Autoencoders

CMPUT 366: Intelligent Systems

GBC 14.0-14.5

Lecture Outline

1. Recap
2. Unsupervised Learning
3. Autoencoders

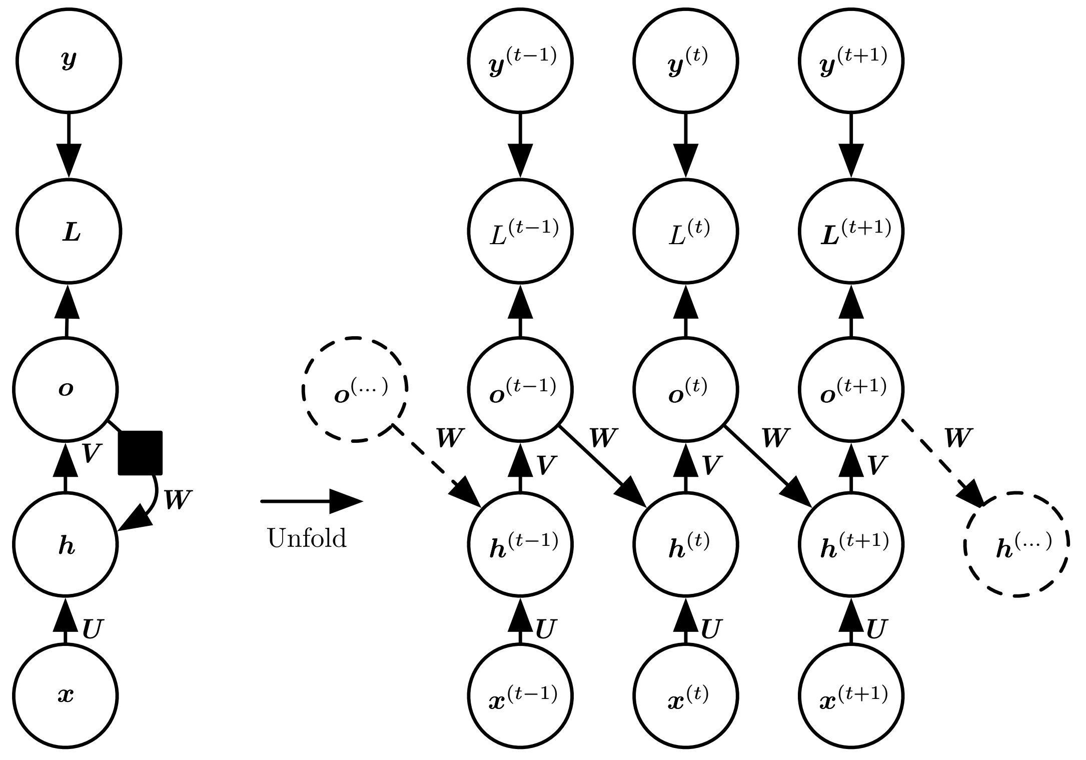
Recap:
Recurrent Neural Networks

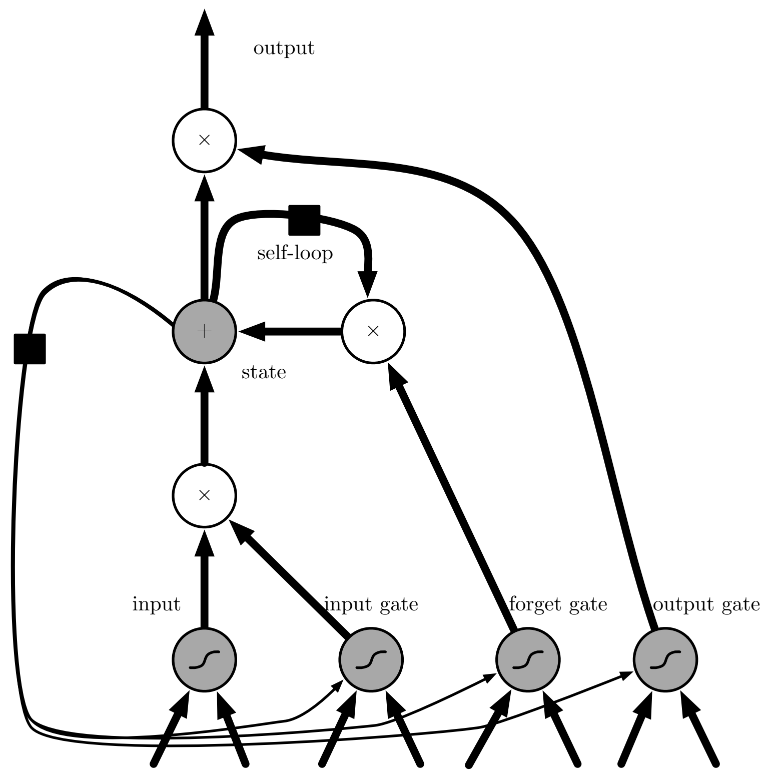
- Recurrent networks: Specialized architecture for **sequences**

- Process each element of the sequence **individually** using the **same parameters**

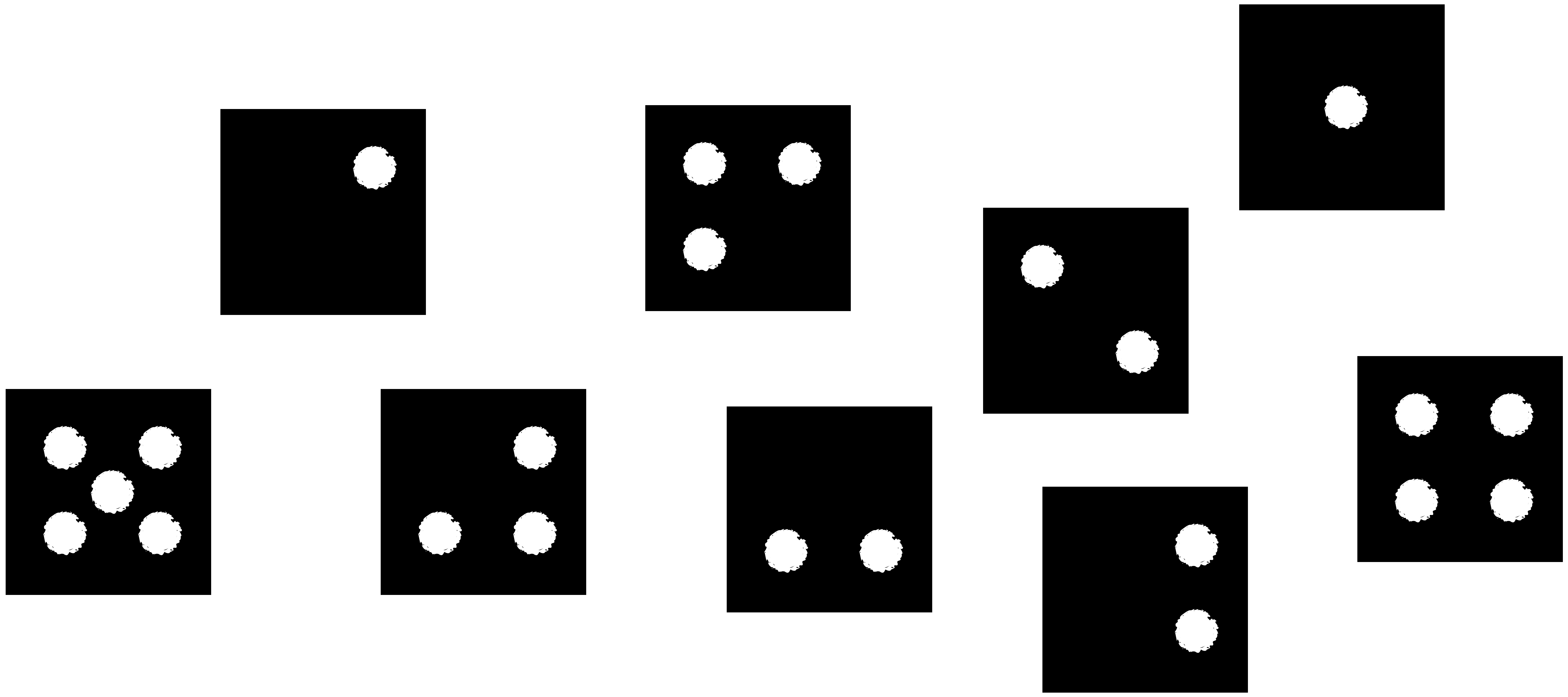
    - **Recurrent** hidden units: stage *t* output is input to stage *t*+1

- **Gated units** (e.g., LSTM) allow mappings to vary **dynamically**



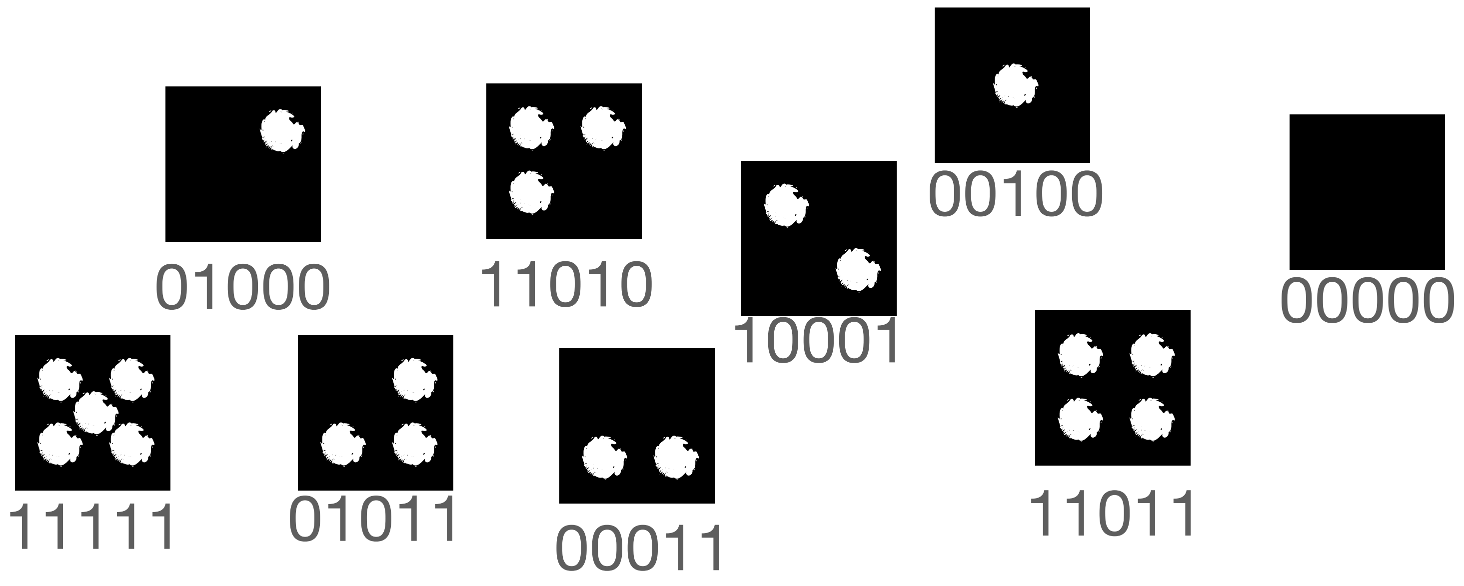


Example: Dots & Squares



- **Question:** How many pixels are in each of these 50x50 images?
- **Question:** How many numbers would you need to to write down to represent these images?

Compression



- We can often represent complicated data (e.g., images) in a very compressed form by exploiting **structure**

- **Question:** Why would this be valuable?

  1. *Compression:* Storing less information is better!

  2. *Learning features:* Rather than having to learn underlying structure for each task, learn it once, then input structured representation directly to supervised learner

Unsupervised Learning

Unsupervised learning is any learning algorithm that operates on **input features** but not **target features**

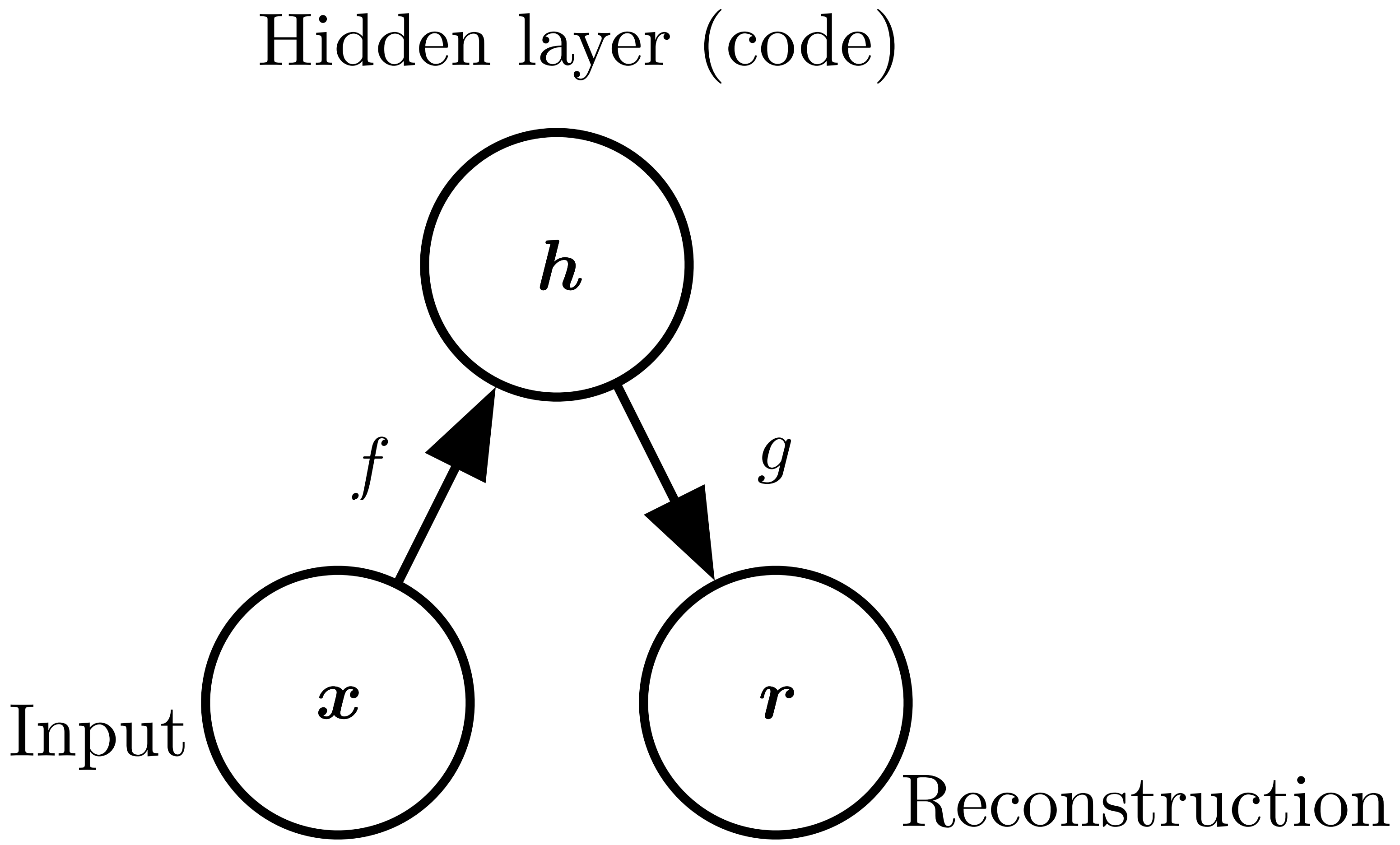
1. *Feature learning:* Learn underlying structure of examples

2. *Generative models:* Learn distribution over examples in order to synthesize plausible instances

3. *Dimensionality reduction:* Learn small representations

Autoencoders

Autoencoder: A neural network that is trained to attempt to copy its input to its output



- **Question:** Why would this be valuable?

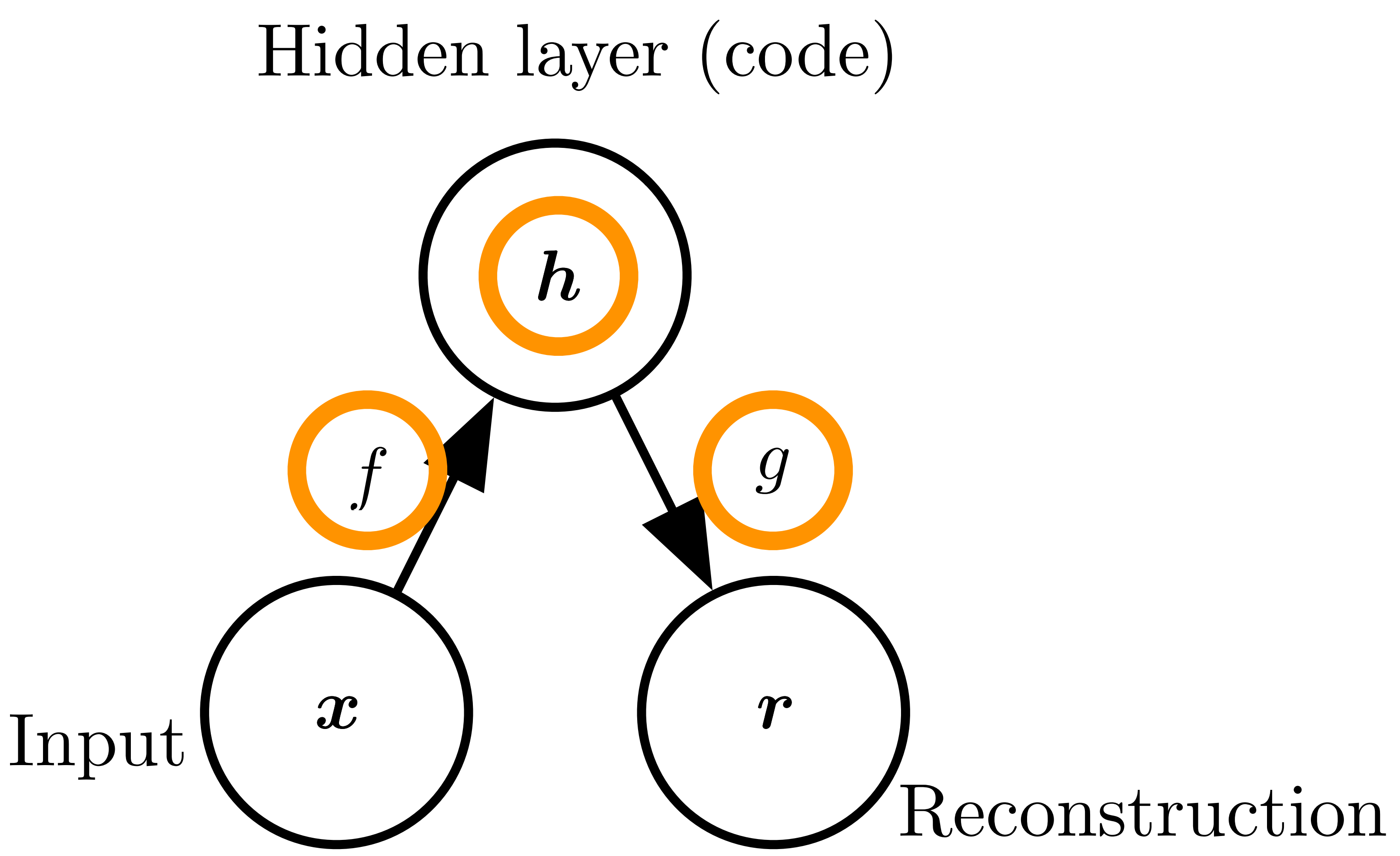
Autoencoders

- A network that is able to **exactly** copy its input to its output is actually kind of useless
- If we make it **impossible** for the network to make a direct copy, then it is forced to **prioritize** which aspects to copy
  - Can often learn **useful properties** of the data
  - E.g., "each image is a black square with 1-5 white dots"

Undercomplete Autoencoders

Question: How can we force the autoencoder to approximate instead of just making a trivial copy?

1. Make ***h*** have lower **dimension** than ***x***
   - E.g., only 5 hidden units for 50×50 image
2. Make *f* and/or *g* have low **capacity**
   - E.g., linear *g*



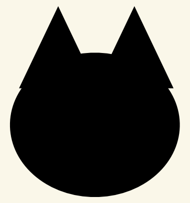
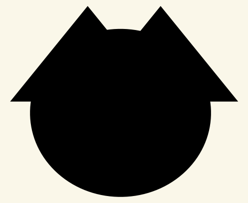
Regularized Autoencoders

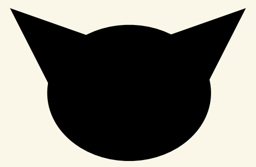
3. Add a term to the cost function **penalizing code complexity**

   - L2/Ridge regularization: Penalize **large** values
   - L1/Lasso regularization: Penalize **nonzero** values

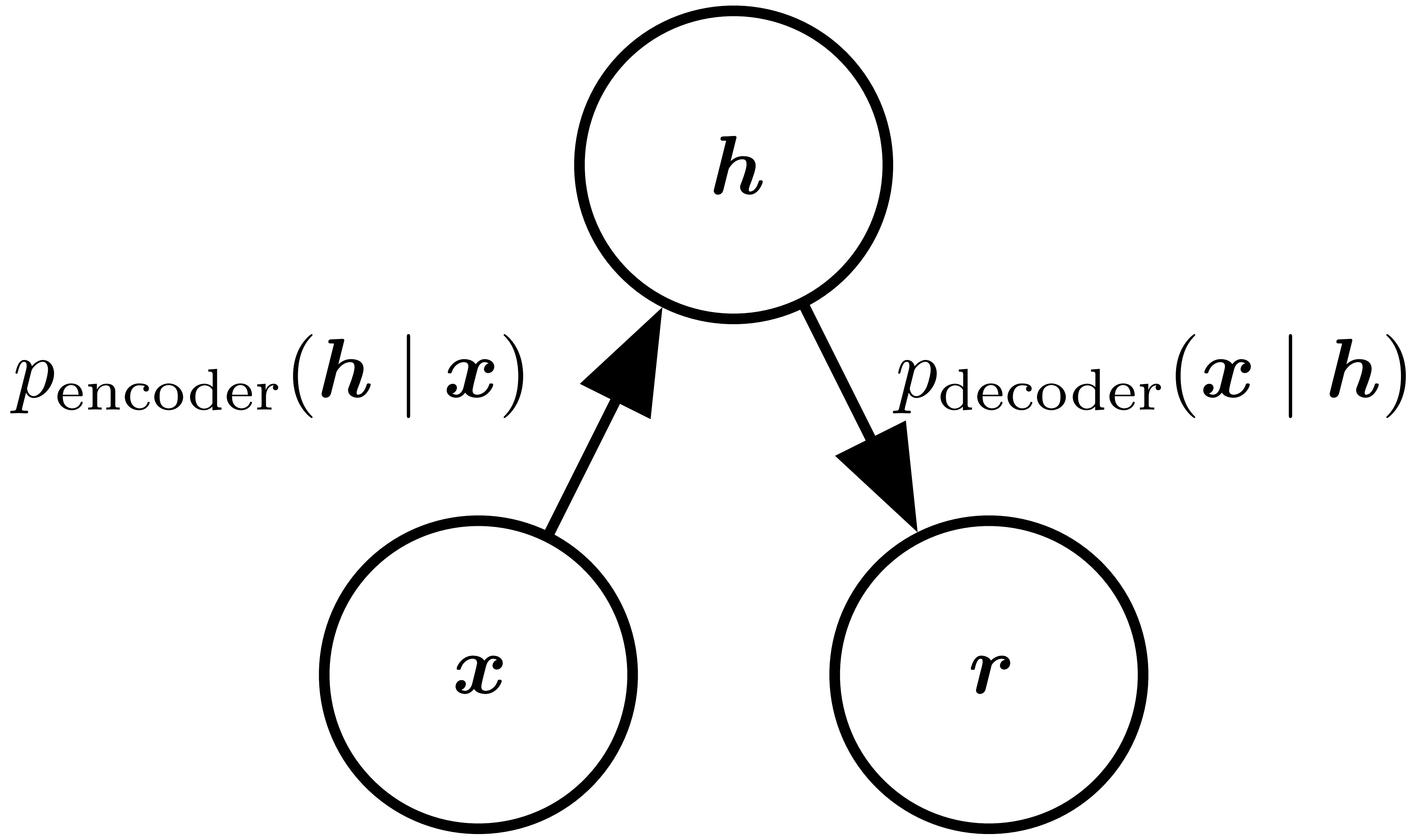
$$
\begin{aligned}
L(\theta_{\mathbf{f}}, \theta_{\mathbf{g}}) &= \ell(\mathbf{r}, \mathbf{x}) + \Omega(\mathbf{h}) \\
&= \ell(g(f(\mathbf{x}; \theta_f); \theta_g), \mathbf{x}) + \Omega(f(\mathbf{x}; \theta_f))
\end{aligned}
$$

Stochastic Outputs

X	Y_{cat}	Y_{dog}	Y_{panda}
	1	0	0
	0	1	0

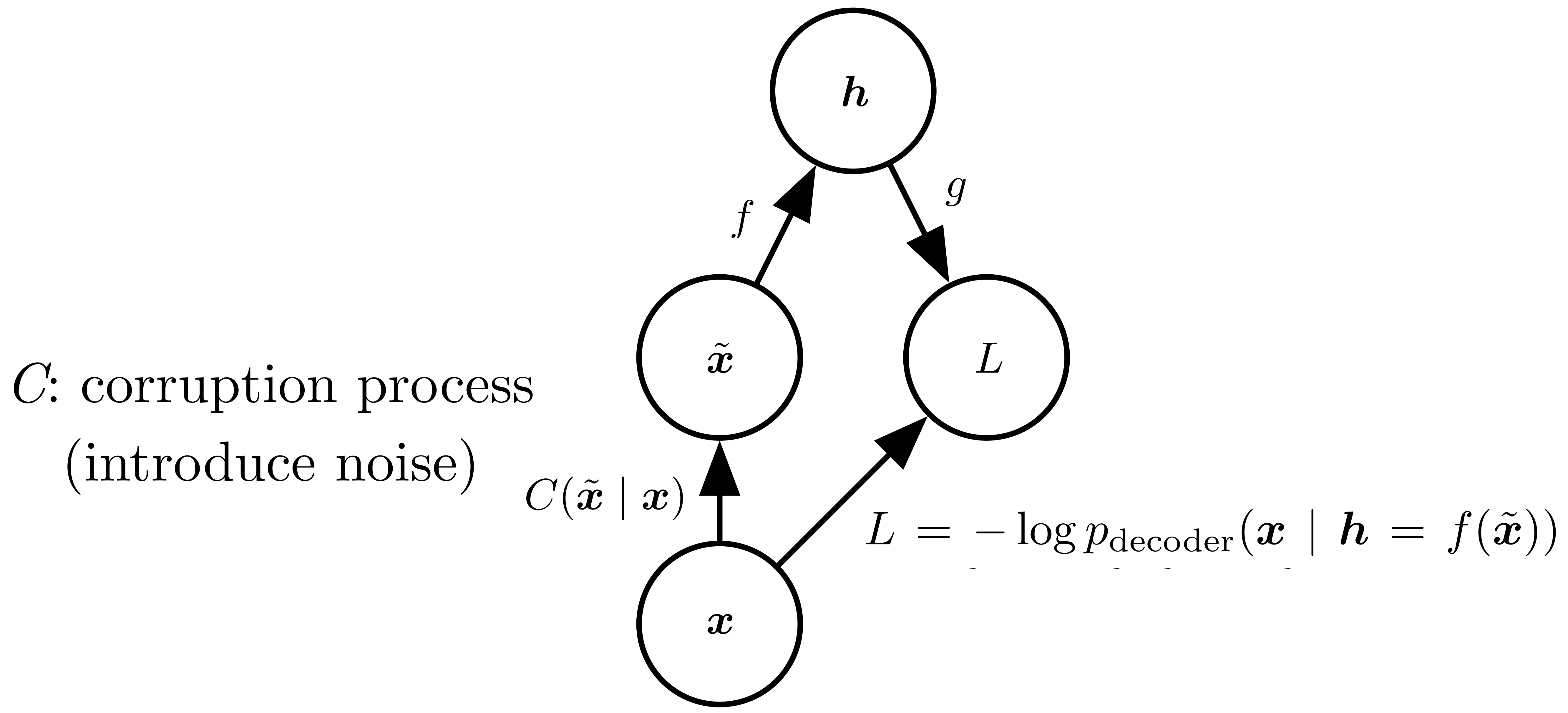
X	$\hat{Y}_{cat}$ $Pr(Y{=}cat \mid X)$	$\hat{Y}_{dog}$ $Pr(Y{=}dog \mid X)$	$\hat{Y}_{panda}$ $Pr(Y{=}panda \mid X)$
	0.50	0.45	0.05

Stochastic Autoencoders



- Decoder gives **distribution** over **inputs** given hidden layer
- Encoder gives **distribution** over **hidden layer** given inputs

Denoising Autoencoders



4. Train on **noisy version** $\tilde{\mathbf{x}}$ of the input $\mathbf{x}$

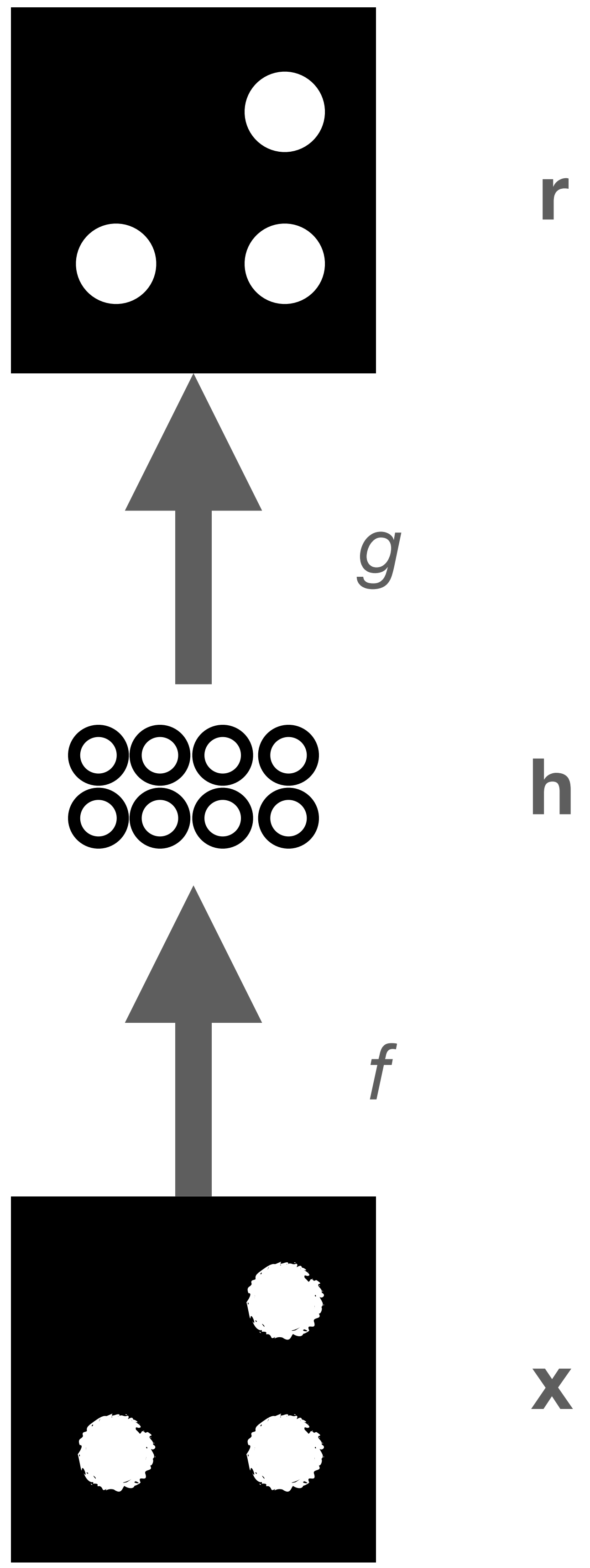
    - Loss computed by how well **original** $\mathbf{x}$ is reconstructed from **corrupted** $\tilde{\mathbf{x}}$

Representing Distributions

Question: What does the **output layer** look like in a stochastic autoencoder?

- Indicator variables often won't work, since the input features are usually unstructured and high-dimensional

- Instead, usually learn mean and variance of a **Gaussian** for each output unit



Summary

- Neural networks: Not just for supervised learning!
- **Autoencoders:** Input is $\mathbf{x}$, output is $\mathbf{r}$, loss is $\ell(\mathbf{x},\mathbf{r})$
- Hidden layer $\mathbf{h}$ can be interpreted as a **code** that captures the most **important properties** of the inputs
- To avoid trivial copying:

1. *Undercomplete autoencoders*:
   Small **dimension** $\mathbf{h}$, small **capacity** encoder/decoder
2. *Regularization:* **Penalize** complex codes
3. *Denoising:* Train on **corrupted** versions of the inputs