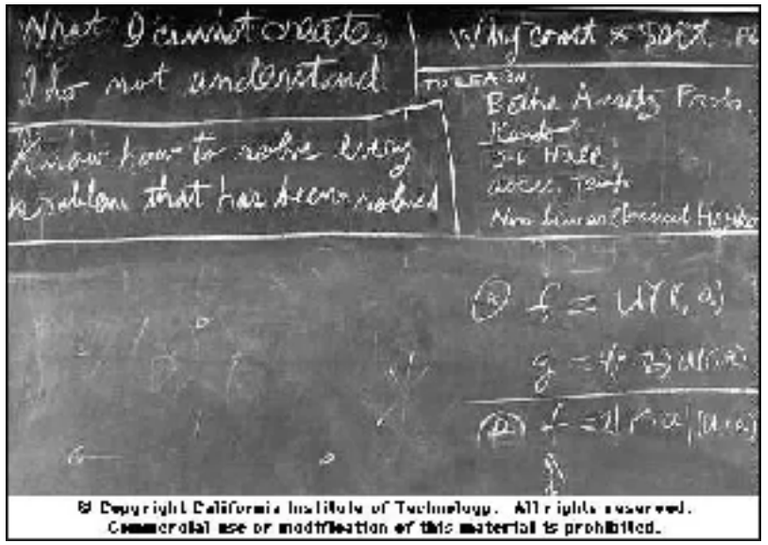
Recent Trends in Deep Learning

Sungjoon Choi
CPSLAB, SNU

CPSLAB
http://cpslab.snu.ac.kr

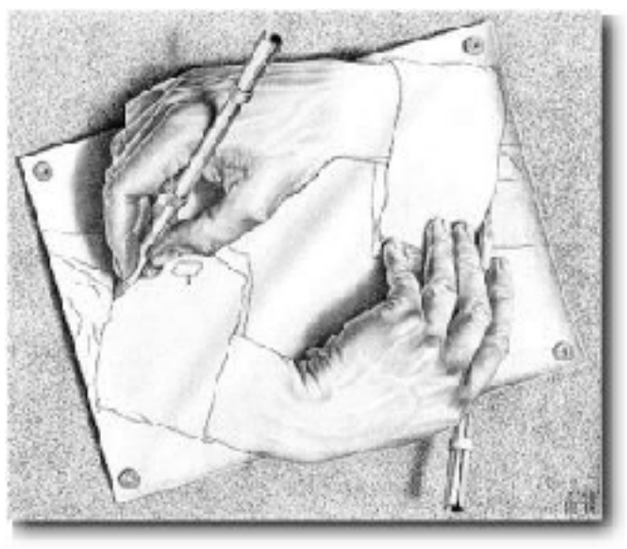
Recent Trends in Deep Learning

Generative Model (14)

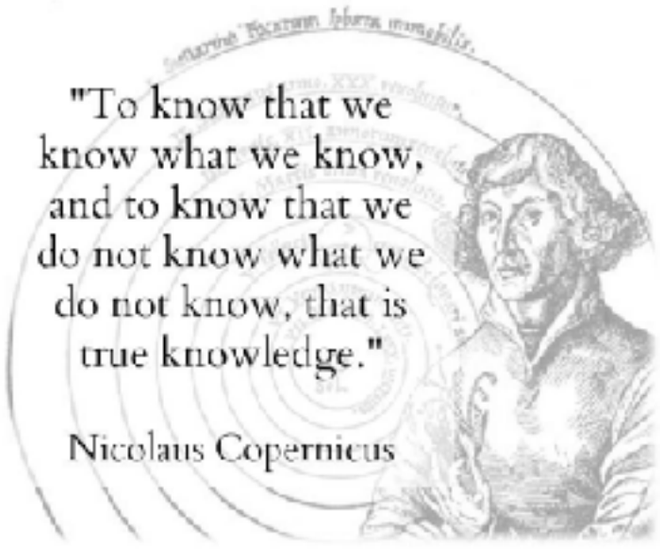


Domain Adaptation (7)

Meta Learning (8)



Uncertainty in Deep Learning (6)

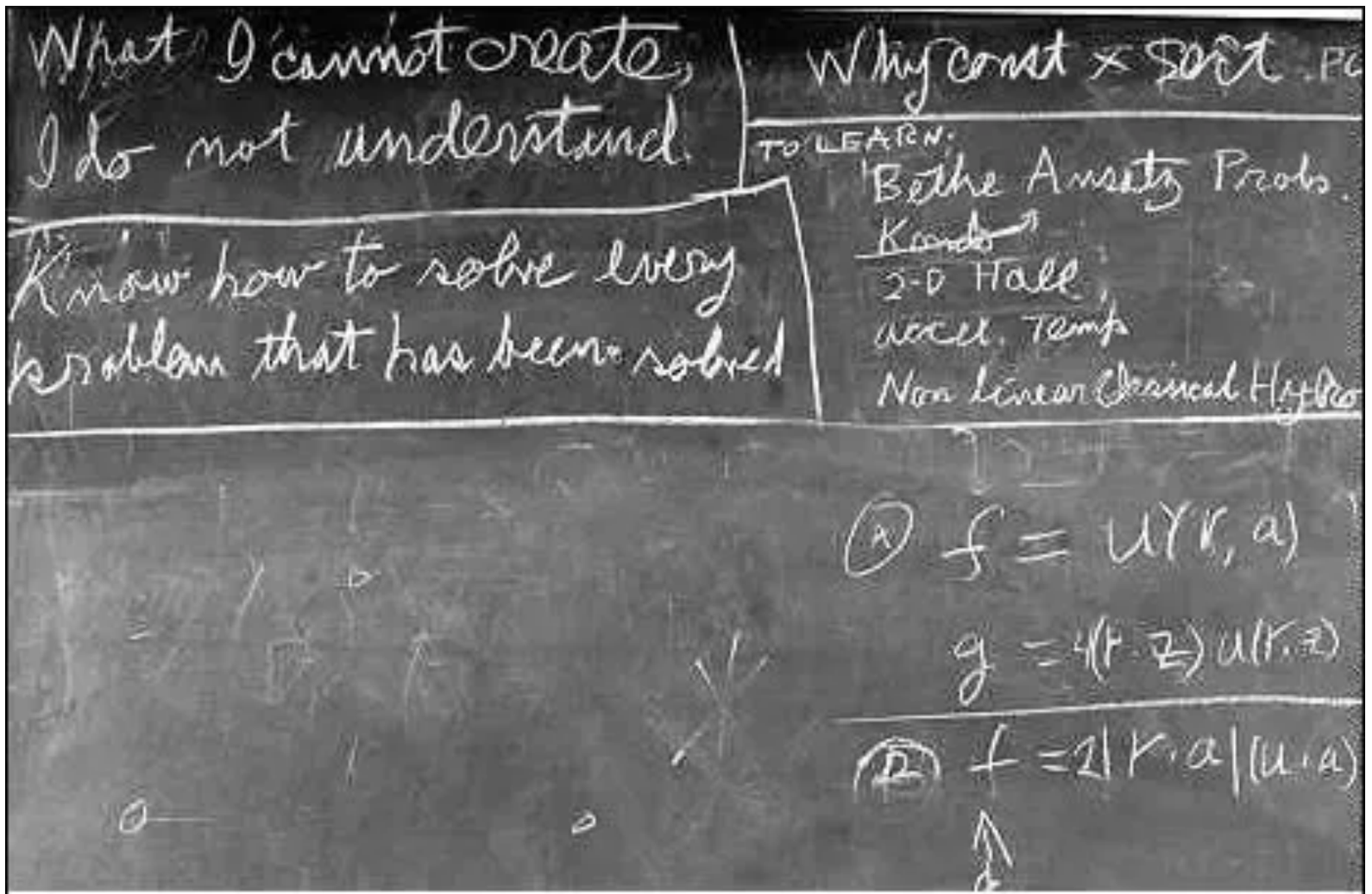


CPSLAB
http://cpslab.snu.ac.kr

Generative Model

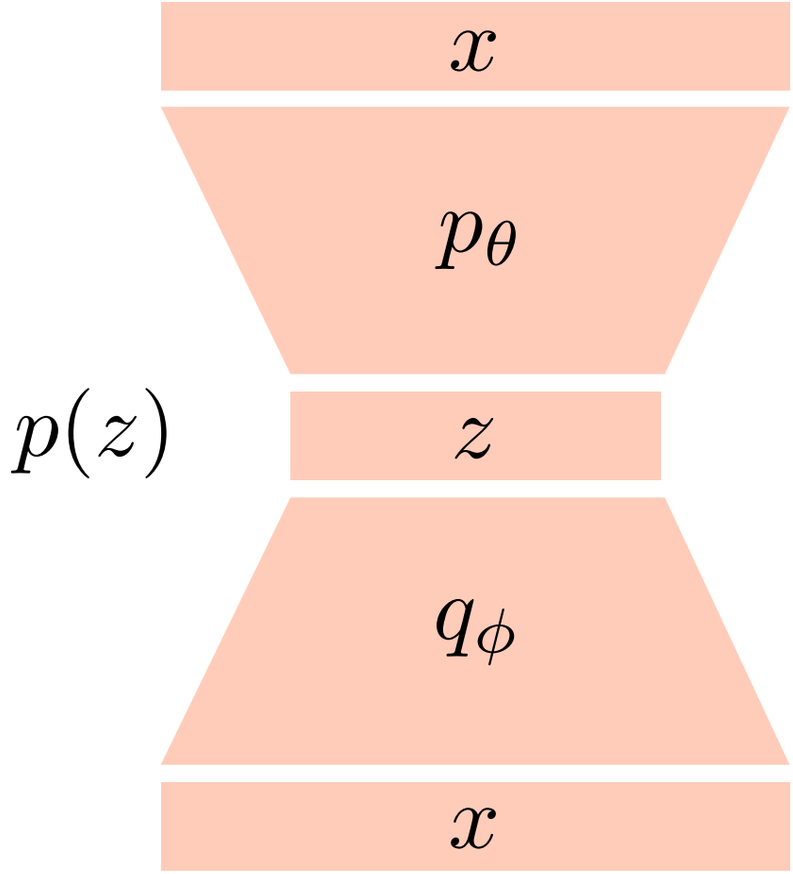
Generative Model

What I cannot create, I do not understand

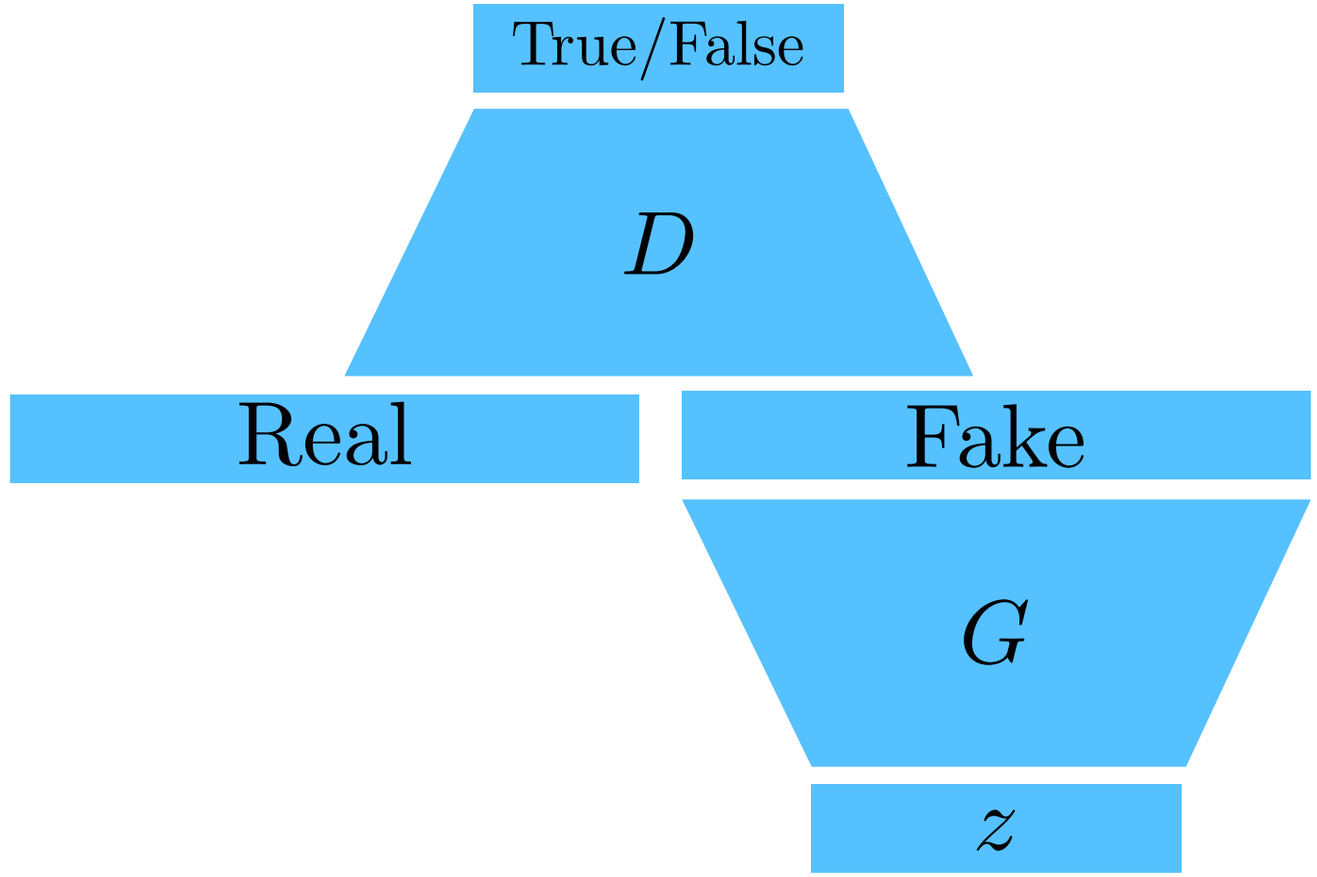


Generative Model

Generative Adversarial Nets vs. Variational Autoencoders



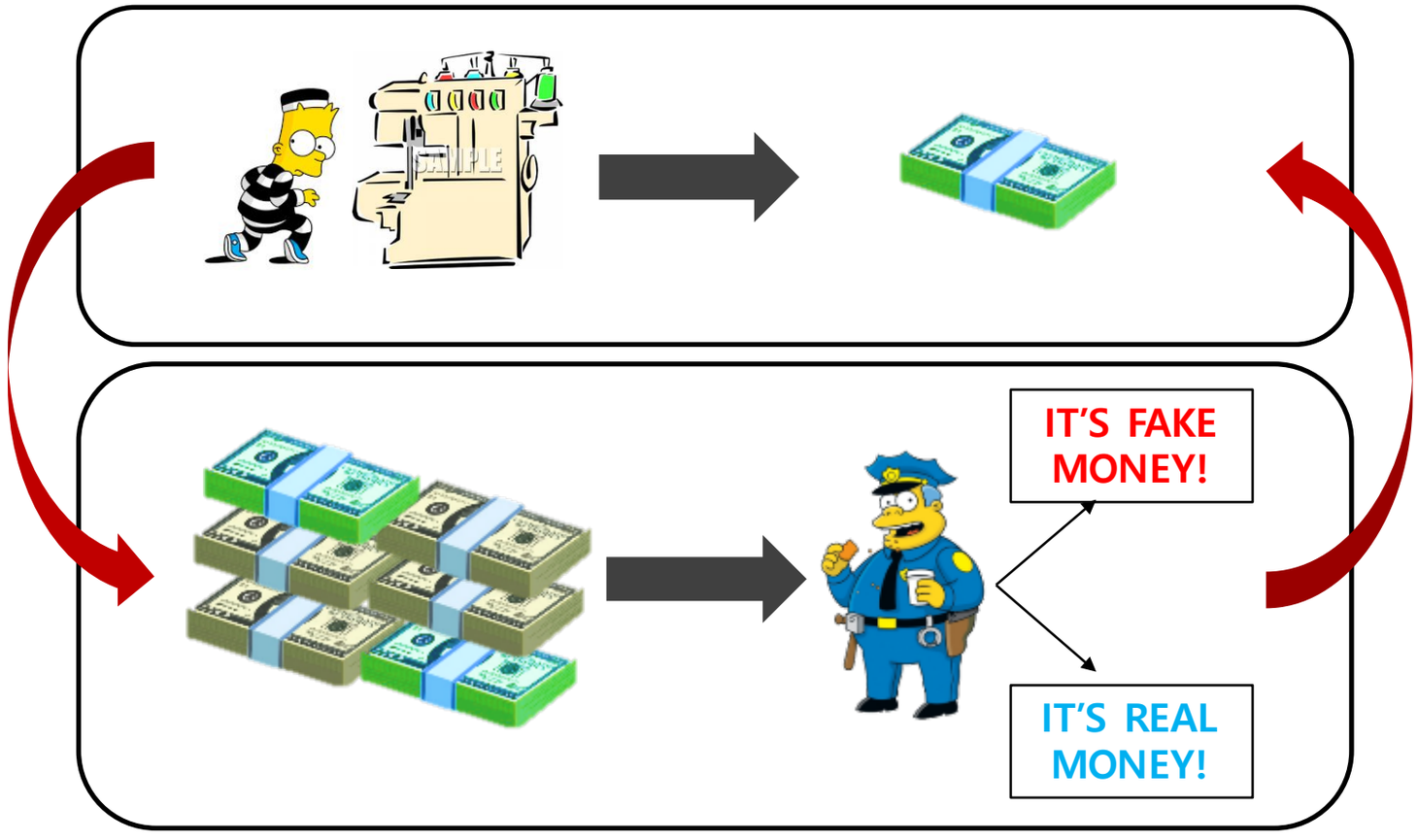
Variational Autoencoder



Generative Adversarial Networks

Generative Model

Generative Adversarial Network (GAN)

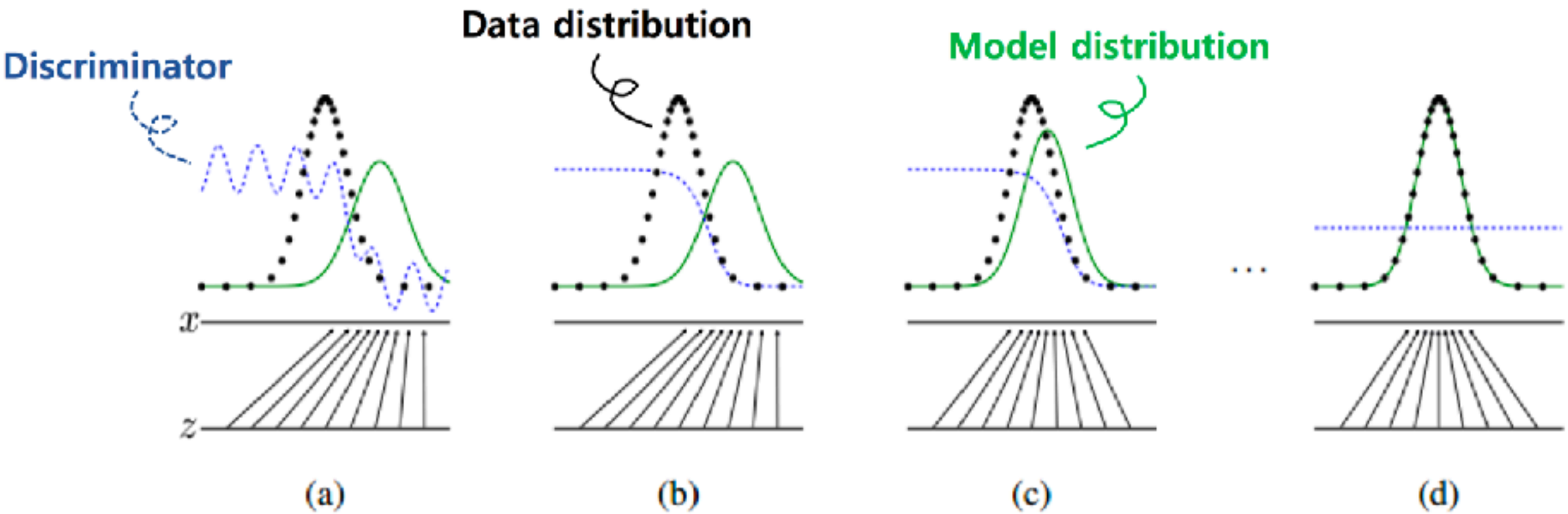


$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

https://arxiv.org/abs/1406.2661

Generative Model

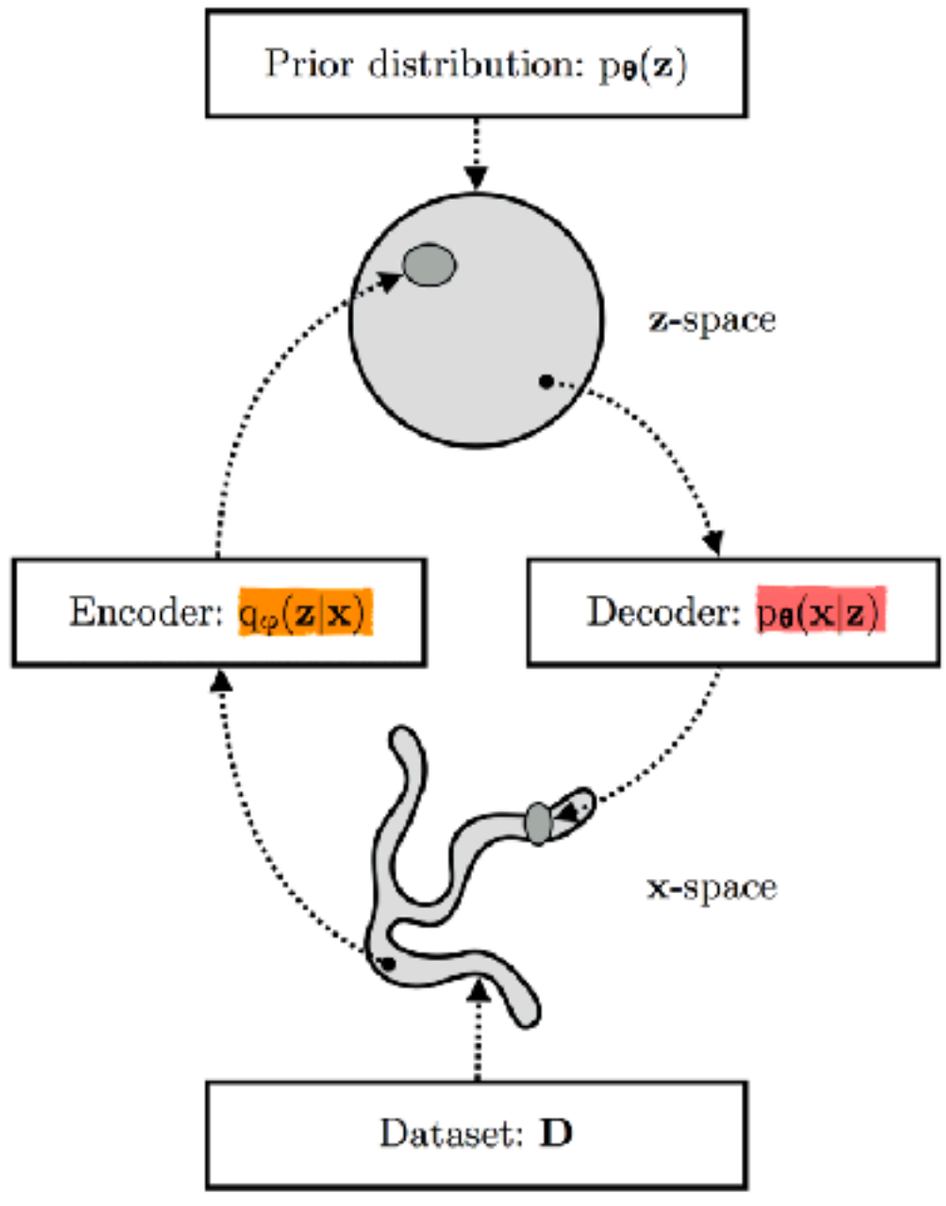
Generative Adversarial Network (GAN)



https://arxiv.org/abs/1406.2661

Generative Model

Variational Autoencoder (VAE)



Kingma's Thesis

Generative Model

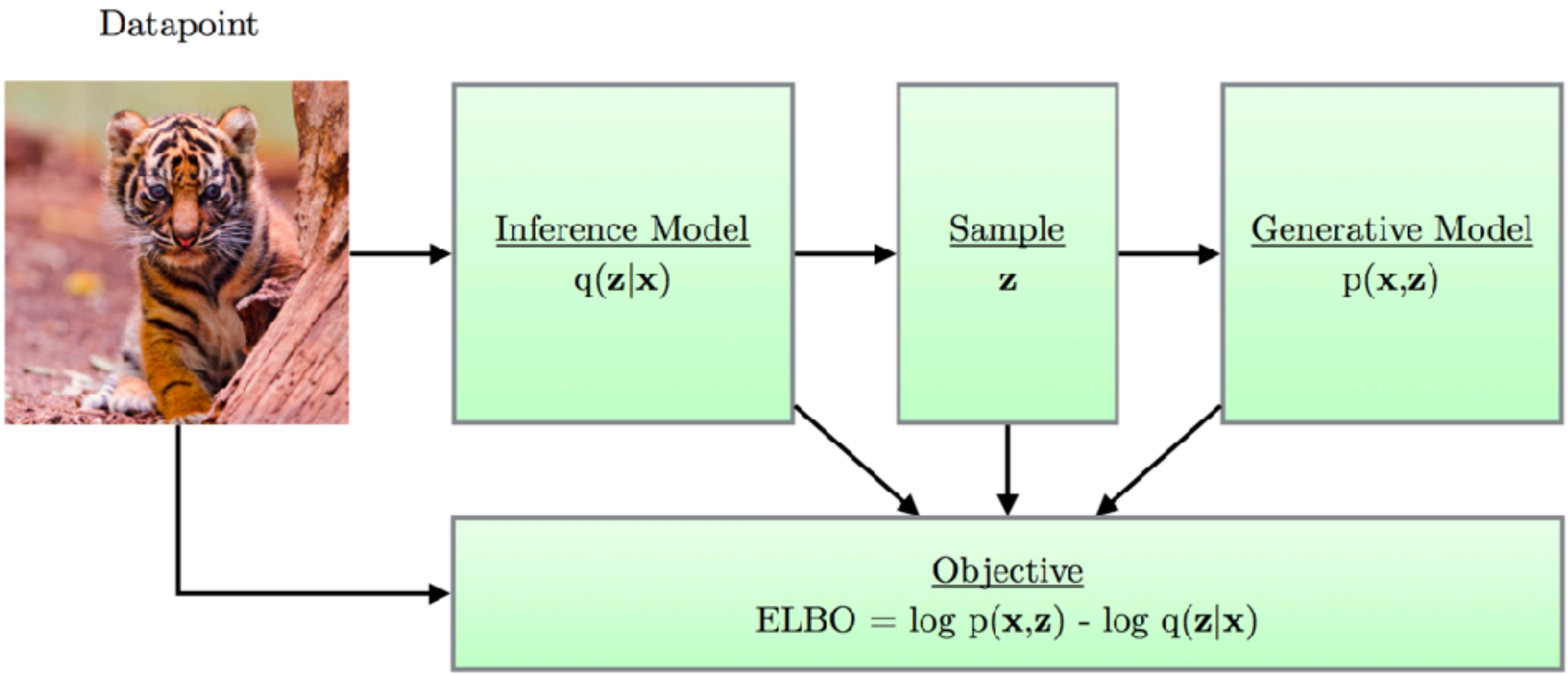
Variational Autoencoder (VAE)

$$
\begin{aligned}
\log p_{\boldsymbol{\theta}}(\mathbf{x}) &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x})\right] \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\right]\right] \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\right]\right] \\
&= \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\right]\right]}_{=\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x})\ \text{(ELBO)}} + \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\right]\right]}_{=D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}))}
\end{aligned}
$$

Kingma’s Thesis

Generative Model

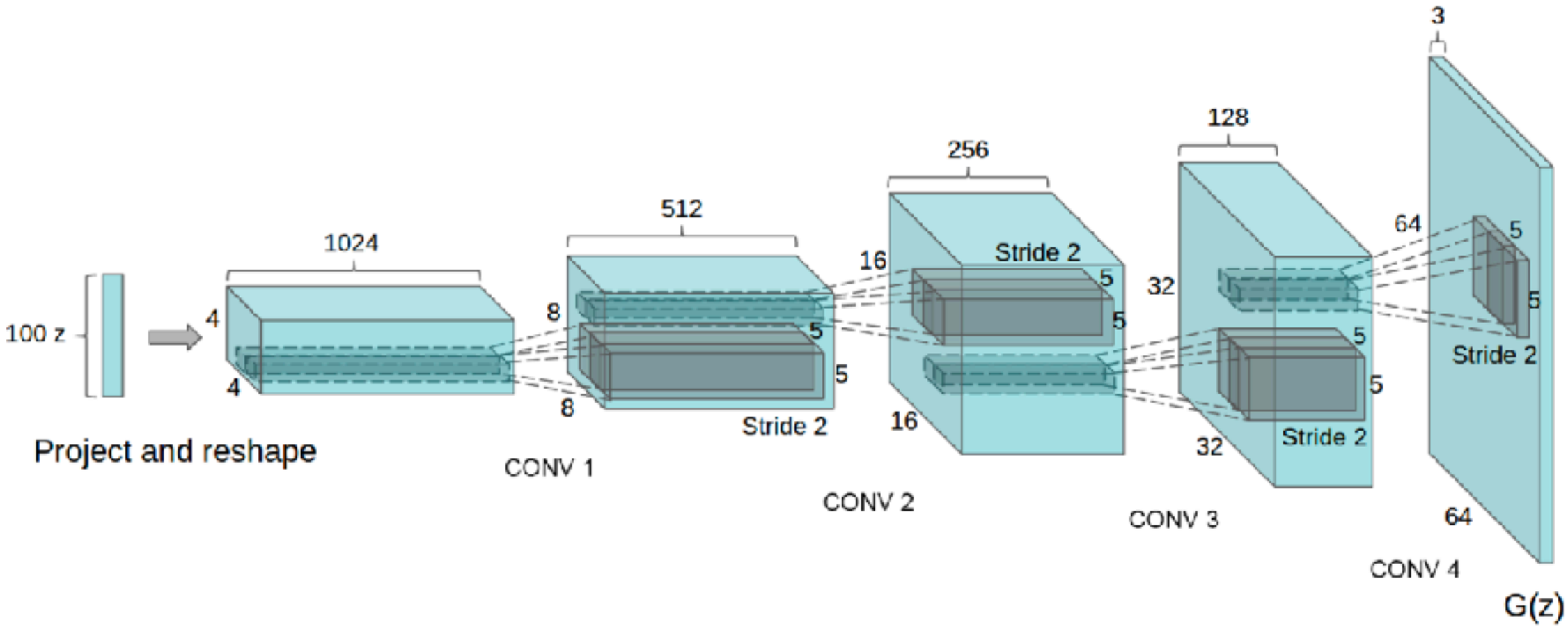
Variational Autoencoder (VAE)



Kingma's Thesis

Generative Model

DCGAN

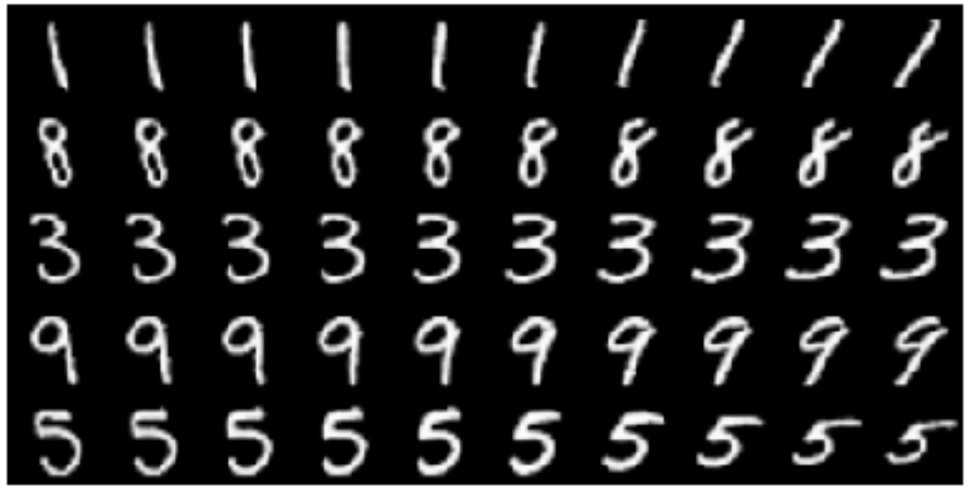


https://arxiv.org/abs/1511.06434

Generative Model

Info GAN

(a) Varying c_1 on InfoGAN (Digit type)

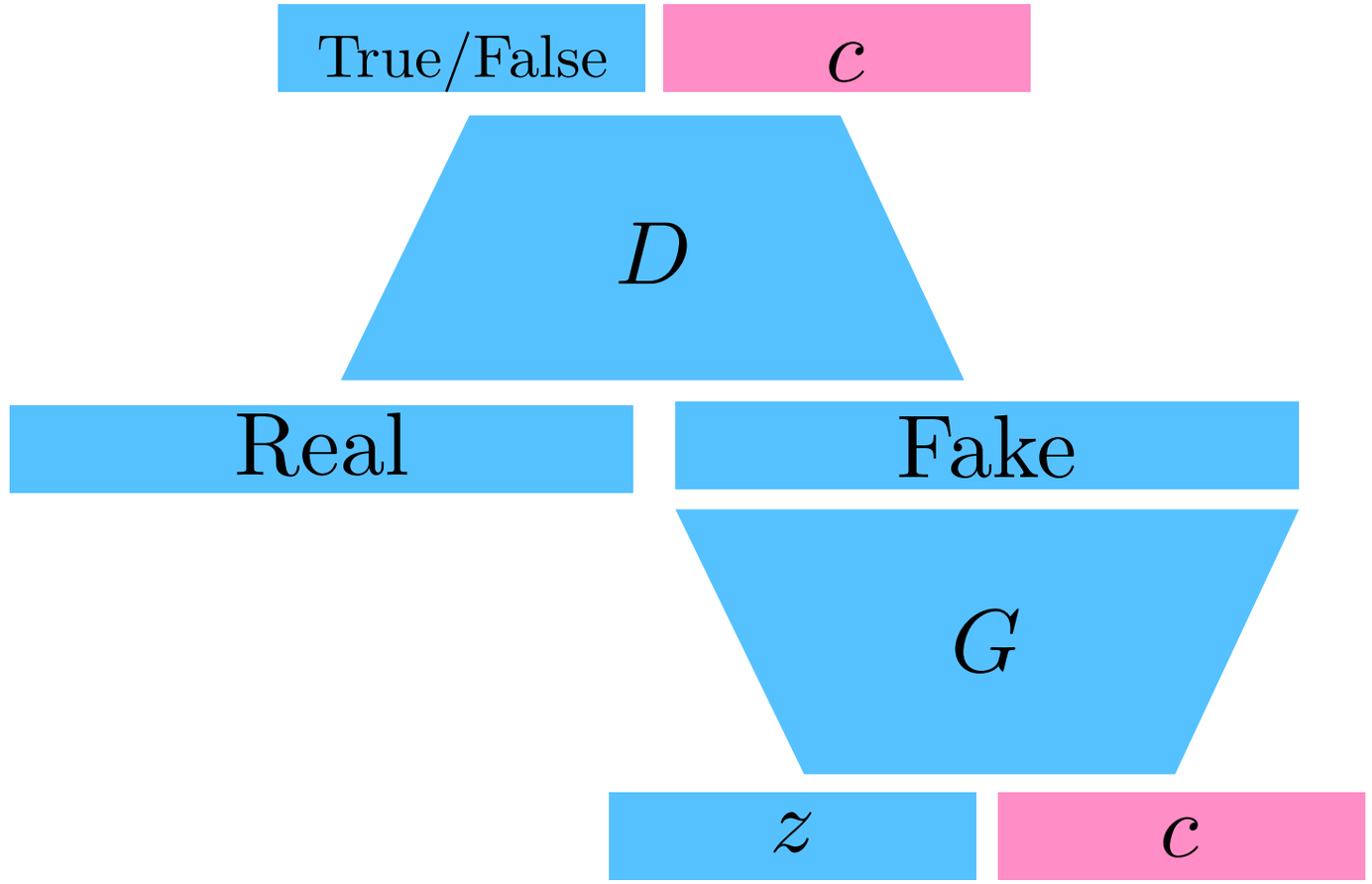


(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation)

https://arxiv.org/abs/1606.03657

Generative Model

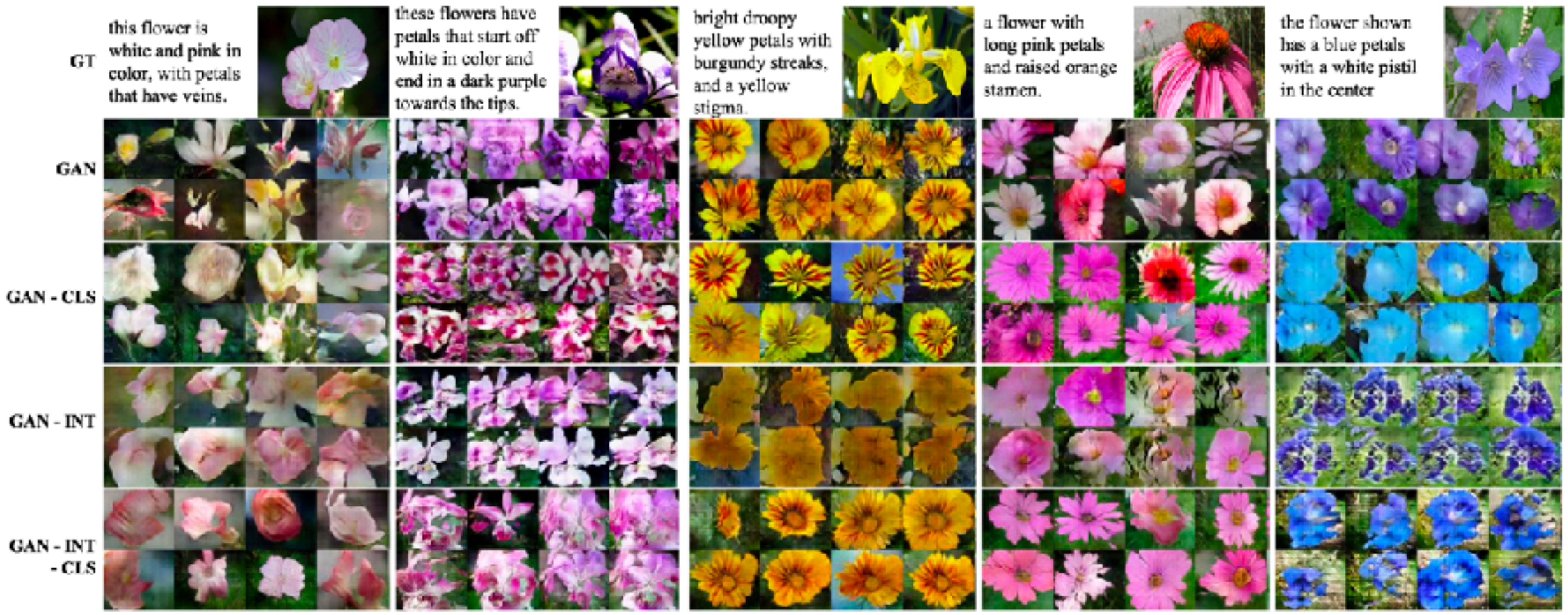
Info GAN



https://arxiv.org/abs/1606.03657

Generative Model

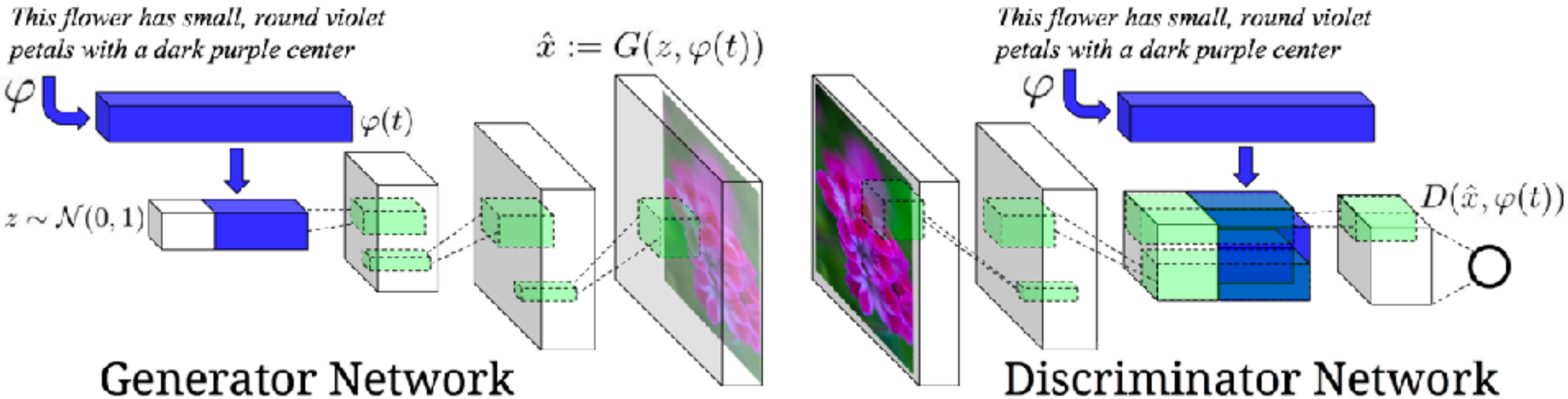
Text2Image



https://arxiv.org/abs/1605.05396

Generative Model

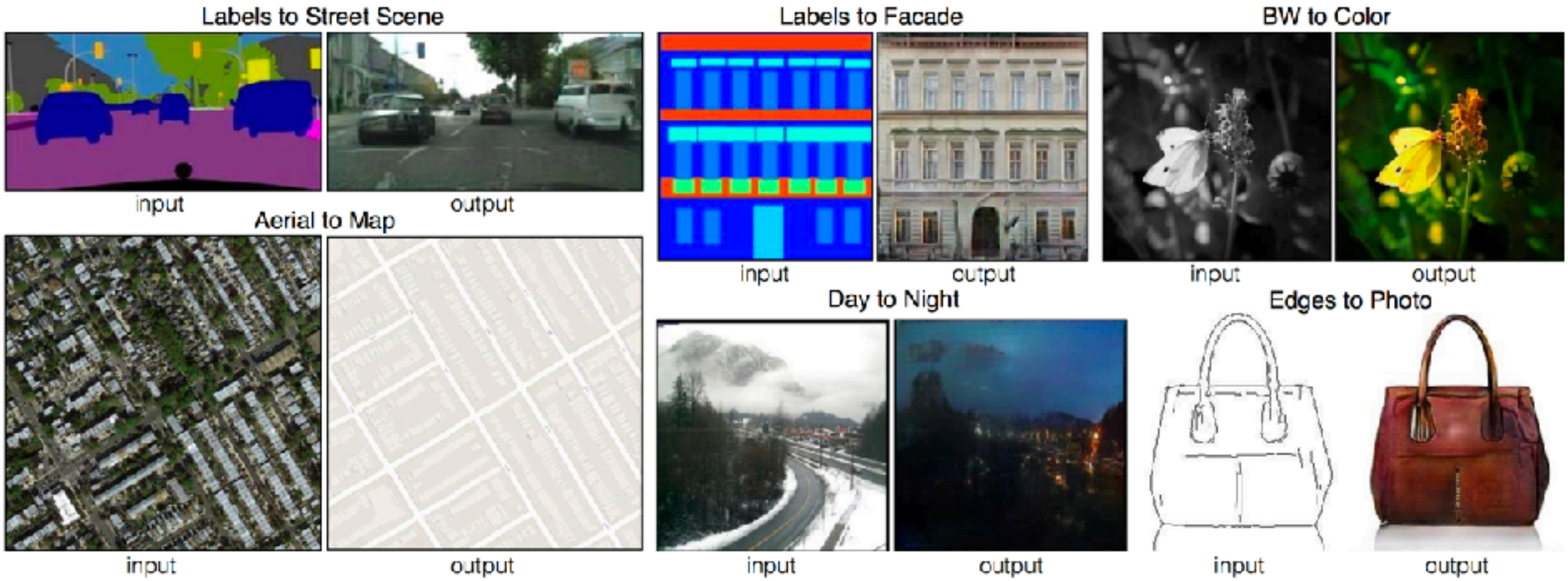
Text2Image



https://arxiv.org/abs/1605.05396

Generative Model

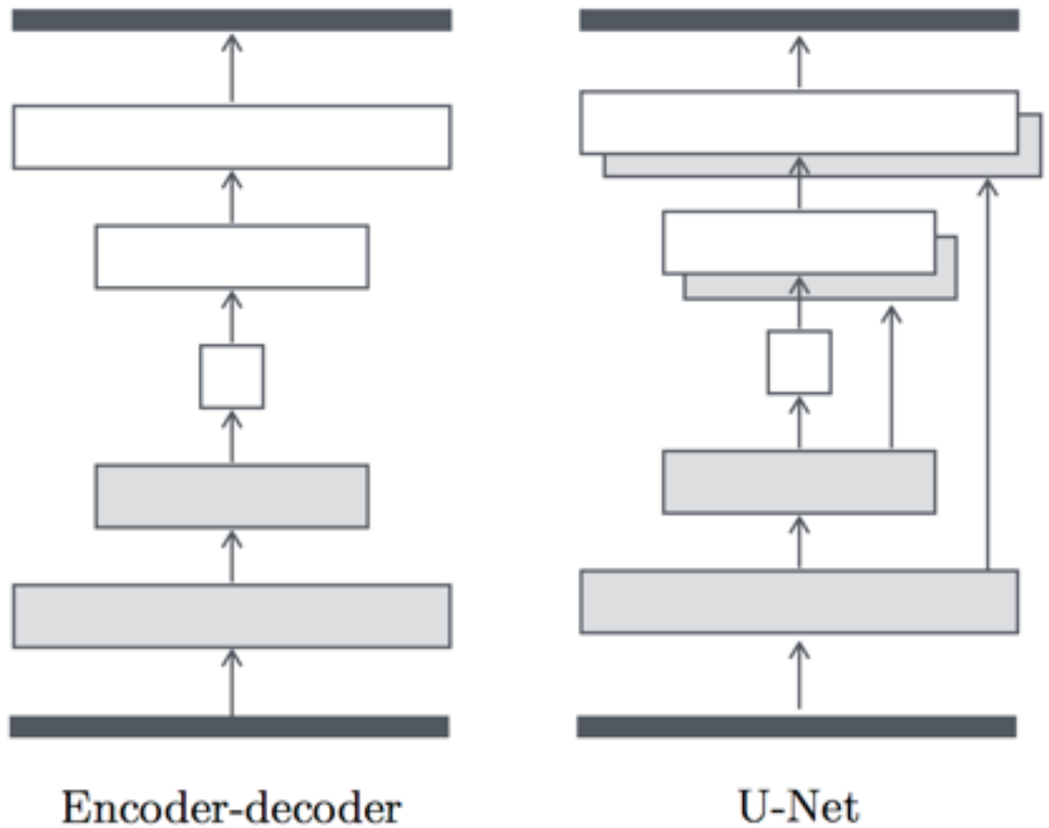
Pix2pix



https://arxiv.org/abs/1611.07004

Generative Model

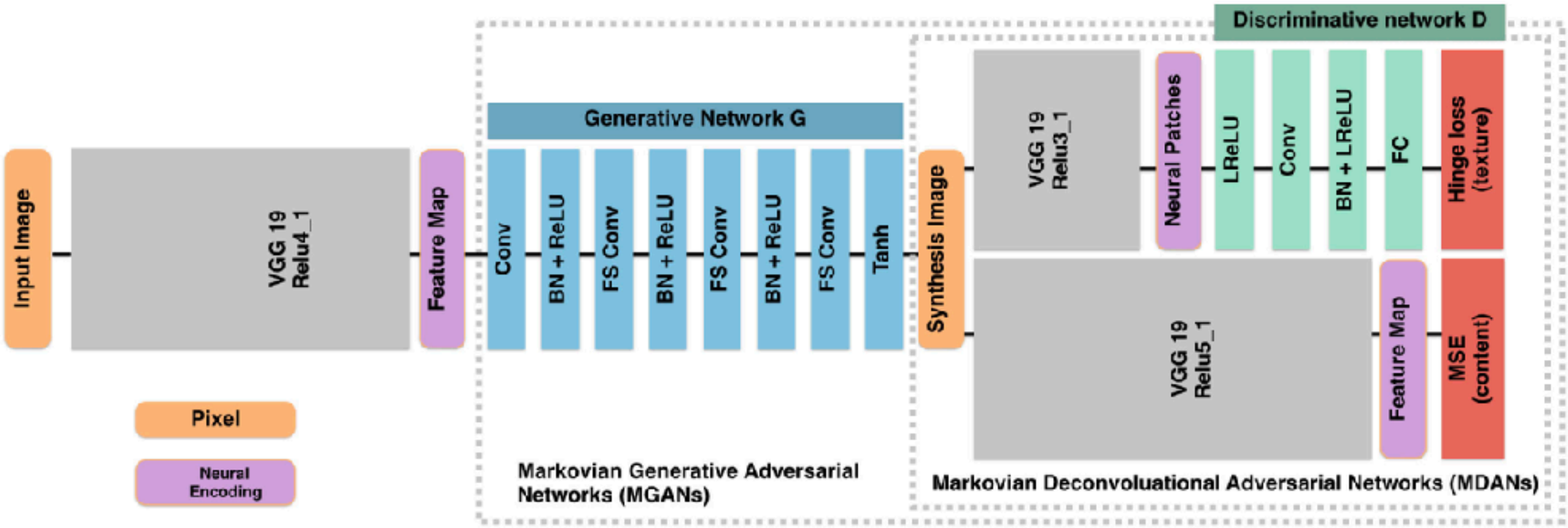
Pix2pix



https://arxiv.org/abs/1611.07004

Generative Model

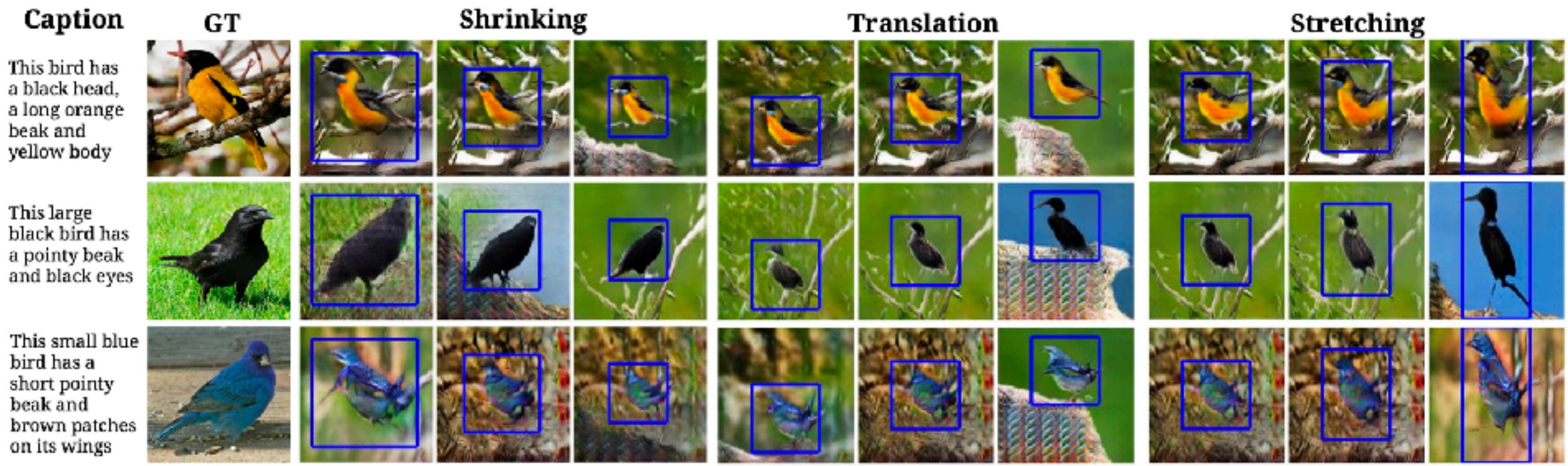
PatchGAN



https://arxiv.org/abs/1604.04382

Generative Model

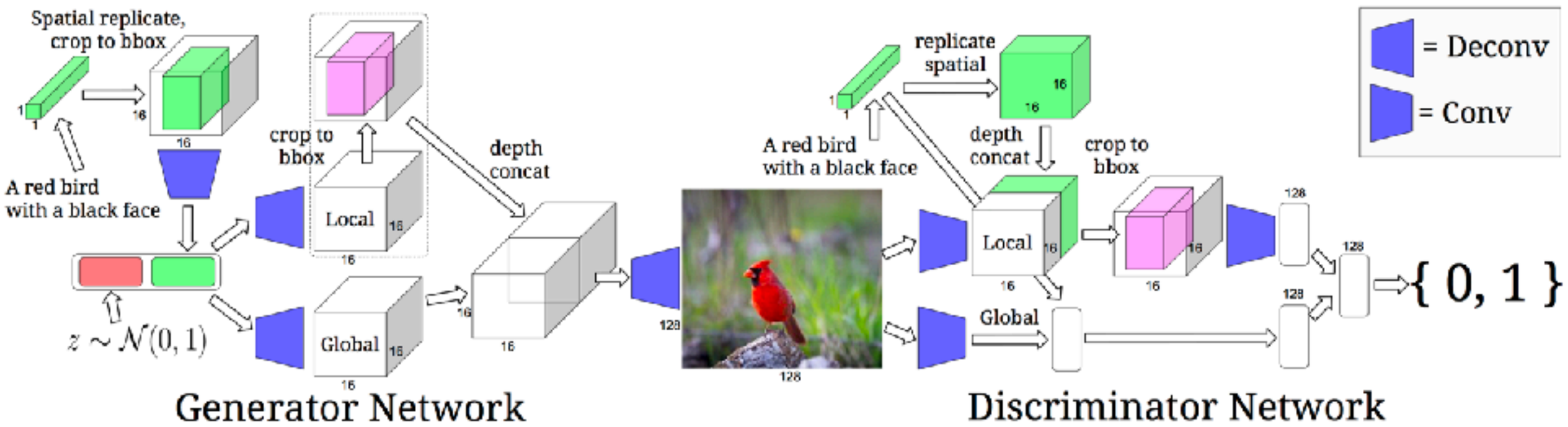
Generative Adversarial What-Where Networks (GAWWN)



https://arxiv.org/abs/1610.02454

Generative Model

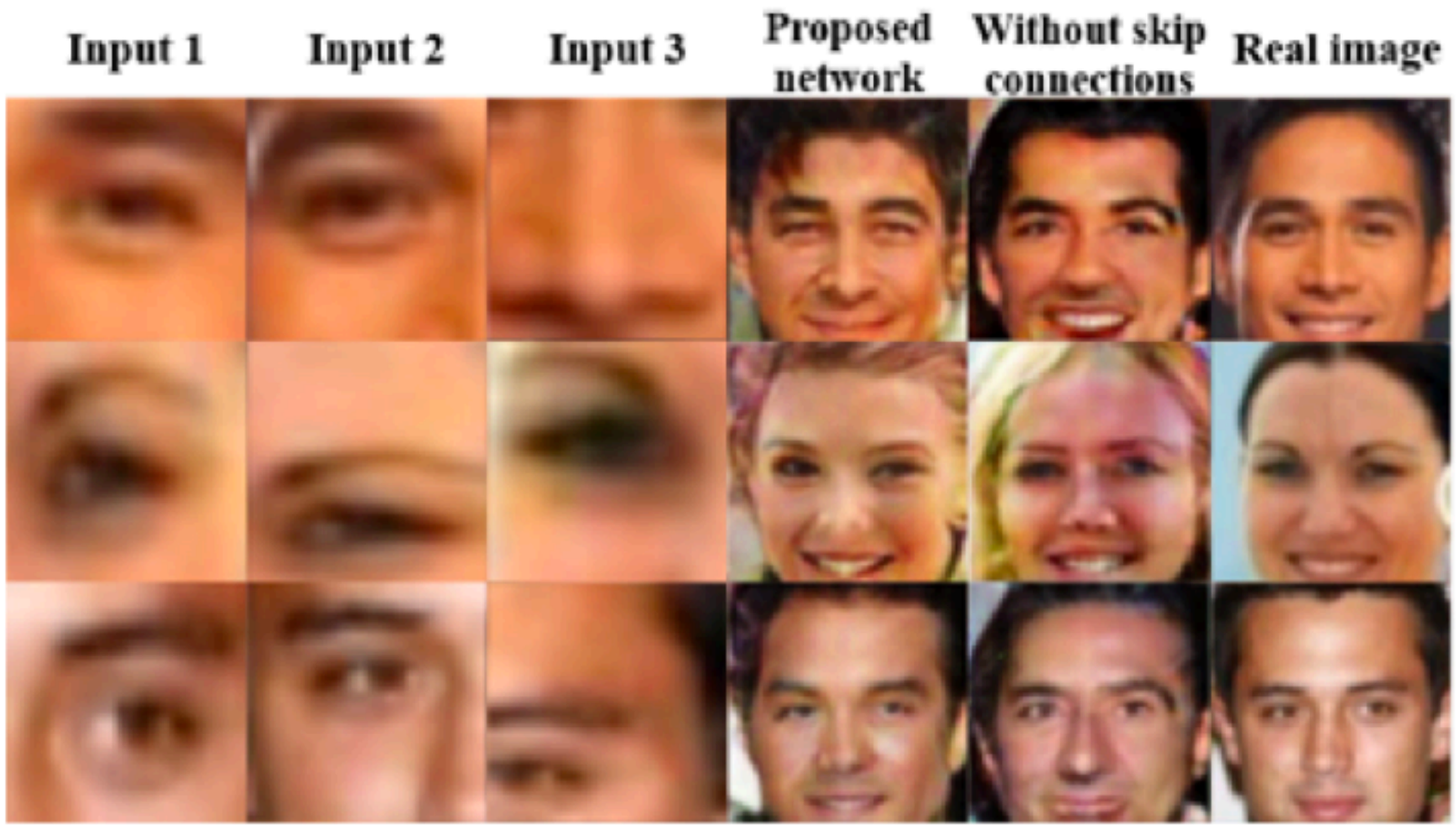
Generative Adversarial What-Where Networks (GAWWN)



https://arxiv.org/abs/1610.02454

Generative Model

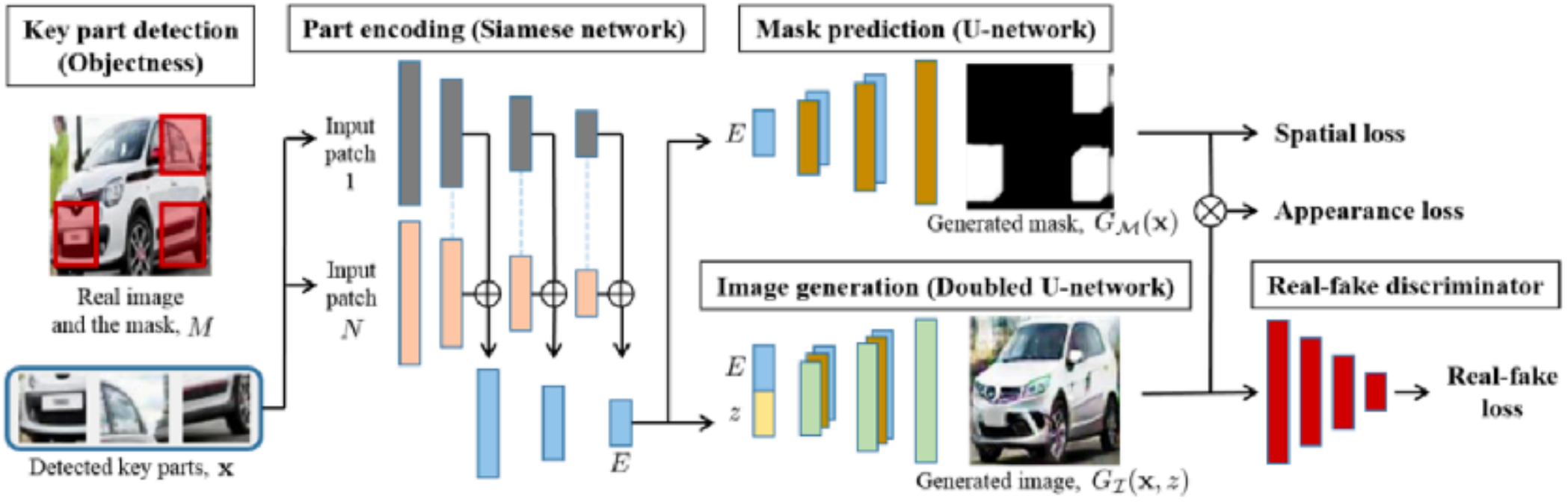
Puzzle-GAN



https://arxiv.org/abs/1703.10730

Generative Model

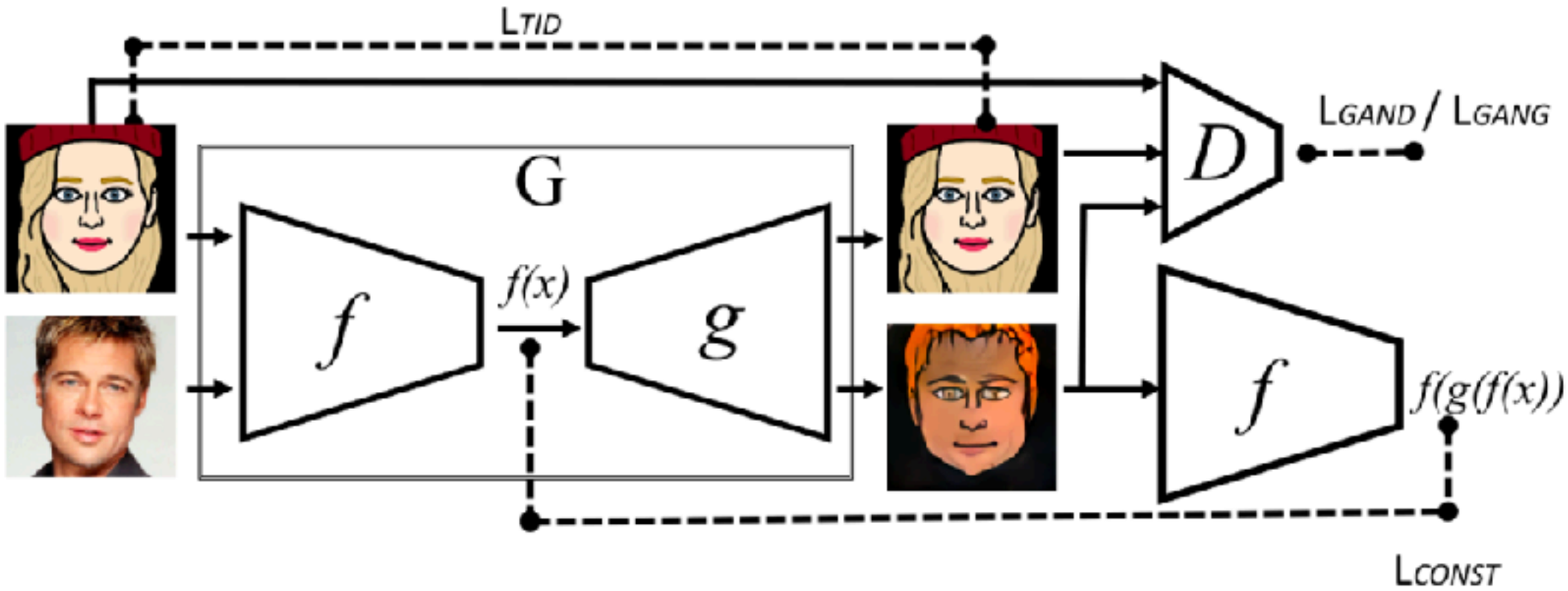
Puzzle-GAN



https://arxiv.org/abs/1703.10730

Generative Model

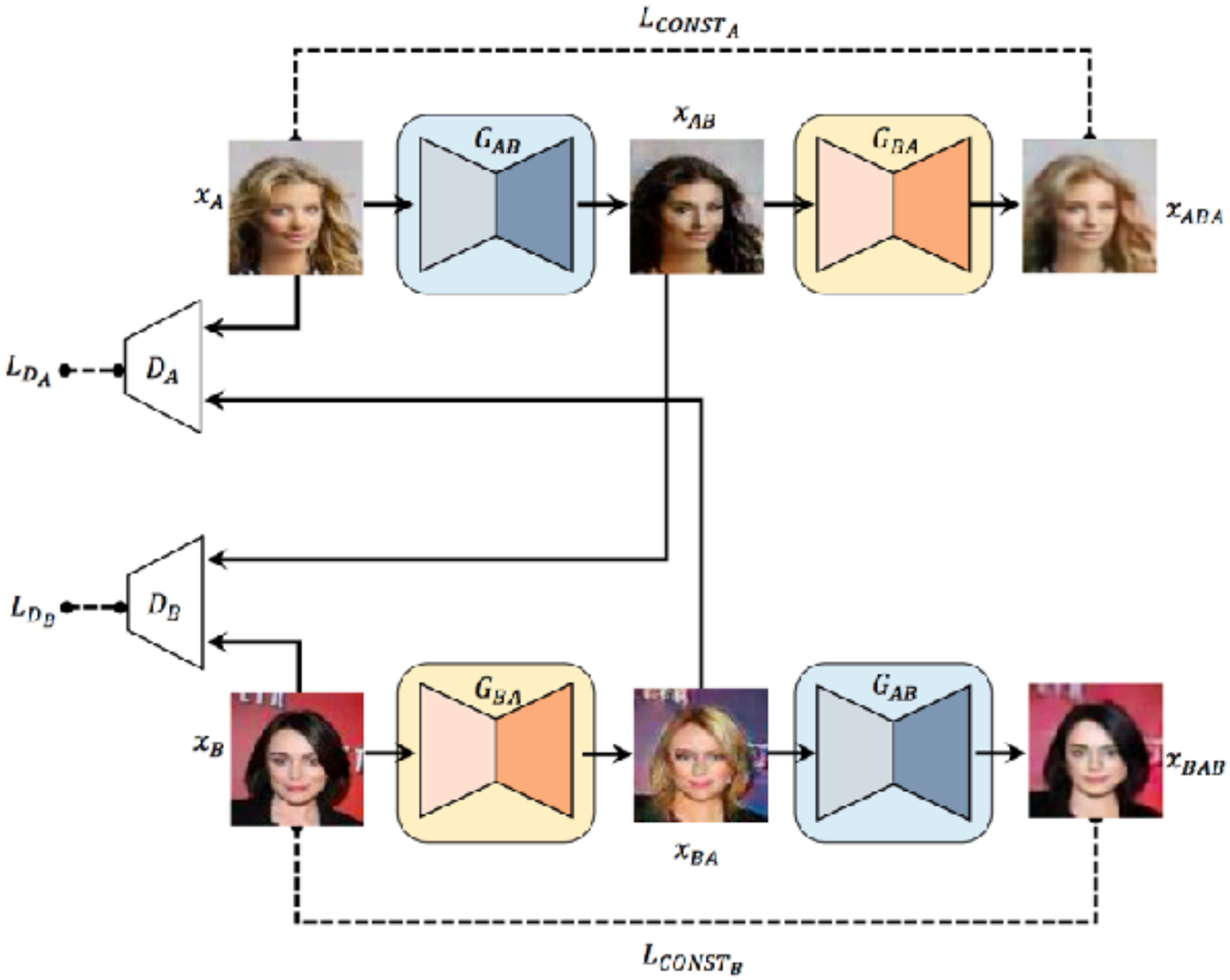
Domain Transfer Network



https://arxiv.org/abs/1611.02200

Generative Model

DiscoGAN



https://arxiv.org/abs/1703.05192

Generative Model

DiscoGAN

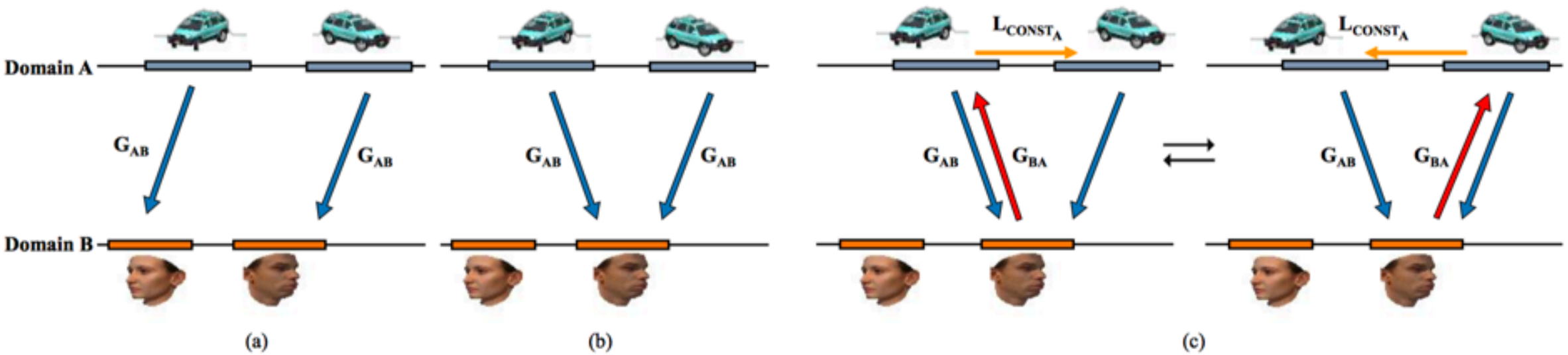
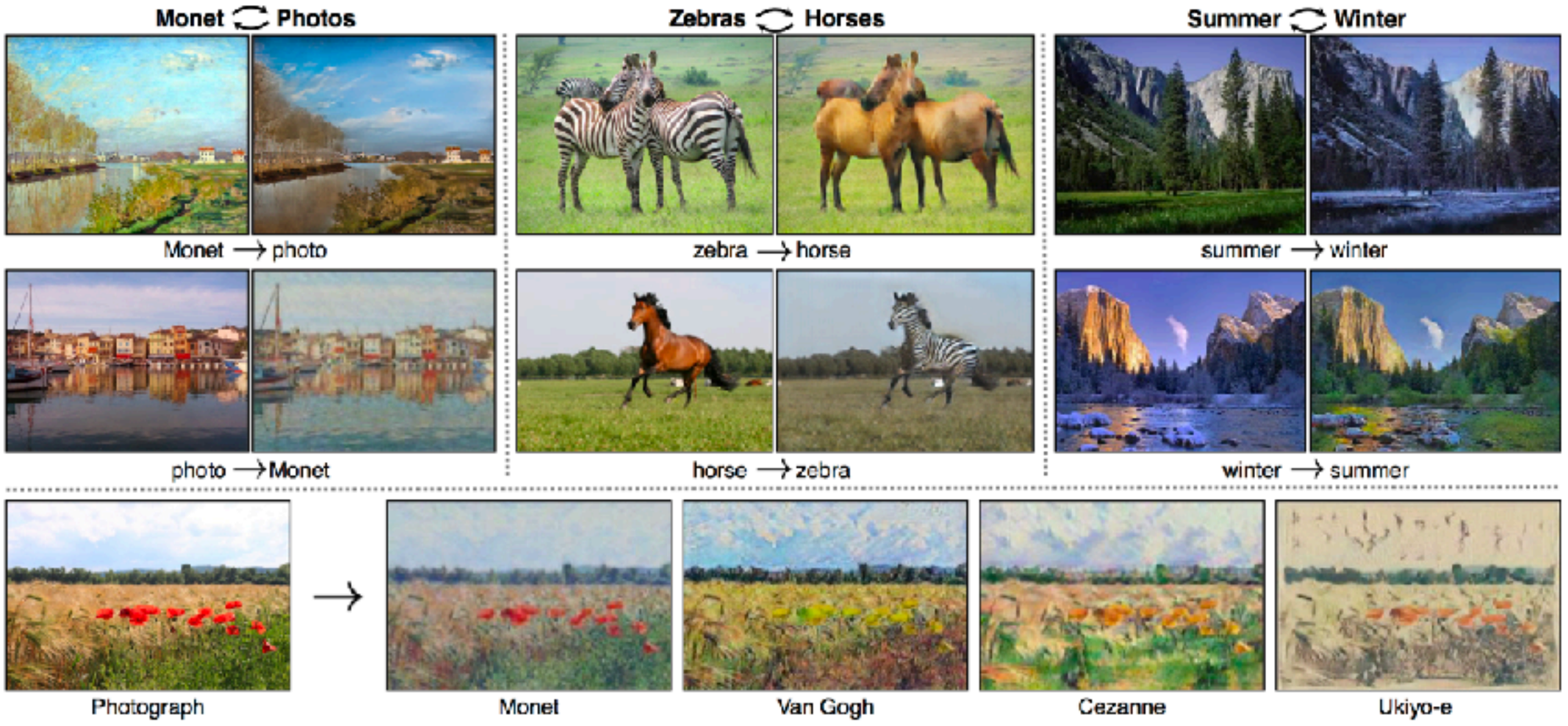


Figure 3. Illustration of our models on simplified one dimensional domains. (a) ideal mapping from domain A to domain B in which the two domain A modes map to two different domain B modes, (b) GAN model failure case, (c) GAN with reconstruction model failure case.

https://arxiv.org/abs/1703.05192

Generative Model

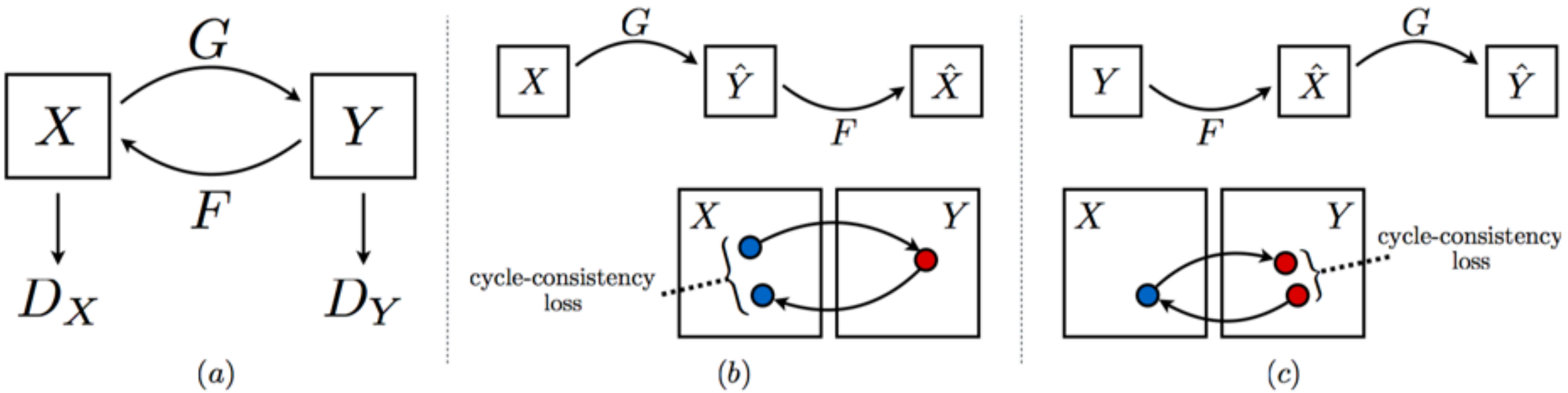
CycleGAN



https://arxiv.org/abs/1703.10593

Generative Model

CycleGAN



https://arxiv.org/abs/1703.10593

Generative Model

StarGAN

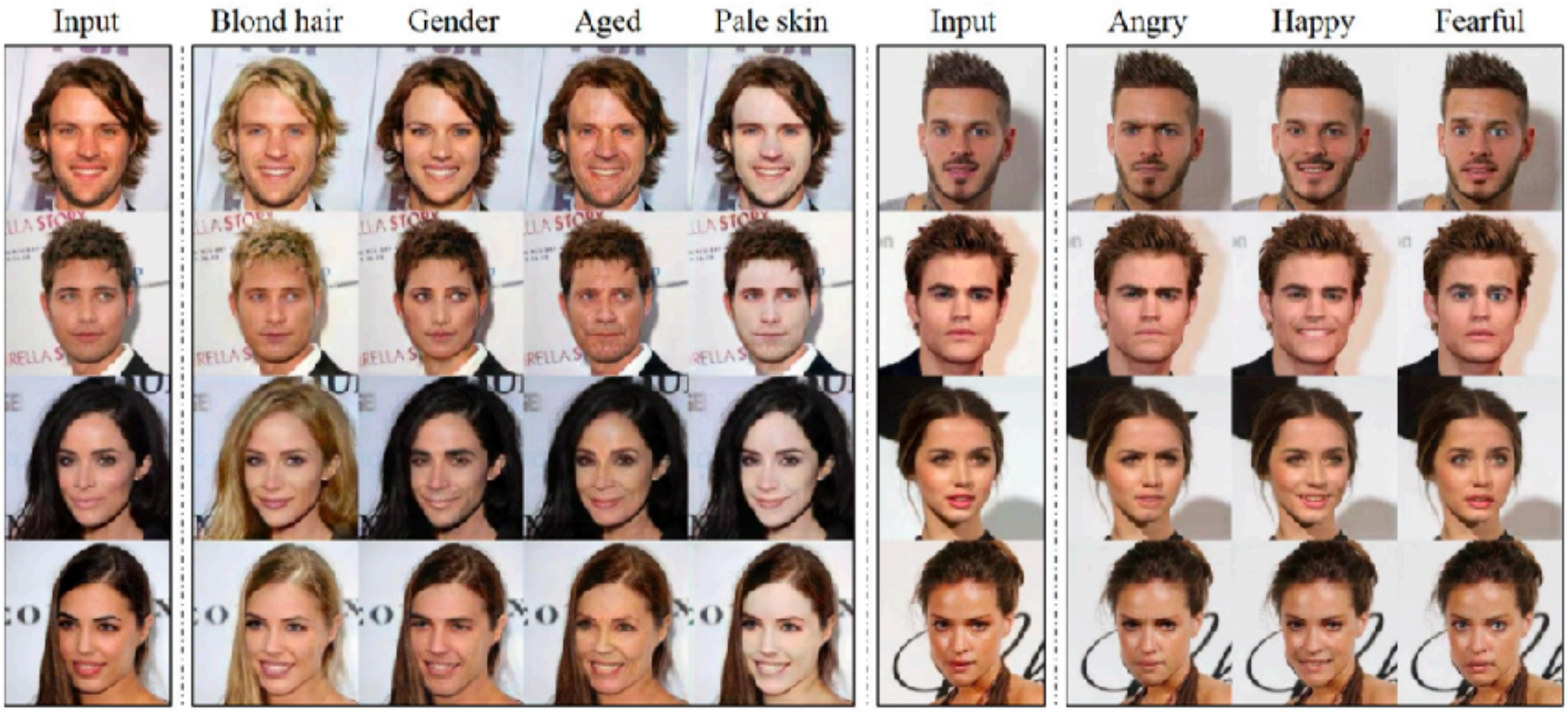
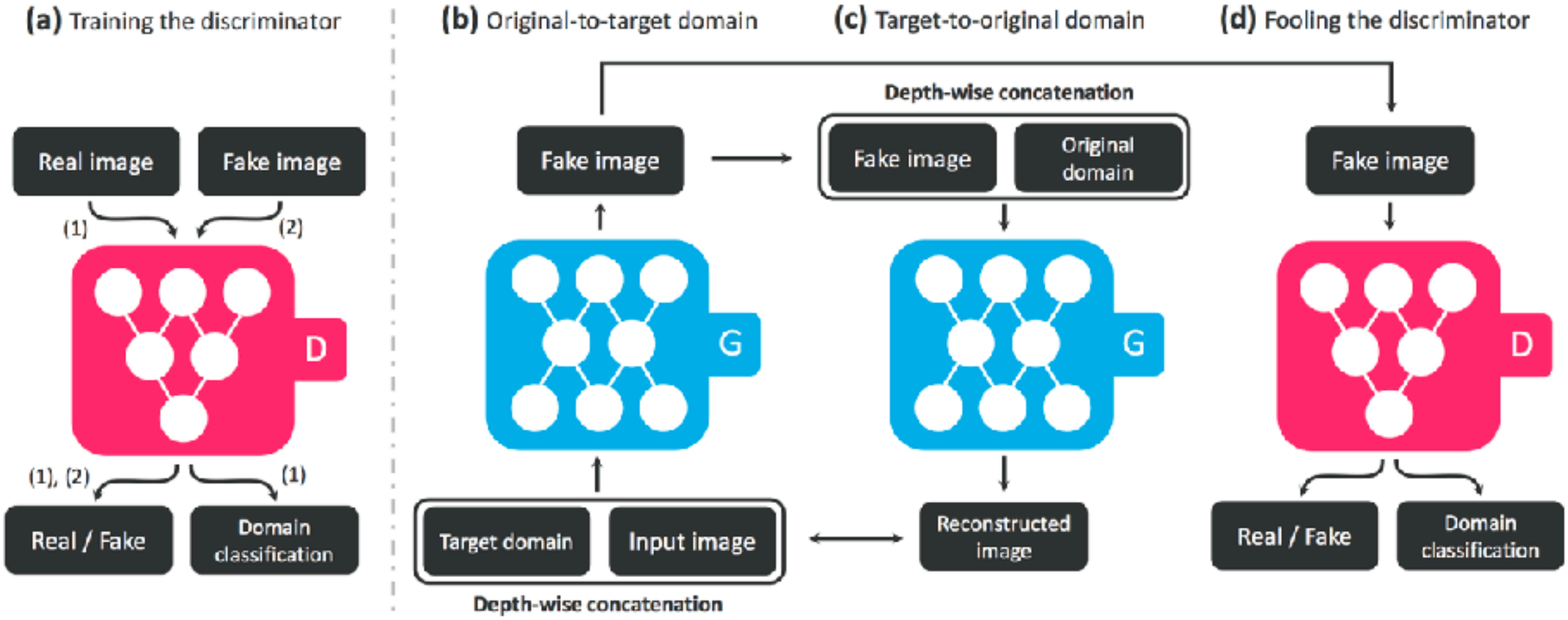


Figure 1. Multi-domain image-to-image translation results on the CelebA dataset via transferring knowledge learned from the RaFD dataset. The first and sixth columns show input images while the remaining columns are images generated by StarGAN. Note that the images are generated by a single generator network, and facial expression labels such as angry, happy, and fearful are from RaFD, not CelebA.

https://arxiv.org/abs/1711.09020

Generative Model

StarGAN



https://arxiv.org/abs/1711.09020

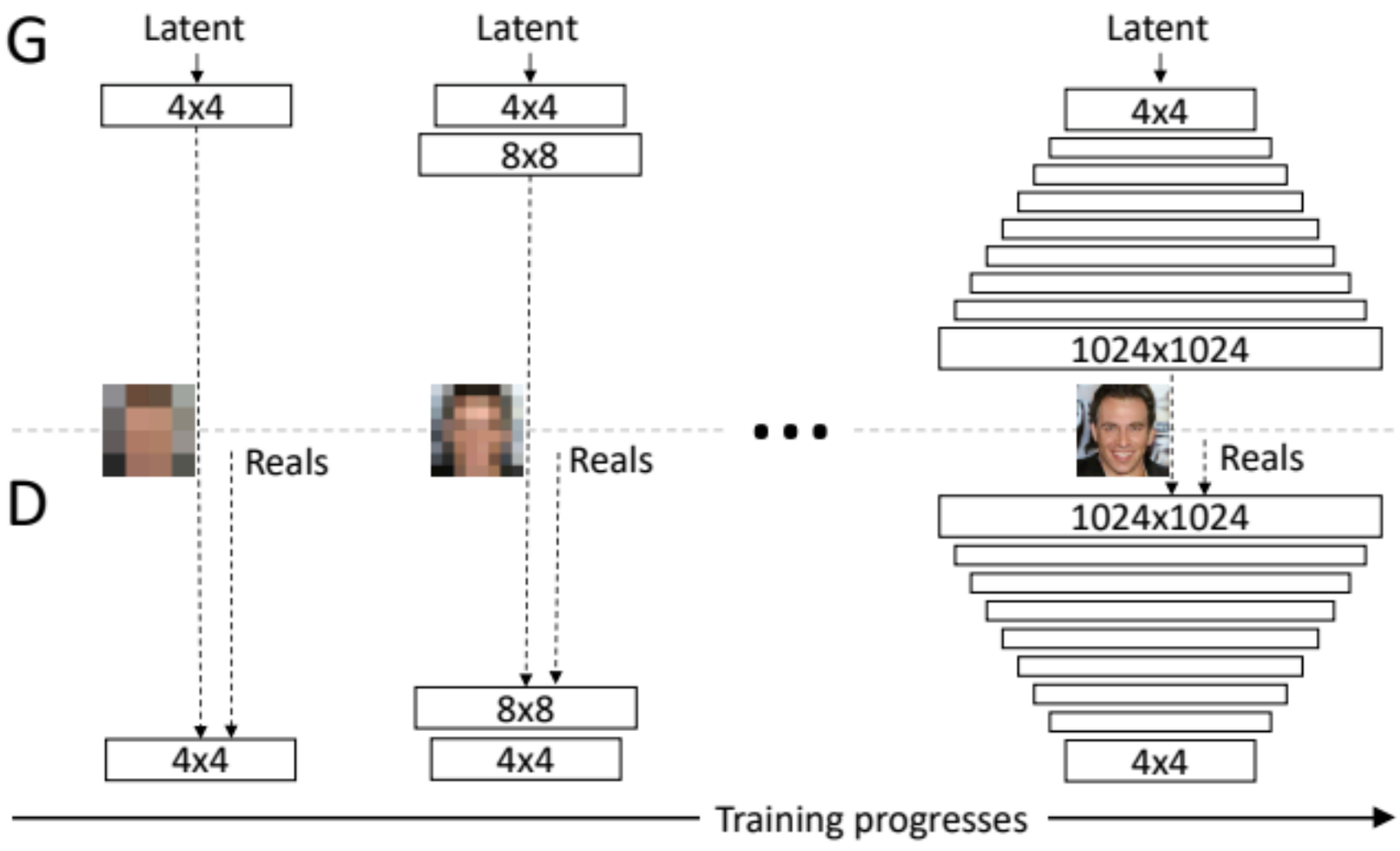
Generative Model

Progressive GAN

https://arxiv.org/abs/1710.10196

Generative Model

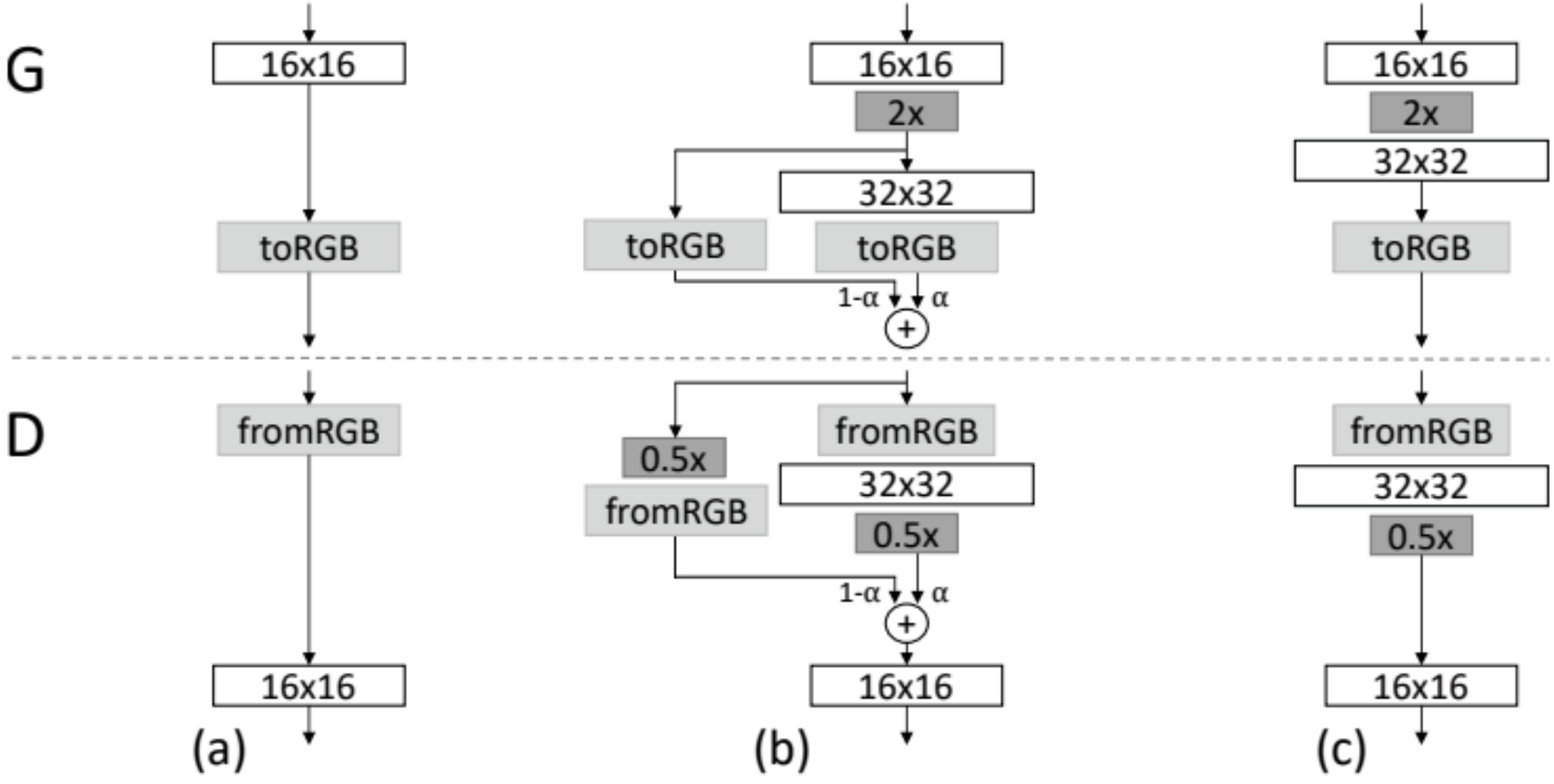
Progressive GAN



https://arxiv.org/abs/1710.10196

Generative Model

Progressive GAN



https://arxiv.org/abs/1710.10196

Domain Adaptation

Domain Adaptation

A step towards practical deployment

Domain Adaptation

Analysis for domain adaptation

Theorem 2 *Let $\mathcal{R}$ be a fixed representation function from $\mathcal{X}$ to $\mathcal{Z}$ and $\mathcal{H}$ be a hypothesis space of VC-dimension d.*

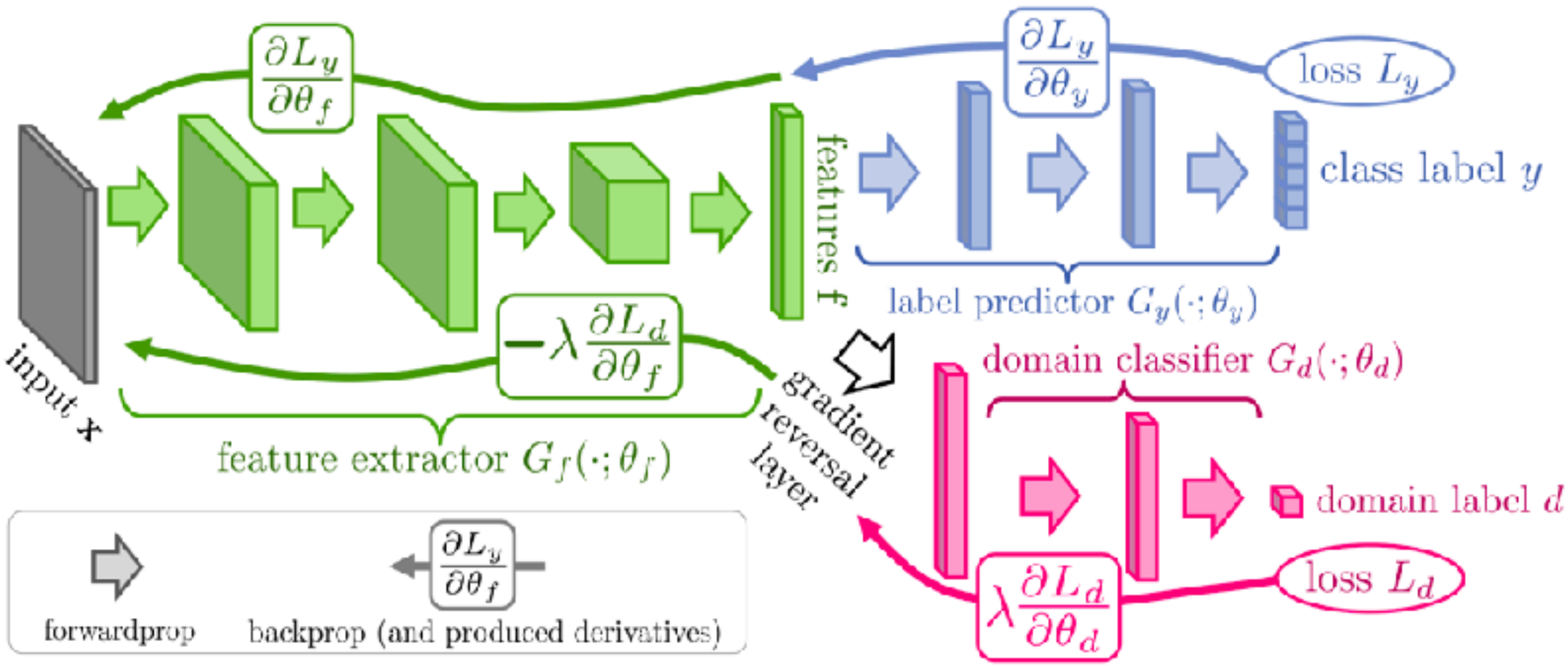
If a random labeled sample of size m is generated by applying R to a $\mathcal{D}_S$ - i.i.d. sample labeled according to f, and $\tilde{\mathcal{U}}_S, \tilde{\mathcal{U}}_T$ are unlabeled samples of size m' each, drawn from $\tilde{\mathcal{D}}_S$ and $\tilde{\mathcal{D}}_T$ respectively, then with probability at least $1-\delta$ (over the choice of the samples), for every $h \in \mathcal{H}$:

$$\epsilon_T(h) \leq \hat{\epsilon}_S(h) + \frac{4}{m}\sqrt{\left(d\log\frac{2em}{d} + \log\frac{4}{\delta}\right)} + \lambda + d_{\mathcal{H}}(\tilde{\mathcal{U}}_S, \tilde{\mathcal{U}}_T) + 4\sqrt{\frac{d\log(2m') + \log(\frac{4}{\delta})}{m'}}$$

http://www.john.blitzer.com/papers/nips06.pdf

Domain Adaptation

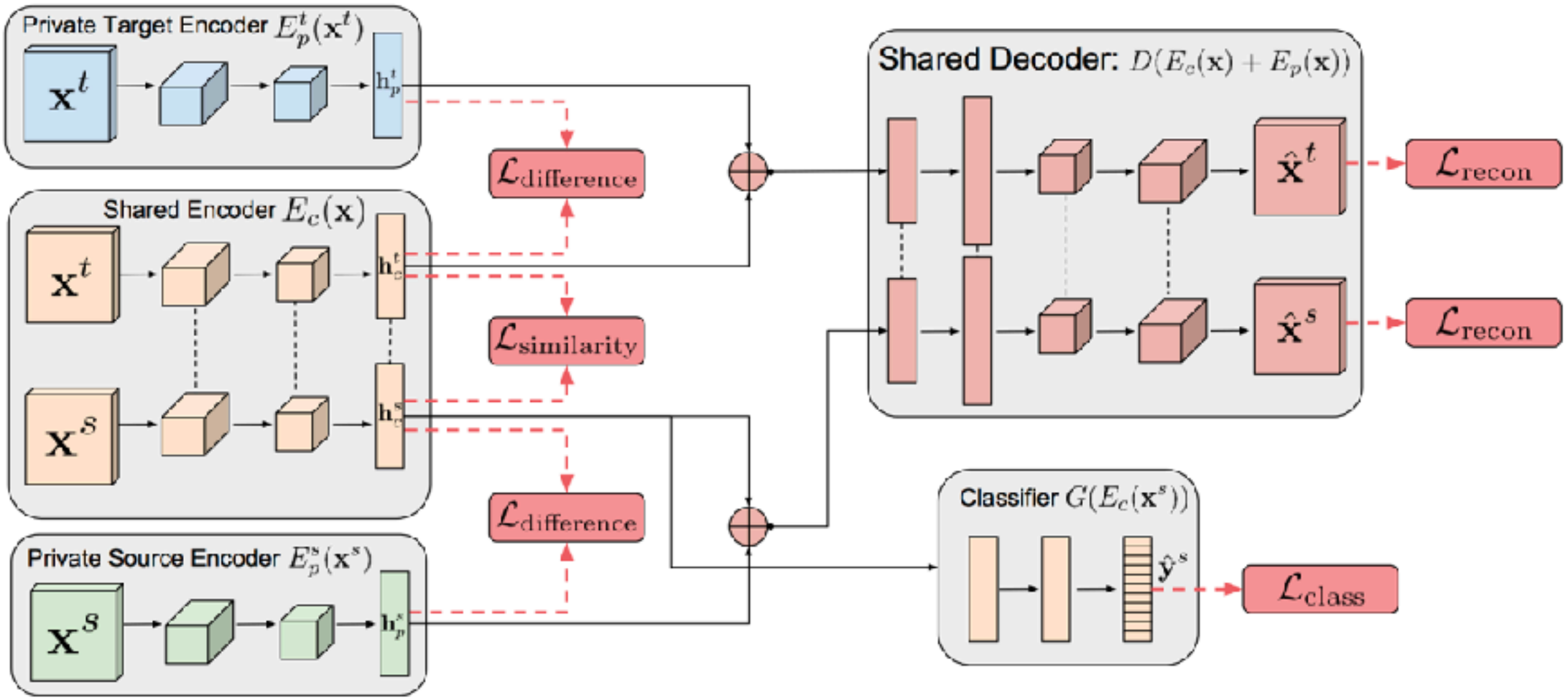
Domain Adversarial Neural Network (DANN)



https://arxiv.org/abs/1505.07818

Domain Adaptation

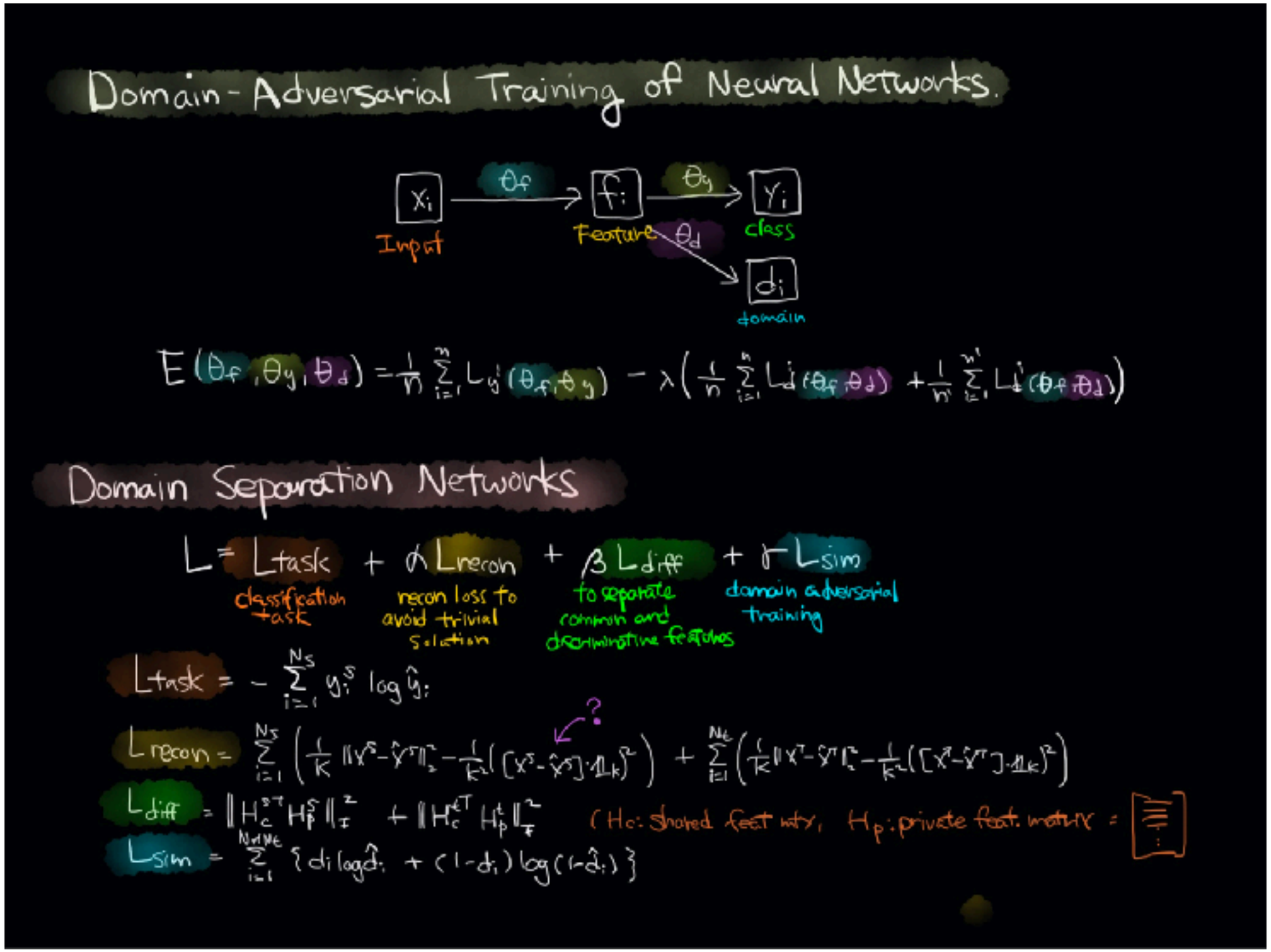
Domain Separation Network (DSN)



https://arxiv.org/abs/1608.06019

Domain Adaptation

DANN & DSN



Domain Adaptation

Coupled GAN

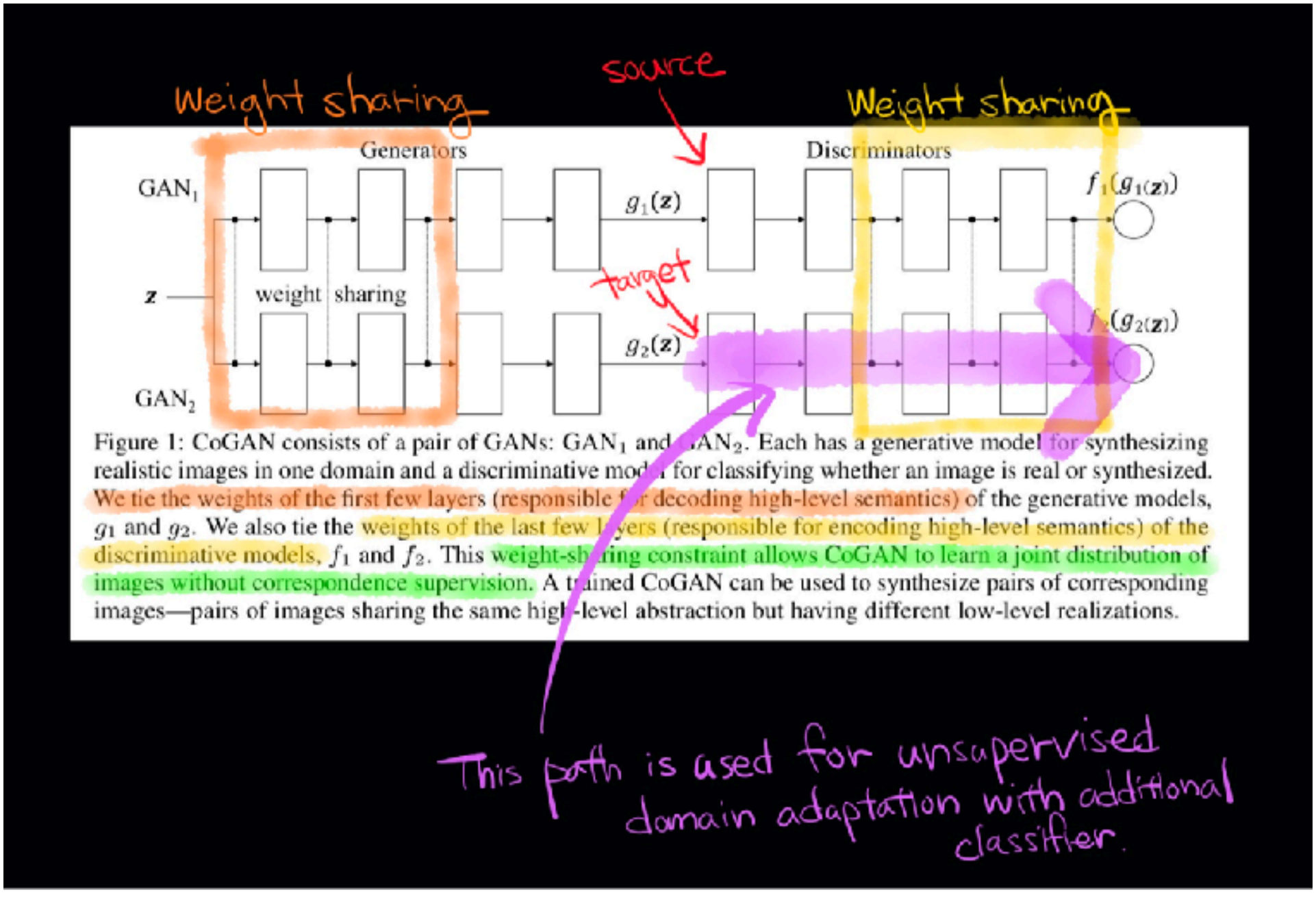
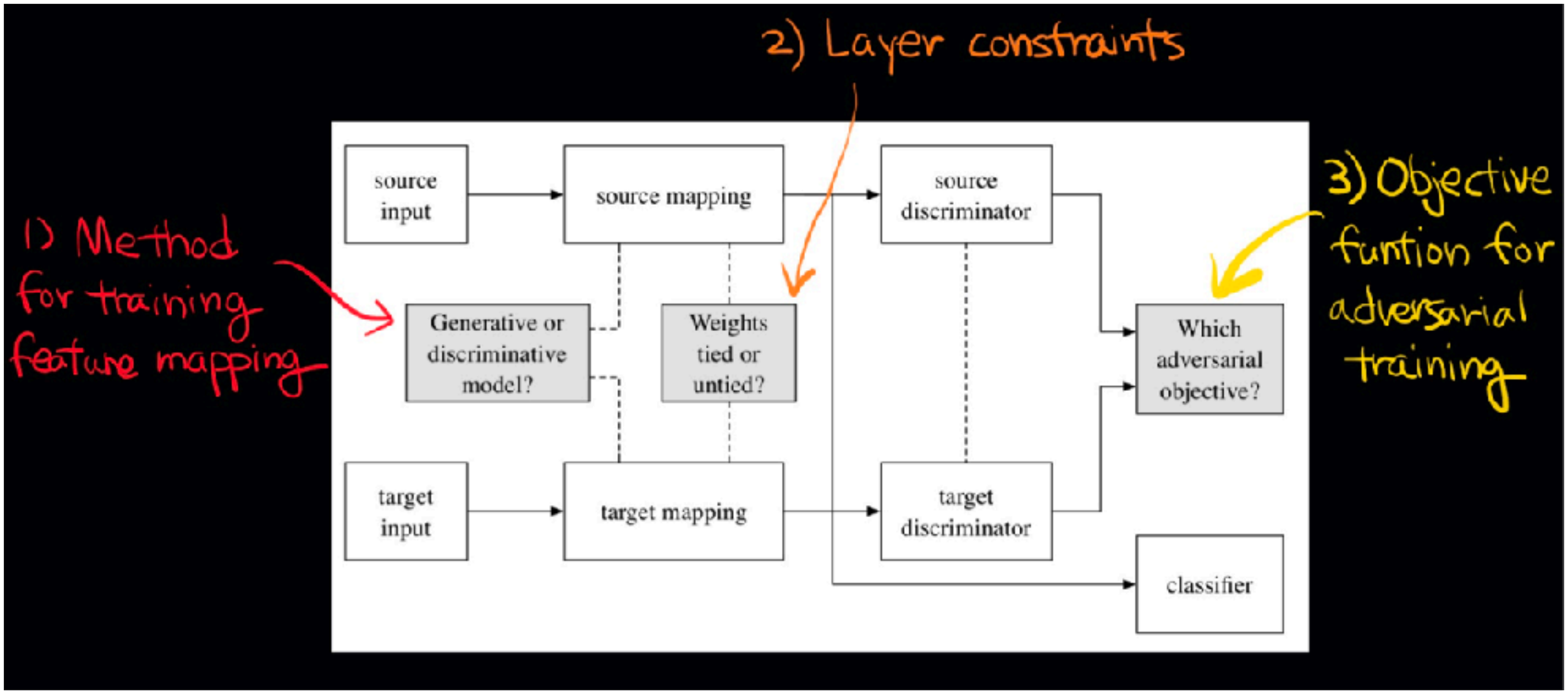


Figure 1: CoGAN consists of a pair of GANs: GAN_1 and GAN_2. Each has a generative model for synthesizing realistic images in one domain and a discriminative model for classifying whether an image is real or synthesized. We tie the weights of the first few layers (responsible for decoding high-level semantics) of the generative models, g_1 and g_2. We also tie the weights of the last few layers (responsible for encoding high-level semantics) of the discriminative models, f_1 and f_2. This weight-sharing constraint allows CoGAN to learn a joint distribution of images without correspondence supervision. A trained CoGAN can be used to synthesize pairs of corresponding images—pairs of images sharing the same high-level abstraction but having different low-level realizations.

https://arxiv.org/abs/1606.07536

Domain Adaptation

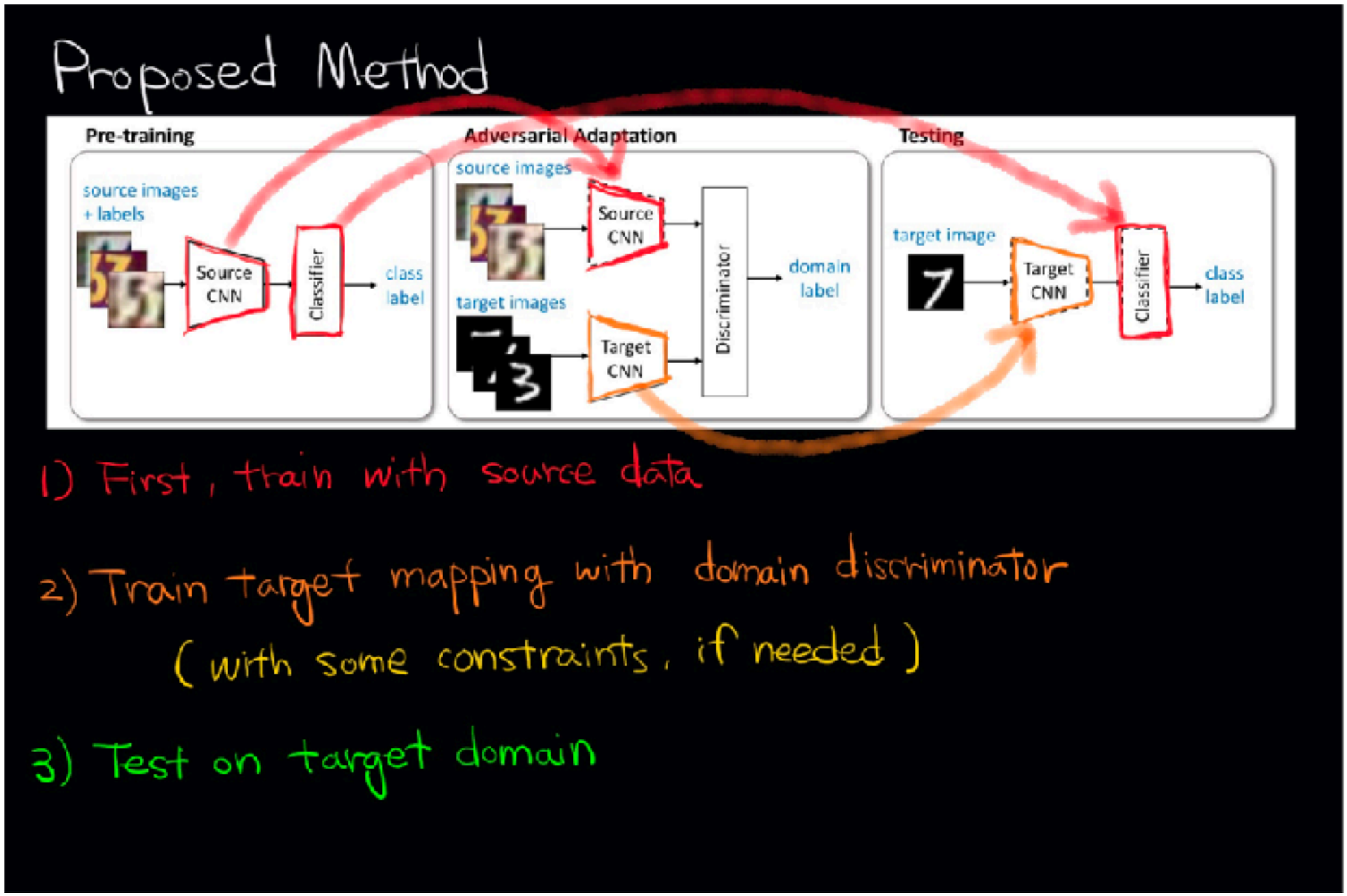
Adversarial Discriminative Domain Adaptation



https://arxiv.org/abs/1606.07536

Domain Adaptation

Adversarial Discriminative Domain Adaptation



https://arxiv.org/abs/1606.07536

Domain Adaptation

Appearance adaptation

https://arxiv.org/abs/1703.01461

Domain Adaptation

Appearance adaptation

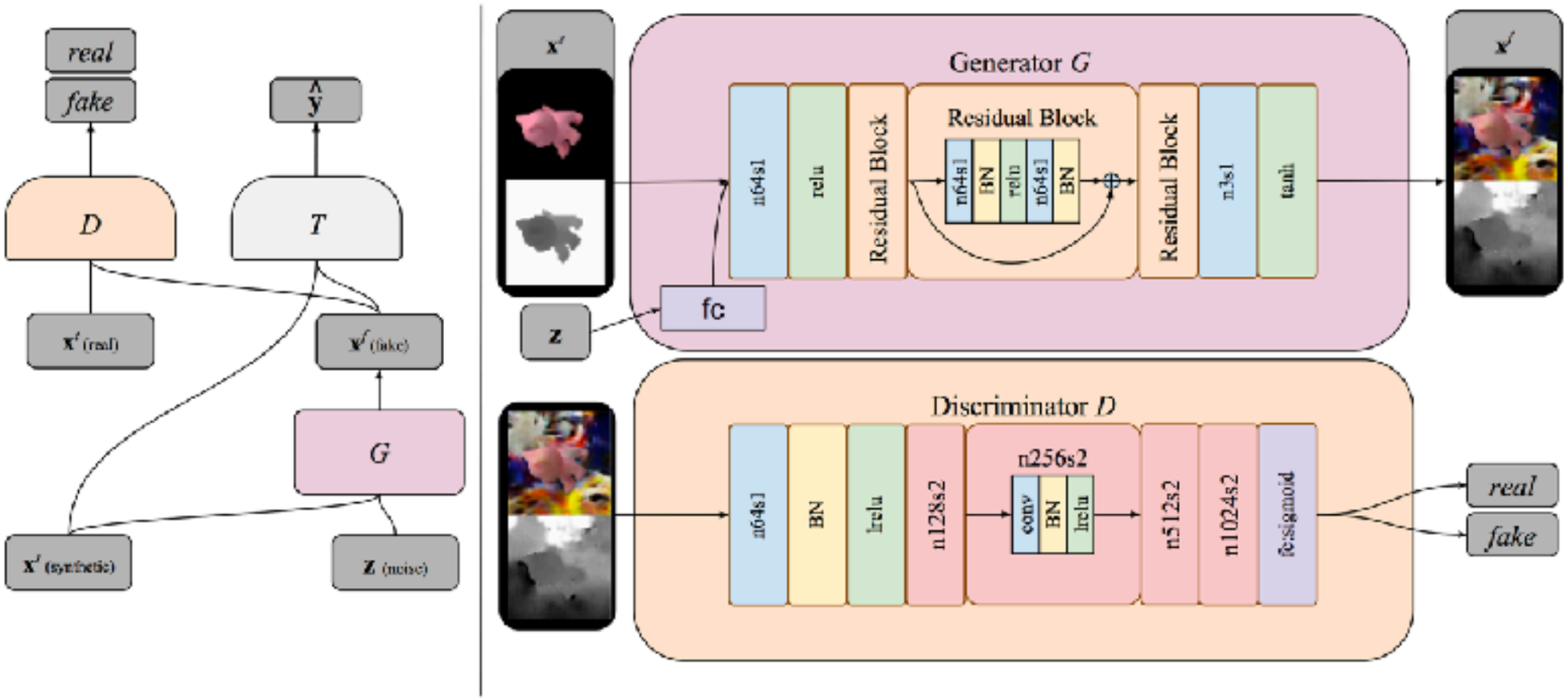
FCV-VGG16

FCV-VGG16 w ADA

https://arxiv.org/abs/1703.01461

Domain Adaptation

DA-GAN



https://arxiv.org/abs/1612.05424

Domain Adaptation

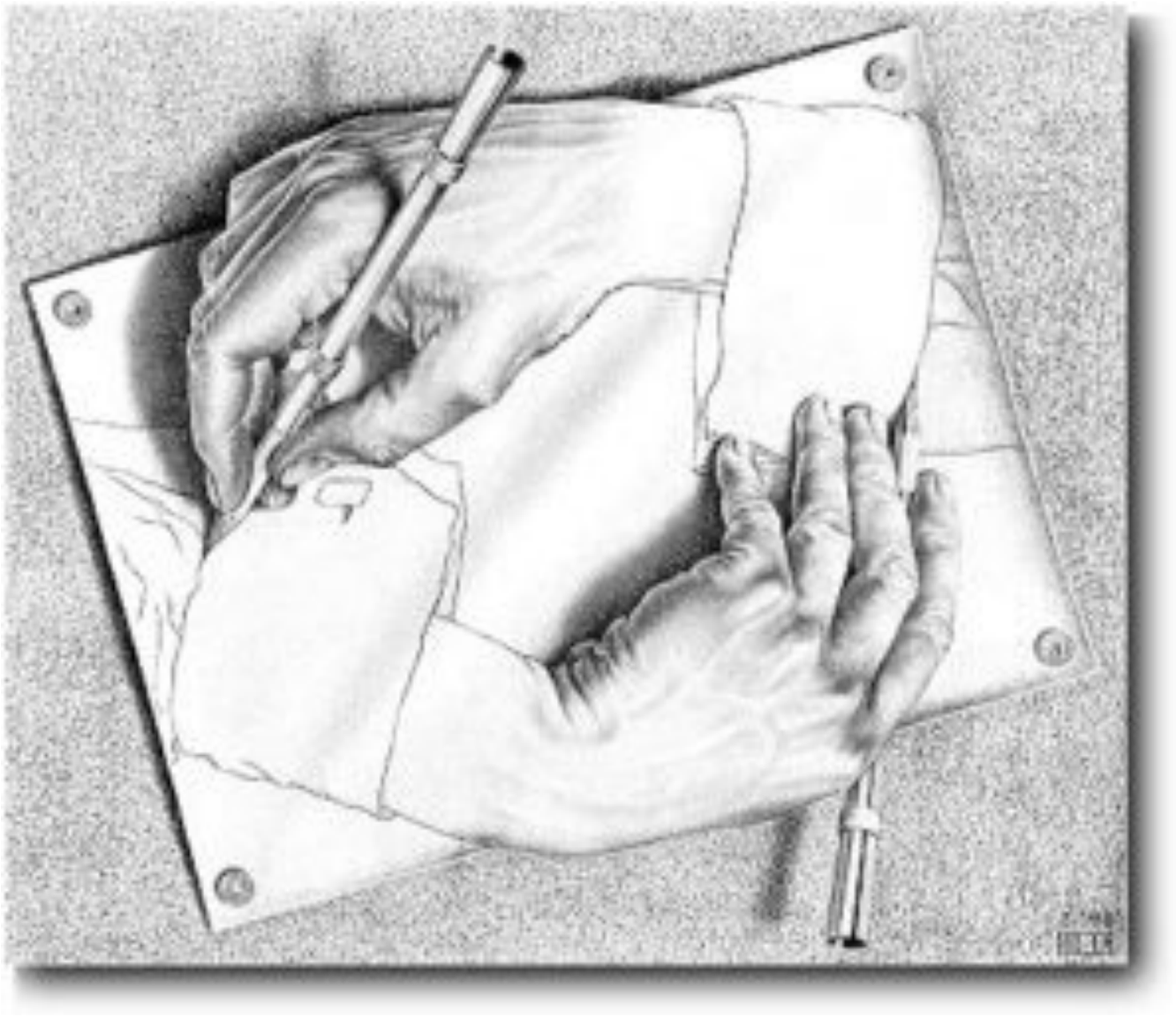
DA-GAN

https://arxiv.org/abs/1612.05424

Meta Learning

Meta Learning

Learn to learn



Meta Learning

Learning to Learn without Gradient Descent by Gradient Descent

2. Learning Black-box Optimization

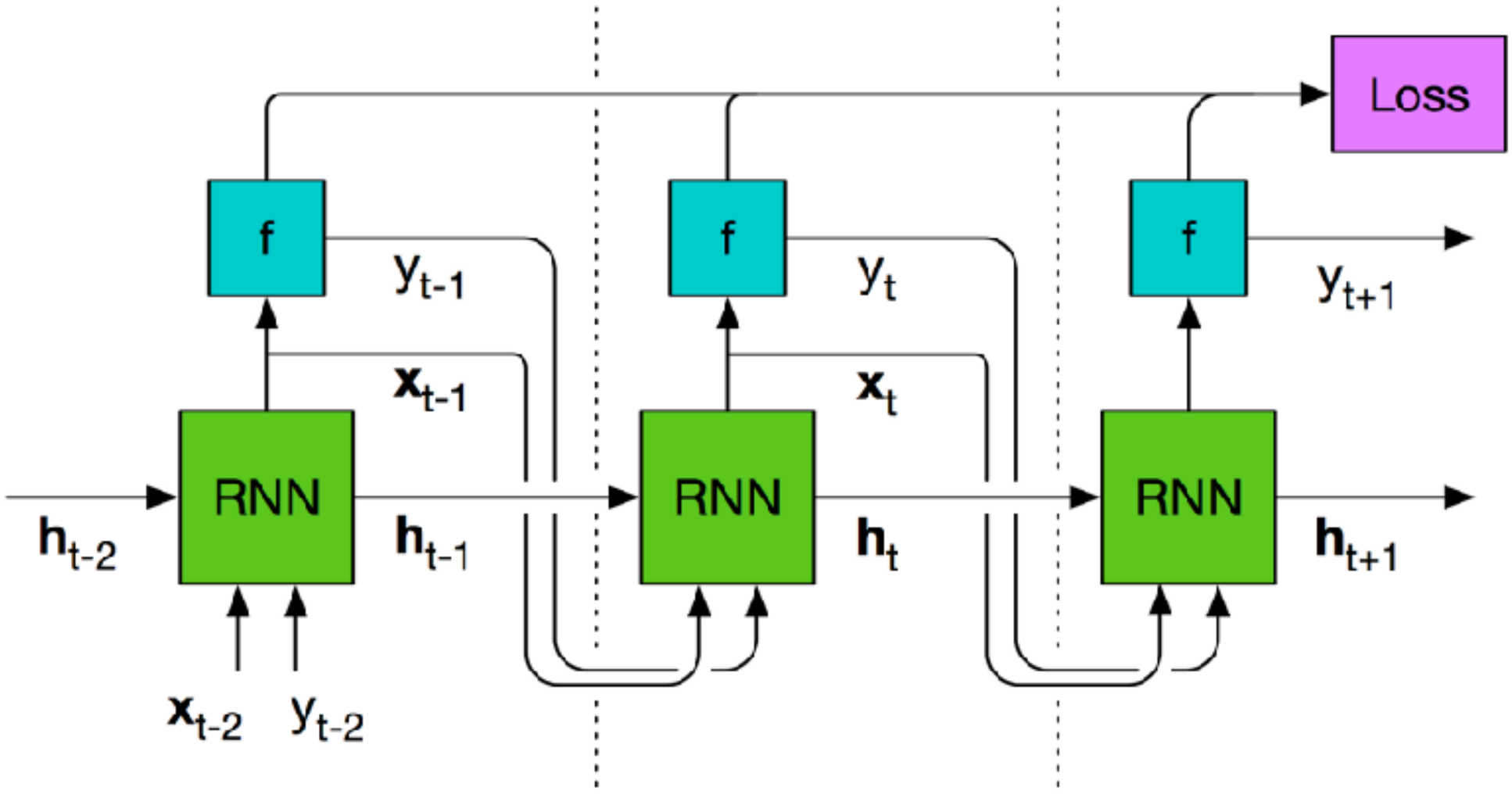
A black-box optimization algorithm can be summarized by the following loop:

1. Given the current state of knowledge $\mathbf{h}_t$ propose a query point $\mathbf{x}_t$
2. Observe the response y_t
3. Update any internal statistics to produce $\mathbf{h}_{t+1}$

https://arxiv.org/abs/1611.03824

Meta Learning

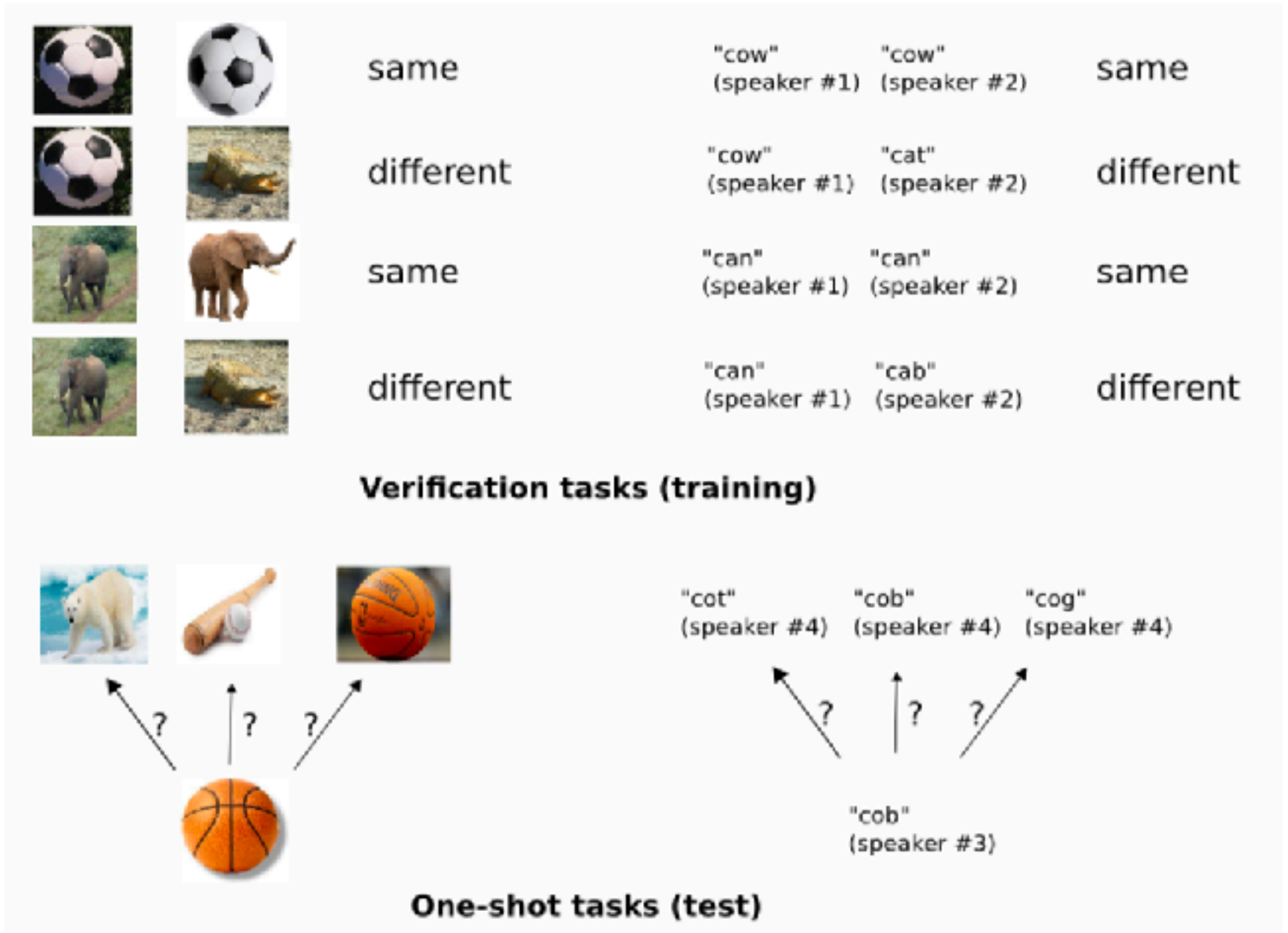
Learning to Learn without Gradient Descent by Gradient Descent



https://arxiv.org/abs/1611.03824

Meta Learning

Siamese Neural Networks



https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf

Meta Learning

Siamese Neural Networks

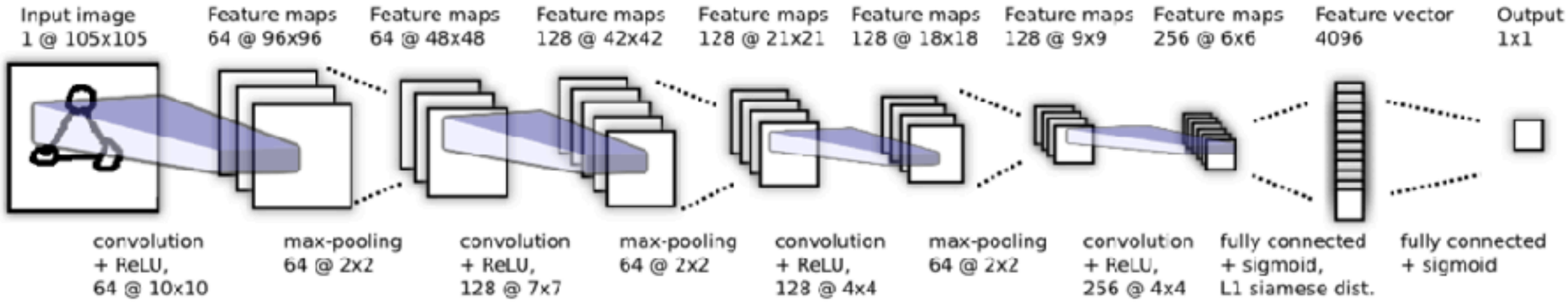


Figure 4. Best convolutional architecture selected for verification task. Siamese twin is not depicted, but joins immediately after the 4096 unit fully-connected layer where the L1 component-wise distance between vectors is computed.

https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf

Meta Learning

Triplet Networks

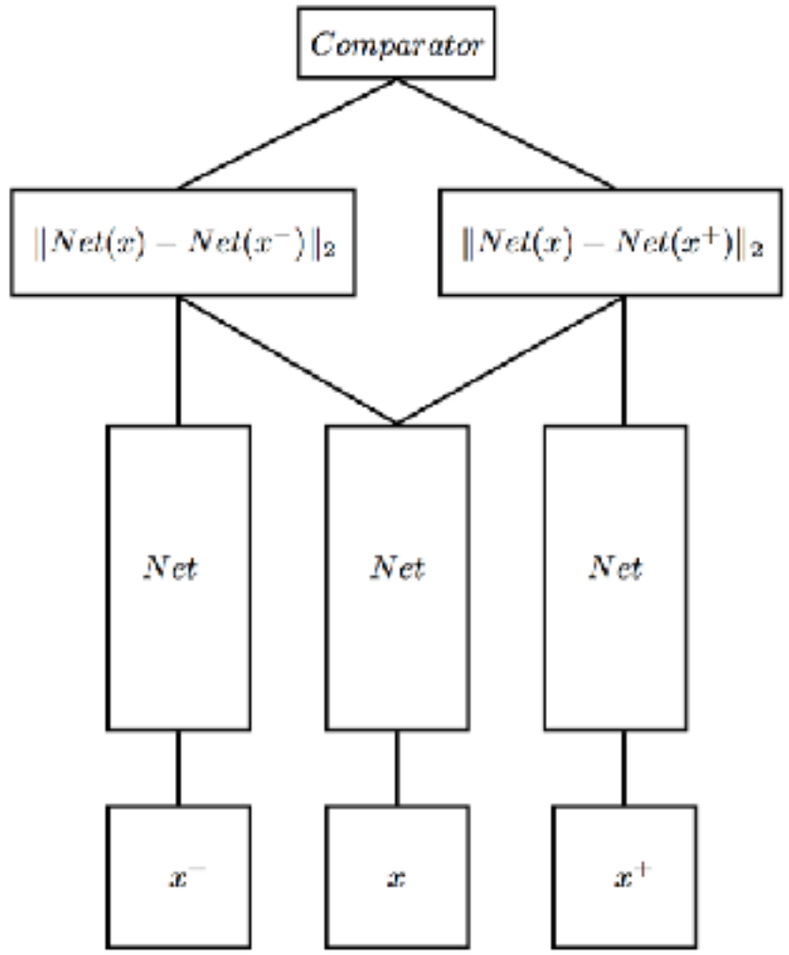


Figure 1: Triplet network structure

https://arxiv.org/abs/1412.6622

Meta Learning

Matching Network

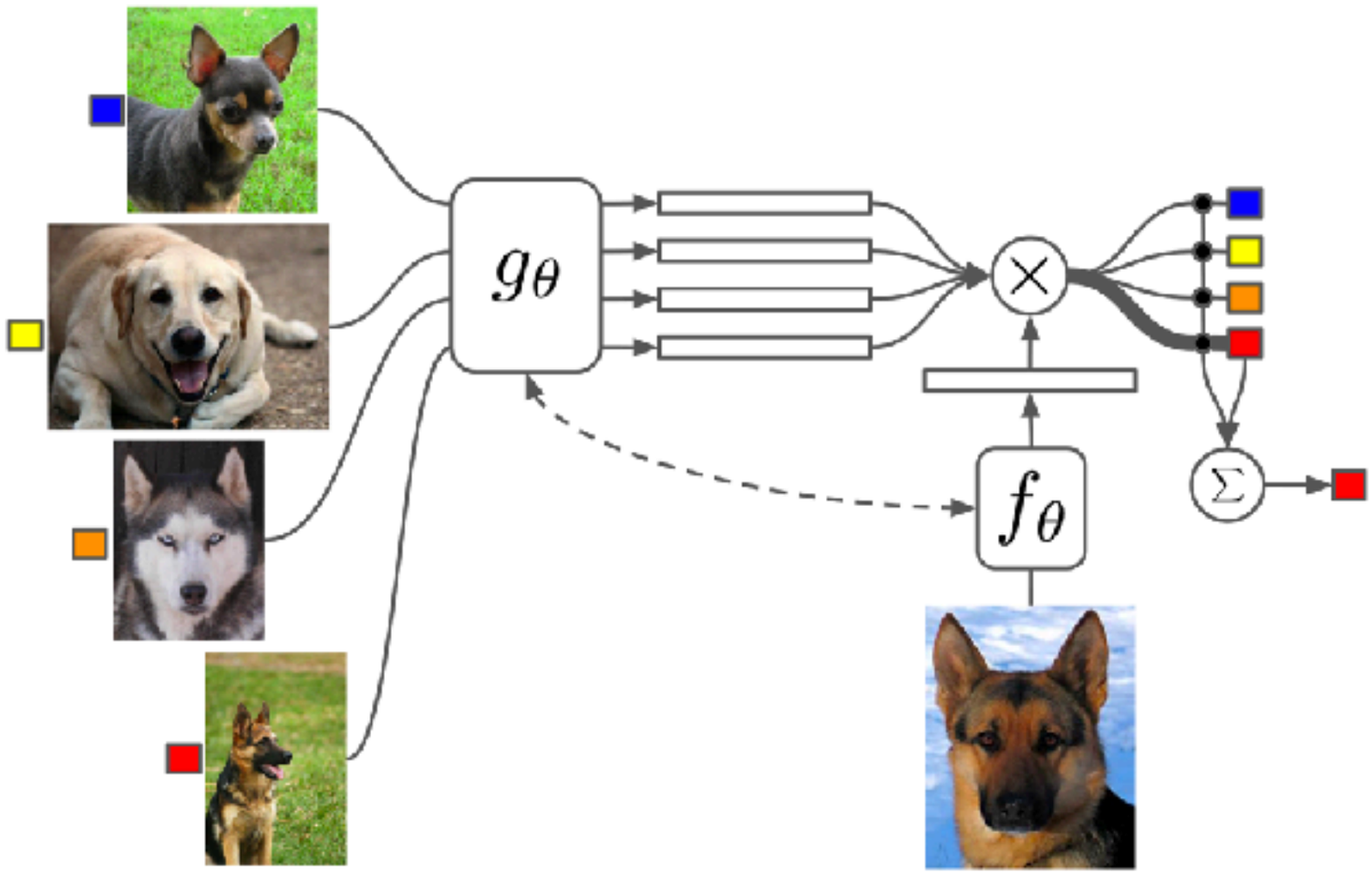


Figure 1: Matching Networks architecture

https://arxiv.org/abs/1606.04080

Meta Learning

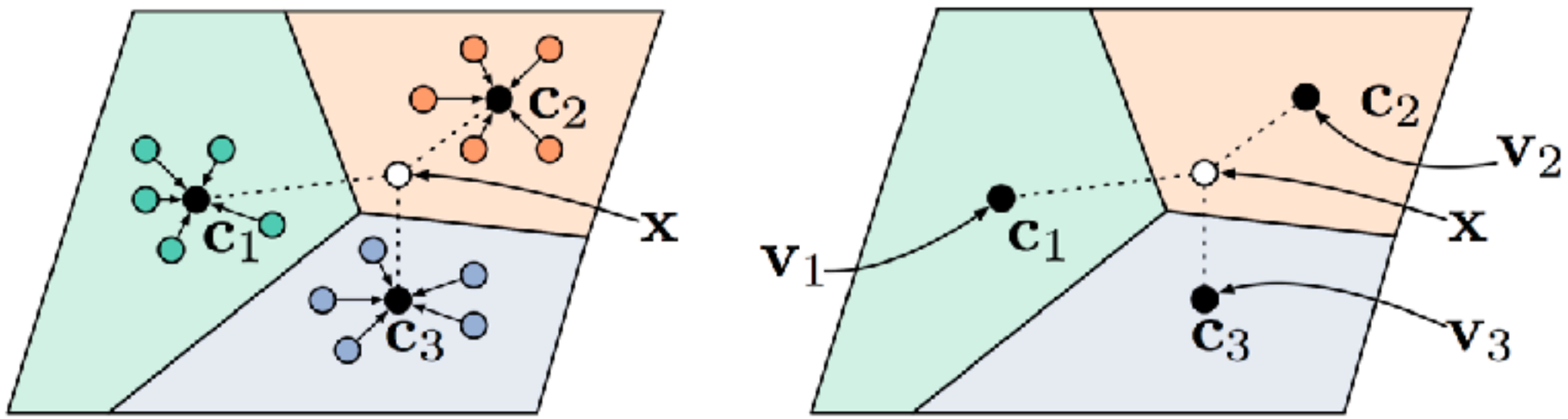
Matching Network

$$\hat{y} = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i$$

https://arxiv.org/abs/1606.04080

Meta Learning

Prototypical Networks



(a) Few-shot (b) Zero-shot

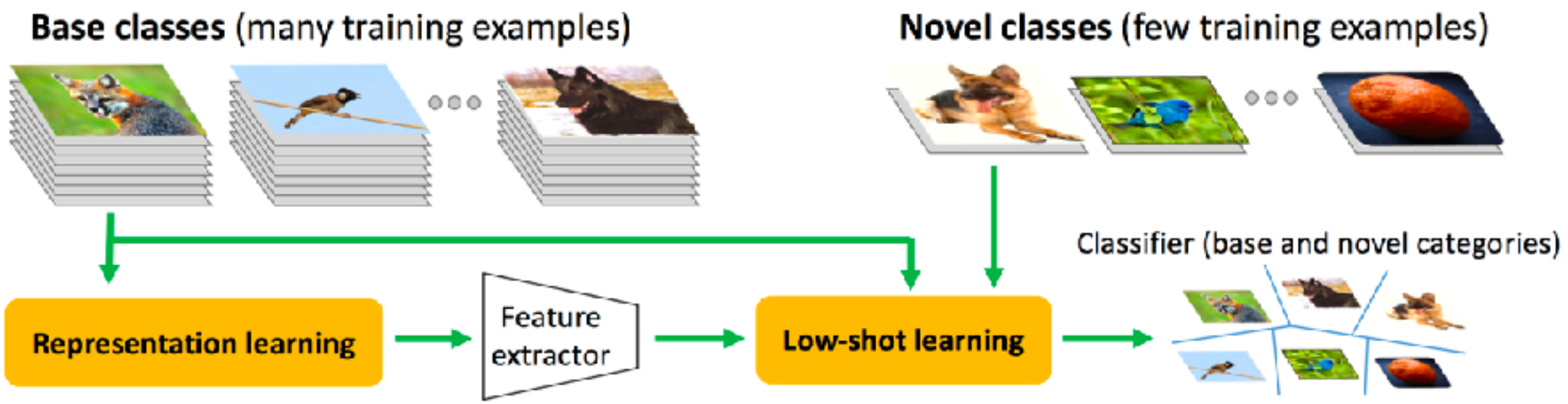
Figure 1: Prototypical networks in the few-shot and zero-shot scenarios. **Left**: Few-shot prototypes $\mathbf{c}_k$ are computed as the mean of embedded support examples for each class. **Right**: Zero-shot prototypes $\mathbf{c}_k$ are produced by embedding class meta-data $\mathbf{v}_k$. In either case, embedded query points are classified via a softmax over distances to class prototypes: $p_{\phi}(y = k|\mathbf{x}) \propto \exp(-d(f_{\phi}(\mathbf{x}), \mathbf{c}_k))$.

Prototypical networks learn **a metric space** in which classification can be performed by computing distances to prototype representations of each class.

https://arxiv.org/abs/1703.05175

Meta Learning

Low-shot learning



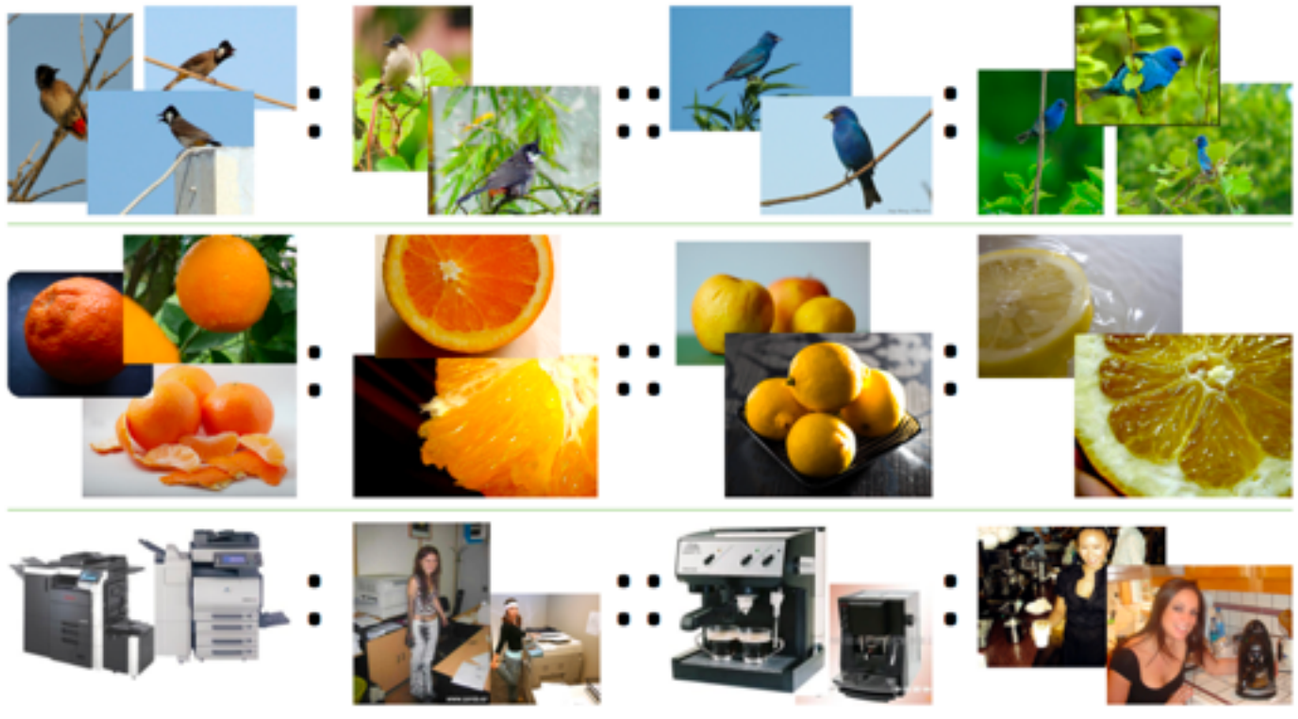
Low-shot visual learning—the ability to recognize novel object categories from very few examples—is a hallmark of human visual intelligence.

https://arxiv.org/abs/1606.02819

Meta Learning

Low-shot learning

Learning to **generate** new examples



https://arxiv.org/abs/1606.02819

Meta Learning

Design a CNN with RL

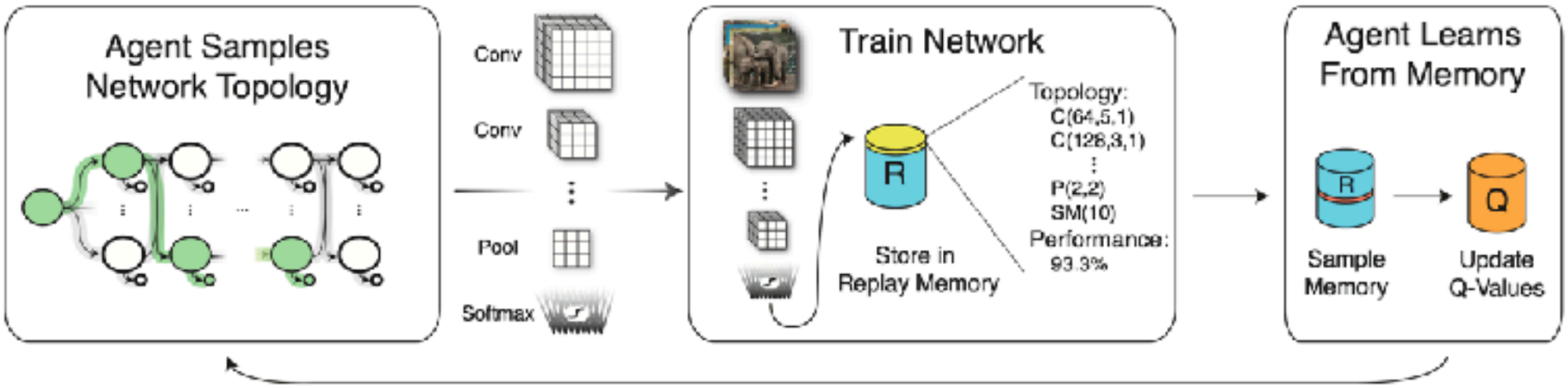
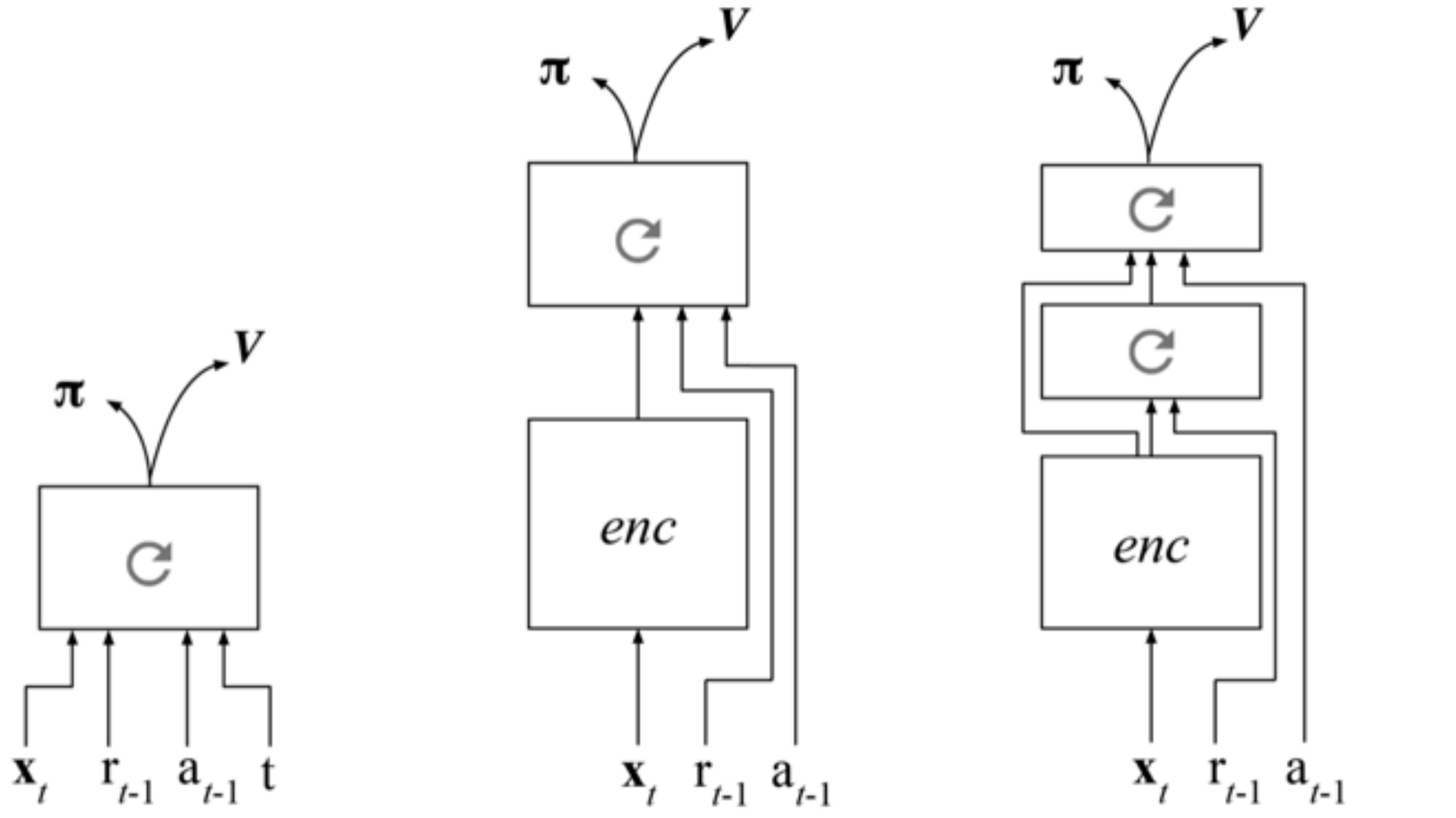


Figure 1: **Designing CNN Architectures with *Q*-learning:** The agent begins by sampling a Convolutional Neural Network (CNN) topology conditioned on a predefined behavior distribution and the agent's prior experience (left block). That CNN topology is then trained on a specific task; the topology description and performance, e.g. validation accuracy, are then stored in the agent's memory (middle block). Finally, the agent uses its memories to learn about the space of CNN topologies through *Q*-learning (right block).

https://arxiv.org/pdf/1611.02167

Meta Learning

Deep meta RL

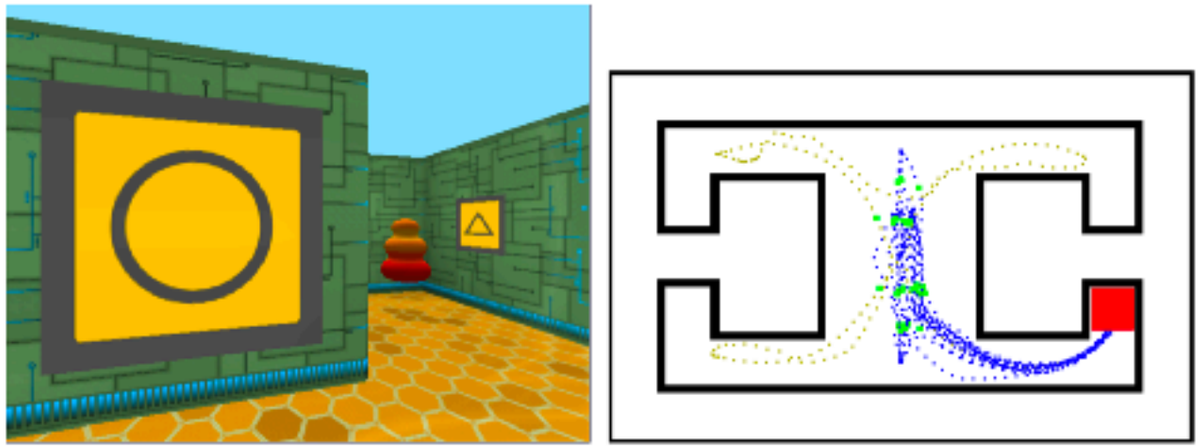


(a) LSTM A2C (b) LSTM A3C (c) Stacked-LSTM A3C

https://arxiv.org/abs/1611.05763

Meta Learning

Deep meta RL



(a) Labryinth I-maze (b) Illustrative Episode

(c) Performance (d) Value Function

Figure 7: a) view of I-maze showing goal object in one of the 4 alcoves b) following initial exploration (light trajectories), agent repeatedly goes to goal (blue trajectories) c) Performance of stacked LSTM (termed "Nav A3C") and feedforward ("FF A3C") architectures, per episode (goal = 10 points) averaged across top 5 hyperparameters. e) following initial goal discovery (goal hits marked in red), value function occurs well in advance of the agent seeing the goal which is hidden in an alcove. Figure used with permission from Mirowski et al. (2016).

https://arxiv.org/abs/1611.05763

Meta Learning

Model-Agnostic Meta-Learning

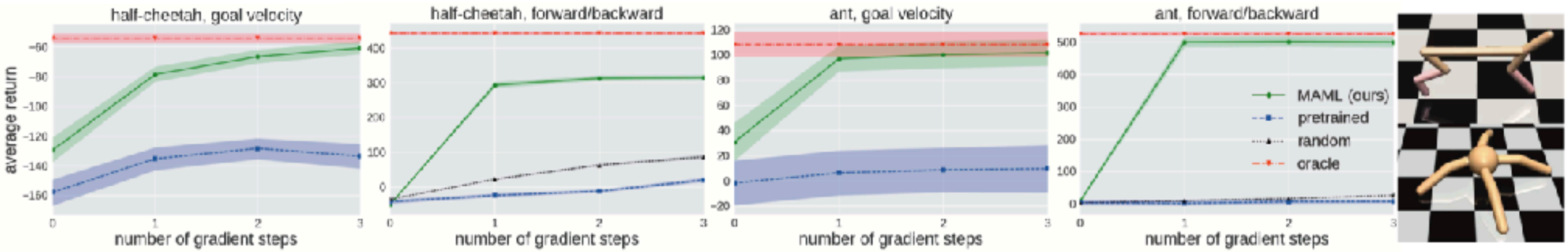


Figure 5. Reinforcement learning results for the half-cheetah and ant locomotion tasks, with the tasks shown on the far right. Each gradient step requires additional samples from the environment, unlike the supervised learning tasks. The results show that MAML can adapt to new goal velocities and directions substantially faster than conventional pretraining or random initialization, achieving good performs in just two or three gradient steps. We exclude the goal velocity, random baseline curves, since the returns are much worse (< -200 for cheetah and < -25 for ant).

https://arxiv.org/abs/1703.03400

Meta Learning

Model-Agnostic Meta-Learning

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
6: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
7: **end for**
8: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
9: **end while**

https://arxiv.org/abs/1703.03400

Meta Learning

Model-Agnostic Meta-Learning

Algorithm 3 MAML for Reinforcement Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample K trajectories $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{a}_1, ...\mathbf{x}_H)\}$ using f_θ in $\mathcal{T}_i$
6: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 4
7: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
8: Sample trajectories $\mathcal{D}_i' = \{(\mathbf{x}_1, \mathbf{a}_1, ...\mathbf{x}_H)\}$ using $f_{\theta_i'}$ in $\mathcal{T}_i$
9: **end for**
10: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$ using each $\mathcal{D}_i'$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 4
11: **end while**

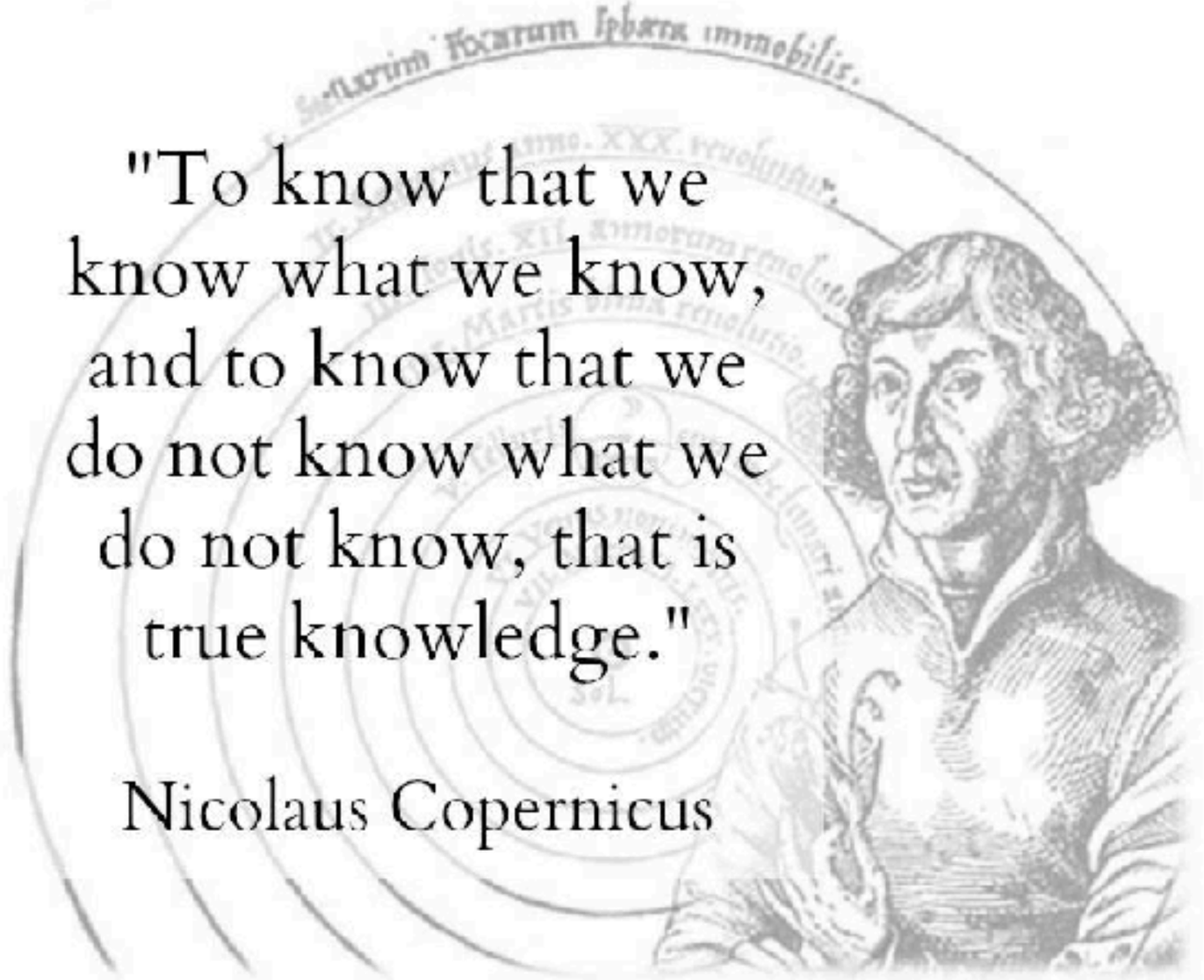
In reinforcement learning (RL), the goal of few-shot meta-learning is to enable an agent to **quickly acquire a policy for a new test task** using only a small amount of experience in the test setting

https://arxiv.org/abs/1703.03400

Uncertainty in Deep Learning

Uncertainty in Deep Learning

What we do not know



Uncertainty in Deep Learning

What we do not know

Uncertainty in Deep Learning

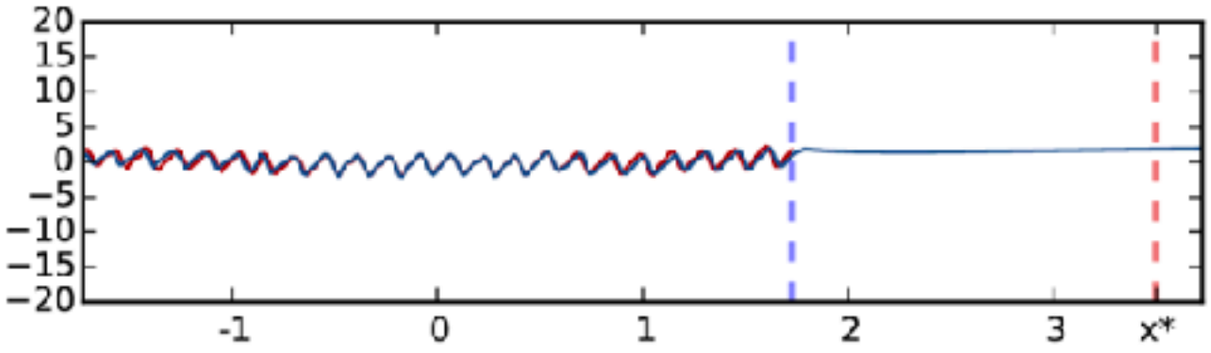
What we do not know

Uncertainty in Deep Learning

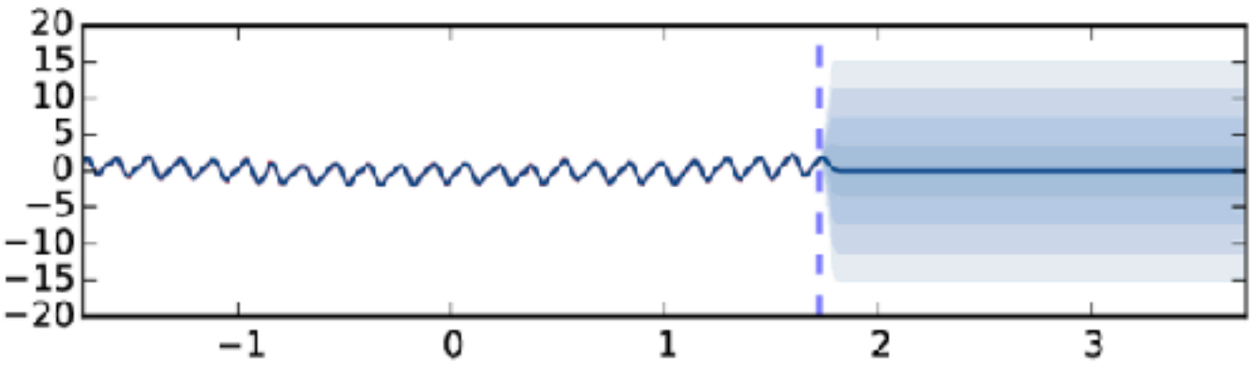
What we do not know

Uncertainty in Deep Learning

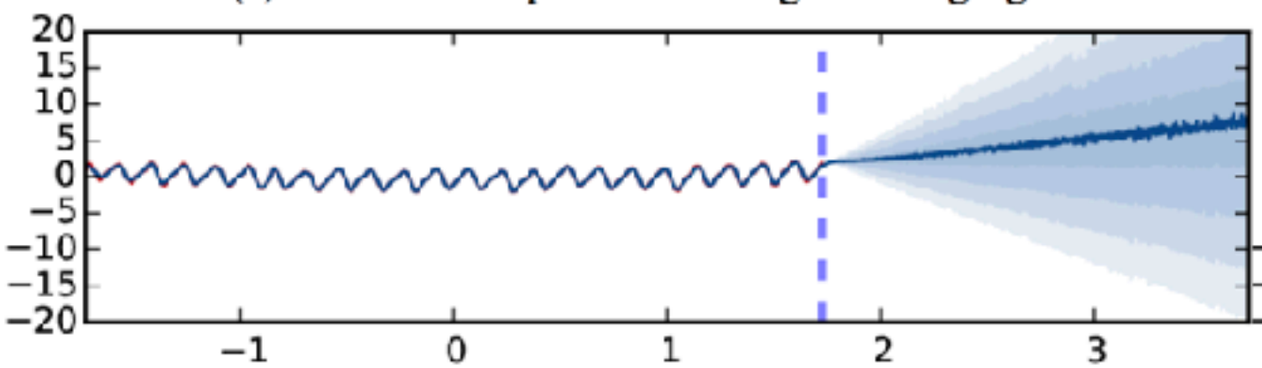
Dropout as Bayes Approximation



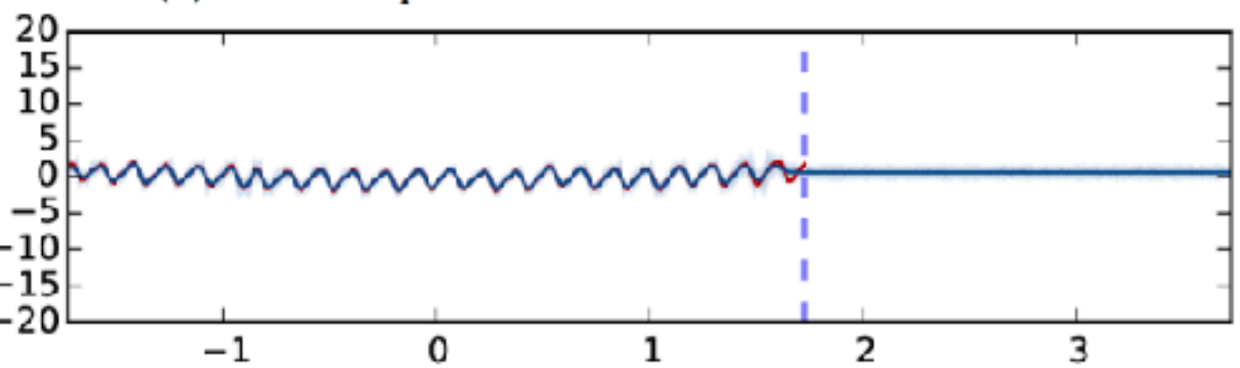
(a) Standard dropout with weight averaging



(b) Gaussian process with SE covariance function



(c) MC dropout with ReLU non-linearities



(d) MC dropout with TanH non-linearities

https://arxiv.org/abs/1506.02142

Uncertainty in Deep Learning

Dropout as Bayes Approximation

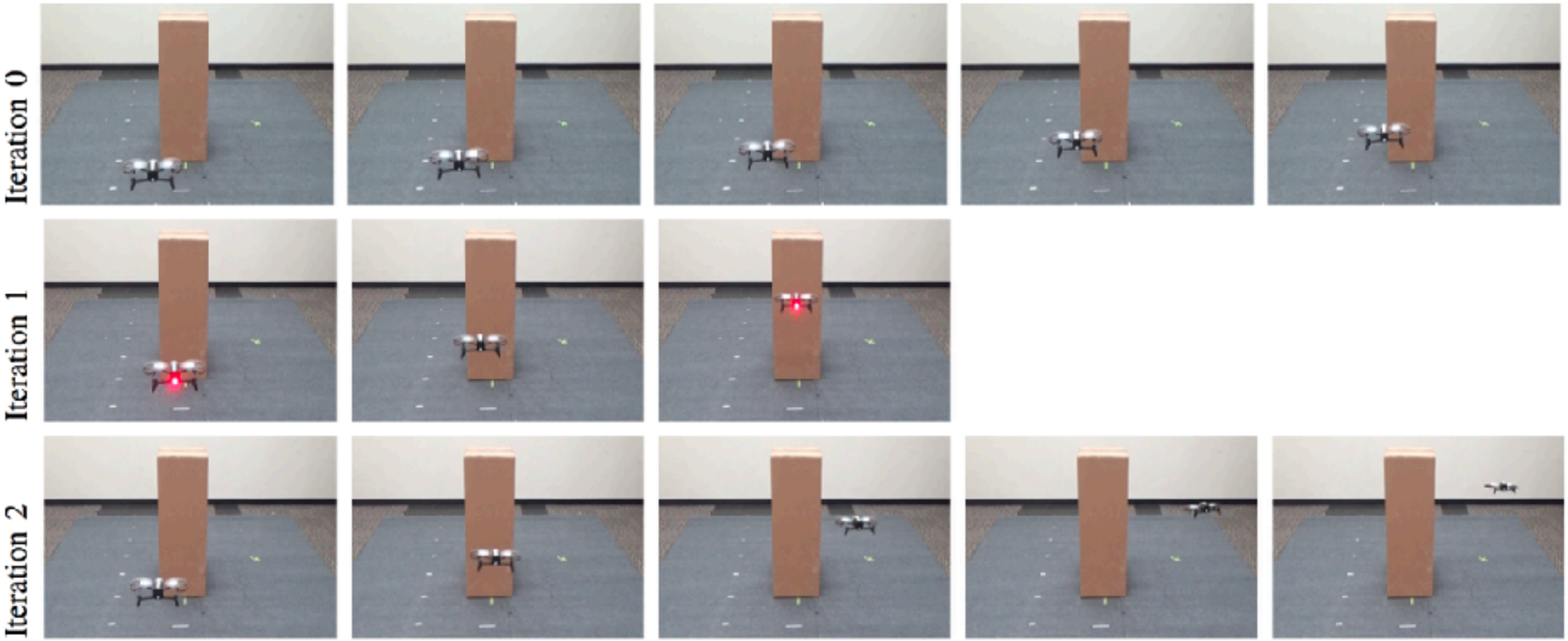
$$\begin{aligned}\mathrm{Var}_{q(\mathbf{y}^*|\mathbf{x}^*)}(\mathbf{y}^*) &\approx \tau^{-1}\mathbf{I}_D \\ &+ \frac{1}{T}\sum_{t=1}^{T}\widehat{\mathbf{y}}^*(\mathbf{x}^*, \mathbf{W}_1^t, ..., \mathbf{W}_L^t)^T\widehat{\mathbf{y}}^*(\mathbf{x}^*, \mathbf{W}_1^t, ..., \mathbf{W}_L^t) \\ &- \mathbb{E}_{q(\mathbf{y}^*|\mathbf{x}^*)}(\mathbf{y}^*)^T\mathbb{E}_{q(\mathbf{y}^*|\mathbf{x}^*)}(\mathbf{y}^*)\end{aligned}$$

which equals the sample variance of T stochastic forward passes through the NN plus the inverse model precision.

https://arxiv.org/abs/1506.02142

Uncertainty in Deep Learning

Uncertainty-Aware RL



https://arxiv.org/abs/1702.01182

Uncertainty in Deep Learning

Uncertainty-Aware RL

Algorithm 1 Neural net training with bootstrapping and dropout

input: dataset $\mathcal{D} = \{\mathbf{x}_t^{(i)}, \mathbf{u}_{t:t+H}^{(i)}, \mathbf{o}_t^{(i)}\}$, neural network model NN
for $b = 1$ to B **do**
  Sample a dataset of subsequences $\mathcal{D}^{(b)}$ from the full dataset $\mathcal{D}$ with replacement
  Initialize neural network $\text{NN}^{(b)}$ with random weights
  for number of SGD iterations **do**
    Sample datapoint $(\mathbf{x}_t, \mathbf{u}_{t:t+H}, \mathbf{o}_t)$ from $\mathcal{D}^{(b)}$
    Sample $\text{NN}_d^{(b)}$ by masking the units in $\text{NN}^{(b)}$ using dropout
    Run forward pass on $\text{NN}_d^{(b)}$ using $(\mathbf{x}_t, \mathbf{u}_{t:t+H}, \mathbf{o}_t)$
    Run backward pass on $\text{NN}_d^{(b)}$ to get gradient $g_d^{(b)}$
    Update model $\text{NN}^{(b)}$ parameters using $g_d^{(b)}$
  end for
end for

https://arxiv.org/abs/1702.01182

Uncertainty in Deep Learning

Novelty Detection

(a) LIDAR-mapped hallway environment for training and testing.

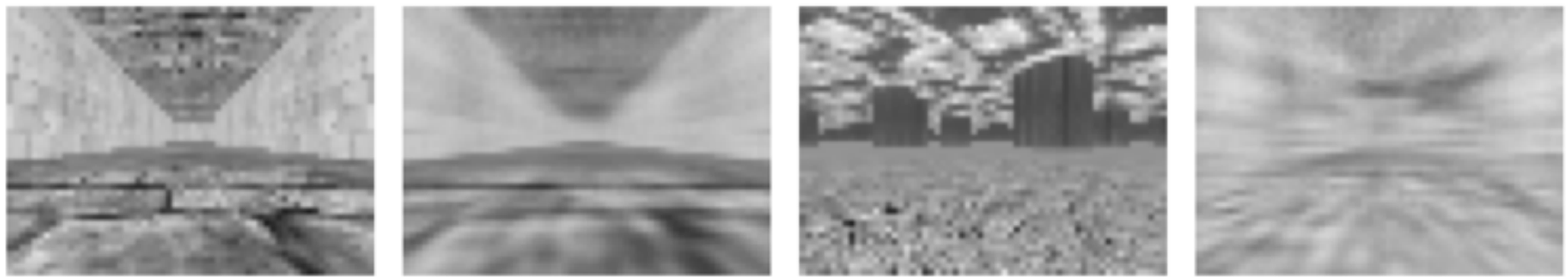
(b) Xtion-mapped novel environment used for LIDAR-free testing.

http://roboticsproceedings.org/rss13/p64.pdf

Uncertainty in Deep Learning

Novelty Detection

(a) Input. (b) Output. (c) Input. (d) Output.

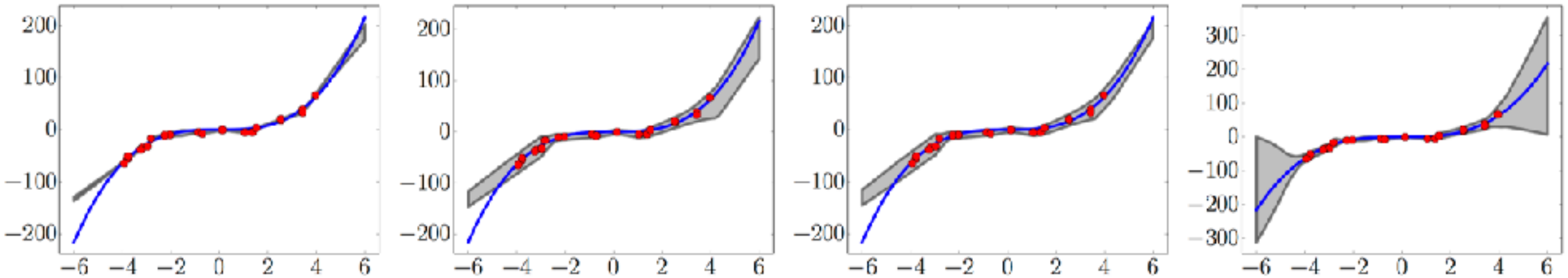


(a) Input. (b) Output. (c) Input. (d) Output.

http://roboticsproceedings.org/rss13/p64.pdf

Uncertainty in Deep Learning

Deep Ensemble



Empirical variance (5) Learning variance (1) Adversarial Ensemble (5)

https://arxiv.org/abs/1612.01474

Uncertainty in Deep Learning

Deep Ensemble

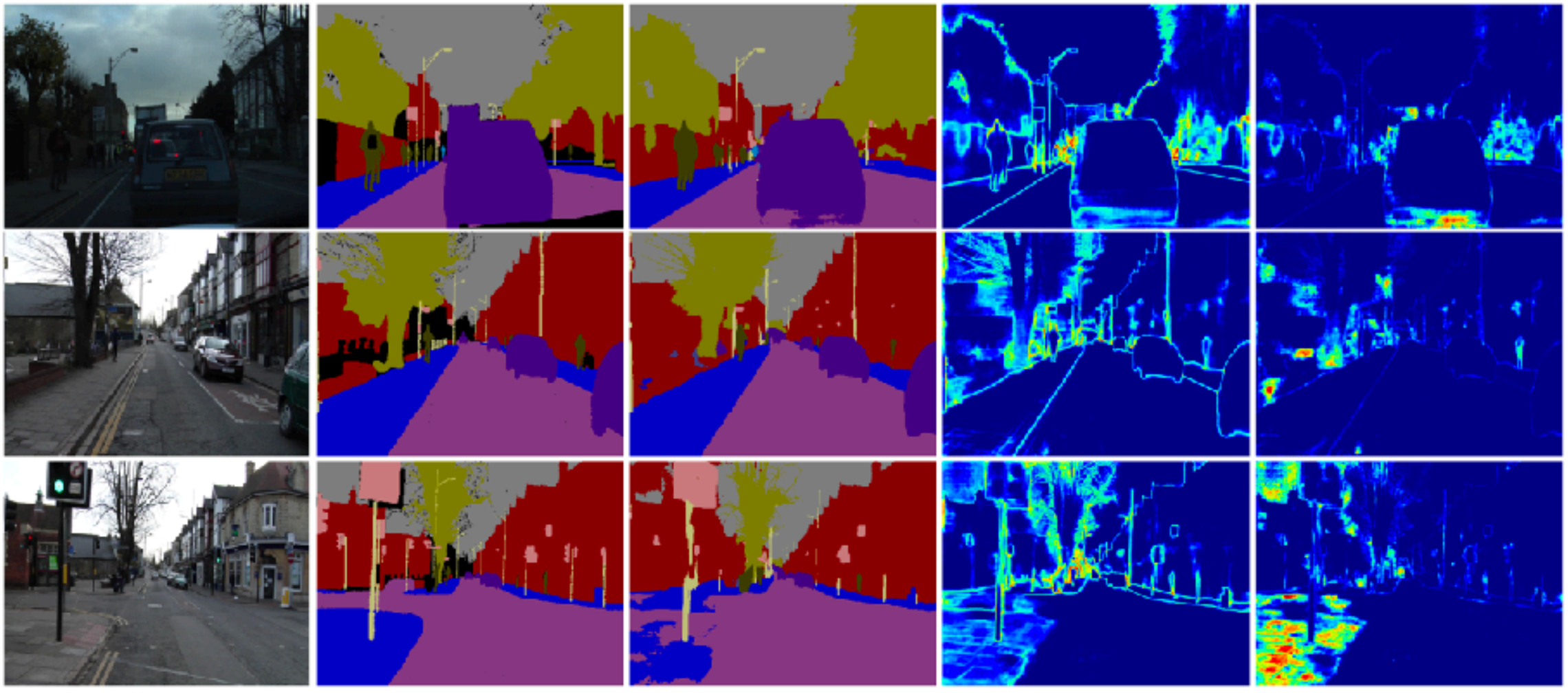
Algorithm 1 Pseudocode of the training procedure for our method

1: Initialise $\theta_1, \theta_2, \ldots, \theta_M$ randomly
2: **for** $m = 1 : M$ **do** ▷ *train networks independently in parallel*
3: Sample data point n_m randomly for each net ▷ *single n_m for clarity, minibatch in practice*
4: Generate adversarial example using $\mathbf{x}'_{n_m} = \mathbf{x}_{n_m} + \epsilon \operatorname{sign}\left(\nabla_{\mathbf{x}_{n_m}} \ell(\theta_m, \mathbf{x}_{n_m}, y_{n_m})\right)$
5: Minimise $\ell(\theta_m, \mathbf{x}_{n_m}, y_{n_m}) + \ell(\theta_m, \mathbf{x}'_{n_m}, y_{n_m})$ w.r.t. θ_m ▷ *adversarial training*

https://arxiv.org/abs/1612.01474

Uncertainty in Deep Learning

Epistemic and Aleatoric Uncertainty



(a) Input Image (b) Ground Truth (c) Semantic Segmentation (d) Aleatoric Uncertainty (e) Epistemic Uncertainty

https://arxiv.org/abs/1703.04977

Uncertainty in Deep Learning

Epistemic and Aleatoric Uncertainty

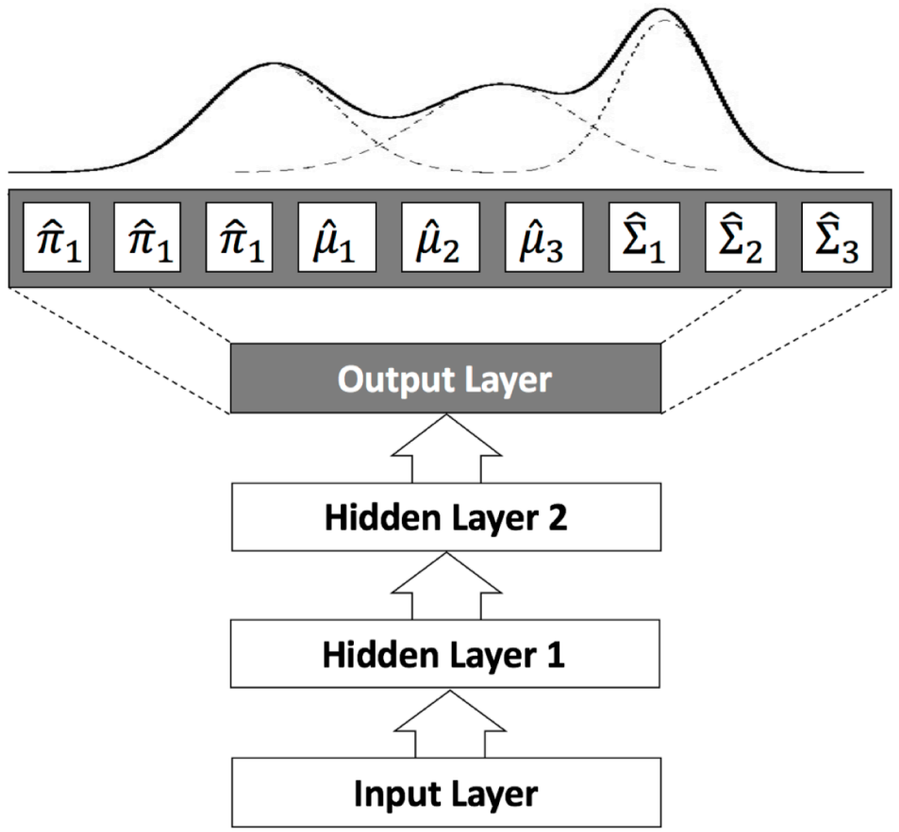
$$\mathrm{Var}(x) \approx \frac{1}{T}\sum_{t=1}^{T}\hat{x}_t^2 - \left(\frac{1}{T}\sum_{t=1}^{T}\hat{x}_t\right)^2 + \frac{1}{T}\sum_{t=1}^{T}\hat{\sigma}_t^2$$

with $\{\hat{x}_t, \hat{\sigma}_t^2\}_{t=1}^{T}$ a set of T sampled outputs: $\hat{x}_t, \hat{\sigma}_t^2 = f^{\widehat{\mathbf{W}}_t}(I)$ for randomly masked weights $\widehat{\mathbf{W}}_t \sim q(\mathbf{W})$.

https://arxiv.org/abs/1703.04977

Uncertainty in Deep Learning

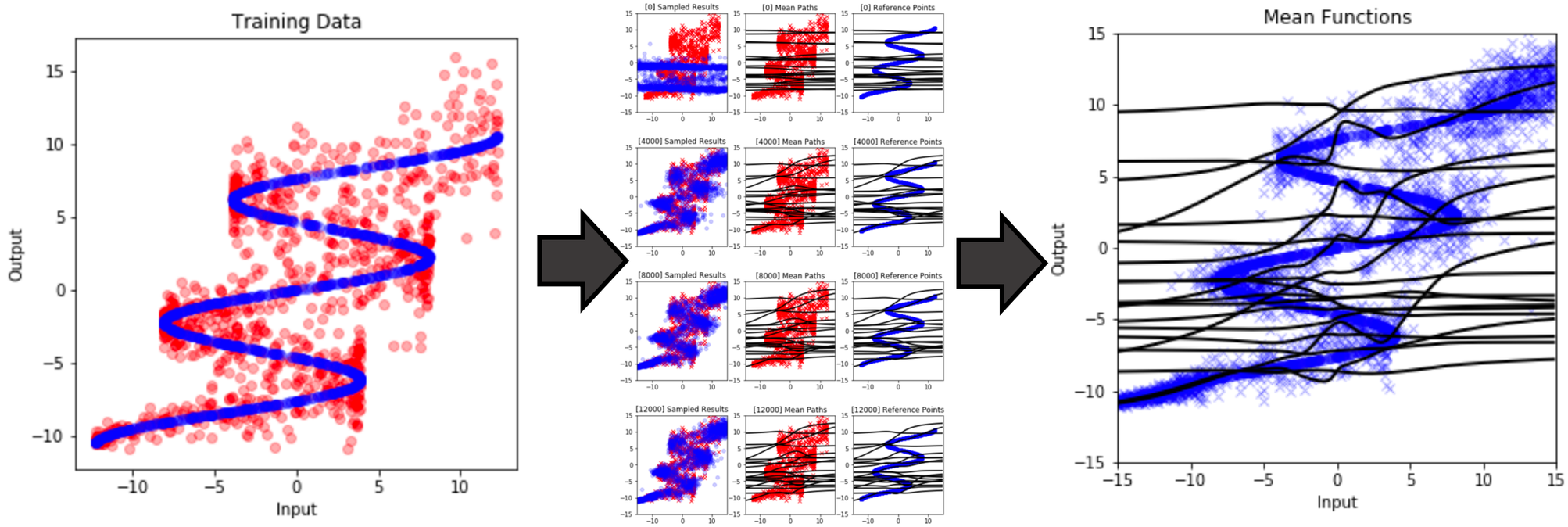
Uncertainty-Aware Learning from Demonstration



https://arxiv.org/abs/1709.02249

Uncertainty in Deep Learning

Uncertainty-Aware Learning from Demonstration



https://arxiv.org/abs/1709.02249

Uncertainty in Deep Learning

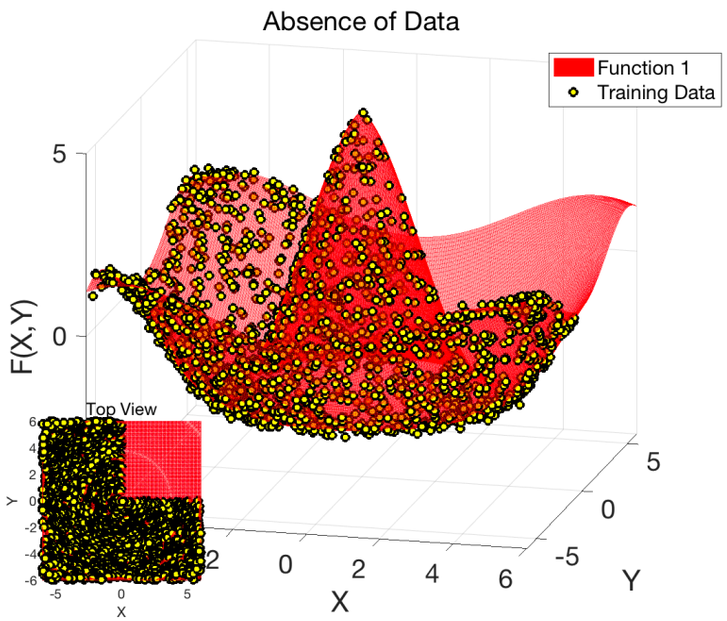
Uncertainty-Aware Learning from Demonstration

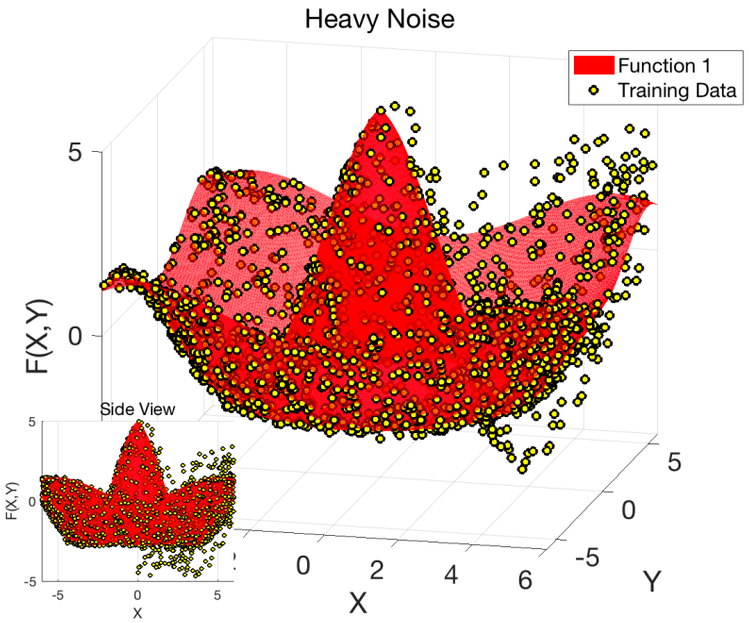
$$\mathbb{V}(\mathbf{y}|\mathbf{x}) = \overbrace{\sum_{j=1}^{K} \pi_j(\mathbf{x})\Sigma_j(\mathbf{x})}^{\textit{unexplained} \text{ variance}} + \underbrace{\sum_{j=1}^{K} \pi_j(\mathbf{x}) \left\| \mu_j(\mathbf{x}) - \sum_{k=1}^{K} \pi_k(\mathbf{x})\mu_k(\mathbf{x}) \right\|^2}_{\textit{explained} \text{ variance}}.$$

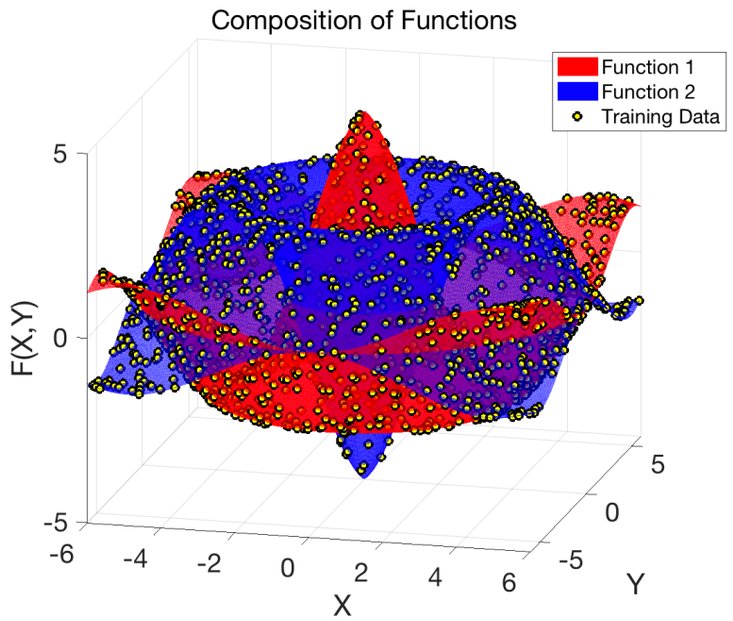
https://arxiv.org/abs/1709.02249

Uncertainty in Deep Learning

Uncertainty-Aware Learning from Demonstration







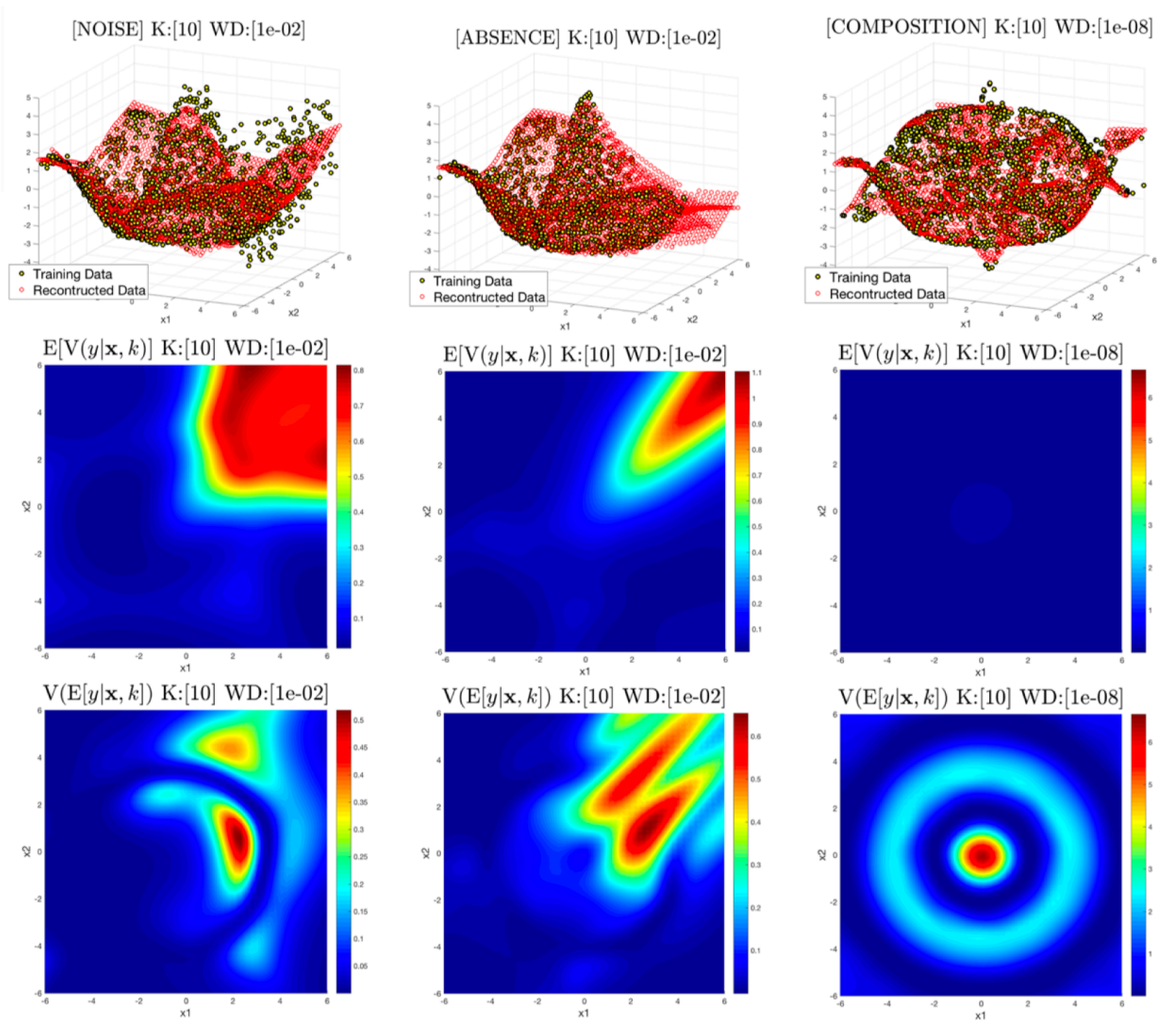
https://arxiv.org/abs/1709.02249

Uncertainty in Deep Learning

Uncertainty-Aware Learning from Demonstration

Unexplained Variance

Explained Variance

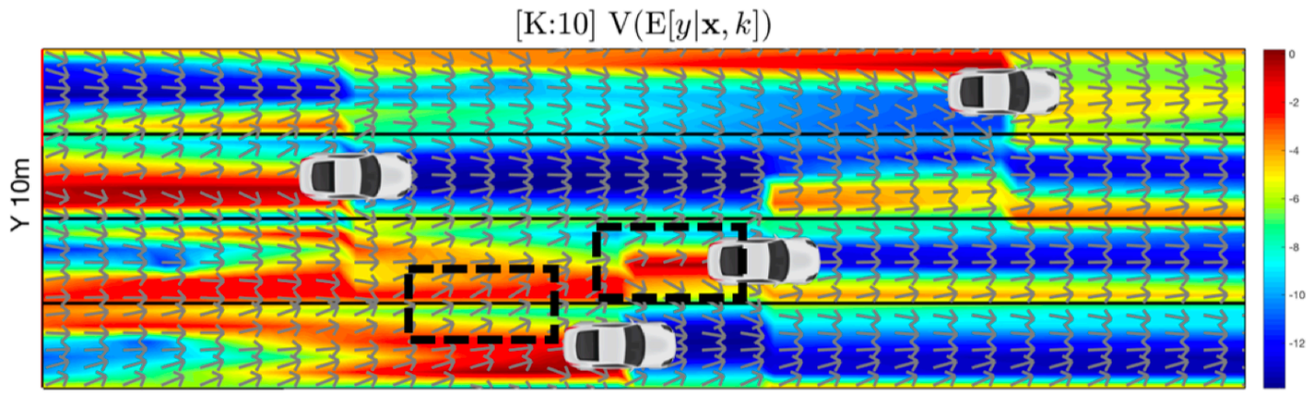


https://arxiv.org/abs/1709.02249

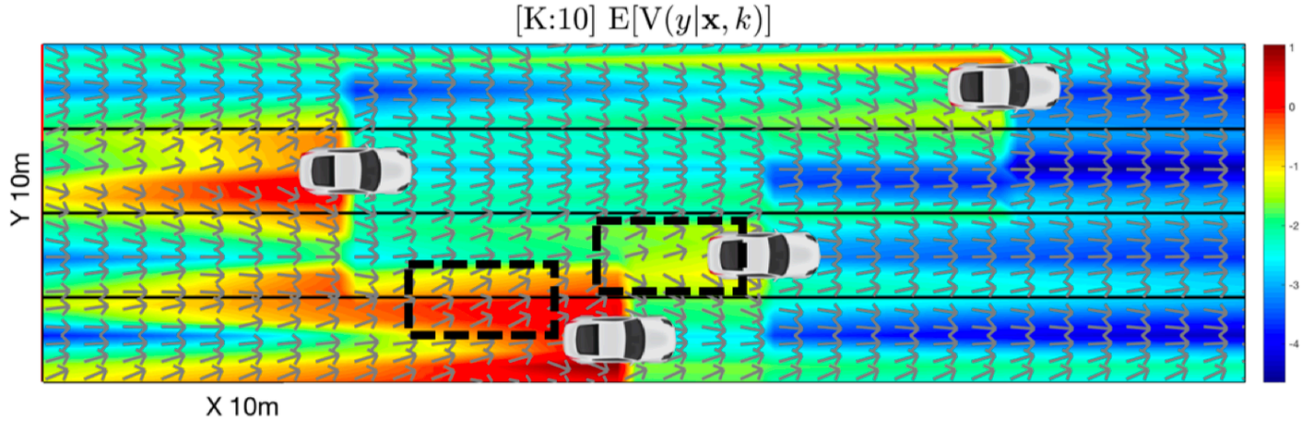
Uncertainty in Deep Learning

Uncertainty-Aware Learning from Demonstration

Explained
Variance



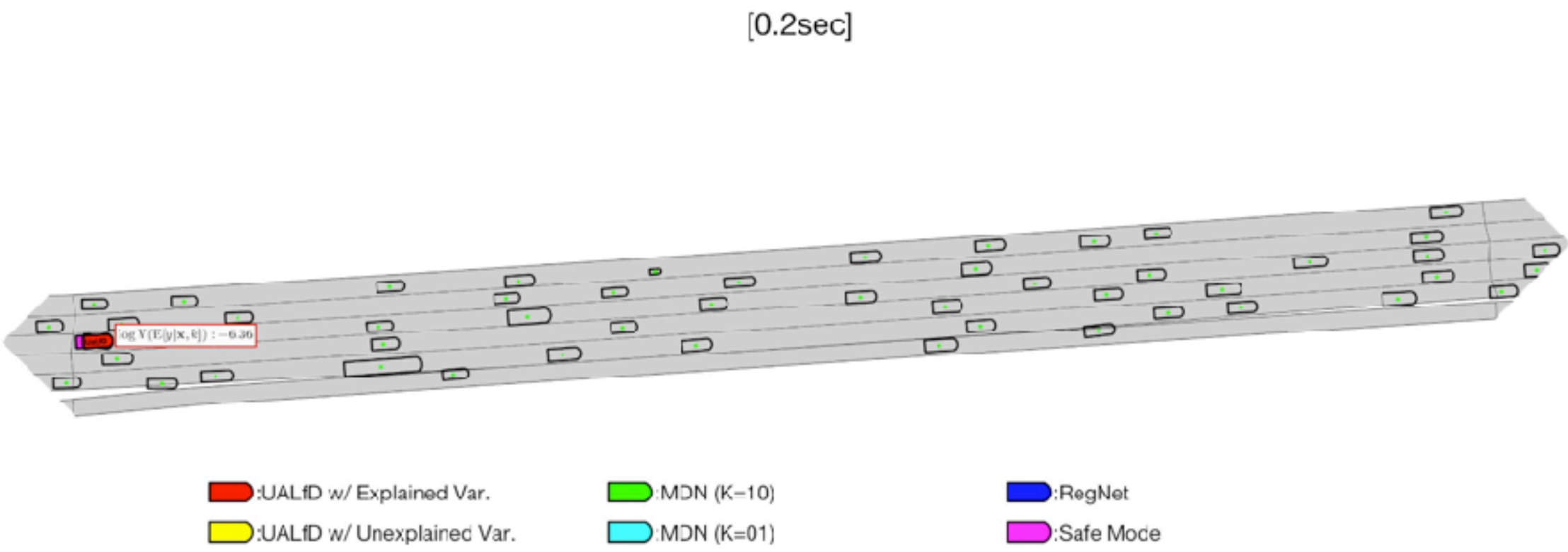
Unexplained
Variance



https://arxiv.org/abs/1709.02249

Uncertainty in Deep Learning

Uncertainty-Aware Learning from Demonstration



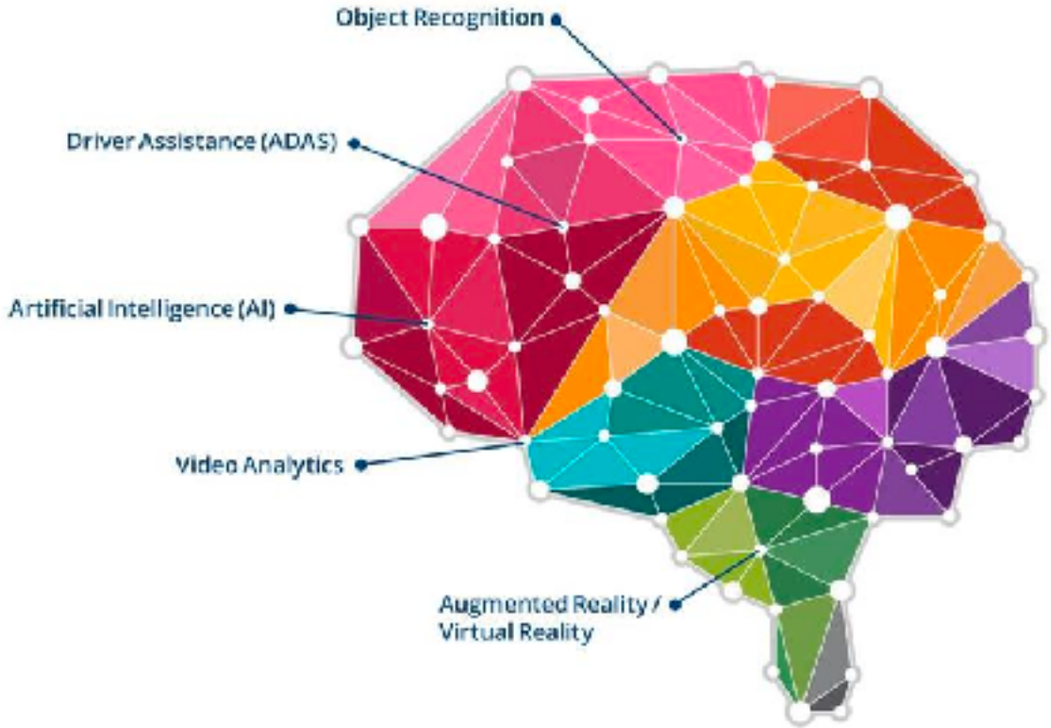
https://arxiv.org/abs/1709.02249

Conclusion

No more ImageNet challenge

Conclusion

Focus on applications



Conclusion

Consider real problems

Thank you for your attention.

Contact information (sungjoon.choi@cpslab.snu.ac.kr)