# MSc. Thesis - Deep Restoration
# Progress Updates

Frederik Harder

August 15, 2017

## sooner rather than later

1. skim simonyan papers, find the relevant ones
2. do search on samplings from FoE models
3. at some point look into hybrid mc sampling

## 1 August 15.

1. train lc - gc 1800f prior to convergence (amlab)
2. train lc - rescale 1200f prior (laptop)
3. train channel prior (laptop)
4. write part on score matching, whitening

## 2 August 14.

1. train conv1 lin prior with local full mean (center each sample) and global channel standard deviation normalization
2. filters don't look so good yet
3. changed log-sum-exp expression in ica prior to a more special case, which should be numerically stable no matter what.

## 3 August 13.

1. pre-image init was bad. could perhaps still be tweaked (right now, adam needs to be restarted after the initial 500 steps, because its parameters get much too conservative)

2. reproduced previous results on new pipeline (finally)

3. feature maps still have way more eigenvalues close to zero.

# 4 August 12.

1. write out all the preprocessing

   (a) target and pre-image in range 0 - 255

   (b) if no sdev normalization takes place, images are trained on rescaled patches in range 0-1 for later numerical stability of the ica prior.

   (c) check that all the variables are loaded correctly

# 5 August 9.

1. integrate mean and sdev loading into the priors, adapt data-filenaming conventions

2. create image dataset with gc,gc normalization

3. enforce consistent feature map flattening - always flatten from [channels, height, width] - so each feature map lies on a continuous interval on the vector

4. train gc,gc image ica prior

5. train and test conv1lin feature map priors (1 ica, 1 cica)

# 6 August 8.

1. integrate mean and sdev processing funtion into the priors

2. reproduce image priors (1 ica, 1 cica)

# August 7.

1. investigate assumptions behind current patch whitening procedure

   (a) typically one is suppose to center and rescale each feature before using pca & whitening

   (b) by stationarity assumption of stationarity, every feature should have the same mean. Subtracting the patch mean per channel (given equal variance) then takes care of the feature mean as well

   (c) on natural images roughly equal variance across features (including different channels) is assumed.

(d) the stationarity assumption should hold for feature maps as well. the assumption of equal variance, however, is not as well-motivated.

(e) comparing patches from conv2/relu, conv1/lin_cw and image layer, this can be observed to some degree, where standard deviations by channel range from roughly 20 to 30 in the image layer and from 2 to 70 and 12 to 300 in the other two layers. notably, within a channel in conv1/lin_cw the values are again quite consistent, which supports the assumption of stationarity again.

2. consider adding variance normalization, creating a new preprocessing pipeline

   (a) store truly raw (non-centered) patches in raw_mat

   (b) center all features by respective channel mean (global mean seems ill-advised even for priors with no channel independence assumptions), store channel means for retrieval by prior

   (c) then divide features by channel standard deviation. store those as well for retrieval by the prior

   (d) only then accumulate the covariance matrix

   (e) proceed with eigen-decomposition for pca whitening.

3. there is also a second option, which throws away a little more information, but might be less prone to a bad set of training patches (and maybe the information thrown away is irrelevant in the first place)

   (a) as in 2. but instead of training set mean, take the sample mean per channel (like right now)

   (b) also divide by the sample standard deviation. this step might be throwing away too much though. could also use sample mean and stored std_dev instead

4. So the necessary implementations steps look like this

   (a) separate out raw mat generation (channel separate even for full cov) (done) separate patch extraction and covariance accumulation (done)

   (b) add switches (done) mean: sample/dataset, channel/global sdev: none/sample/dataset (global would just be a rescale)

   (c) implement switch functionalities (partially done)

## August 6.

1. compare cica results to foe and ica results

2. visualize generated conv1 feature maps for different priors

3. try CICA conv1 priors with bigger patches

## August 5.

1. image CICA prior tests. results roughly on par with m&v 2016, but lack behind ICA prior (6e-7 weight, no img scaling)
2. train conv1/lin and conv2/lin CICA priors, set up single layer inversion experiments, track

## August 4.

1. half day, got CICA to finally work

## August 3.

1. set up independent ICA prior (more work than expected)
2. visualize feature maps

## August 2.

1. set up independent ICA prior (more work than expected)
2. set up small inversion scenario with mse loss tracking and subsequent projection to image layer

## August 1.

1. set up independent ICA prior
2. trained conv2lin foe prior on 6400 comps

## July 31.

1. track ICA convergence problems
2. tried FoE prior in comparison

## July 29.

1. started setting up channel-independent ica prior
2. found convergence problems training large ICA priors

## July 28.

1. finished full conv2relu prior training
2. got das4 access

## July 27.

1. meeting with jörn
   (a) look into distributed computation for tensorflow
   (b) sign up for das4
   (c) visualize learned filters
   (d) consider making ica priors for independent feature maps (independence assumptions are strong, but the computations save a lot of space)
   (e) consider training priors not on relu outputs but on conv output, because structure might be smoother
   (f) isolate even smaller subproblem: invert single conv block (e.g. pool2 to pool1 or conv2/lin to conv1/lin). visualize reconstructed feature maps with and without prior.
   (g) set up L-BFGS as optimizer for potentially better results
2. set up filter vis for feature map patch priors
   (a) dim reduced priors look wavelet-ish which is good but could be due to pca
   (b) non-reduced priors must be done remotely. training conv2 relu prior for that
3. set up up L-BFGS as pre-img optimizer (need to set up callbacks/logging) initial results don't look very good though

## July 26.

1. ran 32k prior series
2. fixed some logging inefficiencies

## July 25.

1. half day (Bosch interview + prep)
2. accumulated 1k prior results, experimented with img prior weighting

## July 24.

1. off day (BCCN interview, prepping Bosch interview)

## July 23.

1. experiment with combinations of conv4 mse loss, img ica prior and conv2 and conv3 ica priors
2. different weightings of (dim reduced) conv2 prior (laptop)
3. differnce between dim reduced and full conv3 prior (amlab pc)

## July 23.

1. off day

## July 21.

1. trained and experimented with dimensionality reduced feat map priors

## July 20.

1. some learning rate experiments with M&V 16 code
2. trained conv2/relu and conv3/relu priors
3. compiled update

## July 19.

1. made patch data for pooling layer 1
2. trained pool1 3x3 prior (with little to no discernible effect)

## July 18.

1. read invertible resnet paper
2. revisited rim for application to net inversion

## July 17.

1. revisited clip op in mv16
2. got mv16 to mostly work. colors in fc reconstructions still seem a bit off (less green bias)
3. started setting up feature map patch data creation

## July 14.

1. sent mail to Max Welling
2. worked on M&V 16

## July 13.

1. read papers from jörn about quality metrics
2. draft related works (so the big names are connected)
3. ongoing: more ica based reconstructions

## 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

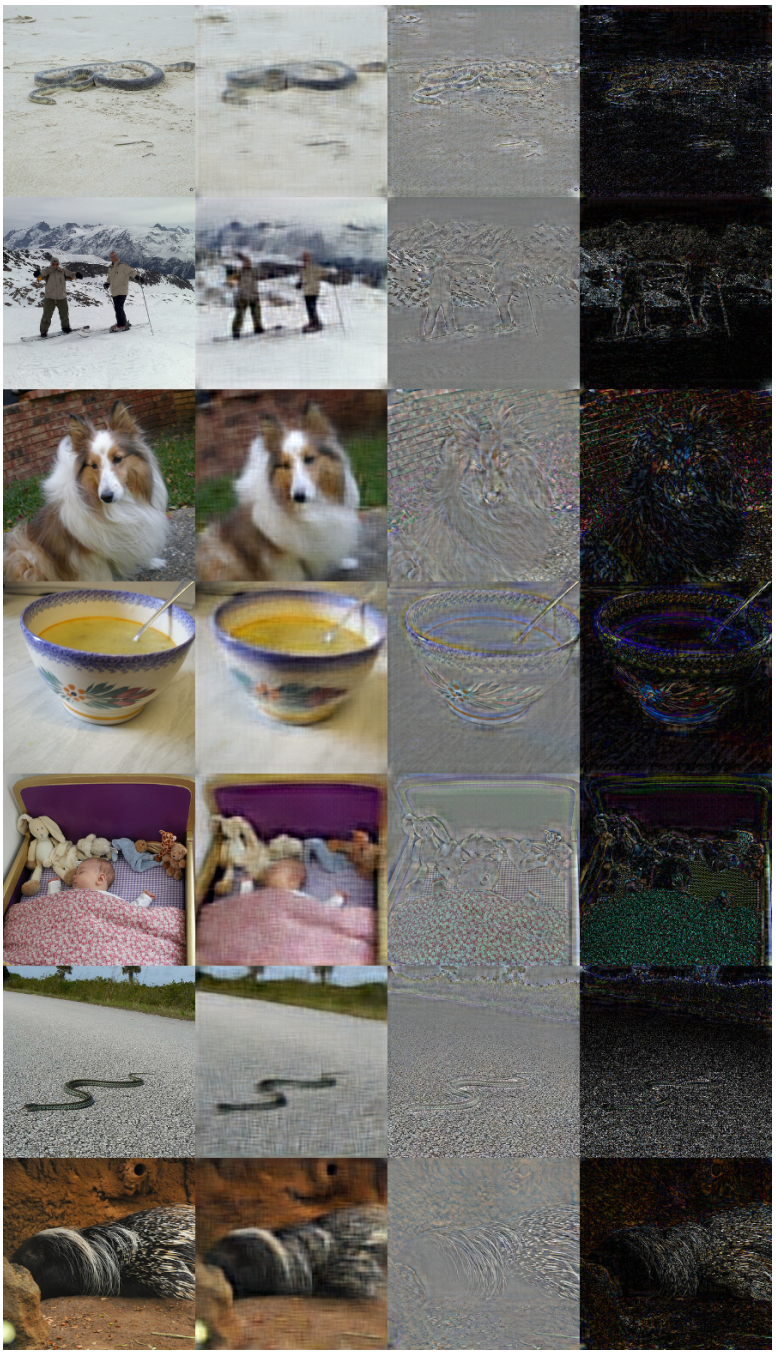

Figure 4: AlexNet: pool1 to input. two modules stacked

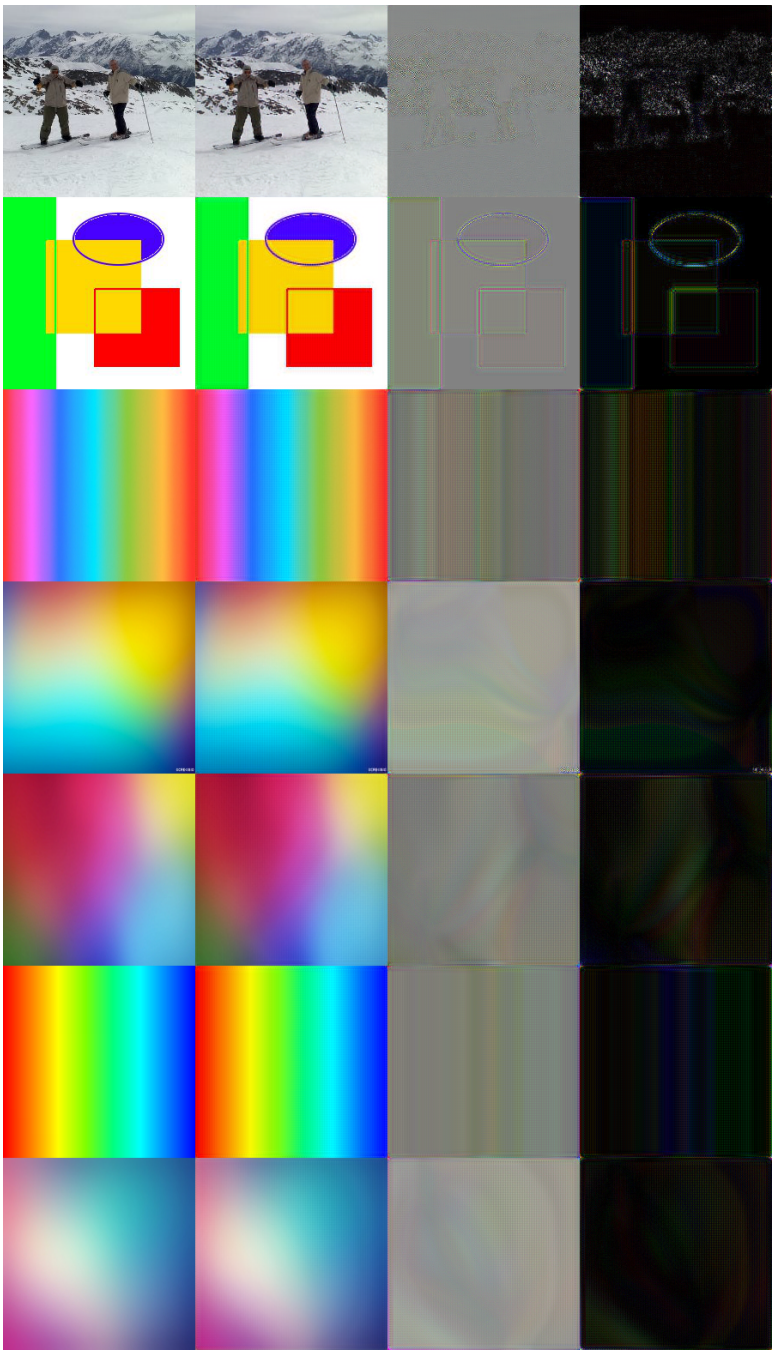

Figure 5: Vgg16: pool1 to input. 3 modules trained end to end

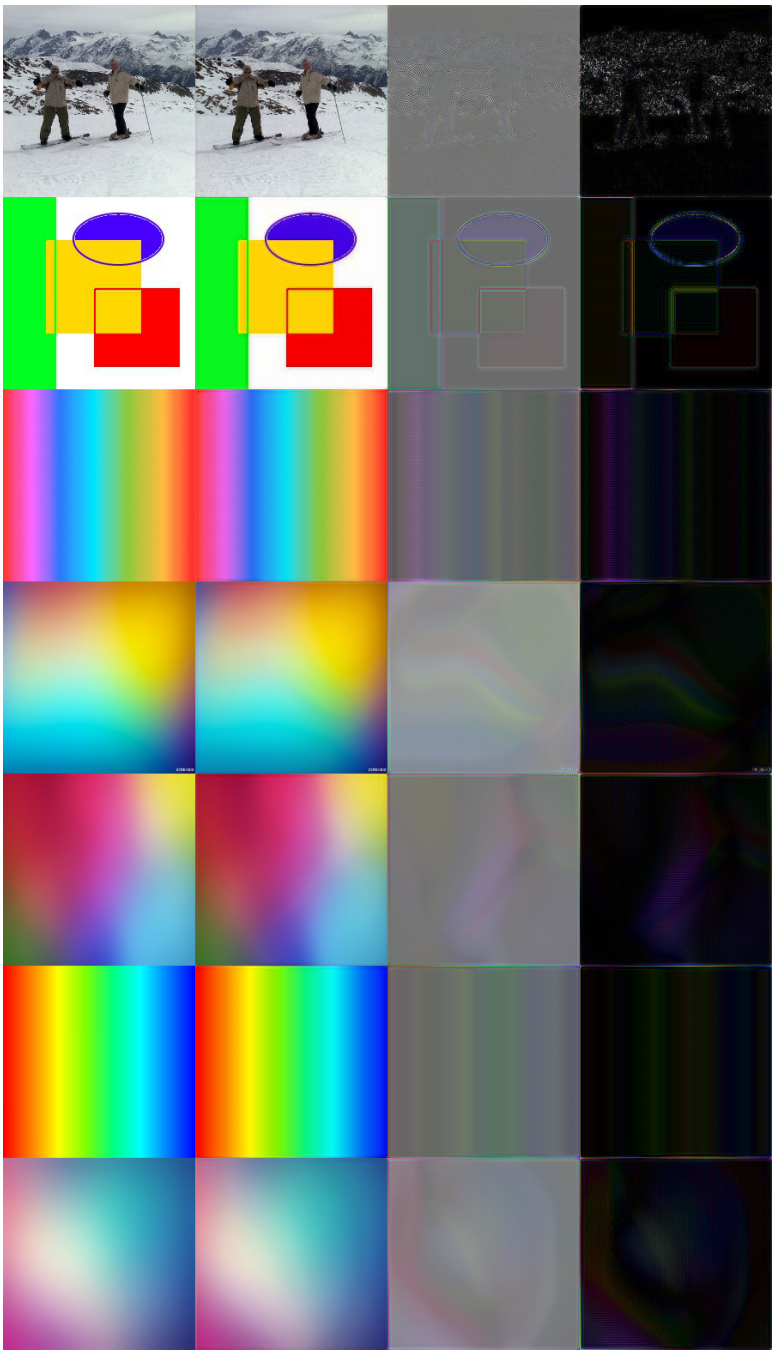

Figure 6: Vgg16: pool1 to input. 3 modules trained separately and stacked

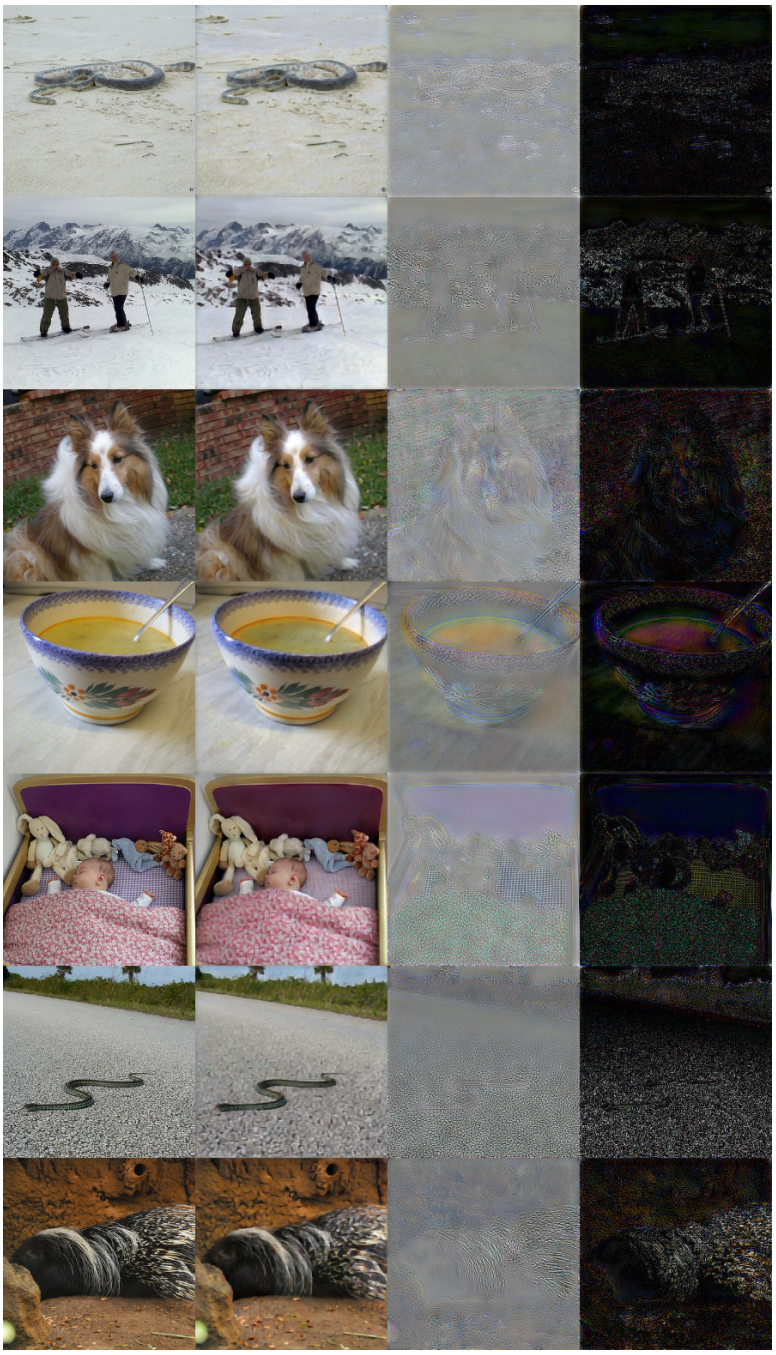

Figure 7: Vgg16: pool2 to input: 6 modules stacked

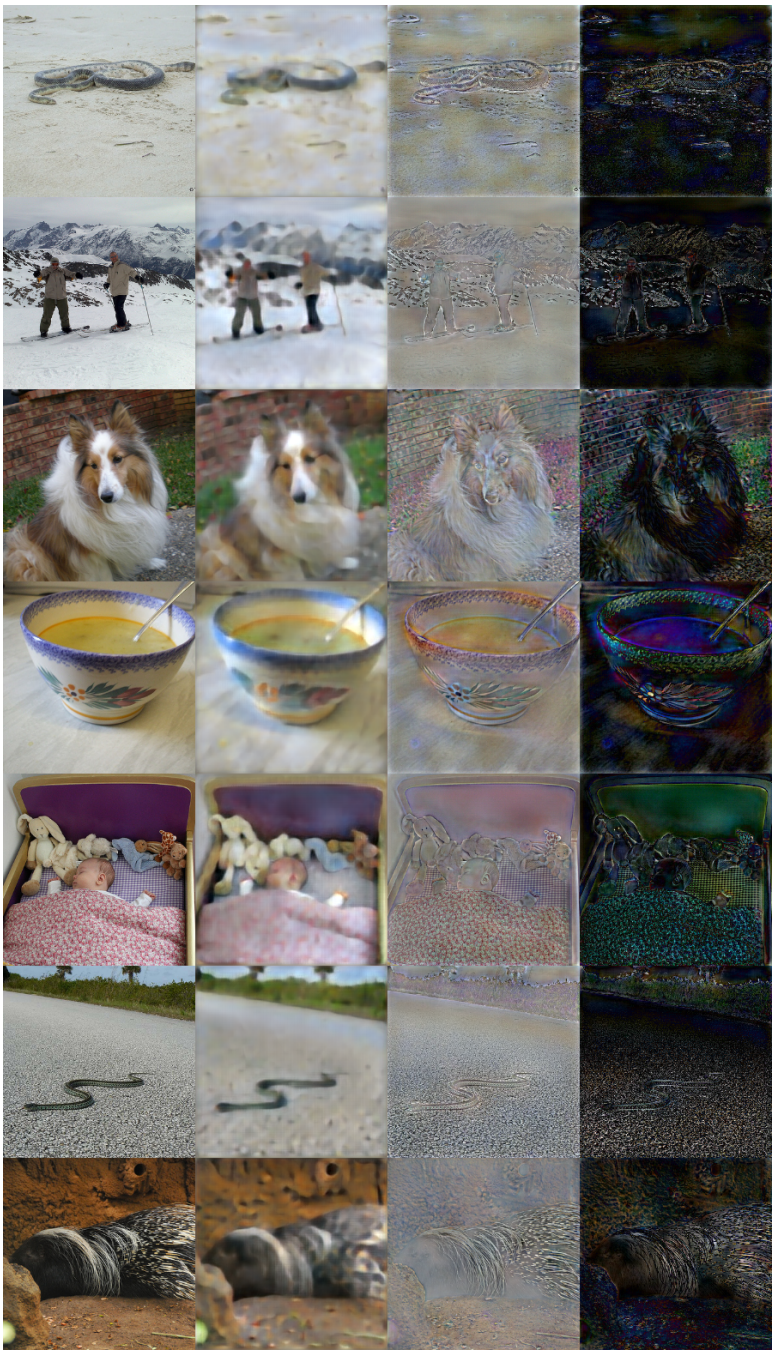

Figure 8: Vgg16: pool3 to input: 10 modules stacked

### 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

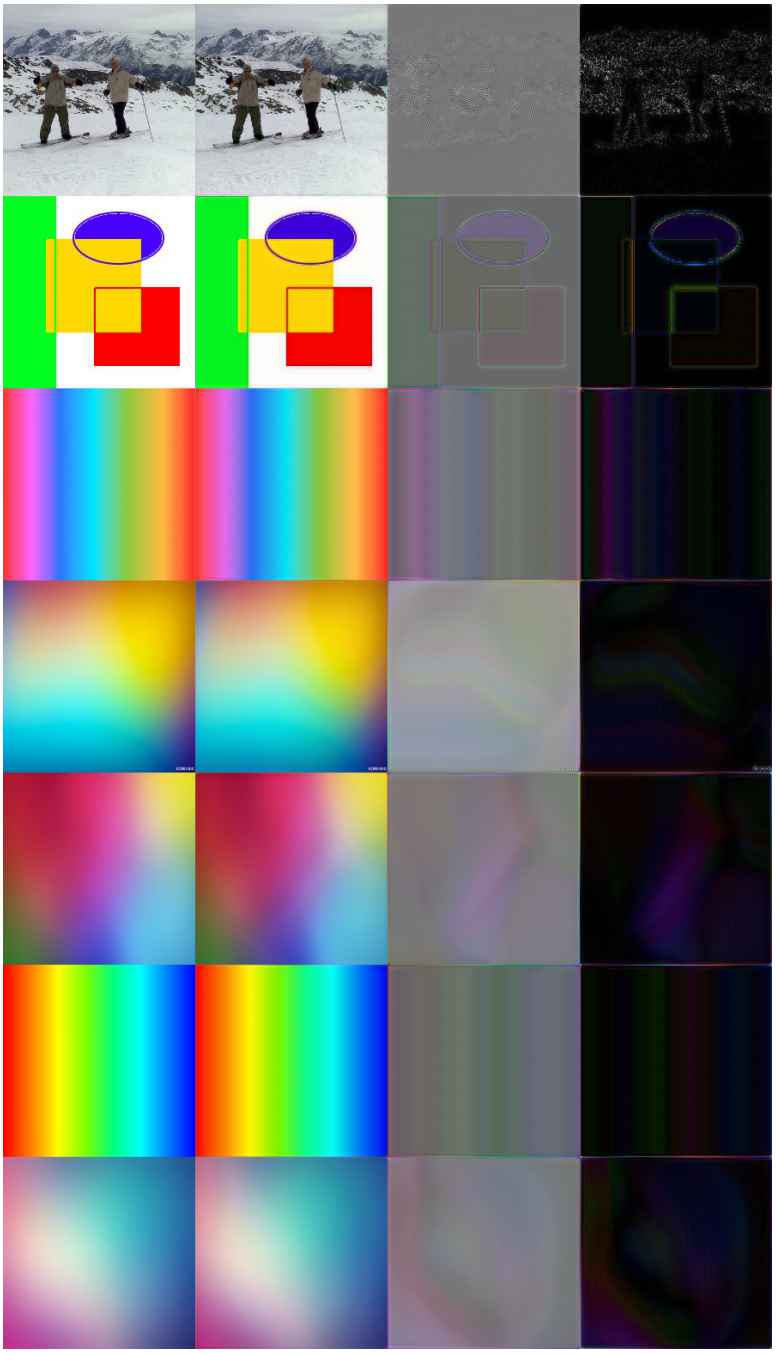

Figure 10: more reconstructions from pool1 layer by 3 stacked conv-deconv models

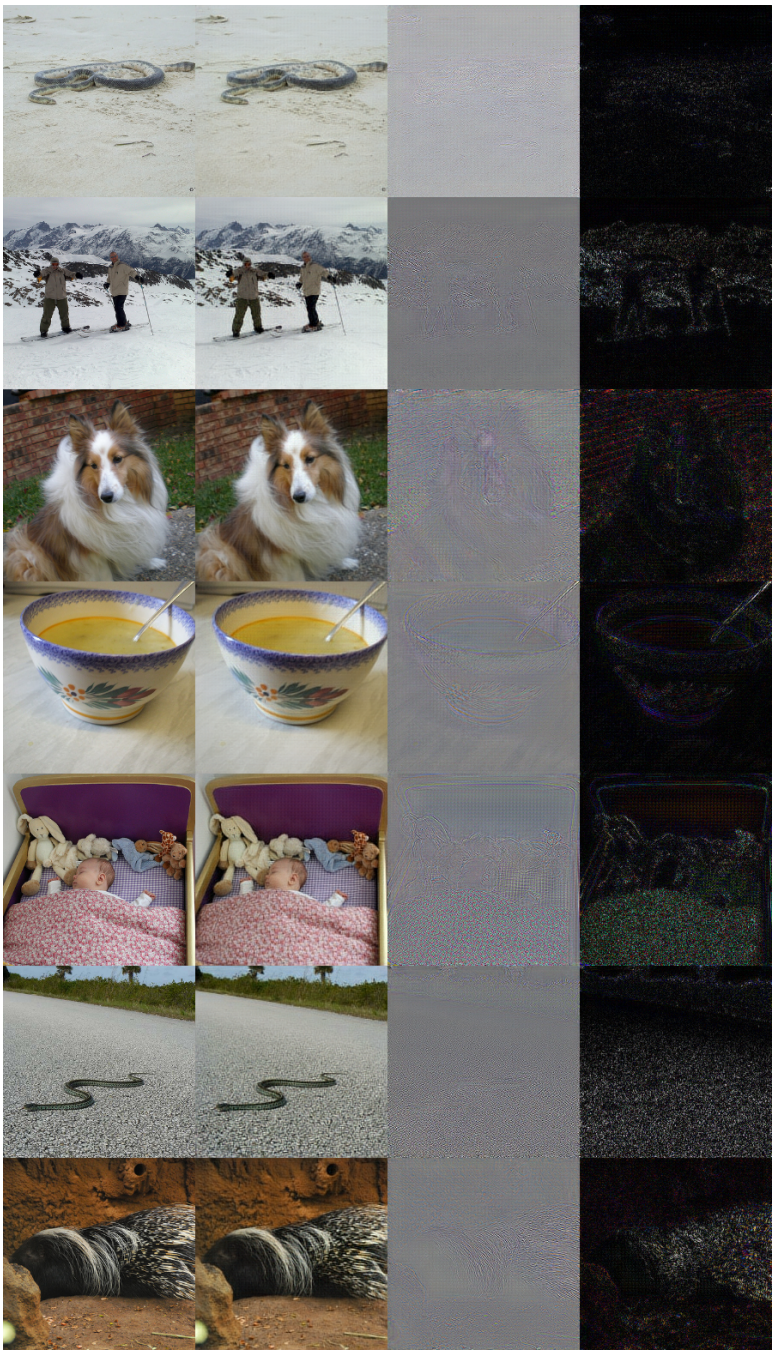

Figure 11: reconstructions of the first alexnet layer using Conv-Deconv model

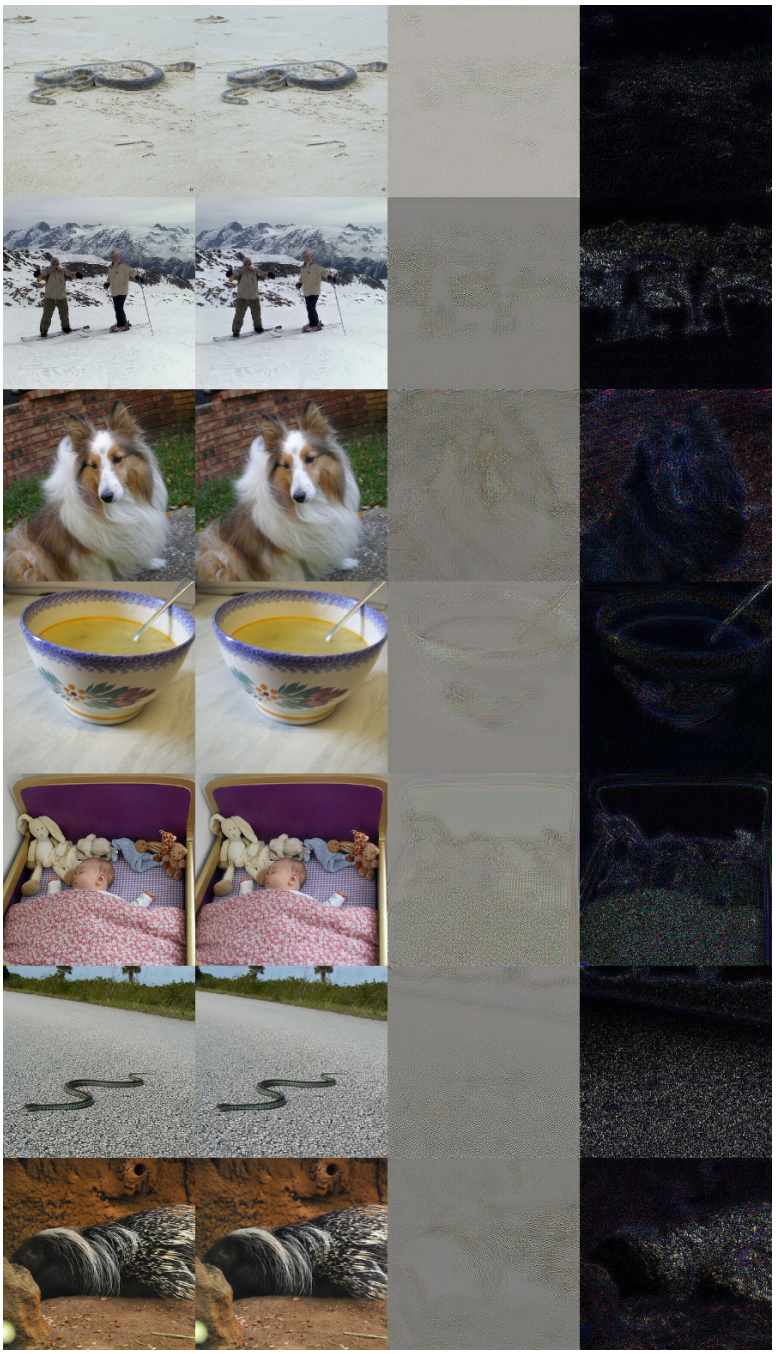

Figure 12: reconstructions of the first alexnet layer using Deconv-Conv model

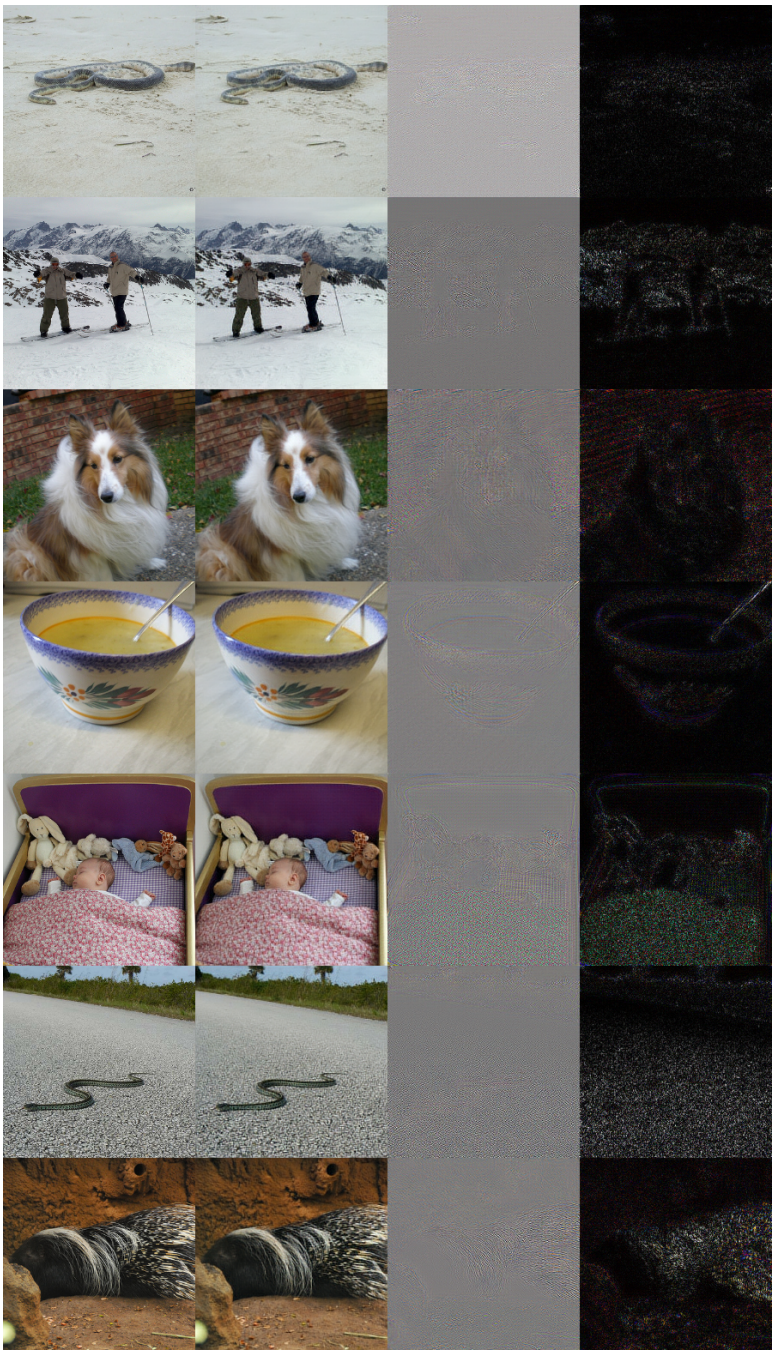

Figure 13: reconstructions of the first alexnet layer using Deconv-Deconv model

## 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