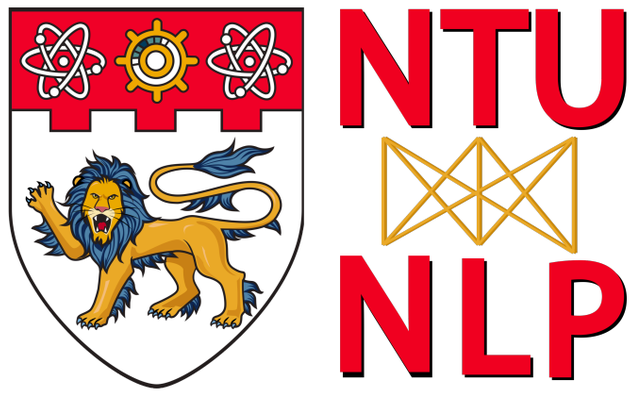
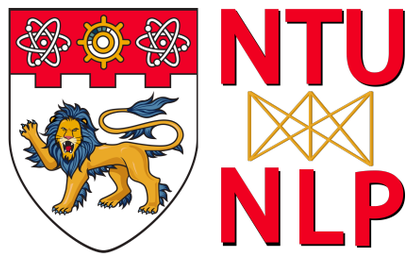
Deep Learning for Natural Language Processing

Shafiq Joty



Lecture 12: Adversarial NLP

Where we are



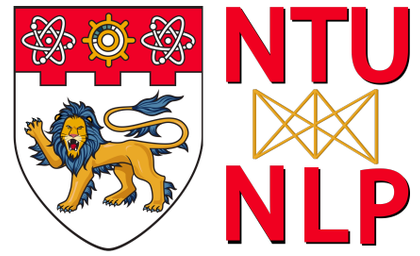
Models/Algorithms

- Linear models
- Feed-forward Neural Nets (FNN)
- Window-based methods
- Convolutional Nets
- Recurrent Neural Nets
- Recursive Neural Nets

NLP tasks/applications

- Word meaning
- Language modelling
- Sequence tagging
- Sequence encoding
- Parsing
- Hierarchical encoding

Where we are



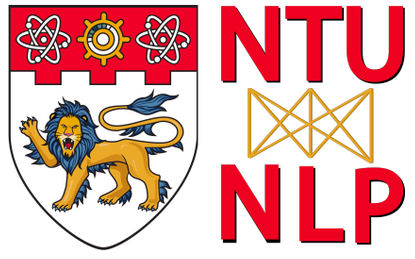
Models/Algorithms

- Seq2Seq
  - + Attention
  - + Subword
- Seq2Seq Varints
- Transformer Seq2Seq

NLP tasks/applications

- Machine Translation
- Summarization
- Parsing
- Dialogue generation

Where we are



Models/Algorithms

NLP tasks/applications

- Self-supervised pretraining
- Multilingual NLP
- Robust & adversarial NLP
- Ethics, Bias in NLP

Today's Plan: Adversarial NLP

NTU NLP

Adversarial Nets

- Generative adversarial nets (GANs)
- Domain adversarial nets (DANs)

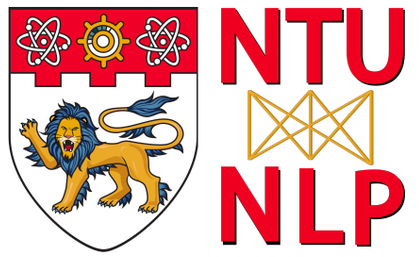
Adversarial Examples & Training

- Adversarial attacks
- Defense

NLP tasks/applications

- Machine Translation
- Domain Transfer
- Robust NLP

Adversarial Nets



Adversarial nets have been used in **two ways**:

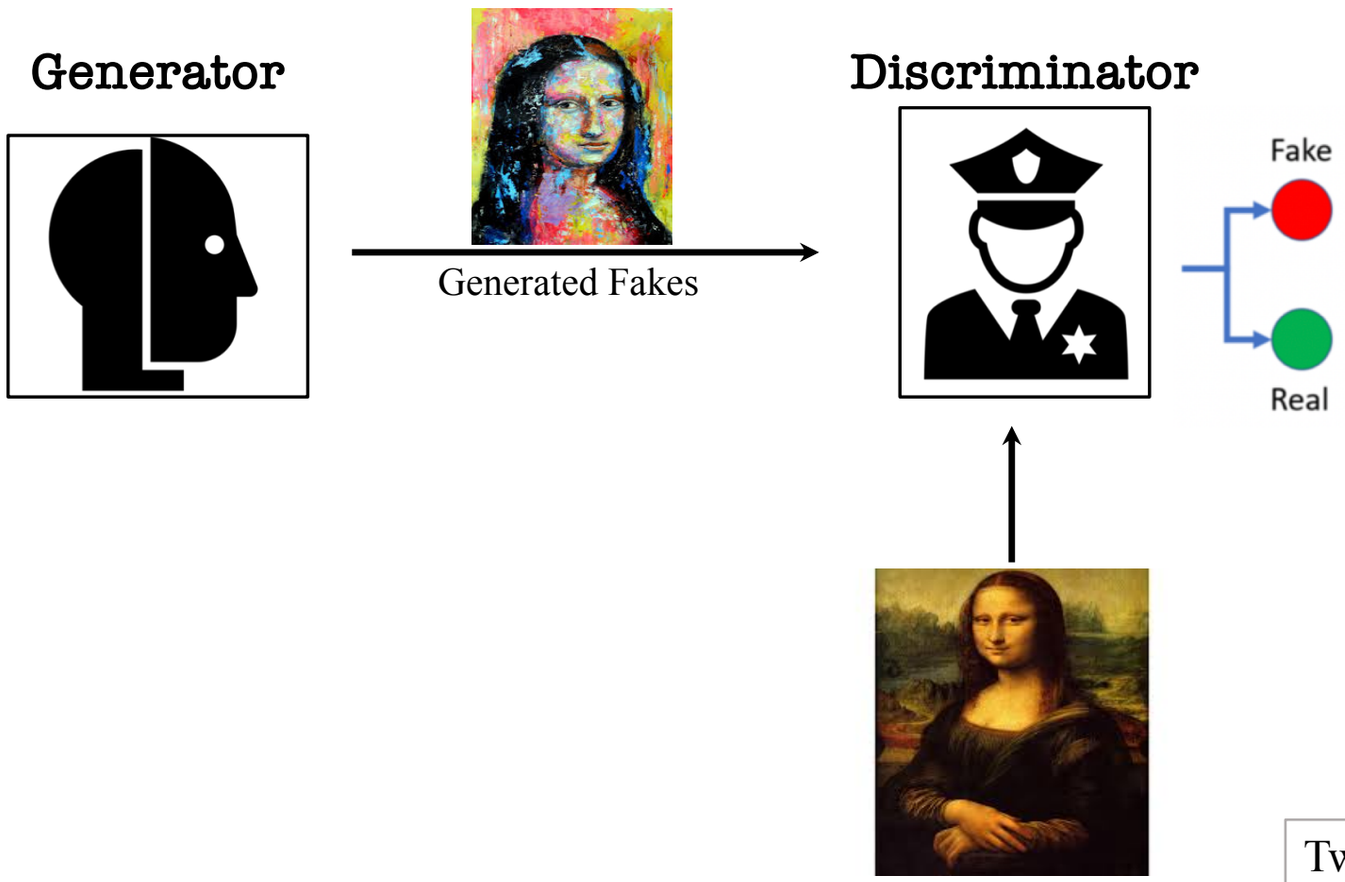
1. **Generative modeling**: GANs [Goodfellow et al., 2014].
   - Generate real-like image samples from random vectors
   - One neural net is pit against another:
     (*i*) Generator, (*ii*) Discriminator
   - Other Examples: image-to-image translation, Unsup. NMT

2. **Transfer learning/domain adaptation**:
   DANNs [Ganin et al., 2016]
   - Map samples from two domains into a common feature space
   - Generally a **three-player** game:
     (*i*) Encoder, (*ii*) Classifier (*iii*) Discriminator
   - Examples: MNIST $\Rightarrow$ USPS, X-lingual NER

Generative Adversarial Nets

NTU NLP

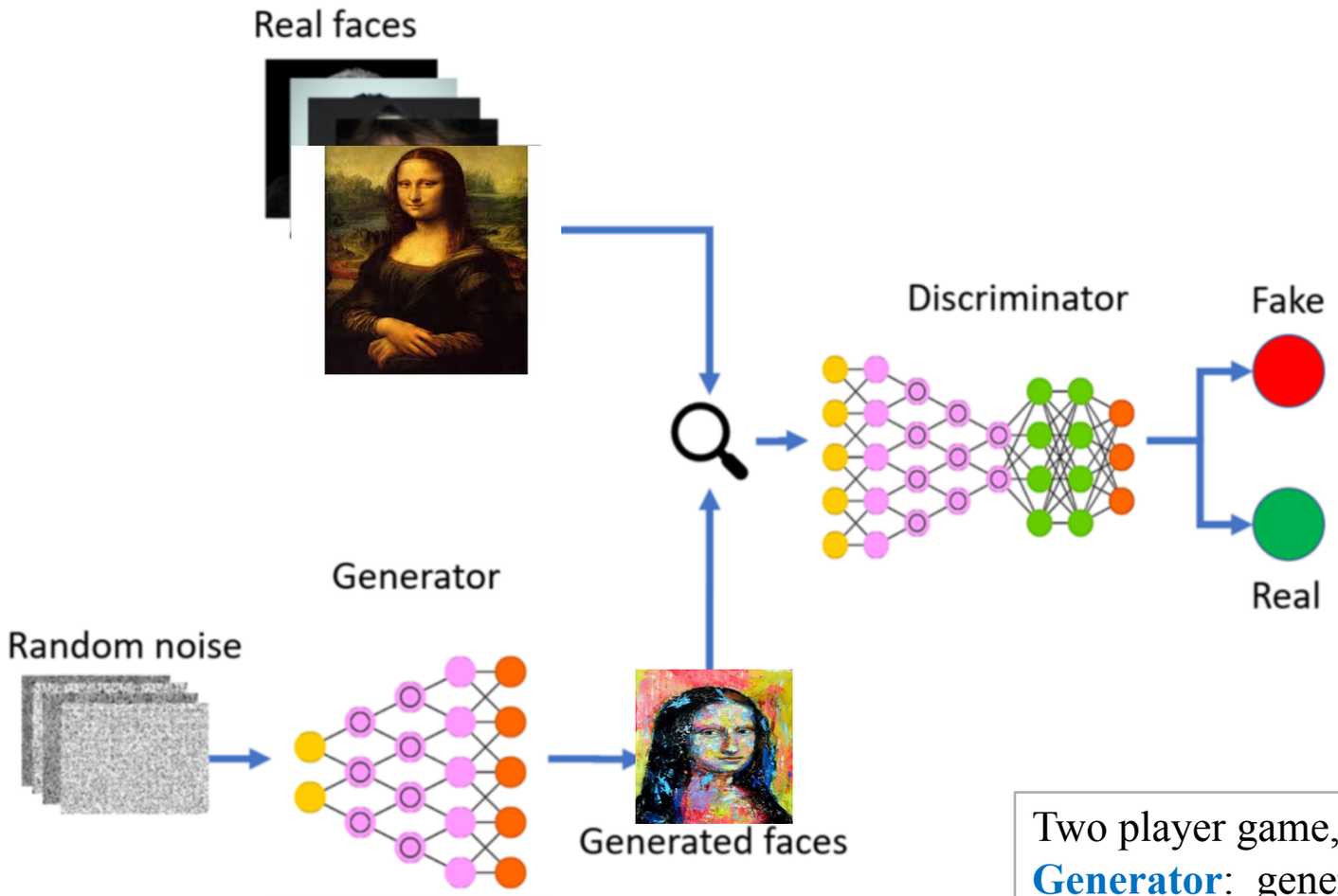
[Goodfellow et al., 2014]



Two player game, where
Generator: generates fake examples
Discriminator: differentiate between real and fake

Generative Adversarial Nets

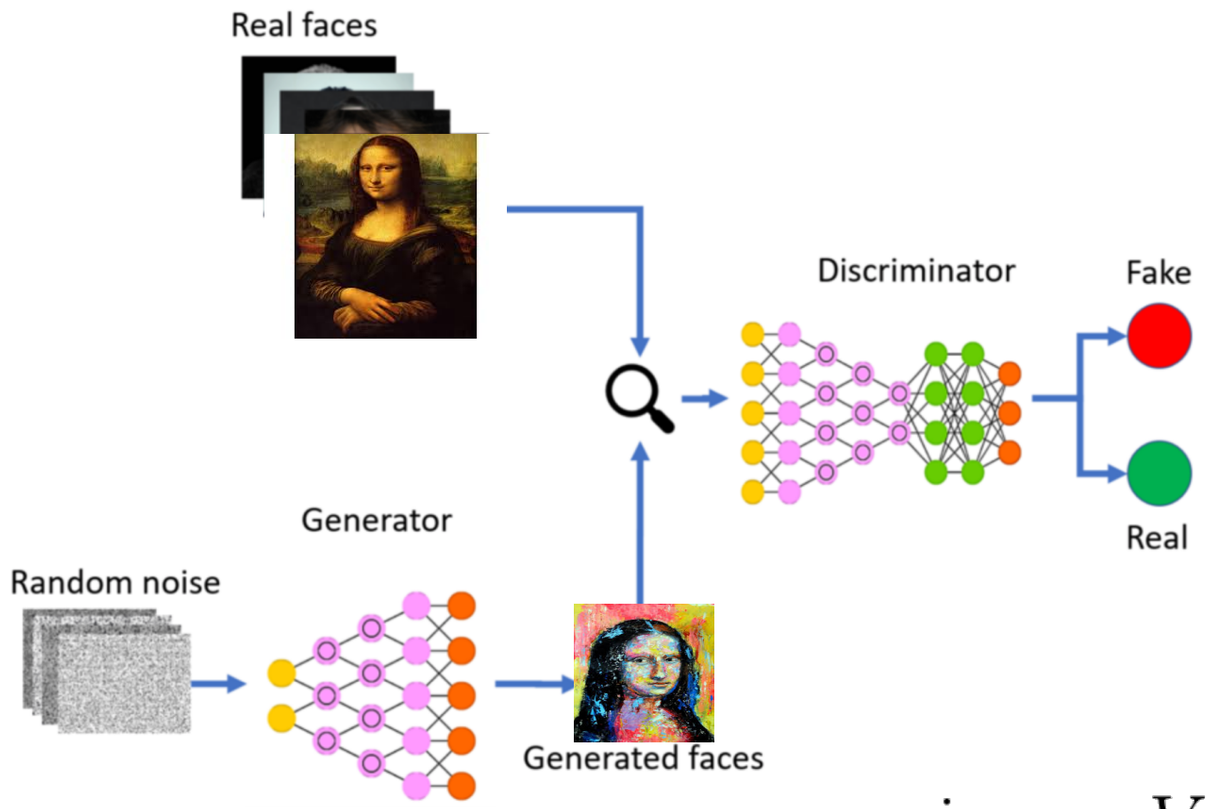
[Goodfellow et al., 2014]



Two player game, where
Generator: generates fake examples
Discriminator: differentiate between real and fake

Generative Adversarial Nets

[Goodfellow et al., 2014]



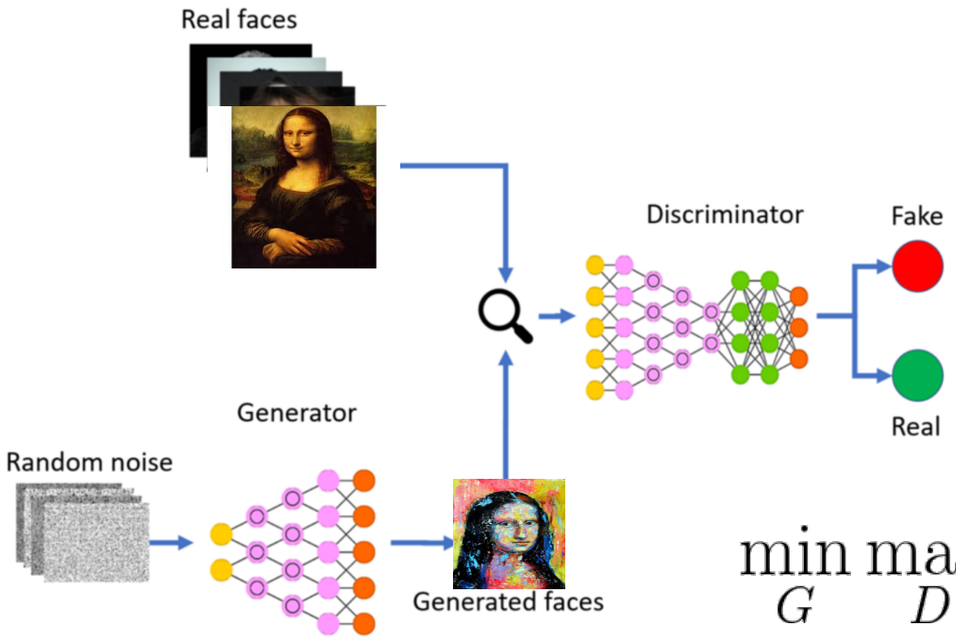
$$\min_G \max_D V(D,G)$$

$$= \mathbb{E}_{q(\mathbf{x})}[\log(D(\mathbf{x}))] + \mathbb{E}_{p(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$$

$D(x)$: the probability that x came from the data rather than generator

Generative Adversarial Nets

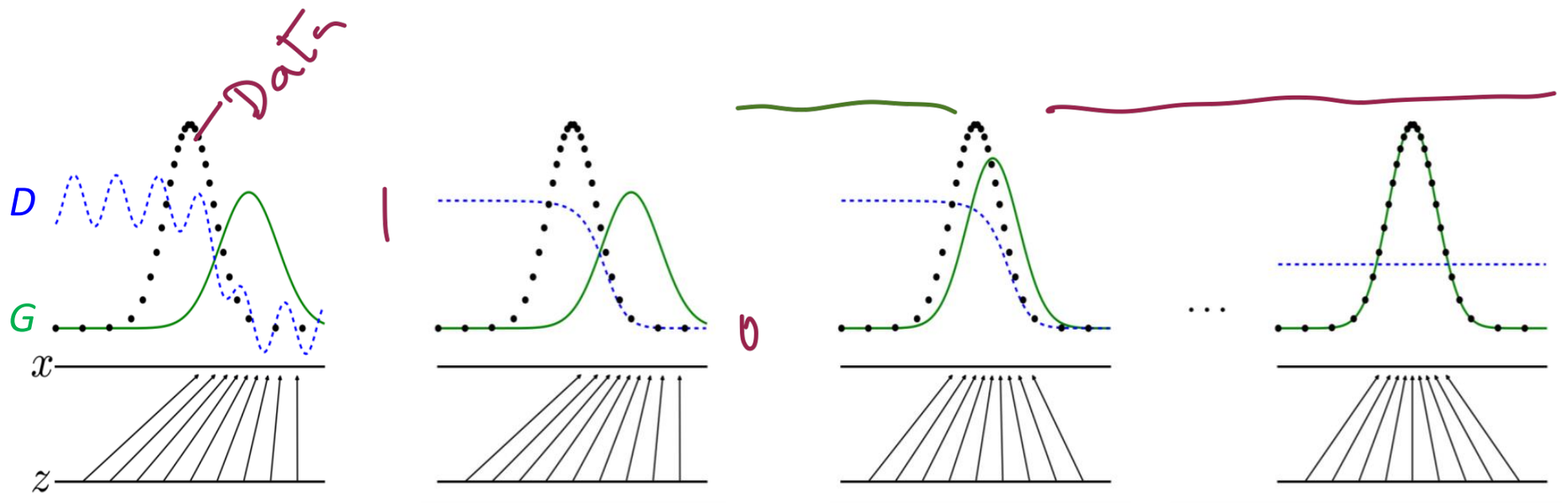
[Goodfellow et al., 2014]



$$\min_G \max_D V(D,G)$$

$D(x)$: the probability that x came from the data rather than generator

$$= \mathbb{E}_{q(\mathbf{x})}[\log(D(\mathbf{x}))] + \mathbb{E}_{p(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$$



Generative Adversarial Nets

[Goodfellow et al., 2014]

NTU NLP

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

Discriminator

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

Generator

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Generative Adversarial Nets

[Goodfellow et al., 2014]

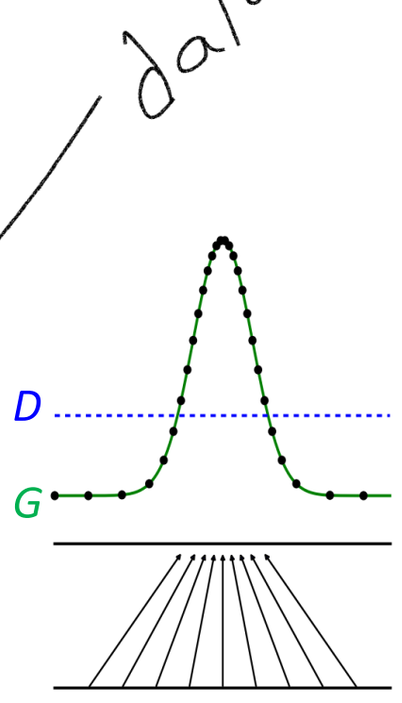
GAN Equilibrium

- Global optimality
  - Discriminator

$$D^*(\mathbf{x}) = \frac{q(\mathbf{x})}{q(\mathbf{x}) + p(\mathbf{x})}$$

target dist

data dist

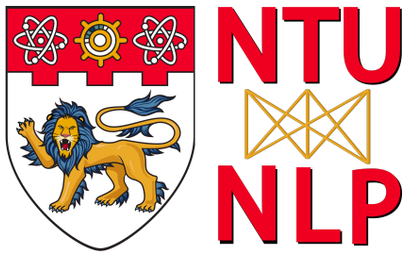


  - Generator

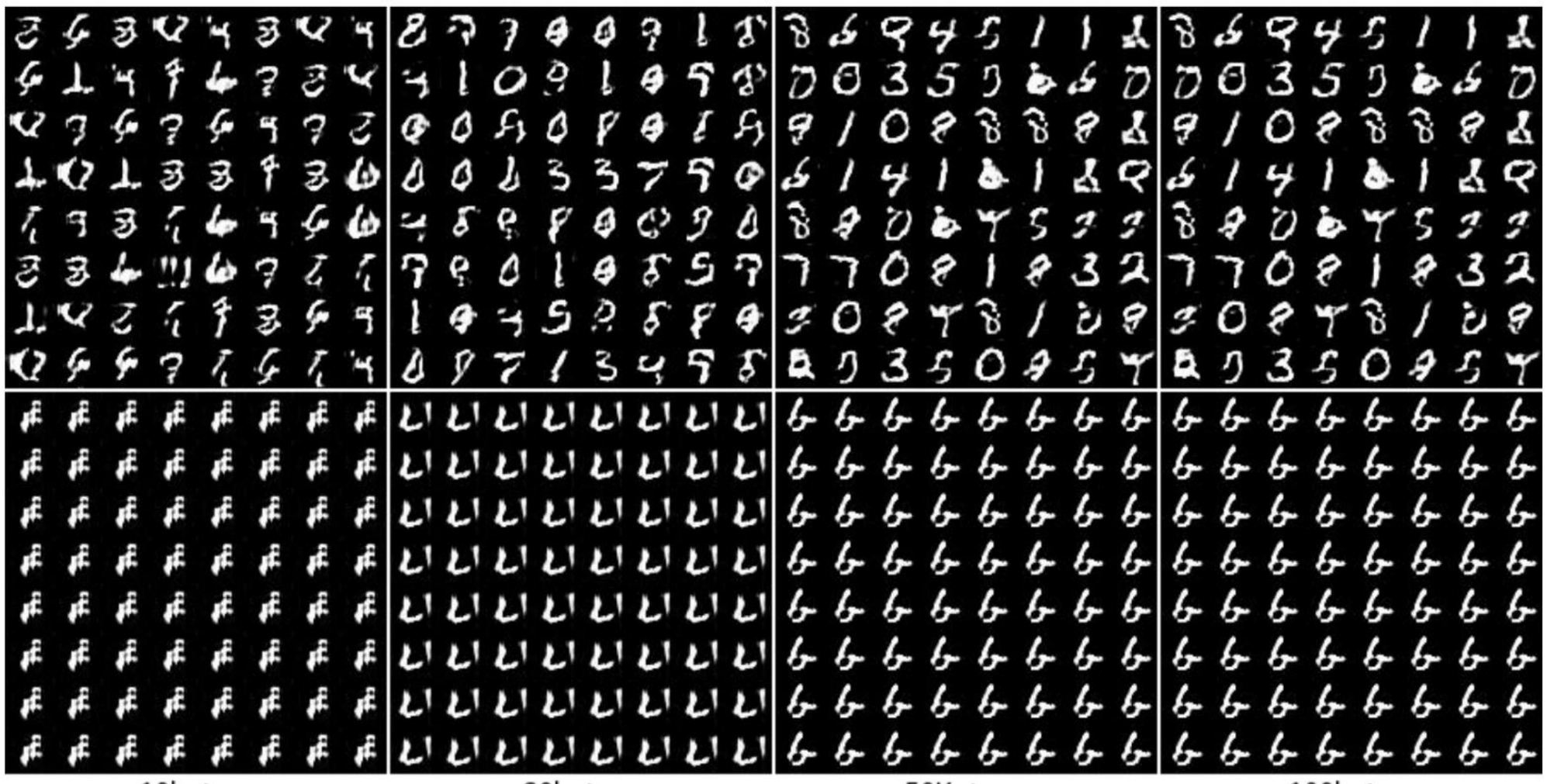
$$G^*(\mathbf{z}) \text{ s.t. } p(\mathbf{z}) = q(\mathbf{x})$$

model dist

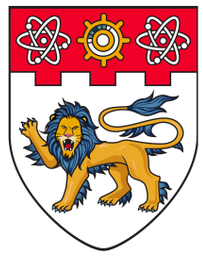
Major Issues of GAN Training



- Mode Collapse (unable to produce diverse samples)



Major Issues of GAN Training



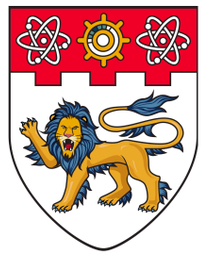
NTU NLP

- Unstable Adversarial Training
  - We are dealing with two networks / learners / agents
  - Should we update them at the same rate?
- The discriminator might overpower the generator.

Particularly for NLP

- GANs were originally designed for images, which is continuous — $D(G(\mathbf{z}))$
- But text is discrete — can't back-propagate

Major Issues of GAN Training



NTU NLP

Particularly for NLP

- GANs were originally designed for images, which is continuous — $D(G(\mathbf{z}))$
- But text is discrete — can't back-propagate

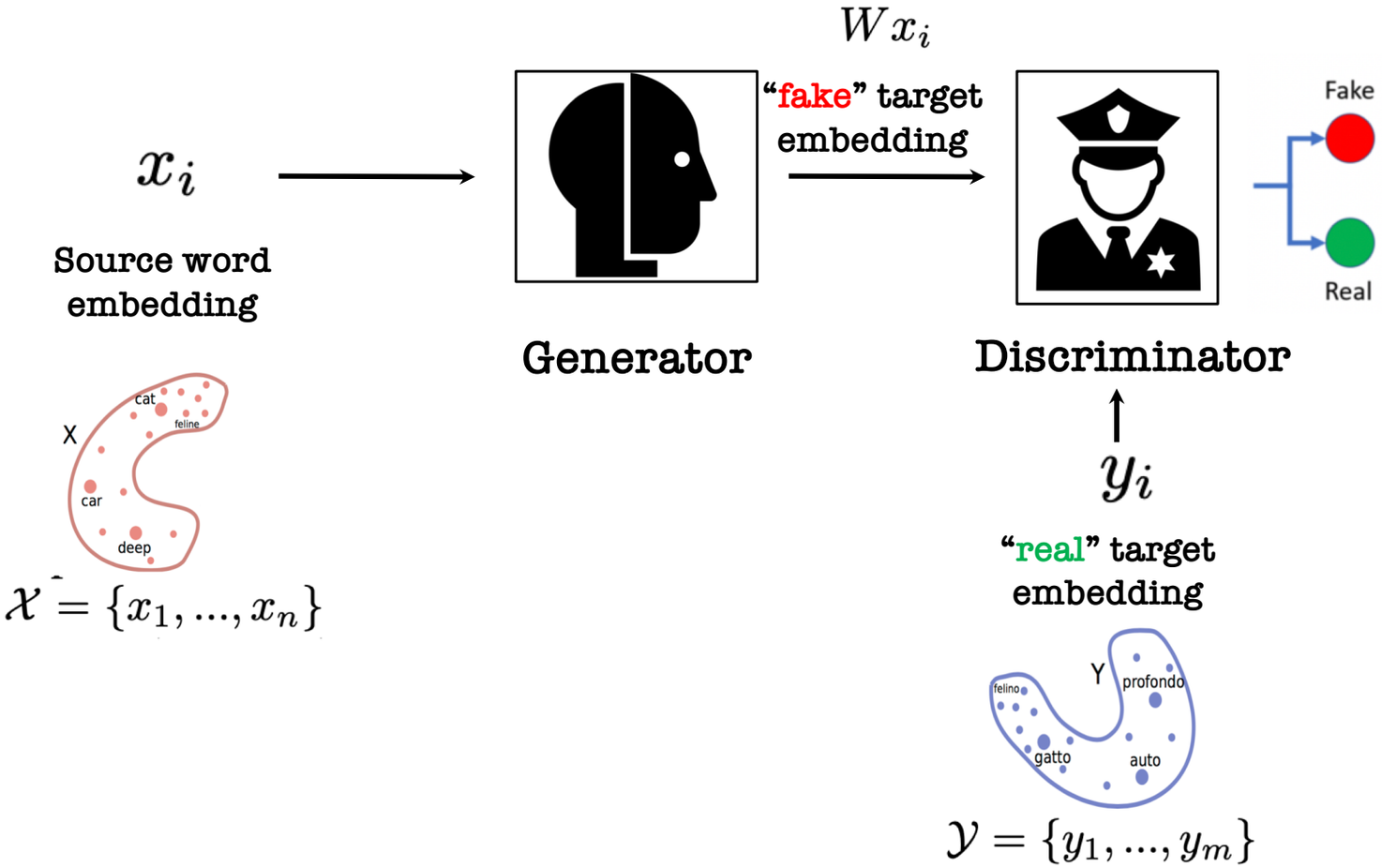
Some workaround

- Use it in **embedding** space
- Use Reinforcement learning with **policy gradient**

Applications of GANs in NLP

NTU NLP

Unsupervised Word Translation



- Generator needs to match distribution of target language in order to fool discriminator consistently.
- *Hypothesis*: Best way to do this is to align words with their translations.

Generator: projects source word embedding x_i into the target language using linear mapper W ⇒ Wx_i

Discriminator: differentiate between "fake" projected embeddings and "true" target language embeddings y_i

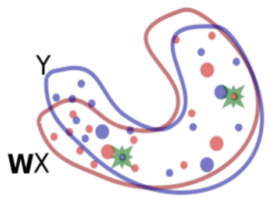
Conneau et al. (2018) Popularly known as **MUSE**

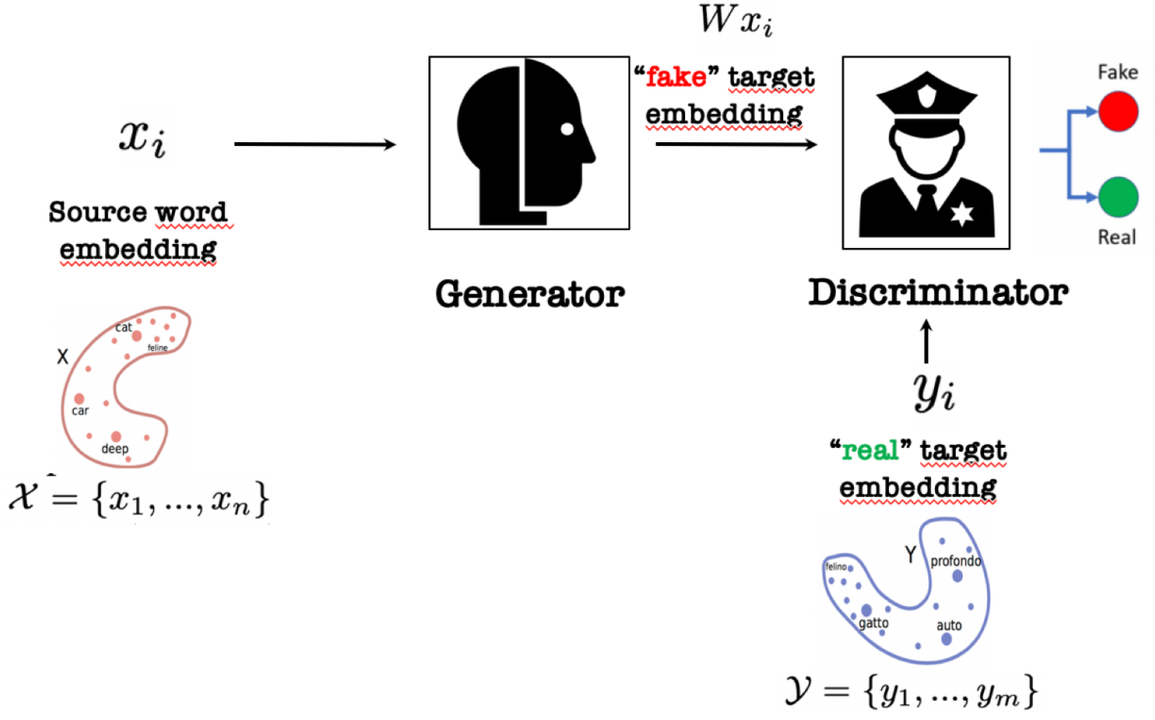
Source: [NAACL Tutorial]

Applications of GANs in NLP

NTU NLP

Unsupervised Word Translation





Objective

$\theta_D \Rightarrow$ Discriminator parameters

$P_{\theta_D}(\text{source} = 1|z) \Rightarrow$ Probability that z is the mapping of a source embedding

Discriminator Loss

An embedding in X — An embedding in Y

$$\mathcal{L}_D(\theta_D|W) = -\frac{1}{n}\sum_{i=1}^{n} \log P_{\theta_D}(\text{source} = 1|Wx_i) - \frac{1}{m}\sum_{i=1}^{m} \log P_{\theta_D}(\text{source} = 0|y_i)$$

Maximize probability of predicting correct source

Generator/Mapper Loss

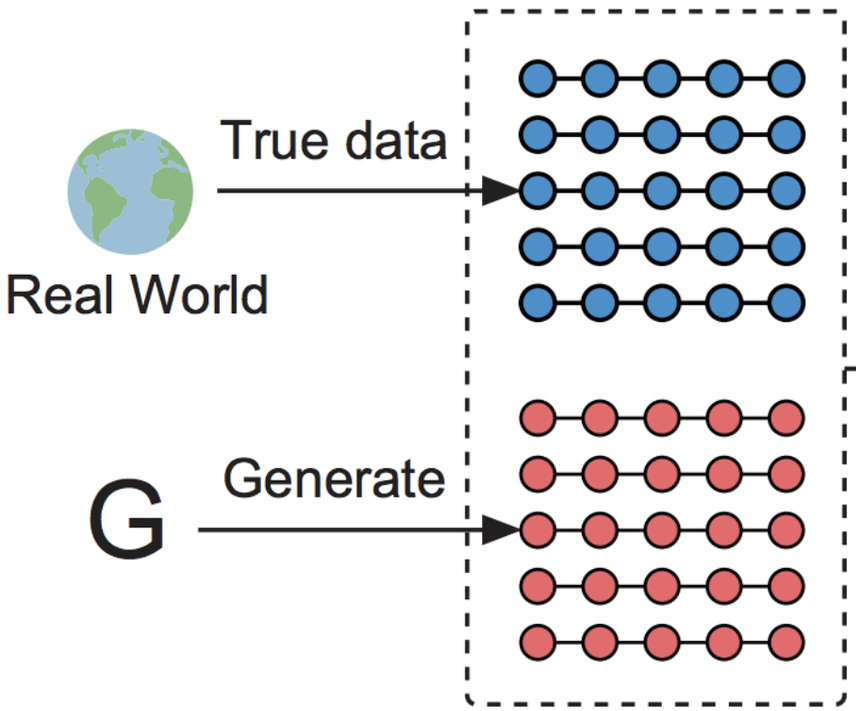
$$\mathcal{L}_W(W|\theta_D) = -\frac{1}{n}\sum_{i=1}^{n} \log P_{\theta_D}(\text{source} = 0|Wx_i) - \frac{1}{m}\sum_{i=1}^{m} \log P_{\theta_D}(\text{source} = 1|y_i)$$

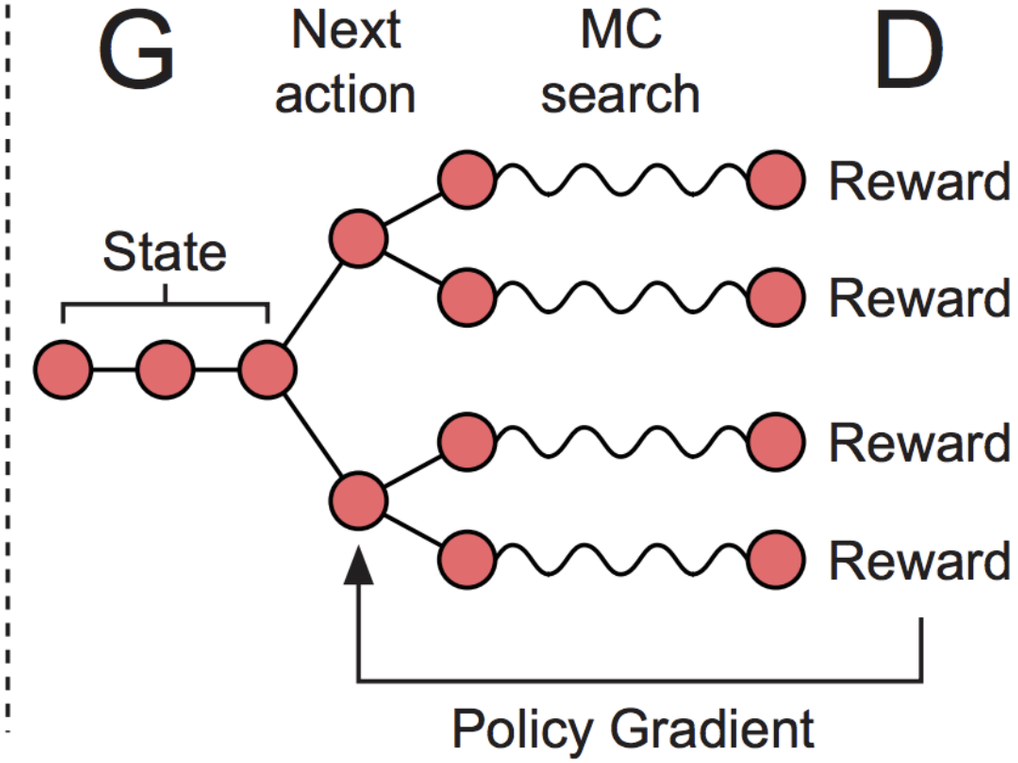
Maximize probability of fooling discriminator

Conneau et al. (2018) Popularly known as **MUSE**

Source: [NAACL Tutorial]

SeqGAN: policy gradient for generating sequences





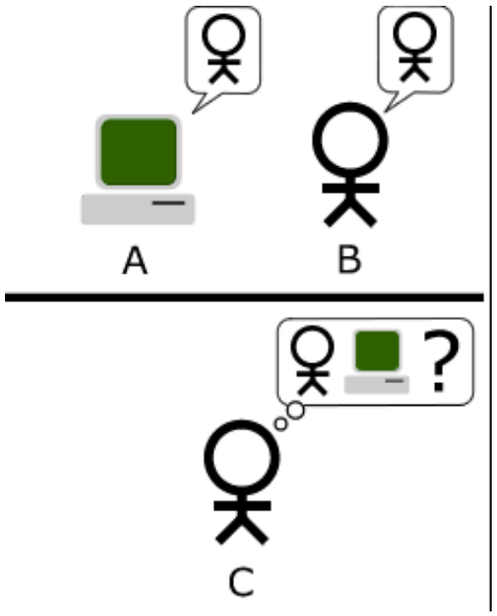
The discriminator is often learning a metric.

(Yu et al., 2017)

GAN for Dialogue

NTU NLP

Turing Test



Reward for Dialogue

I'm 25.

How old are you ?

I don't know what you are talking about

A human evaluator/ judge

Reward for Dialogue

NTU NLP

P= 90% human generated

I'm 25.

How old are you ?

I don't know what you are talking about

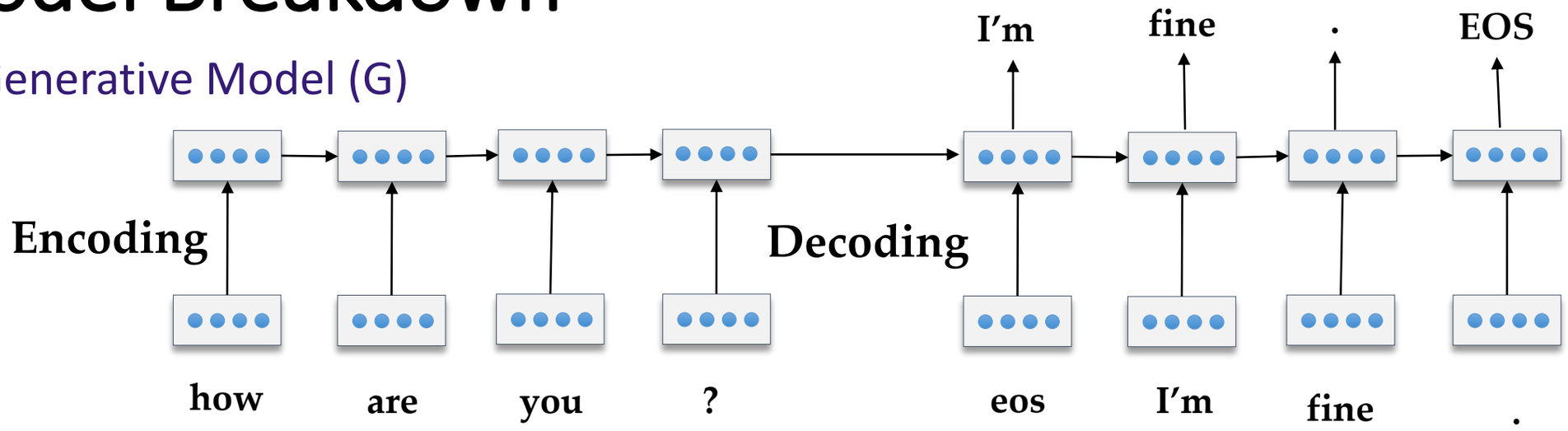
P= 10% human generated

Reward for Dialogue

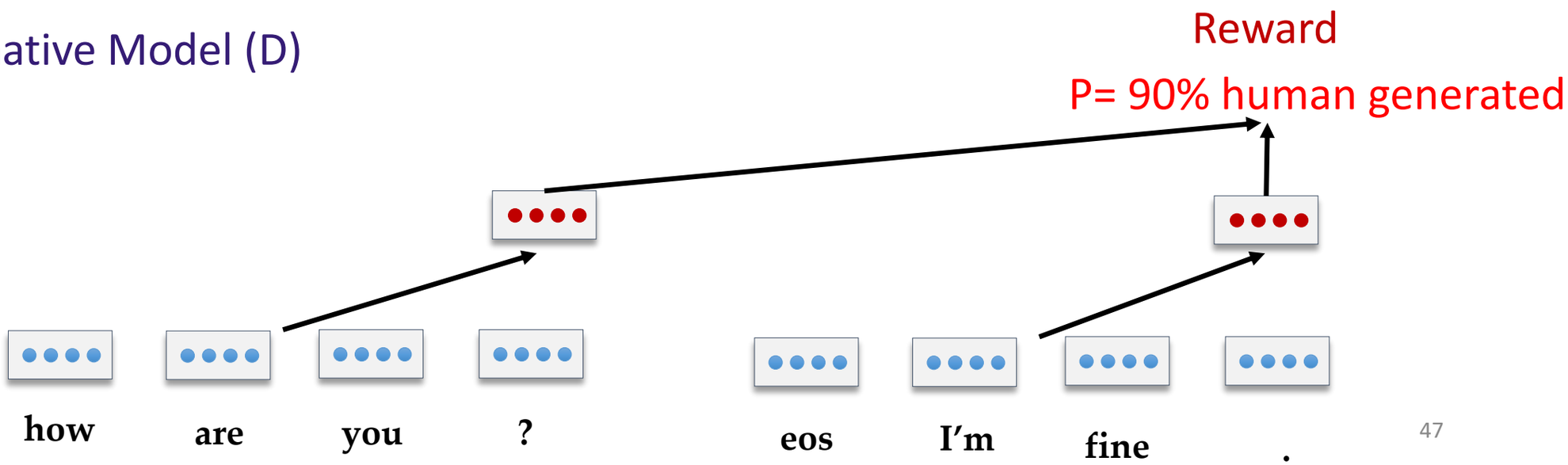
NTU NLP

Model Breakdown

Generative Model (G)



Discriminative Model (D)

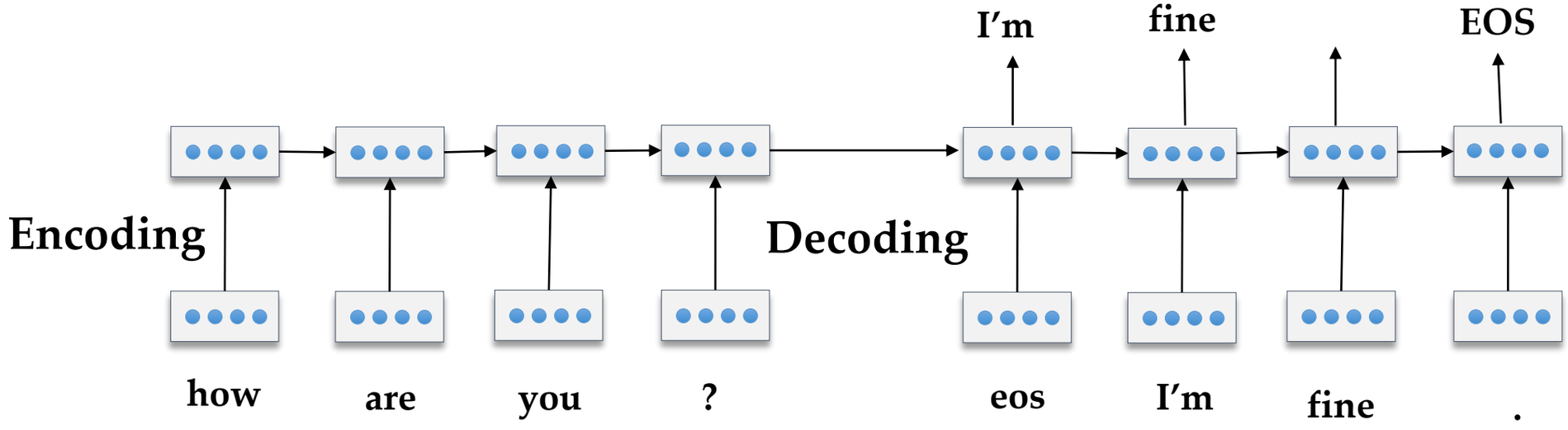


Reward for Dialogue

NTU NLP

Policy Gradient

Generative Model (G)



REINFORCE Algorithm (William,1992)

$$J = E[R(y)]$$

Reward for Dialogue

NTU NLP

For number of training iterations **do**

. **For** i=1,D-steps **do**
. Sample (X,Y) from real data
. Sample $\hat{Y} \sim G(\cdot|X)$
. Update D using (X, Y) as positive examples and $(X, \hat{Y})$ as negative examples.
. **End**

For i=1,G-steps **do**
Sample (X,Y) from real data
Sample $\hat{Y} \sim G(\cdot|X)$
Compute Reward r for $(X, \hat{Y})$ using D.
Update G on $(X, \hat{Y})$ using reward r
Teacher-Forcing: Update G on (X, Y)
End

End

Update the Discriminator

Update the Generator

The discriminator forces the generator to produce correct responses

Reward for Dialogue

Human Evaluator

vs a vanilla generation model

Adversarial Win	Adversarial Lose	Tie
62%	18%	20%

Adversarial Nets

Adversarial nets have been used in **two ways**:

1. **Generative modeling**: GANs [Goodfellow et al., 2014].
   - Generate real-like image samples from random vectors
   - One neural net is pit against another:
     (*i*) Generator, (*ii*) Discriminator
   - Other Examples: image-to-image translation, Unsup. NMT

2. **Transfer learning/domain adaptation**:
   DANNs [Ganin et al., 2016]
   - Map samples from two domains into a common feature space
   - Generally a **three-player** game:
     (*i*) Encoder, (*ii*) Classifier (*iii*) Discriminator
   - Examples: MNIST $\Rightarrow$ USPS, X-lingual NER

Domain Adversarial Nets

NTU NLP

Training Data:

$$\begin{aligned} \mathcal{D}_s &= \{(\mathbf{x}_i, y_i)\}_{i=1}^{N} \\ \mathcal{D}_t &= \{\mathbf{x}_i\}_{i=1}^{M} \end{aligned}$$

- $\mathcal{D}_s$ denotes source (labeled) dataset.
- $\mathcal{D}_t$ denotes target (unlabeled) dataset.
- Distribution (domain) shift from $\mathcal{D}_s$ to $\mathcal{D}_t$

Examples

NLP Sentiment analysis (Movie $\Rightarrow$ Book), NER (Eng $\Rightarrow$ Ger)

CV Image classification: MNIST $\Rightarrow$ USPS

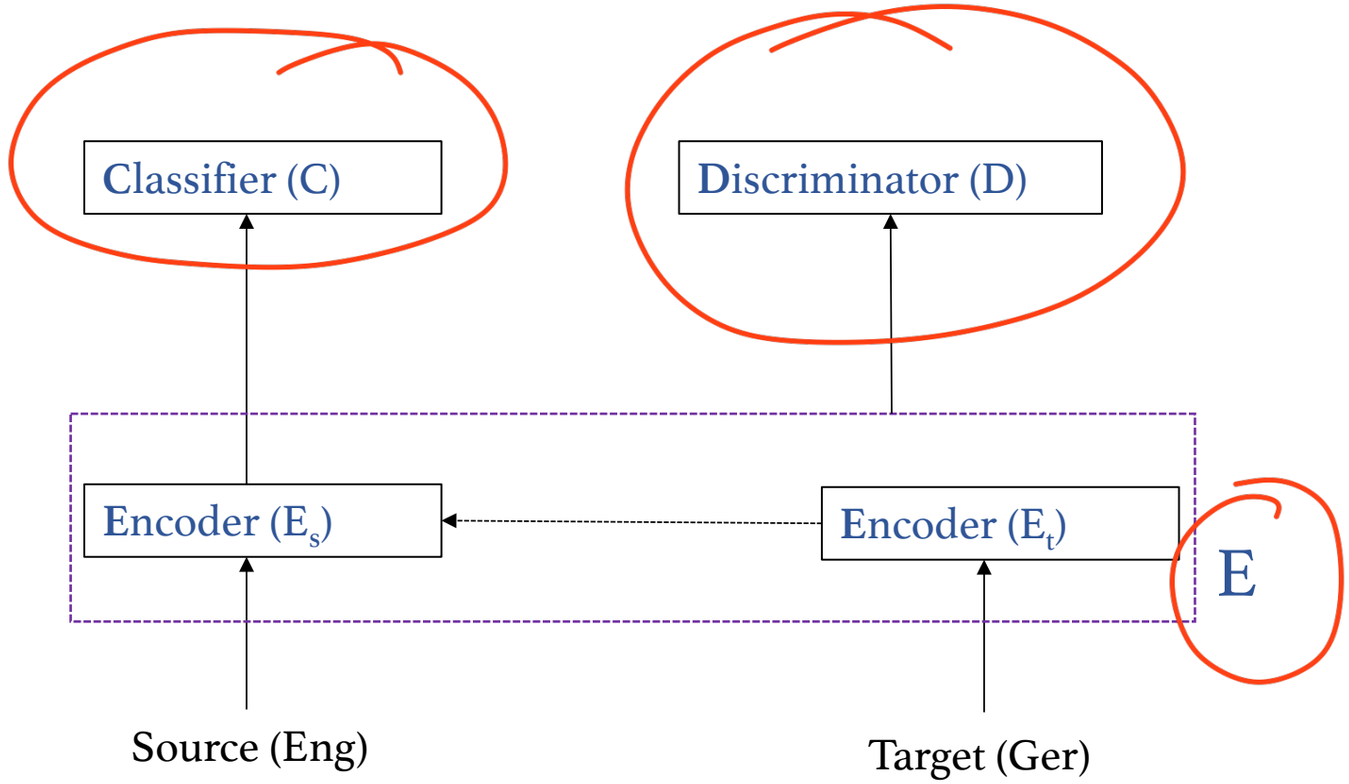
Domain Adversarial Nets

DANN Approach: Learn a common feature space for $\mathcal{D}_s$ and $\mathcal{D}_t$

DANN Framework

Three players

(i) Encoder; (ii) Classifier; (iii) Discriminator



Domain Adversarial Nets

NTU NLP

Design Choices

Three players

(i) Encoder; (ii) Classifier; (iii) Discriminator

Encoder

- Shared [Ganin et al., 2016] vs. Separate [Tzeng et al., 2017]
- Train Encoders concurrently or in steps.
- Generative [Volpi et al., 2017] vs. Discriminative [Tzeng et al., 2017]

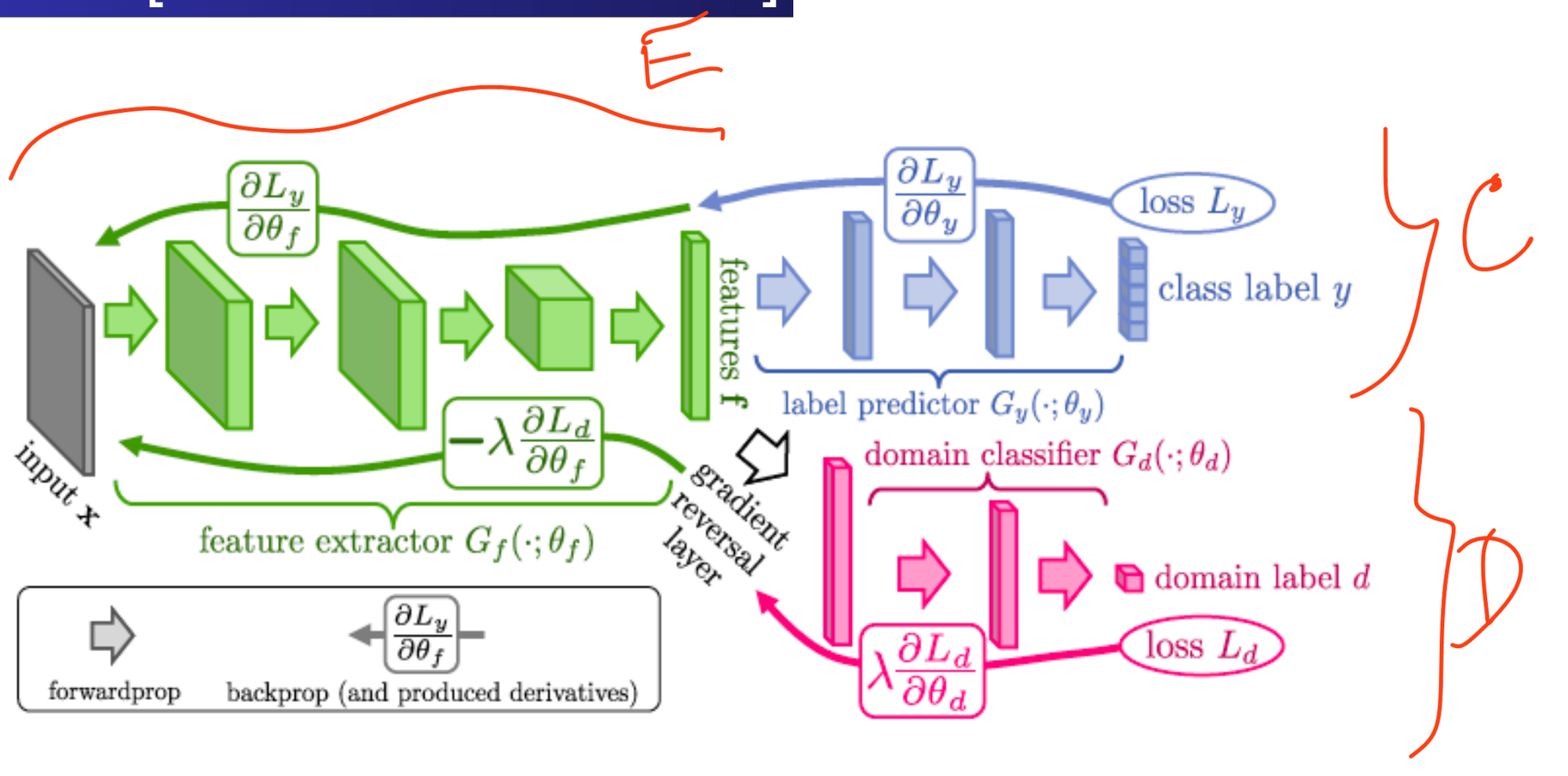
Adversary Loss

- Flip gradient [Ganin et al., 2016]
- GAN loss [Tzeng et al., 2017]

Domain Adversarial Nets

NTU NLP

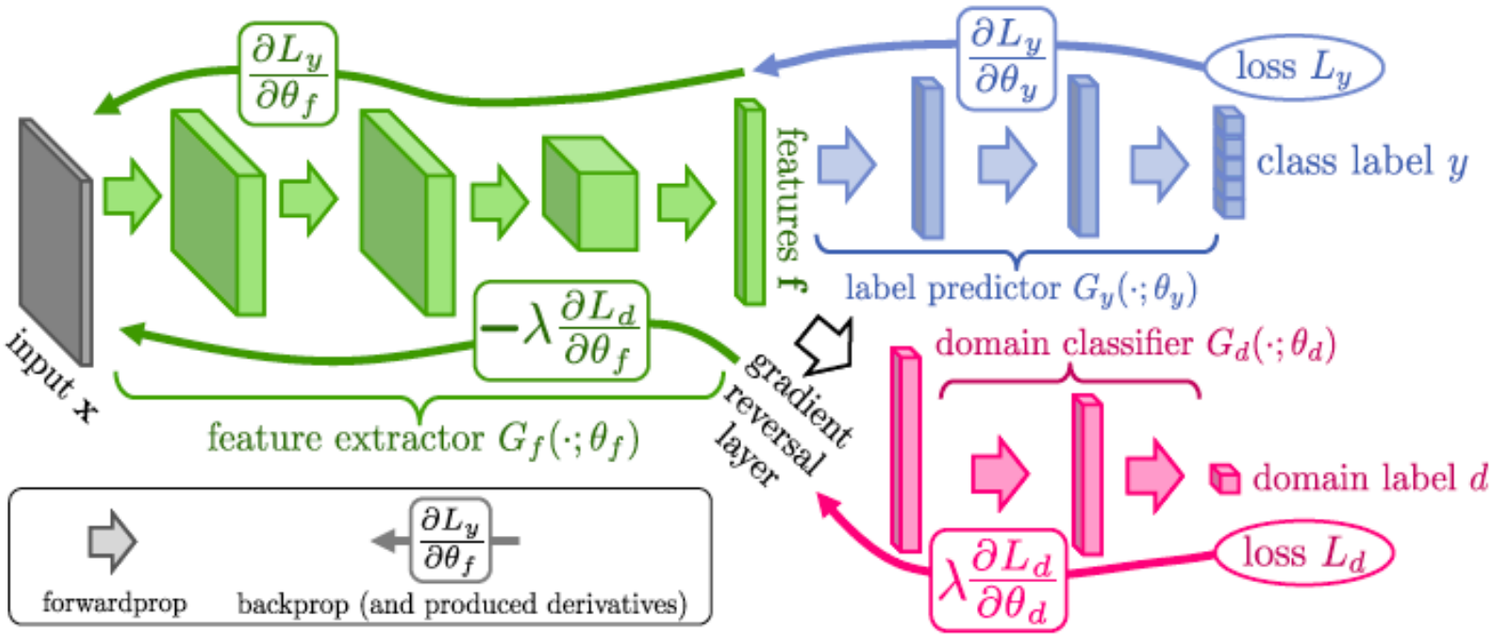
DANN [Ganin et al., 2016]



Domain Adversarial Nets

NTU NLP

DANN [Ganin et al., 2016]



Choices:

- Shared **encoder**. Train source/target concurrently.
- **Discriminatively** trained source.
- Flip gradient for **adversary**.

Domain Adversarial Nets

NTU NLP

DANN [Ganin et al., 2016]

- **Discriminator:**

$$-\mathbb{E}_{x_s \sim X_s} \log D\left(E(x_s)\right) - \mathbb{E}_{x_t \sim X_t} \log\left(1 - D\left(E(x_t)\right)\right).$$

- **Flip Gradient Adversary** (to Encoder):
  $\Rightarrow$ **Encoder (for source):** $\nabla_{\theta_e} \mathbb{E}_{x_s \sim X_s} \log D\left(E(x_s)\right)$
  $\Rightarrow$ **Encoder (for target):** $\nabla_{\theta_e} \mathbb{E}_{x_t \sim X_t} \log\left(1 - D\left(E(x_t)\right)\right)$

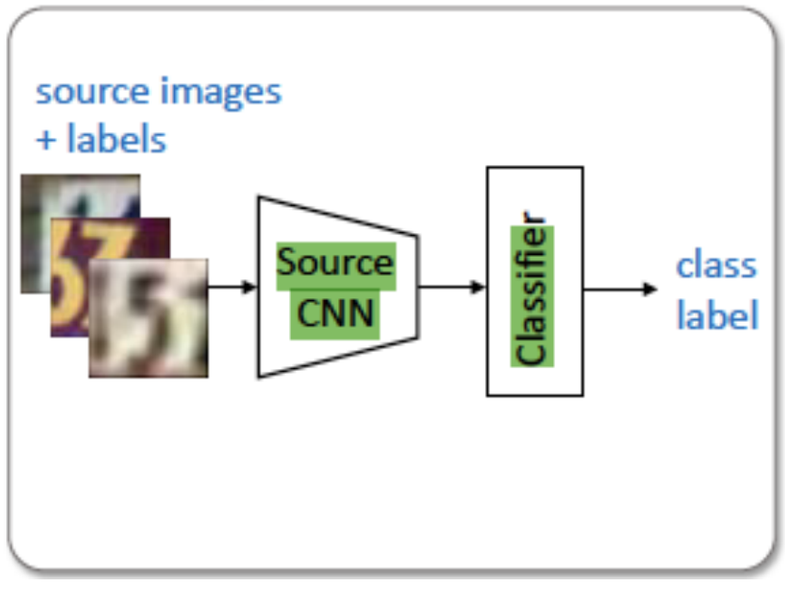
Domain Adversarial Nets

NTU NLP

ADDA [Tzeng et al., 2017]

Step 1: Pre-training on Source

Pre-training

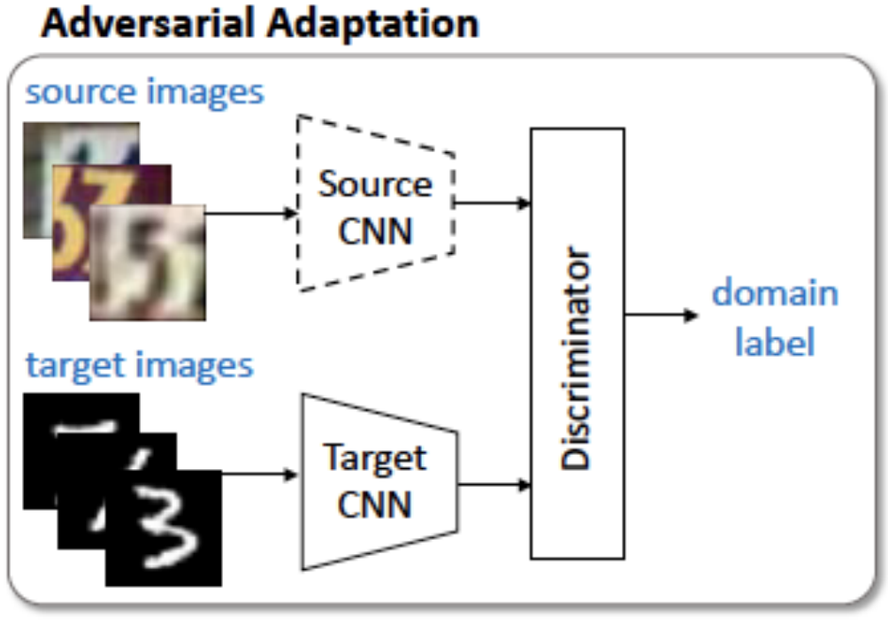


Domain Adversarial Nets

NTU NLP

ADDA [Tzeng et al., 2017]

Step 2: Adapt Target towards (fixed) Source

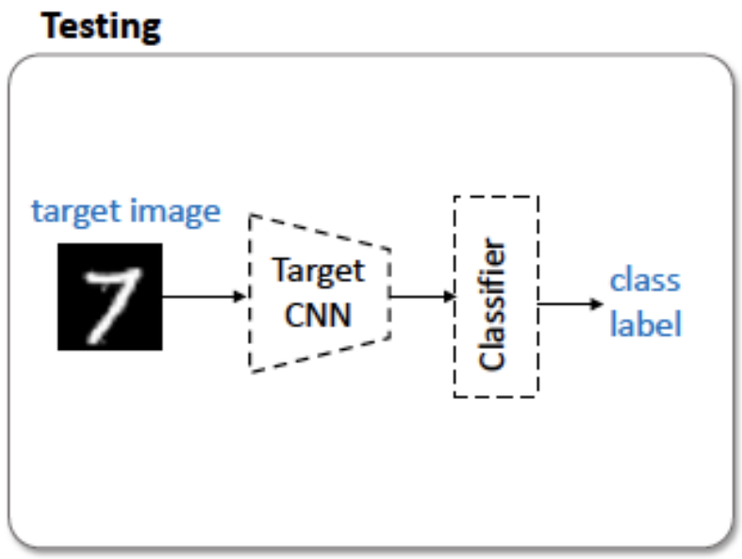


Domain Adversarial Nets

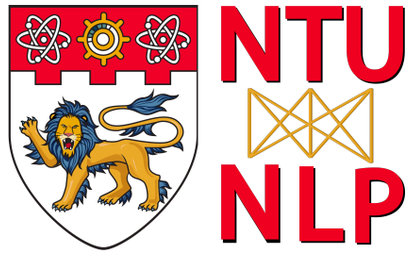
NTU NLP

ADDA [Tzeng et al., 2017]

Step 3: Testing



Domain Adversarial Nets



ADDA [Tzeng et al., 2017]

ADDA Design Choices:

- Separate **encoder** for source and target.
- Train source/target separately.
- **Discriminatively** trained source.
- GAN loss for **adversary**.

Domain Adversarial Nets

NTU NLP

ADDA [Tzeng et al., 2017]

- **Discriminator:**
  $-\mathbb{E}_{x_s \sim X_s} \log D\left(E(x_s)\right) - \mathbb{E}_{x_t \sim X_t} \log\left(1 - D\left(E(x_t)\right)\right).$

- **GAN Adversary**:
  $\Rightarrow$ **Encoder (for target):** - $\nabla_{\theta_e} \mathbb{E}_{x_t \sim X_t} \log\left(D\left(E(x_t)\right)\right)$

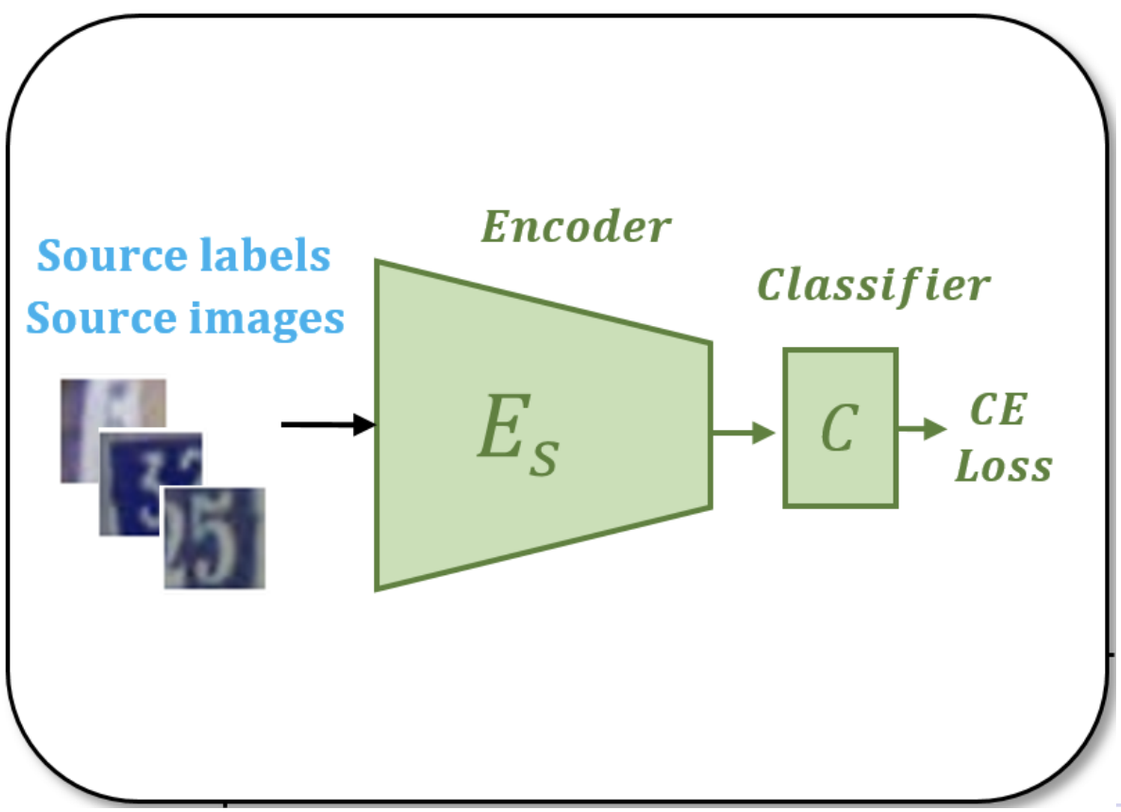
- Recall Adversary for Flip Gradient [Ganin et al., 2016]
  $\Rightarrow$ **Encoder (for source):** $\nabla_{\theta_e} \mathbb{E}_{x_s \sim X_s} \log D\left(E(x_s)\right)$
  $\Rightarrow$ **Encoder (for target):** $\nabla_{\theta_e} \mathbb{E}_{x_t \sim X_t} \log\left(1 - D\left(E(x_t)\right)\right)$

Domain Adversarial Nets

NTU NLP

Feature Augmented DANN [Volpi et al., 2017]

Step 0: Pre-training on Source

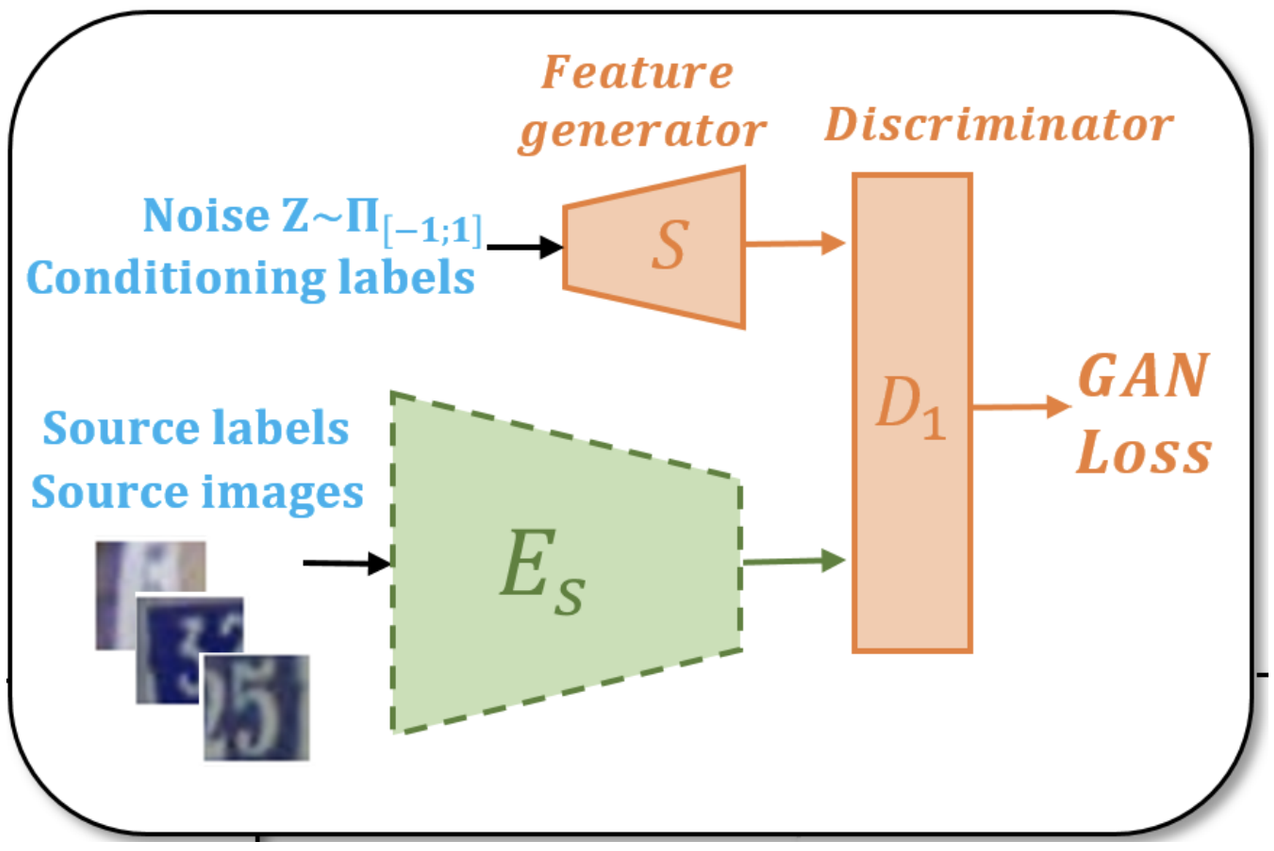


Domain Adversarial Nets

NTU NLP

Feature Augmented DANN [Volpi et al., 2017]

Step 1: Feature Generation with CGAN



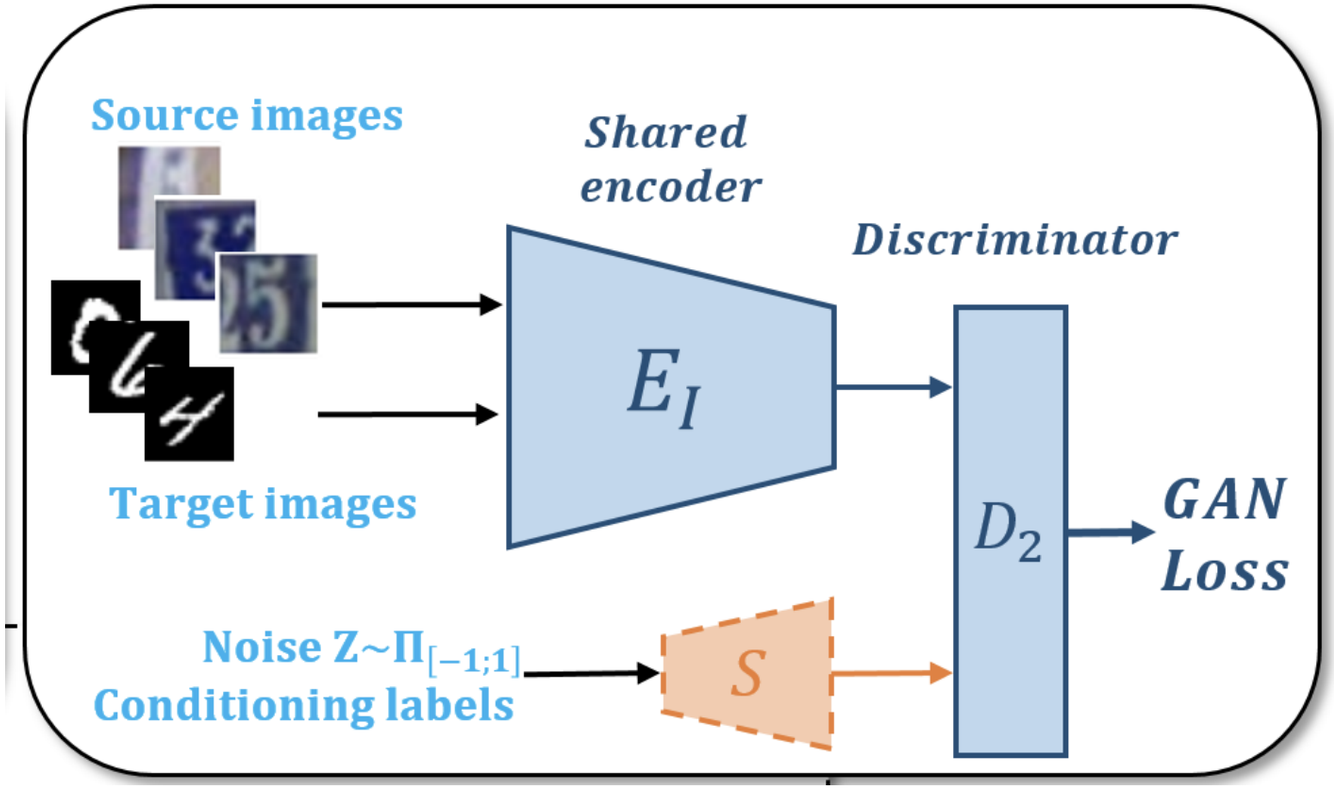
Domain Adversarial Nets

NTU NLP

Feature Augmented DANN [Volpi et al., 2017]

Step 2: Feature Adaptation

Step 2: training E_I, D_2

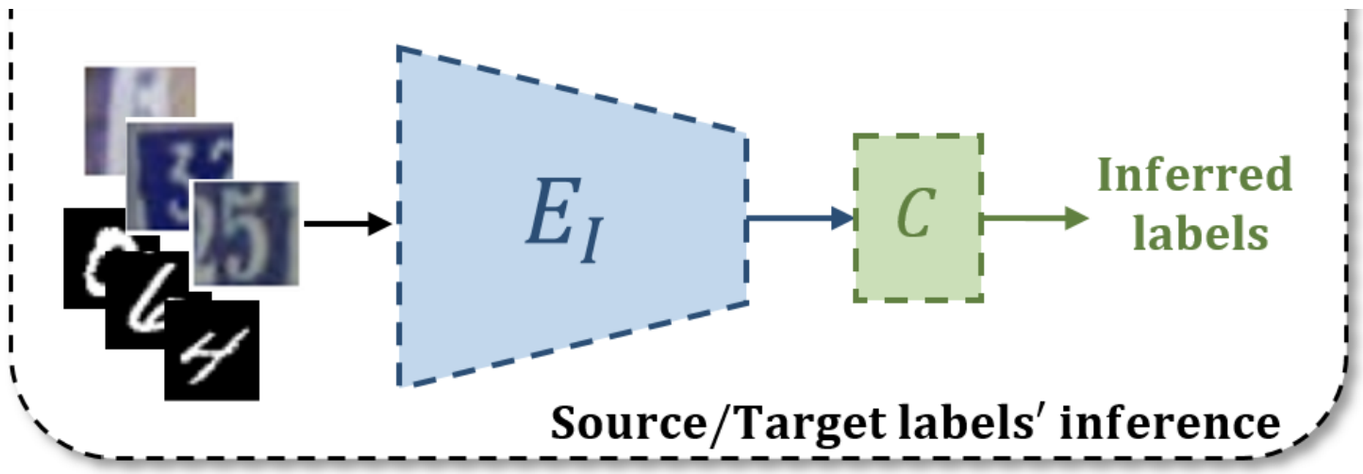


Domain Adversarial Nets

NTU NLP

Feature Augmented DANN [Volpi et al., 2017]

Step 3: Testing



Domain Adversarial Nets

NTU NLP

Feature Augmented DANN [Volpi et al., 2017]

FA-DANN Design Choices:

- Same shared **encoder** for source and target.
- Train source/target separately.
- **Discriminatively** trained source + **Generative** features.
- GAN loss for **adversary**.

Domain Adversarial Nets

NTU NLP

Results

- ADDA [Tzeng et al., 2017]

Method	MNIST → USPS	USPS → MNIST	SVHN → MNIST
Source only	0.752 ± 0.016	0.571 ± 0.017	0.601 ± 0.011
Gradient reversal	0.771 ± 0.018	0.730 ± 0.020	0.739 [19]
Domain confusion	0.791 ± 0.005	0.665 ± 0.033	0.681 ± 0.003
CoGAN	0.912 ± 0.008	0.891 ± 0.008	did not converge
ADDA (Ours)	0.894 ± 0.002	0.901 ± 0.008	0.760 ± 0.018

Table 2: Experimental results on unsupervised adaptation among MNIST, USPS, and SVHN.

Domain Adversarial Nets

NTU NLP

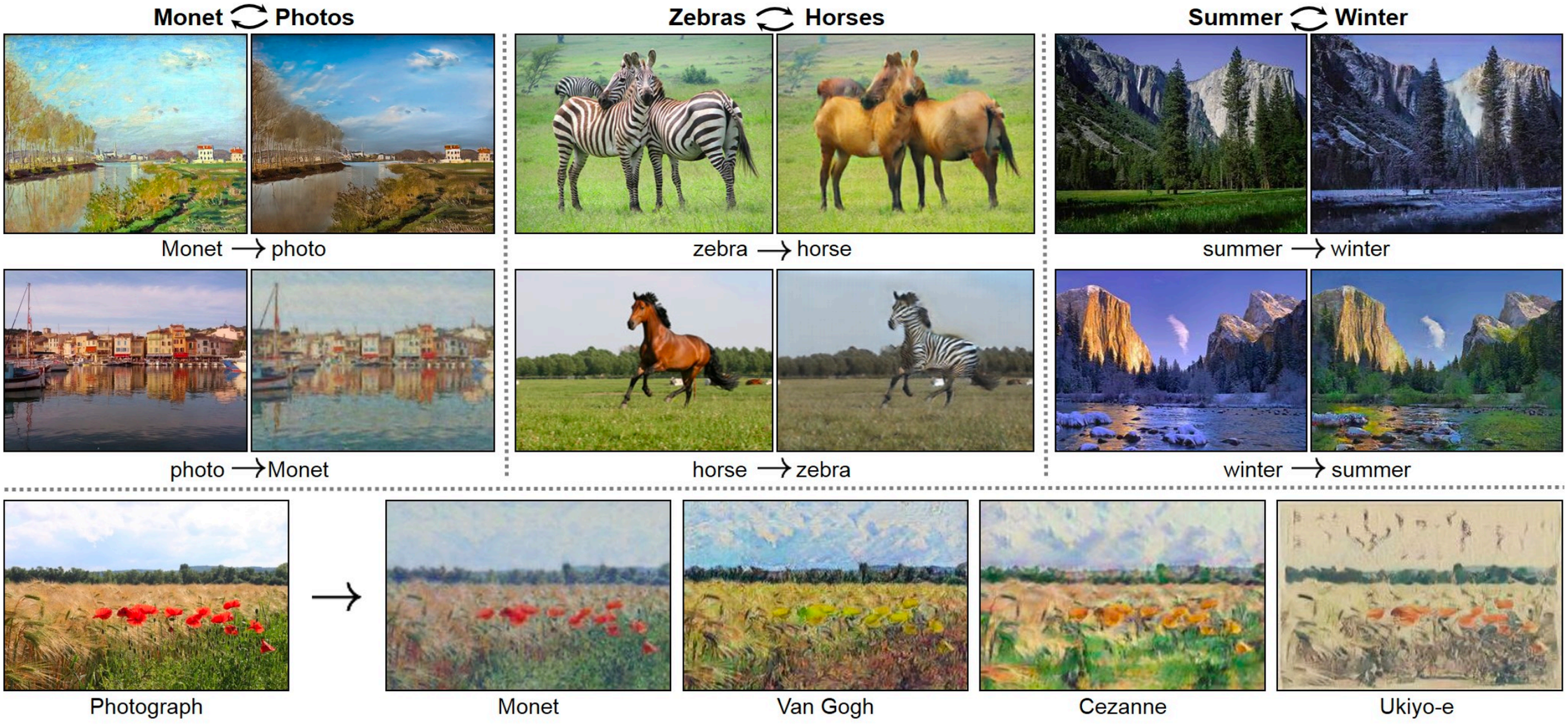
Results

- FA-DANN [Volpi et al., 2017]

	SVHN→MNIST	MNIST→USPS$_{P1}$	MNIST→USPS$_{P2}$	USPS→MNIST	SYN→SVHN	NYUD
Source	0.682	0.723	0.797	0.627	0.885	0.139
DANN [9, 10]	0.739	0.771 ± 0.018 [35]	-	0.730 ± 0.020 [35]	0.911	-
DDC [35]	0.681 ± 0.003	0.791 ± 0.005	-	0.665 ± 0.033		-
DSN [3]	0.827	-	-	-	0.912	-
ADDA [35]	0.760 ± 0.018	0.894 ± 0.002	-	**0.901 ± 0.008**		0.211
Tri [29]	0.862	-	-	-	**0.931**	-
DTN [33]	0.844*	-	-	-	-	-
PixelDA** [2]	-	-	0.959	-	-	-
UNIT [18]	**0.905***	-	0.960	-	-	-
CoGANs [19]	no conv. [35]	0.912 ± 0.008	0.957 [18]	0.891 ± 0.008	-	-
LS-ADDA	0.743 ± 0.028	0.914 ± 0.000	0.912 ± 0.003	**0.910 ± 0.004**	0.908 ± 0.004	no conv.
Ours (DI)	0.851 ± 0.026	0.914 ± 0.000	0.954 ± 0.002	0.879 ± 0.005	0.925 ± 0.002	0.287 ± 0.002
Ours (DIFA)	**0.897 ± 0.020**	**0.923 ± 0.001**	0.962 ± 0.002	0.897 ± 0.005	**0.930 ± 0.002**	**0.313 ± 0.002**
Target	0.992	0.999	0.999	0.975	0.913	0.468 [35]

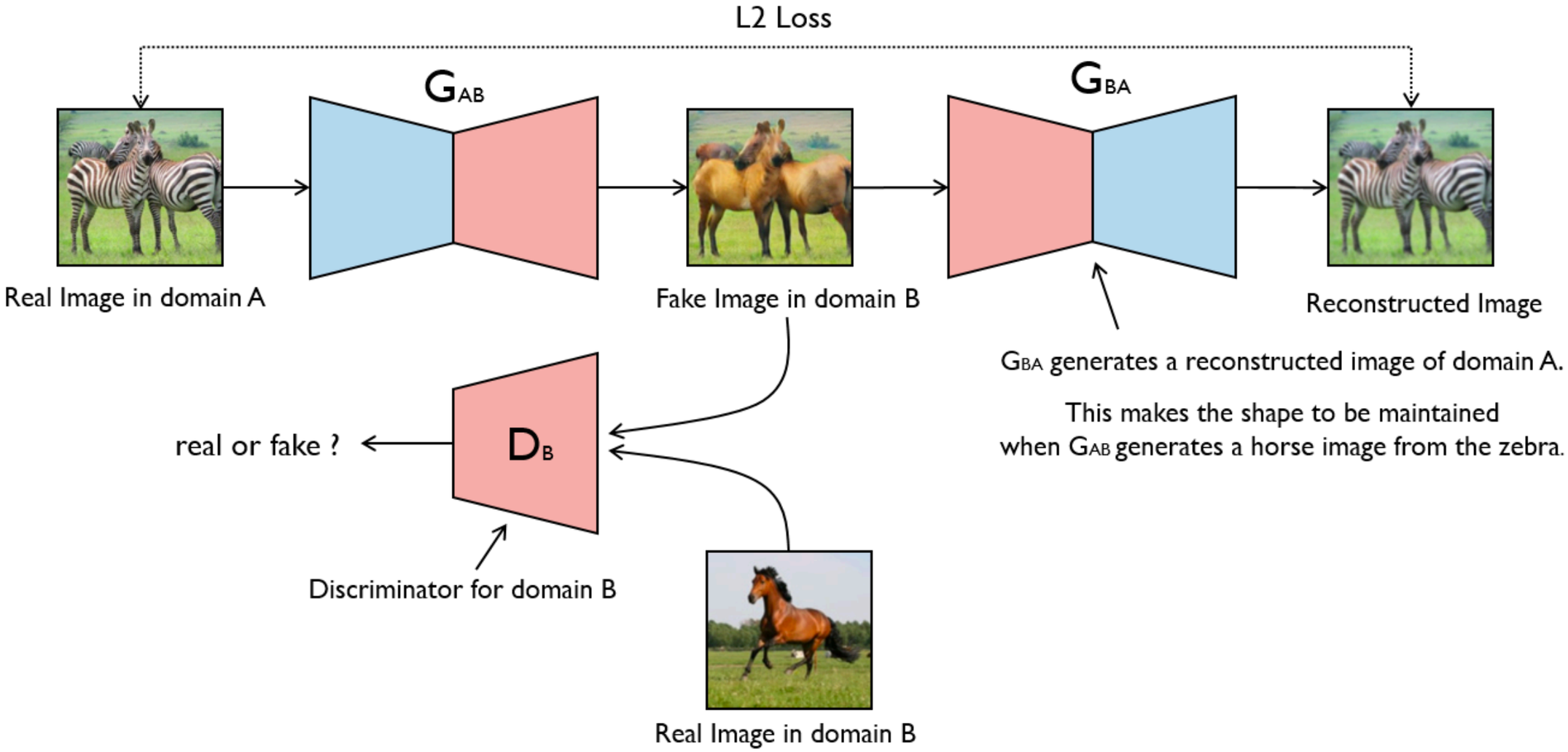
More Examples of Adversarial Learning

NTU NLP



CycleGAN (Zhu et al., 2017)

More Examples of Adversarial Learning

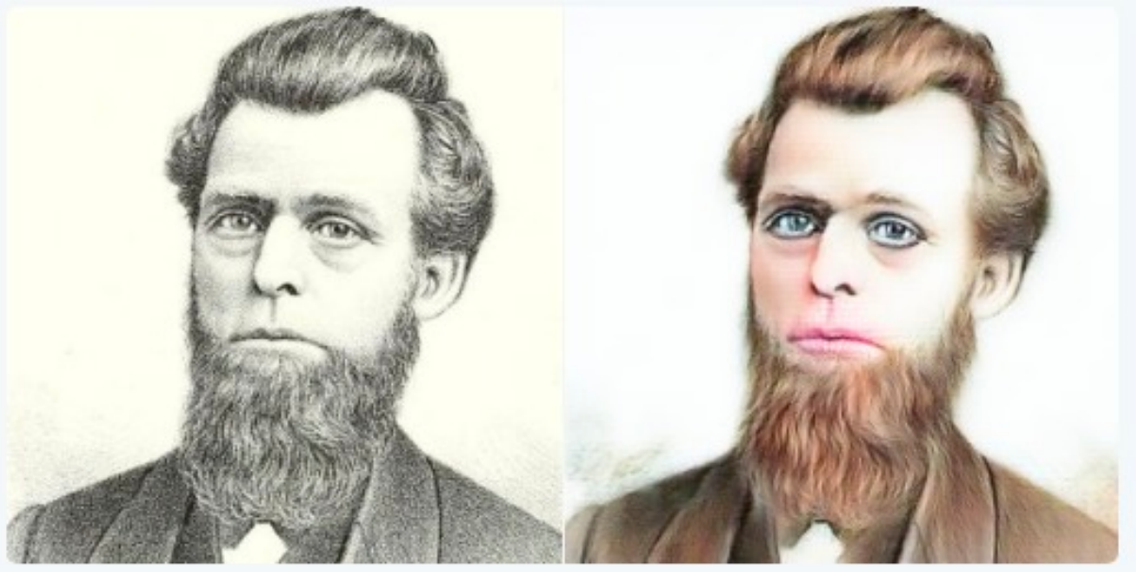


CycleGAN (Zhu et al., 2017)

More Examples of Adversarial Learning

CycleGAN (Zhu et al., 2017)

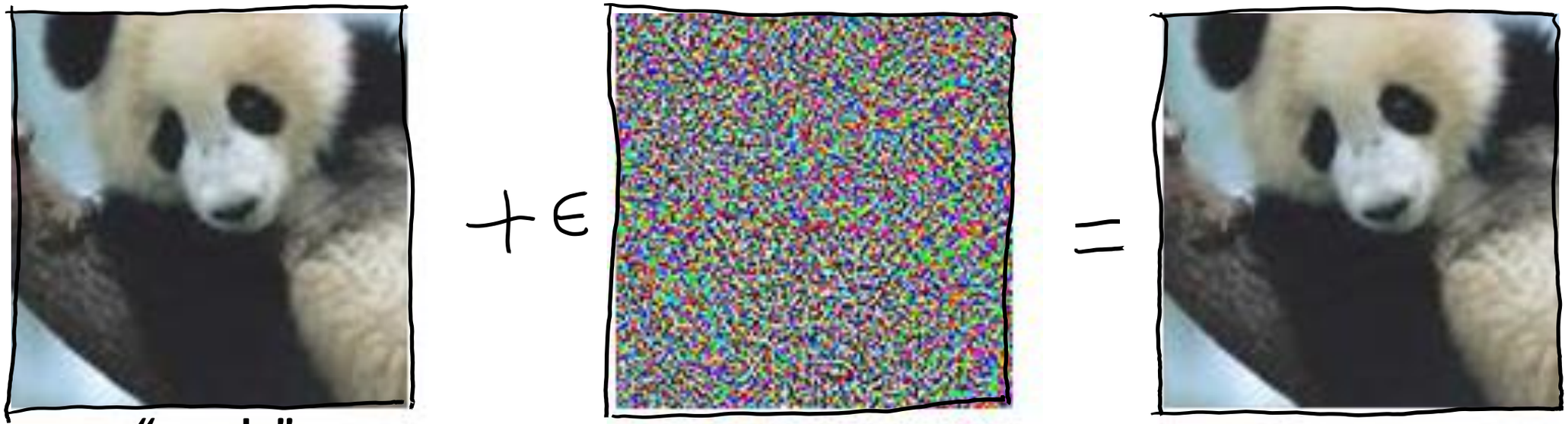
More Examples of Adversarial Learning



Failure case

CycleGAN (Zhu et al., 2017)

Part 2: Adversarial Attacks



“panda”

57.7% confidence

“gibbon”

99.3% confidence

[Goodfellow et al, ICLR 2015]