Domain transfer

Karel Zimmermann
http://cmp.felk.cvut.cz/~zimmerk/

Vision for Robotics and Autonomous Systems
https://cyber.felk.cvut.cz/vras/

Center for Machine Perception
https://cmp.felk.cvut.cz

Department for Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
Motivation

• State-of-the-art algorithms based on deep-learning
• Deep learning is data-hungry.
• Learning on different domains often yields domain bias (training/testing distribution discrepancy)
• Domain transfer is about exploiting data from different domains.
The most straightforward method is to train on a simulator

- Physical simulation based Open Dynamics Engine with physical parameters tuned on real sequences
Tracked robot simulation
Pecka, Zimmermann et al., [IROS, 2017]
The most straightforward method is to train on a simulator

- Physical simulation based Open Dynamics Engine with physical parameters tuned on real sequences.
- Reverse engineering of GTA 5 (RAGE engine).
RGB images
Depth images
Stencil layer
Stencil layer - **cars**

**glass is not car**
Stencil layer - *humans*

driver’s hand
Stencil layer - trees
Stencil layer - sky
Stencil layer - artificial light
Stencil layer - *artificial light*
Other annotations for objects (e.g. cars, humans)

- 2D bounding box in Image Coordinates (IC)
- 3D bounding box in World Coordinates (WC)
- Position and rotation in WC
- Entity type (e.g. car, pedestrian)
- Entity class (e.g. sedan, SUV, coupe …)
- Unique ID (create trajectories, estimate motion flow)
Other annotations for objects (e.g. cars, humans)
Other annotations for objects (e.g. cars, humans)
What can be controlled

• Motion of objects:
  • explicit (e.g. shift in WCF),
  • implicit (e.g. car driving, autopilot)
• **Time in the day/night cycle**
What can be controlled
What can be controlled

• Motion of objects:
  • explicit (e.g. shift in WCF),
  • implicit (e.g. car driving, autopilot)
• Time in the day/night cycle
• **Weather** (ExtraSunny, Clear, Clouds, Smog, Foggy, Overcast, Raining, ThunderStorm, Clearing, Neutral, Snowing, Blizzard, Snowlight, Christmas, Halloween)
ExtraSunny

Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics
SnowLight
What can be controlled

- Motion of objects:
  - explicit (e.g. shift in WCF),
  - implicit (e.g. car driving, autopilot)
- Time in the day/night cycle
- Weather (ExtraSunny, Clear, Clouds, Smog, Foggy, Overcast, Raining, ThunderStorm, Clearing, Neutral, Snowing, Blizzard, Snowlight, Christmas, Halloween)
- **Visual mods** (Redux, NaturalVision, Vanilla)
What can be controlled

• Motion of objects:
  • explicit (e.g. shift in WCF),
  • implicit (e.g. car driving, autopilot)
• Time in the day/night cycle
• Weather (ExtraSunny, Clear, Clouds, Smog, Foggy, Overcast, Raining, ThunderStorm, Clearing, Neutral, Snowing, Blizzard, Snowlight, Christmas, Halloween)
• Visual mods (Redux, NaturalVision, Vanilla)
  • **Custom object models from CAD**
What can be controlled

• Motion of objects:
  • explicit (e.g. shift in WCF),
  • implicit (e.g. car driving, autopilot)
• Time in the day/night cycle
• Weather (ExtraSunny, Clear, Clouds, Smog, Foggy, Overcast, Raining, ThunderStorm, Clearing, Neutral, Snowing, Blizzard, Snowlight, Christmas, Halloween)
• Visual mods (Redux, NaturalVision, Vanilla)
• Custom vehicle models from CAD
• **Custom maps a scenarios** (probability of spawning different objects in different areas, complex scripts)
Map editor
Tsunami mods

https://cs.gta5-mods.com/misc/tsunami-mod
Plane crashing mods

https://cs.gta5-mods.com/scripts/planes-hails

Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics
Driving in the matrix [Roberson ICRA 2017]
https://arxiv.org/abs/1610.01983

• Reverse engineering of GTA 5 (RAGE engine)
The most straightforward method is to train on a simulator

- Physical simulation based Open Dynamics Engine with physical parameters tuned on real sequences.
- Reverse engineering of GTA 5 (RAGE engine).
- Training data from simulator added to the training set, but we can do better.
- Generative models can be trained to generate new training samples.
Generative models - variational encoders

- Training set (2D vectors)
- CNN
- Low dimensional encoding
- Reconstructed training set

• Learning the self-reconstruction with L2 reconstruction loss

\[ \arg \min_w \| \mathbf{x} - \hat{\mathbf{x}}(w) \|_2^2 \]
Generative models - variational encoders

- Learning the self-reconstruction with L2 reconstruction loss

\[ \arg \min_w \|x - \hat{x}(w)\|_2^2 \]

- If CNN linear functions then closed-form solution is PCA
Generative models

• New samples generated from random vectors in low-dimensional encoding.
Input: training set faces with and without facial hair
Output: new faces with facial hair
Deep Feature interpolations [Upchurch CVPR 2017]

- Input: training set faces with and without facial hair
- Output: new faces with facial hair
- Transfer between different domain usually means unpaired correspondences !!!
- Such problem does not allow for direct minimization of a supervised loss !!!
Deep Feature interpolations [Upchurch CVPR 2017]  

- All samples projected to deep feature space by VGG-19

Czech Technical University in Prague  
Faculty of Electrical Engineering, Department of Cybernetics
Deep Feature interpolations [Upchurch CVPR 2017]

• All samples projected to deep feature space by VGG-19
Deep Feature interpolations [Upchurch CVPR 2017]

• New sample (without facial hair) projected to deep feature space
Deep Feature interpolations [Upchurch CVPR 2017]

- Nearest neighbour in deep feature space found
Deep Feature interpolations [Upchurch CVPR 2017]

- New sample generated by interpolation in deep feature space.
Deep Feature interpolations [Upchurch CVPR 2017]

- Reconstruction: gradient descent optimization with loss:
  - similarity in feature space
  - smoothness RGB values
Deep Feature interpolations [Upchurch CVPR 2017]

• Reconstruction: gradient descent optimization with loss:
  • similarity in feature space
  • smoothness RGB values
Deep Feature interpolations [Upchurch CVPR 2017]

Deep Feature interpolations [Upchurch CVPR 2017]
Generative Adversarial Nets [Goodfellow NIPS 2014]
https://arxiv.org/abs/1406.2661
real samples

2D vectors from uniform distribution

fake samples

generator

discriminator

real/fake
Generative Adversarial Nets [Goodfellow NIPS 2014]
https://arxiv.org/abs/1406.2661

2D vectors from uniform distribution

generator

discriminator

fake samples

real samples

real/fake
Generative Adversarial Nets [Goodfellow NIPS 2014]
https://arxiv.org/abs/1406.2661

real samples

2D vectors from uniform distribution

generator

discriminator

real/fake

fake samples
Generative Adversarial Nets [Goodfellow NIPS 2014]
https://arxiv.org/abs/1406.2661

2D vectors from uniform distribution

real samples

fake samples

real/fake

generator

discriminator
Generative Adversarial Nets [Goodfellow NIPS 2014]
https://arxiv.org/abs/1406.2661

2D vectors from uniform distribution

\[ x \in X \]

\[ y \in Y \]

\[ g(x) \]

\[ d(y) \]

real samples

real/fake

fake samples
Generative Adversarial Nets [Goodfellow NIPS 2014]
https://arxiv.org/abs/1406.2661

\[ \mathcal{L}(d, g) = \sum_{x \in X} -\log(d(g(x))) + \sum_{y \in Y} -\log(1 - d(y)) \]
Generative Adversarial Nets [Goodfellow NIPS 2014]
https://arxiv.org/abs/1406.2661

2D vectors from uniform distribution

\[ \mathbf{x} \in X \]

\[ \mathbf{y} \in Y \]

\[ \mathbf{d}(\mathbf{y}) \]

\[ \mathbf{d}(\mathbf{x}) \]

\[ \mathbf{g}(\mathbf{x}) \]

\[ \mathbf{L}(\mathbf{d}, \mathbf{g}) = \sum_{\mathbf{x} \in X} - \log(\mathbf{d}(\mathbf{g}(\mathbf{x}))) + \sum_{\mathbf{y} \in Y} - \log(1 - \mathbf{d}(\mathbf{y})) \]

\[ (\mathbf{d}^*, \mathbf{g}^*) = \arg \min_{\mathbf{d}} \arg \max_{\mathbf{g}} \mathbf{L}(\mathbf{d}, \mathbf{g}) \]
Equilibrium in saddle point implies that generator generates samples from the real distribution (asymptotically consistent in contrast to VAE)

You can play with GANs online:
https://poloclub.github.io/ganlab/

\[(d^*, g^*) = \arg\min_d \arg\max_g \mathcal{L}(d, g)\]
Direct application to domain transfer
real samples $y \in Y$

$g(x)$

real/fake

$d(y)$

GTA samples $x \in X$

fake samples
Direct application to domain transfer

GTA samples $x \in X$  fake samples

[Jasek DP 2017]

Czech Technical University in Prague  Faculty of Electrical Engineering, Department of Cybernetics
Cycle-GAN [Zhu ICCV 2017]
https://arxiv.org/abs/1703.10593

real X-domain samples $x \in X$

$g_x(x)$

real Y-domain samples $y \in Y$

d_y(y)

fake Y-domain samples $\hat{y}$

real/fake
Cycle-GAN [Zhu ICCV 2017]
https://arxiv.org/abs/1703.10593
Cycle-GAN [Zhu ICCV 2017]  
https://arxiv.org/abs/1703.10593

\[ \mathcal{L}_{GAN}(d_x, d_y, g_x, g_y) = \mathcal{L}(d_y, g_x) + \mathcal{L}(d_x, g_y) \]

Real Y-domain samples \( y \in Y \)  
Fake X-domain samples \( \hat{x} \)  
Real X-domain samples \( x \in X \)  
Real Y-domain samples \( y \in Y \)
Cycle-GAN [Zhu ICCV 2017]
https://arxiv.org/abs/1703.10593

\[
\mathcal{L}_{GAN}(d_x, d_y, g_x, g_y) = \mathcal{L}(d_y, g_x) + \mathcal{L}(d_x, g_y) + |g_x(g_y(y) - \hat{y})|
\]

real/fake samples \( y \in Y \)

real Y-domain samples

fake Y-domain samples \( \hat{y} \)

real X-domain samples \( x \in X \)

fake X-domain samples \( \hat{x} \)

real/fake samples

Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Cybernetics
Cycle-GAN [Zhu ICCV 2017]
https://arxiv.org/abs/1703.10593

$$\mathcal{L}_{GAN}(d_x, d_y, g_x, g_y) = \mathcal{L}(d_y, g_x) + \mathcal{L}(d_x, g_y) + |g_x(g_y(y) - \hat{y}|$$

real Y-domain samples $y \in Y$
real Y-domain samples $y \in Y$
real Y-domain samples $y \in Y$
real Y-domain samples $y \in Y$
real Y-domain samples $y \in Y$
real Y-domain samples $y \in Y$

VAE with X-embedding

real X-domain samples $x \in X$
real X-domain samples $x \in X$
real X-domain samples $x \in X$
real X-domain samples $x \in X$
real X-domain samples $x \in X$
real X-domain samples $x \in X$

real/fake samples $\hat{x}$
real/fake samples $\hat{x}$
real/fake samples $\hat{x}$
real/fake samples $\hat{x}$
real/fake samples $\hat{x}$
real/fake samples $\hat{x}$
Cycle-GAN [Zhu ICCV 2017]
https://arxiv.org/abs/1703.10593
Cycle-GAN [Zhu ICCV 2017]
https://arxiv.org/abs/1703.10593
Cycle-GAN [Zhu ICCV 2017]
https://arxiv.org/abs/1703.10593
CyCaDa-GAN [Hoffman CVPR 2018]

• Cycle consistency helps, but it still allows to learn totally semantically inconsistent transfer

\[
\mathcal{L}_{GAN}(d_x, d_y, g_x, g_y) = \mathcal{L}(d_y, g_x) + \mathcal{L}(d_x, g_y) + |g_x(g_y(y) - \hat{y}|
\]
CyCaDa-GAN [Hoffman CVPR 2018]
CyCaDa-GAN [Hoffman CVPR 2018]
CyCaDa-GAN [Hoffman CVPR 2018]

• Semantic consistency enforce transformation to be semantically consistent

Source images (SVHN)  Adapted source images (Ours)
Adversarial attacks [Arnab CVPR 2018]
https://github.com/hmph/adversarial-attacks

• Given trained network and training example, find closest example on which the network fails
Domain transfer summary

- It seems to be a solution for always insufficient number of training examples.
- It can be seen as regularization (e.g. additional prior tells that the classifier should work also on transformed images from other domain and the transformation should be simple)
- Few more links:
  - Time contrastive nets: https://sermanet.github.io/imitate/
  - UNIT GAN https://github.com/mingyuliutw/UNIT