High Level Computer Vision

Adversarial Networks & Applications

Bernt Schiele - schiele@mpi-inf.mpg.de
Mario Fritz - mfritz@mpi-inf.mpg.de

https://www.mpi-inf.mpg.de/hlcv
 Discriminative deep learning

- Recipe for success
Generative modeling

- Have training examples \( x \sim p_{data}(x) \)
- Want a model that can draw samples: \( x \sim p_{model}(x) \)
- Where \( p_{model} \approx p_{data} \)
Generative Adversarial Networks (GANs)

Ian Goodfellow, OpenAI Research Scientist
NIPS 2016 tutorial
Barcelona, 2016-12-4
Generative Modeling

- Density estimation

- Sample generation

Training examples  Model samples

(Goodfellow 2016)
Why study generative models?

- Excellent test of our ability to use high-dimensional, complicated probability distributions
- Simulate possible futures for planning or simulated RL
- Missing data
  - Semi-supervised learning
- Multi-modal outputs
- Realistic generation tasks

(Goodfellow 2016)
Realistic image generation

Progressive Growing of GANs for Improved Quality, Stability, and Variation
Karras et. al, ICLR 2018

High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs
Wang et. al, CVPR 2018
Single Image Super-Resolution

original
bicubic
(21.59dB/0.6423)
SRResNet
(23.44dB/0.7777)
SRGAN
(20.34dB/0.6562)

(Ledig et al 2016)
Image to Image Translation

(Isola et al 2016)

(Goodfellow 2016)
Image Manipulation

Stargan, Choi et.al, CVPR 2018
Roadmap

- Why study generative modeling?
- How do generative models work? How do GANs compare to others?
- How do GANs work?
- Tips and tricks
- Research frontiers
- Combining GANs with other methods

(Goodfellow 2016)
Maximum Likelihood Method

- Learning = Estimation of parameter $\theta$ (given data $X$)

- Likelihood of $\theta$
  - defined as the probability of the data $X$ being generated from the model distribution with parameter $\theta$
  - Likelihood $L(\theta)$: $L(\theta) = p(X|\theta)$
Maximum Likelihood Method

• Calculation of Likelihood
  ‣ a single datapoint: \( p(x_n | \theta) \)
  ‣ assume: all N data points are independent
    - data points are i.i.d = independent identically distributed

\[
L(\theta) = p(X | \theta) = \prod_{n=1}^{N} p(x_n | \theta)
\]

• often used is log-likelihood:
  ‣ often easier to calculate and manipulate

\[
E = -\ln L(\theta) = - \sum_{n=1}^{N} \ln p(x_n | \theta)
\]

• parameter estimation = learning
  ‣ maximize likelihood or log-likelihood or
  ‣ minimize negative log-likelihood
Maximum Likelihood

$$\theta^* = \arg \max_\theta \mathbb{E}_{x \sim p_{data}} \log p_{model}(x \mid \theta)$$
Taxonomy of Generative Models

Maximum Likelihood

- Explicit density
  - Fully visible belief nets
  - NADE
  - MADE
  - PixelRNN
  - Change of variables
  - Models (nonlinear ICA)

- Implicit density
  - Variational density
  - Variational autoencoder
  - Boltzmann machine

- Markov Chain
  - GSN

Direct GAN

(Goodfellow 2016)
Fully Visible Belief Nets

- Explicit formula based on chain (Frey et al, 1996) rule:

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

- Disadvantages:
  - \(O(n)\) sample generation cost
  - Generation not controlled by a latent code

PixelCNN elephants
(van den Ord et al 2016)
Variational Autoencoder
(Kingma and Welling 2013, Rezende et al 2014)

$$\log p(x) \geq \log p(x) - D_{KL}(q(z) \parallel p(z \mid x))$$
$$= \mathbb{E}_{z \sim q} \log p(x, z) + H(q)$$

Disadvantages:
- Not asymptotically consistent unless $q$ is perfect
- Samples tend to have lower quality

CIFAR-10 samples
(Kingma et al 2016)
GANs

- Use a latent code
- Asymptotically consistent (unlike variational methods)
- No Markov chains needed
- Often regarded as producing the best samples
  - No good way to quantify this

(Goodfellow 2016)
Roadmap

- Why study generative modeling?
- How do generative models work? How do GANs compare to others?
- How do GANs work?
- Tips and tricks
- Research frontiers
- Combining GANs with other methods
Adversarial nets framework

Real data samples

"Fake" Generated Samples

Discriminator

Classify as real (1) or fake (0)

Backpropagate

Generated progressively learning
Generator Network

\[ x = G(z; \theta^{(G)}) \]

- Must be differentiable
- No invertibility requirement
- Trainable for any size of \( z \)
- Some guarantees require \( z \) to have higher dimension than \( x \)
- Can make \( x \) conditionally Gaussian given \( z \) but need not do so

(Goodfellow 2016)
Training Procedure

- Use SGD-like algorithm of choice (Adam) on two minibatches simultaneously:
  - A minibatch of training examples
  - A minibatch of generated samples
- Optional: run $k$ steps of one player for every step of the other player.
Minimax Game

\[
J^{(D)} = -\frac{1}{2} \mathbb{E}_{x \sim p_{\text{data}}} \log D(x) - \frac{1}{2} \mathbb{E}_{z} \log (1 - D(G(z)))
\]

\[
J^{(G)} = -J^{(D)}
\]

- Equilibrium is a saddle point of the discriminator loss
- Resembles Jensen-Shannon divergence
- Generator minimizes the log-probability of the discriminator being correct

(Goodfellow 2016)
Exercise 1

\[ J^{(D)} = -\frac{1}{2} \mathbb{E}_{x \sim p_{\text{data}}} \log D(x) - \frac{1}{2} \mathbb{E}_{z} \log (1 - D(G(z))) \]

\[ J^{(G)} = -J^{(D)} \]

- What is the solution to \( D(x) \) in terms of \( p_{\text{data}} \) and \( p_{\text{generator}} \)?

- What assumptions are needed to obtain this solution?
Solution

- Assume both densities are nonzero everywhere

- If not, some input values $x$ are never trained, so some values of $D(x)$ have undetermined behavior.

- Solve for where the functional derivatives are zero:

\[
\frac{\delta}{\delta D(x)} J^{(D)} = 0
\]

(Goodfellow 2016)
Discriminator Strategy

Optimal $D(x)$ for any $p_{\text{data}}(x)$ and $p_{\text{model}}(x)$ is always

$$D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)}$$

Estimating this ratio using supervised learning is the key approximation mechanism used by GANs.
Learning process

\[ p_D(\text{data}) \rightarrow \text{Data distribution} \rightarrow \text{Model distribution} \]

Poorly fit model
Learning process

\[ p_D(\text{data}) \]

Data distribution

Model distribution

Poorly fit model

After updating D
Learning process

\( p_D(\text{data}) \)

Data distribution

Model distribution

Poorly fit model

After updating D

After updating G
Learning process

\( p_D(\text{data}) \)

Data distribution

Model distribution

Poorly fit model

After updating D

After updating G

Mixed strategy equilibrium
Non-Saturating Game

\[ J^{(D)} = -\frac{1}{2} E_{x \sim p_{\text{data}}} \log D(x) - \frac{1}{2} E_{z} \log (1 - D(G(z))) \]

\[ J^{(G)} = -\frac{1}{2} E_{z} \log D(G(z)) \]

- Equilibrium no longer describable with a single loss
- Generator maximizes the log-probability of the discriminator being mistaken
- Heuristically motivated; generator can still learn even when discriminator successfully rejects all generator samples

(Goodfellow 2016)
Overcoming instabilities in training GANs

Architectural Improvements
DCGAN Architecture

Most “deconvs” are batch normalized

- First stable architecture for generators which works across datasets

- However, limited to smaller resolutions

(Radford et al 2015)

(Goodfellow 2016)
DCGANs for LSUN Bedrooms

(Radford et al 2015)
Progressive growing of GANs (Karras et. al ICLR 2018)

- DCGAN architecture struggles beyond 128x128 resolutions
- Why is higher resolution hard
  - Discrimination task is easier at higher resolutions.
  - Easy to detect artifacts and defects in the generated samples
- Solution: Start training at lower resolutions and progressively increase the resolution.
Progressive growing of GANs (Karras et. al ICLR 2018)

- Mix-in the new layer gradually

- Change $\alpha$ from 0 to 1 gradually
Can generate at 1024x1024 resolution
Overcoming instabilities in training GANs

Better loss functions
Standard GAN is hard to train!

• Gradients are unstable
  ‣ After a few epochs of training, discriminator gets near perfect.
  ‣ This leads to generator having close to zero gradients or exploding gradients (using non-saturating loss).

• Mode collapse

![Figure from “Unrolled generative adversarial networks”, Metz et. al, ICLR 2017](image-url)
Why the instabilities?

- The minmax game minimizes Jensen-Shannon divergence.
- JSD is the average of KL divergences.

\[
K_{\text{L}}(P_r \parallel P_g) = \int_{x} P_r(x) \log \frac{P_r(x)}{P_g(x)} \, dx
\]

\[
\text{JSD}(P_r \parallel P_g) = 0.5 \, \text{KL}(P_r \parallel Q) + 0.5 \, \text{KL}(P_g \parallel Q)
\]

- When the support of the two distributions don’t overlap JSD is constant.

\[
\begin{align*}
P_{G_0} & \quad \longleftrightarrow \quad P_{\text{data}} \quad \longleftrightarrow \quad P_{G_1} \quad \longleftrightarrow \quad P_{\text{data}} \quad \cdots \quad P_{G_{100}} & \quad \longleftrightarrow \quad P_{\text{data}}
\end{align*}
\]

\[
\begin{align*}
\text{JS}(P_{G_0}, P_{\text{data}}) &= \log 2 \\
\text{JS}(P_{G_1}, P_{\text{data}}) &= \log 2 \\
\cdots & \\
\text{JS}(P_{G_{100}}, P_{\text{data}}) &= 0
\end{align*}
\]

- Intuitively: If the supports don’t overlap, binary classifier is perfect!
Solution - Use a better metric: Wasserstein distance

- Earth mover's distance or Wasserstein distance
- Intuitively: Minimum amount of “mass” you need to transport to make $P_g = P_r$

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} E(x, y) \sim \gamma \left[ \| x - y \| \right]$$

Many ways to move the mass. Choose the most efficient way to define the distance.
Is wasserstein distance really better?

\[ J_S(P_{G_0}, P_{data}) = \log 2 \]
\[ W(P_{G_0}, P_{data}) = d_0 \]

\[ J_S(P_{G_{50}}, P_{data}) = \log 2 \]
\[ W(P_{G_{50}}, P_{data}) = d_{50} \]

\[ J_S(P_{G_{100}}, P_{data}) = 0 \]
\[ W(P_{G_{100}}, P_{data}) = 0 \]

Figure credit: Hung-yi Lee
How do you compute it: Wasserstein GAN

• Use the dual for for wasserstein distance

\[
W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]
\]

  ‣ Here f is family of 1-Lipschitz functions

• K-Lipschitz functions
  ‣ Slope never exceeds K. Basically a smoothness constraint on the function f(x)

• Use neural networks to approximate f → Wasserstein GAN formulation.
**Wasserstein GAN** (Arjovsky et. al, ICML 2017)

- Discriminator maximizes the loss function

\[
V(G, D) = \max_{D \in 1-Lipschitz} \{E_{x \sim P_{data}}[D(x)] - E_{x \sim P_{G}}[D(x)]\}
\]

D has to be smooth enough.

- Generator minimizes \(V(G, D)\)

- Comparing to standard GAN, **log is removed** from the loss and there is **no non-linearity at the output** of D.

![Figure credit: Hung-yi Lee](Hung-yiLee.png)

They don’t move.

Wasserstein GAN
Enforcing the Lipschitz constraint on D

• **Weight clipping** (Arjovsky et. al, ICML 2017)
  ‣ All parameters of the discriminator constrained to be in a box \([-c, c]\)
  ‣ Difficult to find the right value for c for different problems.

• **Gradient penalty** (Improved training of WGAN, Gulrajani et. al, NIPS 2017)
  ‣ Constrain the gradients of the discriminator w.r.t the input to be small
    \[ \lambda \mathbb{E}_{\hat{x} \sim \mathcal{P}_x} \left[ \left( \| \nabla_{\hat{x}} D(\hat{x}) \|_2 - 1 \right)^2 \right] \]
  ‣ Ideally should be everywhere in the image space, but too expensive.
  ‣ Impose the penalty only on a line between generated sample and real sample
## Comparing Standard GAN and WGAN

<table>
<thead>
<tr>
<th></th>
<th>DCGAN</th>
<th>LSGAN</th>
<th>WGAN (clipping)</th>
<th>WGAN-GP (ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baseline</strong></td>
<td><img src="image1" alt="Baseline" /></td>
<td><img src="image2" alt="Baseline" /></td>
<td><img src="image3" alt="Baseline" /></td>
<td><img src="image4" alt="Baseline" /></td>
</tr>
<tr>
<td><strong>G</strong>: No BN and a constant number of filters, <strong>D</strong>: DCGAN</td>
<td><img src="image5" alt="G: No BN and a constant number of filters" /></td>
<td><img src="image6" alt="G: No BN and a constant number of filters" /></td>
<td><img src="image7" alt="G: No BN and a constant number of filters" /></td>
<td><img src="image8" alt="G: No BN and a constant number of filters" /></td>
</tr>
<tr>
<td><strong>G</strong>: 4-layer 512-dim ReLU MLP, <strong>D</strong>: DCGAN</td>
<td><img src="image9" alt="G: 4-layer 512-dim ReLU MLP" /></td>
<td><img src="image10" alt="G: 4-layer 512-dim ReLU MLP" /></td>
<td><img src="image11" alt="G: 4-layer 512-dim ReLU MLP" /></td>
<td><img src="image12" alt="G: 4-layer 512-dim ReLU MLP" /></td>
</tr>
<tr>
<td><strong>No normalization in either G or D</strong></td>
<td><img src="image13" alt="No normalization in either G or D" /></td>
<td><img src="image14" alt="No normalization in either G or D" /></td>
<td><img src="image15" alt="No normalization in either G or D" /></td>
<td><img src="image16" alt="No normalization in either G or D" /></td>
</tr>
<tr>
<td><strong>Gated multiplicative nonlinearities everywhere in G and D</strong></td>
<td><img src="image17" alt="Gated multiplicative nonlinearities everywhere in G and D" /></td>
<td><img src="image18" alt="Gated multiplicative nonlinearities everywhere in G and D" /></td>
<td><img src="image19" alt="Gated multiplicative nonlinearities everywhere in G and D" /></td>
<td><img src="image20" alt="Gated multiplicative nonlinearities everywhere in G and D" /></td>
</tr>
<tr>
<td><strong>tanh nonlinearities everywhere in G and D</strong></td>
<td><img src="image21" alt="tanh nonlinearities everywhere in G and D" /></td>
<td><img src="image22" alt="tanh nonlinearities everywhere in G and D" /></td>
<td><img src="image23" alt="tanh nonlinearities everywhere in G and D" /></td>
<td><img src="image24" alt="tanh nonlinearities everywhere in G and D" /></td>
</tr>
<tr>
<td><strong>101-layer ResNet G and D</strong></td>
<td><img src="image25" alt="101-layer ResNet G and D" /></td>
<td><img src="image26" alt="101-layer ResNet G and D" /></td>
<td><img src="image27" alt="101-layer ResNet G and D" /></td>
<td><img src="image28" alt="101-layer ResNet G and D" /></td>
</tr>
</tbody>
</table>

WGAN-GP is stable to architectural changes.
Learning What and Where to Draw

Scott Reed\textsuperscript{1,3}, Zeynep Akata\textsuperscript{2}, Santosh Mohan\textsuperscript{1}, Samuel Tenka\textsuperscript{1}, Bernt Schiele\textsuperscript{2}, Honglak Lee\textsuperscript{1}
Motivation

a pitcher is about to throw the ball to the batter.

What object is meant to be drawn here? Can we control its location?
Idea: condition on location in addition to text

1. Bounding box

   ![Bounding box example](image1)

   This bird is completely black.

2. Keypoints, e.g. 15 parts of a bird

   ![Keypoint example](image2)

   This bird is bright blue.
Text-conditional GAN

$$\min_G \max_D V(D, G) = \mathbb{E}_{x,t \sim p_{data}(x,t)}[\log D(x, t)] + \mathbb{E}_{z \sim p_z(z), t \sim p_{data}(t)}[\log(1 - D(G(z, t)))].$$

- The discriminator $D$ tries to distinguish real (text, image) pairs from synthetic.
- The generator $G$ tries to fool $D$. 
Text-conditional GAN

This flower has small, round violet petals with a dark purple center

$z \sim \mathcal{N}(0, 1)$

Generator Network
Text-conditional GAN

This flower has small, round violet petals with a dark purple center

$z \sim \mathcal{N}(0, 1)$

Generator Network
Text-conditional GAN

This flower has small, round violet petals with a dark purple center

$z \sim \mathcal{N}(0, 1)$

Generator Network

Discriminator Network

This flower has small, round violet petals with a dark purple center
Conditioning on bounding box

A red bird with a black face

Generator Network
Conditioning on bounding box

Spatial replicate, crop to bbox

A red bird with a black fa

Generator Network
Conditioning on bounding box

Generator Network
Conditioning on bounding box

Spatial replicate, crop to bbox

A red bird with a black face

\( z \sim \mathcal{N}(0, 1) \)

Generator Network
Conditioning on bounding box

Generator Network
Conditioning on bounding box

Spatial replicate, crop to bbox

A red bird with a black face

$z \sim \mathcal{N}(0, 1)$

Local

Global

depth concat

Generator Network
Conditioning on bounding box

Generator Network

Discriminator Network

A red bird with a black face

$z \sim \mathcal{N}(0, 1)$
Moving the bird around with bounding box (noize z fixed)

Caption

This bird has a black head, a long orange beak and yellow body.
Moving the bird around with bounding box (noize z fixed)

Caption: This bird has a black head, a long orange beak and yellow body.
Moving the bird around with bounding box (noize $z$ fixed)

**Caption**

This bird has a black head, a long orange beak and yellow body.

**GT**

**Translation**
Moving the bird around with bounding box (noize z fixed)

<table>
<thead>
<tr>
<th>Caption</th>
<th>GT</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bird has a black head, a long orange beak and yellow body</td>
<td></td>
</tr>
</tbody>
</table>
Moving the bird around with bounding box (noize z fixed)

<table>
<thead>
<tr>
<th>Caption</th>
<th>GT</th>
<th>Shrinking</th>
<th>Translation</th>
<th>Stretching</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bird has a black head, a long orange beak and yellow body</td>
<td><img src="image1.png" alt="Bird image" /></td>
<td><img src="image2.png" alt="Bird image" /></td>
<td><img src="image3.png" alt="Bird image" /></td>
<td><img src="image4.png" alt="Bird image" /></td>
</tr>
<tr>
<td>This large black bird has a pointy beak and black eyes</td>
<td><img src="image5.png" alt="Bird image" /></td>
<td><img src="image6.png" alt="Bird image" /></td>
<td><img src="image7.png" alt="Bird image" /></td>
<td><img src="image8.png" alt="Bird image" /></td>
</tr>
<tr>
<td>This small blue bird has a short pointy beak and brown patches on its wings</td>
<td><img src="image9.png" alt="Bird image" /></td>
<td><img src="image10.png" alt="Bird image" /></td>
<td><img src="image11.png" alt="Bird image" /></td>
<td><img src="image12.png" alt="Bird image" /></td>
</tr>
</tbody>
</table>
Moving the bird around with key points (noize z fixed)

Caption
This bird has a black head, a long orange beak and yellow body
Moving the bird around with key points (noize $z$ fixed)

**Caption**

This bird has a black head, a long orange beak and yellow body.

**GT**

**Translation**
Moving the bird around with key points (noize \( z \) fixed)

**Caption**

This bird has a black head, a long orange beak and yellow body

**GT**

**Stretching**
Moving the bird around with key points (noize $z$ fixed)

Caption

This bird has a black head, a long orange beak and yellow body.

This large black bird has a pointy beak and black eyes.

This small blue bird has a short pointy beak and brown patches on its wings.
Deep Model Adaptation using Domain Adversarial Training

Victor Lempitsky,

*joint work with* Yaroslav Ganin

Skoltech

*Skolkovo Institute of Science and Technology (Skoltech)*

*Moscow region, Russia*
Deep supervised neural networks

- are a “big thing” in computer vision and beyond
- are hungry for labeled data
Where to get the data?

Lots of modalities do not have large labeled data sets:
- Biomedical
- Unusual cameras / image types
- Videos
- Data with expert-level annotation (not mTurkable)
- ....

**Surrogate training data** often available:
- Borrow from adjacent modality
- Generate synthetic imagery (computer graphics)
- Use data augmentation to amplify data *(image-based rendering, morphing, re-synthesis, ....)*

Resulting training data are shifted. *Domain adaptation* needed.
Example: Internet images -> Webcam sensor

[Saenko et al. ECCV2010]

Deep Model Adaptation using Domain Adversarial Training
Assumptions and goals

- Lots of labeled data in the source domain (e.g. synthetic images)
- Lots of unlabeled data in the target domain (e.g. real images)
- **Goal:** train a deep neural net that does well on the target domain

Large-scale deep unsupervised domain adaptation
Domain shift in a deep architecture

\[ f = G_f(x; \theta_f) \]
\[ y = G_y(f; \theta_y) \]

When trained on source only, feature distributions do not match:

\[ S(f) = \{ G_f(x; \theta_f) \mid x \sim S(x) \} \]
\[ T(f) = \{ G_f(x; \theta_f) \mid x \sim T(x) \} \]
Idea 1: domain-invariant features wanted

Feature distribution without adaptation:

Our goal (after adaptation):
Idea 2: measuring domain shift

Domain classifier:

\[ d = G_d(f; \theta_d) \]

Domain loss low

Domain loss high

Deep Model Adaptation using Domain Adversarial Training
Learning with adaptation

1. Build this network
2. Train **feature extractor + class predictor** on source data
3. Train **feature extractor + domain classifier** on source+target data
4. Use **feature extractor + class predictor** at test time

Deep Model Adaptation using Domain Adversarial Training
Idea 3: minimizing domain shift

Emerging features:
- Discriminative (good for predicting $y$)
- Domain-invariant (not good for predicting $d$)
Saddle point interpretation

Our objective (small label prediction loss + large domain classification loss wanted)

$$E(\theta_f, \theta_y, \theta_d) = \sum_{i=1..N}^{d_i=0} L_y^i(\theta_f, \theta_y) - \lambda \sum_{i=1..N} L_d^i(\theta_f, \theta_d)$$

The backprop converges to a saddle point:

$$(\hat{\theta}_f, \hat{\theta}_y) = \arg \min_{\theta_f, \theta_y} E(\theta_f, \theta_y, \theta_d)$$

$$\hat{\theta}_d = \arg \max_{\theta_d} E(\hat{\theta}_f, \hat{\theta}_y, \theta_d).$$

Similar idea for generative networks:

Initial experiments: baselines

Upper bound: training on target domain with labels

Shallow adaptation baseline: [Fernando et al., Unsupervised visual domain adaptation using subspace alignment. ICCV, 2013] applied to the last-but-one layer

Lower bound: training on source domain only
Example: from synthetic to real

“Windows digits”

“House numbers”

Deep Model Adaptation using Domain Adversarial Training
Office dataset

[Saenko et al. ECCV2010]

Deep Model Adaptation using Domain Adversarial Training
## Results on Office dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Source Target</th>
<th>Amazon Webcam</th>
<th>DSLR Webcam</th>
<th>Webcam DSLR</th>
</tr>
</thead>
<tbody>
<tr>
<td>GFK (PLS, PCA) (Gong et al., 2012)</td>
<td></td>
<td>.197</td>
<td>.497</td>
<td>.6631</td>
</tr>
<tr>
<td>SA* (Fernando et al., 2013)</td>
<td></td>
<td>.450</td>
<td>.648</td>
<td>.699</td>
</tr>
<tr>
<td>DLID (Chopra et al., 2013)</td>
<td></td>
<td>.519</td>
<td>.782</td>
<td>.899</td>
</tr>
<tr>
<td>DDC (Tzeng et al., 2014)</td>
<td></td>
<td>.618</td>
<td>.950</td>
<td>.985</td>
</tr>
<tr>
<td>DAN (Long and Wang, 2015)</td>
<td></td>
<td>.685</td>
<td>.960</td>
<td>.990</td>
</tr>
<tr>
<td>Source only</td>
<td></td>
<td>.642</td>
<td>.961</td>
<td>.978</td>
</tr>
<tr>
<td>DANN</td>
<td></td>
<td>.730</td>
<td>.964</td>
<td>.992</td>
</tr>
</tbody>
</table>

Caveats

- Domains should not be too far apart
- Early on, the gradient from the domain classification loss should not be too strong
- The trick used to obtain the results: gradually increase $\lambda$ from 0 to 1
Conclusion

- Scalable method for deep unsupervised domain adaptation
- Based on simple idea. Takes few lines of code (+ defining a specific network architecture). *Caffe* implementation available.
- State-of-the-art results
- Unsupervised parameter tuning is easy (look at the domain classifier error)
- Main challenge: initialization and stepsize

http://sites.skoltech.ru/compvision/projects/grl/
Unsupervised Image Translation

Cycle GAN

Star GAN
Unpaired data

- Easy to get unpaired data. But can’t do supervised learning
- Idea: use domain classifiers to get weak supervision.
Learn from classifier: GAN framework

Figure credit: Hung-yi Lee
Learning from classifier is not sufficient

Figure credit: Hung-yi Lee
Use cycle consistency principle: Cycle GAN

(Zhu et. al ICCV 2017)

Figure credit: Hung-yi Lee
Results

• Impressive visual results

• Caveat! Needs a separate generator for every ordered pair of domains.

• Not scalable.
Stargan (Choi et.al, CVPR 2018)

- Use a multi-class discriminator and a single generator.
- Condition the generator on the target domain, along with the input image.
Allows single model to map from different domains

- Manipulating the target domain condition, generator can map input image to different domains.