Yandex

Generative models

how not to train generative models

NOV 29, 2018

Just Heuristic, jheuristic@yandex-team.ru
thanks to lena-voita@, bkovarsky@, norpadon@

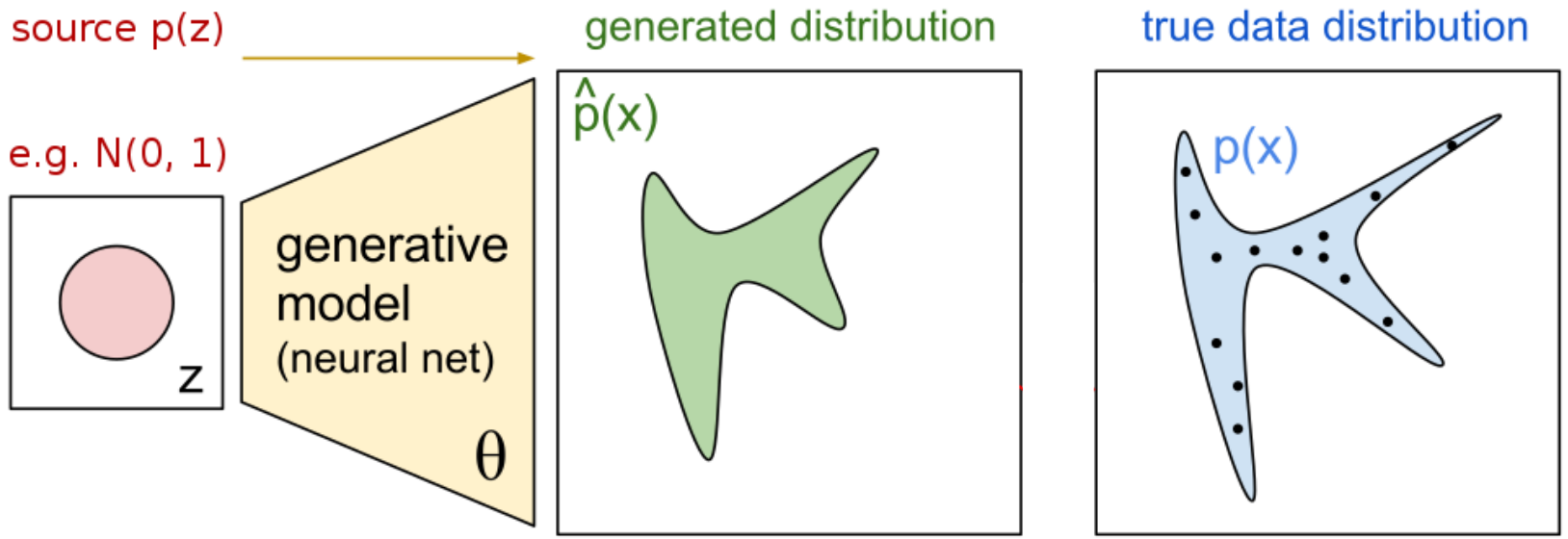
Menu

- Generative Models 101
- Adversarial Networks & Variational Autoencoders
- NLP Case Studies

Learning distributions

Target distribution $p(x)$; Generated distribution $\hat{p}(x)$

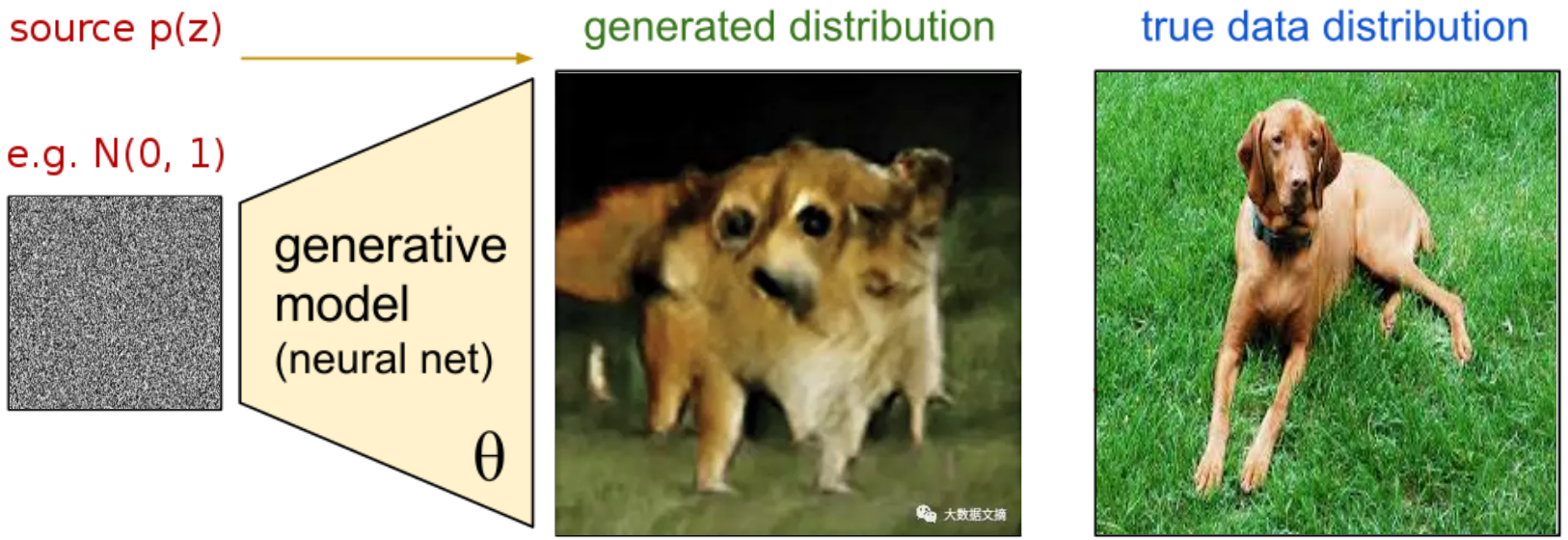
$\hat{p}(x)$ may be conditioned on $z \sim p(z)$



Learning distributions

Target distribution $p(x)$; Generated distribution $\hat{p}(x)$

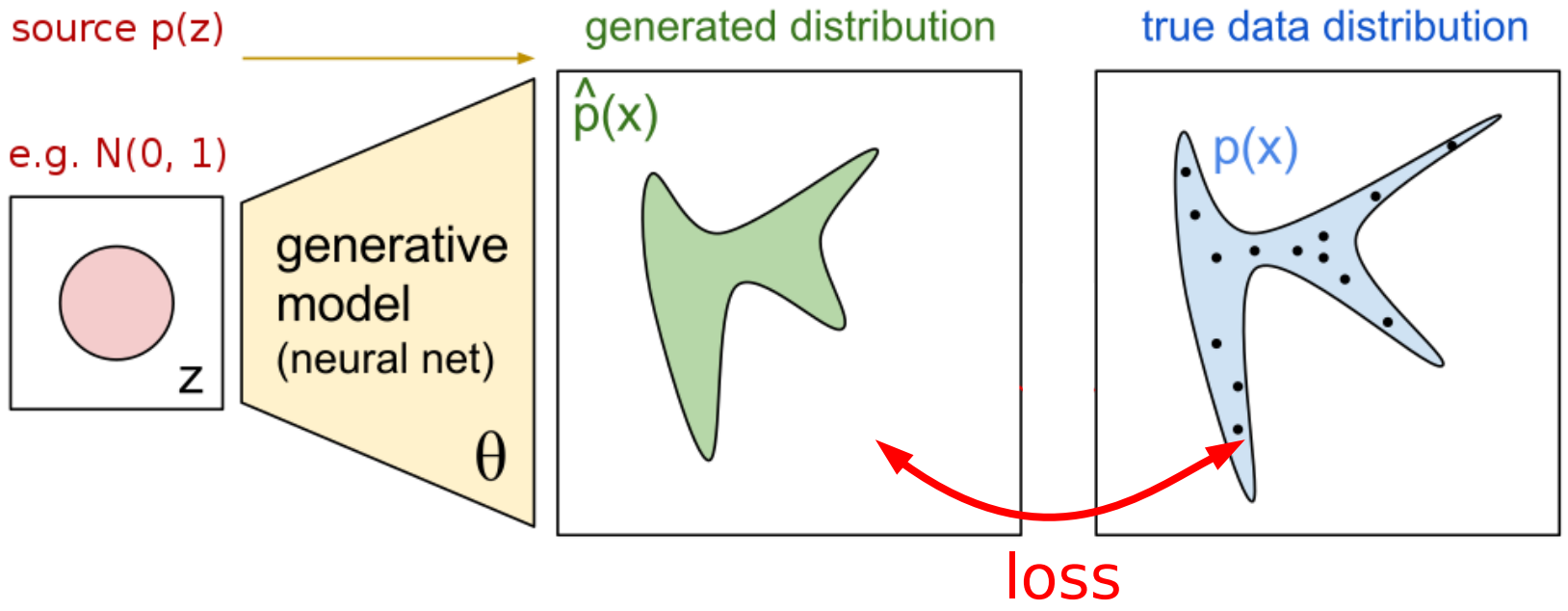
$\hat{p}(x)$ may be conditioned on $z \sim p(z)$



Learning distributions

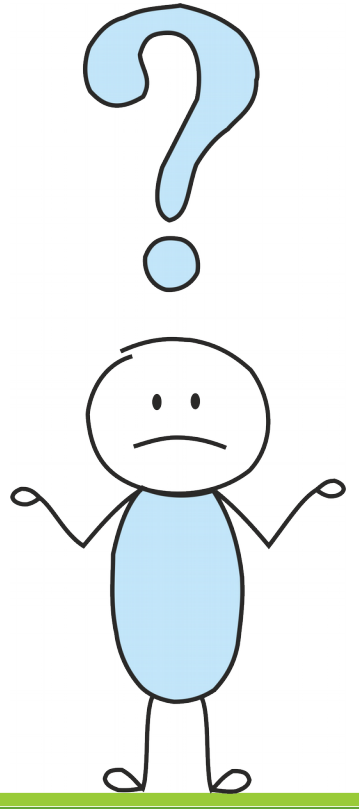
Target distribution $p(x)$; Generated distribution $\hat{p}(x)$

Goal: make $\hat{p}(x)$ match $p(x)$



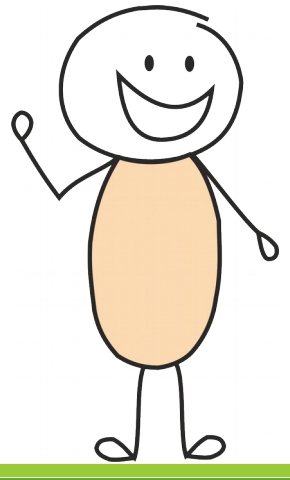
Learning distributions

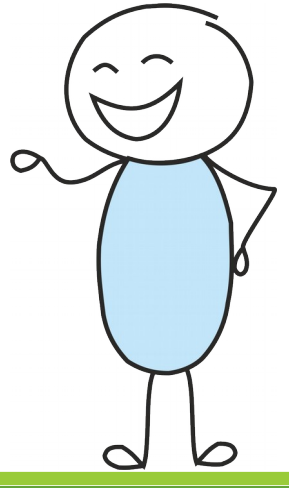
Q: Why would you want to learn a distribution?



Learning distributions

- Language models
- Classification
- Seq2seq
- Embeddings
- Sequence Labeling
- Everything!





Learning distributions

Target distribution $p(x)$; Generated distribution $\hat{p}(x)$

Goal: make $\hat{p}(x)$ match $p(x)$

- Max Likelihood Fit
- Generative Adversarial Networks
- Variational Autoencoders

Learning distributions

Target distribution $p(x)$; Generated distribution $\hat{p}(x)$

Goal: make $\hat{p}(x)$ match $p(x)$

- **Max Likelihood Fit** *(recap)*
- Generative Adversarial Networks
- Variational Autoencoders

Recap: max likelihood

- Sample x ~ p(x) from real data, compute likelihood under $\hat{p}$

- Maximize log-likelihood

$$L(x) = -\sum_{x \sim p(x)} \log \hat{p}(x)$$

aka crossentropy, logloss

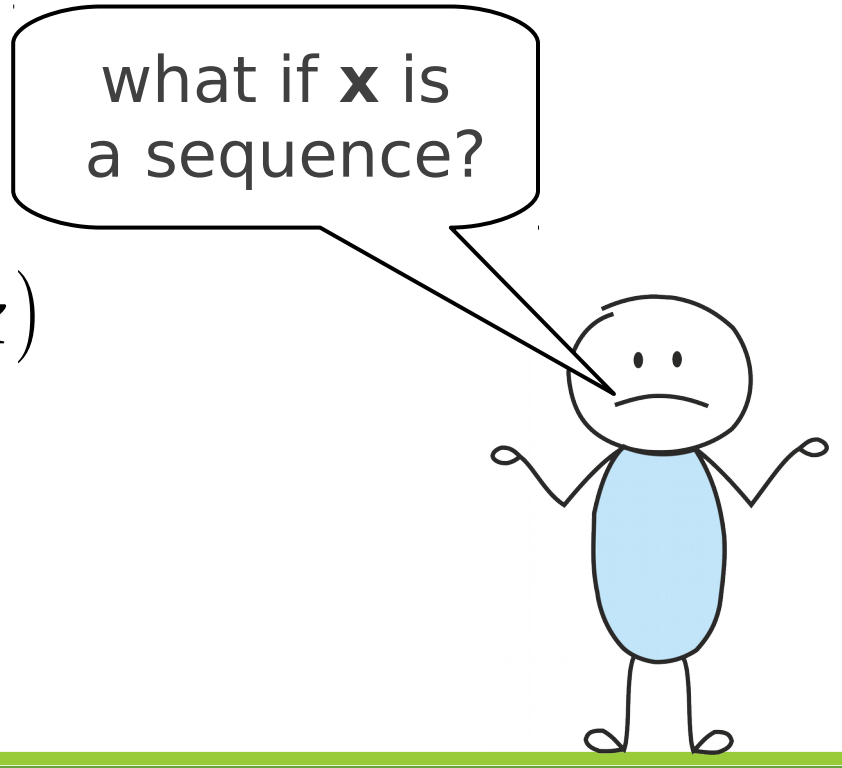
Recap: max likelihood

- Sample x ~ p(x) from real data, compute likelihood under $\hat{p}$

- Maximize log-likelihood

$$L(x) = -\sum_{x \sim p(x)} \log \hat{p}(x)$$

aka crossentropy, logloss



Recap: max likelihood

- If x is structured, apply chain rule

$$\hat{p}(x) = \hat{p}(x_0, \ldots, x_T) = \prod_{i=0}^{T} \hat{p}(x_i | x_0, \ldots, x_{i-1})$$

- Log-likelihood: autoregressive

$$L(x) = -\sum_{x \sim p(x)} \sum_{i=0}^{T} \log \hat{p}(x_i | x_0, \ldots, x_{i-1})$$

Recap: max likelihood

- If x is structured, apply chain rule

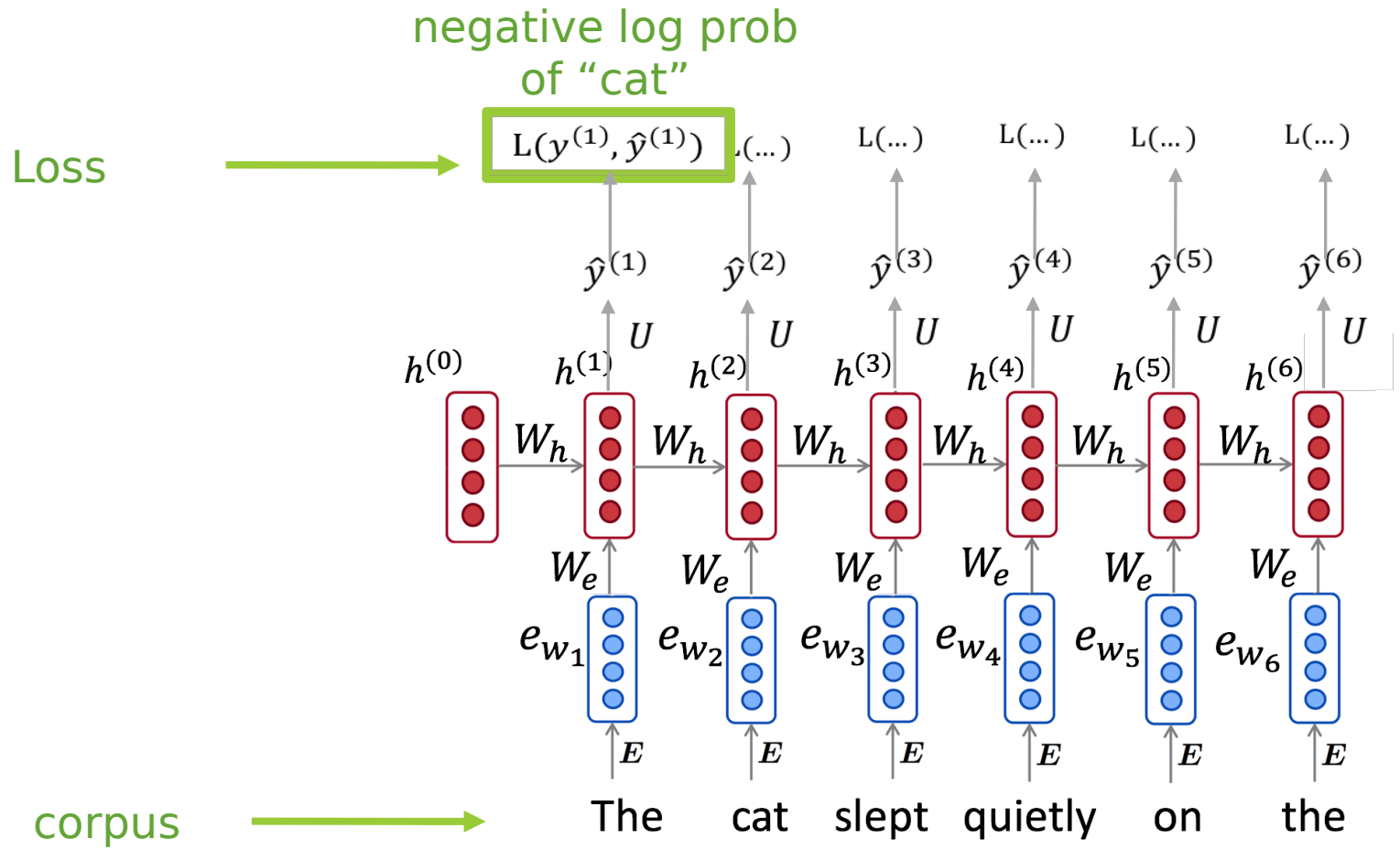
$$\hat{p}(x) = \hat{p}(x_0, \ldots, x_T) = \prod_{i=0}^{T} \hat{p}(x_i | x_0, \ldots, x_{i-1})$$

- Log-likelihood: autoregressive

$$L(x) = -\sum_{x \sim p(x)} \sum_{i=0}^{T} \log \hat{p}(x_i | x_0, \ldots, x_{i-1})$$

Examples: Language models, Seq2Seq, vision: pixelCNN

Recap: language models



Recap: max likelihood

- Pros:
  - Training is easy
  - Explicit probability

- Cons:
  - Requires a fixed order
  - Inference may be hard

Recap: max likelihood

- Pros:
  - Training is easy — *log P(next | prefix)*
  - Explicit probability — *measure perplexity for free*
- Cons:
  - Requires a fixed order — *left to right? and if x=image? graph?*
  - Inference may be hard — *one token at a time, beam search, etc*

Learning distributions

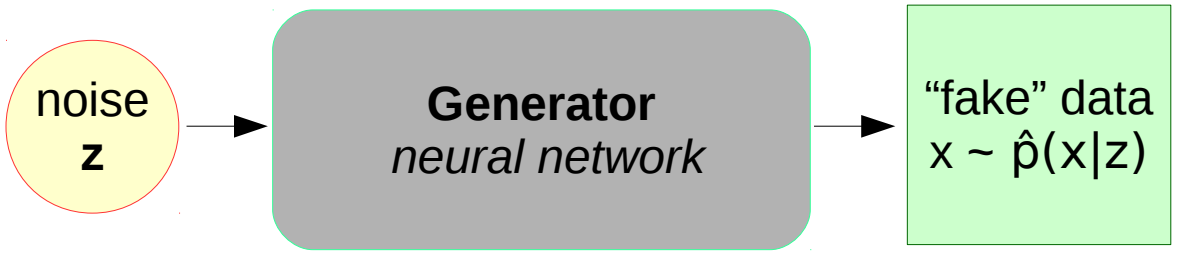
Target distribution $p(x)$; Generated distribution $\hat{p}(x)$

Goal: make $\hat{p}(x)$ match $p(x)$

- Max Likelihood Fit
- **Generative Adversarial Networks**
- Variational Autoencoders

Generative Adversarial Nets

Generator: $\hat{p}(\mathbf{x} \mid \mathbf{z})$

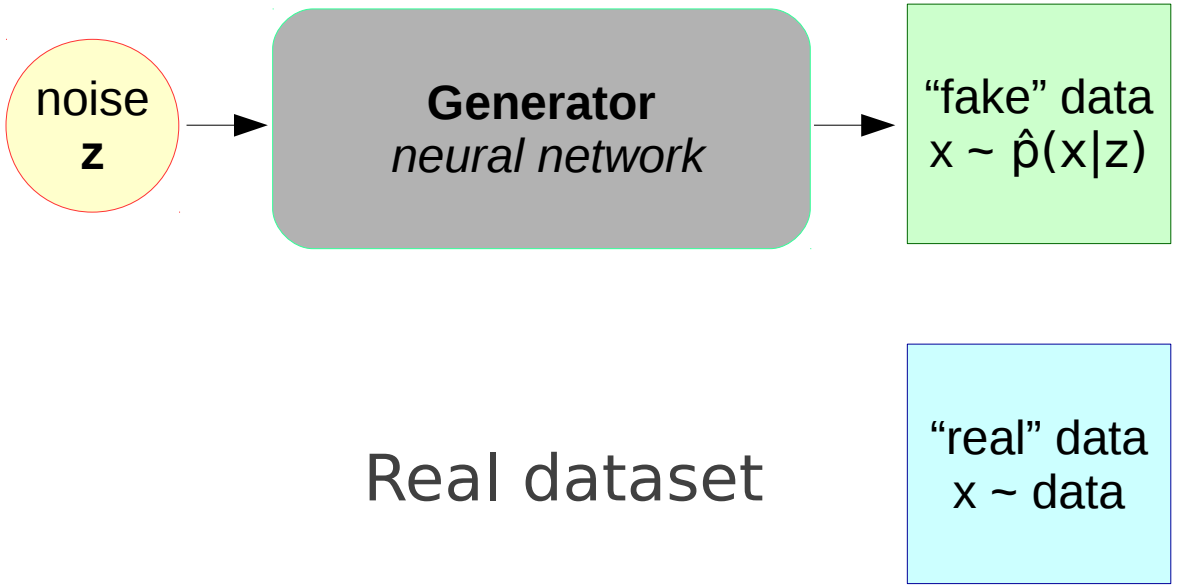


takes random noise, e.g. U(0,1)

tries to generate realistic objects

Generative Adversarial Nets

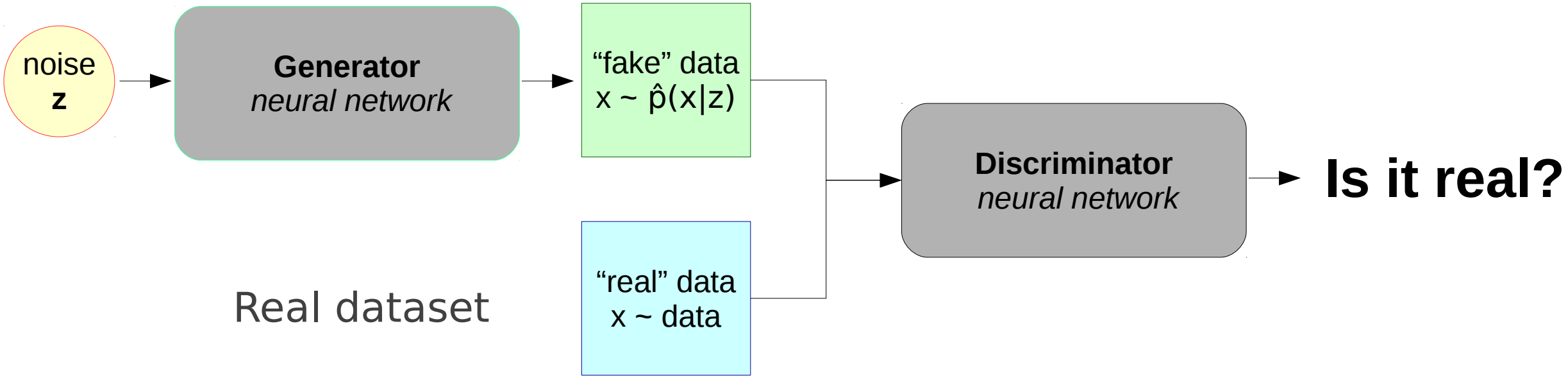
Generator: $\hat{p}(\mathbf{x} \mid \mathbf{z})$



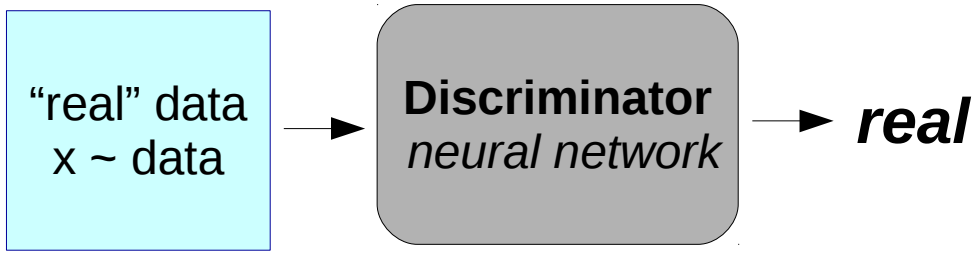
Generative Adversarial Nets

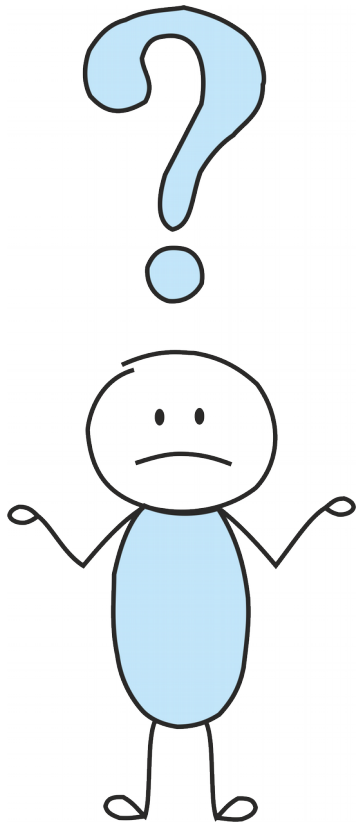
Generator: $\hat{p}(\mathbf{x} \mid \mathbf{z})$

Discriminator: $P(\text{real} \mid \mathbf{x})$



Training discriminator



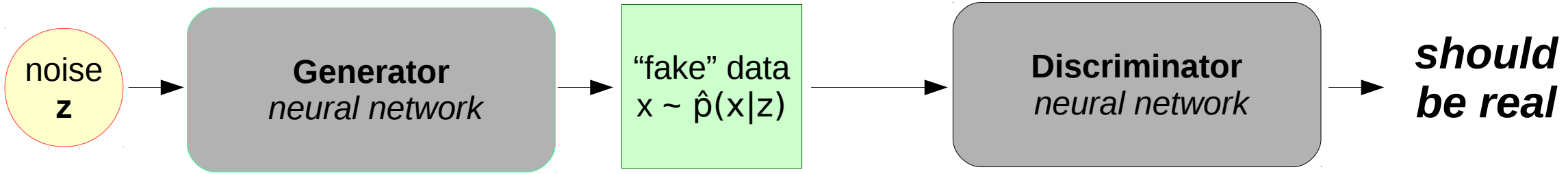


Q: *where do we get negative samples?*

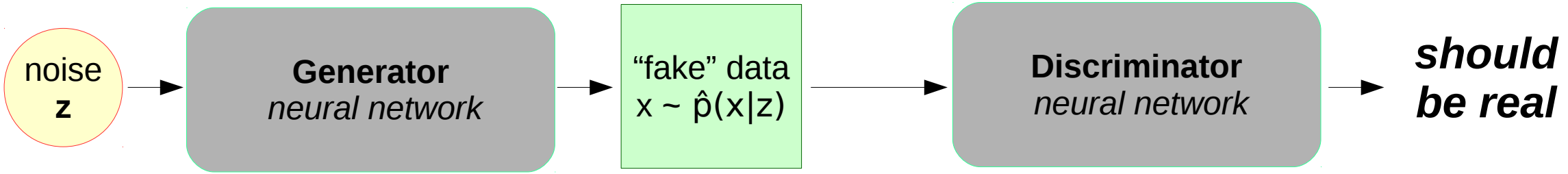
Training discriminator

$$Loss_{disc} = -E_{x \sim data} \log p(real|x) \; - \; E_{z \sim p(z)} \log p(fake|gen(z))$$

Training generator



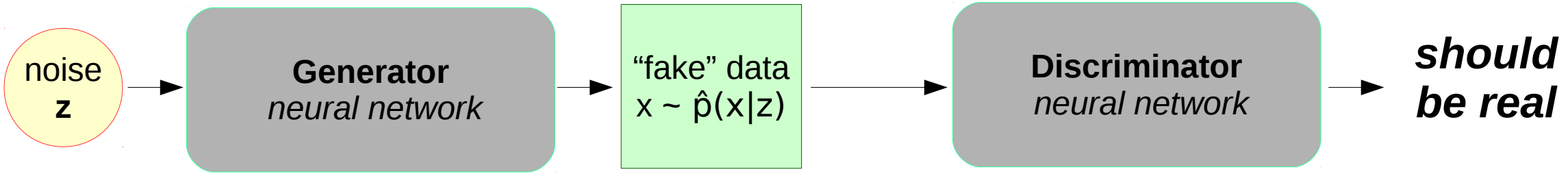
Training generator



$$Loss_{gen} = -E_{z \sim p(z)} \log p\left(real | gen(z)\right)$$

Q: *How do we train?*

Training generator



$$Loss_{gen} = -E_{z \sim p(z)} \log p\left(real | gen(z)\right)$$

Training: *just backprop :)*

Training GANs

Forever:

- for k in 1...K
  train_discriminator(data, noise)
- For m in 1...M
  train_generator(noise)

Training GANs

Forever:

- for k in 1...K
  train_discriminator(data, noise)
- For m in 1...M
  train_generator(noise)

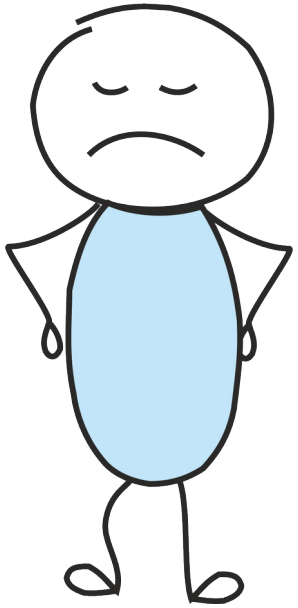
Q: what happens if we mess up **K** or **M**?

Training GANs

What happens if we mess up **K** or **M**?

Discriminator is too fast: gradients die

Generator is too fast: instability
or collapse to local optimum (mode)



Adversarial Zoo

- Original GAN: crossentropy loss — arXiv:1406.2661
- Energy-based GAN: autoencoders — arXiv:1609.03126
- LS GAN: squared error to +1 or -1 — arXiv:1611.04076

- Tons of other GANs! — https://bit.ly/2zqYI9F

Wasserstein GAN

Engineer’s view:

- Discriminator output is arbitrary real number
- Discriminator: D(x) - D(G(z)) → max
- Generator: D(G(z)) → max

Discriminator is constrained to be k-lipshitz by regularization or weight clipping

NLP applications

Why would you ever need GANs?

Adversarial dictionary learning

You are given a large dataset of texts in english and ~~elvish~~ another less known language.

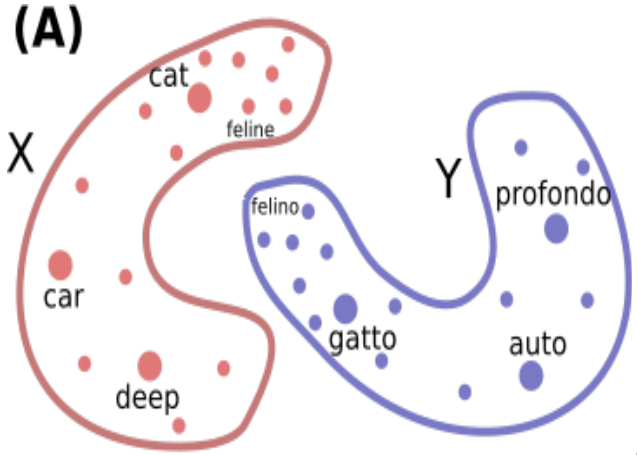
You were only able to translate a few words

Goal: learn translations for each word

Source: https://arxiv.org/abs/1710.04087

Adversarial dictionary learning

a) learn monolingual embeddings (w2v/glove/fasttext)

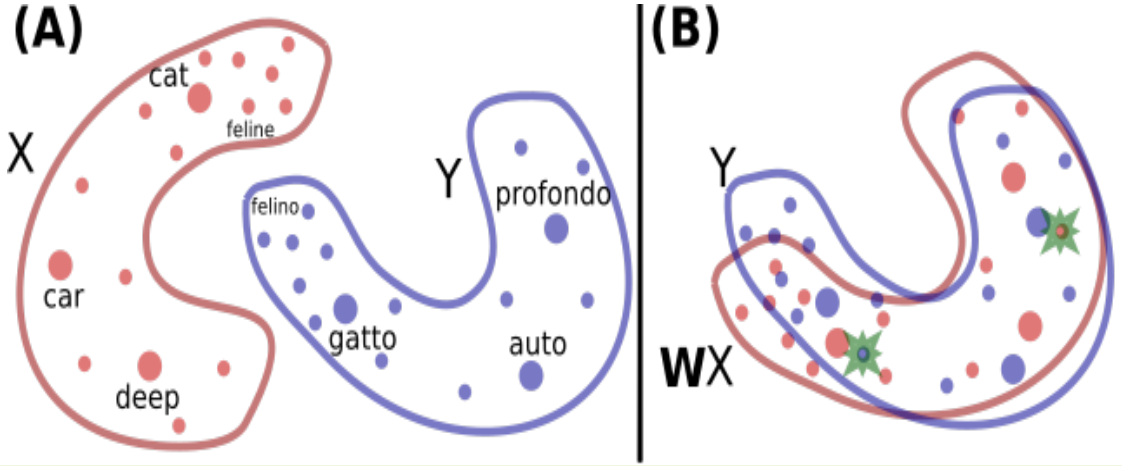


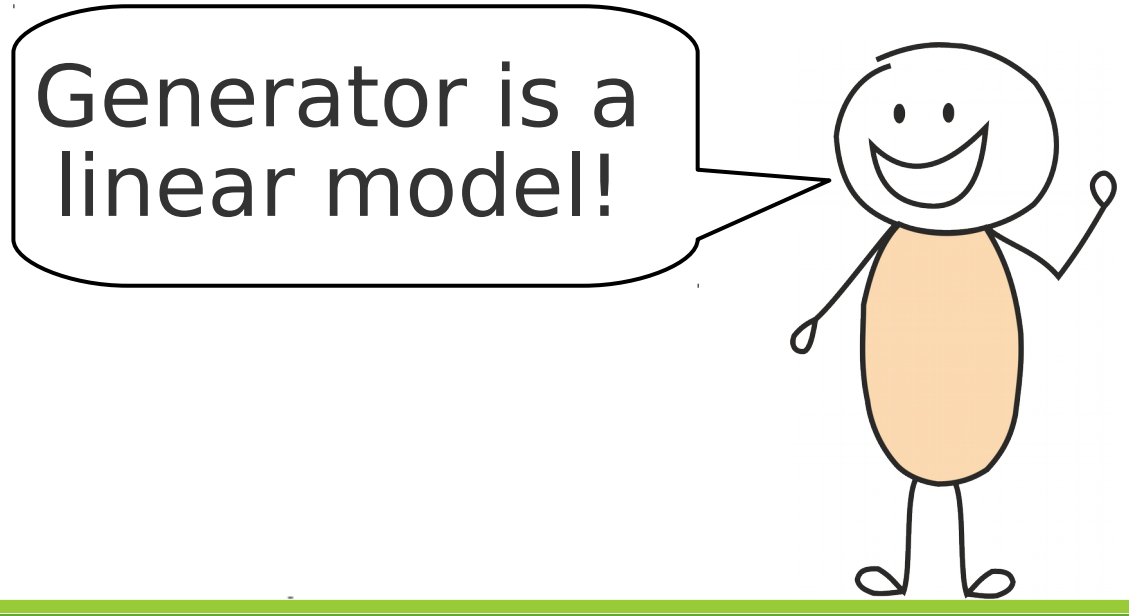
Source: https://arxiv.org/abs/1710.04087

Adversarial dictionary learning

a) learn monolingual embeddings (w2v/glove/fasttext)

b) map embeddings together with GAN loss





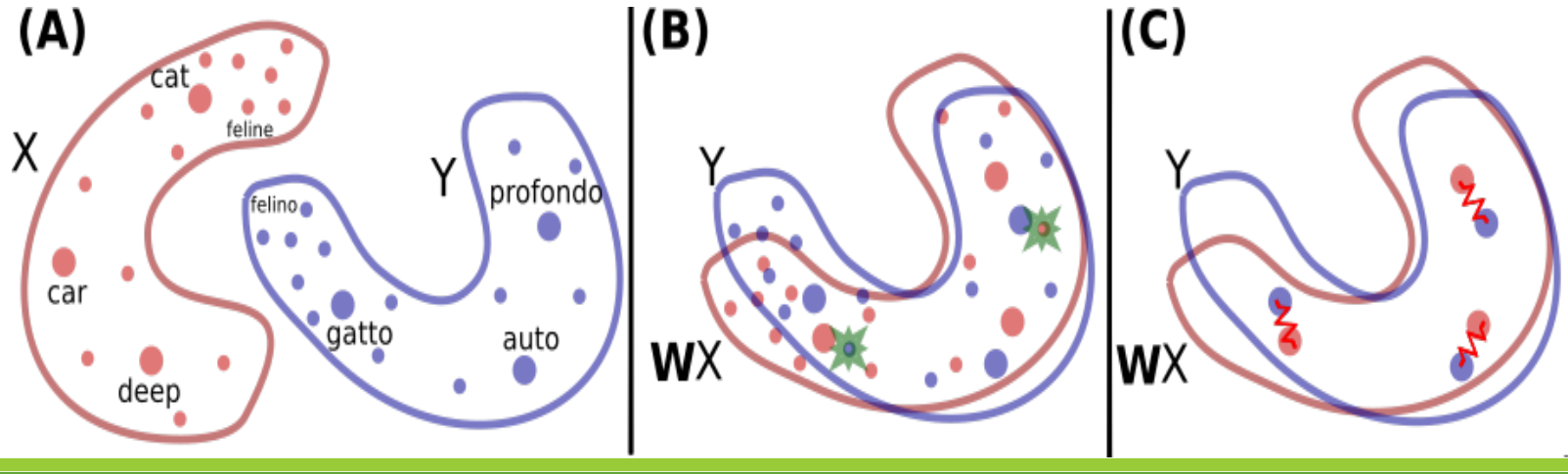
Source: https://arxiv.org/abs/1710.04087

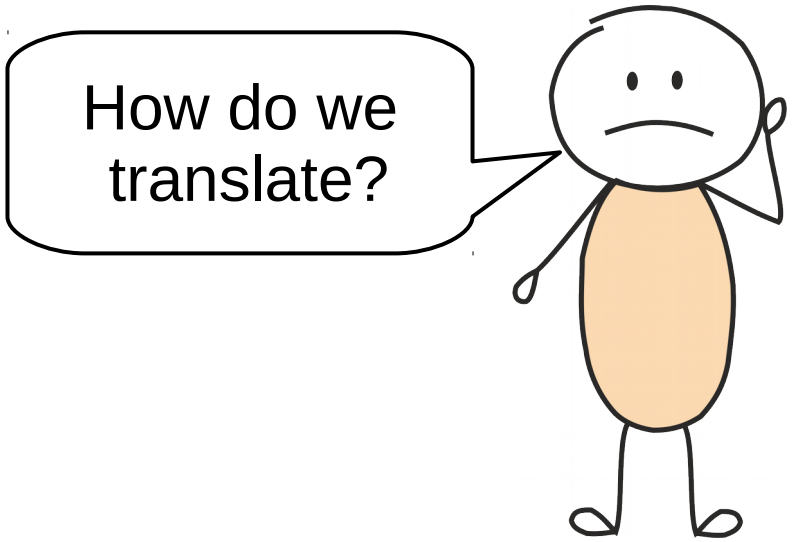
Adversarial dictionary learning

a) learn monolingual embeddings (w2v/glove/fasttext)

b) map embeddings together with GAN loss

c) use MSE on few translated words





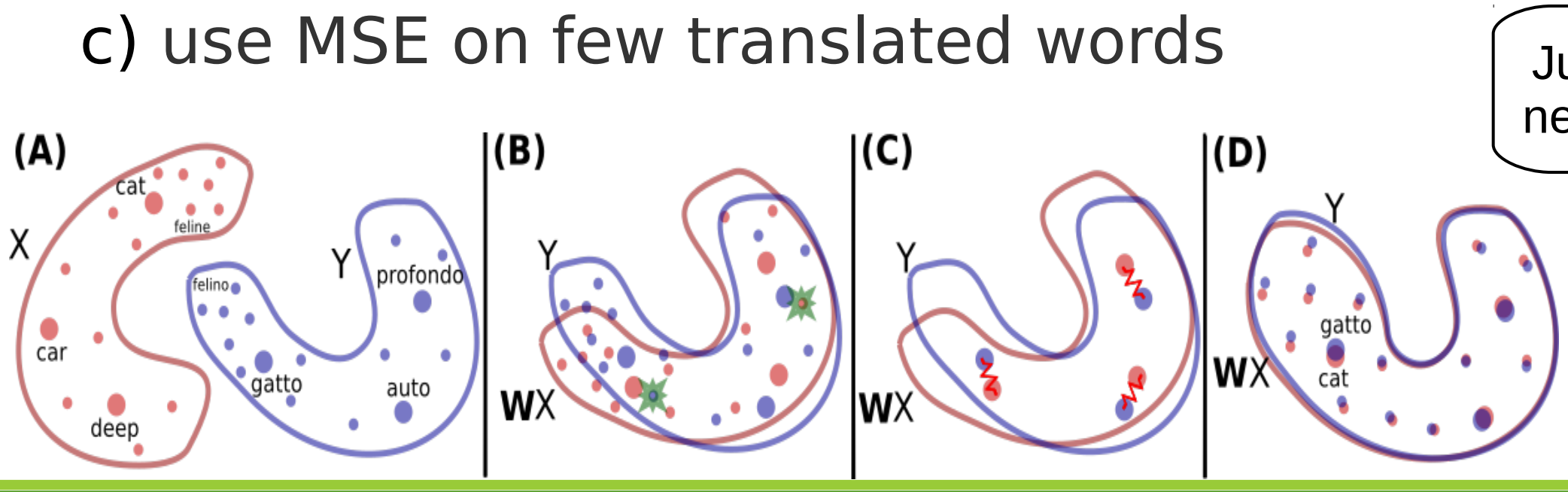
Source: https://arxiv.org/abs/1710.04087

Adversarial dictionary learning

a) learn monolingual embeddings (w2v/glove/fasttext)

b) map embeddings together with GAN loss

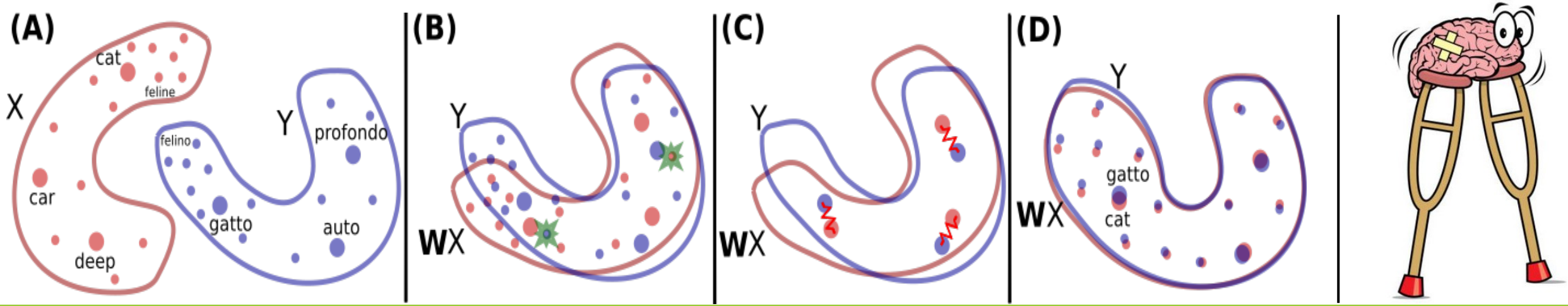
c) use MSE on few translated words



Source: https://arxiv.org/abs/1710.04087

Adversarial dictionary learning

to be continued...
in your homework assignment



Source: https://arxiv.org/abs/1710.04087

Adversarial domain adaptation

- Consider example:
  - Source domain: IMDB (labeled)

    Without a doubt one of the best and most influential movies of all time, the I-TITLE is the defining science fiction film of …

  - Target domain: Twitter (unlabeled)

    I cant srand that baby cuttin baboon butt face I-PER I-PER sucks!

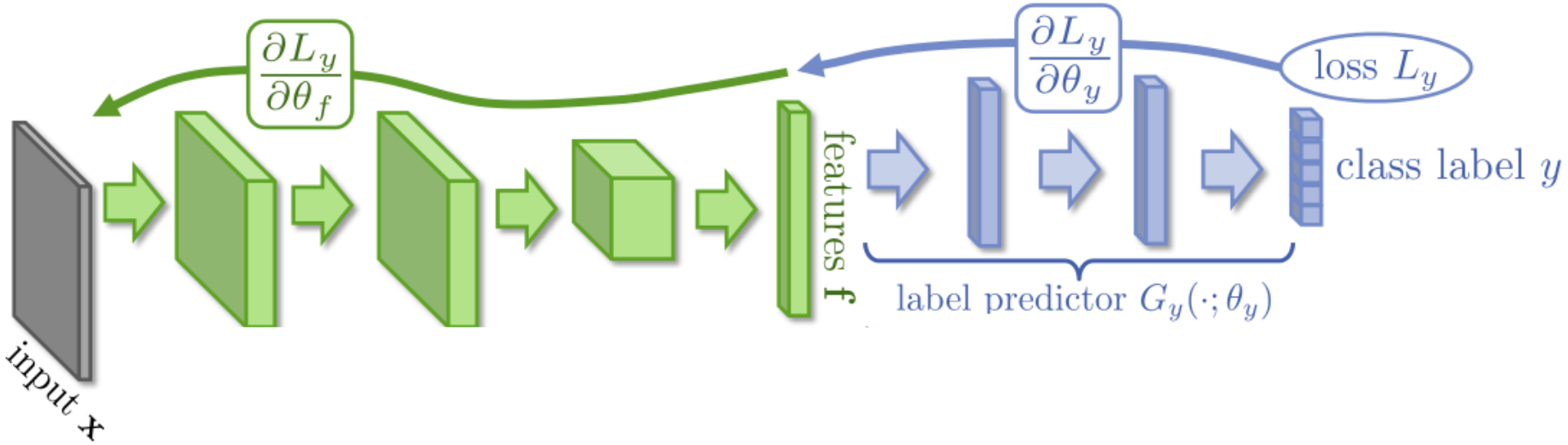
Adversarial domain adaptation

- Goal: learn sentiment classifier on target domain

Core idea: model should learn features that work **the same** on source and target domain

Source: https://arxiv.org/abs/1409.7495

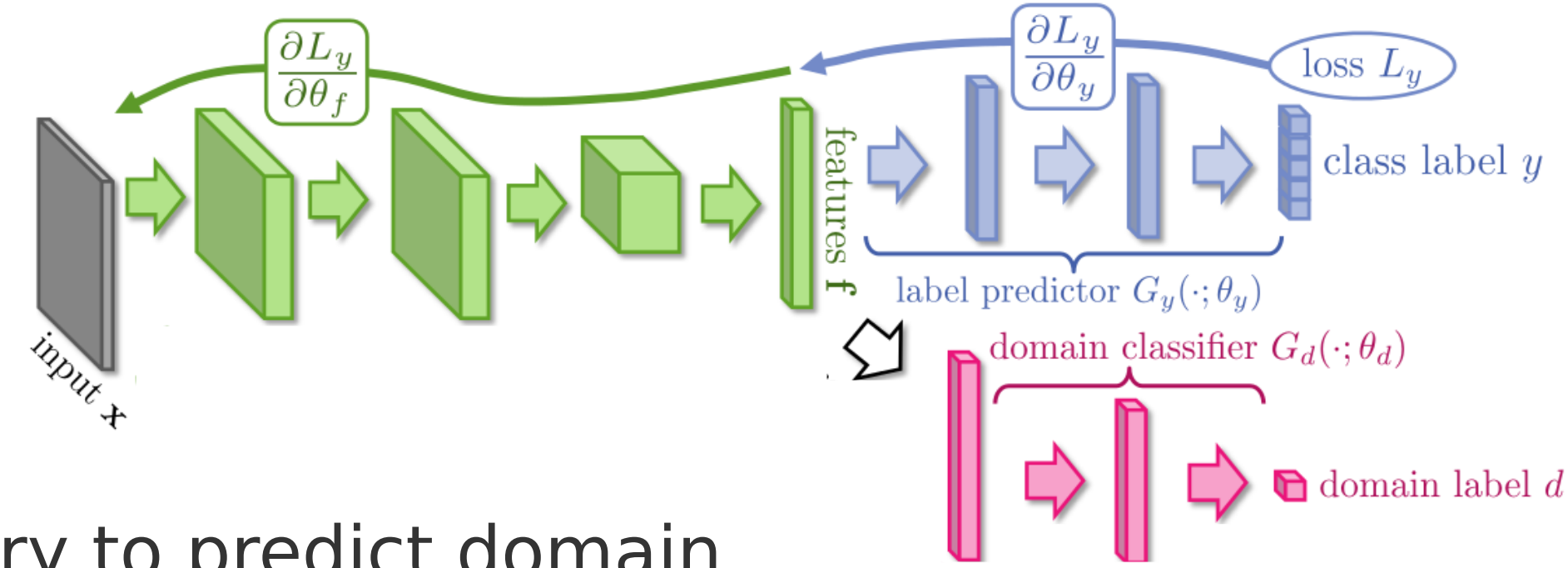
Adversarial domain adaptation



1) build “normal” model
cnn / *bi-lstm / attn*

Source: https://arxiv.org/abs/1409.7495

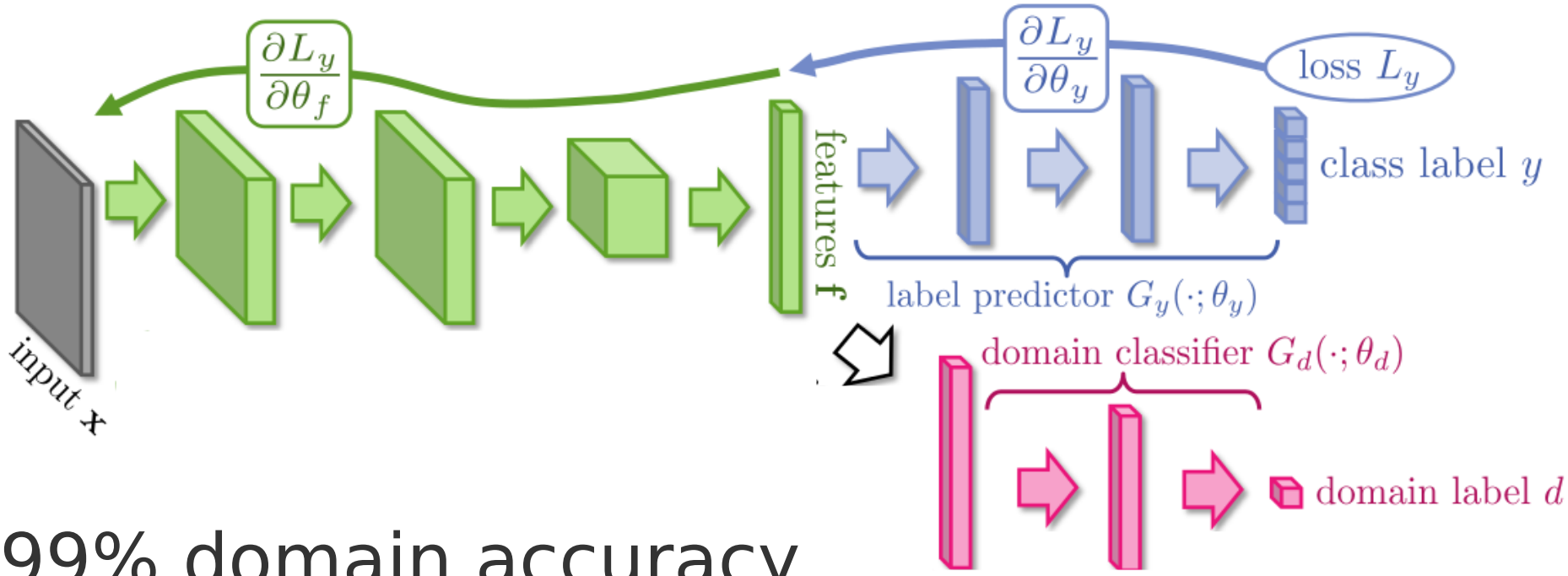
Adversarial domain adaptation



2) try to predict domain given hidden layer

Source: https://arxiv.org/abs/1409.7495

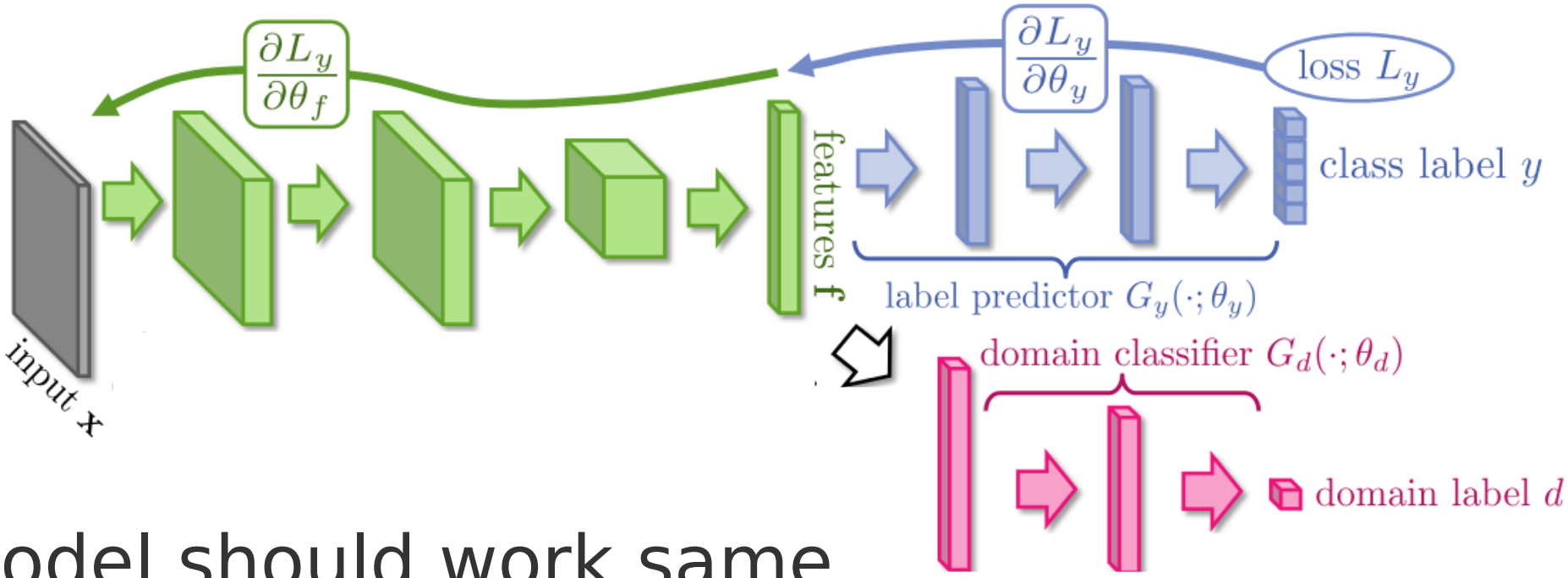
Adversarial domain adaptation



Q: Is 99% domain accuracy a good sign?

Source: https://arxiv.org/abs/1409.7495

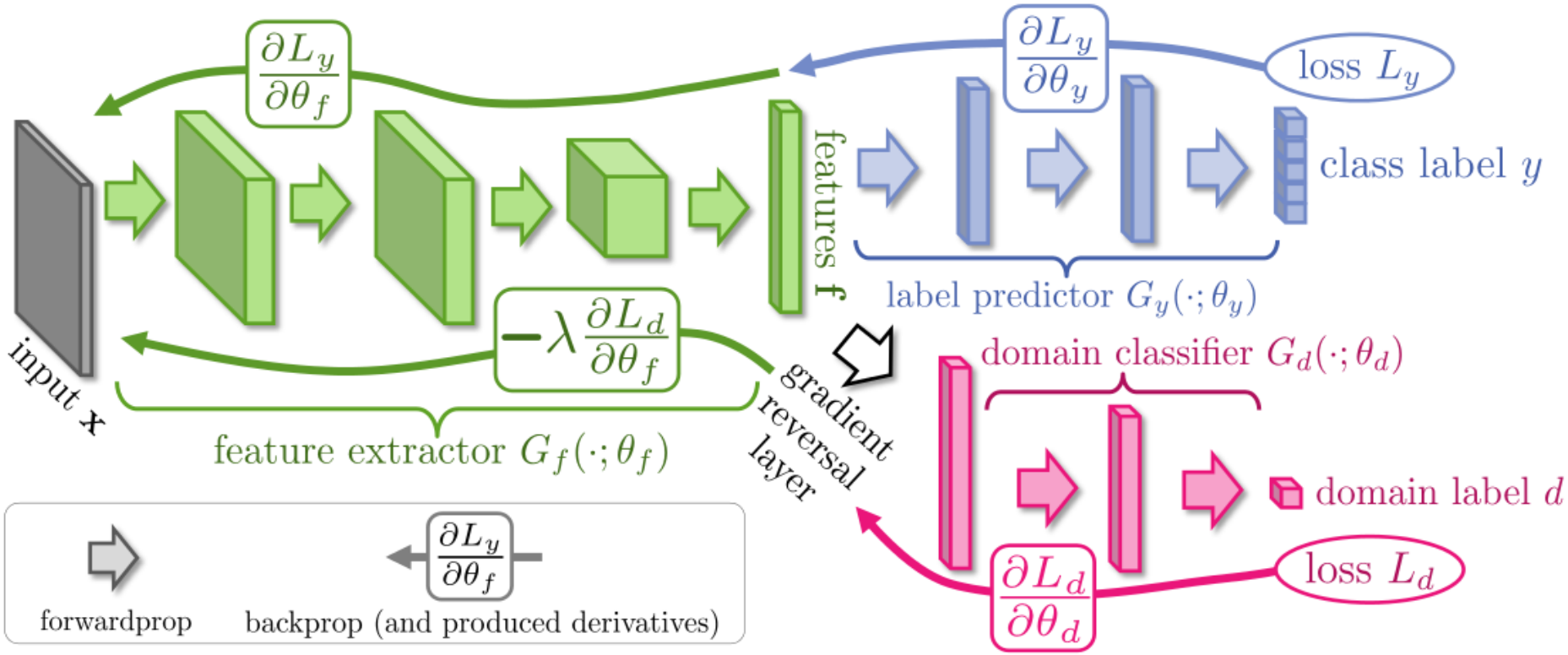
Adversarial domain adaptation



NO! model should work same on both domains

Source: https://arxiv.org/abs/1409.7495

Adversarial domain adaptation



Source: https://arxiv.org/abs/1409.7495

Adversarial domain adaptation

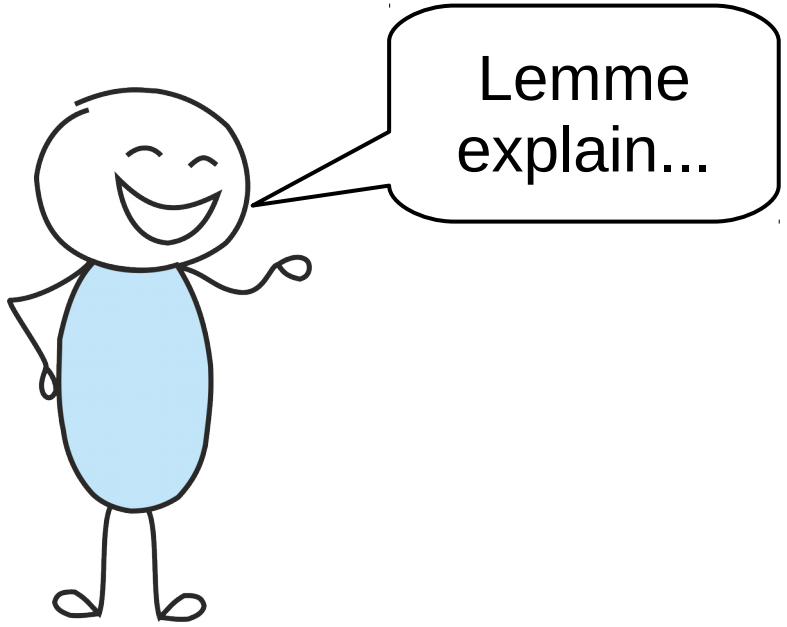
$$-\log P\left(real \middle| h\left(x_{real}\right)\right) - \log\left[1 - P\left(real \middle| h\left(x_{mc}\right)\right)\right] \rightarrow \min_{discriminator}$$

$$L_{classifier}\left(y_{mc}, y\left(h\left(x_{mc}\right)\right)\right) - \log P\left(real \middle| h\left(x_{mc}\right)\right) \rightarrow \min_{classifier}$$

Source: https://arxiv.org/abs/1409.7495

Adversarial seq2seq

- Idea: train seq2seq as generator in GAN
- Why?



Label bias

- Supervised seq2seq training (aka teacher-forcing)

$$P(y_{t+1}|x, y_{0:t}), \qquad y_{0:t} \sim reference$$

- Inference time

$$P(y_{t+1}|x, \hat{y}_{0:t}), \qquad \hat{y}_{0:t} \sim \textbf{???}$$

Label bias

- Supervised seq2seq training (aka teacher-forcing)

$$P(y_{t+1}|x, y_{0:t}), \qquad y_{0:t} \sim reference$$

- Inference time

$$P(y_{t+1}|x, \hat{y}_{0:t}), \qquad \hat{y}_{0:t} \sim model$$

Label bias

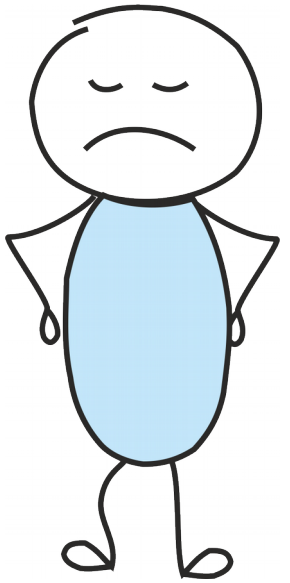
- Supervised seq2seq training (aka teacher-forcing)

$$P(y_{t+1}|x, y_{0:t}), \qquad y_{0:t} \sim reference$$

- Inference time

$$P(y_{t+1}|x, \hat{y}_{0:t}), \qquad \hat{y}_{0:t} \sim model$$

- Model can get unstable after first mistake



Multimodality problem

There's more then one correct translation.

Source: 在 找 给 家里 人 的 礼物 .

Versions:

```
i 'm searching for some gifts for my family.
i want to find something for my family as presents.
i 'm about to buy some presents for my family.
i 'd like to buy my family something as a gift.
i 'm looking for a present for my family.
```

sample from IWSLT 2009 Ch-En, http://bit.ly/2o404Tz

Multimodality problem

There's more then one correct translation.

you usually need just one

Source: 在 找 给 家里 人 的 礼物 .

Versions:

```
i 'm searching for some gifts for my family.
i want to find something for my family as presents.
i 'm about to buy some presents for my family.
i 'd like to buy my family something as a gift.
i 'm looking for a present for my family.
```

sample from IWSLT 2009 Ch-En, http://bit.ly/2o404Tz

Multimodality problem

Versions:	Model 1 **p(y|x)**	Model 2 **p(y|x)**
(version 1)	1e-2	0.99
(version 2)	2e-2	1e-100
(version 3)	1e-2	1e-100
(all rubbish)	0.96	0.01

incorrect translations

sample from IWSLT 2009 Ch-En, http://bit.ly/2o404Tz

Multimodality problem

Versions:	Model 1 **p(y|x)**	Model 2 **p(y|x)**
(version 1)	1e-2	0.99
(version 2)	2e-2	1e-100
(version 3)	1e-2	1e-100
(all rubbish)	0.96	0.01

Trivia: which model has better
Mean log p(y|x) ?

sample from IWSLT 2009 Ch-En, http://bit.ly/2o404Tz

Multimodality problem

Versions:	Model 1 **p(y|x)**	Model 2 **p(y|x)**
(version 1)	1e-2	0.99
(version 2)	2e-2	1e-100
(version 3)	1e-2	1e-100
(all rubbish)	0.96	0.01
	better llh 96% rubbish	worse llh 1% rubbish

sample from IWSLT 2009 Ch-En, http://bit.ly/2o404Tz

Adversarial seq2seq

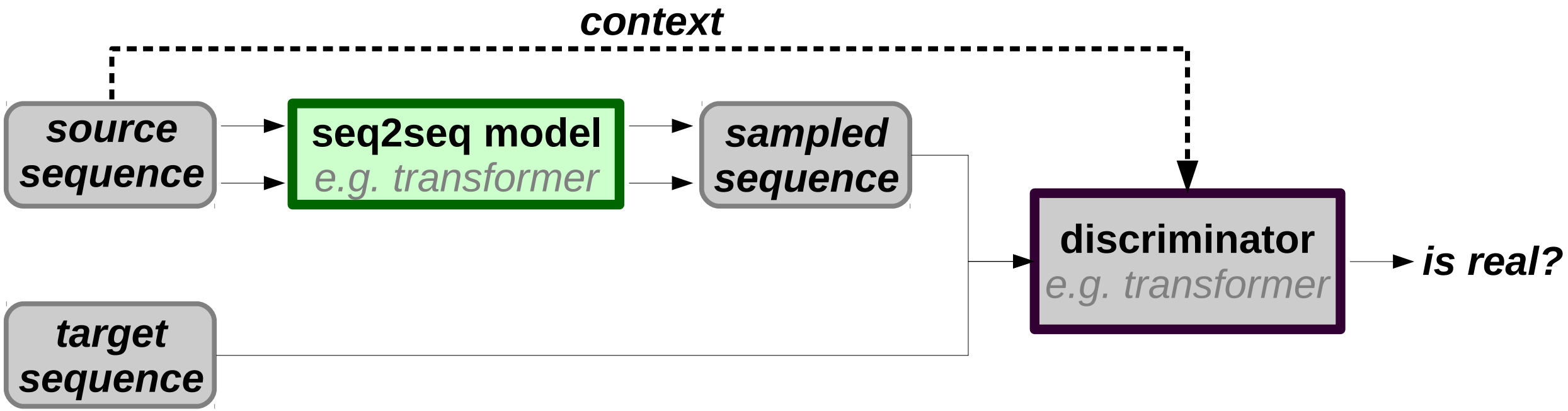
- Idea: train seq2seq as generator in GAN
- Why?
  - No label bias
  - Better loss

Adversarial seq2seq

- Sketch: conditional GAN

Adversarial seq2seq

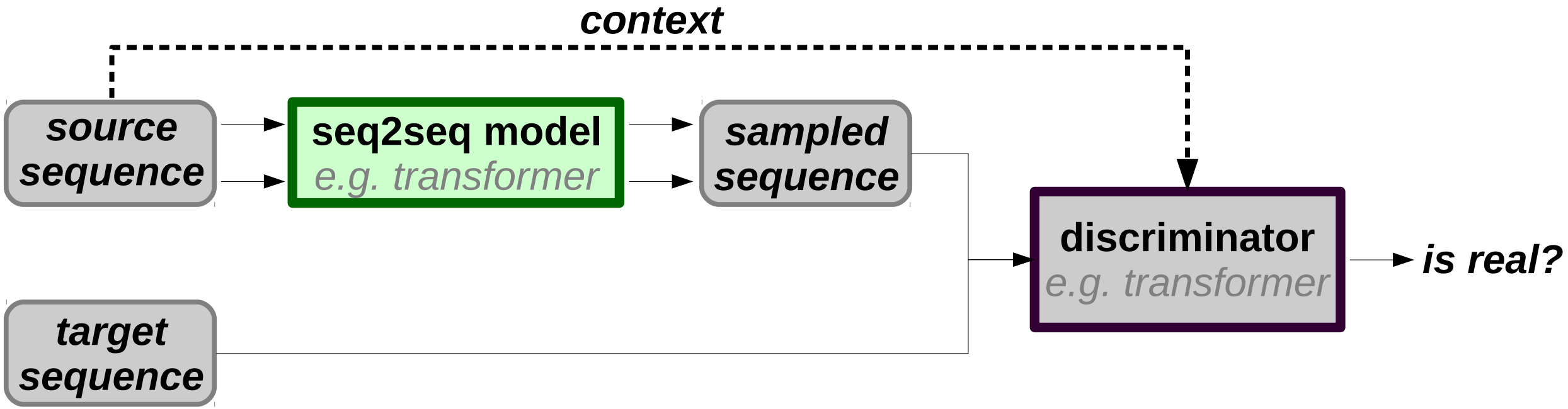
- Sketch: conditional GAN



Adversarial seq2seq

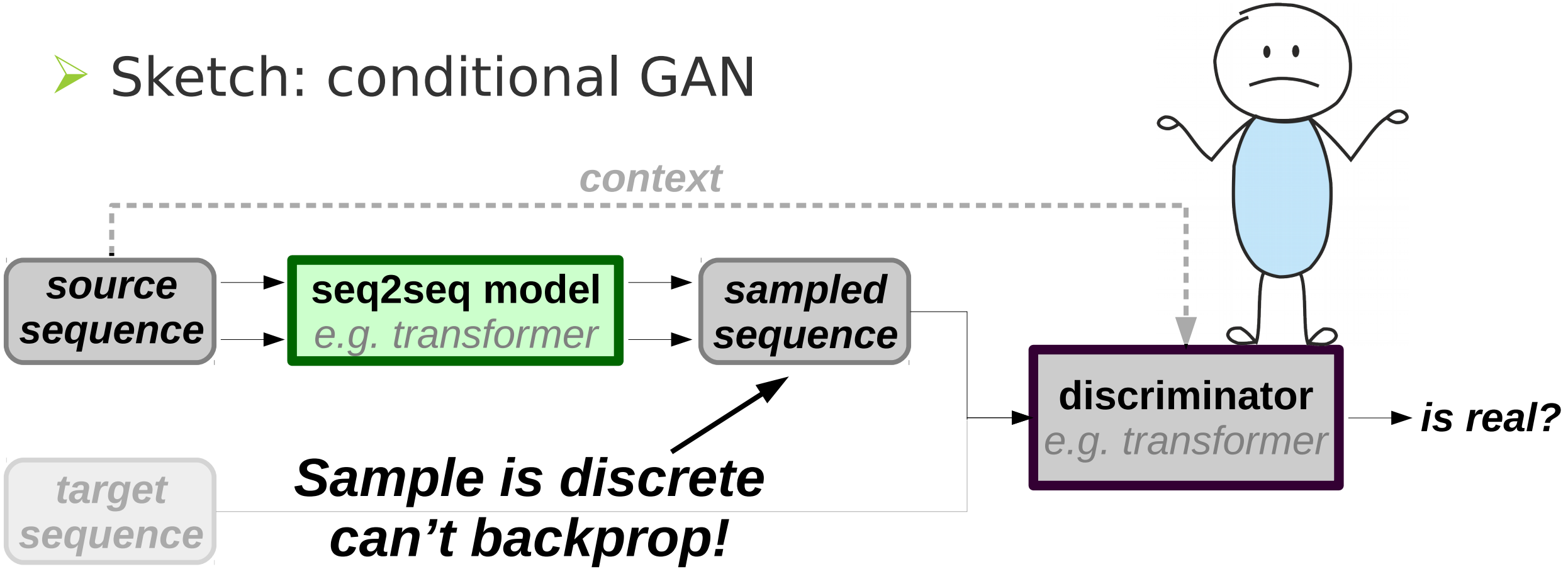
- Sketch: conditional GAN

Q: Any problems?



Adversarial seq2seq

- Sketch: conditional GAN



REINFORCE

- Generator objective

$$Loss_{gen} = -E_{\hat{y} \sim p(\hat{y}|x)} \log p(real|x, \hat{y}) = -\sum_{\hat{y}} p(\hat{y}|x) \cdot \log p(real|x, \hat{y})$$

- Gradient

$$\nabla Loss_{gen} = -\sum_{\hat{y}} \nabla p(\hat{y}|x) \cdot \log p(real|x, \hat{y})$$

REINFORCE

$$\nabla Loss_{gen} = -\sum_{\hat{y}} \nabla p(\hat{y}|x) \cdot \log p(real|x, \hat{y})$$

$$\nabla p(x) = p(x) \cdot \nabla \log p(x)$$

REINFORCE

$$\nabla Loss_{gen} = -\sum_{\hat{y}} \nabla p(\hat{y}|x) \cdot \log p(real|x, \hat{y})$$

$$\nabla p(x) = p(x) \cdot \nabla \log p(x)$$

$$\nabla Loss_{gen} = -\sum_{\hat{y}} p(\hat{y}|x) \nabla \log p(\hat{y}|x) \cdot \log p(real|x, \hat{y})$$

REINFORCE

$$\nabla Loss_{gen} = -\sum_{\hat{y}} \nabla p(\hat{y}|x) \cdot \log p(real|x, \hat{y})$$

$$\nabla p(x) = p(x) \cdot \nabla \log p(x)$$

$$\nabla Loss_{gen} = -E_{\hat{y} \sim p(\hat{y}|x)} \nabla \log p(\hat{y}|x) \cdot \log p(real|x, \hat{y})$$

Adversarial seq2seq

- Discriminator: train as usual

$$Loss_{disc} = -E_{x \sim data} \log p(real|x) \; - \; E_{z \sim p(z)} \log p(fake|gen(z))$$

- Generator: use REINFORCE

$$\nabla Loss_{gen} = -E_{\hat{y} \sim p(\hat{y}|x)} \nabla \log p(\hat{y}|x) \cdot \log p(real|x, \hat{y})$$

Adversarial seq2seq

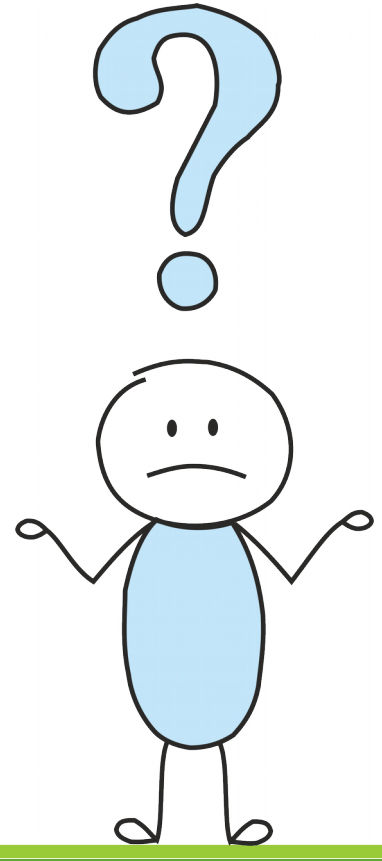
Model	Chinese-English				English-German
	NIST03	NIST04	NIST05	average	newstest2014
RNNSearch (Bahdanau et al., 2014)	33.93	35.67	32.24	33.94	21.2
Transformer (Vaswani et al., 2017)	42.23	42.17	41.02	41.80	27.30
RNNSearch+BR-CSGAN($\lambda = 1.0$)	35.21	36.51	33.45	35.05	22.1
RNNSearch+BR-CSGAN($\lambda = 0.7$)	35.97	37.32	34.03	**35.77**	**22.89**
RNNSearch+BR-CSGAN($\lambda = 0$)	34.57	35.93	33.07	34.52	21.75
Transformer+BR-CSGAN($\lambda = 1.0$)	42.67	42.79	41.54	42.30	27.75
Transformer+BR-CSGAN($\lambda = 0.8$)	43.01	42.96	41.86	**42.61**	**27.92**
Transformer+BR-CSGAN($\lambda = 0$)	42.41	42.74	41.29	42.14	27.49

Table 1: BLEU score on Chinese-English and English-German translation tasks. The hyper-parameter λ is selected according to the development set. For the Transformer, following (Vaswani et al., 2017), we report the result of a single model obtained by averaging the 5 checkpoints around the best model selected on the development set.

- https://arxiv.org/abs/1703.04887
- https://arxiv.org/abs/1704.06933

Generative Adversarial Nets

- Pros:
  - Great samples
  - Non-autoregressive
- Cons:
  - ???



Generative Adversarial Nets

- Pros:
  - Great samples
  - Non-autoregressive
- Cons:
  - Unstable training
  - Mode collapse
  - No explicit probability

Generative Adversarial Nets

A few more GAN-related things we left uncovered

- Learning from non-parallel data with Cycle-GANs
  https://arxiv.org/abs/1703.10593
- Mitigating label bias with Professor Forcing
  https://arxiv.org/abs/1610.09038

Learning distributions

Target distribution $p(x)$; Generated distribution $\hat{p}(x)$

Goal: make $\hat{p}(x)$ match $p(x)$

- Max Likelihood Fit
- Generative Adversarial Networks
- **Variational Autoencoders**

Learning distributions

Target distribution $p(x)$; Generated distribution $\hat{p}(x)$

Goal: make $\hat{p}(x)$ match $p(x)$

- Max Likelih
- Generative
- **Variational Autoencoders**

Learning distributions

Target distribution $p(x)$; Generated distribution $\hat{p}(x)$

Goal: make $\hat{p}(x)$ match $p(x)$

- Max Likelihood Fit
- Generative Adversarial Networks
- **Variational Autoencoders**

Variational Autoencoders

Disclaimer: this is an extremely practial view of autoencoders

We cover the **how to**s but not the **why**s

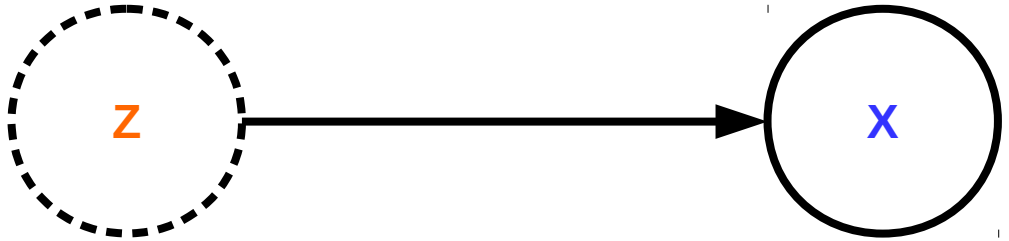
Wanna know more?

kvfrans.com/variational-autoencoders-explained

or enroll for «Bayesian deep learning» at YSDA

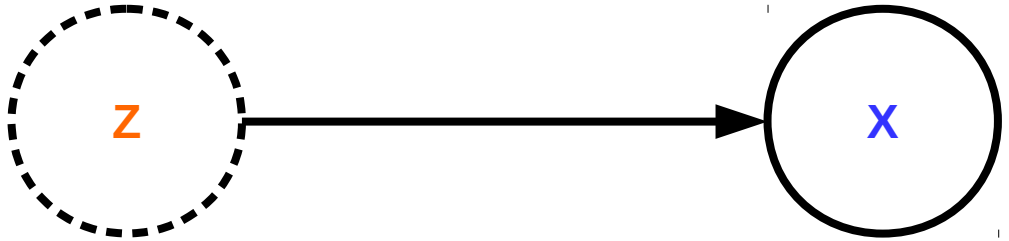
Variational Autoencoders

- Assume there is a hidden variable **z** that determines **x**



Variational Autoencoders

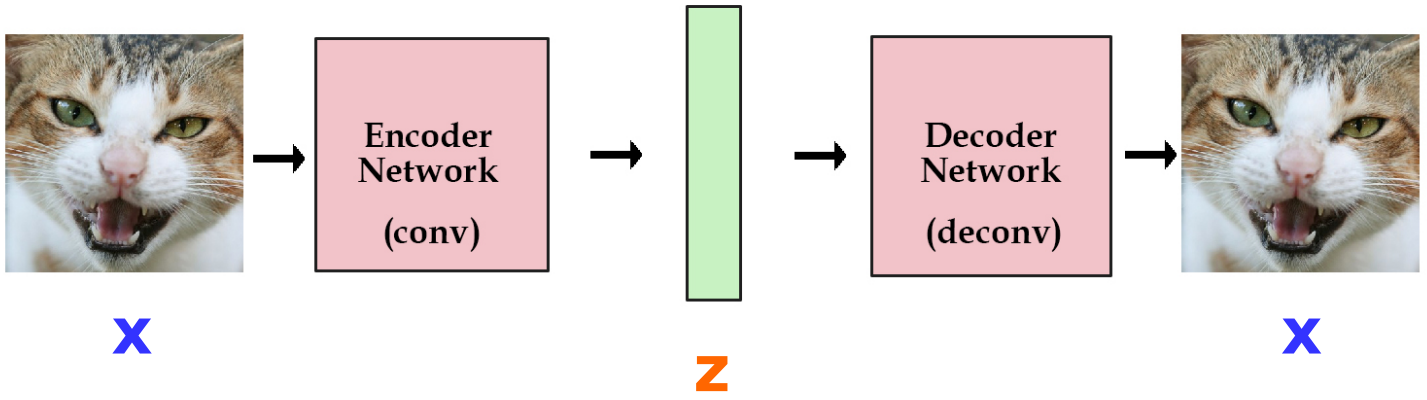
- Assume there is a hidden variable **z** that determines **x**



- What do we want?
  - p(**x** | **z**) — generate data given «thought» vector
  - q(**z** | **x**) — infer what thoughts was **x** generated **from**

Variational Autoencoders

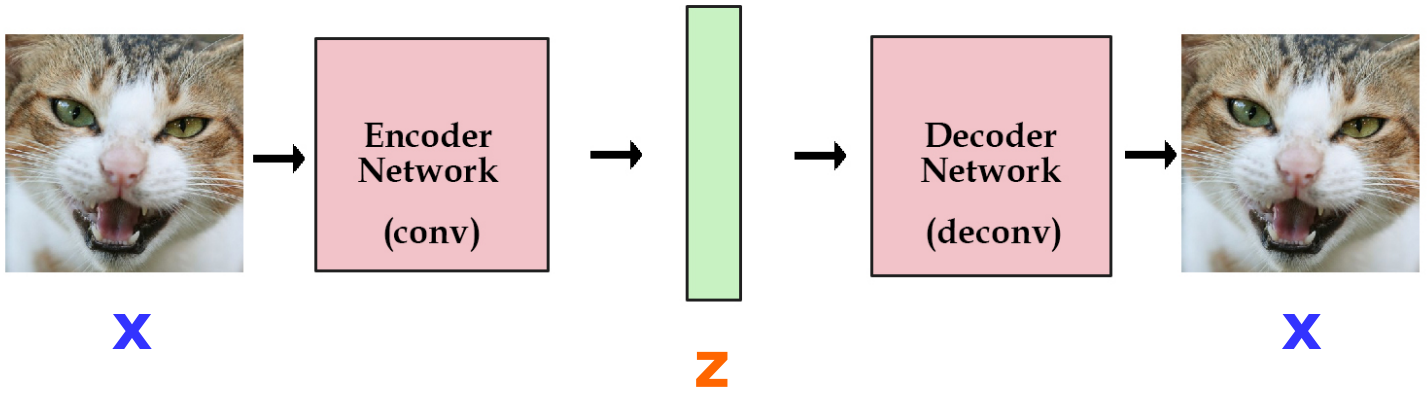
- Learn both q(**z** | **x**) and p(**x** | **z**)



Variational Autoencoders

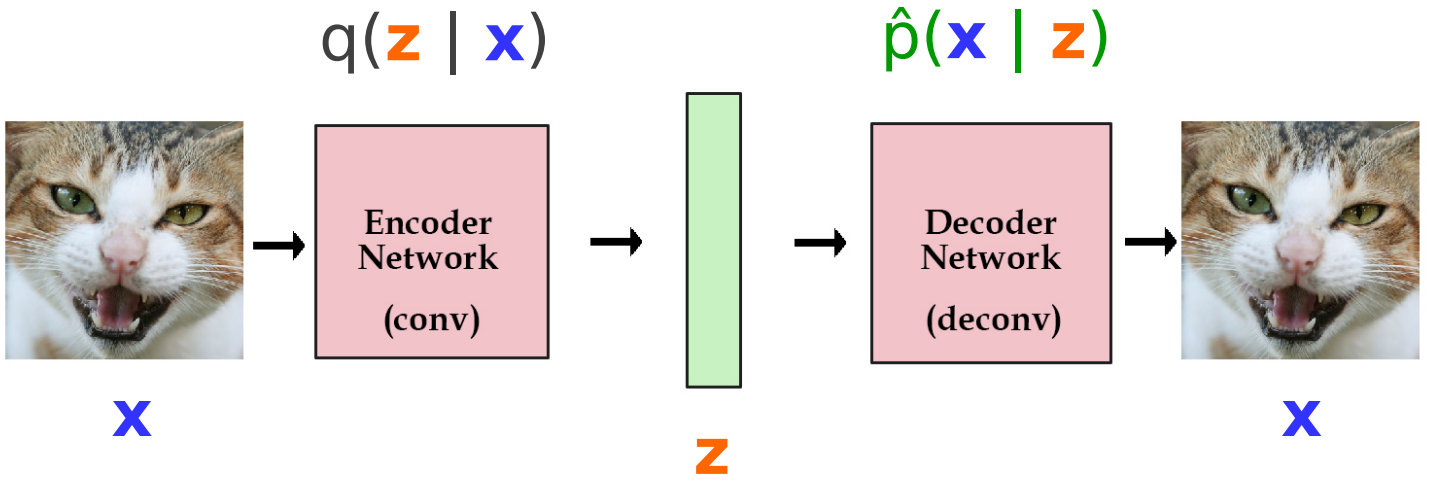
- Learn both q(**z** | **x**) and p(**x** | **z**)

Q: find **p̂** and **q** on the scheme



Variational Autoencoders

- Learn both q(**z** | **x**) and p(**x** | **z**)



Variational Autoencoders

VAE: example

normal **z**

(vector)

normal **x**

(each pixel)

VAE: example

normal **z**

$$q_\varphi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\mu_\varphi(\boldsymbol{x}), \sigma_\varphi(\boldsymbol{x}))$$

normal **x**

$$p_\theta(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\mu_\theta(\boldsymbol{z}), \sigma_\theta(\boldsymbol{z}))$$

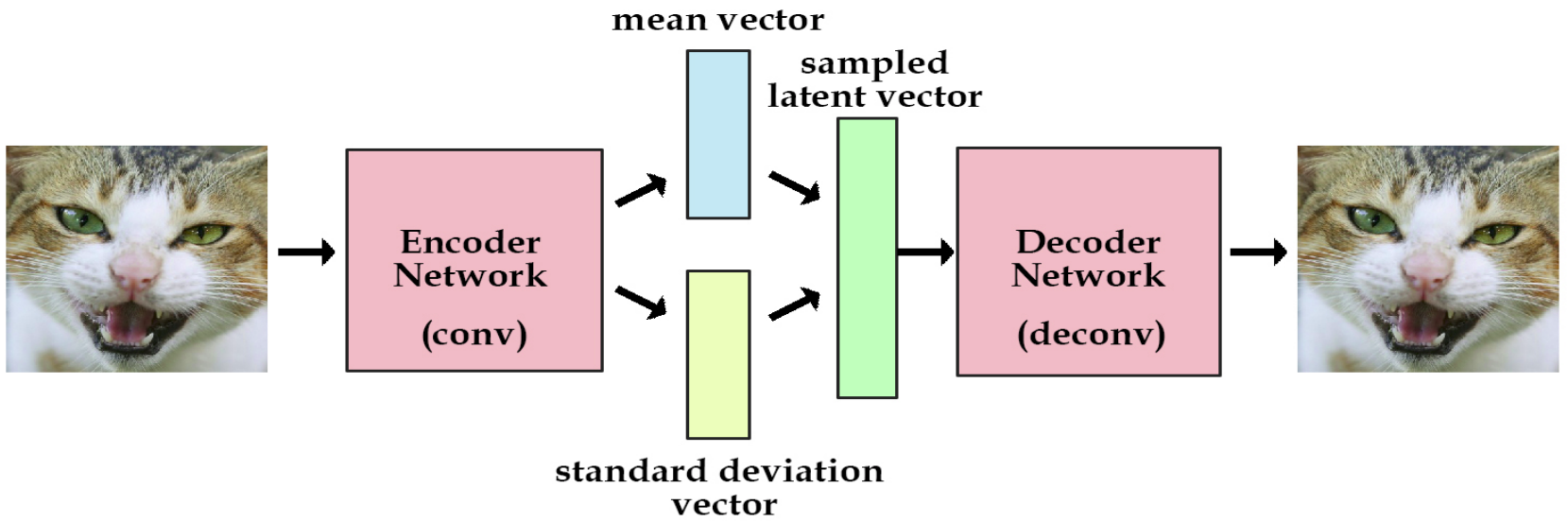
VAE: example

normal **z**

$$q_\varphi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\mu_\varphi(\boldsymbol{x}), \sigma_\varphi(\boldsymbol{x}))$$

normal **x**

$$p_\theta(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\mu_\theta(\boldsymbol{z}), \sigma_\theta(\boldsymbol{z}))$$



VAE: example

normal **z**

$$q_\varphi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\mu_\varphi(\boldsymbol{x}), \sigma_\varphi(\boldsymbol{x}))$$

normal **x**

$$p_\theta(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\mu_\theta(\boldsymbol{z}), \sigma_\theta(\boldsymbol{z}))$$

Objective:

- encode **x** into **z** and restore original **x**
- overall **z** should be approximately N(0,1)

VAE: example

normal **z**

$$q_\varphi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\mu_\varphi(\boldsymbol{x}), \sigma_\varphi(\boldsymbol{x}))$$

normal **x**

$$p_\theta(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\mu_\theta(\boldsymbol{z}), \sigma_\theta(\boldsymbol{z}))$$

Objective:

- encode **x** into **z** and restore original **x**
- overall **z** should be approximately N(0,1)

or any other $p_{prior}(z)$
used for sampling

VAE: example

normal **z**

$$q_\varphi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\mu_\varphi(\boldsymbol{x}), \sigma_\varphi(\boldsymbol{x}))$$

normal **x**

$$p_\theta(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\mu_\theta(\boldsymbol{z}), \sigma_\theta(\boldsymbol{z}))$$

Objective:

$$-E_{z\sim q(z|x)} \log p(x|z) + KL(q_\theta(z|x)||p_{prior}(z)) \rightarrow min$$

VAE: example

normal **z**

$$q_\varphi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\mu_\varphi(\boldsymbol{x}), \sigma_\varphi(\boldsymbol{x}))$$

normal **x**

$$p_\theta(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\mu_\theta(\boldsymbol{z}), \sigma_\theta(\boldsymbol{z}))$$

Objective:

$$-E_{z\sim q(z|x)} \log p(x|z) + KL(q_\theta(z|x)||N(0,1)) \rightarrow min$$

VAE: example

normal **z**

$$q_\varphi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\mu_\varphi(\boldsymbol{x}), \sigma_\varphi(\boldsymbol{x}))$$

normal **x**

$$p_\theta(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\mu_\theta(\boldsymbol{z}), \sigma_\theta(\boldsymbol{z}))$$

Objective:

$$-E_{z\sim q(z|x)}\log p(x|z) + KL(q_\theta(z|x)||N(0,1)) \rightarrow min$$

- encode **x** into **z** and restore original **x** **(MSE)**

VAE: example

normal **z**

$$q_\varphi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\mu_\varphi(\boldsymbol{x}), \sigma_\varphi(\boldsymbol{x}))$$

normal **x**

$$p_\theta(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\mu_\theta(\boldsymbol{z}), \sigma_\theta(\boldsymbol{z}))$$

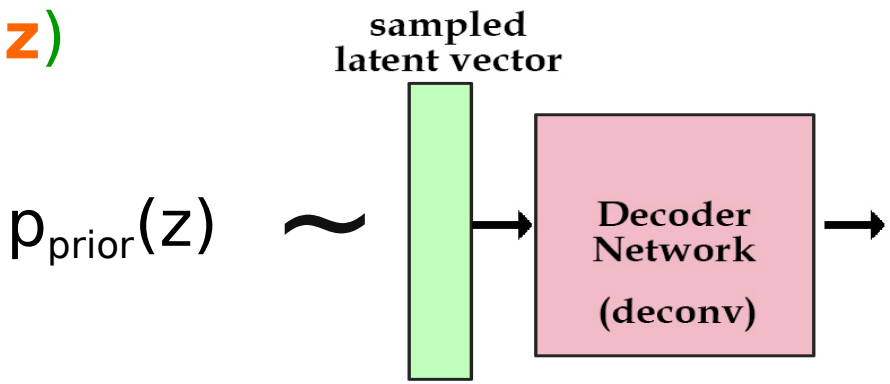
Objective:

$$-E_{z\sim q(z|x)} \log p(x|z) + KL(q_\theta(z|x)||N(0,1)) \rightarrow min$$

- overall **z** should be approximately N(0,1)
  closed form *in terms of μ(x), σ(x)*

Generating with VAE

- Sample $z \sim p_{prior}(z)$
- Generate $\mathbf{x} \sim \hat{p}(\mathbf{x} \mid \mathbf{z})$



Generating with VAE

- Sample $z \sim p_{prior}(z)$
- Generate **x** ~ $\hat{p}$(**x** | **z**)

Morphing:

- change **z** in small increments
- small semantic changes in **x**

Variational Autoencoders

- Pros:
  - Training is easy
  - No fixed order

- Cons:
  - Lower bound on probability
  - Independence Assumptions

Variational Autoencoders

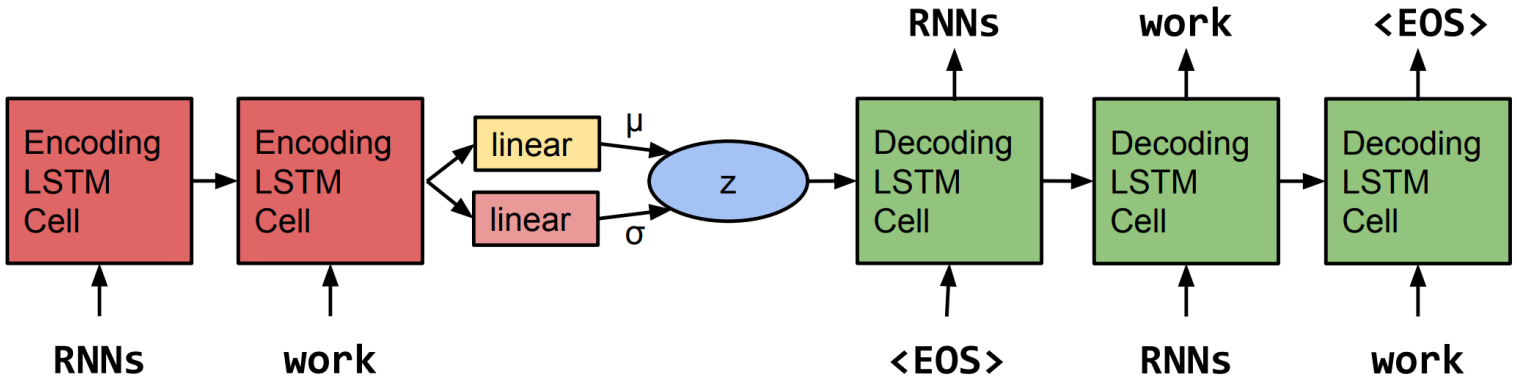
- Pros:
  - Training is easy — *encode, decode, backprop*
  - No fixed order — *quick inference for any structure*

- Cons:
  - Lower bound on probability — *perplexity is worse than MLE*
  - Independence Assumptions — *hence «blurry» outputs*

read more: https://arxiv.org/abs/1711.00464

VAE for NLP tasks

- VAE Language Model https://arxiv.org/abs/1511.06349



- Features: phrase morphing, ~~+9000% approval among bayesians~~

read more: https://bit.ly/2Q1Gsy3 (by iconix)

VAE for NLP tasks

- VAE prototype editor, https://arxiv.org/abs/1709.08878

  Applies learned «edit vector» to change x_1 into x_2

  Edits turn out to be interpretable by humans

	Example 1	Example 2
Context	he comes home tired and happy .	i went with a larger group to <person> 's .
Edit	+ was - is	+ good - better
Result	= he came home happy and tired .	= i went to <person> 's with a large group .

VAE for NLP tasks

- VAE prototype editor, https://arxiv.org/abs/1709.08878

Model	Perplexity (Yelp)	Perplexity (BILLIONWORD)
KN5	56.546	78.361
KN5+MEMORIZATION	55.184	73.468
NLM	39.026	55.146
NLM+MEMORIZATION	38.086	50.969
NLM+KN5	37.312	**47.472**
NEURALEDITOR($\kappa = 0$)	**26.87**	48.755
NEURALEDITOR($\kappa = 25$)	27.41	48.921

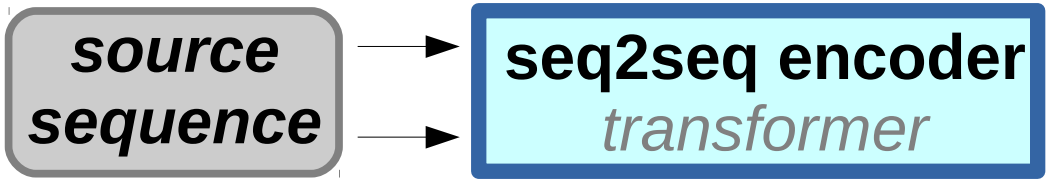
VAE for NLP tasks

- VAE for fast inference in seq2seq https://arxiv.org/abs/1803.03382

- **Task:** neural machine translation (sequence-to-sequence)
- **Sketch:** use conditional independence to speed-up inference

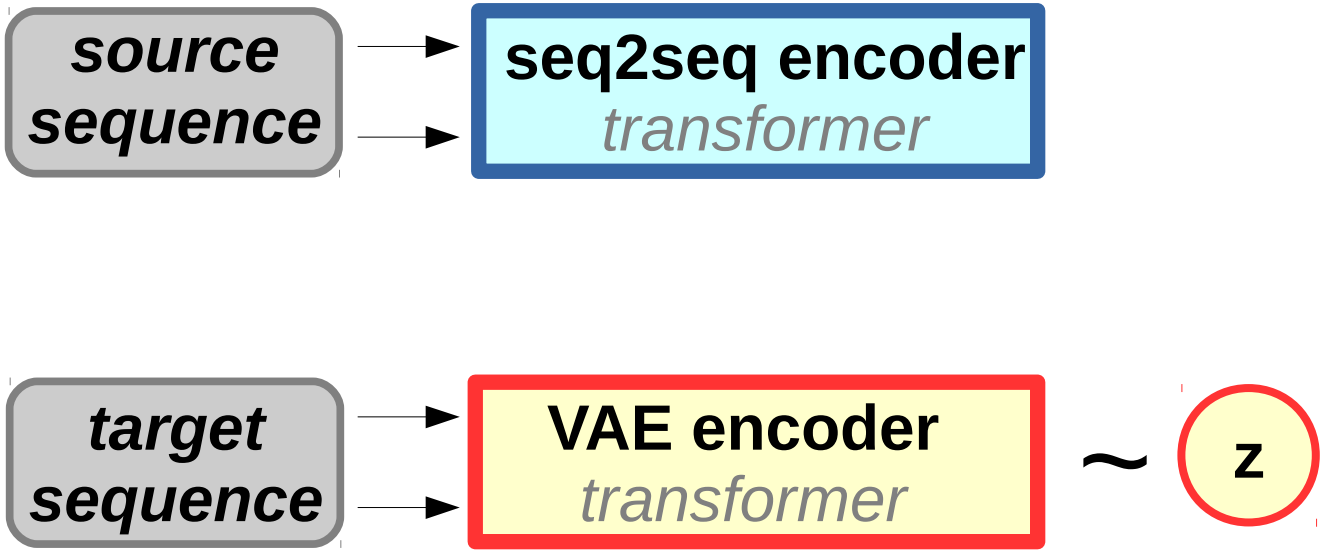
VAE for NLP tasks

- VAE for fast inference in seq2seq https://arxiv.org/abs/1803.03382



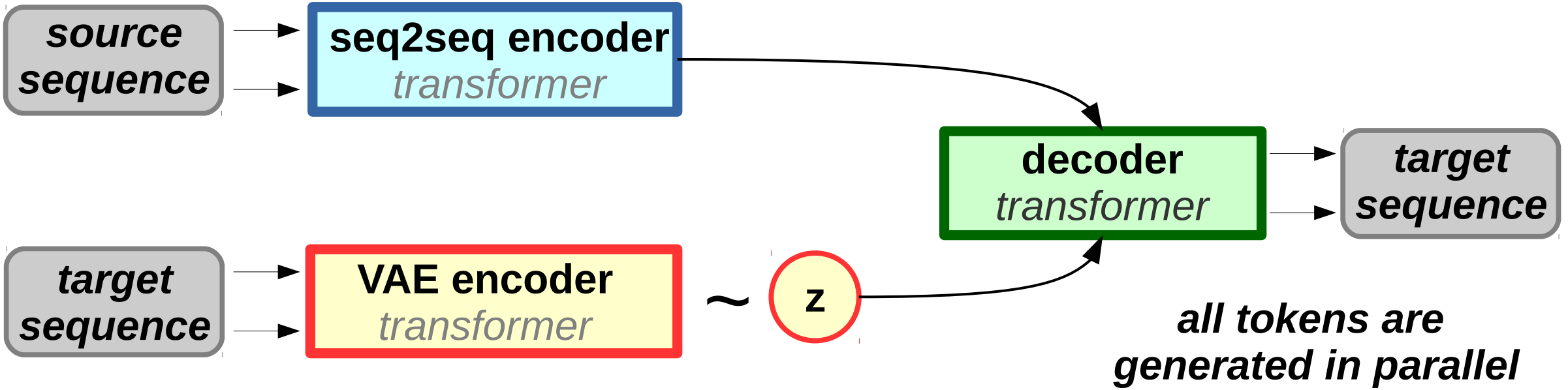
VAE for NLP tasks

- VAE for fast inference in seq2seq https://arxiv.org/abs/1803.03382



VAE for NLP tasks

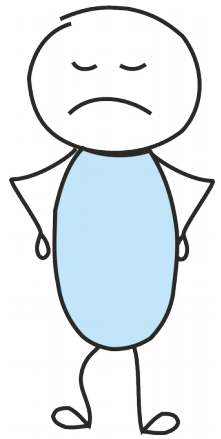
- VAE for fast inference in seq2seq https://arxiv.org/abs/1803.03382



VAE for NLP tasks

- VAE for fast inference in seq2seq https://arxiv.org/abs/1803.03382

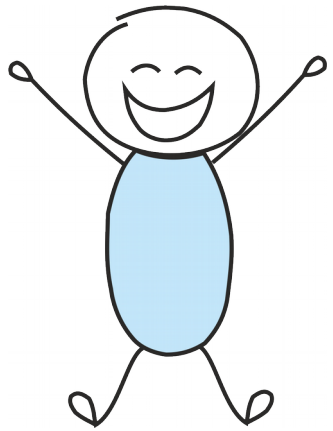
Model	BLEU
Baseline Transformer [1]	27.3
Baseline Transformer [2]	23.5
Baseline Transformer [2] (no beam-search)	22.7
NAT+FT (no NPD) [2]	17.7
LT without rescoring $\left(\frac{n}{m} = 8\right)$	19.8
NAT+FT (NPD rescoring 10) [2]	18.7
LT rescornig top-10 $\left(\frac{n}{m} = 8\right)$	21.0
NAT+FT (NPD rescoring 100) [2]	19.2
LT rescornig top-100 $\left(\frac{n}{m} = 8\right)$	22.5



VAE for NLP tasks

- VAE for fast inference in seq2seq https://arxiv.org/abs/1803.03382

Model	BLEU	Latency	
		$b = 1$	$b = 64$
Baseline (no beam-search)	22.7	408 ms	-
NAT	17.7	39 ms	-
NAT+NPD=10	18.7	79 ms	-
NAT+NPD=100	19.2	257 ms	-
LT, Improved Semhash	19.8	105 ms	8 ms
LT, VQ-VAE	2.78	148 ms	7 ms
LT, s-DVQ	19.7	177 ms	7 ms
LT, p-DVQ	19.8	182 ms	8 ms



Learning distributions

Target distribution $p(x)$; Generated distribution $\hat{p}(x)$

Goal: make $\hat{p}(x)$ match $p(x)$

- Max Likelihood Fit
- Generative Adversarial Networks
- Variational Autoencoders

Learning distributions

Target distribution $p(x)$; Generated distribution $\hat{p}(x)$

Goal: make $\hat{p}(x)$ match $p(x)$

- Max Likelihood Fit

  \+ **???** - **???**

- Generative Adversarial Networks

- Variational Autoencoders

Learning distributions

Target distribution $p(x)$; Generated distribution $\hat{p}(x)$

Goal: make $\hat{p}(x)$ match $p(x)$

- Max Likelihood Fit

  + Explicit, Stable - Slow inference, requires order

- Generative Adversarial Networks

  + **???** - **???**

- Variational Autoencoders

Learning distributions

Target distribution $p(x)$; Generated distribution $\hat{p}(x)$

Goal: make $\hat{p}(x)$ match $p(x)$

- Max Likelihood Fit

  + Explicit, Stable - Slow inference, requires order

- Generative Adversarial Networks

  + Great samples, Fast - Implicit, Unstable training

- Variational Autoencoders

  + **???** - **???**

Learning distributions

Target distribution $p(x)$; Generated distribution $\hat{p}(x)$

Goal: make $\hat{p}(x)$ match $p(x)$

- Max Likelihood Fit

  + Explicit, Stable - Slow inference, requires order

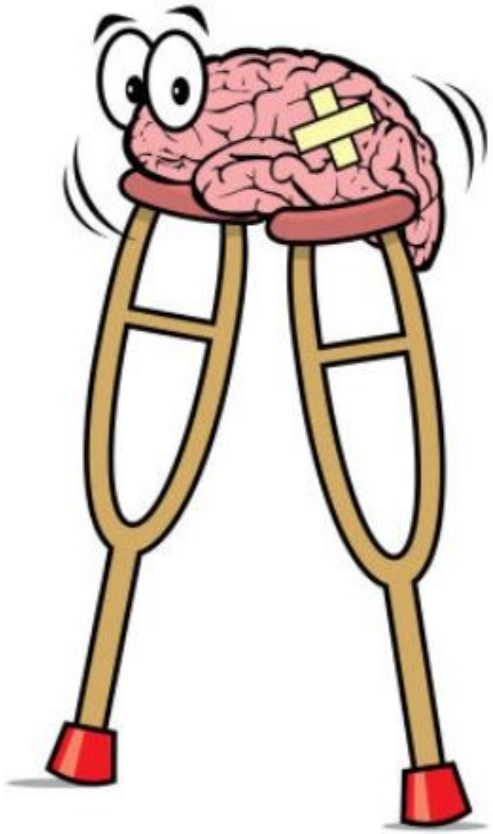
- Generative Adversarial Networks

  + Great samples, Fast - Implicit, Unstable training

- Variational Autoencoders

  + Explicit, Stable, Fast - «blurry» samples

Hack of the day



Hack of the day

~~Avoid VAE and GAN at all cost!~~

https://github.com/soumith/ganhacks

Hack of the day

Seq2seq model can be trained as:

- Supervised model (crossentropy)
- GAN generator
- RL agent
- VAE decoder

Each option has strengths and drawbacks

Hack of the day

Seq2seq model can be trained as:

- Supervised model — simple | label bias, etc.
- GAN generator — great samples | bad start, unstable
- RL agent — direct optimization | bad start, slow
- VAE decoder — continuous z | blurry samples

Q: Any ideas on how to dodge the drawbacks?

Hack of the day

Seq2seq model can be trained as:

- Supervised model — simple | label bias, etc.
- GAN generator — great samples | bad start, unstable
- RL agent — direct optimization | bad start, slow
- VAE decoder — continuous z | blurry samples

Can you combine supervised and GAN?

Hack of the day

Some combinations

- Start with supervised, fine-tune with GAN/RL
- VAE as conditioinal GAN (VAE-GAN)
- Mix supervised and GAN/RL losses
- You name it!

thanks to lena-voita@, bkovarsky@, norpadon@

EOS