Assignment 1: Why Generative Models

Discussion: April 27th

In this assignment, we will be looking at why we might need generative models by considering how other models fail at modeling data distributions properly. Use this as an opportunity to set up training/experimentation workflows that you can reuse in later assignments.

General Assignment Notes

You may work on the assignments in groups (up to three people). They will generally consist of writing code, usually to implement some kind of generative model and to apply it/play around with it a little bit. If something is unclear, **ask**. Tasks are often deliberately open-ended in order to encourage exploration. Don't concern yourself with achieving X% of points. These assignments are for you to gain practical experience and to get into the habit of asking (and answering) interesting questions.

Unfortunately, we currently cannot supply GPU servers for the class as we did last year. You will have to work on your own machine or with Google Colab. If you need a refresher, see the general notes of last semester's IDL class **here (https://ovgu-ailab.github.io/idl2019/ass1.html)**. Note that TF 2.x is now the default in Colab, so you don't have to specify it anymore. Also, you might want to install the `tensorflow-probability` library – it might come in handy.

For each assignment, ideally your final results are in a Jupyter notebook so that we can have a look at them in class. Use images/plots liberally. Also, feel free to use markdown cells to write text to explain or discuss your code/results. You can send your notebook via email to Jens and/or bring them to class on your laptop.

Revisiting Autoencoders

Since autoencoders consist of an encoder and a decoder, they could in principle be used to generate data through use of the decoder on its own. Try this:

- Build and train an autoencoder on a dataset of your choice. You may start with a simple fully-connected network and MNIST (…).
- Confirm that your autoencoder has appropriate "generative capacity" by checking that reconstructions look reasonable (ideally on the test set).
- Use the decoder to construct data from randomly generated codes. Note that you will first need a way to generate "reasonable" codes. One example could be uniform random numbers within the bounds set by the encodings of the training set. This is not necessarily a good method, just an example!
- Plot some of the results.

Are you happy with the outputs? You most likely aren't. Offer some hypotheses as to why this might be. Think about and propose (and, if you want to, implement) ways to produce better codes/images.

Ideas for Further Exploration

- Use a different dataset. You should be able to easily switch to other small image datasets such as CIFAR. The results will likely be more “dramatic”(ally bad).
- Try to use CNNs for the encoder/decoder instead.
- Look for other ways to explore the code space and its relation to the image space. For example: Encode two images A and B. Interpolate between the two codes, decoding some number of codes (maybe 10 or so) you encounter “along the way”. This can give you an impression of how the code transitions between different kinds of data (e.g. different MNIST digits). Arguably the simplest interpolation method is linear interpolation. Can you think of “better” ones?