Advanced Generation Methods

Hsiao-Ching Chang, Ameya Patil, Anand Bhattad

“What I cannot create, I do not understand.” —Richard Feynman

M. C. Escher, 1948
DL Toolbox:
1) DNN, CNN & RNN
2) Variational autoencoders, ...
3) Adam, SGD, dropout, ...
   etc......

Something Extra:
1) Intuitions and insights into the problems
2) Ways of putting DL tools together
   etc......

Advanced Generation Methods
Image Generation: How Machines do it

The techniques that we learned: **GANs and VAEs**

They attempt to generate image in one-shot projection!!
Image Generation: How We Do It

We pay **attention** on each subpart, we **iterate** in a **feedback** loop.

Can we teach machines to do the same?
Advanced Generation Methods:

• **Pixel-by-pixel generation:**

A simple way to iterate, employ feedback and capture pixel dependencies

• **Iterative attentive generation:**

More advanced techniques involving iterative formation of an abstract schema
Pixel-by-pixel generation:
Outline

• Intuition
• Basic models
  • PixelRNN
  • PixelCNN
• Variants of PixelRNN and PixelCNN
  • Multi-scale version
  • Conditional image generation
  • Other recent improvements
Outline

• Intuition

• Basic models
  • PixelRNN
  • PixelCNN

• Variants of PixelRNN and PixelCNN
  • Multi-scale version
  • Conditional image generation
  • Other recent improvements
Intuition ...(A customary CAT slide!)

How to include statistical dependencies over hundreds of pixels?

Intuition

Pixel recurrent neural networks, ICML 2016
Intuition

\[ p(\mathbf{x}) = p(x_1, x_2, \ldots, x_{n^2}) \]

Bayes Theorem:

\[ p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i|x_1, \ldots, x_{i-1}) \]
Intuition

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

Bayes Theorem:

A sequential model!

Pixel recurrent neural networks, ICML 2016
LSTM
Intuition

• Question: Can we use plain-LSTM to generate images pixels by pixels?

[Image of a grid with a highlighted section]

*Pixel recurrent neural networks*, ICML 2016
Intuition

• Question: Can we use plain-LSTM to generate images pixels by pixels?

• Ensure information is well propagated in two dimensions

Pixel recurrent neural networks, ICML 2016
Intuition

• Question: Can we use plain-LSTM to generate images pixels by pixels?

• Ensure information is well propagated in two dimensions

• spatial LSTM (sLSTM)
Spatial LSTM

Adapted from: Generative image modeling using spatial LSTM. Theis & Bethge, 2015
Spatial LSTM

Adapted from: Generative image modeling using spatial LSTM. Theis & Bethge, 2015
Details about Soft Max

• Treat pixels as discrete variables:
  • To estimate a pixel value, do classification in every channel (256 classes indicating pixel values 0-255)
  • Implemented with a final softmax layer

Figure: Example softmax outputs in the final layer, representing probability distribution over 256 classes.

Figure from: Oord et al.
PixelRNN: A specific Multidimensional LSTM

RowLSTM

image  sLSTM-1  sLSTM-2  sLSTM-12

softmax layer

Pixel recurrent neural networks, ICML 2016
First LSTM Layer

Image layer

Row LSTM

Pixel recurrent neural networks, ICML 2016
PixelRNN: A specific Multidimensional LSTM

Pixel recurrent neural networks, ICML 2016
Diagonal LSTM

Pixel recurrent neural networks, ICML 2016
Receptive Field

Row LSTM  Diagonal LSTM  Diagonal BiLSTM

Pixel recurrent neural networks, ICML 2016
Diagonal LSTM

• To optimize, we skew the feature maps so it can be parallelized

Pixel recurrent neural networks, ICML 2016
PixelCNN

image  Conv-1  Conv-2  Conv-15

softmax layer
PixelCNN

- 2D convolution on previous layer
- Apply masks so a pixel does not see future pixels (in sequential order)

Pixel recurrent neural networks, ICML 2016
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>PixelCNN</th>
<th>PixelRNN – Row LSTM</th>
<th>PixelRNN – Diagonal BiLSTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full dependency field</td>
<td>Full dependency field</td>
<td>Triangular receptive field</td>
<td>Full dependency field</td>
</tr>
<tr>
<td>Fastest</td>
<td>Fastest</td>
<td>Slow</td>
<td>Slowest</td>
</tr>
<tr>
<td>Worst log-likelihood</td>
<td>Worst log-likelihood</td>
<td>-</td>
<td>Best log-likelihood</td>
</tr>
</tbody>
</table>

Figure from: [Oord et al.](https://arxiv.org/abs/1609.03364)
Architecture

- Residual connections
- Channel masks
  - Sequential order: $R \rightarrow G \rightarrow B$
  - Used in input-to-state convolutions
  - Two types of masks:
    - Channels are connected to themselves
      - Used in all other subsequent layers
    - Channels are not connected to themselves
      - Only used in first layer

Figure from: Oord et al.
## Architecture

<table>
<thead>
<tr>
<th>PixelCNN</th>
<th>Row LSTM</th>
<th>Diagonal BiLSTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 x 7 conv mask A</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Multiple residual blocks:** (see fig 5)

<table>
<thead>
<tr>
<th>Conv 3 x 3 mask B</th>
<th>Row LSTM i-s: 3 x 1 mask B</th>
<th>s-s: 3 x 1 no mask</th>
<th>Diagonal BiLSTM i-s: 1 x 1 mask B</th>
<th>s-s: 1 x 2 no mask</th>
</tr>
</thead>
</table>

ReLU followed by 1 x 1 conv, mask B (2 layers)

256-way Softmax for each RGB color (Natural images) or Sigmoid (MNIST)

Figure from: [Oord et al.](#)
Results

Figure: 32x32 ImageNet results from Diagonal BiLSTM model.

Figure from: Oord et al.
Outline

• Intuition

• Basic models
  • PixelCNN
  • PixelRNN

• Variants of PixelRNN and PixelCNN
  • Multi-scale version
  • Conditional image generation
  • Other recent improvements
Multi-scale PixelRNN

• Take subsampled pixels as additional input pixels
• Can capture better global information (visually more coherent)
• Performance is similar to normal one

Figure from: Oord et al.
Multi-scale PixelRNN

Figure: 64x64 ImageNet results from normal Diagonal BiLSTM model (left) and multi-scale model (right).

Figure from: Oord et al.
Conditional Image Generation

- Given a high-level **image description vector** $h$

\[
p(x) = p(x_1, x_2, \ldots, x_{n^2})
\]

\[
p(x|h) = p(x_1, x_2, \ldots, x_{n^2}|h)
\]

*Conditional image generation with pixelcnn decoders*. NIPS 2016
Conditional Image Generation

• $h$ is location-independent
  • For example,
    • One-hot encoding representing a specific class
    • Latent representation embedding
  • Model joint probability conditioned on $h$

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

\[
y = \tanh(W_{k,f} \ast x + V_{k,f}^T h) \odot \sigma(W_{k,g} \ast x + V_{k,g}^T h)\]

*Conditional image generation with pixelcnn decoders*. NIPS 2016
Conditional Image Generation

- $h$ is location-dependent
  - $h$ contains both object and location information
  - Use an additional deconvolutional neural network to estimate $s = m(h)$, where $s$ has same size as images

$$p(x|h) = \prod_{i=1}^{n^2} p(x_i|x_1, \ldots, x_{i-1}, h)$$

$$y = \tanh(W_{k,f} * x + V_{k,f} * s) \odot \sigma(W_{k,g} * x + V_{k,g} * s)$$

1x1 convolution 1x1 convolution

[Conditional image generation with pixelcnn decoders. NIPS 2016](https://arxiv.org/abs/1606.05908)
Results

Figure from: Oord et al.
Other Recent Improvements

• Gated PixelCNN ([Oord et al.](#))
  • Improve PixelCNN by removing blind spots and replacing ReLU units
• PixelCNN++ ([Salimans et al.](#))
  • Improve PixelCNN by optimization techniques
• Video Pixel Networks ([Kalchbrenner et al.](#))
  • Extend the work to 4 dimension
## Comparison with GANs and VAEs

<table>
<thead>
<tr>
<th>Autoregressive models (PixelRNNs, PixelCNNs)</th>
<th>GAN</th>
<th>VAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Simple and stable training process (e.g. softmax loss)</td>
<td>• Sharpest images</td>
<td>• Easy to relate image with low-dimensional latent variables</td>
</tr>
<tr>
<td>• Best log likelihoods so far</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Inefficient during sampling</td>
<td>• Difficult to optimize due to unstable training dynamics</td>
<td>• Tends to have blurry outputs</td>
</tr>
<tr>
<td>• Don't easily provide simple low-dimensional codes for images</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Credit: [https://openai.com/blog/generative-models/](https://openai.com/blog/generative-models/), Oord et al. and Larsen et al.
Iterative Attentive Generation:
What We Saw Previously

Pixel-by-pixel generation:
Inference decisions at the pixel-level

Generation using VAEs and GANs:

Employ one-shot generation
- Limited for highly complex scenes
- Alignments of objects should be considered only after objects are generated

Random latent vector

Complex high dimensional, non-linear projection

Generated image
The Human Way

Attention
Focus on a subpart at a time

Iteration
Use multiple strokes
Eraser

Feedback
Draw, take a look and then see what to draw next

Enabling machines to employ above attributes

Iterative Attentive Generation using deep learning models
DRAW: A Recurrent Neural Network For Image Generation

Karol Gregor
Ivo Danihelka
Alex Graves
Danilo Jimenez Rezende
Daan Wierstra
Google DeepMind

May 2015

KAROLG@GOOGLE.COM
DANIELKA@GOOGLE.COM
GRAVESA@GOOGLE.COM
DANILOR@GOOGLE.COM
WIERSTRA@GOOGLE.COM
How can a machine employ “Attention”?

What is attention?

Ability to focus on a part of an image, either to understand it or to modify it.

Original image
500 × 500

Glimpse
60 × 60

For machine, it is a process of “Glimpse” extraction.
Employing Attention:

**Attentive read**: Reading a particular glimpse from an image

Input Image: $A \times B$

[Diagram of attentive read process]

Extracted glimpse: $N \times N$

**Attentive write**: Writing the modified glimpse to the image

Modified glimpse: $N \times N$

[Diagram of attentive write process]

Updated image: $A \times B$
Read Attention

Filter definitions:

\[ F_X[i, a] = \frac{1}{Z_X} \exp \left( -\frac{(a - \mu_X^i)^2}{2\sigma_X^2} \right) \]

\[ F_Y[j, b] = \frac{1}{Z_Y} \exp \left( -\frac{(b - \mu_Y^j)^2}{2\sigma_Y^2} \right) \]
Choosing filter parameters

$\delta$

$\mu^i_x = g_x + (i - N/2 - 0.5) \delta$

$\mu^j_y = g_y + (j - N/2 - 0.5) \delta$

$g_x, g_y, \gamma, \delta$ are learned. Details about it later....
Write Attention:

Filter $X: N \times B$

Filter $Y^T: A \times N$

Glimpse: $N \times N$

Updated image $A \times B$

Again, we use Gaussian filters in the exact same setting. Write filters are distinct from read filters.
Machines trying to generate images in a manner similar to us
How can machines employ “Iterations”? 

Machines actually iterate all the time...!

But for image generation, you need something to iterate upon....

A canvas !!!

It is a matrix $c$, typically of the same size that of the original image

Write attention operation updates parts in $c$

Final generated image $= f(c)$
How can machines employ “Feedback”?

Remember RNNs ..!!

RNNs
(Figure courtesy: Arun Mallya)

So, why don’t combine these two..!

And we want to generate images..!

Variational Autoencoders
Putting it all together: Ta Da...!!

Conventional Variational autoencoder (operating on \textit{glimpse})

Attentive read:

Input image \\
RNN-based feedback and loop:

Attentive write and canvas update:

\[ C_{t-1} \xrightarrow{\text{write}} C_t \]

- \( h_{t-1}^{\text{dec}} \)
- \( h_{t-1}^{\text{enc}} \)
- \( Q(z_t | x, z_{1:t-1}) \)
- \( z_t \)
- \( \hat{h}_{t-1} \)
- \( h_{t-1} \)
- \( x \)
Training Procedure

Goal in training:
Learn to reconstruct the input image $x$ in $T$ iteration...

So, for every iteration $t$:
Training Procedure

**Loss function:**

\[- \log D(x | c_T) + \sum_{t=1}^{T} KL(Q(Z_t | h_t^{enc}) || P(Z_t))\]

- **Reconstruction loss**
- **Regularization** (for each iteration)

Recall: VAE lecture

Typically, \( P(Z_t) \sim N(0, I) \) and \( Q(Z_t | h_t^{enc}) \sim N(\mu_t(h_t^{enc}), \sigma_t^2(h_t^{enc})) \)
Training Procedure

Given input image, compute the feedforward path for $T$ iterations

Compute the final loss function

Compute the gradients and propagate them back
(recall: reparametrization trick in VAEs)
Image Generation During Test:

No encoder is used while generating images, just like VAEs
Let’s See How it Performs:

DRAW: A Recurrent Neural Network For Image Generation by Google DeepMind
Final Results: Generated images

<table>
<thead>
<tr>
<th>Task</th>
<th>glimpses</th>
<th>LSTM #h</th>
<th>#z</th>
<th>Read Size</th>
<th>Write Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST Model</td>
<td>64</td>
<td>256</td>
<td>100</td>
<td>2 × 2</td>
<td>5 × 5</td>
</tr>
<tr>
<td>SVHN Model</td>
<td>32</td>
<td>800</td>
<td>100</td>
<td>12 × 12</td>
<td>12 × 12</td>
</tr>
<tr>
<td>CIFAR Model</td>
<td>64</td>
<td>400</td>
<td>200</td>
<td>5 × 5</td>
<td>5 × 5</td>
</tr>
</tbody>
</table>

**DRAW:** a recurrent neural network for image generation. ICML 2015

MNIST

SVHN
The images still seem somewhat blur
Final Results:

The red square indicates “glimpse” used for attentive write in canvas.
So, now we have a machine that can employ “attention”, “iteration” and “feedback”

(Lossy) Image Compression

Improve it and use it for

One Shot Generalization
Image Generation to Compression:

How humans generate images ..!

1. Some bipedal entity
2. A human-like being
3. A young boy sitting, has some hair, wearing shirt, shoes.....
4. Final exact image:

Conceptual hierarchy

A way to “conceptually” compress images
“Conceptual” Lossy Compression

Given 4, can a machine infer 1, 2 and 3?

1 can also mean: 2 can also mean: 3 can also mean: Actual Image:

Ultimate (lossy) compression: Store only absolutely essential information (concept). Let the deep generative model generate the details...

Image credits: https://s-media-cache-ak0.pinimg.com/236x/86/f2/12/86f212a460b81fb5d12fe3d7460aa0e0.jpg, https://goo.gl/images/afnkkg
Towards Conceptual Compression

Karol Gregor
Frederic Besse
Danilo Jimenez Rezende
Ivo Danihelka
Daan Wierstra
Google DeepMind, London, United Kingdom

April 2015

Compression
Deep Generative Model: Recall

**Inference Model**

- **Generative Model**
  - Sample $z_t \sim Q(. | h_t^{enc})$
  - Encoder RNN
    - $h_t^{enc}$
    - $h_{t-1}^{enc}$
    - $h_{t-1}^{dec}$
  - Glimpse
    - $r_t$
  - Attentive Read
    - $c_{t-1}$
    - Input image $x$

**Generative Model**

- **Inference Model**
  - Latest canvas $c_{t-1}$
  - Attentive Write
    - $h_t^{dec}$
    - $h_{t-1}^{dec}$
  - Decoder RNN
    - $h_t^{dec}$
  - $z_t$
  - Updated canvas

**Inference Model**

**Generative Model**
Compression: Basic Idea

• Train a deep generative model for $T$ iterations
• Given a new image $x$, run the model for first $t_{str} < T$ iterations

Store the means of inferred posterior latent distributions: $z_1, \ldots, z_{t_{str}}$

These $z_1, \ldots, z_{t_{str}}$ represent the compressed image
Reconstruction from Compressed Image:

- Start with $z_1, \ldots, z_{t_{str}}$
- Run first $t_{str} < T$ iterations of the generative model “deterministically”
- Run $t_{str} + 1$ to $T$ iterations while sampling those latent variables from learned prior $p(z_t|h_{t-1}^d)$

Note: The prior here depends upon $h_{t-1}^d$
Some More Details:

• Training procedure and training loss are same as we saw in the case of DRAW work

\[ L = \beta L^x + \sum_{t=1}^{T} L^z_t \]

Parameter $\beta$ controls emphasis on “pixel-level reconstruction”

• The latent variables $z_1$ to $z_{t_{str}}$ can be further compressed by discretization and arithmetic coding etc.
Reconstruction from “conceptually compressed”

Towards conceptual compression. NIPS 2016

<table>
<thead>
<tr>
<th>t_{str}</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>22</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
</tbody>
</table>

Reconstructed with prior variance (WPV)

\[ Z_t \sim p(z|h_{t-1}^d) \]
Results: Lossy Compression

<table>
<thead>
<tr>
<th></th>
<th>JPEG</th>
<th>JPEG2000</th>
<th>conv-DRAW, WPV</th>
<th>conv-DRAW, WoPV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits per image</td>
<td>153</td>
<td>307</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Towards conceptual compression. NIPS 2016
Results: Lossy Compression

When latent variables are sampled from prior distribution, the reconstructed images are sharper but have artifacts.
Results: Lossy Compression

JPEG
JPEG2000
conv-DRAW, WPV
conv-DRAW, WoPV

bits per image: 612

Towards conceptual compression. NIPS 2016
Results: Image Generation

$\beta$: weighing parameter for reconstruction loss

$\beta = 0.4$

$\beta = 0.5$

$\beta = 0.6$

$\beta = 0.8$

$\beta = 1$

Lower the $\beta$, lesser the emphasis on pixel-level details, and more on learning the latent representative structure.
One-shot Generalization

Train your machine to obtain the final parameters...!!

Now, show a new example, that is **not a part of training set**

Then, can the machines generate following examples... ?

Humans have this ability of one-shot generalization

Image credits: Lake et al., science 2015
One-Shot Generalization in Deep Generative Models

Danilo J. Rezende*
Shakir Mohamed*
Ivo Danihelka
Karol Gregor
Daan Wierstra
Google DeepMind, London

Given one new example

Examples generated by the model

May 2016

DANILOR@GOOGLE.COM
SHAKIR@GOOGLE.COM
DANIELKA@GOOGLE.COM
KAROLG@GOOGLE.COM
WIERSTRA@GOOGLE.COM
Deep Generative Model: Recall

Inference Model

Generative Model
Generative Model for One-shot Generalization

Input image: $x$

Latest canvas: $c_{t-1}$

Encoder RNN:
- $h_{t-1}^{enc}$
- $r_t$
- $h_t^{enc}$

Sample $z_t \sim Q(. | h_t^{enc})$

Decoder RNN:
- $h_{t-1}^{dec}$
- $c_{t-1}$
- $h_t^{dec}$

Attentive Write:
- $h_t^{dec}$

Output image: $x_1$

New image:
One-Shot Generalization: Results

Given new example:

Examples generated by the network:

In this work, they used spatial transformer networks for attentive read and write.

More details at: https://arxiv.org/abs/1506.02025
Summary:

• Deep learning tools can be put together in many different innovative ways to obtain interesting results for different applications
  • PixelRNN, Deep generative model for conceptual compression

• We essentially studied extensions of “one-shot” image generation techniques to include some feedback in them using RNNs
  • Such extension is possible for GAN-based generation as well
Topics We Didn’t Cover

• Image Style Transfer: Take style of one image and draw the contents of other image in that style (Heard of Prisma App?)
  
  • J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. ECCV 2016

• Combining GANs and RNNs to form Generative: DRAW paper combined VAEs and RNNs. Similar combination of GANs and RNNs can be achieved (See backup slides)
• B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. *Human-level concept learning through probabilistic program induction*. Science 2015
Backup Slides
What We Achieved

Deep Generative model that we studied

\[ \text{RNNs} \]

(Figure courtesy: Arun Mallya)

But, GANs generate sharper images
Are You Wondering??

\[ ? = \text{RNNs} + \text{GANs} \]

RNNs (Figure courtesy: Arun Mallya)

GANs (Figure courtesy: TUM wiki system)
Generating images with recurrent adversarial networks

Daniel Jiwoong Im¹
Montreal Institute for Learning Algorithms
University of Montreal
imdaniel@iro.umontreal.ca

Hui Jiang
Department of Engineering and Computer Science
York University
hj@cse.yorku.ca

Chris Dongjoo Kim
Department of Engineering and Computer Science
York University
kimdon20@gmail.com

Roland Memisevic
Montreal Institute for Learning Algorithms
University of Montreal
memisevr@iro.umontreal.ca

December 2016
DRAW Network

Generative Recurrent Adverserial Network (GRAN)
They Generated Much Sharper Images