# Generative Deep Networks

Jakob Verbeek
INRIA, Grenoble, France

January 18, 2017

Thanks to Aaron Courville, Ian Goodfellow, Durk Kingma and Kevin McGuinness for figures and slides

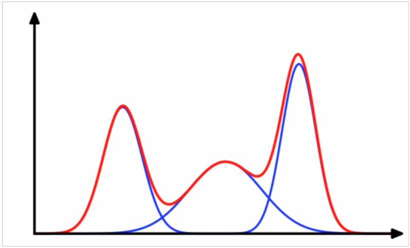
# What is a generative model?

- A model $p_\theta(x)$ we can draw samples from
  - For example, a Gaussian mixture model

$$p_\theta(x) = \sum_{k=1}^{K} p_\theta(z = k) p_\theta(x|z = k) \quad (1)$$

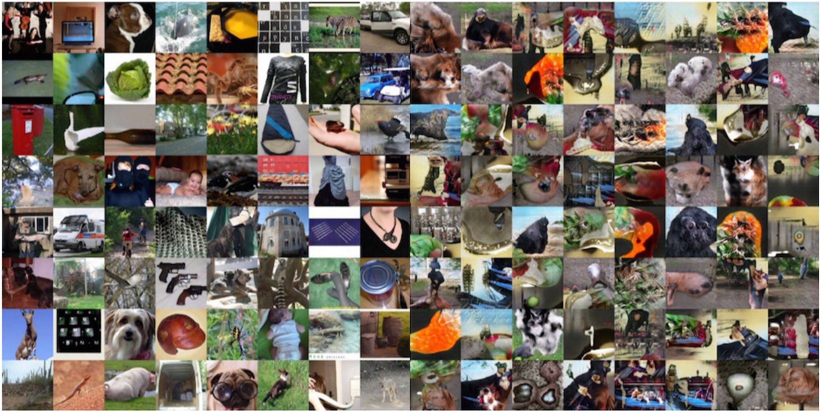
  - Estimation with Expectation-Maximization algorithm
  - Sampling: pick component from prior distribution $p_\theta(k)$, then draw sample from selected Gaussian

- Need more complex distributions in practice



# Example: modeling images

- Modeling the distribution of $10^6$ ImageNet samples

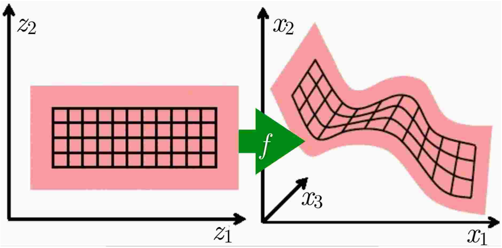


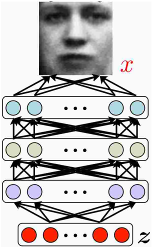
# Why is generative modeling important?

- Unsupervised learning to regularize supervised learning
- Generate training data for discriminative models
- Discriminative tasks where the output has multiple modes
- Generate novel visual content (in-painting)
  - Proxy-task to study complex generative models

# How to design complex generative models?

- Generate a latent variable $z$ from a simple distribution $p(z)$, *e.g.* standard Gaussian
- Map this latent variable to an observation of interest $x$ by a (non-linear) deep network $f_\theta(\cdot)$
- Induces complex distribution $p_\theta(x)$ on $x$



# How to learn deep generative models?

- Marginal distribution on $x$ obtained by integrating out $z$

$$p(z) = \mathcal{N}(z; 0, I), \quad (2)$$

$$p_\theta(x|z) = \delta(x, f_\theta(z)), \quad (3)$$

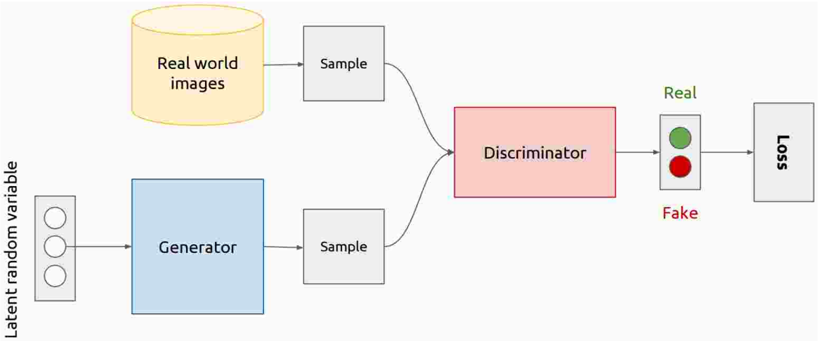
$$p_\theta(x) = \int_z p(z) p_\theta(x|z). \quad (4)$$

- Evaluation of $p_\theta(x)$ intractable due to integral involving non-linear deep net $f_\theta(\cdot)$
- Maximum likelihood estimation non-trivial
- Two recent promising approaches
  - Generative adversarial networks
  - Auto-encoding variational Bayes

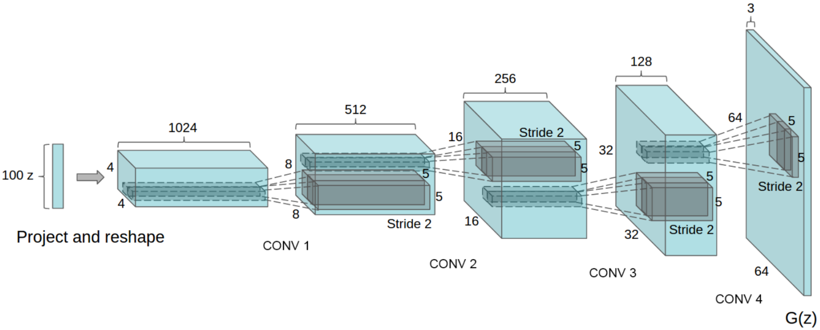
# Generative adversarial networks

- Introduced by Goodfellow *et al.* in 2014 [GPAM$^+$14]
- Don't try to evaluate $p_\theta(x)$, just learn to sample from it
  - Sample $z$, map it using deep net to $x = f_\theta(z)$
- Avoids dealing with intractable integral
- Idea: pit generative model against a discriminative model
- Discriminator tries to tell samples from generative model from real samples
- Discriminator is a second deep network, train both in competition

# Schematic setup of adversarial training

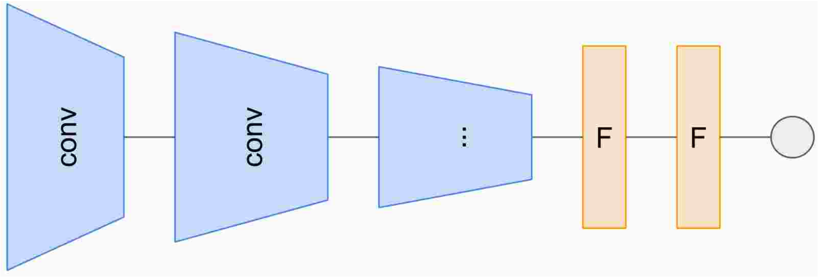


# Typical generator architecture, for images



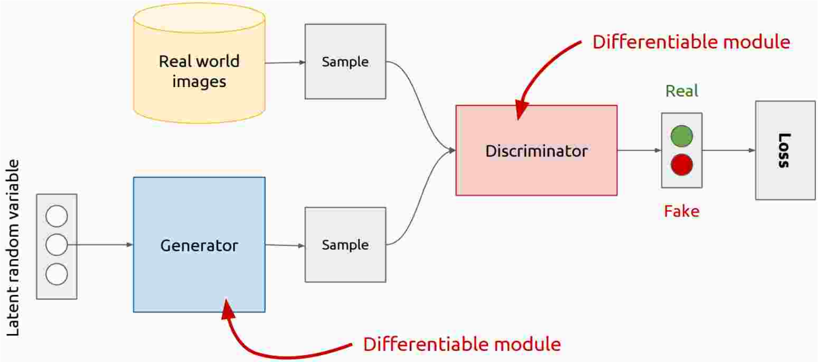

- Unit Gaussian distribution on $z$, typically 10-100 dim.
- Up-convolutional deep network (reverse recognition CNN)

# Typical discriminator architecture, for images



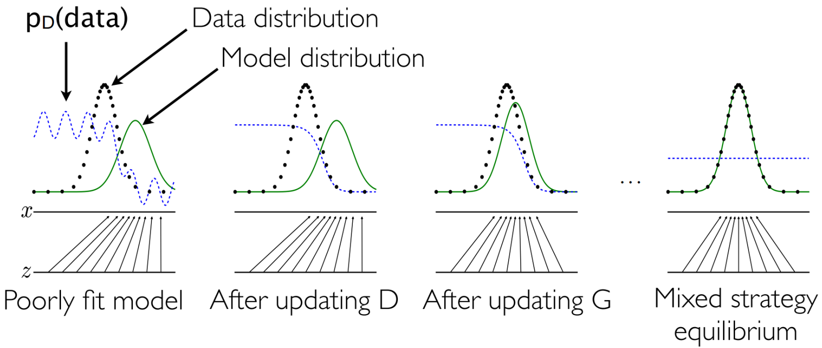

- Recognition CNN model
- Binary classification output: real / synthetic

# Training GANs




- Discriminator: maximum likelihood on correct class label, given generator
- Generator: minimize likelihood on correct class label, given discriminator

# Learning process



# Theoretical properties

$$\min_{\theta} \max_{\phi} V(\phi, \theta) = \mathbb{E}_{x \sim p_{\mathrm{data}}(x)}[\ln D_\phi(x)] + \mathbb{E}_{z \sim p(z)}[\ln(1 - D_\phi(f_\theta(z)))] \quad (5)$$

- Theoretical properties, assuming infinite data, infinite model capacity, reaching optimal discriminator given the generator at each iteration
  - Unique global optimum
  - Optimum corresponds to data distribution
  - Convergence to optimum guaranteed

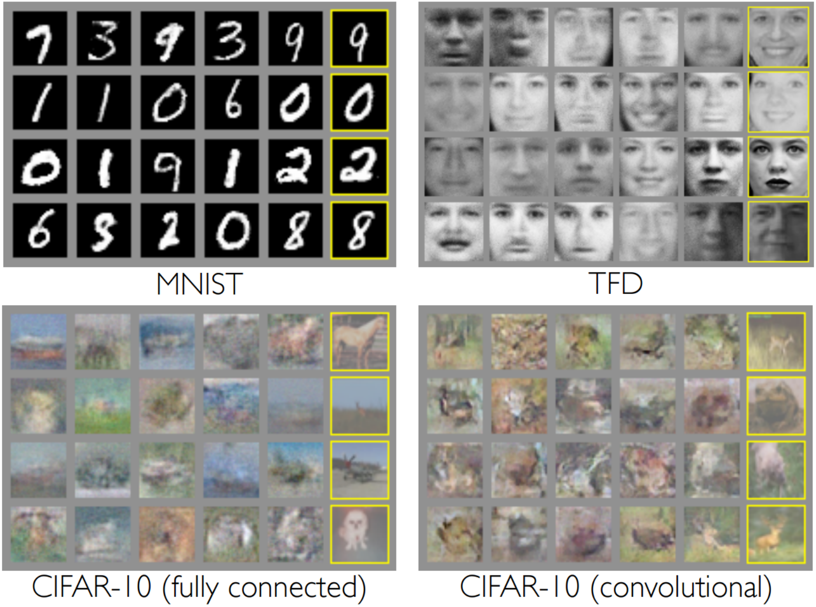
# How to evaluate the generative model?

- By construction intractable to compute $p_\theta(x^*)$, in particular for points in a test set
- Approximate value of $p_\theta(x^*)$ with Parzen window estimator using samples $x_l \sim p_\theta(x)$, see [BBV11]

$$p_{\text{parzen}}(x^*) = \frac{1}{L} \sum_{l=1}^{L} \mathcal{N}\left(x^*; x_l, \sigma^2 I\right) \tag{6}$$

| Model | MNIST | TFD |
|---|---|---|
| DBN [3] | $138 \pm 2$ | $1909 \pm 66$ |
| Stacked CAE [3] | $121 \pm 1.6$ | $\mathbf{2110 \pm 50}$ |
| Deep GSN [6] | $214 \pm 1.1$ | $1890 \pm 29$ |
| Adversarial nets | $\mathbf{225 \pm 2}$ | $\mathbf{2057 \pm 26}$ |

# Schematic setup of adversarial training



# Generating hotel bedrooms

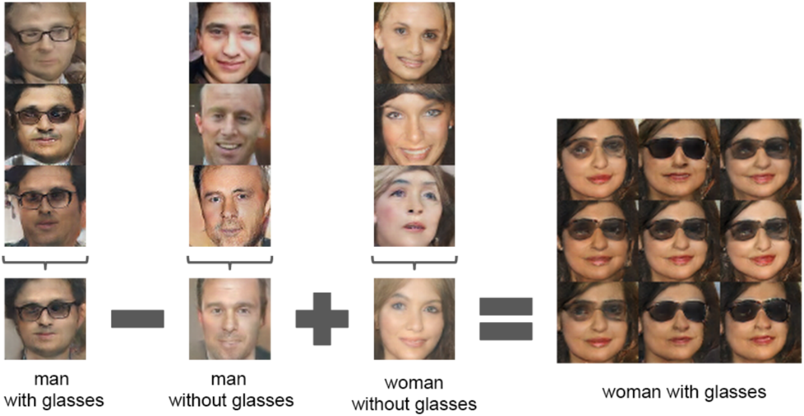
- Trained on LSUN dataset, 3 million images [RMC16]
- Linear trajectory in latent space between $z_1$ and $z_2$
  - Smooth transitions suggest generalization
  - Sharp transitions would suggest literal memorization

# Vector arithmetic on faces

- Word embedding with word2vec shows regularities of type

$$z_{\text{king}} - z_{\text{man}} + z_{\text{woman}} \approx z_{\text{queen}} \quad (7)$$

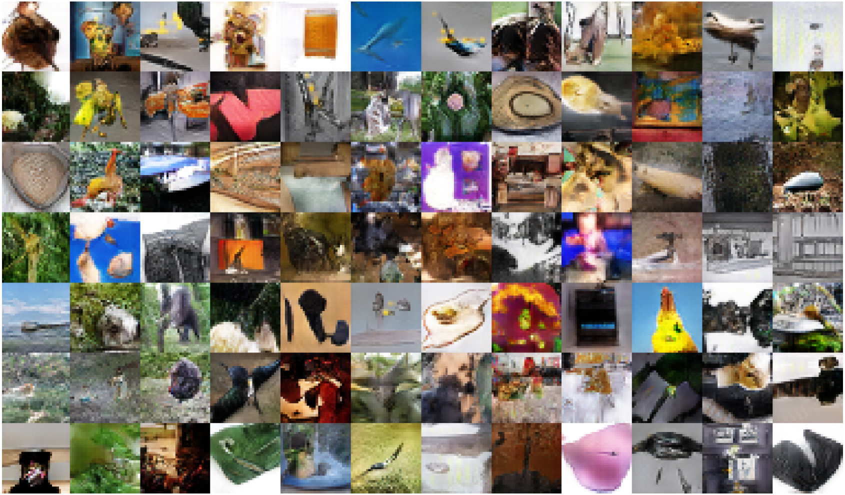
- For faces, averaging $z$ vectors over three samples for stability



# More fun with faces: approximately linear pose embedding

# More face samples

# ImageNet samples



# Issues in practice

- GANs are known to be very difficult to train in practice
- Formulated as mini-max objective between two networks
- Optimization can oscillate between solutions
- Hard to pick “compatible” architectures between generator and discriminator
- Generator can collapse to represent part of the training data, and miss another part

# Back to design of complex generative models

- Generate a latent variable $z$ from a simple distribution, *e.g.* standard Gaussian
- Map latent variable to an observation $x$ by a deep net, parameterized by $\theta$, this time in a non-deterministic manner
- For example, using deep net that outputs mean $\mu_\theta(\cdot)$ and variance $\sigma_\theta(\cdot)$ of iid Gaussian model on output variables

$$p(z) = \mathcal{N}(z; 0, I), \tag{8}$$

$$p_\theta(x|z) = \mathcal{N}(x; \mu_\theta(z), \sigma_\theta^2(z)), \tag{9}$$

$$p(x) = \int_z p(z) p_\theta(x|z). \tag{10}$$

# Auto-encoding variational Bayes (AEVB)

- Introduced by Kingma & Welling in 2014 [KW14], see also tutorial by Carl Doersch [Doe]
- Latent variable models typically learned with Expectation - Maximization algorithm, think EM for mixture of Gaussians
- In case of generative model based on deep net defining $p_\theta(x|z)$, posterior $p_\theta(z|x)$ intractable
- Work with approximate posterior distribution instead, leads to "variational EM" algorithm

# Variational bound on log-likelihood

- General approach underlying the EM algorithm
- Lower-bound marginal likelihood on $x$ with KL-divergence over posterior $p_\theta(z|x)$

$$p_\theta(x) = \int_z p(z) p_\theta(x|z), \tag{11}$$

$$F \equiv \ln p_\theta(x) - D\big(q(z)||p_\theta(z|x)\big) \leq \ln p_\theta(x) \tag{12}$$

- Kullback-Leibler divergence non-negative, and zero if and only if $q = p$

$$D(q||p) = \int_z q(z) \ln \frac{q(z)}{p(z)} \tag{13}$$

# Standard EM as bound optimization algorithm

$$F \equiv \ln p_\theta(x) - D\big(q(z)||p_\theta(z|x)\big) \tag{14}$$
$$= \mathbb{E}_q[\ln p(z) + \ln p_\theta(x|z)] + H(q) \tag{15}$$
$$\tag{16}$$

- Two forms used in conventional EM algorithms
  - E-step: keep model fixed, optimize over $q(z)$, see (14)
  - M-step: keep $q(z)$ fixed, optimize over parameters $\theta$, see (15). This is generally easier since expectation of conditional log-likelihood, rather than log of marginal likelihood.
- In classic mixture of Gaussian (MoG) case
  - Bound log-lik. per data point, sum over points in data set
  - Inference on latent variable is done per data point
  - Exact inference is tractable to compute, leads to tight bound

# Variational EM with inference net

- In the case of a deep generative model
  - Exact inference is intractable due to non-linearities
  - SGD training on large data makes iterative variation inference cumbersome, a one-shot posterior approximation is desirable
- Settle for optimizing non-tight bound $F$ on log-lik.
  - Referred to as “Vartiational EM” learning
  - No guarantees on true log-lik., we improve a bound instead
- Use a second “inference network”, parameterized by $\phi$, that computes approximate posterior on $z$ given $x$
  - No need to store and iteratively estimate variational distribution parameters

$$q_\phi(z|x) = \mathcal{N}\left(z; \mu_\phi(x), \sigma^2_\phi(x)\right) \tag{17}$$

# Yet a different form of the variational bound

$$F(x, \theta, \phi) = \mathbb{E}_{q_\phi}[\ln p_\theta(x|z)] - D\big(q_\phi(z|x)||p(z)\big) \qquad (18)$$

- First, "reconstruction", term measures to what extent $q$ gives the "right" $z$ for a given $x$
- Second, "regularization", term keeps $q$ from collapsing to a single point $z$
  - Can be computed in closed form if both terms are Gaussian
  - Differentiable function of inference net parameters

$$p(z) = \mathcal{N}\left(z; 0, I\right), \qquad (19)$$

$$q_\phi(z|x) = \mathcal{N}\left(z; \mu_\phi(x), \sigma^2_\phi(x)\right), \qquad (20)$$

$$D(q||p) = \frac{1}{2}\left[1 + \ln \sigma^2_\phi(x) - \mu^2_\phi(x) - \sigma^2_\phi(x)\right] \qquad (21)$$

# Approximating the reconstruction term

$$F(x, \theta, \phi) = \mathbb{E}_{q_\phi}[\ln p_\theta(x|z)] - D\big(q_\phi(z|x)||p(z)\big) \qquad (22)$$

- Expectation in reconstruction term is intractable to compute
- Approximate with a sample average over $z_s \sim q_\phi(z|x)$

$$R \equiv \mathbb{E}_{q_\phi}[\ln p_\theta(x|z)] \approx \frac{1}{S}\sum_{s=1}^{S} \ln p_\theta(x|z_s) \qquad (23)$$

- Estimator is non-differentiable due to sampling operator

# Re-parametrization trick

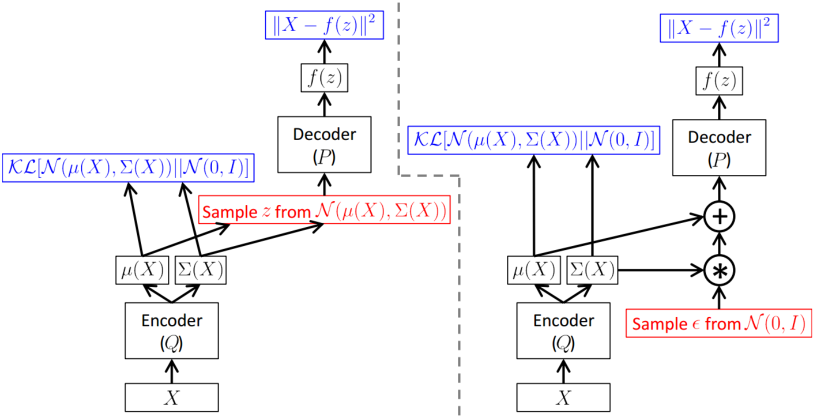
- Side-step non-differentiable sampling operator by re-parametrizing samples $z_s \sim q_\phi(z|x) = \mathcal{N}\left(z; \mu_\phi(x), \sigma^2_\phi(x)\right)$
- Use inference net to modulate samples from a unit Gaussian

$$z_s = \mu_\phi(x) + \sigma_\phi(x)\epsilon_s, \qquad \epsilon_s \sim \mathcal{N}\left(\epsilon_s; 0, I\right) \tag{24}$$

- Sample estimator is now a differentiable function of inference net, given unit Gaussian samples
- Entire objective function approximated in unbiased manner by differentiable function

$$F(x, \theta, \phi) \approx F(x, \theta, \phi, \{\epsilon_s\}) = \frac{1}{S}\sum_{s=1}^{S} \ln p_\theta(x|z_s) - D\big(q_\phi(z|x)||p(z)\big) \tag{25}$$

# Re-parametrization trick, in a cartoon

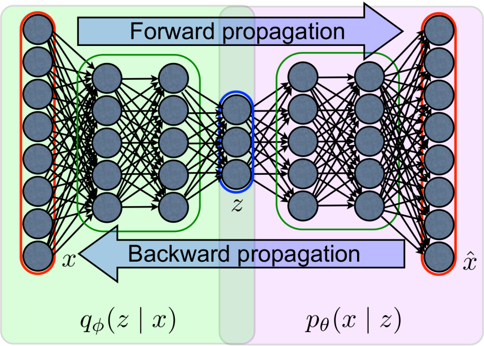


# Auto-encoder view

- Encoder: inference net takes example $x$, computes encoding $z$
- Decoder: generative net takes code $z$, computes sample $x$

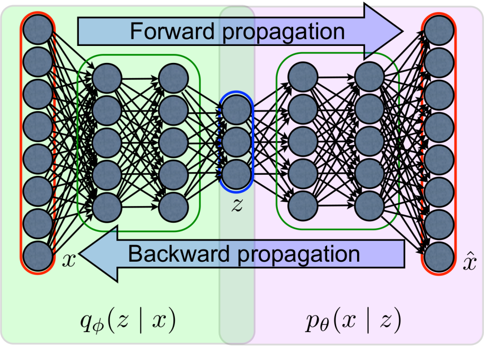
- Two terms in loss function
  - KL divergence at central bottleneck (code) layer
  - Reconstruction term at decoder output (last) layer

$$F(x, \theta, \phi, \{\epsilon_s\}) = \mathbb{E}_{q_\phi}[\ln p_\theta(x|z)] - D\big(q_\phi(z|x)||p(z)\big) \quad (26)$$

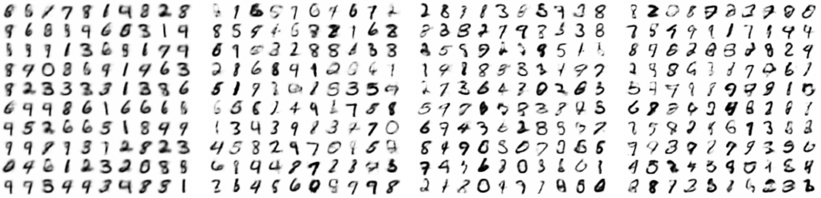


# Auto-Encoding Variational Bayes training algorithm

- Repeat:
  - Sample random training data point $x$, or mini-batch
  - Sample one or multiple values $\{\epsilon_s\}$
  - Use back-propagation to compute $g_\phi = \nabla_\phi F(x, \theta, \phi, \{\epsilon_s\})$ and $g_\theta = \nabla_\theta F(x, \theta, \phi, \{\epsilon_s\})$
  - Update parameters, set $\phi \leftarrow \phi + \alpha g_\phi$ and $\theta \leftarrow \theta + \alpha g_\theta$



# Random samples from AEVB model fit on MNIST



(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

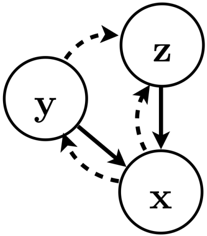
# Application of AEVB in a supervised generative model

- Variant introduced by Kingma *et al.* NIPS'14 [KRMW14]
- Class label $y$, latent variable $z$, observation $x$

$$p_\pi(y) = \mathrm{Cat}\left(y;\pi\right) \tag{27}$$
$$p(z) = \mathcal{N}\left(z;0,I\right) \tag{28}$$
$$p_\theta(x|y,z) = \mathcal{N}\left(x;\mu_\theta(y,z),\sigma^2_\theta(y,z)\right) \tag{29}$$

- Approximate posterior

$$q_\phi(y|x) = \mathrm{Cat}\left(y;\pi_\phi(x)\right), \tag{30}$$
$$q_\phi(z|x,y) = \mathcal{N}\left(z;\mu_\phi(x,y),\sigma^2_\phi(x,y)\right) \tag{31}$$



# Objective function for semi-supervised model

- Complete objective function has three terms
- Generative term for unlabeled data

$$p(x) \geq \mathcal{U}(x) \tag{32}$$
$$\mathcal{U}(x) = \mathbb{E}_{q_\phi(y,z|x)}[\ln p_\theta(x|y,z) + \ln p_\theta(y) + \ln p_\pi(z) - \ln q_\phi(y,z|x)]$$

- Generative term for labeled data,

$$p(x,y) \geq \mathcal{L}(x,y) \tag{33}$$
$$\mathcal{L}(x,y) = \mathbb{E}_{q_\phi(z|x,y)}[\ln p_\theta(x|y,z) + \ln p_\theta(y) + \ln p_\pi(z) - \ln q_\phi(z|x,y)]$$

- Discriminative term for labeled data: encoder used as classifier, otherwise encoder is only trained from unlabeled data

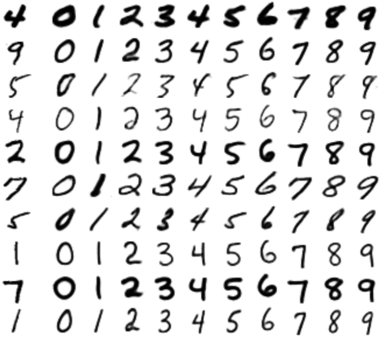
$$\mathcal{J} = \sum_{(x)\sim\tilde{p}_u} \mathcal{U}(x) + \sum_{(x,y)\sim\tilde{p}_l} \mathcal{L}(x,y) + \alpha \sum_{(x,y)\sim\tilde{p}_l} \ln q_\phi(y|x) \tag{34}$$

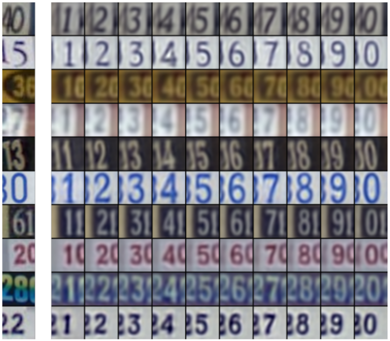
# Examples of generated images

- Handwriting styles by fixing class label $y$, and varying 2 dimensional latent variable $z$

# Examples of generated images

- The leftmost columns show images from the test set.
- The other columns show generated images $x$, where the latent variable $z$ of each row is set to the value inferred from the test-set image on the left. Each column corresponds to a class label $y$.





# References I


[BBV11] O. Breuleux, Y. Bengio, and P. Vincent, *Quickly generating representative samples from an RBM-derived process*, Neural Computation **23** (2011), no. 8, 2053–2073.

[Doe] C. Doersch, *Tutorial on variational autoencoders*, arXiv:1606.05908.

[GPAM+14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative adversarial nets*, NIPS, 2014.

[KRMW14] D. Kingma, D. Rezende, S. Mohamed, and M. Welling, *Semi-supervised learning with deep generative models*, NIPS, 2014.

[KW14] D. Kingma and M. Welling, *Auto-encoding variational Bayes*, ICLR, 2014.

[RMC16] A. Radford, L. Metz, and S. Chintala, *Unsupervised representation learning with deep convolutional generative adversarial networks*, ICLR, 2016.