MSc. Thesis - Deep Restoration
Progress Updates

Frederik Harder

July 13, 2017

sooner rather than later

1. skim simonyan papers, find the relevant ones
2. do search on samplings from FoE models
3. revisit clip op in mv16

July 13.

1. Week goal 1: send summary + timeline out to potential examiners before friday.
2. read papers from jörn about quality metrics
3. draft related works (so the big names are connected)

July 12.

1. reworked summary:
   split methods and results
   gave some more details on result setups
   drafted motivation

July 11.

1. refactoring, integrated M&V into net inversion class
2. generated first deep ica reconstruction

July 10.

1. found out, why first ICA prior results are so bad. (sigma missing)
2. trained first batch of foe priors (looking very similar to ica priors.)
3. trained ica prior with more (1024) components: filters look similar. laptop gpu out of memory when computing ICA prior (but only after about 100 steps. what's going on there?)
4. trained ica prior reconstruction with different weightings. lower weights are slower to converge, but result in crisper images.

July 2.

1. drafted some thoughts on sampling
2. think about weighting maybe normalize priors for same average score on natural images?

July 1.

1. set up FoE prior

June 30.

1. worked out score matching for field of experts
2. read patternnet paper

June 29.

1. drafted timeline
2. drafted summary + goals from now on

June 15.

Completed

1. enable loading option for optimizer parameters (done)
2. add split loss option per module (done)
3. work through and re-implement mahendran & vedaldi's paper (done)
4. test M&V model on vgg and reproduce alexnet results (done)

5. adapt code to allow stacking models (done - for now)
6. pretty up plotting functions (done - for now)
7. continue training of later vgg layers to actual convergence (done - for now)
8. consider adding parts of curet texture data set to selected images (decided not to - for now)
9. implement deconvs as upsampling-conv operations
10. implement score matching ICA and overcomplete ICA models from Hyvärinen
11. adapt code to allow arbitrary losses and priors

Results

Figure 1: Mahendran & Vedaldi: alexnet reconstructions

Figure 2: Mahendran & Vedaldi: vgg16 reconstructions

Next Steps

1. Some Deconv-Conv models need redoing (with upsample-conv-conv)
2. MV-2016 code is not producing good results yet
3. get Hyvärinen ICA to work
4. expand into field of experts
5. repeat MV work with ICA priors
6. explore sampling options
7. write-up in paper form

May 10.

Completed

- change architecture: each module specifies in and out tensor (optionally either from last module or classifier) and whether reconstruction becomes part of the loss. enables flexible training of multiple modules at once
- create reconstructions for deeper vgg layers and alexnet
- enable loading weights to continue training from previous sessions
- create resized dataset for runtime speedups

Results

Three things:

1. Reconstructions in Vgg from the first pooling layer suggest that 3 stacked modules which are individually trained on each operation (conv1_1-conv1_2-pool1) yield better results than the same model trained end-to-end.
2. Reconstruction quality decays gradually, as expected, when starting at deeper layers in vgg.
3. In alexnet, the Local Response Normalization layer rapidly decreases reconstruction quality. A lot of information is lost in this step, explaining the results obtained by Dosovitskiy and Brox.
4. A single initial test (more to come) indicates that multiple modules can be trained together, taking other reconstructions as input and optimizing the sum of the individual losses.

Figure 3: AlexNet: pool1 to input. two modules trained simultaneously

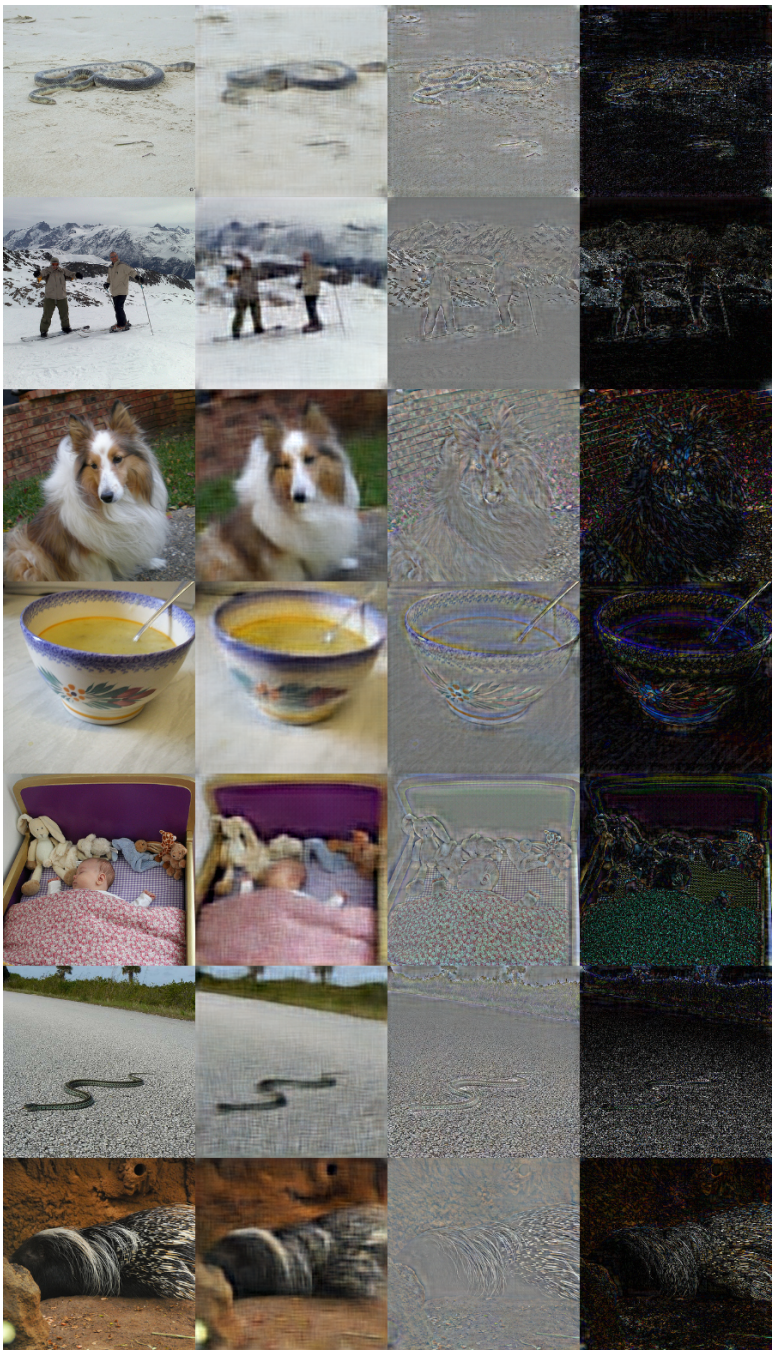

Figure 4: AlexNet: pool1 to input. two modules stacked

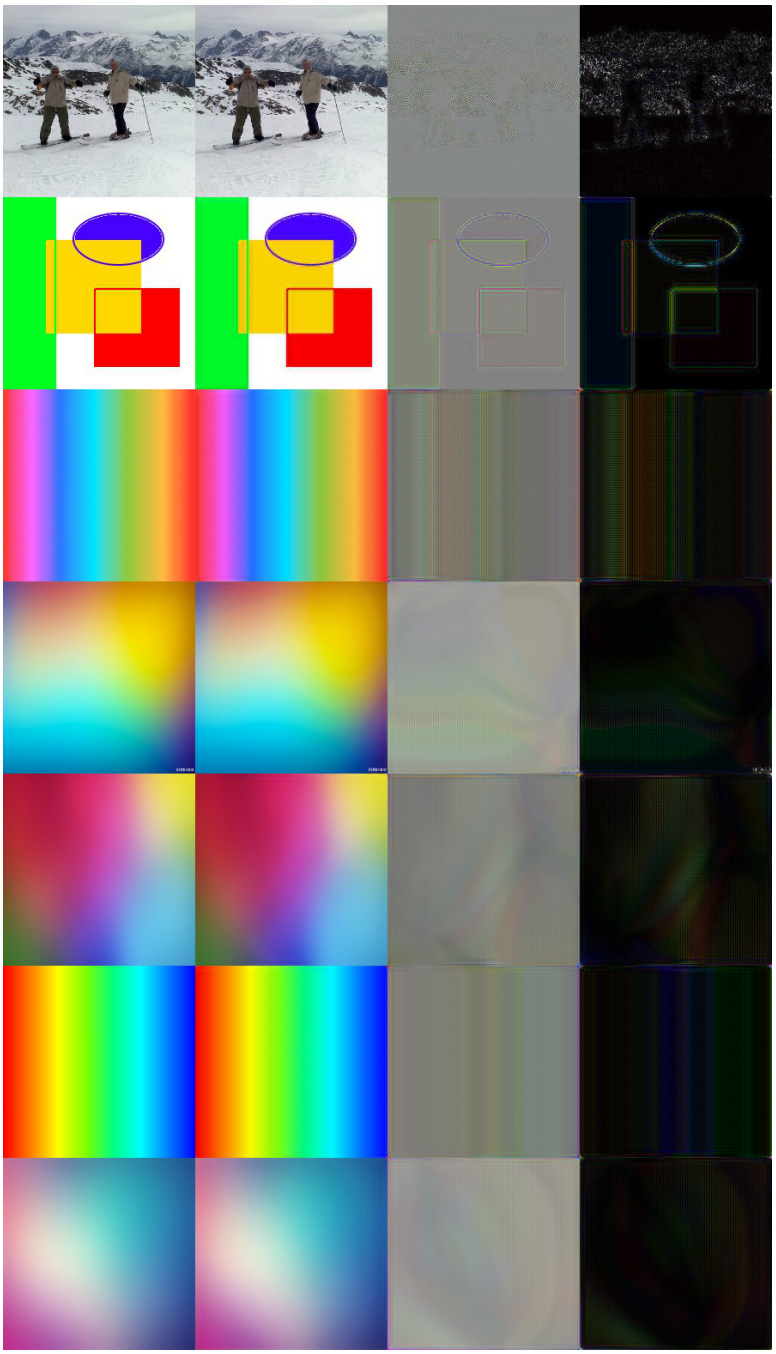

Figure 5: Vgg16: pool1 to input. 3 modules trained end to end

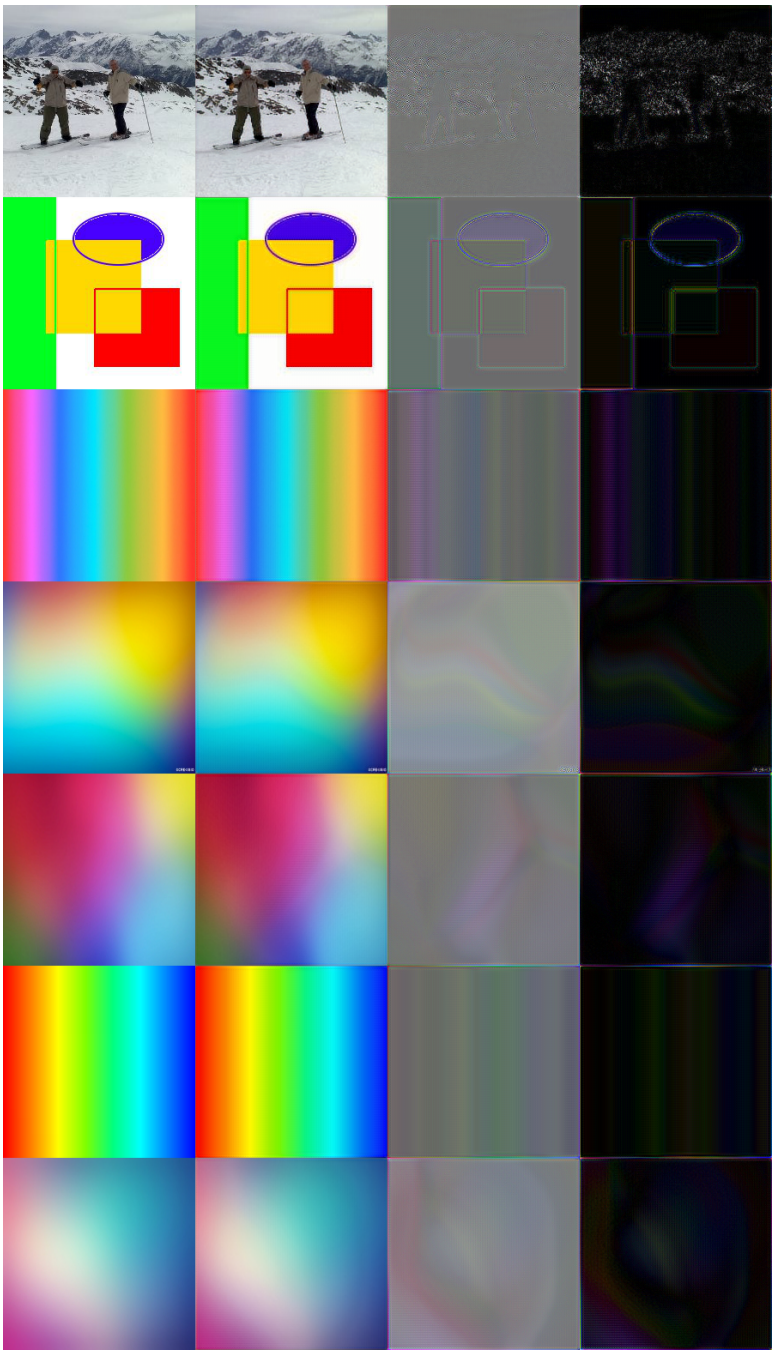

Figure 6: Vgg16: pool1 to input. 3 modules trained separately and stacked

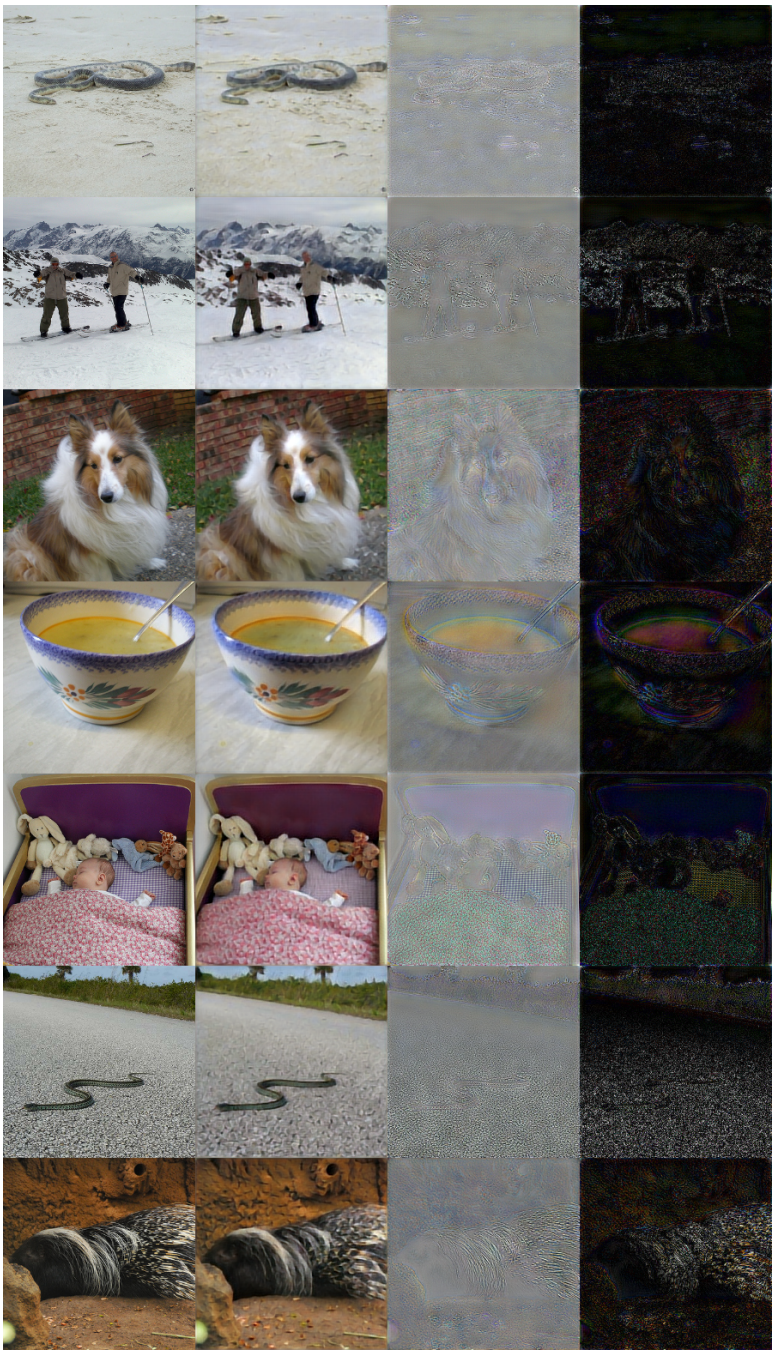

Figure 7: Vgg16: pool2 to input: 6 modules stacked

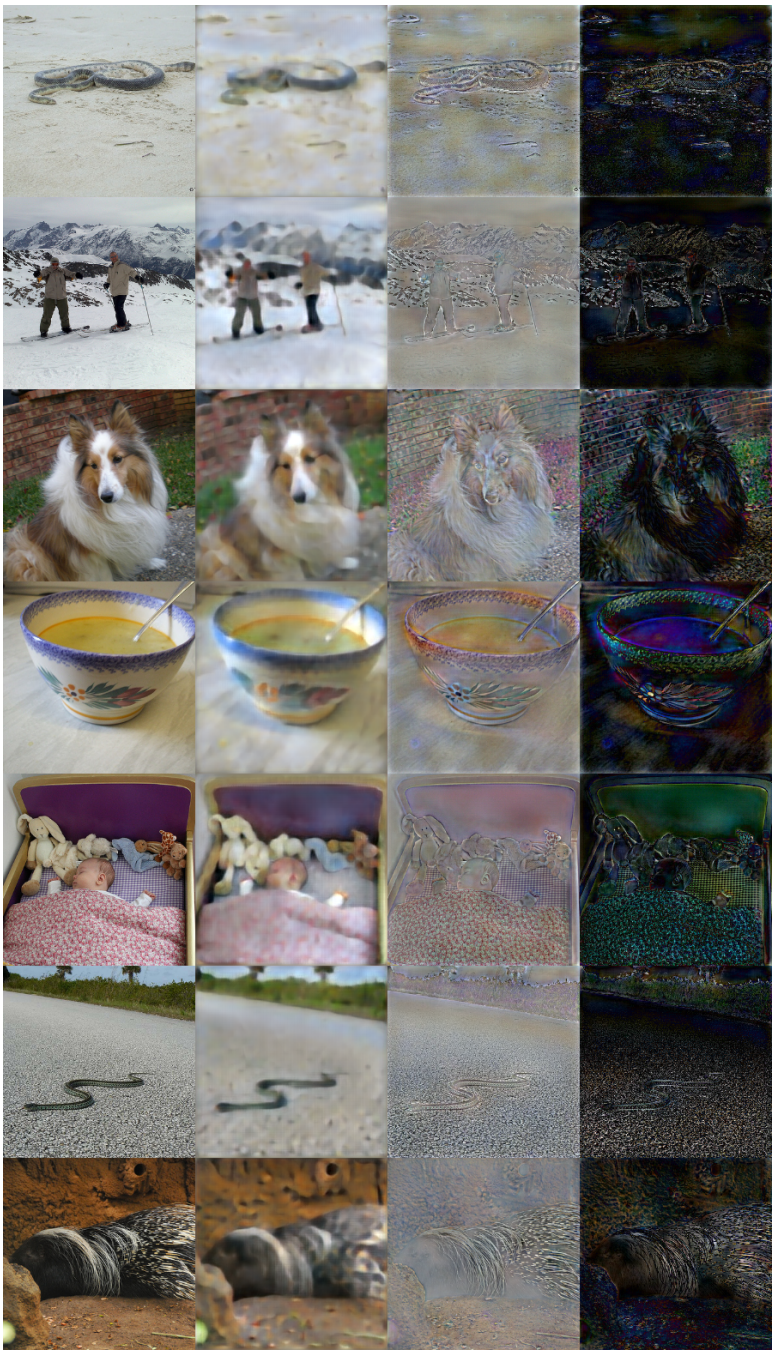

Figure 8: Vgg16: pool3 to input: 10 modules stacked

Next Steps

- add option for loss logging per module
- continue training of later vgg layers to full convergence
- build caffe and load dosovitskiy's model weights. rebuild model in tensorflow.
- add curet texture database to selected images
- re-implement mahendran & vedaldi
- build sparse coding modules
- plan out remainder of the project and register the thesis

May 3.

Completed

- step to later layers, for simplicity with vgg first.
- adapt code to allow stacking models (somewhat hacked together, needs rework later)
- evaluate which model (cd, dc, dd) works best on alexnet and vgg
- reproduce cd runs on alexnet
- vary learning rate
- redo vgg experiments with BGR mean order
- test artificial data with uniform areas to test vggnet
- track source of black spots, try renormalizing again
- rework run logging, so used parameters can be read off
- unify vvg and alexnet layer inversion classes
- track validation set loss
- find good way to log loss per channel
- pretty up plotting functions (should be sufficient for now)

Results

This week has yielded two major findings:

1. the first three layers of Vgg16 (two convolutions and one pooling layer) can be inverted with decent results using three stacked convolution-deconvolution models trained individually.

2. both deconvolution-convolution and double deconvolution model (where the first operation has stride) yield fewer artifacts when inverting the first Alexnet layer (conv+relu)

Also, the grey spots and color inaccuracies found in last weeks results were apparently owed to the model's slow learning of color biases and disappear entirely, when the normalized BGR image is used as a target.

The figures below show reconstructions of the first 3 stacked Vgg16 layers and of the first Alexnet layer using the three different models. Because differences between image and reconstruction can be quite small, two difference measures have been added. the first is a color/channel accurate measure, showing the difference between image and reconstruction, renormalized to the interval [0,1]. So if, for example, the reconstructed image has less blue in it, the measure will be blue. Areas without error (or equal errors across channels) are grey. The second difference measure is the renormalized absolute difference, which highlights areas with errors more clearly. Areas without error are black. Given renormalization, pixel intensity only informs about relative error within the image, not absolute error compared to other images. Without normalization, the errors were hardly visible. The figures are ordered left to right as: image, reconstruction, color measure, absolute measure.

Figure 9: reconstruction from pool1 layer by 3 stacked conv-deconv models. Left to right: Image, reconstruction, color accurate error, relative intensity accurate error

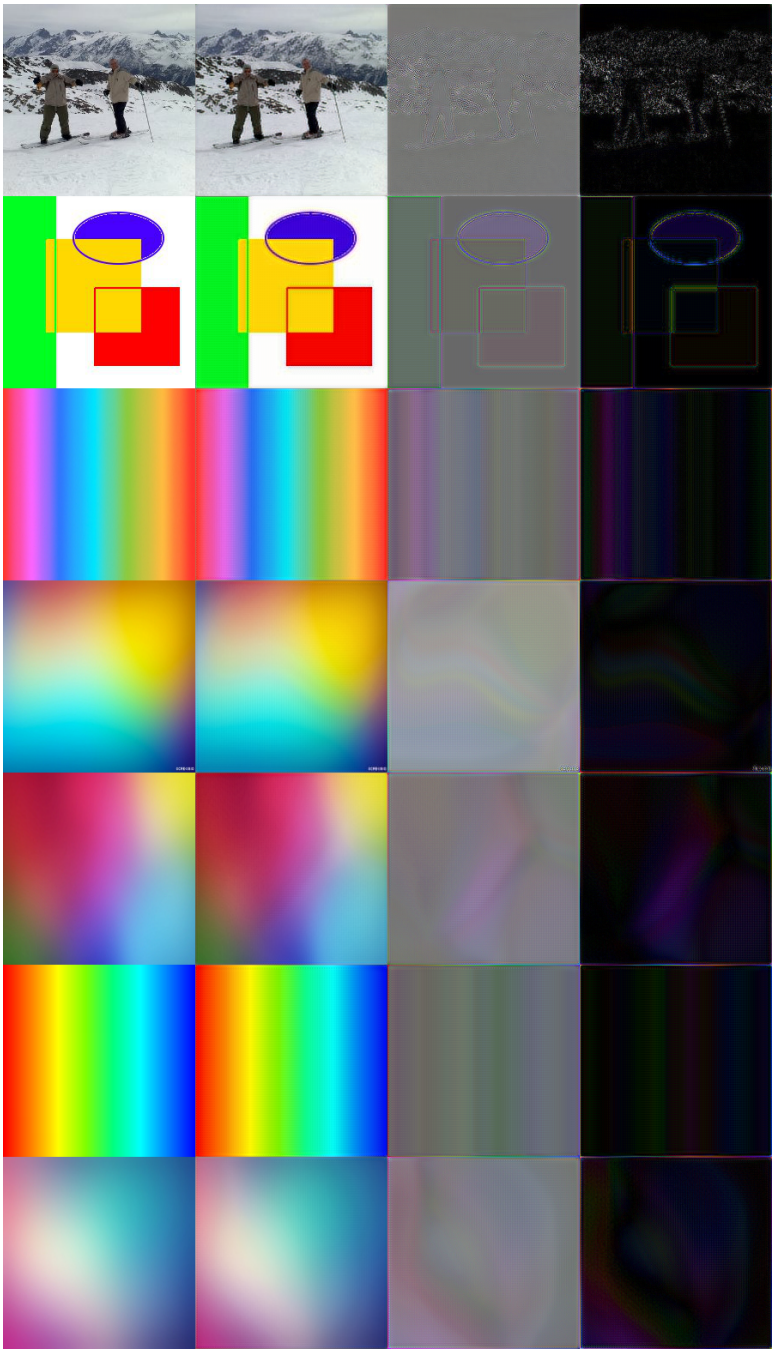

Figure 10: more reconstructions from pool1 layer by 3 stacked conv-deconv models

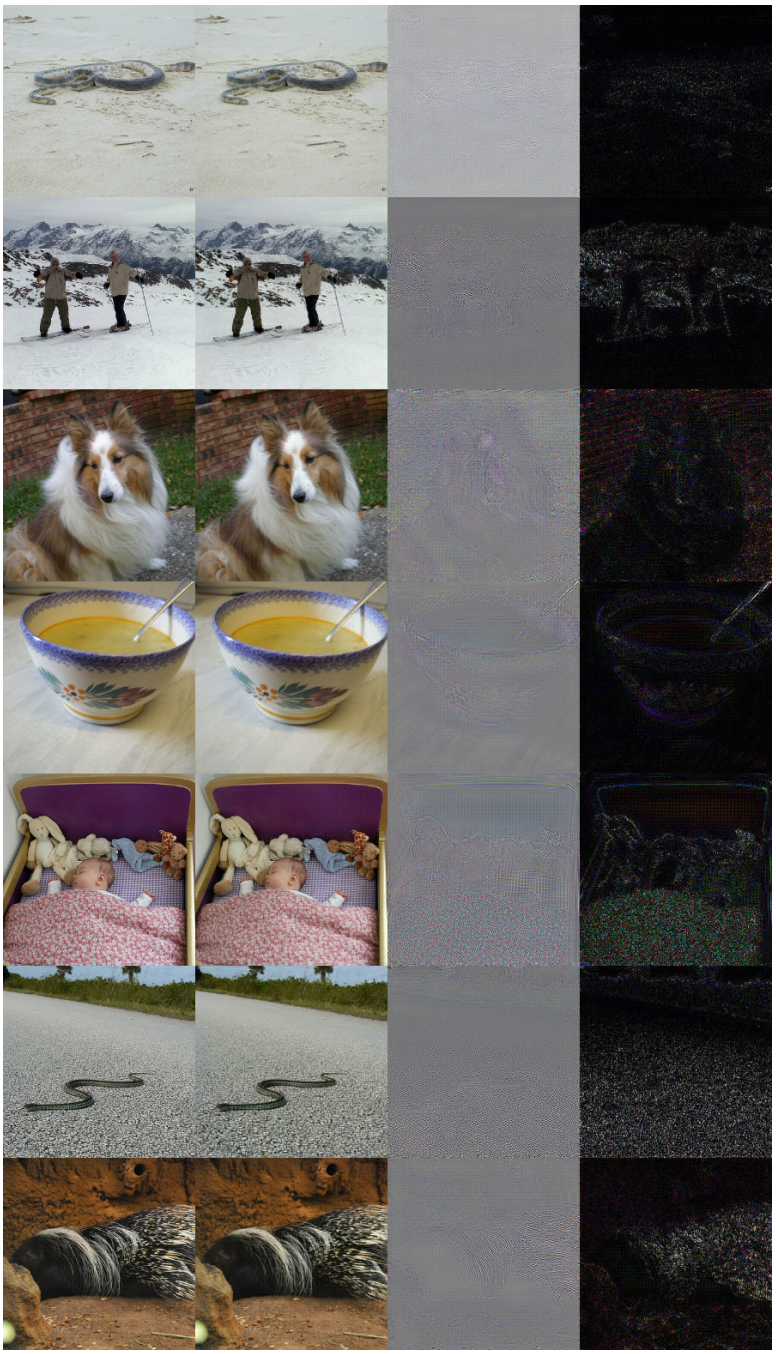

Figure 11: reconstructions of the first alexnet layer using Conv-Deconv model

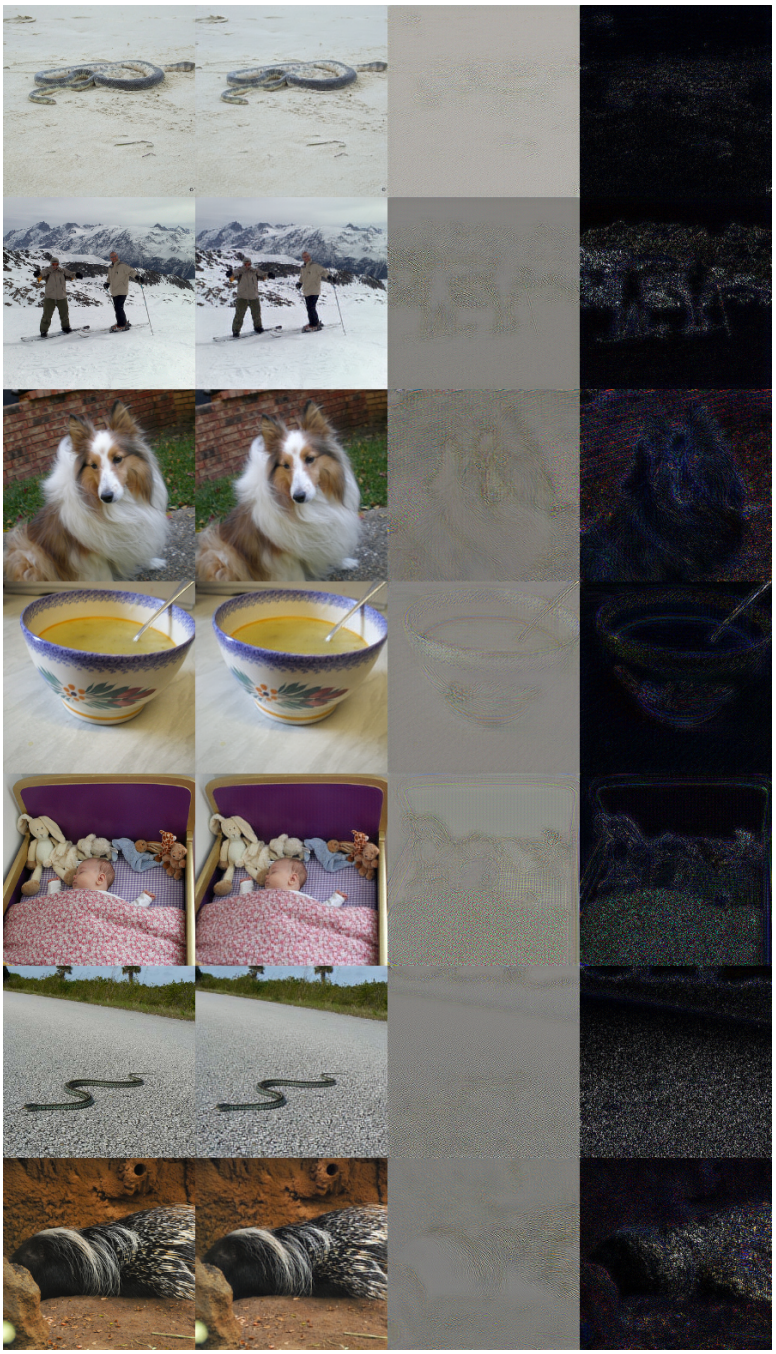

Figure 12: reconstructions of the first alexnet layer using Deconv-Conv model

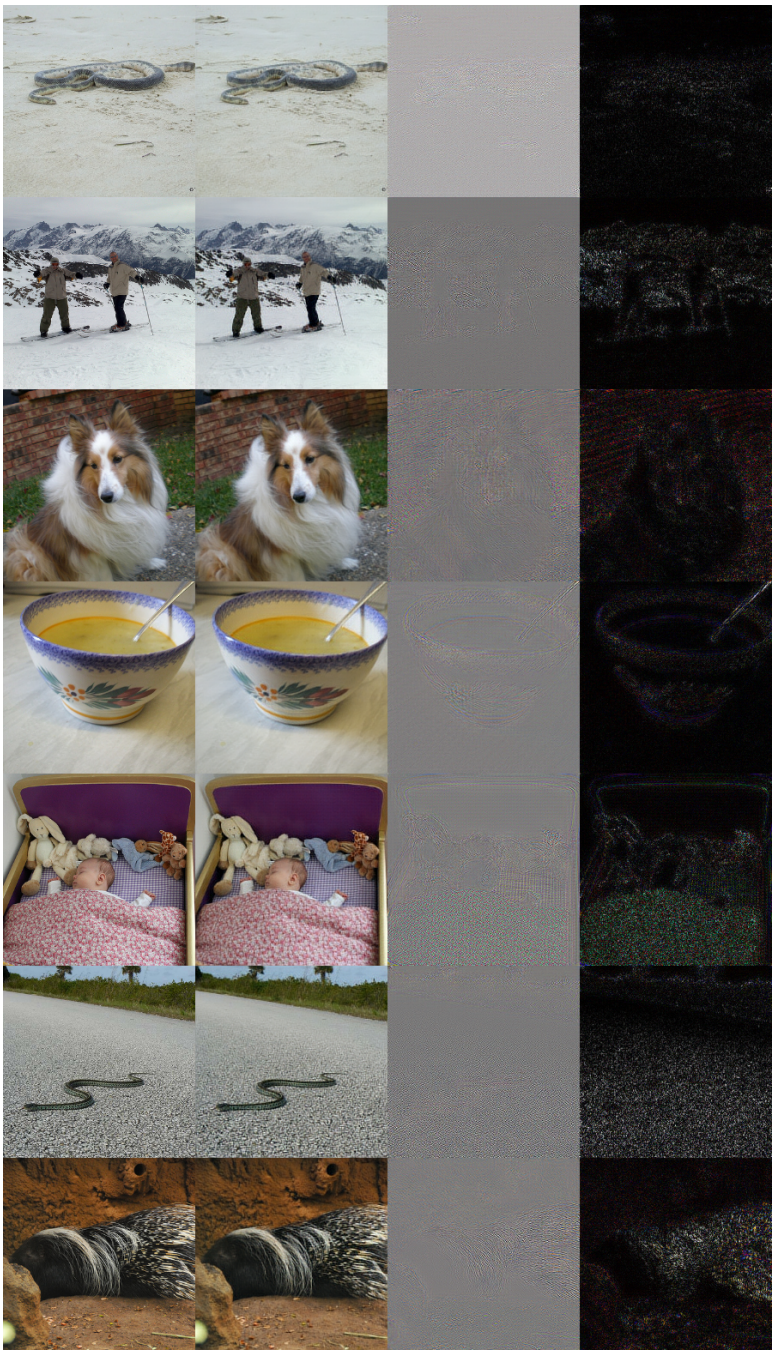

Figure 13: reconstructions of the first alexnet layer using Deconv-Deconv model

Next Steps

- invert pool1 with a deconv conv model, which deals better with strides
- train and stack 3 layer inversions, compare to same model trained in one go.
- find second source on image net means
- increase number of stacked layers and extend results to alexnet
- work out details of ICA based model
- specify roadmap for remaining thesis and officially register the project