Generative Adversial Networks

David Coufal
Institute of Computer Science
The Czech Academy of Sciences
david.coufal@cs.cas.cz

Vision for Robotics - FEL CTU
December 9, 2019
Neural networks

- A neural network is a complex composite function built from individual layers of neurons, neurons represent simple computation units.

- Neurons are parametrized, so the whole network is a highly parametrized function.

- Adjustment of parameters is called network learning back propagation of an error represented by some loss function.

- Shallow networks - only one hidden layer of neurons.

- Deep networks - multiple layers (up to 200 layers, millions of parameters).
Standard neural networks

- standard neuron $h : \mathbb{R}^d \rightarrow \mathbb{R}$ has form

$$h(x) = act(wx + b)$$

- $act(z) = \max(0, z)$ (relu), $act(z) = \frac{1}{1 + e^{-\beta z}}$ (sigmoid)

- $w, b \in \mathbb{R}^d$ - parameters
Convolutional neural networks

- convolution filters moving over the input

- down-sampling and up-sampling operations, pooling

Well recognized DL tasks

- **classification**
  ImageNet Large Scale Visual Recognition Challenge AlexNet CNN network won the contest using convolutional implementation (2012)

- **recurrent neural networks (RNNs)**
  LSTM, GRU - units, NLP tasks, Google Translator

- **reinforcement learning** DeepMind (UK, Google 2014)
  AlphaGo vs. Lee Sedol (4:1, 2016), AlphaGoZero vs. AlphaGo (100:0, 2017) AlphaZero vs. Stockfish (28:72:0, 2018), Dota 2 tournaments ...

- **generative programming**
Elementary concepts

- random variable $X \sim P_X$, $(\Omega, \mathcal{A}, P_X)$
  - $\Omega$ - space of elementary events $X \in \Omega$
  - $\mathcal{A}$ - sigma algebra of measurable events
  - $P_X$ - distribution of $X$

- distribution of $X$
  - set function on $\mathcal{A}$, $P_X : \mathcal{A} \rightarrow [0, 1]$
  - obeys Kolmogorov’s laws of probability
  - typically $\Omega \in \mathbb{R}^d$ and $\mathcal{A} = \mathcal{B}(\mathbb{R}^d)$

- data $D = \{x_i \in \mathbb{R}^d\}_{i=1}^n$ comes from distribution $P_D$
  i.e., we assume that there exists a random variable $D$
  such that $D \sim P_D$ (sometimes we use $P_{\text{data}}$ instead of $P_D$)

- How to specify $P_D$ on the basis of $D$?
Elementary concepts

- if Ω is countable, $P_D$ can be given by enumeration, i.e., $P_D(\omega_i) = p_i$, for $i = 1, \ldots, n$ (finite) or $i \in \mathbb{N}$ (countable)

- if $\Omega = \mathbb{R}^d$, specification of cdf is possible, but inconvenient in higher dimensions, so the most common approach is to specify a density $p_D : \mathbb{R}^d \rightarrow [0, \infty)$ of $P_D$ and one has

$$P_D(A) = \int_A p_D(x) \, dx \quad \text{for } A \in \mathcal{B}(\mathbb{R}^d)$$

- cannot handle distributions which do not have densities, complex formulas in high dimensions for dependent data

- How to get the density from empirical data?
Elementary concepts

• if \( p_D \in \{ p_\theta, \theta \in \Theta \} \) (a parametric set of densities) task reduces to estimate \( \theta^* \) from data \( D \) and \( p_D = p_{\theta^*} \) maximum likelihood estimation

• in a non-parametric context, kernel density estimation is the standard choice

\[
p^*_D(x) = \frac{1}{nh^d} \sum_{k=1}^{n} K \left( \frac{x - x_i}{h} \right)
\]

• \( K : \mathbb{R}^d \to \mathbb{R} \), a kernel (bump) function, \( h > 0 \) is the bandwidth practically applicable for \( d \) up to 5

• How to sample from a given distribution/density?
Distance of probability distributions

- space of probability distributions on $\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d)$: 
  $\mathcal{P} = \{P : \text{probability distribution on } (\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))\}$
  $\mathcal{P}$ is metrizable, e.g., using Lévy-Prokhorov metric $\pi : \mathcal{P}^2 \to [0, \infty)$, complicated formulas

- another "metric" is the Kullback-Leibler divergence
  let $P, Q \in \mathcal{P}$, $P \ll Q$ (if $Q(x) = 0$, then $P(x) = 0$)
  \[
  KL(P||Q) = \int \frac{dP}{dQ} \ dP \\
  = \int \log \left( \frac{p(x)}{q(x)} \right) p(x) \ dx
  \]

- properties:
  $KL(P||Q) \neq KL(Q||P)$, $KL(P||Q) \geq 0$, $KL(P||P) = 0$,

- tight relation to theory of information (relative entropy), theory of large deviations
Kullback-Leibler divergence

- (Wikipedia entry ...) In applications, $P$ typically represents the "true" distribution of data, observations, or a precisely calculated theoretical distribution, while $Q$ typically represents a theory, model, description, or approximation of $P$. In order to find a distribution $Q$ that is closest to $P$, we can minimize KL divergence and compute (reverse) information projection

- Kullback-Leibler divergence is a special case of a broader class of statistical divergences called $f$-divergences

- Jensen-Shannon divergence - symmetrized KL divergence

$$JS(P||Q) = \frac{1}{2}KL(P||M) + \frac{1}{2}KL(Q||M)$$

where $M = \frac{1}{2}(P + Q)$
Reverse information projection (M-projection)

- let $P \in \mathcal{P}$ and $Q \subset \mathcal{P}$ (subset of prob. distributions)

$$Q_{KL}^* = \arg \min_{Q \in Q} KL(P \parallel Q)$$

or for $JSD$

$$Q_{JSD}^* = \arg \min_{Q \in Q} JSD(P \parallel Q)$$

$Q^*$ is the closest distribution from subset of $Q$ to $P$

- easy to state, generally hard to solve (i.e., to find $Q^*$)
Specification of $\mathcal{Q} \subset \mathcal{P}$

- via parametrized densities $\mathcal{Q} = \{p_\theta, \theta \in \Theta\}$

- via parametrized transformations
e.g., let $X \sim \mathcal{N}(0,1)$ then $X^2 \sim \chi^2(1)$
  $X$ has some simple distribution which is easy to sample from and is transformed to a complex one using a deterministic function $G$
  (above $G(z) = z^2$)

- $\mathcal{Q}$ is given by set of parametrized functions $G_\theta$, $\theta \in \Theta$
  (neural networks parametrized via their weights)

- easy sampling from $G_\theta(X)$, sample $x \sim X$ (easy)
  and then pass $x$ through $G_\theta(X)$, i.e., compute $G_\theta(x)$

- How to solve the information projection problem?
Maximum likelihood estimation

- **task**
  
  given set of data \( \{x_i \sim P_D\}_{i=1}^{n} \) describe distribution \( P_D \)

- **MLE estimate** \( P_D \in P_\theta = \{P_\theta, \theta \in \Theta\} \)
  
  assume that \( P_\theta \) has density, i.e., \( dP_\theta = p_\theta(x) \, dx \)
  
  assume that \( x_i \) i.i.d.
  
  search for optimal \( \theta_{\text{mle}} \in \Theta \) and set \( P_D = P_{\theta_{\text{mle}}} \)

  \[
  \theta_{\text{mle}} = \arg\max_\theta \mathbb{E}_{x \sim P_D} \log p_\theta(x)
  \]

  estimate \( \theta^*_{\text{mle}} = \arg\max_\theta \frac{1}{n} \sum_{i=1}^{n} \log p_\theta(x_i) \)

- **optimization in terms of KL-divergence**

  \[
  \theta_{\text{mle}} = \arg\min_\theta KL(P_D(x) \parallel P_\theta(x))
  \]

  \[
  = \arg\min_\theta \int p_D(x) \frac{p_D(x)}{p_\theta(x)} \, dx
  \]
MLE in terms of KL-divergence

- best approximation of $P_D$ using $P_\theta$
  - $\hat{P}_D$ proxy for $P_D$, $\hat{P}_D(dx) = \frac{1}{n} \delta_{x_i}(dx)$ (Dirac m.)
  - $P_\theta$ - model distribution with density $p_{\text{model}}(x|\theta)$

- maximization MLE = minimization of $KL(P_D||P_\theta)$

$$KL(P_D||P_\theta) = \int \log \frac{dP_D}{dP_\theta} dP_D = \int \log \frac{P_D(x)}{p_\theta(x)} dP_D$$

$$= \int \log p_D(x) dP_D - \int \log p_\theta(x) dP_D$$

$$\approx -H[P_D] - \int p_\theta(x) d\hat{P}_D \quad (P_D \approx \hat{P}_D)$$

$$\propto - \int \log p_\theta(x) d\hat{P}_D \quad \text{(integration over Dirac)}$$

$$\propto - \frac{1}{n} \sum_{i=1}^{n} \log p_\theta(x_i)$$

$$= \text{MLE}$$
Generative modeling

- purpose
  given data from an unknown distribution \( x \sim p(x) \)
  model \( p(x) \) using a differentiable mapping \( G \) so that

  \[
p(x) \sim G_{\theta_g}(p(z)) = G(p(z); \theta_g))
  \]

  where \( p(z) \) is a selected, simple prior, e.g. mv Gaussian

- maximum likelihood estimation direct setting of density under i.i.d. assumption, \( \text{KL divergence minimization} \)
Generative modeling

- solution to the information projection problem
  - KL-divergence minimalization
  - via playing discriminator, generator adversarial game

source: https://towardsdatascience.com/generative-adversarial-networks-learning-to-create-8b15709587c9
Partial criterions

• an ideal discriminator
  \[ D : \mathbb{R}^d \rightarrow (0, 1), \text{ i.e., } \log D : \mathbb{R} \rightarrow (-\infty, 0) \]
  we would like \( D_{\theta_d}(x^{\text{real}}) \rightarrow 1, \ D_{\theta_d}(x^{\text{fake}}) \rightarrow 0 \)
  i.e., maximize w.r.t. \( \theta_d \)
  \[ \log(D_{\theta_d}(x^{\text{real}})) + \log((1 - D_{\theta_d}(x^{\text{fake}}))) \]

• an ideal generator
  generator wants to fool discriminator,
  i.e., it generates \( x^{\text{fake}} \) so that \( D_{\theta_d}(x^{\text{fake}}) \rightarrow 1 \)
  tune weights of the generator to minimize
  \[ \log((1 - D_{\theta_d}(x^{\text{fake}}))) = \log((1 - D_{\theta_d}(D(\theta_g(z))))) \]
  w.r.t \( \theta_g \) for \( \theta_d \) fixed
**Compound criterion**

- **compound criterion**

\[
V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D_{\theta_d}(x)] + \mathbb{E}_{x \sim p_z(x)}[\log(1 - D_{\theta_d}(G_{\theta_g}(z)))]
\]

- **minimax optimization** - set \( \theta_d, \theta_g \) using

\[
\min_{\theta_g} \max_{\theta_d} V(D_{\theta_d}, G_{\theta_g})
\]

- **alternate optimization**

  - for fixed generator \( G_{\theta_g} \) maximize \( V(D_{\theta_d}, \cdot) \)
  
  - for fixed discriminator \( D_{\theta_d} \) minimize \( V(\cdot, G_{\theta_g}) \)
Theoretical analysis

- **Proposition 1.** For any $G$ fixed, the optimal discriminator $D^*_G$ computes the function

$$D^*_G = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)}$$

- **Proposition 2.** Let $C(G) = V(D^*_G, G)$, then global minimum of $\min_G C(G)$ is achieved if and only if $p_g = p_{\text{data}}$. At that point $C(G)$ achieves value $-\log 4$

- **Proposition 3.** Optimizing $\min_G \max_D V(D, G)$ corresponds to minimizing $JS(p_{\text{data}} \parallel p_g)$, which is minimal ($=0$) if and only if $p_{\text{data}} = p_g$

source: https://arxiv.org/abs/1406.2661
A GAN concept

source: https://medium.com/sigmoid/a-brief-introduction-to-gans
Learning algorithm

**Algorithm 1** Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, $k$, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

```latex
\begin{align*}
\text{for} \ number \ of \ training \ iterations \ do \\
\quad \text{for} \ k \ steps \ do \\
\qquad \bullet \ \text{Sample minibatch of } m \ \text{noise samples } \{ z^{(1)}, \ldots, z^{(m)} \} \ \text{from noise prior } p_g(z). \\
\qquad \bullet \ \text{Sample minibatch of } m \ \text{examples } \{ x^{(1)}, \ldots, x^{(m)} \} \ \text{from data generating distribution } p_{data}(x). \\
\qquad \bullet \ \text{Update the discriminator by ascending its stochastic gradient:} \\
\qquad \quad \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D \left( x^{(i)} \right) + \log \left( 1 - D \left( G \left( z^{(i)} \right) \right) \right) \right]. \\
\text{end for} \\
\text{end for} \\
\text{end for}\end{align*}
```

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

MNIST dataset

• 60000/10000 - 28x28 greyscale images of handwritten digits

http://yann.lecun.com/exdb/mnist/
MNIST dataset

- 60000/10000 - 28x28 greyscale images of handwritten digits
- GAN architecture: $D,G$ - perceptron networks
MNIST dataset

- 60000/10000 - 28x28 greyscale images of handwritten digits
  GAN architecture: D,G - convolution networks
cGAN - 2014


- unconditional vs. conditional GAN, \( y \) — *condition*

\[
\mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{x \sim p_{z}(x)}[\log(1 - D(G(z)))] \\
\mathbb{E}_{x \sim p_{data}(x)}[\log D(x|y)] + \mathbb{E}_{x \sim p_{z}(x)}[\log(1 - D(G(z|y)))]
\]

- conditioning by extending latent variable of generator
MNIST dataset
DCGAN - 2015


- architecture - uses convolutional layers
**LSUN dataset**

- 10 categories, (church_outdoor, bedroom, bridge ... )

https://www.yf.io/p/lsun
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.
StackGAN - 2016

• **StackGAN**: *Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks* https://arxiv.org/abs/1612.03242

• **Caltech-UCSD Birds 200 Dataset**
  http://www.vision.caltech.edu/visipedia/CUB-200-2011.html

• **102 Category Flower Dataset**
  https://www.robots.ox.ac.uk/ vgg/data/flowers/102/
- a bird has a bright golden crown and throat, it's breast is yellow, and back is black
- upper body yellow and lower black with black color around beak
- this bird has a bright yellow crown, a long straight bill, and white wingbars
- this is a black bird with a yellow head and breast ...
Figure 2. The architecture of the proposed StackGAN. The Stage-I generator draws a low-resolution image by sketching rough shape and basic colors of the object from the given text and painting the background from a random noise vector. Conditioned on Stage-I results, the Stage-II generator corrects defects and adds compelling details into Stage-I results, yielding a more realistic high-resolution image.
StackGAN - 2016

Figure 3. Example results by our StackGAN conditioned on text descriptions from CUB test set.

Figure 4. Example results by our StackGAN conditioned on text descriptions from Oxford-102 test set and COCO validation set.
StackGAN - 2016

<table>
<thead>
<tr>
<th>Text description</th>
<th>Stage-I images</th>
<th>Stage-II images</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bird is blue with white and has a very short beak</td>
<td><img src="image1.png" alt="Image 1" /></td>
<td><img src="image2.png" alt="Image 2" /></td>
</tr>
<tr>
<td>This bird has wings that are brown and has a yellow belly</td>
<td><img src="image3.png" alt="Image 3" /></td>
<td><img src="image4.png" alt="Image 4" /></td>
</tr>
<tr>
<td>A white bird with a black crown and yellow beak</td>
<td><img src="image5.png" alt="Image 5" /></td>
<td><img src="image6.png" alt="Image 6" /></td>
</tr>
<tr>
<td>This bird is white, black, and brown in color, with a brown beak</td>
<td><img src="image7.png" alt="Image 7" /></td>
<td><img src="image8.png" alt="Image 8" /></td>
</tr>
<tr>
<td>The bird has small beak, with reddish brown crown and gray belly</td>
<td><img src="image9.png" alt="Image 9" /></td>
<td><img src="image10.png" alt="Image 10" /></td>
</tr>
<tr>
<td>This is a small, black bird with a white breast and white on the wingbars.</td>
<td><img src="image11.png" alt="Image 11" /></td>
<td><img src="image12.png" alt="Image 12" /></td>
</tr>
<tr>
<td>This bird is white black and yellow in color, with a short black beak</td>
<td><img src="image13.png" alt="Image 13" /></td>
<td><img src="image14.png" alt="Image 14" /></td>
</tr>
</tbody>
</table>

Figure 5. Samples generated by our StackGAN from unseen texts in CUB test set. Each column lists the text description, images generated from the text by Stage-I and Stage-II of StackGAN.

- https://github.com/hanzhanggit/StackGAN
BEGAN - 2017

• **BEGAN: Boundary Equilibrium Generative Adversarial Networks**
  
  https://arxiv.org/abs/1703.10717

• **energy based GAN**, discriminator assigns low energy values to real data and high otherwise, generator produces samples assigned with low energy by discriminator - generalized view of loss functions training minimization of loss

\[
V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[D_{\theta_d}(x)] + \mathbb{E}_{x \sim p_z(x)}[(m - D_{\theta_d}(G_{\theta_g}(z)))_+] \\
\]

where \( m \) is a positive margin and \( 0 \leq D_{\theta_d} \leq m \)
BEGAN - 2017

- architecture - uses convolutional layers

Figure 1: Network architecture for the generator and discriminator.
BEGAN - 2017

BEGAN - 2017

- generated fake images

Figure 3: Random 64x64 samples at varying $\gamma \in \{0.3, 0.5, 0.7\}$
PGGAN - 2018


- architecture - uses convolutional layers

Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of $4\times4$ pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout the process. Here $N \times N$ refers to convolutional layers operating on $N \times N$ spatial resolution. This allows stable synthesis in high resolutions and also speeds up training considerably. On the right we show six example images generated using progressive growing at $1024 \times 1024$. 
PGGAN - 2018

• architecture - uses convolutional layers

Figure 5: 1024 × 1024 images generated using the CELEBA-HQ dataset. See Appendix F for a larger set of results, and the accompanying video for latent space interpolations.
PGGAN - 2018

- architecture - uses convolutional layers

Figure 5: 1024 × 1024 images generated using the CELEBA-HQ dataset. See Appendix F for a larger set of results, and the accompanying video for latent space interpolations.
ImageNet

- over 14 mil. of images from 20 thousand categories
  based on the WordNet database (a dictionary)
BigGAN - 2019

- **Large Scale GAN Training for High Fidelity Natural Image Synthesis**
  https://arxiv.org/abs/1809.11096

- we show that GANs benefit dramatically from scaling, and train models with two to four times as many parameters and eight times the batch size compared to prior art

- training on 128 to 512 cores of a Google TPUv3 Pod

### Table 1: Fréchet Inception Distance (FID, lower is better) and Inception Score (IS, higher is better) for ablation of our proposed modifications. 
**Batch** is batch size, **Param** is total number of parameters, **Ch.** is the channel multiplier representing the number of units in each layer, **Shared** is using shared embeddings, **Skip-z** is using skip connections from the latent to multiple layers, **Ortho.** is Orthogonal Regularization, and **Itr** indicates if the setting is stable to $10^6$ iterations, or it collapses at the given iteration. Other than rows 1-4, results are computed across 8 random initializations.
BigGAN - 2019

- architecture - uses convolutional layers

Figure 1: Class-conditional samples generated by our model.
Open questions

• What sorts of distributions can GANs model?

• How can we scale GANs beyond image synthesis? (text, audio, computer-aided drug design - https://insilico.com)

• What can we say about the global convergence of the training dynamics?

• How does GAN training scale with batch size?

• What is the relationship between GANs and adversarial examples?

source: https://distill.pub/2019/gan-open-problems