Deep Generative Models

Changyou Chen

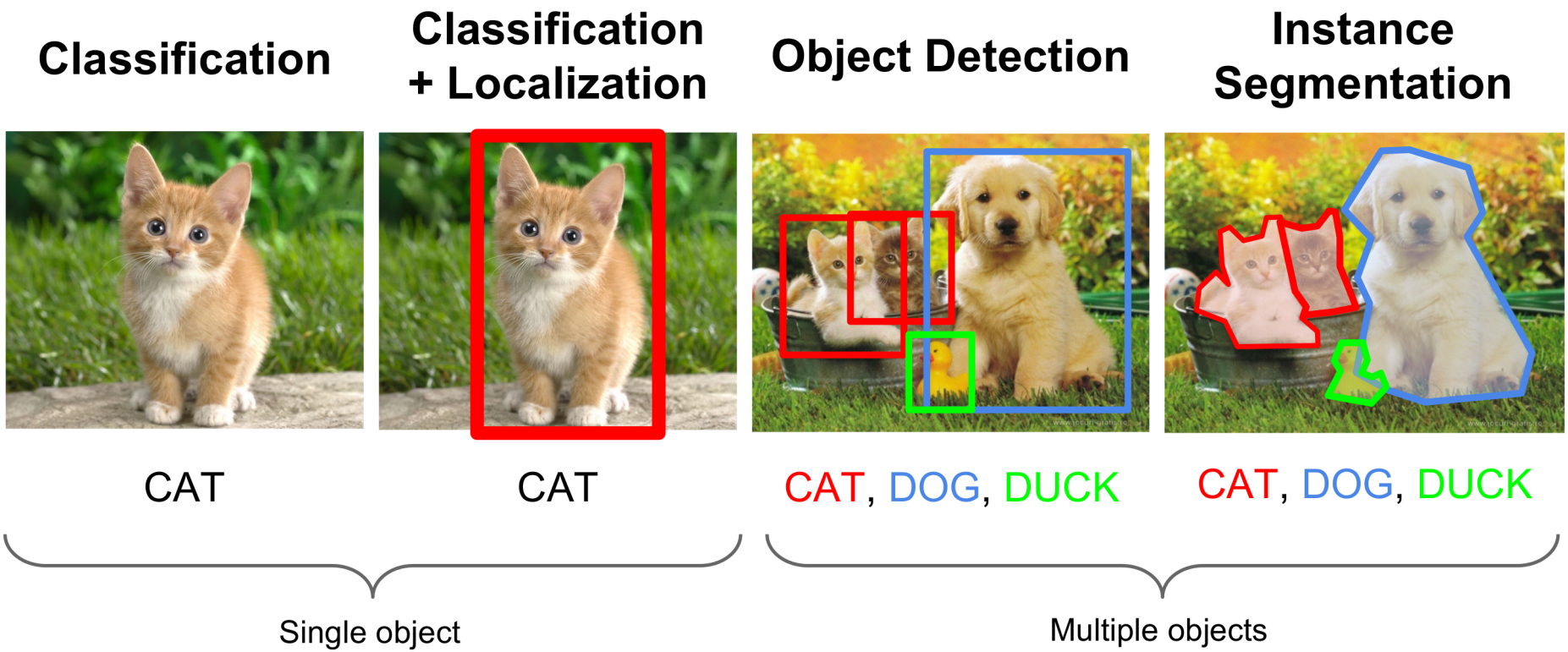
Department of Computer Science and Engineering
Universitpy at Buffalo, SUNY
changyou@buffalo.edu

April 1, 2019

Supervised vs. Unsupervised Learning[1]

Supervised Learning

- **Data** $(\mathbf{x}, y)$: $\mathbf{x}$ is data, y is label/output.
- **Goal**: Learn a function to map $\mathbf{x} \rightarrow y$.



[1] Partially adapted from http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Supervised vs. Unsupervised Learning[1]

Supervised Learning

- **Data** $(\mathbf{x}, y)$: $\mathbf{x}$ is data, y is label/output.
- **Goal**: Learn a function to map $\mathbf{x} \rightarrow y$.

1. A park with a clock tower in the background
2. A place with a tall building in the background
3. A park with a tall tree in the middle

[1] Partially adapted from http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Supervised vs. Unsupervised Learning

Unsupervised Learning

- **Data x**: no labels.
- **Goal**: Learn some underlying hidden structure of the data:
  - Define different objective functions for different models.

Supervised vs. Unsupervised Learning

Unsupervised Learning

- **Data x**: no labels.
- **Goal**: Learn some underlying hidden structure of the data:
  - Define different objective functions for different models.

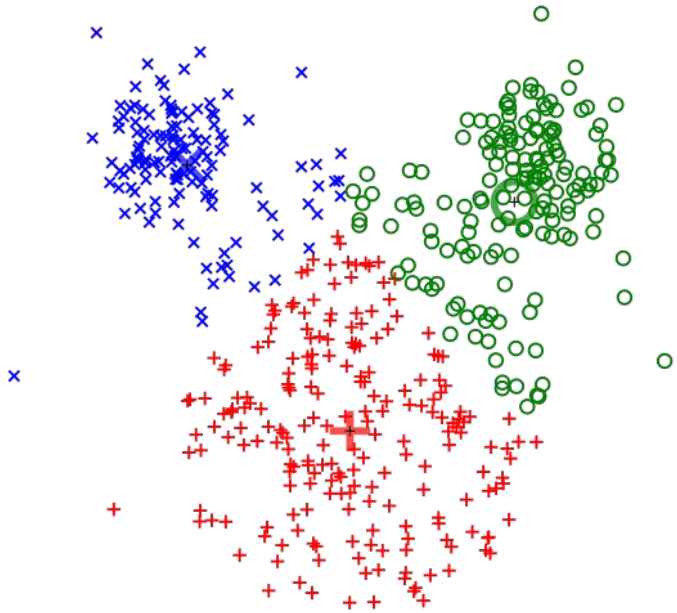


Figure: Kmeans

Supervised vs. Unsupervised Learning

Unsupervised Learning

- **Data x**: no labels.
- **Goal**: Learn some underlying hidden structure of the data:
  - Define different objective functions for different models.

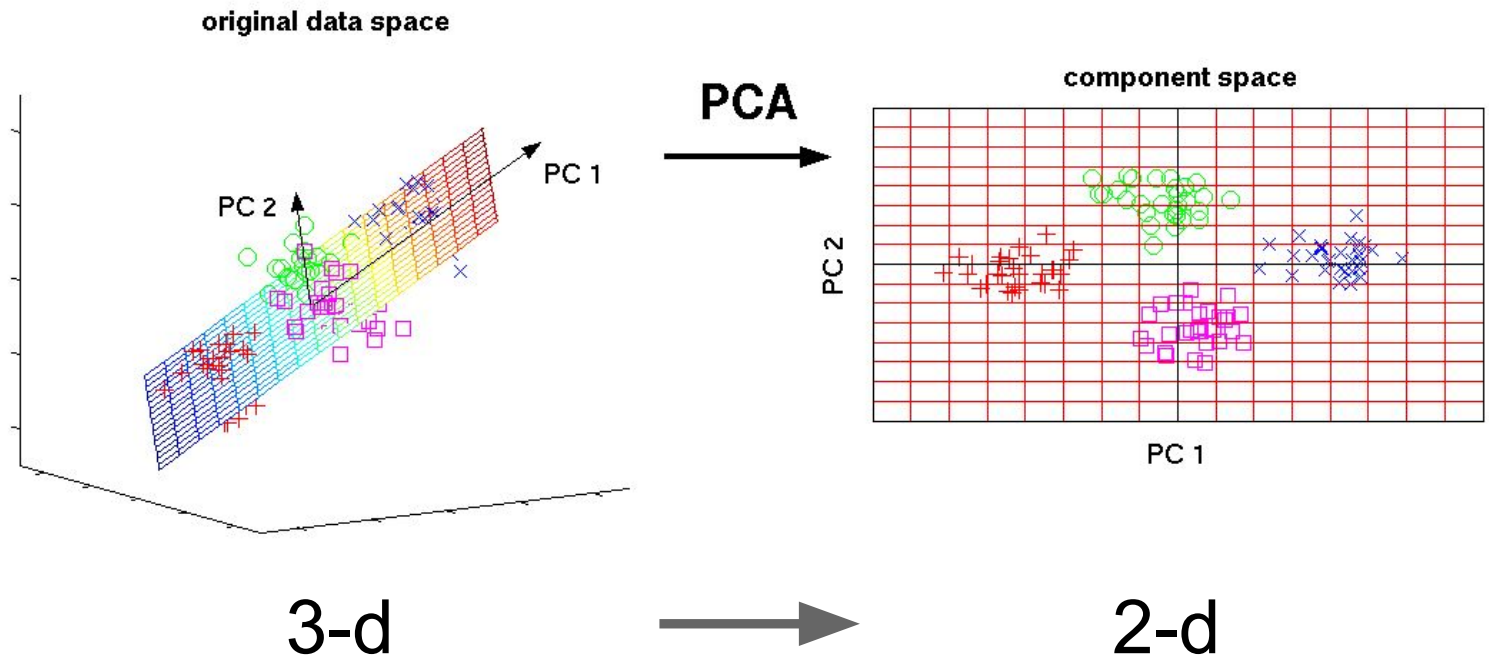


Figure: Principal component analysis (dimension reduction)

Supervised vs. Unsupervised Learning

Unsupervised Learning

- **Data x**: no labels.
- **Goal**: Learn some underlying hidden structure of the data:
  - Define different objective functions for different models.

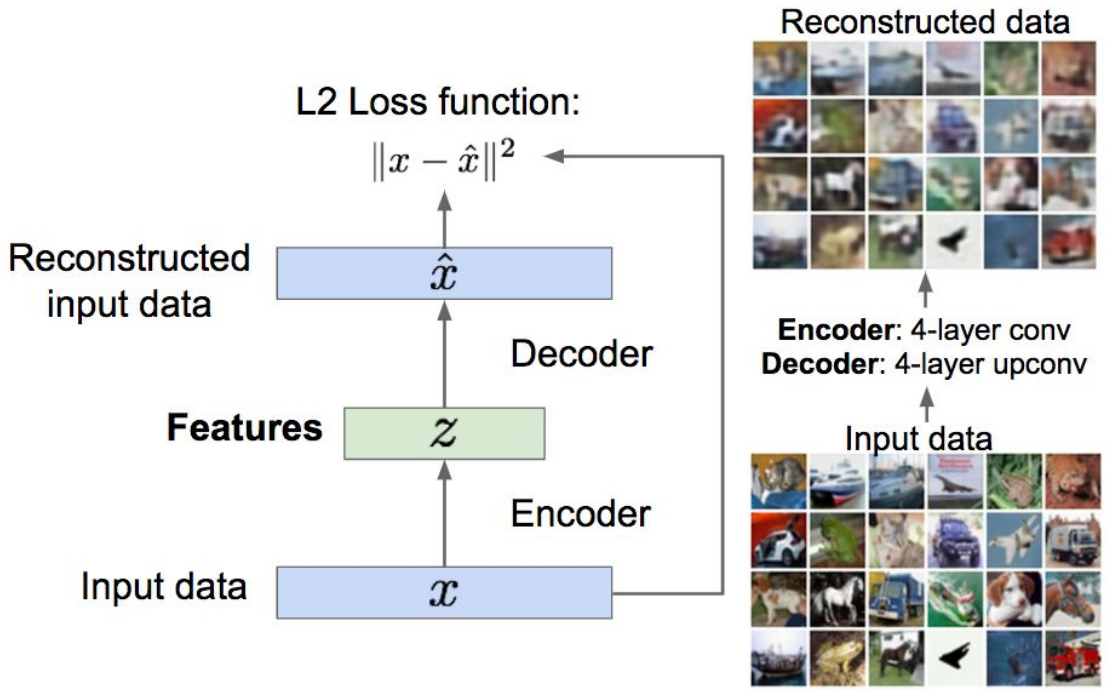
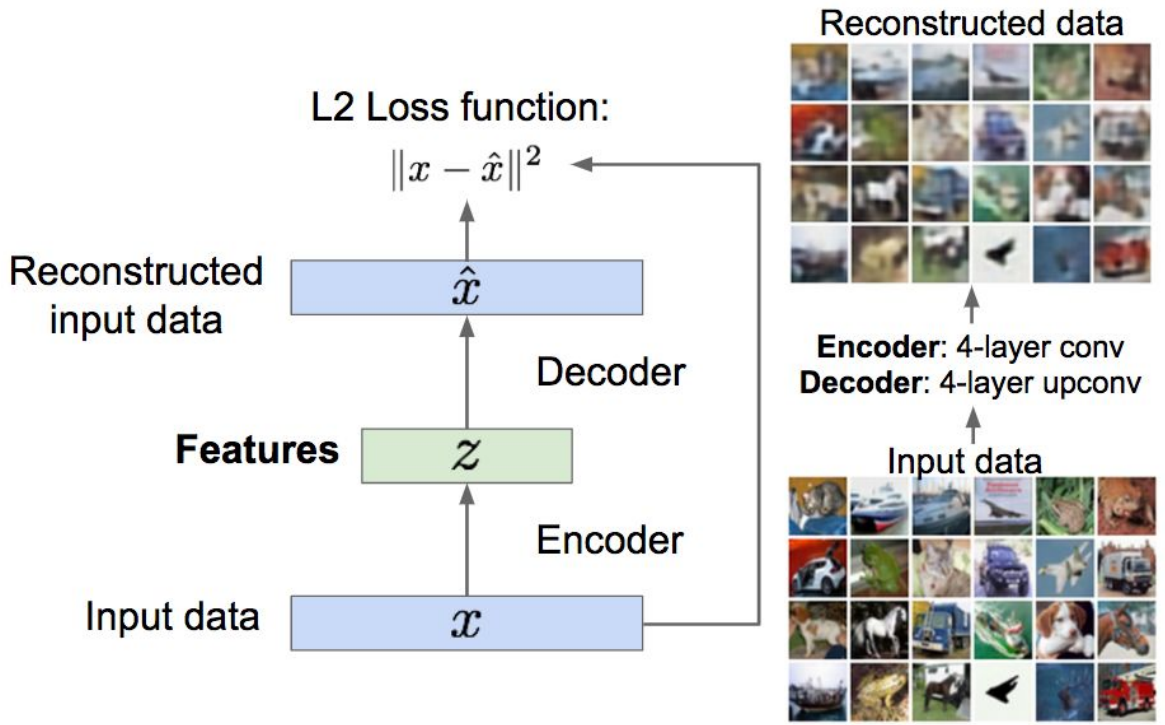


Figure: Autoencoders (feature learning)

Supervised vs. Unsupervised Learning

Unsupervised Learning

- **Data x**: no labels.
- **Goal**: Learn some underlying hidden structure of the data:
  - Define different objective functions for different models.



$\Rightarrow$ Variational autoencoder (VAE)

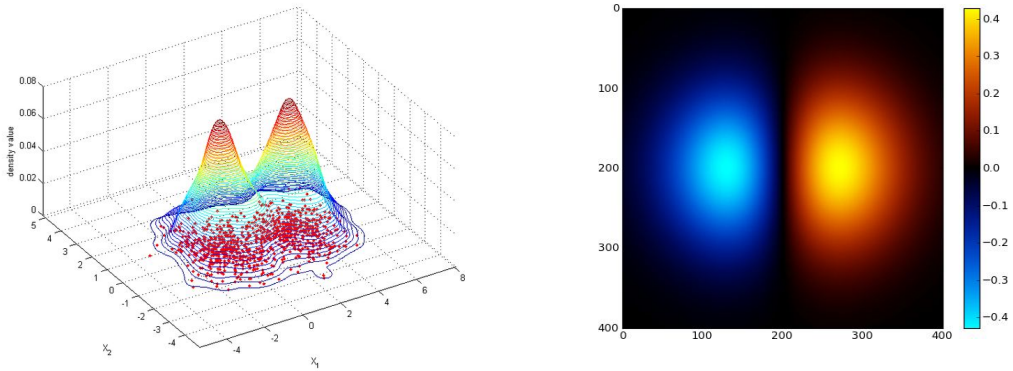
Figure: Autoencoders (feature learning)

Supervised vs. Unsupervised Learning

Unsupervised Learning

- **Data x**: no labels.
- **Goal**: Learn some underlying hidden structure of the data:
  - Define different objective functions for different models.

1-d density estimation



2-d density estimation

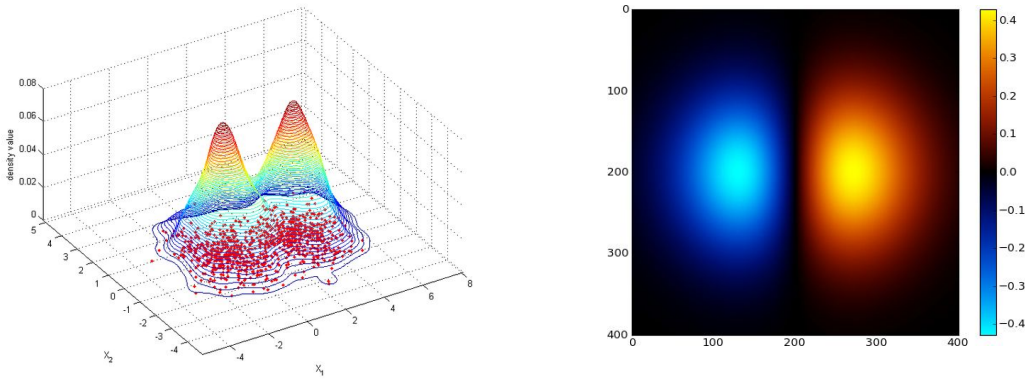
Supervised vs. Unsupervised Learning

Unsupervised Learning

- **Data x**: no labels.
- **Goal**: Learn some underlying hidden structure of the data:
  - Define different objective functions for different models.

Figure copyright Ian Goodfellow, 2016. Reproduced with permission.

1-d density estimation



2-d density estimation

$\Rightarrow$

Generative adversarial networks (GAN)

Generative Models

Generate new samples from the same data distribution.

training data $\sim p_{\text{data}}(\mathbf{x})$

generated samples $\sim p_{\text{model}}(\mathbf{x})$

- Goal is to learn $p_{\text{model}}(\mathbf{x})$ such that it is close to $p_{\text{data}}(\mathbf{x})$.

Generative Models

Generate new samples from the same data distribution.

training data $\sim p_{\text{data}}(\mathbf{x})$

generated samples $\sim p_{\text{model}}(\mathbf{x})$

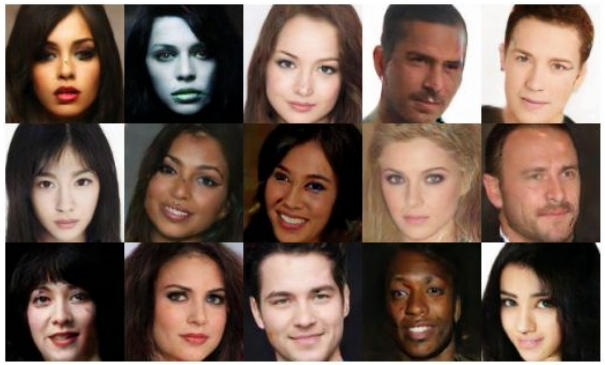
- Goal is to learn $p_{\text{model}}(\mathbf{x})$ such that it is close to $p_{\text{data}}(\mathbf{x})$.

Several flavors

- Explicit density estimation: explicitly define and solve for $p_{\text{model}}(\mathbf{x})$:
  - *e.g.*, variational autoencoder (VAE) (in this case, $\mathbf{x}$ refers to the latent representation of the data)
- Implicit density estimation: learn a model that can sample from $p_{\text{model}}(\mathbf{x})$ w/o explicitly defining it:
  - *e.g.*, generative adversarial nets (GAN)

Why Generative Models?

1. Realistic samples for artwork, super-resolution, colorization *etc*.


2. Generative models of time-series data can be used for simulation and planning $\Rightarrow$ reinforcement learning applications.
3. Training generative models can also enable inference of latent representations that can be useful as general features.
4. Effective use of unlabeled data.

Taxonomy of Generative Models

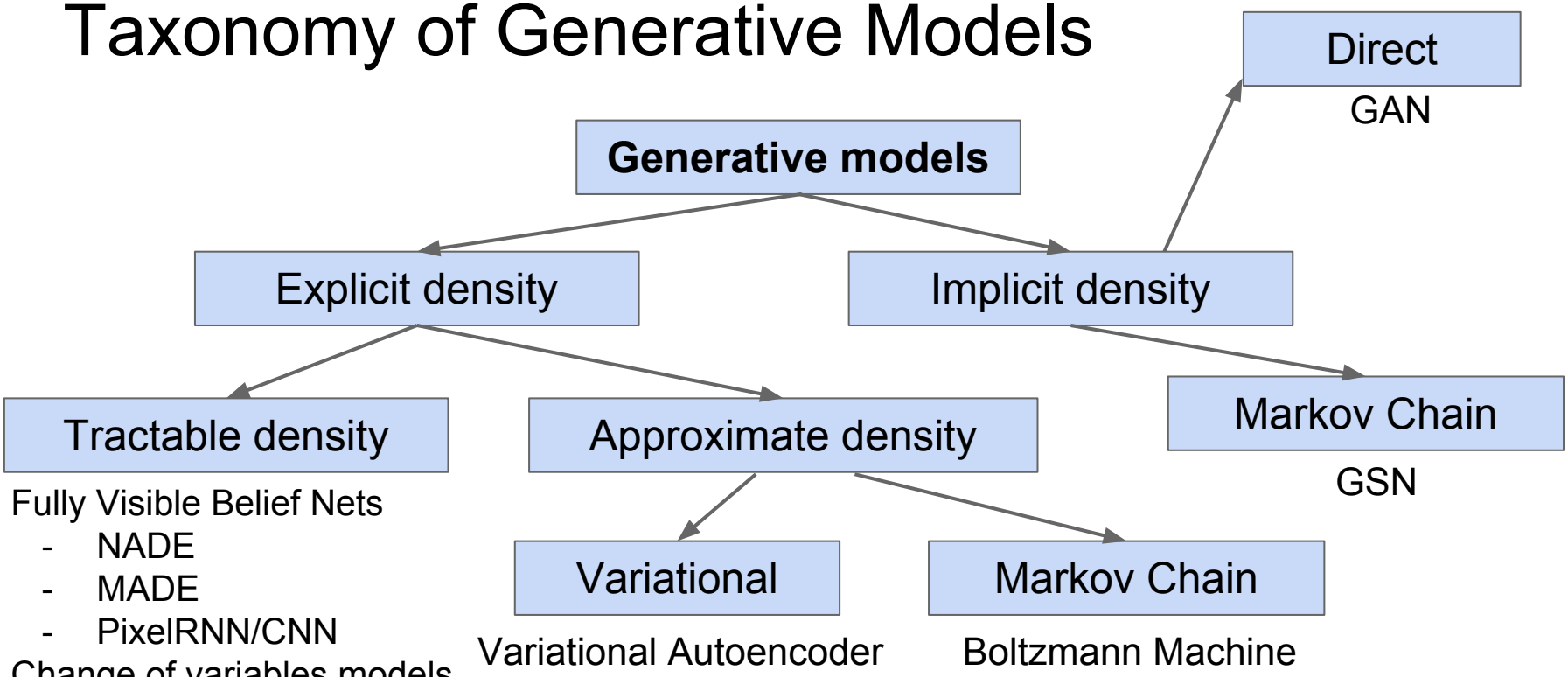


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Taxonomy of Generative Models

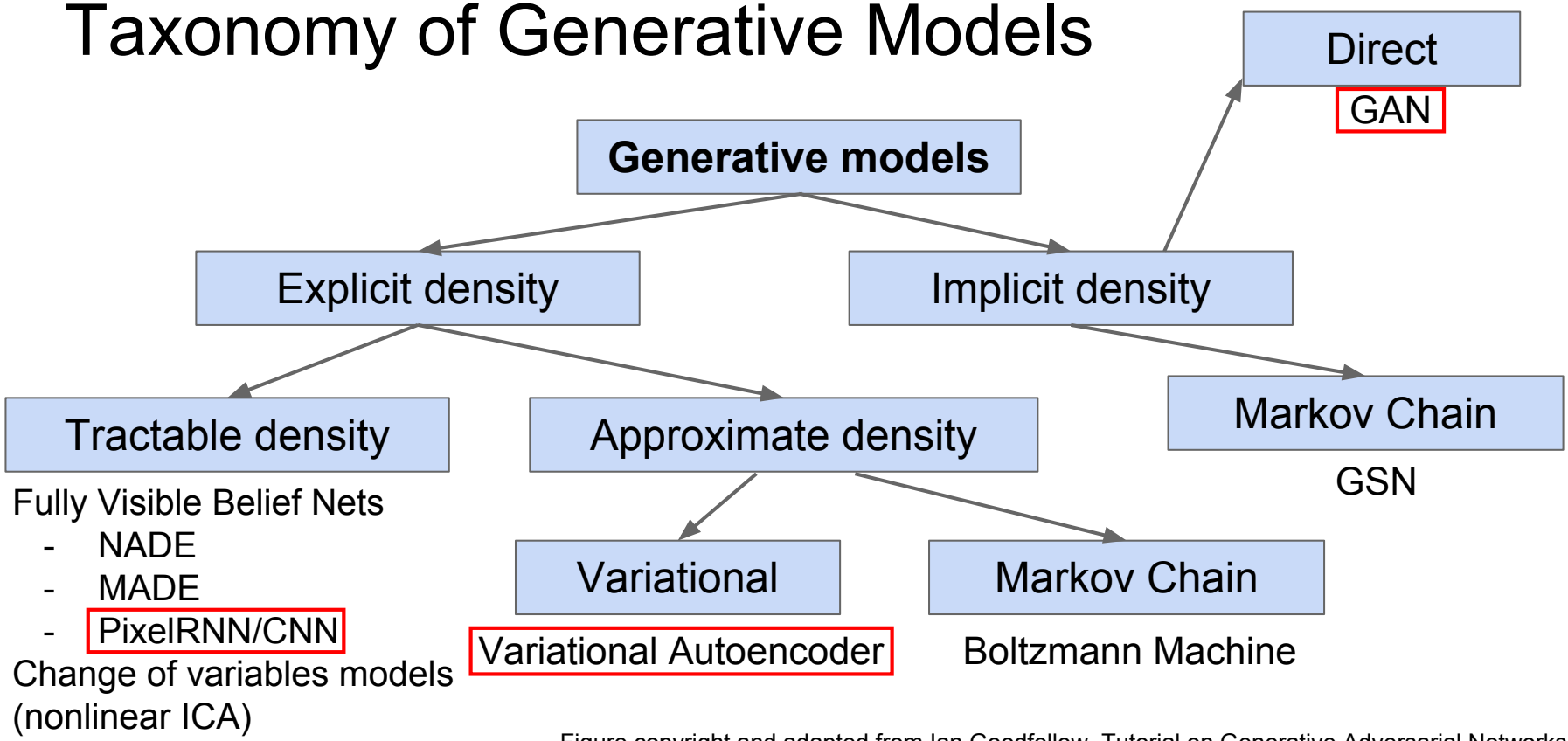


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

PixelRNN/PixelCNN

Fully Visible Belief Network

Explicit density model

- Decompose likelihood of an image **x** into product of conditional distributions for all pixels:

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i | x_1, \cdots, x_{i-1})$$

Likelihood of image $\boldsymbol{x}$

Probability of i-th pixel given all previous pixels

- Solve by maximum likelihood estimator.
- Parameterize $p(x_i|x_1, \cdots, x_{i-1})$ with a neural network to represent a complex distribution:
  - we will use RNN and CNN.
- How to define the order of $\{x_i\}$?
  - rely on specific problems.

Fully Visible Belief Network

Explicit density model

- Decompose likelihood of an image **x** into product of conditional distributions for all pixels:

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i | x_1, \cdots, x_{i-1})$$

Likelihood of image $\boldsymbol{x}$

Probability of i-th pixel given all previous pixels

- Solve by maximum likelihood estimator.
- Parameterize $p(x_i|x_1, \cdots, x_{i-1})$ with a neural network to represent a complex distribution:
  - we will use RNN and CNN.
- How to define the order of $\{x_i\}$?
  - rely on specific problems.

Fully Visible Belief Network

Explicit density model

- Decompose likelihood of an image **x** into product of conditional distributions for all pixels:

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i | x_1, \cdots, x_{i-1})$$

Likelihood of image $\boldsymbol{x}$

Probability of i-th pixel given all previous pixels

- Solve by maximum likelihood estimator.
- Parameterize $p(x_i|x_1, \cdots, x_{i-1})$ with a neural network to represent a complex distribution:
  - we will use RNN and CNN.
- How to define the order of $\{x_i\}$?
  - rely on specific problems.

Fully Visible Belief Network

Explicit density model

- Decompose likelihood of an image **x** into product of conditional distributions for all pixels:

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i | x_1, \cdots, x_{i-1})$$

Likelihood of image $\boldsymbol{x}$

Probability of i-th pixel given all previous pixels

- Solve by maximum likelihood estimator.
- Parameterize $p(x_i|x_1, \cdots, x_{i-1})$ with a neural network to represent a complex distribution:
  - we will use RNN and CNN.
- How to define the order of $\{x_i\}$?
  - rely on specific problems.

Fully Visible Belief Network

Explicit density model

- Decompose likelihood of an image **x** into product of conditional distributions for all pixels:

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i | x_1, \cdots, x_{i-1})$$

Likelihood of image $\boldsymbol{x}$

Probability of i-th pixel given all previous pixels

- Solve by maximum likelihood estimator.
- Parameterize $p(x_i|x_1, \cdots, x_{i-1})$ with a neural network to represent a complex distribution:
  - we will use RNN and CNN.
- How to define the order of $\{x_i\}$?
  - rely on specific problems.

Fully Visible Belief Network

Explicit density model

- Decompose likelihood of an image **x** into product of conditional distributions for all pixels:

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i | x_1, \cdots, x_{i-1})$$

Likelihood of image $\boldsymbol{x}$

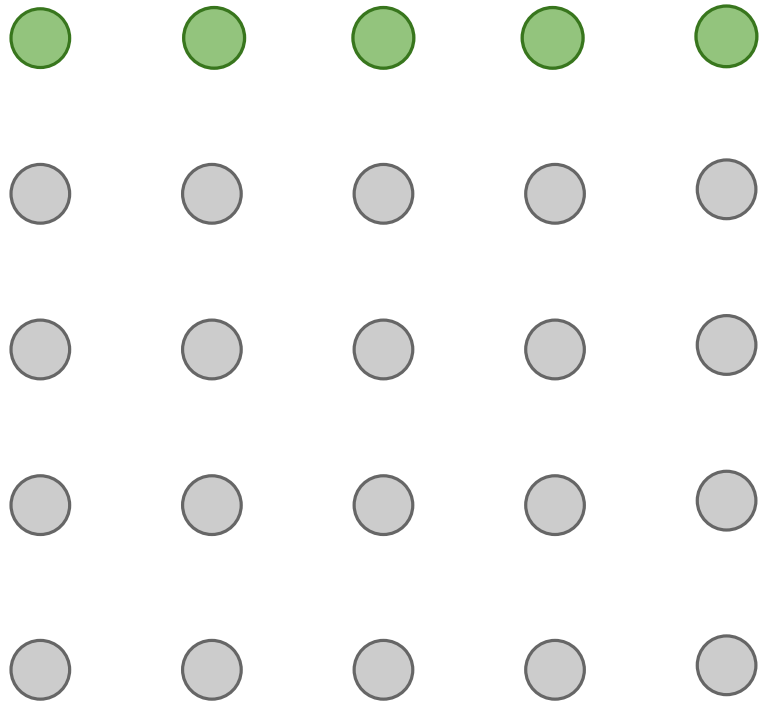
Probability of i-th pixel given all previous pixels

- Solve by maximum likelihood estimator.
- Parameterize $p(x_i|x_1, \cdots, x_{i-1})$ with a neural network to represent a complex distribution:
  - we will use RNN and CNN.
- How to define the order of $\{x_i\}$?
  - rely on specific problems.

Pixel RNN

Let's now consider modeling an image.

1. Dependency on previous pixels modeled using an RNN (LSTM).
2. Implemented via Row LSTM and Diagonal BiLSTM.
3. For Row LSTM, generate image pixels row by row.
4. For Diagonal BiLSTM, generate image pixels starting from corner.

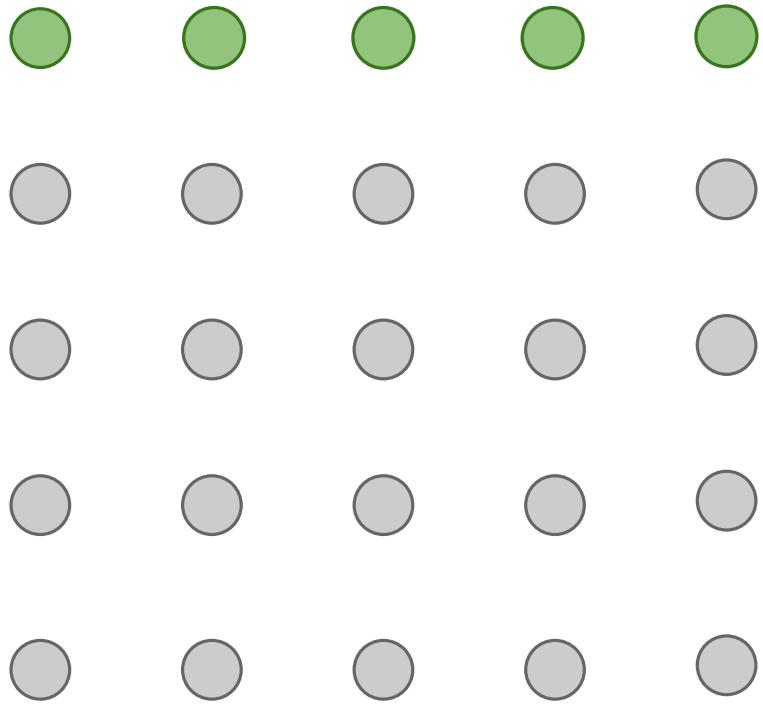


van der Oord *et al.* 2016

Pixel RNN

Let's now consider modeling an image.

1. Dependency on previous pixels modeled using an RNN (LSTM).
2. Implemented via Row LSTM and Diagonal BiLSTM.
3. For Row LSTM, generate image pixels row by row.
4. For Diagonal BiLSTM, generate image pixels starting from corner.

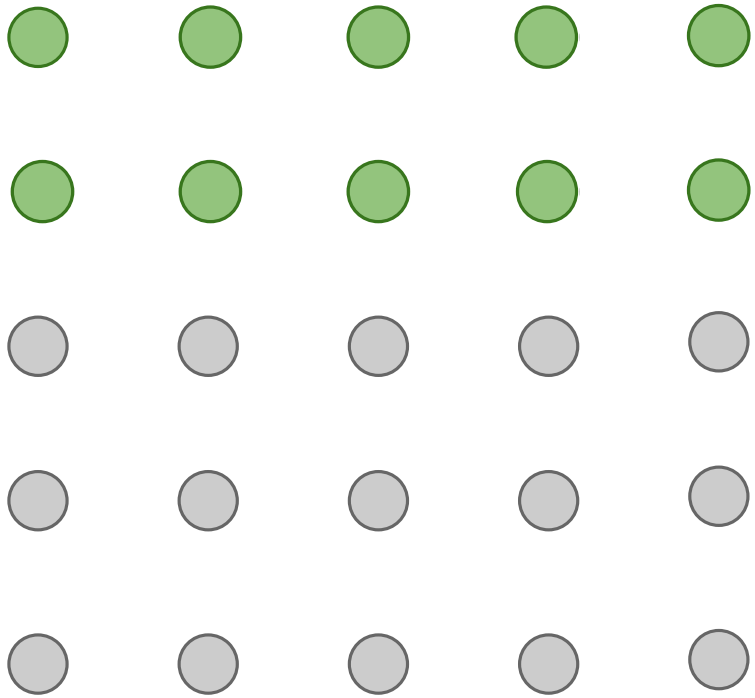


van der Oord *et al.* 2016

Pixel RNN

Let's now consider modeling an image.

1. Dependency on previous pixels modeled using an RNN (LSTM).
2. Implemented via Row LSTM and Diagonal BiLSTM.
3. For Row LSTM, generate image pixels row by row.
4. For Diagonal BiLSTM, generate image pixels starting from corner.

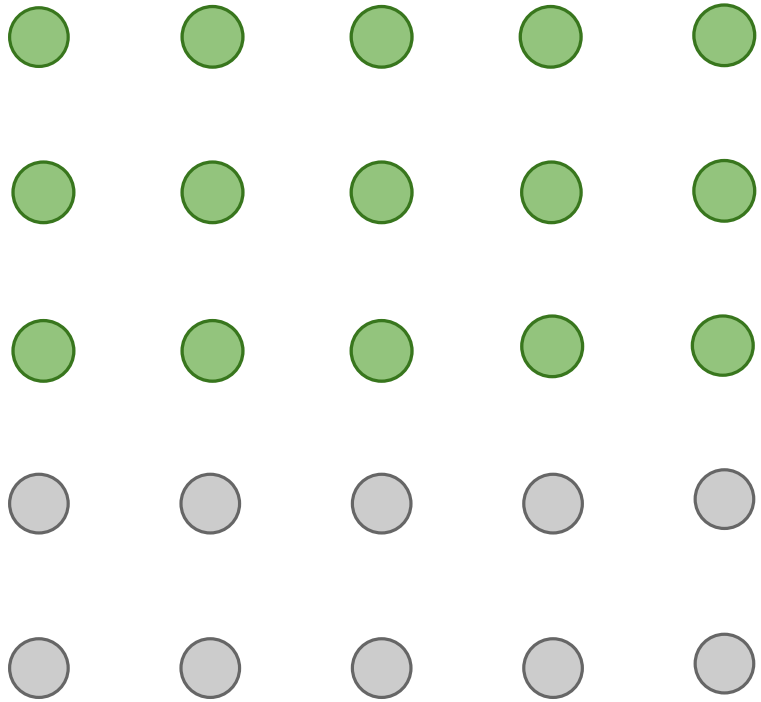


van der Oord *et al.* 2016

Pixel RNN

Let's now consider modeling an image.

1. Dependency on previous pixels modeled using an RNN (LSTM).
2. Implemented via Row LSTM and Diagonal BiLSTM.
3. For Row LSTM, generate image pixels row by row.
4. For Diagonal BiLSTM, generate image pixels starting from corner.

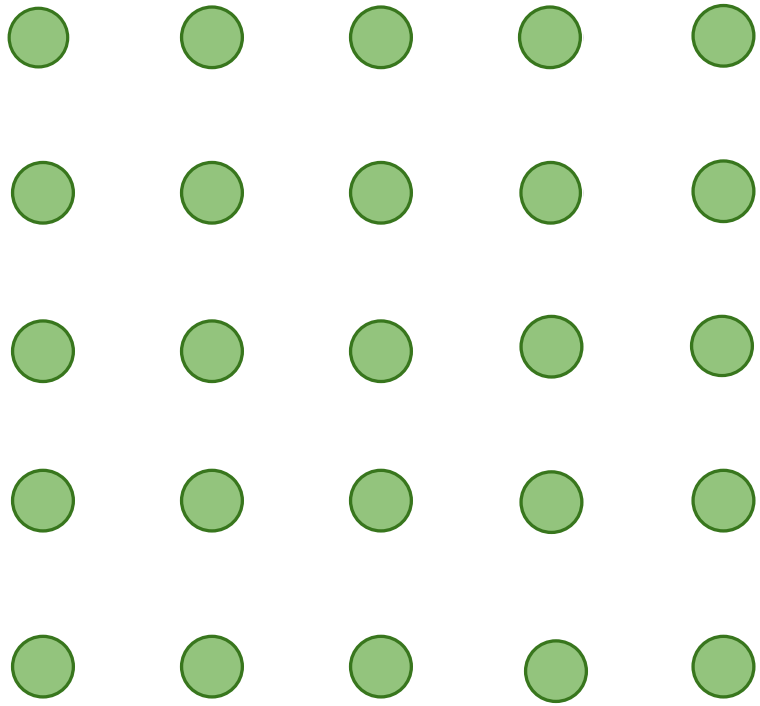


van der Oord *et al.* 2016

Pixel RNN

Let's now consider modeling an image.

1. Dependency on previous pixels modeled using an RNN (LSTM).
2. Implemented via Row LSTM and Diagonal BiLSTM.
3. For Row LSTM, generate image pixels row by row.
4. For Diagonal BiLSTM, generate image pixels starting from corner.

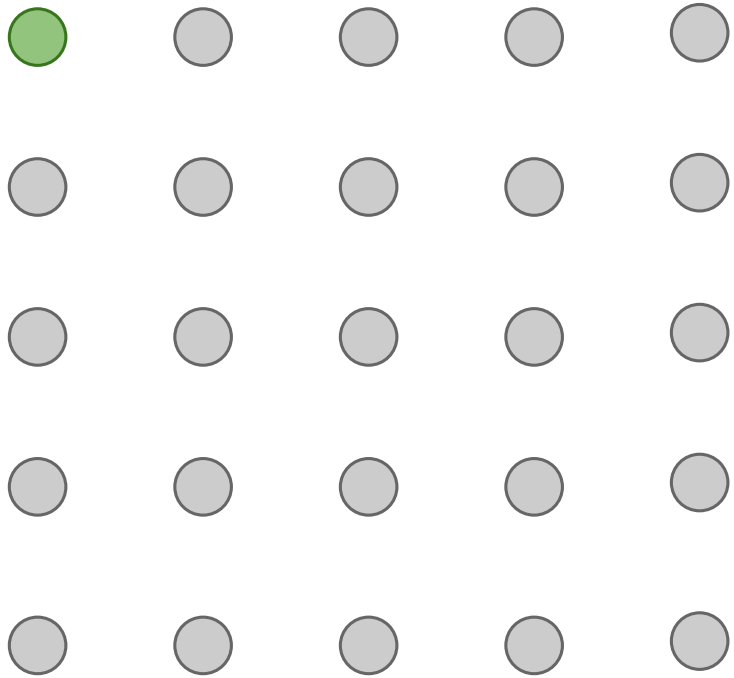


van der Oord *et al.* 2016

Pixel RNN

Let's now consider modeling an image.

1. Dependency on previous pixels modeled using an RNN (LSTM).
2. Implemented via Row LSTM and Diagonal BiLSTM.
3. For Row LSTM, generate image pixels row by row.
4. For Diagonal BiLSTM, generate image pixels starting from corner.

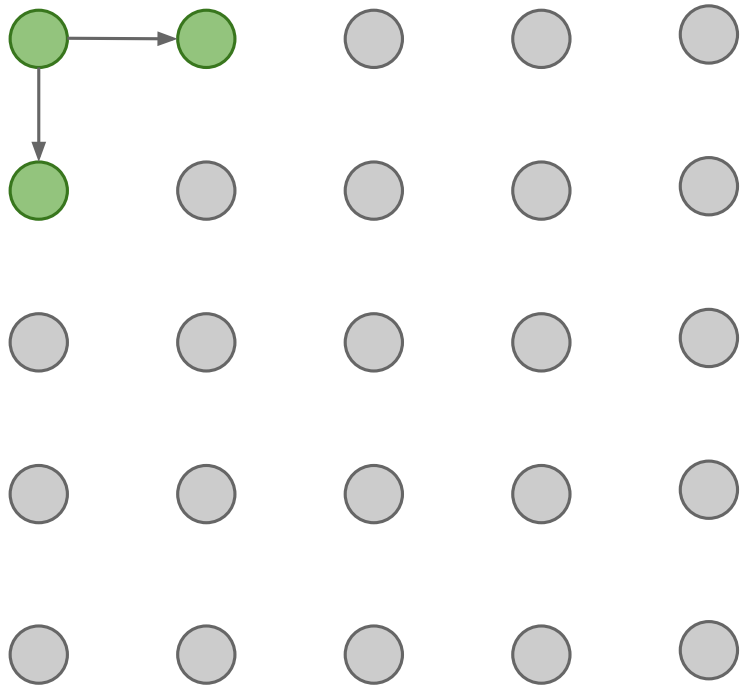


van der Oord *et al.* 2016

Pixel RNN

Let's now consider modeling an image.

1. Dependency on previous pixels modeled using an RNN (LSTM).
2. Implemented via Row LSTM and Diagonal BiLSTM.
3. For Row LSTM, generate image pixels row by row.
4. For Diagonal BiLSTM, generate image pixels starting from corner.

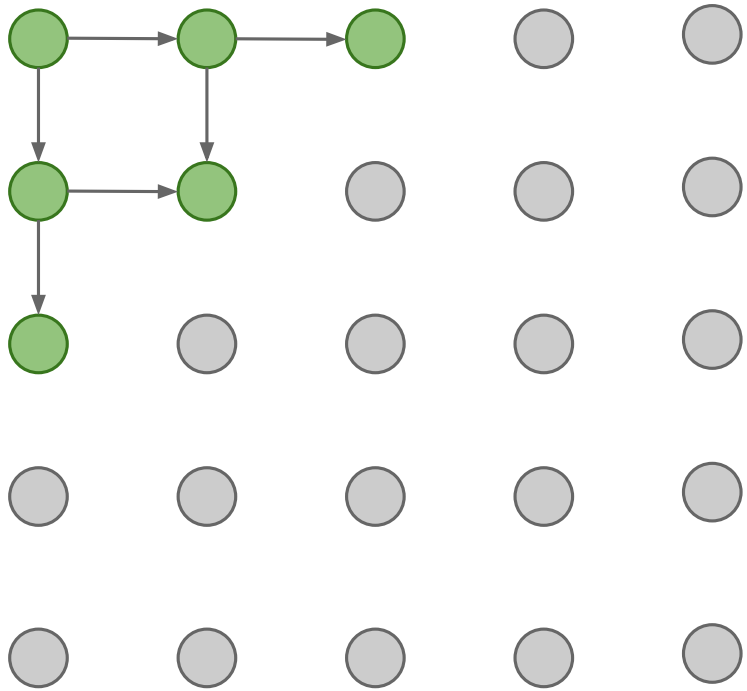


van der Oord *et al.* 2016

Pixel RNN

Let's now consider modeling an image.

1. Dependency on previous pixels modeled using an RNN (LSTM).
2. Implemented via Row LSTM and Diagonal BiLSTM.
3. For Row LSTM, generate image pixels row by row.
4. For Diagonal BiLSTM, generate image pixels starting from corner.

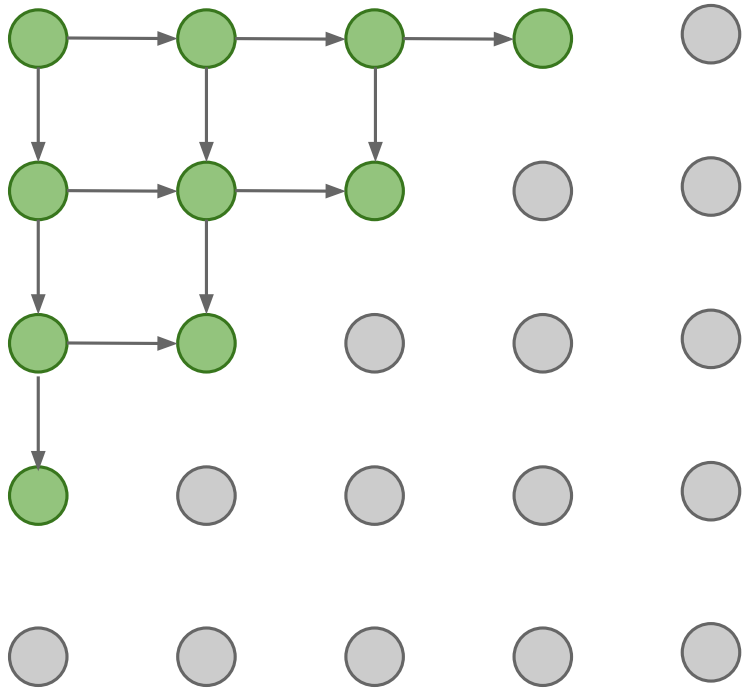


van der Oord *et al.* 2016

Pixel RNN

Let's now consider modeling an image.

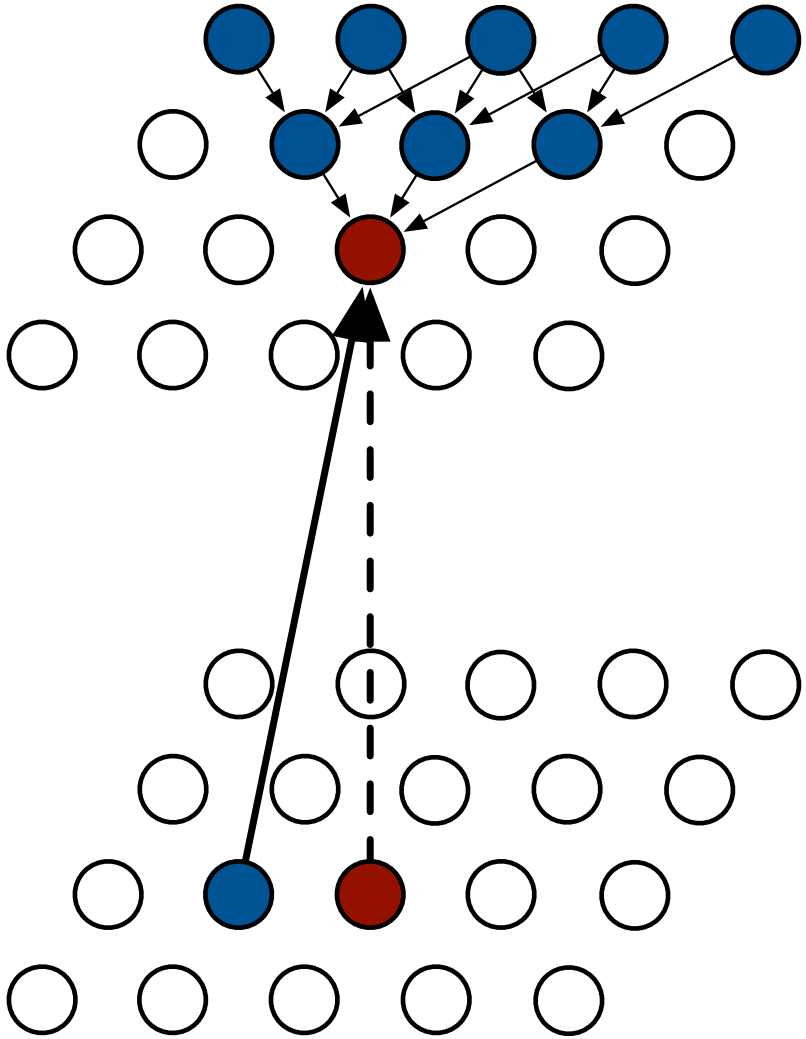
1. Dependency on previous pixels modeled using an RNN (LSTM).
2. Implemented via Row LSTM and Diagonal BiLSTM.
3. For Row LSTM, generate image pixels row by row.
4. For Diagonal BiLSTM, generate image pixels starting from corner.



van der Oord *et al.* 2016

Row LSTM

- $\mathbf{h}_i$: hidden representation of the i-th row – $h \times n$
- $\mathbf{x}_i$: input map of the i-th row, obtained by doing 1-D convolution on the original image – $h \times n$
- The figure shows the case of $h = 1$



$$[\mathbf{o}_i^j, \mathbf{f}_i^j, \mathbf{i}_i^j, \mathbf{g}_i^j] = \sigma\left(\mathbf{K}^{ss} * \mathbf{h}_{i-1}^{j-1:j+1} + \mathbf{K}^{is} * \mathbf{x}_i^{j-1:j}\right)$$

$$\mathbf{c}_i = \mathbf{f}_i \odot \mathbf{c}_{i-1} + \mathbf{i}_i \odot \mathbf{g}_i$$

$$\mathbf{h}_i = \mathbf{o}_i \odot \tanh(\mathbf{c}_i)$$

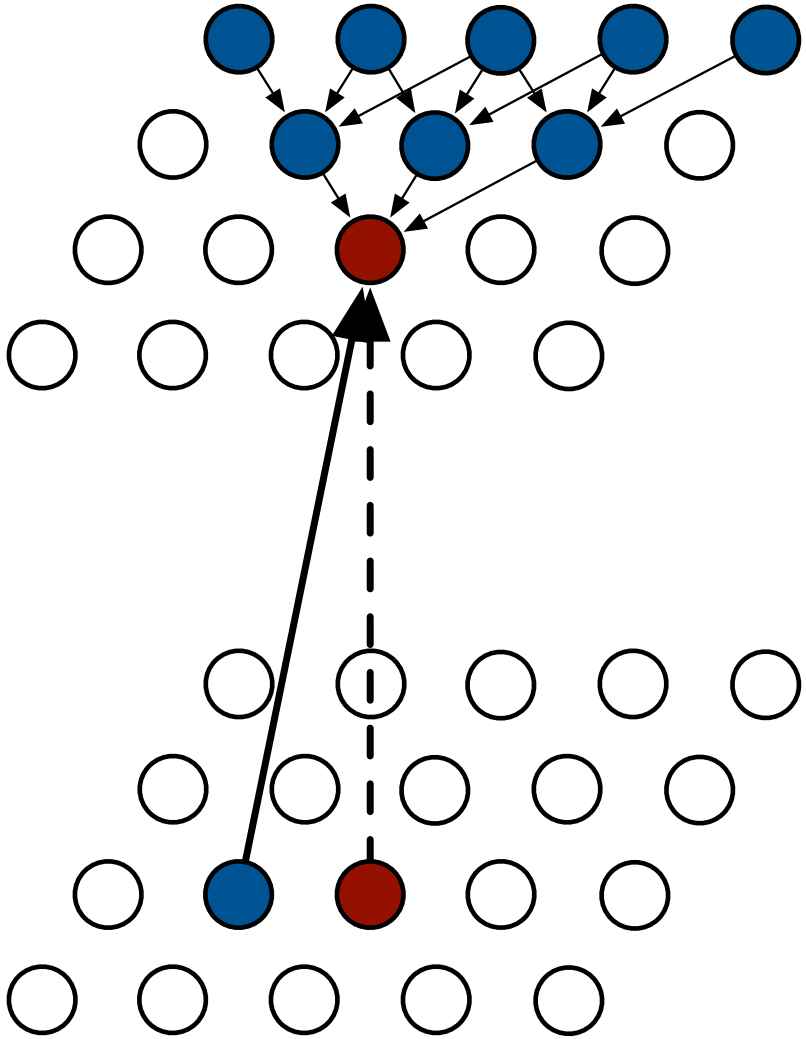
Row LSTM

$$[\mathbf{o}_i, \mathbf{f}_i, \mathbf{i}_i, \mathbf{g}_i] = \sigma\left(\mathbf{K}^{ss} * \mathbf{h}_{i-1} + \mathbf{K}^{is} * \mathbf{x}_i\right)$$

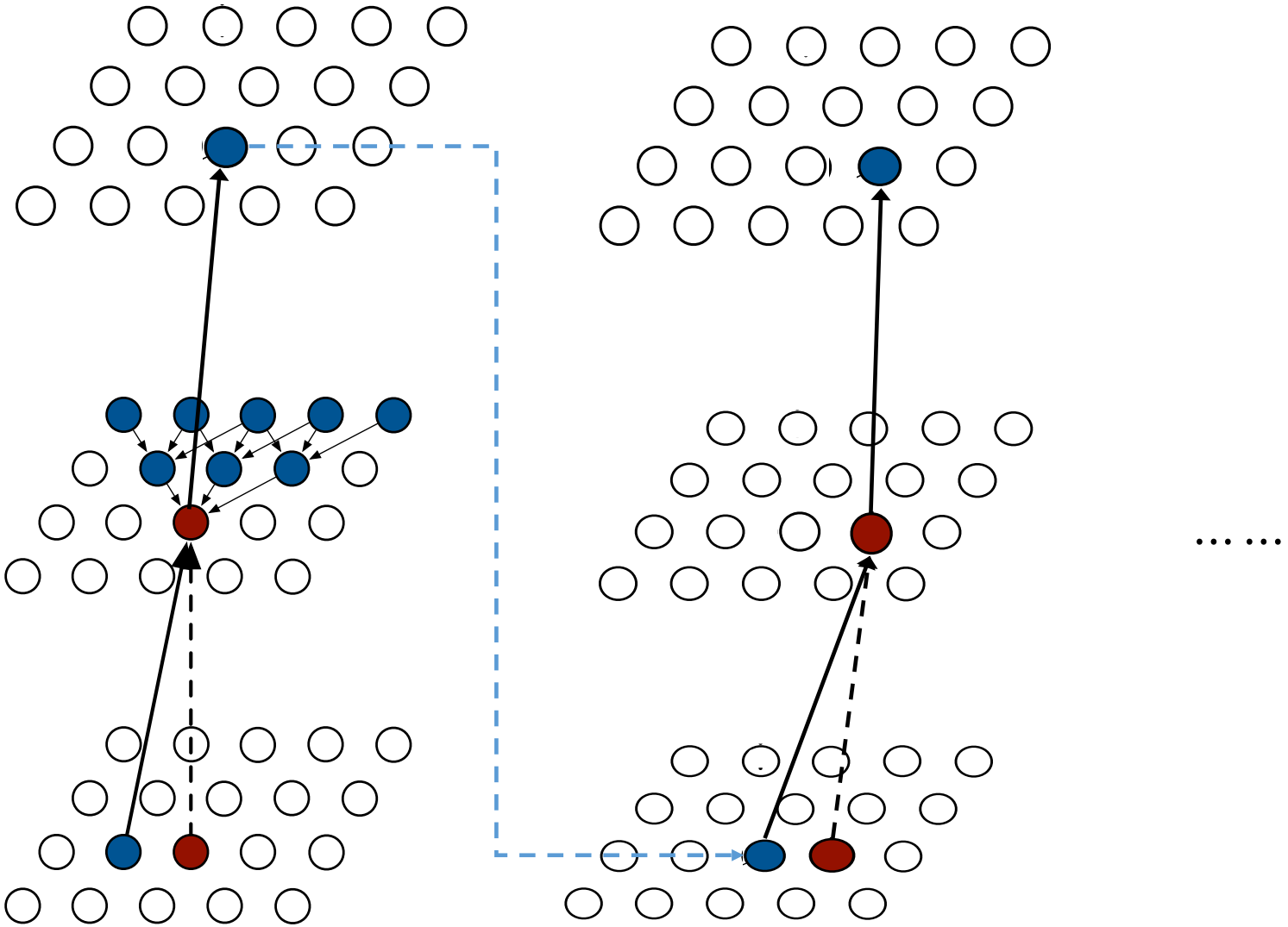
$$\mathbf{c}_i = \mathbf{f}_i \odot \mathbf{c}_{i-1} + \mathbf{i}_i \odot \mathbf{g}_i$$

$$\mathbf{h}_i = \mathbf{o}_i \odot \tanh(\mathbf{c}_i)$$

- Has a triangular receipt filed, unable to capture the entire available context.

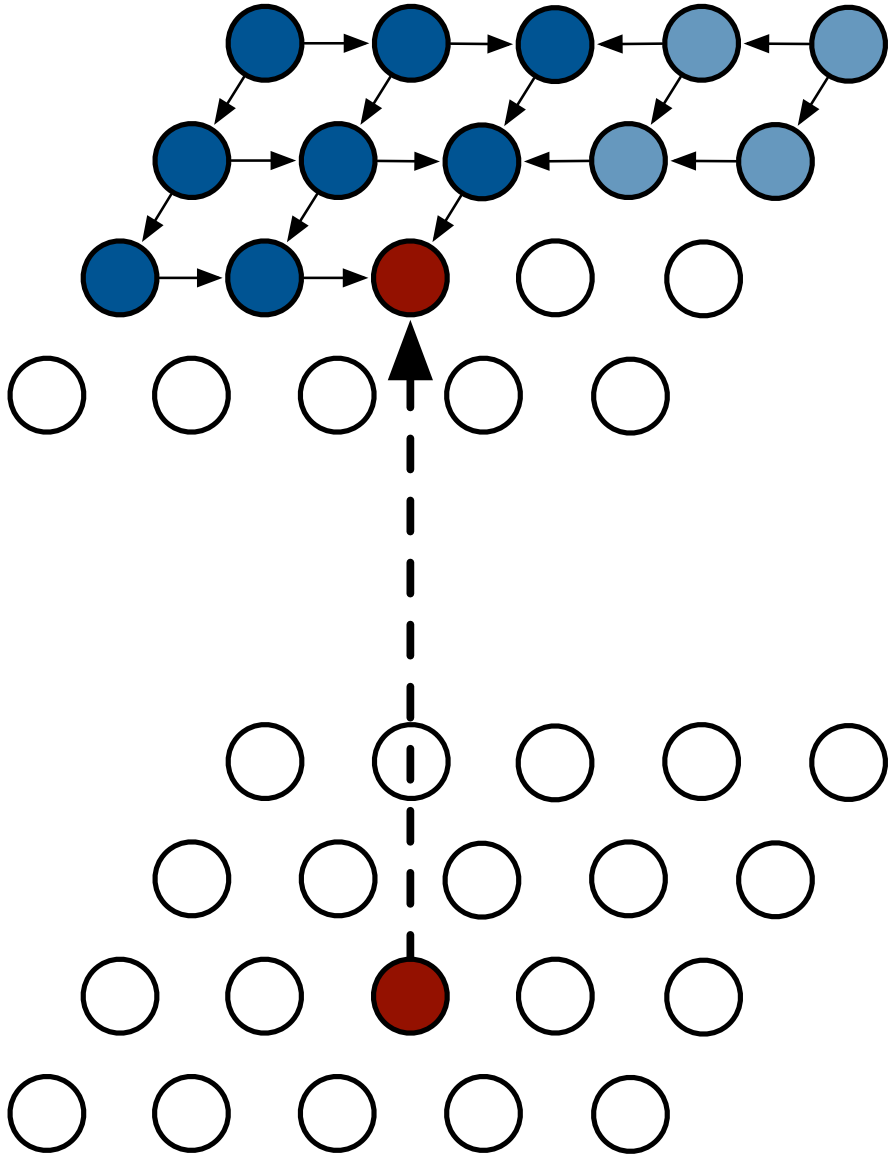


Row LSTM: Generation



Diagonal BiLSTM

- Be able to reach the whole dependent receipt field.



Pixel CNN

1. Still generate image pixels starting from corner.
2. Dependency on previous pixels now modeled using a CNN over context region.

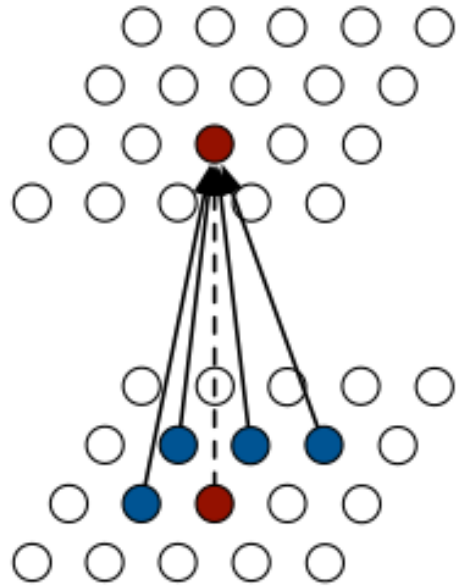
$$[\mathbf{o}_i, \mathbf{f}_i, \mathbf{i}_i, \mathbf{g}_i] = \sigma\left(\mathbf{K}^{ss} * \mathbf{h}_{i-1} + \mathbf{K}^{is} * \mathbf{x}_i\right)$$

$$\mathbf{c}_i = \mathbf{f}_i \odot \mathbf{c}_{i-1} + \mathbf{i}_i \odot \mathbf{g}_i$$

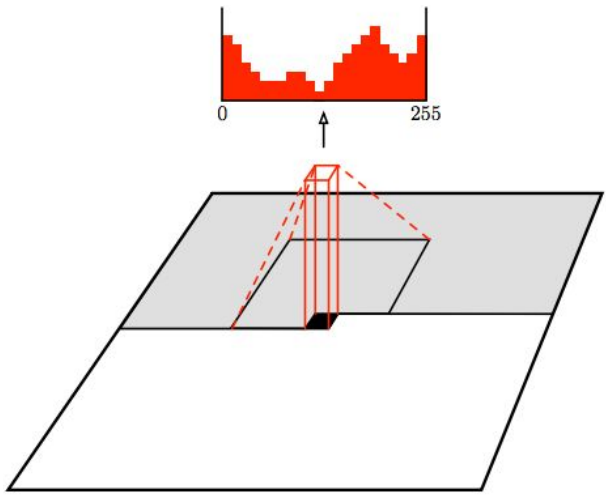
$$\mathbf{h}_i = \mathbf{o}_i \odot \tanh(\mathbf{c}_i)$$

$\Rightarrow$ CNN for $\mathbf{h}_i$

3. Training is faster than PixelRNN
   - can parallelize convolutions since context region values known from training images.
4. Generation must still proceed sequentially, thus slow.



PixelCNN



van der Oord *et al.* 2016

Pixel CNN

1. Still generate image pixels starting from corner.
2. Dependency on previous pixels now modeled using a CNN over context region.

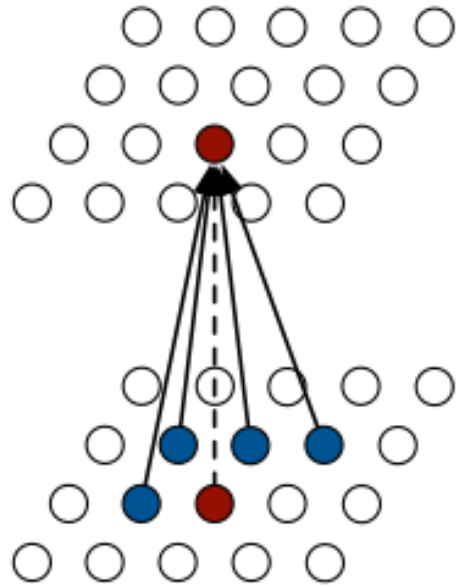
$$[\mathbf{o}_i, \mathbf{f}_i, \mathbf{i}_i, \mathbf{g}_i] = \sigma\left(\mathbf{K}^{ss} * \mathbf{h}_{i-1} + \mathbf{K}^{is} * \mathbf{x}_i\right)$$

$$\mathbf{c}_i = \mathbf{f}_i \odot \mathbf{c}_{i-1} + \mathbf{i}_i \odot \mathbf{g}_i$$

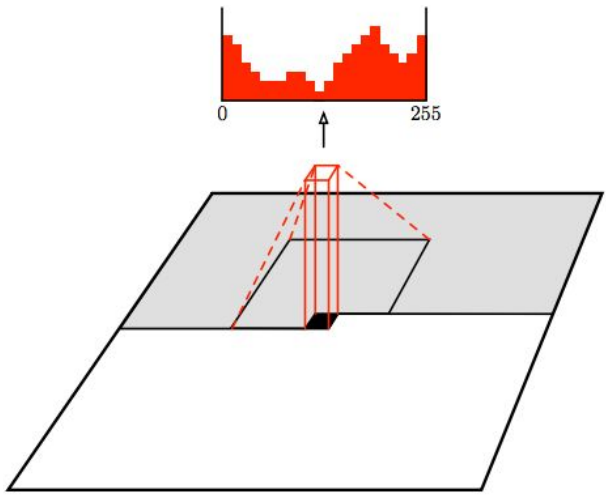
$$\mathbf{h}_i = \mathbf{o}_i \odot \tanh(\mathbf{c}_i)$$

$\Rightarrow$ CNN for $\mathbf{h}_i$

3. Training is faster than PixelRNN
   - can parallelize convolutions since context region values known from training images.
4. Generation must still proceed sequentially, thus slow.



PixelCNN



van der Oord *et al.* 2016

Pixel CNN

1. Still generate image pixels starting from corner.
2. Dependency on previous pixels now modeled using a CNN over context region.

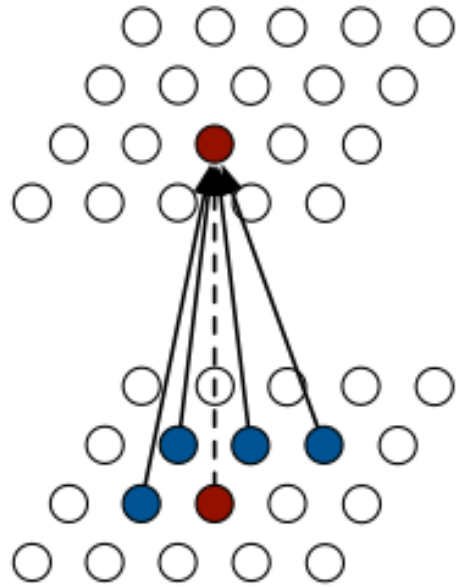
$$[\mathbf{o}_i, \mathbf{f}_i, \mathbf{i}_i, \mathbf{g}_i] = \sigma\left(\mathbf{K}^{ss} * \mathbf{h}_{i-1} + \mathbf{K}^{is} * \mathbf{x}_i\right)$$

$$\mathbf{c}_i = \mathbf{f}_i \odot \mathbf{c}_{i-1} + \mathbf{i}_i \odot \mathbf{g}_i$$

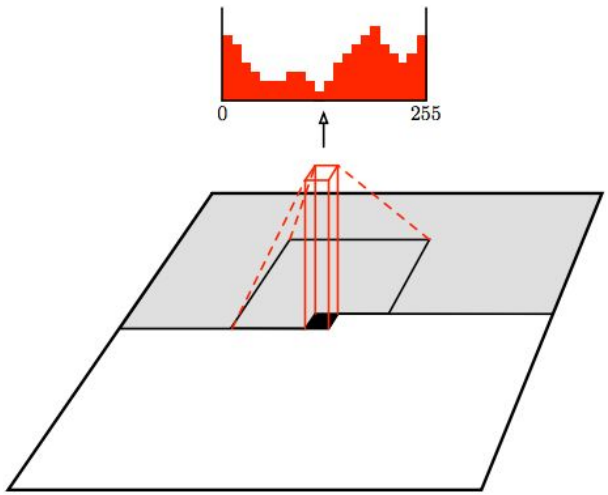
$$\mathbf{h}_i = \mathbf{o}_i \odot \tanh(\mathbf{c}_i)$$

$\Rightarrow$ CNN for $\mathbf{h}_i$

3. Training is faster than PixelRNN
   - can parallelize convolutions since context region values known from training images.
4. Generation must still proceed sequentially, thus slow.



PixelCNN



van der Oord *et al.* 2016

Pixel CNN

1. Still generate image pixels starting from corner.
2. Dependency on previous pixels now modeled using a CNN over context region.

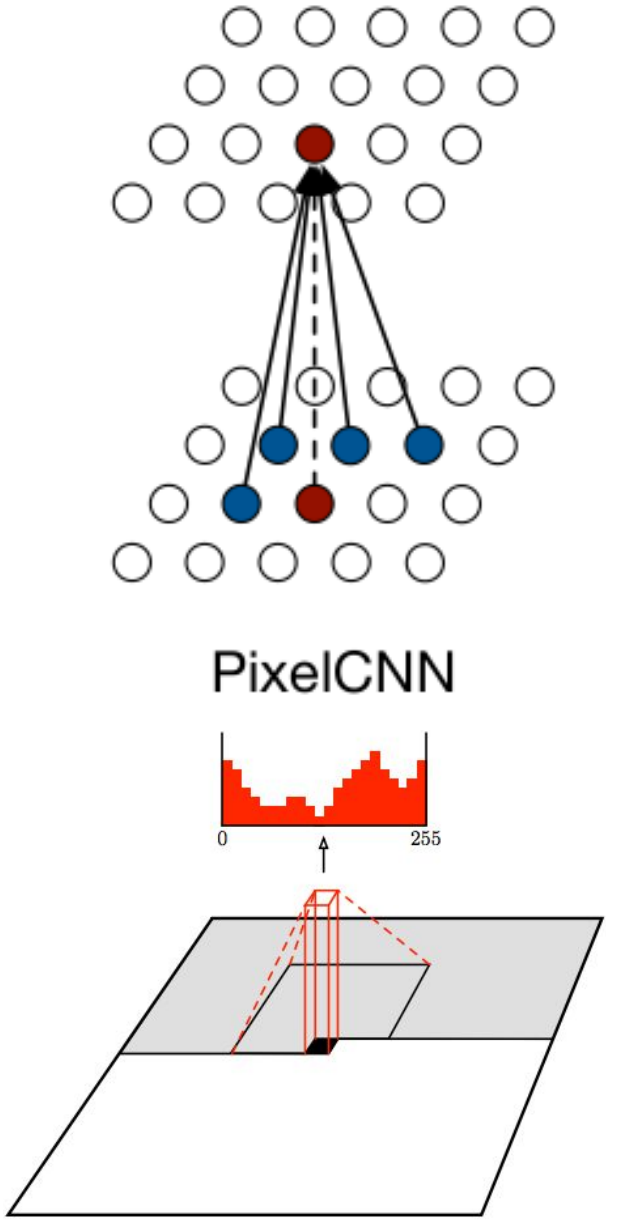
$$[\mathbf{o}_i, \mathbf{f}_i, \mathbf{i}_i, \mathbf{g}_i] = \sigma\left(\mathbf{K}^{ss} * \mathbf{h}_{i-1} + \mathbf{K}^{is} * \mathbf{x}_i\right)$$

$$\mathbf{c}_i = \mathbf{f}_i \odot \mathbf{c}_{i-1} + \mathbf{i}_i \odot \mathbf{g}_i$$

$$\mathbf{h}_i = \mathbf{o}_i \odot \tanh(\mathbf{c}_i)$$

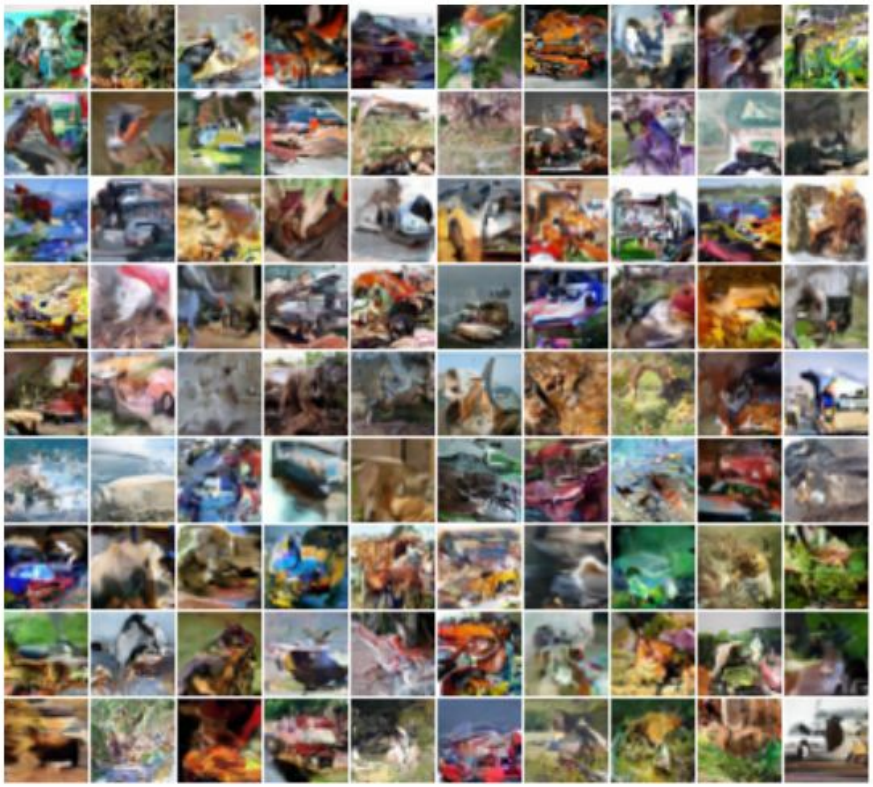
$\Rightarrow$ CNN for $\mathbf{h}_i$

3. Training is faster than PixelRNN
   - can parallelize convolutions since context region values known from training images.
4. Generation must still proceed sequentially, thus slow.

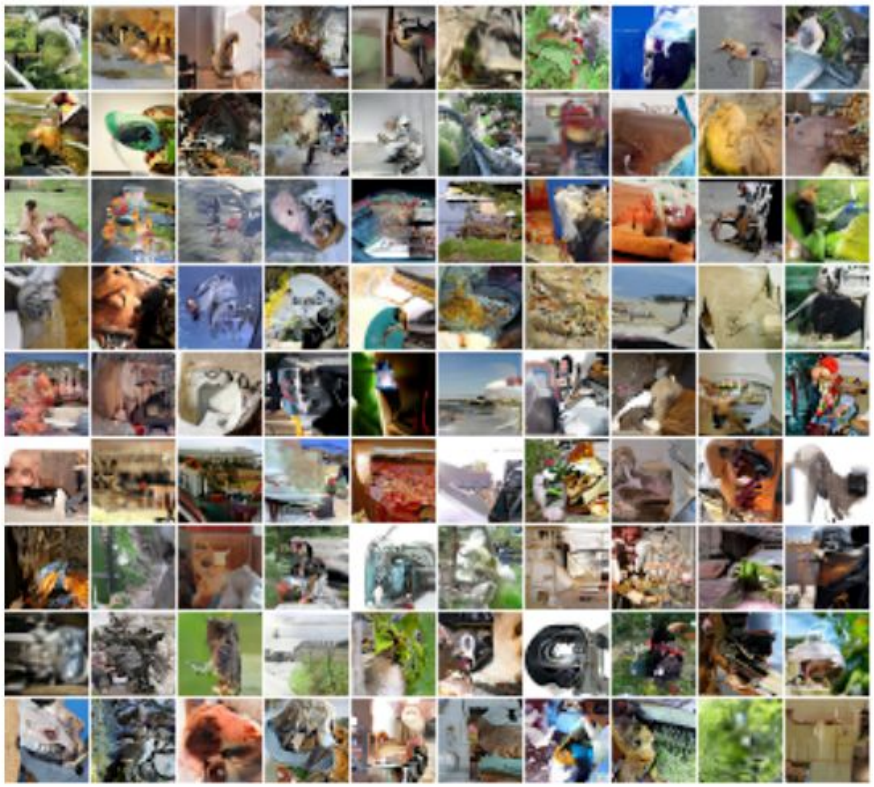


van der Oord *et al.* 2016

Generation Samples with Diagonal BiLSTM



32x32 CIFAR-10

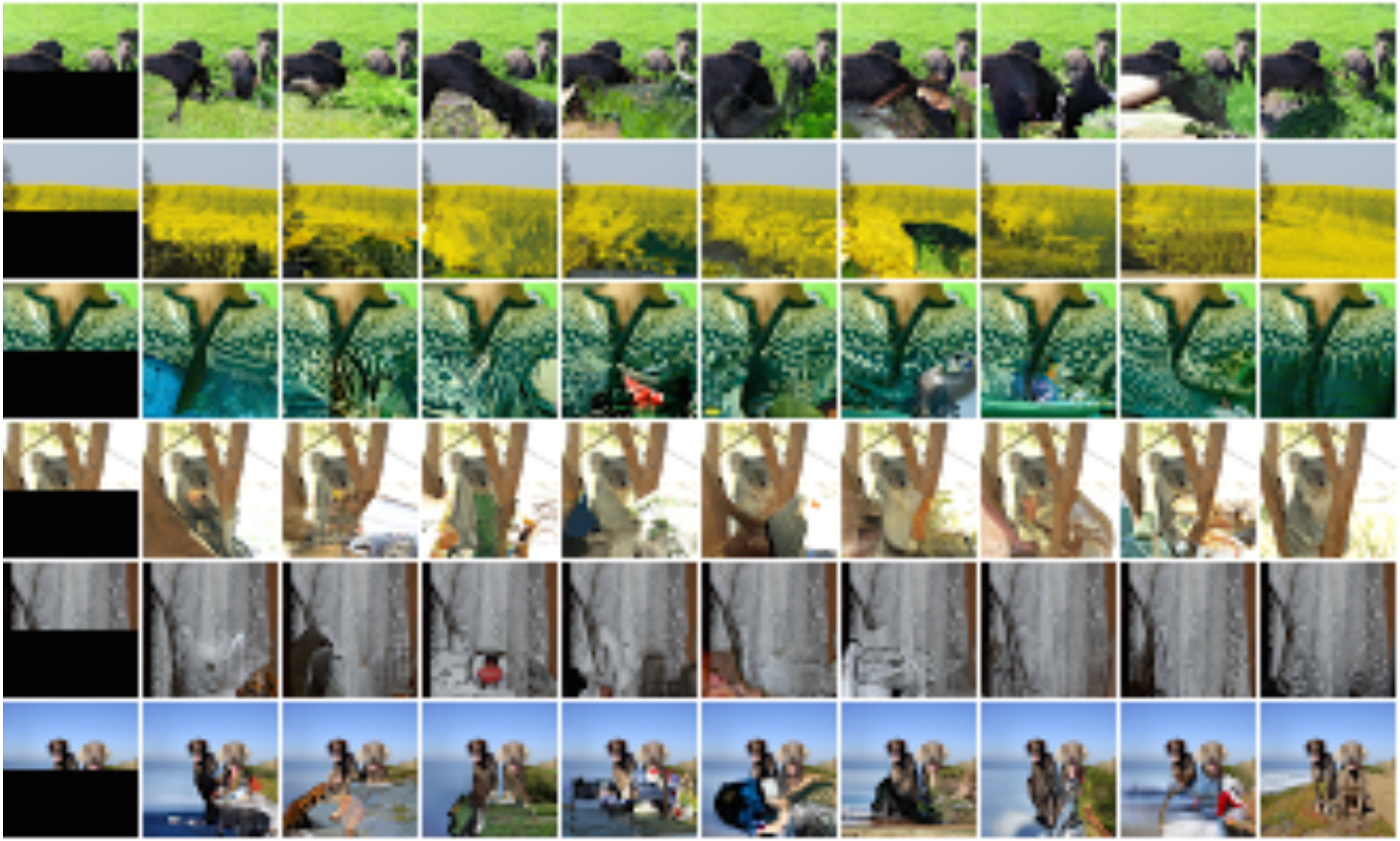


32x32 ImageNet

van der Oord *et al.* 2016

Image Completion

occluded completions original



Recap:Deep Generative Models

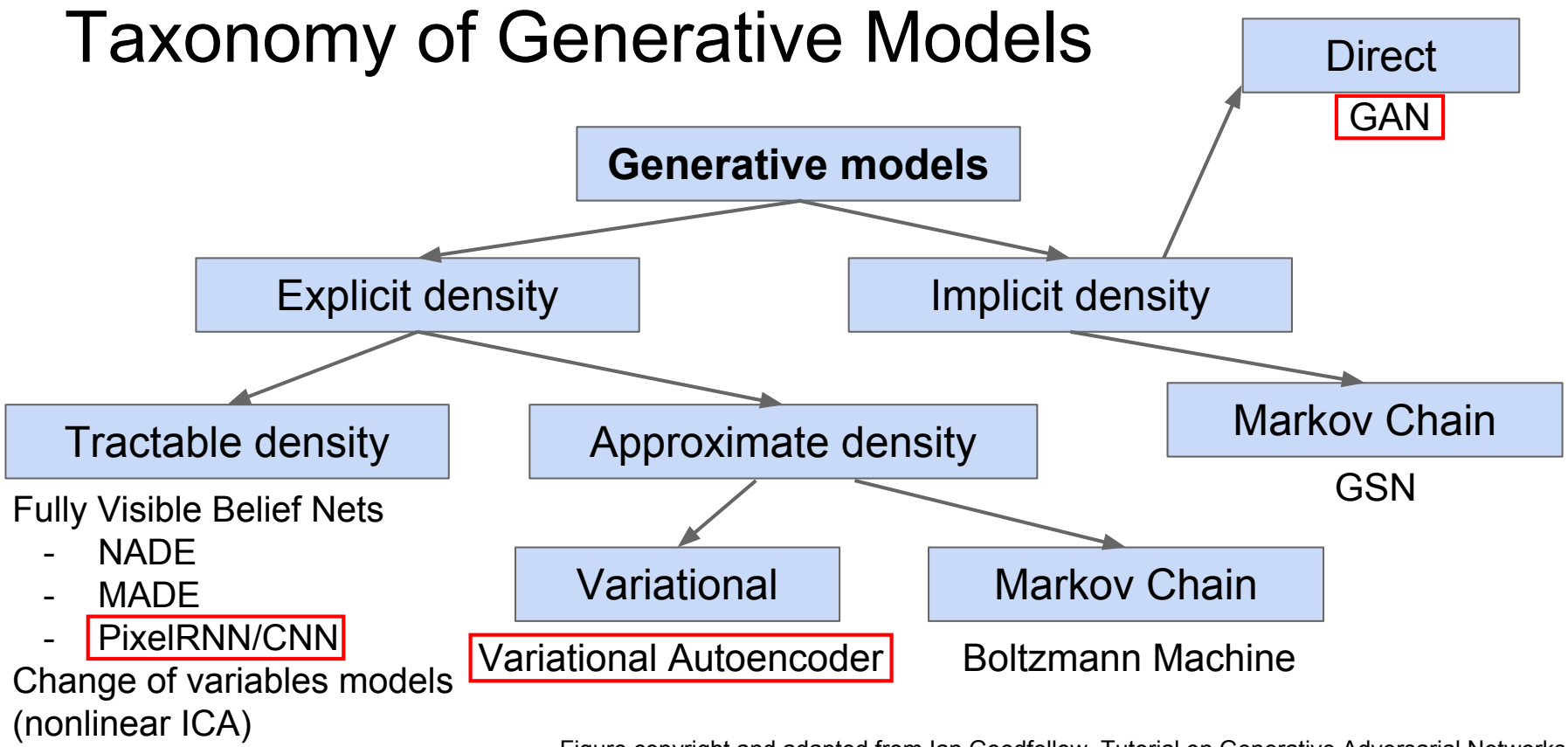


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Recap: Pixel RNN/CNN

Explicit density model

- Decompose likelihood of an image **x** into product of conditional distributions for all pixels:

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i | x_1, \cdots, x_{i-1})$$

Likelihood of image $\boldsymbol{x}$

Probability of i-th pixel given all previous pixels

- Solve by maximum likelihood estimator.
- Parameterize $p(x_i|x_1, \cdots, x_{i-1})$ with a neural network to represent a complex distribution:
  - Use RNN and CNN.

Variational Autoencoders[2] (VAE)

[2] Partially adapted from http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

VAE Overlook

1. PixelCNNs define tractable density function; Perform learning via maximum likelihood estimator:

$$p_\theta(\mathbf{x}) = \prod_{i=1}^{n} p_\theta(x_i | x_1, \cdots, x_{i-1})$$

2. VAEs introduce latent variable $\mathbf{z}$ into the model; the likelihood is an intractable integration:

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{z}) p_\theta(\mathbf{x} \mid \mathbf{z}) \mathrm{d}\, \mathbf{z}$$

3. Learning/inference has to resorted to some approximations, *e.g.*, by optimizing on a lower bound of $p_\theta(\mathbf{x})$:
   - variational inference

VAE Overlook

1. PixelCNNs define tractable density function; Perform learning via maximum likelihood estimator:

$$p_\theta(\mathbf{x}) = \prod_{i=1}^{n} p_\theta(x_i | x_1, \cdots, x_{i-1})$$

2. VAEs introduce latent variable $\mathbf{z}$ into the model; the likelihood is an intractable integration:

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{z}) p_\theta(\mathbf{x} \,|\, \mathbf{z}) \mathrm{d}\, \mathbf{z}$$

3. Learning/inference has to resorted to some approximations, *e.g.*, by optimizing on a lower bound of $p_\theta(\mathbf{x})$:
   - variational inference

VAE Overlook

1. PixelCNNs define tractable density function; Perform learning via maximum likelihood estimator:

$$p_\theta(\mathbf{x}) = \prod_{i=1}^{n} p_\theta(x_i | x_1, \cdots, x_{i-1})$$

2. VAEs introduce latent variable $\mathbf{z}$ into the model; the likelihood is an intractable integration:

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{z}) p_\theta(\mathbf{x} \,|\, \mathbf{z}) \mathrm{d}\, \mathbf{z}$$

3. Learning/inference has to resorted to some approximations, *e.g.*, by optimizing on a lower bound of $p_\theta(\mathbf{x})$:
   - variational inference

Background: Variational Inference[3] (VAE)

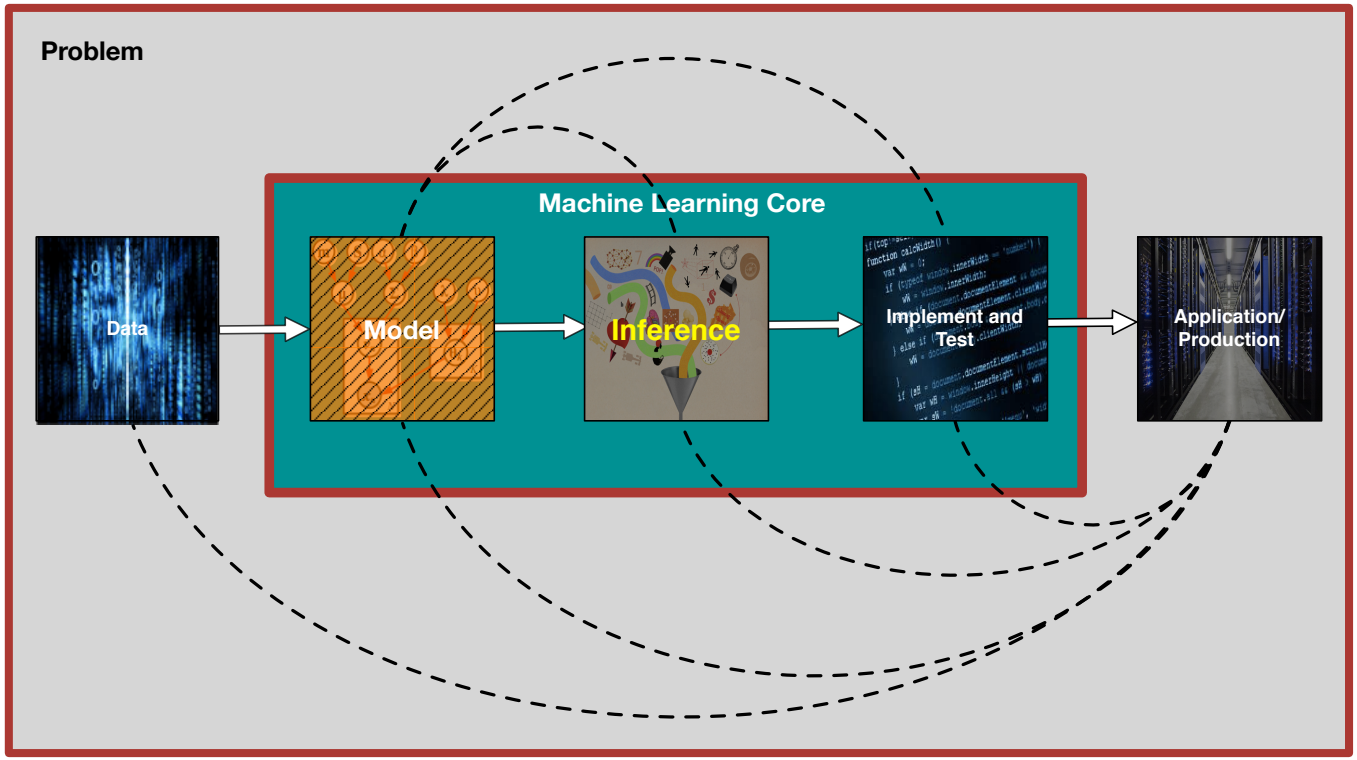
[3] Partially adapted from http://shakirm.com/papers/VITutorial.pdf

Probability Reasoning

Modern applications and data strongly favour probabilistic modelling:

- Noise in the data and account for our lack of knowledge.
- Non-i.i.d., non-stationary data.
- Explore and extract the underlying structure in the data.
- Consistency in our beliefs about the data and systems we study.

Probabilistic Inference



In probabilistic models, we must reason over the probability of events

Statistical Inference

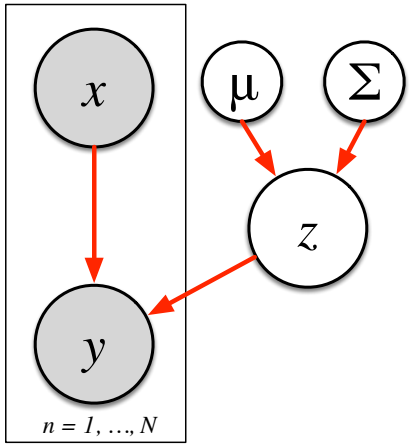
Any mechanism by which we deduce the probabilities in our model based on data.

Inference links the observed data with our statistical assumptions and allows us to ask questions of our data: predictions, visualisation, model selection.

Modeling and Inference

Probabilistic modelling will involve:

- Decide on a priori beliefs.
- Posit an explanation of how the observed data is generated, i.e. provide a probabilistic description.



Regression: Linear combination of inputs to give response.

Bayes' rule highlights many of the inferential problems we will face.

$$\underbrace{p(z|y)}_{\text{Posterior}} = \frac{\overbrace{p(y|z)}^{\text{Likelihood}}\ \overbrace{p(z)}^{\text{Prior}}}{\underbrace{\int p(y,z)dz}_{\text{Marginal likelihood/ Model evidence}}}$$

Inference Problems

Most inference problems will be one of:

- Marginalization:

$$p(y) = \int p(y, \theta) \mathrm{d}\theta$$

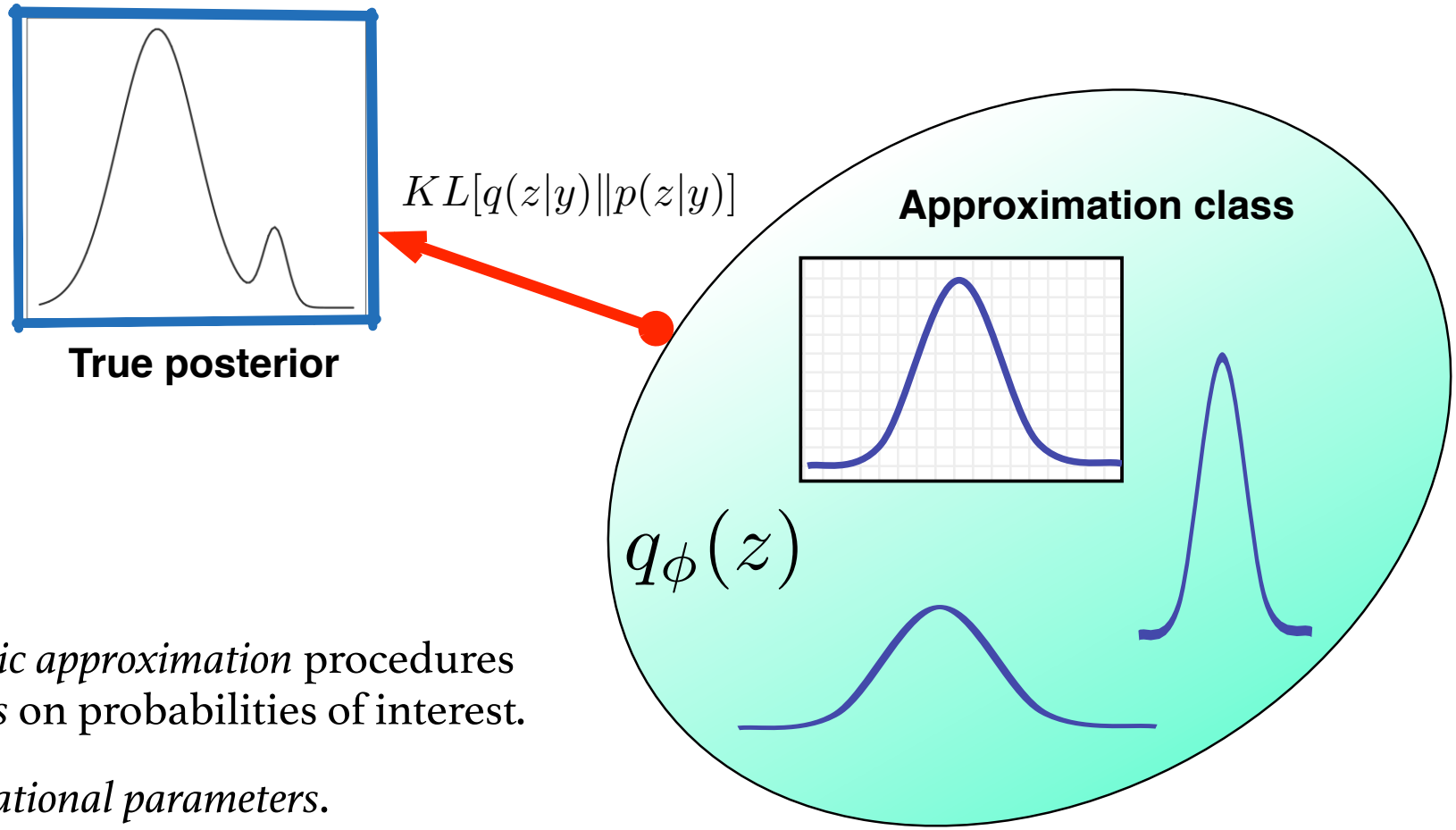
- Expectation:

$$\mathbb{E}\left[f(y)|x\right] = \int f(y) p(y|x) \mathrm{d}y$$

- Prediction:

$$p(y_{t+1}) = \int p(y_{t+1}|y_t) p(y_t) \mathrm{d}y_t$$

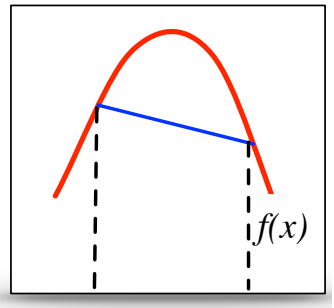
What is a Variational Method?



Deterministic *approximation* procedures with *bounds* on probabilities of interest.

Fit the *variational parameters*.

Jensen's Inequality



An important result from convex analysis:

For concave functions *f(.)*

$$f(\mathbb{E}[x]) \geq \mathbb{E}[f(x)]$$

Logarithms are strictly *concave* allowing us to use Jensen's inequality.

$$\log \int p(x)g(x)dx \geq \int p(x) \log g(x)dx$$

Variational Inference

Integral problem

$$\log p(y) = \log \int p(y|z)p(z)dz$$

Proposal

$$\log p(y) = \log \int p(y|z)p(z)\frac{q(z)}{q(z)}dz$$

Importance Weight

$$\log p(y) = \log \int p(y|z)\frac{p(z)}{q(z)}q(z)dz$$

Jensen's inequality

$$\log \int p(x)g(x)dx \geq \int p(x) \log g(x)dx$$

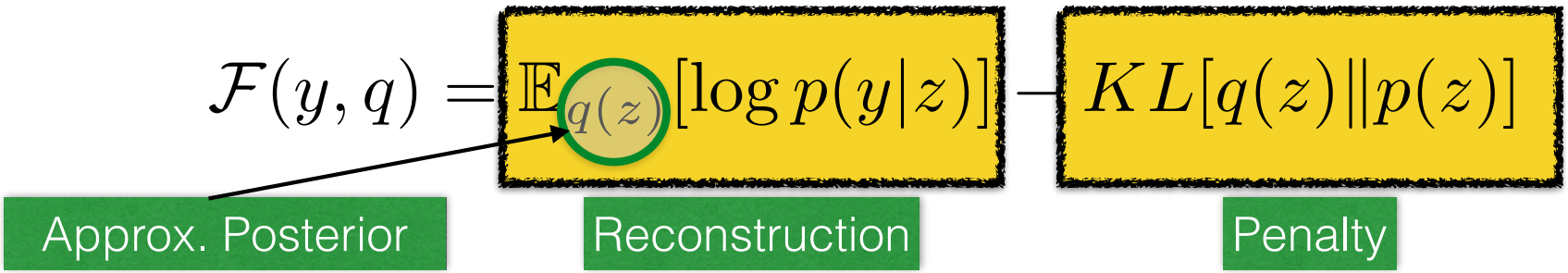
$$\log p(y) \geq \int q(z) \log \left(p(y|z)\frac{p(z)}{q(z)} \right) dz$$

$$= \int q(z) \log p(y|z) - \int q(z) \log \frac{q(z)}{p(z)}$$

Variational lower bound

$$= \mathbb{E}_{q(z)}[\log p(y|z)] - KL[q(z)\|p(z)]$$

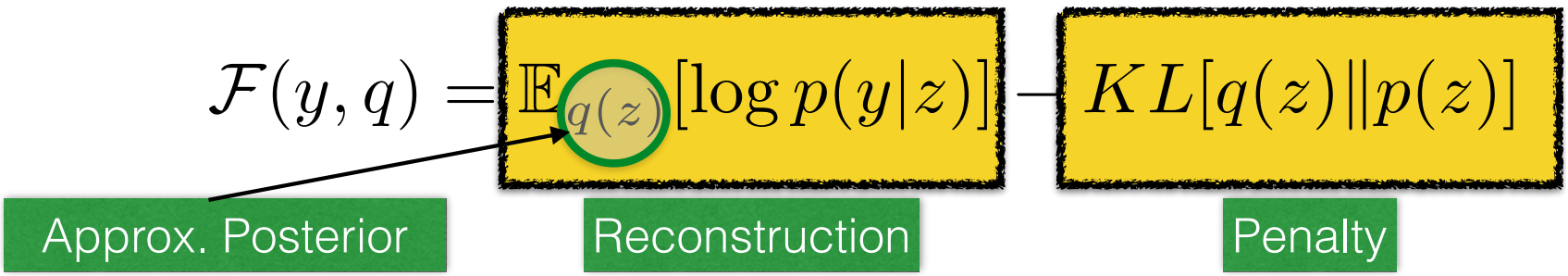
Variational Inference



Interpretation

- Approximate posterior distribution $q(z)$: Best match to true posterior $p(z|y)$, one of the unknown inferential quantities of interest to us.
- Reconstruction cost: The expected log-likelihood measure how well samples from $q(z)$ are able to explain the data y.
- Penalty: Ensures the the explanation of the data $q(z)$ does not deviate too far from your beliefs $p(z)$. A mechanism for realizing Okham's razor.

Variational Inference



Some comments on q

- Integration is now optimization: optimize for $q(z)$ directly:

  z typically depends on data y, but write $q(z)$ for notation simplicity. Easy convergence assessment.

- Variational parameters: parameters of $q(z)$:

  e.g., if a Gaussian, the parameters are mean and variance. Optimization allows us to tighten the bound and get as close as possible to the true marginal likelihood.

Free-form and Fixed-form Solutions

Free-form

- Solves for the exact distribution $q(z)$ by setting the functional derivative to zero via calculus of variations:

$$\frac{\delta \mathcal{F}(y, q)}{\delta q(z)} = 0, \quad s.t. \quad \int q(z) \mathrm{d}z = 1$$
$$\Rightarrow q(z) \propto p(z)p(y|z, \theta)$$

- The optimal solution is the true posterior distribution, but solving for the normalization is our original problem.

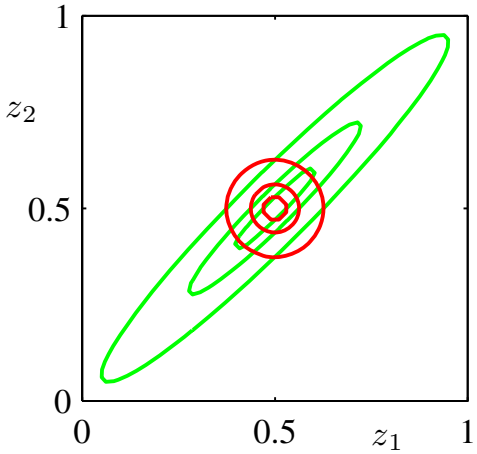
Fixed-form

- Specify an explicit form of $q(z)$, *e.g.*, a normal distribution.
- Parameter in q is called the variational parameter.

Mean-field Variational Inference

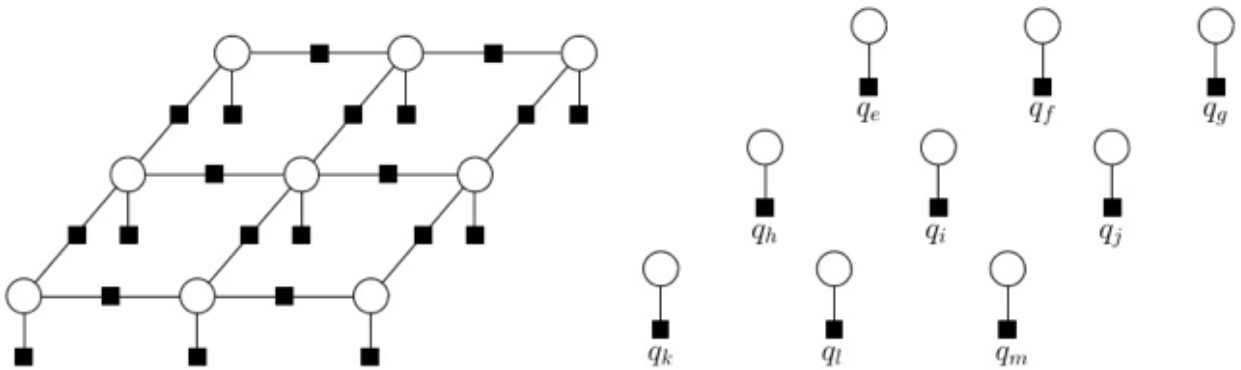
Mean-field methods assume that the distribution is factorised.

$$q(z) = \prod_i q_i(z_i)$$



Restricted class of approximations: every dimension (or subset of dimensions) of the posterior is independent.

$$q(z) = \prod_i \mathcal{N}(z_i|\mu_i, \sigma_i^2)$$



Example: Mean-field for Latent Gaussian Models

Generative model:

$$\mathbf{z} \sim \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I}), \quad \mathbf{y} \sim p(\mathbf{y} \,|\, f_\theta(\mathbf{z}))$$

Variational distribution:

$$q(\mathbf{z}) = \prod_i \mathcal{N}(z_i; \mu_i, \sigma_i^2)$$

Example: Mean-field for Latent Gaussian Models

$$
\begin{aligned}
\mathcal{F}(\mathbf{y}, q) &= \mathbb{E}_{q(\mathbf{z})}\left[\log p(\mathbf{y} \mid \mathbf{z})\right] - KL\left[q(\mathbf{z}) \| p(\mathbf{z})\right] \\
&= \mathbb{E}_{q(\mathbf{z})}\left[\log p(\mathbf{y} \mid \mathbf{z})\right] - \sum_i KL\left[q(z_i) \| p(z_i)\right] \\
&= \mathbb{E}_{q(\mathbf{z})}\left[\log p(\mathbf{y} \mid \mathbf{z})\right] - \sum_i KL\left[\mathcal{N}(z_i; \mu_i, \sigma_i^2) \| \mathcal{N}(z_i; 0, 1)\right] \\
&= \mathbb{E}_{q(\mathbf{z})}\left[\log p(\mathbf{y} \mid \mathbf{z})\right] - \frac{1}{2}\sum_i \left(\sigma_i^2 + \mu_i^2 - 1 - \log \sigma_i^2\right)
\end{aligned}
$$

KL between two Gaussians

Let $p(x) = \mathcal{N}(x; \mu_1, \sigma_1^2)$, $q(x) = \mathcal{N}(x; \mu_2, \sigma_2^2)$, then

$$
KL(p(x) \| q(x)) = \log \frac{\sigma_2^2}{\sigma_1^2} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}
$$

Example: Mean-field for Latent Gaussian Models

$$
\begin{aligned}
\mathcal{F}(\mathbf{y}, q) &= \mathbb{E}_{q(\mathbf{z})}\left[\log p(\mathbf{y} \mid \mathbf{z})\right] - KL\left[q(\mathbf{z}) \| p(\mathbf{z})\right] \\
&= \mathbb{E}_{q(\mathbf{z})}\left[\log p(\mathbf{y} \mid \mathbf{z})\right] - \sum_i KL\left[q(z_i) \| p(z_i)\right] \\
&= \mathbb{E}_{q(\mathbf{z})}\left[\log p(\mathbf{y} \mid \mathbf{z})\right] - \sum_i KL\left[\mathcal{N}(z_i; \mu_i, \sigma_i^2) \| \mathcal{N}(z_i; 0, 1)\right] \\
&= \mathbb{E}_{q(\mathbf{z})}\left[\log p(\mathbf{y} \mid \mathbf{z})\right] - \frac{1}{2}\sum_i \left(\sigma_i^2 + \mu_i^2 - 1 - \log \sigma_i^2\right)
\end{aligned}
$$

Optimize parameters of $q(z)$ by gradient descent.

Optimization for the Variational Bound

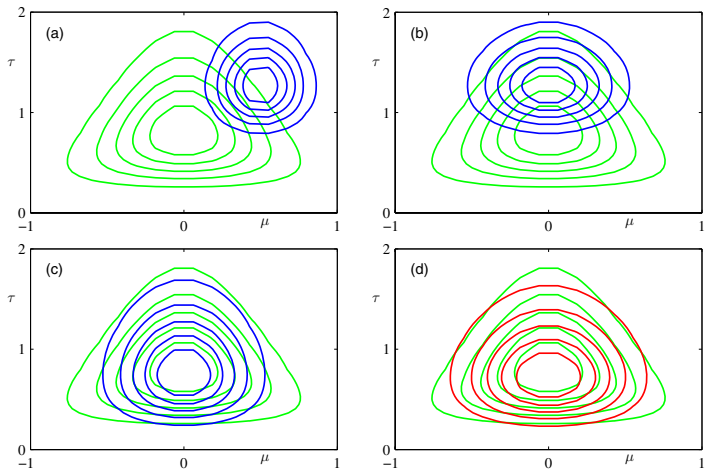
$$\max_{q,\theta} \mathcal{F}(y, q) = \mathbb{E}_{q(z)}[\log p(y|z, \theta)] - KL[q(z)||p(z)]$$

Approx. Posterior

Reconstruction

Penalty

- *Variational EM*
- *Stochastic Variational Inference*
- *Doubly Stochastic Variational Inference*
- *Amortised Inference*



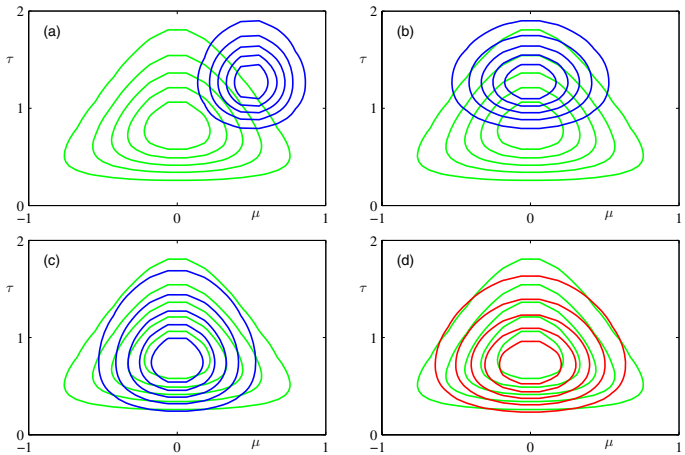
Optimization for the Variational Bound

$$\max_{q,\theta} \mathcal{F}(y, q) = \mathbb{E}_{q(z)}[\log p(y|z, \theta)] - KL[q(z)\|p(z)]$$

Approx. Posterior | Reconstruction | Penalty

- *Variational EM*

- *Stochastic Variational Inference*
- *Doubly Stochastic Variational Inference*
- *Amortised Inference*



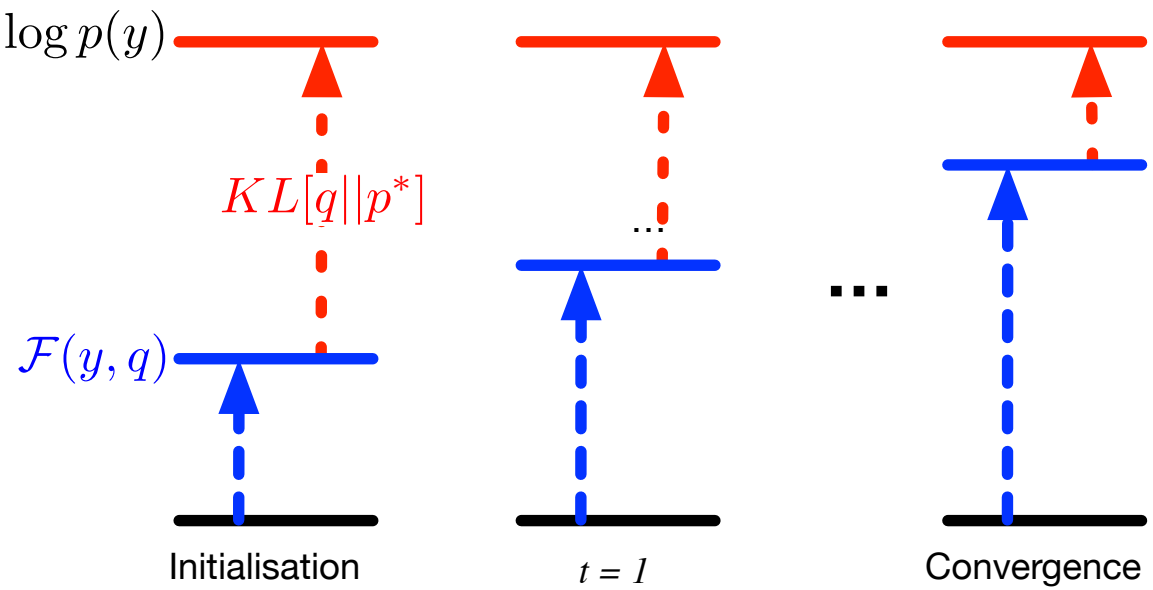
Dealt with big data and complicated $q(z)$

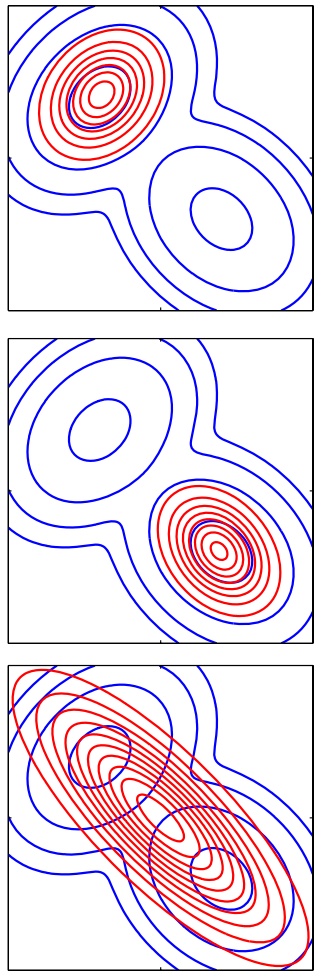
Variational Expectation Maximization

Alternating optimization for the variational parameters and then model parameters.

Repeat:

E-step	$\phi \propto \nabla_\phi \mathcal{F}(y, q)$	*Var. params*
M-step	$\theta \propto \nabla_\theta \mathcal{F}(y, q)$	*Model params*





Variational Expectation Maximization

Alternating optimization for the variational parameters and then model parameters.

Repeat:

E-step *(Inference)*

For i = 1, ... N

$$\phi_n \propto \nabla_\phi \mathbb{E}_{q_\phi(z)}[\log p_\theta(y_n|z_n)] - \nabla_\phi KL[q(z_n)||p(z_n)]$$

M-step *(Parameter Learning)*

$$\theta \propto \frac{1}{N}\sum_n \mathbb{E}_{q_\phi(z)}[\nabla_\theta \log p_\theta(y_n|z_n)]$$