

Introduction of this course

李宏毅

Hung-yi Lee

Welcome our TAs

TA 信箱 : ntu.mldsta@gmail.com

TAs



劉浩然



劉記良

TAs



孫凡耕



吳思霖

TAs



宋易霖



賴至得

TAs



陳力維



陳佳佑

TAs



陳冠宇



劉達融 (大助教)

What are we
going to learn?

本課程內容和《機器學習》沒有重疊

課程名稱解釋

機器學習
及其深層與結構化

Machine Learning
and having it Deep and Structured



Method

Deep Learning 可以解決一切 ...

遇到問題，用 c4 就對了！

用 deep learning “硬 train 一發”

萬事皆可 train



Practice of Deep Learning

- Previous machine learning developers
 - Carefully design your algorithm
 - Theoretically know its performance
- Deep learning
 - Try first
 - Many results contradict our intuition
 - Find some reasons to explain what we observed
 - More like chemistry
 - Or even worse



Practice of Deep Learning

- Even a simple model can be hard to train

Practice of Deep Learning

- Interesting facts

From: Boris
To: Ali

"On Friday, someone on another team changed the default rounding mode of some Tensorflow internals (from "truncate toward 0" to "round to even").

Our training broke. Our error rate went from <25% error to ~99.97% error (on a standard 0-1 binary loss)."

Ali Rahimi, Test of Time Award, NIPS 2017

Theory of Deep Learning

Theory 1:
Expressiveness

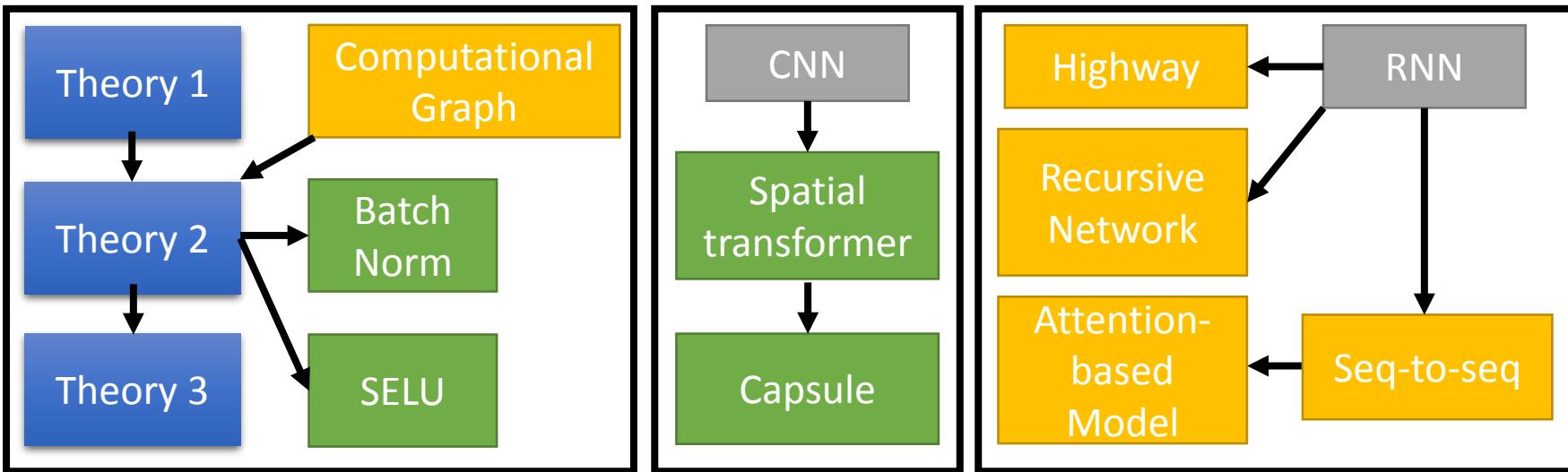
- A network structure defines a function set
- Is deep better than shallow?

Theory 2:
Optimization

- How can we optimize by gradient descent?
- There are local minima

Theory 3:
Generalization

- Why deep network does not overfit?
- Although it can



課程名稱解釋

機器學習
及其深層與結構化

Machine Learning
and having it Deep and Structured



Task

Structured Learning

Machine learning is to find a function f

$$f : X \rightarrow Y$$

Regression: output a scalar

Classification: output a “class” (one-hot vector)

1	0	0
---	---	---

Class 1

0	1	0
---	---	---

Class 2

0	0	1
---	---	---

Class 3

Structured Learning/Prediction: output a sequence, a matrix, a graph, a tree

Output is composed of components with dependency

Output Sequence

$$f : X \rightarrow Y$$

Machine Translation

X ：“機器學習及其深層與
結構化”
(sentence of language 1)

Y ：“Machine learning and
having it deep and structured”
(sentence of language 2)

Speech Recognition

X ：
(speech)

Y ：感謝大家來上課”
(transcription)

Chat-bot

X ：“How are you?”
(what a user says)

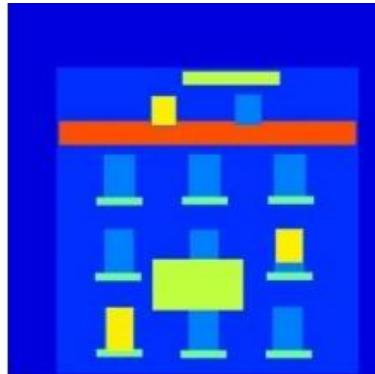
Y ：“I'm fine.”
(response of machine)

Output Matrix

$$f : X \rightarrow Y$$

Image to Image

$X :$



$Y :$



Colorization:



Ref: <https://arxiv.org/pdf/1611.07004v1.pdf>

Text to Image

$X :$ “this white and yellow flower
have thin white petals and a
round yellow stamen”

$Y :$



ref: <https://arxiv.org/pdf/1605.05396.pdf>

Reinforcement Learning



Action:
“right”



Action:
“fire”

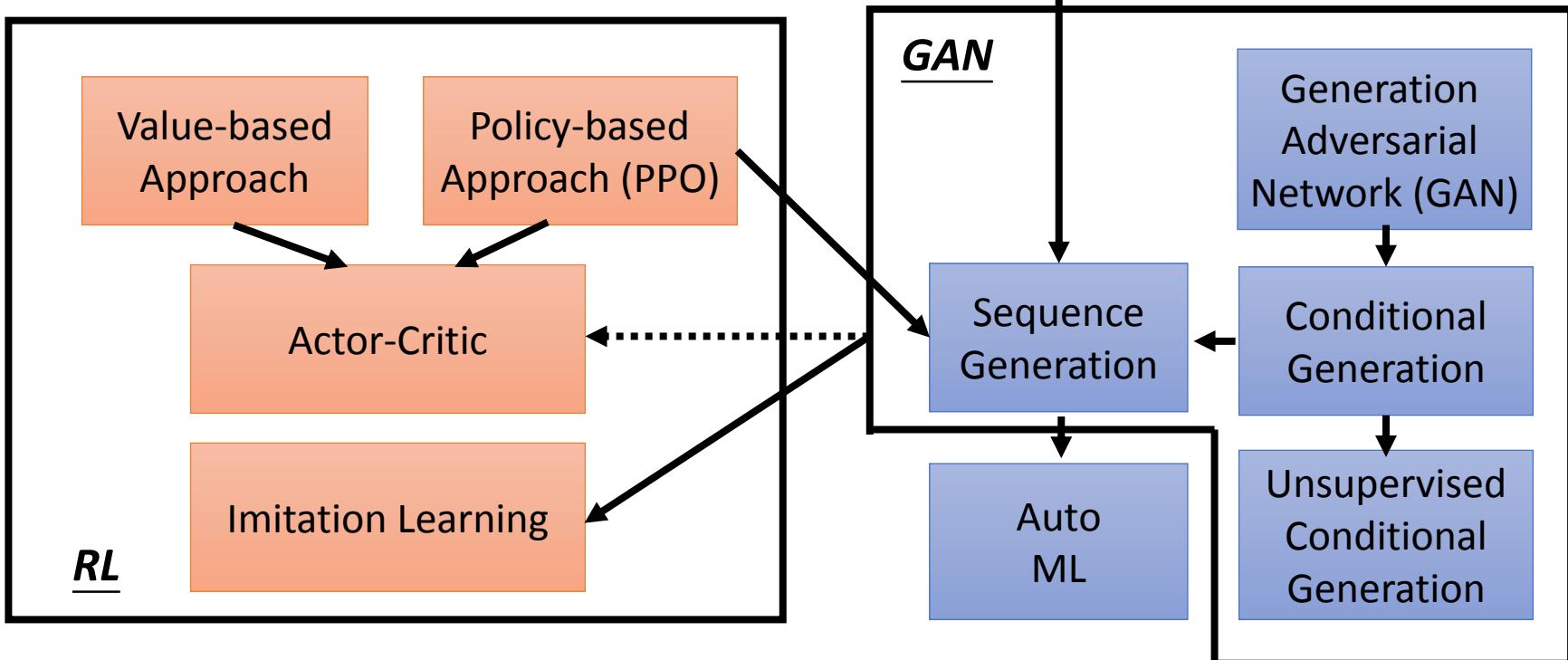
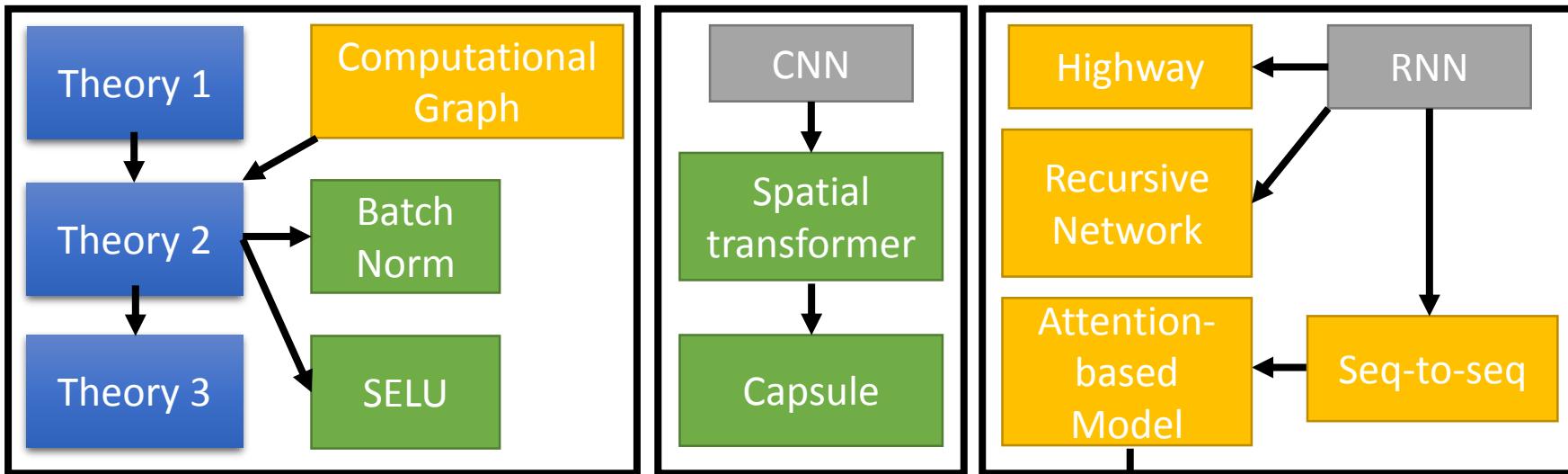


Action:
“left”

A sequence of decisions

Regression, Classification



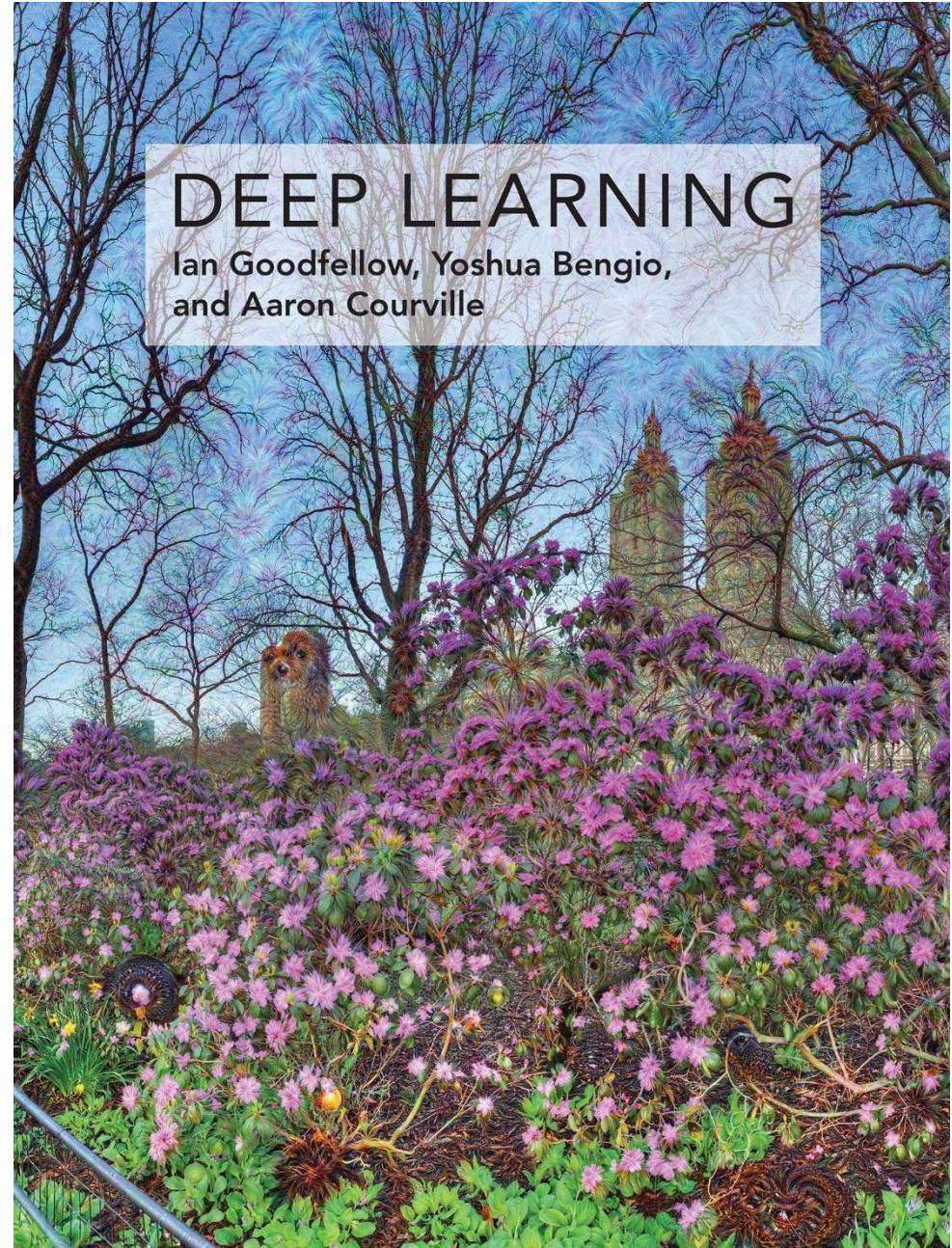


參考書籍

Original image:

<http://www.danielambrosi.com/Grand-Format-Collection/i-jbhqVhS/A>

<http://www.deeplearningbook.org/>



Schedule

四次作業

- HW1 : 深度學習流言終結者
 - 1-1: Deep is better than shallow?
 - 1-2: Is local minima an issue?
 - 1-3: Is deep learning generalizable?
- HW2 : Seq-to-seq model
 - 2-1: Video caption generation
 - 2-2: Chat-bot (option)

四次作業

- HW3: Generative Adversarial Network (GAN)
 - 3-1: Generation
 - 3-2: Conditional Generation
 - 3-3: Unsupervised Conditional Generation
(option)
- HW4: Reinforcement learning
 - 4-1: Policy gradient
 - 4-2: Q-learning
 - 4-3: Actor-critic

週次	日期	單元主題	
第1週	3/02	Course Introduction	
第2週	3/09	Theory I: Is deep structure better than shallow one? (HW1-1 released)	
第3週	3/16	Theory II: Why local minima is not a problem? (HW1-2 released)	
第4週	3/23	Theory III: Why large network does not overfit (even though theoretically it should)? (HW1-3 released)	
第5週	3/30	Sequence-to-sequence Model (HW2-1 released)	
第6週	4/06 (放假)	按學校行事曆放假一週 (HW1 due)	
第7週	4/13	Special Network Structures (HW2-2 released)	
第8週	4/20	期中考前一週放假	
第9週	4/27 (期中考週)	作業一表現優異同學上台分享	
第10週	5/04	Deep Generative Model (HW2 due, HW3-1 released)	
第11週	5/11	Deep Generative Model (HW3-2 released)	
第12週	5/18	Deep Generative Model (HW3-3 released)	
第13週	5/25	Deep Generative Model	
第14週	6/01	Deep Reinforcement Learning (HW3 due, HW4-1 released)	
第15週	6/08	Deep Reinforcement Learning (HW4-2 released)	
第16週	6/15	Deep Reinforcement Learning (HW4-3 released)	
第17週	6/22	期末考前一週放假	https://ceiba.ntu.edu.tw/modules/index.php?csn=8e8d96&default_fun=syllabus
第18週	6/29 (期末考週)	作業二、三、四表現優異同學上台分享 (HW4 due)	https://ceiba.ntu.edu.tw/modules/index.php?csn=8e8d96&default_fun=syllabus

Policy

評量方式

- 不點名、不考試
- 作業一 (25%)、作業二 (25%)、作業三 (25%)、作業四 (25%)

成績是相對的

成績是相對的

成績是相對的

組隊

- 每組 1 ~ 3 人，其中一人為組長
- 以組為單位進行所有作業
- 隊伍登記方法之後和作業一起說明
- 組內互評：學期結束前會有組內互評，會影響成績

上課方式

- 上課投影片和錄音會放到李宏毅的個人網頁上
 - 李宏毅的個人網頁：
http://speech.ee.ntu.edu.tw/~tlkagk/courses_MLSD18.html
- 社團：“Machine learning and having it deep and structured (2018 spring)”
 - <https://www.facebook.com/groups/907901649378100/>
 - 有問題歡迎直接在 FB社團上發問
 - 如果有同學知道答案歡迎回答
- 有任何和機器學習相關的想法都可以在 FB社團上發言

提醒

- 深度學習必要之惡：訓練過程需要時間、調參數需要時間.....
 - 作業早點開始
 - 死線前爆氣沒有什麼幫助
 - 健全的心靈
 - 試著調適等待過程的焦慮
 - 運算資源
 - 會有 MS Azure 的贊助，但是自備運算資源更好
- 請勿有作弊行為 (例如：抄襲同學的作業和程式、抄襲前人的作業和程式等等)
 - 請遵守各作業的規定
 - 如有規定未盡之處，則依照一般社會常識

需要的基礎能力和知識

- 本課程的定位為機器學習進階課程
- 程式能力：能夠使用某一個深度學習框架 (e.g. Tensorflow, pyTorch)
 - 使用 Keras 無法完成所有的作業
 - 本學期不會教深度學習框架的使用，請自學
 - Tensorflow
 - <https://fgc.stpi.narl.org.tw/activity/videoDetail/4b1141305d9cd231015d9d07dbe1002a>
 - <https://fgc.stpi.narl.org.tw/activity/videoDetail/4b1141305d9cd231015d9d0852c5002b>
 - <https://fgc.stpi.narl.org.tw/activity/videoDetail/4b1141305d9cd231015d9d08fb62002d>
 - pyTorch
 - <https://fgc.stpi.narl.org.tw/activity/videoDetail/4b1141305d9cd231015d9d0992ef0030>

需要的基礎能力和知識

- 基礎知識：希望聽課同學具備深度學習的基礎知識
- 《機器學習》錄影
 - DNN: <https://www.youtube.com/watch?v=Dr-WRIEFefw>
 - Tips for DNN: <https://www.youtube.com/watch?v=xki61j7z-30>
 - CNN: <https://www.youtube.com/watch?v=FrKWiRv254g>
 - RNN (Part 1): <https://www.youtube.com/watch?v=xCGidAeyS4M>
 - RNN (Part 2): <https://www.youtube.com/watch?v=xCGidAeyS4M>
 - Why Deep: <https://www.youtube.com/watch?v=XsC9byQkUH8>
 - Auto-encoder: <https://www.youtube.com/watch?v=Tk5B4seA-AU>
 - Deep generative model (Part 1):
<https://www.youtube.com/watch?v=YNUek8ioAJk>
 - Deep generative model (Part 2):
<https://www.youtube.com/watch?v=8zomhgKrsmQ>
 - Reinforcement Learning:
<https://www.youtube.com/watch?v=W8XF3ME8G2I>

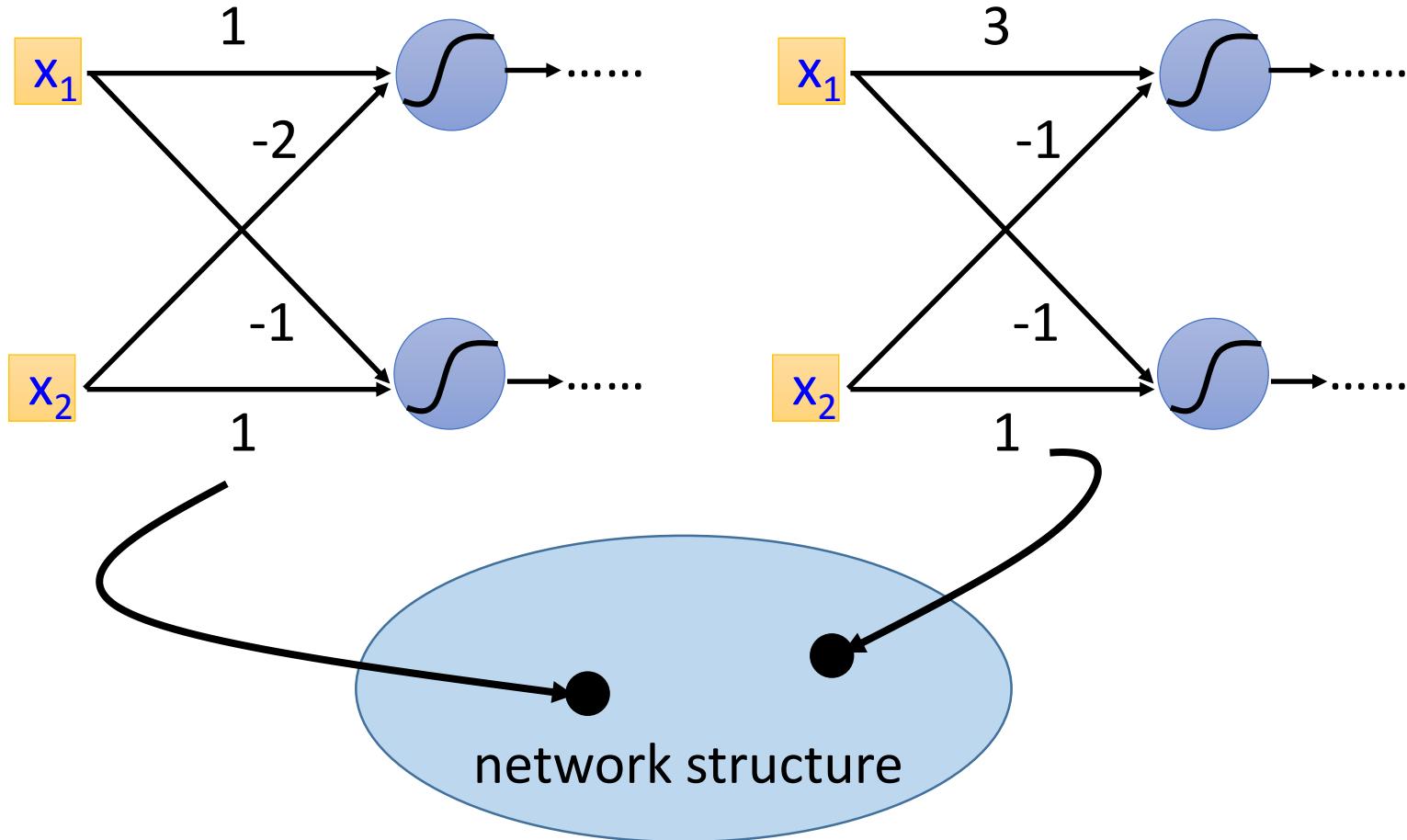
加簽

- 等一下助教會公告作業 0
 - 作業零其實是《機器學習》這門課的作業
 - 變相擋修《機器學習》
 - 可以用任何機器學習方法完成，不限深度學習，只要達到要求的正確率就行
 - 助教不會改作業零的程式
 - **以個人為單位完成**
- 本課程預期修課人數為 40 ~ 80 人，上限為 120 人
 - 如果完成作業零人數超過上限，則按照完成作業的時間排序
 - 完成作業 0 後，授權碼取得方式另行公告

Theory I: Why Deep Structure?

李宏毅
Hung-yi Lee

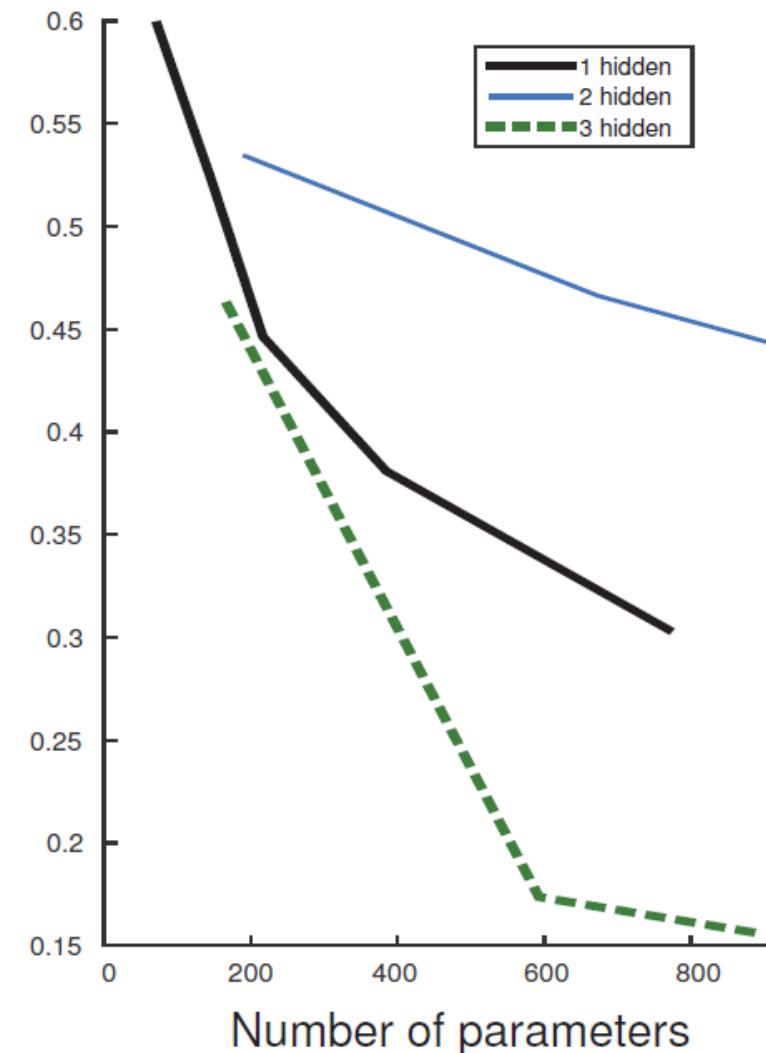
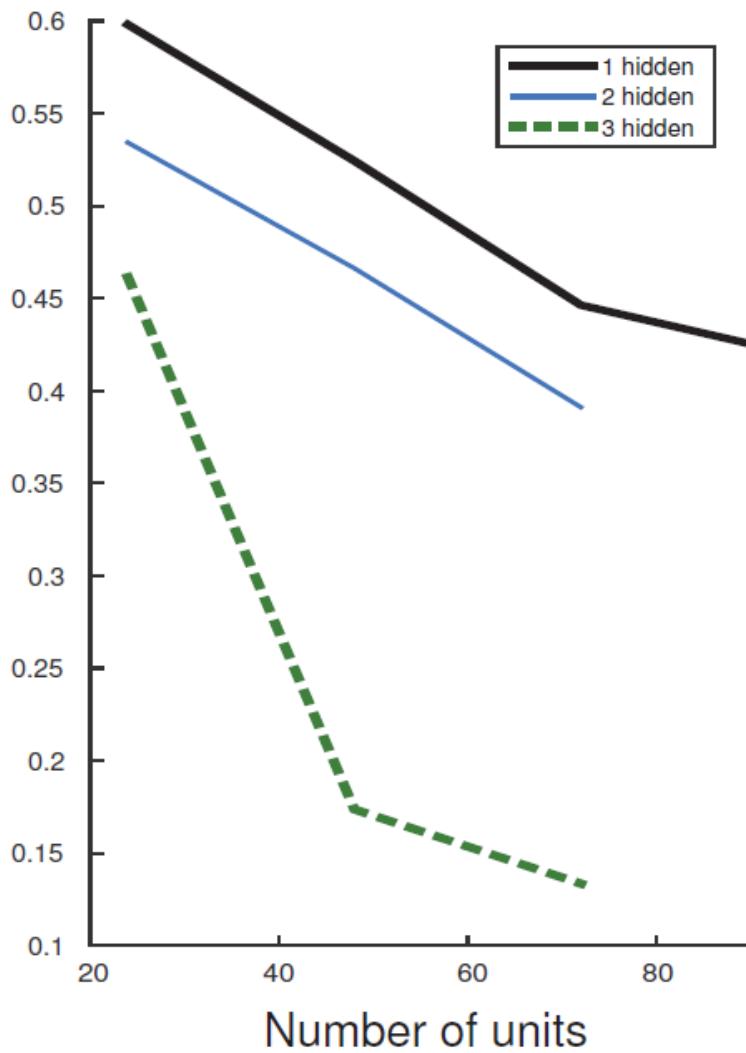
Review



Given structure, each set
of parameter is a function.

The network structure
defines a function set.

$$f(x) = 2(2\cos^2(x)-1)^2 - 1$$



Source of image:

<https://www.aaai.org/ocs/index.php/AAAI/AAAI17/paper/viewPaper/14849>

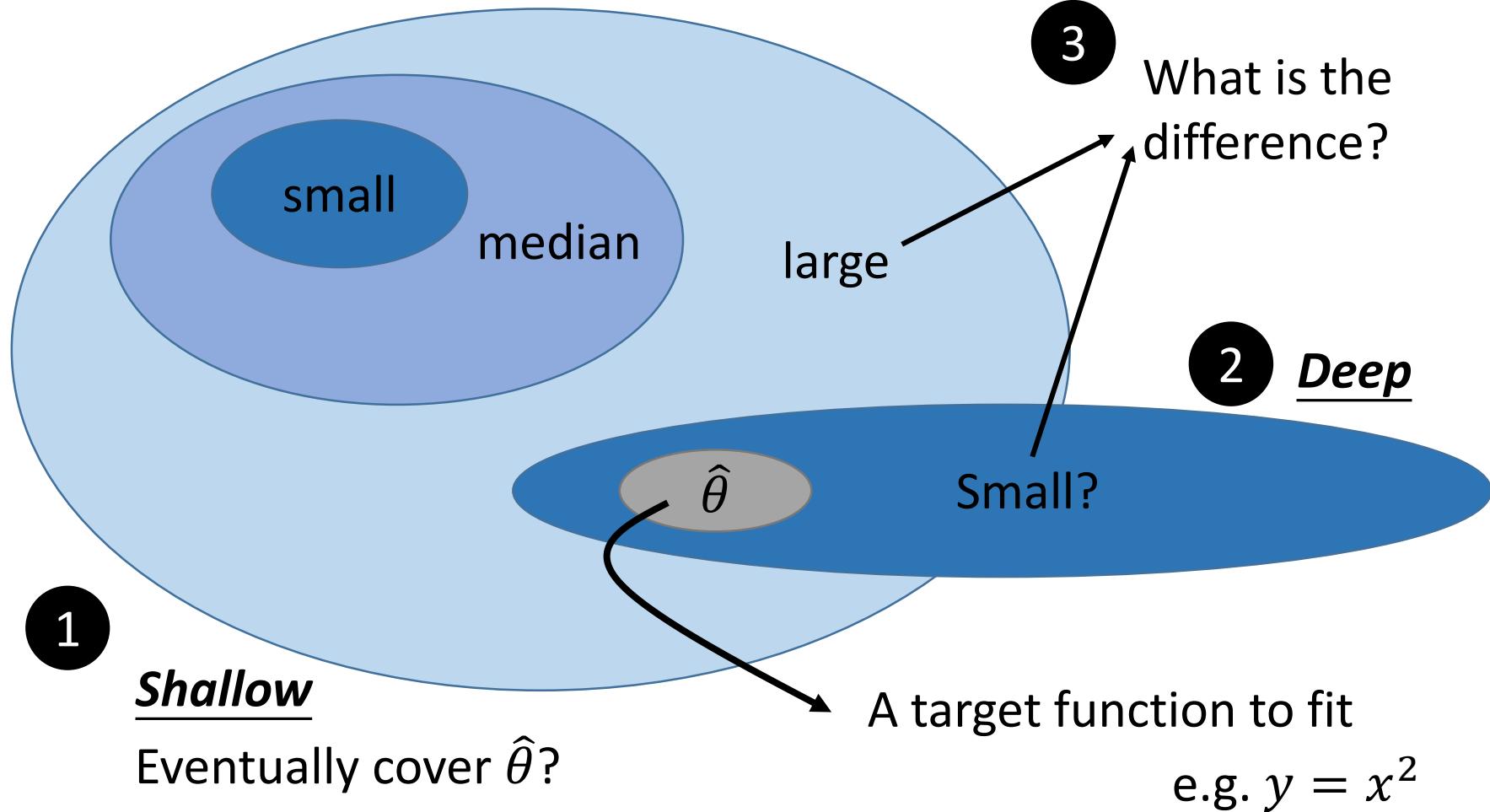
Outline

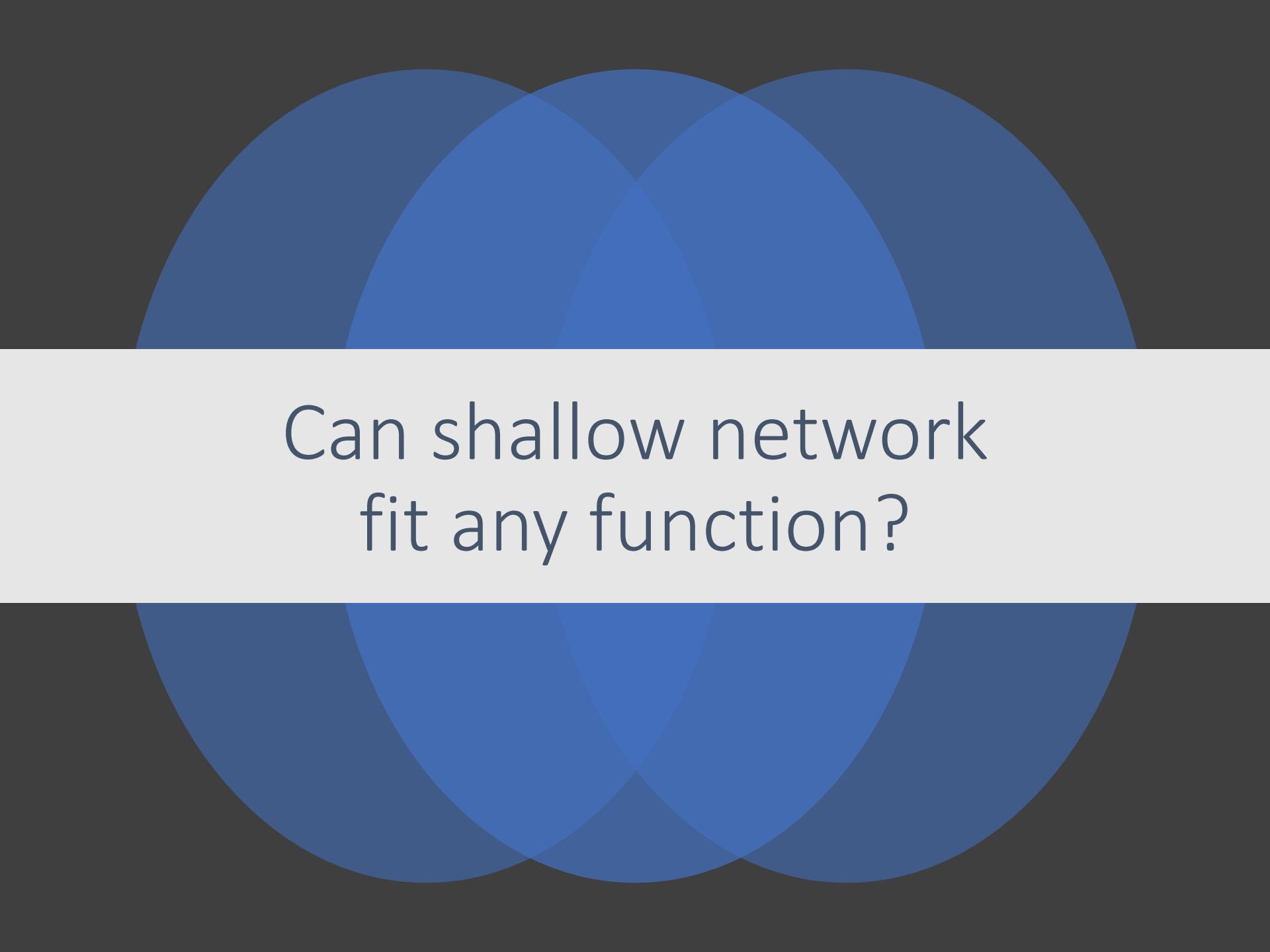
- Q1: Can shallow network fit any function?
- Potential of deep
- Q2: How to use deep to fit functions?
- Q3: Is deep better than shallow?
- Review some related theories



Outline

Notice: We do not discuss
optimization and generation today.

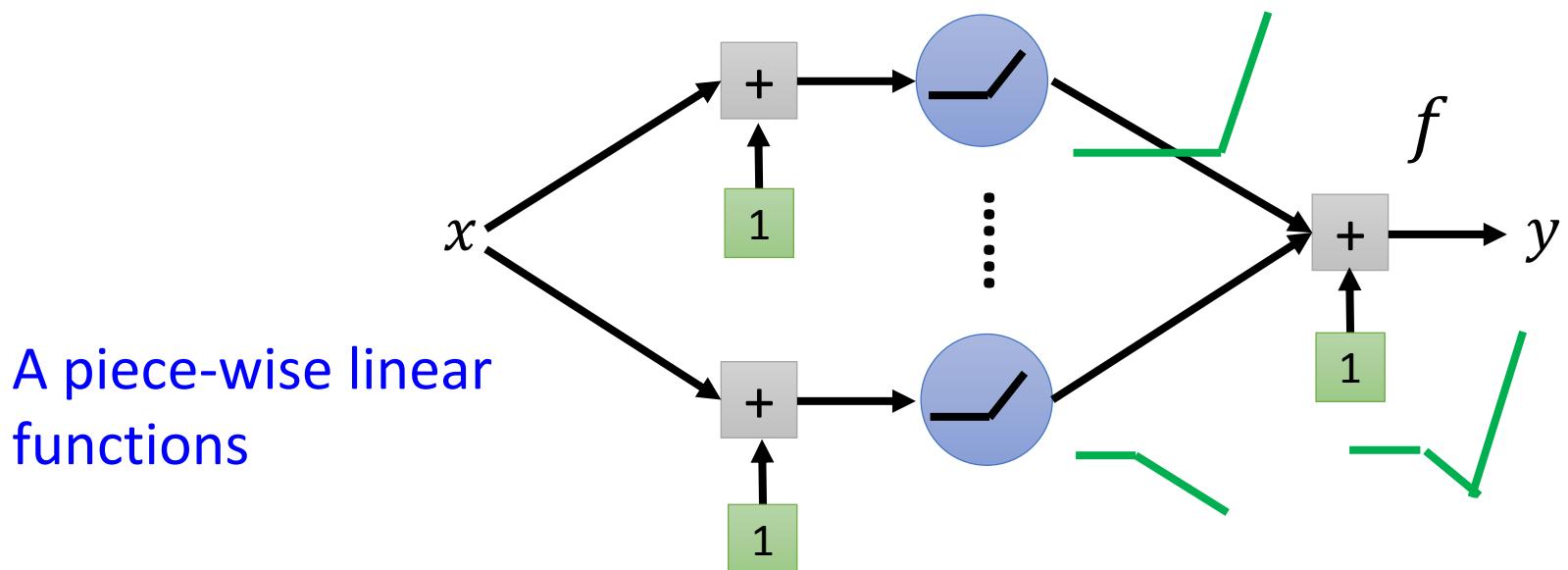




Can shallow network
fit any function?

Universality

- Given a shallow network structure with one hidden layer with ReLU activation and linear output



- Given a L-Lipschitz function f^*
 - How many neurons are needed to approximate f^* ?

Universality

- Given a L-Lipschitz function f^*
 - How many neurons are needed to approximate f^* ?

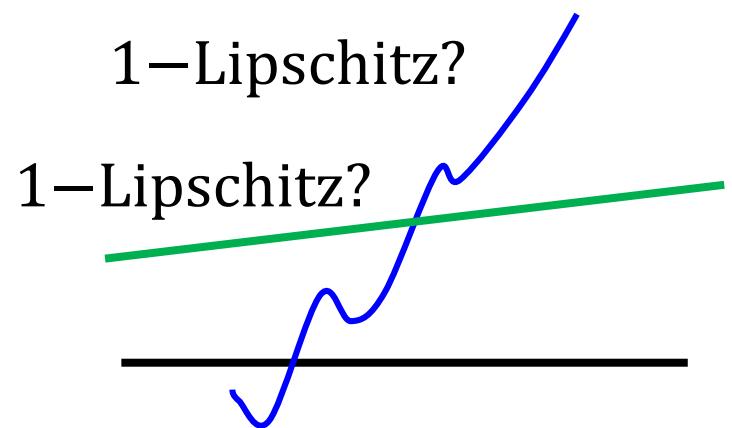
L-Lipschitz Function (smooth)

$$\|f(x_1) - f(x_2)\| \leq L\|x_1 - x_2\|$$

Output
change

Input
change

$L=1$ for "1 – Lipschitz"



Universality

$$\max_{0 \leq x \leq 1} |f(x) - f^*(x)| \leq \varepsilon$$

$$\sqrt{\int_0^1 |f(x) - f^*(x)|^2 dx} \leq \varepsilon$$

- Given a L-Lipschitz function f^*
 - How many neurons are needed to approximate f^* ?

$$f \in N(K)$$

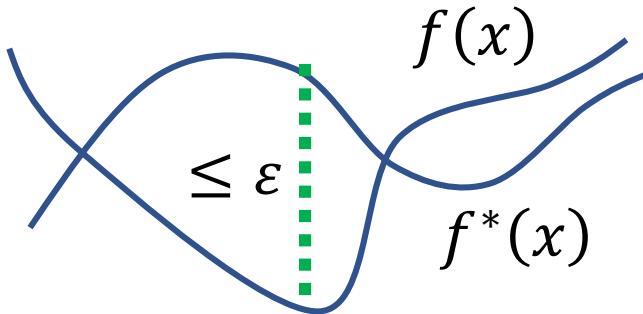
The function space defined by the network with K neurons.

Given a small number $\varepsilon > 0$

What is the number of K such that

$$\text{Exist } f \in N(K), \max_{0 \leq x \leq 1} |f(x) - f^*(x)| \leq \varepsilon$$

The difference between $f(x)$ and $f^*(x)$ is smaller than ε .



Universality

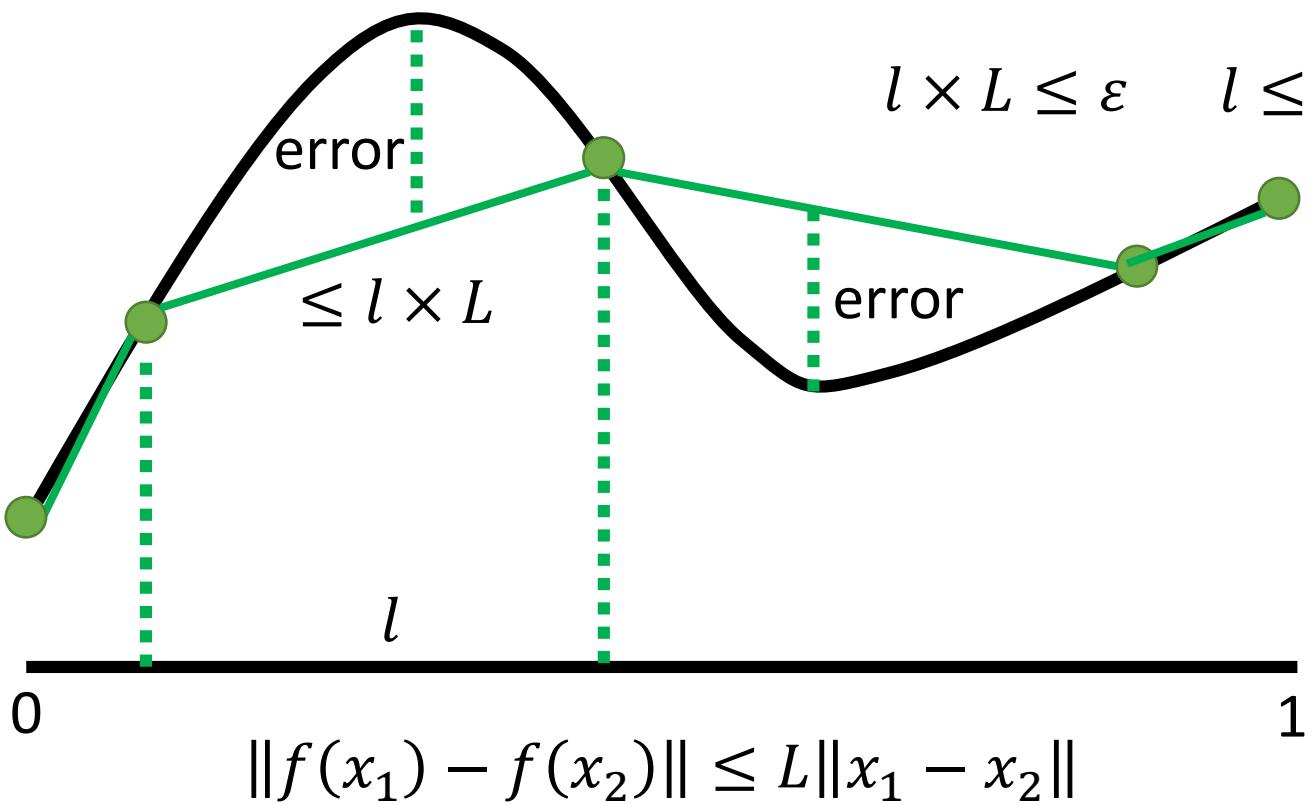
- L-Lipschitz function f^*

All the functions in $N(K)$ are piecewise linear.

Approximate f^* by a piecewise linear function f

How to make the errors $\leq \varepsilon$

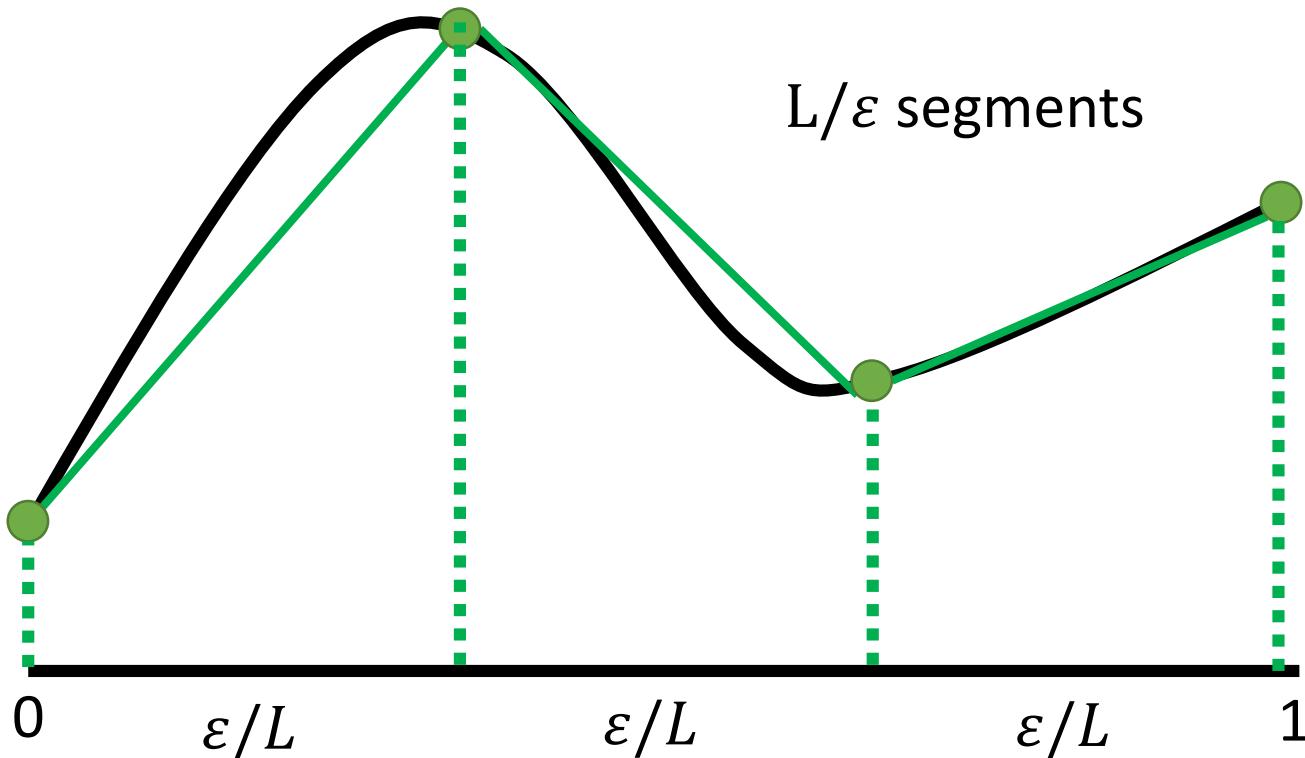
$$l \times L \leq \varepsilon \quad l \leq \varepsilon/L$$

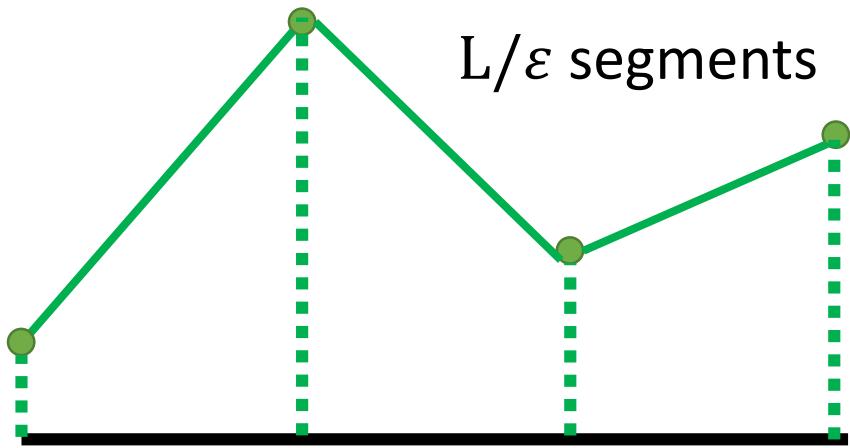


Universality

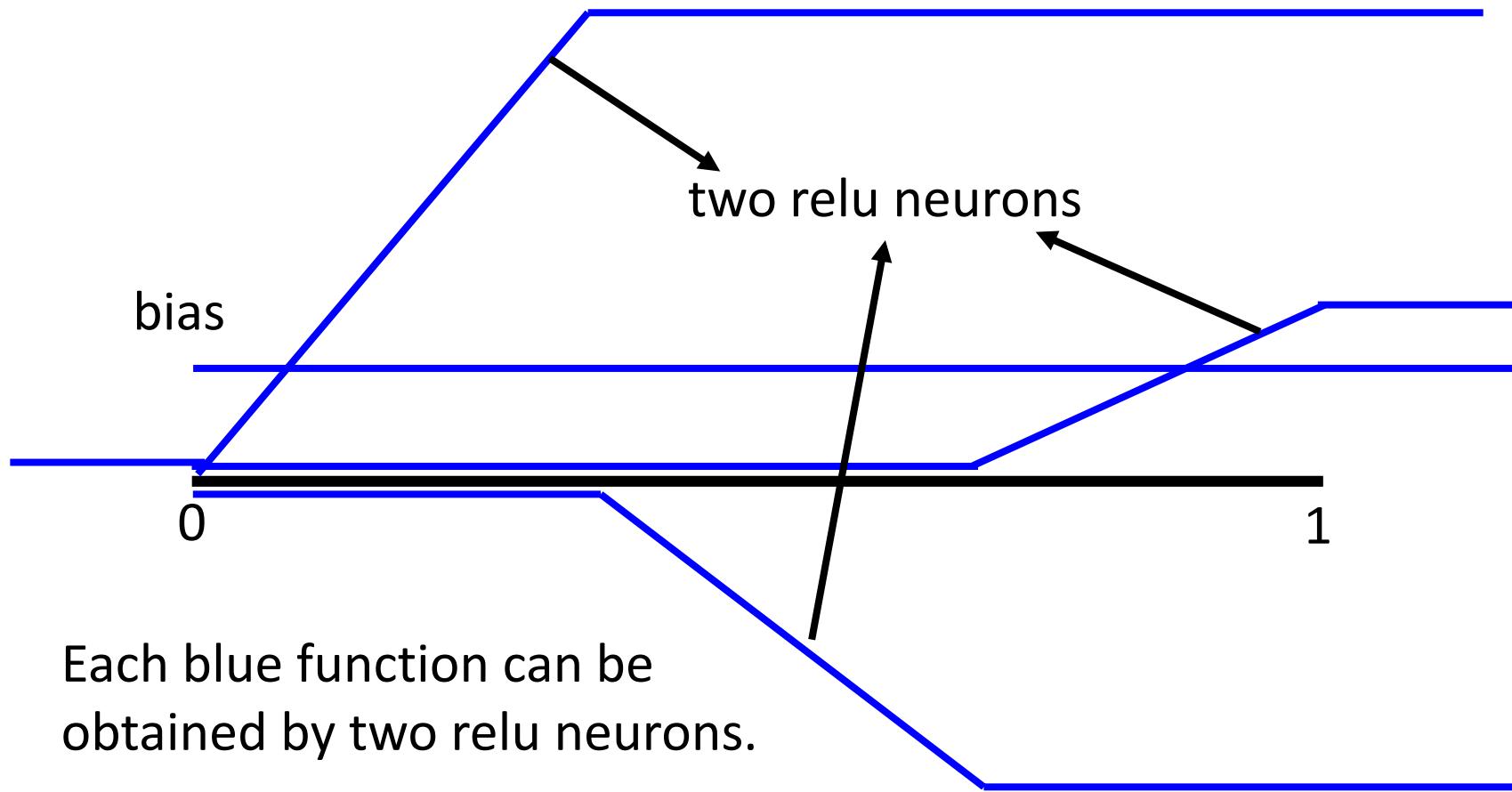
- L-Lipschitz function f^*

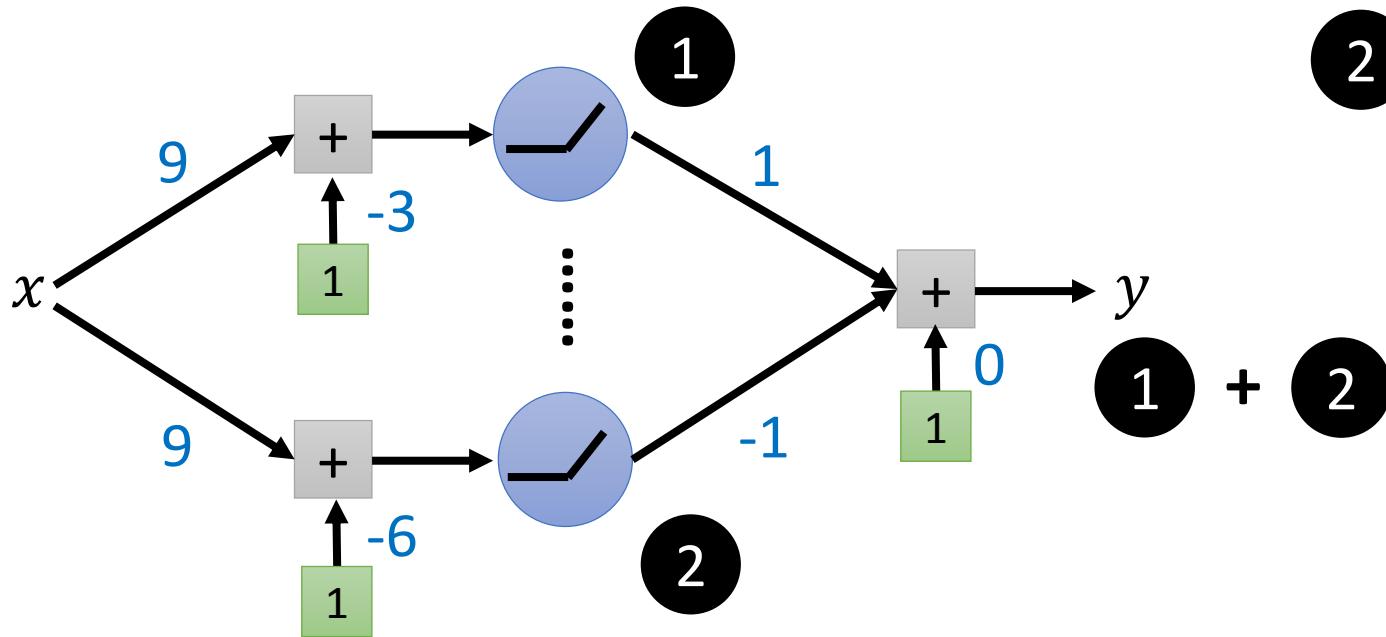
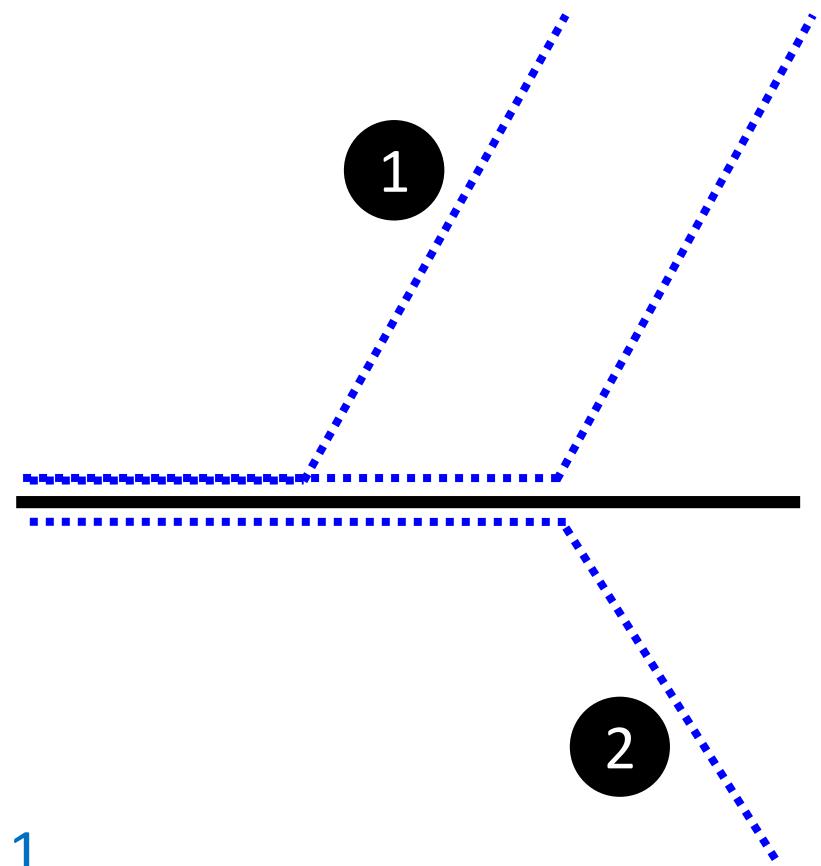
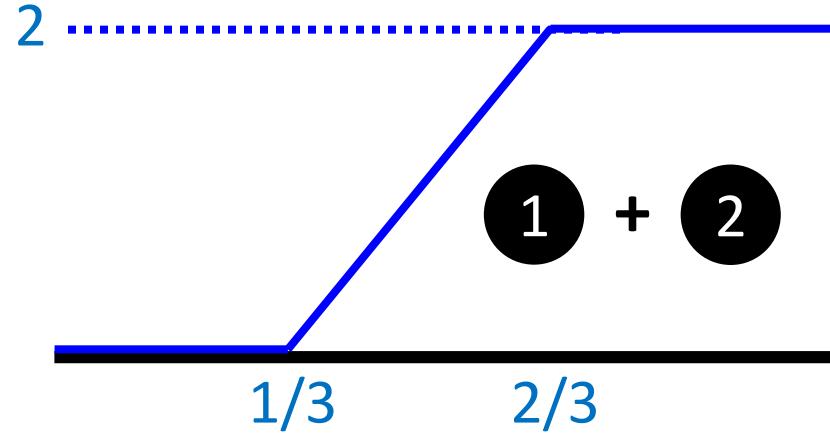
How to make a 1 hidden layer relu network have the output like green curve?

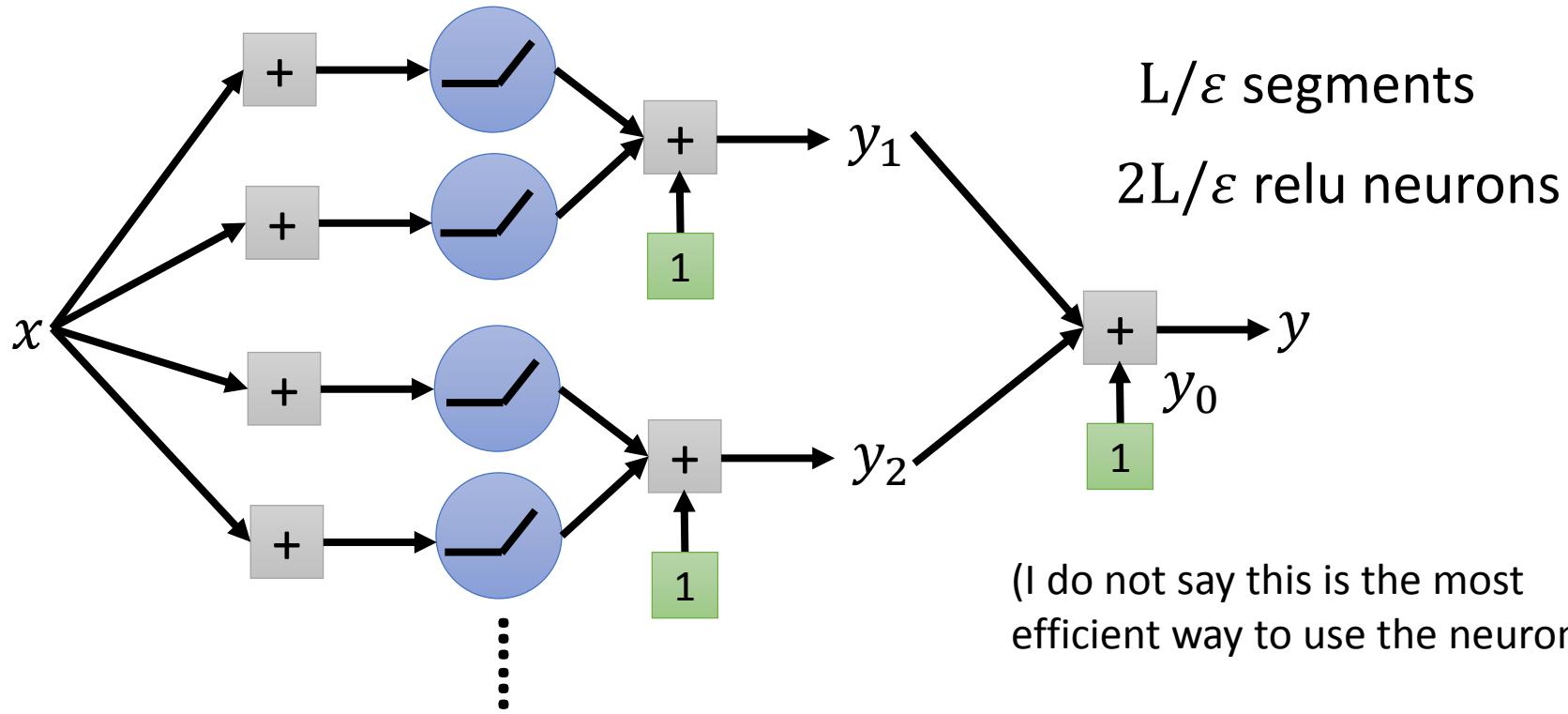
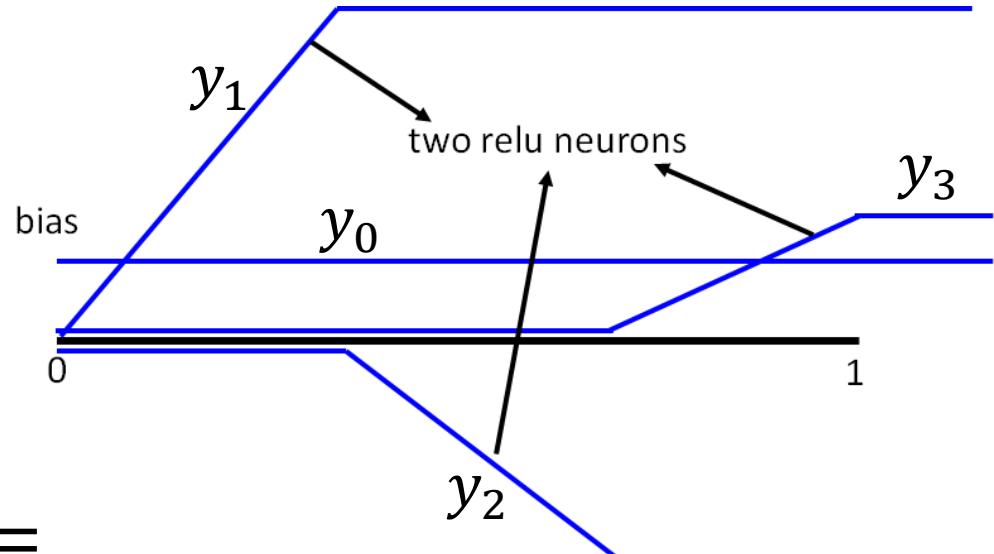
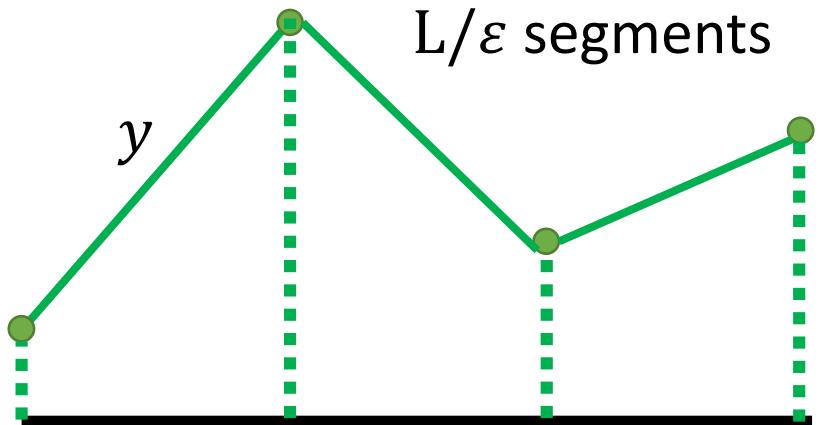




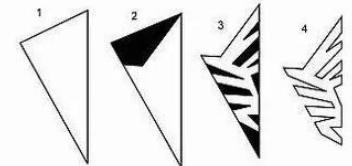
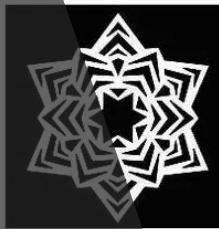
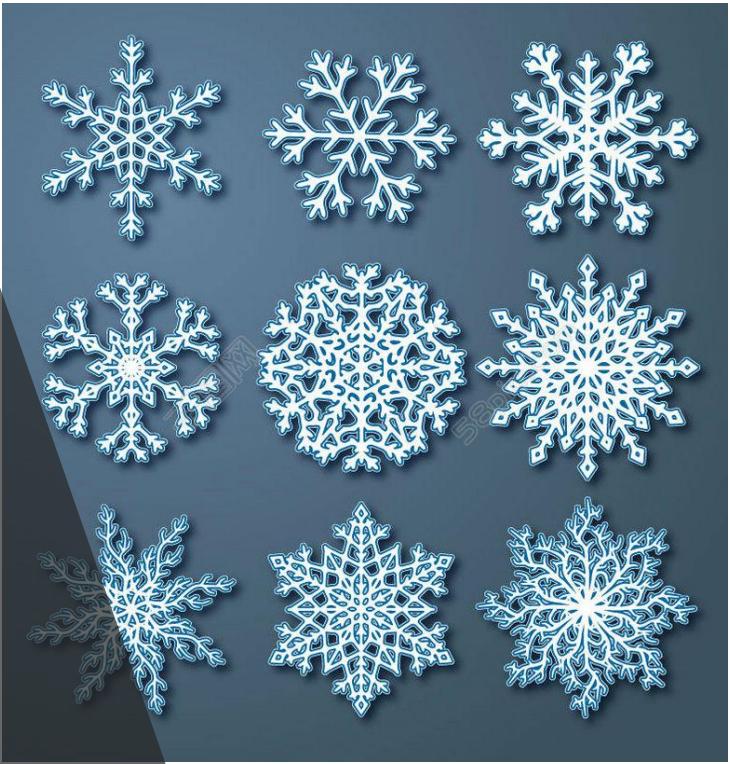
The summation of the blue functions is the green one.





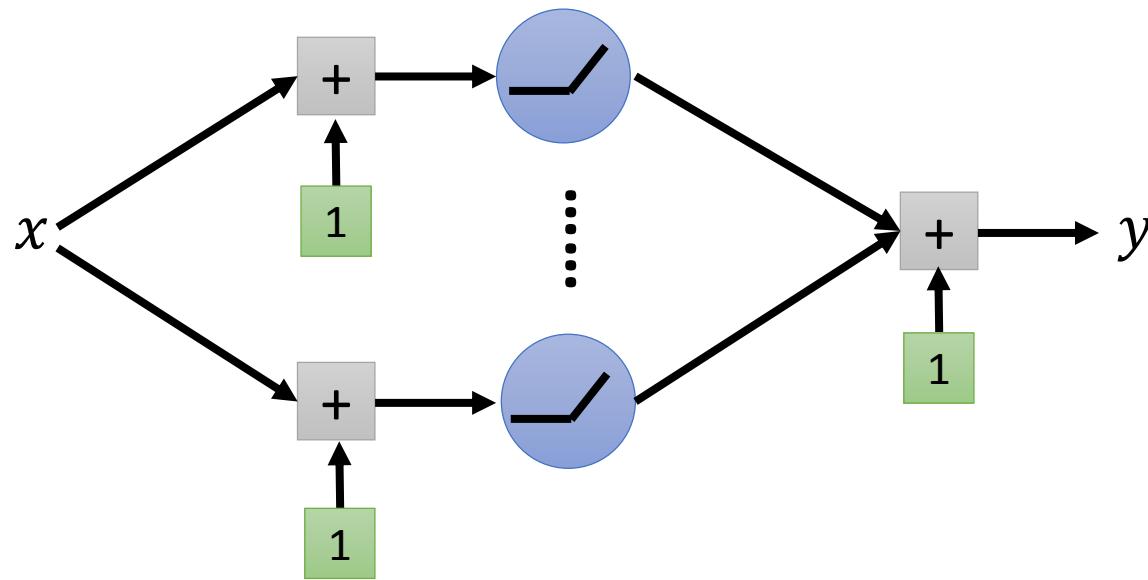


Potential of deep



头像 / 幼师宝典

Why we need deep?



Yes, shallow network can represent any function.

However, using deep structure is more effective.

Analogy – Programming

- Solve any problem by two lines (shallow)

- Input = K
- Line 1: row no. = MATCH_KEY(K)
- Line 2: Output the value at row no.

Input (key)	Output (value)
A	A'
B	B'
C	C'
D	D'
.....

- Considering SVM with kernel

$$y = \sum_n \alpha_n K(x^n, x)$$

- Using multiple steps to solve problems is more efficient (deep)

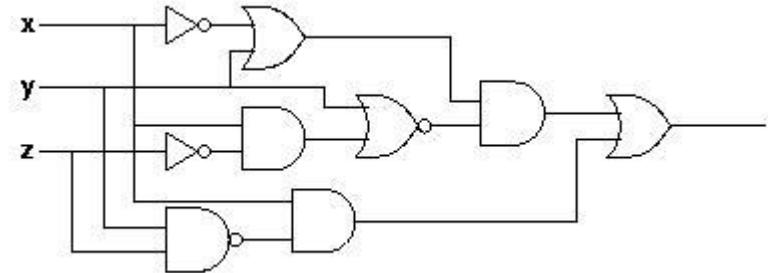
Analogy

Logic circuits

- Logic circuits consists of **gates**
- **A two layers of logic gates** can represent **any Boolean function.**
- Using multiple layers of logic gates to build some functions are much simpler



less gates needed



Neural network

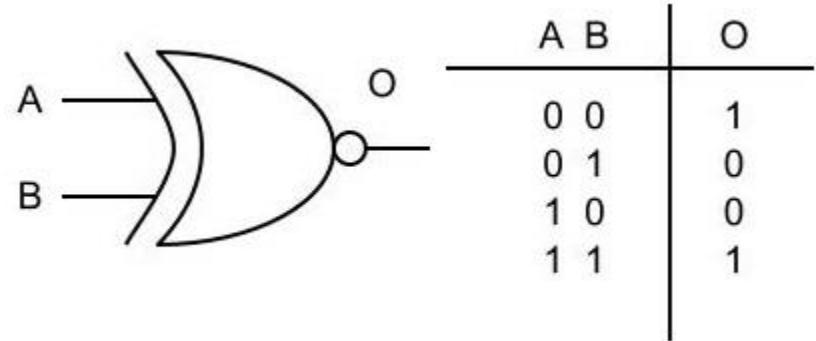
- Neural network consists of **neurons**
- **A hidden layer network** can represent **any continuous function.**
- Using multiple layers of neurons to represent some functions are much simpler



less neurons

This page is for EE background.

Analogy

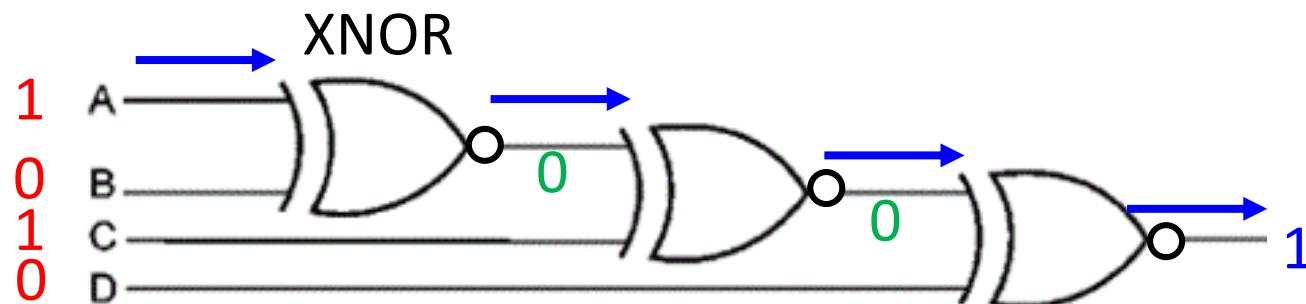


- E.g. parity check



For input sequence
with d bits,

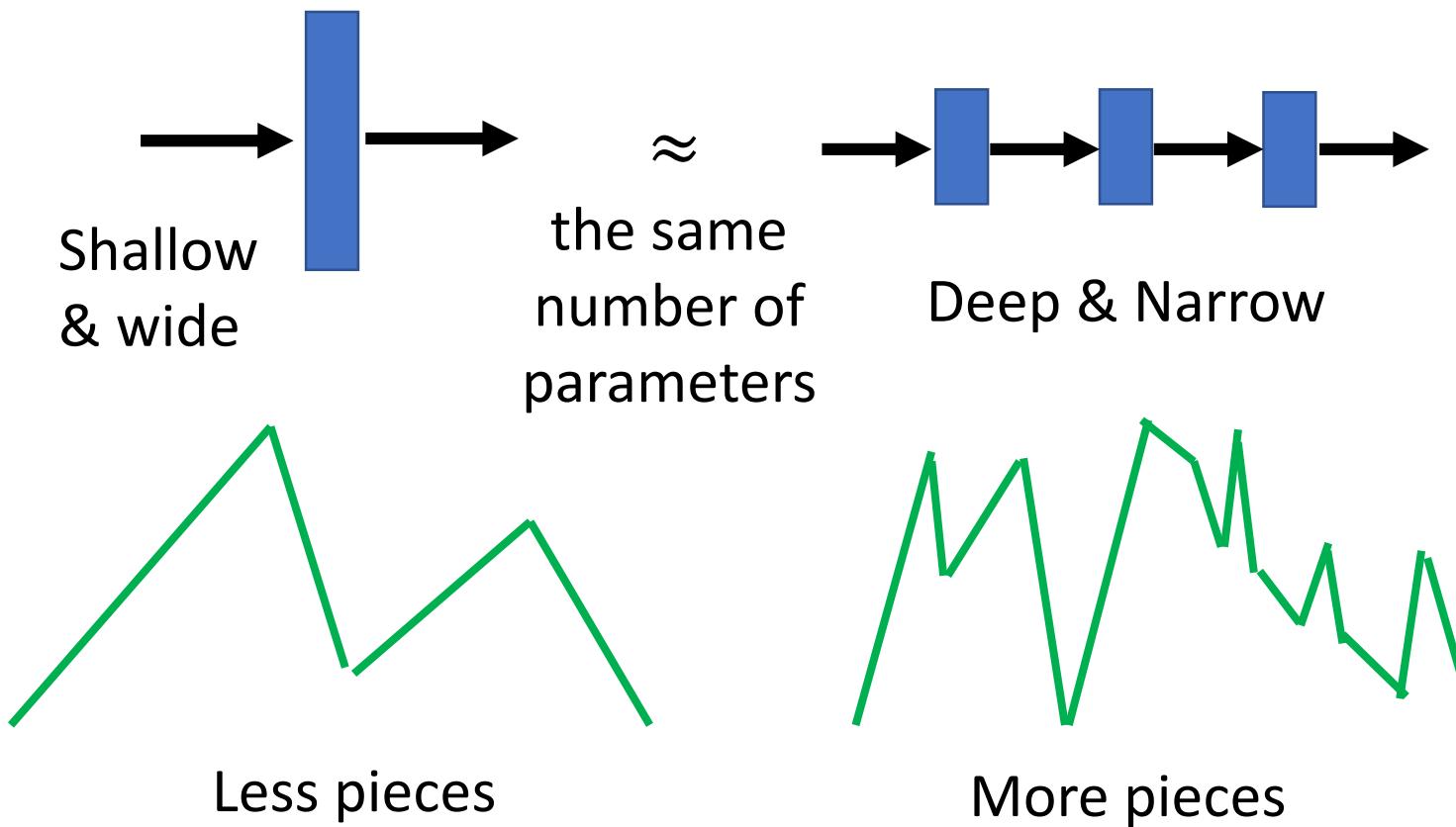
Two-layer circuit
need $O(2^d)$ gates.



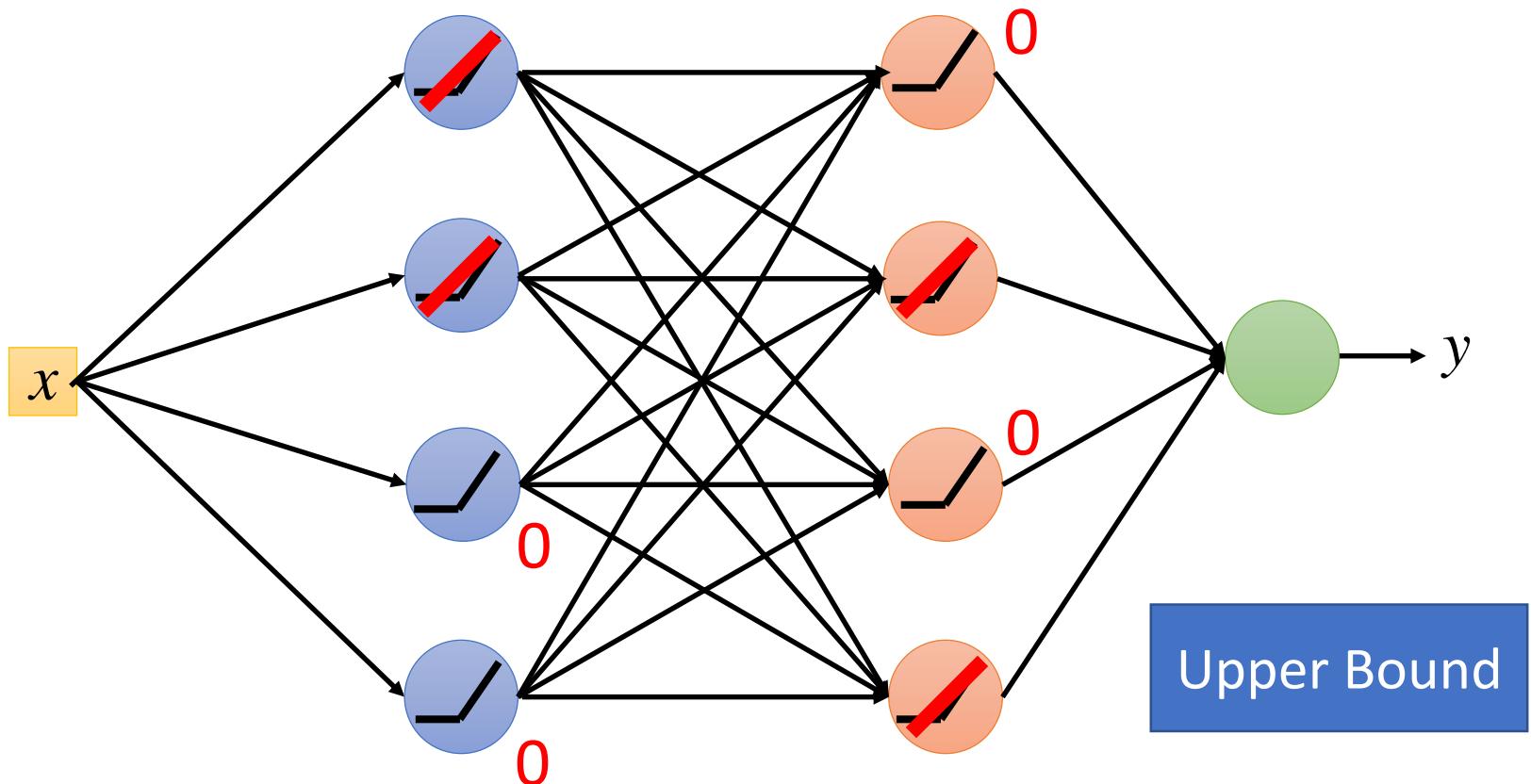
With multiple layers, we need only $O(d)$ gates.

Why we need deep?

- ReLU networks can represent piecewise linear functions



Upper Bound of Linear Pieces

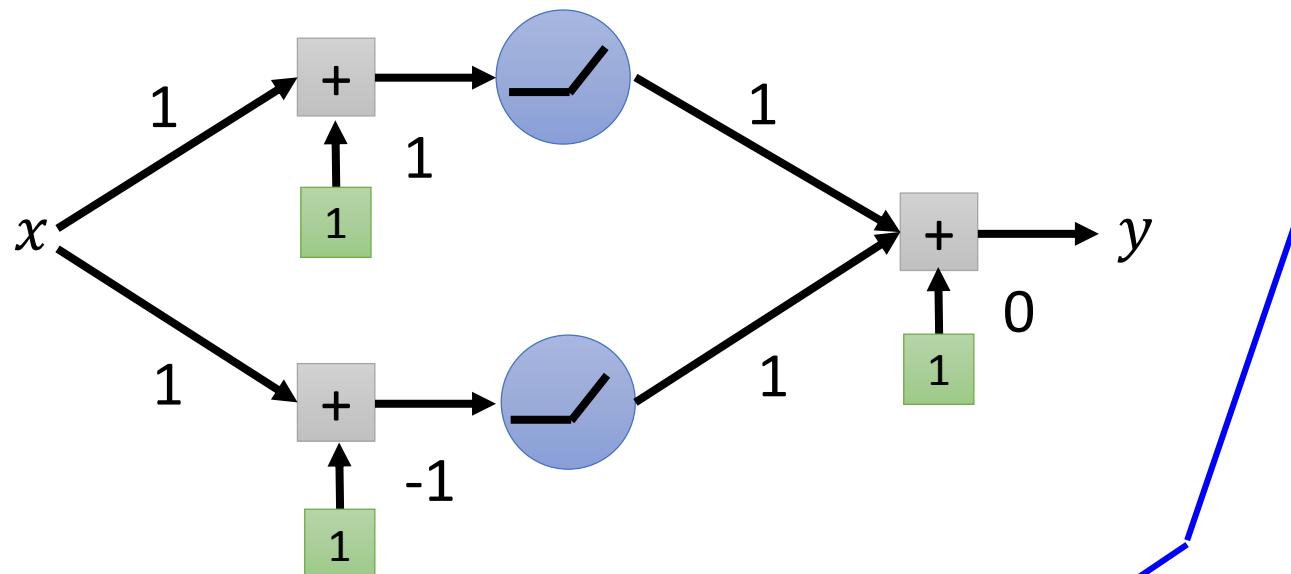


Each “activation pattern” defines a linear function.

N neurons \rightarrow 2^N “activation patterns” \rightarrow 2^N “linear pieces”

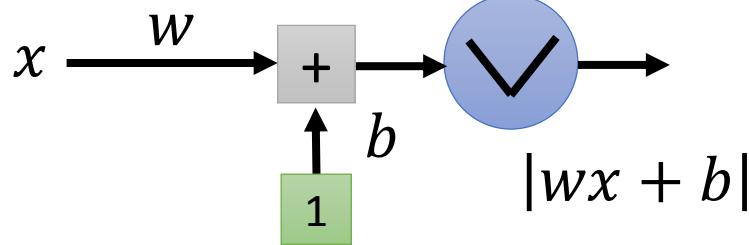
Upper Bound of Linear Pieces

- Not all the “activation patterns” available

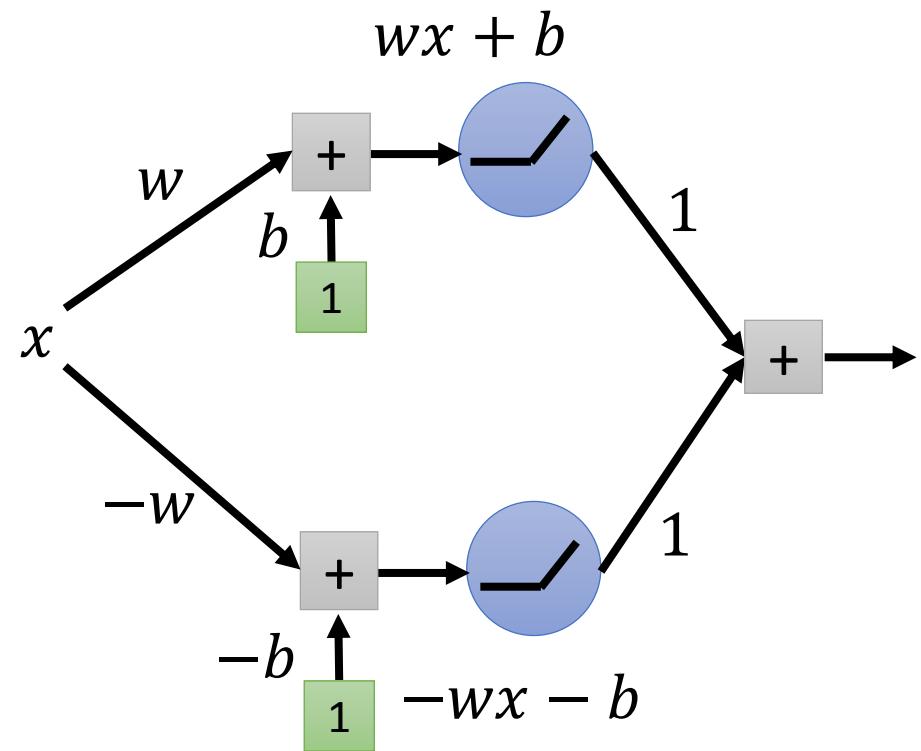


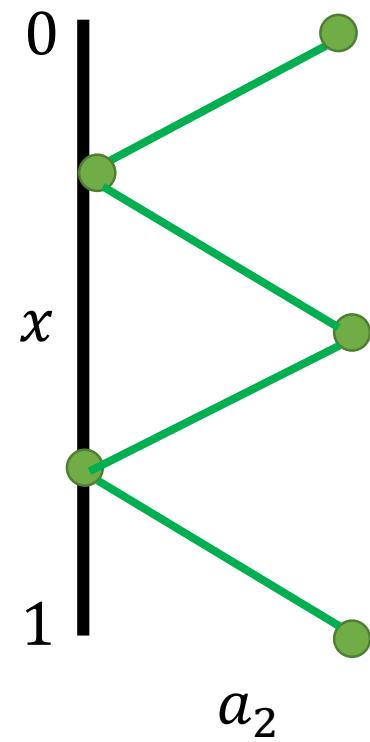
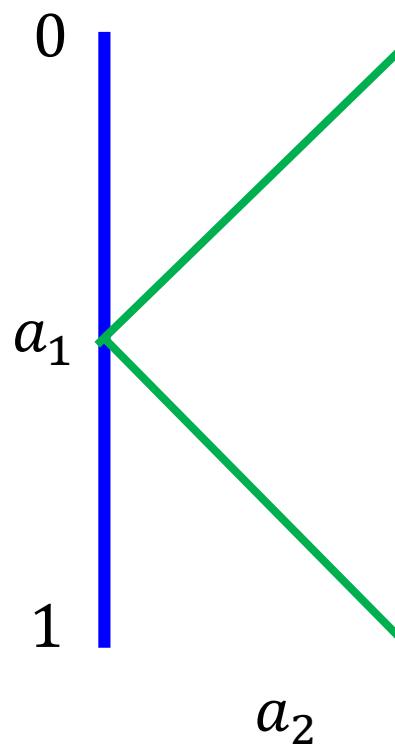
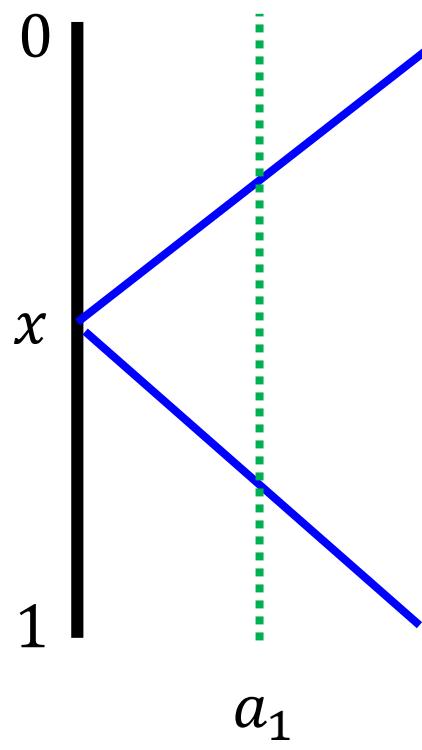
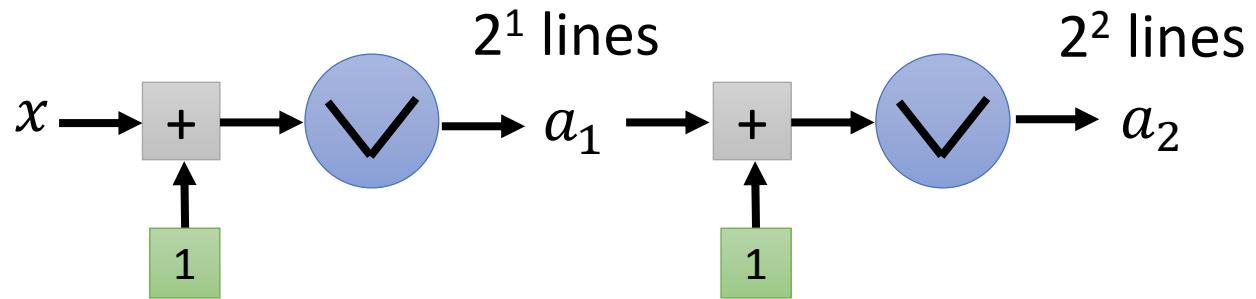
In shallow network, each neuron only provides one linear piece.

Abs Activation Function

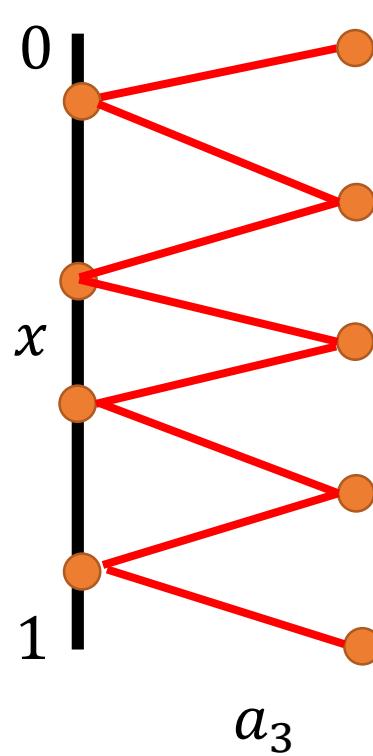
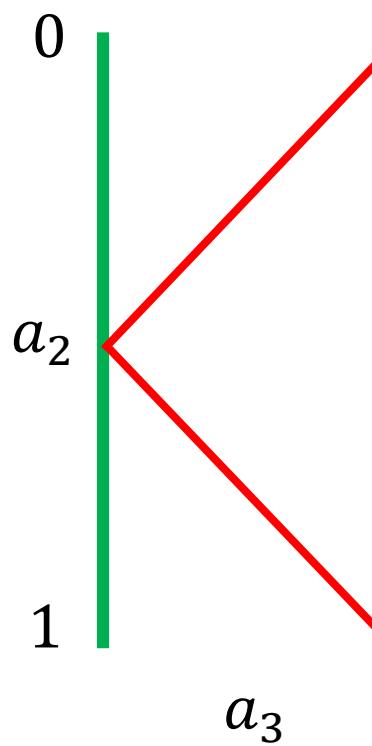
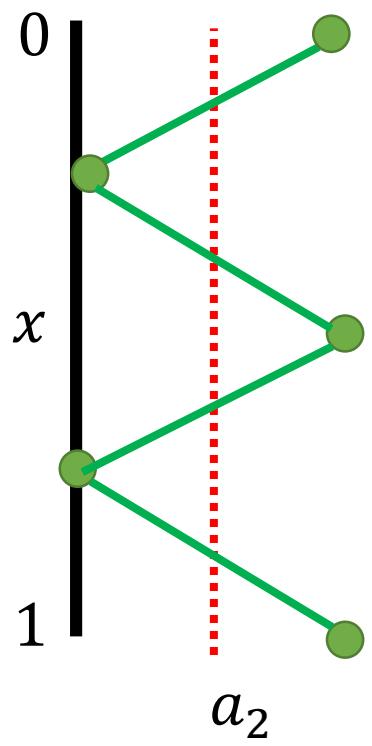
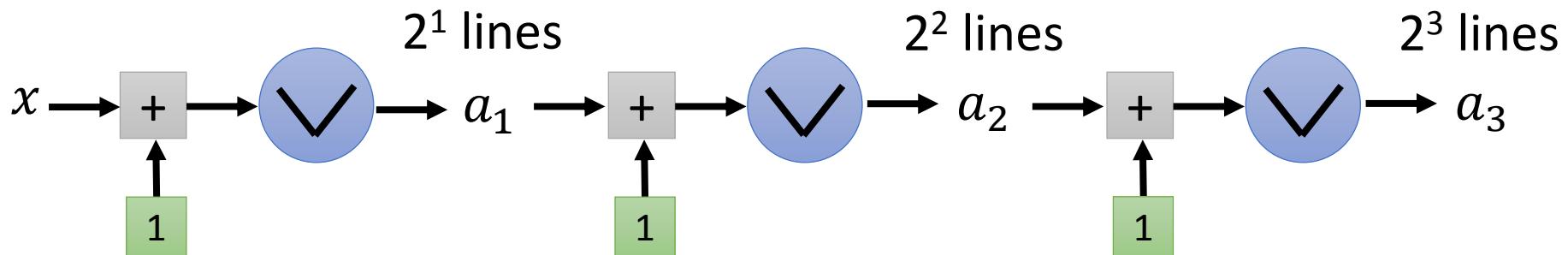


Use two relu to implement an abs activation function

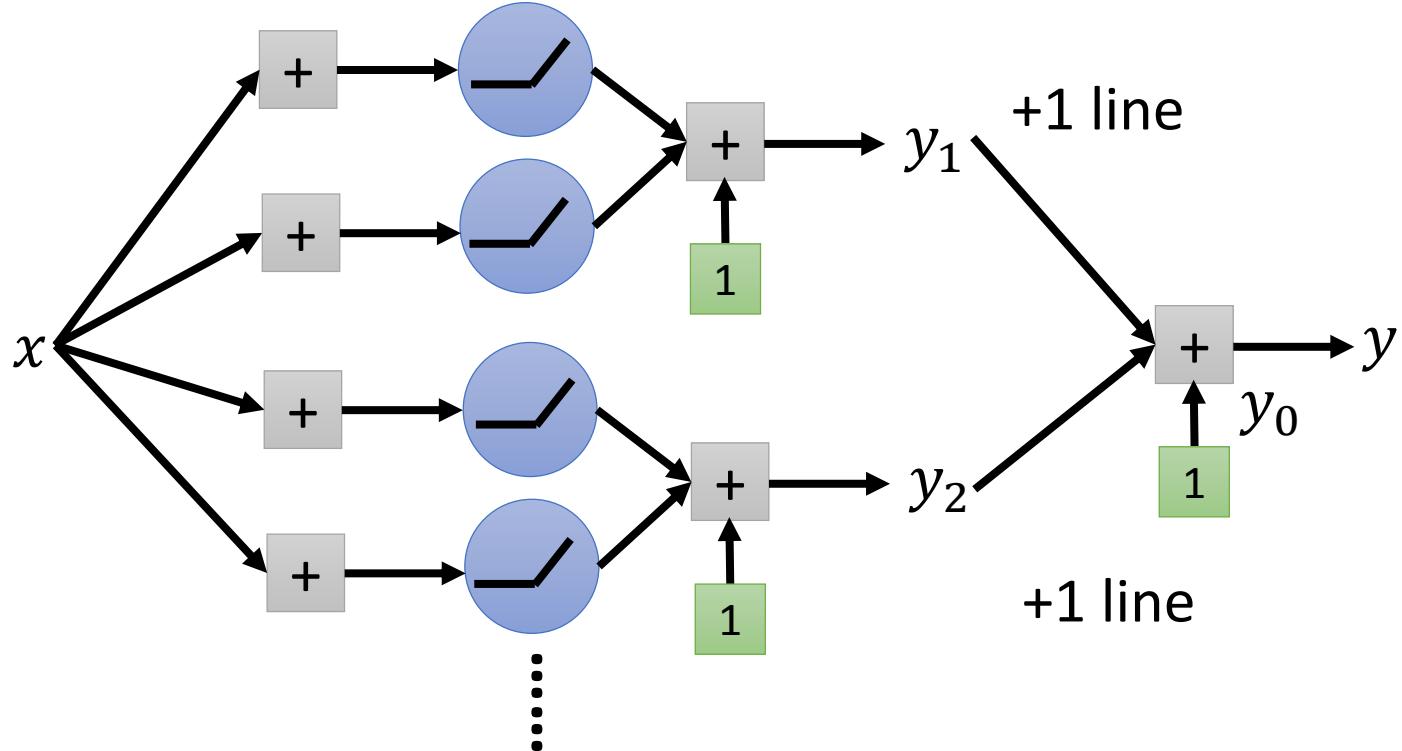




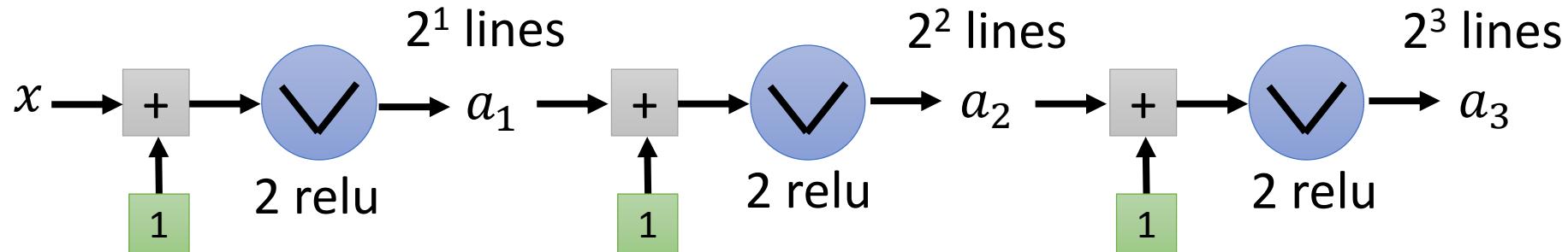
Each node added  The regions are twice.



Shallow



Deep



Lower Bound of Linear Pieces

If K is width, H is depth

We can have at least K^H pieces

Depth has much larger influence than depth.

Razvan Pascanu, Guido Montufar, Yoshua Bengio, "On the number of response regions of deep feed forward networks with piece-wise linear activations", ICLR, 2014

Guido F. Montufar, Razvan Pascanu, Kyunghyun Cho, Yoshua Bengio, "On the Number of Linear Regions of Deep Neural Networks", NIPS, 2014

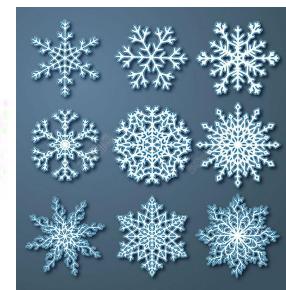
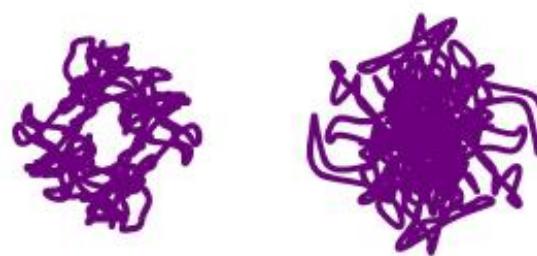
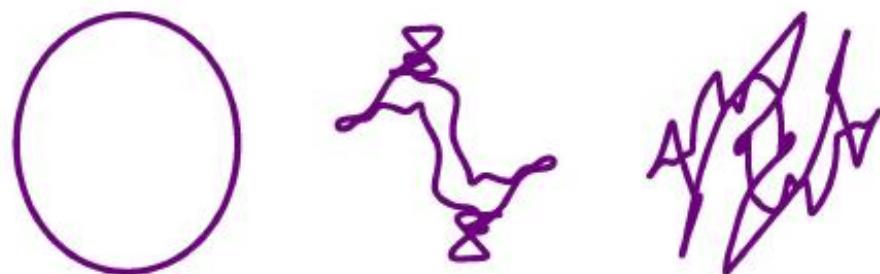
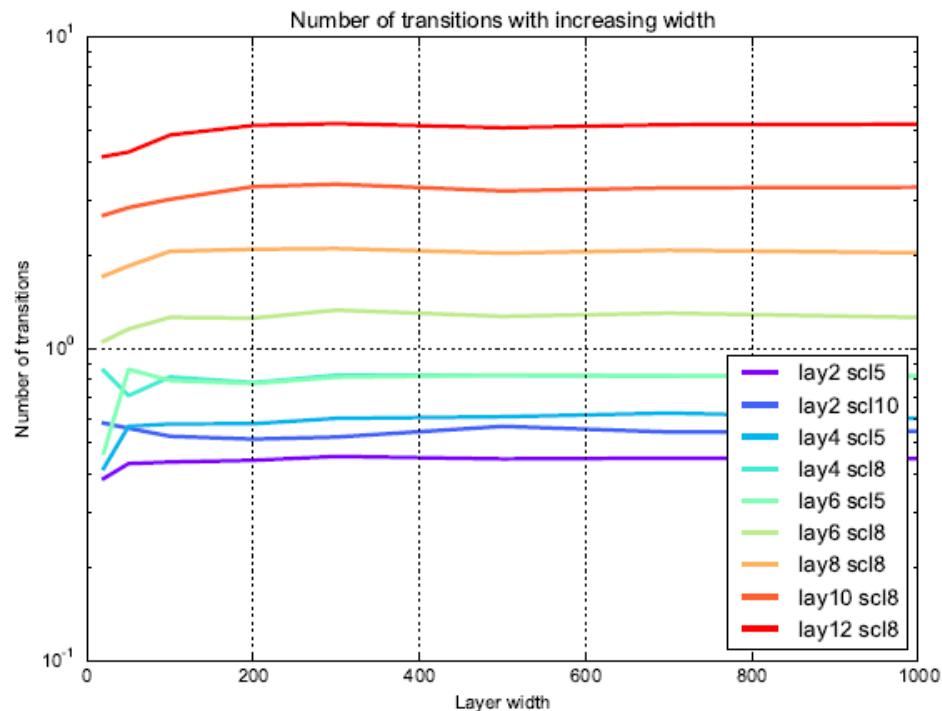
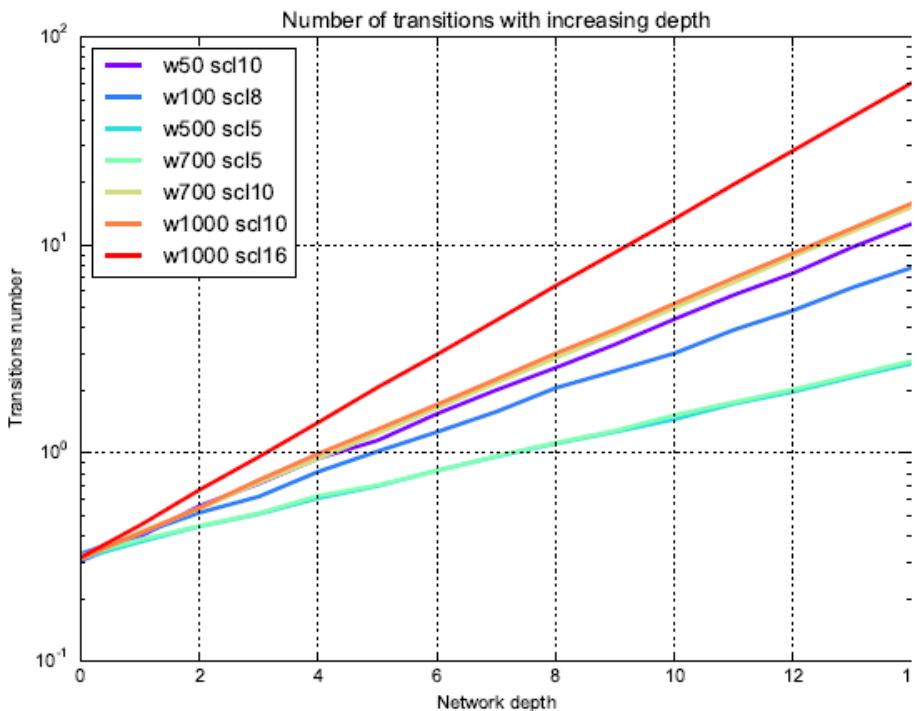
Raman Arora, Amitabh Basu, Poorya Mianjy, Anirbit Mukherjee, "Understanding Deep Neural Networks with Rectified Linear Units", ICLR 2018

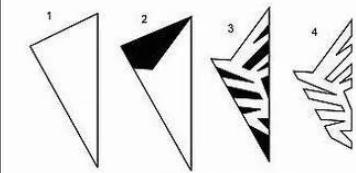
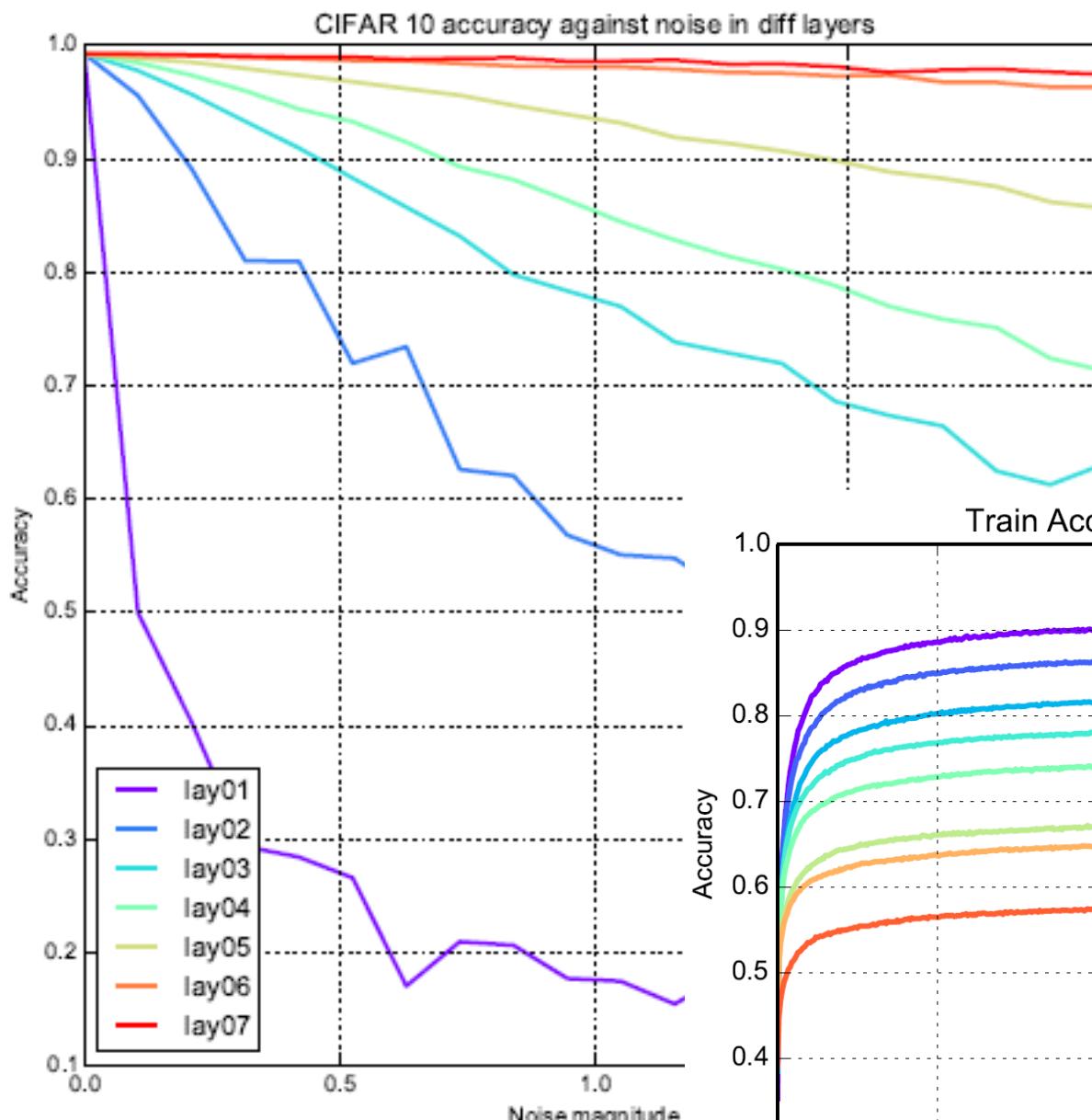
Thiago Serra, Christian Tjandraatmadja, Srikanth Ramalingam, "Bounding and Counting Linear Regions of Deep Neural Networks", arXiv, 2017

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, Jascha Sohl-Dickstein, On the Expressive Power of Deep Neural Networks, ICML, 2017

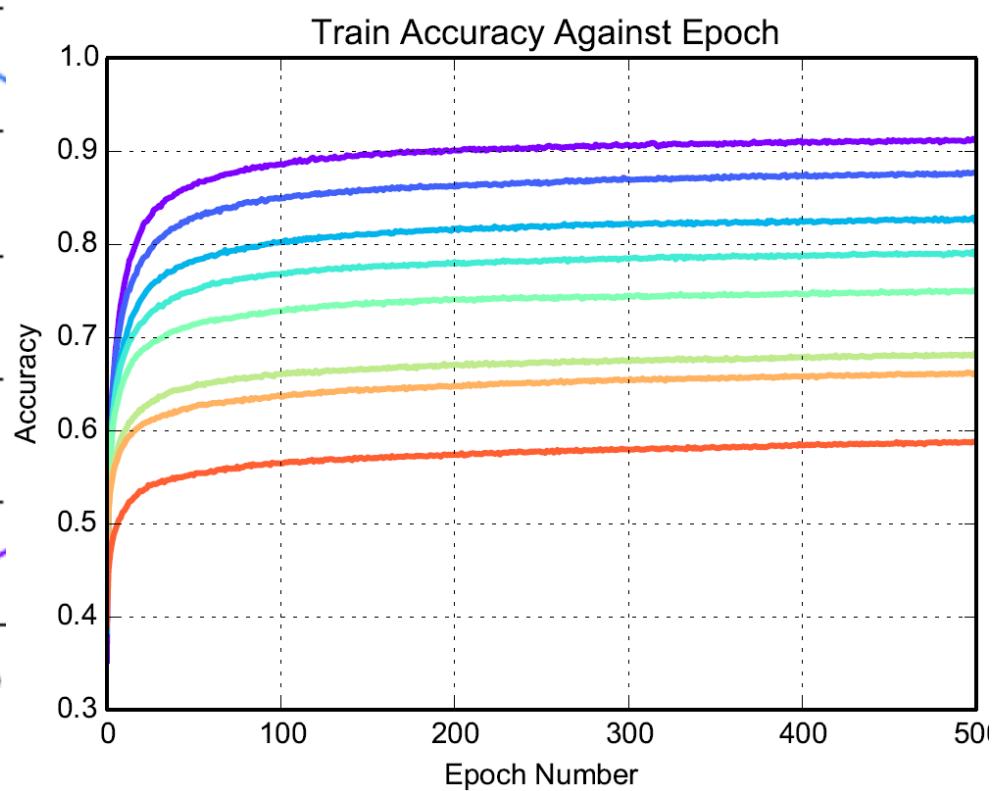
Experimental Results

(MNIST)



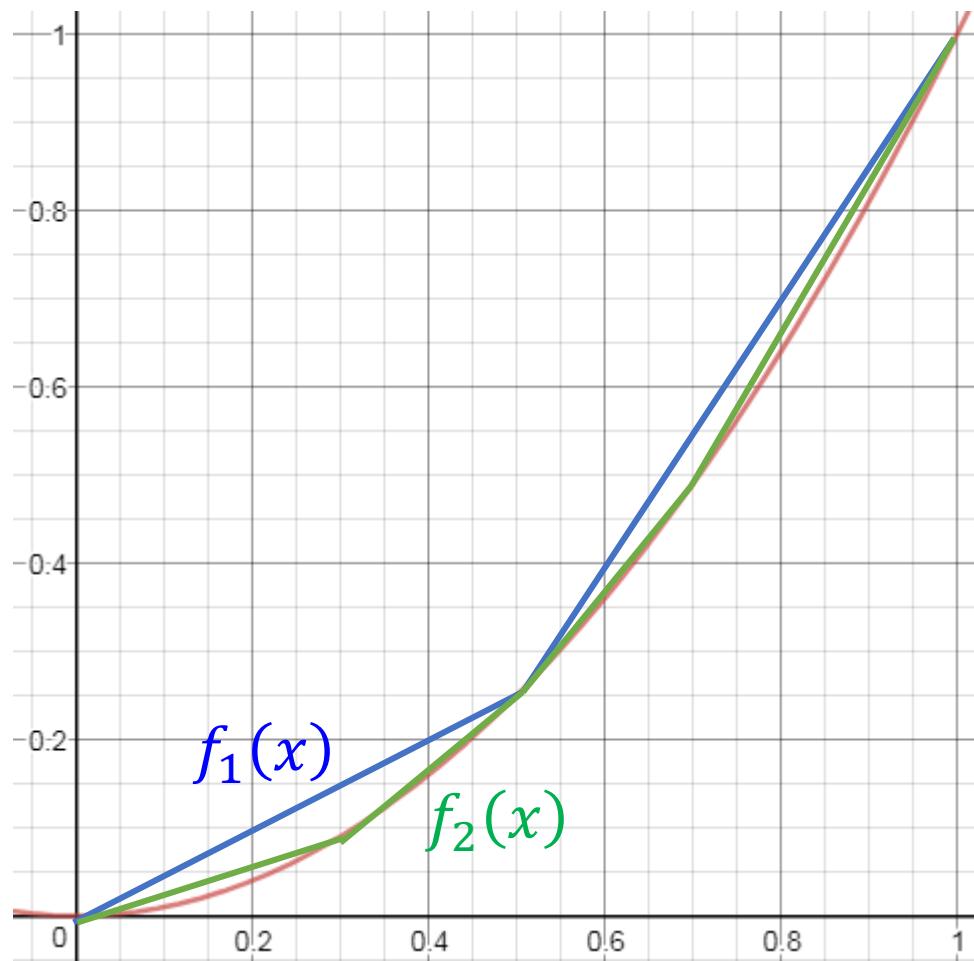


头条号 / 物理宝典



How much is deep
better than shallow?

$$f(x) = x^2$$



Fit the function by equally spaced linear pieces

$f_m(x)$: a function with 2^m pieces

$$\max_{0 \leq x \leq 1} |f(x) - f_m(x)| \leq \varepsilon$$

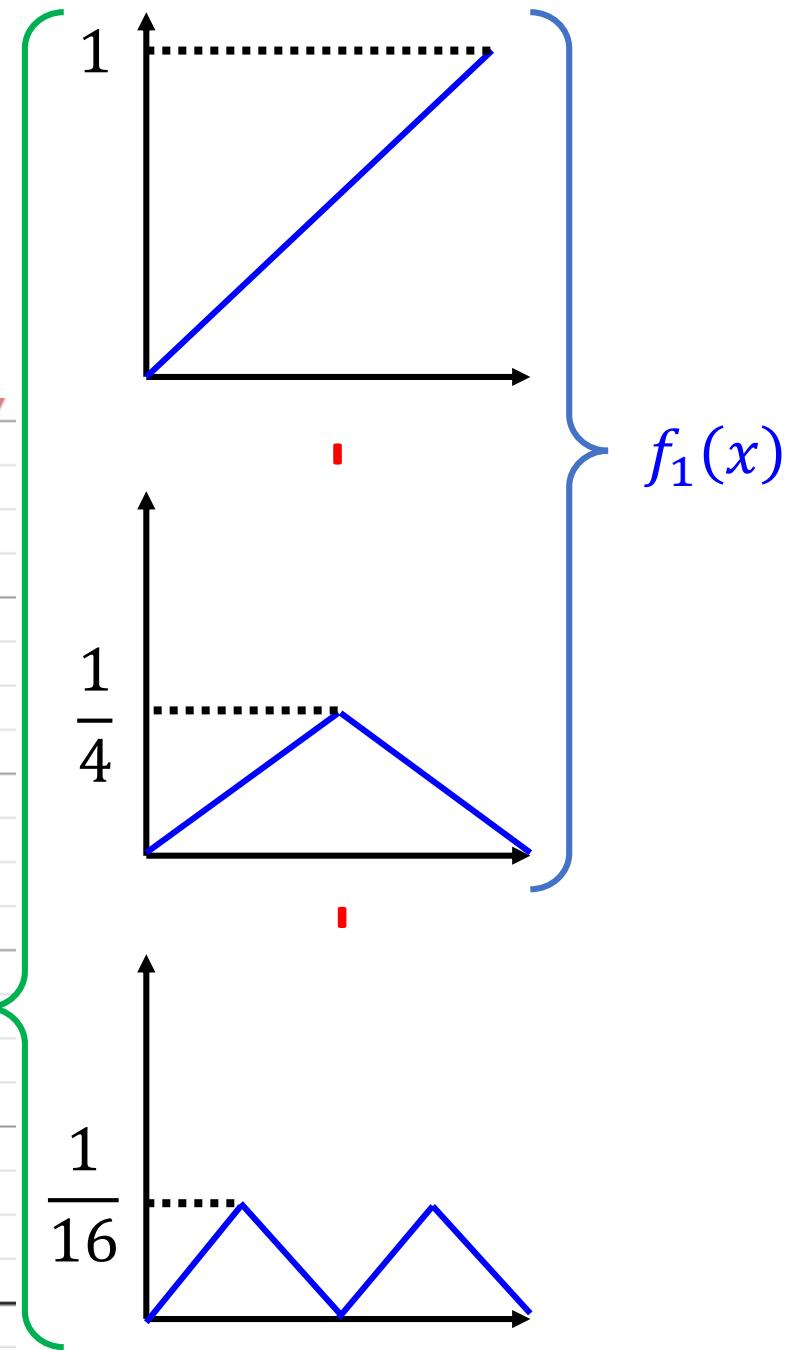
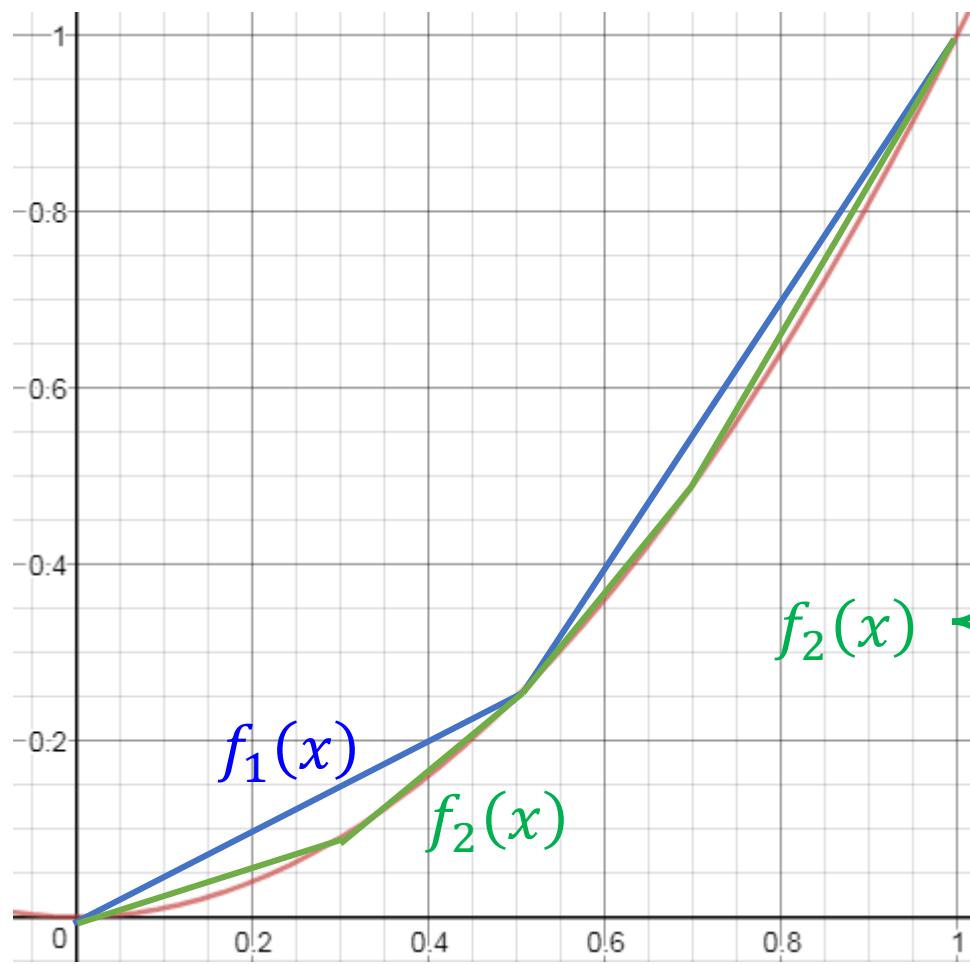
What is the minimum m ?

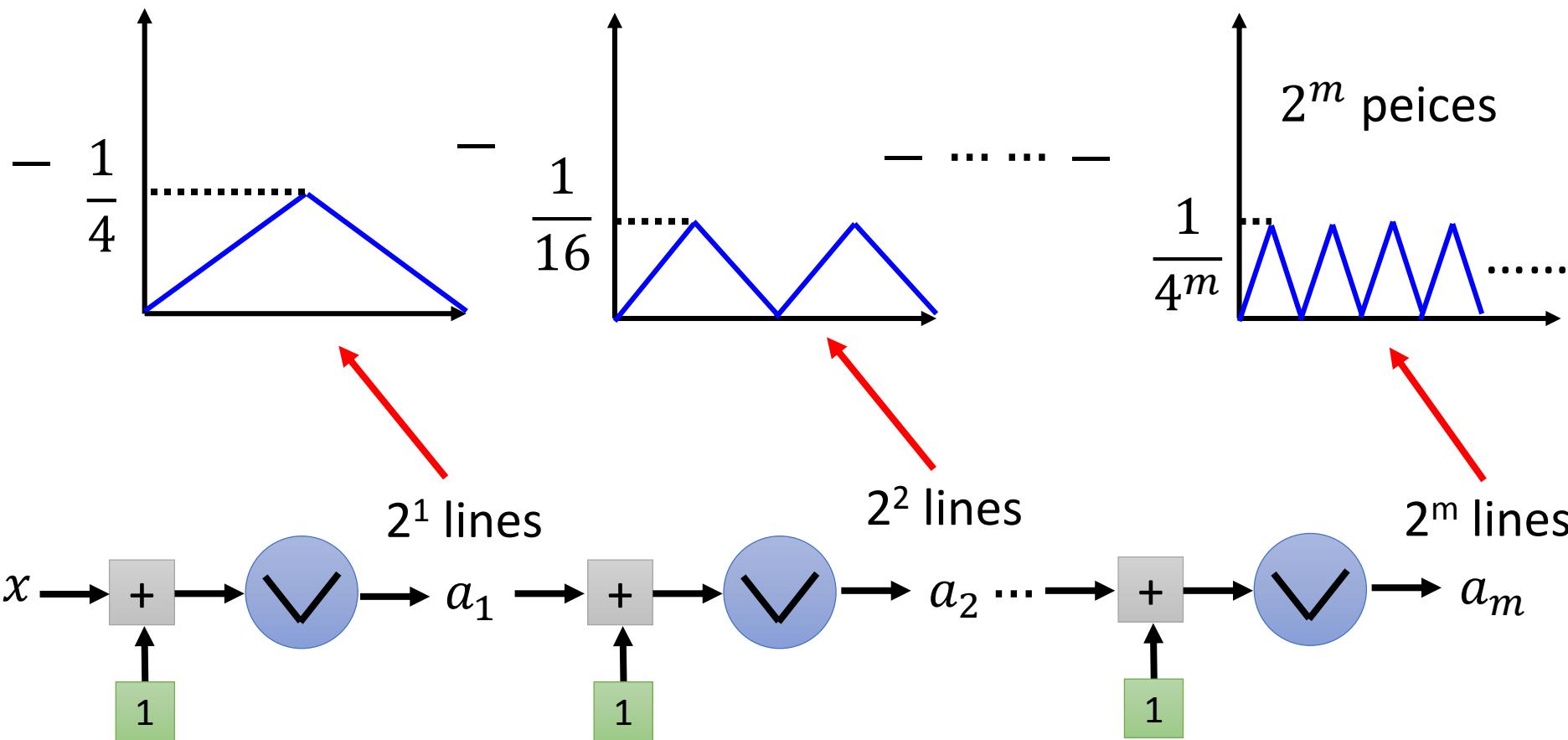
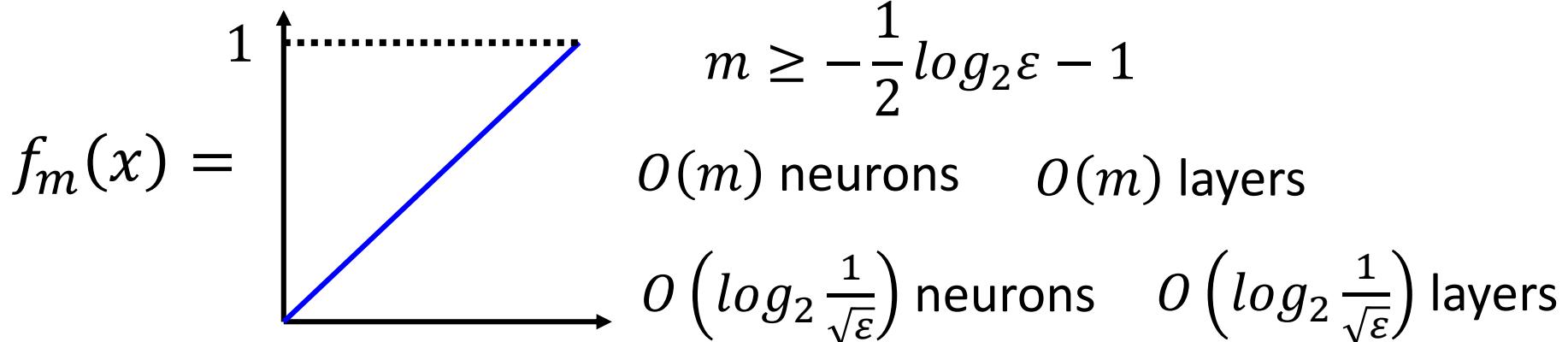
$$m \geq -\frac{1}{2} \log_2 \varepsilon - 1$$

$$2^m \geq \frac{1}{2} \frac{1}{\sqrt{\varepsilon}} \text{ pieces}$$

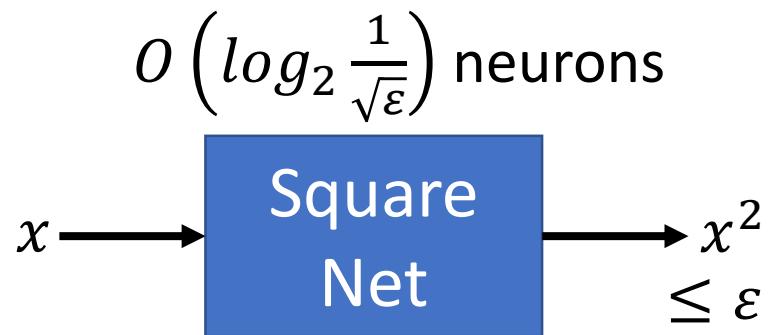
Shallow: $O\left(\frac{1}{\sqrt{\varepsilon}}\right)$ neurons

$$f(x) = x^2$$



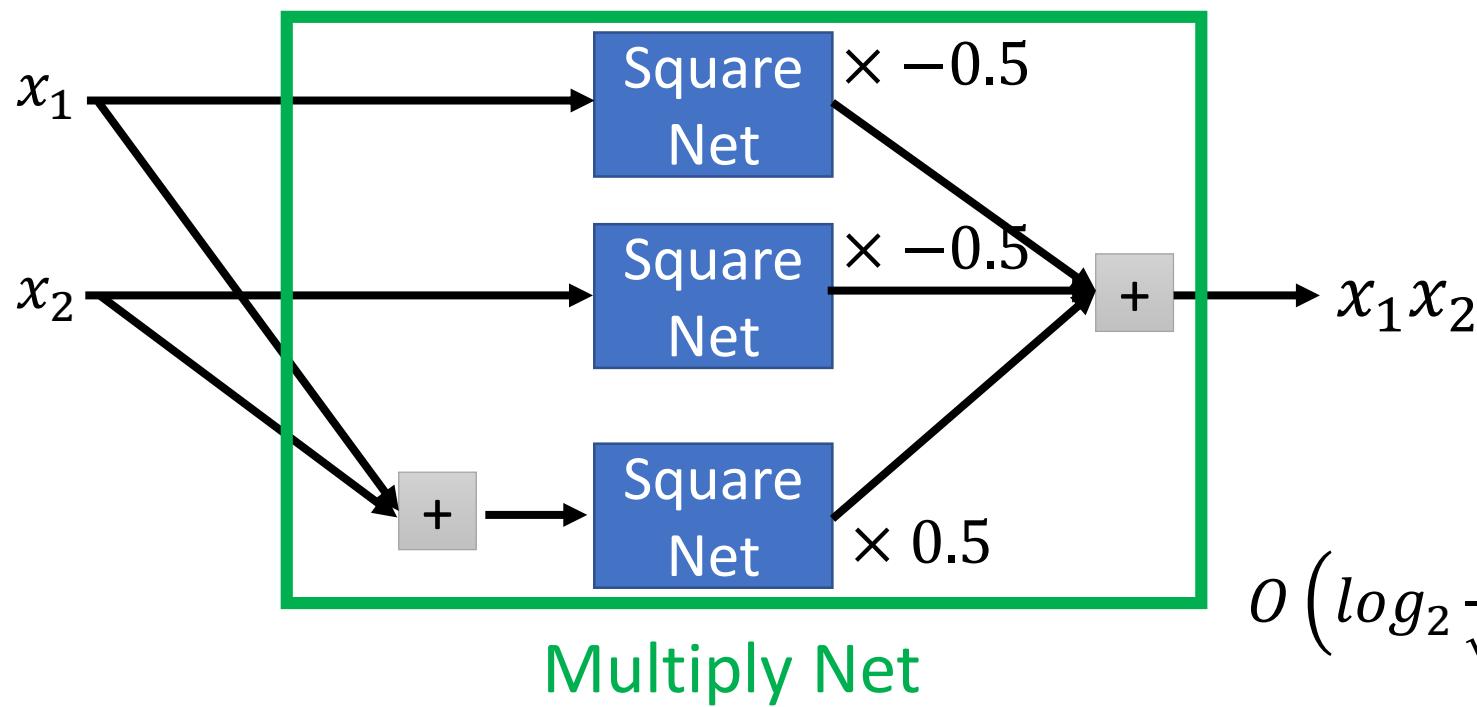


Why care about $y = x^2$?



$$y = x_1 x_2$$

$$= \frac{1}{2} \left((x_1 + x_2)^2 - x_1^2 - x_2^2 \right)$$



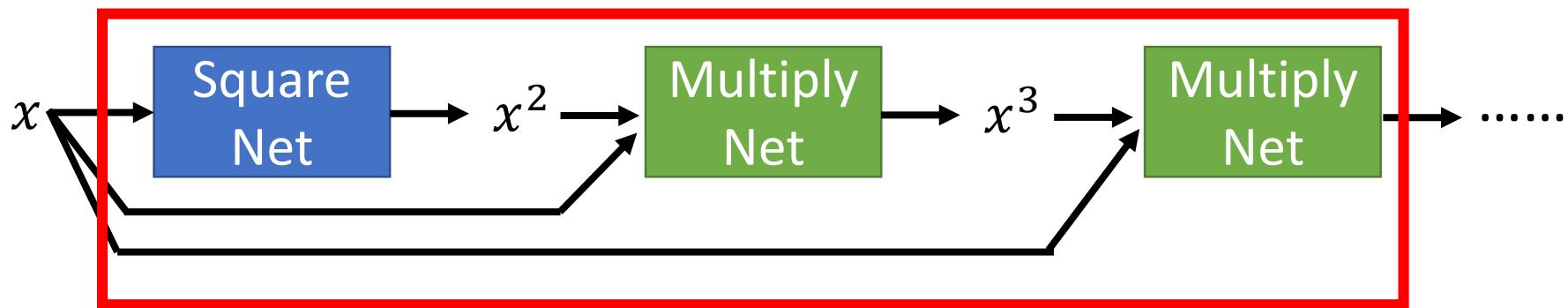
$O\left(\log_2 \frac{1}{\sqrt{\varepsilon}}\right)$ neurons

Polynomial

$$y = x^n$$

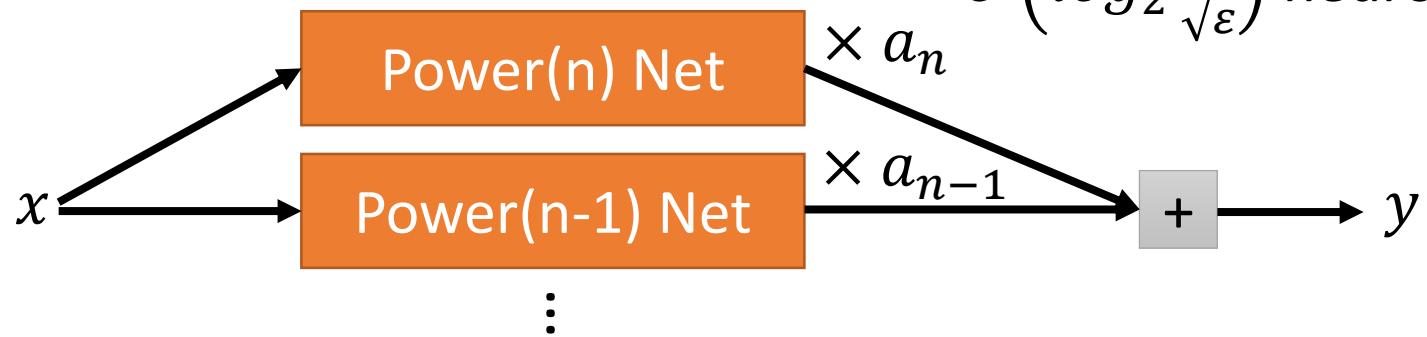
Power(n) Net

$O\left(\log_2 \frac{1}{\sqrt{\varepsilon}}\right)$ neurons



$$y = a_n x^n + a_{n-1} x^{n-1} + \dots + a_0$$

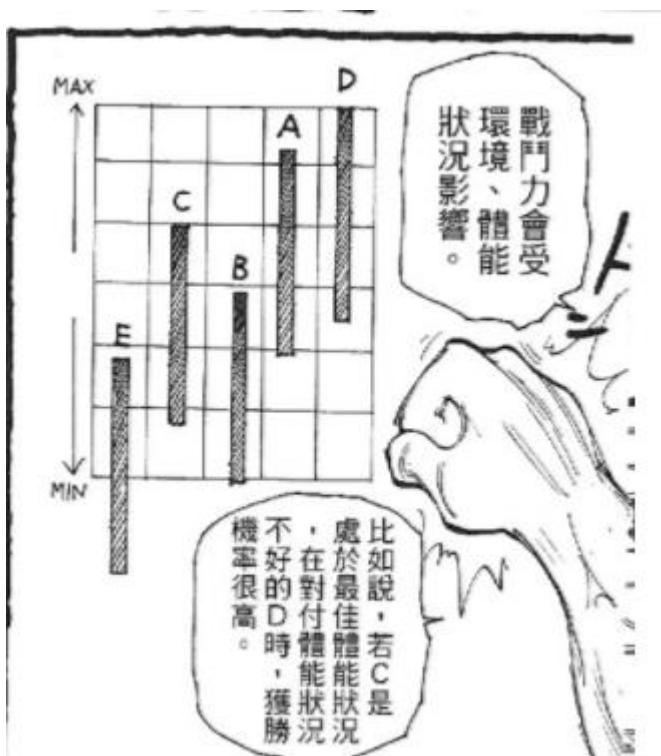
$O\left(\log_2 \frac{1}{\sqrt{\varepsilon}}\right)$ neurons



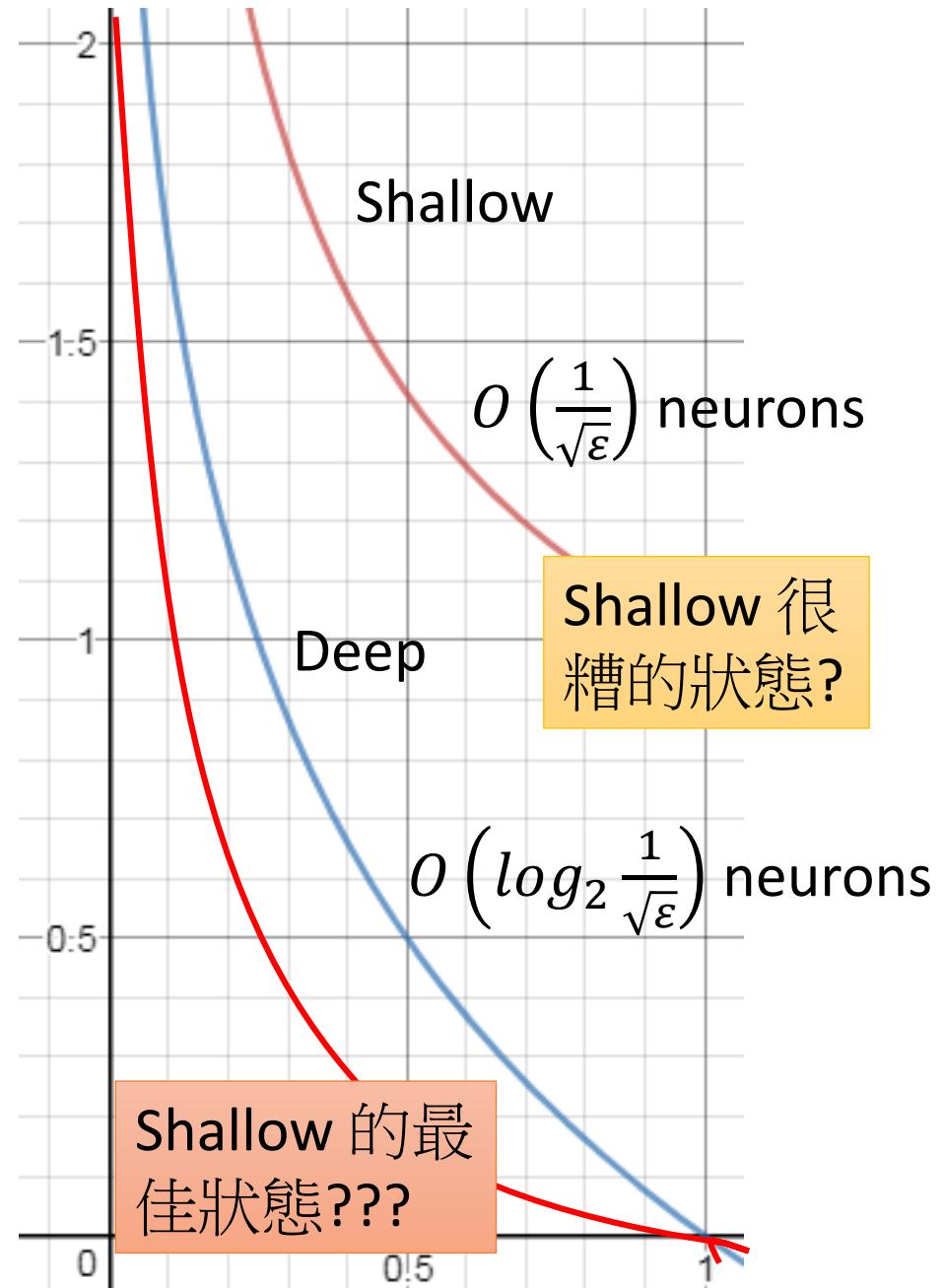
Use polynomial function to fit other functions.

Deep v.s. Shallow

This is not sufficient to show the power of deep.



(獵人第二十卷)



Is Deep better
than Shallow?

Best of Shallow

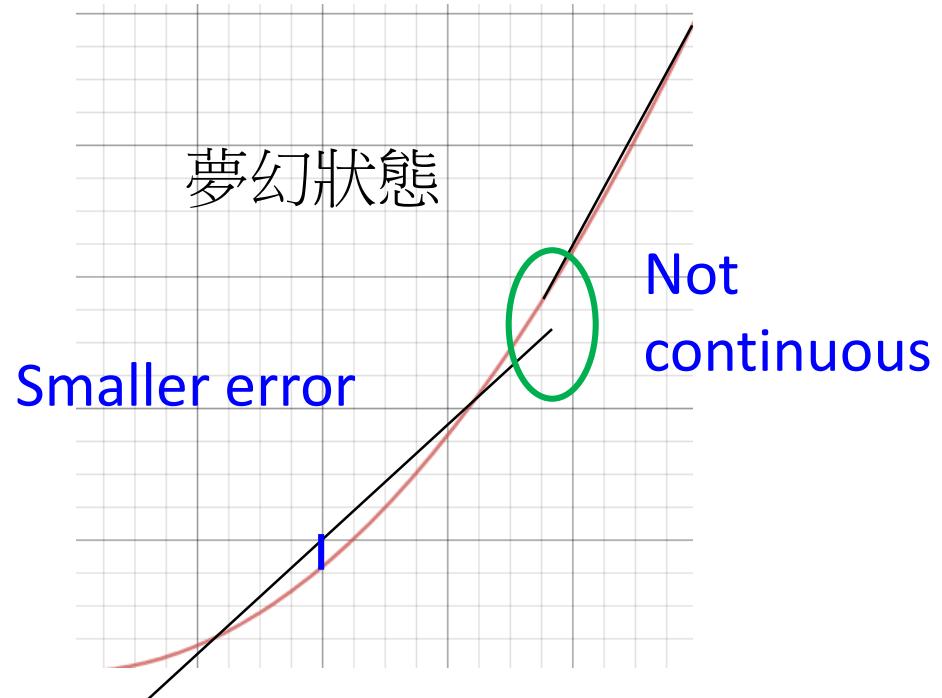
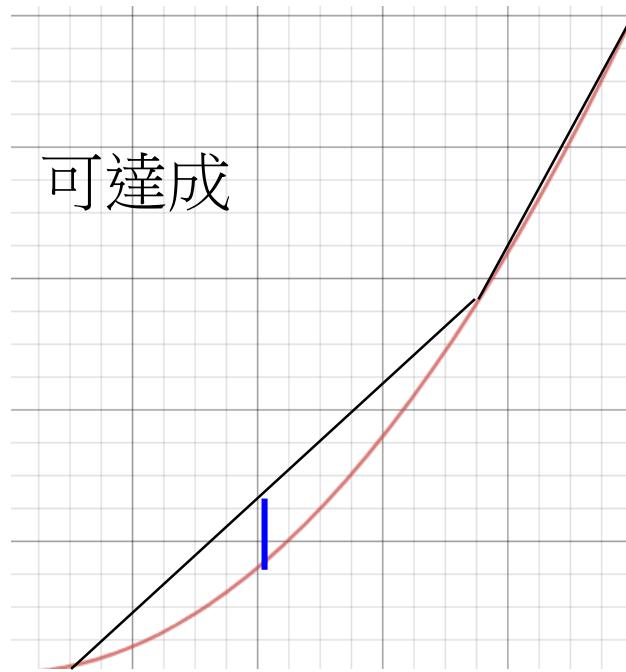
$$\max_{0 \leq x \leq 1} |f(x) - f^*(x)| \leq \varepsilon$$

↓

$$\sqrt{\int_0^1 |f(x) - f^*(x)|^2 dx} \leq \varepsilon$$

Use Euclidean

- A relu network is a piecewise linear function.
- Using the least pieces to fit the target function.



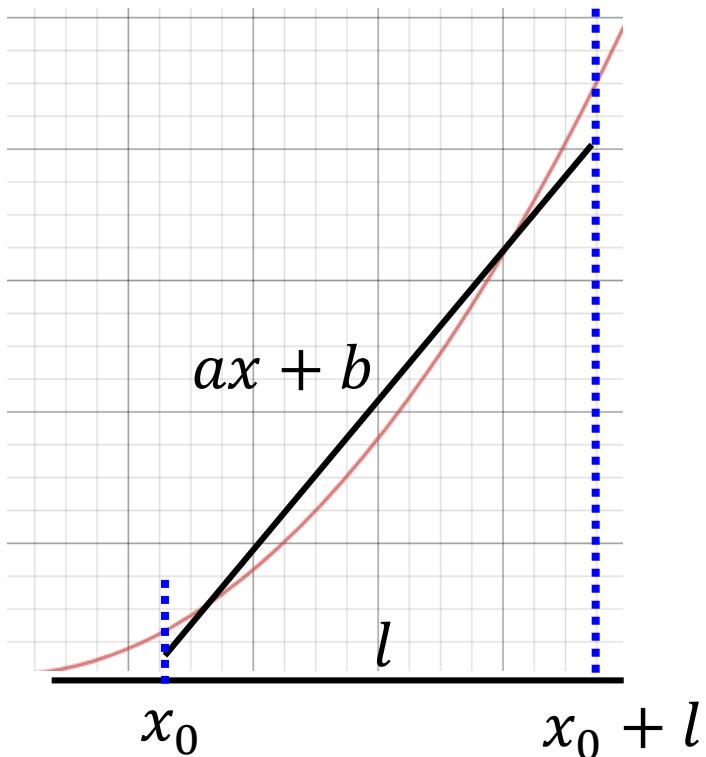
The lines do not have to connect the end points.

Best of Shallow

$$\sqrt{\int_0^1 |f(x) - f^*(x)|^2 dx} \leq \varepsilon$$

Use Euclidean

- Given a piece, what is the smallest error



$$e^2 = \int_{x_0}^{x_0+l} (x^2 - (ax + b))^2 dx$$

Find a and b to minimize e^2

The minimum value of e^2 is $\frac{l^5}{180}$

Warning of Math

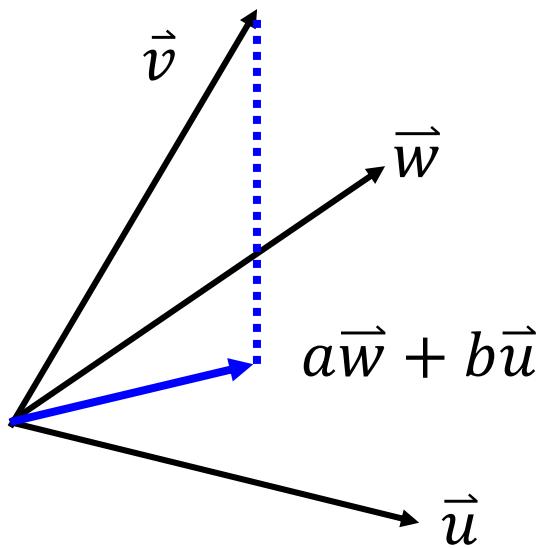
Intuition

$$e^2 = \int_{x_0}^{x_0+l} (x^2 - (ax + b))^2 dx$$

$$f_v = x^2 \quad f_w = x \quad f_u = 1$$

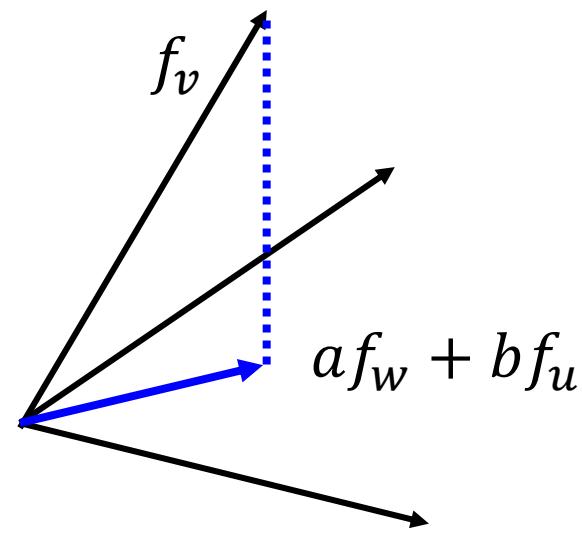
Minimize

$$\|\vec{v} - (a\vec{w} + b\vec{u})\|^2$$



Minimize

$$\|f_v - (af_w + bf_u)\|^2$$



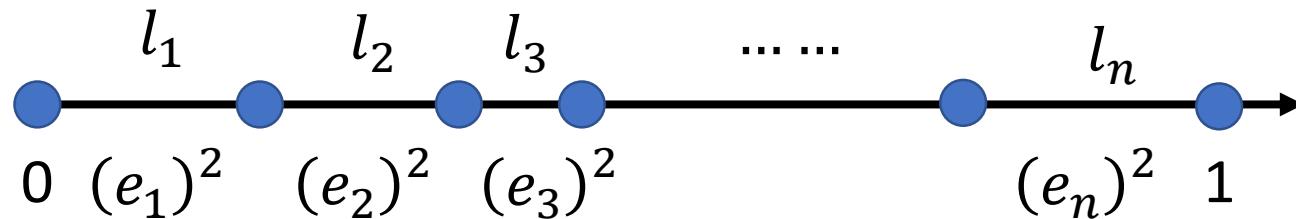
End of Warning

Best of Shallow

The minimum value of e^2 is $\frac{l^5}{180}$

- If you have n pieces, what is the best way to arrange the n pieces.

$$\sum_{i=1}^n l_i = 1$$



$$E^2 = \sum_{i=1}^n (e_i)^2 = \sum_{i=1}^n \frac{(l_i)^5}{180}$$

$$l_i = 1/n$$

The best way is “equal segment”

$$E^2 = \sum_{i=1}^n \frac{(1/n)^5}{180} = \frac{1}{180} \frac{1}{n^4}$$

Warning of Math

Hölder's inequality

$$\sum_{i=1}^n l_i = 1$$

$$\text{Minimize } \sum_{i=1}^n (l_i)^5$$

- Given $\{a_1, a_2, \dots, a_n\}$ and $\{b_1, b_2, \dots, b_n\}$

$$\sum_{i=1}^n |a_i b_i| \leq \left(\sum_{i=1}^n |a_i|^p \right)^{1/p} \left(\sum_{i=1}^n |b_i|^q \right)^{1/q} \quad \frac{1}{p} + \frac{1}{q} = 1$$

$$1 + \frac{p}{q} = p \quad 1 - p = -\frac{p}{q}$$

- Given $\{l_1, l_2, \dots, l_n\}$ and $\{1, 1, \dots, 1\}$

$$\boxed{\sum_{i=1}^n l_i} \leq \left(\sum_{i=1}^n l_i^p \right)^{1/p} \left(\sum_{i=1}^n 1^q \right)^{1/q}$$

$= 1$

$= n$

$$n^{-1/q} \leq \left(\sum_{i=1}^n l_i^p \right)^{1/p}$$

$$n^{1-p} \leq \sum_{i=1}^n l_i^p \quad \text{p=5}$$

$$n^{-4} \leq \sum_{i=1}^n l_i^5$$

End of Warning

Best of Shallow

The minimum value of e^2 is $\frac{l^5}{180}$

- If you have n pieces, what is the best way to arrange the n pieces.

$$E^2 = \frac{1}{180} \frac{1}{n^4} \rightarrow E = \sqrt{\frac{1}{180} \frac{1}{n^2}}$$

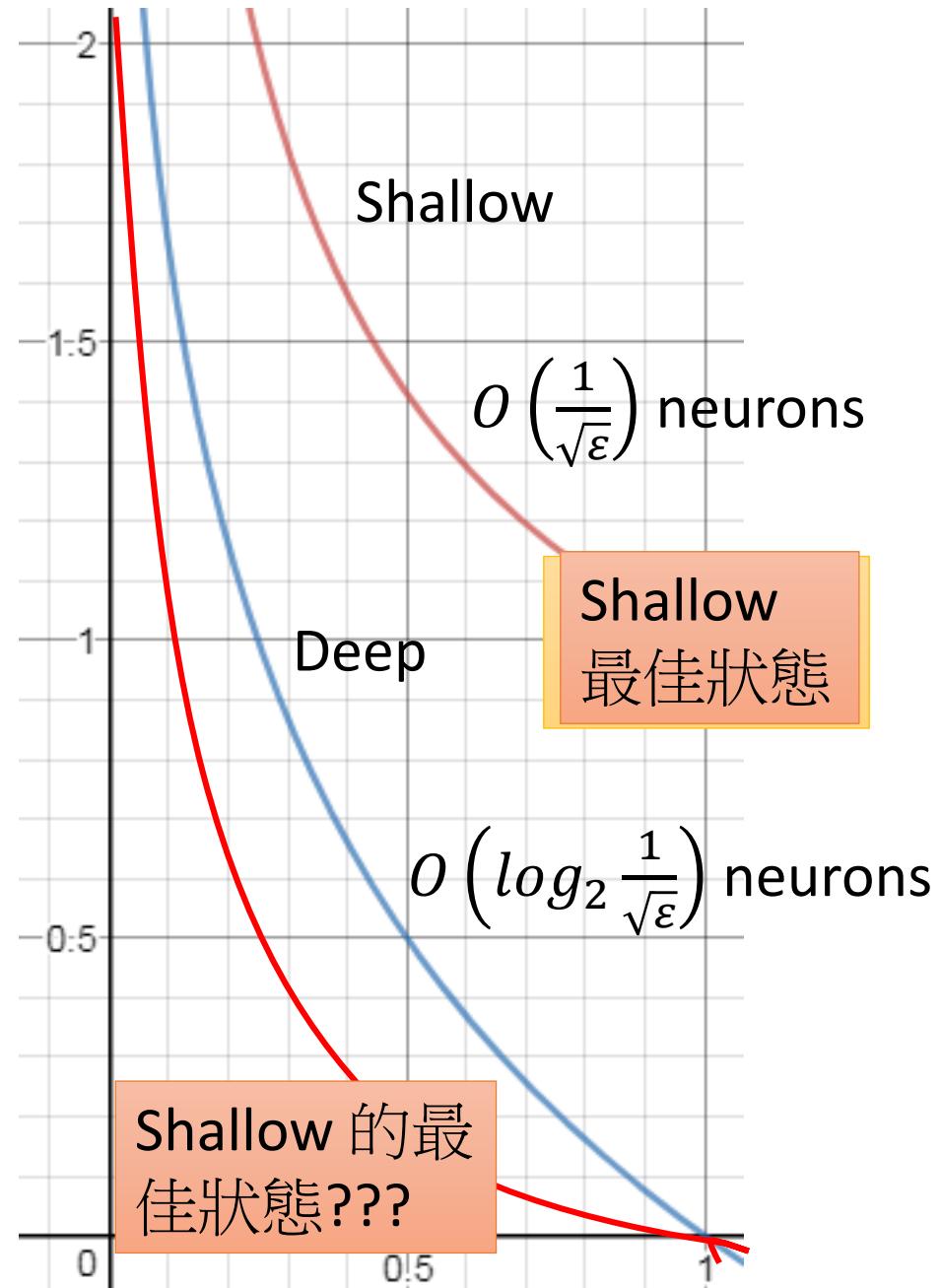
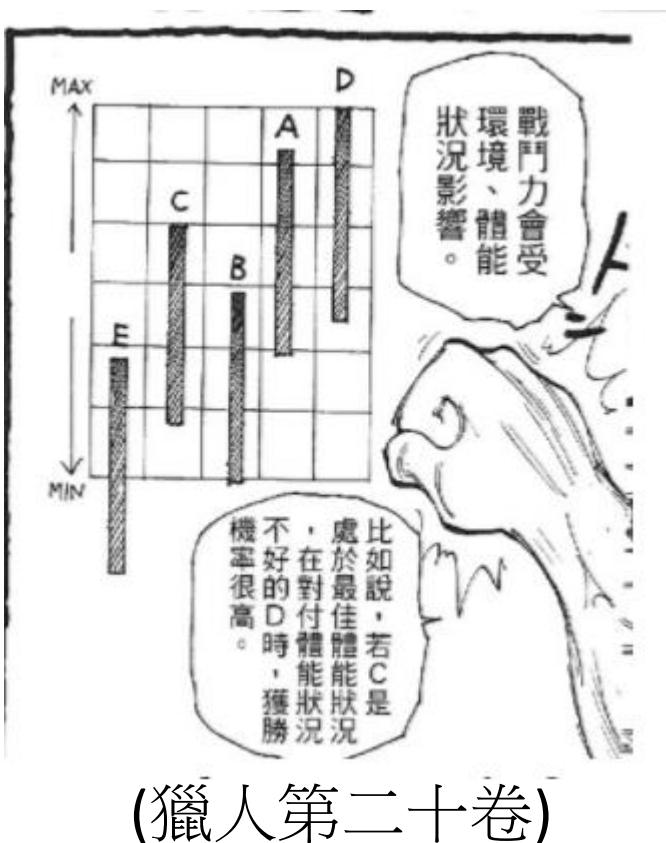
To make $E \leq \varepsilon$, what is the n we need?

$$E = \sqrt{\frac{1}{180} \frac{1}{n^2}} \leq \varepsilon \quad n^2 \geq \sqrt{\frac{1}{180} \frac{1}{\varepsilon}} \quad n \geq \sqrt[4]{\frac{1}{180}} \sqrt{\frac{1}{\varepsilon}}$$

At least $O\left(\frac{1}{\sqrt{\varepsilon}}\right)$ neurons

Deep v.s. Shallow

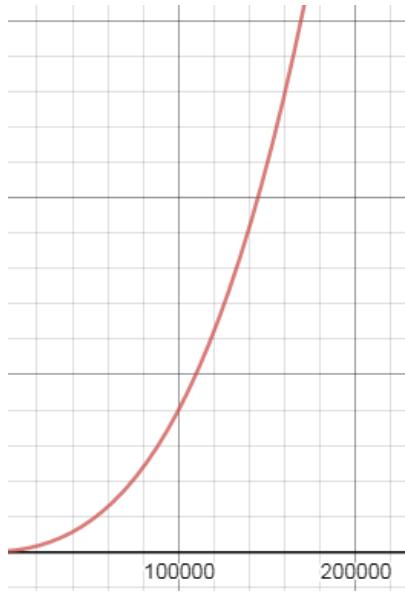
Deep is exponentially better than shallow.



More related theories

More Theories

- A function expressible by a 3-layer feedforward network cannot be approximated by 2-layer network.
 - Unless the width of 2-layer network is VERY large
 - Applied on activation functions beyond relu



The width of 3-layer network is K.

The width of 2-layer network
should be $Ae^{BK^{4/19}}$.

Ronen Eldan, Ohad Shamir, "The Power of Depth for Feedforward Neural Networks", COLT, 2016

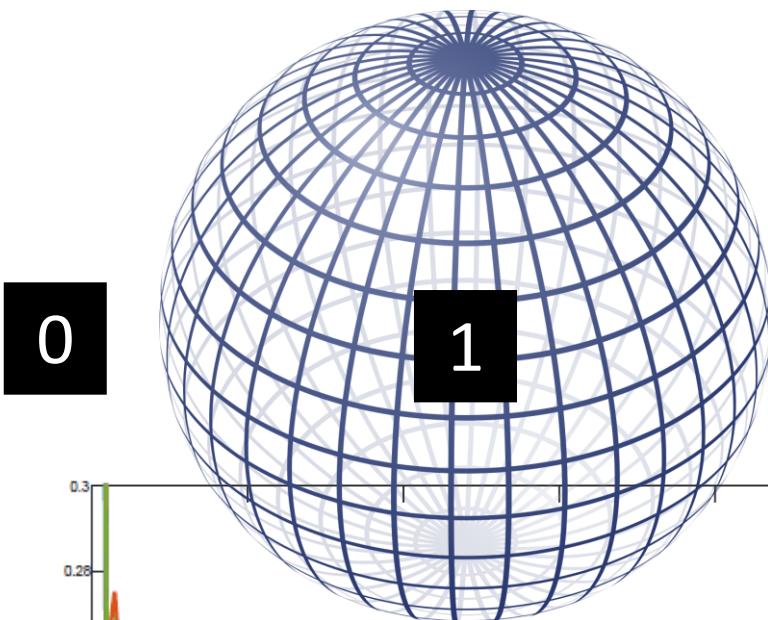
More Theories

- A function expressible by a deep feedforward network cannot be approximated by a shallow network.
 - Unless the width of the shallow network is VERY large
 - Applied on activation functions beyond relu

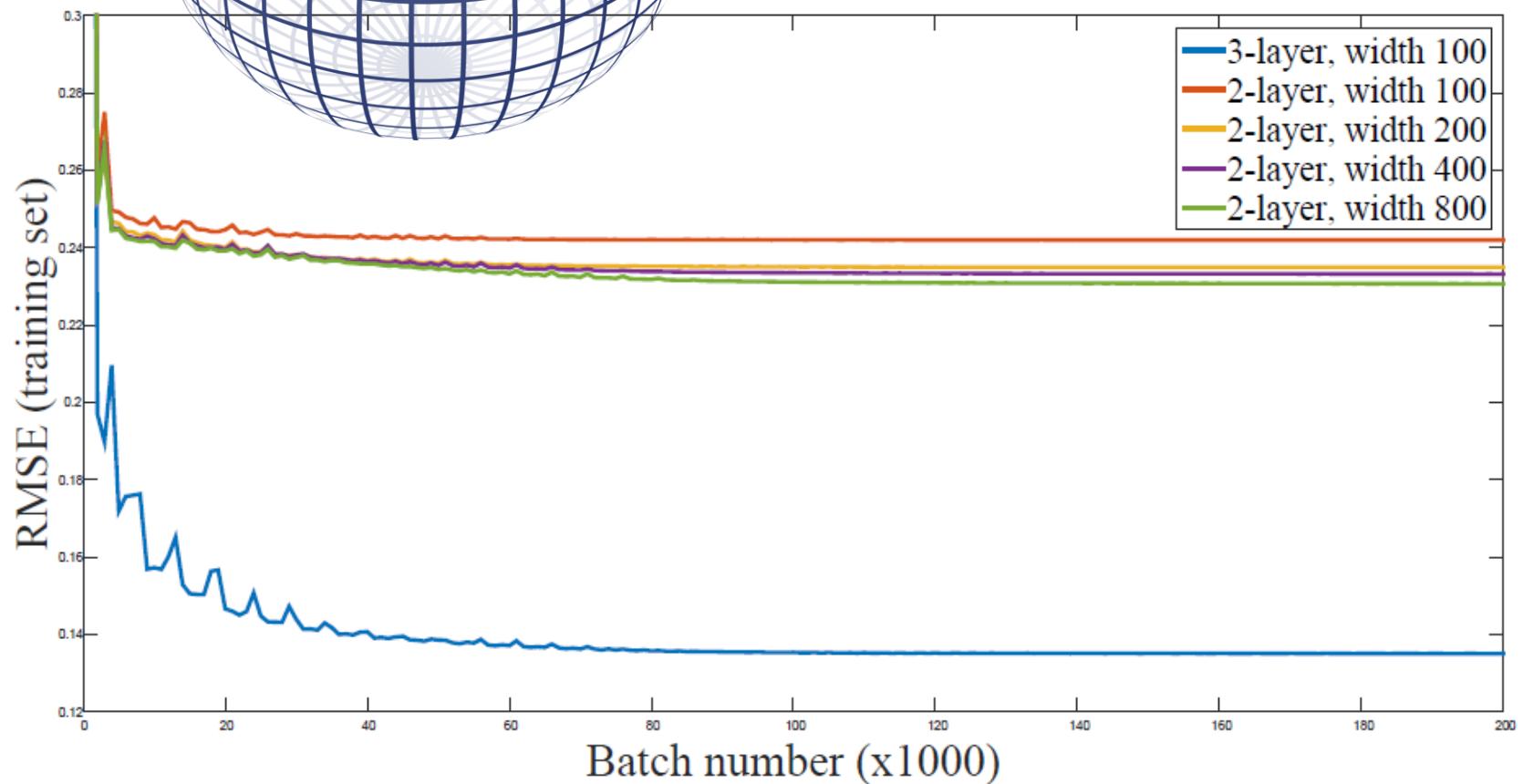
Deep Network:

$\Theta(k^3)$ layers, $\Theta(1)$ nodes per layer, $\Theta(1)$ distinct parameters

Shallow Network: $\Theta(k)$ layers  $\Omega(2^k)$ nodes



Itay Safran, Ohad Shamir, "Depth-Width Tradeoffs in Approximating Natural Functions with Neural Networks", ICML, 2017



More Theories

Dmitry Yarotsky, “Error bounds for approximations with deep ReLU networks”, arXiv, 2016

Dmitry Yarotsky, “Optimal approximation of continuous functions by very deep ReLU networks”, arXiv 2018

Shiyu Liang, R. Srikant, “Why Deep Neural Networks for Function Approximation?”, ICLR, 2017

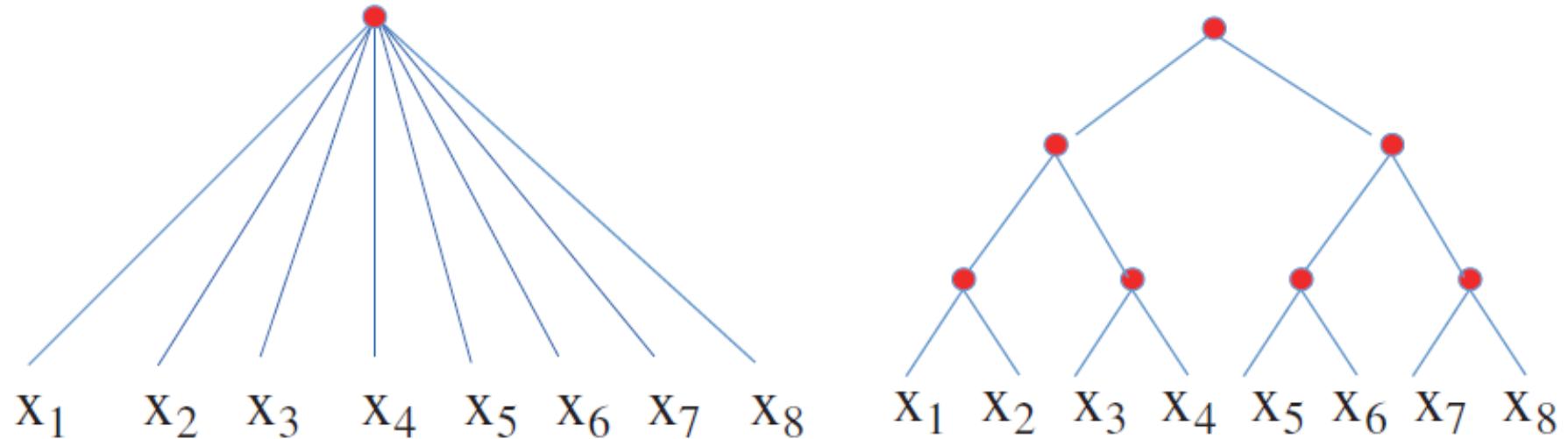
Itay Safran, Ohad Shamir, “Depth-Width Tradeoffs in Approximating Natural Functions with Neural Networks”, ICML, 2017

If a function f has “certain degree of complexity”

Approximating f to accuracy ε in the L2 norm using a fixed depth ReLU network requires at least $\text{poly}(1/\varepsilon)$

There exist a ReLU network of depth and width at most $\text{poly}(\log(1/\varepsilon))$ that can achieve the approximation.

The Nature of Functions



Hrushikesh Mhaskar, Qianli Liao, Tomaso Poggio, When and Why Are Deep Networks Better Than Shallow Ones?, AAAI, 2017

Concluding Remarks

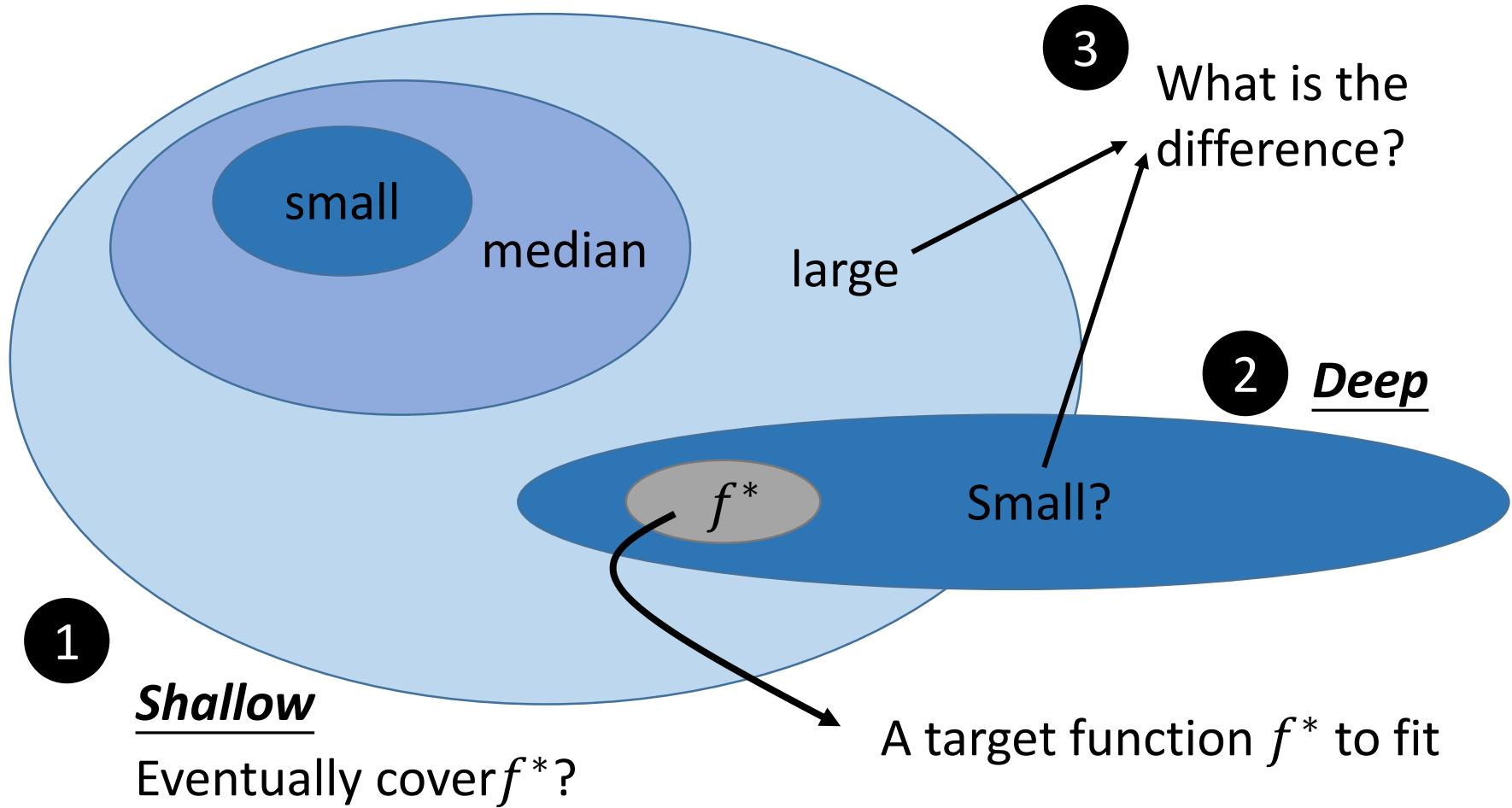
Optimization

李宏毅

Hung-yi Lee

Last time ...

Optimization: Is it possible to find f^* in the function space.



Optimization

Optimization \neq Learning

Network: $f_\theta(x)$

Training data:

$$(x^1, \hat{y}^1)$$

$$(x^2, \hat{y}^2)$$

⋮

$$(x^R, \hat{y}^R)$$

$$L(\theta) = \sum_{r=1}^R l(f_\theta(x^r) - \hat{y}^r)$$

$$\theta^* = \arg \min_{\theta} L(\theta)$$

In Deep Learning, $L(\theta)$ is
not convex.

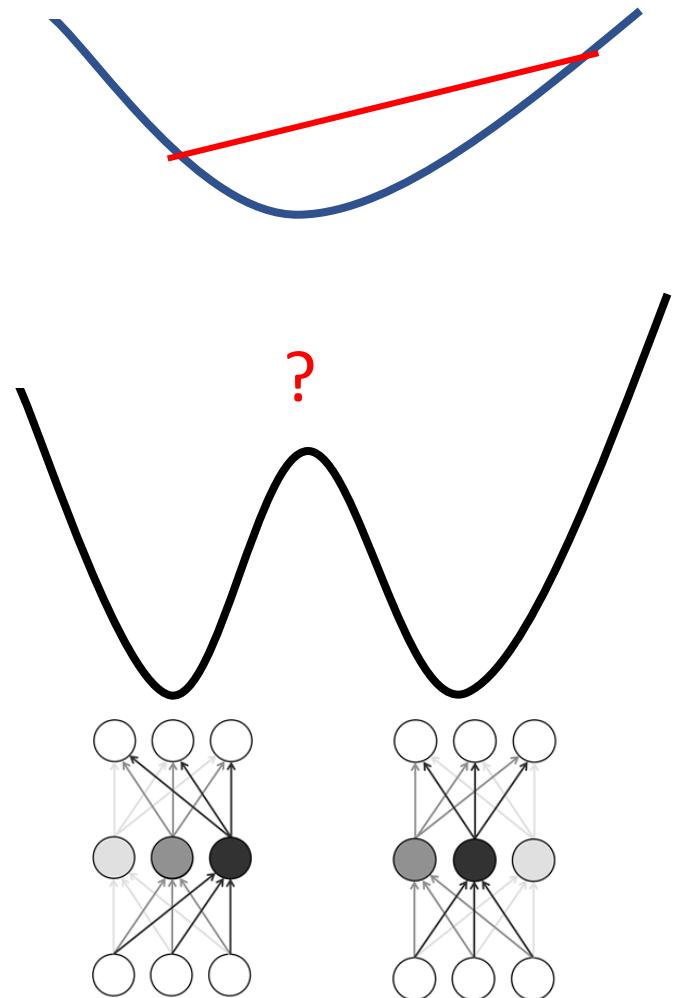
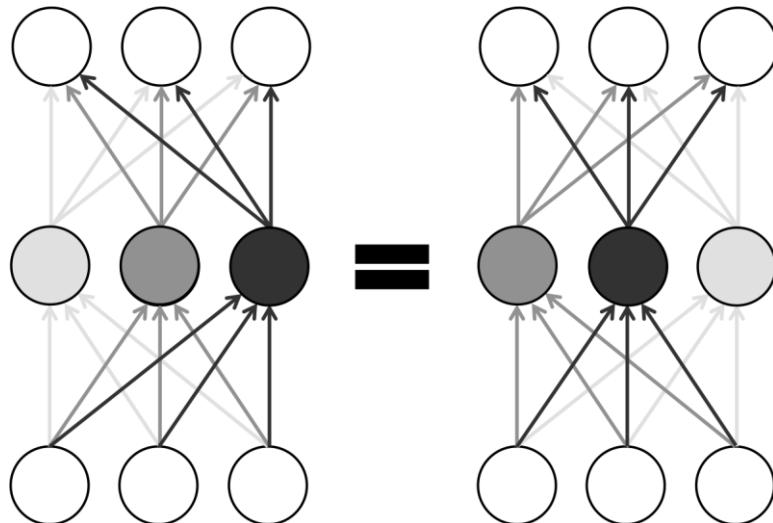
Non-convex optimization is NP-hard.

Why can we solve the problem by gradient descent?

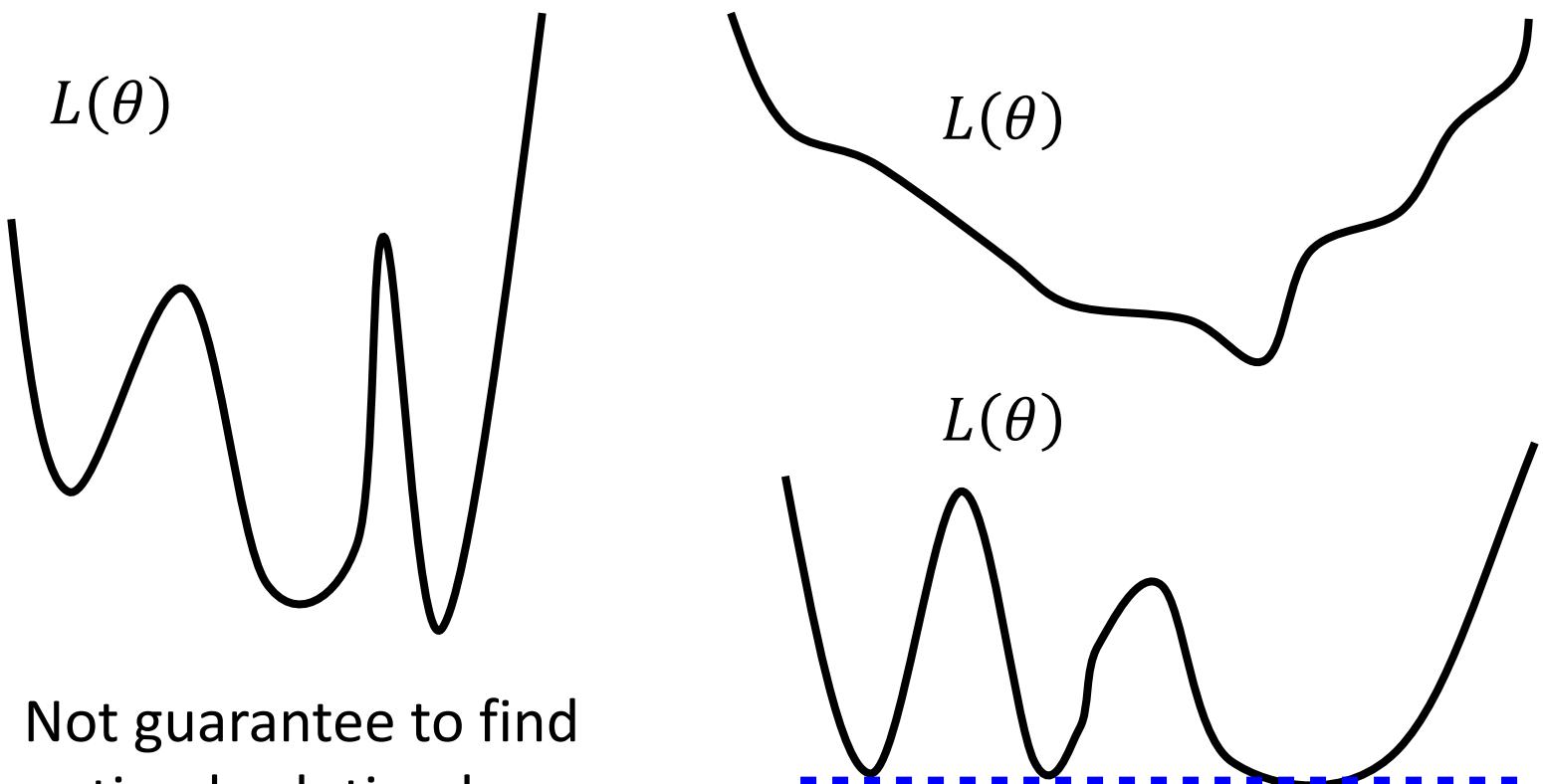
Loss of Deep Learning is not convex

There are at least exponentially many global minima for a neural net.

Permutating the neurons in one layer does not change the loss.



Non-convex \neq Difficult



Outline

Review: Hessian

Deep Linear Model

Deep Non-linear Model

Conjecture about Deep Learning

Empirical Observation about Error Surface

Hessian Matrix: When Gradient is Zero

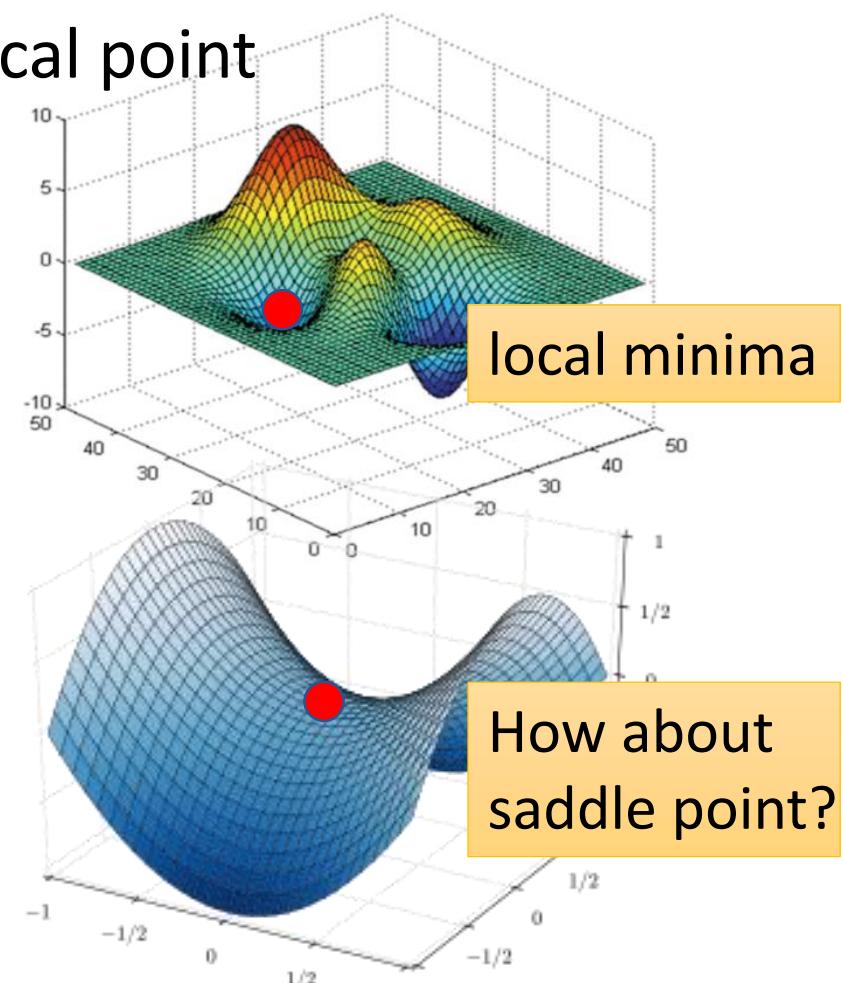
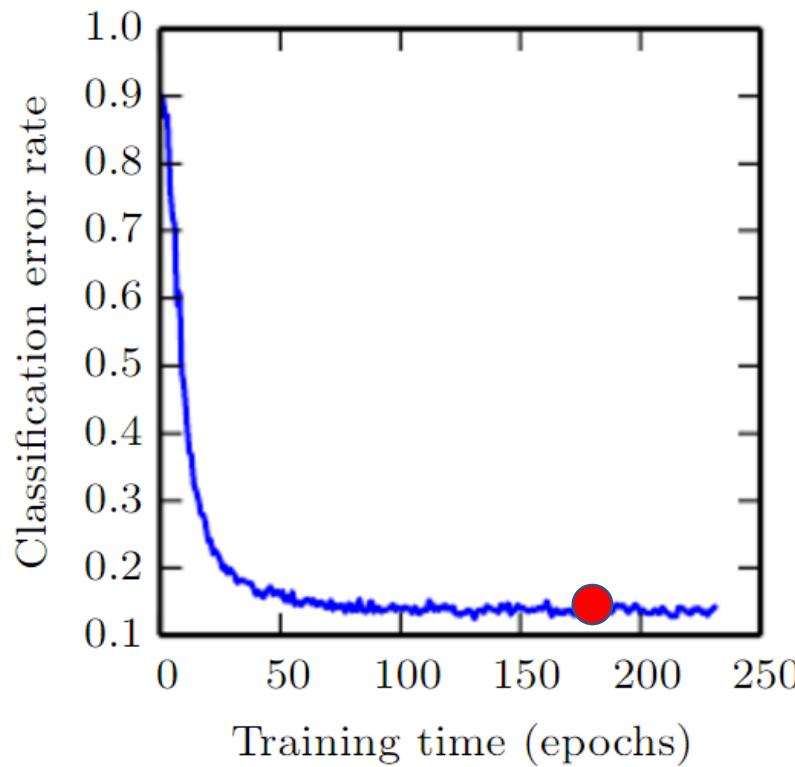
Some examples in this part are from:

[https://www.math.upenn.edu/~kazdan/312F12/Notes/
max-min-notesJan09/max-min.pdf](https://www.math.upenn.edu/~kazdan/312F12/Notes/max-min-notesJan09/max-min.pdf)

Training stops

critical point:
gradient is zero

- People believe training stuck because the parameters are near a critical point



When Gradient is Zero

$$f(\theta) = f(\theta^0) + (\theta - \theta^0)^T g + \frac{1}{2} (\theta - \theta^0)^T H (\theta - \theta^0) + \dots$$

Gradient g is a **vector**

$$g_i = \frac{\partial f(\theta^0)}{\partial \theta_i} \quad \nabla f(\theta^0)$$

Hessian H is a **matrix**

symmetric

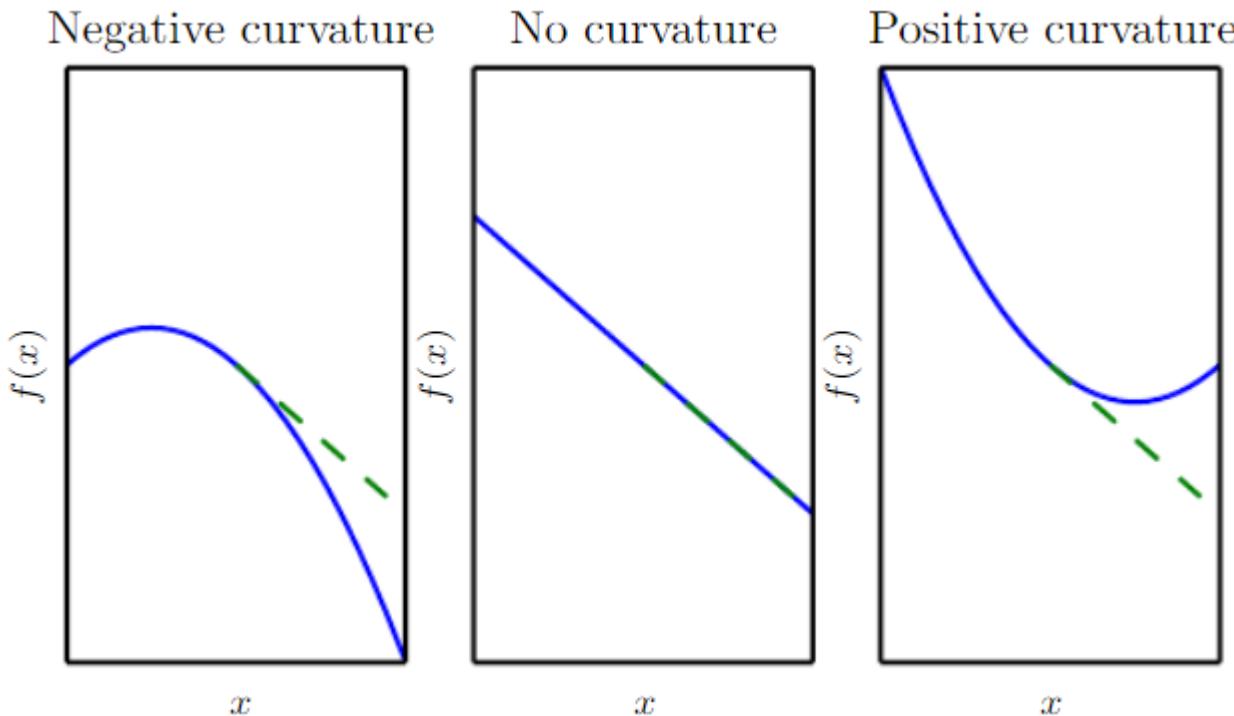
$$\begin{aligned} H_{ij} &= \frac{\partial^2}{\partial \theta_i \partial \theta_j} f(\theta^0) \\ &= \frac{\partial^2}{\partial \theta_j \partial \theta_i} f(\theta^0) = H_{ji} \end{aligned}$$

Hessian

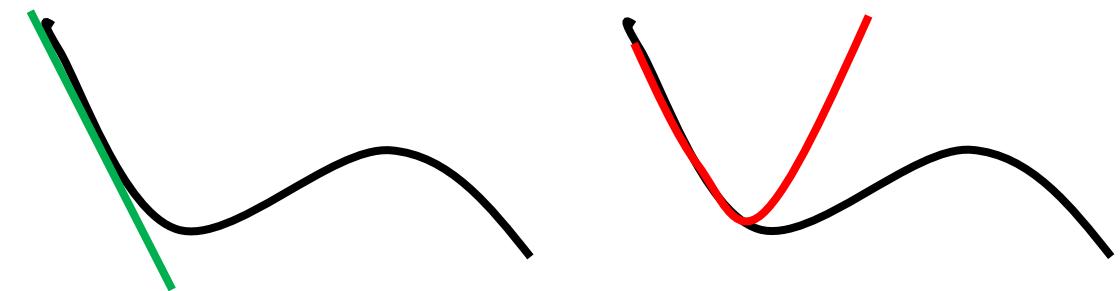
Source of image:
<http://www.deeplearningbook.org/contents/numerical.html>

$$f(\theta) = f(\theta^0) + (\theta - \theta^0)^T g + \frac{1}{2} (\theta - \theta^0)^T H(\theta - \theta^0) + \dots$$

H determines the curvature



Hessian



$$f(\theta) = f(\theta^0) + (\theta - \theta^0)^T g + \frac{1}{2} (\theta - \theta^0)^T H(\theta - \theta^0) + \dots$$

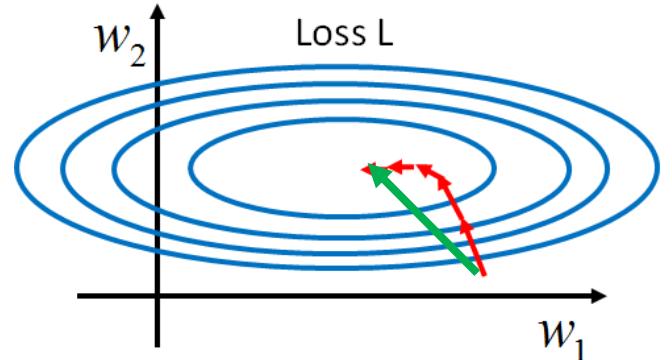
Newton's method  Find the space such that $\nabla f(\theta) = 0$

$$\begin{aligned}\nabla f(\theta) &\approx \underline{\nabla[(\theta - \theta^0)^T g]} + \underline{\nabla \left[\frac{1}{2} (\theta - \theta^0)^T H(\theta - \theta^0) \right]} \\ &= g \\ &\quad H(\theta - \theta^0)\end{aligned}$$

$$\frac{\partial [(\theta - \theta^0)^T g]}{\partial \theta_i} = g_i$$

$$\frac{\partial \left[\frac{1}{2} (\theta - \theta^0)^T H(\theta - \theta^0) \right]}{\partial \theta_i}$$

Hessian



$$f(\theta) = f(\theta^0) + (\theta - \theta^0)^T g + \frac{1}{2} (\theta - \theta^0)^T H(\theta - \theta^0) + \dots$$

Newton's method

$$\begin{aligned}\nabla f(\theta) &\approx \underline{\nabla[(\theta - \theta^0)^T g]} + \underline{\nabla \left[\frac{1}{2} (\theta - \theta^0)^T H(\theta - \theta^0) \right]} \\ &= g \qquad \qquad \qquad H(\theta - \theta^0)\end{aligned}$$

$$\nabla f(\theta) \approx g + H(\theta - \theta^0) = 0$$

$$H(\theta - \theta^0) = -g \qquad \qquad \theta = \theta^0 - \boxed{H^{-1}} g \quad \text{v.s.} \quad \theta = \theta^0 - \eta g$$

$$\theta - \theta^0 = -H^{-1}g \qquad \text{Change the direction, determine step size}$$

Hessian

Source of image:

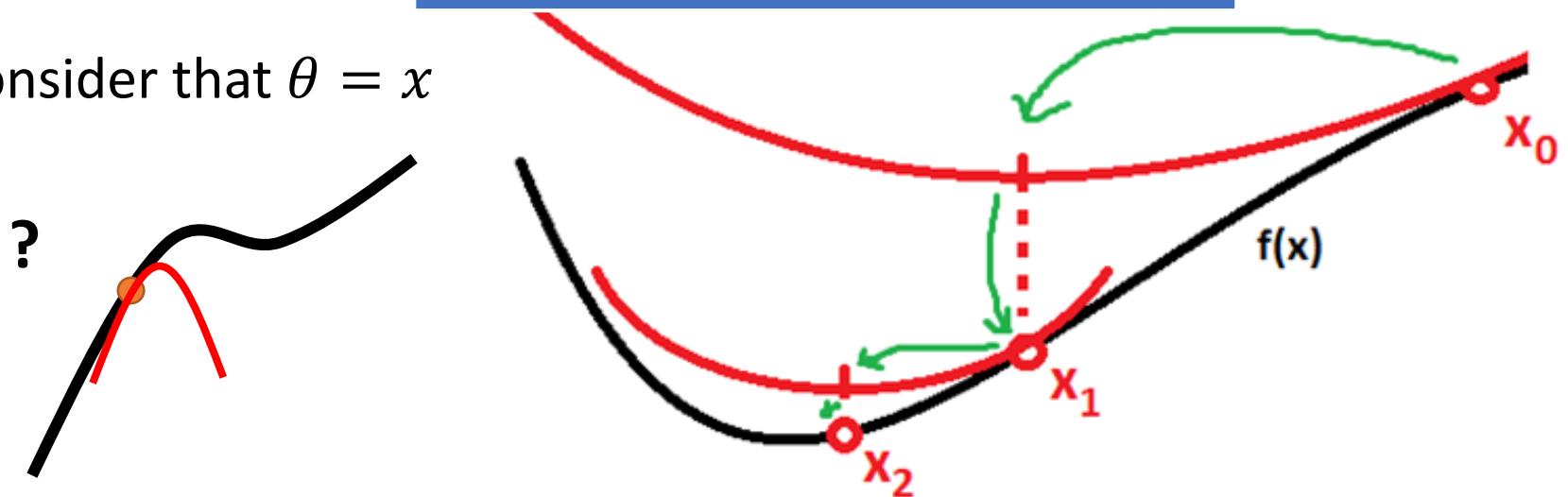
<https://math.stackexchange.com/questions/609680/newtons-method-intuition>

$$f(\theta) = f(\theta^0) + (\theta - \theta^0)^T g + \frac{1}{2} (\theta - \theta^0)^T H(\theta - \theta^0) + \dots$$

Newton's method

Not suitable for Deep Learning

Consider that $\theta = x$



If $f(x)$ is a quadratic function, obtain critical point in one step.

What is the problem?

Source of image:

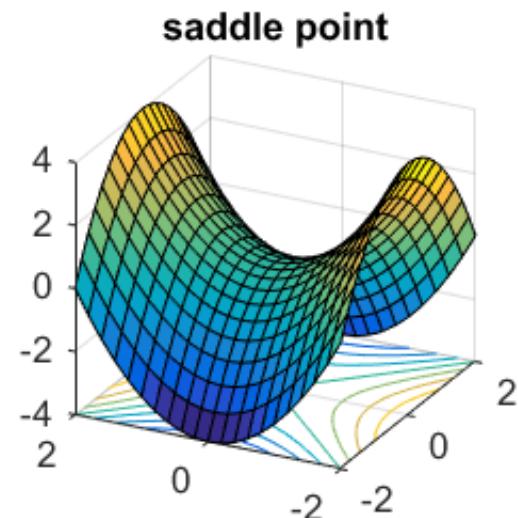
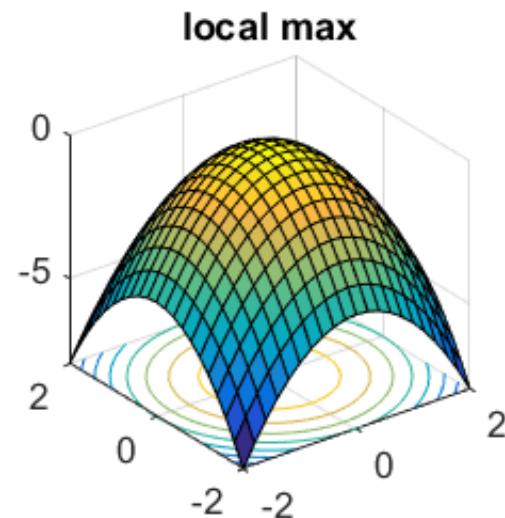
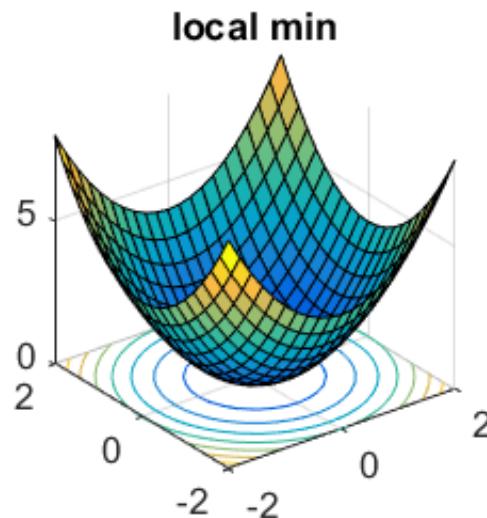
<http://www.offconvex.org/2016/03/22/saddlepoints/>

Hessian

$$f(\theta) = f(\theta^0) + \cancel{(\theta - \theta^0)^T g} + \frac{1}{2} (\theta - \theta^0)^T H(\theta - \theta^0) + \dots$$

At critical point ($g = 0$)

H tells us the properties of critical points.



Review: Linear Algebra

- If $A\boldsymbol{v} = \lambda\boldsymbol{v}$ (\boldsymbol{v} is a vector, λ is a scalar)
 - \boldsymbol{v} is an eigenvector of A **excluding zero vector**
 - λ is an eigenvalue of A that corresponds to \boldsymbol{v}

A must be square

$$\begin{bmatrix} 5 & 2 & 1 \\ -2 & 1 & -1 \\ 2 & 2 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ -4 \\ 4 \end{bmatrix} = 4 \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

Eigen value

Eigen vector

Review: Positive/Negative Definite

- An $n \times n$ matrix A is symmetric.
- For every non-zero vector x ($x \neq 0$)

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

positive definite: $x^T A x > 0 \leftrightarrow$

All eigen values
are positive.

positive semi-definite: $x^T A x \geq 0 \leftrightarrow$

All eigen values
are non-negative.

negative definite: $x^T A x < 0 \leftrightarrow$

All eigen values
are negative.

negative semi-definite: $x^T A x \leq 0 \leftrightarrow$

All eigen values
are non-positive.

Hessian

At critical point:

$$x^T H x$$

$$f(\theta) \approx f(\theta^0) + \frac{1}{2} (\theta - \theta^0)^T H (\theta - \theta^0)$$

H is positive definite $\rightarrow x^T H x > 0$ $x^T H x \geq 0?$

All eigen values are positive.

\rightarrow Around θ^0 : $f(\theta) > f(\theta^0)$ \rightarrow Local minima

H is negative definite $\rightarrow x^T H x < 0$ $x^T H x \leq 0?$

All eigen values are negative

\rightarrow Around θ^0 : $f(\theta) < f(\theta^0)$ \rightarrow Local maxima

Sometimes $x^T H x > 0$, sometimes $x^T H x < 0$

\rightarrow Saddle point

Hessian

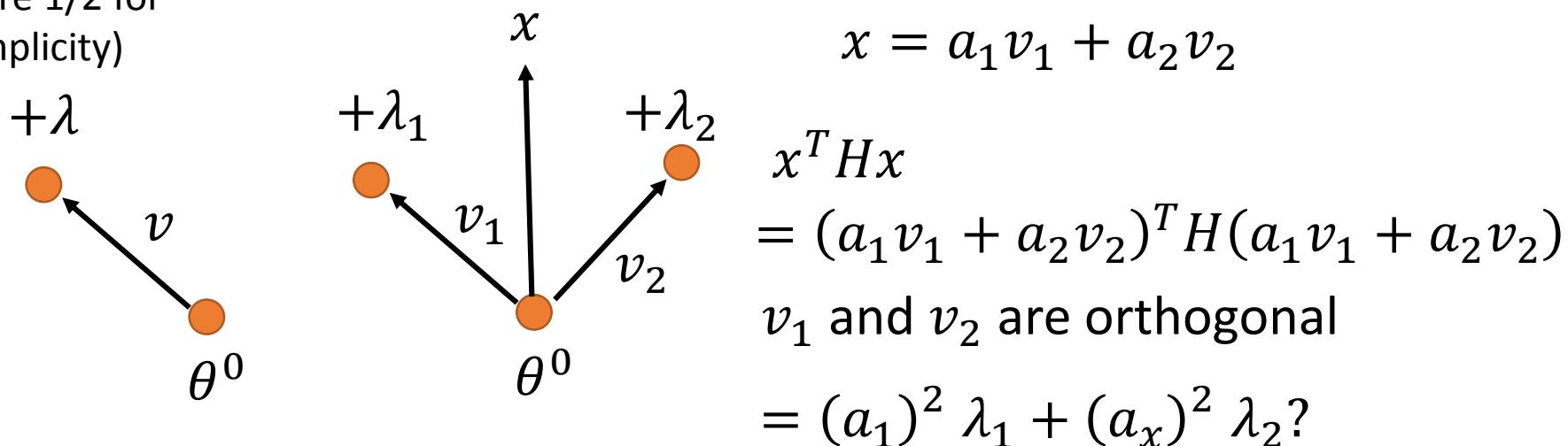
At critical point:

$$f(\theta) \approx f(\theta^0) + \frac{1}{2} (\theta - \theta^0)^T H (\theta - \theta^0)$$

v is an eigen vector $\rightarrow v^T H v = v^T (\lambda v) = \lambda \|v\|^2$

Unit vector

(Ignore 1/2 for simplicity)



Because H is an $n \times n$ symmetric matrix,

H can have eigen vectors $\{v_1, v_2, \dots, v_n\}$ form a orthonormal basis.

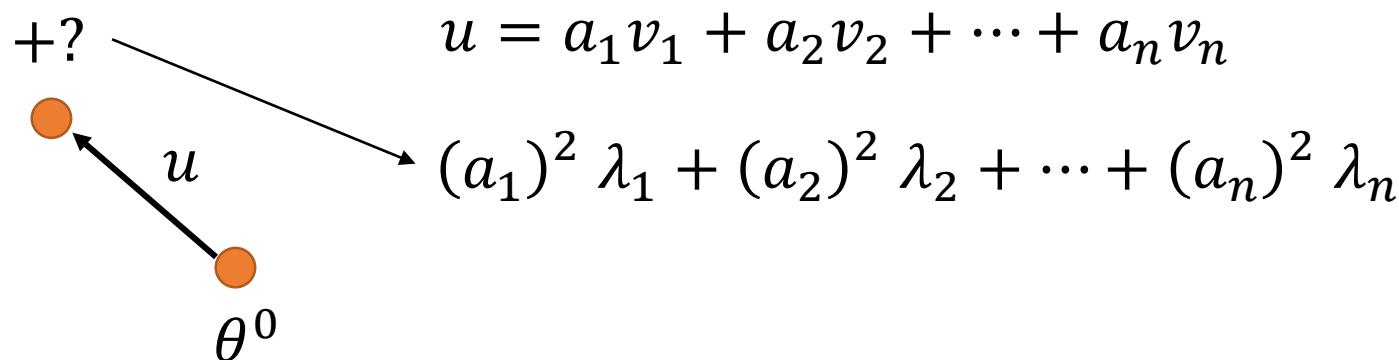
Hessian

At critical point:

$$f(\theta) \approx f(\theta^0) + \frac{1}{2} (\theta - \theta^0)^T H (\theta - \theta^0)$$

v is an eigen vector $\rightarrow v^T H v = v^T (\lambda v) = \lambda \|v\|^2$

Unit vector $= \lambda$



Because H is an $n \times n$ symmetric matrix,

H can have eigen vectors $\{v_1, v_2, \dots, v_n\}$ form a orthonormal basis.

$$f(x, y) = x^2 + 3y^2$$

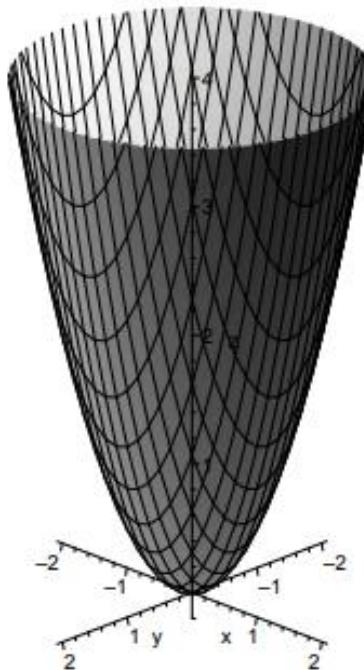
Examples

$$\frac{\partial f(x, y)}{\partial x} = 2x \quad \frac{\partial f(x, y)}{\partial y} = 6y$$

$$x = 0, y = 0$$

$$\begin{aligned}\frac{\partial^2}{\partial x \partial x} f(x, y) &= 2 \\ \frac{\partial^2}{\partial x \partial y} f(x, y) &= 0\end{aligned}$$

$$\begin{aligned}\frac{\partial^2}{\partial y \partial x} f(x, y) &= 0 \\ \frac{\partial^2}{\partial y \partial y} f(x, y) &= 6\end{aligned}$$



$$H = \begin{bmatrix} 2 & 0 \\ 0 & 6 \end{bmatrix}$$

Positive-definite
Local minima

$$f(x, y) = -x^2 + 3y^2$$

Examples

$$\frac{\partial f(x, y)}{\partial x} = -2x \quad \frac{\partial f(x, y)}{\partial y} = 6y$$

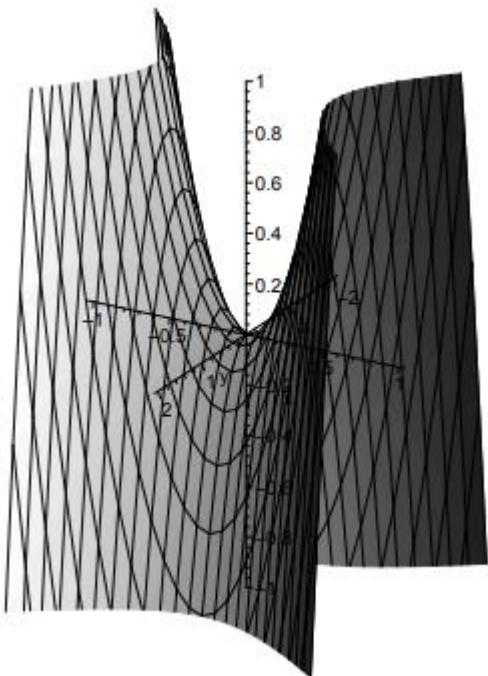
$$x = 0, y = 0$$

$$\begin{aligned}\frac{\partial^2}{\partial x \partial x} f(x, y) \\ = -2\end{aligned}$$

$$\begin{aligned}\frac{\partial^2}{\partial y \partial x} f(x, y) \\ = 0\end{aligned}$$

$$\begin{aligned}\frac{\partial^2}{\partial x \partial y} f(x, y) \\ = 0\end{aligned}$$

$$\begin{aligned}\frac{\partial^2}{\partial y \partial y} f(x, y) \\ = 6\end{aligned}$$



$$H = \begin{bmatrix} -2 & 0 \\ 0 & 6 \end{bmatrix}$$

Saddle

Degenerate

- Degenerate Hessian has at least one zero eigen value

$$f(x, y) = x^2 + y^4$$

$$\frac{\partial f(x, y)}{\partial x} = 2x \quad \frac{\partial f(x, y)}{\partial y} = 4y^3 \quad x = y = 0$$

$$\frac{\partial^2}{\partial x \partial x} f(x, y) = 2 \quad \frac{\partial^2}{\partial x \partial y} f(x, y) = 0 \quad H = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\frac{\partial^2}{\partial y \partial x} f(x, y) = 0 \quad \frac{\partial^2}{\partial y \partial y} f(x, y) = 12y^2$$

Degenerate

- Degenerate Hessian has at least one zero eigen value

$$f(x, y) = x^2 + y^4$$

$$g(x, y) = x^2 - y^4$$

$$x = y = 0$$

$$g = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$x = y = 0$$

$$g = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$H = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$$

$$H = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$$

No Difference

Degenerate

$$h(x, y) = 0$$

$$x = y = 0$$

$$g = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

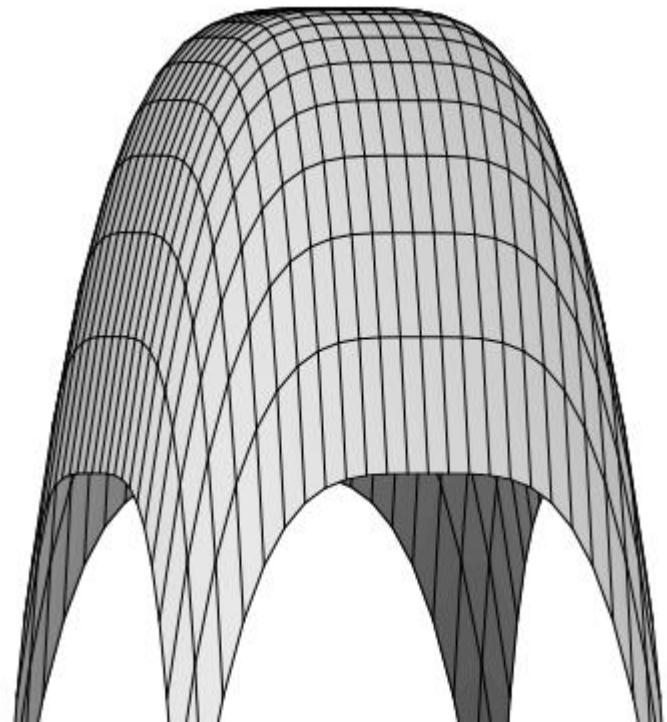
$$f(x, y) = -x^4 - y^4$$

$$H = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\frac{\partial f(x, y)}{\partial x} = -4x^3 \quad \frac{\partial f(x, y)}{\partial y} = -4y^3$$

$$\frac{\partial^2}{\partial x \partial x} f(x, y) = -12x^2 \quad \frac{\partial^2}{\partial x \partial y} f(x, y) = 0$$

$$\frac{\partial^2}{\partial y \partial x} f(x, y) = 0 \quad \frac{\partial^2}{\partial y \partial y} f(x, y) = -12y^2$$



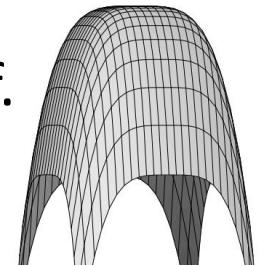
$$H = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

<http://homepages.math.uic.edu/~julius/monkeysaddle.html>



Monkey Saddle

c.f.



$$\frac{\partial f(x, y)}{\partial x} = 3x^2 - 3y^2$$

$$\frac{\partial f(x, y)}{\partial y} = -6xy$$

$$\frac{\partial^2}{\partial x \partial x} f(x, y) = 6x$$

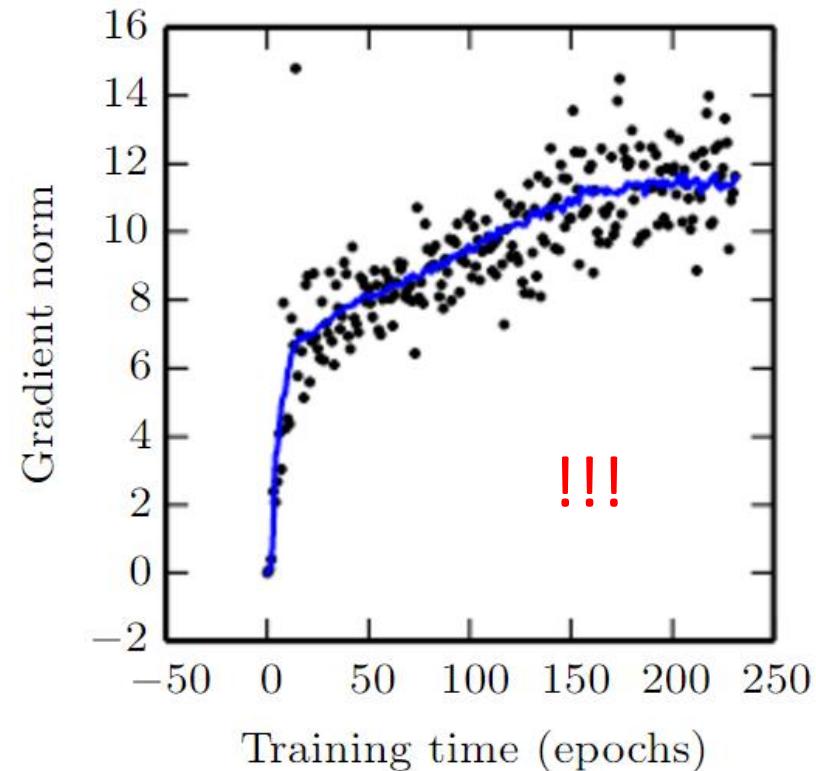
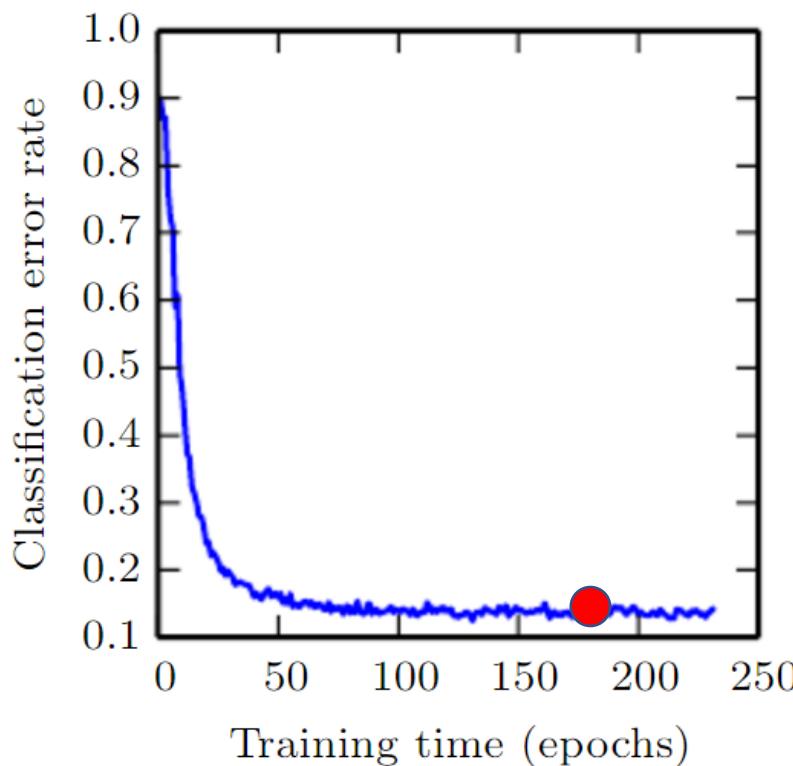
$$\frac{\partial^2}{\partial x \partial y} f(x, y) = -6y$$

$$\frac{\partial^2}{\partial y \partial x} f(x, y) = -6y$$

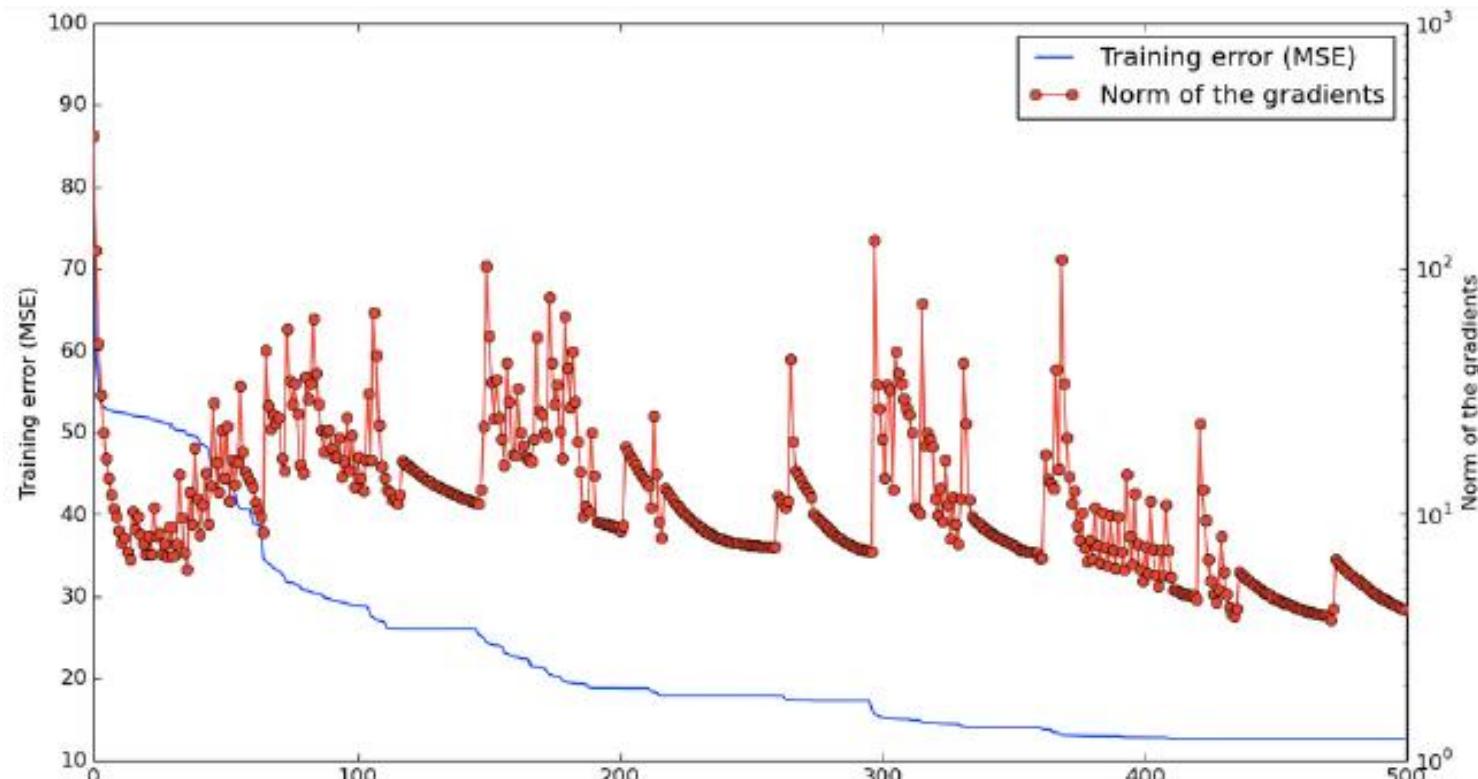
$$\frac{\partial^2}{\partial y \partial y} f(x, y) = -6x$$

Training stuck \neq Zero Gradient

- People believe training stuck because the parameters are around a critical point

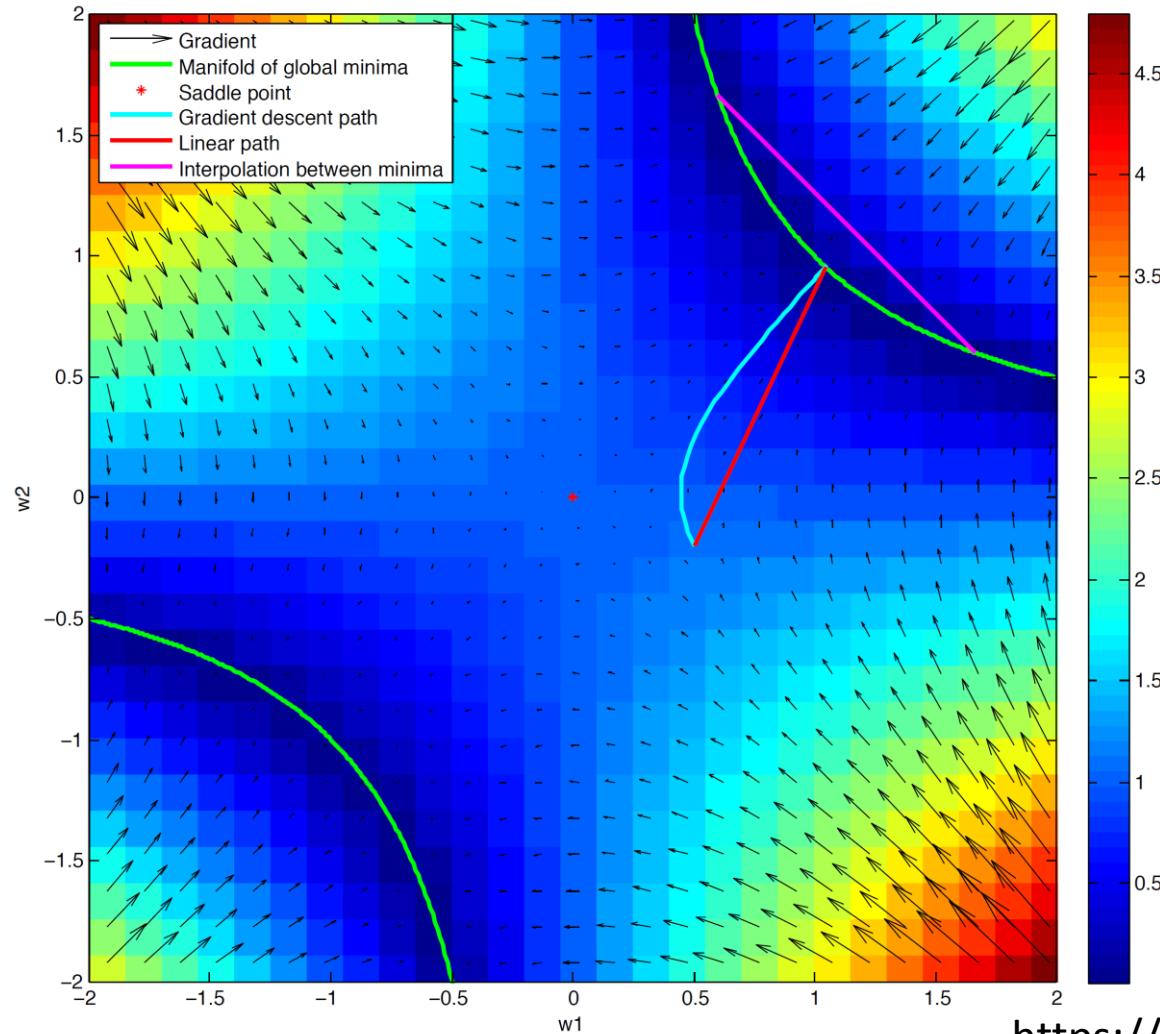
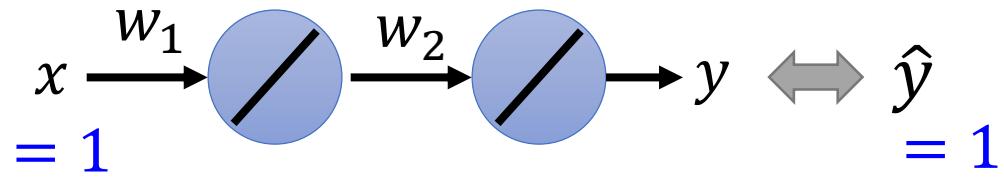


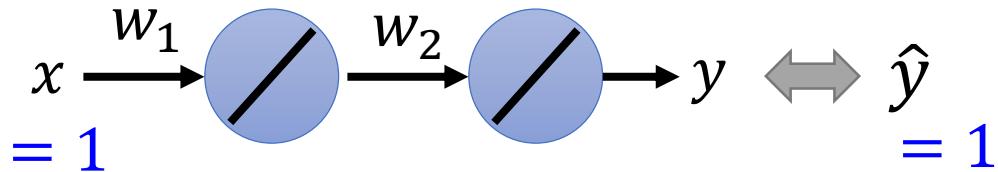
Training stuck \neq Zero Gradient



Approach a saddle point, and then escape

Deep Linear Network





$$L = (\hat{y} - w_1 w_2 x)^2 = (1 - w_1 w_2)^2$$

$$\frac{\partial L}{\partial w_1} = 2(1 - w_1 w_2)(-w_2)$$

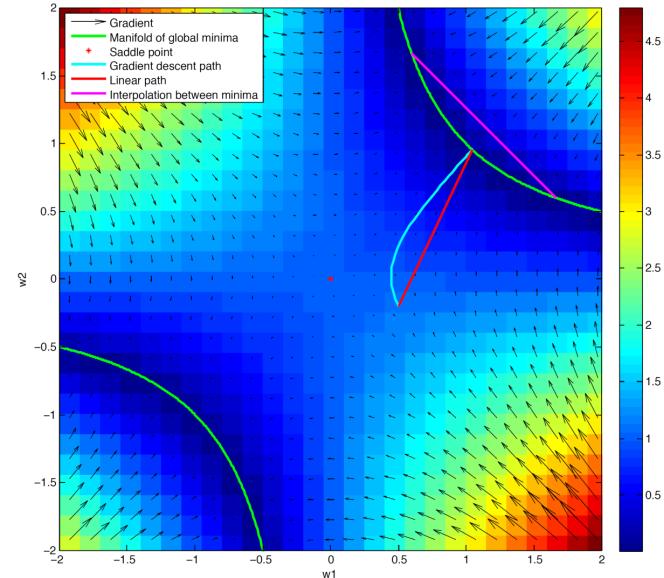
$$\frac{\partial L}{\partial w_2} = 2(1 - w_1 w_2)(-w_1)$$

$$\frac{\partial^2 L}{\partial w_1^2} = 2(-w_2)(-w_2)$$

$$\frac{\partial^2 L}{\partial w_2 \partial w_1} = -2 + 4w_1 w_2$$

$$\frac{\partial^2 L}{\partial w_1 \partial w_2} = -2 + 4w_1 w_2$$

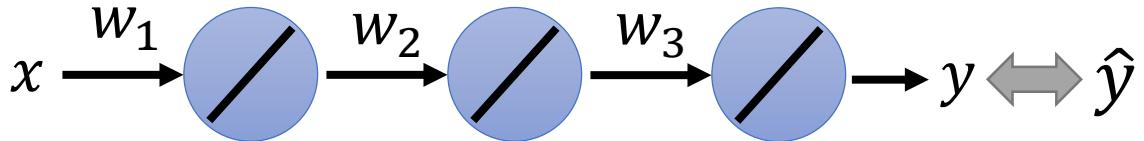
$$\frac{\partial^2 L}{\partial w_2^2} = 2(-w_1)(-w_1)$$



The probability of stuck as saddle point is almost zero.
Easy to escape

2-hidden layers

$$L = (1 - w_1 w_2 w_3)^2$$



$$\frac{\partial L}{\partial w_1} = 2(1 - w_1 w_2 w_3)(-w_2 w_3)$$

$$w_1 w_2 w_3 = 1 \quad \text{global minima}$$

$$\frac{\partial L}{\partial w_2} = 2(1 - w_1 w_2 w_3)(-w_1 w_3)$$

$$w_1 = w_2 = w_3 = 0$$

$$\frac{\partial L}{\partial w_3} = 2(1 - w_1 w_2 w_3)(-w_1 w_2)$$

$$H = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

So flat

$$\frac{\partial^2 L}{\partial w_1^2} = 2(-w_2 w_3)^2$$

$$w_1 = w_2 = 0, w_3 = k$$

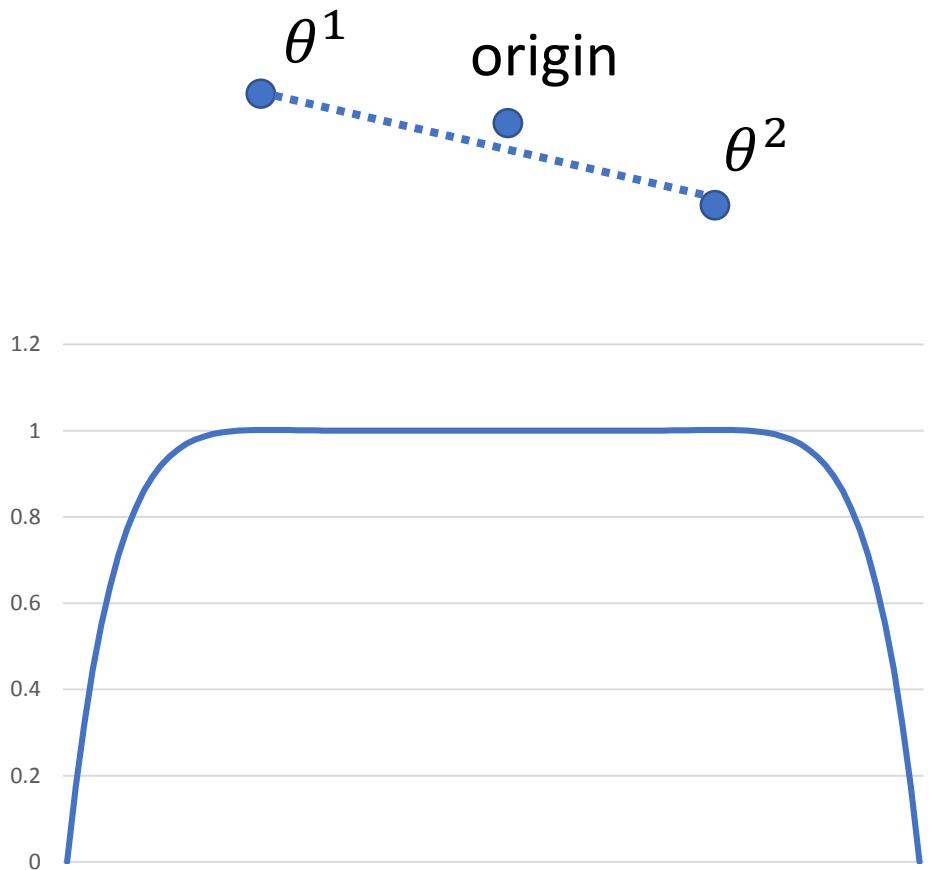
$$\frac{\partial^2 L}{\partial w_2 \partial w_1} = -2w_3 + 4w_1 w_2 (w_3)^2$$

$$H = \begin{bmatrix} 0 & -2 & 0 \\ -2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

All minima are global, some critical points are “bad”.

Saddle point

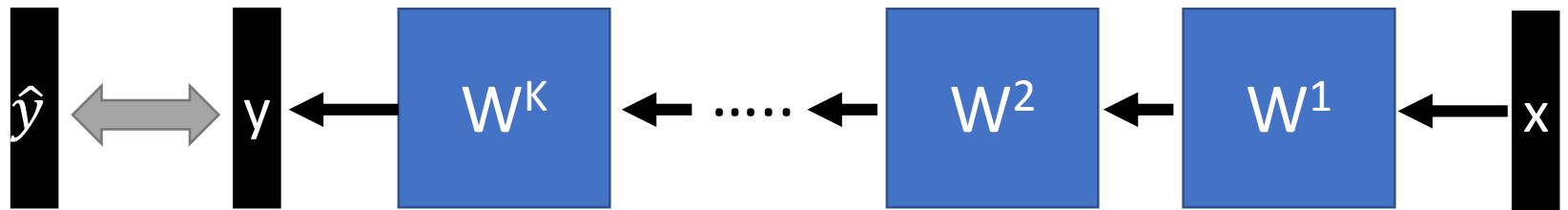
10 hidden layers



10-hidden layers

Demo

Deep Linear Network



$$y = W^K W^{K-1} \dots W^2 W^1 x \quad L = \sum_{n=1}^N (x^n - \hat{y}^n)^2$$

Hidden layer size \geq Input dim, output dim

More than two hidden layers can produce saddle point without negative eigenvalues.

Reference

- Kenji Kawaguchi, Deep Learning without Poor Local Minima, NIPS, 2016
- Haihao Lu, Kenji Kawaguchi, Depth Creates No Bad Local Minima, arXiv, 2017
- Thomas Laurent, James von Brecht, Deep linear neural networks with arbitrary loss: All local minima are global, arXiv, 2017
- Maher Nouiehed, Meisam Razaviyayn, Learning Deep Models: Critical Points and Local Openness, arXiv, 2018

Non-linear Deep Network

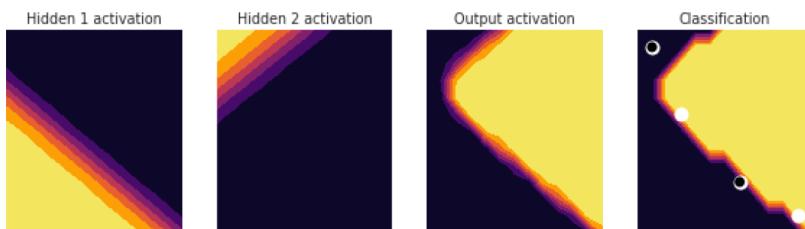
Does it have local minima?

證明事情不存在很難，證明事情存在相對容易

感謝曾子家同學發現投影片上的錯字

Even Simple Task can be Difficult

h	XOR					Jellyfish				
	ReLU	Sigmoid	ReLU	Sigmoid	ReLU	Sigmoid	ReLU	Sigmoid	ReLU	Sigmoid
2	Adam	28%	79%	7%	0%	GD	23%	90%	16%	62%
3	Adam	52%	98%	34%	0%	GD	47%	100%	33%	100%
4	Adam	68%	100%	50%	2%	GD	70%	100%	66%	100%
5	Adam	81%	100%	51%	27%	GD	80%	100%	68%	100%
6	Adam	91%	100%	61%	17%	GD	89%	100%	69%	100%
7	Adam	97%	100%	69%	58%	GD	89%	100%	86%	100%

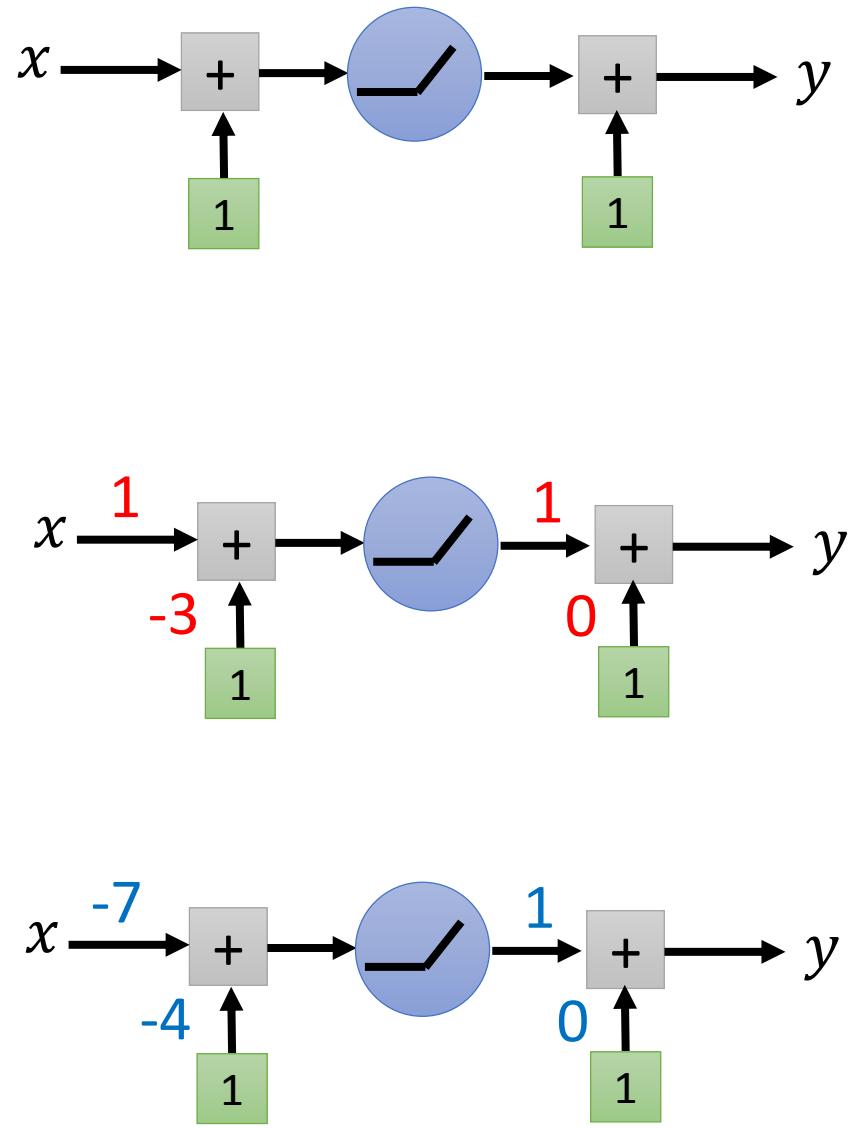
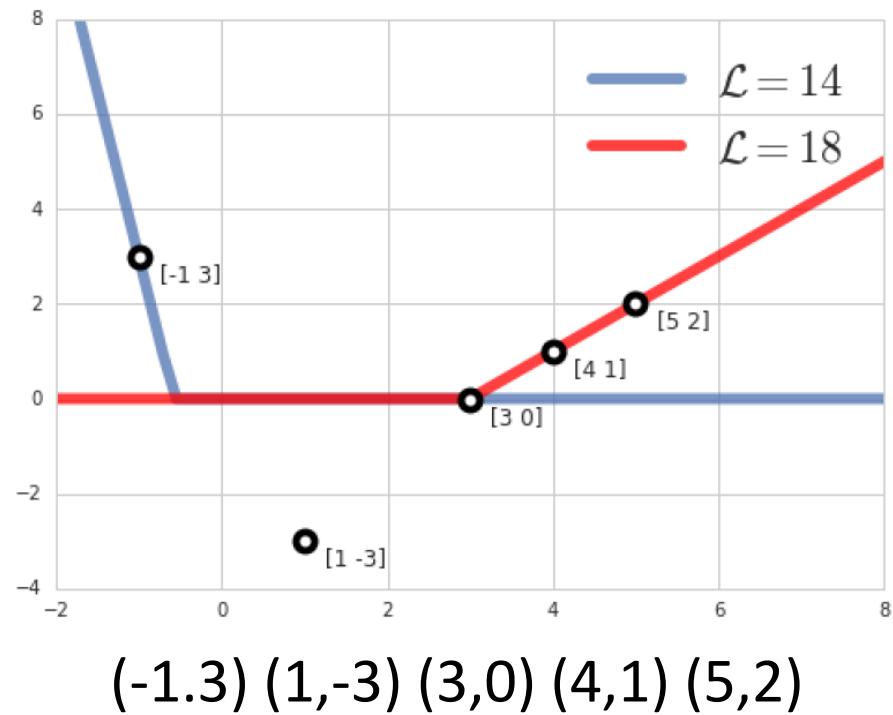


(a) Optimally converged net for Jellyfish.



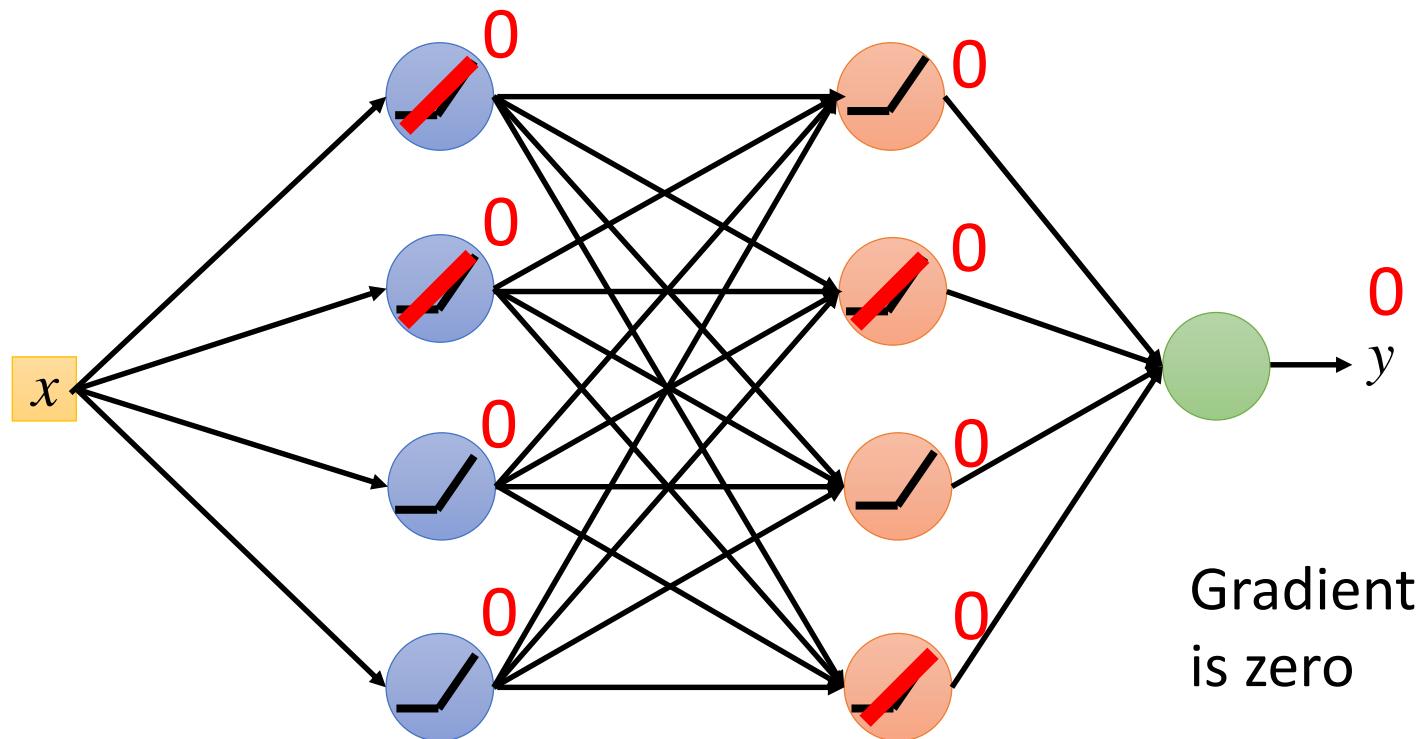
(b) Stuck net for Jellyfish.

ReLU has local



This relu network has local minima.

“Blind Spot” of ReLU

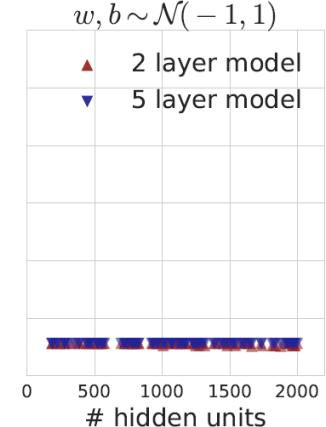
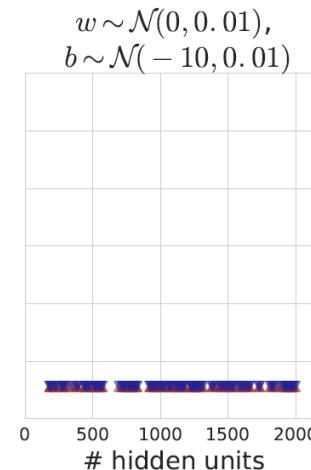
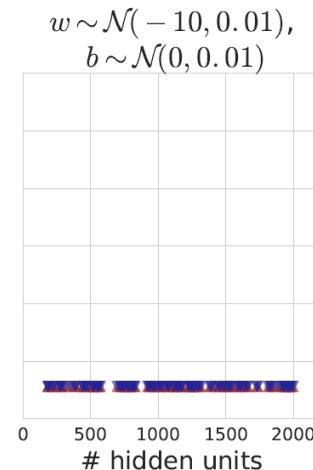
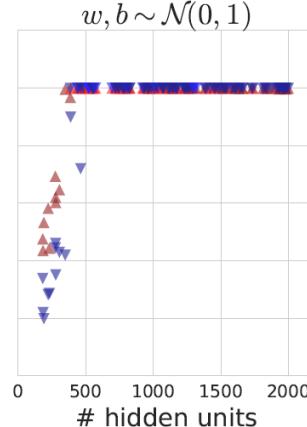
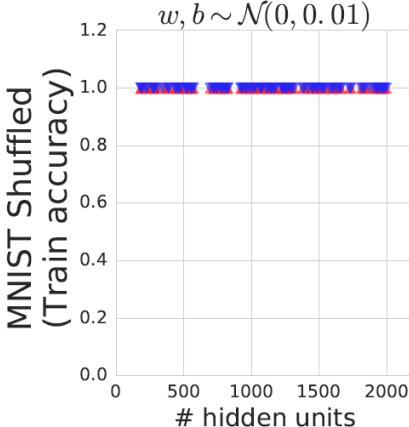
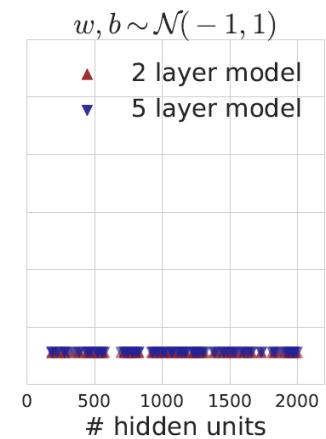
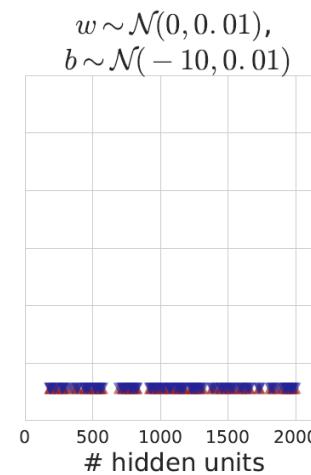
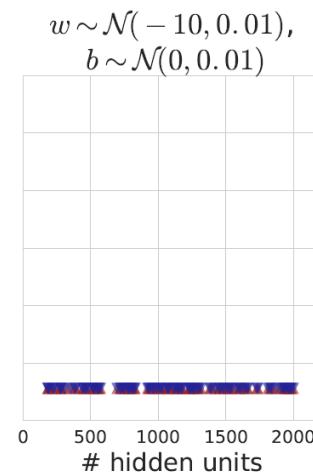
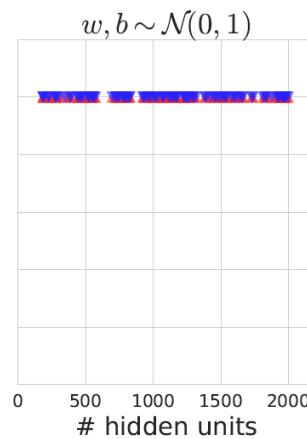
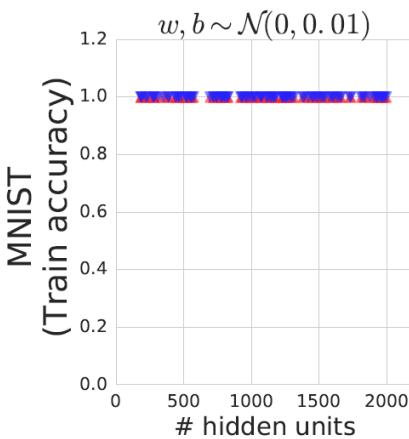


It is pretty easy to make this happens

“Blind Spot” of ReLU

Consider your
initialization

- MNIST, Adam, 1M updates



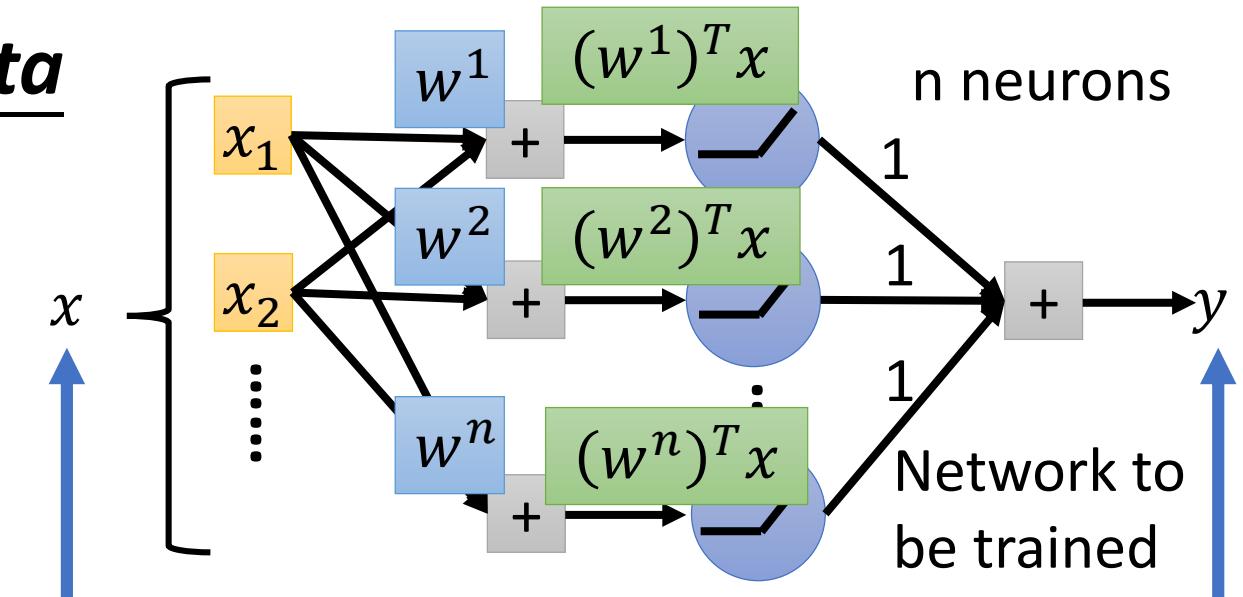
Considering Data

If $n \geq k$ and
 $w^i = v^i$

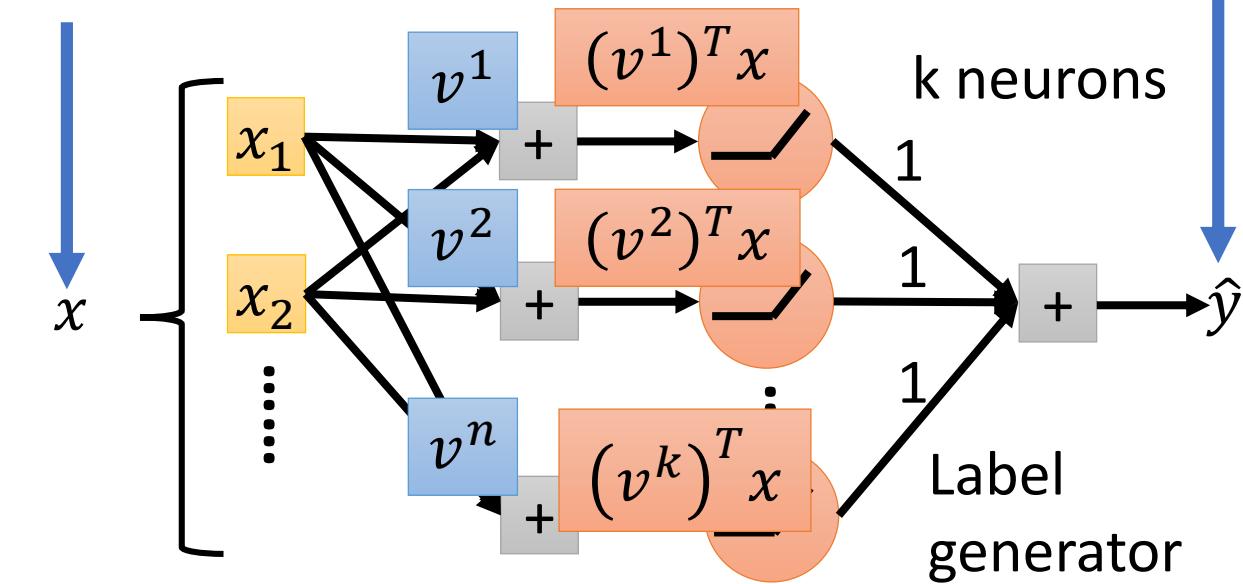


We obtain
global minima

The number of
k and n matters



$N(0,1)$



Considering Data

Table 1: Spurious local minima found for $n = k$

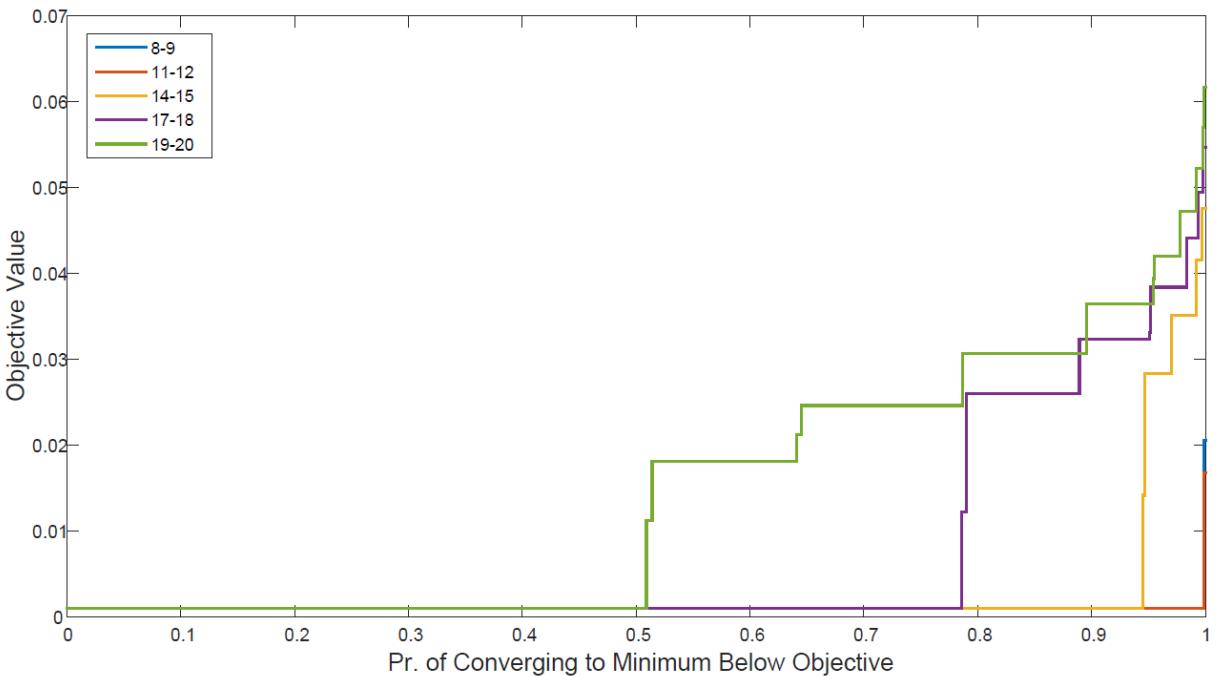
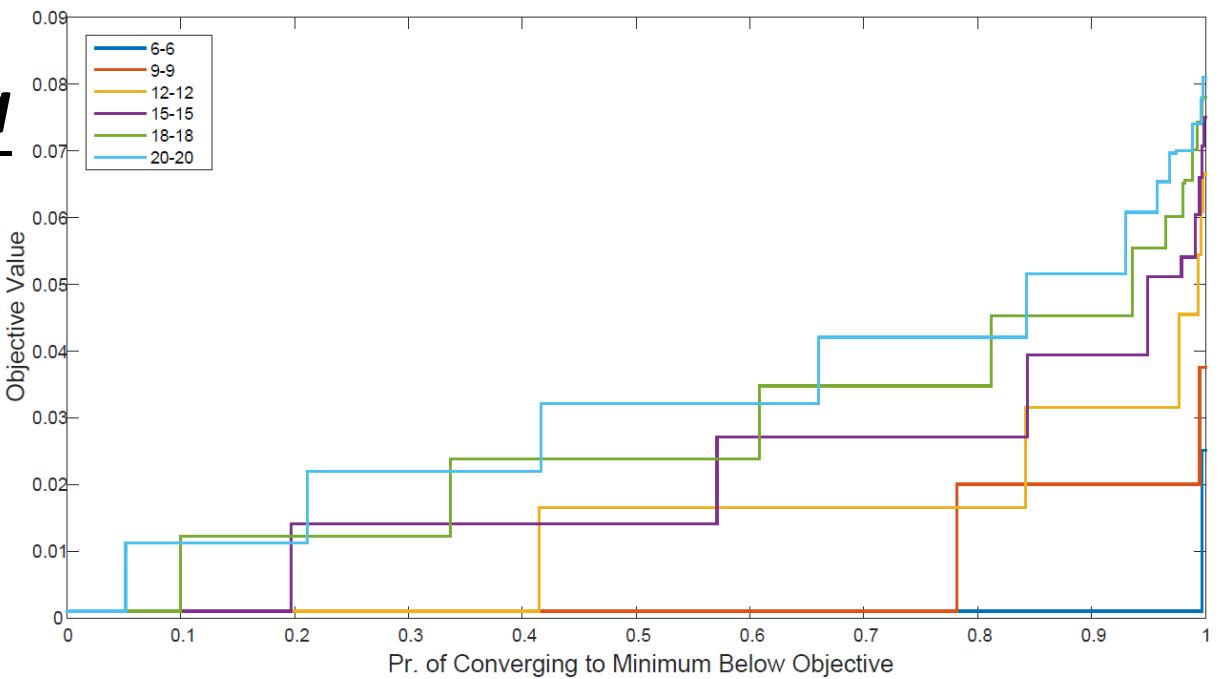
k	n	% of runs converging to local minima	Average minimal eigenvalue	Average objective value
6	6	0.3%	0.0047	0.025
7	7	5.5%	0.014	0.023
8	8	12.6%	0.021	0.021
9	9	21.8%	0.027	0.02
10	10	34.6%	0.03	0.022
11	11	45.5%	0.034	0.022
12	12	58.5%	0.035	0.021
13	13	73%	0.037	0.022
14	14	73.6%	0.038	0.023
15	15	80.3%	0.038	0.024
16	16	85.1%	0.038	0.027
17	17	89.7%	0.039	0.027
18	18	90%	0.039	0.029
19	19	93.4%	0.038	0.031
20	20	94%	0.038	0.033

Table 2: Spurious local minima found for $n \neq k$

k	n	% of runs converging to local minima	Average minimal eigenvalue	Average objective value
8	9	0.1%	0.0059	0.021
10	11	0.1%	0.0057	0.018
11	12	0.1%	0.0056	0.017
12	13	0.3%	0.0054	0.016
13	14	1.5%	0.0015	0.038
14	15	5.5%	0.002	0.033
15	16	10.1%	0.004	0.032
16	17	18%	0.0055	0.031
17	18	20.9%	0.007	0.031
18	19	36.9%	0.0064	0.028
19	20	49.1%	0.0077	0.027

No local for $n \geq k + 2$

Considering Data



Reference

- Grzegorz Swirszcz, Wojciech Marian Czarnecki, Razvan Pascanu, “Local minima in training of neural networks”, arXiv, 2016
- Itay Safran, Ohad Shamir, “Spurious Local Minima are Common in Two-Layer ReLU Neural Networks”, arXiv, 2017
- Yi Zhou, Yingbin Liang, “Critical Points of Neural Networks: Analytical Forms and Landscape Properties”, arXiv, 2017
- Shai Shalev-Shwartz, Ohad Shamir, Shaked Shammah, “Failures of Gradient-Based Deep Learning”, arXiv, 2017

The theory should looks like ...

Under some conditions (initialization, data,),

We can find global optimal.

Conjecture about Deep Learning

Almost all local minimum have very similar loss to the global optimum, and hence finding a local minimum is good enough.

Analyzing Hessian

- When we meet a critical point, it can be saddle point or local minima.
- Analyzing H

If the network has N parameters

$$\begin{array}{ccccc} v_1 & v_2 & v_3 & \dots & v_N \\ \lambda_1 & \lambda_2 & \lambda_3 & \dots & \lambda_N \end{array}$$

We assume λ has 1/2 (?) to be positive, 1/2 (?) to be negative.

Analyzing Hessian

- If $N=1$: v_1 λ_1 1/2 local minima, 1/2 local maxima,
 Saddle point is almost impossible
- If $N=2$: $v_1 \quad v_2$ $\lambda_1 \quad \lambda_2$ $\begin{matrix} + & + \\ + & - \\ - & + \end{matrix}$ $\begin{matrix} - \\ - \end{matrix}$
 1/4 local minima, 1/4 local maxima,
 1/2 Saddle points
- If $N=10$: 1/1024 local minima, 1/1024 local maxima,
 Almost every critical point is saddle point

When a network is very large,

It is almost impossible to meet a local minima.

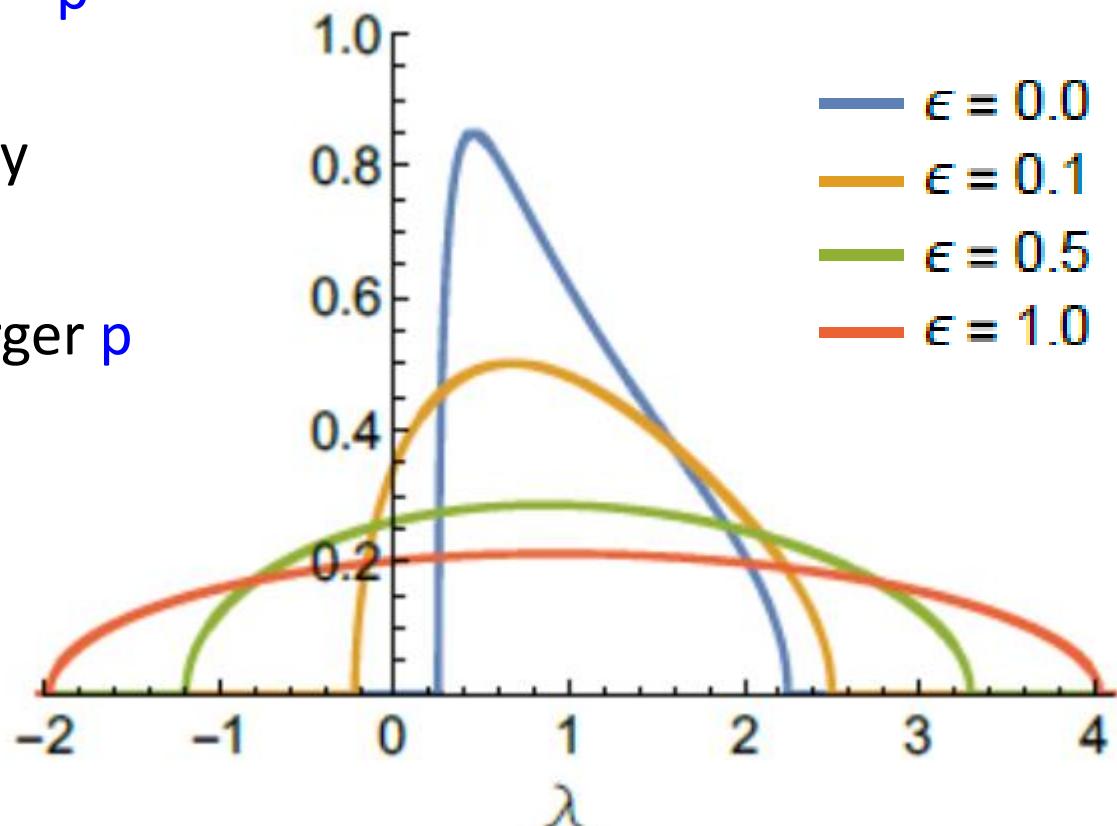
Saddle point is what you need to worry about.

Error v.s. Eigenvalues

We assume λ has $1/2 (?)$
to be negative. p

p is a probability
related to error

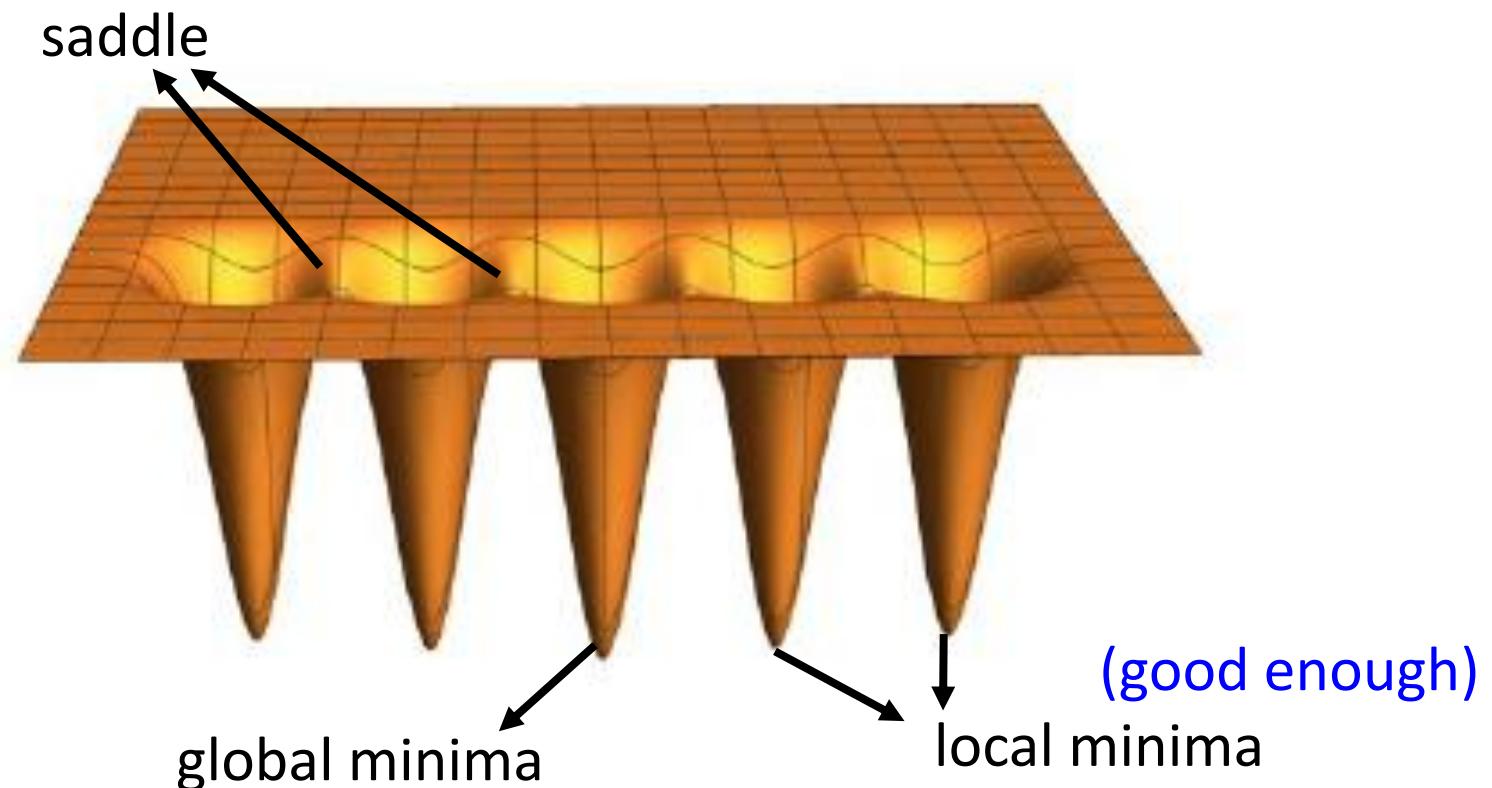
Larger error, larger p



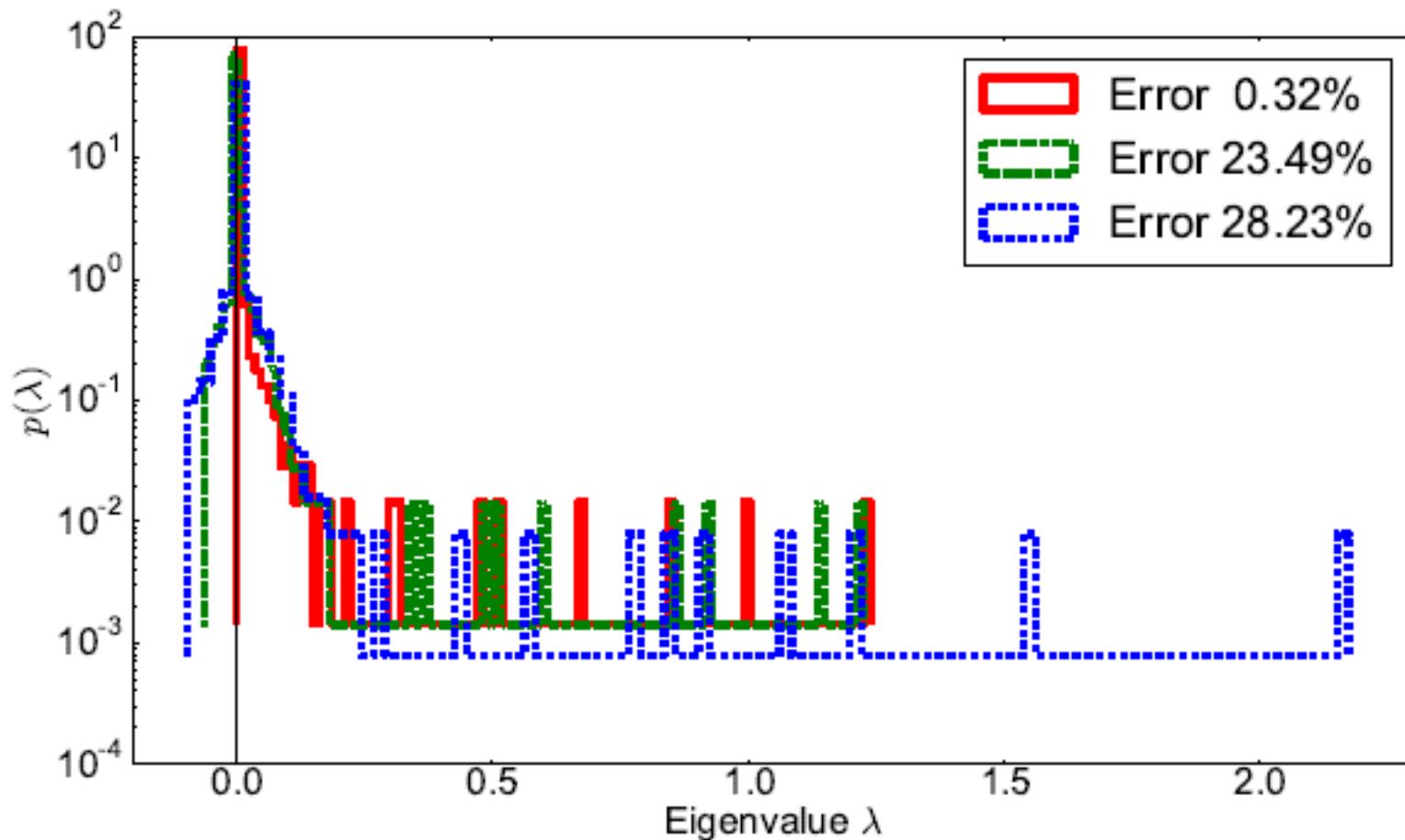
Source of image:

<http://proceedings.mlr.press/v70/pennington17a/pennington17a.pdf>

Guess about Error Surface

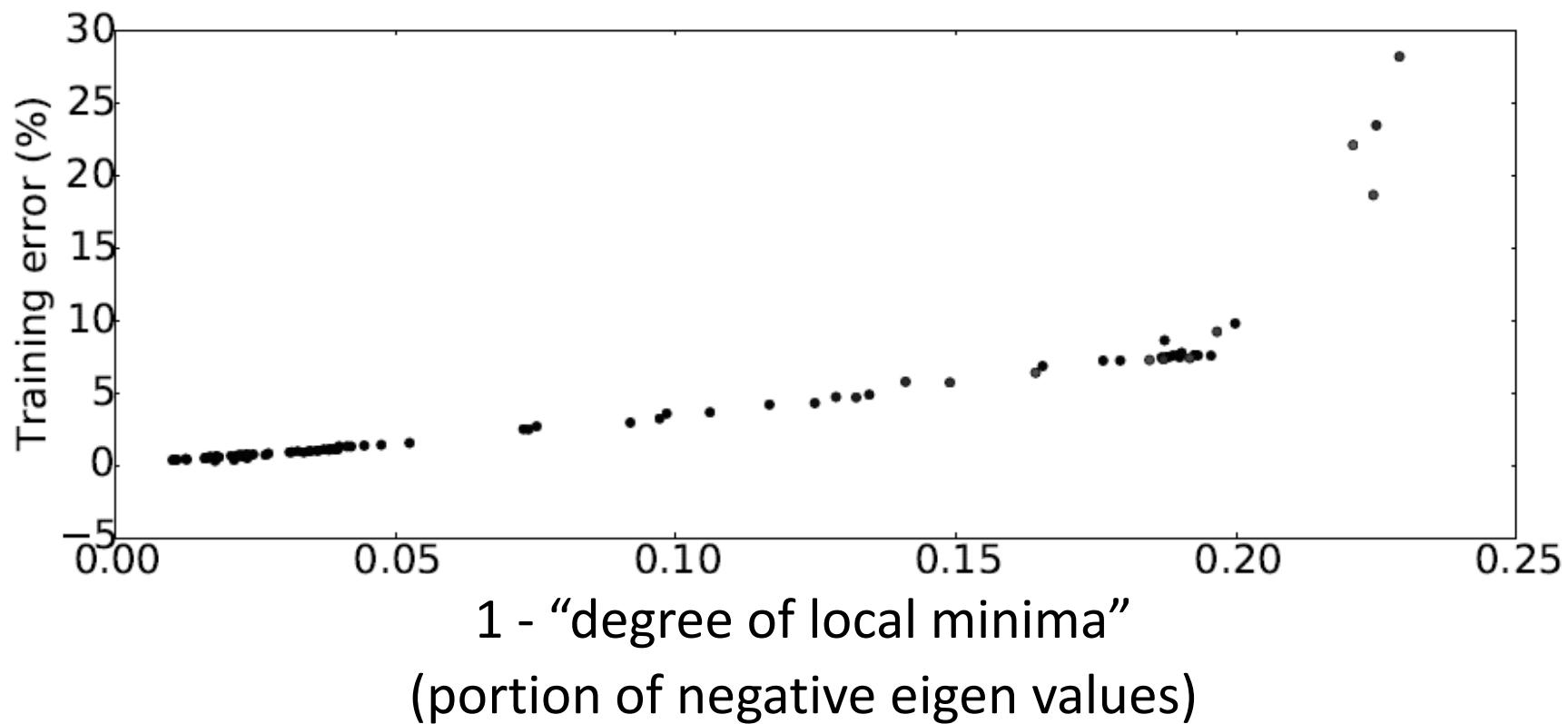


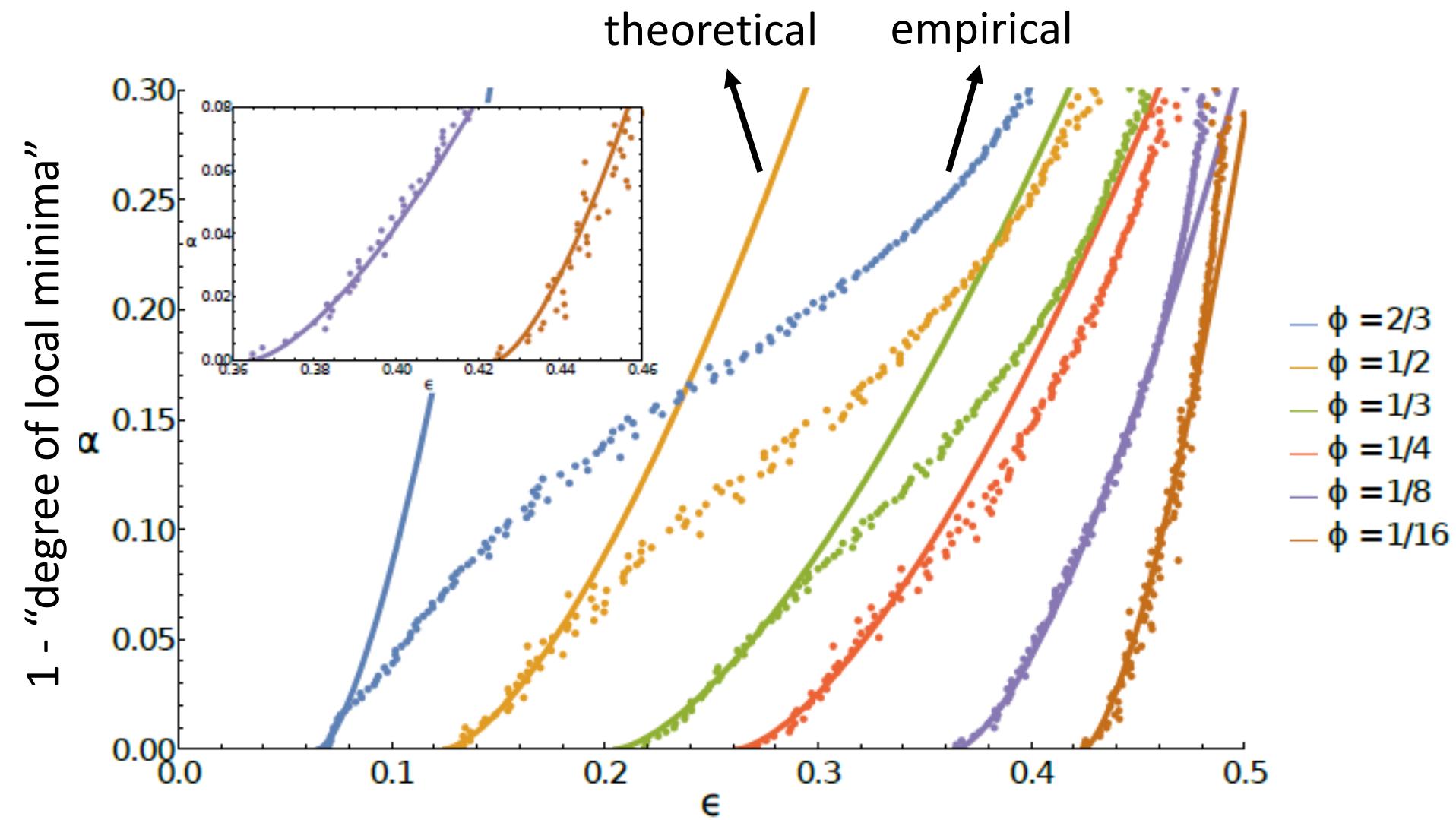
Training Error v.s. Eigenvalues



Training Error v.s. Eigenvalues

Portion of positive eigenvalues → “Degree of Local Minima”



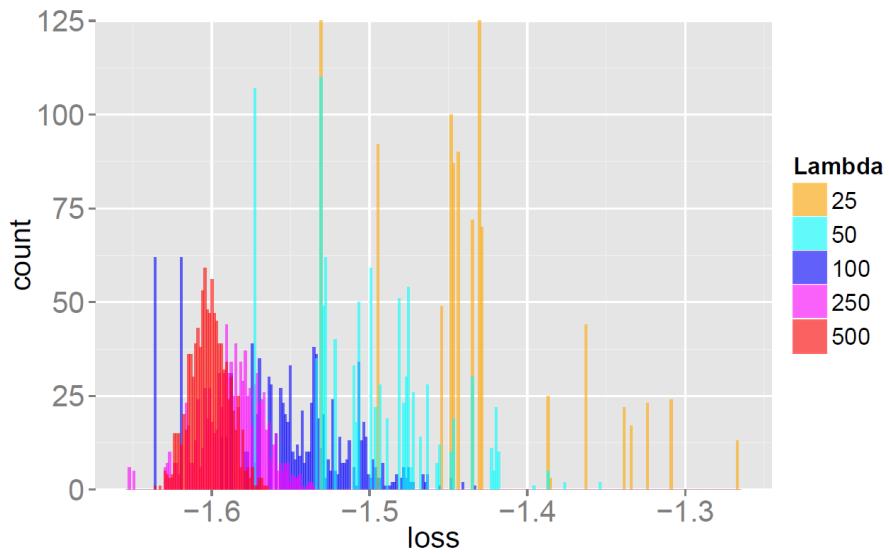


$1 - \text{"degree of local minima"}$
(portion of negative eigen values)

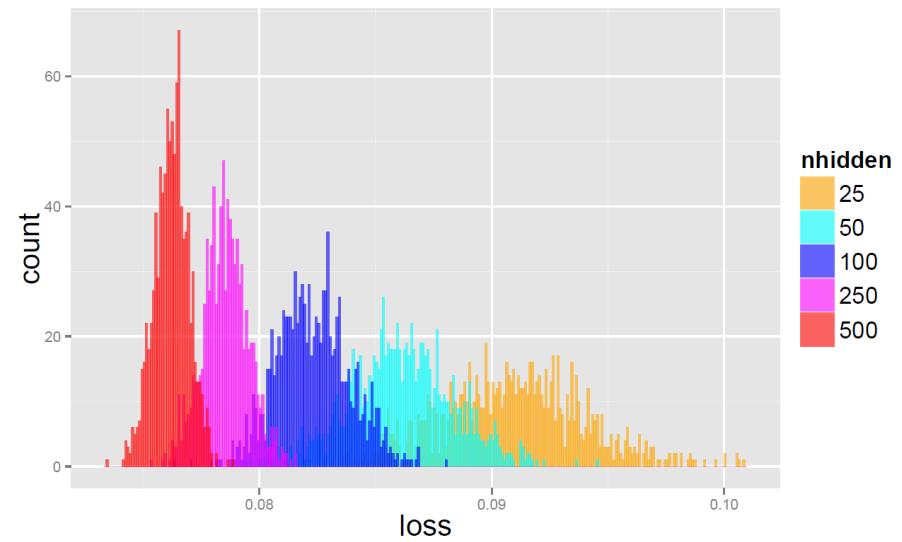
$$\alpha \propto \left(\frac{\epsilon}{c} - 1\right)^{3/2}$$

Spin Glass v.s. Deep Learning

- Deep learning is the same as spin glass model with **7 assumptions.**



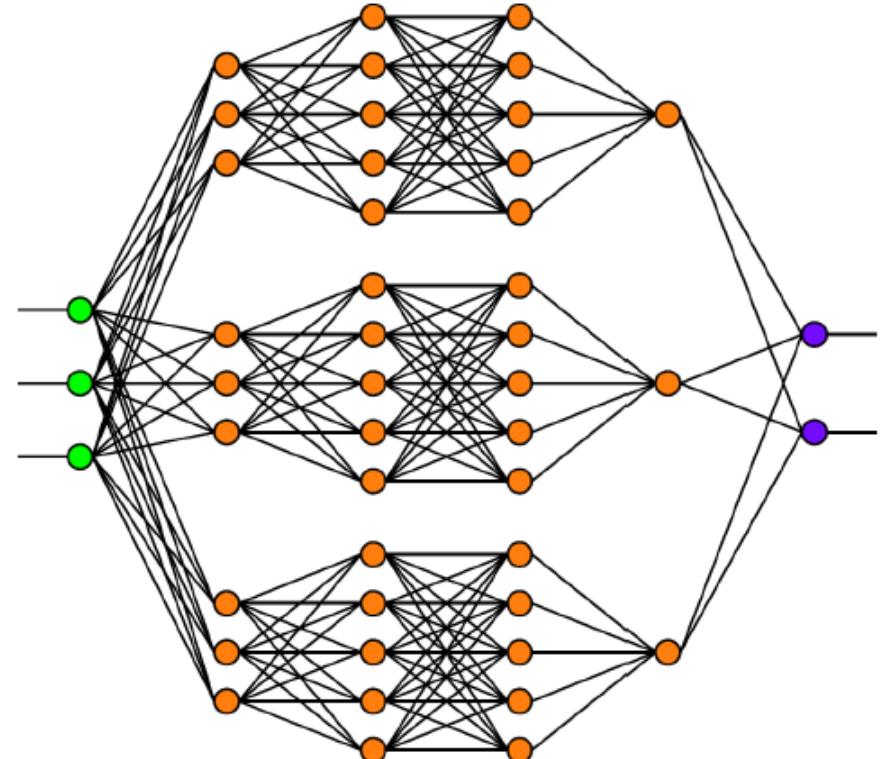
spin glass model



network

More Theory

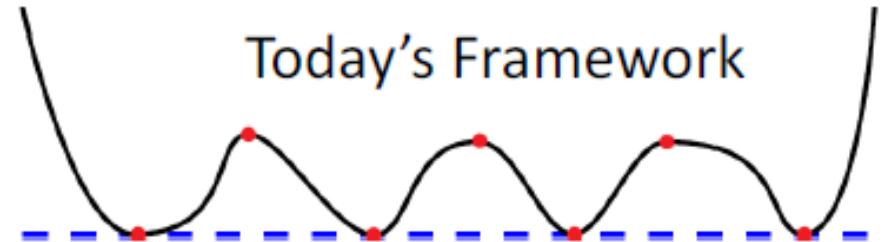
- If the size of network is **large enough**, we can find global optimal by gradient descent
 - Independent to initialization



Non-Convex Function



Today's Framework

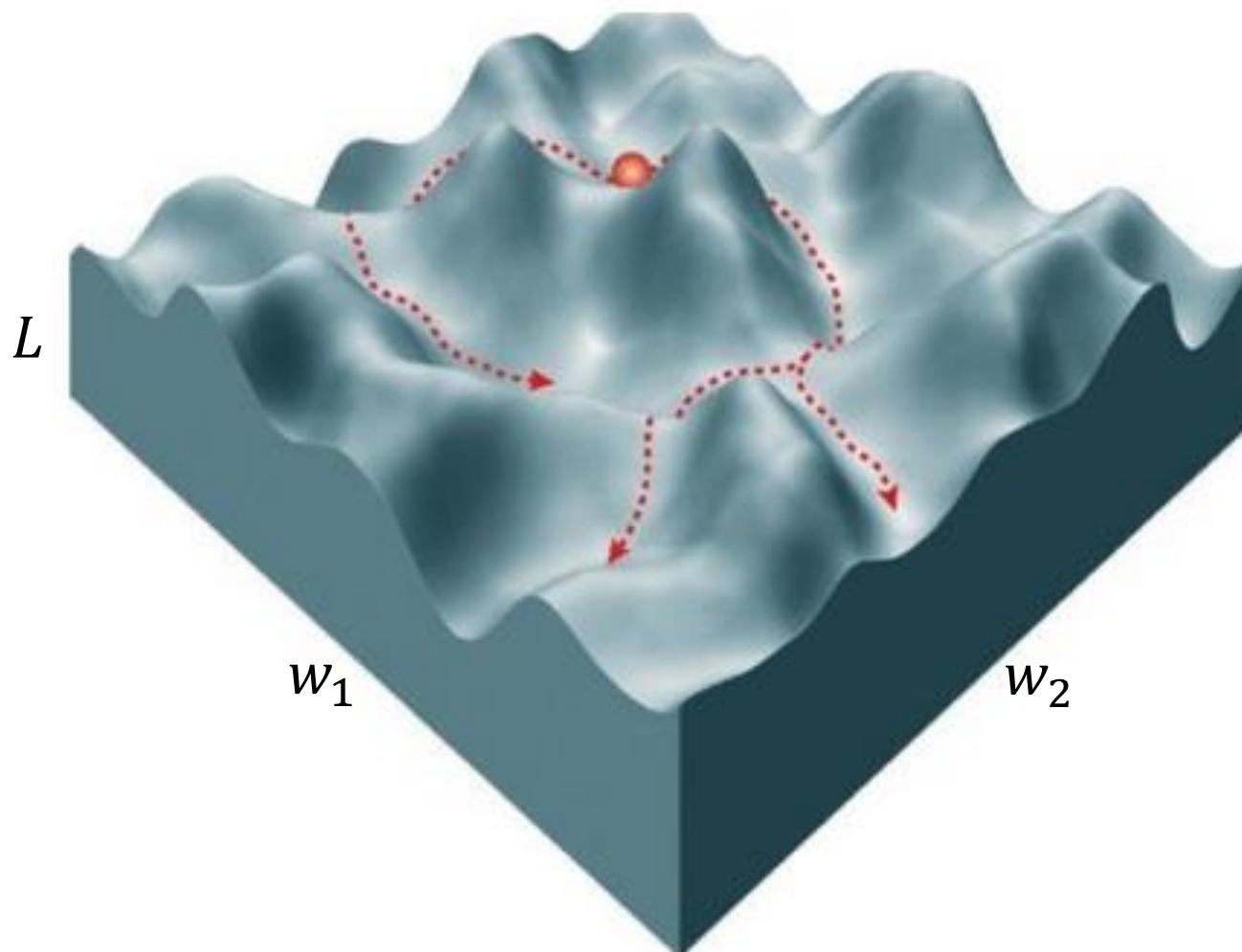


Reference

- Razvan Pascanu, Yann N. Dauphin, Surya Ganguli, Yoshua Bengio, On the saddle point problem for non-convex optimization, arXiv, 2014
- Yann Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, Yoshua Bengio, “Identifying and attacking the saddle point problem in high-dimensional non-convex optimization”, NIPS, 2014
- Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, Yann LeCun, “The Loss Surfaces of Multilayer Networks”, PMLR, 2015
- Jeffrey Pennington, Yasaman Bahri, “Geometry of Neural Network Loss Surfaces via Random Matrix Theory”, PMLR, 2017
- Benjamin D. Haeffele, Rene Vidal, ” Global Optimality in Neural Network Training”, CVPR, 2017

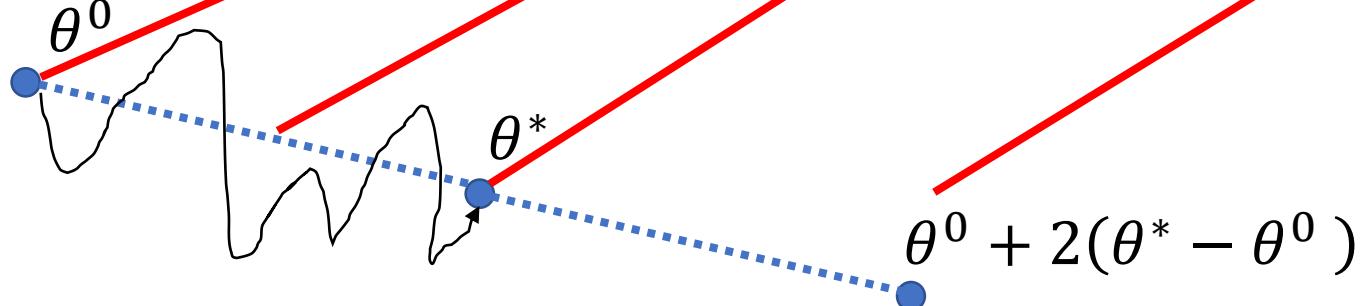
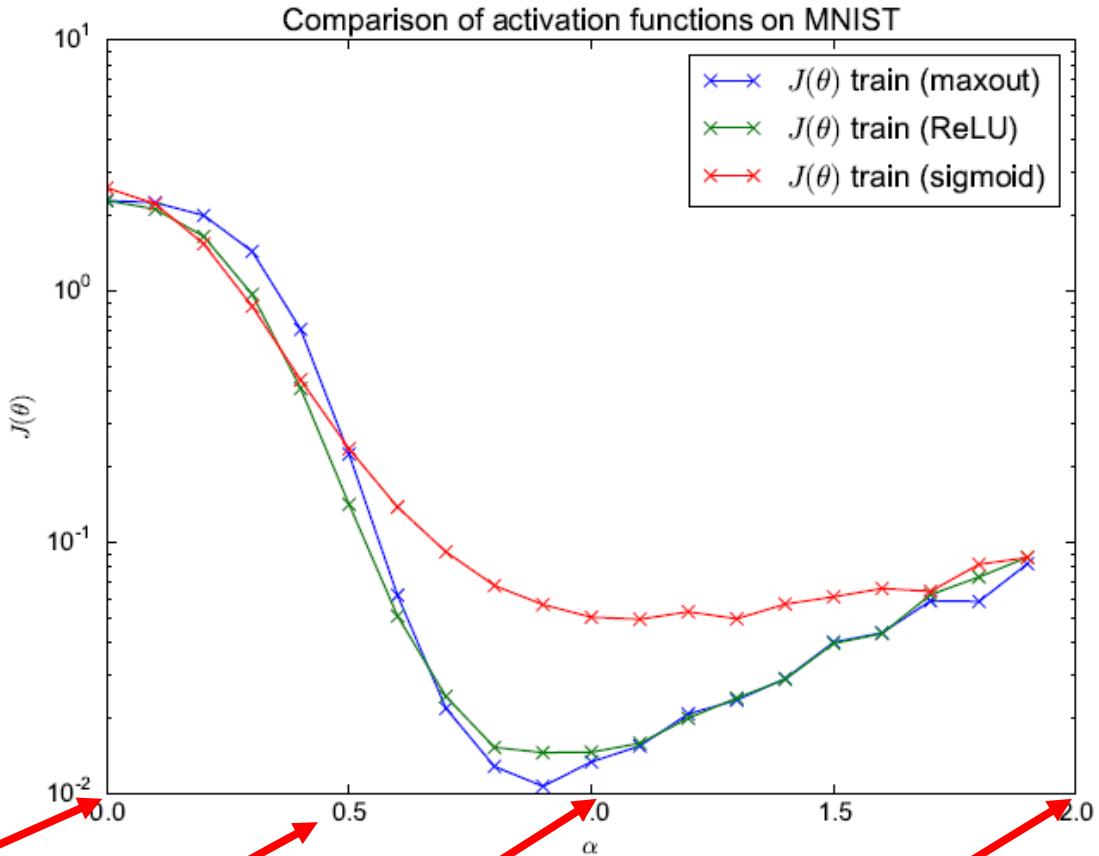
What does the Error
Surface look like?

Error Surface

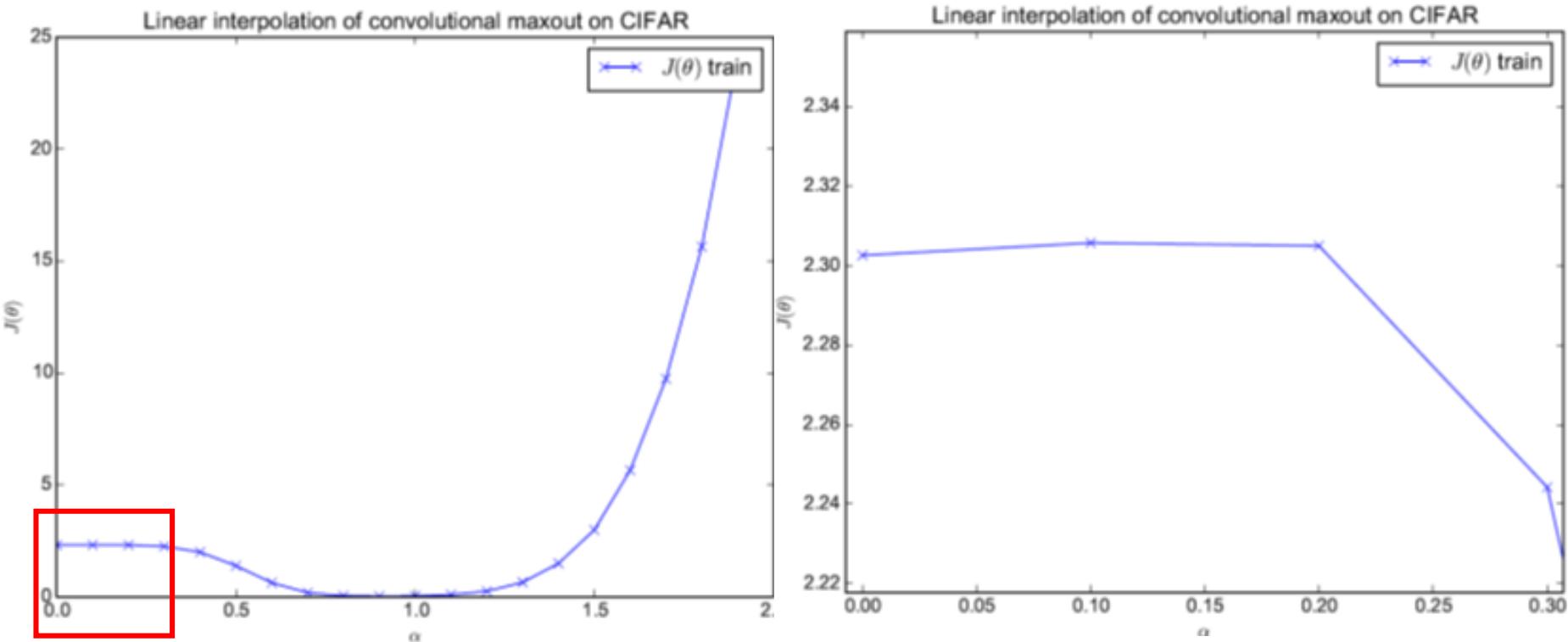


Profile

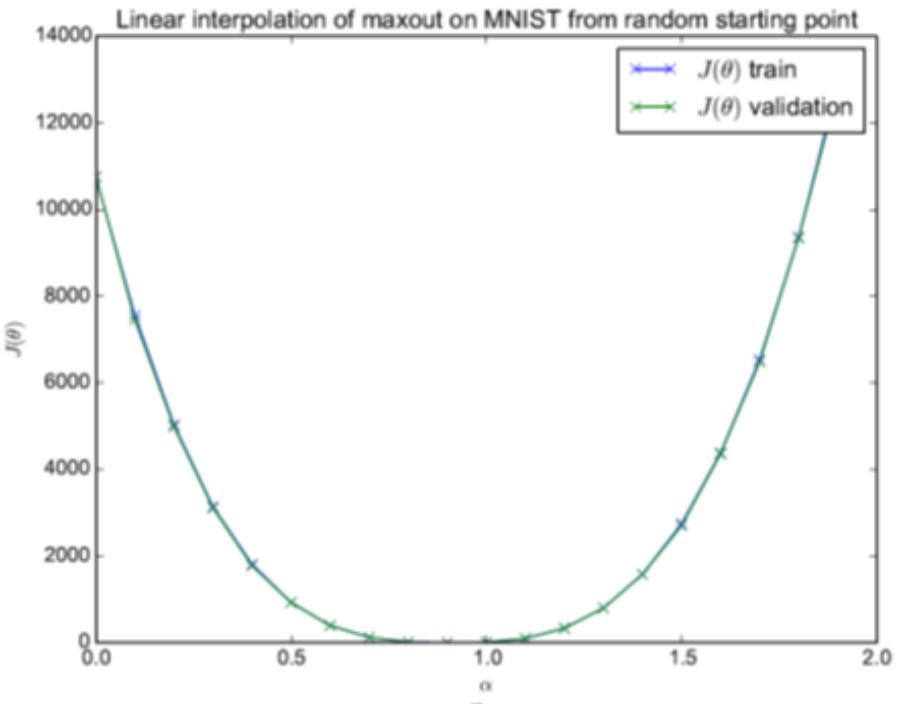
local minima is rare?



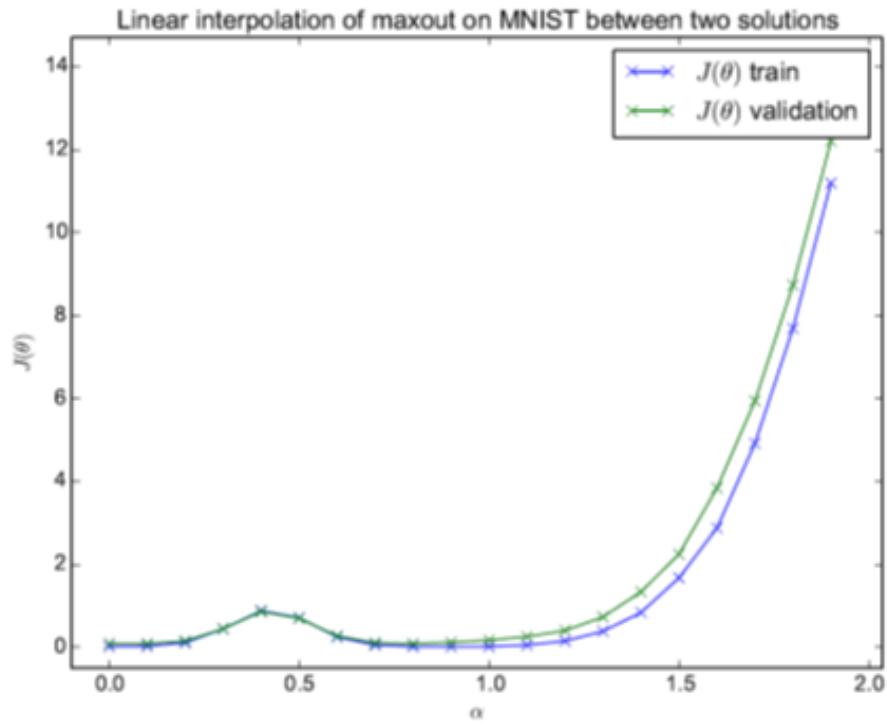
Profile



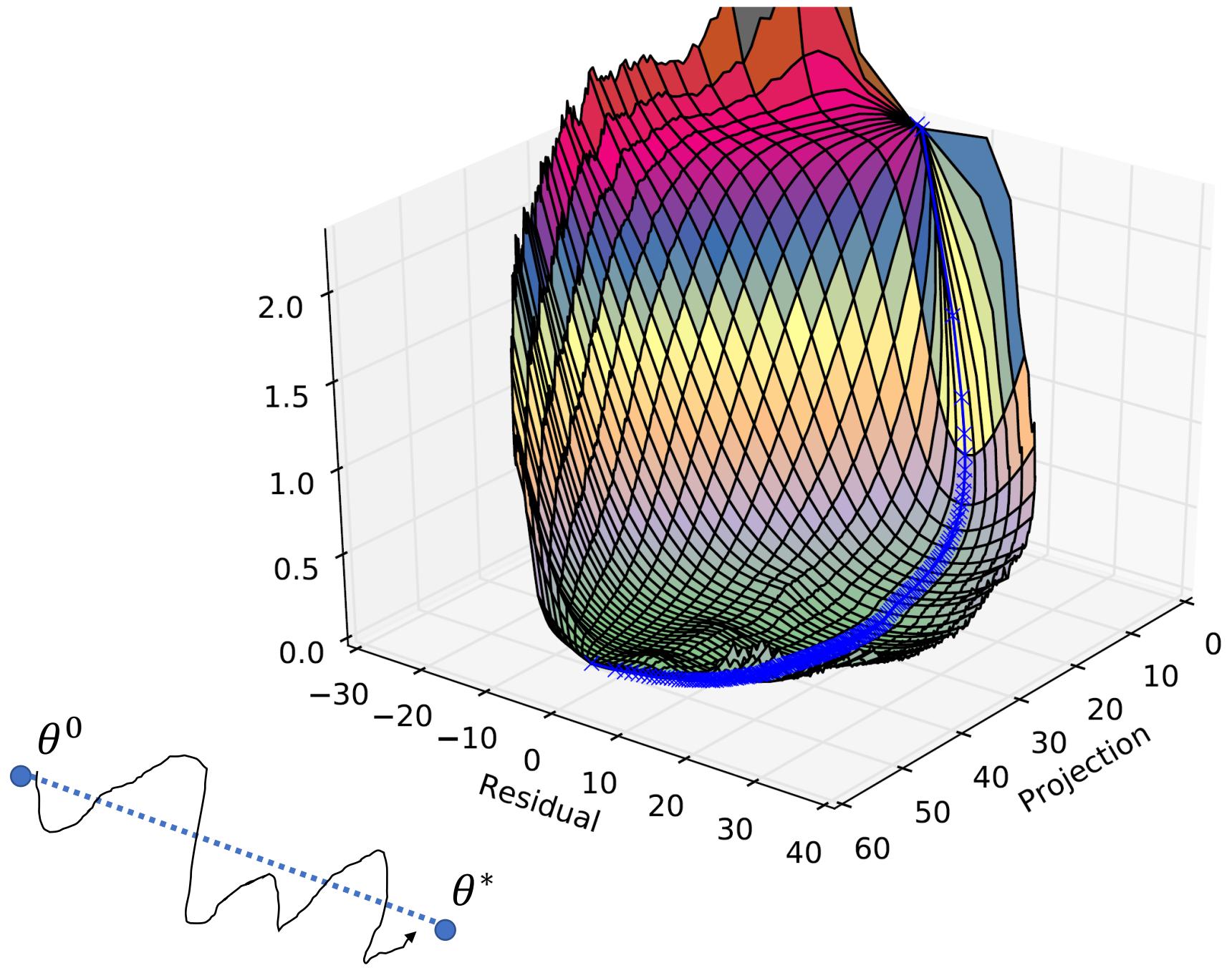
Profile

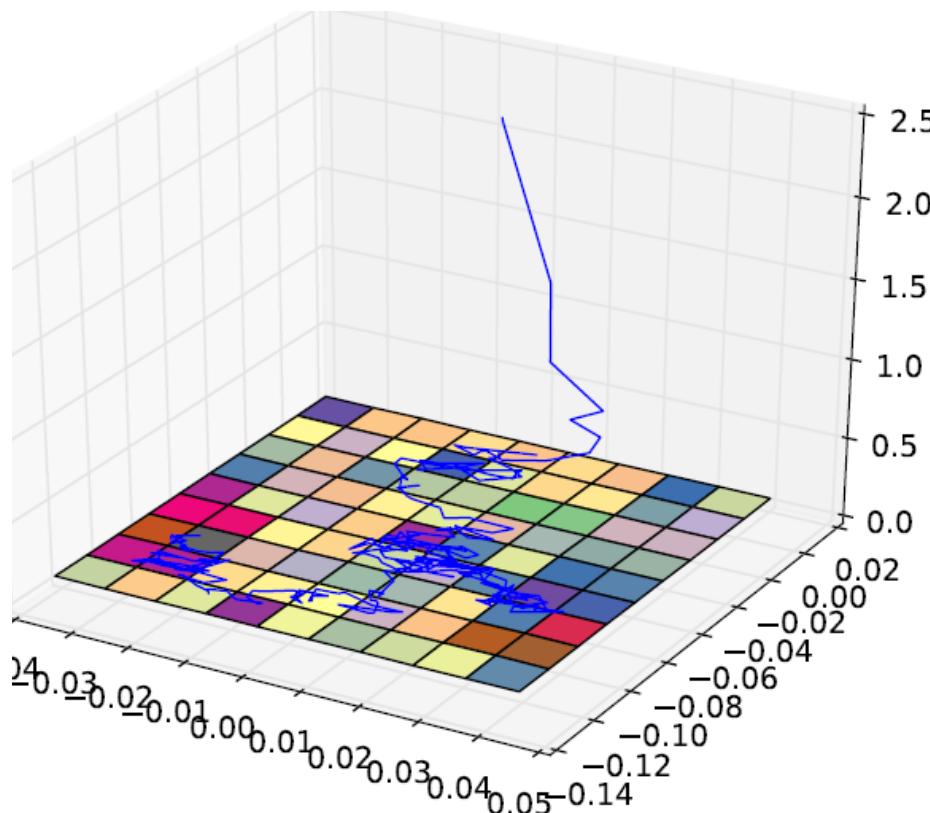
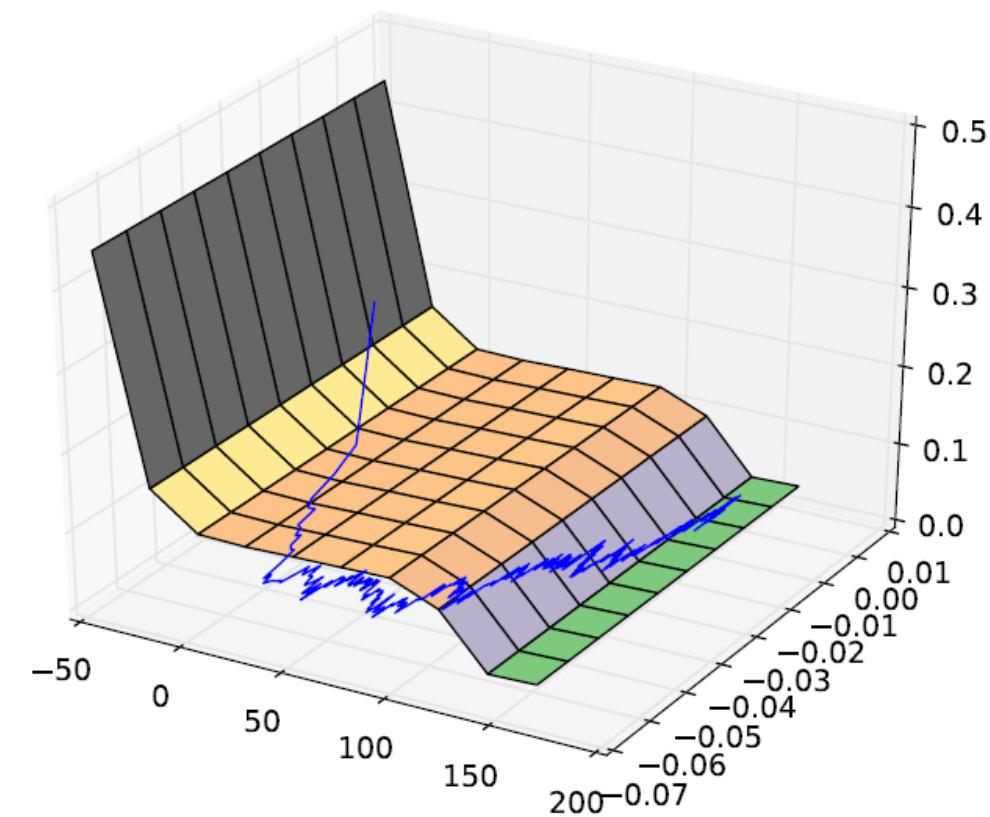


two random starting points

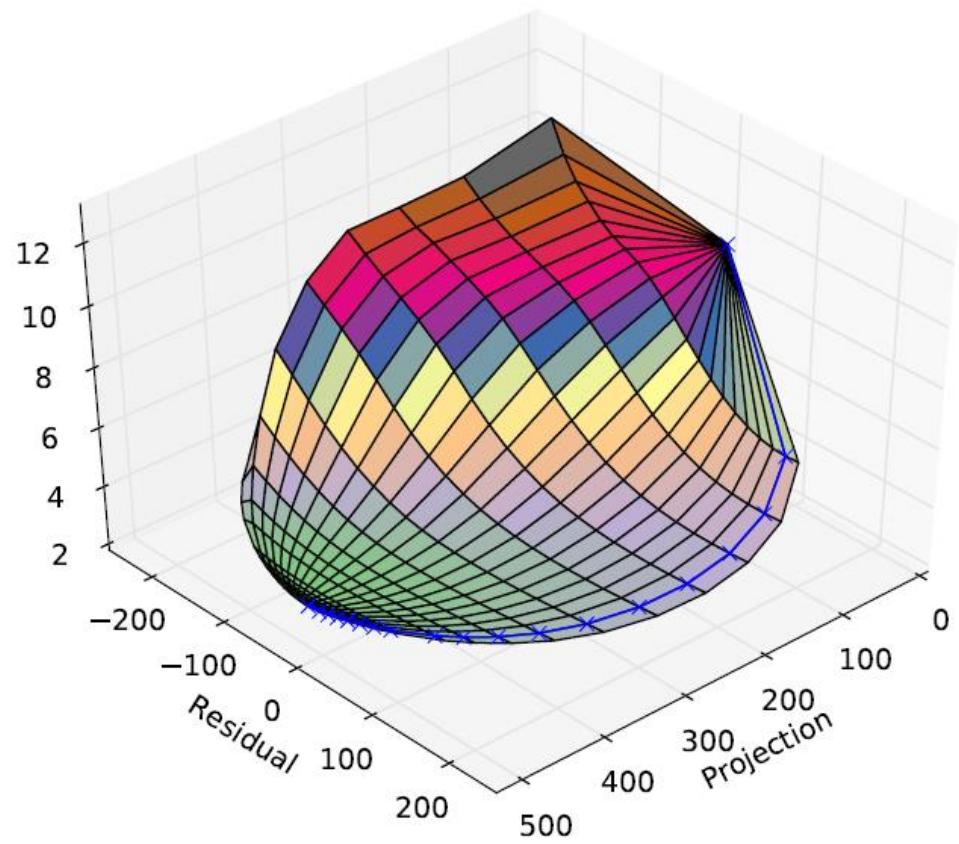
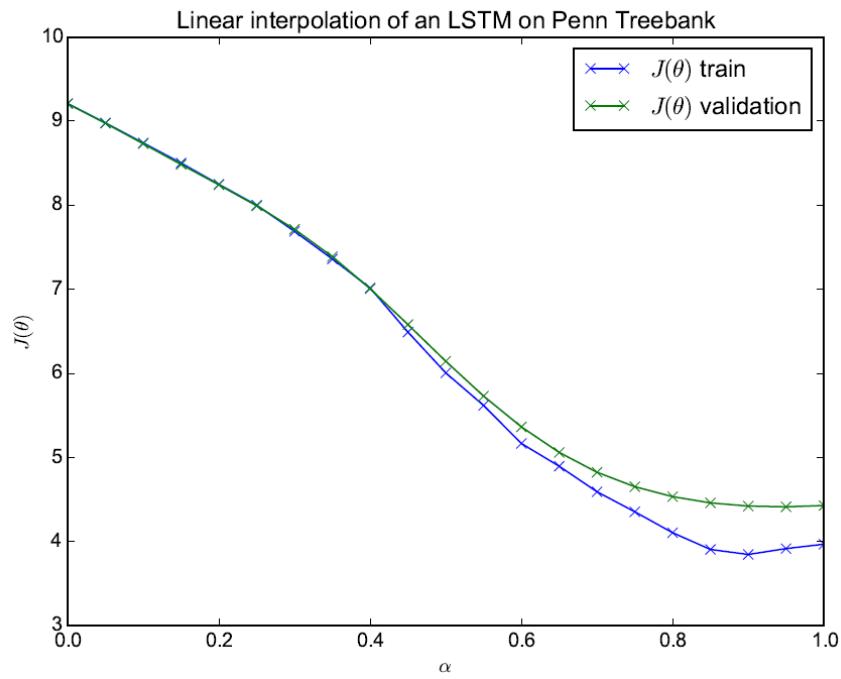


two “solutions”



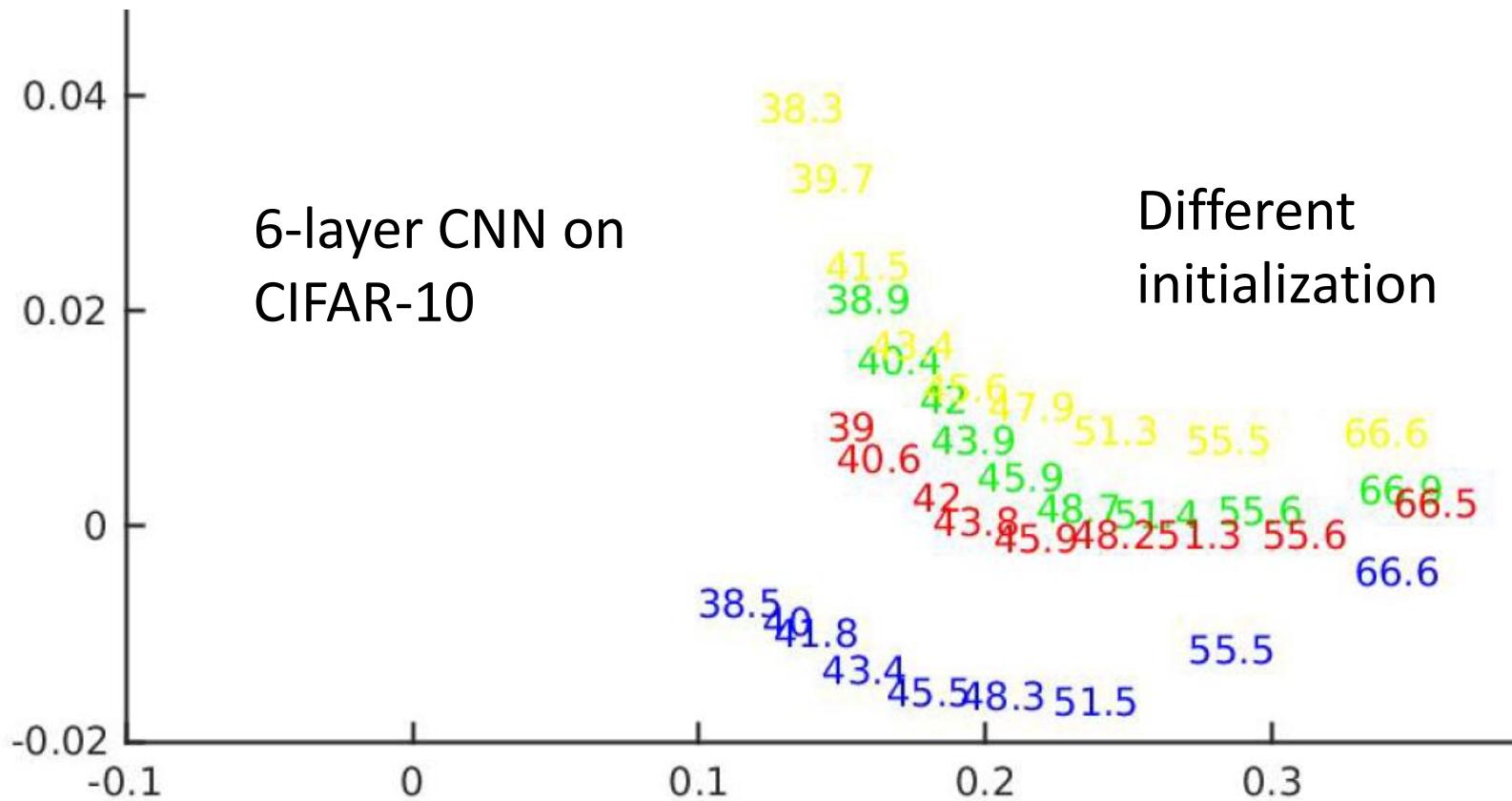


Profile - LSTM



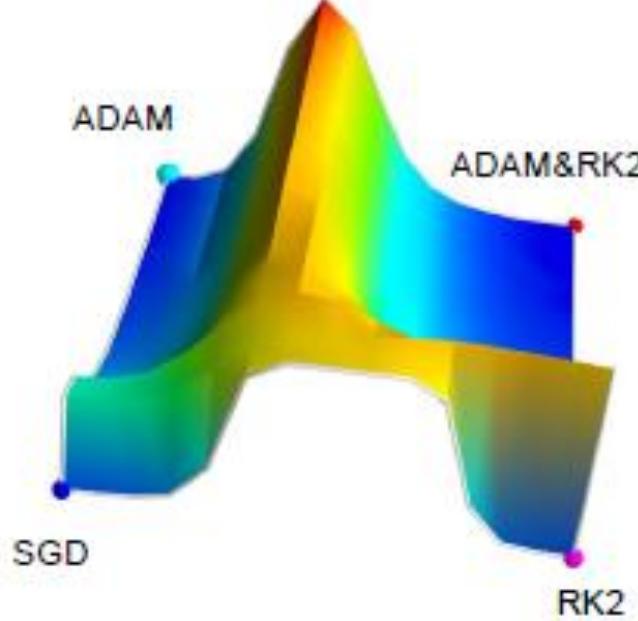
Training Processing

Different initialization / different strategies usually lead to similar loss (there are some exceptions).

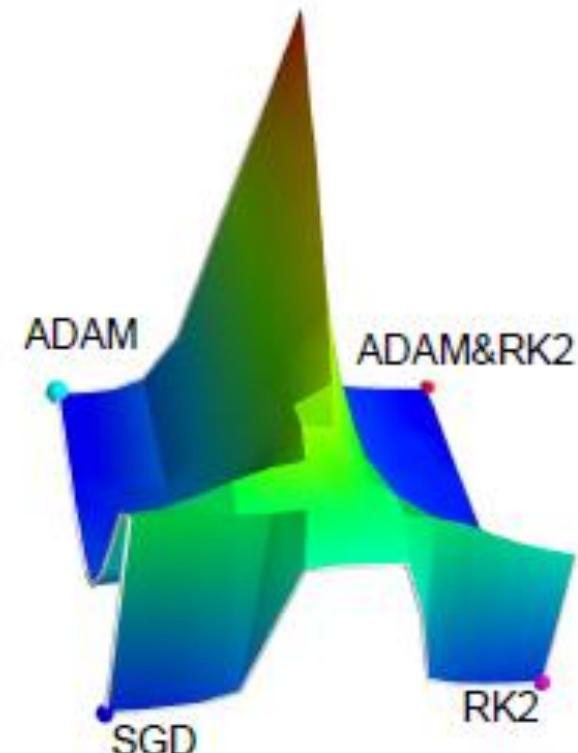


Training Processing

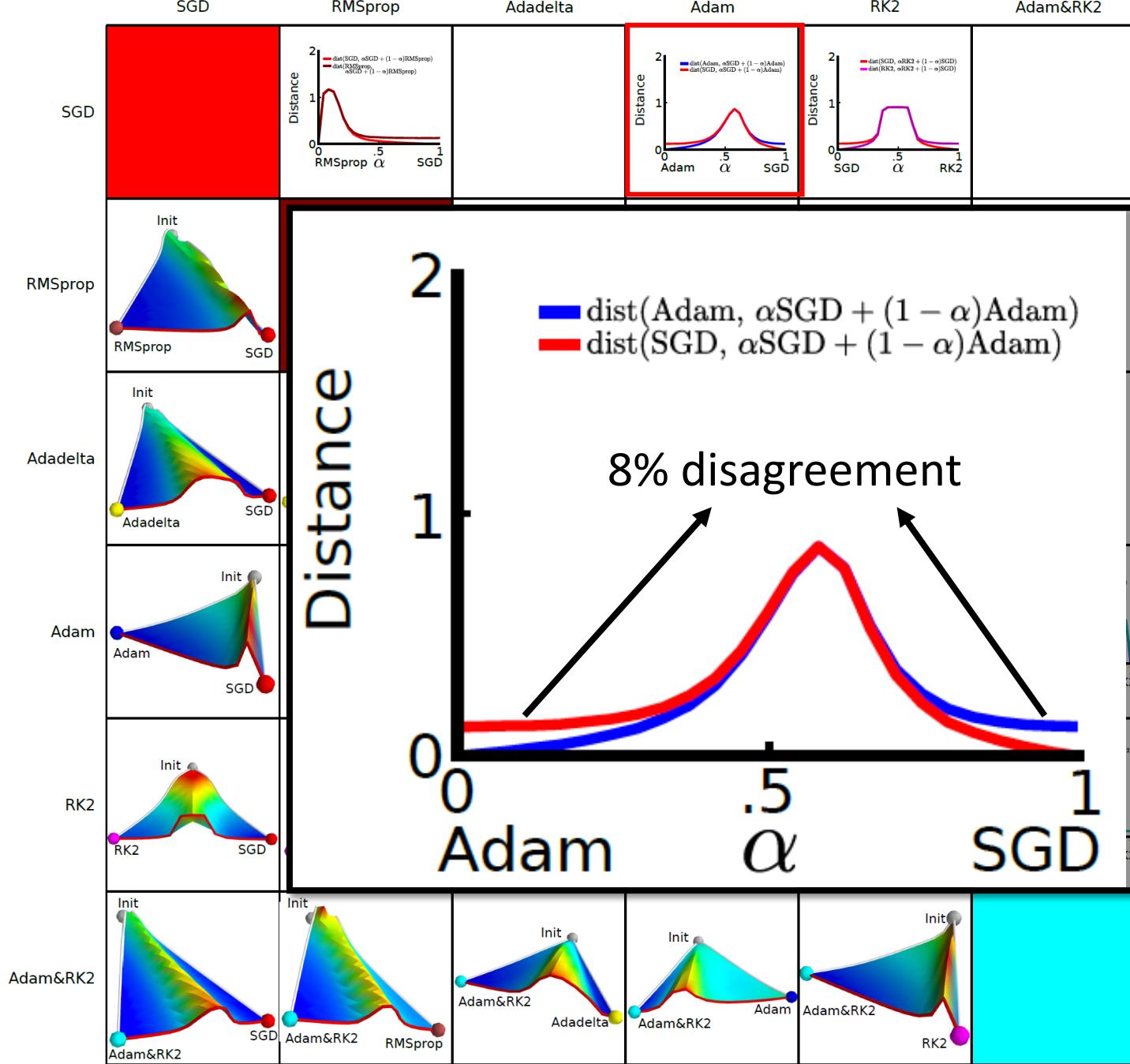
- Different strategies (the same initialization)



(a) NIN, CIFAR10

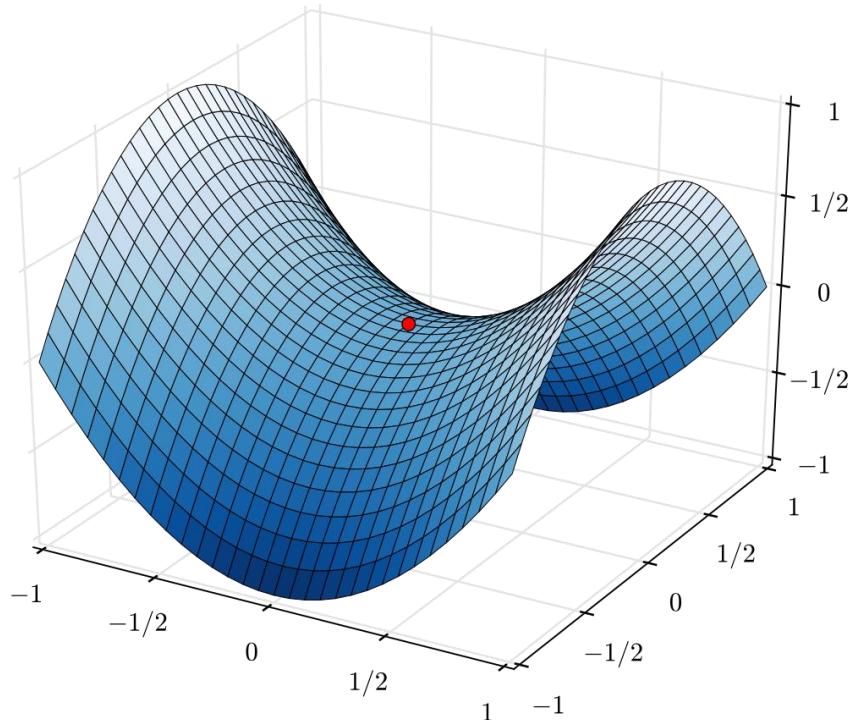


(b) VGG, CIFAR10

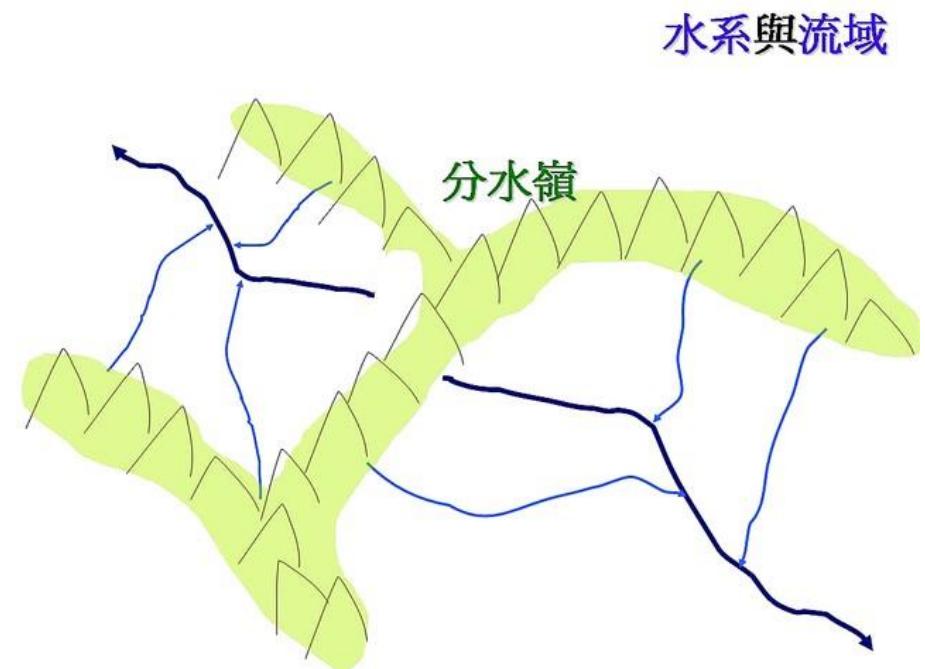


(c) VGG, CIFAR10

Training Processing



何時分道揚鑣？



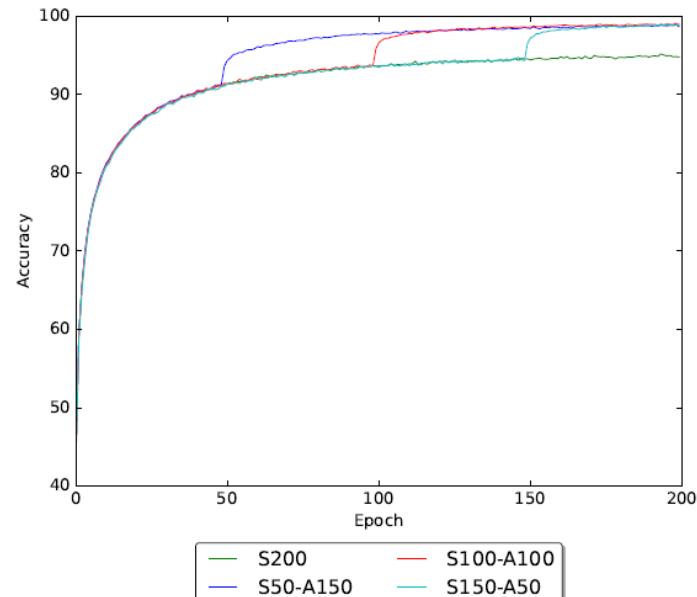
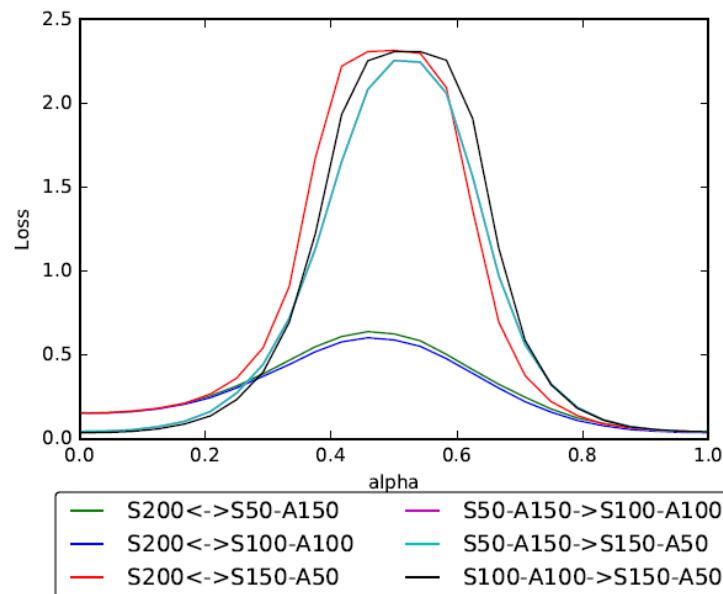
Different training strategies



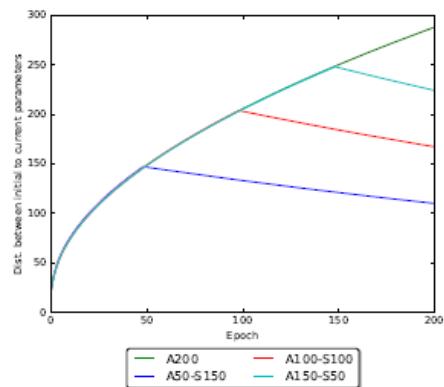
Different basins

Training Processing

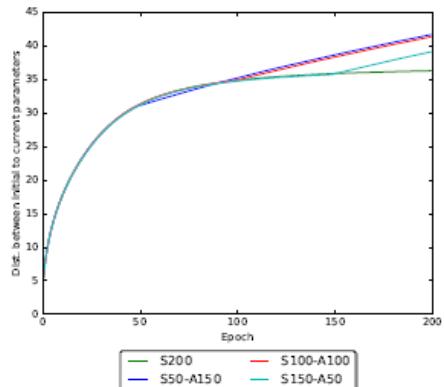
- Training strategies make difference at all stages of training



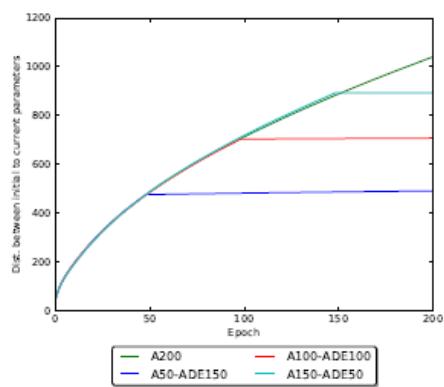
(b) NIN: Switching from SGD (S, $\eta = .1$) to Adam (A, $\eta = .0001$).



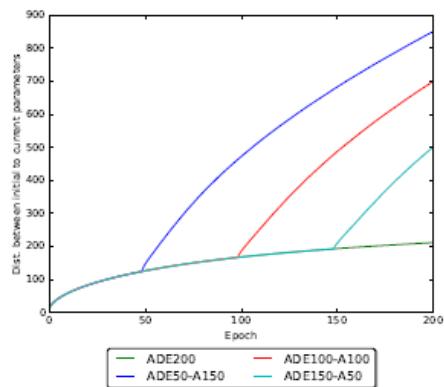
(a)



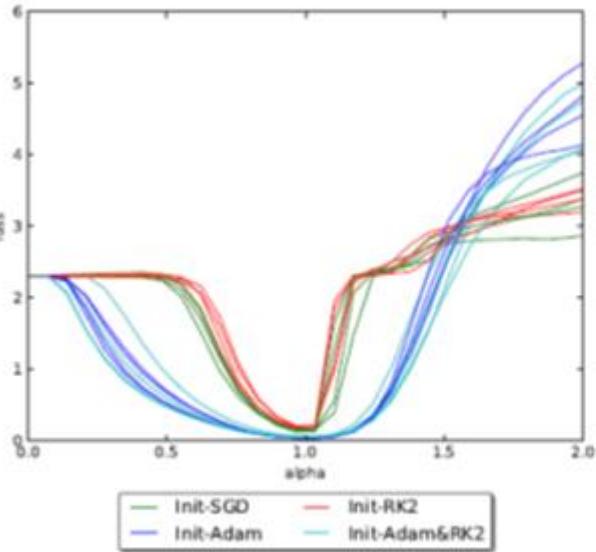
(b)



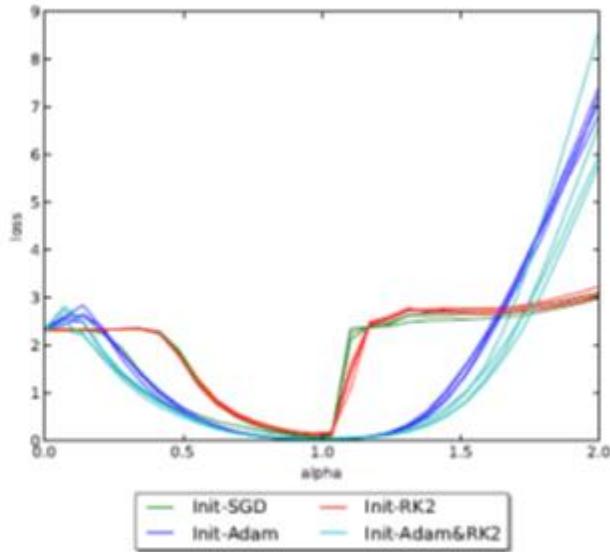
(c)



(d)



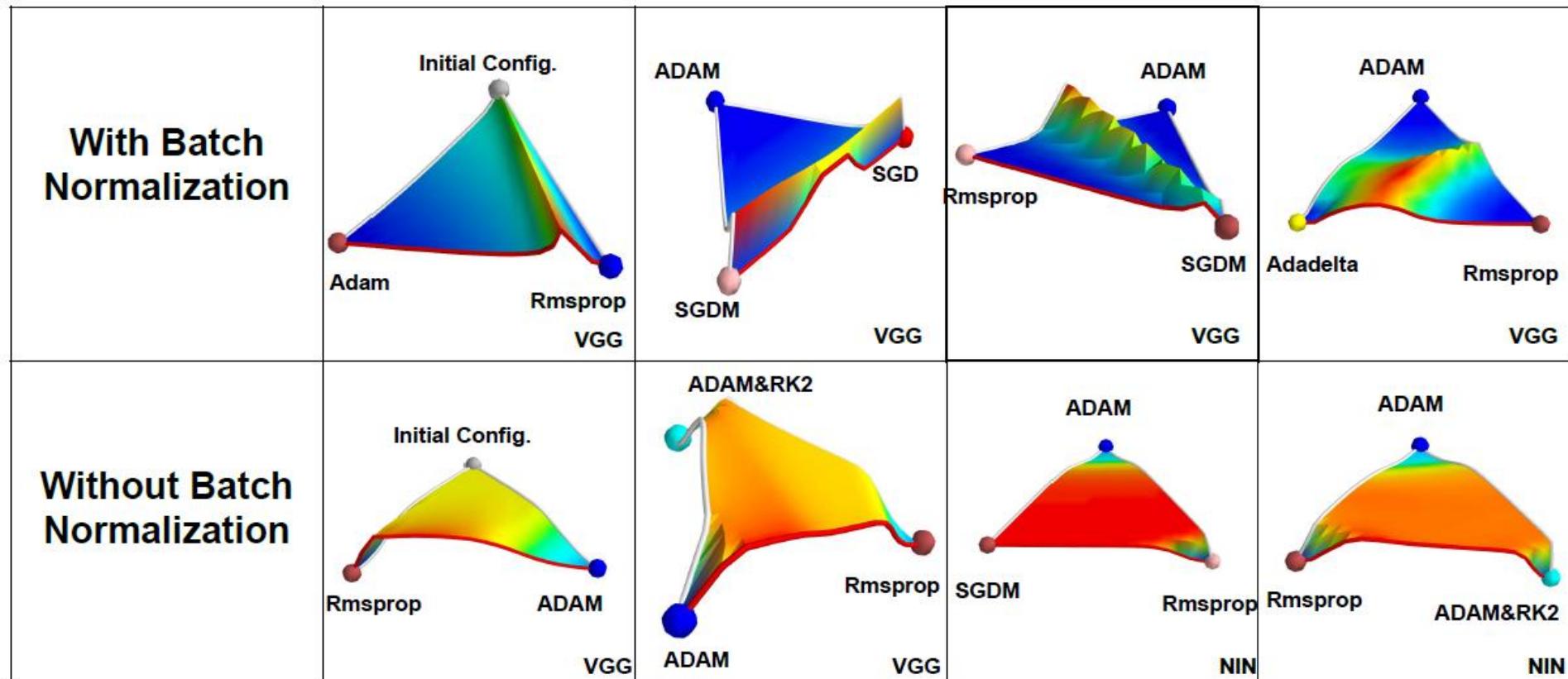
(a) CIFAR10, NIN



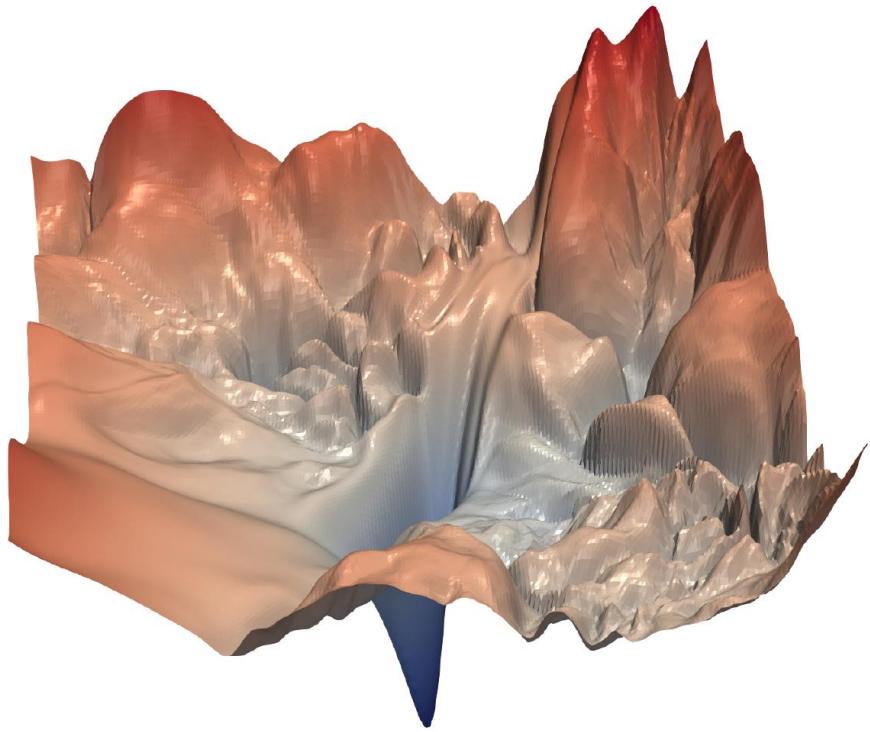
(b) CIFAR10, VGG

Larger basin
for Adam

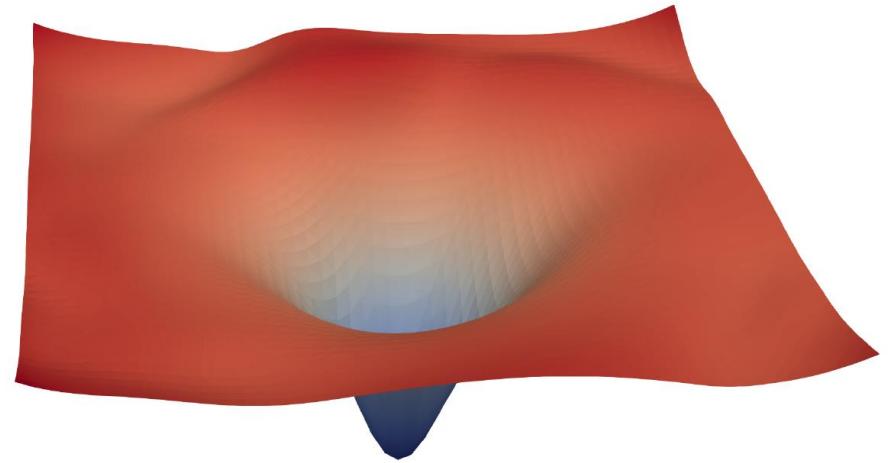
Batch Normalization



Skip Connection



(a) without skip connections



(b) with skip connections

Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The vertical axis is logarithmic to show dynamic range. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.

Reference

- Ian J. Goodfellow, Oriol Vinyals, Andrew M. Saxe, "Qualitatively characterizing neural network optimization problems", ICLR 2015
- Daniel Jiwoong Im, Michael Tao, Kristin Branson, "An Empirical Analysis of Deep Network Loss Surfaces", arXiv 2016
- Qianli Liao, Tomaso Poggio, "Theory II: Landscape of the Empirical Risk in Deep Learning", arXiv 2017
- Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, Tom Goldstein, "Visualizing the Loss Landscape of Neural Nets", arXiv 2017

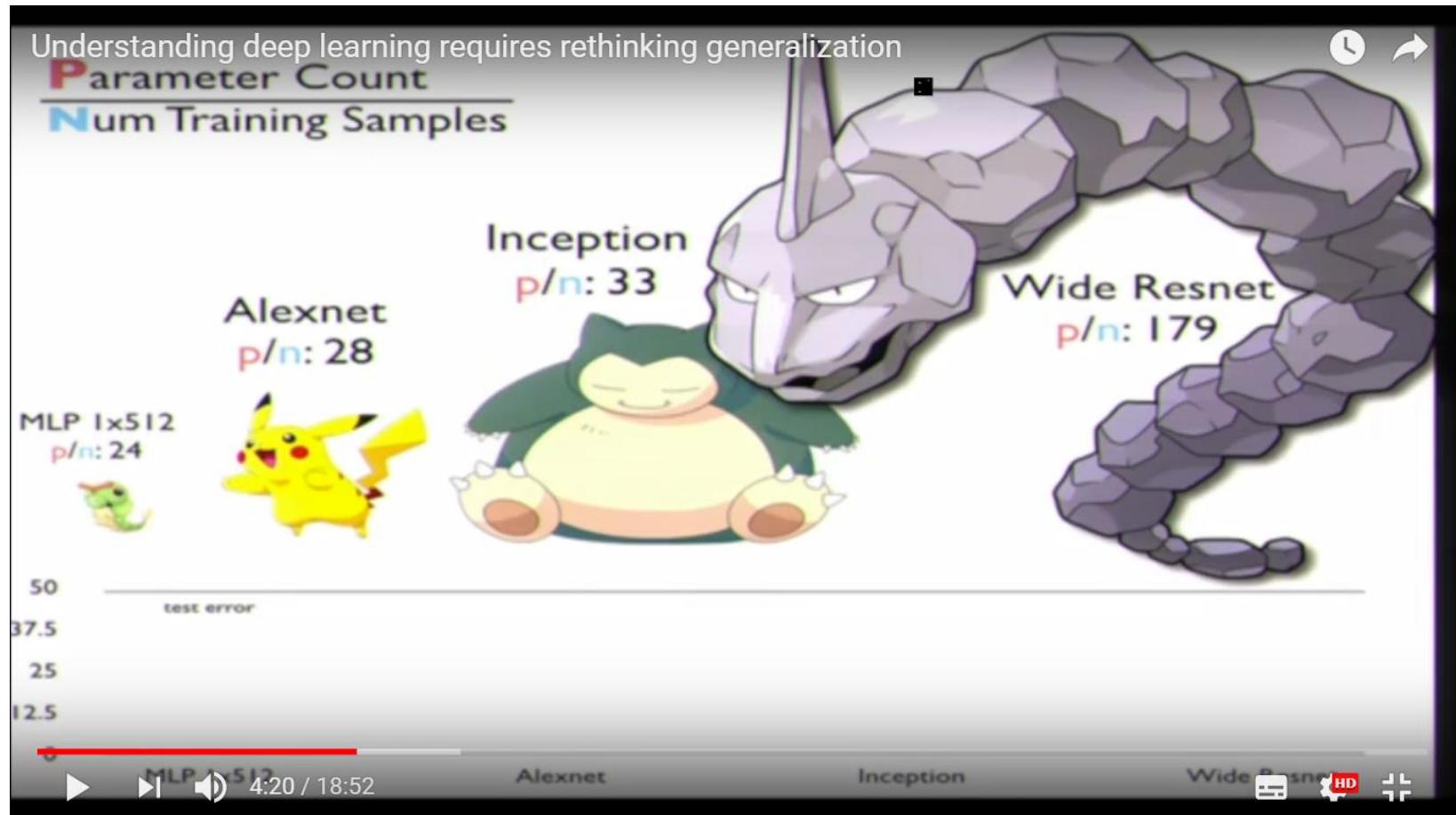
Concluding Remarks

Concluding Remarks

- Deep linear network is not convex, but all the local minima are global minima.
 - There are saddle points which are hard to escape
- Deep network has local minima.
 - We need more theory in the future
- Conjecture:
 - When training a larger network, it is rare to meet local minima.
 - All local minima are almost as good as global
- We can try to understand the error surface by visualization.
 - The error surface is not as complexed as imagined.

Generalization Ability

We use very large network today



Source of image: <https://www.youtube.com/watch?v=kCj51pTQPKI>

Generalization Gap

No matter the data distribution

With probability $1 - \delta$

$$E_{train} \leq E_{test} \leq E_{train} + \Omega(R, M, \delta)$$

Smaller δ , larger Ω

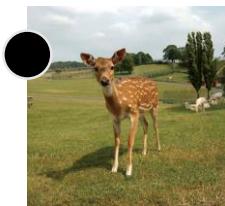
R is the number of training data  Larger R , smaller Ω

M is the “capacity” of your model
 (“size” of the function set)  Larger M , larger Ω

How to measure the “capacity”?

VC dimension (d_{VC})

Given 3
data points



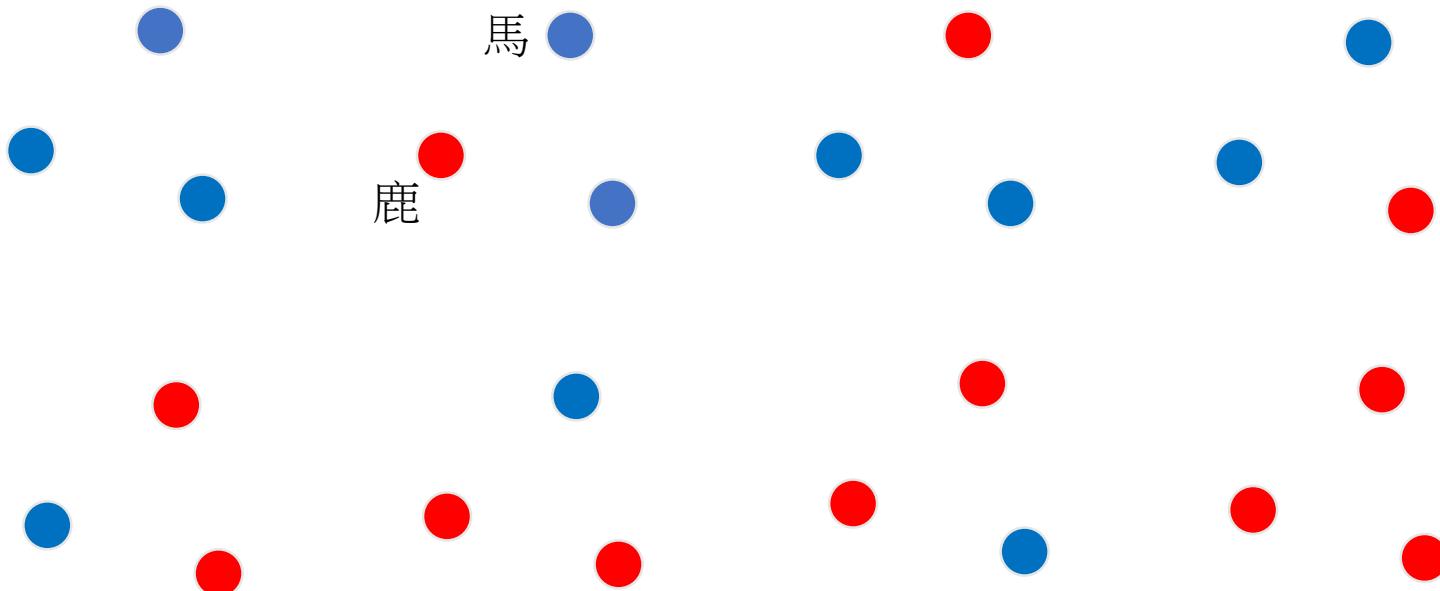
Random label (故意亂教)

Model M can always achieve 0%
error rate

(亂教 Model M 都學得會)

VC dimension (d_{VC}) of
Model M ≥ 3

e.g. linear model



Random label (故意亂教)

There are some cases linear model can not learn.

(來亂的，所以學不會)

VC dimension (d_{VC}) of
Linear Model < 4



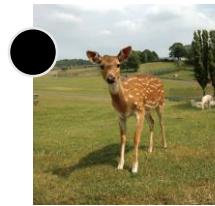
鹿



馬



馬

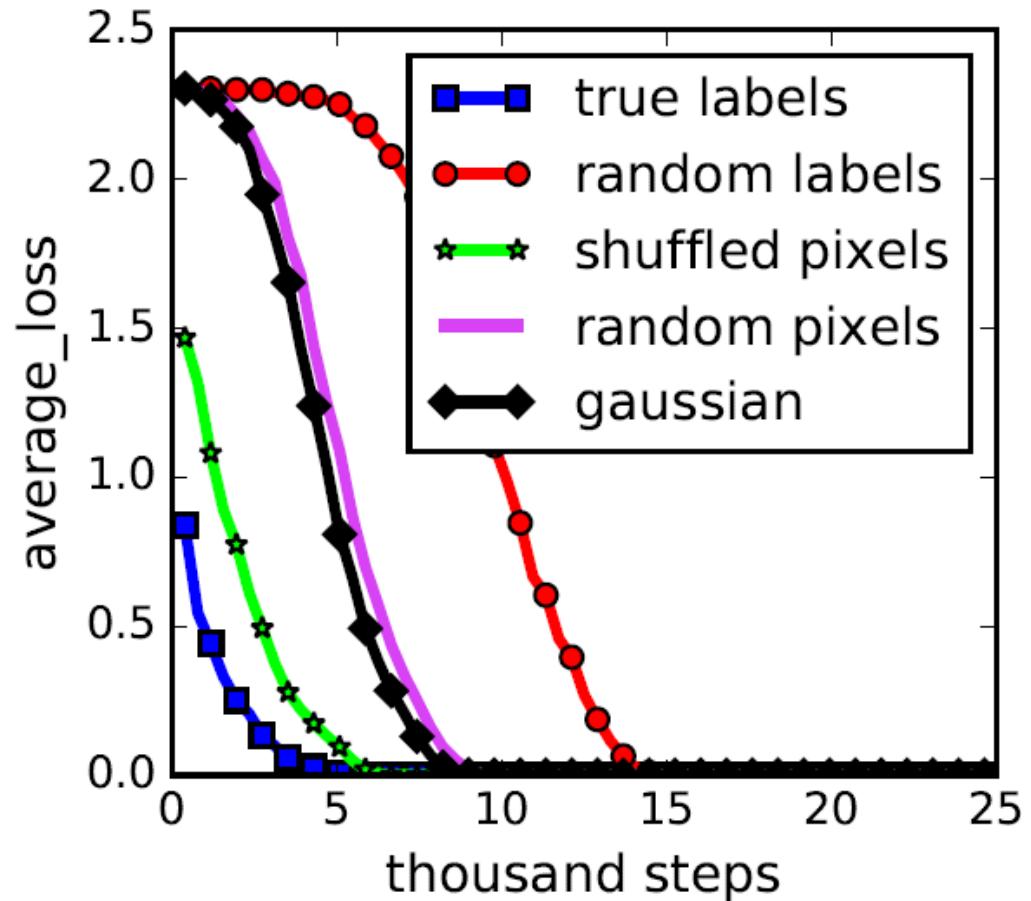


鹿

Given 4 data points

What is the capacity of deep models?

Inception model
on the CIFAR10



Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, Oriol Vinyals,
"Understanding deep learning requires rethinking generalization", ICLR 2017

Overparameterized Network?

No matter the data distribution

With probability $1 - \delta$

$$E_{test} \leq E_{train} + \Omega(R, M, \delta)$$

Smaller δ , larger Ω

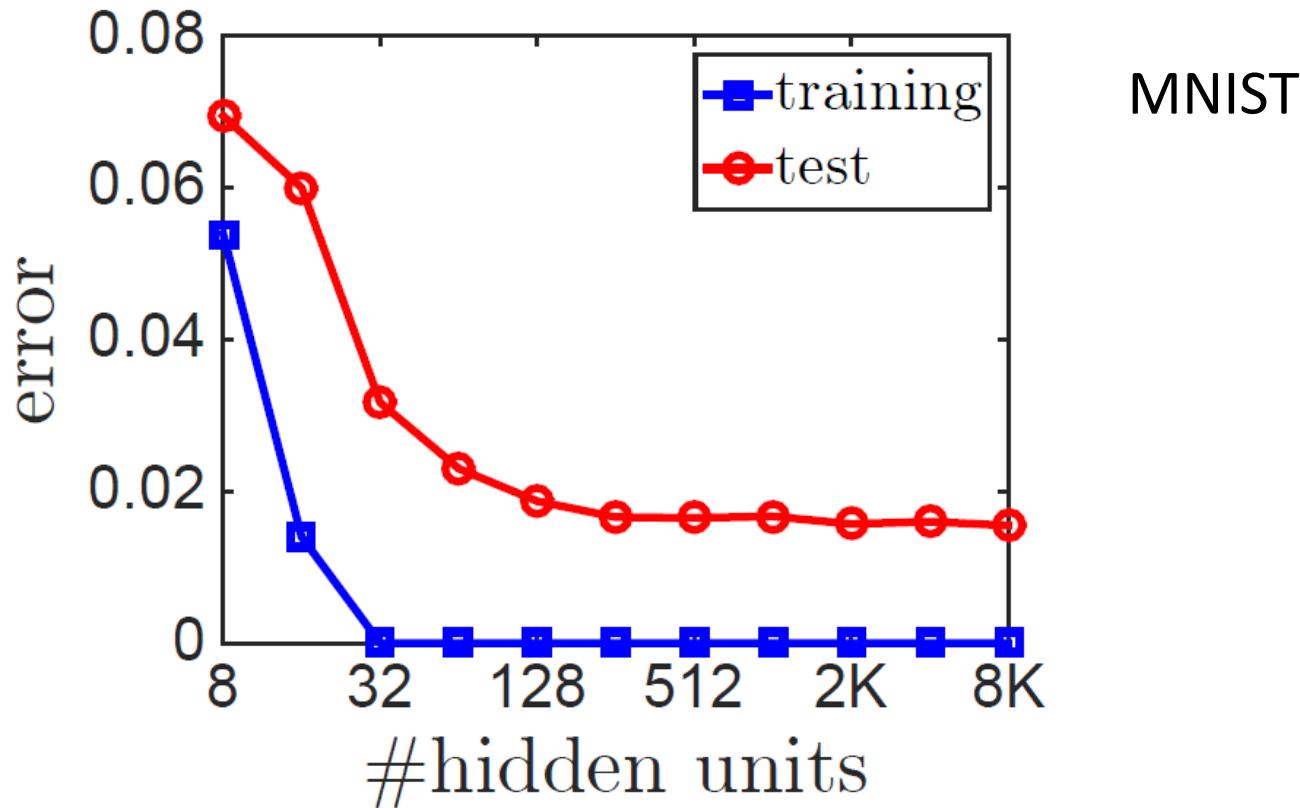
R is the number of training data  Larger R , smaller Ω

M is the “capacity” of your model
 (“size” of the function set)  Larger M , larger Ω

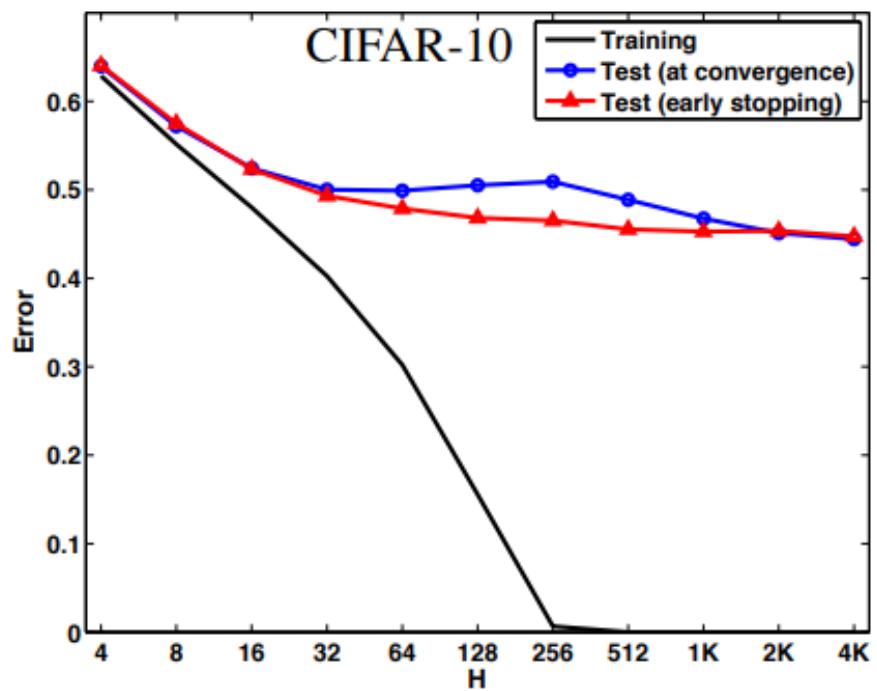
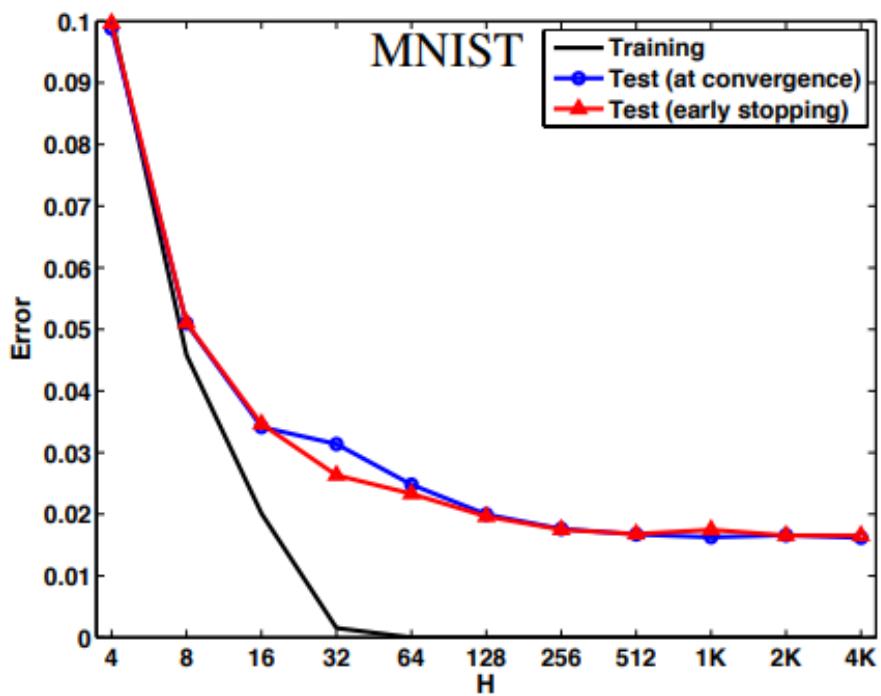
If two models have the same E_{train}  Select the one with
smaller capacity

Demo

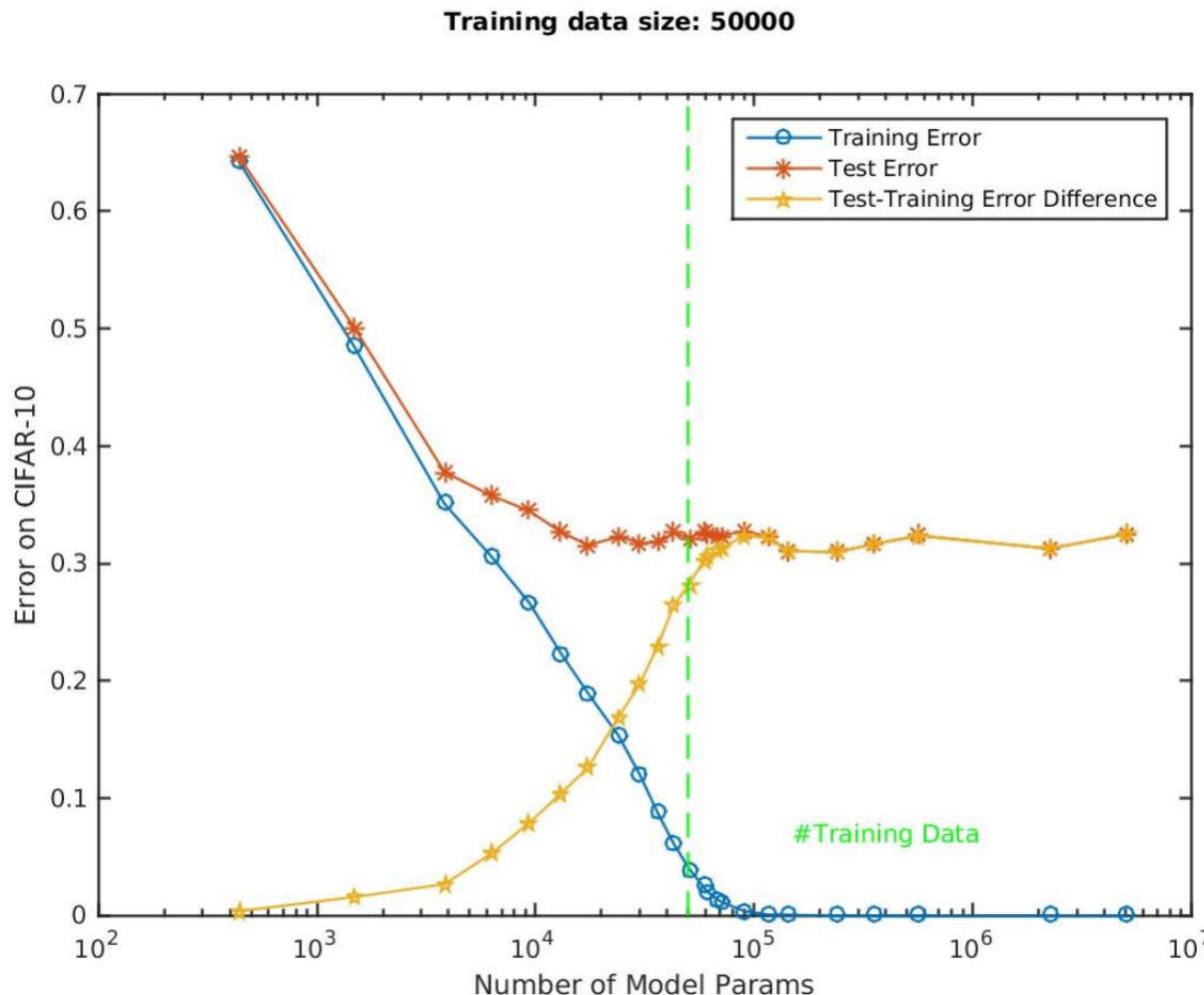
Overparameterized Network?



Overparameterized Network?



Overparameterized Network?



Generalization gap

0.8

0.6

0.4

100k

1M

10M

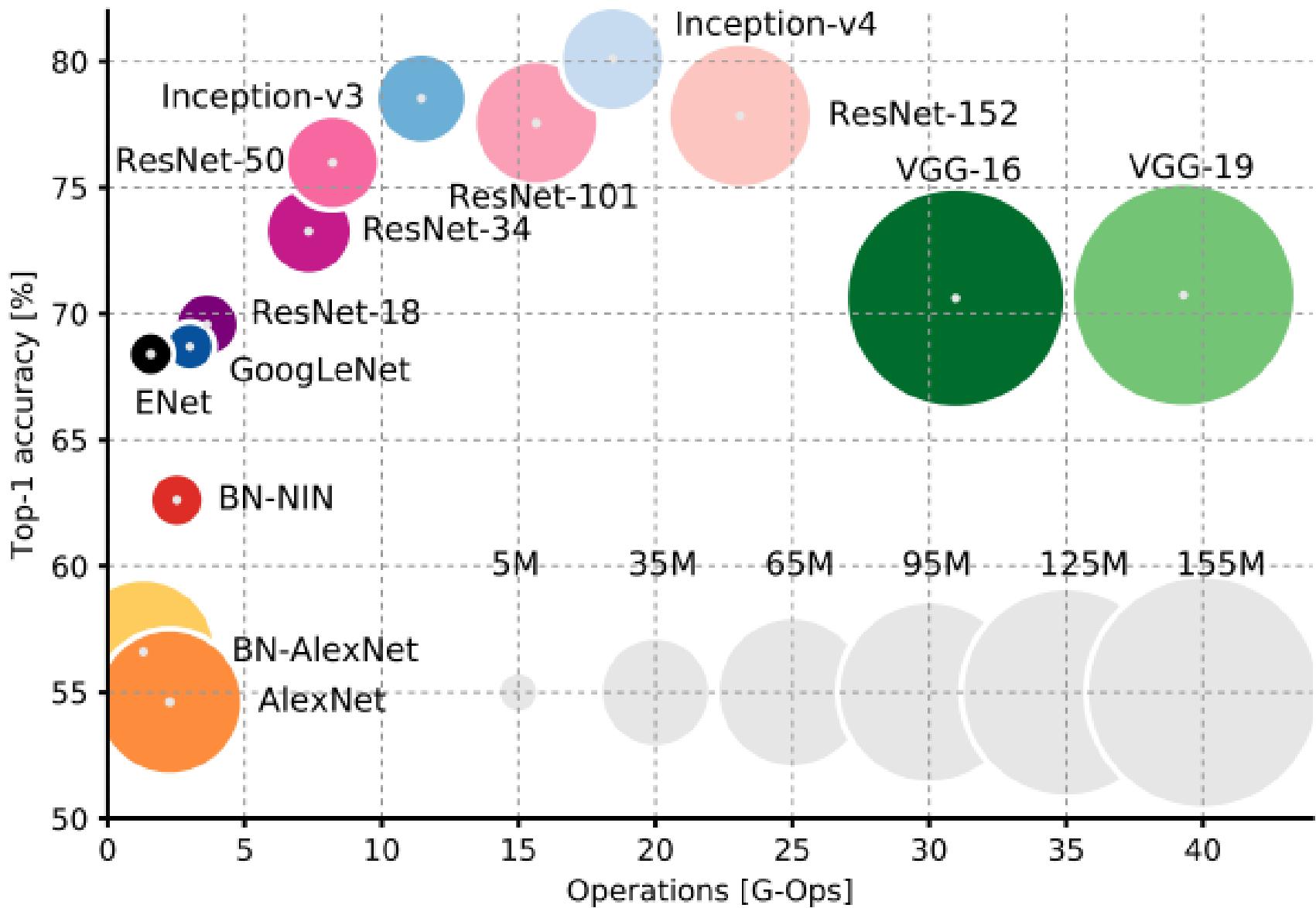
100M

1B

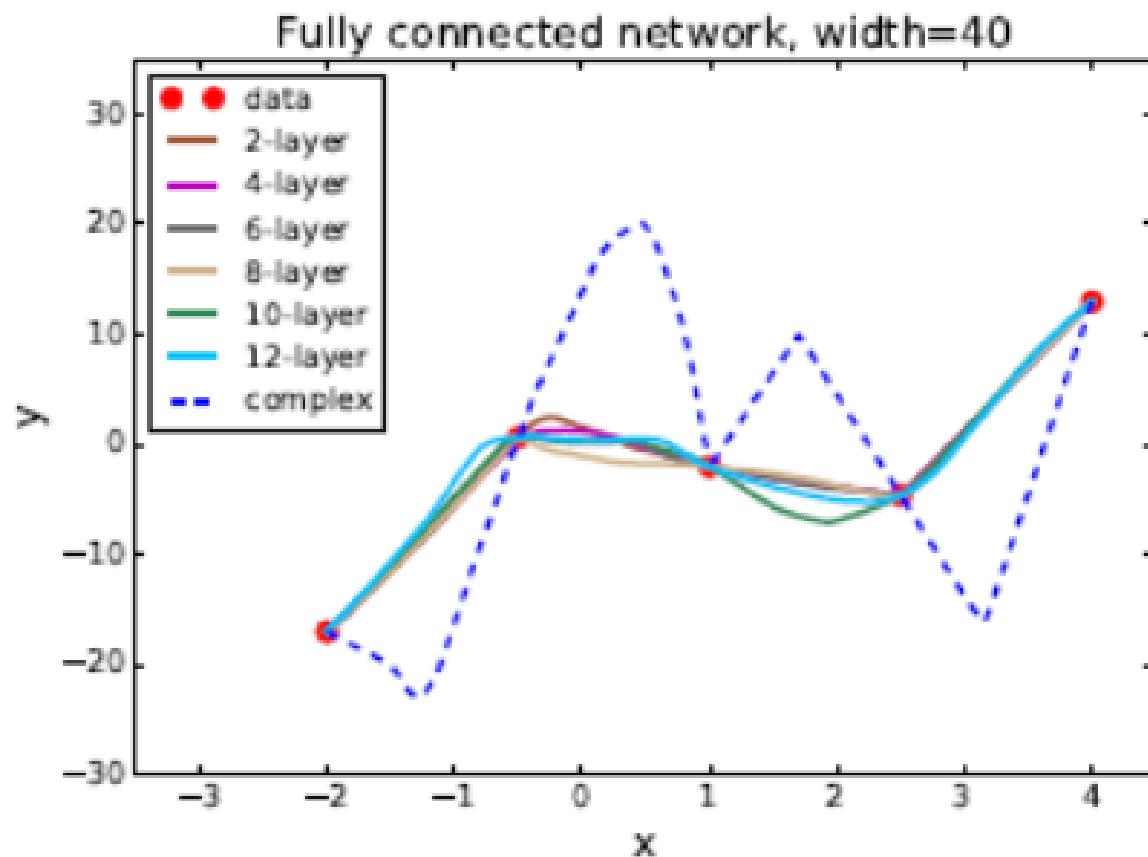
CIFIR-10, 100%
training accuracy

<https://arxiv.org/pdf/1802.08760.pdf>

Number of weights



Network regularizes itself?

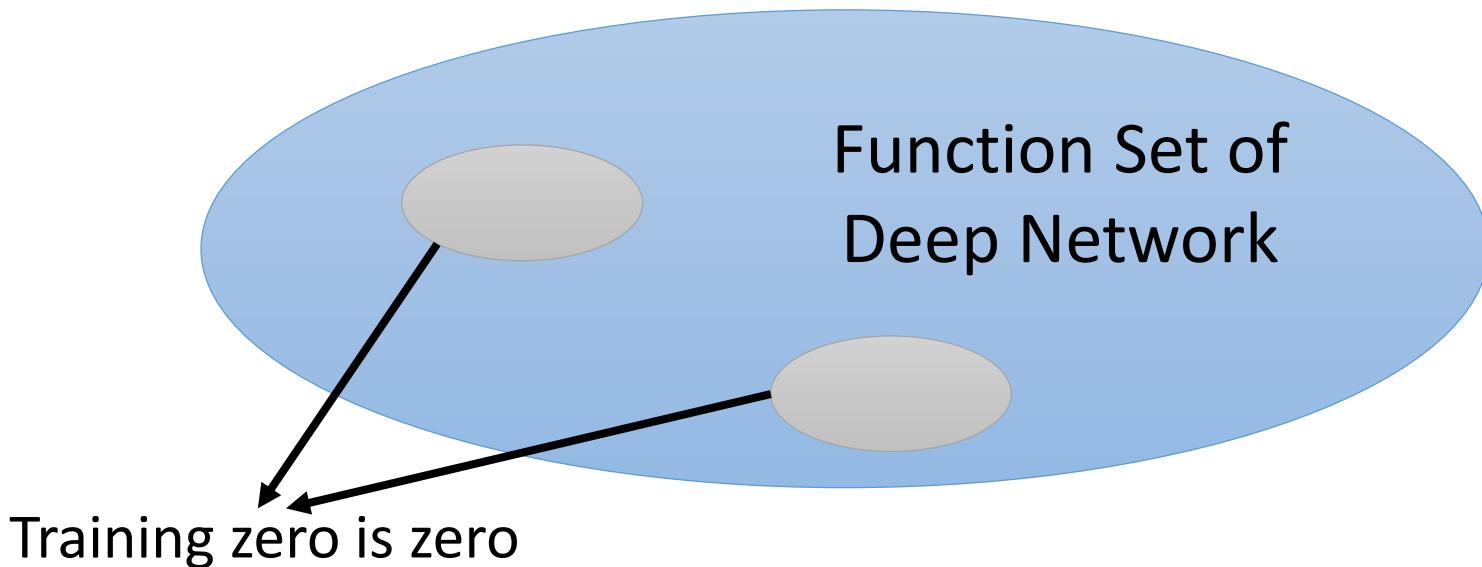
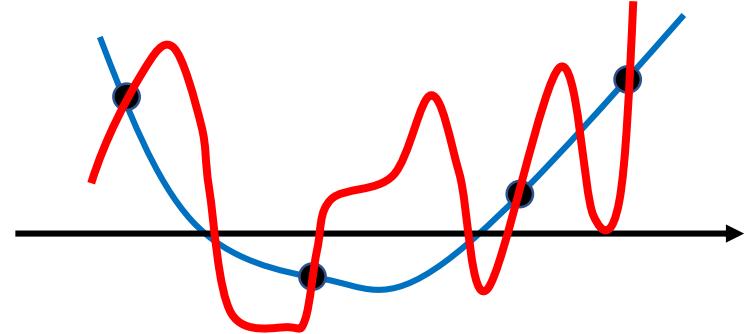


Concluding Remarks

- The capacity of deep model is large.
- However, it does not overfit!
- The reason is not clear yet.

Indicator of Generalization

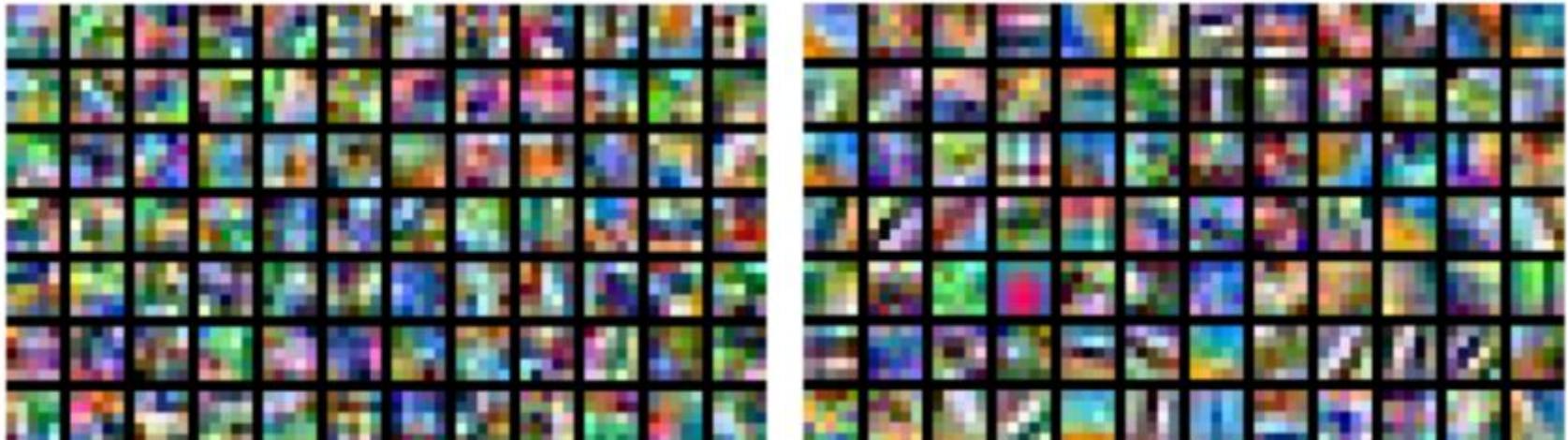
Introduction



- If many global optimums can zero training errors, which one can obtain generalized results?
- Use the indicator to find solution that generalizes well.
- ***Sharpness and Sensitivity***

Brute-force Memorization ?

- Real labels v.s. random labels

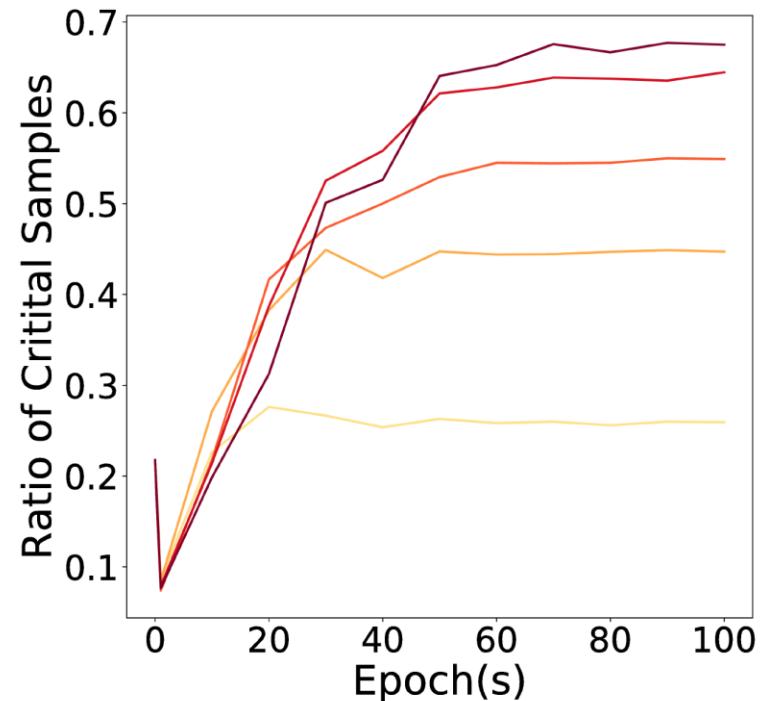
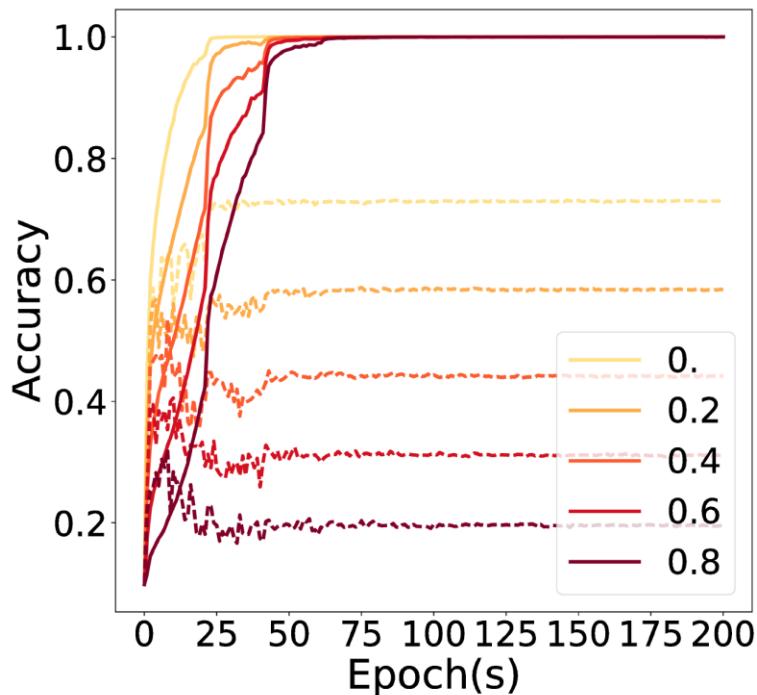


First layer of CIFAR-10

<https://arxiv.org/pdf/1706.05394.pdf>

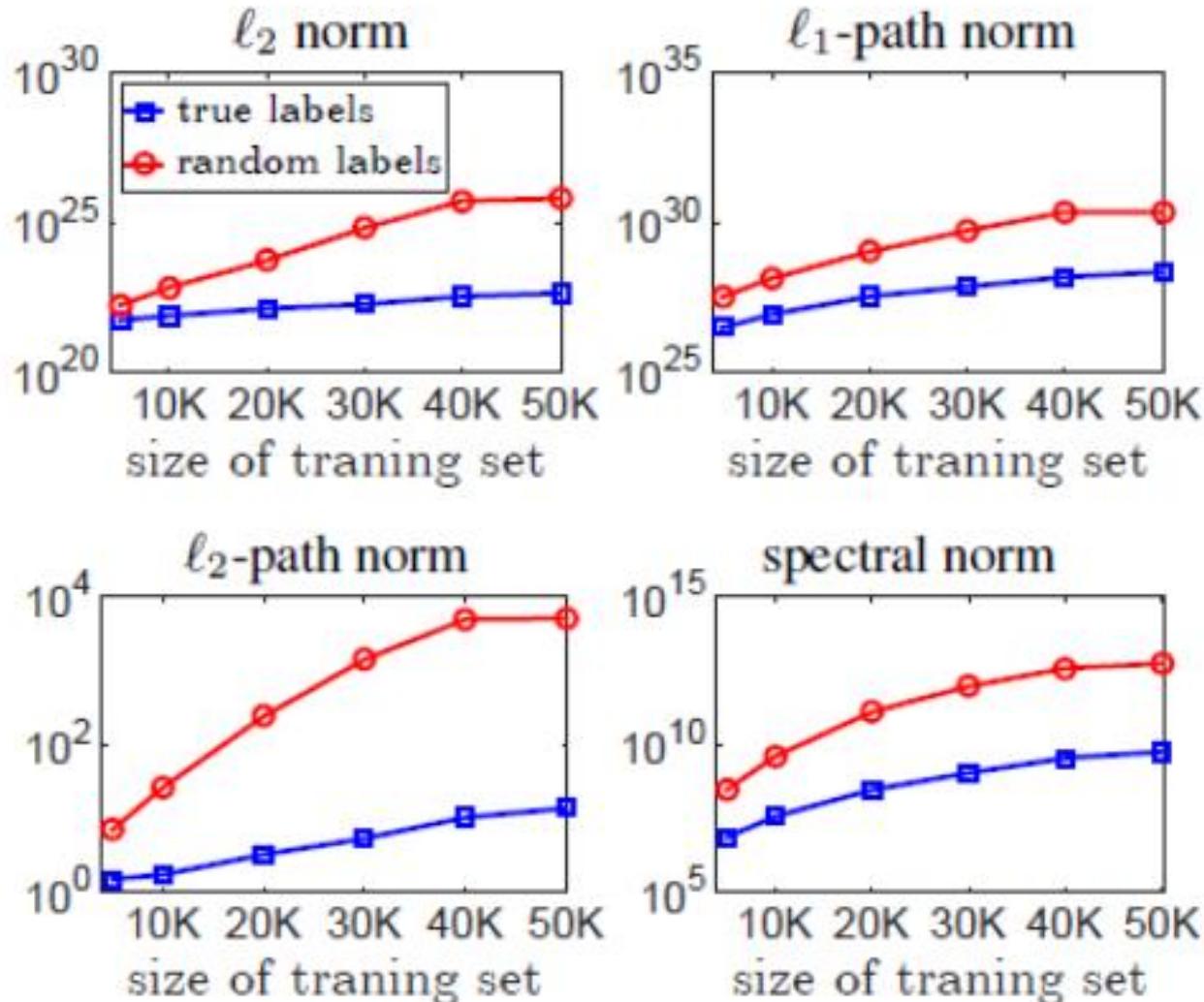
Brute-force Memorization ?

- Simple pattern first, then memorize exception



(b) Noise added on classification labels.

Brute-force Memorization ?



Sensitivity

Jacobian Matrix

$$y = f(x) \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$\frac{\partial y}{\partial x} = \left. \begin{array}{c} \text{size of } y \\ \text{size of } x \end{array} \right\}$$

Example

$$\begin{bmatrix} x_1 + x_2 x_3 \\ 2x_3 \end{bmatrix} = f \left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \right) \quad \frac{\partial y}{\partial x} = []$$

Sensitivity

- Given a network f , the sensitivity of a data point x is the Frobenius norm of the Jacobian

$$y = f(x) \quad \frac{\partial y}{\partial x} = \begin{bmatrix} \partial y_1 / \partial x_1 & \partial y_1 / \partial x_2 & \partial y_1 / \partial x_3 \\ \partial y_2 / \partial x_1 & \partial y_2 / \partial x_2 & \partial y_2 / \partial x_3 \end{bmatrix}$$

$$\text{Sensitivity of } x = \sqrt{\sum_i \sum_j \left(\frac{\partial y_j}{\partial x_i} \right)^2}$$

By the sensitivity of a test data x , we can predict the performance.

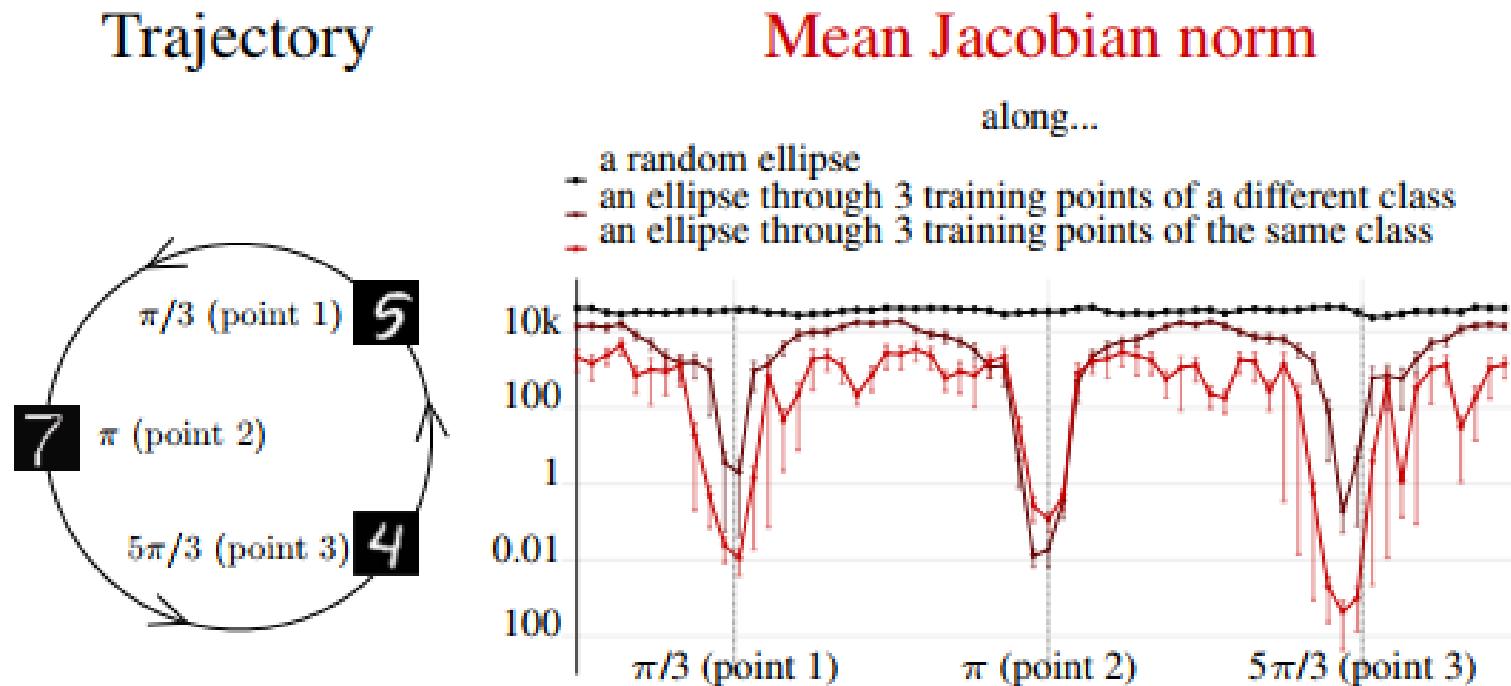
Without label

It is not surprise that sensitivity is related to generalization.

Regularization is kind of minimizing sensitivity.

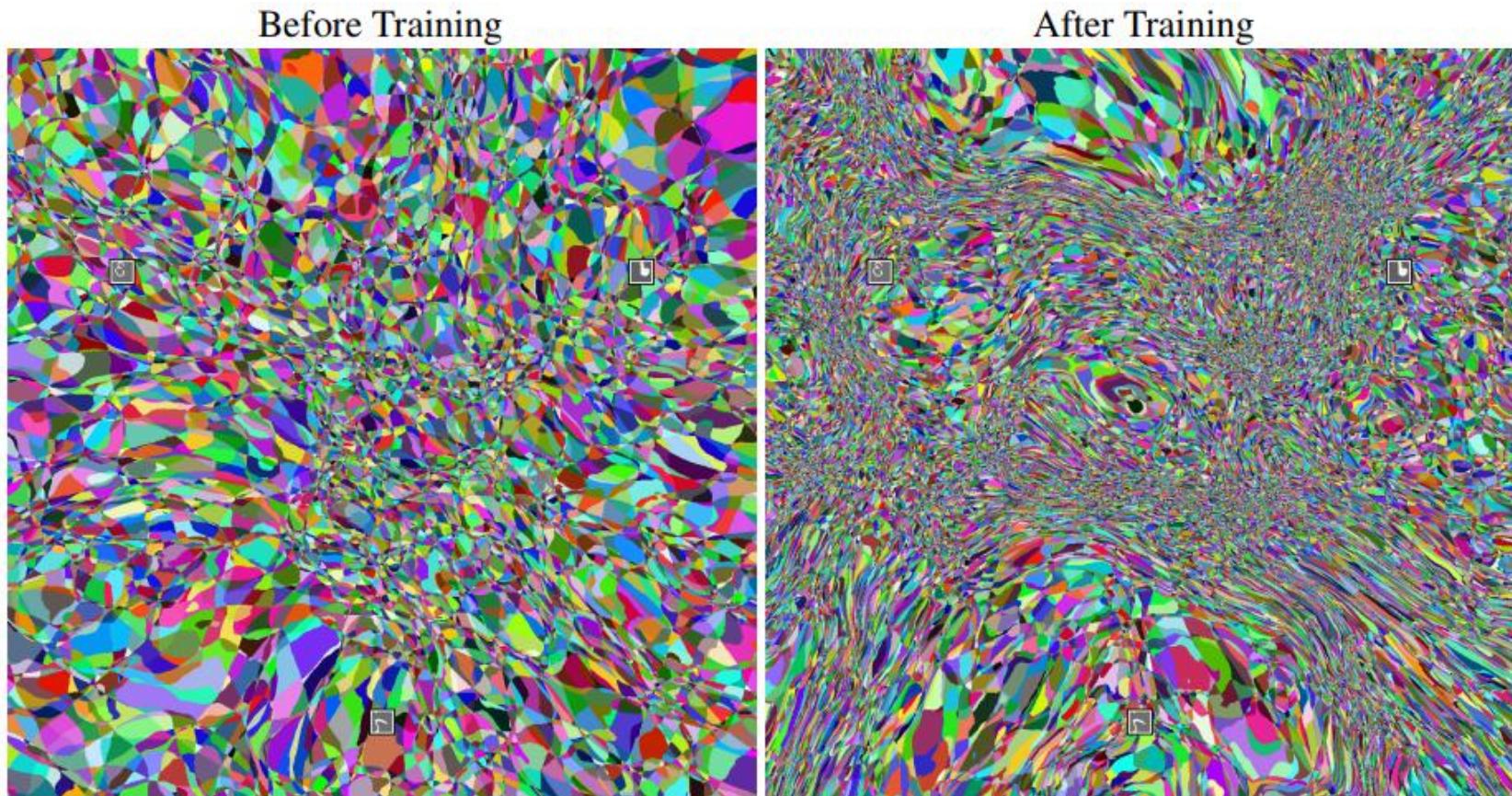
Sensitivity – Empirical Results

- Sensitivity on and off the training data manifold

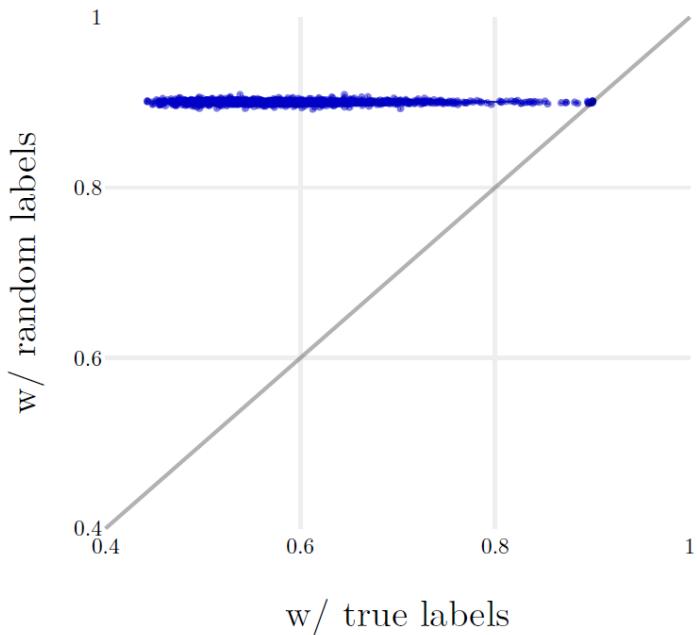


Sensitivity – Empirical Results

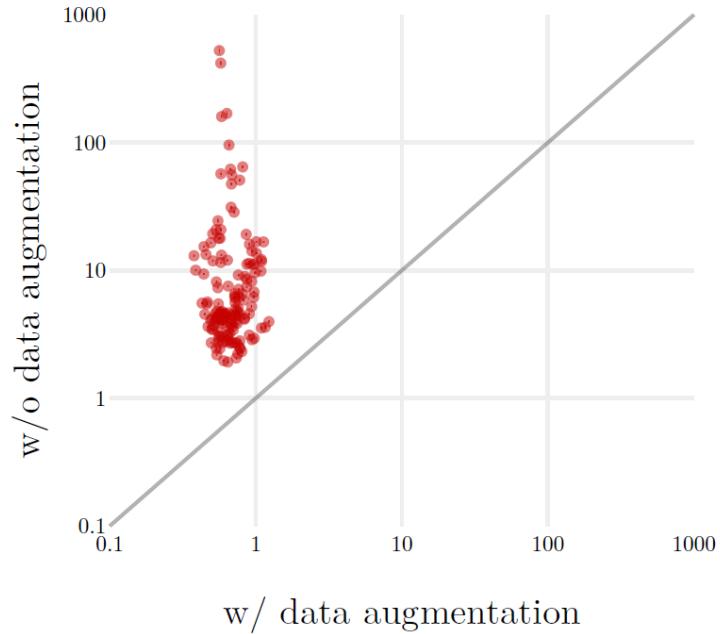
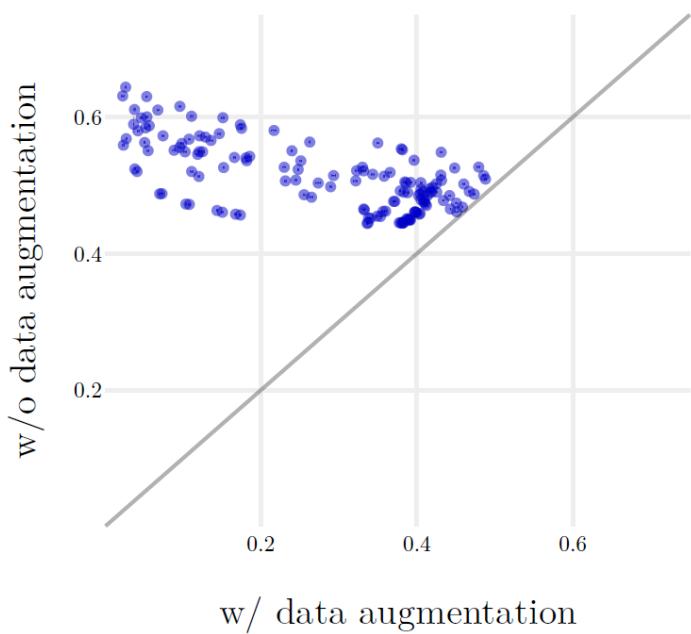
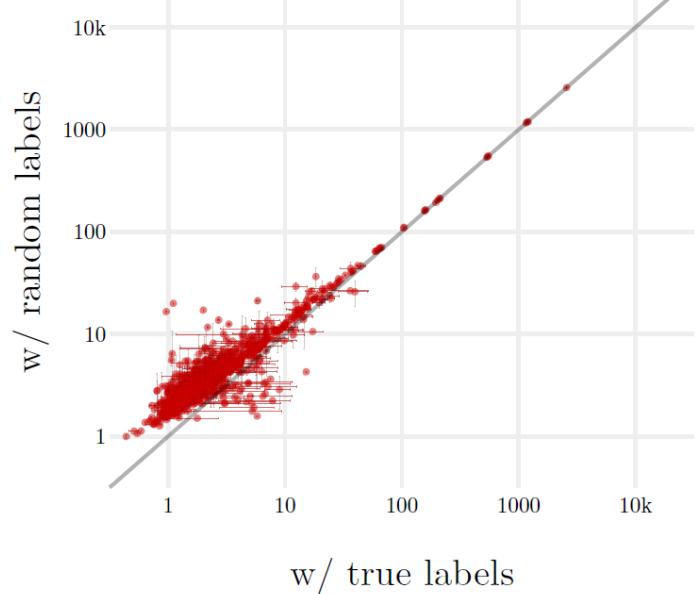
- Sensitivity on and off the training data manifold

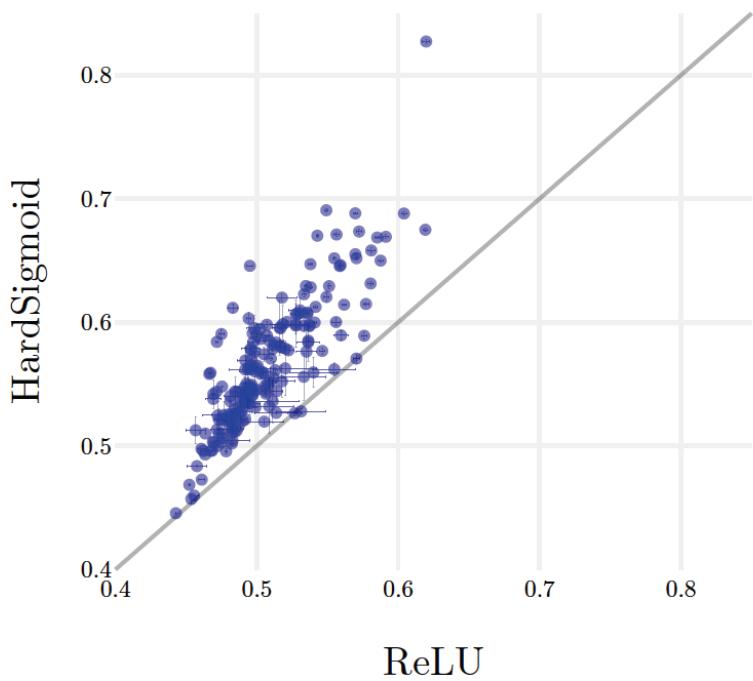


Generalization Gap

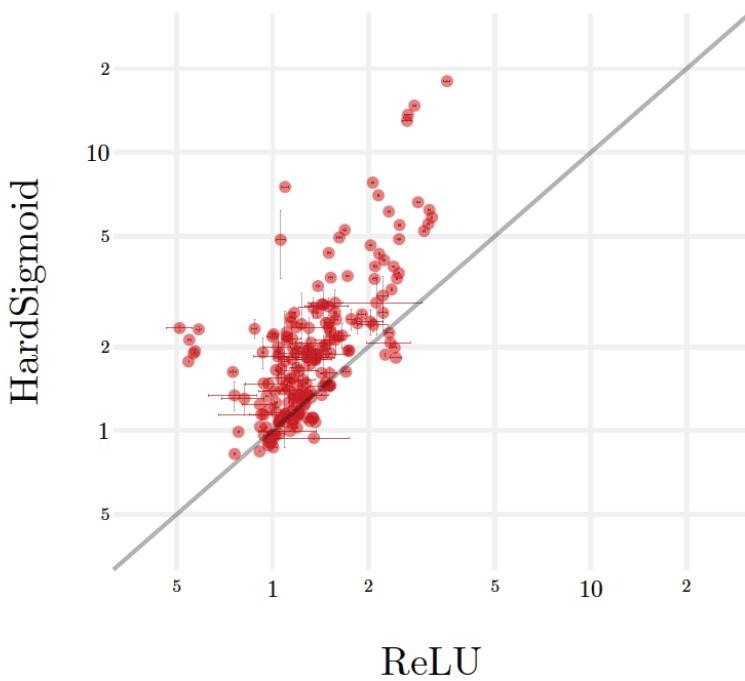


Jacobian norm

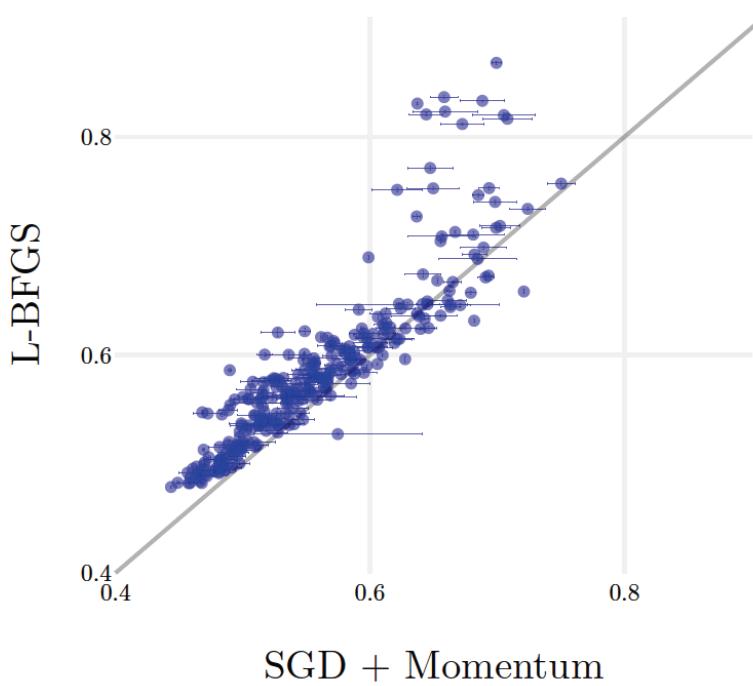




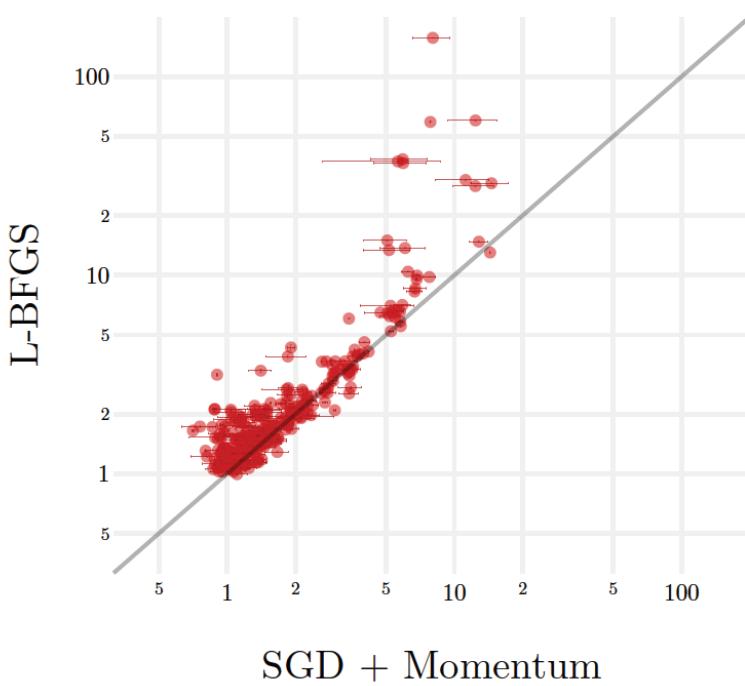
ReLU



ReLU



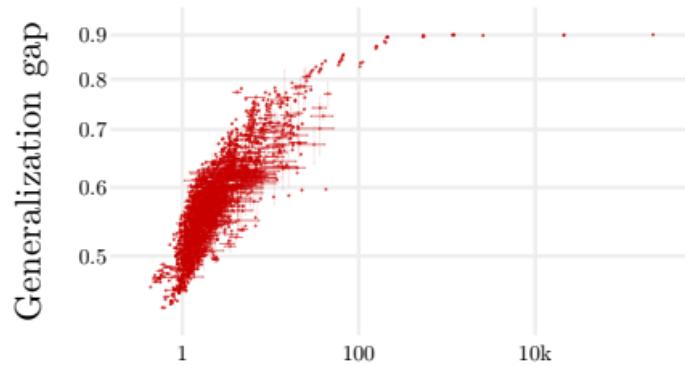
SGD + Momentum



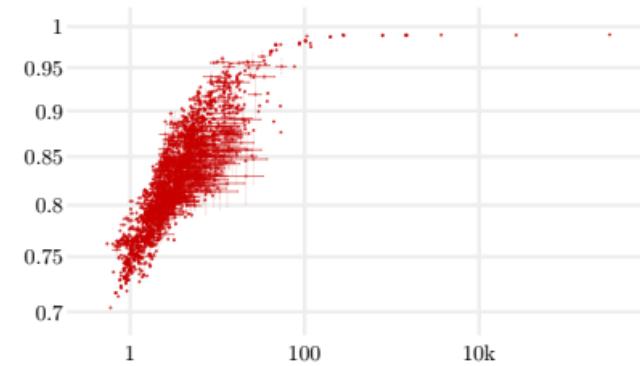
SGD + Momentum

Sensitivity v.s. Generalization

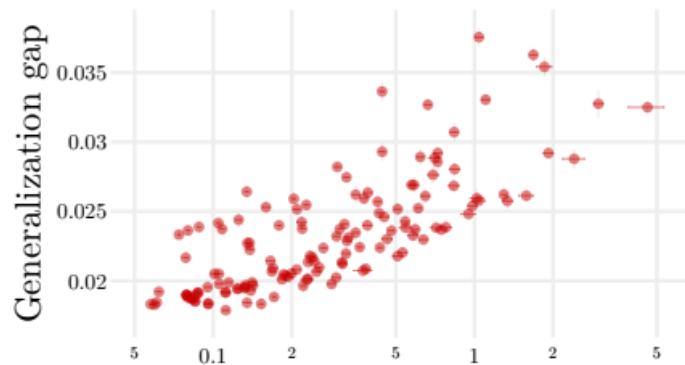
CIFAR10



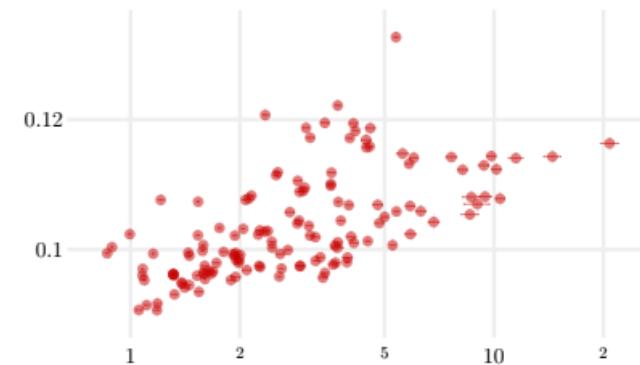
CIFAR100



MNIST



FASHION_MNIST



Jacobian norm

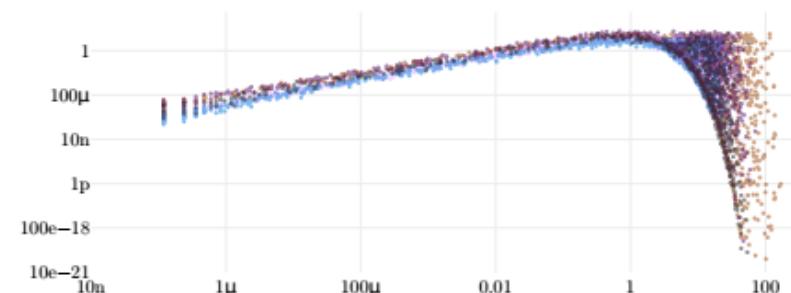
Sensitivity v.s. Generalization

- individual points

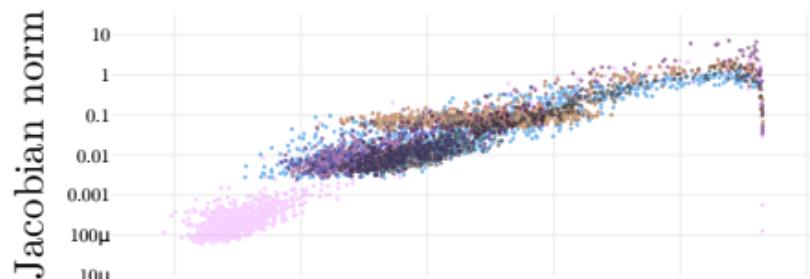
MNIST



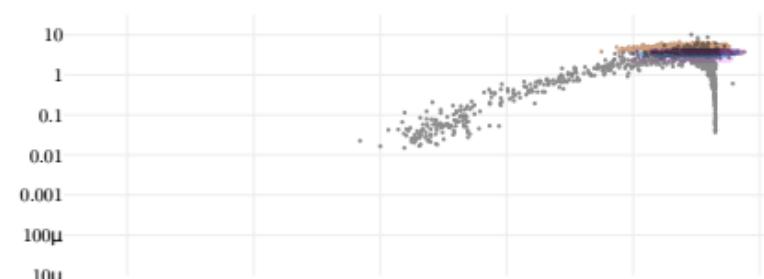
CIFAR10



Cross-entropy loss

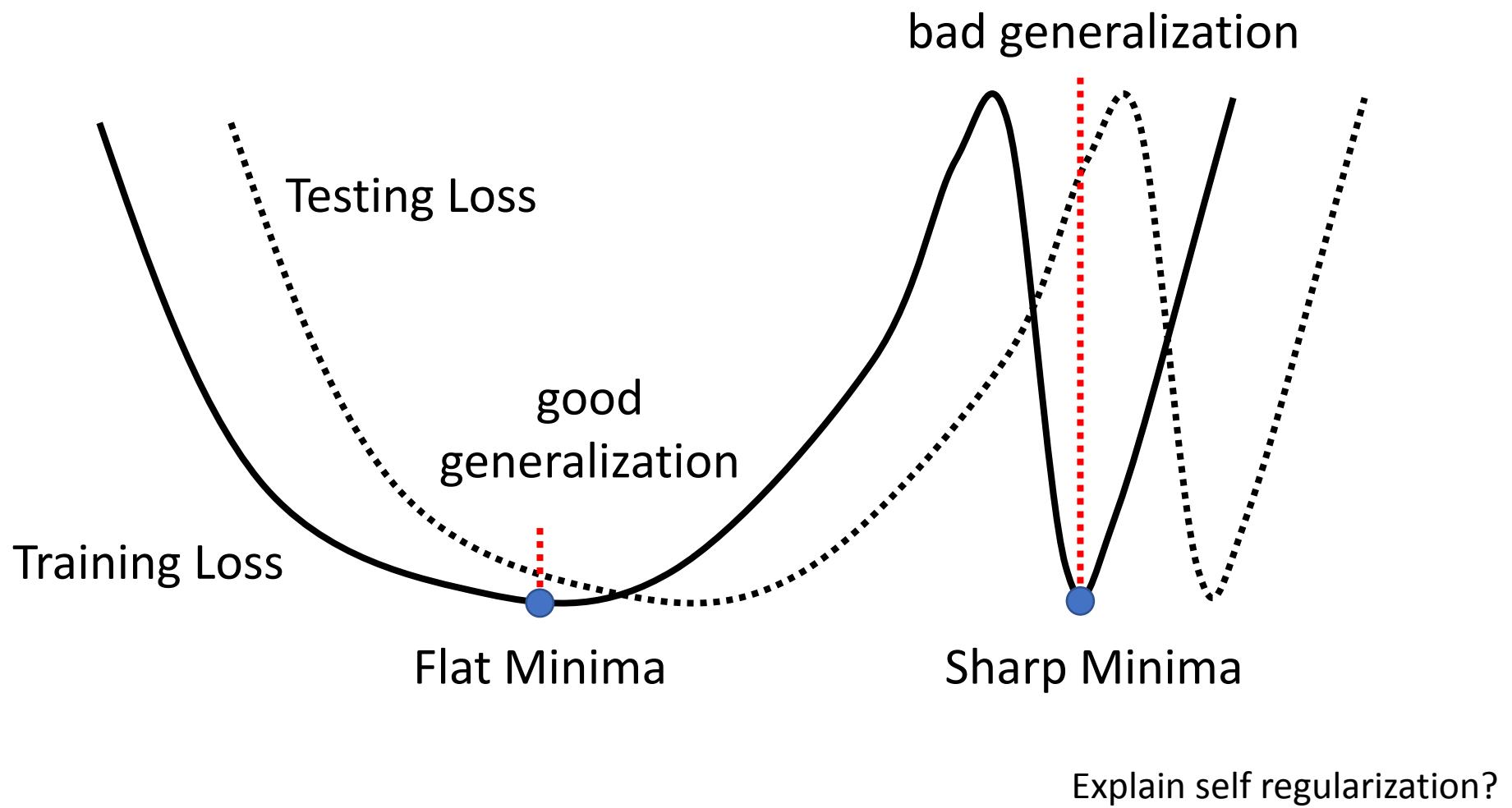


ℓ_2 -loss



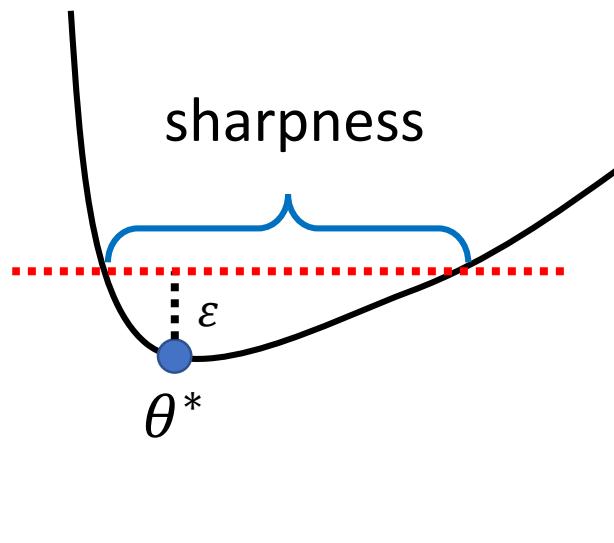
Sharpness

Sharp Minima v.s Flat Minima



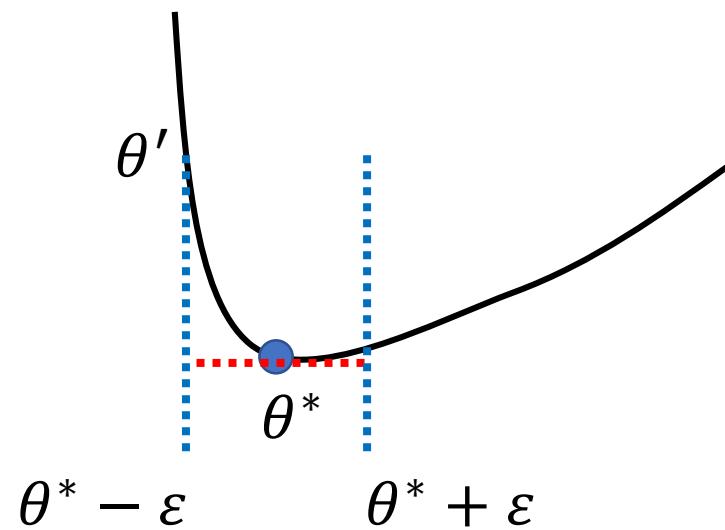
Definition of Sharpness

Definition 1



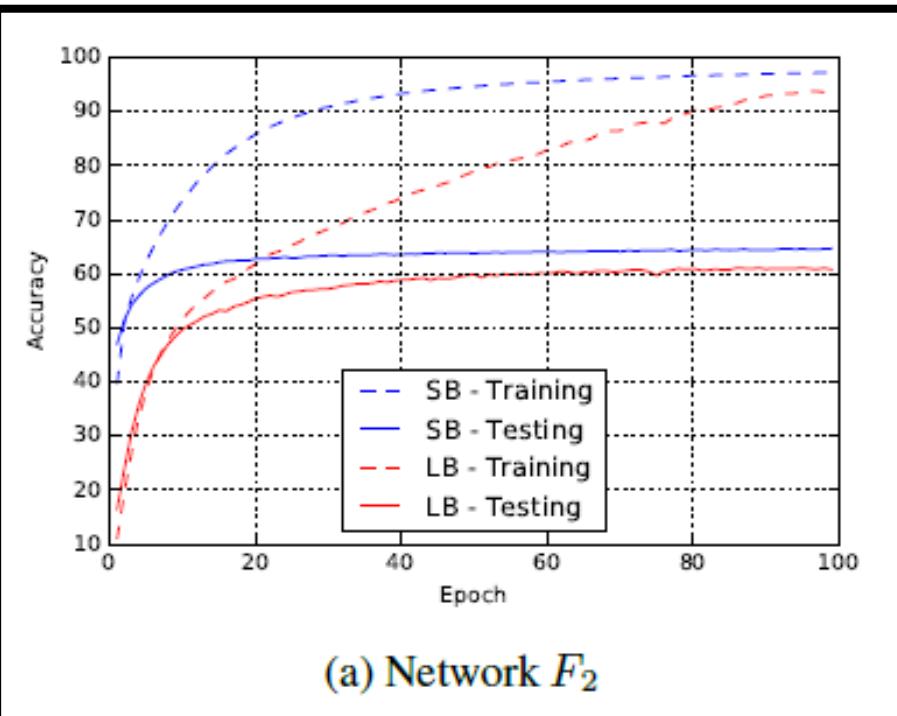
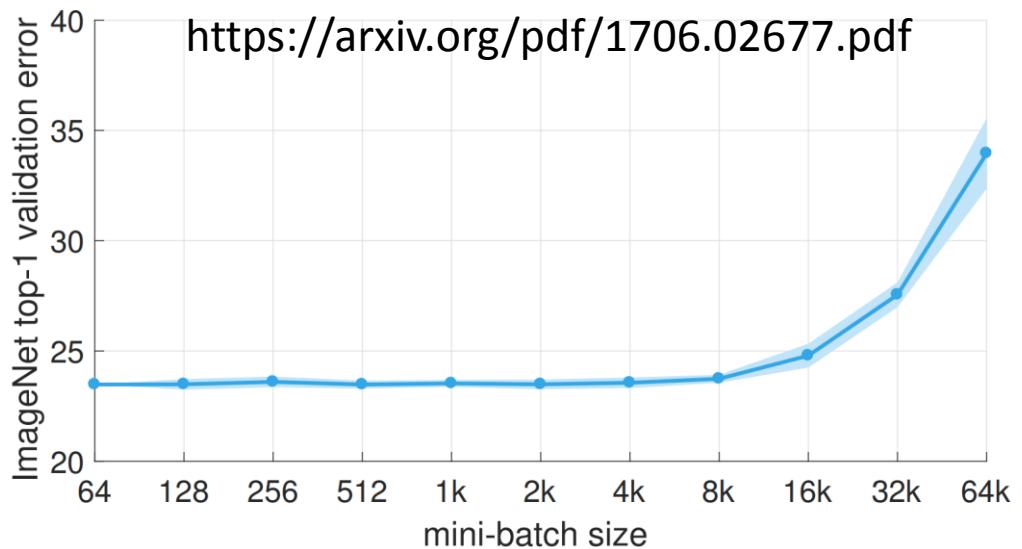
Definition 2

$$\text{Sharpness} = L(\theta') - L(\theta^*)$$

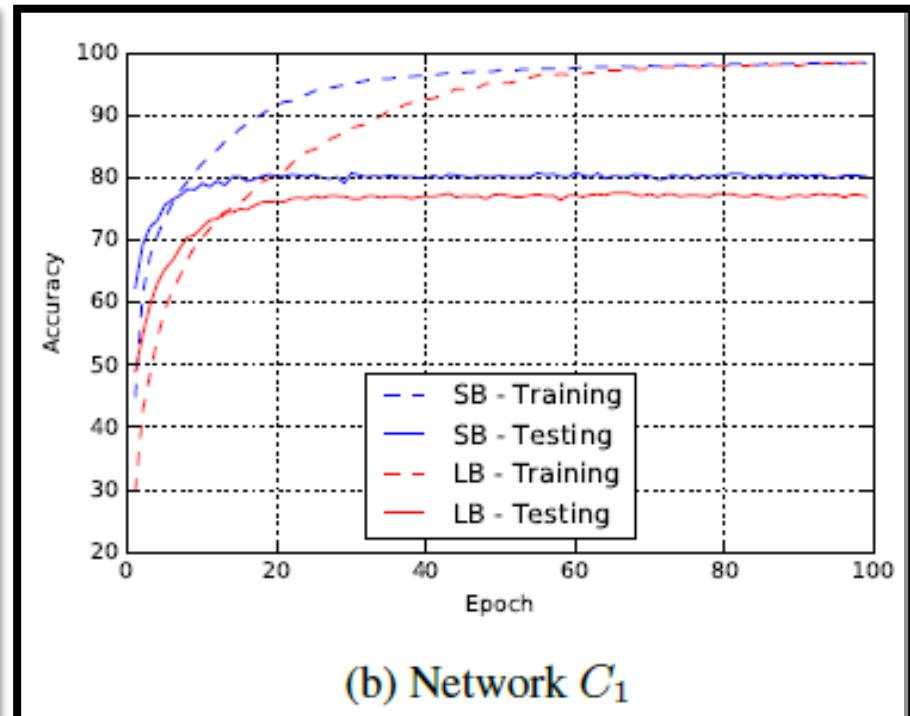


Batch Size

<https://arxiv.org/pdf/1706.02677.pdf>



(a) Network F_2



(b) Network C_1

Batch Size v.s. Sharpness

Name	Network Type	Data set
F_1	Fully Connected	MNIST (LeCun et al., 1998a)
F_2	Fully Connected	TIMIT (Garofolo et al., 1993)
C_1	(Shallow) Convolutional	CIFAR-10 (Krizhevsky & Hinton, 2009)
C_2	(Deep) Convolutional	CIFAR-10
C_3	(Shallow) Convolutional	CIFAR-100 (Krizhevsky & Hinton, 2009)
C_4	(Deep) Convolutional	CIFAR-100

Name	Training Accuracy		Testing Accuracy	
	SB	LB	SB	LB
F_1	$99.66\% \pm 0.05\%$	$99.92\% \pm 0.01\%$	$98.03\% \pm 0.07\%$	$97.81\% \pm 0.07\%$
F_2	$99.99\% \pm 0.03\%$	$98.35\% \pm 2.08\%$	$64.02\% \pm 0.2\%$	$59.45\% \pm 1.05\%$
C_1	$99.89\% \pm 0.02\%$	$99.66\% \pm 0.2\%$	$80.04\% \pm 0.12\%$	$77.26\% \pm 0.42\%$
C_2	$99.99\% \pm 0.04\%$	$99.99\% \pm 0.01\%$	$89.24\% \pm 0.12\%$	$87.26\% \pm 0.07\%$
C_3	$99.56\% \pm 0.44\%$	$99.88\% \pm 0.30\%$	$49.58\% \pm 0.39\%$	$46.45\% \pm 0.43\%$
C_4	$99.10\% \pm 1.23\%$	$99.57\% \pm 1.84\%$	$63.08\% \pm 0.5\%$	$57.81\% \pm 0.17\%$

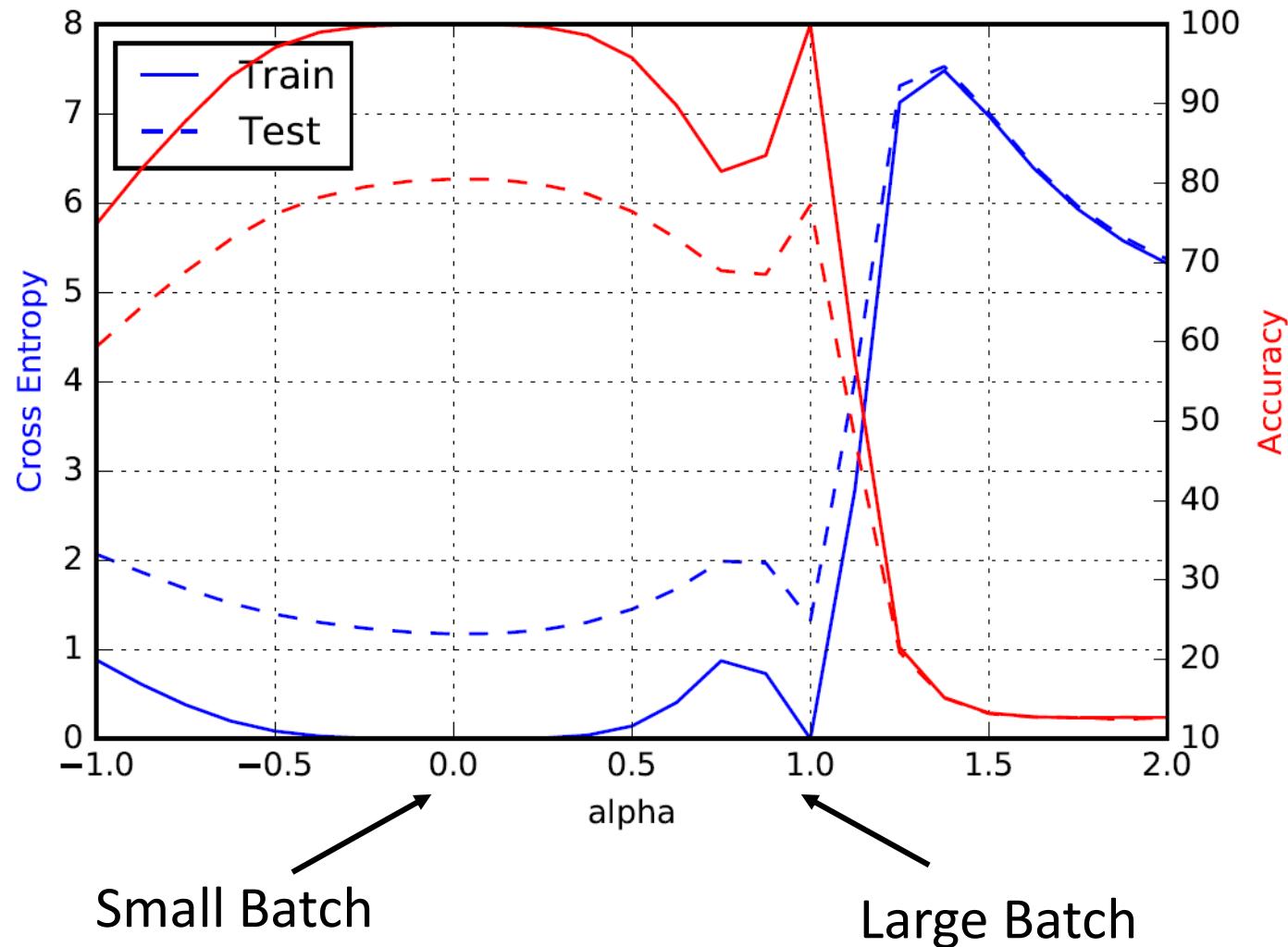
SB = 256

LB =

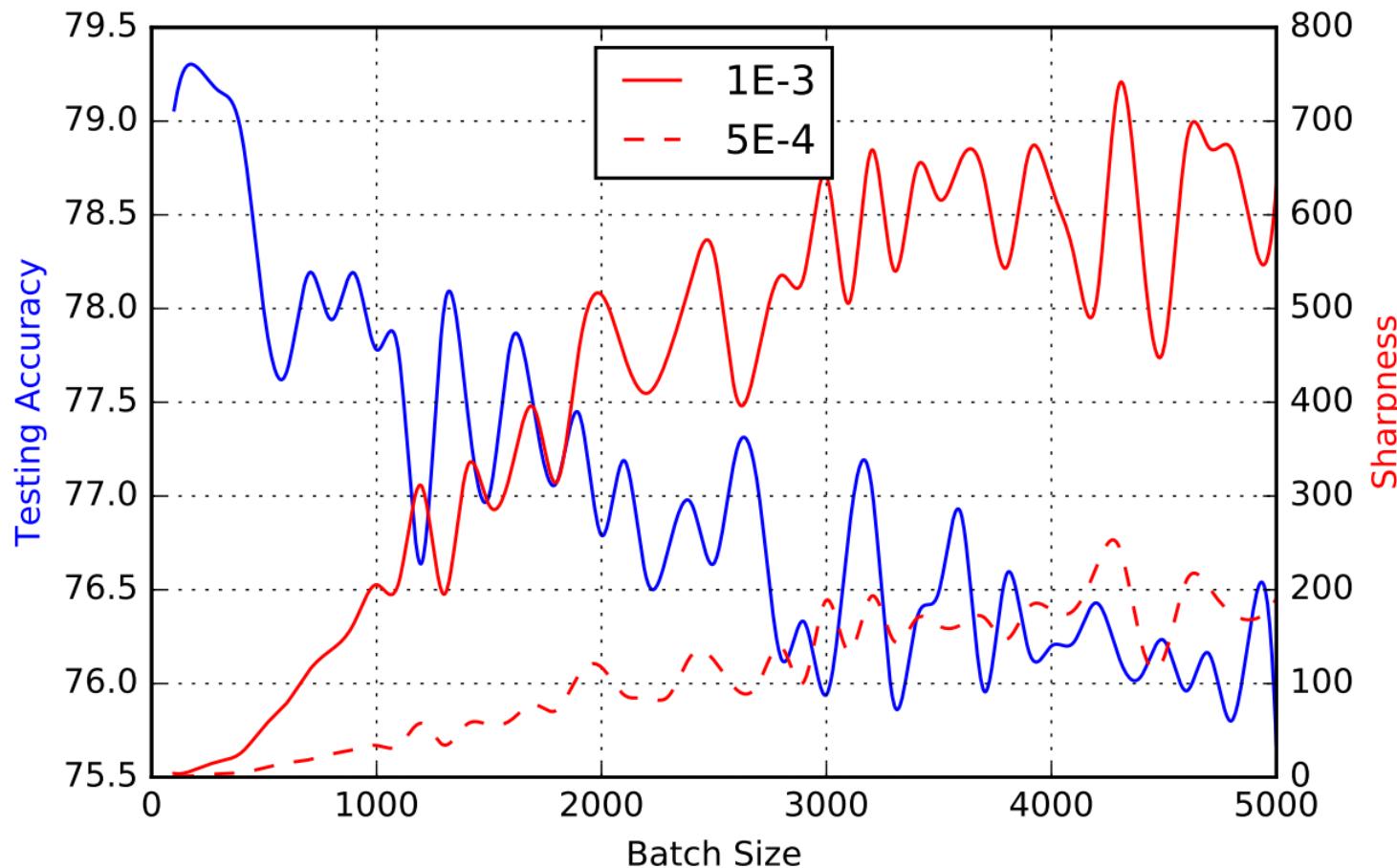
$0.1 \times$ data set

	$\epsilon = 10^{-3}$		$\epsilon = 5 \cdot 10^{-4}$	
	SB	LB	SB	LB
F_1	1.23 ± 0.83	205.14 ± 69.52	0.61 ± 0.27	42.90 ± 17.14
F_2	1.39 ± 0.02	310.64 ± 38.46	0.90 ± 0.05	93.15 ± 6.81
C_1	28.58 ± 3.13	707.23 ± 43.04	7.08 ± 0.88	227.31 ± 23.23
C_2	8.68 ± 1.32	925.32 ± 38.29	2.07 ± 0.86	175.31 ± 18.28
C_3	29.85 ± 5.98	258.75 ± 8.96	8.56 ± 0.99	105.11 ± 13.22
C_4	12.83 ± 3.84	421.84 ± 36.97	4.07 ± 0.87	109.35 ± 16.57

Batch Size v.s. Sharpness



Batch Size v.s. Sharpness



Concluding Remarks

Summary

- Good generalization are associated with sensitivity
- Good generalization are associated with flatness (?)
- Understanding the indicator for generalization helps us develop algorithm in the future

Reference

- Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, Simon Lacoste-Julien, “A Closer Look at Memorization in Deep Networks”, ICML, 2017
- Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, Ping Tak Peter Tang, “On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima”, ICLR, 2017
- Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, Riccardo Zecchina, “Entropy-SGD: Biassing Gradient Descent Into Wide Valleys”, ICLR, 2017
- Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, Nathan Srebro, Exploring Generalization in Deep Learning, NIPS, 2017
- Laurent Dinh, Razvan Pascanu, Samy Bengio, Yoshua Bengio, Sharp Minima Can Generalize For Deep Nets, PMLR, 2017
- Roman Novak, Yasaman Bahri, Daniel A. Abolafia, Jeffrey Pennington, Jascha Sohl-Dickstein, Sensitivity and Generalization in Neural Networks: an Empirical Study, ICLR, 2018

Computing Gradient

Hung-yi Lee

李宏毅

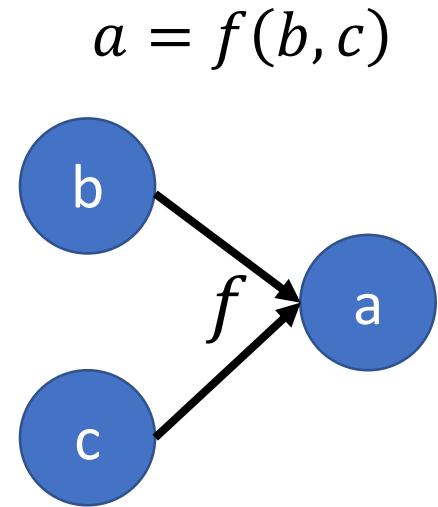
Introduction

- Backpropagation: an efficient way to compute the gradient
- Prerequisite
 - Backpropagation for feedforward net:
 - http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/DNN%20backprop.ecm.mp4/index.html
 - Simple version: <https://www.youtube.com/watch?v=ibJpTrp5mcE>
 - Backpropagation through time for RNN:
[http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/RNN%20training%20\(v6\).ecm.mp4/index.html](http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/RNN%20training%20(v6).ecm.mp4/index.html)
- Understanding backpropagation by computational graph
 - Tensorflow, Theano, CNTK, etc.

Computational Graph

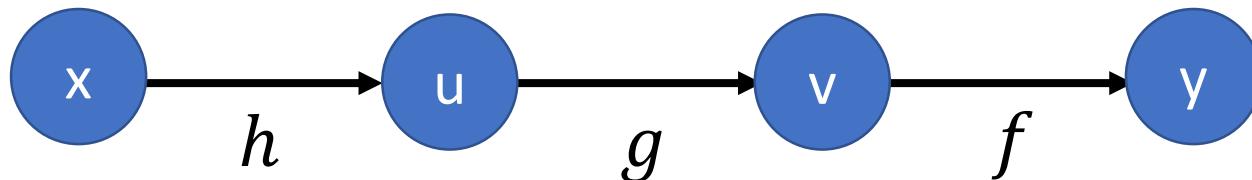
Computational Graph

- A “language” describing a function
 - **Node**: variable (scalar, vector, tensor))
 - **Edge**: operation (simple function)



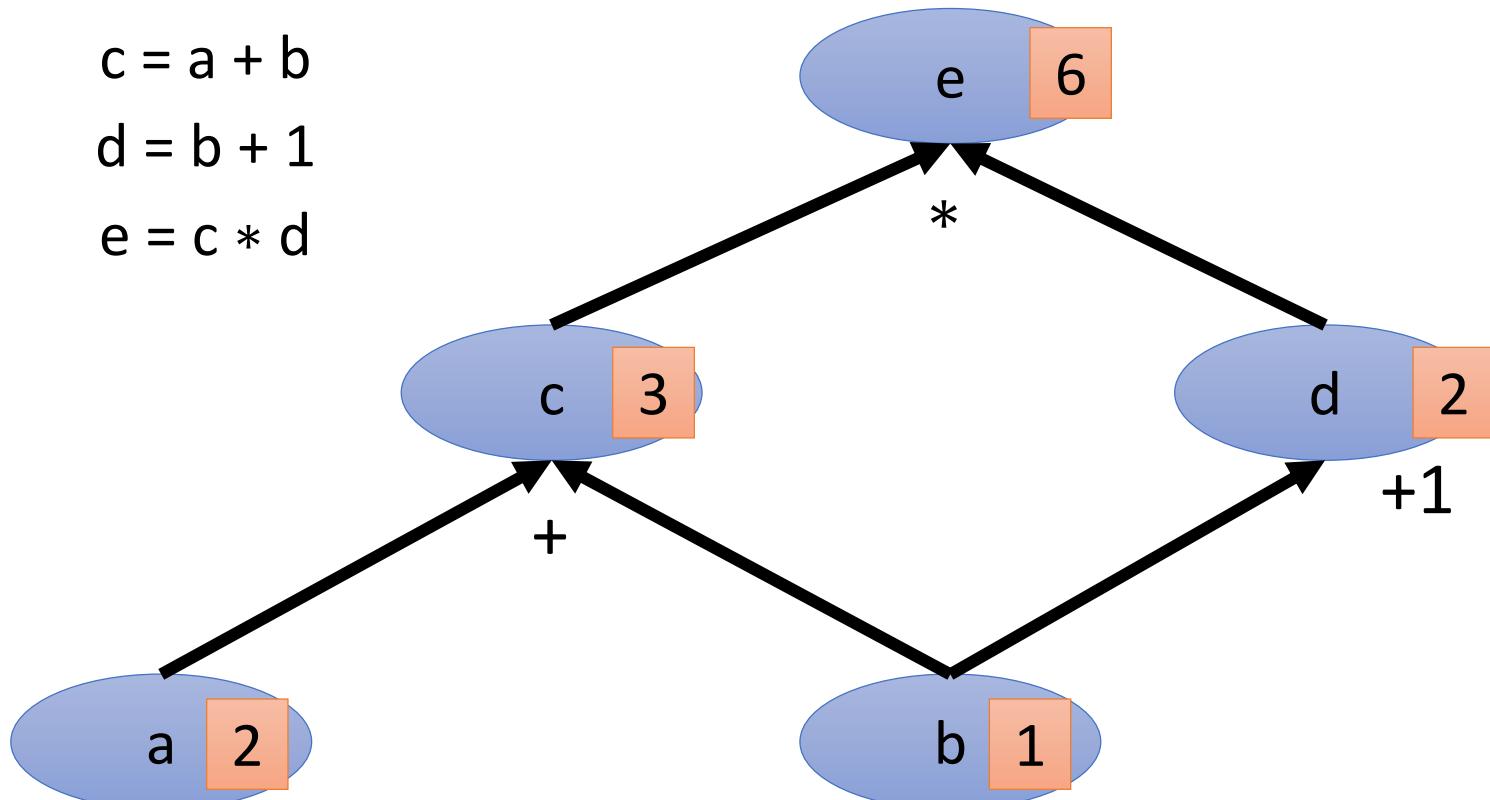
Example $y = f(g(h(x)))$

$$u = h(x) \quad v = g(u) \quad y = f(v)$$



Computational Graph

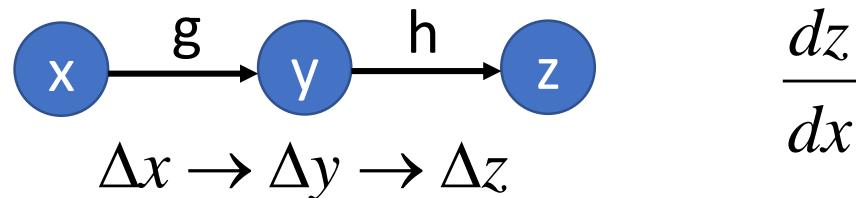
- Example: $e = (a+b) * (b+1)$



Review: Chain Rule

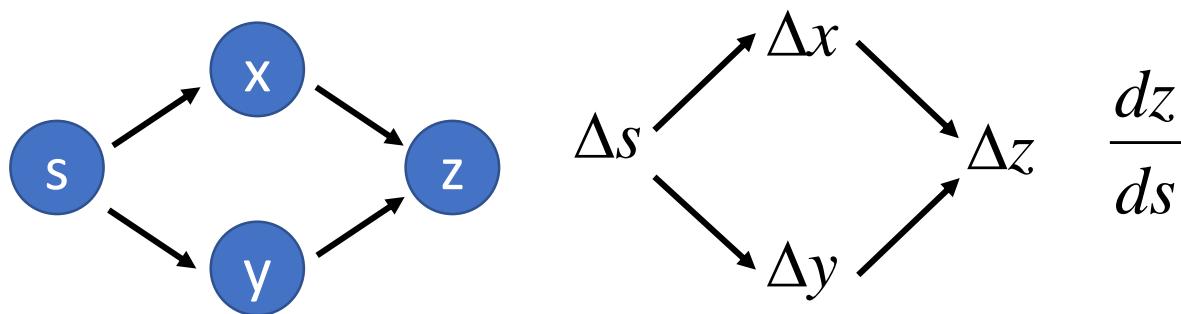
Case 1

$$z = f(x) \rightarrow y = g(x) \quad z = h(y)$$



Case 2

$$z = f(s) \rightarrow x = g(s) \quad y = h(s) \quad z = k(x, y)$$



Computational Graph

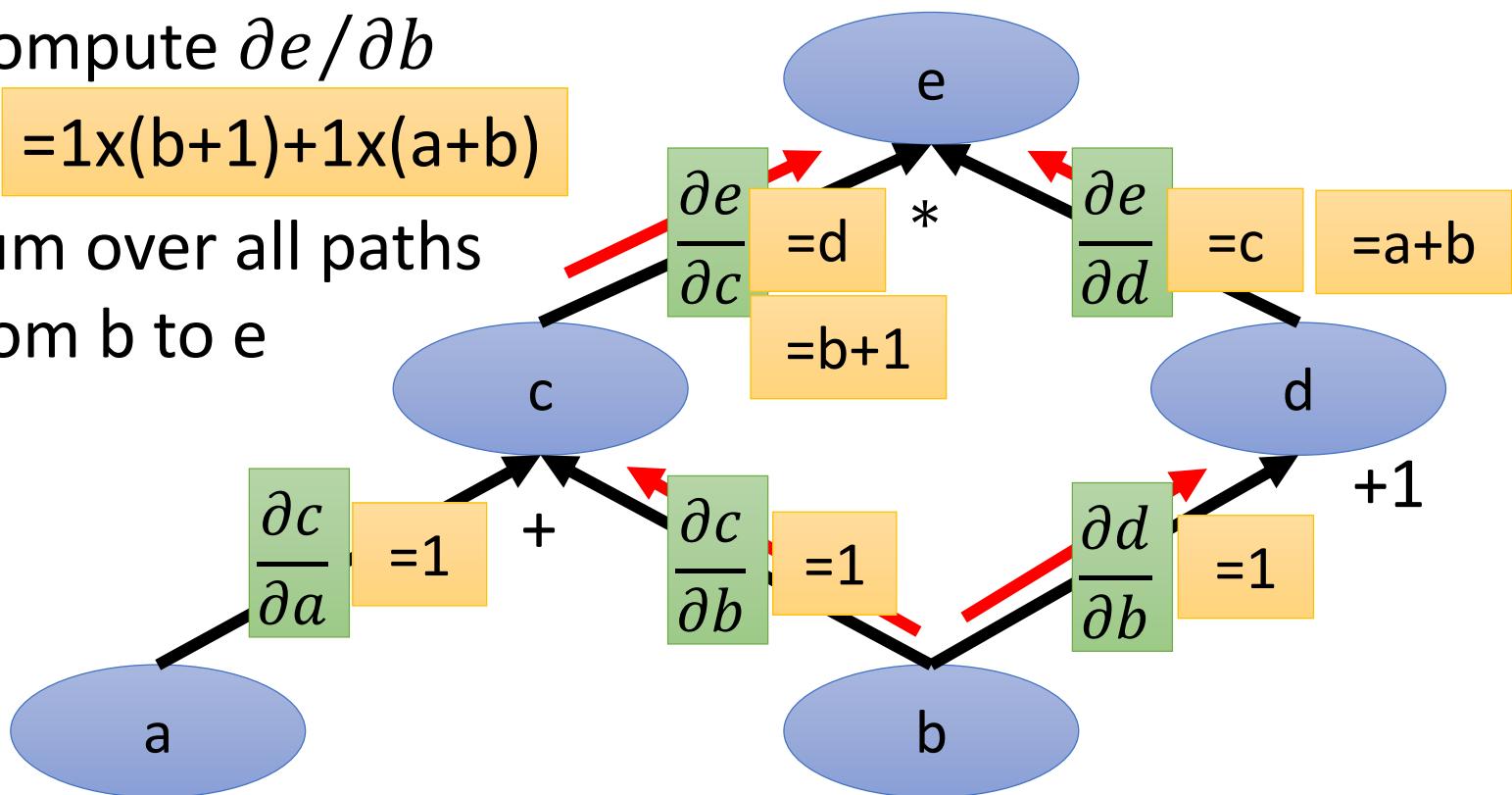
- Example: $e = (a+b) * (b+1)$

$$\frac{\partial e}{\partial b} = \frac{\partial e}{\partial c} \frac{\partial c}{\partial b} + \frac{\partial e}{\partial d} \frac{\partial d}{\partial b}$$

Compute $\frac{\partial e}{\partial b}$

$$=1 \times (b+1) + 1 \times (a+b)$$

Sum over all paths
from b to e

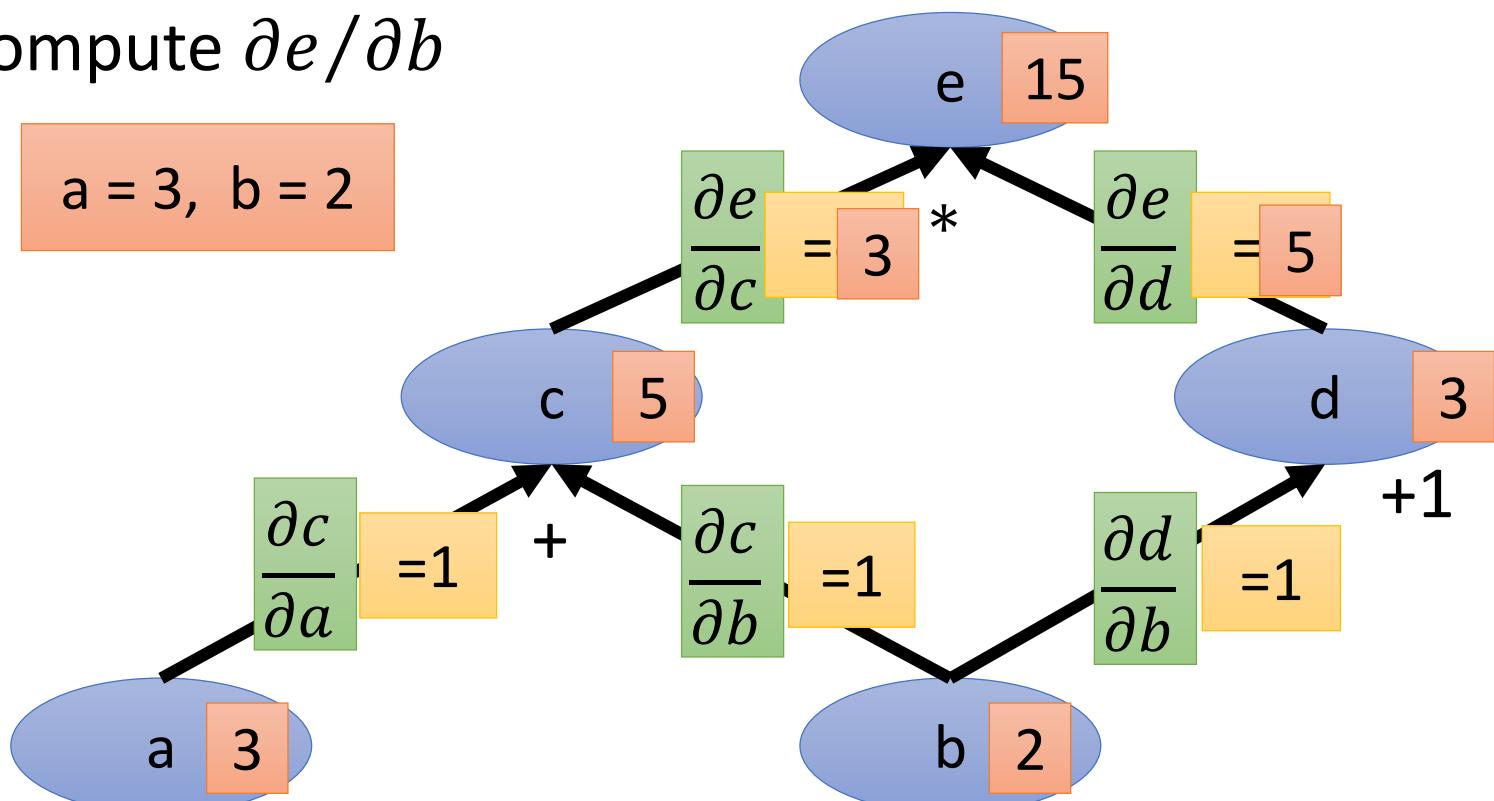


Computational Graph

- Example: $e = (a+b) * (b+1)$

$$\frac{\partial e}{\partial b} = \frac{\partial e}{\partial c} \frac{\partial c}{\partial b} + \frac{\partial e}{\partial d} \frac{\partial d}{\partial b}$$

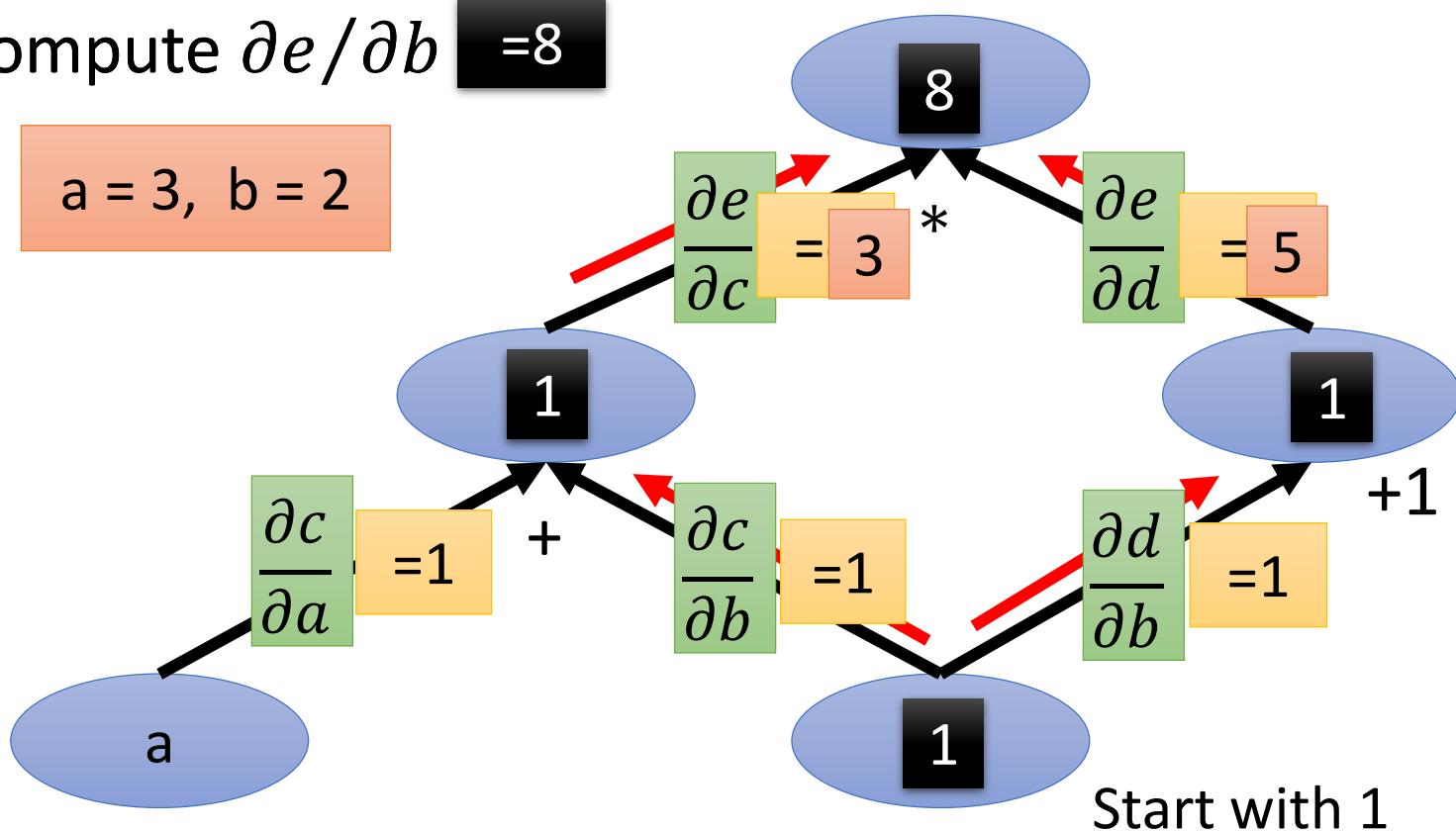
Compute $\frac{\partial e}{\partial b}$



Computational Graph

- Example: $e = (a+b) * (b+1)$

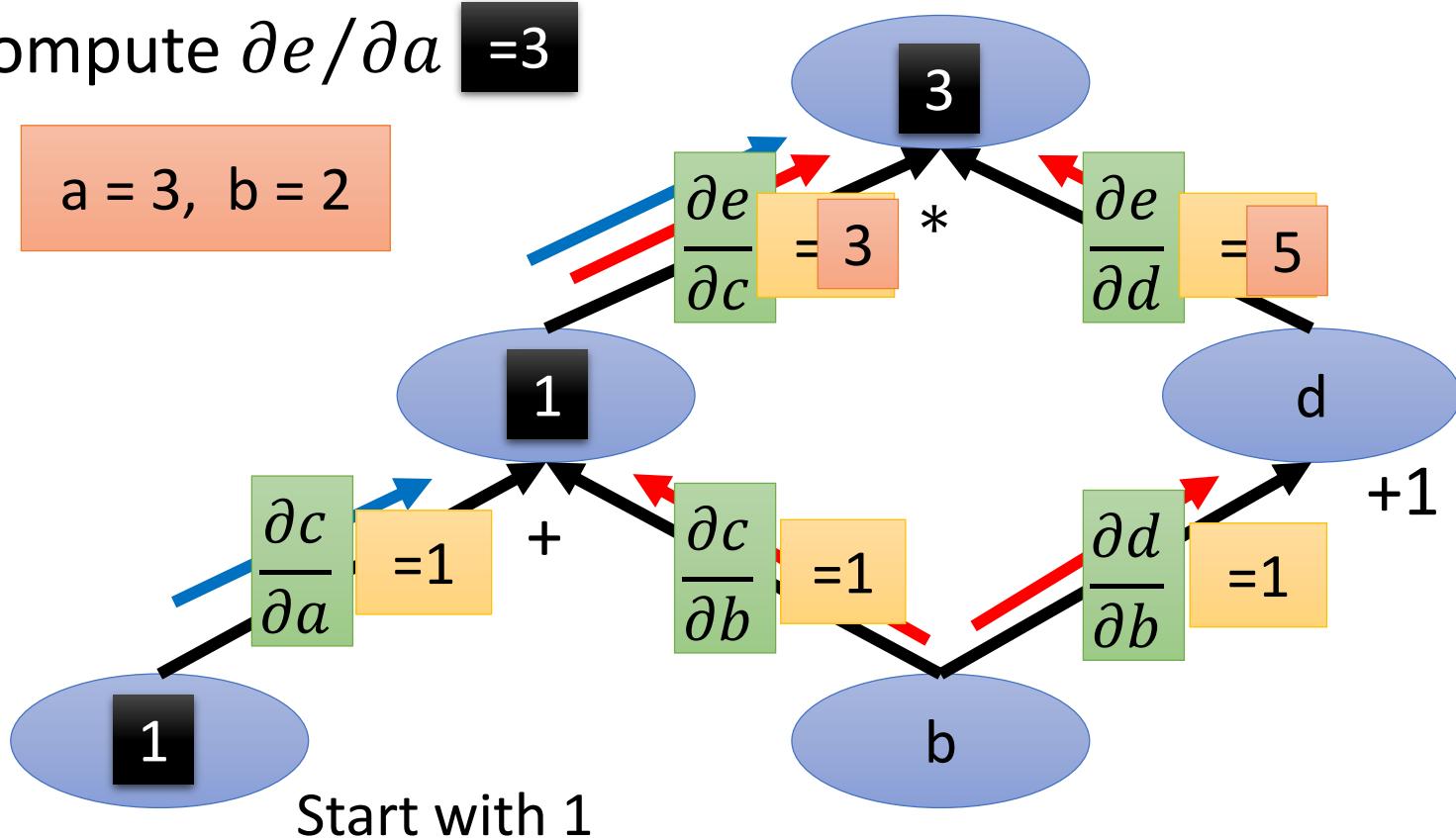
Compute $\partial e / \partial b$ = 8



Computational Graph

- Example: $e = (a+b) * (b+1)$

Compute $\partial e / \partial a = 3$

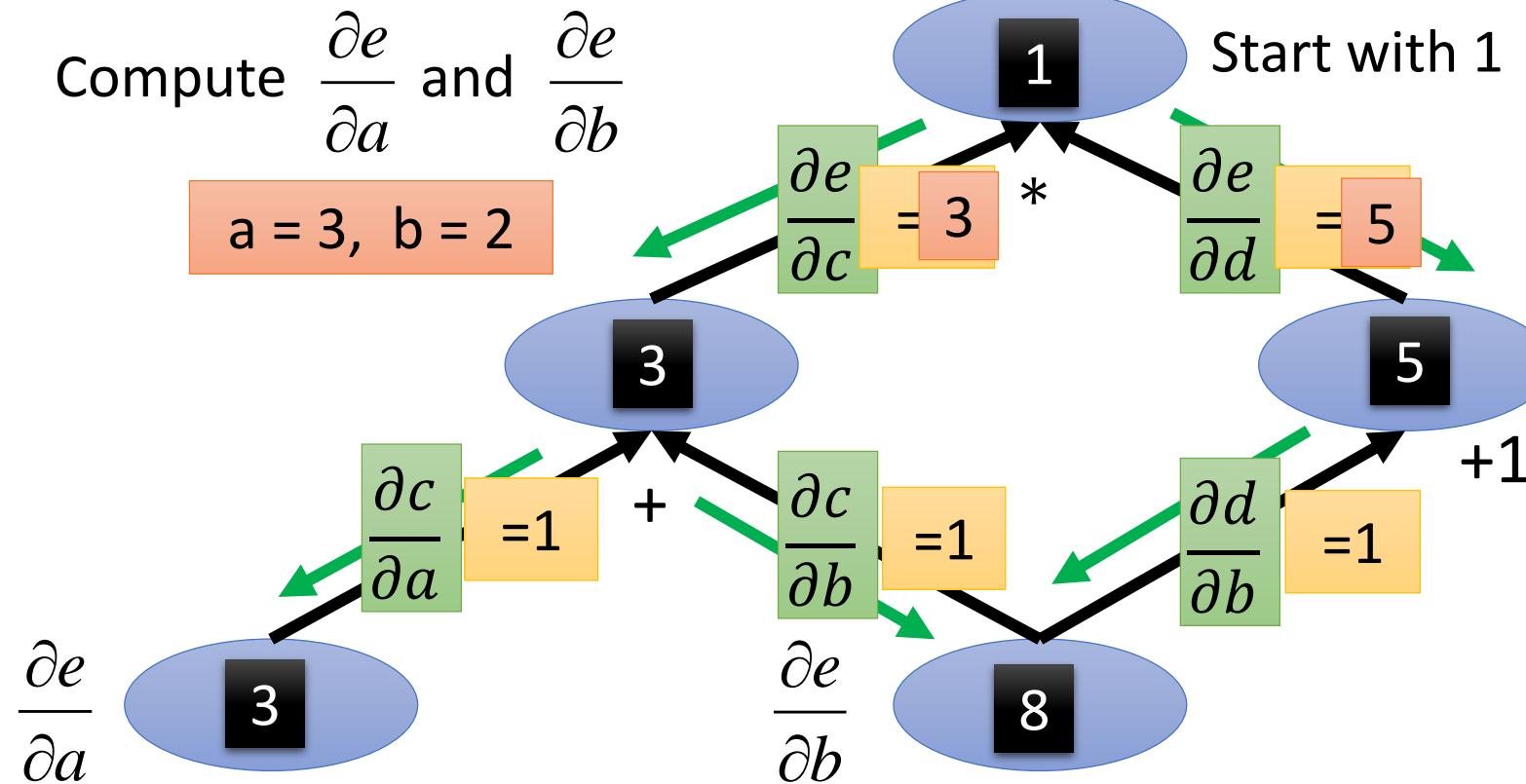


Computational Graph

Reverse mode

- Example: $e = (a+b) * (b+1)$

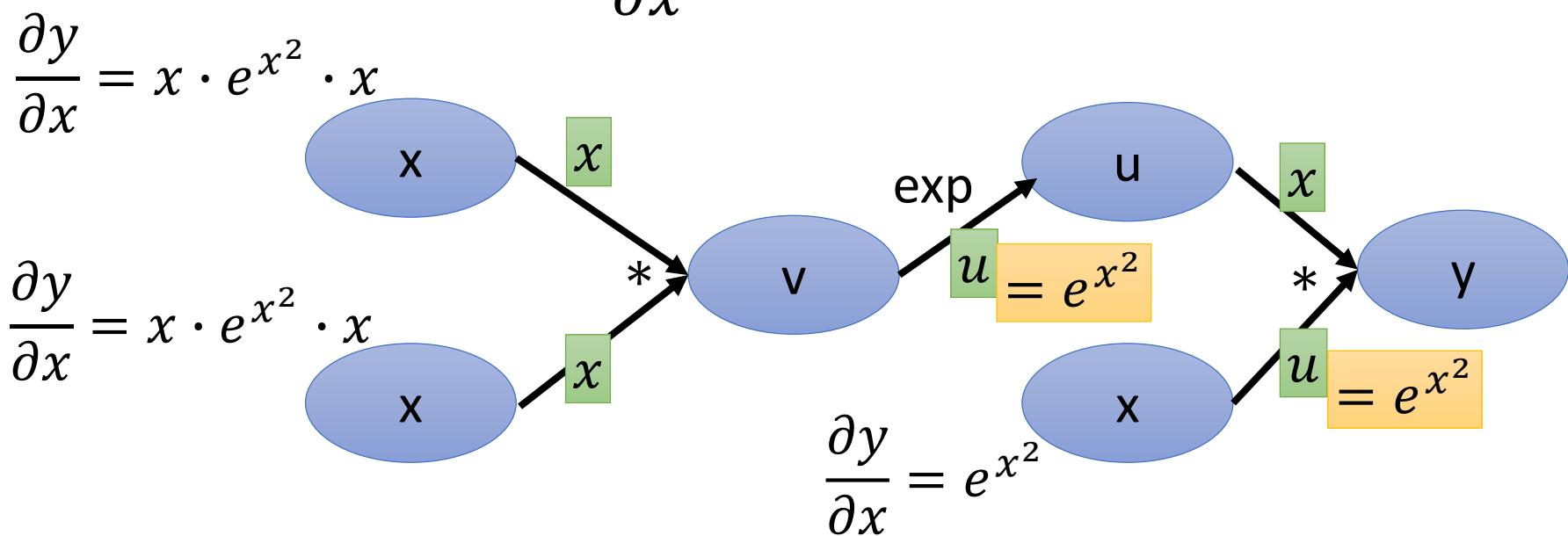
What is the benefit?



Computational Graph

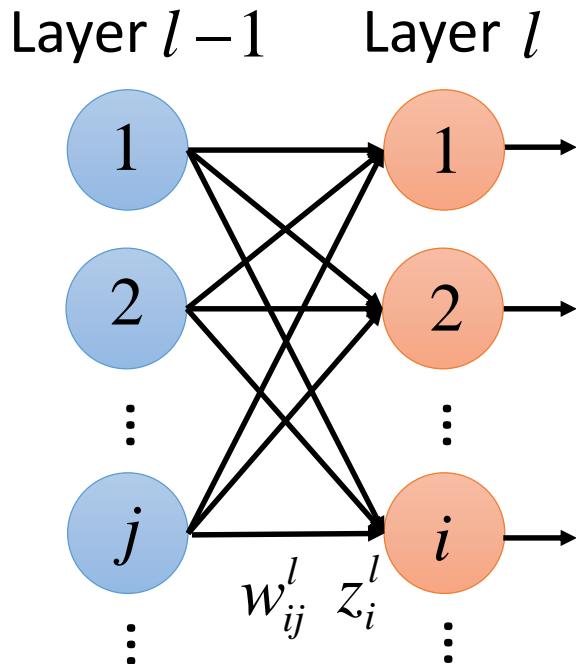
- **Parameter sharing:** the same parameters appearing in different nodes

$$y = xe^{x^2} \quad \frac{\partial y}{\partial x} = ? \quad e^{x^2} + x \cdot e^{x^2} \cdot 2x$$



Computational Graph for Feedforward Net

Review: Backpropagation



$$\begin{cases} a_j^{l-1} & l > 1 \\ x_j & l = 1 \end{cases}$$

Forward Pass

$$z^1 = W^1 x + b^1$$

$$a^1 = \sigma(z^1)$$

.....

$$z^{l-1} = W^{l-1} a^{l-2} + b^{l-1}$$

$$a^{l-1} = \sigma(z^{l-1})$$

$$\frac{\partial C}{\partial w_{ij}^l} = \boxed{\frac{\partial z_i^l}{\partial w_{ij}^l}} \boxed{\frac{\partial C}{\partial z_i^l}}$$

Error signal

δ_i^l

Backward Pass

$$\delta^L = \sigma'(z^L) \bullet \nabla_y C$$

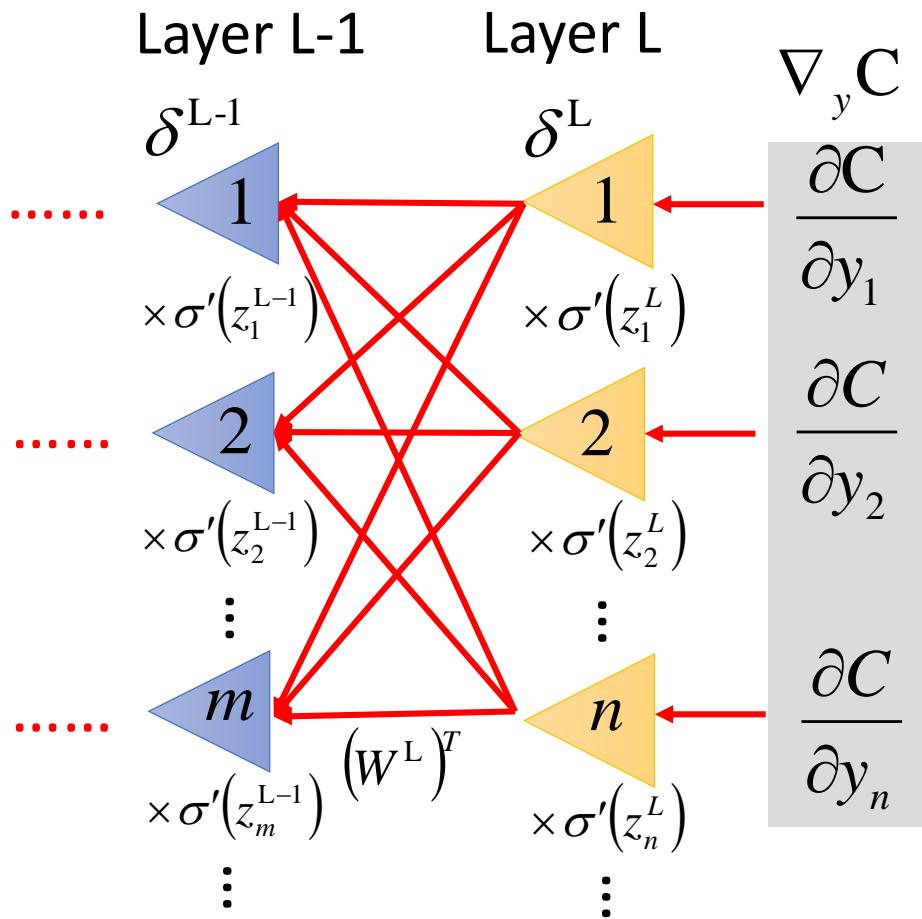
$$\delta^{L-1} = \sigma'(z^{L-1}) \bullet (W^L)^T \delta^L$$

.....

$$\delta^l = \sigma'(z^l) \bullet (W^{l+1})^T \delta^{l+1}$$

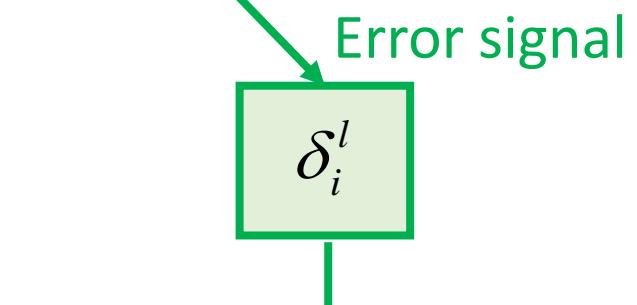
.....

Review: Backpropagation



(we do not use softmax here)

$$\frac{\partial C}{\partial w_{ij}^l} = \boxed{\frac{\partial z_i^l}{\partial w_{ij}^l}} \boxed{\frac{\partial C}{\partial z_i^l}}$$



Backward Pass

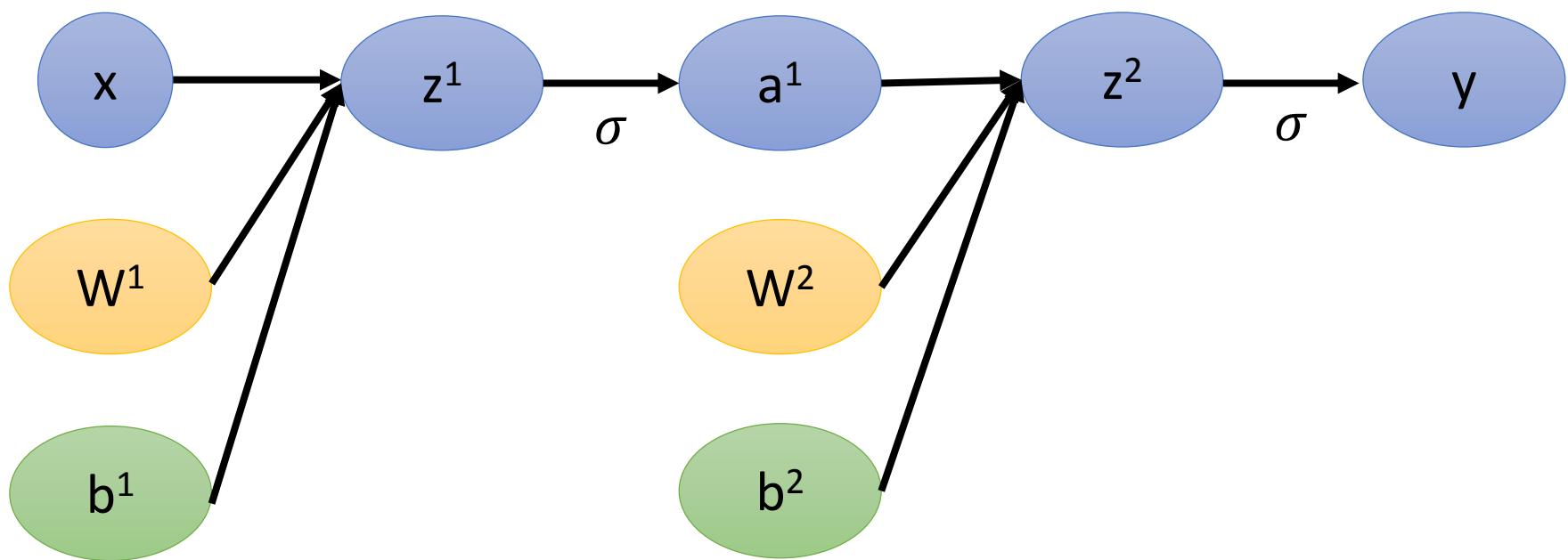
$$\begin{aligned}\delta^L &= \sigma'(z^L) \bullet \nabla_y C \\ \delta^{L-1} &= \sigma'(z^{L-1}) \bullet (W^L)^T \delta^L \\ &\dots \\ \delta^l &= \sigma'(z^l) \bullet (W^{l+1})^T \delta^{l+1} \\ &\dots\end{aligned}$$

Feedforward Network

$$y = \sigma(W^L \dots \sigma(W^2 \sigma(W^1 x + b^1) + b^2) \dots + b^L)$$

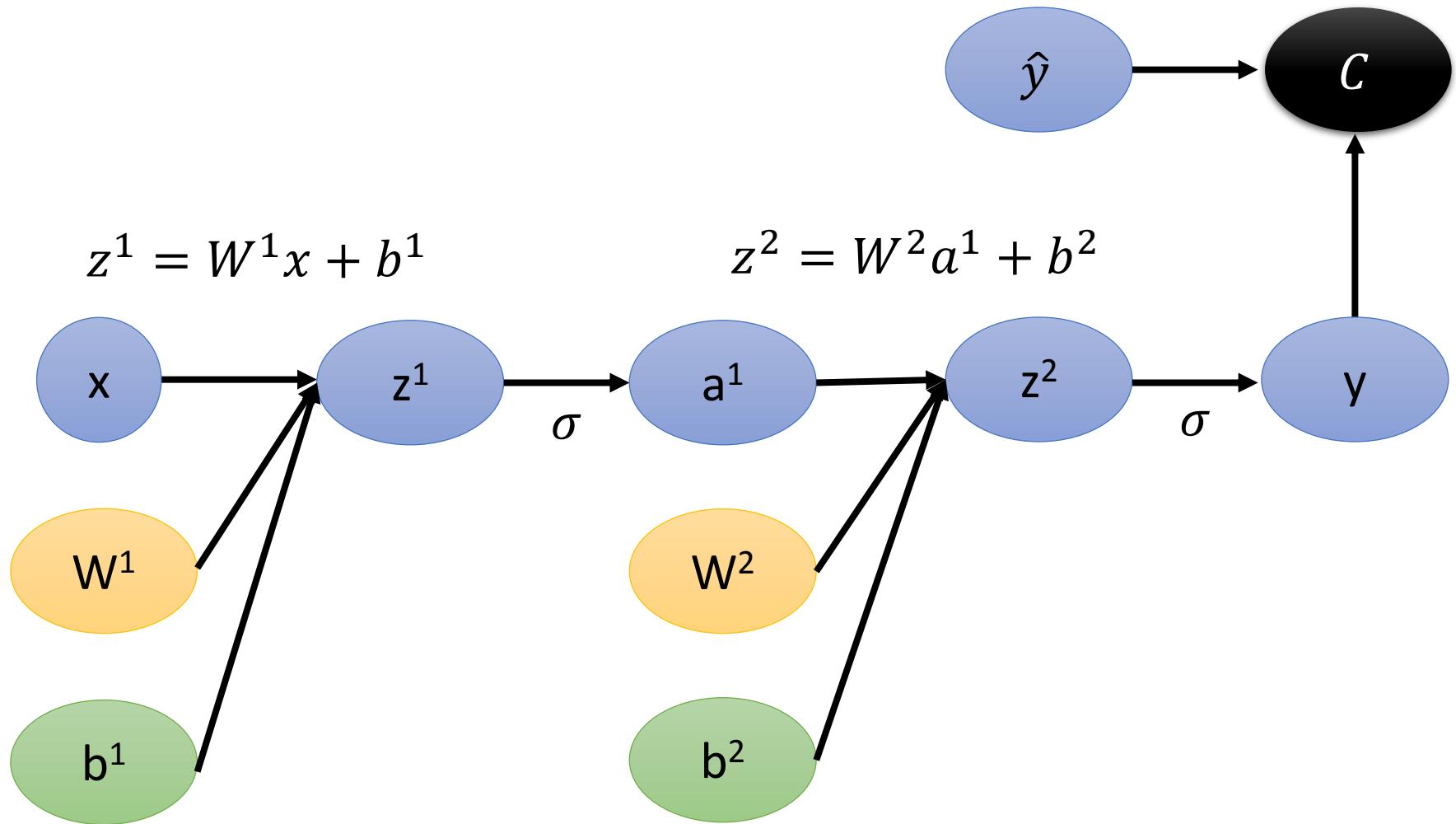
$$z^1 = W^1 x + b^1$$

$$z^2 = W^2 a^1 + b^2$$



Loss Function of Feedforward Network

$$c = L(y, \hat{y})$$



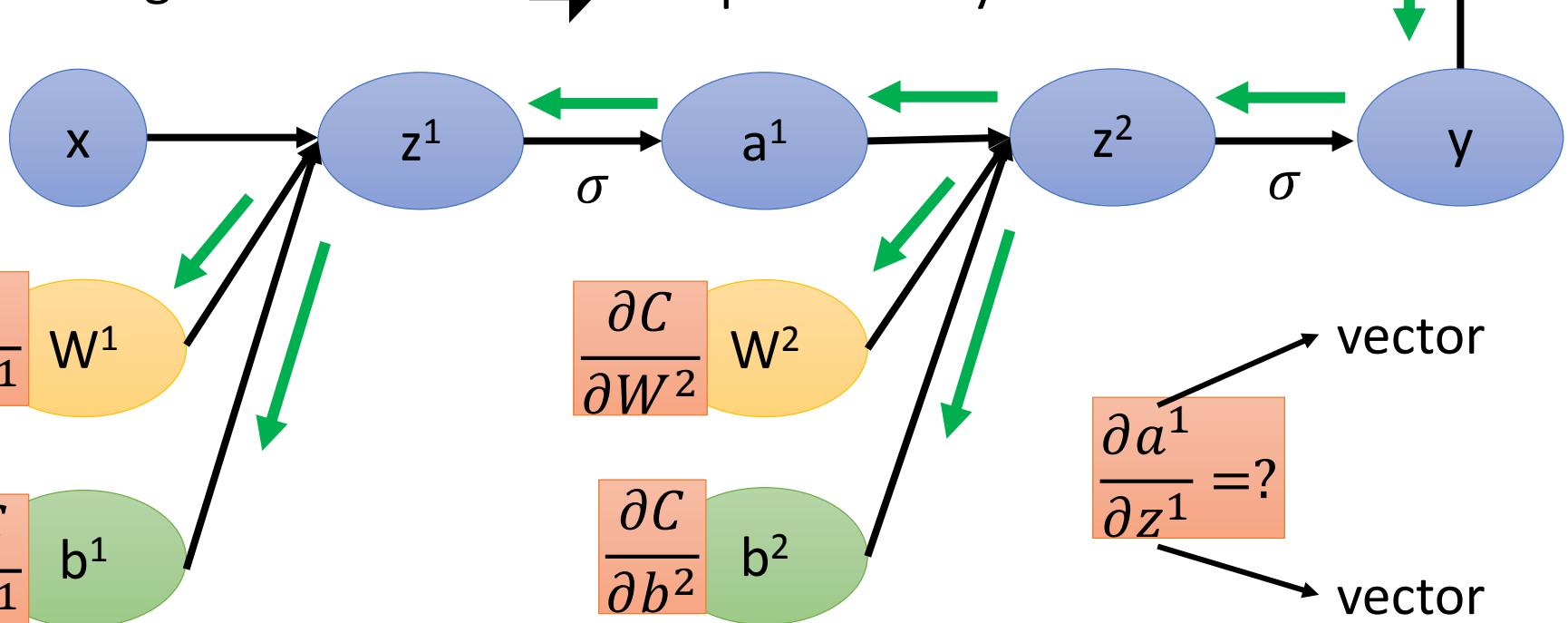
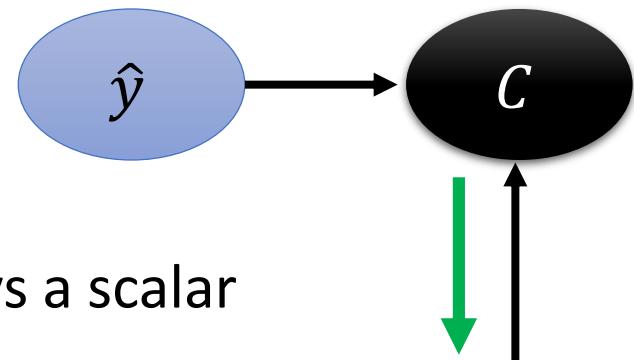
Gradient of Cost Function

To compute the gradient ...

Computing the partial derivative
on the edge

Using reverse mode  Output is always a scalar

$$C = L(y, \hat{y})$$



Jacobian Matrix

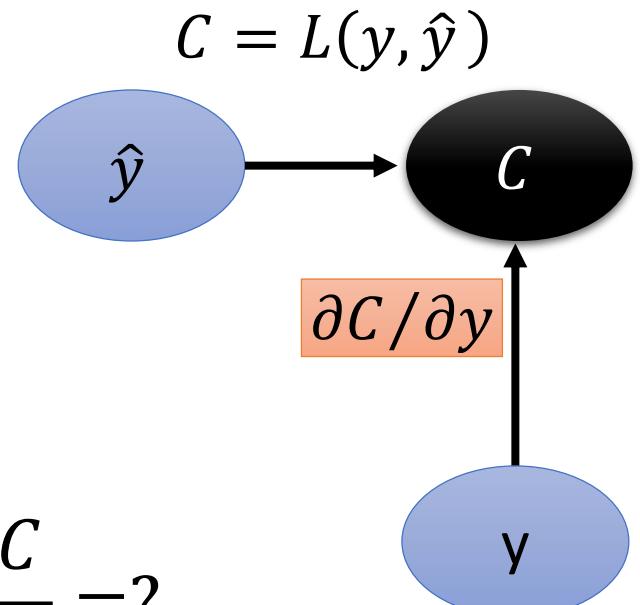
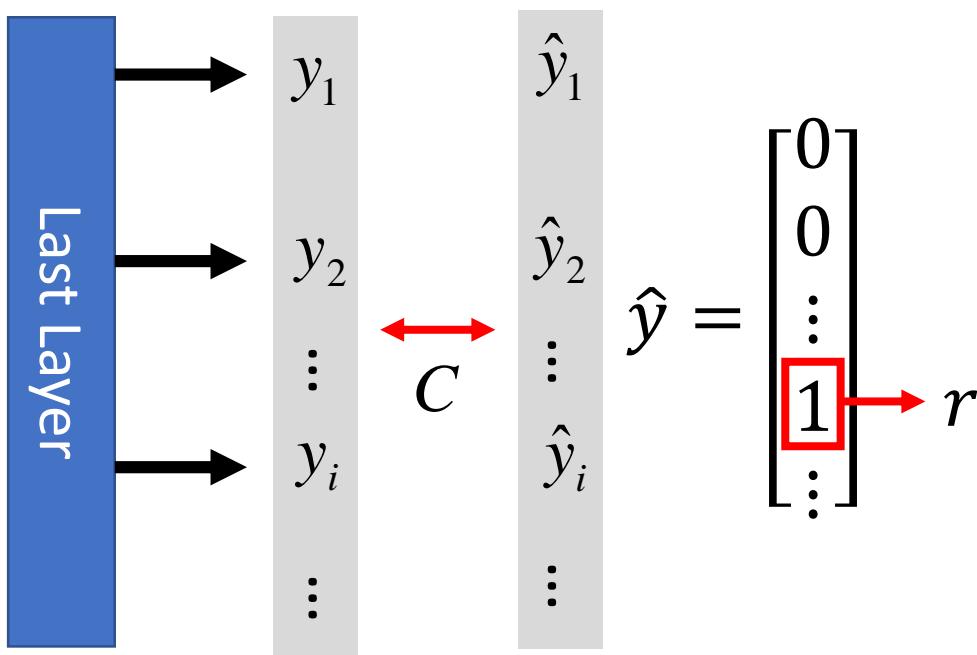
$$y = f(x) \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$\frac{\partial y}{\partial x} = \left. \begin{array}{c} \text{size of } y \\ \text{size of } x \end{array} \right\}$$

Example

$$\begin{bmatrix} x_1 + x_2 x_3 \\ 2x_3 \end{bmatrix} = f \left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \right) \quad \frac{\partial y}{\partial x} = []$$

Gradient of Cost Function



$$\frac{\partial C}{\partial y_i} = ?$$

Cross Entropy: $C = -\log y_r$

$$\frac{\partial C}{\partial y} = [\quad]$$

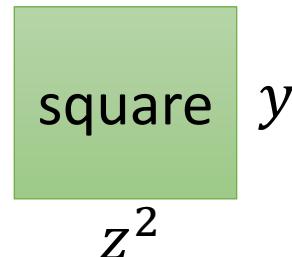
$$i = r:$$

$$\frac{\partial C}{\partial y_r} = -1/y_r$$

$$i \neq r: \quad \frac{\partial C}{\partial y_i} = 0$$

Gradient of Cost Function

$\frac{\partial y}{\partial z^2}$ is a Jacobian matrix



i-th row, j-th column: $\partial y_i / \partial z_j^2$

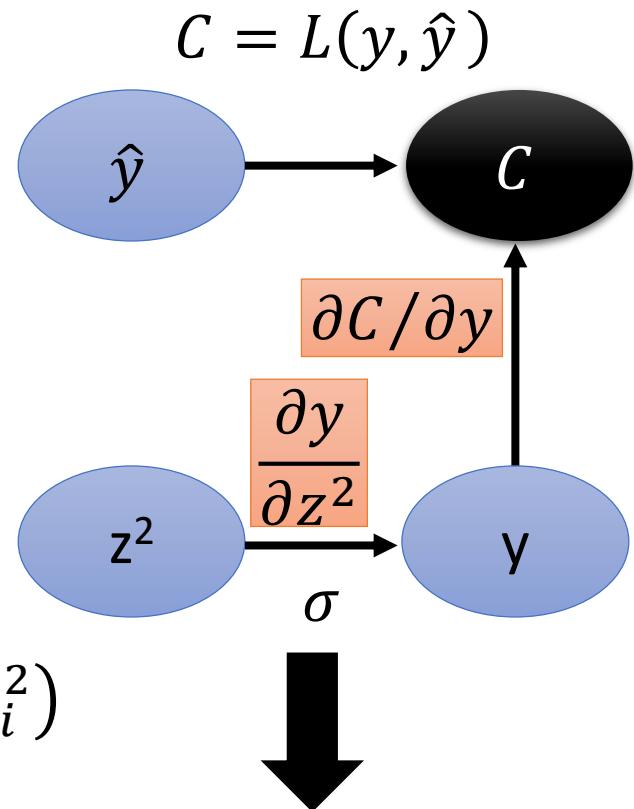
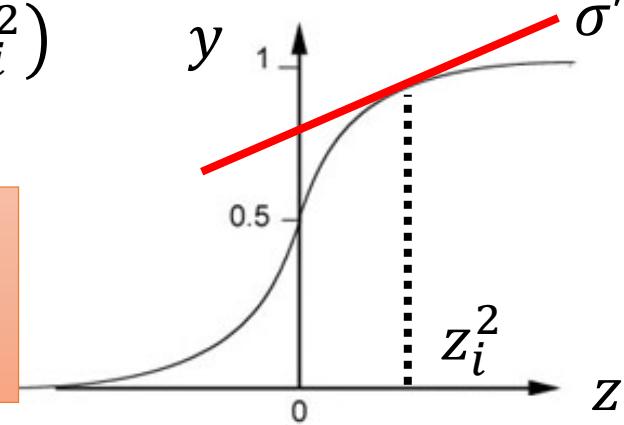
$$i \neq j: \quad \partial y_i / \partial z_j^2 = 0$$

$$i = j: \quad \partial y_i / \partial z_i^2 = \sigma'(z_i^2)$$

$$y_i = \sigma(z_i^2)$$

$$\sigma'(z_i^2)$$

How about softmax? ☺



Diagonal
Matrix

$\frac{\partial z^2}{\partial a^1}$ is a Jacobian matrix

i-th row, j-th column:

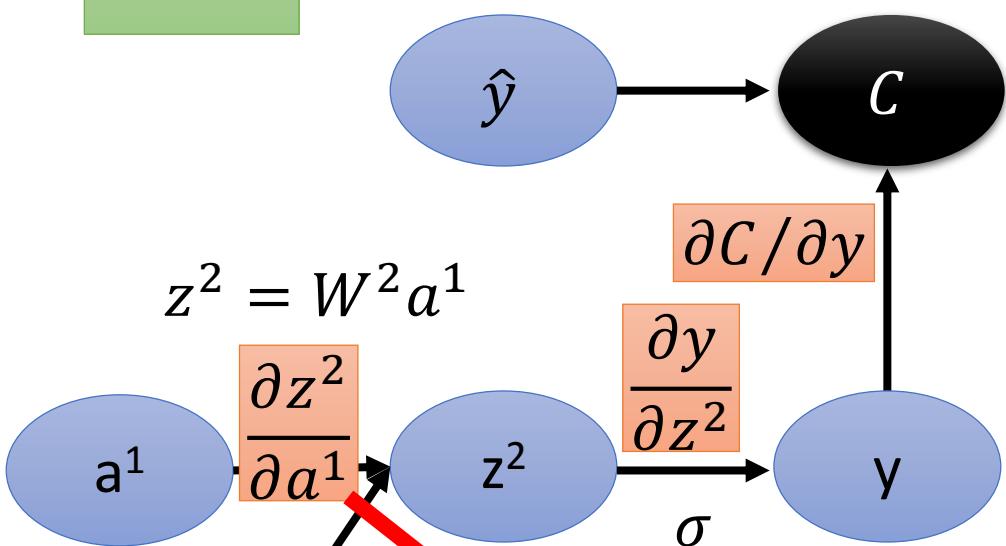
$$\frac{\partial z_i^2}{\partial a_j^1} =$$

$$z_i^2 = w_{i1}^2 a_1^1 + w_{i2}^2 a_2^1 + \dots + w_{in}^2 a_n^1$$

$$\begin{bmatrix} z_1^l \\ z_2^l \\ \vdots \\ z_i^l \\ \vdots \end{bmatrix} = \begin{bmatrix} w_{11}^l & w_{12}^l & \cdots & & \\ w_{21}^l & w_{22}^l & & \ddots & \\ \vdots & & & & \\ & & & & \end{bmatrix} \begin{bmatrix} a_1^{l-1} \\ a_2^{l-1} \\ \vdots \\ a_i^{l-1} \\ \vdots \end{bmatrix} + \begin{bmatrix} b_1^l \\ b_2^l \\ \vdots \\ b_i^l \\ \vdots \end{bmatrix}$$

$$z^2 \quad a^1$$

$$C = L(y, \hat{y})$$



$$z^2 = W^2 a^1$$

$$\frac{\partial z^2}{\partial a^1}$$

$$\frac{\partial y}{\partial z^2}$$

$$\sigma$$

$$\frac{\partial z^2}{\partial W^2}$$

$$W^2$$

$$\frac{\partial z^2}{\partial W^2} = m$$

mxn

$(j-1)xn+k$

i

$\frac{\partial z_i^2}{\partial W_{jk}^2}$

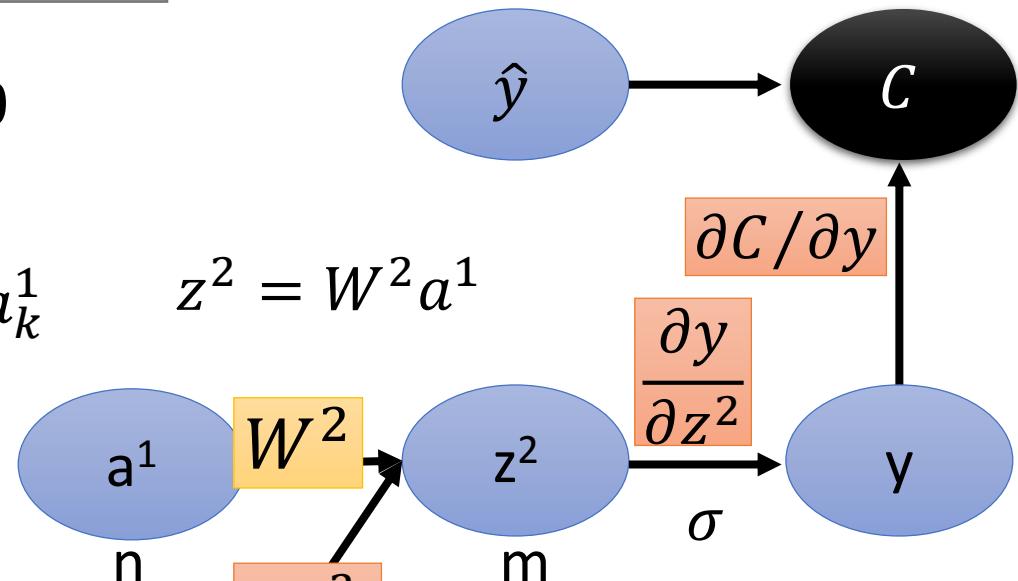
$$\frac{\partial z_i^2}{\partial W_{jk}^2} = ? \quad i \neq j: \frac{\partial z_i^2}{\partial W_{jk}^2} = 0$$

$$i = j: \frac{\partial z_i^2}{\partial W_{ik}^2} = a_k^1$$

$$z_i^2 = w_{i1}^2 a_1^1 + w_{i2}^2 a_2^1 + \dots + w_{in}^2 a_n^1$$

$$\begin{bmatrix} z_1^l \\ z_2^l \\ \vdots \\ z_i^l \\ \vdots \end{bmatrix} = \begin{bmatrix} w_{11}^l & w_{12}^l & \cdots & \ddots & \end{bmatrix} \begin{bmatrix} a_1^{l-1} \\ a_2^{l-1} \\ \vdots \\ a_i^{l-1} \\ \vdots \end{bmatrix} + \begin{bmatrix} b_1^l \\ b_2^l \\ \vdots \\ b_i^l \\ \vdots \end{bmatrix}$$

W^2
mxn



$$C = L(y, \hat{y})$$

\hat{y}

C

z^2

m

σ

Considering W^2
as a mxn **vector**

(considering $\partial z^2 / \partial W^2$ as a tensor makes thing easier)

$$\frac{\partial z^2}{\partial W^2} = m$$

mxn

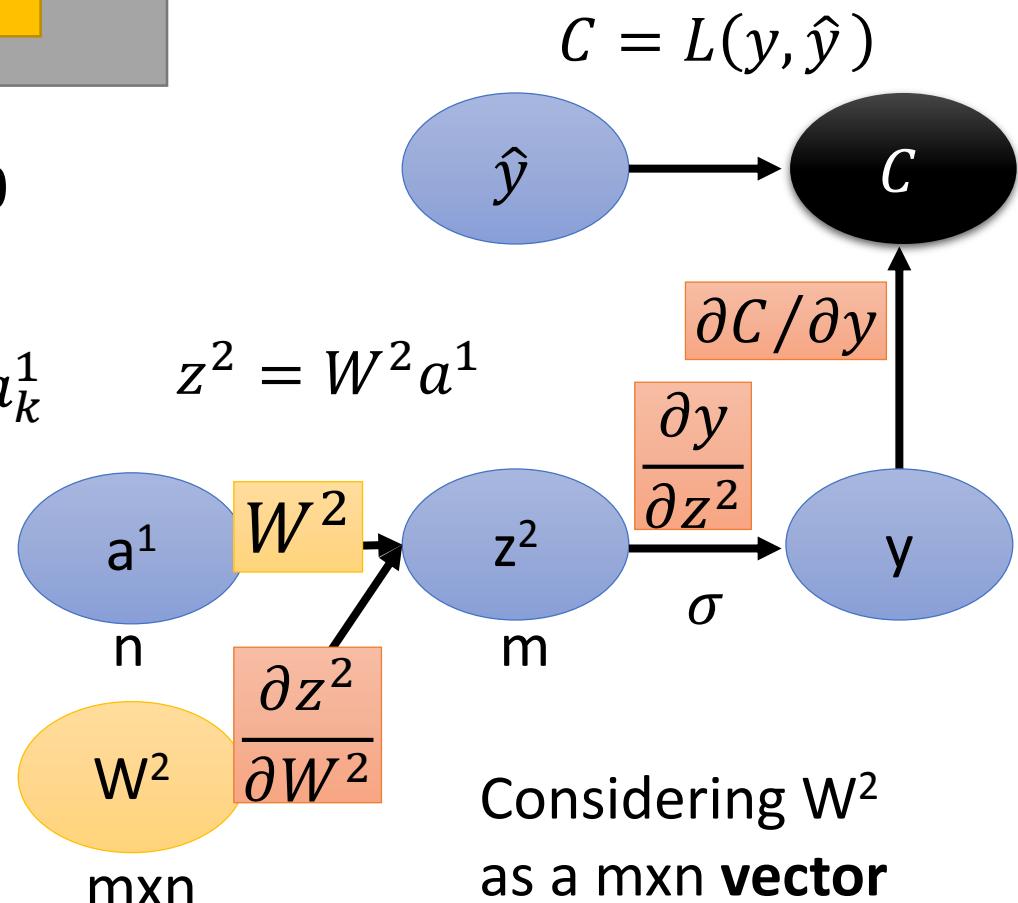
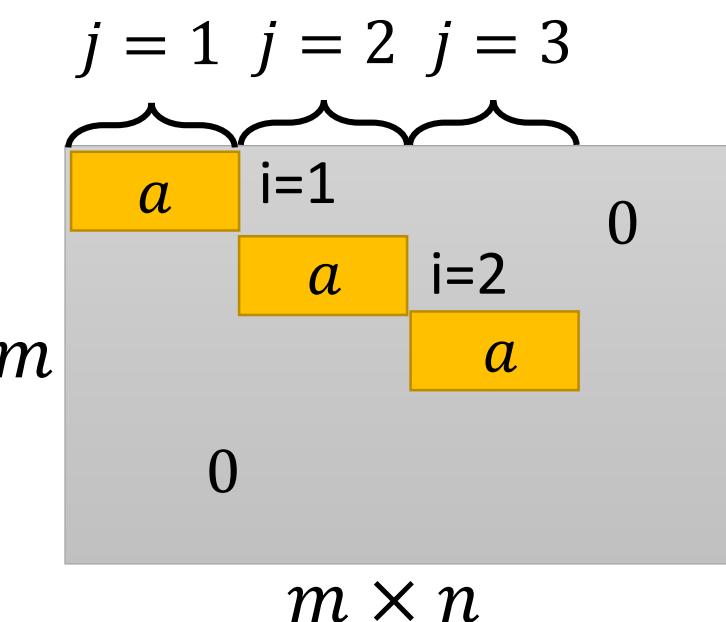
$(j-1)xn+k$

i

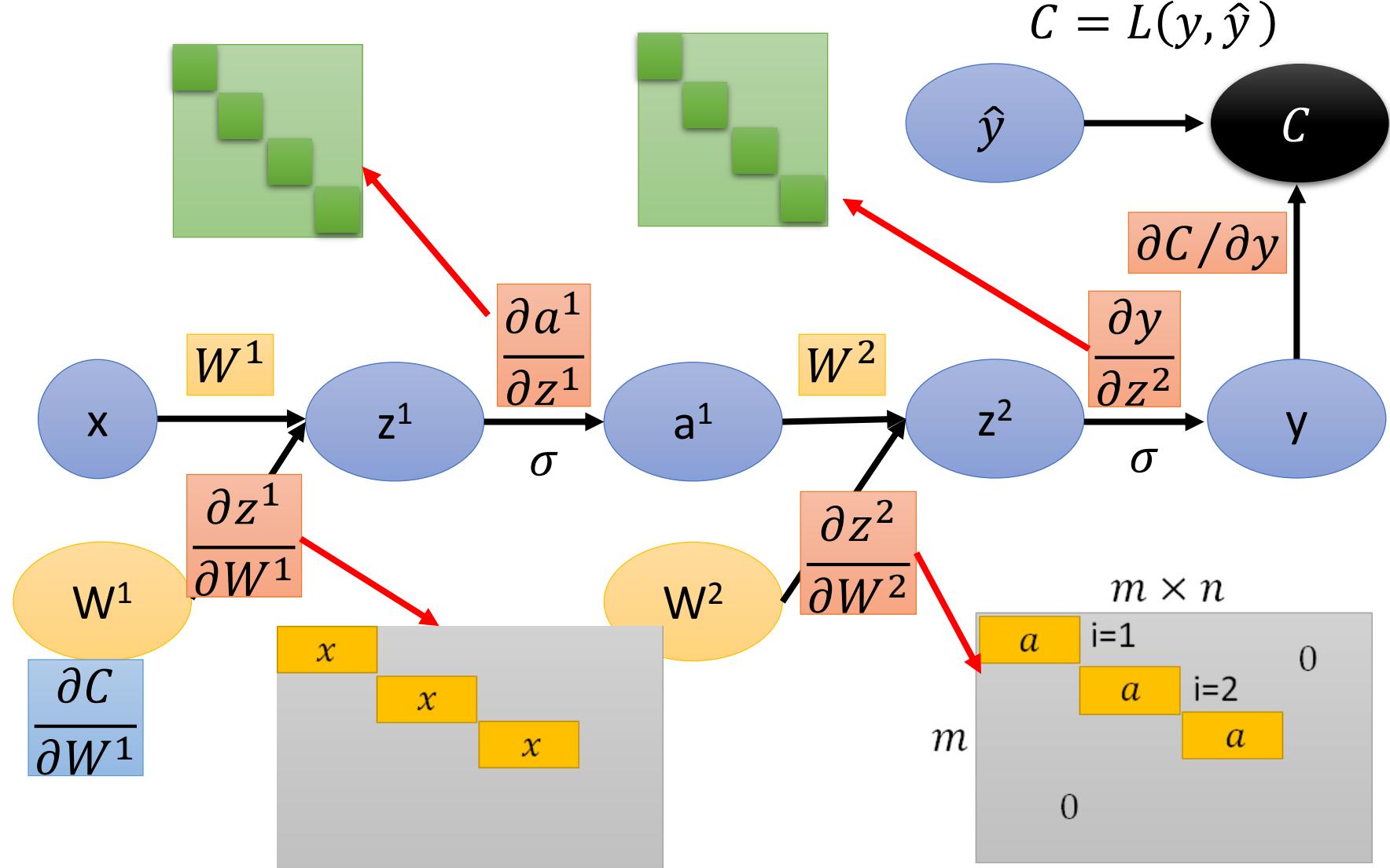
$\frac{\partial z_i^2}{\partial W_{jk}^2}$

$$\frac{\partial z_i^2}{\partial W_{jk}^2} = ? \quad i \neq j: \frac{\partial z_i^2}{\partial W_{jk}^2} = 0$$

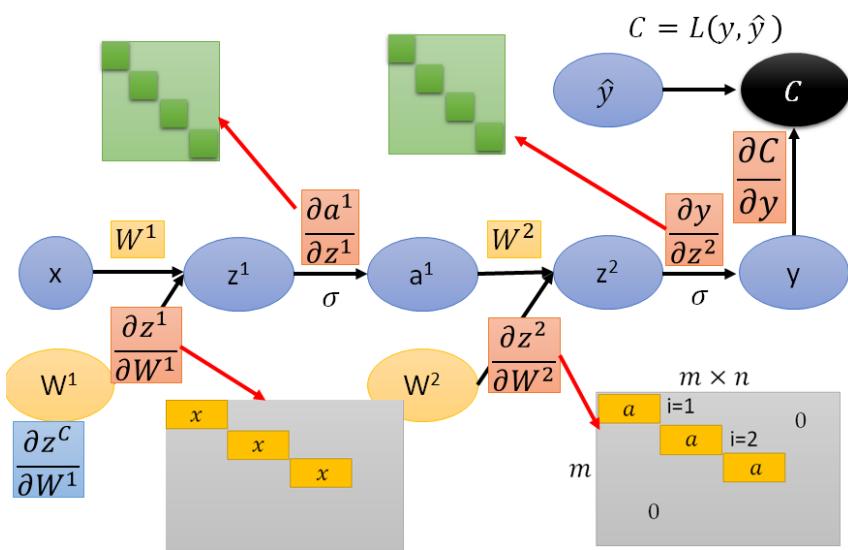
$$i = j: \frac{\partial z_i^2}{\partial W_{ik}^2} = a_k^1$$



$$\frac{\partial C}{\partial W^1} = \frac{\partial C}{\partial y} \frac{\partial y}{\partial z^2} W^2 \frac{\partial a^1}{\partial z^1} \frac{\partial z^1}{\partial W^1} = [\cdots \frac{\partial C}{\partial W_{ij}^1} \cdots]$$



Question



$$\frac{\partial C}{\partial w_{ij}^l} = \boxed{\frac{\partial z_i^l}{\partial w_{ij}^l}} \boxed{\frac{\partial C}{\partial z_i^l}}$$

Error signal

Forward Pass

$$z^1 = W^1 x + b^1$$

$$a^1 = \sigma(z^1)$$

.....

$$z^{l-1} = W^{l-1} a^{l-2} + b^{l-1}$$

$$a^{l-1} = \sigma(z^{l-1})$$

Backward Pass

$$\delta^L = \sigma'(z^L) \bullet \nabla_y C$$

$$\delta^{L-1} = \sigma'(z^{L-1}) \bullet (W^L)^T \delta^L$$

$$\delta^l = \sigma'(z^l) \bullet (W^{l+1})^T \delta^{l+1}$$

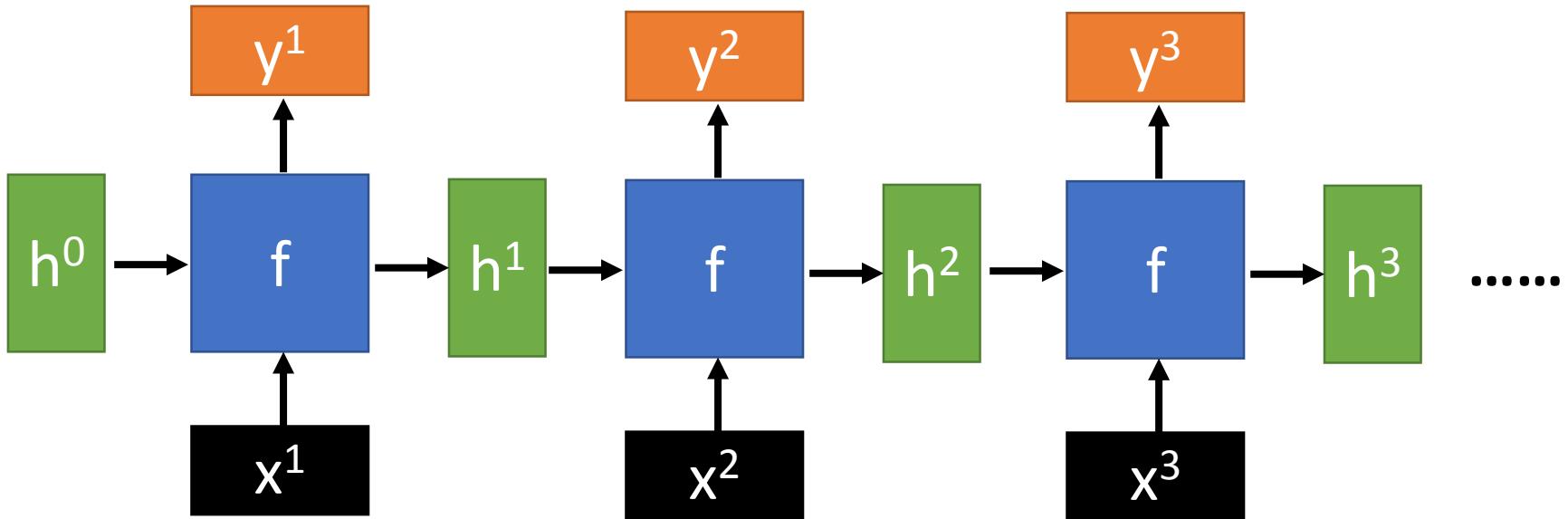
.....

Q: Only backward pass for computational graph?

Q: Do we get the same results from the two different approaches?

Computational Graph for Recurrent Network

Recurrent Network



$$y^t, h^t = f(x^t, h^{t-1}; W^i, W^h, W^o)$$

$$h^t = \sigma(W^i x^t + W^h h^{t-1})$$

$$y^t = softmax(W^o h^t)$$

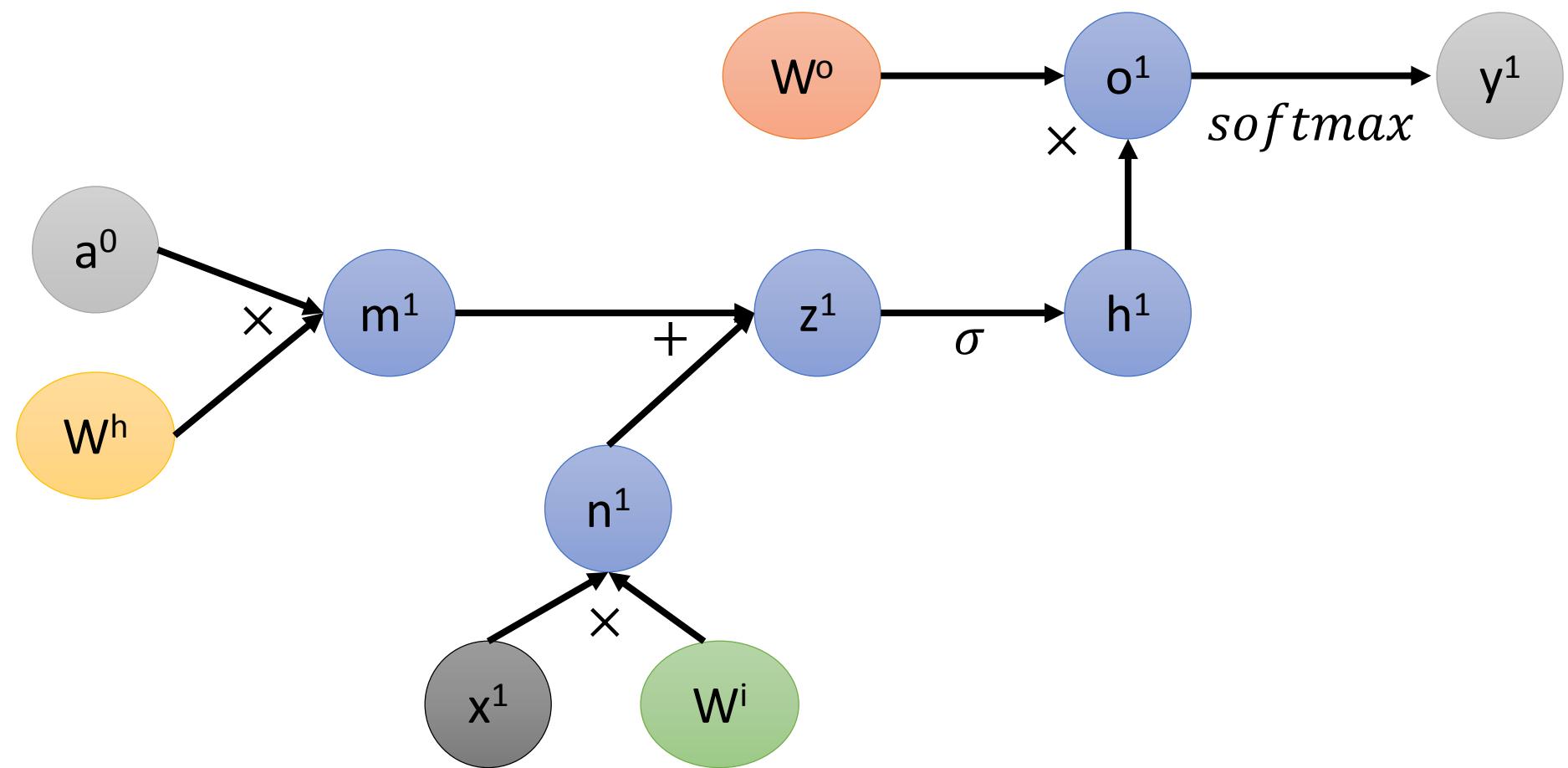
(biases are ignored here)

Recurrent Network

$$y^t, h^t = f(x^t, h^{t-1}; W^i, W^h, W^o)$$

$$a^t = \sigma(W^i x^t + W^h h^{t-1})$$

$$y^t = \text{softmax}(W^o h^t)$$

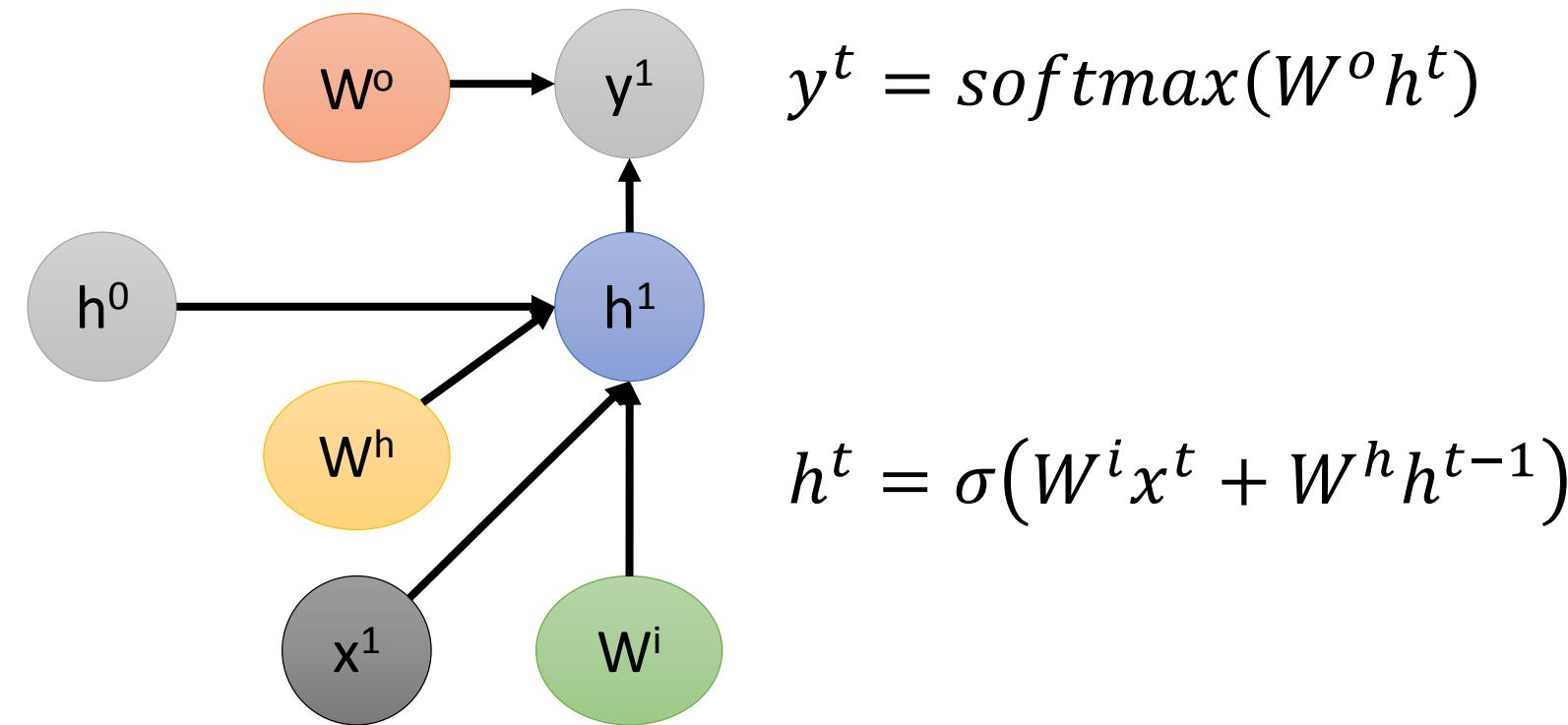


Recurrent Network

$$y^t, h^t = f(x^t, h^{t-1}; W^i, W^h, W^o)$$

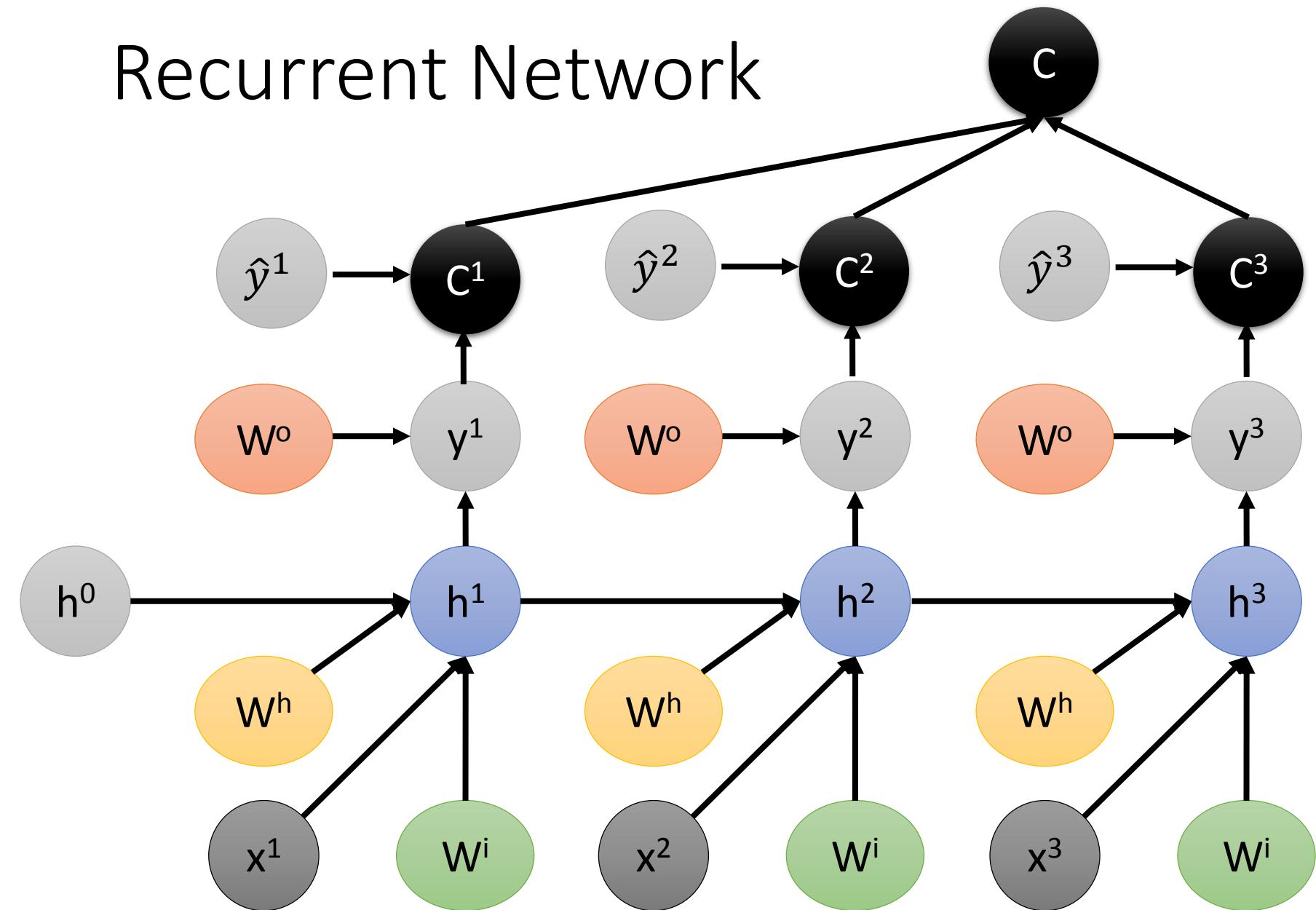
$$a^t = \sigma(W^i x^t + W^h h^{t-1})$$

$$y^t = \text{softmax}(W^o h^t)$$



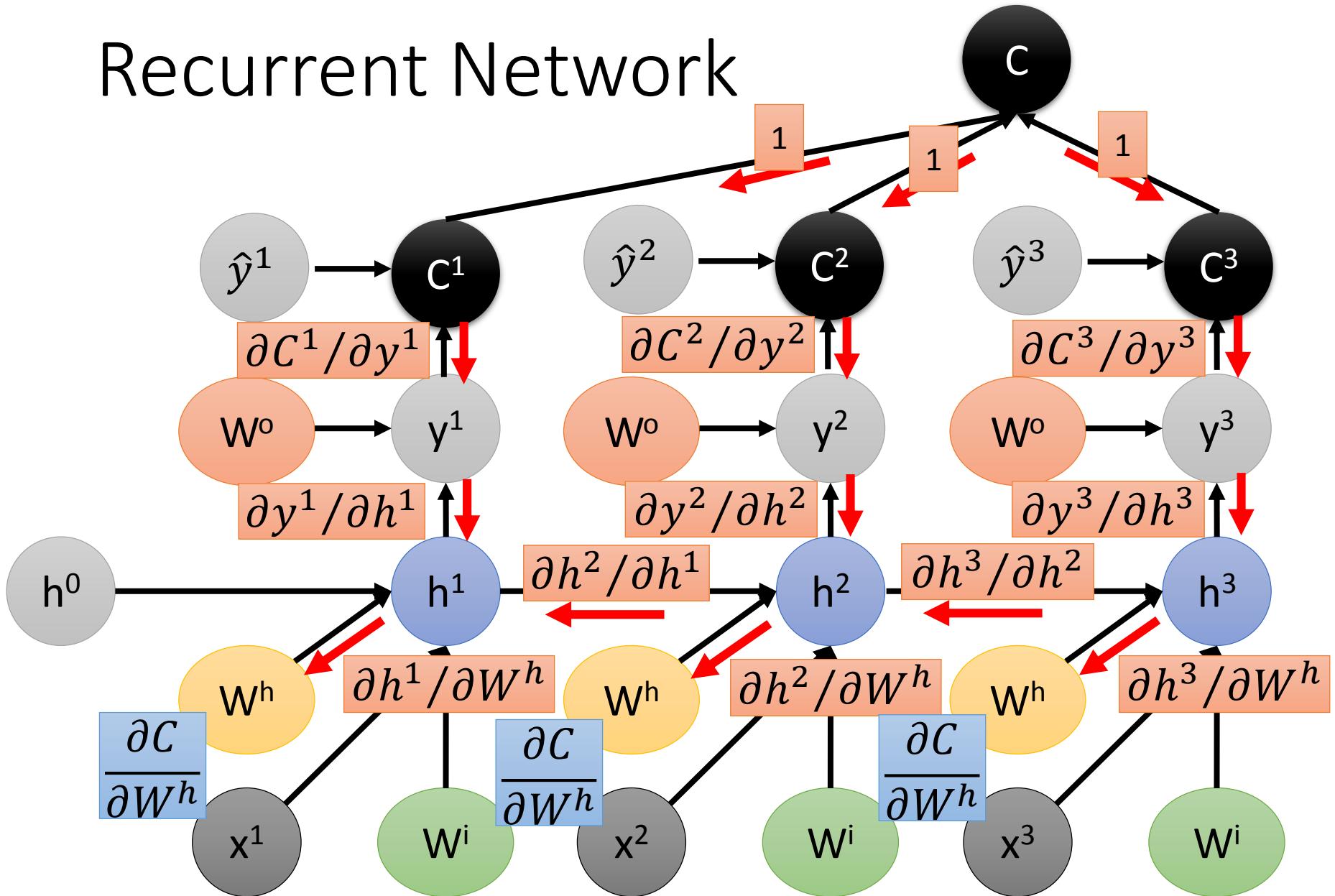
$$C = C^1 + C^2 + C^3$$

Recurrent Network



$$C = C^1 + C^2 + C^3$$

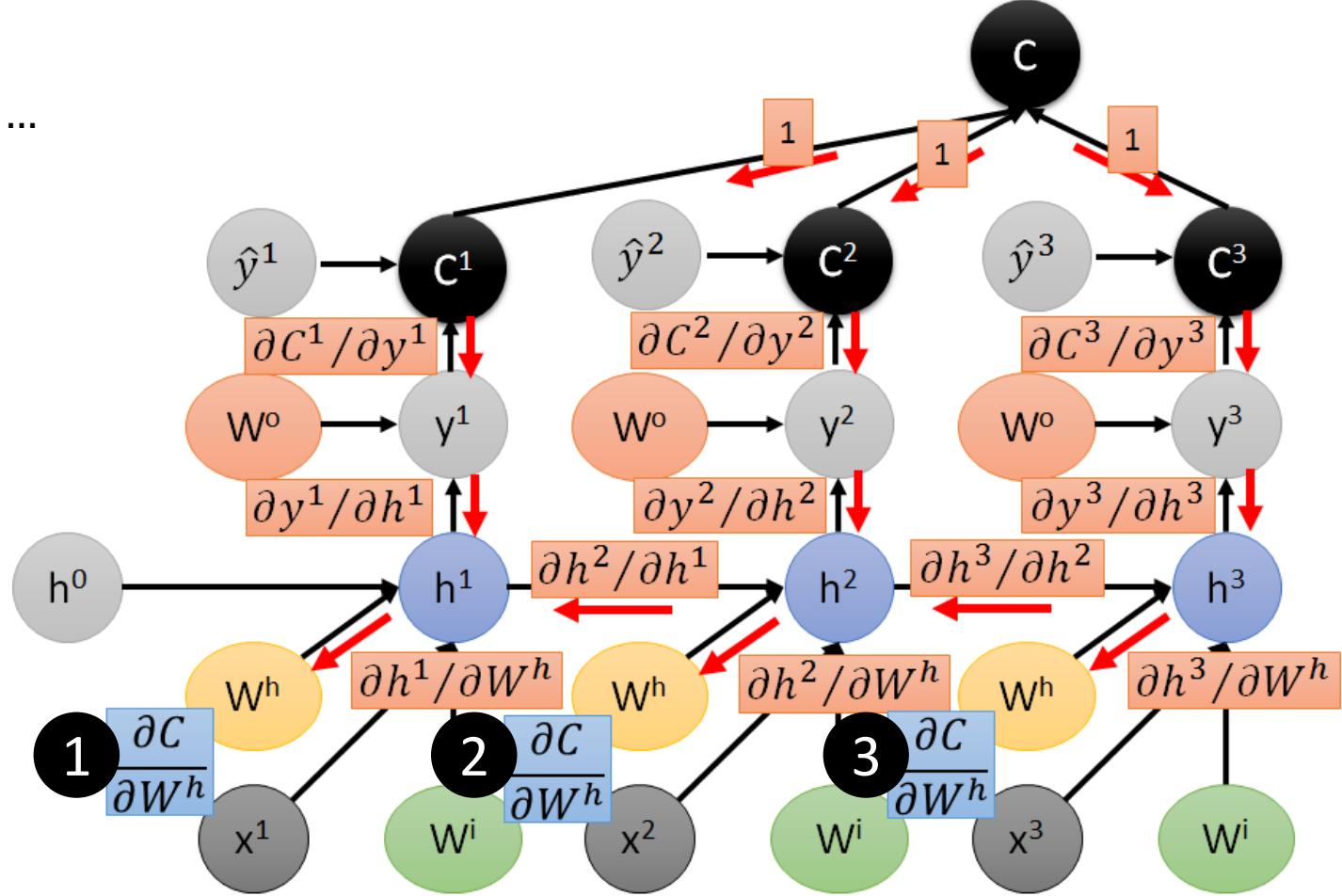
Recurrent Network



$$1 \frac{\partial C}{\partial W^h} = \left[\frac{\partial C^1}{\partial y^1} \frac{\partial y^1}{\partial h^1} + \frac{\partial C^2}{\partial y^2} \frac{\partial y^2}{\partial h^2} \frac{\partial h^2}{\partial h^1} + \frac{\partial C^3}{\partial y^3} \frac{\partial y^3}{\partial h^3} \frac{\partial h^3}{\partial h^2} \frac{\partial h^2}{\partial h^1} \right] \frac{\partial h^1}{\partial W^h}$$

$$2 \frac{\partial C}{\partial W^h} = \dots \dots \quad \frac{\partial C}{\partial W^h} = 1 \frac{\partial C}{\partial W^h} + 2 \frac{\partial C}{\partial W^h} + 3 \frac{\partial C}{\partial W^h}$$

$$3 \frac{\partial C}{\partial W^h} = \dots \dots$$



Reference

- Textbook: Deep Learning
 - Chapter 6.5
- Calculus on Computational Graphs:
Backpropagation
 - <https://colah.github.io/posts/2015-08-Backprop/>
- On chain rule, computational graphs, and backpropagation
 - <http://outlace.com/Computational-Graph/>

Acknowledgement

- 感謝 翁丞世 同學找到投影片上的錯誤

Sequence Generation

Hung-yi Lee

李宏毅

Outline

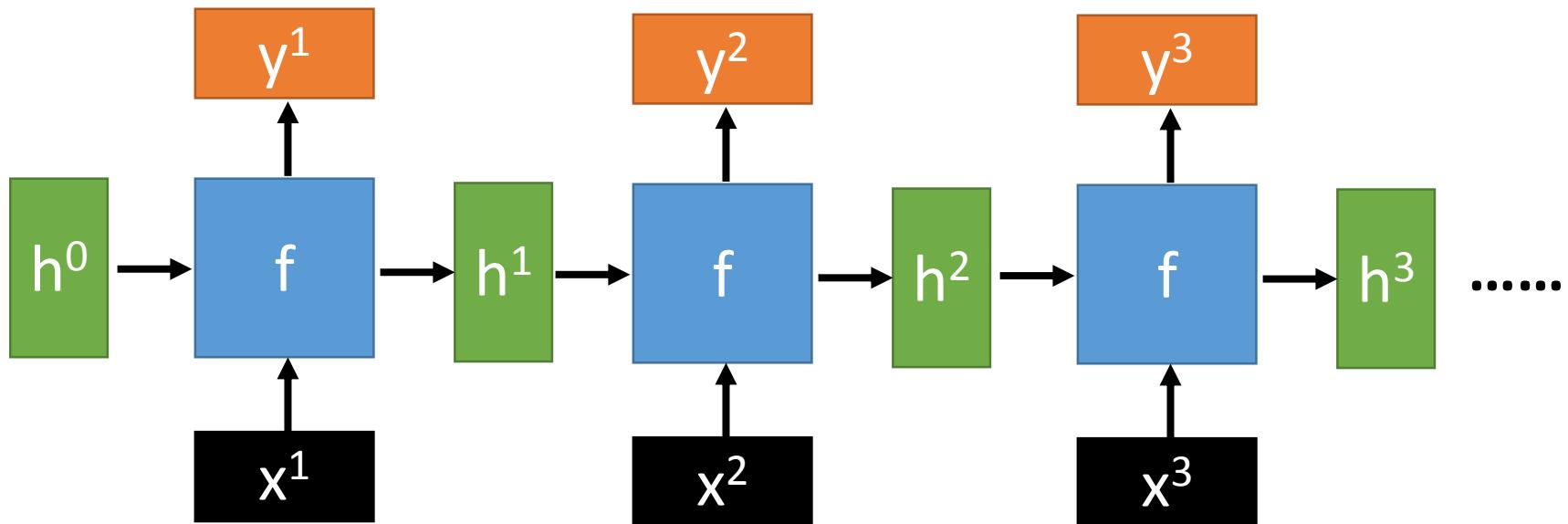
- RNN with Gated Mechanism
- Sequence Generation
- Conditional Sequence Generation
- Tips for Generation

RNN with Gated Mechanism

Recurrent Neural Network

- Given function $f: h', y = f(h, x)$

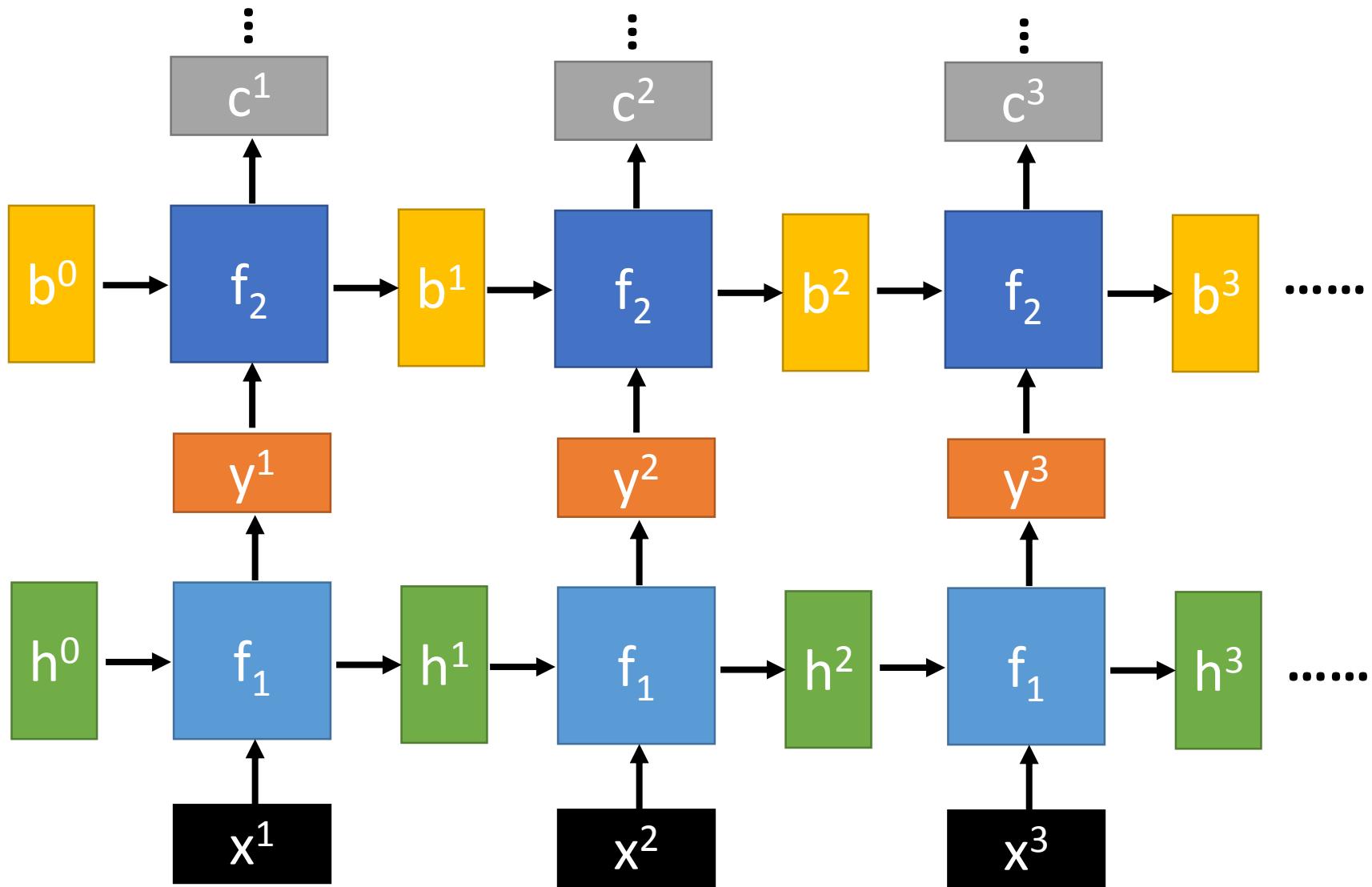
h and h' are vectors with the same dimension



No matter how long the input/output sequence is,
we only need one function f

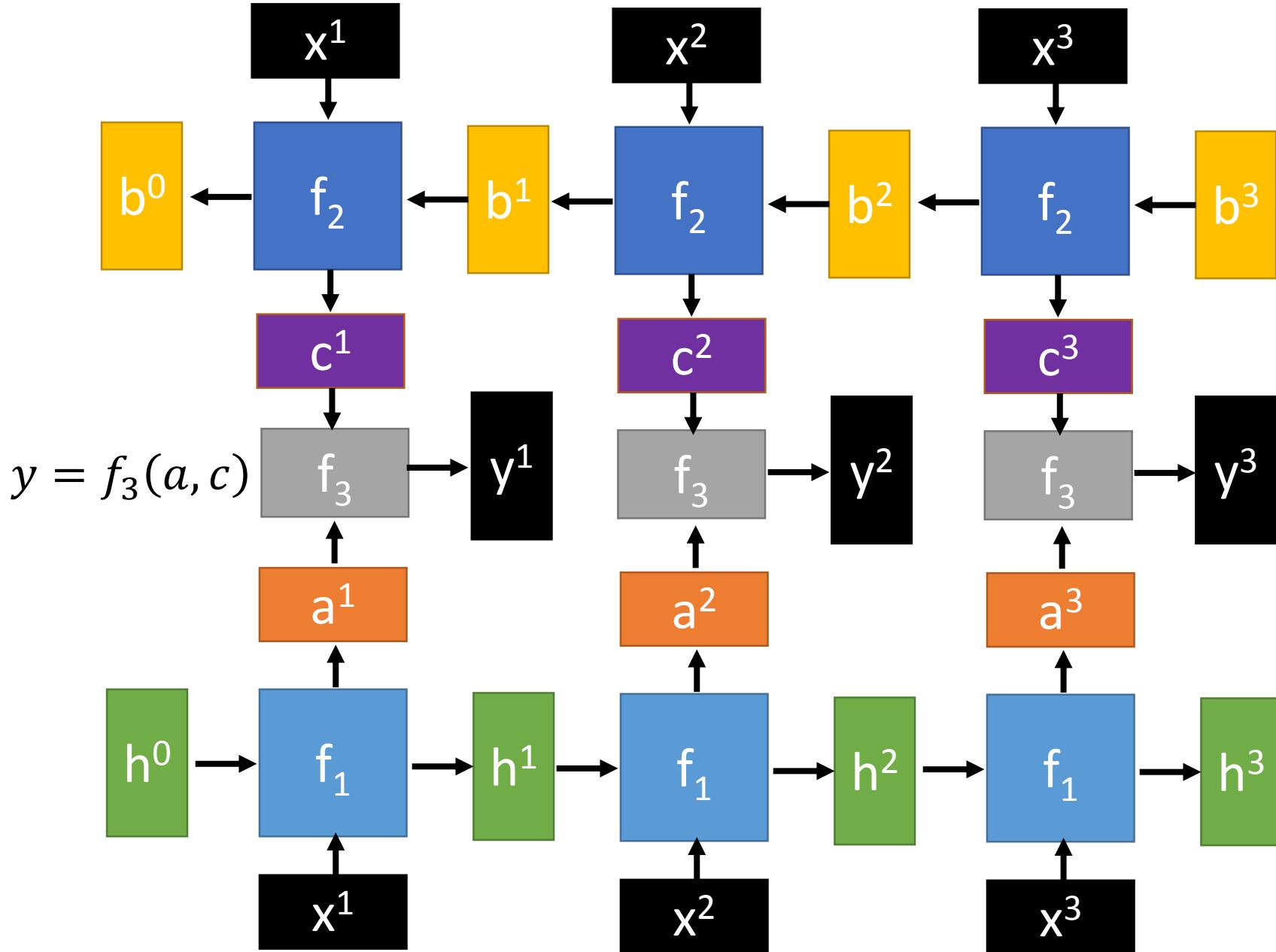
Deep RNN

$$h', y = f_1(h, x) \quad b', c = f_2(b, y) \quad \dots$$



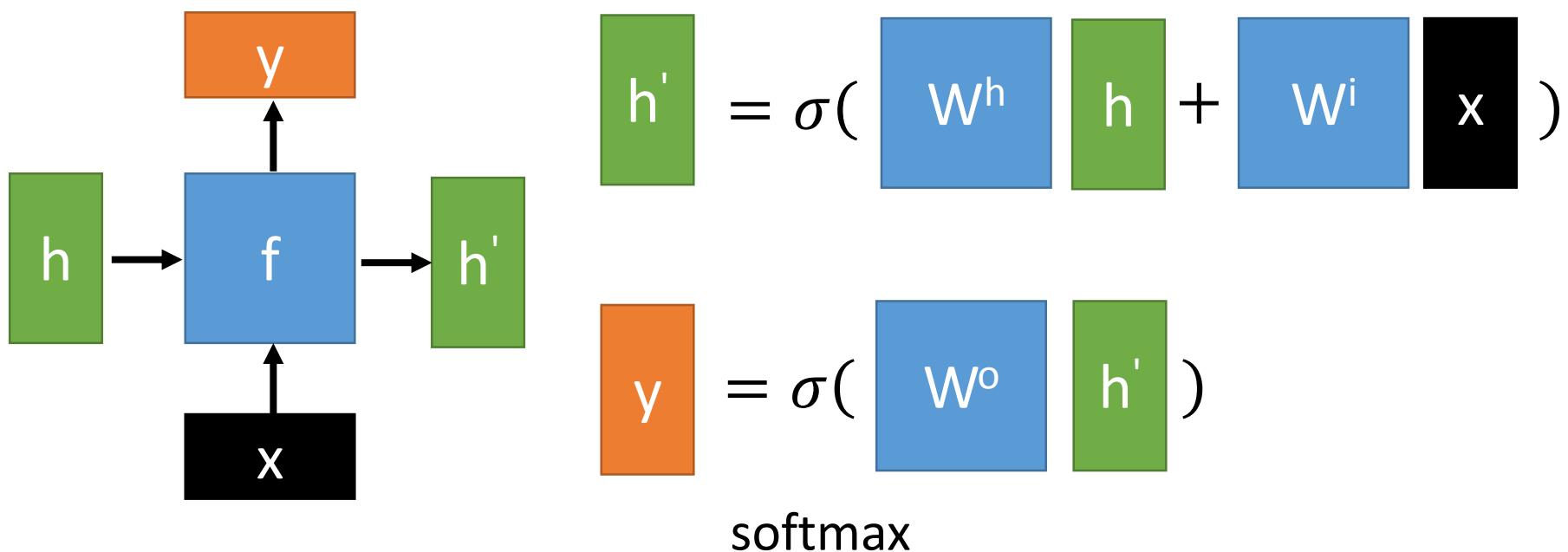
Bidirectional RNN

$$h', a = f_1(h, x) \quad b', c = f_2(b, x)$$



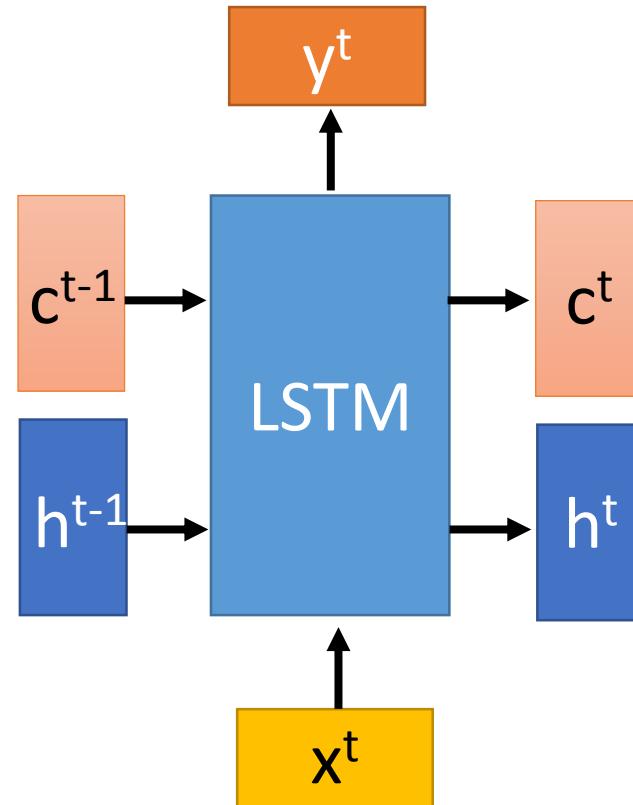
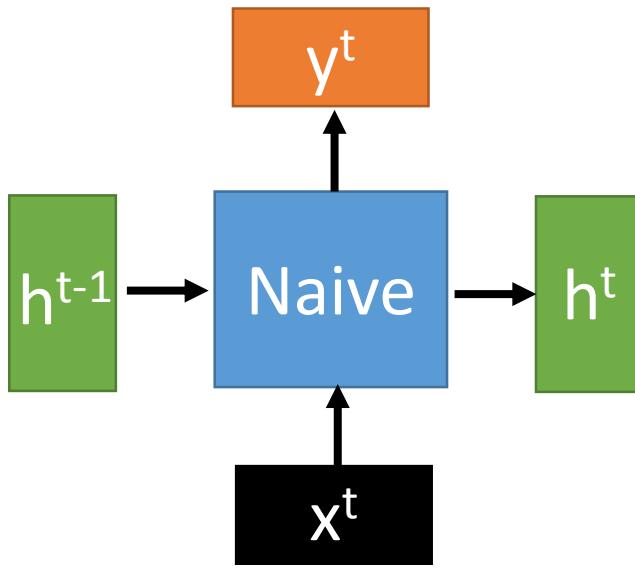
Naïve RNN

- Given function $f: h', y = f(h, x)$



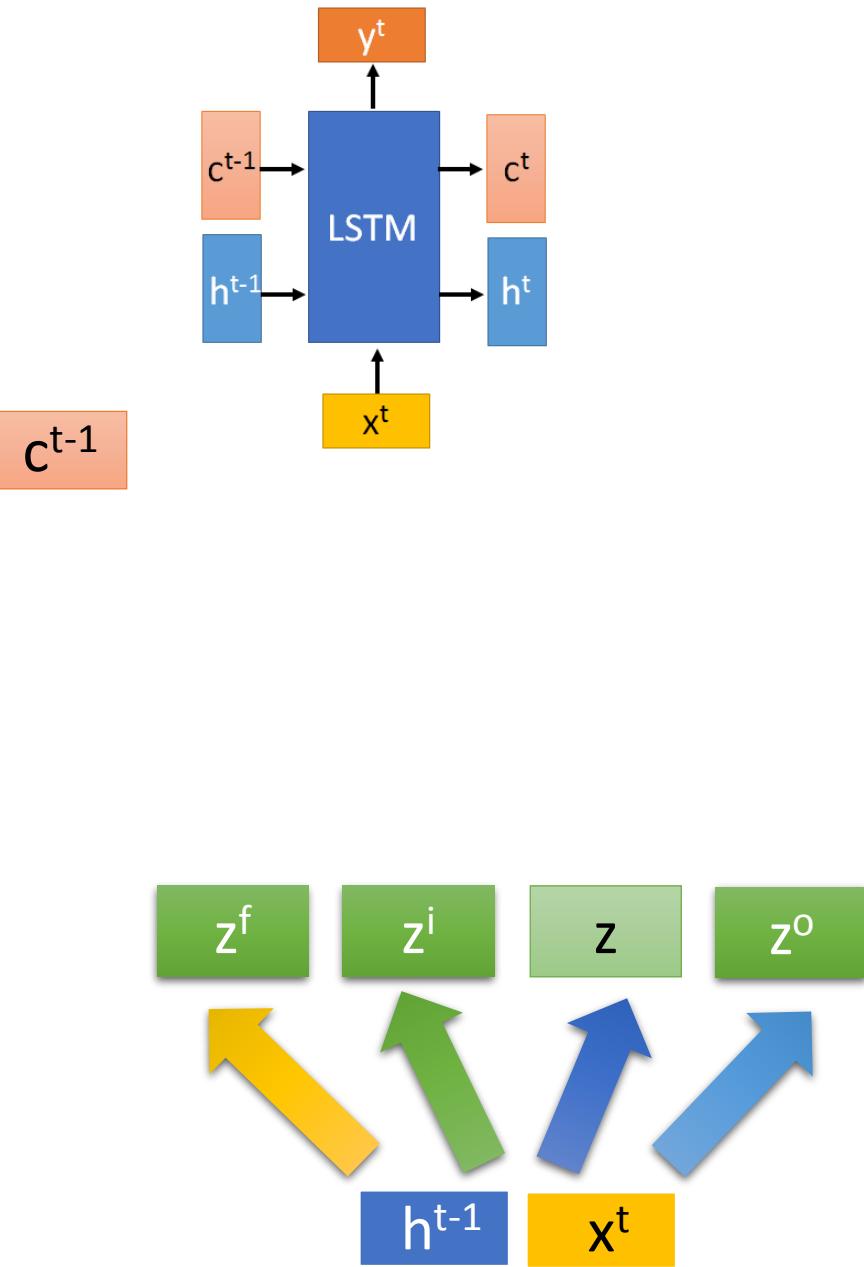
Ignore bias here

LSTM



c changes slowly $\rightarrow c^t$ is c^{t-1} added by something

h changes faster $\rightarrow h^t$ and h^{t-1} can be very different



$$z = \tanh(W h^{t-1} + x^t)$$

$$z^i = \sigma(W^i h^{t-1} + x^t)$$

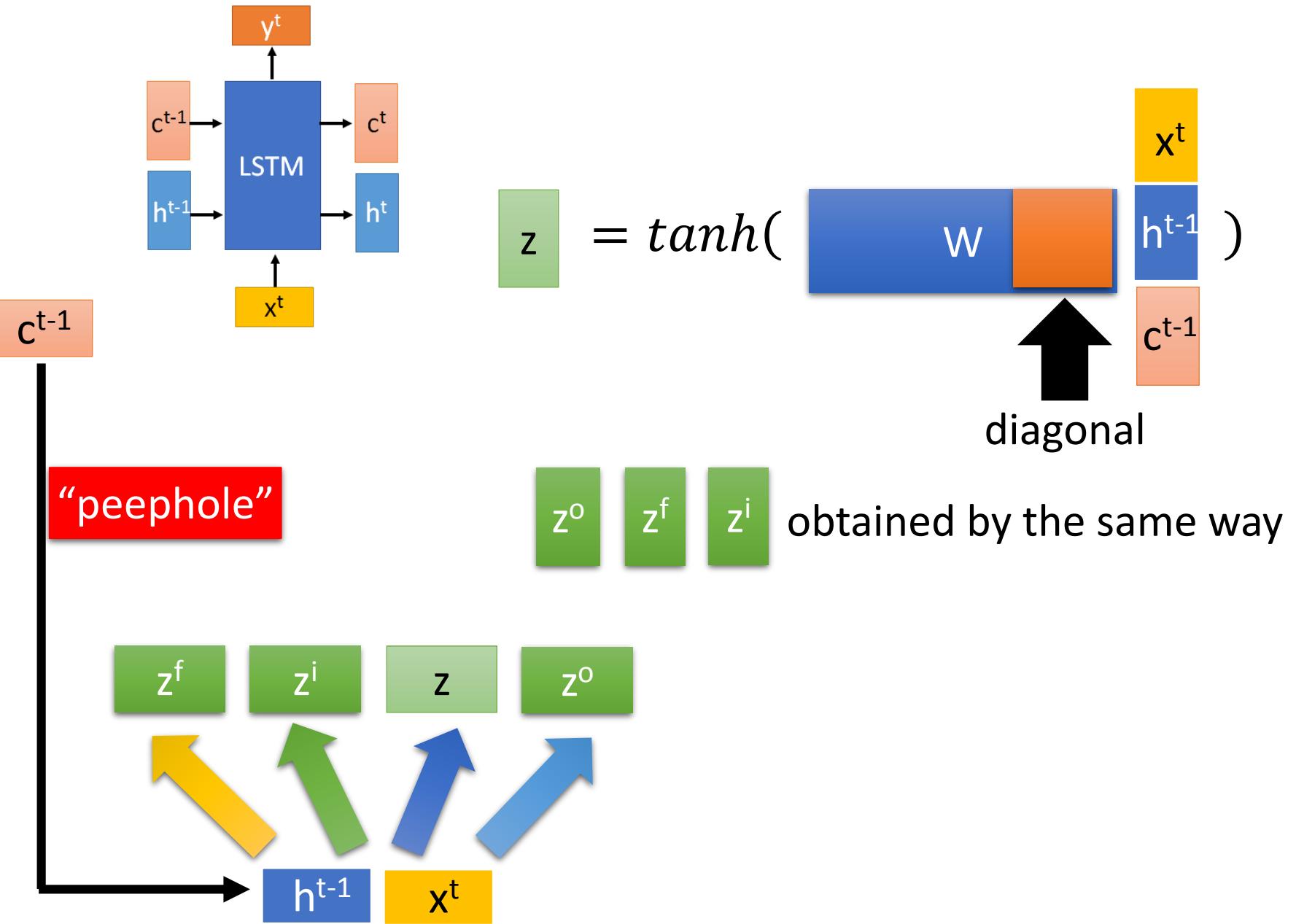
Input gate

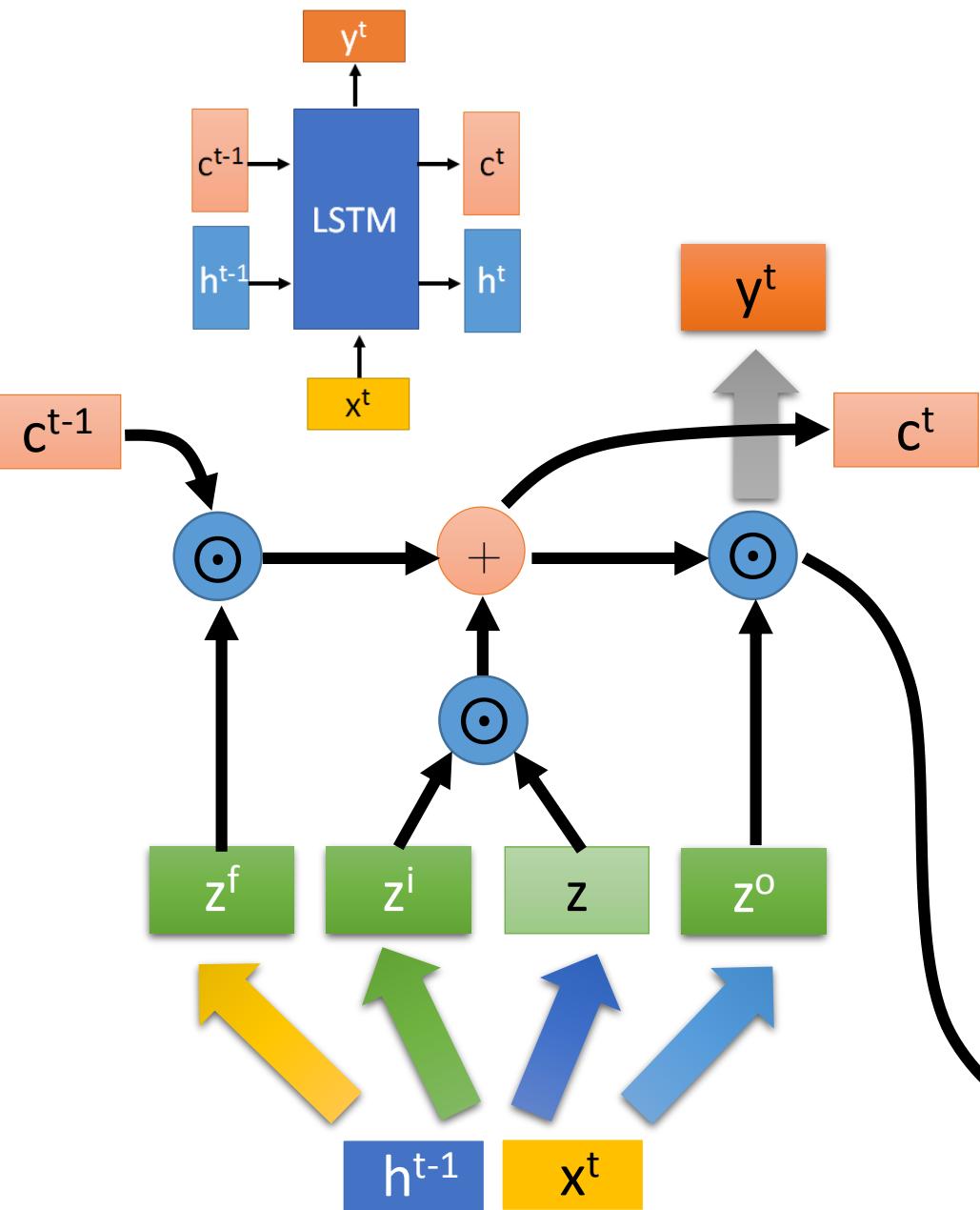
$$z^f = \sigma(W^f h^{t-1} + x^t)$$

forget gate

$$z^o = \sigma(W^o h^{t-1} + x^t)$$

output gate





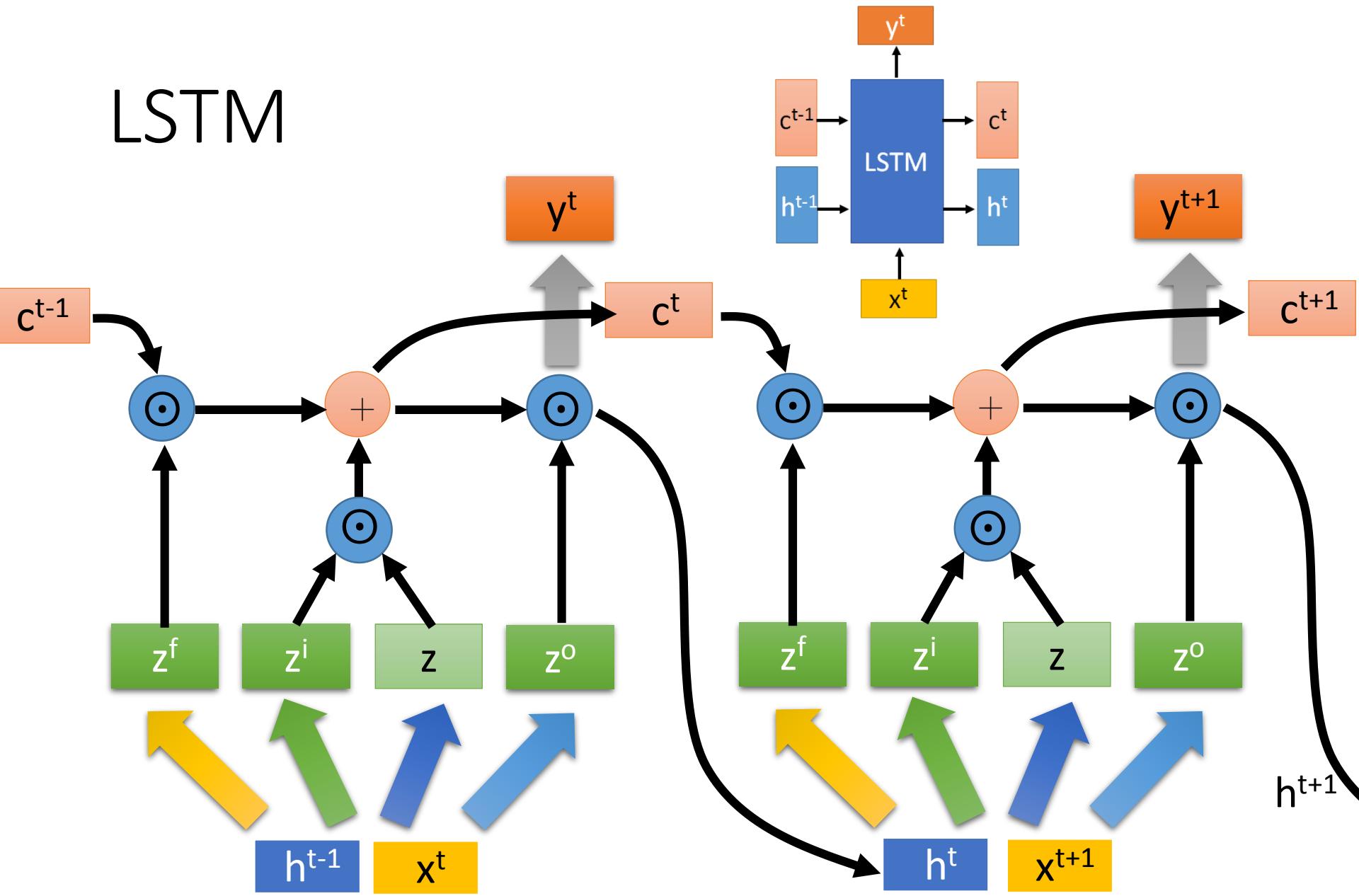
$$c^t = z^f \odot c^{t-1} + z^i \odot z$$

$$h^t = z^o \odot \tanh(c^t)$$

$$y^t = \sigma(W' h^t)$$

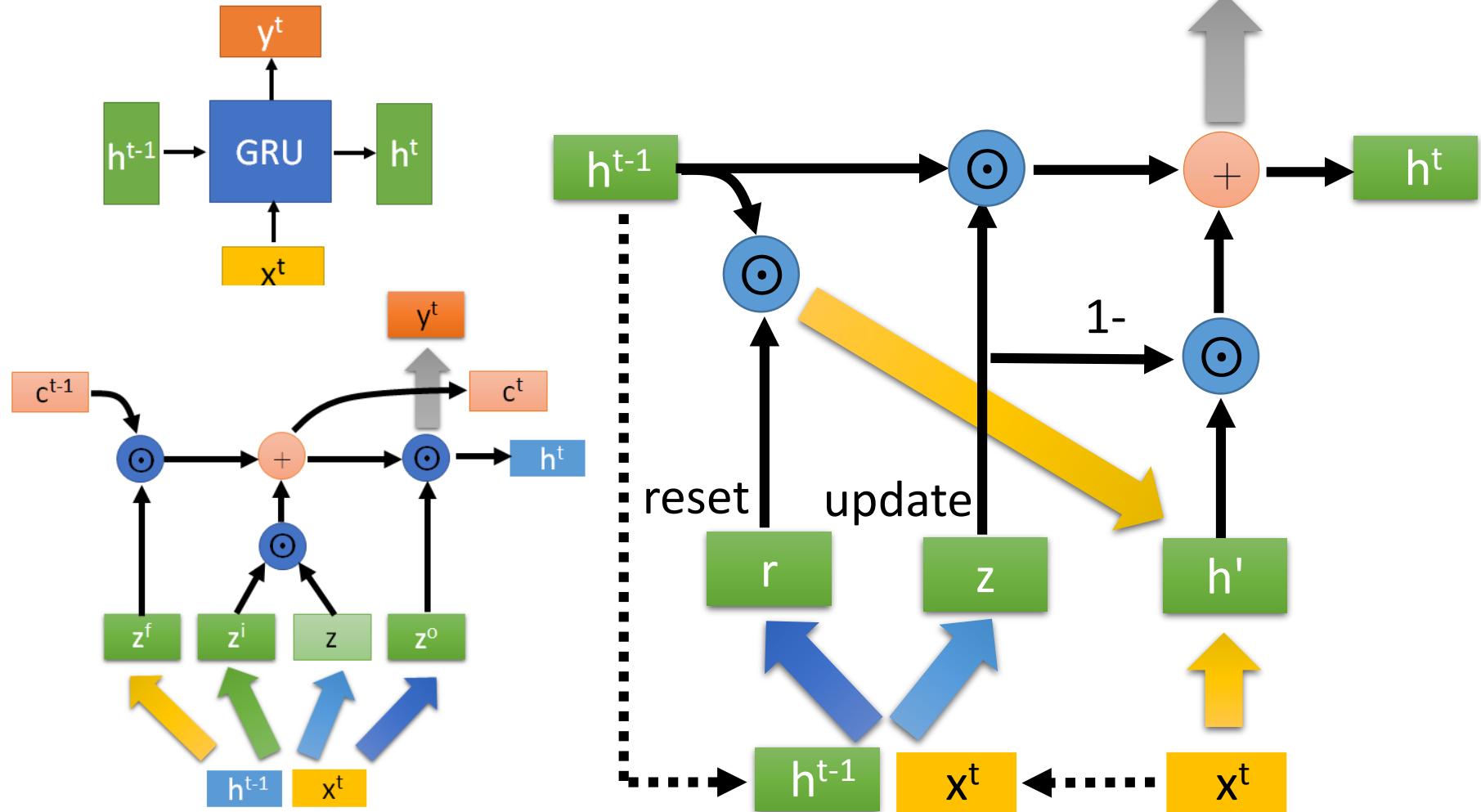
h^t

LSTM

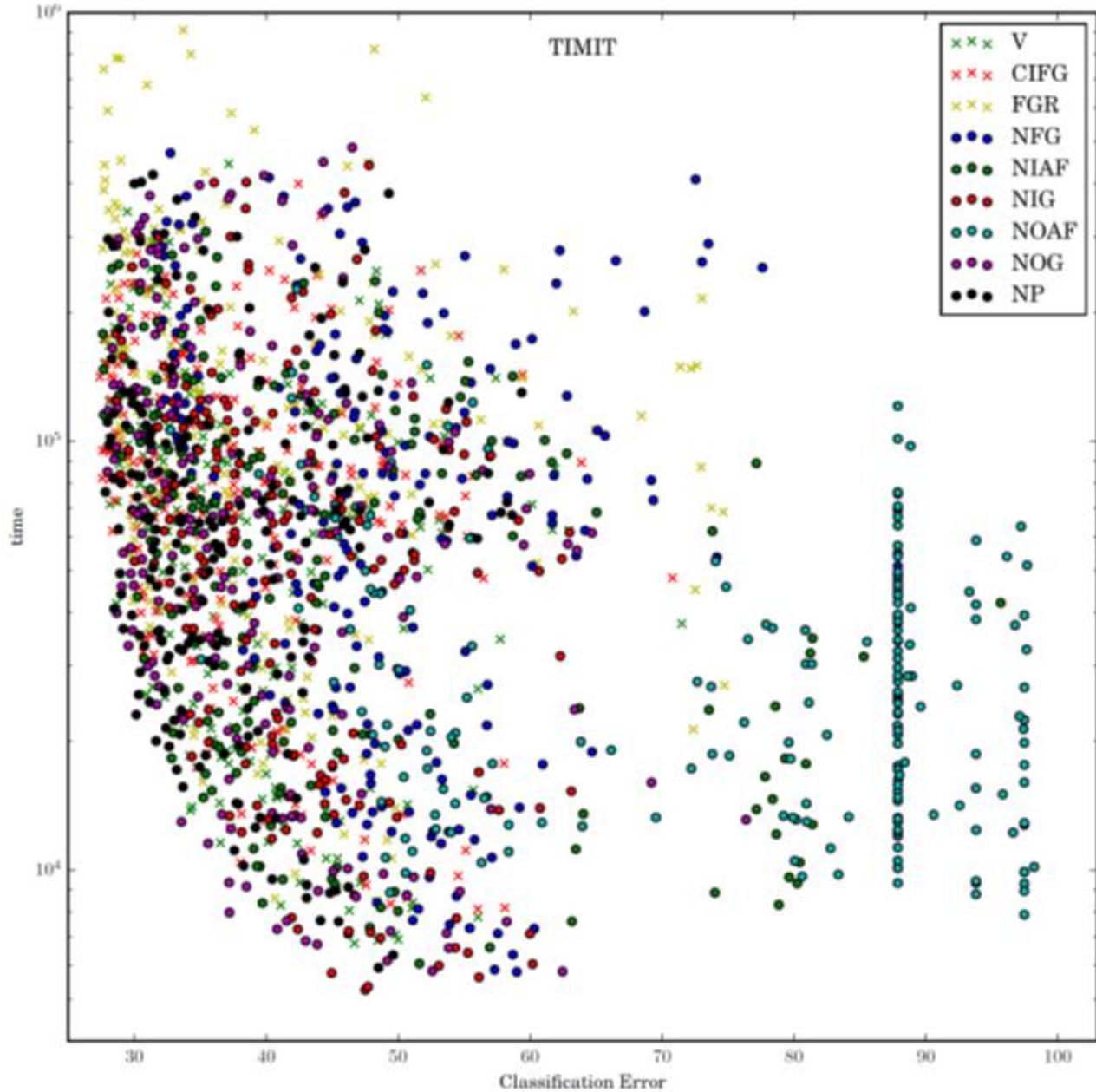


GRU

$$h^t = z \odot h^{t-1} + (1 - z) \odot h'$$

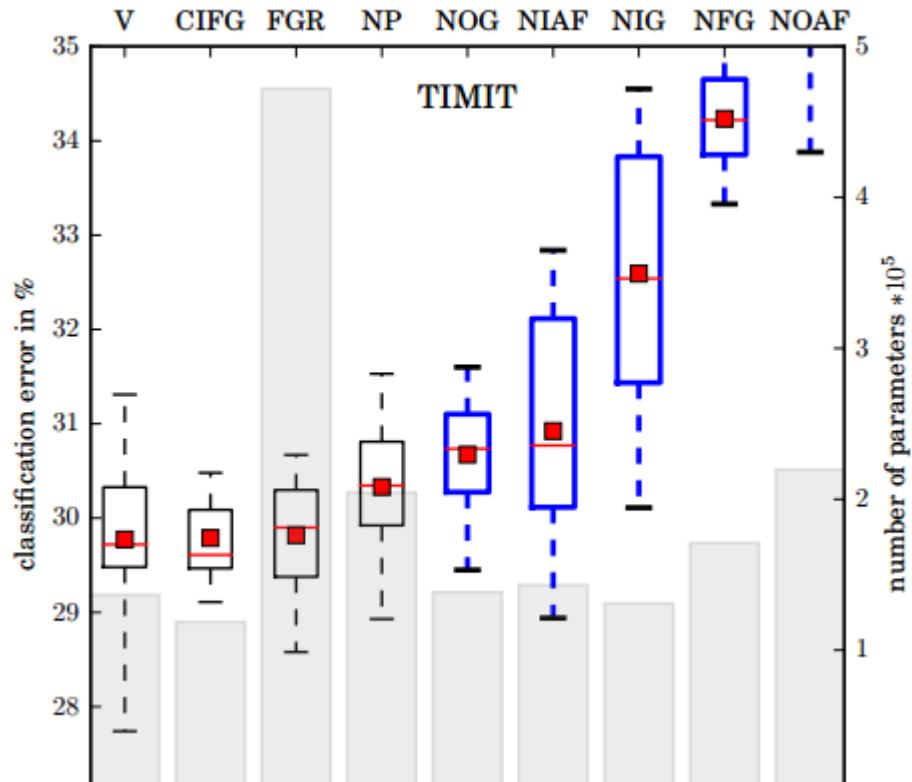


LSTM: A Search Space Odyssey



LSTM: A Search Space Odyssey

1. No Input Gate (NIG)
2. No Forget Gate (NFG)
3. No Output Gate (NOG)
4. No Input Activation Function (NIAF)
5. No Output Activation Function (NOAF)
6. No Peepholes (NP)
7. Coupled Input and Forget Gate (CIFG)
8. Full Gate Recurrence (FGR)



Standard LSTM works well

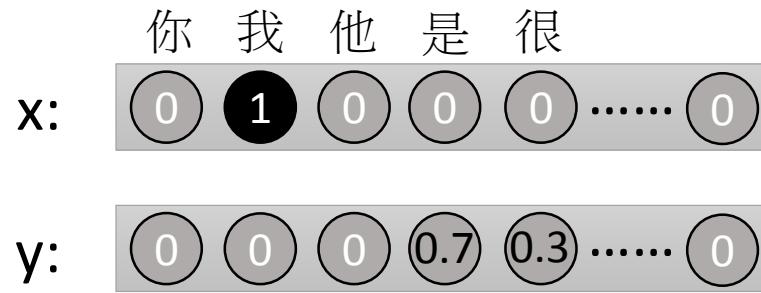
Simply LSTM: coupling input and forget gate, removing peephole

Forget gate is critical for performance

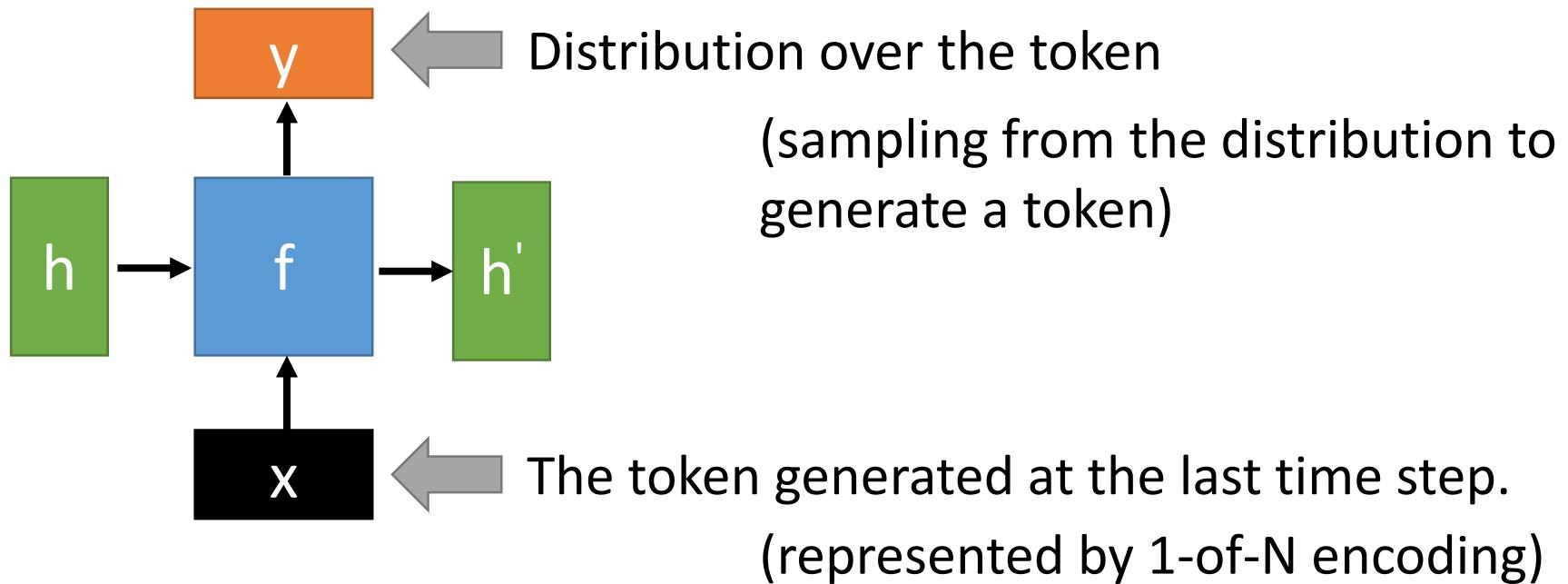
Output gate activation function is critical

Sequence Generation

Generation



- Sentences are composed of characters/words
- Generating a character/word at each time by RNN



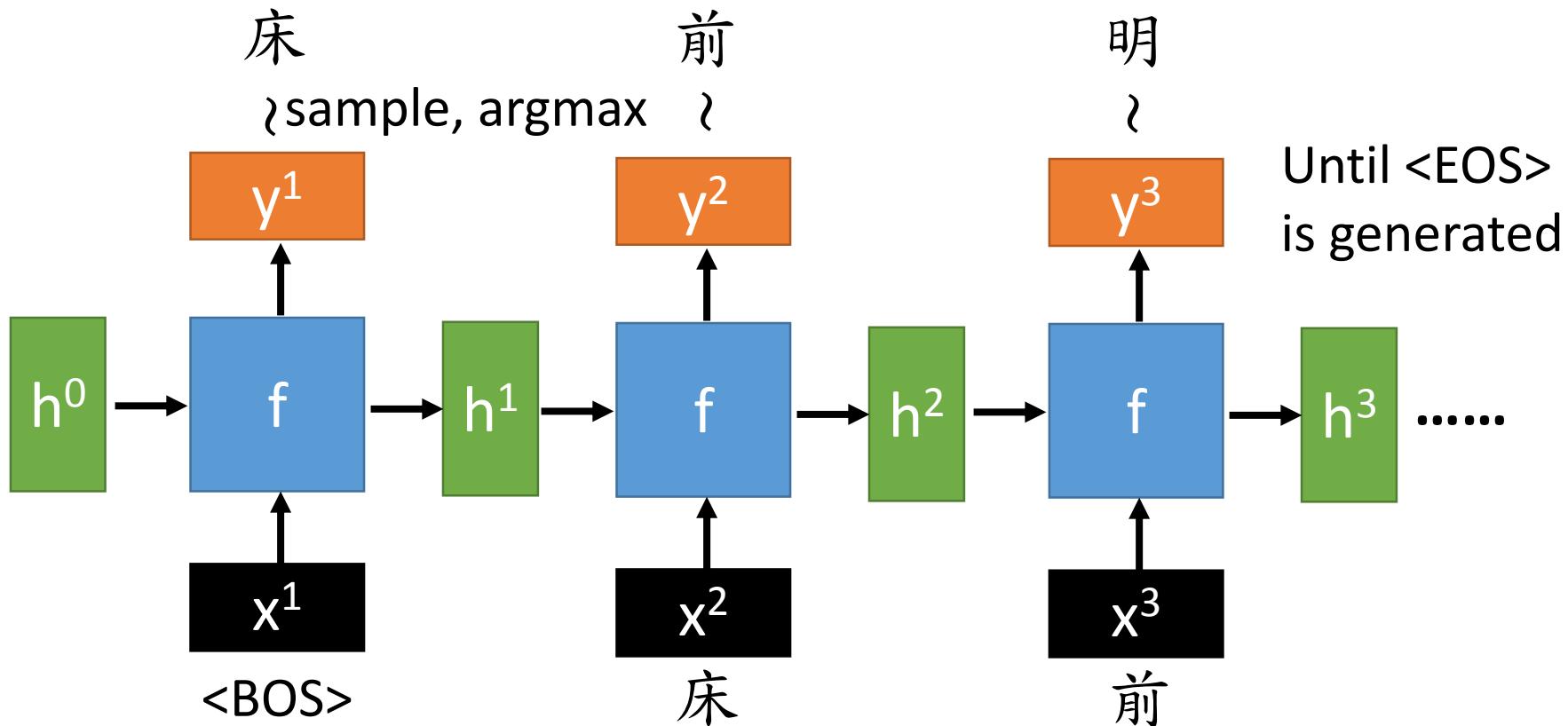
Generation

$y^1: P(w | \text{<BOS>})$

$y^2: P(w | \text{<BOS>, 床})$

$y^3: P(w | \text{<BOS>, 床, 前})$

- Sentences are composed of characters/words
- Generating a character/word at each time by RNN



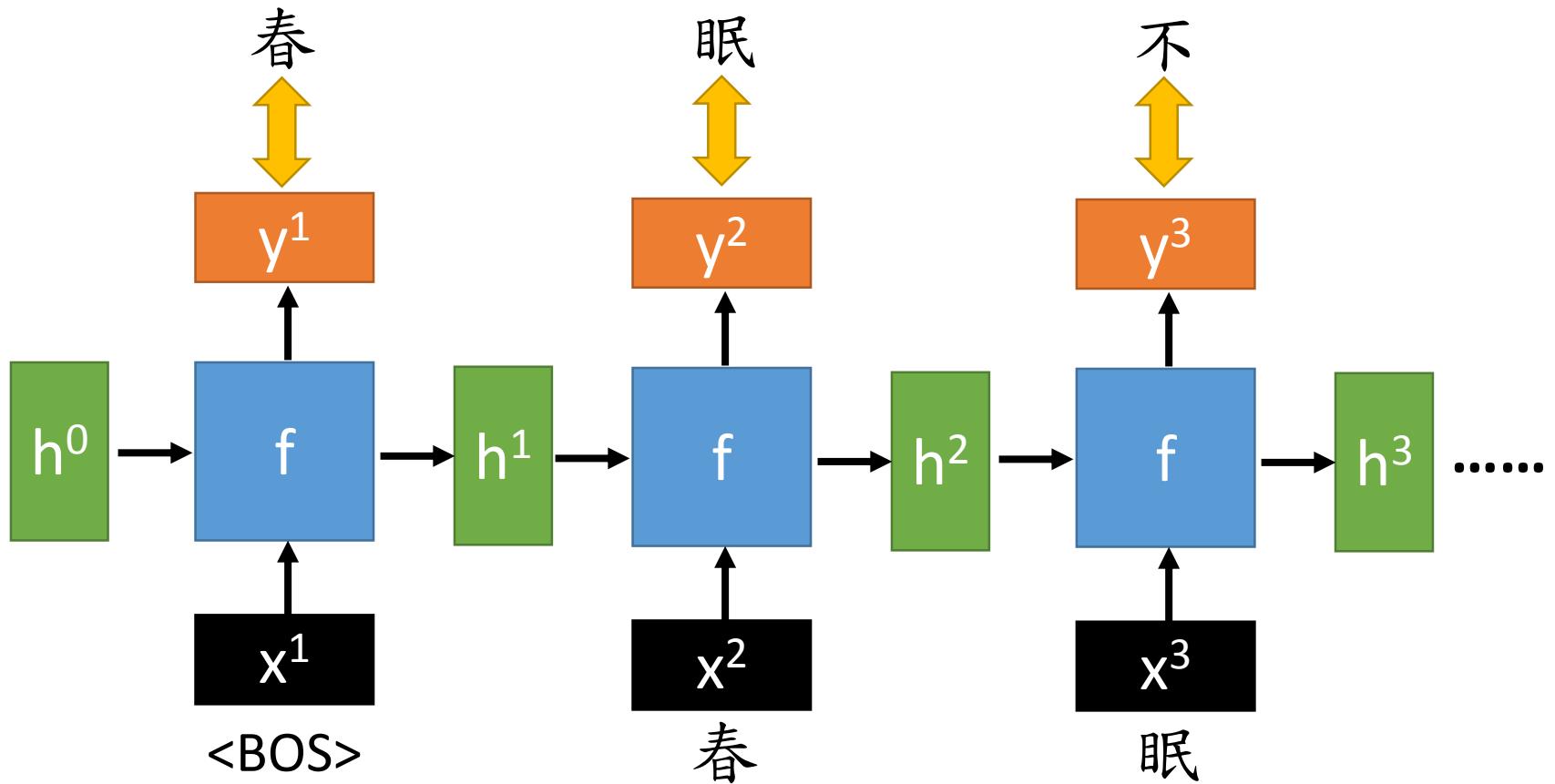
Generation



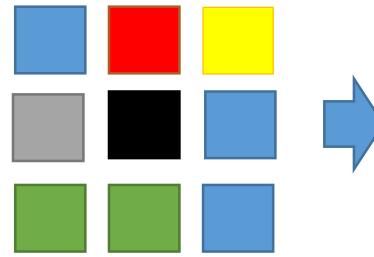
: minimizing cross-entropy

- Training

Training data: 春 眠 不 覺 曉



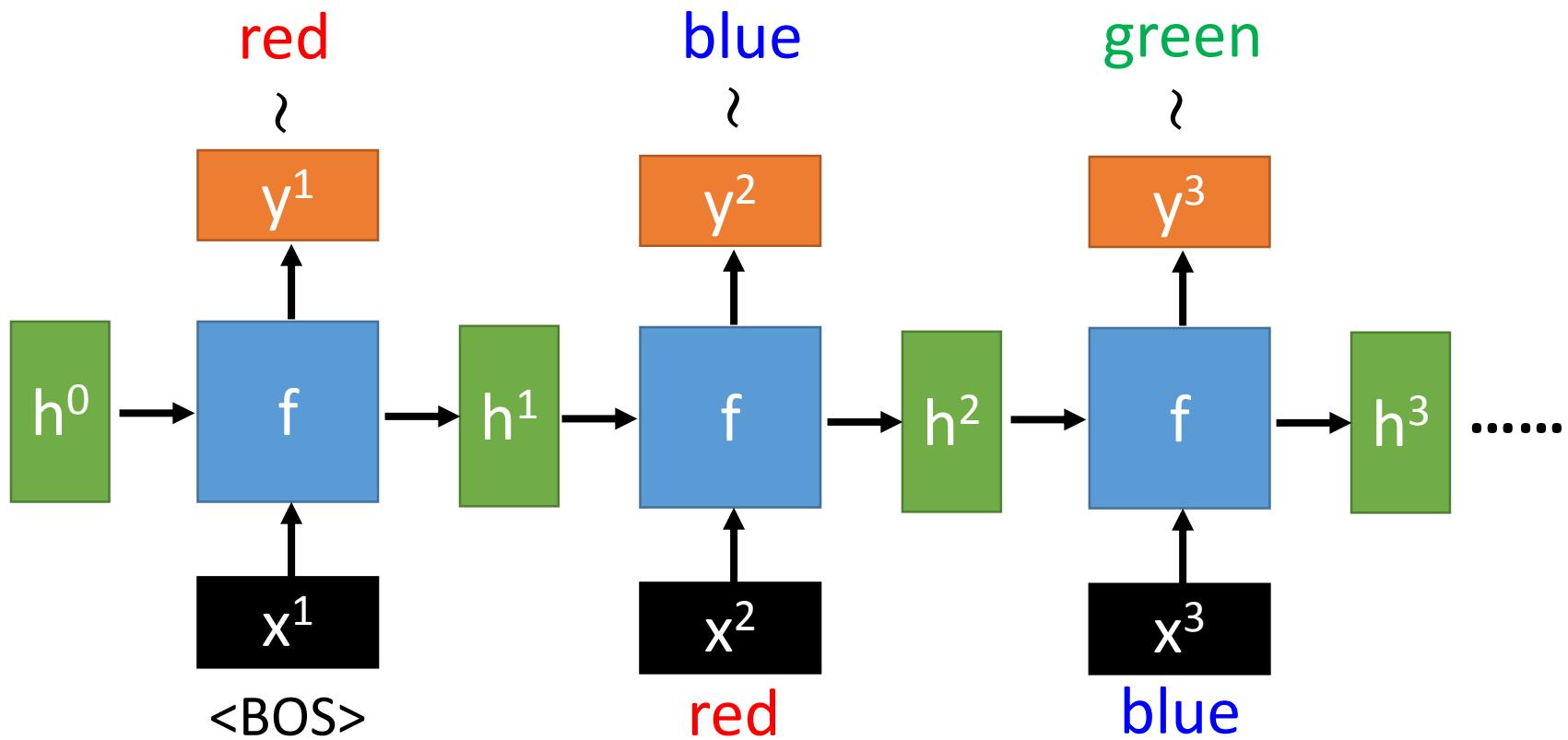
Generation



Consider as a sentence
blue red yellow gray

Train a RNN based on the
“sentences”

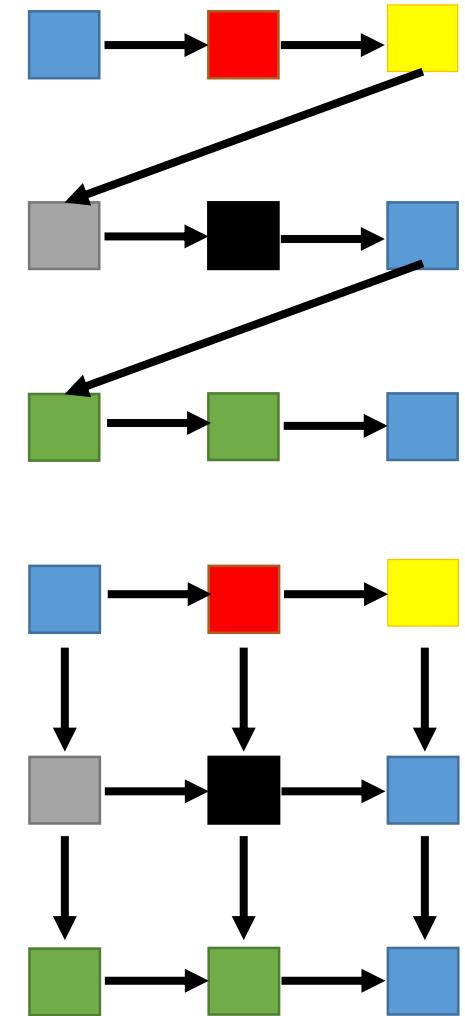
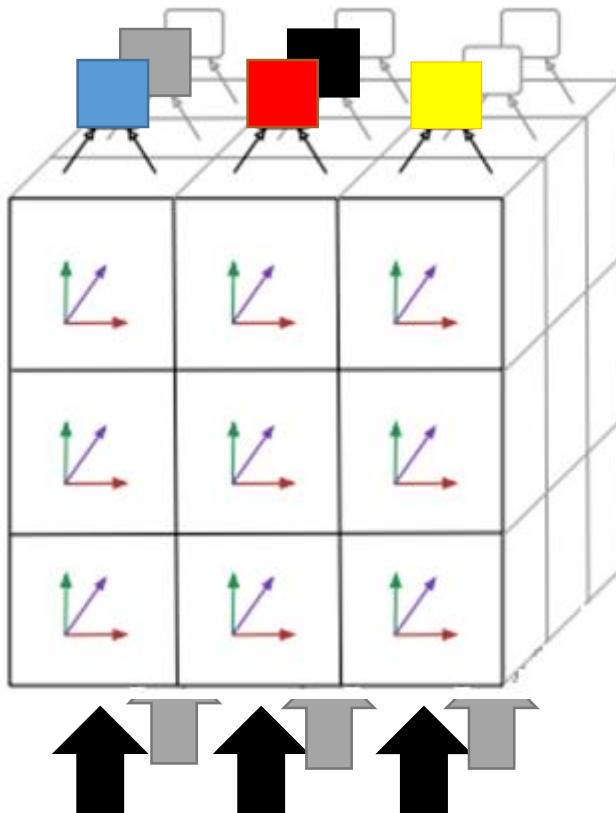
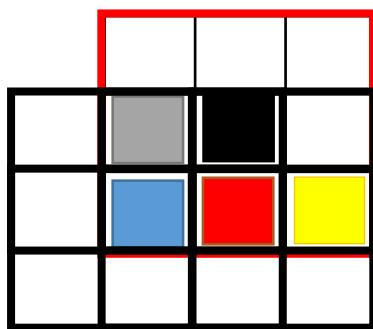
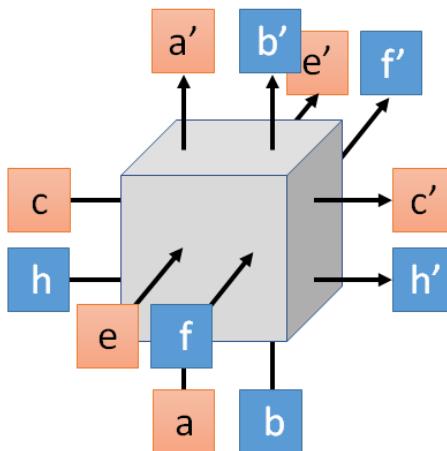
- Images are composed of pixels
- Generating a pixel at each time by RNN



Generation - PixelRNN

3 x 3 images

- Images are composed of pixels



Conditional Sequence Generation

Conditional Generation

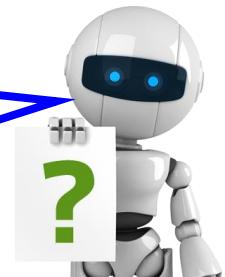
- We don't want to simply generate some random sentences.
- Generate sentences based on conditions:

Caption Generation

Given
condition:

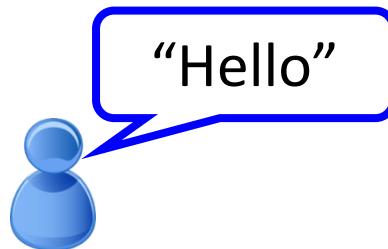


“A young girl
is dancing.”

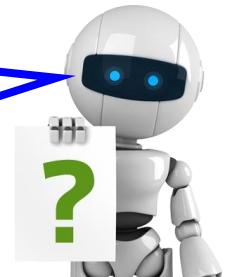


Chat-bot

Given
condition:



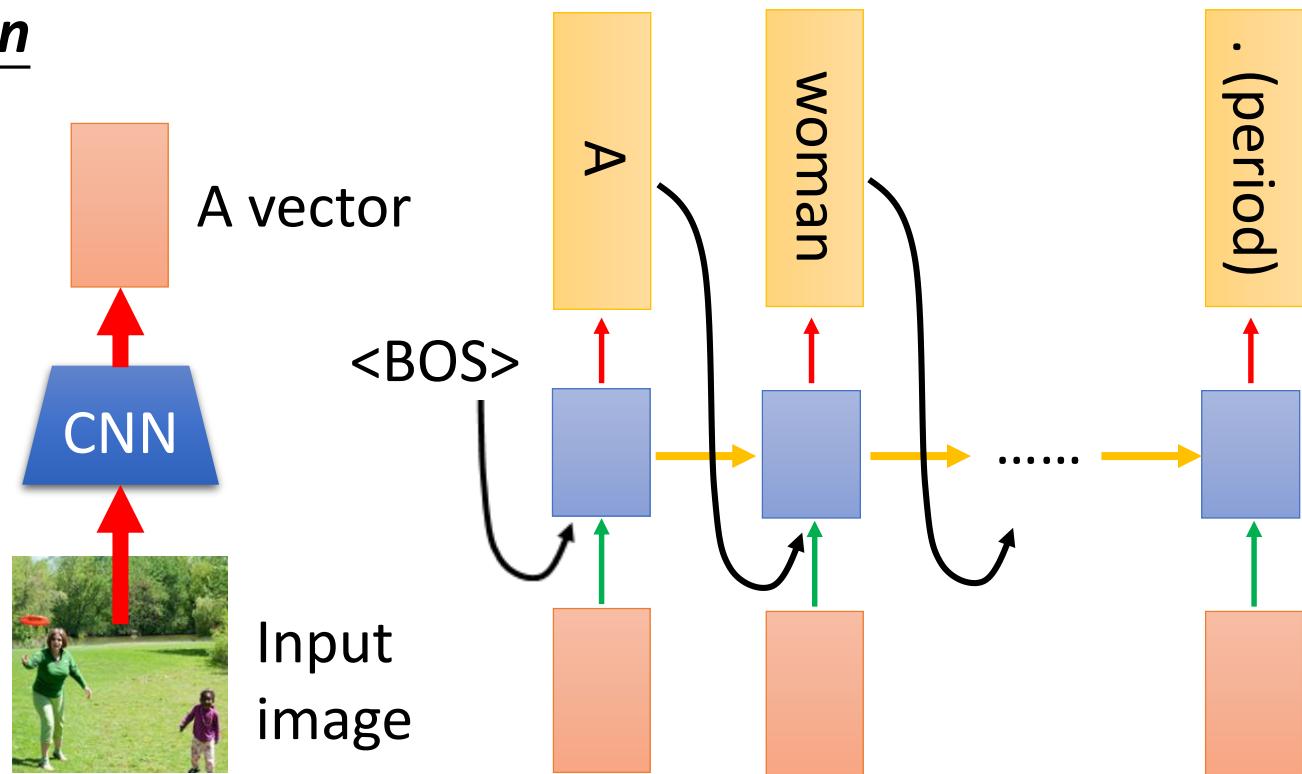
“Hello. Nice
to see you.”



Conditional Generation

- Represent the input condition as a vector, and consider the vector as the input of RNN generator

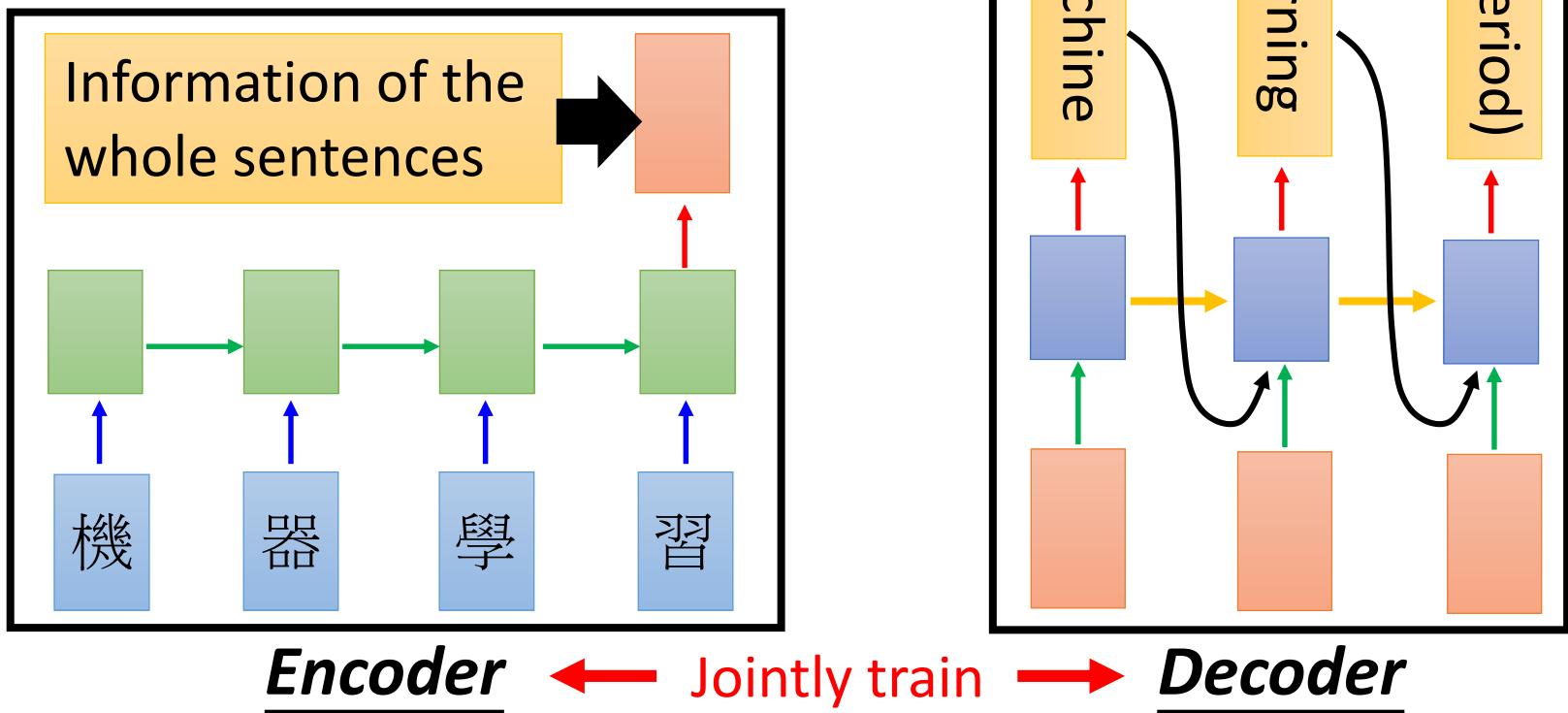
Image Caption Generation



Conditional Generation

Sequence-to-sequence learning

- Represent the input condition as a vector, and consider the vector as the input of RNN generator
- E.g. Machine translation / Chat-bot



Conditional Generation

M: Hello

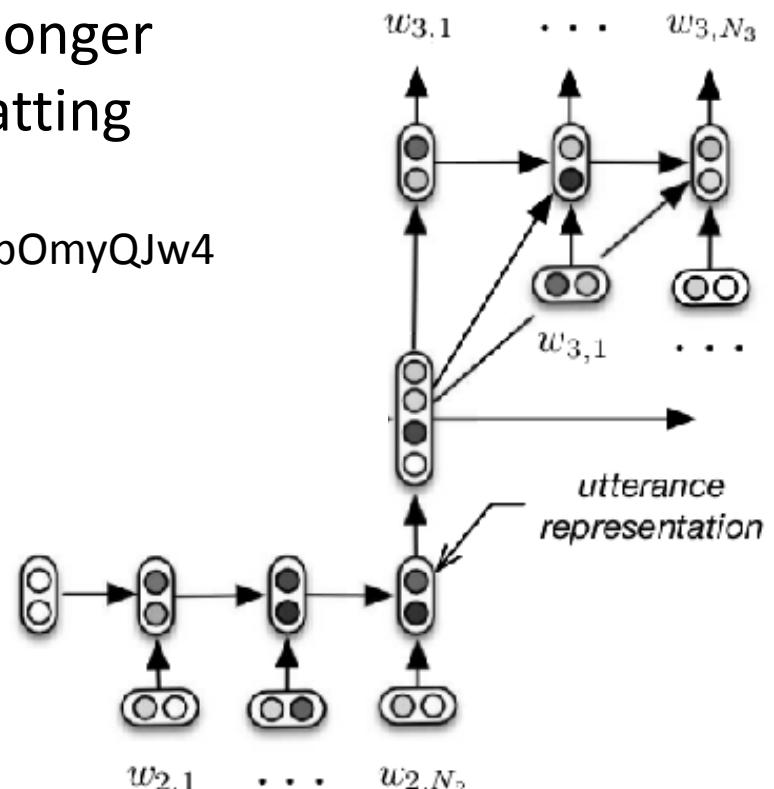
U: Hi

M: Hi

<https://www.youtube.com/watch?v=e2MpOmyQJw4>

Need to consider longer context during chatting

M: Hi 

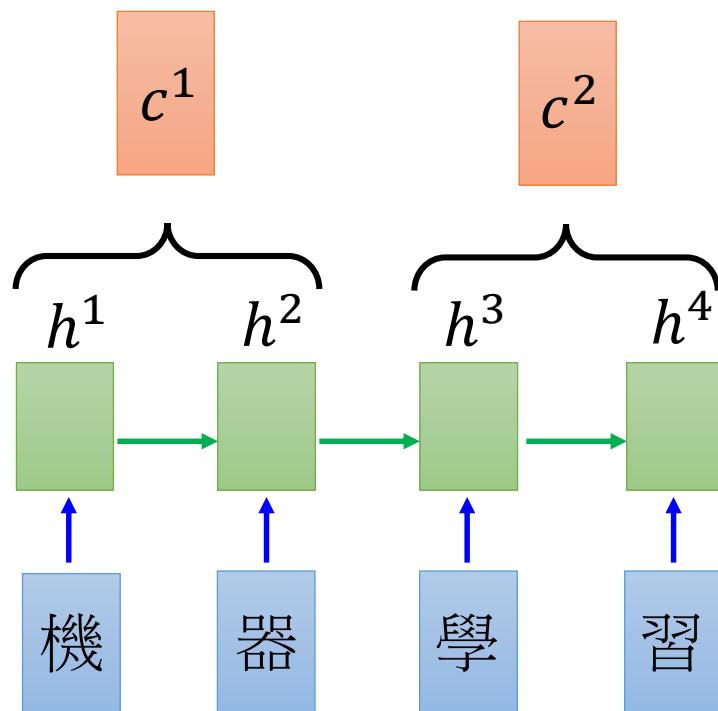


M: Hello

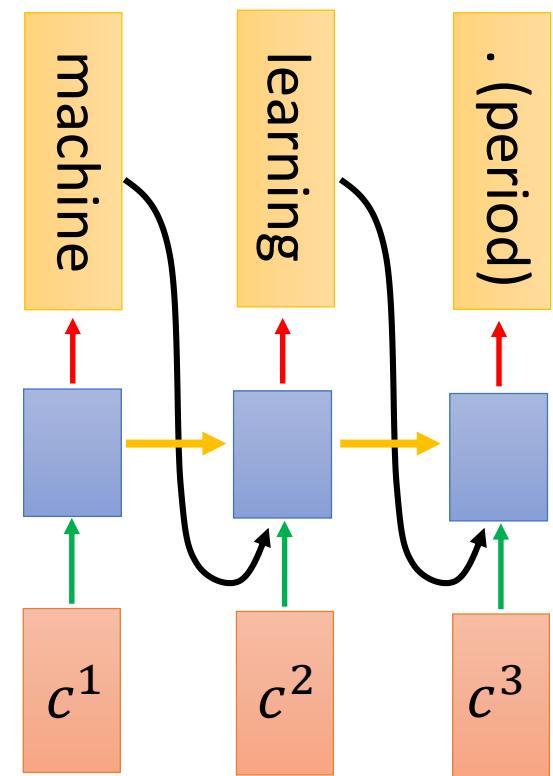
U: Hi

Serban, Iulian V., Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau, 2015
"Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models."

Dynamic Conditional Generation



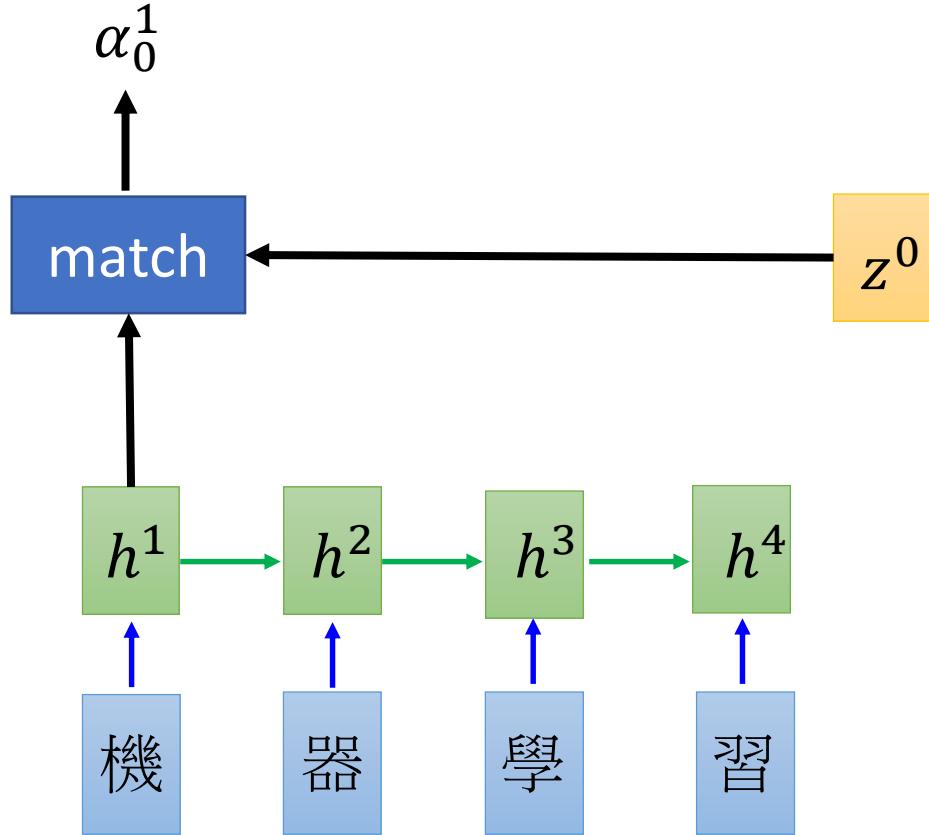
Encoder



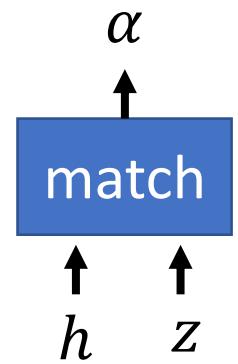
Decoder

Machine Translation

- Attention-based model



Jointly learned
with other part
of the network



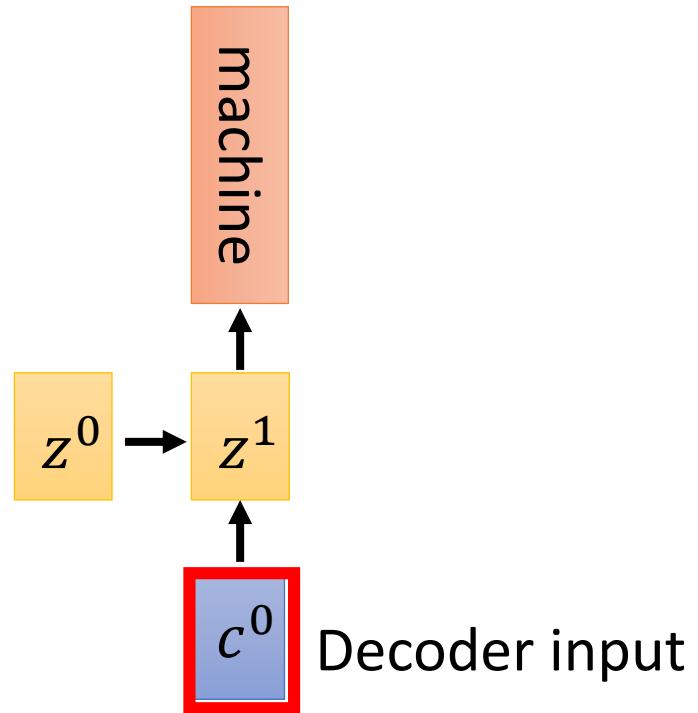
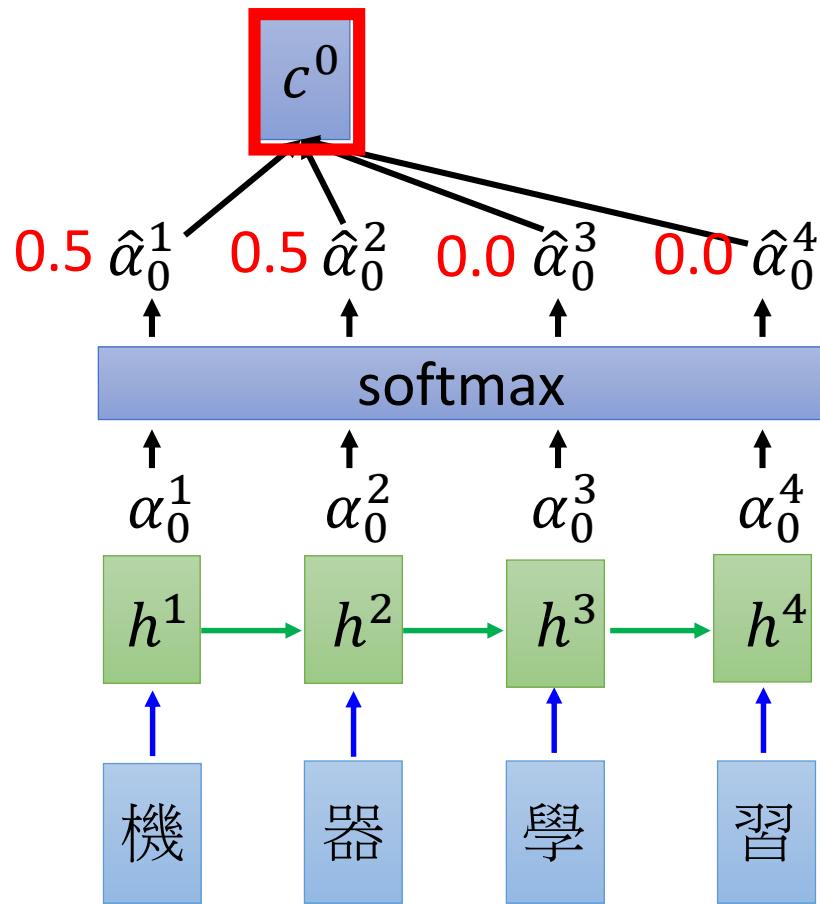
What is **match** ?

Design by yourself

- Cosine similarity of z and h
- Small NN whose input is z and h , output a scalar
- $\alpha = h^T W z$

Machine Translation

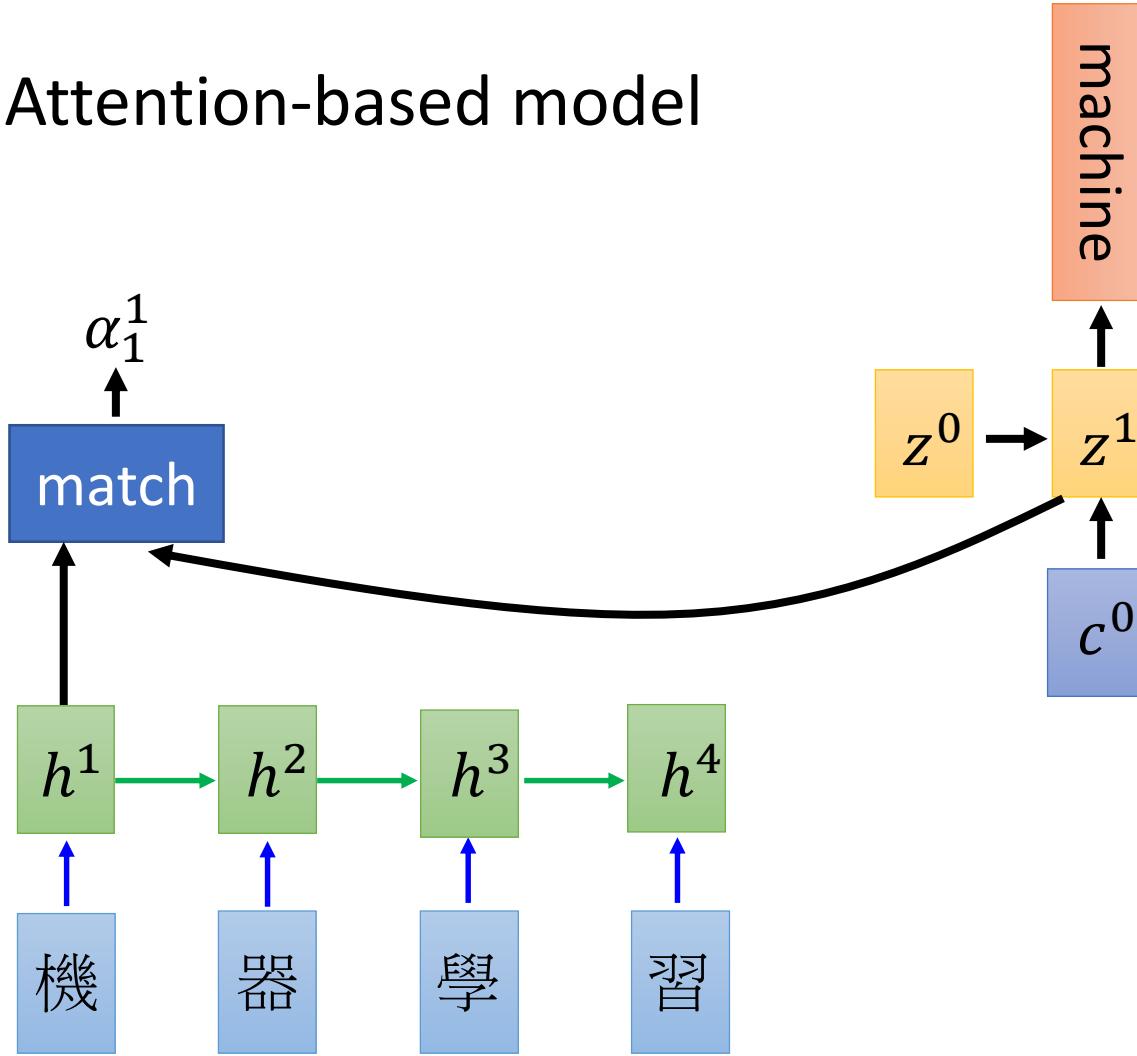
- Attention-based model



$$\begin{aligned}c^0 &= \sum \hat{\alpha}_0^i h^i \\&= 0.5h^1 + 0.5h^2\end{aligned}$$

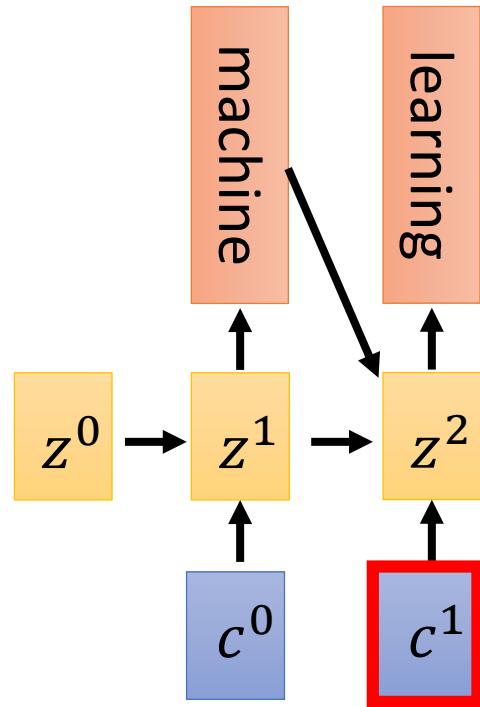
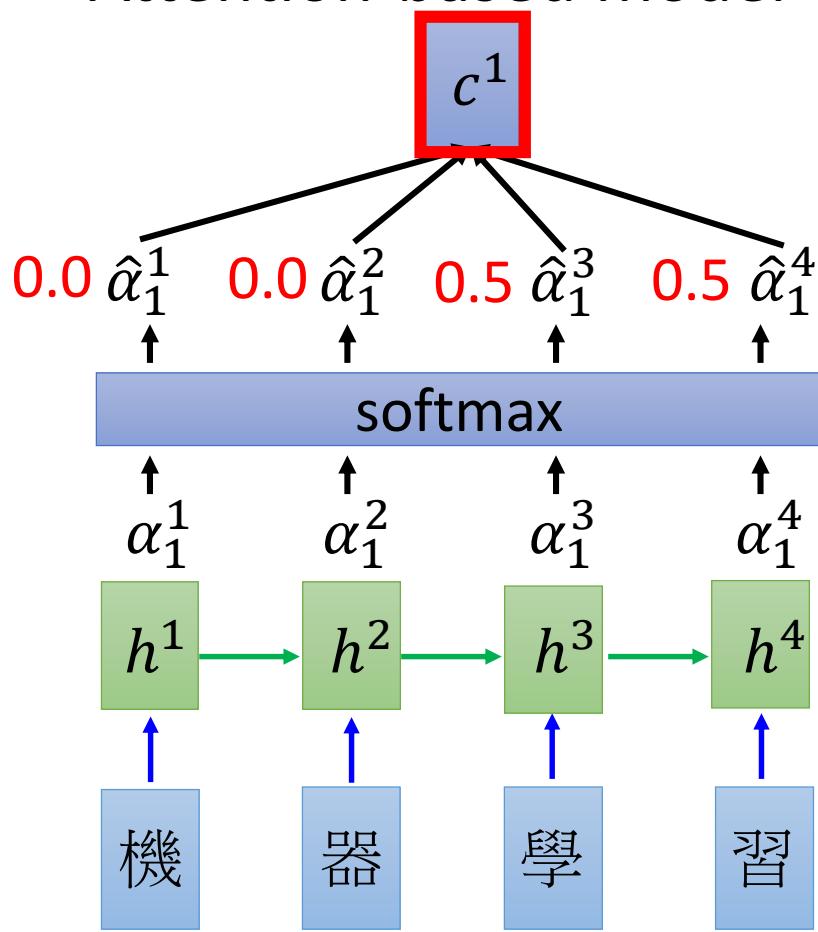
Machine Translation

- Attention-based model



Machine Translation

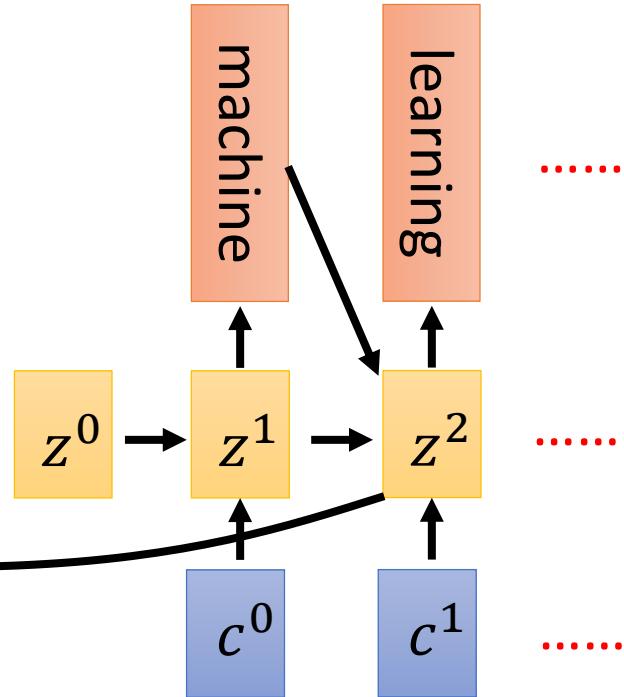
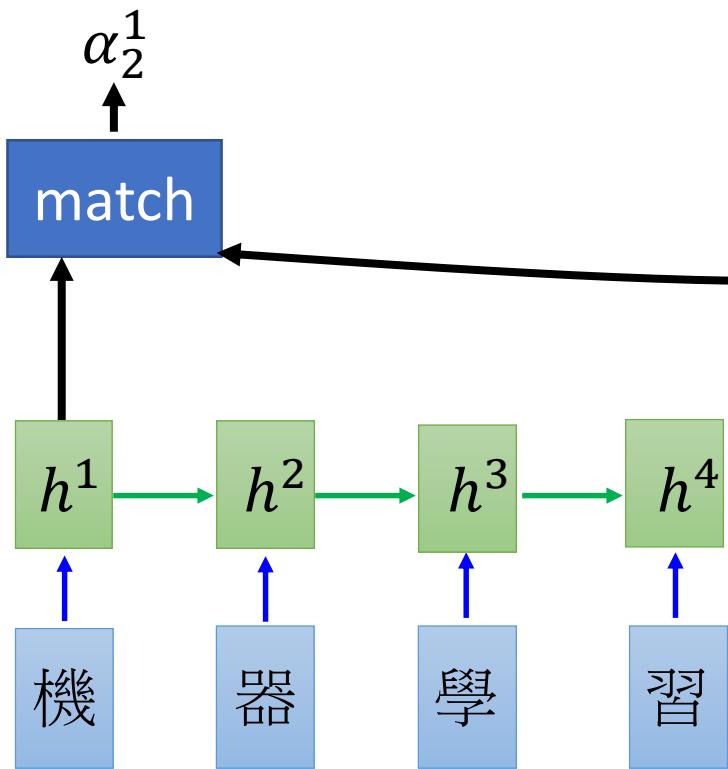
- Attention-based model



$$\begin{aligned} c^1 &= \sum \hat{\alpha}_1^i h^i \\ &= 0.5h^3 + 0.5h^4 \end{aligned}$$

Machine Translation

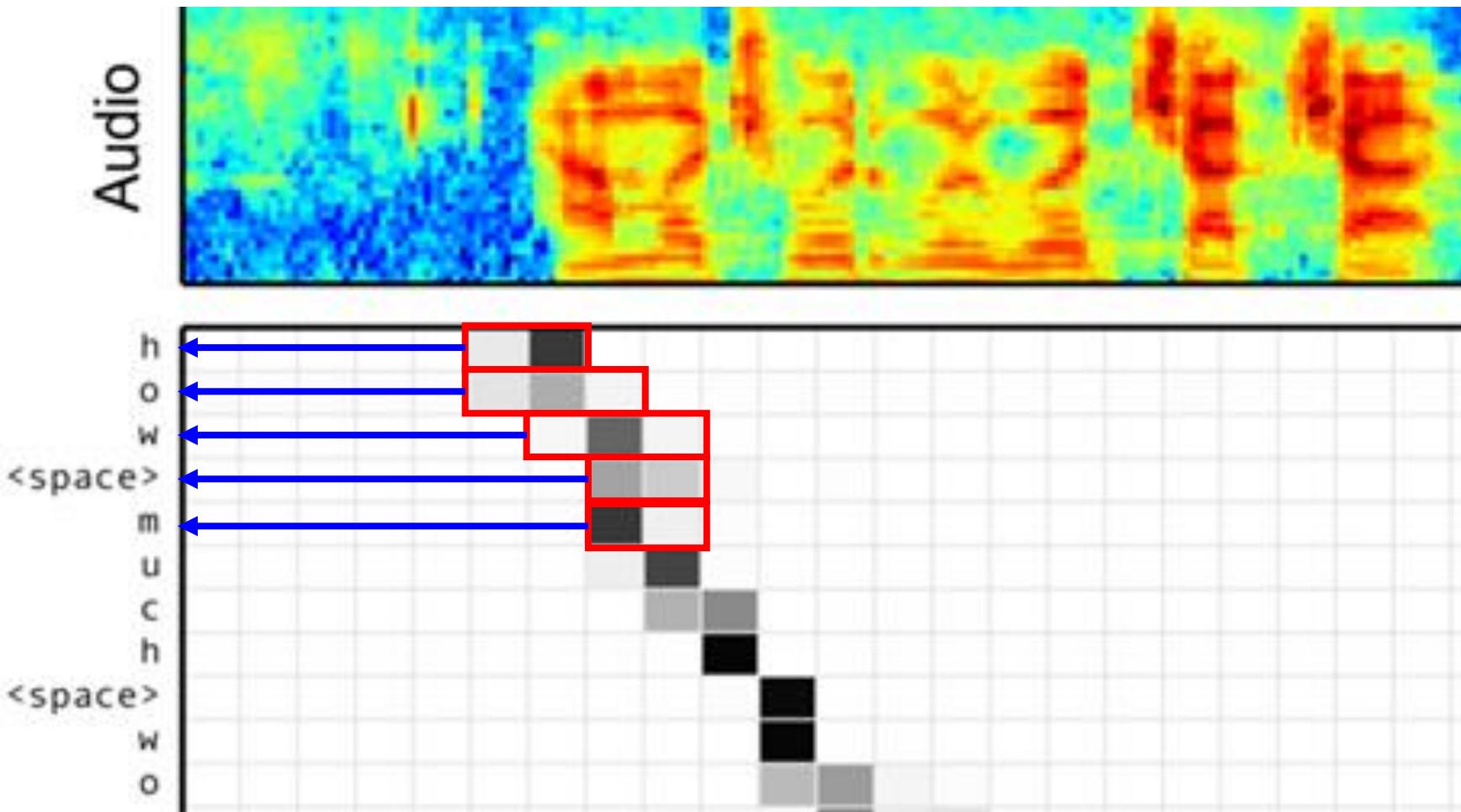
- Attention-based model



The same process repeat
until generating

<EOS>

Speech Recognition



Model	Clean WER	Noisy WER
CLDNN-HMM [22]	8.0	8.9
LAS	14.1	16.5
LAS + LM Rescoring	10.3	12.0

William Chan, Navdeep Jaitly, Quoc V. Le, Oriol Vinyals, “Listen, Attend and Spell”, ICASSP, 2016

Image Caption Generation

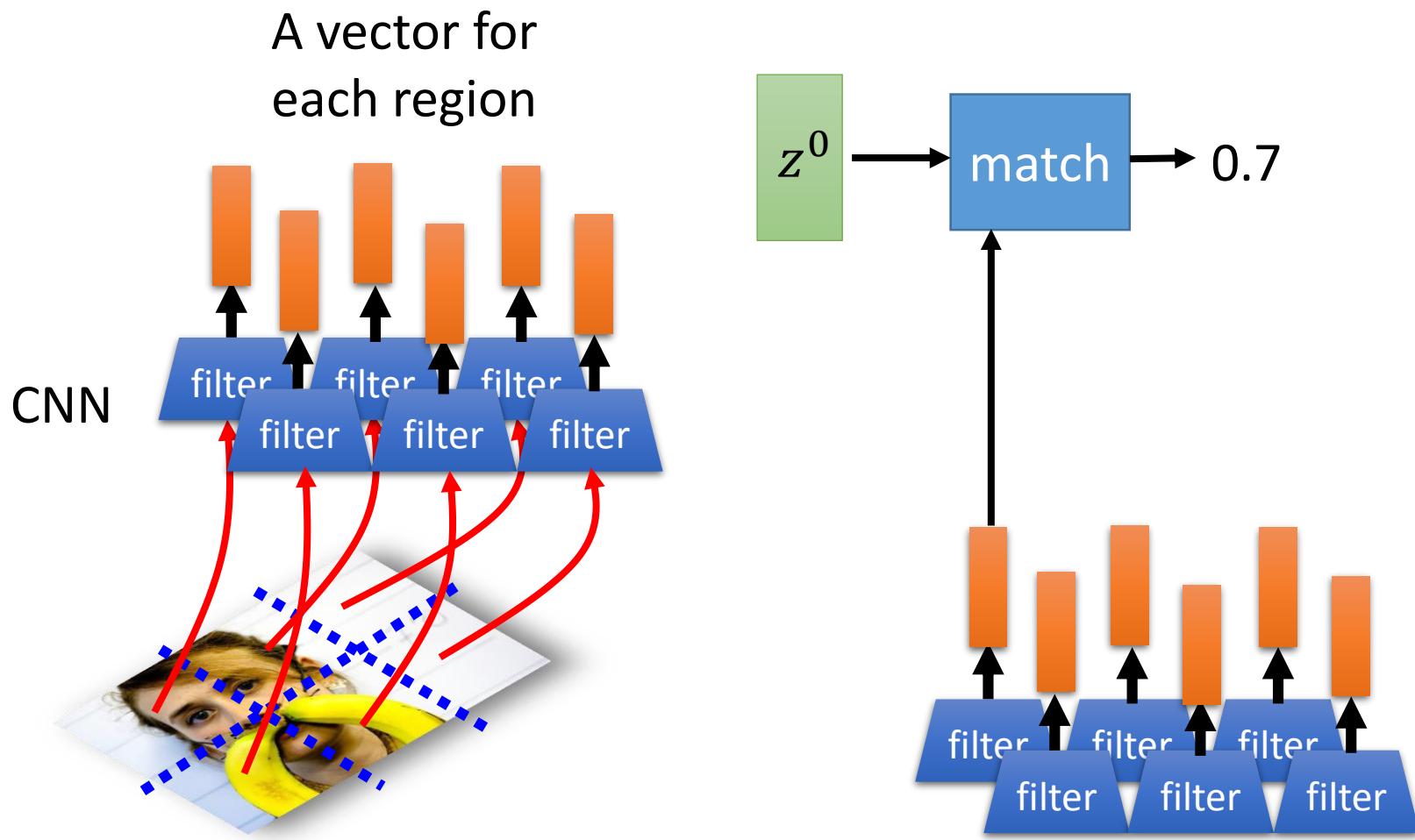


Image Caption Generation

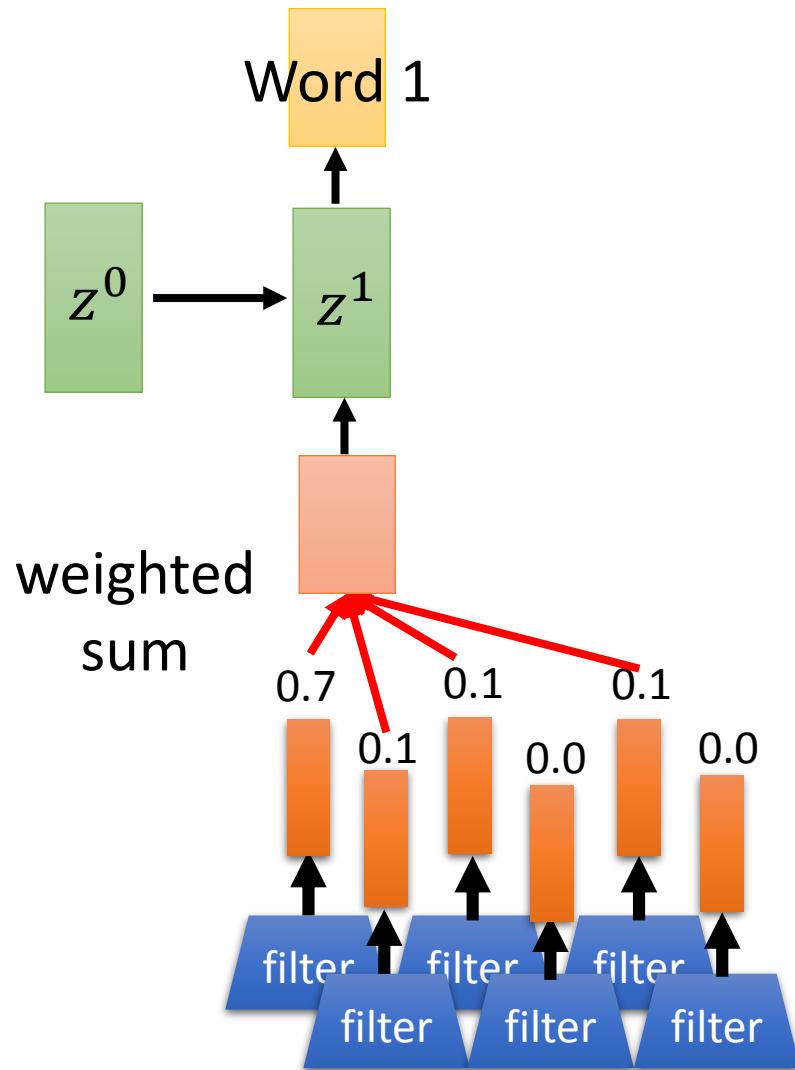
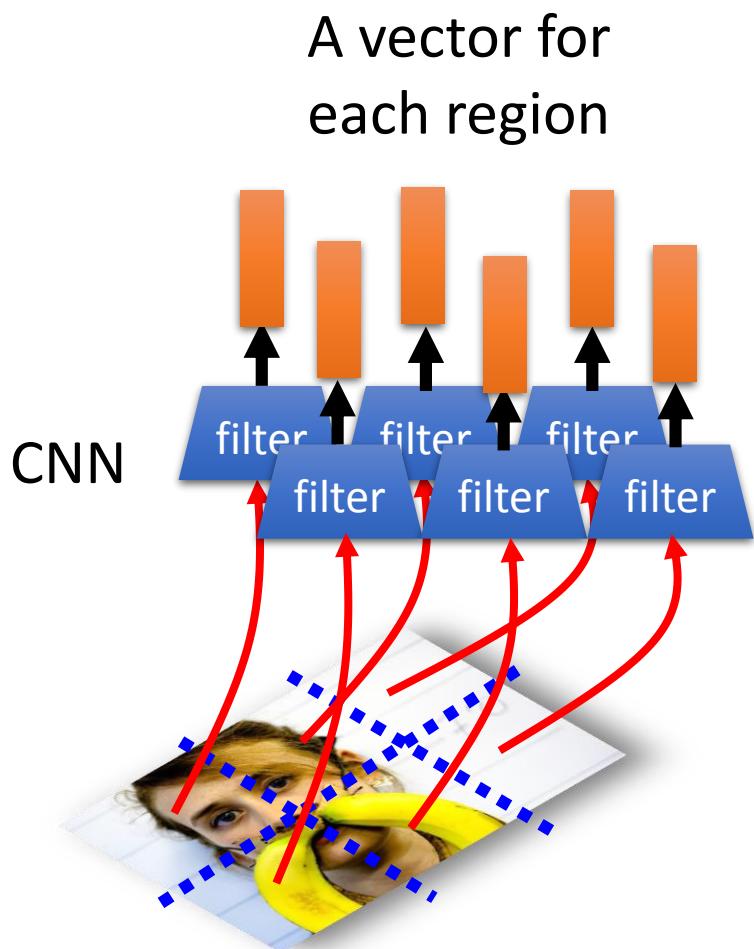


Image Caption Generation

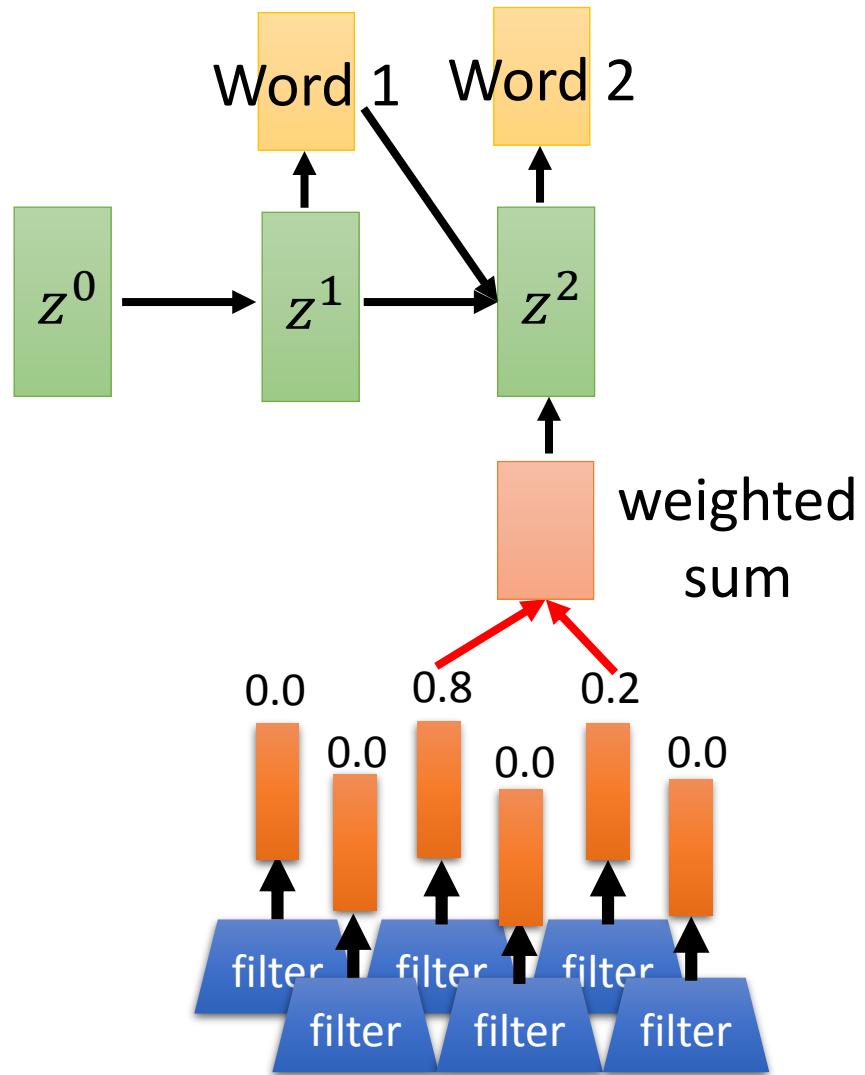
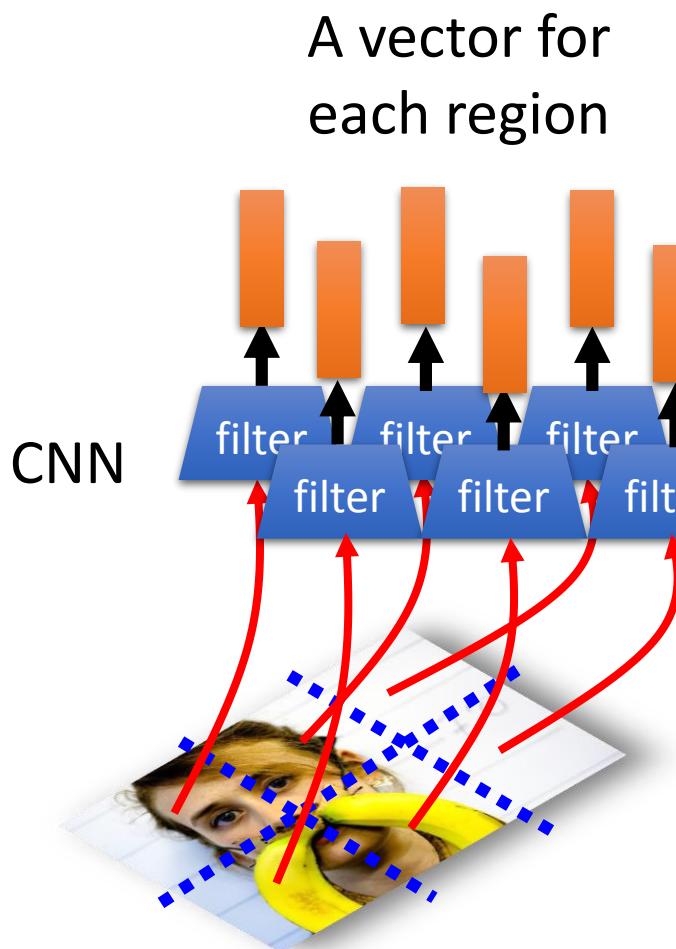


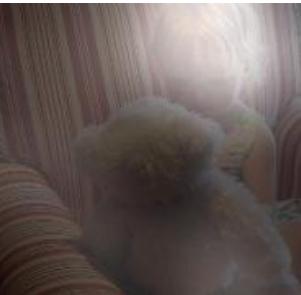
Image Caption Generation



A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

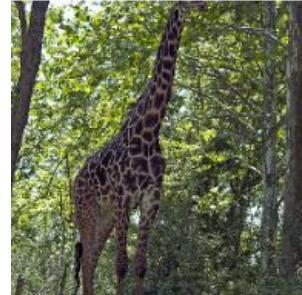
A stop sign is on a road with a mountain in the background.



A little girl sitting on a bed with a teddy bear.



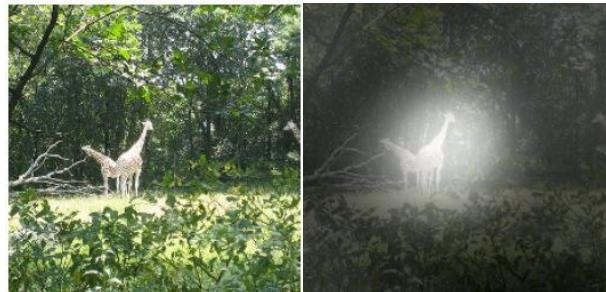
A group of people sitting on a boat in the water.



A giraffe standing in a forest with trees in the background.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, Yoshua Bengio, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML, 2015

Image Caption Generation



A large white bird standing in a forest.



A woman holding a clock in her hand.



A man wearing a hat and a hat on a skateboard.



A person is standing on a beach with a surfboard.



A woman is sitting at a table with a large pizza.



A man is talking on his cell phone while another man watches.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, Yoshua Bengio, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML, 2015



Ref: A man and a woman ride a motorcycle
A **man** and a **woman** are **talking** on the **road**



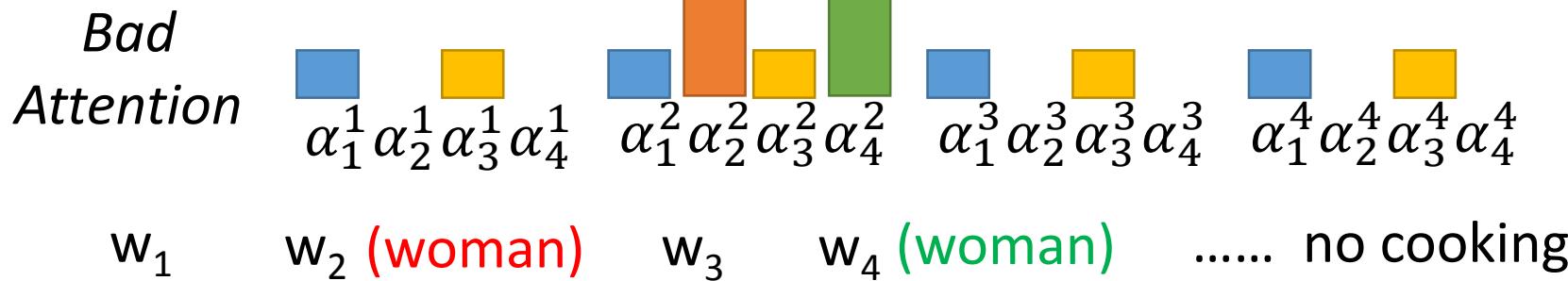
Ref: A woman is frying food
Someone is **frying** a **fish** in a **pot**

Tips for Generation

Attention

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, Yoshua Bengio, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML, 2015

component
 α_t^i → time



Good Attention: each input component has approximately the same attention weight

E.g. Regularization term: $\sum_i \left(\tau - \sum_t \alpha_t^i \right)^2$

For each component Over the generation

Mismatch between Train and Test

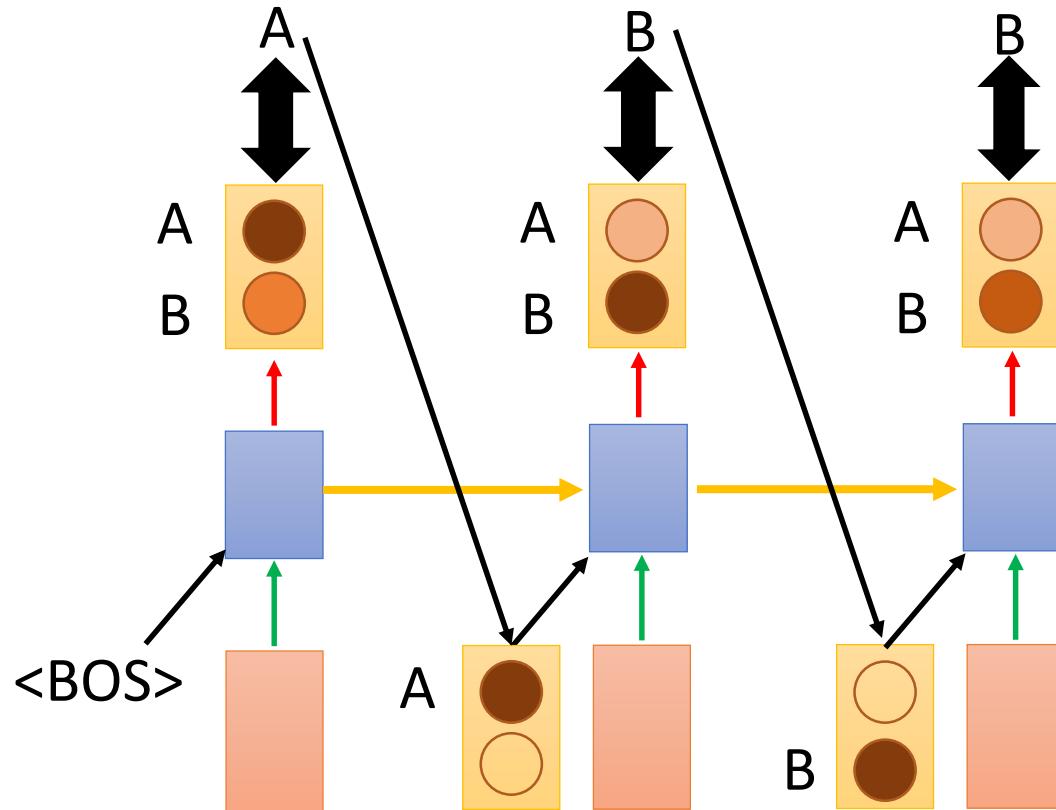
- Training

$$C = \sum_t C_t$$

Minimizing
cross-entropy of
each component

 : condition

Reference:



Mismatch between Train and Test

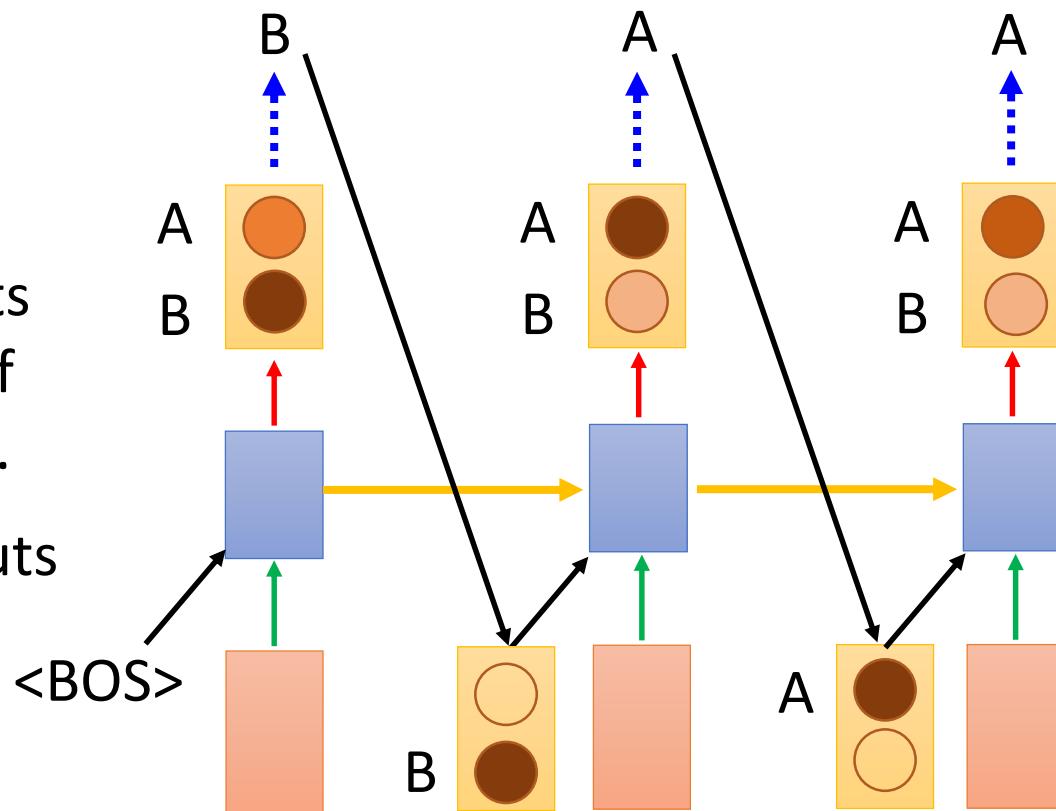
- *Generation*

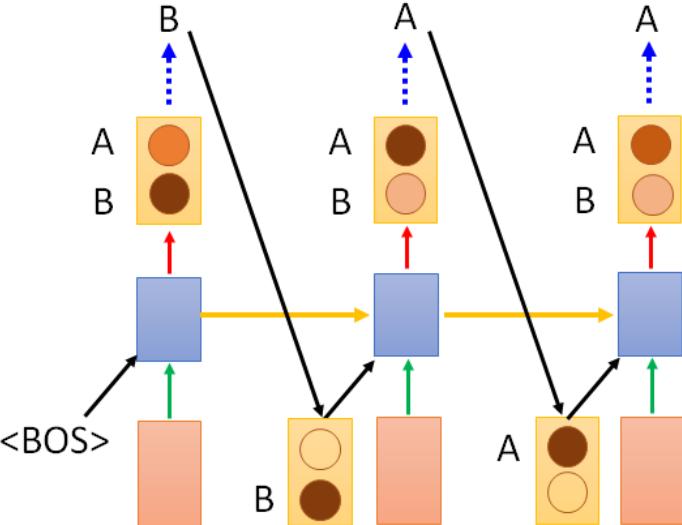
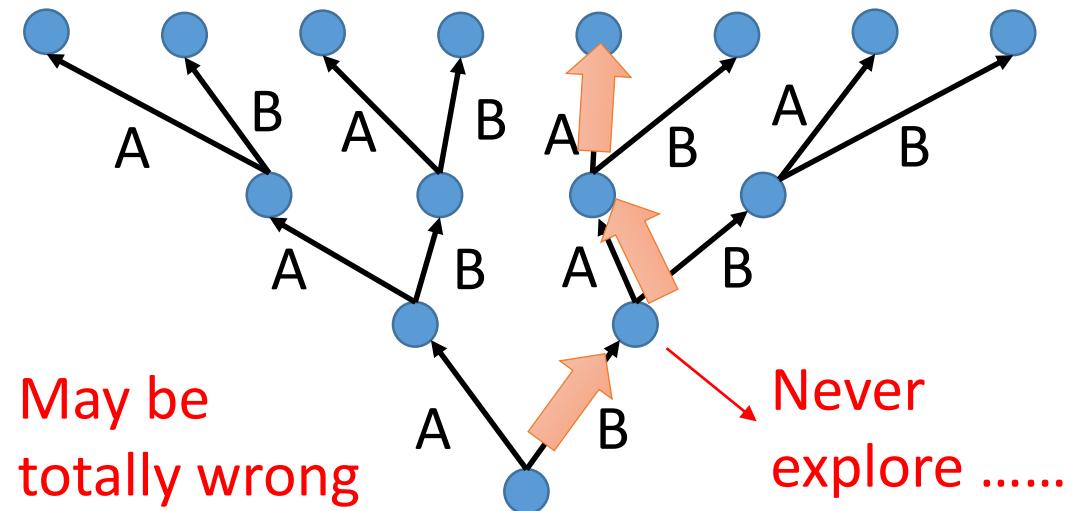
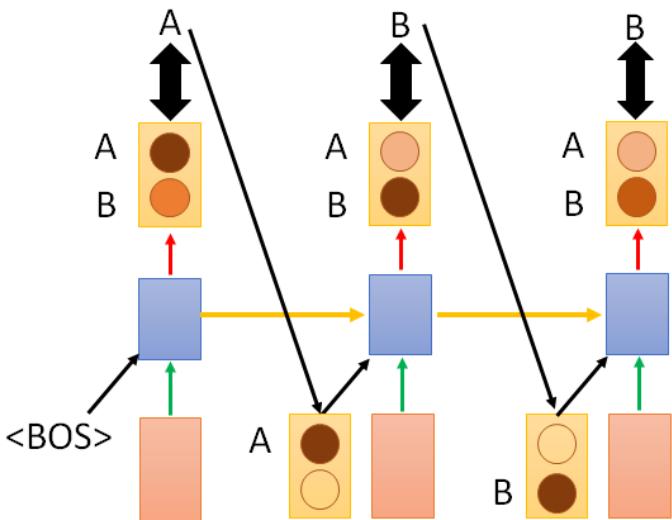
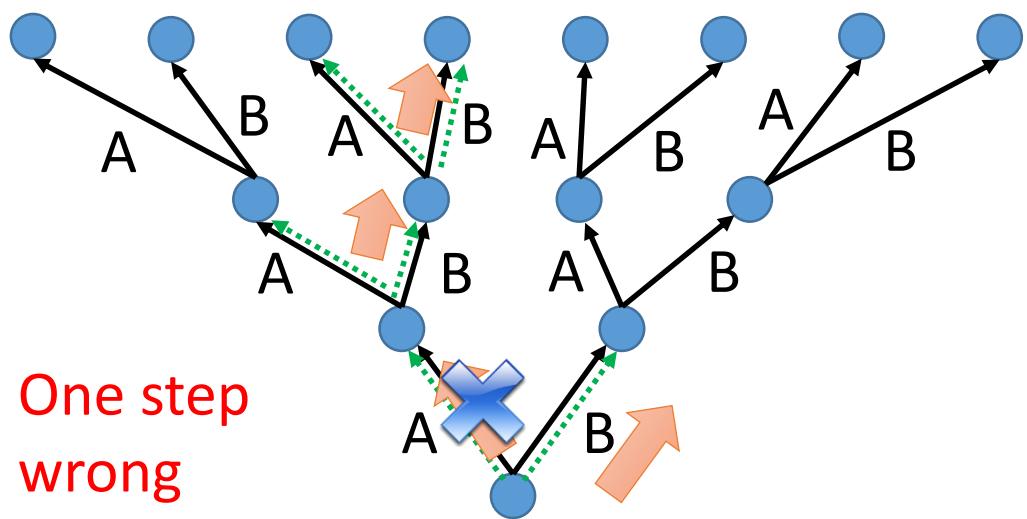
We do not know
the reference

Testing: The inputs
are the outputs of
the last time step.

Training: The inputs
are reference.

Exposure Bias





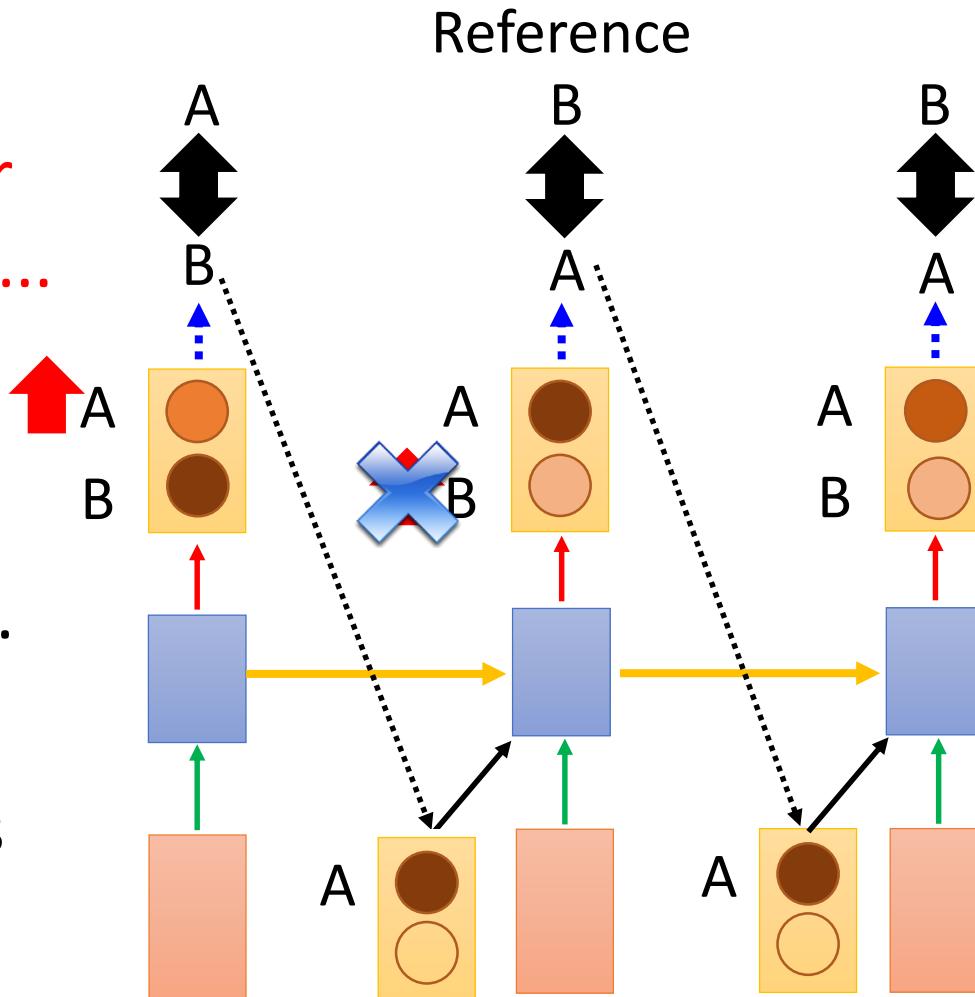
一步錯，步步錯

Modifying Training Process?

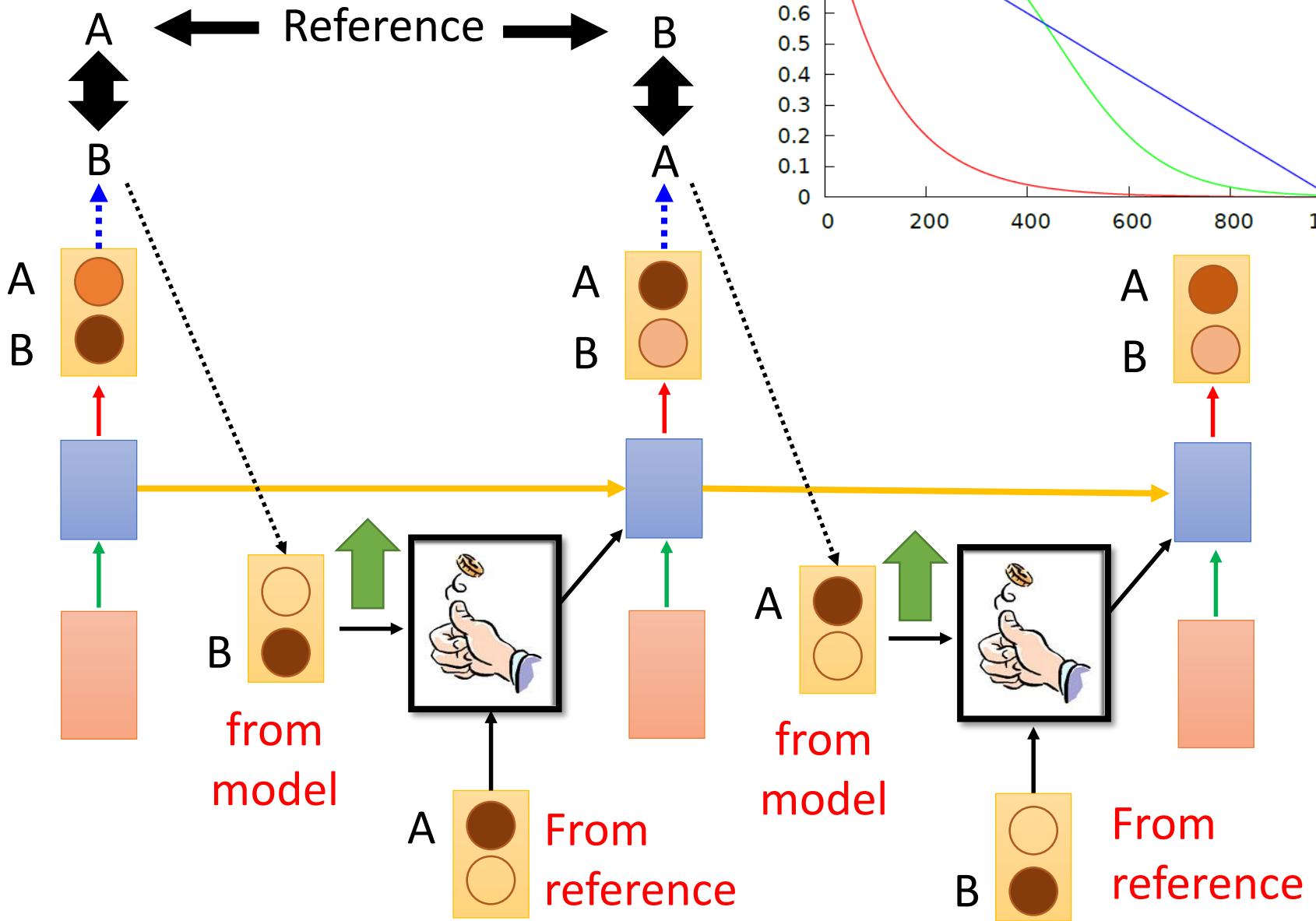
When we try to
decrease the loss for
both steps 1 and 2

Training is
matched to testing.

In practice, it is
hard to train in this
way.



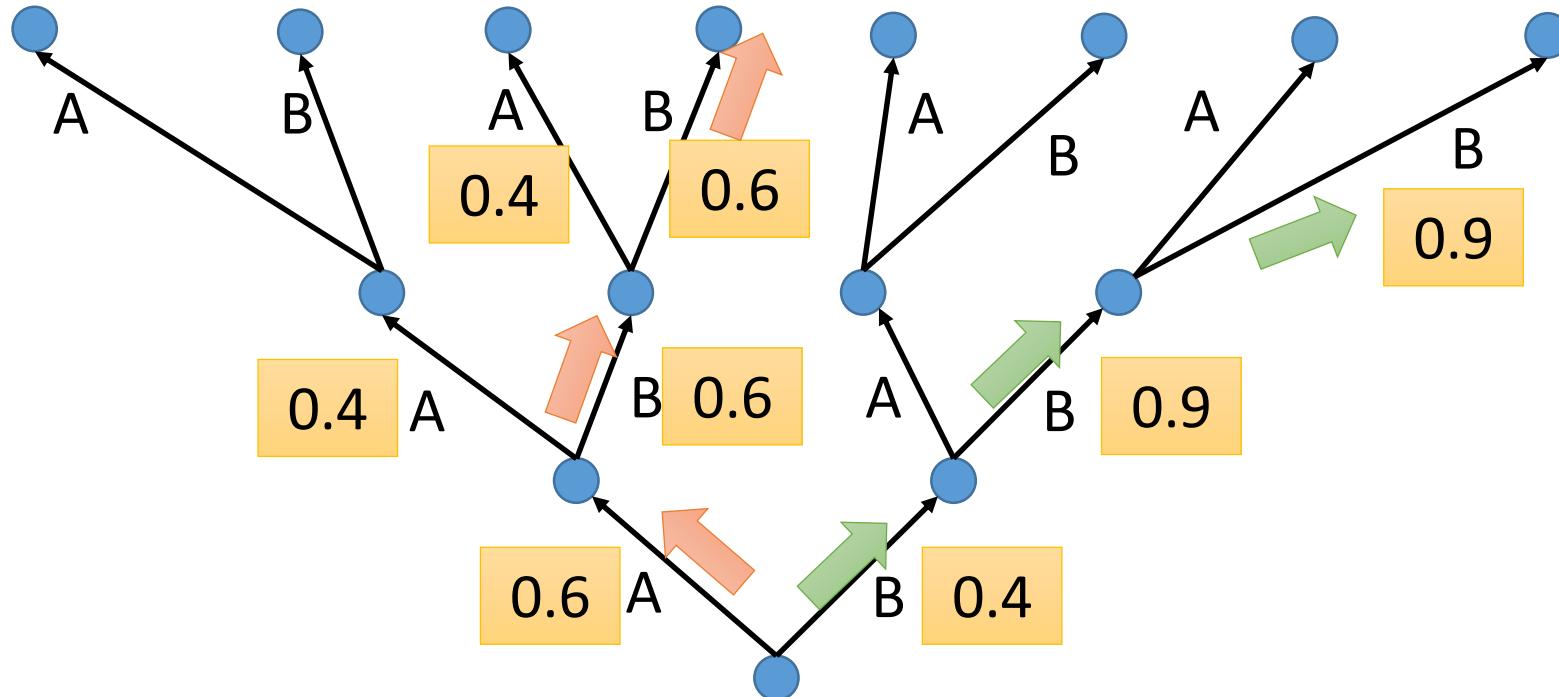
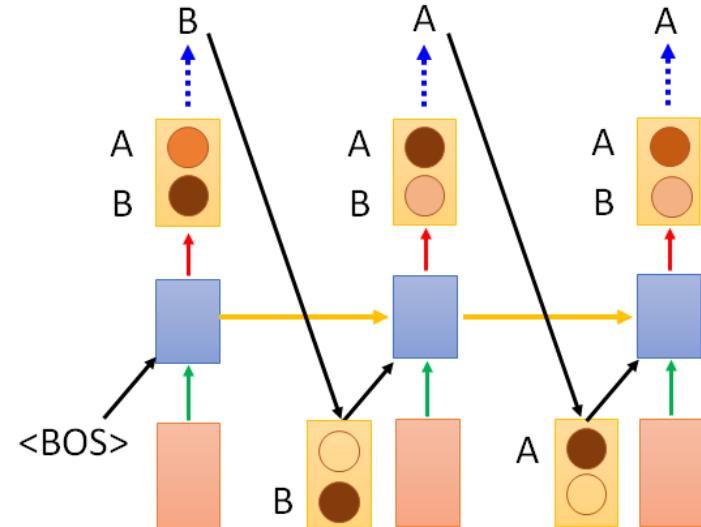
Scheduled Sampling



Beam Search

The green path has higher score.

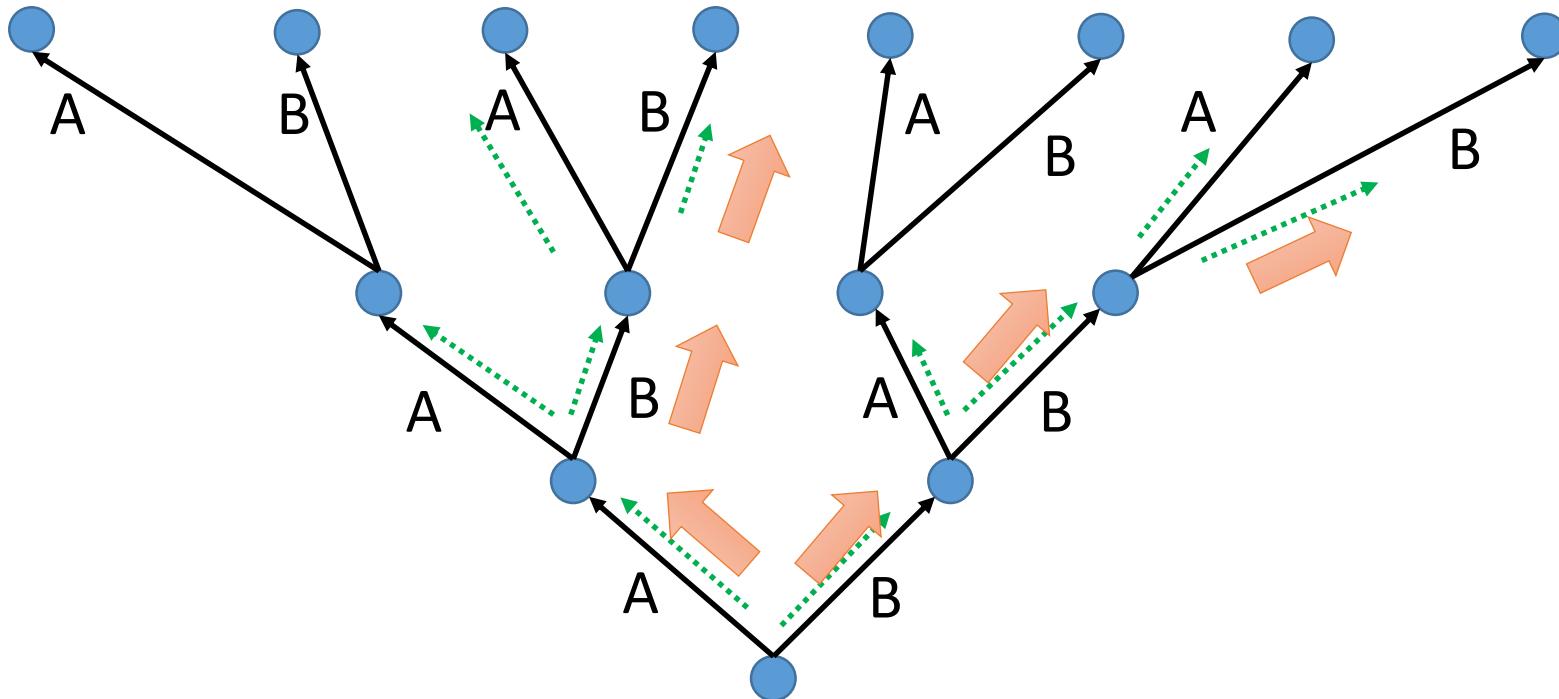
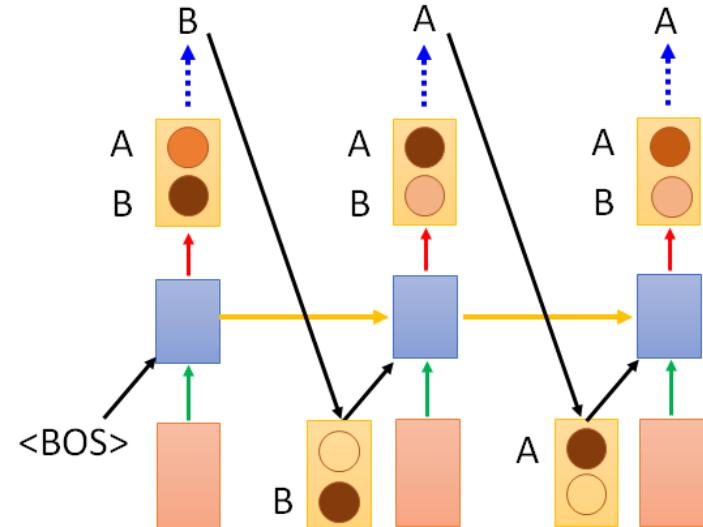
Not possible to check all the paths



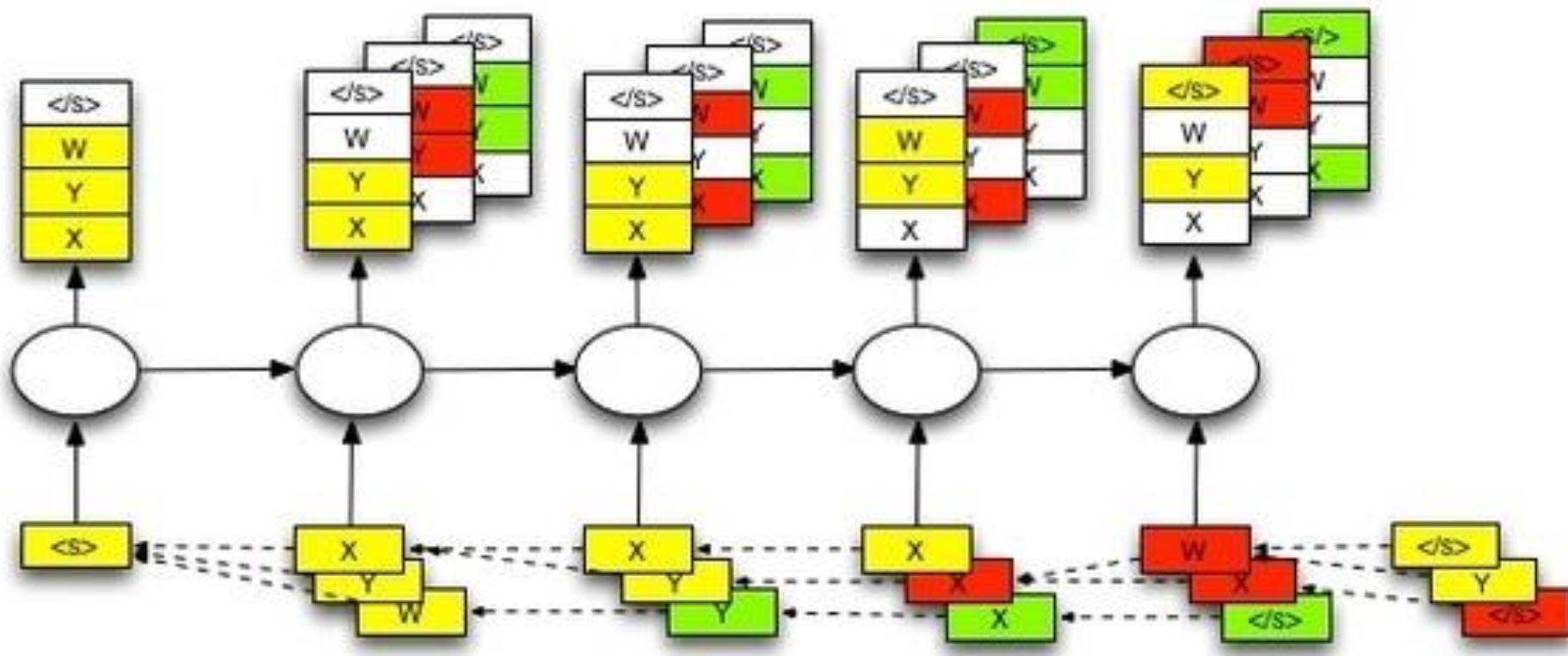
Beam Search

Keep several best path at each step

Beam size = 2



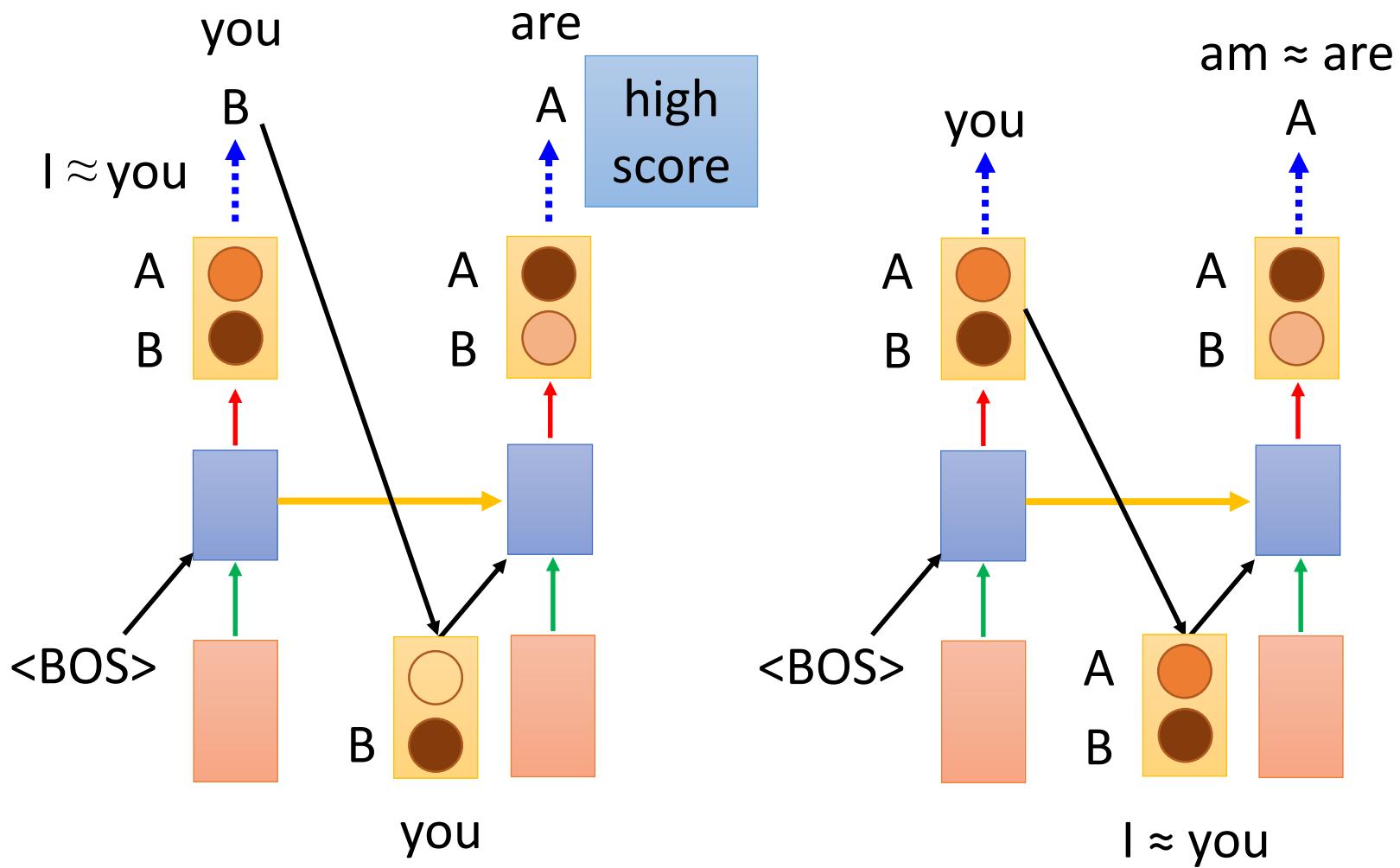
Beam Search



The size of beam is 3 in this example.

Better Idea?

I am ✓
You are ✓
I are ✗
You am ✗



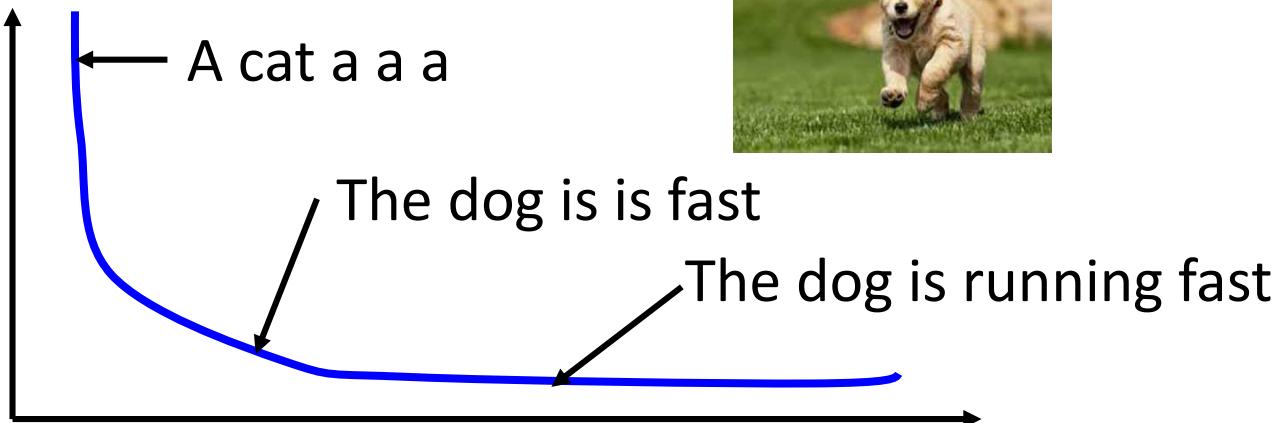
Object level v.s. Component level

- Minimizing the error defined on component level is not equivalent to improving the generated objects

Ref: The dog is running fast

$$C = \sum_t C_t$$

Cross-entropy
of each step



Optimize object-level criterion instead of component-level cross-entropy. object-level criterion: $R(y, \hat{y})$ Gradient Descent?

y : generated utterance, \hat{y} : ground truth

Reinforcement learning?

Start with
observation s_1

Observation s_2

Observation s_3



Obtain reward
 $r_1 = 0$

Obtain reward
 $r_2 = 5$



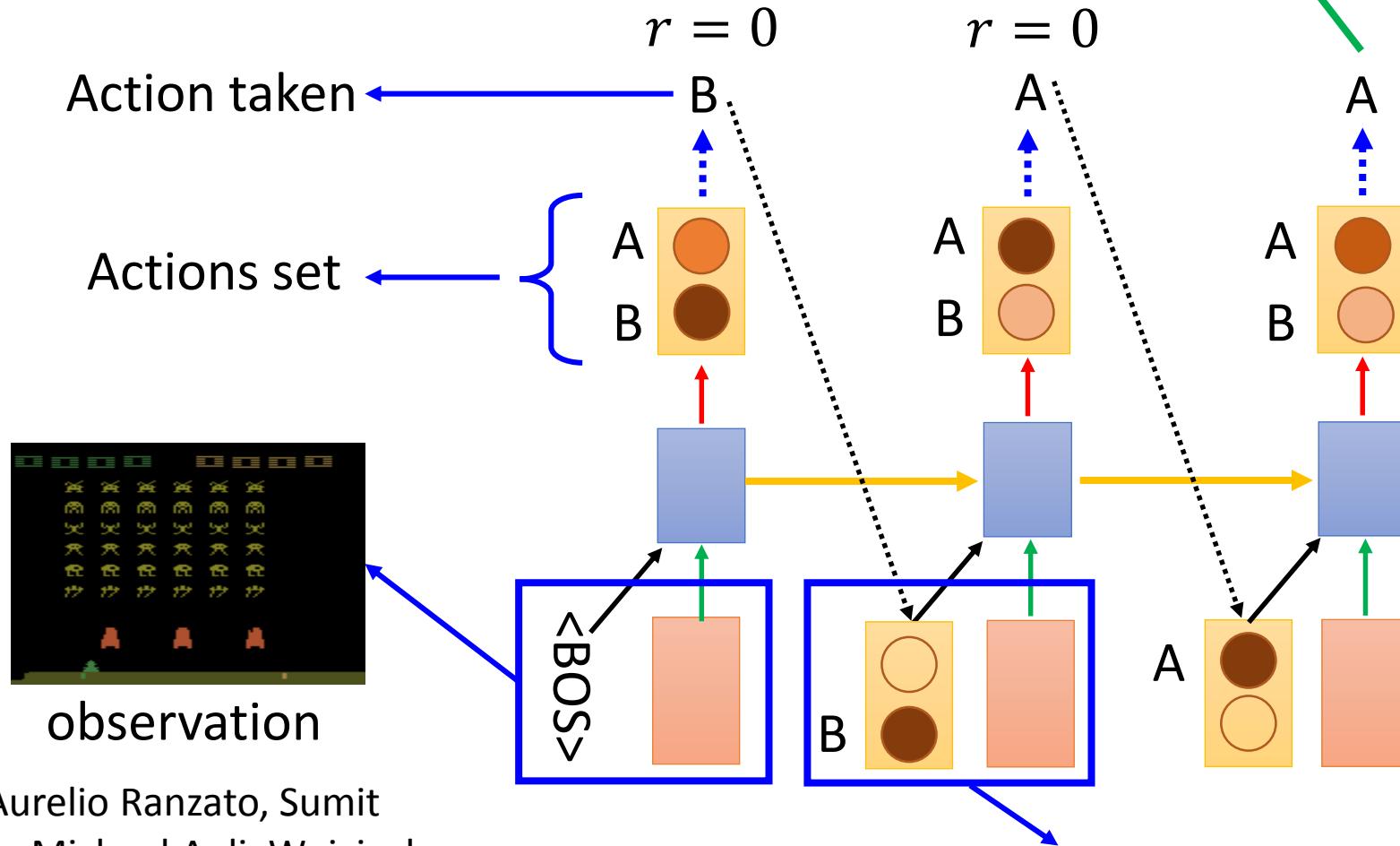
Action a_1 : "right"



Action a_2 : "fire"
(kill an alien)

Reinforcement learning?

reward:
 $R("BAA", \text{reference})$

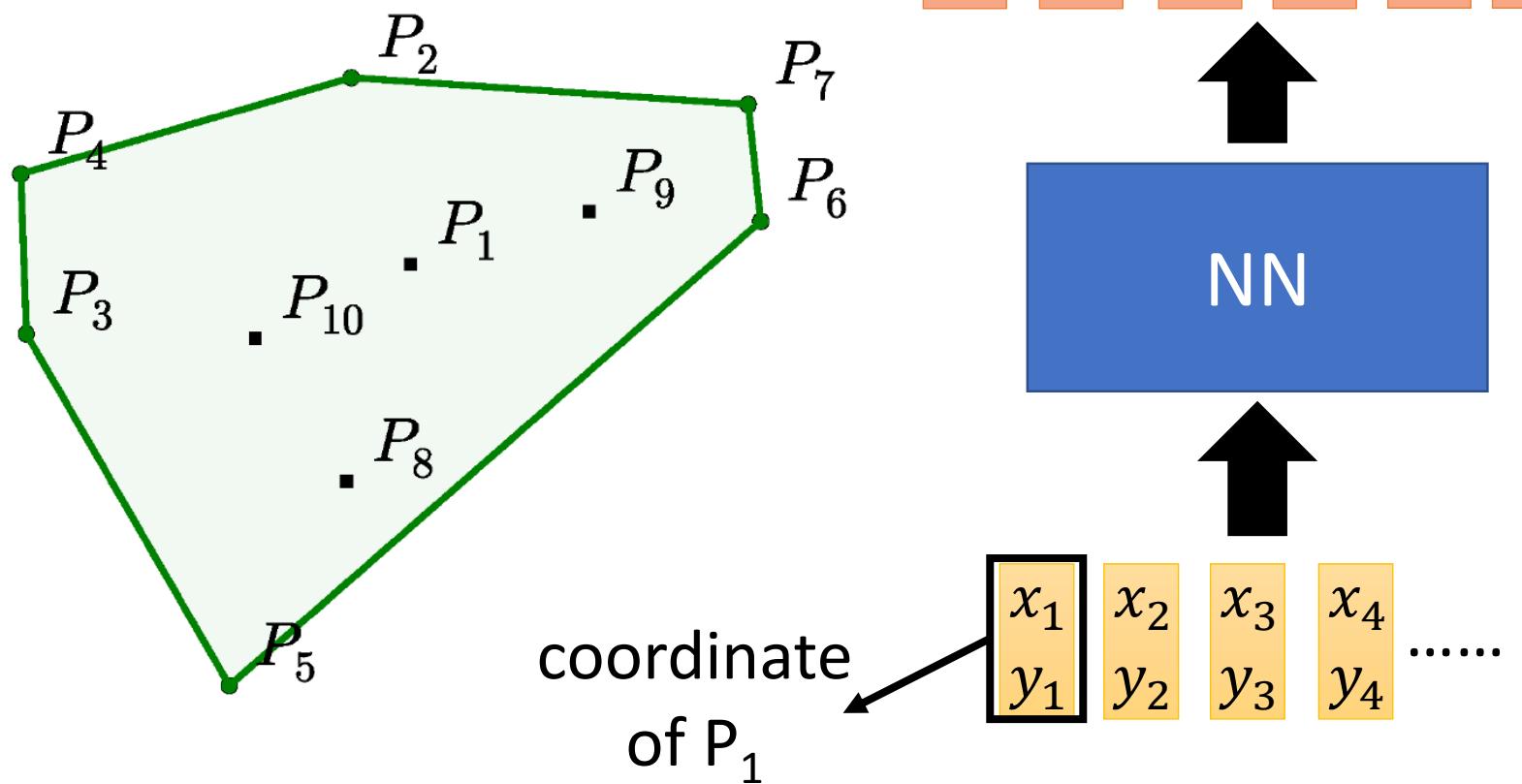


Marc'Aurelio Ranzato, Sumit
Chopra, Michael Auli, Wojciech
Zaremba, "Sequence Level Training with
Recurrent Neural Networks", ICLR, 2016

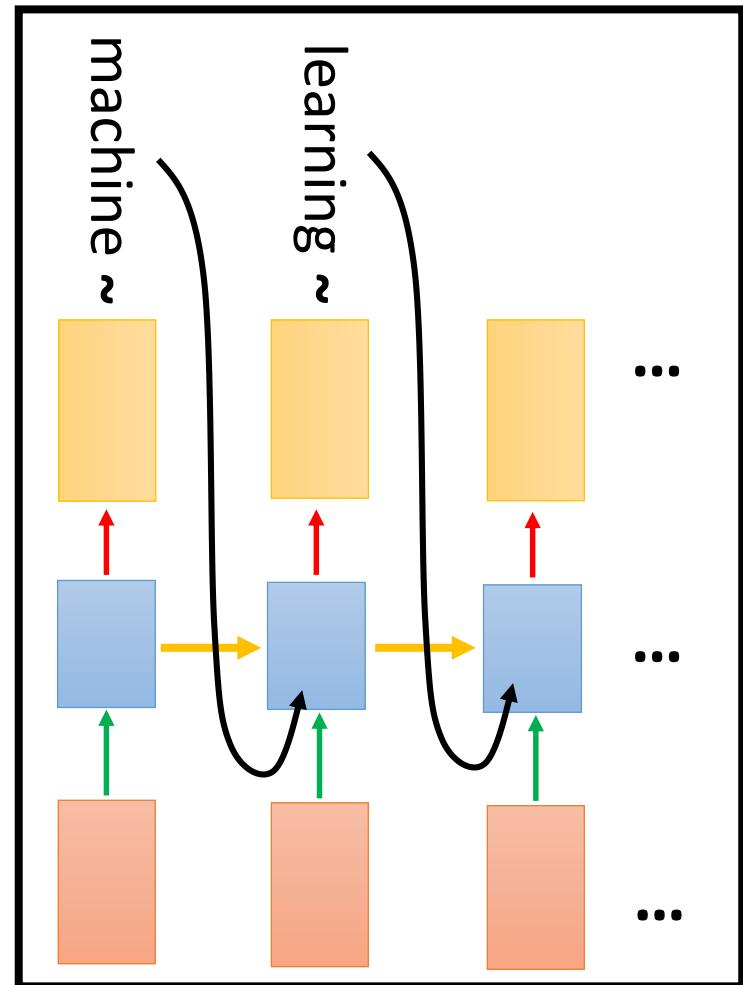
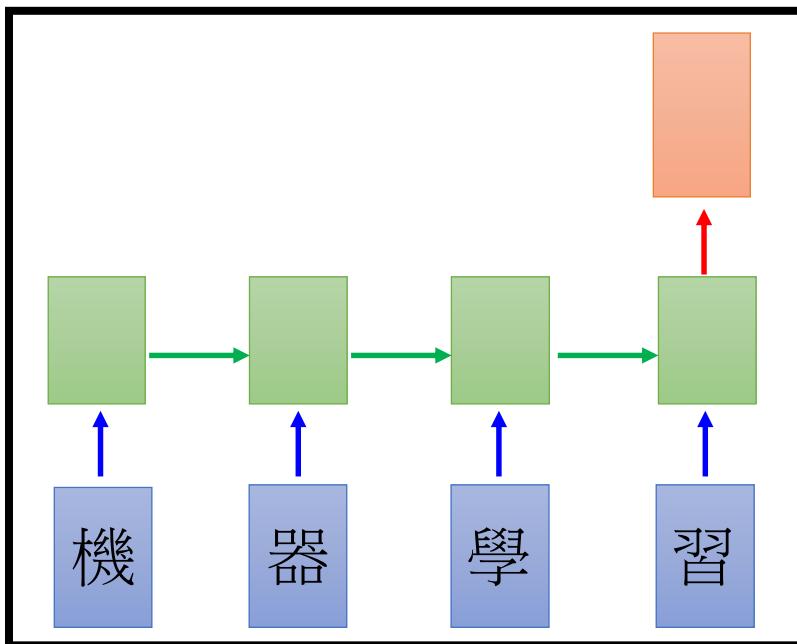
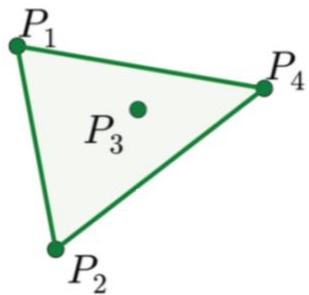
The action we take influence
the observation in the next step

Pointer Network

Pointer Network

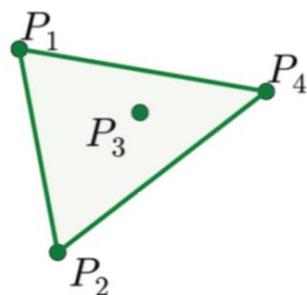


Sequence-to-sequence?



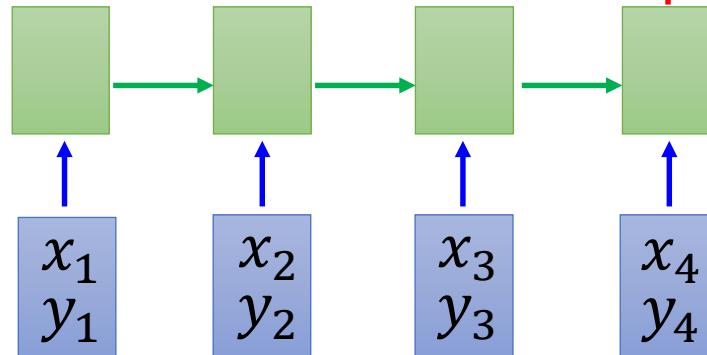
Problem?

Sequence-to-sequence?

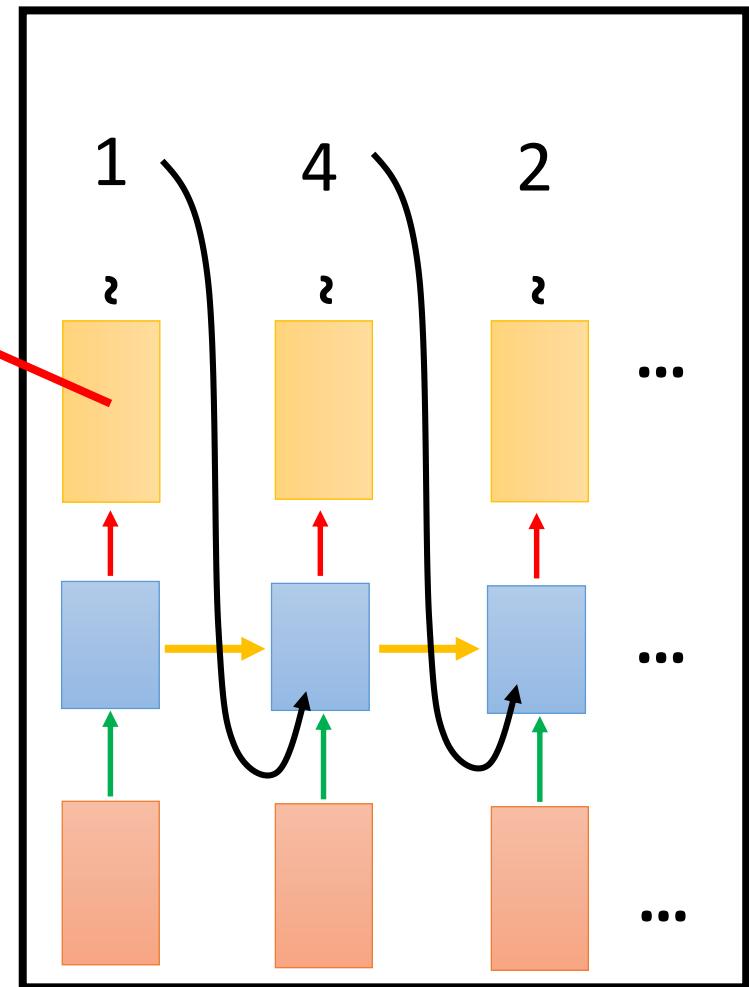


{1, 2, 3, 4, END}

Of course, one can
add attention.



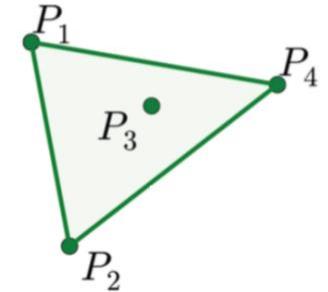
Encoder



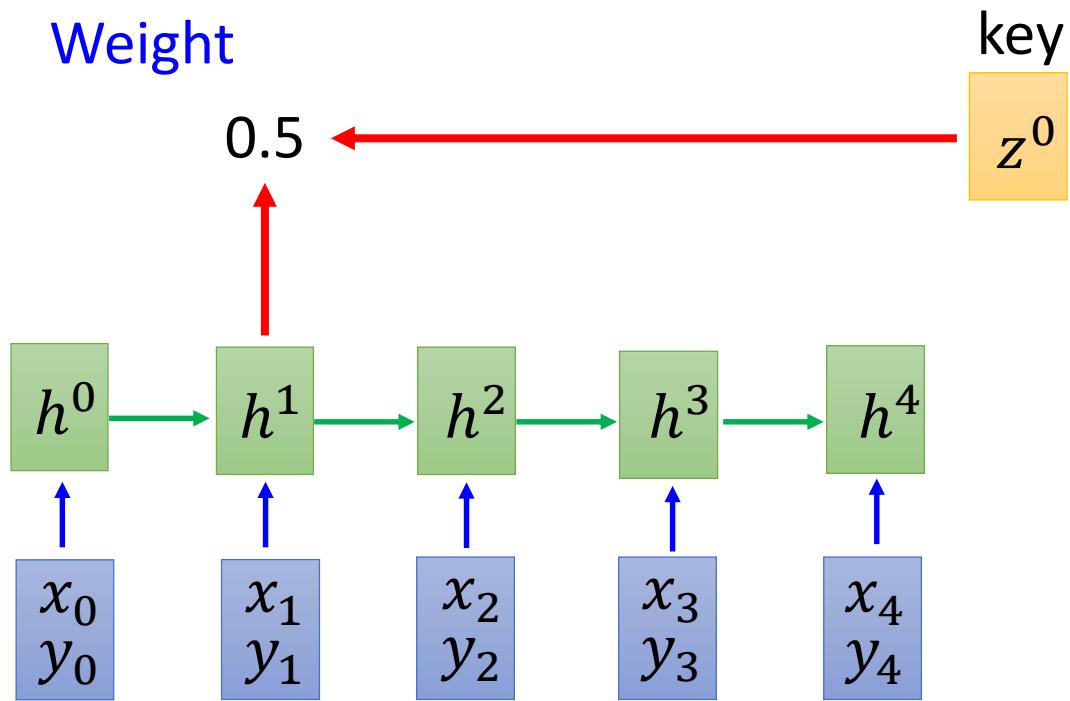
Decoder

Pointer Network

x_0
 y_0 : END

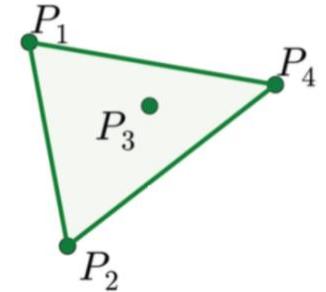


Attention
Weight



Pointer Network

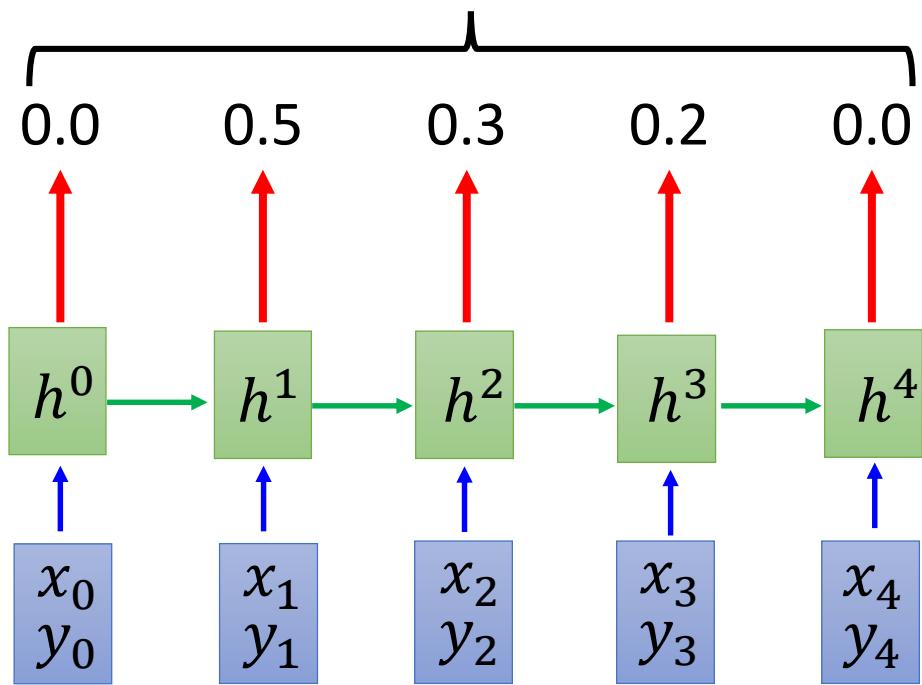
x_0
 y_0 : END



Output: 1

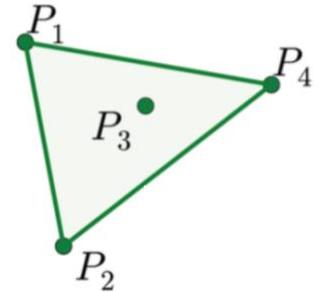
?

argmax from this distribution



Pointer Network

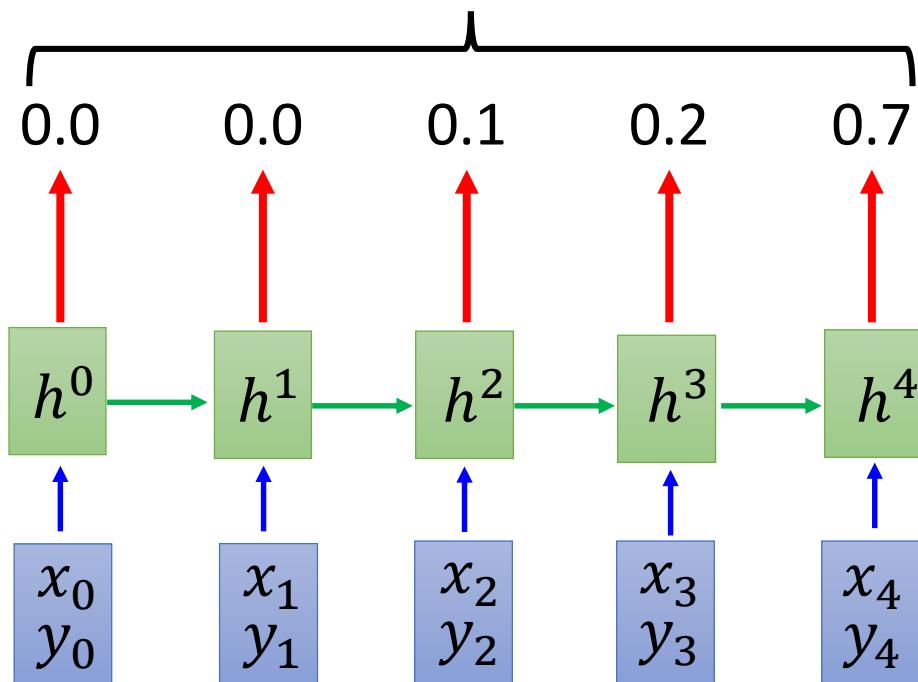
x_0
 y_0 : END



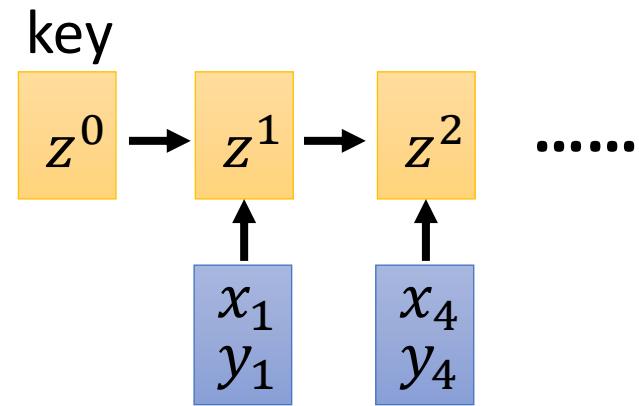
Output: 4

?

argmax from this distribution

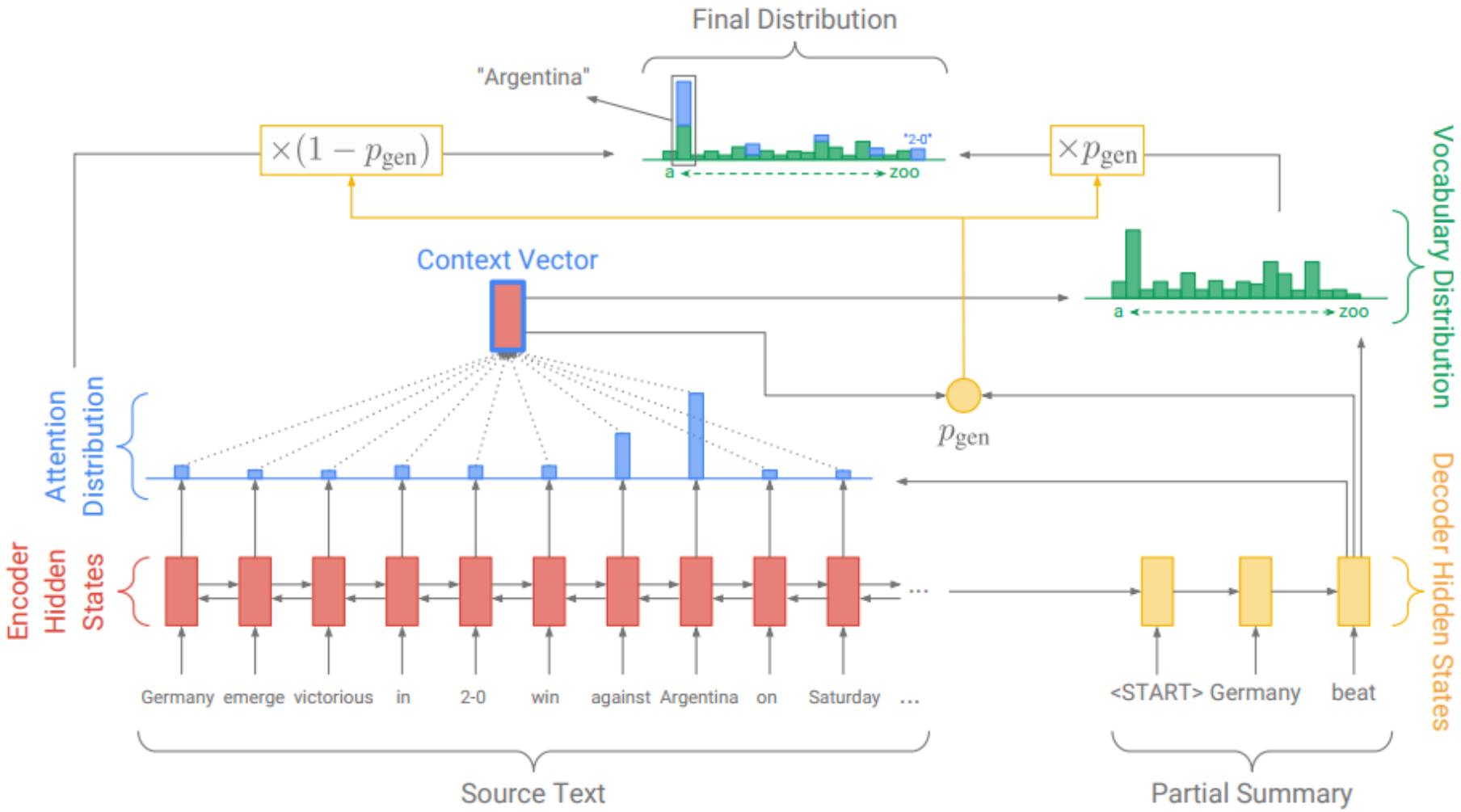


What decoder can output depends on the input.



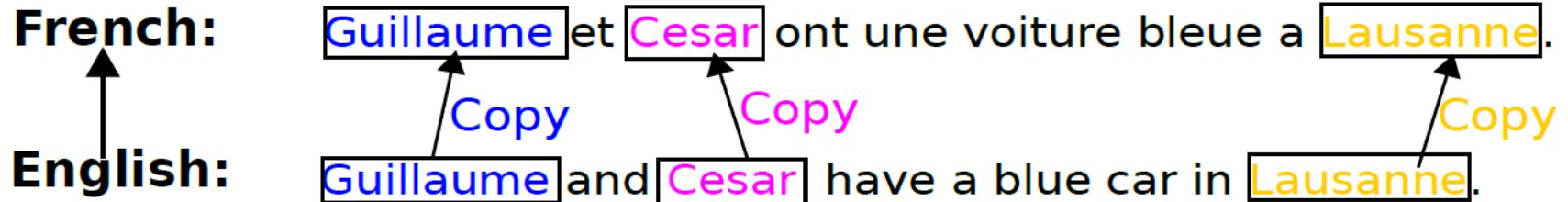
The process stops when “END” has the largest attention weights.

Applications - Summarization

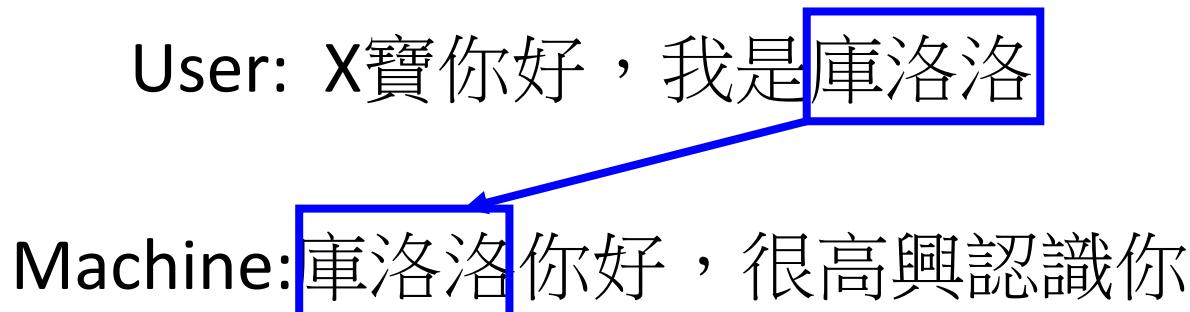


More Applications

Machine Translation

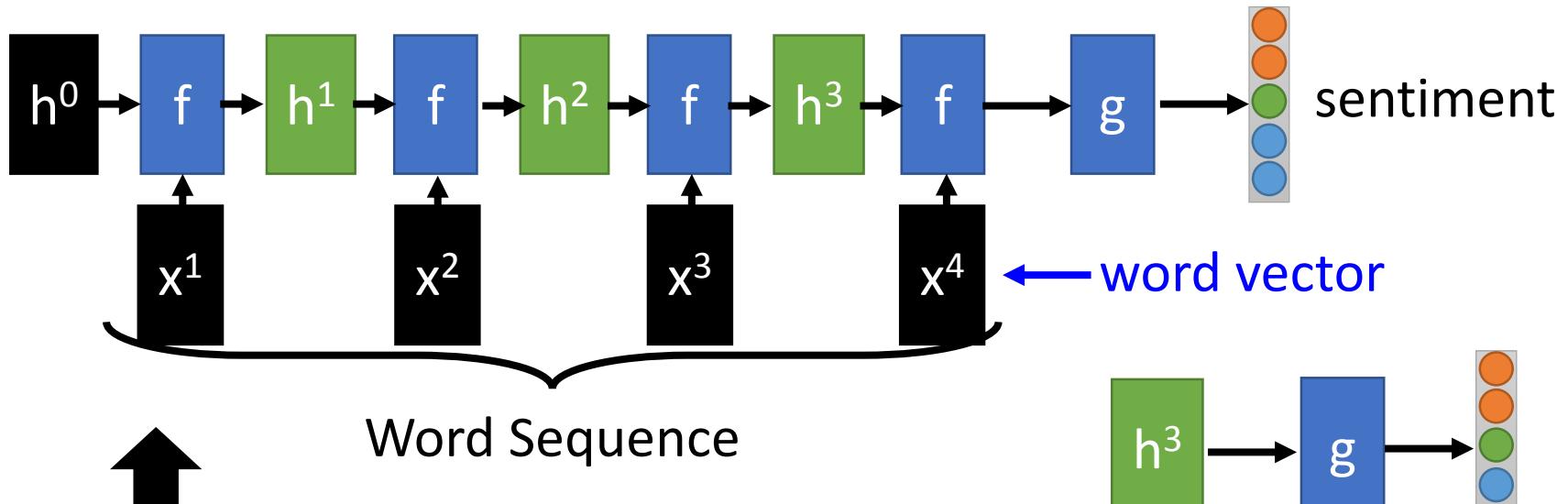


Chat-bot



Recursive Structure

Application: Sentiment Analysis

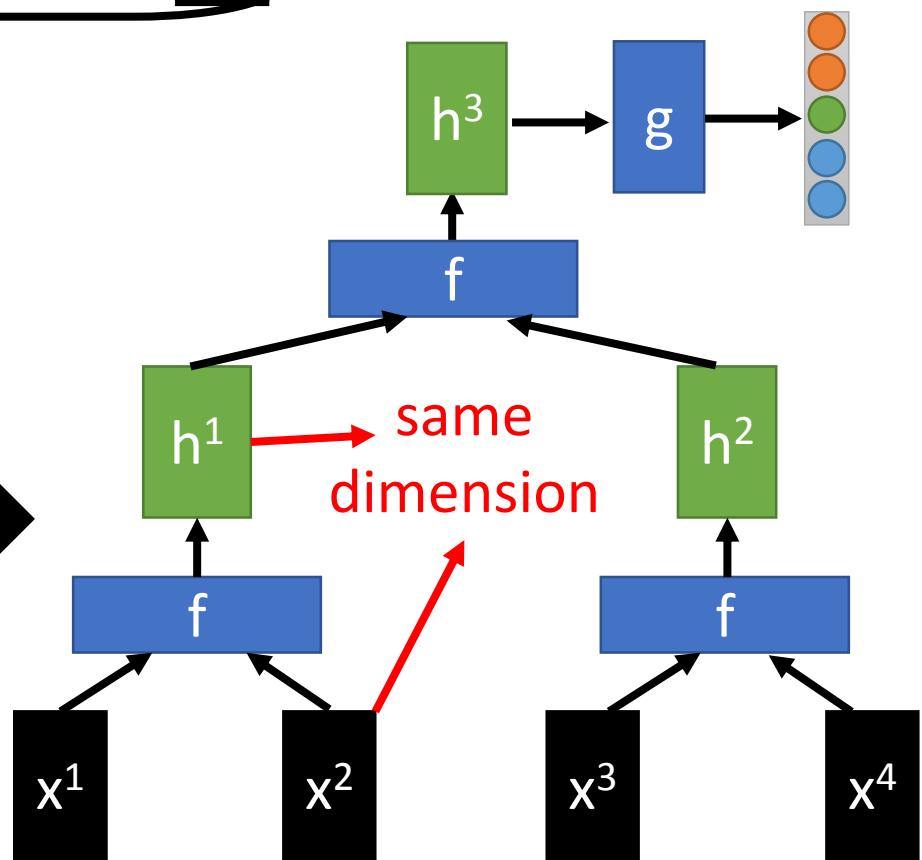


Recurrent Structure

Special case of recursive structure

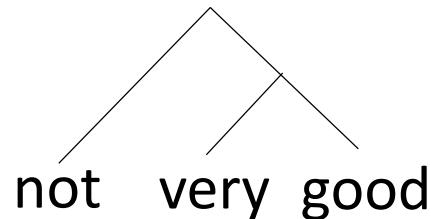
Recursive Structure

How to stack function f is already determined



Recursive Model

syntactic structure



How to do it is out
of the scope

word sequence:

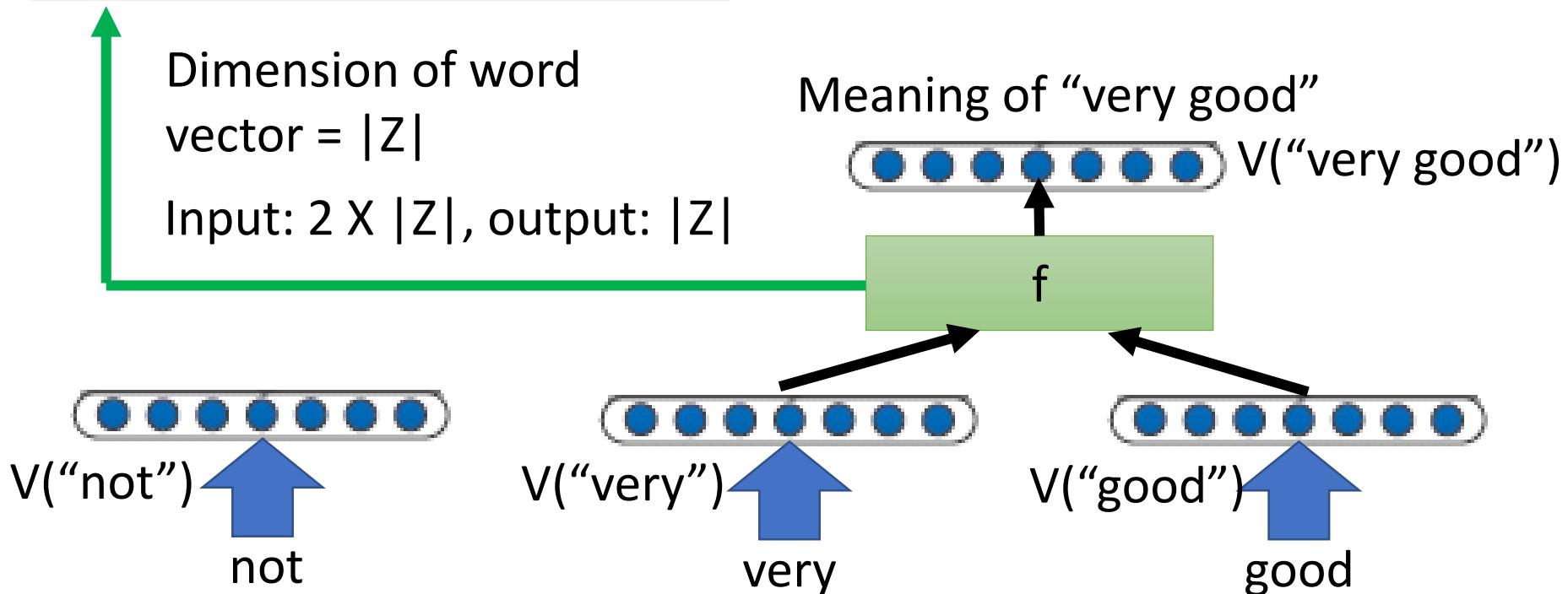
not

very

good

Recursive Model

By composing the two meaning, what should the meaning be.



Recursive Model

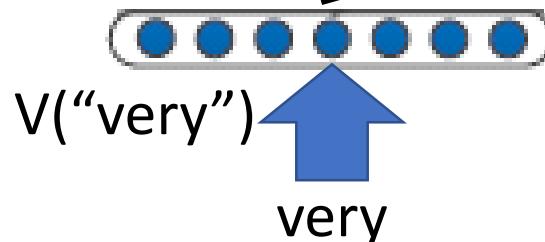
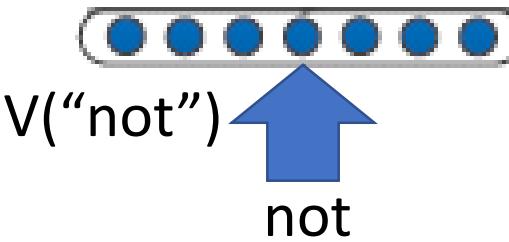
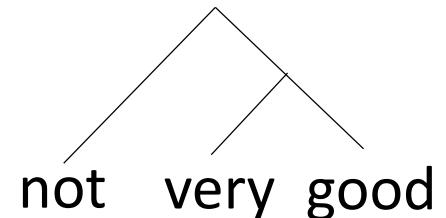
$$V(w_A w_B) \neq V(w_A) + V(w_B)$$

“not”: neutral

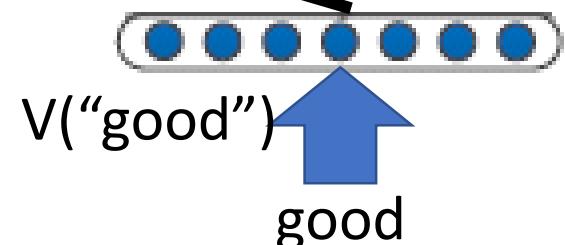
“good”: positive

“not good”: negative

syntactic structure



Meaning of “very good”



Recursive Model

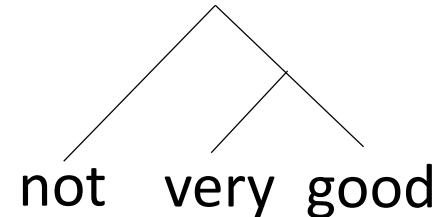
$$V(w_A w_B) \neq V(w_A) + V(w_B)$$

“棒”: positive

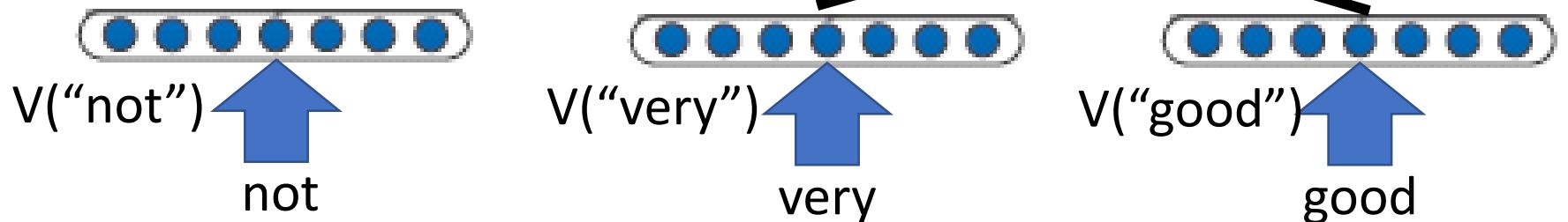
“好棒”: positive

“好棒棒”: negative

syntactic structure

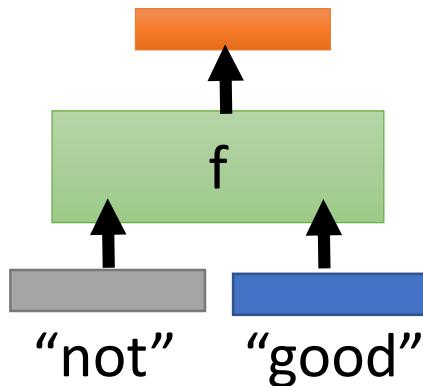


Meaning of “very good”

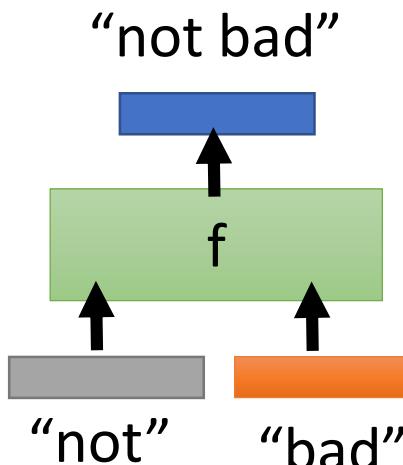


Recursive Model

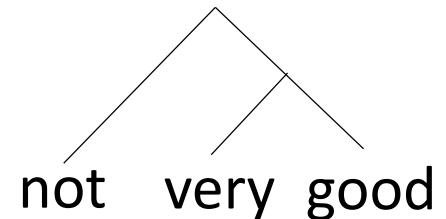
“not good”



“not bad”



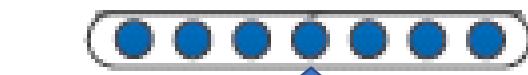
syntactic structure



Meaning of “very good”

: “reverse” another input

“not”



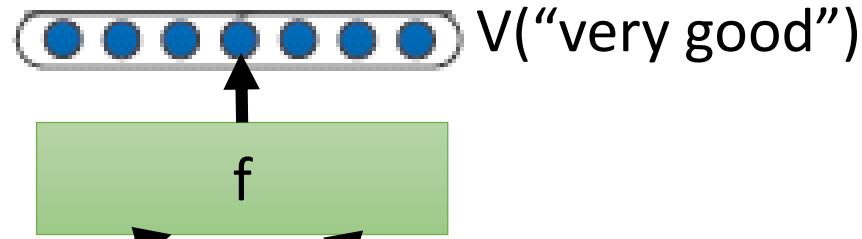
$V(\text{"not"})$

not



$V(\text{"very"})$

very

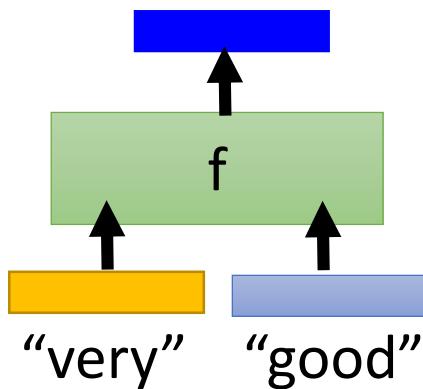


$V(\text{"good"})$

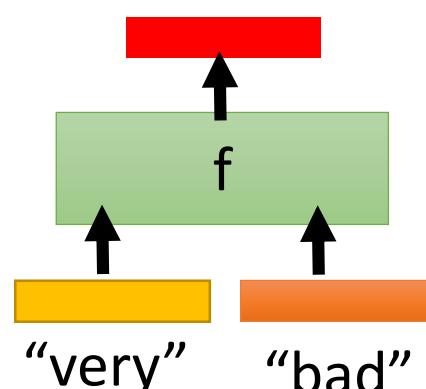
good

Recursive Model

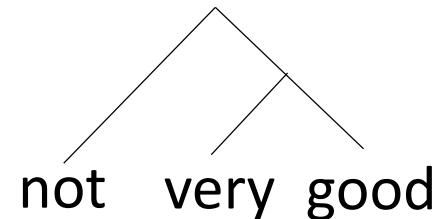
“very good”



“very bad”



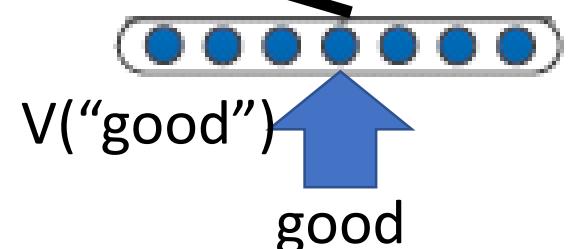
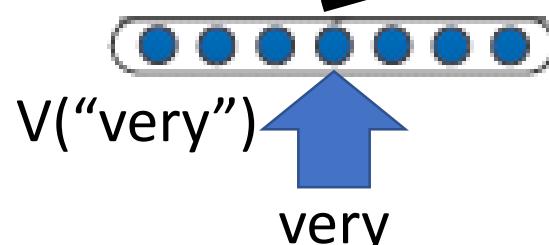
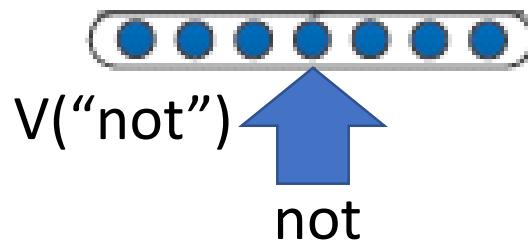
syntactic structure

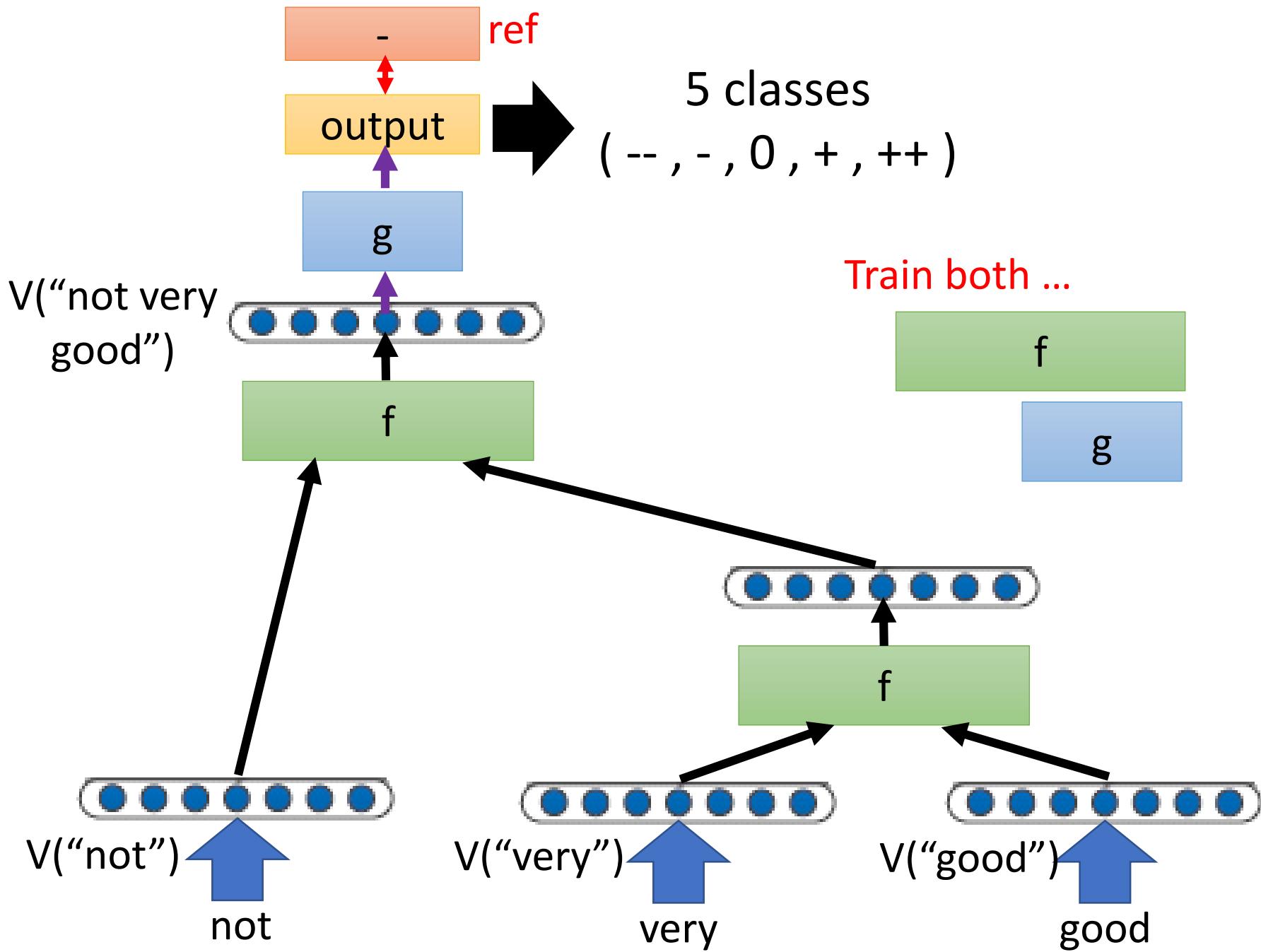


Meaning of “very good”

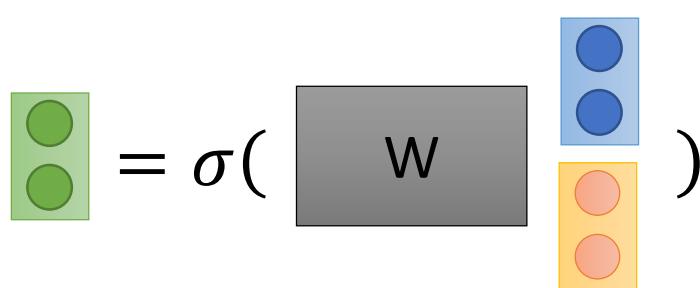


: “emphasize” another input
“very”

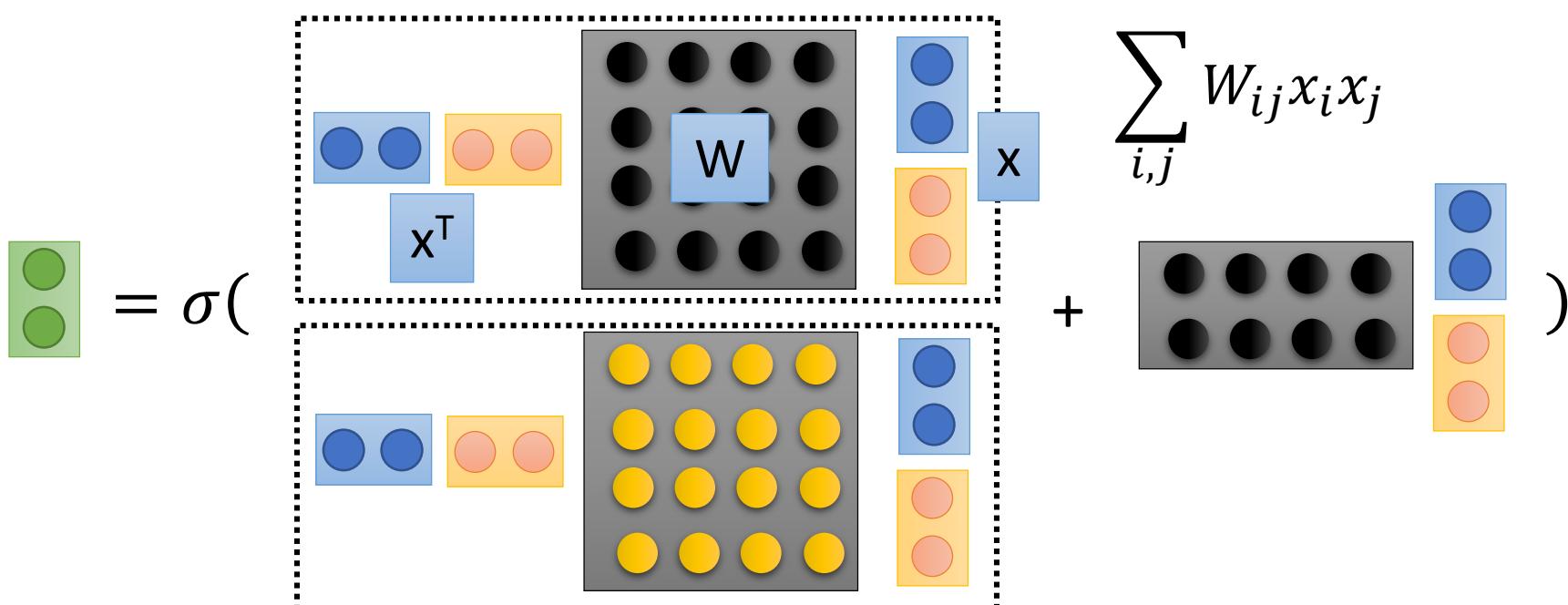
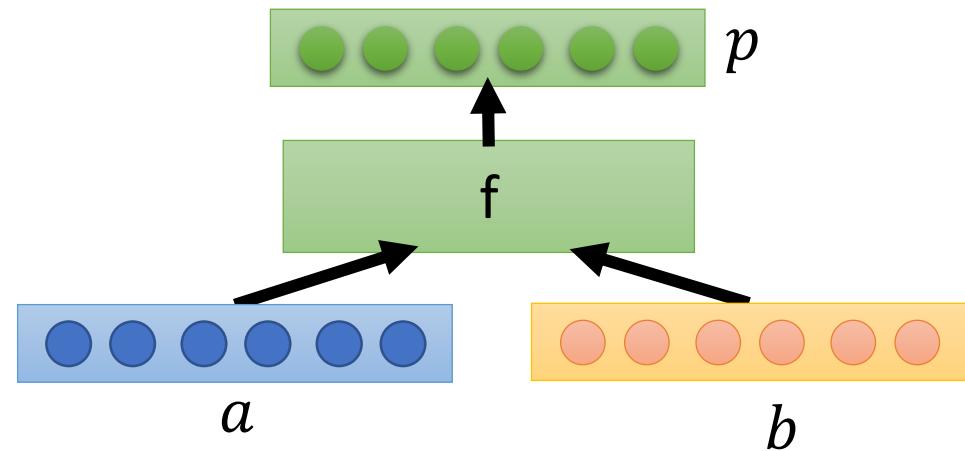




Recursive Neural Tensor Network

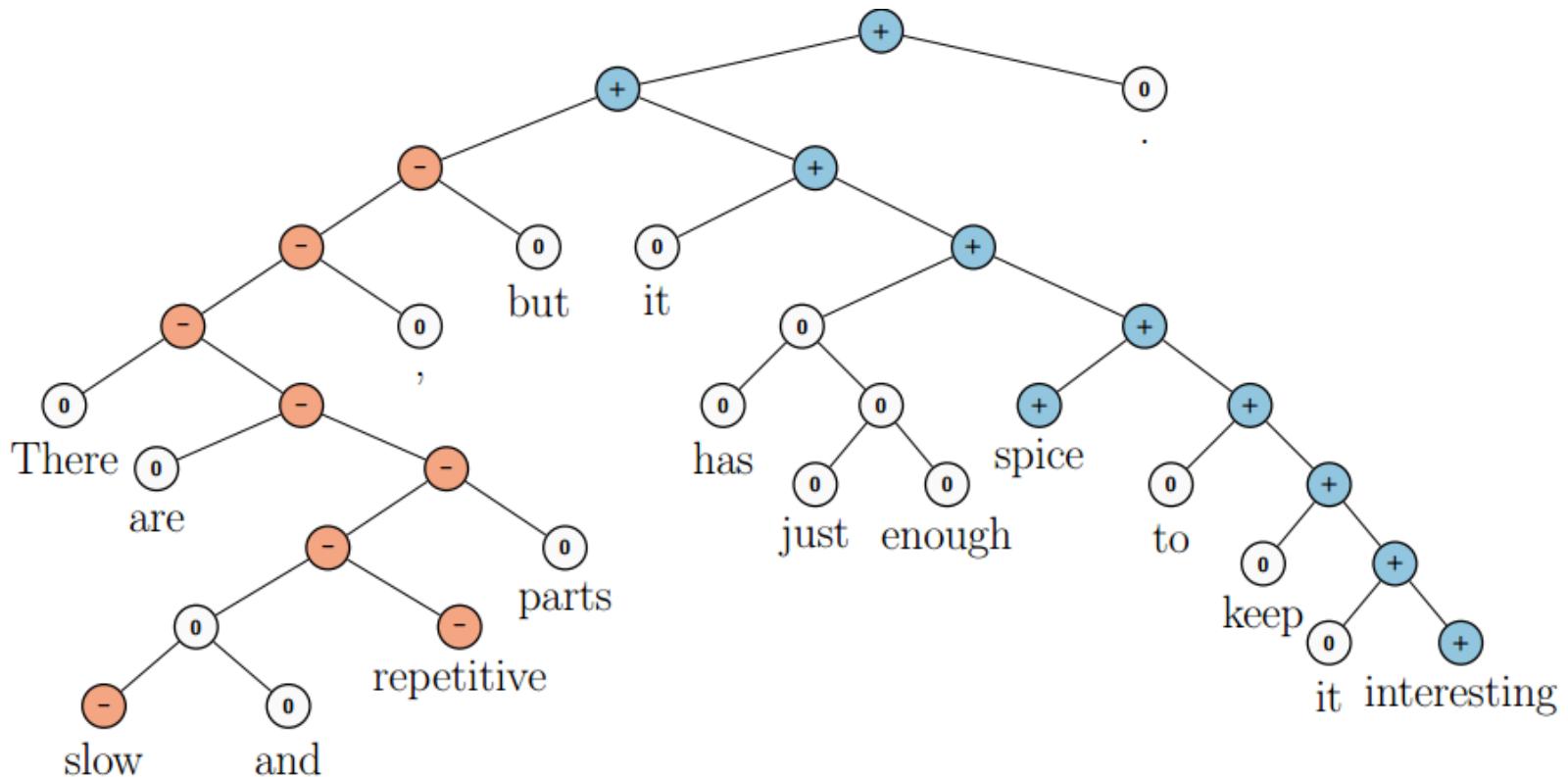


Little interaction between
a and b



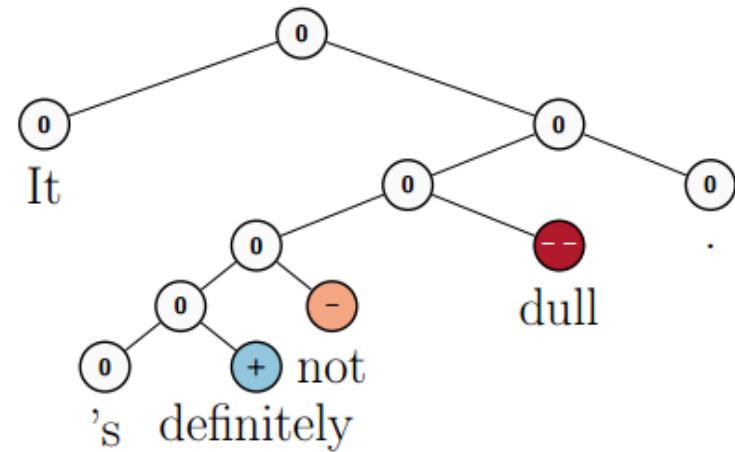
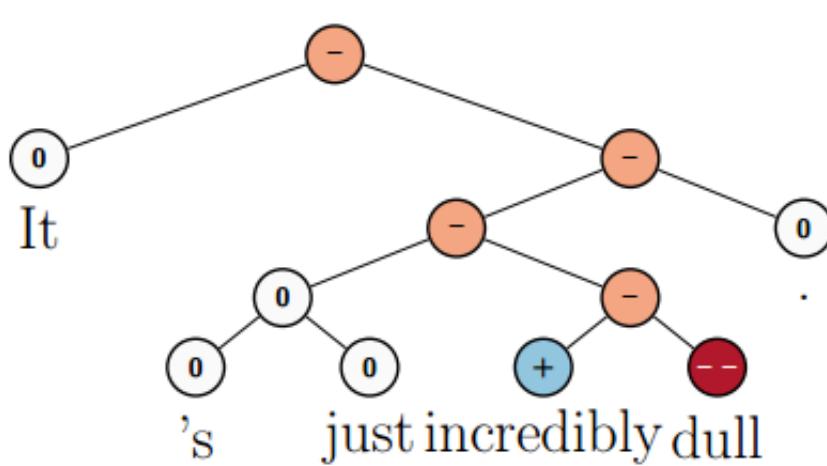
Experiments

5-class sentiment classification (-- , - , 0 , + , ++)



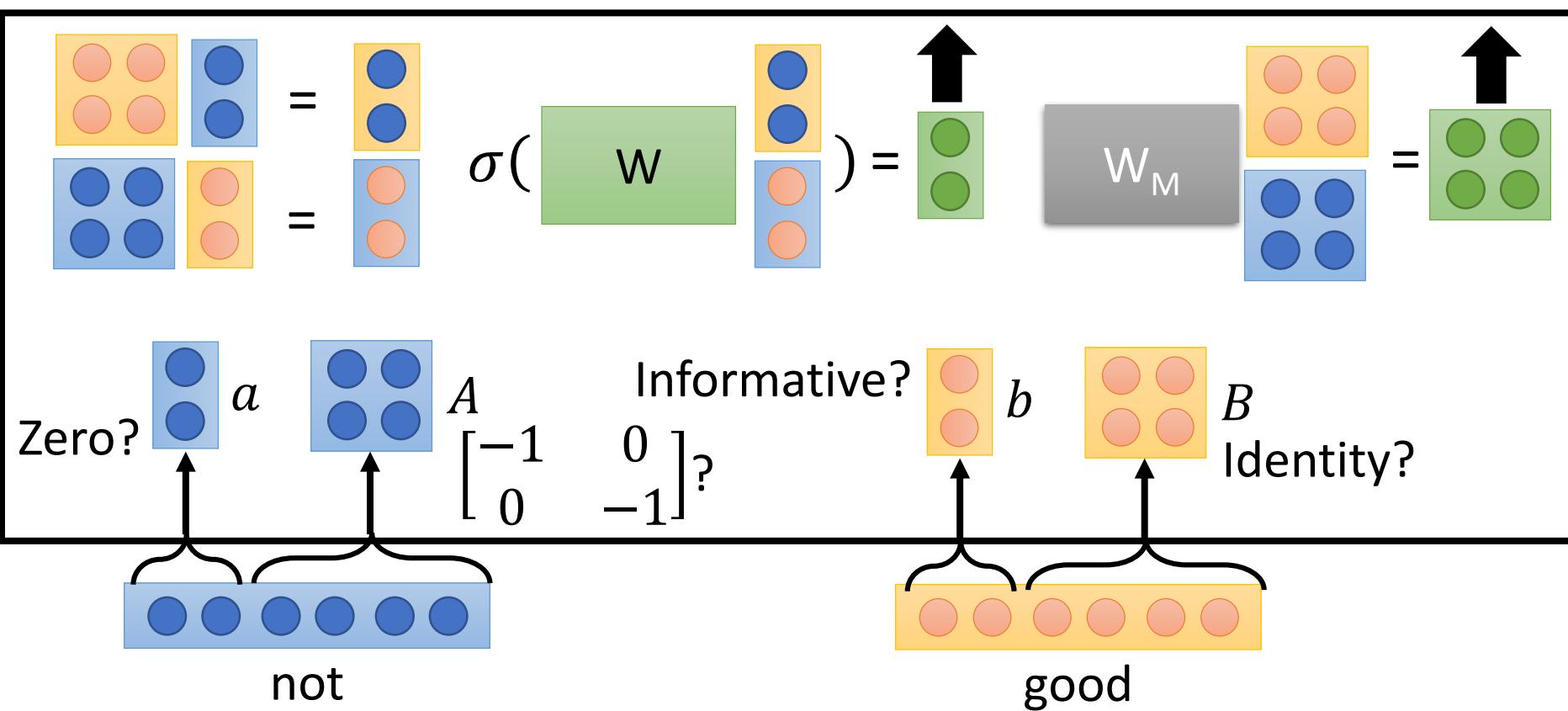
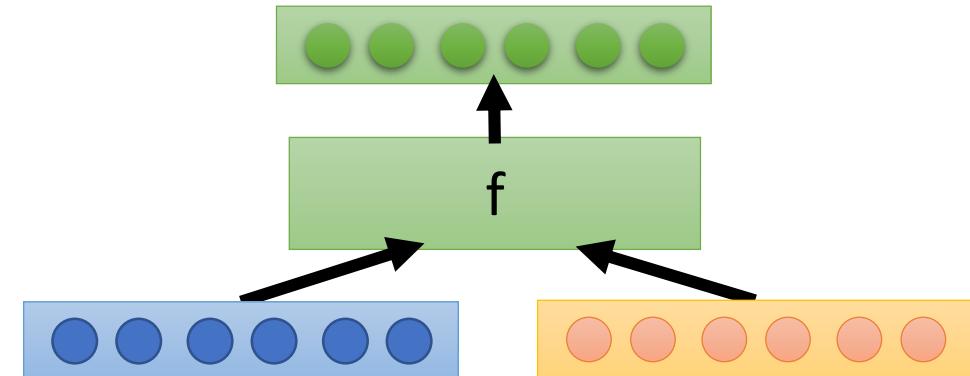
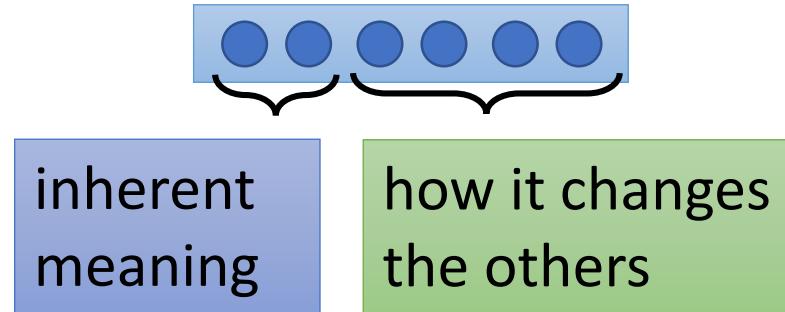
Demo: <http://nlp.stanford.edu:8080/sentiment/rnntDemo.html>

Experiments



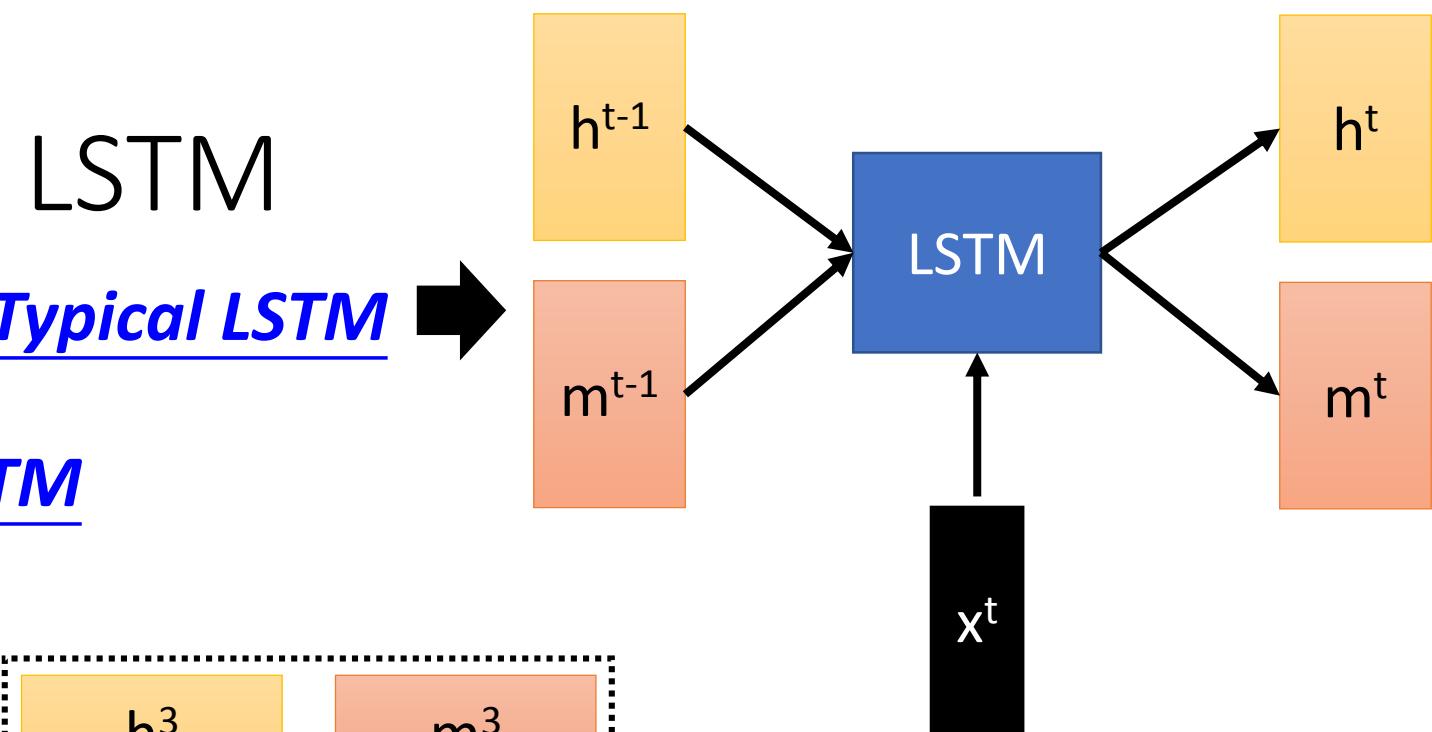
Socher, Richard, et al. "Recursive deep models for semantic compositionality over a sentiment treebank." *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*. Vol. 1631. 2013.

Matrix-Vector Recursive Network

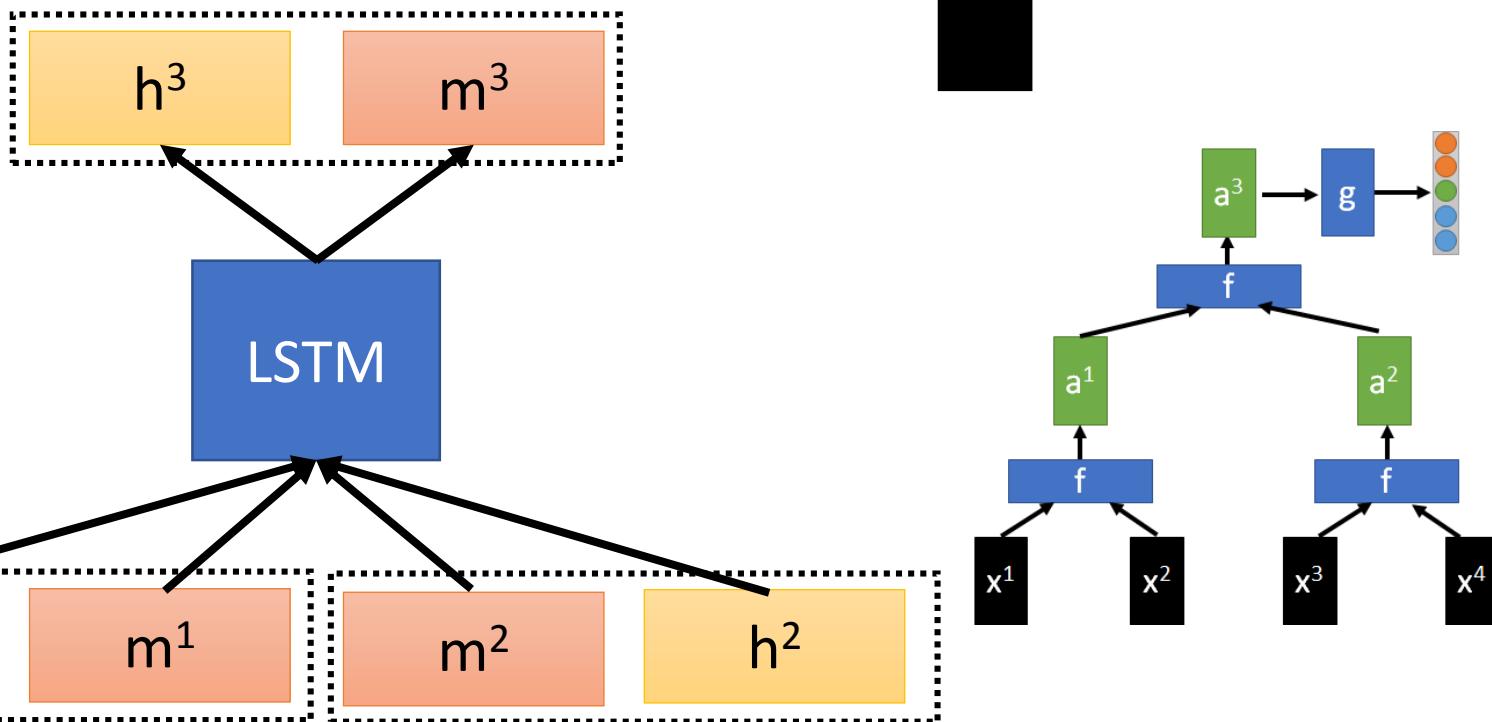
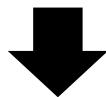


Tree LSTM

Typical LSTM



Tree LSTM



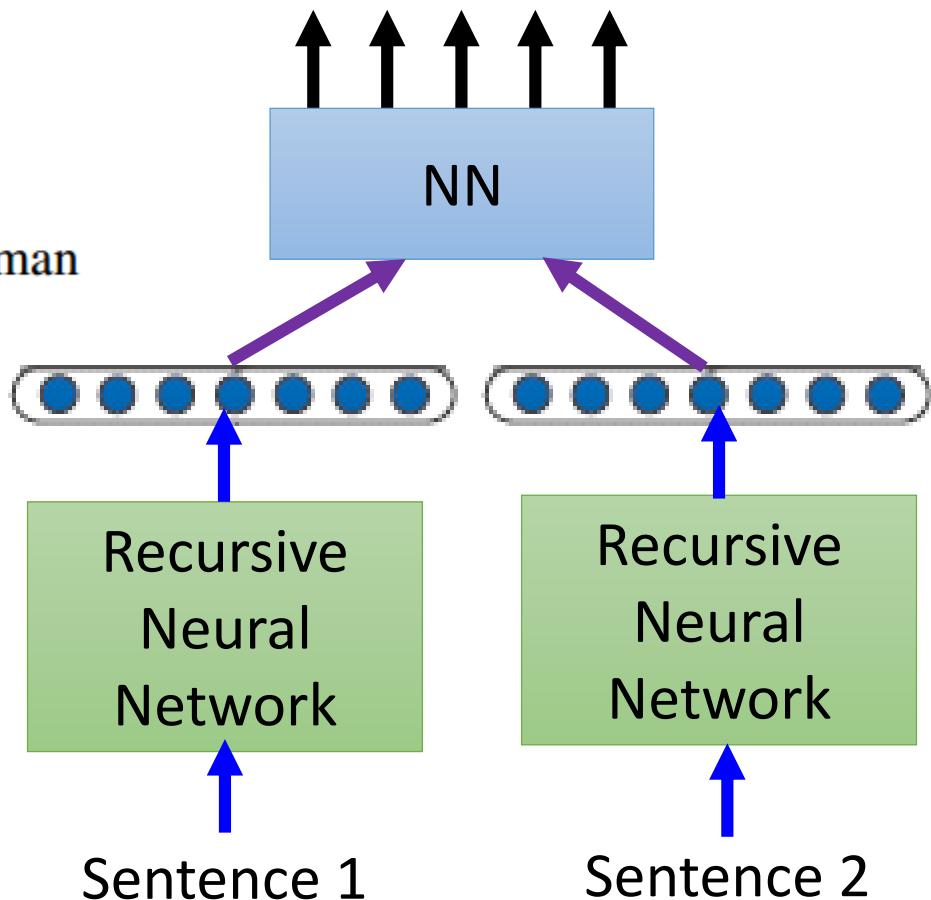
More Applications

- Sentence relatedness

a woman is slicing potatoes

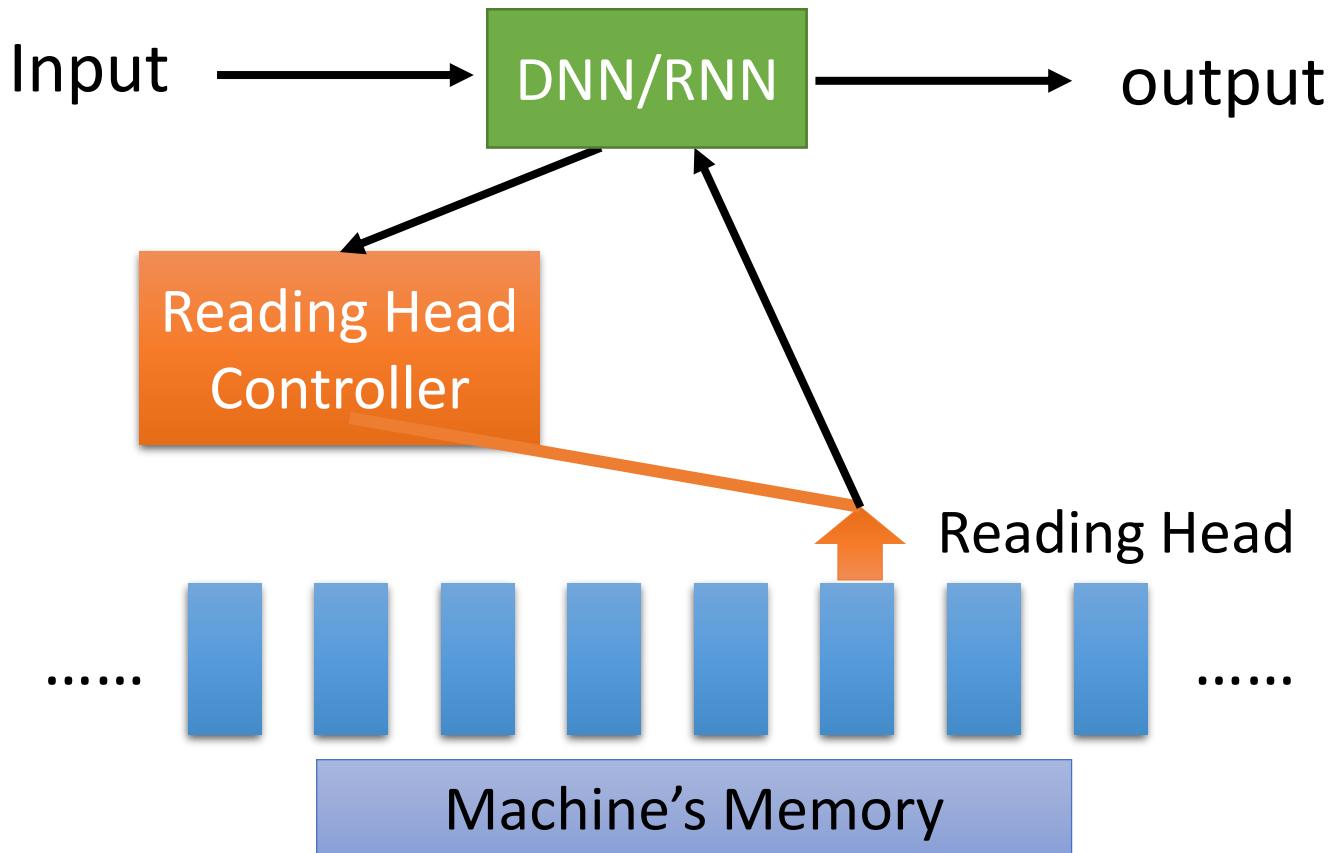
- 4.82 a woman is cutting potatoes
- 4.70 potatoes are being sliced by a woman
- 4.39 tofu is being sliced by a woman

Tai, Kai Sheng, Richard Socher, and Christopher D. Manning. "Improved semantic representations from tree-structured long short-term memory networks." *arXiv preprint arXiv:1503.00075* (2015).



Attention-based Model

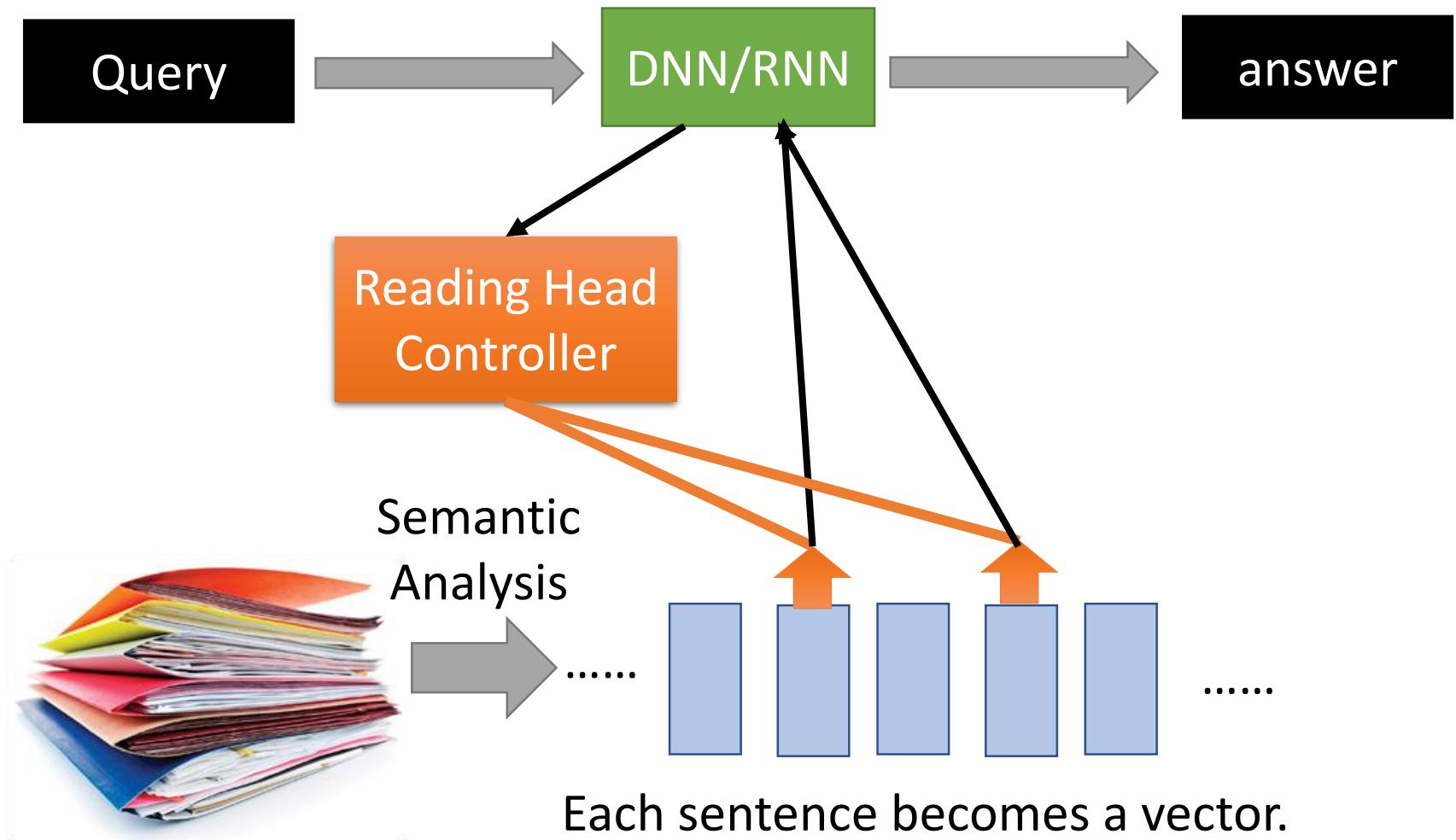
External Memory



Ref:

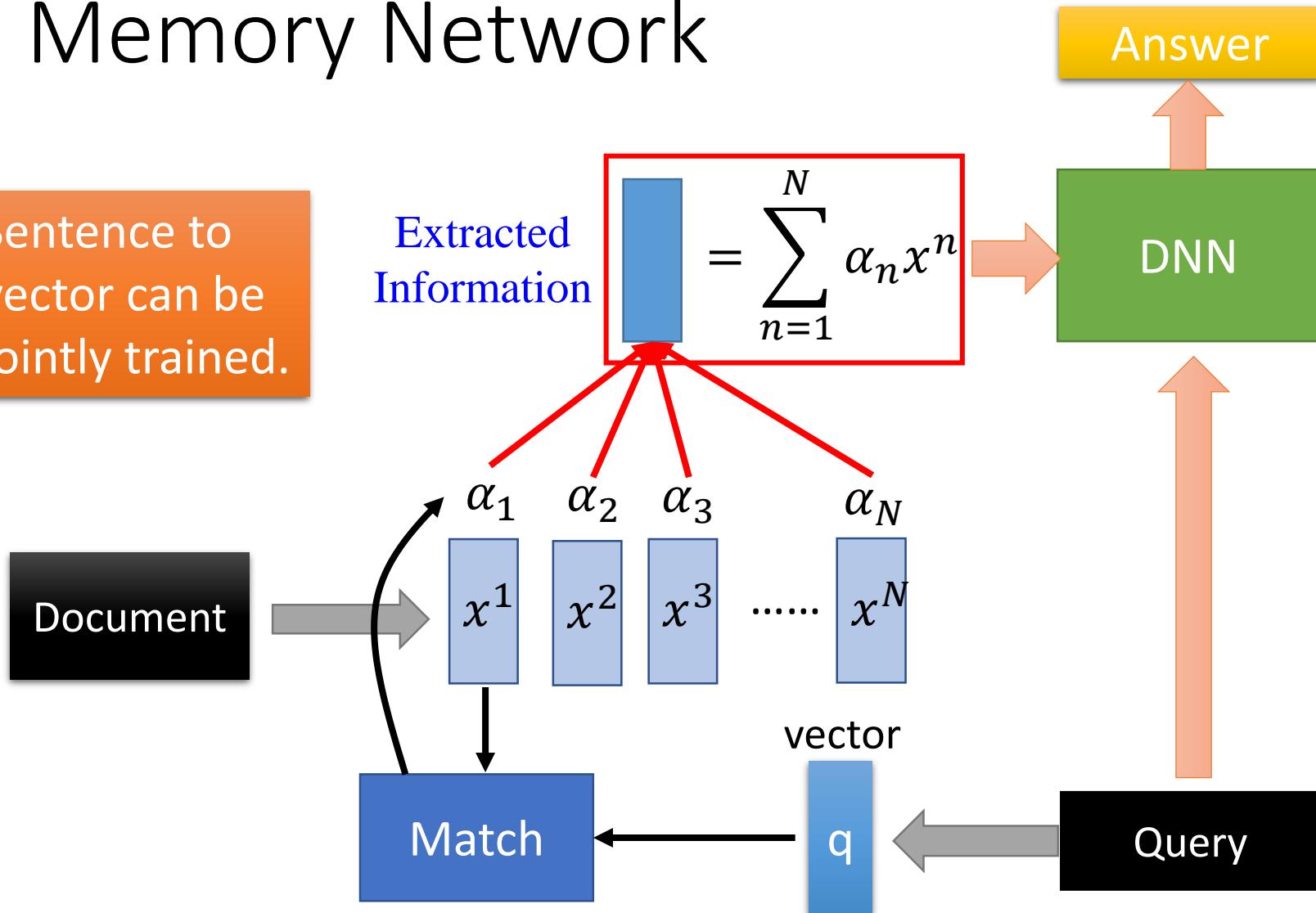
[http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/Attain%20\(v3\).ecm.mp4/index.html](http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/Attain%20(v3).ecm.mp4/index.html)

Reading Comprehension



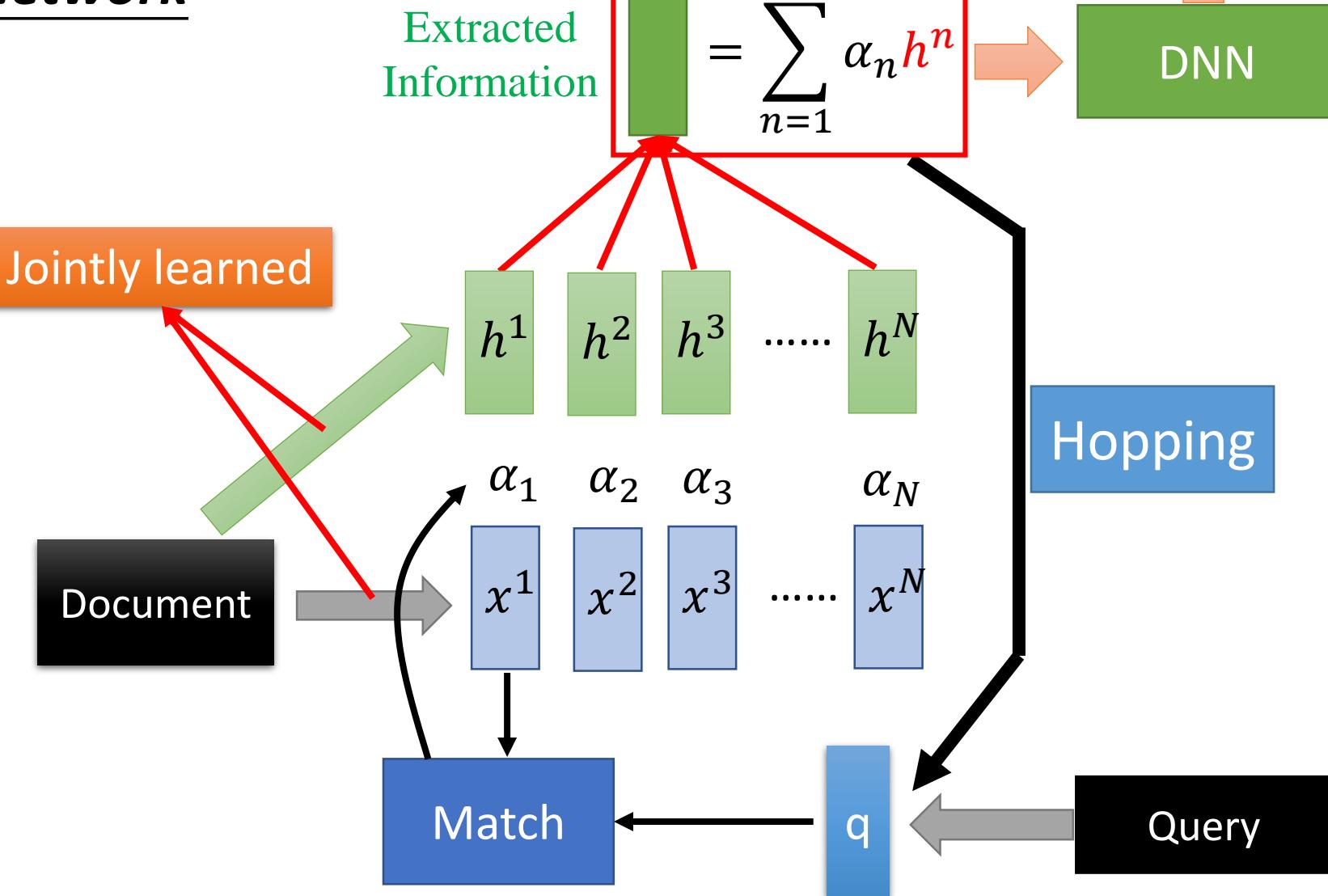
Memory Network

Sentence to vector can be jointly trained.

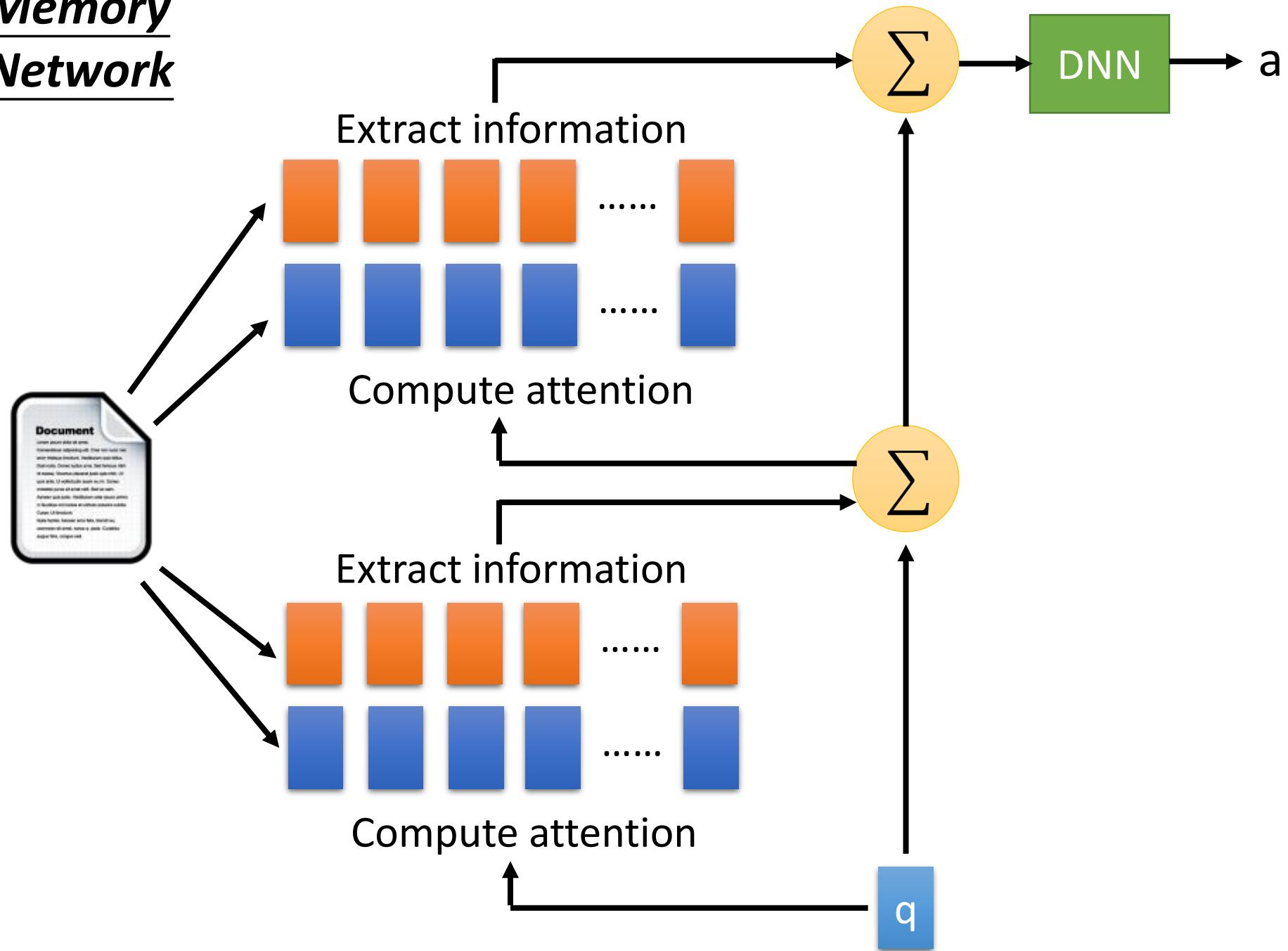


Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, Rob Fergus, "End-To-End Memory Networks", NIPS, 2015

Memory Network



Memory Network



Multiple-hop

- End-To-End Memory Networks. S. Sukhbaatar, A. Szlam, J. Weston, R. Fergus. NIPS, 2015.

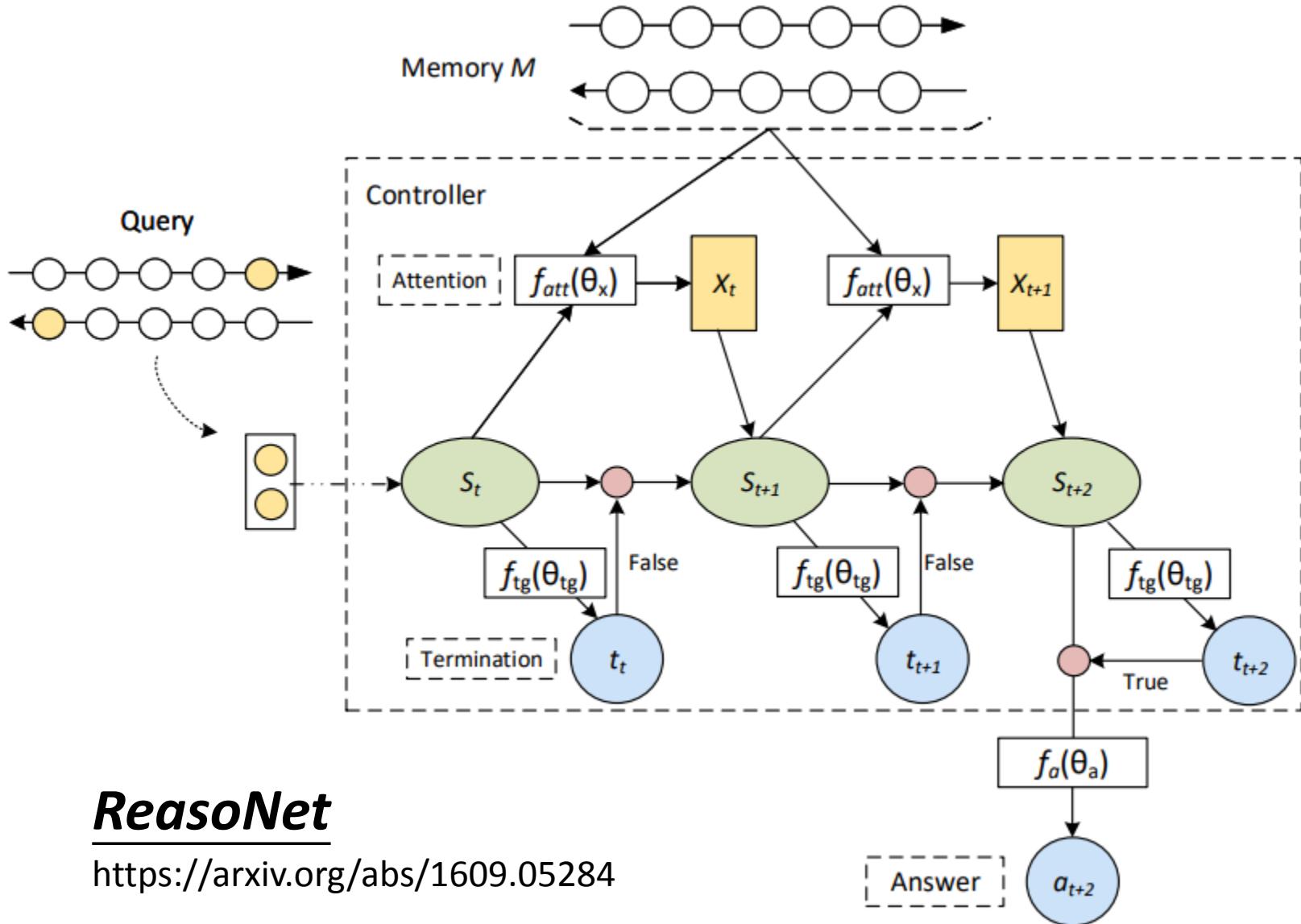
The position of reading head:

Story (16: basic induction)	Support	Hop 1	Hop 2	Hop 3
Brian is a frog.	yes	0.00	0.98	0.00
Lily is gray.		0.07	0.00	0.00
Brian is yellow.	yes	0.07	0.00	1.00
Julius is green.		0.06	0.00	0.00
Greg is a frog.	yes	0.76	0.02	0.00
What color is Greg? Answer: yellow		Prediction: yellow		

Keras has example:

https://github.com/fchollet/keras/blob/master/examples/babi_memnn.py

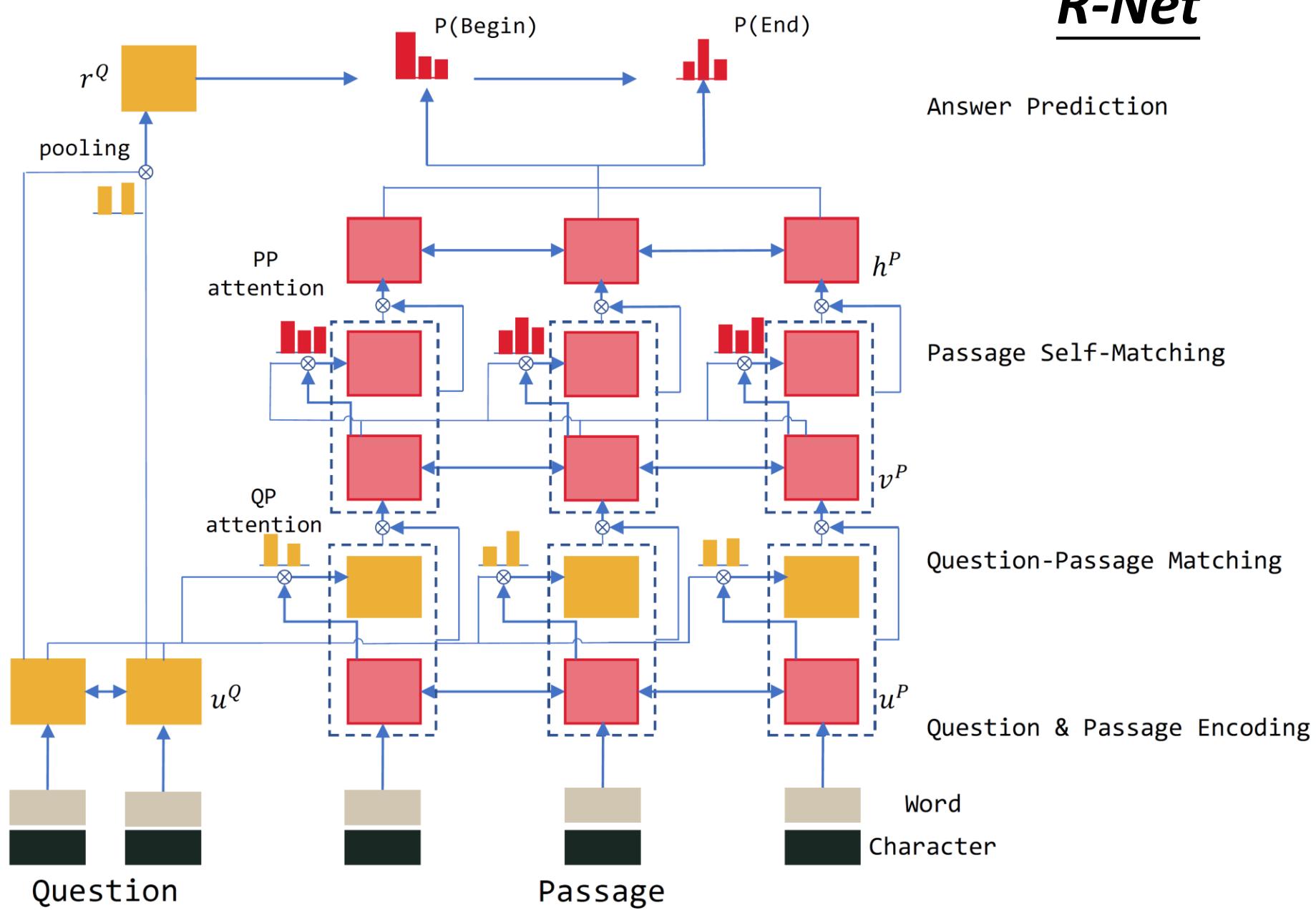
Multiple-hop

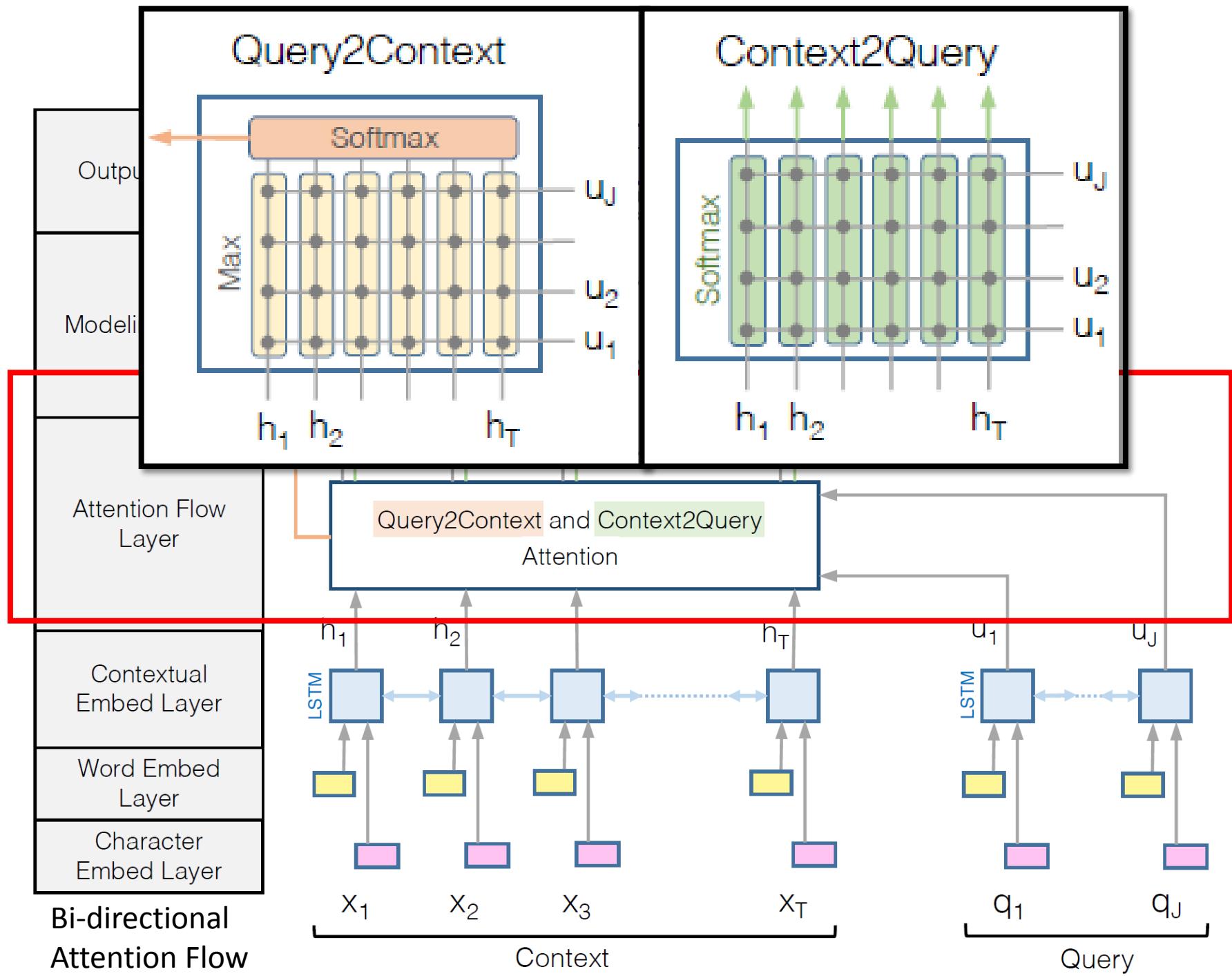


ReasoNet

<https://arxiv.org/abs/1609.05284>

R-Net





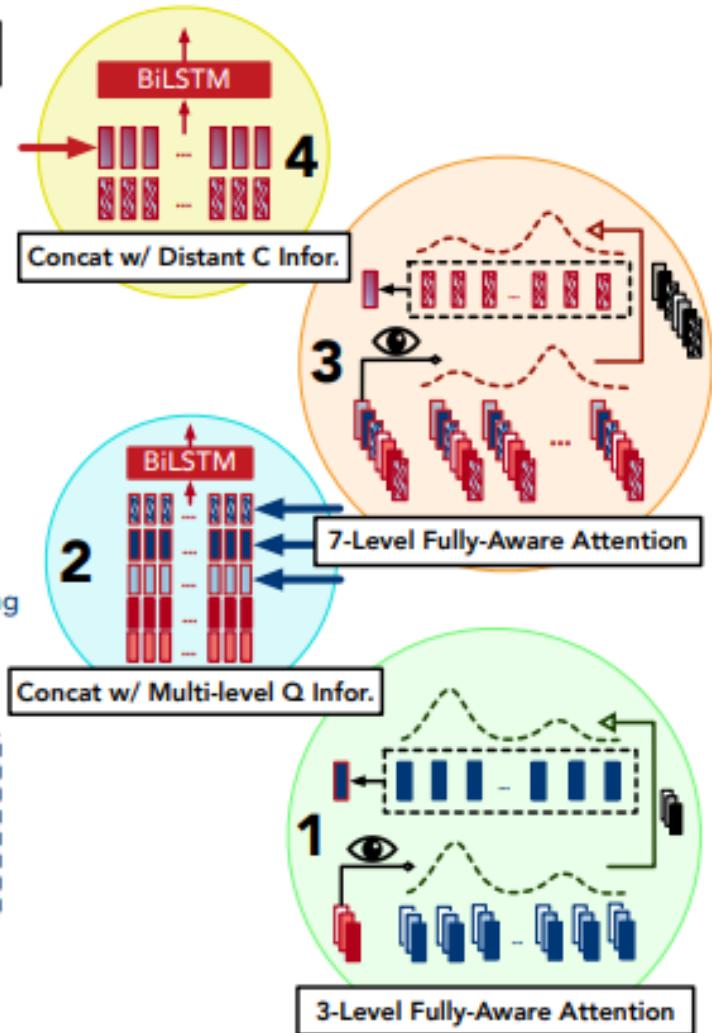
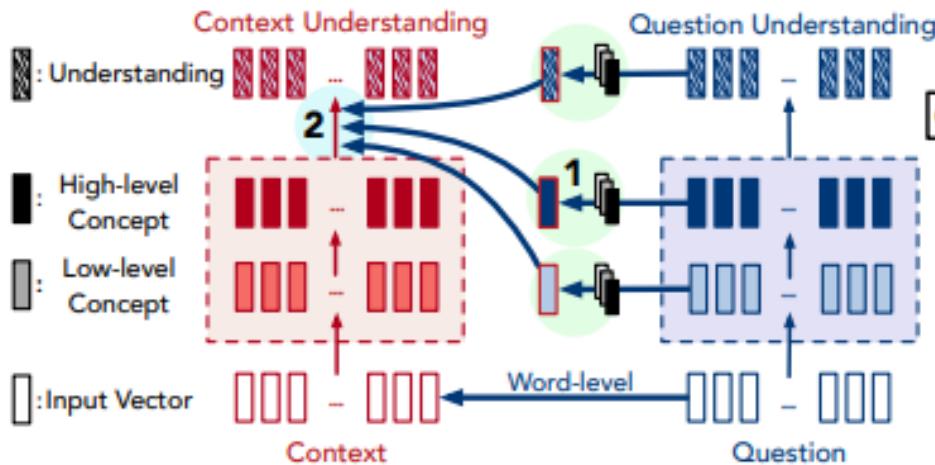
Fully-Aware Fusion Network

Fully-Aware Self-Boosted Fusion

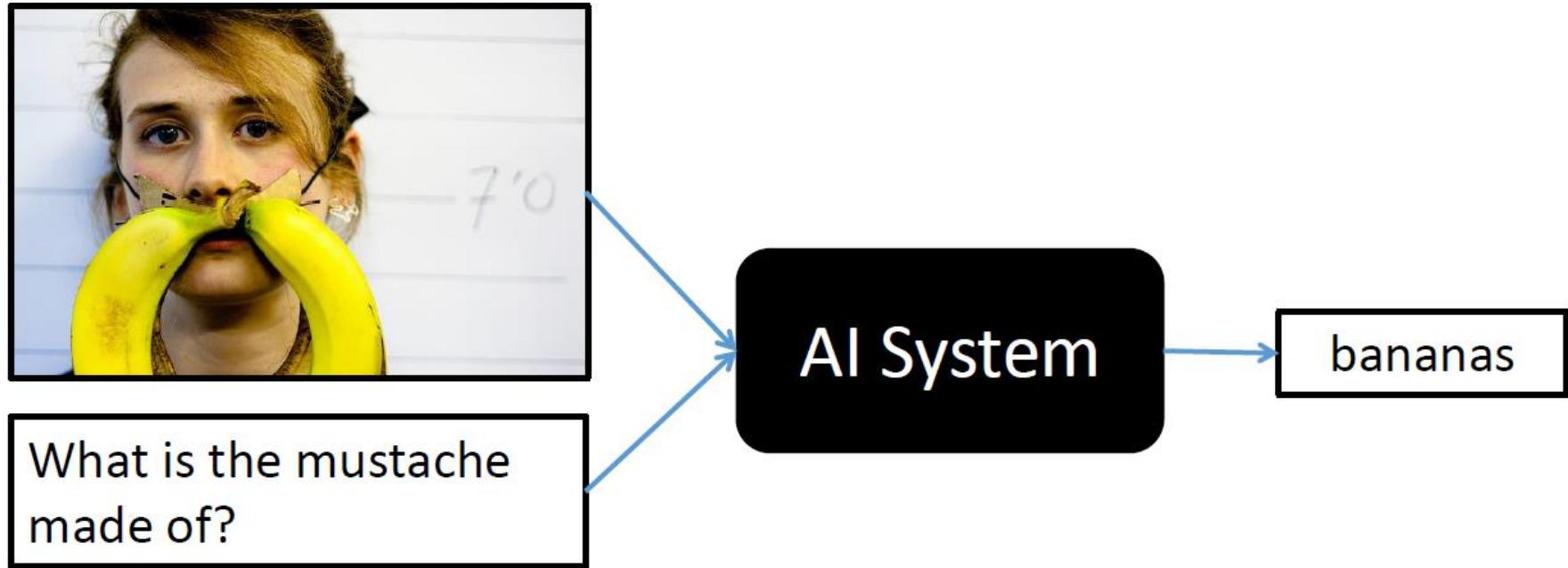


The above can be used to capture long range info.

Fully-Aware Multi-level Fusion

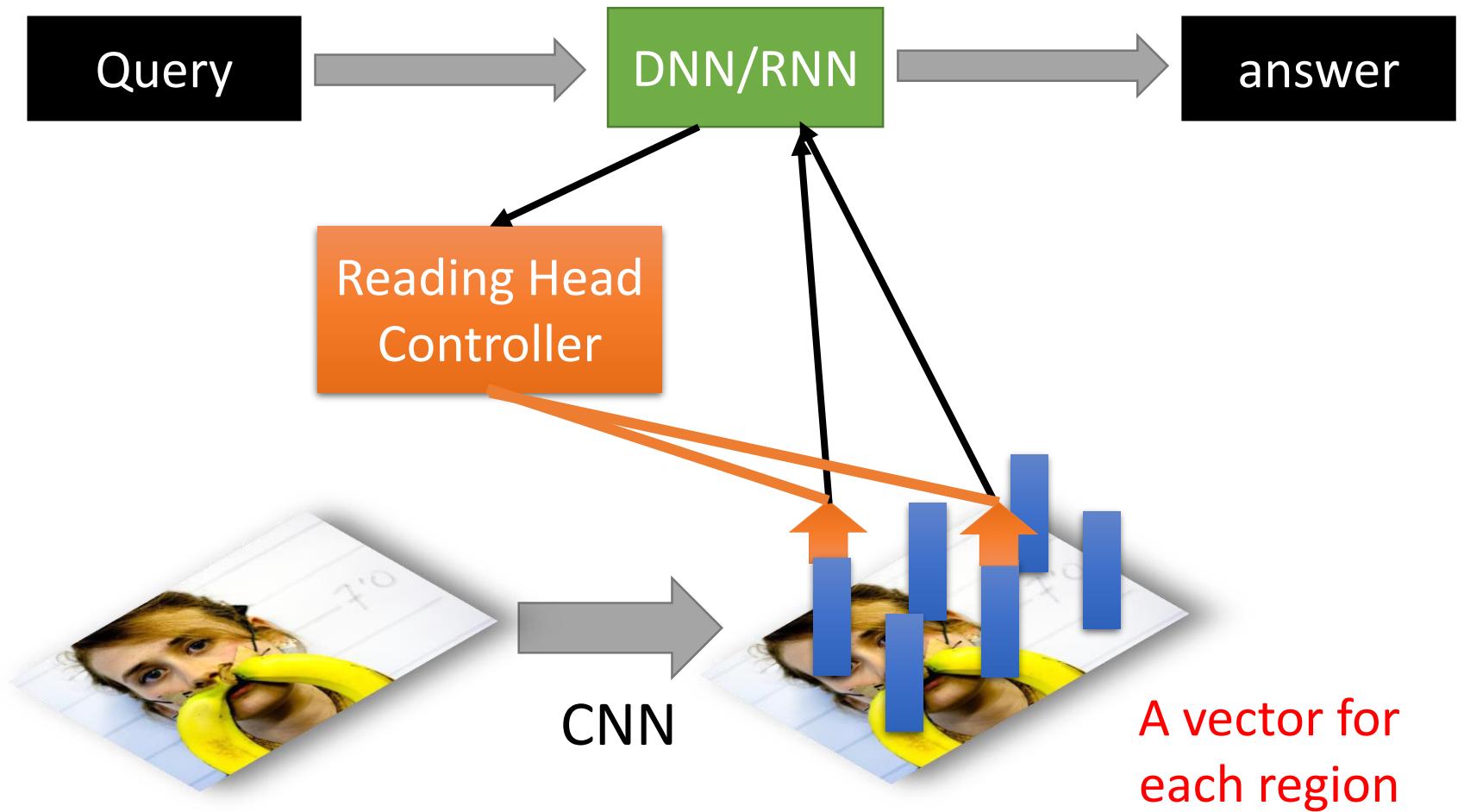


Visual Question Answering



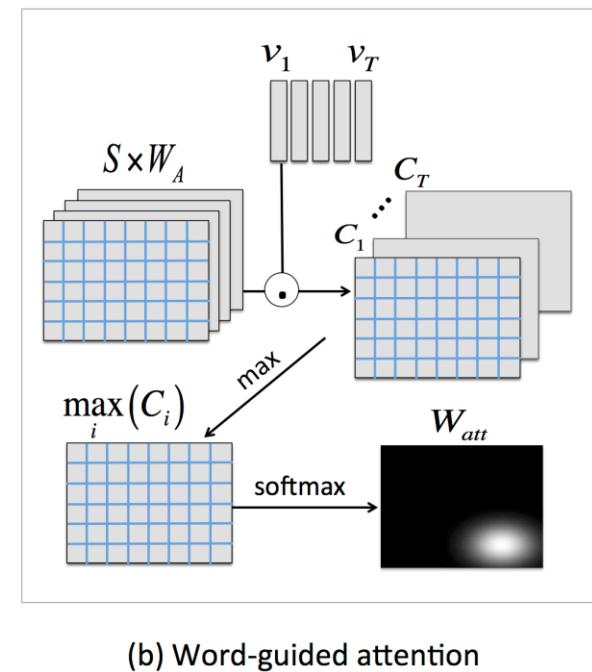
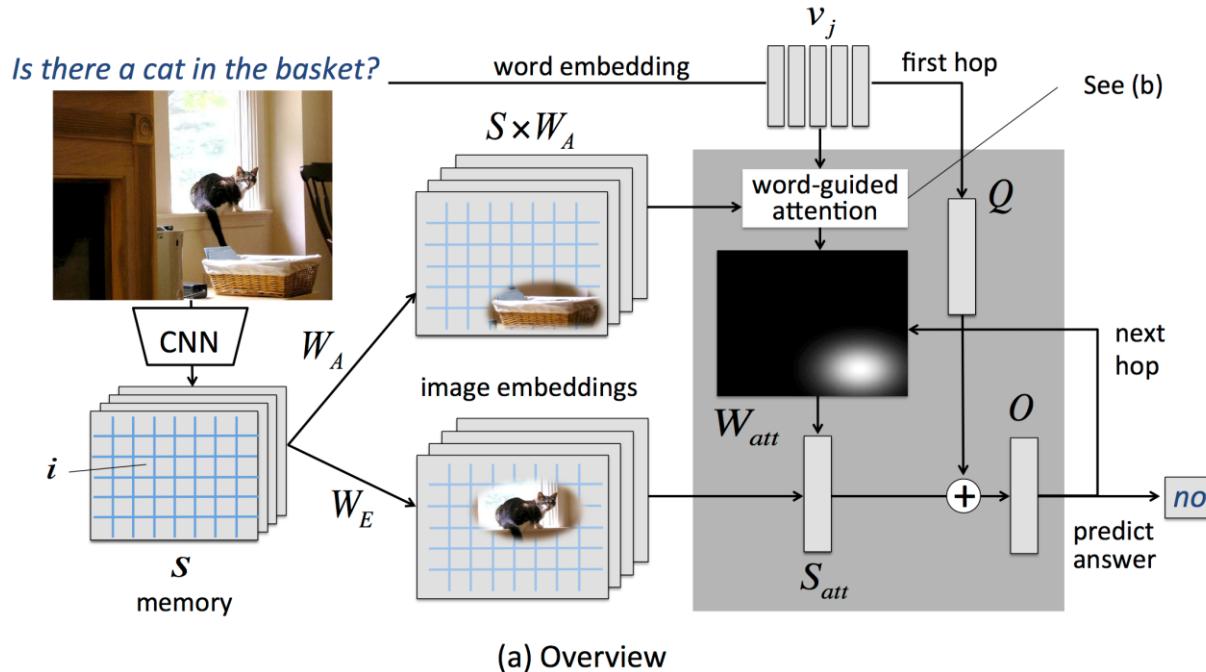
source: <http://visualqa.org/>

Visual Question Answering



Visual Question Answering

- Huijuan Xu, Kate Saenko. Ask, Attend and Answer: Exploring Question-Guided Spatial Attention for Visual Question Answering. arXiv Pre-Print, 2015



Visual Question Answering

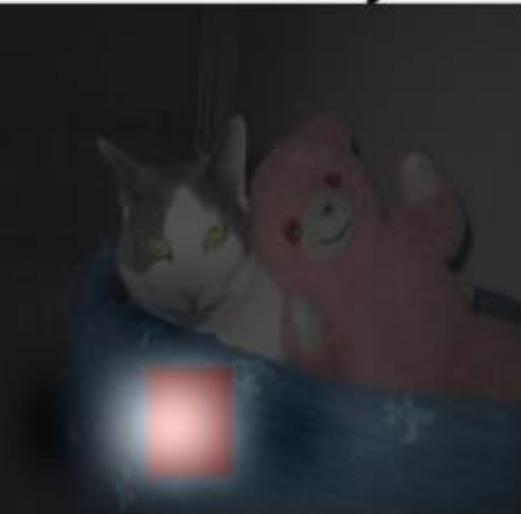
- Huijuan Xu, Kate Saenko. Ask, Attend and Answer: Exploring Question-Guided Spatial Attention for Visual Question Answering. arXiv Pre-Print, 2015

Is there a red square on the bottom of the cat?

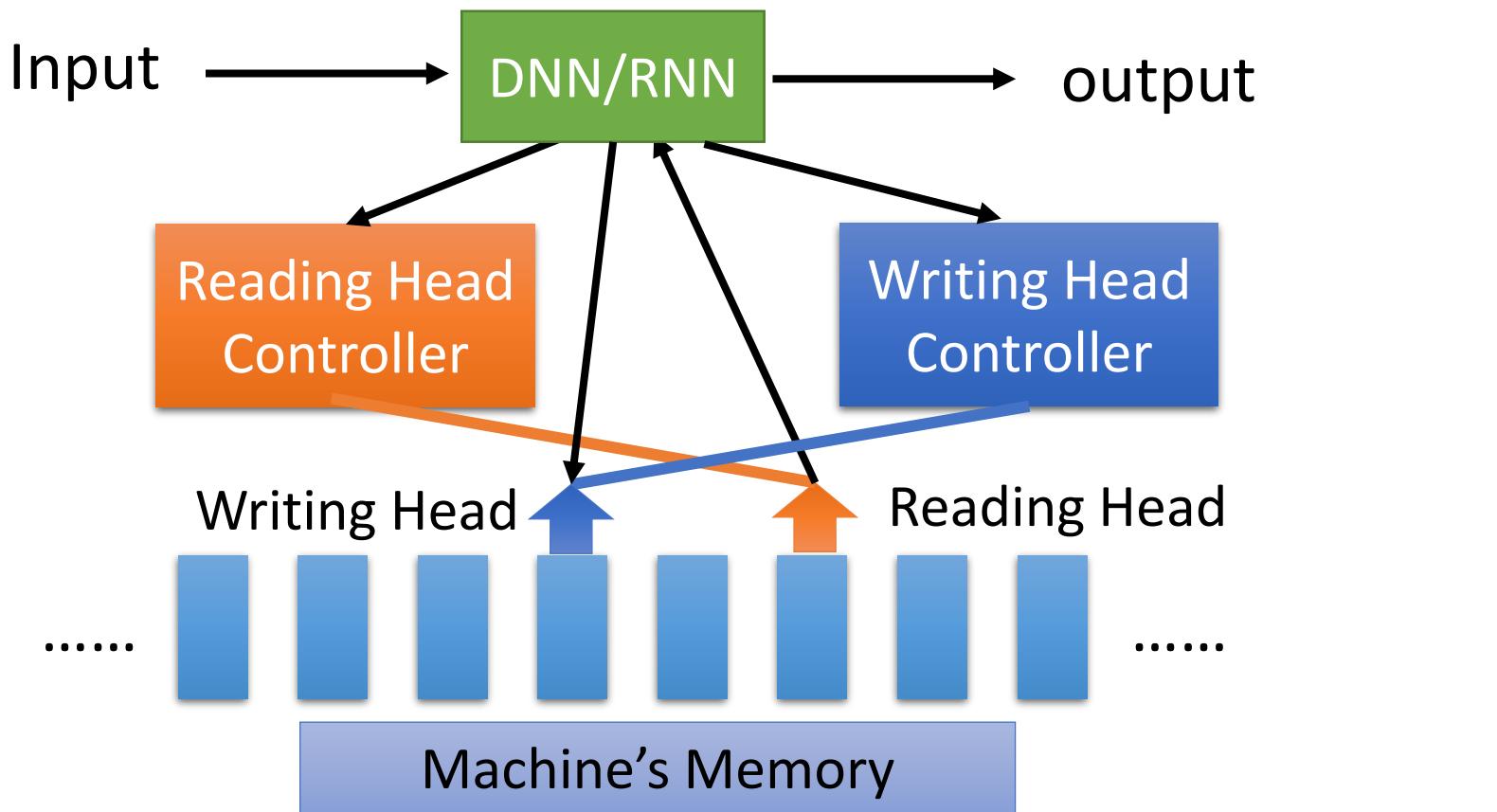
GT: yes



Prediction: yes

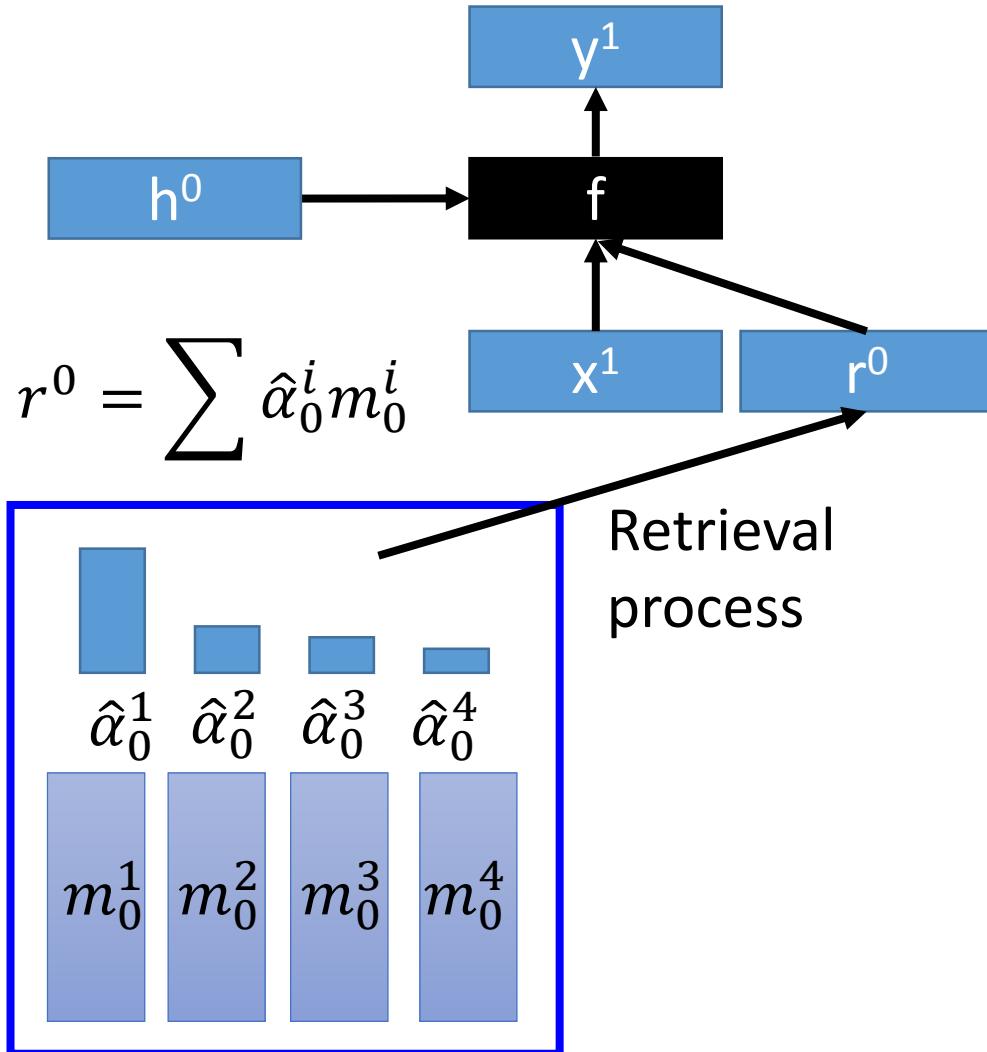


External Memory v2

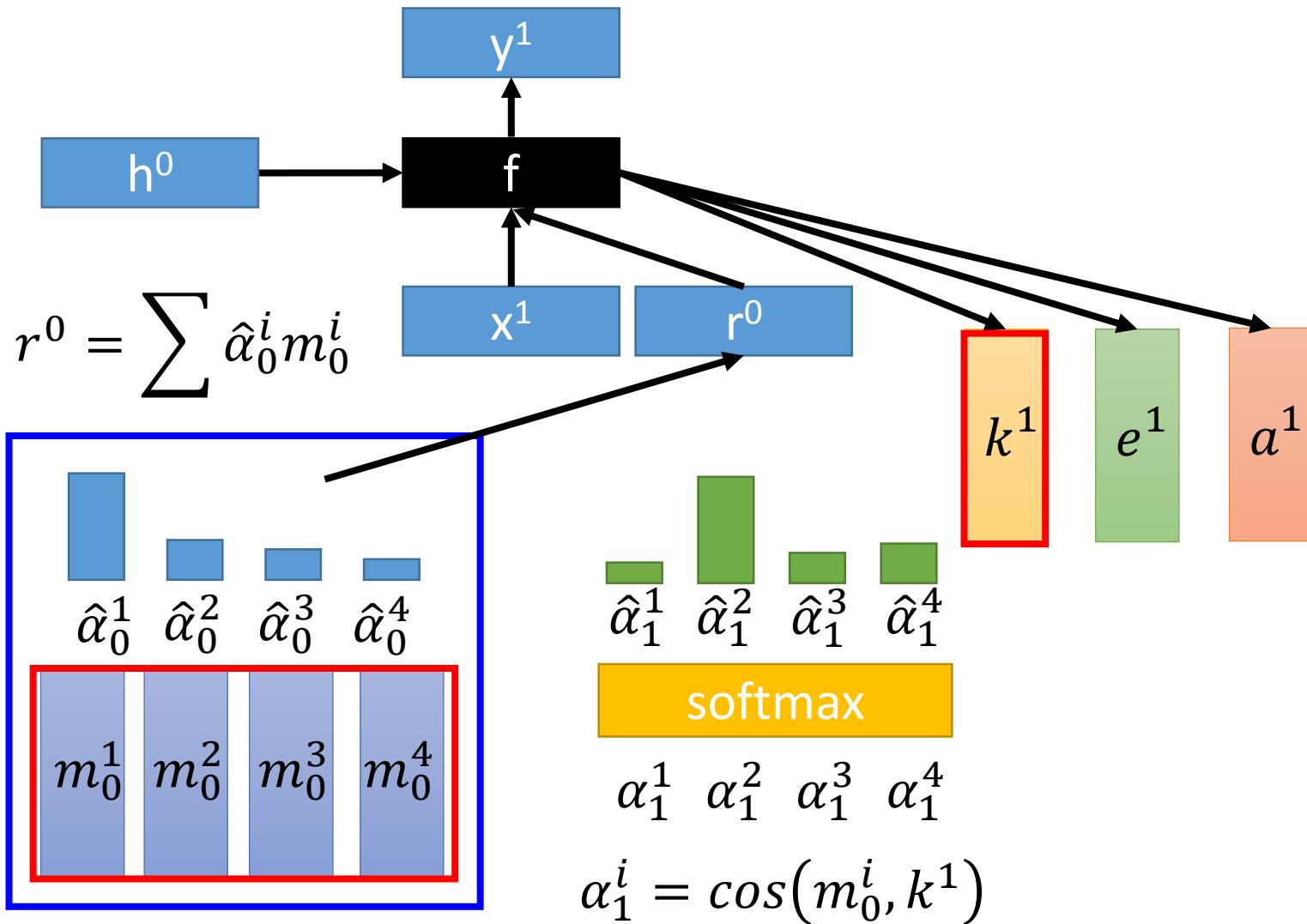


Neural Turing Machine

Neural Turing Machine



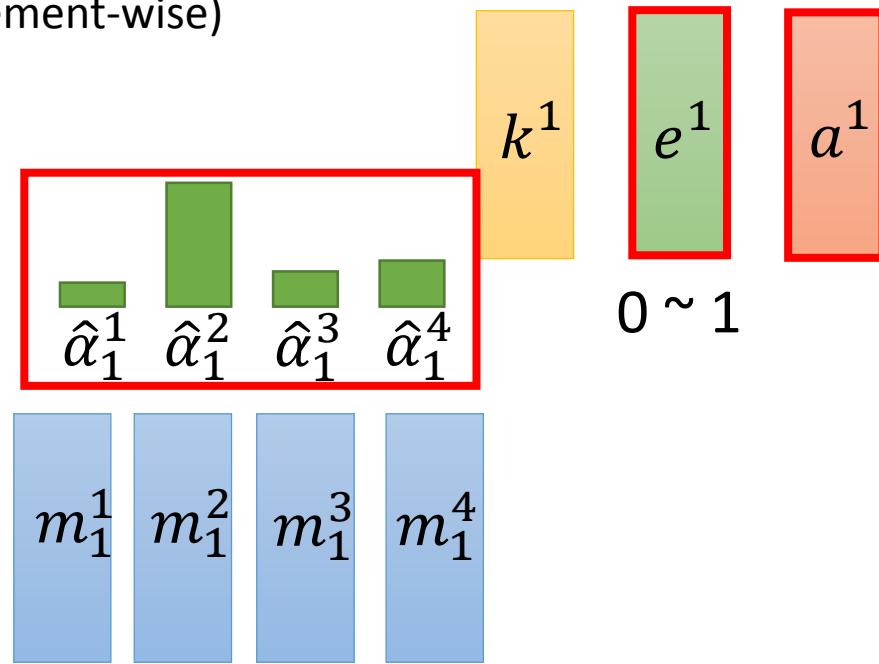
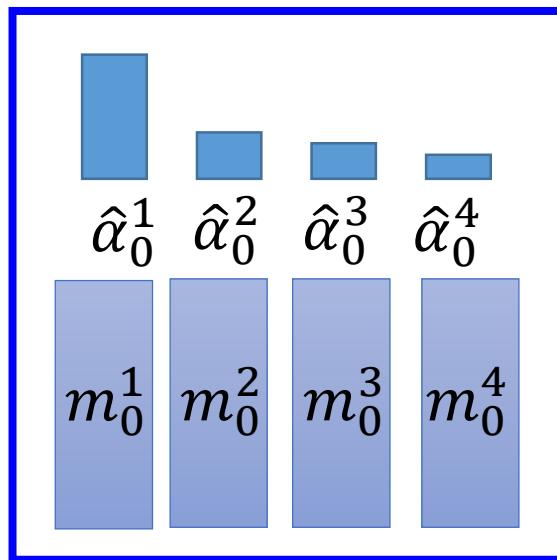
Neural Turing Machine



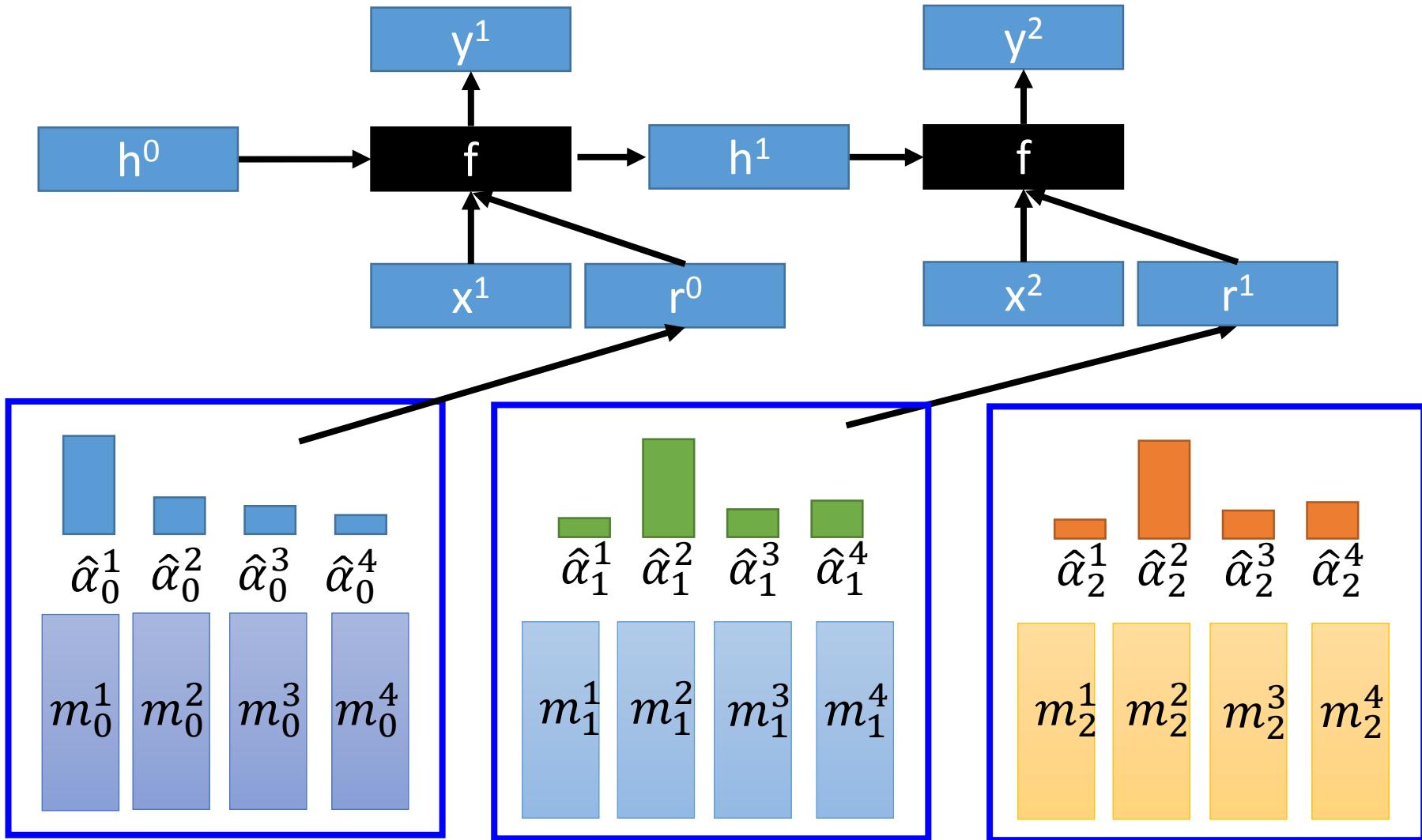
Neural Turing Machine

$$m_1^i = m_0^i - \hat{\alpha}_1^i e^1 \odot m_0^i + \hat{\alpha}_1^i a^1$$

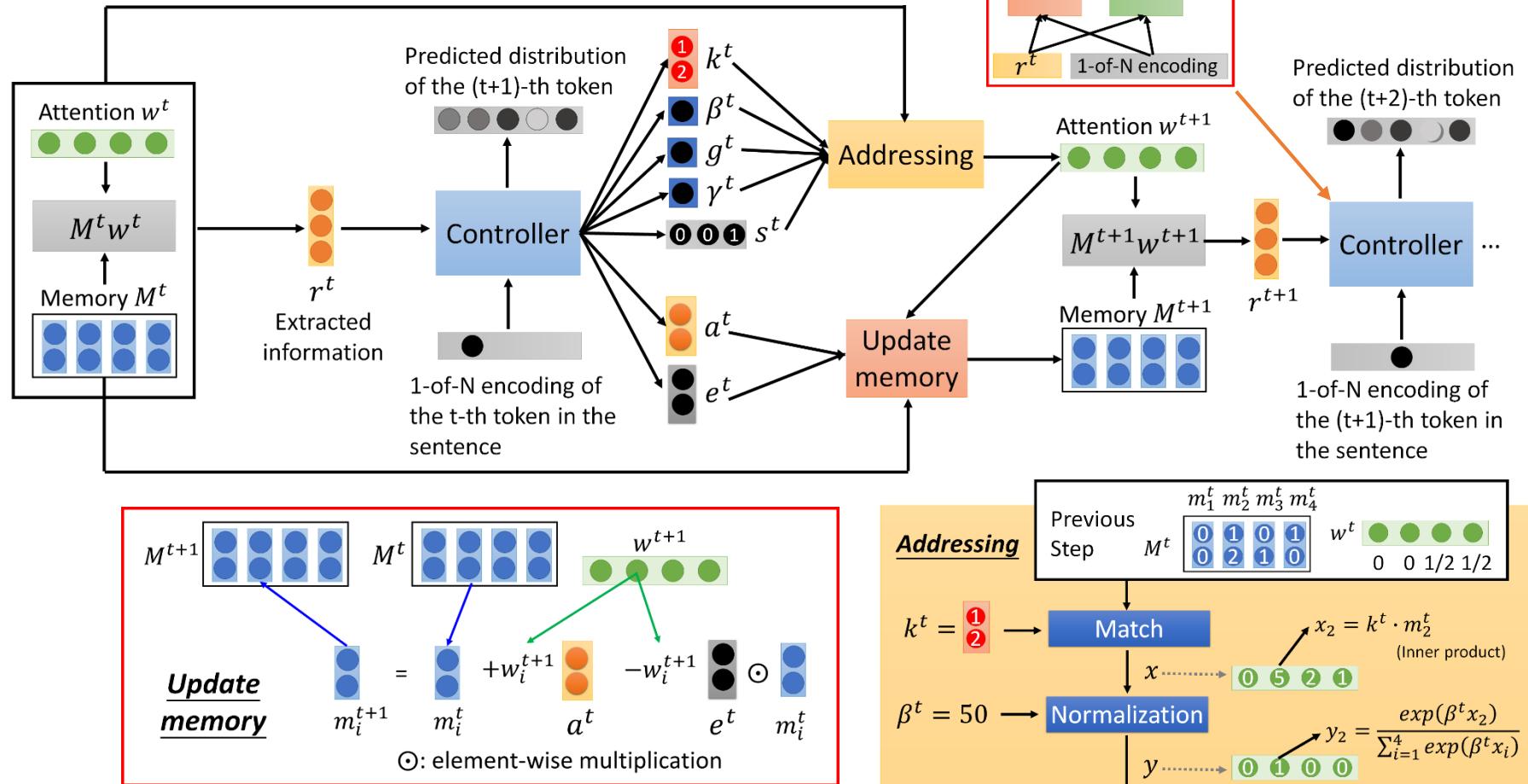
(element-wise)



Neural Turing Machine

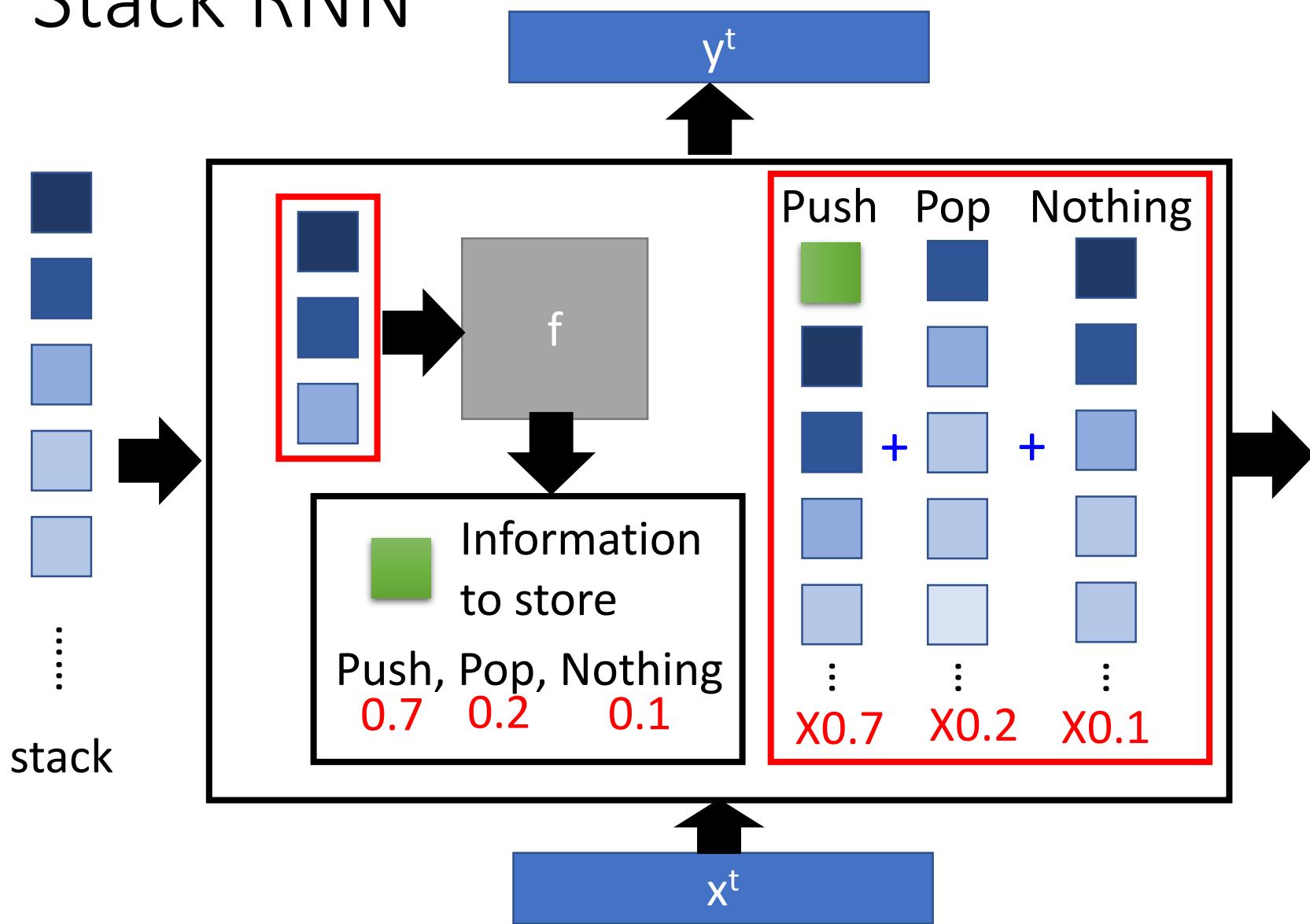


Neural Turing Machine for LM



Wei-Jen Ko, Bo-Hsiang Tseng, Hung-yi Lee,
 “Recurrent Neural Network based Language
 Modeling with Controllable External Memory”,
 ICASSP, 2017

Stack RNN



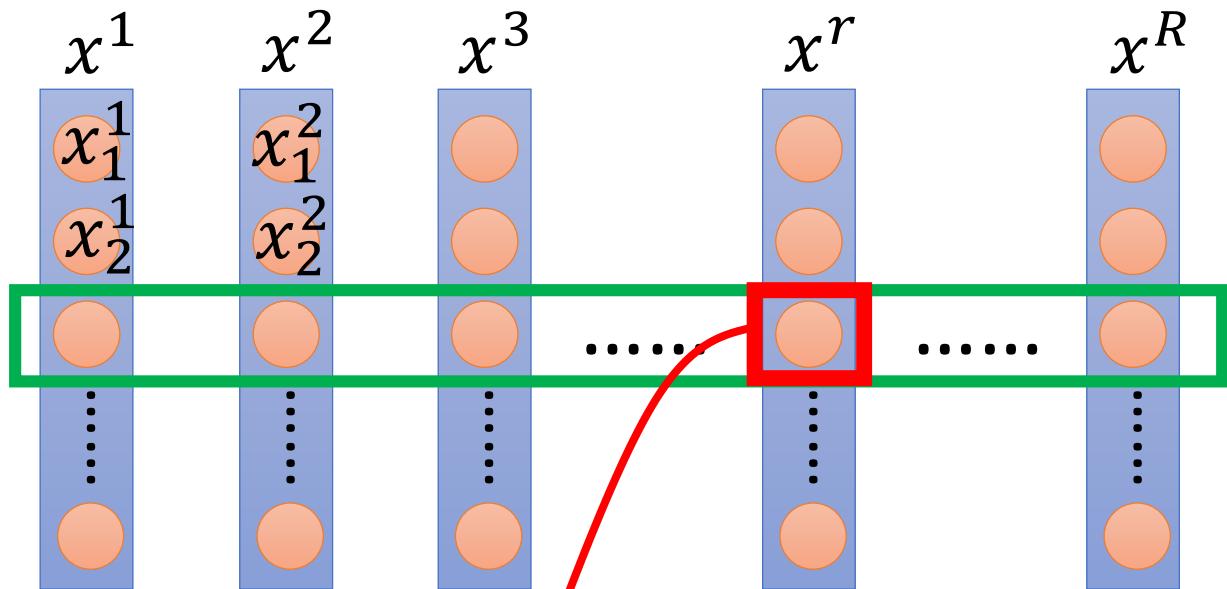
Tips for Training Deep Network

Output

- Training Strategy: Batch Normalization
- Activation Function: SELU
- Network Structure: Highway Network

Batch Normalization

Feature Scaling



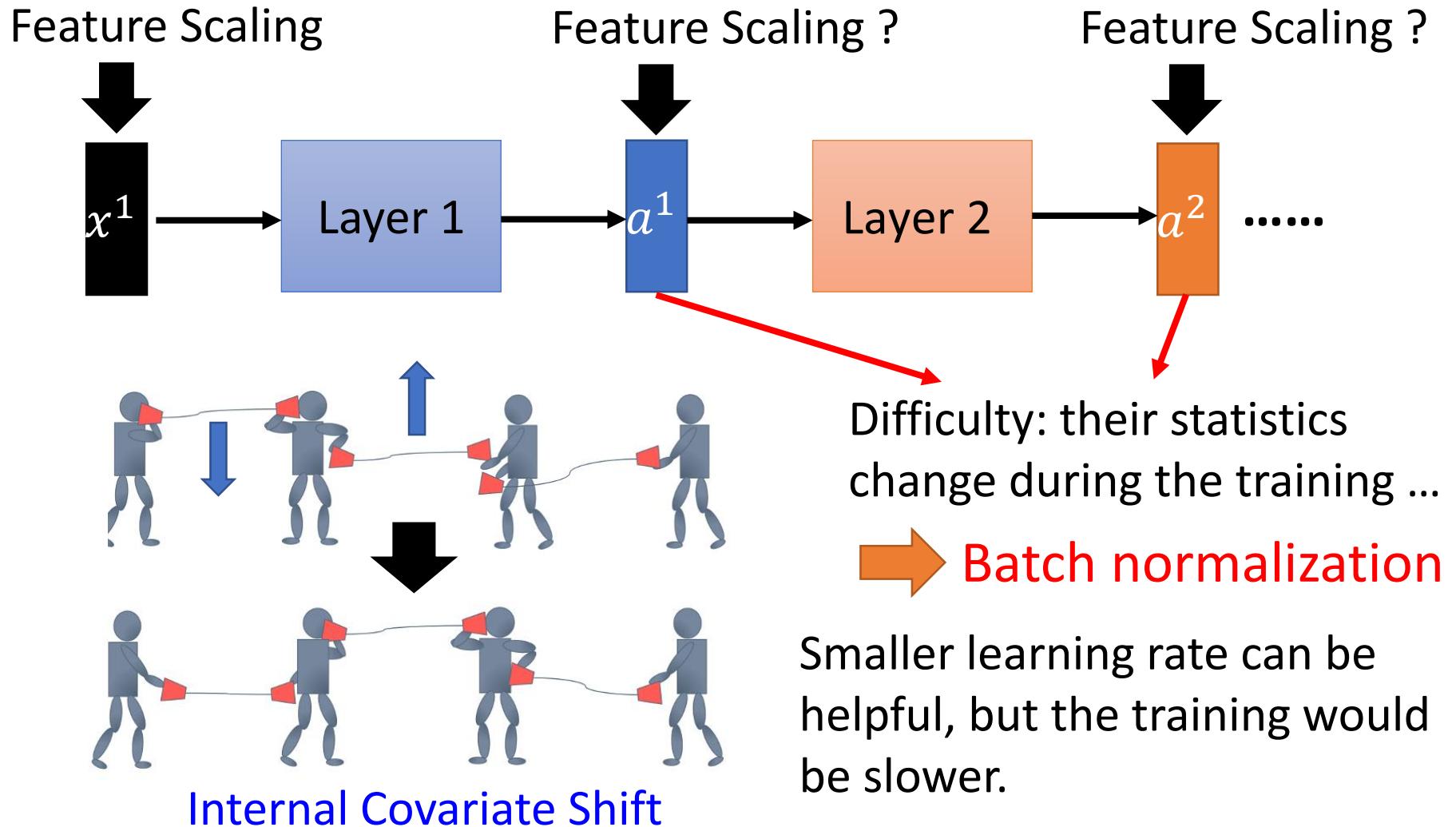
$$x_i^r \leftarrow \frac{x_i^r - m_i}{\sigma_i}$$

The means of all dimensions are 0,
and the variances are all 1

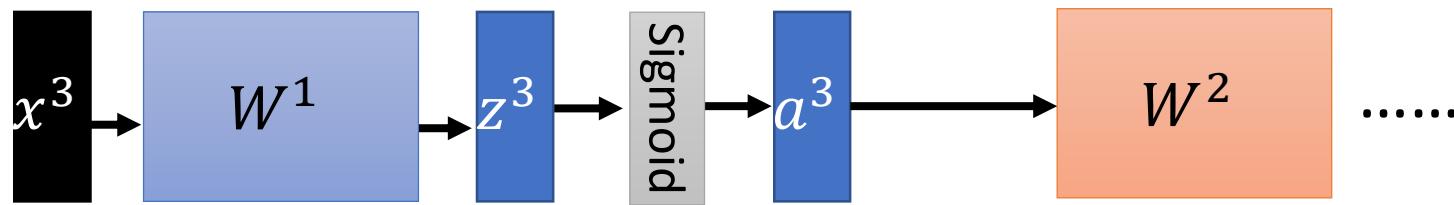
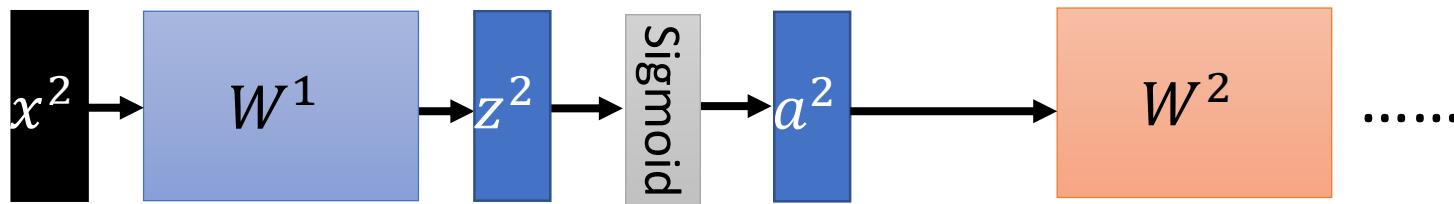
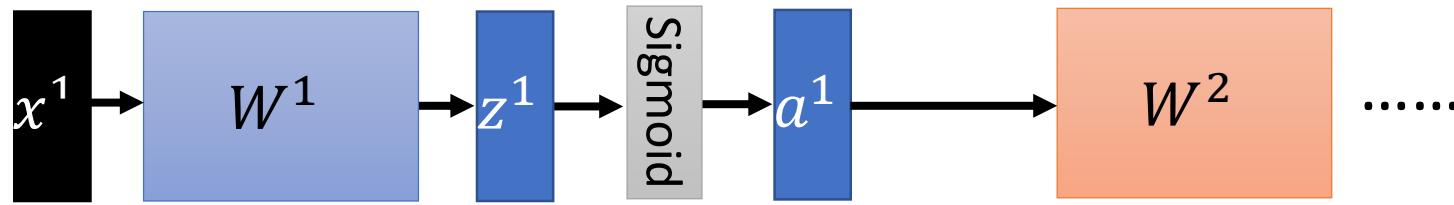
In general, gradient descent converges much faster
with feature scaling than without it.

For each dimension i:
mean: m_i
standard deviation: σ_i

How about Hidden Layer?



Batch



Batch

$$\begin{matrix} z^1 & z^2 & z^3 \end{matrix} = \begin{matrix} W^1 \\ x^1 & x^2 & x^3 \end{matrix}$$

Batch normalization



$$\mu = \frac{1}{3} \sum_{i=1}^3 z^i$$

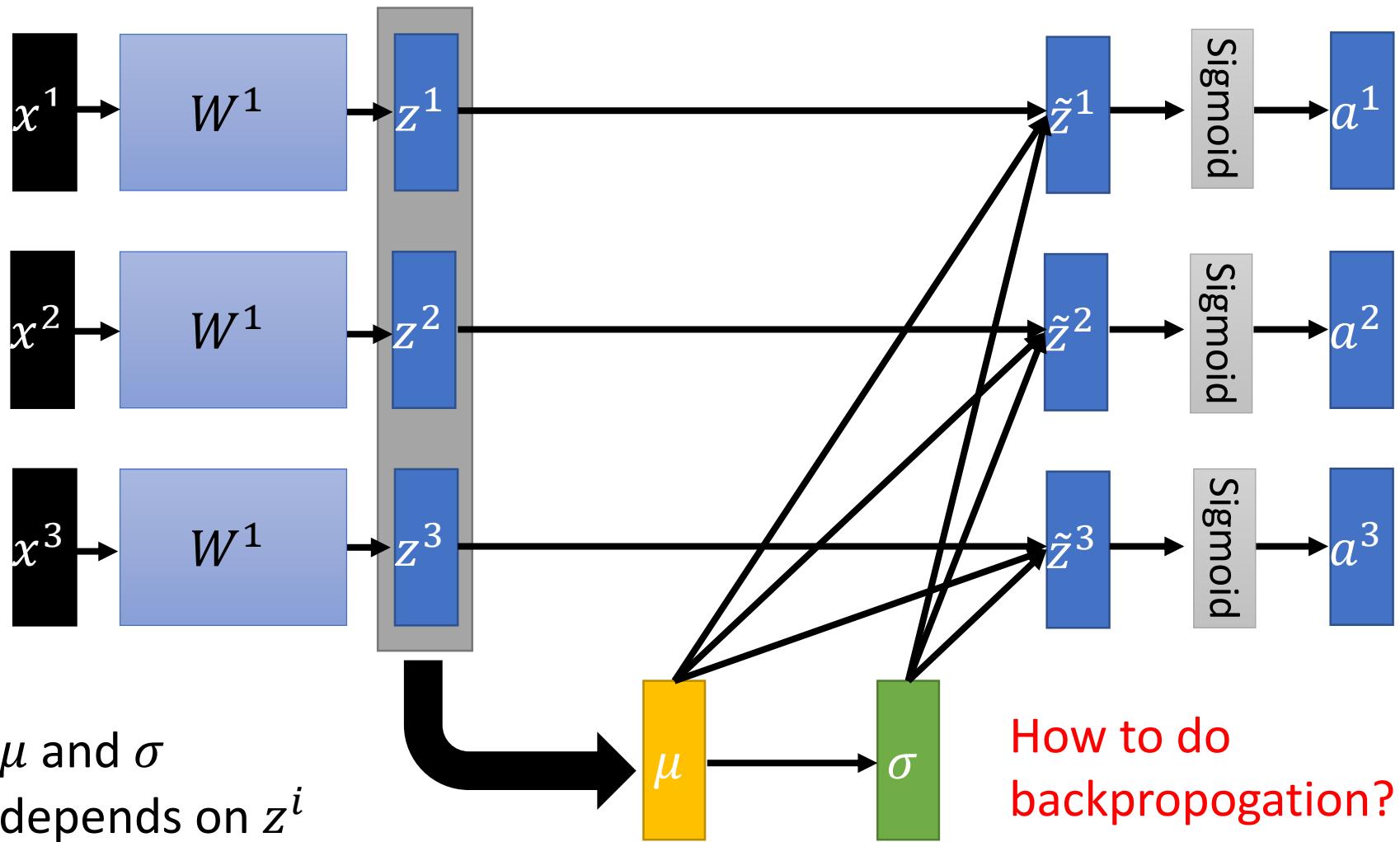
$$\sigma = \sqrt{\frac{1}{3} \sum_{i=1}^3 (z^i - \mu)^2}$$

μ and σ
depends on z^i

Note: Batch normalization
cannot be applied on
small batch.

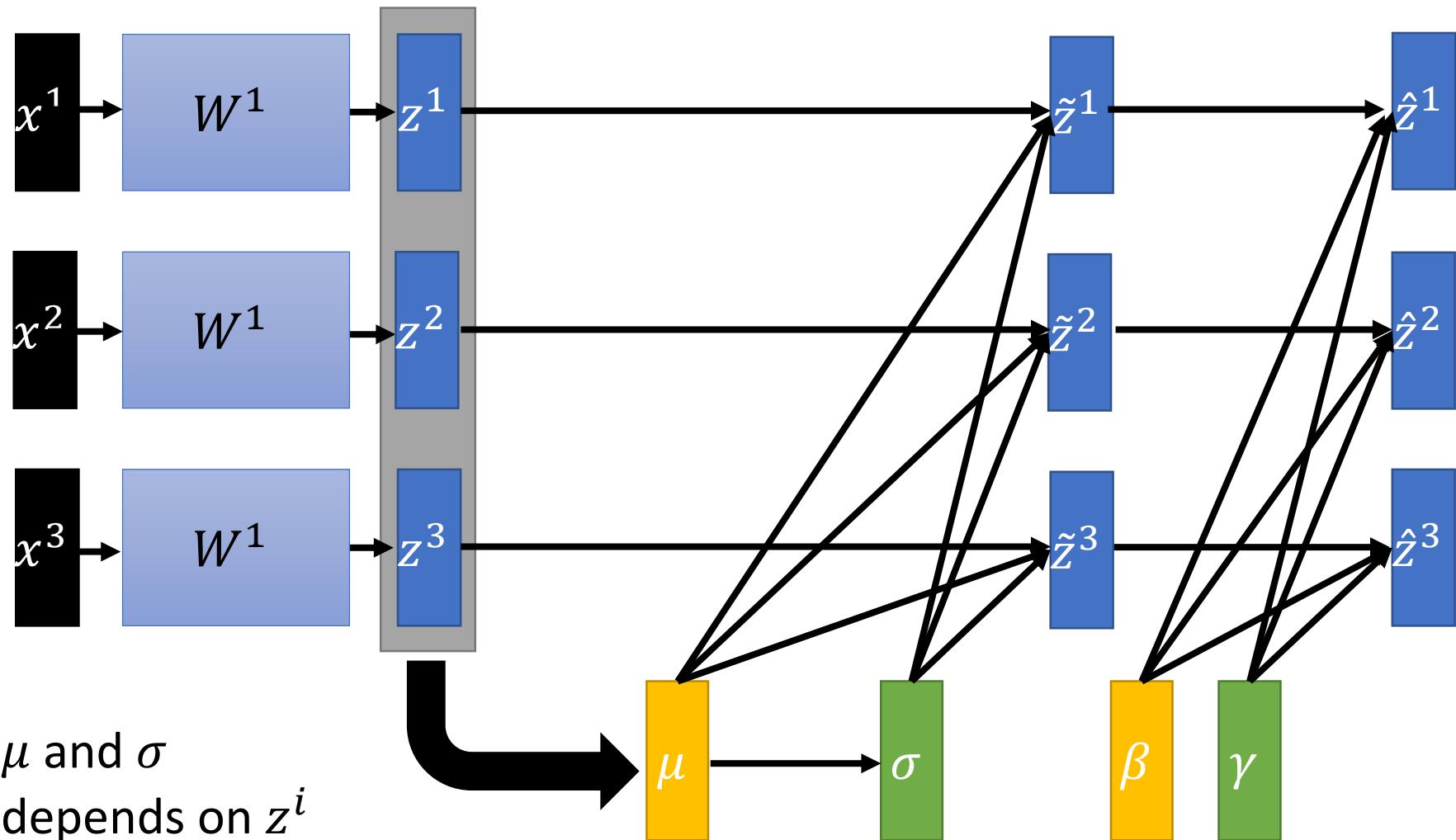
Batch normalization

$$\tilde{z}^i = \frac{z^i - \mu}{\sigma}$$



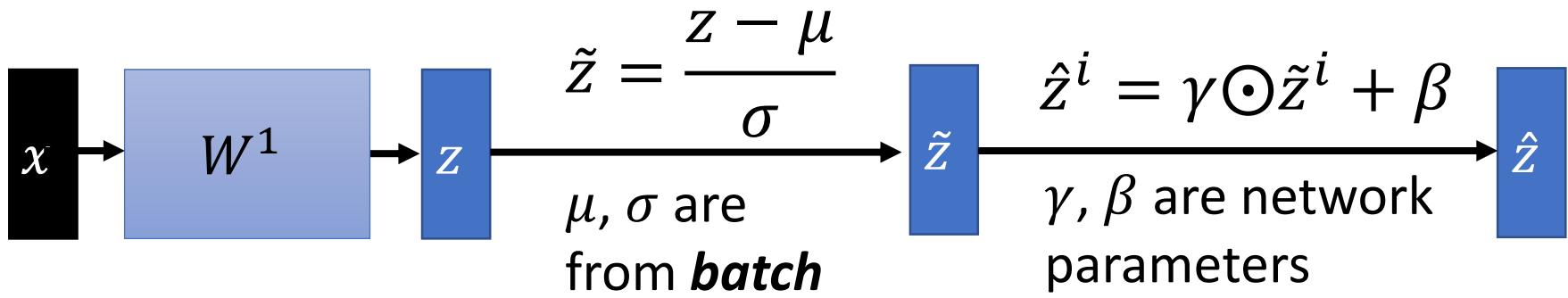
$$\tilde{z}^i = \frac{z^i - \mu}{\sigma}$$

$$\hat{z}^i = \gamma \odot \tilde{z}^i + \beta$$



Batch normalization

- At testing stage:



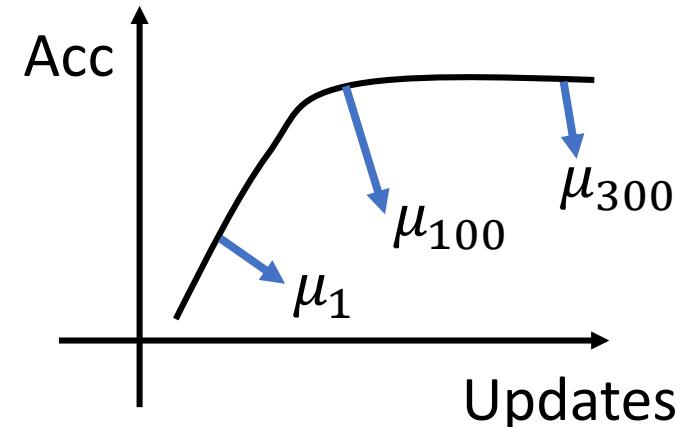
We do not have batch at testing stage.

Ideal solution:

Computing μ and σ using the whole training dataset.

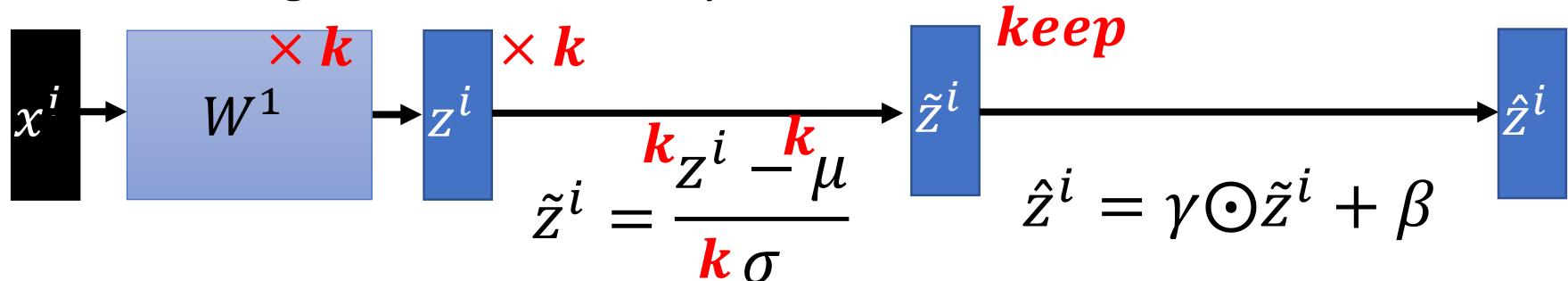
Practical solution:

Computing the moving average of μ and σ of the batches during training.



Batch normalization - Benefit

- BN reduces training times, and make very deep net trainable.
 - Because of less Covariate Shift, we can use larger learning rates.
 - Less exploding/vanishing gradients
 - Especially effective for sigmoid, tanh, etc.
- Learning is less affected by initialization.



- BN reduces the demand for regularization.

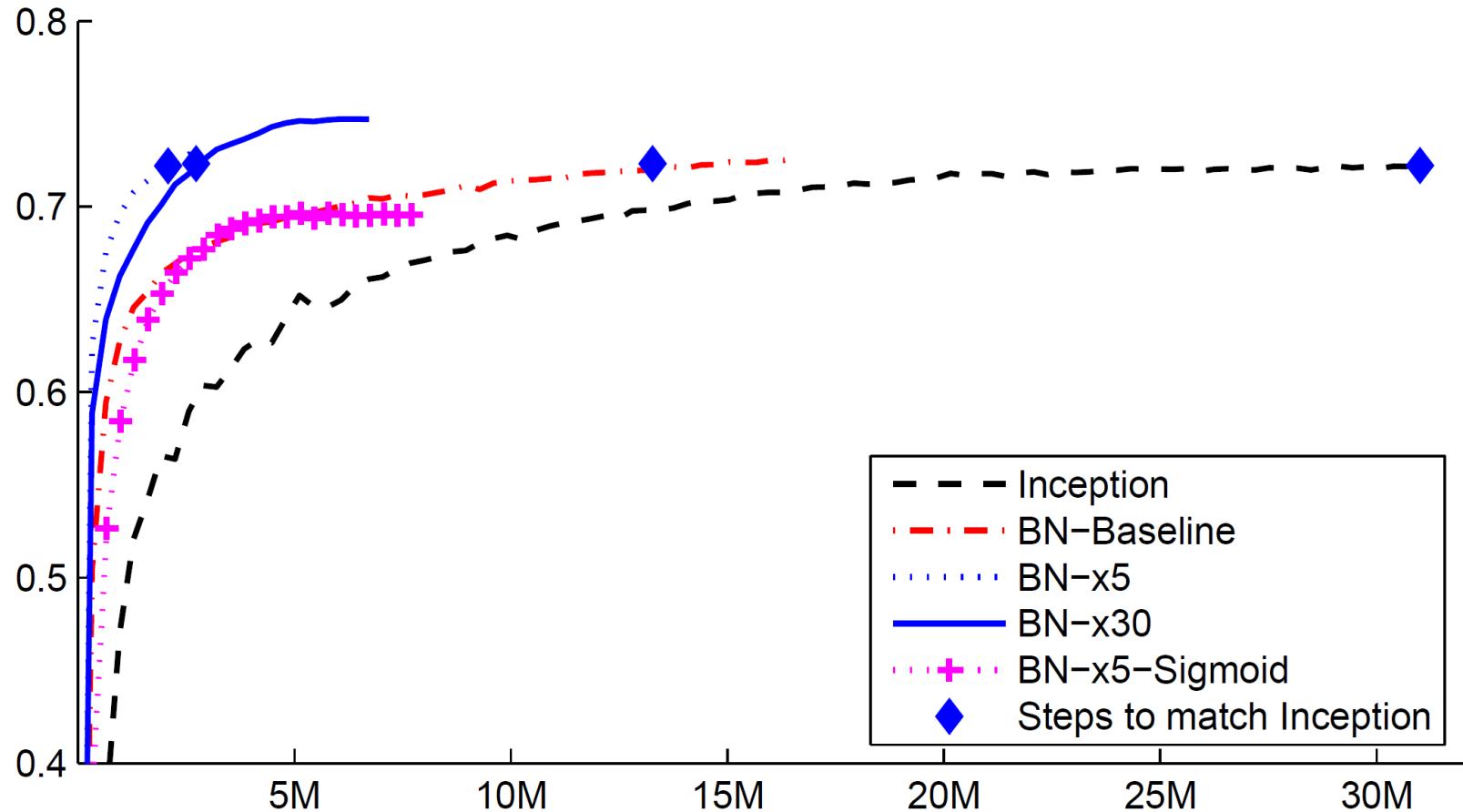


Figure 2: *Single crop validation accuracy of Inception and its batch-normalized variants, vs. the number of training steps.*

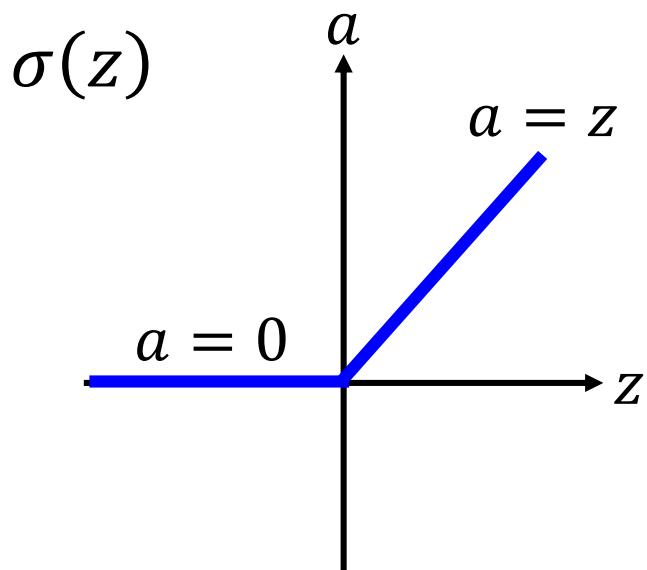
To learn more

- Batch Renormalization
- Layer Normalization
- Instance Normalization
- Weight Normalization
- Spectrum Normalization

Activation Function: SELU

ReLU

- Rectified Linear Unit (ReLU)

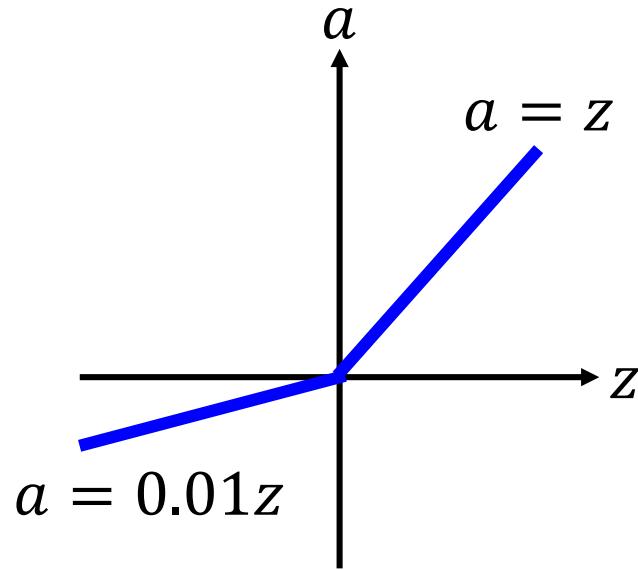


Reason:

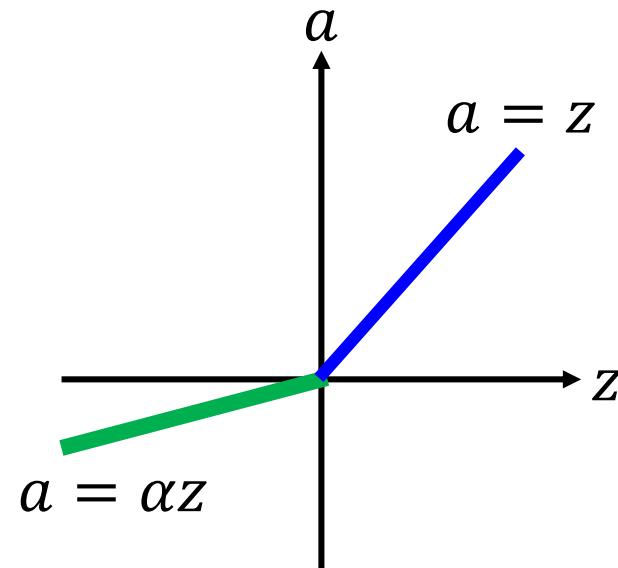
1. Fast to compute
2. Biological reason
3. Infinite sigmoid with different biases
4. Vanishing gradient problem

ReLU - variant

Leaky ReLU



Parametric ReLU



α also learned by
gradient descent

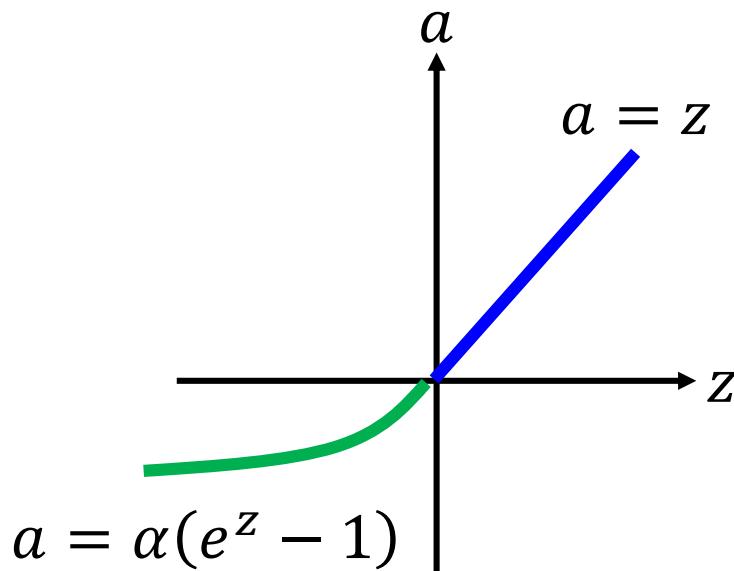
(1) Definition of scaled exponential linear units (SELU)

In [3]:

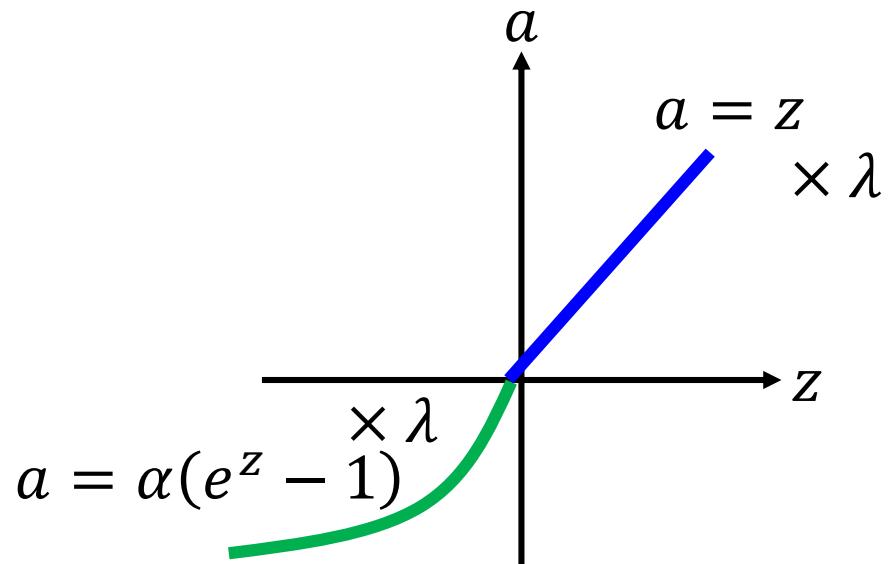
```
def selu(x):
    with ops.name_scope('elu') as scope:
        alpha = 1.6732632423543772848170429916717
        scale = 1.0507009873554804934193349852946
        return scale*tf.where(x>=0.0, x, alpha*tf.nn.elu(x))
```

<https://github.com/bioinf-jku/SNNs>

Exponential Linear
Unit (ELU)

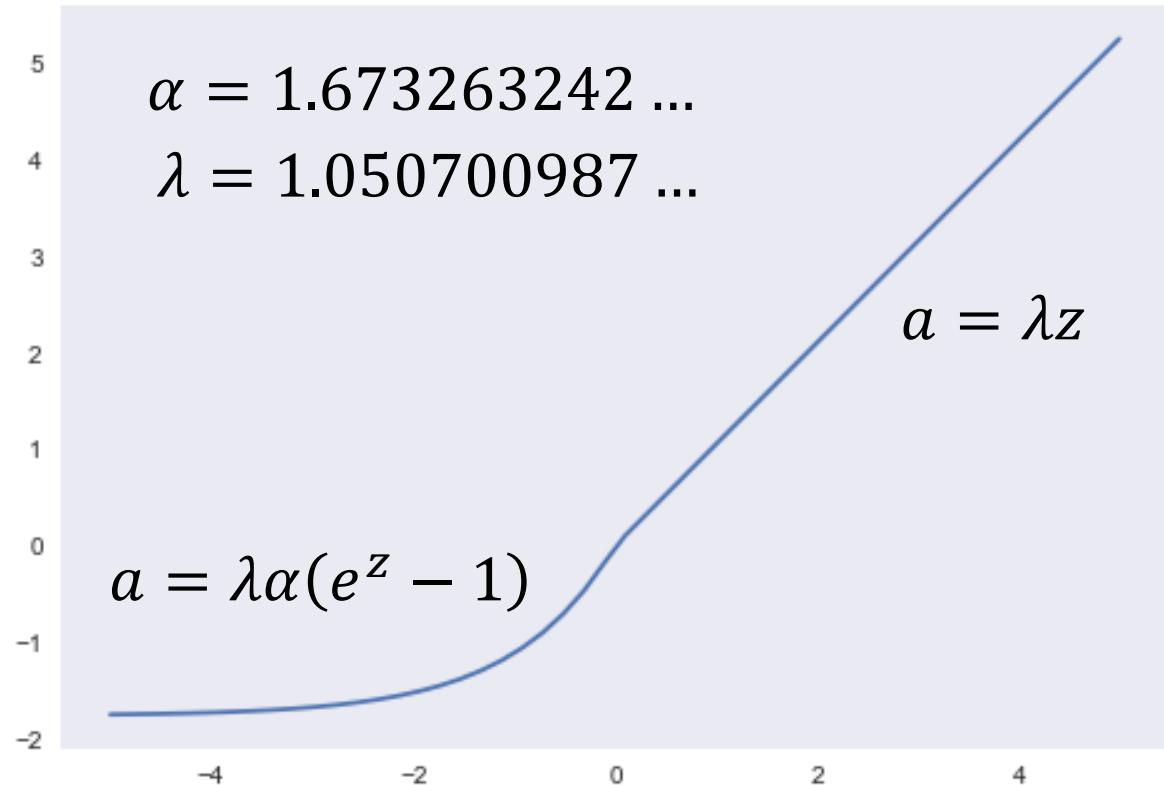


Scaled ELU (SELU)



$$\begin{aligned}\alpha &= 1.6732632423543772848170429916717 \\ \lambda &= 1.0507009873554804934193349852946\end{aligned}$$

SELU



Positive and negative values

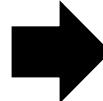
→ The whole ReLU family has this property except the original ReLU.

Saturation region



ELU also has this property

Slope larger than 1



Only SELU also has this property

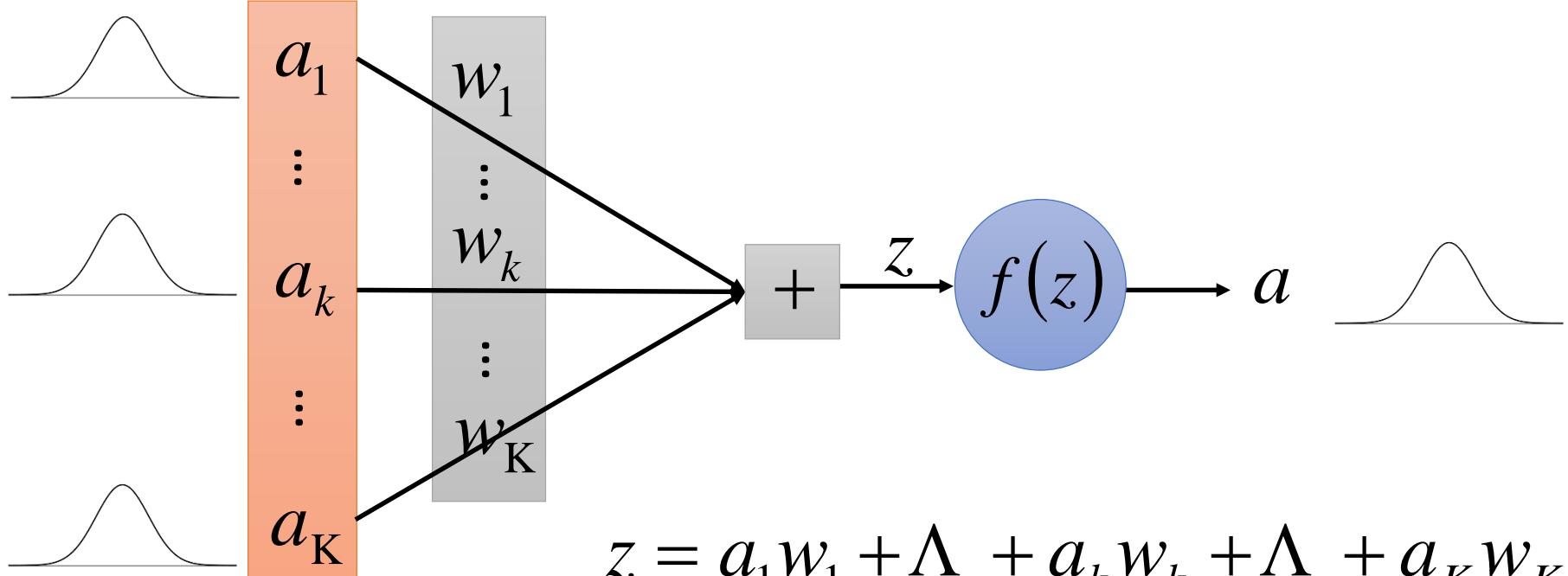
SELU

$$\mu_z = E[z]$$

$$= \sum_{k=1}^K \frac{E[a_k]}{\mu} w_k = \mu \sum_{k=1}^K w_k = \mu \cdot K \mu_w$$

$$=0 =0$$

The inputs are i.i.d random variables with mean μ and variance σ^2 .
 a_1, a_2, \dots, a_K $= 0$



Do not have to be Gaussian

SELU

$$\mu_z = 0 \quad \mu_w = 0$$

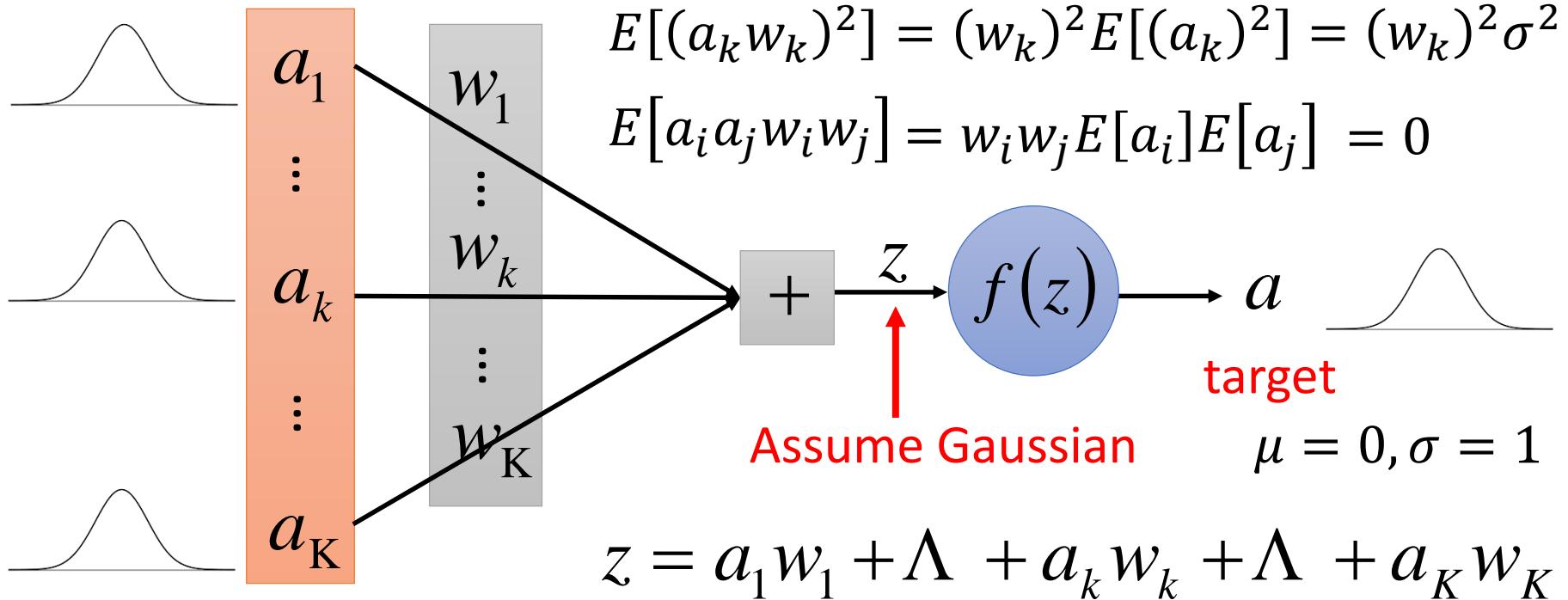
$$\sigma_z^2 = E[(z - \mu_z)^2] = E[z^2]$$

$$= E[(a_1 w_1 + a_2 w_2 + \dots)^2]$$

$$= \sum_{k=1}^K (w_k)^2 \sigma^2 = \sigma^2 \cdot K \sigma_w^2 = 1$$

$= 1 \quad = 1$

The inputs are i.i.d random variables with mean μ and variance σ^2 . $\mu = 0$ $\sigma^2 = 1$



Demo

93 頁的證明

$$\begin{aligned} & \frac{2(2x-y)(2x+y)2.911}{(\sqrt{2}\sqrt{x})\left(\sqrt{\pi\left(\frac{2x+y}{2\sqrt{x}}\right)^2+2.911^2+\frac{(2.911-1)\sqrt{\pi}(2x+y)}{\sqrt{2}\sqrt{x}}}\right)}\sqrt{\pi}-0.0003= \\ & (3x-y)+\left(\frac{(\sqrt{2}\sqrt{x}2.911)(x-y)(x+y)}{\left(\sqrt{\pi(x+y)^2+2\cdot2.911^2x+(2.911-1)(x+y)\sqrt{\pi}}\right)(\sqrt{2}\sqrt{x})}\right)- \\ & \frac{2(2x-y)(2x+y)(\sqrt{2}\sqrt{x}2.911)}{(\sqrt{2}\sqrt{x})\left(\sqrt{\pi(2x+y)^2+2\cdot2.911^2x+(2.911-1)(2x+y)\sqrt{\pi}}\right)}\sqrt{\pi}-0.0003= \\ & (3x-y)+2.911\left(\frac{(x-y)(x+y)}{(2.911-1)(x+y)+\sqrt{(x+y)^2+\frac{2\cdot2.911^2x}{\pi}}}-\right. \\ & \left.\frac{2(2x-y)(2x+y)}{(2.911-1)(2x+y)+\sqrt{(2x+y)^2+\frac{2\cdot2.911^2x}{\pi}}}\right)-0.0003\geq \\ & (3x-y)+2.911\left(\frac{(x-y)(x+y)}{(2.911-1)(x+y)+\sqrt{\left(\frac{2.911^2}{\pi}\right)^2+(x+y)^2+\frac{2\cdot2.911^2x}{\pi}+\frac{2\cdot2.911^2y}{\pi}}}-\right. \\ & \left.\frac{2(2x-y)(2x+y)}{(2.911-1)(2x+y)+\sqrt{(2x+y)^2+\frac{2\cdot2.911^2x}{\pi}}}\right)-0.0003= \\ & (3x-y)+2.911\left(\frac{(x-y)(x+y)}{(2.911-1)(x+y)+\sqrt{\left(x+y+\frac{2.911^2}{\pi}\right)^2}}-\right. \\ & \left.\frac{2(2x-y)(2x+y)}{(2.911-1)(2x+y)}\right)-0.0003= \\ & (2.911-1)(2x+y) \end{aligned}$$

$$\begin{aligned} & (3x-y)+2.911 \\ & (3x-y)+\frac{(x-y)}{(x+y)} \\ & (3x-y)+\frac{(x-y)}{(x+y)} \\ & (-2(2x-y)2.911) \\ & \left((x+y)+\frac{2.911}{\pi}\right) \\ & (x-y)(x+y)\left(\left((x+y)+\frac{2.911}{\pi}\right)\right) \end{aligned}$$

Source of joke:

<https://zhuanlan.zhihu.com/p/27336839>

SELU is actually more general.

Andrej Karpathy 
@karpathy

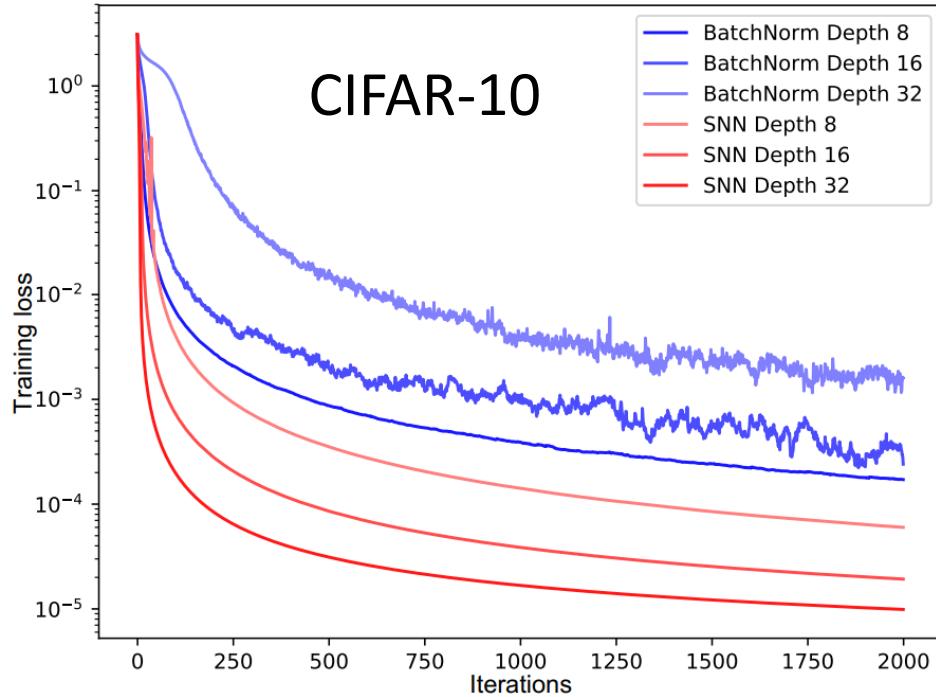
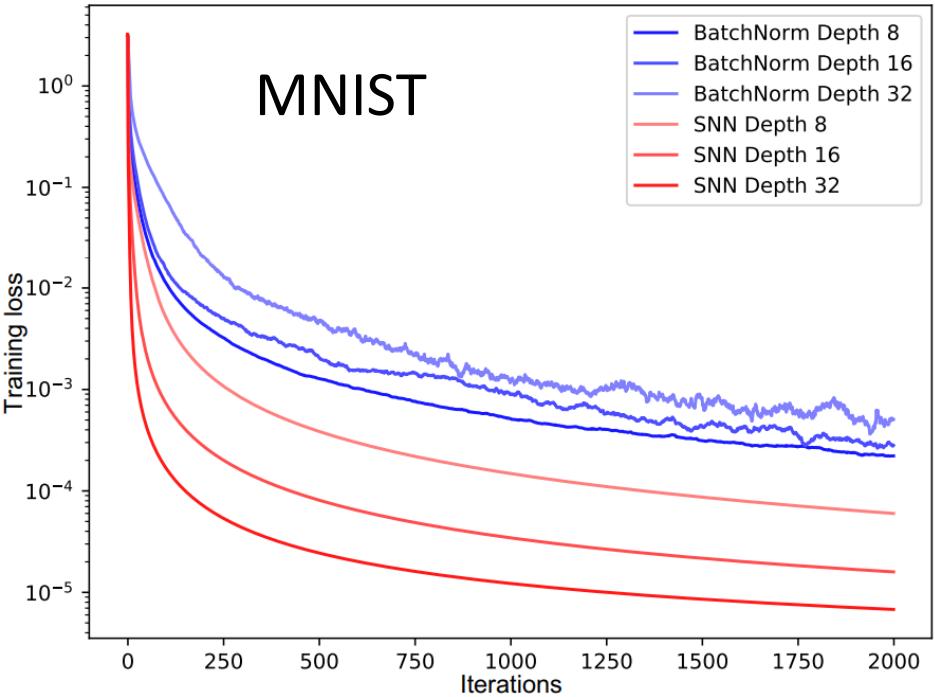
Following

maybe it's all generated by a char-rnn. I suspect we will never know.

RETWEETS LIKES
4 41

2:54 AM - 10 Jun 2017

5 4 41



FNN method comparison

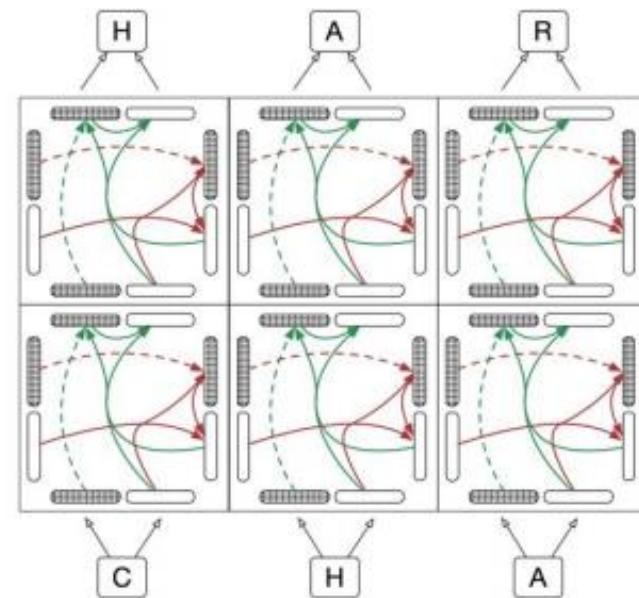
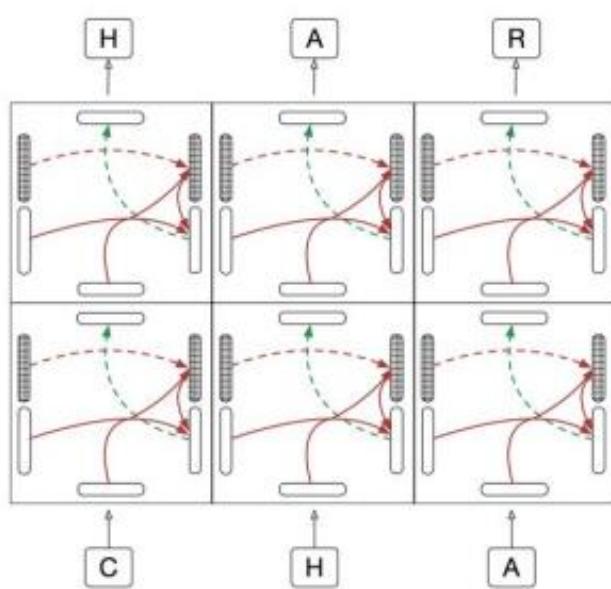
Method	avg. rank diff.	p-value
SNN	-0.756	
MSRAinit	-0.240*	2.7e-02
LayerNorm	-0.198*	1.5e-02
Highway	0.021*	1.9e-03
ResNet	0.273*	5.4e-04
WeightNorm	0.397*	7.8e-07
BatchNorm	0.504*	3.5e-06

ML method comparison

Method	avg. rank diff.	p-value
SNN	-6.7	
SVM	-6.4	5.8e-01
RandomForest	-5.9	2.1e-01
MSRAinit	-5.4*	4.5e-03
LayerNorm	-5.3	7.1e-02
Highway	-4.6*	1.7e-03
...

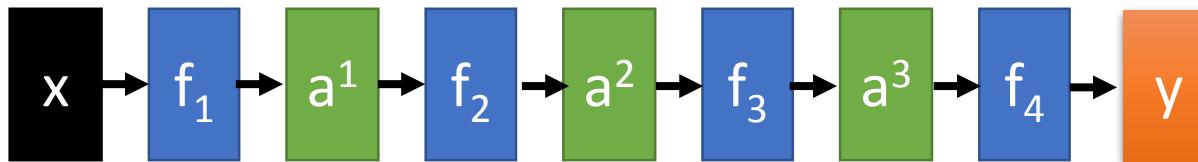
Demo

Highway Network & Grid LSTM



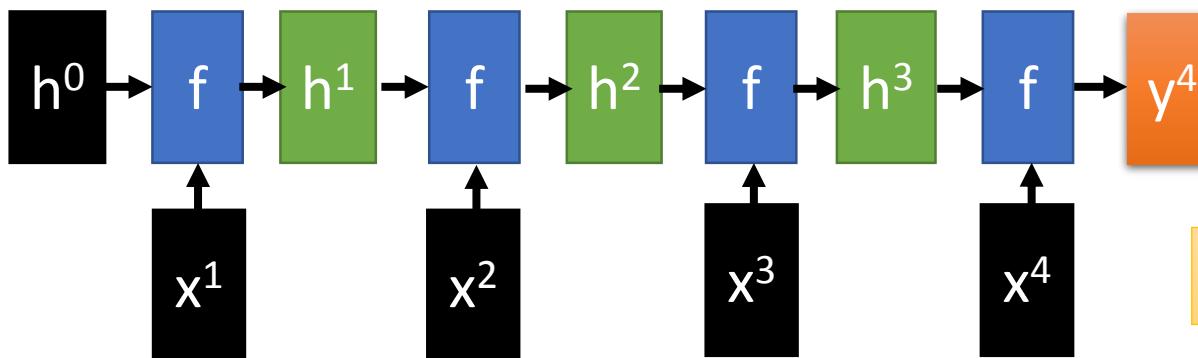
Feedforward v.s. Recurrent

1. Feedforward network does not have input at each step
2. Feedforward network has different parameters for each layer



$$a^t = f_l(a^{t-1}) = \sigma(W^t a^{t-1} + b^t)$$

t is layer



$$h^t = f(h^{t-1}, x^t) = \sigma(W^h h^{t-1} + W^i x^t + b^i)$$

t is time step

Applying gated structure in feedforward network

GRU → Highway Network

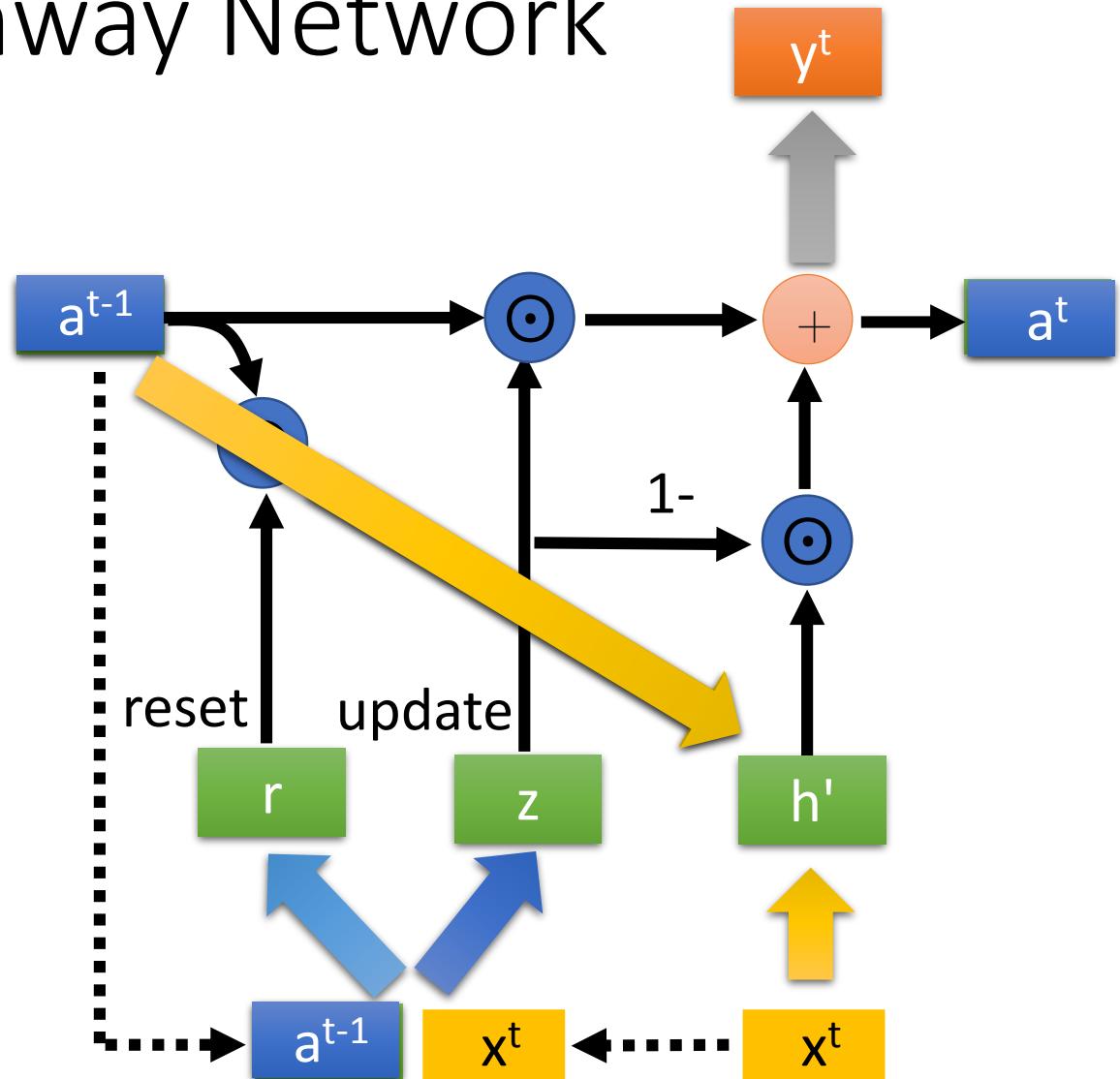
No input x^t at each step

No output y^t at each step

a^{t-1} is the output of the $(t-1)$ -th layer

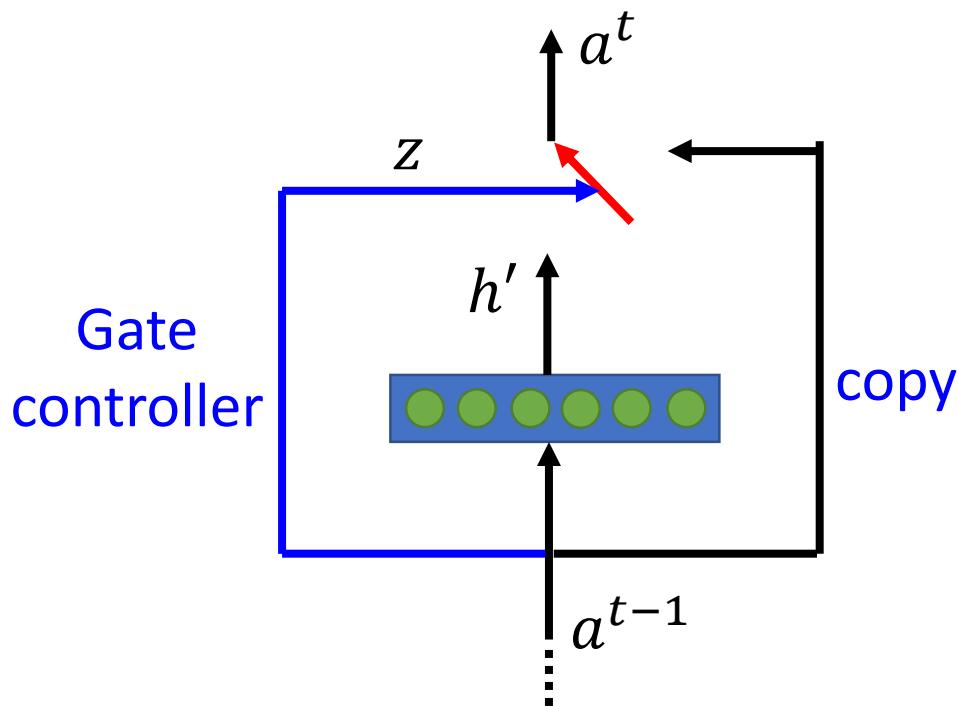
a^t is the output of the t -th layer

No reset gate



Highway Network

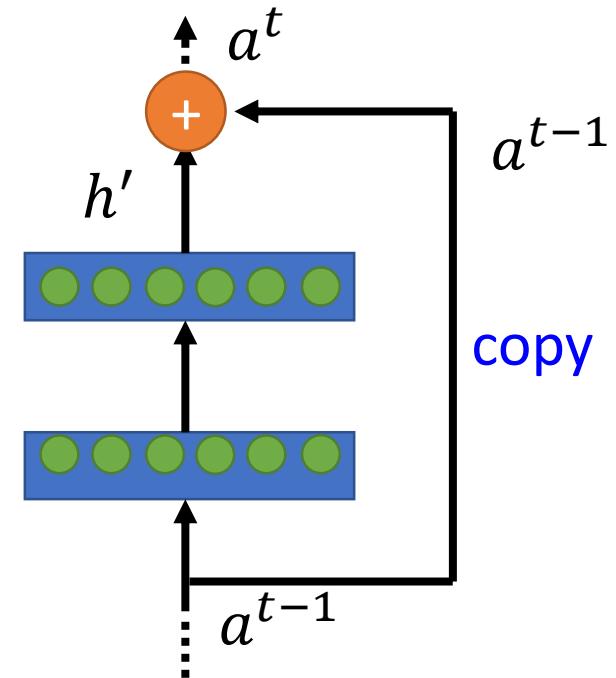
- **Highway Network**



Training Very Deep Networks
<https://arxiv.org/pdf/1507.06228v2.pdf>

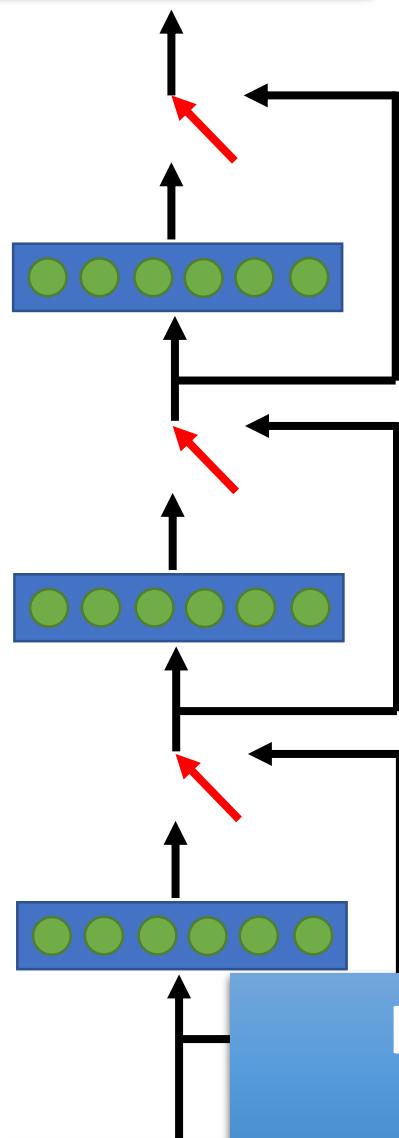
$$h' = \sigma(Wa^{t-1})$$
$$z = \sigma(W'a^{t-1})$$
$$a^t = z \odot a^{t-1} + (1 - z) \odot h$$

- **Residual Network**

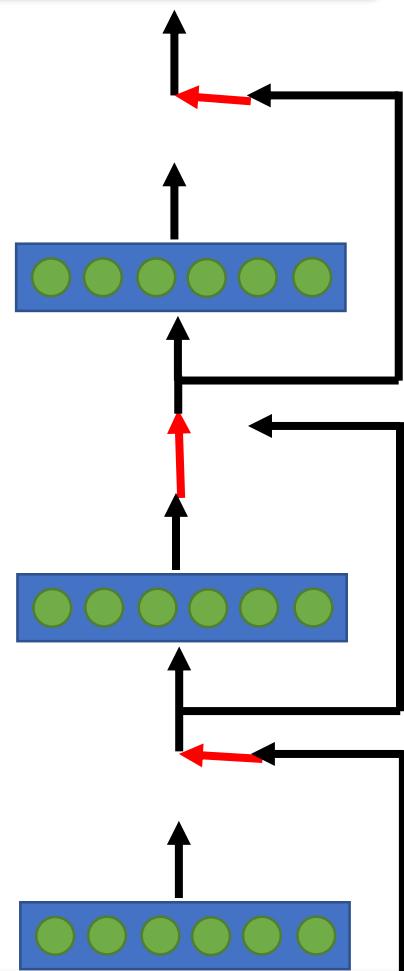


Deep Residual Learning for Image Recognition
<http://arxiv.org/abs/1512.03385>

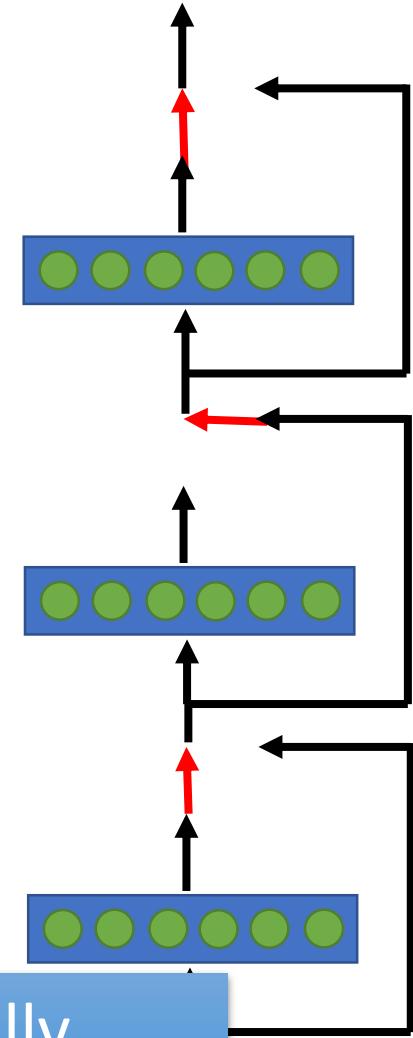
output layer



output layer



output layer



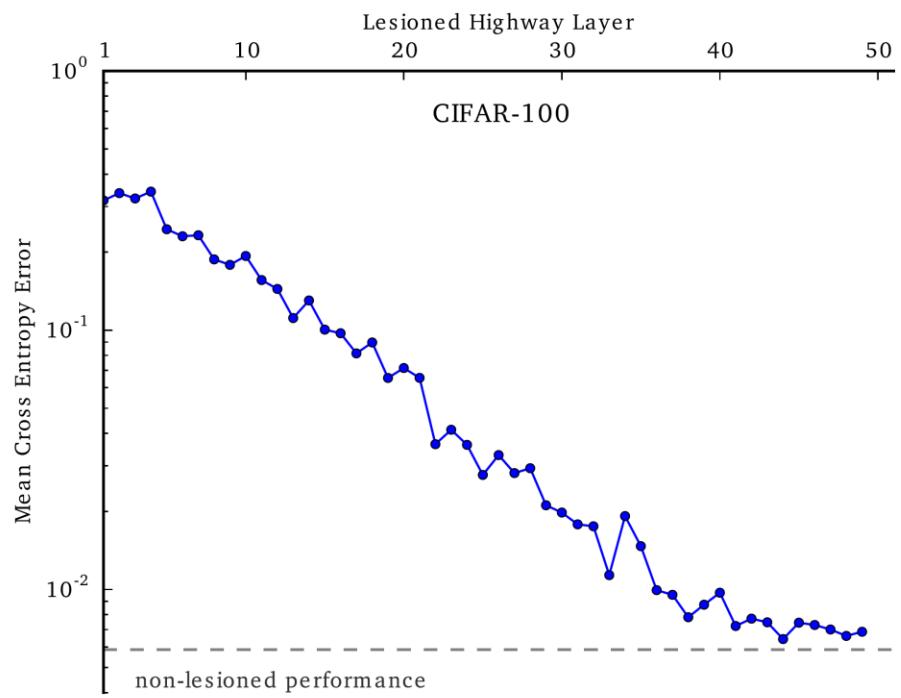
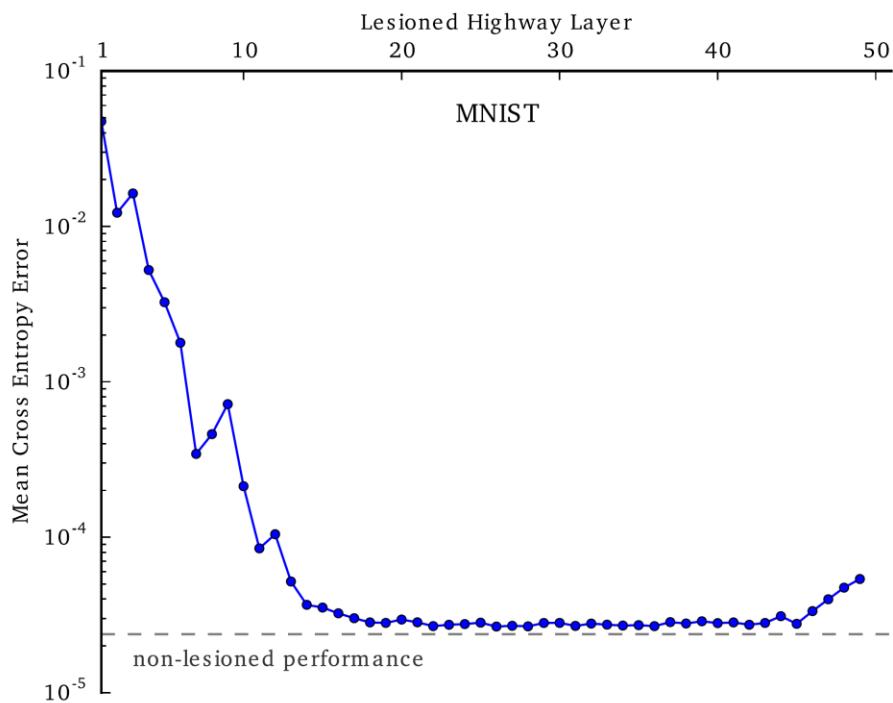
Highway Network automatically
determines the layers needed!

Input layer

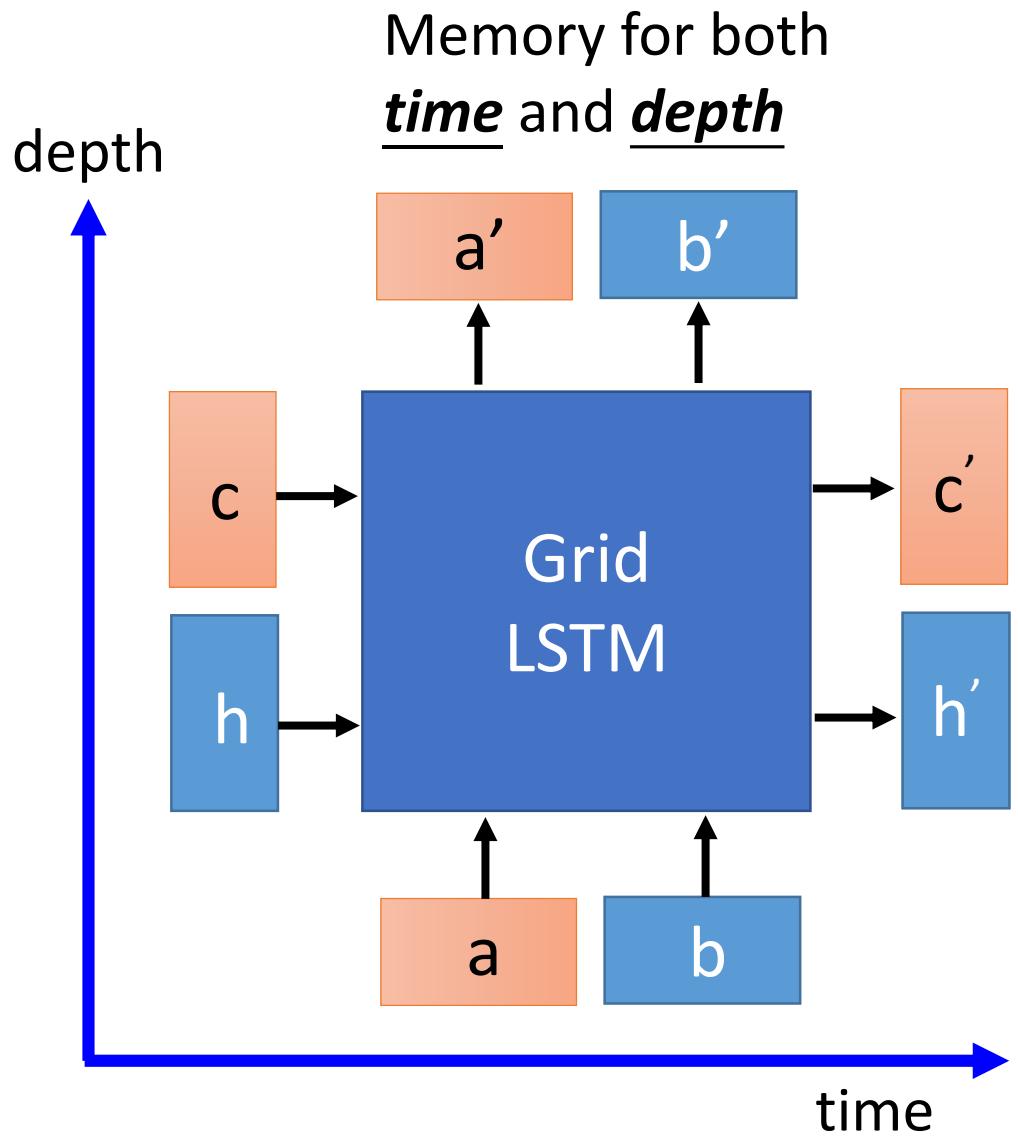
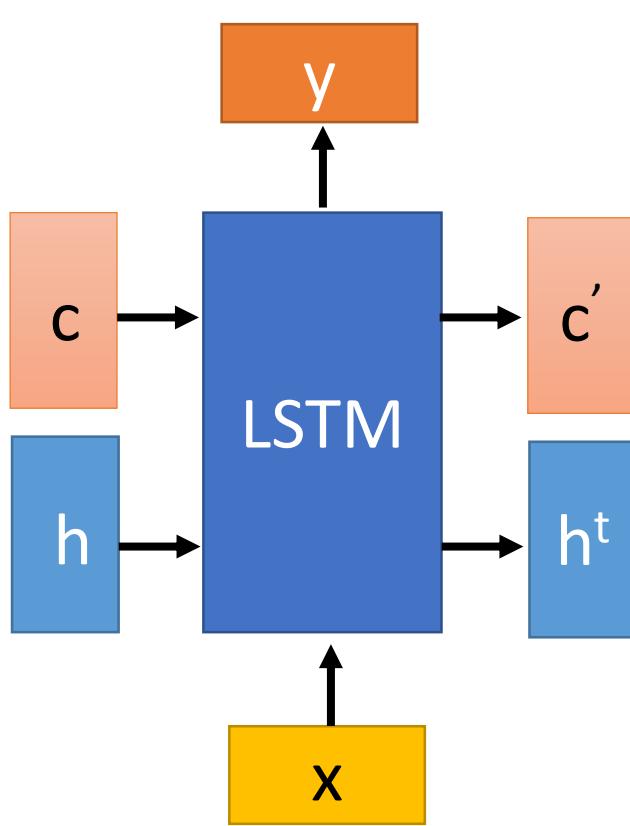
Input layer

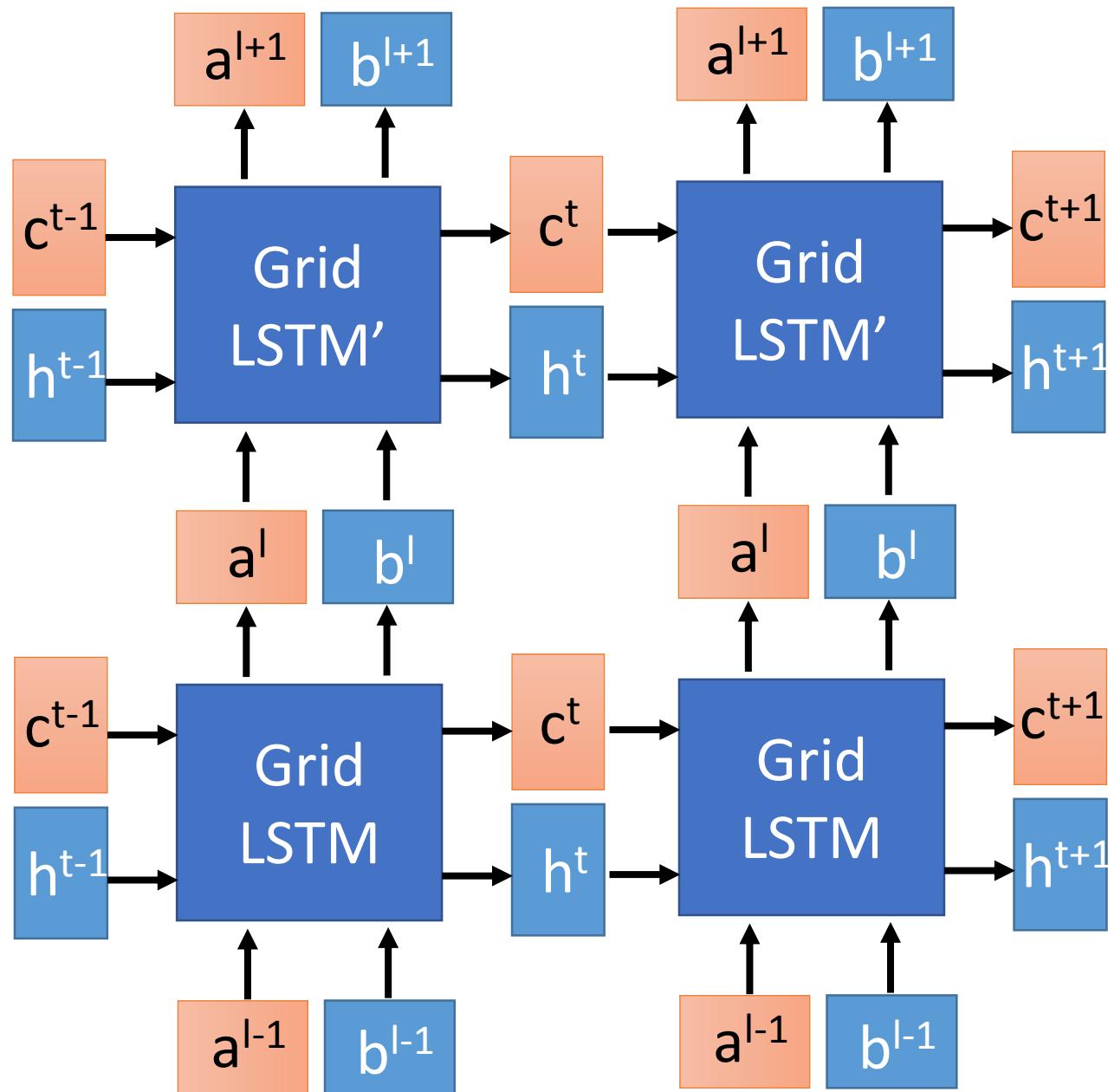
Input layer

Highway Network

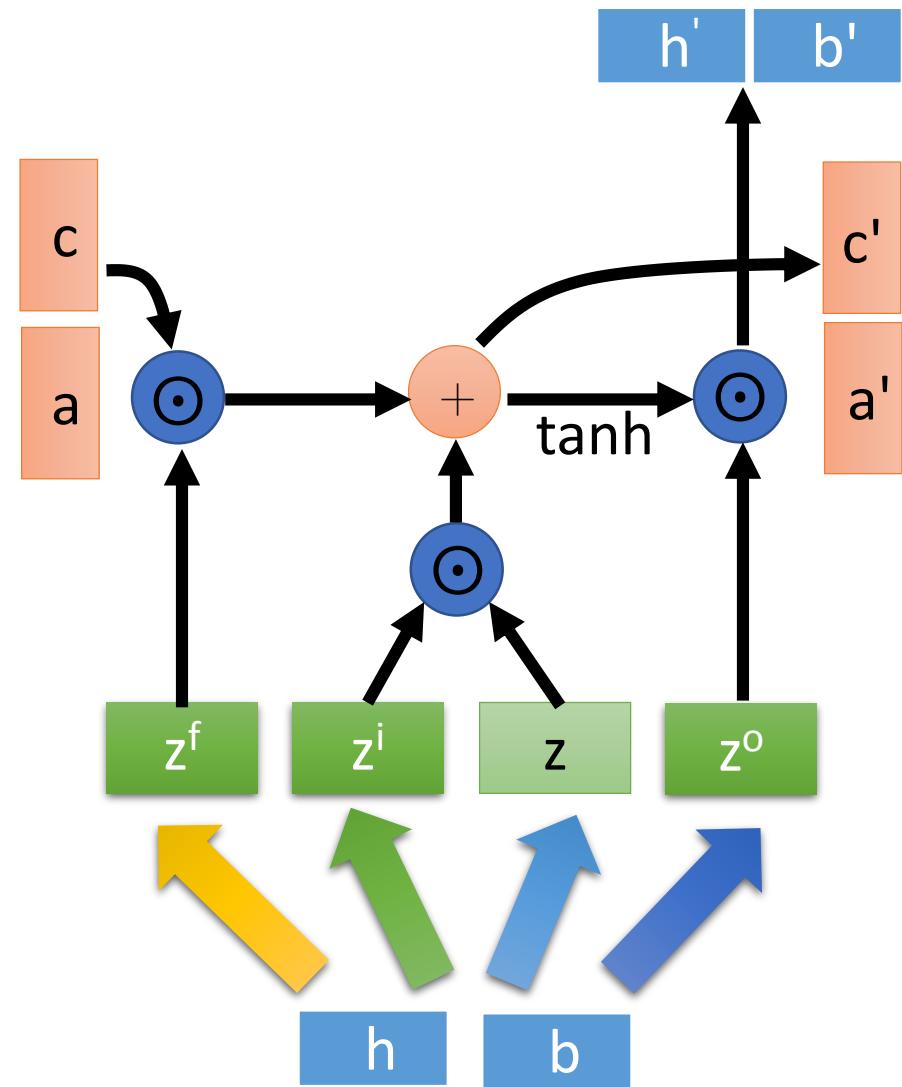
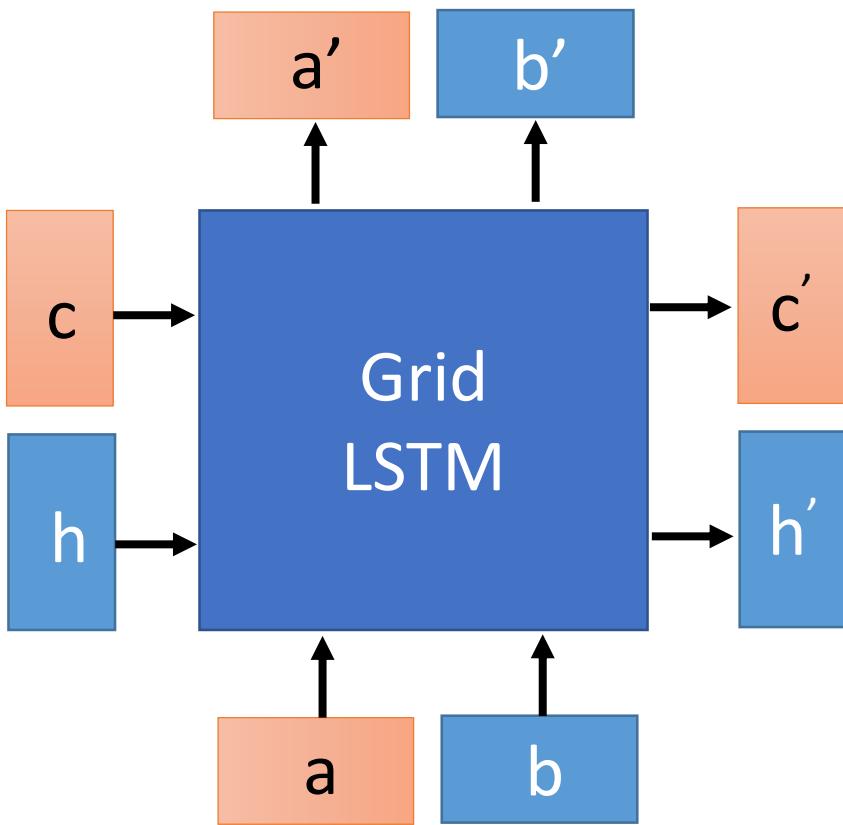


Grid LSTM

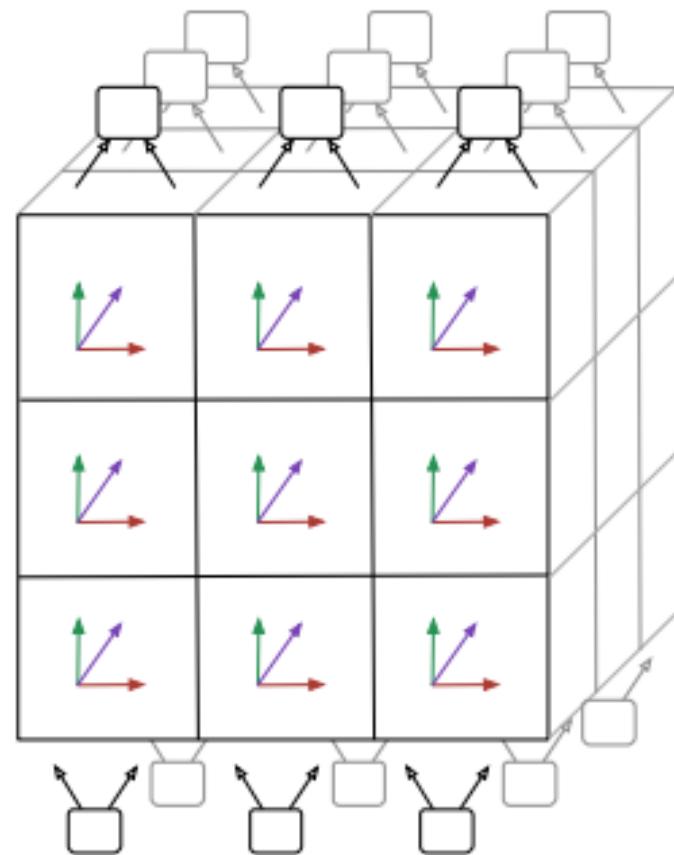
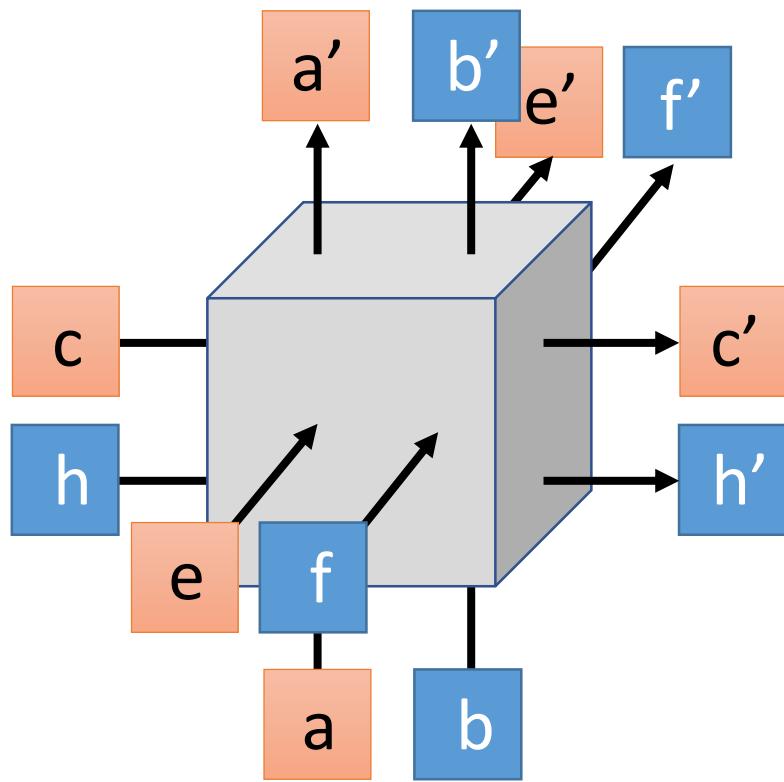




Grid LSTM



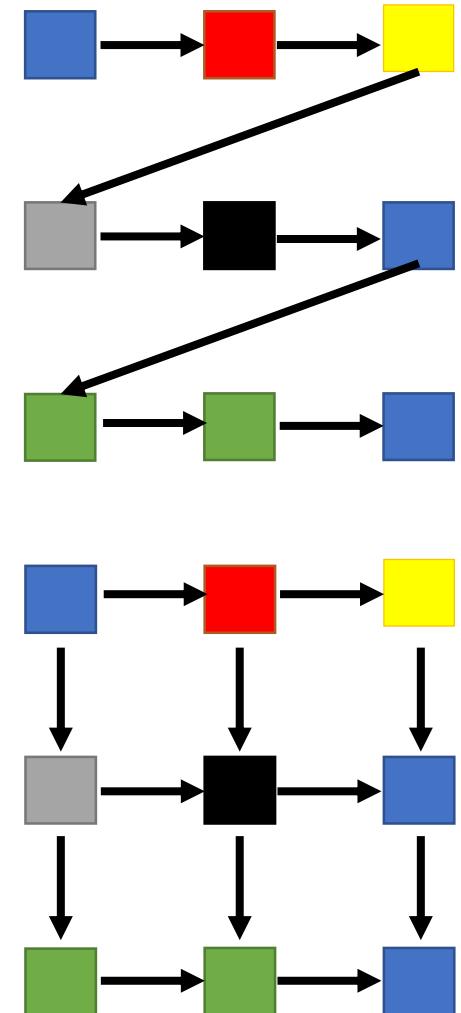
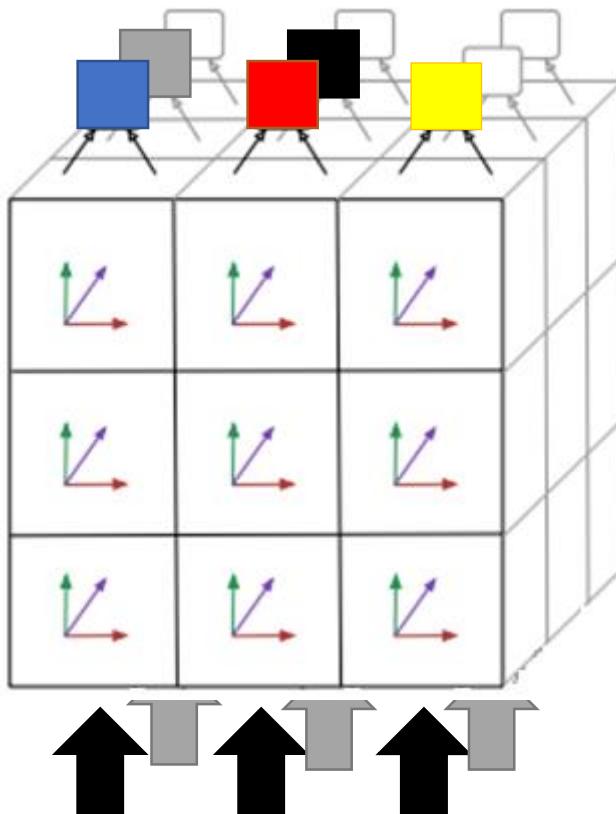
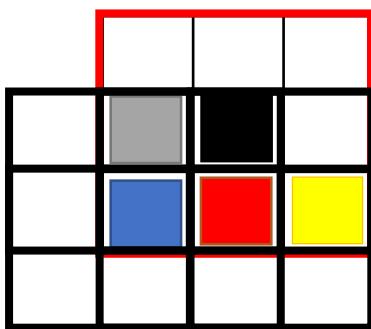
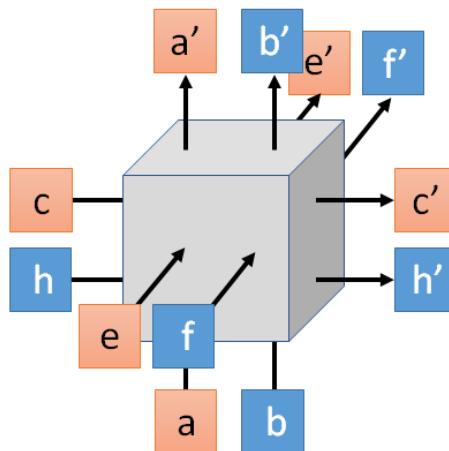
3D Grid LSTM



3D Grid LSTM

3 x 3 images

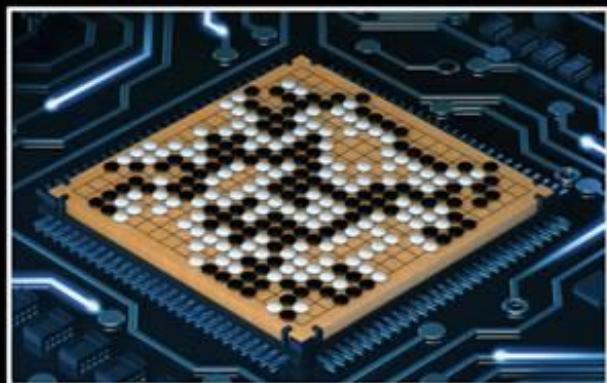
- Images are composed of pixels



Automatically Determining Hyperparameters

Source of iamge: <https://medium.com/intuitionmachine/the-brute-force-method-of-deep-learning-innovation-58b497323ae5> (Denny Britz's graphic)

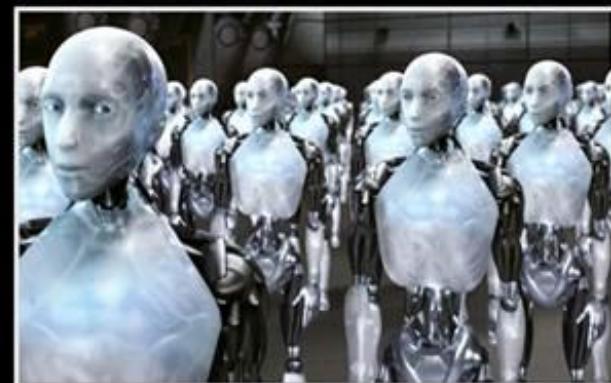
Deep Learning研究生



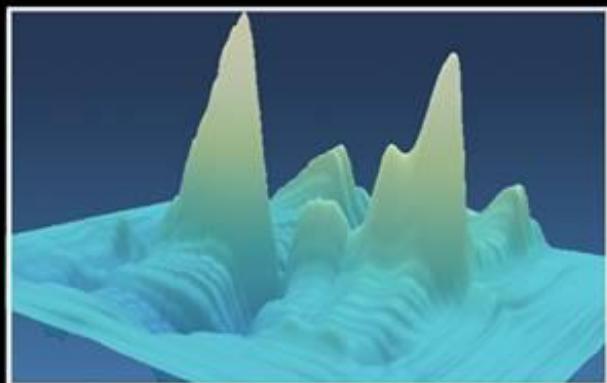
朋友覺得我在



我媽覺得我在



大眾覺得我在



指導教授覺得我在

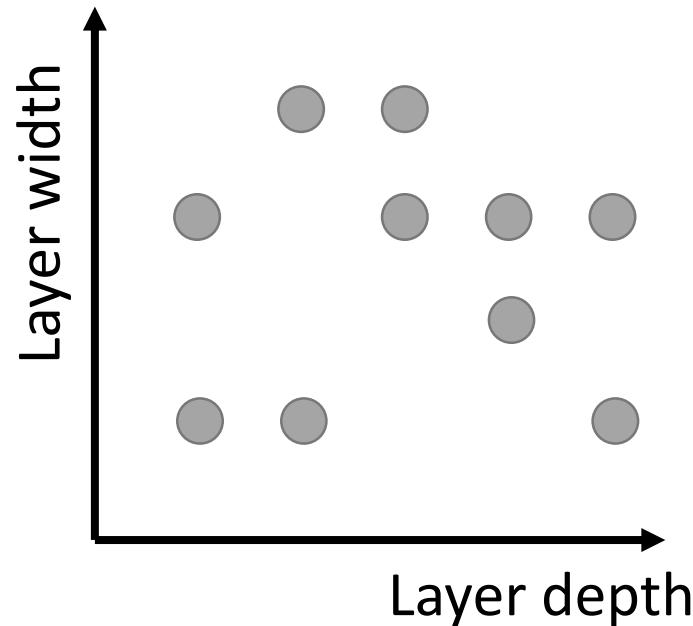
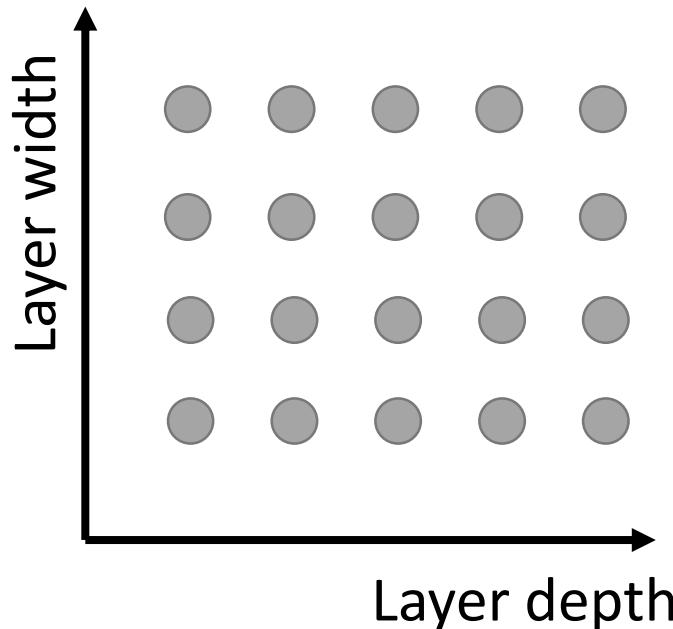


我以為我在

事實上我在

感謝 沈昇勳 同學提供圖檔

Grid Search v.s. Random Search



Assumption: top K results are good enough

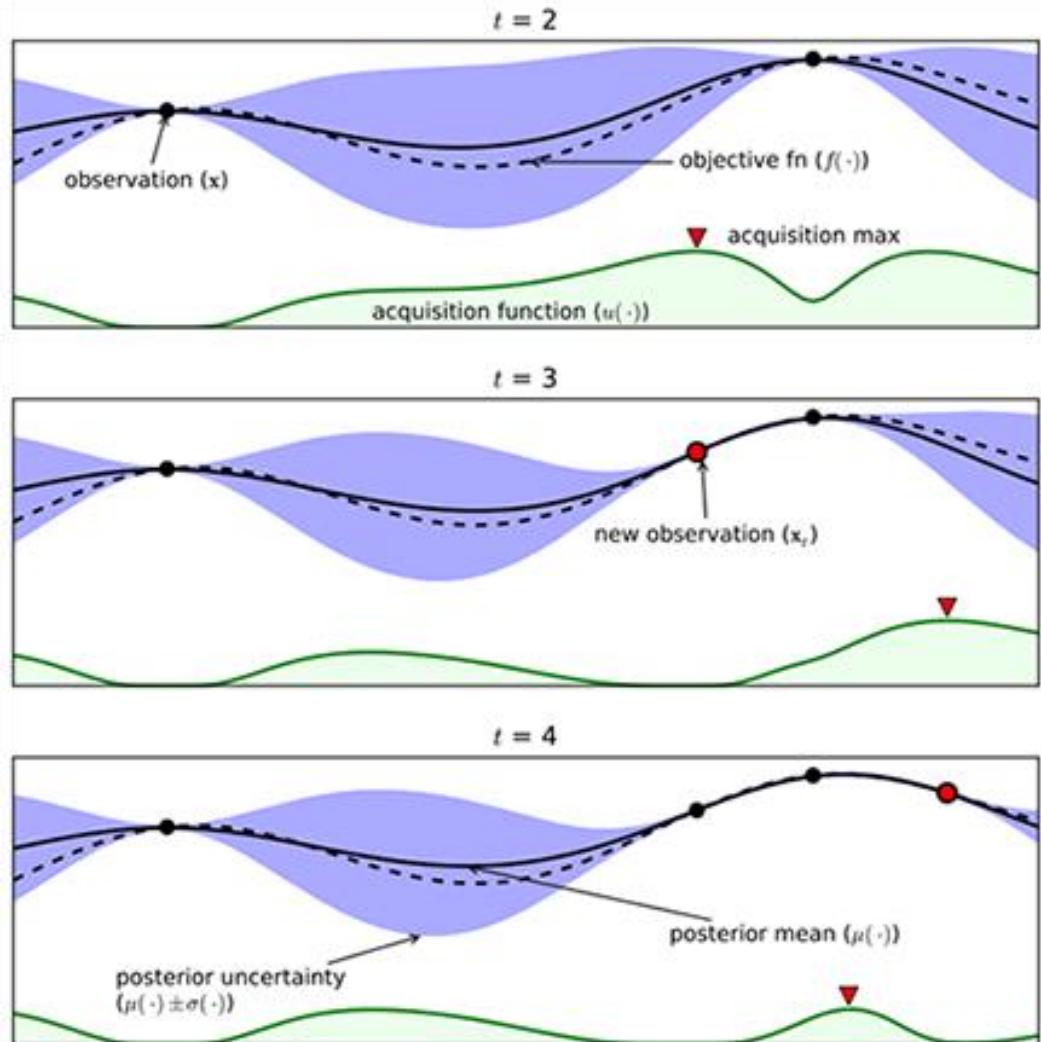
If there are N points, probability K/N that your sample is in top K

Sample x times: $1 - (1 - K/N)^x > 90\%$

$$\text{If } N = 1000, K = 10 \longrightarrow x = 230$$

$$K = 100 \longrightarrow x = 22$$

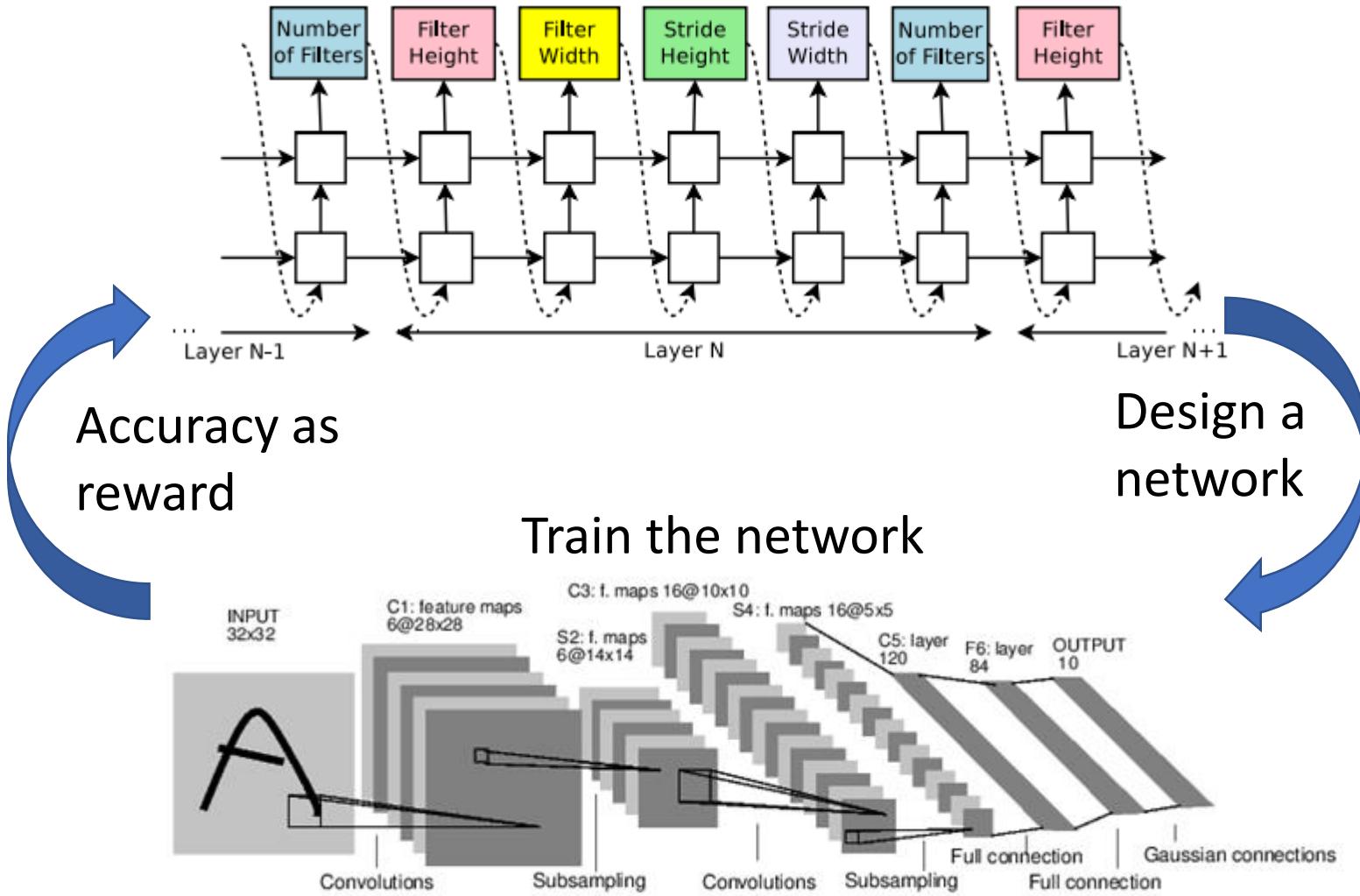
Model-based Hyperparameter Optimization



<https://cloud.google.com/blog/big-data/2017/08/hyperparameter-tuning-in-cloud-machine-learning-engine-using-bayesian-optimization>

Reinforcement Learning

One kind of meta learning (or learn to learn)

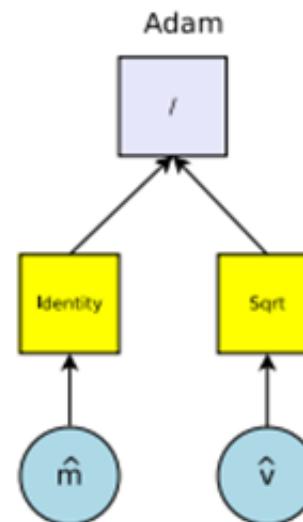
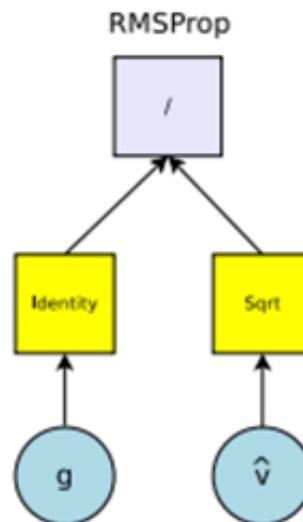
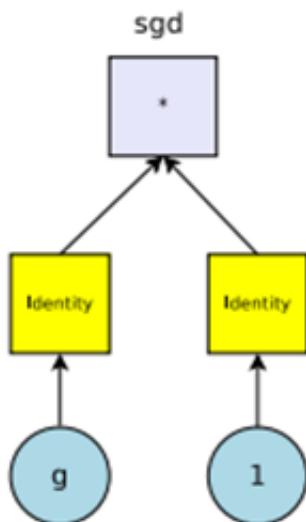
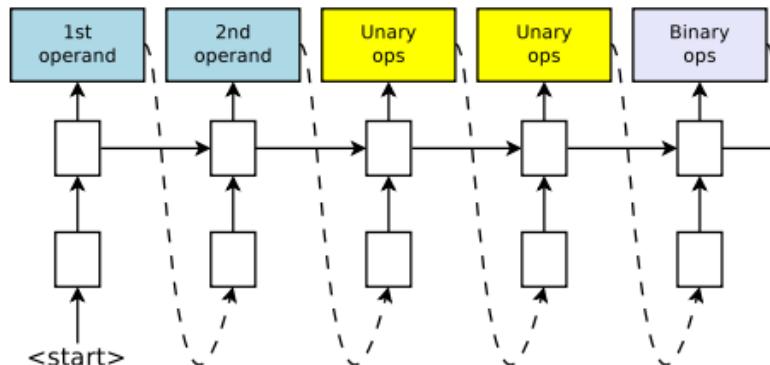


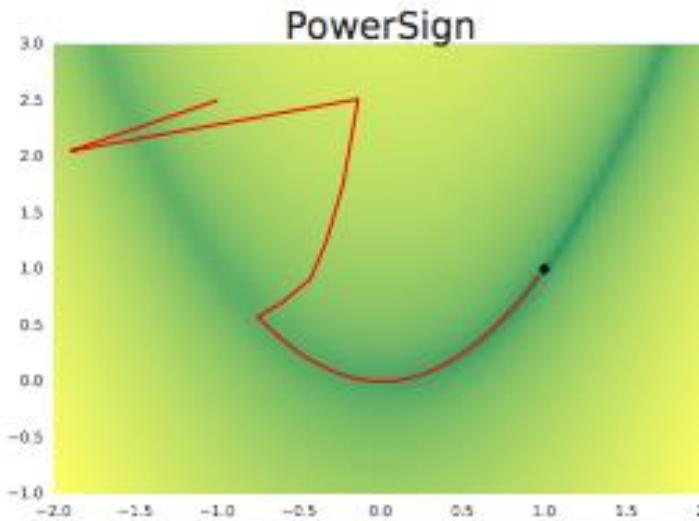
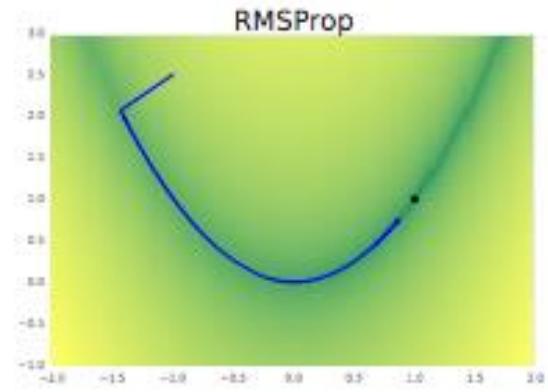
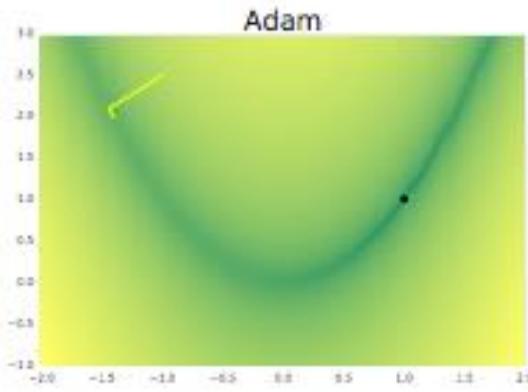
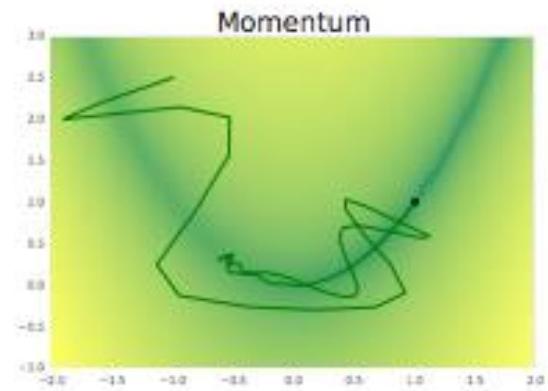
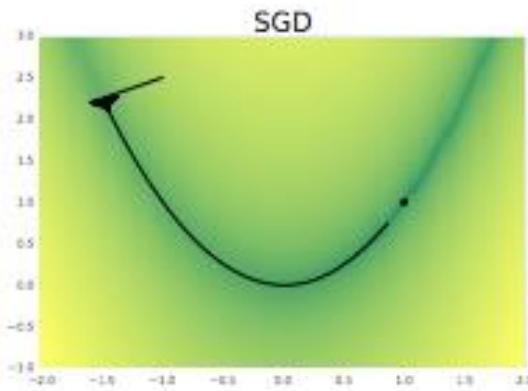
A Full Convolutional Neural Network (LeNet)

Learning Rate

- **Operands:** $g, g^2, g^3, \hat{m}, \hat{v}, \hat{\gamma}$, $\text{sign}(g)$, $\text{sign}(\hat{m})$, 1, 2, $\epsilon \sim N(0, 0.01)$, $10^{-4}w$, $10^{-3}w$, $10^{-2}w$, $10^{-1}w$, Adam and RMSProp.

- **Unary functions** which map input x to: $x, -x, e^x, \log|x|, \sqrt{|x|}, \text{clip}(x, 10^{-5}), \text{clip}(x, 10^{-4}), \text{clip}(x, 10^{-3}), \text{drop}(x, 0.1), \text{drop}(x, 0.3), \text{drop}(x, 0.5)$ and $\text{sign}(x)$.
- **Binary functions** which map (x, y) to $x + y$ (addition), $x - y$ (subtraction), $x * y$ (multiplication), $\frac{x}{y+\delta}$ (division), x^y (exponentiation) or x (keep left).

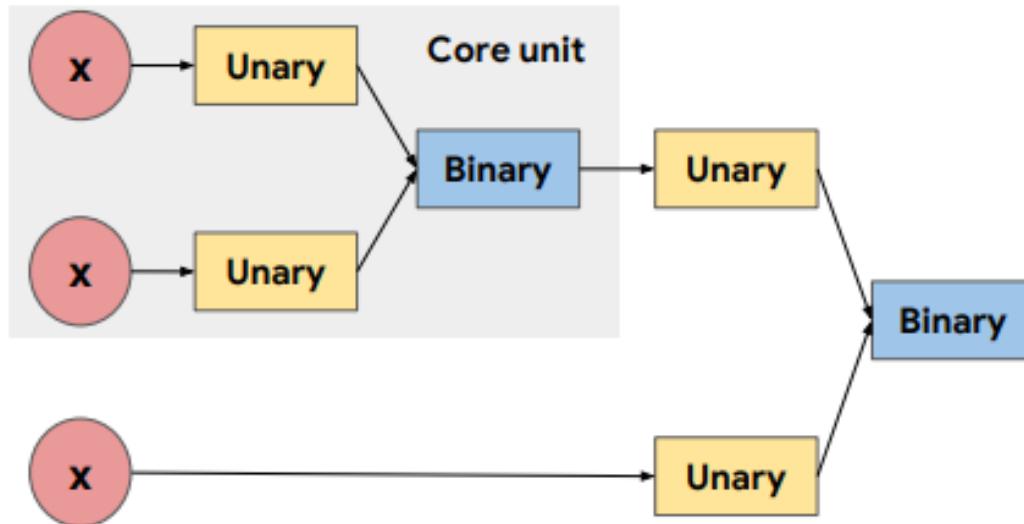




$$e^{\text{sign}(g) * \text{sign}(m)} * g$$

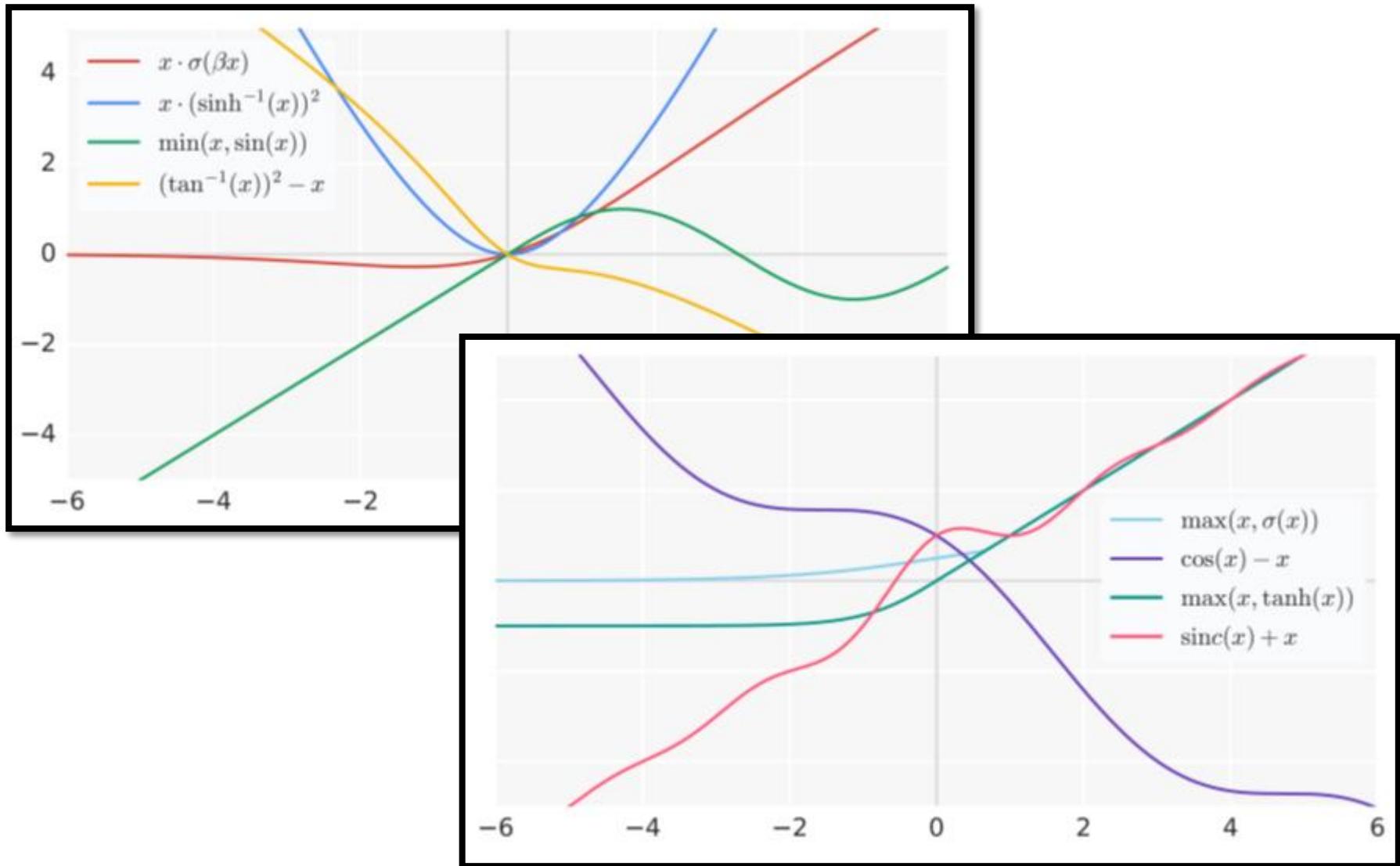
Can transfer to
new tasks

Activation Function

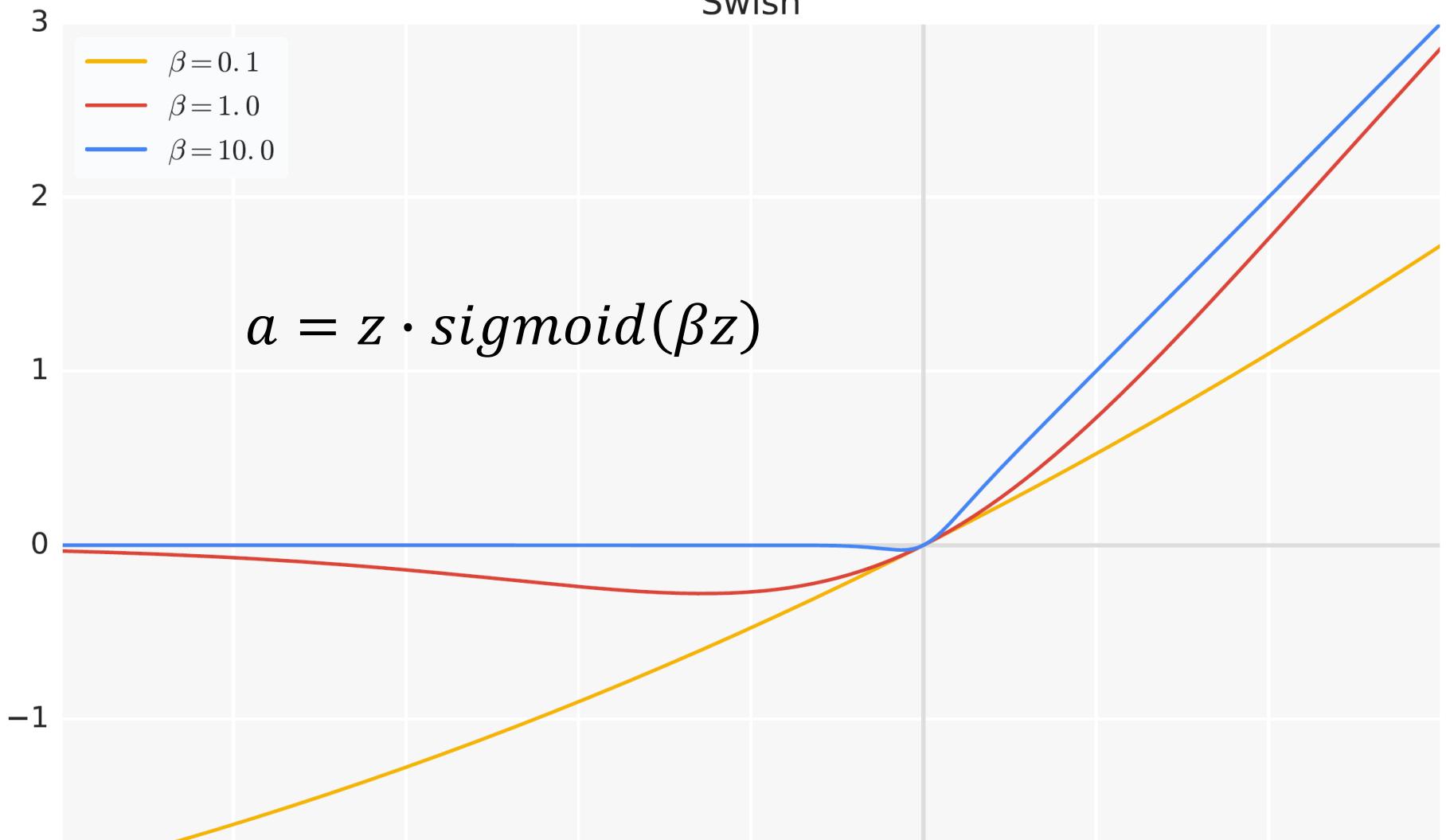


- **Unary functions:** $x, -x, |x|, x^2, x^3, \sqrt{x}, \beta x, x + \beta, \log(|x| + \epsilon), \exp(x) \sin(x), \cos(x), \sinh(x), \cosh(x), \tanh(x), \sinh^{-1}(x), \tan^{-1}(x), \text{sinc}(x), \max(x, 0), \min(x, 0), \sigma(x), \log(1 + \exp(x)), \exp(-x^2), \text{erf}(x), \beta$
- **Binary functions:** $x_1 + x_2, x_1 \cdot x_2, x_1 - x_2, \frac{x_1}{x_2 + \epsilon}, \max(x_1, x_2), \min(x_1, x_2), \sigma(x_1) \cdot x_2, \exp(-\beta(x_1 - x_2)^2), \exp(-\beta|x_1 - x_2|), \beta x_1 + (1 - \beta)x_2$

Activation Function



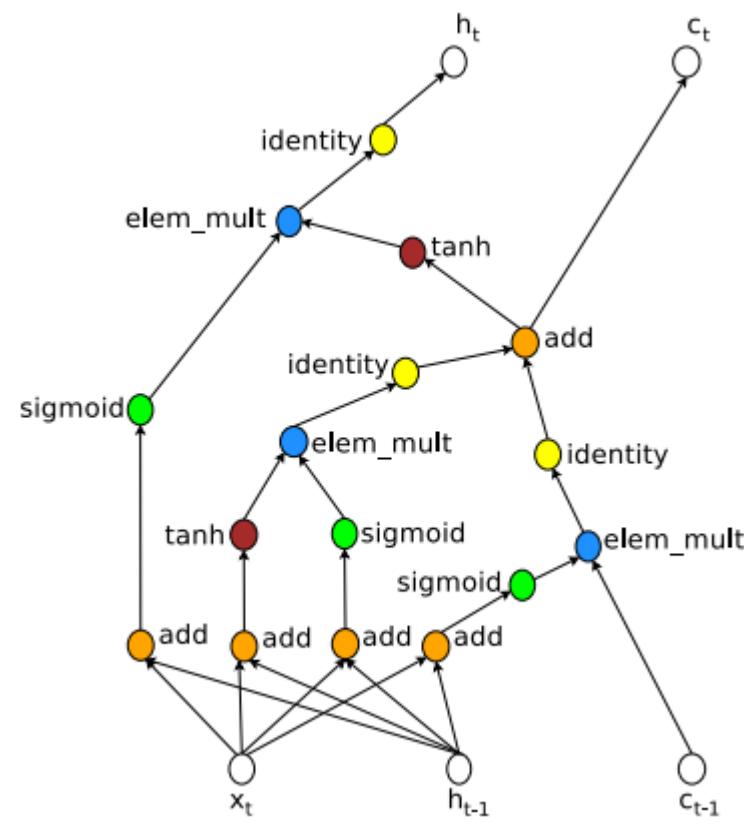
Swish



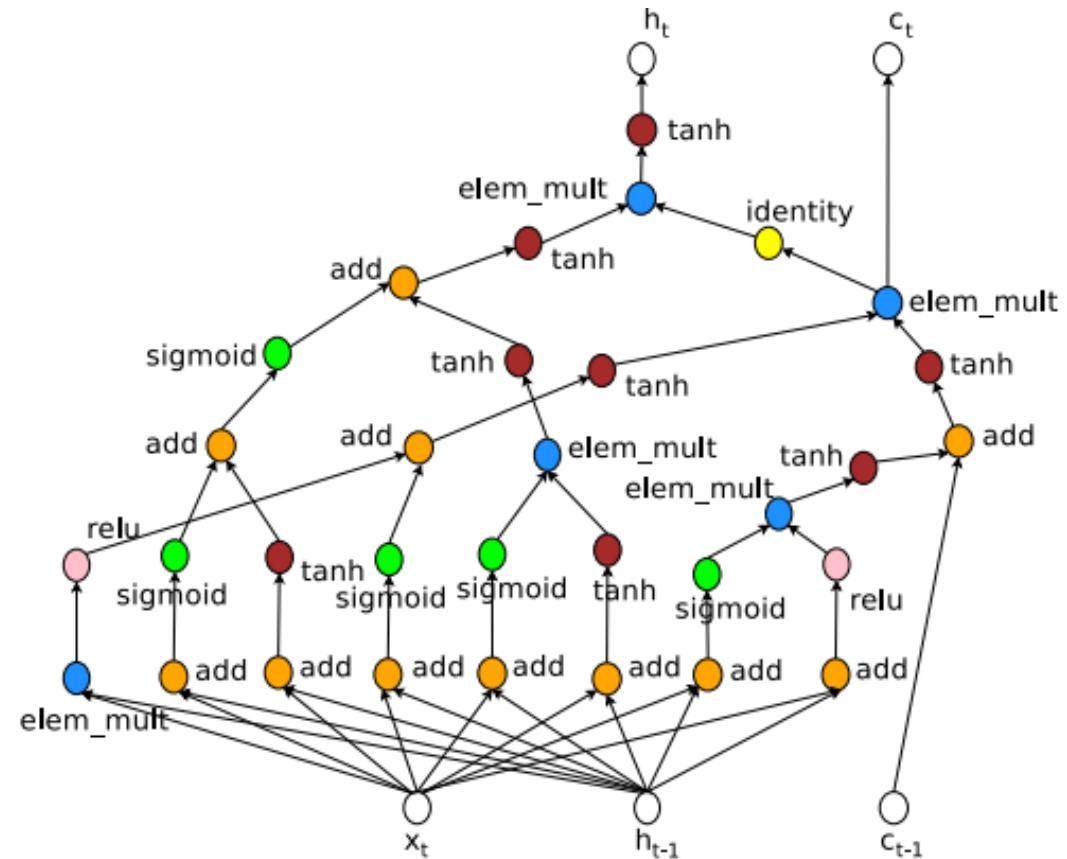
Baselines	ReLU	LReLU	PReLU	Softplus	ELU	SELU	GELU
Swish > Baseline	9	7	6	6	8	8	8
Swish = Baseline	0	1	3	2	0	1	1
Swish < Baseline	0	1	0	1	1	0	0

Neural Architecture Search with Reinforcement Learning

LSTM



From Reinforcement Learning



Computation Issue?

- Original version: 450 GPUs for 3-4 days (32,400-43,200 GPU hours).
- New version: Nvidia GTX 1080Ti GPU takes less than 16 hours.
- Main idea: forcing all child models to share weights instead of training from scratch.

Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, Jeff Dean, "Efficient Neural Architecture Search via Parameter Sharing", arXiv, 2018

Introduction of Generative Adversarial Network (GAN)

李宏毅

Hung-yi Lee

Generative Adversarial Network (GAN)

- How to pronounce “GAN”?



Google 小姐

Yann LeCun's comment

What are some recent and potentially upcoming breakthroughs in unsupervised learning?



Yann LeCun, Director of AI Research at Facebook and Professor at NYU



Written Jul 29 · Upvoted by Joaquin Quiñonero Candela, Director Applied Machine Learning at Facebook and Huang Xiao

Adversarial training is the coolest thing since sliced bread.

I've listed a bunch of relevant papers in a previous answer.

Expect more impressive results with this technique in the coming years.

What's missing at the moment is a good understanding of it so we can make it work reliably. It's very finicky. Sort of like ConvNet were in the 1990s, when I had the reputation of being the only person who could make them work (which wasn't true).

Yann LeCun's comment

What are some recent and potentially upcoming breakthroughs in deep learning?



Yann LeCun, Director of AI Research at Facebook and Professor at NYU

Written Jul 29 · Upvoted by Joaquin Quiñonero Candela, Director Applied Machine Learning at Facebook and Nikhil Garg, I lead a team of Quora engineers working on ML/NLP problems



.....

The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks). This is an idea that was originally proposed by Ian Goodfellow when he was a student with Yoshua Bengio at the University of Montreal (he since moved to Google Brain and recently to OpenAI).

This, and the variations that are now being proposed is the most interesting idea in the last 10 years in ML, in my opinion.

<https://www.quora.com/What-are-some-recent-and-potentially-upcoming-breakthroughs-in-deep-learning>

All Kinds of GAN ...

<https://github.com/hindupuravinash/the-gan-zoo>

GAN

ACGAN

BGAN

CGAN

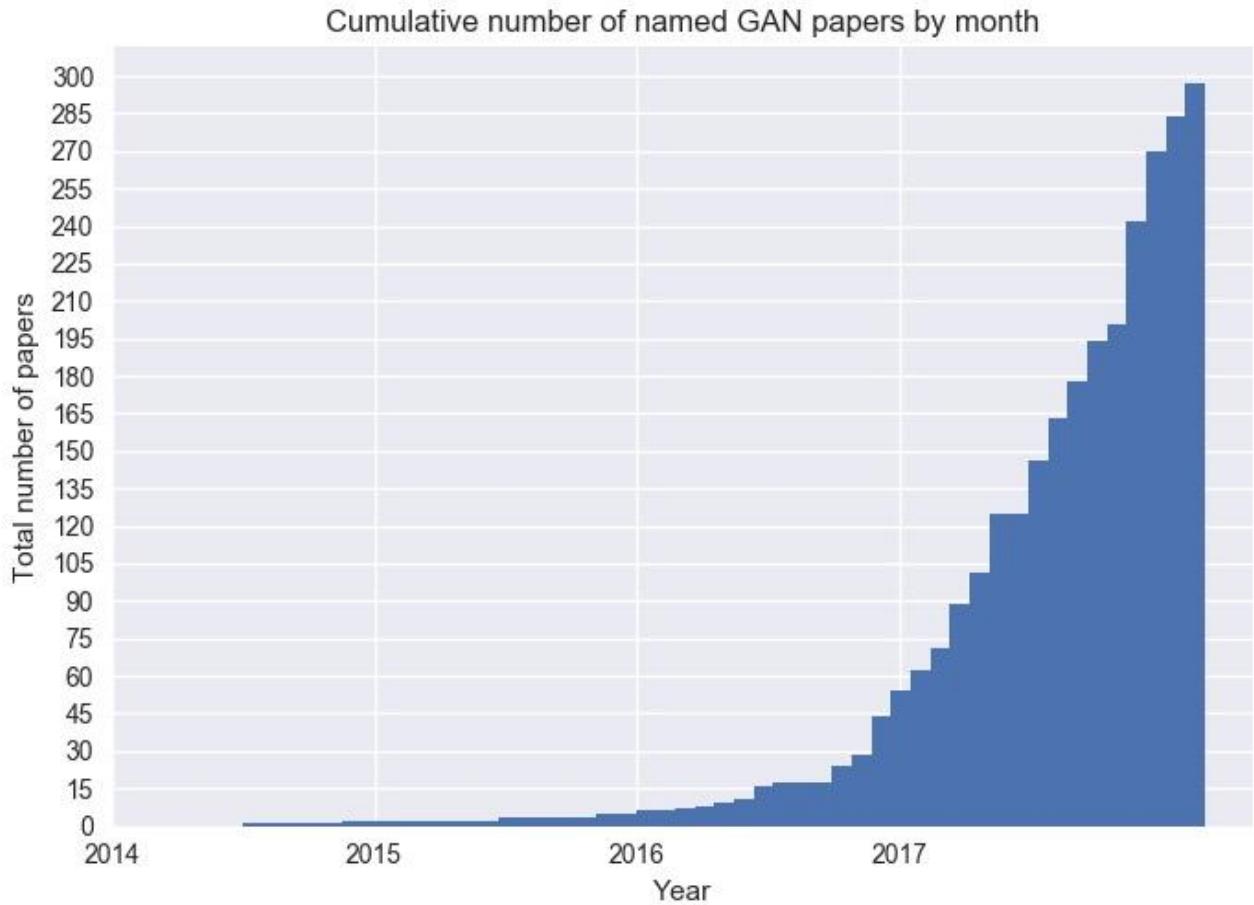
DCGAN

EBGAN

fGAN

GoGAN

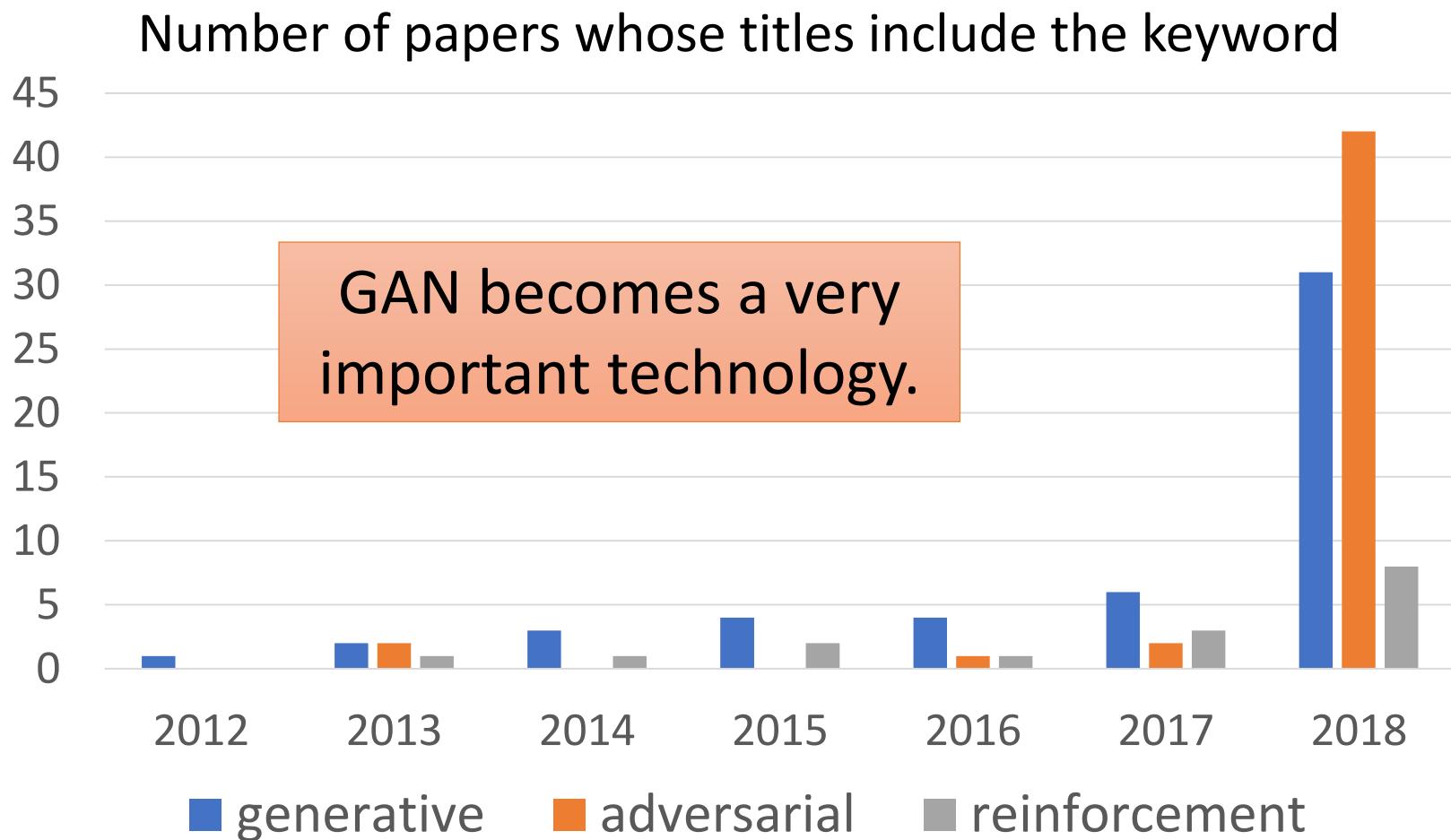
⋮



Mihaela Rosca, Balaji Lakshminarayanan, David Warde-Farley, Shakir Mohamed, "Variational Approaches for Auto-Encoding Generative Adversarial Networks", arXiv, 2017

²We use the Greek α prefix for α -GAN, as AEGAN and most other Latin prefixes seem to have been taken
<https://deephunt.in/the-gan-zoo-79597dc8c347>.

Keyword search on session index page,
so session names are included.



Outline

Basic Idea of GAN

GAN as structured learning

Can Generator learn by itself?

Can Discriminator generate?

A little bit theory

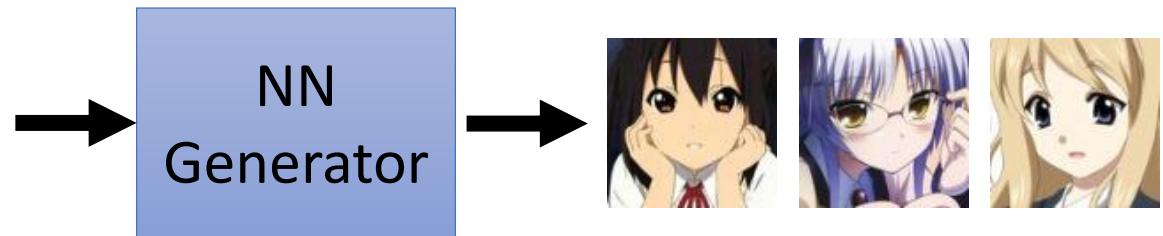
Generation

We will control what to generate latter. → Conditional Generation

Image Generation

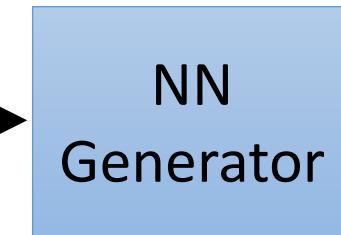
$$\begin{bmatrix} 0.3 \\ -0.1 \\ \vdots \\ -0.7 \end{bmatrix} \begin{bmatrix} 0.1 \\ -0.1 \\ \vdots \\ 0.7 \end{bmatrix} \begin{bmatrix} -0.3 \\ 0.1 \\ \vdots \\ 0.9 \end{bmatrix}$$

In a specific range



Sentence Generation

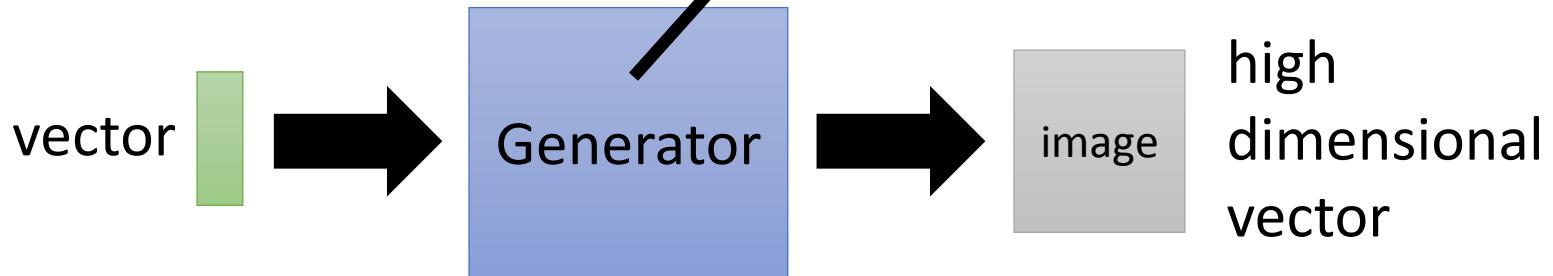
$$\begin{bmatrix} 0.3 \\ -0.1 \\ \vdots \\ -0.7 \end{bmatrix} \begin{bmatrix} 0.1 \\ -0.1 \\ \vdots \\ 0.2 \end{bmatrix} \begin{bmatrix} -0.3 \\ 0.1 \\ \vdots \\ 0.5 \end{bmatrix}$$



How are you?
Good morning.
Good afternoon.

Basic Idea of GAN

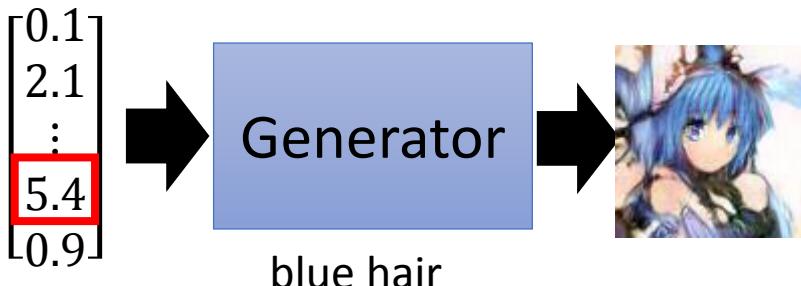
It is a neural network (NN), or a function.



Each dimension of input vector represents some characteristics.



Longer hair



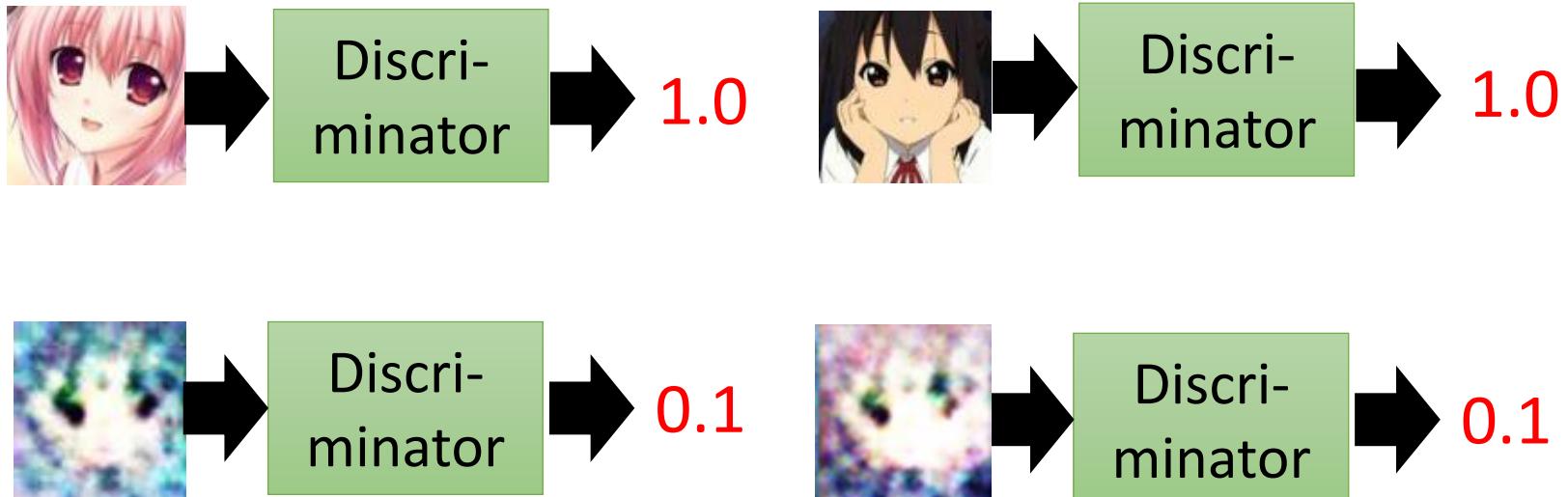
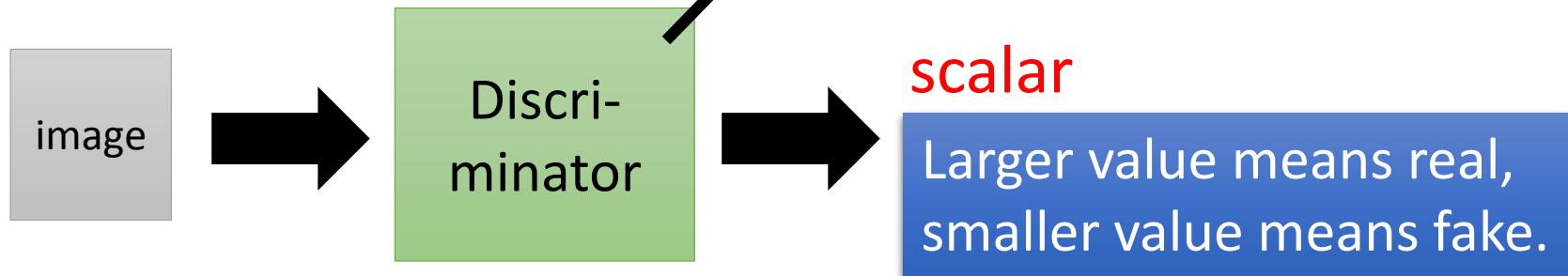
blue hair



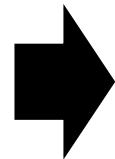
Open mouth

Basic Idea of GAN

It is a neural network (NN), or a function.



Basic Idea of GAN



Brown

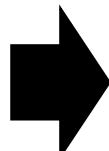


veins

.....

Butterflies are
not brown

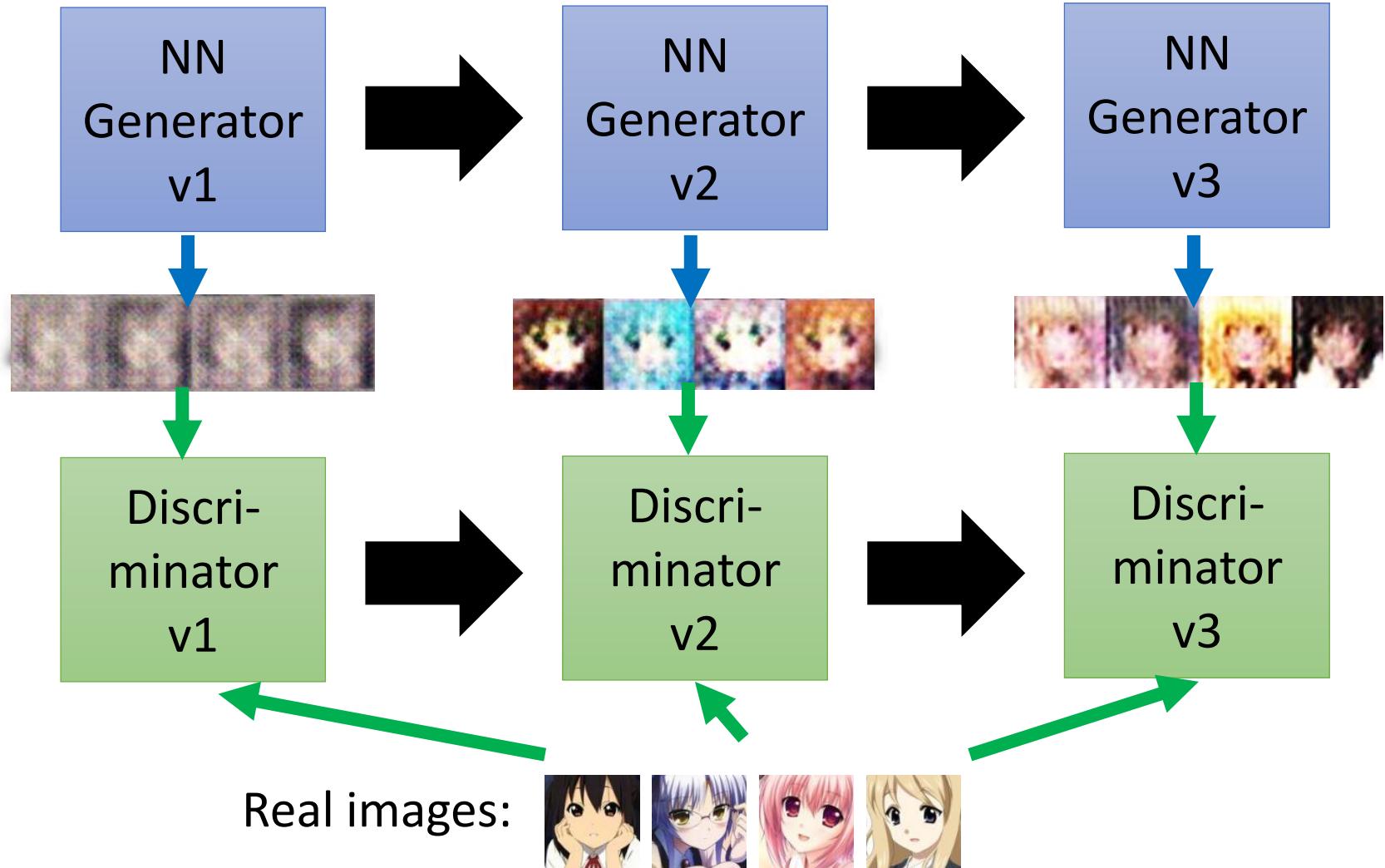
Butterflies do
not have veins



Discriminator

Basic Idea of GAN

This is where the term
“adversarial” comes from.
You can explain the process
in different ways.....



Basic Idea of GAN (和平的比喻)

Generator
(student)

Discriminator
(teacher)



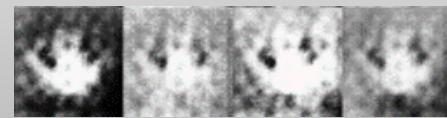
Generator
v1



Discriminator
v1

沒有兩個圈

Generator
v2



Discriminator
v2

沒有彩色

Generator
v3



為什麼不自己學？

為什麼不自己做？

Generator v.s. Discriminator

- 寫作敵人，唸做朋友

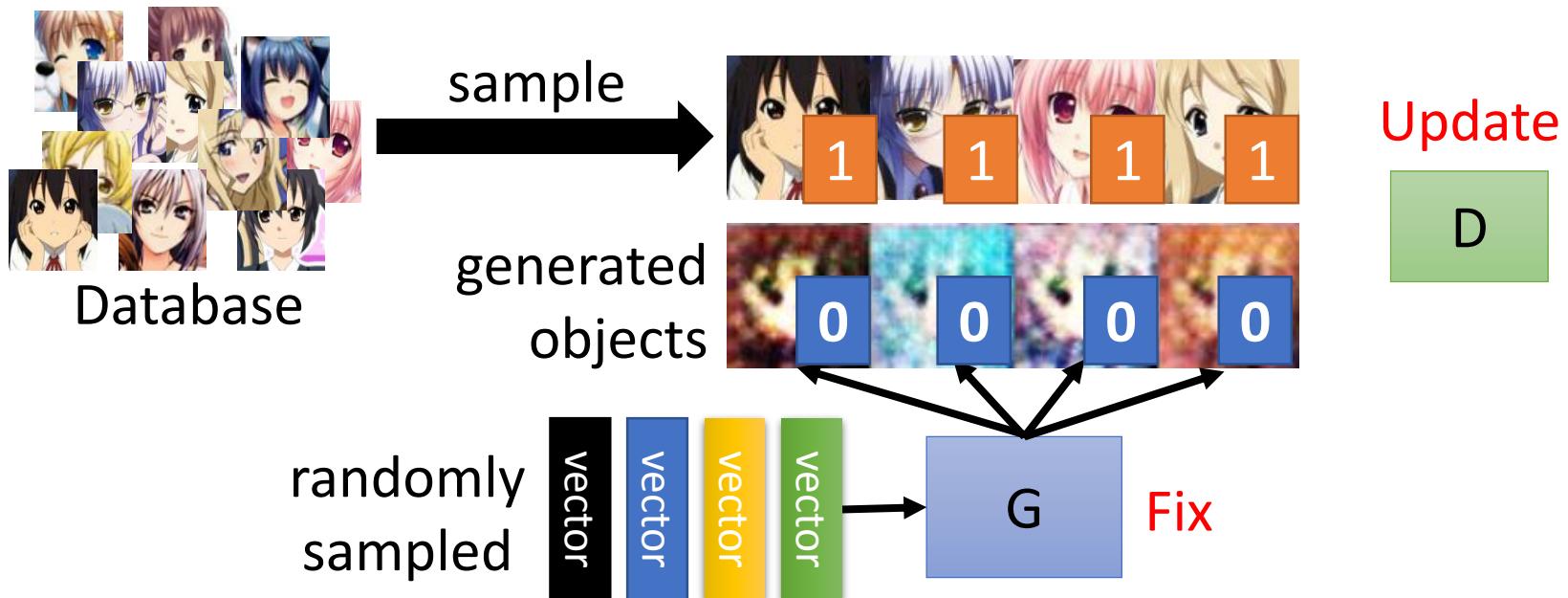


Algorithm

- Initialize generator and discriminator
- In each training iteration:



Step 1: Fix generator G, and update discriminator D



Discriminator learns to assign high scores to real objects and low scores to generated objects.

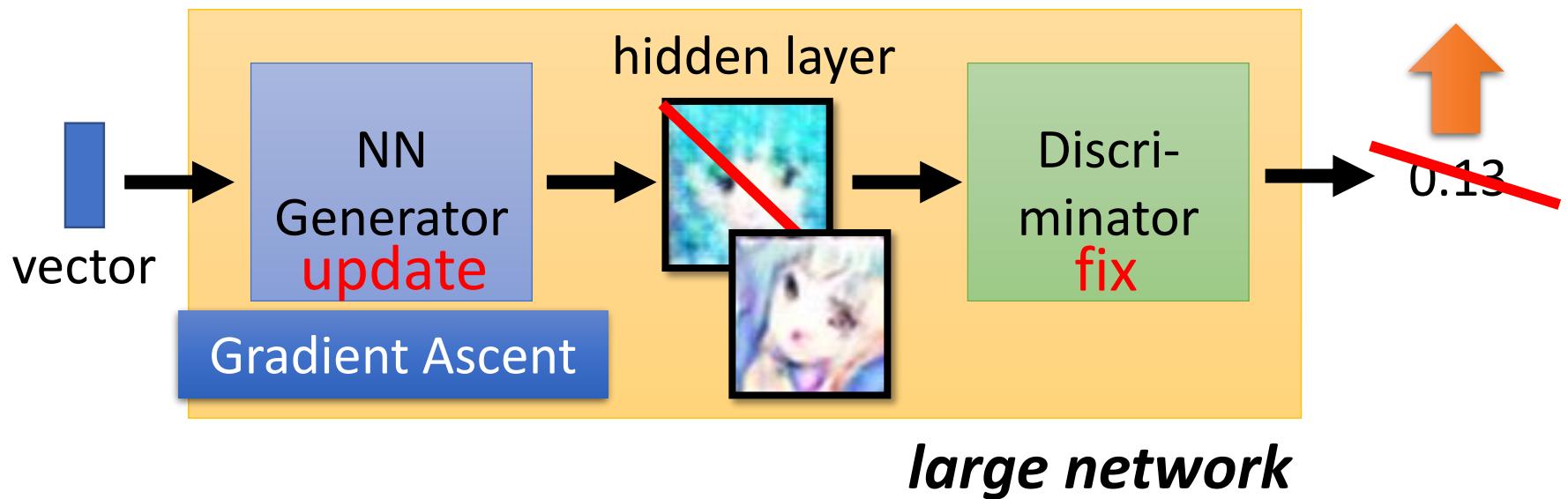
Algorithm

- Initialize generator and discriminator
- In each training iteration:



Step 2: Fix discriminator D, and update generator G

Generator learns to “fool” the discriminator



Algorithm

Initialize θ_d for D and θ_g for G

- In each training iteration:

- Sample m examples $\{x^1, x^2, \dots, x^m\}$ from database
- Sample m noise samples $\{z^1, z^2, \dots, z^m\}$ from a distribution
- Obtaining generated data $\{\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^m\}, \tilde{x}^i = G(z^i)$
- Update discriminator parameters θ_d to maximize
 - $\tilde{V} = \frac{1}{m} \sum_{i=1}^m \log D(x^i) + \frac{1}{m} \sum_{i=1}^m \log (1 - D(\tilde{x}^i))$
 - $\theta_d \leftarrow \theta_d + \eta \nabla \tilde{V}(\theta_d)$

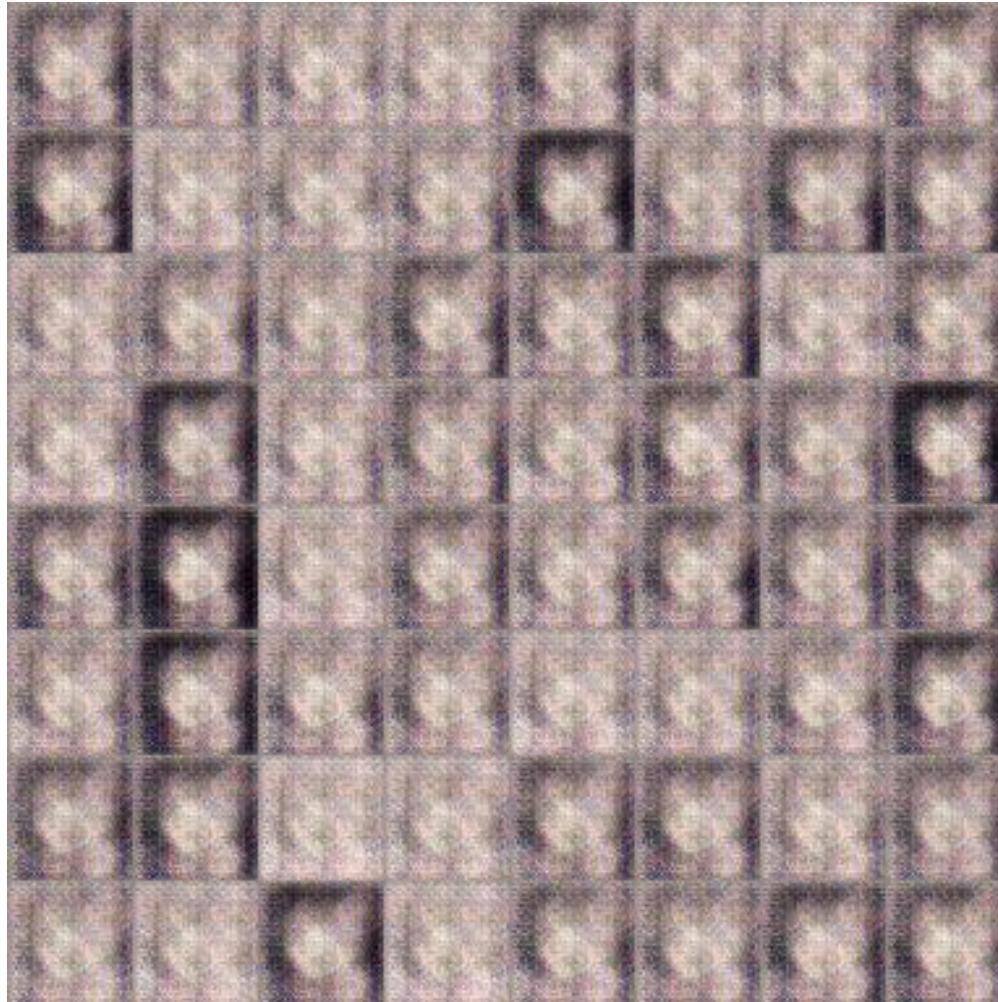
- Sample m noise samples $\{z^1, z^2, \dots, z^m\}$ from a distribution
- Update generator parameters θ_g to maximize
 - $\tilde{V} = \frac{1}{m} \sum_{i=1}^m \log (D(G(z^i)))$
 - $\theta_g \leftarrow \theta_g - \eta \nabla \tilde{V}(\theta_g)$

Learning
D

Learning
G

Anime Face Generation

100 updates



Source of training data: <https://zhuanlan.zhihu.com/p/24767059>

Anime Face Generation



1000 updates

Anime Face Generation

2000 updates



Anime Face Generation

5000 updates



Anime Face Generation

10,000 updates



Anime Face Generation

20,000 updates



Anime Face Generation

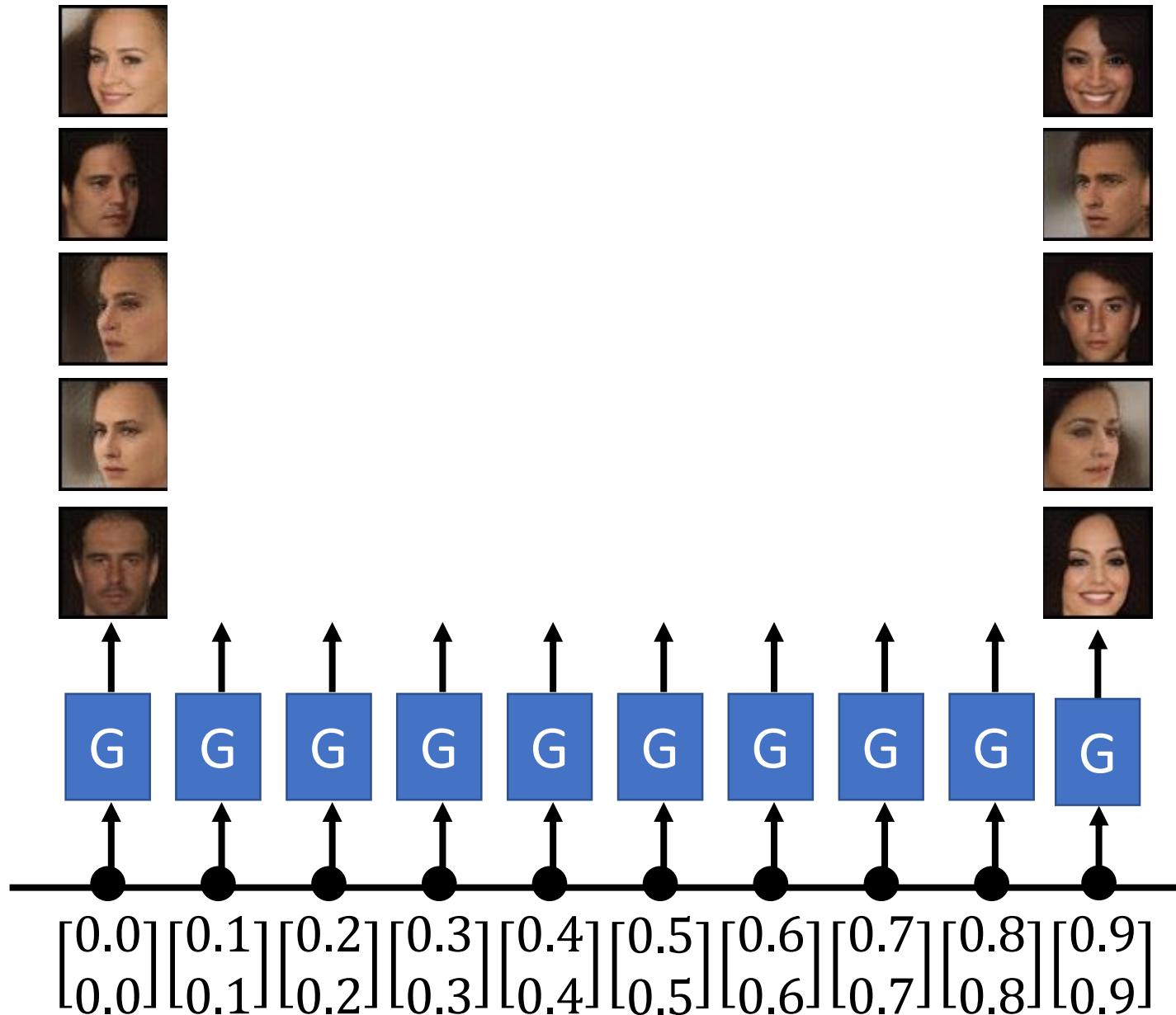
50,000 updates





The faces
generated by
machine.

圖片生成：
吳宗翰、謝濬丞、
陳延昊、錢柏均



感謝陳柏文同學提供實驗結果

Outline

Basic Idea of GAN

GAN as structured learning

Can Generator learn by itself?

Can Discriminator generate?

A little bit theory

Structured Learning

Machine learning is to find a function f

$$f : X \rightarrow Y$$

Regression: output a scalar

Classification: output a “class” (one-hot vector)

1	0	0
---	---	---

Class 1

0	1	0
---	---	---

Class 2

0	0	1
---	---	---

Class 3

Structured Learning/Prediction: output a sequence, a matrix, a graph, a tree

Output is composed of components with dependency

Output Sequence

$$f : X \rightarrow Y$$

Machine Translation

X ：“機器學習及其深層與
結構化”
(sentence of language 1)

Y ：“Machine learning and
having it deep and structured”
(sentence of language 2)

Speech Recognition

X ：
(speech)

Y ：感謝大家來上課”
(transcription)

Chat-bot

X ：“How are you?”
(what a user says)

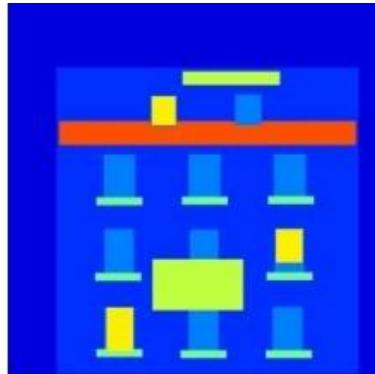
Y ：“I'm fine.”
(response of machine)

Output Matrix

$$f : X \rightarrow Y$$

Image to Image

$X :$



$Y :$



Colorization:



Ref: <https://arxiv.org/pdf/1611.07004v1.pdf>

Text to Image

$X :$ “this white and yellow flower
have thin white petals and a
round yellow stamen”

$Y :$



ref: <https://arxiv.org/pdf/1605.05396.pdf>

Why Structured Learning Challenging?

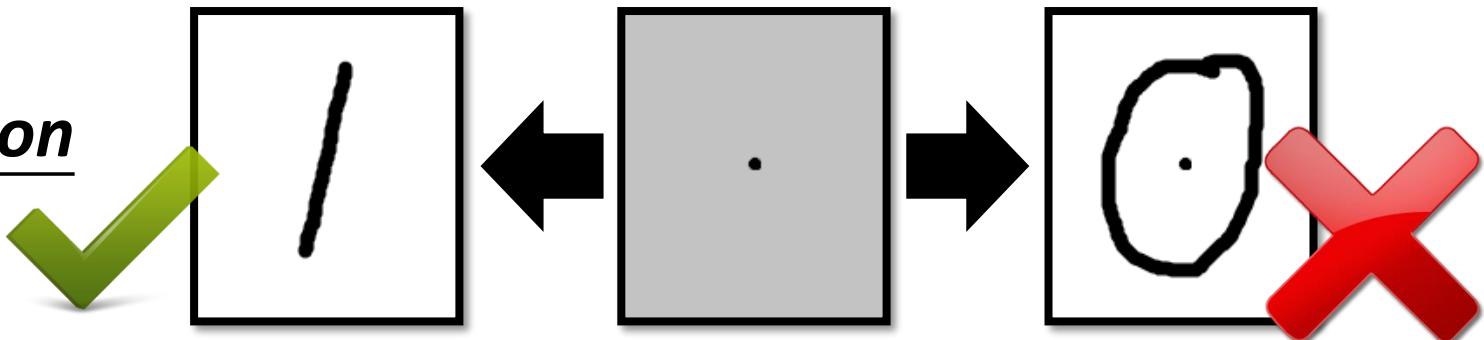
- **One-shot/Zero-shot Learning:**
 - In classification, each class has some examples.
 - In structured learning,
 - If you consider each possible output as a “class”
 - Since the output space is huge, most “classes” do not have any training data.
 - Machine has to create new stuff during testing.
 - Need more intelligence

Why Structured Learning Challenging?

- Machine has to learn to do *planning*
 - Machine generates objects component-by-component, but it should have a big picture in its mind.
 - Because the output components have dependency, they should be considered globally.

Image

Generation



Sentence

Generation

這個婆娘不是人

九天玄女下凡塵



Structured Learning Approach

Generator

Learn to generate
the object at the
component level



Discriminator

Evaluating the
whole object, and
find the best one



Outline

Basic Idea of GAN

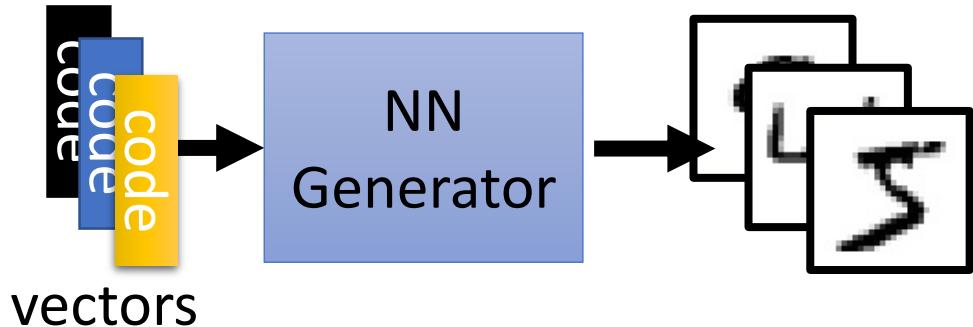
GAN as structured learning

Can Generator learn by itself?

Can Discriminator generate?

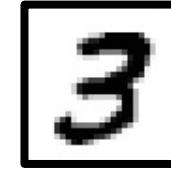
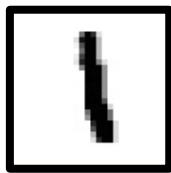
A little bit theory

Generator



code:
(where does they
come from?)

Image:

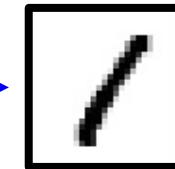


$$\begin{bmatrix} 0.1 \\ 0.9 \end{bmatrix}$$

NN
Generator

As close as possible

image



c.f.



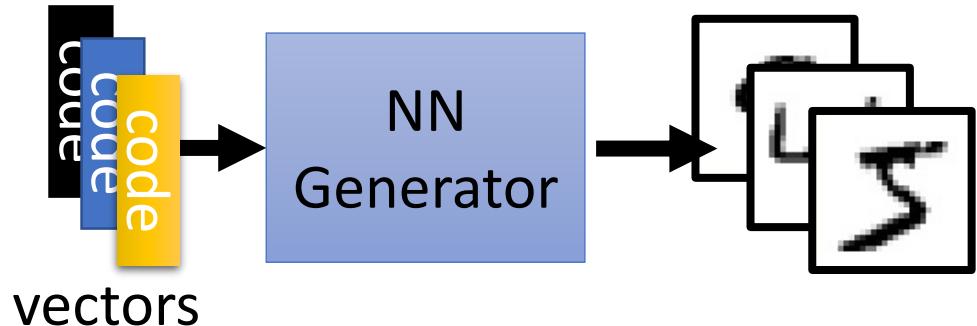
NN
Classifier

As close as possible

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 0 \\ \vdots \end{bmatrix}$$

Generator



code:
[0.1
-0.5]

Image:

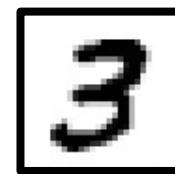
[0.1
0.9]



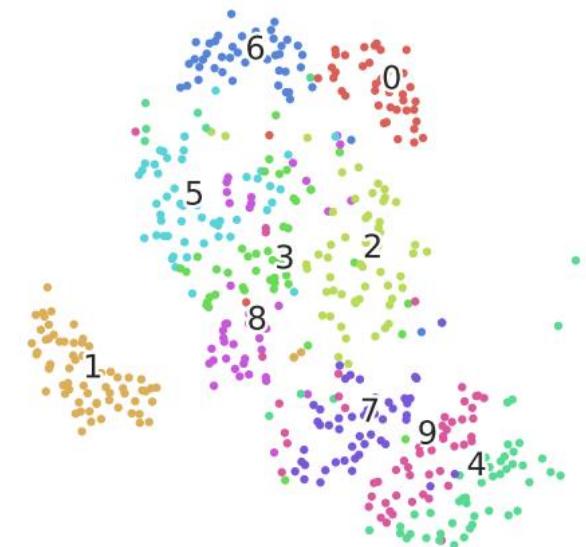
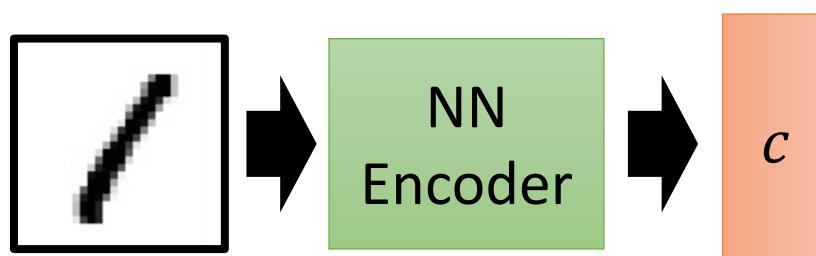
[0.2
-0.1]



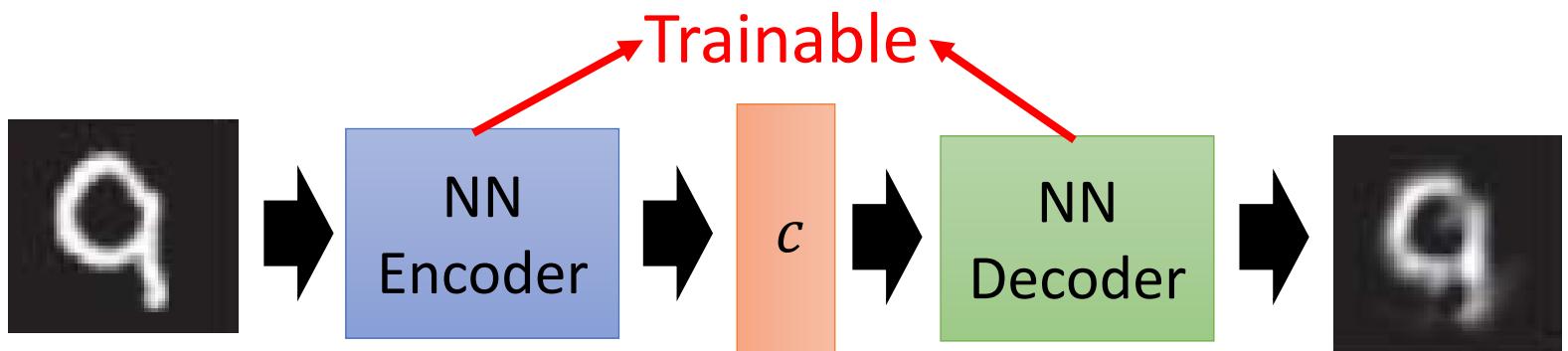
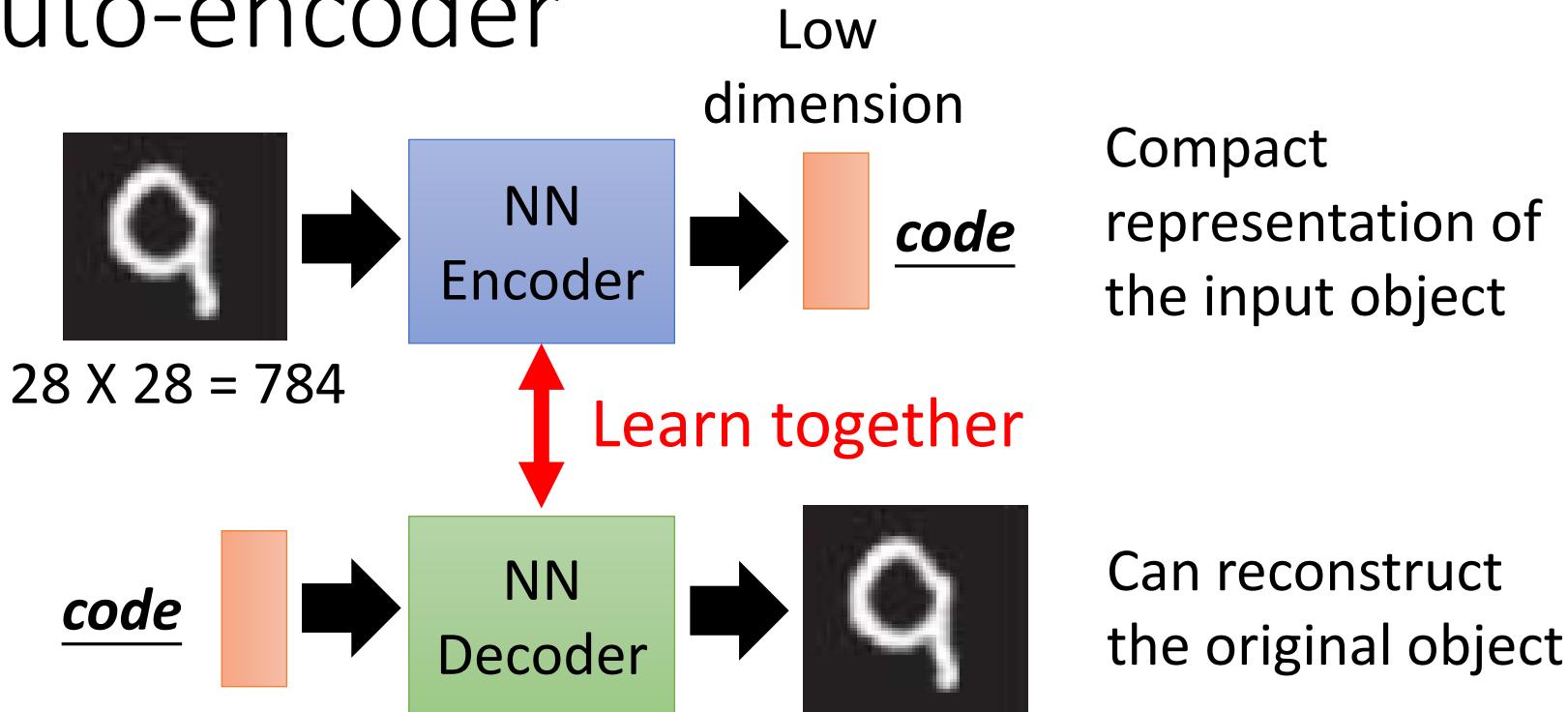
[0.3
0.2]



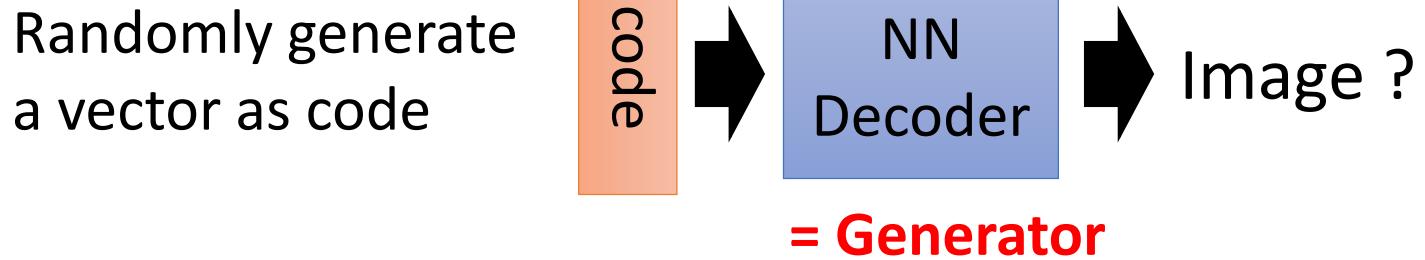
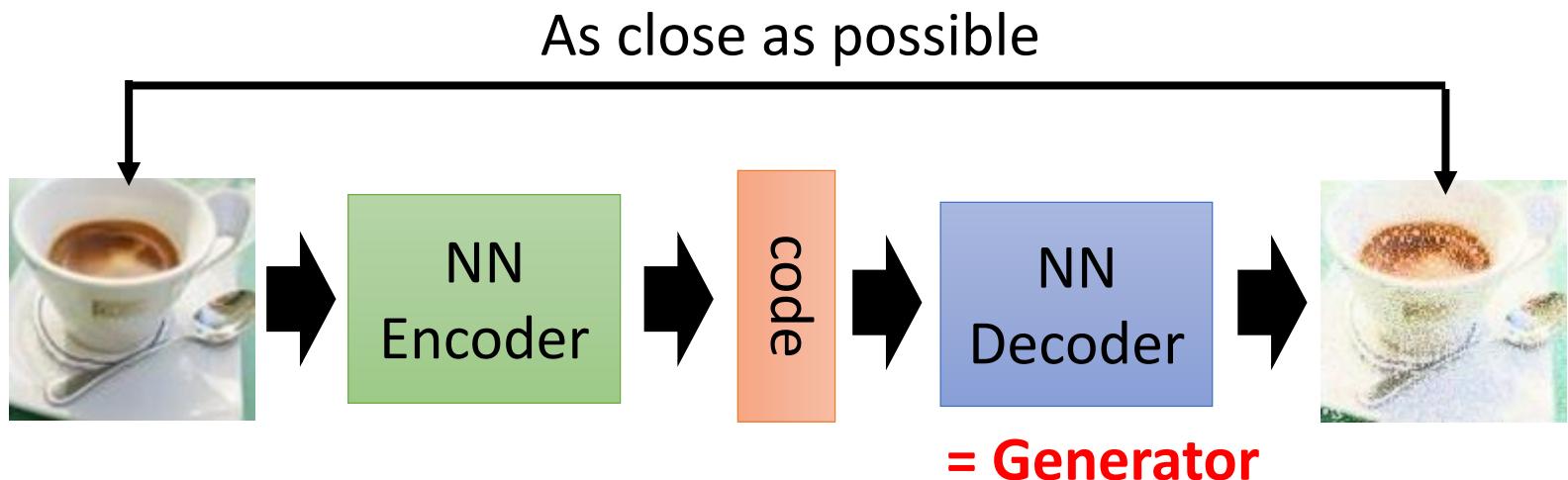
Encoder in auto-encoder
provides the code 😊



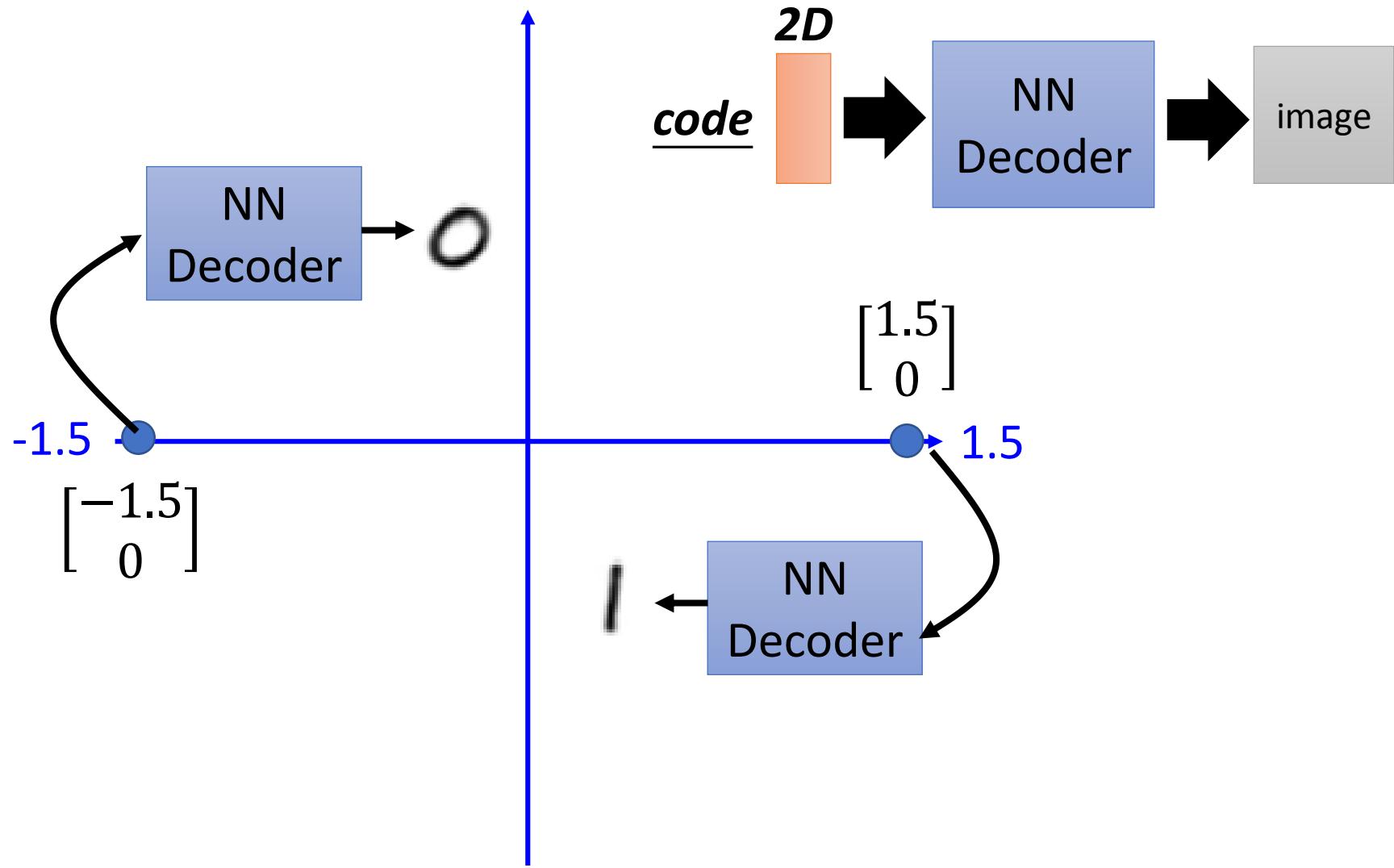
Auto-encoder



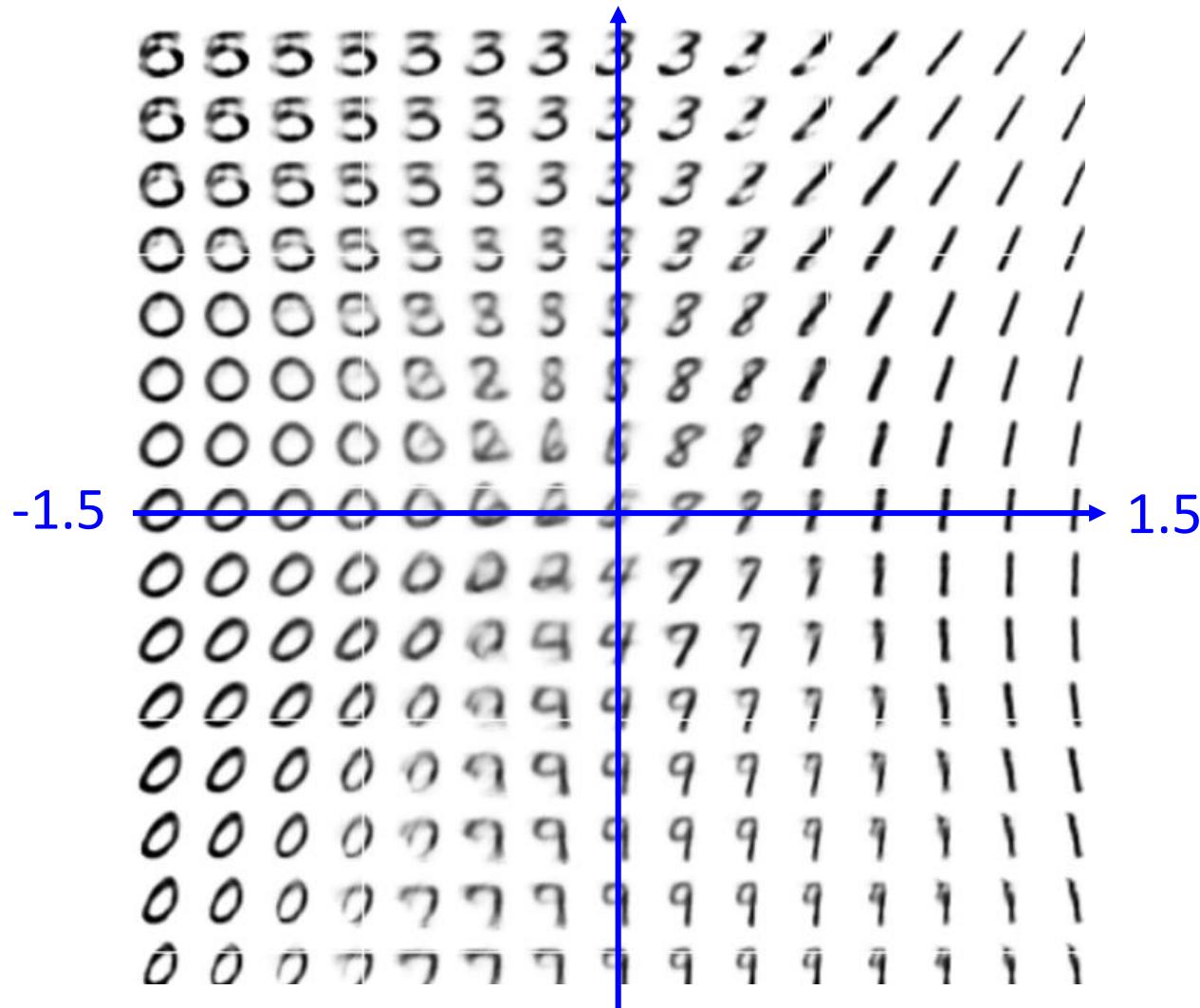
Auto-encoder



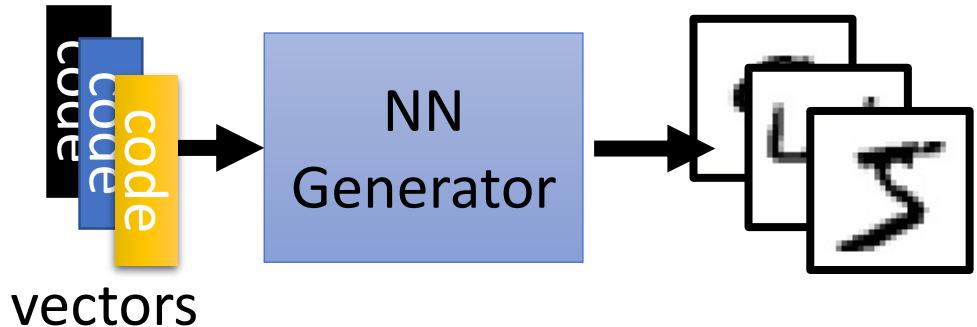
Auto-encoder



Auto-encoder



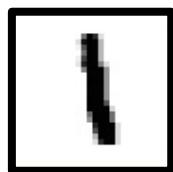
Auto-encoder



code:
[0.1]
[-0.5]

(where does them
come from?)

Image:



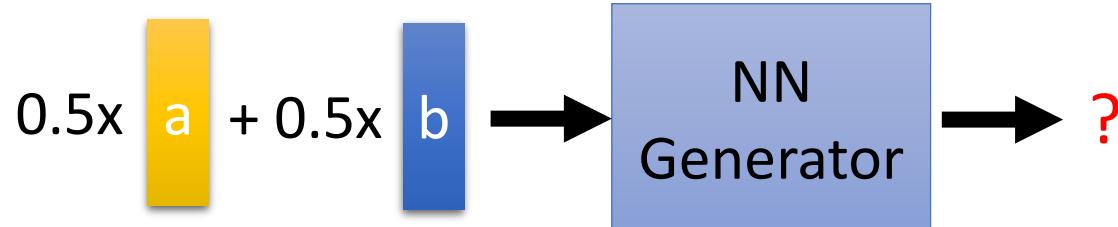
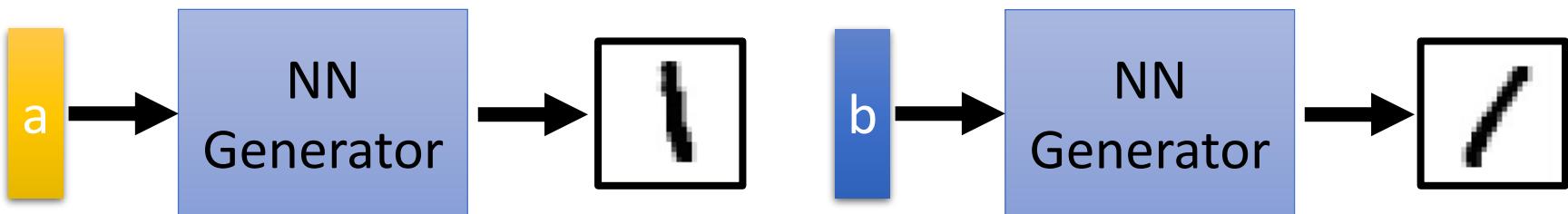
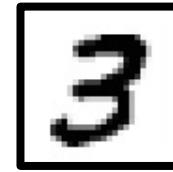
[0.1]
[0.9]



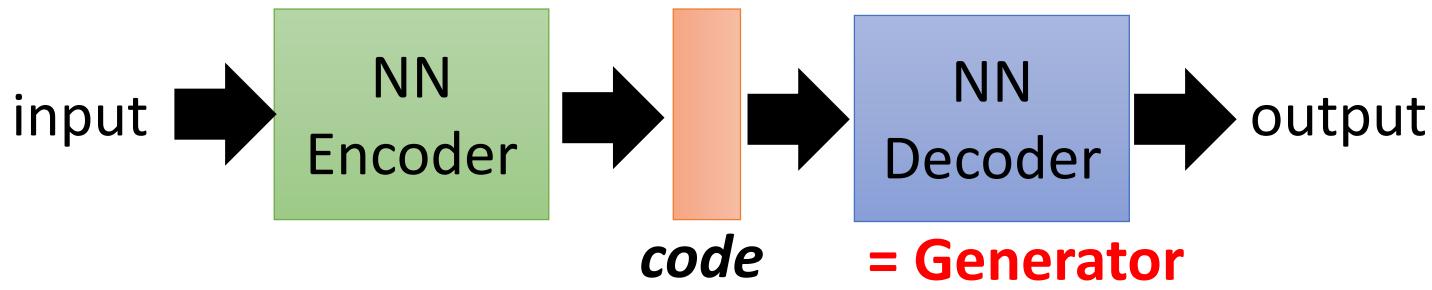
[0.2]
[-0.1]



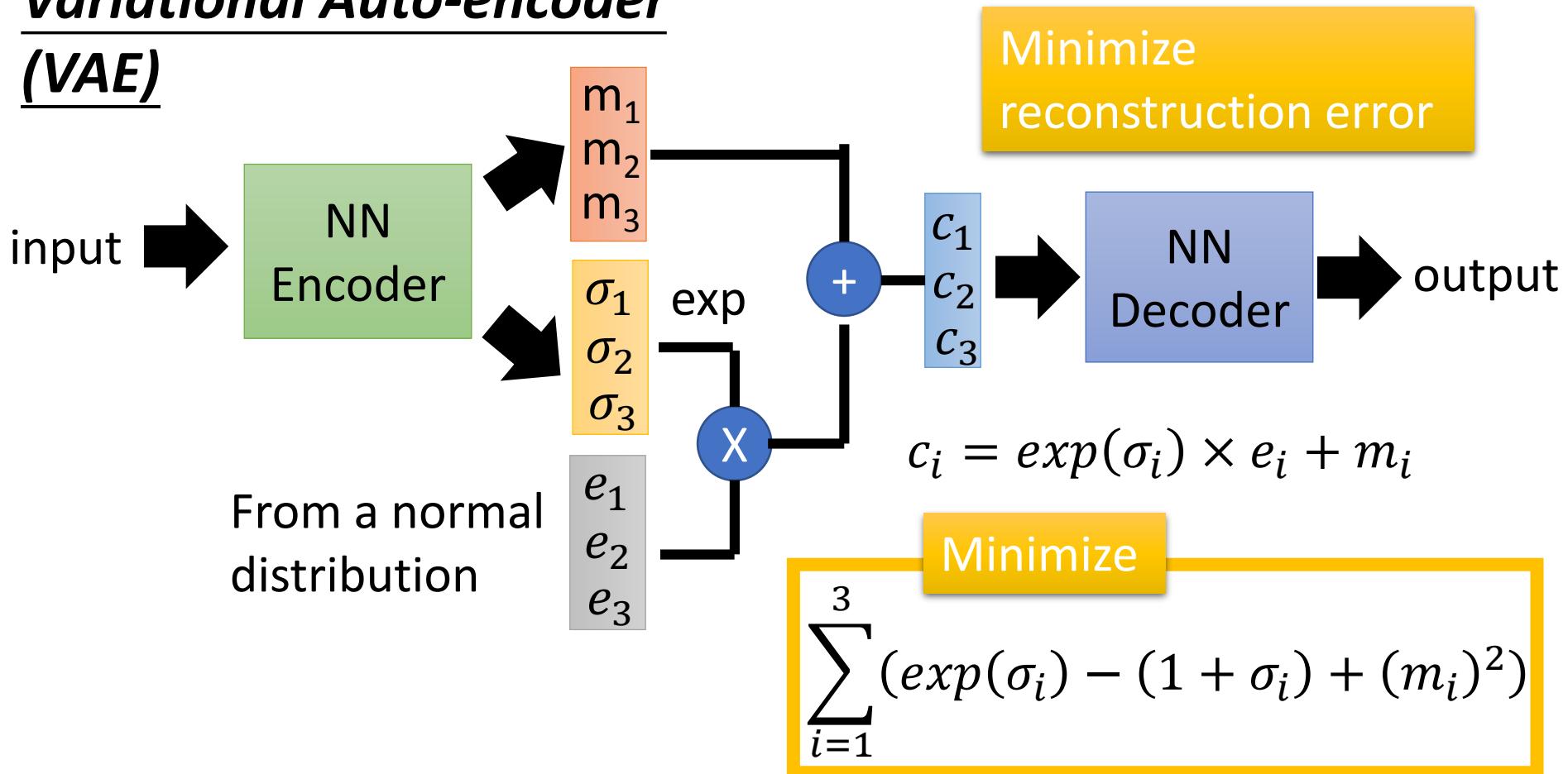
[0.3]
[0.2]



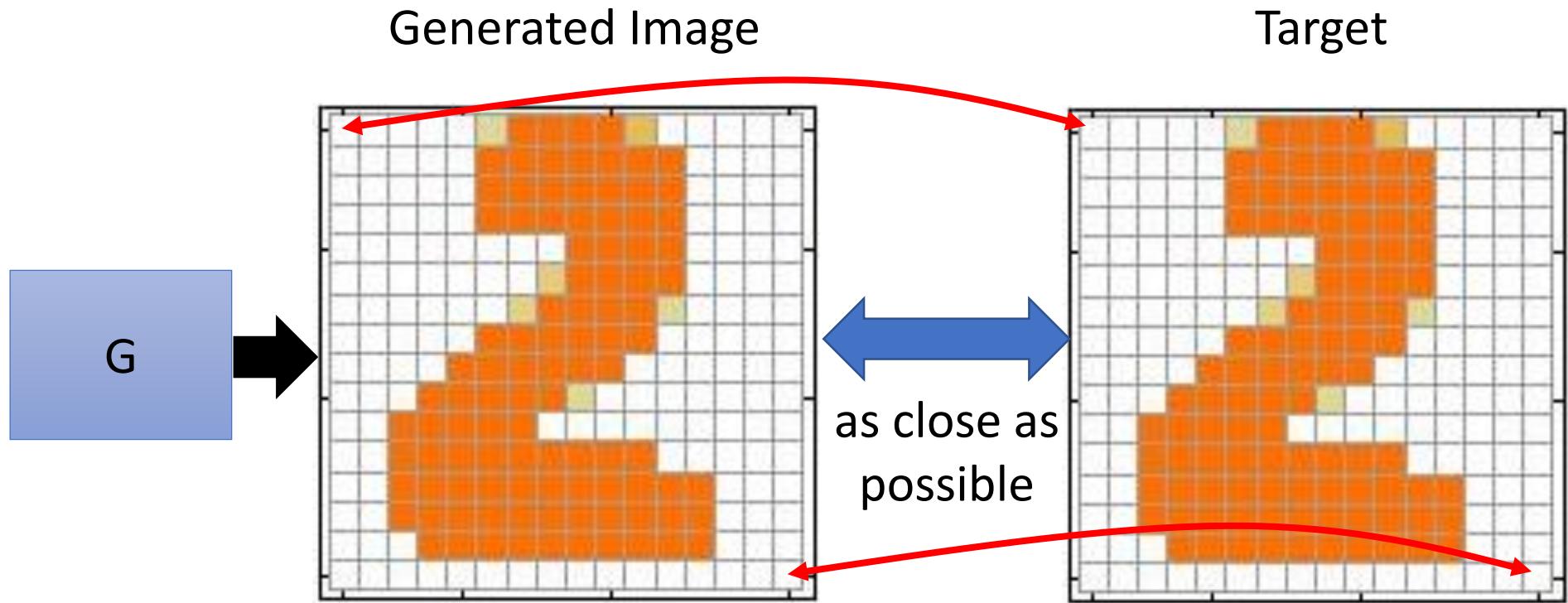
Auto-encoder



Variational Auto-encoder (VAE)



What do we miss?



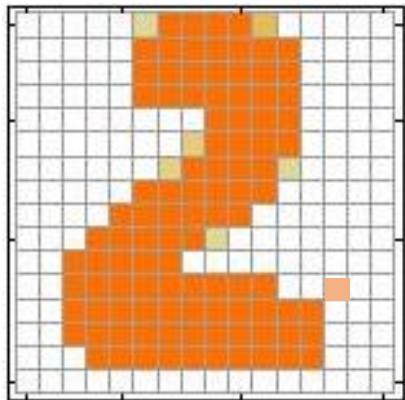
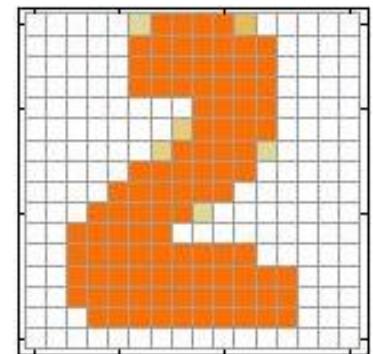
It will be fine if the generator can truly copy the target image.

What if the generator makes some mistakes

Some mistakes are serious, while some are fine.

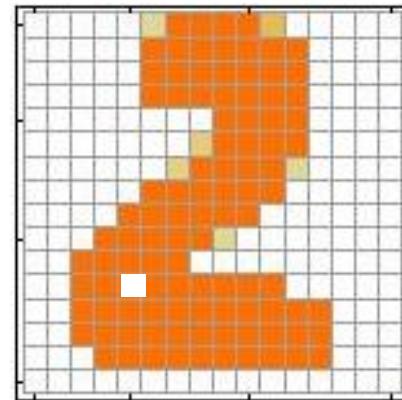
What do we miss?

Target



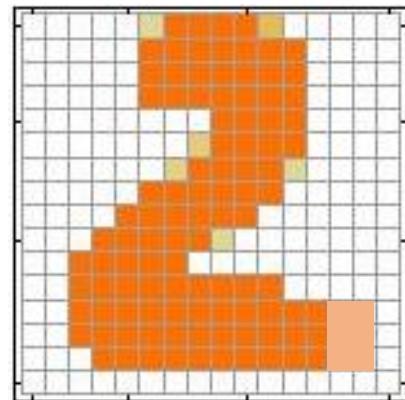
1 pixel error

我覺得不行



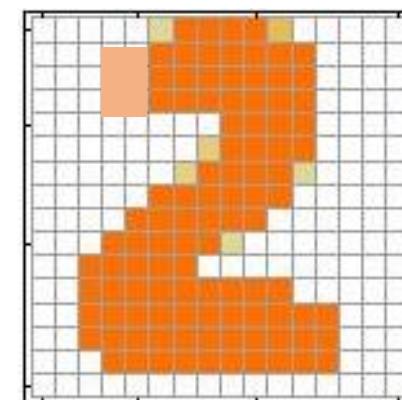
1 pixel error

我覺得不行



6 pixel errors

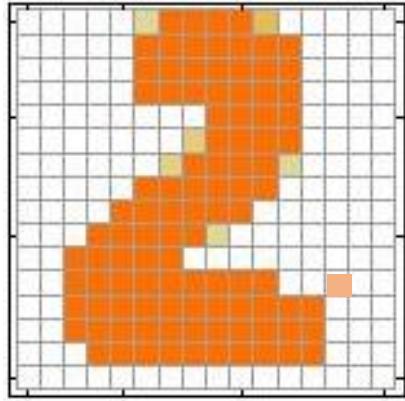
我覺得其實
可以



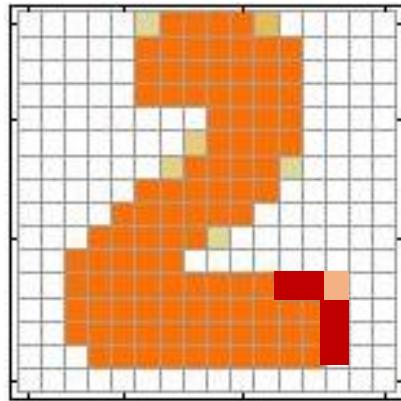
6 pixel errors

我覺得其實
可以

What do we miss?

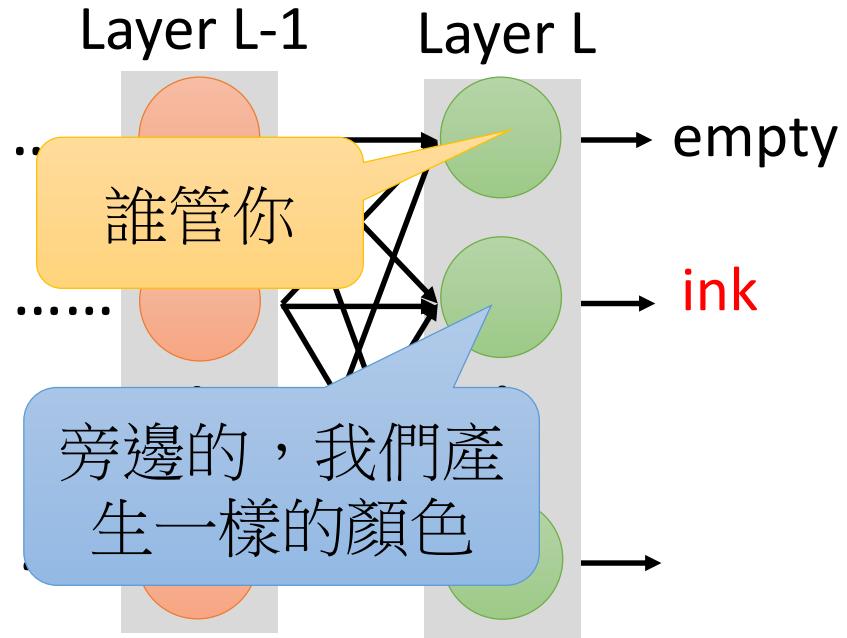


我覺得不行



我覺得其實可以

Each neural in output layer corresponds to a pixel.

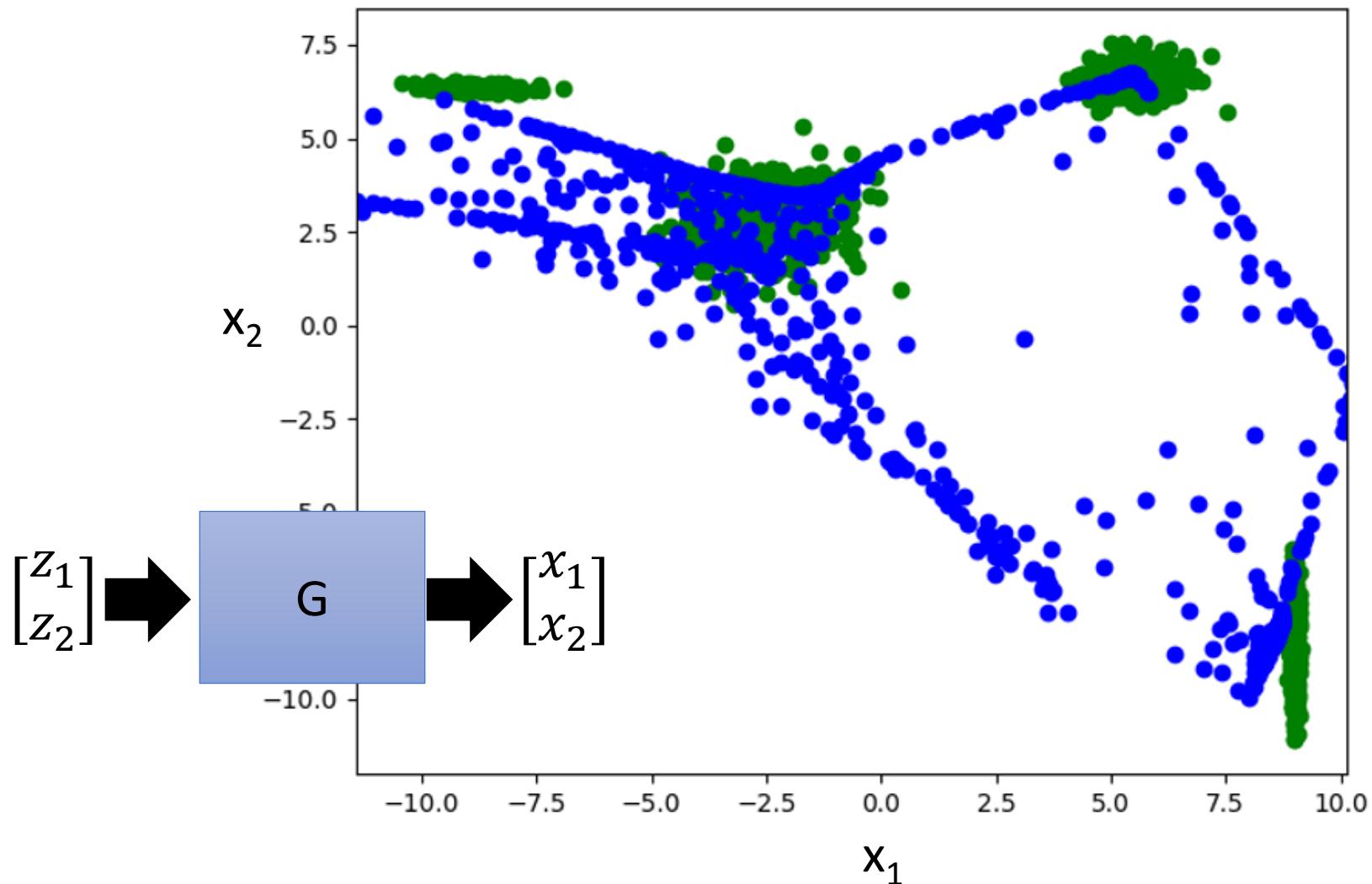


The relation between the components are critical.

Although highly correlated, they cannot influence each other.

Need deep structure to catch the relation between components.

(Variational) Auto-encoder



Outline

Basic Idea of GAN

GAN as structured learning

Can Generator learn by itself?

Can Discriminator generate?

A little bit theory

Discriminator

Evaluation function, Potential Function, Energy Function ...

- Discriminator is a function D (network, can deep)

$$D: X \rightarrow \mathbb{R}$$

- Input x : an object x (e.g. an image)
- Output $D(x)$: scalar which represents how “good” an object x is

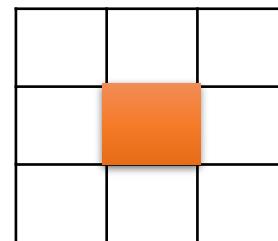
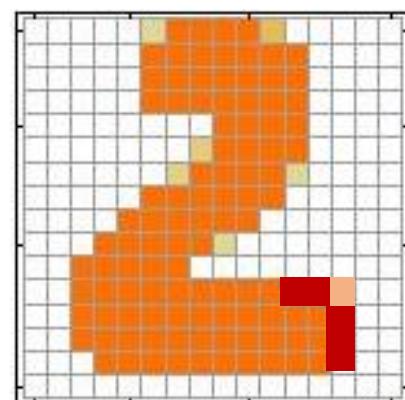
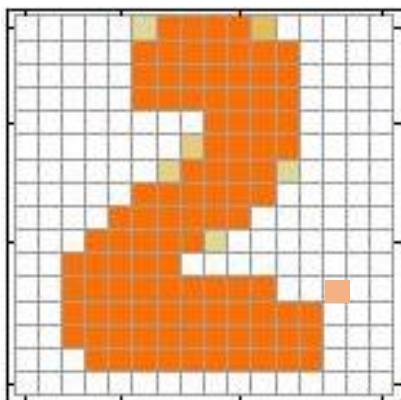


Can we use the discriminator to generate objects?

Yes.

Discriminator

- It is easier to catch the relation between the components by top-down evaluation.



This CNN filter is
good enough.

Discriminator

- Suppose we already have a good discriminator
 $D(x)$...

Inference

- Generate object \tilde{x} that

$$\tilde{x} = \arg \max_{x \in X} D(x)$$

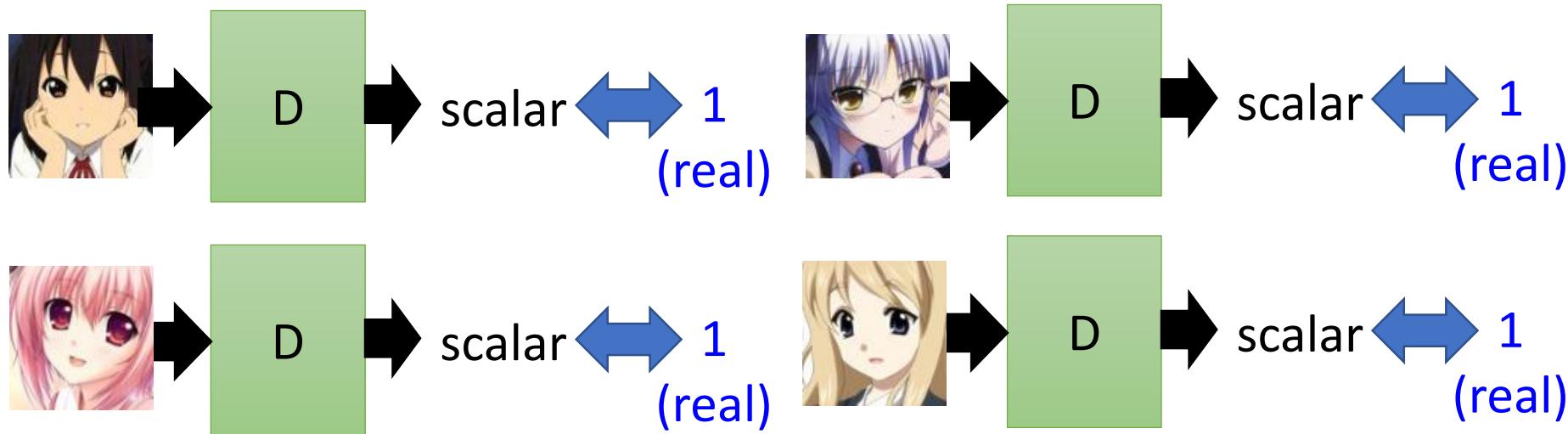
Enumerate all possible x !!!

It is feasible ???

How to learn the discriminator?

Discriminator - Training

- I have some real images

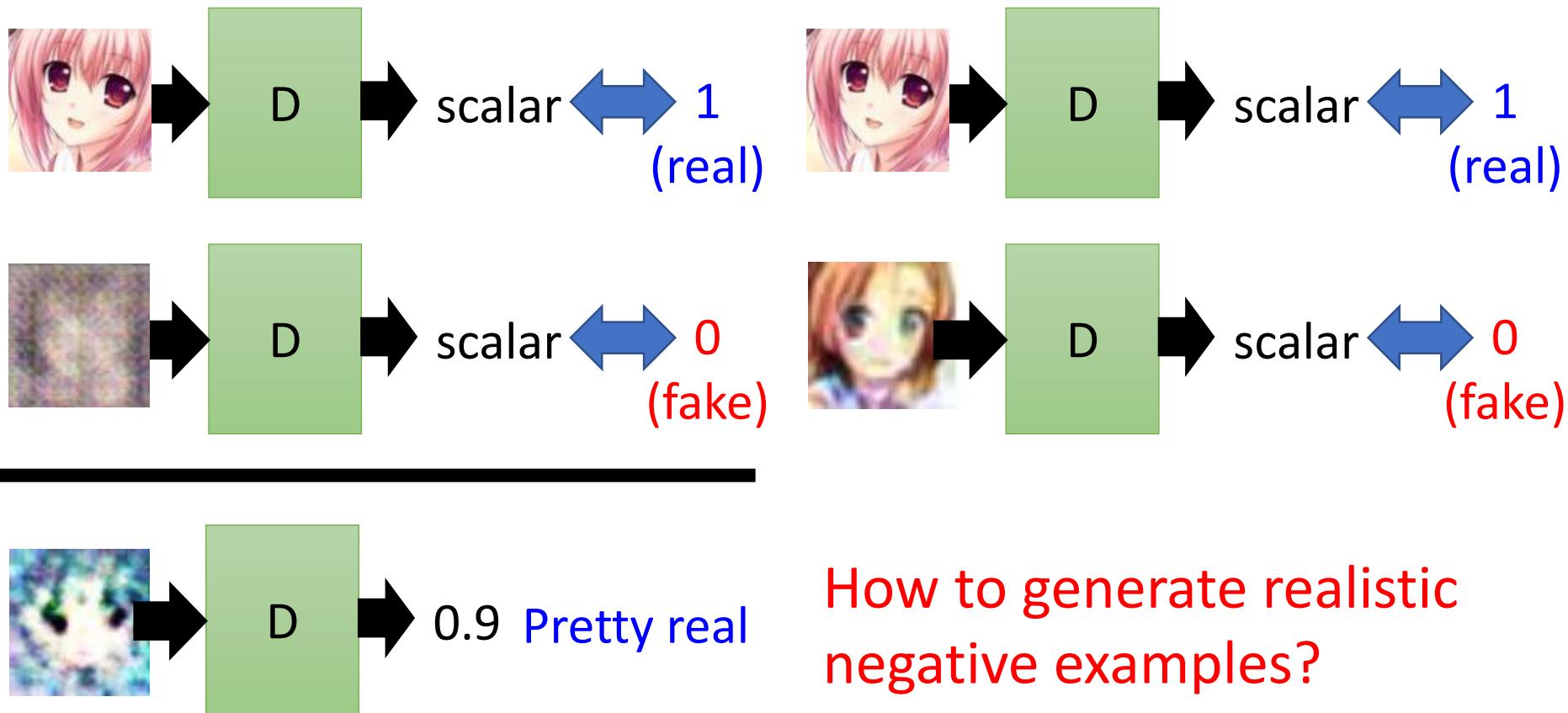


Discriminator only learns to output “1” (real).

Discriminator training needs some negative examples.

Discriminator - Training

- Negative examples are critical.



Discriminator - Training

- General Algorithm

- Given a set of **positive examples**, randomly generate a set of **negative examples**.



- In each iteration

- Learn a discriminator D that can discriminate positive and negative examples.



v.s.



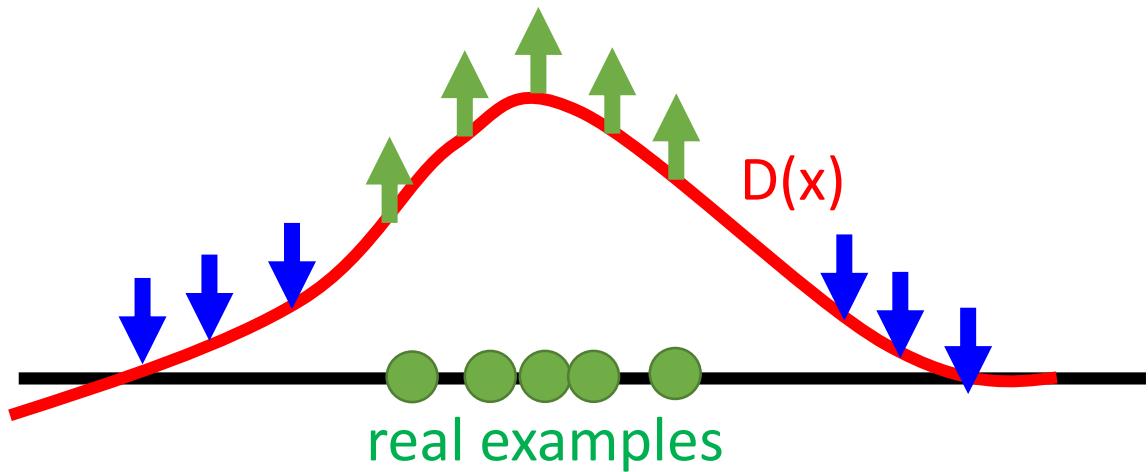
D

- Generate negative examples by discriminator D



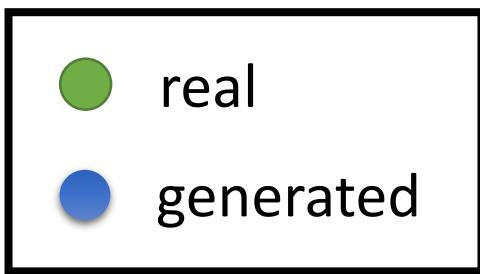
$$\tilde{x} = \arg \max_{x \in X} D(x)$$

Discriminator - Training

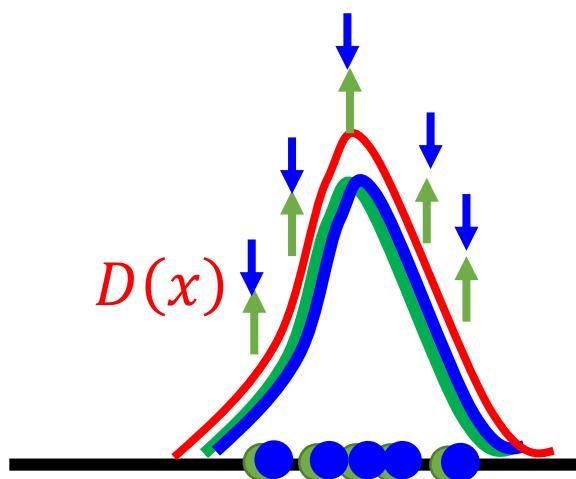
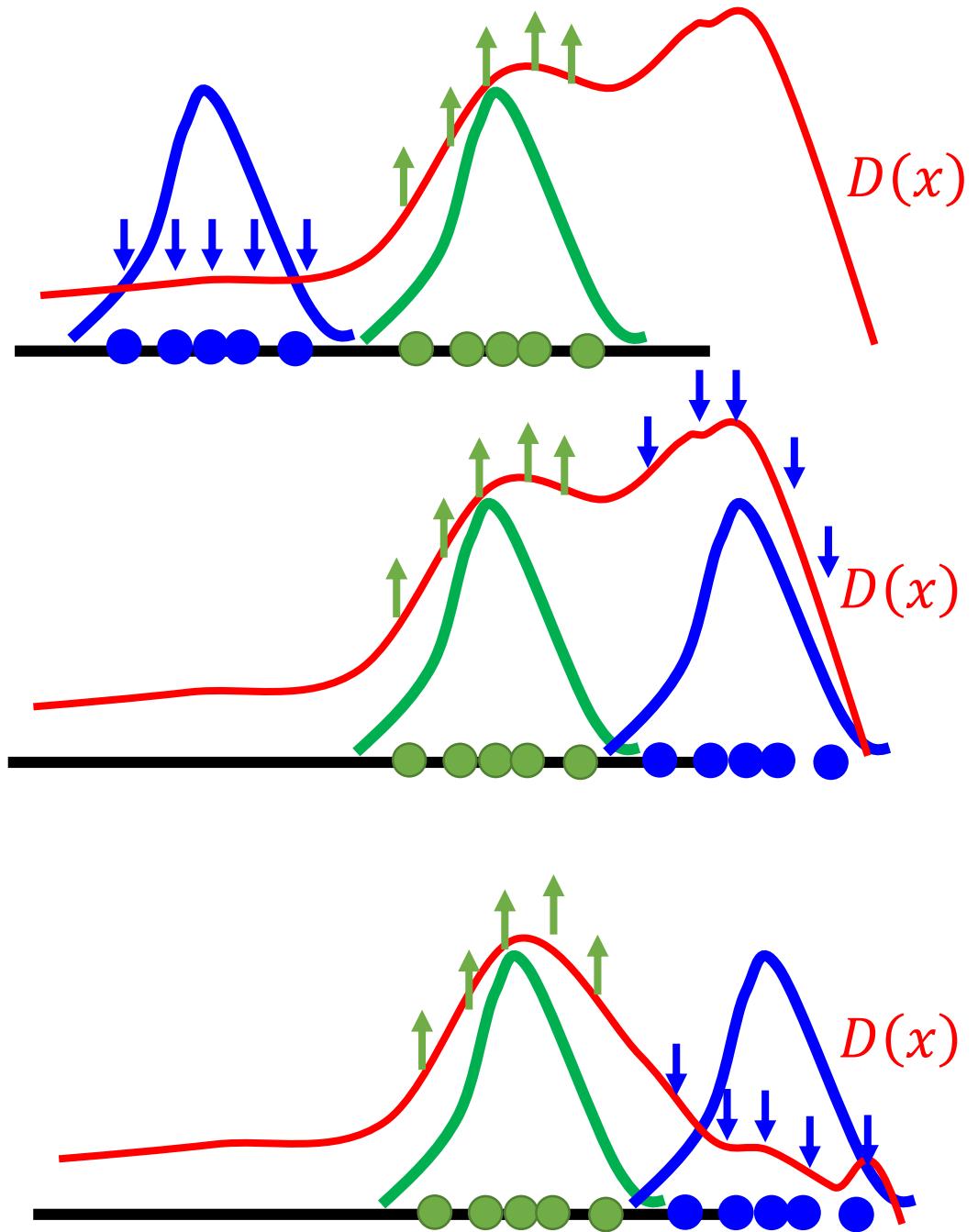


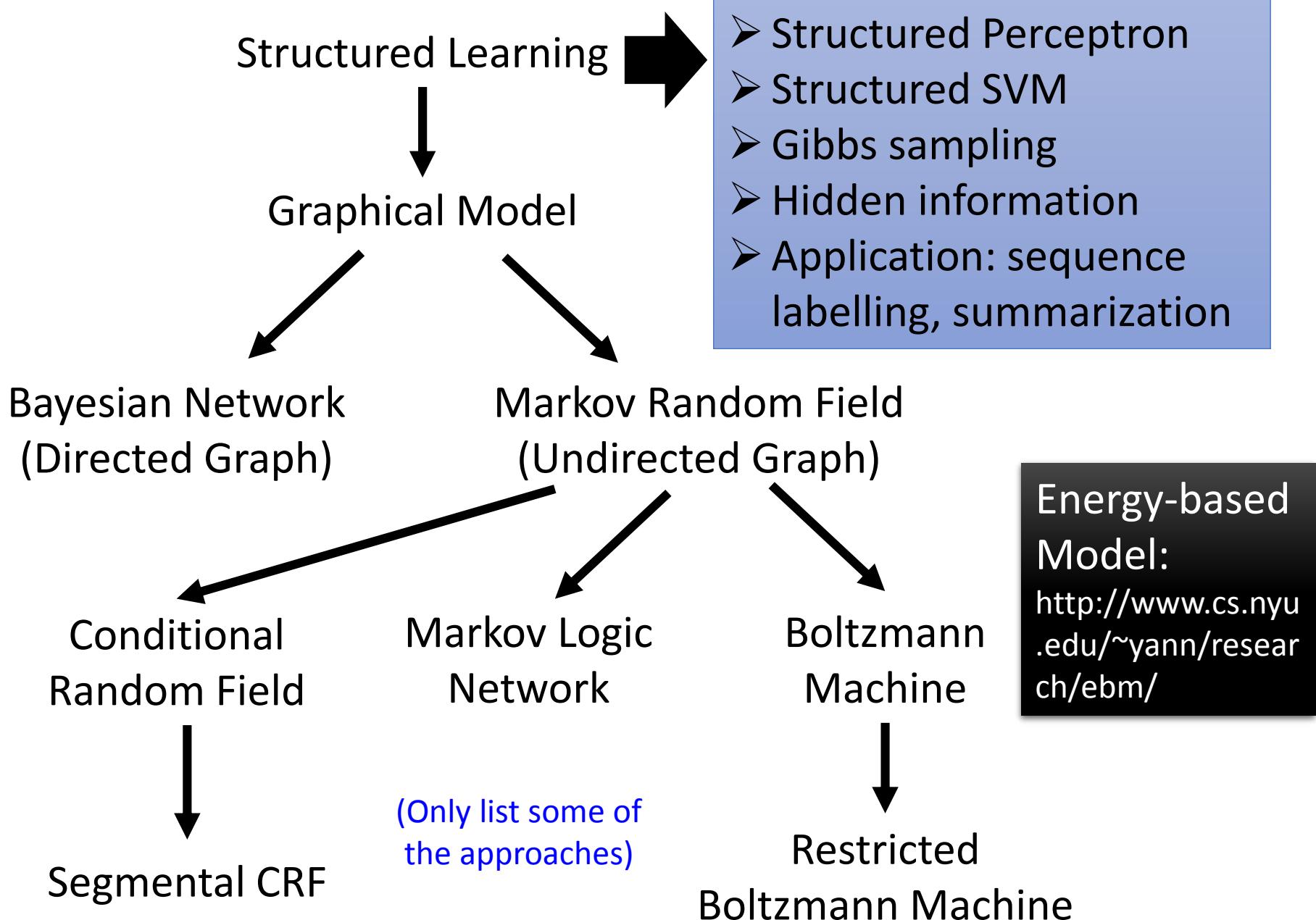
In practice, you cannot decrease all the x other than real examples.

Discriminator - Training



In the end





Generator v.s. Discriminator

- **Generator**

- Pros:
 - Easy to generate even with deep model
- Cons:
 - Imitate the appearance
 - Hard to learn the correlation between components

- **Discriminator**

- Pros:
 - Considering the big picture
- Cons:
 - Generation is not always feasible
 - Especially when your model is deep
 - How to do negative sampling?

Generator + Discriminator

- General Algorithm

- Given a set of **positive examples**, randomly generate a set of **negative examples**.



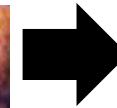
- In each iteration



- Learn a discriminator D that can discriminate positive and negative examples.



v.s.



D

- Generate negative examples by discriminator D

$$\boxed{G \rightarrow \tilde{x}}$$

$$= \boxed{\tilde{x} = \arg \max_{x \in X} D(x)}$$

Benefit of GAN

- From Discriminator's point of view
 - Using generator to generate negative samples

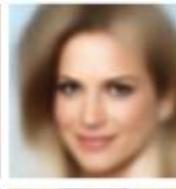
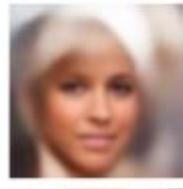
$$\boxed{\begin{array}{c} \text{G} \\ \longrightarrow \tilde{x} \end{array}} = \boxed{\tilde{x} = \arg \max_{x \in X} D(x)}$$

efficient

- From Generator's point of view
 - Still generate the object component-by-component
 - But it is learned from the discriminator with global view.

GAN

VAE

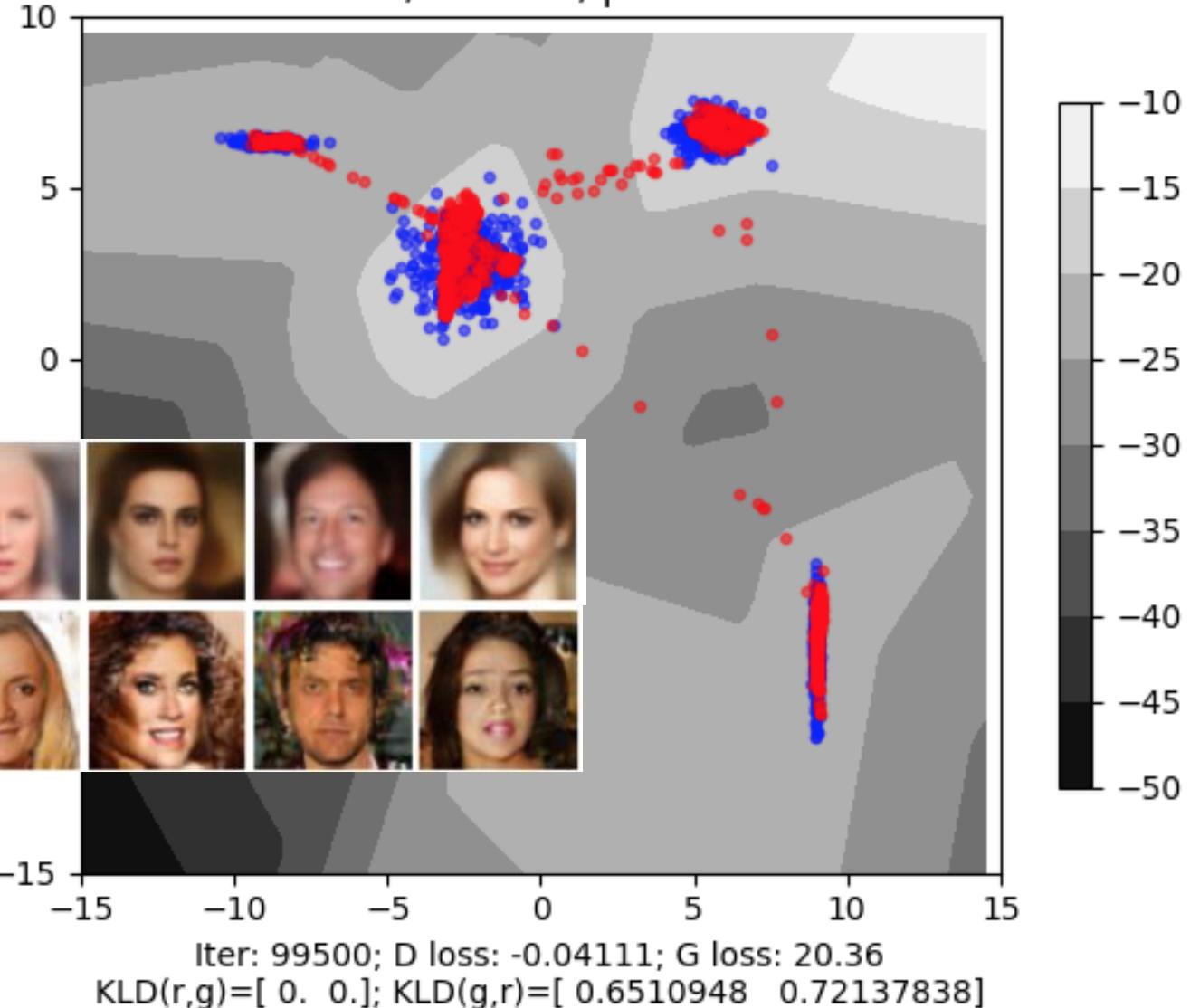


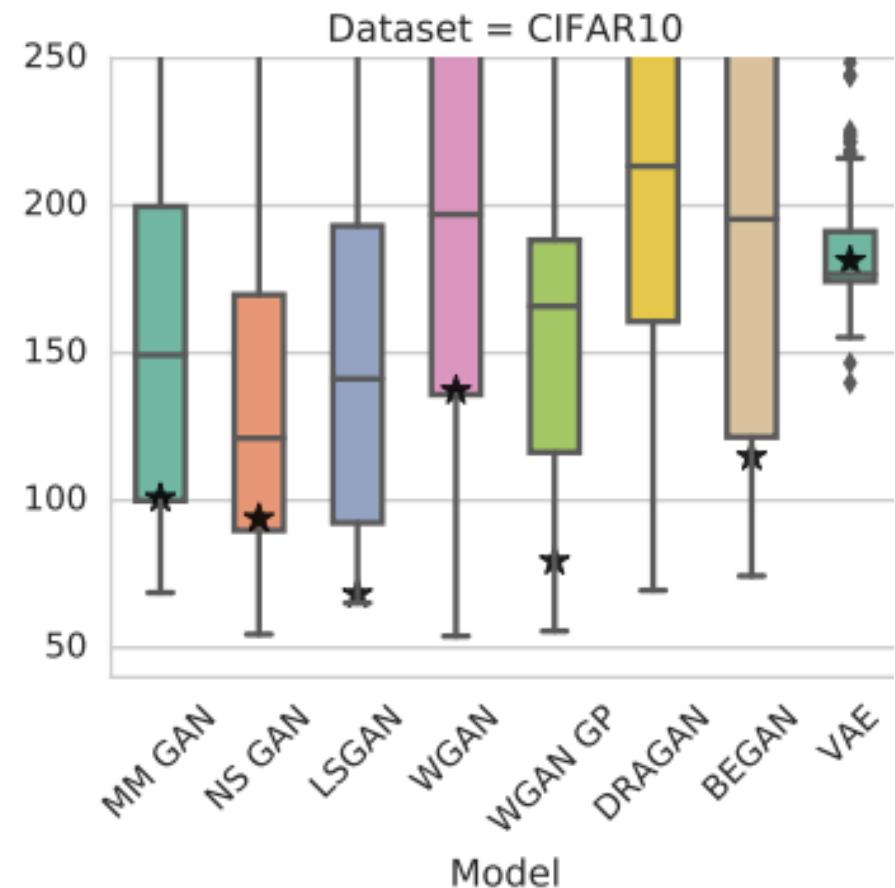
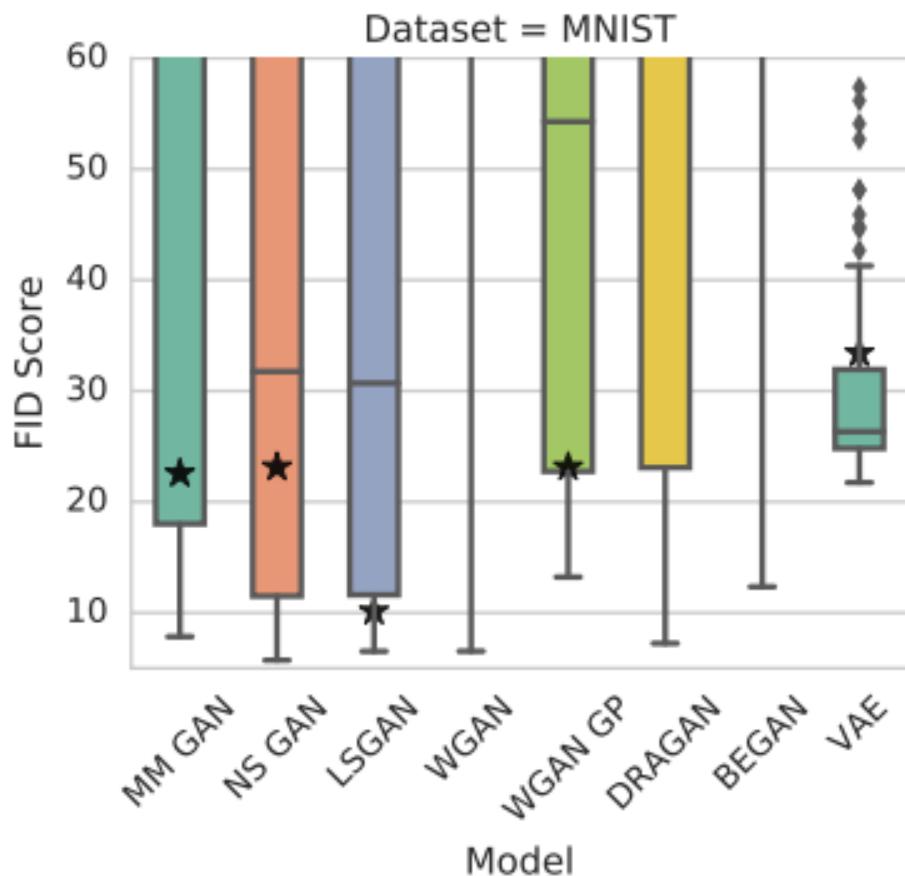
GAN



<https://arxiv.org/abs/1512.09300>

wgan-gp-sub1000-gauss4
Samples and Decision Boundary
G: 2*20; D: 4*10; prior dim: 2





FID [[Martin Heusel, et al., NIPS, 2017](#)]: Smaller is better

Next Time

- Preview
 - <https://youtu.be/0CKeqXI5IY0>
 - <https://youtu.be/KSN4QYgAtao>

Outline

Basic Idea of GAN

GAN as structured learning

Can Generator learn by itself?

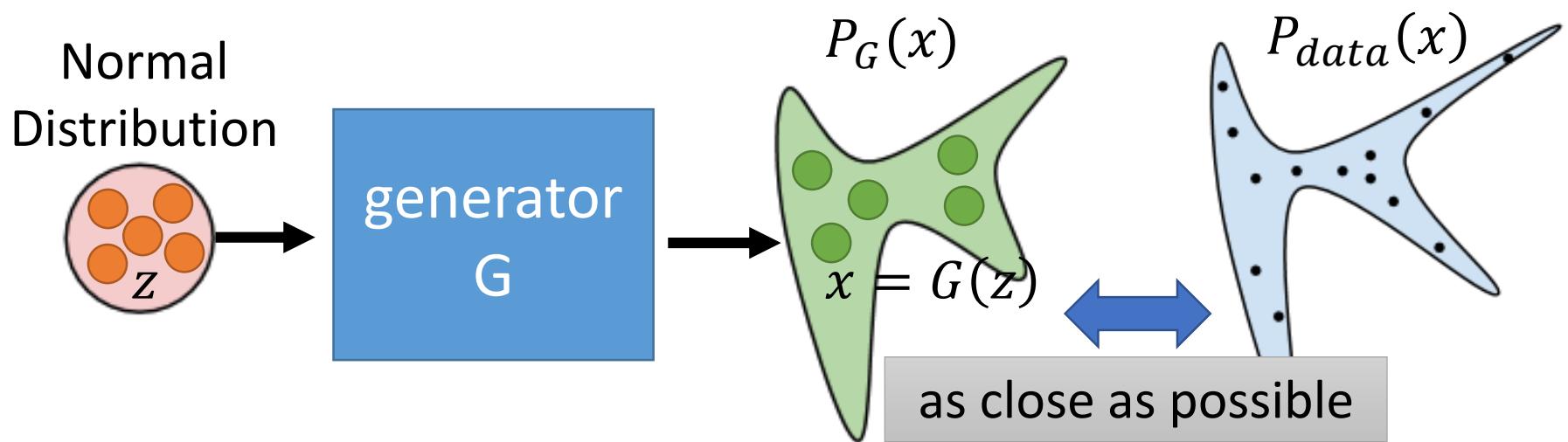
Can Discriminator generate?

A little bit theory

Generator

x : an image (a high-dimensional vector)

- A generator G is a network. The network defines a probability distribution P_G



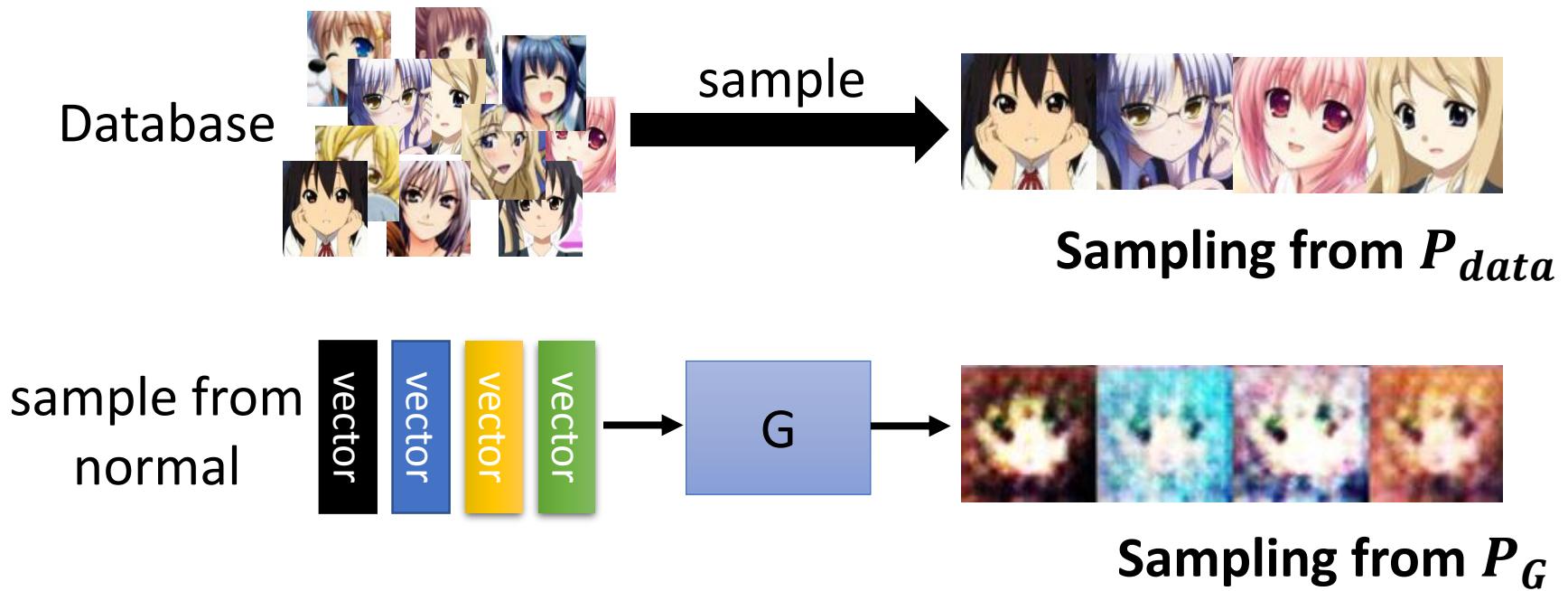
$$G^* = \arg \min_G \underline{\text{Div}}(P_G, P_{data})$$

Divergence between distributions P_G and P_{data}
How to compute the divergence?

Discriminator

$$G^* = \arg \min_G \text{Div}(P_G, P_{data})$$

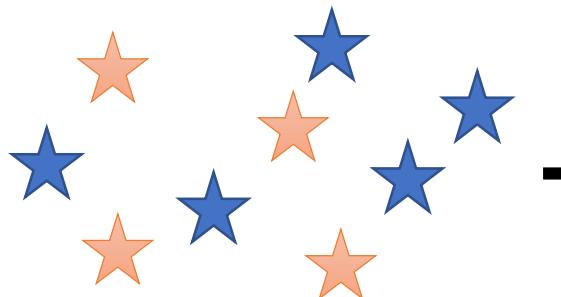
Although we do not know the distributions of P_G and P_{data} , we can sample from them.



Discriminator

$$G^* = \arg \min_G \text{Div}(P_G, P_{data})$$

- ★ : data sampled from P_{data}
- ☆ : data sampled from P_G



Using the example objective function is exactly the same as training a binary classifier.

Example Objective Function for D

$$V(G, D) = E_{x \sim P_{data}} [\log D(x)] + E_{x \sim P_G} [\log(1 - D(x))]$$

↑
(G is fixed)

Training: $D^* = \arg \max_D V(D, G)$

The maximum objective value is related to JS divergence.

Discriminator

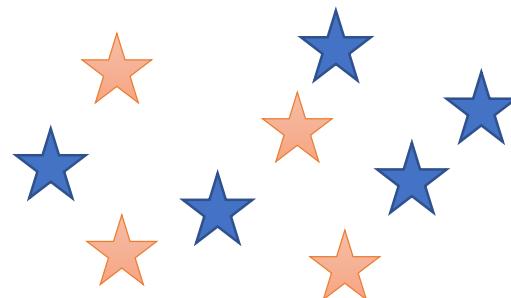
$$G^* = \arg \min_G \text{Div}(P_G, P_{data})$$

★ : data sampled from P_{data}

☆ : data sampled from P_G

Training:

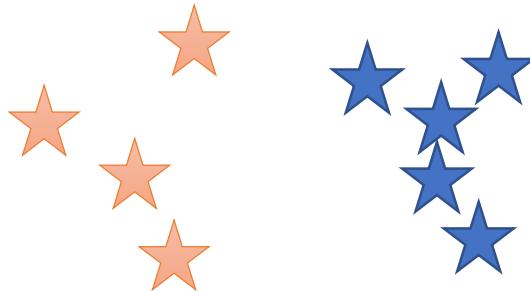
$$D^* = \arg \max_D V(D, G)$$



small divergence

train

Discriminator



large divergence

train

hard to discriminate
(cannot make objective large)

Discriminator

easy to discriminate

$$G^* = \arg \min_G \max_D V(G, D)$$

$$D^* = \arg \max_D V(D, G)$$

The maximum objective value is related to JS divergence.

- Initialize generator and discriminator
- In each training iteration:

Step 1: Fix generator G , and update discriminator D

Step 2: Fix discriminator D , and update generator G

Can we use other divergence?

Name	$D_f(P\ Q)$	Generator $f(u)$
Total variation	$\frac{1}{2} \int p(x) - q(x) dx$	$\frac{1}{2} u - 1 $
Kullback-Leibler	$\int p(x) \log \frac{p(x)}{q(x)} dx$	$u \log u$
Reverse Kullback-Leibler	$\int q(x) \log \frac{q(x)}{p(x)} dx$	$-\log u$
Pearson χ^2	$\int \frac{(q(x)-p(x))^2}{p(x)} dx$	$(u - 1)^2$
Neyman χ^2	$\int \frac{(p(x)-q(x))^2}{q(x)} dx$	$\frac{(1-u)^2}{u}$
Squared Hellinger	$\int \left(\sqrt{p(x)} - \sqrt{q(x)} \right)^2 dx$	$(\sqrt{u} - 1)^2$
Jeffrey	$\int (p(x) - q(x)) \log \left(\frac{p(x)}{q(x)} \right) dx$	$(u - 1) \log u$
Jensen-Shannon	$\frac{1}{2} \int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} dx$	$-(u + 1) \log \frac{1+u}{2} + u \log u$
Jensen-Shannon-weighted	$\int p(x)\pi \log \frac{p(x)}{\pi p(x)+(1-\pi)q(x)} + (1-\pi)q(x) \log \frac{q(x)}{\pi p(x)+(1-\pi)q(x)} dx$	$\pi u \log u - (1 - \pi + \pi u) \log(1 - \pi + \pi u)$
GAN	$\int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} dx - \log(4)$	$u \log u - (u + 1) \log(u + 1)$

Name	Conjugate $f^*(t)$
Total variation	t
Kullback-Leibler (KL)	$\exp(t - 1)$
Reverse KL	$-1 - \log(-t)$
Pearson χ^2	$\frac{1}{4}t^2 + t$
Neyman χ^2	$2 - 2\sqrt{1-t}$
Squared Hellinger	$\frac{t}{1-t}$
Jeffrey	$W(e^{1-t}) + \frac{1}{W(e^{1-t})} + t - 2$
Jensen-Shannon	$-\log(2 - \exp(t))$
Jensen-Shannon-weighted	$(1 - \pi) \log \frac{1-\pi}{1-\pi e^{t/\pi}}$
GAN	$-\log(1 - \exp(t))$

Using the divergence
you like ☺

Conditional Generation by GAN

李宏毅

Hung-yi Lee

Text-to-Image

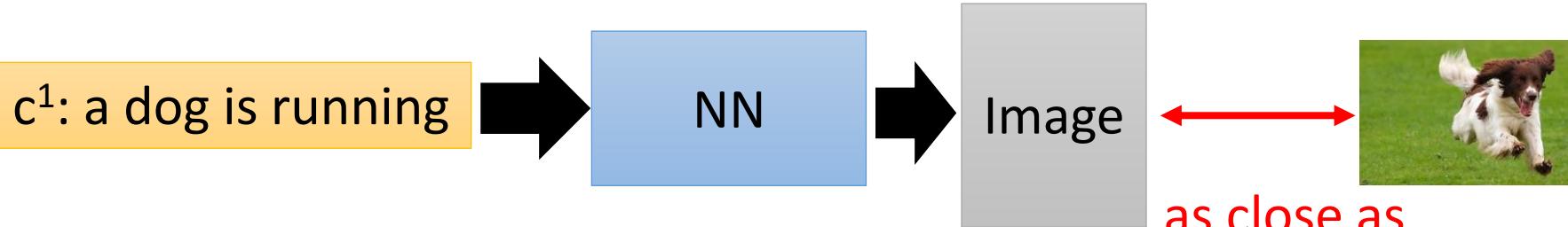
a dog is running



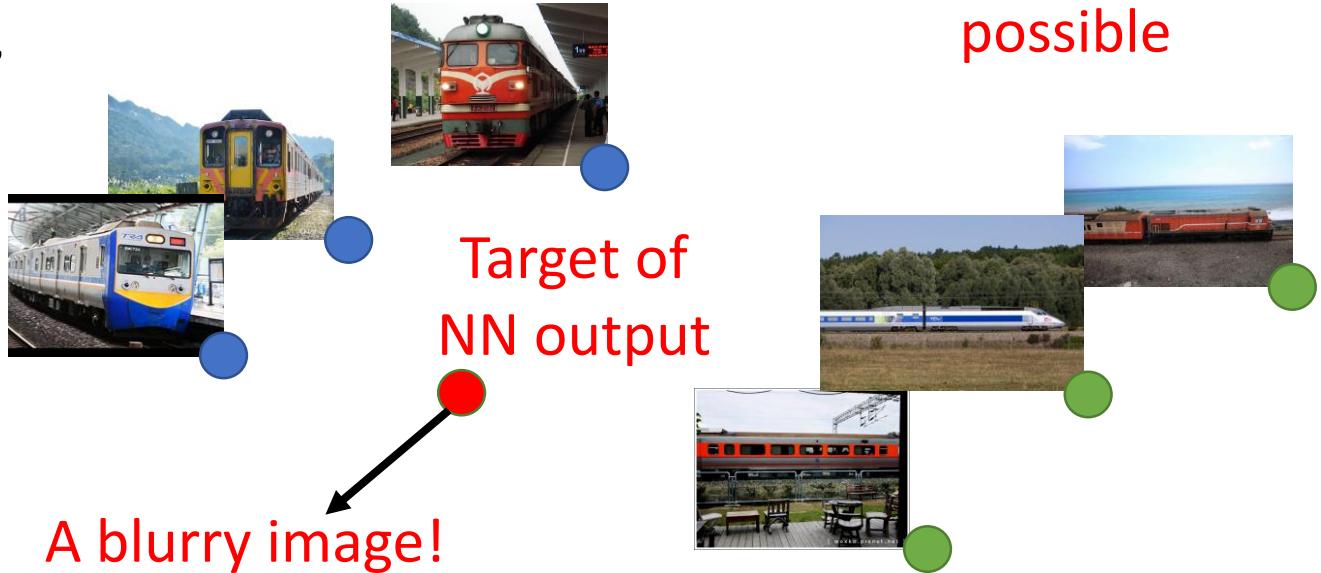
a bird is flying



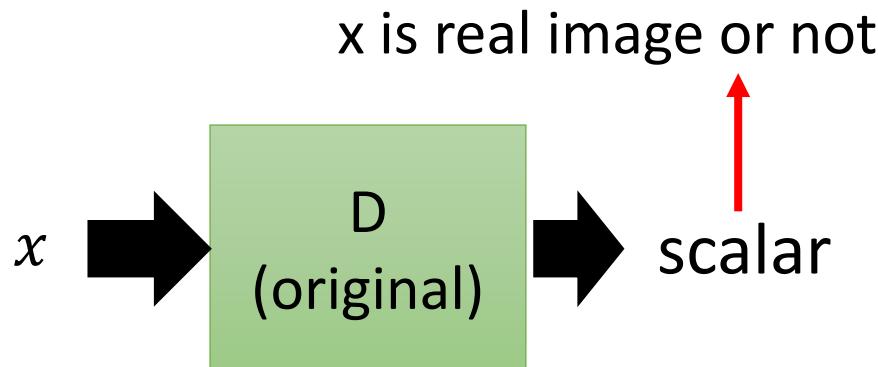
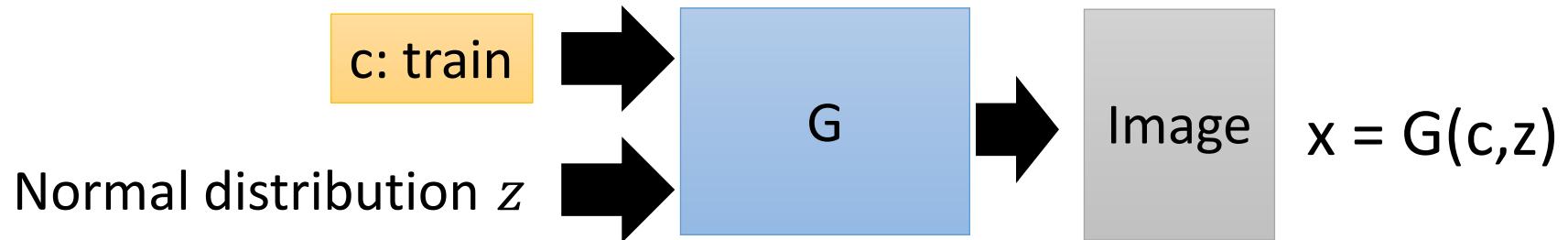
- Traditional supervised approach



Text: “train”

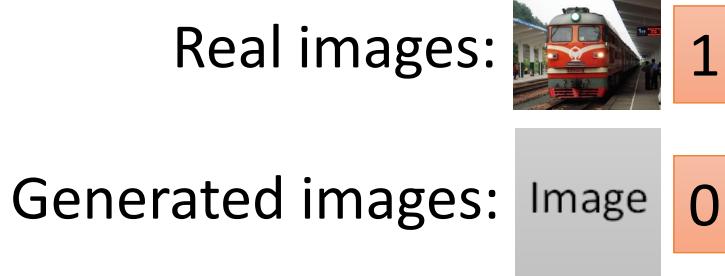


Conditional GAN

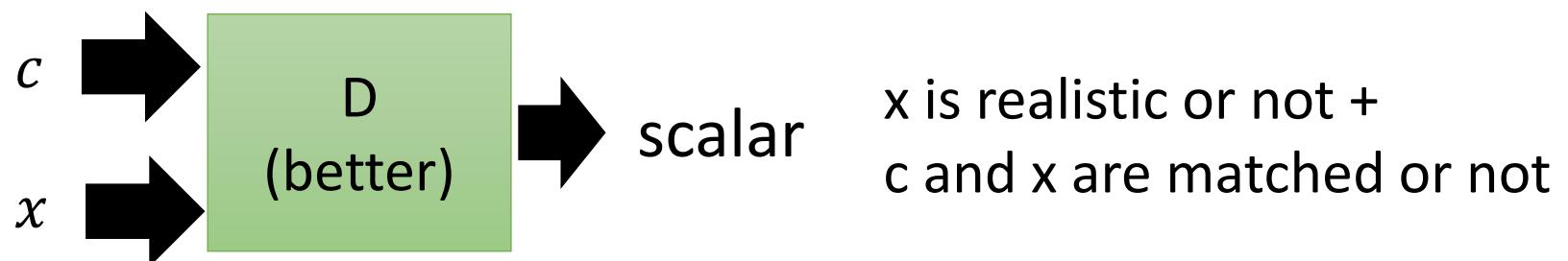
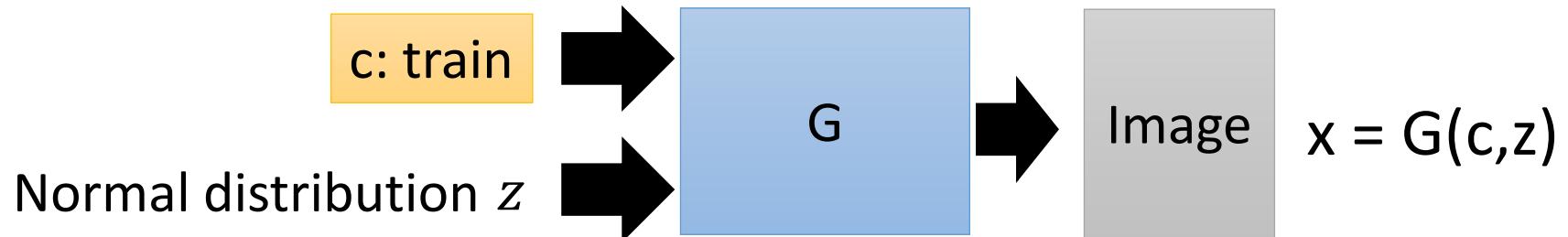


Generator will learn to
generate realistic images ...

But completely ignore the
input conditions.



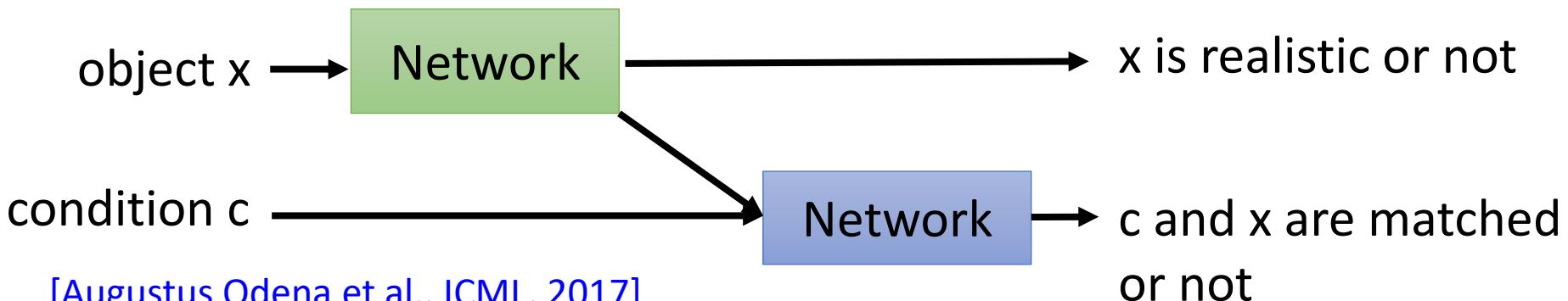
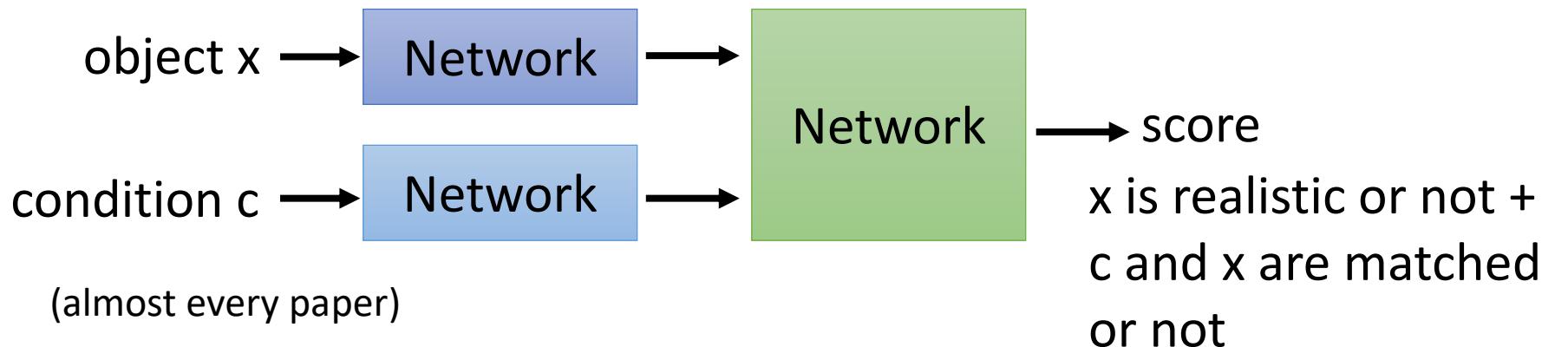
Conditional GAN



True text-image pairs: (train , ) 1

(cat , ) 0 (train , ) 0

Conditional GAN - Discriminator



[Augustus Odena et al., ICML, 2017]

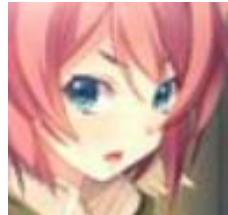
[Takeru Miyato, et al., ICLR, 2018]

[Han Zhang, et al., arXiv, 2017]

Conditional GAN

The images are generated by
Yen-Hao Chen, Po-Chun Chien,
Jun-Chen Xie, Tsung-Han Wu.

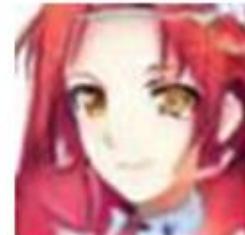
paired data



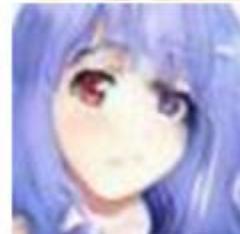
blue eyes
red hair
short hair

Collecting anime faces
and the description of its
characteristics

red hair,
green eyes



blue hair,
red eyes



Stack GAN

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, Dimitris Metaxas, "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks", ICCV, 2017

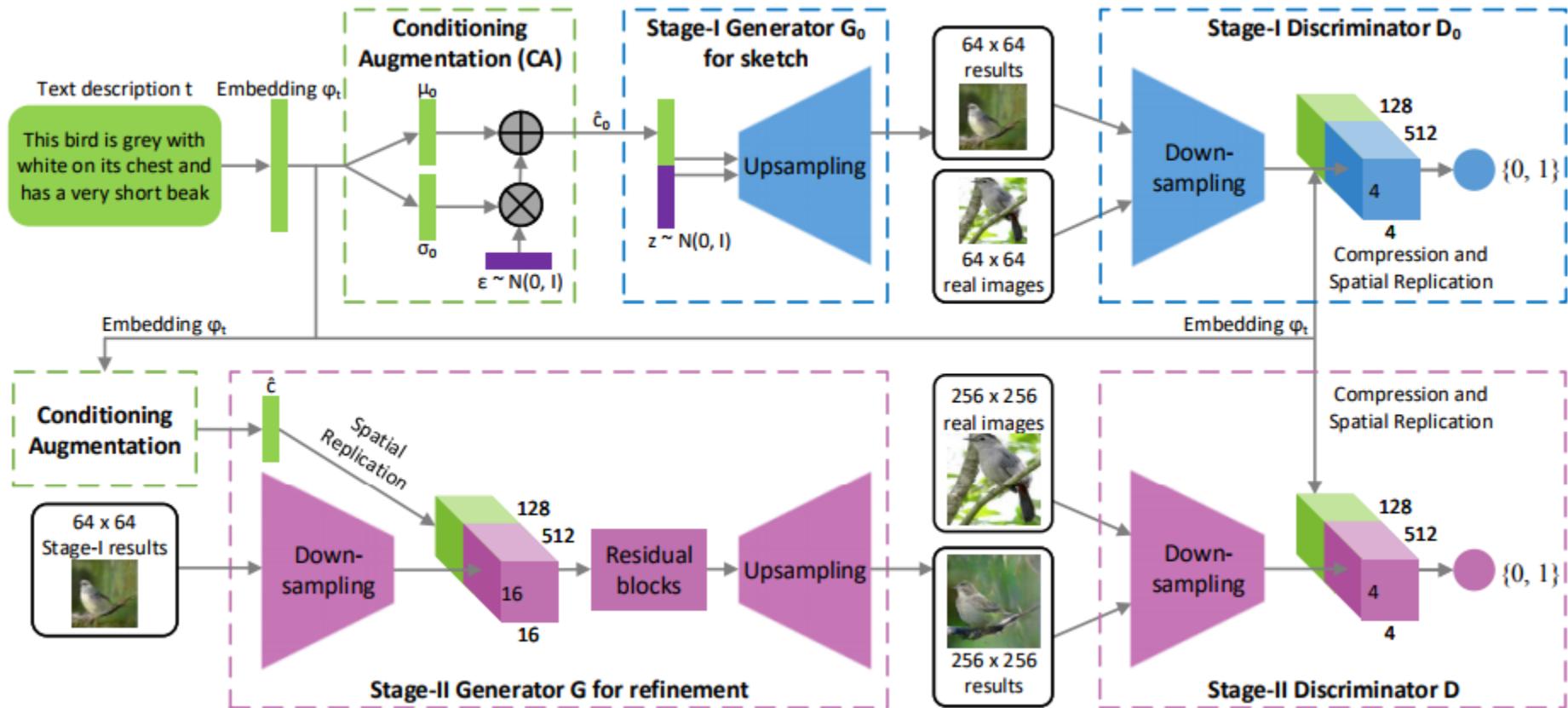
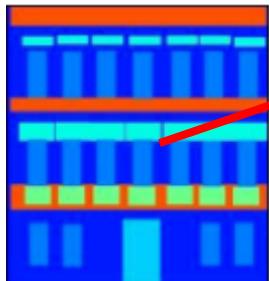


Image-to-image



c
 z



$$x = G(c, z)$$



Labels to Street Scene

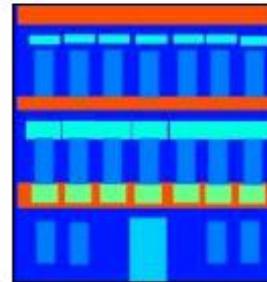


input



output

Labels to Facade



input



output

BW to Color



input



output

Aerial to Map



input



output

Day to Night



input



output

Edges to Photo

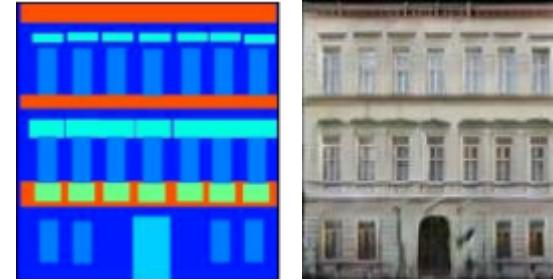


input

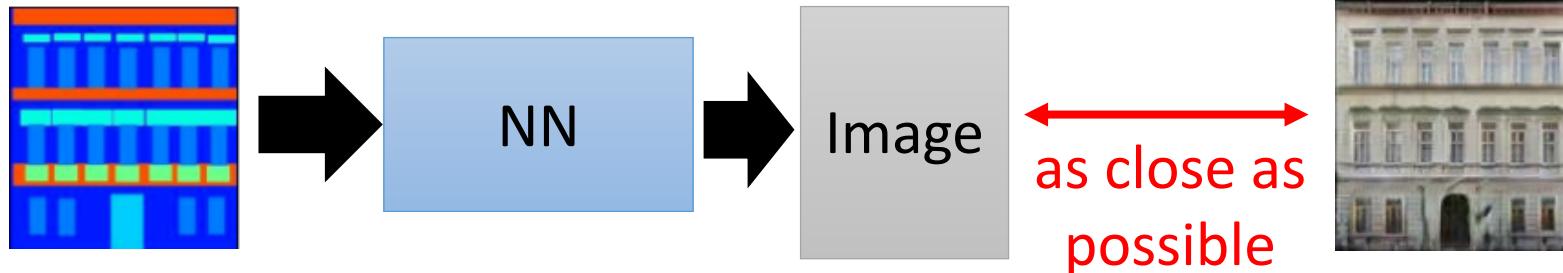


output

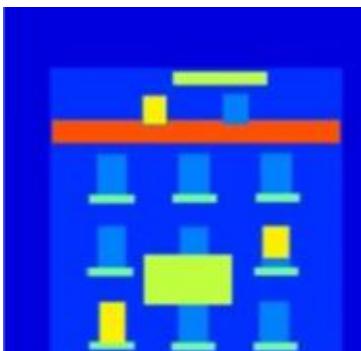
Image-to-image



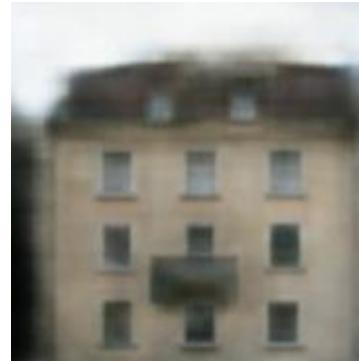
- Traditional supervised approach



Testing:



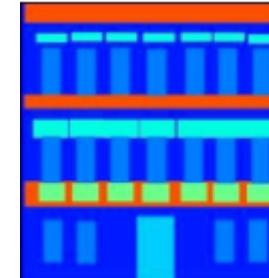
input



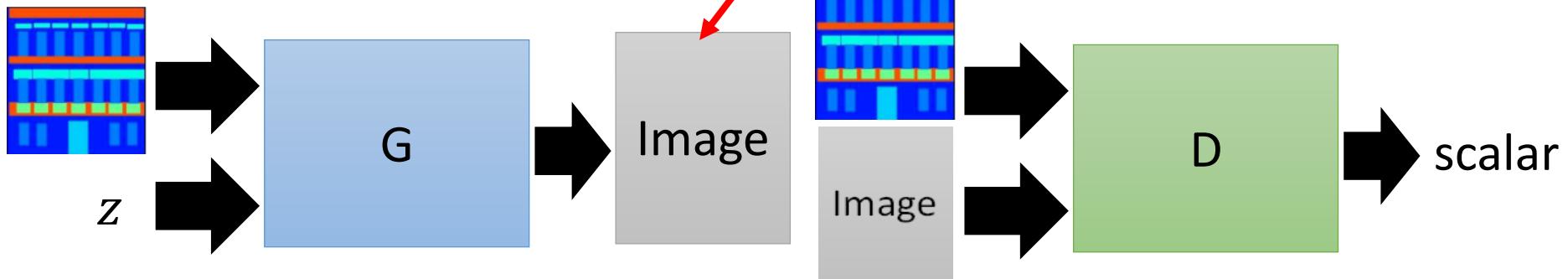
close

It is blurry because it is the average of several images.

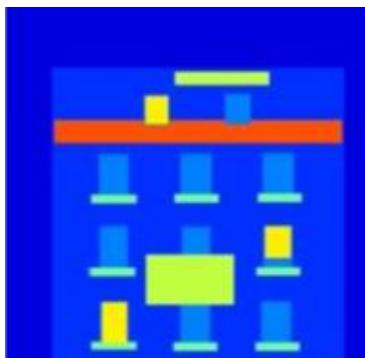
Image-to-image



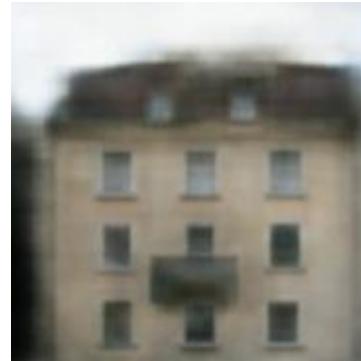
- Experimental results



Testing:



input



close



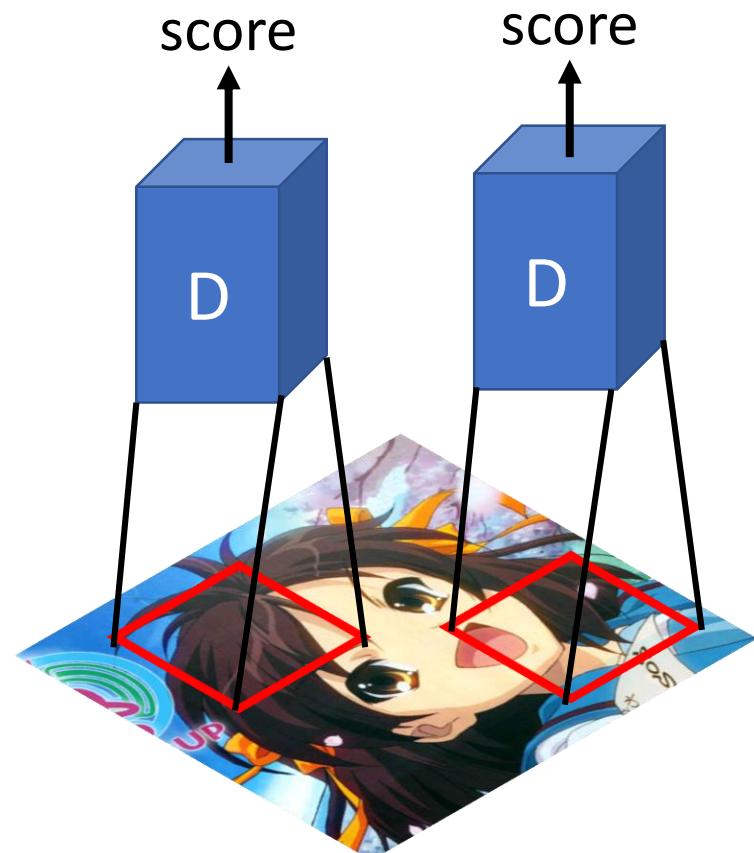
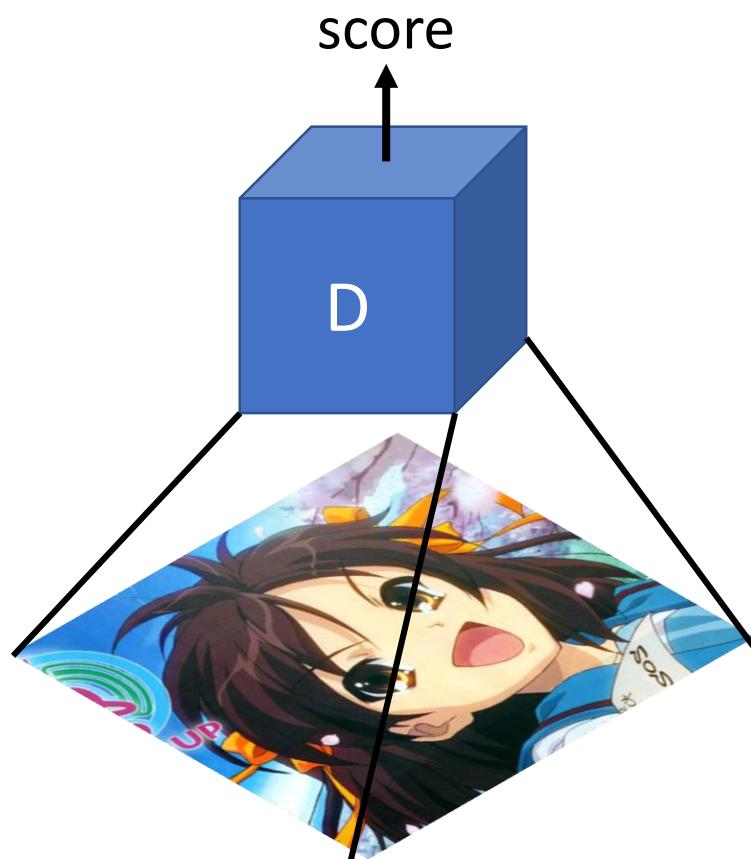
GAN



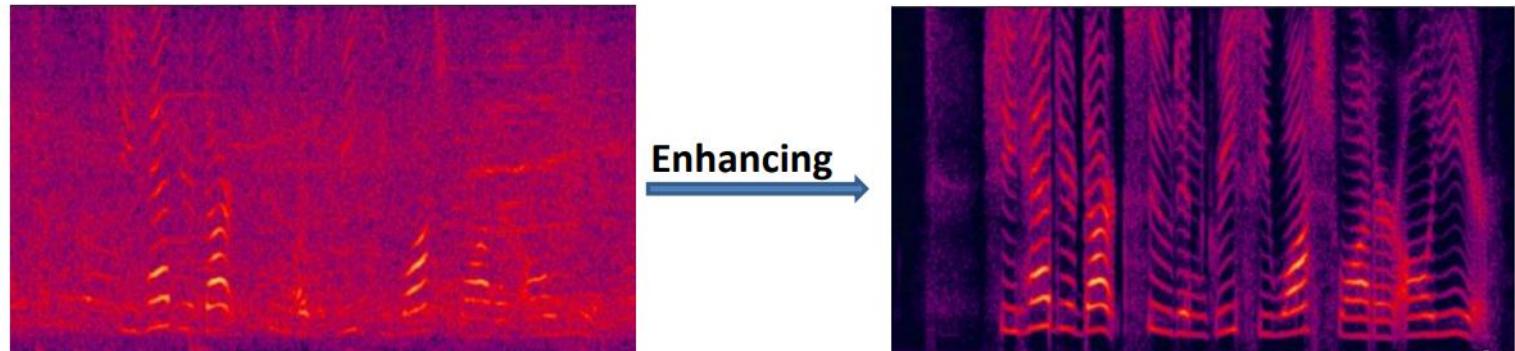
GAN + close

Patch GAN

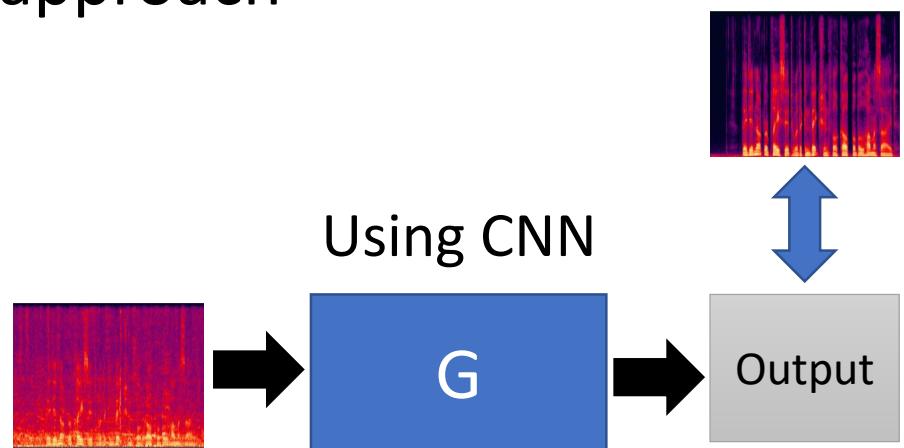
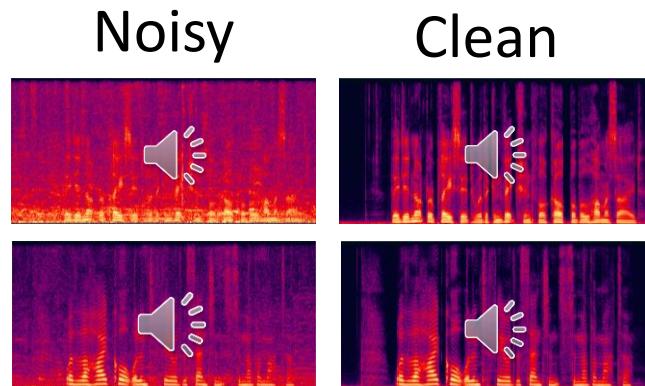
<https://arxiv.org/pdf/1611.07004.pdf>



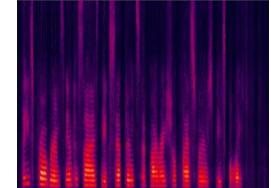
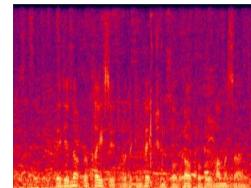
Speech Enhancement



- Typical deep learning approach

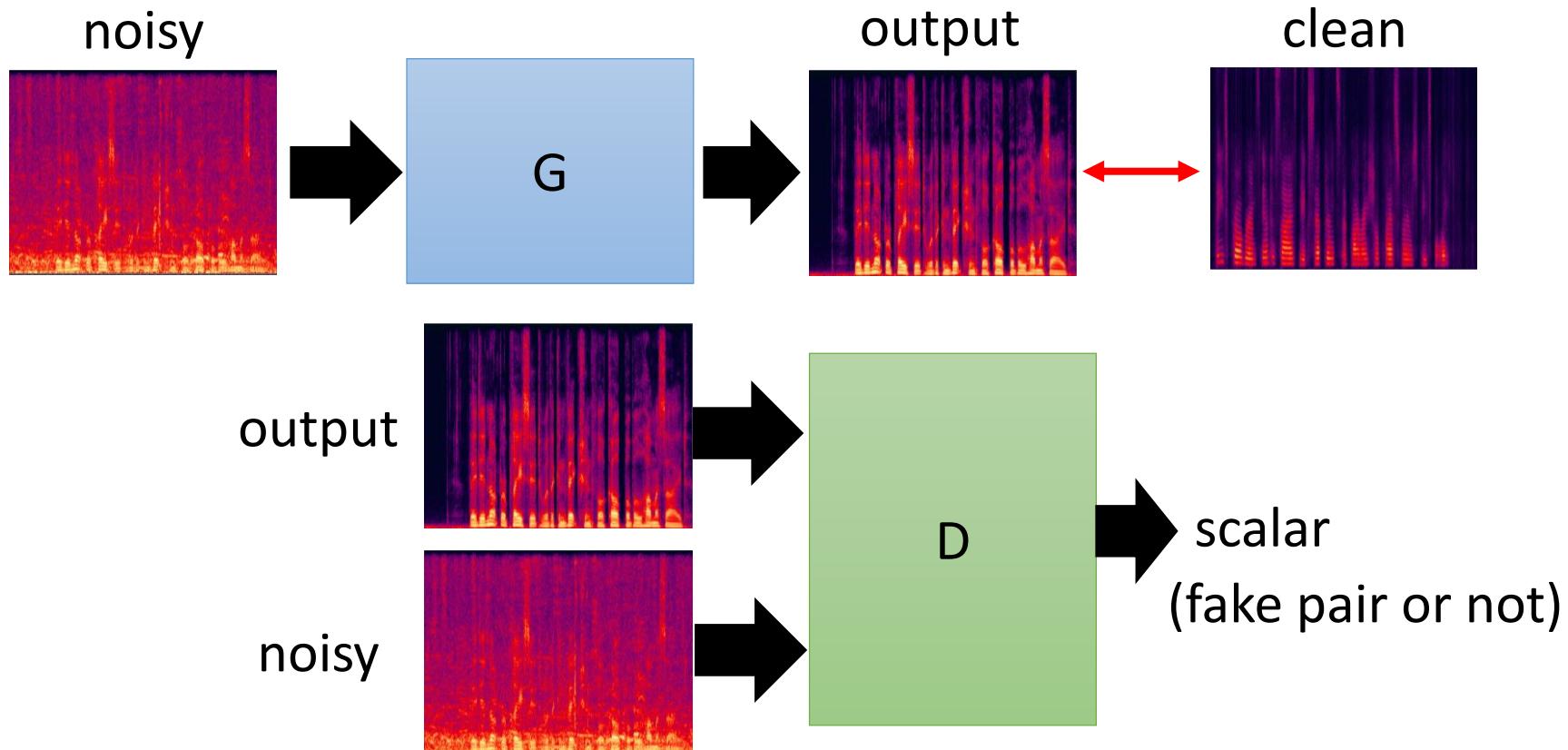


training data

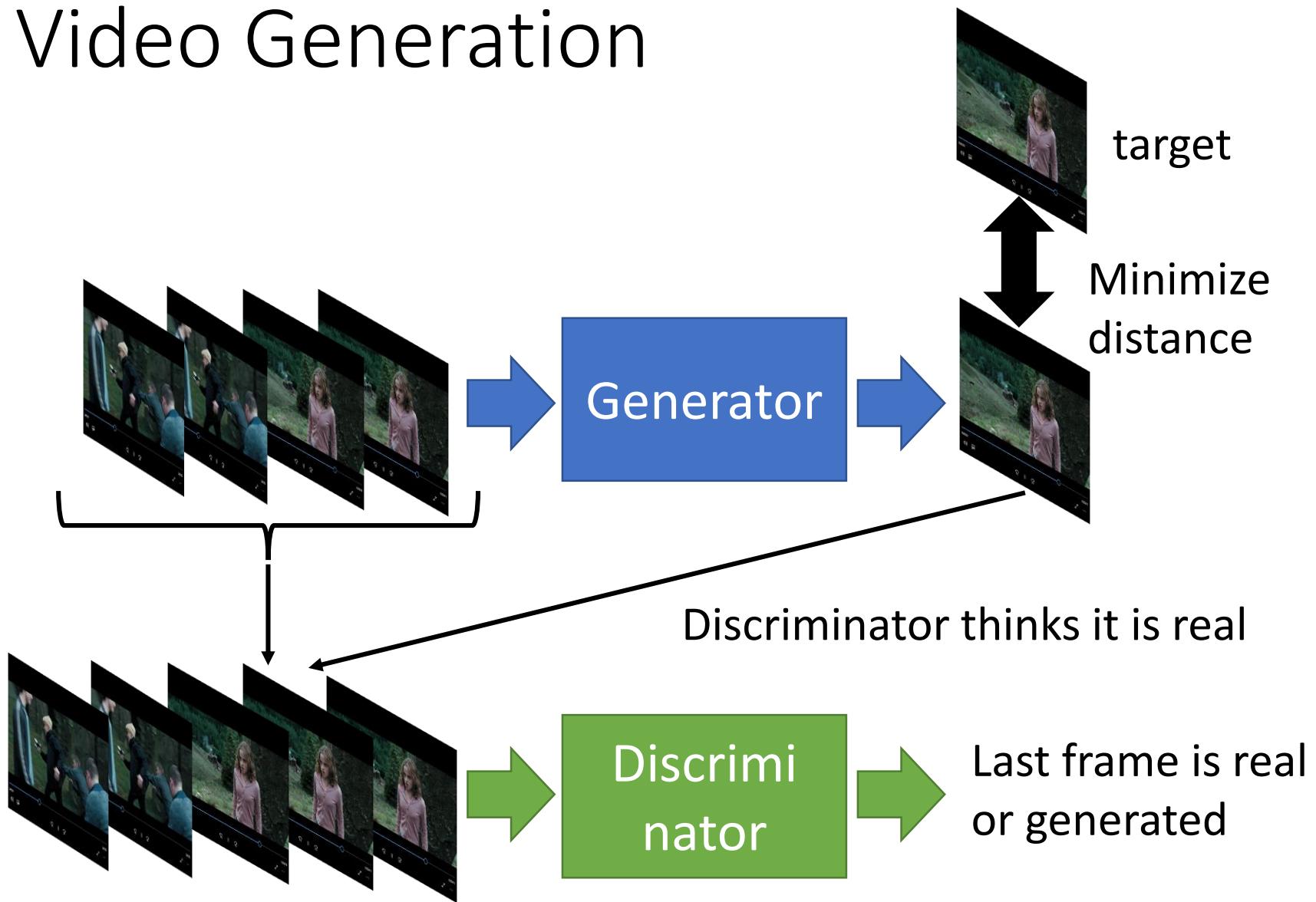


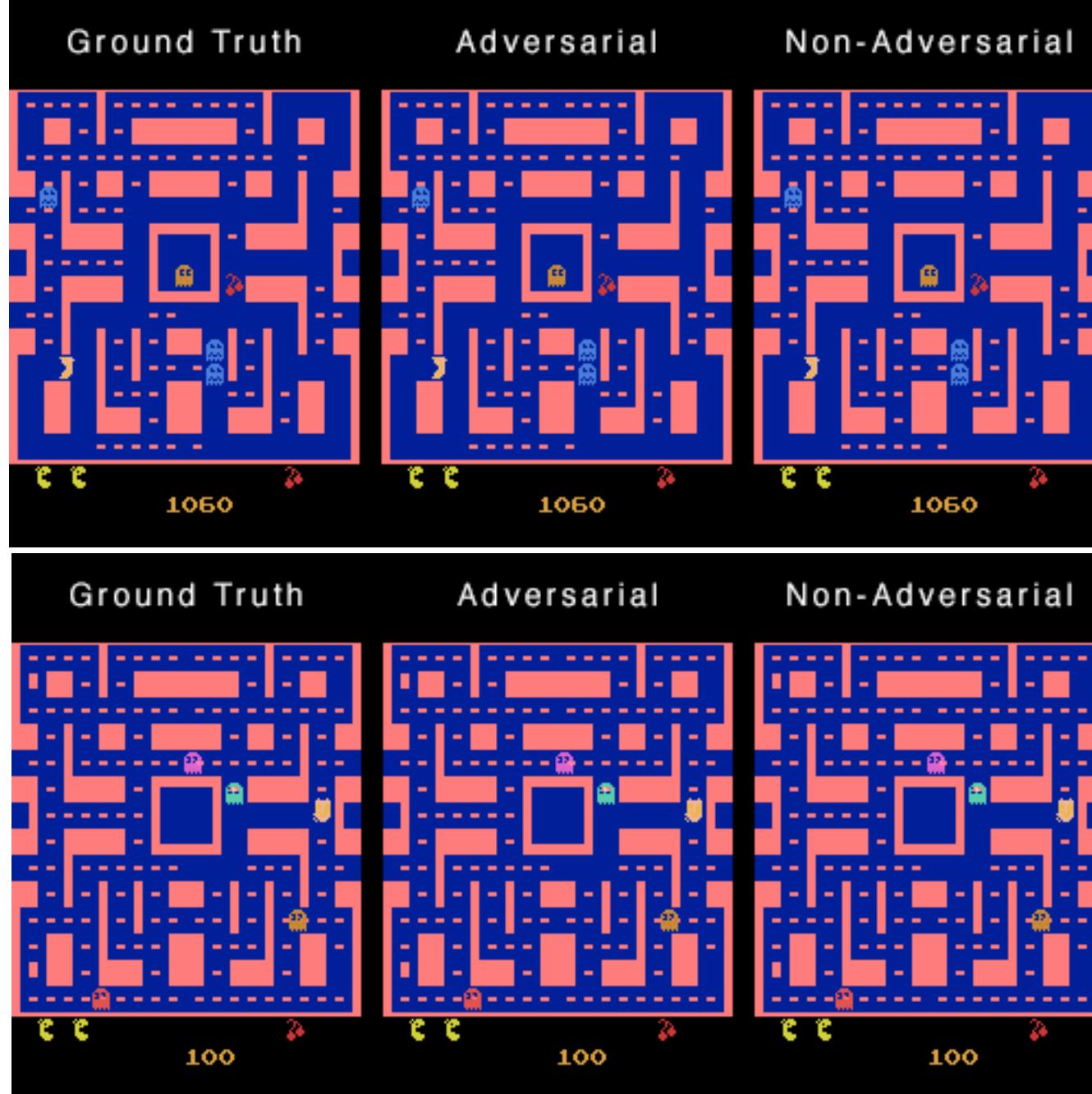
Speech Enhancement

- Conditional GAN



Video Generation



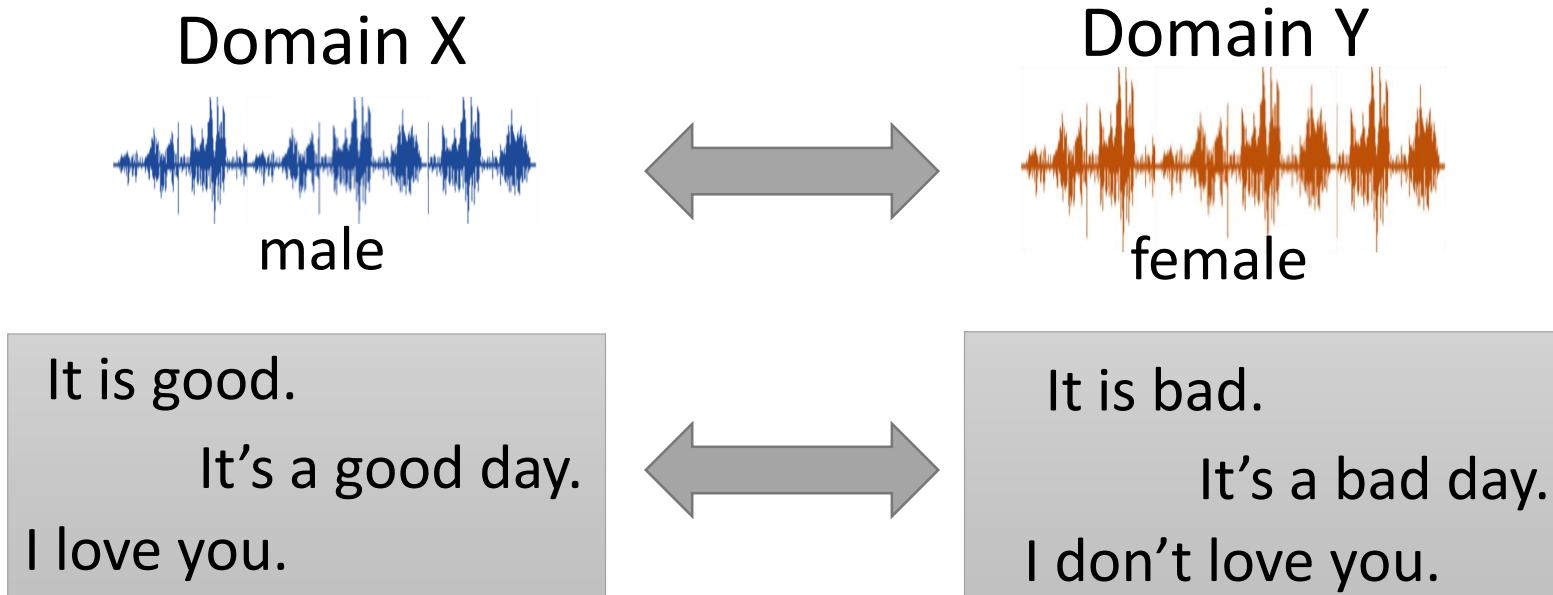


Unsupervised Conditional Generation

Unsupervised Conditional Generation

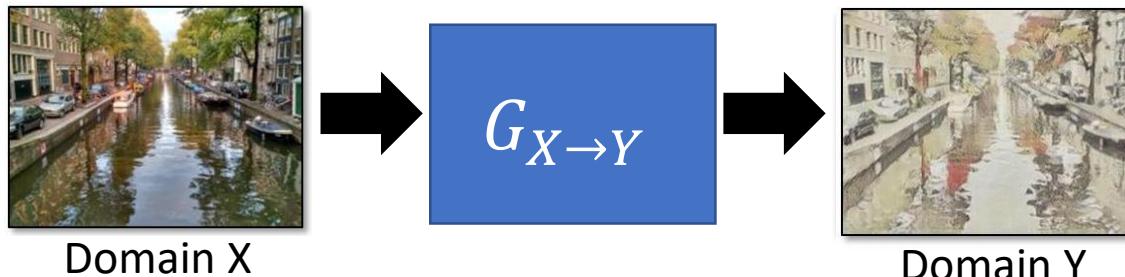


Transform an object from one domain to another
without paired data (e.g. style transfer)

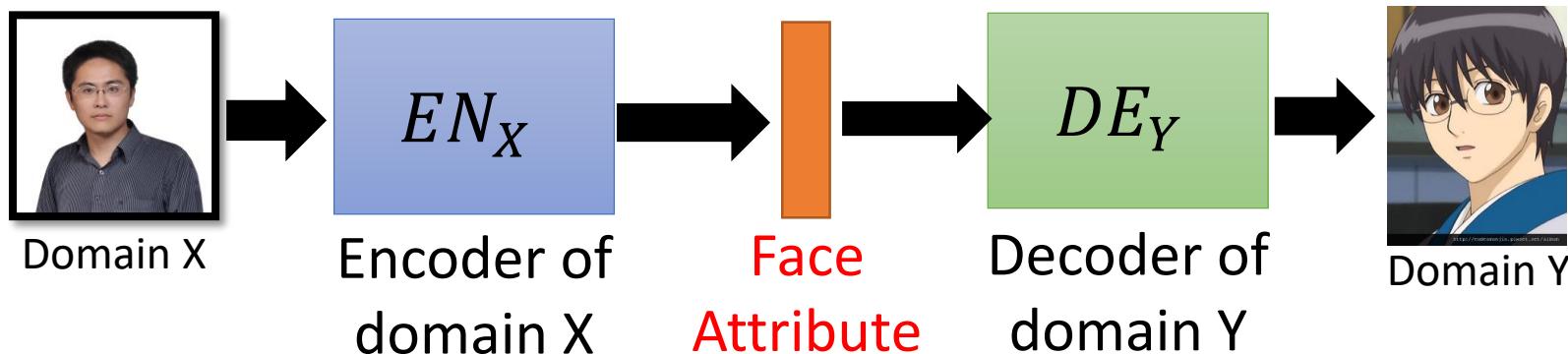


Unsupervised Conditional Generation

- Approach 1: Direct Transformation

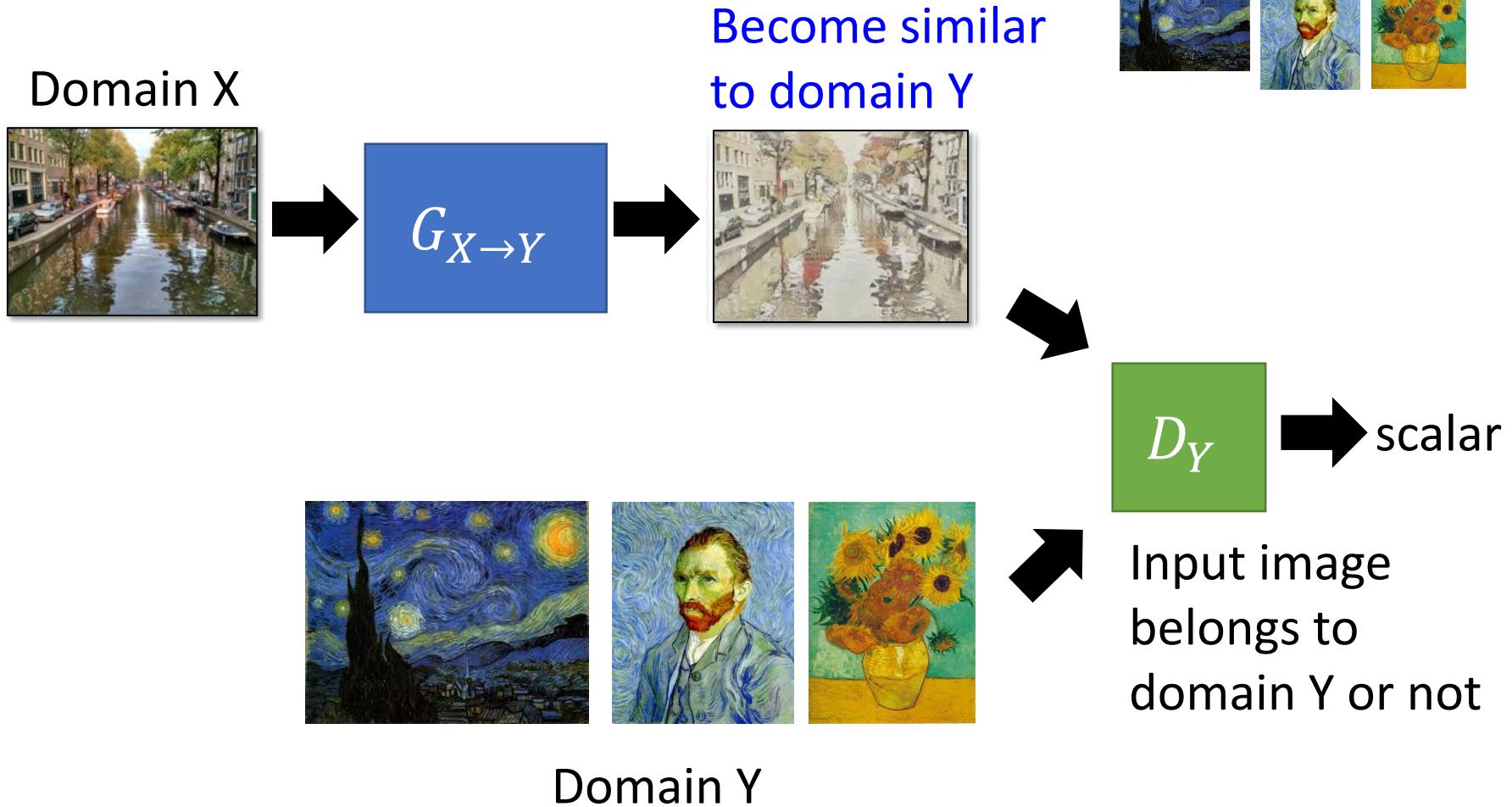


- Approach 2: Projection to Common Space

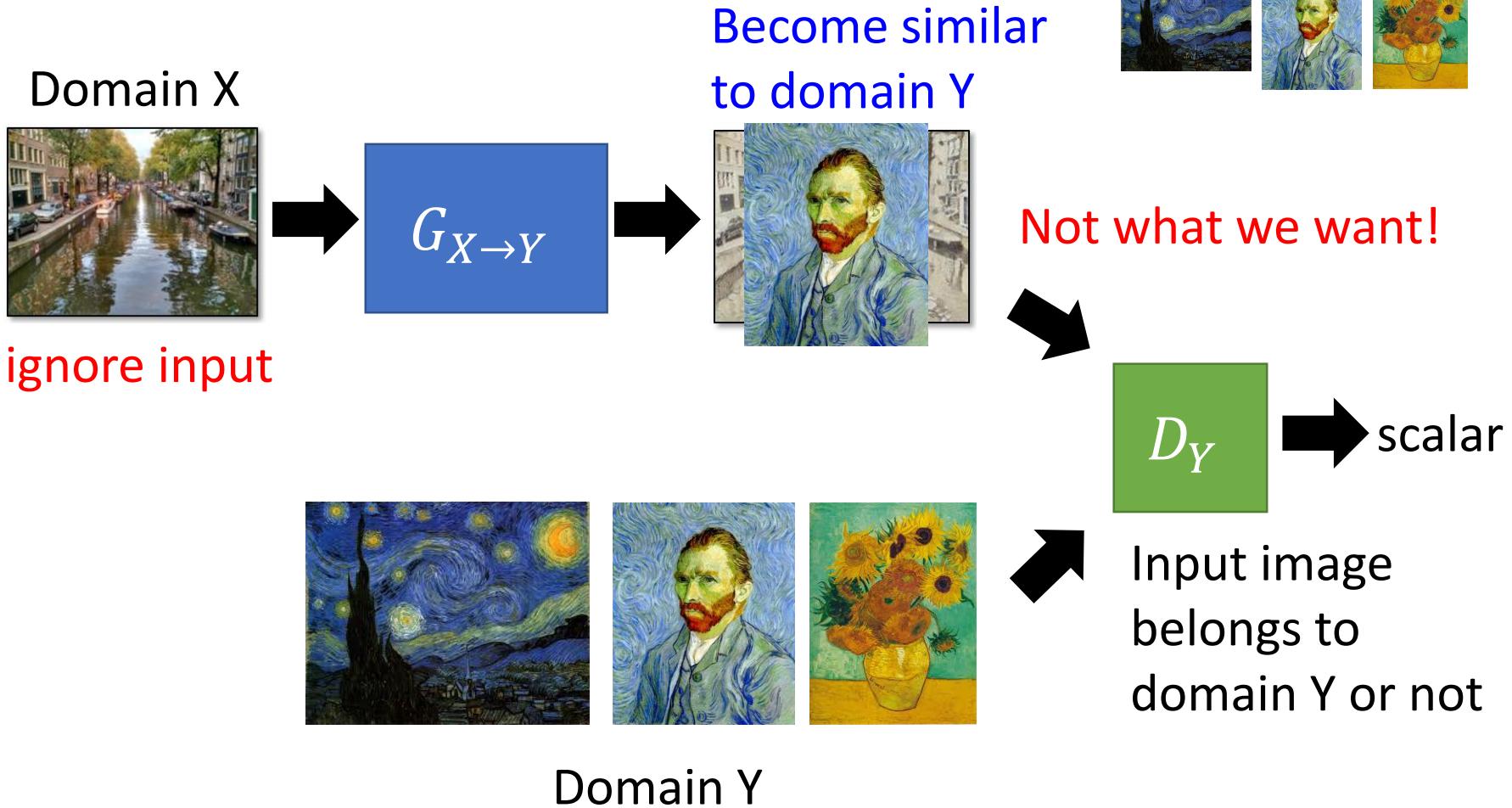


Larger change, only keep the semantics

Direct Transformation



Direct Transformation



Direct Transformation

Domain X



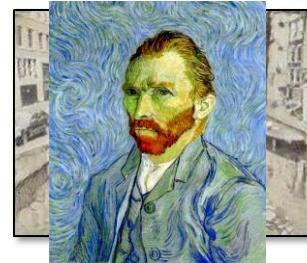
ignore input



$$G_{X \rightarrow Y}$$



Become similar
to domain Y



Not what we want!



$$D_Y$$



scalar

The issue can be avoided by network design.

Simpler generator makes the input and output more closely related.



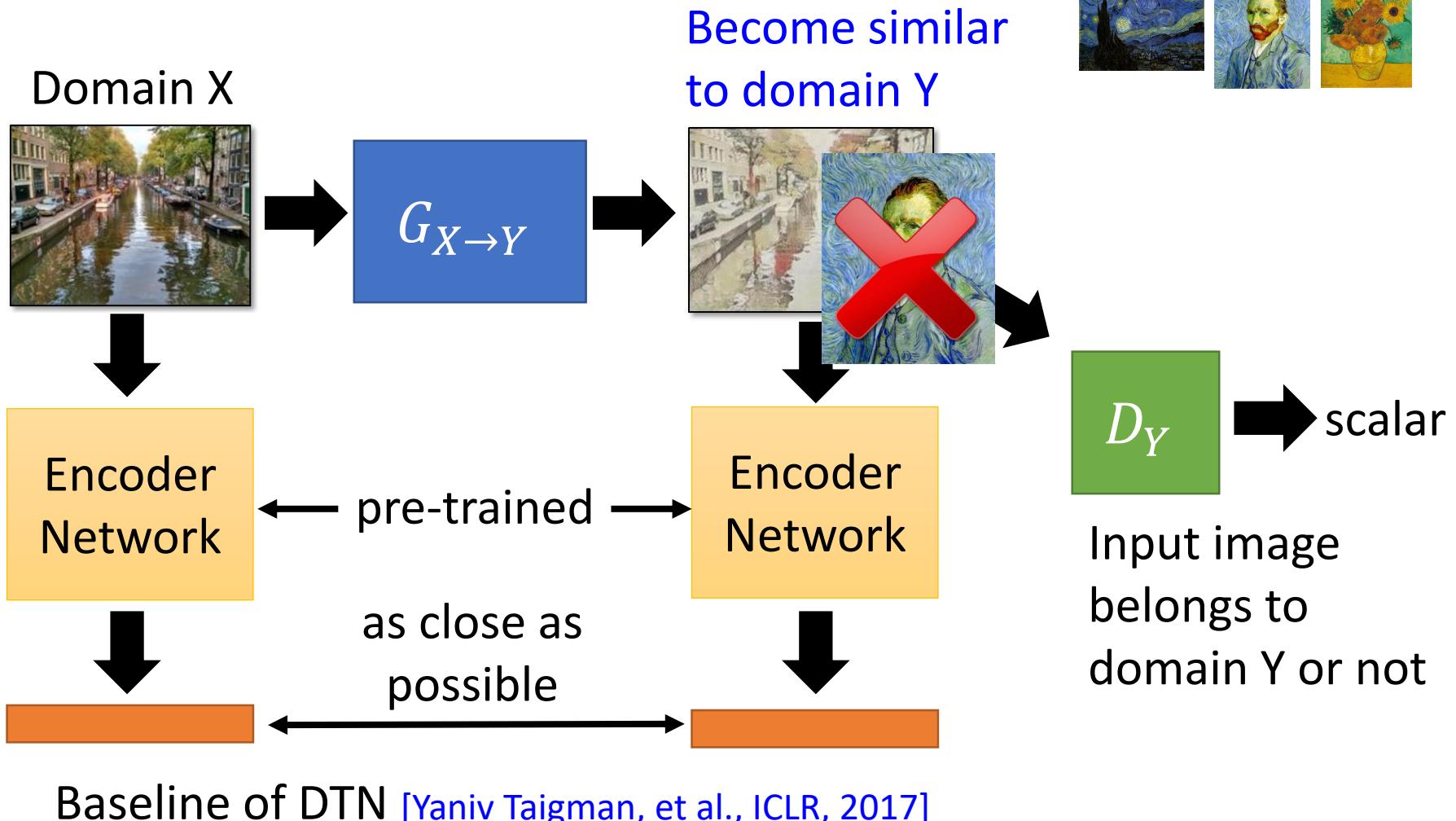
Domain Y



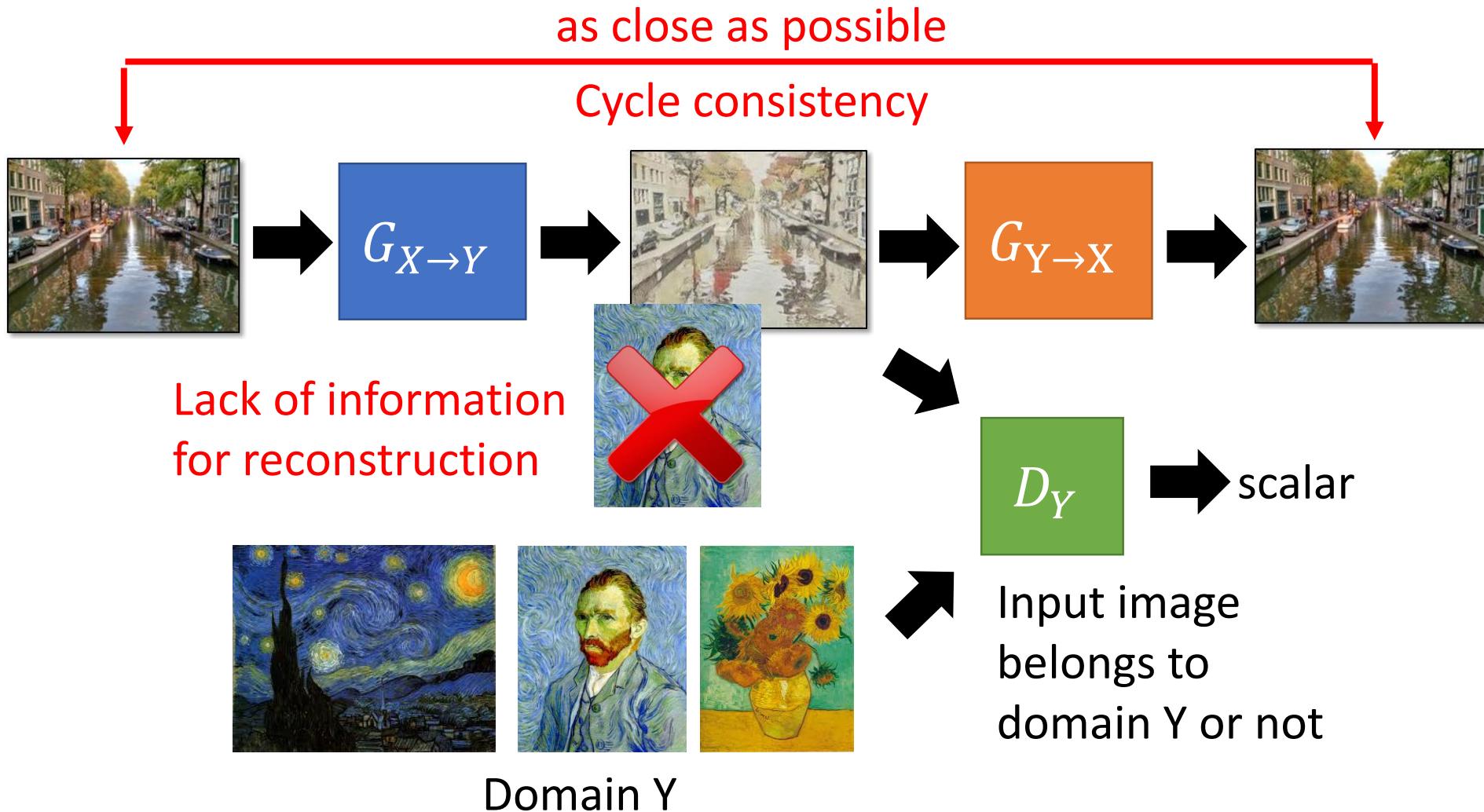
Input image
belongs to
domain Y or not

[Tomer Galanti, et al. ICLR, 2018]

Direct Transformation

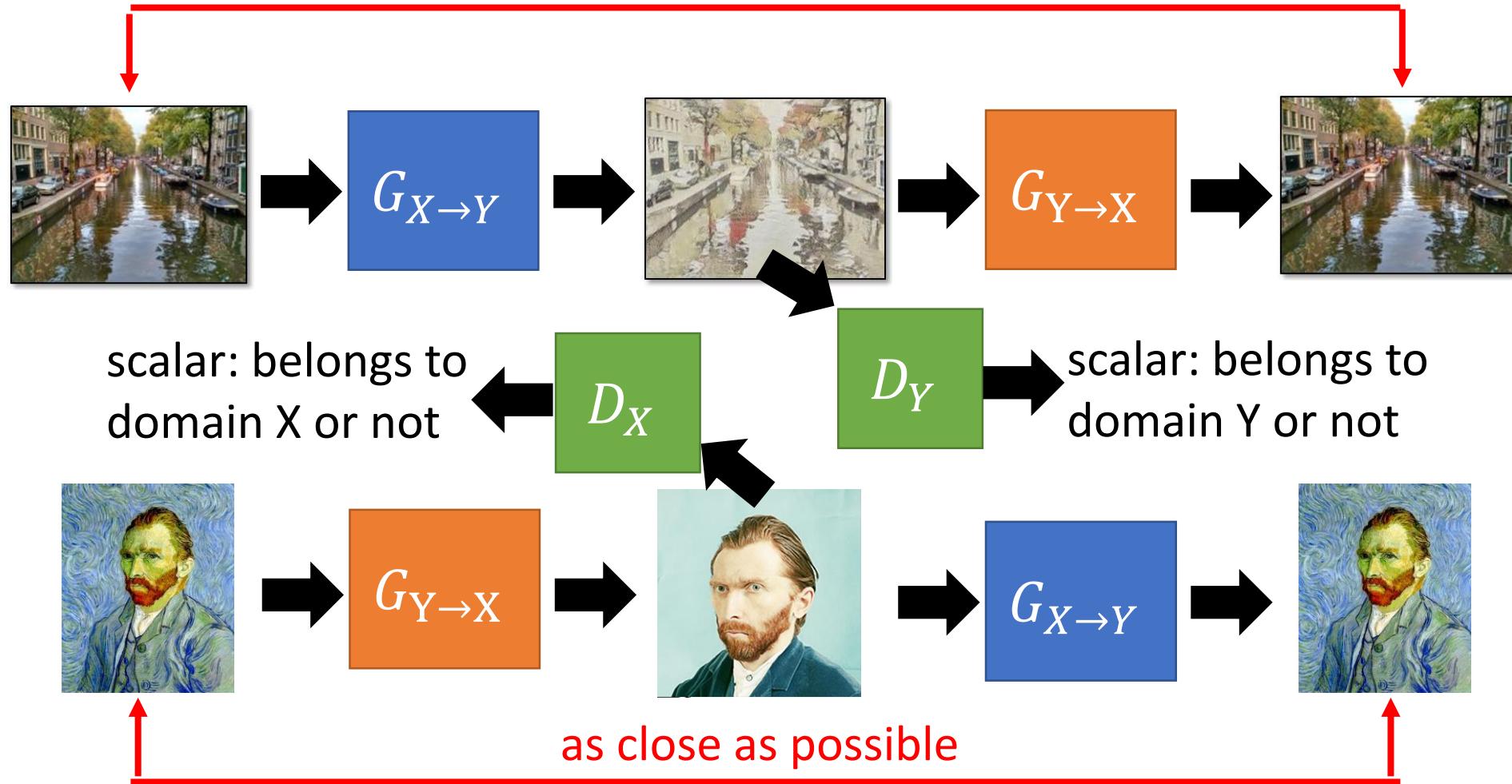


Direct Transformation



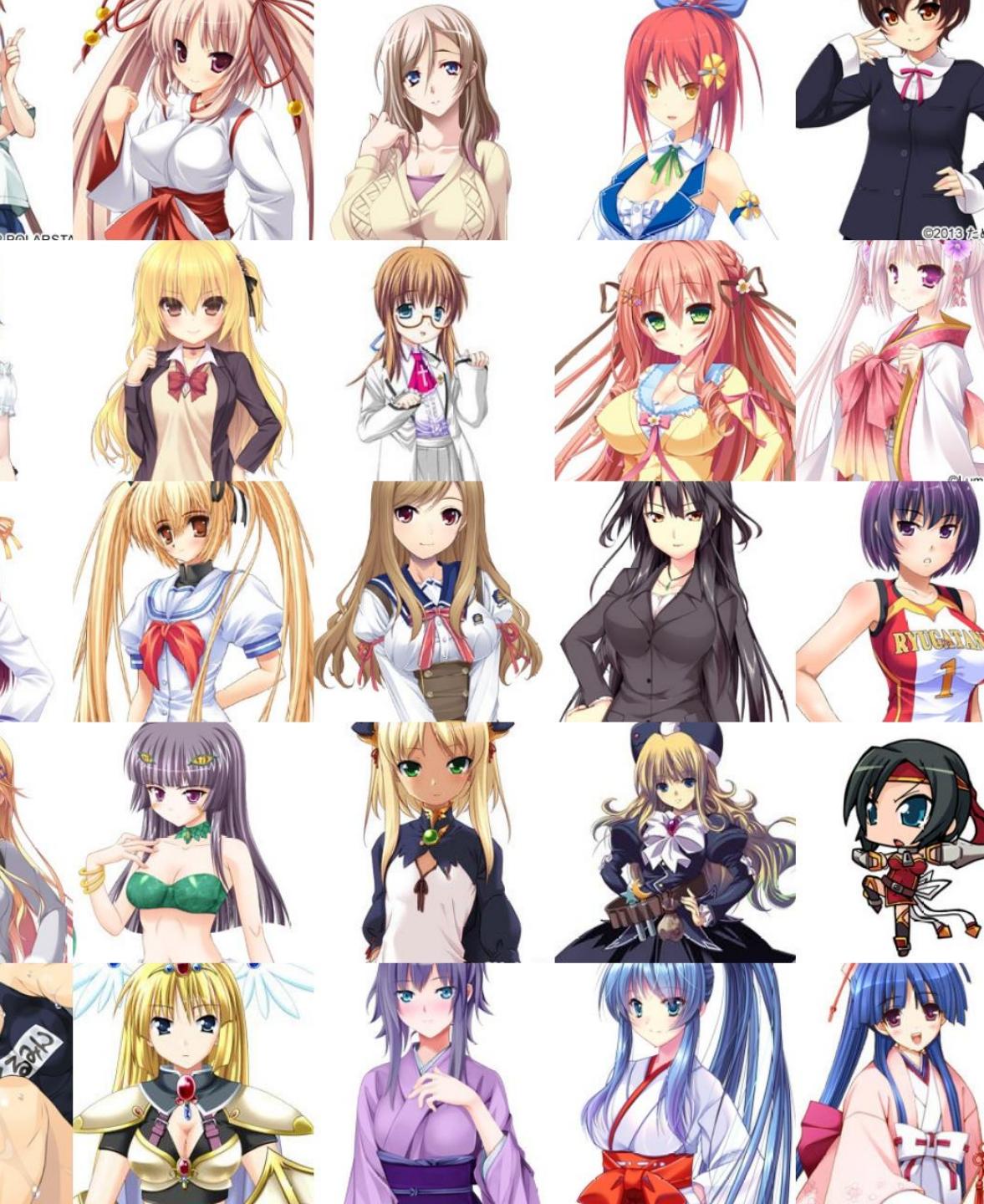
Direct Transformation

as close as possible



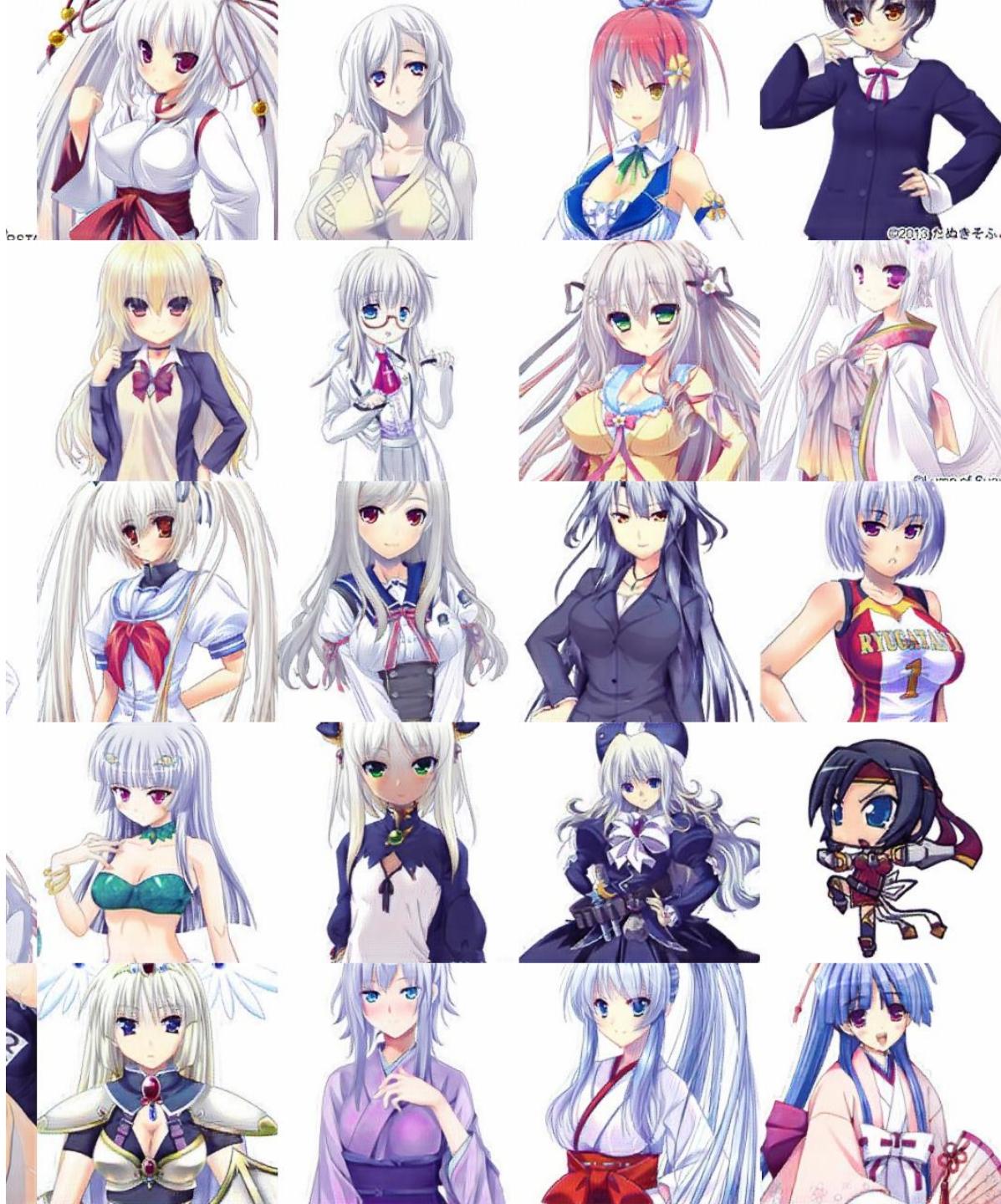
Cycle GAN – Silver Hair

- <https://github.com/Aixile/chainer-cyclegan>



Cycle GAN – Silver Hair

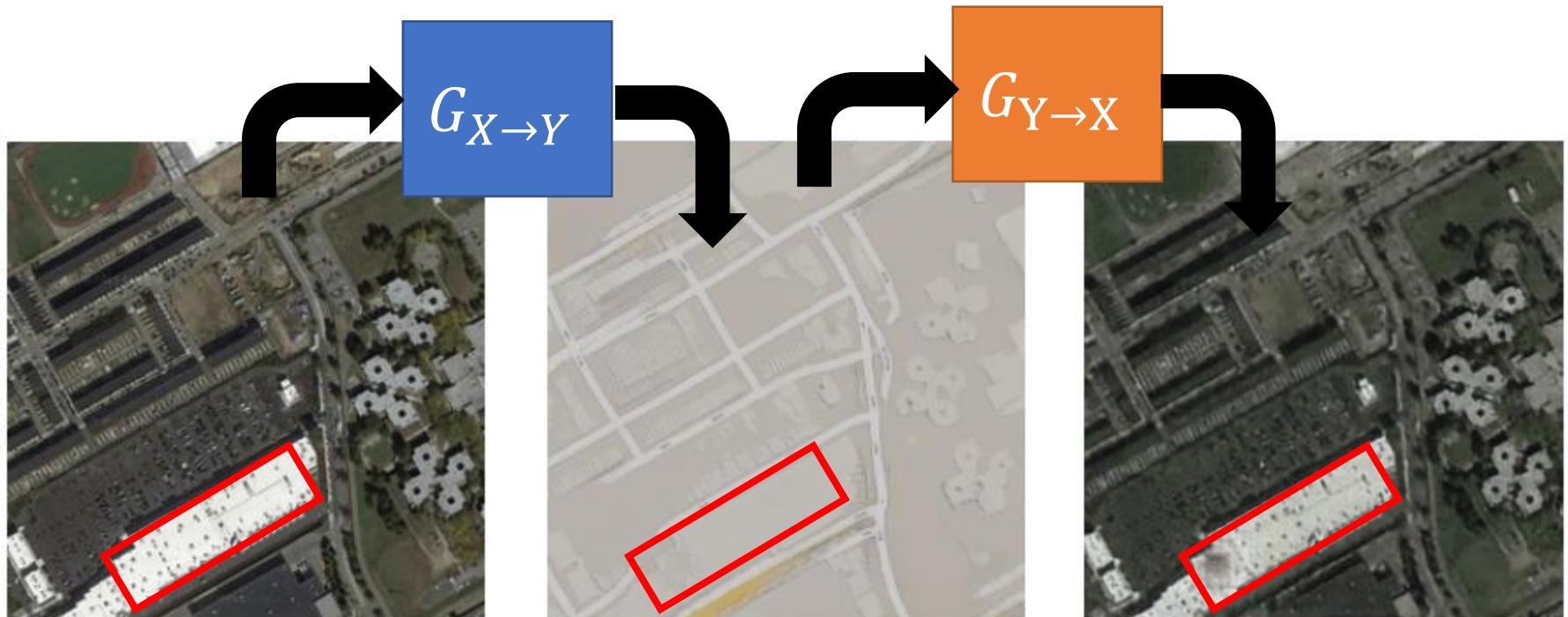
- <https://github.com/Aixile/chainer-cyclegan>



Issue of Cycle Consistency

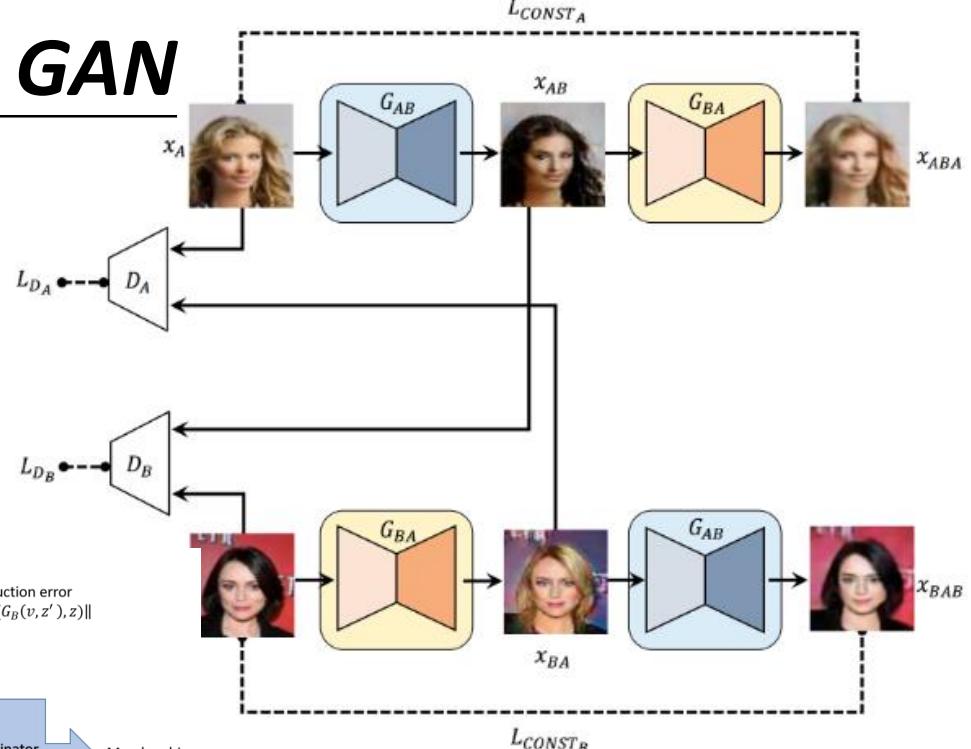
- **CycleGAN: a Master of Steganography (隱寫術)**

[Casey Chu, et al., NIPS workshop, 2017]



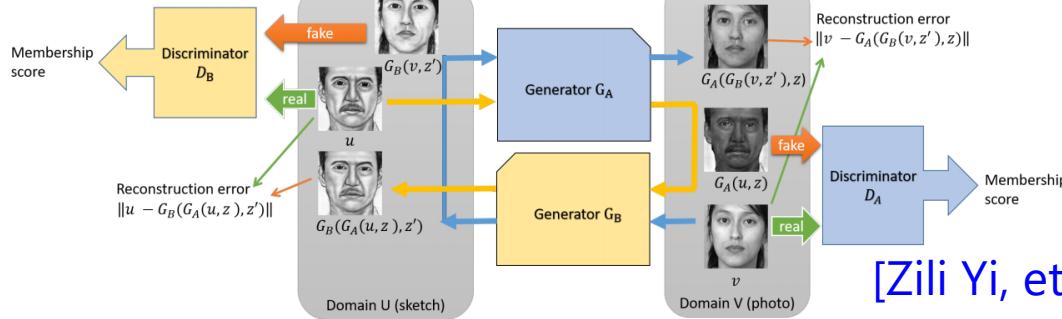
The information is hidden.

Disco GAN



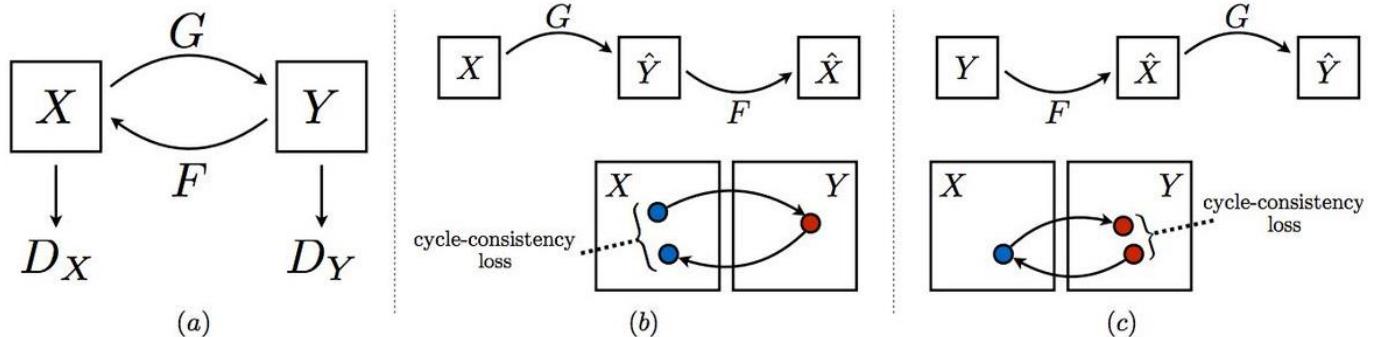
[Taeksoo Kim, et al., ICML, 2017]

Dual GAN



[Zili Yi, et al., ICCV, 2017]

Cycle GAN



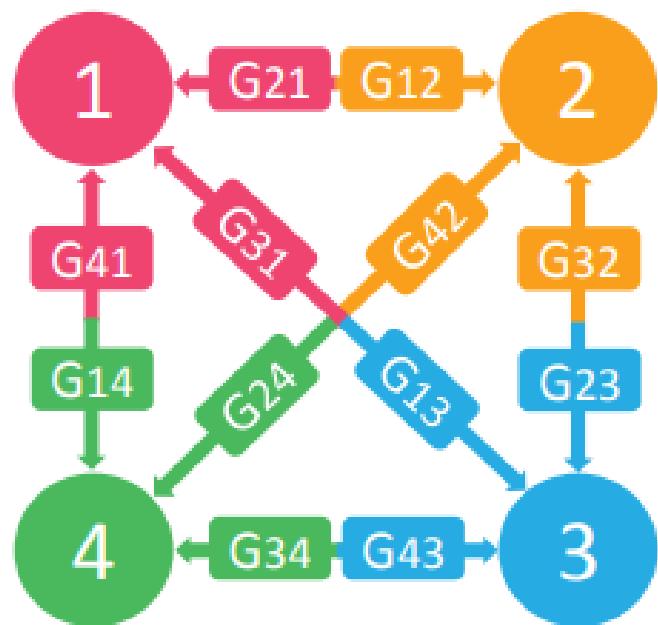
[Jun-Yan Zhu, et al., ICCV, 2017]

StarGAN

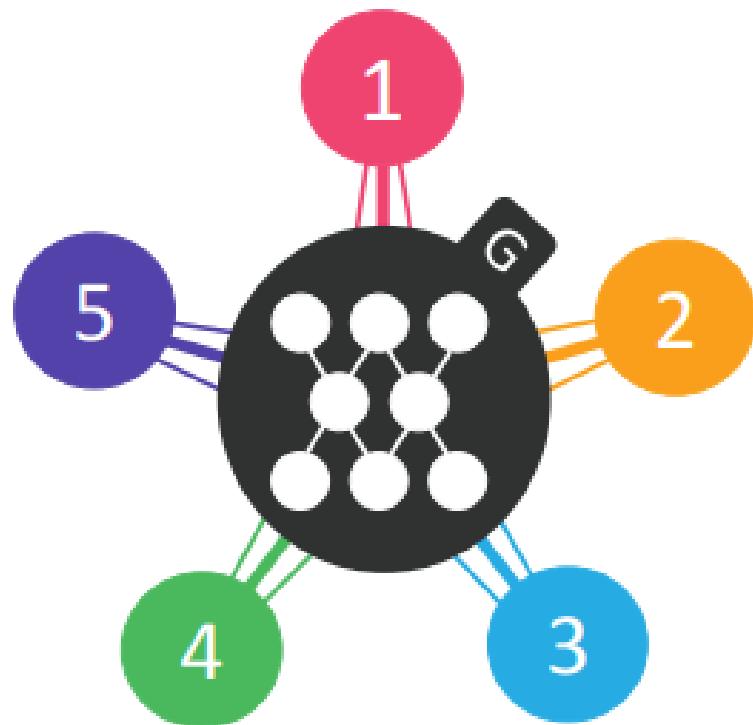
For multiple domains,
considering starGAN

[Yunjey Choi, arXiv, 2017]

(a) Cross-domain models

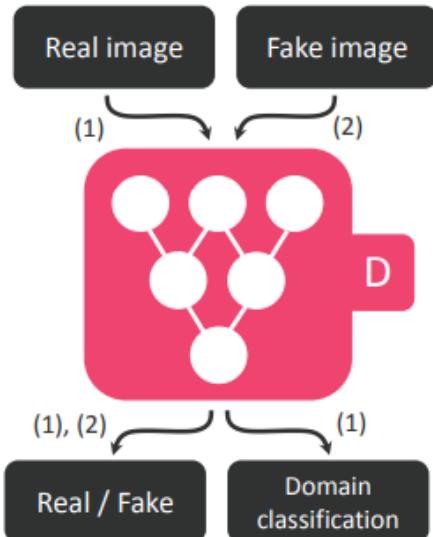


(b) StarGAN

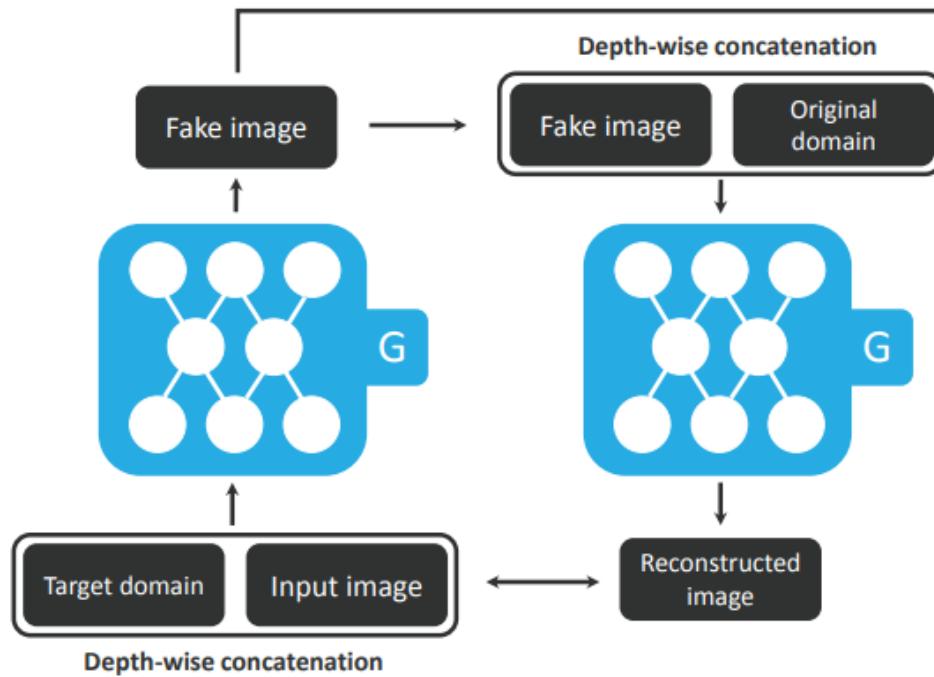


StarGAN

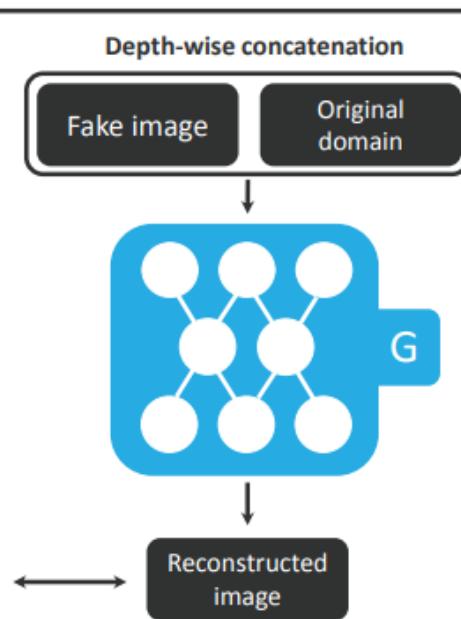
(a) Training the discriminator



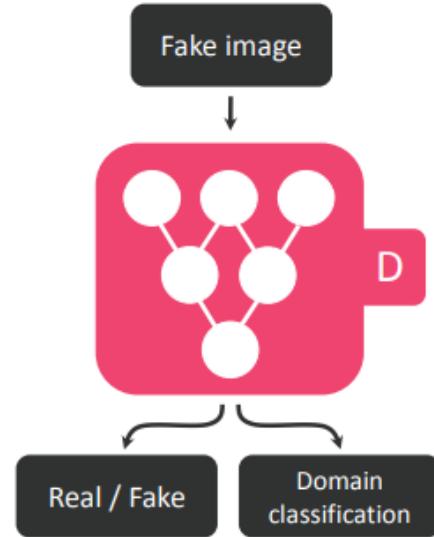
(b) Original-to-target domain



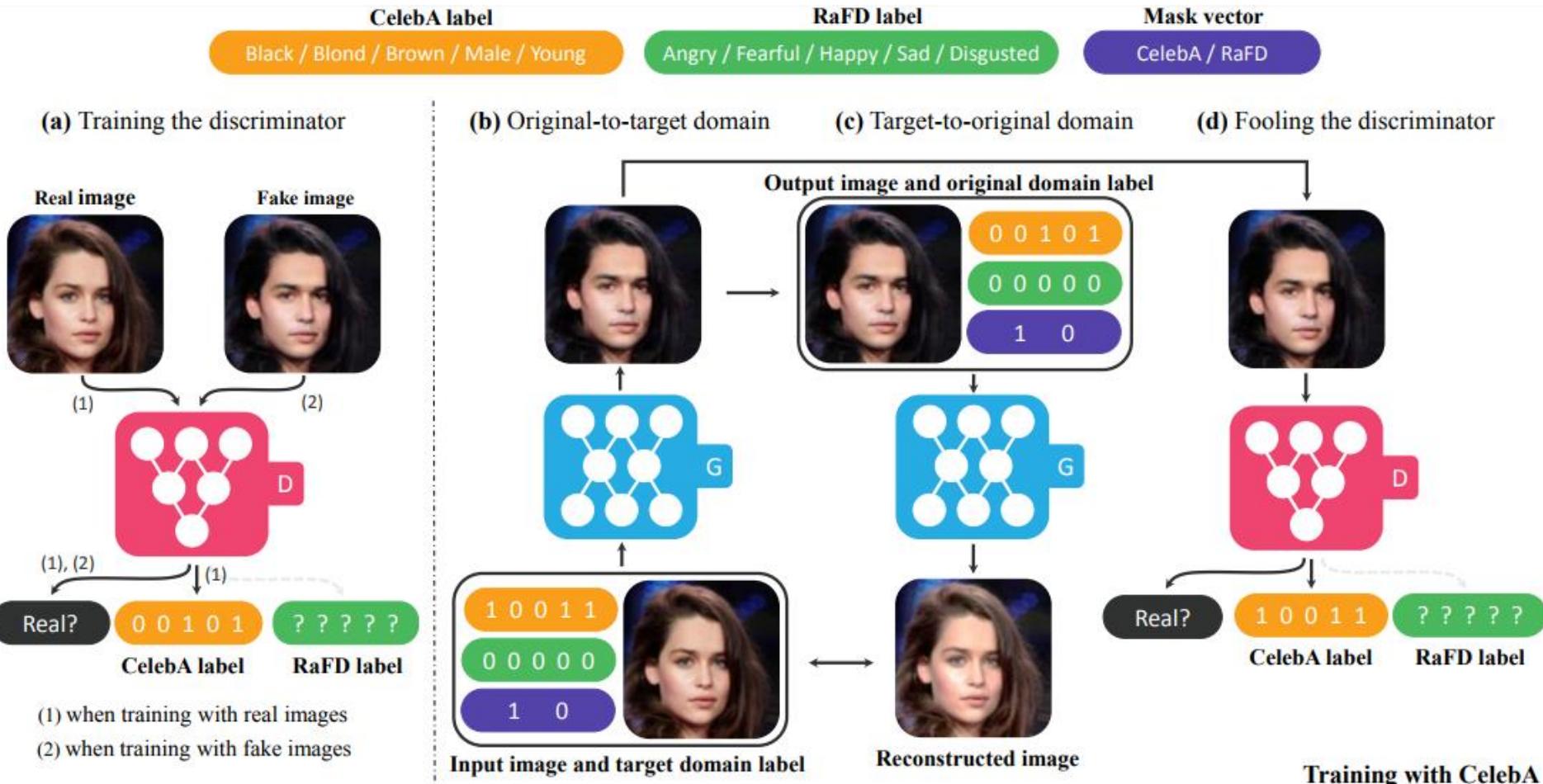
(c) Target-to-original domain



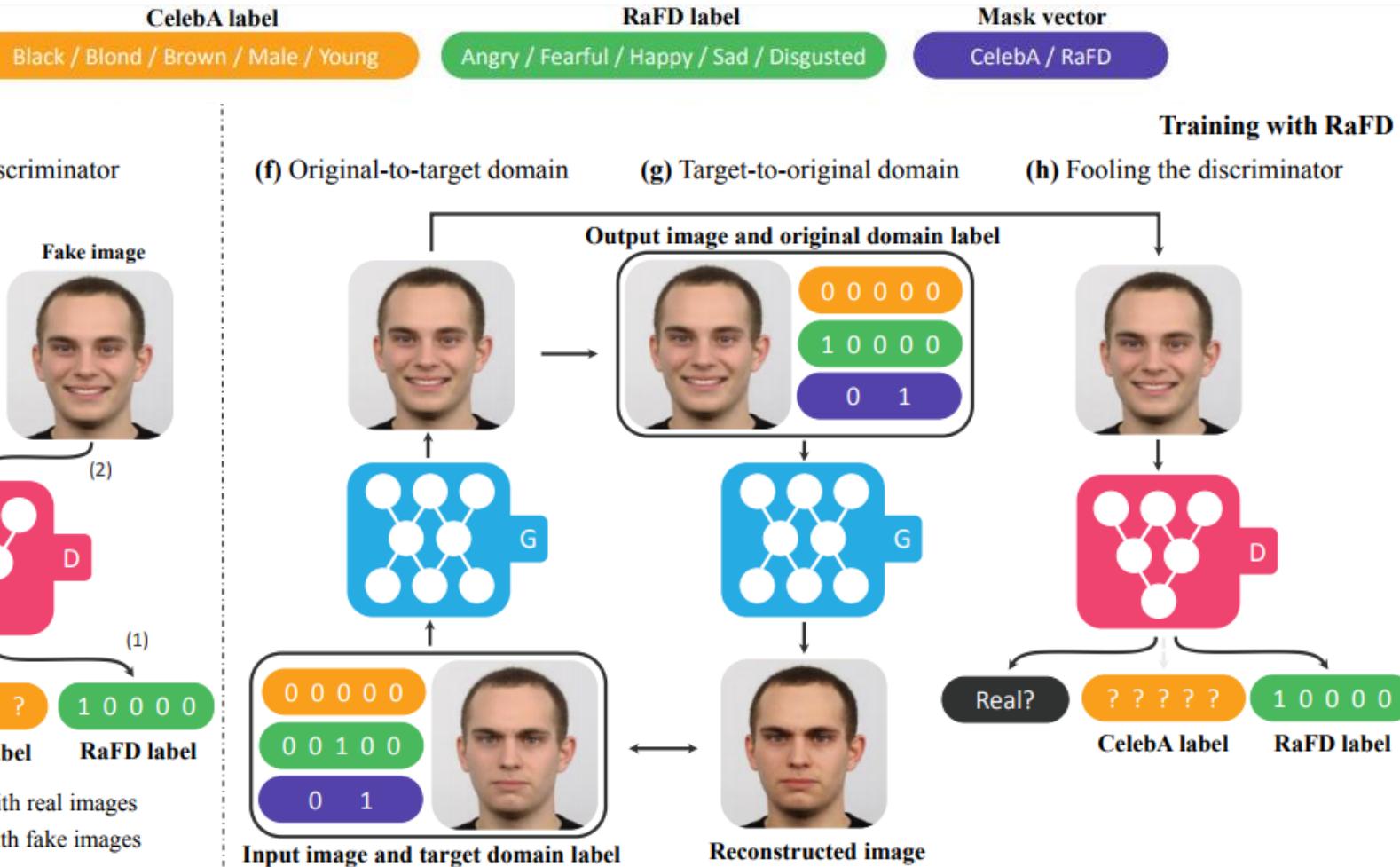
(d) Fooling the discriminator



StarGAN

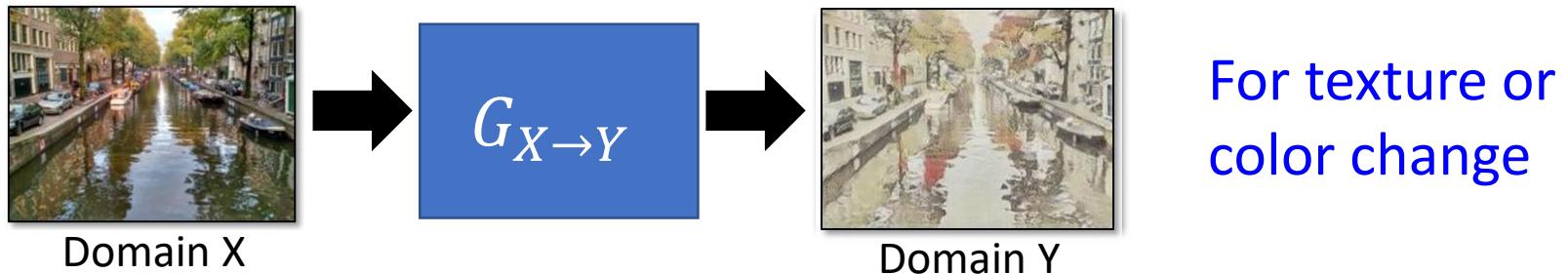


StarGAN

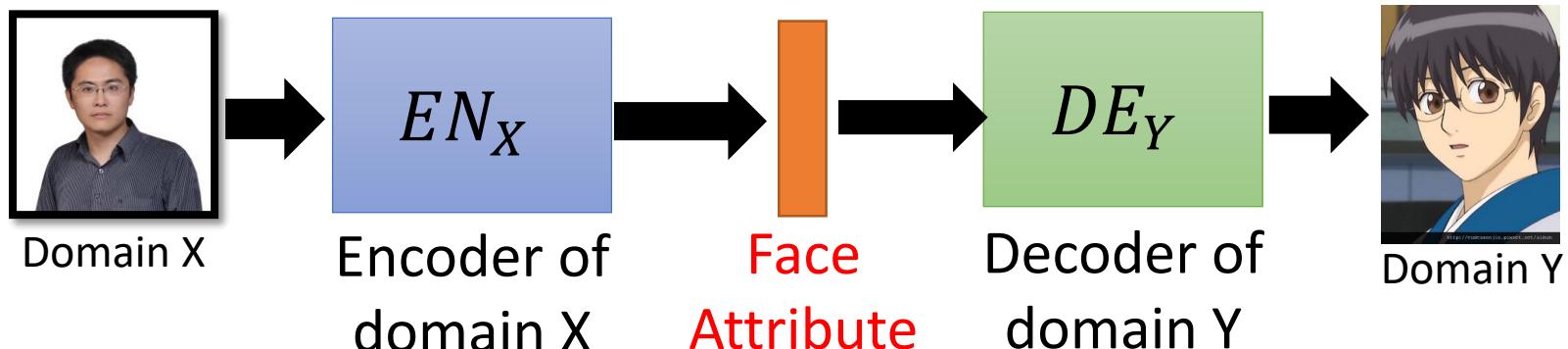


Unsupervised Conditional Generation

- Approach 1: Direct Transformation



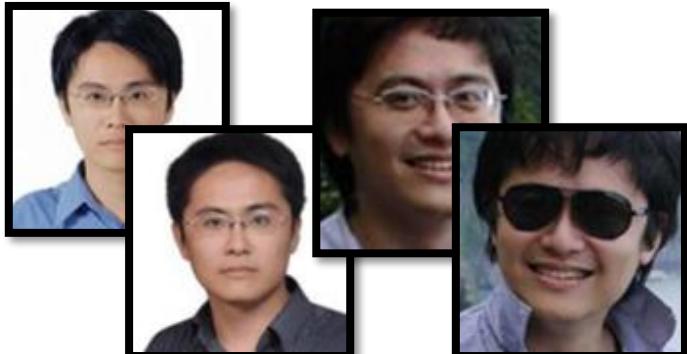
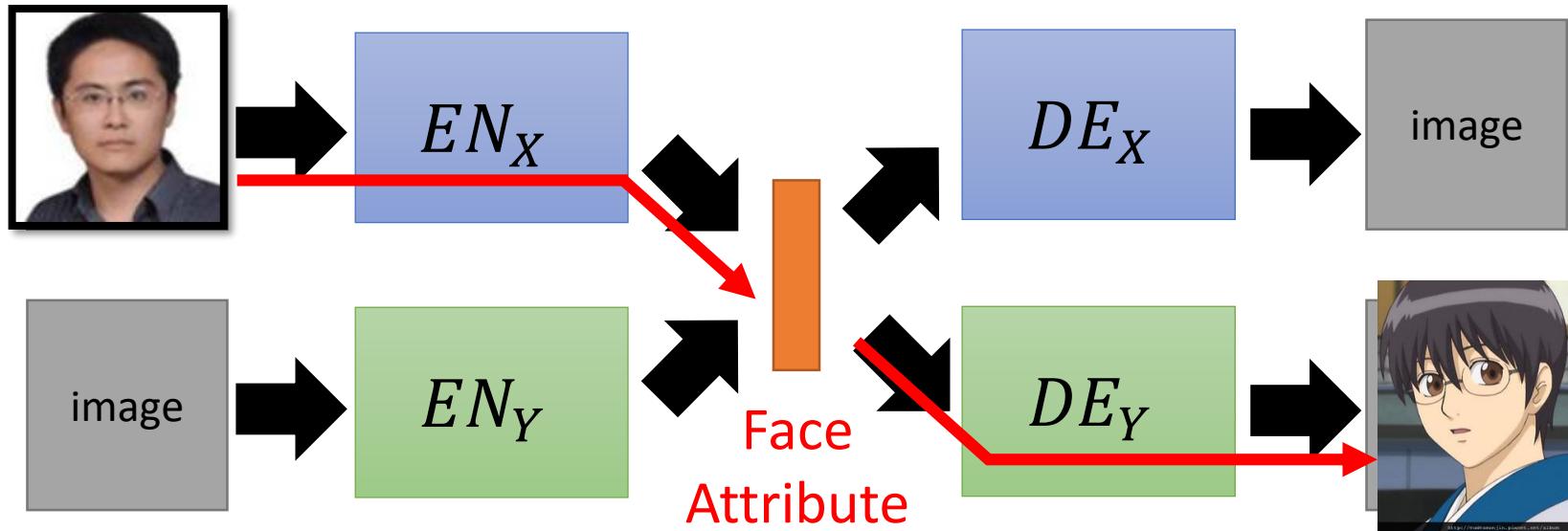
- Approach 2: Projection to Common Space



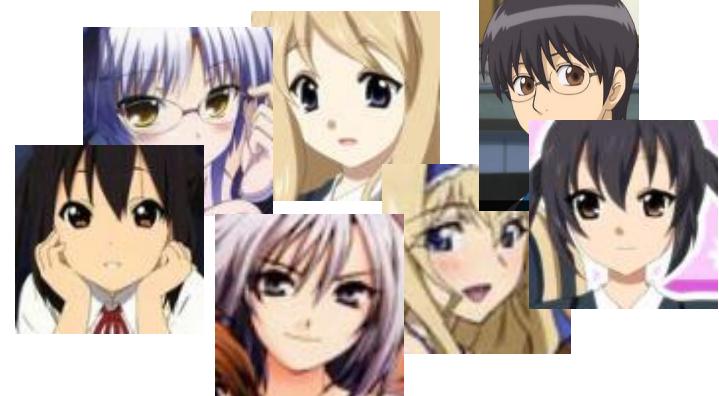
Larger change, only keep the semantics

Projection to Common Space

Target



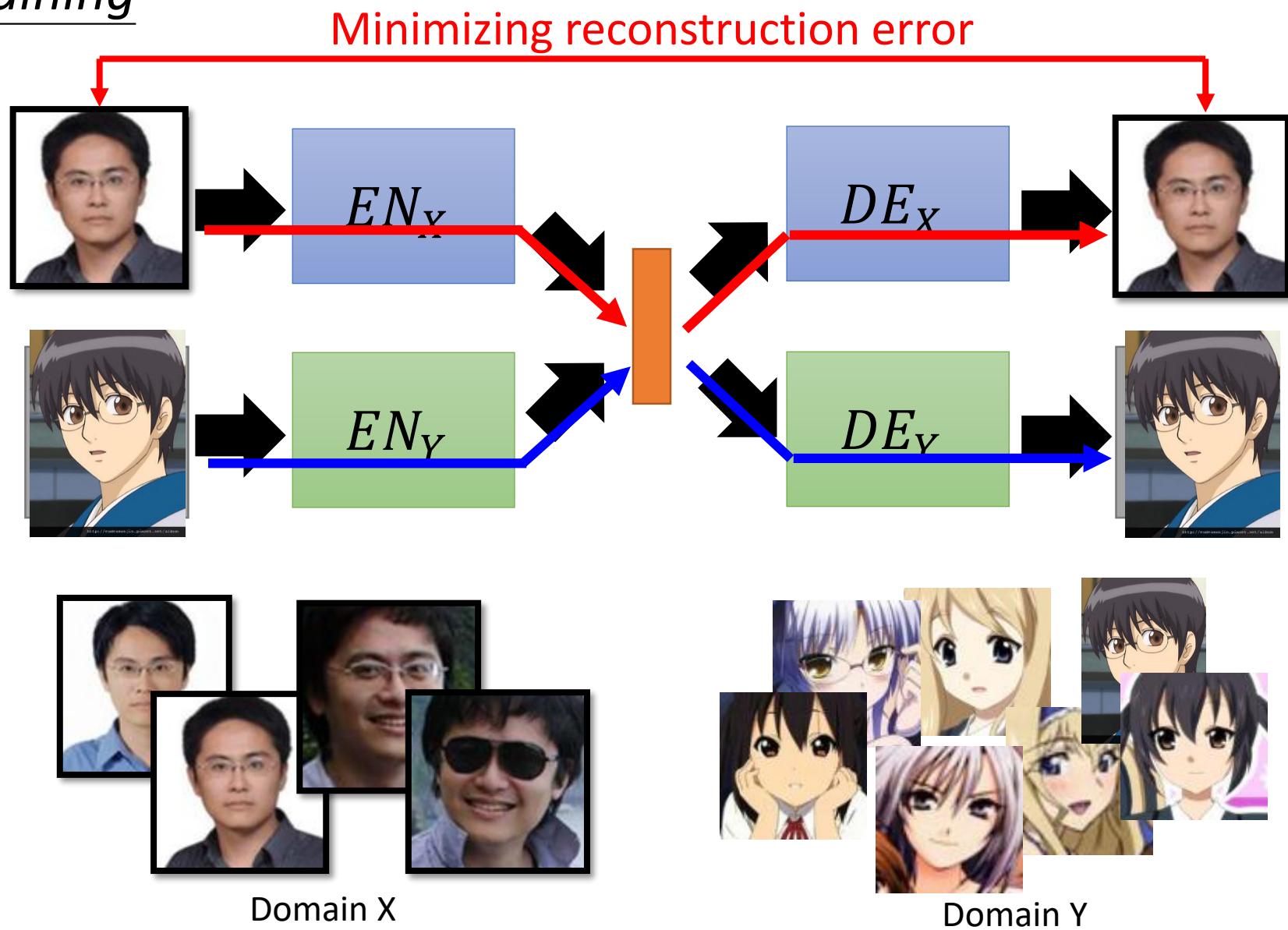
Domain X



Domain Y

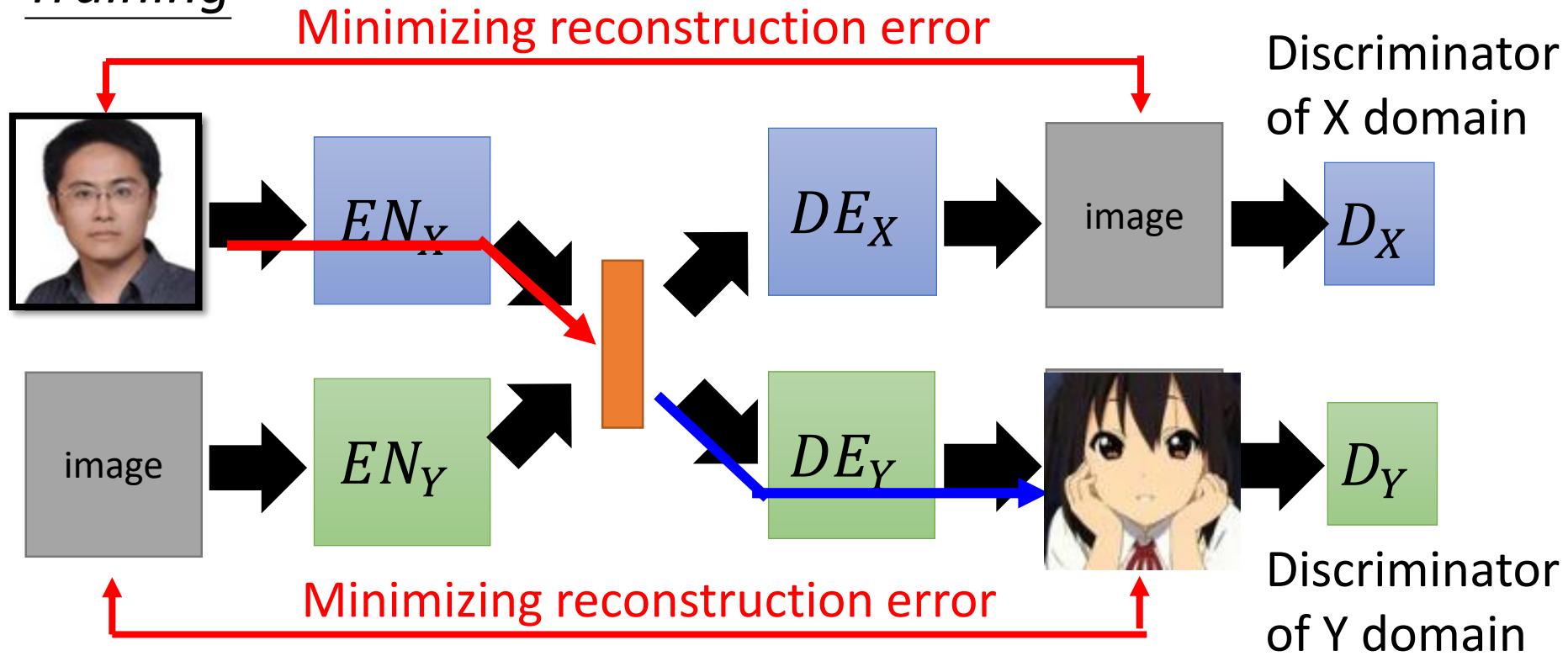
Projection to Common Space

Training



Projection to Common Space

Training

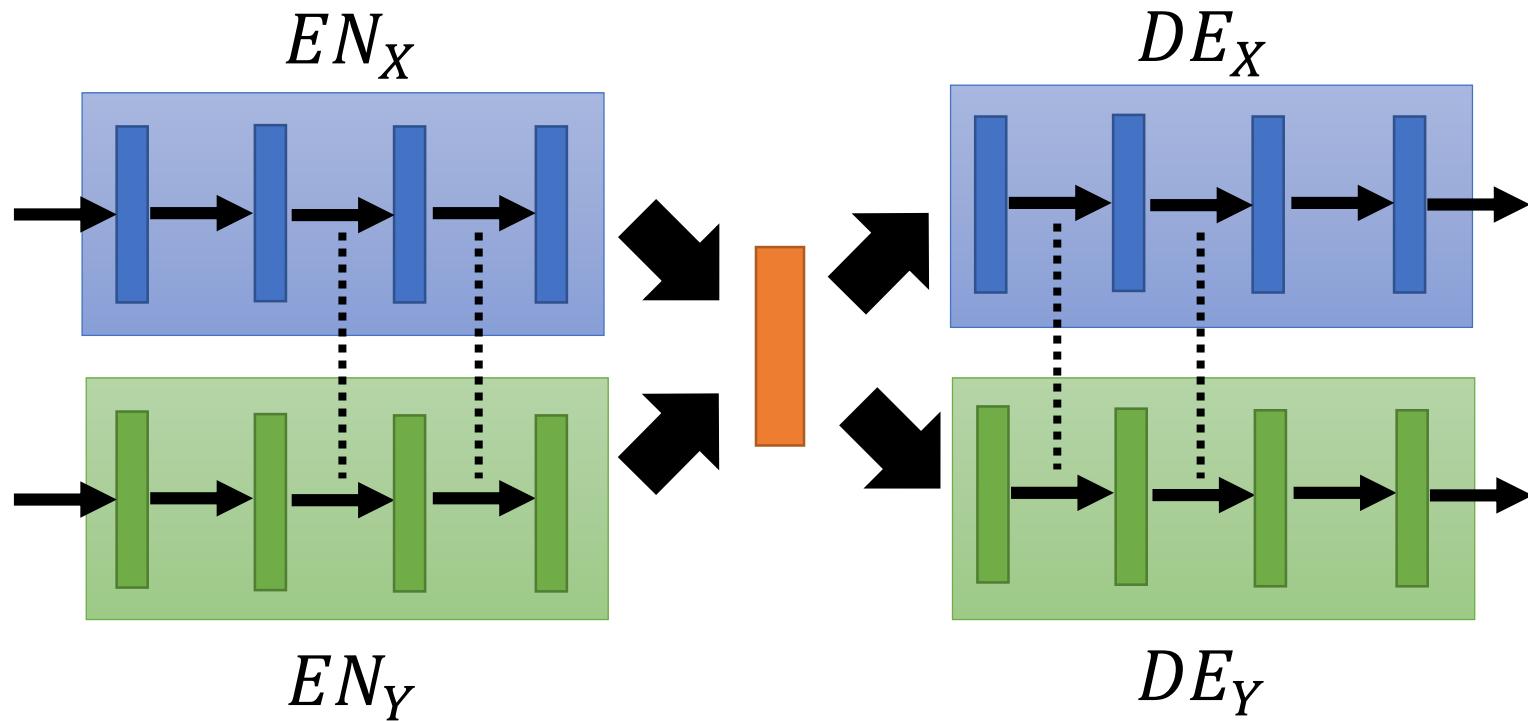


Because we train two auto-encoders separately ...

The images with the same attribute may not project to the same position in the latent space.

Projection to Common Space

Training

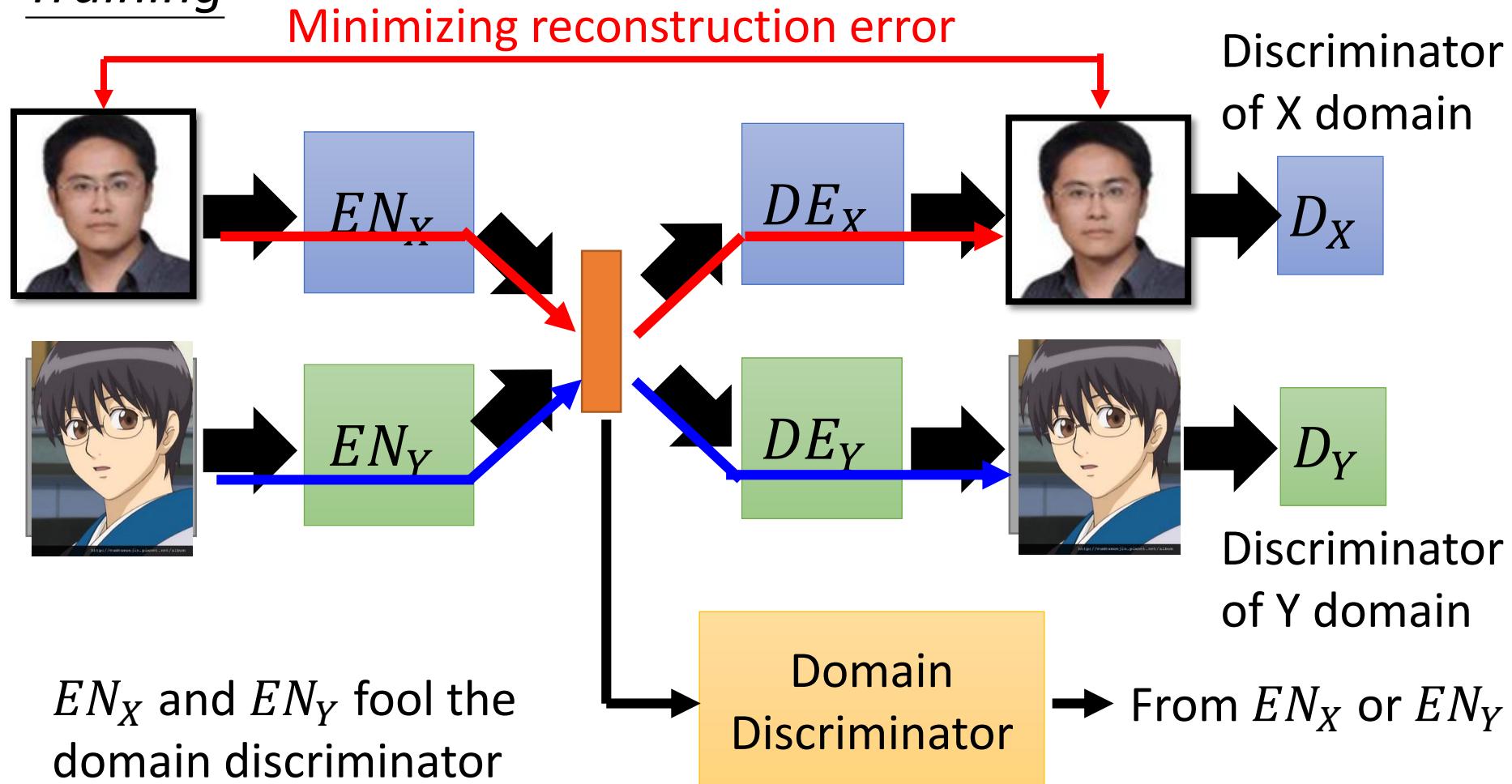


Sharing the parameters of encoders and decoders

Couple GAN [Ming-Yu Liu, et al., NIPS, 2016]
UNIT [Ming-Yu Liu, et al., NIPS, 2017]

Projection to Common Space

Training

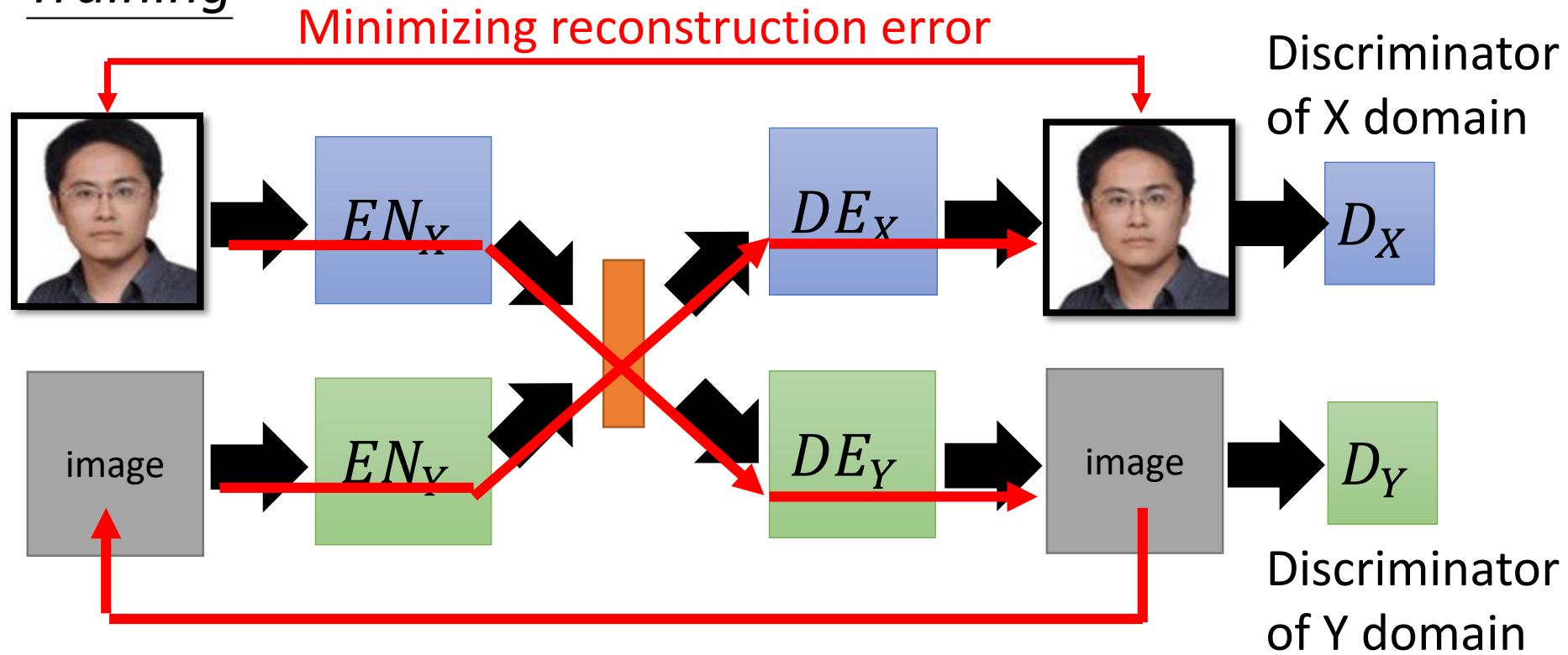


The domain discriminator forces the output of EN_X and EN_Y have the same distribution.

[Guillaume Lample, et al., NIPS, 2017]

Projection to Common Space

Training

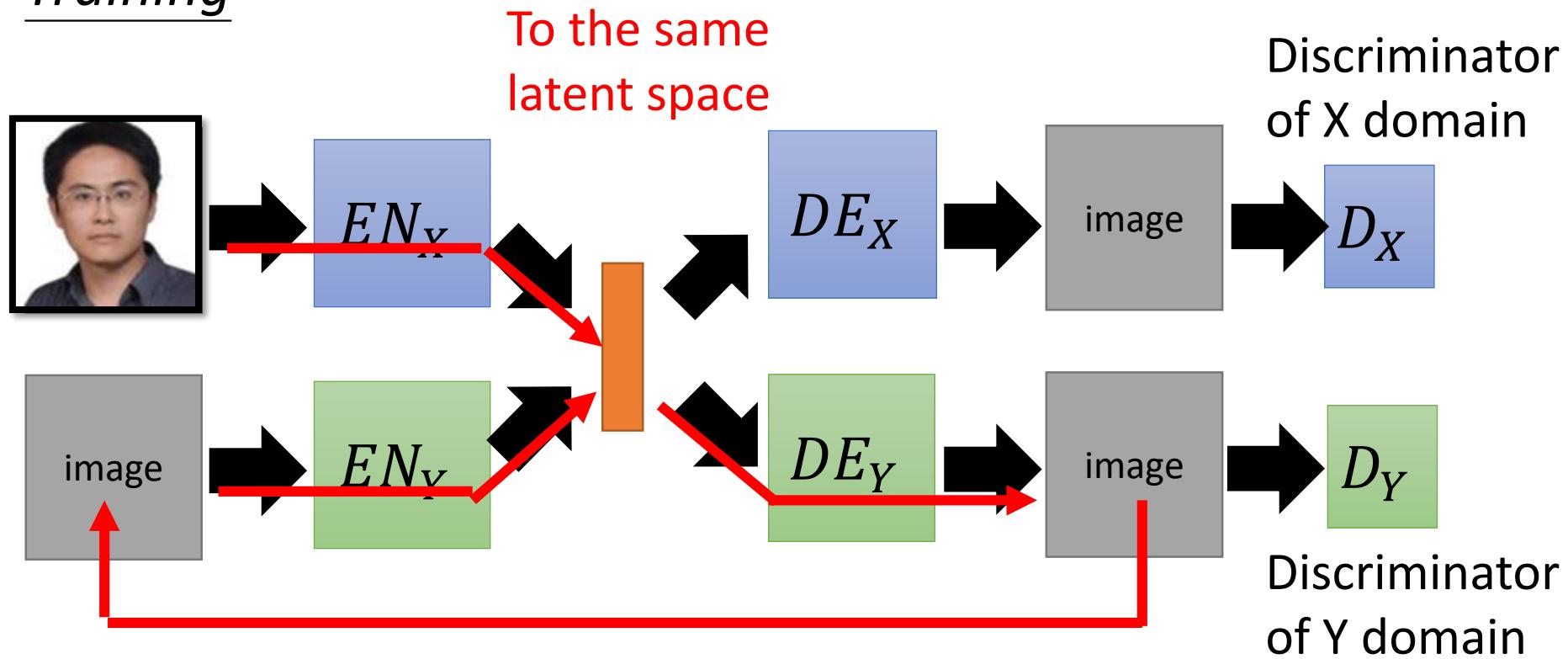


Cycle Consistency:

Used in ComboGAN [Asha Anoosheh, et al., arXiv, 017]

Projection to Common Space

Training



Semantic Consistency:

Used in DTN [Yaniv Taigman, et al., ICLR, 2017] and
XGAN [Amélie Royer, et al., arXiv, 2017]

世界二次元化

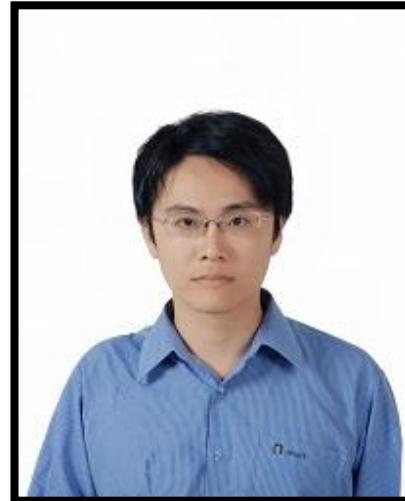
- Using the code:
https://github.com/Hiking/kawaii_creator
- It is not cycle GAN,
Disco GAN



input



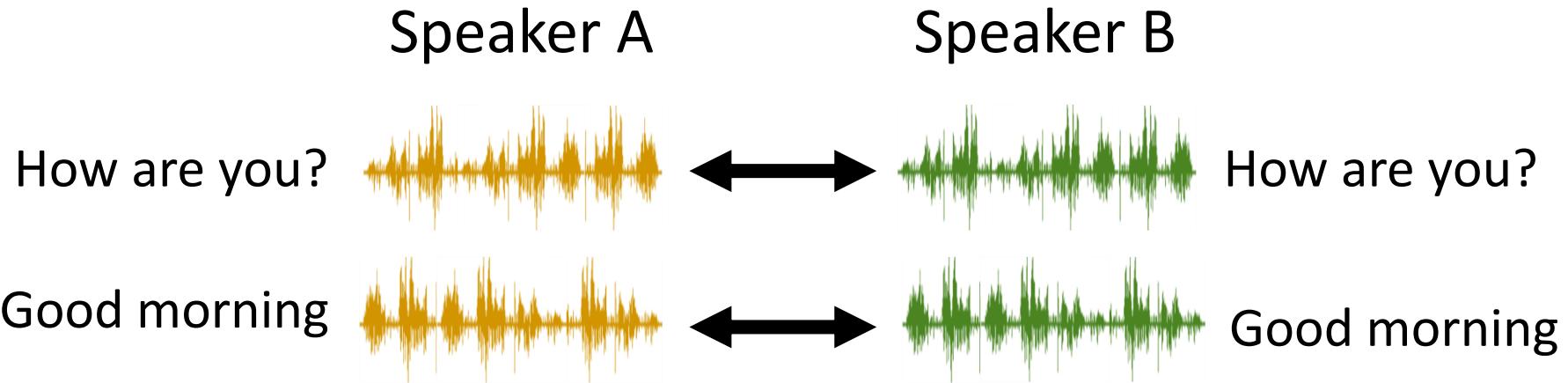
output domain



Voice Conversion



In the past



Today



Speakers A and B are talking about completely different things.

Speaker A

我



Speaker B



感謝周儒杰同學提供實驗結果

Reference

- Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros, Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV, 2017
- Zili Yi, Hao Zhang, Ping Tan, Minglun Gong, DualGAN: Unsupervised Dual Learning for Image-to-Image Translation, ICCV, 2017
- Tomer Galanti, Lior Wolf, Sagie Benaim, The Role of Minimal Complexity Functions in Unsupervised Learning of Semantic Mappings, ICLR, 2018
- Yaniv Taigman, Adam Polyak, Lior Wolf, Unsupervised Cross-Domain Image Generation, ICLR, 2017
- Asha Anoosheh, Eirikur Agustsson, Radu Timofte, Luc Van Gool, ComboGAN: Unrestrained Scalability for Image Domain Translation, arXiv, 2017
- Amélie Royer, Konstantinos Bousmalis, Stephan Gouws, Fred Bertsch, Inbar Mosseri, Forrester Cole, Kevin Murphy, XGAN: Unsupervised Image-to-Image Translation for Many-to-Many Mappings, arXiv, 2017

Reference

- Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic Denoyer, Marc'Aurelio Ranzato, Fader Networks: Manipulating Images by Sliding Attributes, NIPS, 2017
- Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, Jiwon Kim, Learning to Discover Cross-Domain Relations with Generative Adversarial Networks, ICML, 2017
- Ming-Yu Liu, Oncel Tuzel, “Coupled Generative Adversarial Networks”, NIPS, 2016
- Ming-Yu Liu, Thomas Breuel, Jan Kautz, Unsupervised Image-to-Image Translation Networks, NIPS, 2017
- Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, Jaegul Choo, StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation, arXiv, 2017

Theory behind GAN

Generation

Using Generative
Adversarial
Network (GAN)

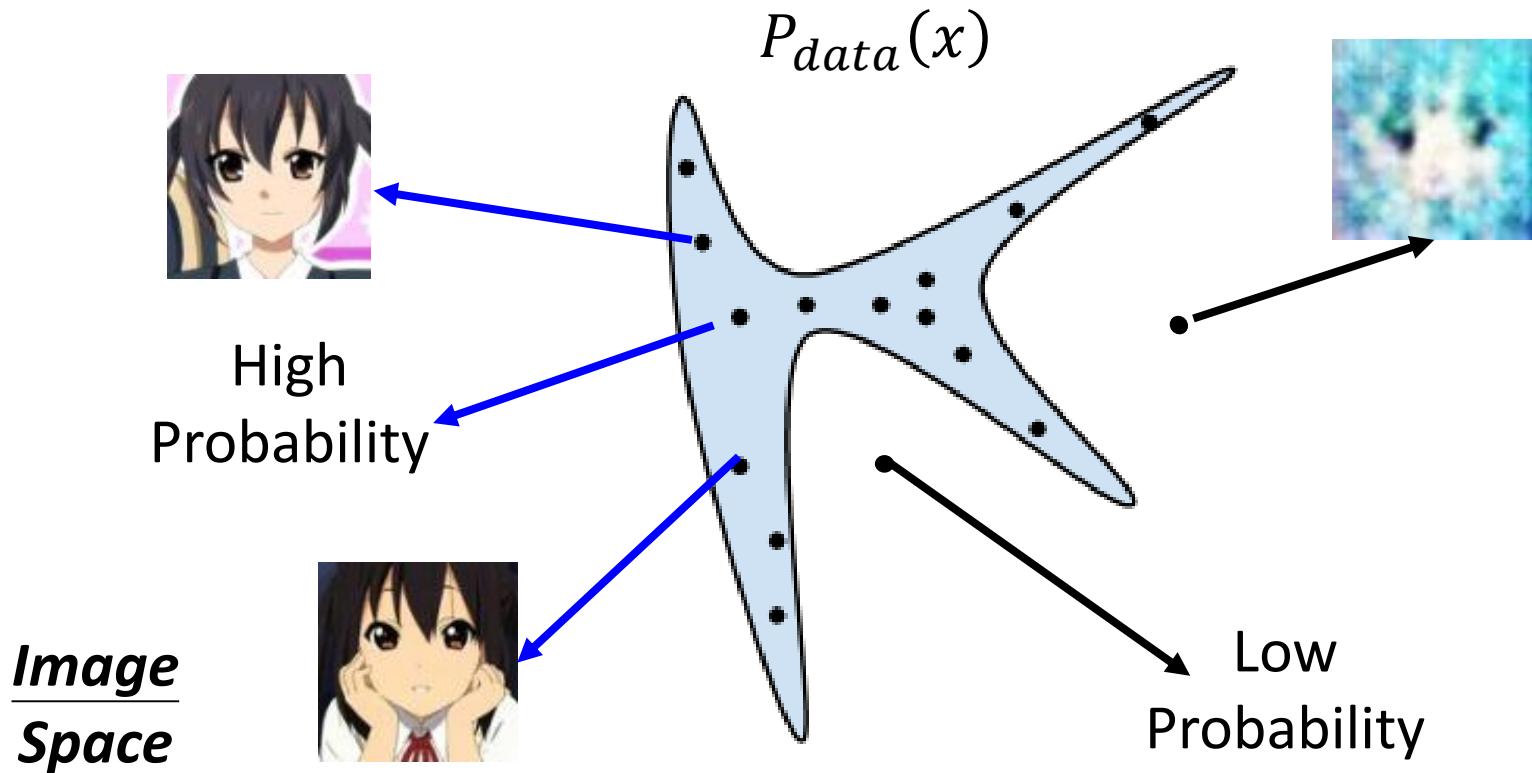


Drawing?

Generation

x : an image (a high-dimensional vector)

- We want to find data distribution $P_{data}(x)$



Maximum Likelihood Estimation

- Given a data distribution $P_{data}(x)$ (We can sample from it.)
- We have a distribution $P_G(x; \theta)$ parameterized by θ
 - We want to find θ such that $P_G(x; \theta)$ close to $P_{data}(x)$
 - E.g. $P_G(x; \theta)$ is a Gaussian Mixture Model, θ are means and variances of the Gaussians

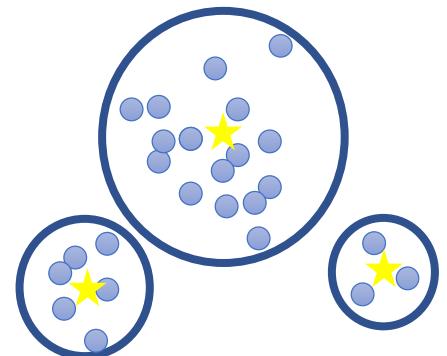
Sample $\{x^1, x^2, \dots, x^m\}$ from $P_{data}(x)$

We can compute $P_G(x^i; \theta)$

Likelihood of generating the samples

$$L = \prod_{i=1}^m P_G(x^i; \theta)$$

Find θ^* maximizing the likelihood



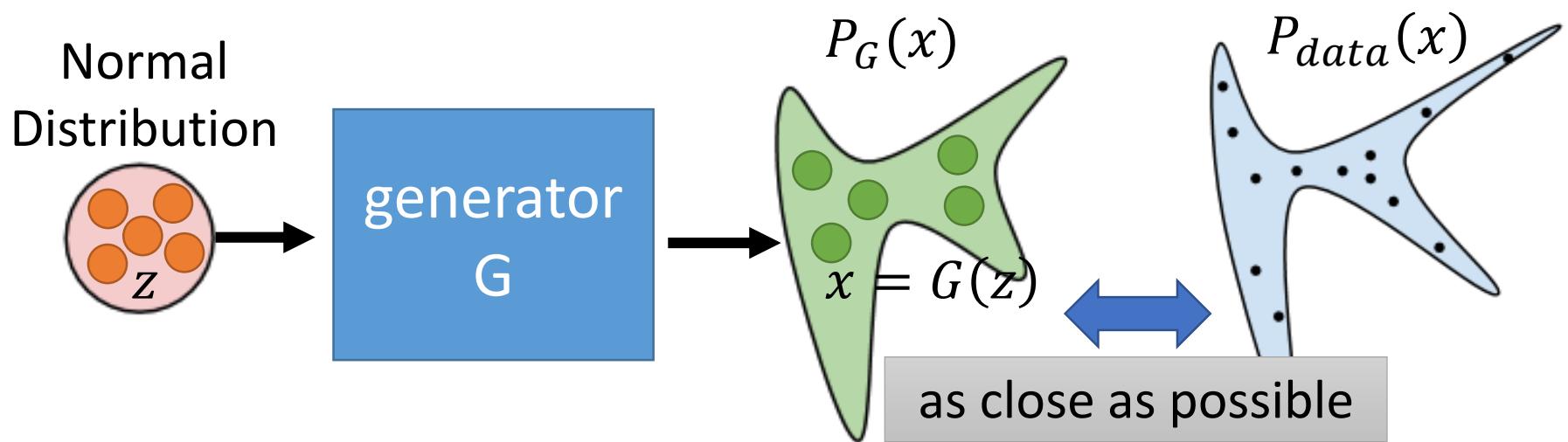
Maximum Likelihood Estimation = Minimize KL Divergence

$$\begin{aligned}\theta^* &= \arg \max_{\theta} \prod_{i=1}^m P_G(x^i; \theta) = \arg \max_{\theta} \log \prod_{i=1}^m P_G(x^i; \theta) \\ &= \arg \max_{\theta} \sum_{i=1}^m \log P_G(x^i; \theta) \quad \{x^1, x^2, \dots, x^m\} \text{ from } P_{data}(x) \\ &\approx \arg \max_{\theta} E_{x \sim P_{data}} [\log P_G(x; \theta)] \\ &= \arg \max_{\theta} \int_x P_{data}(x) \log P_G(x; \theta) dx - \int_x P_{data}(x) \log P_{data}(x) dx \\ &= \arg \min_{\theta} KL(P_{data} || P_G) \quad \text{How to define a general } P_G?\end{aligned}$$

Generator

x : an image (a high-dimensional vector)

- A generator G is a network. The network defines a probability distribution P_G



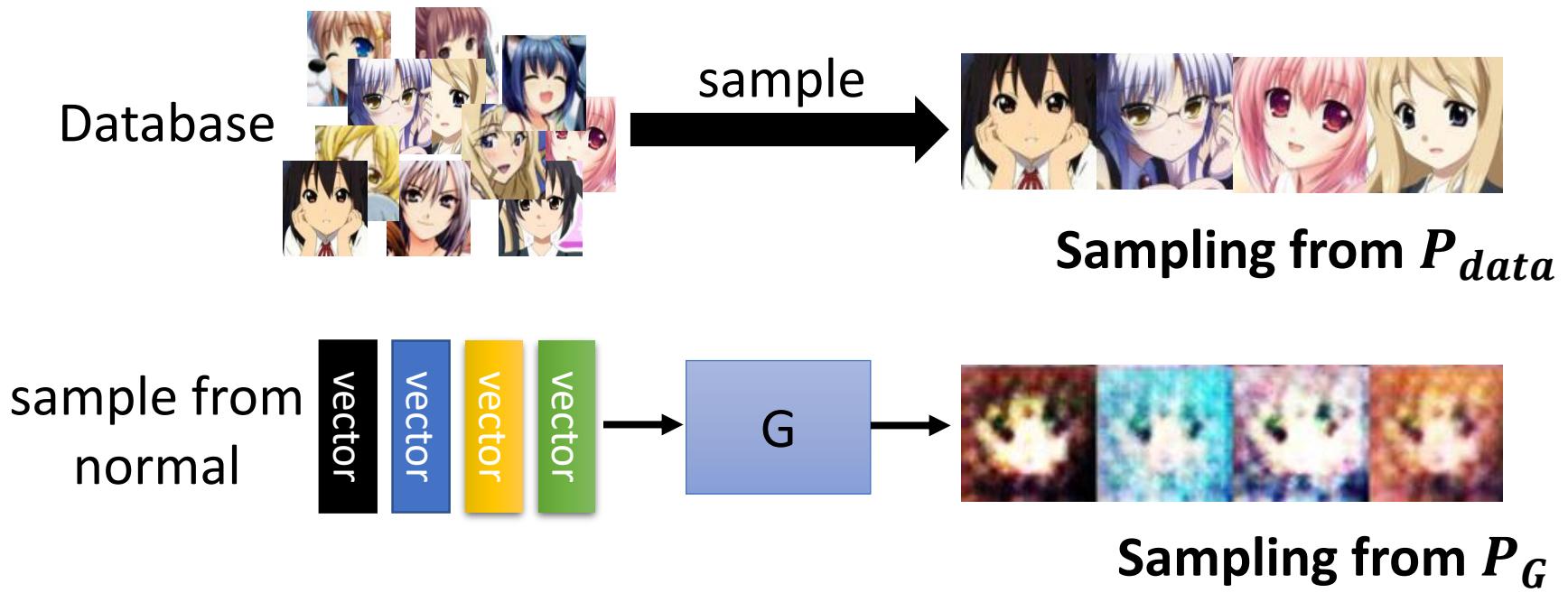
$$G^* = \arg \min_G \underline{Div(P_G, P_{data})}$$

Divergence between distributions P_G and P_{data}
How to compute the divergence?

Discriminator

$$G^* = \arg \min_G \text{Div}(P_G, P_{data})$$

Although we do not know the distributions of P_G and P_{data} , we can sample from them.



Discriminator

$$G^* = \arg \min_G \text{Div}(P_G, P_{data})$$

- ★ : data sampled from P_{data}
- ☆ : data sampled from P_G



Using the example objective function is exactly the same as training a binary classifier.



Example Objective Function for D

$$V(G, D) = E_{x \sim P_{data}} [\log D(x)] + E_{x \sim P_G} [\log(1 - D(x))]$$

↑
(G is fixed)

Training: $D^* = \arg \max_D V(D, G)$

The maximum objective value is related to JS divergence.

Discriminator

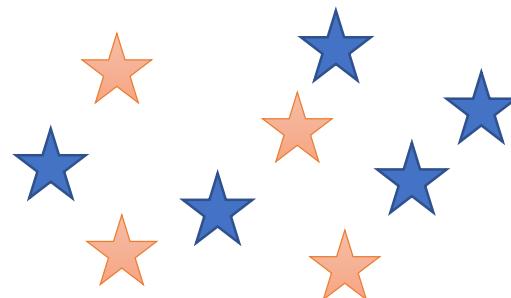
$$G^* = \arg \min_G \text{Div}(P_G, P_{data})$$

★ : data sampled from P_{data}

☆ : data sampled from P_G

Training:

$$D^* = \arg \max_D V(D, G)$$

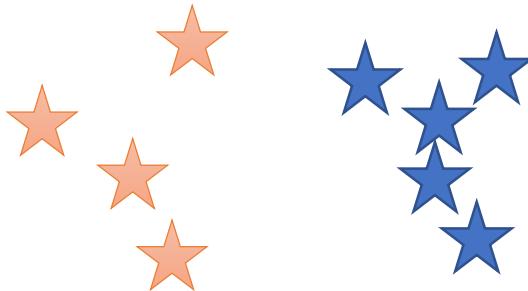


small divergence

train

Discriminator

hard to discriminate
(cannot make objective large)



large divergence

train

Discriminator

easy to discriminate

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

$$V = E_{x \sim P_{data}}[\log D(x)] + E_{x \sim P_G}[\log(1 - D(x))]$$

- Given G, what is the optimal D* maximizing

$$\begin{aligned} V &= E_{x \sim P_{data}}[\log D(x)] + E_{x \sim P_G}[\log(1 - D(x))] \\ &= \int_x P_{data}(x) \log D(x) dx + \int_x P_G(x) \log(1 - D(x)) dx \\ &= \int_x [P_{data}(x) \log D(x) + P_G(x) \log(1 - D(x))] dx \end{aligned}$$

Assume that D(x) can be any function

- Given x, the optimal D* maximizing

$$P_{data}(x) \log D(x) + P_G(x) \log(1 - D(x))$$

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

$$V = E_{x \sim P_{data}}[\log D(x)] + E_{x \sim P_G}[\log(1 - D(x))]$$

- Given x , the optimal D^* maximizing

$$P_{data}(x) \log D(x) + P_G(x) \log(1 - D(x))$$

a **D** **b** **D**

- Find D^* maximizing: $f(D) = a \log(D) + b \log(1 - D)$

$$\frac{df(D)}{dD} = a \times \frac{1}{D} + b \times \frac{1}{1-D} \times (-1) = 0$$

$$a \times \frac{1}{D^*} = b \times \frac{1}{1 - D^*} \quad a \times (1 - D^*) = b \times D^*$$

$$a - aD^* = bD^* \quad a = (a + b)D^*$$

$$D^* = \frac{a}{a + b} \quad \rightarrow \quad 0 <$$

$$D^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_G(x)} < 1$$

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

$$V = E_{x \sim P_{data}}[\log D(x)] + E_{x \sim P_G}[\log(1 - D(x))]$$

$$\max_D V(G, D) = V(G, D^*)$$

$$D^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_G(x)}$$

$$= E_{x \sim P_{data}} \left[\log \frac{P_{data}(x)}{P_{data}(x) + P_G(x)} \right]$$

$$+ E_{x \sim P_G} \left[\log \frac{P_G(x)}{P_{data}(x) + P_G(x)} \right]$$

$$= \int_x P_{data}(x) \log \frac{\frac{1}{2} P_{data}(x)}{\underline{\frac{P_{data}(x) + P_G(x)}{2}}} dx$$

$$+ 2 \log \frac{1}{2} - 2 \log 2$$

$$+ \int_x P_G(x) \log \frac{\frac{1}{2} P_G(x)}{\underline{\frac{P_{data}(x) + P_G(x)}{2}}} dx$$

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

$$\begin{aligned} \text{JSD}(P \parallel Q) &= \frac{1}{2}D(P \parallel M) + \frac{1}{2}D(Q \parallel M) \\ M &= \frac{1}{2}(P + Q) \end{aligned}$$

$$\max_D V(G, D) = V(G, D^*)$$

$$D^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_G(x)}$$

$$\begin{aligned} &= -2\log 2 + \int_x P_{data}(x) \log \frac{P_{data}(x)}{(P_{data}(x) + P_G(x))/2} dx \\ &\quad + \int_x P_G(x) \log \frac{P_G(x)}{(P_{data}(x) + P_G(x))/2} dx \end{aligned}$$

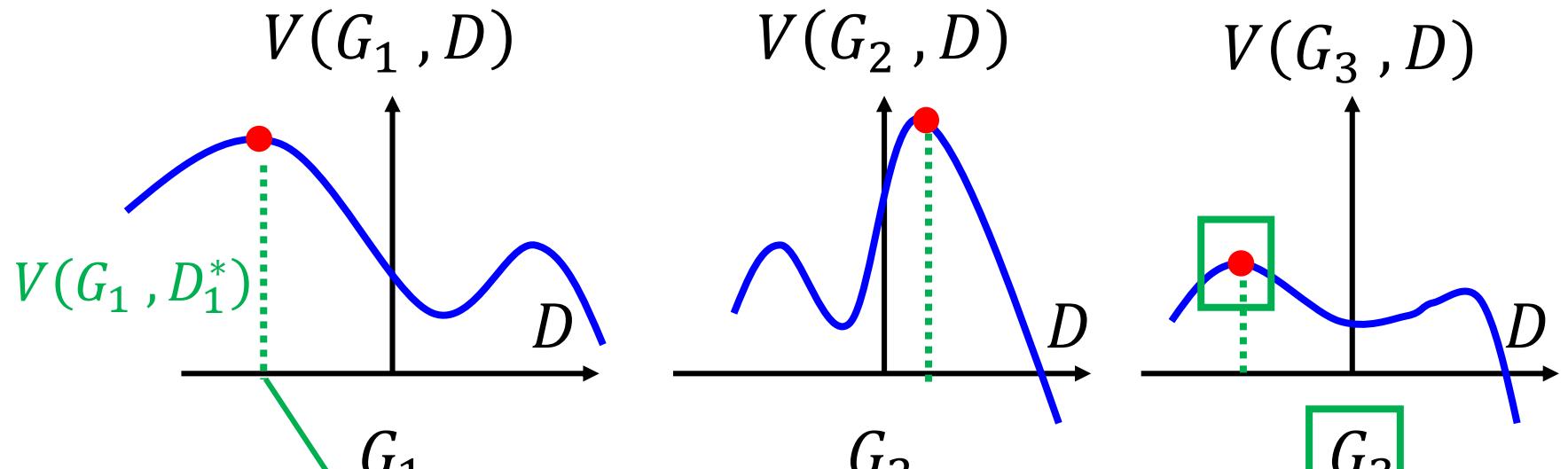
$$= -2\log 2 + \text{KL}\left(P_{data} \parallel \frac{P_{data} + P_G}{2}\right) + \text{KL}\left(P_G \parallel \frac{P_{data} + P_G}{2}\right)$$

$$= -2\log 2 + 2\text{JSD}(P_{data} \parallel P_G) \quad \text{Jensen-Shannon divergence}$$

$$G^* = \arg \min_G \max_D V(G, D)$$

$$D^* = \arg \max_D V(D, G)$$

The maximum objective value is related to JS divergence.



Divergence between P_{G_1} and P_{data}

$$G^* = \arg \min_G \max_D V(G, D)$$

$$D^* = \arg \max_D V(D, G)$$

The maximum objective value is related to JS divergence.

- Initialize generator and discriminator
- In each training iteration:

Step 1: Fix generator G , and update discriminator D

Step 2: Fix discriminator D , and update generator G

Algorithm

$$G^* = \arg \min_G \max_D V(G, D)$$

$L(G)$

- To find the best G minimizing the loss function $L(G)$,

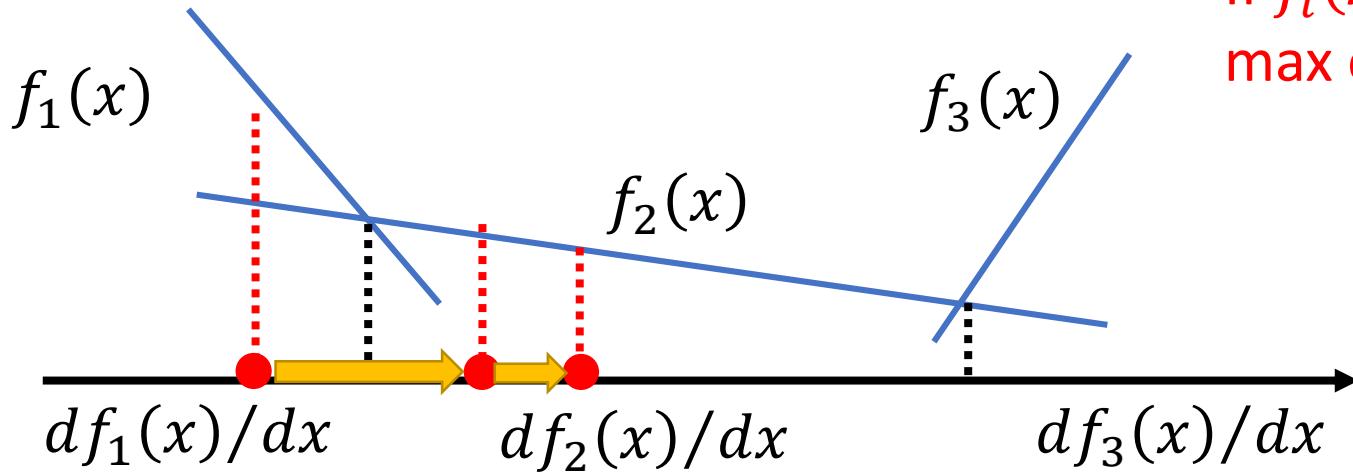
$$\theta_G \leftarrow \theta_G - \eta \partial L(G) / \partial \theta_G$$

θ_G defines G

$$f(x) = \max\{f_1(x), f_2(x), f_3(x)\}$$

$$\frac{df(x)}{dx} = ? \quad df_i(x)/dx$$

If $f_i(x)$ is the
max one



Algorithm

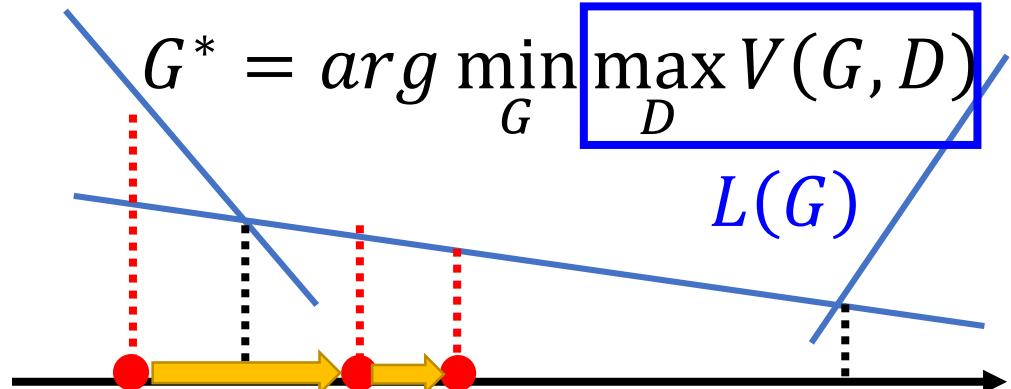
- Given G_0
- Find D_0^* maximizing $V(G_0, D)$ **Using Gradient Ascent**

$V(G_0, D_0^*)$ is the JS divergence between $P_{data}(x)$ and $P_{G_0}(x)$

- $\theta_G \leftarrow \theta_G - \eta \partial V(G, D_0^*) / \partial \theta_G \rightarrow$ Obtain G_1 Decrease JS divergence(?)
- Find D_1^* maximizing $V(G_1, D)$

$V(G_1, D_1^*)$ is the JS divergence between $P_{data}(x)$ and $P_{G_1}(x)$

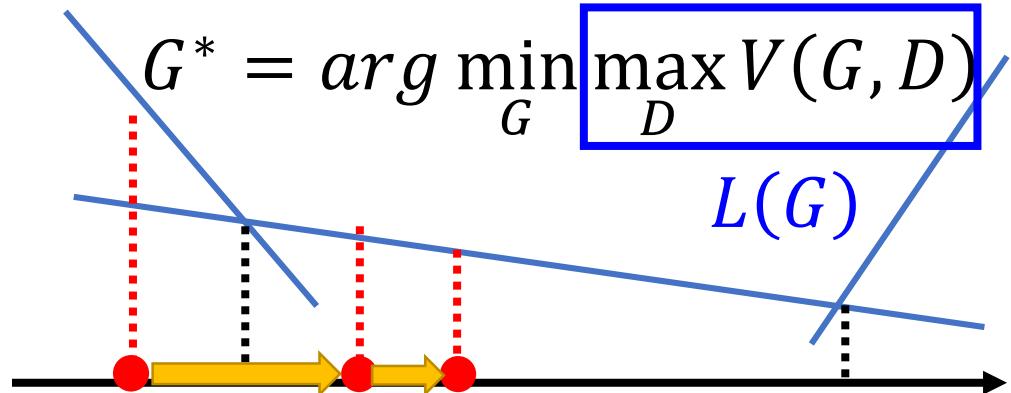
- $\theta_G \leftarrow \theta_G - \eta \partial V(G, D_1^*) / \partial \theta_G \rightarrow$ Obtain G_2 Decrease JS divergence(?)
-



Algorithm

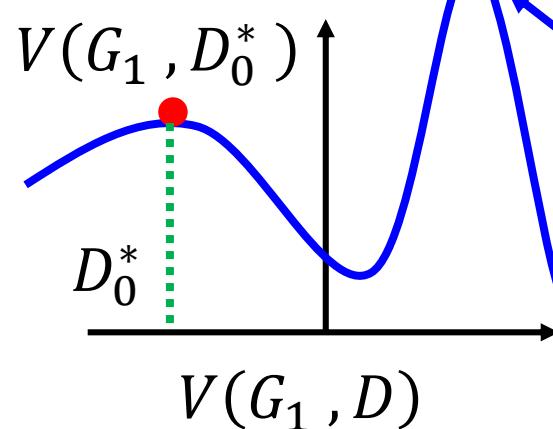
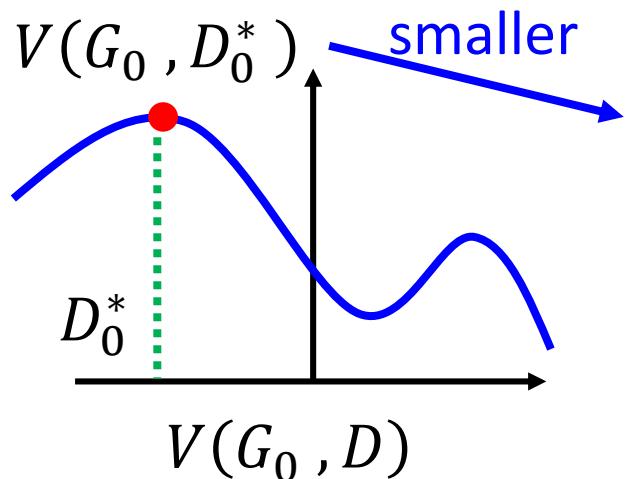
- Given G_0
- Find D_0^* maximizing $V(G_0, D)$

$V(G_0, D_0^*)$ is the JS divergence between $P_{data}(x)$ and $P_{G_0}(x)$



- $\theta_G \leftarrow \theta_G - \eta \partial V(G, D_0^*) / \partial \theta_G \rightarrow$ Obtain G_1

Decrease JS divergence(?)



$V(G_1, D_1^*) \dots$

Assume $D_0^* \approx D_1^*$

Don't update G too much

In practice ...

$$V = E_{x \sim P_{data}}[\log D(x)] + E_{x \sim P_G}[\log(1 - D(x))]$$

- Given G , how to compute $\max_D V(G, D)$
 - Sample $\{x^1, x^2, \dots, x^m\}$ from $P_{data}(x)$, sample $\{\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^m\}$ from generator $P_G(x)$

Maximize $\tilde{V} = \frac{1}{m} \sum_{i=1}^m \log D(x^i) + \frac{1}{m} \sum_{i=1}^m \log(1 - D(\tilde{x}^i))$

||

Binary Classifier

D is a binary classifier with sigmoid output (can be deep)

$\{x^1, x^2, \dots, x^m\}$ from $P_{data}(x)$ \rightarrow Positive examples

$\{\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^m\}$ from $P_G(x)$ \rightarrow Negative examples

Minimize Cross-entropy

Algorithm

Initialize θ_d for D and θ_g for G

- In each training iteration:

Can only find
lower bound of

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

- Sample m examples $\{x^1, x^2, \dots, x^m\}$ from data distribution $P_{data}(x)$
- Sample m noise samples $\{z^1, z^2, \dots, z^m\}$ from the prior $P_{prior}(z)$
- Obtaining generated data $\{\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^m\}$, $\tilde{x}^i = G(z^i)$
- Update discriminator parameters θ_d to maximize

- $$\tilde{V} = \frac{1}{m} \sum_{i=1}^m \log D(x^i) + \frac{1}{m} \sum_{i=1}^m \log (1 - D(\tilde{x}^i))$$
- $$\theta_d \leftarrow \theta_d + \eta \nabla \tilde{V}(\theta_d)$$

- Sample another m noise samples $\{z^1, z^2, \dots, z^m\}$ from the prior $P_{prior}(z)$

- Update generator parameters θ_g to minimize

- $$\tilde{V} = \frac{1}{m} \sum_{i=1}^m \log D(x^i) + \frac{1}{m} \sum_{i=1}^m \log (1 - D(G(z^i)))$$
- $$\theta_g \leftarrow \theta_g - \eta \nabla \tilde{V}(\theta_g)$$

Learning
D

Repeat
k times

Learning
G

Only
Once

Objective Function for Generator in Real Implementation

$$V = E_{x \sim P_{data}} [\log D(x)] + E_{x \sim P_G} [\log(1 - D(x))]$$

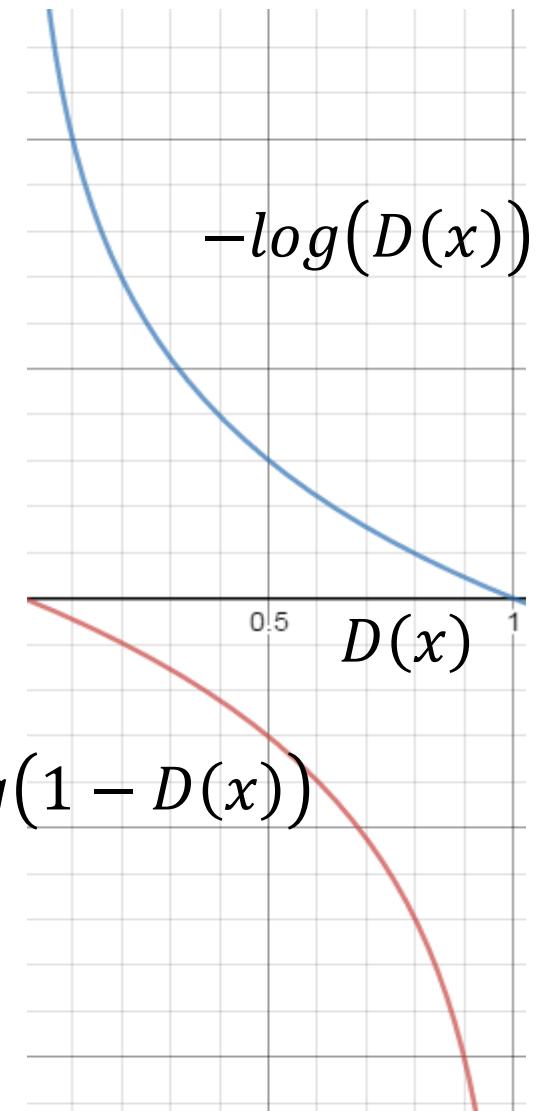
Slow at the beginning

Minimax GAN (MMGAN)

$$V = E_{x \sim P_G} [-\log(D(x))]$$

Real implementation:
label x from P_G as positive

Non-saturating GAN (NSGAN)

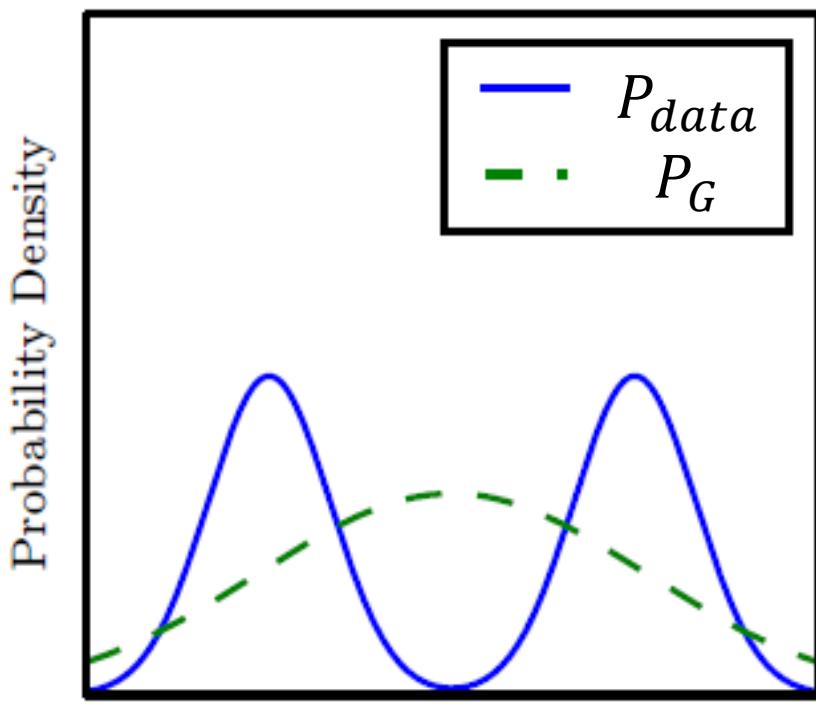


fGAN: General Framework of GAN

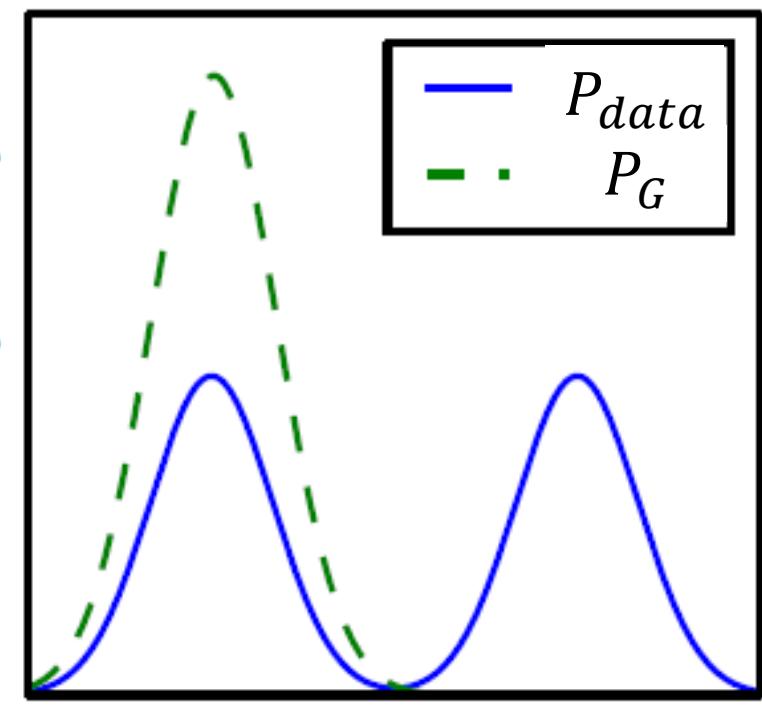
Flaw in Optimization?

$$KL = \int P_{data} \log \frac{P_{data}}{P_G} dx$$

$$\text{Reverse } KL = \int P_G \log \frac{P_G}{P_{data}} dx$$



x
Maximum likelihood
(minimize $KL(P_{data} || P_G)$)



x
Minimize $KL(P_G || P_{data})$
(reverse KL)

f-divergence

P and Q are two distributions. $p(x)$ and $q(x)$ are the probability of sampling x .

$$D_f(P||Q) = \int_x q(x)f\left(\frac{p(x)}{q(x)}\right)dx \quad \begin{array}{l} f \text{ is convex} \\ f(1) = 0 \end{array}$$

$D_f(P||Q)$ evaluates the difference of P and Q

If $p(x) = q(x)$ for all x

smallest

$$D_f(P||Q) = \int_x q(x)f\left(\frac{p(x)}{q(x)}\right)dx = 0$$

$$D_f(P||Q) = \int_x q(x)f\left(\frac{p(x)}{q(x)}\right)dx$$

Because f is convex

$$\geq f\left(\int_x q(x) \frac{p(x)}{q(x)} dx\right)$$

$$= f(1) = 0$$

If P and Q are the same distributions,
 $D_f(P||Q)$ has the smallest value, which is 0

f-divergence

$$D_f(P||Q) = \int_x q(x)f\left(\frac{p(x)}{q(x)}\right)dx \quad \begin{array}{l} f \text{ is convex} \\ f(1) = 0 \end{array}$$

$$f(x) = x \log x$$

$$D_f(P||Q) = \int_x q(x) \frac{p(x)}{q(x)} \log \left(\frac{p(x)}{q(x)} \right) dx = \int_x p(x) \log \left(\frac{p(x)}{q(x)} \right) dx \quad \text{KL}$$

$$f(x) = -\log x$$

$$D_f(P||Q) = \int_x q(x) \left(-\log \left(\frac{p(x)}{q(x)} \right) \right) dx = \int_x q(x) \log \left(\frac{q(x)}{p(x)} \right) dx \quad \text{Reverse KL}$$

$$f(x) = (x - 1)^2$$

$$D_f(P||Q) = \int_x q(x) \left(\frac{p(x)}{q(x)} - 1 \right)^2 dx = \int_x \frac{(p(x) - q(x))^2}{q(x)} dx \quad \text{Chi Square}$$

Fenchel Conjugate

$$D_f(P||Q) = \int_x q(x) f\left(\frac{p(x)}{q(x)}\right) dx$$

f is convex, $f(1) = 0$

- Every convex function f has a conjugate function f^*

$$f^*(t) = \max_{x \in \text{dom}(f)} \{xt - f(x)\}$$

$$f^*(\mathbf{t}_1) = \max_{x \in \text{dom}(f)} \{x\mathbf{t}_1 - f(x)\}$$

$$x_1 \mathbf{t}_1 - f(x_1)$$



$$f^*(\mathbf{t}_2) = \max_{x \in \text{dom}(f)} \{x\mathbf{t}_2 - f(x)\}$$

$$x_2 \mathbf{t}_1 - f(x_2)$$



$$x_3 \mathbf{t}_2 - f(x_3)$$



$$x_3 \mathbf{t}_1 - f(x_3)$$



$$x_2 \mathbf{t}_2 - f(x_2)$$



$$x_1 \mathbf{t}_2 - f(x_1)$$



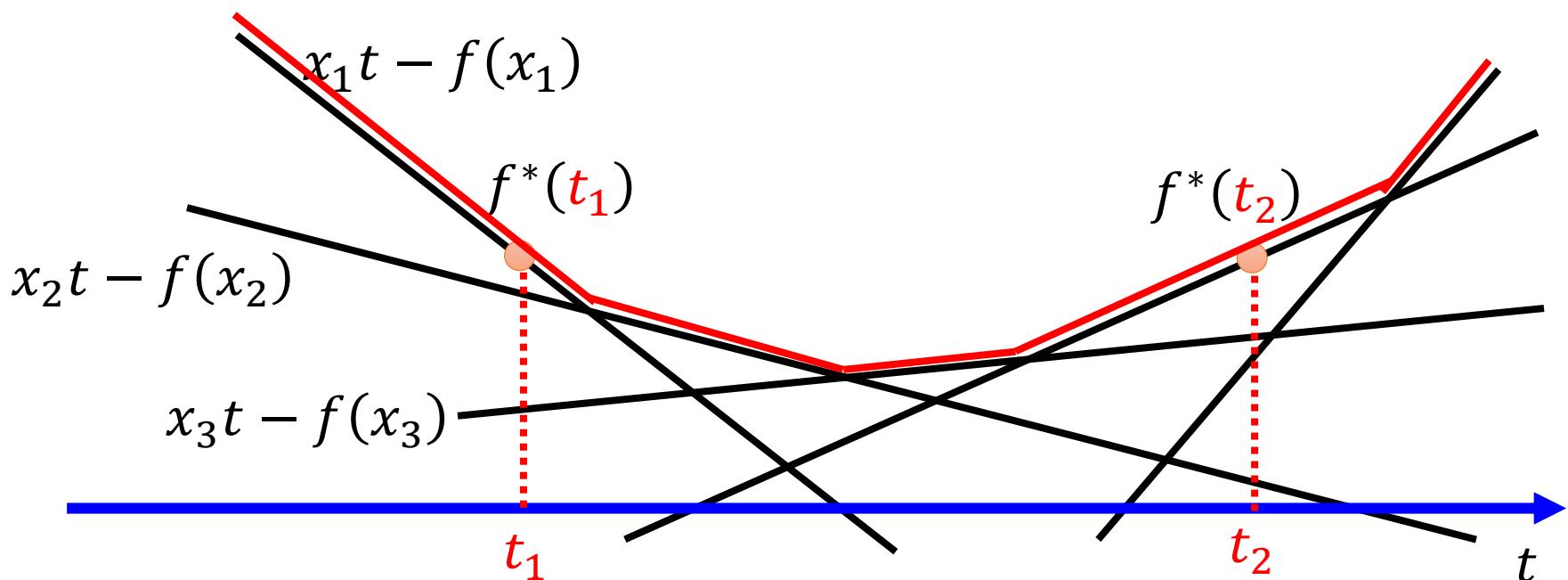
Fenchel Conjugate

$$D_f(P||Q) = \int_x q(x) f\left(\frac{p(x)}{q(x)}\right) dx$$

f is convex, $f(1) = 0$

- Every convex function f has a conjugate function f^*

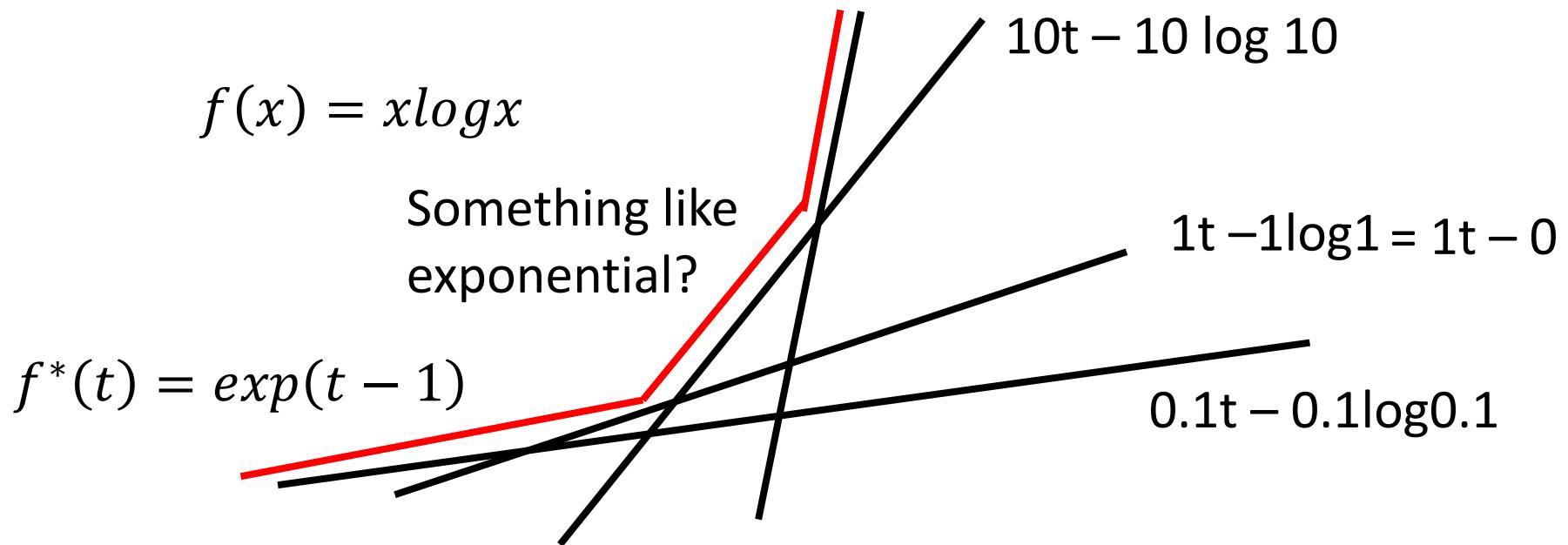
$$f^*(t) = \max_{x \in \text{dom}(f)} \{xt - f(x)\}$$



Fenchel Conjugate

- Every convex function f has a conjugate function f^*

$$f^*(t) = \max_{x \in \text{dom}(f)} \{xt - f(x)\}$$



Fenchel Conjugate

- Every convex function f has a conjugate function f^*
- $(f^*)^* = f$

$$f^*(t) = \max_{x \in \text{dom}(f)} \{xt - f(x)\}$$

$$f(x) = x \log x \quad \longleftrightarrow \quad f^*(t) = \exp(t - 1)$$

$$f^*(t) = \max_{x \in \text{dom}(f)} \{xt - x \log x\}$$

$$g(x) = xt - x \log x \quad \text{Given } t, \text{ find } x \text{ maximizing } g(x)$$

$$t - \log x - 1 = 0 \quad x = \exp(t - 1)$$

$$f^*(t) = \exp(t - 1) \times t - \exp(t - 1) \times (t - 1) = \exp(t - 1)$$

Connection with GAN

$$f^*(t) = \max_{x \in \text{dom}(f)} \{xt - f(x)\} \leftrightarrow f(\underline{x}) = \max_{t \in \text{dom}(f^*)} \{\underline{x}t - f^*(t)\}$$

$$D_f(P||Q) = \int_x q(x) f\left(\frac{p(x)}{q(x)}\right) dx \quad \boxed{\frac{p(x)}{q(x)}} \quad \boxed{\frac{p(x)}{q(x)}}$$

$$= \int_x q(x) \left(\max_{t \in \text{dom}(f^*)} \left\{ \frac{p(x)}{q(x)} \underline{t} - f^*(\underline{t}) \right\} \right) dx$$

$$\approx \max_D \int_x p(x) D(x) dx - \int_x q(x) f^*(D(x)) dx$$

D is a function
whose input is x,
and output is t

$$D_f(P||Q) \geq \int_x q(x) \left(\frac{p(x)}{q(x)} \underline{D(x)} - f^*(\underline{D(x)}) \right) dx$$

$$= \int_x p(x) D(x) dx - \int_x q(x) f^*(D(x)) dx$$

Connection with GAN

$$\begin{aligned} D_f(P||Q) &\approx \max_{\text{D}} \int_x p(x)D(x)dx - \int_x q(x)f^*(D(x))dx \\ &= \max_{\text{D}} \left\{ E_{x \sim P}[D(x)] - E_{x \sim Q}[f^*(D(x))] \right\} \end{aligned}$$

Samples from P Samples from Q

$$D_f(P_{data}||P_G) = \max_{\text{D}} \left\{ E_{x \sim P_{data}}[D(x)] - E_{x \sim P_G}[f^*(D(x))] \right\}$$

$$\begin{aligned} G^* &= \arg \min_G D_f(P_{data}||P_G) && \text{Original GAN has} \\ &= \arg \min_G \max_D \left\{ E_{x \sim P_{data}}[D(x)] - E_{x \sim P_G}[f^*(D(x))] \right\} && \text{different V(G,D)} \\ &= \arg \min_G \max_D V(G, D) \quad \text{familiar? } \smiley \end{aligned}$$

$$D_f(P_{data} || P_G) = \max_D \{ E_{x \sim P_{data}} [D(x)] - E_{x \sim P_G} [f^*(D(x))] \}$$

Name	$D_f(P\ Q)$	Generator $f(u)$
Total variation	$\frac{1}{2} \int p(x) - q(x) dx$	$\frac{1}{2} u - 1 $
Kullback-Leibler	$\int p(x) \log \frac{p(x)}{q(x)} dx$	$u \log u$
Reverse Kullback-Leibler	$\int q(x) \log \frac{q(x)}{p(x)} dx$	$-\log u$
Pearson χ^2	$\int \frac{(q(x)-p(x))^2}{p(x)} dx$	$(u - 1)^2$
Neyman χ^2	$\int \frac{(p(x)-q(x))^2}{q(x)} dx$	$\frac{(1-u)^2}{u}$
Squared Hellinger	$\int \left(\sqrt{p(x)} - \sqrt{q(x)} \right)^2 dx$	$(\sqrt{u} - 1)^2$
Jeffrey	$\int (p(x) - q(x)) \log \left(\frac{p(x)}{q(x)} \right) dx$	$(u - 1) \log u$
Jensen-Shannon	$\frac{1}{2} \int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} dx$	$-(u + 1) \log \frac{1+u}{2} + u \log u$
Jensen-Shannon-weighted	$\int p(x) \pi \log \frac{p(x)}{\pi p(x)+(1-\pi)q(x)} + (1 - \pi)q(x) \log \frac{q(x)}{\pi p(x)+(1-\pi)q(x)} dx$	$\pi u \log u - (1 - \pi + \pi u) \log(1 - \pi + \pi u)$
GAN	$\int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} dx - \log(4)$	$u \log u - (u + 1) \log(u + 1)$

Using the f-divergence
you like ☺

<https://arxiv.org/pdf/1606.00709.pdf>

Name	Conjugate $f^*(t)$
Total variation	t
Kullback-Leibler (KL)	$\exp(t - 1)$
Reverse KL	$-1 - \log(-t)$
Pearson χ^2	$\frac{1}{4}t^2 + t$
Neyman χ^2	$2 - 2\sqrt{1-t}$
Squared Hellinger	$\frac{t}{1-t}$
Jeffrey	$W(e^{1-t}) + \frac{1}{W(e^{1-t})} + t - 2$
Jensen-Shannon	$-\log(2 - \exp(t))$
Jensen-Shannon-weighted	$(1 - \pi) \log \frac{1 - \pi}{1 - \pi e^{t/\pi}}$
GAN	$-\log(1 - \exp(t))$

Tips for Improving GAN

Martin Arjovsky, Soumith Chintala, Léon Bottou, Wasserstein GAN, arXiv preprint, 2017

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, Aaron Courville,
“Improved Training of Wasserstein GANs”, arXiv preprint, 2017

JS divergence is not suitable

- In most cases, P_G and P_{data} are not overlapped.
- 1. The nature of data

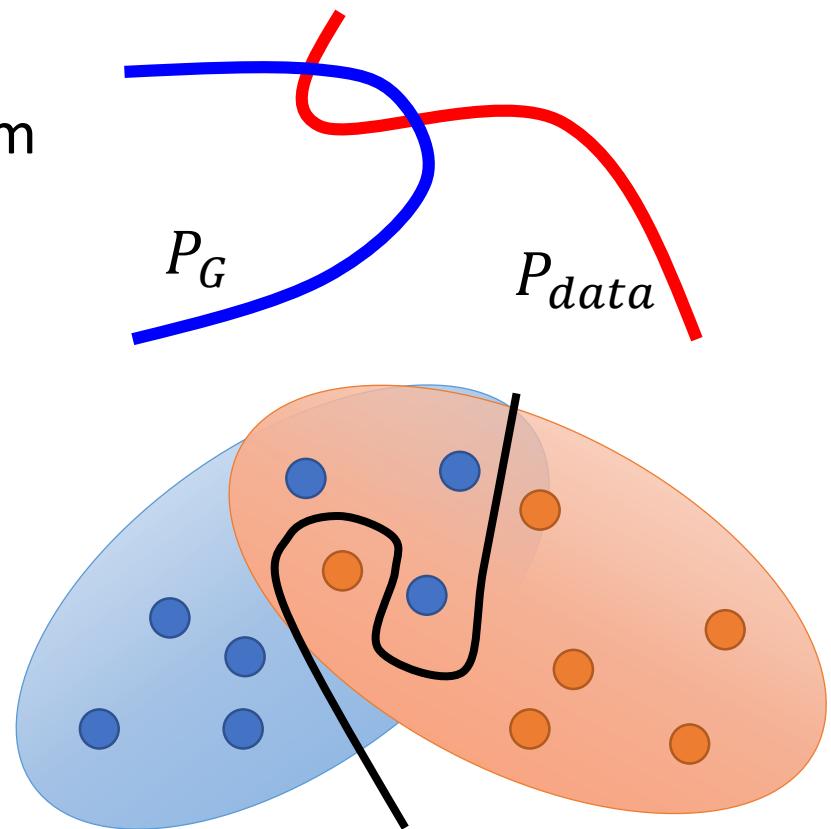
Both P_{data} and P_G are low-dim manifold in high-dim space.

The overlap can be ignored.

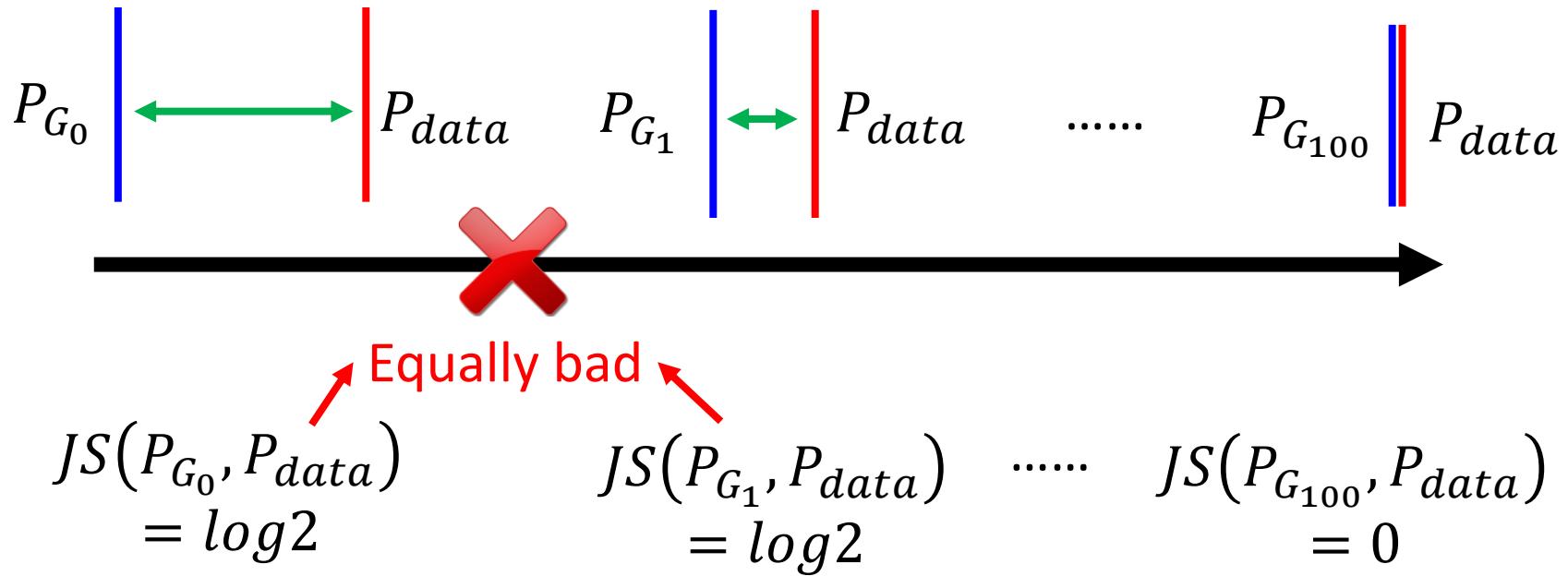
- 2. Sampling

Even though P_{data} and P_G have overlap.

If you do not have enough sampling



What is the problem of JS divergence?

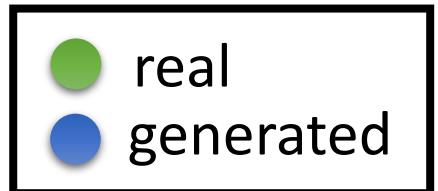


JS divergence is $\log 2$ if two distributions do not overlap.

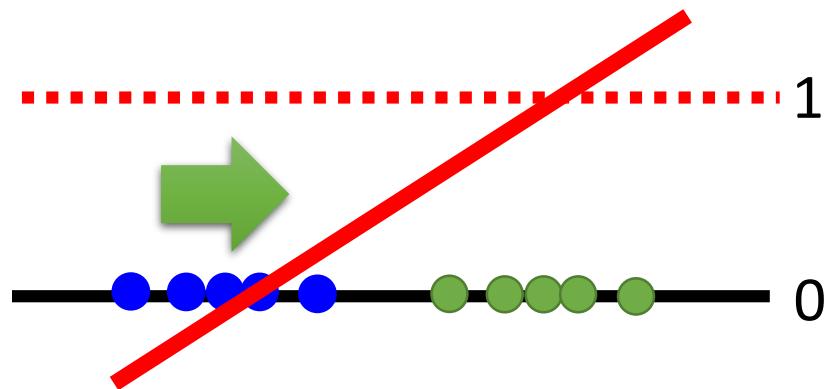
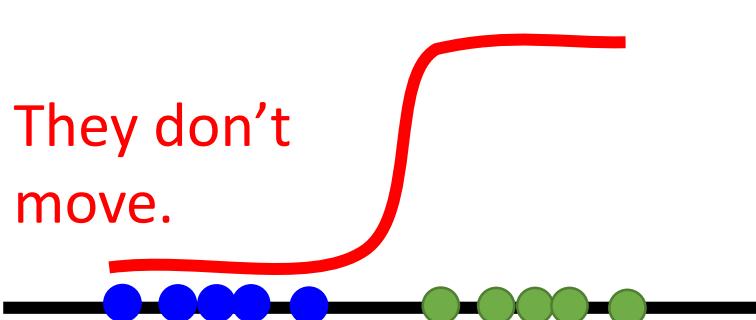
Intuition: If two distributions do not overlap, binary classifier achieves 100% accuracy

→ Same objective value is obtained. → Same divergence

Least Square GAN (LSGAN)

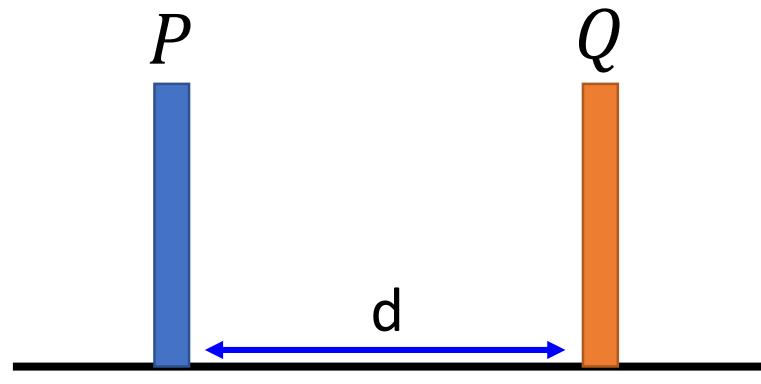


- Replace sigmoid with linear (replace classification with regression)



Wasserstein GAN (WGAN): Earth Mover's Distance

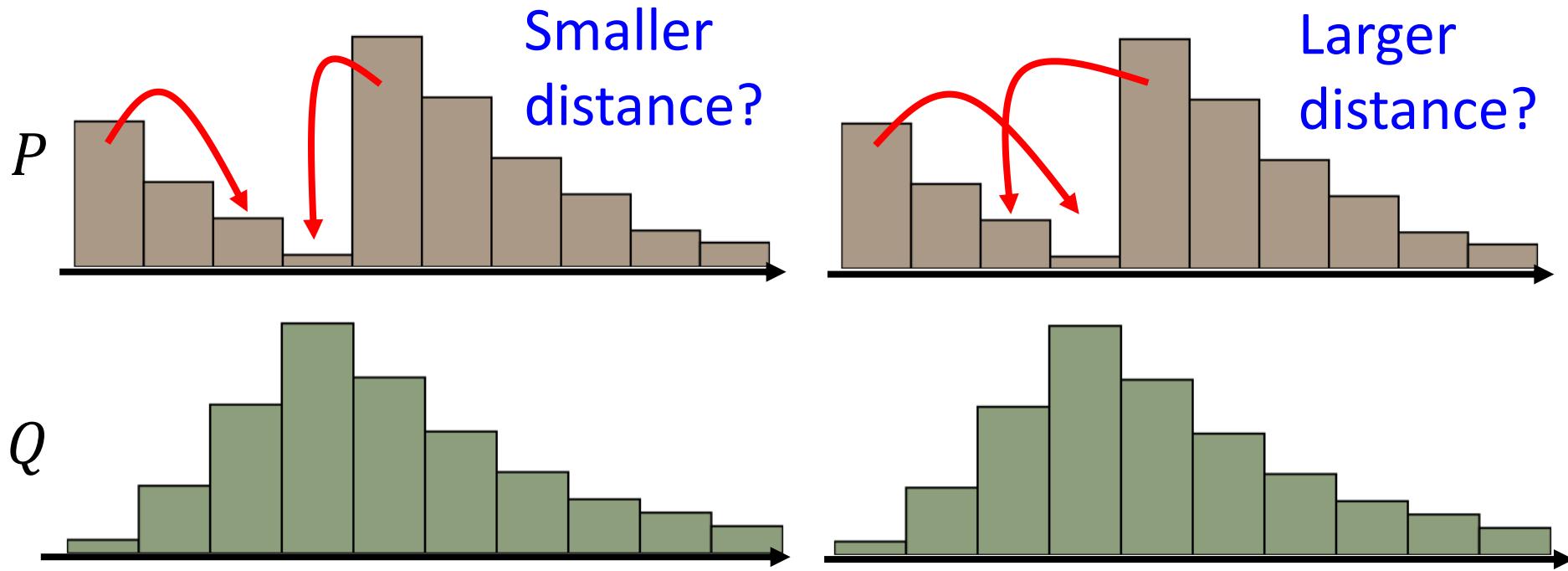
- Considering one distribution P as a pile of earth, and another distribution Q as the target
- The average distance the earth mover has to move the earth.



$$W(P, Q) = d$$



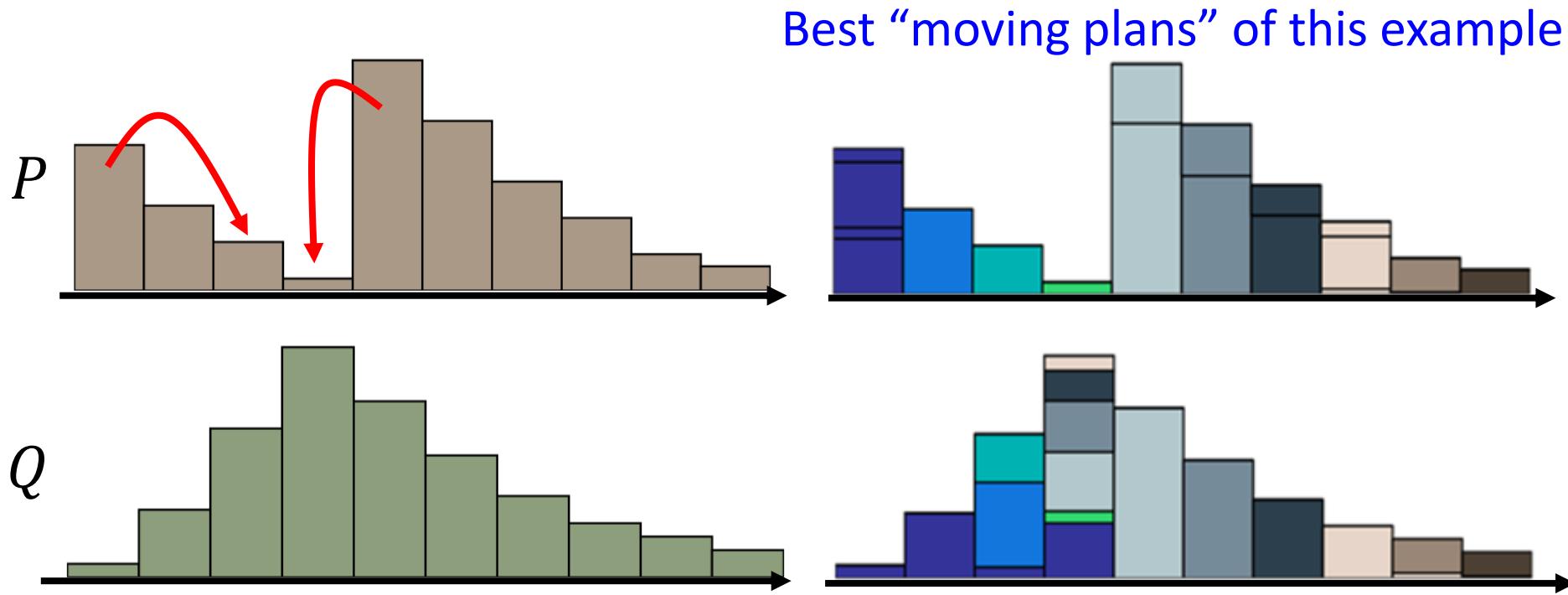
WGAN: Earth Mover's Distance



There many possible “moving plans”.

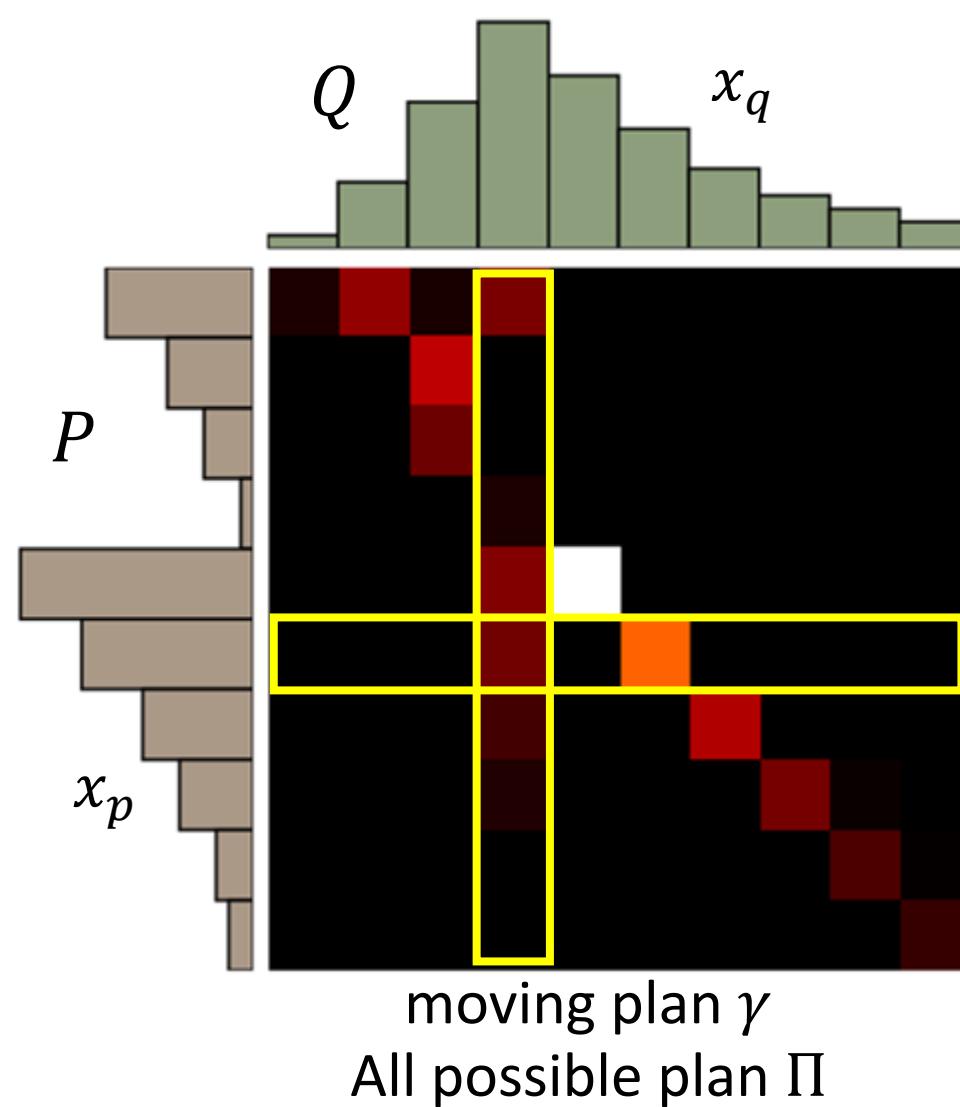
Using the “moving plan” with the smallest average distance to define the earth mover’s distance.

WGAN: Earth Mover's Distance



There many possible “moving plans”.

Using the “moving plan” with the smallest average distance to define the earth mover’s distance.



A “moving plan” is a matrix
The value of the element is the
amount of earth from one
position to another.

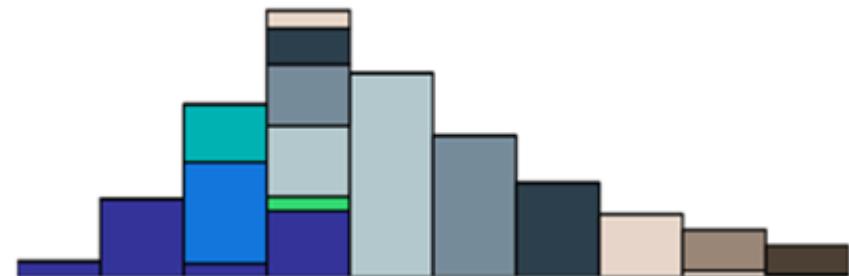
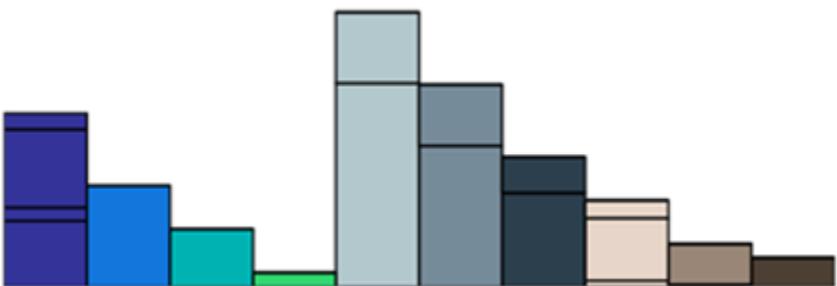
Average distance of a plan γ :

$$B(\gamma) = \sum_{x_p, x_q} \gamma(x_p, x_q) \|x_p - x_q\|$$

Earth Mover’s Distance:

$$W(P, Q) = \min_{\gamma \in \Pi} B(\gamma)$$

The best plan

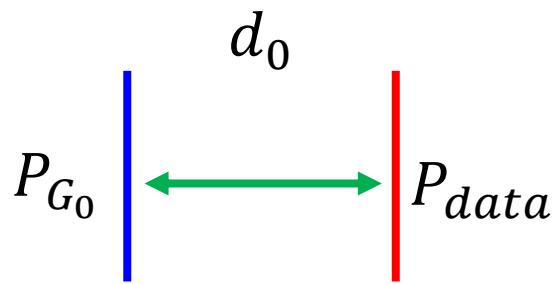
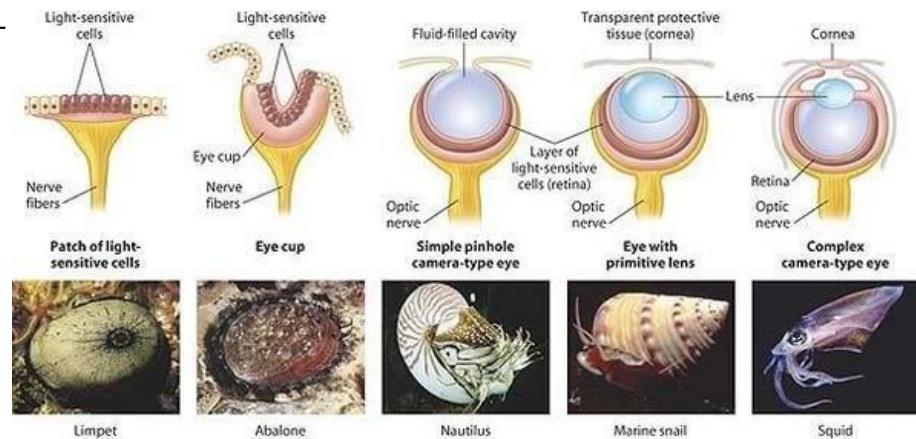


Why Earth Mover's Distance?

$$D_f(P_{data} || P_G)$$

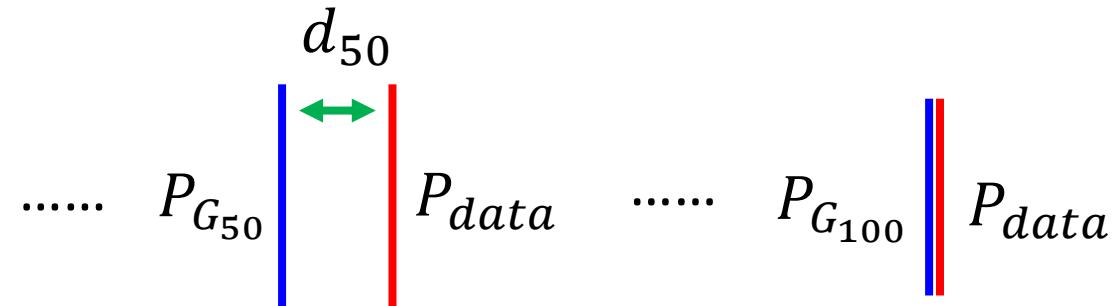


$$W(P_{data}, P_G)$$



$$JS(P_{G_0}, P_{data}) = \log 2$$

$$W(P_{G_0}, P_{data}) = d_0$$



$$JS(P_{G_{50}}, P_{data}) = \log 2$$

$$W(P_{G_{50}}, P_{data}) = d_{50}$$

$$JS(P_{G_{100}}, P_{data}) = 0$$

$$W(P_{G_{100}}, P_{data}) = 0$$

WGAN

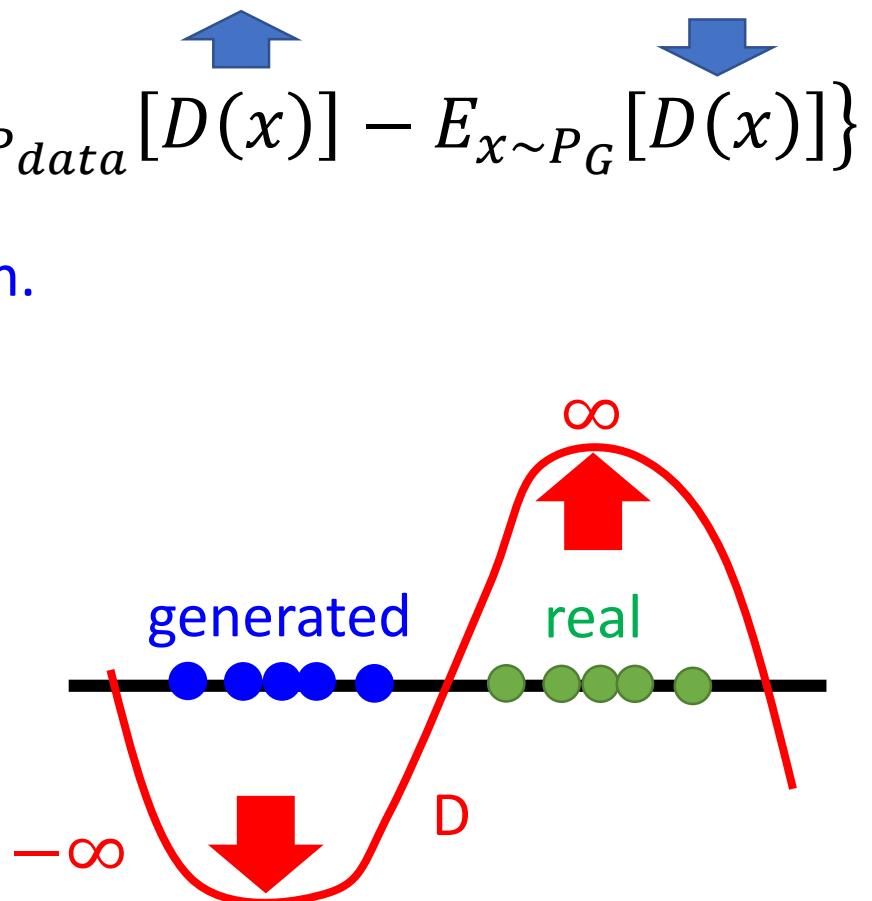
Evaluate wasserstein distance between P_{data} and P_G

$$V(G, D) = \max_{\substack{D \in 1-\text{Lipschitz}}} \left\{ E_{x \sim P_{data}} [D(x)] - E_{x \sim P_G} [D(x)] \right\}$$

D has to be smooth enough.

Without the constraint, the training of D will not converge.

Keeping the D smooth forces $D(x)$ become ∞ and $-\infty$



Weight Clipping [Martin Arjovsky, et al., arXiv, 2017]

WGAN

Force the parameters w between c and -c
After parameter update, if $w > c$, $w = c$;
if $w < -c$, $w = -c$

Evaluate wasserstein distance between P_{data} and P_G

$$V(G, D) = \max_{D \in \text{1-Lipschitz}} \left\{ E_{x \sim P_{data}}[D(x)] - E_{x \sim P_G}[D(x)] \right\}$$

D has to be smooth enough. How to fulfill this constraint?

Lipschitz Function

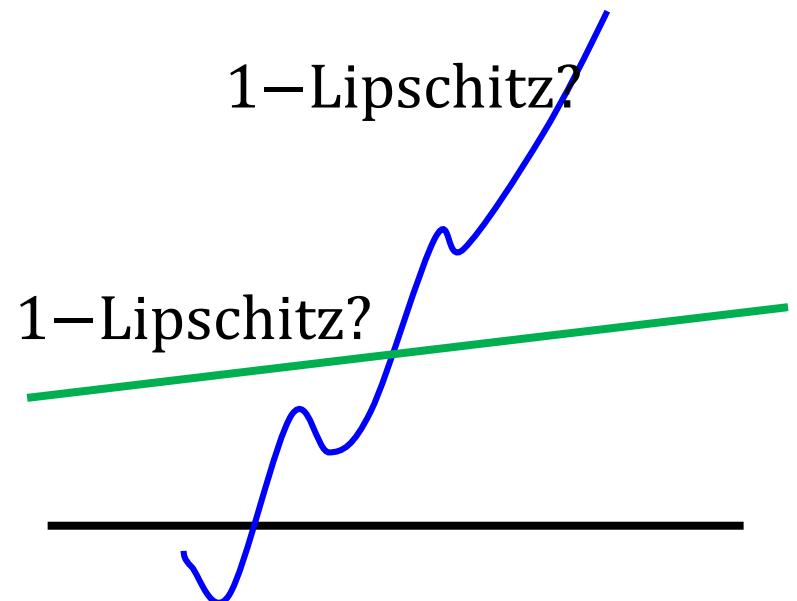
$$\|f(x_1) - f(x_2)\| \leq K \|x_1 - x_2\|$$

Output
change

Input
change

K=1 for "1 – Lipschitz"

Do not change fast



Improved WGAN (WGAN-GP)

$$V(G, D) = \max_{D \in 1-\text{Lipschitz}} \{E_{x \sim P_{data}}[D(x)] - E_{x \sim P_G}[D(x)]\}$$

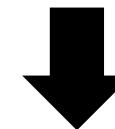
A differentiable function is 1-Lipschitz if and only if it has gradients with norm less than or equal to 1 everywhere.

$$D \in 1 - \text{Lipschitz} \iff \|\nabla_x D(x)\| \leq 1 \text{ for all } x$$

$$V(G, D) \approx \max_D \{E_{x \sim P_{data}}[D(x)] - E_{x \sim P_G}[D(x)]\}$$

$$- \lambda \int_x \max(0, \|\nabla_x D(x)\| - 1) dx\}$$

Prefer $\|\nabla_x D(x)\| \leq 1$ for all x

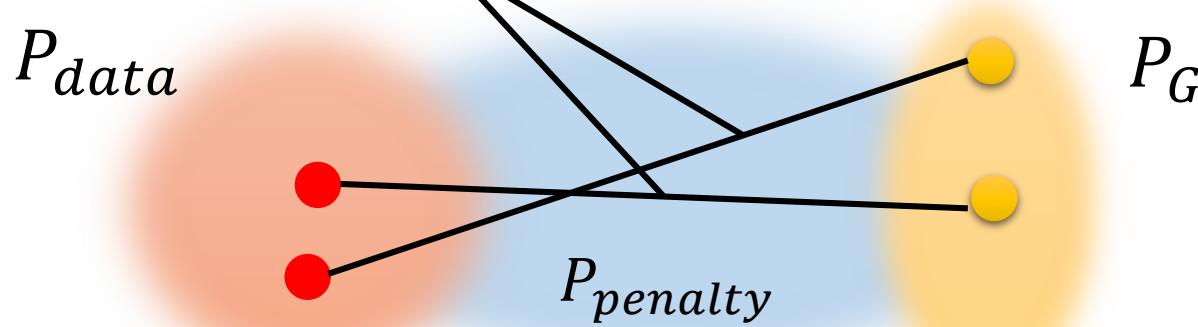


$$- \lambda E_{x \sim P_{penalty}} [\max(0, \|\nabla_x D(x)\| - 1)]\}$$

Prefer $\|\nabla_x D(x)\| \leq 1$ for x sampling from $x \sim P_{penalty}$

Improved WGAN (WGAN-GP)

$$V(G, D) \approx \max_D \{ E_{x \sim P_{data}}[D(x)] - E_{x \sim P_G}[D(x)] \\ - \lambda E_{x \sim P_{penalty}}[\max(0, \|\nabla_x D(x)\| - 1)] \}$$

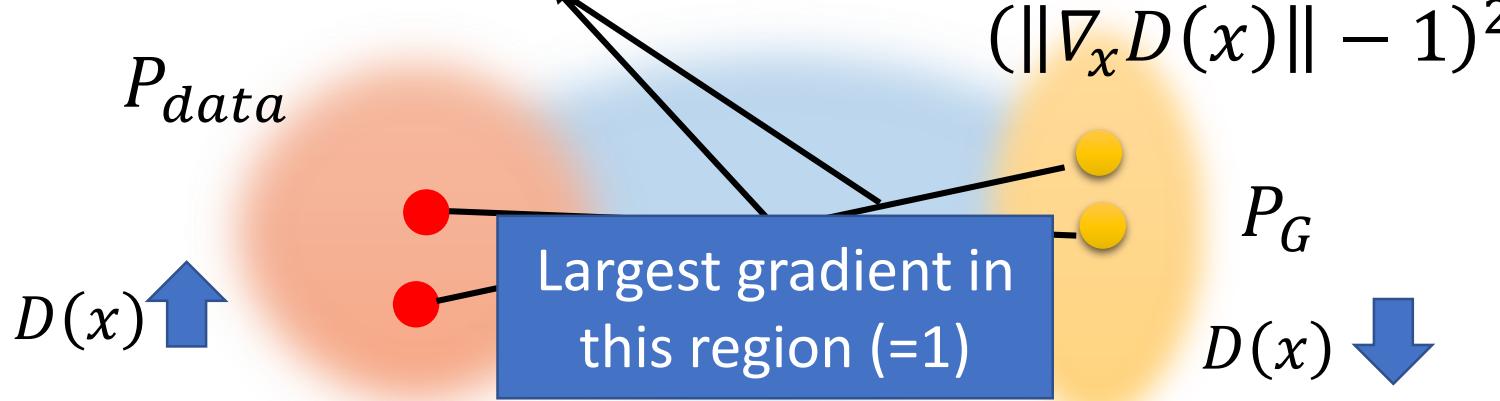


“Given that enforcing the Lipschitz constraint everywhere is intractable, enforcing it ***only along these straight lines*** seems sufficient and experimentally results in good performance.”

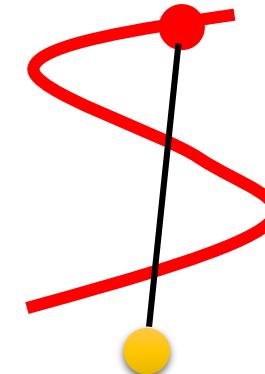
Only give gradient constraint to the region between P_{data} and P_G because they influence how P_G moves to P_{data}

Improved WGAN (WGAN-GP)

$$V(G, D) \approx \max_D \{ E_{x \sim P_{data}}[D(x)] - E_{x \sim P_G}[D(x)] \\ - \lambda E_{x \sim P_{penalty}} [\max(0, \|\nabla_x D(x)\| - 1)] \}$$



“Simply penalizing overly large gradients also works in theory, but experimentally we found that this approach converged faster and to better optima.”



Spectrum Norm

Spectral Normalization → Keep gradient norm smaller than 1 everywhere [Miyato, et al., ICLR, 2018]



Algorithm of

WGAN

- In each training iteration:

No sigmoid for the output of D

- Sample m examples $\{x^1, x^2, \dots, x^m\}$ from data distribution $P_{data}(x)$
- Sample m noise samples $\{z^1, z^2, \dots, z^m\}$ from the prior $P_{prior}(z)$
- Obtaining generated data $\{\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^m\}$, $\tilde{x}^i = G(z^i)$
- Update discriminator parameters θ_d to maximize

$$\cdot \tilde{V} = \frac{1}{m} \sum_{i=1}^m D(x^i) - \frac{1}{m} \sum_{i=1}^m D(\tilde{x}^i)$$

$$\cdot \theta_d \leftarrow \theta_d + \eta \nabla \tilde{V}(\theta_d)$$

- Sample another m noise samples $\{z^1, z^2, \dots, z^m\}$ from the prior $P_{prior}(z)$

Weight clipping /
Gradient Penalty ...

- Update generator parameters θ_g to minimize

$$\cdot \tilde{V} = \frac{1}{m} \sum_{i=1}^m \log D(x^i) - \frac{1}{m} \sum_{i=1}^m \log D(G(z^i))$$

$$\cdot \theta_g \leftarrow \theta_g - \eta \nabla \tilde{V}(\theta_g)$$

Learning
D

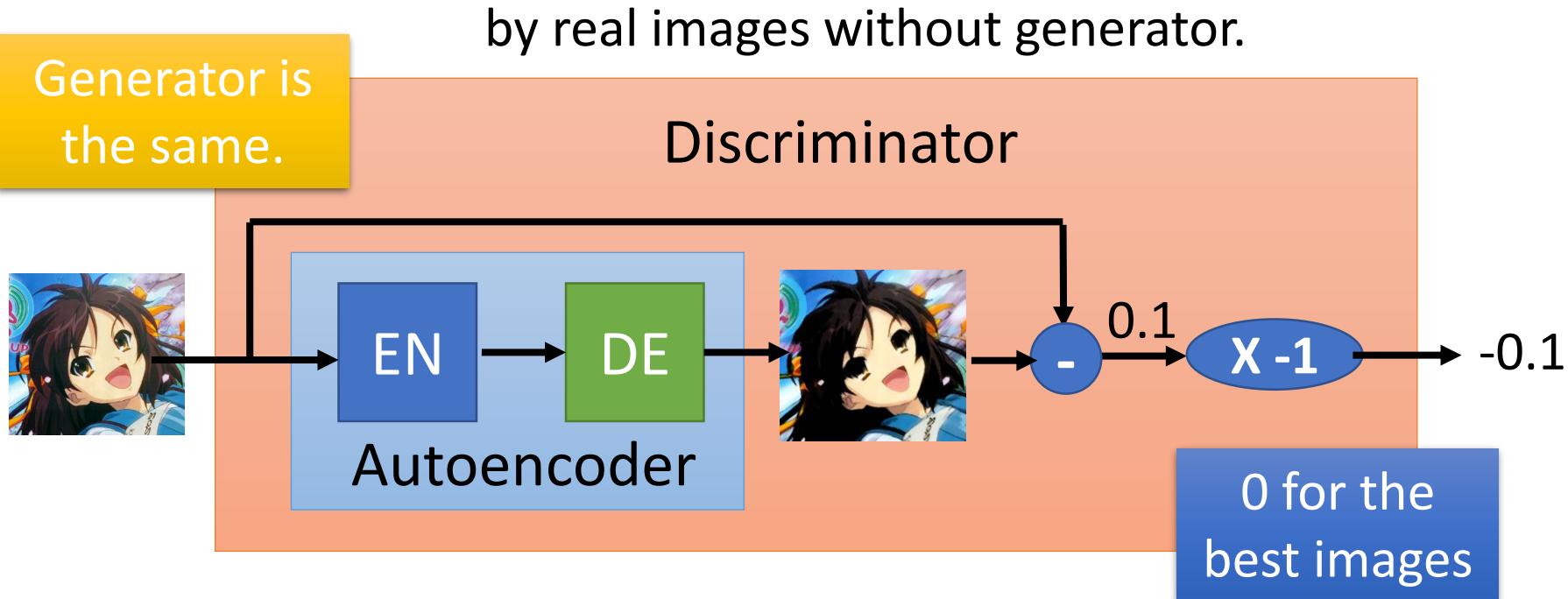
Repeat
k times

Learning
G

Only
Once

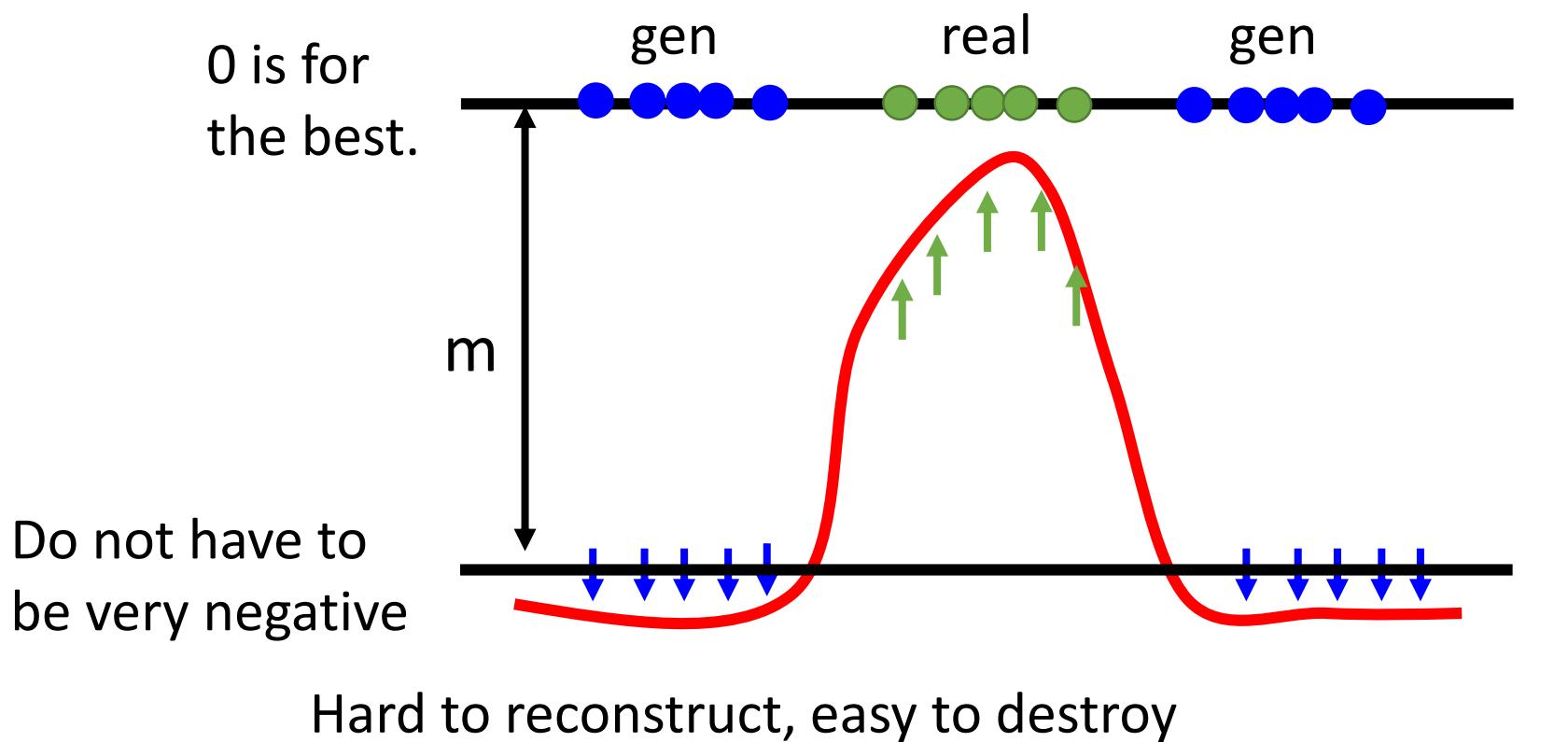
Energy-based GAN (EBGAN)

- Using an autoencoder as discriminator D
 - Using the negative reconstruction error of auto-encoder to determine the goodness
 - **Benefit:** The auto-encoder can be pre-train by real images without generator.



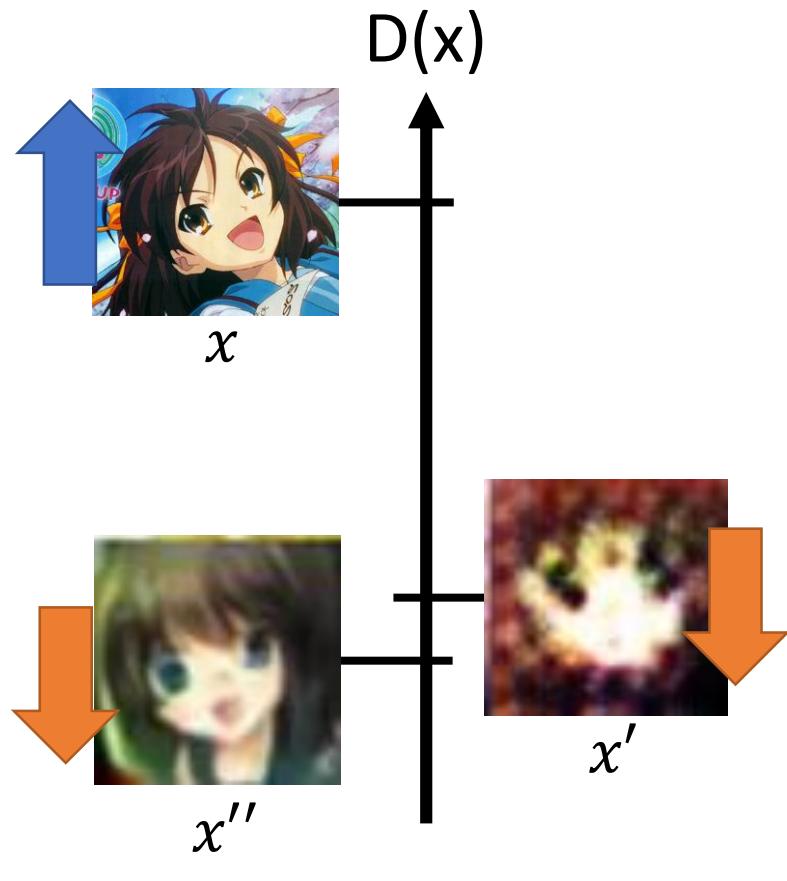
EBGAN

Auto-encoder based discriminator
only gives limited region large value.

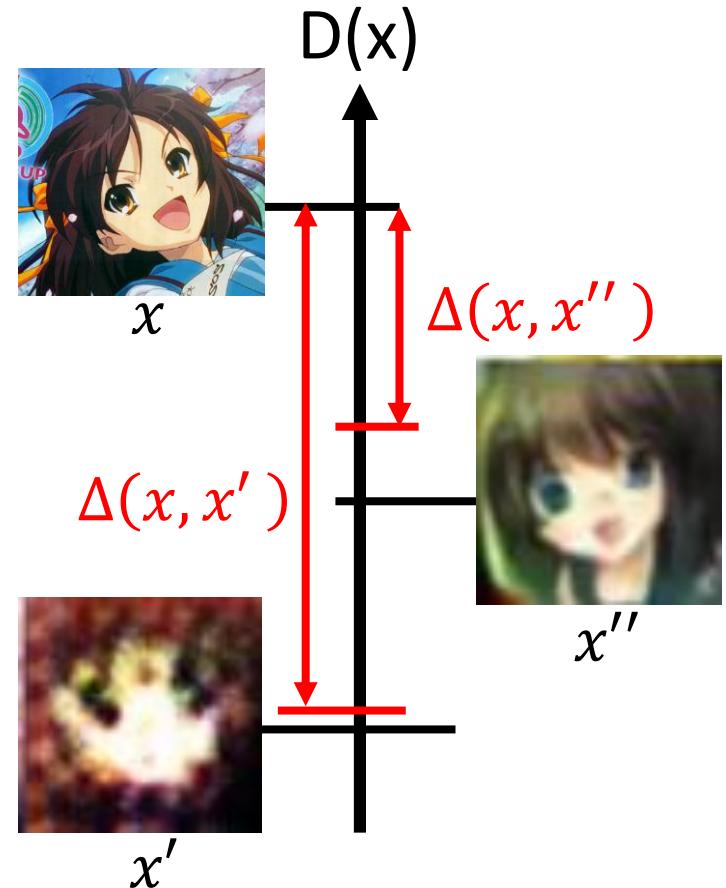


Outlook: Loss-sensitive GAN (LSGAN)

WGAN



LSGAN



Reference

- Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, Generative Adversarial Networks, NIPS, 2014
- Sebastian Nowozin, Botond Cseke, Ryota Tomioka, “f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization”, NIPS, 2016
- Martin Arjovsky, Soumith Chintala, Léon Bottou, Wasserstein GAN, arXiv, 2017
- Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, Aaron Courville, Improved Training of Wasserstein GANs, NIPS, 2017
- Junbo Zhao, Michael Mathieu, Yann LeCun, Energy-based Generative Adversarial Network, arXiv, 2016
- Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, Olivier Bousquet, “Are GANs Created Equal? A Large-Scale Study”, arXiv, 2017
- Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen Improved Techniques for Training GANs, NIPS, 2016

Reference

- Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Sepp Hochreiter, GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium, NIPS, 2017
- Naveen Kodali, Jacob Abernethy, James Hays, Zsolt Kira, “On Convergence and Stability of GANs”, arXiv, 2017
- Xiang Wei, Boqing Gong, Zixia Liu, Wei Lu, Liqiang Wang, Improving the Improved Training of Wasserstein GANs: A Consistency Term and Its Dual Effect, ICLR, 2018
- Takeru Miyato, Toshiki Kataoka, Masanori Koyama, Yuichi Yoshida, Spectral Normalization for Generative Adversarial Networks, ICLR, 2018

Feature Extraction

InfoGAN

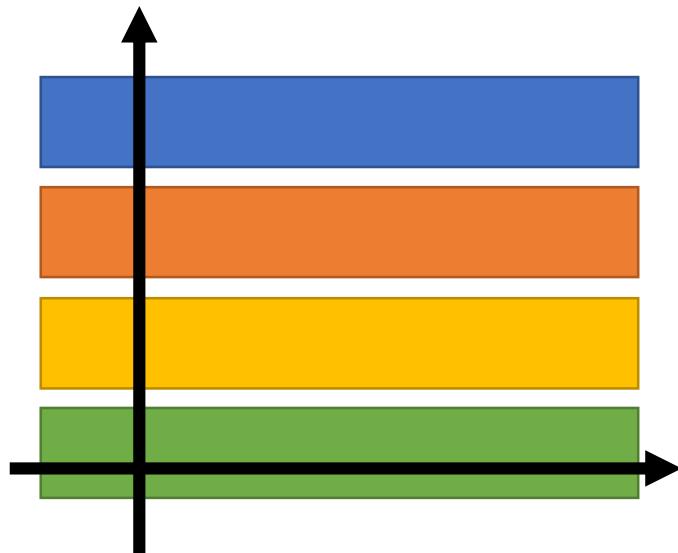
(The colors represents the characteristics.)

Regular
GAN

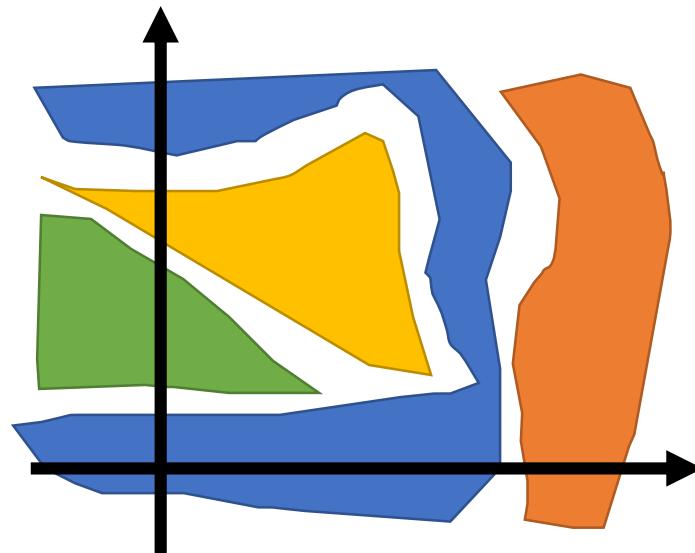


Modifying a specific dimension,
no clear meaning

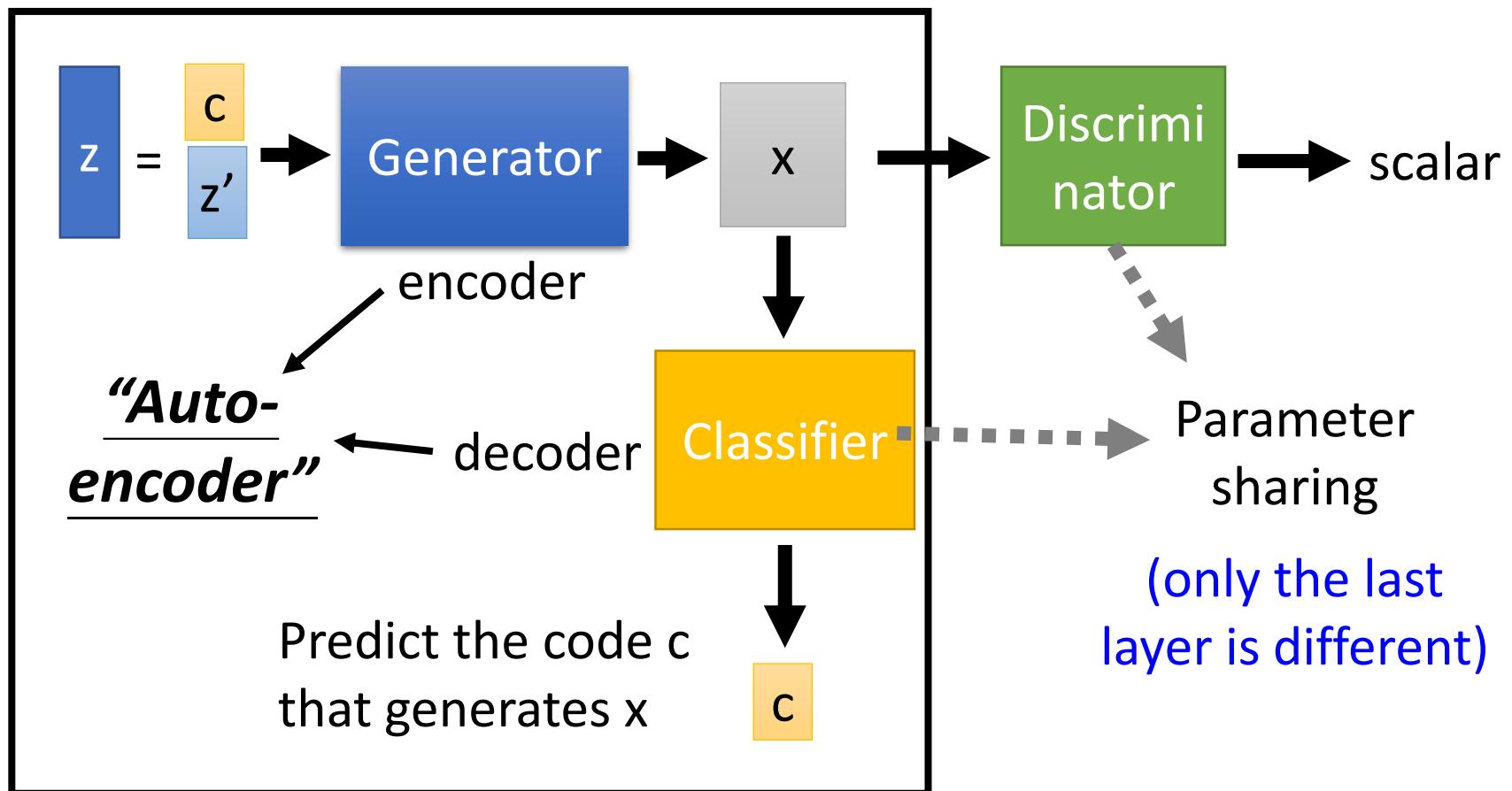
What we expect



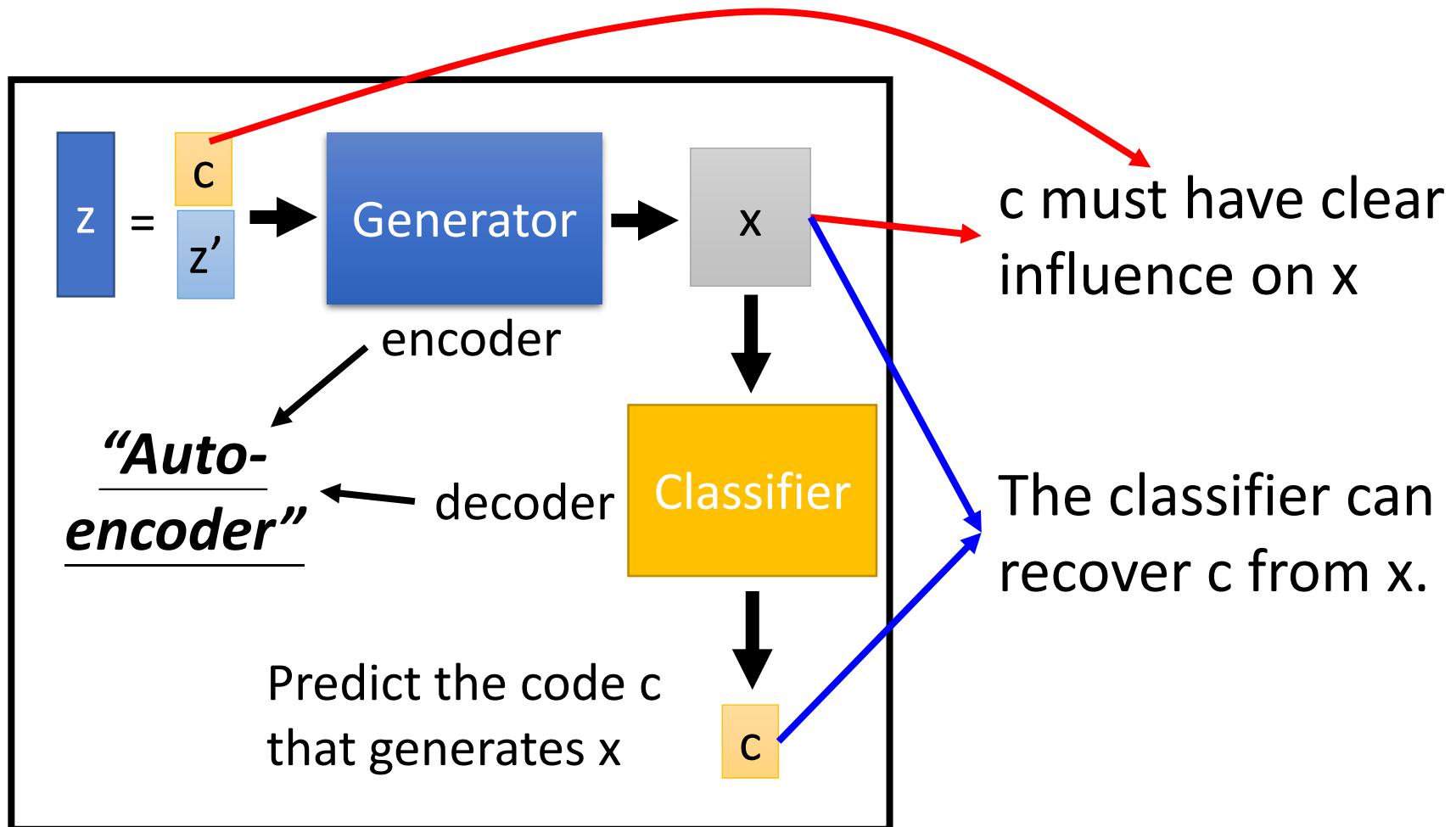
Actually ...

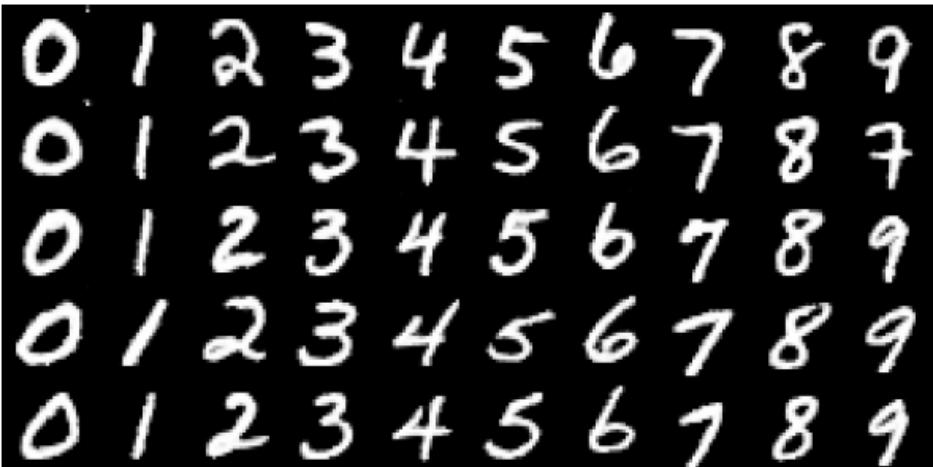


What is InfoGAN?

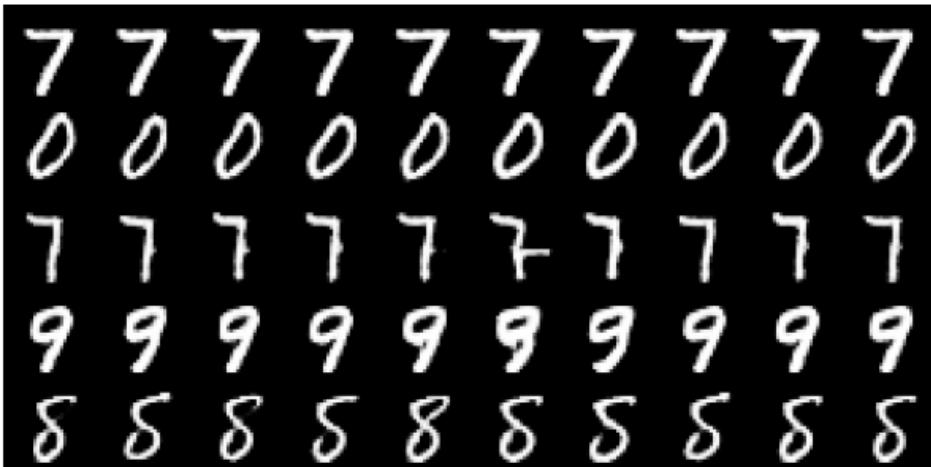


What is InfoGAN?

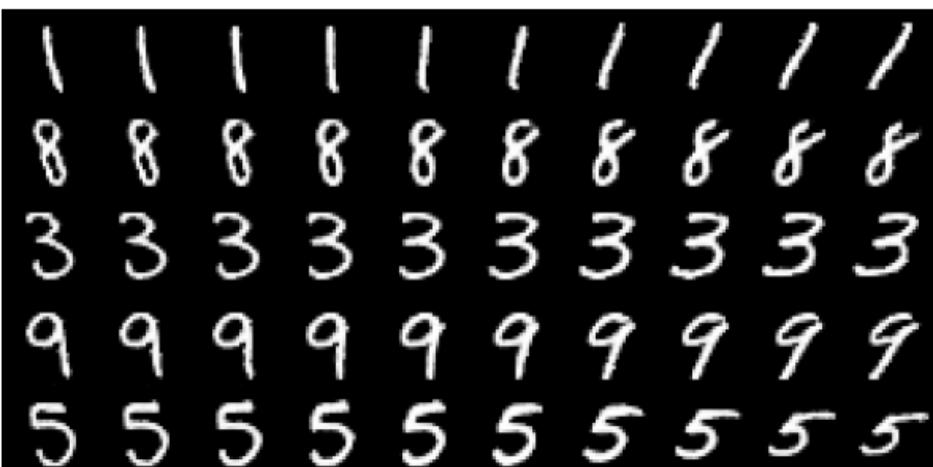




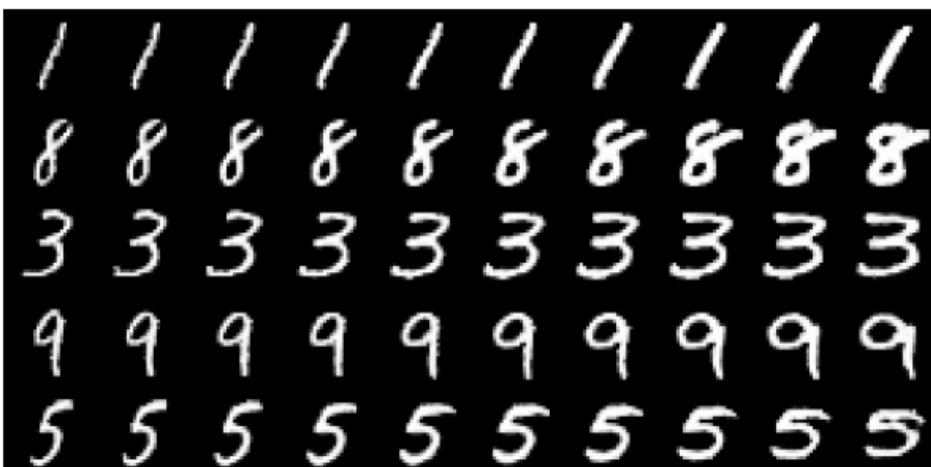
(a) Varying c_1 on InfoGAN (Digit type)



(b) Varying c_1 on regular GAN (No clear meaning)



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

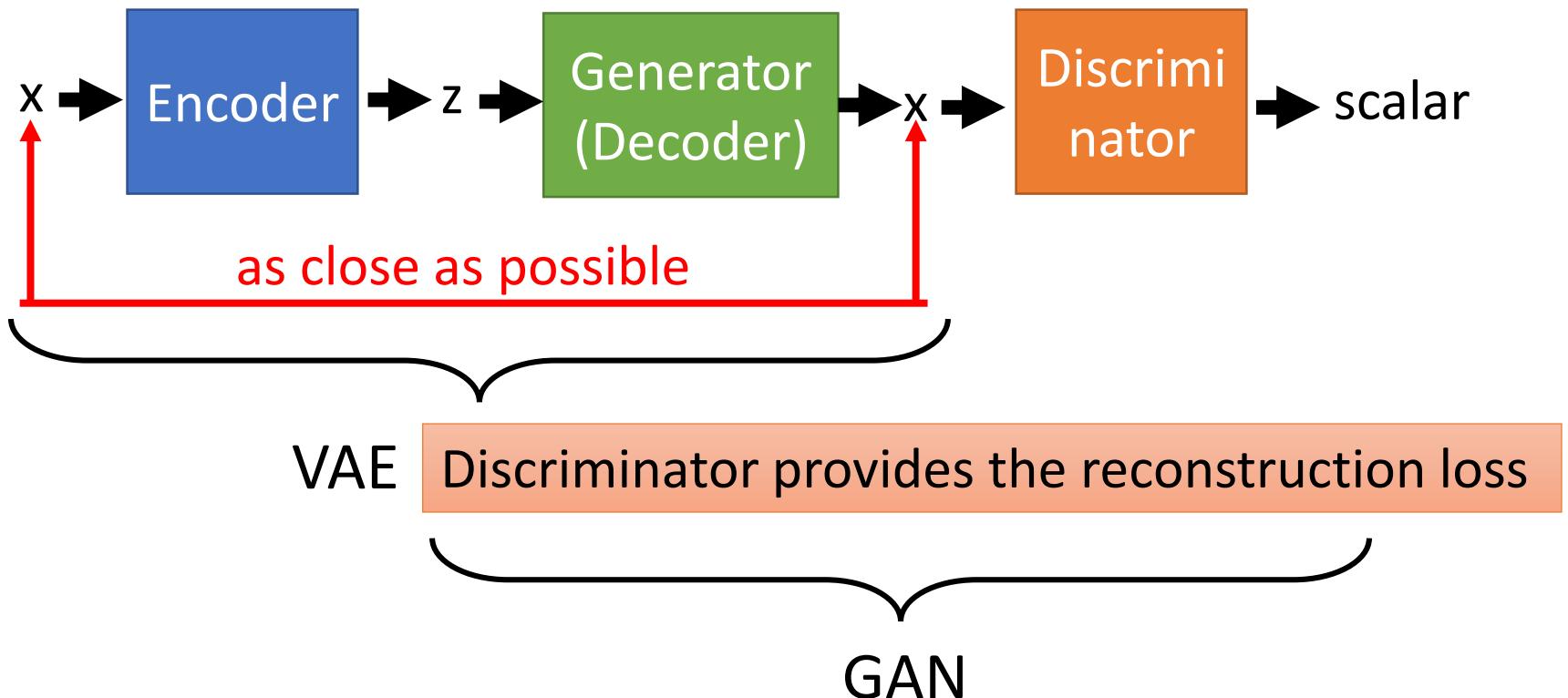


(d) Varying c_3 from -2 to 2 on InfoGAN (Width)

VAE-GAN

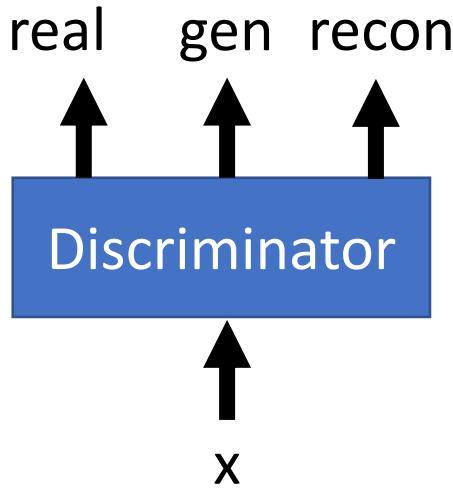
Anders Boesen, Lindbo Larsen, Søren Kaae
Sønderby, Hugo Larochelle, Ole Winther, "Autoencoding
beyond pixels using a learned similarity metric", ICML. 2016

- Minimize reconstruction error
- z close to normal
- Minimize reconstruction error
- Cheat discriminator
- Discriminate real, generated and reconstructed images



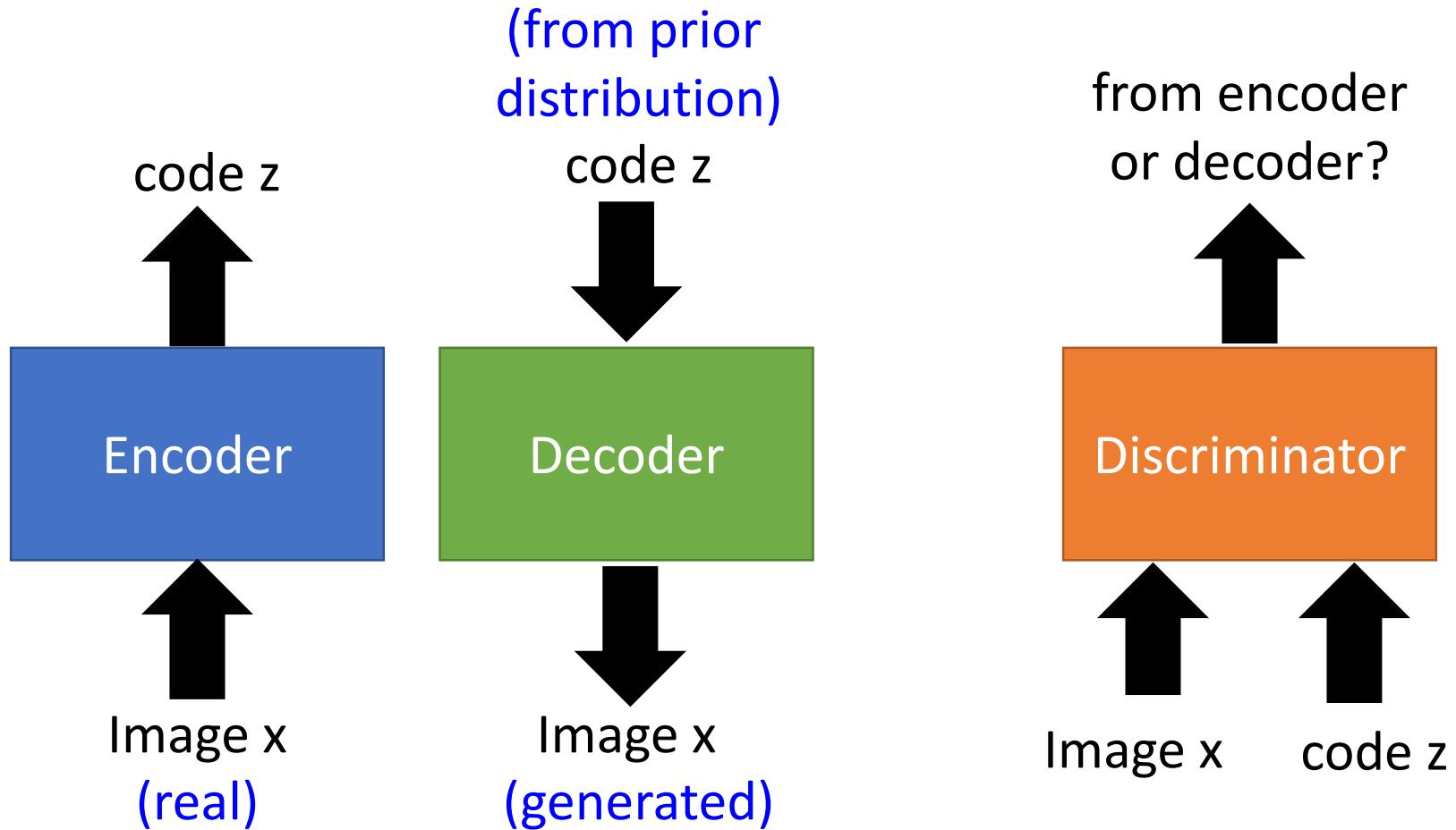
Algorithm

Another kind of discriminator:



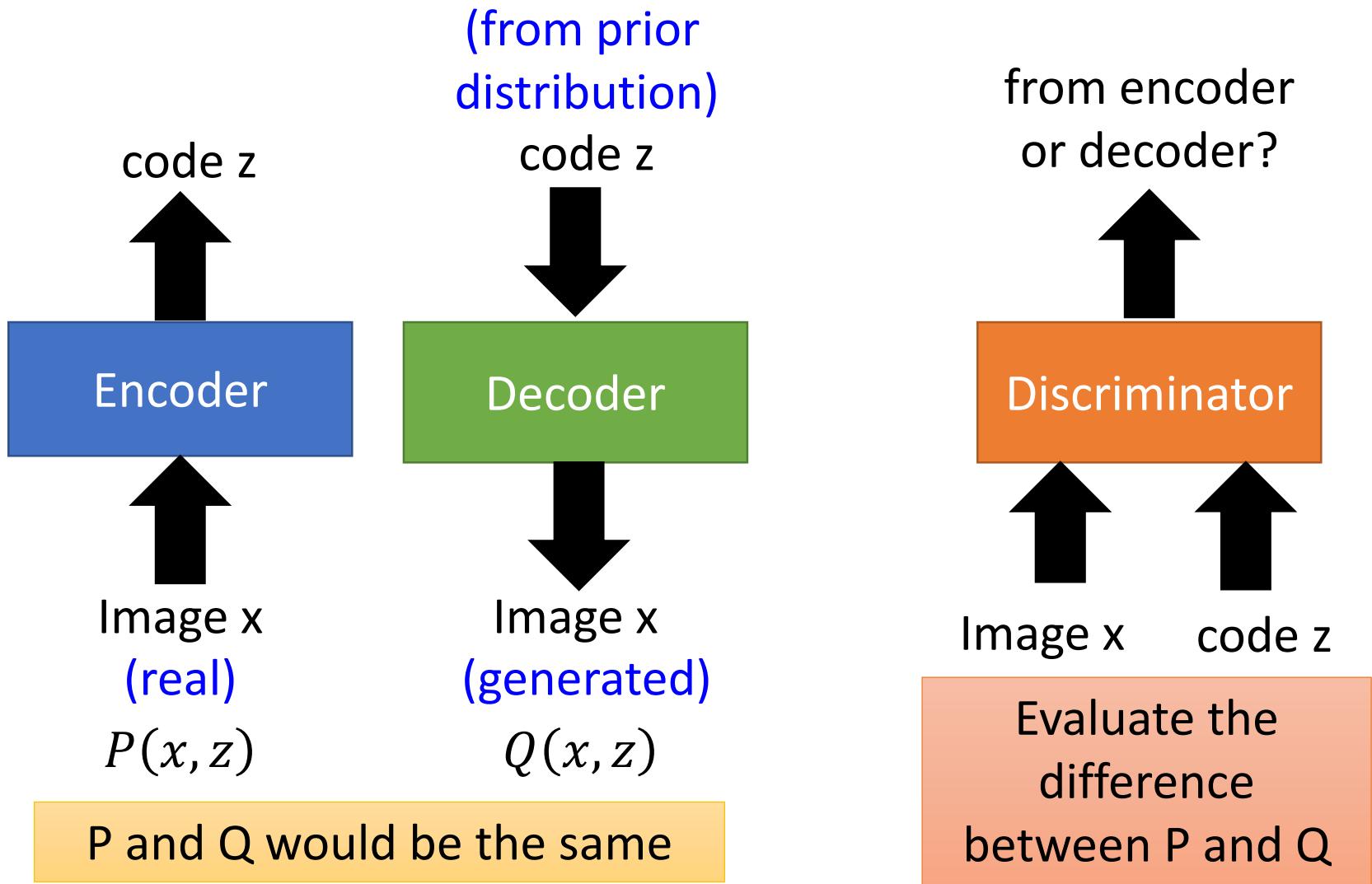
- Initialize En, De, Dis
- In each iteration:
 - Sample M images x^1, x^2, \dots, x^M from database
 - Generate M codes $\tilde{z}^1, \tilde{z}^2, \dots, \tilde{z}^M$ from encoder
 - $\tilde{z}^i = En(x^i)$
 - Generate M images $\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^M$ from decoder
 - $\tilde{x}^i = De(\tilde{z}^i)$
 - Sample M codes z^1, z^2, \dots, z^M from prior P(z)
 - Generate M images $\hat{x}^1, \hat{x}^2, \dots, \hat{x}^M$ from decoder
 - $\hat{x}^i = De(z^i)$
 - Update En to decrease $\|\tilde{x}^i - x^i\|$, decrease $KL(P(\tilde{z}^i | x^i) || P(z))$
 - Update De to decrease $\|\tilde{x}^i - x^i\|$, increase $Dis(\tilde{x}^i)$ and $Dis(\hat{x}^i)$
 - Update Dis to increase $Dis(x^i)$, decrease $Dis(\tilde{x}^i)$ and $Dis(\hat{x}^i)$

BiGAN



Algorithm

- Initialize encoder En, decoder De, discriminator Dis
- In each iteration:
 - Sample M images x^1, x^2, \dots, x^M from database
 - Generate M codes $\tilde{z}^1, \tilde{z}^2, \dots, \tilde{z}^M$ from encoder
 - $\tilde{z}^i = En(x^i)$
 - Sample M codes z^1, z^2, \dots, z^M from prior $P(z)$
 - Generate M codes $\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^M$ from decoder
 - $\tilde{x}^i = De(z^i)$
 - Update Dis to increase $Dis(x^i, \tilde{z}^i)$, decrease $Dis(\tilde{x}^i, z^i)$
 - Update En and De to decrease $Dis(x^i, \tilde{z}^i)$, increase $Dis(\tilde{x}^i, z^i)$

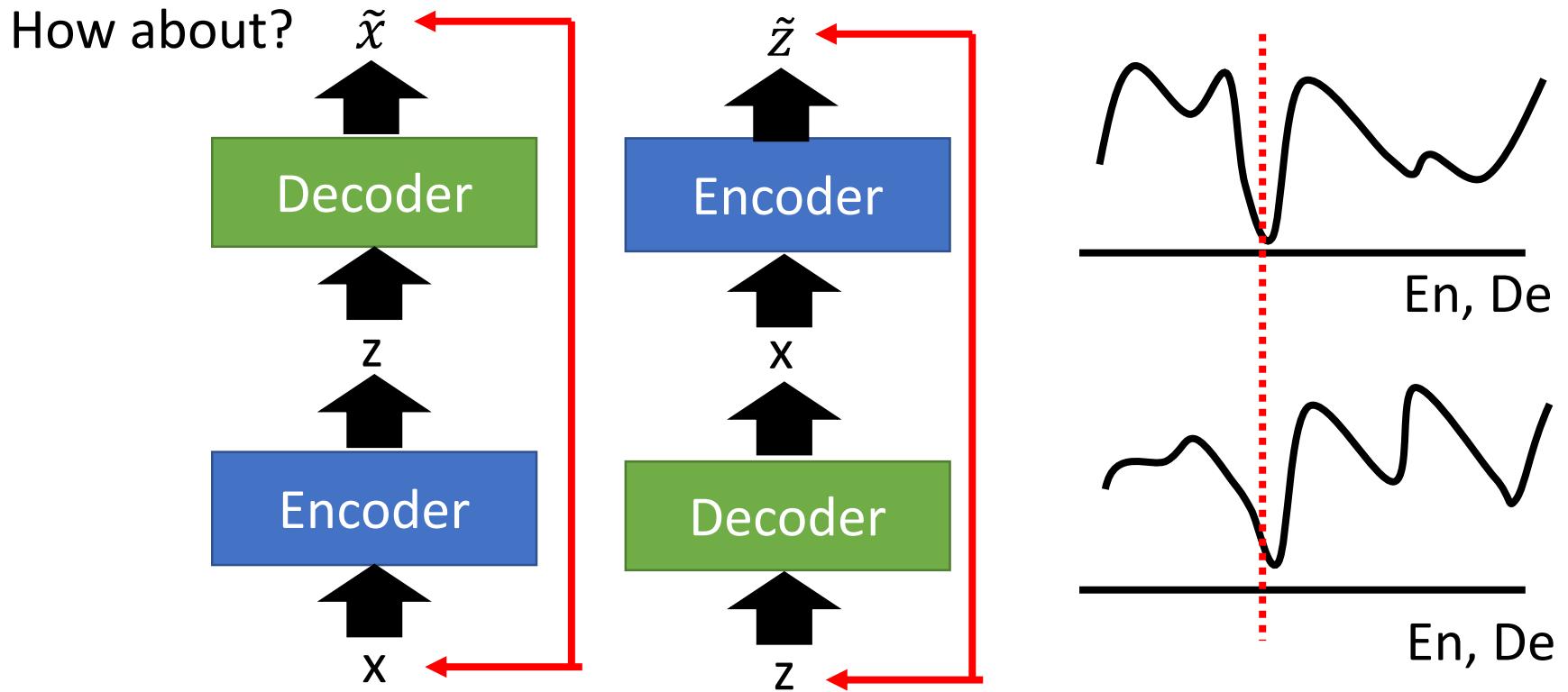


Optimal encoder
and decoder:

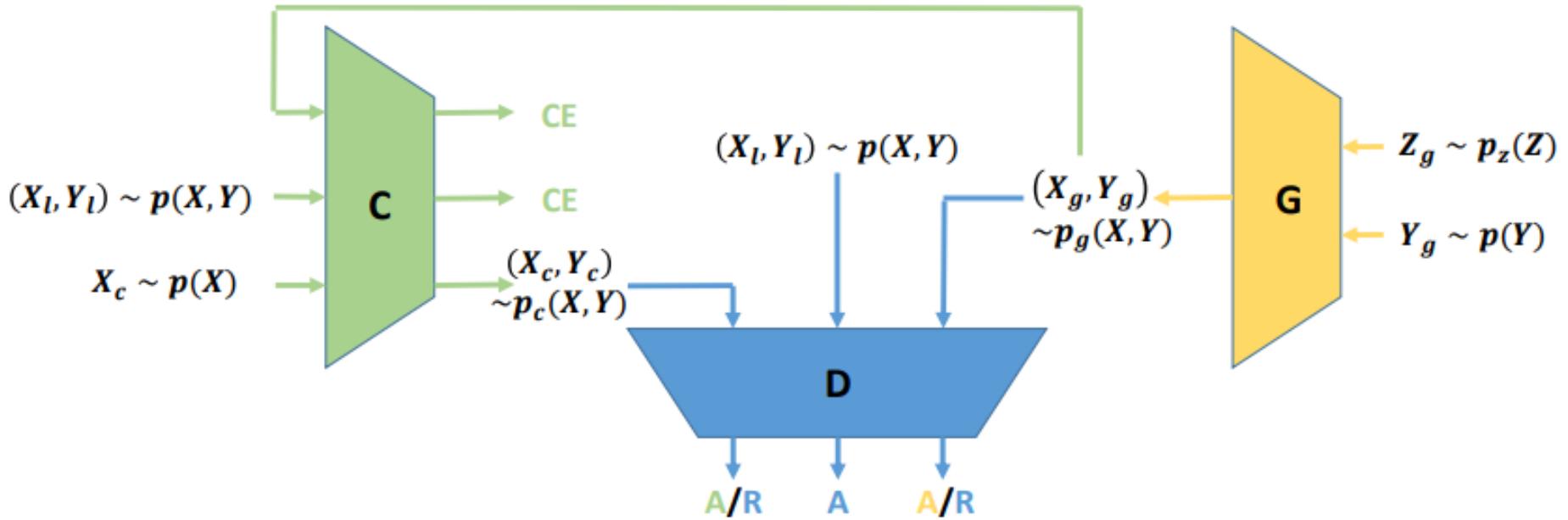
$$\begin{aligned}
 \text{En}(x') &= z' & \xrightarrow{\quad} \text{De}(z') &= x' & \text{For all } x' \\
 \text{De}(z'') &= x'' & \xrightarrow{\quad} \text{En}(x'') &= z'' & \text{For all } z''
 \end{aligned}$$

BiGAN

Optimal encoder
and decoder:

$$\text{En}(x') = z' \rightarrow \text{De}(z') = x' \quad \text{For all } x'$$
$$\text{De}(z'') = x'' \rightarrow \text{En}(x'') = z'' \quad \text{For all } z''$$


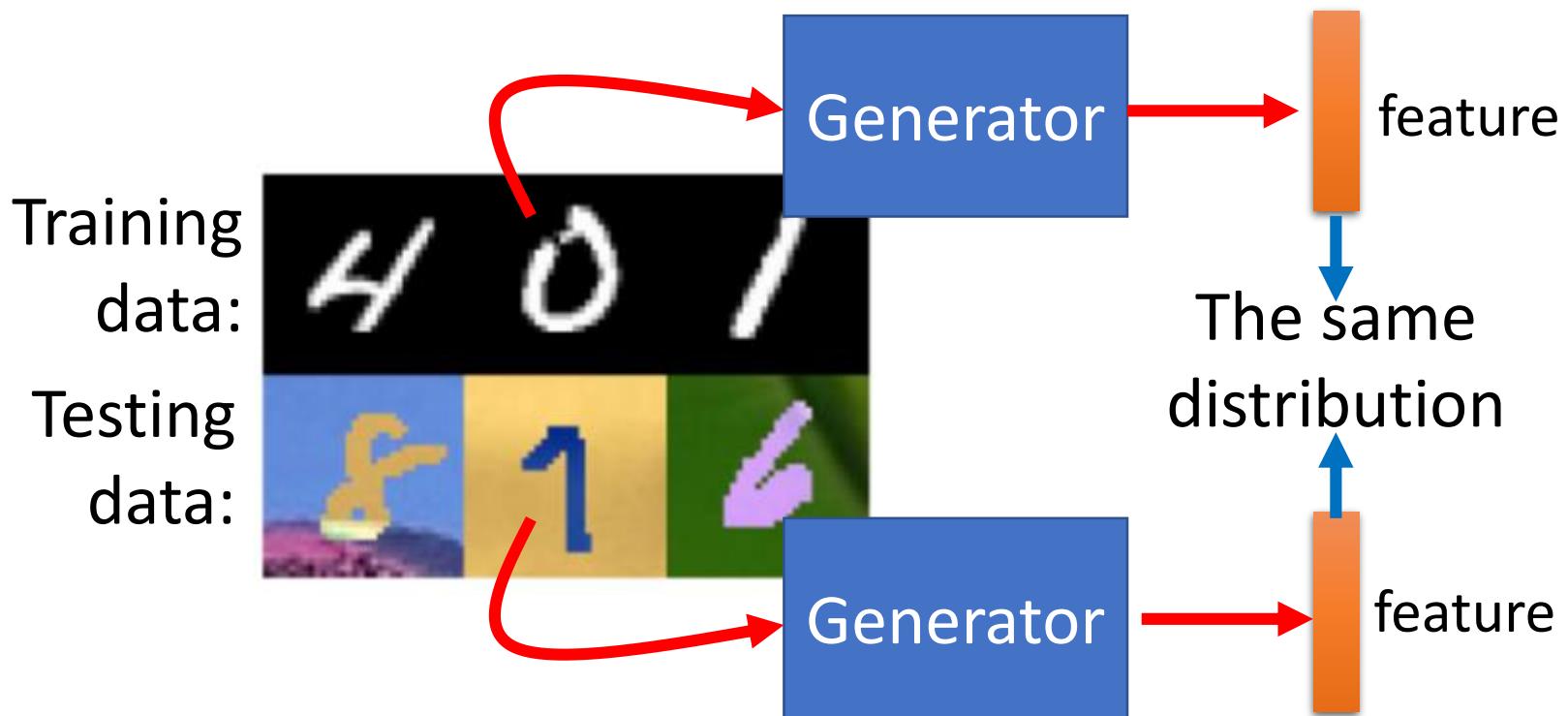
Triple GAN



Chongxuan Li, Kun Xu, Jun Zhu, Bo Zhang, "Triple Generative Adversarial Nets", arXiv 2017

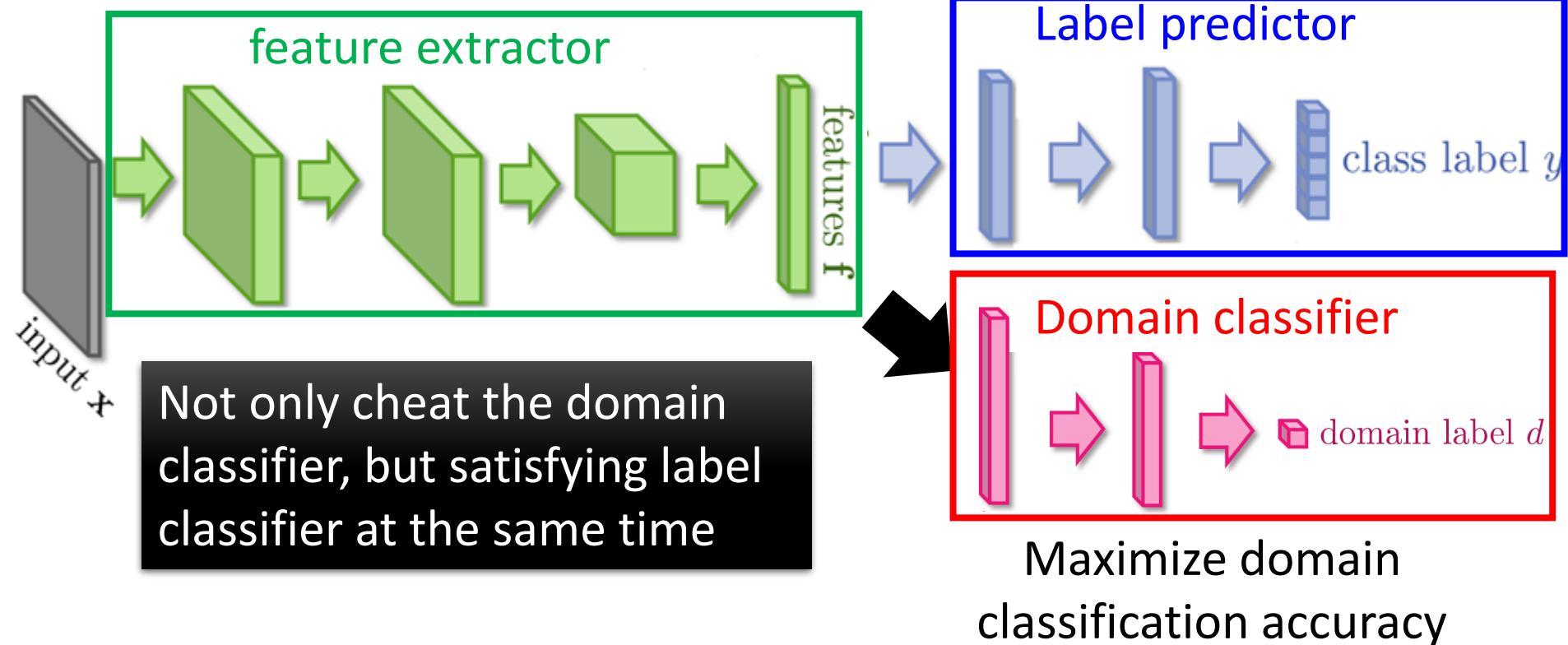
Domain-adversarial training

- Training and testing data are in different domains



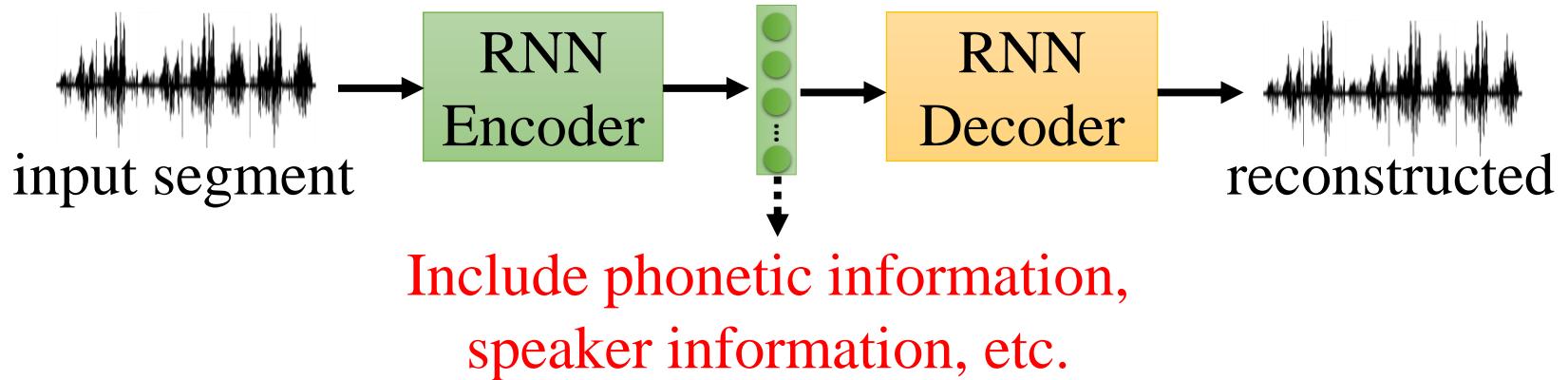
Domain-adversarial training

Maximize label classification accuracy +
minimize domain classification accuracy

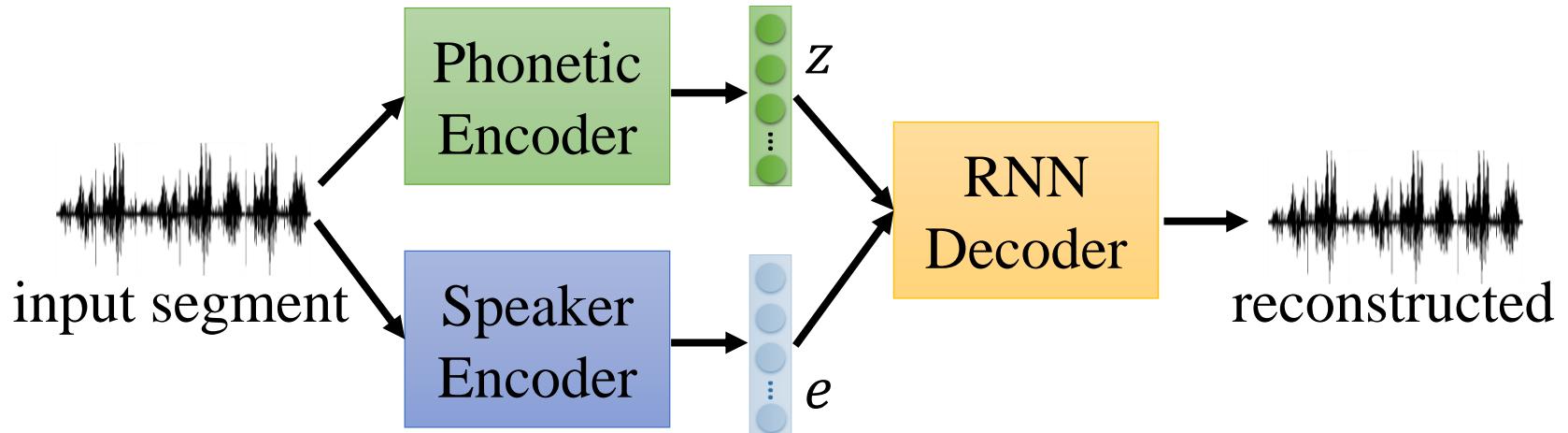


This is a big network, but different parts have different goals.

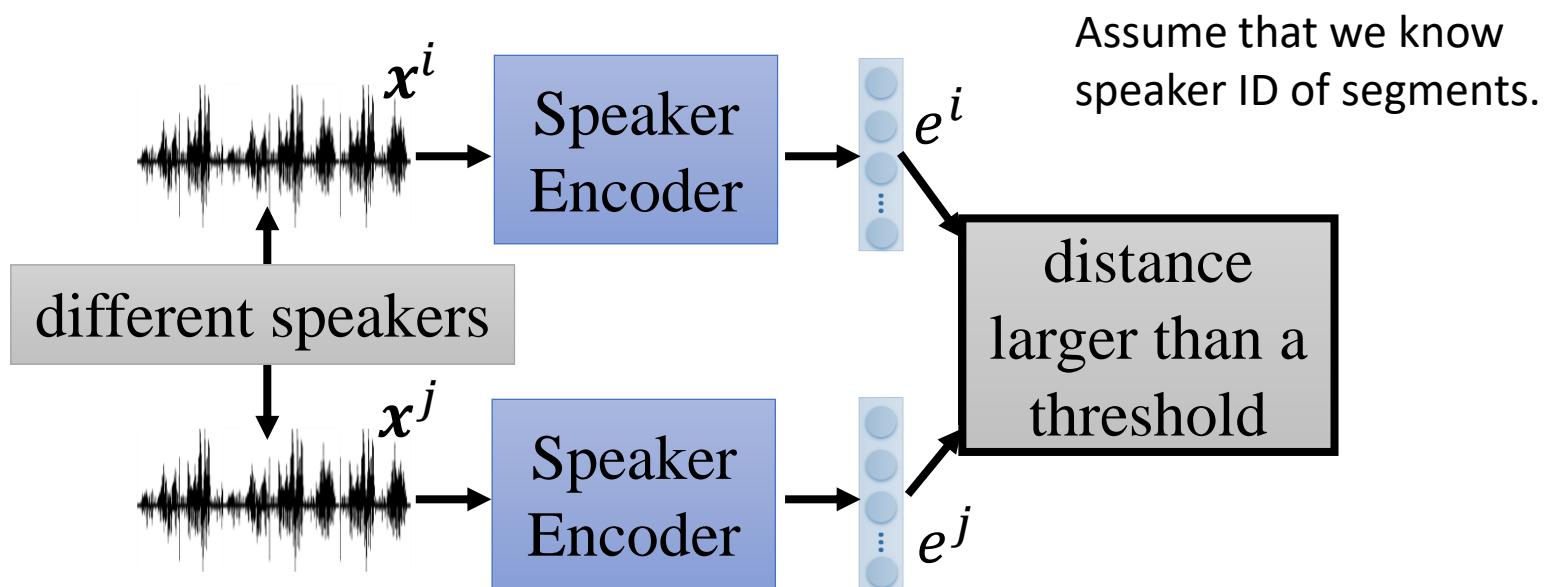
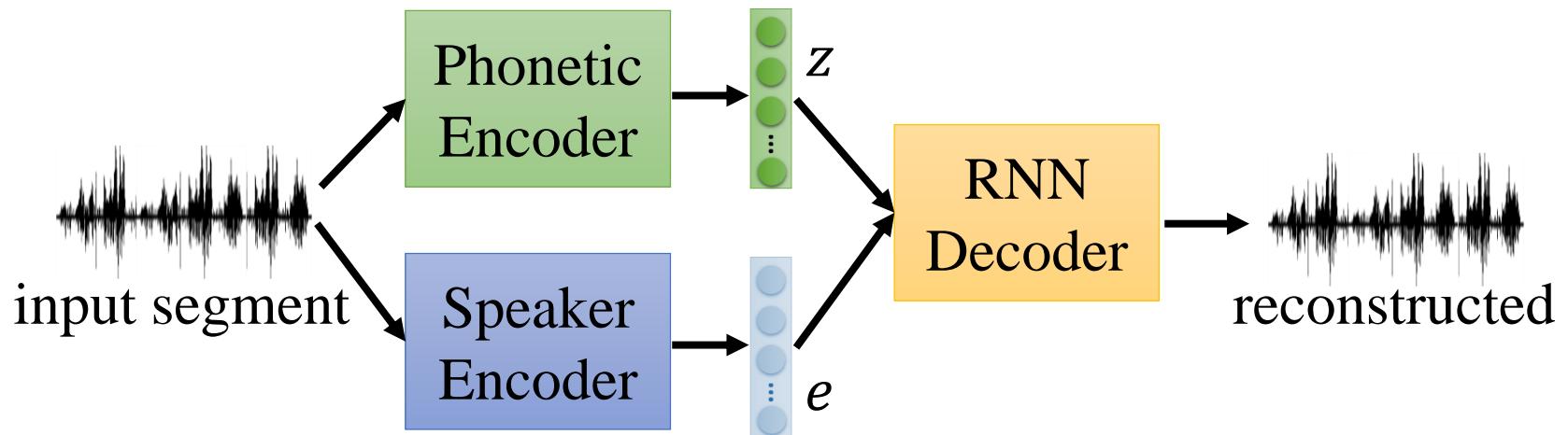
Original Seq2seq Auto-encoder



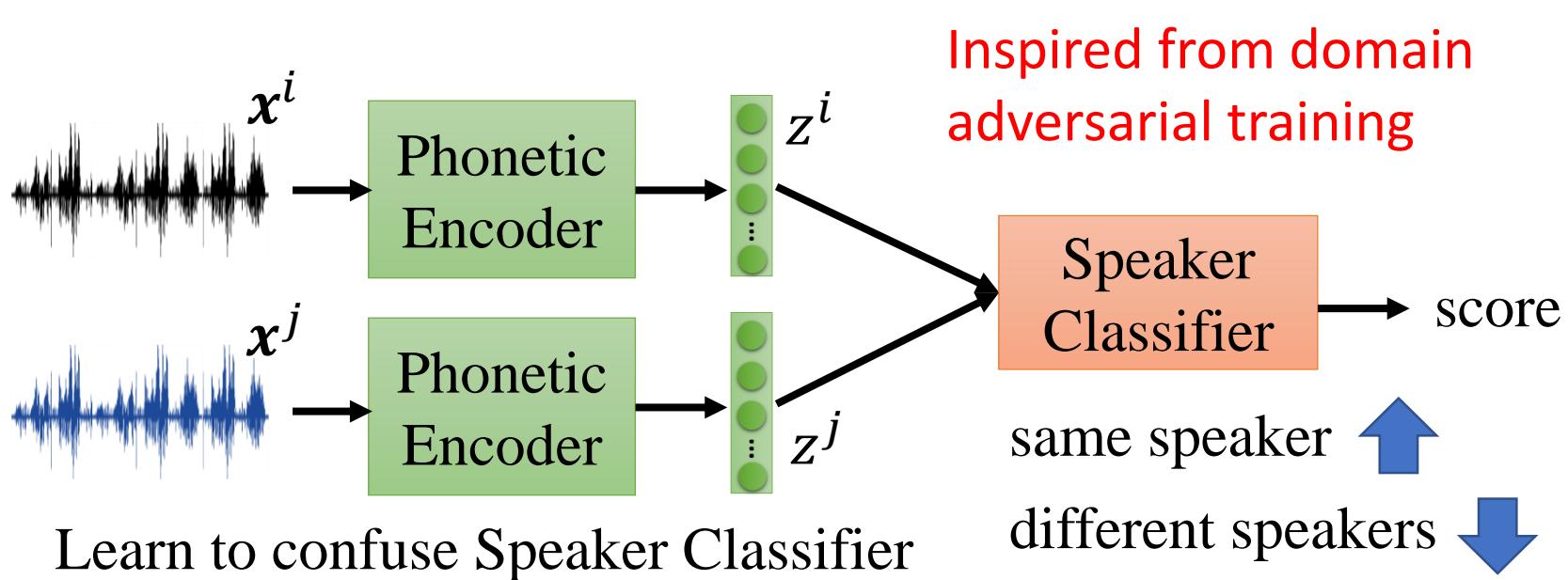
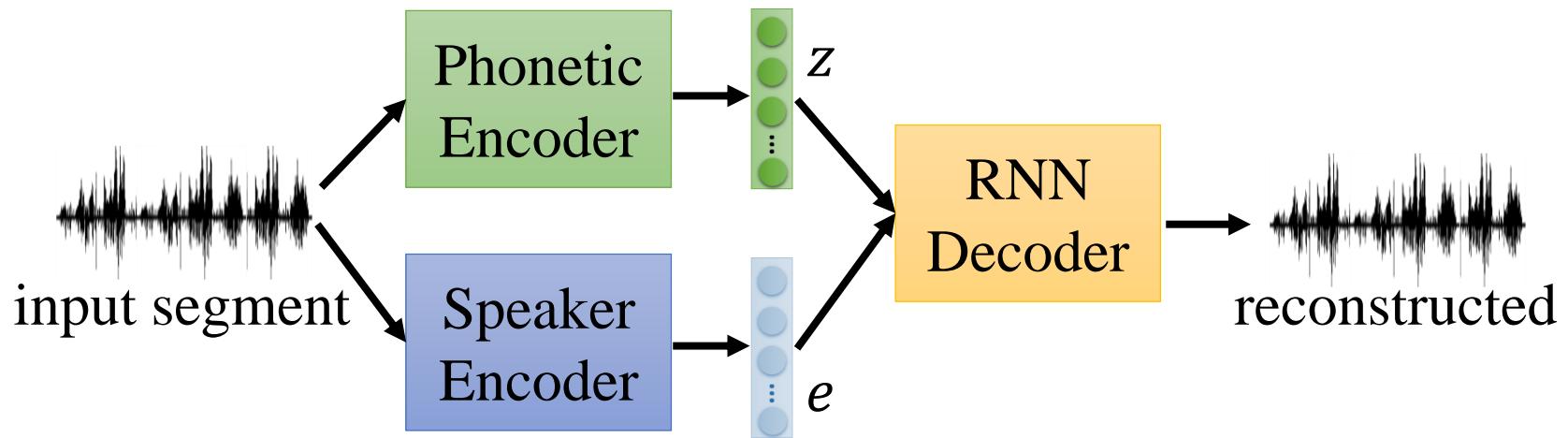
Feature Disentangle

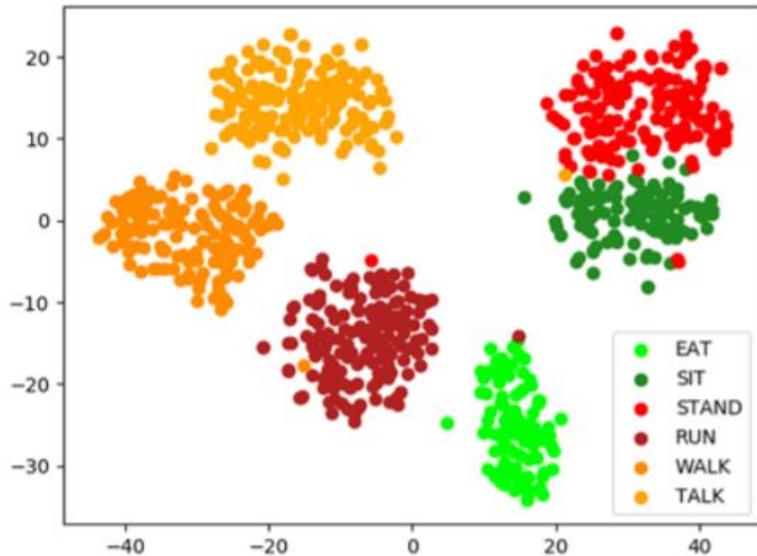


Feature Disentangle

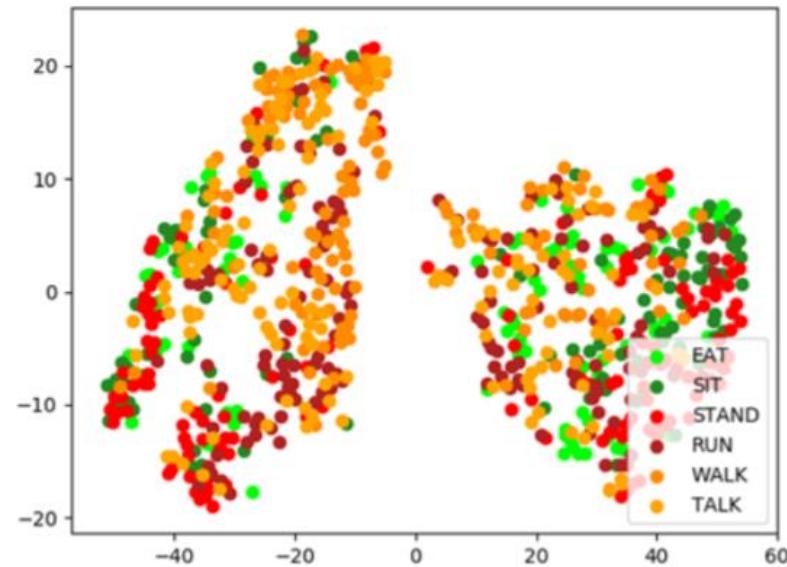


Feature Disentangle

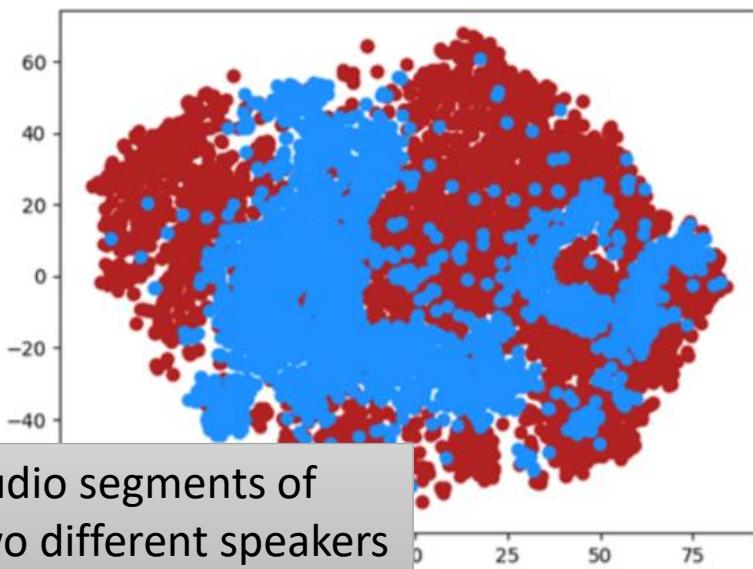




phonetic embedding z

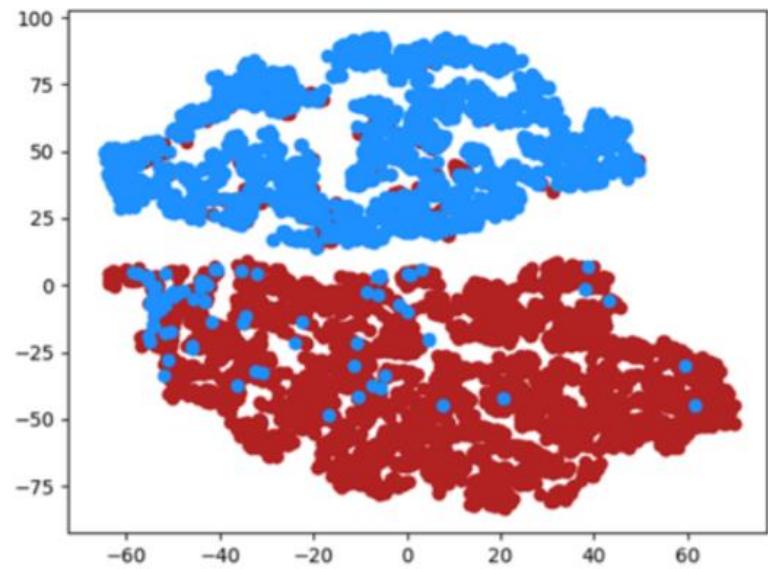


speaker embedding e



Audio segments of
two different speakers

phonetic embedding z



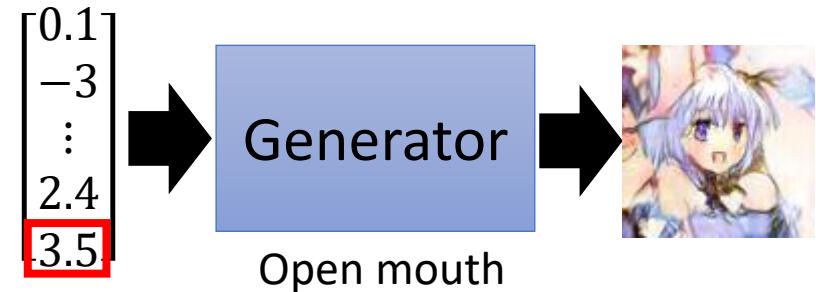
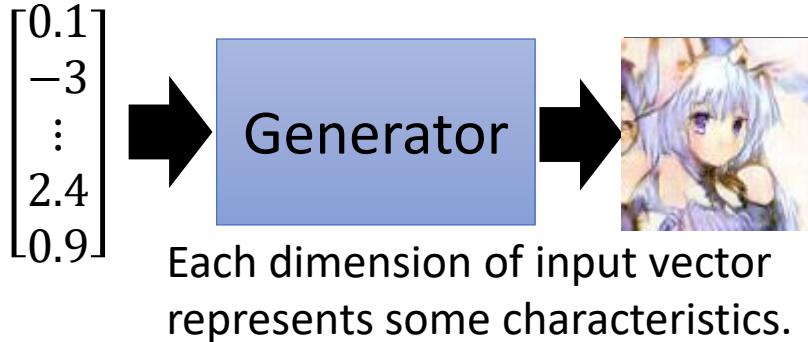
speaker embedding e

Acknowledgement

- 感謝 許傑盛 同學發現投影片上的打字錯誤

Intelligent Photo Editing

Modifying Input Code



- The input code determines the generator output.
- Understand the meaning of each dimension to control the output.

Connecting Code and Attribute



(c) Hair style

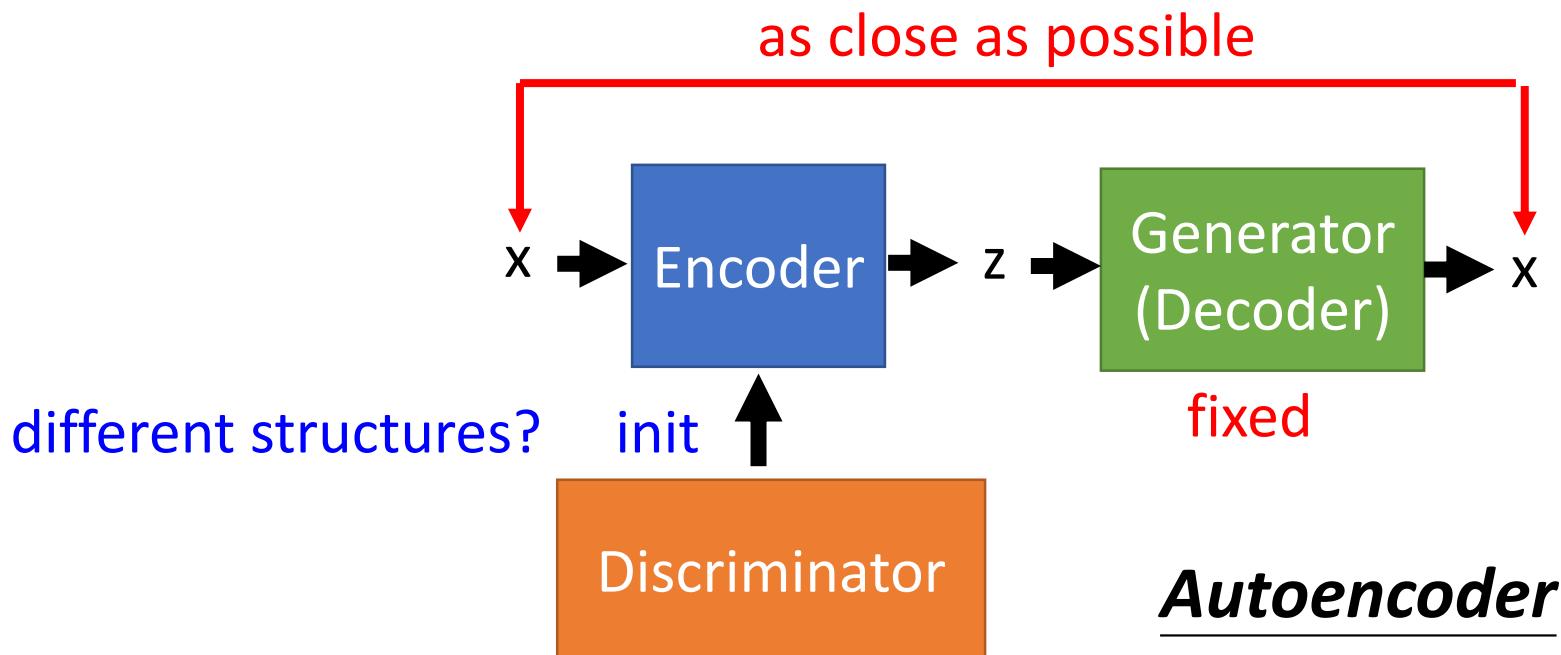
(d) Emotion

CelebA

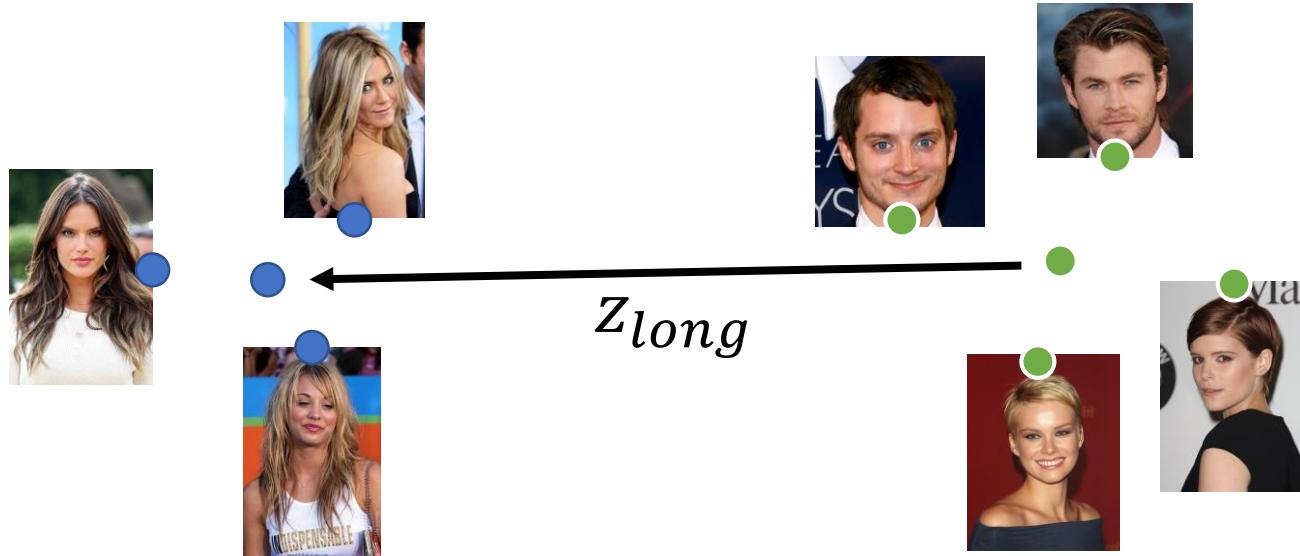
Image	Attributes
	Arched eyebrows, attractive, brown hair, heavy makeup, high cheekbones, mouth slightly open, no beard, pointy nose, smiling, straight hair, wearing earrings, wearing lipstick, young.
	5 o'clock shadows, attractive, bags under eyes, big lips, big nose, black hair, bushy eyebrows, male, no beard, pointy nose, straight hair, young.

GAN+Autoencoder

- We have a generator (input z , output x)
- However, given x , how can we find z ?
 - Learn an encoder (input x , output z)



Attribute Representation



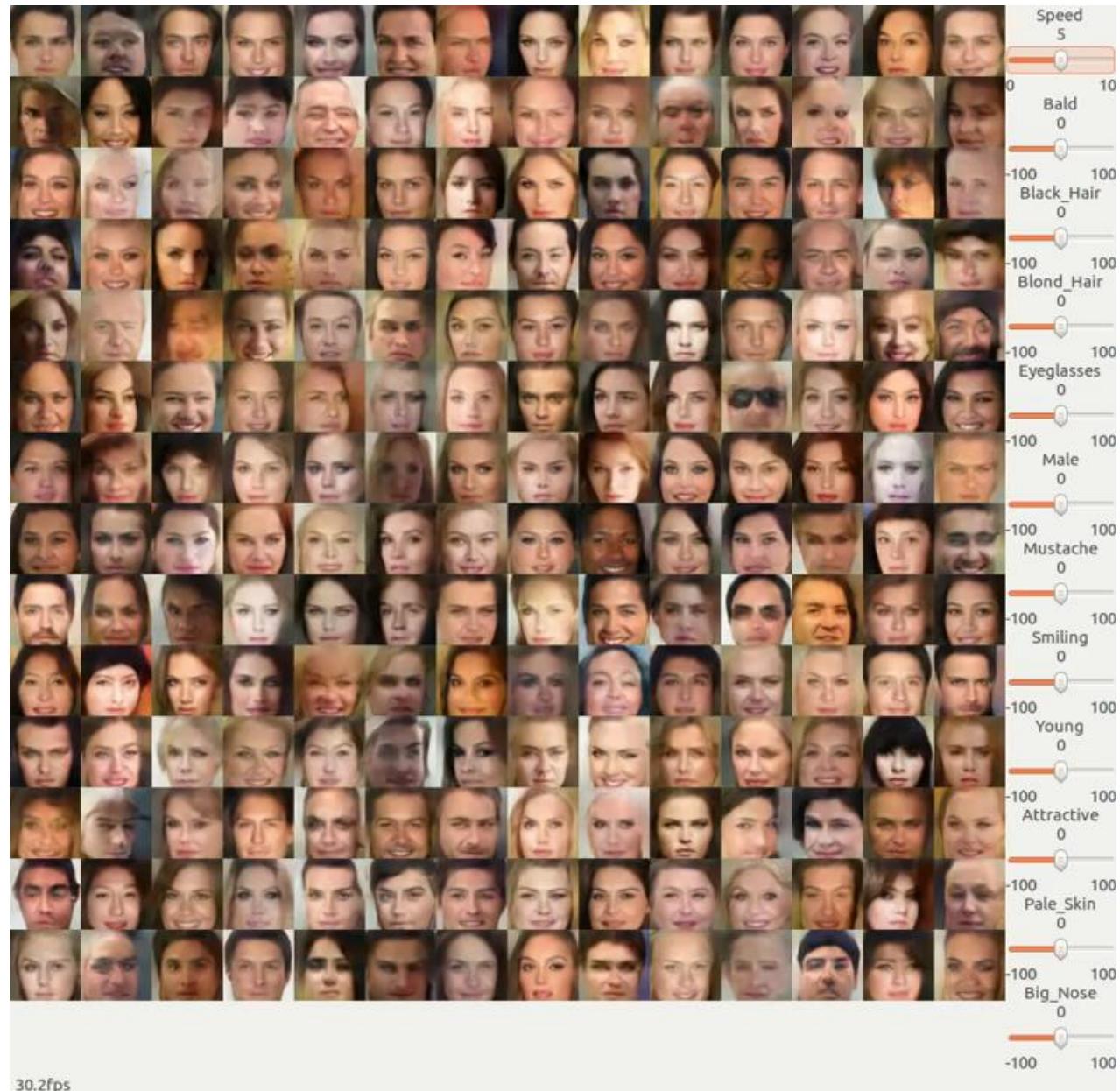
$$z_{long} = \frac{1}{N_1} \sum_{x \in long} En(x) - \frac{1}{N_2} \sum_{x' \notin long} En(x')$$

Short
Hair

$$x \rightarrow En(x) + z_{long} = z' \rightarrow Gen(z')$$

Long
Hair

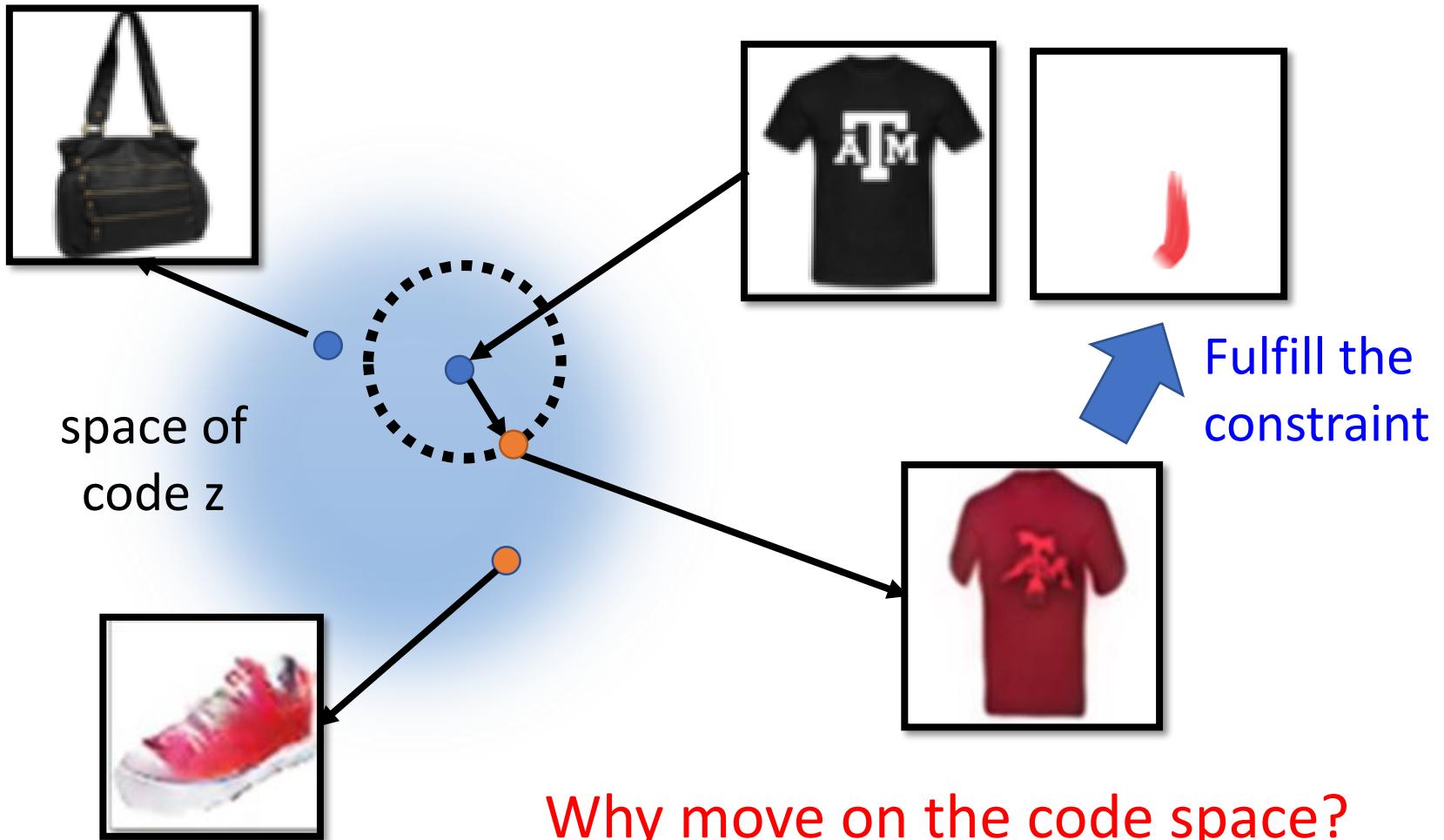
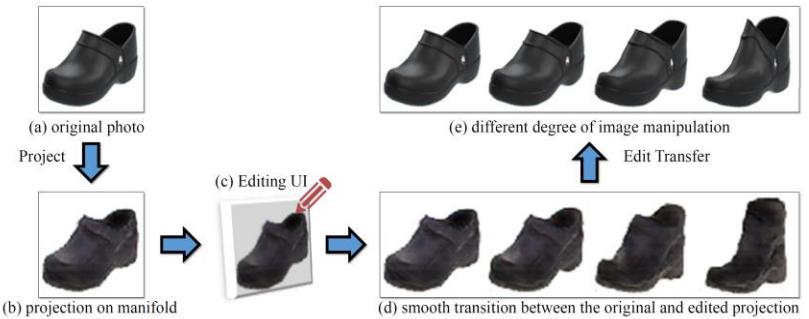
Photo Editing



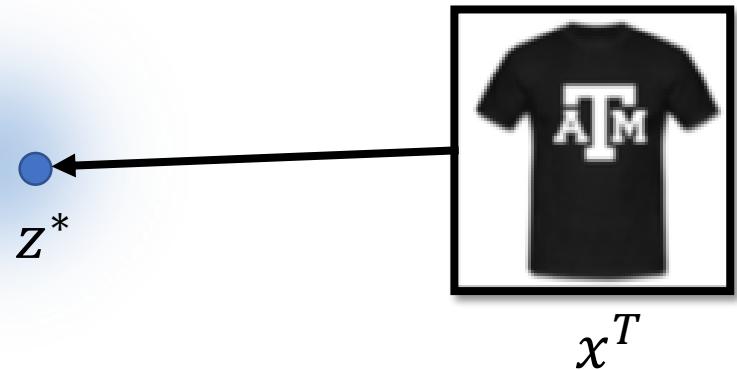
30.2fps

<https://www.youtube.com/watch?v=kPEIJJsQr7U>

Basic Idea



Back to z



- **Method 1**

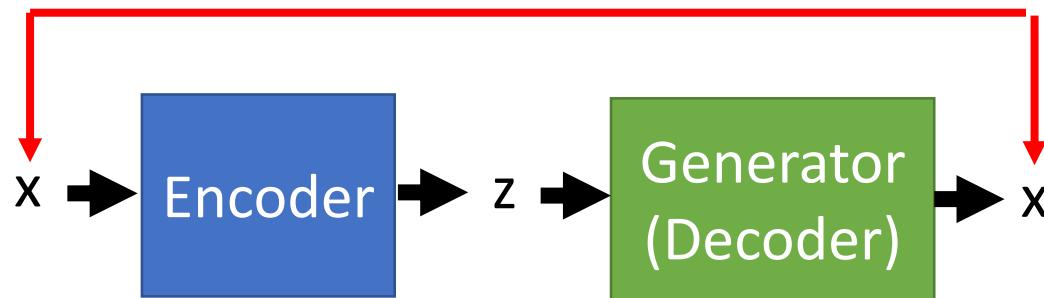
$$z^* = \arg \min_z L(G(z), x^T)$$

Gradient Descent

→ Difference between $G(z)$ and x^T
➤ Pixel-wise
➤ By another network

- **Method 2**

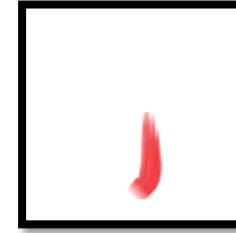
as close as possible



- **Method 3**

Using the results from **method 2** as the initialization of **method 1**

Editing Photos



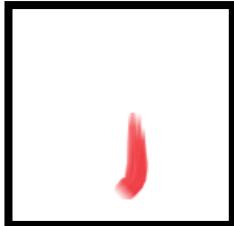
- z_0 is the code of the input image



Using discriminator
to check the image is
realistic or not

$$z^* = \arg \min_z U(G(z)) + \lambda_1 \|z - z_0\|^2 - \lambda_2 D(G(z))$$

Not too far away from
the original image



Does it fulfill the constraint of editing?

Generative Visual Manipulation on the Natural Image Manifold

Jun-Yan Zhu

Philipp Krähenbühl

Eli Shechtman

Alexei A. Efros



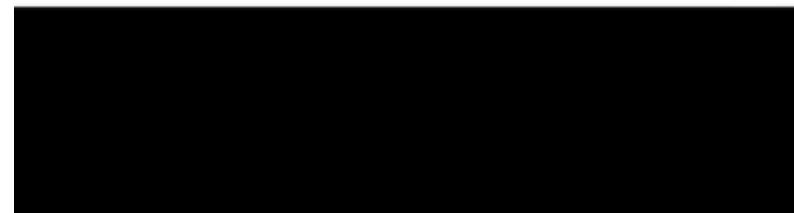
<https://www.youtube.com/watch?v=9c4z6YsBGQ0>

Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman and Alexei A. Efros. "Generative Visual Manipulation on the Natural Image Manifold", ECCV, 2016.



Neural Photo Editing

Andrew Brock



Andrew Brock, Theodore Lim, J.M. Ritchie, Nick Weston, **Neural Photo Editing with Introspective Adversarial Networks**, arXiv preprint, 2017

Image super resolution

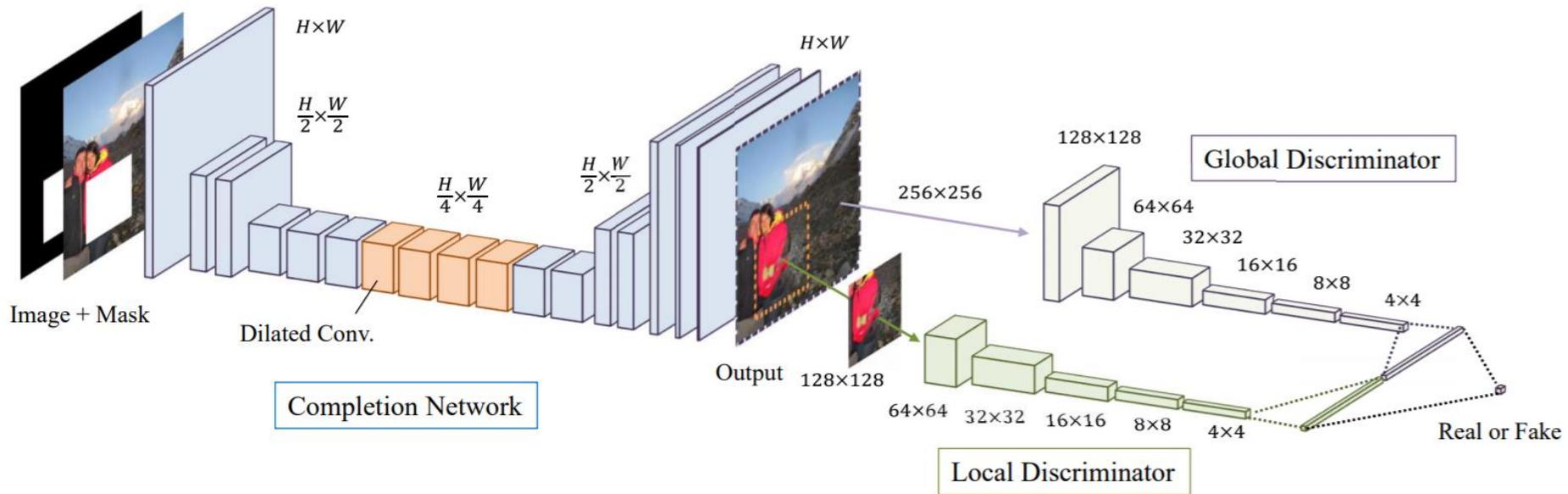
- Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, Wenzhe Shi, “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network”, CVPR, 2016



Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

Image Completion

<http://hi.cs.waseda.ac.jp/~iizuka/projects/completion/en/>



Demo

Image completion is a very complicated task...



Previous approach



Previous approach

<https://www.youtube.com/watch?v=5Ua4NUKowPU>

Improving Sequence Generation by GAN

李宏毅

Hung-yi Lee

Outline

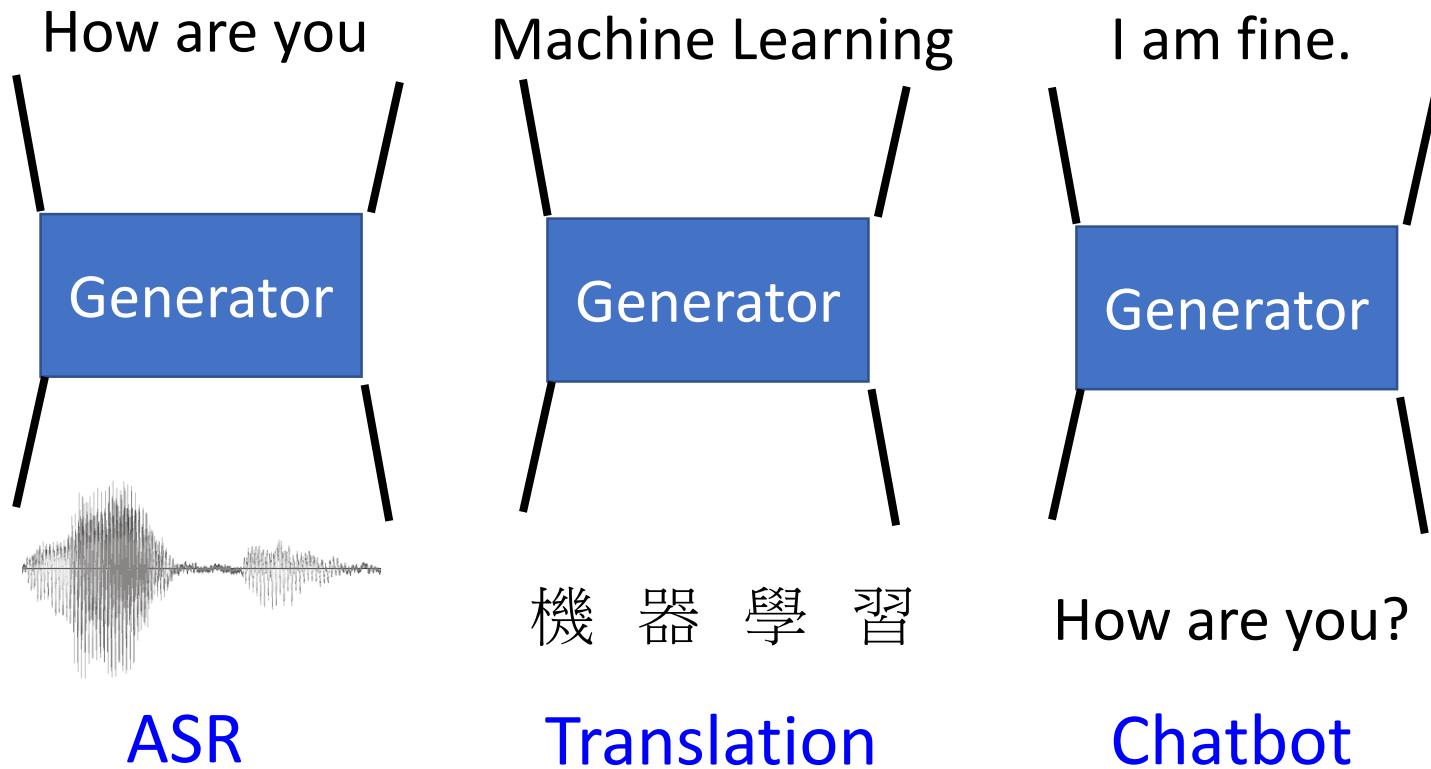
Conditional Sequence Generation

- RL (human feedback)
- GAN (discriminator feedback)

Unsupervised Conditional Sequence Generation

- Text Style Transfer
- Unsupervised Abstractive Summarization
- Unsupervised Translation

Conditional Sequence Generation



The generator is a typical seq2seq model.

With GAN, you can train seq2seq model in another way.

Review: Sequence-to-sequence

- Chat-bot as example

Output:	Not bad	I'm John.
Human	better	
Training Criterion		better

Training
data:
⋮

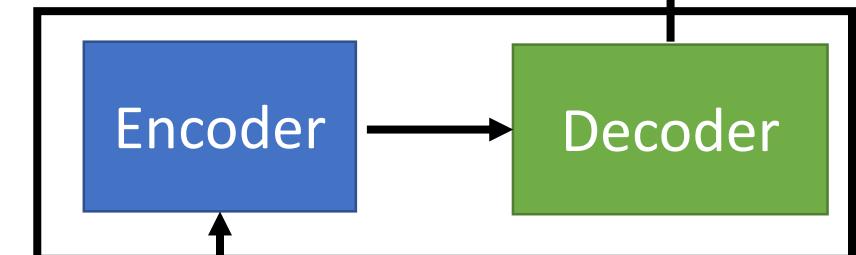
A: How are you ?

B: I'm good.
⋮

Maximize
likelihood

I'm good.

output
sentence x



Input sentence c
How are you ?

Outline of Part III

Improving Supervised Seq-to-seq Model

- RL (human feedback)
- GAN (discriminator feedback)

Unsupervised Seq-to-seq Model

- Text Style Transfer
- Unsupervised Abstractive Summarization
- Unsupervised Translation

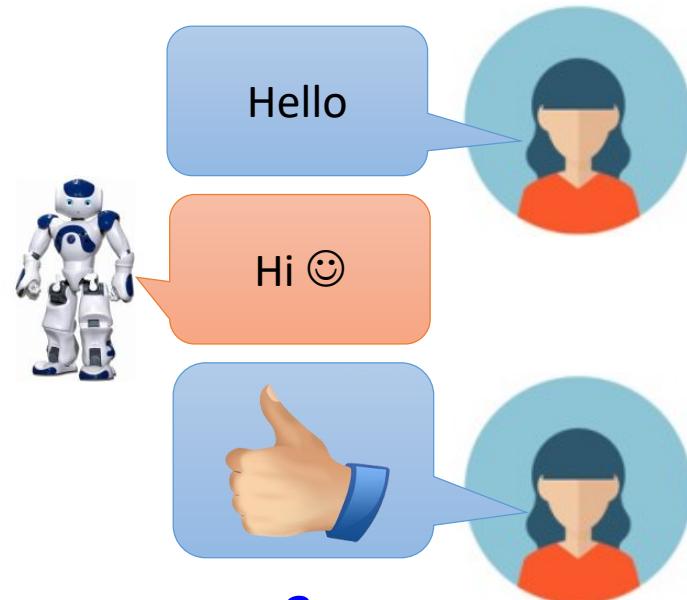
Introduction

- Machine obtains feedback from user



-10

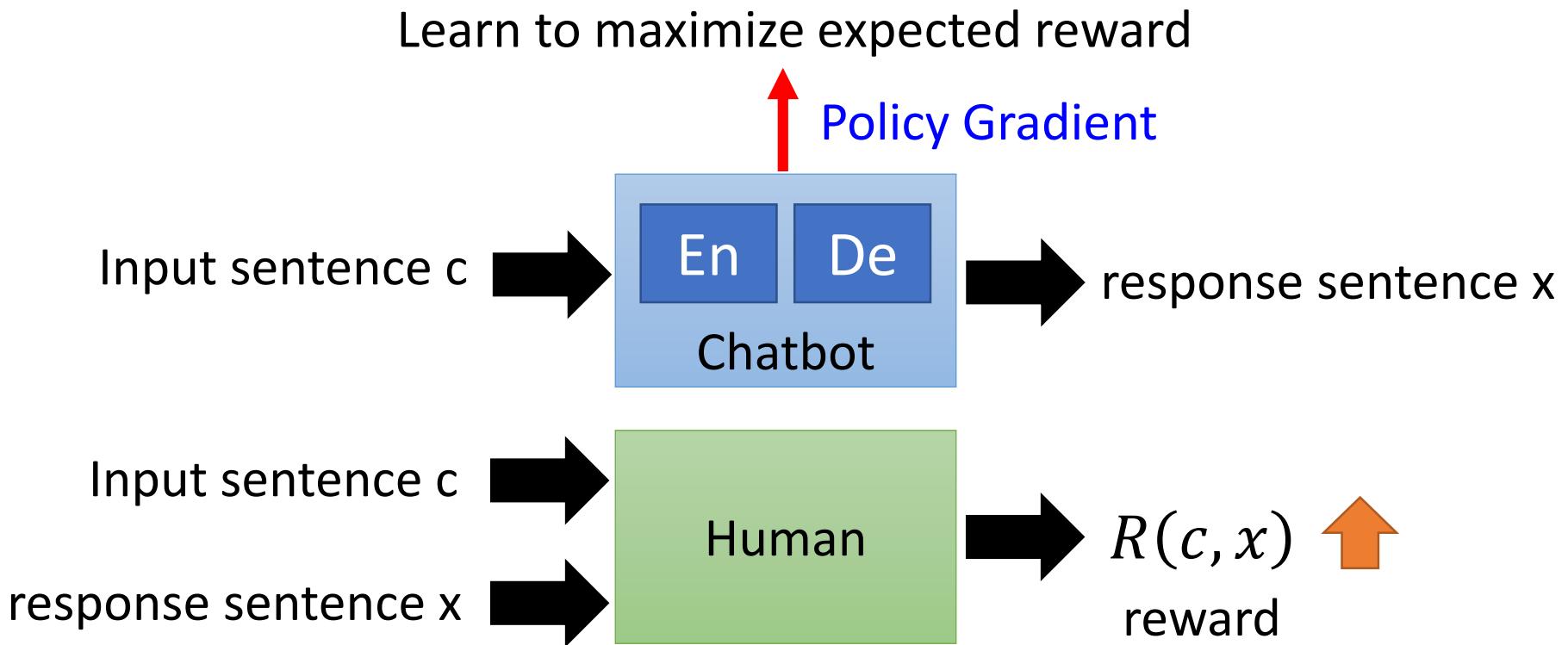
https://image.freepik.com/free-vector/variety-of-human-avatars_23-2147506285.jpg
http://www.freepik.com/free-vector/variety-of-human-avatars_766615.htm



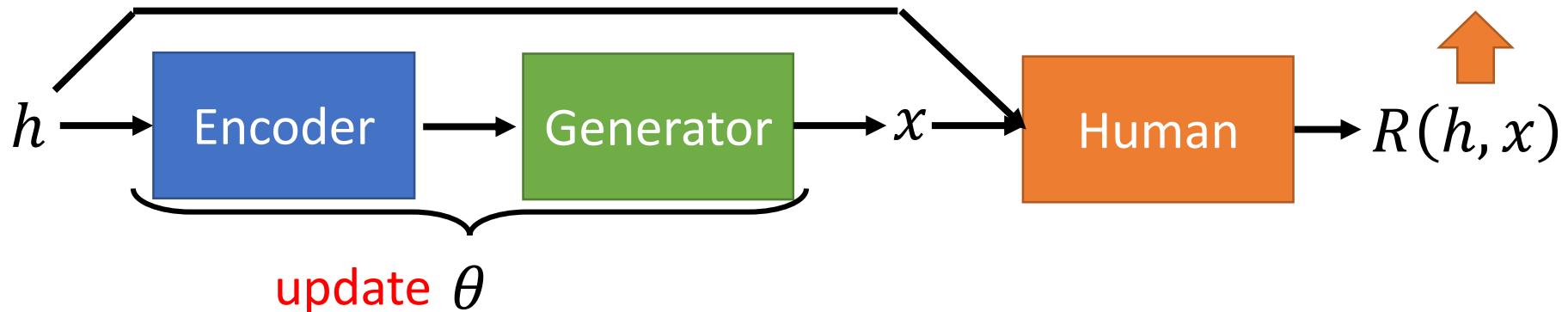
3

- Chat-bot learns to maximize the *expected reward*

Maximizing Expected Reward



Maximizing Expected Reward

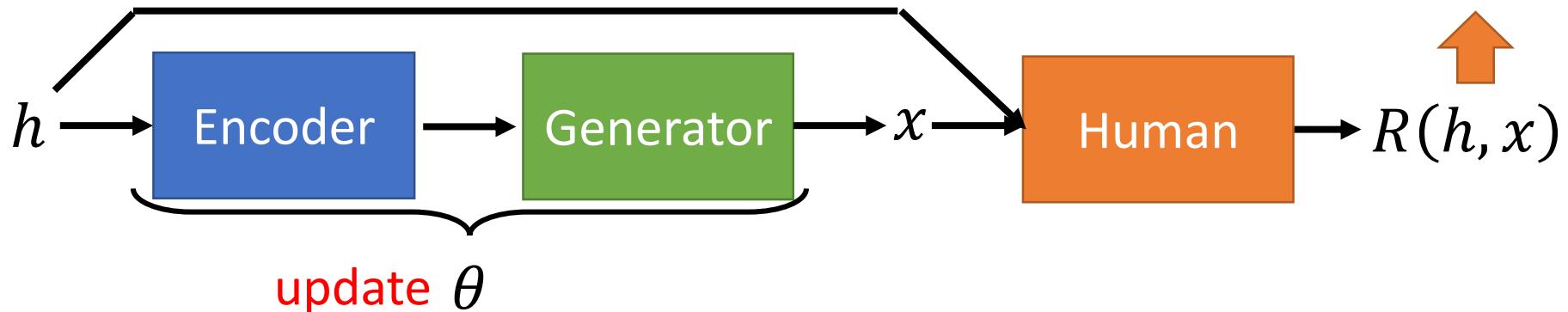


$$\theta^* = \arg \max_{\theta} \bar{R}_{\theta} \quad \leftarrow \text{Maximizing expected reward}$$

$$\bar{R}_{\theta} = \sum_h P(h) \sum_x R(h, x) \frac{P_{\theta}(x|h)}{\text{Randomness in generator}}$$

Probability that the input/history is h

Maximizing Expected Reward



$$\theta^* = \arg \max_{\theta} \bar{R}_{\theta} \quad \leftarrow \text{Maximizing expected reward}$$

$$\begin{aligned}\bar{R}_{\theta} &= \sum_h P(h) \sum_x R(h, x) P_{\theta}(x|h) = E_{h \sim P(h)} \left[E_{x \sim P_{\theta}(x|h)} [R(h, x)] \right] \\ &= E_{h \sim P(h), x \sim P_{\theta}(x|h)} [R(h, x)] \approx \frac{1}{N} \sum_{i=1}^N R(h^i, x^i)\end{aligned}$$

Sample: $(h^1, x^1), (h^2, x^2), \dots, (h^N, x^N)$

Where
is θ ?

Policy Gradient

$$\frac{d \log(f(x))}{dx} = \frac{1}{f(x)} \frac{df(x)}{dx}$$

$$\bar{R}_\theta = \sum_h P(h) \sum_x R(h, x) P_\theta(x|h) \approx \frac{1}{N} \sum_{i=1}^N R(h^i, x^i)$$

$$\nabla \bar{R}_\theta = \sum_h P(h) \sum_x R(h, x) \nabla P_\theta(x|h) \approx \frac{1}{N} \sum_{i=1}^N R(h^i, x^i) \nabla \log P_\theta(x|h)$$

$$= \sum_h P(h) \sum_x R(h, x) P_\theta(x|h) \frac{\nabla P_\theta(x|h)}{P_\theta(x|h)}$$

$$= \sum_h P(h) \sum_x R(h, x) P_\theta(x|h) \boxed{\nabla \log P_\theta(x|h)}$$

$$= E_{h \sim P(h), x \sim P_\theta(x|h)} [R(h, x) \nabla \log P_\theta(x|h)]$$

Sampling



Policy Gradient

- Gradient Ascent

$$\theta^{new} \leftarrow \theta^{old} + \eta \nabla \bar{R}_{\theta^{old}}$$

$$\nabla \bar{R}_{\theta} \approx \frac{1}{N} \sum_{i=1}^N R(h^i, x^i) \nabla \log P_{\theta}(x^i | h^i)$$

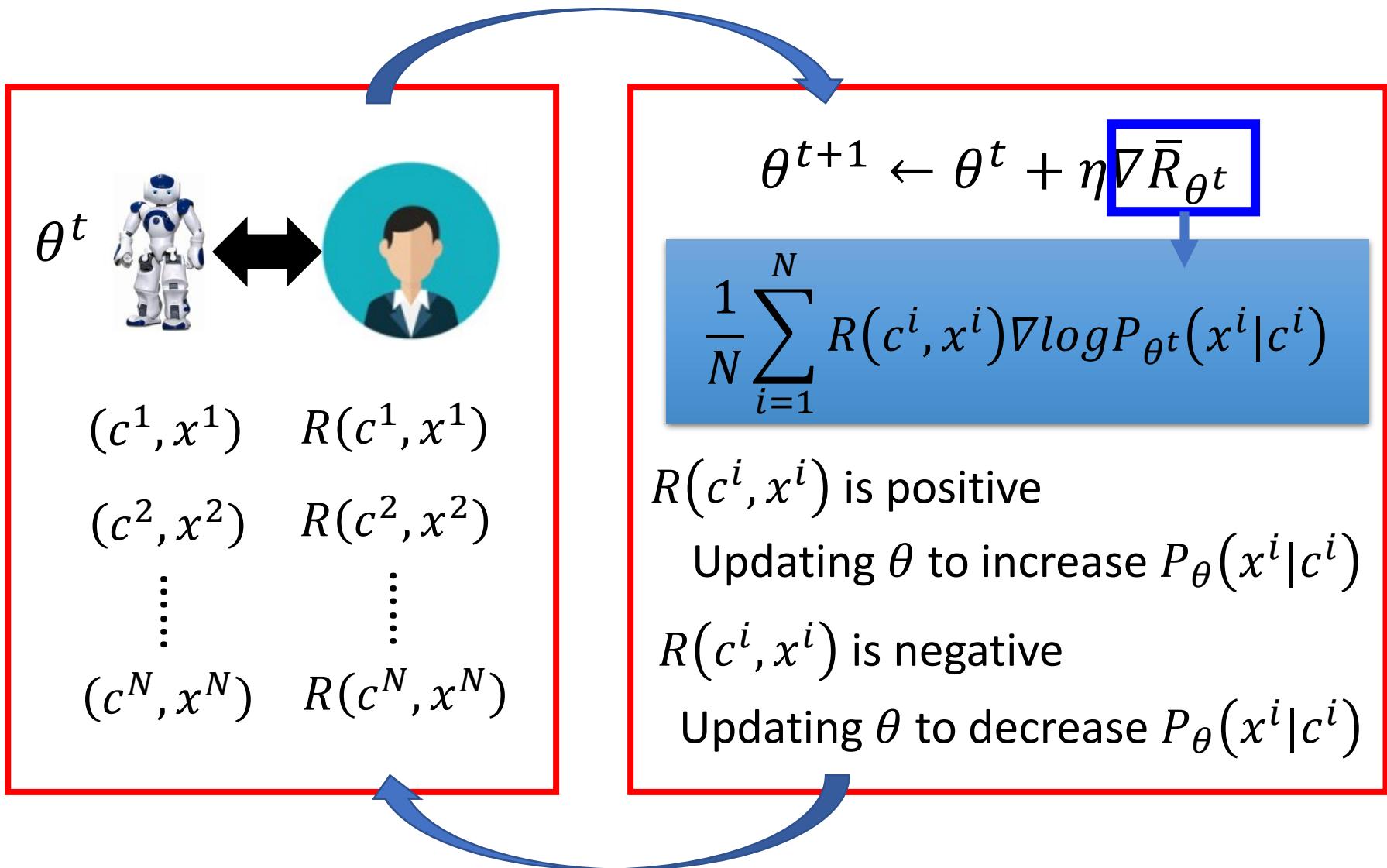
$R(h^i, x^i)$ is positive

→ After updating θ , $P_{\theta}(x^i | h^i)$ will increase

$R(h^i, x^i)$ is negative

→ After updating θ , $P_{\theta}(x^i | h^i)$ will decrease

Policy Gradient - Implementation



Comparison

	Maximum Likelihood	Reinforcement Learning
Objective Function	$\frac{1}{N} \sum_{i=1}^N \log P_\theta(\hat{x}^i c^i)$	$\frac{1}{N} \sum_{i=1}^N R(c^i, x^i) \log P_\theta(x^i c^i)$
Gradient	$\frac{1}{N} \sum_{i=1}^N \nabla \log P_\theta(\hat{x}^i c^i)$	$\frac{1}{N} \sum_{i=1}^N R(c^i, x^i) \nabla \log P_\theta(x^i c^i)$
Training Data	$\{(c^1, \hat{x}^1), \dots, (c^N, \hat{x}^N)\}$ $R(c^i, \hat{x}^i) = 1$	$\{(c^1, x^1), \dots, (c^N, x^N)\}$ obtained from interaction weighted by $R(c^i, x^i)$

Alpha GO style training !



- Let two agents talk to each other



How old are you?



See you.



How old are you?



I am 16.



See you.



See you.



I thought you were 12.



What make you
think so?

Using a pre-defined evaluation function to compute $R(h,x)$

Outline of Part III

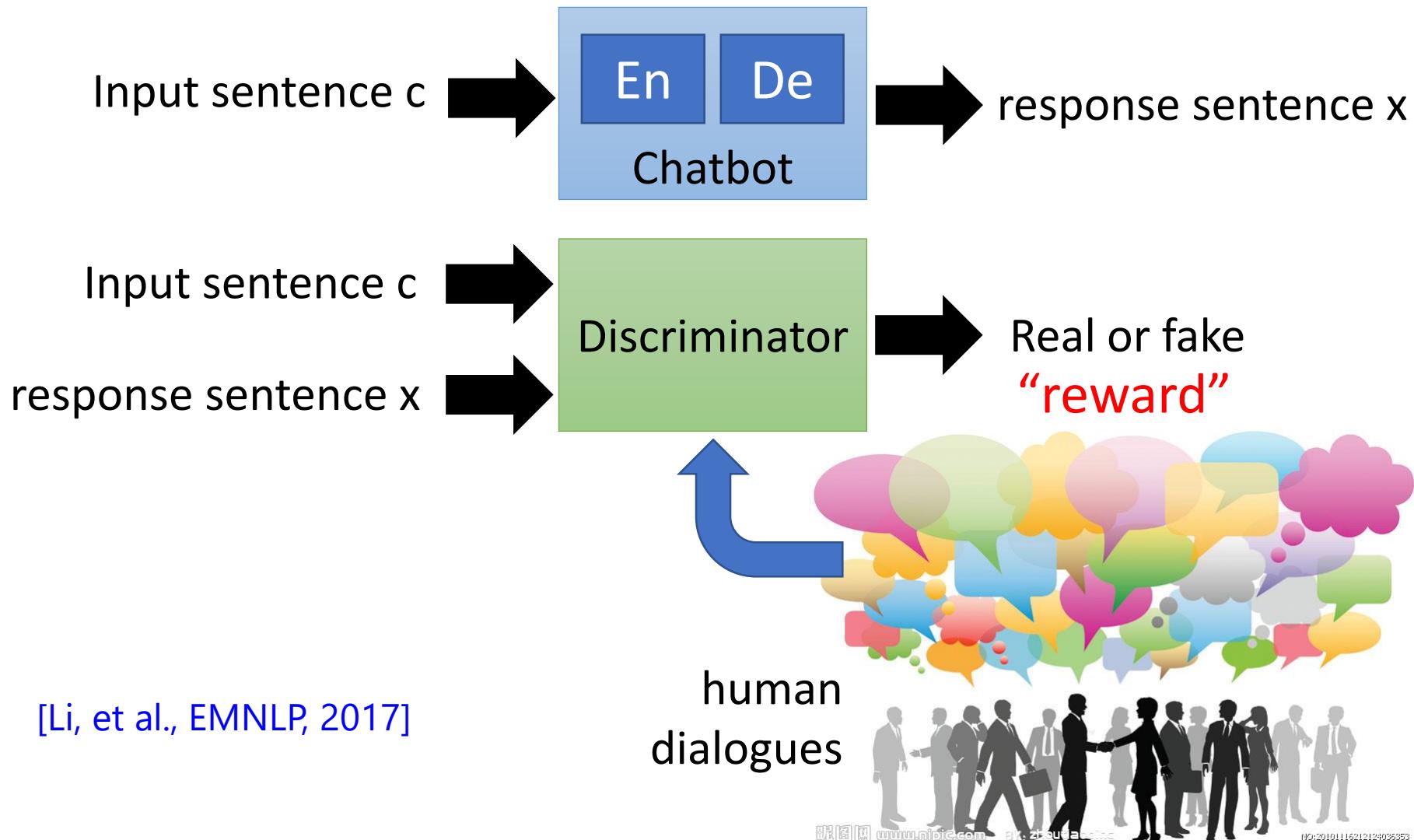
Improving Supervised Seq-to-seq Model

- RL (human feedback)
- GAN (discriminator feedback)

Unsupervised Seq-to-seq Model

- Text Style Transfer
- Unsupervised Abstractive Summarization
- Unsupervised Translation

Conditional GAN



Algorithm

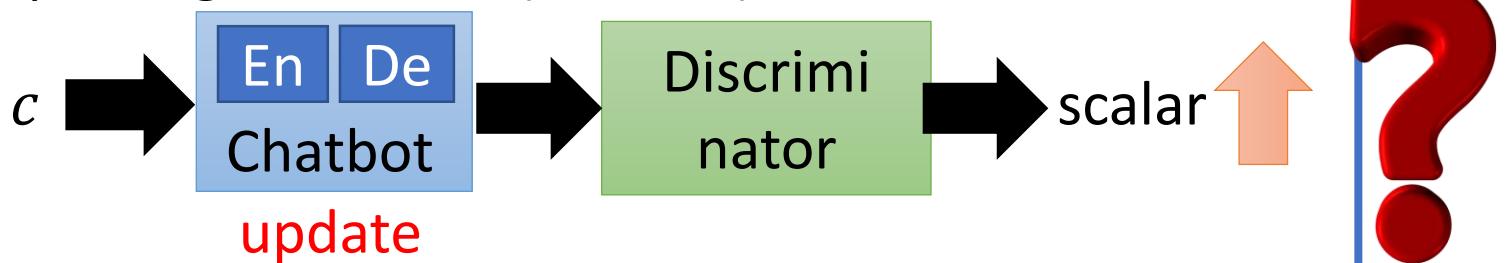
Training data:

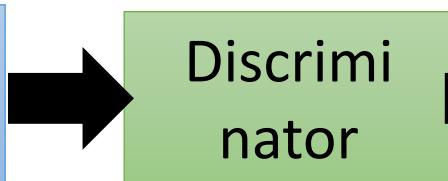
Pairs of conditional input c and response x

- Initialize generator G (chatbot) and discriminator D
- In each iteration:

- Sample input c and response x from training set
- Sample input c' from training set, and generate response \tilde{x} by $G(c')$
- Update D to increase $D(c, x)$ and decrease $D(c', \tilde{x})$

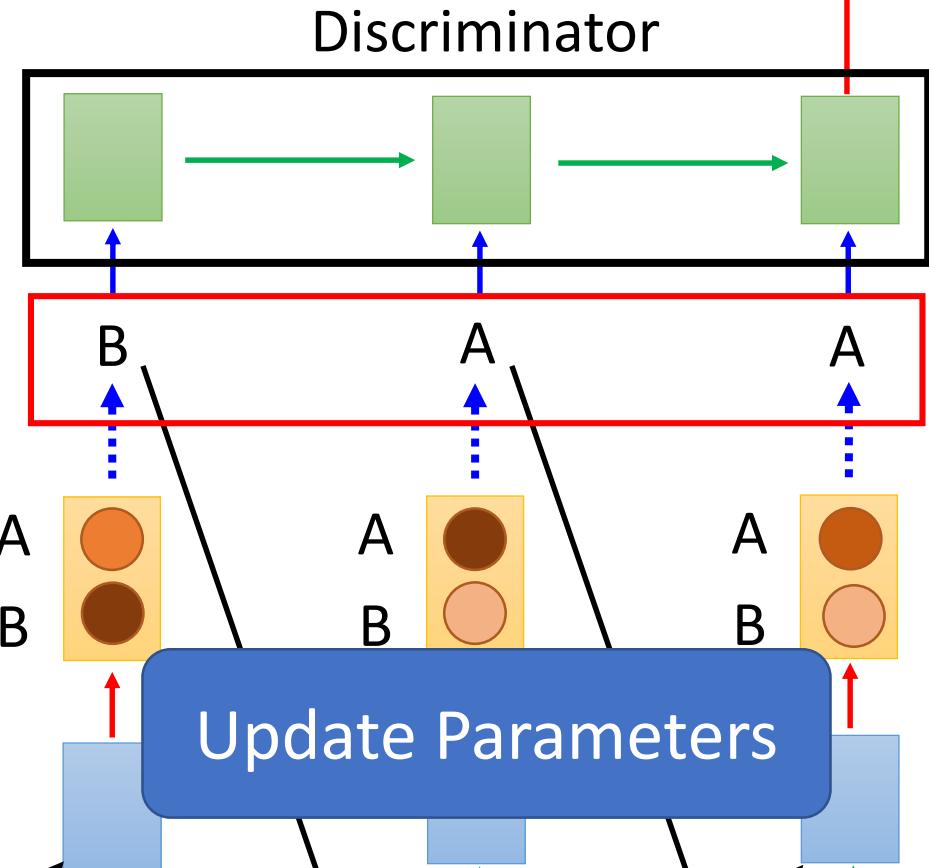
- Update generator G (chatbot) such that



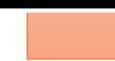


Can we use
gradient ascent?

NO!



Due to the sampling process, “discriminator+ generator”
is not differentiable



Three Categories of Solutions

Gumbel-softmax

- [Matt J. Kusner, et al, arXiv, 2016]

Continuous Input for Discriminator

- [Sai Rajeswar, et al., arXiv, 2017][Ofir Press, et al., ICML workshop, 2017][Zhen Xu, et al., EMNLP, 2017][Alex Lamb, et al., NIPS, 2016][Yizhe Zhang, et al., ICML, 2017]

“Reinforcement Learning”

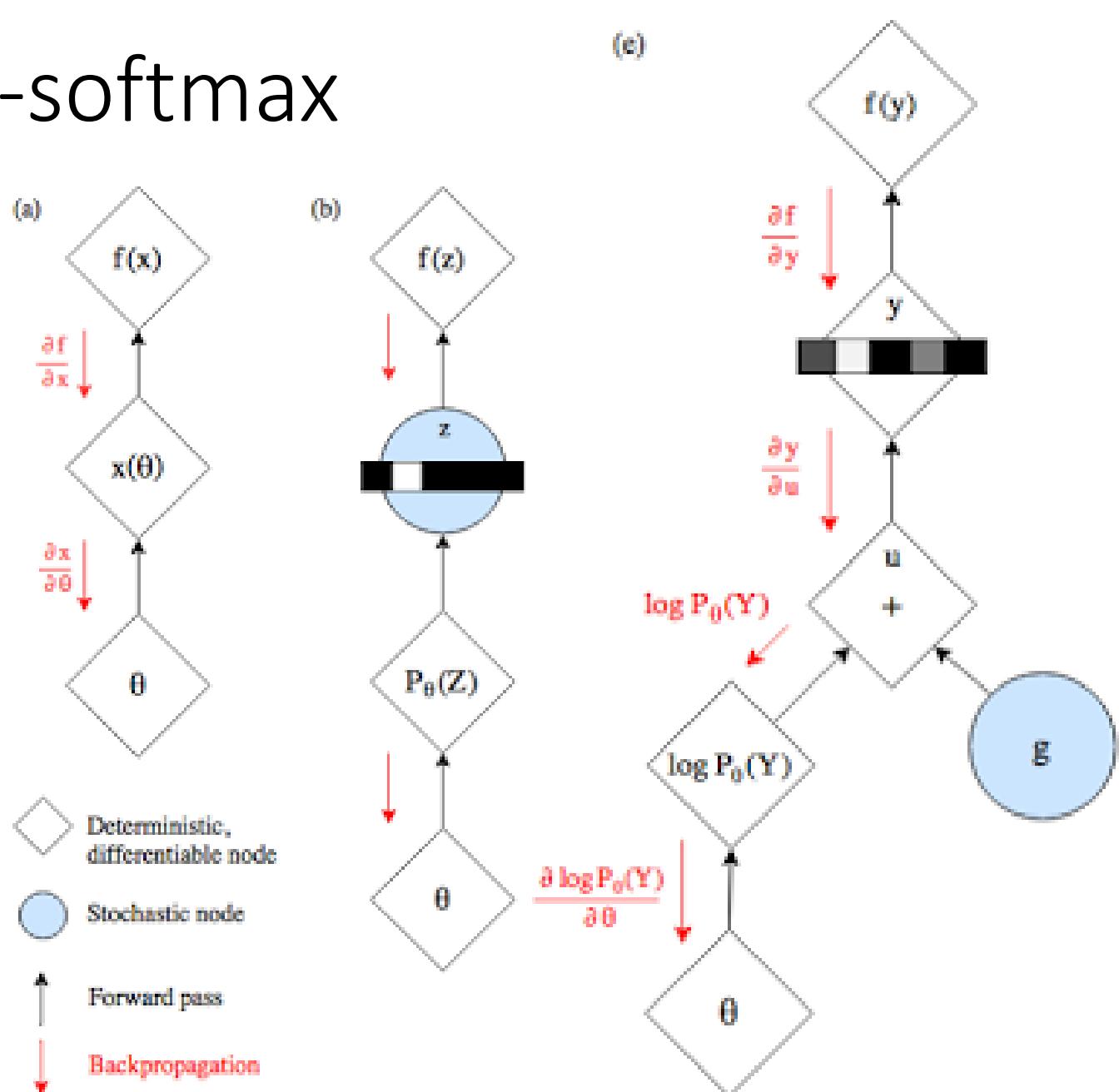
- [Yu, et al., AAAI, 2017][Li, et al., EMNLP, 2017][Tong Che, et al, arXiv, 2017][Jiaxian Guo, et al., AAAI, 2018][Kevin Lin, et al, NIPS, 2017][William Fedus, et al., ICLR, 2018]

Gumbel-softmax

<https://gabrielhuang.gitbooks.io/machine-learning/reparametrization-trick.html>

<https://casmls.github.io/general/2017/02/01/GumbelSoftmax.html>

<http://blog.evjang.com/2016/11/tutorial-categorical-variational.html>



Three Categories of Solutions

Gumbel-softmax

- [Matt J. Kusner, et al, arXiv, 2016]

Continuous Input for Discriminator

- [Sai Rajeswar, et al., arXiv, 2017][Ofir Press, et al., ICML workshop, 2017][Zhen Xu, et al., EMNLP, 2017][Alex Lamb, et al., NIPS, 2016][Yizhe Zhang, et al., ICML, 2017]

“Reinforcement Learning”

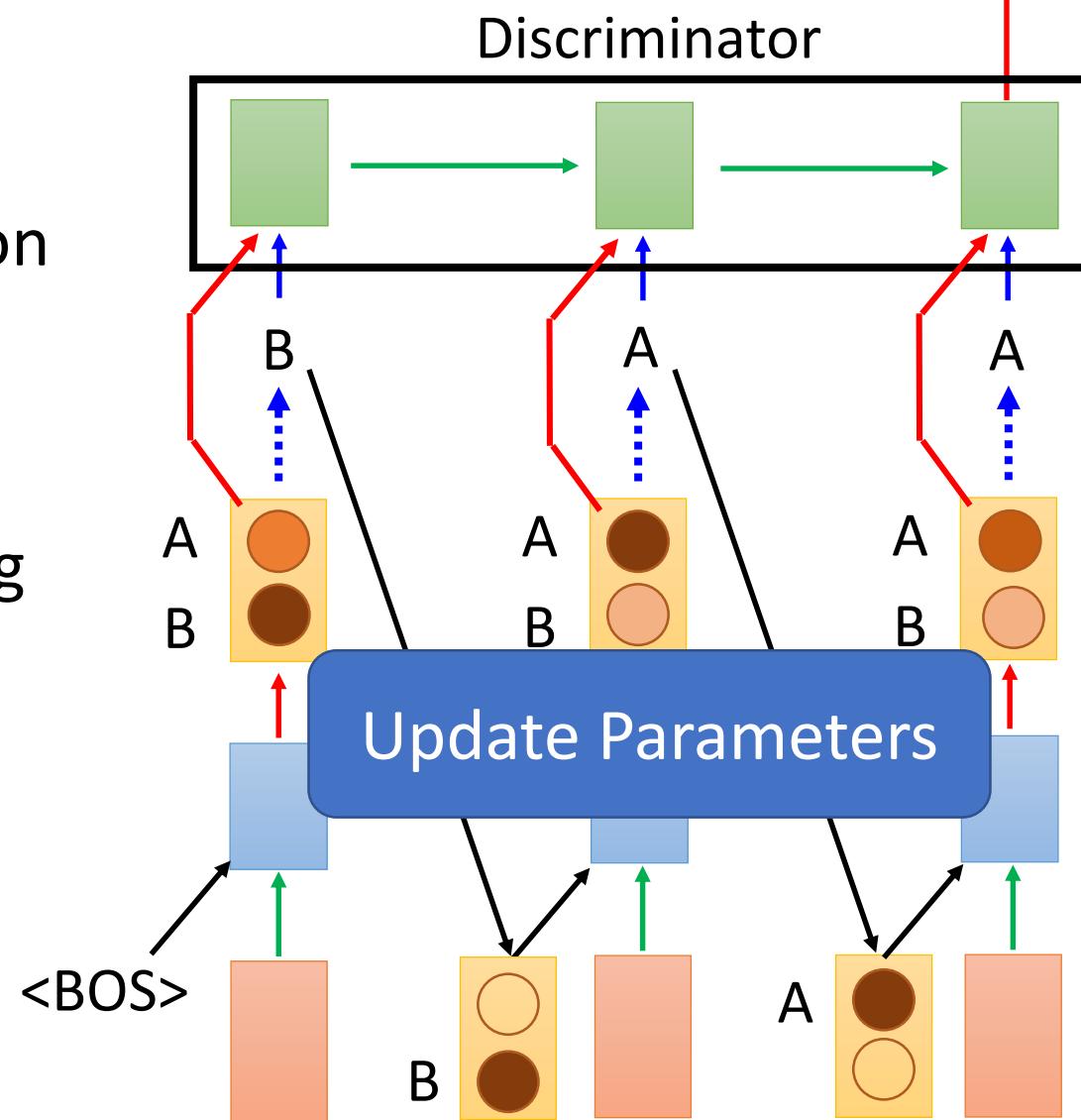
- [Yu, et al., AAAI, 2017][Li, et al., EMNLP, 2017][Tong Che, et al, arXiv, 2017][Jiaxian Guo, et al., AAAI, 2018][Kevin Lin, et al, NIPS, 2017][William Fedus, et al., ICLR, 2018]



Use the distribution as the input of discriminator

Avoid the sampling process

We can do backpropagation now.



What is the problem?

- Real sentence

1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Discriminator can immediately find the difference.

- Generated

Can never be 1-of-N

0.9	0.1	0.1	0	0
0.1	0.9	0.1	0	0
0	0	0.7	0.1	0
0	0	0.1	0.8	0.1
0	0	0	0.1	0.9

WGAN is helpful

Three Categories of Solutions

Gumbel-softmax

- [Matt J. Kusner, et al, arXiv, 2016]

Continuous Input for Discriminator

- [Sai Rajeswar, et al., arXiv, 2017][Ofir Press, et al., ICML workshop, 2017][Zhen Xu, et al., EMNLP, 2017][Alex Lamb, et al., NIPS, 2016][Yizhe Zhang, et al., ICML, 2017]

“Reinforcement Learning”

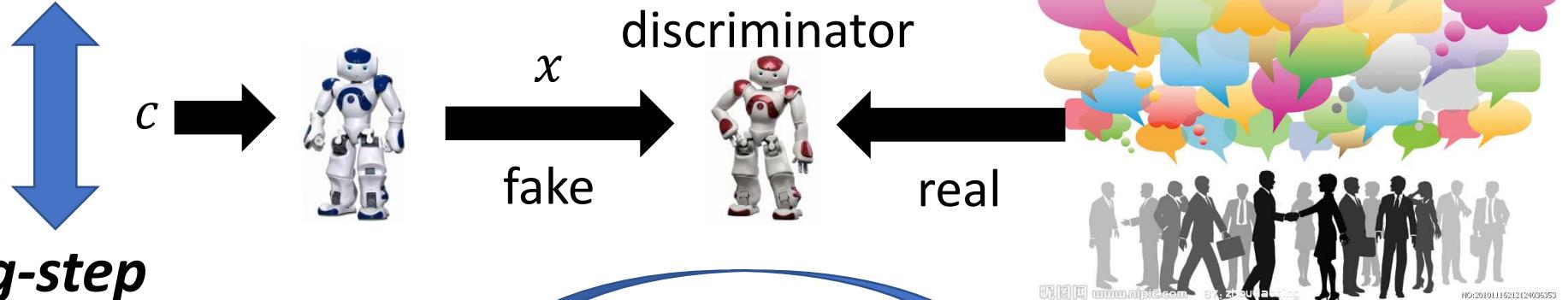
- [Yu, et al., AAAI, 2017][Li, et al., EMNLP, 2017][Tong Che, et al, arXiv, 2017][Jiaxian Guo, et al., AAAI, 2018][Kevin Lin, et al, NIPS, 2017][William Fedus, et al., ICLR, 2018]

Reinforcement Learning?

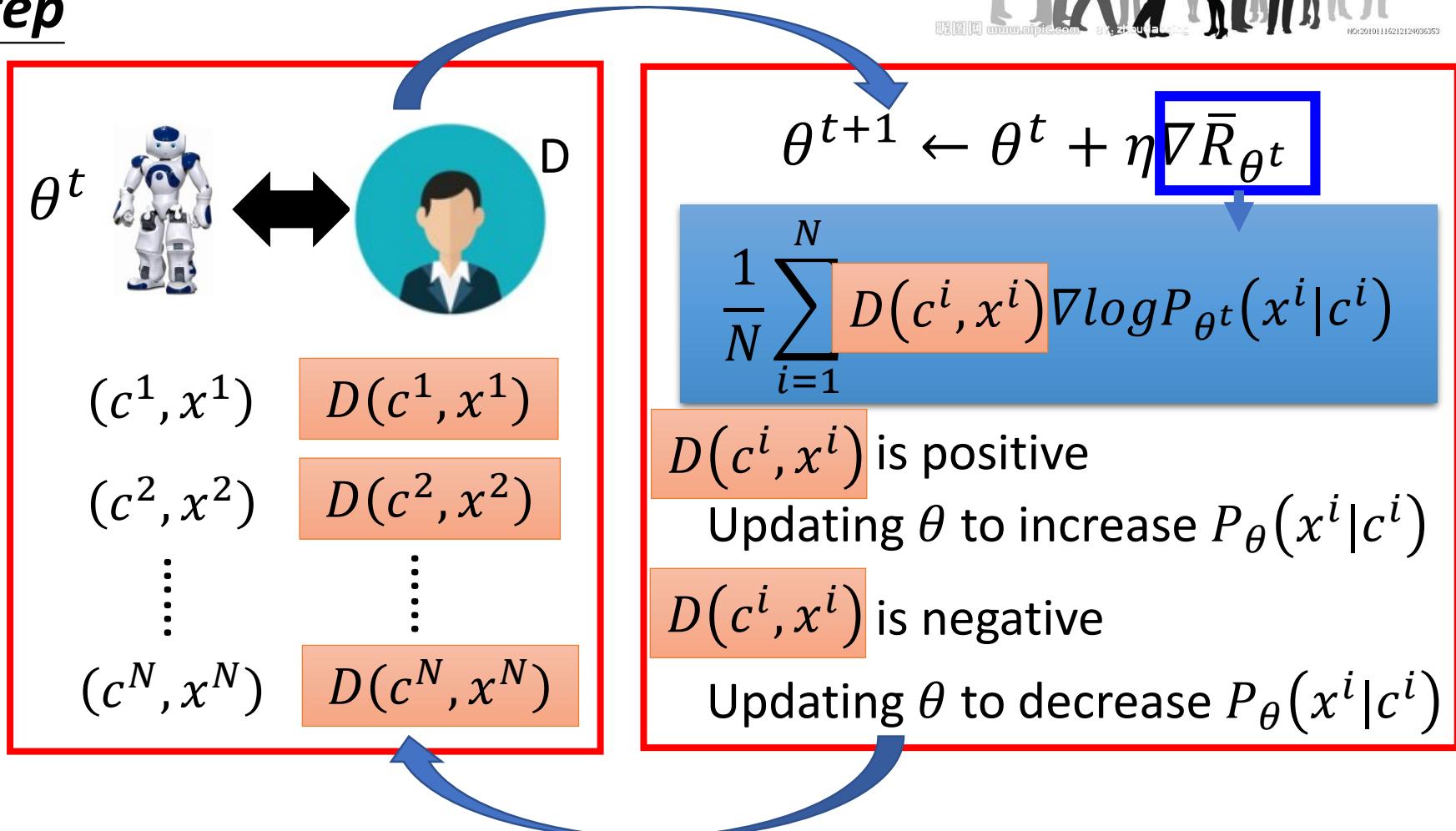


- Consider the output of discriminator as **reward**
 - Update generator to increase discriminator = to get maximum reward
 - Using the formulation of policy gradient, replace reward $R(c, x)$ with discriminator output $D(c, x)$
- Different from typical RL
 - The discriminator would update

d-step



g-step



Reward for Every Generation Step

$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{i=1}^N D(c^i, x^i) \nabla \log P_\theta(x^i | c^i)$$

c^i = “What is your name?” $D(c^i, x^i)$ is negative

x^i = “I don’t know” Update θ to decrease $\log P_\theta(x^i | c^i)$

$$\log P_\theta(x^i | c^i) = \log P(x_1^i | c^i) + \log P(x_2^i | c^i, x_1^i) + \log P(x_3^i | c^i, x_{1:2}^i)$$

$P("I" | c^i)$    

c^i = “What is your name?” $D(c^i, x^i)$ is positive

x^i = “I am John” Update θ to increase $\log P_\theta(x^i | c^i)$

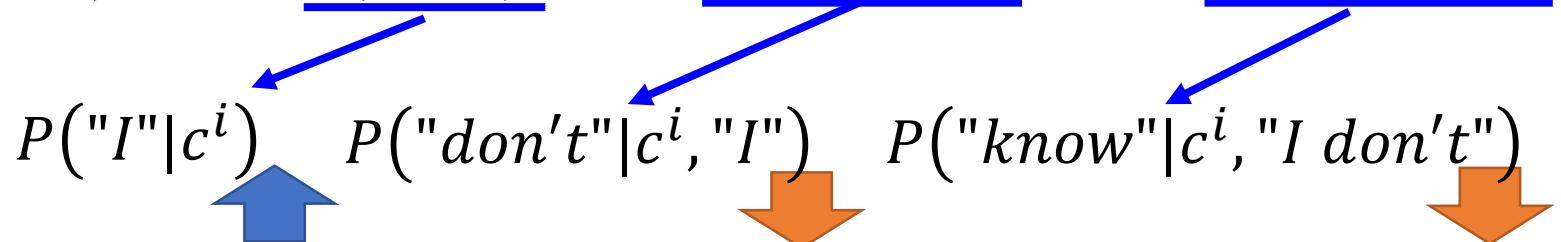
$$\log P_\theta(x^i | c^i) = \log P(x_1^i | c^i) + \log P(x_2^i | c^i, x_1^i) + \log P(x_3^i | c^i, x_{1:2}^i)$$

$P("I" | c^i)$   

Reward for Every Generation Step

$h^i = \text{"What is your name?"}$ $x^i = \text{"I don't know"}$

$$\log P_{\theta}(x^i | h^i) = \underbrace{\log P(x_1^i | c^i)} + \underbrace{\log P(x_2^i | c^i, x_1^i)} + \underbrace{\log P(x_3^i | c^i, x_{1:2}^i)}$$



$$\nabla \bar{R}_{\theta} \approx \frac{1}{N} \sum_{i=1}^N \underbrace{D(c^i, x^i)}_{\text{green}} \nabla \underbrace{\log P_{\theta}(x^i | c^i)}_{\text{red}}$$

$$\nabla \bar{R}_{\theta} \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \underbrace{(Q(c^i, x_{1:t}^i) - b)}_{\text{green}} \nabla \underbrace{\log P_{\theta}(x_t^i | c^i, x_{1:t-1}^i)}_{\text{red}}$$

Method 1. Monte Carlo (MC) Search [Yu, et al., AAAI, 2017]

Method 2. Discriminator For Partially Decoded Sequences

[Li, et al., EMNLP, 2017]

Tips: RankGAN

Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, Ming-Ting Sun,
"Adversarial Ranking for Language Generation", NIPS 2017

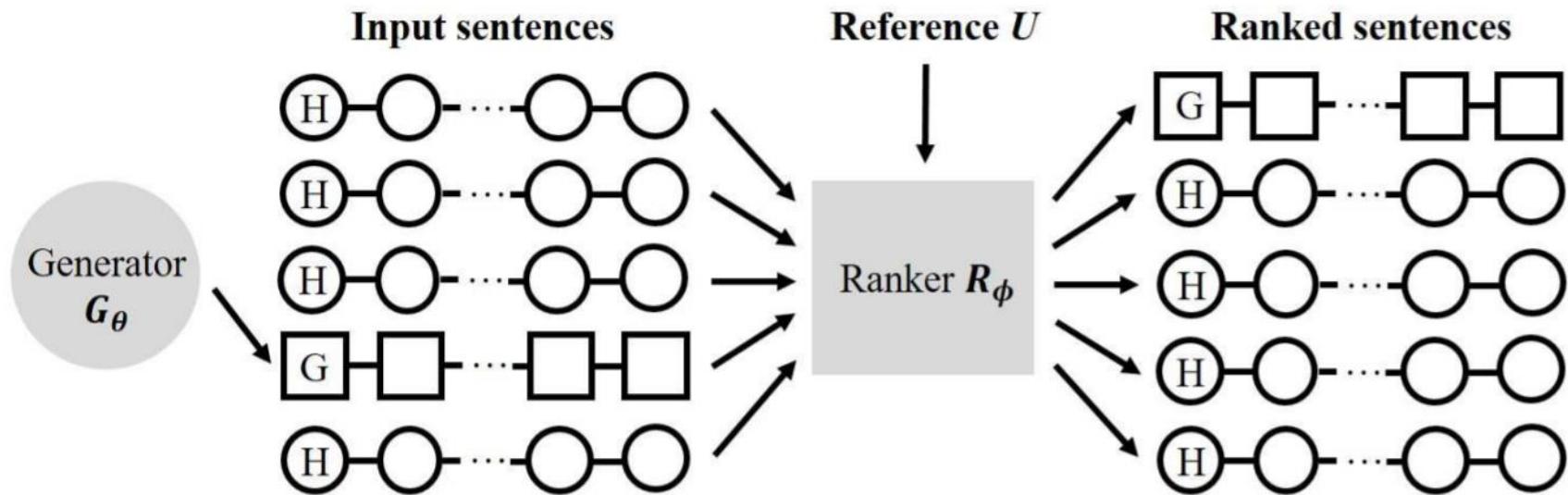


Image caption generation:

Method	BLEU-2	BLEU-3	BLEU-4
MLE	0.781	0.624	0.589
SeqGAN	0.815	0.636	0.587
RankGAN	0.845	0.668	0.614

Method	Human score
SeqGAN	3.44
RankGAN	4.61
Human-written	6.42

Experimental Results

Input	We've got to look for another route.
MLE	I'm sorry.
GAN	You're not going to be here for a while.
Input	You can save him by talking.
MLE	I don't know.
GAN	You know what's going on in there, you know what I mean?

- MLE frequently generates “I’m sorry”, “I don’t know”, etc. (corresponding to fuzzy images?)
- GAN generates longer and more complex responses (however, no strong evidence shows that they are better)

Find more comparison in the survey papers.

[Lu, et al., arXiv, 2018][Zhu, et al., arXiv, 2018]

More Applications

- Supervised machine translation [Wu, et al., arXiv 2017][Yang, et al., arXiv 2017]
- Supervised abstractive summarization [Liu, et al., AAAI 2018]
- Image/video caption generation [Rakshith Shetty, et al., ICCV 2017][Liang, et al., arXiv 2017]

If you are using seq2seq models,
consider to improve them by GAN.

Outline of Part III

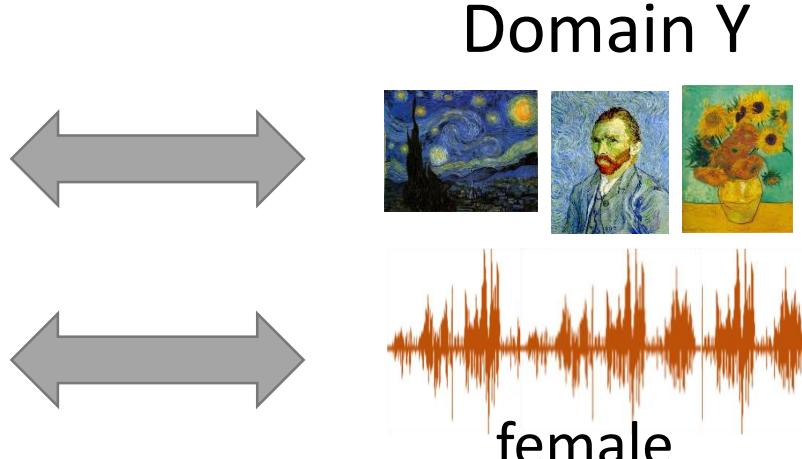
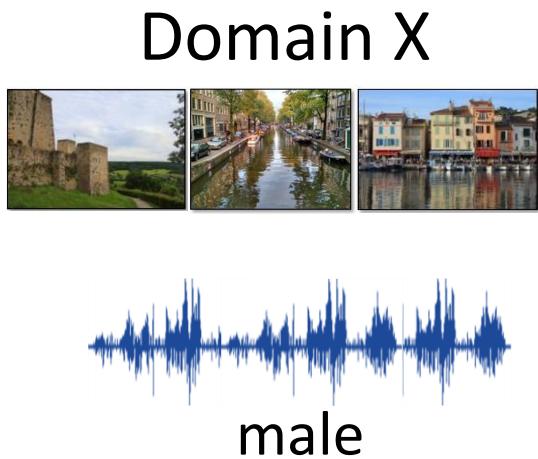
Conditional Sequence Generation

- RL (human feedback)
- GAN (discriminator feedback)

Unsupervised Conditional Sequence Generation

- Text Style Transfer
- Unsupervised Abstractive Summarization
- Unsupervised Translation

Text Style Transfer



It is good.
It's a good day.
I love you.

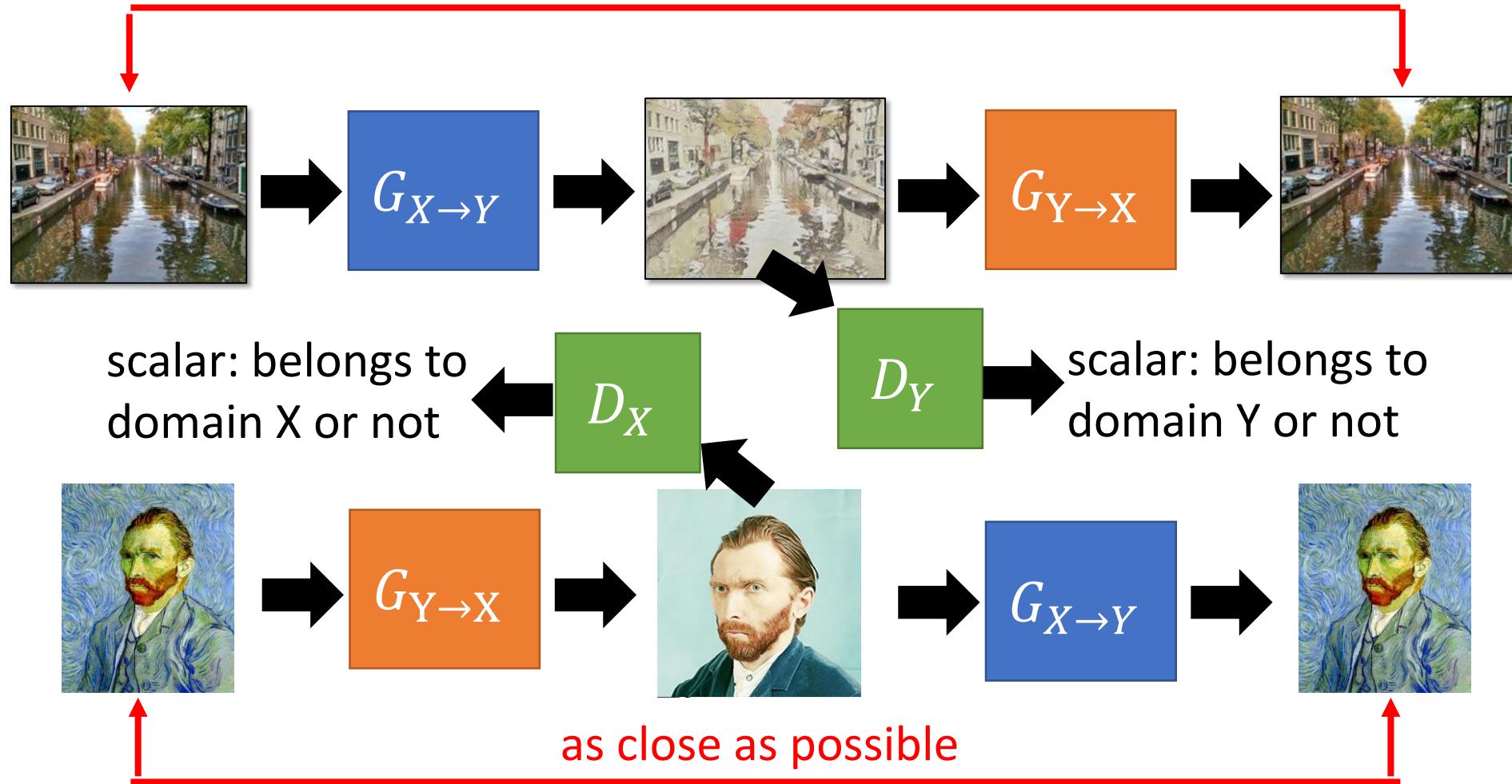
positive sentences

It is bad.
It's a bad day.
I don't love you.

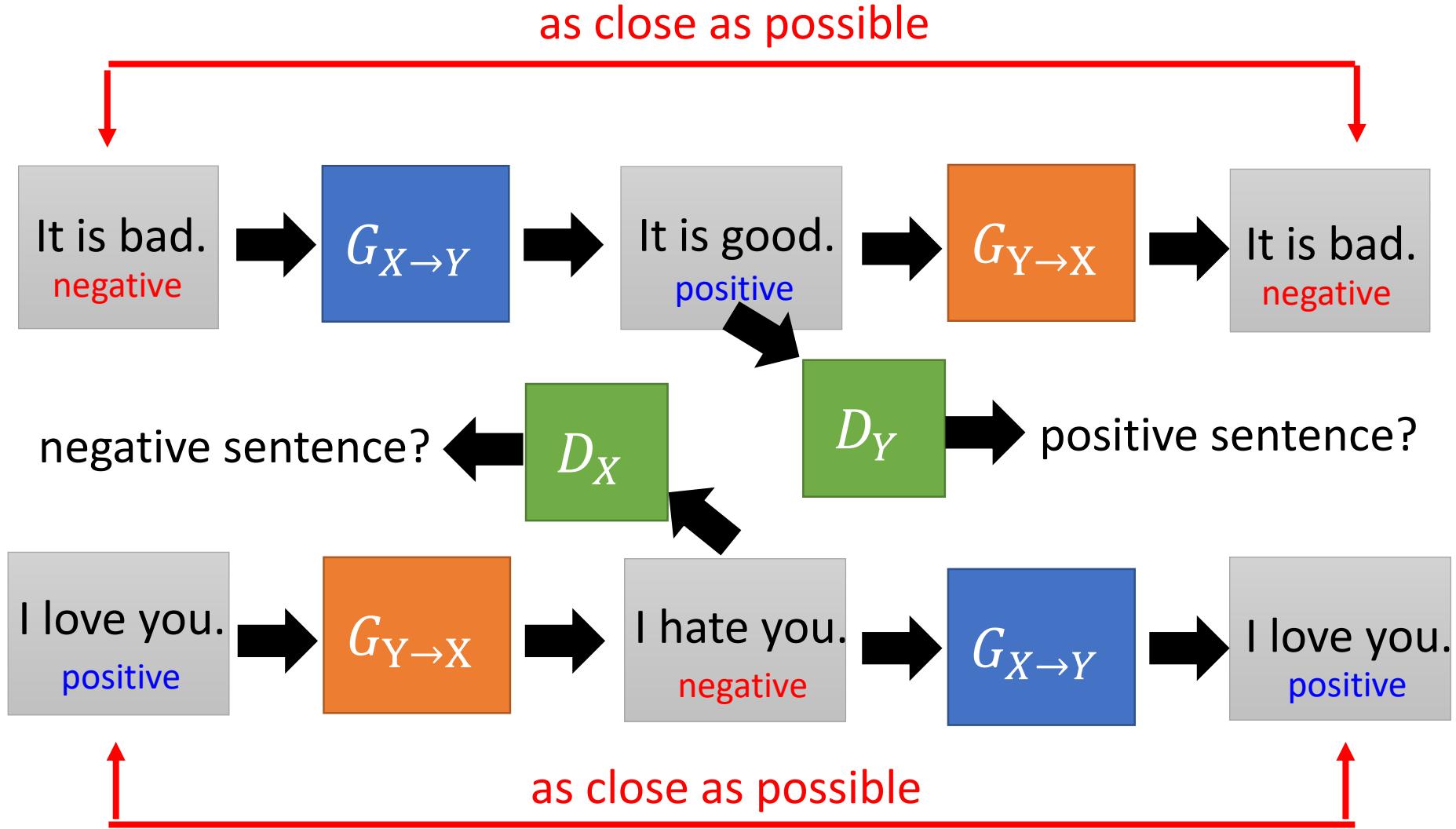
negative sentences

Direct Transformation

as close as possible



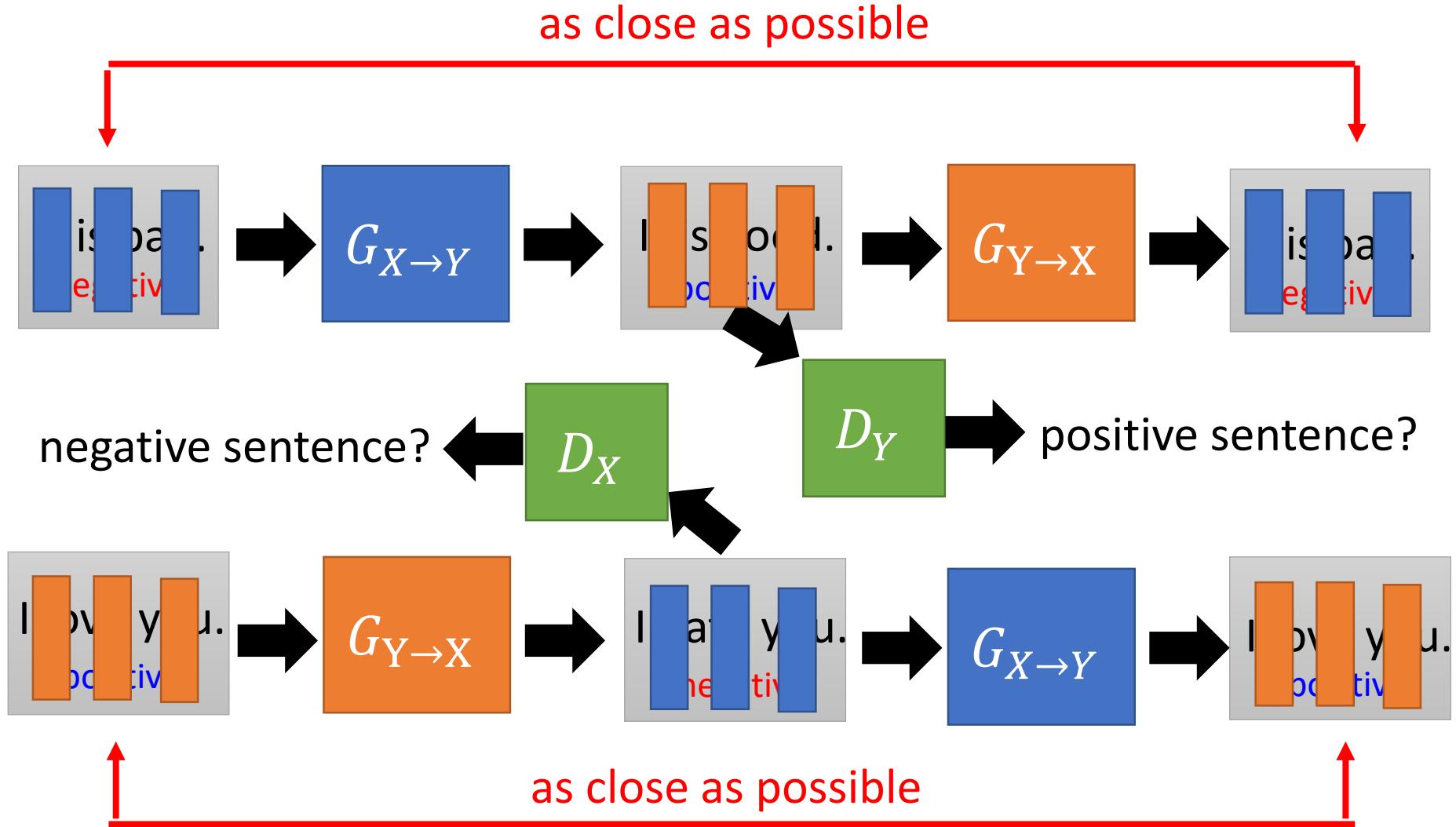
Direct Transformation



Direct Transformation

Discrete?

Word embedding
[Lee, et al., ICASSP, 2018]



- **Negative sentence to **positive** sentence:**

it's a crappy day → it's a great day

i wish you could be here → you could be here

it's not a good idea → it's good idea

i miss you → i love you

i don't love you → i love you

i can't do that → i can do that

i feel so sad → i happy

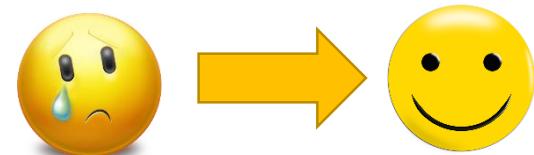
it's a bad day → it's a good day

it's a dummy day → it's a great day

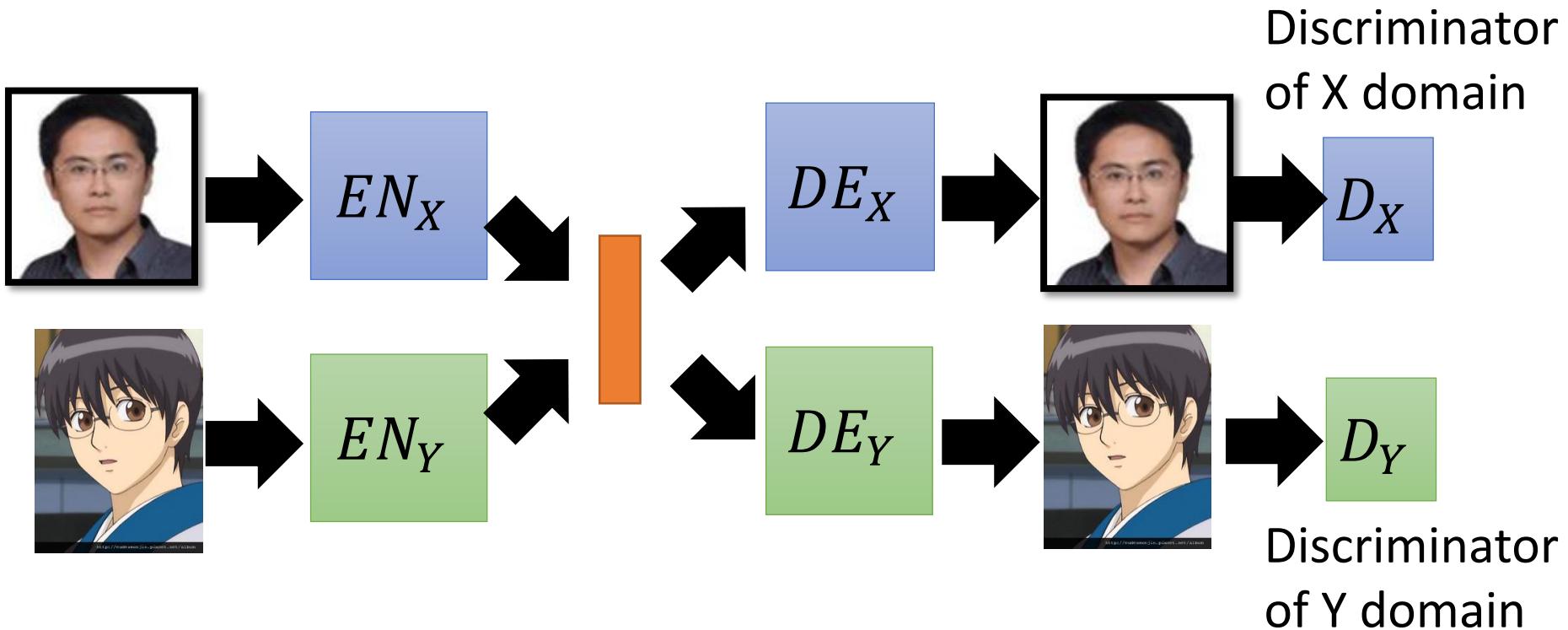
sorry for doing such a horrible thing → thanks for doing a great thing

my doggy is sick → my doggy is my doggy

my little doggy is sick → my little doggy is my little doggy



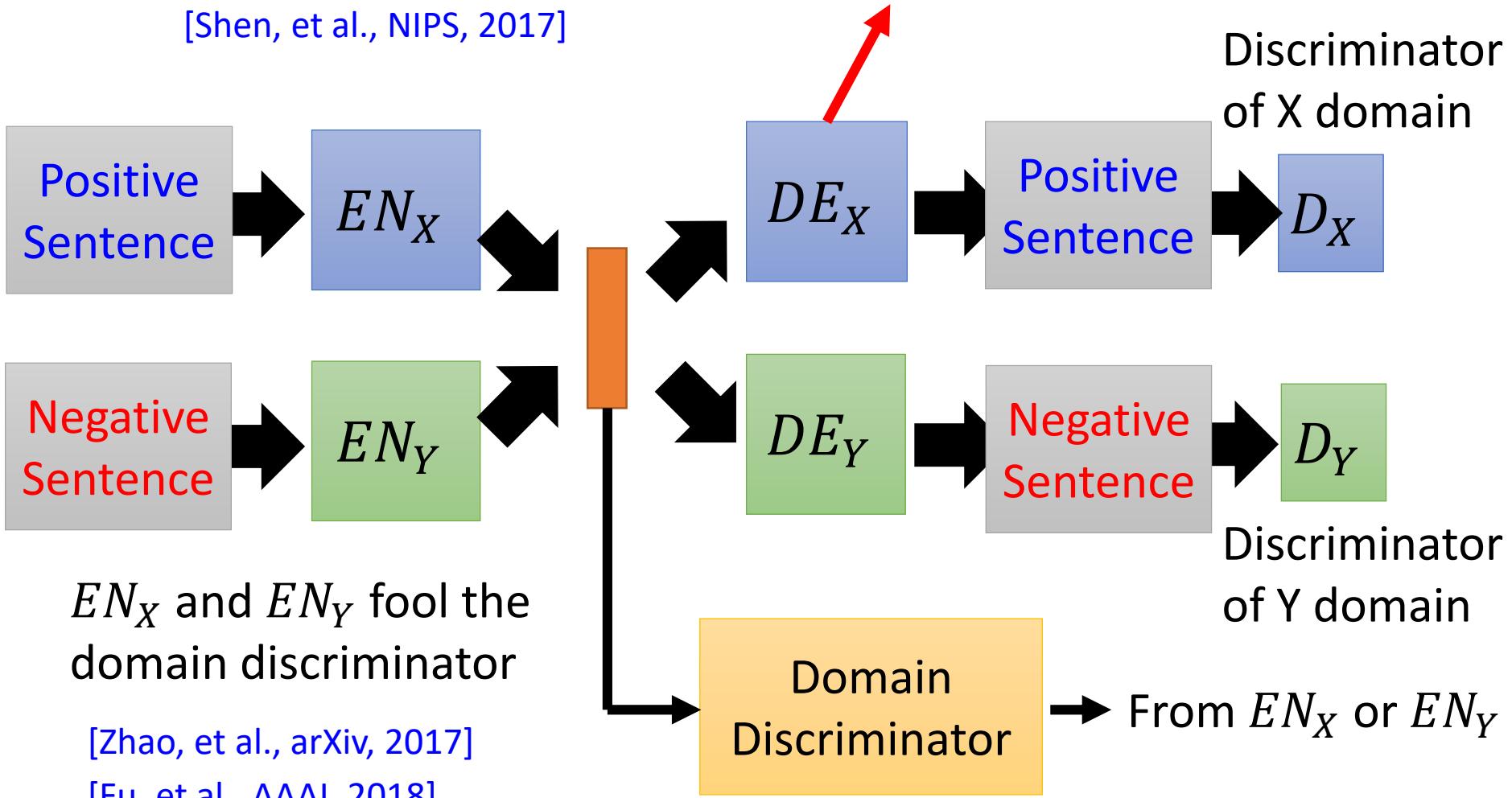
Projection to Common Space



Projection to Common Space

Decoder hidden layer as discriminator input

[Shen, et al., NIPS, 2017]



Outline of Part III

Improving Supervised Seq-to-seq Model

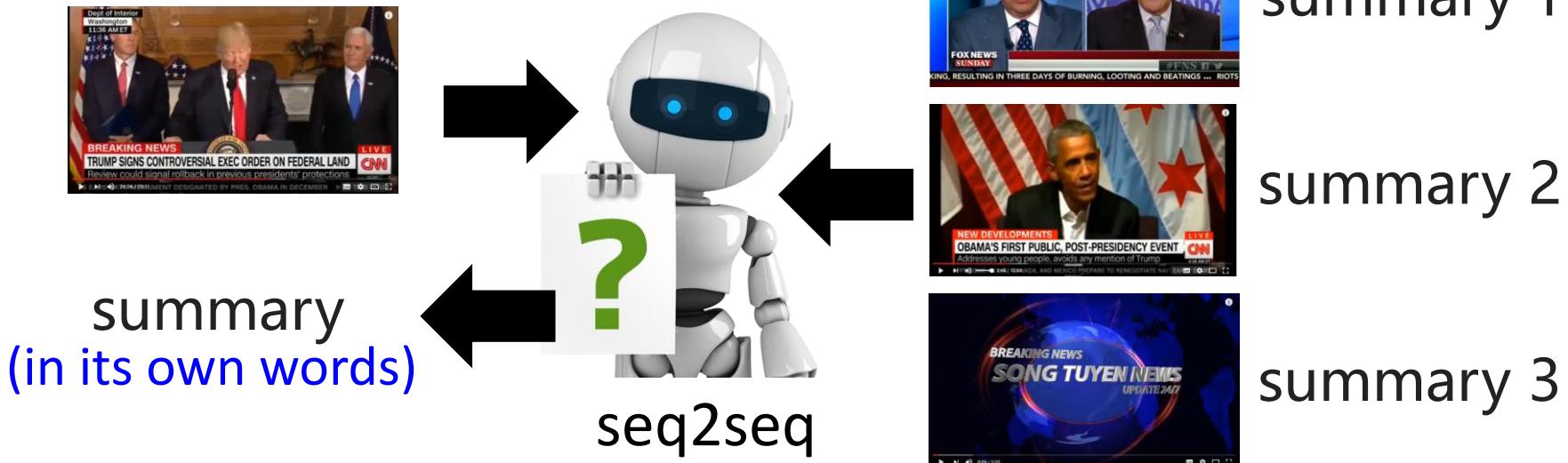
- RL (human feedback)
- GAN (discriminator feedback)

Unsupervised Seq-to-seq Model

- Text Style Transfer
- Unsupervised Abstractive Summarization
- Unsupervised Translation

Abstractive Summarization

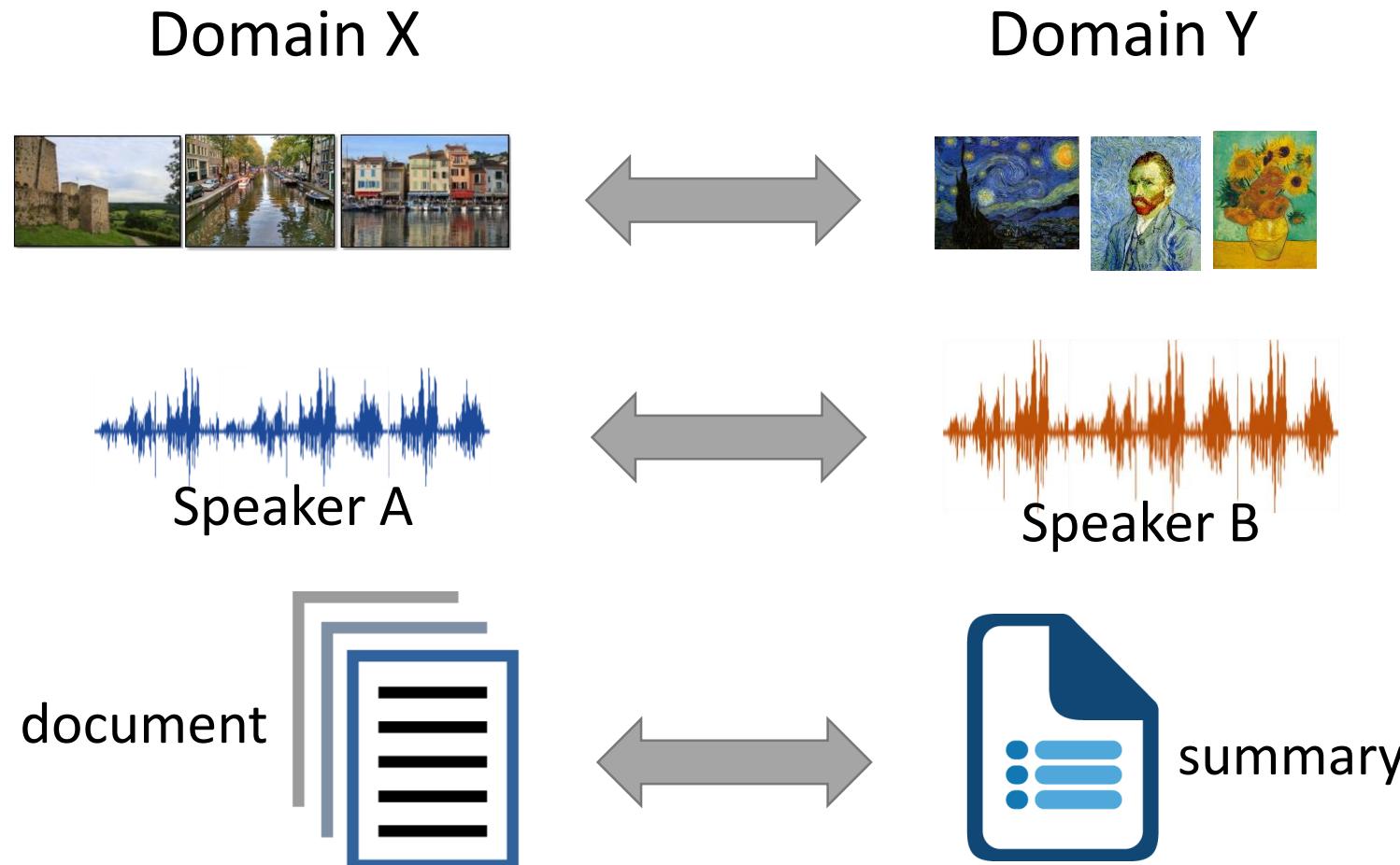
- Now machine can do **abstractive summary** by seq2seq (write summaries in its own words)



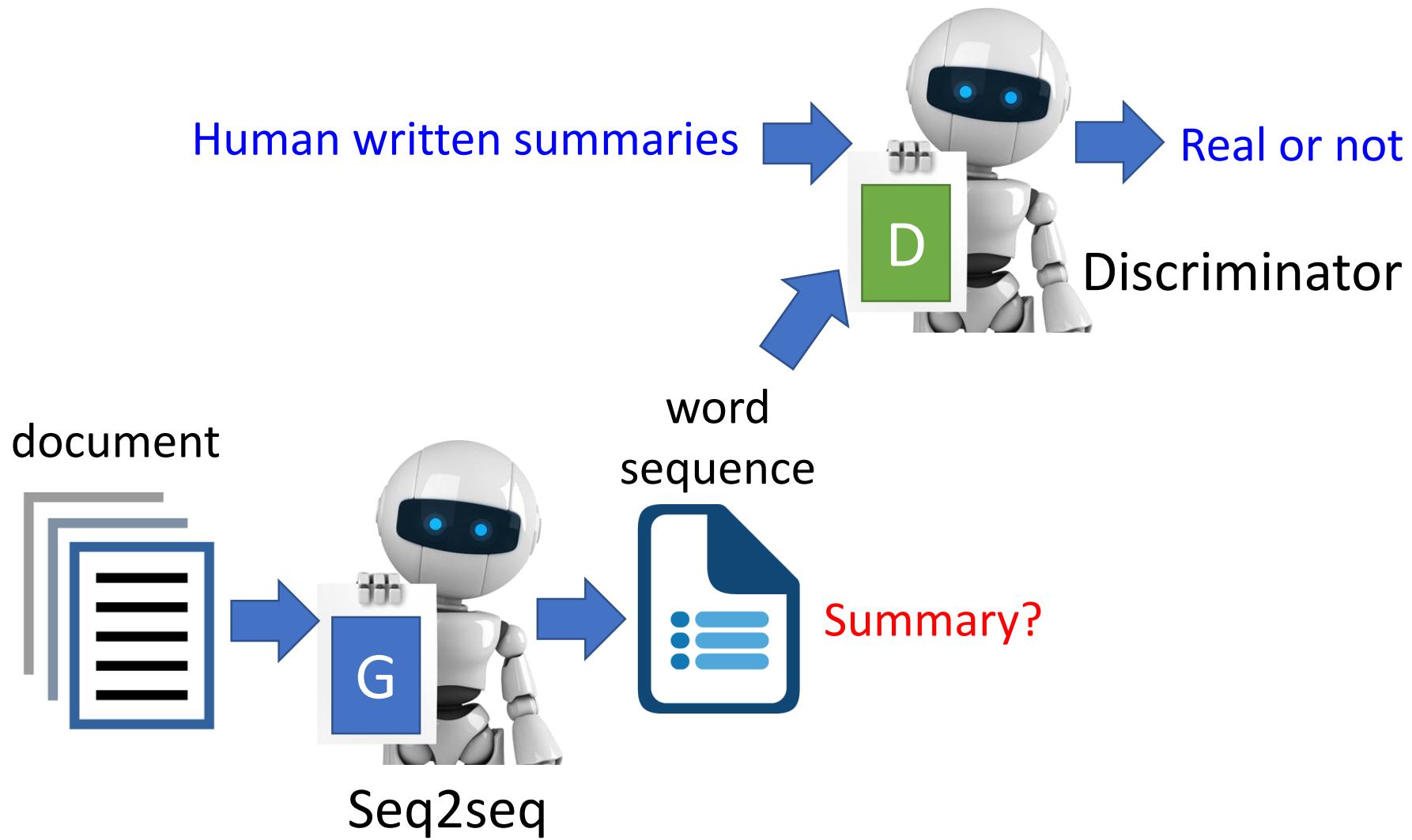
Supervised: We need lots of labelled training data.

Training Data

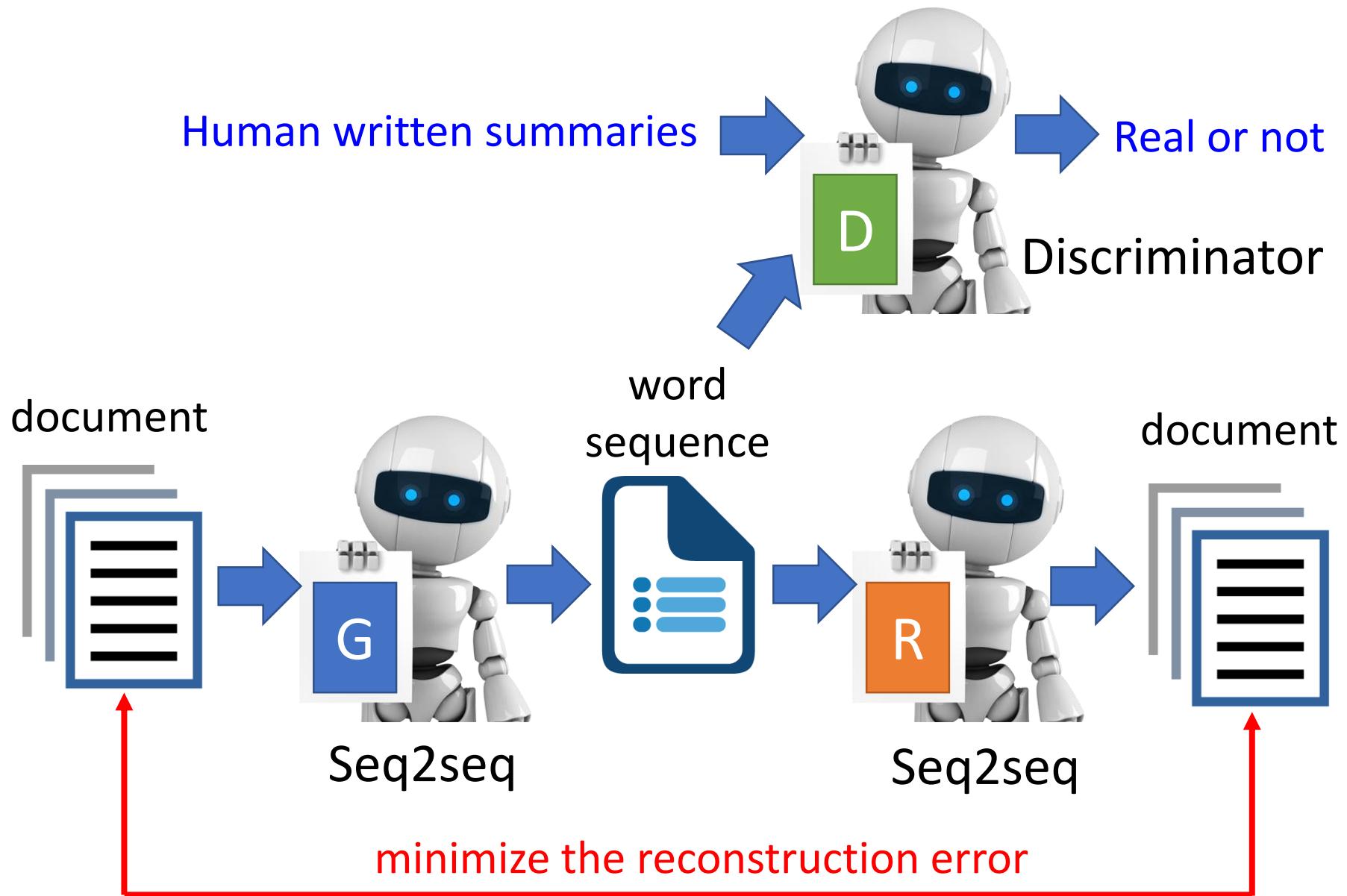
Review: Unsupervised Conditional Generation



Unsupervised Abstractive Summarization



Unsupervised Abstractive Summarization



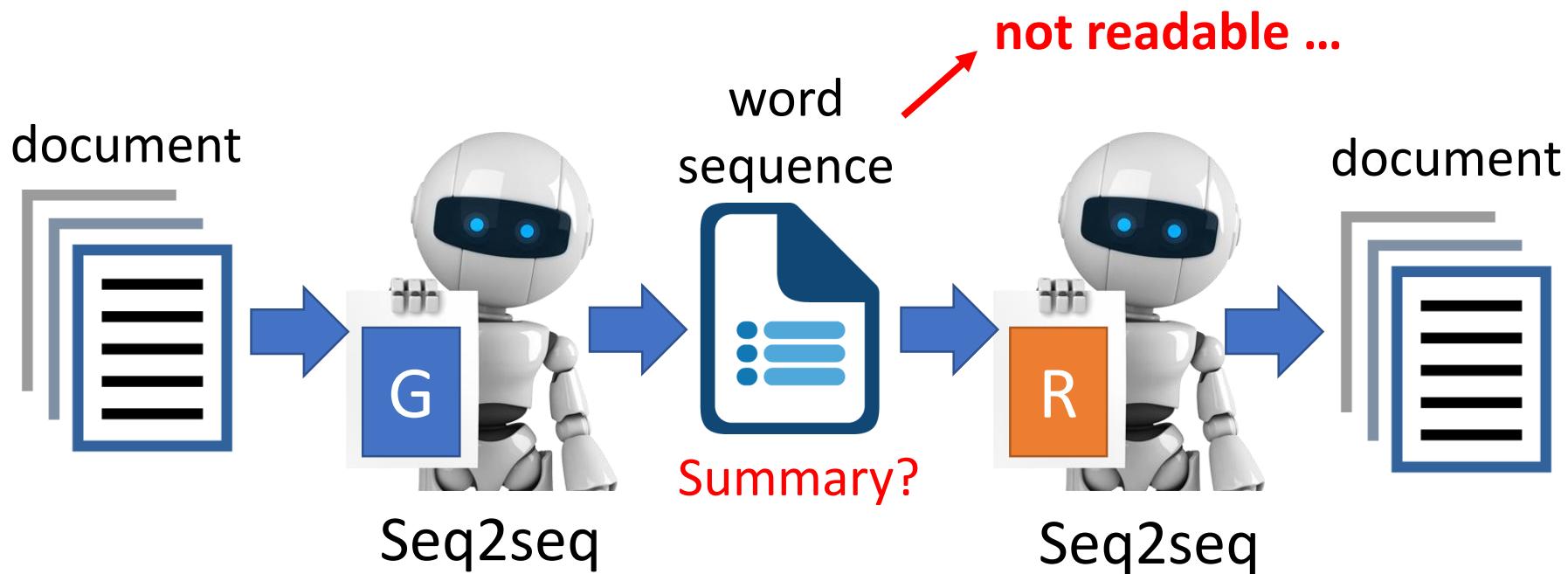
Unsupervised Abstractive Summarization

Only need a lot
of documents to
train the model



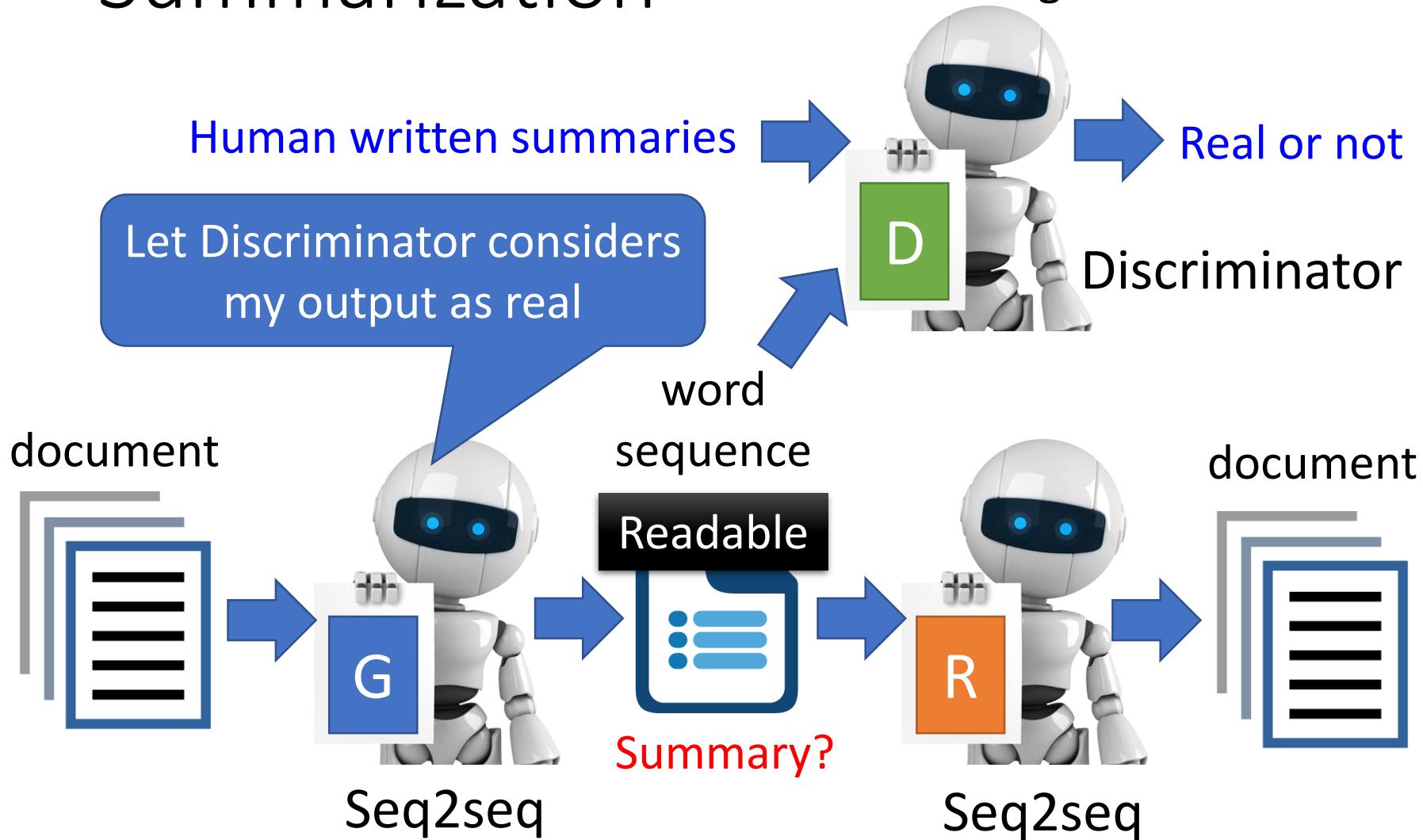
This is a ***seq2seq2seq auto-encoder***.

Using a sequence of words as latent representation.



Unsupervised Abstractive Summarization

REINFORCE algorithm is used.



Unsupervised Abstractive Summarization

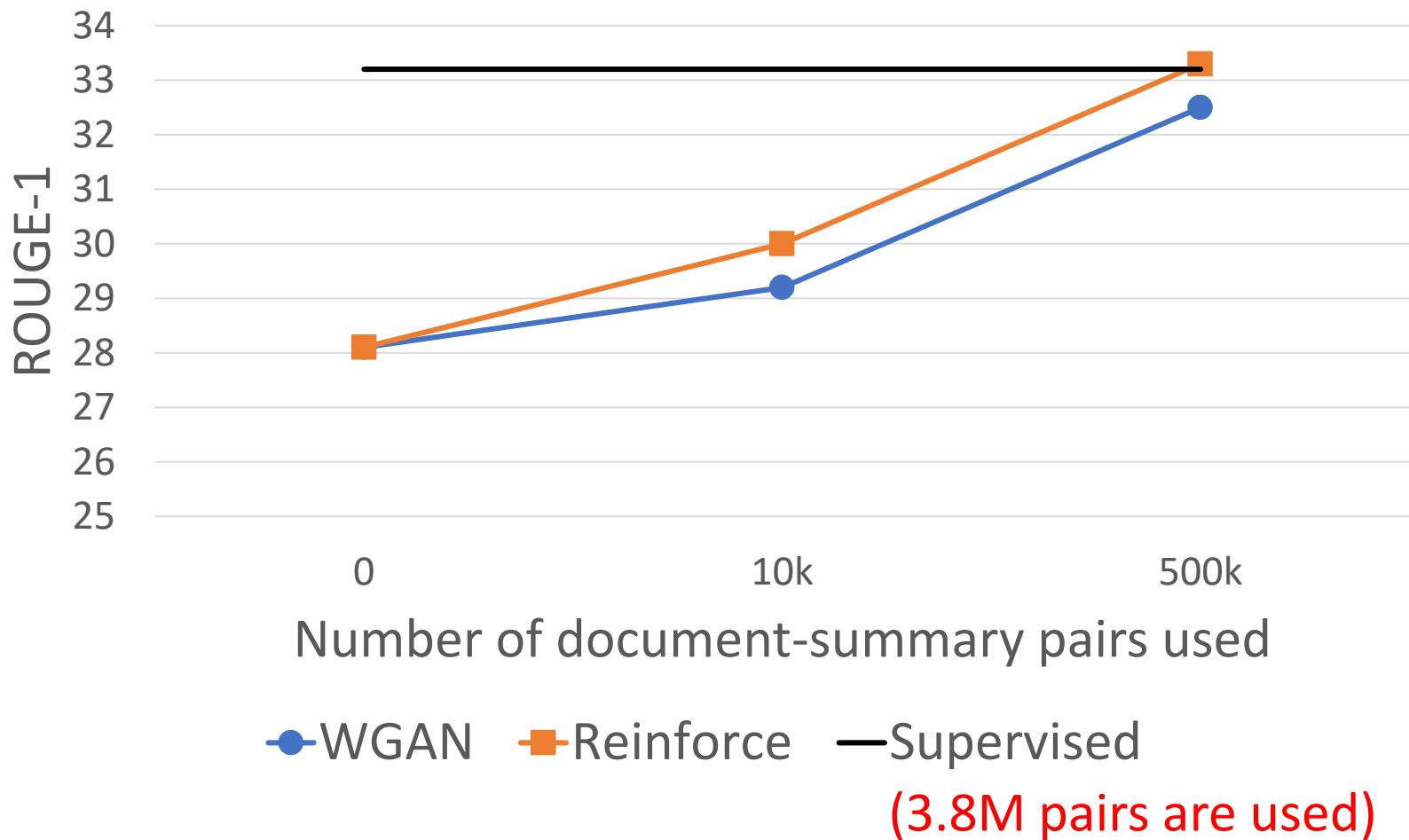
- **Document:** 澳大利亞今天與13個國家簽署了反興奮劑雙邊協議，旨在加強體育競賽之外的藥品檢查並共享研究成果
- **Summary:**
 - **Human:** 澳大利亞與13國簽署反興奮劑協議
 - **Unsupervised:** 澳大利亞加強體育競賽之外的藥品檢查
- **Document:** 中華民國奧林匹克委員會今天接到一九九二年冬季奧運會邀請函，由於主席張豐緒目前正在中南美洲進行友好訪問，因此尚未決定是否派隊赴賽
- **Summary:**
 - **Human:** 一九九二年冬季奧運會函邀我參加
 - **Unsupervised:** 奧委會接獲冬季奧運會邀請函

Unsupervised Abstractive Summarization

- **Document:**據此間媒體27日報道,印度尼西亞蘇門答臘島的兩個省近日來連降暴雨,洪水泛濫導致塌方,到26日為止至少已有60人喪生,100多人失蹤
- **Summary:**
 - **Human:**印尼水災造成60人死亡
 - **Unsupervised:**印尼門洪水泛濫導致塌雨
- **Document:**安徽省合肥市最近為領導幹部下基層做了新規定:一律輕車簡從,不準搞迎來送往、不準搞層層陪同
- **Summary:**
 - **Human:**合肥規定領導幹部下基層活動從簡
 - **Unsupervised:**合肥領導幹部下基層做搞迎來送往規定:一律簡

Semi-supervised Learning

Using
matched data



Outline of Part III

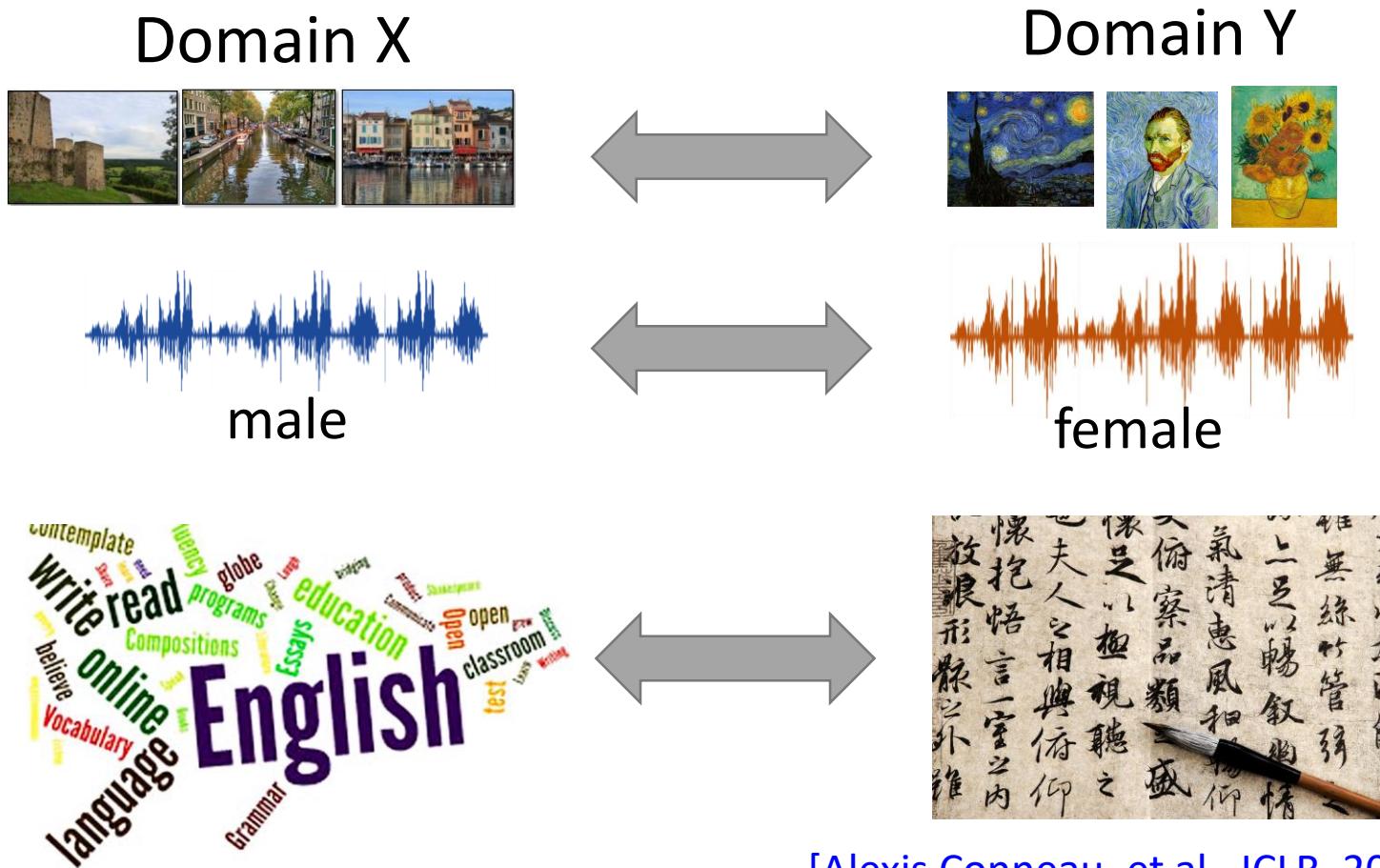
Improving Supervised Seq-to-seq Model

- RL (human feedback)
- GAN (discriminator feedback)

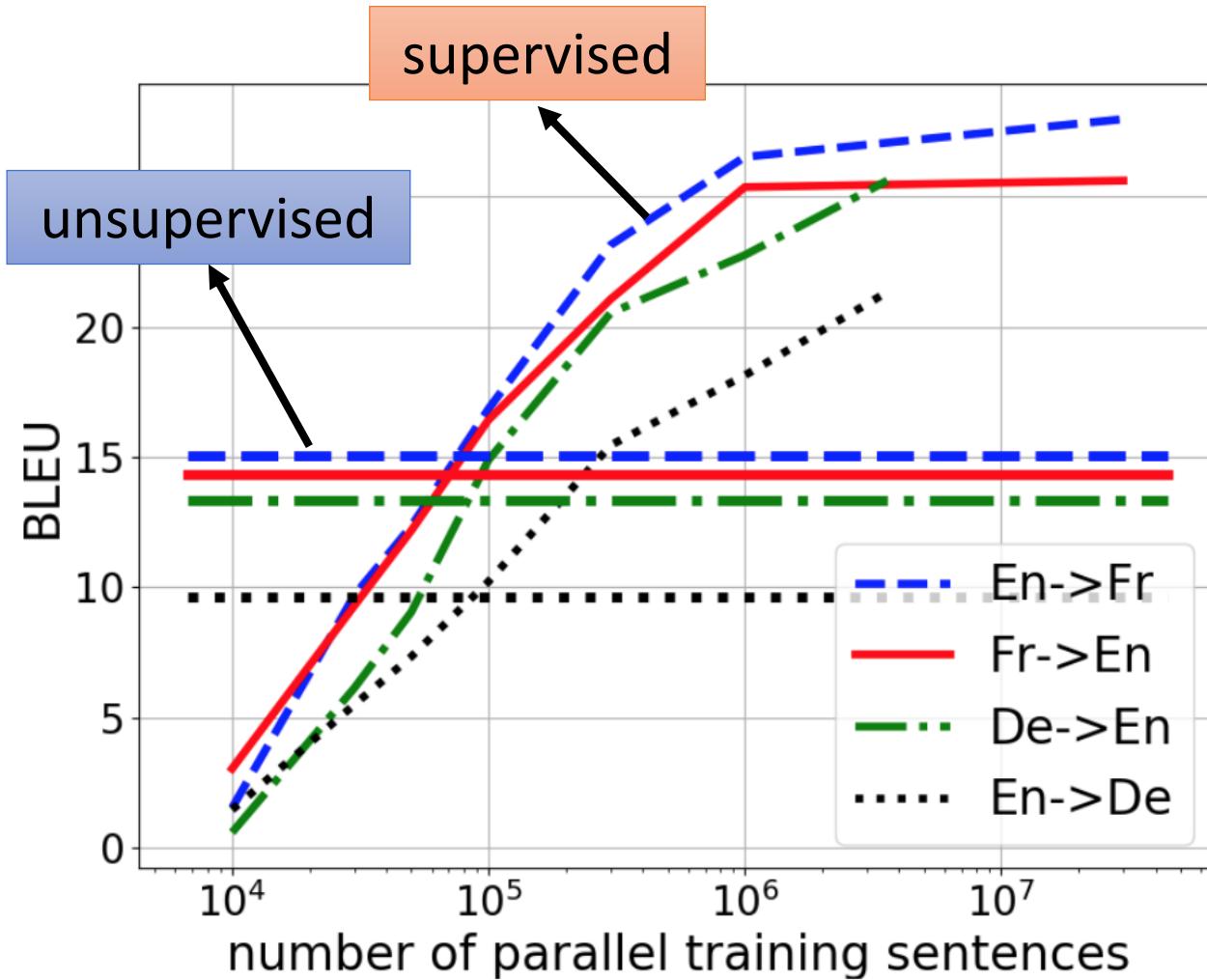
Unsupervised Seq-to-seq Model

- Text Style Transfer
- Unsupervised Abstractive Summarization
- Unsupervised Translation

Unsupervised Machine Translation



[Alexis Conneau, et al., ICLR, 2018]
[Guillaume Lample, et al., ICLR, 2018]

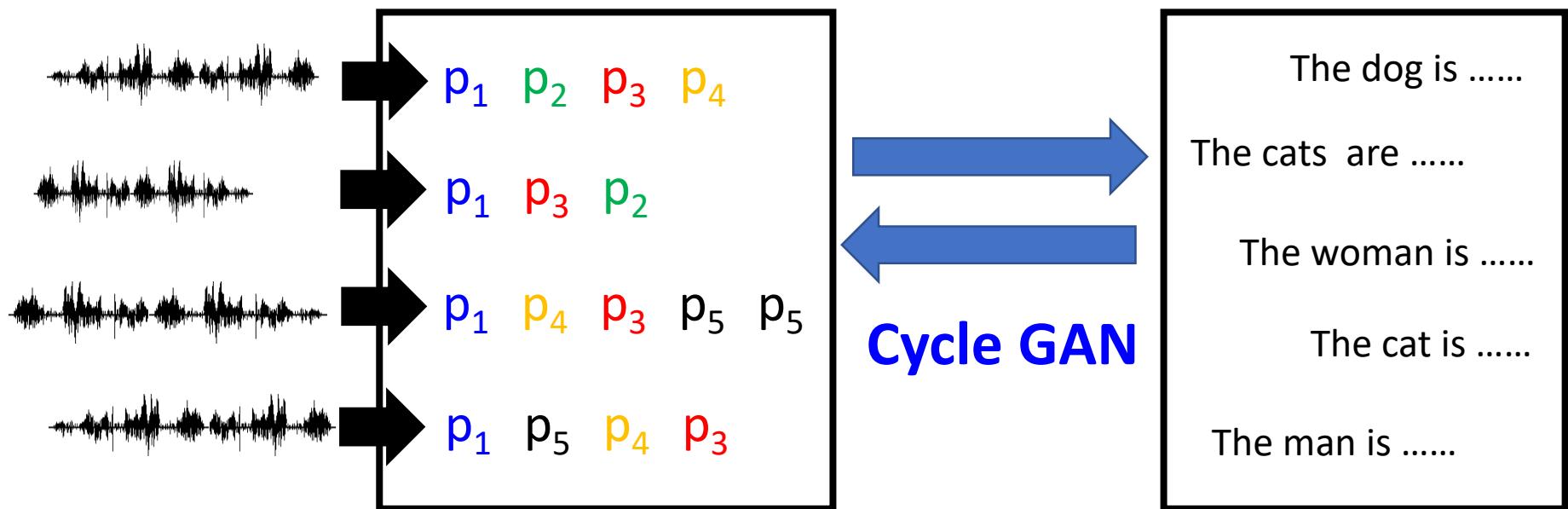


**Unsupervised learning
with 10M sentences**

=

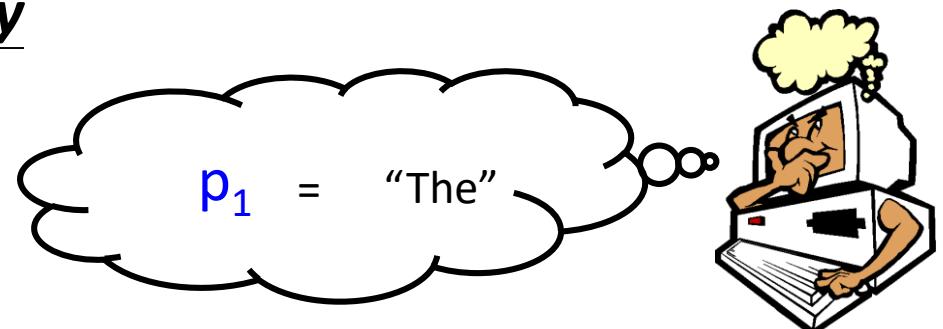
**Supervised learning with
100K sentence pairs**

Unsupervised Speech Recognition



Acoustic Pattern Discovery

Can we achieve
unsupervised speech
recognition?

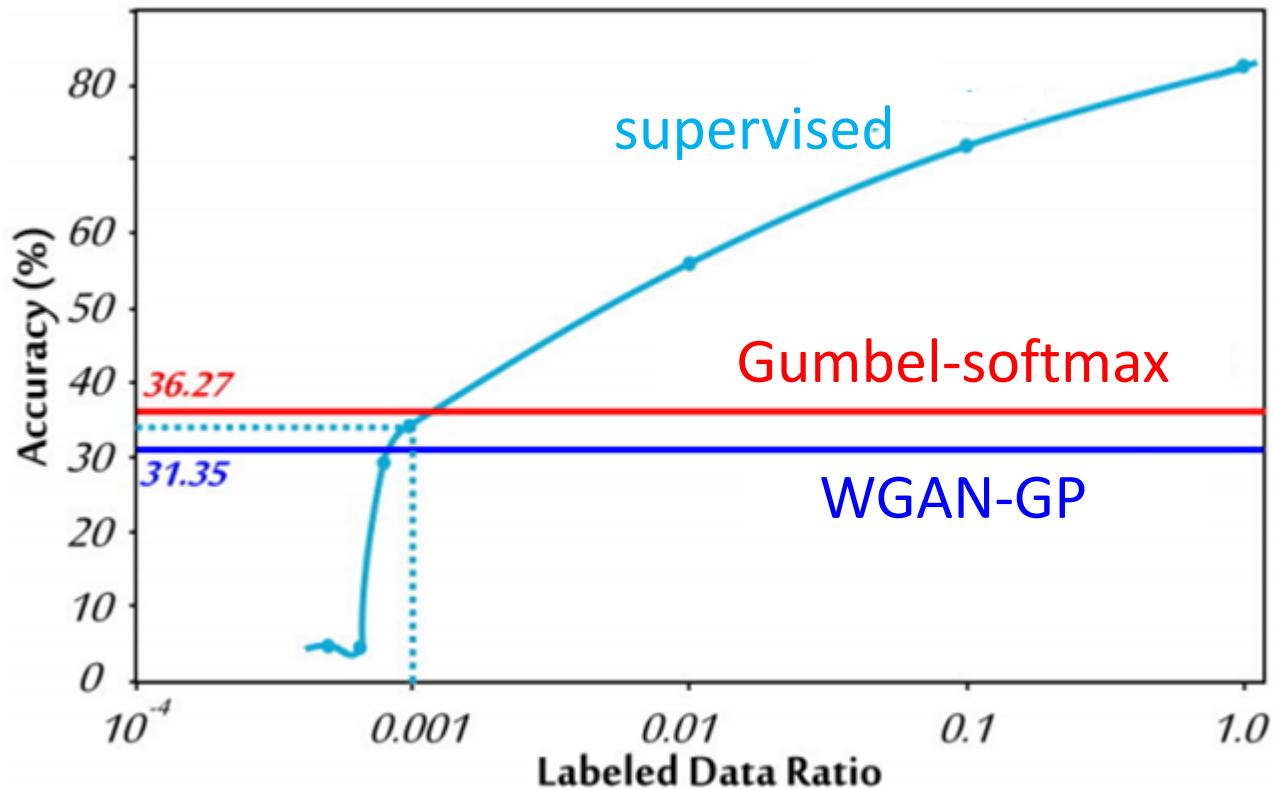


[Liu, et al., arXiv, 2018] [Chen, et al., arXiv, 2018]

Unsupervised Speech Recognition

- Phoneme recognition

Audio: TIMIT
Text: WMT



Concluding Remarks

Conditional Sequence Generation

- RL (human feedback)
- GAN (discriminator feedback)

Unsupervised Conditional Sequence Generation

- Text Style Transfer
- Unsupervised Abstractive Summarization
- Unsupervised Translation

Concluding Remarks

from A to Z

A

B

C

D

E

F

ACGAN

BiGAN

CycleGAN

DCGAN

EBGAN

fGAN

DuelGAN

G

H

I

J

K

L

GAN

?

InfoGAN

?

?

LSGAN

(only list those mentioned in class)

M**N****O****P****Q****R**

MMGAN

NSGAN

?

Progressive
GAN

?

Rank
GAN**S****T****U****V****W****X**

StackGAN

Triple
GANUnroll
GAN

VAEGAN

WGAN

XGAN

StarGAN

SeqGAN

Y

?

Z

?

Reference

- **Conditional Sequence Generation**
 - Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, Dan Jurafsky, Deep Reinforcement Learning for Dialogue Generation, EMNLP, 2016
 - Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, Dan Jurafsky, Adversarial Learning for Neural Dialogue Generation, EMNLP, 2017
 - Matt J. Kusner, José Miguel Hernández-Lobato, GANS for Sequences of Discrete Elements with the Gumbel-softmax Distribution, arXiv 2016
 - Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, Yoshua Bengio, Maximum-Likelihood Augmented Discrete Generative Adversarial Networks, arXiv 2017
 - Lantao Yu, Weinan Zhang, Jun Wang, Yong Yu, SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient, AAAI 2017

Reference

- **Conditional Sequence Generation**

- Sai Rajeswar, Sandeep Subramanian, Francis Dutil, Christopher Pal, Aaron Courville, Adversarial Generation of Natural Language, arXiv, 2017
- Ofir Press, Amir Bar, Ben Bogin, Jonathan Berant, Lior Wolf, Language Generation with Recurrent Generative Adversarial Networks without Pre-training, ICML workshop, 2017
- Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie Sun, Xiaolong Wang, Zhuoran Wang, Chao Qi , Neural Response Generation via GAN with an Approximate Embedding Layer, EMNLP, 2017
- Alex Lamb, Anirudh Goyal, Ying Zhang, Saizheng Zhang, Aaron Courville, Yoshua Bengio, Professor Forcing: A New Algorithm for Training Recurrent Networks, NIPS, 2016
- Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, Lawrence Carin, Adversarial Feature Matching for Text Generation, ICML, 2017
- Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, Jun Wang, Long Text Generation via Adversarial Training with Leaked Information, AAAI, 2018
- Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, Ming-Ting Sun, Adversarial Ranking for Language Generation, NIPS, 2017
- William Fedus, Ian Goodfellow, Andrew M. Dai, MaskGAN: Better Text Generation via Filling in the _____, ICLR, 2018

Reference

- **Conditional Sequence Generation**
 - Sidi Lu, Yaoming Zhu, Weinan Zhang, Jun Wang, Yong Yu, Neural Text Generation: Past, Present and Beyond, arXiv, 2018
 - Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, Yong Yu, Texygen: A Benchmarking Platform for Text Generation Models, arXiv, 2018
 - Zhen Yang, Wei Chen, Feng Wang, Bo Xu, Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets, NAACL, 2018
 - Lijun Wu, Yingce Xia, Li Zhao, Fei Tian, Tao Qin, Jianhuang Lai, Tie-Yan Liu, Adversarial Neural Machine Translation, arXiv 2017
 - Linqing Liu, Yao Lu, Min Yang, Qiang Qu, Jia Zhu, Hongyan Li, Generative Adversarial Network for Abstractive Text Summarization, AAAI 2018
 - Rakshith Shetty, Marcus Rohrbach, Lisa Anne Hendricks, Mario Fritz, Bernt Schiele, Speaking the Same Language: Matching Machine to Human Captions by Adversarial Training, ICCV 2017
 - Xiaodan Liang, Zhiting Hu, Hao Zhang, Chuang Gan, Eric P. Xing, Recurrent Topic-Transition GAN for Visual Paragraph Generation, arXiv 2017

Reference

- **Text Style Transfer**
 - Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, Rui Yan, Style Transfer in Text: Exploration and Evaluation, AAAI, 2018
 - Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola, Style Transfer from Non-Parallel Text by Cross-Alignment, NIPS 2017
 - Chih-Wei Lee, Yau-Shian Wang, Tsung-Yuan Hsu, Kuan-Yu Chen, Hung-Yi Lee, Lin-shan Lee, Scalable Sentiment for Sequence-to-sequence Chatbot Response with Performance Analysis, ICASSP, 2018
 - Junbo (Jake) Zhao, Yoon Kim, Kelly Zhang, Alexander M. Rush, Yann LeCun, Adversarially Regularized Autoencoders, arxiv, 2017

Reference

- **Unsupervised Machine Translation**
 - Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, Hervé Jégou, Word Translation Without Parallel Data, ICRL 2018
 - Guillaume Lample, Ludovic Denoyer, Marc'Aurelio Ranzato, Unsupervised Machine Translation Using Monolingual Corpora Only, ICRL 2018
- **Unsupervised Speech Recognition**
 - Da-Rong Liu, Kuan-Yu Chen, Hung-Yi Lee, Lin-shan Lee, Completely Unsupervised Phoneme Recognition by Adversarially Learning Mapping Relationships from Audio Embeddings, arXiv, 2018
 - Yi-Chen Chen, Chia-Hao Shen, Sung-Feng Huang, Hung-yi Lee, Towards Unsupervised Automatic Speech Recognition Trained by Unaligned Speech and Text only, arXiv, 2018

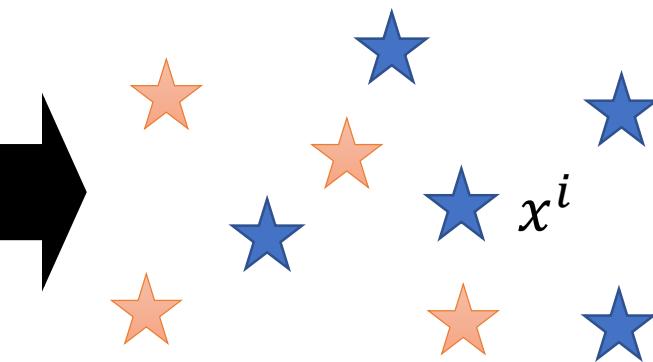
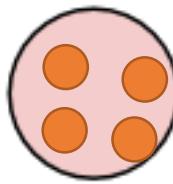
Evaluation

Ref: Lucas Theis, Aäron van den Oord, Matthias Bethge, “A note on the evaluation of generative models”, arXiv preprint, 2015

Likelihood

- ★ : real data (not observed during training)
- ★ : generated data

Prior
Distribution



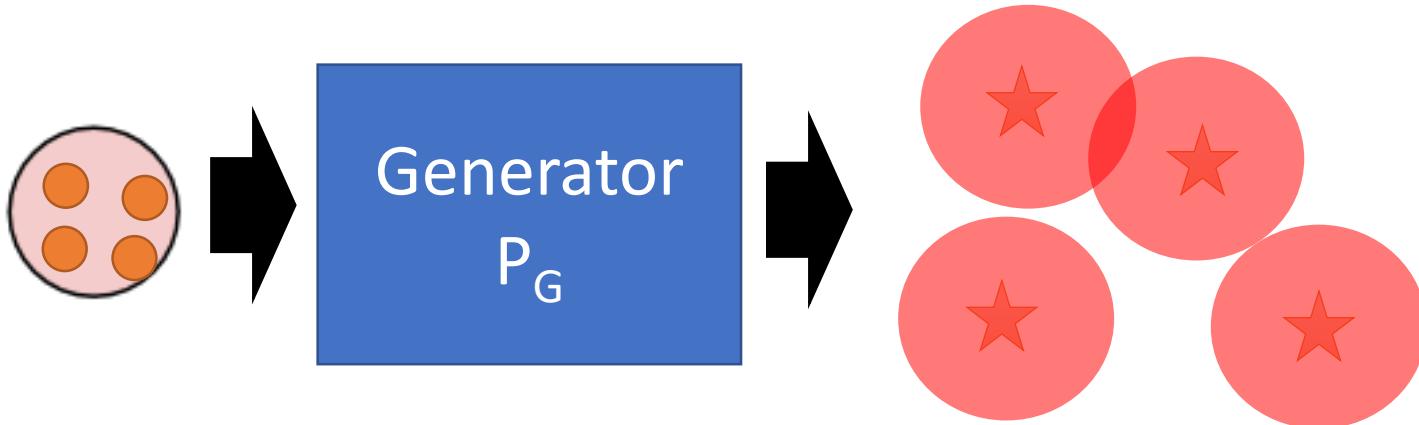
$$\text{Log Likelihood: } L = \frac{1}{N} \sum_i \log P_G(x^i)$$

We cannot compute $P_G(x^i)$. We can only sample from P_G .

Likelihood

- Kernel Density Estimation

- Estimate the distribution of $P_G(x)$ from sampling



Each sample is the mean of a Gaussian with the same covariance.

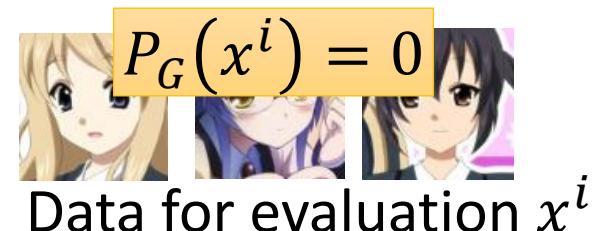
Now we have an approximation of P_G , so we can compute $P_G(x^i)$ for each real data x^i

Then we can compute the likelihood.

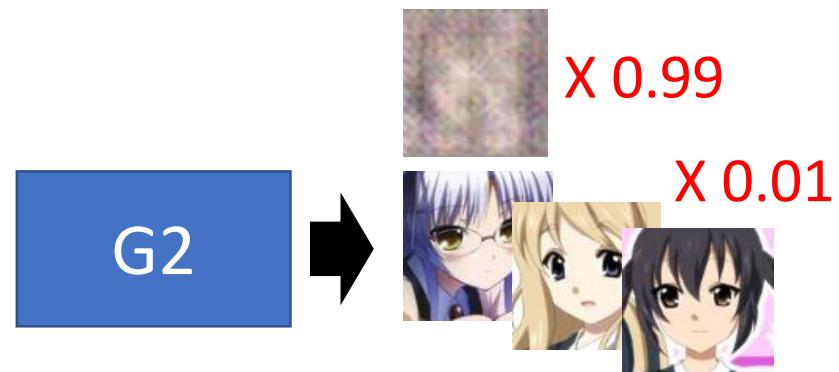
Likelihood v.s. Quality

- Low likelihood, high quality?

Considering a model generating good images (small variance)



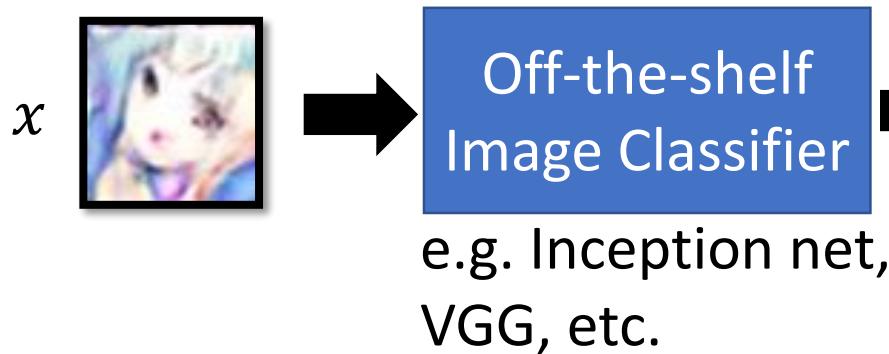
- High likelihood, low quality?



$$L = \frac{1}{N} \sum_i \log \frac{P_G(x^i)}{100} = -\log 100 + \frac{1}{N} \sum_i \log P_G(x^i)$$

4.6

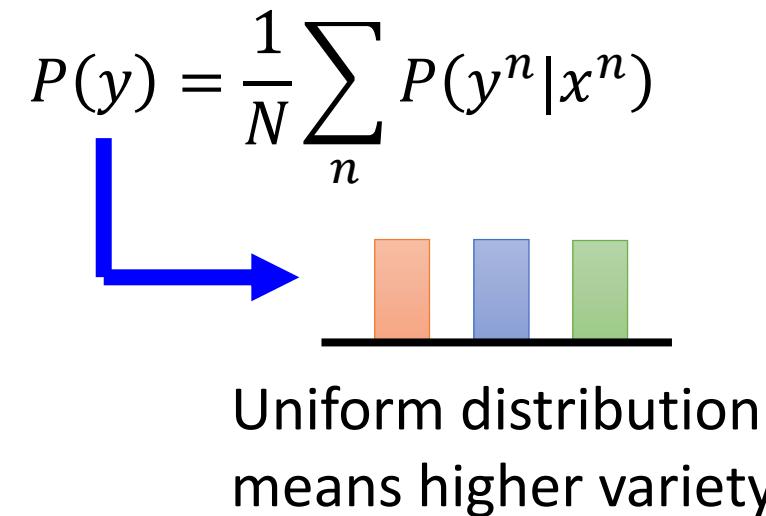
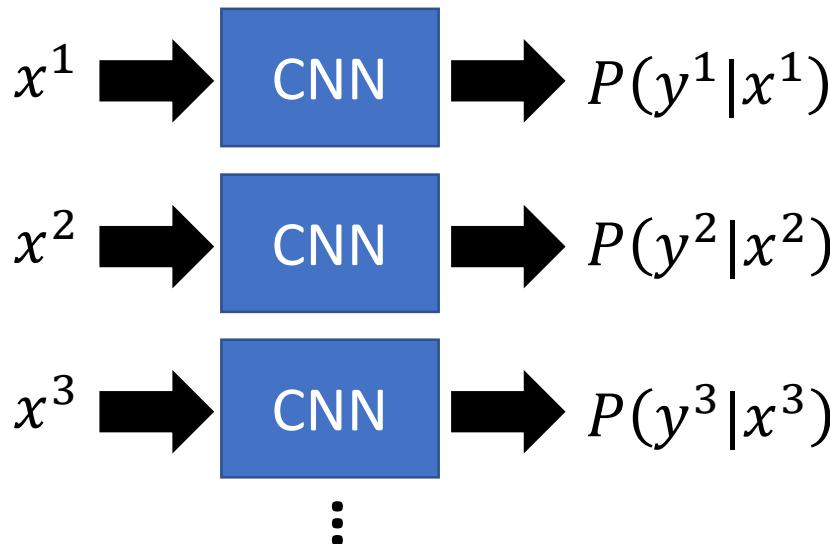
Objective Evaluation



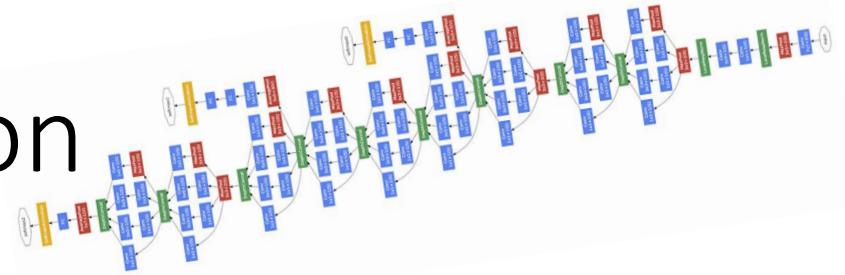
x : image
 y : class (output of CNN)



Concentrated distribution means higher visual quality



Objective Evaluation



$$P(y|x)$$

class 2



class 1

class 3

$$P(y) = \frac{1}{N} \sum_n P(y^n|x^n)$$



Inception Score

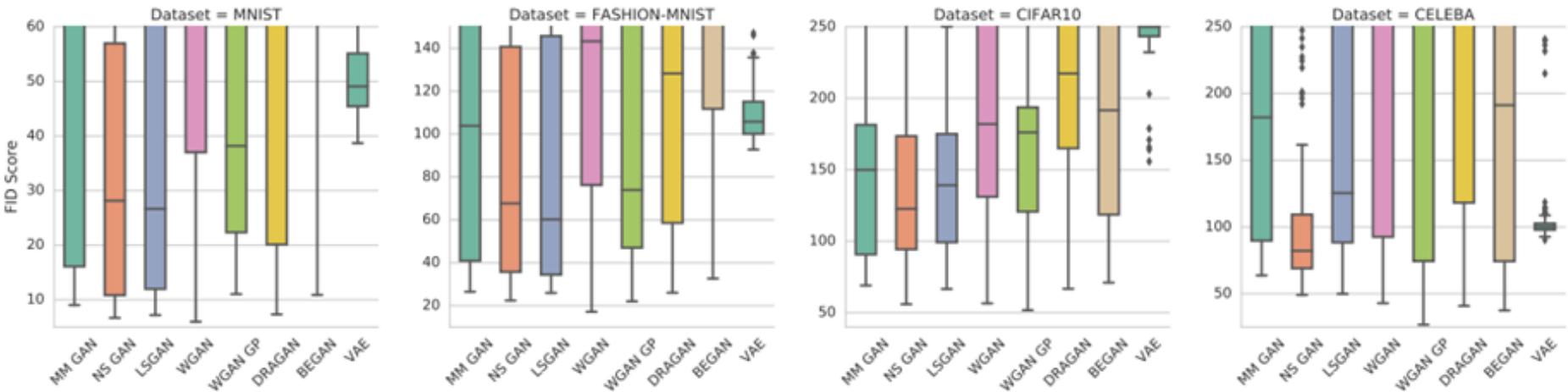
$$= \sum_x \sum_y P(y|x) \log P(y|x)$$

Negative entropy of $P(y|x)$

$$- \sum_y P(y) \log P(y)$$

Entropy of $P(y)$

GAN	DISCRIMINATOR LOSS	GENERATOR LOSS
MM GAN	$\mathcal{L}_D^{GAN} = -\mathbb{E}_{x \sim p_d} [\log(D(x))] + \mathbb{E}_{\hat{x} \sim p_g} [\log(1 - D(\hat{x}))]$	$\mathcal{L}_G^{GAN} = -\mathcal{L}_D^{GAN}$
NS GAN	$\mathcal{L}_D^{NSGAN} = \mathcal{L}_D^{GAN}$	$\mathcal{L}_G^{NSGAN} = \mathbb{E}_{\hat{x} \sim p_g} [\log(D(\hat{x}))]$
WGAN	$\mathcal{L}_D^{WGAN} = -\mathbb{E}_{x \sim p_d} [D(x)] + \mathbb{E}_{\hat{x} \sim p_g} [D(\hat{x})]$	$\mathcal{L}_G^{WGAN} = -\mathcal{L}_D^{WGAN}$
WGAN GP	$\mathcal{L}_D^{WGAN} = \mathcal{L}_D^{WGAN} + \lambda \mathbb{E}_{\hat{x} \sim p_g} [(\nabla D(\alpha x + (1 - \alpha)\hat{x}) _2 - 1)^2]$	$\mathcal{L}_G^{WGAN} = -\mathbb{E}_{\hat{x} \sim p_g} [D(\hat{x})]$
LS GAN	$\mathcal{L}_D^{LSGAN} = -\mathbb{E}_{x \sim p_d} [(D(x) - 1)^2] + \mathbb{E}_{\hat{x} \sim p_g} [D(\hat{x})^2]$	$\mathcal{L}_G^{LSGAN} = -\mathbb{E}_{\hat{x} \sim p_g} [(D(\hat{x}) - 1)^2]$
DRAGAN	$\mathcal{L}_D^{DRAGAN} = \mathcal{L}_D^{GAN} + \lambda \mathbb{E}_{\hat{x} \sim p_d + \mathcal{N}(0, c)} [(\nabla D(\hat{x}) _2 - 1)^2]$	$\mathcal{L}_G^{DRAGAN} = -\mathcal{L}_D^{NSGAN}$
BEGAN	$\mathcal{L}_D^{BEGAN} = \mathbb{E}_{x \sim p_d} [x - AE(x) _1] - k_t \mathbb{E}_{\hat{x} \sim p_g} [\hat{x} - AE(\hat{x}) _1]$	$\mathcal{L}_G^{BEGAN} = \mathbb{E}_{\hat{x} \sim p_g} [\hat{x} - AE(\hat{x}) _1]$



Smaller is better

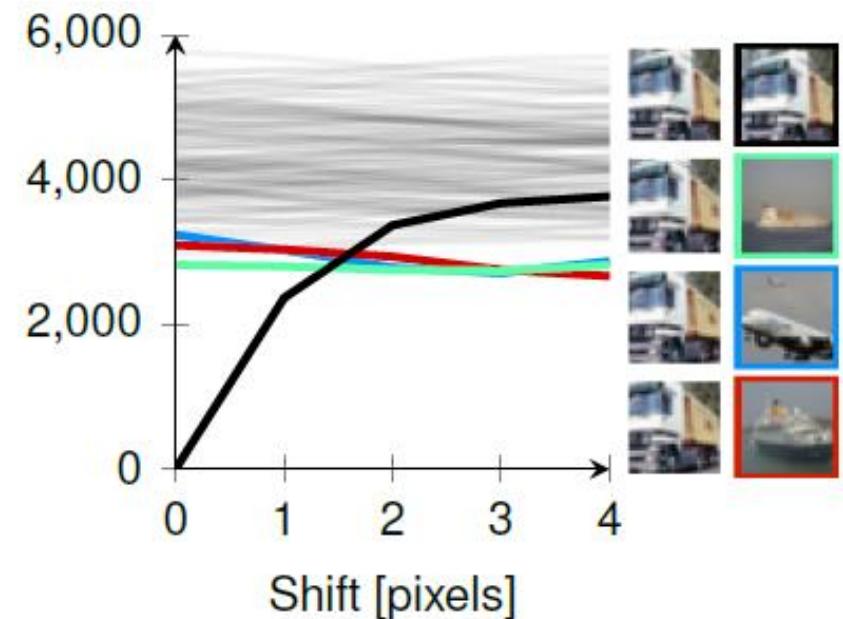
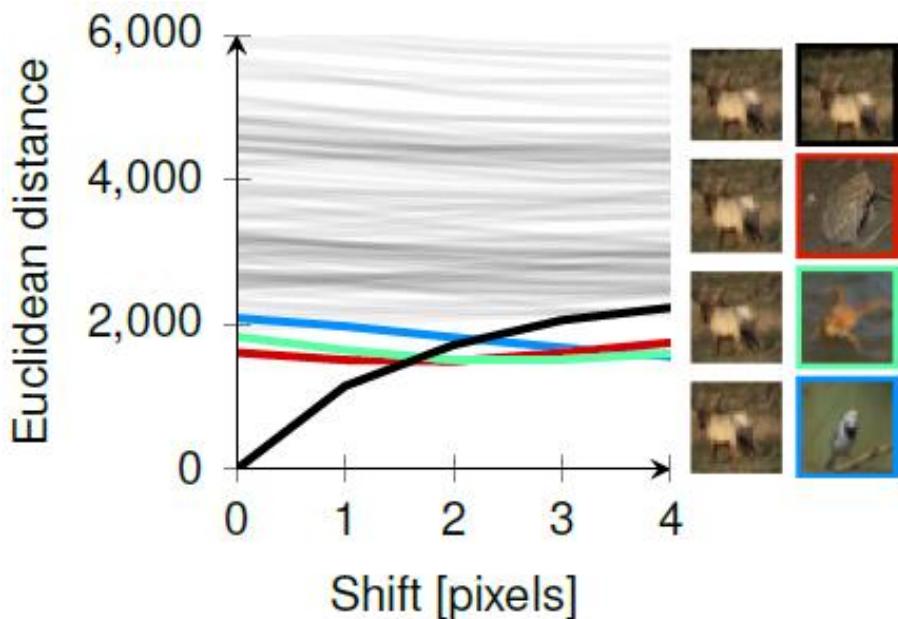
FIT:

<https://arxiv.org/pdf/1706.08500.pdf>

Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, Olivier Bousquet, "Are GANs Created Equal? A Large-Scale Study", arXiv, 2017

We don't want memory GAN.

- Using k-nearest neighbor to check whether the generator generates new objects



Missing Mode ?

Mode collapse is
easy to detect.



?



Sentence Representation

- Using RNN to encode the input text
- But you already know the testing input



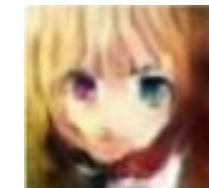
One-hot encoding

Why not 15*15
one-hot encoding?

number of data	number of data
brown hair brown eyes	8710
blonde hair blue eyes	7290
blue hair blue eyes	5590
pink hair black eyes	10
red hair black eyes	10
aqua hair gray eyes	10

Data Augmentation

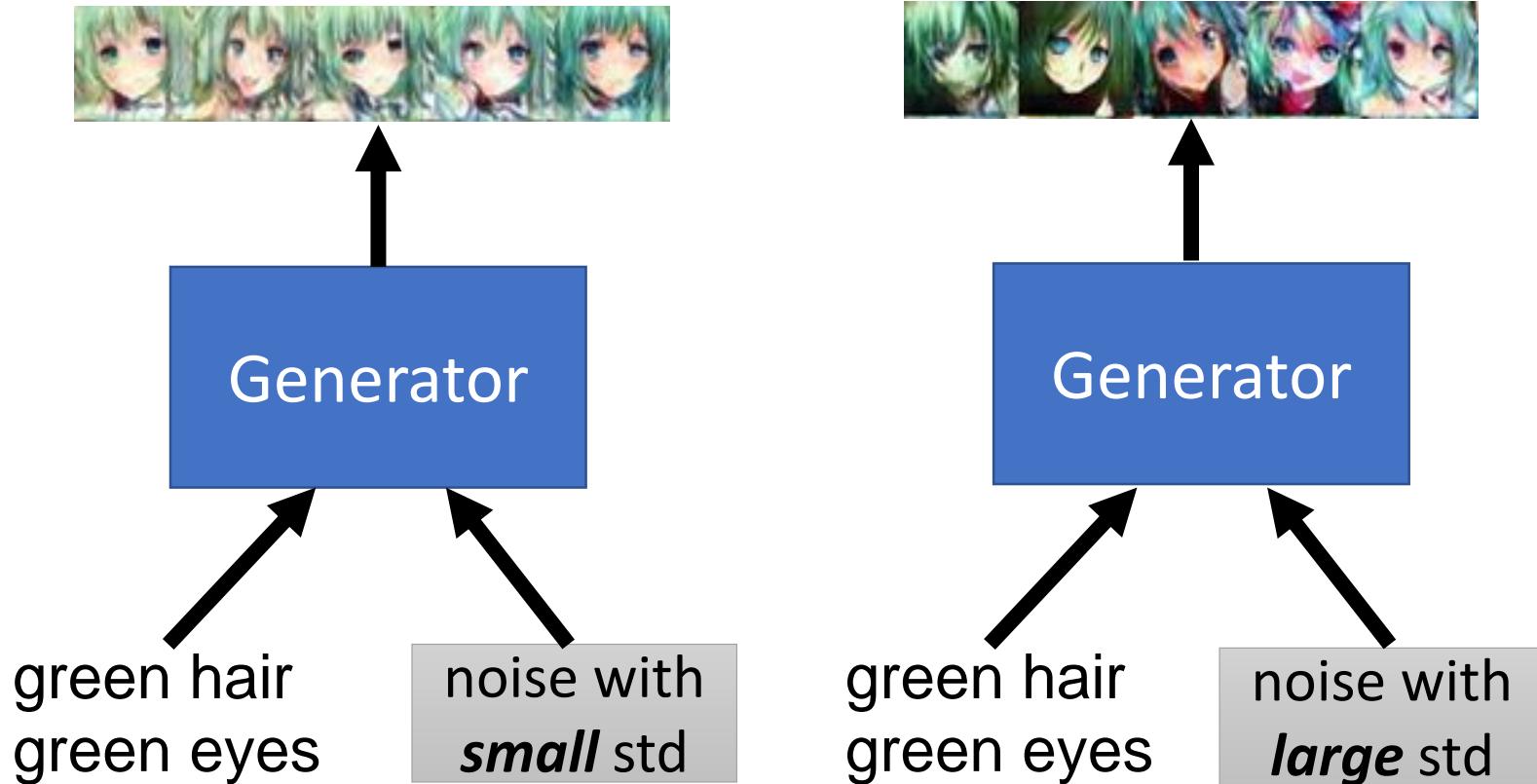
- Total training data: 33.4K
 - Only 14.8K is useful
 - Not sufficient for image generation?
- Data Augmentation
 - Horizontal Flipping + Rotation



Sampled Noise v.s. Image Quality



Sampled Noise v.s. Image Quality



Ensemble

