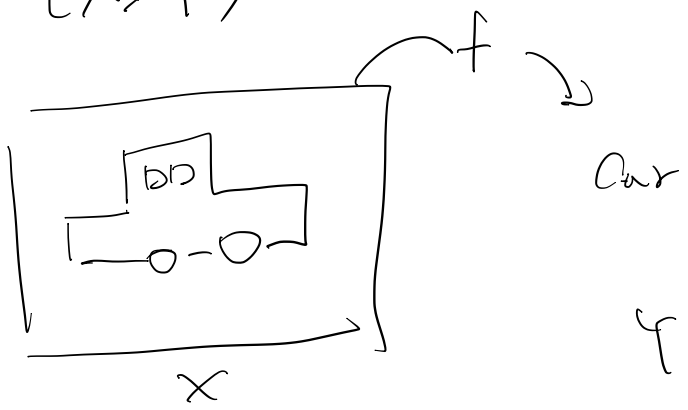
Variational Autoencoder (VAE)

Supervised learning

Data: (X, Y)



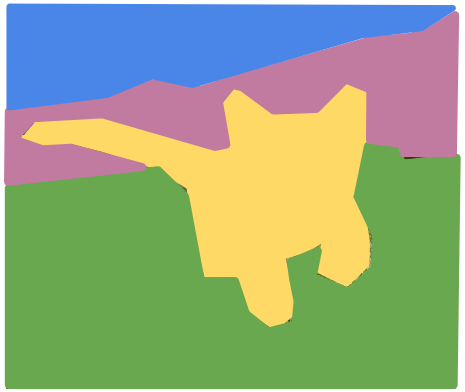
Goal: $f(x) \approx Y$

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

Supervised learning

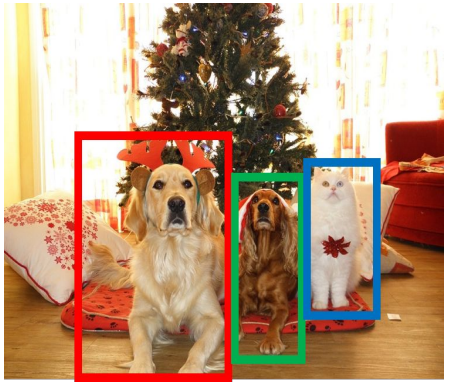
Cat

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

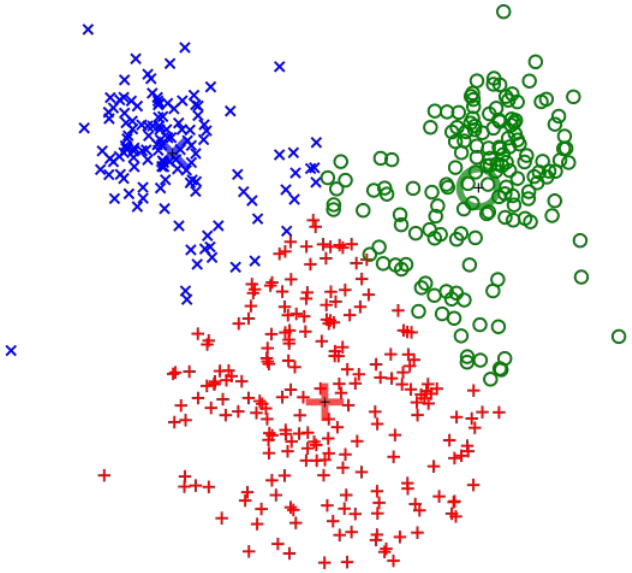
Goal: Learn underlying "hidden" structure in X

⇒ Anomaly detection.

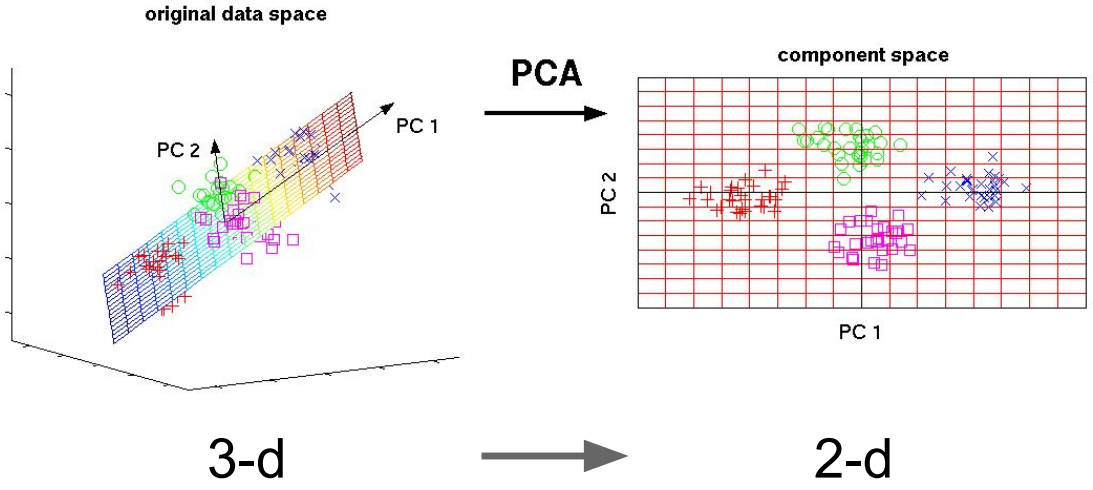
noise reduction

data compression.

Unsupervised learning

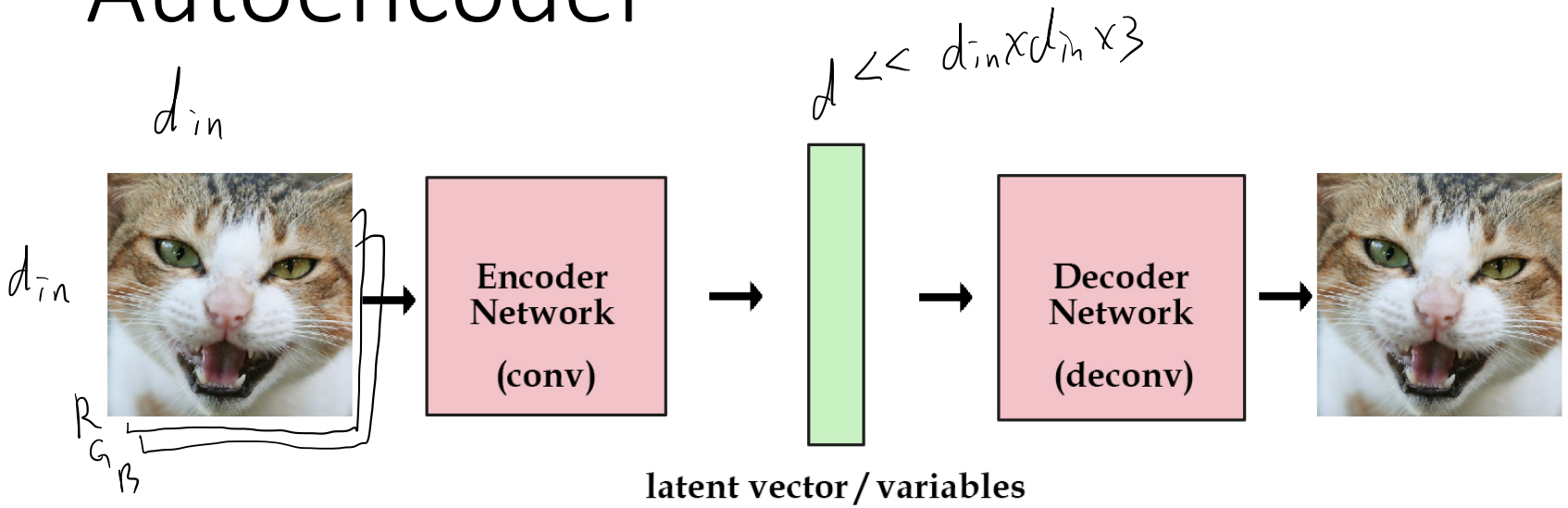


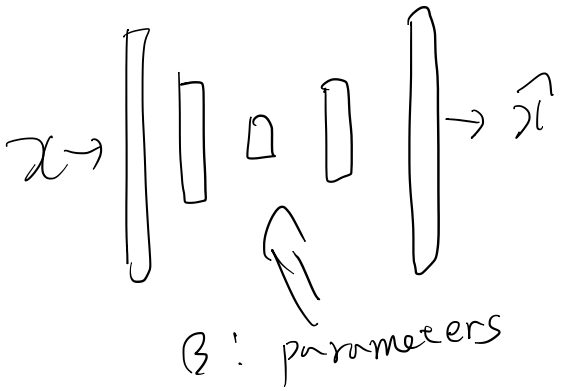
K-means clustering



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[1★] and Francesco Visin[2★†]

[★]MILA, Université de Montréal
[†]AIRLab, Politecnico di Milano

Transposed convolution

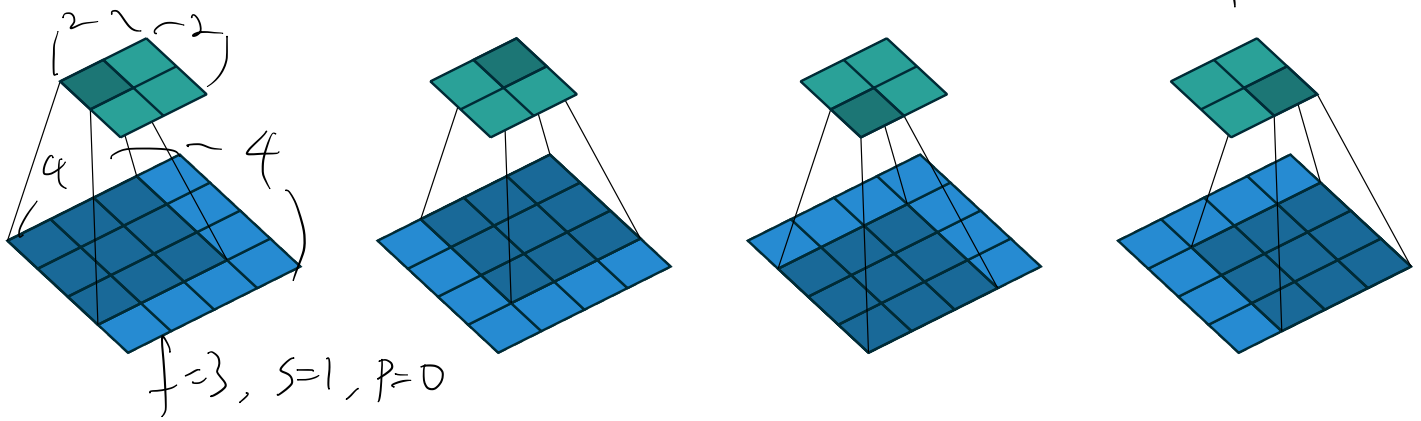
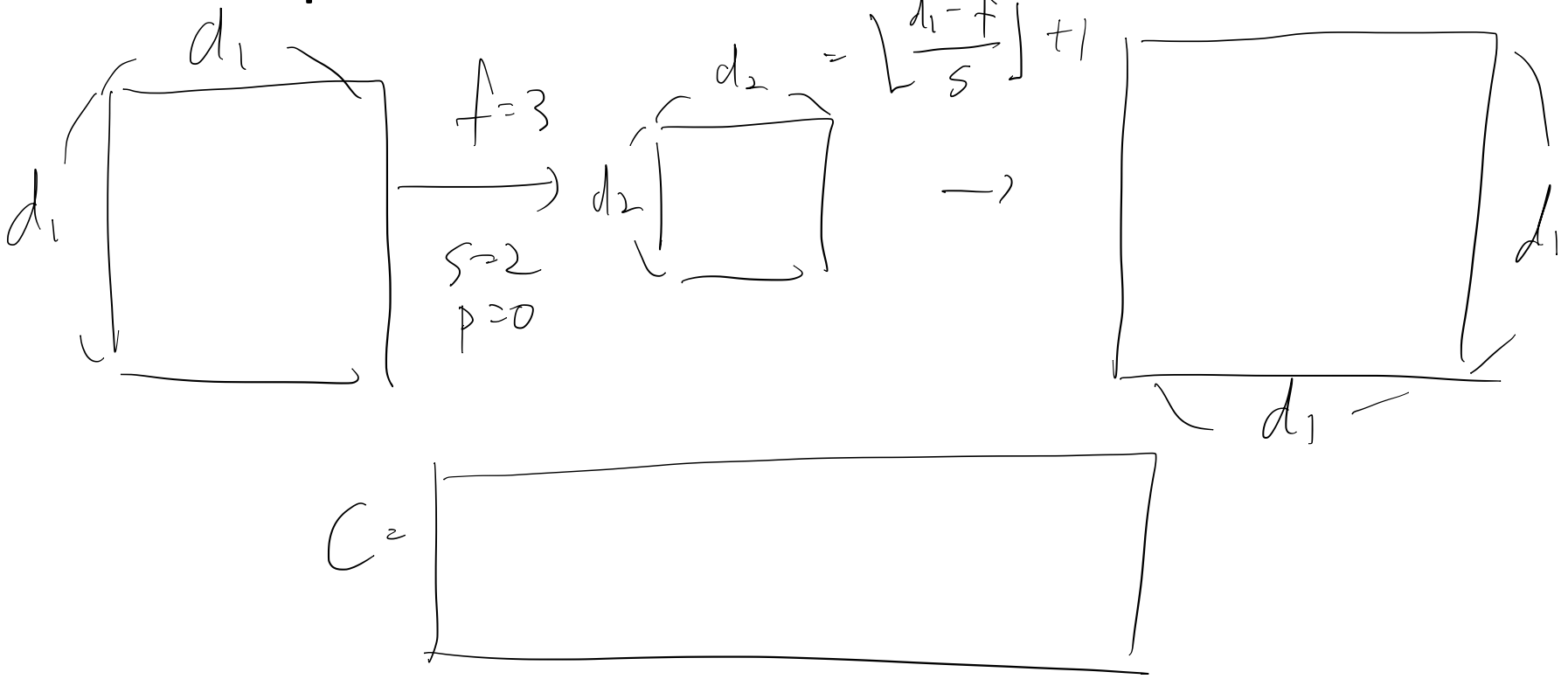


Figure 2.1: (No padding, unit strides) Convolving a 3×3 kernel over a 4×4 input using unit strides (i.e., $i = 4$, $k = 3$, $s = 1$ and $p = 0$).

C=

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

Transposed convolution



Transposed convolution (example)

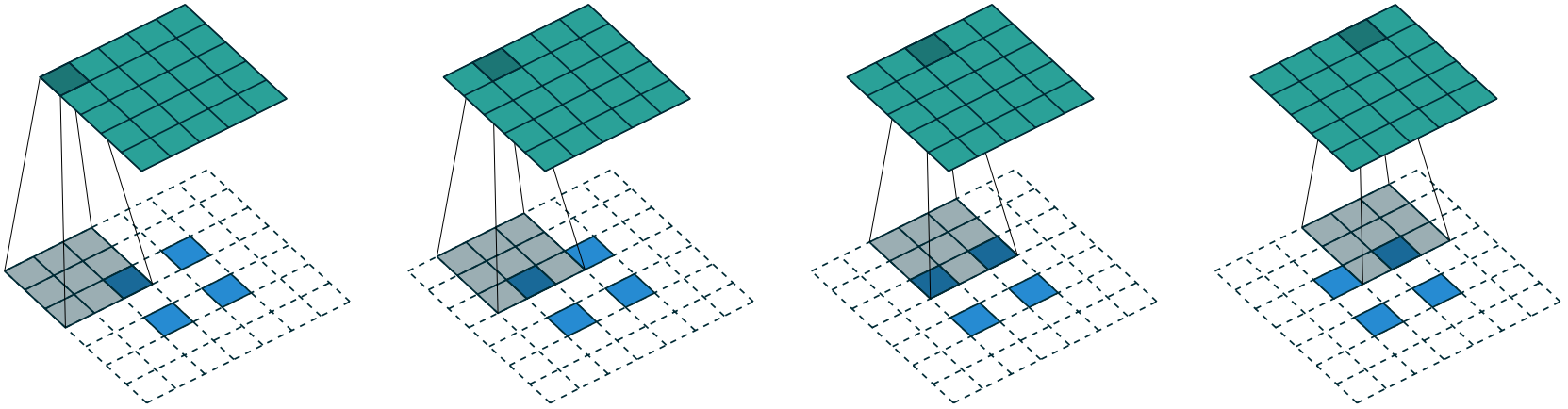
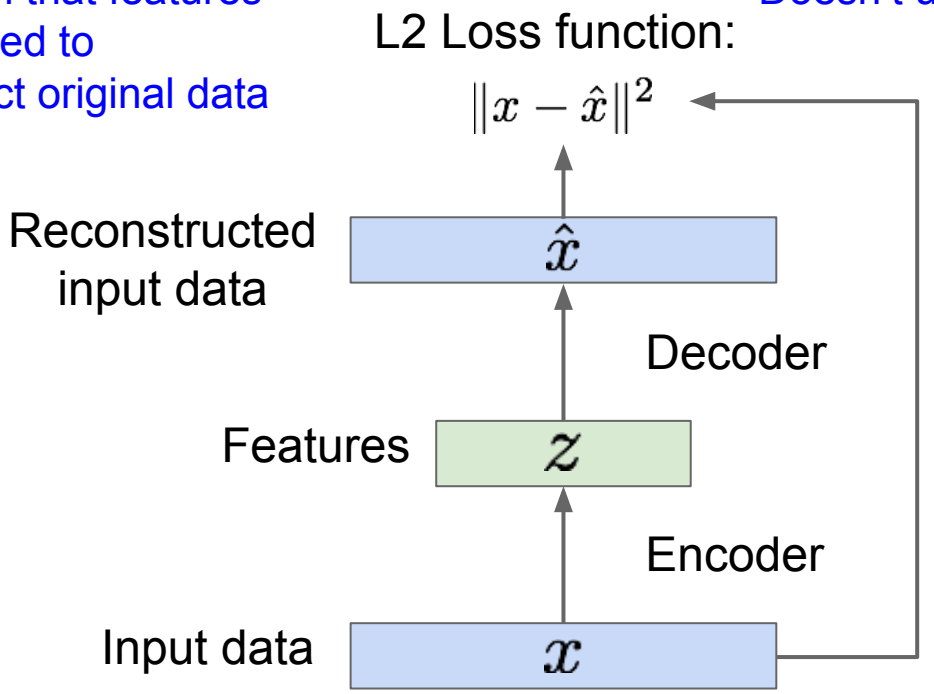


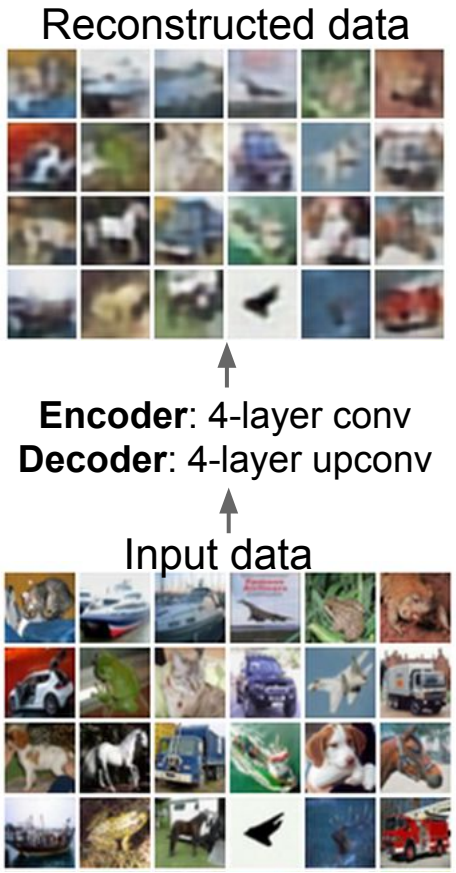
Figure 4.5: The transpose of convolving a 3×3 kernel over a 5×5 input using 2×2 strides (i.e., $i = 5$, $k = 3$, $s = 2$ and $p = 0$). It is equivalent to convolving a 3×3 kernel over a 2×2 input (with 1 zero inserted between inputs) padded with a 2×2 border of zeros using unit strides (i.e., $i' = 2$,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

Some background first: Autoencoders

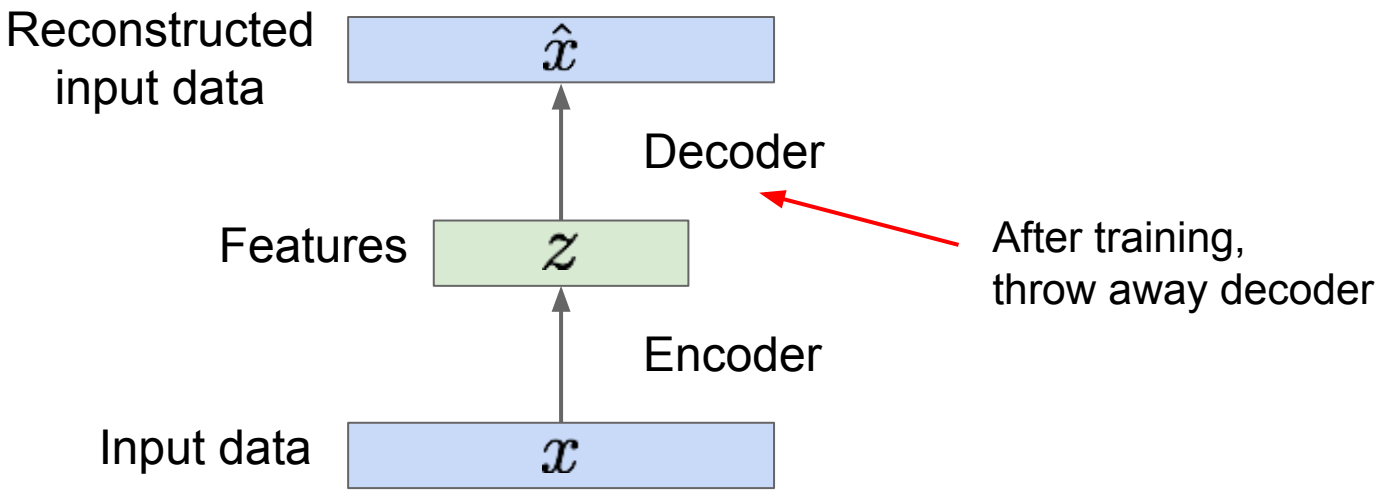
Train such that features can be used to reconstruct original data

Doesn't use labels!



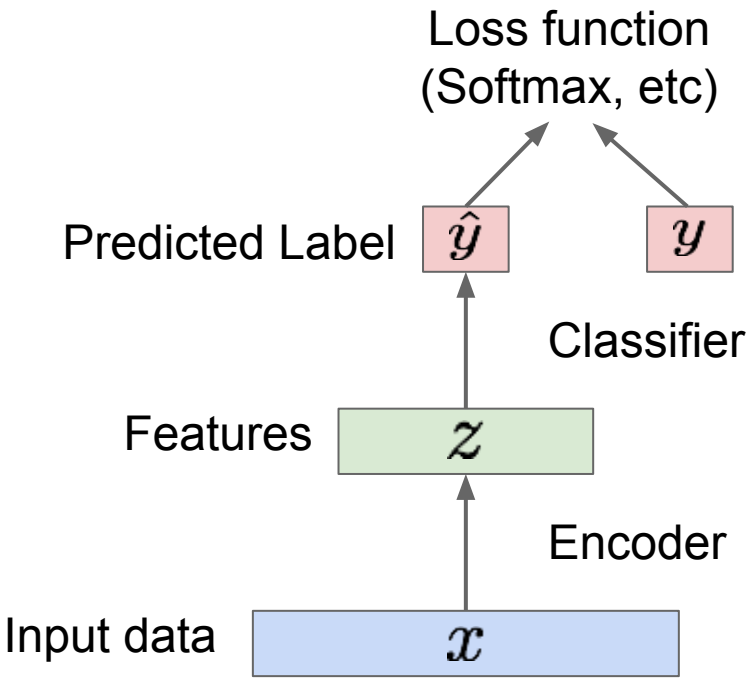


Some background first: Autoencoders



Some background first: Autoencoders

Encoder can be used to initialize a **supervised** model



Fine-tune encoder jointly with classifier

bird plane
dog deer truck

Train for final task (sometimes with small data)

Variational Autoencoder

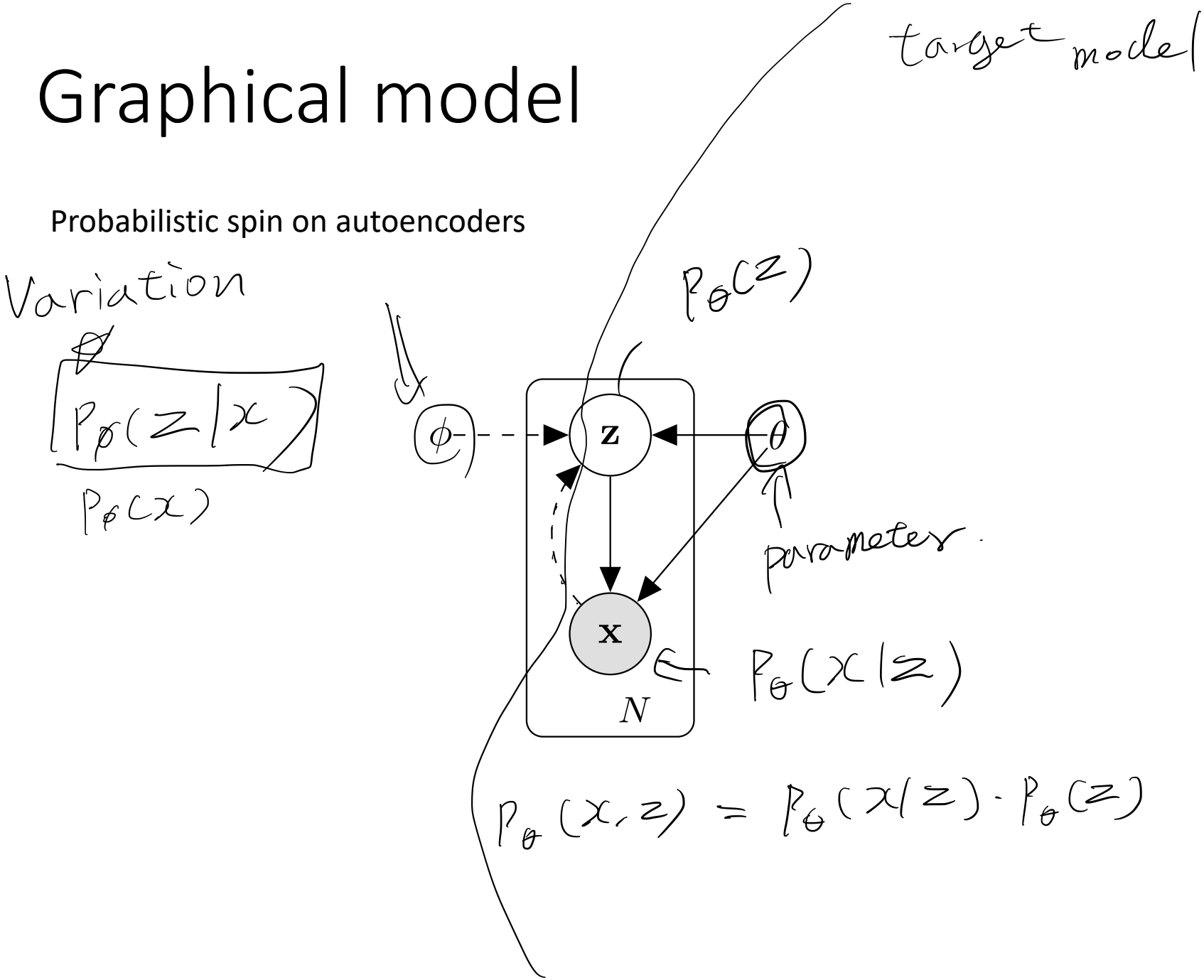
Auto-Encoding Variational Bayes

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Graphical model

Probabilistic spin on autoencoders



Why we need VAE?

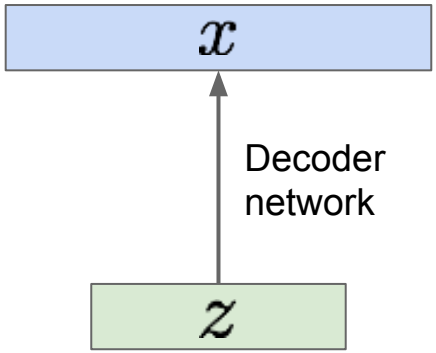
$$\min KL(p_{data}(x) \| \int p_\theta(x|z)p_\theta(z)dz) \qquad KL(p_{data}(x) \| \int p_\theta(x|z)p_\theta(z)dz)\, dx$$

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

We want to estimate the true parameters θ^* of this generative model.

Sample from true conditional $p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior $p_{\theta^*}(z)$



How to train the model?

Remember strategy for training generative models from FVBNs. 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?

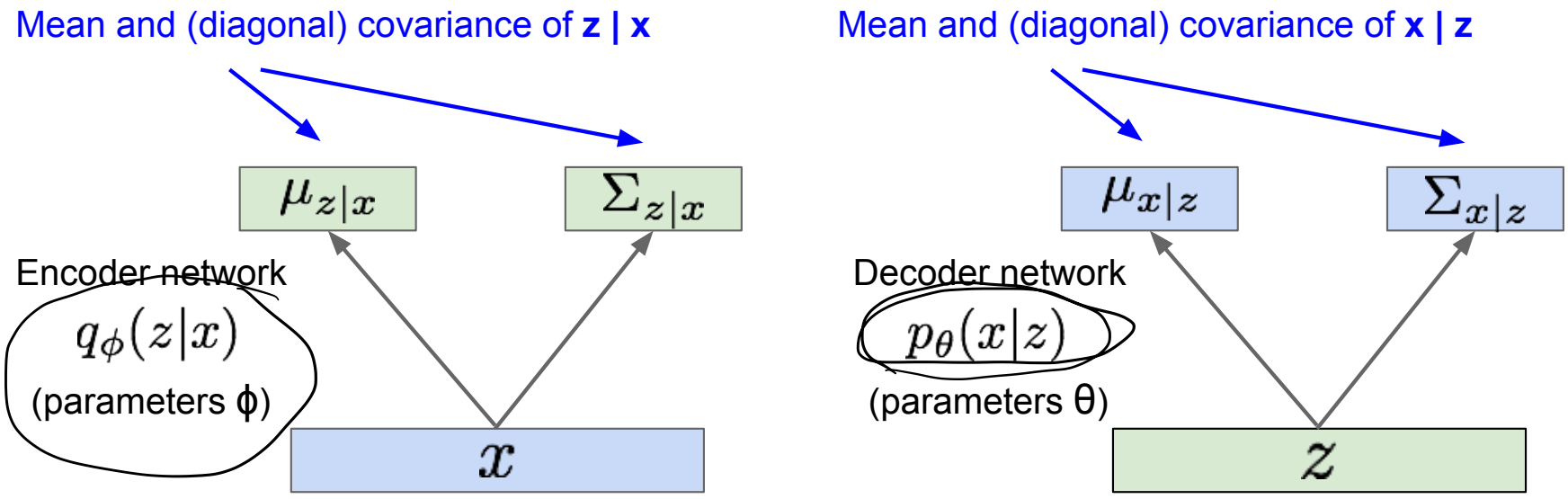
Intractable!

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

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

Variational Autoencoder

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



Evidence Lower Bound (ELBO)

$$\log p_\theta(x_i) \geq \mathcal{L}(\theta, \phi, x_i)$$

$$\log p_\theta(x_i) = \log \int p_\theta(x_i|z)\, p_\theta(z)\, dz$$

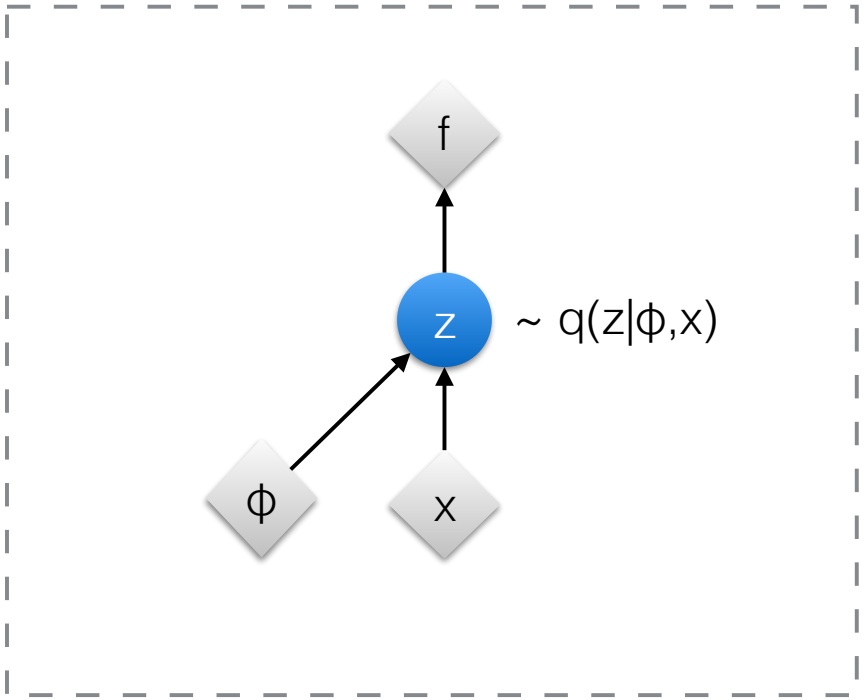
$$\mathcal{L}(\theta, \phi, x_i) = ?$$

ELBO and KL divergence

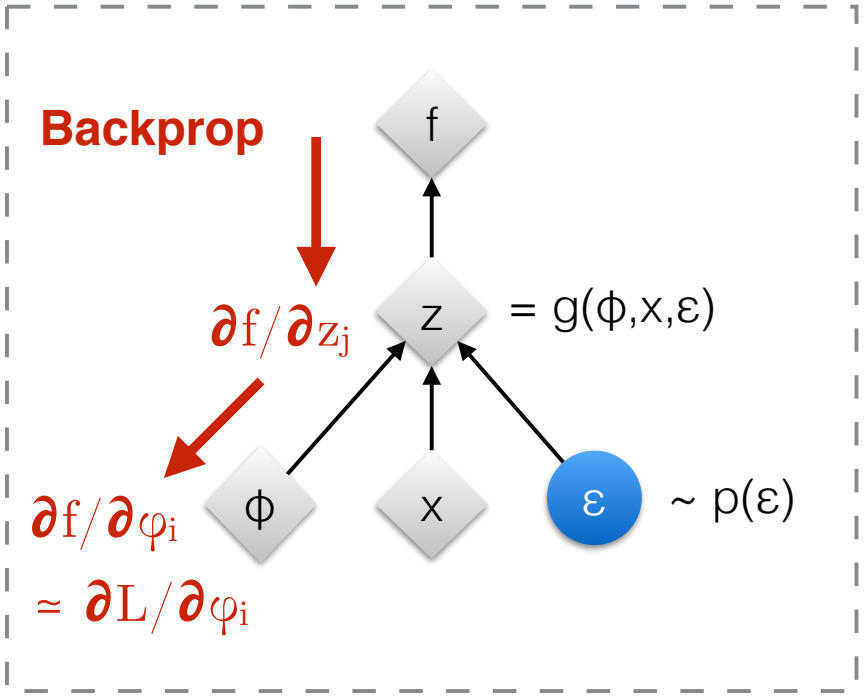
Tractable ELBO

Reparameterization Trick

Original form



Reparameterised form



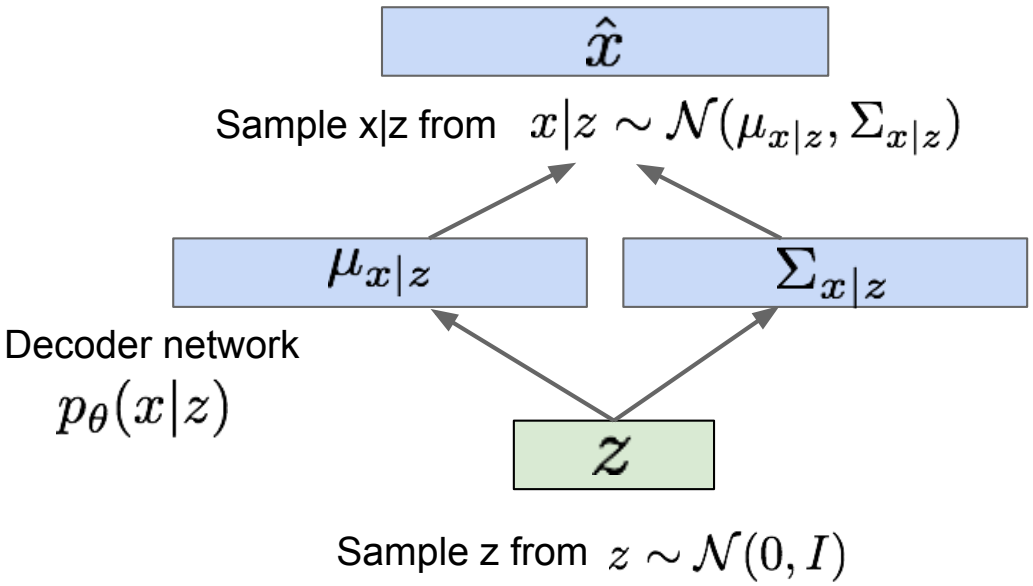
: Deterministic node

: Random node

[Kingma, 2013]
[Bengio, 2013]
[Kingma and Welling 2014]
[Rezende et al 2014]

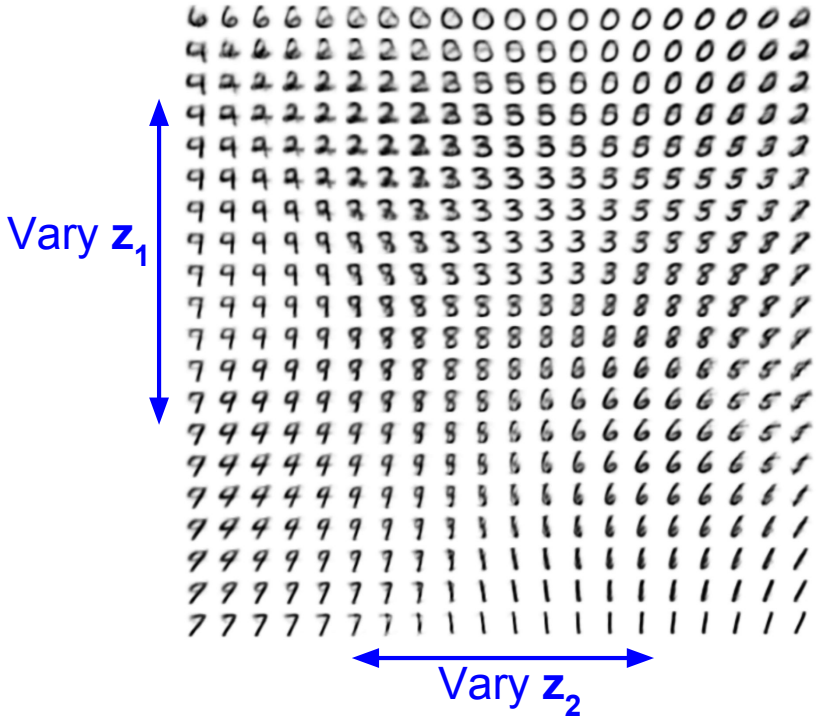
Variational Autoencoders: Generating Data!

Use decoder network. Now sample z from prior!



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

Data manifold for 2-d **z**



Variational Autoencoders: Generating Data!

Diagonal prior on **z** => independent latent variables

Different dimensions of **z** encode interpretable factors of variation

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

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