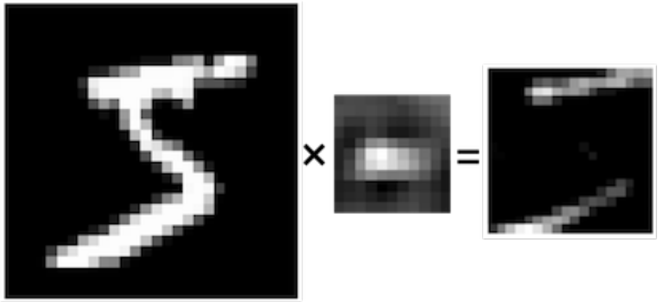
Universität Potsdam

Introduction to Deep Learning

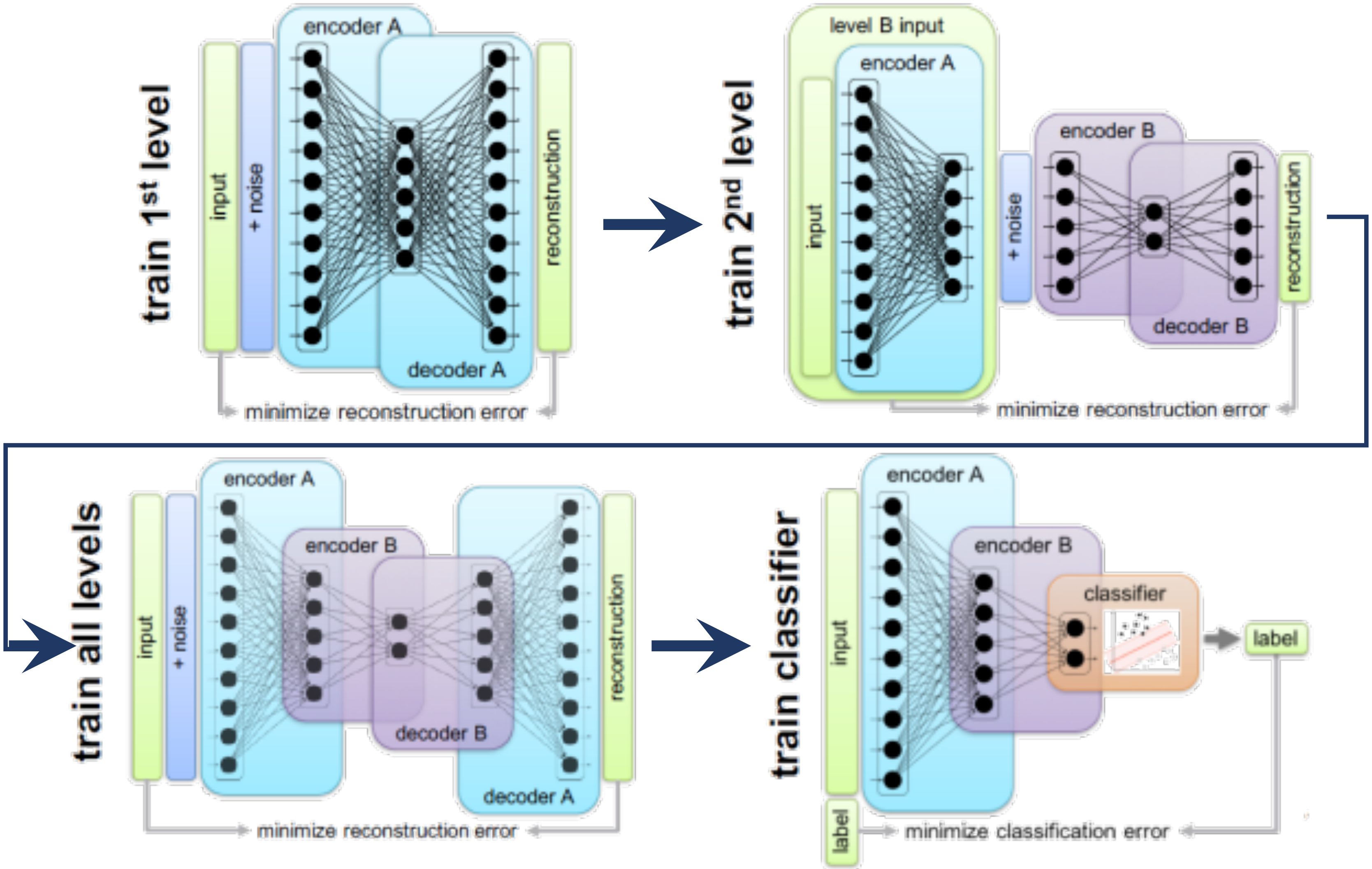
Autoencoders



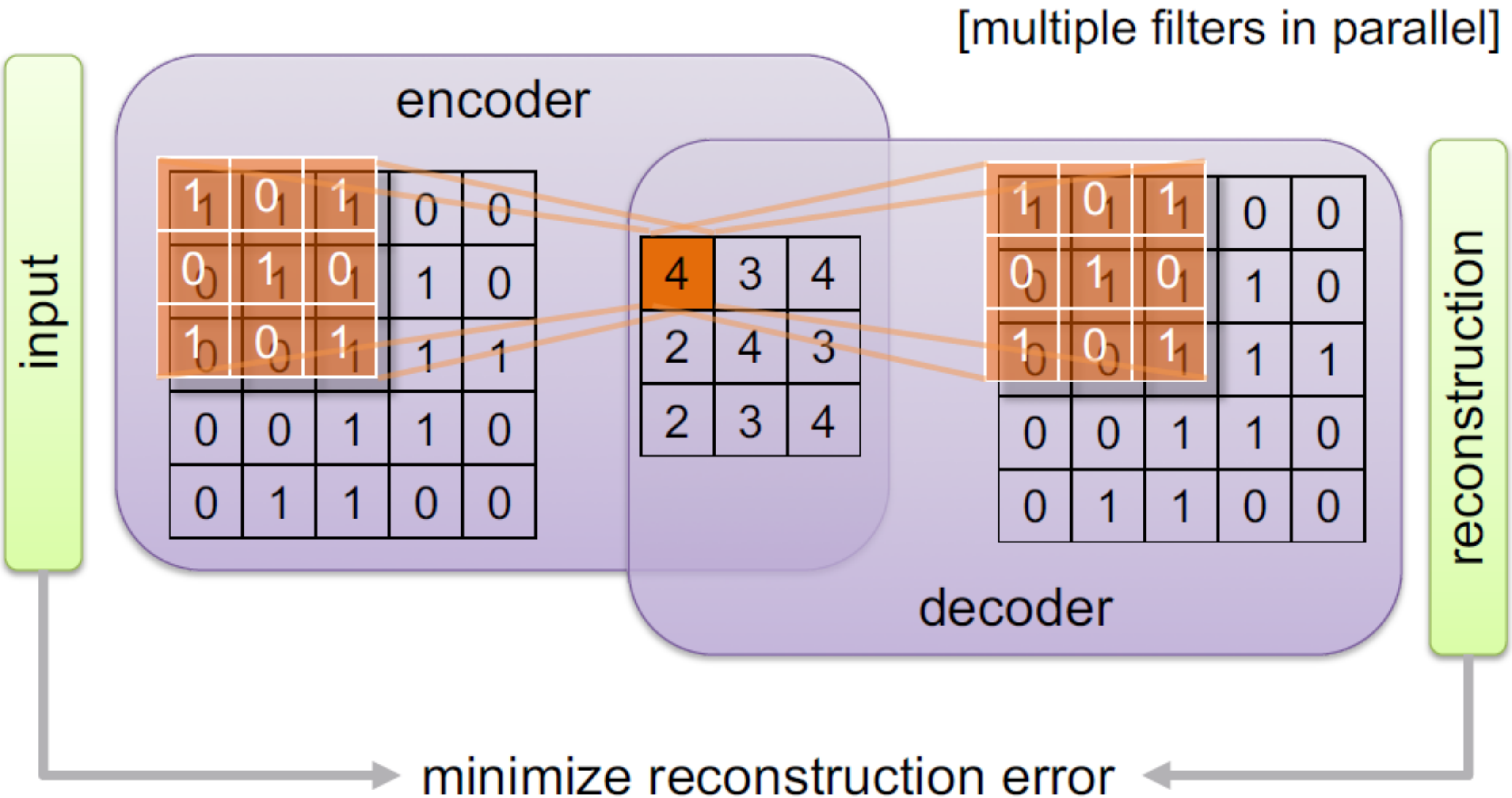
Andreas Krug, M.Sc.
ankrug@uni-potsdam.de

14. January 2019

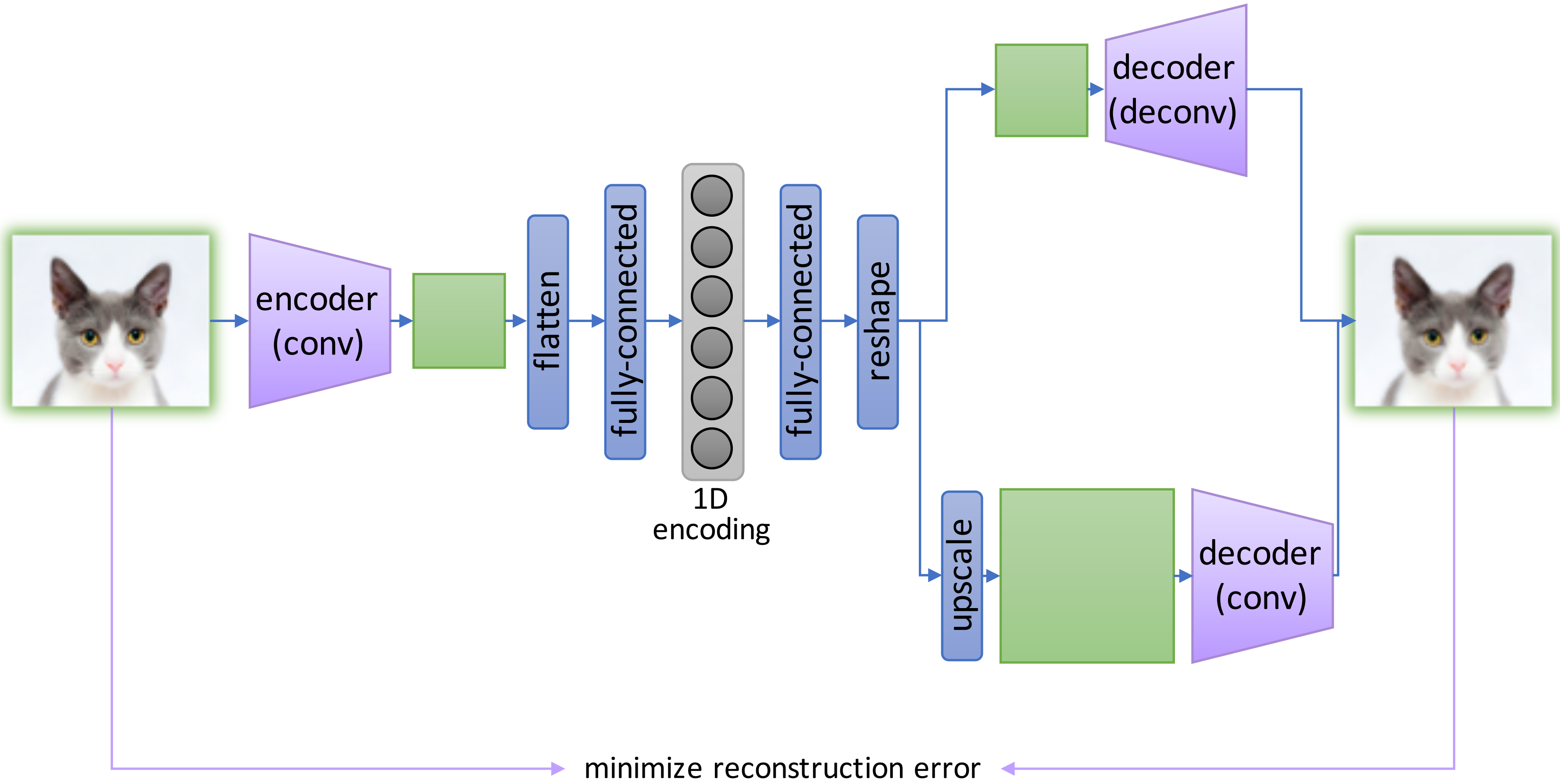
Stacked (denoising) Autoencoder



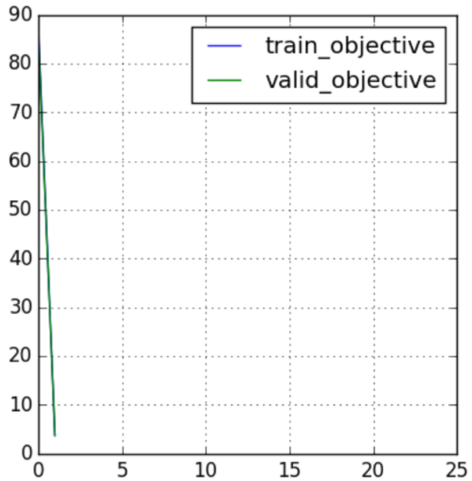
convolutional autoencoder (single-level, no pooling)



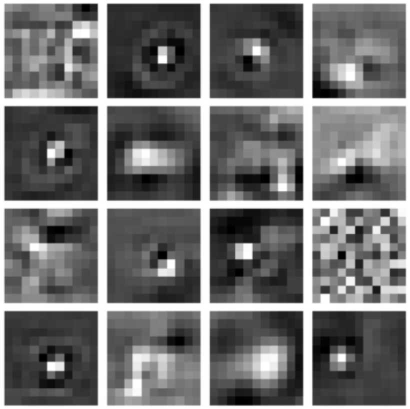
convolutional autoencoder (with 1D encodings)



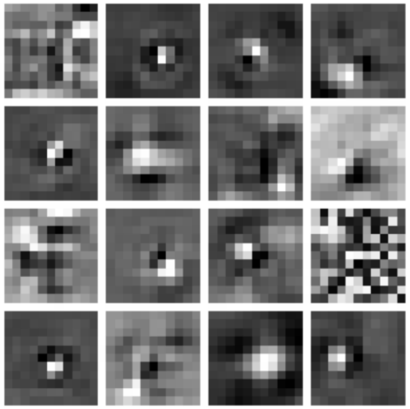
unsupervised pre-training (convolutional autoencoder)

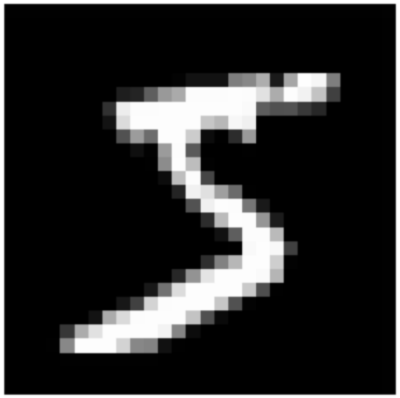


encoder filters

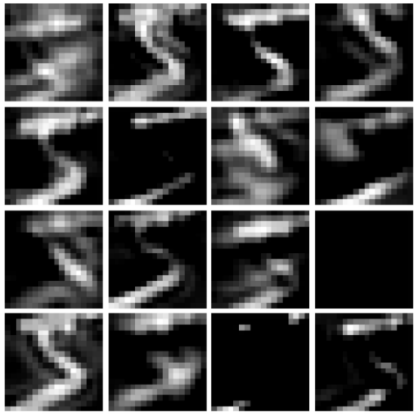


decoder filters





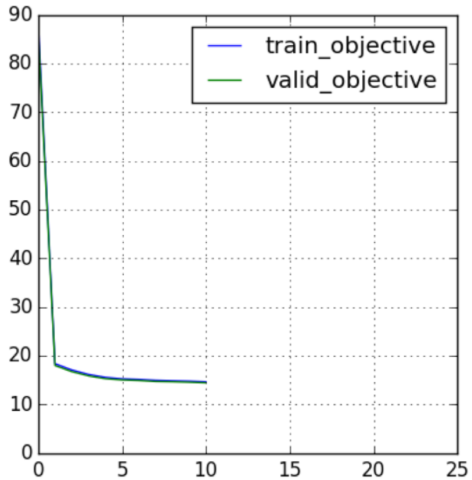
example input



hidden activations

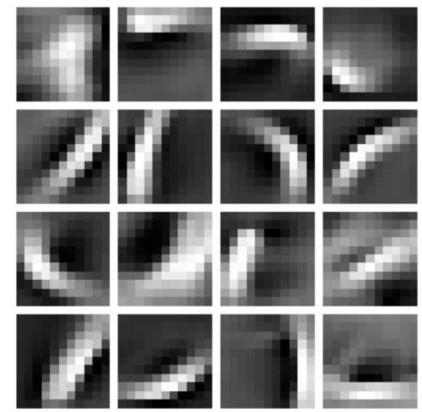
output activations

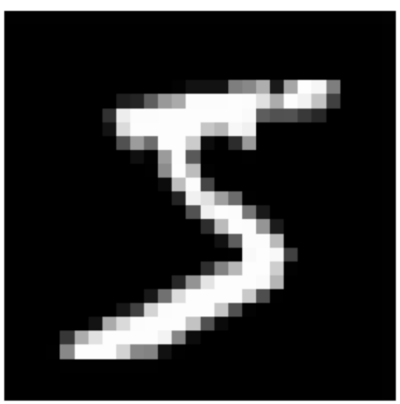
unsupervised pre-training (winner-take-all autoencoder)



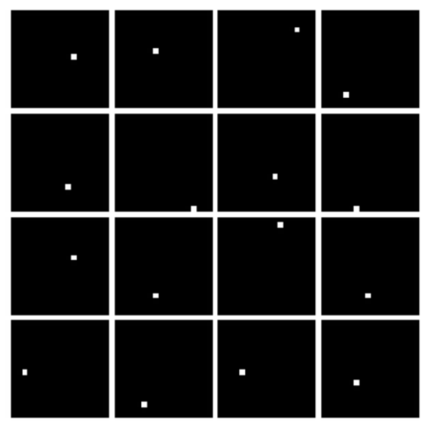
encoder filters

decoder filters

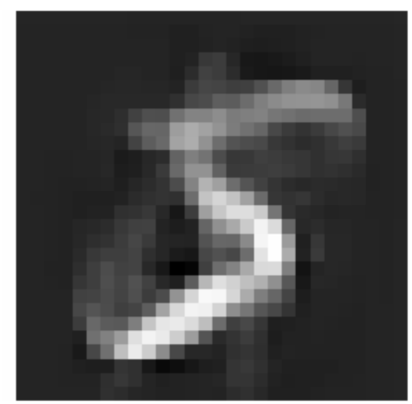




example input

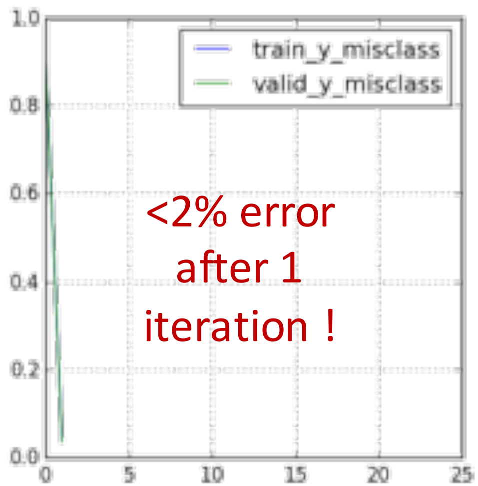


hidden activations

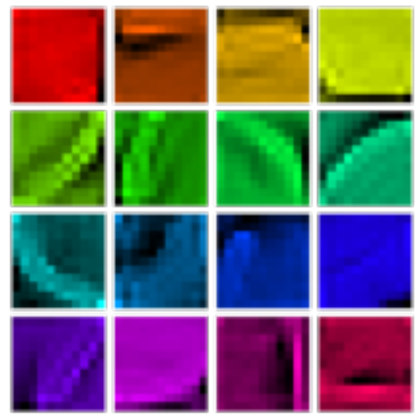


output activations

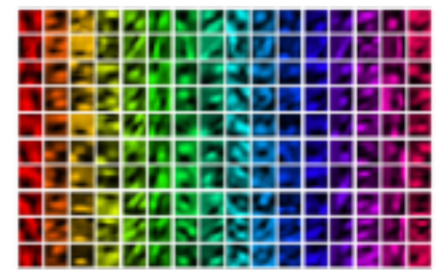
supervised training



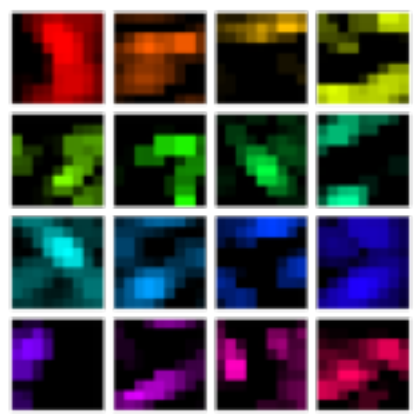
encoder filters



output weights



example input



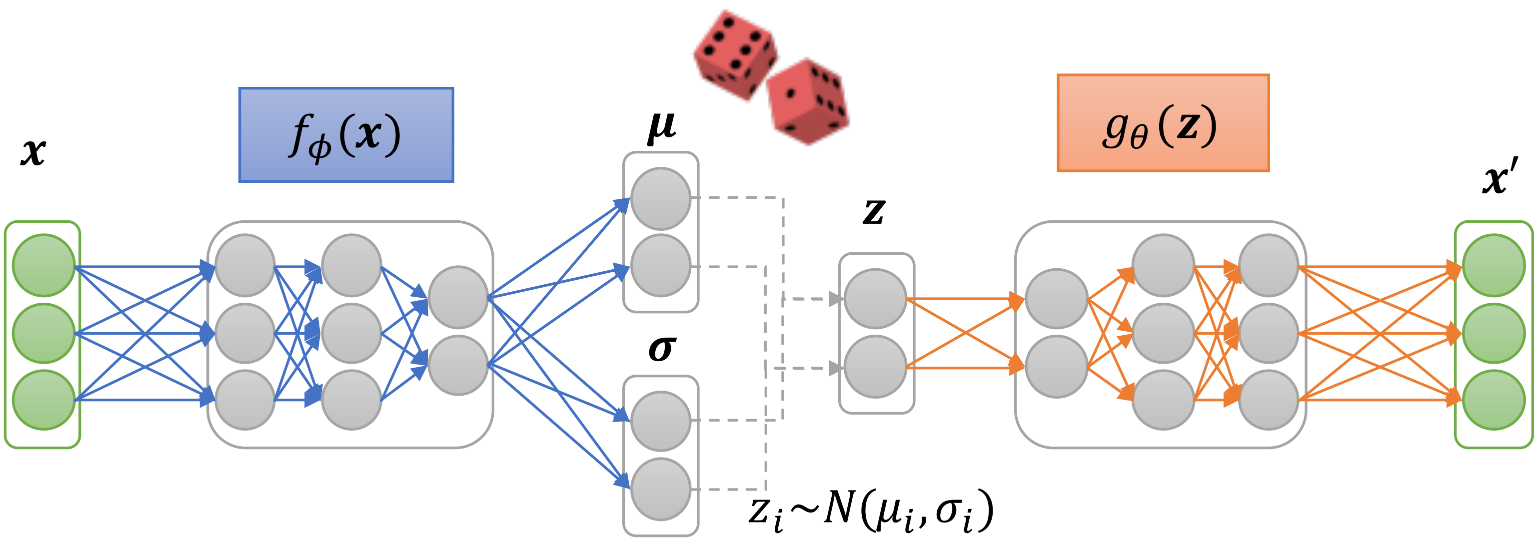
hidden activations

output activations

Outlook
Variational Autoencoder

(part of “Learning Generative Models)

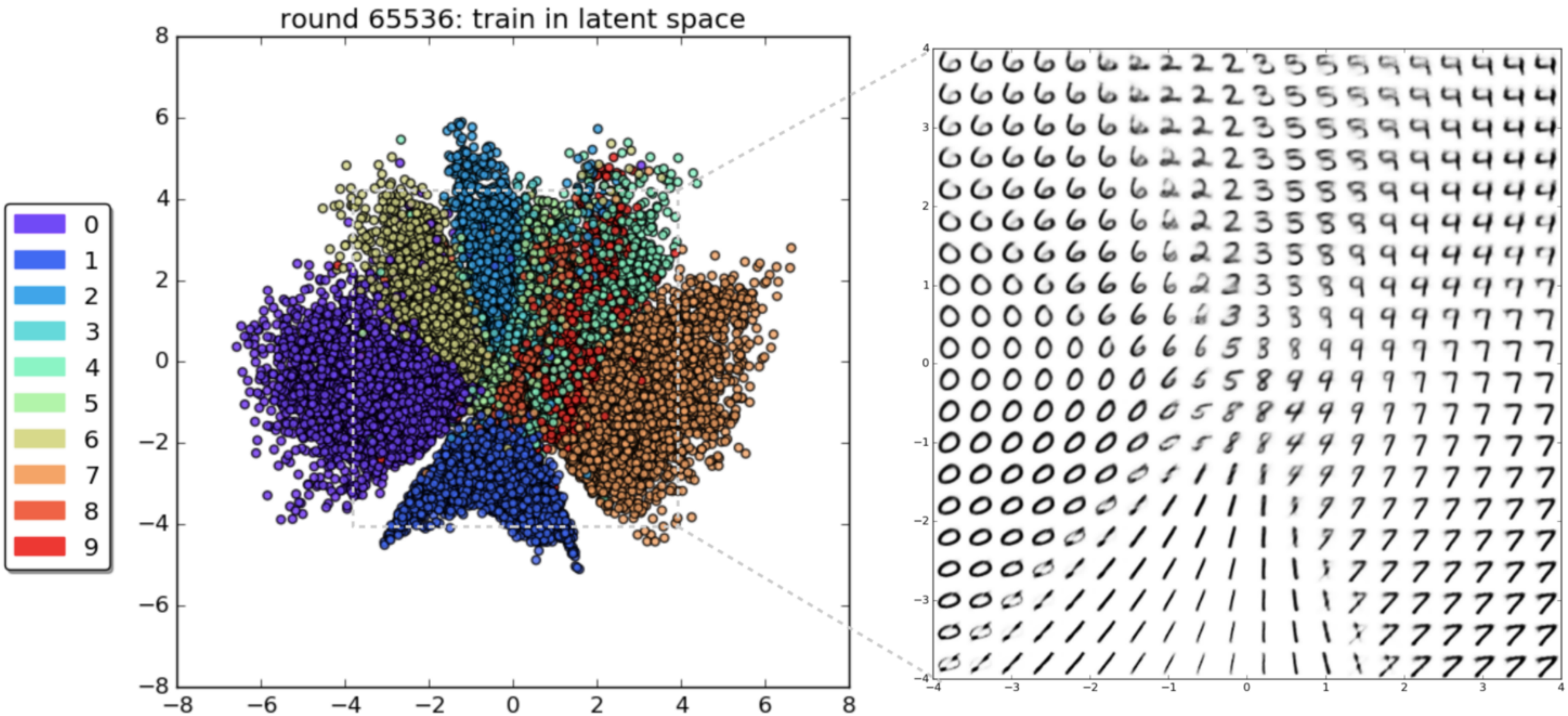
Variational Autoencoder



inference network | latent variables | generator network

$$loss = \underbrace{-E_{z\sim q_\phi(z|x)}(\log p_\theta(x|z))}_{\text{reconstruction loss}} + \underbrace{KL(q_\phi(z|x)||p(z))}_{\text{regularizing term}}$$

VAE latent space

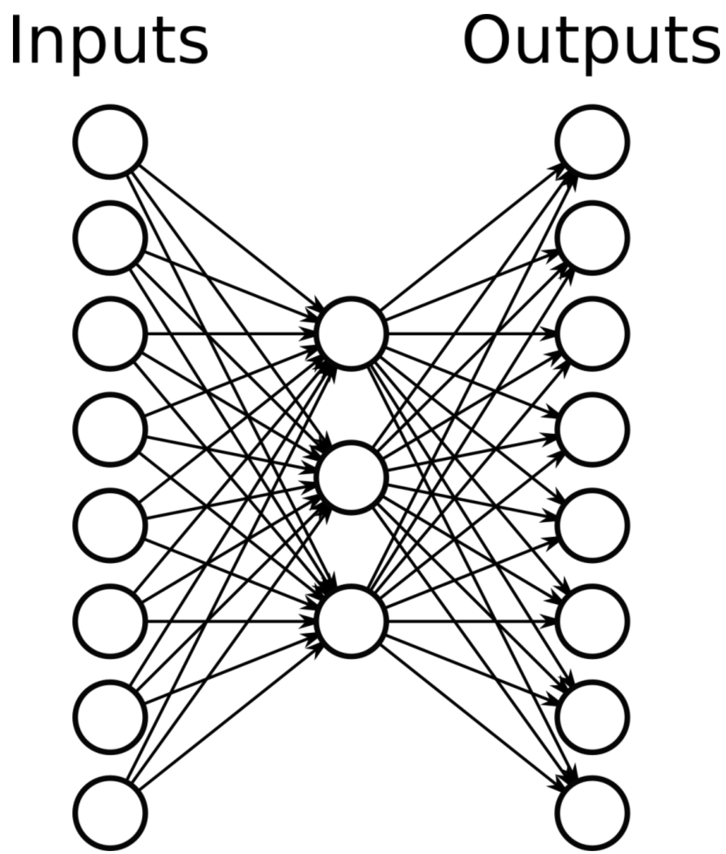


http://blog.fastforwardlabs.com/2016/08/12/introducing-variational-autoencoders-in-prose-and.html

Exercise

autoencoder toy example

Exercise



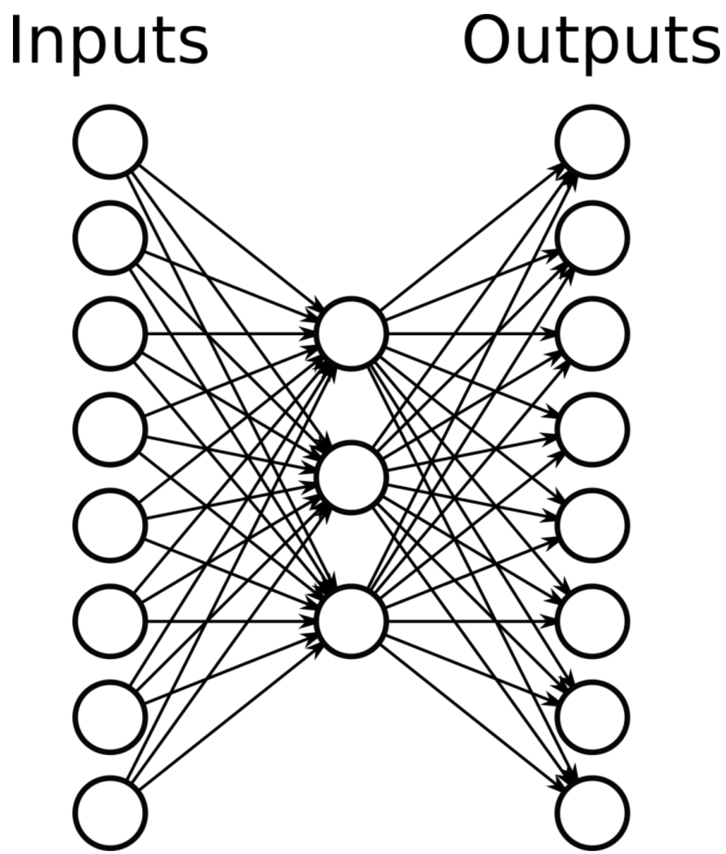
Inputs		Hidden Values		Outputs
10000000	→	.89 .04 .08	→	10000000
01000000	→	.01 .11 .88	→	01000000
00100000	→	.01 .97 .27	→	00100000
00010000	→	.99 .97 .71	→	00010000
00001000	→	.03 .05 .02	→	00001000
00000100	→	.22 .99 .99	→	00000100
00000010	→	.80 .01 .98	→	00000010
00000001	→	.60 .94 .01	→	00000001

Figures adapted from
T. Mitchell, *Machine Learning*, McGraw Hill, 1997.

What do you expect the network to learn?

Bonus question: Why is using sigmoid units a good idea here?

Exercise

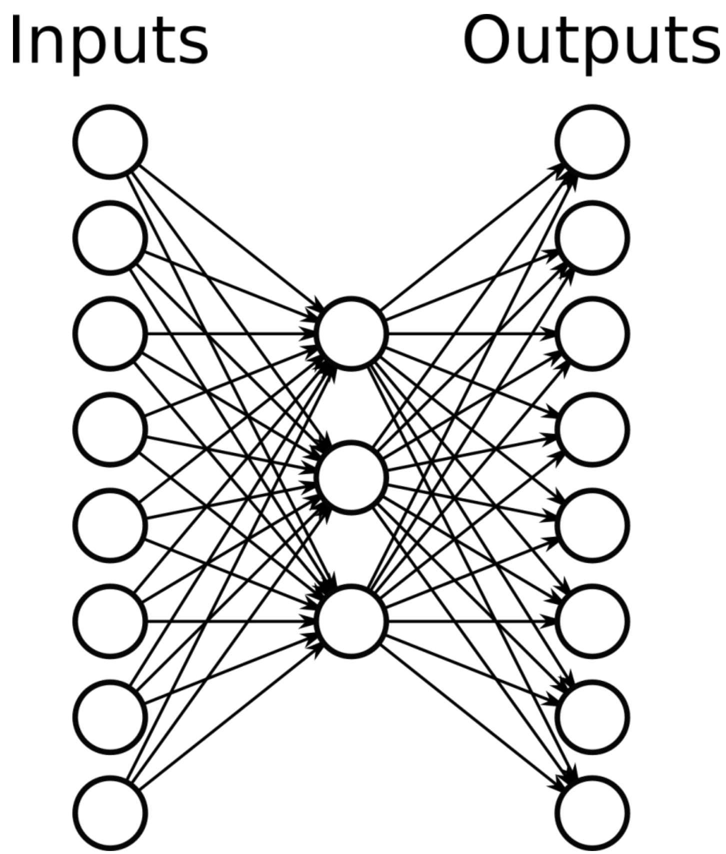


Inputs		Hidden Values		Outputs
10000000	→		→	10000000
01000000	→	any	→	01000000
00100000	→	mapping	→	00100000
00010000	→	to binary	→	00010000
00001000	→	code	→	00001000
00000100	→	possible	→	00000100
00000010	→		→	00000010
00000001	→		→	00000001

Figures adapted from
T. Mitchell, *Machine Learning*, McGraw Hill, 1997.

Model Capacity: Show that this network can represent any mapping to a **binary code** in the hidden layer using linear hidden units
(Bonus) and reconstruction using sigmoid output units and bias!

Exercise

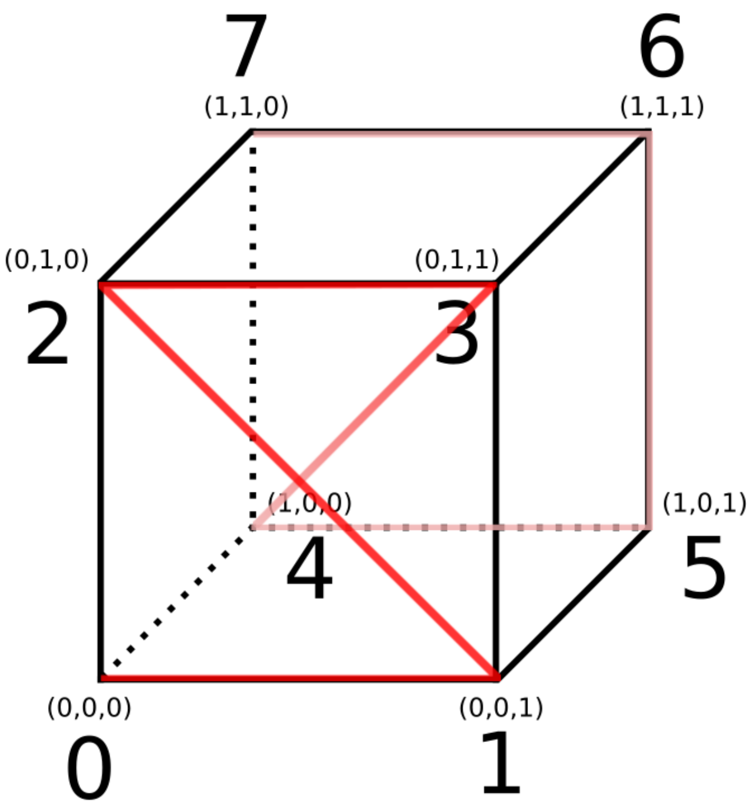
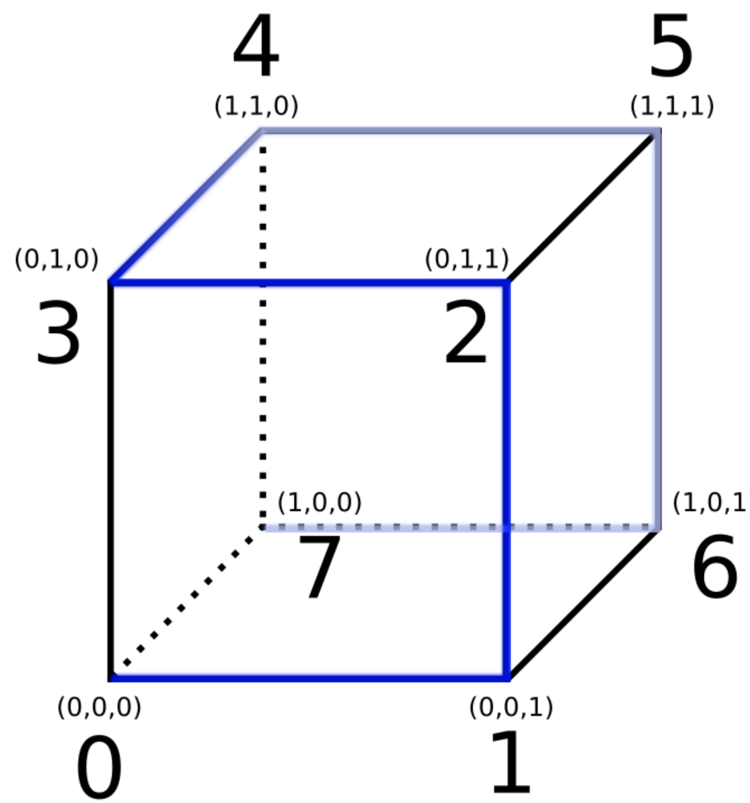


Capture order of the inputs

Inputs		Hidden Values		Outputs
10000000	→	0 0 0	→	10000000
01000000	→	0 0 1	→	01000000
00100000	→	0 1 1	→	00100000
00010000	→	0 1 0	→	00010000
00001000	→	1 1 0	→	00001000
00000100	→	1 1 1	→	00000100
00000010	→	1 0 1	→	00000010
00000001	→	1 0 0	→	00000001

Figures adapted from
T. Mitchell, *Machine Learning*, McGraw Hill, 1997.

Regularization: Some codes/representations might be more preferable than others.
How could an order of the inputs be encouraged and represented?

standard binary code	Gray code
	

(equidistance between neighbors)

Assignments

- Responsible for recap: nobody :(
- Reading on model introspection
- Participate in the Doodle poll (see Mattermost)
- Time for your project
- prepare a presentation about your project
  Write to me, if and how long you want to present!

Slides & assignments on: https://mlcogup.github.io/idl_ws18/schedule