Sparse Variational Autoencoder

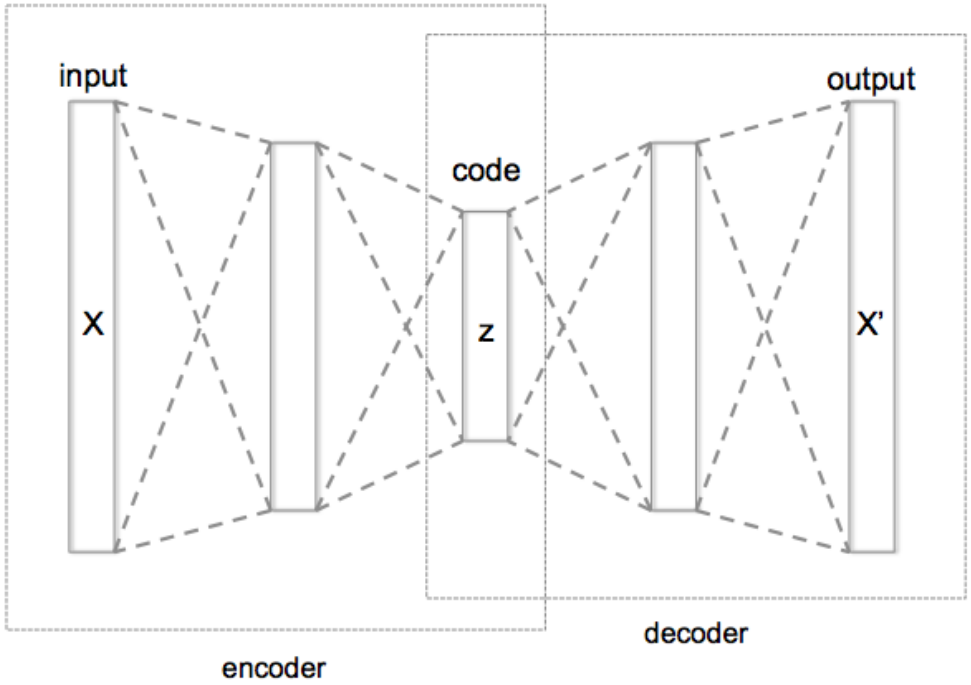
Ryan Pyle

May 5, 2019

Outline

- Auto-encoders
- Sparsity and Sparse auto-encoders
- Variational Auto-encoders
- A Spare VAE
- Results

Auto-encoders

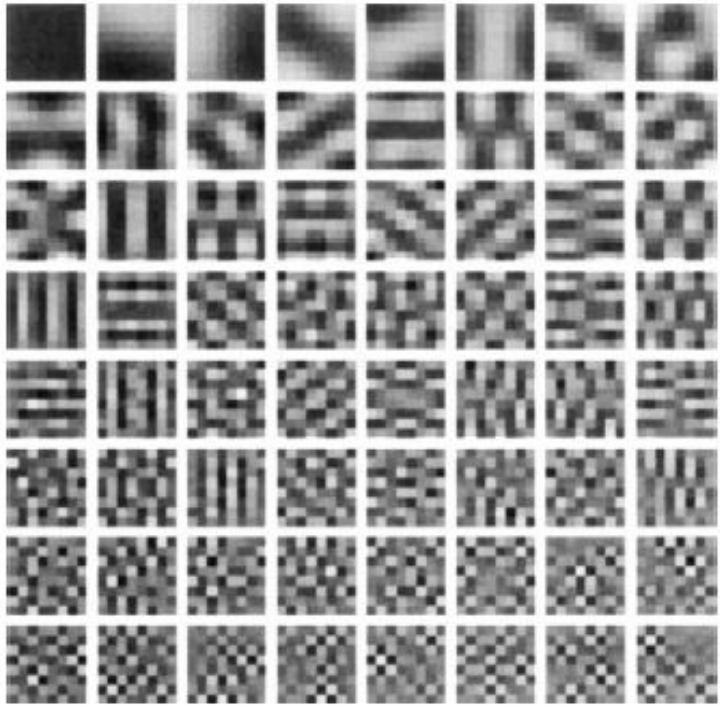


- Input X
- Goal to reproduce $X' = X$
- Trained by standard backpropogation through ANN layers
- all information about X' must be passed through lower dimension layer z

Sparsity

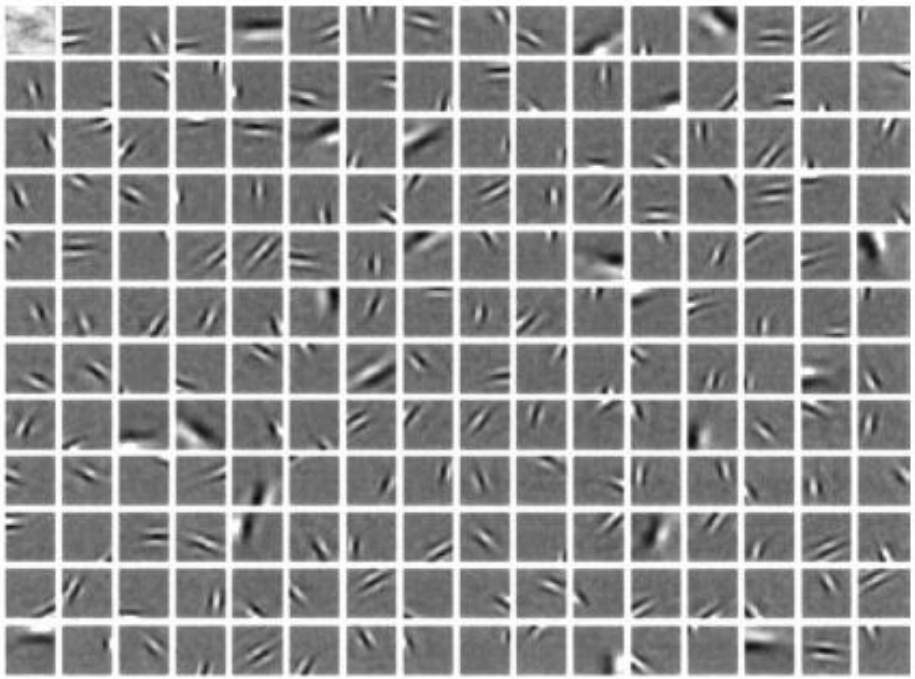
- Sparsity is the property of only having a limited number of active units
- Tradeoff between capacity and ease learning/storing representations
  - Local code - one item per unit
  - Dense code - 50% unit activity, code depends on whole population activity
- Biological constraints on energy may encourage sparse codes

Sparsity - Olhausen and Fields

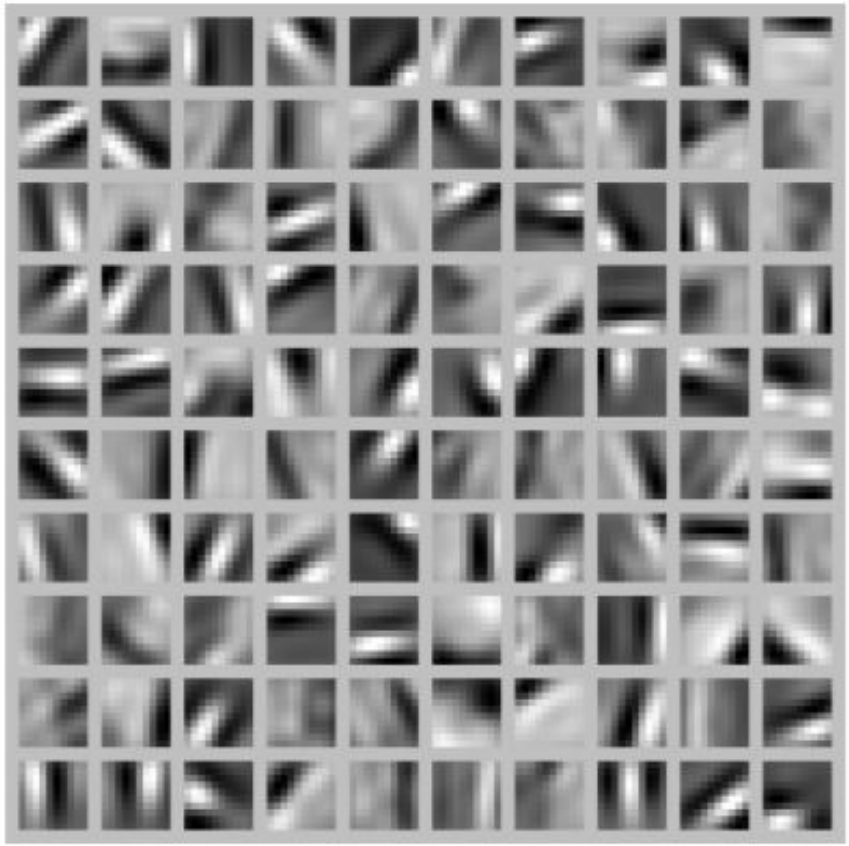


- Visual receptive fields in mammalian primary visual cortex are localized, oriented, and bandpass.
- Standard PCA code for visual images is not

Sparsity - Olhausen and Fields



- Adding a sparsity term to loss function and optimizing gave much more biologically-realistic outputs
- Resulting basis set is overcomplete, but only a few are active at any one time



- Sparsity constraint can be added to auto-encoder by adding to the loss function
- Frequently, just L1 norm over latent variables z

VAE

Variational auto-encoders (VAEs) are a powerful class of machine learning tool for grouping and producing data. The main idea of a VAE is to use bayesian inference to model the underlying distribution of the data, so that new samples can be generated by drawing from that underlying distribution.

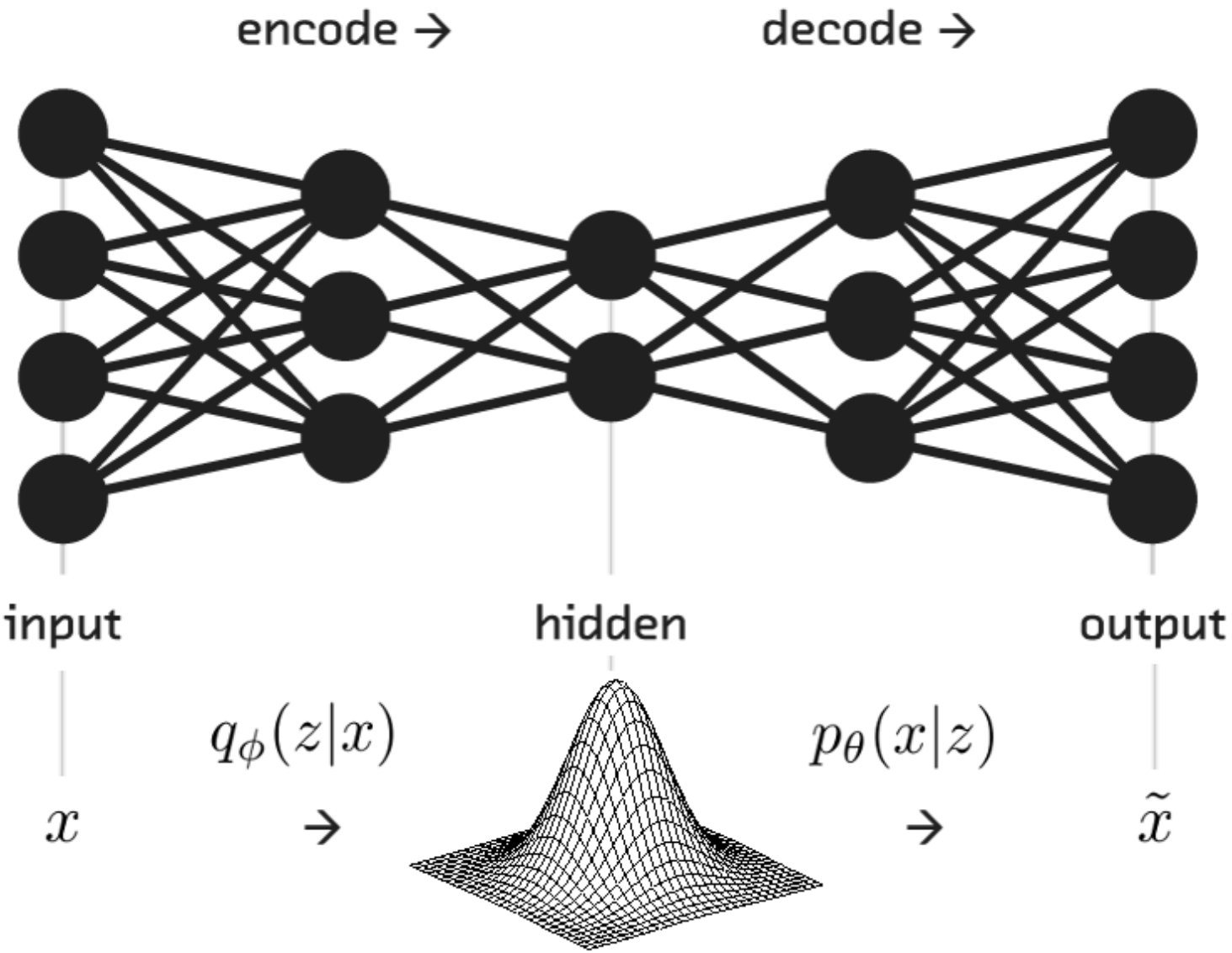
The main difference from auto-encoders is that they model the distribution of z, rather than point estimates.

Can apply to classes of problems where sampling, EM, or VB cannot apply, due to large X, intractible posteriors or integrals

VAE

- High dimensional data x
- Low dimensional latent variables z
- True underlying $P(z)$, $f(z, \theta) \rightarrow x$
- Unobserved θ, z
- Goals:
  - Encoder network $q(z|x)$
  - Decoder network $p(x|z)$
  - Coupled together, by sampling z for p from distribution generated by q
  - Can be used to generate new data

VAE


encode →
decode →
input
hidden
output
$q_\phi(z|x)$
$p_\theta(x|z)$
x
→
→
$\tilde{x}$

VAE

- Goal : maximize $P(X) = \int P(X|z,\theta)P(z)dz$
- Simplifying assumption : $P(X|z,\theta) = N(X|f(z,\theta),\sigma^2 I)$
  - this allows for gradient descent
- Simplifying assumption : $P(z) = N(0, I)$
  - Not true, but the mapping to true $P(z)$ can be learned as part of decoder
- The standard way of computing this via sampling would have an intractibly long runtime
- New function $Q(z|X)$ allows us to sample only z that are likely to produce X like our output
- Need to relate $Q(z|X)$ back to our original $P(x)$, where z is drawn from $N(0, I)$ rather than Q

Taking the KL divergence between arbitrary Q and $P(z|X)$ yields

$$KL(Q(z)||P(z|X)) = E_Q[\log(Q(z)) - \log(P(z|X))]$$

applying Bayes rule to $P(z|X)$

$$KL(Q(z)||P(z|X)) = E_Q[\log(Q(z)) - \log(P(X|z)) - \log(P(z))] + \log(P(X))$$

where we pulled out the term that didn't depend on z.

Rearranging terms,

$$log(P(X)) - KL(Q(z)||P(z|X)) = E_Q[\log(Q(z)) - \log(P(X|z)) - \log(P(z))$$

$$= E_Q[\log(P(X|z))] - KL(Q(z)||P(z))$$

Note that since $Q(z)$ was arbitrary, we can replace it with $Q(z|X)$ without issue. This forms the central equation of the VAE:

$$log(P(X)) - KL(Q(z|X)||P(z|X)) = E_Q[\log(P(X|z))] - KL(Q(z|X)||P(z)$$

Note the similarities to the evidence lower bound in EM.

VAE

$$log(P(X)) - KL(Q(z|X)||P(z|X)) = E_Q[\log(P(X|z))] - KL(Q(z|X)||P(z))$$

- LHS is what we want to find, with a regularization term keeping Q near the true $P(z|X)$.
- RHS is significantly easier to work with, can be solved by standard gradient descent

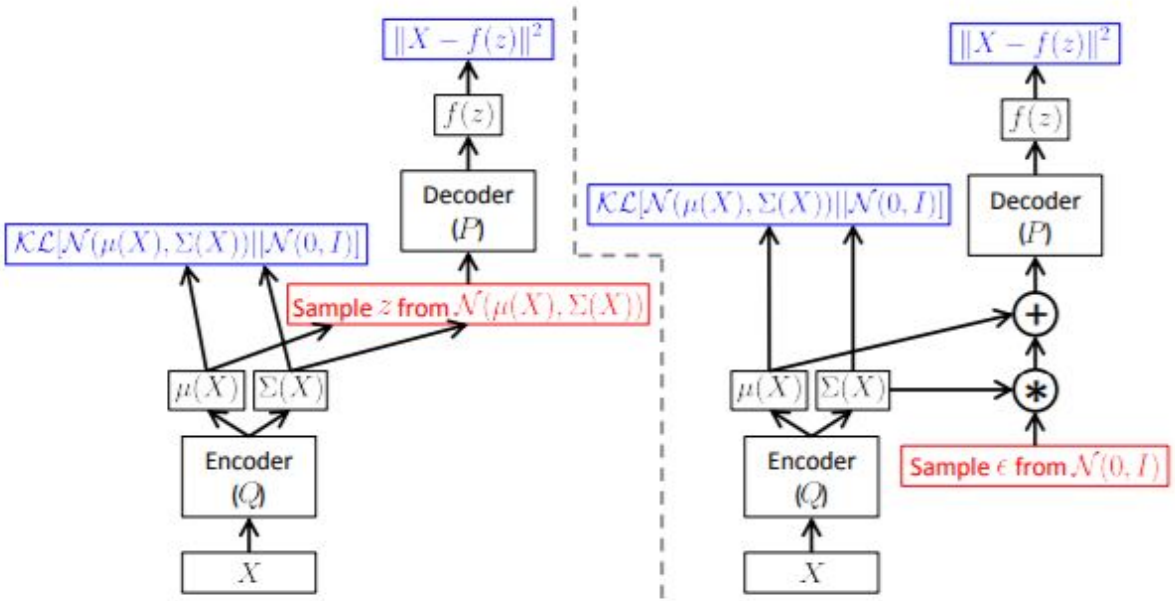
- Need to make an assumption about Q to proceed further
- Usually, take $Q(z|X) = N(z|u(X,\theta), \Sigma(X,\theta))$
- This allows for computing $KL(Q||P)$ in closed form

$$KL(Q(z|X)||P(z)) = \frac{1}{2}[tr(\Sigma) + u'u - k - \log(det(\Sigma))]$$

- Remaining term $E_Q[\log(P(X||z))]$ would be expensive to estimate at each step.
- However, in SGD we are actually interested in the gradient of $E_X[E_Q[log(P(X))] - KL(Q(z|X)||P(z|X))]$
- The gradient can be moved inside the expectation, so we can just take a single X and z from Q to get $log(P(X)) - KL(Q(z|X)||P(z|X))$
- This should converge to the above over a minibatch of X values

VAE

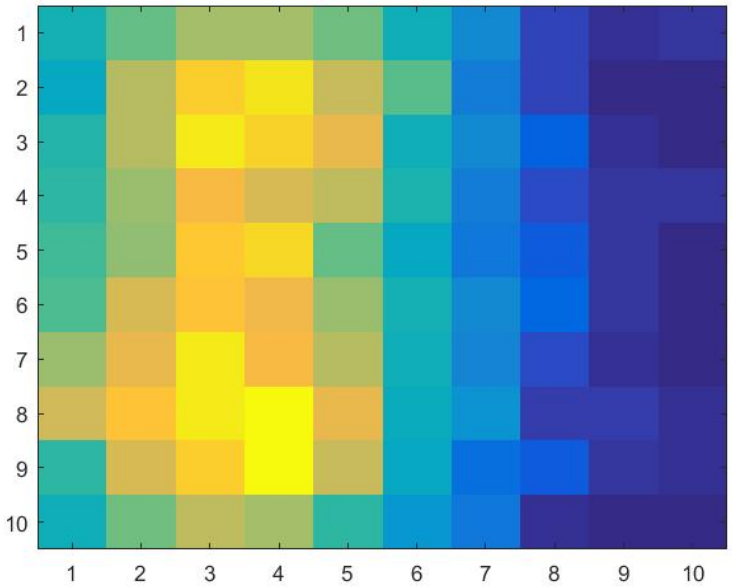
- z is sampled from $Q(z|X) = N(z|u(X, \theta), \Sigma(X, \theta))$
- This is a stochastic operation with no gradient
- The reparameterization trick allows us to get around this - sample a random ϵ, and use it to build our z

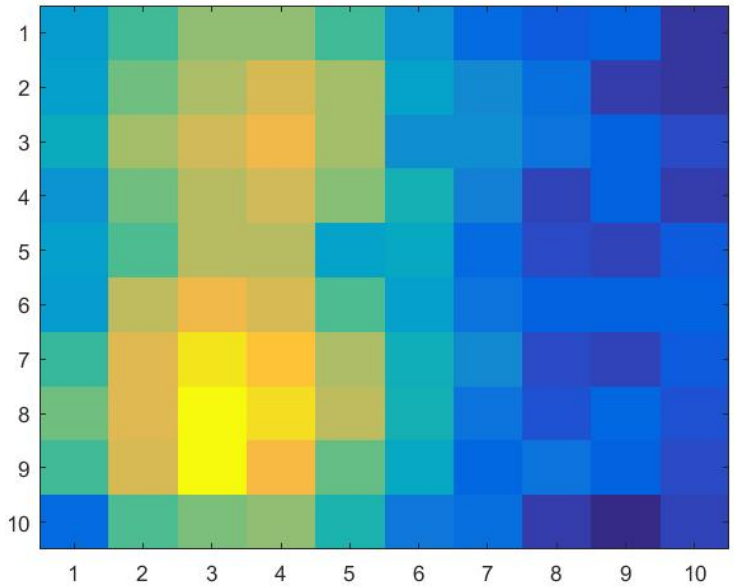


VAE example

I implemented a simple VAE in matlab: X inputs are 10x10 images made a weighted sum of two templates plus noise. Encoder and decoder networks were each 1 layer deep, with a latent dimension of 2.

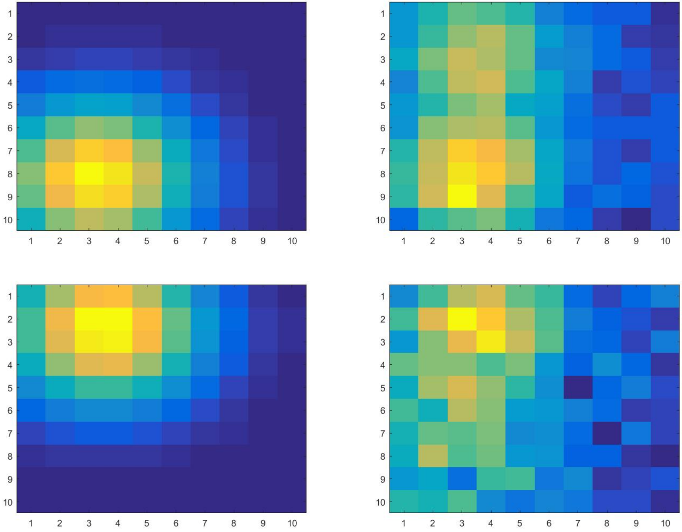
Input vs Reconstruction:





VAE example

Templates vs Latent Representations:



A Sparse VAE

- Sparsity could give a more sensible hidden layer
- For VAE, sparsity analogue is amount of latent variables z active at once
- One possibility is to give $p(z)$ a 'sparse' prior
- Issue : VAE is tractible because of simplifications made assuming $p(z) \sim N(0, 1)$

A Sparse VAE

- Laplace is often considered a sparse prior
- Setting $p(z) \sim Laplace$ renders our $KL(Q(z|X)||P(z))$ unsolveable analytically
- but $KL(Q(z|X)||P(z)) = E_Q[log(\frac{Q(z|X)}{P(z)})]$
- We already dealt with a $E_Q[\log(P(X|z))]$ term earlier, and the same strategy can apply

$$E_X[E_{\epsilon\ N(0,I)}[log(P(X|z) = u(X,\theta)+\Sigma^{\frac{1}{2}}(X,\theta)\epsilon - \log(\frac{Q(z = u(X,\theta) + \Sigma^{\frac{1}{2}}(X,\theta)\epsilon|X)}{P(z = u(X,\theta) + \Sigma^{\frac{1}{2}}(X,\theta)\epsilon)})]]$$

A Sparse VAE - 2

An alternative approach:

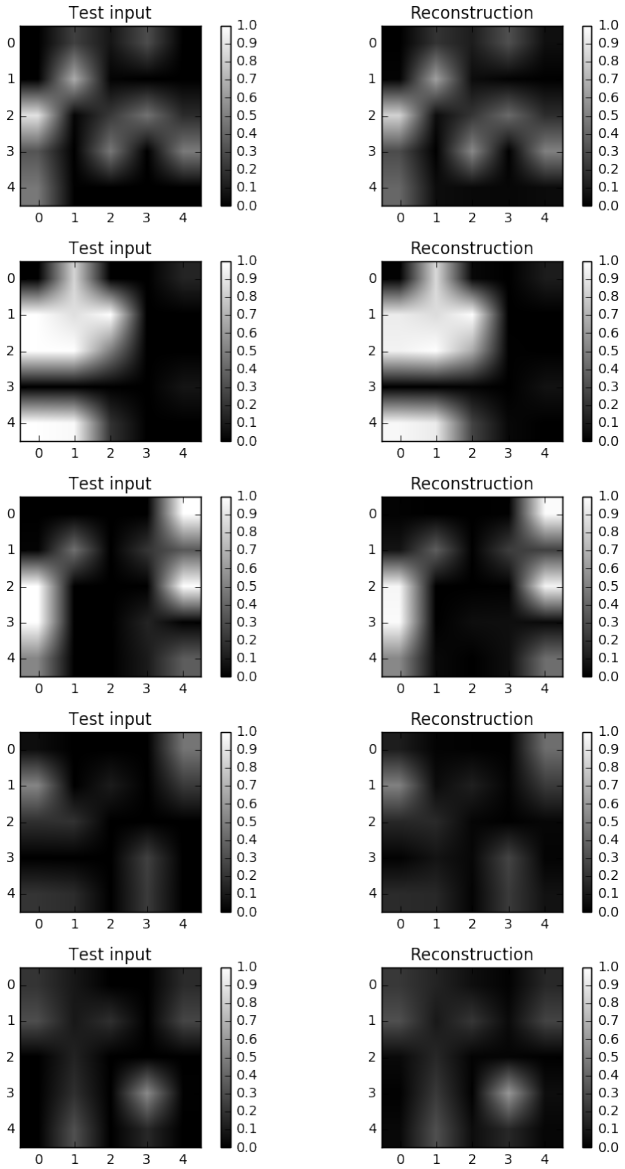
- In VAE, in order to sample we wish to have P(Z) be a known distribution eg N(0,1)
- Q(Z|X) is constrained to be 'like' P(Z) in terms of KL divergence
  - This is meant to enforce Q(Z|X) $=$ P(Z)
- This is only possible if both the distributions are additive
- This means we should use an additive, or Levy alpha-stable distribution
- Cauchy distribution is the only other standard Levy alpha-stable distribution
- Note both $P(z)$ and $Q(z|X)$ are changed to be Cauchy in this case

Results

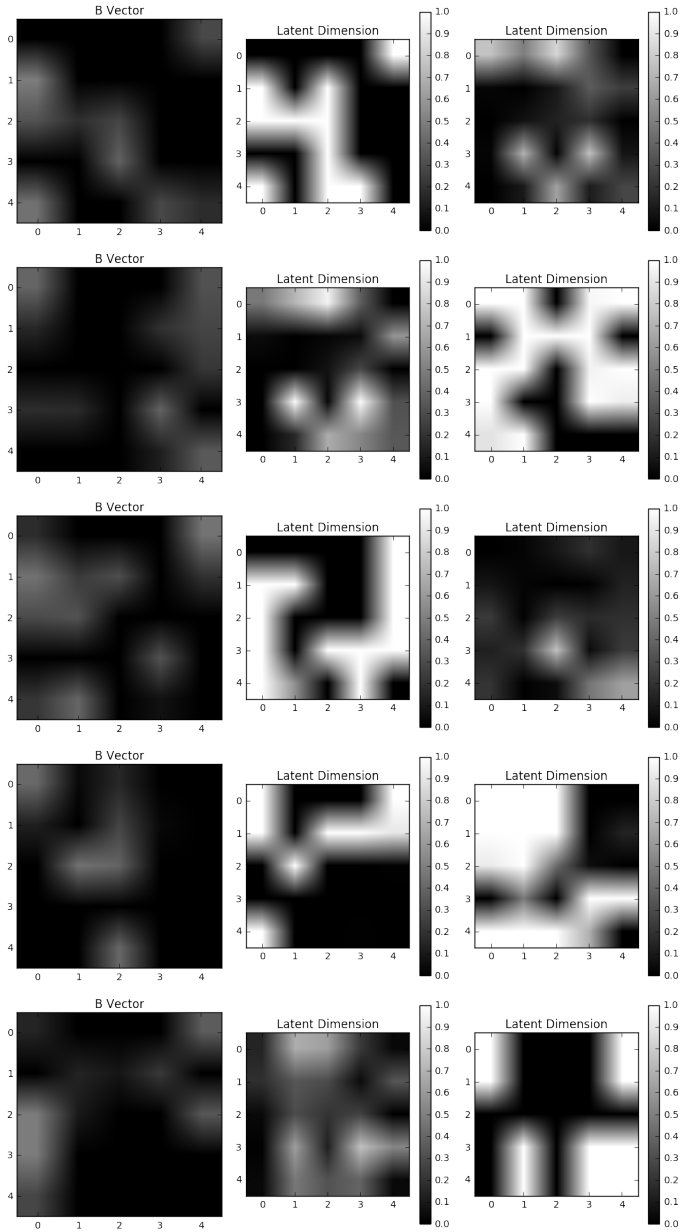
I ran all three algorithms (VAE, laplace prior VAE and cauchy prior/cauchy noise VAE) based off of a tensorflow VAE implementation.

- Task 1 : Synthetic image data (5x5)
- Task 2 : MNIST digits
- Task 3 : CIFAR natural image data, changed to black and white and subsampled to lower resolution

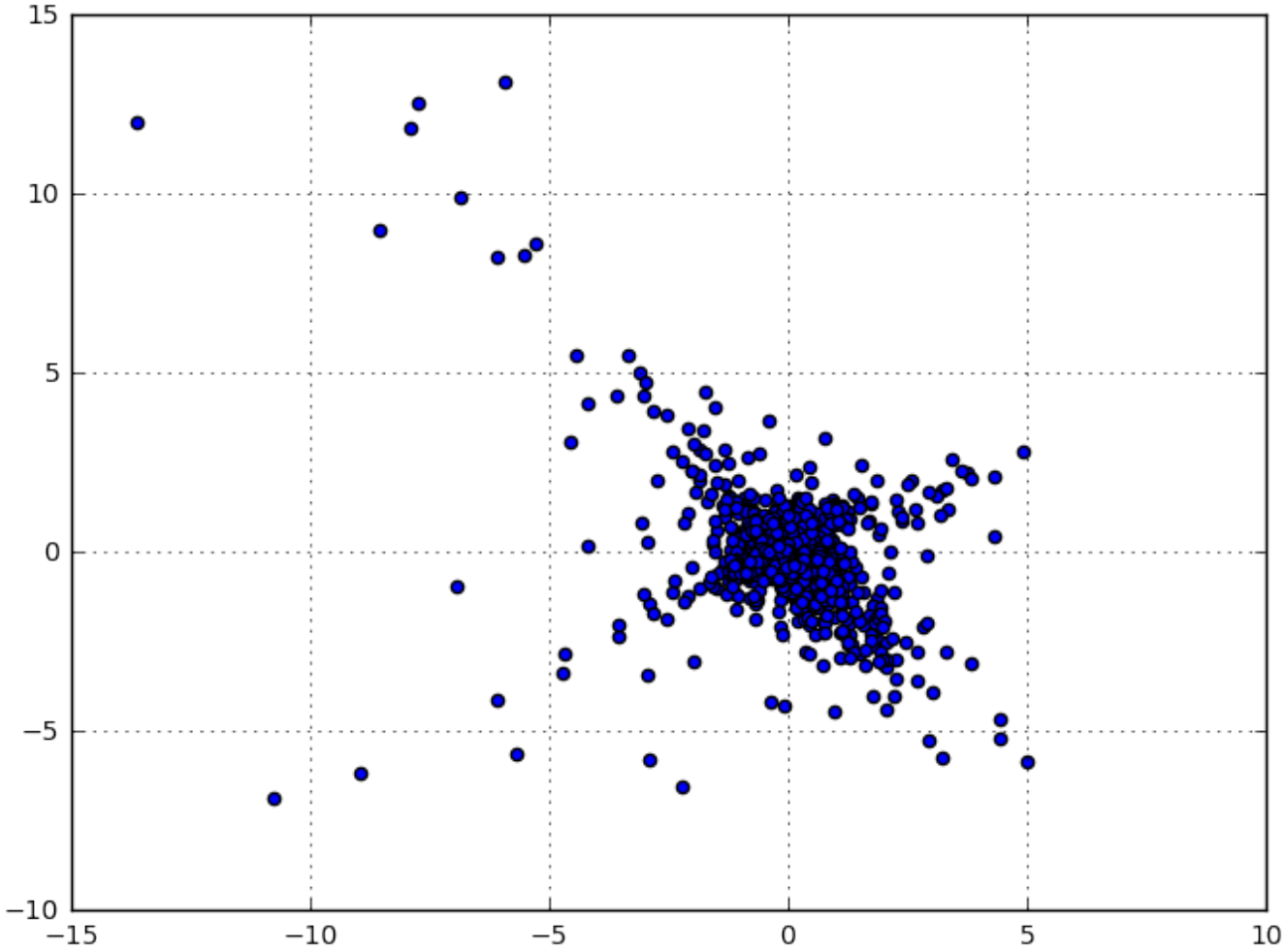
Synthetic Data - Normal VAE



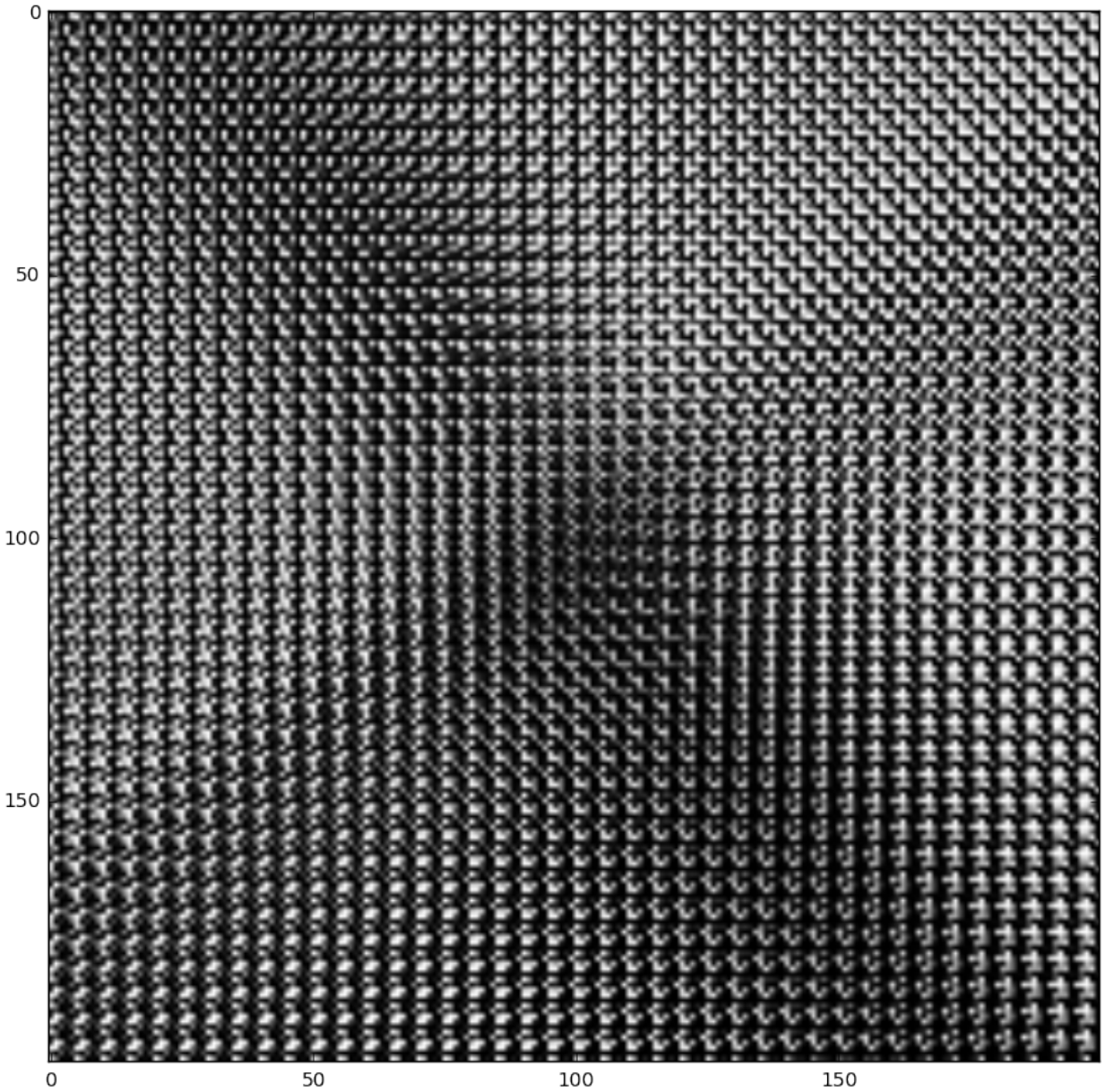
Synthetic Data - Normal VAE



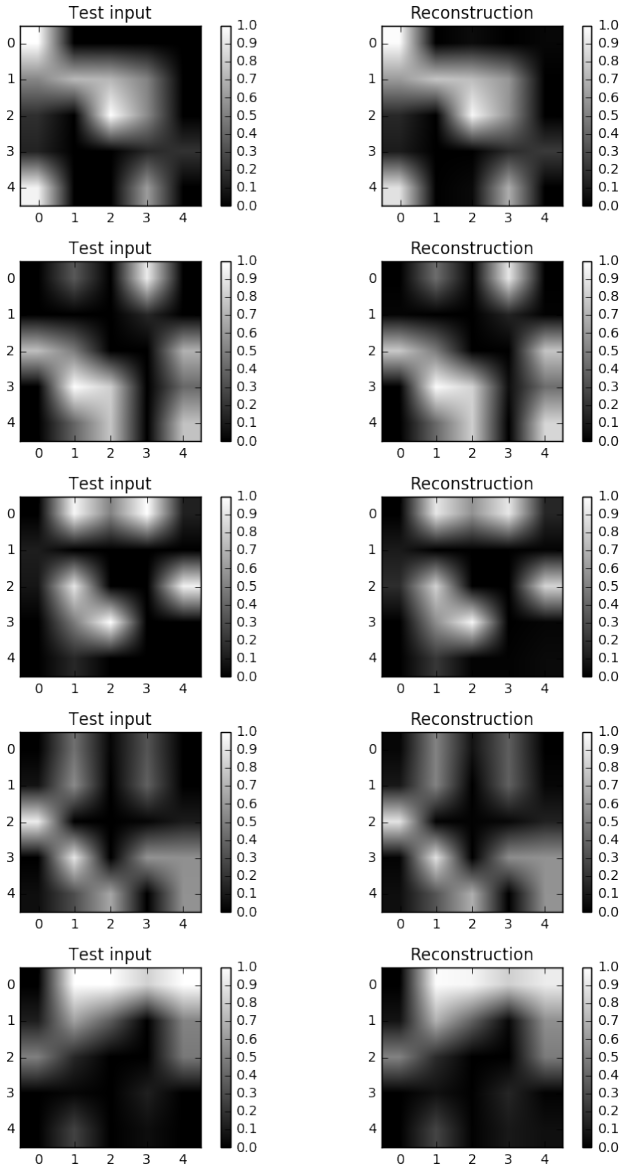
Synthetic Data - Normal VAE



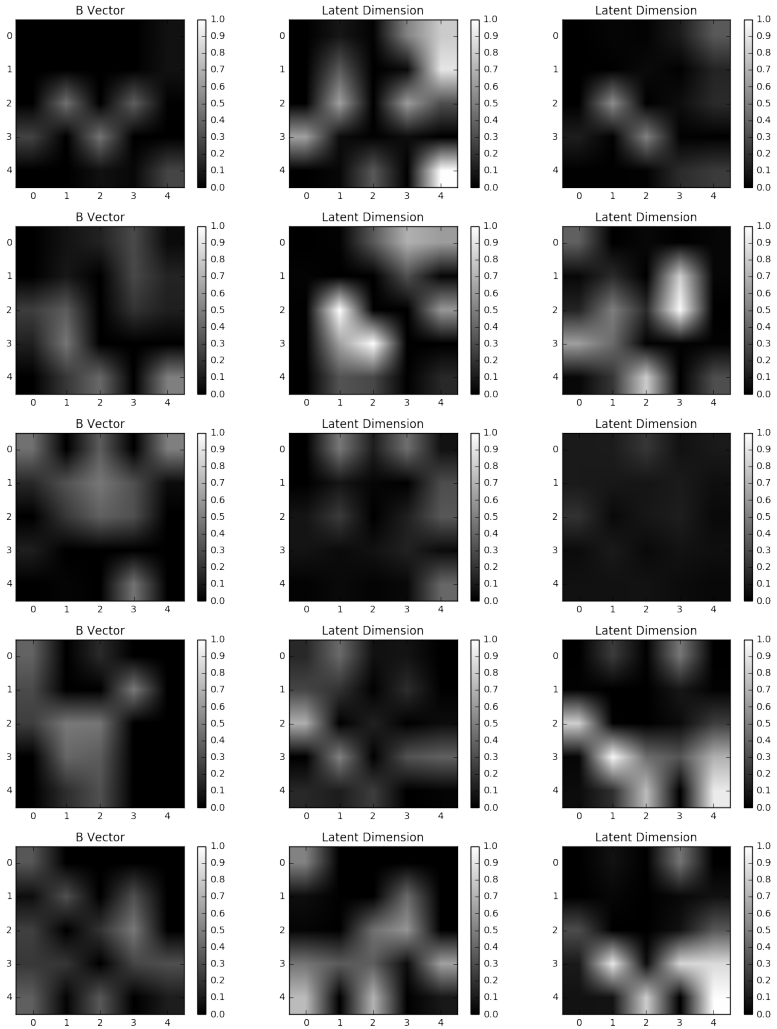
Synthetic Data - Normal VAE



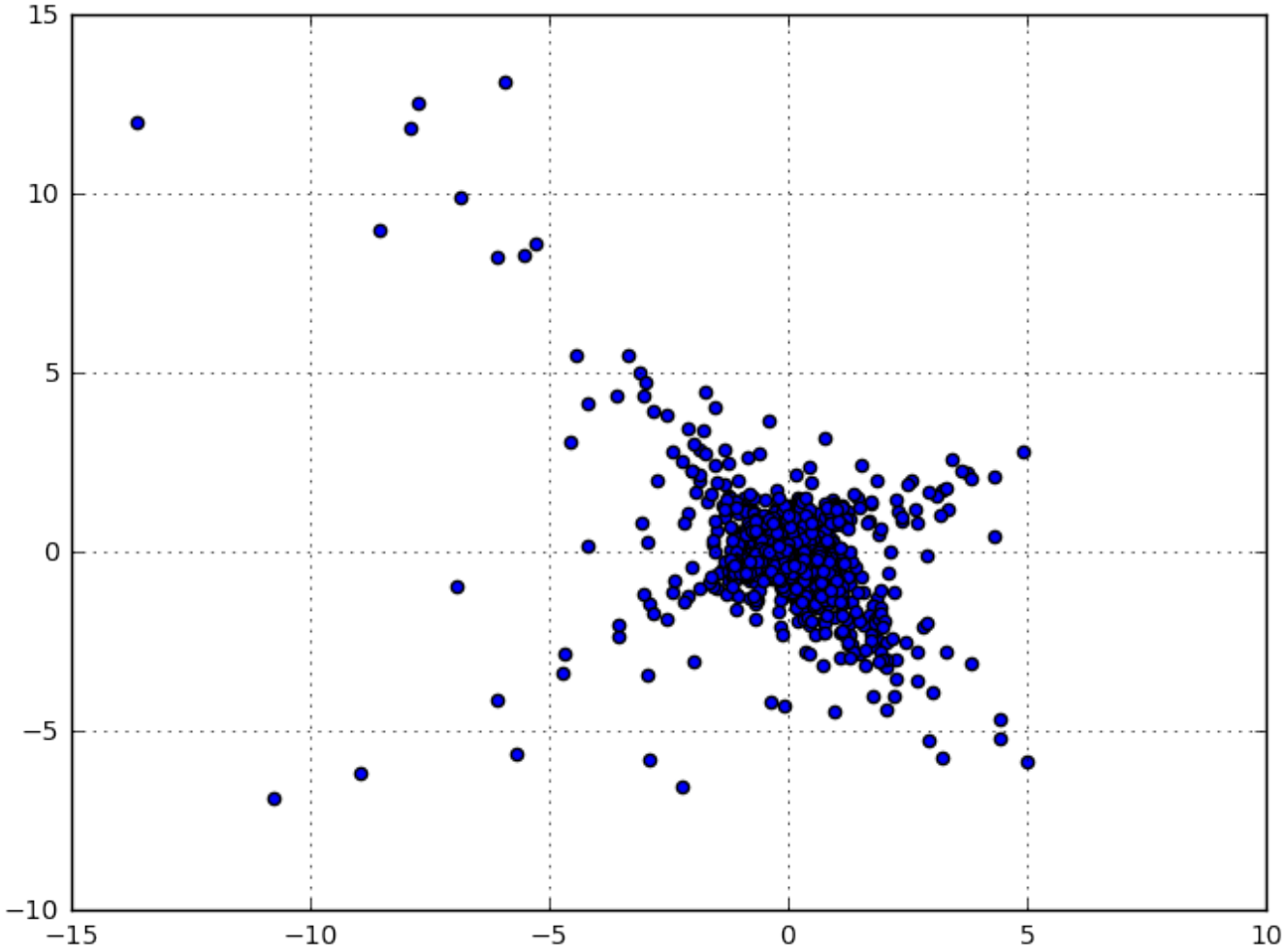
Synthetic Data - Sparse Laplace VAE



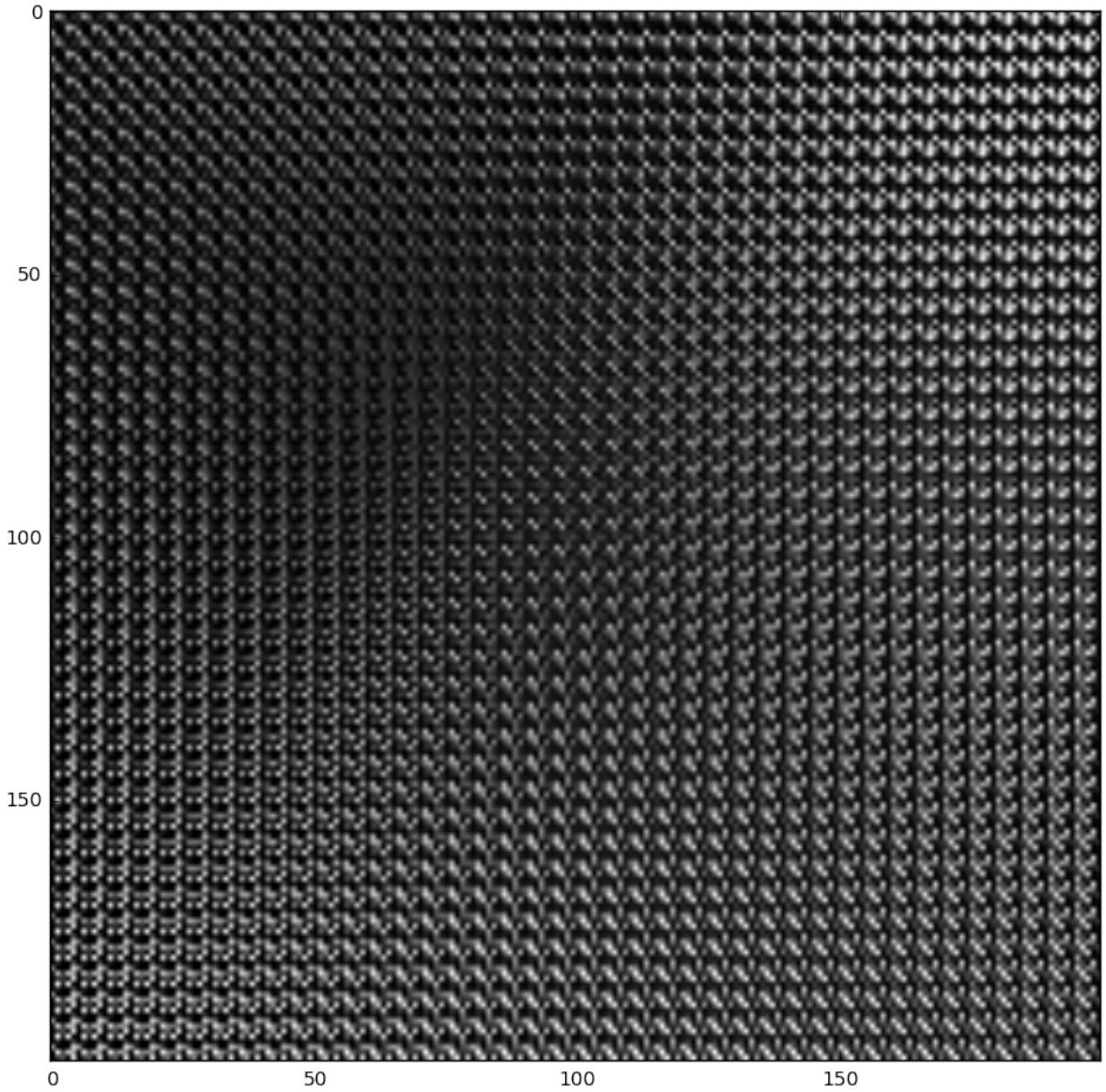
Synthetic Data - Sparse Laplace VAE



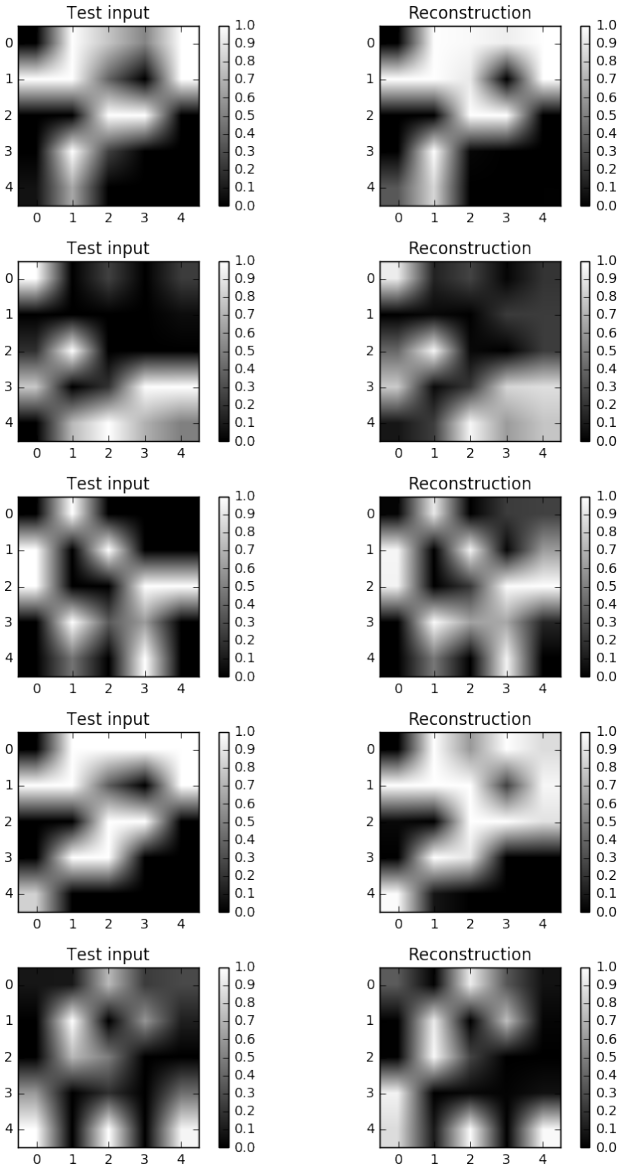
Synthetic Data - Sparse Laplace VAE



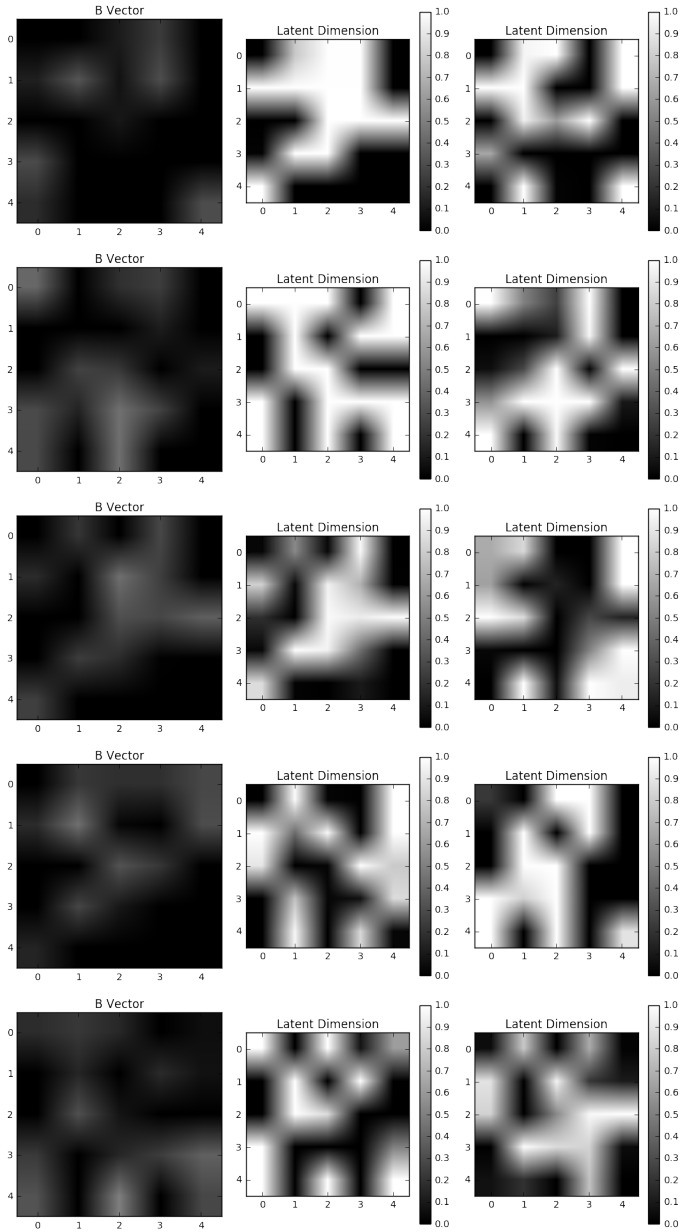
Synthetic Data - Sparse Laplace VAE



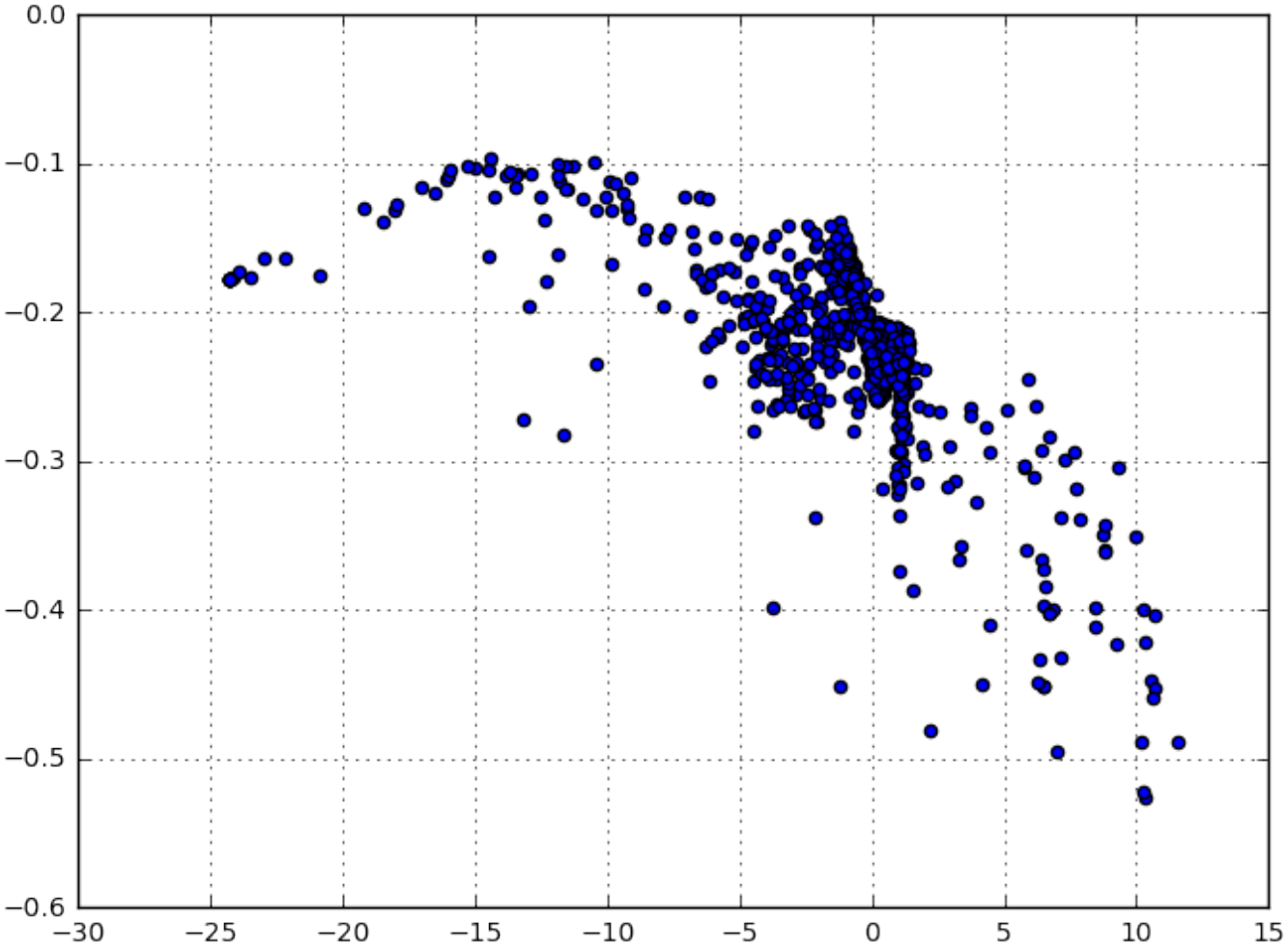
Synthetic Data - Sparse Cauchy VAE



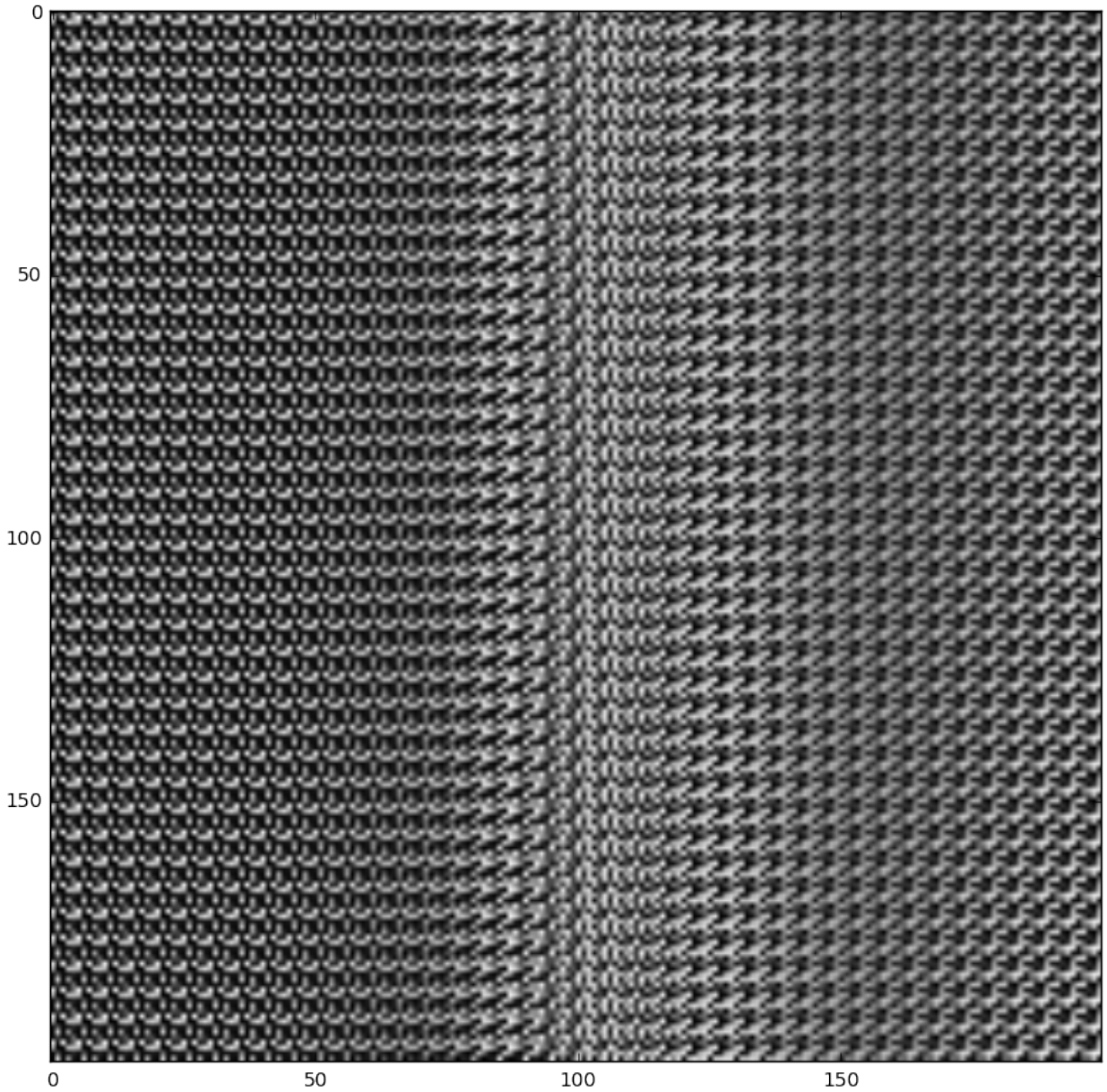
Synthetic Data - Sparse Cauchy VAE



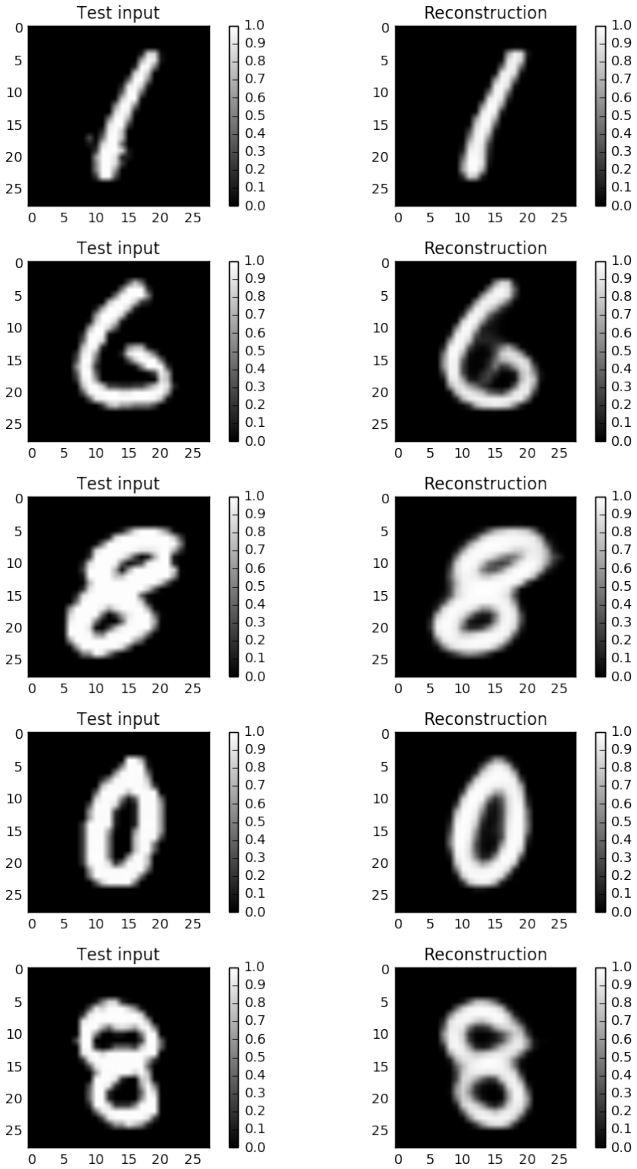
Synthetic Data - Sparse Cauchy VAE



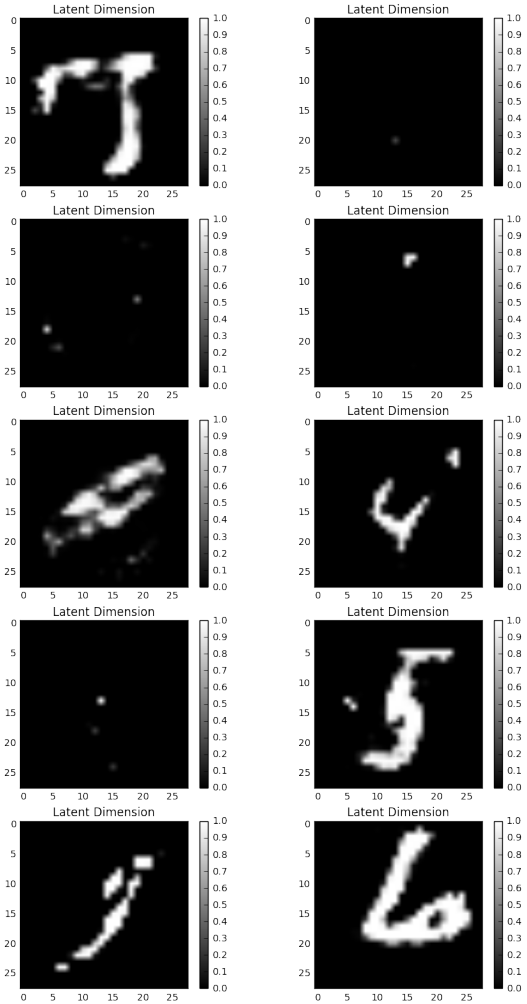
Synthetic Data - Sparse Cauchy VAE



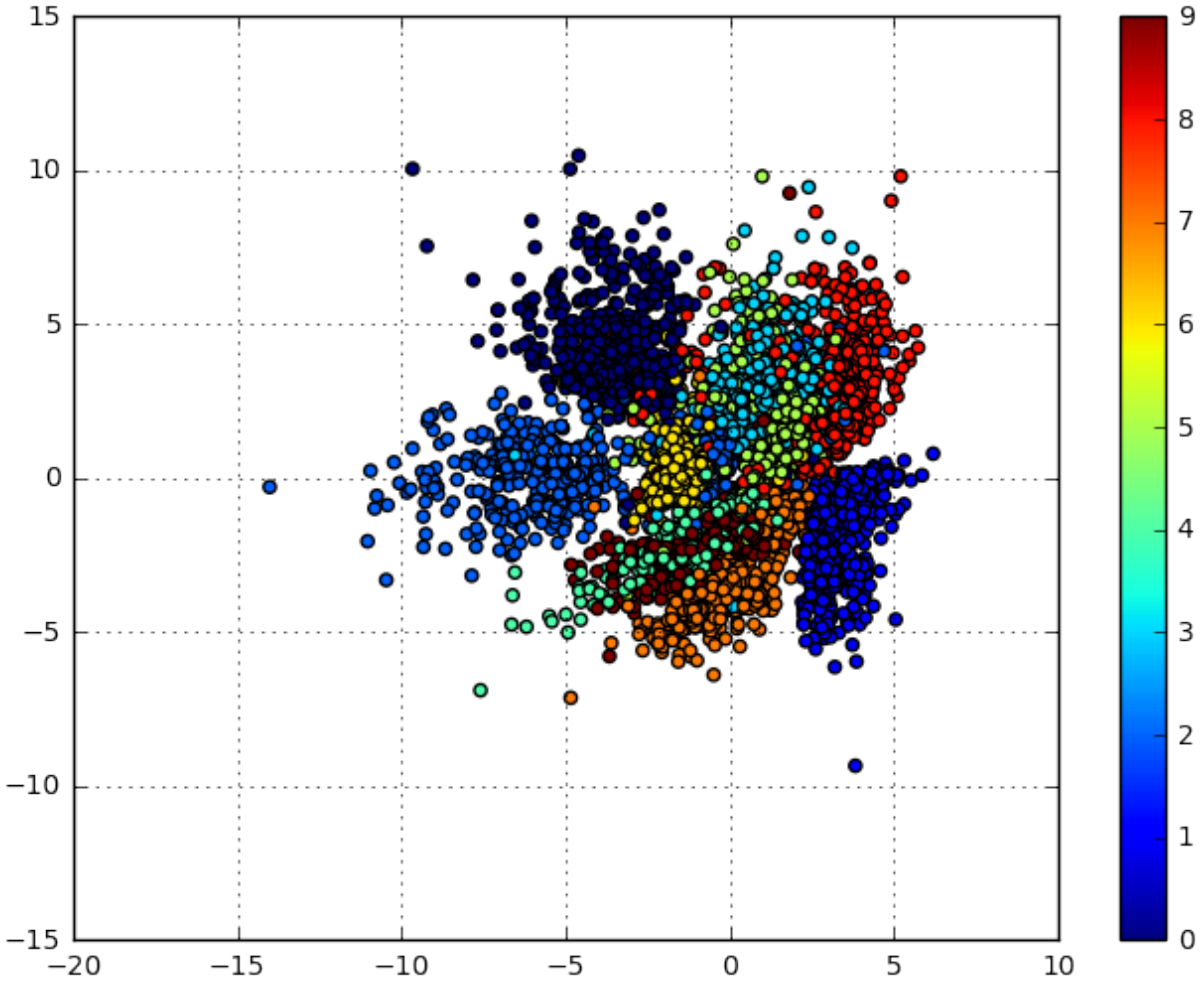
MNIST - Normal VAE



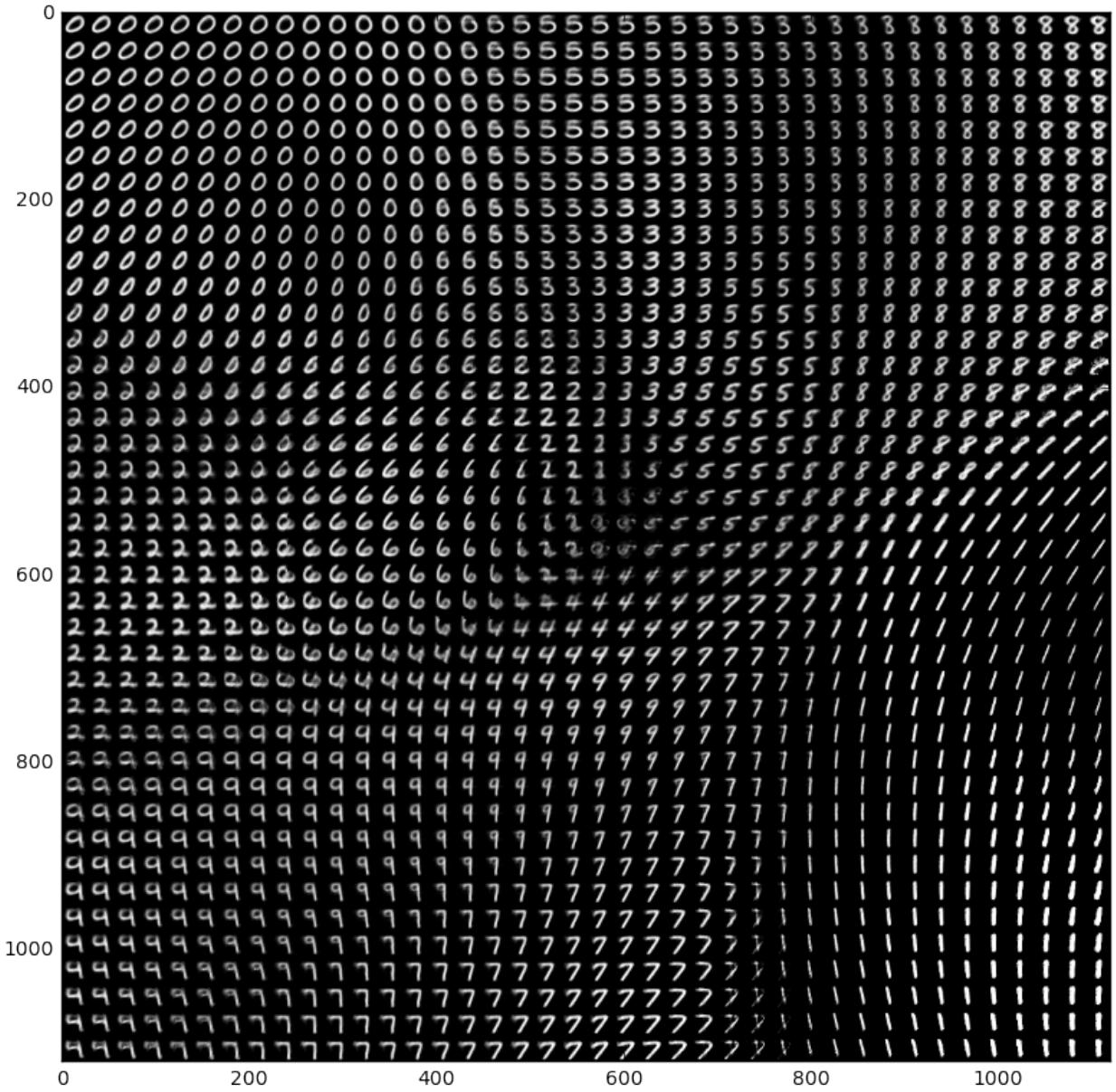
MNIST - Normal VAE



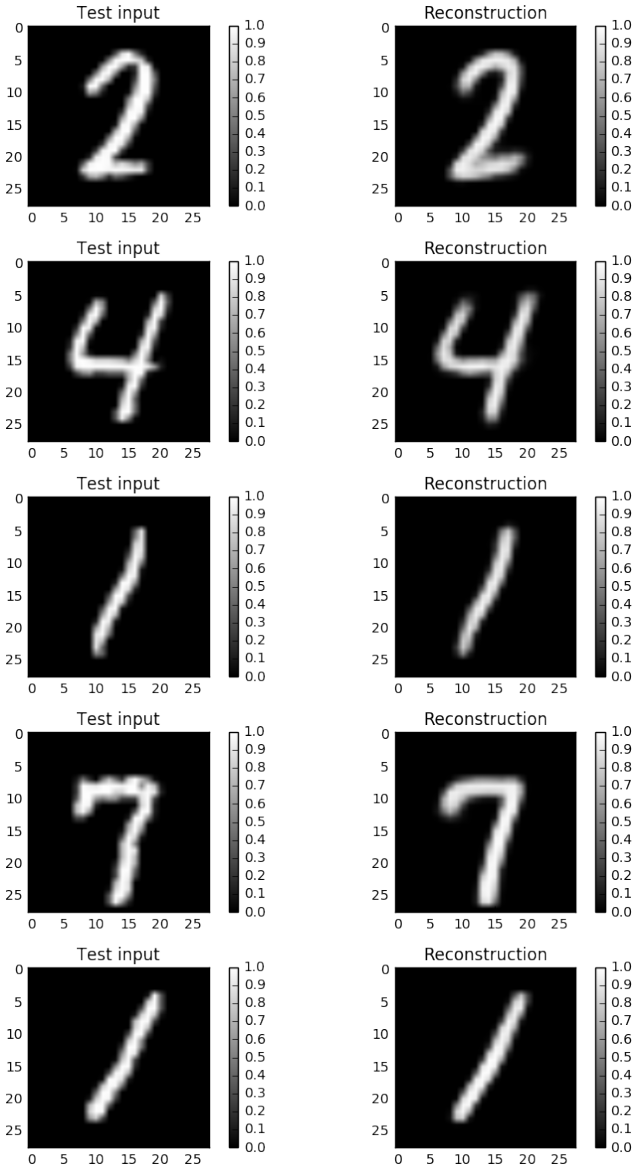
MNIST - Normal VAE



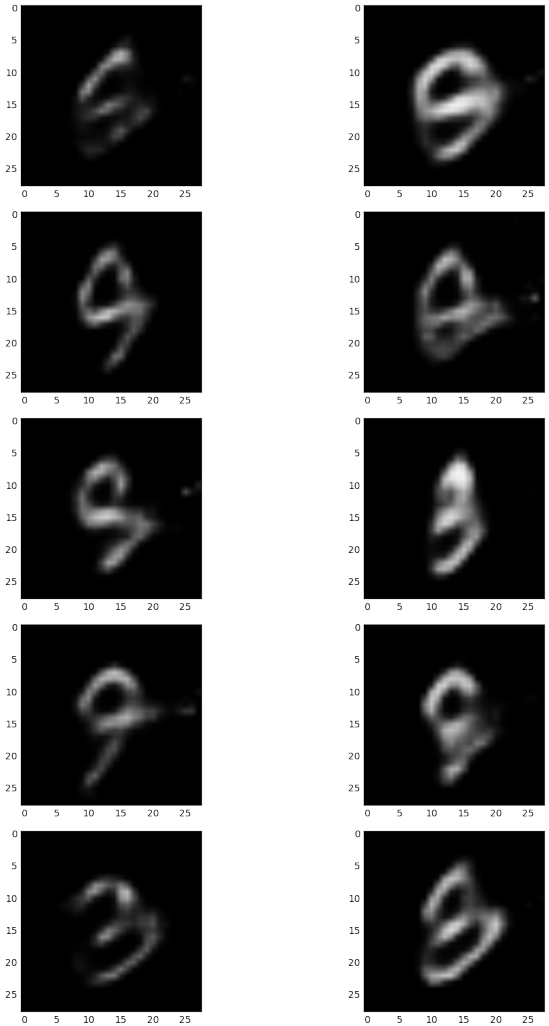
MNIST - Normal VAE



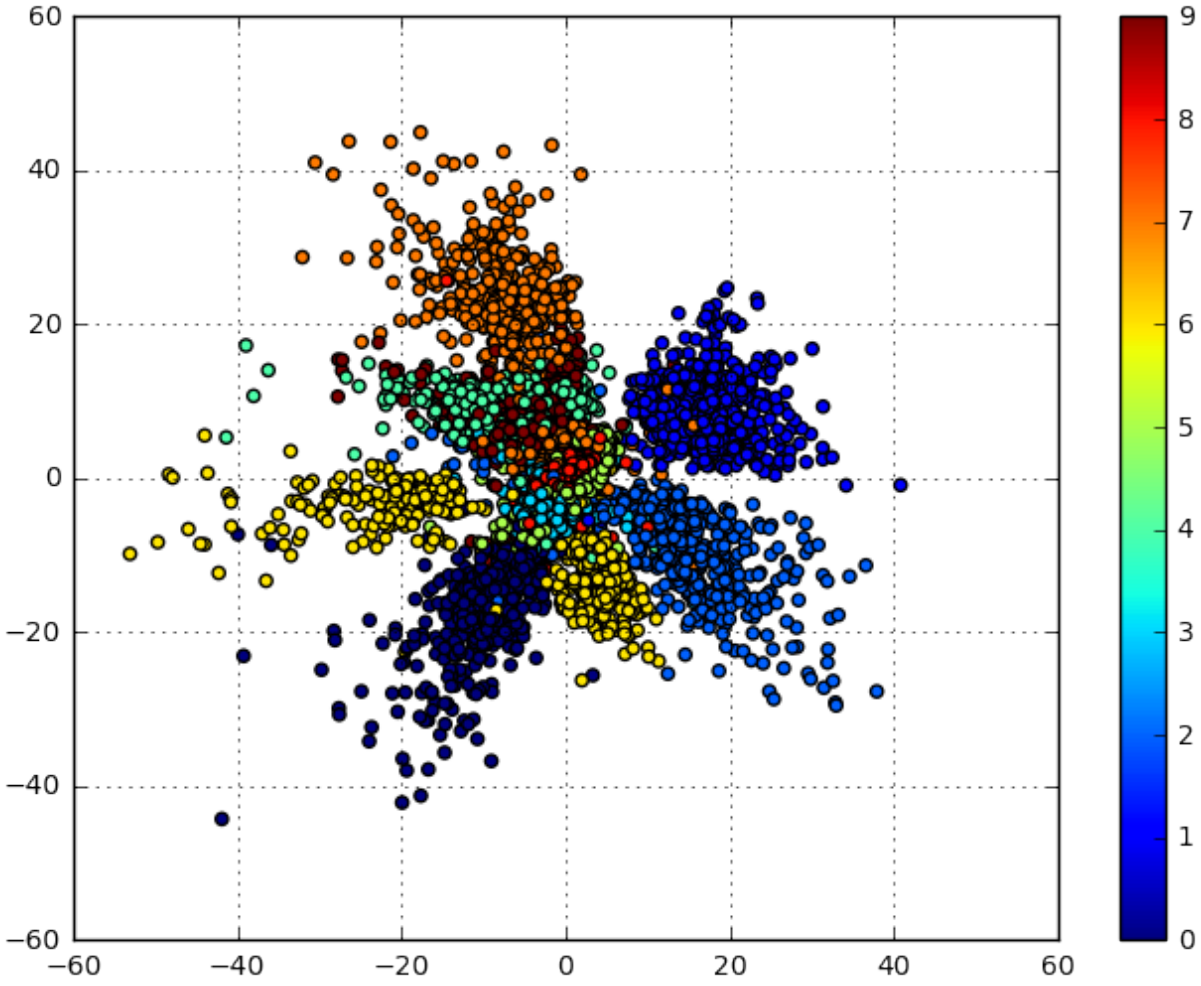
MNIST - Sparse Laplace VAE



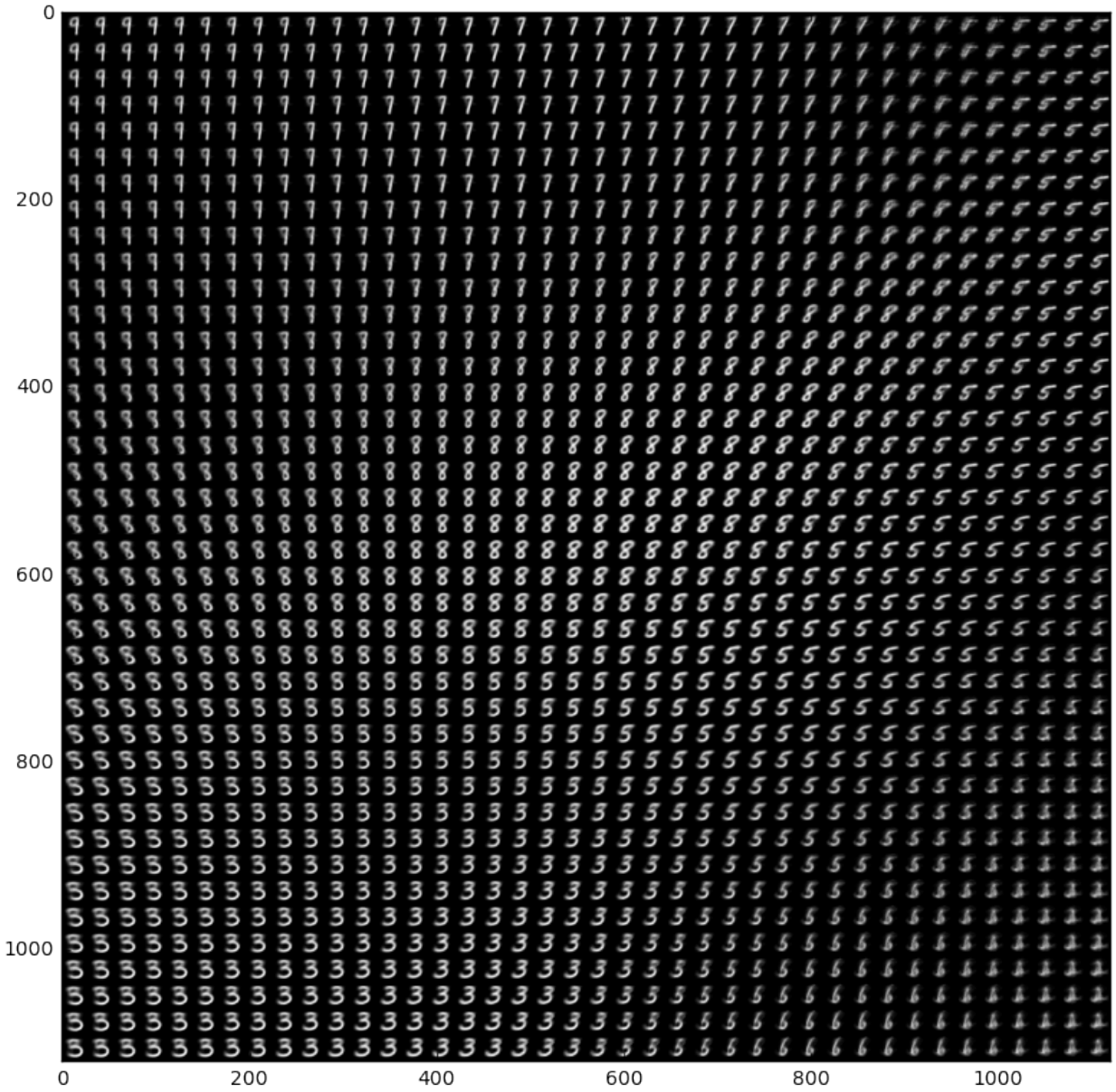
MNIST - Sparse Laplace VAE



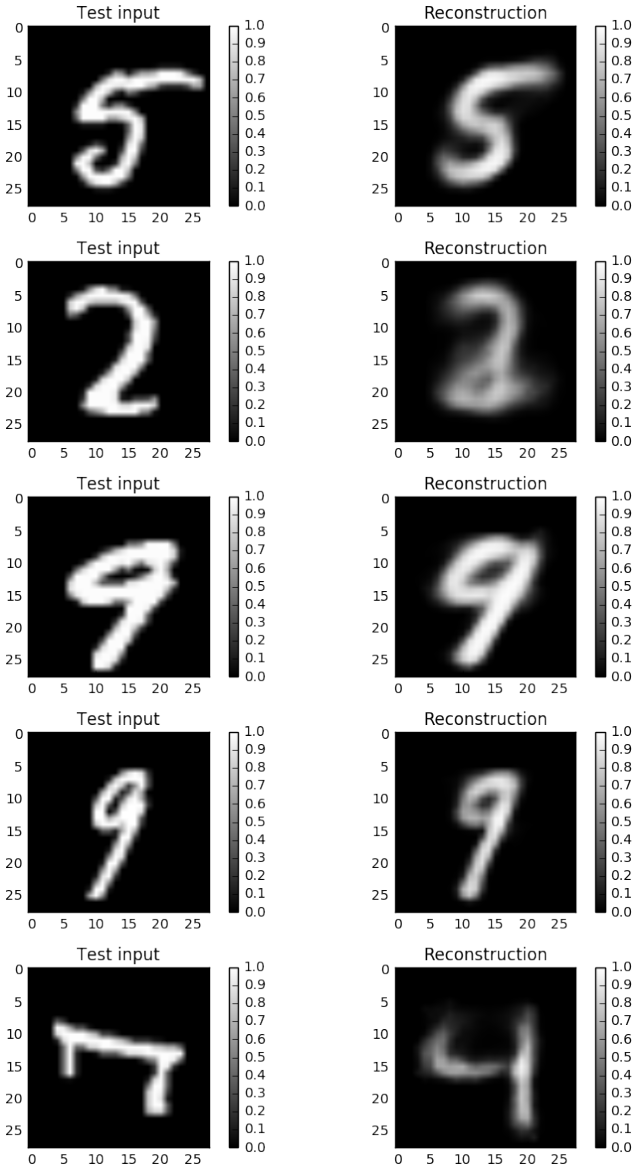
MNIST - Sparse Laplace VAE



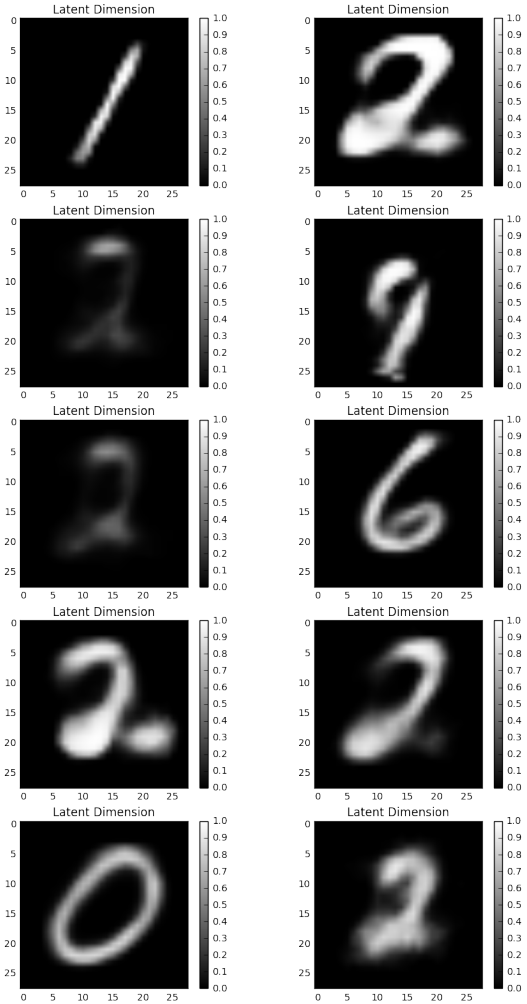
MNIST - Sparse Laplace VAE



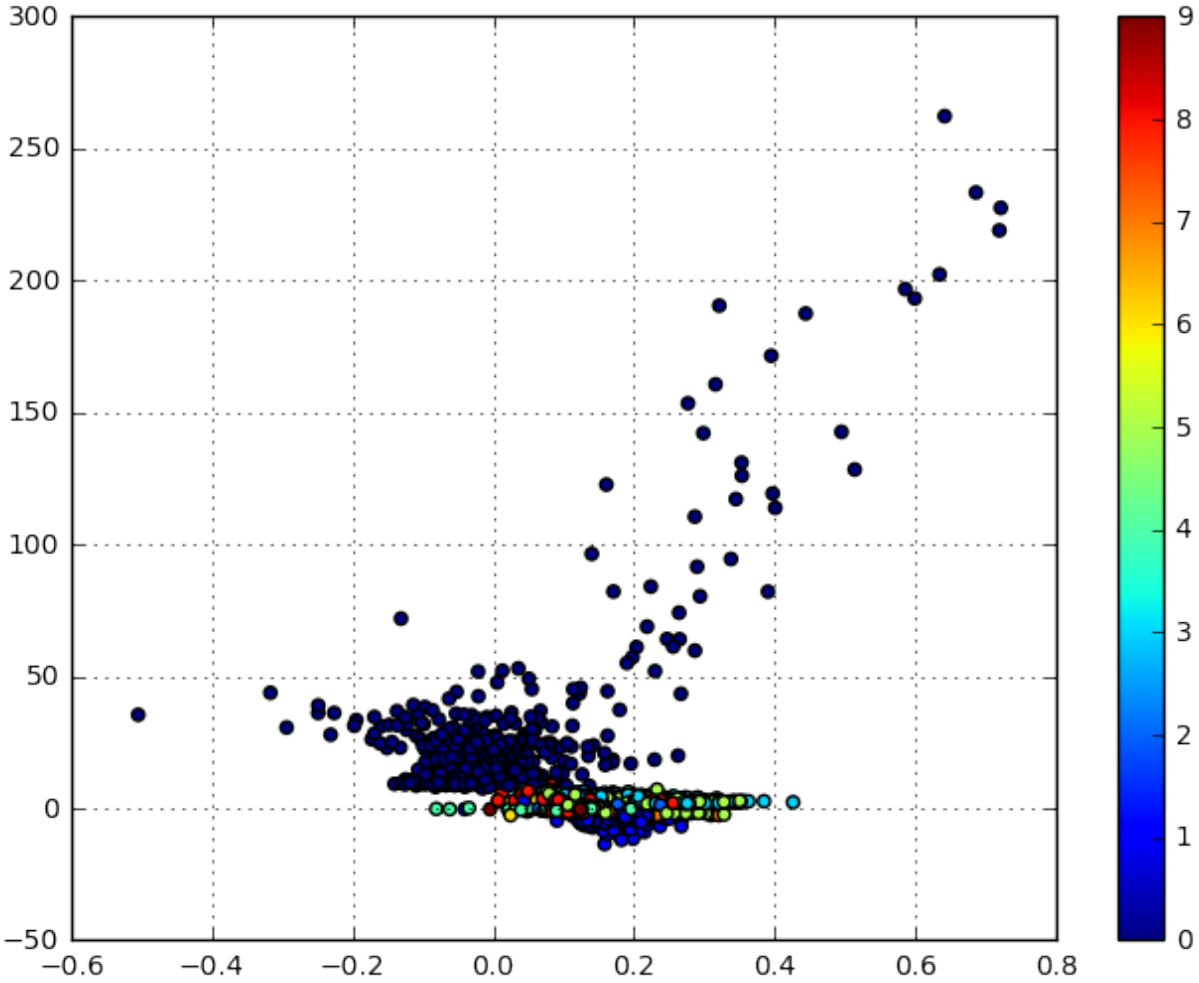
MNIST - Sparse Cauchy VAE



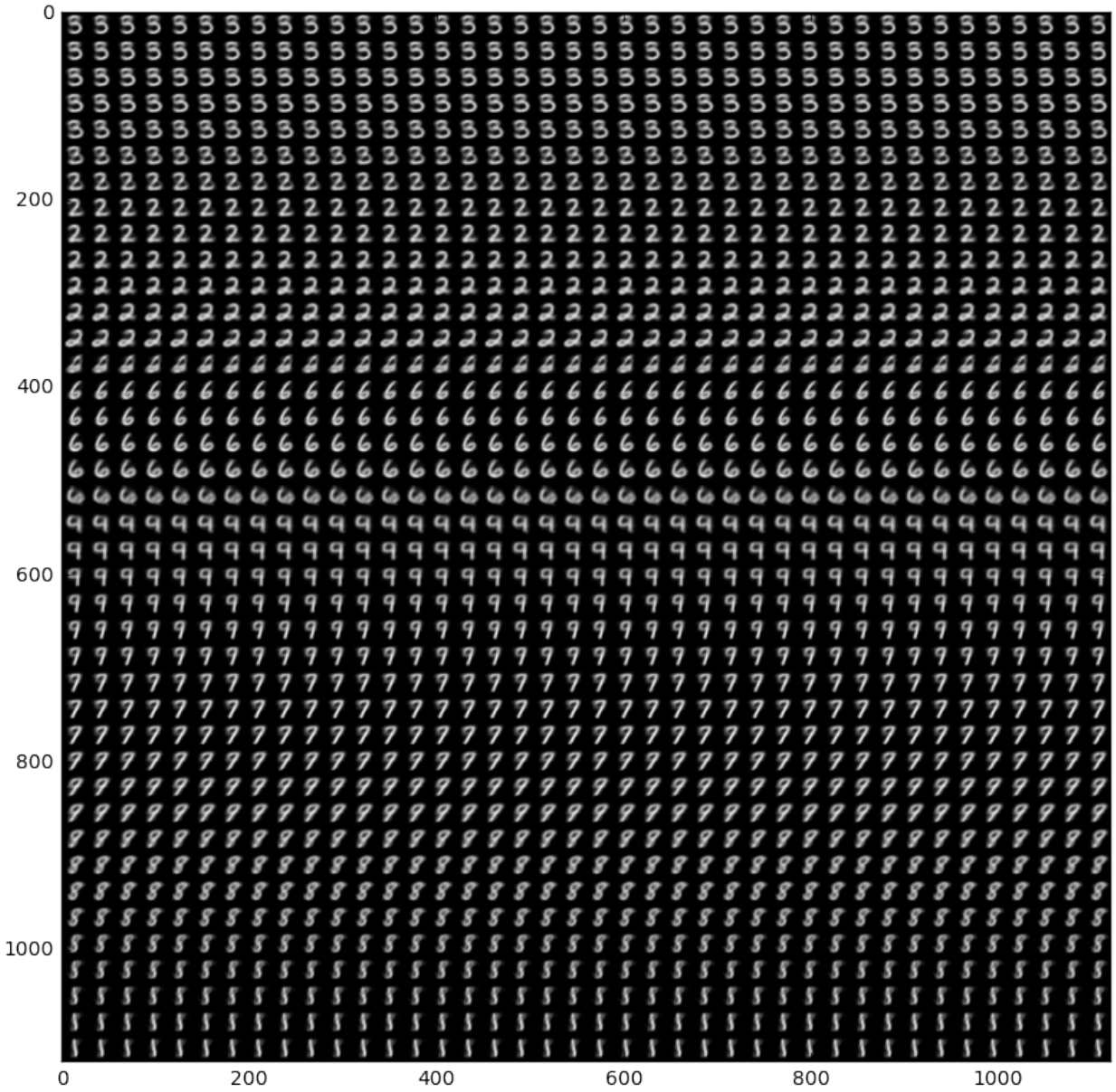
MNIST - Sparse Cauchy VAE



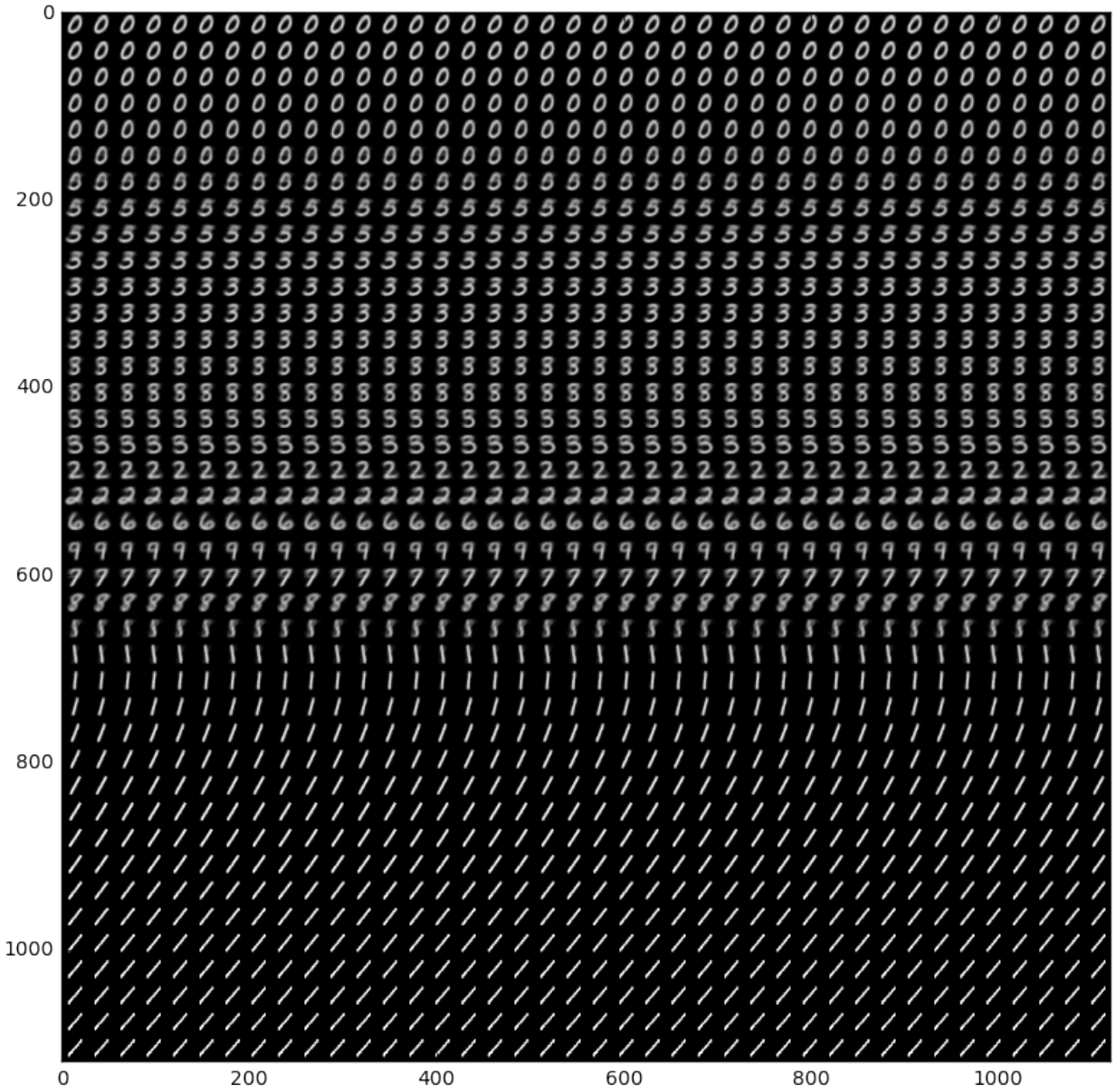
MNIST - Sparse Cauchy VAE



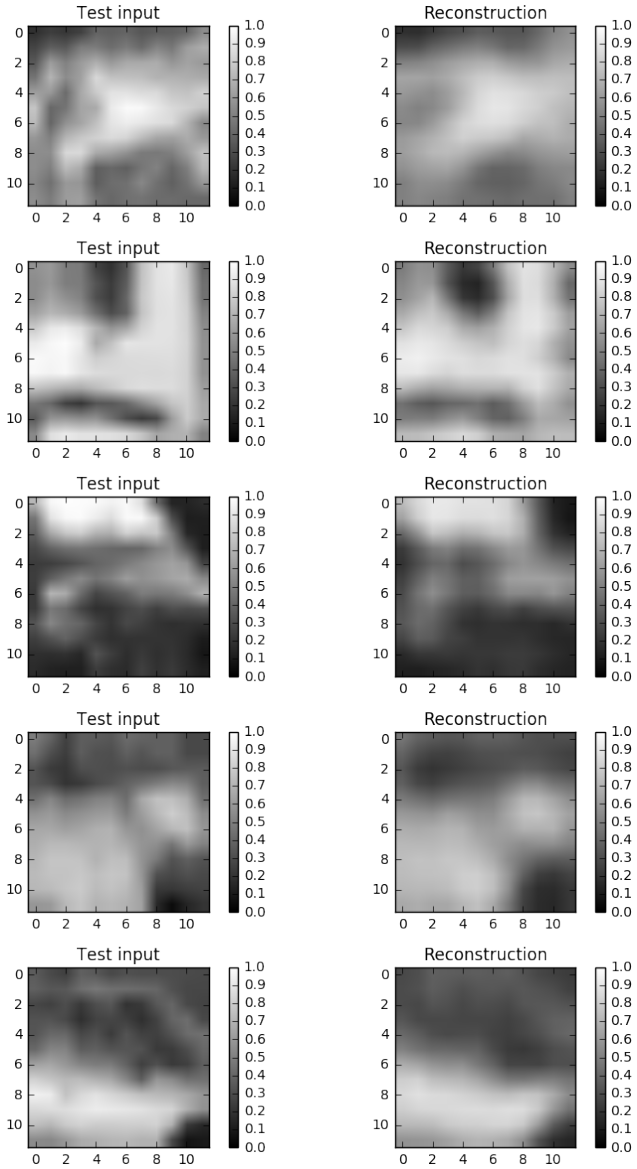
MNIST - Sparse Cauchy VAE



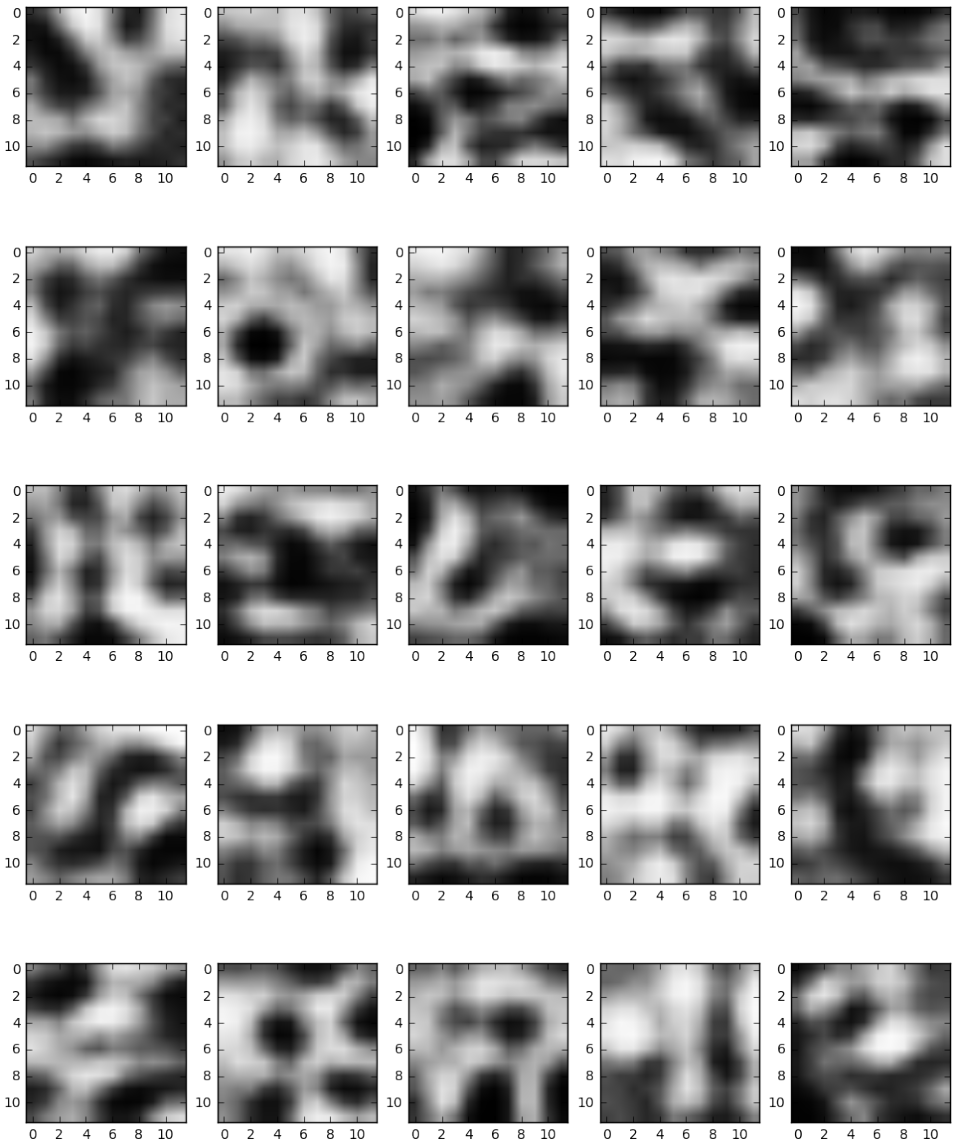
MNIST - Sparse Cauchy VAE (zoomed out)



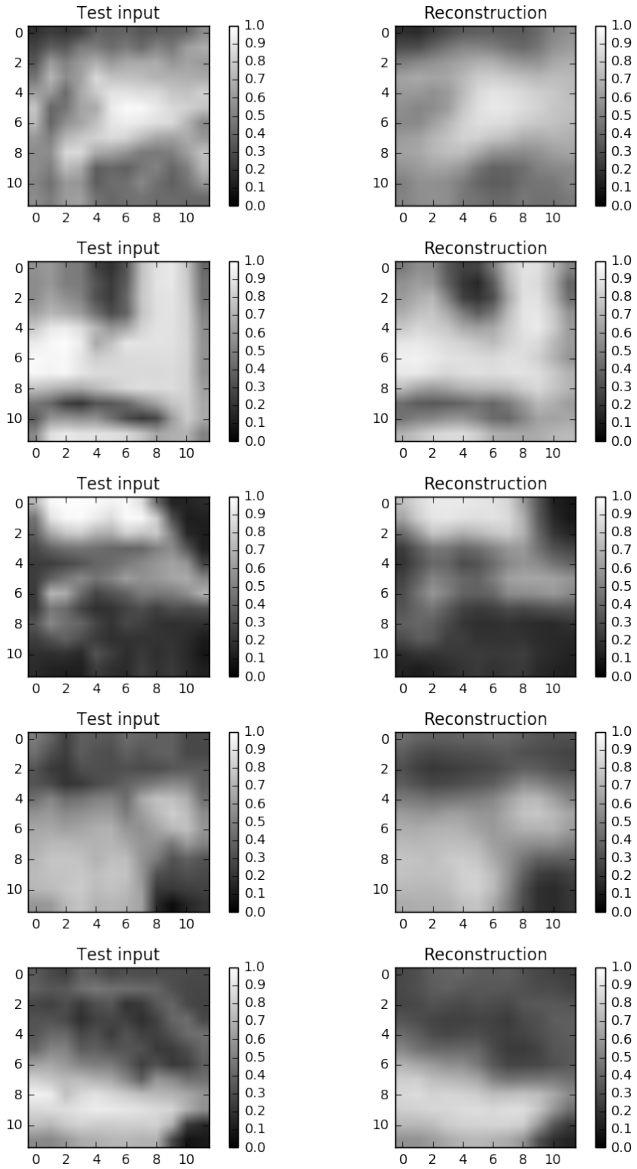
CIFAR - Normal VAE



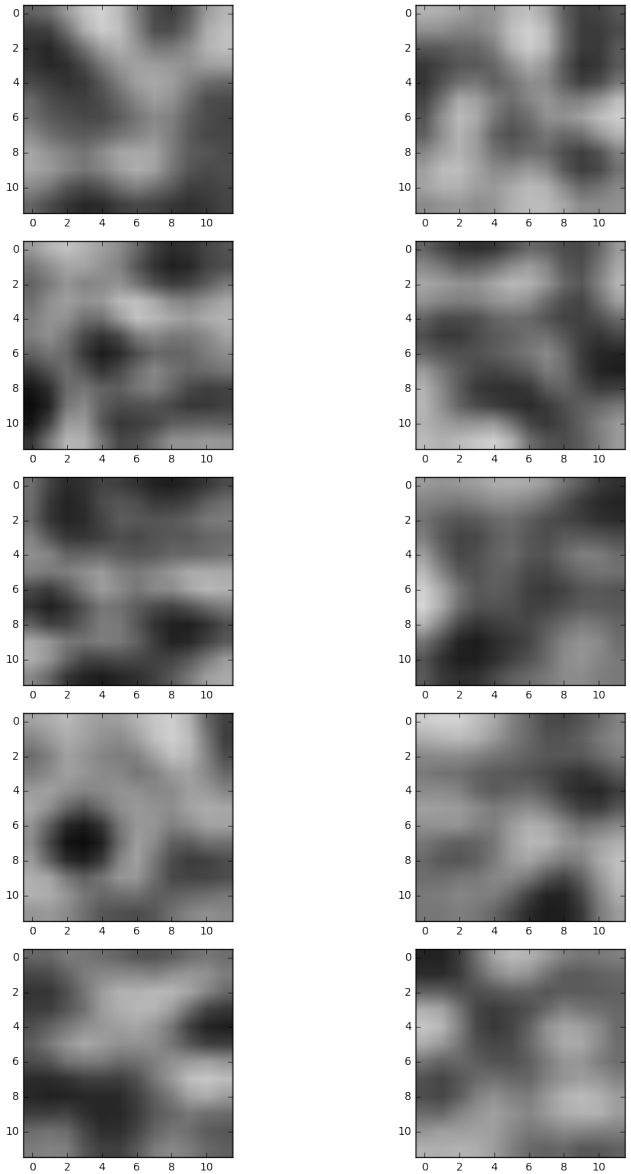
CIFAR - Normal VAE



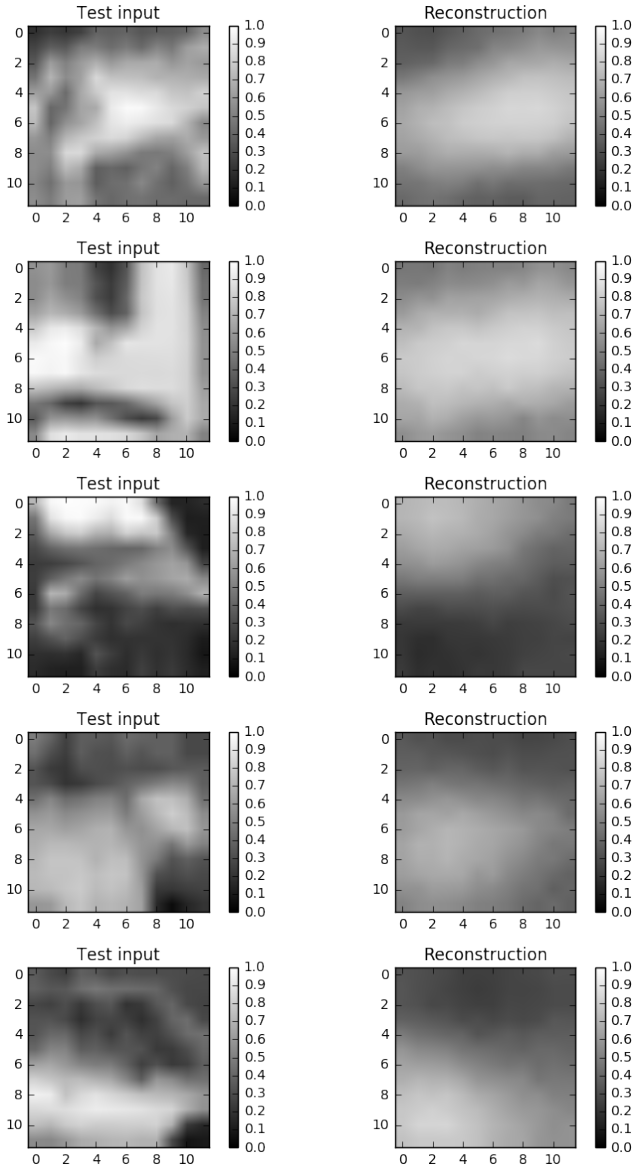
CIFAR - Sparse Laplace VAE



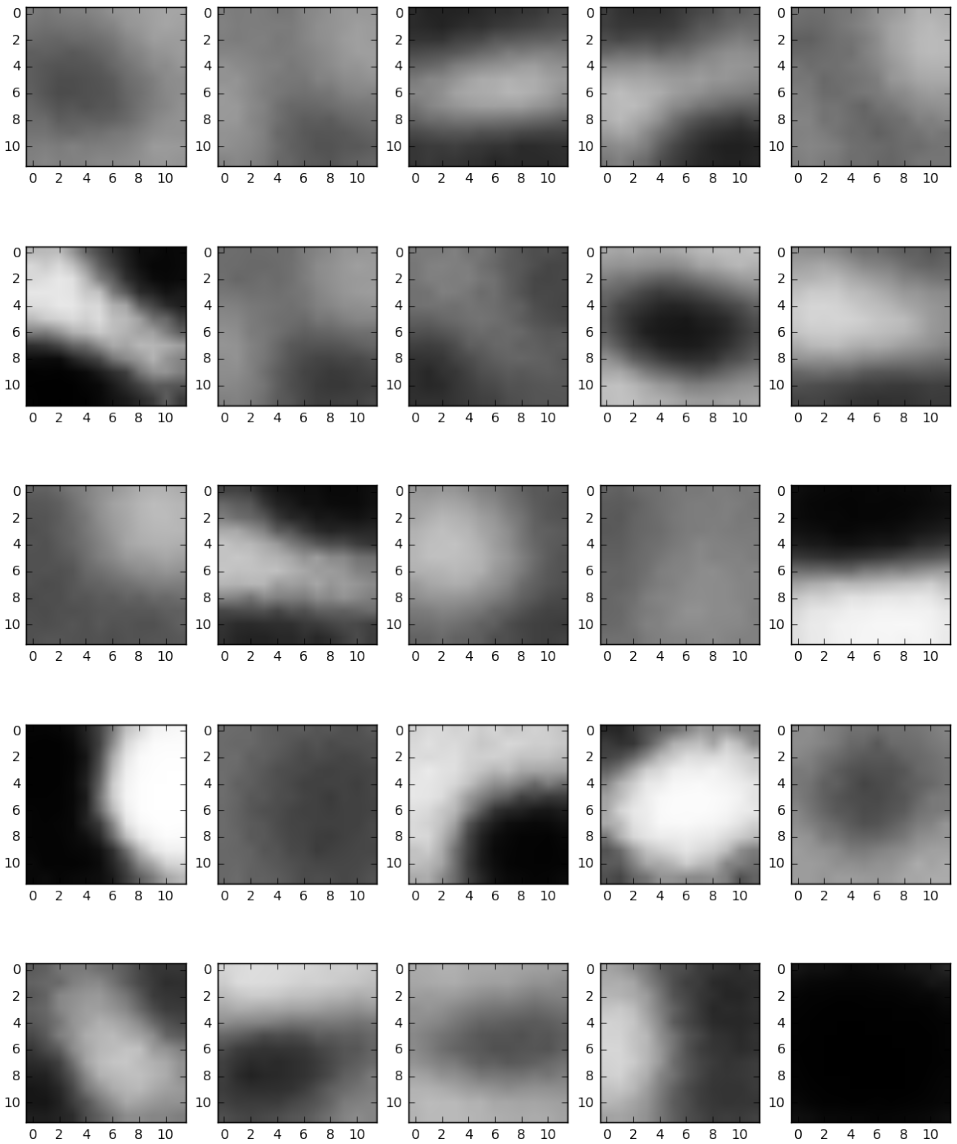
CIFAR - Sparse Laplace VAE



CIFAR - Sparse Cauchy VAE



CIFAR - Sparse Cauchy VAE



Conclusions

- All 3 do fairly well at reproduction across all tasks. VAE is best, laplace VAE slightly worse, cauchy even worse (mostly noticeable only on CIFAR task)
- Laplace VAE and Cauchy VAE have significantly more informative latent dimensions in the MNIST and CIFAR tasks. Cauchy VAE is noticeably better than Laplace in both cases.
- Laplace CIFAR recreats something approaching a biologically realistic retina receptive field.
- Laplace VAE has a tendency to only use 1 dimension in the 2 dimension test outputs.