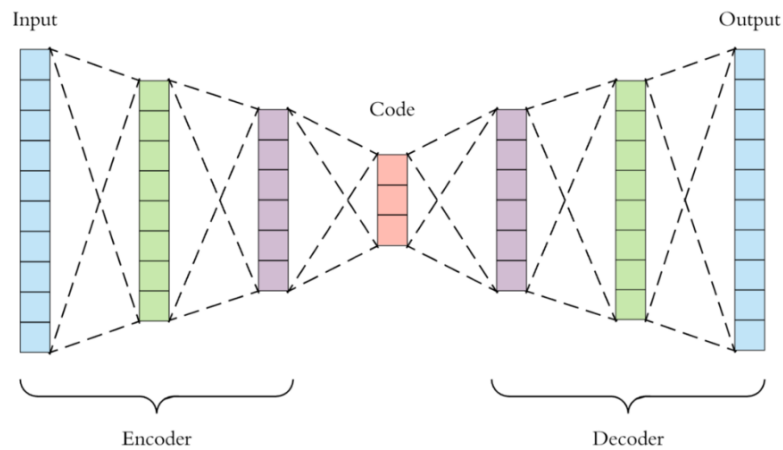
Autoencoder Implementation Issues

Autoencoder is a type of Artificial Neural Network (ANN) used to learn efficient data codings in an unsupervised manner. The aim of the autoencoder is to learn a representation (encoding) for a set of data, typically for the purpose of dimensionality reduction. For instance, consider the famous MNIST dataset, each MNIST image is a scan of a handwritten digit in a 28x28 image, so our "inputs" are in 28x28 = 784 dimensions. We can represent each MNIST digit as a vector in 25 dimensions, and this is where we can see the utility of an autoencoder, it is a feature extraction algorithm it helps us find a representation for our data and we can feed that representation to other algorithms, for example a classifier. Autoencoders can be stacked and trained in a progressive way, we train an autoencoder and then we take the middle layer generated by the autoencoder and use it as input for another autoencoder and so on. This is the first step towards deep learning, the stacked autoencoders will learn how to represent data, the first level will have a basic representation, the second level will combine that representation to create a higher-level representation and so on. Think about images, a first level autoencoder will learn to detect borders as features, the second level will combine those borders to learn traces and patterns and so on.

The following are the implementation issues associated with autoencoders:

- An autoencoder may not always work. It may always give you the average of the input set, always reconstruct the input set exactly, or combine those two defects in sneaky ways.
- Training an autoencoder is tedious. A lot of data, processing time, hyperparameter tuning, and model validation before you

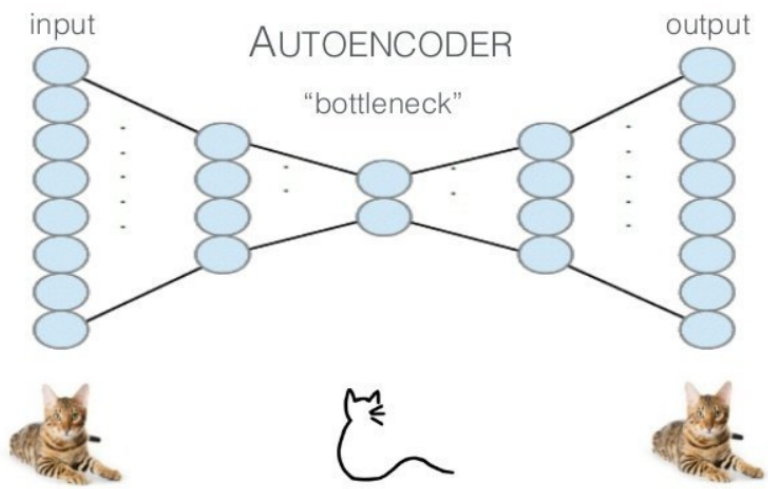
start to build the real model.



The figure above depicts the architecture of an autoencoder. The input first passes through the encoder, which is a fully-connected ANN to produce code, then the decoder, with a similar ANN structure, produces output using the code

- An autoencoder learns to efficiently represent a manifold on which the training data lies. If your training data is not representative of your testing data, then you wind up obscuring information rather than clarifying it.
- An autoencoder learns to capture as much information as possible rather than as much relevant information as possible i.e. if the information most relevant to your problem makes up only a small (in magnitude) part of the input, the autoencoder may lose a lot of it.
- Using an autoencoder also destroys any interpretability your model may otherwise have had.
- Autoencoders are not that efficient compared to Generative Adversarial Networks in reconstructing an image. As the complexity of the images increase, autoencoders struggle to keep up and images start to get blurry.

- Separate Classifier Autoencoder (SCA) has the drawback that the learned representation is prone to redundancy since the autoencoder will encode some semantic information (Portenier, Hu, Favaro, Zwicker).



The figure above depicts an example of an autoencoder: left side of the network is an autoencoder and is responsible for reduction and the right side of the network is an autoencoder that is in charge of enlargement

- Autoencoder-based CF (Collaborative Filtering) faces the problem of training on a large sparse target, the loss function of Autoencoder based CF is evaluated on just known values (Sun, Ballesteros, Pamucar).
- Variational Autoencoders (VAEs) require strong assumptions about the structure in the data. They might make severe approximations, leading to suboptimal models. They might rely on computationally expensive inference procedures like Markov Chain Monte Carlo (MCMC) (Meyer, 2016).
- Autoencoders face problems with respect to generalization when datapoints are missing from the training set. This causes a larger

issue for higher-level applications whose data have higher intrinsic dimensionality and hence are more likely to include such 'holes'.

- Determining the weights for each pair-wise distance is very challenging and this practice is risky in converting GAE (Generalized Autoencoder) into a supervised model when some relationships are over emphasized, and others are neglected. Meanwhile focusing on minimizing the loss of data relationships but ignoring the loss of data itself is very likely to lose information (Meng, Ctchpoole, Skilicorn, Kennedy).

Portenier, T. Hu, Q. Favaro, P. Zwicker, M. Fine-Grained Retrieval with Autoencoders. Retrieved from: http://www.scitepress.org

Sun, S. Ballesteros, T. Pamucar, D. Fuzzy Systems and Data Miningll: Proceedings of FSDM. 2016. Retrieved from: https://books.google.co.in

Meyer, D. Notes on Variational Autoencoders. 2016. Retrieved from: http://www.1-4-5.net

Meng, Q. Catchpoole, D. Skilicorn, D. Kennedy, P. Relational Autoencoder for Feature Extraction. Retrieved from: https://arxiv.org