ML Praktikum WS 17/18
Introduction to Unsupervised Neural Networks

Martin Mundt

FIAS / Goethe Uni Frankfurt

February 2018

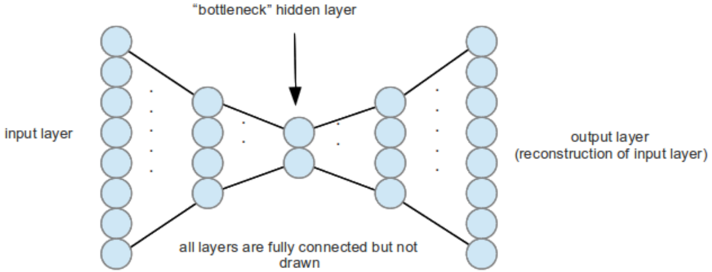
1. General Concepts & Types
2. Autoencoders
3. Research & Open Challenges
4. Our next practical session

General concepts

- Make use of large quantities of **unlabeled data**
- Learn the **structure of the data**. Could be used for e.g. in clustering (k-means, mixture models) for information retrieval, data compression, statistical data analysis etc.
- In NNs for **"Representation Learning"**: a meaningful & complete set of features describing the data.
- In statistics to e.g. learn parameters of **"Latent Variable Models"** (set of unobserved variables)

Autoencoders

- Learn a meaningful representation ("encoding") of the complete data
- An **encoder** maps to a **"code"** or **"latent variables"/"latent representations"** and is then fed into a **decoder** to **reconstruct** the input.
- In the simplest version a decoder is a "flipped" version of the encoder (with shared weights).



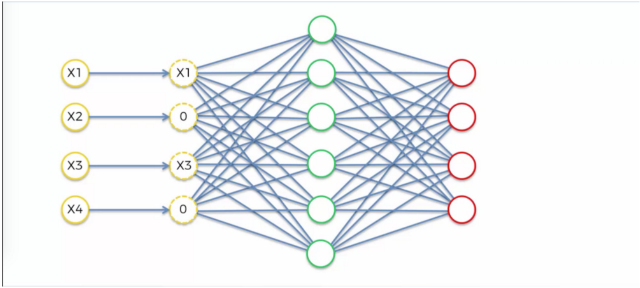
http://nghiaho.com/wp-content/uploads/2012/12/autoencoder_network1.png

Autoencoders

- Can come in all sorts of flavors (fully-connected layers, convolutions etc.)
- Can be trained and stacked (greedy-layer-wise training)
- Can be extended to deep networks by adding more hidden layers in both encoder and decoder
- For large quantities of unlabeled data, can be used as pre-training for semi-supervised learning

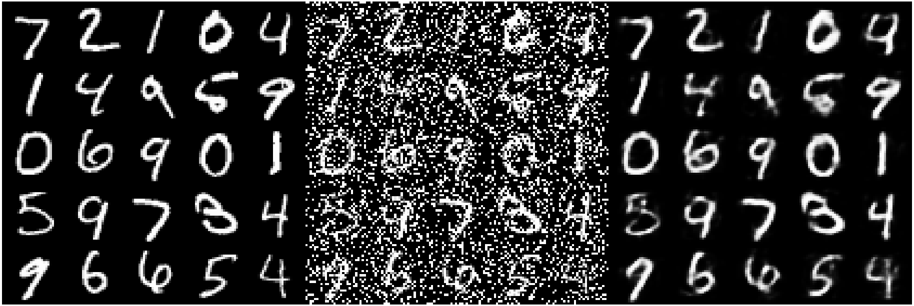
Denoising Autoencoders

- If the architecture doesn't have a bottleneck (i.e. more hidden nodes than input nodes) the AE can just learn an identity
- We can randomly corrupt the data on purpose to solve this issue, that is add a noise process
- Keep in mind that we still compute the reconstruction error with the original data!



https://towardsdatascience.com/denoising-autoencoders-explained-dbb82467fc2

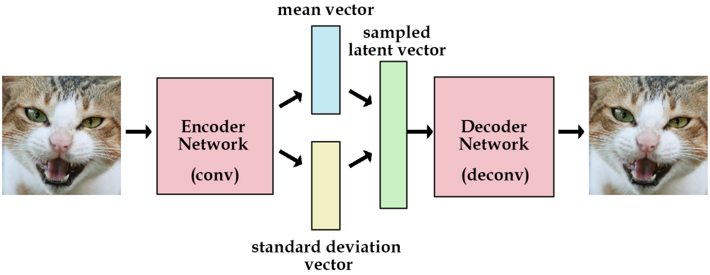
Denoising Autoencoders



https://www.doc.ic.ac.uk/ js4416/163/website/img/autoencoders/denoising-example.png

Variational Autoencoders

- Our AE has a latent vector/variables in the hidden layer connecting encoder to decoder. But difficult to grasp as it is unconstrained.
- We can add a constraint such as forcing the latent vector to follow a unit Gaussian (e.g. by optimizing KL divergence in addition to reconstruction, which measures how close we are to a unit Gaussian).



http://kvfrans.com/content/images/2016/08/vae.jpg

Variational Autoencoders

- We can then sample images by specifying mean and variance and feeding it into the decoder network portion.

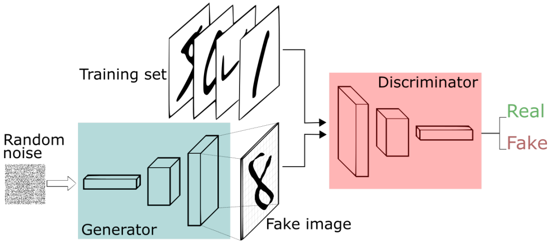
http://kvfrans.com/variational-autoencoders-explained/

Other variants of Autoencoders

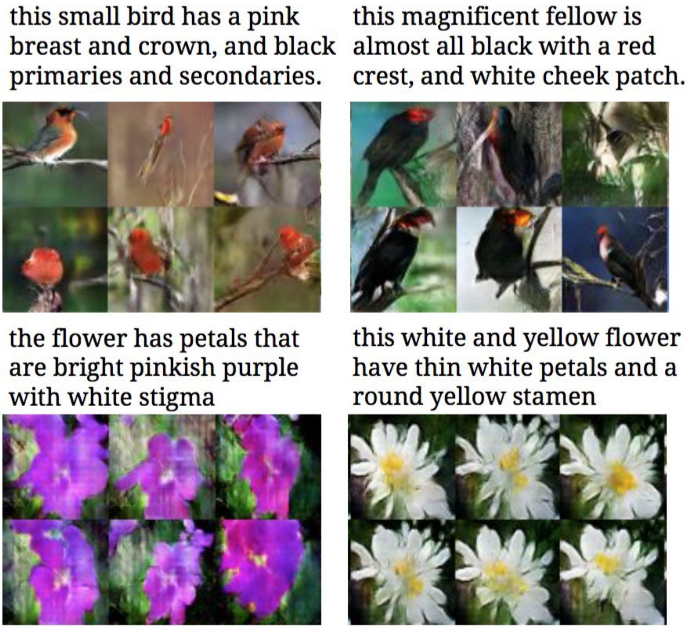
- **Sparse Autoencoders**: similar to Denoising AE; useful for large number of hiddens by imposing sparsity
- **Other generative Autoencoders**
- **Contractive Autoencoders**: also similar to Denoising AE: Adds explicit regularizer to objective function to learn robustness to variations

Generation using CNN based Generative Adversarial Networks (GAN)

- Various methods to use deconvolutional NNs to generate data
- GANs are 1 option: filters of the DeconvNet are trained to generate output that resembles some training data to fool a different "discriminator" (C)NN.



https://deeplearning4j.org/generative-adversarial-network



https://medium.com/intuitionmachine/navigating-the-unsupervised-learning-landscape-951bd5842df9

- Unsupervised pre-training (semi-supervised learning) doesn't hold-up to the promise when enough labeled data is available. Supervised learning seems to have more meaningful features with respect to a task.
- It is difficult to distinguish "background concepts" from "content" in unsupervised learning.
- In images, learning is conducted with metrics on a pixel-level (and not e.g. concepts or entities).
- In clustering the objective is not taken into account in the distance metric. Number of clusters typically not known a priori.
- Stability of very deep NN optimization, especially for GANs.
- Validation of generative models (such as GANs) or clustering algorithms can be difficult.

1. Take your Fashion-MNIST notebook and add both an Autoencoder and a convolutional Autoencoder. You can re-use the pieces of your MLP and CNN for this, but you no longer need labels.
2. While training the AEs, visualize the reconstructed images
3. (Optional): Visualize the weights/filters of the first layer of the encoder. Hint: In Pytorch you can access your model's layers with "model.modules()" and a layer's weights with "module.weight.data"
4. Re-use the learned feature base of your convolutional Autoencoder & train a classifier (MLP with e.g. 1 hidden-layer) on top using a subset of the train data (with labels).
5. How well does your semi-supervised model (task 4) do with limited amounts of training data (200-500 examples). Could you train your supervised CNN with this amount of data?