first term measures the reconstruction loss - the difference between the original and the reconstructed data - and the second term is the KL divergence, that acts as a regularizer and is

needed to keep the representation of each input belonging to the same class as similar as possible and maximize the divergence between different classes.

The variational lower bound $\mathcal{L}$ can be decomposed with respect to each datapoint x_i:

$$\mathcal{L}_{x_i}(x, p, q) = E_{z\sim q(z|x_i)}\Big[\log p(x_i|z)\Big] - D_{KL}\big(q(z|x_i)||p(z)\big) \tag{7}$$

and stochastic gradient descent can be applied to the parameters of the distribution q.

Thus the encoder and decoder are probabilistic and both output a mean μ and a diagonal covariance Σ representing the distributions $q(z|x)$ and $p(x|z)$ respectively and then we sample from the distributions to get x and z.

2.1.4 Reparametrization

An additional aspect characterizing the VAE algorithm implementation is the use of a backpropagation algorithm for the weights update: but how it is possible, given the fact that it would imply taking a derivative with respect to the parameters of a stochastic variable? (We want to take derivatives of a function of z, drawn from a distribution $q(z|x)$, with respect to the parameters of q) Yet we can make the stochasticity of our model independent from the parameters, because it is possible, for our particular gaussian distribusion, to reparametrize it so that the parameters of the samples we take depend deterministically on the distribution. That is, a normally-distributed variable with mean μ and standard deviation σ, can be sampled as follows

$$z = \mu + \sigma \odot \epsilon, \tag{8}$$

where $\epsilon \sim Normal(0, 1)$. In this way we have a function that depends on the parameters in a deterministic way. So we can take derivatives of the functions $f(z)$ with respect to the parameters μ and σ.

This is why in the variational autoencoder we need a mean and a variance as outputs by an inference network. The distribution q has now some parameters that we can optimize: we can backpropagate with the objective function (the ELBO), a function of samples from the latent variables z.

2.2 Datasets

2.2.1 MNIST dataset

For the first part of the project, the variational autoencoder was trained on the MNIST dataset, which contains 28x28 greyscale images of handwritten digits.

2.2.2 Midi data

In the second part of the project, the VAE was trained with midi music files, that can be found at: `http://www.jsbach.net/midi/midi_solo_lute.html`. To handle midi files, they were converted into arrays to feed to the network

Additionally, we tried to train on midi versions of songs from different genres (Blues, Jazz, Disco, Rock), taken from: `https://freemidi.org/`.

3 Results

We can visualize the learning of a VAE not just by looking at the change of the loss in time, but also by taking samples from the prior or the posterior. A more detailed version of this processes can be seen in the jupyter notebook. As discussed, a VAE model creates, for the data it takes in input, a posterior distribution for its latent space (Fig. 3). This can be visualized by simply feeding many input images to the encoder to get their sample for the latent variable z. In this way we can visualize what feature of the data our latent dimension variables encode. Furthermore, this can be plotted during the training, to see how the posterior adapts from being close to the prior (a round gaussian distribution) to differentiate specific classes of data in the latent space.

We can also look at the prior of the distribution of the decoder data (Fig. 2). This is simply done by sampling latent variables and inputting them to the decoder: in this way we can obtain the data points reconstructed from the encoding, notice the degree of reconstruction (that will inevitably be blurrier for images due to the previous dimensionality reduction), and notice what features the decoder associates with the samples of the latent space.

Other interesting observations can be done also regarding the evolution of the loss term. In fact, its two components, the data likelihood under the posterior (the reconstruction term) and the KL divergence of the posterior from the prior, indicate two different aspects of the learning: the reconstruction term makes the output close to the input the data, while the KL term forces the encoded variable to be close to the gaussian prior. This means that at first the VAE learns to output avarage data points distributed like the prior, because this is the most efficient way to minimize each loss term. But then a coordination arises in the loss term: the KL divergence increases a little to permit more information to be held into the posterior, while the reconstruction loss term now drops drastically because it can use the more efficient encoding (Fig. 4). Another way of seeing this is that a training that would just make the KL term drop to zero would mean setting the probability distribution of $q(z|x)$ equal to the prior $p(z)$, making the bayesian infererence reconstruction much less powerful. Unfortunately this sometimes happens, and is the reason for the use of "weak" decoding methods that force the model to use the latent encoding. This is also the reason why the KL cost can be multiplied by a variable weight that is set to zero at the start of the training, so that the model learns as much encoding as possible in z, and then is gradually increased to one during the training to force the model to pack the encoding into an efficient prior (this is indeed like annealing from a general autoencoder architecture to a VAE)[6].

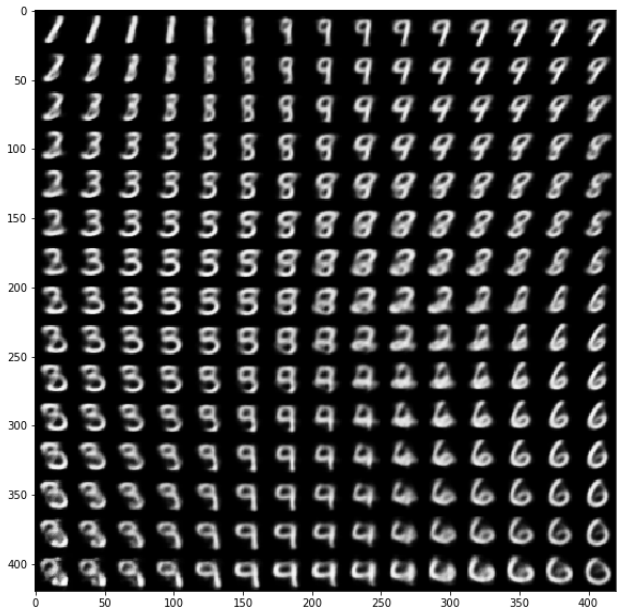

Figure 2: MNIST - Images generated by the decoder from randomly sampled latent variables

As we can see, at this epoch the clusters have become more separated while at the beginning they were distributed as a unit Gaussian, meaning that the network is learning to differentiate between the different classes and properly encode the images:

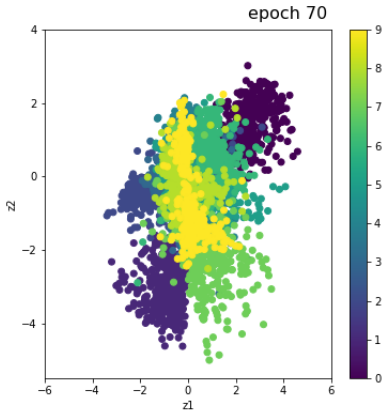

Figure 3: MNIST - Latent variables generated by the encoder representing the test images, with a different colour for each digit, at the end of training.

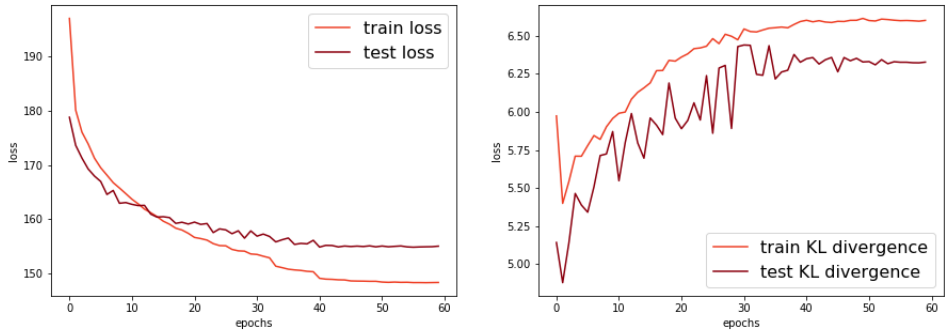

Figure 4: MNIST - Left: train and test total loss over epochs. Right: train and test KL divergence over epochs

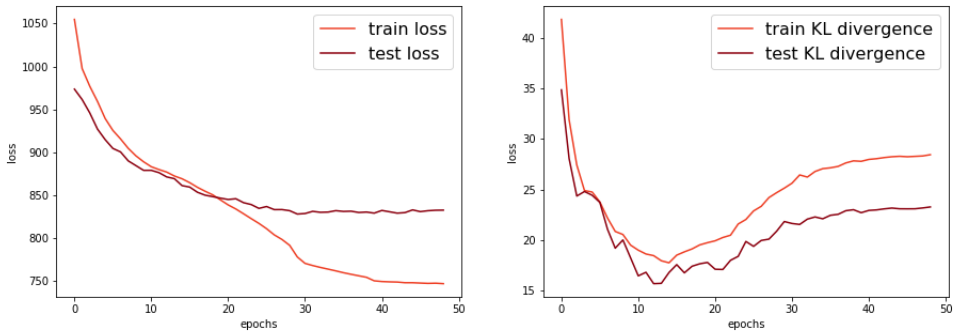

Figure 5: MIDI - Left: train and test total loss over epochs. Right: train and test KL divergence over epochs

The previous results were generated by learning on the MNIST dataset. We then tested the generative capabilities of the VAE in music generation by learning on midi files. As we can see (Fig. 5), learning is possible also with music midi files, with KL increase signifying successful encoding. A sample of the music results can be found in the aforementioned jupyter notebook.

4 Discussion

We have shown that variational autoencoders have good feature extraction and generative properties. Furthermore, they have some important advantages when compared to other unsupervised methods.

For example, when compared to generative adversarial networks, another popular approach for unsupervised learning, a VAE has the edge of being able to generate data with specific features, without having to search over the entire distribution. Furthermore, GANs are trained by discrimination between "real" and "fake" images, without any constraint on the actual class of data fed into the network. This could produce results where the data generated just picks up a collateral feature of the data, without the wanted feature generated (e.g. producing a picture with the style that looks like the input data, but without the wanted subject). Instead, VAEs compare their generated data instances with the originals, something which is impossible with GANs, often preventing this problem. One advantage that GANs seem to have is the capacity to generate more definite images by using their adversarial training, while VAEs produce more blurry images. Therefore, there have been some efforts to combine variational autoencoders and generative adversarial networks. For example by using the encoding and decoding network of a VAE, but utilizing adversarial training for the decoder[7].

Another famous algorithm, principal component analysis (PCA), can be compared to VAEs, since it is also based on dimensionality reduction, and used for unsupervised learning. The advantage that VAEs have over PCA is that, thanks to the use of a neural network, the dimensionality mapping is not restricted to a linear map, and can use nonlinear encoding and decoding of data (an encoder with a single layer will span the same subspace of a PCA, but the possibility of using many more layers, types of layers, and activation functions makes autoencoders capable of stronger dimensionality reductions).

Another property of autoencoders that is seldom mentioned is that their architecture is based on a decoding component that resembles recent theories of Bayesian brain architecture: the decoder "predicts" an output based on an encoding of the input. This can be compared to neural codes of sensory stimuli that are used to compute predictions that are to be compared to the environment to minimize the sensory error. In this predictive coding paradigm the brain is modeled as a Bayesian inference device, constantly producing hypothesis predicting the next sensory input[8]. In this framework the brain solves the problem of computing predictions of lower-level sensory inputs with backward connections from higher hierarchy cortical areas. This is possible thanks to the statistical regularities of the environment that the brain encodes to construct top-down generative models that predict and suppress lower level sensory inputs. This is often described in Bayesian terms, with the prediction measure being labeled as 'priors' and the sensory input being the 'likelihood'. Then the difference measure between the two (the 'prediction error', or 'surprise' in information entropy terms), is used to update the network to better predicts future sensory stimuli. Although the neural mechanism of such predictive processes are not known exactly, there is an ample literature regarding the use of Bayesian models in explanations of neurophisiological data. For example, Kersten et al. (2014)[9] did a meta-analysis of a number of fMRI works that show that activity in V1 actually decreases when visual information is given as whole well-known objects, while activity in higher areas increases. This could be explained by high-level areas taking the actual role of interpreting the data, and then suppressing low level areas to save on the cost of spiking cells metabolism. The idea, first proposed by Mumford (1992)[10], is that the higher level areas compute some hypotheses that try to feedback a prediction to sensory areas. These hypotheses would then be scored based on a signal difference. This would predict a small activity in bottom up areas for good fits of activation. A similar model named "predictive-coding" would later be introduced[11]: the top-down signal, based on Kalman filtering reconstructed probability distributions, would feedback to lower areas, that would then bring a bottom up error signal. This would be implemented in a chain of connected cortical areas, each with their own feedback and feedforward mechanism.

Another model[12] proposes a Bayesian framework of hierarchical top-down feedbacks, based on both neurophysiological data and computational models,inspired by particle filtering. Furthermore, a series of recent studies[13] showed some feedback activation in early visual neurons during the perception of perceptual illusion stimuli. That is, higher cortical areas predict the presence of an illusory contour, and feedback this information to V1 layer neurons. This result is quite interesting, as it is quite similar to a simple experiment with VAEs: if they are trained with, say, digits, and then they are fed an incomplete digit (like a zero missing part of the picture), they will still predict in output a whole number, and display a sort of completion perceptual illusion.

In conclusion, the autoencoder architecture and its Bayesian inference framework can be considered an original and effective line of research in machine learning. Additionally, some of its components can be considered both cognitively and neurophysiologically plausible, as they implement the idea of cognition as prediction from a higher "encoded" area, with learning based on feedback from previously shown input stimuli.

References

[1] Ian Goodfellow et al. "Generative Adversarial Networks". In: *Advances in neural information processing systems* (2014), 2672—2680. DOI: arXiv:1406.2661.

[2] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *Proceedings of the 2nd International Conference on Learning Representations* (2013). DOI: arxiv.org/abs/1312.6114.

[3] Yoshua Bengio. "Learning deep architectures for AI". In: *Foundations andTrends in Machine Learning* (2009), pp. 1–127. DOI: 10.1561/2200000006.

[4] Yann Le Cun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* (2015). DOI: 10.1038/nature14539.

[5] Jost Tobias Springenberg. "Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks". In: *Cornell University Journal* (2016). DOI: arXiv:1511.06390.

[6] Samuel R. Bowman et al. "Generating Sentences from a Continuous Space". In: *Cornell University Journal* (2016). DOI: arXiv:1511.06349.

[7] Anders Boesen Lindbo Larsen et al. "Autoencoding beyond pixels using a learned similarity metric". In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48* (2015), pp. 1558–1566. DOI: arXiv:1512.09300.

[8] Andy Clark. "Whatever next? Predictive brains, situated agents, and the future of cognitive science." In: *Behavioral and Brain Sciences* (2013). DOI: 10.1017/S0140525X12000477.

[9] Daniel Kersten, Pascal Mamassian, and Alan Yuille. "Object perception as Bayesian inference". In: *Annual Review of Psychology* (2014). DOI: 10.1146/annurev.psych.55.090902.142005.

[10] Peter Honey and Alan Mumford. *The manual of learning styles*. 1986. ISBN: 9780950844404.

[11] Dana H. Ballard et al. "Deictic codes for the embodiment of cognition". In: *Journal of the Optical Society of America A* (1997), 723–767.

[12] Tai Sing Lee and David Mumford. "Hierarchical Bayesian inference in the visual cortex". In: *Journal of the Optical Society of America A* (2003), pp. 1434–1448. DOI: 10.1364/JOSAA.20.001434.

[13] Lauren J. Bains et al. "Selective Activation of the Deep Layers of the Human Primary Visual Cortex by Top-Down Feedback". In: *Current biology* (2016), 371–376. DOI: 10.1016/j.cub.2015.12.038.