Generative Adversarial Networks

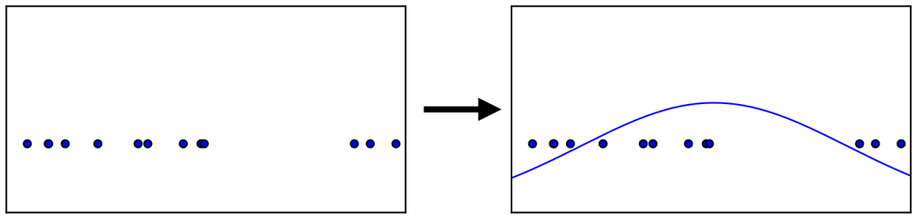
Lecture 11

Slides from: Emily Denton, Ian Goodfellow, Soumith Chintala

Generative Adversarial Networks

- [Generative Adversarial Nets, Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, NIPS 2014]
- Focus on sample generation

Generative Modeling: Density Estimation



Training Data | Density Function

Generative Modeling: Sample Generation



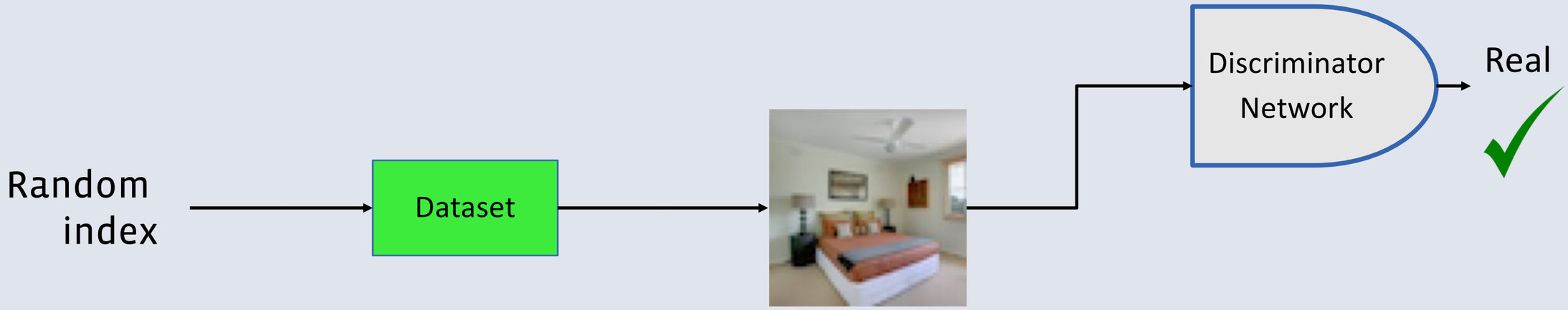
Training Data (CelebA) | Sample Generator (Karras et al, 2017)

(Goodfellow 2018)

Generative Adversarial Network

[Goodfellow et al. NIPS 2014]

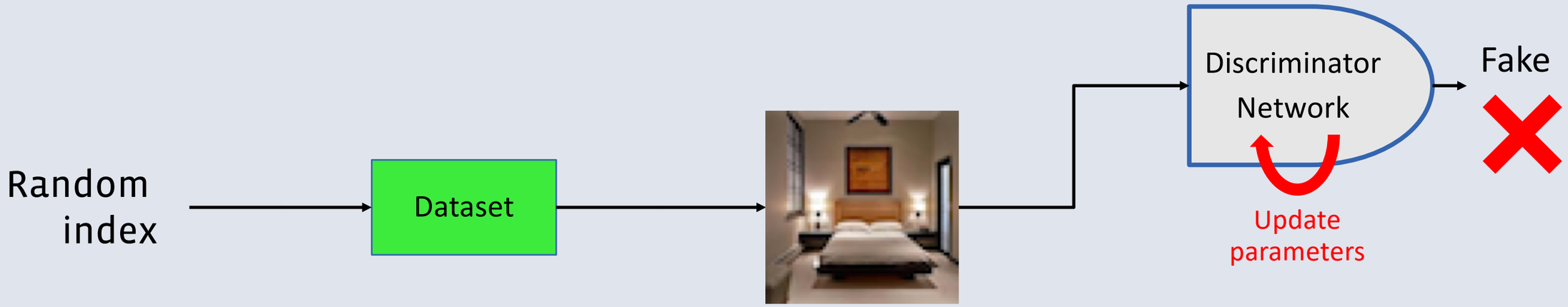
- Initial application to still images
- Way to train generative model to match **distribution** of data
- Discriminator network predicts if input image is from data (real) or model (fake)



Generative Adversarial Network

[Goodfellow et al. NIPS 2014]

- Initial application to still images
- Way to train generative model to match **distribution** of data
- Discriminator network predicts if input image is from data (real) or model (fake)



Generative Adversarial Network

[Goodfellow et al. NIPS 2014]

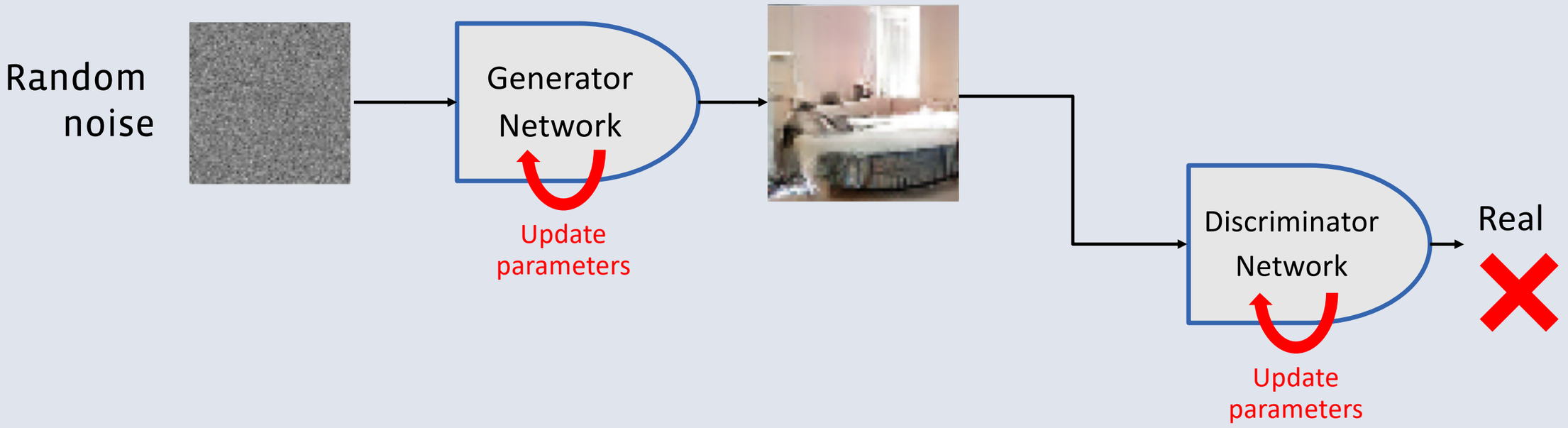
- Initial application to still images
- Way to train generative model to match **distribution** of data
- Discriminator network predicts if input image is from data (real) or model (fake)
- Generator network tries to confuse Discriminator



Generative Adversarial Network

[Goodfellow et al. NIPS 2014]

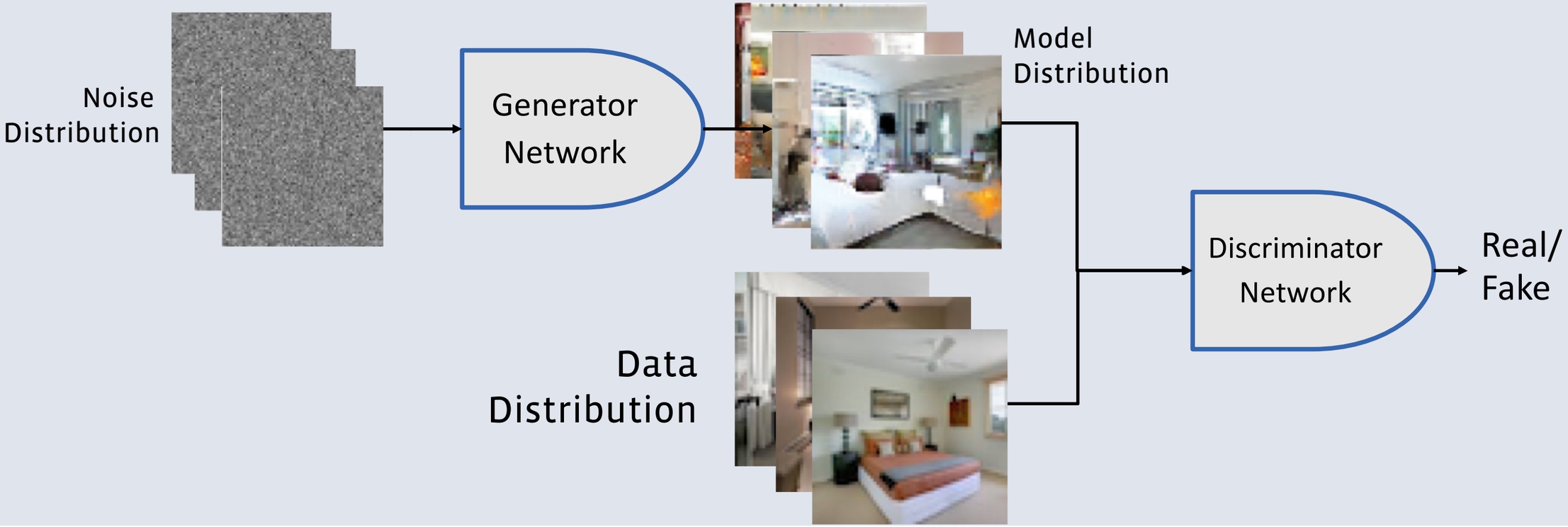
- Initial application to still images
- Way to train generative model to match **distribution** of data
- Discriminator network predicts if input image is from data (real) or model (fake)
- Generator network tries to confuse discriminator



Generative Adversarial Network

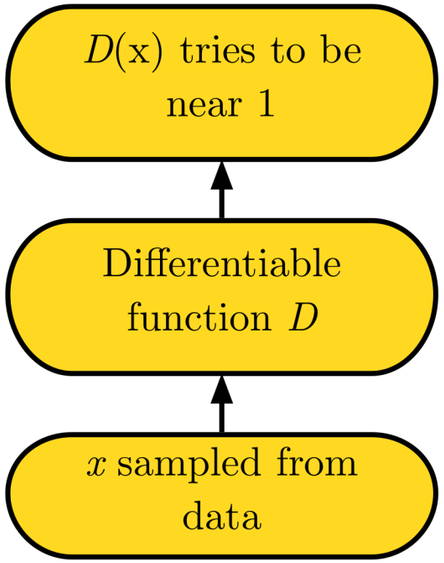
[Goodfellow et al. NIPS 2014]

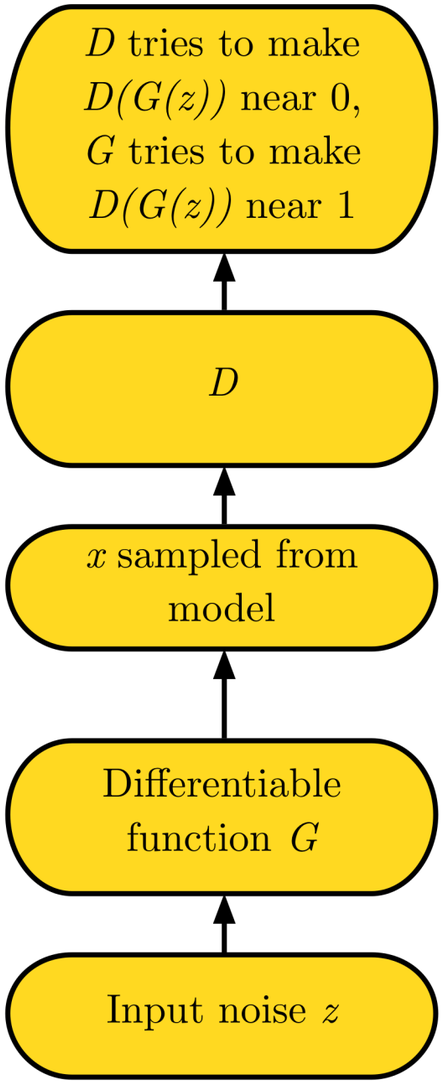
- Initial application to still images
- Way to train generative model to match **distribution** of data
- Discriminator network predicts if input image is from data (real) or model (fake)
- Generator network tries to confuse Discriminator

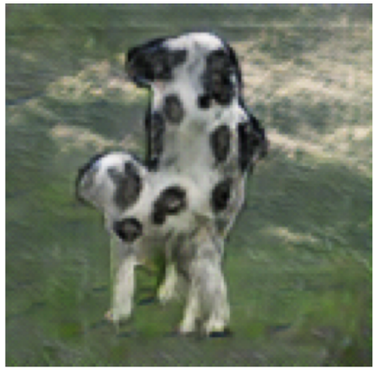


Generative Adversarial Networks









(Goodfellow et al., 2014)

Generative Adversarial Networks

[Generative Adversarial Nets, Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, NIPS 2014]

- Minimax value function:

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

Generator pushes down

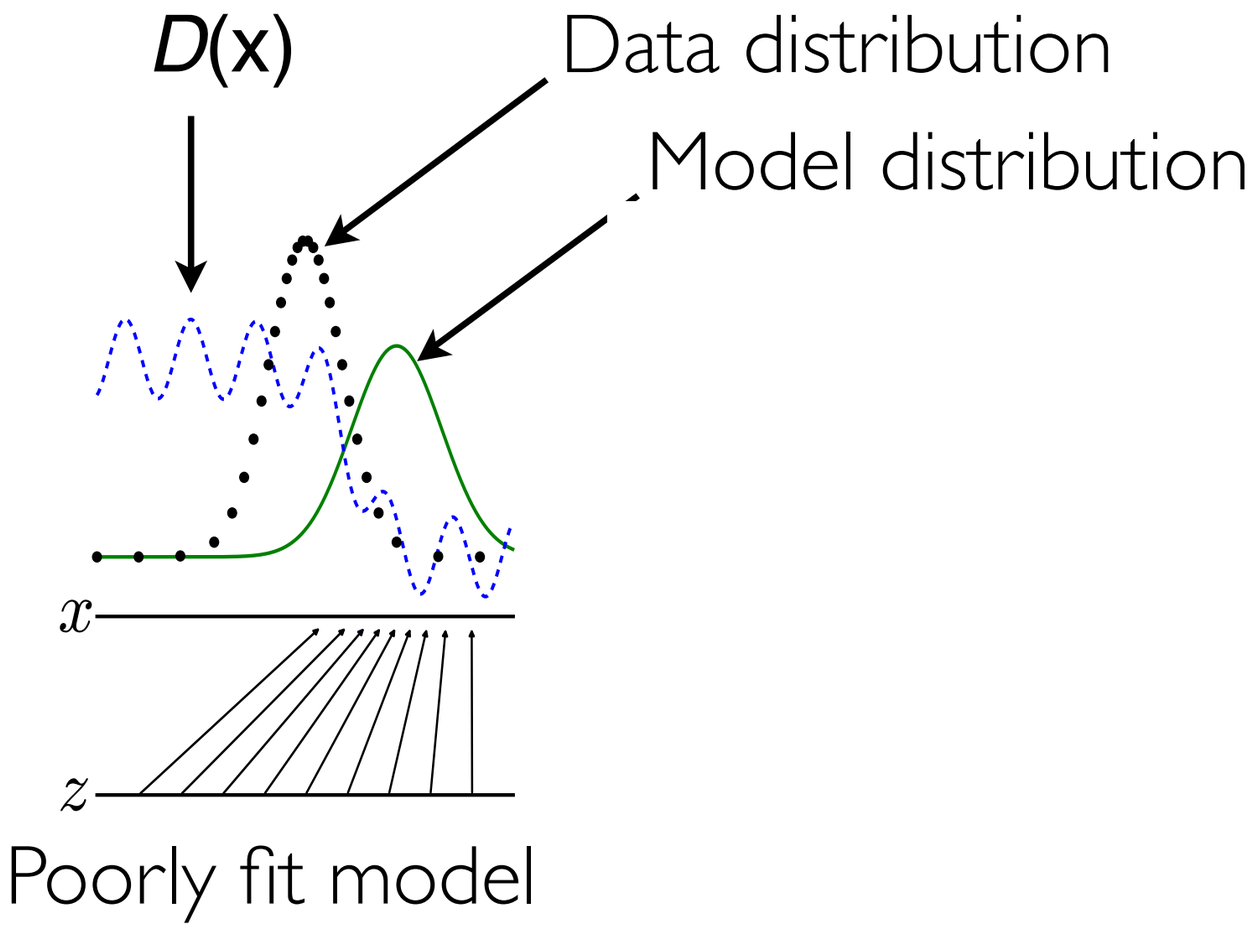
Discriminator pushes up

Discriminator's ability to recognize data as being real

Discriminator's ability to recognize generator samples as being fake

[Slide: Ian Goodfellow, Deep Learning workshop, ICML 2015]

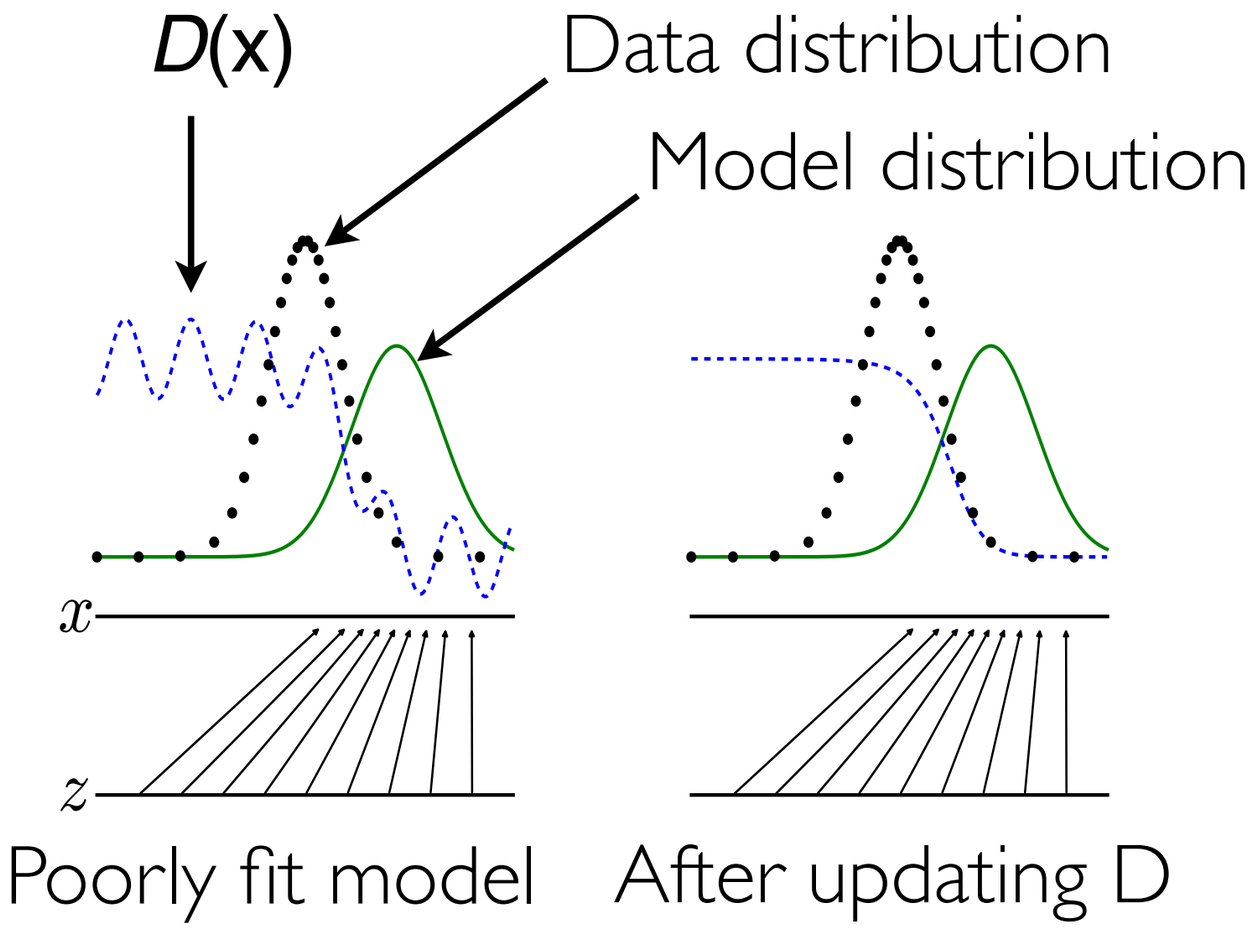
Generative Adversarial Networks



Poorly fit model

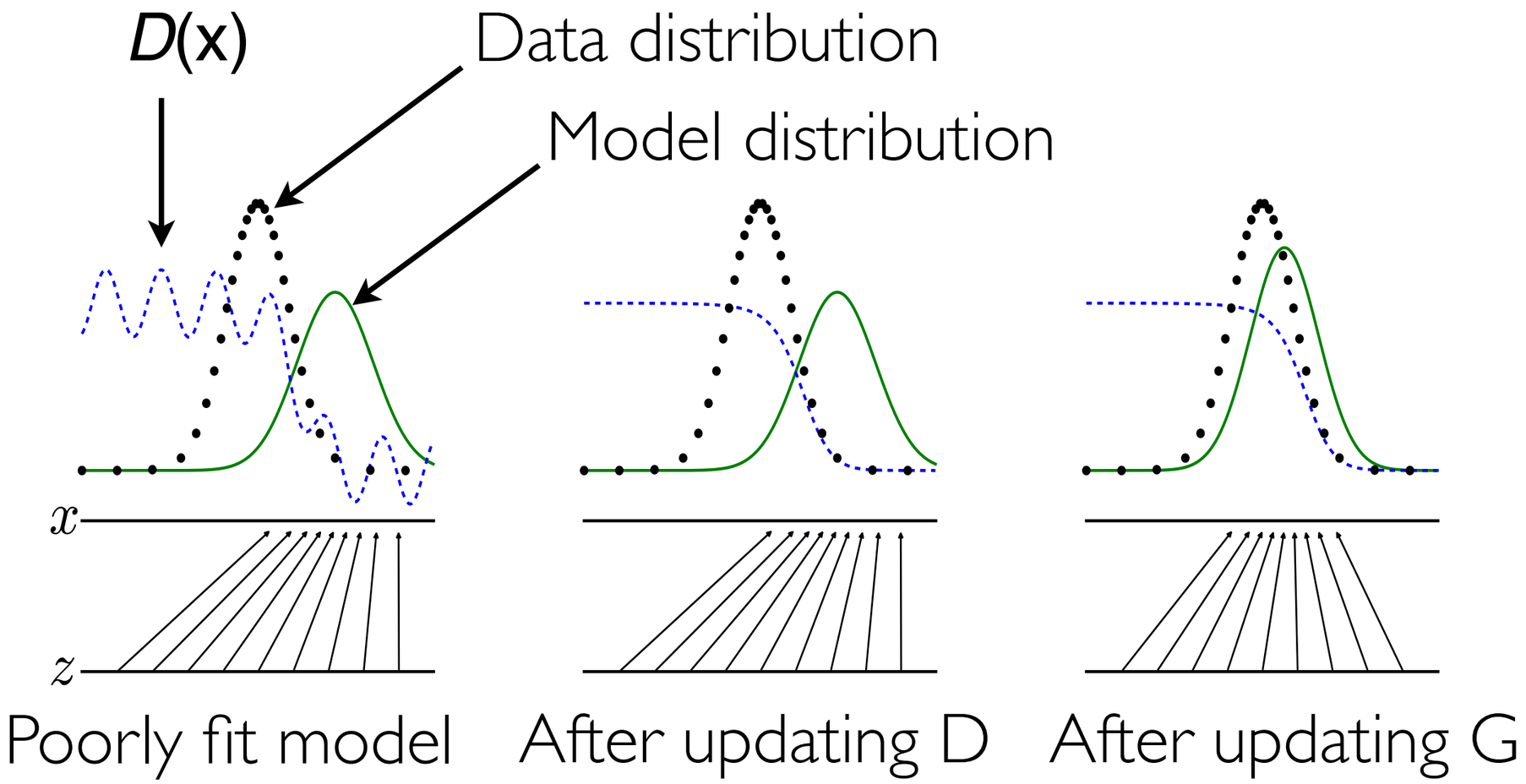
[Slide: Ian Goodfellow, Deep Learning workshop, ICML 2015]

Generative Adversarial Networks



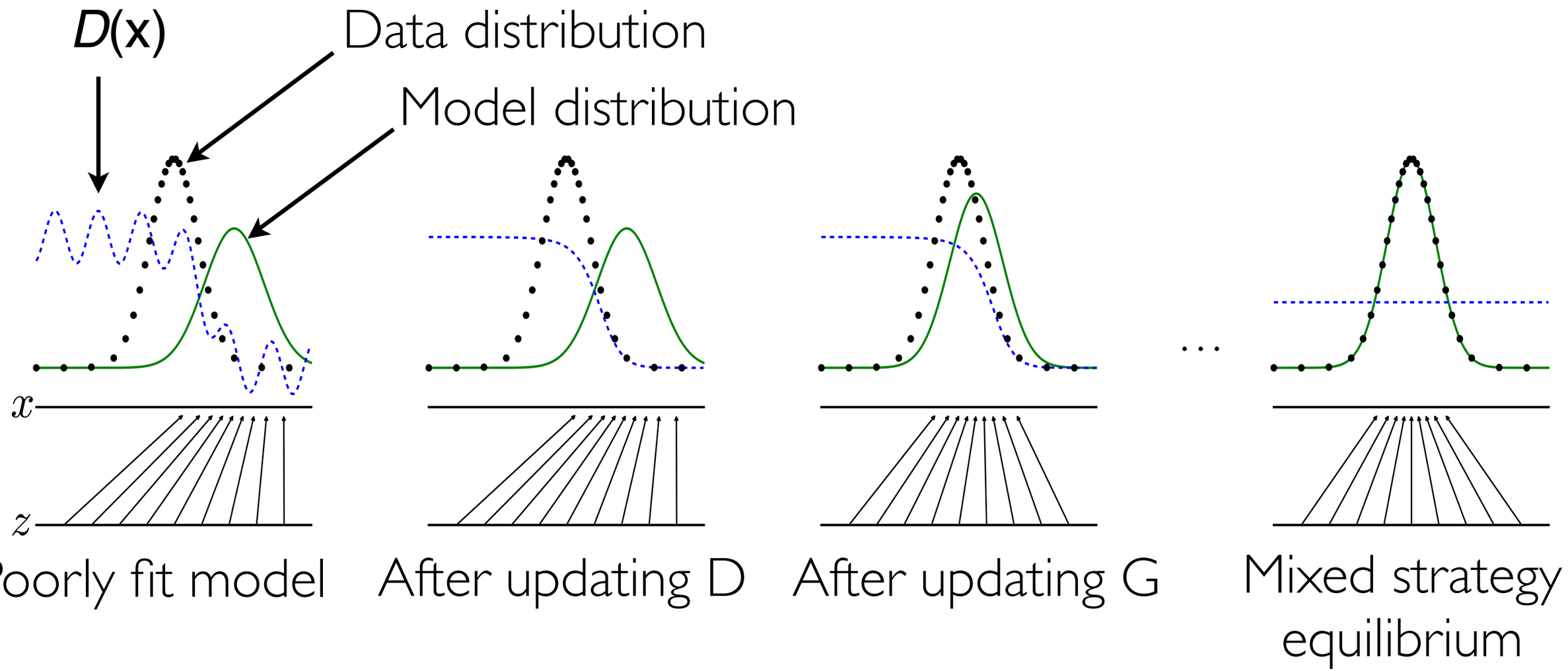
[Slide: Ian Goodfellow, Deep Learning workshop, ICML 2015]

Generative Adversarial Networks



[Slide: Ian Goodfellow, Deep Learning workshop, ICML 2015]

Generative Adversarial Networks

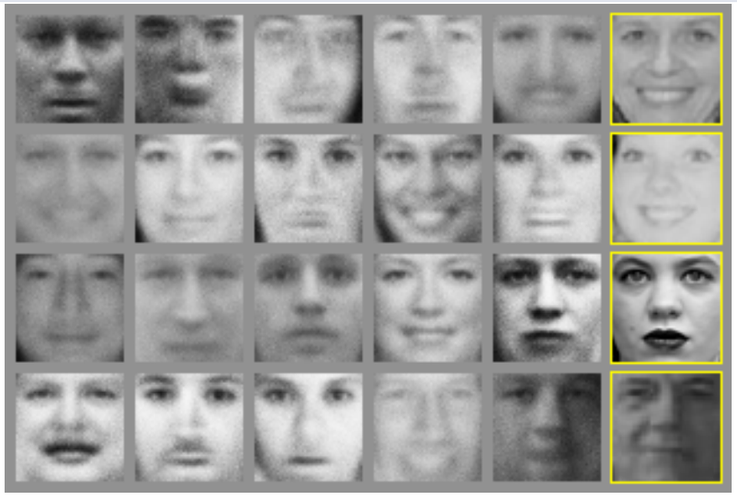


[Slide: Ian Goodfellow, Deep Learning workshop, ICML 2015]

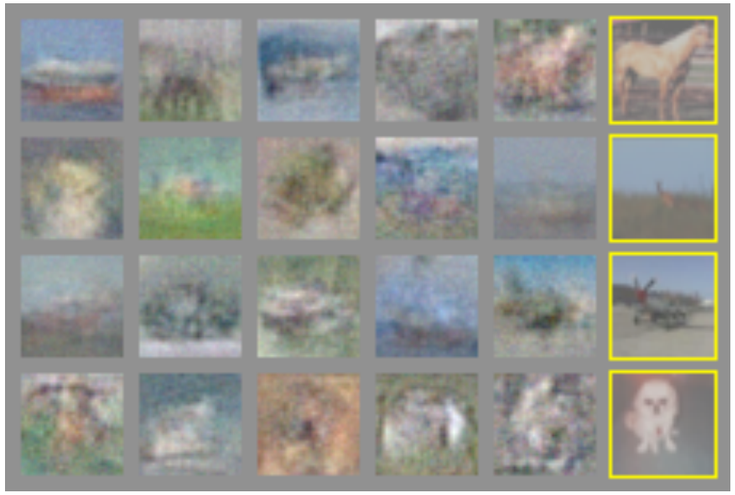
Adversarial Network Samples



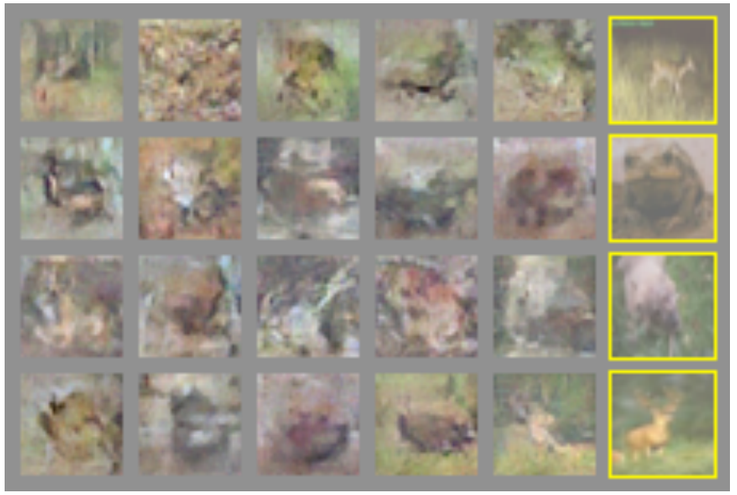
MNIST



TFD



CIFAR-10 (fully connected)



CIFAR-10 (convolutional)

[Slide: Ian Goodfellow, Deep Learning workshop, ICML 2015]

DCGAN

UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS

Alec Radford & Luke Metz
indico Research
Boston, MA
{alec,luke}@indico.io

Soumith Chintala
Facebook AI Research
New York, NY
soumith@fb.com

ICLR 2016

- First to generate plausible results at 64x64.
- Improved architectures for generator/discriminator
- Most GAN architectures used now are similar



- Lots of tricks to get GANs to train well

3.5 Years of Progress on Faces

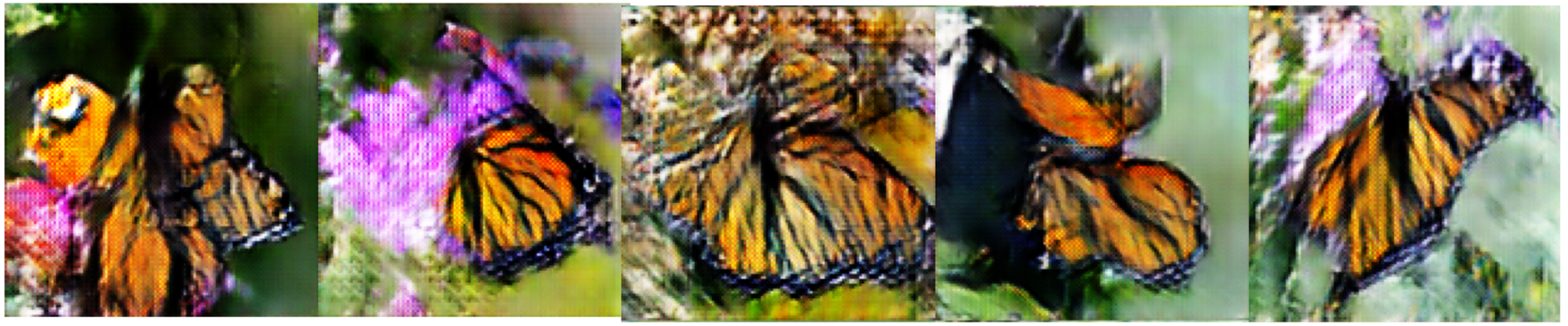


2014 2015 2016 2017

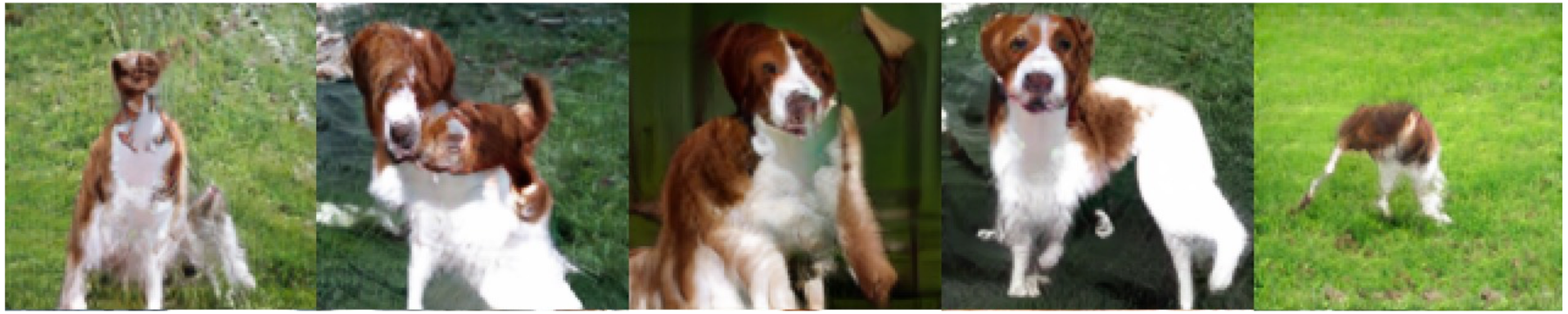
(Brundage et al, 2018)

<2 Years of Progress on ImageNet

Odena et al 2016



Miyato et al 2017



Zhang et al 2018



Evaluation of GANs

- Short answer: hard; look at quality of samples
- Computing log-likelihood not directly possible

  LLH is problematic.

  See [A note on the evaluation of generative models, Lucas Theis, Aäron van den Oord, Matthias Bethge, ICLR 2016]

- Inception score
- User-study: can humans tell fake from real?

Inception Score

Proposed in 2016

Improved Techniques for Training GANs

Tim Salimans tim@openai.com
Ian Goodfellow ian@openai.com
Wojciech Zaremba woj@openai.com
Vicki Cheung vicki@openai.com
Alec Radford alec.radford@gmail.com
Xi Chen peter@openai.com

Abstract

Inception Score

- Send generated image through Inception model (trained on Imagenet)

generated image to get the conditional label distribution $p(y|\boldsymbol{x})$. Images that contain meaningful objects should have a conditional label distribution $p(y|\boldsymbol{x})$ with low entropy. Moreover, we expect the model to generate varied images, so the marginal $\int p(y|\boldsymbol{x} = G(z))dz$ should have high entropy. Combining these two requirements, the metric that we propose is: $\exp(\mathbb{E}_{\boldsymbol{x}}\mathrm{KL}(p(y|\boldsymbol{x})||p(y)))$, where

Inception Score

- Send generated image through Inception model (trained on Imagenet)

generated image to get the conditional label distribution $p(y|\boldsymbol{x})$. Images that contain meaningful objects should have a conditional label distribution $p(y|\boldsymbol{x})$ with low entropy. Moreover, we expect the model to generate varied images, so the marginal $\int p(y|\boldsymbol{x} = G(z))dz$ should have high entropy. Combining these two requirements, the metric that we propose is: $\exp(\mathbb{E}_{\boldsymbol{x}}\mathrm{KL}(p(y|\boldsymbol{x})||p(y)))$, where

How to Train a GAN

Emily Denton, Martin Arjovsky, Michael Mathieu
New York University

Ian Goodfellow
Google

Soumith Chintala
Facebook AI Research

The stability of GANs



Timeline - the stability of GANs



Goodfellow et. al. “Generative Adversarial Networks”

2014

Timeline - the stability of GANs



model architecture generator

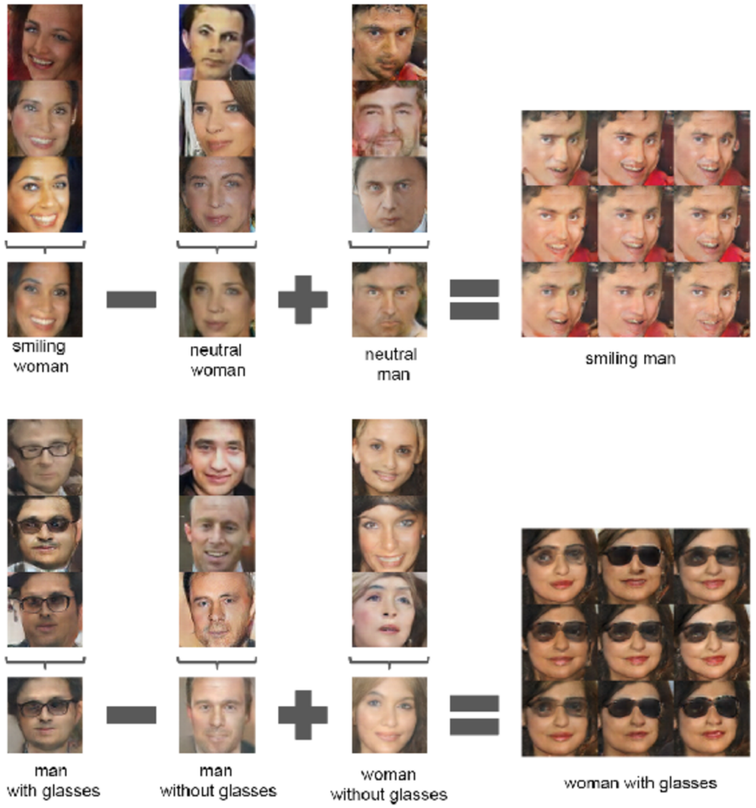
visual inspection

countless failed stability hacks

Denton et. al. “Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks”

2015

Timeline - the stability of GANs



countless hours finding stable models

stable upto 64x64

mode dropping

underfitting

Radford et. al. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”

2015

Timeline - the stability of GANs



more heuristics

more stability

Salimans et. al. “Improved Techniques for Training GANs”

2015

Timeline - the stability of GANs

gradient norm regularization

Least-Squares
Boundary Equilibrium

Gulrajani et. al. "Improved Training of Wasserstein GANs"

Xudong et al. "Least squares generative adversarial networks."

Berthelot et. al. "Began: Boundary equilibrium generative adversarial networks."



2016-2017

Timeline - the stability of GANs

https://github.com/khanrc/tf.gans-comparison
by Junbum Cha

GANs comparison without cherry-picking

Implementations of some theoretical generative adversarial nets: DCGAN, EBGAN, LSGAN, WGAN, WGAN-GP, BEGAN, DRAGAN and CoulombGAN.

I implemented the structure of model equal to the structure in paper and compared it on the CelebA dataset and LSUN dataset without cherry-picking.

Comparison to Classification ConvNets

- Throw things at the wall and see what sticks
- Intuition is poorer
- Theoretical work is somewhat improving but still far away
- Objective validation metrics are not there yet

#1: Normalize the inputs

- normalize the images between -1 and 1
- Tanh as the last layer of the generator output
  - or some kind of bounds normalization

#2: Modified loss function (classic GAN)

- In papers people write min (log 1-D), but in practice folks practically use max log D
  - because the first formulation has vanishing gradients early on
  - Goodfellow et. al (2014)
- In practice:
  - Flip labels when training generator: real = fake, fake = real

#2: Modified loss function (classic GAN)

- In papers people write min (log 1-D), but in practice folks practically use max log D
  - because the first formulation has vanishing gradients early on
  - Goodfellow et. al (2014)
- In practice:
  - Flip labels when training generator: real = fake, fake = real

LOT OF NEW LOSS FORMULATIONS

#2: Modified loss function (classic GAN)

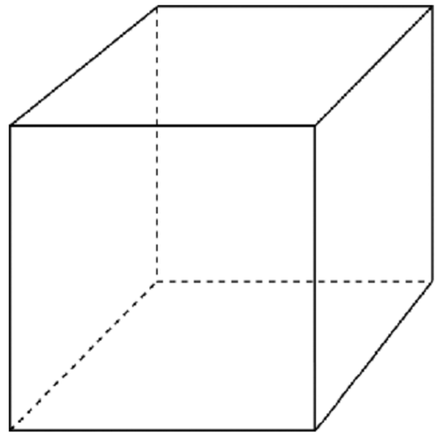
https://github.com/hwalsuklee/tensorflow-generative-model-collections

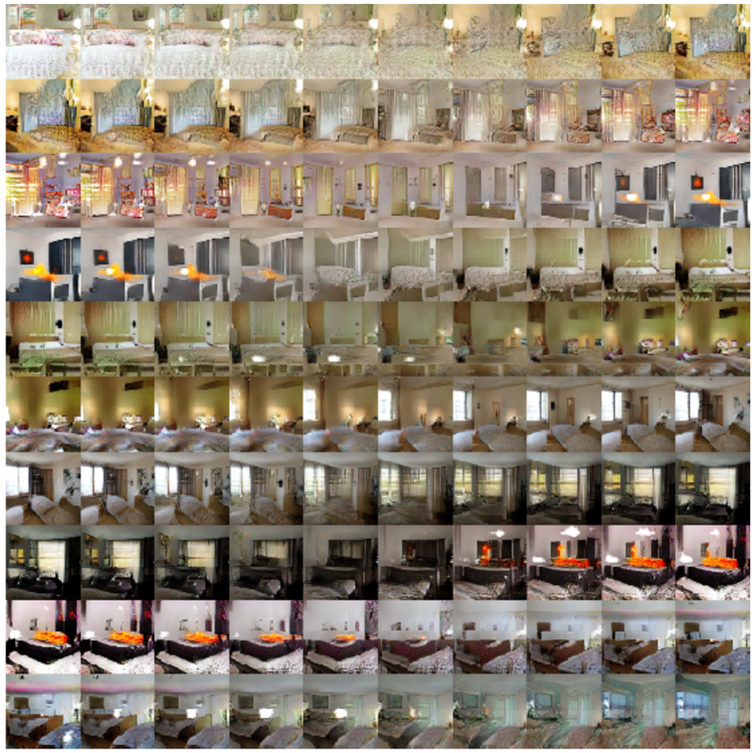
https://github.com/znxlwm/pytorch-generative-model-collections

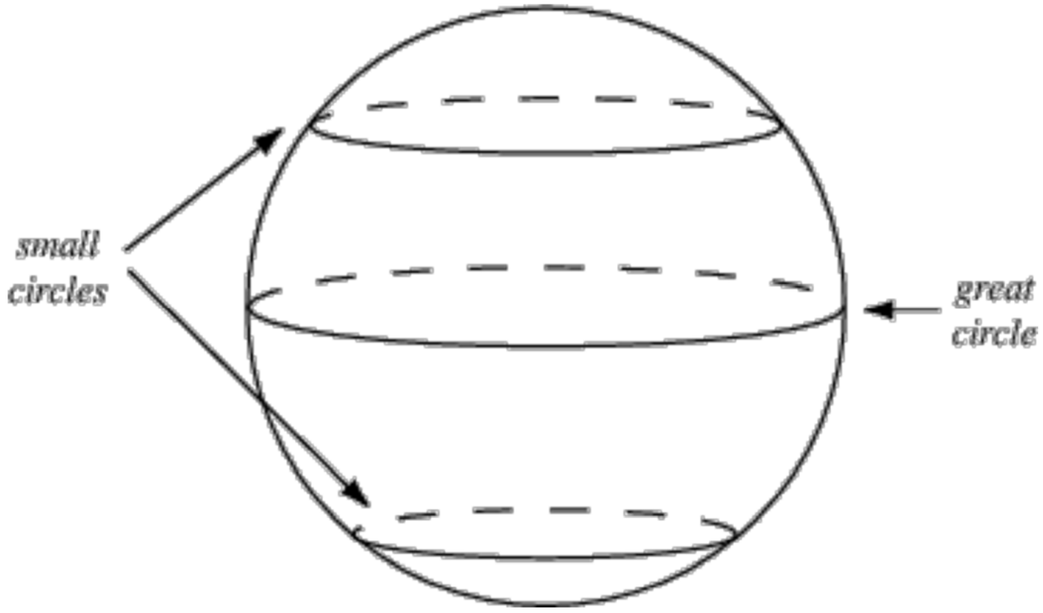
Name	Paper Link	Value Function
GAN	Arxiv	$L_D^{GAN} = E\left[\log\left(D(x)\right)\right] + E\left[\log\left(1 - D(G(z))\right)\right]$ $L_G^{GAN} = E\left[\log\left(D(G(z))\right)\right]$
LSGAN	Arxiv	$L_D^{LSGAN} = E[(D(x) - 1)^2] + E[D(G(z))^2]$ $L_G^{LSGAN} = E[(D(G(z)) - 1)^2]$
WGAN	Arxiv	$L_D^{WGAN} = E[D(x)] - E[D(G(z))]$ $L_G^{WGAN} = E[D(G(z))]$ $W_D \leftarrow clip_by_value(W_D, -0.01, 0.01)$
WGAN-GP	Arxiv	$L_D^{WGAN_GP} = L_D^{WGAN} + \lambda E[(\lvert\nabla D(\alpha x - (1 - \alpha G(z)))\rvert - 1)^2]$ $L_G^{WGAN_GP} = L_G^{WGAN}$
DRAGAN	Arxiv	$L_D^{DRAGAN} = L_D^{GAN} + \lambda E[\left(\lvert\nabla D(\alpha x - (1 - \alpha x_p))\rvert - 1\right)^2]$ $L_G^{DRAGAN} = L_G^{GAN}$
CGAN	Arxiv	$L_D^{CGAN} = E\left[\log\left(D(x, c)\right)\right] + E\left[\log\left(1 - D(G(z), c)\right)\right]$ $L_G^{CGAN} = E\left[\log\left(D(G(z), c)\right)\right]$
infoGAN	Arxiv	$L_{D,Q}^{infoGAN} = L_D^{GAN} - \lambda L_I(c, c')$ $L_G^{infoGAN} = L_G^{GAN} - \lambda L_I(c, c')$
ACGAN	Arxiv	$L_{D,Q}^{ACGAN} = L_D^{GAN} + E[P(class = c\lvert x)] + E[P(class = c\lvert G(z))]$ $L_G^{ACGAN} = L_G^{GAN} + E[P(class = c\lvert G(z))]$
EBGAN	Arxiv	$L_D^{EBGAN} = D_{AE}(x) + \max(0, m - D_{AE}(G(z)))$ $L_G^{EBGAN} = D_{AE}(G(z)) + \lambda \cdot PT$
BEGAN	Arxiv	$L_D^{BEGAN} = D_{AE}(x) - k_t D_{AE}(G(z))$ $L_G^{BEGAN} = D_{AE}(G(z))$ $k_{t+1} = k_t + \lambda(\gamma D_{AE}(x) - D_{AE}(G(z)))$

#3: Use spherical z

- interpolation via great circle
- Tom White “Sampling Generative Networks”
  - https://arxiv.org/abs/1609.04468

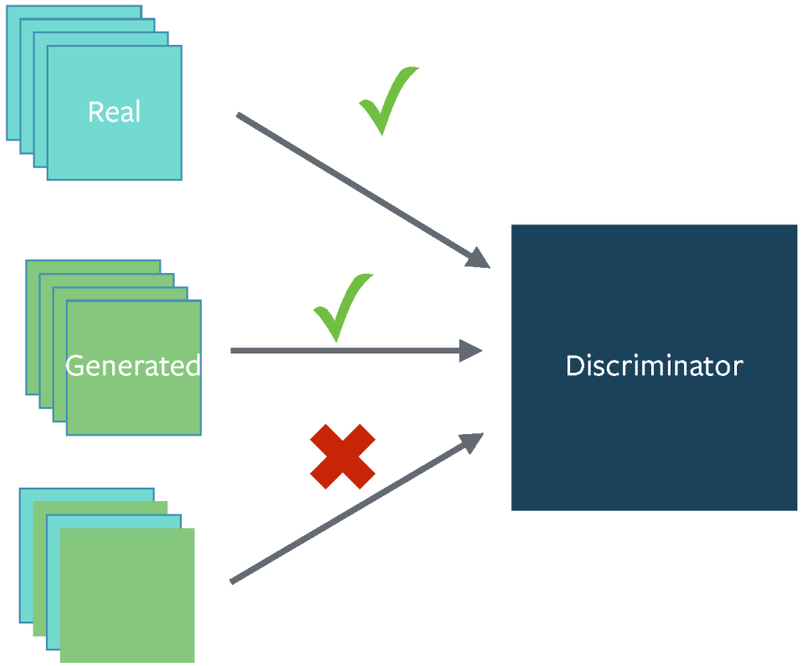






#4: BatchNorm

- different batches for real and fake
- when batchnorm is not an option use instance norm



#5: Avoid Sparse Gradients: ReLU, MaxPool

- the stability of the GAN game suffers
- LeakyReLU (both G and D)
- Downsampling: Average Pooling, Conv2d + stride
- Upsampling: PixelShuffle, ConvTranspose2d + stride
  - PixelShuffle: https://arxiv.org/abs/1609.05158

#6: Soft and Noisy Labels

- Label Smoothing
- making the labels the noisy a bit for the discriminator, sometimes

-Salimans et. al. 2016

#7: Architectures: DCGANs / Hybrids

- DCGAN when you can

- if you cant use DCGANs and no model is stable,
- use a hybrid model : KL + GAN or VAE + GAN

- ResNets from WGAN-gp also work pretty well (but are very slow)
  - https://github.com/igul222/improved_wgan_training

- Width matters more than Depth

#8: Stability tricks from RL

- Experience replay
- Things that work for deep deterministic policy gradients
- See Pfau & Vinyals (2016)

#9: Optimizer: ADAM

- optim.Adam rules!
  - See Radford et. al. 2015
- [MMathieu] Use SGD for discriminator and ADAM for generator

#10: Use Gradient Penalty

- Regularize the norm of the gradients
  - multiple theories on why this is useful (WGAN-GP, DRAGAN, Stabilizing GANs by Regularization etc.)

#11: Dont balance via loss statistics (classic GAN)

- Dont try to find a (number of G / number of D) schedule to uncollapse training

- while lossD > X:
  - train D
- while lossG > X:
  - train G

#12: If you have labels, use them

- if you have labels available, training the discriminator to also classify the samples: auxillary GANs

#13: Add noise to inputs, decay over time

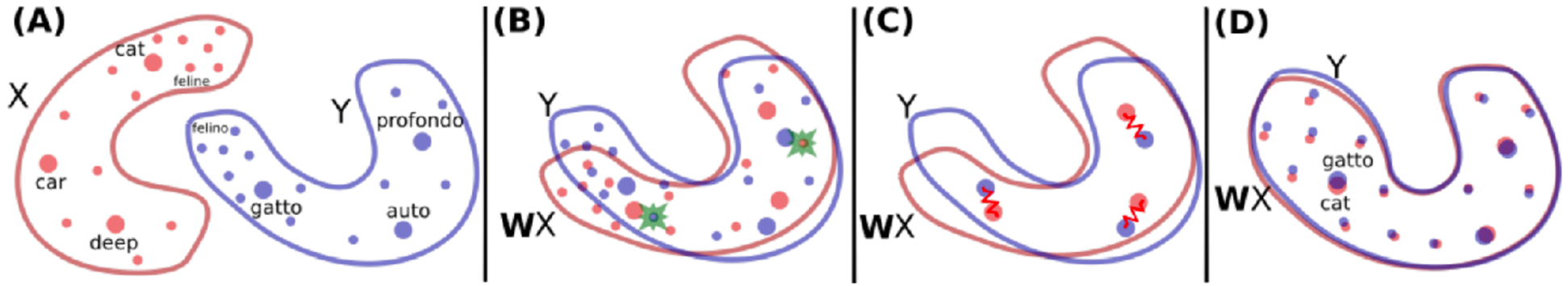
- Add some artificial noise to inputs to D (Arjovsky et. al., Huszar, 2016)
  - http://www.inference.vc/instance-noise-a-trick-for-stabilising-gan-training/
  - https://openreview.net/forum?id=Hk4_qw5xe
- adding gaussian noise to every layer of generator (Zhao et. al. EBGAN)
- Improved GANs: OpenAI code also has it (commented out)

#14: Train discriminator more

- especially when you have noise
- hard to find a schedule of number of D iterations vs G iterations
- WGAN/WGAN-gp papers suggest 5x D iterations per G iteration

#15: Avoid discrete spaces

- Pose the generation as a continuous prediction

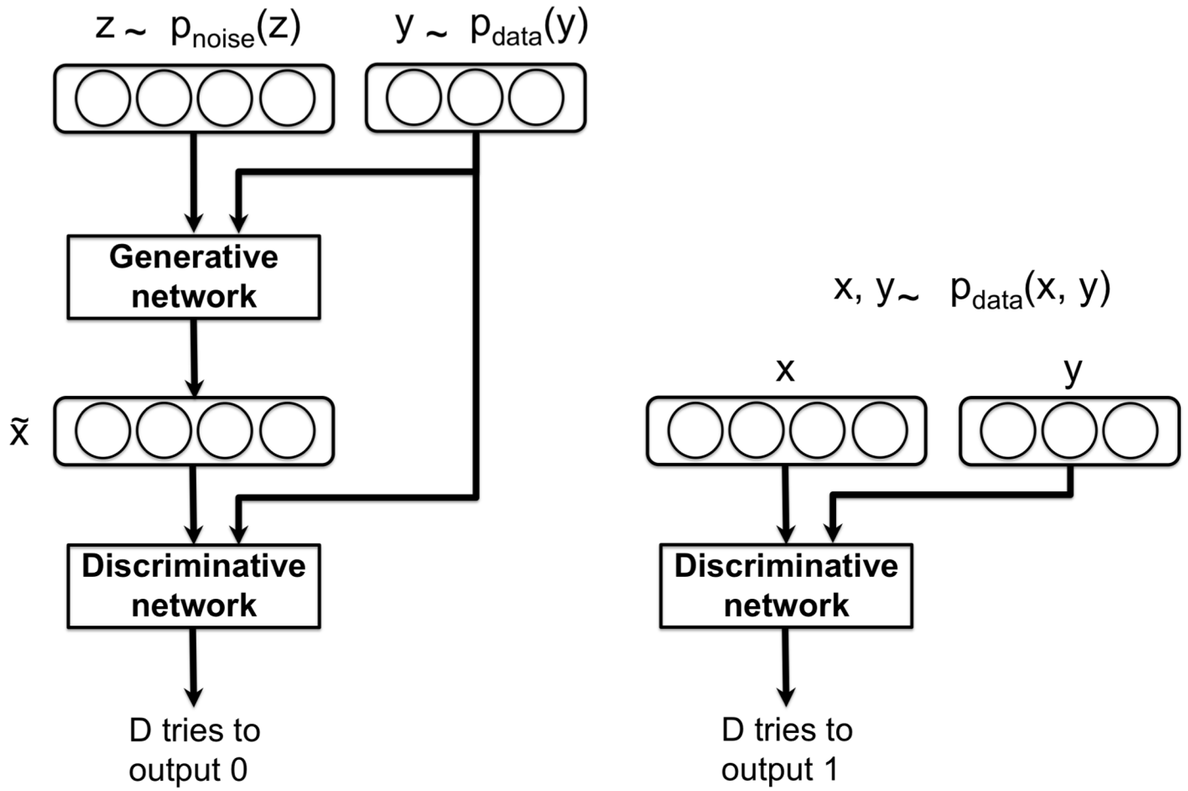


Conneau, Lample et. al. "Word Translation Without Parallel Data"

Conditional GANs

Conditional generative adversarial networks (CGAN)

- Condition generation on additional info **y** (e.g. class label, another image)
- D has to determine if samples are realistic given **y**



[Mirza and Osindero (2014); Gauthier (2014)]

#16: Discrete Variables

- Use an Embedding layer
- Add as additional channels to images
- Keep embedding dimensionality low and upsample to match image channel size

Conclusion

- Model stability is improving
- Theory is improving
- Hacks are a stop-gap

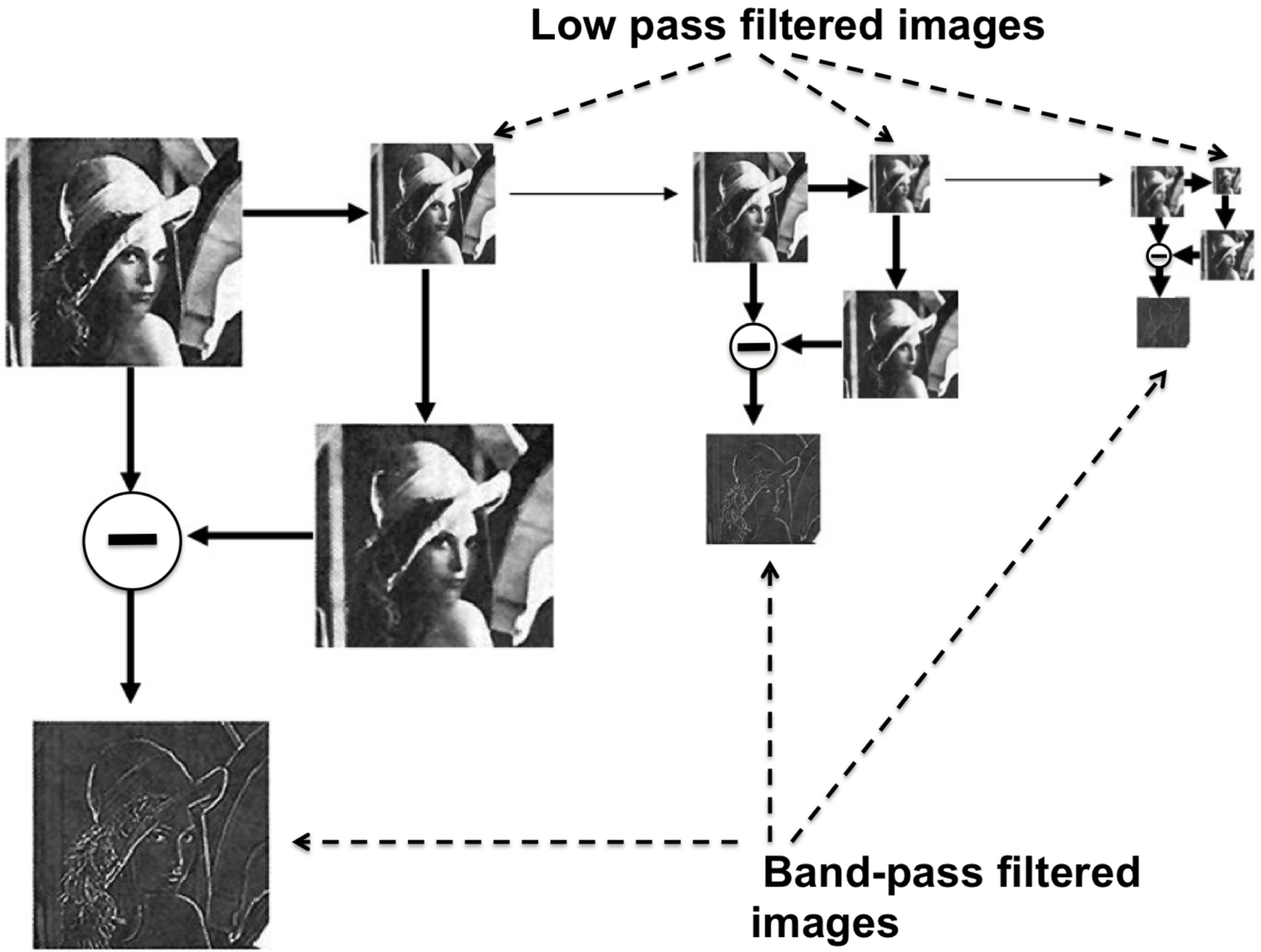
Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

Emily Denton[1*], Soumith Chintala[2*],
Arthur Szlam[2], Rob Fergus[2]

[1]New York University
[2]Facebook AI Research
[*]Denotes equal contribution

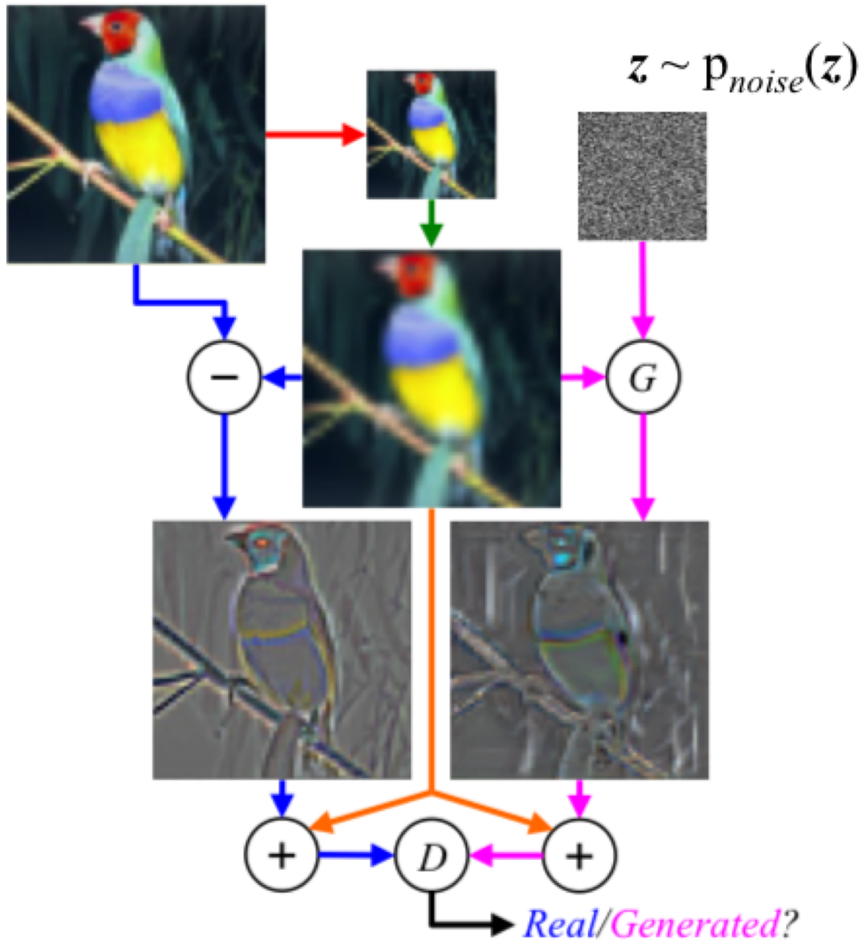
December 16, 2015

Laplacian pyramid (Burt & Adelson, 1983)



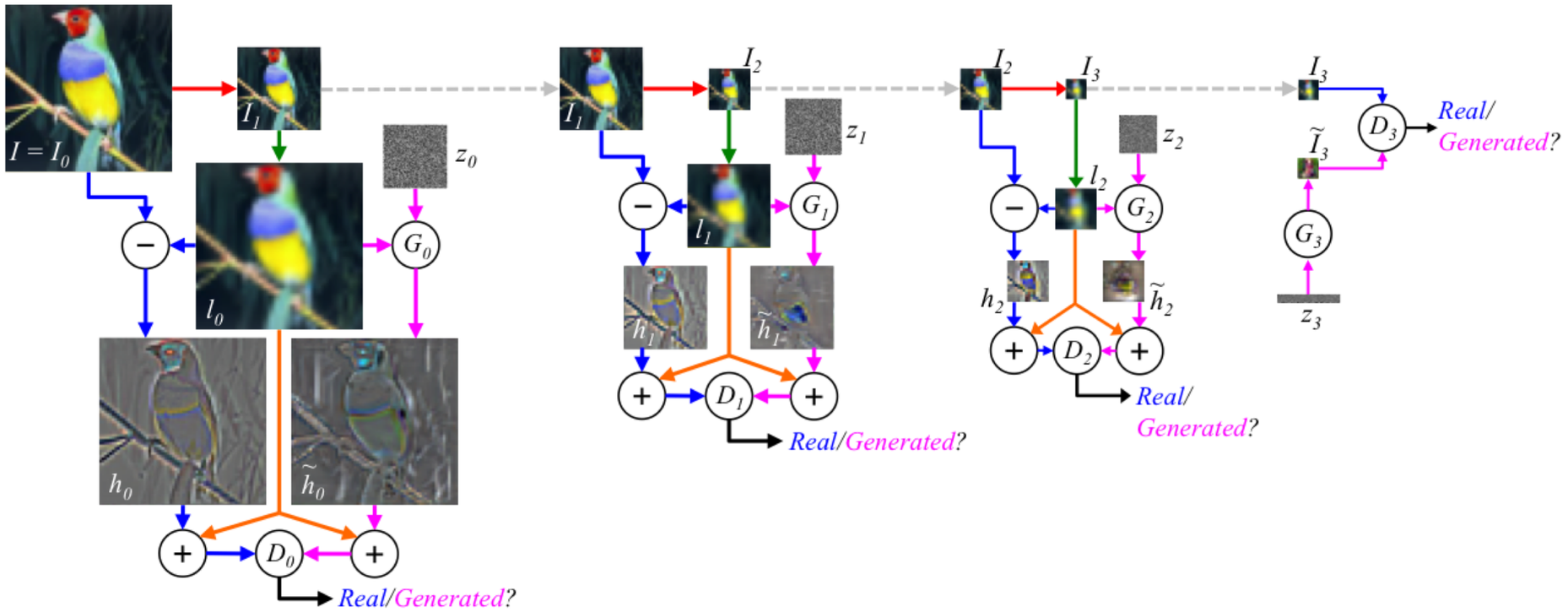
Training procedure

- Train conditional GAN for each level of Laplacian pyramid
- G learns to generate high frequency structure consistent with low frequency image

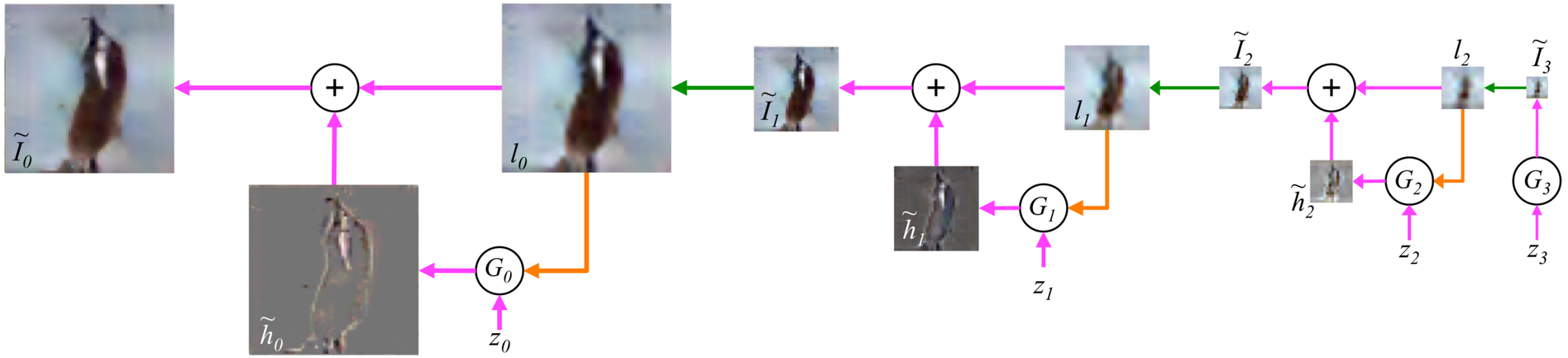


Training procedure

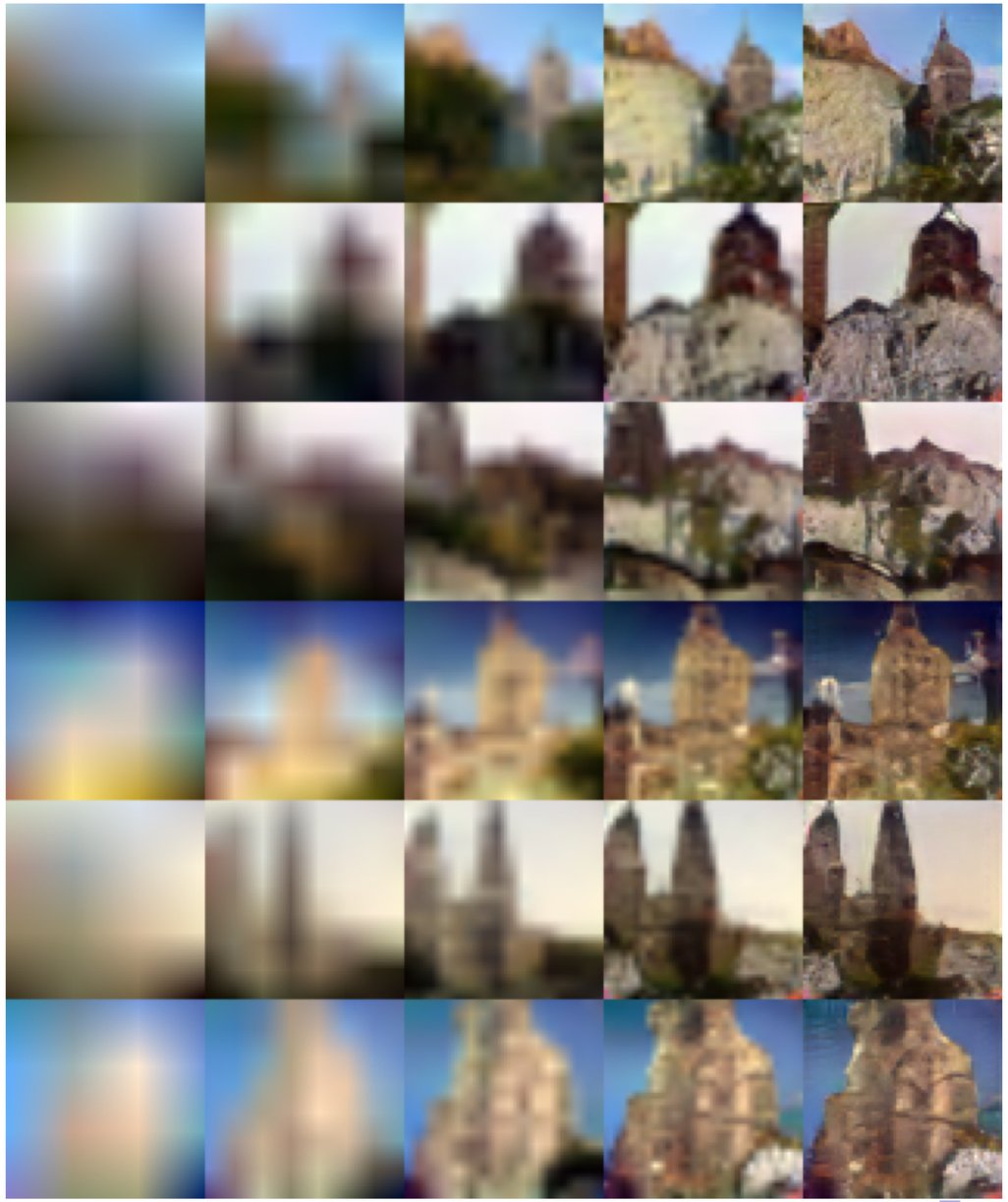
Each level of Laplacian pyramid trained independently



Sampling procedure



LSUN coarse-to-fine chain



LSUN church samples



PROGRESSIVE GROWING OF GANS FOR IMPROVED QUALITY, STABILITY, AND VARIATION

Tero Karras NVIDIA
Timo Aila NVIDIA
Samuli Laine NVIDIA
Jaakko Lehtinen NVIDIA and Aalto University
{tkarras,taila,slaine,jlehtinen}@nvidia.com

ICLR 2018

(Probably) current state-of-art generations

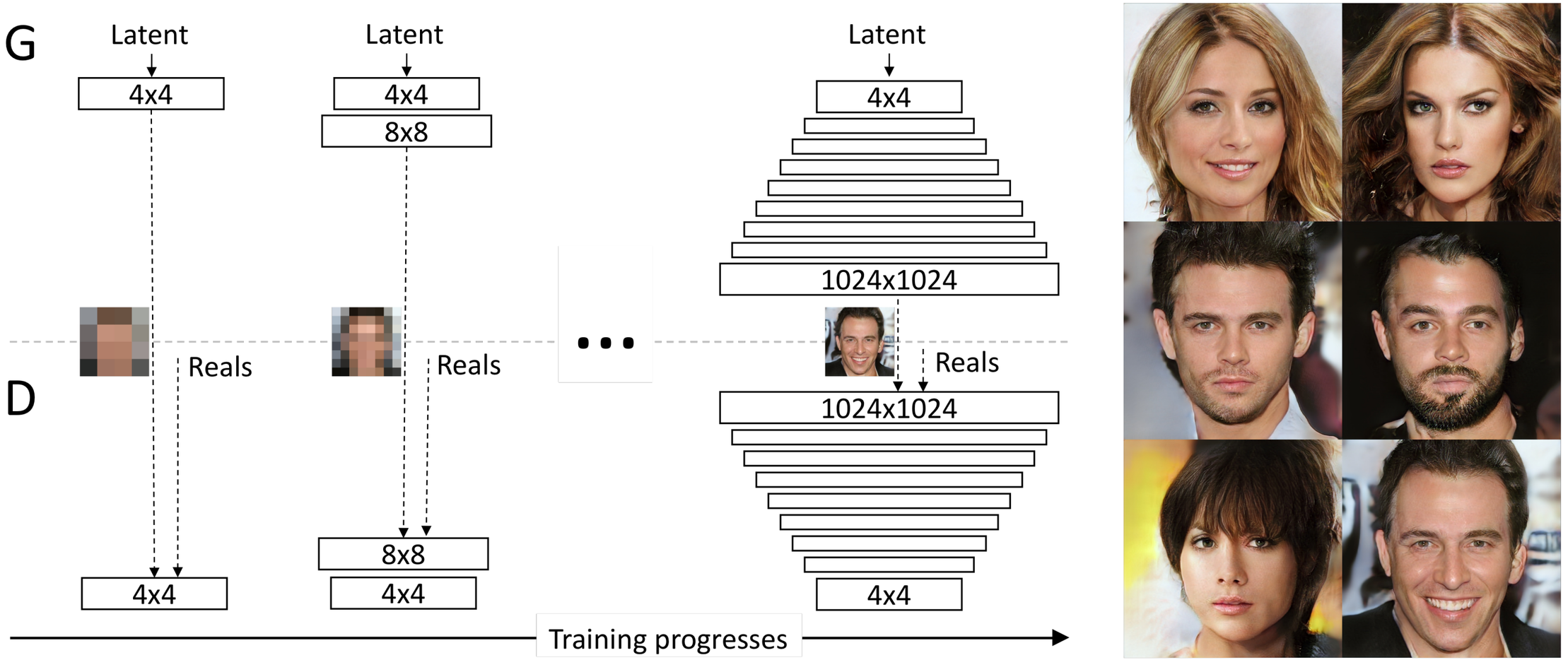
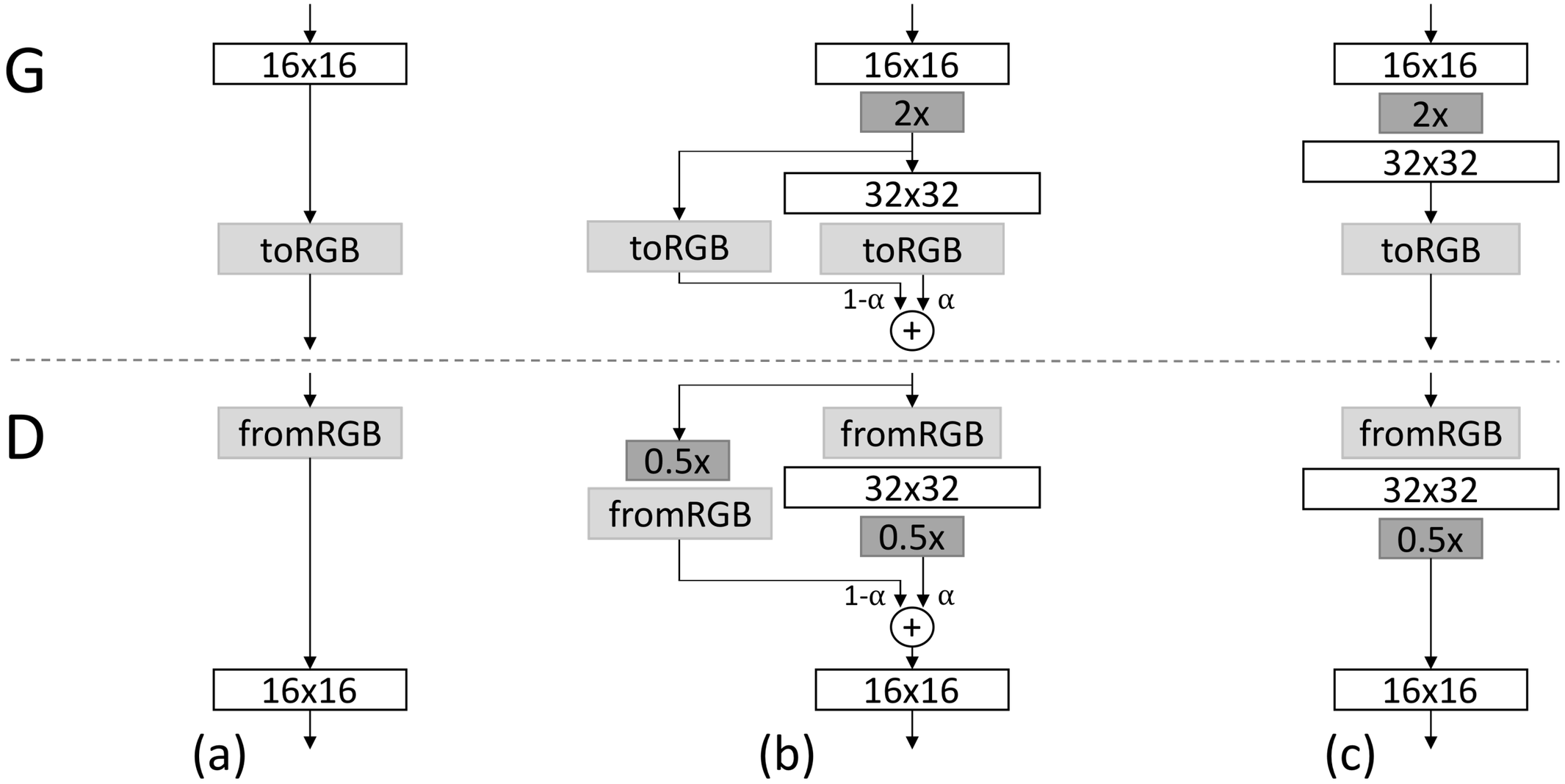


Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout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

Smooth blending with scale increase



Training configuration	CELEBA						LSUN BEDROOM					
	Sliced Wasserstein distance $\times 10^3$					MS-SSIM	Sliced Wasserstein distance $\times 10^3$					MS-SSIM
	128	64	32	16	Avg		128	64	32	16	Avg	
(a) Gulrajani et al. (2017)	12.99	7.79	7.62	8.73	9.28	0.2854	11.97	10.51	8.03	14.48	11.25	**0.0587**
(b) + Progressive growing	4.62	**2.64**	3.78	6.06	4.28	**0.2838**	7.09	6.27	7.40	9.64	7.60	0.0615
(c) + Small minibatch	75.42	41.33	41.62	26.57	46.23	0.4065	72.73	40.16	42.75	42.46	49.52	0.1061
(d) + Revised training parameters	9.20	6.53	4.71	11.84	8.07	0.3027	7.39	5.51	3.65	9.63	6.54	0.0662
(e*) + Minibatch discrimination	10.76	6.28	6.04	16.29	9.84	0.3057	10.29	6.22	5.32	11.88	8.43	0.0648
(e) Minibatch stddev	13.94	5.67	2.82	5.71	7.04	0.2950	7.77	5.23	3.27	9.64	6.48	0.0671
(f) + Equalized learning rate	4.42	3.28	2.32	7.52	4.39	0.2902	**3.61**	3.32	**2.71**	6.44	4.02	0.0668
(g) + Pixelwise normalization	**4.06**	3.04	**2.02**	**5.13**	**3.56**	0.2845	3.89	**3.05**	3.24	**5.87**	**4.01**	0.0640
(h) Converged	2.42	2.17	2.24	4.99	2.96	0.2828	3.47	2.60	2.30	4.87	3.31	0.0636

Table 1: Sliced Wasserstein distance (SWD) between the generated and training images (Section 5) and multi-scale structural similarity (MS-SSIM) among the generated images for several training setups at 128×128. For SWD, each column represents one level of the Laplacian pyramid, and the last one gives an average of the four distances.

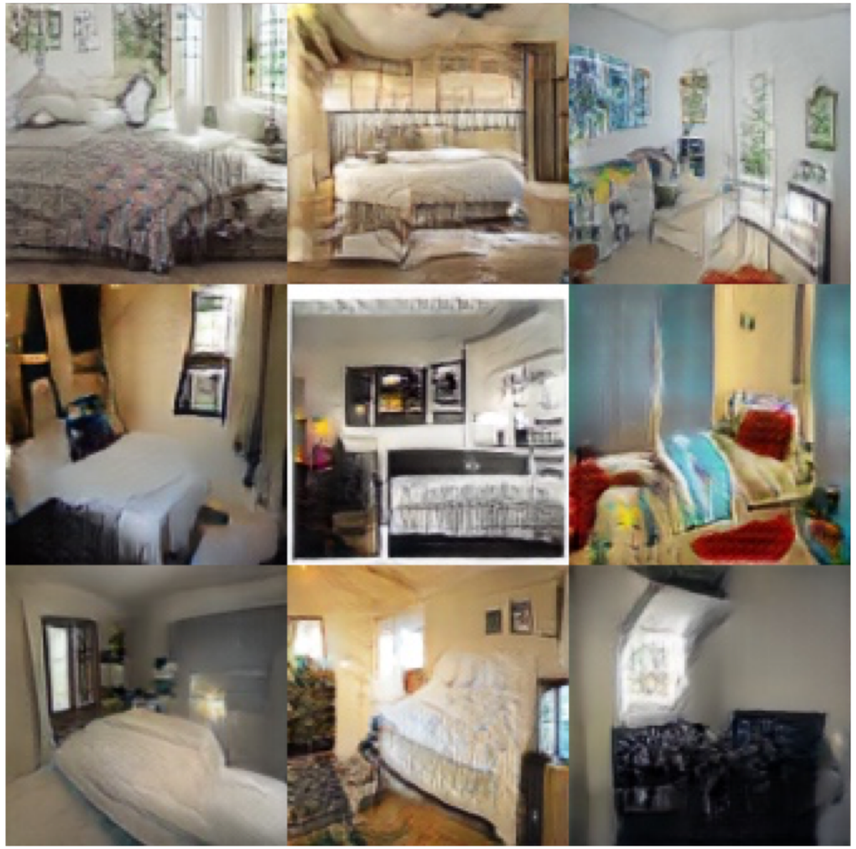


(a) (b) (c) (d) (e∗) (e) (f) (g) (h) Converged

Figure 3: (a) – (g) CELEBA examples corresponding to rows in Table 1. These are intentionally non-converged. (h) Our converged result. Notice that some images show aliasing and some are not sharp – this is a flaw of the dataset, which the model learns to replicate faithfully.



Figure 5: 1024 × 1024 images generated using the CELEBA-HQ dataset. See Appendix F for a



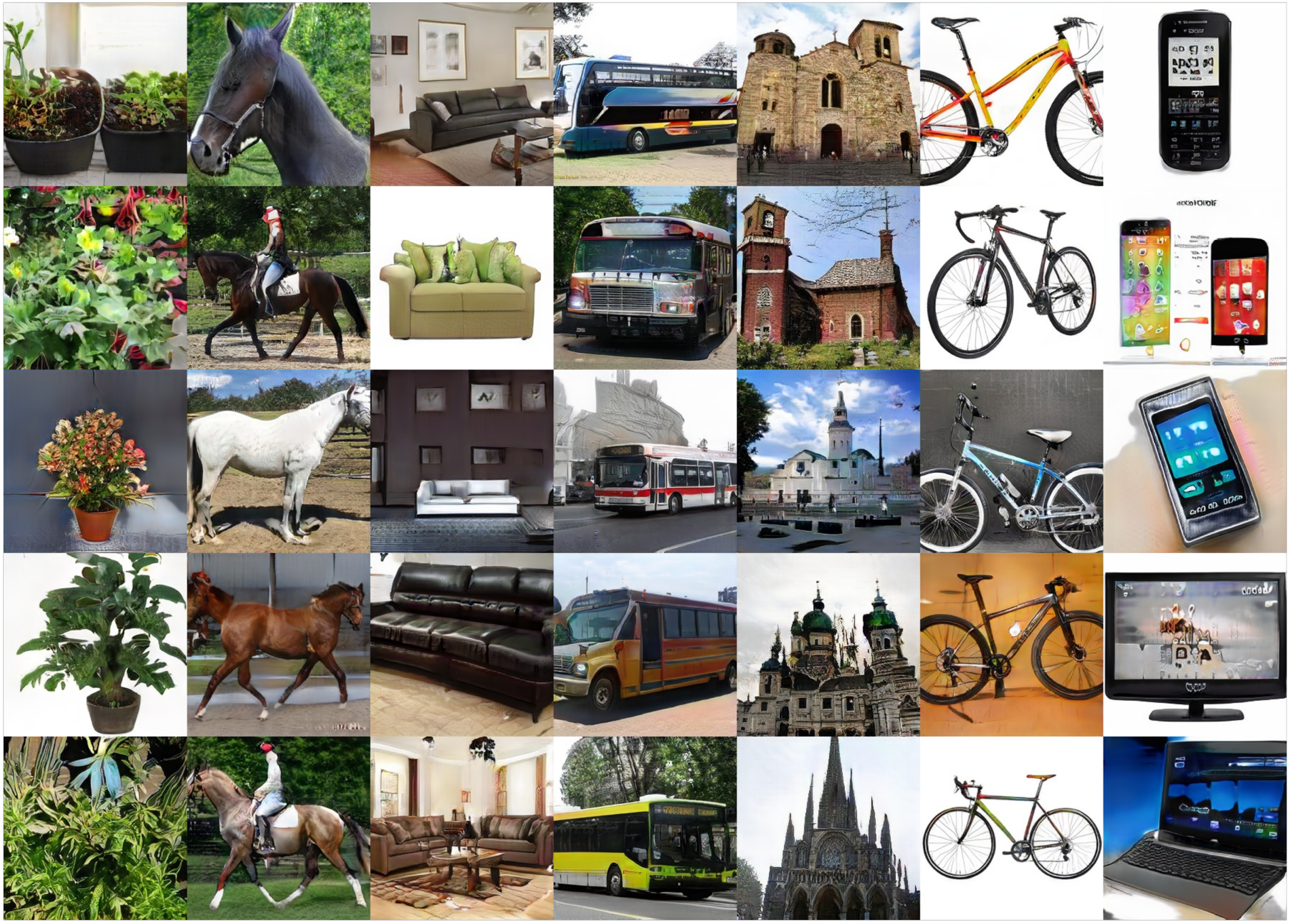
Mao et al. (2016b) (128 × 128)



Gulrajani et al. (2017) (128 × 128)



Our (256 × 256)



POTTEDPLANT HORSE SOFA BUS CHURCHOUTDOOR BICYCLE TVMONITOR

Nearest-Neighbor Sanity Check



Unpaired Image-to-Image Translation with CycleGAN

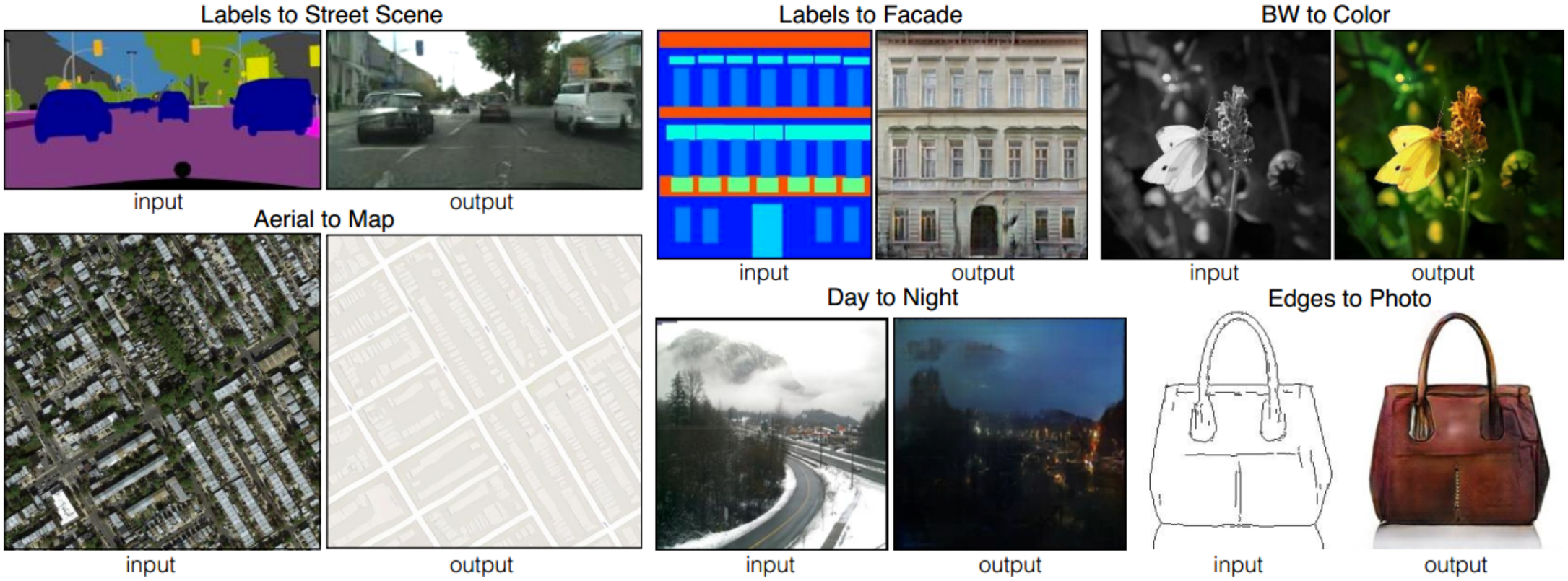
Jun-Yan Zhu and Taesung Park
Joint work with Phillip Isola and Alexei A. Efros

BAIR
BERKELEY ARTIFICIAL INTELLIGENCE RESEARCH

CSAIL



Image-to-Image Translation with **pix2pix**



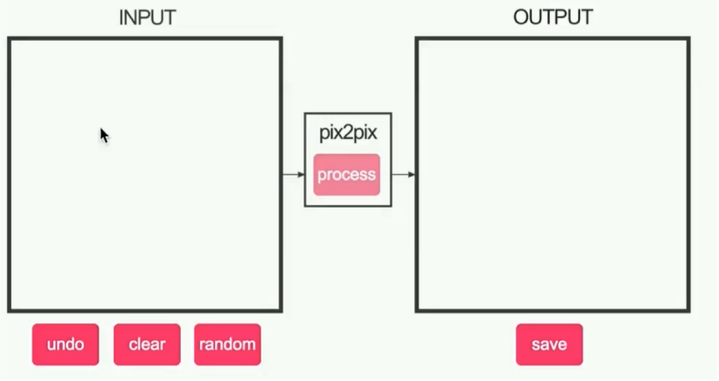
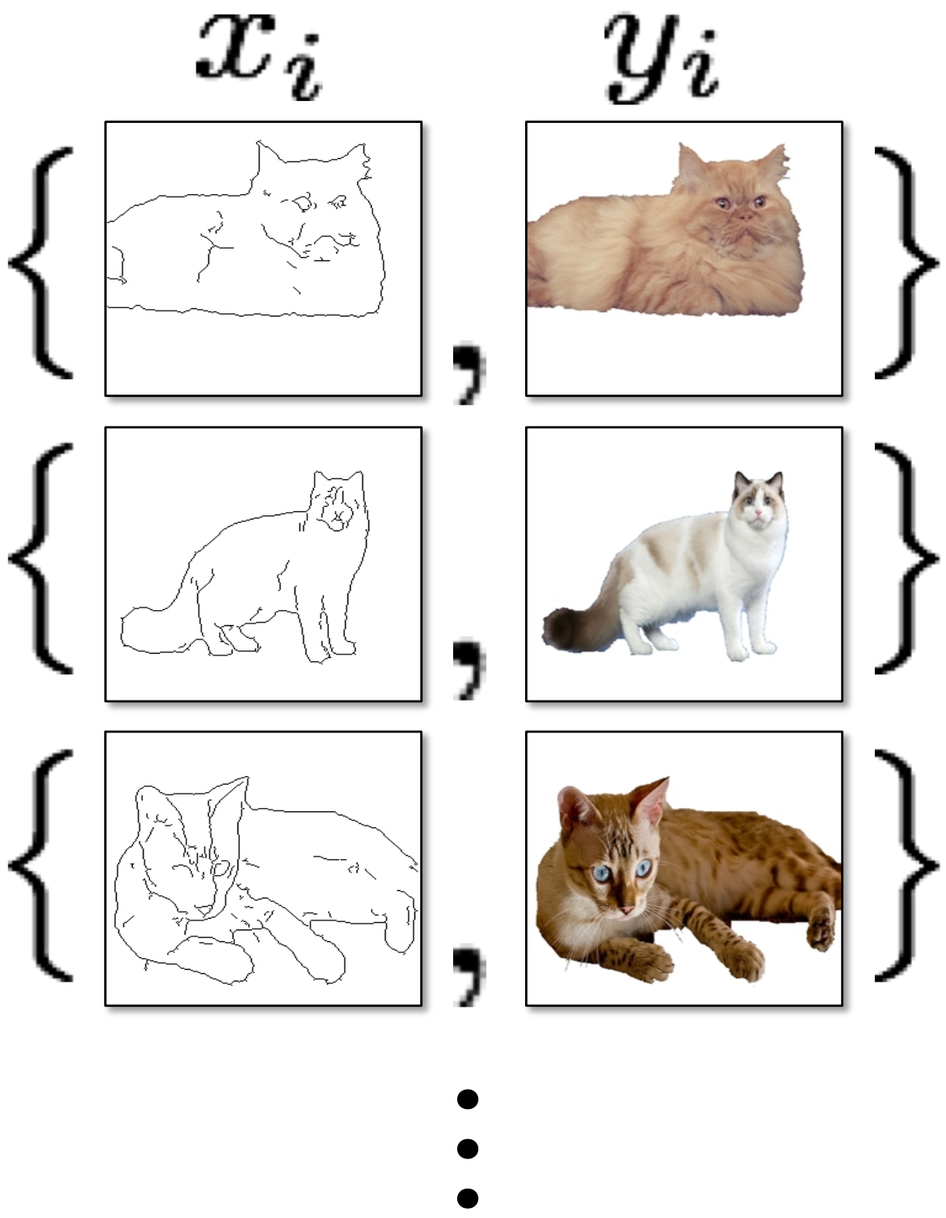
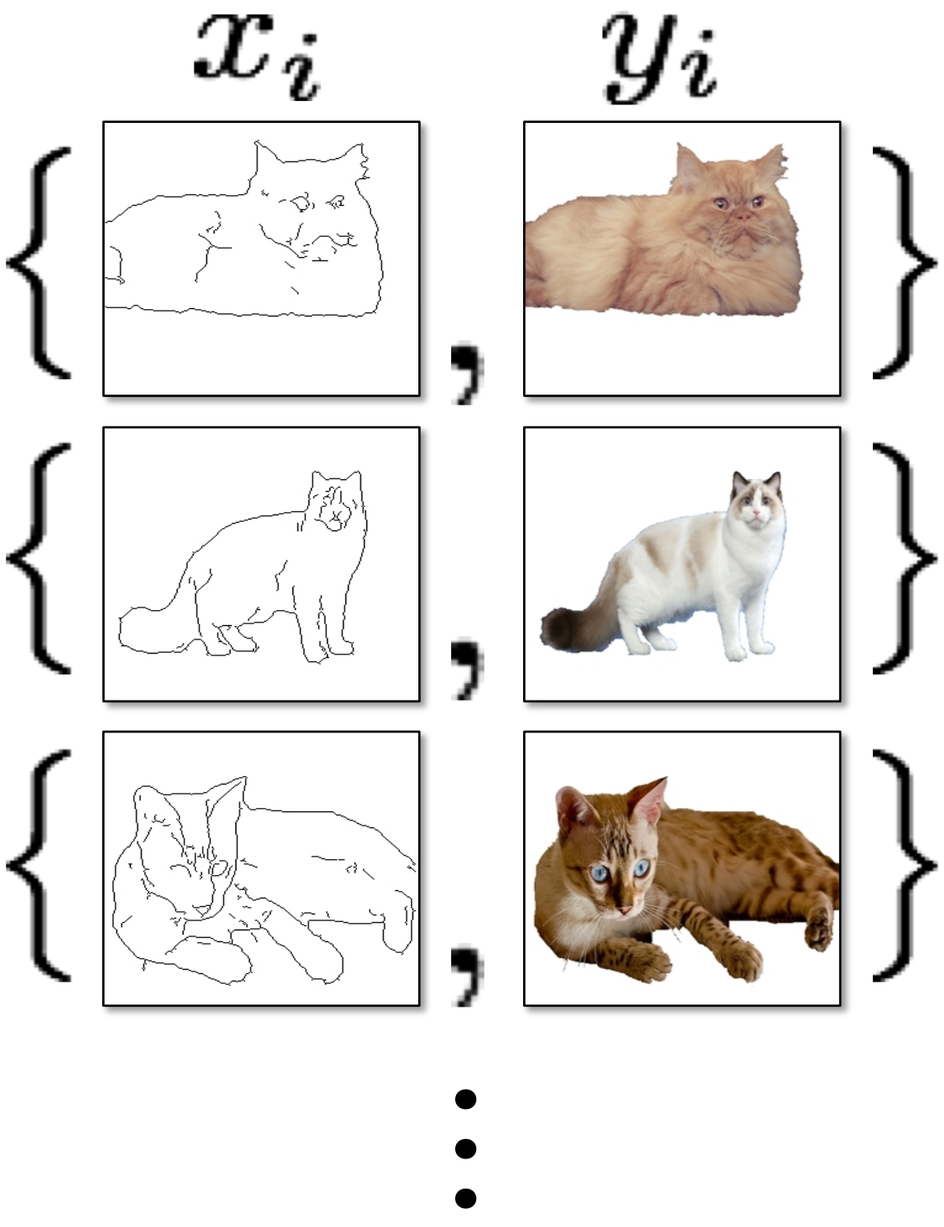


Image-to-image Translation with Conditional Adversarial Nets
Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros. CVPR 2017

Paired



Paired







Label ↔ photo: per-pixel labeling

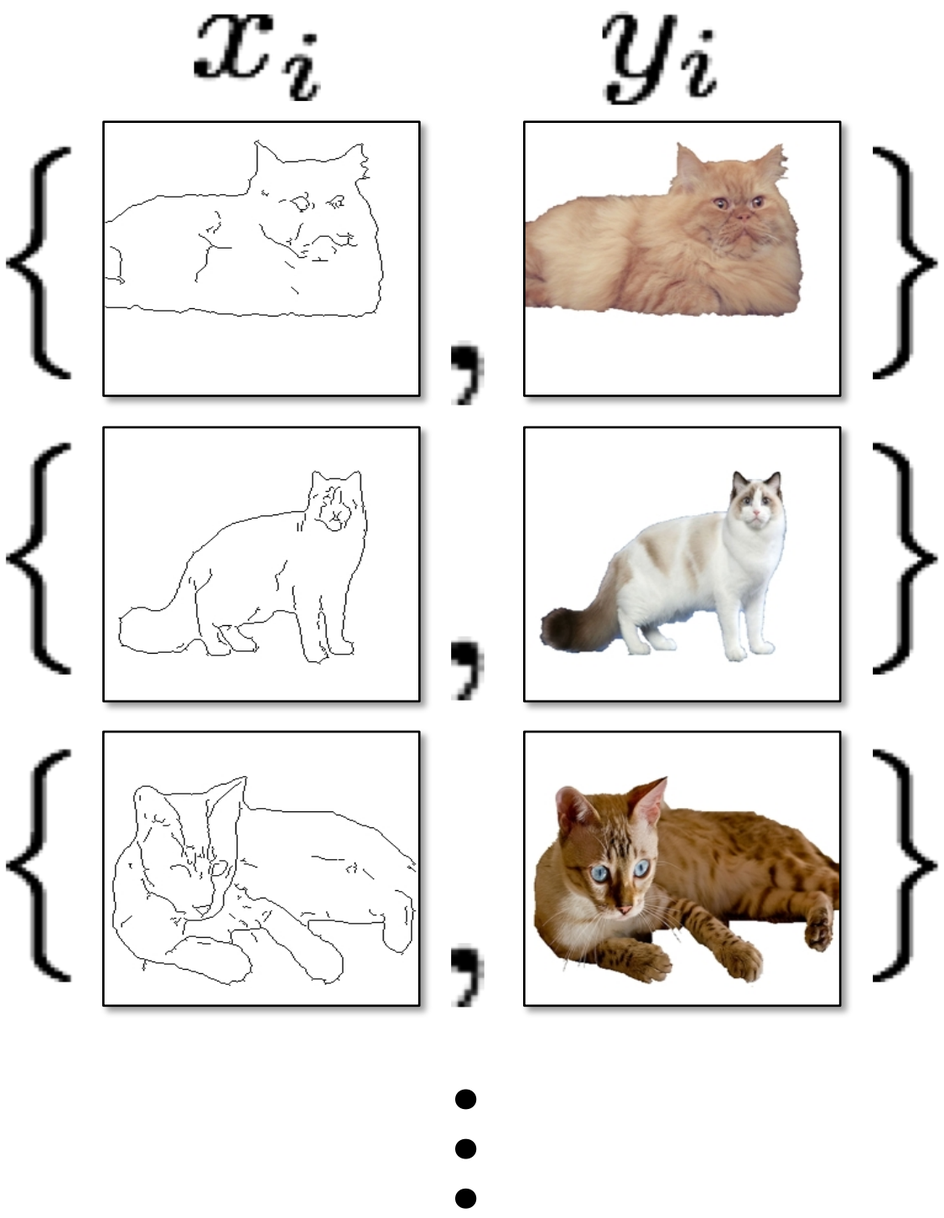




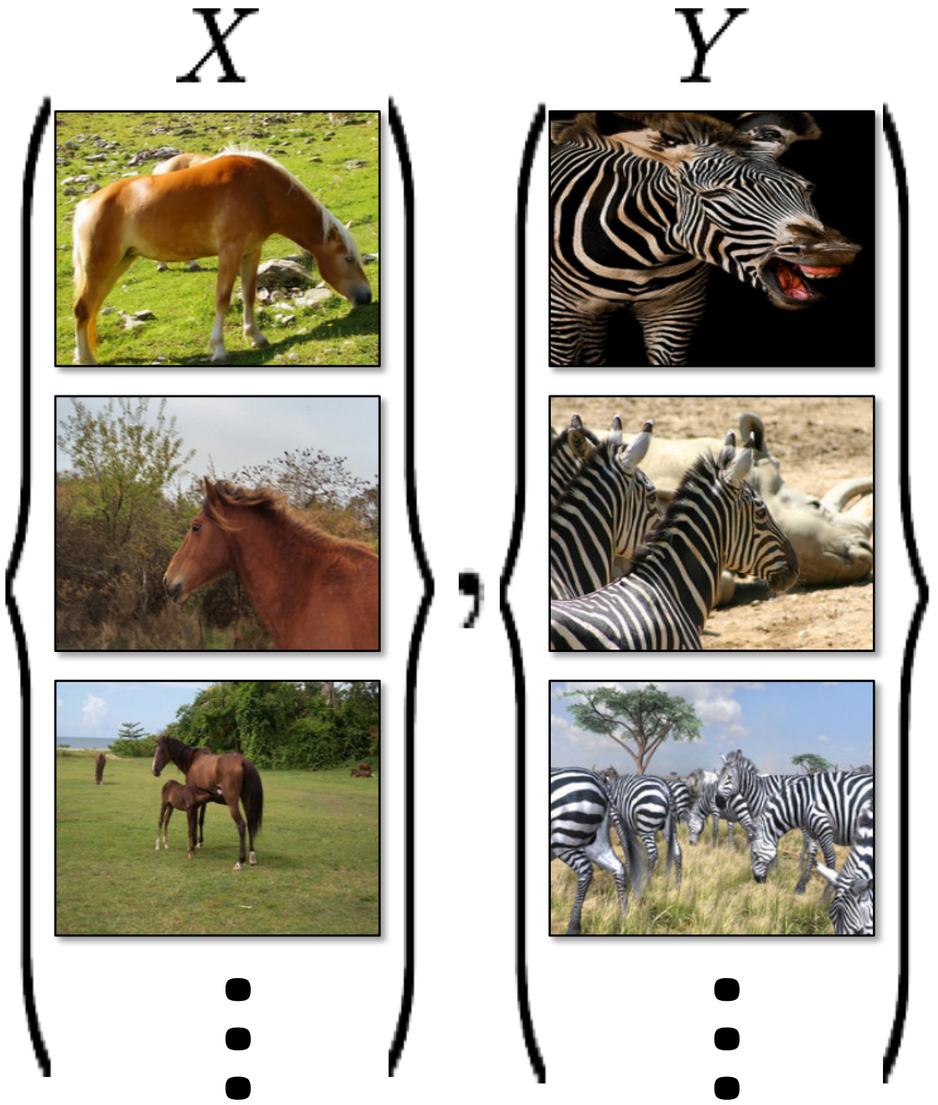
Horse ↔ zebra: how to get zebras?

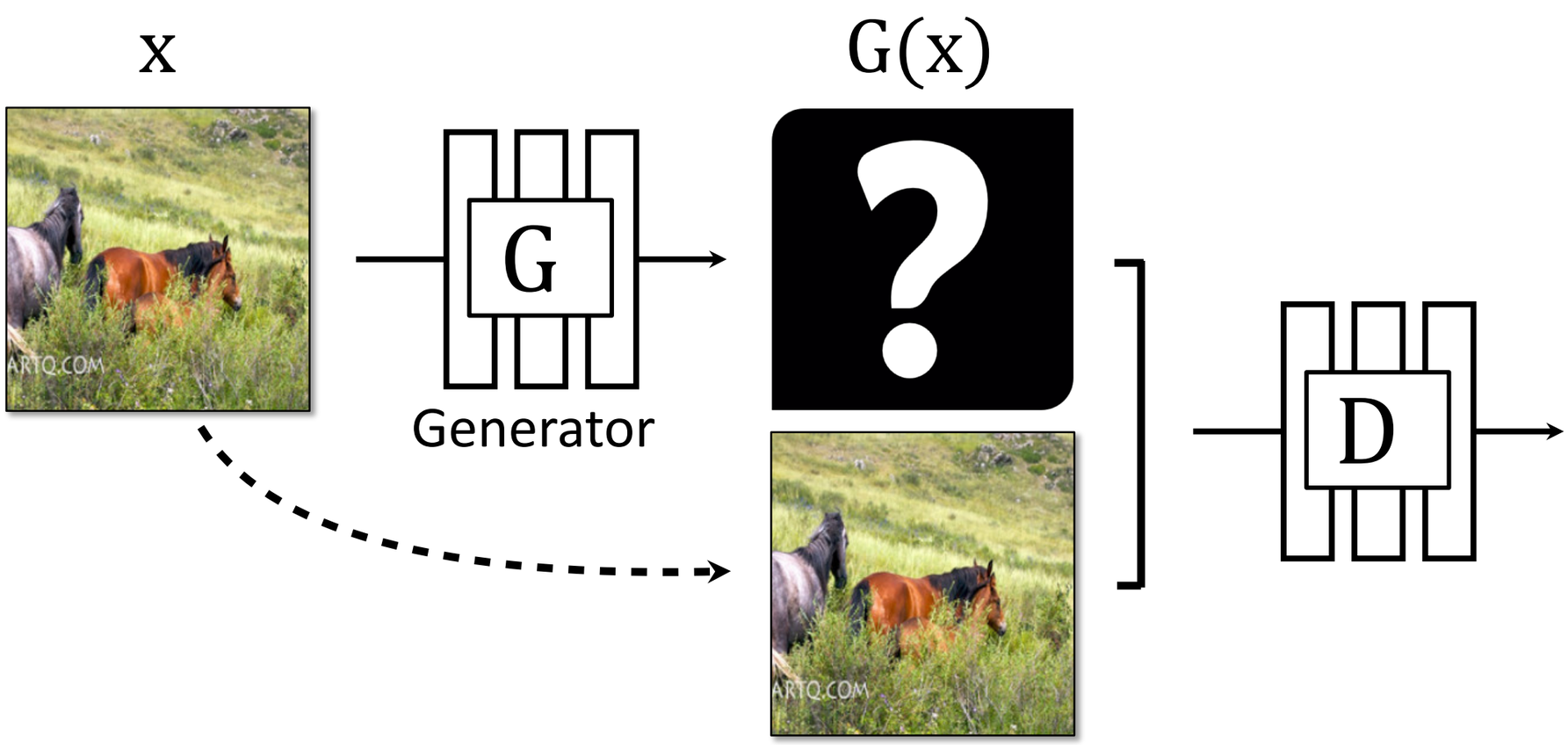
- Expensive to collect pairs.
- Impossible in many scenarios.

Paired



Unpaired





No input-output pairs!

x



G

Generator

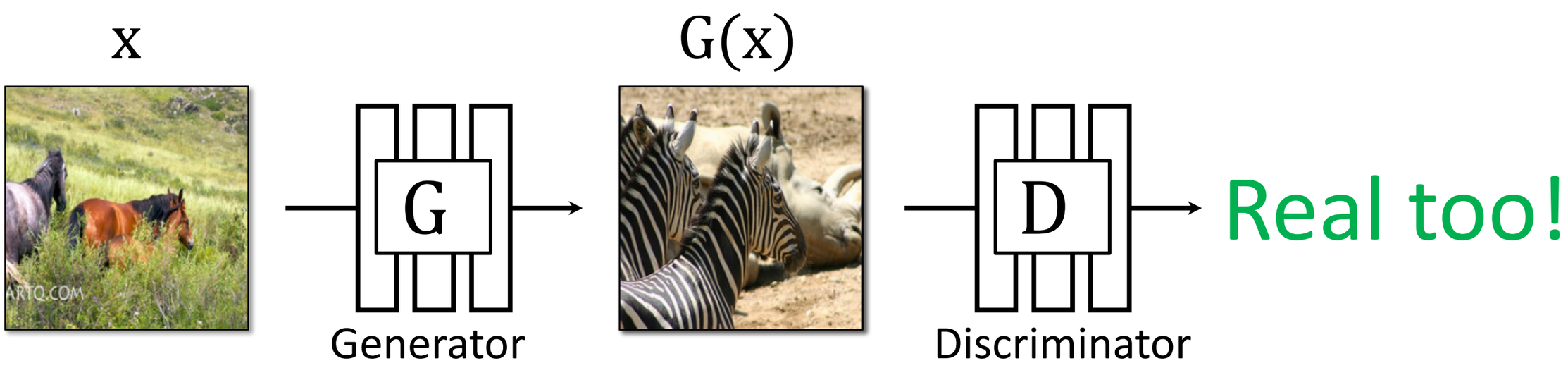
G(x)



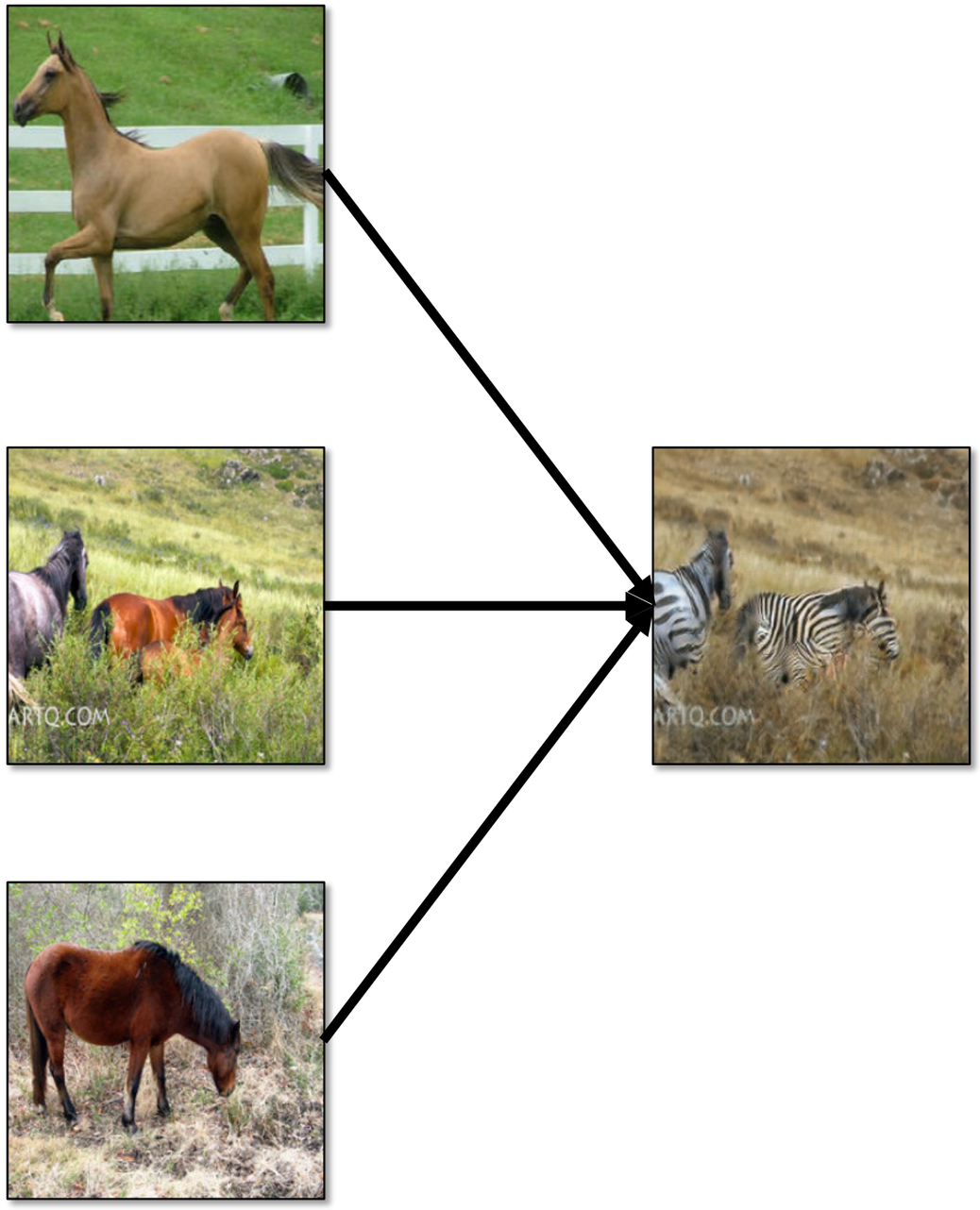
D

Discriminator

Real!

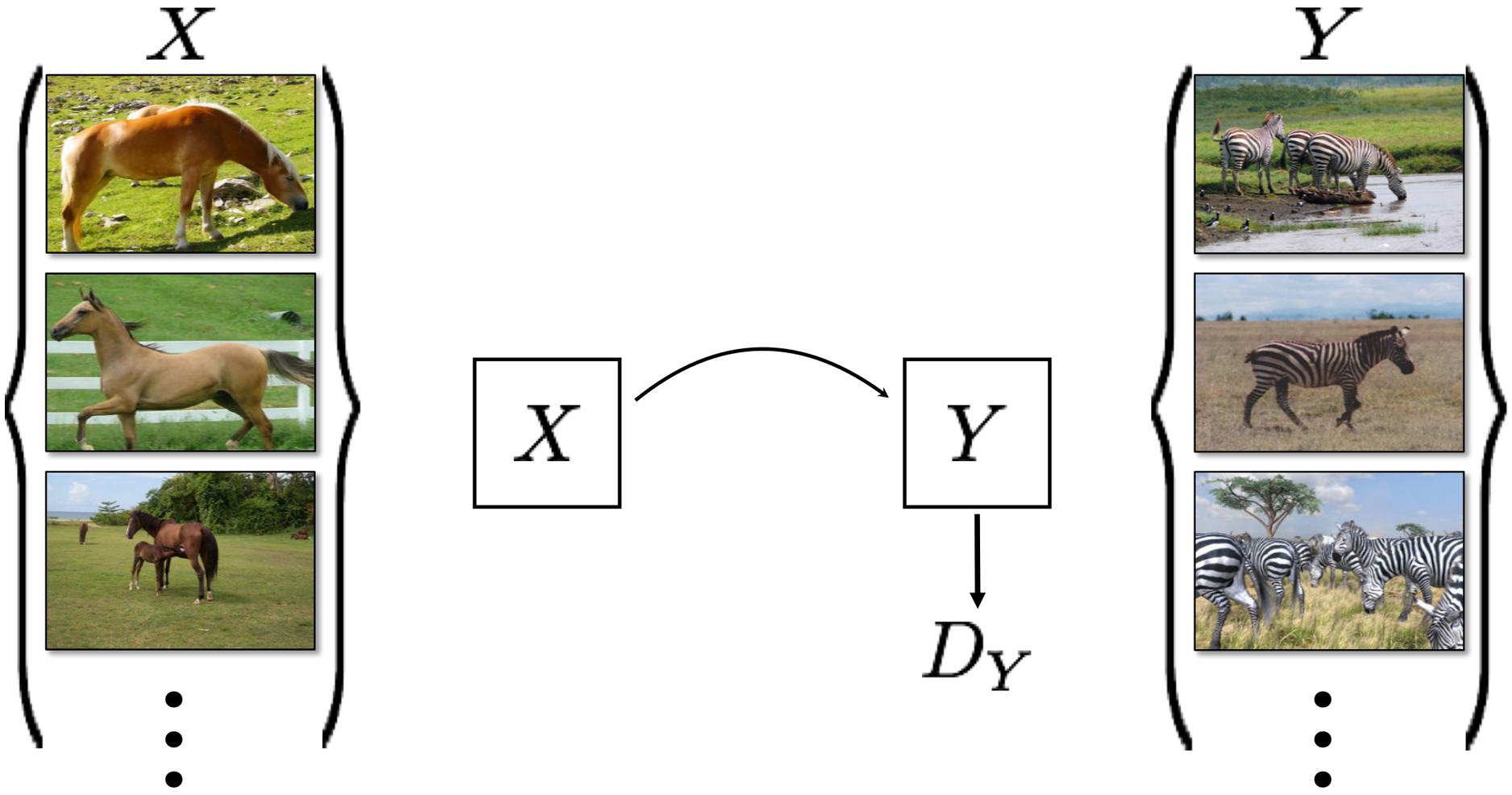


GANs do **not** force output to correspond to input



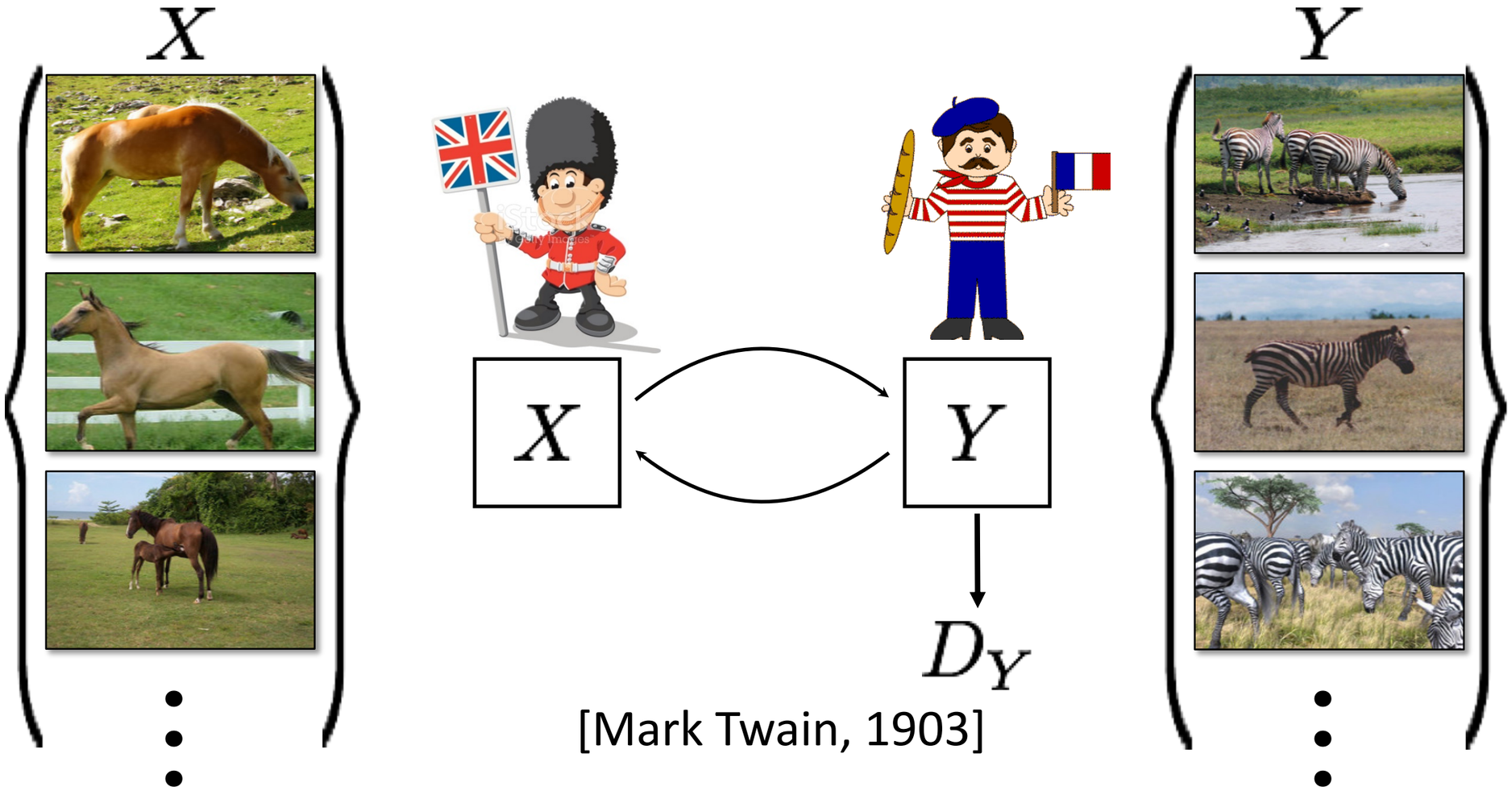
mode collapse!

Cycle-Consistent Adversarial Networks



[Zhu*, Park*, Isola, and Efros, ICCV 2017]

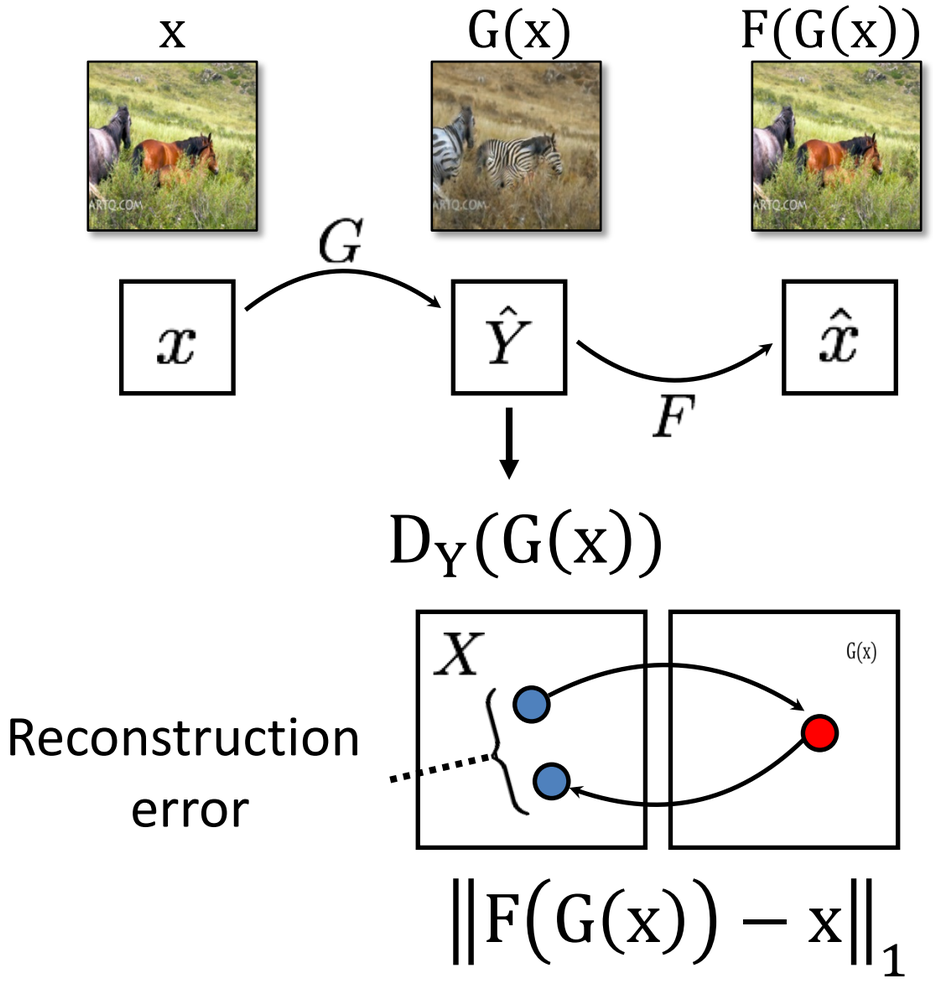
Cycle-Consistent Adversarial Networks



[Mark Twain, 1903]

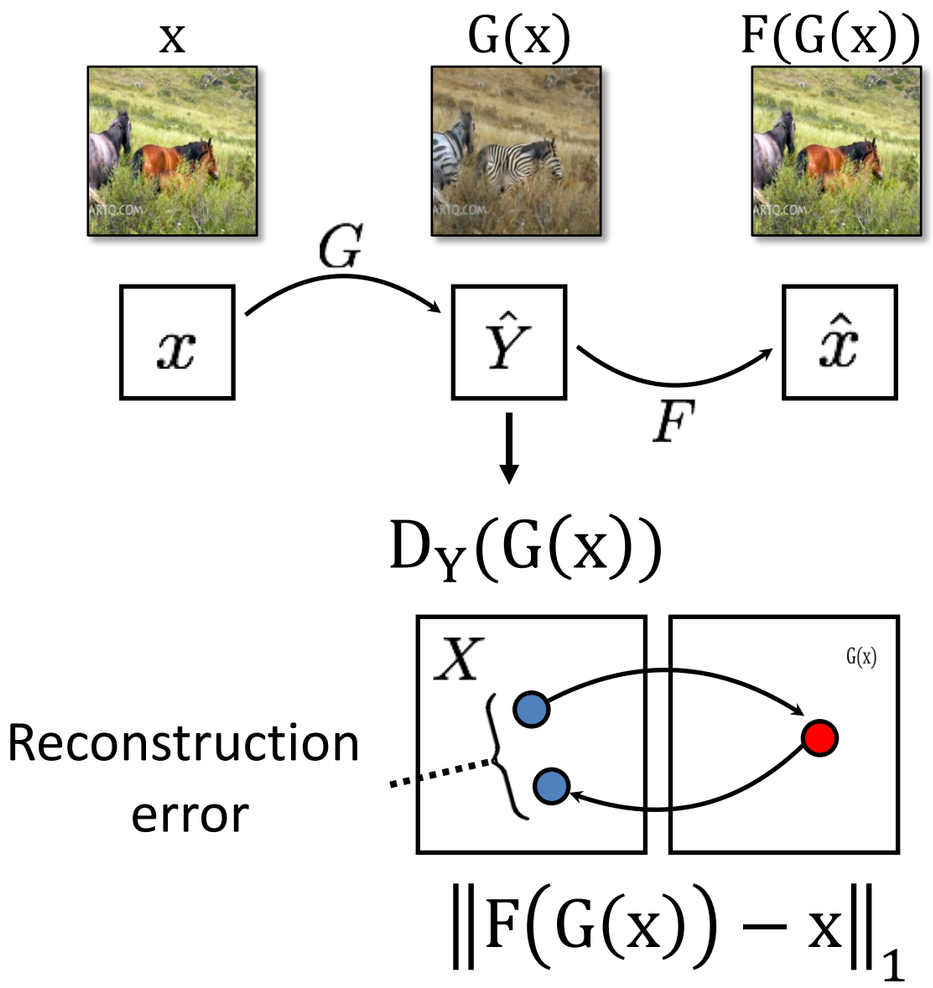
[Zhu*, Park*, Isola, and Efros, ICCV 2017]

Cycle-Consistent Adversarial Networks

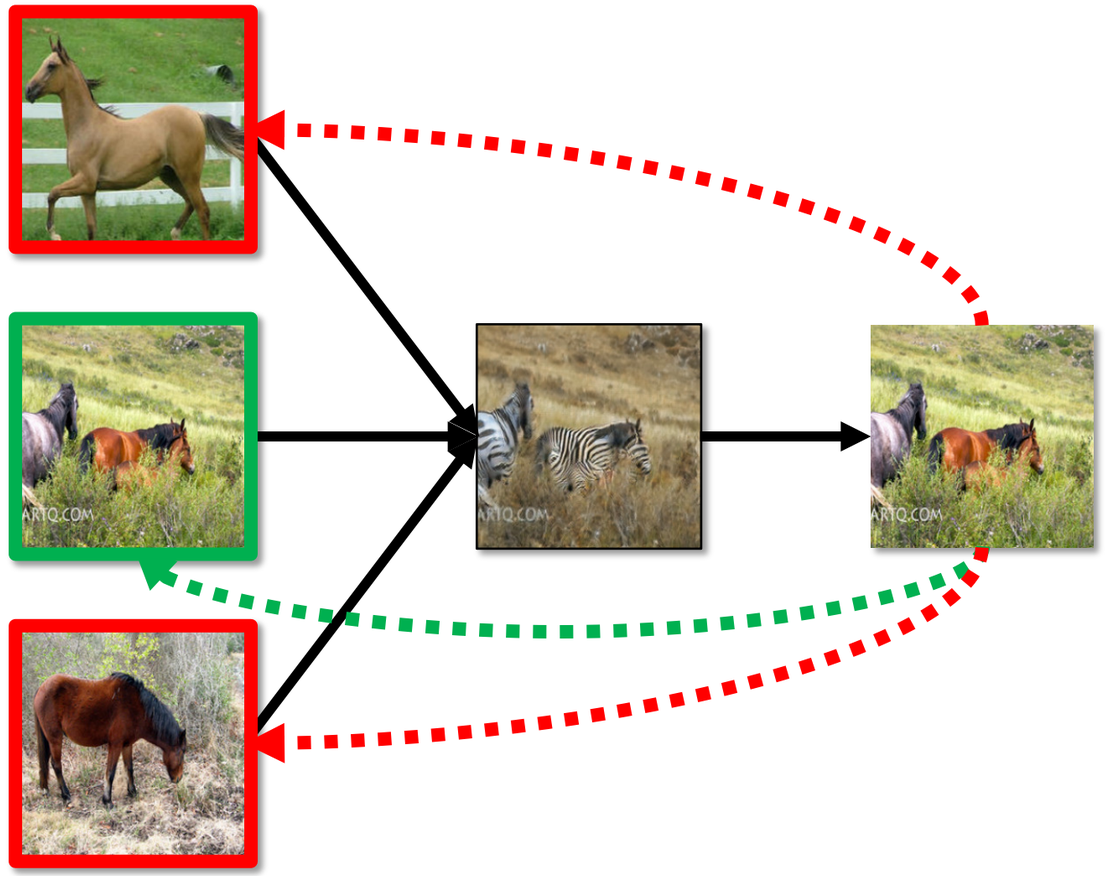


[Zhu*, Park*, Isola, and Efros, ICCV 2017]

Cycle Consistency Loss

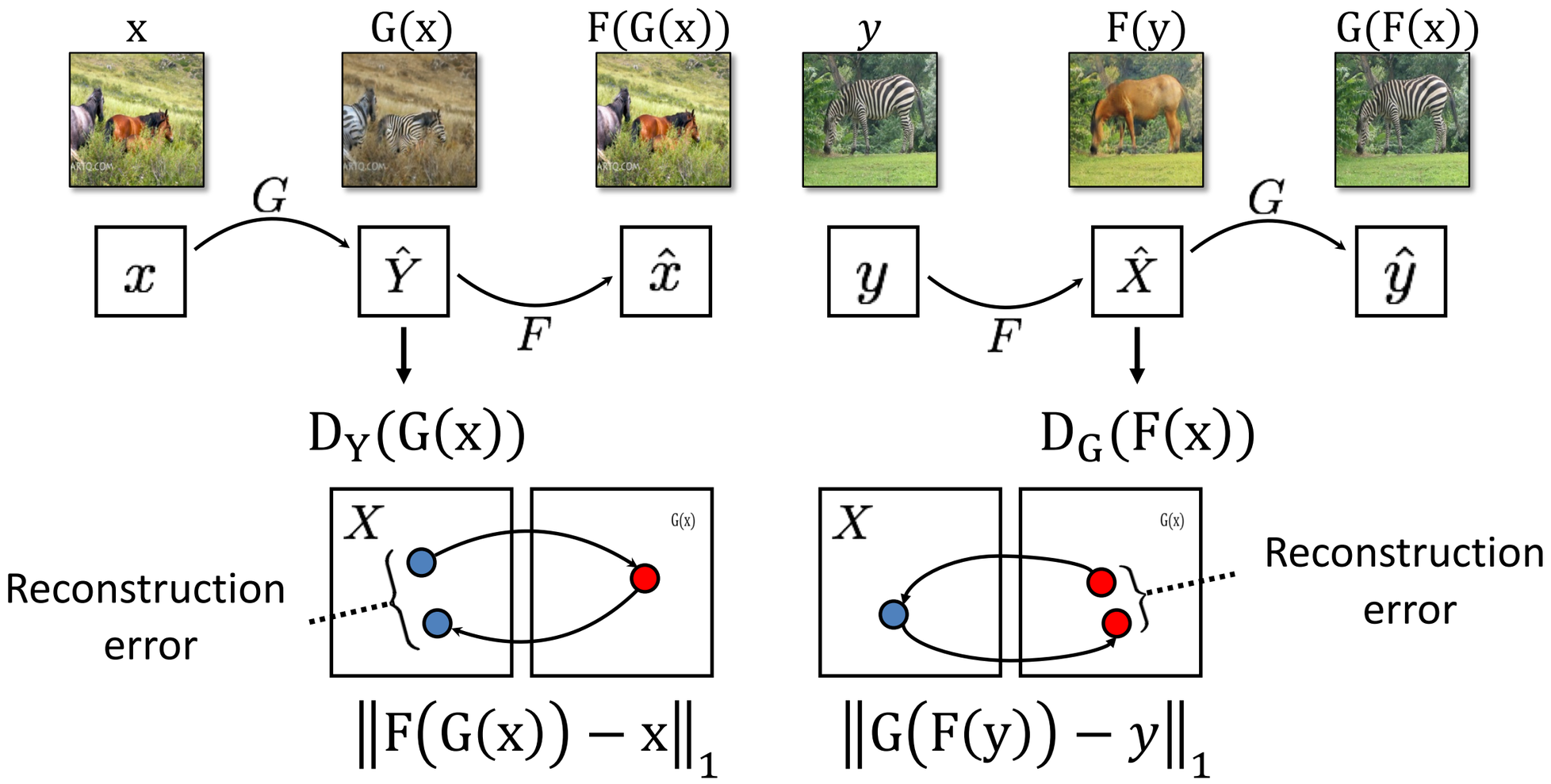


Small cycle loss



[Zhu*, Park*, Isola, and Efros, ICCV 2017]

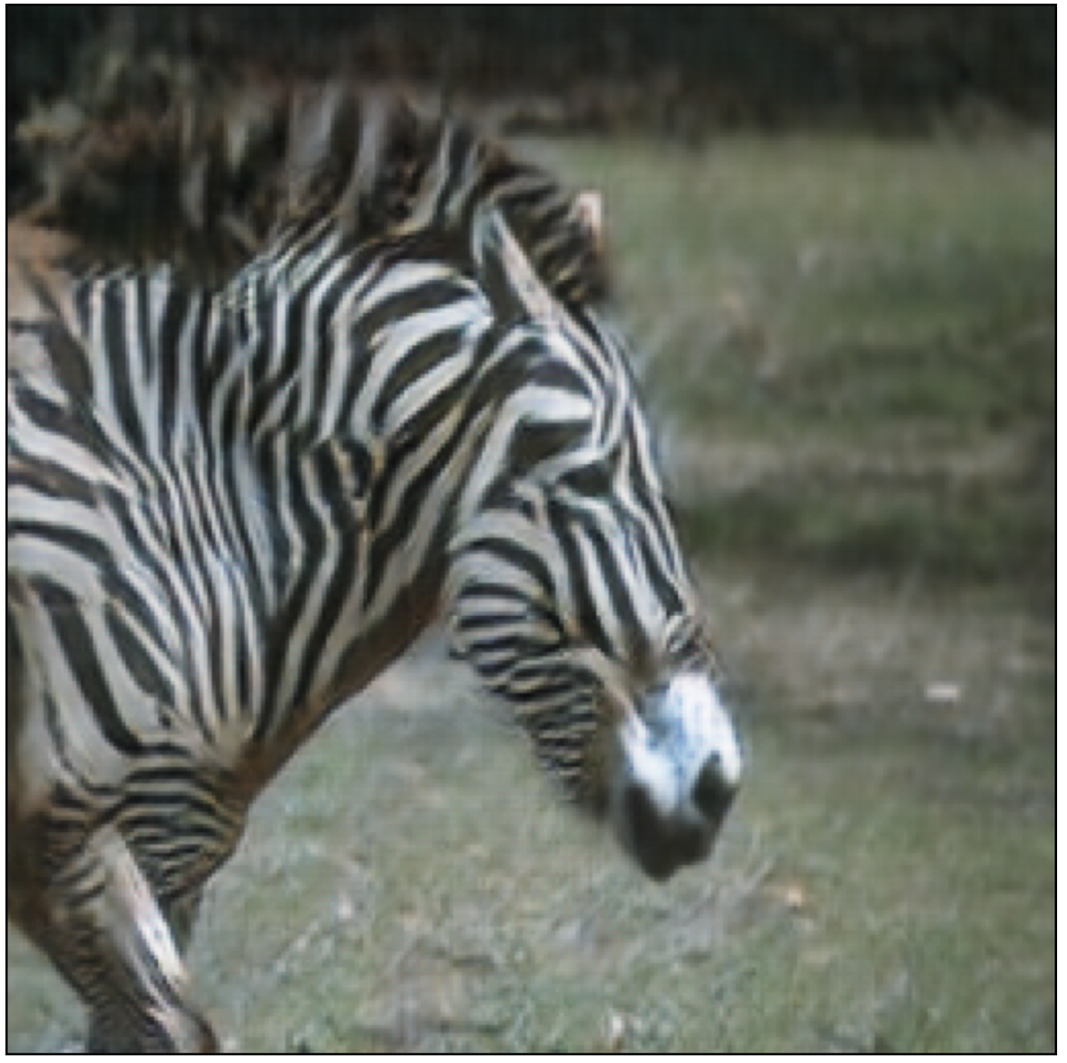
Cycle Consistency Loss



See similar formulations [Yi et al. 2017], [Kim et al. 2017] [Zhu*, Park*, Isola, and Efros, ICCV 2017]

Results









Collection Style Transfer



Photograph
@ Alexei Efros



Monet



Van Gogh



Cezanne



Ukiyo-e


Input
Monet
Van Gogh
Cezanne
Ukiyo-e

Monet’s paintings → photos





Monet’s paintings → photos













Loss	Map → Photo % Turkers labeled *real*	Photo → Map % Turkers labeled *real*
CoGAN [30]	0.6% ± 0.5%	0.9% ± 0.5%
BiGAN/ALI [8, 6]	2.1% ± 1.0%	1.9% ± 0.9%
SimGAN [45]	0.7% ± 0.5%	2.6% ± 1.1%
Feature loss + GAN	1.2% ± 0.6%	0.3% ± 0.2%
CycleGAN (ours)	**26.8% ± 2.8%**	**23.2% ± 3.4%**

AMT ‘real vs fake’ test on maps ↔ aerial

Loss	Per-pixel acc.	Per-class acc.	Class IOU
CoGAN [30]	0.40	0.10	0.06
BiGAN/ALI [8, 6]	0.19	0.06	0.02
SimGAN [45]	0.20	0.10	0.04
Feature loss + GAN	0.06	0.04	0.01
CycleGAN (ours)	**0.52**	**0.17**	**0.11**

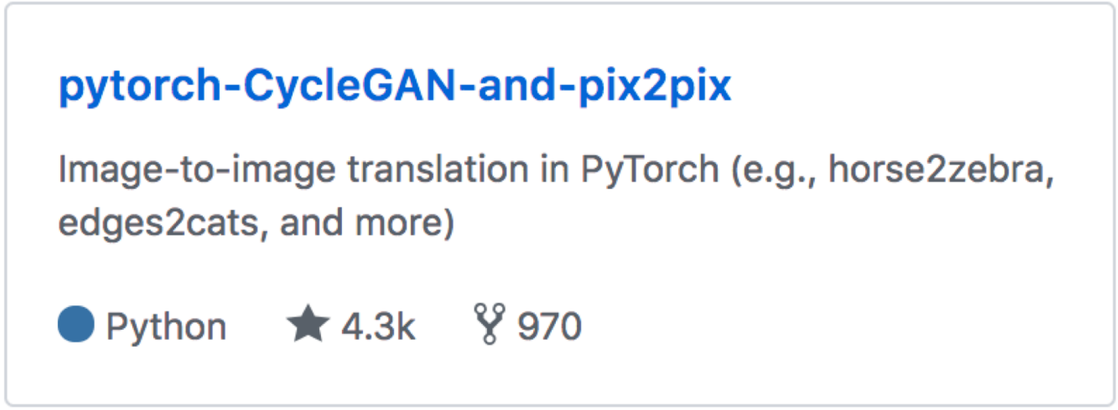
FCN scores on cityscapes labels→ photos

Loss	Per-pixel acc.	Per-class acc.	Class IOU
CoGAN [30]	0.45	0.11	0.08
BiGAN/ALI [8, 6]	0.41	0.13	0.07
SimGAN [45]	0.47	0.11	0.07
Feature loss + GAN	0.50	0.10	0.06
CycleGAN (ours)	**0.58**	**0.22**	**0.16**

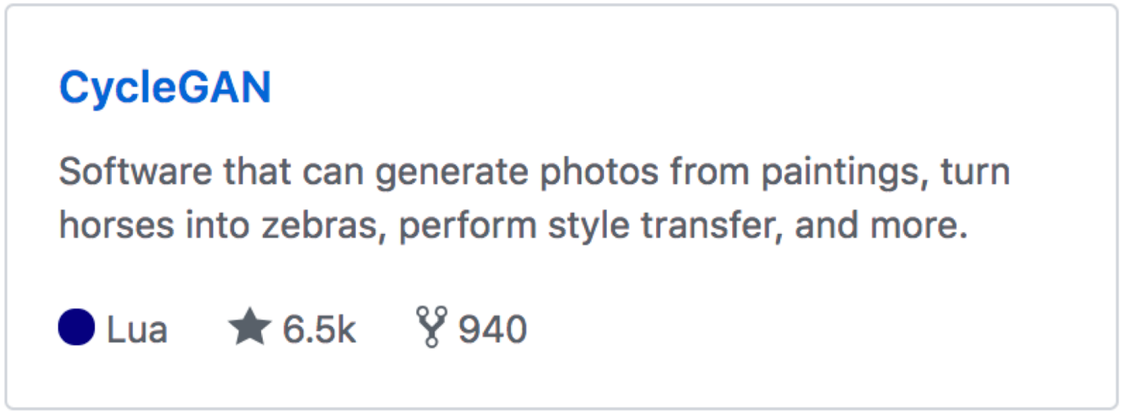
Classification performance of photo→labels

CycleGAN implementations

PyTorch



Torch



20+ implementations by researchers/developers:

- Tensorflow, Chainer, mxnet, Lasagne, Keras…

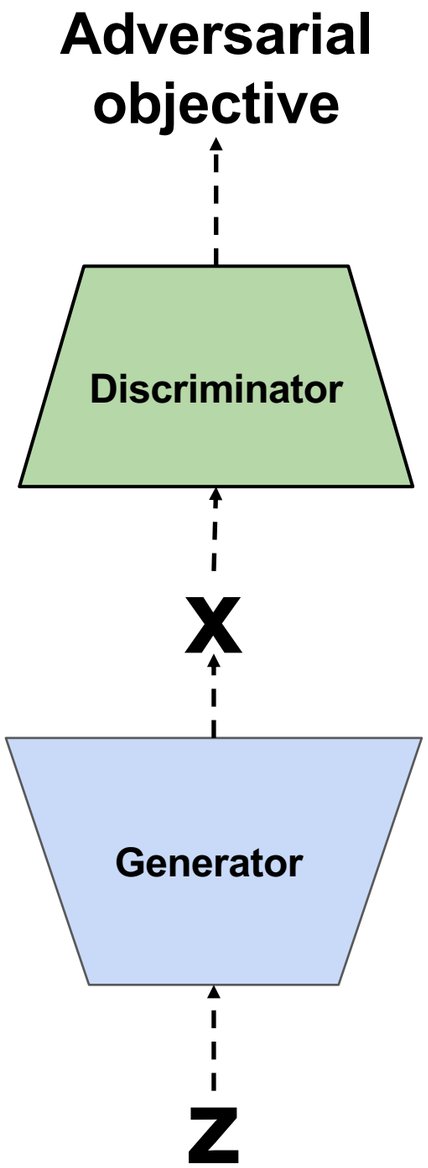
Disentangling Content and Pose with an Adversarial loss

Emily Denton

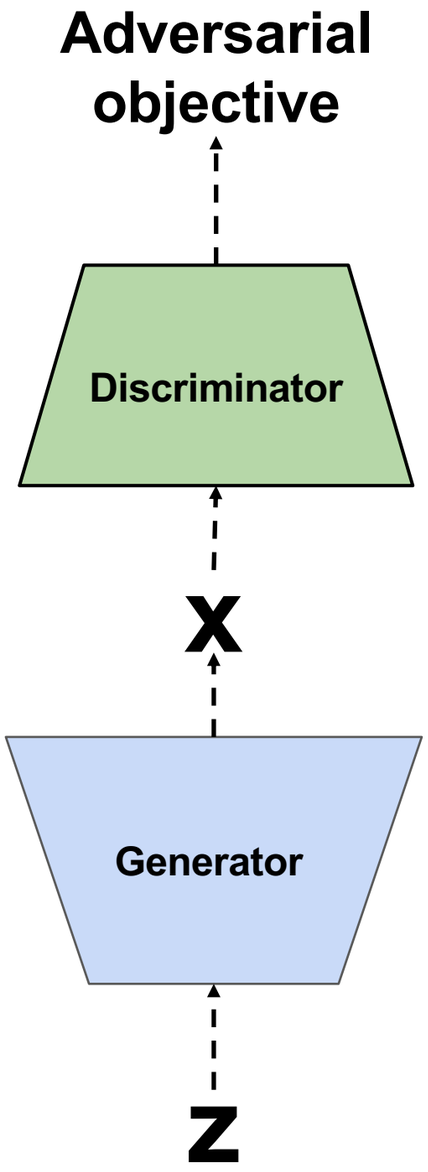


CVPR GAN Tutorial
June 2018

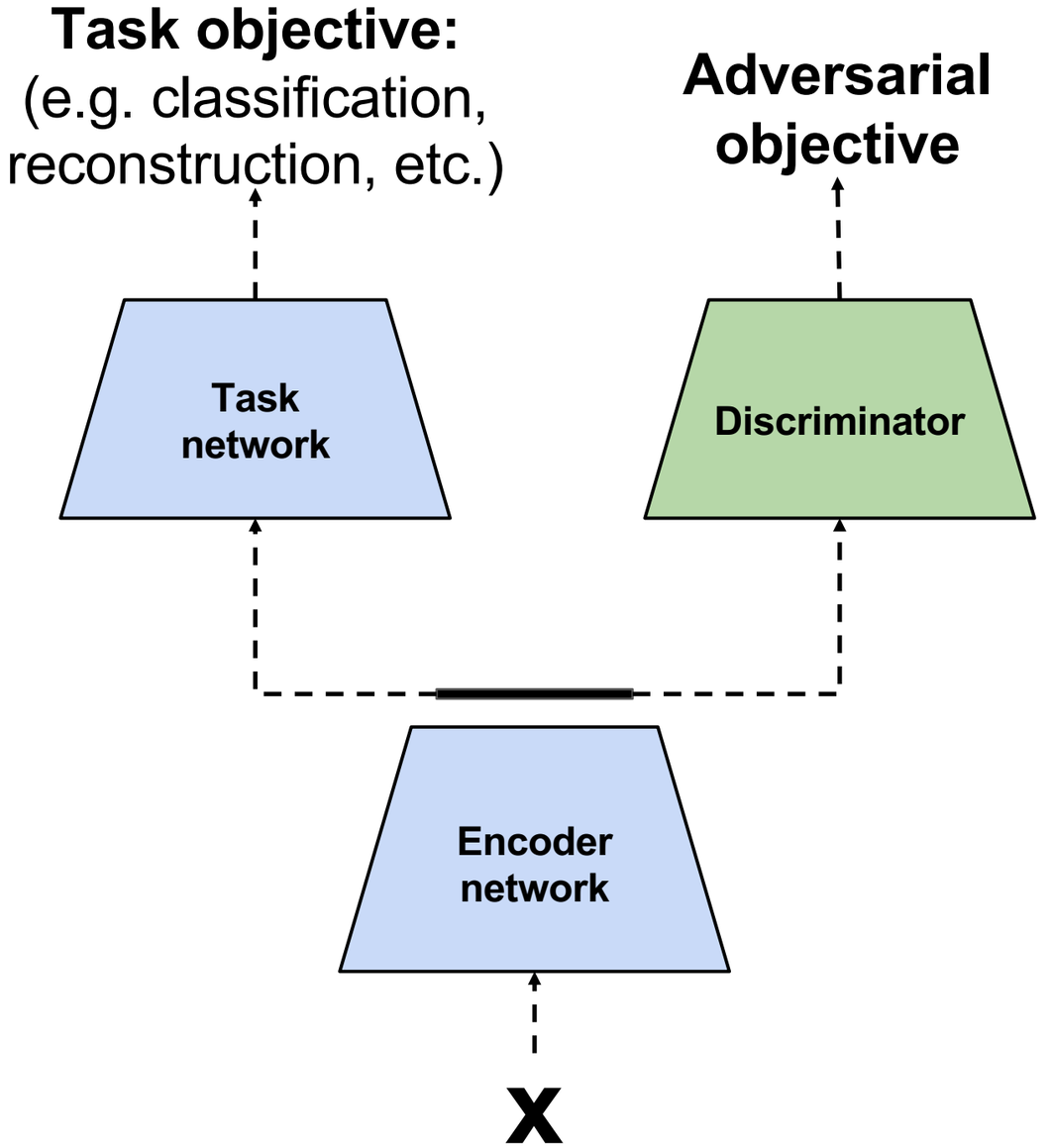
Generative adversarial network framework:



Generative adversarial network framework:



Adversarial losses to shape representations:



Part I: Disentangling content and pose with an adversarial loss

Denton and Birodkar. *Unsupervised Learning of Disentangled Representations from Video*. NIPS, 2017

Part II: Survey of adversarial losses in feature space

Disentangled Representation Net (DrNet)

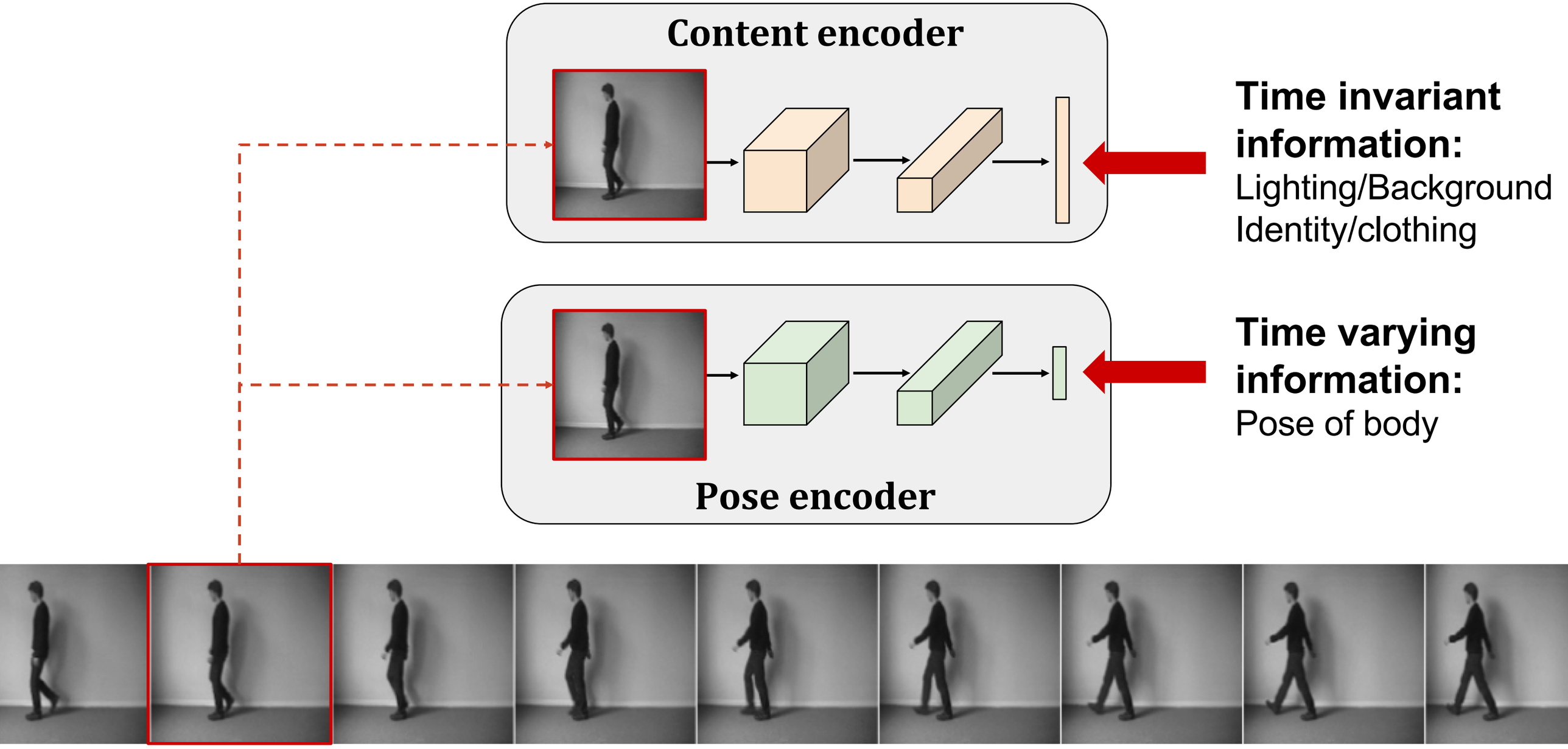
Disentangling auto-encoder that factorizes image sequences into **temporally constant (content)** and **temporally varying (pose)** components

Time varying information: Pose of body



Time invariant information: Lighting, background, identity, clothing

DrNet: two seperate encoders



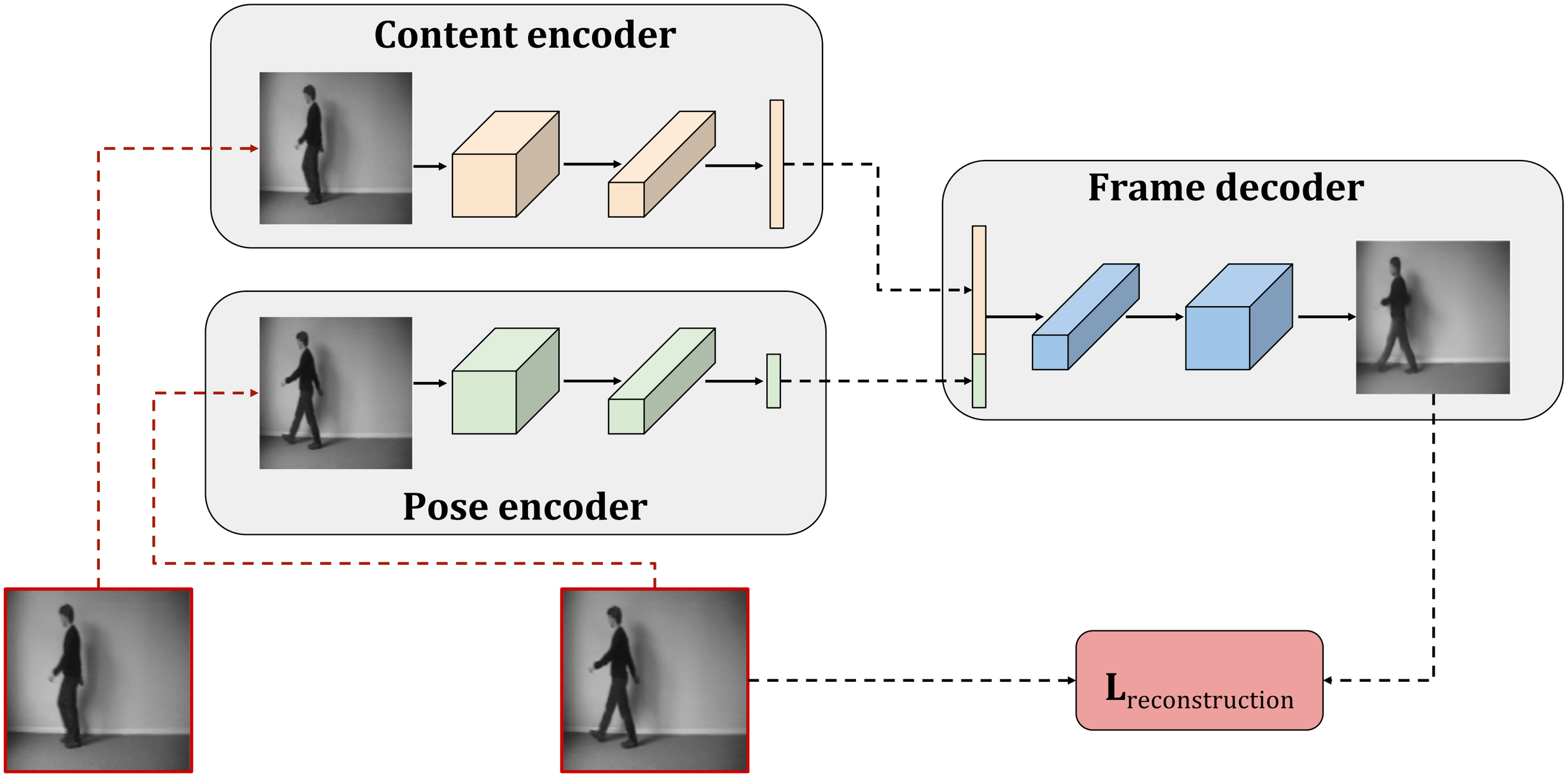
DrNet: training

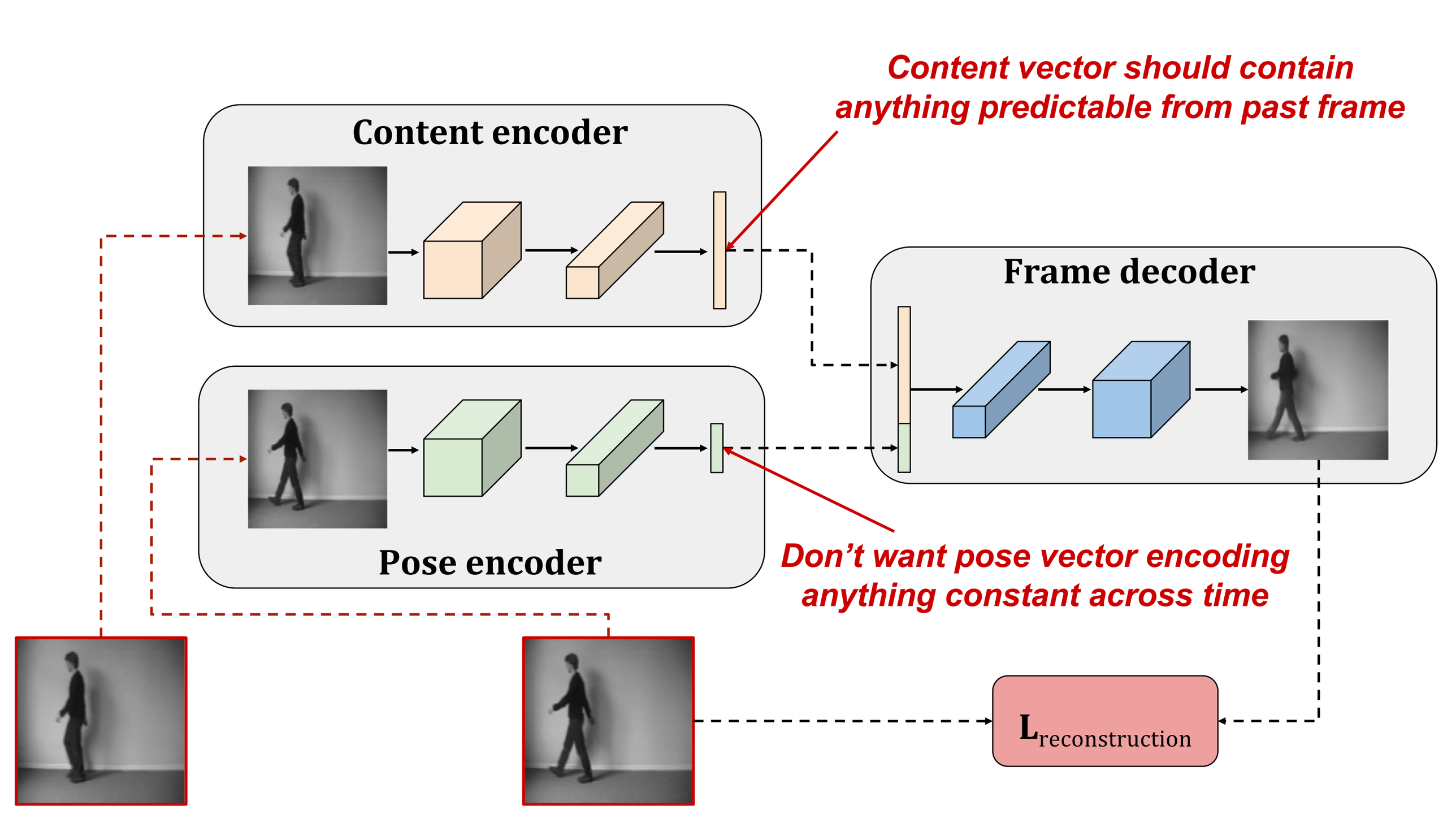
- **Reconstruction loss** drives training
- **Similarity loss** makes content vectors invariant across time
- **Adversarial loss** enforces pose vectors to only contain info that changes across time

DrNet: training

- **Reconstruction loss** drives training
- **Similarity loss** makes content vectors invariant across time
- **Adversarial loss** enforces pose vectors to only contain info that changes across time



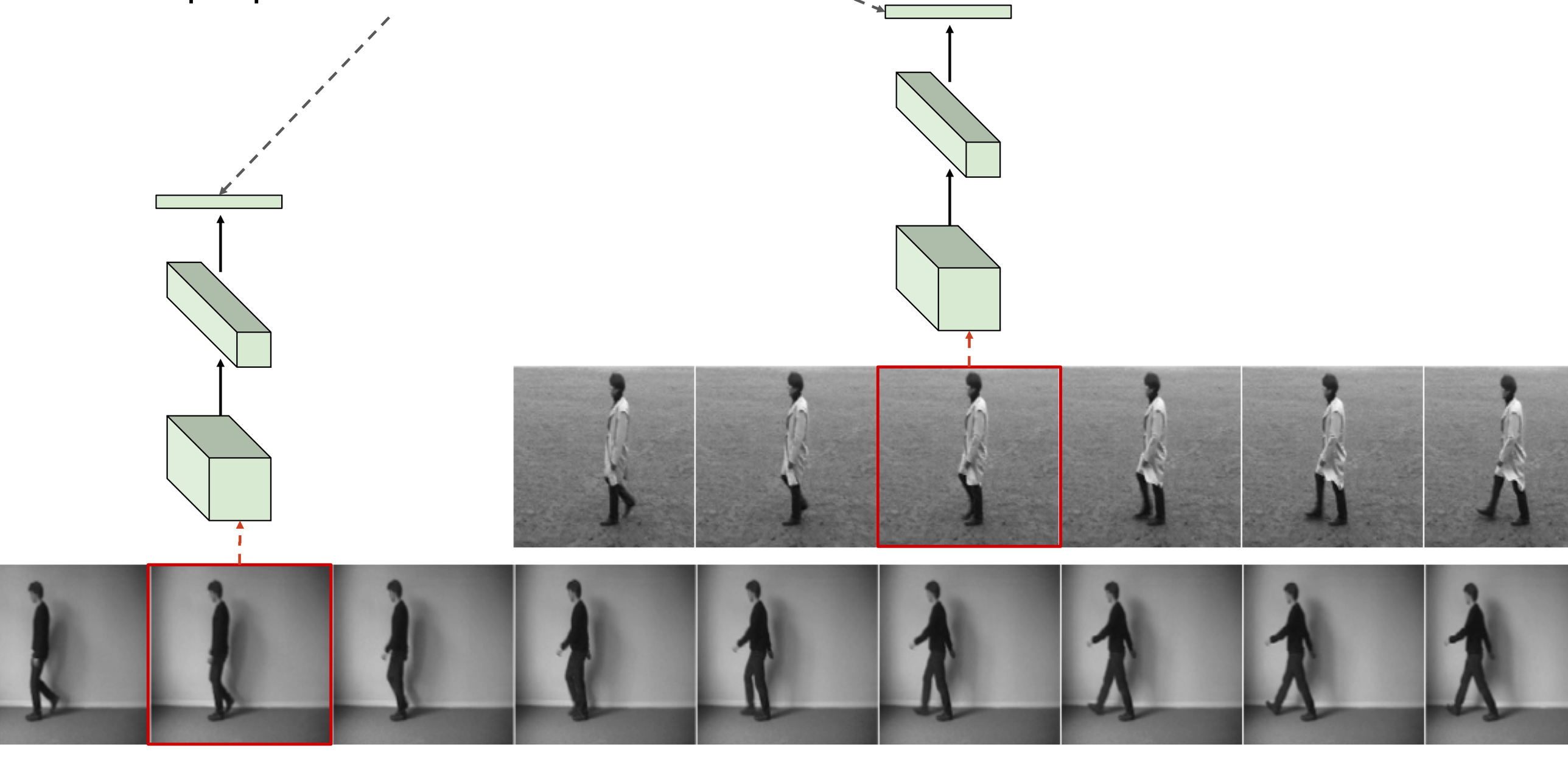

Content encoder
Frame decoder
Pose encoder
$L_{reconstruction}$

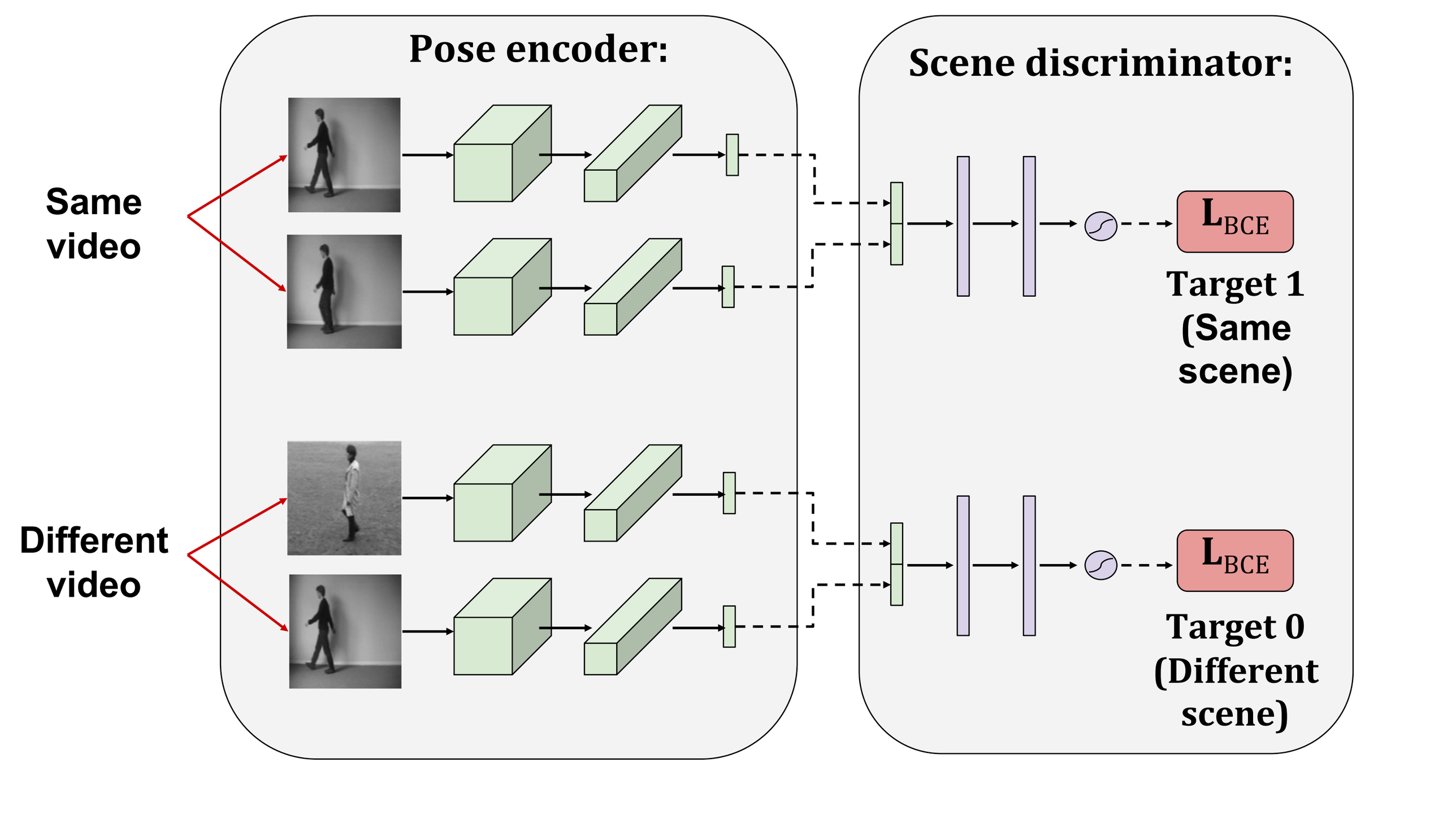

Content vector should contain anything predictable from past frame
Content encoder
Frame decoder
Pose encoder
Don’t want pose vector encoding anything constant across time
$L_{reconstruction}$

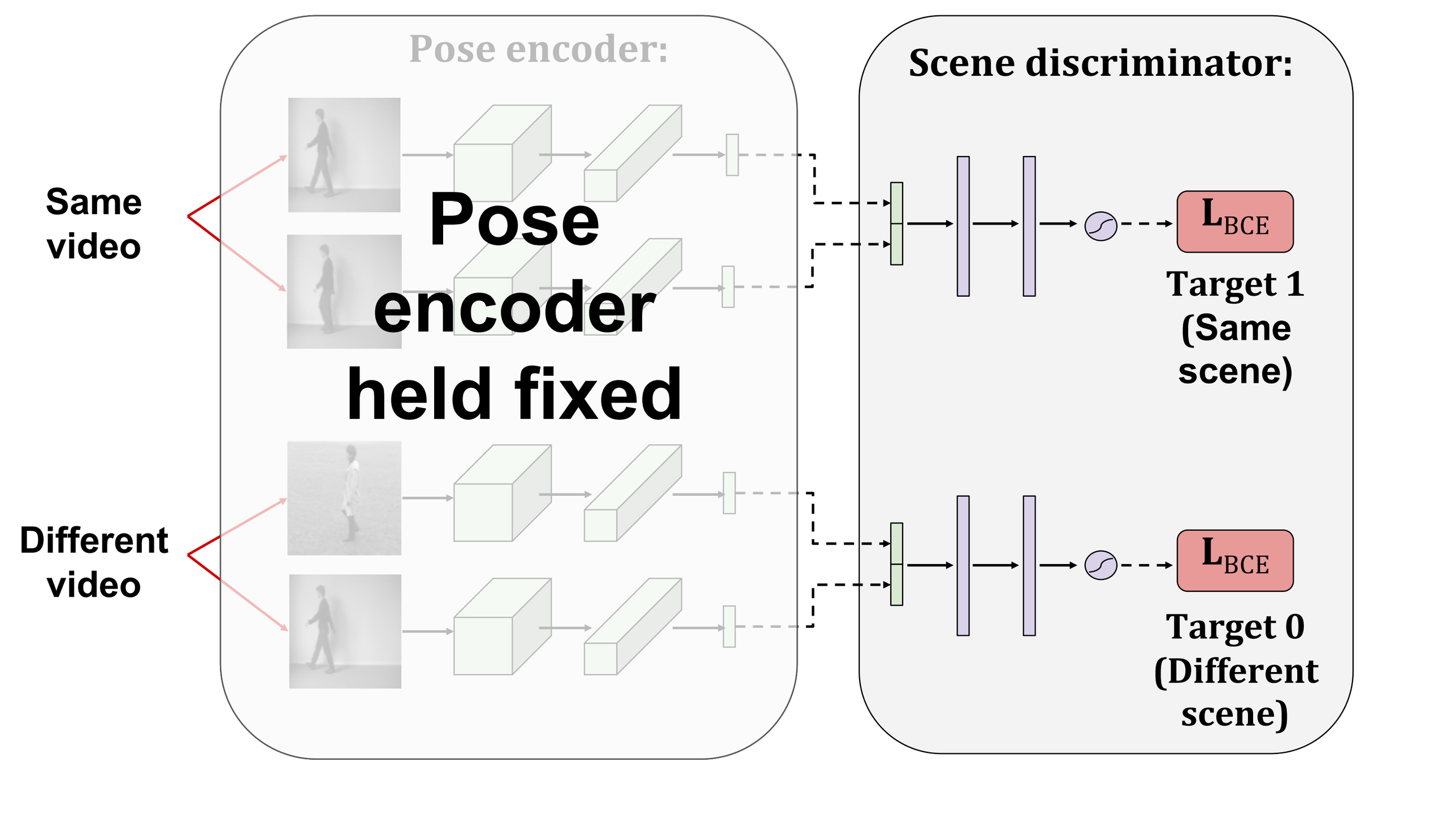
DrNet: training

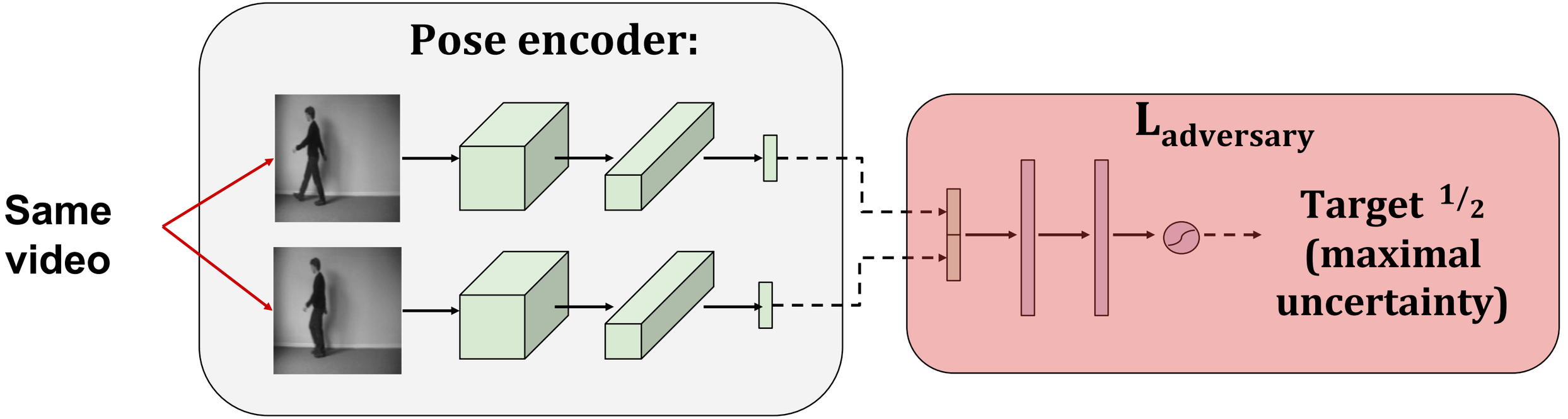
- **Reconstruction loss** drives training
- **Similarity loss** makes content vectors invariant across time
- **Adversarial loss** enforces pose vectors to only contain info that changes across time

Should not be able to distinguish which video clip a pose vector comes from



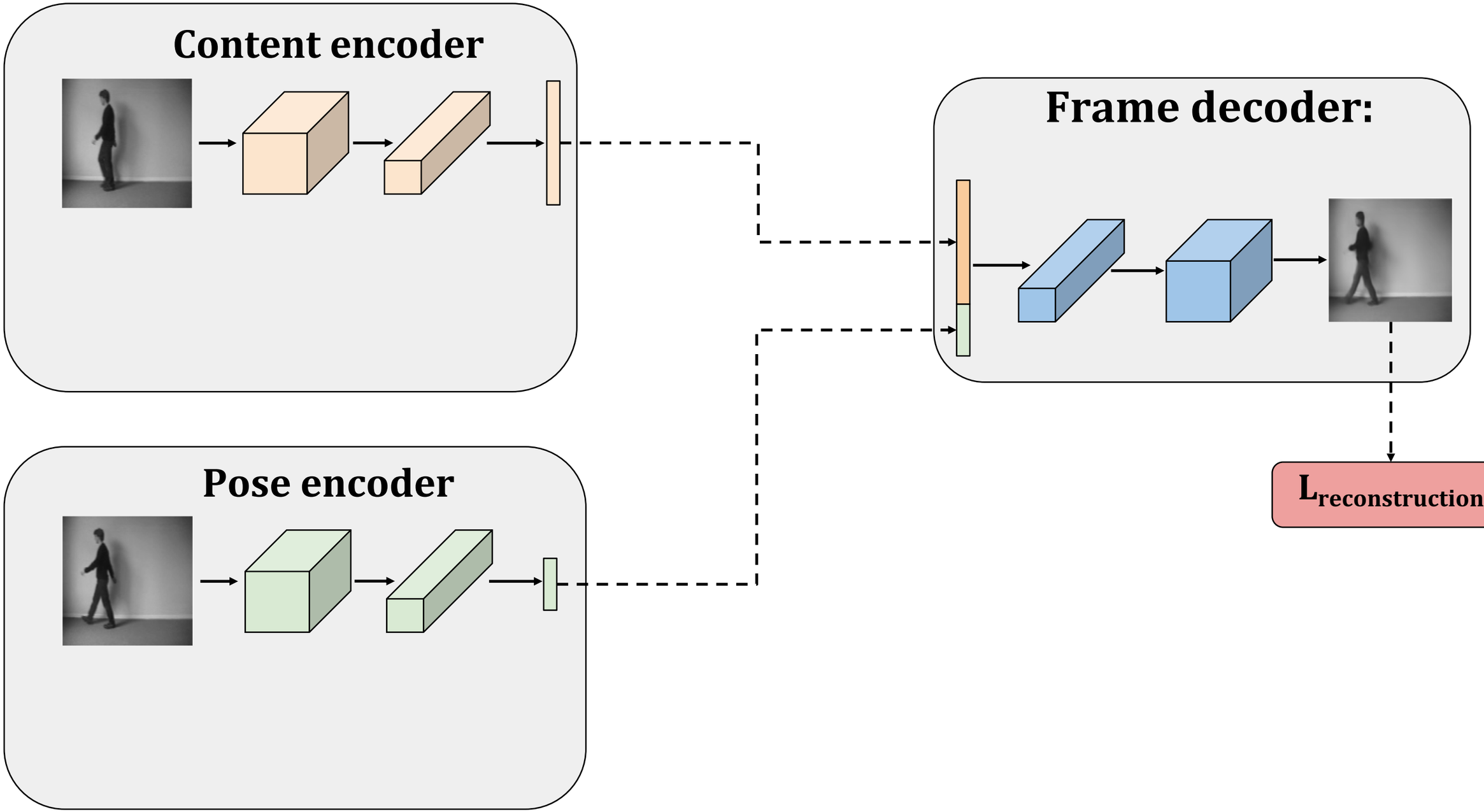

Pose encoder:
Scene discriminator:
Same video
Different video
L_{BCE}
Target 1 (Same scene)
L_{BCE}
Target 0 (Different scene)


Pose encoder:
Scene discriminator:
Same video
Different video
Pose encoder held fixed
L_{BCE}
Target 1 (Same scene)
L_{BCE}
Target 0 (Different scene)

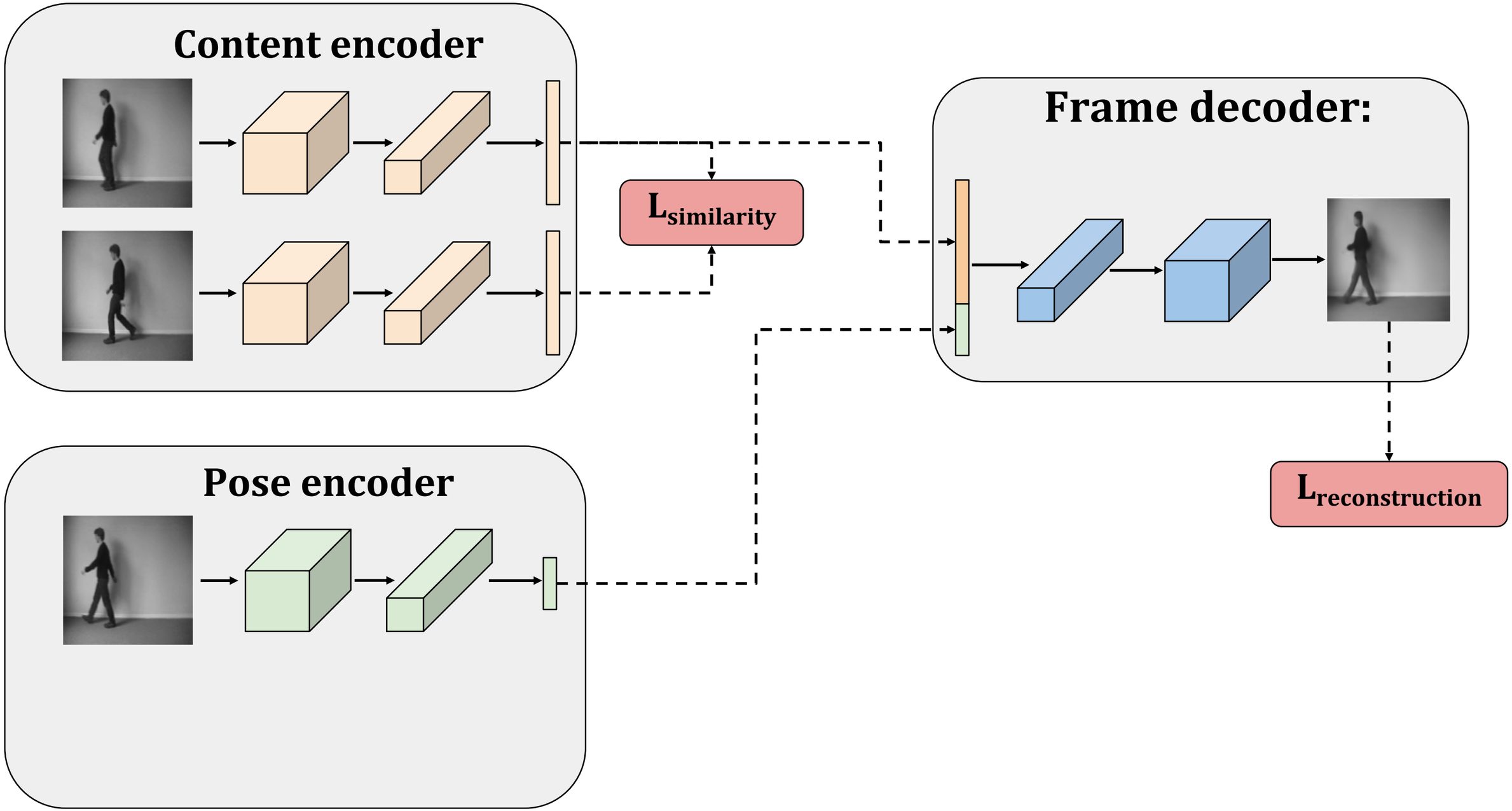


Train pose encoder to produce pose vectors that make the discriminator **maximally uncertain** about the content of the video

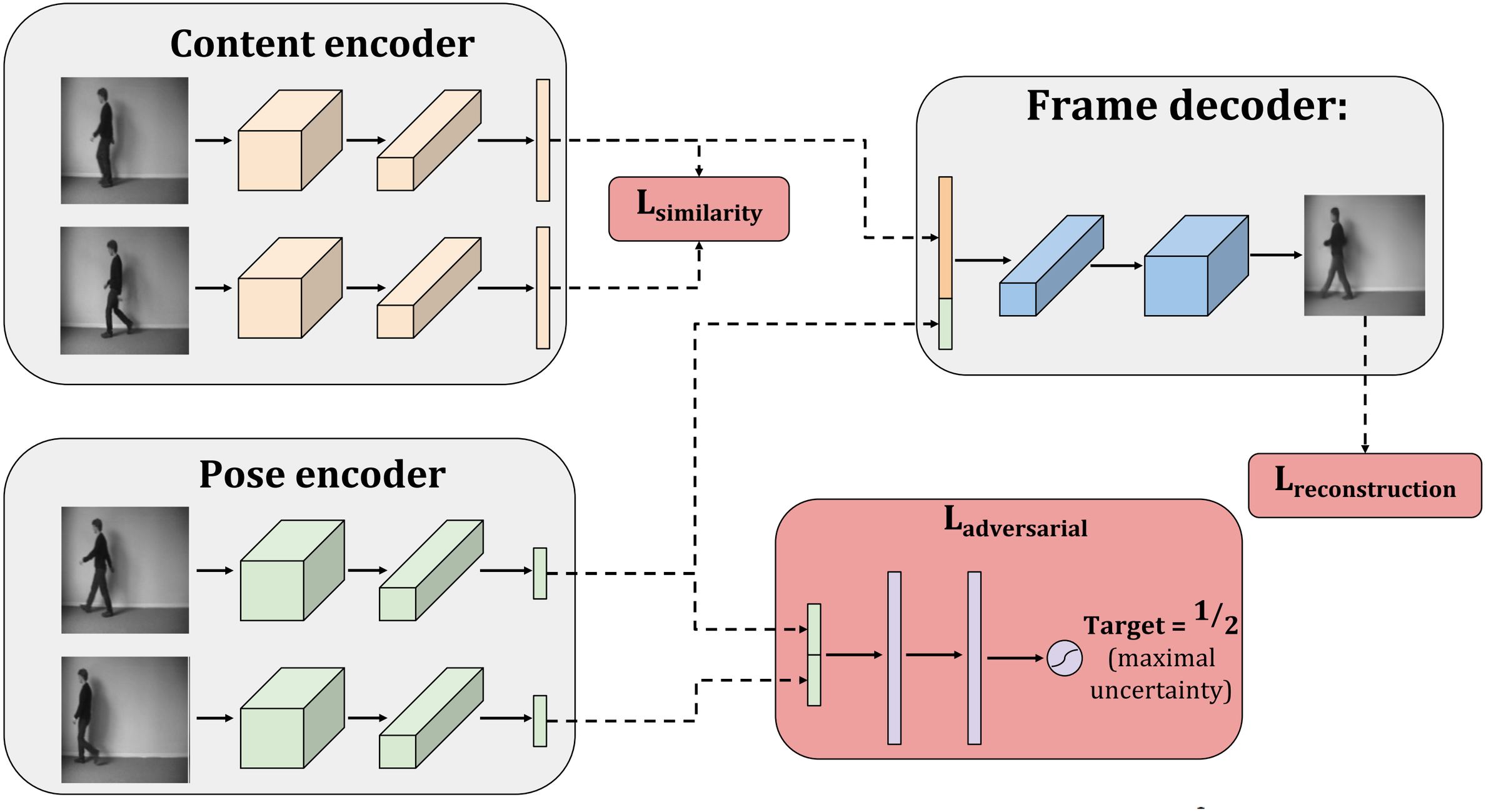
Scene discriminator held fixed, only used to compute gradients for pose encoder



$$\mathcal{L} = \mathcal{L}_{reconstruction}(E_c, E_p, D) + \alpha \mathcal{L}_{similarity}(E_c) + \beta(\mathcal{L}_{adversarial}(E_p) + \mathcal{L}_{adversarial}(C))$$



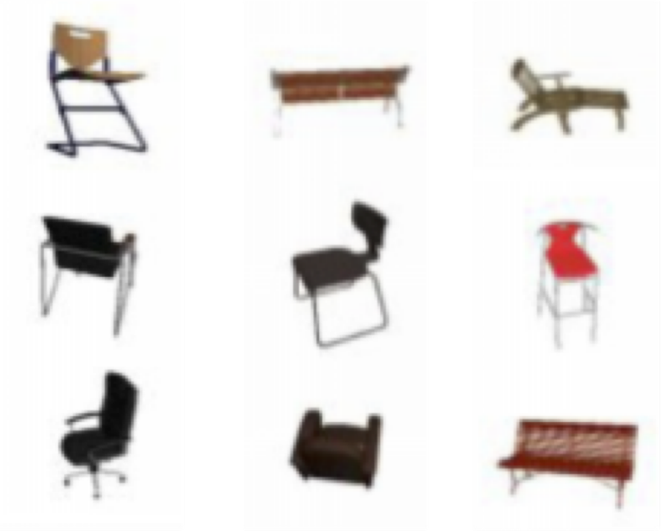
$$\mathcal{L} = \mathcal{L}_{reconstruction}(E_c, E_p, D) + \alpha \mathcal{L}_{similarity}(E_c) + \beta(\mathcal{L}_{adversarial}(E_p) + \mathcal{L}_{adversarial}(C))$$



$$\mathcal{L} = \mathcal{L}_{reconstruction}(E_c, E_p, D) + \alpha \mathcal{L}_{similarity}(E_c) + \beta(\mathcal{L}_{adversarial}(E_p) + \mathcal{L}_{adversarial}(C))$$

SUNCG dataset: rotating objects

- 280 chair models, 5 elevations, large variability
- Video sequence: camera rotates around chair

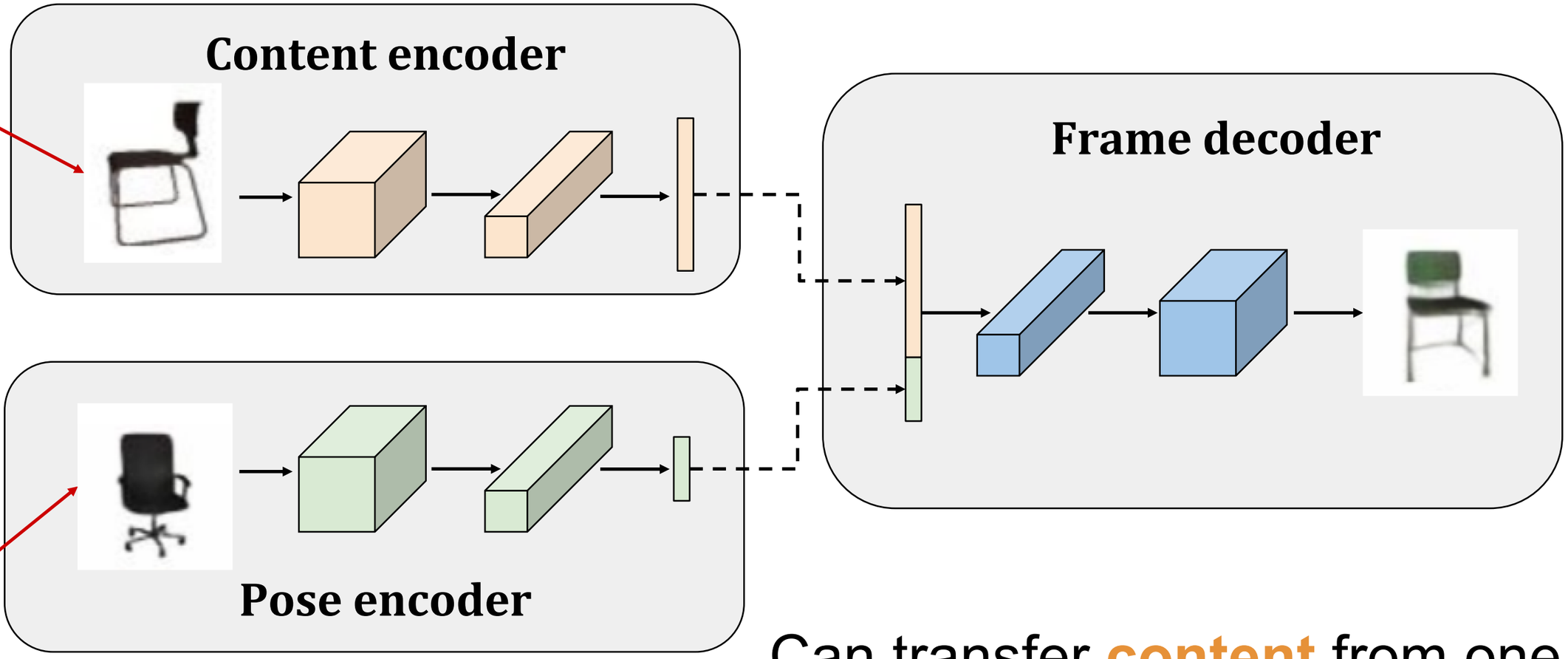




S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene comp

Image synthesis by analogy

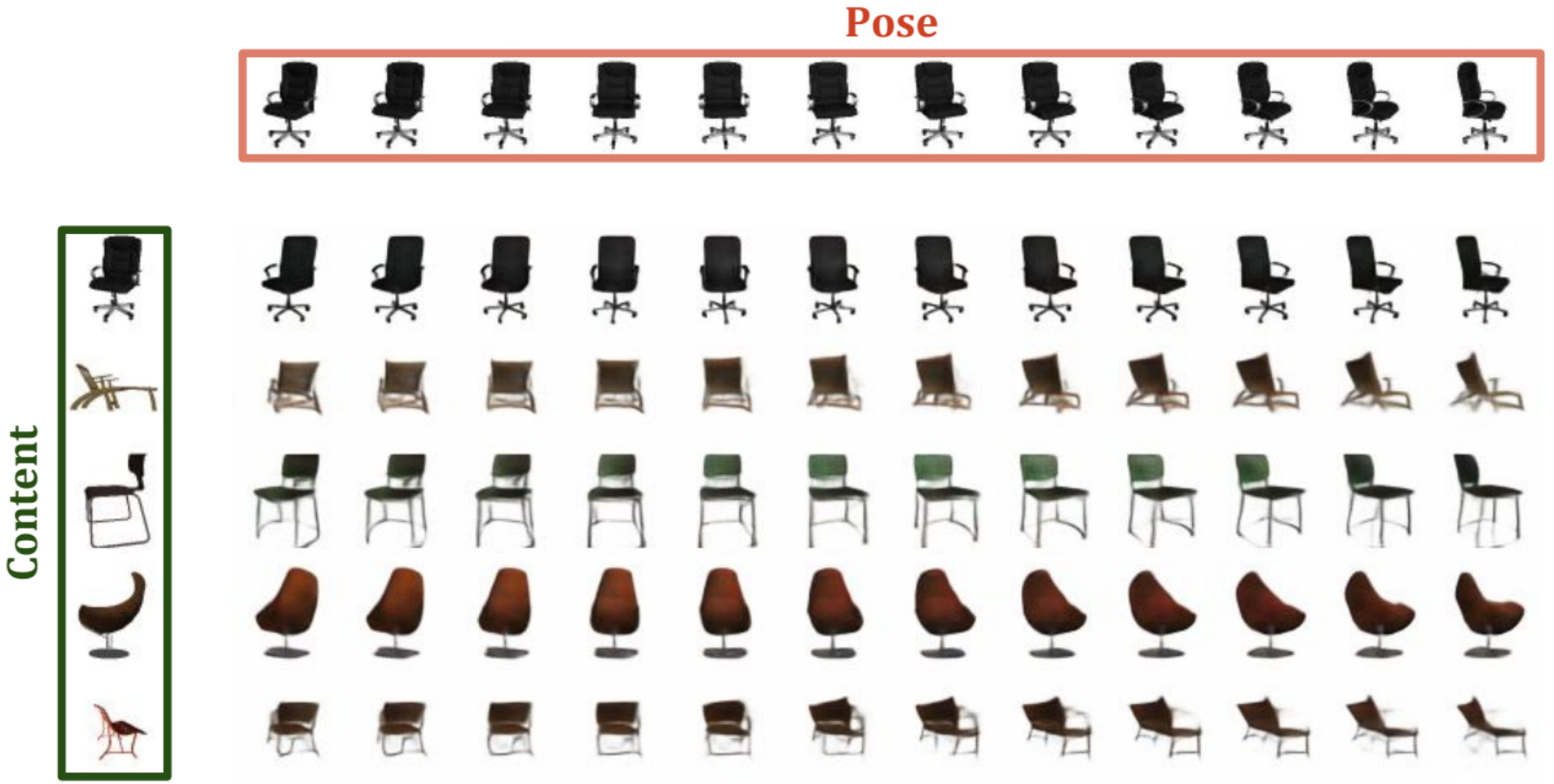
Content image



Pose image

Can transfer **content** from one image and **pose** from another to synthesize a **new image**

Image synthesis by analogy



Interpolation in pose space



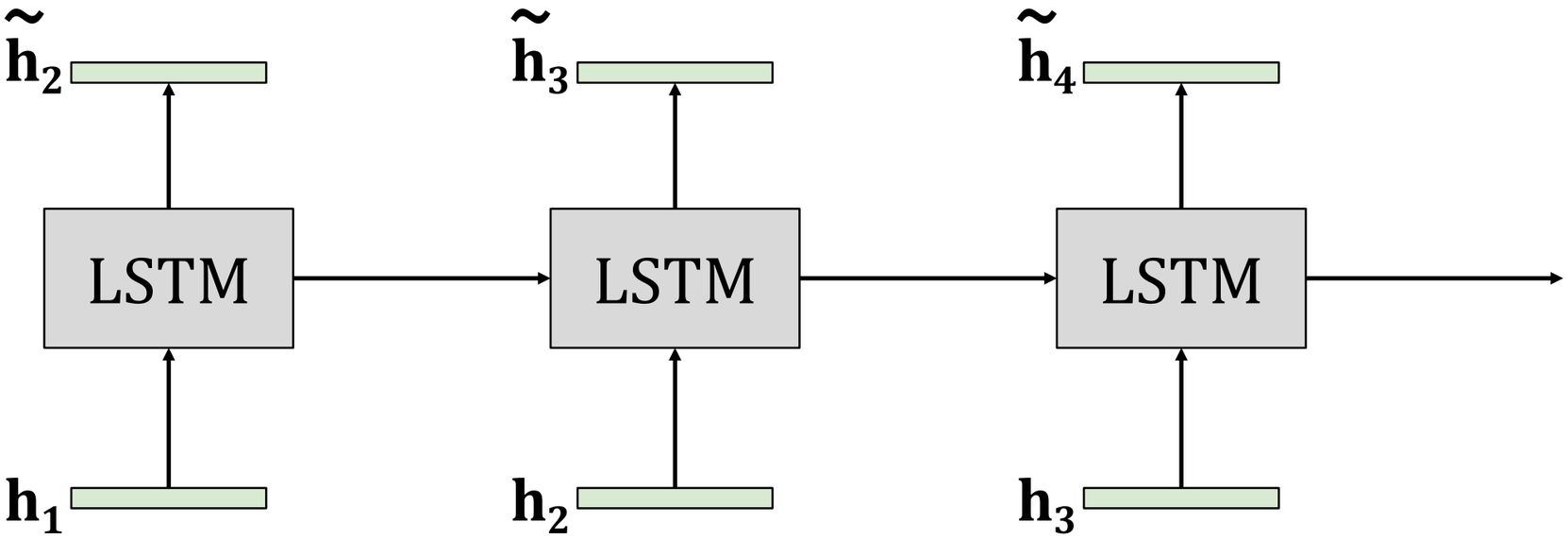
Video prediction

- A representation that factorizes into temporally constant and temporally varying components is particularly useful for video prediction

- Instead of modeling how the entire scene changes, **only need to predict the temporally varying component**

- **Prediction** done entirely in latent **pose space**

Train LSTM to predict future **pose** vectors

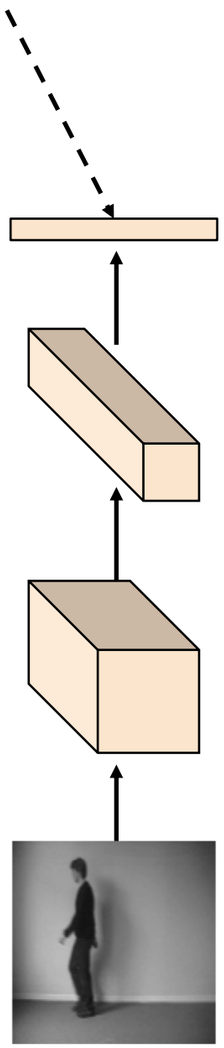


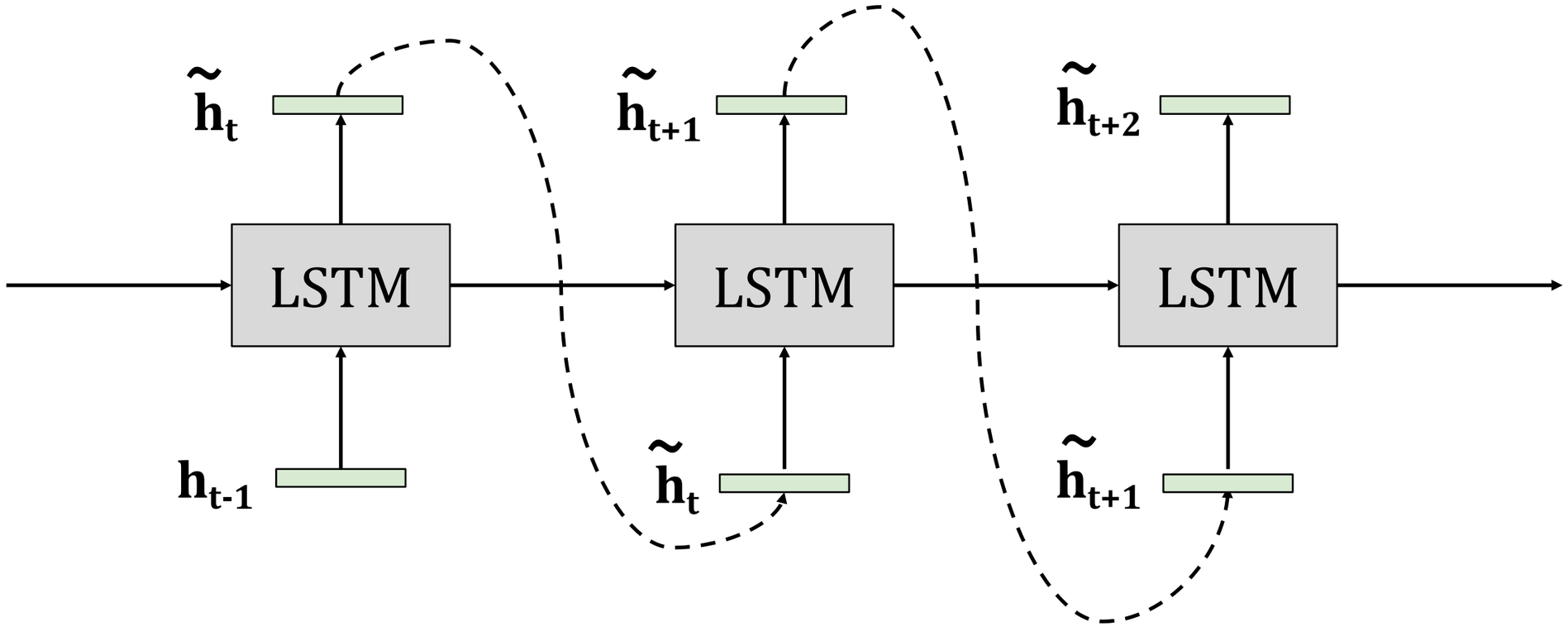
Don’t have to worry about content vectors - they are fixed across time by design

Test time: generating a video sequence

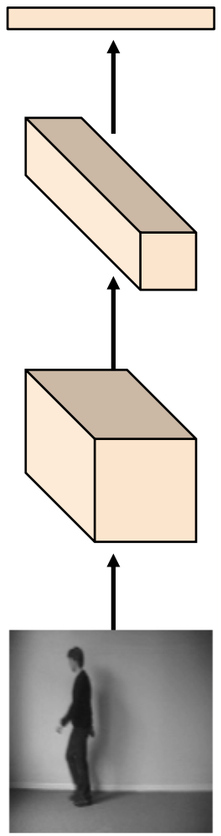
Content vector from any past frame

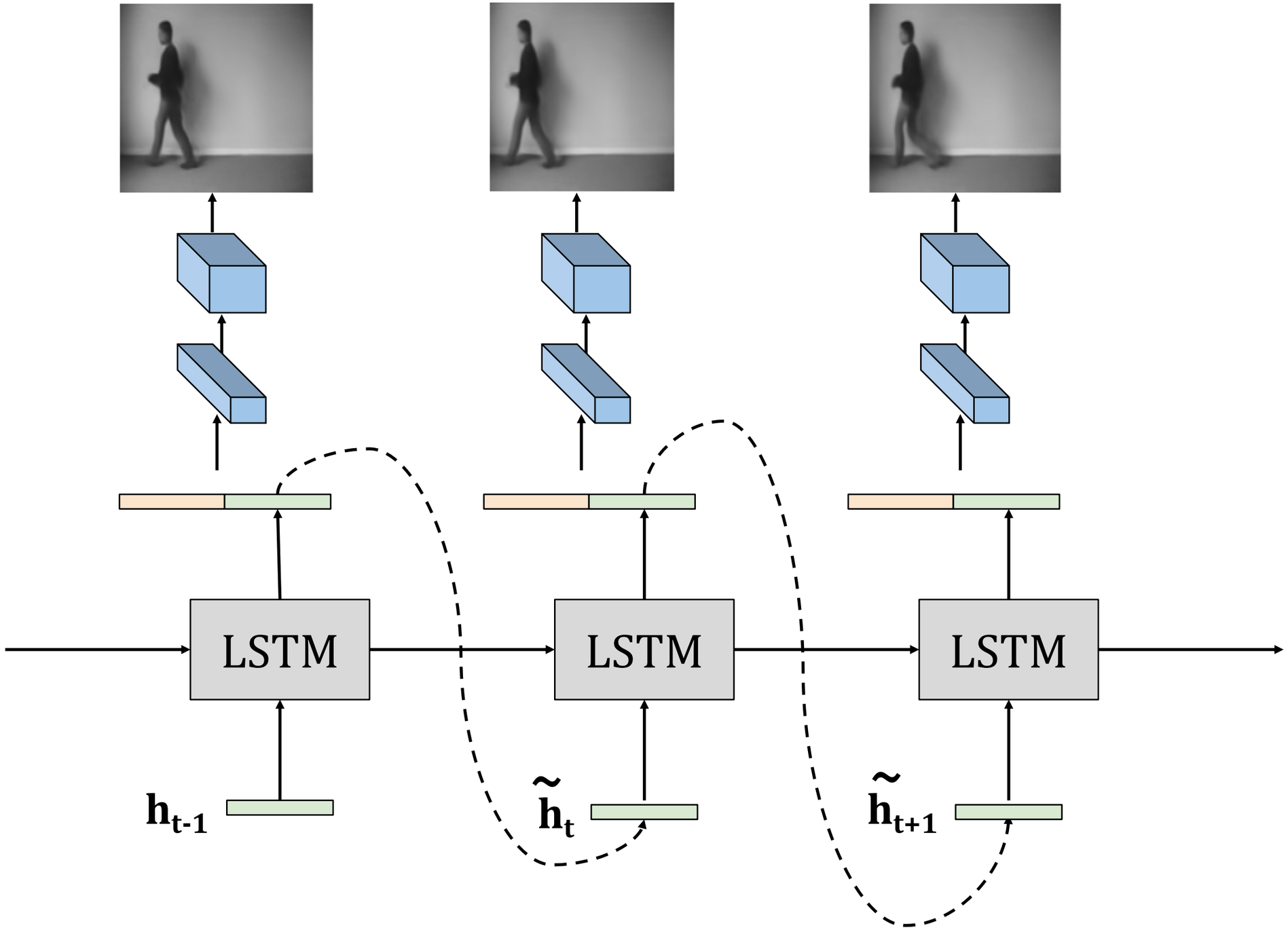
Feed predicted pose vectors back into model



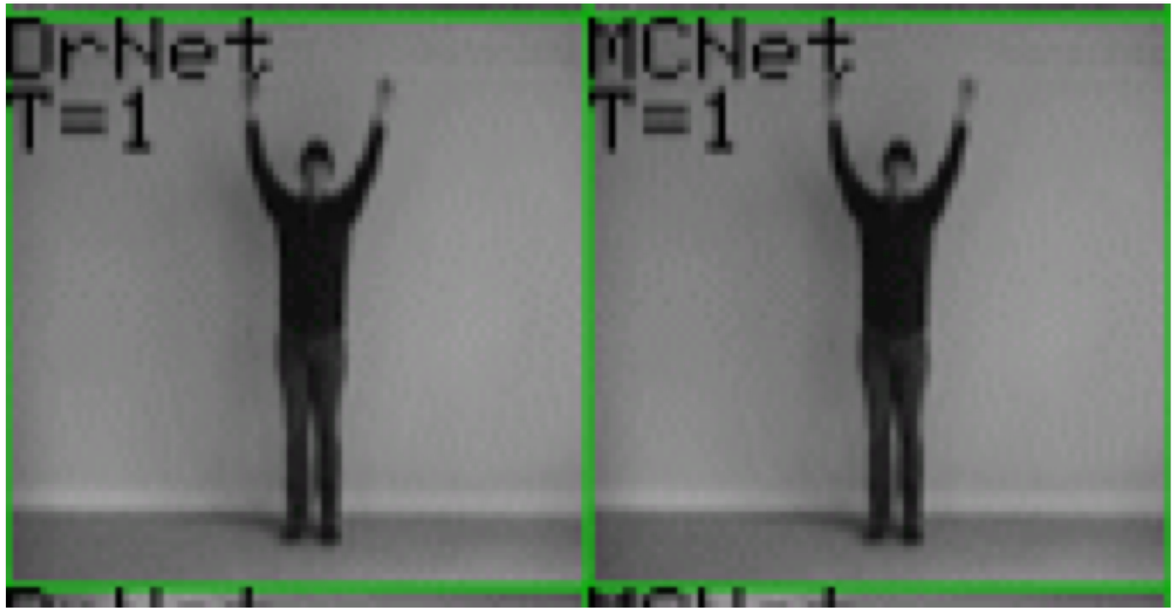


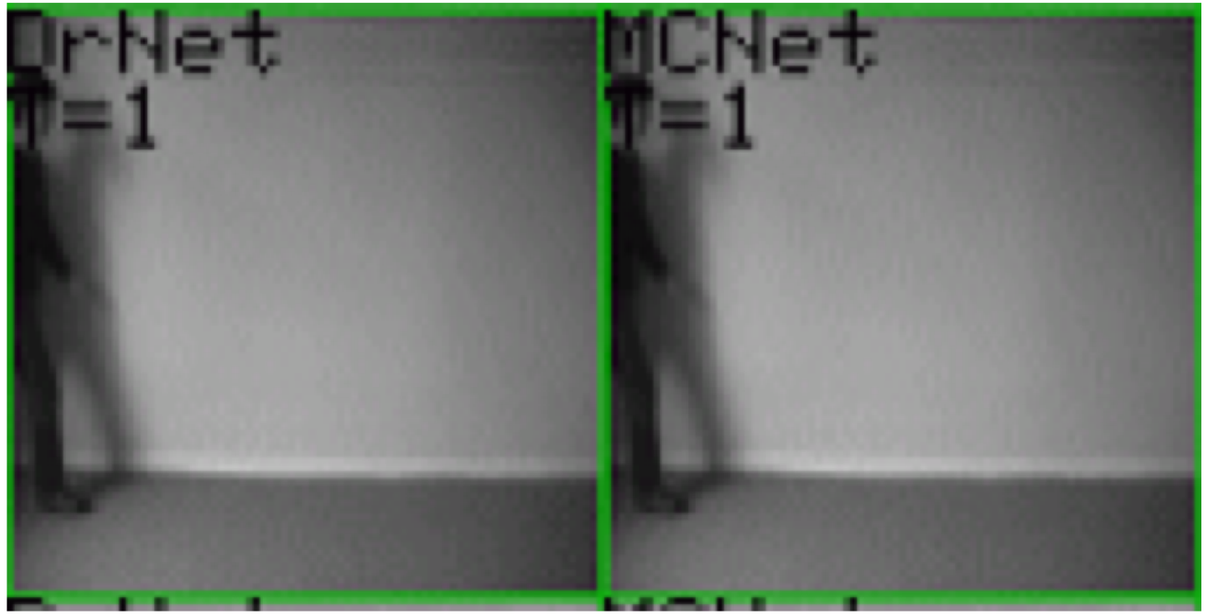
Decoder maps back to pixels:





KTH long term video generation

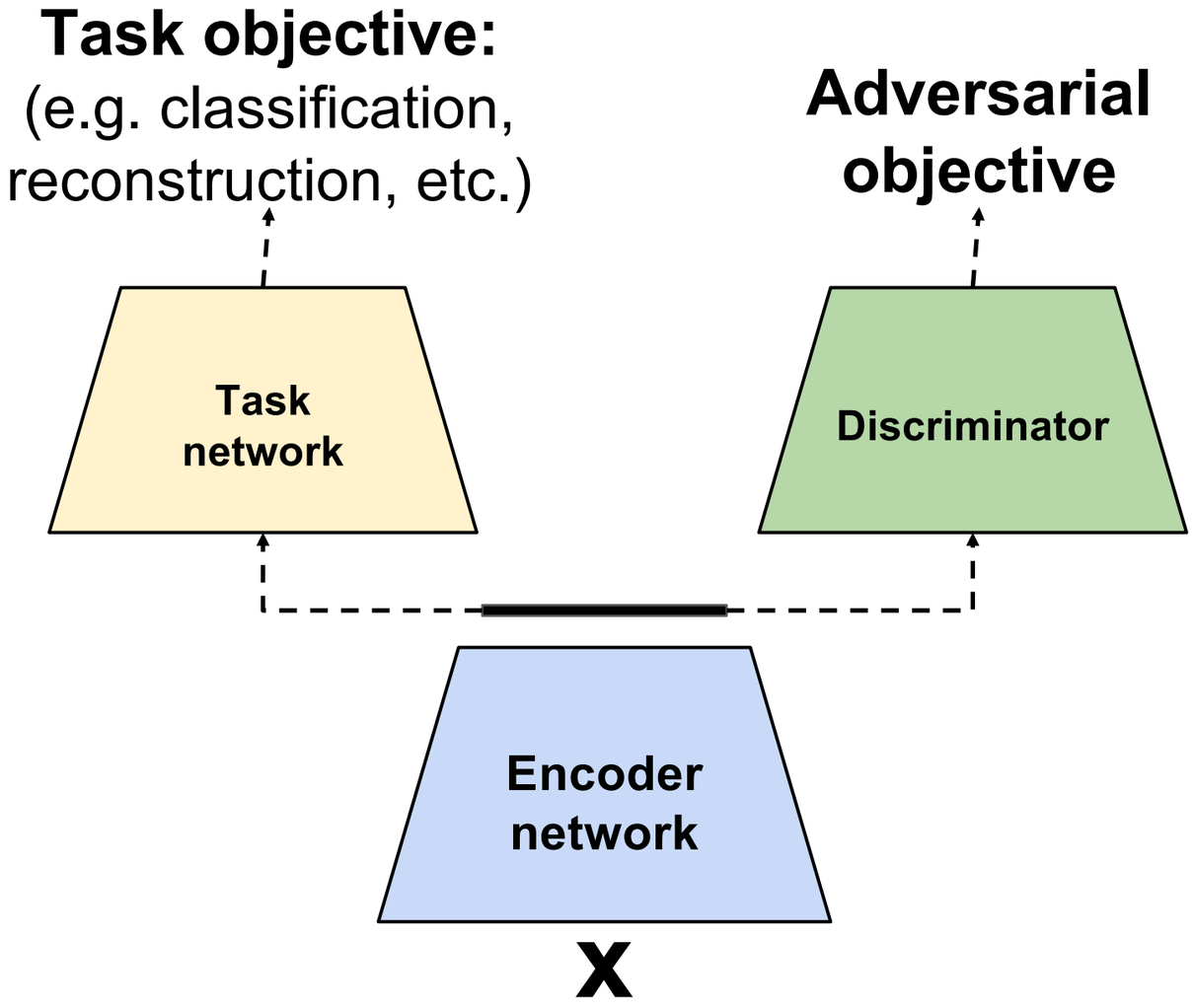




Part I: Disentangling content and pose with an adversarial loss

Denton and Birodkar. *Unsupervised Learning of Disentangled Representations from Video*. NIPS, 2017

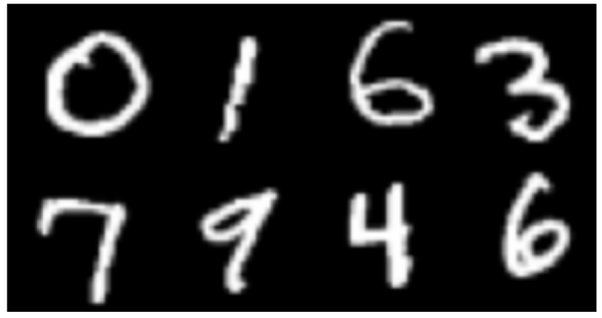
Part II: Survey of adversarial losses in feature space



Domain adaptation

Labelled examples from **source domain**,
few or no labels from **target domain**

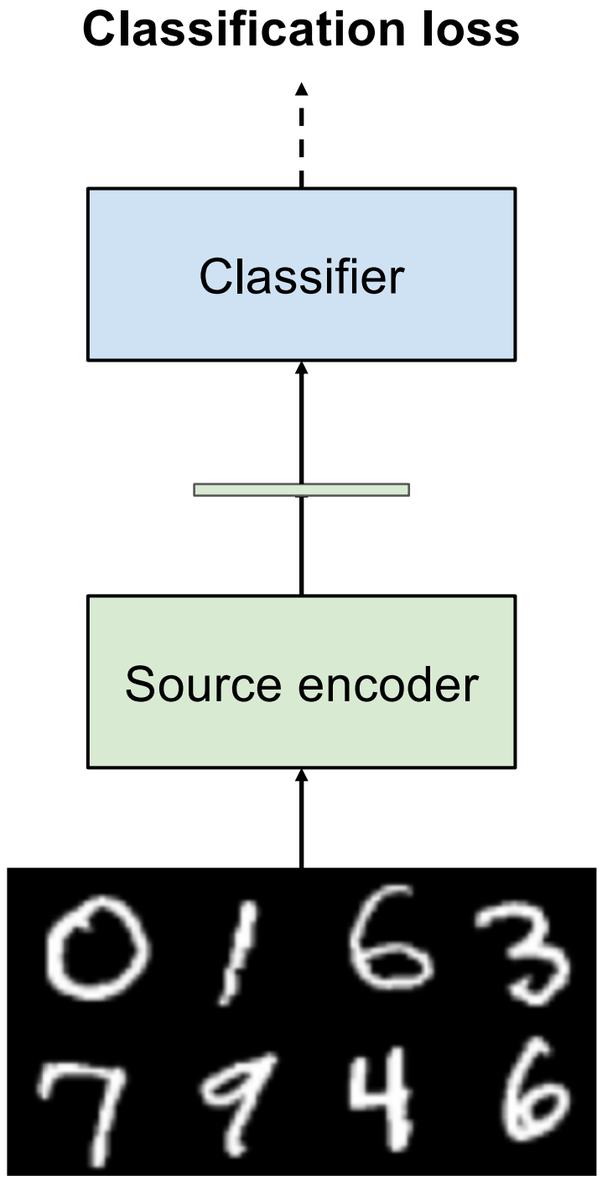
Source domain



Target domain



Domain adaptation



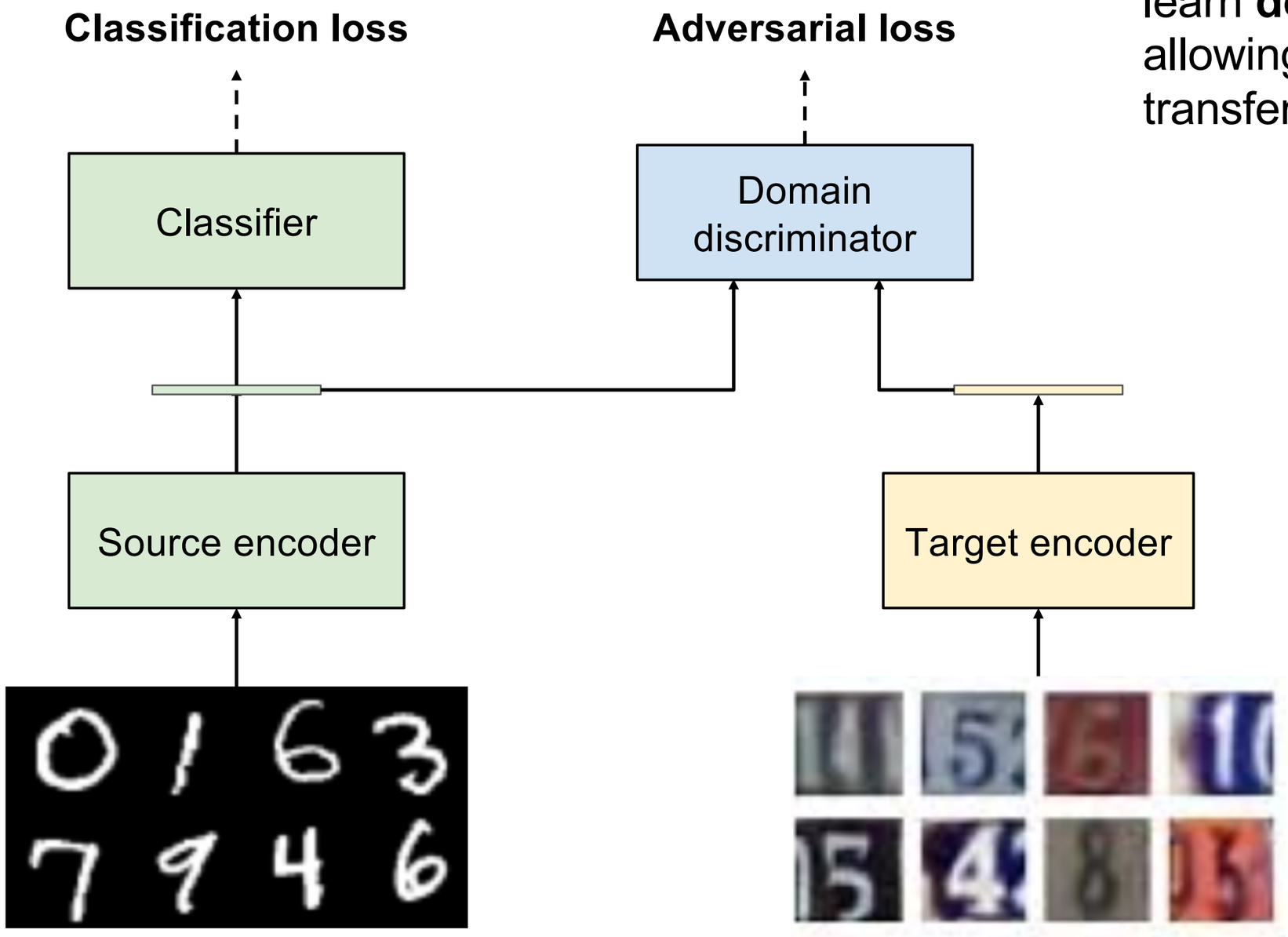
Labelled examples from **source domain**,
few or no labels from **target domain**

Target domain

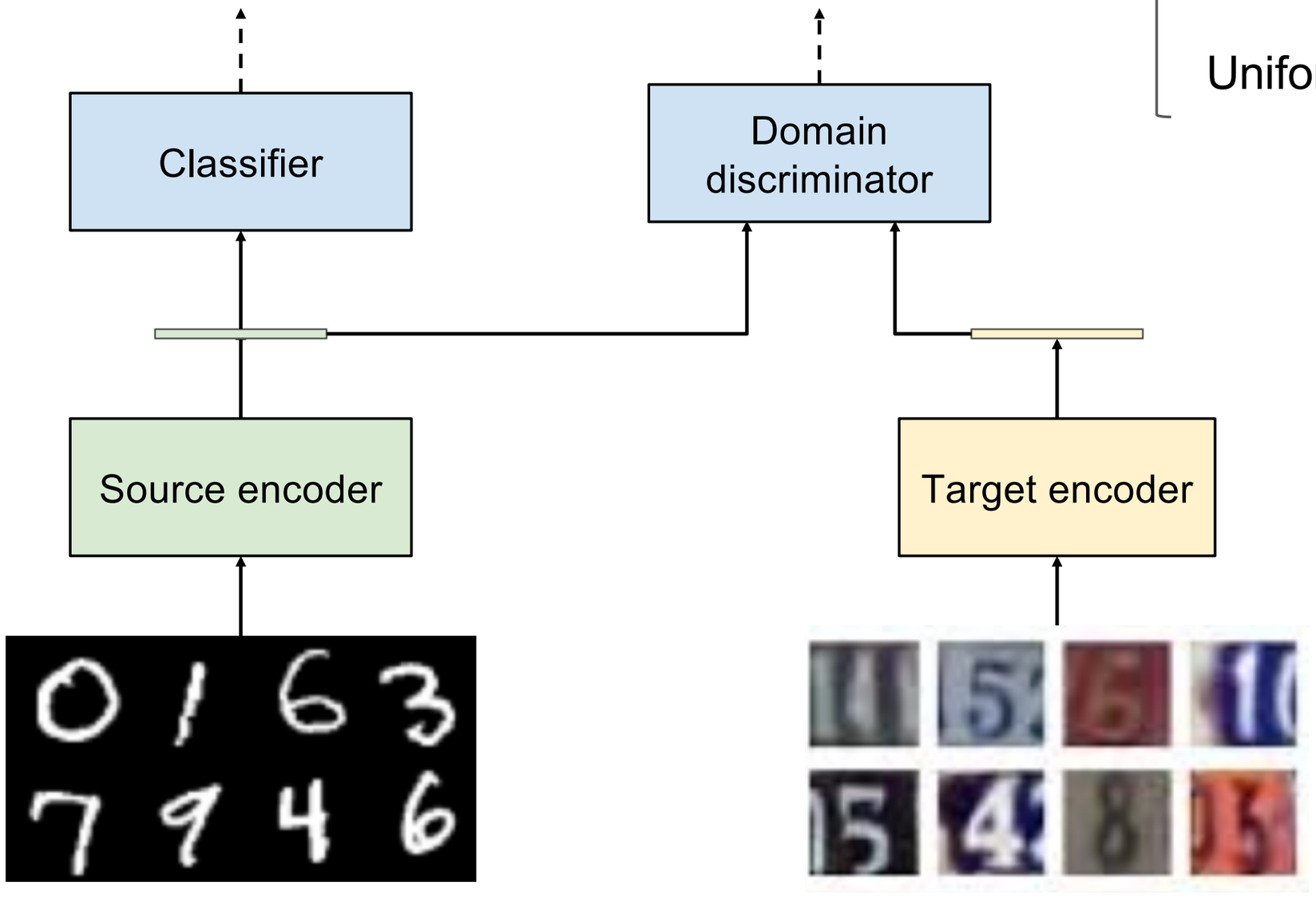


Domain adaptation

Adversarial loss can be used to learn **domain invariant features,** allowing source classifier to transfer to target domain



Domain adaptation

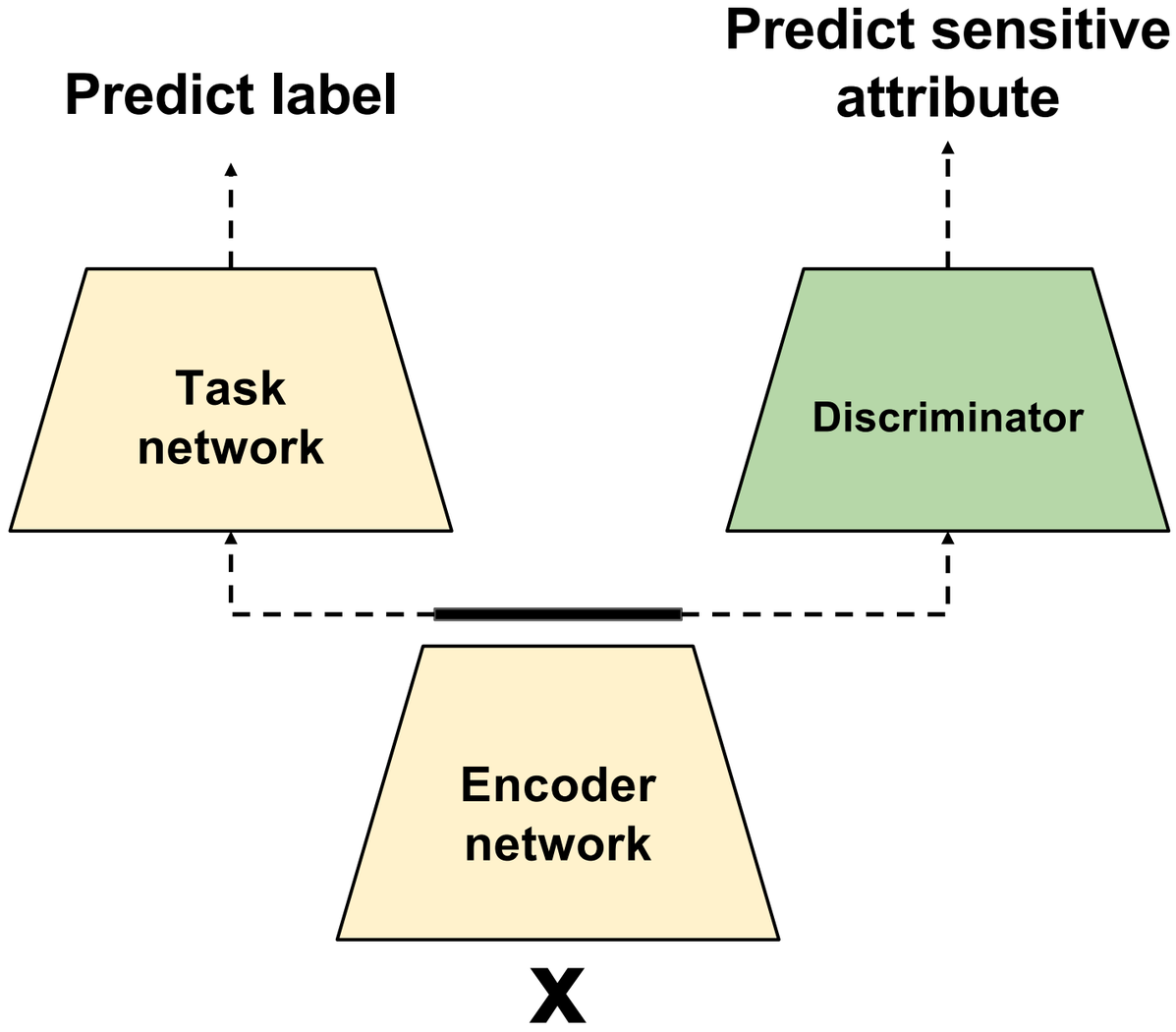


- Gradient reversal [*Ganin and Lempitsky, 2015*]
- Label flip [*Tzeng et al. 2017*]
- Uniform target [*Tzeng et al. 2015*]

Learning fair representations

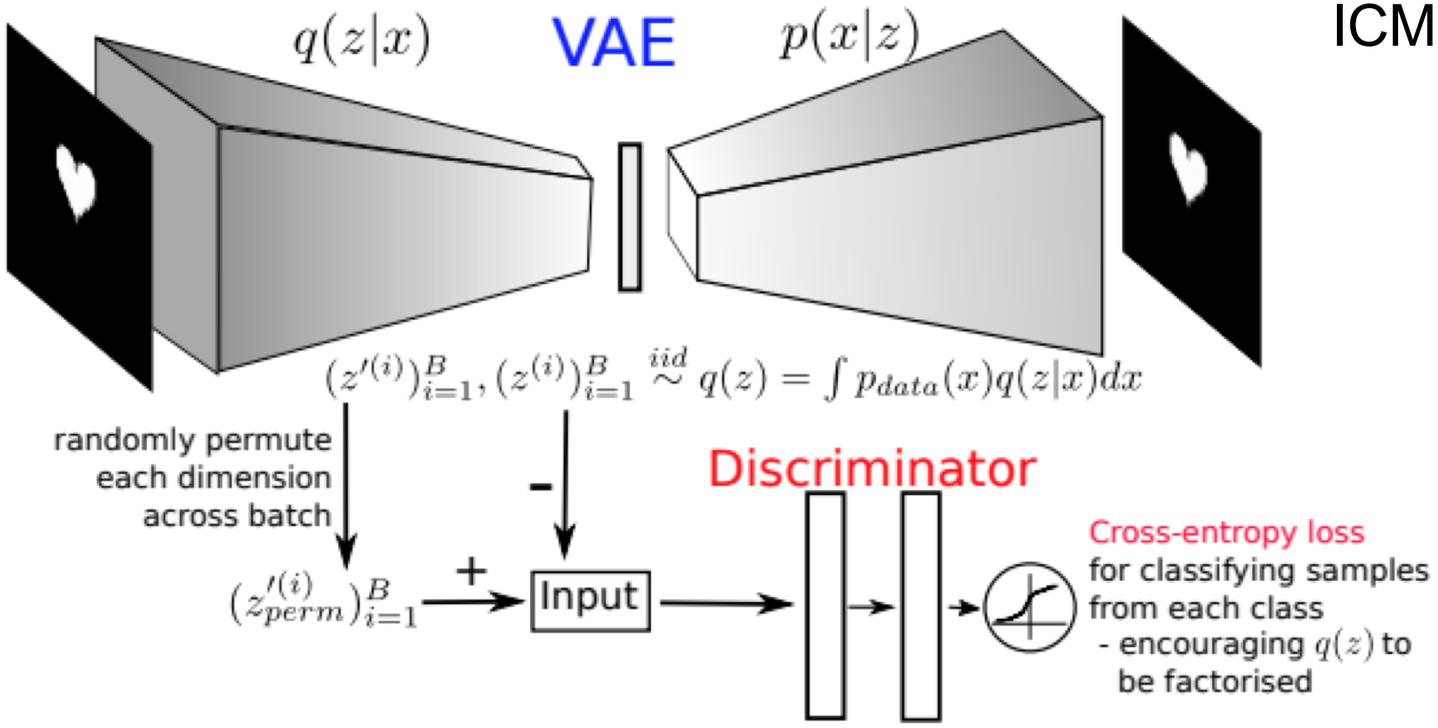
- Closely related to problem of domain adaptation
  - source/transfer domain vs. demographic groups

- Different formulations of adversarial objectives achieve different notions of fairness
  - Edwards & Storkey, 2016
  - Beutel et al. 2017
  - Zhang et al. 2018
  - Madras et al. 2018



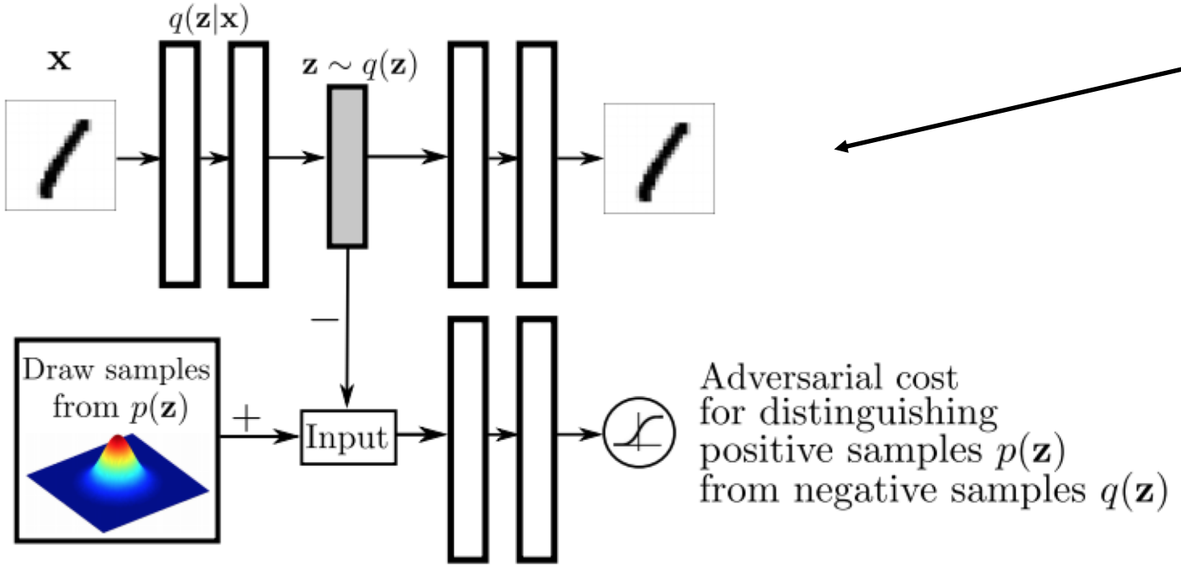
Independent components

Kim and Mnih. Disentangling by Factorising. ICML, 2018



- Discriminate marginal distribution vs. product of marginals: $q(z_1, ..., z_n)$ vs. $\prod q(z_i)$
- Earlier work on discrete code setting by Schmidhuber (1992)

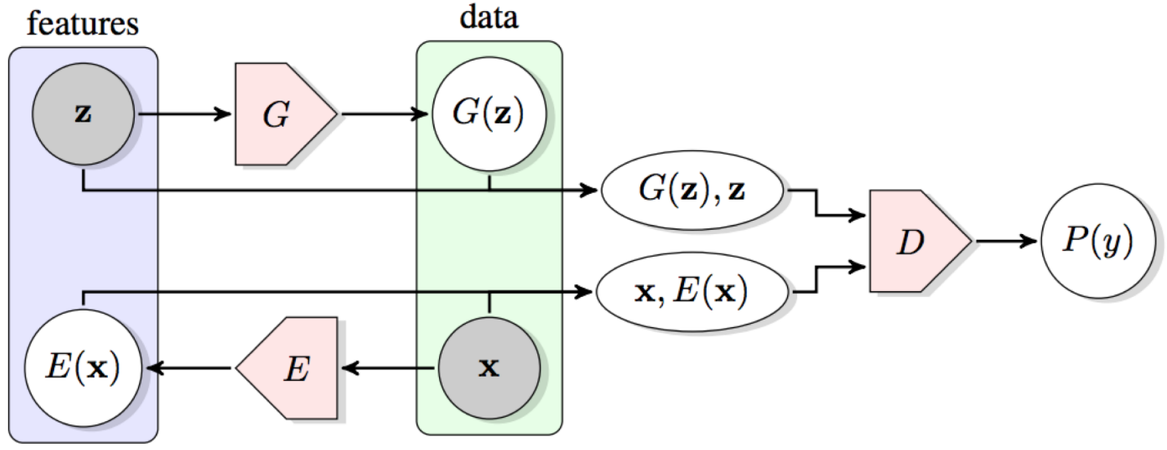
Prior distributions of generative models



Adversarial autoencoders:
Match aggregate approx posterior q(z)
[Makhzani et al. 2016]

Adversarial variational bayes:
Match approx posterior q(z|x)
[Mescheder et al. 2017]

Adversarial feature learning:
GAN loss in image space and latent space
[Dumoulin et al. 2017; Donahue et al. 2017]



References

Beutel et al. *Data decisions and theoretical implications when adversarially learning fair representations*. arXiv:1707.00075, 2017.

Denton and Birodkar. *Unsupervised Learning of Disentangled Representations from Video*. NIPS, 2017.

Donahue et al. *Adversarial Feature Learning*. ICLR, 2017.

Dumoulin et al. *Adversarially Learned Inference.* ICLR, 2017

Edwards & Storkey. *Censoring Representations with an Adversary*. ICLR, 2016.

Ganin and Lempitsky. *Unsupervised domain adaptation by backpropagation.* ICML, 2015.

Kim and Mnih. *Disentangling by Factorising*. ICML, 2018.

Madras et al. *Learning Adversarially Fair and Transferable Representations*. ICML, 2018.

Makhzani et al. *Adversarial Autoencoders*. ICLR Workshop, 2016.

Mescheder et al. *Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks*. ICML, 2017.

Schmidhuber. *Learning factorial codes by predictability minimization.* Neural Computation, 1992.

Tzeng et al. S*imultaneous deep transfer across domains and tasks.* ICCV, 2015.

Tzeng et al. *Adversarial discriminative domain adaptation.* CVPR, 2017.

Villegas, et al. D*ecomposing motion and content for natural video sequence prediction.* In ICLR, 2017.

Zhang et al. M*itigating Unwanted Biases with Adversarial Learning*. AIES, 2018.