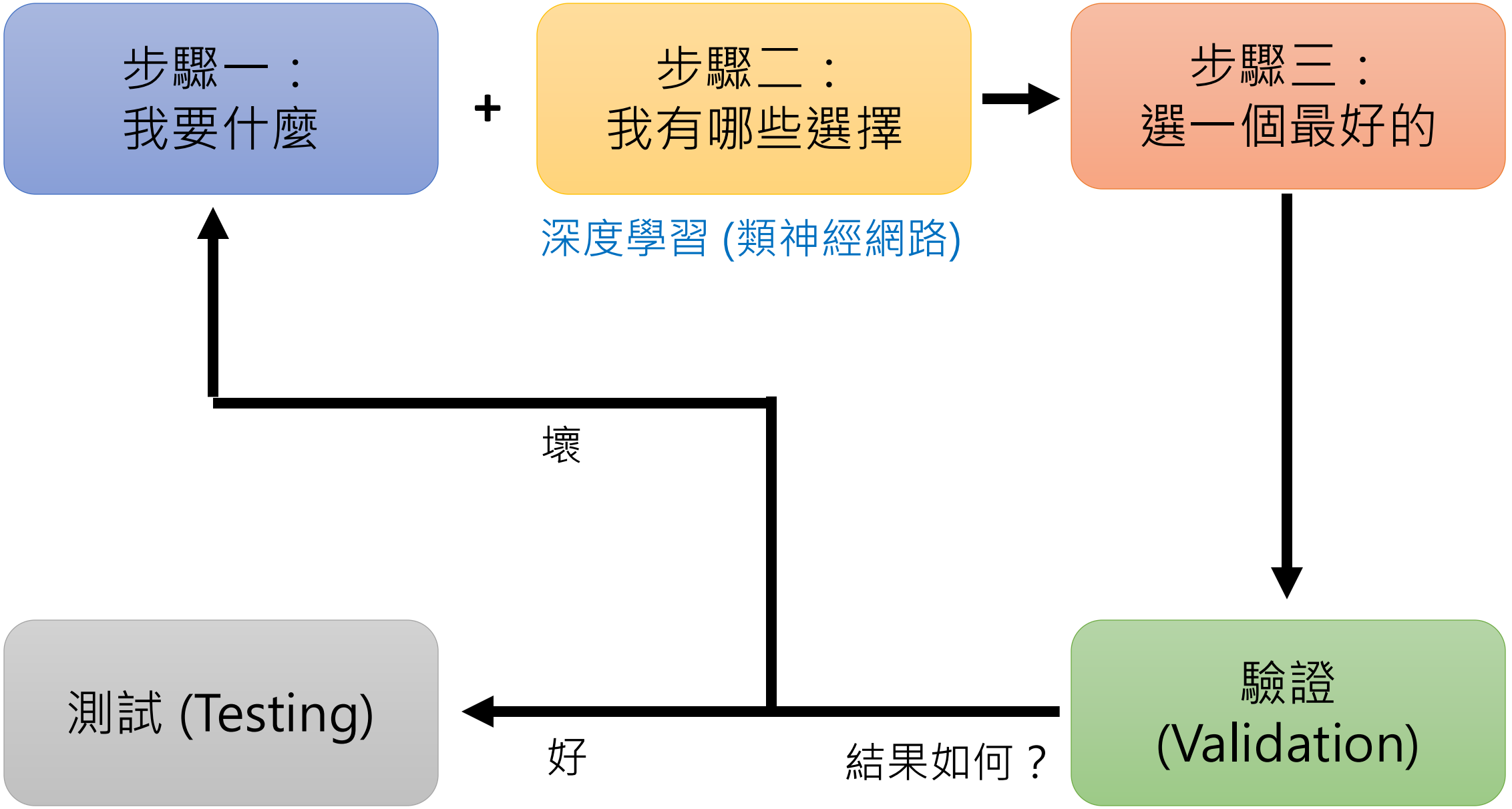
一堂課看懂訓練類神經網路的各種訣竅

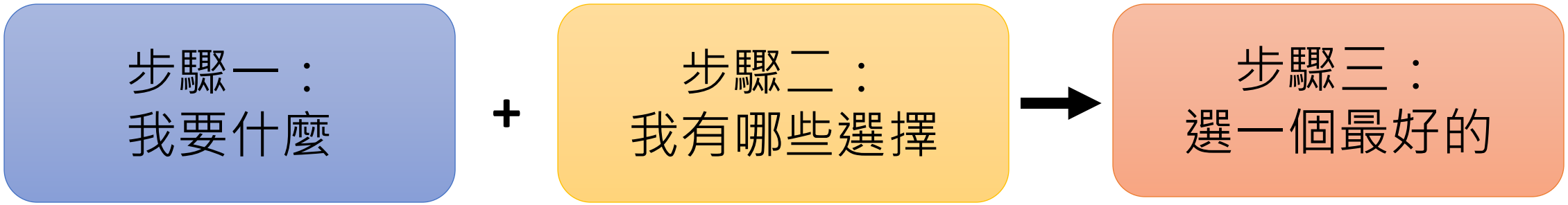
李宏毅


步驟一：
我要什麼
+
步驟二：
我有哪些選擇
步驟三：
選一個最好的
深度學習 (類神經網路)
壞
測試 (Testing)
好
結果如何？
驗證
(Validation)

介紹各種訓練類神經網路常用技巧

- 聽到一個跟訓練類神經網路有關的方法時，你要這樣問自己

方法名	改了那一個步驟	帶來什麼好處
……	……	……



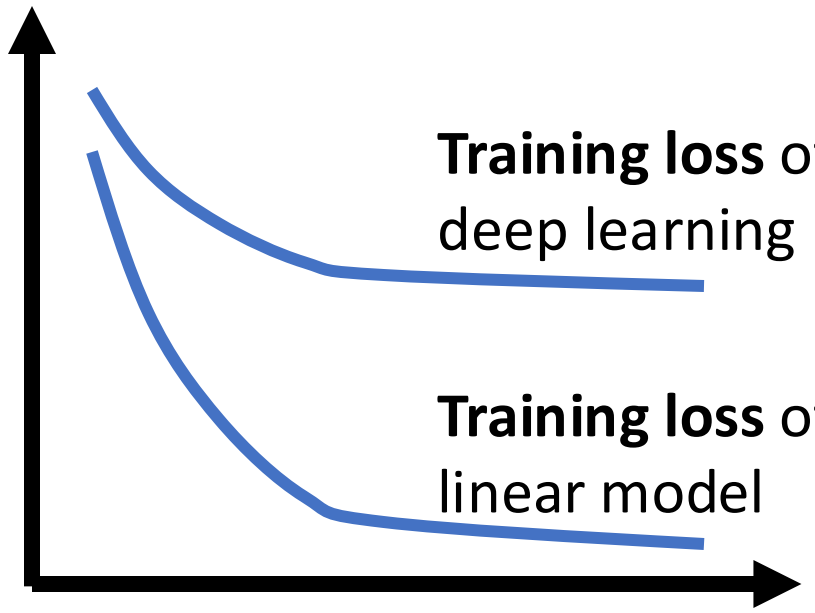
介紹各種訓練類神經網路常用技巧

- 聽到一個跟訓練類神經網路有關的方法時，你要這樣問自己

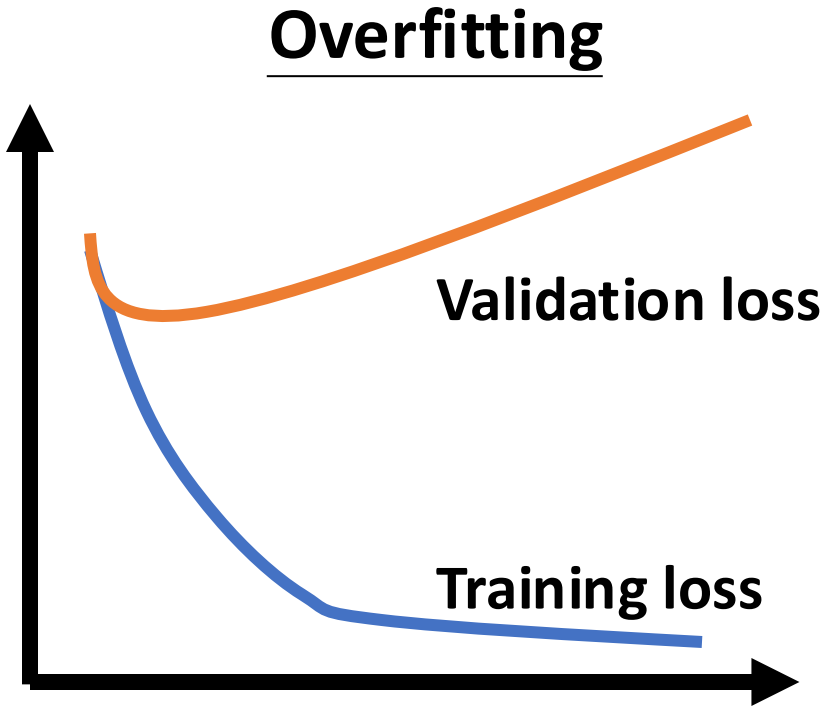
方法名	改了那一個步驟	帶來什麼好處
……	……	……

  - Better Optimization：更低的 Training Loss
  - Better Generalization：更低的 Validation Loss
  - ……

選擇合適的技巧

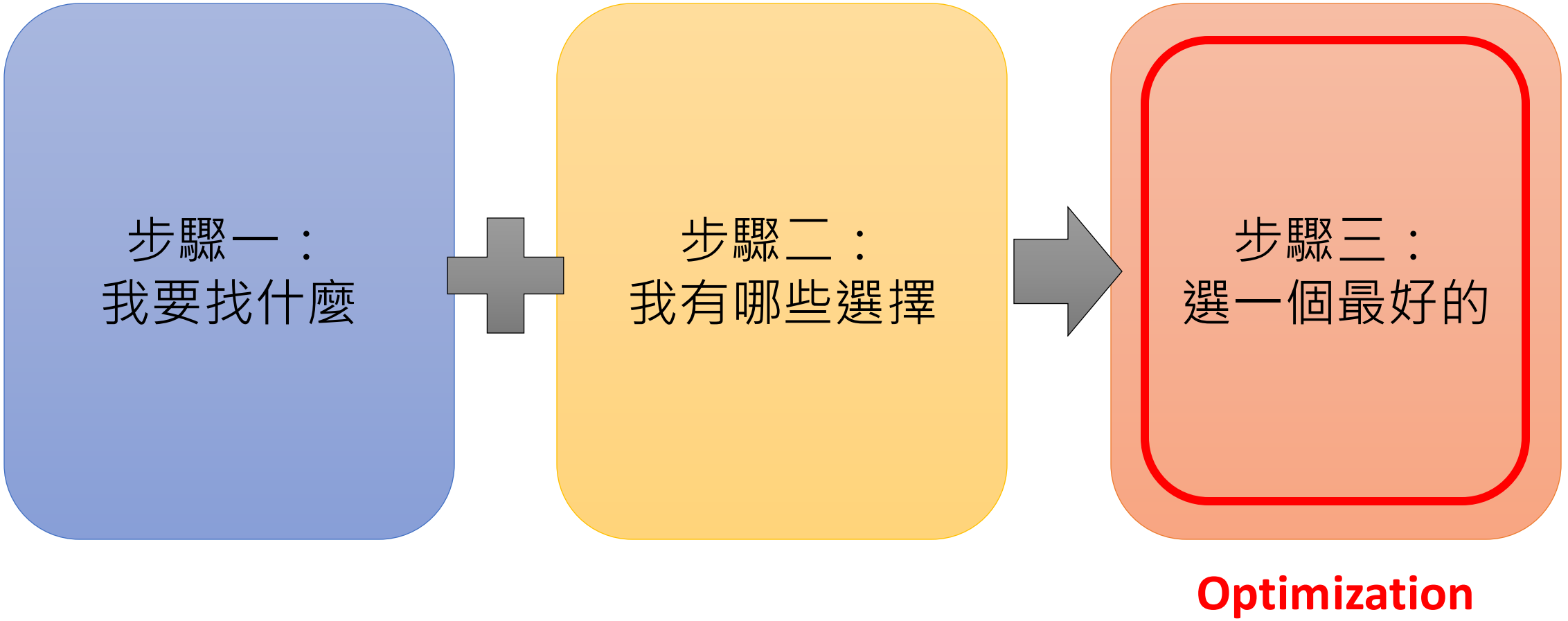


Better Optimization!



Better Generalization!

找函式步驟 3 + 1



Vanilla Gradient Descent

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

- (Randomly) Pick initial values $\boldsymbol{\theta^0}$
- Compute gradient $\boldsymbol{g^0} = \nabla L(\boldsymbol{\theta^0})$

$$\boldsymbol{\theta^1} \leftarrow \boldsymbol{\theta^0} - \eta \boldsymbol{g^0}$$

- Compute gradient $\boldsymbol{g^1} = \nabla L(\boldsymbol{\theta^1})$

$$\boldsymbol{\theta^2} \leftarrow \boldsymbol{\theta^1} - \eta \boldsymbol{g^1}$$

- Compute gradient $\boldsymbol{g^2} = \nabla L(\boldsymbol{\theta^2})$

$$\boldsymbol{\theta^3} \leftarrow \boldsymbol{\theta^2} - \eta \boldsymbol{g^2}$$

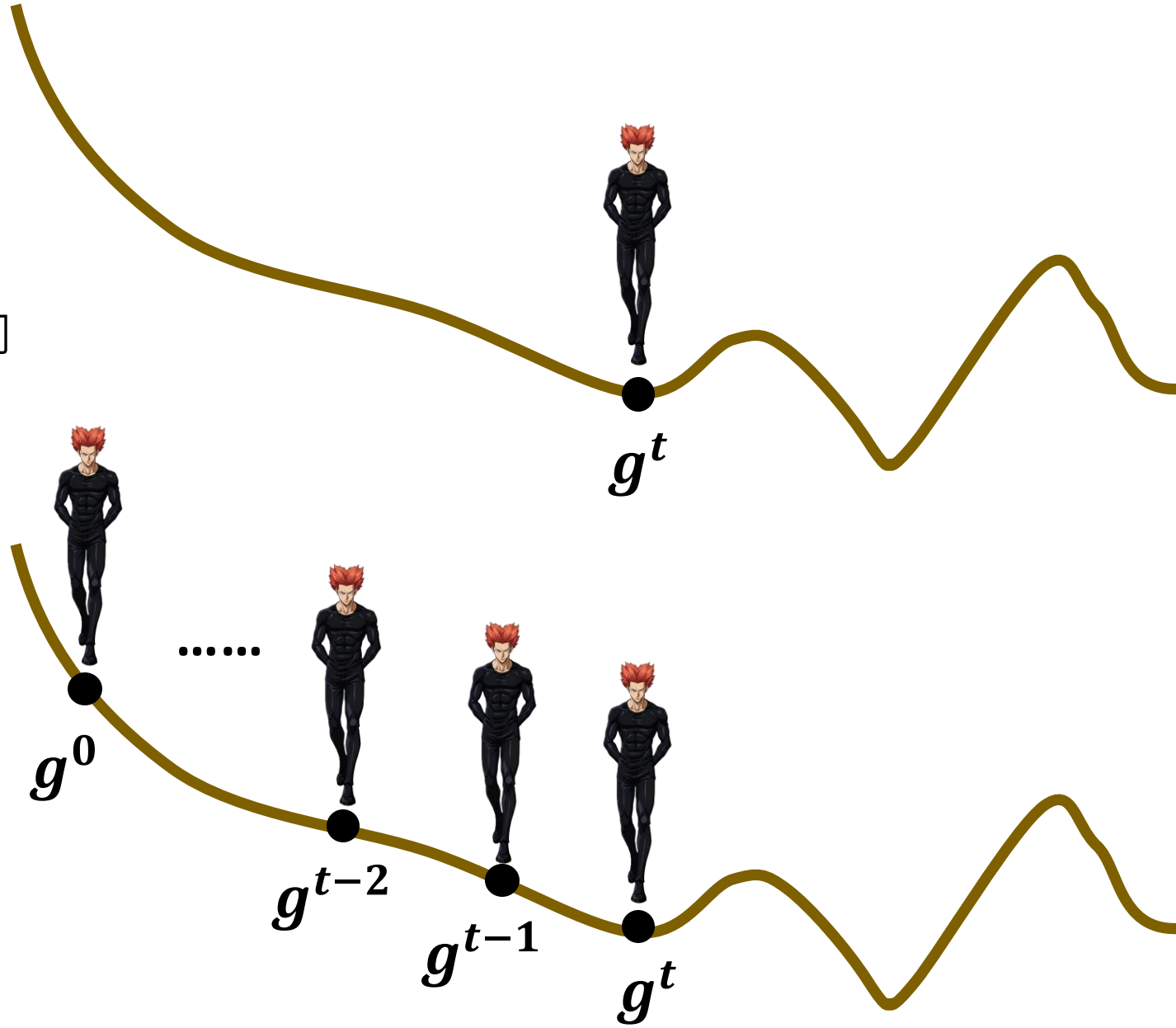
Optimizer

Vanilla Gradient Descent

根據當下算出來的 $\boldsymbol{g^t}$ 來決定方向

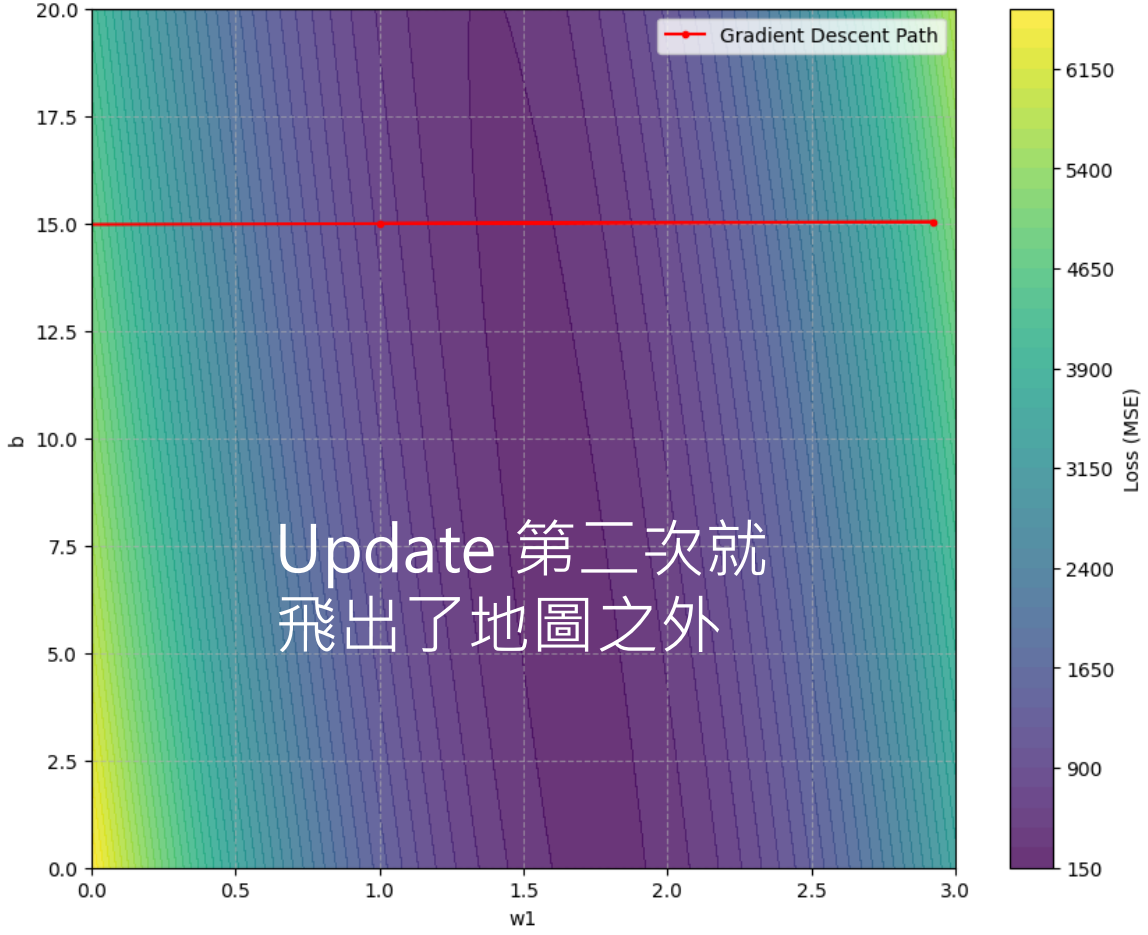
Gradient Descent + Optimizer

根據 $\boldsymbol{g^0, g^1, g^2, \ldots, g^t}$
一起來決定方向

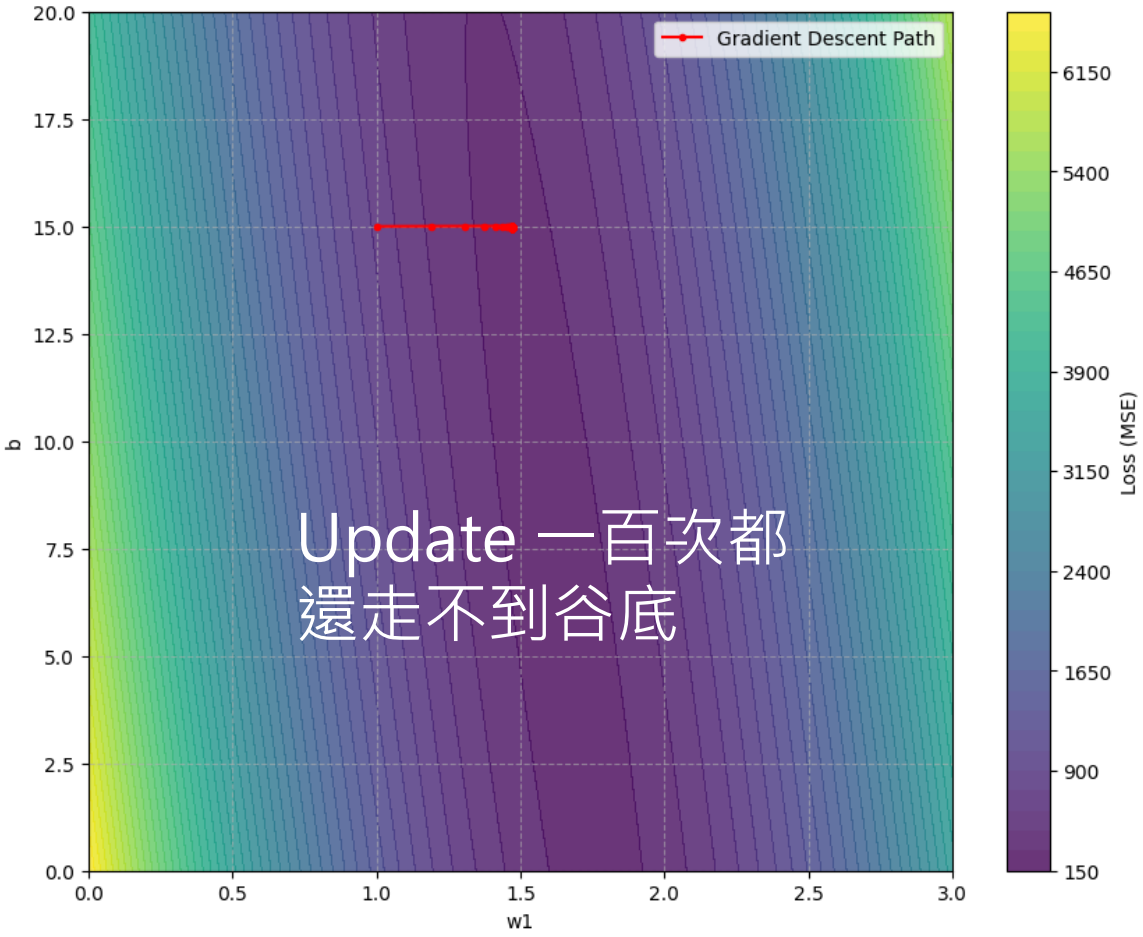


Learning Rate 實在是很難調 ...

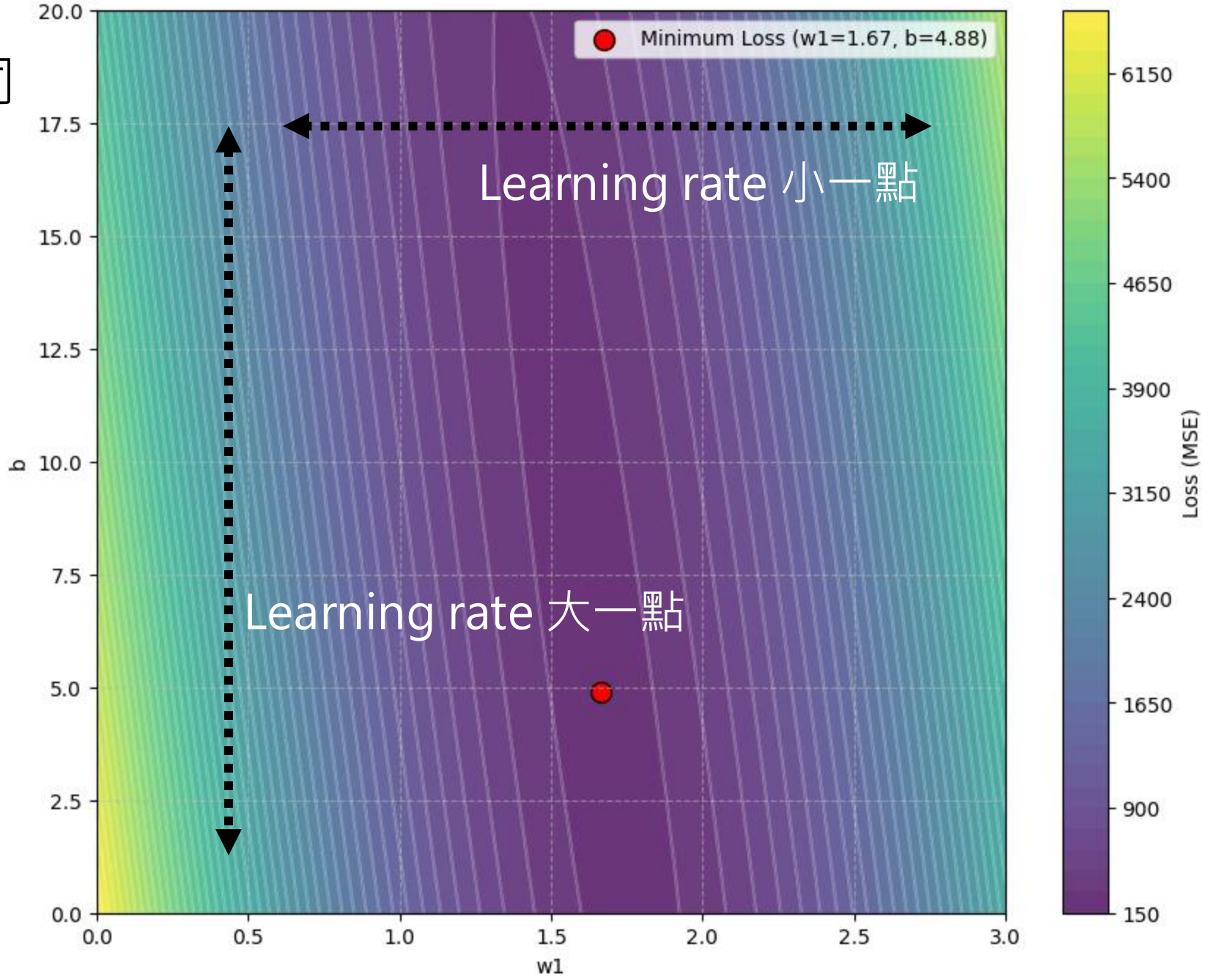
$\eta = 0.001$



$\eta = 0.0001$

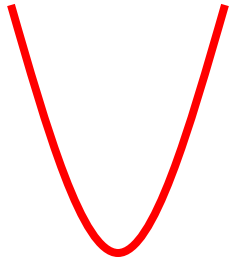


不同參數應該要有不同的 Learning Rate



怎麼知道那一個方向的 gradient 大、那一個小？

用過去的 Gradient 來決定 Learning Rate



Learning rate 小一點

$$g^0 = \begin{bmatrix} 500 \\ 0.4 \end{bmatrix} \quad g^1 = \begin{bmatrix} 432 \\ -0.3 \end{bmatrix} \quad g^2 = \begin{bmatrix} -211 \\ 0.2 \end{bmatrix} \quad g^3 = \begin{bmatrix} 139 \\ 0.1 \end{bmatrix}$$

Learning rate 大一點

Adagrad

- Compute gradient $\boldsymbol{g^0} = \nabla L(\boldsymbol{\theta^0})$

  For each dimension i: $\sigma_i^0 = \sqrt{(g_i^0)^2} = |g_i^0|$

  $\theta_i^1 \leftarrow \theta_i^0 - \frac{\eta}{\sigma_i^0} g_i^0$

- Compute gradient $g^1 = \nabla L(\boldsymbol{\theta}^1)$

  Average?

  For each dimension i: $\sigma_i^1 = \sqrt{\left[(g_i^0)^2 + (g_i^1)^2\right]}$

  $\theta_i^2 \leftarrow \theta_i^1 - \frac{\eta}{\sigma_i^1} g_i^1$

- Compute gradient $g^2 = \nabla L(\boldsymbol{\theta}^2)$

  For each dimension i: $\sigma_i^2 = \sqrt{\left[(g_i^0)^2 + (g_i^1)^2 + (g_i^2)^2\right]}$

  $\theta_i^3 \leftarrow \theta_i^2 - \frac{\eta}{\sigma_i^2} g_i^2$

⋮

- Compute gradient $\boldsymbol{g^t} = \nabla L(\boldsymbol{\theta}^t)$

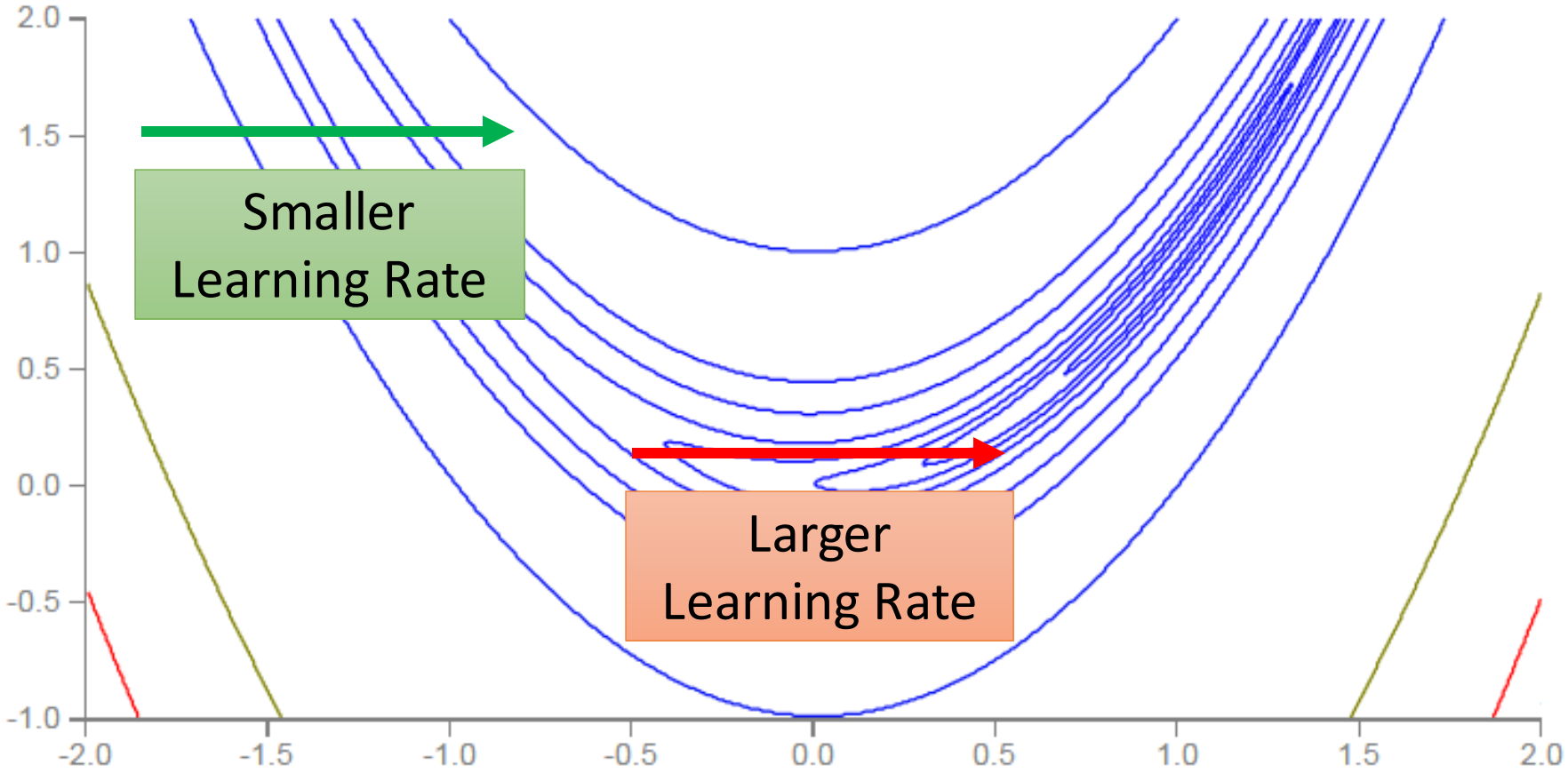
  For each dimension i: $\sigma_i^t = \sqrt{\sum_{i=0}^{t} (g_i^t)^2}$

  $\theta_i^{t+1} \leftarrow \theta_i^t - \frac{\eta}{\sigma_i^t} g_i^t$

範例程式

連結：
https://colab.research.google.com/drive/1XPlU-I77dXL9W74jnevmfb8K8XoEPRso?usp=sharing

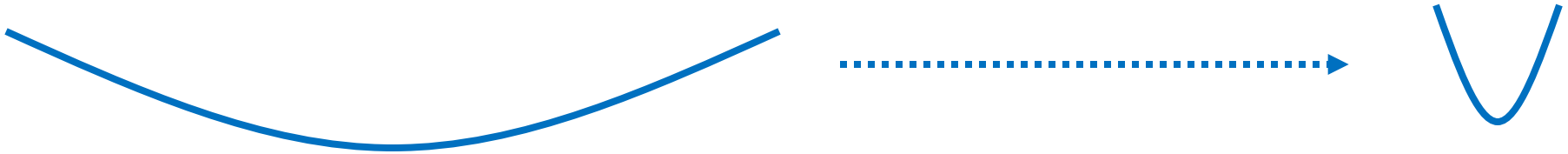
同一個參數的Gradient 大小不會一成不變



同一個參數的Gradient 大小不會一成不變

$$g^0 = \begin{bmatrix} 500 \\ 0.4 \end{bmatrix} \quad g^1 = \begin{bmatrix} 432 \\ -0.3 \end{bmatrix} \quad \cdots\cdots \quad g^{t-1} = \begin{bmatrix} -0.1 \\ 229 \end{bmatrix} \quad g^t = \begin{bmatrix} 0.1 \\ 100 \end{bmatrix}$$

變大　　變大



Adagrad: 全部平方加起來

RMSProp: 最近算出來的 gradient 給比較大的影響

RMSProp

- Compute gradient $\boldsymbol{g^0} = \nabla L(\boldsymbol{\theta^0})$

  For each dimension i: $\sigma_i^0 = \sqrt{(g_i^0)^2}$

  $\theta_i^1 \leftarrow \theta_i^0 - \frac{\eta}{\sigma_i^0} g_i^0$

- Compute gradient $\boldsymbol{g^1} = \nabla L(\boldsymbol{\theta^1})$ $\quad 0 < \alpha < 1$

  For each dimension i: $\sigma_i^1 = \sqrt{\alpha(\sigma_i^0)^2 + (1-\alpha)(g_i^1)^2}$

  $\theta_i^2 \leftarrow \theta_i^1 - \frac{\eta}{\sigma_i^1} g_i^1$

- Compute gradient $\boldsymbol{g^2} = \nabla L(\boldsymbol{\theta^2})$

  For each dimension i: $\sigma_i^2 = \sqrt{\alpha(\sigma_i^1)^2 + (1-\alpha)(g_i^2)^2}$

  $\theta_i^3 \leftarrow \theta_i^2 - \frac{\eta}{\sigma_i^2} g_i^2$

⋮

- Compute gradient $\boldsymbol{g^t} = \nabla L(\boldsymbol{\theta^t})$

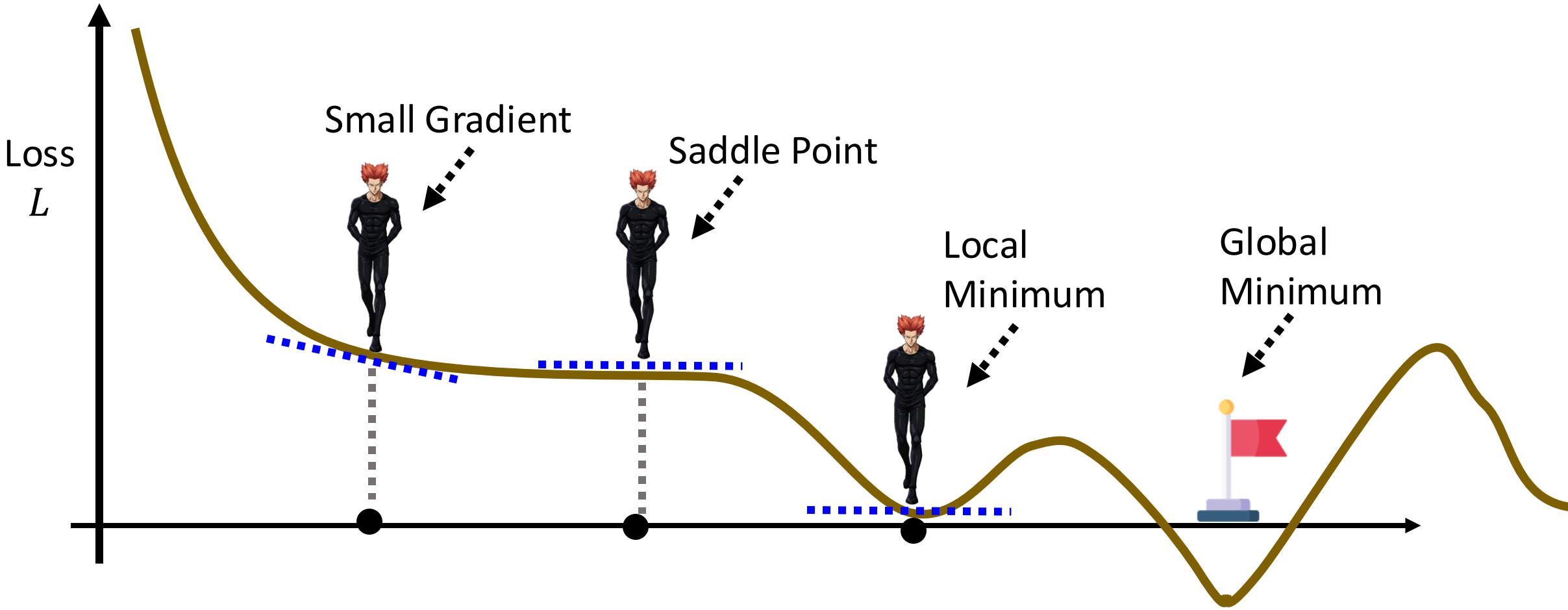
  For each dimension i: $\sigma_i^t = \sqrt{\alpha(\sigma_i^{t-1})^2 + (1-\alpha)(g_i^t)^2}$

  $\theta_i^{t+1} \leftarrow \theta_i^t - \frac{\eta}{\sigma_i^t} g_i^t$

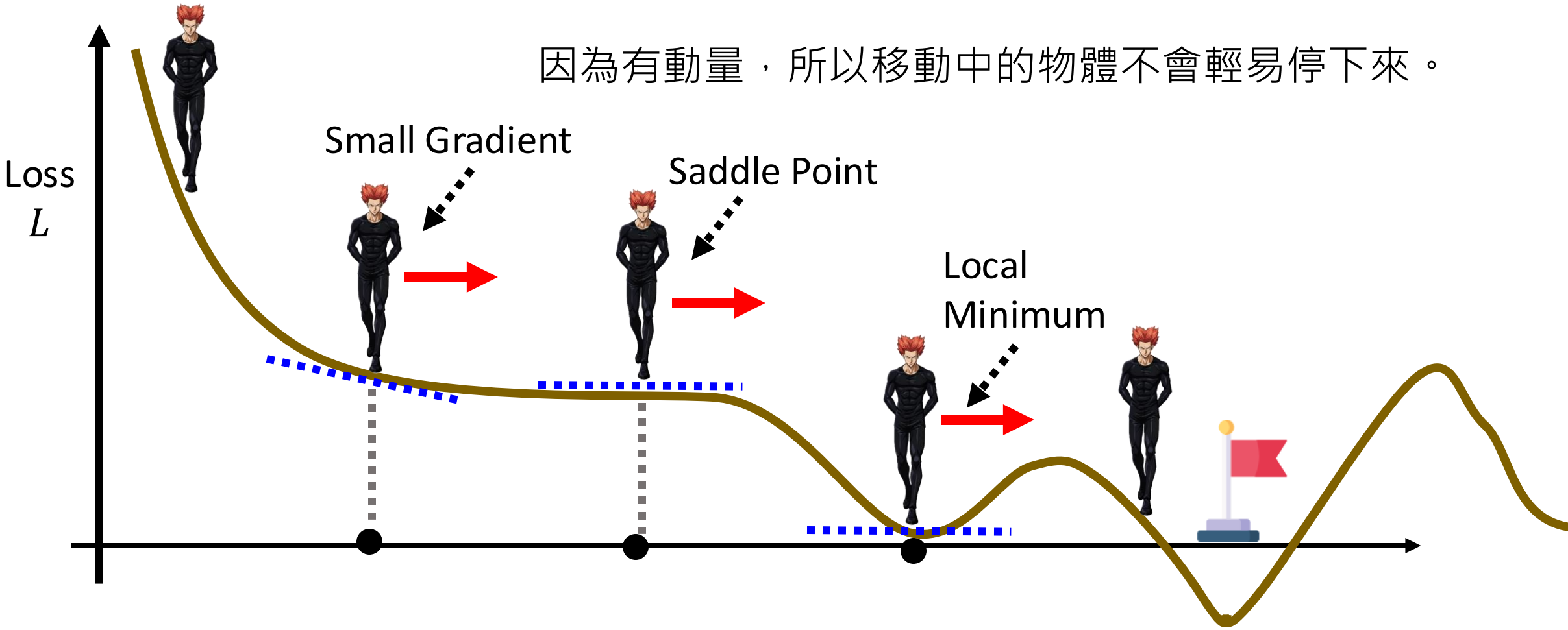
範例程式

連結：
https://colab.research.google.com/drive/1XPlU-I77dXL9W74jnevmfb8K8XoEPRso?usp=sharing

Optimization 會在 Gradient 很小時停止



考慮動量 (Momentum)



Momentum

- Compute gradient $\boldsymbol{g^0} = \nabla L(\boldsymbol{\theta^0})$

  For each dimension i: $m_i^0 = g_i^0$ $\quad \theta_i^1 \leftarrow \theta_i^0 - \eta m_i^0$

- Compute gradient $\boldsymbol{g^1} = \nabla L(\boldsymbol{\theta^1})$

  For each dimension i: $m_i^1 = g_i^0 + g_i^1$ $\quad \theta_i^2 \leftarrow \theta_i^1 - \eta m_i^1$

- Compute gradient $\boldsymbol{g^2} = \nabla L(\boldsymbol{\theta^2})$

  For each dimension i: $m_i^2 = g_i^0 + g_i^1 + g_i^2$ $\quad \theta_i^3 \leftarrow \theta_i^2 - \eta m_i^2$

⋮

- Compute gradient $\boldsymbol{g^t} = \nabla L(\boldsymbol{\theta^t})$

  For each dimension i: $m_i^t = g_i^0 + g_i^1 + g_i^2 + \cdots + g_i^t$ $\quad \theta_i^{t+1} \leftarrow \theta_i^t - \eta m_i^t$

Momentum (這不是經典的 Momentum)

- Compute gradient $\boldsymbol{g^0} = \nabla L(\boldsymbol{\theta^0})$

  For each dimension i: $m_i^0 = g_i^0$ $\quad \theta_i^1 \leftarrow \theta_i^0 - \eta m_i^0$

- Compute gradient $\boldsymbol{g^1} = \nabla L(\boldsymbol{\theta^1})$ $\quad 0 < \beta < 1$

  For each dimension i: $m_i^1 = \beta m_i^0 + (1-\beta) g_i^1$ $\quad \theta_i^2 \leftarrow \theta_i^1 - \eta m_i^1$

- Compute gradient $\boldsymbol{g^2} = \nabla L(\boldsymbol{\theta^2})$

  For each dimension i: $m_i^2 = \beta m_i^1 + (1-\beta) g_i^2$ $\quad \theta_i^3 \leftarrow \theta_i^2 - \eta m_i^2$

⋮

- Compute gradient $\boldsymbol{g^t} = \nabla L(\boldsymbol{\theta^t})$

  For each dimension i: $m_i^t = \beta m_i^{t-1} + (1-\beta) g_i^t$ $\quad \theta_i^{t+1} \leftarrow \theta_i^t - \eta m_i^t$

範例程式

連結：
https://colab.research.google.com/drive/1XPlU-I77dXL9W74jnevmfb8K8XoEPRso?usp=sharing

Adam: RMSProp + Momentum

- Compute gradient $\boldsymbol{g}^t = \nabla L(\boldsymbol{\theta}^t)$

Momentum

For each dimension i: $m_i^t = \beta m_i^{t-1} + (1-\beta) g_i^t$

$\theta_i^{t+1} \leftarrow \theta_i^t - \eta m_i^t$

RMSProp

For each dimension i: $\sigma_i^t = \sqrt{\alpha\left(\sigma_i^{t-1}\right)^2 + (1-\alpha)\left(g_i^t\right)^2}$

$\theta_i^{t+1} \leftarrow \theta_i^t - \frac{\eta}{\sigma_i^t} g_i^t$

Adam

$\theta_i^{t+1} \leftarrow \theta_i^t - \frac{\eta}{\sigma_i^t} m_i^t$

Adam has bias-corrected terms, which are omitted here for simplicity.

範例程式

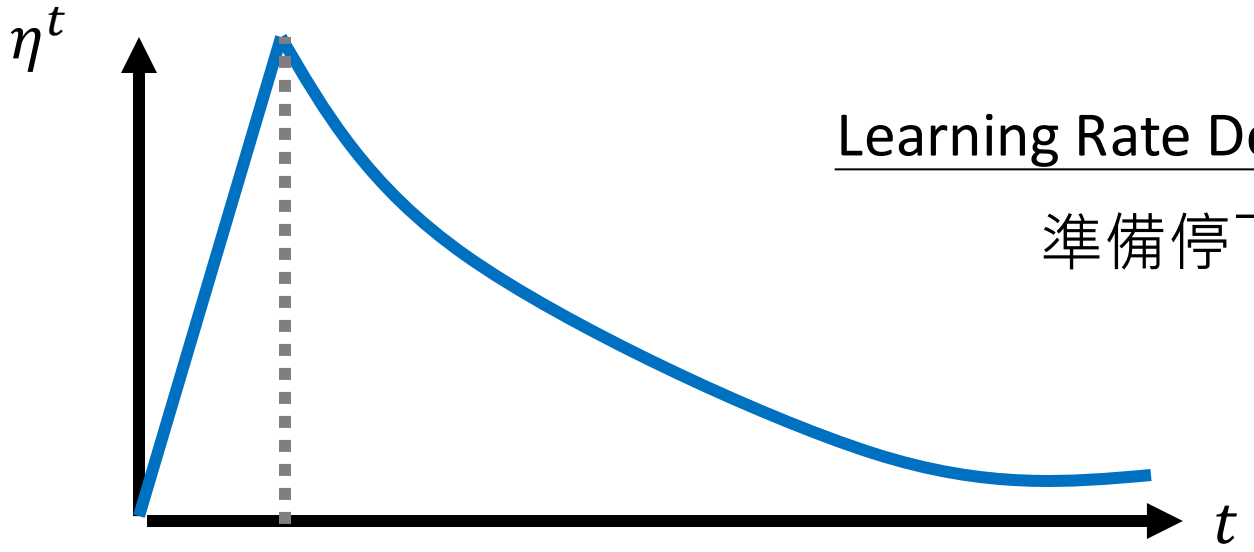
連結：
https://colab.research.google.com/drive/1XPlU-I77dXL9W74jnevmfb8K8XoEPRso?usp=sharing

Learning Rate Scheduling

$$\theta_i^{t+1} \leftarrow \theta_i^t - \frac{\eta}{\sigma_i^t} m_i^t \quad \Rightarrow \quad \theta_i^{t+1} \leftarrow \theta_i^t - \frac{\eta^t}{\sigma_i^t} m_i^t$$

Warm Up

探索地形



Learning Rate Decay

準備停下來

Summarization of Optimizer

$$\theta_i^{t+1} \leftarrow \theta_i^t - \frac{\eta^t}{\sigma_i^t} m_i^t$$

Learning rate scheduling

Momentum: sum of the previous gradients

Consider direction

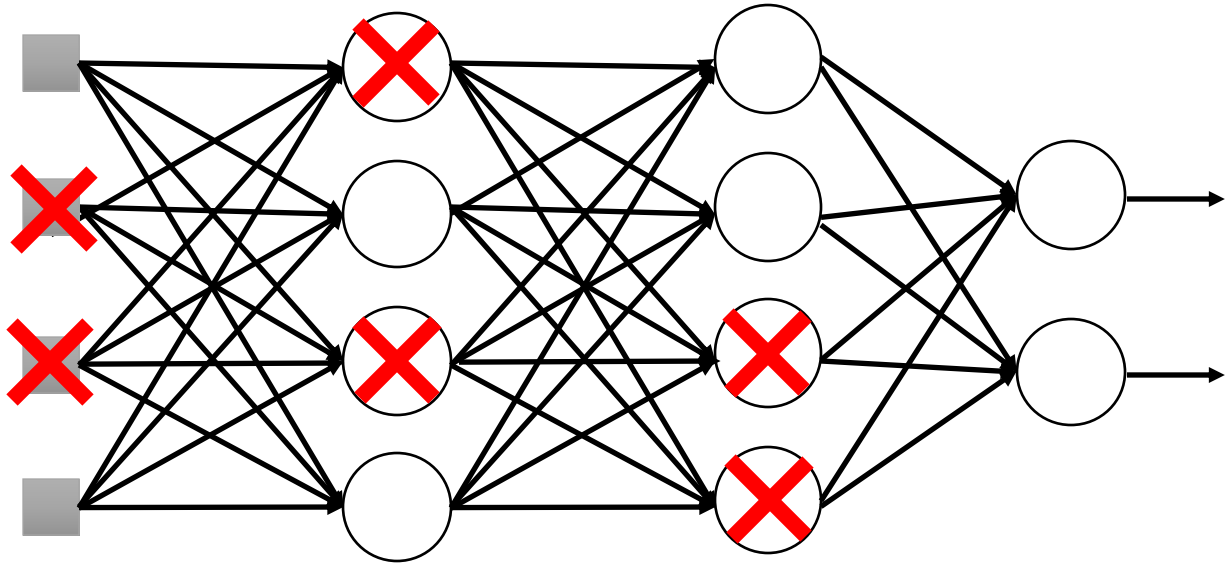
square of the gradients

only magnitude

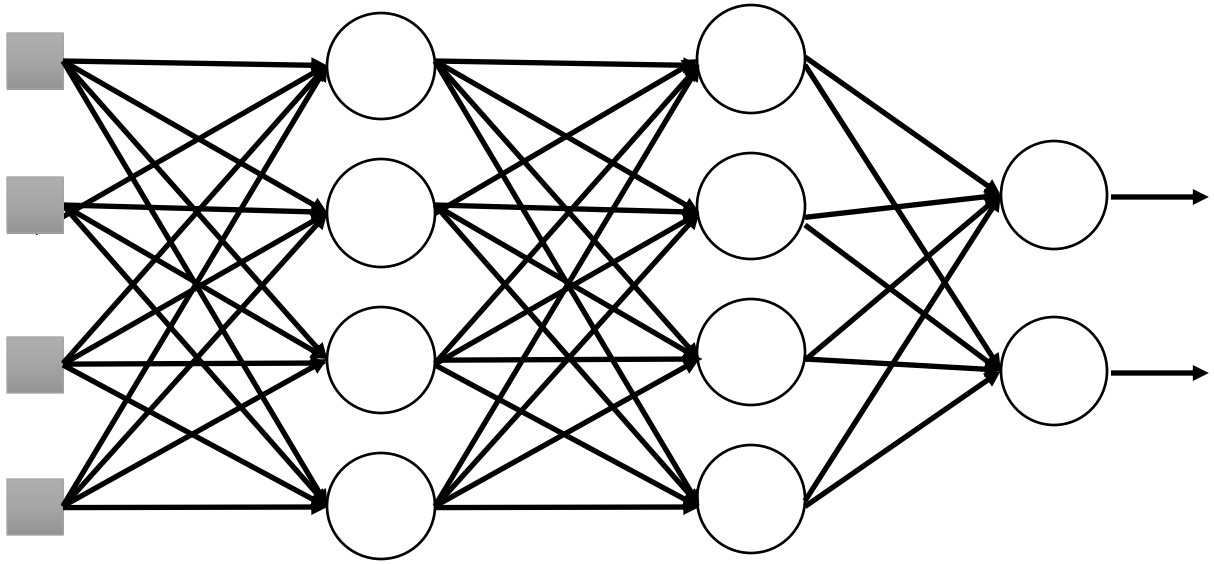
方法名	改了那一個步驟	帶來什麼好處
Adagrad, RMSProp, Momentum, Adam, etc.	找最好的函式	Better Optimization (not for Generalization)

Dropout

訓練時隨機丟掉
一些神經元

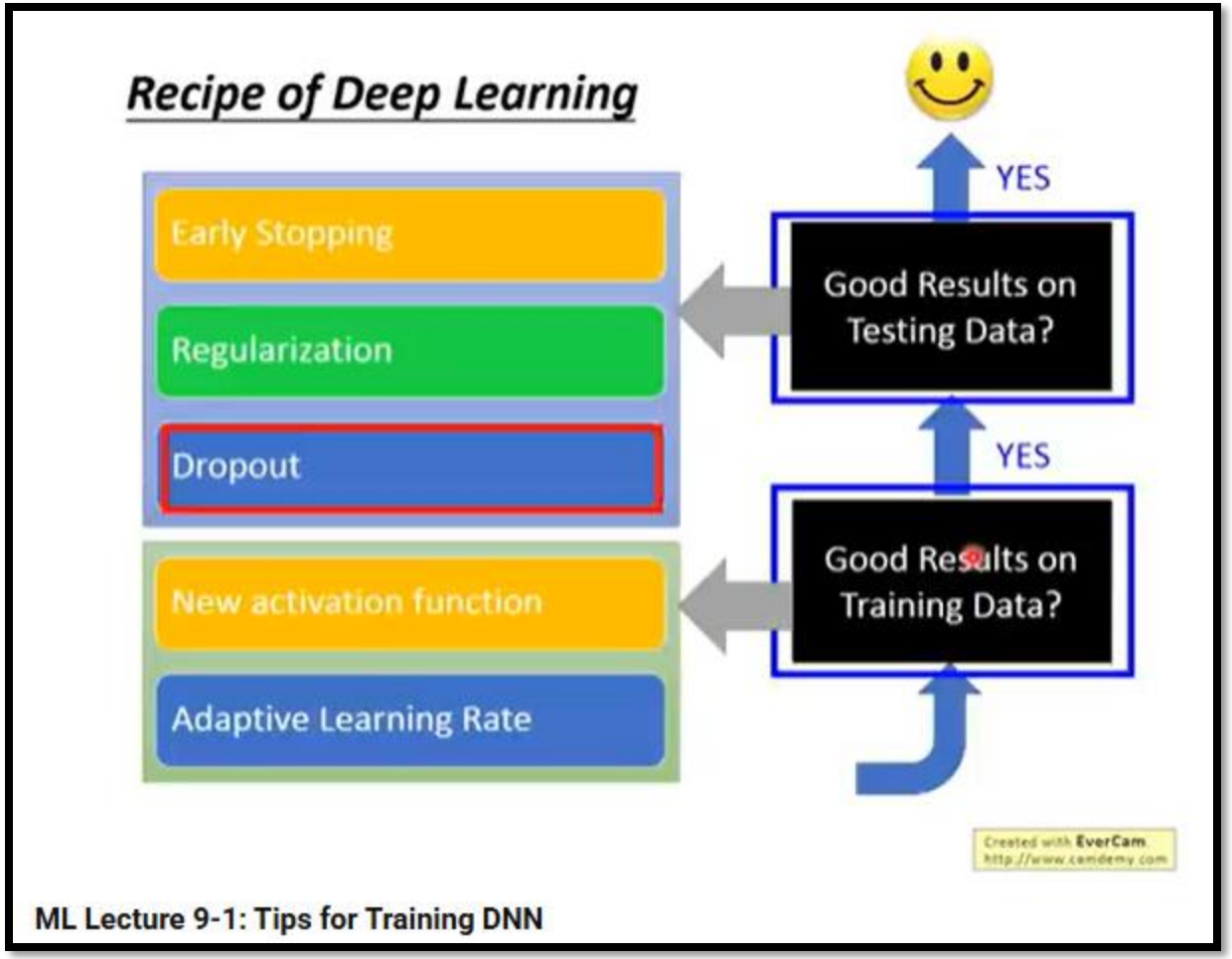


驗證和測試時
火力全開



方法名	改了那一個步驟	帶來什麼好處
Dropout	找最好的函式	Better Generalization (Worse Optimization)

注意 Dropout 的選擇時機!

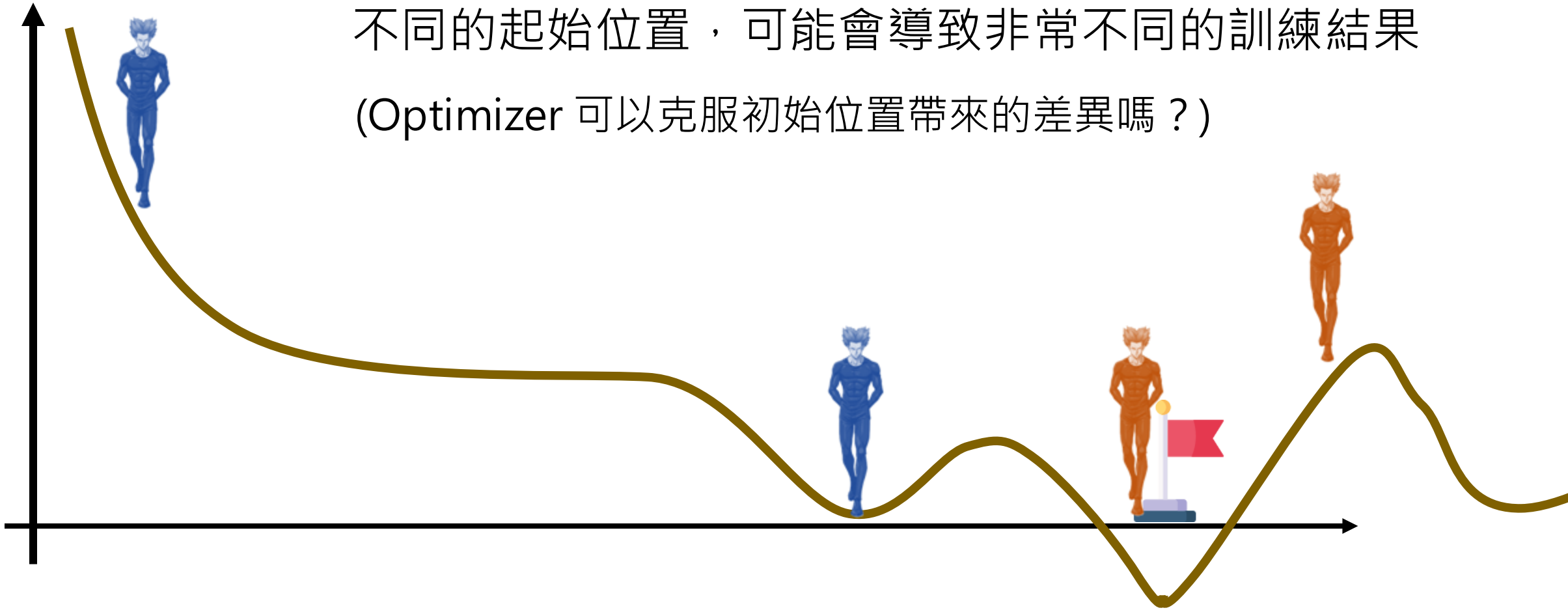


https://youtu.be/xki61j7z-30?si=EoF1VjcP3_L5UWI3&t=4227

Initialization

不同的起始位置，可能會導致非常不同的訓練結果

(Optimizer 可以克服初始位置帶來的差異嗎？)

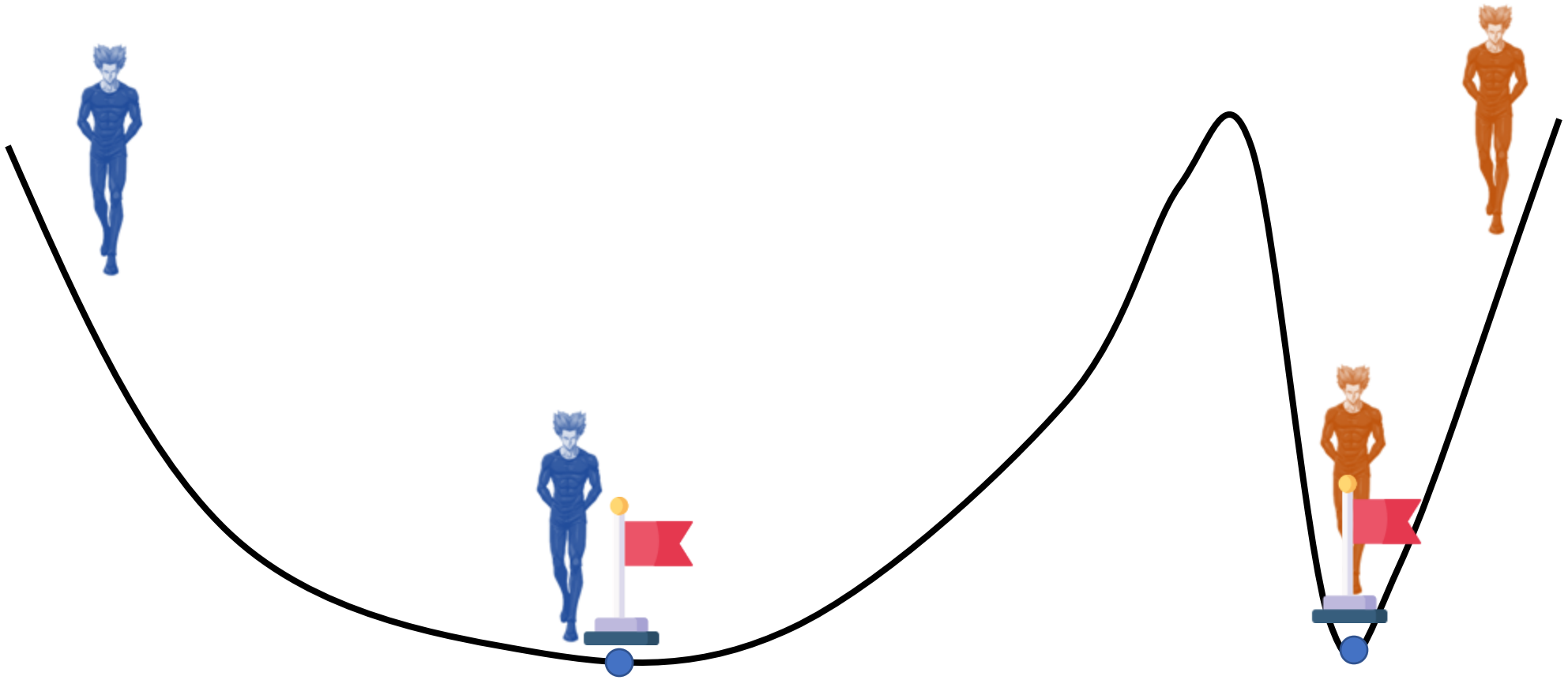


範例程式

連結：
https://colab.research.google.com/drive/1XPlU-I77dXL9W74jnevmfb8K8XoEPRso?usp=sharing

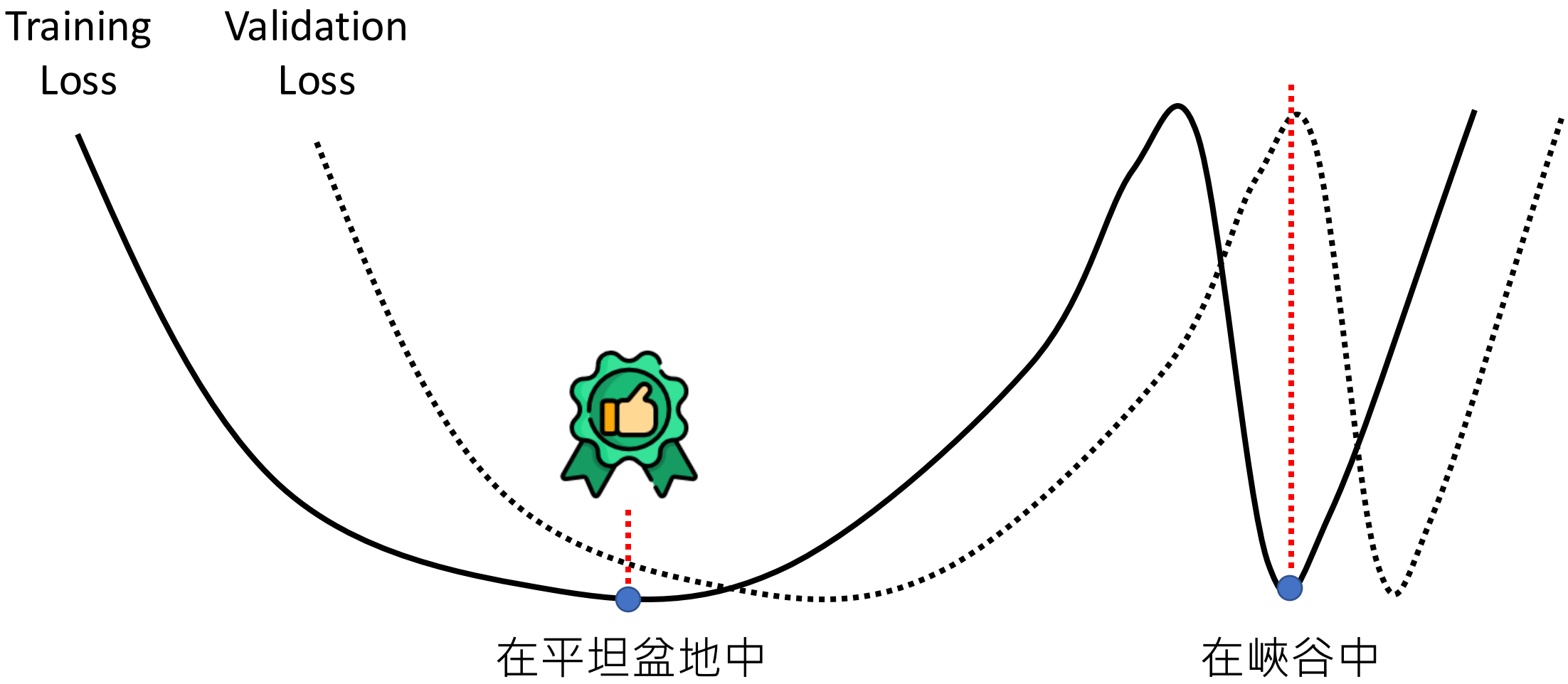
Initialization

當有多個 Training Loss 最低點的時候，
起始位置決定我們走到哪一個最低點



Initialization

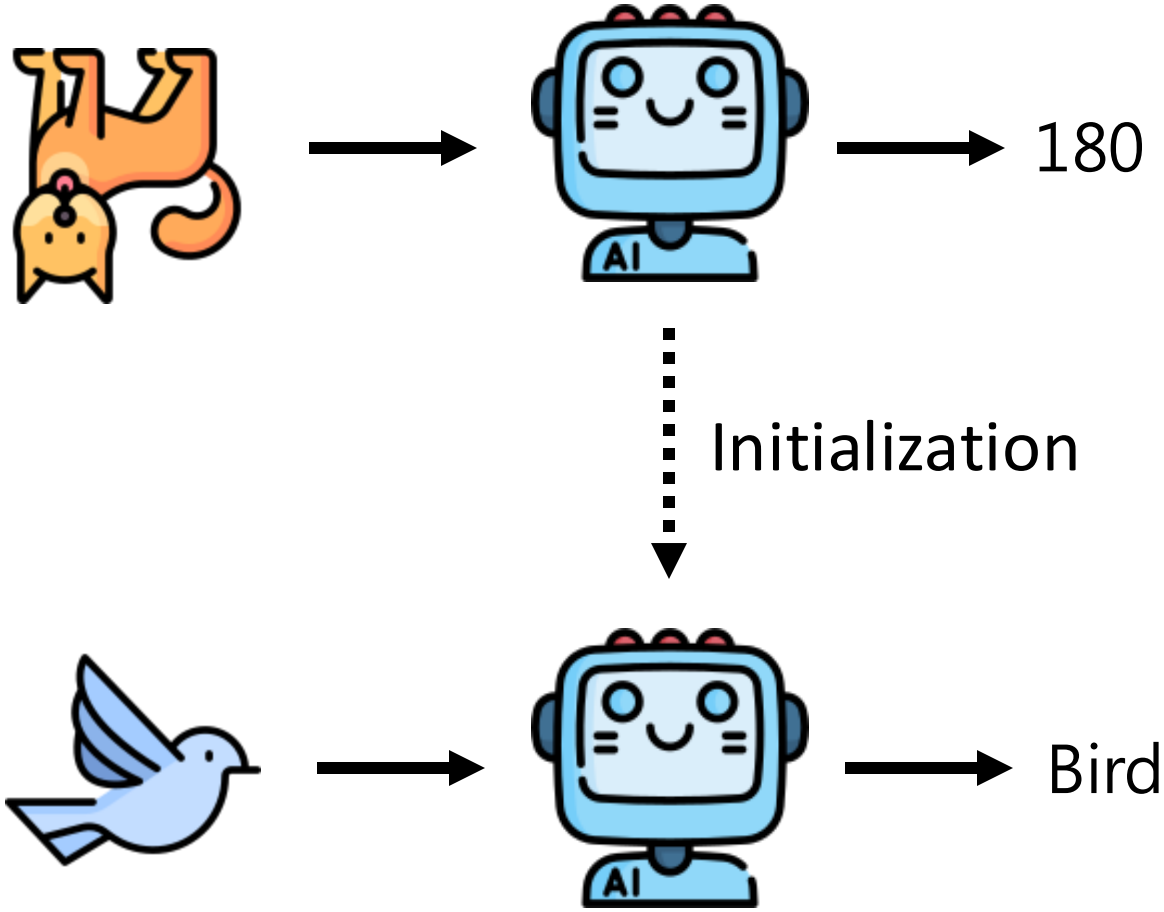
當有多個 Training Loss 最低點的時候，
起始位置決定我們走到哪一個最低點



Initialization

Pretext Task

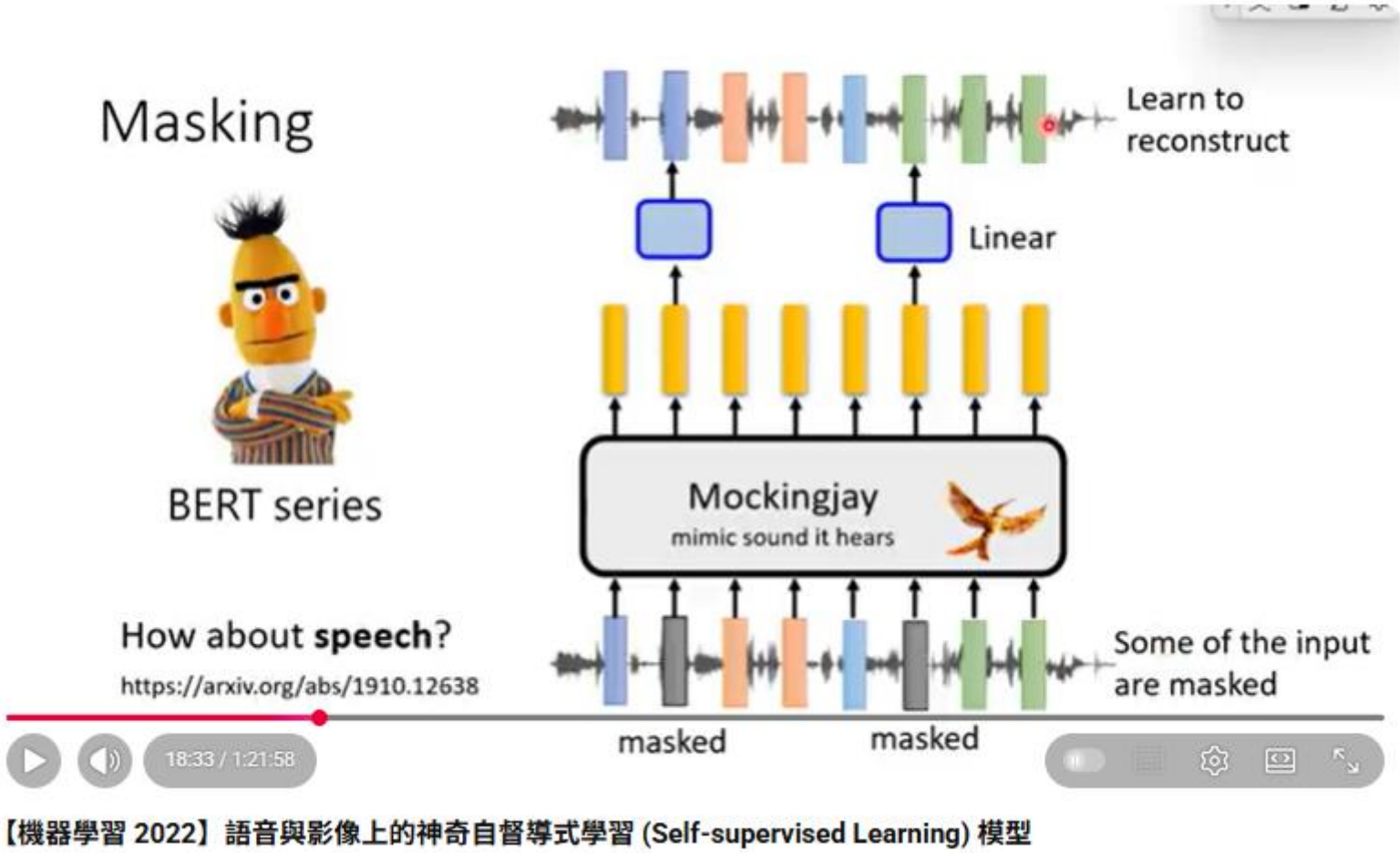
可以輕易蒐集大量資料進行訓練



Pre-train

Downstream Task

More about Pretext Task



https://youtu.be/lMIN1iKYNmA?si=M6hL7pKbOU_ufefo

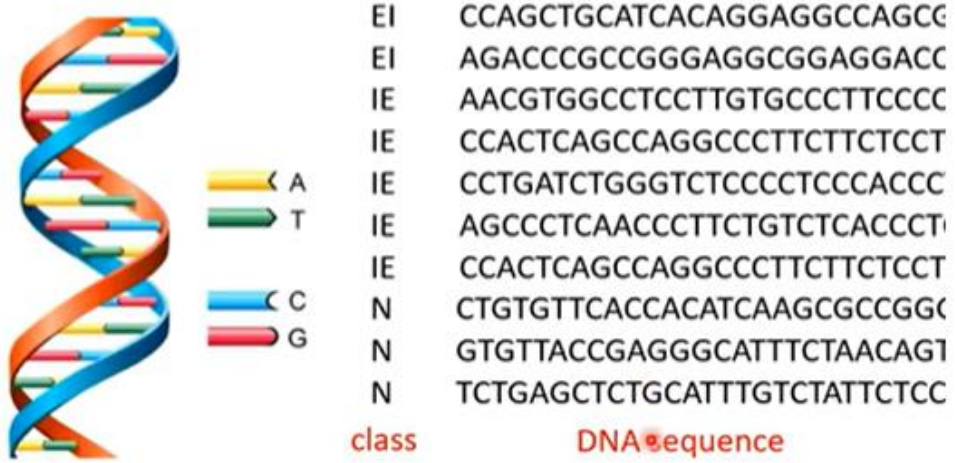
Initialization

Pre-train 對於 Optimization 和 Generalization 都有幫助

https://arxiv.org/abs/2103.07162
This work is done by 高瑋聰

Why does BERT work?

- Applying BERT to **protein, DNA, music classification**

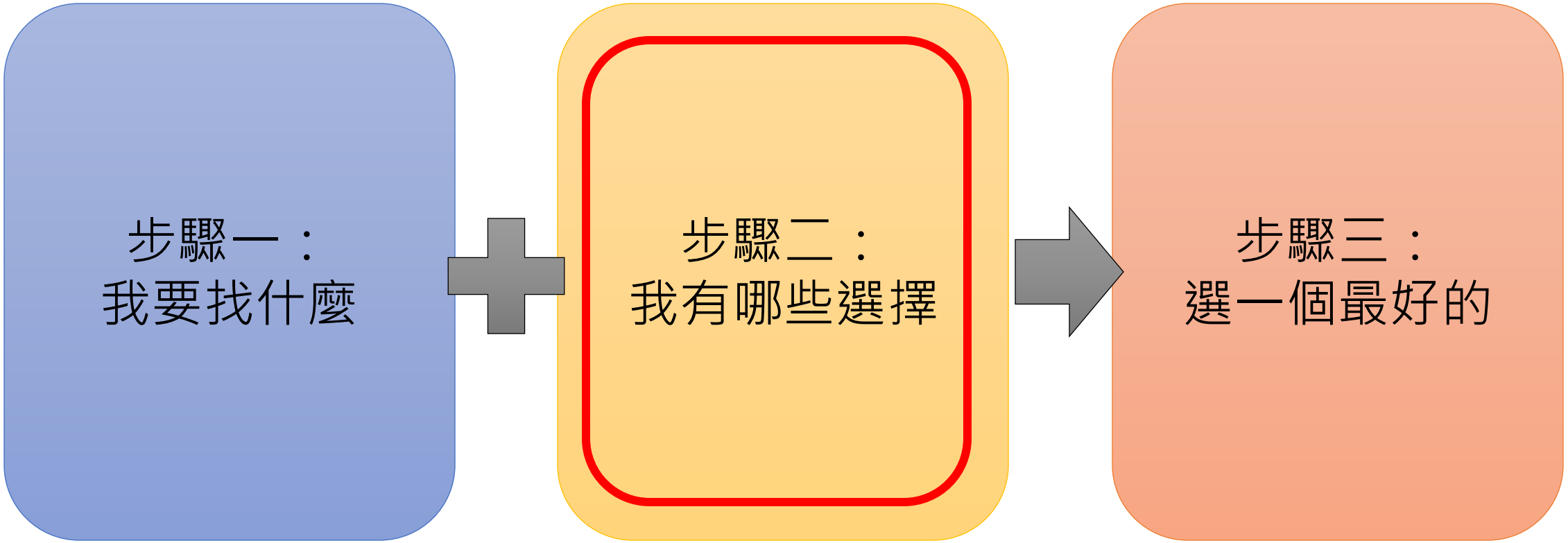


【機器學習2021】自督導式學習 (Self-supervised Learning) (三) – BERT的奇聞軼事

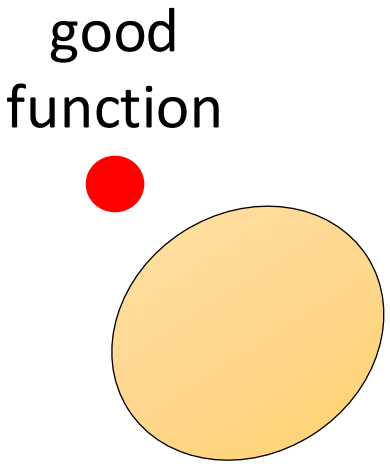
https://youtu.be/ExXA05i8DEQ?si=bpmI5yogGDiJ3FUf&t=508

方法名	改了那一個步驟	帶來什麼好處
Initialization (e.g., pre-train)	找最好的函式	Better Optimization, Better Generalization

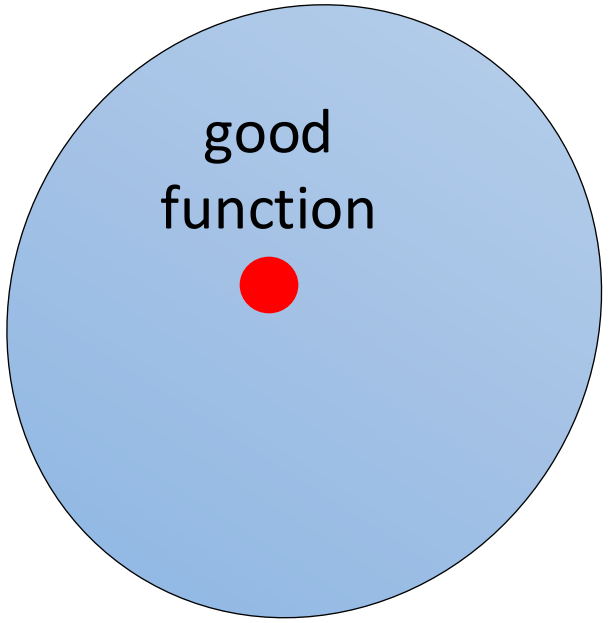
找函式步驟 3 + 1


步驟一：
我要找什麼
步驟二：
我有哪些選擇
步驟三：
選一個最好的
Better Network
Architecture

劃定一個剛剛好的函式範圍

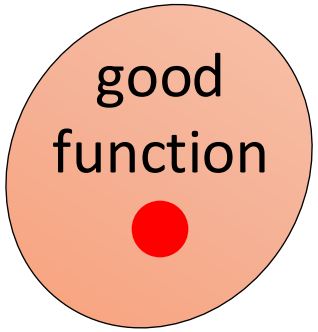


函式集合太小，可能會沒有包含好的函式



函式集合大一點，比較可能包到好的函式

(容易 overfitting)

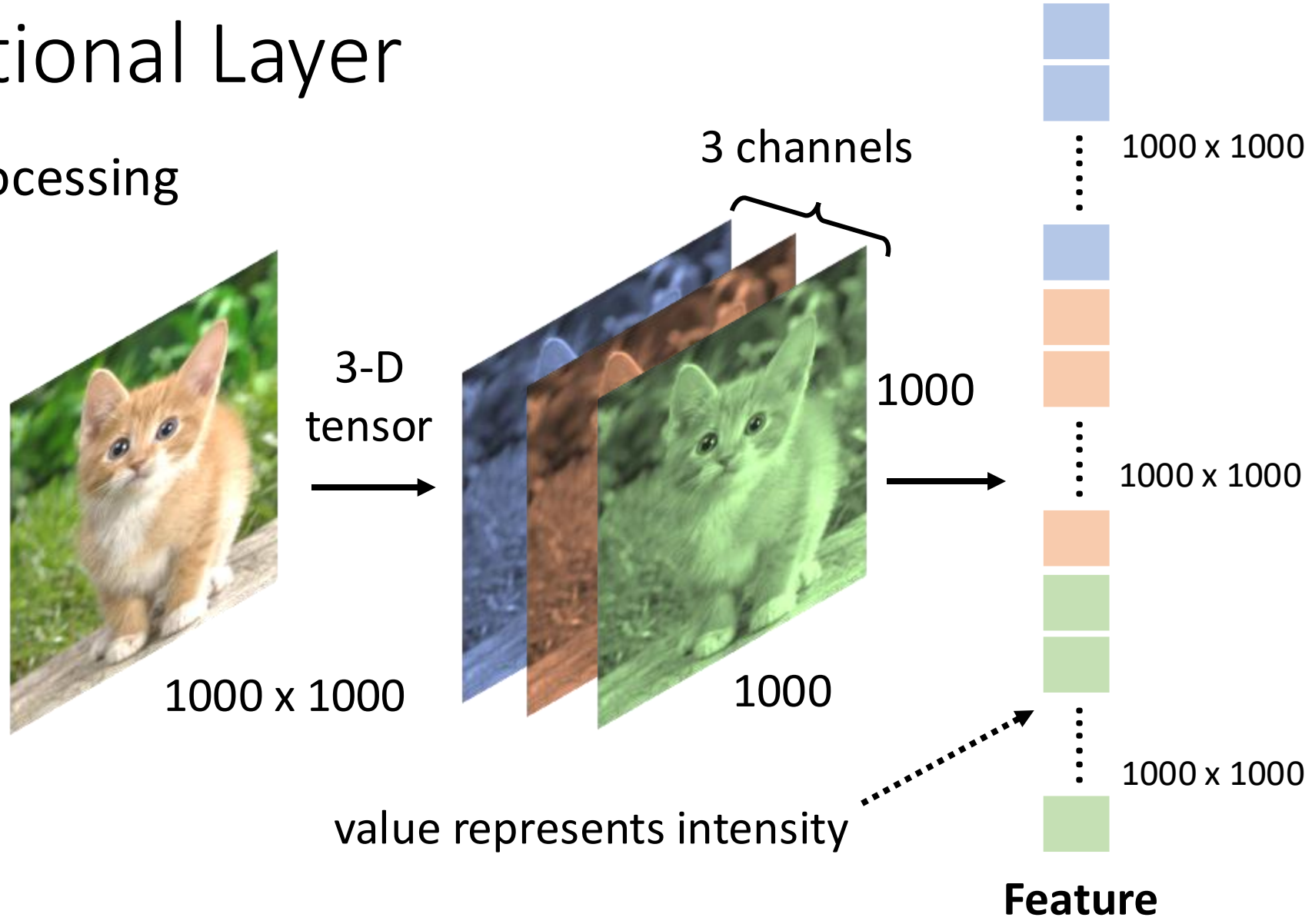


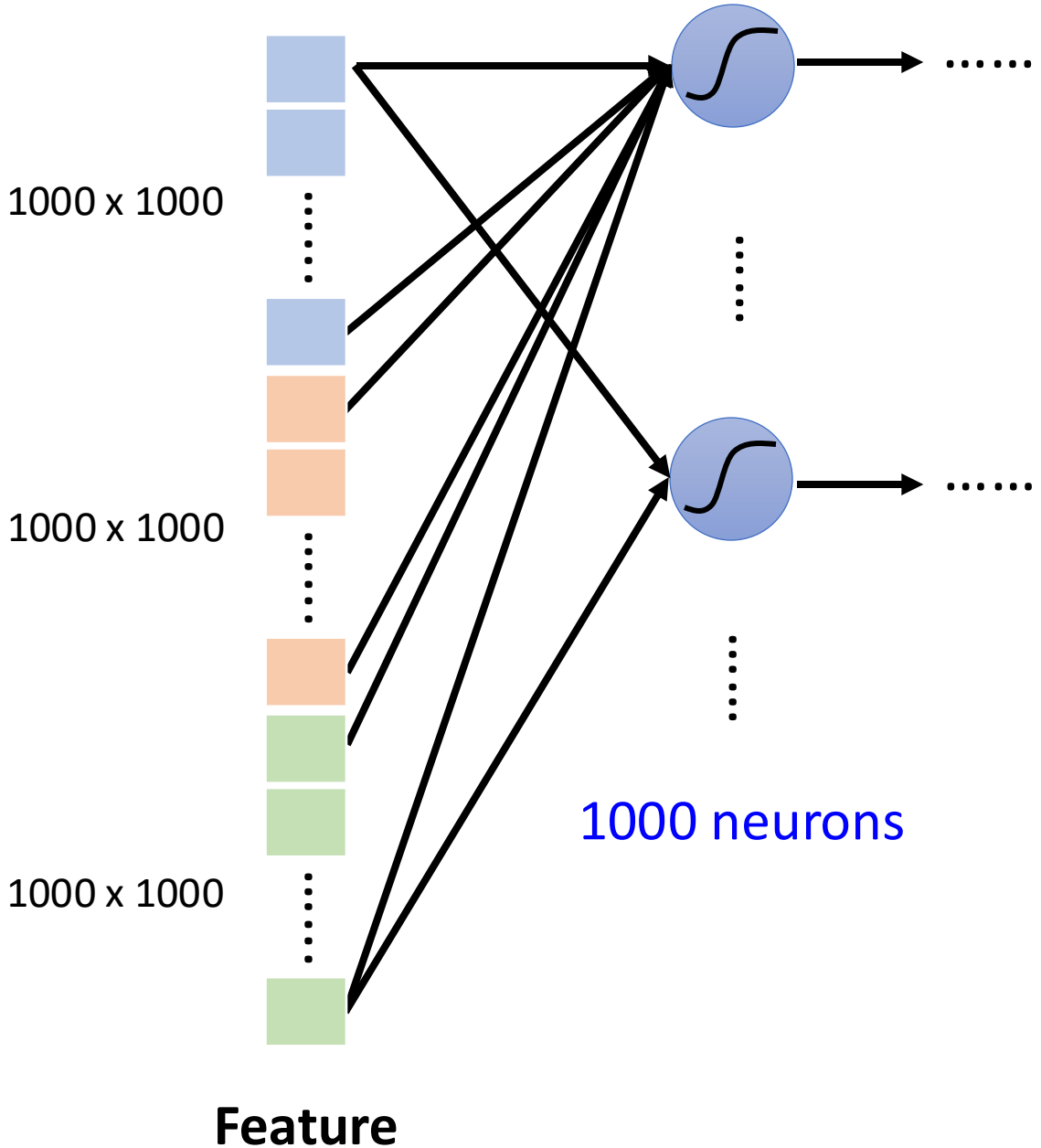
選一個小範圍但剛好包含好的函式

(根據 domain knowledge)

Convolutional Layer

For image processing





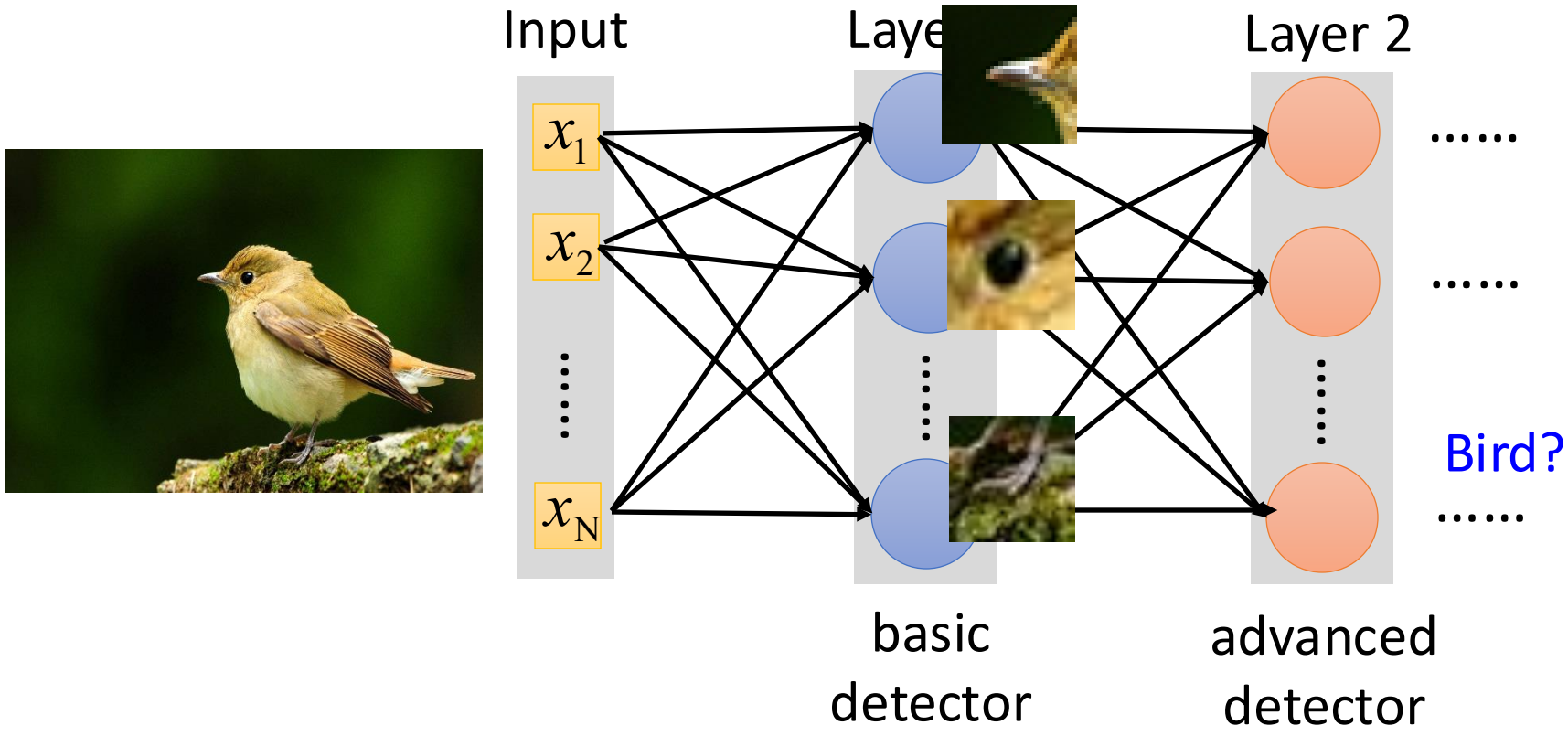
1000 x 1000 x 3 parameters

1000 x 1000 x 3 x 1000 parameters

Typical multi-layer perceptron (MLP) is ***"fully connected".***

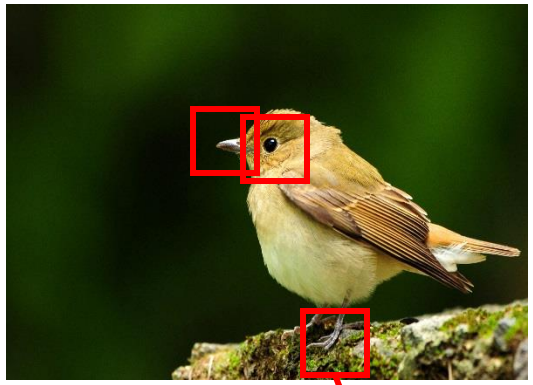
Do we really need ***"fully connected"*** in image processing?

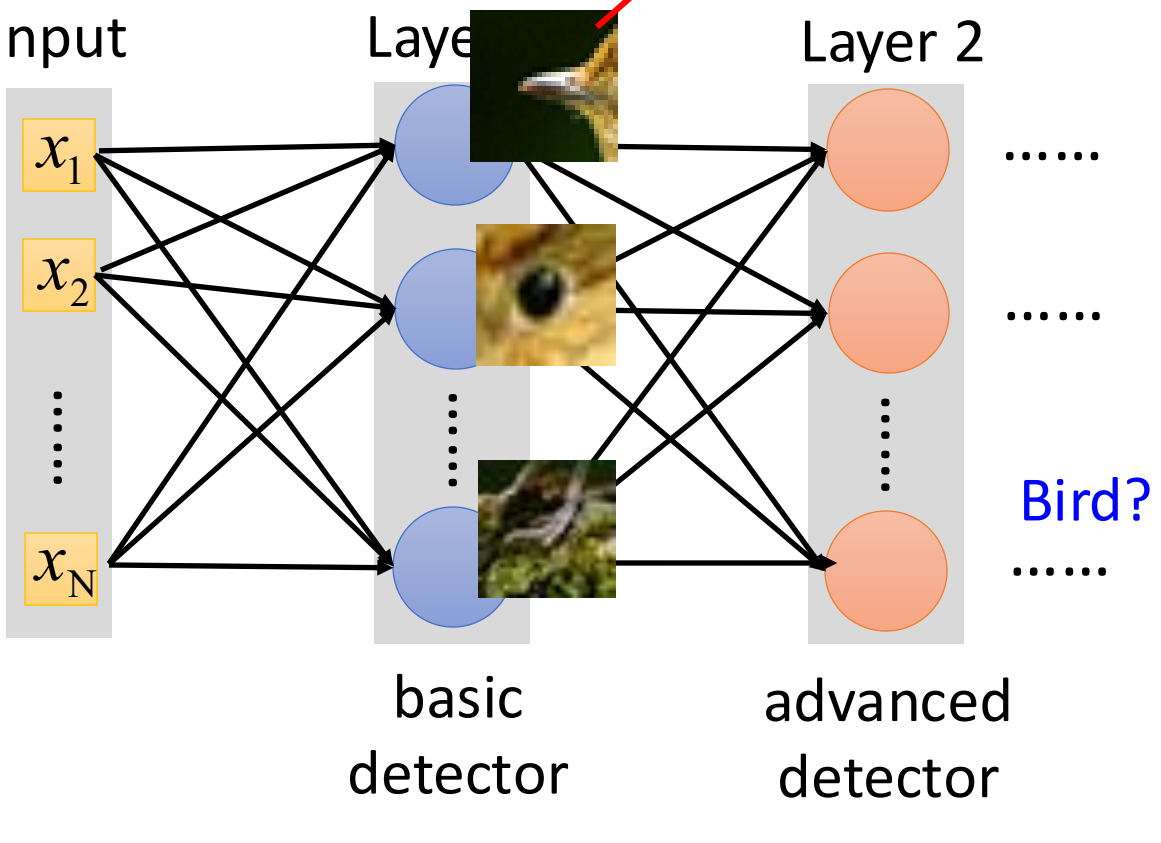
Observation 1



Observation 1

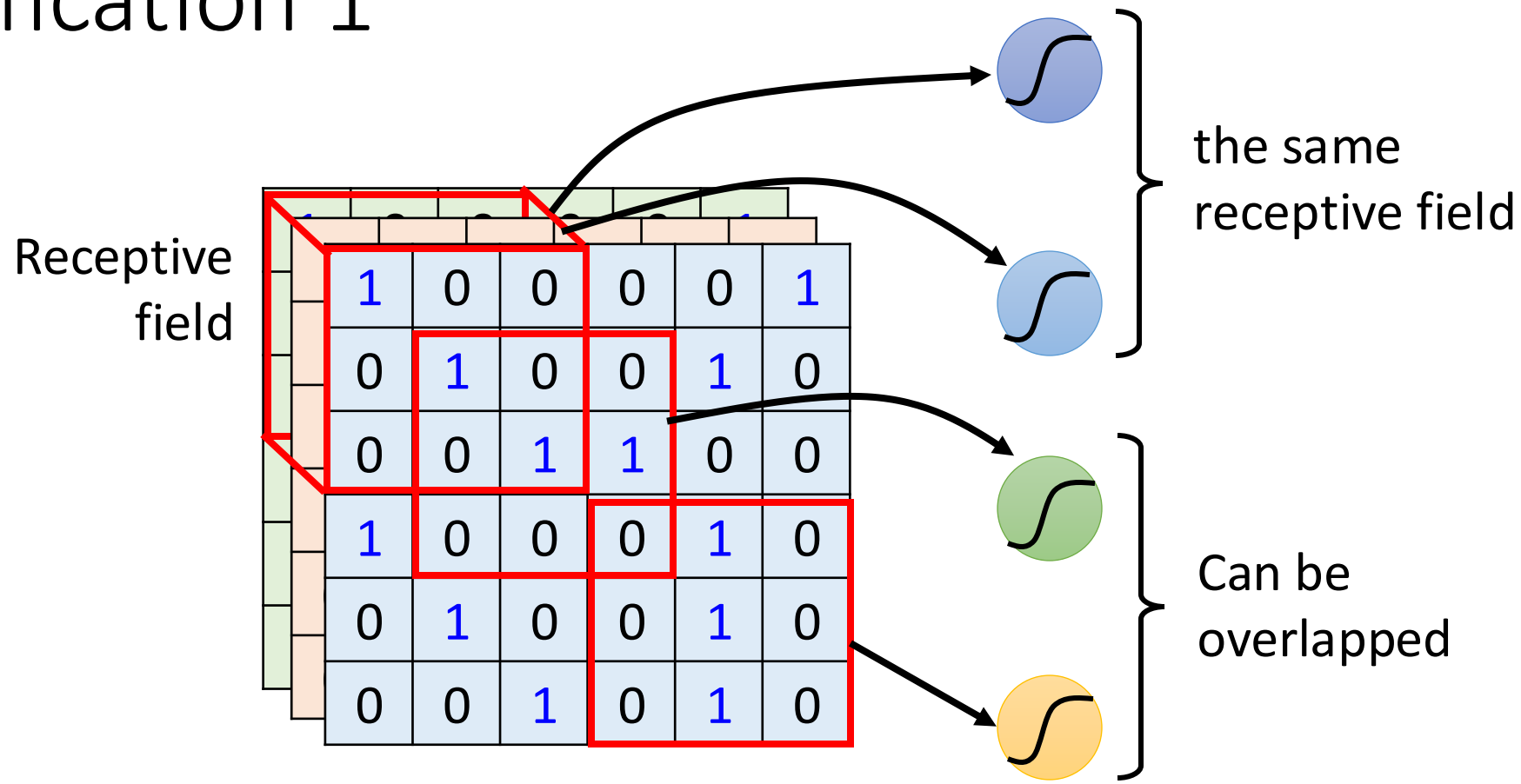
A neuron does not have to see the whole image.





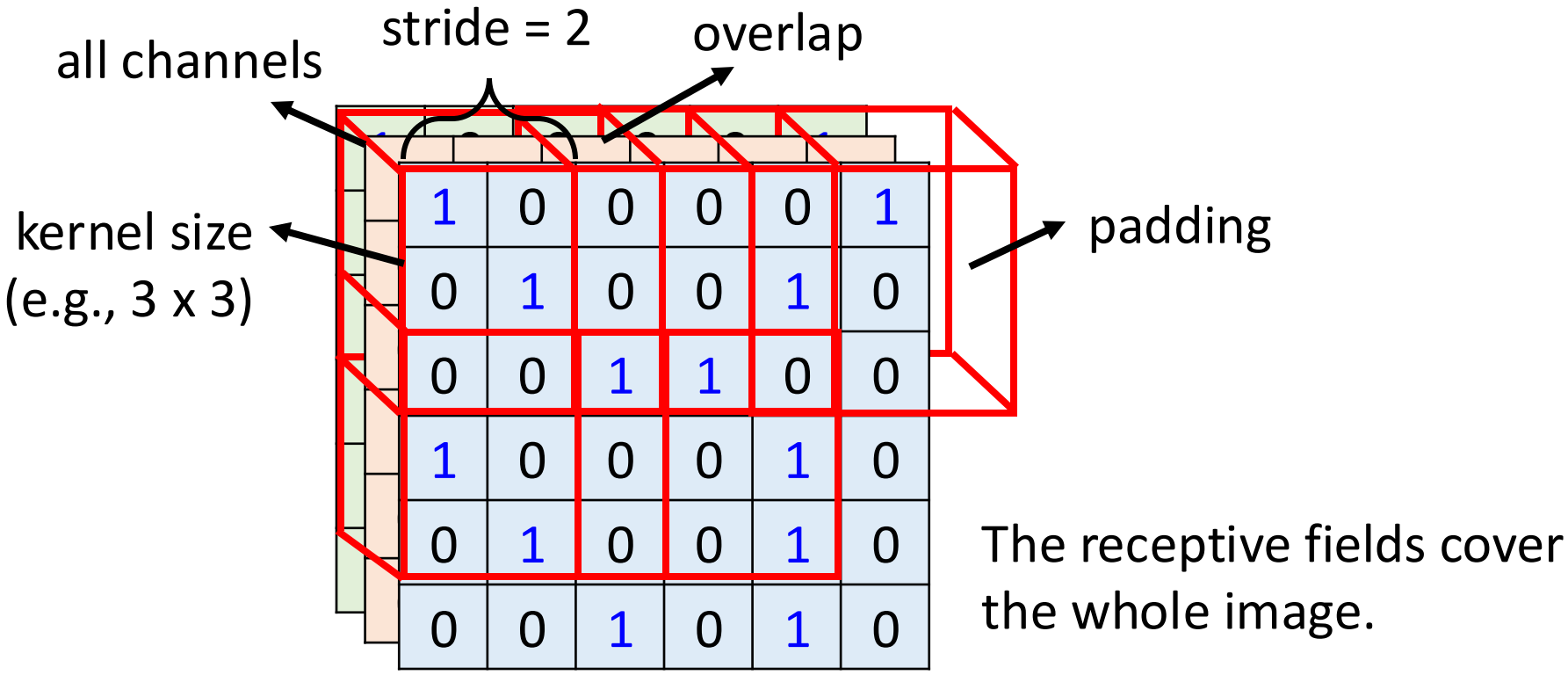
Some patterns are much smaller than the whole image.

Simplification 1

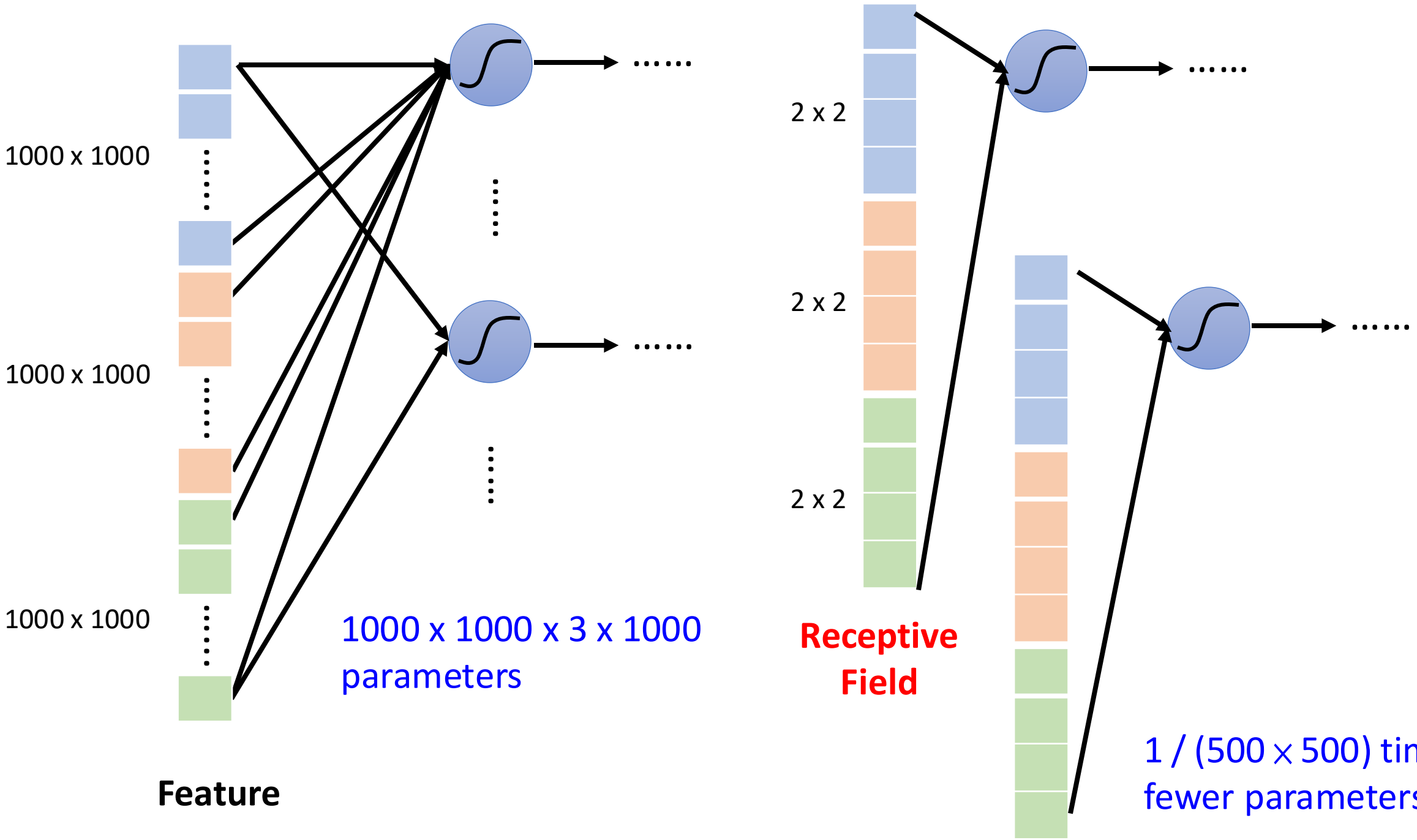


- Can different neurons have different sizes of receptive field?
- Not square receptive field?

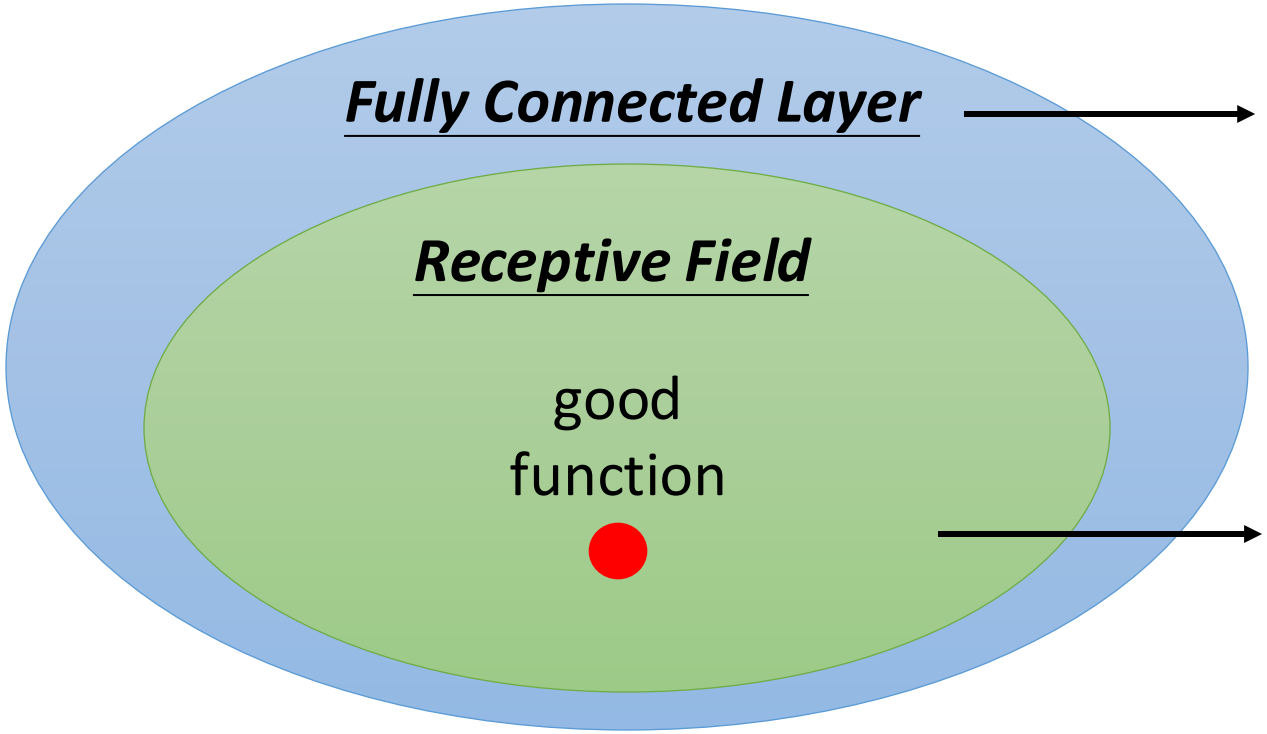
Simplification 1 – Typical Setting



Each receptive field has a set of neurons (e.g., 64 neurons).


1000 x 1000
1000 x 1000
1000 x 1000
1000 x 1000 x 3 x 1000
parameters
Feature
2 x 2
2 x 2
2 x 2
Receptive
Field
1 / (500 x 500) times
fewer parameters

Benefit of Simplification 1



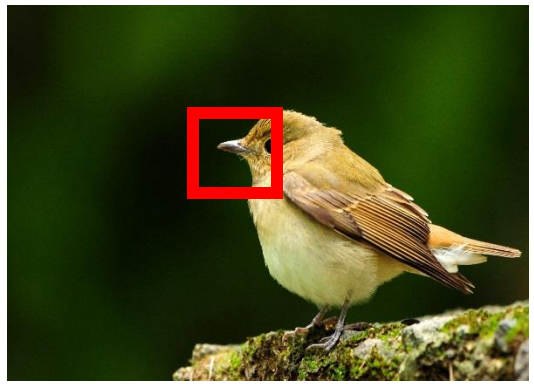
Each neuron processes the whole image.

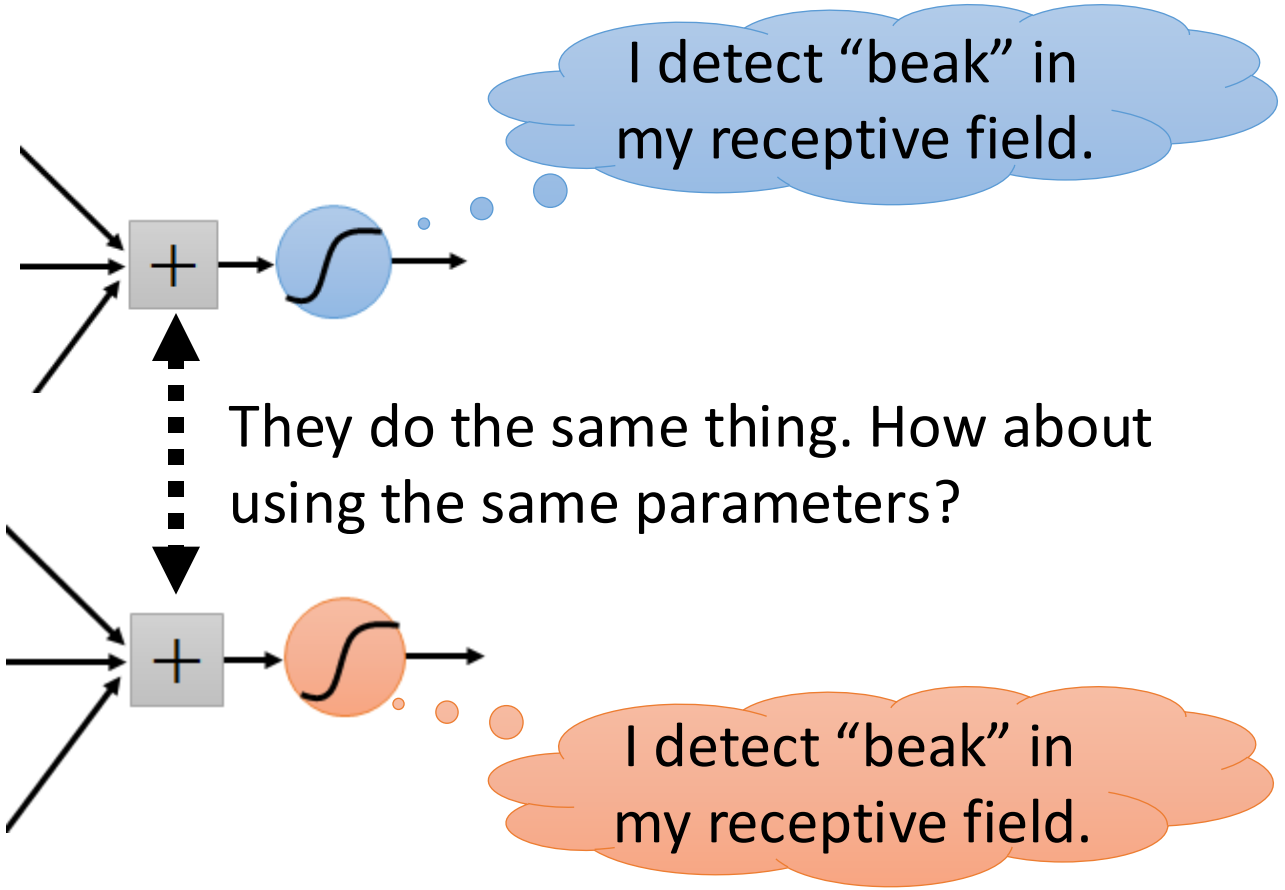
Each neuron only processes a small region (receptive field).

- Basic patterns are much smaller than the whole image. Each neuron only has to process a receptive field.

Observation 2

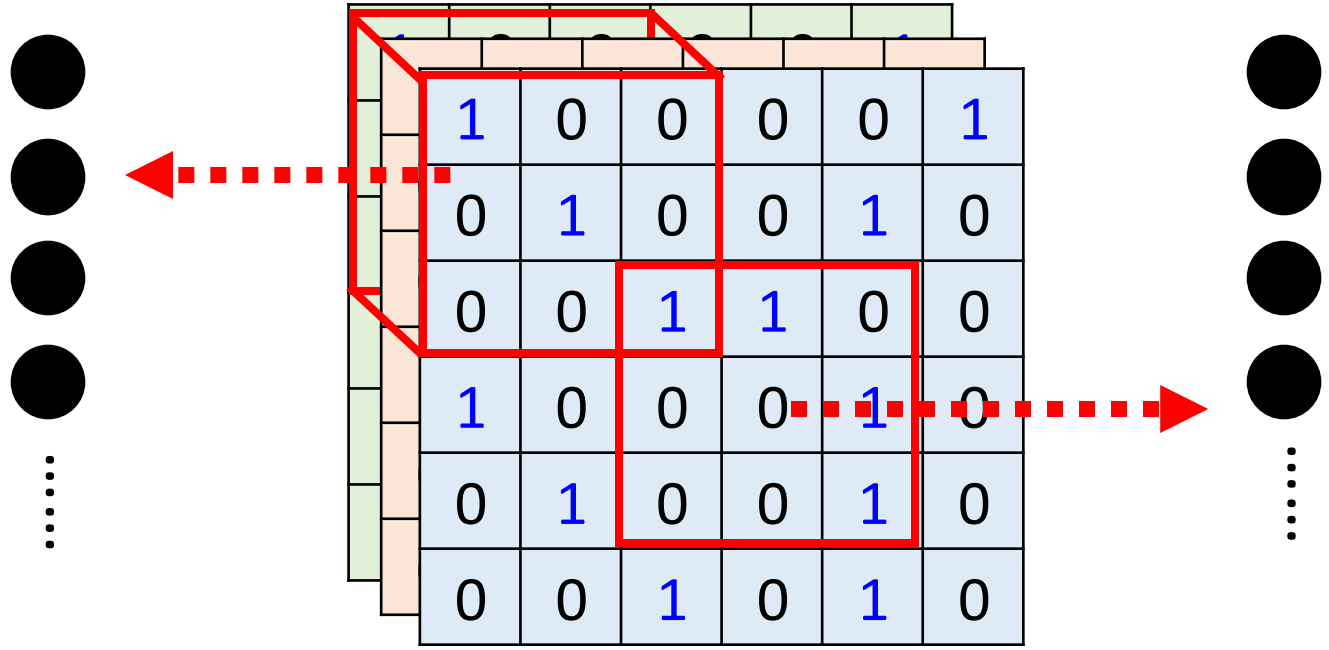
- The same patterns appear in different regions.





Simplification 2 – Sharing parameters

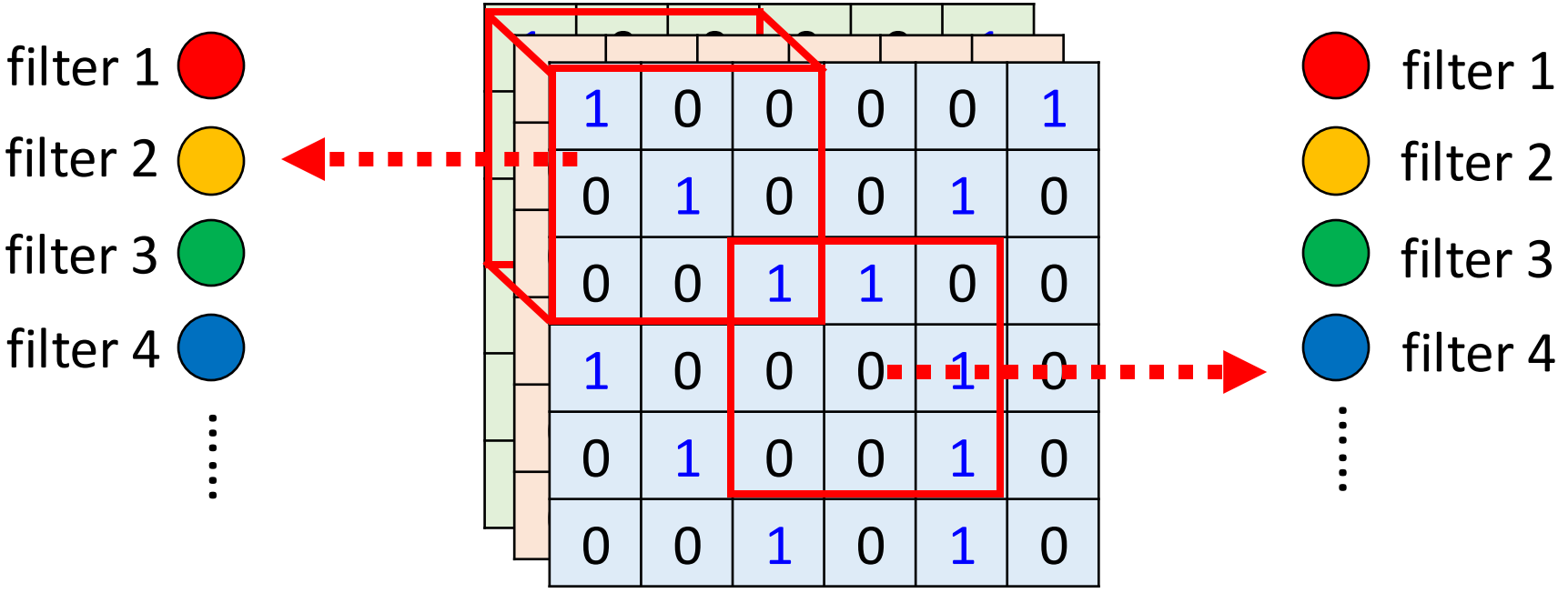
Each receptive field has a set of neurons (e.g., 64 neurons).



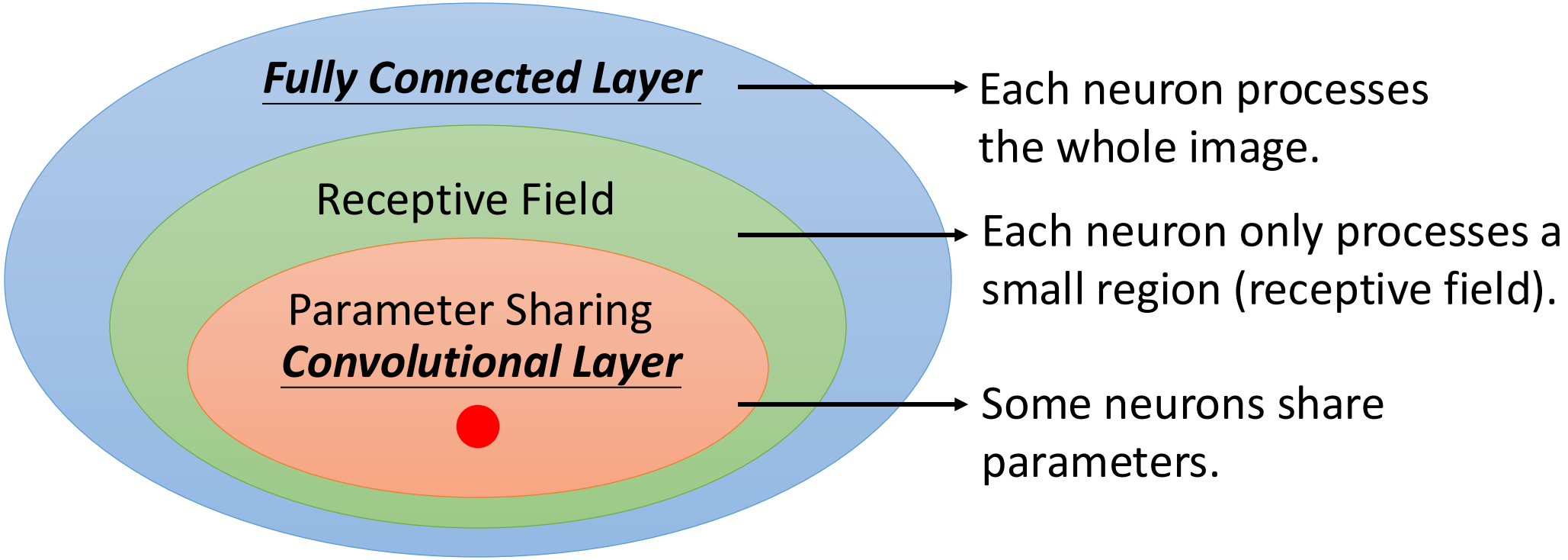
Simplification 2 – Sharing parameters

Each receptive field has a set of neurons (e.g., 64 neurons).

Each receptive field has the neurons with the same set of parameters.



Benefit of Simplication 1 + 2

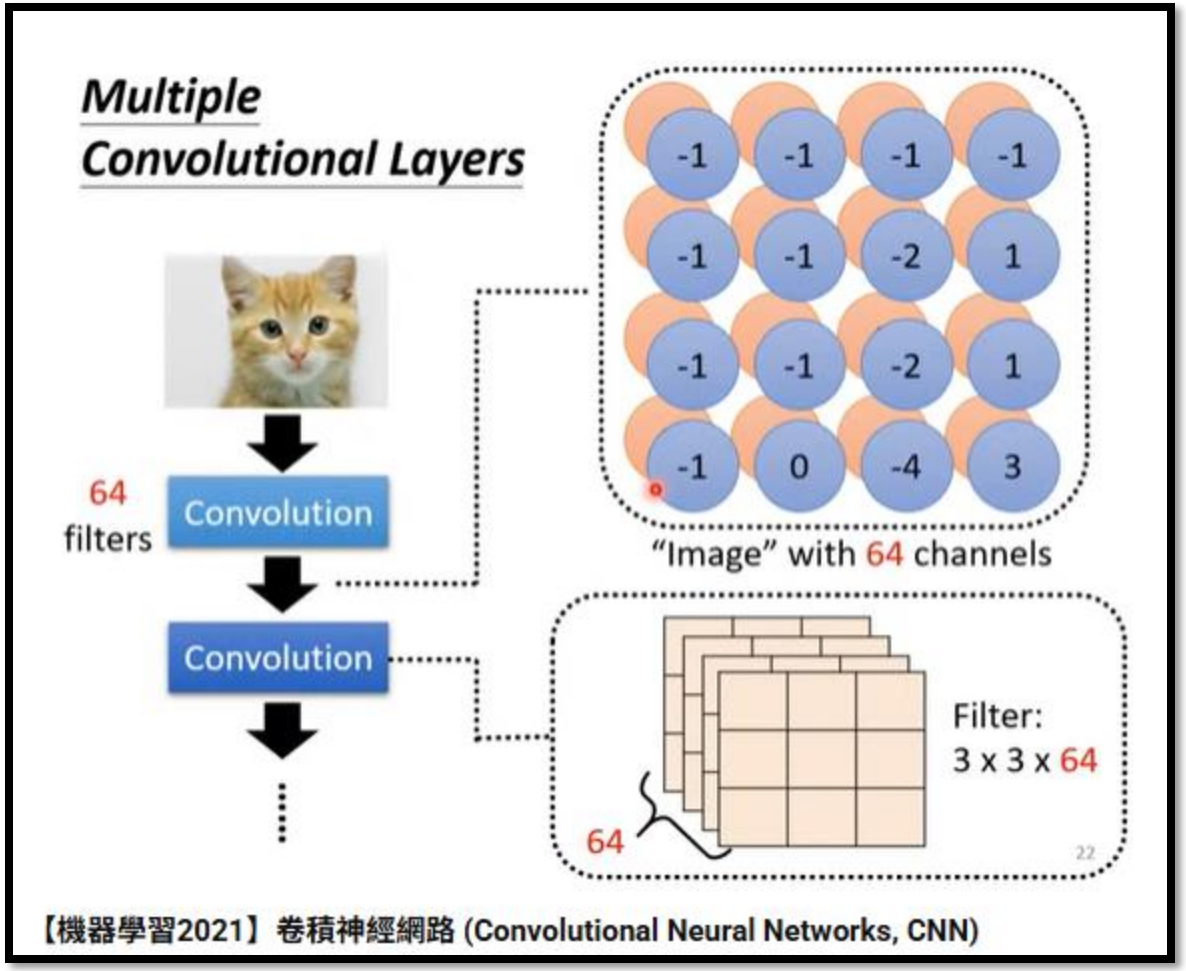


- Basic patterns are much smaller than the whole image. Each neuron only has to process a receptive field.
- Different receptive fields need the same detectors, so neurons for different receptive fields share parameters.

方法名	改了那一個步驟	帶來什麼好處
CNN (e.g., for image)	改變函式搜尋範圍	Better Generalization

To learn more about CNN …

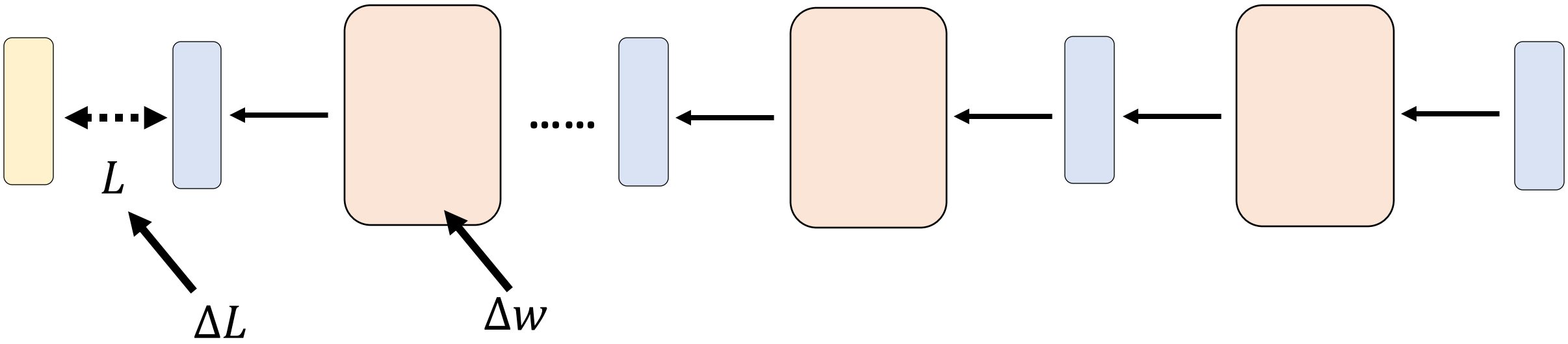
https://youtu.be/OP5HcXJg2Aw?si=cNhXe9W-PgP45jN1&t=1686



(為了簡化，此處沒有畫出 activation 和 bias)

Skip Connection (Residual Connection)

https://arxiv.org/abs/1512.03385

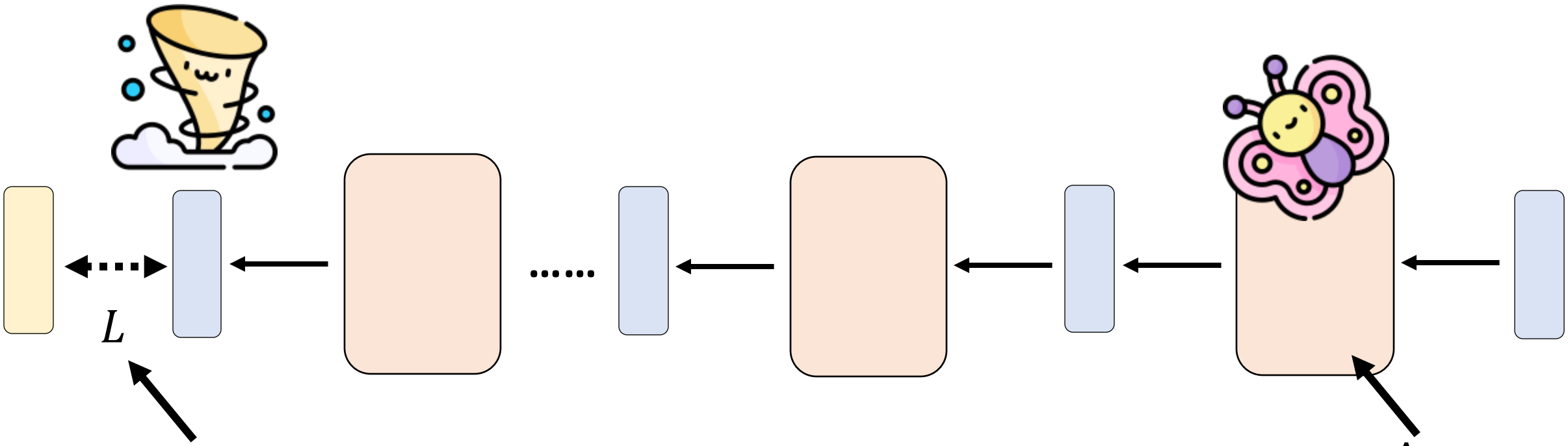


$$\frac{\partial L}{\partial w} \approx \frac{\Delta L}{\Delta w}$$

(為了簡化，此處沒有畫出 activation 和 bias)

Skip Connection (Residual Connection)

https://arxiv.org/abs/1512.03385



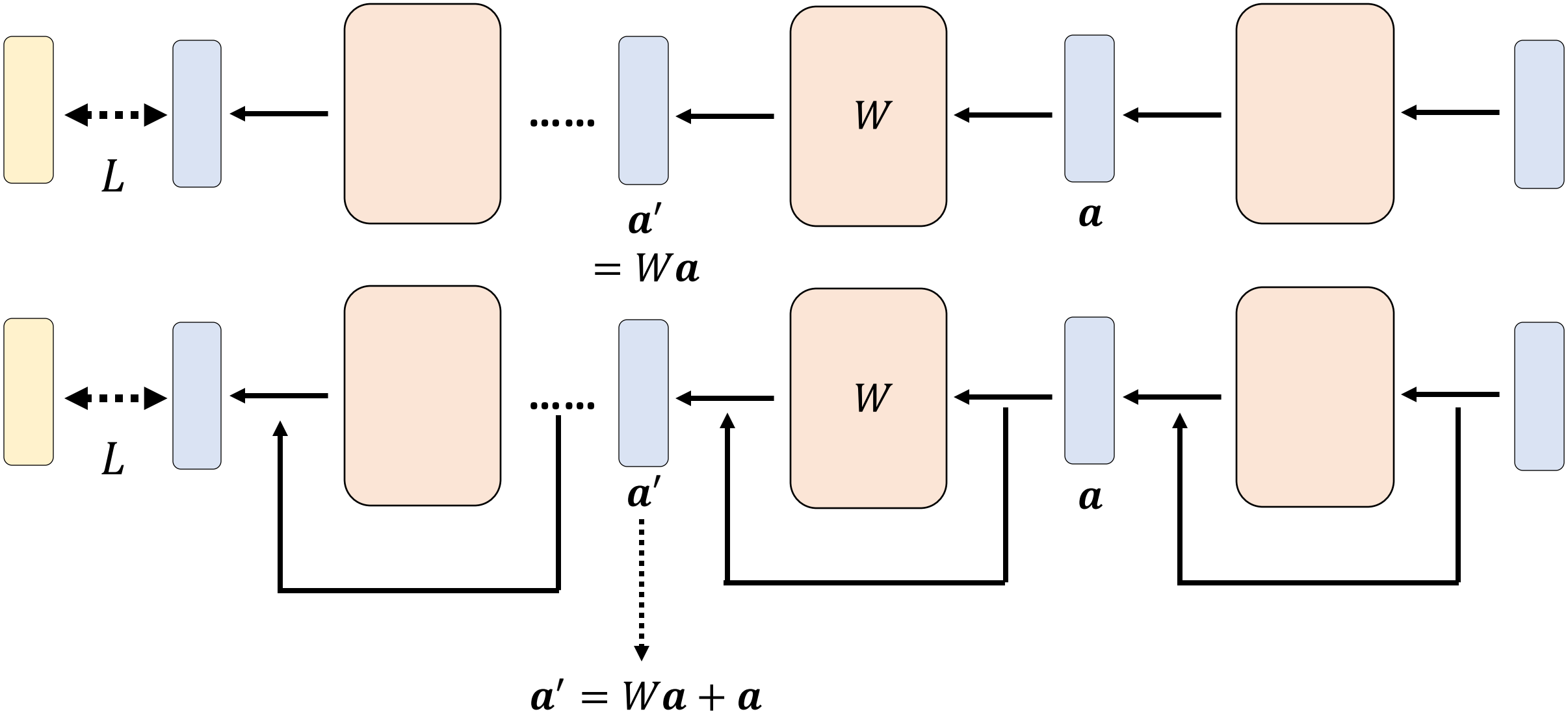
ΔL ⋯⋯▸ 往往很小 (gradient vanishing)

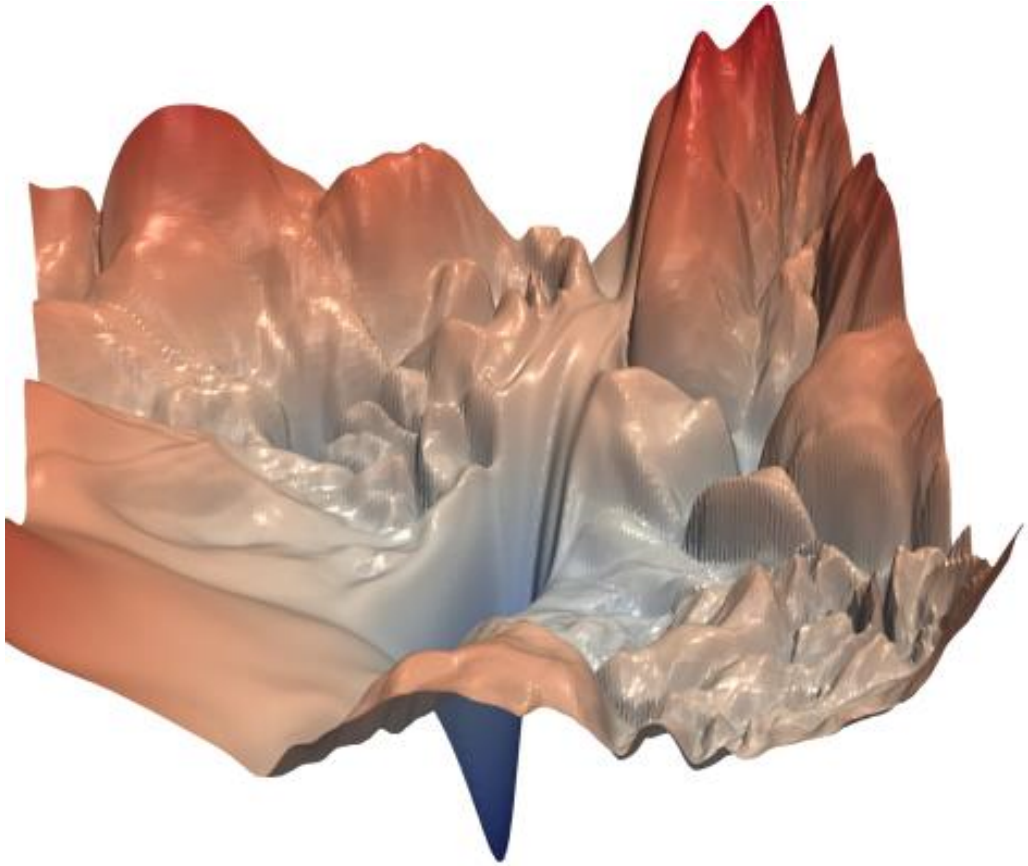
$\frac{\partial L}{\partial w} \approx \frac{\Delta L}{\Delta w}$

有時候又會突然變很大 (gradient exploding)

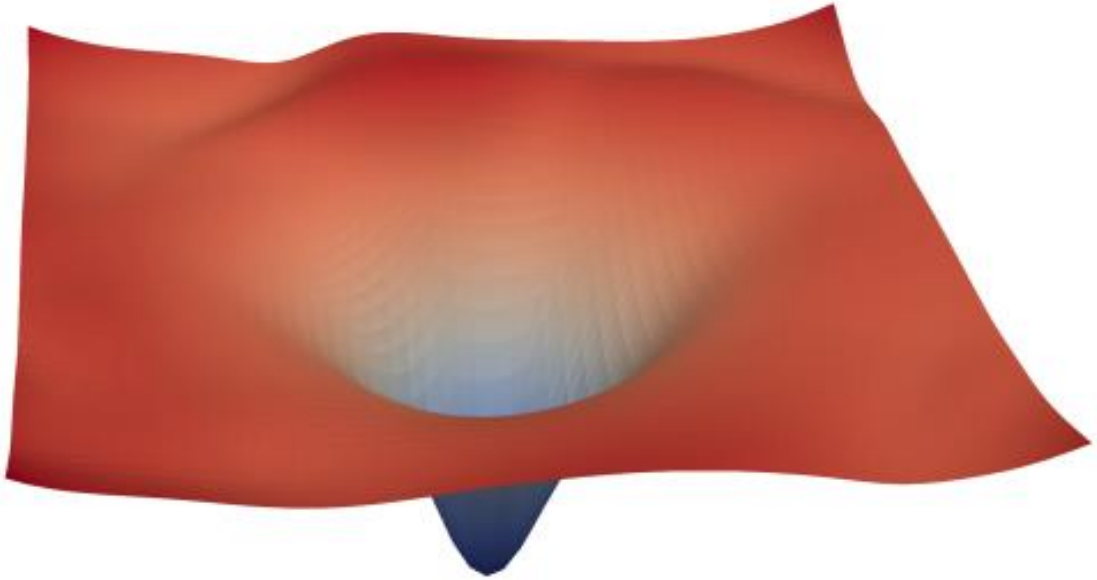
(為了簡化，此處沒有畫出 activation 和 bias)

Skip Connection (Residual Connection)





Source of image: https://arxiv.org/abs/1712.09913

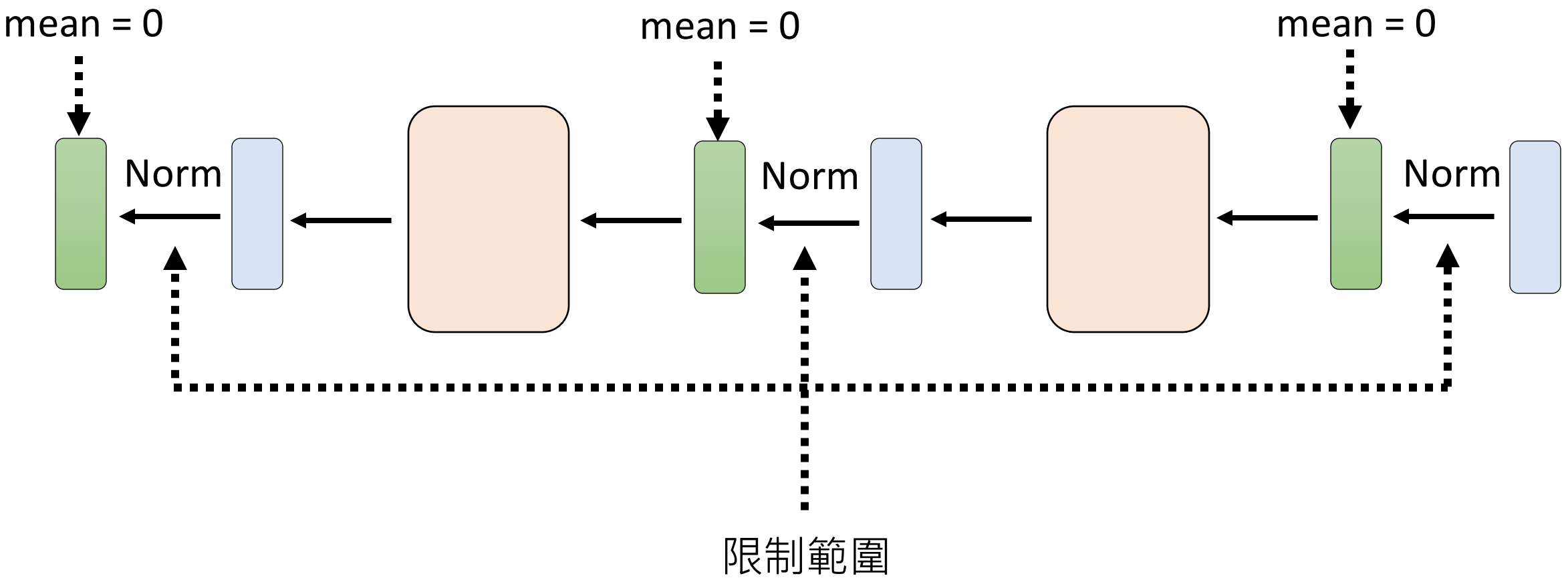


Without skip connection

With skip connection

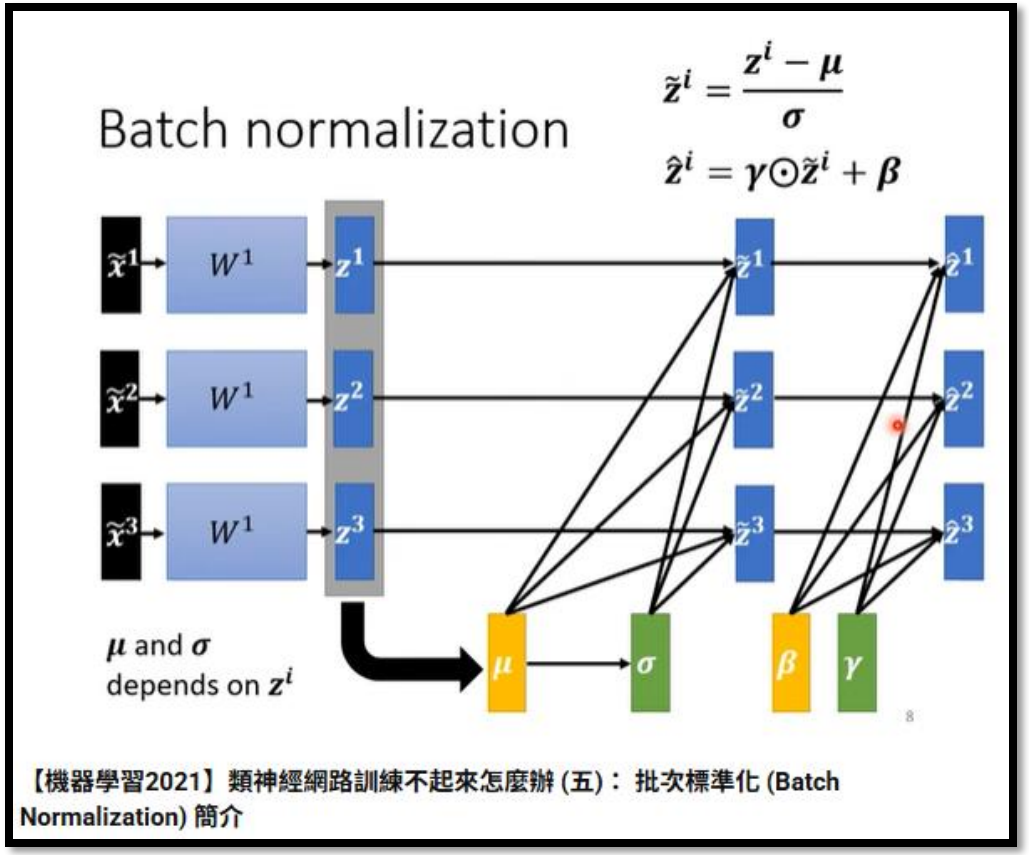
方法名	改了那一個步驟	帶來什麼好處
Skip Connection	改變函式搜尋範圍	Better Optimization

各種 Normalization



Normalization

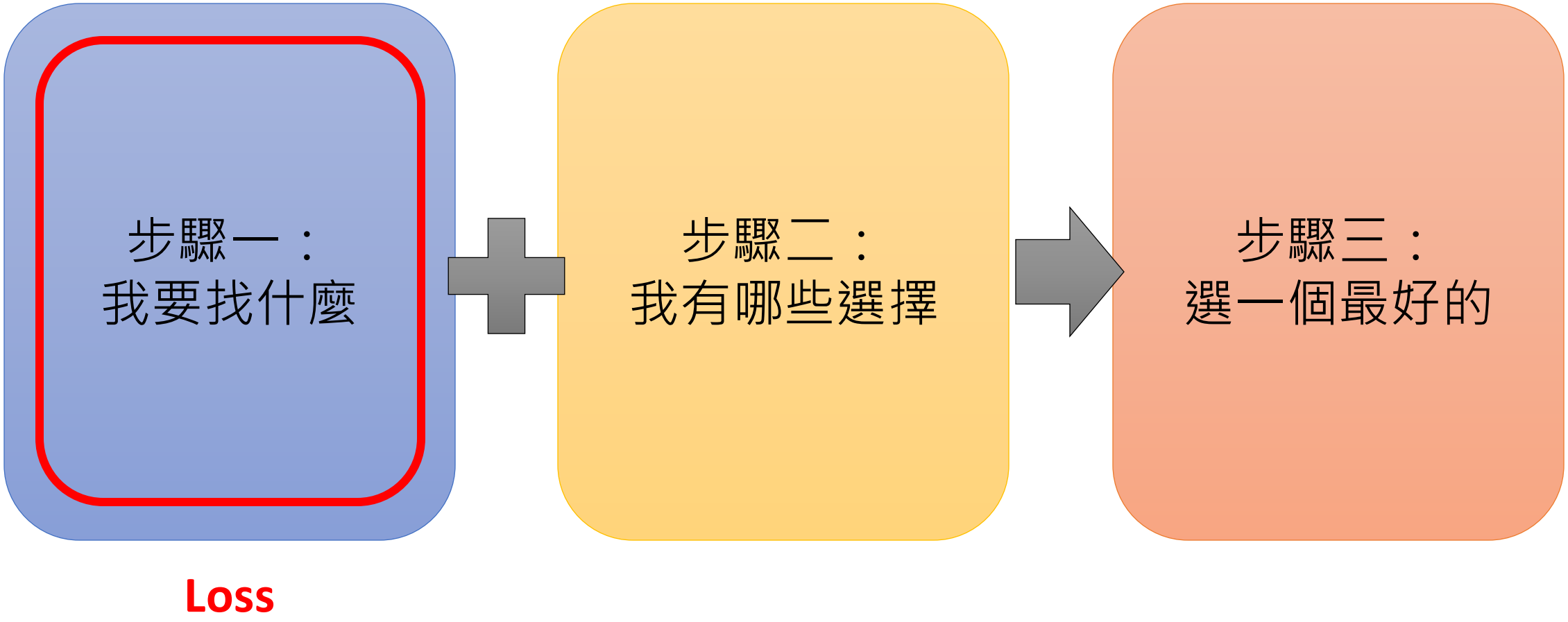
- Batch Renormalization
  - https://arxiv.org/abs/1702.03275
- Layer Normalization
  - https://arxiv.org/abs/1607.06450



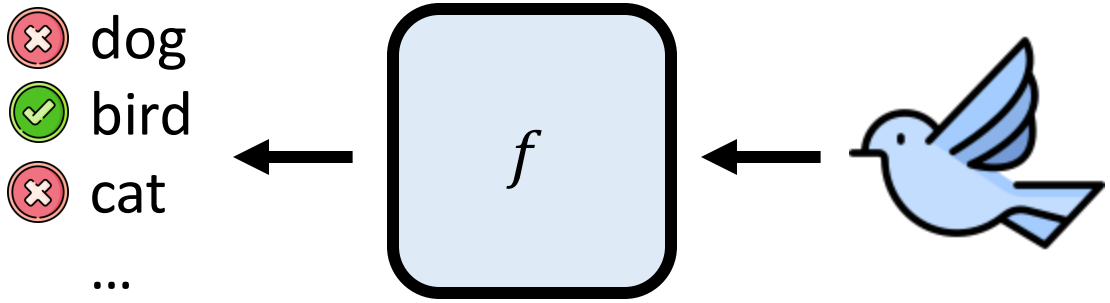
https://youtu.be/BABPWOkSbLE?si=gzwtWMEBzbgf7slI

方法名	改了那一個步驟	帶來什麼好處
Normalization	改變函式搜尋範圍	Better Optimization (Sometimes Better Generalization)

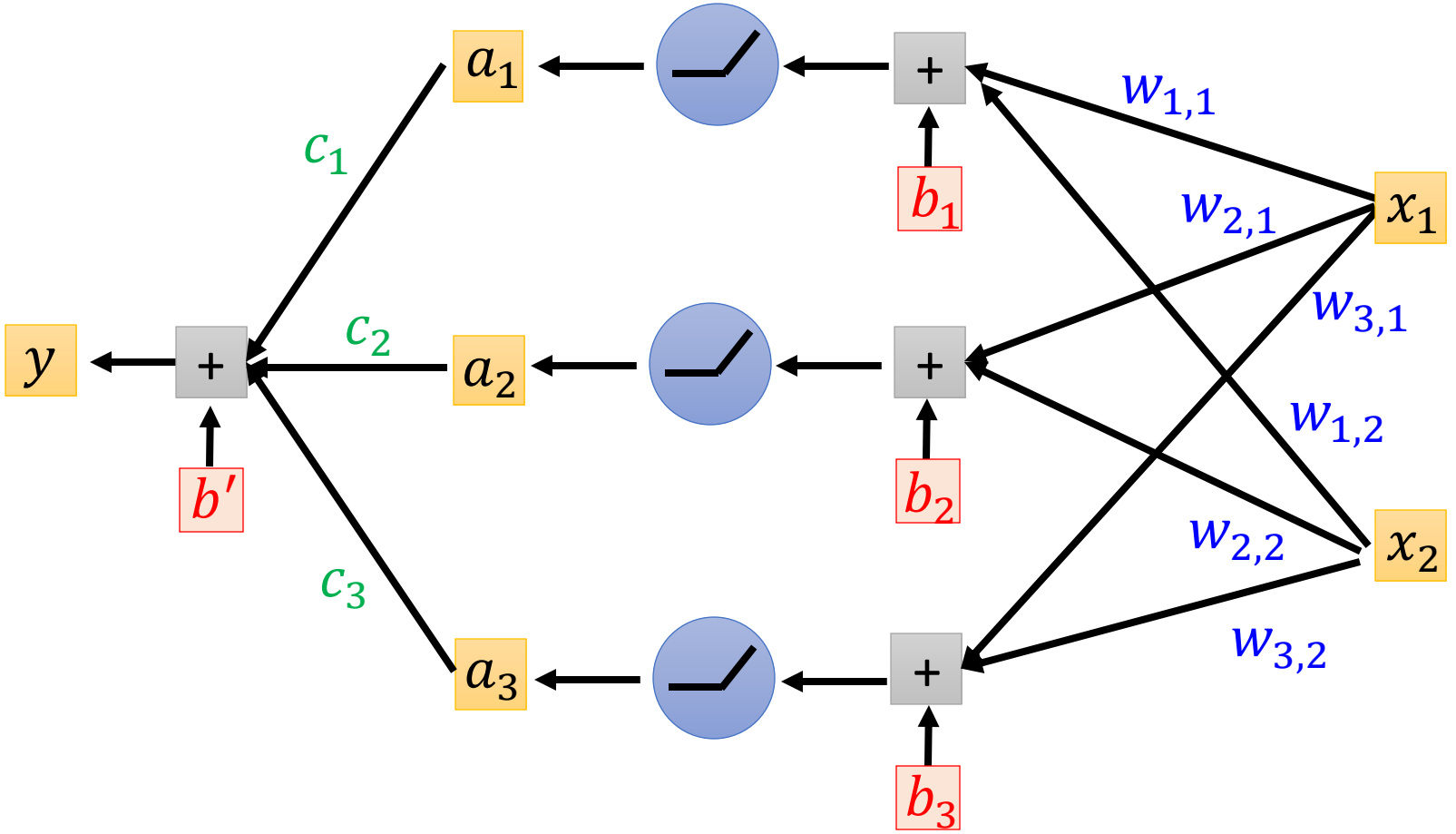
找函式步驟 3 + 1



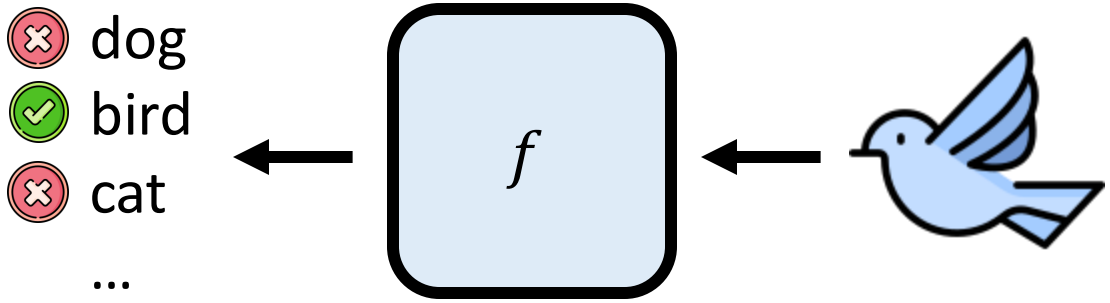
Classification



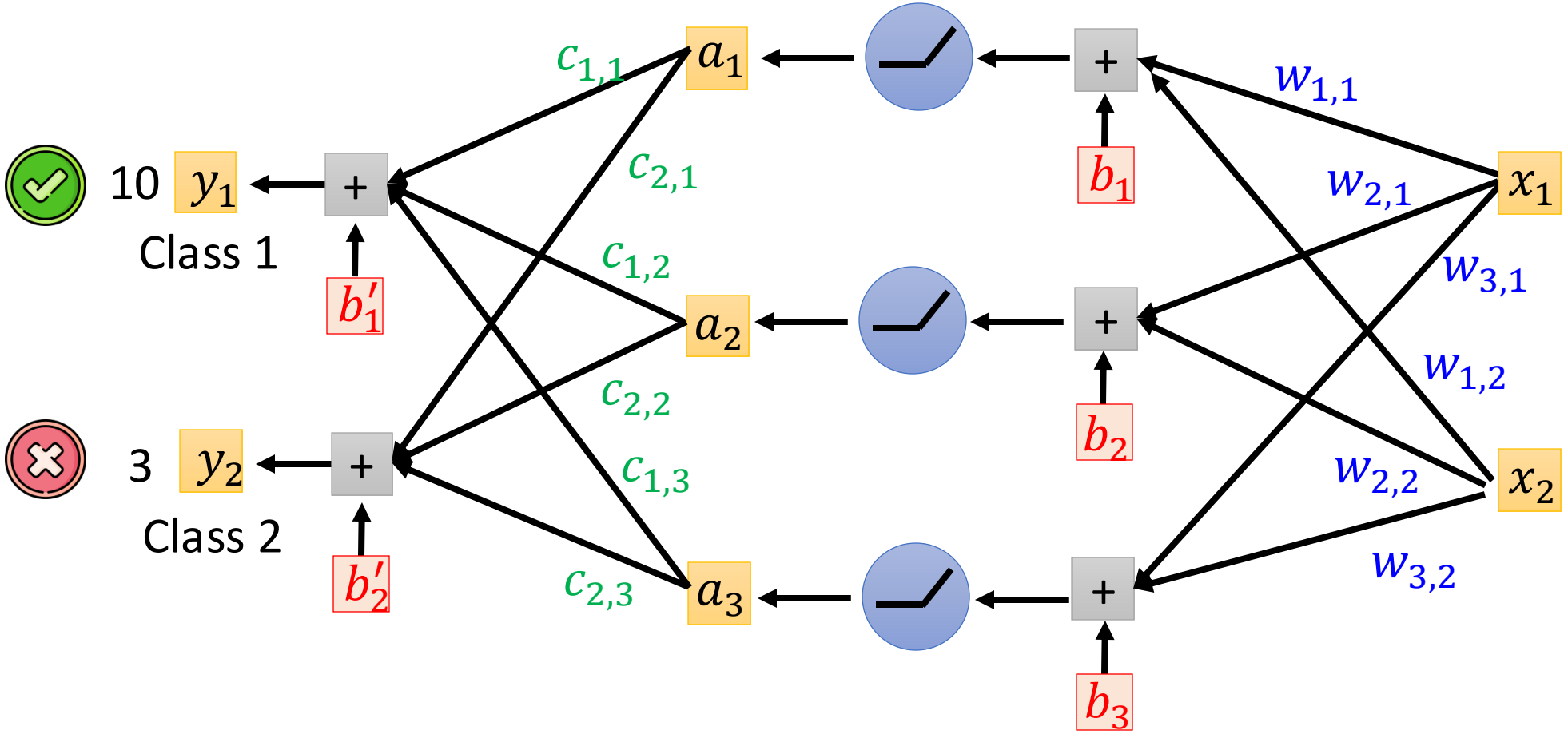
Regression



Classification

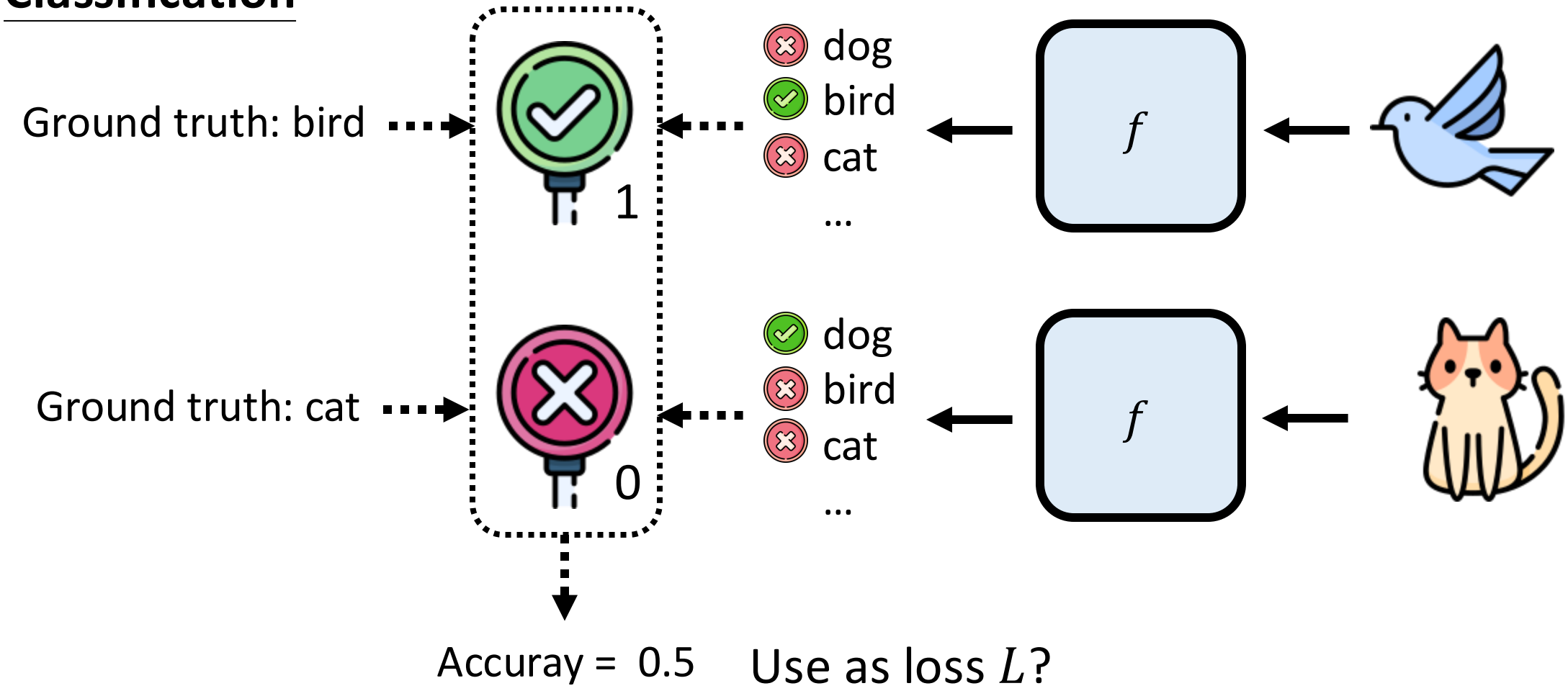


Classification



Loss = Evaluation Metrics?

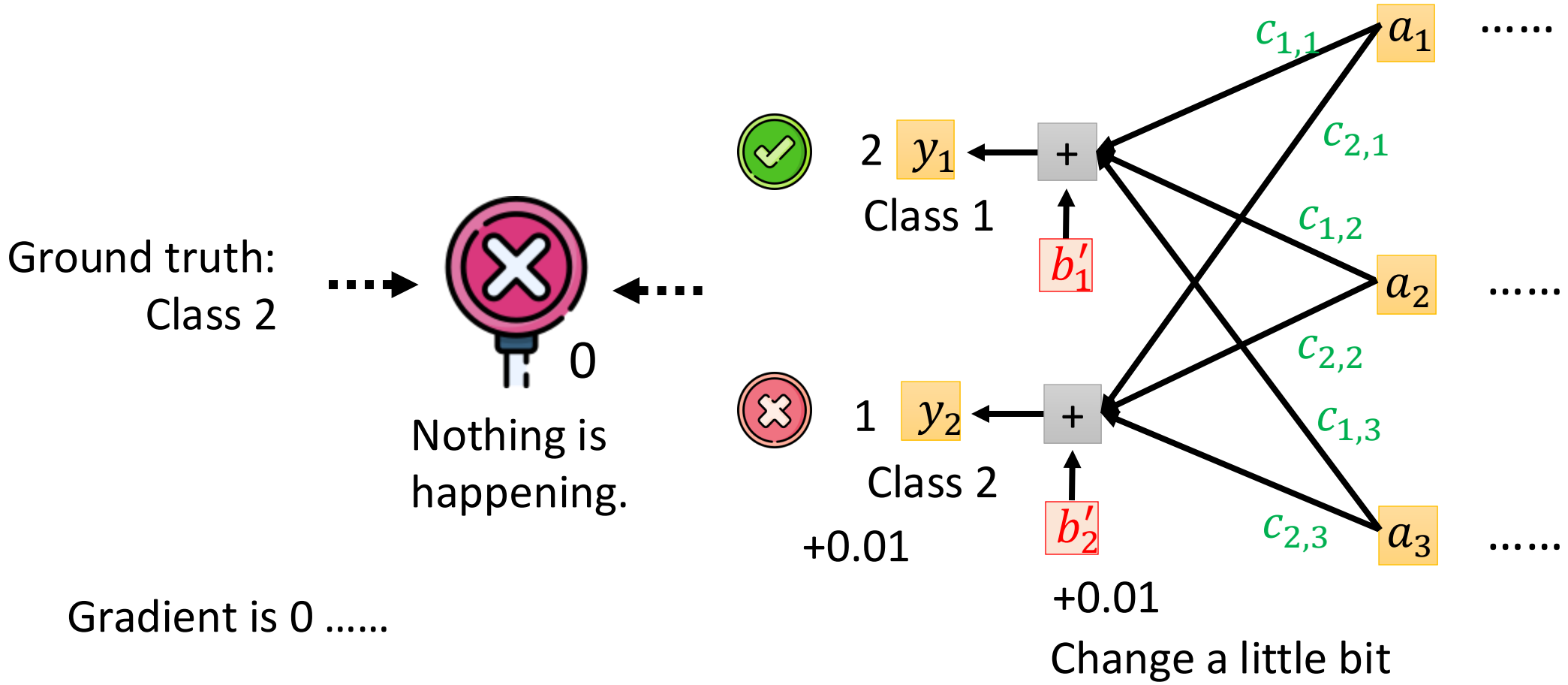
Classification



Loss = Evaluation Metrics?

Classification

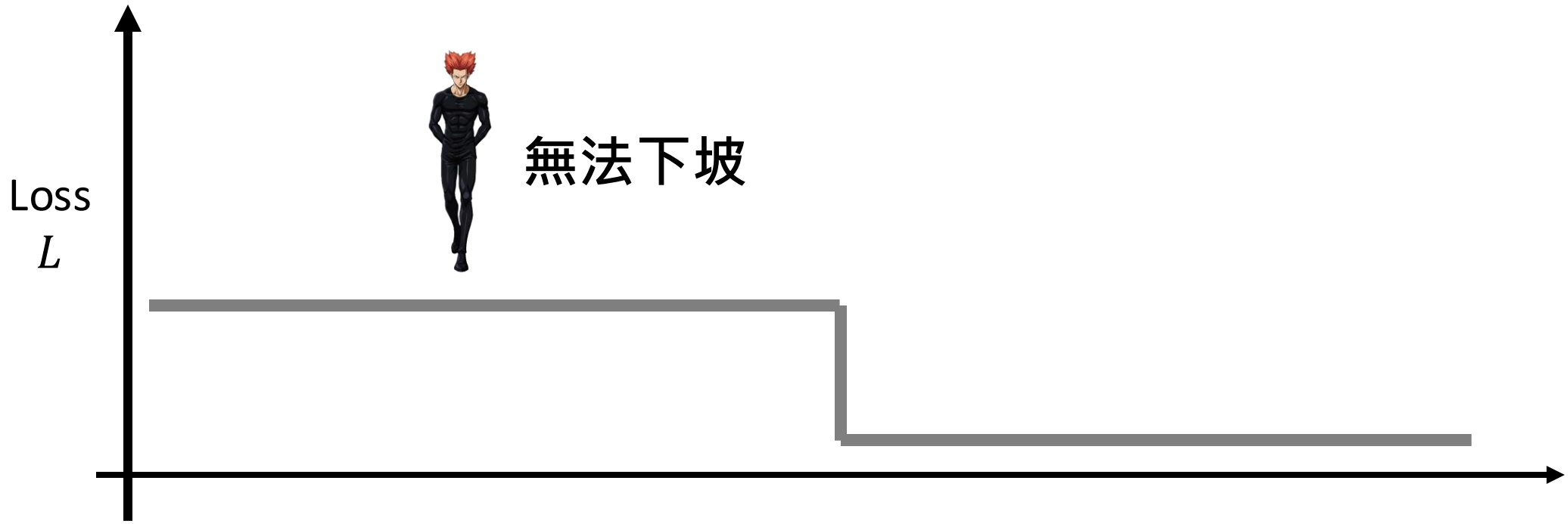
$$\boldsymbol{g} = \nabla L(\boldsymbol{\theta})$$



Gradient is 0 .....

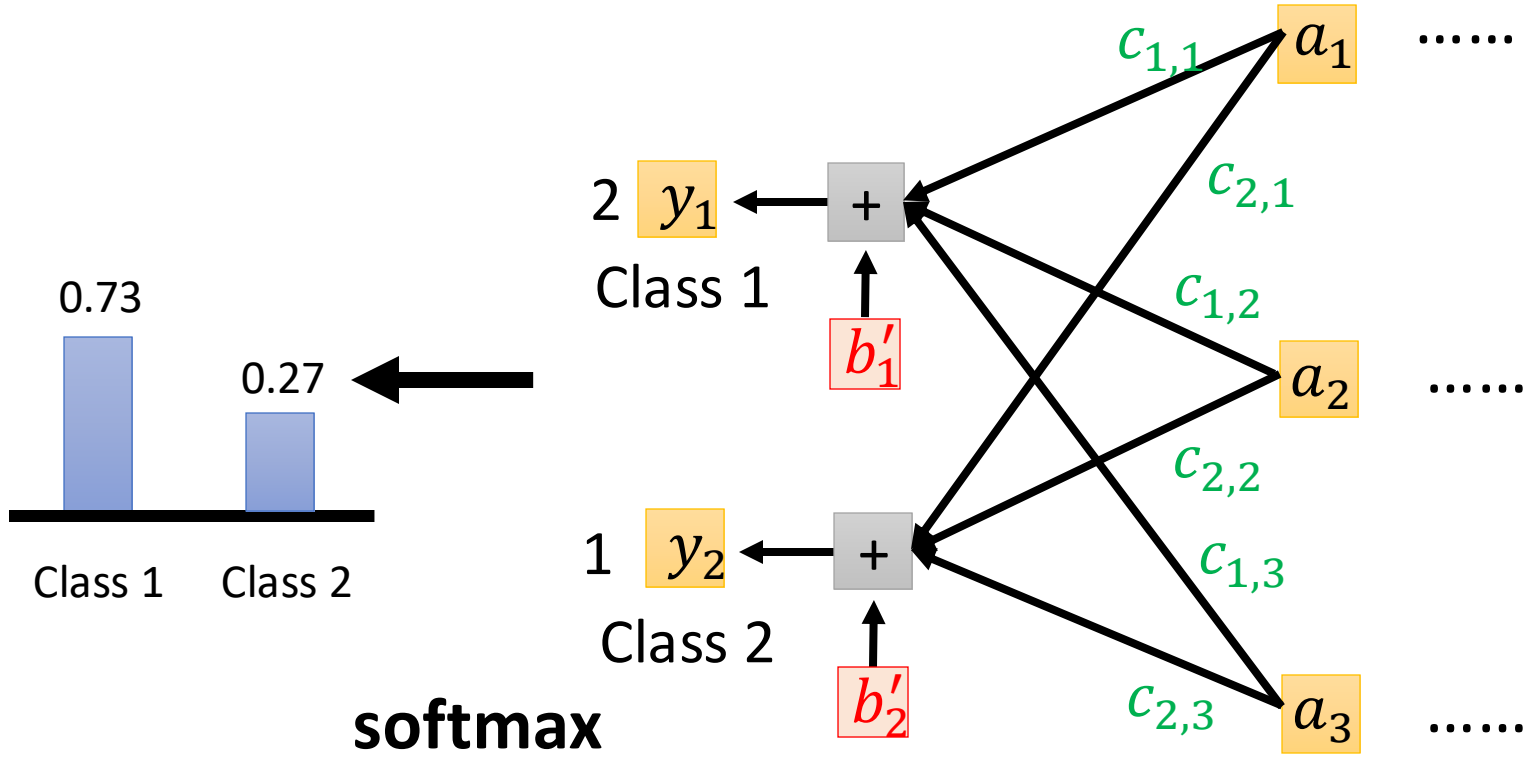
Loss = Evaluation Metrics?

- For classification, if we use accuracy as the loss function …



Cross-entropy as the loss in classification

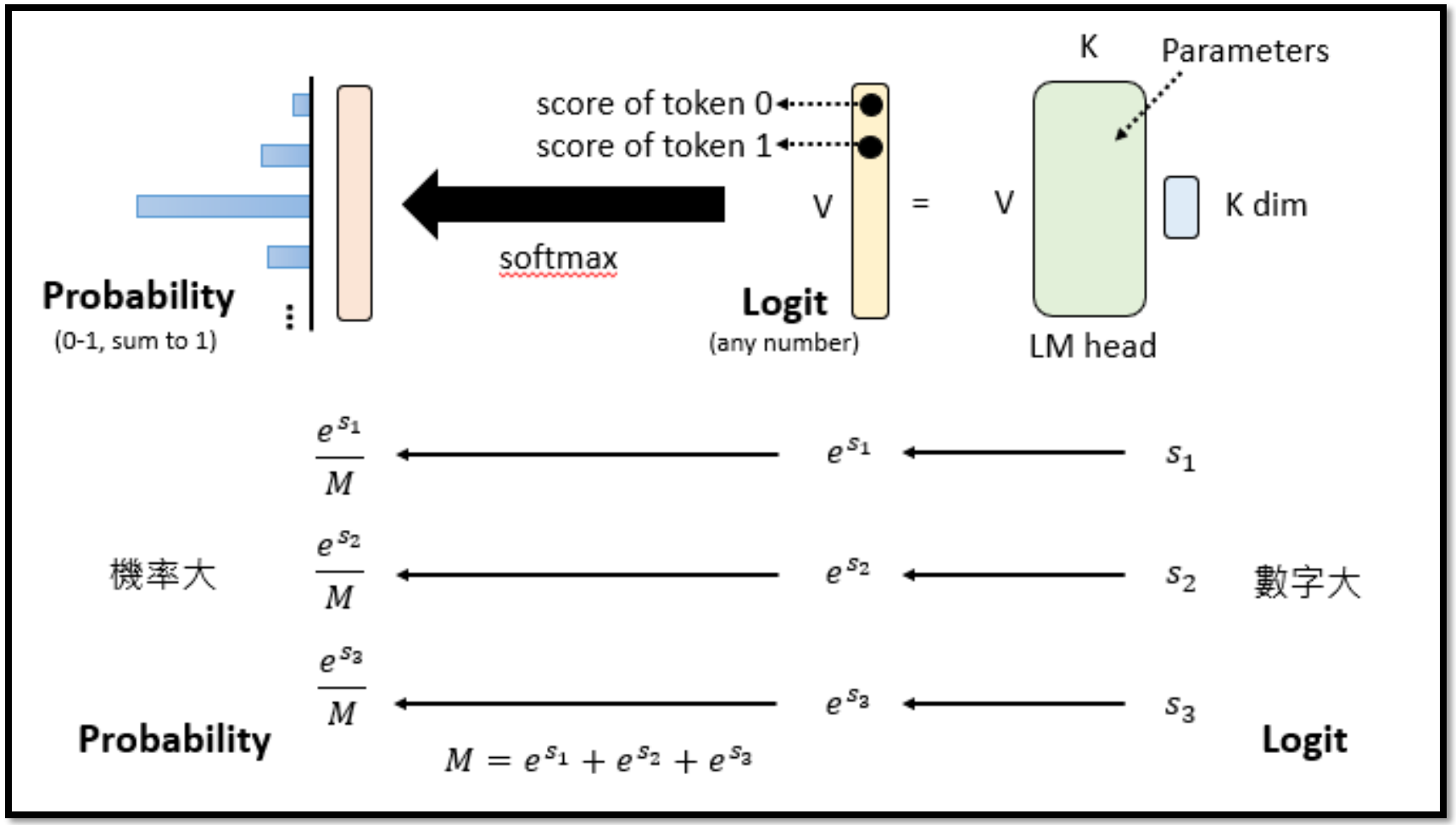
Ground truth: Class 2



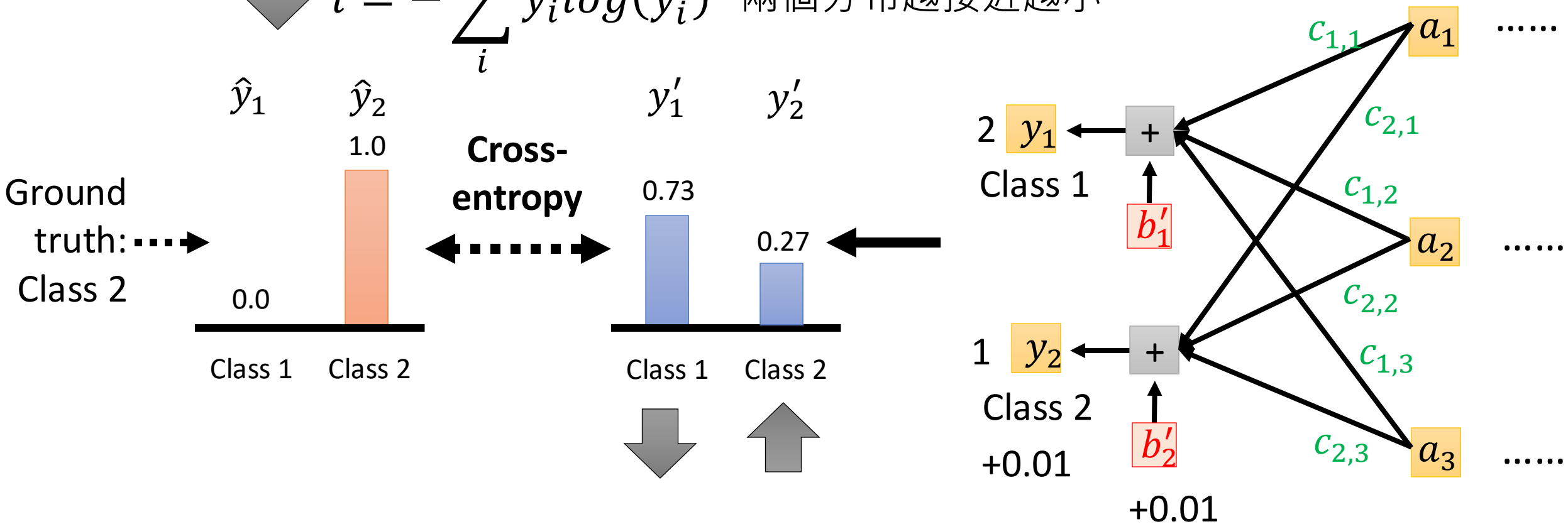
什麼是 Softmax?

請見本課程第三講

https://youtu.be/8iFvM7WUUs8?si=VdfmHEdUEm1TJOlR



$$l = -\sum_i \hat{y}_i log(y_i')$$ 兩個分布越接近越小



Why Cross-entropy?

How to do Classification

- Training data for Classification

$(x^1, \hat{y}^1)$ $(x^2, \hat{y}^2)$ …… $(x^N, \hat{y}^N)$

Classification as Regression?

Binary classification as example

Training: Class 1 means the target is 1; Class 2 means the target is -1

Testing: closer to 1 → class 1; closer to -1 → class 2

ML Lecture 4: Classification

https://youtu.be/fZAZUYEeIMg

	Logistic Regression	***Linear Regression***
Step 1:	$f_{w,b}(x) = \sigma\left(\sum_i w_i x_i + b\right)$	$f_{w,b}(x) = \sum_i w_i x_i + b$
	Output: between 0 and 1	Output: any value
Step 2:	Training data: $(x^n, \hat{y}^n)$	Training data: $(x^n, \hat{y}^n)$
	$\hat{y}^n$: 1 for class 1, 0 for class 2	$\hat{y}^n$: a real number
	$L(f) = \sum_n C(f(x^n), \hat{y}^n)$	$L(f) = \frac{1}{2}\sum_n (f(x^n) - \hat{y}^n)^2$

Cross entropy:

$$C(f(x^n), \hat{y}^n) = -\left[\hat{y}^n ln f(x^n) + (1 - \hat{y}^n) ln\left(1 - f(x^n)\right)\right]$$

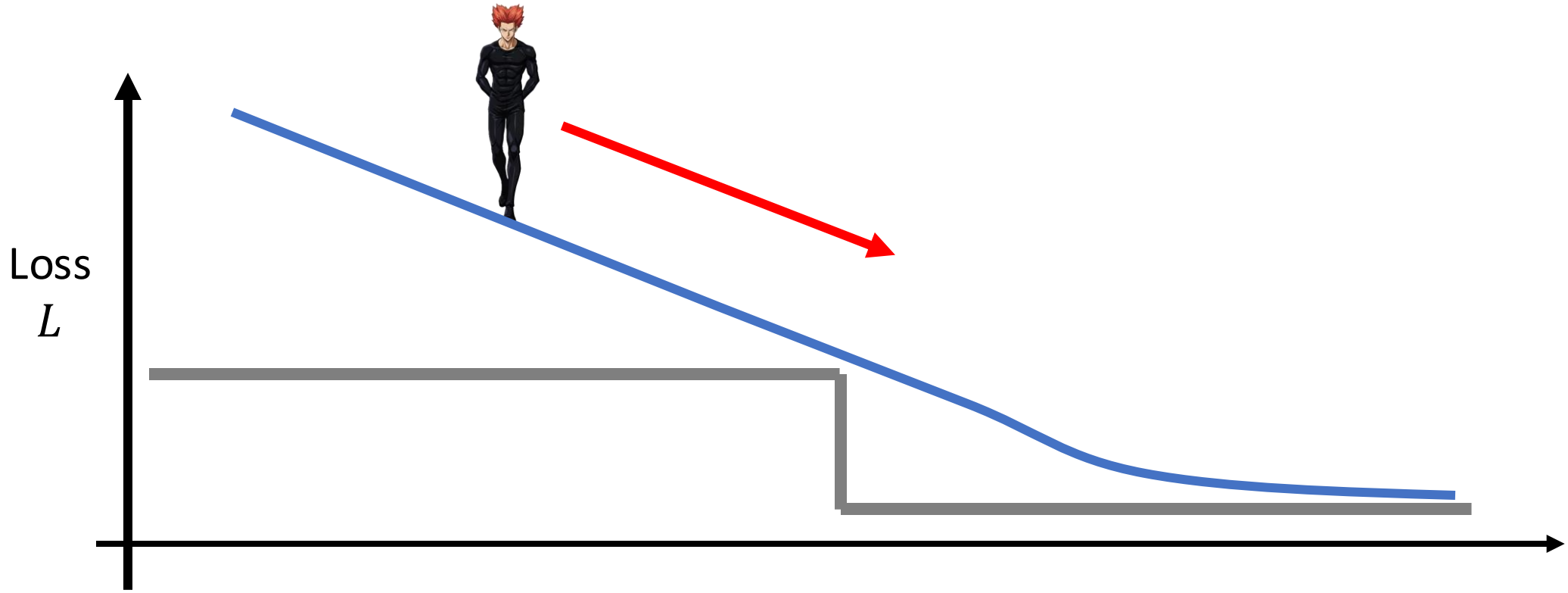
Question: Why don't we simply use square error as linear regression?

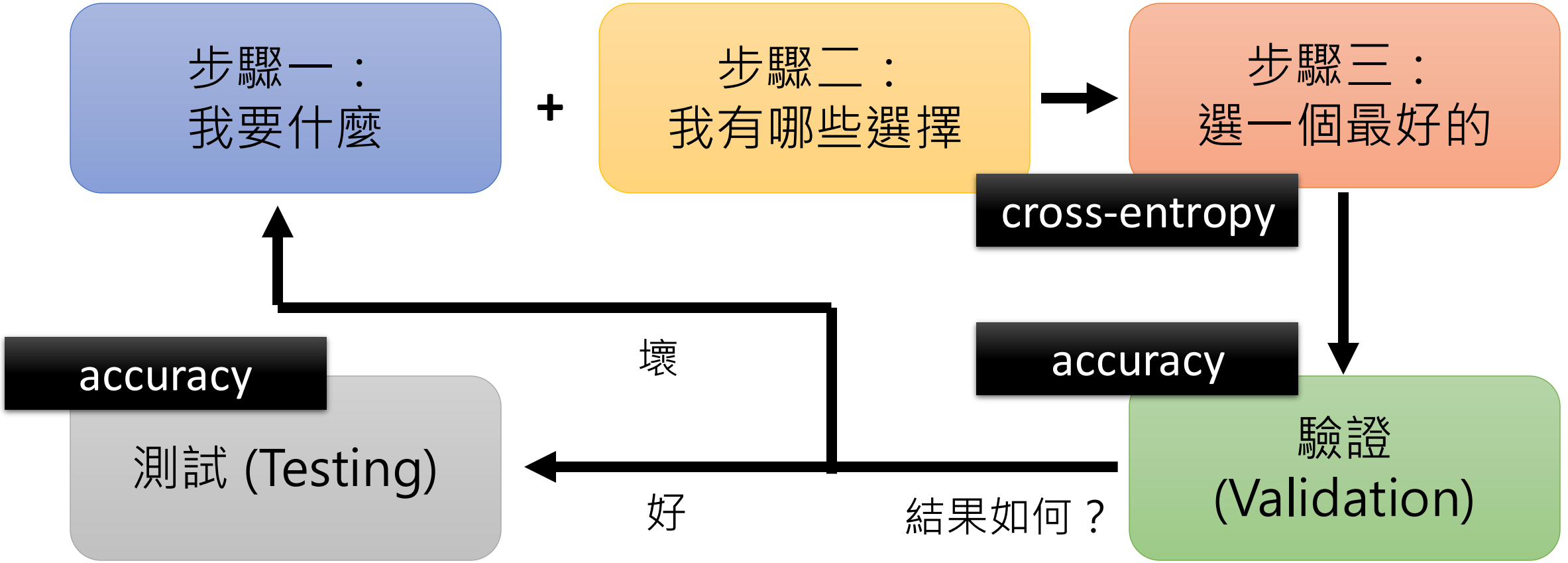
ML Lecture 5: Logistic Regression

https://youtu.be/hSXFuypLukA

Loss = Evaluation Metrics?

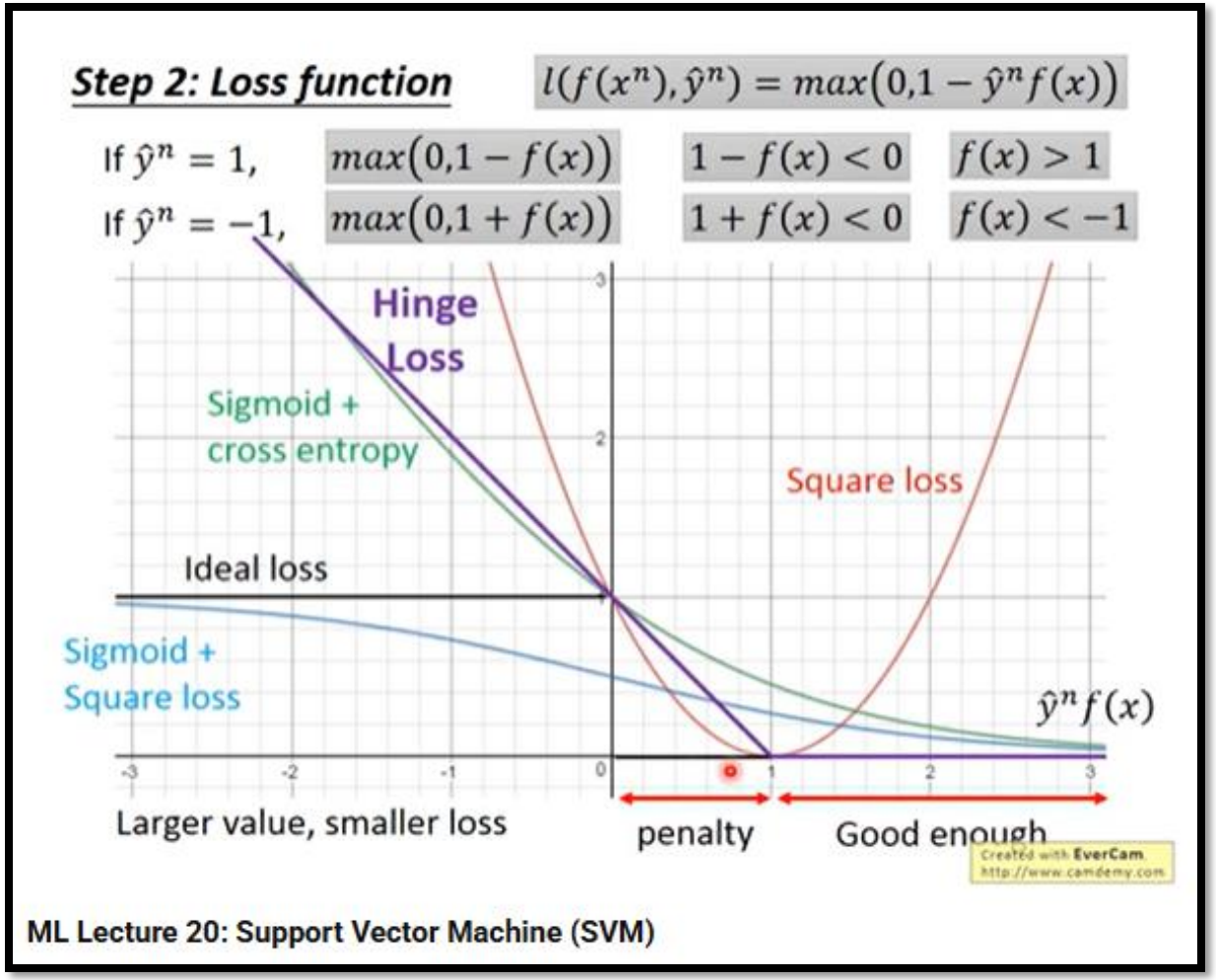
- For classification, if we use cross-entropy as the loss function …





方法名	改了那一個步驟	帶來什麼好處
Do not use accuracy as loss	我要找甚麼	Better Optimization (Not for Generalization)

有沒有其他可能性？



https://youtu.be/QSEPStBgwRQ?si=5oBXckoLuqbaroKY

Need More Training Data …

Overfitting

$$L = \sum_{x \sim Train} l(f(x), \hat{y}) \quad \cdots\cdots\blacktriangleright \quad L = \sum_{x \sim Test} l(f(x), \hat{y})$$

Collecting more training data

(be careful about the representative)

方法名	改了那一個步驟	帶來什麼好處
More training data	我要找甚麼	Better Generalization Not for Optimization

Data Augmentation

Overfitting

$$L = \sum_{x \sim Train} l(f(x), \hat{y}) \quad \cdots\cdots\blacktriangleright \quad L = \sum_{x \sim Test} l(f(x), \hat{y})$$

Collecting more training data

bird

flip

bird

blurry

bird

Hello~

voice conversion

Hello~

change volume

Hello~

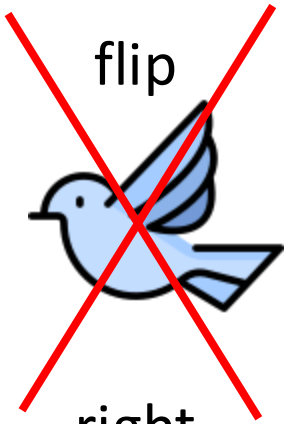
Be careful about what you augment

Overfitting

$$L = \sum_{x \sim Train} l(f(x), \hat{y}) \quad \cdots\cdots\blacktriangleright \quad L = \sum_{x \sim Test} l(f(x), \hat{y})$$

Collecting more training data

right



flip

right

blurry

right

Speaker X

voice conversion

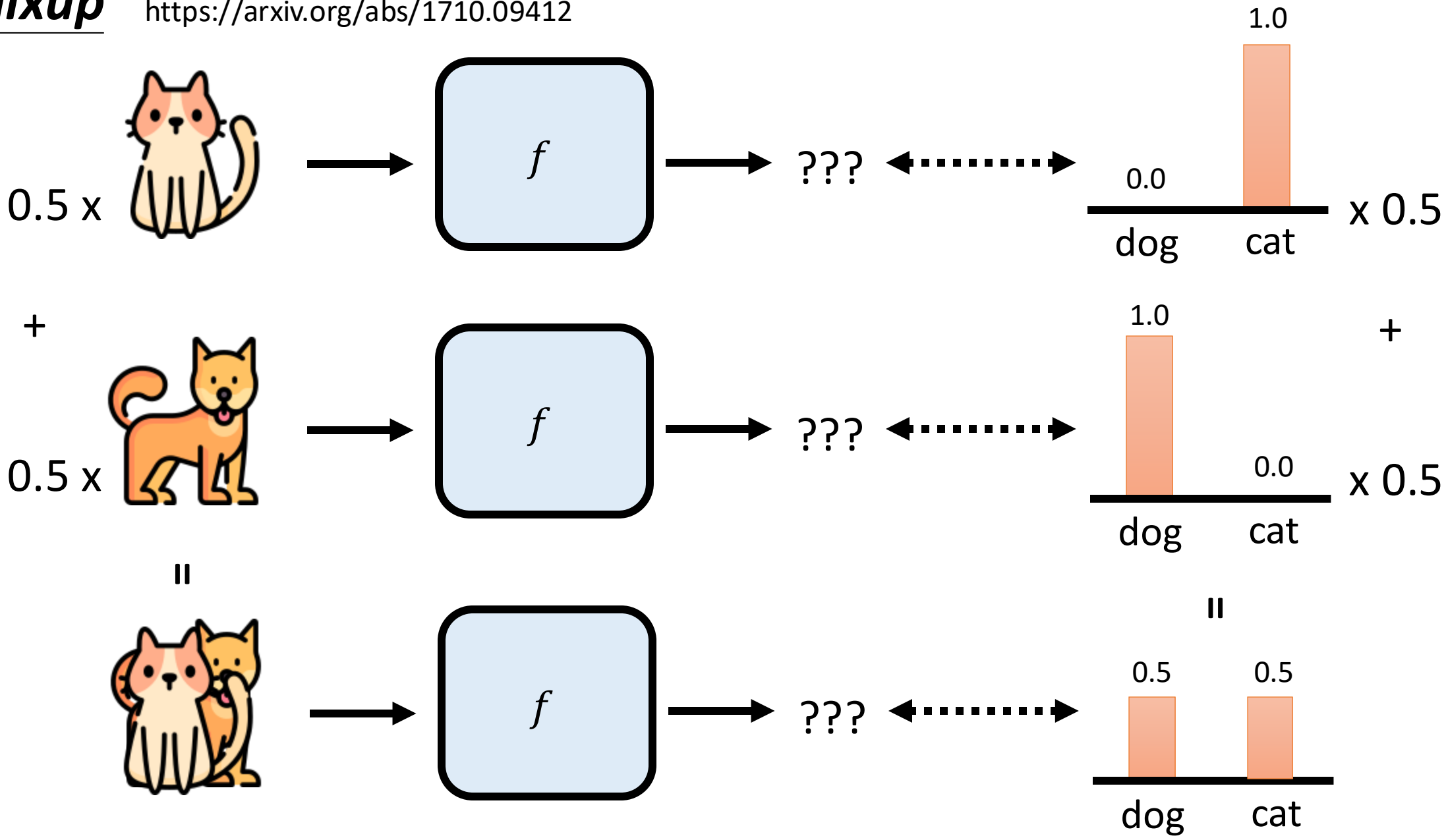
Speaker X

change volume

Speaker X

Mixup

https://arxiv.org/abs/1710.09412



Data Augmentation

Overfitting

$$L = \sum_{x \sim Train} l(f(x), \hat{y}) \quad \cdots\cdots\blacktriangleright \quad L = \sum_{x \sim Test} l(f(x), \hat{y})$$

Collecting more training data

(be careful about the representative)

方法名	改了那一個步驟	帶來什麼好處
Data Augmentation (e.g. Mixup)	我要找甚麼	Better Generalization Not for Optimization

Semi-supervised Learning


bird


cat


dog

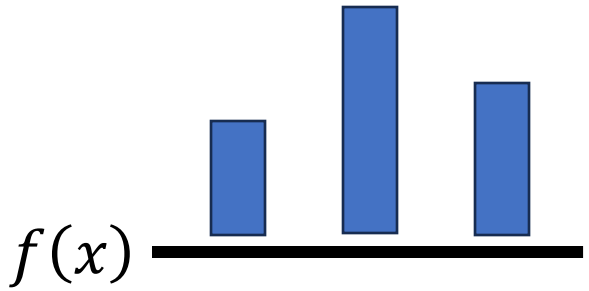
Unlimited
unlabeled data

$$L = \sum_{x \sim Train} l(f(x), \hat{y}) \quad + \lambda \sum_{x \sim Unlabel} ?????$$

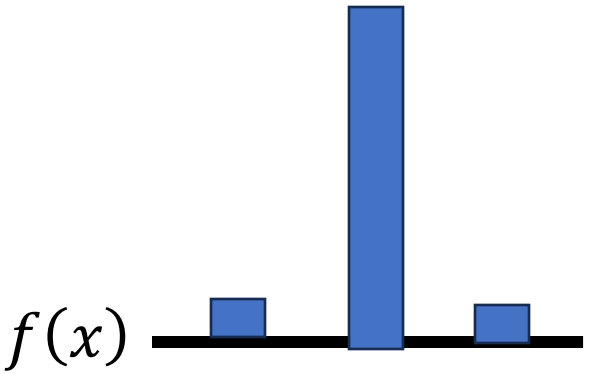
$$L = \sum_{x \sim Train} l(f(x), \hat{y}) + \lambda \sum_{x \sim Unlabel} l'(f(x))$$

entropy　distribution

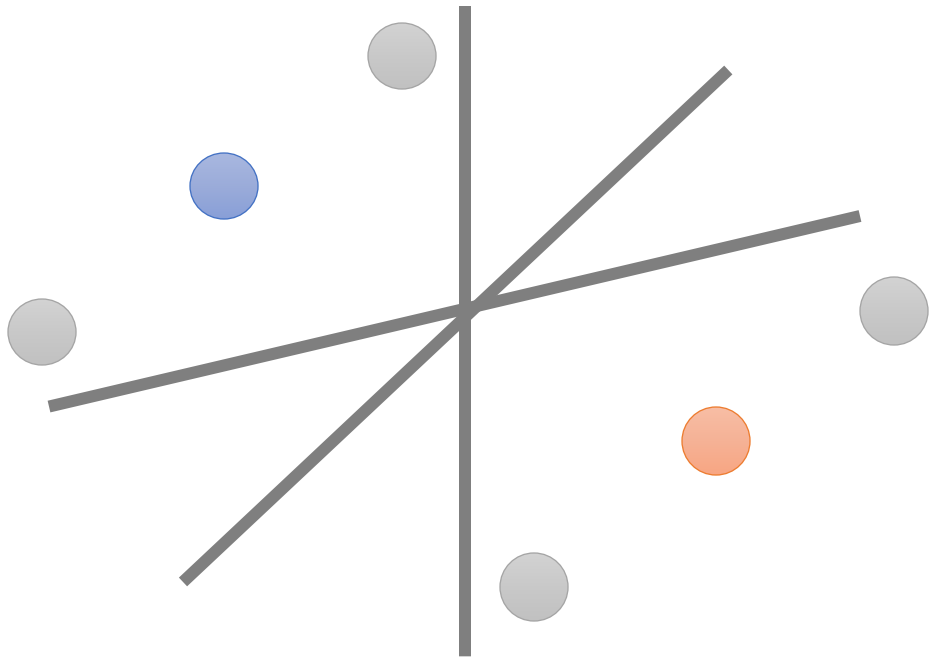
larger entropy



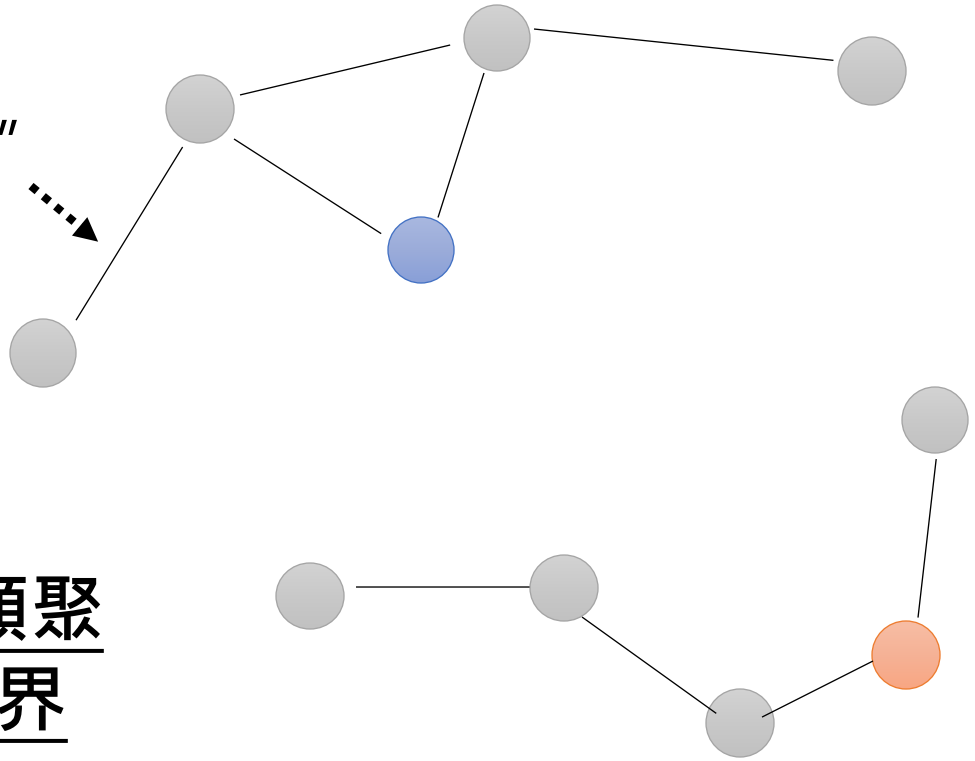
smaller entropy



非黑即白的世界



$$L = \sum_{x \sim Train} l(f(x), \hat{y}) + \lambda \sum_{x,x' \sim Unlabel} g(x,x') l'\big(f(x), f(x')\big)$$



物以類聚的世界

$$g(x,x') = \begin{cases} 1 & connect \\ 0 & otherwise \end{cases}$$

$$l'\big(f(x), f(x')\big)$$

$f(x), f(x')$ 的 "距離"

Semi-supervised Learning

$$L = \sum_{x \sim Train} l(f(x), \hat{y}) + \lambda \sum_{x \sim Unlabel} l'(f(x))$$

$$L = \sum_{x \sim Train} l(f(x), \hat{y}) + \lambda \sum_{x, x' \sim Unlabel} g(x, x') l'\big(f(x), f(x')\big)$$

方法名	改了那一個步驟	帶來什麼好處
Semi-supervised (e.g., Entropy, Graph)	我要找甚麼	Better Generalization Not for Optimization

Parameter Regularization

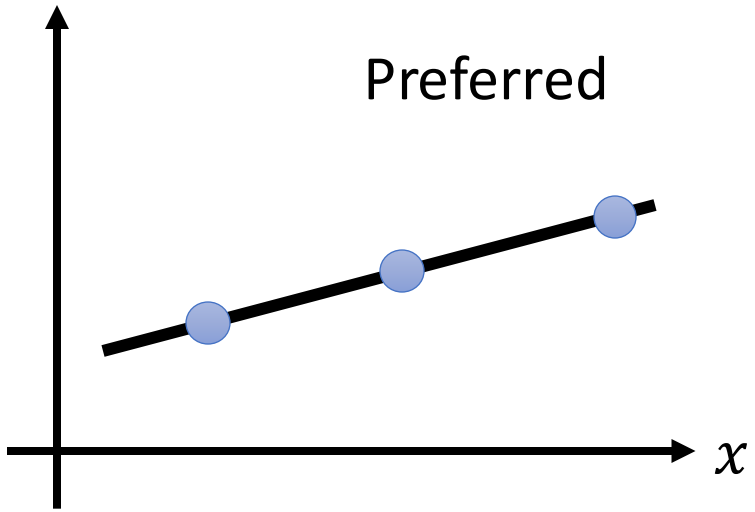
希望 Parameter 越接近 0 越好

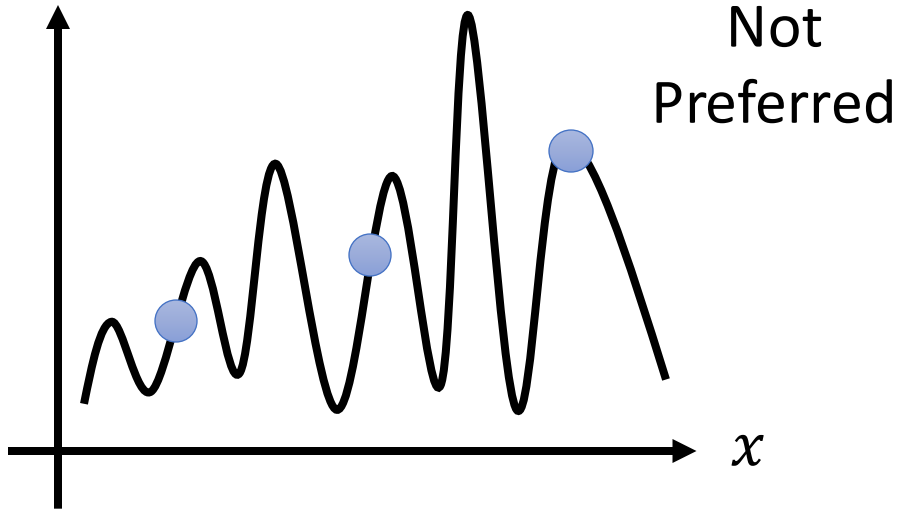
$$L'(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + \lambda R(\boldsymbol{\theta})$$

$$R(\boldsymbol{\theta}) = \sum_i (\theta_i)^2$$

跟資料有關

對於 Parameter 的偏好 (與資料無關)





Parameter Regularization

希望 Parameter 越接近 0 越好

$$L'(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + \lambda R(\boldsymbol{\theta})$$

$$R(\boldsymbol{\theta}) = \sum_i (\theta_i)^2$$

跟資料有關

對於 Parameter 的偏好 (與資料無關)

$$\boldsymbol{g} = \nabla L(\boldsymbol{\theta}) \qquad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \boldsymbol{g}$$

$$\boldsymbol{g}' = \nabla L'(\boldsymbol{\theta}) \qquad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \boldsymbol{g}' = \boldsymbol{\theta} - \eta(\boldsymbol{g} + 2\lambda\boldsymbol{\theta}) = \boldsymbol{\theta} - \eta\boldsymbol{g} - 2\eta\lambda\boldsymbol{\theta}$$

$$= \nabla L(\boldsymbol{\theta}) + \lambda \nabla R(\boldsymbol{\theta}) = \nabla L(\boldsymbol{\theta}) + 2\lambda\boldsymbol{\theta} \qquad = (1 - 2\eta\lambda)\boldsymbol{\theta} - \eta\boldsymbol{g}$$

$\boldsymbol{g}$

weight decay

Parameter Regularization

希望 Parameter 越接近 0 越好

$$L'(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + \lambda R(\boldsymbol{\theta})$$

$$R(\boldsymbol{\theta}) = \sum_i (\theta_i)^2$$

跟資料有關

對於 Parameter 的偏好 (與資料無關)

方法名	改了那一個步驟	帶來什麼好處
Parameter Regularization	我要找甚麼	Better Generalization Not for Optimization
Weight Decay	找最好的函式	Better Generalization Not for Optimization

AdamW

https://arxiv.org/abs/1711.05101

Momentum

For each dimension i: $m_i^t = \beta m_i^{t-1} + (1-\beta) g_i^t$

RMSProp

For each dimension i: $\sigma_i^t = \sqrt{\alpha \left(\sigma_i^{t-1}\right)^2 + (1-\alpha)\left(g_i^t\right)^2}$

Adam

$$\theta_i^{t+1} \leftarrow \theta_i^t - \frac{\eta^t}{\sigma_i^t} m_i^t$$

AdamW

$$\theta_i^{t+1} \leftarrow (1 - 2\eta^t \lambda)\theta_i^t - \frac{\eta^t}{\sigma_i^t} m_i^t$$

方法名	改了那一個步驟	帶來什麼好處
AdamW	找最好的函式	Better Generalization (cf. Adam) Better Optimization (cf. Vanilla)

方法名	改了那一個步驟	帶來什麼好處
Adagrad, RMSProp, Momentum, Adam, etc.	找最好的函式	Better Optimization
AdamW	找最好的函式	Better Generalization (cf. Adam) Better Optimization (cf. Vanilla)
Dropout	找最好的函式	Better Generalization
Weight Decay	找最好的函式	Better Generalization
Initialization (e.g., pre-train)	找最好的函式	Better Optimization, Better Generalization
CNN (e.g., for image)	改變函式搜尋範圍	Better Generalization
Skip Connection	改變函式搜尋範圍	Better Optimization
Normalization	改變函式搜尋範圍	Better Optimization (Sometimes Better Generalization)
Do not use accuracy as loss	我要找什麼	Better Optimization
More training data	我要找什麼	Better Generalization
Data Augmentation (e.g. Mixup)	我要找什麼	Better Generalization
Semi-supervised (e.g., Entropy, Graph)	我要找什麼	Better Generalization
Parameter Regularization	我要找甚麼	Better Generalization