Regularized Autoencoders

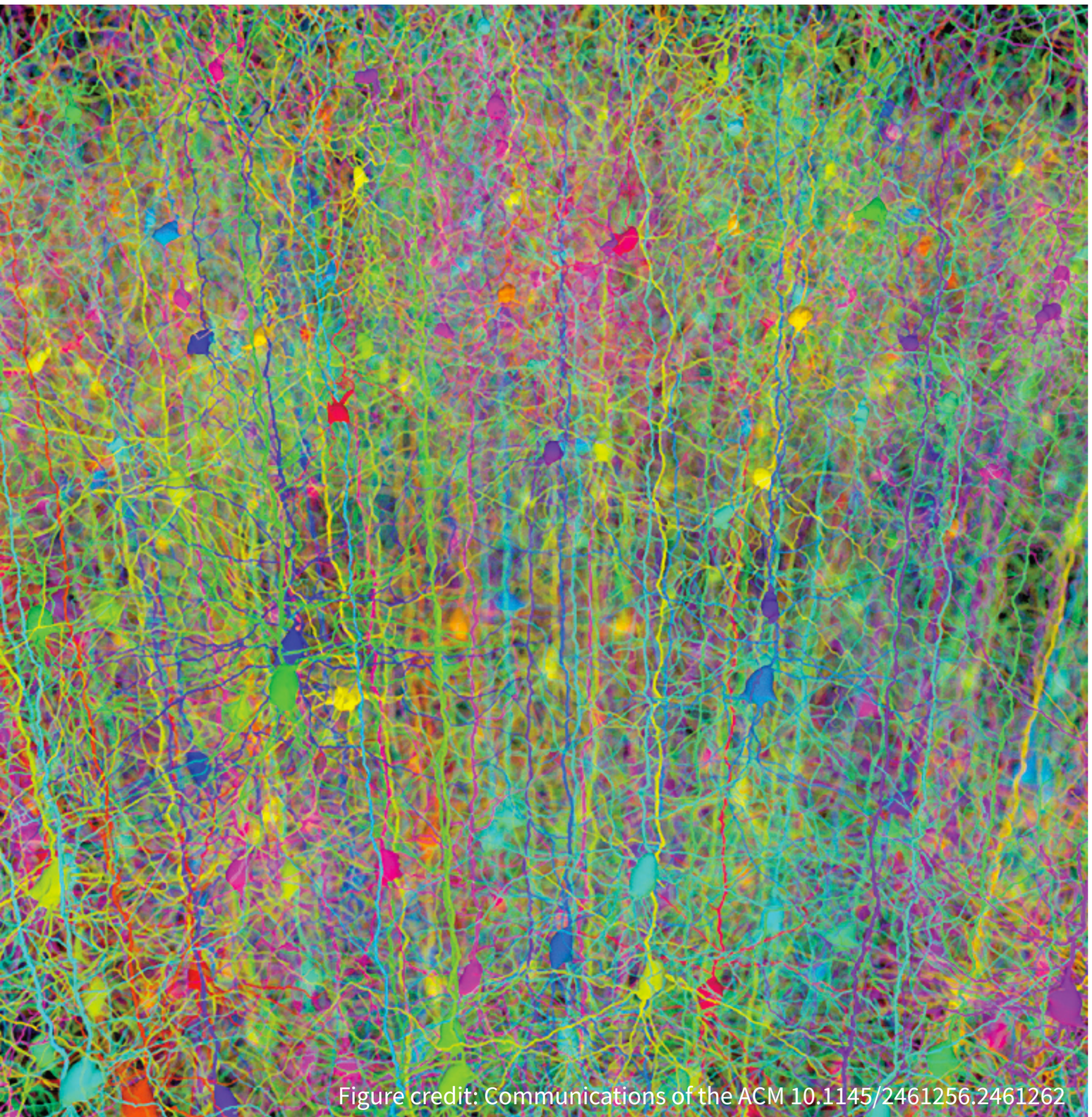


Figure credit: Communications of the ACM 10.1145/2461256.2461262

GRAHAM TAYLOR

VECTOR INSTITUTE

SCHOOL OF ENGINEERING
UNIVERSITY OF GUELPH

CANADIAN INSTITUTE
FOR ADVANCED RESEARCH

UNIVERSITY of GUELPH

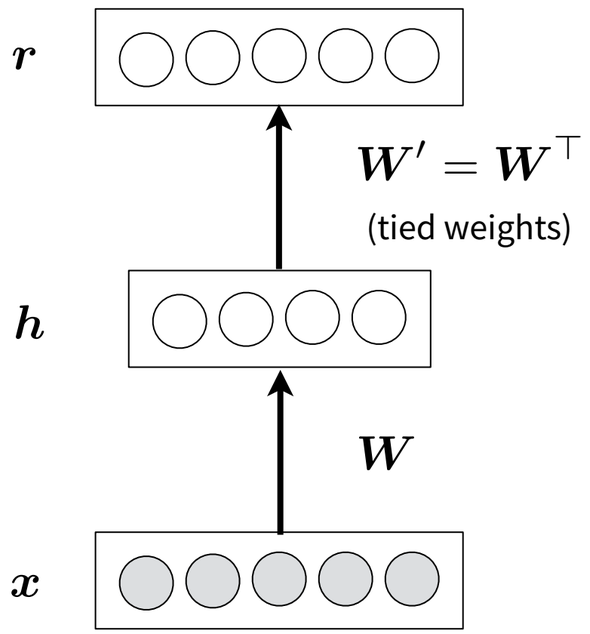
CHANGING LIVES
IMPROVING LIFE

CIFAR
CANADIAN INSTITUTE FOR ADVANCED RESEARCH

Undercomplete Hidden Layer

Hidden layer is undercomplete if smaller than the input layer

- Hidden layer "compresses" the input
- Will compress well only for the training distribution



Hidden units will be:

- good features for the training distribution

- but bad for other types of input

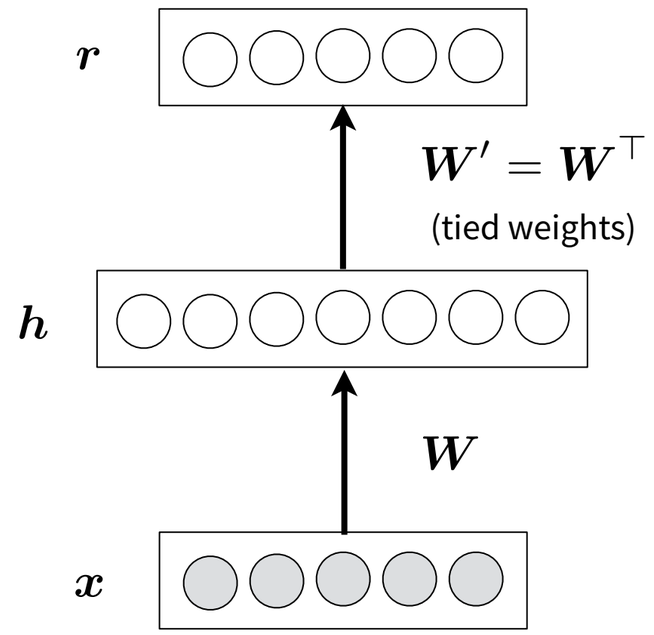
Reproduced from Hugo Larochelle's slides

Overcomplete Hidden Layer

Hidden layer is undercomplete if greater than the input layer

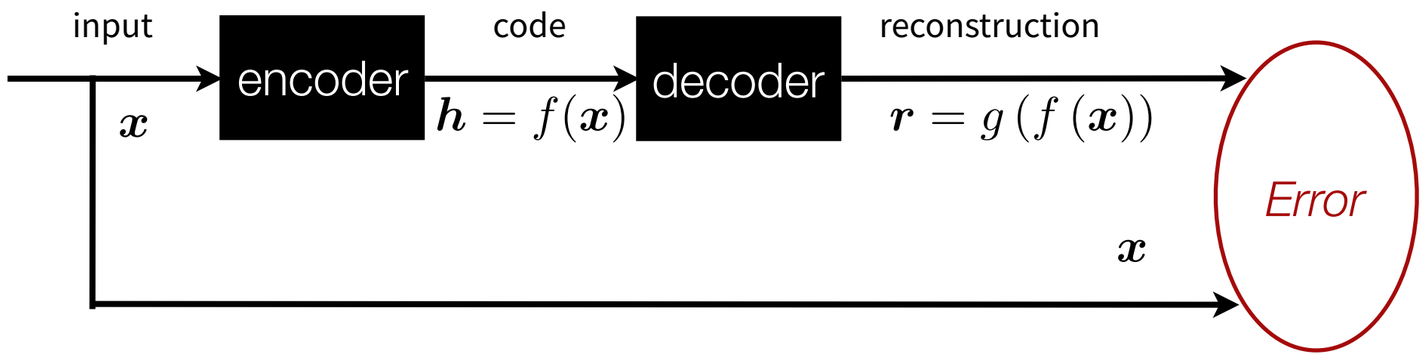
- Hidden layer "expands" the input
- Each hidden unit could copy a different component

No guarantee that the hidden units will extract meaningful structure



Reproduced from Hugo Larochelle's slides

Regularized Auto-encoders



- Permit code to be higher-dimensional than the input

- Capture structure of the training distribution due to predictive opposition b/w reconstruction distribution and regularizer

- Regularizer tries to make enc/dec as simple as possible

Simple?

Simple?

- Reconstruct the input from the code and make the code compact
  (PCA, auto-encoder with bottleneck)

Simple?

- Reconstruct the input from the code and make the code compact
  (PCA, auto-encoder with bottleneck)
- Reconstruct the input from the code and make the code sparse
  (sparse auto-encoders)

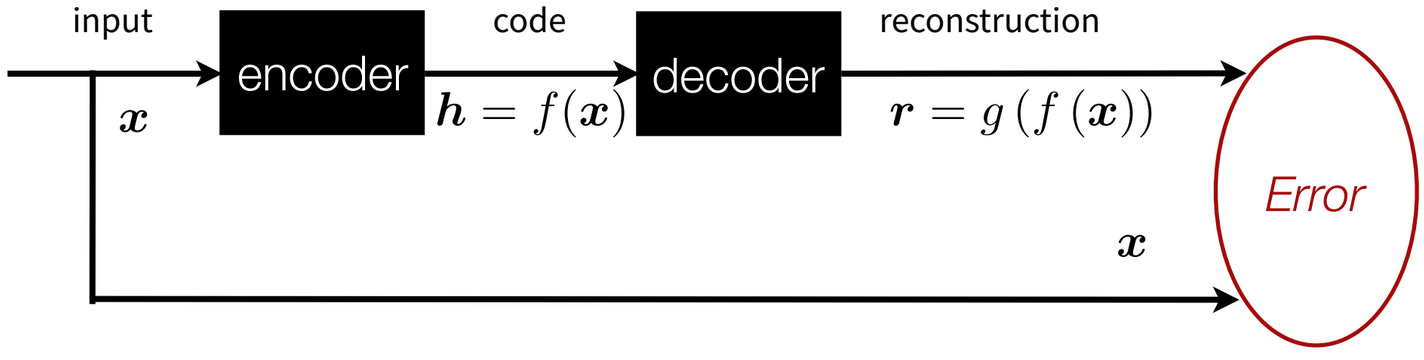
Simple?

- Reconstruct the input from the code and make the code compact
  (PCA, auto-encoder with bottleneck)
- Reconstruct the input from the code and make the code sparse
  (sparse auto-encoders)
- Add noise to the input or code and reconstruct the cleaned-up version
  (denoising auto-encoders)

Simple?

- Reconstruct the input from the code and make the code compact
  (PCA, auto-encoder with bottleneck)
- Reconstruct the input from the code and make the code sparse
  (sparse auto-encoders)
- Add noise to the input or code and reconstruct the cleaned-up version
  (denoising auto-encoders)
- Reconstruct the input from the code and make the code insensitive to the input (contractive auto-encoders)

Sparse Auto-encoders



$$J = L(\boldsymbol{x}, \boldsymbol{r}) + \lambda \sum_j \mathrm{KL}\left(\rho || \hat{\rho}_j\right)$$

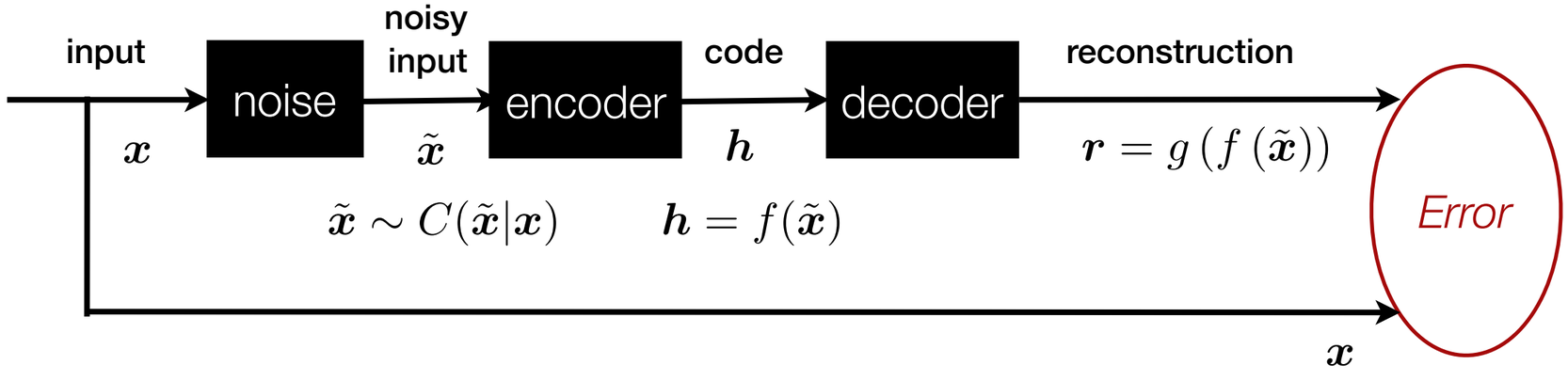
$$\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^{m} h_j(\boldsymbol{x}^{(i)}) : \text{mean activation}$$

ρ : target activation (small)

- Apply a sparsity penalty to the hidden activations
- Also see Predictive Sparse Decomposition (Kavukcuoglu et al. 2008)

(Vincent et al. 2008)

Denoising Auto-encoders



$$J = L(\boldsymbol{x}, g\left(f\left(\tilde{\boldsymbol{x}}\right)\right))$$

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\epsilon}$$

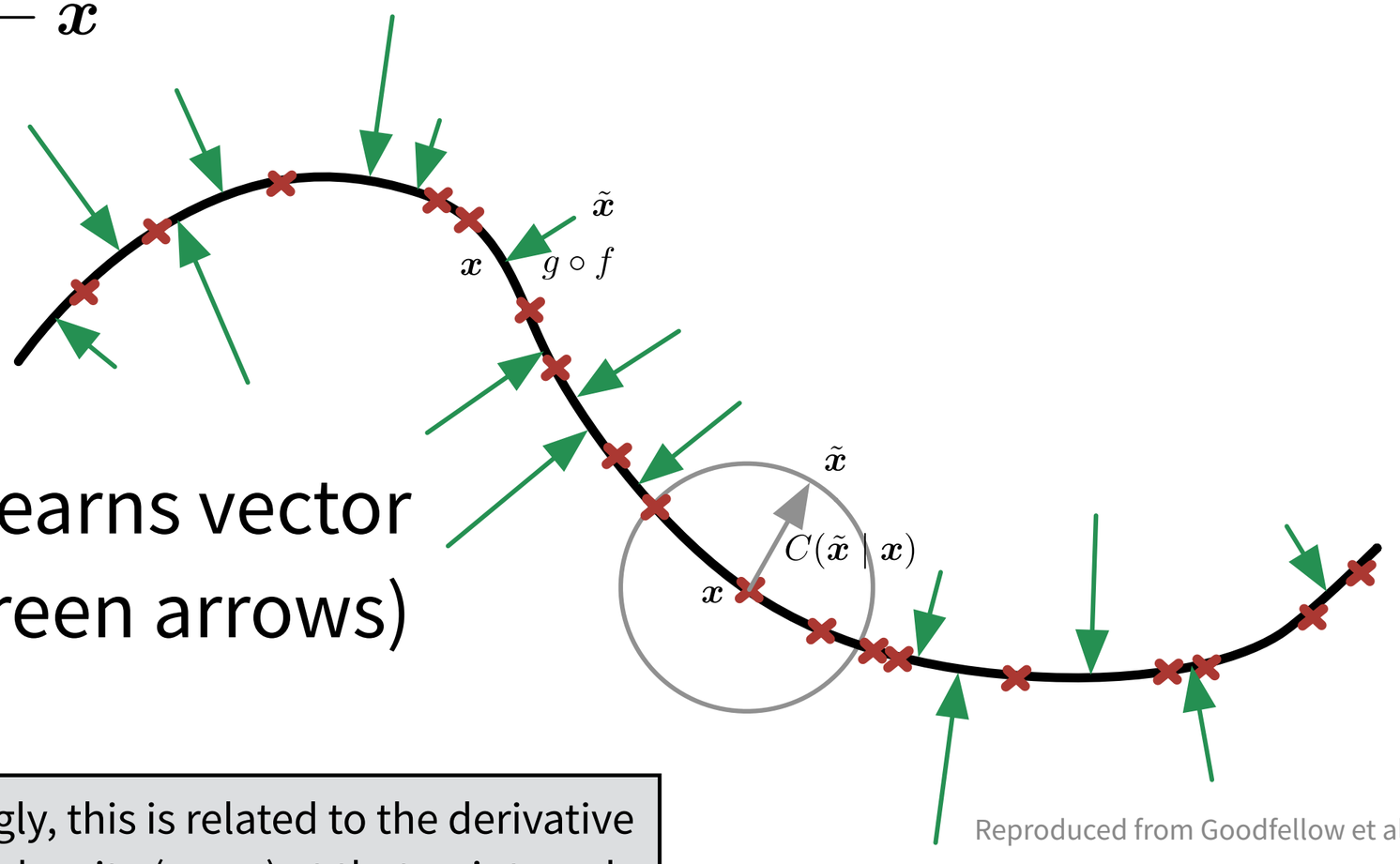
$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$$

only one possible choice of noise model

- The code can be viewed as a lossy compression of the input
- Learning drives it to be a good compressor for training examples (and hopefully others as well) but not arbitrary inputs

What a Denoising Autoencoder Learns

- Trained to map a corrupted data point back to the original data point
- Grey arrow demonstrates how one data point is corrupted
- The vector $g(f(\tilde{\boldsymbol{x}})) - \tilde{\boldsymbol{x}}$ points toward the nearest point on the manifold
- This, autoencoder learns vector field $g(f(\boldsymbol{x})) - \boldsymbol{x}$ (green arrows)

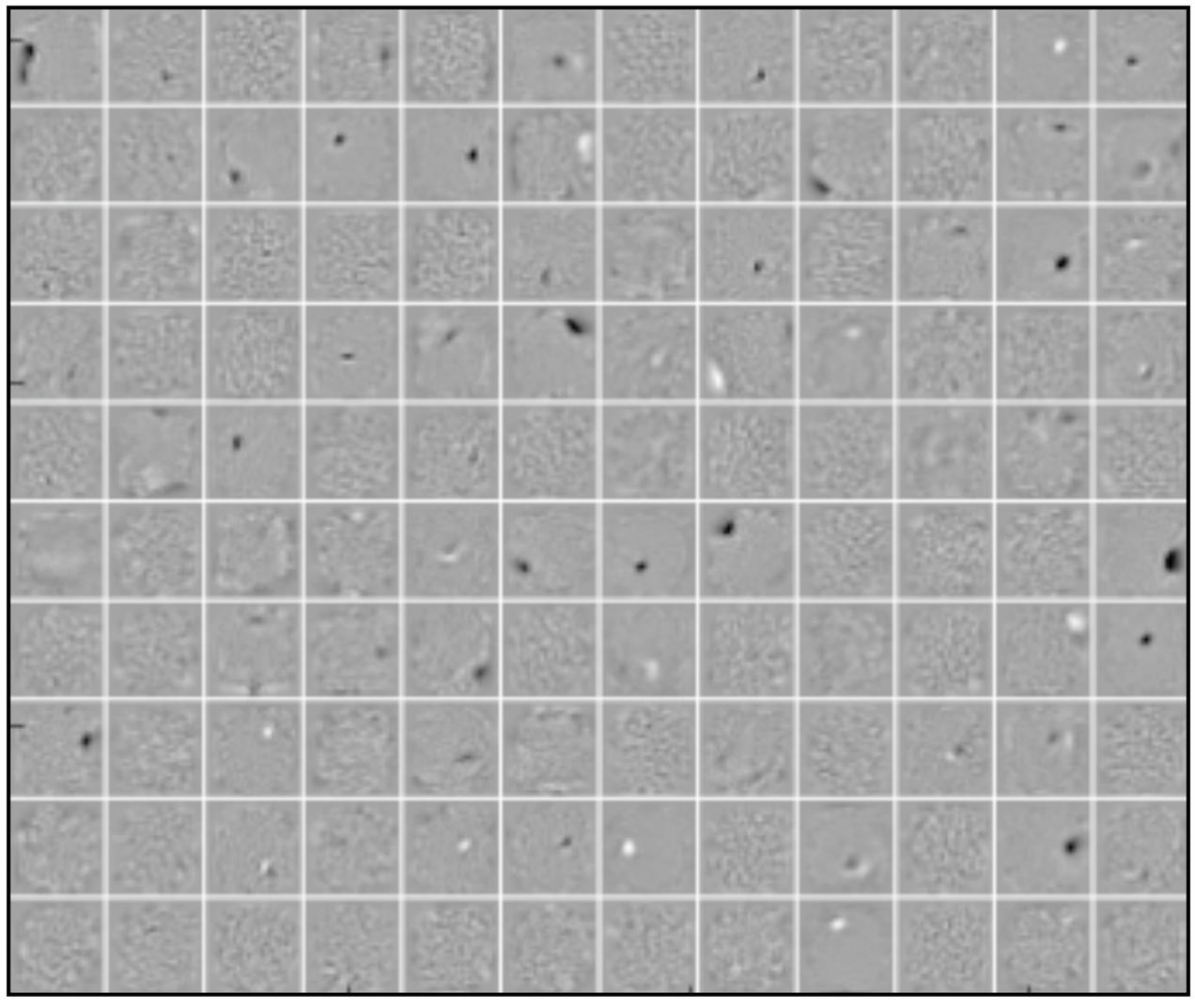


Reproduced from Goodfellow et al.

Interestingly, this is related to the derivative of the log-density (score) at that point, and also to the curvature at that point

Learned Filters: Noiseless

- No corruption, cross-entropy loss

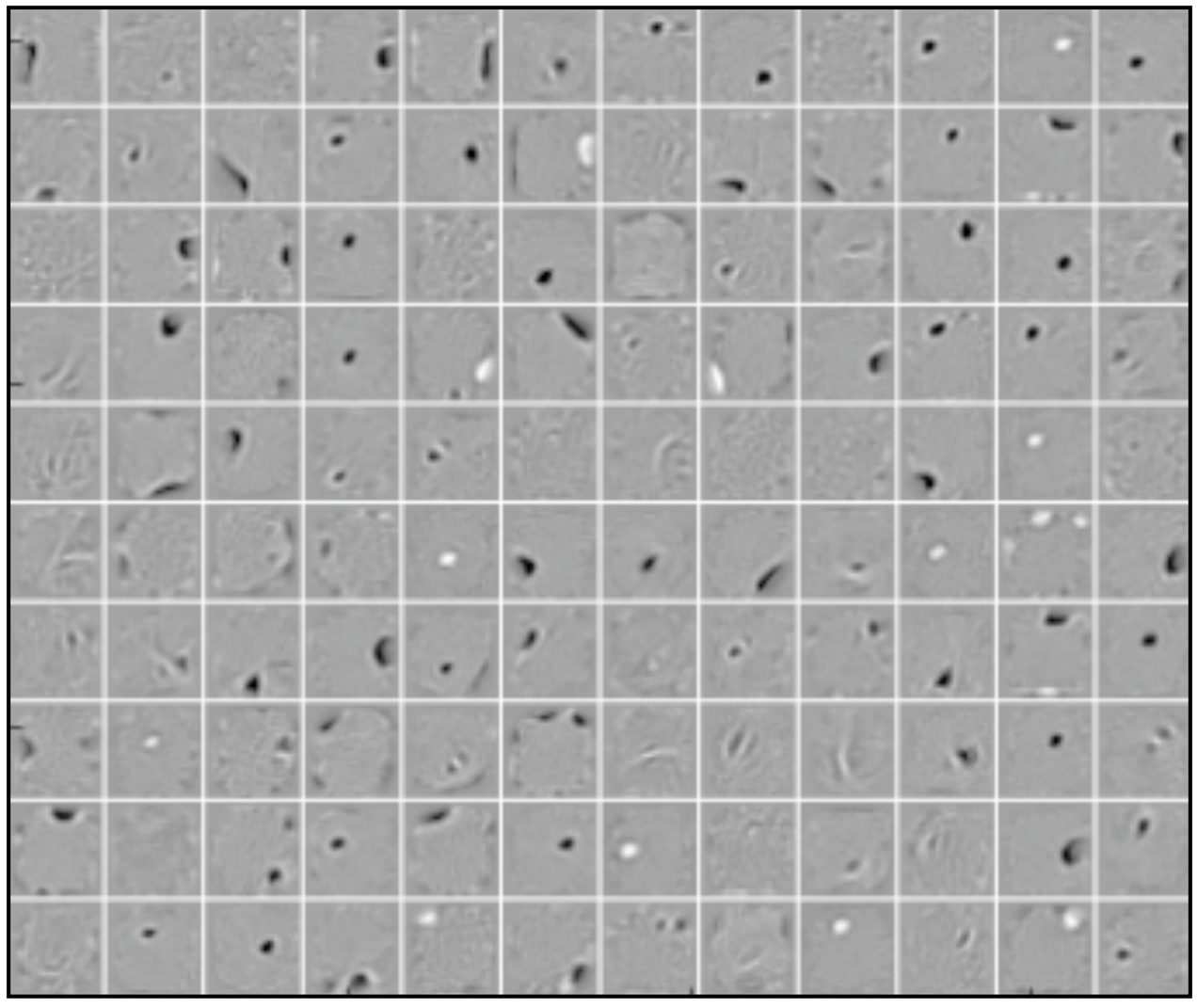


(Vincent et al. 2008)

Reproduced from Hugo Larochelle's slides

Learned Filters: Light Corruption

- 25% corruption

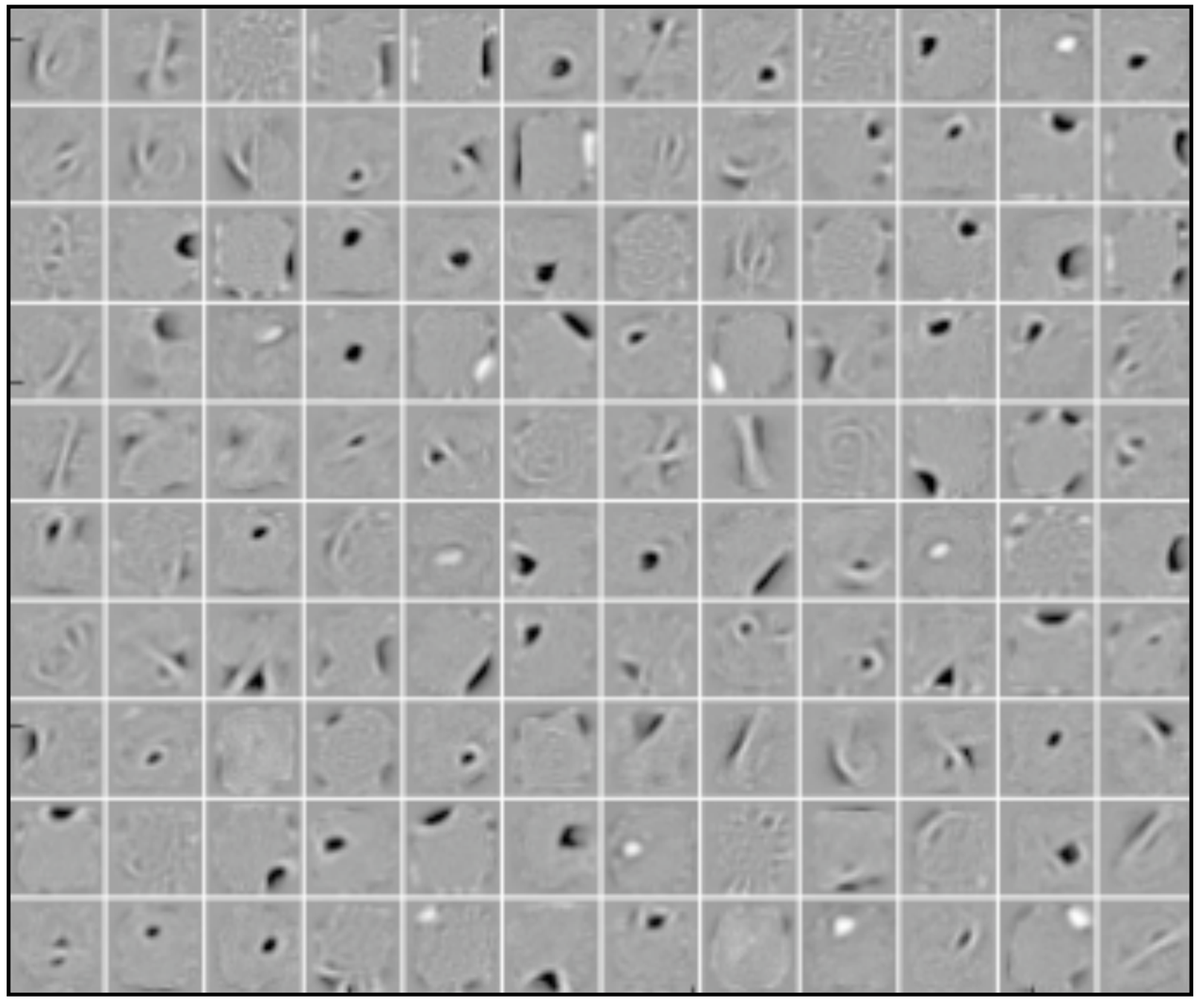


(Vincent et al. 2008)

Reproduced from Hugo Larochelle's slides

Learned Filters: Strong Corruption

- 50% corrupted inputs



(Vincent et al. 2008)

Reproduced from Hugo Larochelle's slides

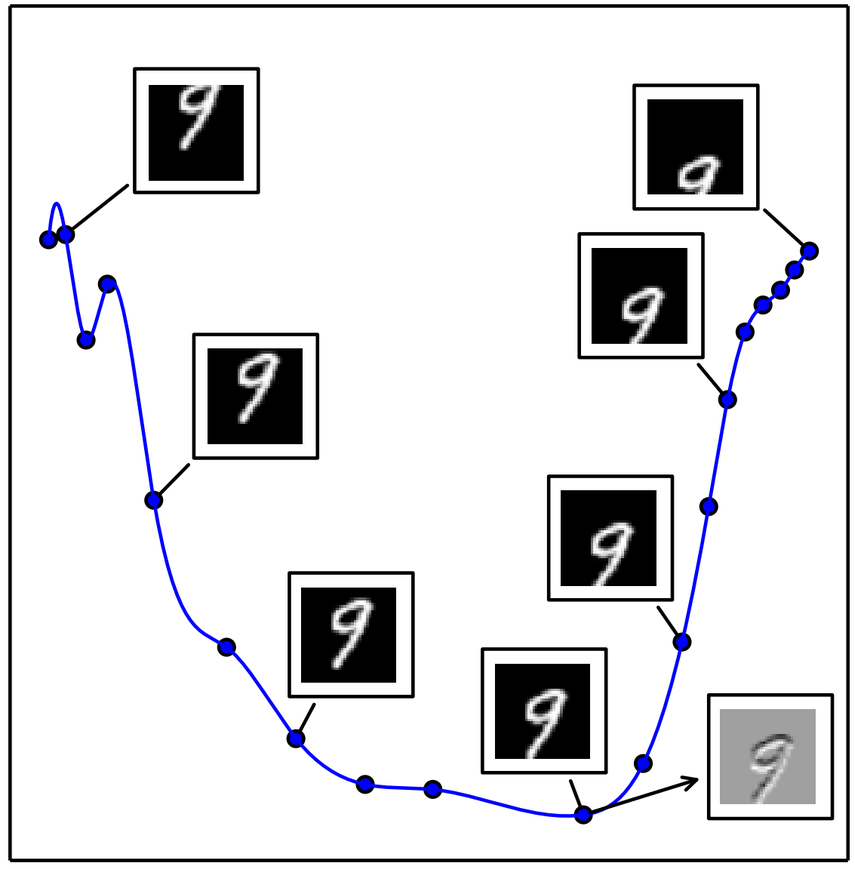
Autoencoders and Manifolds

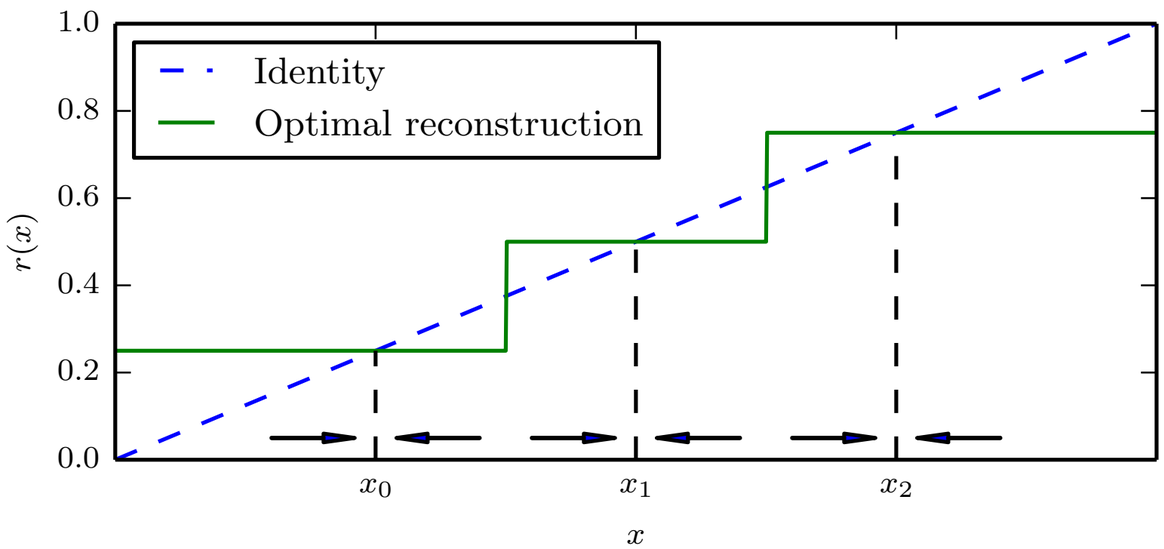
Regularized autoencoder training schemes involve a “battle” between two forces:

1. Learning a representation of a training example such that it can be reconstructed

2. Satisfying the constraint or regularization penalty

    - These techniques typically prefer solutions that are less sensitive to the input

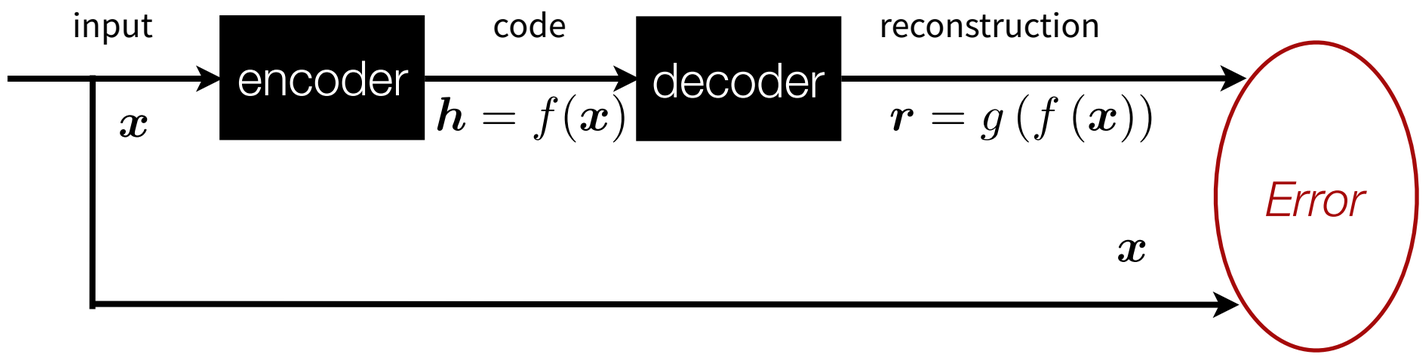




Reproduced from Goodfellow et al.

(Rifai et al. 2011)

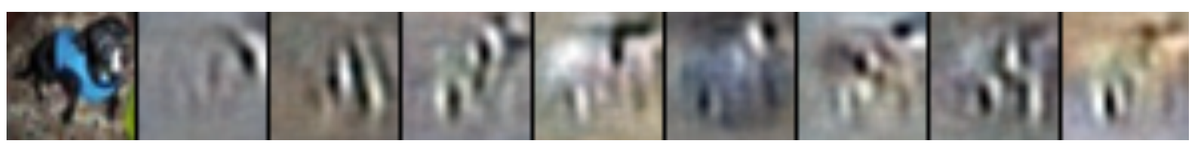
Contractive Auto-encoders



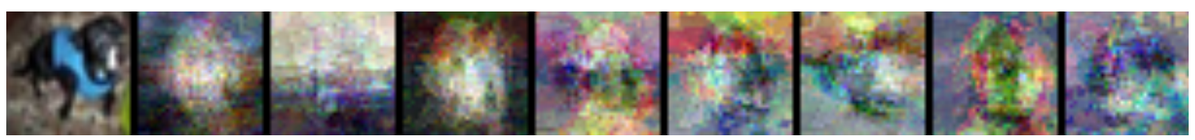
$$J = L(\boldsymbol{x}, \boldsymbol{r}) + \lambda \left|\left| \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} \right|\right|_F^2$$

Tangent vectors on (CIFAR-10) manifold estimated by:

CAE



Local PCA



location on manifold determined by dog image

- Learn good models of high-dimensional data (Bengio et al. 2013)
- Can obtain good representations for classification
- Can produce good quality samples by a random walk near the manifold of high density (Rifai et al. 2012)

via Rifai et al. 2012