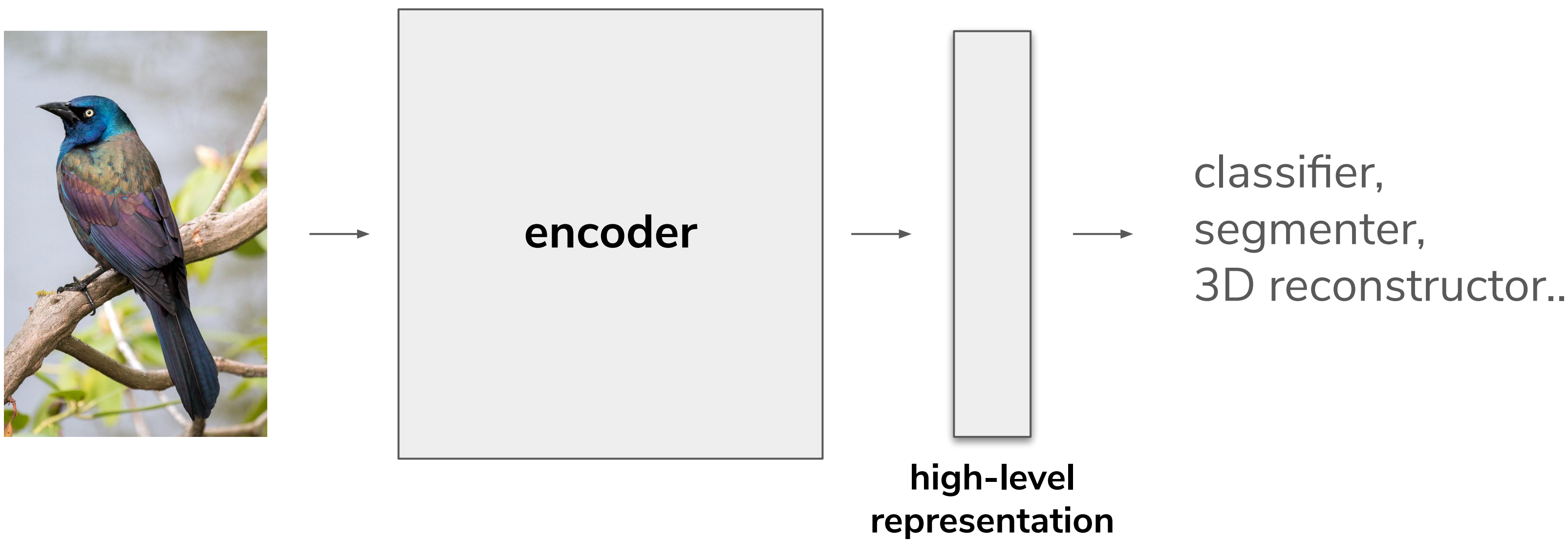
Stacked Denoising Autoencoders (2010)

Presented by **Owen Jow**

Original work by **Vincent**, **Larochelle**, **Lajoie**, **Bengio**, and **Manzagol**

Representation Learning

Goal: learn to produce useful encodings for general downstream tasks.



What Makes a Good Encoder?

A good encoder should retain distinguishing information about the input.

- **Infomax principle:** seek an encoder f_θ which maximizes mutual information between the input X and its higher-level representation $Y = f_\theta(X)$.

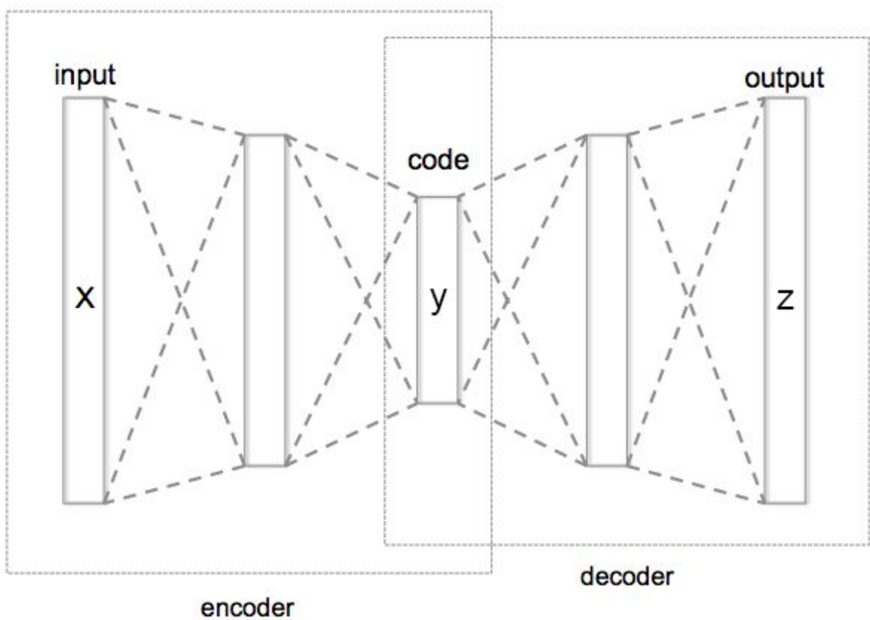
$$\begin{aligned}\max_\theta \mathbb{I}(X;Y) &= \max_\theta \left[\mathbb{H}(X) - \mathbb{H}(X|Y)\right] \\ &= \max_\theta \left[-\mathbb{H}(X|Y)\right] \\ &= \max_\theta \mathbb{E}_{q(X,Y)}\left[\log q(X|Y)\right]\end{aligned}$$

$$\boxed{\begin{aligned}\mathbb{I}&: \text{ mutual information} \\ \mathbb{H}&: \text{ entropy}\end{aligned}}$$

Since Y is a function of X, this is a self-supervised objective!

Autoencoders

The reconstruction objective for a traditional autoencoder maximizes a lower bound on the mutual information between X and Y.

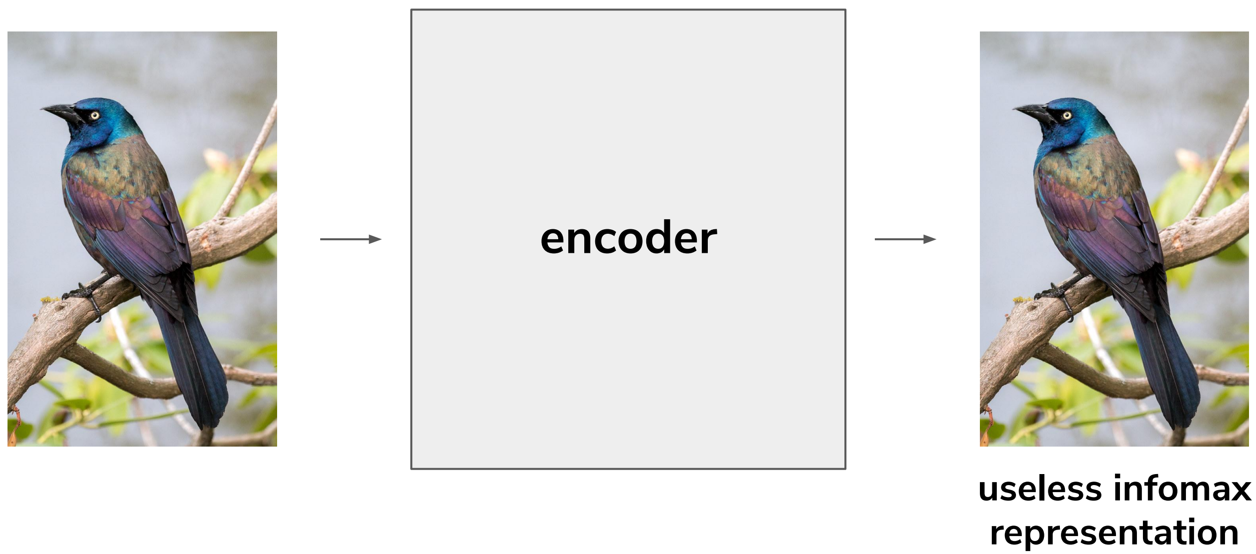


$$\max_{\theta} \mathbb{E}_{q(X,Y)}[\log q(X|Y)]$$
$$\geq \max_{\theta,\theta'} \mathbb{E}_{q^0(X)}[\log p(X|Y = f_\theta(X);\theta')]$$

Are we done?

Merely Retaining Information Is Not Enough

Even if the encoding maximizes mutual information, it might not be "useful."

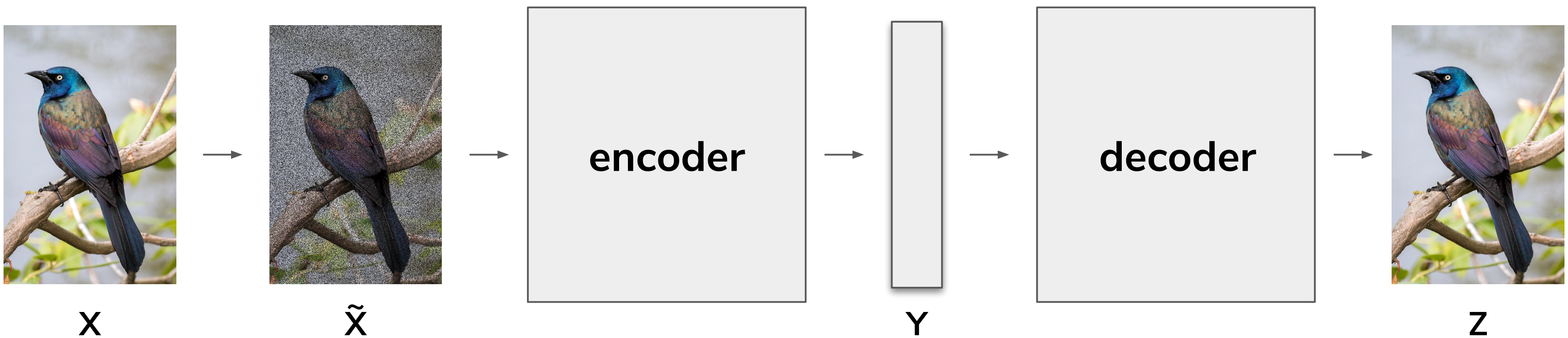


- To avoid learning the identity function, we're forced to constrain the representation somehow (e.g. make it **sparse** or **lower-dimensional**).

Is reconstruction really the best objective to use?

A Better Objective

Denoising is a more challenging objective which better encourages learned representations to capture meaningful correlations in the input data distribution.

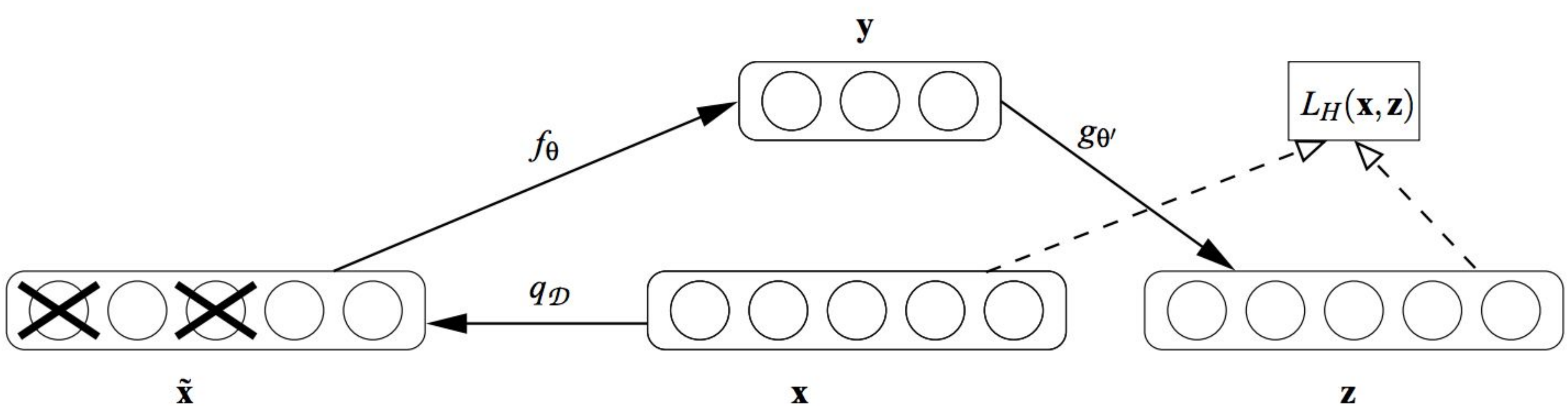


- Representations should be robust to noise in the inputs
- Representations should be useful for recovering clean inputs

Denoising Autoencoders

- Randomly corrupt the input
- Deterministically encode the **corrupted input** (learned)
- Deterministically decode the encoded **corrupted input** (learned)

Train the encoder/decoder to produce an output equivalent to the (clean!) input.

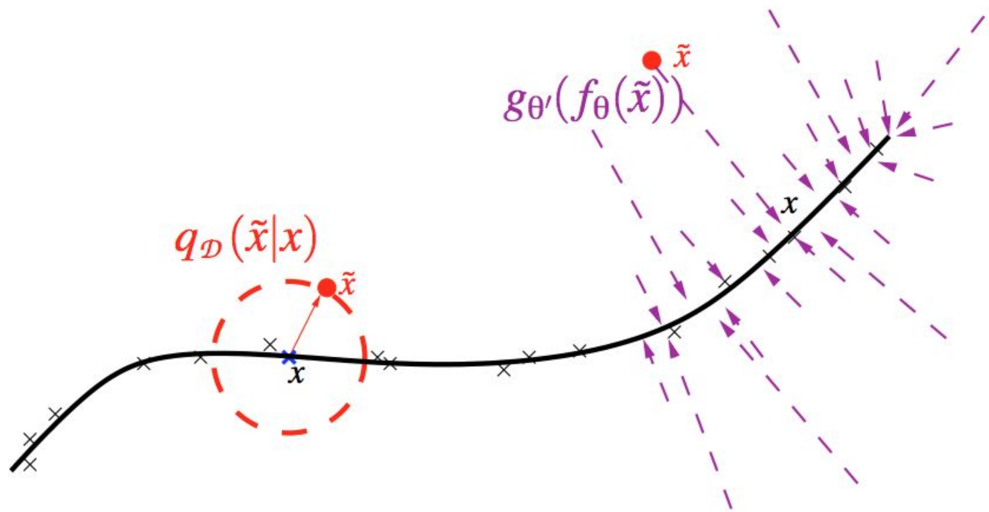


The identity mapping no longer suffices as an encoding scheme.

Manifold Learning Perspective

Assumption: natural data lies on some manifold in high-dimensional space.

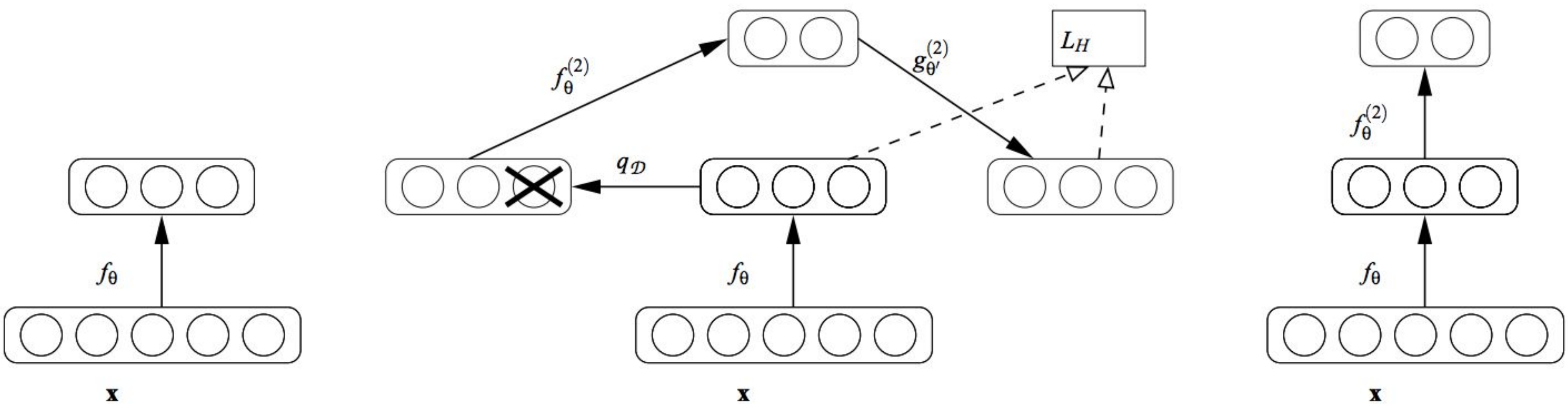
- Corruption takes data points away from the manifold
- Denoising projects them back onto the manifold



In order to bring corrupted points at different locations back to the data manifold, the denoising autoencoder must understand the underlying manifold structure.

Stacked Denoising Autoencoders

As restricted Boltzmann machines become deep belief networks, denoising autoencoders become **stacked** denoising autoencoders.

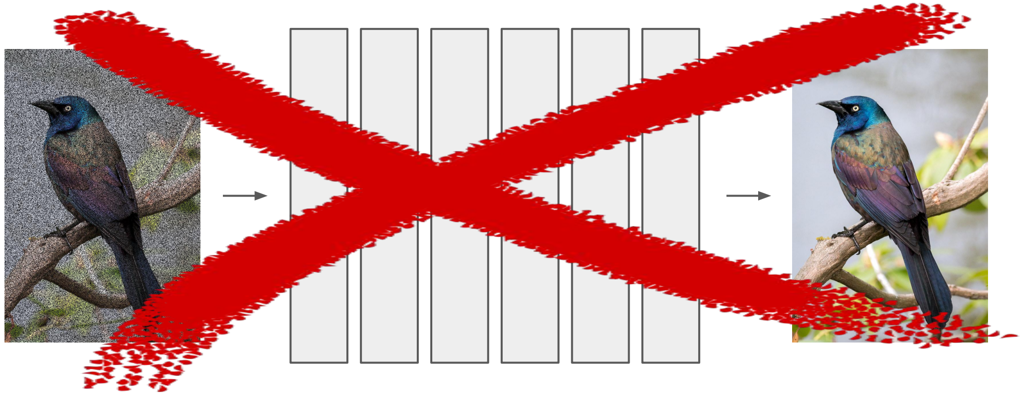


Train layer by layer. Once each layer's encoder is learned, apply it to clean input and use the resulting encoding as clean input to train the next layer.

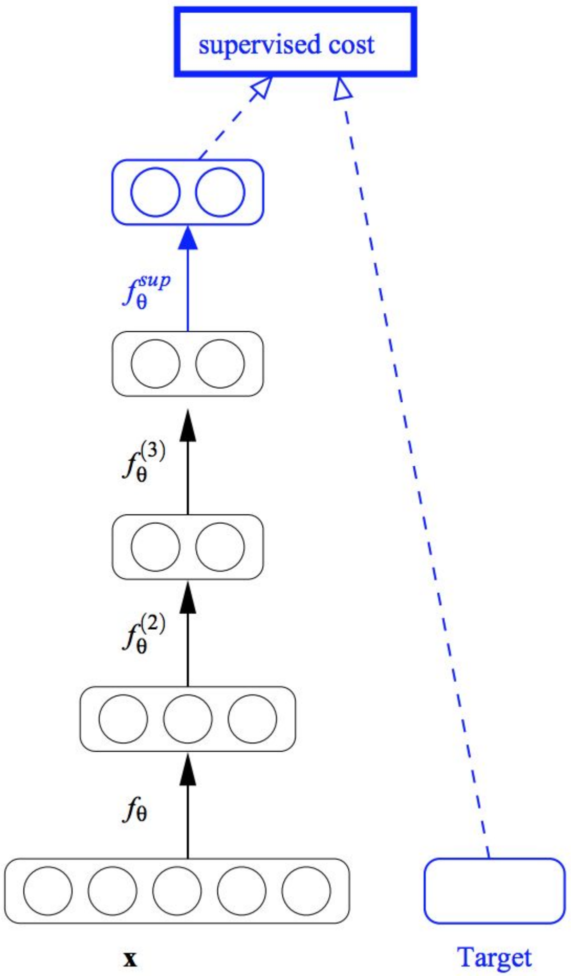
Note: This Isn't Really About Denoising

Denoising is not the end goal here.

During training, later layers attempt to reconstruct whatever representation they receive from the preceding layer (rather than the original input image).



Applying Learned Representations to Downstream Tasks



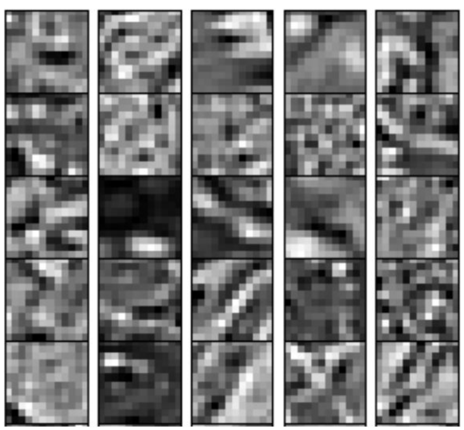
Use the highest-level representation as input to an arbitrary (e.g. supervised learning) algorithm.

- Optionally fine-tune all of the encoders.

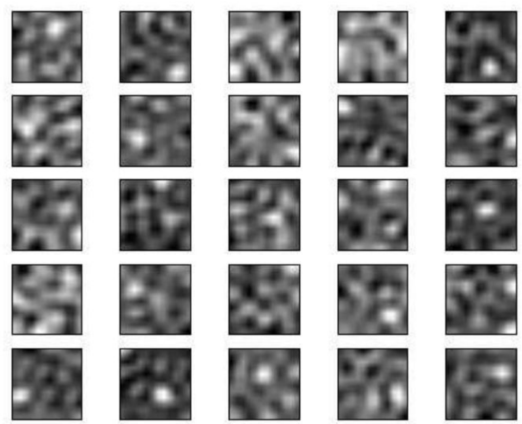
Empirical Validation

Feature Detectors Learned by Regular Autoencoders

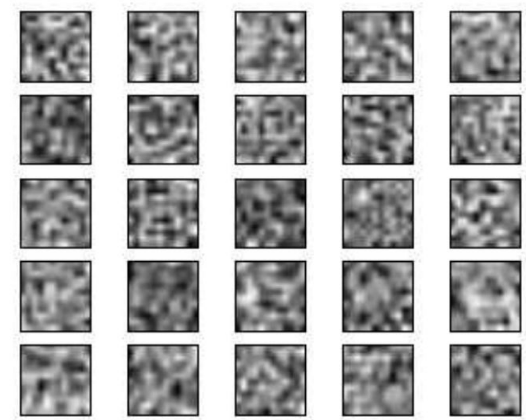
- Train a single **regular autoencoder** on natural image patches
- Visualize learned weights for different neurons in the first hidden layer



example patches used for training



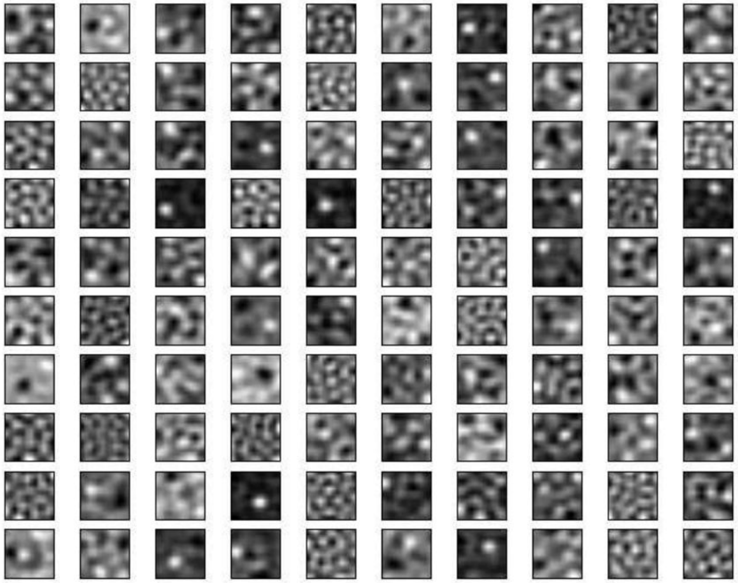
filters learned by under-complete autoencoder



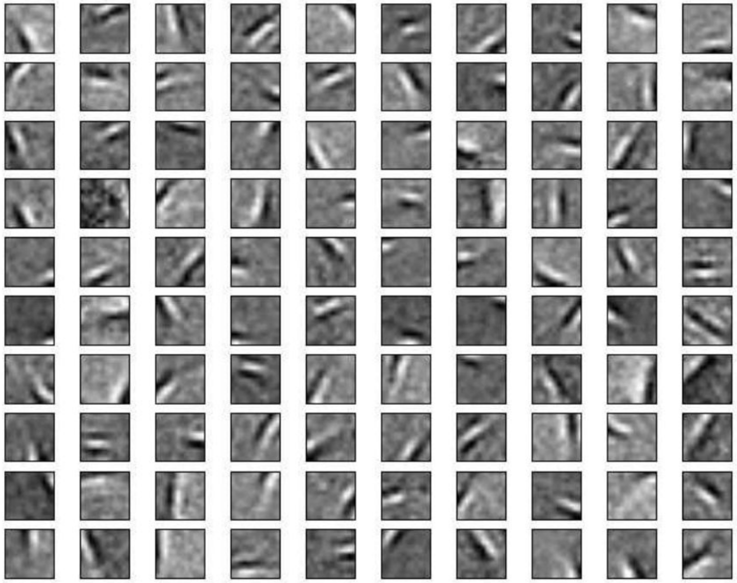
filters learned by over-complete autoencoder

Feature Detectors Learned by Denoising Autoencoders

- Train a single **denoising autoencoder** on natural image patches
- Compare to a regular autoencoder with weight decay regularization



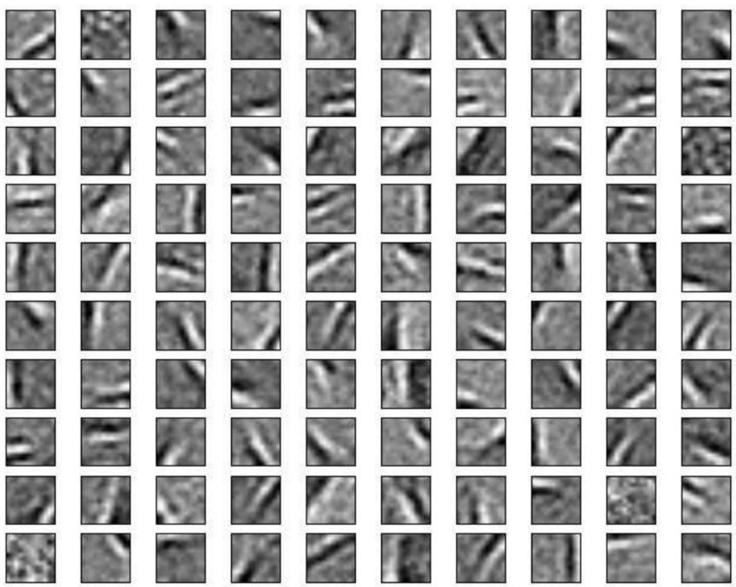
filters learned by over-complete regular autoencoder with L2 weight decay



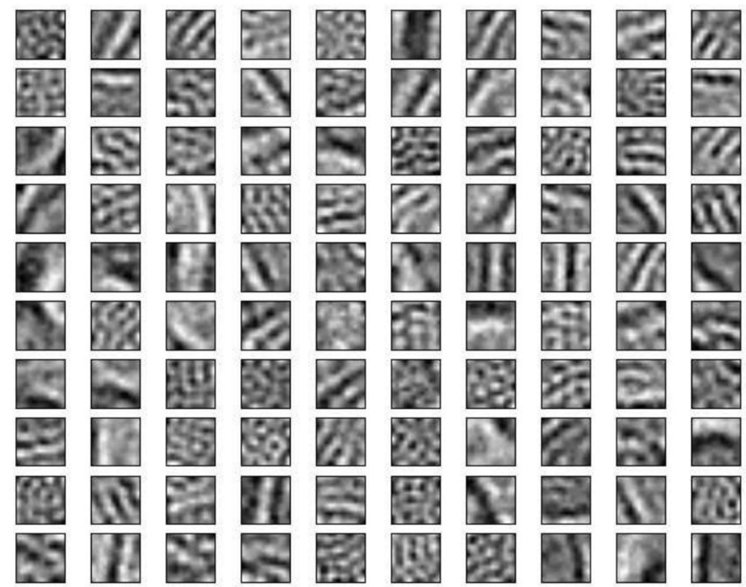
filters learned by denoising autoencoder with Gaussian noise

The Effect of Noise Type

- Try corrupting inputs with different types of noise
- In all cases, filters learned by denoising autoencoders are Gabor-like



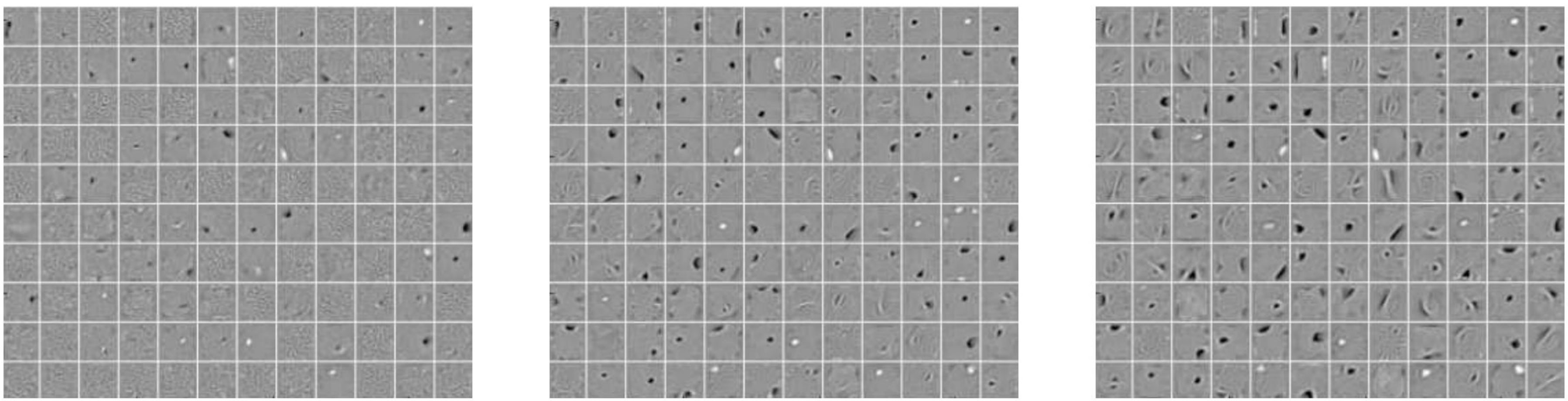
filters learned by denoising autoencoder
with 10% salt-and-pepper noise



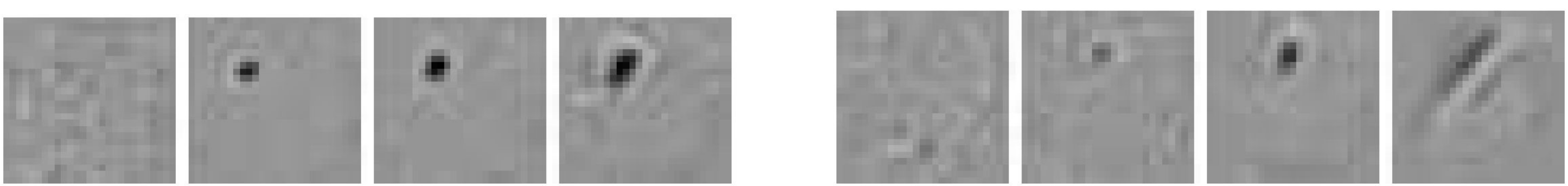
filters learned by denoising autoencoder
with 55% zero-masking noise

Feature Detectors Learned from Handwritten Digits

- Train multiple instances of denoising autoencoders on MNIST data
- Start from the same initialization, but apply varying degrees of masking noise



(a) No corruption (b) 25% corruption (c) 50% corruption



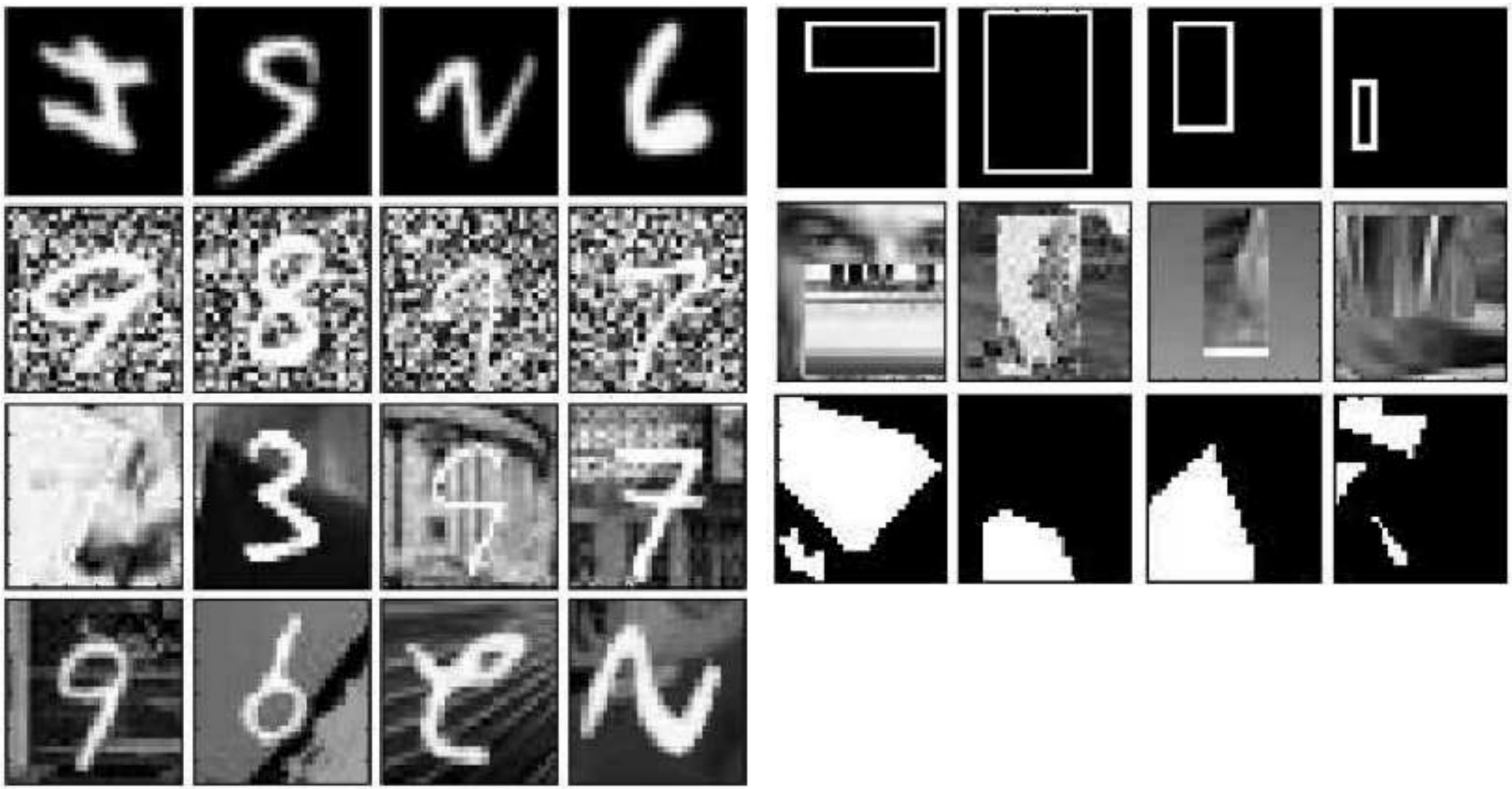
(d) Neuron A (0%, 10%, 20%, 50% corruption) (e) Neuron B (0%, 10%, 20%, 50% corruption)

Classification Problems for Subsequent Experiments

Next: evaluate learned representations across 10 different classification settings.

Name	Description	# Training Samples
MNIST	-	50,000
basic	MNIST subset	10,000
rot	Random rotations	10,000
bg-rand	Random noise backgrounds	10,000
bg-img	Random image backgrounds	10,000
bg-img-rot	Random rotations and image backgrounds	10,000
rect	Classify rectangles as “tall” or “wide”	10,000
rect-img	**rect** with image backgrounds	10,000
convex	Classify shapes as “convex” or “concave”	6,000
tzanetakis	Classify audio clip as belonging to one of 10 genres	10,000

Harder MNIST Variations **Artificial Binary Classification Problems**



(a) *rot*, *bg-rand*, *bg-img*, *bg-img-rot* (b) *rect*, *rect-img*, *convex*

Results

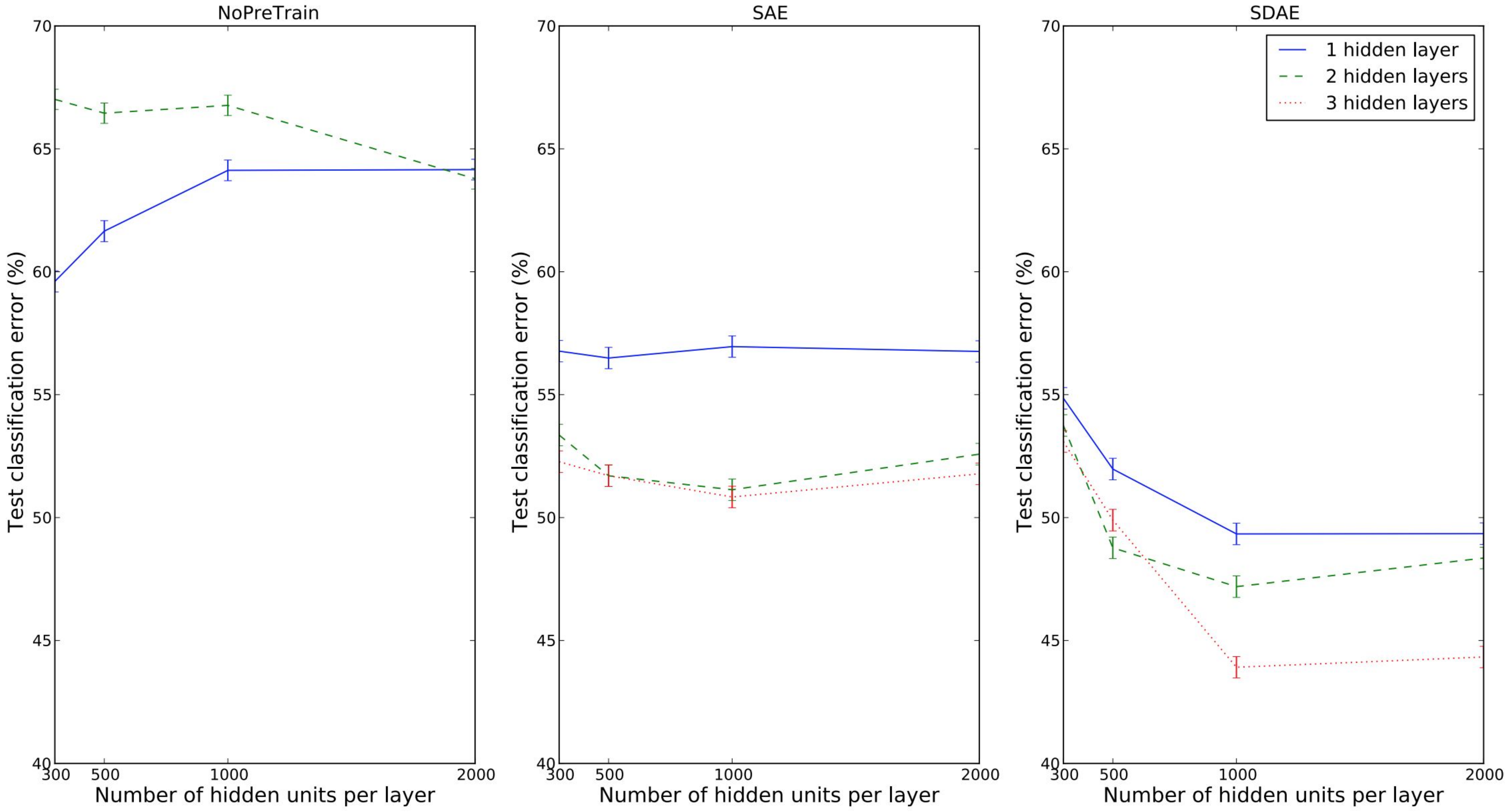
Is it best to pretrain with stacked regular autoencoders (**SAE-3**), deep belief networks (**DBN-3**), or stacked denoising autoencoders (**SDAE-3**)?

- In all but one case, SDAE-3 performs the best within a confidence interval.

Data Set	**SVM**$_{rbf}$	**DBN-1**	**SAE-3**	**DBN-3**	**SDAE-3** (ν)
MNIST	**1.40**±0.23	**1.21**±0.21	**1.40**±0.23	**1.24**±0.22	**1.28**±0.22 (25%)
basic	**3.03**±0.15	3.94±0.17	3.46±0.16	3.11±0.15	**2.84**±0.15 (10%)
rot	11.11±0.28	14.69±0.31	10.30±0.27	10.30±0.27	**9.53**±0.26 (25%)
bg-rand	14.58±0.31	9.80±0.26	11.28±0.28	**6.73**±0.22	10.30±0.27 (40%)
bg-img	22.61±0.37	**16.15**±0.32	23.00±0.37	**16.31**±0.32	**16.68**±0.33 (25%)
bg-img-rot	55.18±0.44	52.21±0.44	51.93±0.44	47.39±0.44	**43.76**±0.43 (25%)
rect	**2.15**±0.13	4.71±0.19	2.41±0.13	2.60±0.14	**1.99**±0.12 (10%)
rect-img	24.04±0.37	23.69±0.37	24.05±0.37	22.50±0.37	**21.59**±0.36 (25%)
convex	19.13±0.34	19.92±0.35	**18.41**±0.34	**18.63**±0.34	**19.06**±0.34 (10%)
tzanetakis	**14.41**±2.18	18.07±1.31	**16.15**±1.95	18.38±1.64	**16.02**±1.04(0.05)

Increasing Model Breadth and Depth

As representational capacity increases, SDAEs see the greatest benefit.



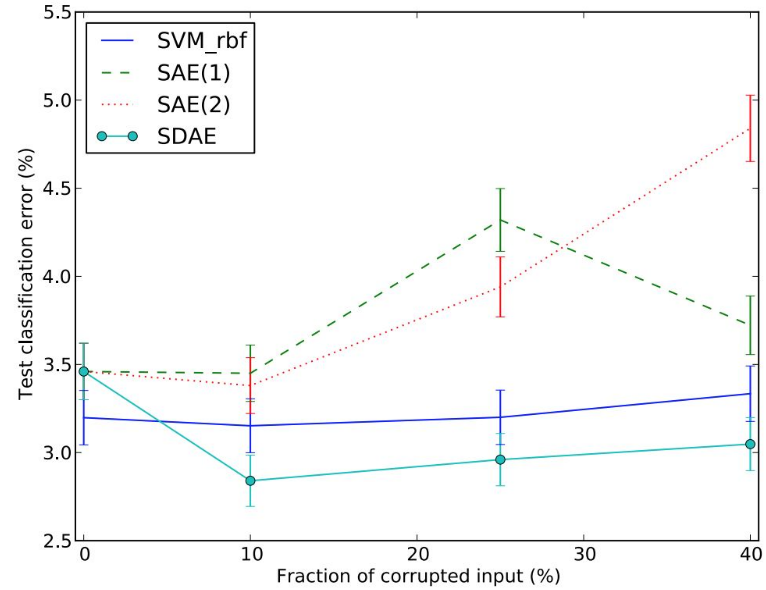
Training with Noisy Inputs

Does training with noisy inputs (jitter) provide the same benefits as SDAEs?

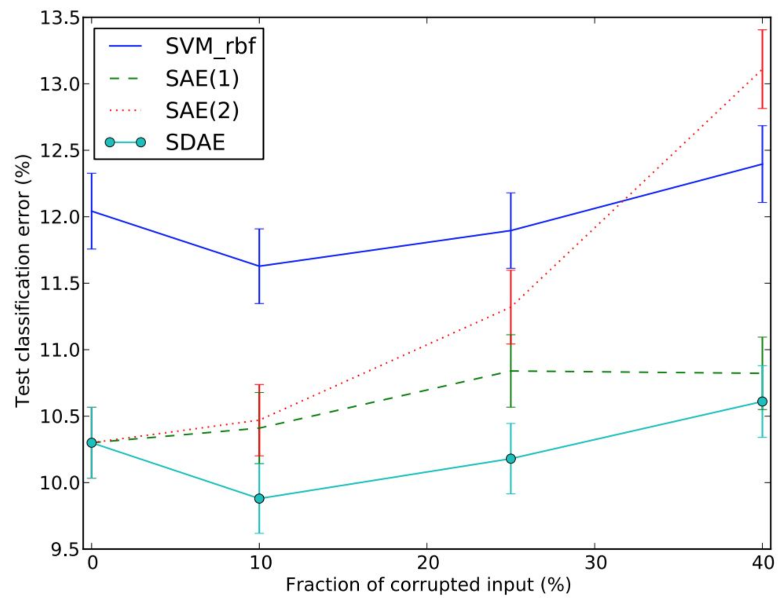
Compare SDAE-3 test error to that of other algorithms trained with noisy inputs.

- **SAE(1):**
  use noisy inputs for pretraining only
- **SAE(2):**
  use noisy inputs for pretraining and supervised fine-tuning

→ **SDAE pretraining consistently wins.**



(a) *basic*



(b) *rot*

Corruption Types and Emphasis

- **Noise types:** masking (MN), salt-and-pepper (SP), additive Gaussian (GS)
- **Emphasis:** prioritize reconstruction of corrupted dimensions
  - More important to **denoise** than to **restore what's already there**
  - For squared loss:

$$L_{2,\alpha}(\mathbf{x},\mathbf{z}) = \alpha\left(\sum_{j\in\mathcal{J}(\tilde{\mathbf{x}})}(\mathbf{x}_j-\mathbf{z}_j)^2\right)+\beta\left(\sum_{j\notin\mathcal{J}(\tilde{\mathbf{x}})}(\mathbf{x}_j-\mathbf{z}_j)^2\right)$$

Model	*basic*	*rot*	*bg-rand*
SVM$_{rbf}$	3.03±0.15	11.11±0.28	14.58±0.31
SAE-3	3.46±0.16	10.30±0.27	11.28±0.28
DBN-3	3.11±0.15	10.30±0.27	**6.73**±0.22
SDAE-3$_{MN}(\nu)$	2.84±0.15(10%)	9.53±0.26(25%)	10.30±0.27(40%)
SDAE-3$_{MN}(\nu)$ **+ emph**	**2.76**±0.14(25%)	10.36±0.27(25%)	9.69±0.26(40%)
SDAE-3$_{SP}(\nu)$	**2.66**±0.14(25%)	9.33±0.25(25%)	10.03±0.26(25%)
SDAE-3$_{SP}(\nu)$ **+ emph**	**2.48**±0.14(25%)	**8.76**±0.29(25%)	8.52±0.24(10%)
SDAE-3$_{GS}(\nu)$	**2.61**±0.14(0.1)	**8.86**±0.28(0.3)	11.73±0.28(0.1)

SVM Performance on Higher-Level SDAE Representations

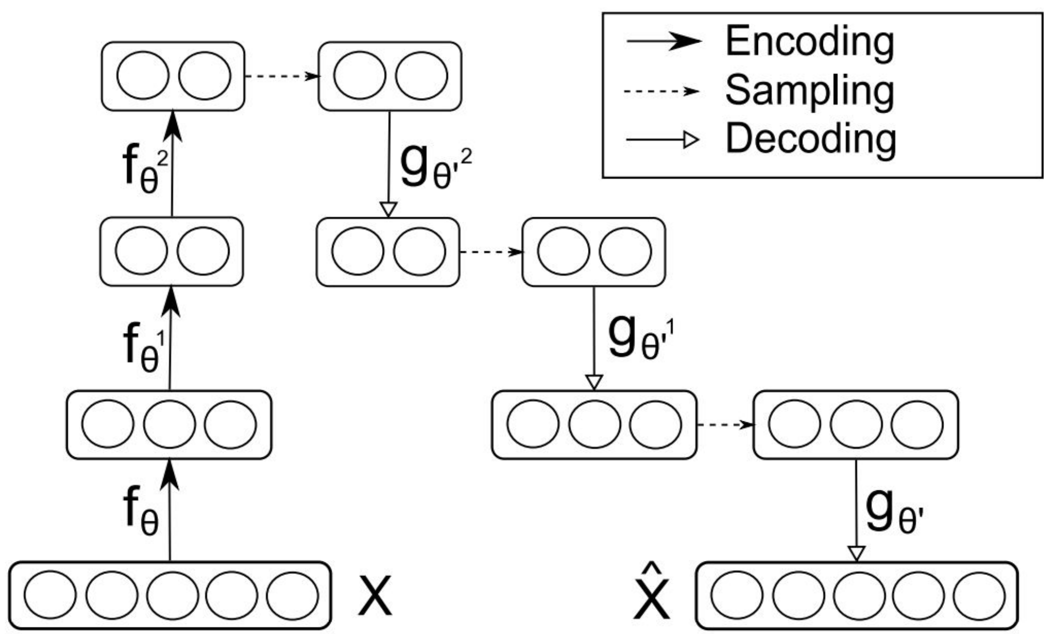
Train an SVM on the i^{th}-level encoding learned by an SDAE.

Data Set	SVM kernel	SVM_0	SVM_1	SVM_2	SVM_3
MNIST	linear	5.33±0.44	1.49 ±0.24	**1.24**±0.22	**1.2**±0.21
	rbf	1.40±0.23	**1.04**±0.20	**0.94**±0.19	**0.95**±0.19
basic	linear	7.32±0.23	3.43±0.16	**2.71**±0.14	**2.63**±0.14
	rbf	3.03±0.15	**2.59**±0.14	**2.55**±0.14	**2.57**±0.14
rot	linear	43.47±0.43	21.74±0.36	15.15±0.31	10.00±0.26
	rbf	11.11±0.28	**8.45**±0.24	**8.27**±0.24	**8.64**±0.25
bg-rand	linear	24.14±0.38	13.58±0.30	13.00±0.29	11.32±0.28
	rbf	14.58±0.31	11.00±0.27	**10.08**±0.26	**10.16**±0.26
bg-img	linear	25.08±0.38	16.72±0.33	20.73±0.36	**14.55**±0.31
	rbf	22.61±0.37	15.91±0.32	16.36±0.32	**14.06**±0.30
bg-img-rot	linear	63.53±0.42	50.44±0.44	50.26±0.44	42.07±0.43
	rbf	55.18±0.44	44.09±0.44	42.28±0.43	**39.07**±0.43
rect	linear	29.04±0.40	6.43±0.22	2.31±0.13	1.80±0.12
	rbf	2.15±0.13	2.19±0.13	1.46±0.11	**1.22**±0.10
rect-img	linear	49.64±0.44	23.12±0.37	23.01±0.37	**21.43**±0.36
	rbf	24.04±0.37	22.27±0.36	21.56±0.36	**20.98**±0.36
convex	linear	45.75±0.44	24.10±0.37	18.40±0.34	**18.06**±0.34
	rbf	19.13±0.34	18.09±0.34	**17.39**±0.33	**17.53**±0.33
tzanetakis	linear	20.72±2.51	12.51±2.05	7.95±1.68	**5.04**±1.36
	rbf	14.41±2.18	7.54±1.64	**5.20**±1.38	**4.13**±1.23

Generating Samples from the Input Distribution

Bottom-up representation inference:
Deterministically encode a random training input to its top-level representation.

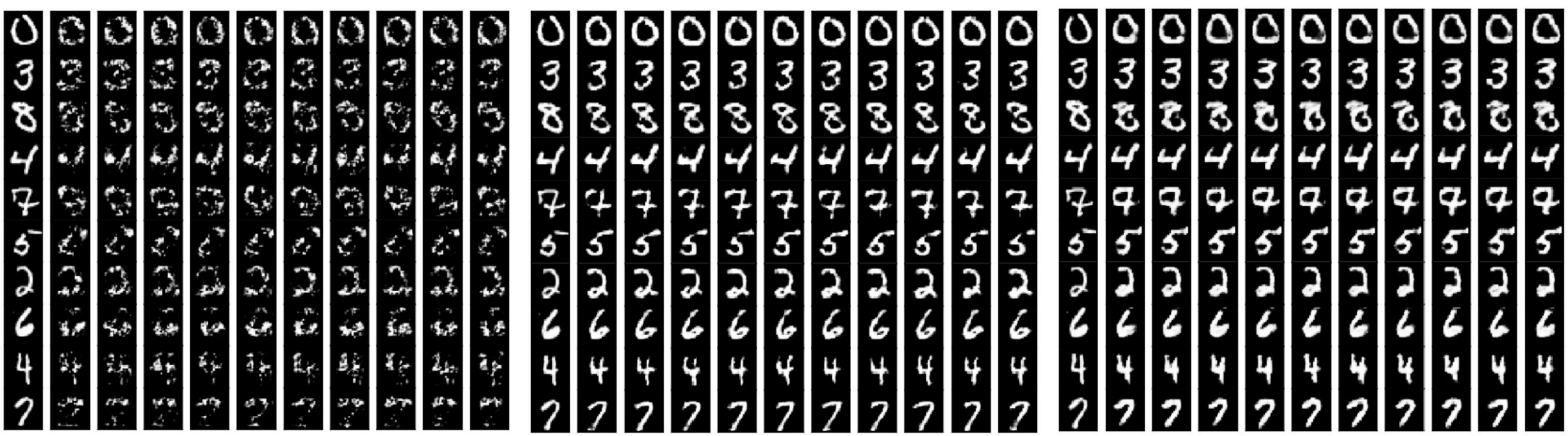
Top-down visible sample generation:
Alternately perform Bernoulli sampling and deterministic decoding.



Variability in Regenerated Samples

For a single fixed input, perform multiple bottom-up/**top-down** samplings.

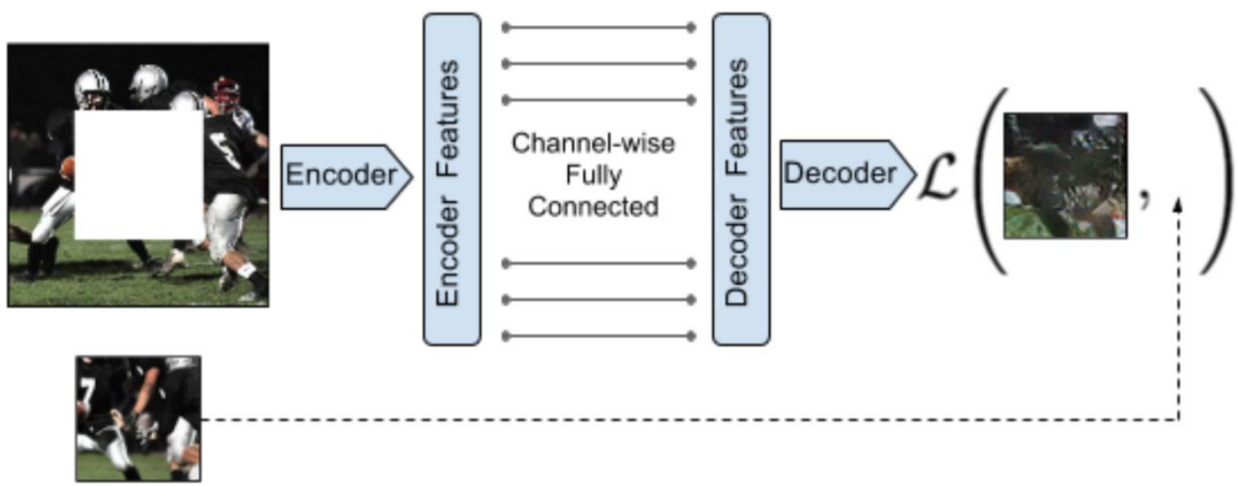
- SDAE replaces the missing hole in the 6, straightens the upper part of the 7



(a) SAE (b) SDAE (c) DBN

Legacy

- 3400+ citations, including dropout (randomly mask out neurons), variational autoencoders (explicitly model probability distributions)
- Laid groundwork for other self-supervised representation learning methods
  - Reconstruct missing parts of the data (inpainting)
  - Reconstruct one “view” of the data from another (colorization, split-brain autoencoders)



Context Encoders, Pathak et al. 2016