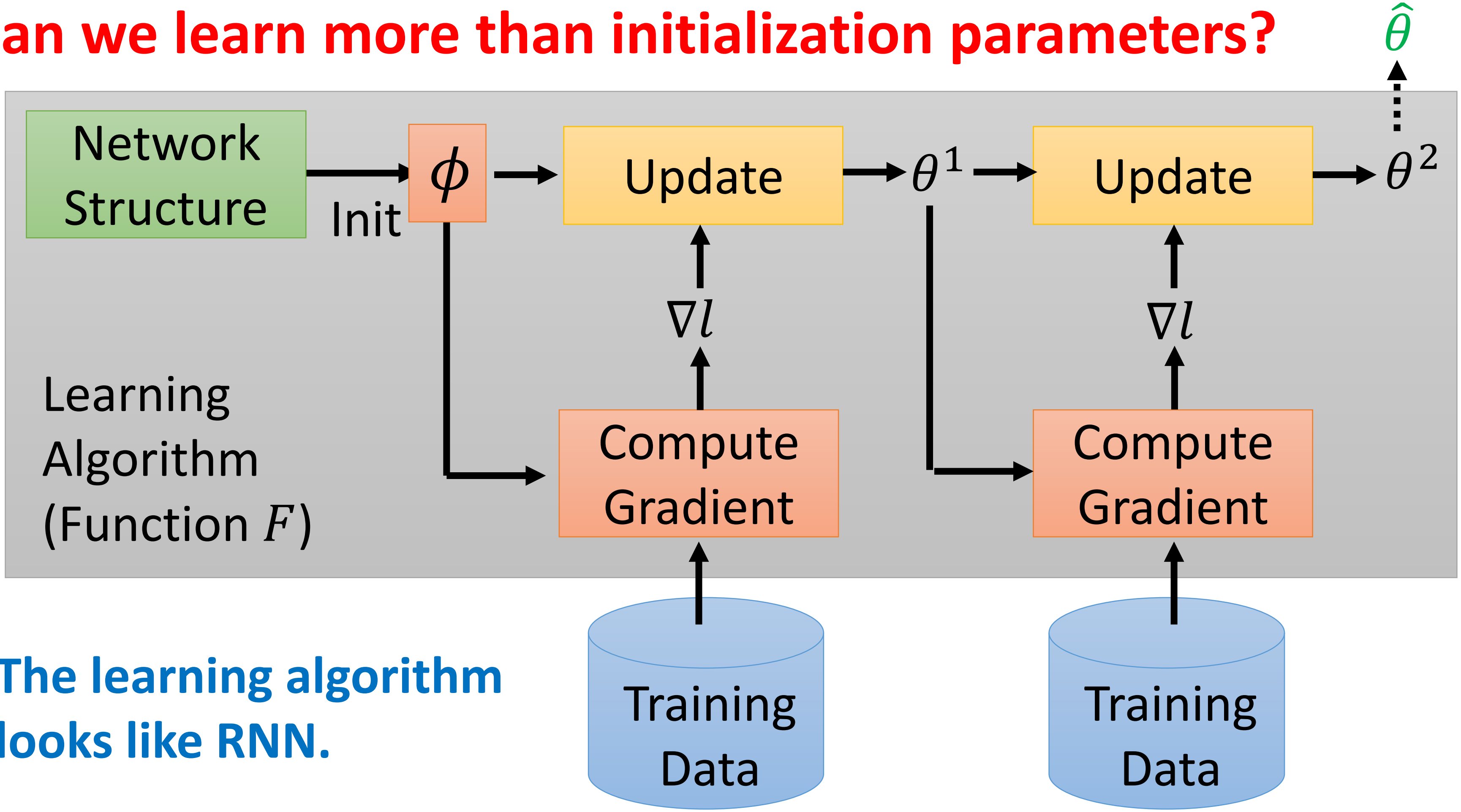
Meta Learning (Part 2): **Gradient Descent as LSTM**

Hung-yi Lee

Can we learn more than initialization parameters?



The learning algorithm looks like RNN.

OPTIMIZATION AS A MODEL FOR FEW-SHOT LEARNING

Sachin Ravi* and Hugo Larochelle
Twitter, Cambridge, USA
{sachinr,hugo}@twitter.com

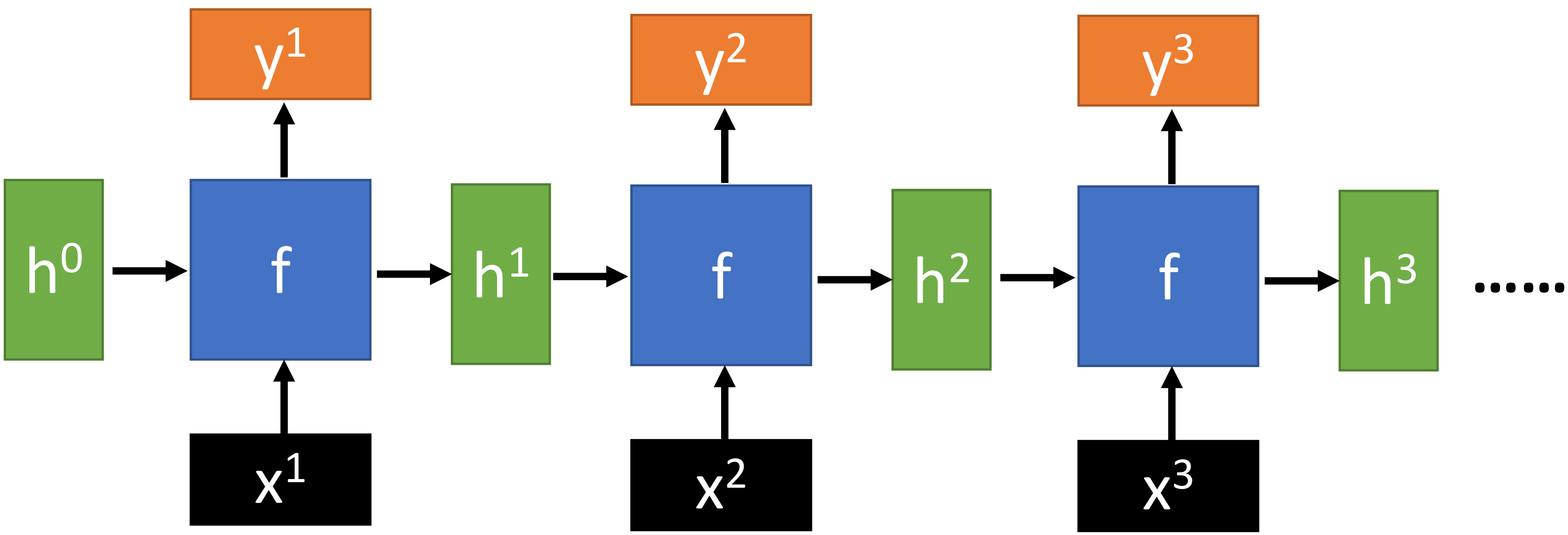
Learning to learn by gradient descent by gradient descent

Marcin Andrychowicz[1], Misha Denil[1], Sergio Gómez Colmenarejo[1], Matthew W. Hoffman[1], David Pfau[1], Tom Schaul[1], Brendan Shillingford[1,2], Nando de Freitas[1,2,3]
[1]Google DeepMind [2]University of Oxford [3]Canadian Institute for Advanced Research

Recurrent Neural Network

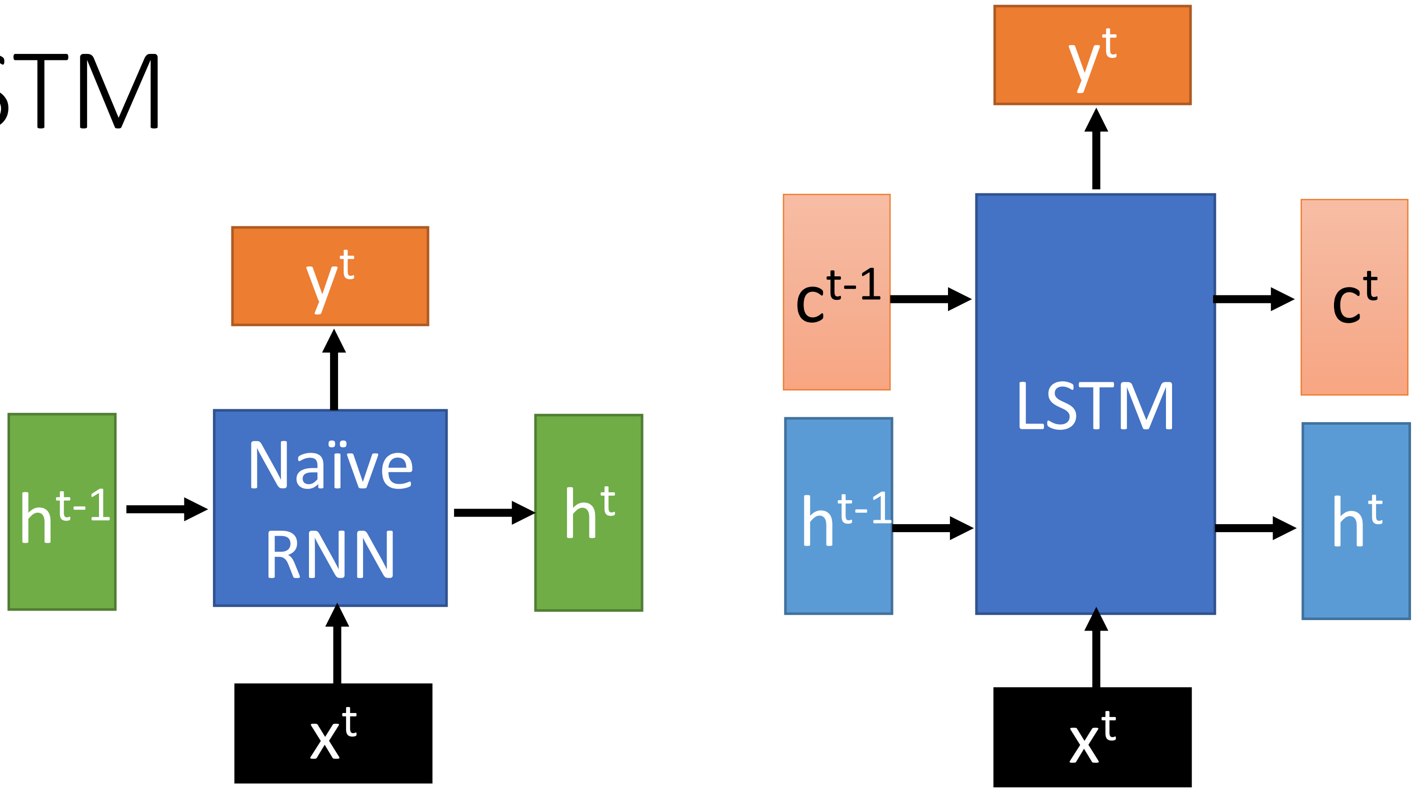
- Given function f: $h', y = f(h, x)$

h and h' are vectors with the same dimension



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

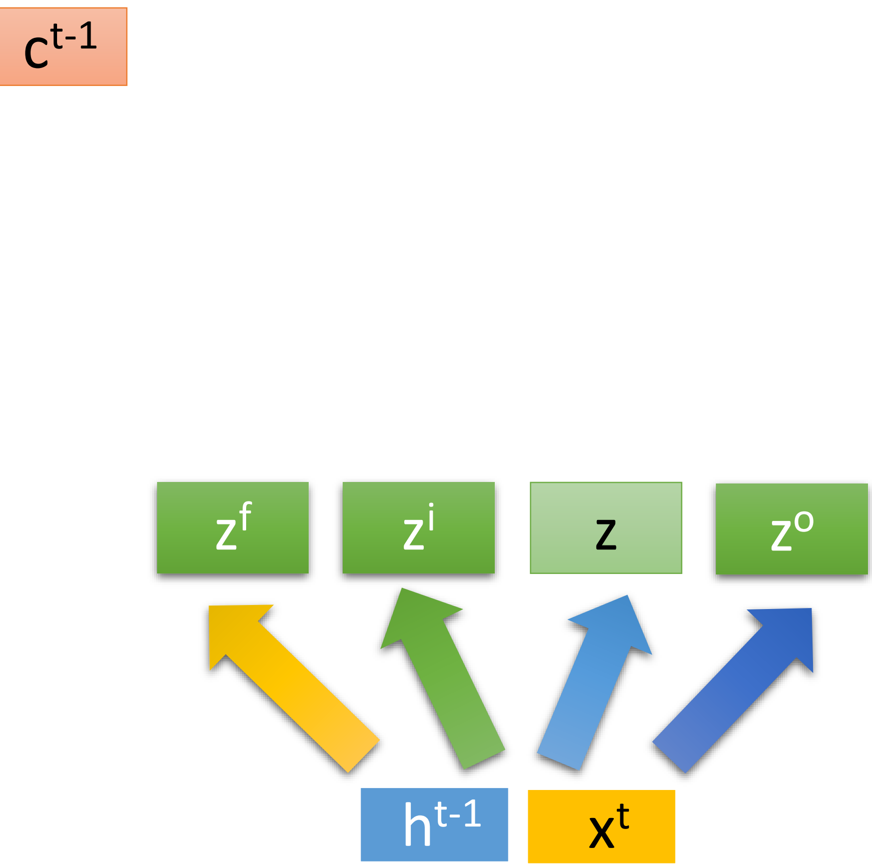
LSTM



c change slowly ➡ c^t is c^{t-1} added by something

h change faster ➡ h^t and h^{t-1} can be very different

Review: LSTM



$$z = tanh\left(W \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix} \right)$$

$$z^i = \sigma\left(W^i \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix} \right)$$

input

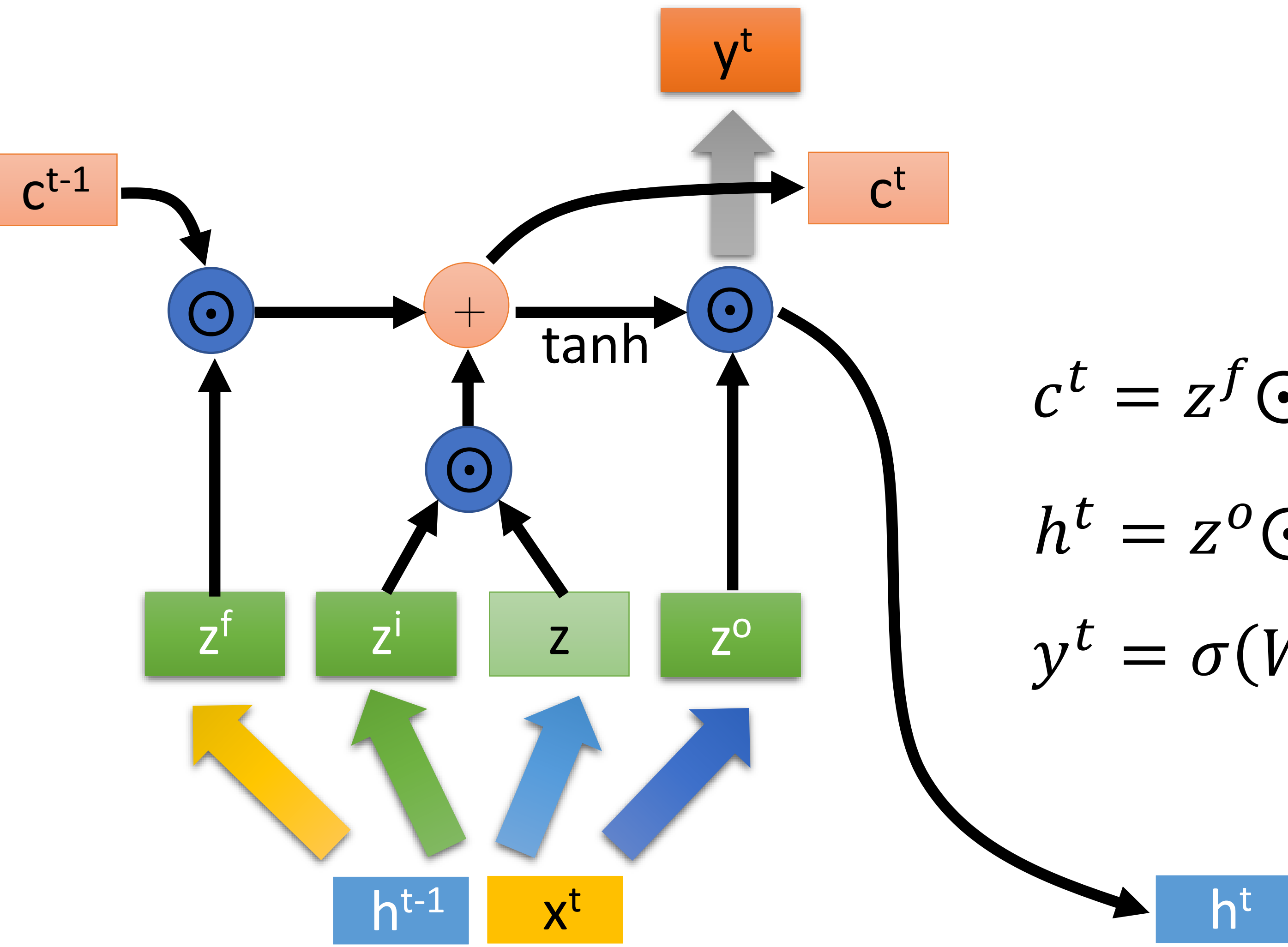
$$z^f = \sigma\left(W^f \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix} \right)$$

forget

$$z^o = \sigma\left(W^o \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix} \right)$$

output

Review: LSTM

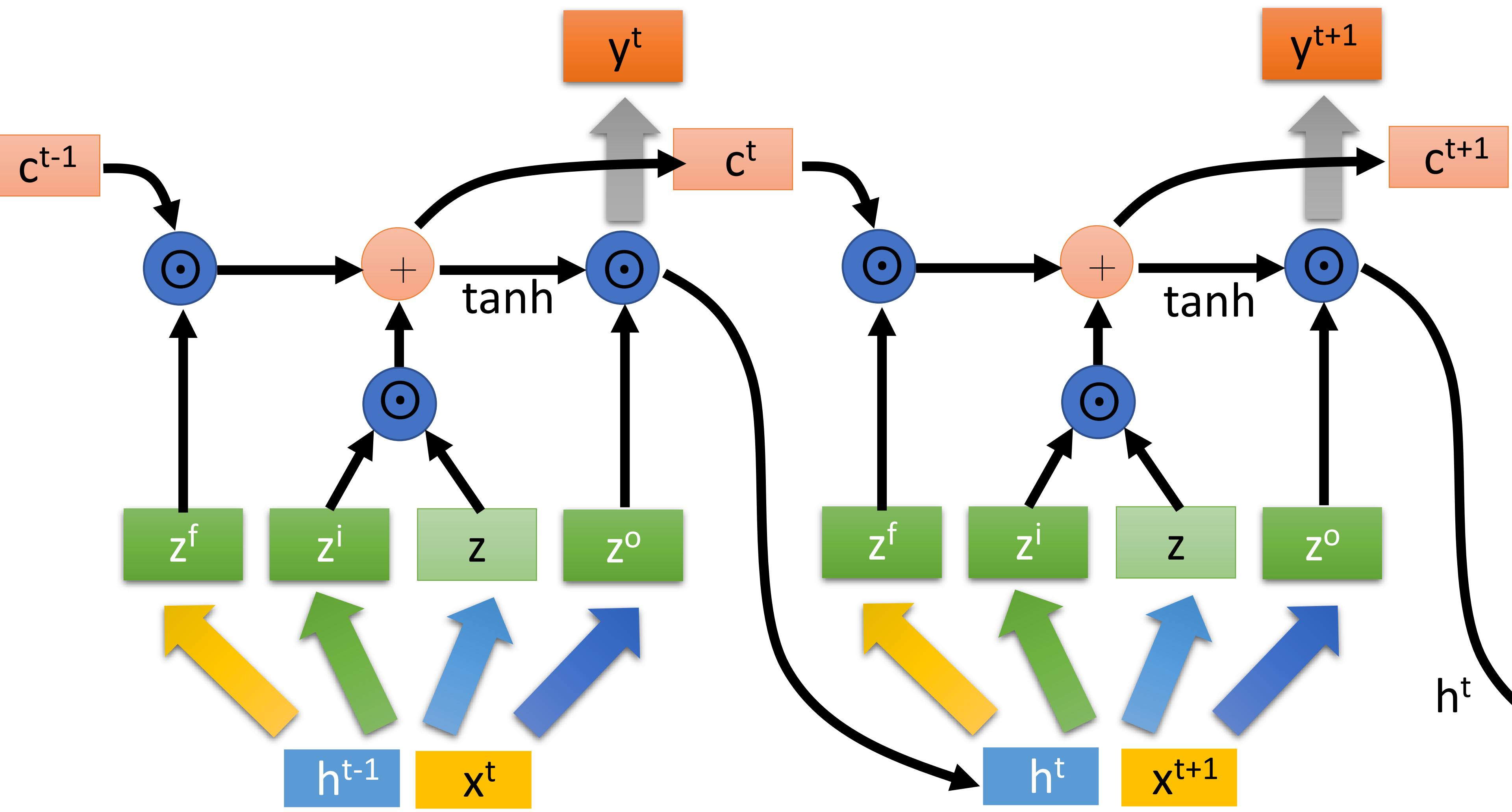


$$c^t = z^f \odot c^{t-1} + z^i \odot z$$

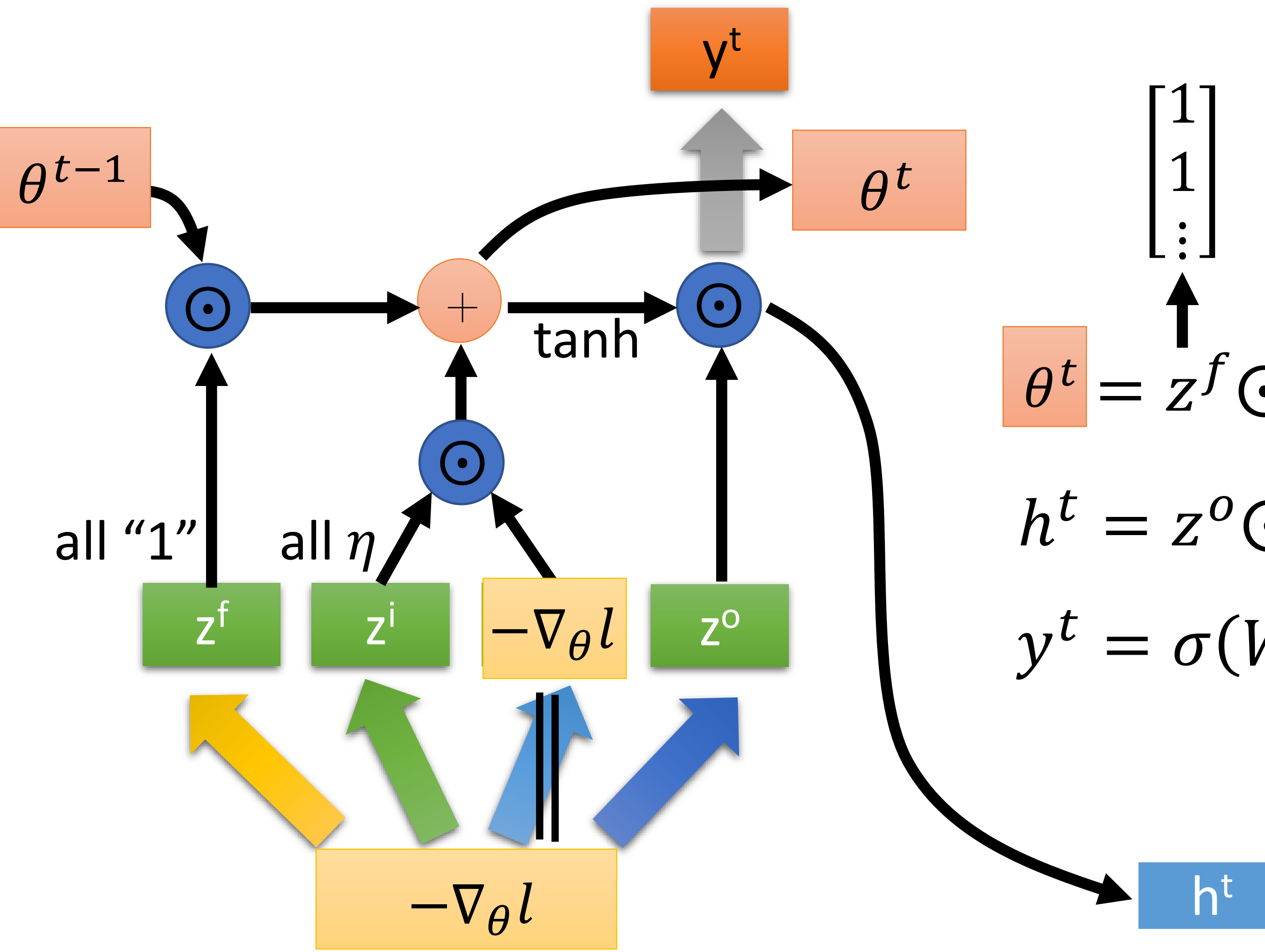
$$h^t = z^o \odot tanh(c^t)$$

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

Review: LSTM



Similar to gradient descent based algorithm



$$\theta^t = \theta^{t-1} - \eta \nabla_\theta l$$

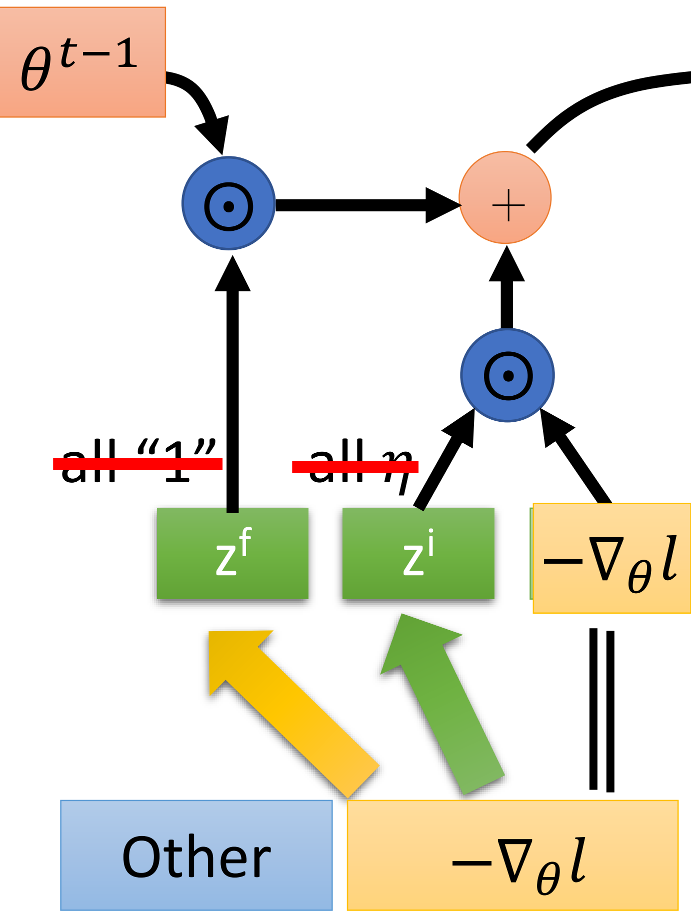
$$\boxed{\theta^t} = \overset{\begin{bmatrix}1\\1\\\vdots\end{bmatrix}}{z^f} \odot \boxed{\theta^{t-1}} + \overset{\begin{bmatrix}\eta\\\eta\\\vdots\end{bmatrix}}{z^i} \odot \boxed{-\nabla_\theta l}$$

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

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

Similar to gradient descent based algorithm

$$\theta^t = \theta^{t-1} - \eta \nabla_\theta l$$



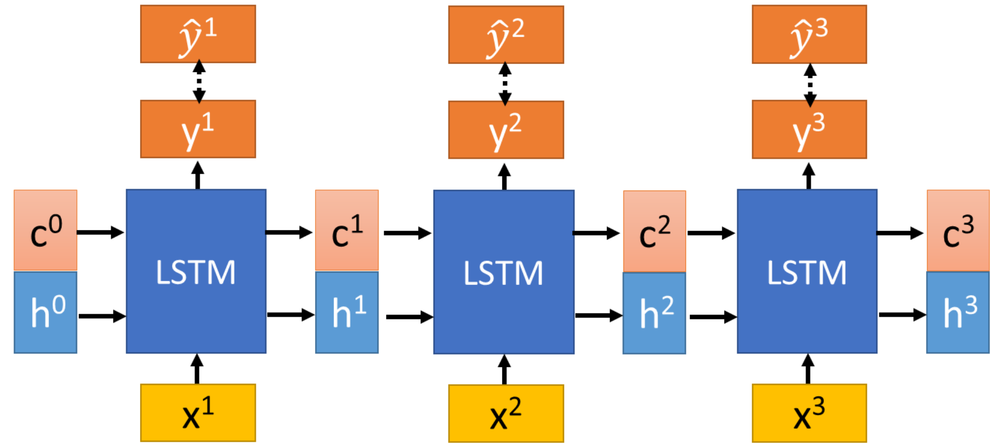
Something like regularization

$$\theta^t = z^f \odot \theta^{t-1} + z^i \odot -\nabla_\theta l$$

Dynamic learning rate

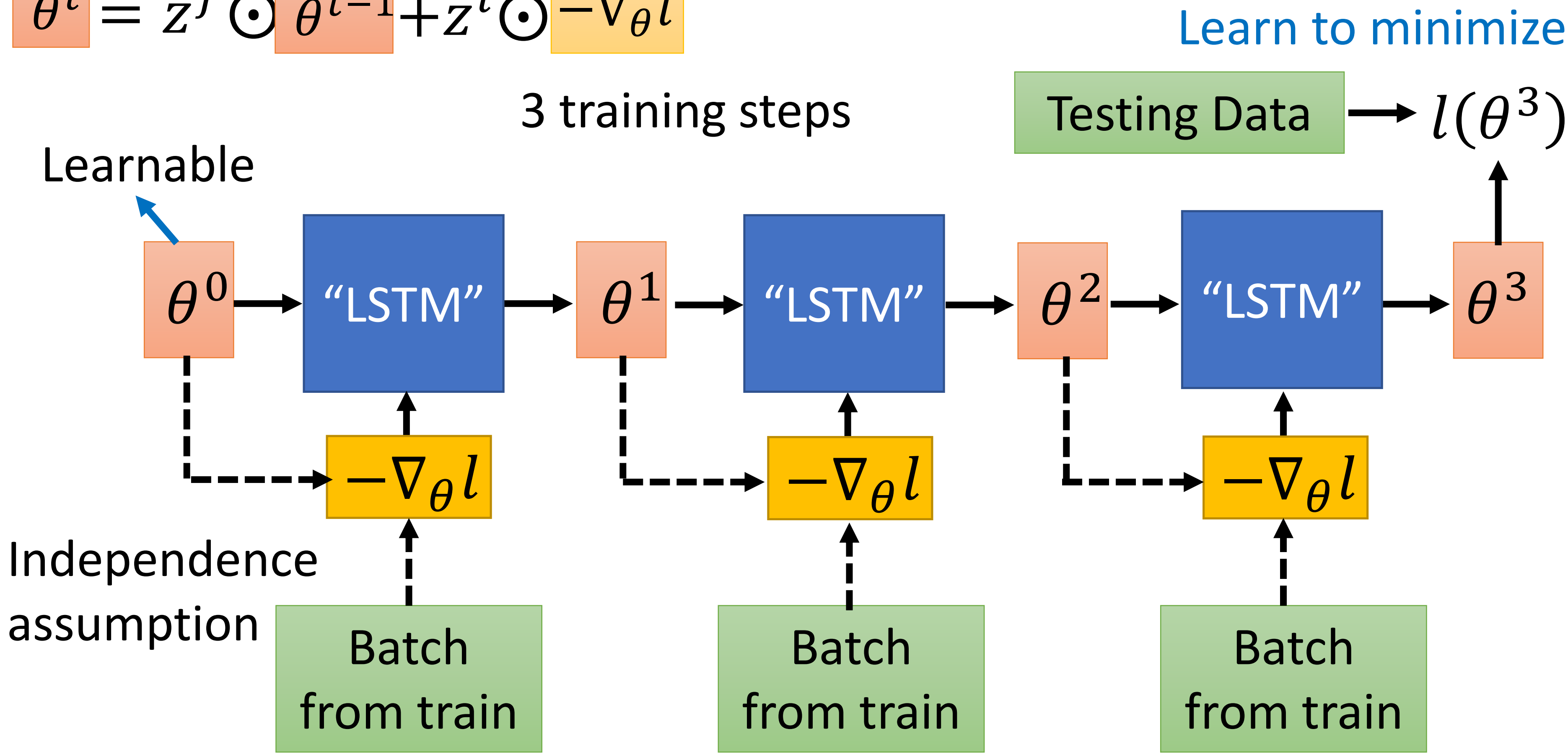
How about machine learn to determine z^f and z^i from $-\nabla_\theta l$ and other information?

Typical LSTM



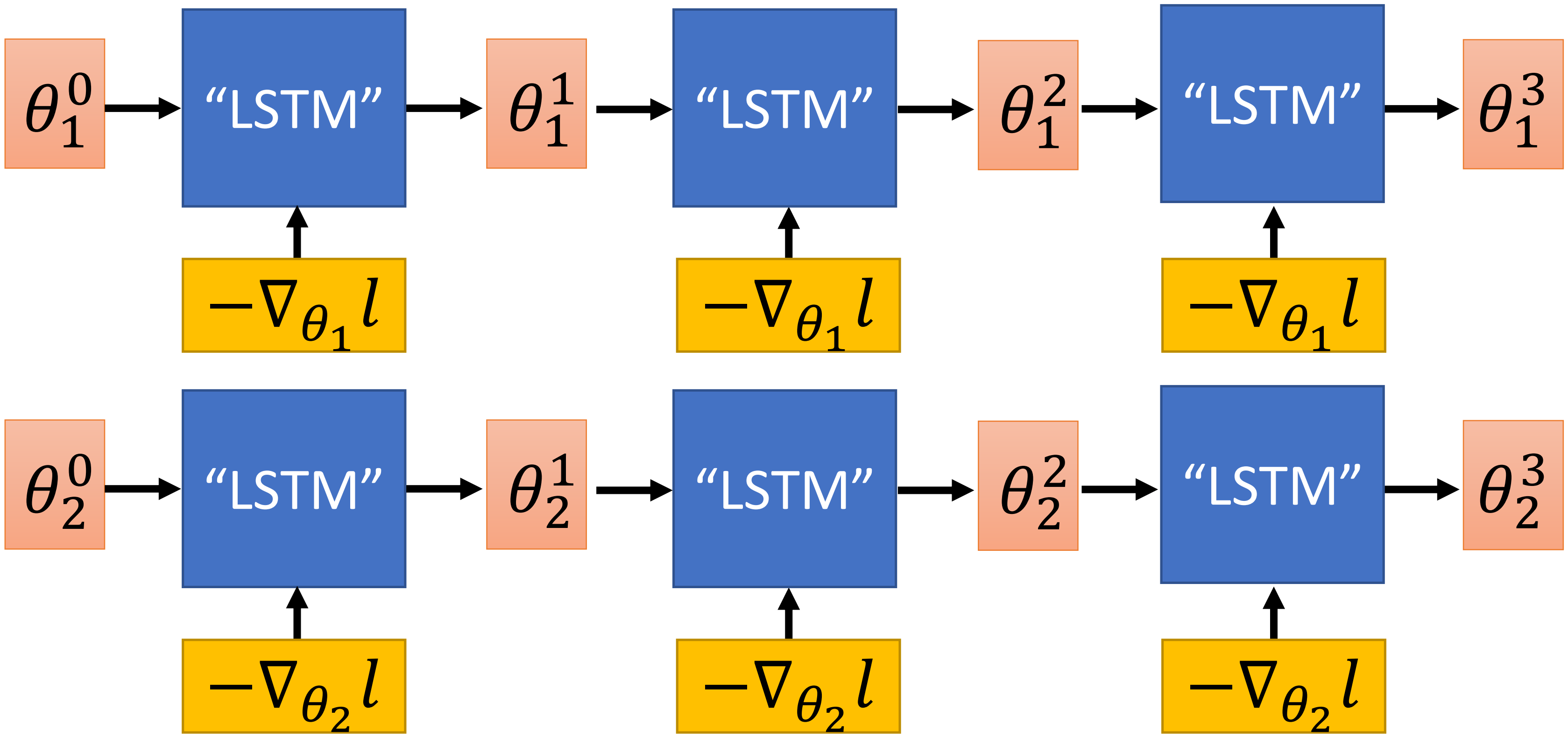
LSTM for Gradient Descent

$$\theta^t = z^f \odot \theta^{t-1} + z^i \odot -\nabla_\theta l$$



Real Implementation

The LSTM used only has one cell. Share across all parameters

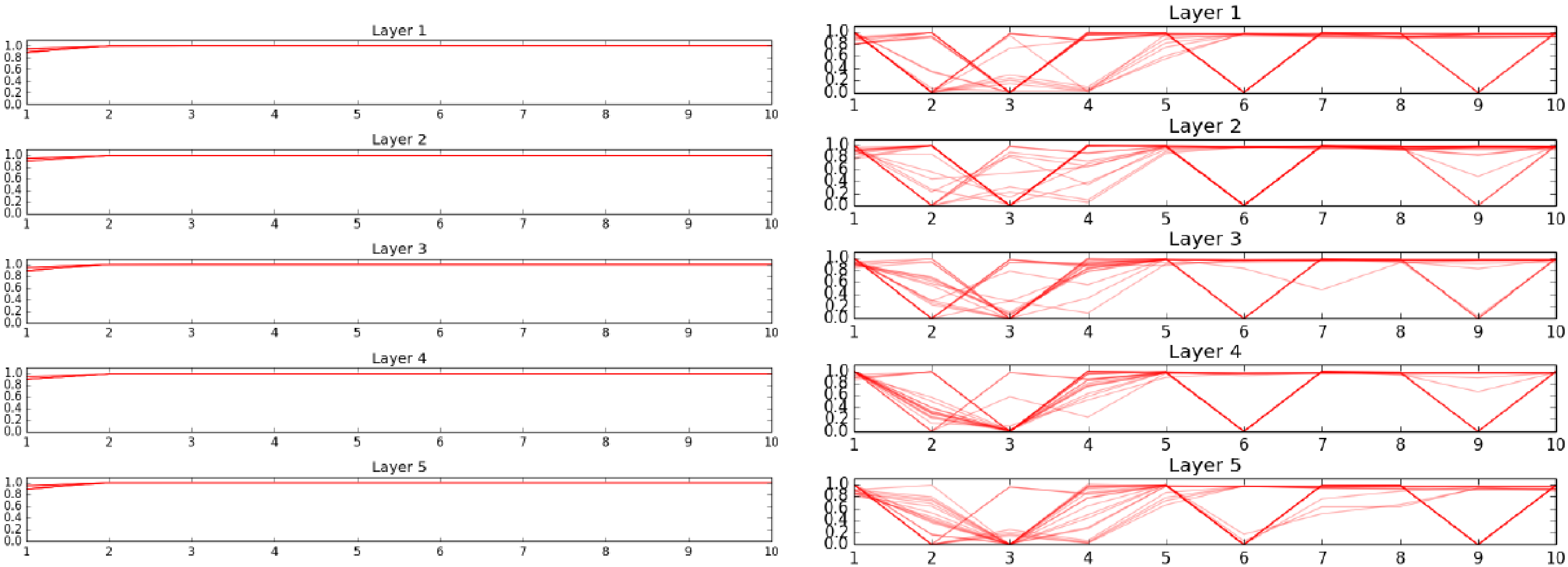


- Reasonable model size
- In typical gradient descent, all the parameters use the same update rule
- Training and testing model architectures can be different.

https://openreview.net/forum?id=rJY0-Kcll&noteId=ryq49XyLg

Experimental Results

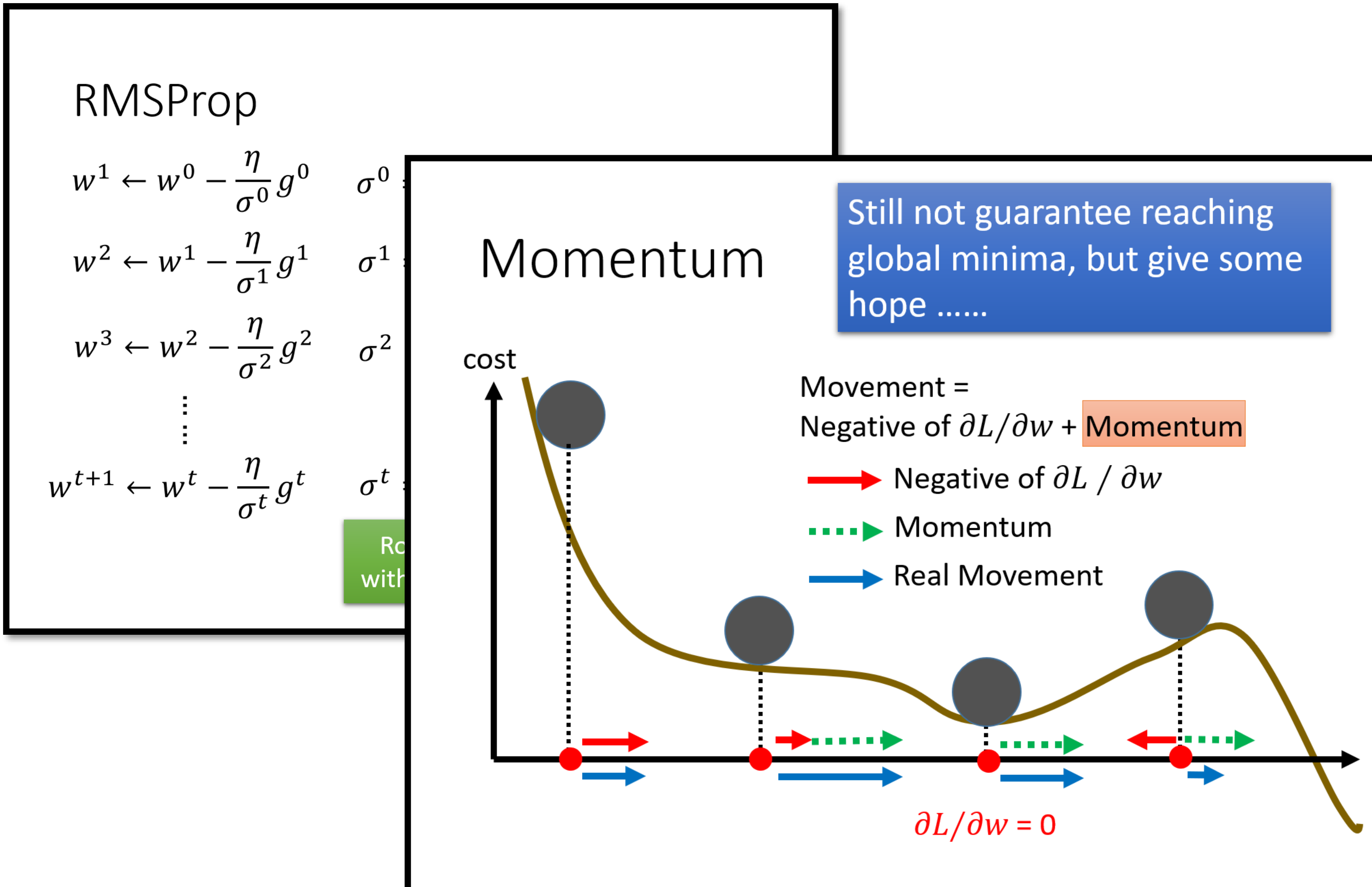
$$\boxed{\theta^{t}} = z^{f} \odot \boxed{\theta^{t-1}} + z^{i} \odot \boxed{-\nabla_{\theta} l}$$



(a) Forget gate values for 1-shot meta-learner

(b) Input gate values for 1-shot meta-learner

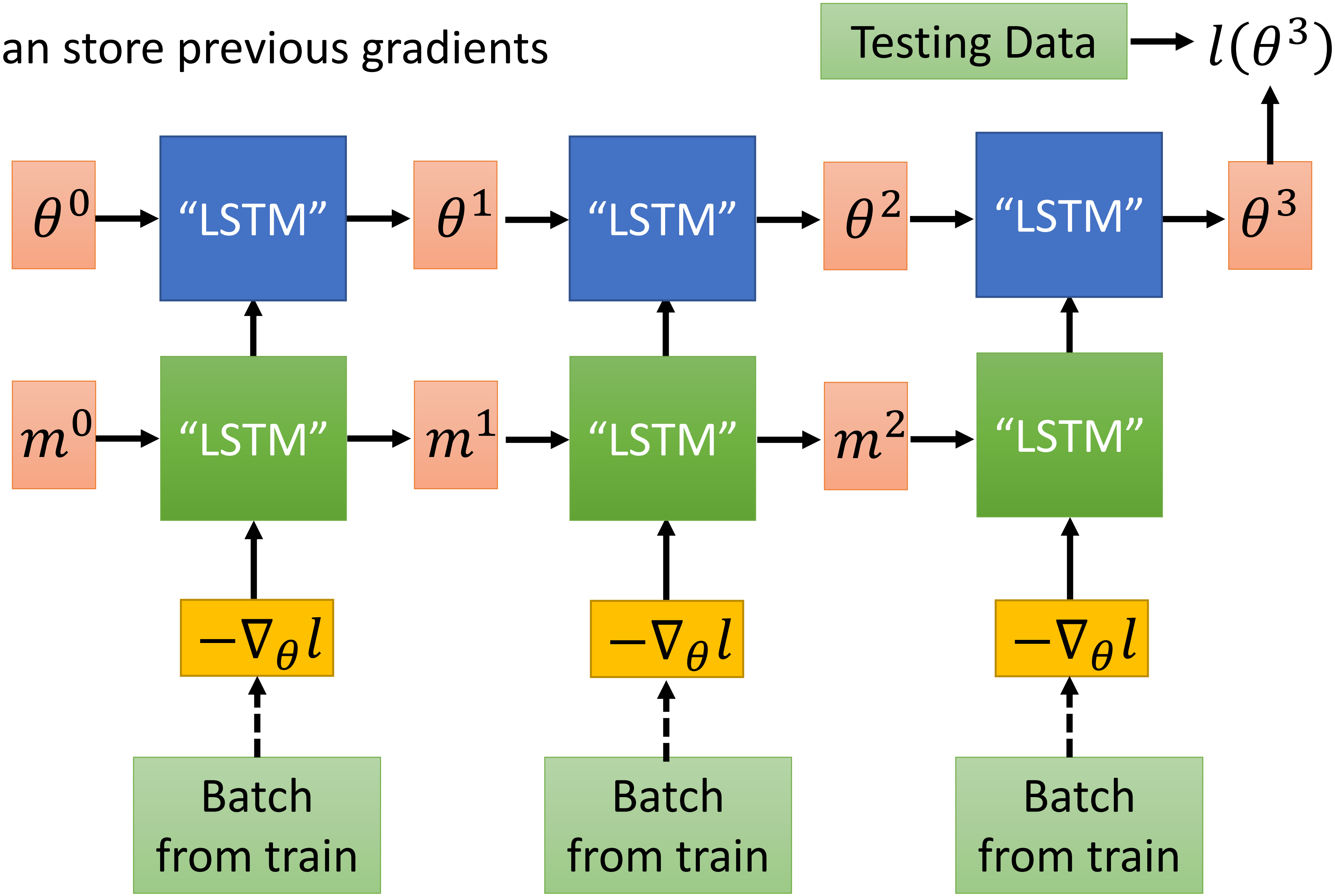
Parameter update depends on not only current gradient, but ***previous gradients***.



LSTM for Gradient Descent (v2)

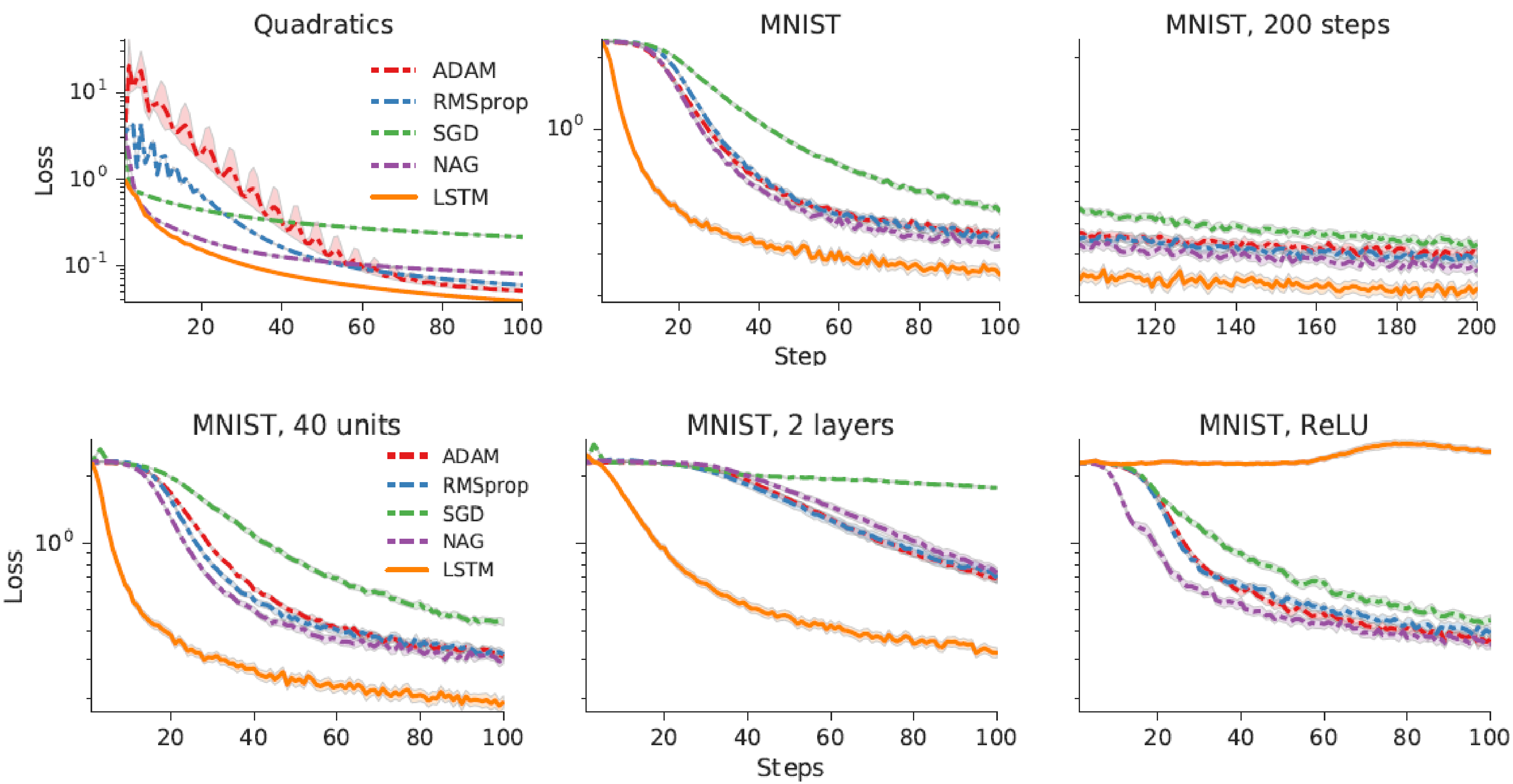
3 training steps

m can store previous gradients



https://arxiv.org/abs/1606.04474

Experimental Results

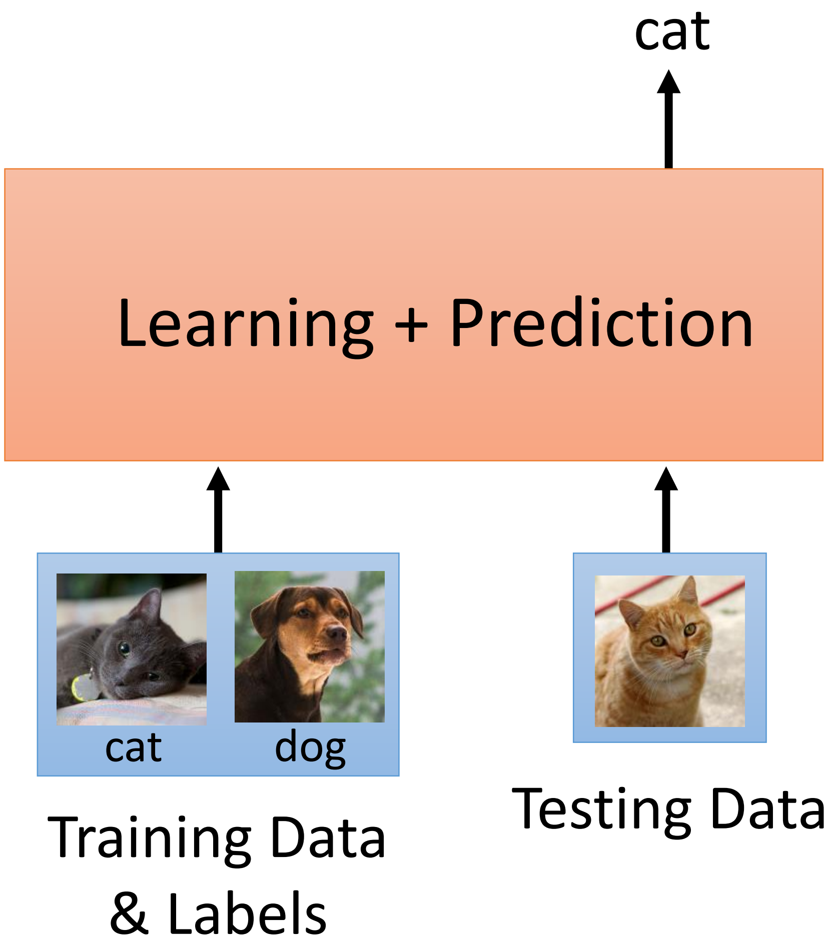


Meta Learning (Part 3)

Hung-yi Lee

Even more crazy idea …

- Input:
  - Training data and their labels
  - Testing data
- Output:
  - Predicted label of testing data



Face Verification

In each task:

Training

Few-shot Learning

Registration (Collecting Training data)

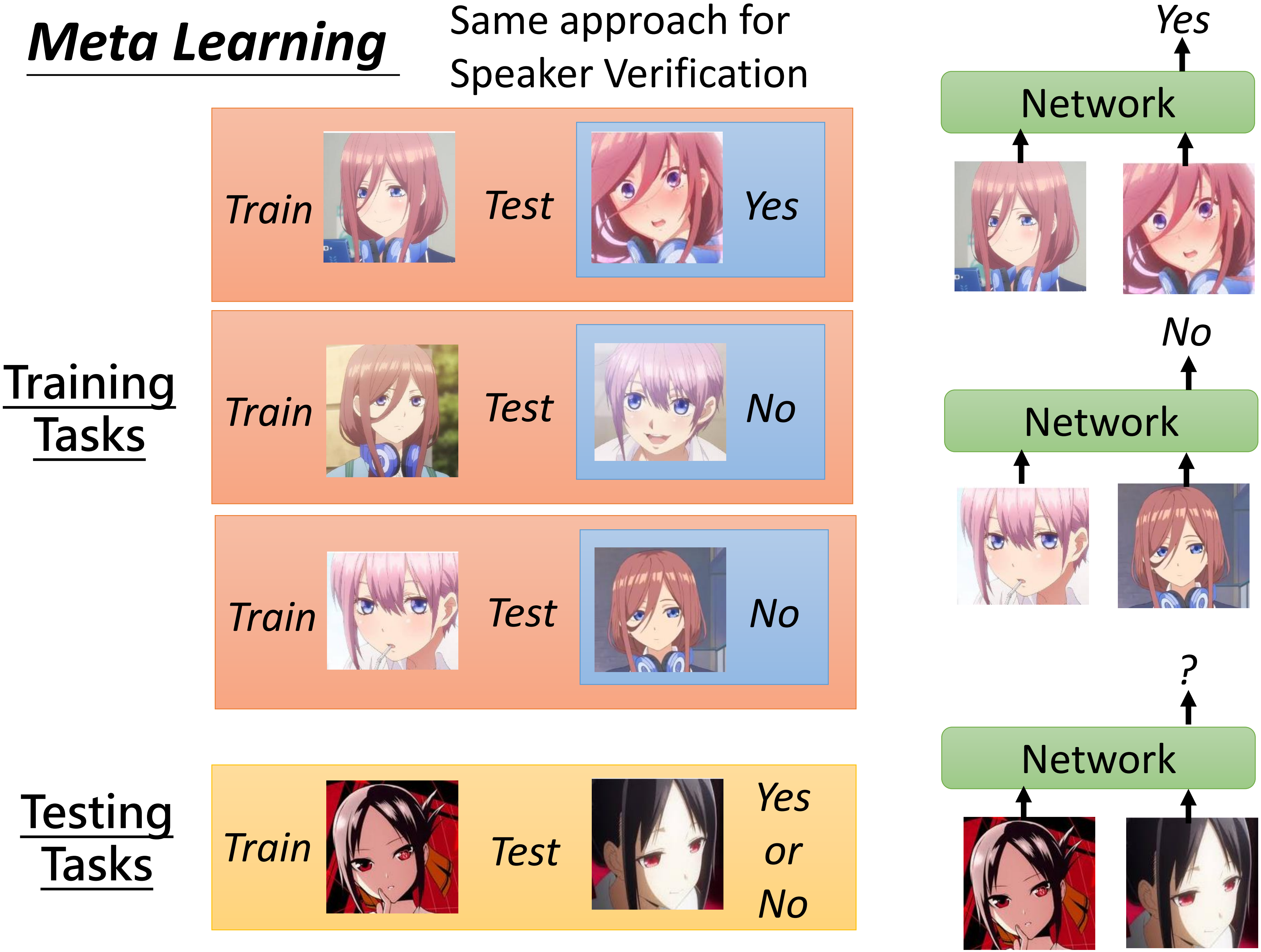
Testing

Unlock your phone by Face

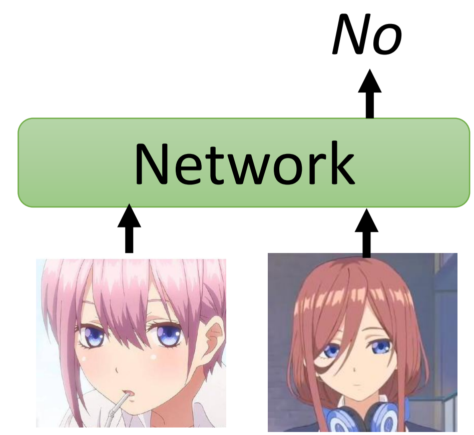
https://support.apple.com/zh-tw/HT208109

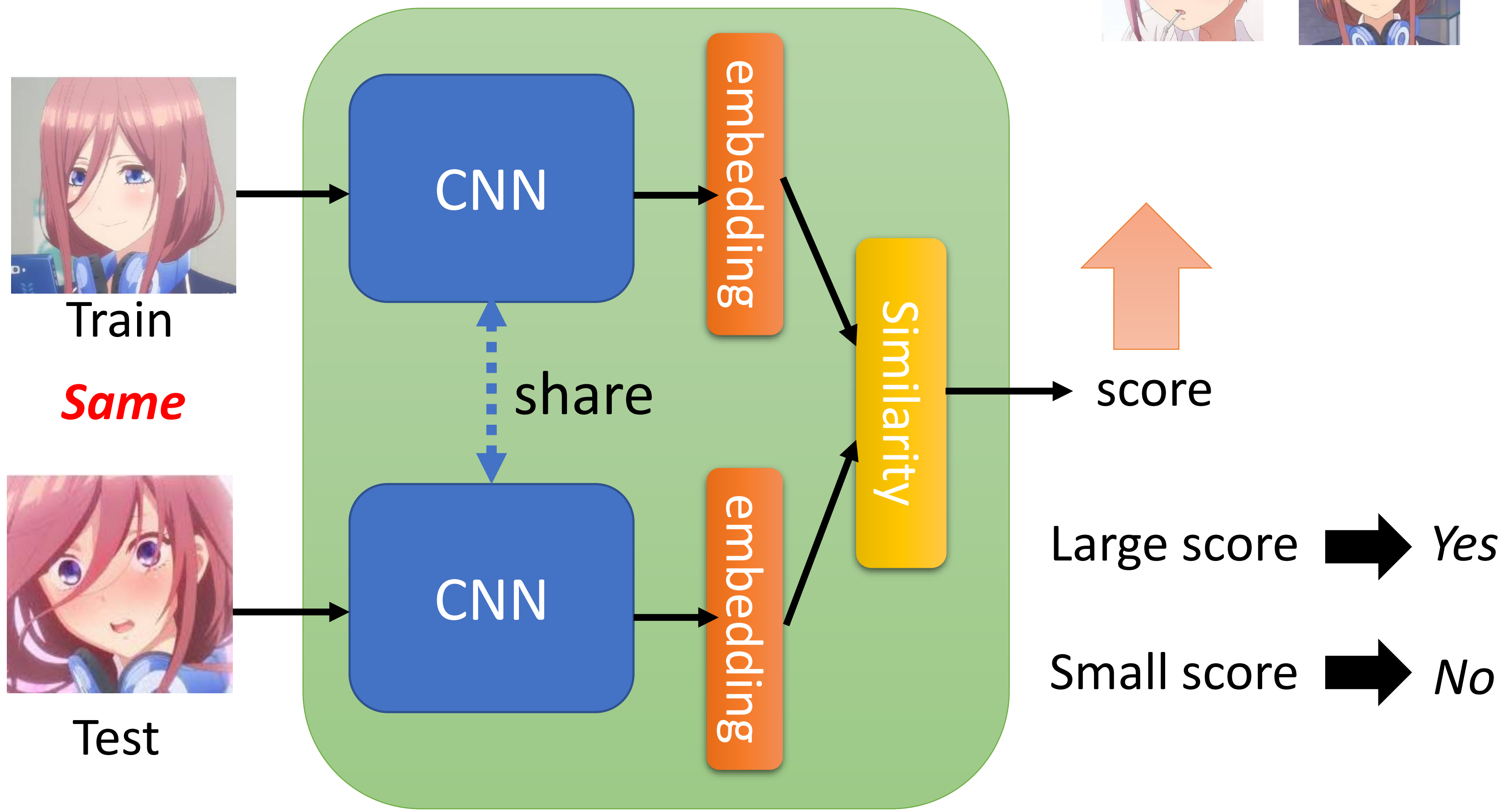
Meta Learning

Same approach for Speaker Verification

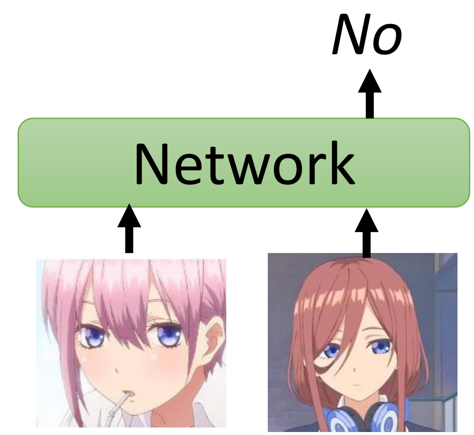


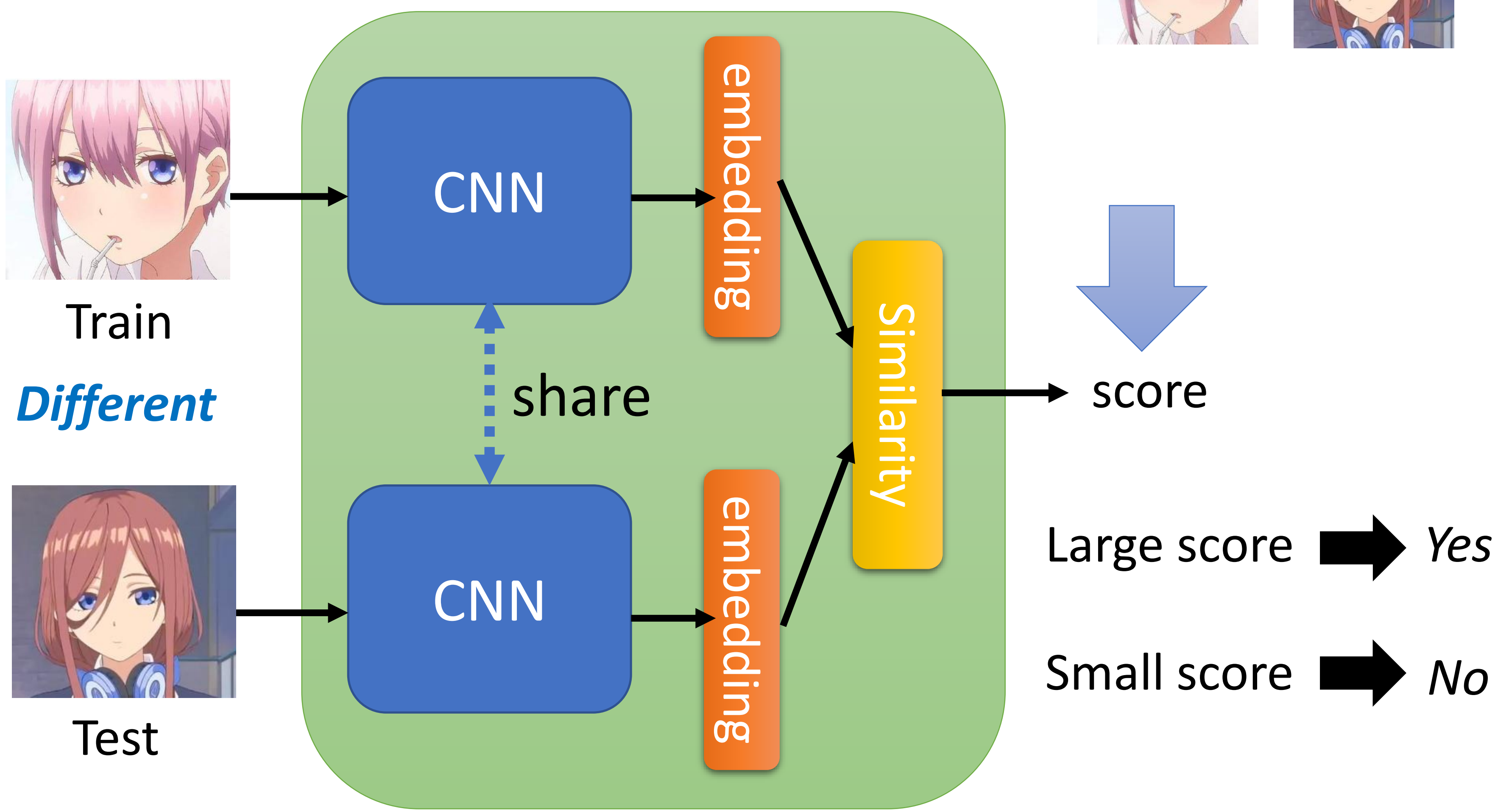
Siamese Network





Siamese Network



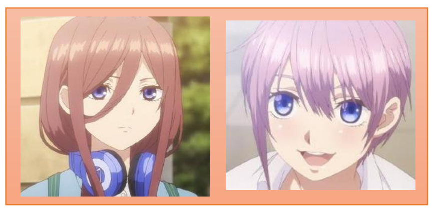


Siamese Network
- Intuitive Explanation

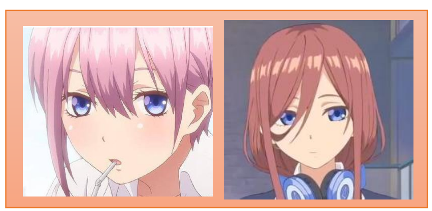
- Binary classification problem: “Are they the same?”

Training Set

same

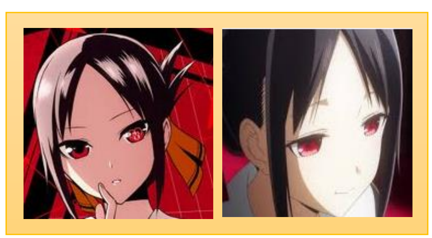


different

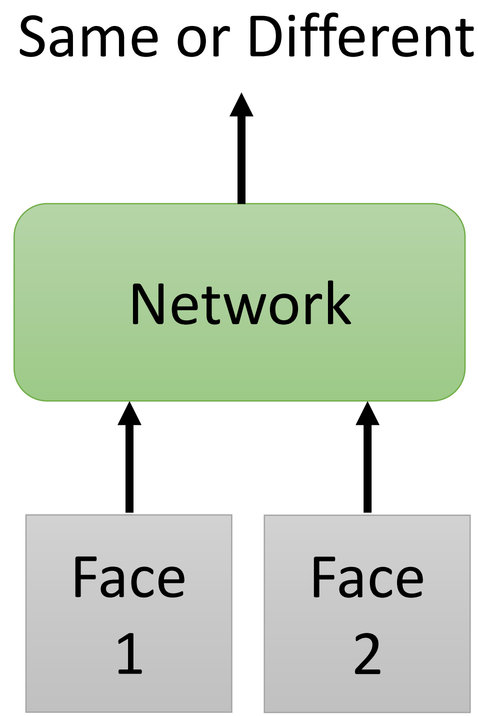


different

Testing Set



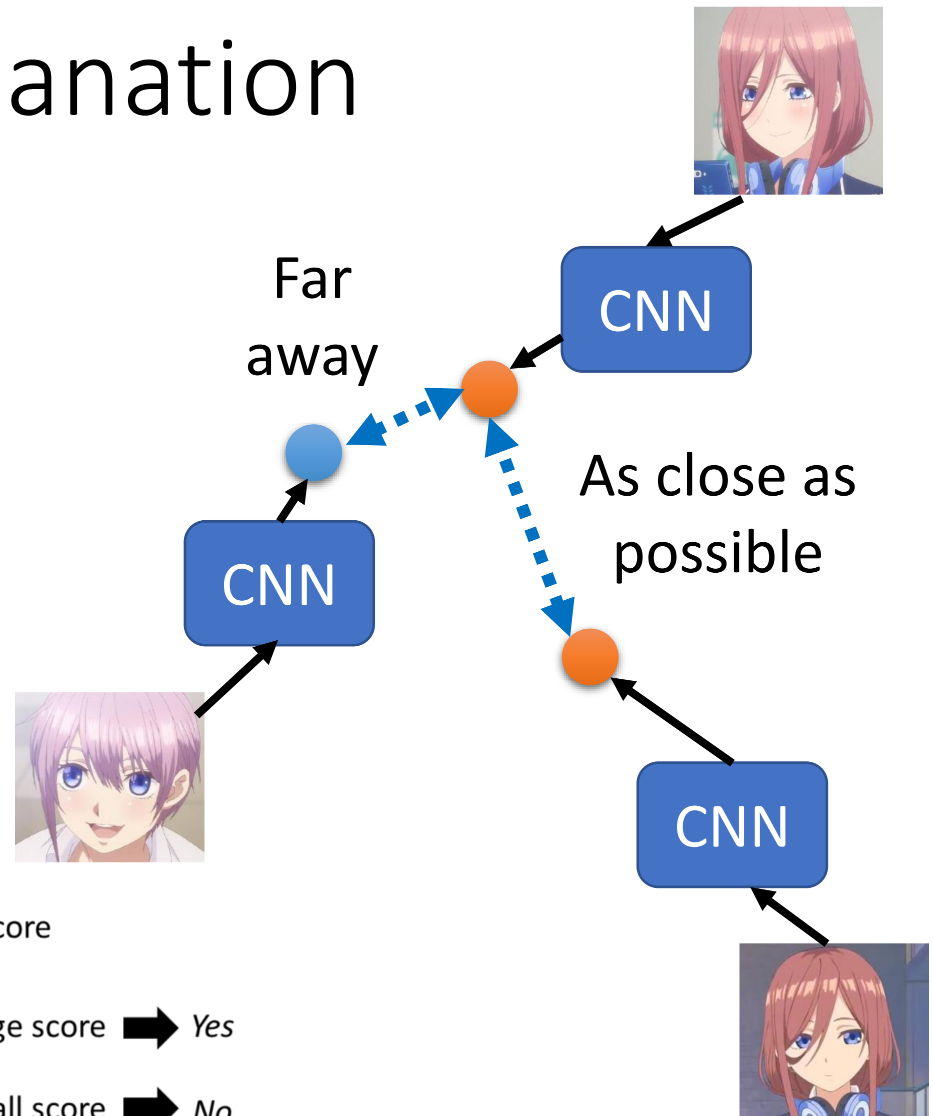
?

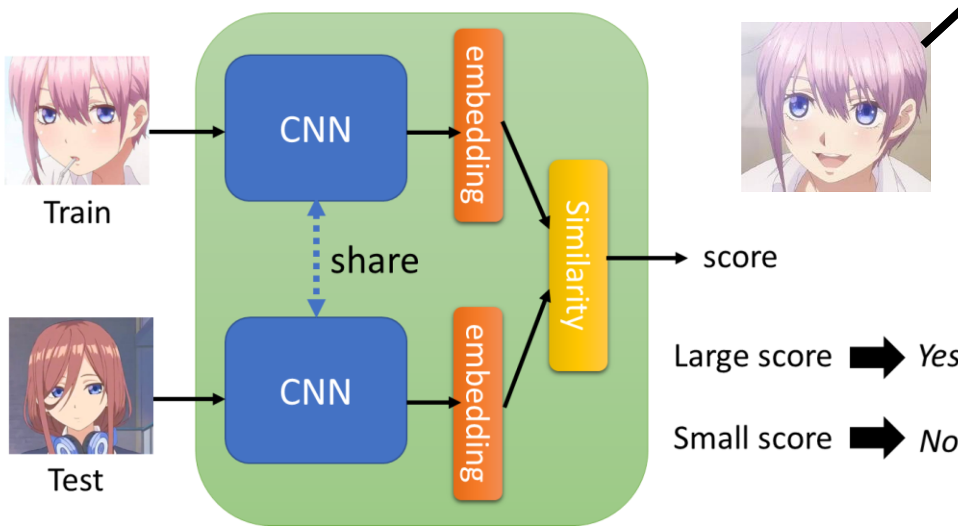


Siamese Network
- Intuitive Explanation

Learning embedding for faces

e.g. learn to ignore the background



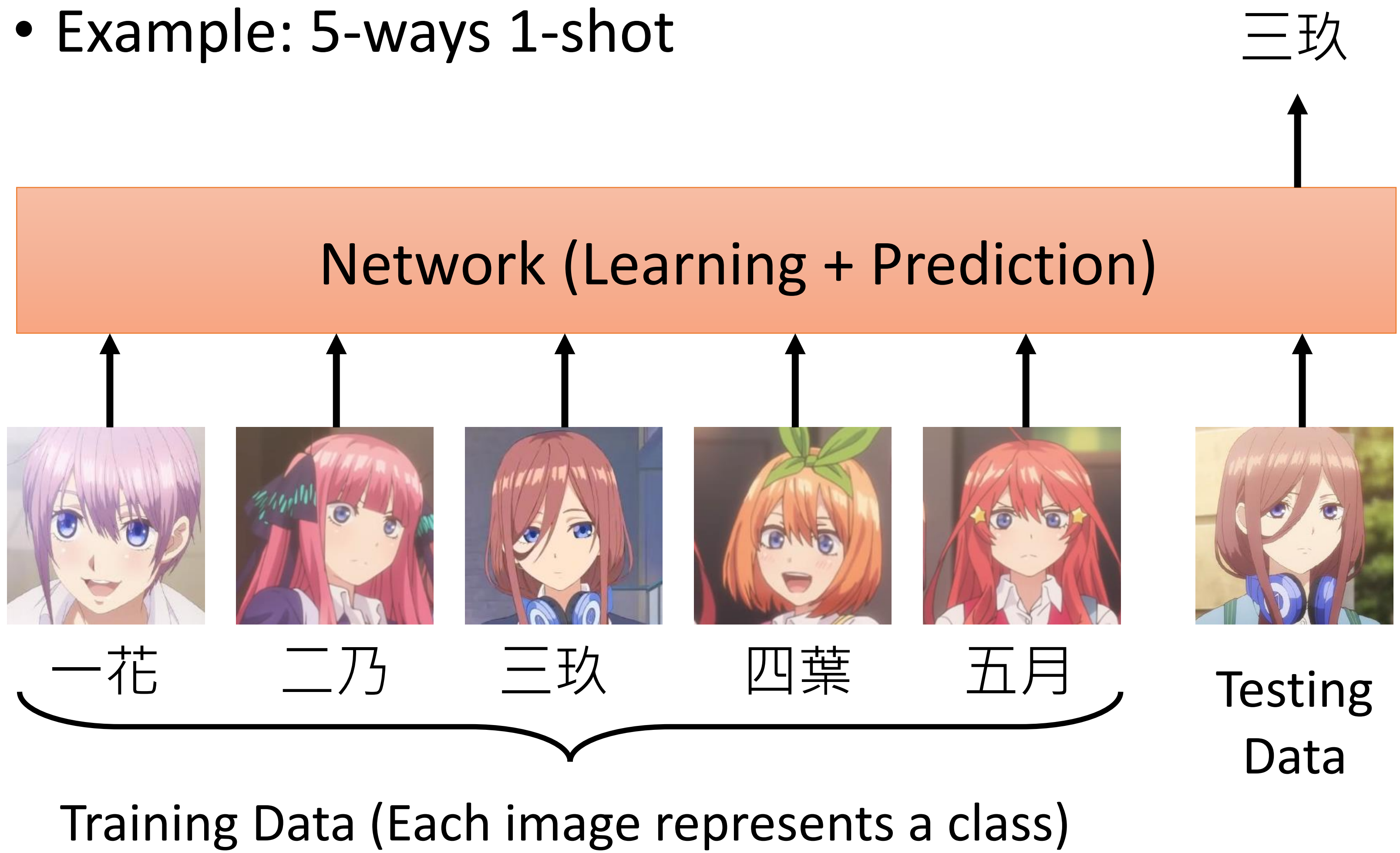


To learn more …

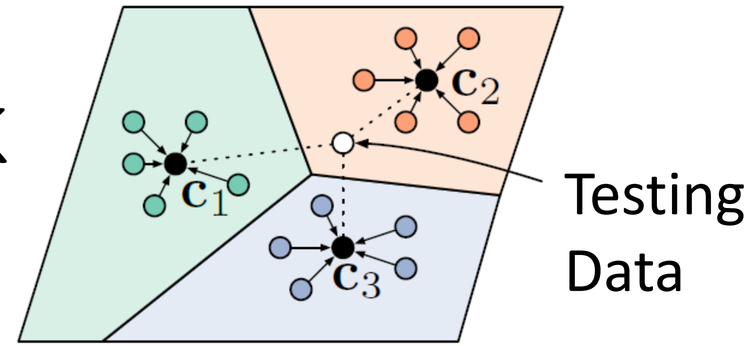
- What kind of distance should we use?
  - SphereFace: Deep Hypersphere Embedding for Face Recognition
  - Additive Margin Softmax for Face Verification
  - ArcFace: Additive Angular Margin Loss for Deep Face Recognition
- Triplet loss
  - Deep Metric Learning using Triplet Network
  - FaceNet: A Unified Embedding for Face Recognition and Clustering

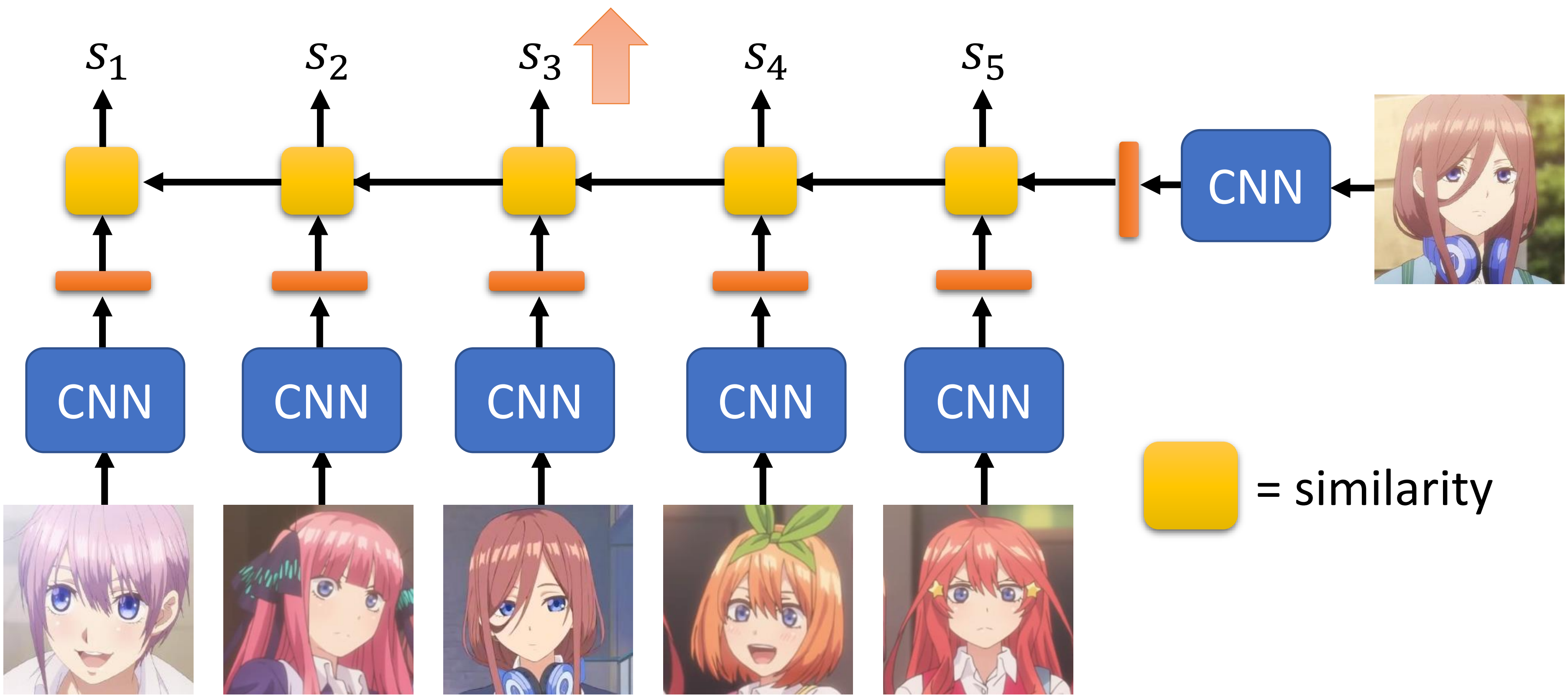
N-way Few/One-shot Learning

- Example: 5-ways 1-shot



Prototypical Network

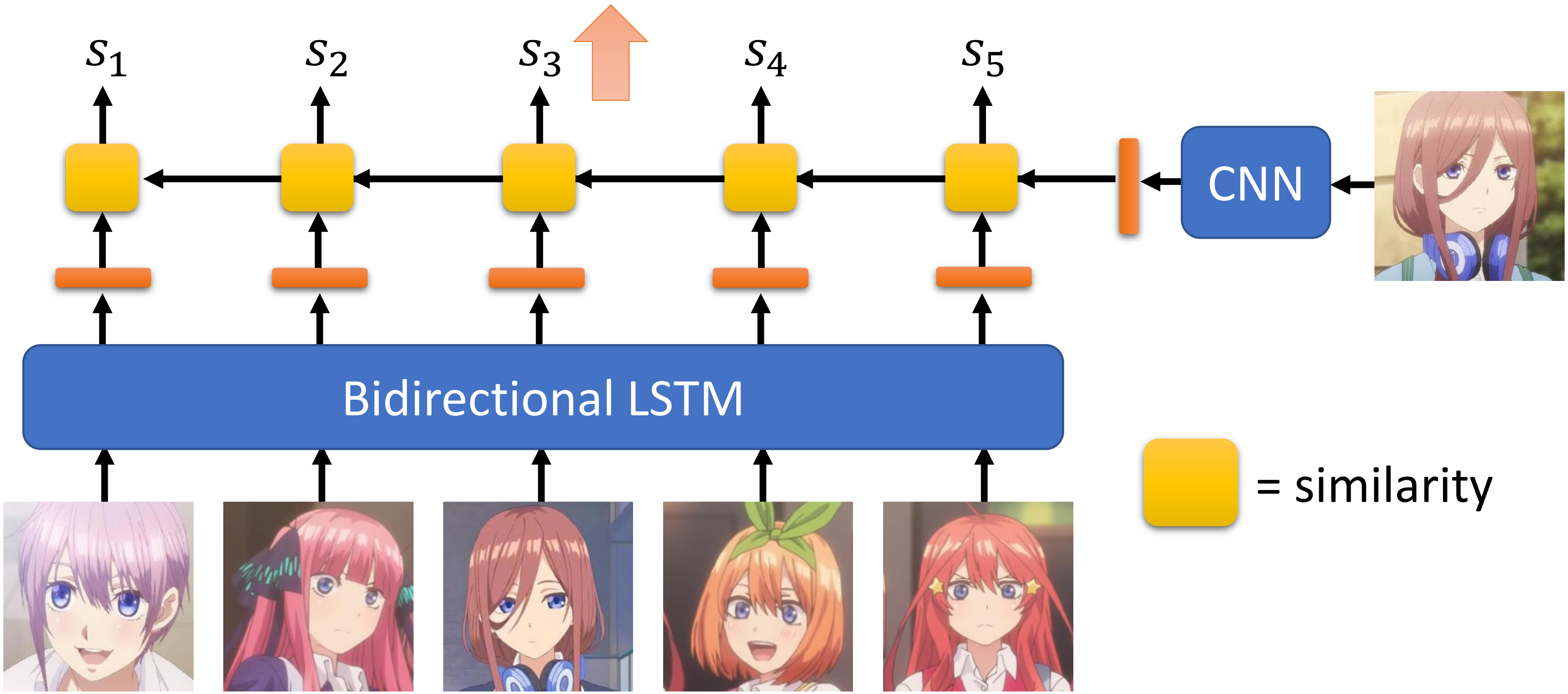




https://arxiv.org/abs/1703.05175

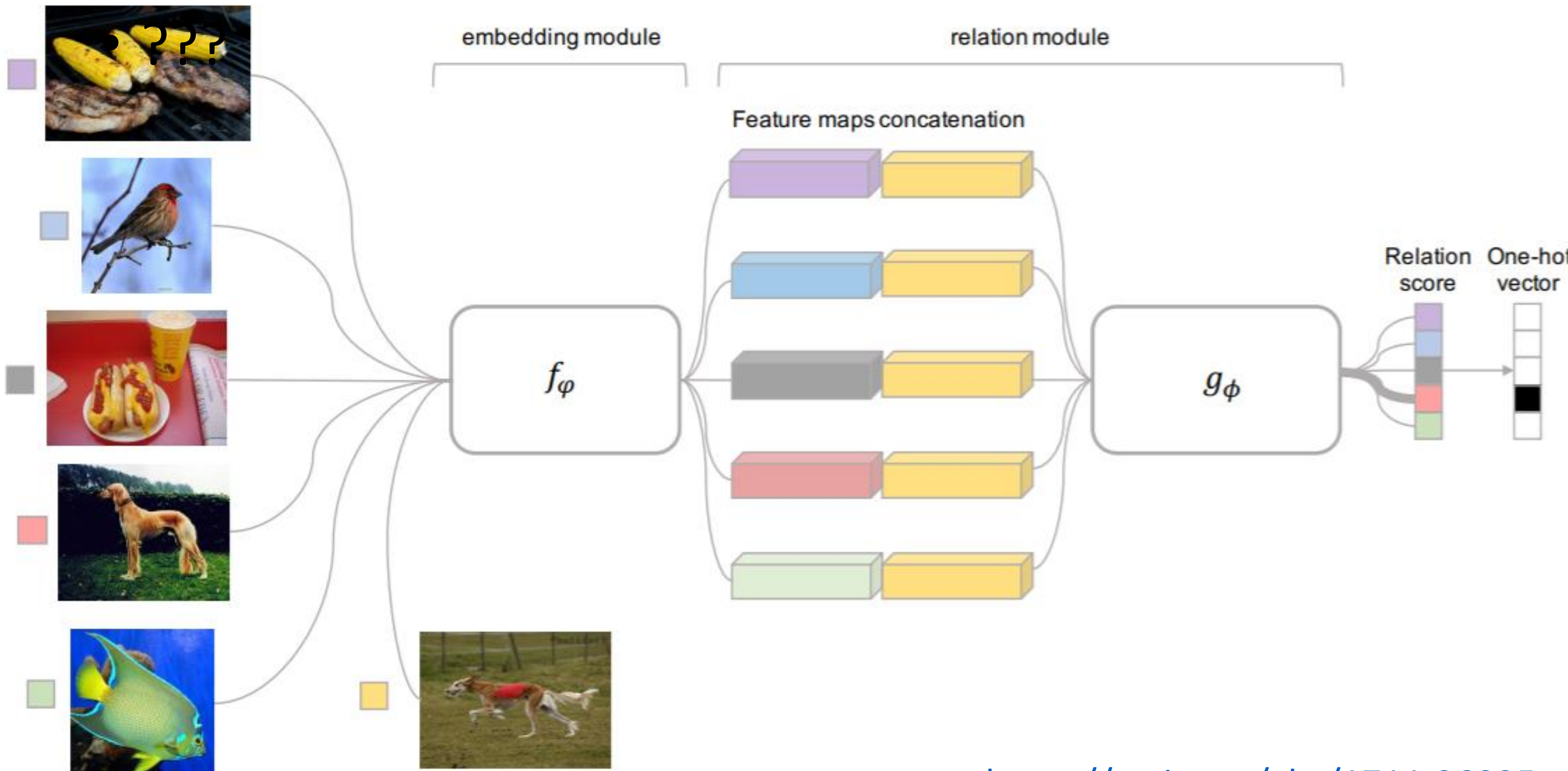
Matching Network

Considering the relationship among the training examples



https://arxiv.org/abs/1606.04080

Relation Network

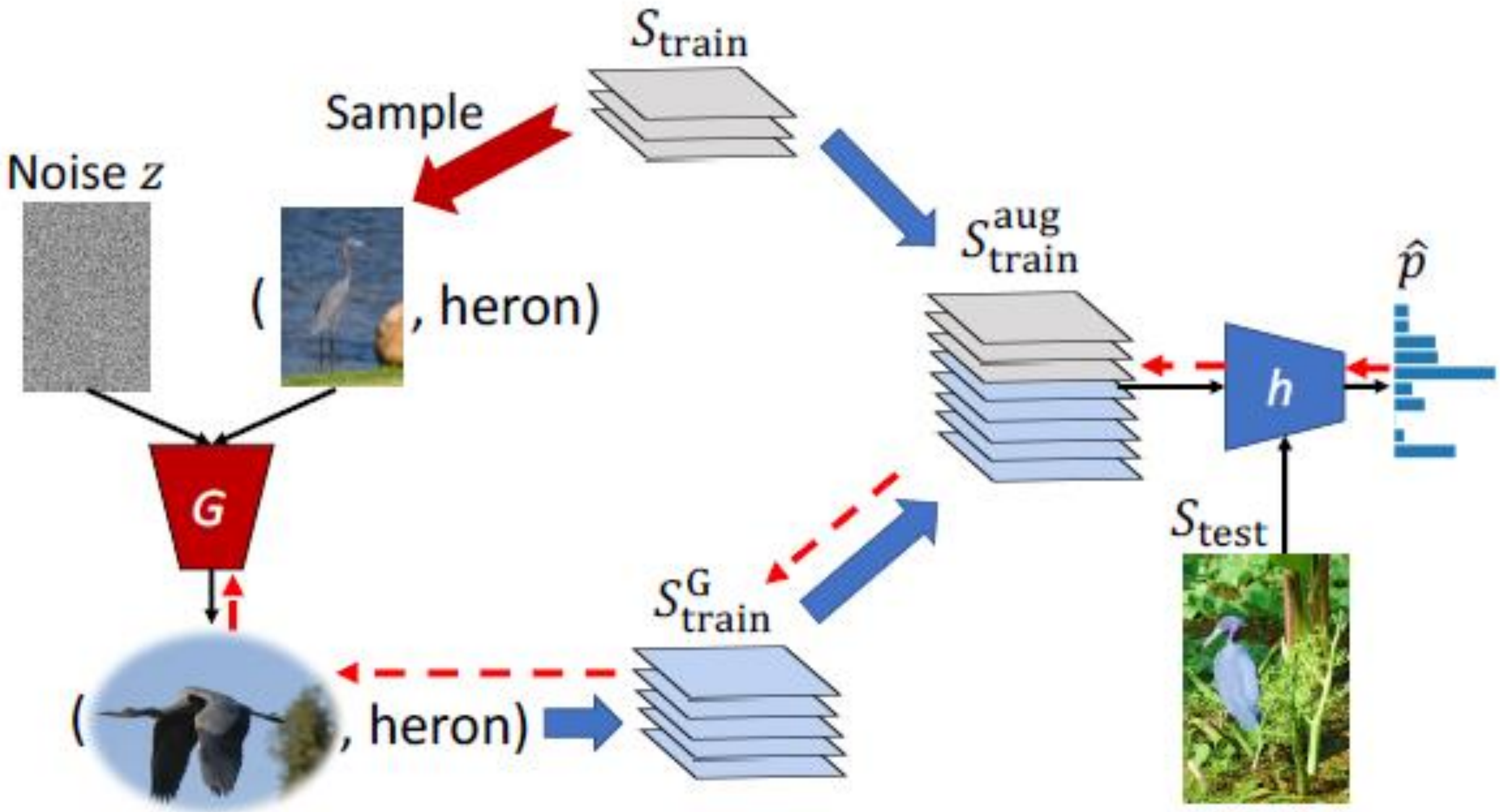


https://arxiv.org/abs/1711.06025

Few-shot learning for imaginary data

https://arxiv.org/abs/1801.05401

Few-shot learning for imaginary data



https://arxiv.org/abs/1801.05401