Lecture #10 – Deep Generative Models
Part I

Aykut Erdem // Hacettepe University // Spring 2018
Previously on CMP784

- Supervised Representation Learning
- Unsupervised Representation Learning
- Sparse Coding
- Auto-encoders
Lecture overview

- Why Generative Models
- Types of Generative Models
- Autoregressive Generative Models
- Variational Autoencoders

Disclaimer: Much of the material and slides for this lecture were borrowed from

— Shakir Mohamed’s talk on “Building Machines that Image and Reason”
— Nal Kalchbrenner’s talks on “Generative Modelling as Sequence Learning” and “Generative Models of Language and Images”
— Russ Salakhutdinov’s CMU 10807 class slides
Why Generative Models
Why Generative Models

Move beyond associating inputs to outputs

Recognize objects in the world and their factors of variation

Establish concepts as useful for reasoning and decision making

Understand and imagine how the world evolves

Detect surprising events in the world

Imagine and generate rich plans for the future

Part of a suite of complementary learning systems
Semi-supervised Classification

In Figure 2 we show the results of that experiment for the "circles" dataset (plots for the other two clusters), the "circles" arrangement (again containing two clusters) and a simple dataset with three clusters. The results for the "two moons" dataset (which contains two clusters) are also shown. Additionally, the generator quickly learns to generate the datasets in all cases. As shown in Figure 2, the CatGAN algorithm performs better than standard k-means clustering and RIM with neural networks.

We either used the full set of labeled examples or a reduced set of labeled examples and kept the remaining examples for semi-supervised learning. On the other hand, the discriminator has to place its decision boundaries such that it can easily detect the correct class assignments without any geometric constraints.

To exemplify the power of the ADGM for semi-supervised learning, we compare the CatGAN algorithm with standard k-means clustering and RIM with neural networks. The results are also comparable to the best model. The A VAE is performing better than the V AE and the IWAE, L=2, IW=1 model.

As shown in Figure 2, the CatGAN algorithm performs better than standard k-means clustering and RIM with neural networks.

Communication and Compression

We finally consider a more complex setup, where we infer the counts, identities and positions of a single object in a scene. We demonstrate the capabilities of this approach by first considering scenes consisting of only one object, such as a red cube, a blue sphere, and a textured cylinder. We train a single-step AIR model to infer the identity and pose of the object present in the image. We show that AIR can learn to perform this task with as little supervision as possible, and indeed we observe that with direct optimization of the likelihood from scratch for every test image (Fig. 6d), and observe that AIR is imposed by the labels, however training is more straightforward when there are no repetitions. AIR is a problem in general. We provide a quantitative comparison of AIR's inference robustness and accuracy to simple methods such as counting the objects in a scene and learning to count, locate, and identify them. Reconstructed AIR's inferences on generated data, as well as real images of a table with varying numbers of plates, in Fig. 6 and Fig. 7. AIR's inferences of counts, identities and positions are reconstructions of AIR's inferences on generated data, as well as real images of a table with varying numbers of plates, in Fig. 6 and Fig. 7. AIR's inferences of counts, identities and positions are multi-modal and difficult to represent using standard network architectures. We also compare the performance of AIR to other methods such as discriminative and generative methods, and observe that AIR produces significantly better reconstructions and count accuracies than a supervised method on data.

Figure 6: (b) AIR produces significantly better reconstructions and count accuracies than a supervised method on data. (c) Reconstructed AIR's inferences on generated data, as well as real images of a table with varying numbers of plates. (d) Reconstructions after direct gradient descent. This approach is less stable and much more susceptible to local minima. (e) Generated and real reconstructions of AIR's inferences on generated data, as well as real images of a table with varying numbers of plates. (f) AIR's inferences of counts, identities and positions are multi-modal and difficult to represent using standard network architectures. We also compare the performance of AIR to other methods such as discriminative and generative methods, and observe that AIR produces significantly better reconstructions and count accuracies than a supervised method on data.
One-shot Generalization
Environment Simulation

Visual Concept Learning
Successful Applications of Generative Models
Progress in Generative Models

**MNIST**

<table>
<thead>
<tr>
<th>Method</th>
<th>Neg log-likelihood (nats)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mixture of Bernoullis</td>
<td>137.6</td>
</tr>
<tr>
<td>Factor Analysis</td>
<td>106</td>
</tr>
<tr>
<td>Wake Sleep</td>
<td>91.3</td>
</tr>
<tr>
<td>NADE</td>
<td>88.3</td>
</tr>
<tr>
<td>RBM</td>
<td>86.3</td>
</tr>
<tr>
<td>FC-VAE</td>
<td>85</td>
</tr>
<tr>
<td>HVI-CVAE</td>
<td>83.5</td>
</tr>
<tr>
<td>Conv-DRAW</td>
<td>80.5</td>
</tr>
<tr>
<td>Pixel RNN</td>
<td>79.2</td>
</tr>
</tbody>
</table>

**Omniglot**

<table>
<thead>
<tr>
<th>Method</th>
<th>Neg log-likelihood (nats)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC-VAE</td>
<td>106</td>
</tr>
<tr>
<td>FC-IWAE</td>
<td>103</td>
</tr>
<tr>
<td>RBM</td>
<td>100</td>
</tr>
<tr>
<td>Conv-DRAW</td>
<td>91</td>
</tr>
</tbody>
</table>
Progress in Generative Models
Machine Learning Framework

1. Models
2. Learning Principles
3. Algorithms
Types of Generative Models

Models

- Fully-observed models: Model observed data directly without introducing any new unobserved local variables.
- Transformation models: Model data as a transformation of an unobserved noise source using a parameterised function.
- Latent variable models: Introduce an unobserved random variable for every observed data point to explain hidden causes.
Types of Generative Models

- **Fully-observed models**
  - Model observed data directly without introducing any new unobserved local variables.
Machines that Imagine and Reason

Types of Generative Models

**Fully-observed models**

Model observed data directly without introducing any new unobserved local variables.

Models

**Transformation models**

Model data as a transformation of an unobserved noise source using a parameterized function.
Types of Generative Models

- **Fully-observed models**: Model observed data directly without introducing any new unobserved local variables.

- **Transformation models**: Model data as a transformation of an unobserved noise source using a parameterized function.

- **Latent variable models**: Introduce an unobserved random variable for every observed data point to explain hidden causes.
Learning Principles

For a given model, there are many competing inference methods.

- Exact methods (conjugacy, enumeration)
- Numerical integration (Quadrature)
- Generalized method of moments
- Maximum likelihood (ML)
- Maximum a posteriori (MAP)
- Laplace approximation
- Integrated nested Laplace approximations (INLA)
- Expectation Maximization (EM)
- Monte Carlo methods (MCMC, SMC, ABC)
- Noise contrastive estimation (NCE)
- Cavity Methods (EP)
- Variational methods
A given model and learning principle can be implemented in many ways.

**Convolutional neural network + penalised maximum likelihood**
- Optimization methods (SGD, Adagrad)
- Regularization (L1, L2, batchnorm, dropout)

**Latent variable model + variational inference**
- VEM algorithm
- Expectation propagation
- Approximate message passing
- Variational auto-encoders

**Restricted Boltzmann Machine + maximum likelihood**
- Contrastive Divergence
- Persistent Contrastive Divergence
- Parallel Tempering
- Natural gradients
Types of Generative Models
Types of Generative Models

Design Dimensions

- **Data**: binary, real-valued, nominal, strings, images.
- **Dependency**: independent, sequential, temporal, spatial.
- **Representation**: continuous or discrete
- **Dimension**: parametric or non-parametric
- **Computational complexity**
- **Modeling capacity**
- **Bias, uncertainty, calibration**
- **Interpretability**
**Fully-Observed Models**

Model observed data directly **without** introducing any new unobserved **local variables**.

\[
x_1 \sim \text{Cat}(x_1 | \pi) \\
x_2 \sim \text{Cat}(x_2 | \pi(x_1)) \\
\ldots \\
x_i \sim \text{Cat}(x_i | \pi(x_{<n}))
\]

\[p(x) = \prod_i p(x_i | f(x_{<i}; \theta))\]

**Markov Models**

- \(x_1 \sim \text{Cat}(x_1 | \pi)\)
- \(x_2 \sim \text{Cat}(x_2 | \pi(x_1))\)
- \(\ldots\)
- \(x_i \sim \text{Cat}(x_i | \pi(x_{<n}))\)

**Model Parameters are global variables.**
**Stochastic activations & unobserved random variables are local variables.**

All conditional probabilities described by deep networks.
Fully-Observed Models

**Properties**
- Can directly encode how observed points are related.
- *Any data* type can be used
- For directed graphical models:
  - **Parameter learning simple**: Log-likelihood is directly computable, no approximation needed.
  - Easy to scale-up to large models, many optimization tools available.
- Order sensitive.
- For undirected models,
  - **Parameter learning difficult**: Need to compute normalizing constants.
- **Generation can be slow**: Iterate through elements sequentially, or using a Markov chain
Model-space Visualization

Fully-observed models

Directed

Discrete

NADE, EoNADE
Fully-visible sigmoid belief networks
Pixel CNN/RNN
RNN Language mod.
Context tree switching

Continuous

Boltzmann Machines
Discrete Markov
Random Fields
Ising, Hopfield
and Potts Models

Gaussian MRFs
Log-linear models

Undirected
Transformation Models

Transformation models

\[ x = f(z) \]

\[ z \sim \mathcal{N}(0, I) \]

Transform an unobserved noise source using a parameterised function.
Transformation Models

Change of variables for invertible functions

\[ p(x) = p(z) \left| \det \frac{\partial f}{\partial z} \right|^{-1} \]

\[ z \sim p(z) \]

\[ x = \mu + Rz \]

Transformation models

Transform an unobserved noise source using a parameterised function.

\[ z \]

\[ f(z) \]

\[ x \]
Transformation Models

Change of variables for invertible functions

\[ z \sim p(z) \]

\[ p(x) = p(z) \left| \det \frac{\partial f}{\partial z} \right|^{-1} \]

\[ x = \mu + Rz \]

\[ z \sim \mathcal{N}(0, I) \]

\[ x = f(z; \theta) \]

The transformation function is parameterised by a linear or deep network (fully-connected, convolutional or recurrent).
Transformation Models

**Properties**

- Easy sampling.
- Easy to compute expectations without knowing final distribution.
- Can exploit with large-scale classifiers and convolutional networks.

- **Difficult to satisfy constraints:** Difficult to maintain invertibility, and challenging optimization.

- **Lack of noise model** (likelihood)
  - Difficult to extend to generic data types.
  - Difficult to account for noise in observed data.
  - Hard to compute marginalized likelihood for model scoring, comparison and selection.

---

Convolutional generative adversarial network

---

Figure 2: Generated bedrooms after one training pass through the dataset. Theoretically, the model could learn to memorize training examples, but this is experimentally unlikely as we train with a small learning rate and minibatch SGD. We are aware of no prior empirical evidence demonstrating memorization with SGD and a small learning rate.

Figure 3: Generated bedrooms after five epochs of training. There appears to be evidence of visual under-fitting via repeated noise textures across multiple samples such as the base boards of some of the beds.

4.3 **MAGENET**

We use Imagenet-1k (Deng et al., 2009) as a source of natural images for unsupervised training. We train on 32 × 32 min-resized center crops. No data augmentation was applied to the images.
Model-space Visualization

Transformation models

Stochastic
Differential Equations
Hamiltonian and
Langevin SDE
Diffusion Models
Non- and volume
preserving flows

One-liners and
inverse sampling
Distrib. warping
Normalising flows
GAN generator nets
Non- and volume
preserving transforms
Model-space Visualization

Transformation models

- Stochastic Differential Equations
- Hamiltonian and Langevin SDE
- Diffusion Models
- Non- and volume preserving flows

- One-liners and inverse sampling
- Distrib. warping
- Normalising flows
- GAN generator nets
- Non- and volume preserving transforms

Diffusions (Continuous time) — Functions (Discrete time)
Latent Variable Models

Latent variable models

Introduce an unobserved local random variables that represents hidden causes.
Latent variable models introduce an unobserved local random variables that represents hidden causes.
Latent variable models

Latent variable models introduce an unobserved local random variables that represents hidden causes.

\[ z_3 \sim \mathcal{N}(0, I) \]
Latent Variable Models

Latent variable models introduce an unobserved local random variables that represents hidden causes.

\[ z_3 \sim \mathcal{N}(0, I) \]
\[ z_2 | z_3 \sim \mathcal{N}(\mu(z_3), \Sigma(z_3)) \]
Latent variable models

*Introduce an unobserved local random variables that represents hidden causes.*

$$
\begin{align*}
    z_3 &\sim \mathcal{N}(0, I) \\
    z_2 | z_3 &\sim \mathcal{N}(\mu(z_3), \Sigma(z_3)) \\
    z_1 | z_2 &\sim \mathcal{N}(\mu(z_2), \Sigma(z_2)) 
\end{align*}
$$
**Latent Variable Models**

Machines that Imagine and Reason

**Deep Latent Gaussian Model**

Latent variable models introduce an unobserved local random variables that represents hidden causes.

\[
\begin{align*}
  z_3 & \sim \mathcal{N}(0, I) \\
  z_2 \mid z_3 & \sim \mathcal{N}(\mu(z_3), \Sigma(z_3)) \\
  z_1 \mid z_2 & \sim \mathcal{N}(\mu(z_2), \Sigma(z_2)) \\
  x \mid z_1 & \sim \mathcal{N}(\mu(z_1), \Sigma(z_1))
\end{align*}
\]
Latent Variable Models

Properties

- Easy sampling.
- Easy way to include hierarchy and depth.
- Easy to encode structure believed to generate the data
- Avoids order dependency assumptions: marginalization of latent variables induces dependencies.
- Latents provide compression and representation of the data.
- Scoring, model comparison and selection possible using the marginalized likelihood.

✗ Inversion process to determine latents corresponding to an input is difficult in general
✗ Difficult to compute marginalized likelihood requiring approximations.
✗ Not easy to specify rich approximations for latent posterior distribution.

Convolutional DRAW

Figure 10. Generated samples from a network trained on 64×64 ImageNet with input scaling $\theta = 0$. Qualitatively asking the model to be less precise seems to lead to visually more appealing samples.
Model-space Visualization

Latent variable models

- Cascaded Indian Buffet process
- Hierarchical Dirichlet process
- Sigmoid Belief Net
- Deep auto-regressive networks (DARN)

- Indian buffet process
- Dirichlet process mixture
- Deep Nonparametric Discrete

- Hidden Markov Model
- Discrete LVM
- Sparse LVMs
- Linear Parametric Discrete

- PCA, factor analysis
- Independent components analysis
- Gaussian LDS
- Latent Gauss Field
- Linear Parametric Continuous

- Gaussian process LVM
- Direct Nonparametric Continuous

- Deep Gaussian processes
- Recurrent Gaussian Process
- GP State space model
- Deep Nonparametric Continuous

- Nonlinear factor analysis
- Nonlinear Gaussian belief network
- Deep Latent Gaussian (VAE, DRAW)
- Deep Parametric Continuous
Inference and Learning
Inference Problems

Common inference problems are:

**Evidence Estimation**

$$p(x) = \int p(x, z) dz$$

**Moment Computation**

$$\mathbb{E}[f(z) | x] = \int f(z) p(z | x) dz$$

**Prediction**

$$p(x_{t+1}) = \int p(x_{t+1} | x_t) p(x_t) dx_t$$

**Hypothesis Testing**

$$B = \log p(x | H_1) - \log p(x | H_2)$$
Bayesian Model Evidence

**Model evidence (or marginal likelihood, partition function):** Integrating out any global and local variables enables model scoring, comparison, selection, moment estimation, normalisation, posterior computation and prediction.

*We take steps to improve the model evidence for given data samples.*

Integral is intractable in general and requires approximation.

\[
p(x) = \int p(x, z) \, dz
\]
**Integral problem**

\[ p(x) = \int p(x|z)p(z)dz \]

**Proposal**

\[ p(x) = \int p(x|z)p(z)\frac{q(z)}{q(z)}dz \]

**Importance Weight**

\[ p(x) = \int p(x|z)\frac{p(z)}{q(z)} q(z)dz \]

\[ w^{(s)} = \frac{p(z)}{q(z)} \quad z^{(s)} \sim q(z) \]

**Monte Carlo**

\[ p(x) = \frac{1}{S} \sum_{s} w^{(s)} p(x|z^{(s)}) \]

**Notation**

Always think of \( q(z|x) \) but often will write \( q(z) \) for simplicity.

**Conditions**

- \( q(z|x) > 0 \), when \( f(z)p(z) \neq 0 \).
- Easy to sample from \( q(z) \).
Integral problem

Proposal

Importance Weight

Jensen's inequality

Variational lower bound

\[ p(x) = \int p(x|z)p(z)dz \]

\[ p(x) = \int p(x|z)p(z) \frac{q(z)}{q(z)} dz \]

\[ p(x) = \int p(x|z) \frac{p(z)}{q(z)} q(z)dz \]

\[ \log p(x) \geq \int q(z) \log \left( p(x|z) \frac{p(z)}{q(z)} \right) dz \]

\[ = \int q(z) \log p(x|z) - \int q(z) \log \frac{q(z)}{p(z)} \]

\[ \mathbb{E}_{q(z)} [\log p(x|z)] - KL[q(z) || p(z)] \]
**Variational Free Energy**

\[ \mathcal{F}(x, q) = \mathbb{E}_{q(z)} \left[ \log p(x|z) \right] - KL[q(z) || p(z)] \]

Interpreting the bound:

- **Approximate posterior distribution** \( q(z|x) \): Best match to true posterior \( p(z|x) \), one of the unknown inferential quantities of interest to us.

- **Reconstruction cost**: The expected log-likelihood measures how well samples from \( q(z|x) \) are able to explain the data \( x \).

- **Penalty**: Ensures that the explanation of the data \( q(z|x) \) doesn’t deviate too far from your beliefs \( p(z) \). A mechanism for realizing Ockham’s razor.
Other Families of Variational Bounds

**Variational Free Energy**

\[ \mathcal{F}(x, q) = \mathbb{E}_{q(z)} \left[ \log p(x|z) \right] - KL[q(z)||p(z)] \]

**Multi-sample Variational Objective**

\[ \mathcal{F}(x, q) = \mathbb{E}_{q(z)} \left[ \log \frac{1}{S} \sum_s \frac{p(z)}{q(z)} p(x|z) \right] \]

**Renyi Variational Objective**

\[ \mathcal{F}(x, q) = \frac{1}{1-\alpha} \mathbb{E}_{q(z)} \left[ \left( \log \frac{1}{S} \sum_s \frac{p(z)}{q(z)} p(x|z) \right)^{1-\alpha} \right] \]

- Other generalized families exist. Optimal solution is the same for all objectives.
Bayesian Two-sample Testing

For some models, we only have access to an unnormalised probability or partial knowledge of the distribution.

Interest is not in estimating the marginal probabilities, only in how they are related.

We compare the estimated distribution to the true distribution using samples.

Basic idea: Transform density ratio estimation into class probability estimation

Learning principle: Two-sample tests

\[
p(\hat{x}) = 1 \quad \Rightarrow \quad p(\hat{x}) = p(\tilde{x})
\]
Bayesian Two-sample Testing

Combining data

Assign labels

Equivalence

Density Ratio \[ \frac{p(\hat{x})}{p(\tilde{x})} \]

Bayes' Rule \[ p(\mathbf{x}|y) = \frac{p(y|x)p(x)}{p(y)} \]

Conditional

Bayes' Subst.

Class probability

Computing a density ratio is equivalent to class probability estimation.
Scoring Function

Bernoulli outcome

Two-sample criterion

\[ p(y = +1|x) = D_\theta(x) \quad p(y = -1|x) = 1 - D_\theta(x) \]

\[ \log p(y|x) = \log D_\theta(\hat{x}) + \log(1 - D_\theta(\tilde{x})) \]

\[ F(x, \theta) = \mathbb{E}_{p(x^{obs})} [\log D_\theta(x^{obs})] + \mathbb{E}_{p(x^{gen})} [\log(1 - D_\theta(x^{gen}))] \]

\[ F(x, \theta, \phi) = \mathbb{E}_{p(x^{obs})} [\log D_\theta(x^{obs})] + \mathbb{E}_{p(z)} [\log(1 - D_\theta(f_\phi(z)))] \]

Alternating optimisation

\[ \min_\phi \max_\theta F(x, \theta, \phi) \]

Instances of testing and inference:
- Two-sample density ratio estimation
- Importance estimation
- Noise-contrastive estimation
- Adversarial learning
Autoregressive
Generative Models
Learning the Distribution of Natural Data

\[ p(x) = \prod_i p(x_i | x_{<}) \]

for 1D sequences such as text or sound
Learning the Distribution of Natural Data

\[ p(x) = \prod_j \prod_i p(x_{i,j} | x_{<}) \]

Autoregressive model for 2D tensors such as images
Learning the Distribution of Natural Data

\[ p(x) = \prod_k \prod_j \prod_i p(x_{i,j,k} | x_{<}) \]

And for 3D tensors such as videos
Learning the Distribution of Natural Data

\[ p(x) = \prod_{k} \prod_{j} \prod_{i} p(x_{i,j,k} | x_{<}) \]

**PixelRNN/ PixelCNN (Images)**
[van den Oord, Kalchbrenner, Kavukcuoglu, 2016]

**Video Pixel Nets (Videos)**
[Kalchbrenner, van den Oord, Simonyan, et al, 2016]

**ByteNet (Language/seq2seq)**
[Kalchbrenner, Espeholt, Simonyan, et al, 2016]

**WaveNet (Audio)**
[van den Oord, Dieleman, Zen, et al, 2016]
Learning the Distribution of Natural Data

\[
p(x) = \prod_{k} \prod_{j} \prod_{i} p(x_{i,j,k} | x_{<})
\]

**Prior work:**

Autoregressive image models: [Larochelle, Murray, 2011] [Theis, Bethge, 2015] [Uria, et al 2016]

Dilated convolutions: [Chen et al, 2015] [Yu, Koltun, 2016] [Holschneider, et al, 1989]

RNN and language/translation modelling: [Hochreiter, Schmidhuber, 1997] [Mikolov et al, 2010] [Kalchbrenner, Blunsom 2013] [Sutskever et al, 2014] [Stollenga et al, 2015] [Kaiser and Bengio, 2016]
Pixel RNN

\[ P(\quad) \]
Pixel RNN

\[ P( ) \]
Pixel RNN

By chain rule and using *pixels* as variables,

\[
P(X) = P(x_1) P(x_2|x_1) P(x_3|x_1, x_2)
\]
Neural Image Model: Pixel RNN

\[ P( ) \]


Idea: use masked convolutions to enforce the autoregressive relationship.

\[ p(x_i | x_1, \ldots, x_i) \]

Figure 1: Left: A visualization of the PixelCNN that maps a neighborhood of pixels to prediction for the next pixel. To generate pixel \( x_i \) the model can only condition on the previously generated pixels \( x_1, \ldots, x_i \).

Middle: an example matrix that is used to mask the 5x5 filters to make sure the model cannot read pixels below (or strictly to the right) of the current pixel to make its predictions.

Right: Top: PixelCNNs have a blind spot in the receptive field that cannot be used to make predictions. Bottom: Two convolutional stacks (blue and purple) allow to capture the whole receptive field.

Combine the strengths of both models by introducing a gated variant of PixelCNN (Gated PixelCNN) that matches the log-likelihood of PixelRNN on both CIFAR and ImageNet, while requiring less than half the training time.

We also introduce a conditional variant of the Gated PixelCNN (Conditional PixelCNN) that allows us to model the complex conditional distributions of natural images given a latent vector embedding.

We show that a single Conditional PixelCNN model can be used to generate images from diverse classes such as dogs, lawn mowers and coral reefs, by simply conditioning on a one-hot encoding of the class. Similarly, one can use embeddings that capture high-level information of an image to generate a large variety of images with similar features. This gives us insight into the invariances encoded in the embeddings — e.g., we can generate different poses of the same person based on a single image. The same framework can also be used to analyse and interpret different layers and activations in deep neural networks.

\[ \text{PixelCNNs (and PixelRNNs) [30]} \text{ model the joint distribution of pixels over an image as the following product of conditional distributions, where } x_i \text{ is a single pixel:} \]

\[ p(x) = \prod_{i=1}^{N^2} p(x_i | x_1, \ldots, x_i) \]

(1)

The ordering of the pixel dependencies is in raster scan order: row by row and pixel by pixel within every row. Every pixel therefore depends on all the pixels above and to the left of it, and not on any of other pixels. The dependency field of a pixel is visualized in Figure 1 (left).

A similar setup has been used by other autoregressive models such as NADE [14] and RIDE [26]. The difference lies in the way the conditional distributions \( p(x_i | x_1, \ldots, x_i) \) are constructed. In PixelCNN every conditional distribution is modelled by a convolutional neural network. To make sure the CNN can only use information about pixels above and to the left of the current pixel, the filters of the convolution are masked as shown in Figure 1 (middle). For each pixel the three colour channels (R, G, B) are modelled successively, with B conditioned on (R, G), and G conditioned on R.

This is achieved by splitting the feature maps at every layer of the network into three and adjusting the centre values of the mask tensors. The 256 possible values for each colour channel are then modelled using a softmax.

PixelCNN typically consists of a stack of masked convolutional layers that takes an \( N \times N \times 3 \) image as input and produces \( N \times N \times 3 \times 256 \) predictions as output. The use of convolutions allows the predictions for all the pixels to be made in parallel during training (all conditional distributions from Equation 1). During sampling the predictions are sequential: every time a pixel is predicted, it is...
Neural Image Model: Pixel RNN

Masked Convolutions

How can convolutions make this raster scan faster?

Use a stack of masked convolutions

Training can be parallelized, though generation is still a sequential operation over pixels

Masked Convolutions
Pixel CNN

Neural Image Model: Pixel RNN

\[ P( \quad ) \]

Pixel CNN

Masked Convolutions

R G B

Context
Pixel CNN

Multiple layers of masked convolutions

\[ p(x_i \mid \mathbf{x}_{<i}) = p(x_{i,R} \mid \mathbf{x}_{<i})p(x_{i,G} \mid x_{i,R}, \mathbf{x}_{<i})p(x_{i,B} \mid x_{i,R}, x_{i,G}, \mathbf{x}_{<i}) \]