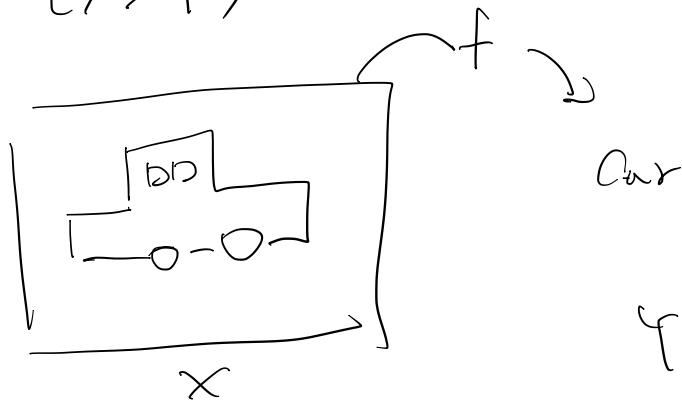


# Variational Autoencoder (VAE)

# Supervised learning

Data:  $(X, Y)$



Goal:  $f(x) \approx Y$

---

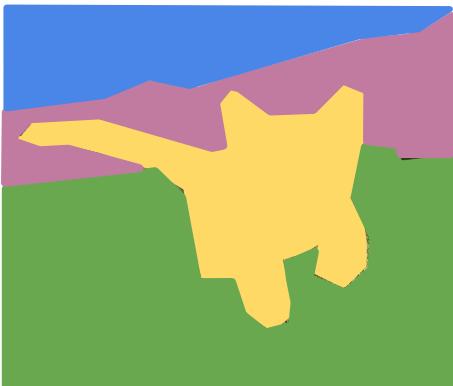
$$\min_{\beta} l(f(x; \beta), Y)$$

# Supervised learning



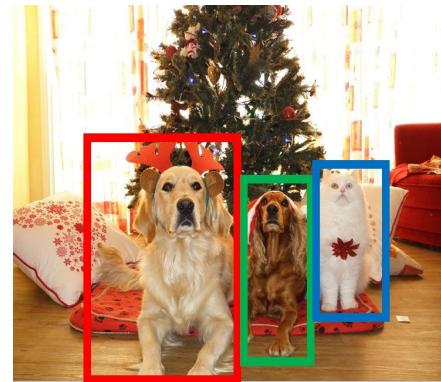
→ Cat

Classification



GRASS, CAT,  
TREE, SKY

Semantic Segmentation



DOG, DOG, CAT

Object Detection



*A cat sitting on a suitcase on the floor*

Image captioning

# Unsupervised learning

Data:  $X$

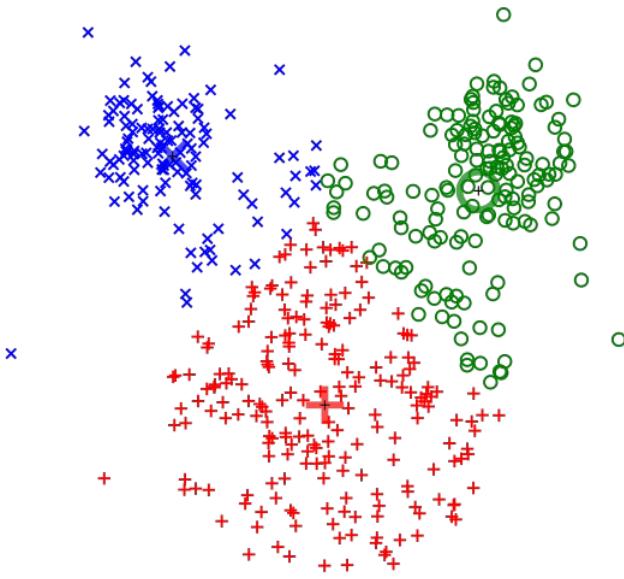
Goal: Learn underlying "hidden"  
structure in  $X$

$\Rightarrow$  Anomaly detection.

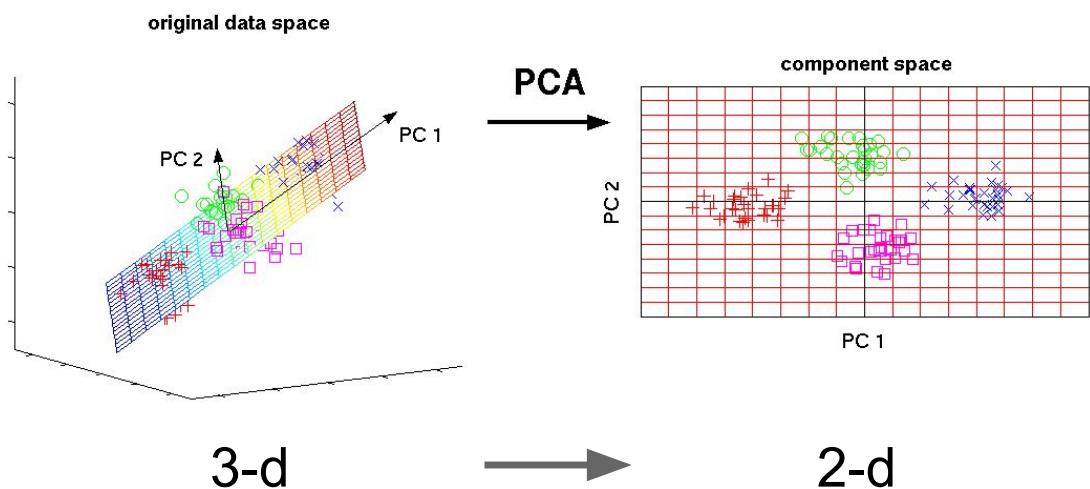
noise reduction

data compression.

# Unsupervised learning

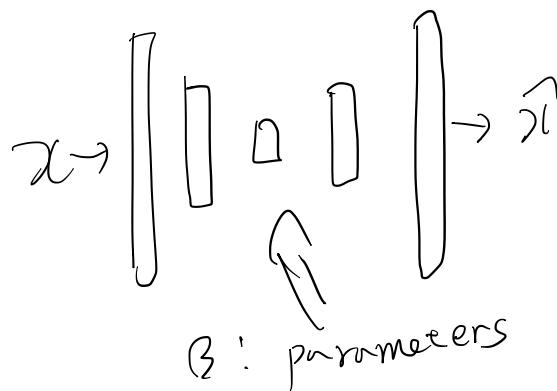
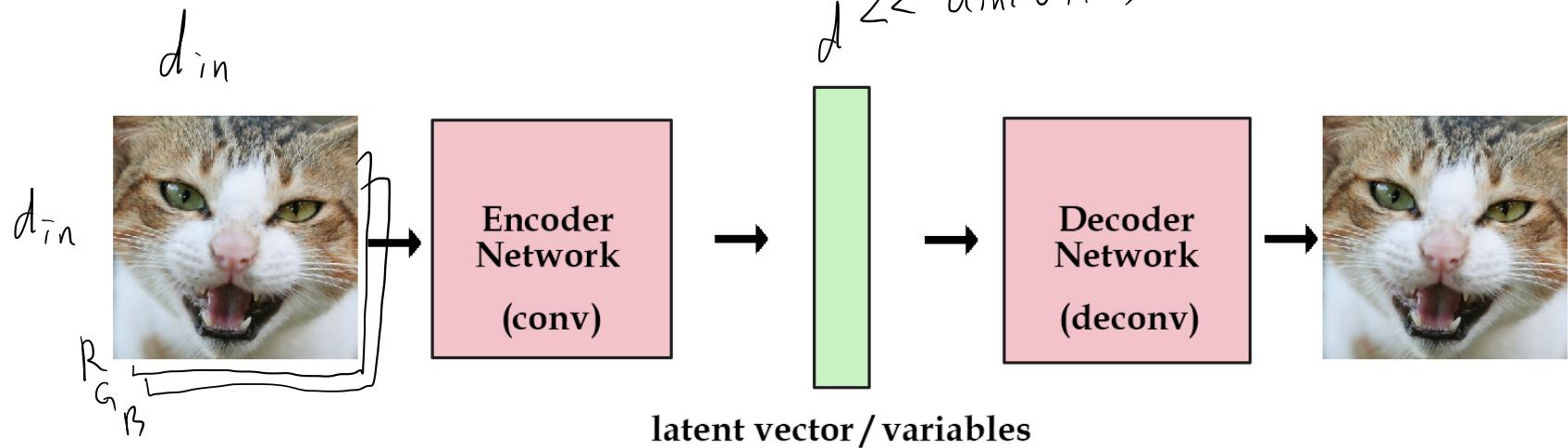


K-means clustering



Principal Component Analysis  
(Dimensionality reduction)

# Autoencoder



$$\arg \min_{\beta} \sum_i l(x_i, \hat{x}_i)$$
$$\|x_i - \hat{x}_i\|_2$$

```
class autoencoder(nn.Module):
    def __init__(self):
        super(autoencoder, self).__init__()
        self.encoder = nn.Sequential(
            nn.Conv2d(1, 16, 3, stride=3, padding=1), # b, 16, 10, 10
            nn.ReLU(True),
            nn.MaxPool2d(2, stride=2), # b, 16, 5, 5
            nn.Conv2d(16, 8, 3, stride=2, padding=1), # b, 8, 3, 3
            nn.ReLU(True),
            nn.MaxPool2d(2, stride=1) # b, 8, 2, 2
        )
        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(8, 16, 3, stride=2), # b, 16, 5, 5
            nn.ReLU(True),
            nn.ConvTranspose2d(16, 8, 5, stride=3, padding=1), # b, 8, 15, 15
            nn.ReLU(True),
            nn.ConvTranspose2d(8, 1, 2, stride=2, padding=1), # b, 1, 28, 28
            nn.Tanh()
        )

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x
```

```
model = autoencoder().cuda()
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate,
                             weight_decay=1e-5)

for epoch in range(num_epochs):
    for data in dataloader:
        img, _ = data
        img = Variable(img).cuda()
        # =====forward=====
        output = model(img)
        loss = criterion(output, img)
        # =====backward=====
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

# Transposed convolution

A guide to convolution arithmetic for deep  
learning

Vincent Dumoulin<sup>1</sup>★ and Francesco Visin<sup>2</sup>★†

★MILA, Université de Montréal

†AIRLab, Politecnico di Milano

# Transposed convolution

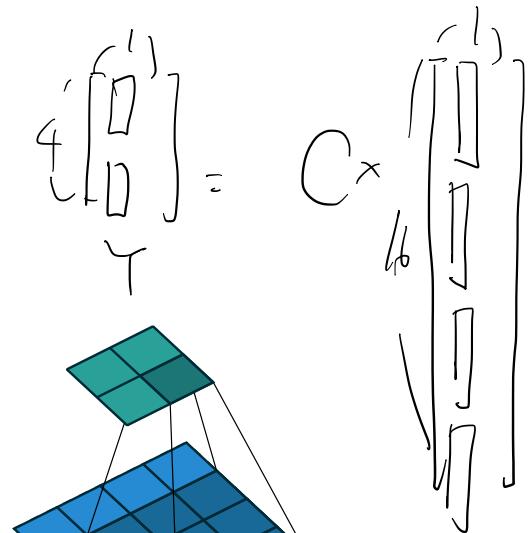
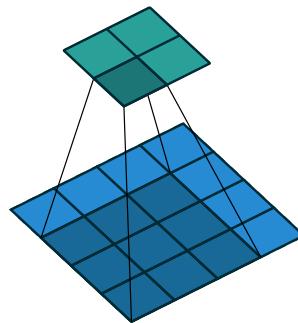
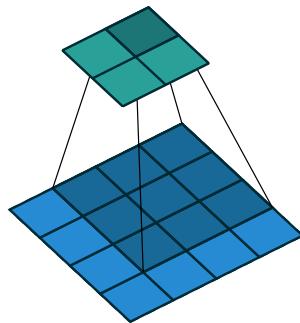
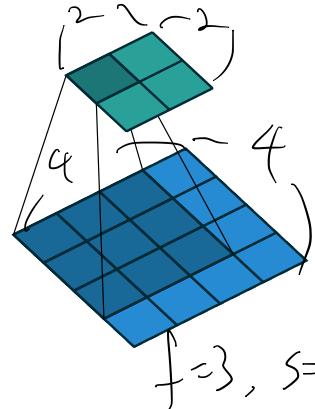


Figure 2.1: (No padding, unit strides) Convolving a  $3 \times 3$  kernel over a  $4 \times 4$  input using unit strides (i.e.,  $i = 4$ ,  $k = 3$ ,  $s = 1$  and  $p = 0$ ).

$$C \in \mathbb{R}^{4 \times 16}$$

$$C \times X$$

$$C^T \in \mathbb{R}^{16 \times 4}$$

$C =$

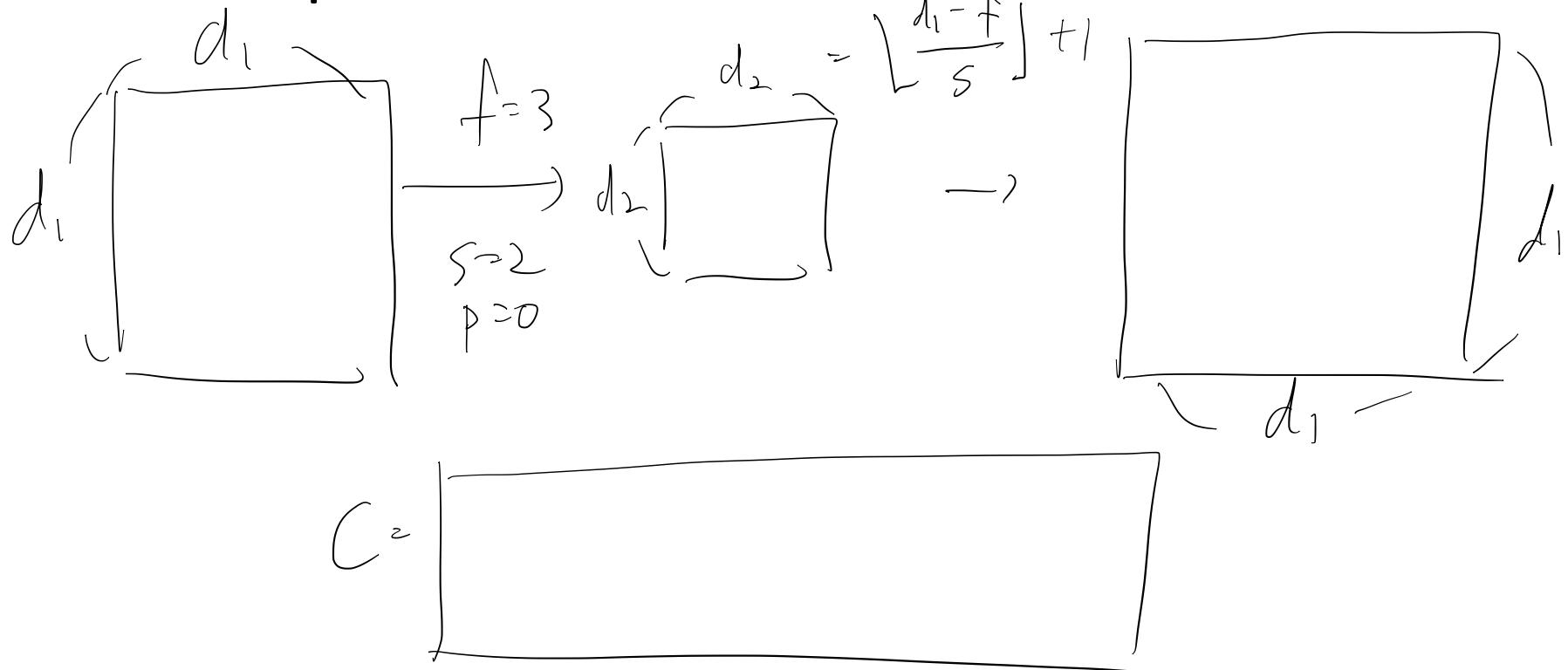
$$\begin{pmatrix} w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} \end{pmatrix}$$

$$2 \times 2 \Rightarrow 4 \times 4$$

$$| \quad 4 \times 4 \Rightarrow 2 \times 2$$

$\begin{cases} \text{Conv} \\ \text{Trans} \\ \text{Conv.} \end{cases}$

# Transposed convolution



# Transposed convolution (example)

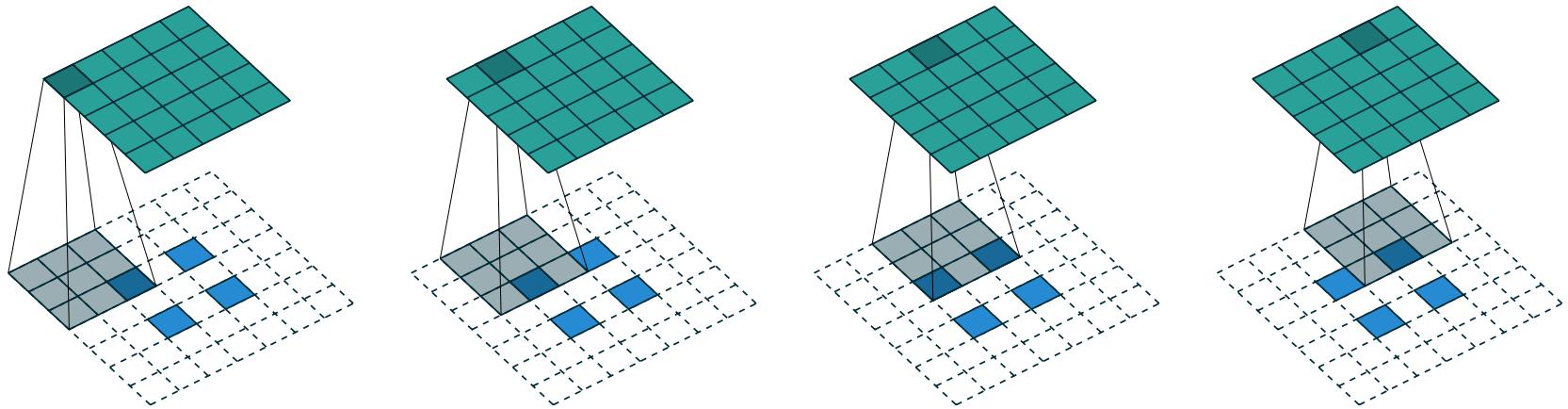
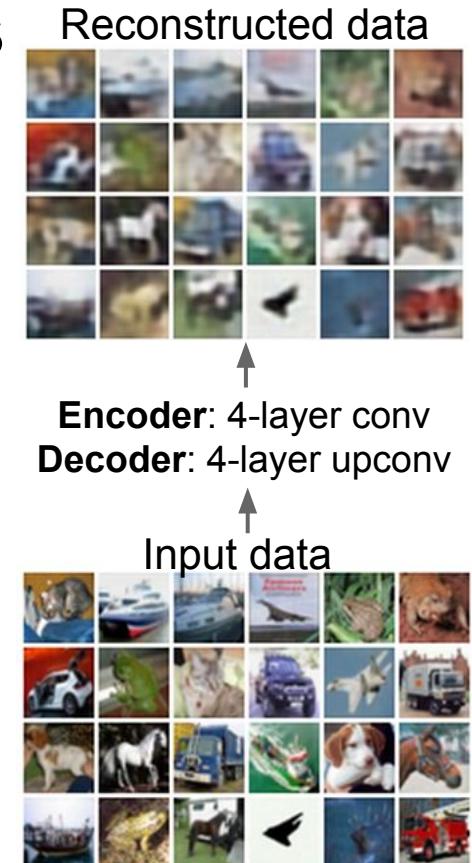
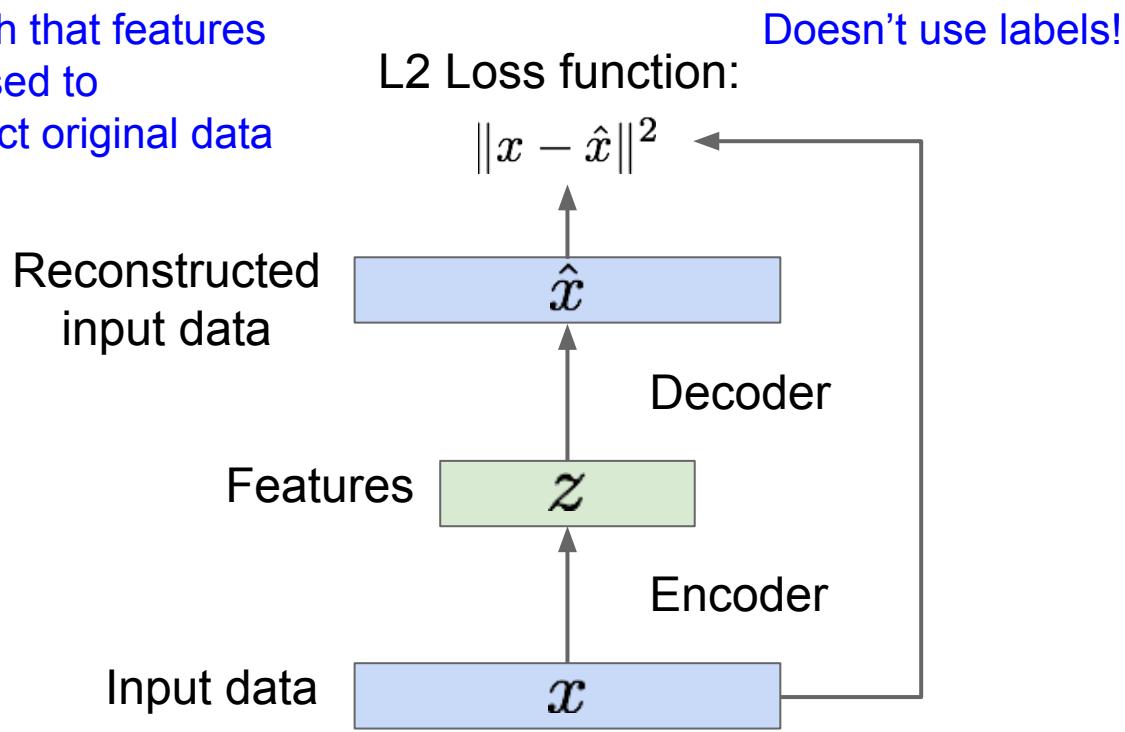


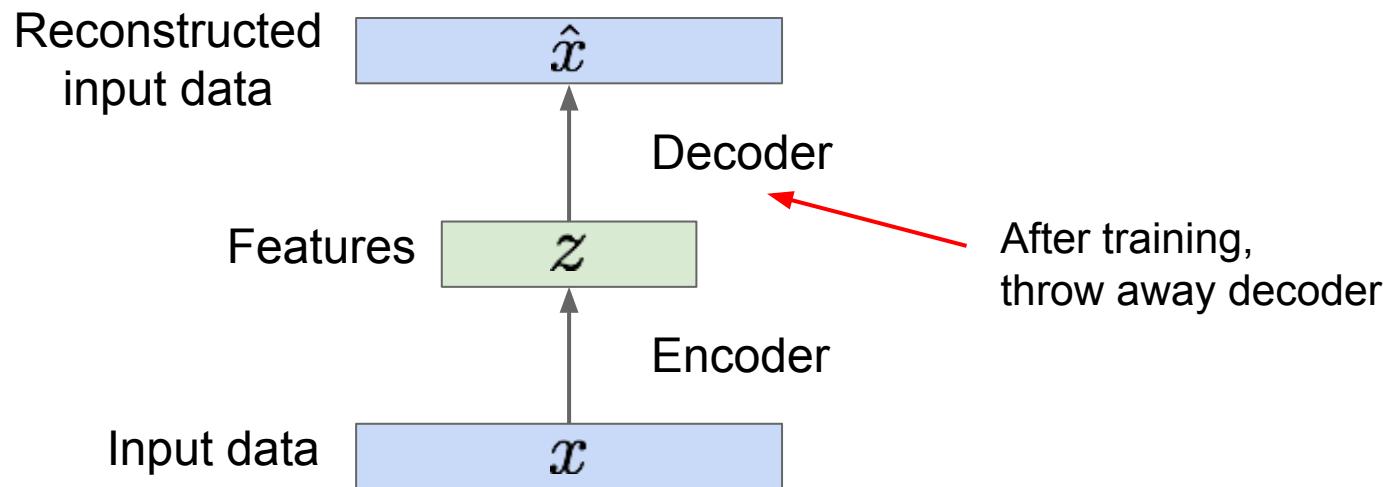
Figure 4.5: The transpose of convolving a  $3 \times 3$  kernel over a  $5 \times 5$  input using  $2 \times 2$  strides (i.e.,  $i = 5$ ,  $k = 3$ ,  $s = 2$  and  $p = 0$ ). It is equivalent to convolving a  $3 \times 3$  kernel over a  $2 \times 2$  input (with 1 zero inserted between inputs) padded with a  $2 \times 2$  border of zeros using unit strides (i.e.,  $i' = 2$ ,  $\tilde{i}' = 3$ ,  $k' = k$ ,  $s' = 1$  and  $p' = 2$ ).

# Some background first: Autoencoders

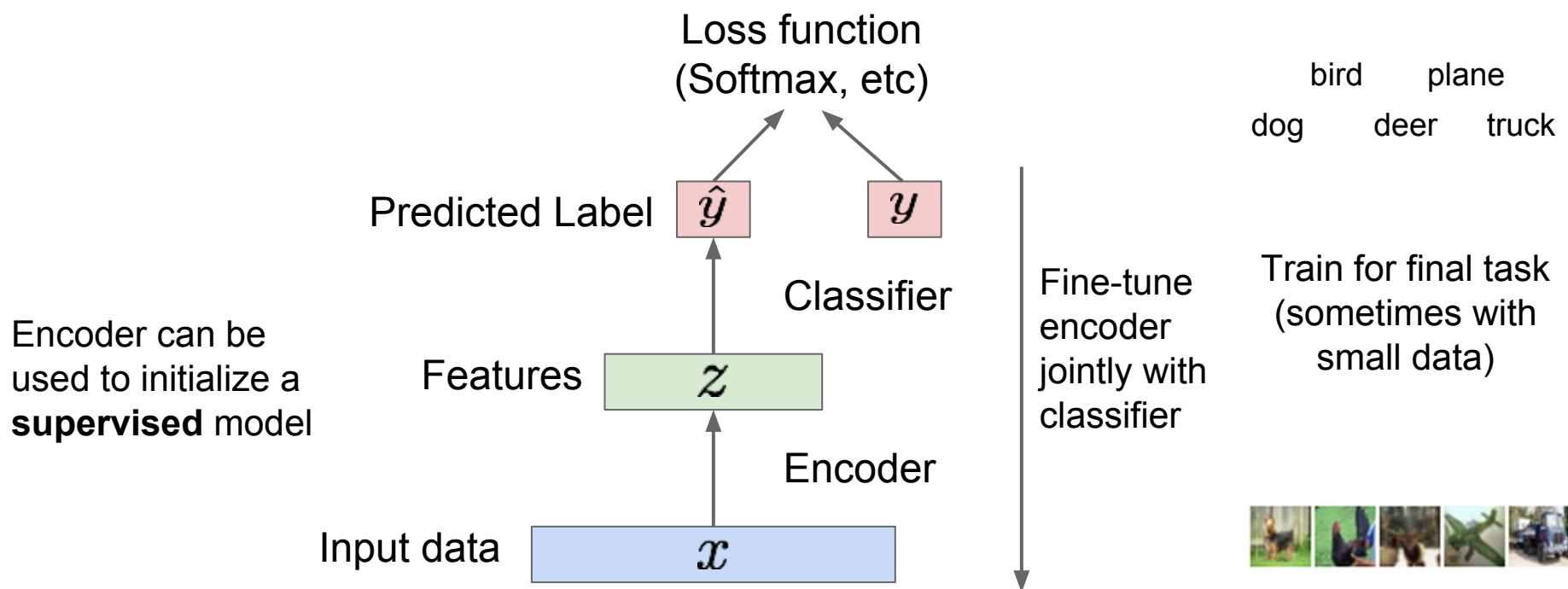
Train such that features  
can be used to  
reconstruct original data



# Some background first: Autoencoders



# Some background first: Autoencoders



VAE  
GAN

# Variational Autoencoder

---

## Auto-Encoding Variational Bayes

---

**Diederik P. Kingma**  
Machine Learning Group  
Universiteit van Amsterdam  
[dpkingma@gmail.com](mailto:dpkingma@gmail.com)

**Max Welling**  
Machine Learning Group  
Universiteit van Amsterdam  
[welling.max@gmail.com](mailto:welling.max@gmail.com)

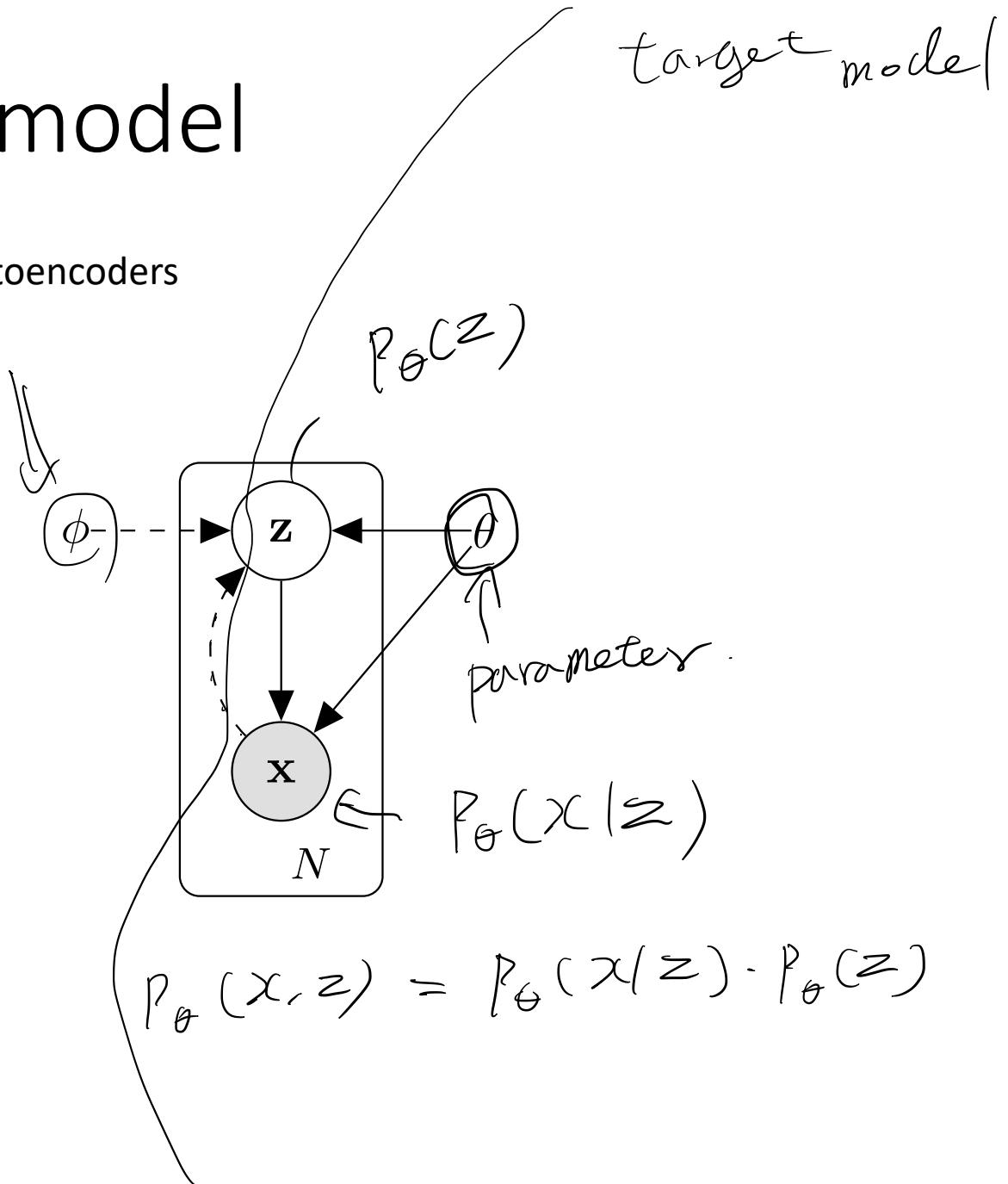
# Graphical model

Probabilistic spin on autoencoders

Variation

$$\boxed{P_\theta(z|x)}$$

$P_\theta(x)$



# Why we need VAE?

$$\min \text{KL}(P_{\text{data}}(z) \parallel \int p_\theta(x|z) P_\theta(z) dz)$$

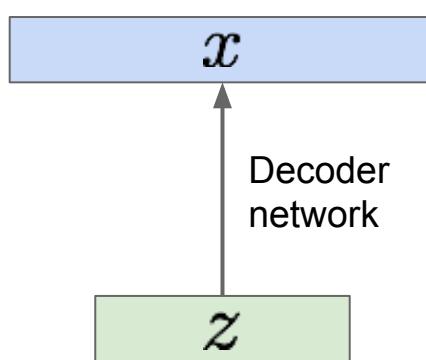
$$\min - \int p_{\text{data}}(x) \log \int p_\theta(x|z) P_\theta(z) dz$$

Sample from true conditional

$$p_{\theta^*}(x | z^{(i)})$$

Sample from true prior

$$p_{\theta^*}(z)$$



We want to estimate the true parameters  $\theta^*$  of this generative model.

How to train the model?

Remember strategy for training generative models from FVBMs. Learn model parameters to maximize likelihood of training data

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

Q: What is the problem with this?

Intractable!

$$p_\theta(z) \sim \mathcal{N}(0, I)$$

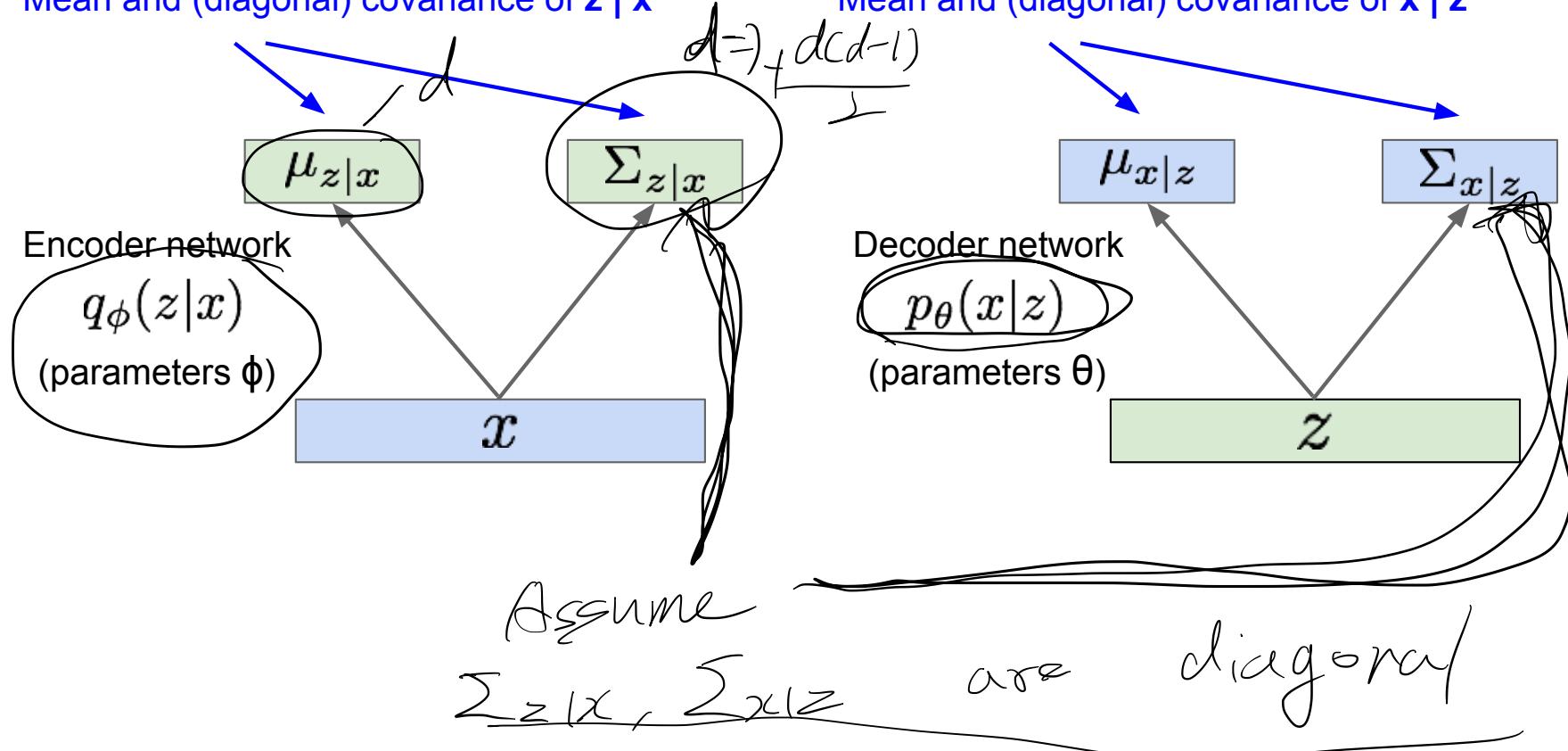
$$\min - \sum_i \log \int p_\theta(x_i | z) P_\theta(z) dz$$

# Variational Autoencoder

$$z \in \mathbb{R}^d$$

Since we're modeling probabilistic generation of data, encoder and decoder networks are probabilistic

Mean and (diagonal) covariance of  $z | x$



# Evidence Lower Bound (ELBO)

$$\log P_{\theta}(x_i) \quad \hat{=} \quad \underbrace{L(\theta, \phi, x_i)}_{\text{!}}$$
$$\int \frac{P_{\theta}(x_i|z) P_{\theta}(z) dz}{P_{\theta}(x_i, z)} \quad \left( \begin{array}{c} \text{Encoder} \\ \text{structure.} \end{array} \right) \quad \left[ \begin{array}{l} -\log q_{\phi}(z|x_i) + \\ \log P_{\theta}(z, z) \end{array} \right]$$
$$x_i \quad \arg \max_{\theta} \sum_i \log P_{\theta}(x_i) \quad \left| \arg \max_{\theta, \phi} \sum_{i=1}^n L(\theta, \phi, x_i) \right.$$

# ELBO and KL divergence

$$\text{iff } q_{\phi}(z|x_i) = p_{\phi}(z|x_i) \forall z, x_i$$

$$\log P_{\theta}(x_i) = \underbrace{\text{KL}(q_{\phi}(z|x_i) || P_{\theta}(z|x_i))}_{\geq 0} + \underbrace{f(\theta, \phi, x_i)}_{\text{ELBO}}$$

KL

$$E_{q_{\phi}(z|x_i)} \left[ \log \frac{q_{\phi}(z|x_i)}{P_{\theta}(z|x_i)} \right] + E_{q_{\phi}(z|x_i)} \left[ -\log q_{\phi}(z|x_i) + \log P_{\theta}(x_i, z) \right]$$

$$= E_{q_{\phi}(z|x_i)} \left[ -\log \frac{P_{\theta}(z|x_i)}{P_{\theta}(x_i, z)} + \log P_{\theta}(x_i, z) \right]$$

$$= E_{q_{\phi}(z|x_i)} \left[ \log P_{\theta}(x_i) \right] = \log P_{\theta}(x_i)$$

# Tractable ELBO

$$\mathcal{L}(\theta, \phi, x_i) = \mathbb{E}_{q_\phi(z|x_i)} \left[ -\log q_\phi(z|x_i) + \log p_\theta(x_i, z) \right]$$

$$= \mathbb{E}_{\theta} \left[ -\log \frac{q_{\theta}(z|x_i)}{p_{\theta}(z)} + \log \frac{p_{\theta}(x_i, z)}{p_{\theta}(z)} \right]$$

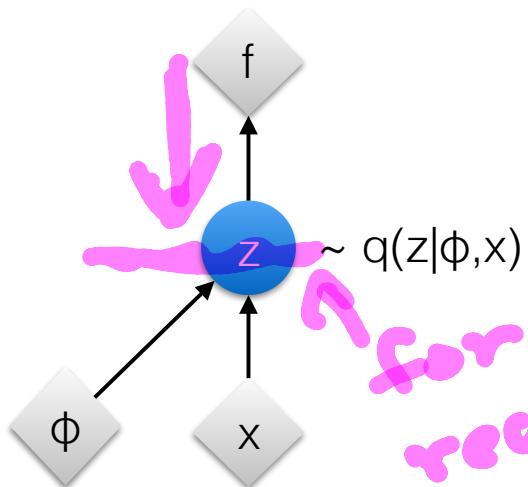
$$= - \text{KL} \left( q_{\phi}(z|x_i) \parallel p_{\theta}(z) \right)$$

← Okay  
because they are normal

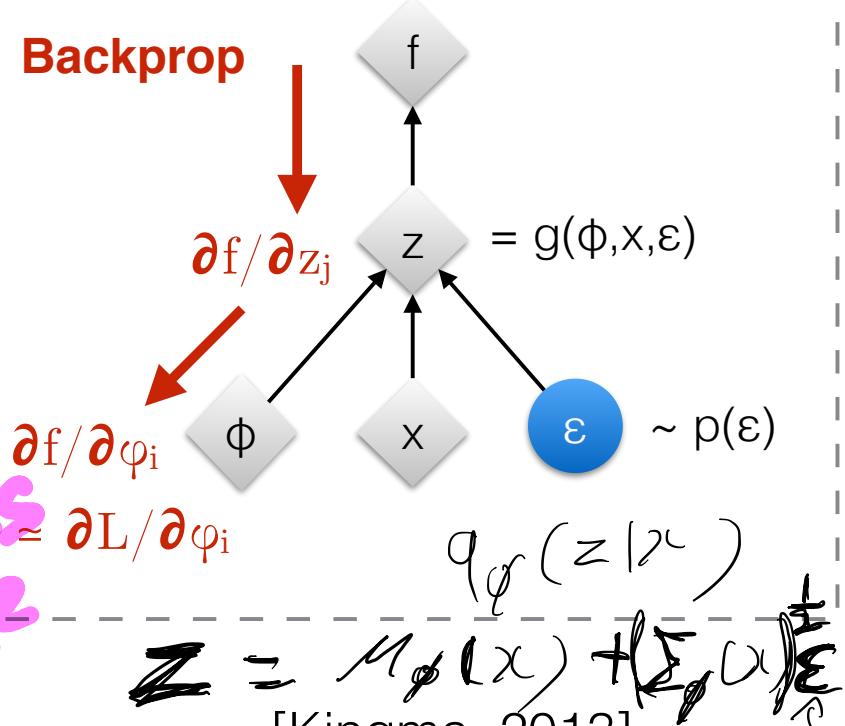
The diagram illustrates an autoencoder architecture. At the bottom, a wavy black line represents the input signal. Above it, a thick pink line represents the latent variable  $z$ . The input signal is processed by an "encoder" (indicated by a downward arrow) to produce the latent variable  $z$ . The latent variable  $z$  is then processed by a "decoder" (indicated by an upward arrow) to reconstruct the original input signal.

# Reparameterization Trick

Original form



Reparameterised form



: Deterministic node



: Random node

[Kingma, 2013]

[Bengio, 2013]

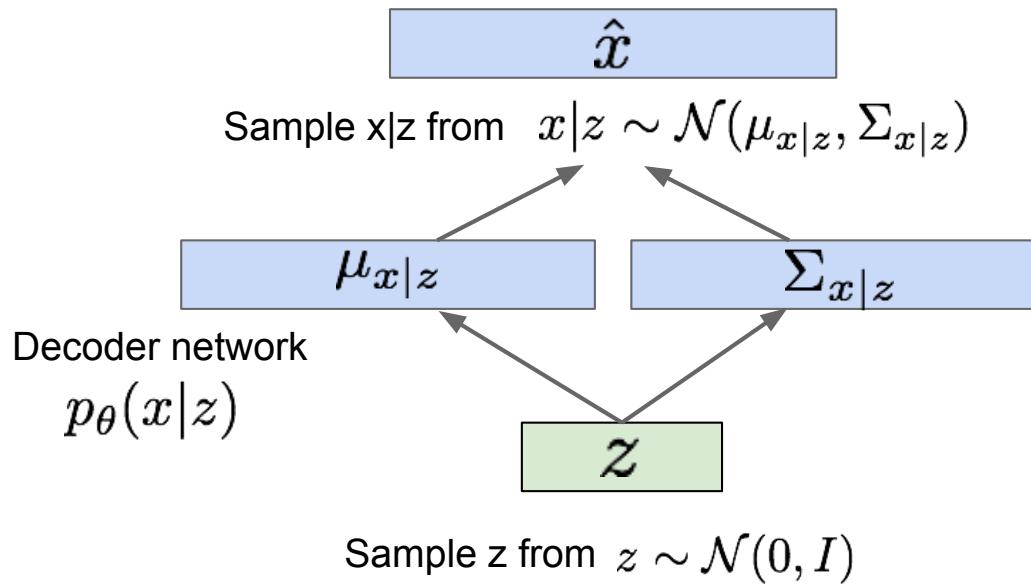
[Kingma and Welling 2014]

[Rezende et al 2014]

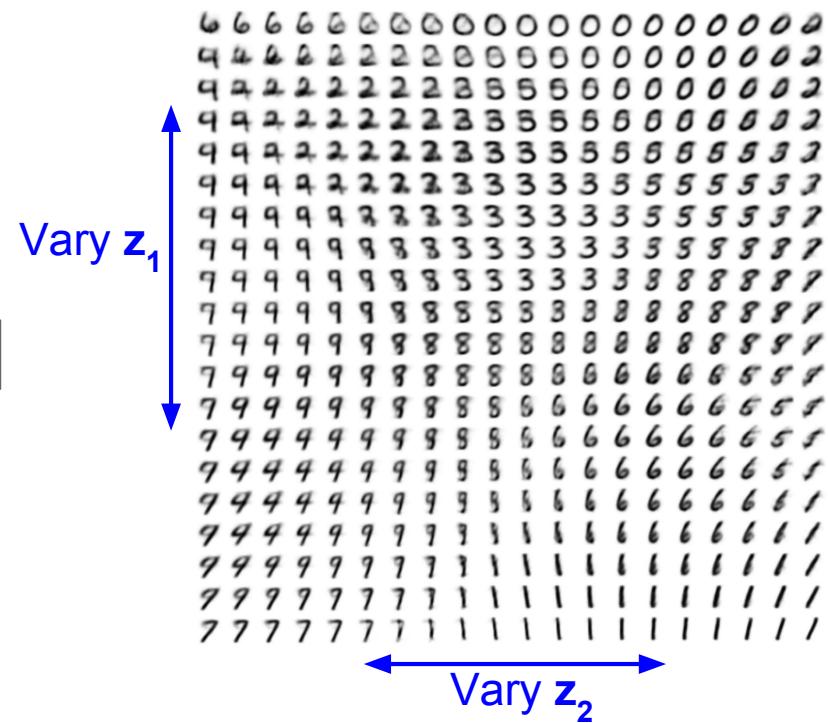
$N(0, I)$

# Variational Autoencoders: Generating Data!

Use decoder network. Now sample z from prior!



Data manifold for 2-d  $z$



Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

# Variational Autoencoders: Generating Data!

Diagonal prior on  $\mathbf{z}$   
=> independent  
latent variables

Different  
dimensions of  $\mathbf{z}$   
encode  
interpretable factors  
of variation

Also good feature representation that  
can be computed using  $q_{\phi}(z|x)$ !

Degree of smile

Vary  $z_1$



Vary  $z_2$

Head pose

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014