Занятие 8
Распознавание людей,
Wasserstein GAN

Дмитрий Яшунин, к.ф.-м.н
IntelliVision

Организационные моменты

Сессия:
на следующей неделе

27 декабря, в этой же аудитории, с 18:00 до 21:00

В прошлый раз: Recurrent Neural Networks

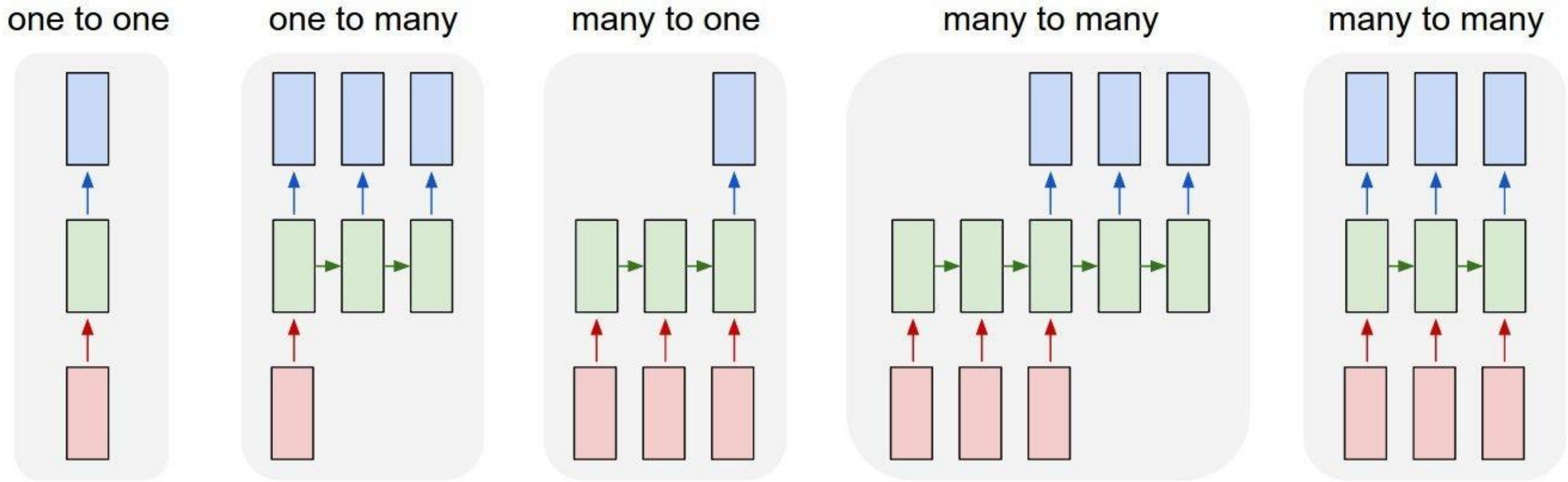


Image Classification

В прошлый раз: Recurrent Neural Networks

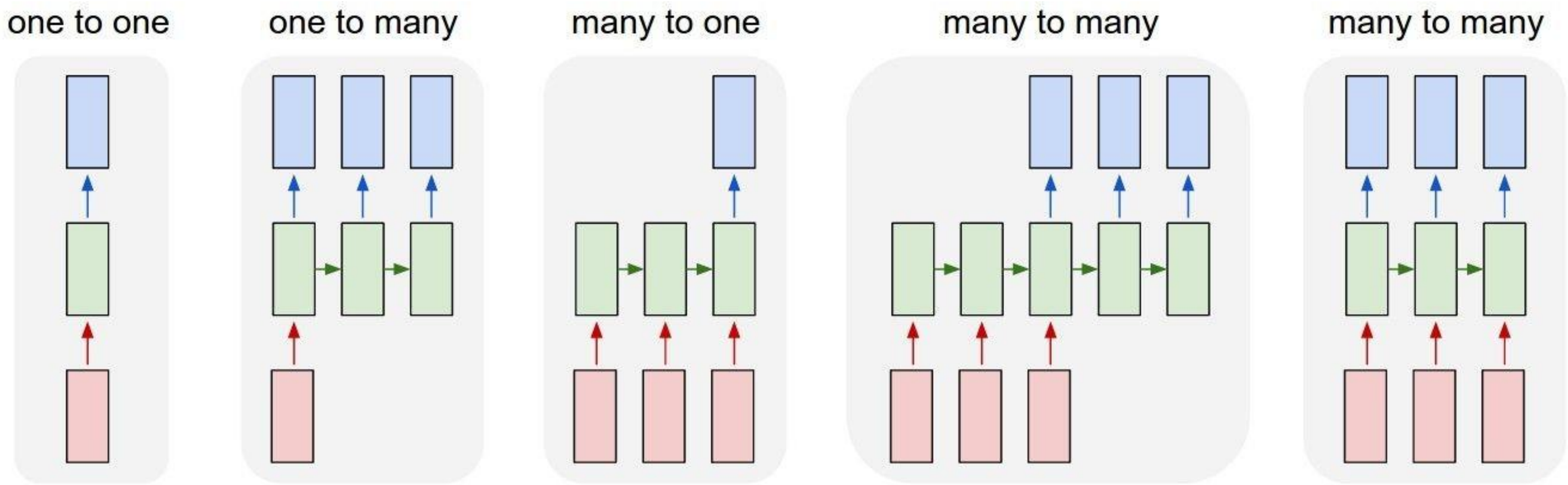


Image Classification **Image Captioning**

В прошлый раз: Recurrent Neural Networks

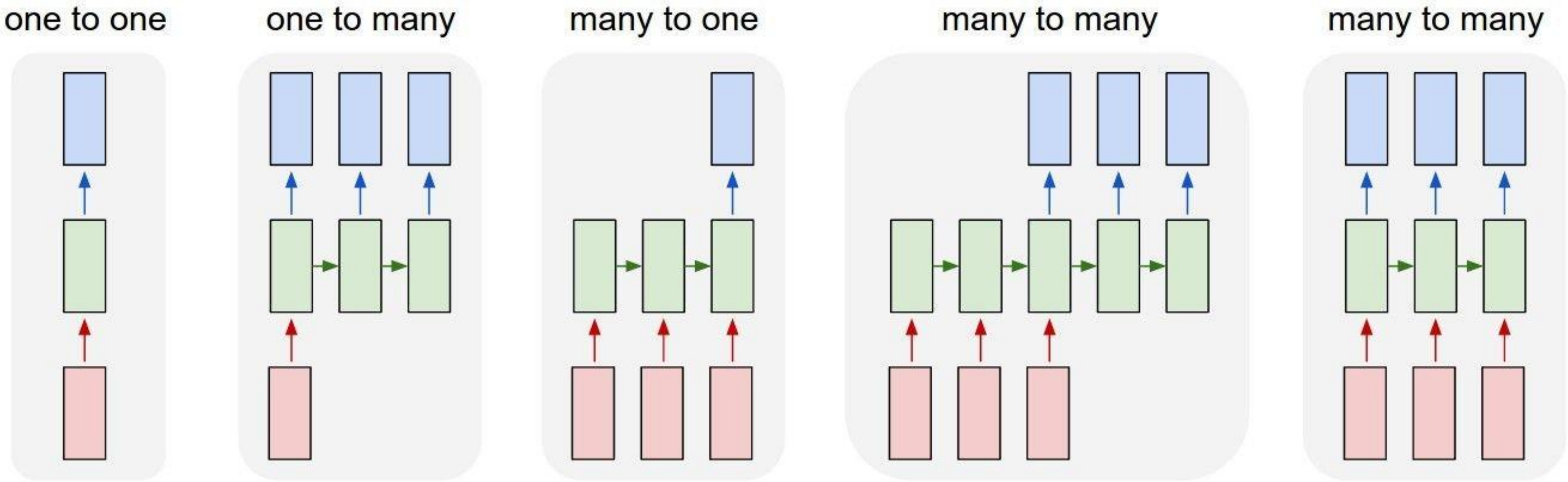


Image Classification

Image Captioning

Person reidentification by sequence of images

В прошлый раз: Recurrent Neural Networks

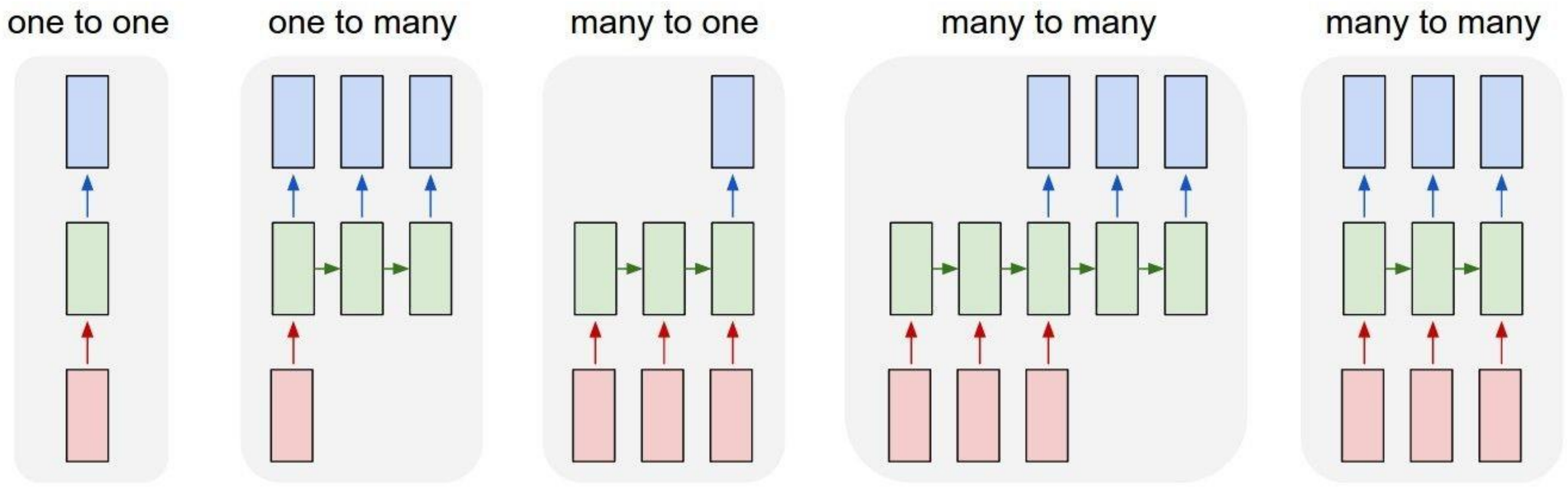


Image Classification

Image Captioning

Person reidentification

Machine Translation

В прошлый раз: Recurrent Neural Networks

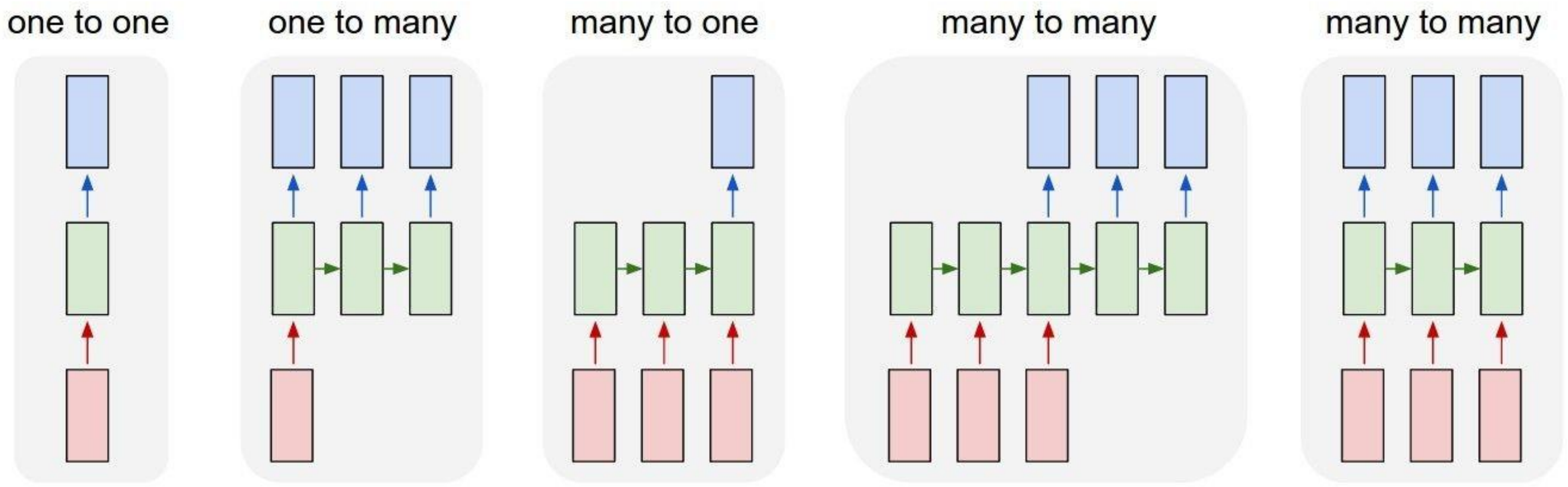


Image Classification

Image Captioning

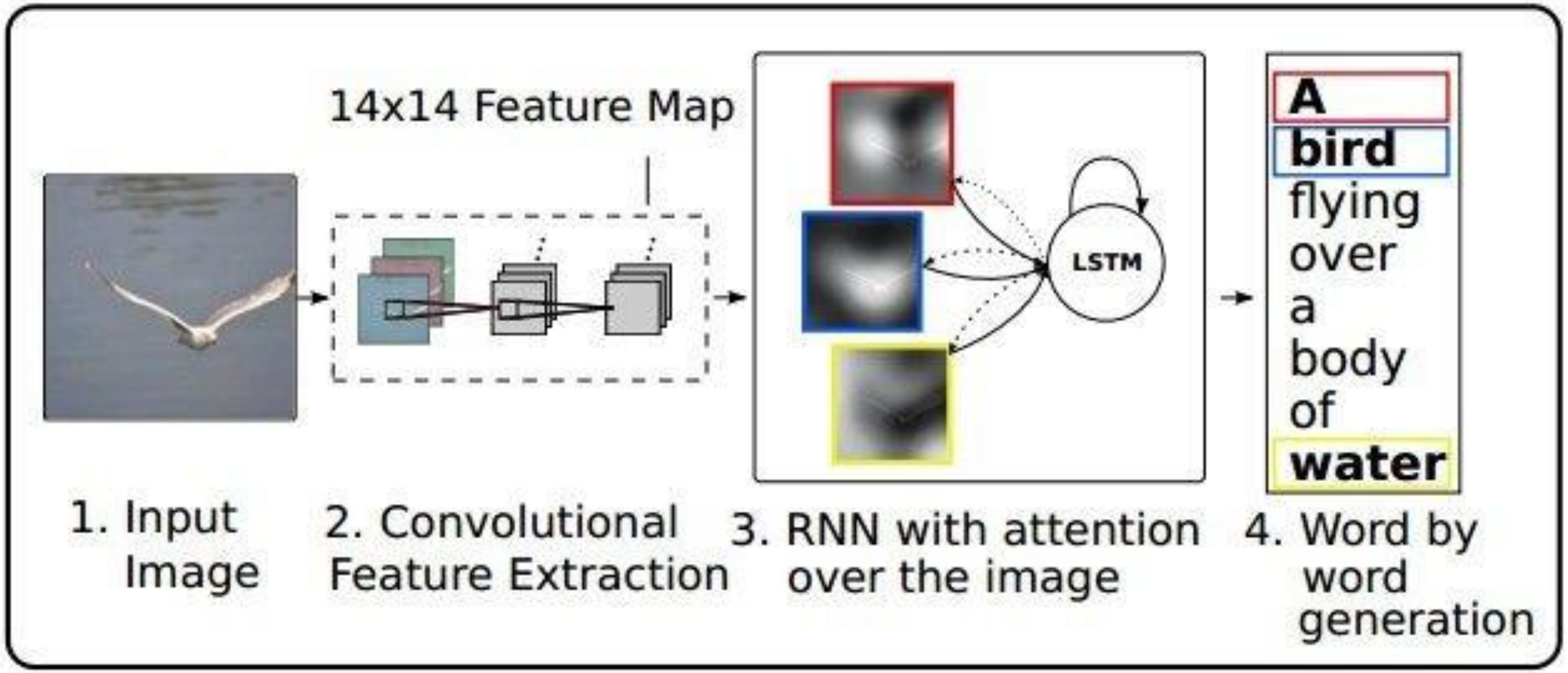
Person reidentification

Machine Translation

Video classification on frame level

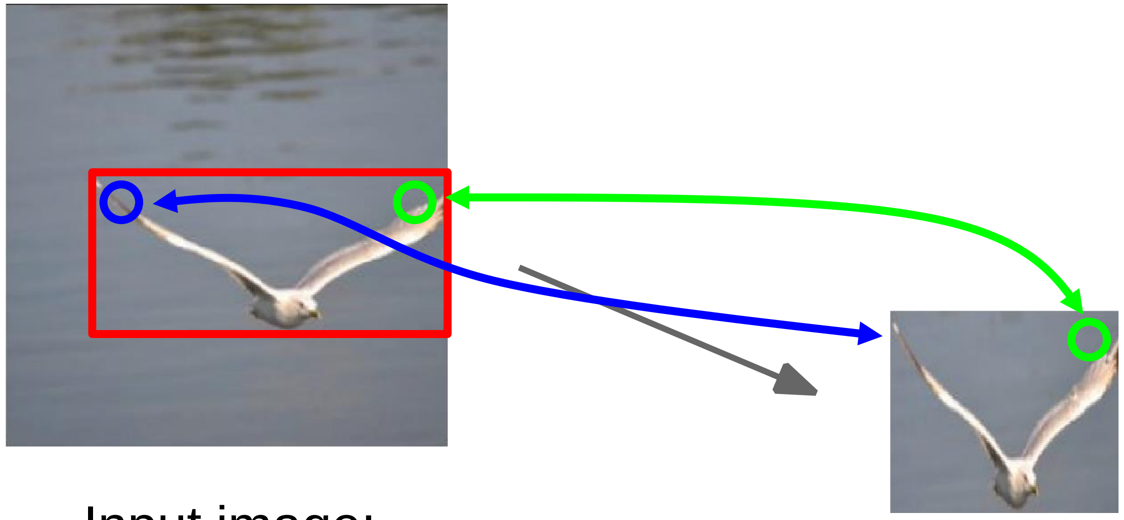
В прошлый раз: **Image Captioning with Attention**

RNN focuses its attention at a different spatial location when generating each word



K Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, 2015

В прошлый раз: **Spatial Transformer Networks**



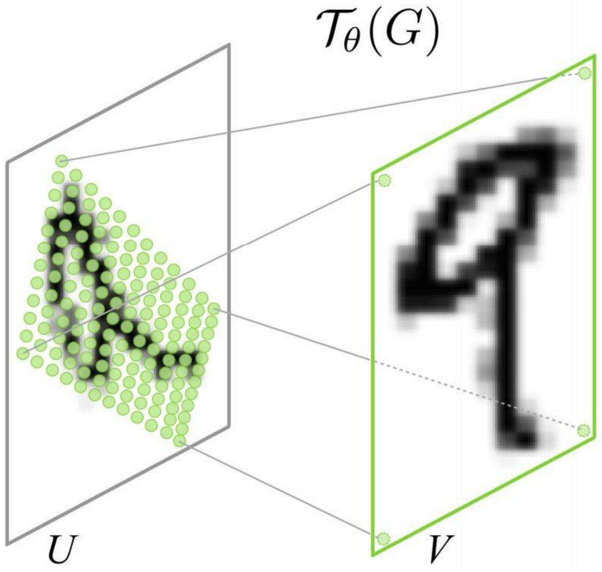
Input image:
H x W x 3

Cropped and
rescaled image:
X x Y x 3

Box Coordinates:
(x_c, y_c, w, h)

Idea: Function mapping *pixel coordinates* (x^t, y^t) of output to *pixel coordinates* (x^s, y^s) of input

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$



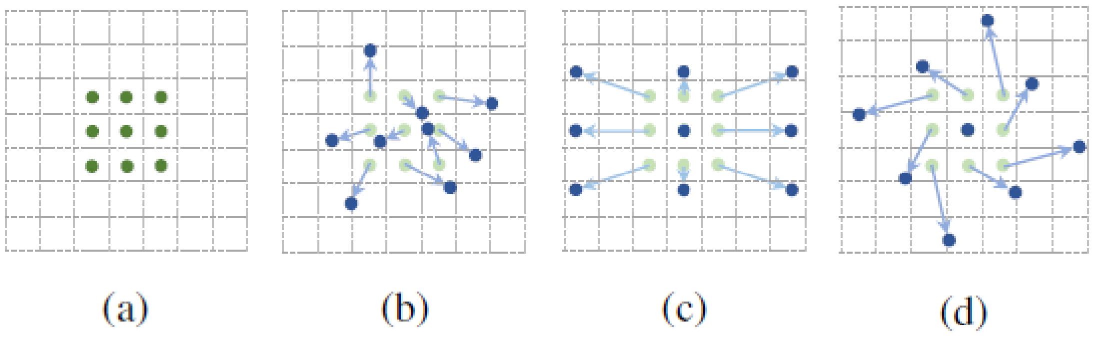
Repeat for all pixels in *output* to get a **sampling grid**

Then use **bilinear interpolation** to compute output

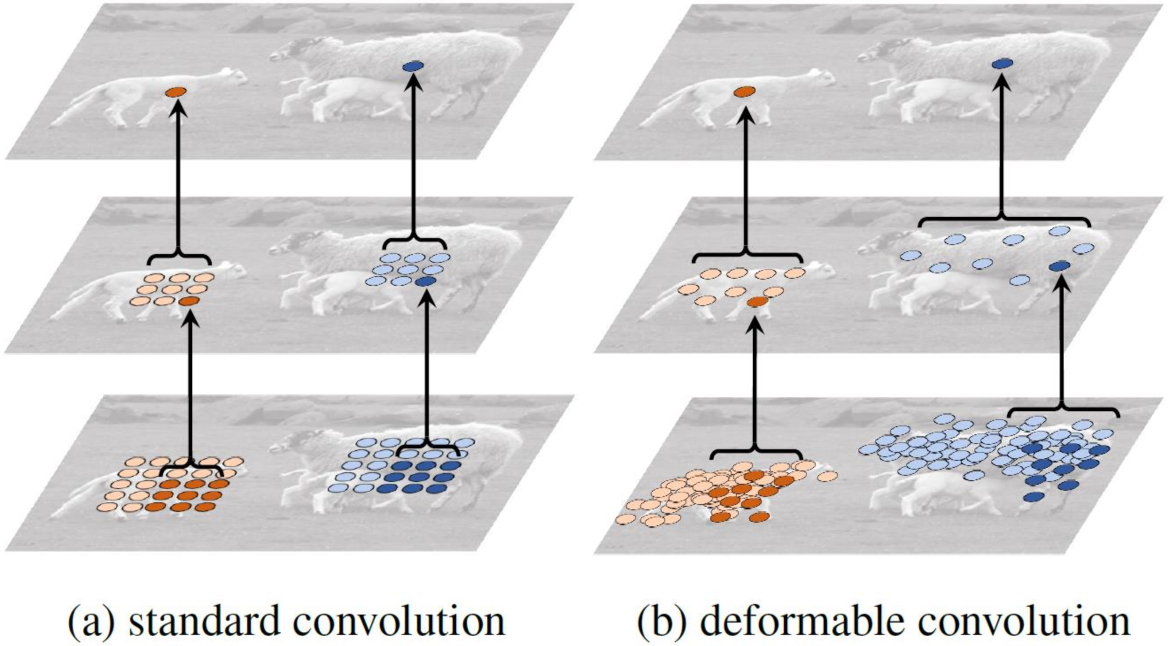
Jaderberg et al, “Spatial Transformer Networks”, NIPS 2015

В прошлый раз: **Deformable Convolutions**

Dynamic & learnable receptive field



(a) regular sampling grid of standard convolution
(b) deformed sampling locations with augmented offsets in deformable convolution
(c)(d) show that the deformable convolution generalizes various transformations for scale, (anisotropic) aspect ratio and rotation



(a) fixed receptive field in standard convolution
(b) adaptive receptive field in deformable convolution

Dai et al, "Deformable Convolutional Networks", 2017

Network Distillation

Transfer knowledge from an ensemble of models to a small single model

Teacher model 1 (Googlenet)

Teacher model 2 (ResNet)

Teacher model 3 (DenseNet)

soft targets

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

Student model

student model has much smaller model size

Сегодня:
Распознавание людей,
Wasserstein GAN

Image similarity: **Image Retrieval**

Query Retrieval

Example retrieval results on Stanford Online Products

Song et al, "Deep Metric Learning via Lifted Structured Feature Embedding", 2015

Image similarity: **Face recognition**

Example face images in MegaFace dataset

Wen et al, "A Discriminative Feature Learning Approach for Deep Face Recognition", 2016

Image similarity: **Person Reidentification**

Xiao et al, "Joint Detection and Identification Feature Learning for Person Search", 2017

Image similarity

Image similarity

Image retrieval

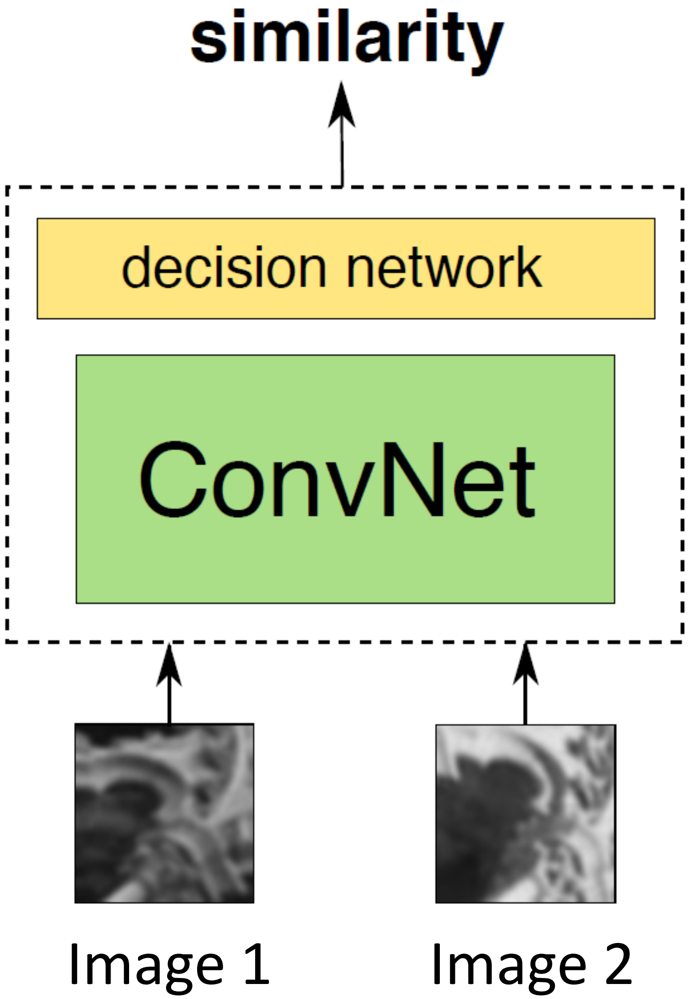
Zero/one shot learning

Face recognition

Person reidentification (ReID)

The same task:
find “similar” images

Image similarity: **Learning Similarity Function**



Zagoruyko and Komodakis, "Learning to Compare Image Patches via Convolutional Neural Networks", 2015

Image similarity: **Learning Similarity Function**

2-channel

Siamese network

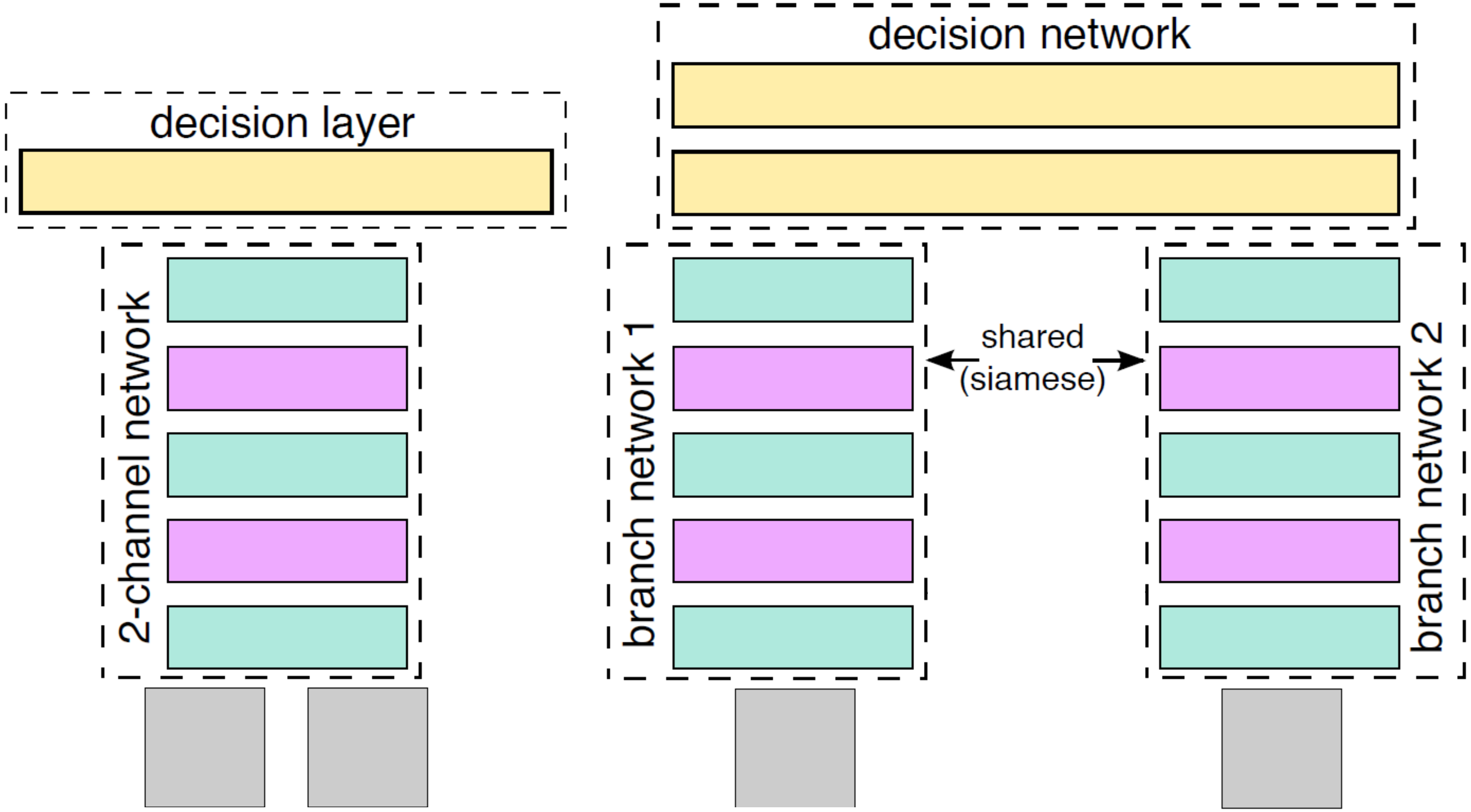


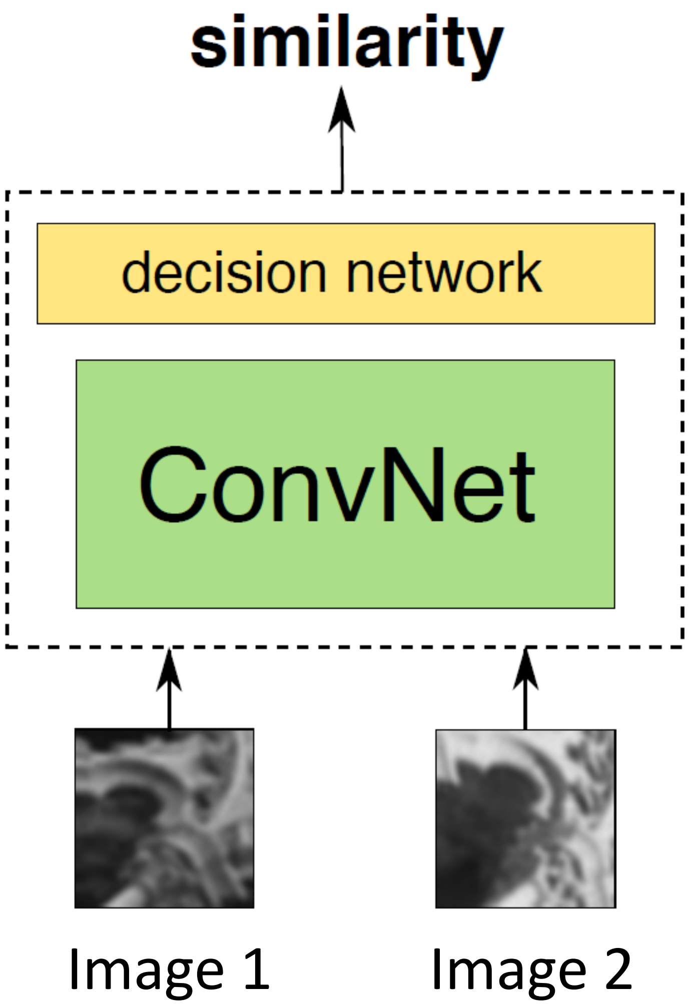
Image 1 Image 2

Image 1

Image 2

Zagoruyko and Komodakis, "Learning to Compare Image Patches via Convolutional Neural Networks", 2015

Image similarity: **Learning Similarity Function**



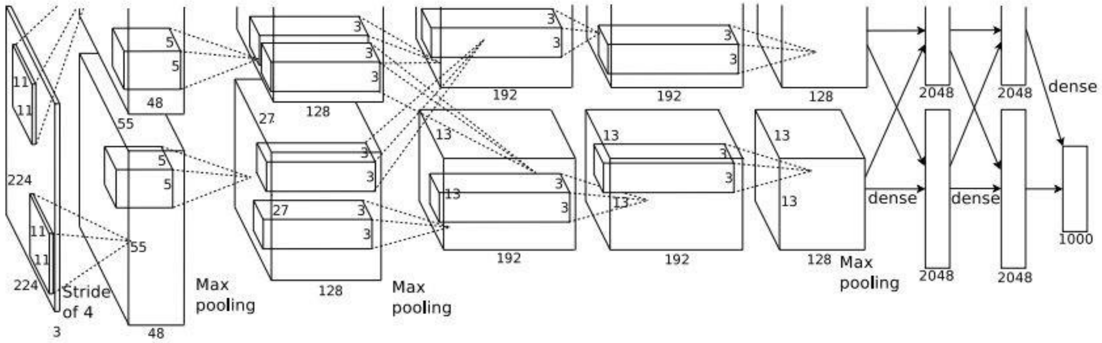
Very slow for many pairs

Zagoruyko and Komodakis, "Learning to Compare Image Patches via Convolutional Neural Networks", 2015

Image Similarity: **Embeddings**

Image

CNN

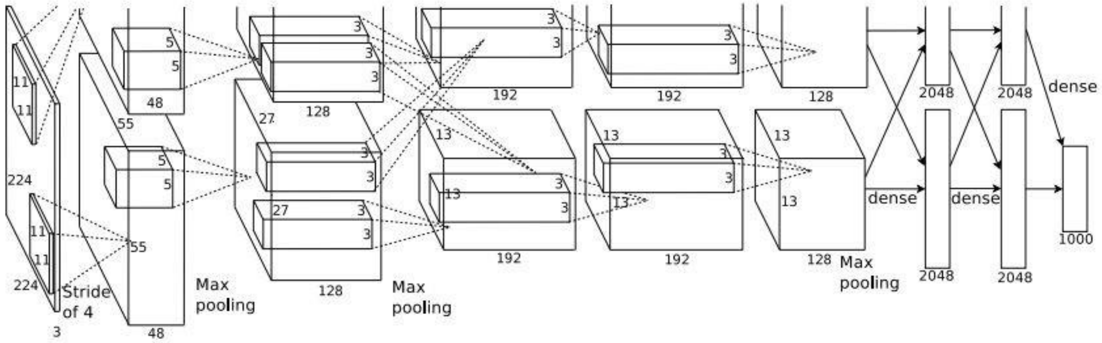


Embedding
vector of fixed size
typical 2^n: 128, 256, 512

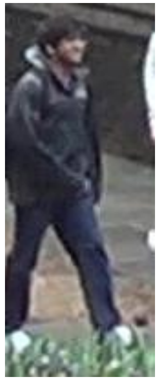
Image Similarity: **Embeddings**

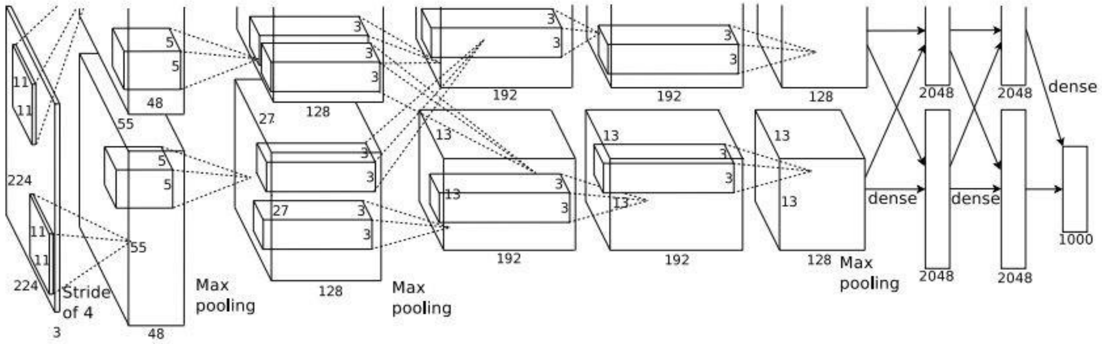
Image

CNN



Embedding 1





Embedding 2

Image Similarity: **Embeddings**

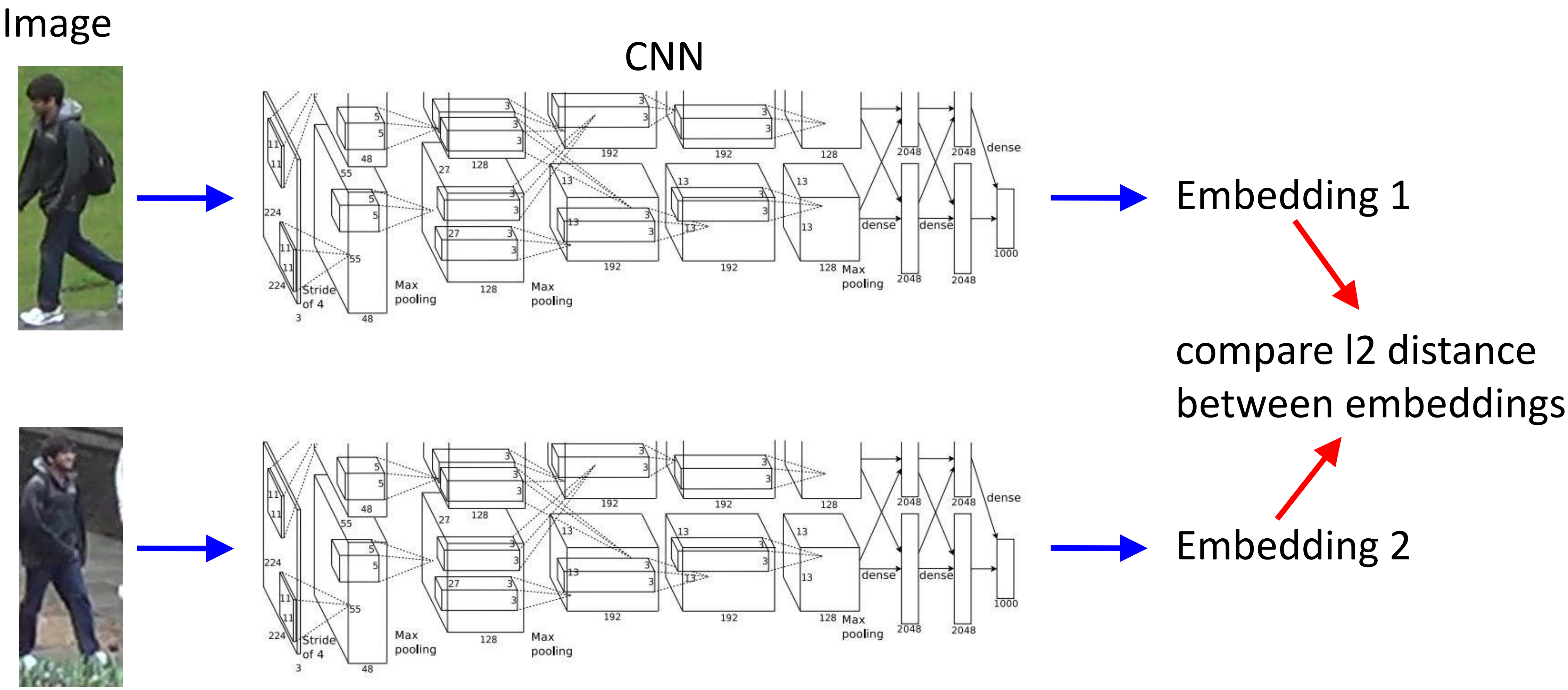


Image Similarity: Loss functions: **Softmax**

Solve classification task

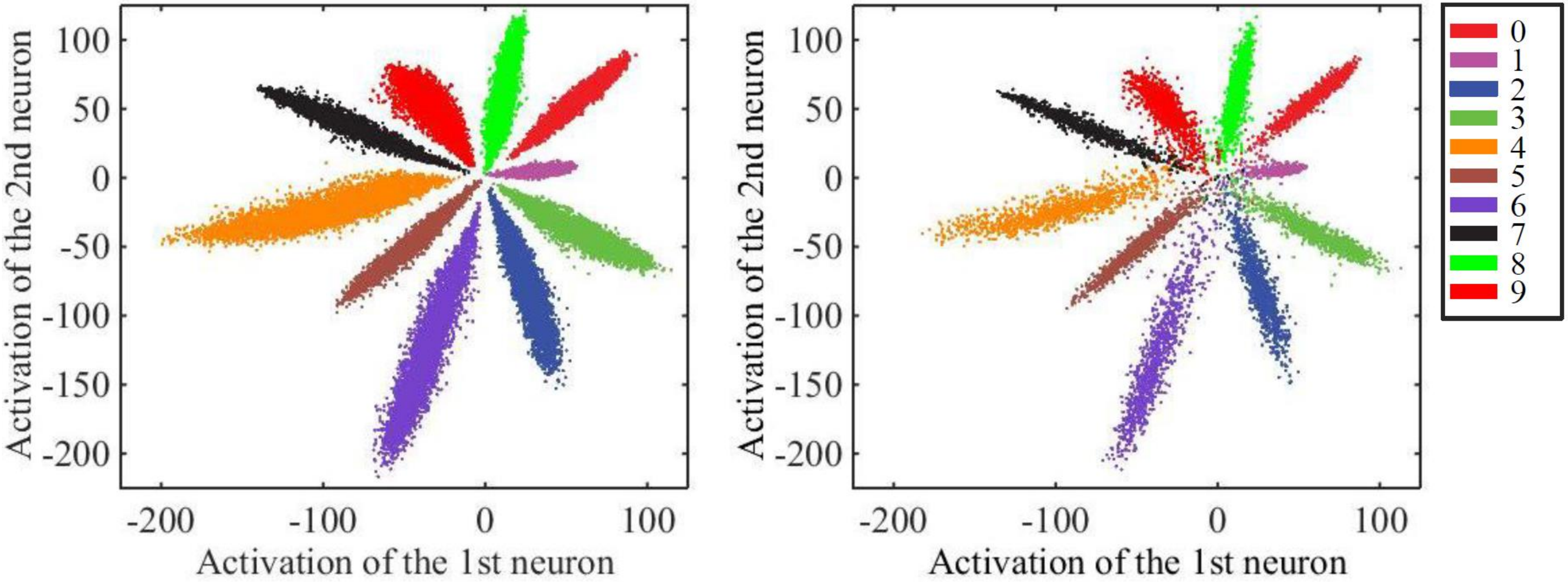
Use output of last layer before softmax as embedding

Test image L2 Nearest neighbors in feature space

Krizhevsky et al, “ImageNet Classification with Deep Convolutional Neural Networks”, NIPS 2012

Image Similarity: Loss functions: **Softmax**

MNIST handwritten digits 0-9



train

test

Wen et al, "A Discriminative Feature Learning Approach for Deep Face Recognition", 2016

Image Similarity: Loss functions: **Center Loss**

$$\mathcal{L}_S = -\sum_{i=1}^{m} \log \frac{e^{W_{y_i}^T \boldsymbol{x}_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T \boldsymbol{x}_i + b_j}}$$

Softmax loss

Wen et al, "A Discriminative Feature Learning Approach for Deep Face Recognition", 2016

Image Similarity: Loss functions: **Center Loss**

$$\mathcal{L}_S = -\sum_{i=1}^{m} \log \frac{e^{W_{y_i}^T \boldsymbol{x}_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T \boxed{\boldsymbol{x}_i} + b_j}}$$

Softmax loss

x_i – output features (before softmax layer)

Wen et al, "A Discriminative Feature Learning Approach for Deep Face Recognition", 2016

Image Similarity: Loss functions: **Center Loss**

$$\mathcal{L}_S = -\sum_{i=1}^{m} \log \frac{e^{W_{y_i}^T \boldsymbol{x}_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T \boldsymbol{x}_i + b_j}}$$

Softmax loss

x_i – output features (before softmax layer)

$$\mathcal{L}_C = \frac{1}{2}\sum_{i=1}^{m} \|\boldsymbol{x}_i - \boxed{\boldsymbol{c}_{y_i}}\|_2^2$$

Center loss: direct clustering of features around centers

c_{yi} – learnable centers for classes y_i

Wen et al, "A Discriminative Feature Learning Approach for Deep Face Recognition", 2016

Image Similarity: Loss functions: **Center Loss**

$$\mathcal{L}_S = -\sum_{i=1}^{m} \log \frac{e^{W_{y_i}^T \boldsymbol{x}_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T \boldsymbol{x}_i + b_j}}$$

Softmax loss

x_i – output features (before softmax layer)

$$\mathcal{L}_C = \frac{1}{2} \sum_{i=1}^{m} \|\boldsymbol{x}_i - \boldsymbol{c}_{y_i}\|_2^2$$

Center loss: direct clustering of features around centers

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_S + \lambda \mathcal{L}_C \\ &= -\sum_{i=1}^{m} \log \frac{e^{W_{y_i}^T \boldsymbol{x}_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T \boldsymbol{x}_i + b_j}} + \frac{\lambda}{2} \sum_{i=1}^{m} \|\boldsymbol{x}_i - \boldsymbol{c}_{y_i}\|_2^2\end{aligned}$$

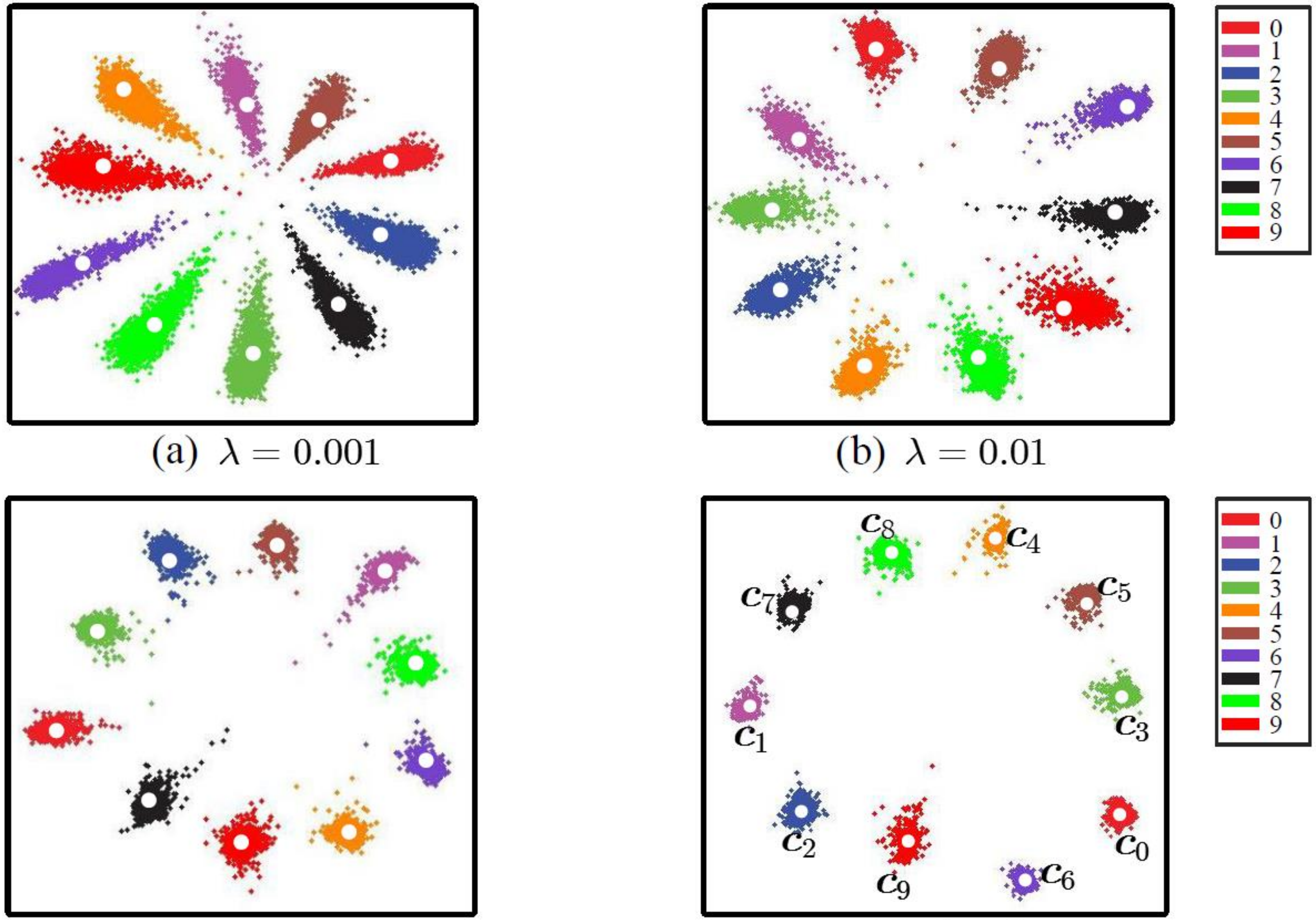
Total loss

Wen et al, "A Discriminative Feature Learning Approach for Deep Face Recognition", 2016

Image Similarity: Loss functions: **Center Loss**

MNIST handwritten digits 0-9

Feature clustering for softmax+center loss



(a) $\lambda = 0.001$ (b) $\lambda = 0.01$ (c) $\lambda = 0.1$ (d) $\lambda = 1$

Wen et al, "A Discriminative Feature Learning Approach for Deep Face Recognition", 2016

Image Similarity: Loss functions: **Contrastive loss**

$$L_i(X_1, X_2) = (1 - Y)D^2_{X_1, X_2} + Y\max\left(0, m - D_{X_1, X_2}\right)^2$$

Hadsell et al, "Dimensionality Reduction by Learning an Invariant Mapping", 2006

Image Similarity: Loss functions: **Contrastive loss**

X_1 and X_2 images to compare

$$L_i\boxed{(X_1, X_2)} = (1 - Y)D^2_{X_1, X_2} + Y\max\left(0, m - \boxed{D_{X_1, X_2}}\right)^2$$

D – distance between image embeddings

Hadsell et al, "Dimensionality Reduction by Learning an Invariant Mapping", 2006

Image Similarity: Loss functions: **Contrastive loss**

X_1 and X_2 images to compare

margin

$$L_i\boxed{(X_1, X_2)} = (1 - \boxed{Y})D^2_{X_1,X_2} + Y\max(0, \boxed{m} - \boxed{D_{X_1,X_2}})^2$$

Y=0 if X_1 and X_2 are similar
Y=1 if X_1 and X_2 dissimilar

D – distance between image embeddings

Hadsell et al, "Dimensionality Reduction by Learning an Invariant Mapping", 2006

Image Similarity: Loss functions: **Contrastive loss**

X_1 and X_2 images to compare

margin

$$L_i(\boxed{X_1, X_2}) = (1 - \boxed{Y})D^2_{X_1,X_2} + Y\max(0, \boxed{m} - \boxed{D_{X_1,X_2}})^2$$

D – distance between image embeddings

Y=0 if X_1 and X_2 are similar $\quad L_i(X_1, X_2) = D^2$

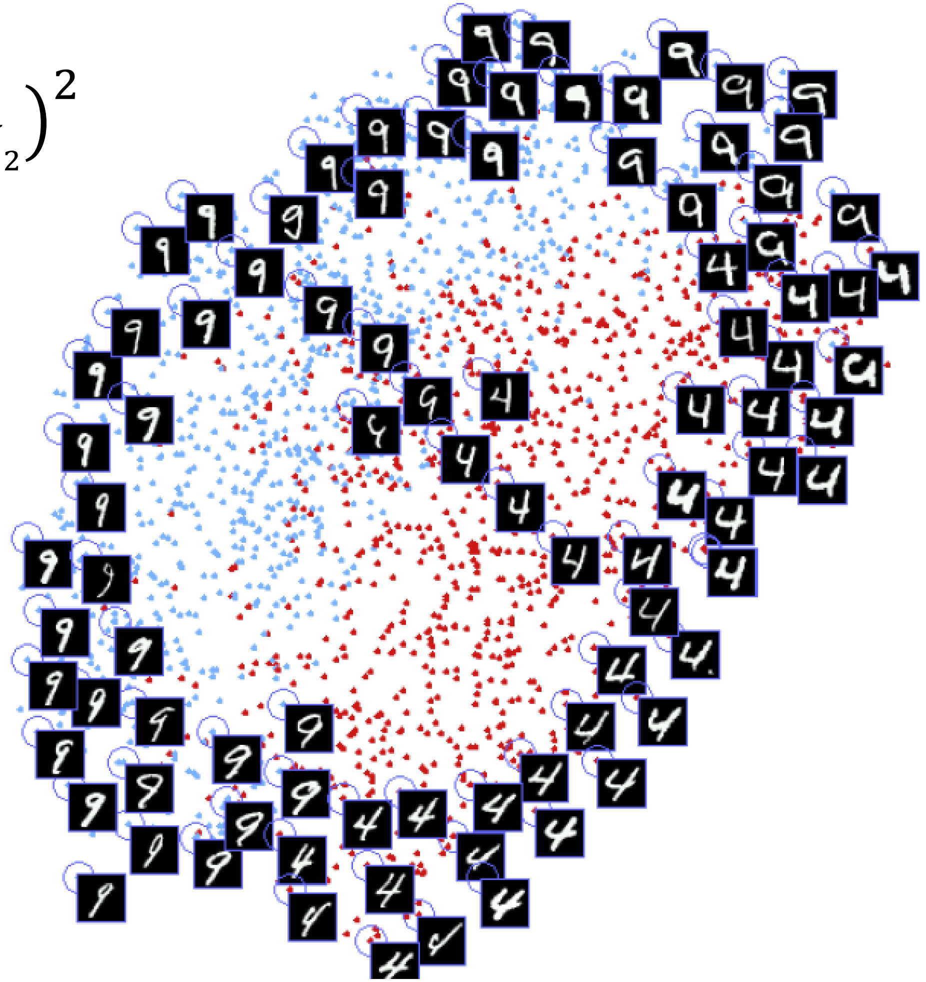
Y=1 if X_1 and X_2 dissimilar $\quad L_i(X_1, X_2) = \max(0, m - D)^2$

Hadsell et al, "Dimensionality Reduction by Learning an Invariant Mapping", 2006

Image Similarity: Loss functions: **Contrastive loss**

MNIST handwritten digits 0-9

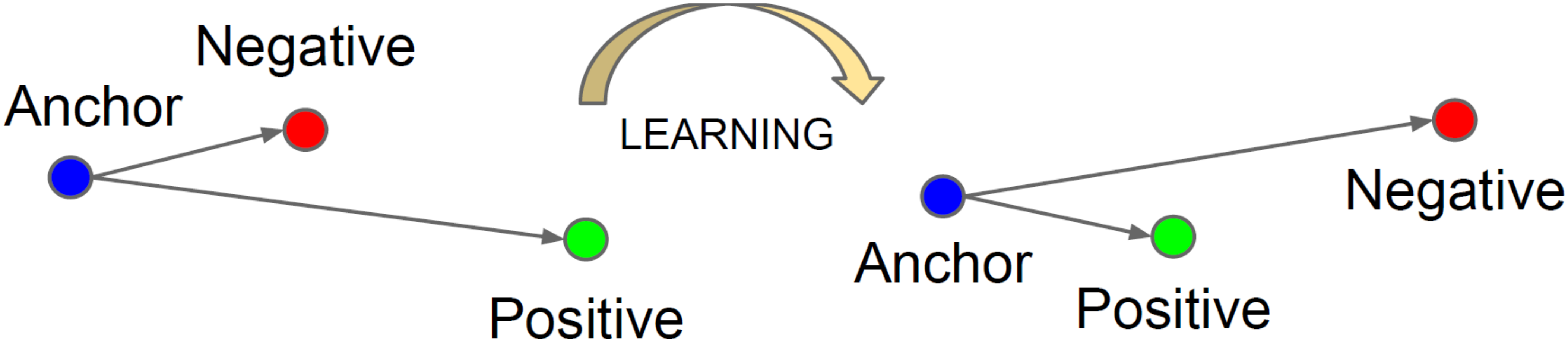
$$L_i(X_1, X_2) = (1 - Y)D^2_{X_1, X_2} + Y\max\left(0, m - D_{X_1, X_2}\right)^2$$



test

Hadsell et al, "Dimensionality Reduction by Learning an Invariant Mapping", 2006

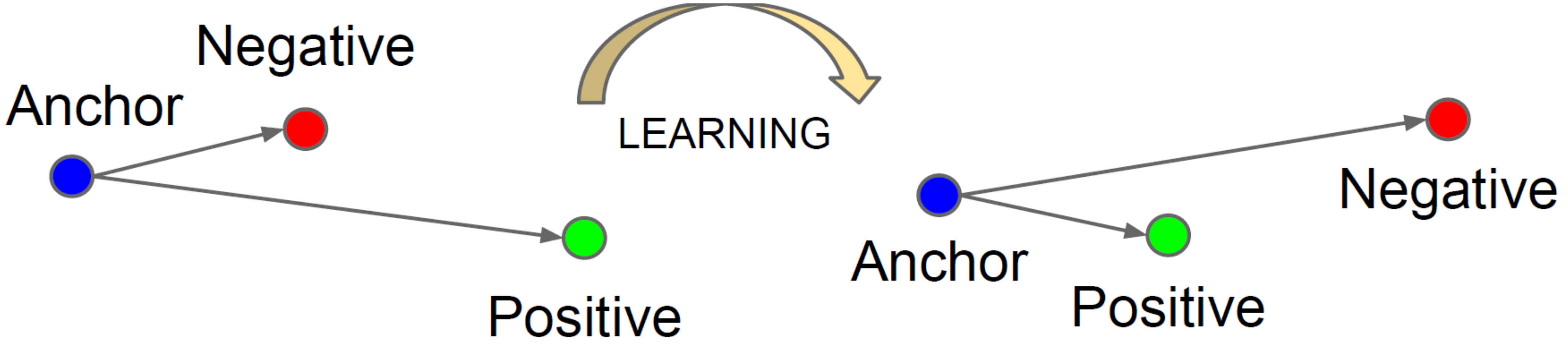
Image Similarity: Loss functions: **Triplet Loss**



$$L_i(a,p,n) = \max(0, D_{a,p}^2 - D_{a,n}^2 + m)$$

Schroff et al, "FaceNet: A Unified Embedding for Face Recognition and Clustering", 2015

Image Similarity: Loss functions: **Triplet Loss**



margin

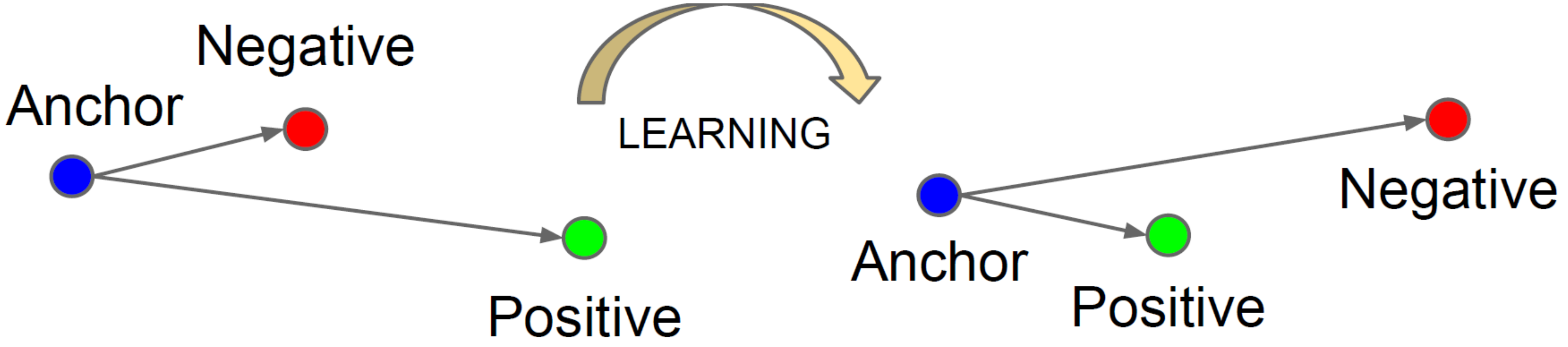
$$L_i(\boxed{a, p, n}) = \max(0, \boxed{D^2_{a,p}} - \boxed{D^2_{a,n}} + \boxed{m})$$

a – anchor
p – positive
n – negative

distances between image embeddings

Schroff et al, "FaceNet: A Unified Embedding for Face Recognition and Clustering", 2015

Image Similarity: Loss functions: **Triplet Loss**



$$L_i(a, p, n) = \max(0, D_{a,p}^2 - D_{a,n}^2 + m)$$

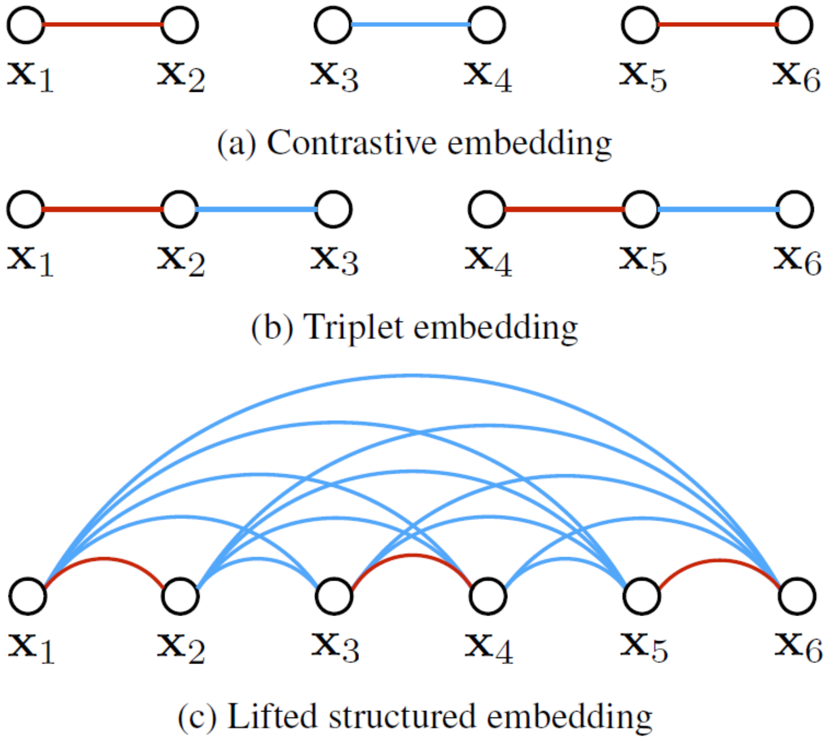
Hard negative mining:
for anchor-positive pair
search for difficult negative:

$$\min_n\left(D_{a,n}^2\right)$$

Schroff et al, "FaceNet: A Unified Embedding for Face Recognition and Clustering", 2015

Image Similarity: Loss functions: **Lifted Structured Loss**

Automatic hard negative mining within batch



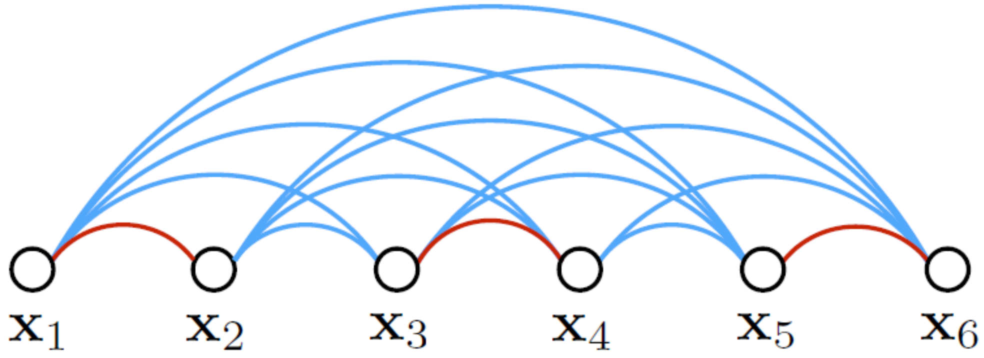
Training batch with six examples. Red edges and blue edges represent similar and dissimilar examples respectively.

Song et al, "Deep Metric Learning via Lifted Structured Feature Embedding", 2015

Image Similarity: Loss functions: **Lifted Structured Loss**

$$\tilde{J}_{i,j} = \log\left(\sum_{(i,k)\in N} \exp(m - D_{i,k}) + \sum_{(j,l)\in N} \exp(m - D_{j,l})\right) + D_{i,j}$$

$$L_{batch} = \frac{1}{2|P|} \sum_{(i,j)\in P} max(0,\ \tilde{J}_{i,j})^2$$



(c) Lifted structured embedding

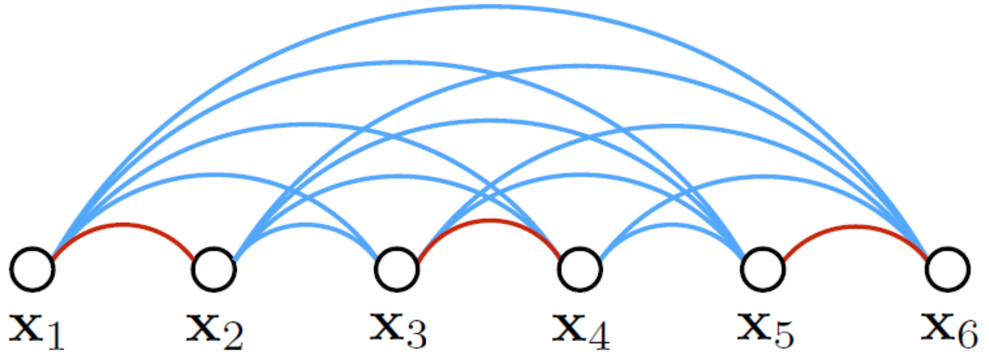
Song et al, "Deep Metric Learning via Lifted Structured Feature Embedding", 2015

Image Similarity: Loss functions: **Lifted Structured Loss**

$$\tilde{J}_{i,j} = \log\left(\sum_{(i,k)\in N} \exp(m - D_{i,k}) + \sum_{(j,l)\in N} \exp(m - D_{j,l})\right) + D_{i,j}$$

$$L_{batch} = \frac{1}{2|P|} \sum_{\boxed{(i,j)\in P}} max(0,\ \tilde{J}_{i,j})^2$$

all positive pairs
within batch



(c) Lifted structured embedding

Song et al, "Deep Metric Learning via Lifted Structured Feature Embedding", 2015

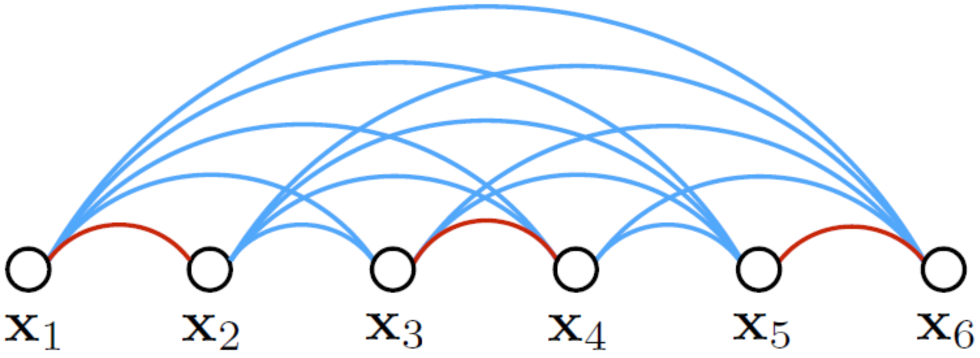
Image Similarity: Loss functions: **Lifted Structured Loss**

margin

$$\tilde{J}_{i,j} = \log\left(\sum_{(i,k)\in N} \exp(\boxed{m} - D_{i,k}) + \sum_{(j,l)\in N} \exp(m - D_{j,l})\right) + D_{i,j}$$

$$L_{batch} = \frac{1}{2|P|} \sum_{\boxed{(i,j)\in P}} max(0,\ \tilde{J}_{i,j})^2$$

all positive pairs
within batch



(c) Lifted structured embedding

Song et al, "Deep Metric Learning via Lifted Structured Feature Embedding", 2015

Image Similarity: Loss functions: **Lifted Structured Loss**

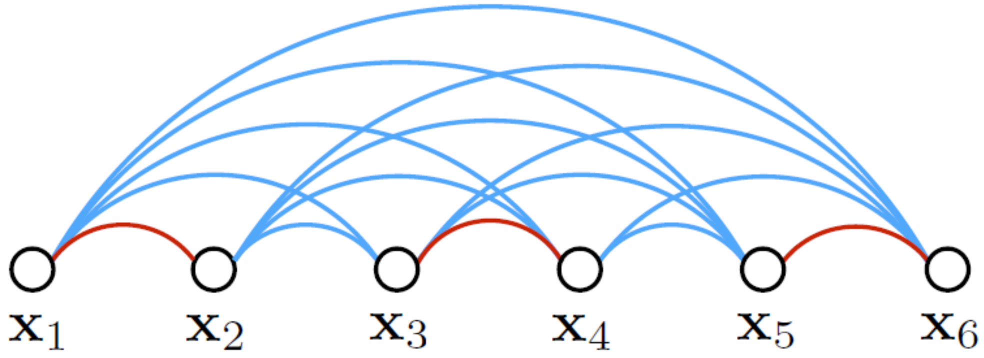
$$\tilde{J}_{i,j} = \log\left(\sum_{\boxed{(i,k)\in N}} \exp(m - D_{i,k}) + \sum_{\boxed{(j,l)\in N}} \exp(m - D_{j,l})\right) + D_{i,j}$$

all negative pairs for *i*

all negative pairs for *j*

$$L_{batch} = \frac{1}{2|P|} \sum_{\boxed{(i,j)\in P}} max(0,\ \tilde{J}_{i,j})^2$$

all positive pairs within batch



(c) Lifted structured embedding

Song et al, "Deep Metric Learning via Lifted Structured Feature Embedding", 2015

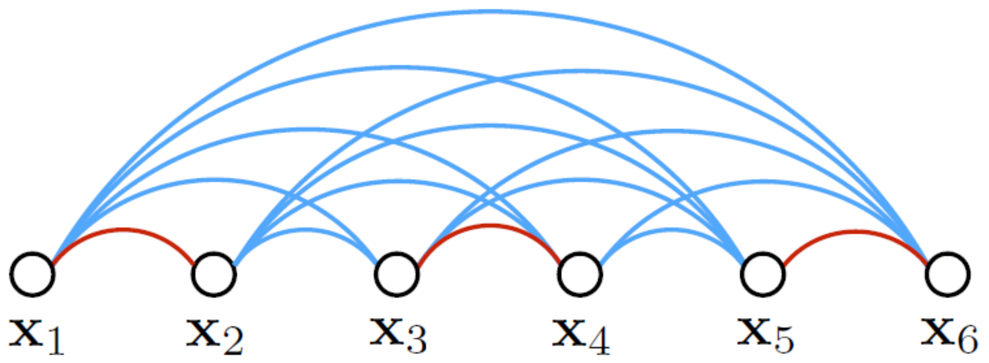
Image Similarity: Loss functions: **Lifted Structured Loss**

softmax ☺ gives hardest negative for *i* and *j*

$$\tilde{J}_{i,j} = \boxed{\log\left(\sum_{(i,k)\in N} \exp(m - D_{i,k}) + \sum_{(j,l)\in N} \exp(m - D_{j,l})\right)} + D_{i,j}$$

$$L_{batch} = \frac{1}{2|P|} \sum_{\boxed{(i,j)\in P}} max(0,\ \tilde{J}_{i,j})^2$$

all positive pairs
within batch



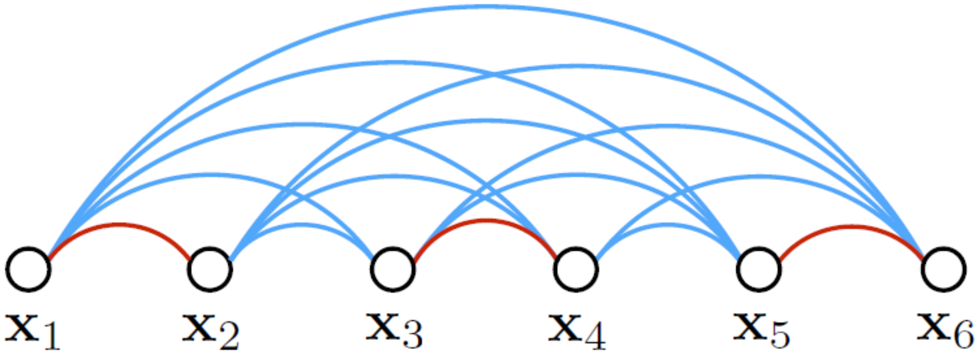
(c) Lifted structured embedding

Song et al, "Deep Metric Learning via Lifted Structured Feature Embedding", 2015

Image Similarity: Loss functions: **Lifted Structured Loss**

$$L_{batch} \sim \frac{1}{2|P|} \sum_{(i,j)\in P} max(0,\ D_{i,j} - D_{i|j,hardest\,negative} + \mathrm{m})^2$$

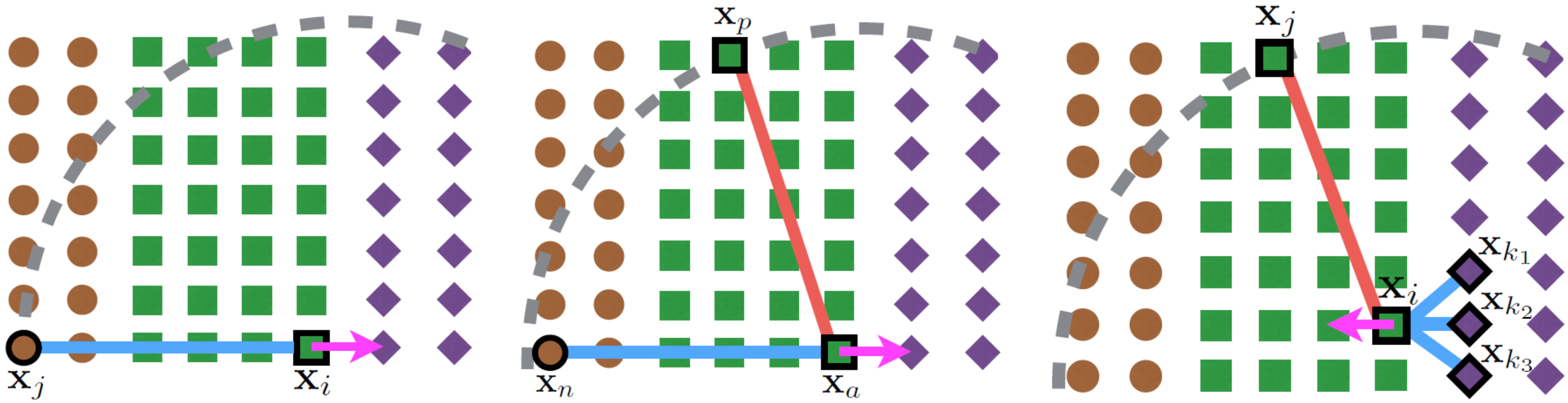
Similar to triplet loss but with automatic hard negative mining within batch!



(c) Lifted structured embedding

Song et al, "Deep Metric Learning via Lifted Structured Feature Embedding", 2015

Image Similarity: Loss functions: **Lifted Structured Loss**



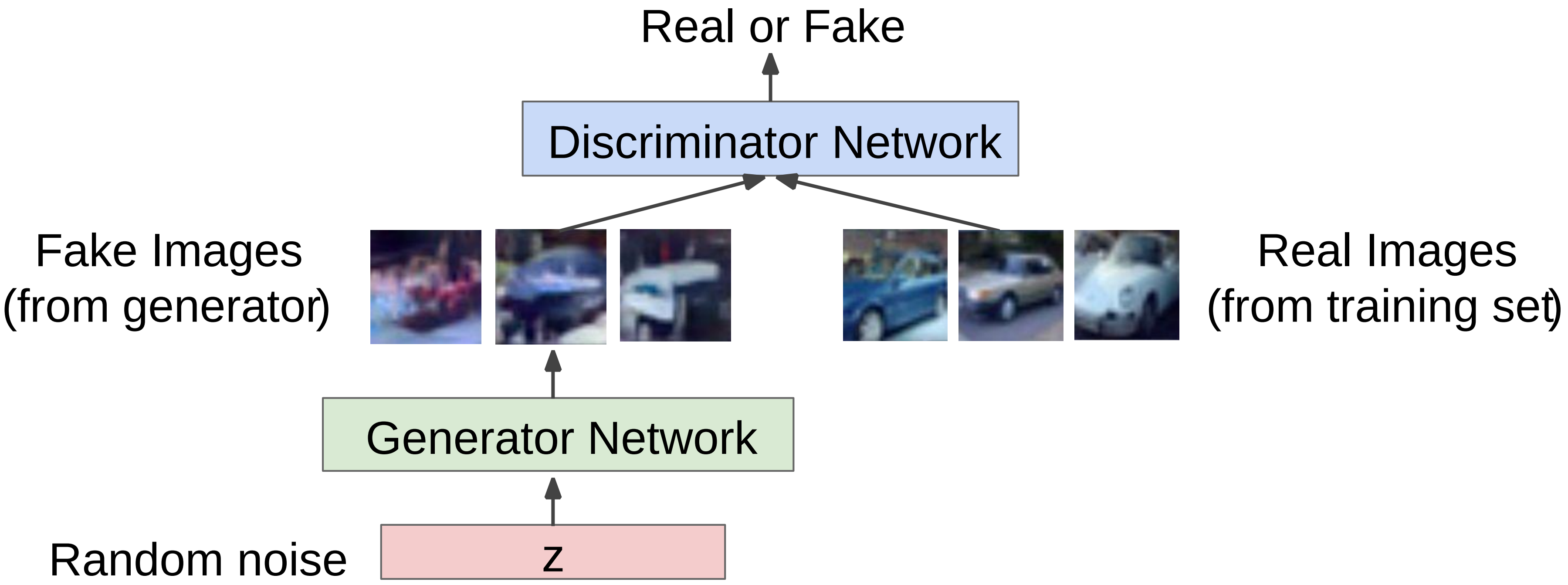
(a) Contrastive embedding (b) Triplet embedding (c) Lifted structured similarity

Illustration of failure modes of contrastive and triplet loss with randomly sampled training batch.

Song et al, "Deep Metric Learning via Lifted Structured Feature Embedding", 2015

GANs

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images



Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Training GANs: Two-player game

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images

Train jointly in **minimax game**

Minimax objective function:

Discriminator outputs likelihood in (0,1) of real image

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}) \right]$$

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Wasserstein GAN

Minimize Earth-Mover (EM) distance or Wasserstein distance between P_r **real data distribution** and P_g **generated data distribution**

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\big[\, \|x - y\| \,\big]$$

Arjovsky et al, “Wasserstein GAN”, 2017

Wasserstein GAN

Minimize Earth-Mover (EM) distance or Wasserstein distance between P_r **real data distribution** and P_g **generated data distribution**

$$W(\boxed{\mathbb{P}_r}, \boxed{\mathbb{P}_g}) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[\, \|x - y\| \,\right]$$

Real data distribution

Generated data distribution

Arjovsky et al, "Wasserstein GAN", 2017

Wasserstein GAN

Minimize Earth-Mover (EM) distance or Wasserstein distance between P_r **real data distribution** and P_g **generated data distribution**

$$W(\boxed{\mathbb{P}_r}, \boxed{\mathbb{P}_g}) = \inf_{\gamma \in \boxed{\Pi(\mathbb{P}_r, \mathbb{P}_g)}} \mathbb{E}_{(x,y)\sim\gamma}\left[\, \|x - y\| \,\right]$$

Real data distribution

Generated data distribution

Set of all joint distribution

Arjovsky et al, “Wasserstein GAN”, 2017

Wasserstein GAN

Minimize Earth-Mover (EM) distance or Wasserstein distance between P_r **real data distribution** and P_g **generated data distribution**

$$W(\boxed{\mathbb{P}_r}, \boxed{\mathbb{P}_g}) = \inf_{\gamma \in \boxed{\Pi(\mathbb{P}_r, \mathbb{P}_g)}} \mathbb{E}_{(x,y)\sim\gamma}\big[\, \|x - y\| \,\big]$$

Real data distribution

Generated data distribution

Set of all joint distribution

Intuitively, $\gamma(x, y)$ indicates how much "mass" must be transported from *x* to *y* in order to transform the distributions P_r into the distribution P_g.

The EM distance is the "cost" of the optimal transport plan.

Arjovsky et al, "Wasserstein GAN", 2017

Wasserstein GAN: CNN approximation

Minimize Earth-Mover (EM) distance or Wasserstein distance between P_r **real data distribution** and P_g **generated data distribution**

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\big[\, \|x - y\| \,\big]$$

$\Downarrow$ Kantorovich-Rubinstein theorem

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

supremum is over all the 1-Lipschitz functions

Arjovsky et al, “Wasserstein GAN”, 2017

Wasserstein GAN: CNN approximation

Minimize Earth-Mover (EM) distance or Wasserstein distance between
P_r **real data distribution** and P_g **generated data distribution**

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\big[\, \|x - y\| \,\big]$$

$$\Downarrow$$

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

$$\Downarrow$$

approximate *f(x)* by critic CNN $f_w(x)$

$$\max_{w\in\mathcal{W}} \mathbb{E}_{x\sim\mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z\sim p(z)}[f_w(g_\theta(z))]$$

Arjovsky et al,
“Wasserstein GAN”, 2017

Arjovsky et al, “Wasserstein GAN”, 2017

Wasserstein GAN: Training

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
    for $t = 0, ..., n_{\text{critic}}$ **do**
        Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.
        Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
        $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$
        $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
        $w \leftarrow \text{clip}(w, -c, c)$
    end for
    Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
    $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
    $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

Arjovsky et al, “Wasserstein GAN”, 2017

Wasserstein GAN: Training

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**

  for $t = 0, ..., n_{\text{critic}}$ **do**

    Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.

    Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.

    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$

    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$

    $w \leftarrow \text{clip}(w, -c, c)$

  end for

  Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.

  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$

  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$

end while

train critic

Arjovsky et al, “Wasserstein GAN”, 2017

Wasserstein GAN: Training

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**

    for $t = 0, ..., n_{\text{critic}}$ **do**

        Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.

        Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.

        $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$

        $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$

        $w \leftarrow \text{clip}(w, -c, c)$

    end for

    Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.

    $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$

    $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$

end while

train generator

Arjovsky et al, "Wasserstein GAN", 2017

Wasserstein GAN: Training

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**

  for $t = 0, ..., n_{\text{critic}}$ **do**

    Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.

    Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.

    $\boxed{g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]}$

    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$

    $w \leftarrow \text{clip}(w, -c, c)$

  end for

  Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.

  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$

  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$

end while

gradient of critic parameters using Wasserstein distance approximation with neural network

Arjovsky et al, “Wasserstein GAN”, 2017

Wasserstein GAN: Training

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$
    $\boxed{w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)}$ update of critic parameters
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

Arjovsky et al, “Wasserstein GAN”, 2017

Wasserstein GAN: Training

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $\boxed{w \leftarrow \text{clip}(w, -c, c)}$ clip parameters of critic
  end for
  Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

Arjovsky et al, “Wasserstein GAN”, 2017

Wasserstein GAN: Training

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
  $\boxed{g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))}$ gradient of generator parameters
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

Wasserstein GAN: Training

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**

  for $t = 0, ..., n_{\text{critic}}$ **do**

    Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.

    Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.

    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$

    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$

    $w \leftarrow \text{clip}(w, -c, c)$

  end for

  Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.

  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$

  $\boxed{\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)}$

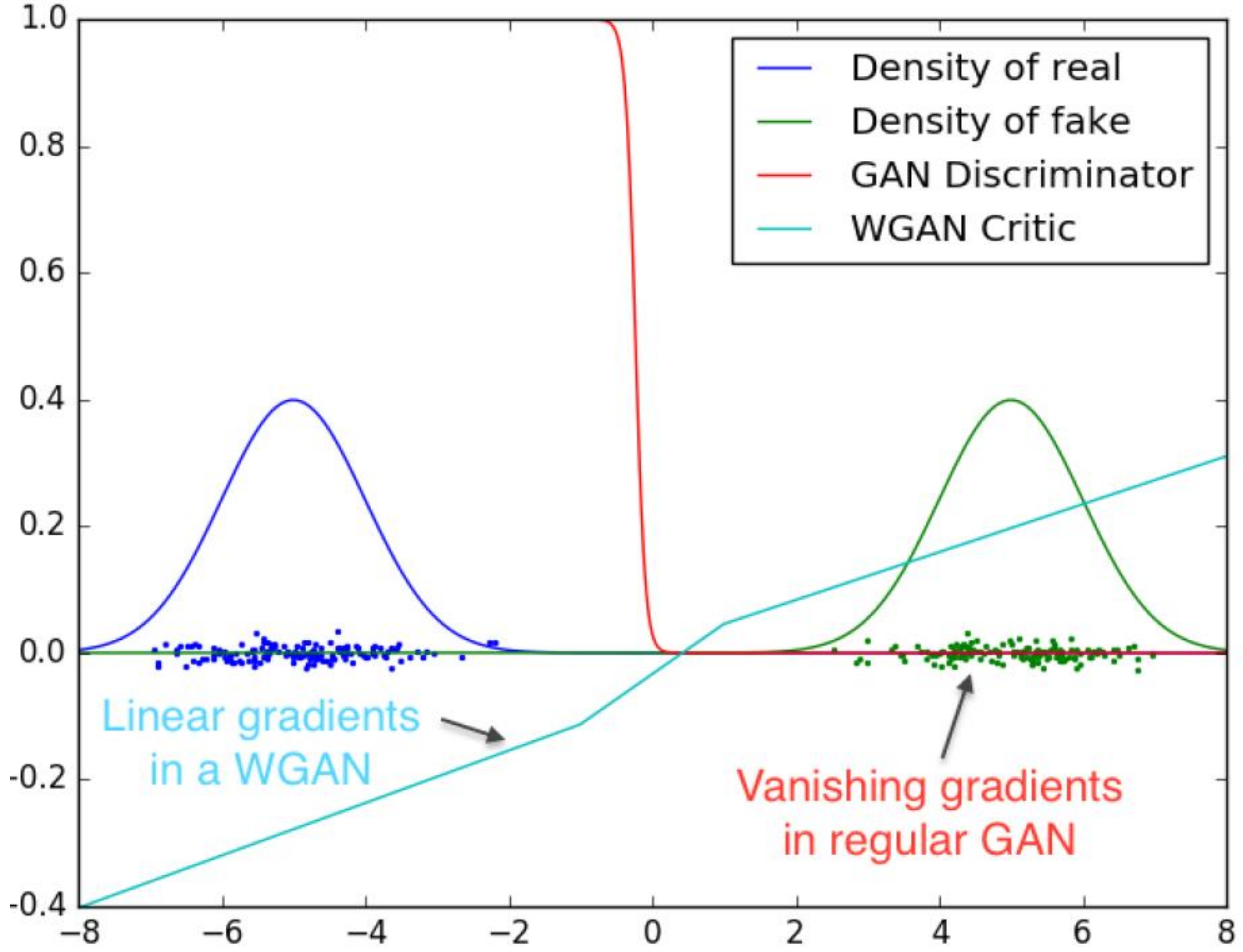
end while

update of generator parameters

Wasserstein GAN: No Vanishing Gradients

Optimal discriminator and critic when learning to differentiate two Gaussians. Traditional GAN discriminator saturates and results in vanishing gradients.

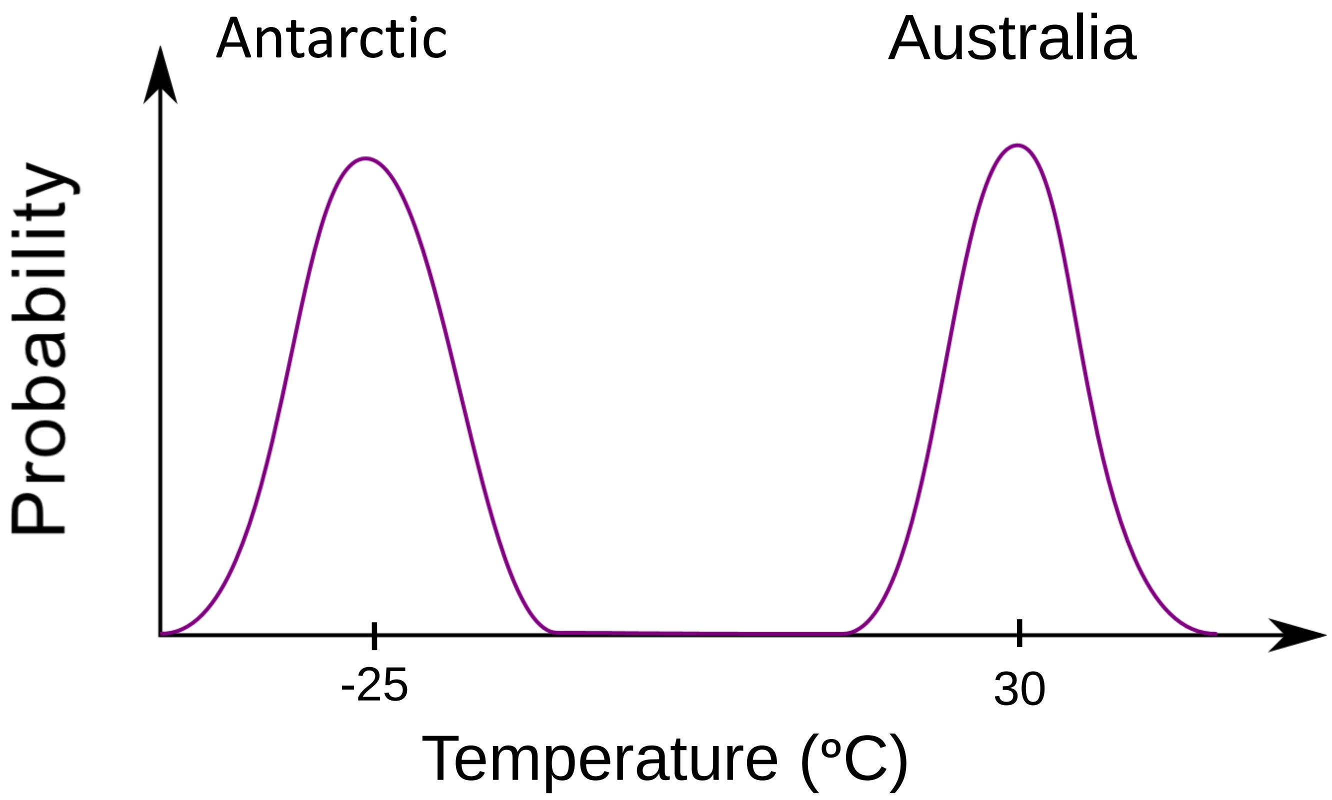
Wasserstein GAN critic provides very clean gradients on all parts of the space.



Arjovsky et al, “Wasserstein GAN”, 2017

Mode collapse in GANs

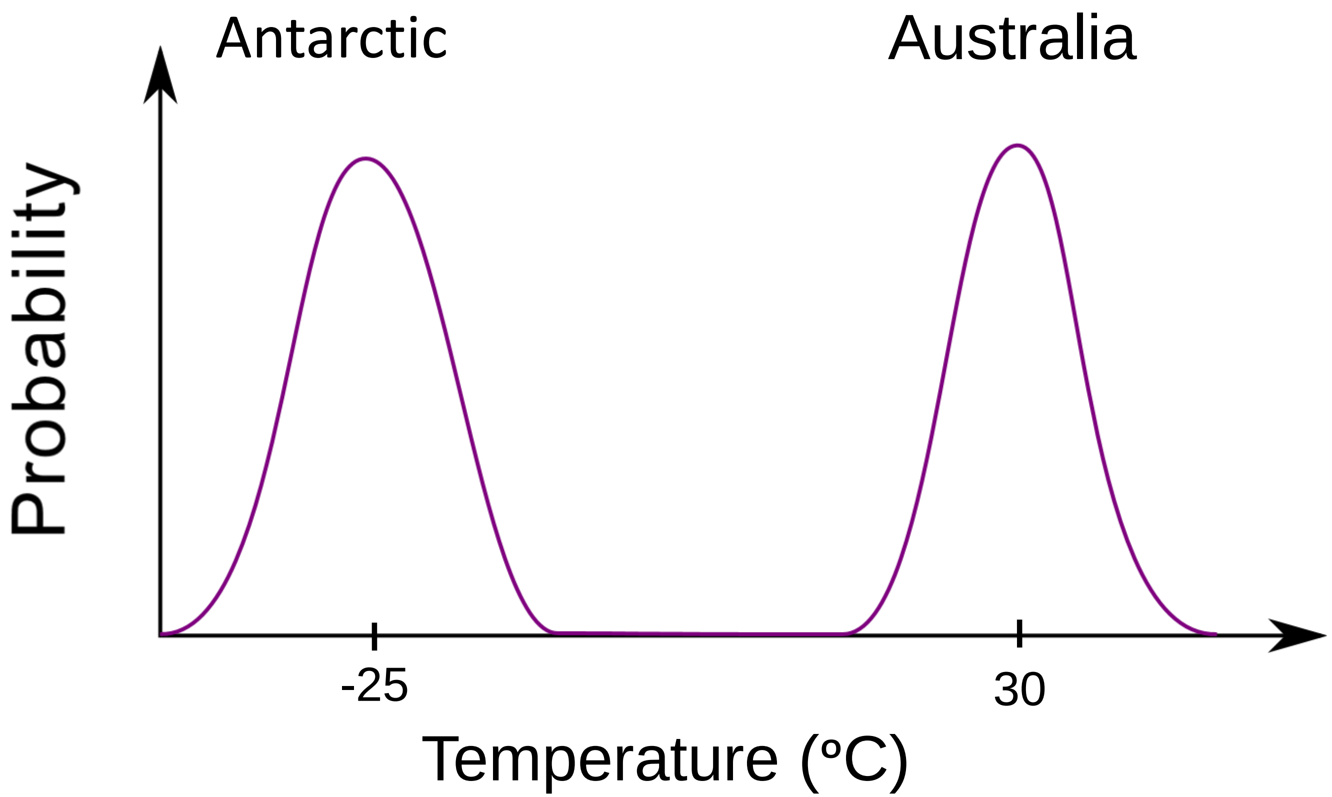
1. Generator learns that it can fool the discriminator by producing values close to Antarctic temperatures
2. The discriminator counters by learning that all Australian temperatures are real, and guesses whether Antarctic temperatures are real or fake
3. The generator exploits the discriminator by switching modes to produce values close to Australian temperatures instead, abandoning the Antarctic mode
4. The discriminator now assumes that all Australian temperatures are fake and Antarctic temperatures are real
5. Return to step 1



http://aiden.nibali.org/blog/2017-01-18-mode-collapse-gans/

Mode collapse in GANs

1. Generator learns that it can fool the discriminator by producing values close to Antarctic temperatures
2. The discriminator counters by learning that all Australian temperatures are real, and guesses whether Antarctic temperatures are real or fake
3. The generator exploits the discriminator by switching modes to produce values close to Australian temperatures instead, abandoning the Antarctic mode
4. The discriminator now assumes that all Australian temperatures are fake and Antarctic temperatures are real
5. Return to step 1

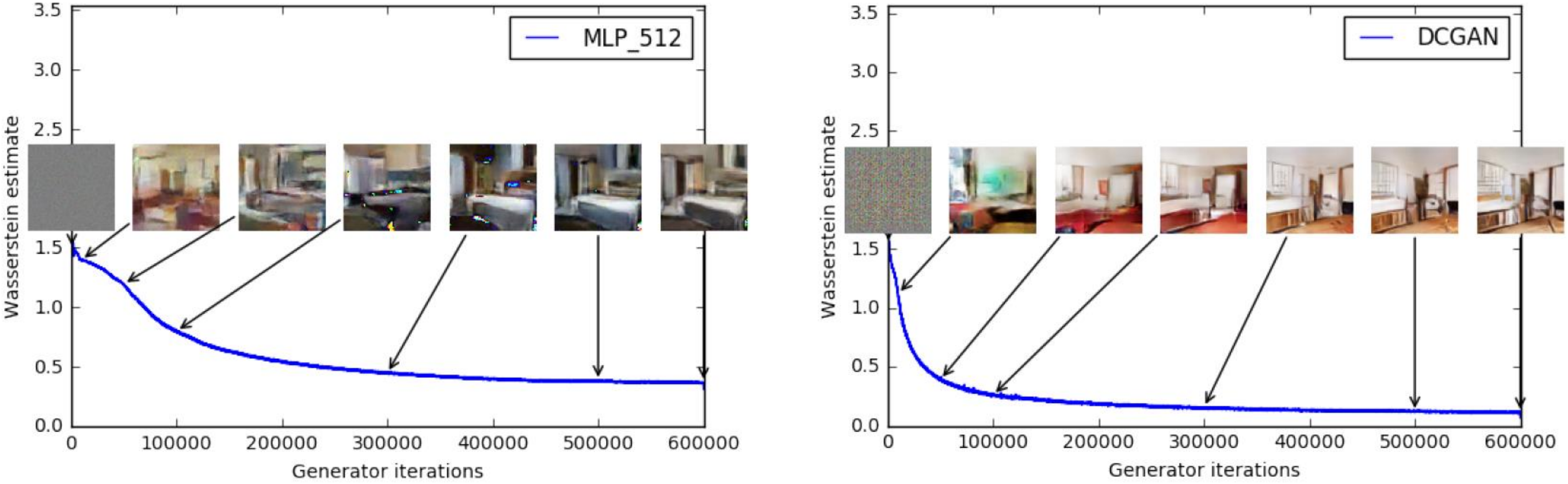


Wasserstein GANs do not have mode collapse

http://aiden.nibali.org/blog/2017-01-18-mode-collapse-gans/

Wasserstein GAN

Nice property lower loss better sample quality



Arjovsky et al, “Wasserstein GAN”, 2017

ICCV 2017: Venice, Italy, Oct 22-29

Growth of paper submissions
~30%

Growth of participants
~200%

ICCV 2017: We are hiring

We are hiring ☺