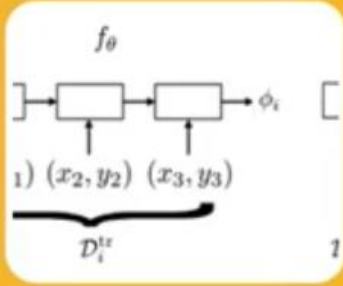


Meta-learning

王慧琪

Categories

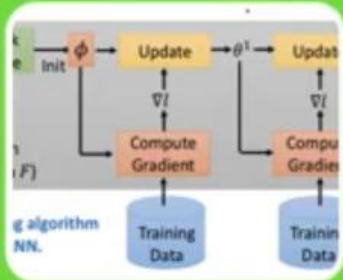
(Not unified, but generally...)



Black-box ((memory) Model) based

MANN
SNAIL
MetaNet...

Meta-LSTM

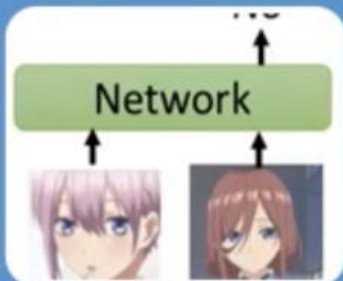


Optimization (**Gradient**) based
(Parametric)

• Learn to **initialize**

MAML
Reptile...

iMAML
MAML++...

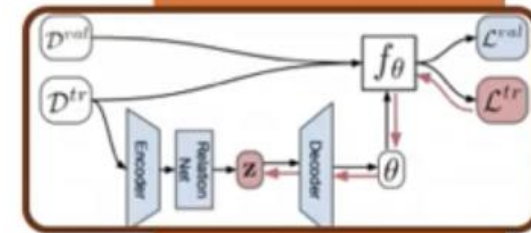


Non-parametric (**Metric** based)

• Learn to **compare**

Siamese Network
Matching Network
Prototypical Network
Relation Network...
IMP...

Hybrid



LEO...

Bayesian

PLATIPUS...

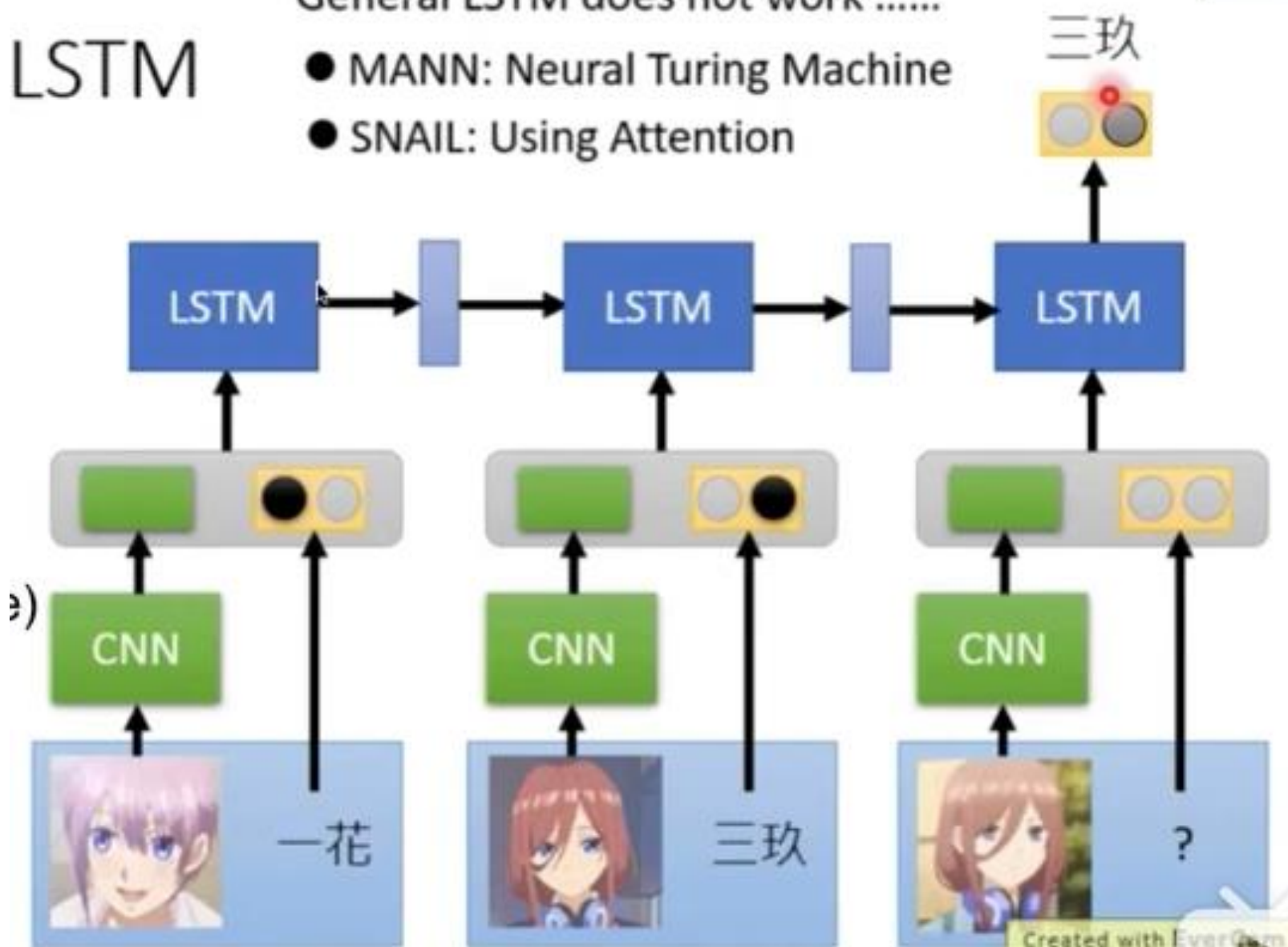
Model based

1. 加上 memory
(Neural Turing Machine)
(LST“M”)
2. 加上 attention
(Transformer)

LSTM

General LSTM does not work

- MANN: Neural Turing Machine
- SNAIL: Using Attention



MANN

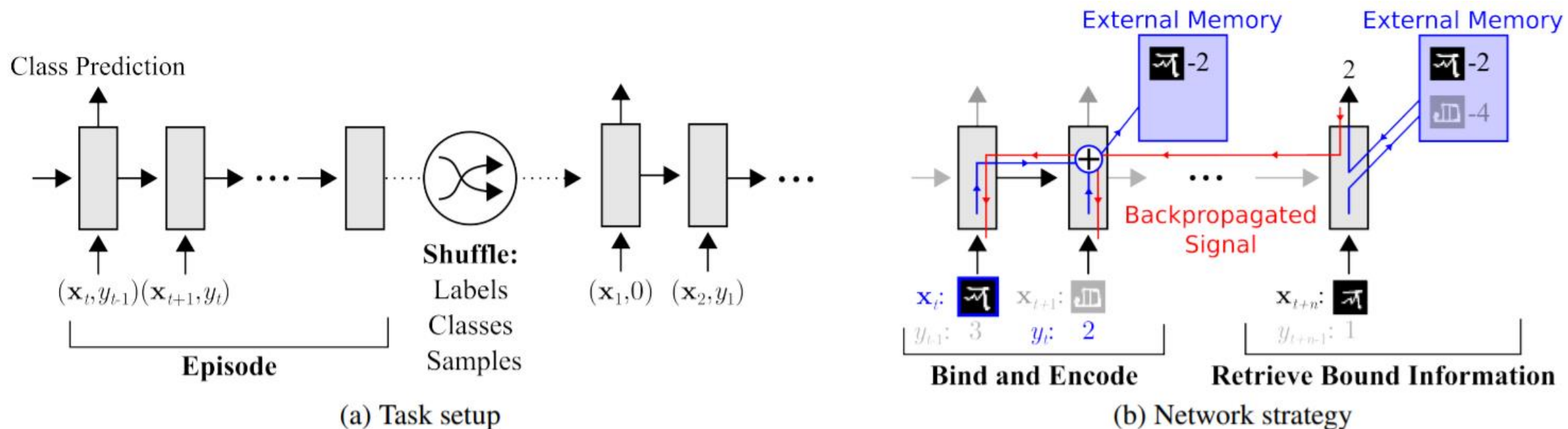


Figure 1. Task structure. (a) Omniglot images (or x -values for regression), \mathbf{x}_t , are presented with time-offset labels (or function values), y_{t-1} , to prevent the network from simply mapping the class labels to the output. From episode to episode, the classes to be presented in the episode, their associated labels, and the specific samples are all shuffled. (b) A successful strategy would involve the use of an external memory to store bound sample representation-class label information, which can then be retrieved at a later point for successful classification when a sample from an already-seen class is presented. Specifically, sample data \mathbf{x}_t from a particular time step should be bound to the appropriate class label y_t , which is presented in the subsequent time step. Later, when a sample from this same class is seen, it should retrieve this bound information from the external memory to make a prediction. Backpropagated error signals from this prediction step will then shape the weight updates from the earlier steps in order to promote this binding strategy.

Gradient based

MAML

Loss Function:

$$L(\phi) = \sum_{n=1}^N l^n(\hat{\theta}^n)$$

$\hat{\theta}^n$: model learned from task n

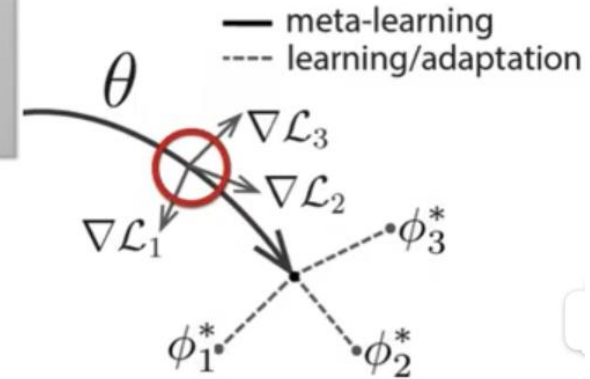
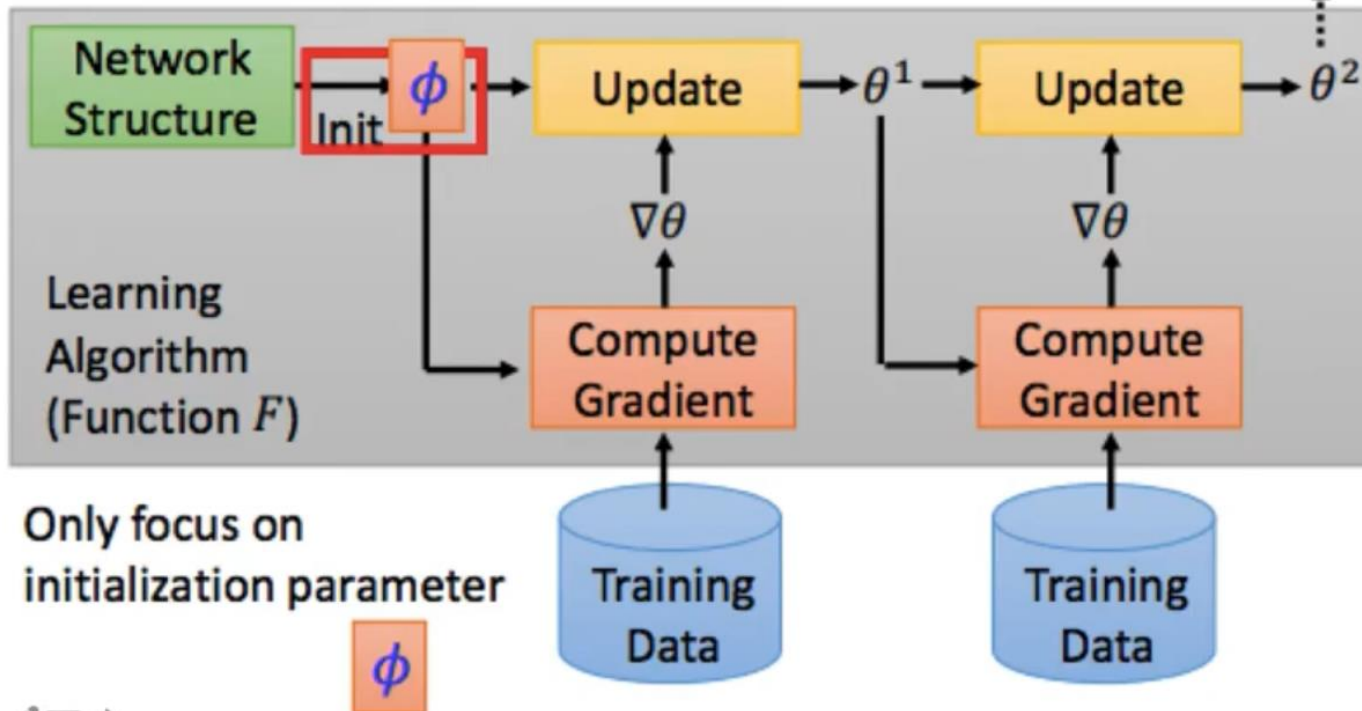
$\hat{\theta}^n$ depends on ϕ

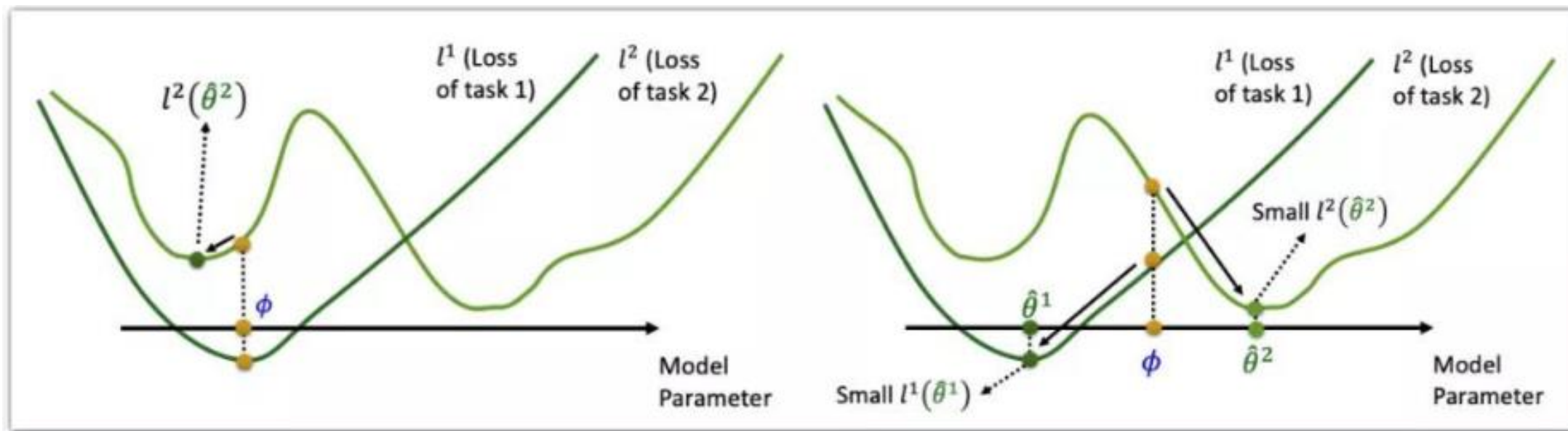
$l^n(\hat{\theta}^n)$: loss of task n on the testing set of task n

$$\phi \leftarrow \phi - \eta \nabla_{\phi} L(\phi)$$

Considering one-step training:

$$\hat{\theta} = \phi - \varepsilon \nabla_{\phi} l(\phi)$$

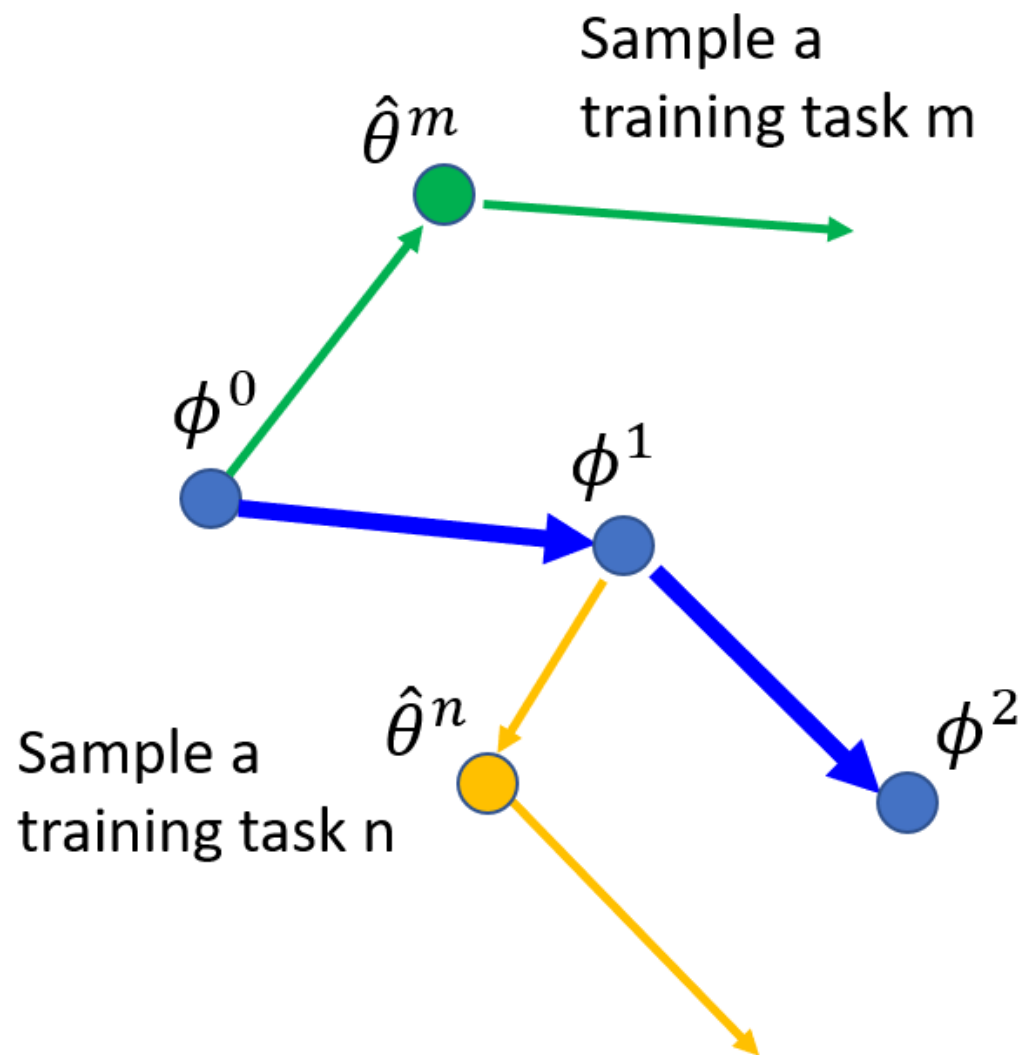




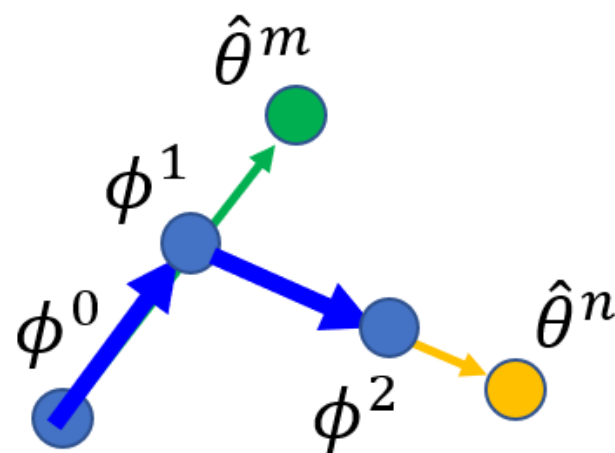
$$L(\phi) = \sum_{n=1}^N l^n(\phi)$$

$$L(\phi) = \sum_{n=1}^N l^n(\hat{\theta}^n)$$

MAML – Real Implementation



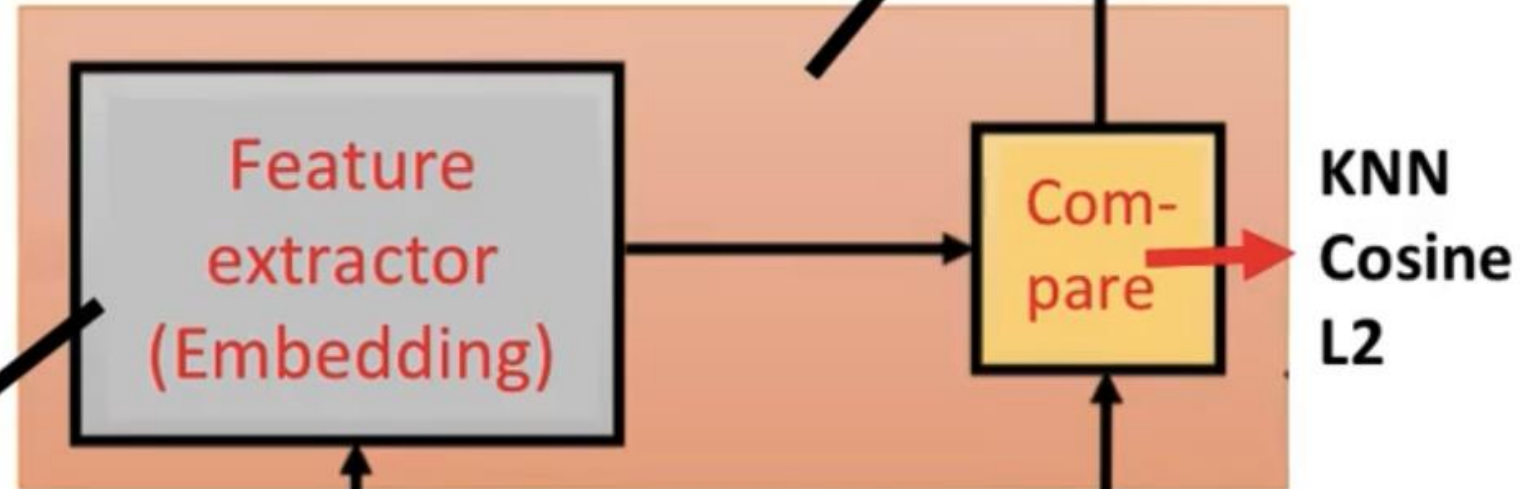
Model Pre-training



Metric-based / non-parametric

Learn to **compare!**

Network
as an algorithm



A bigger function



Training Data

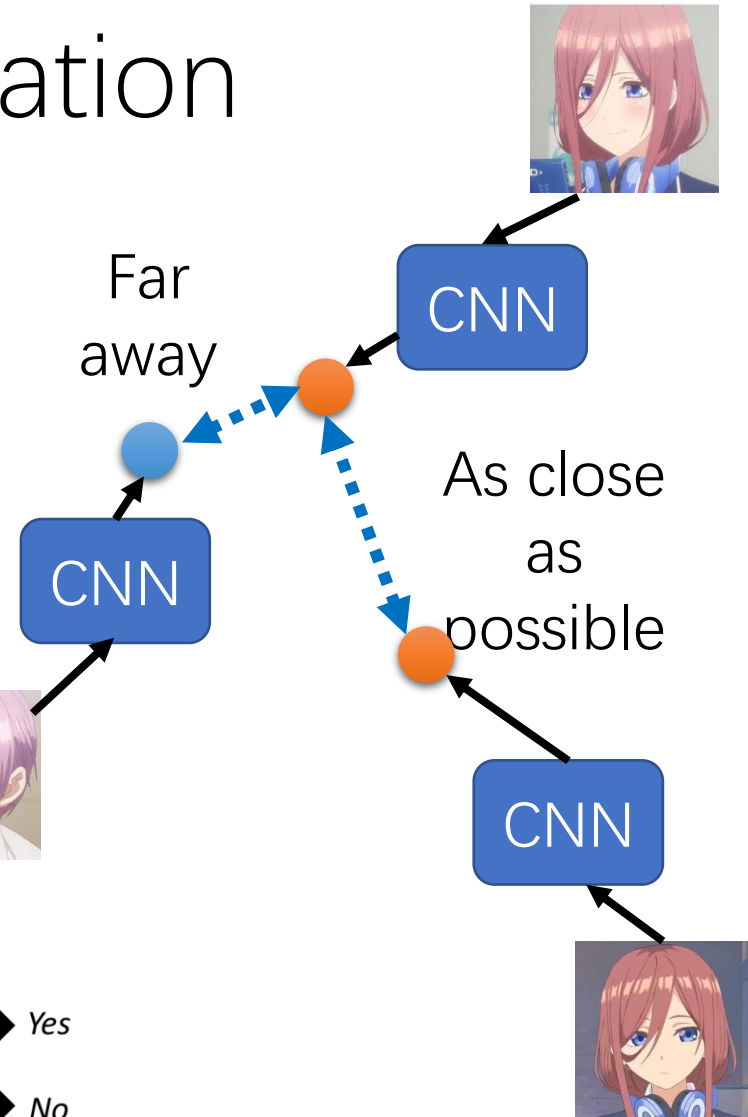
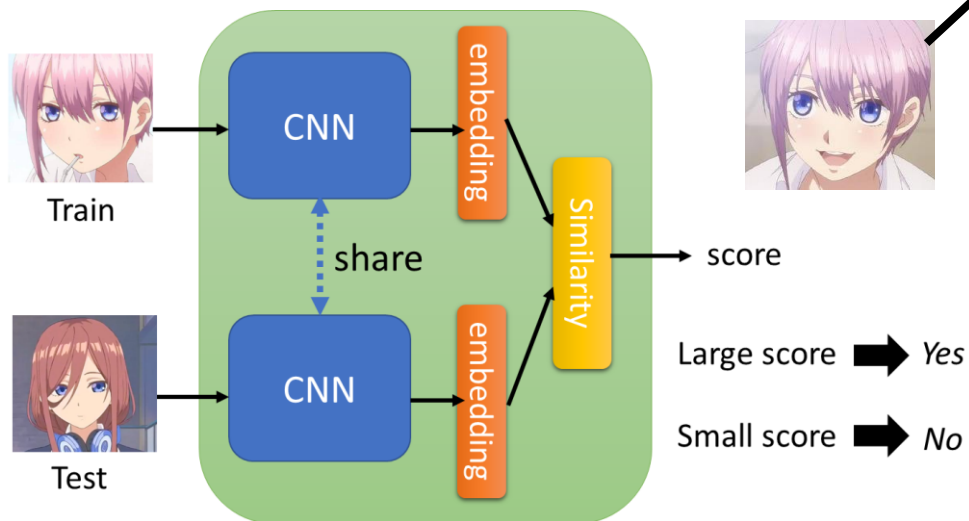


Testing Data

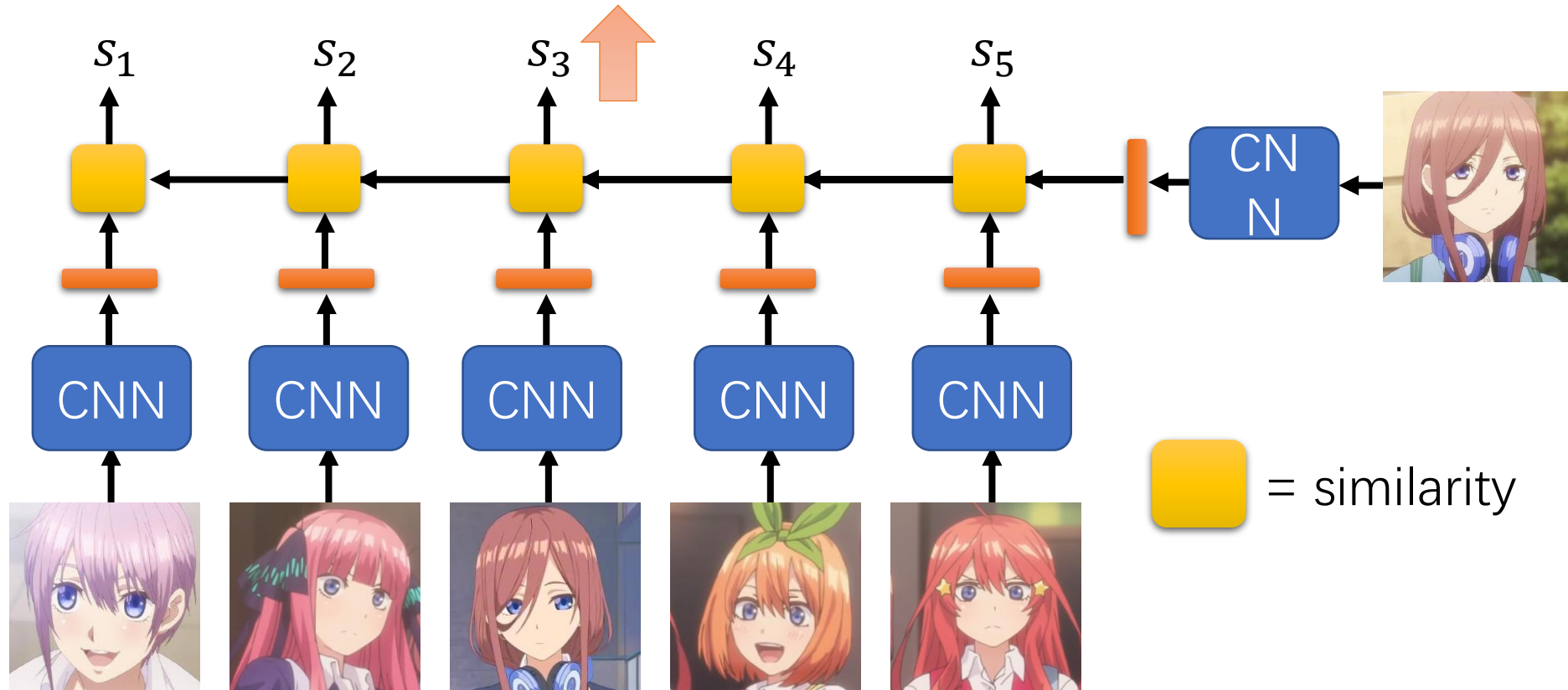
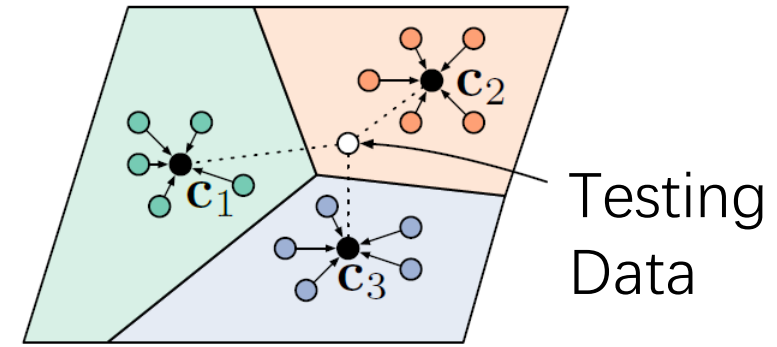
Siamese Network

- Intuitive Explanation

Learning
embedding for
faces
e.g. learn to ignore the
background

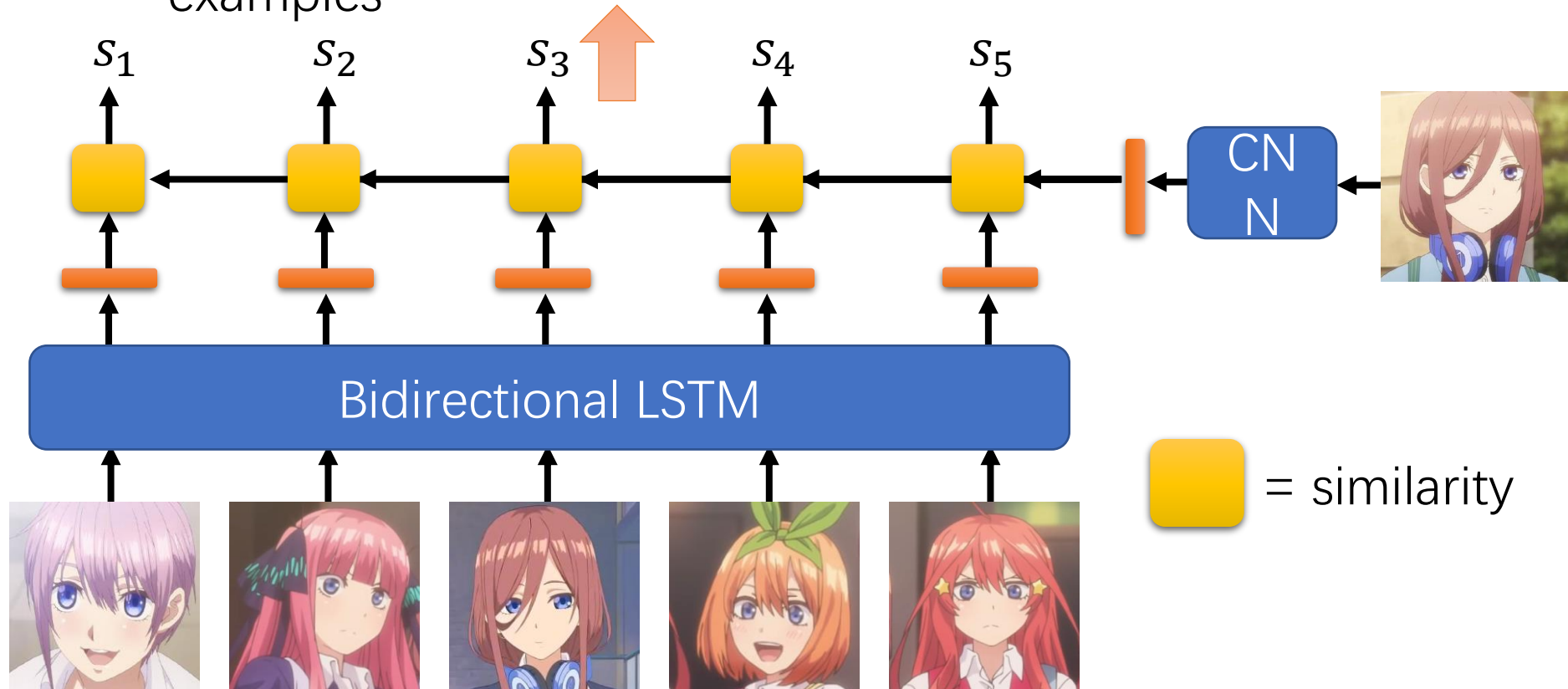


Prototypical Network



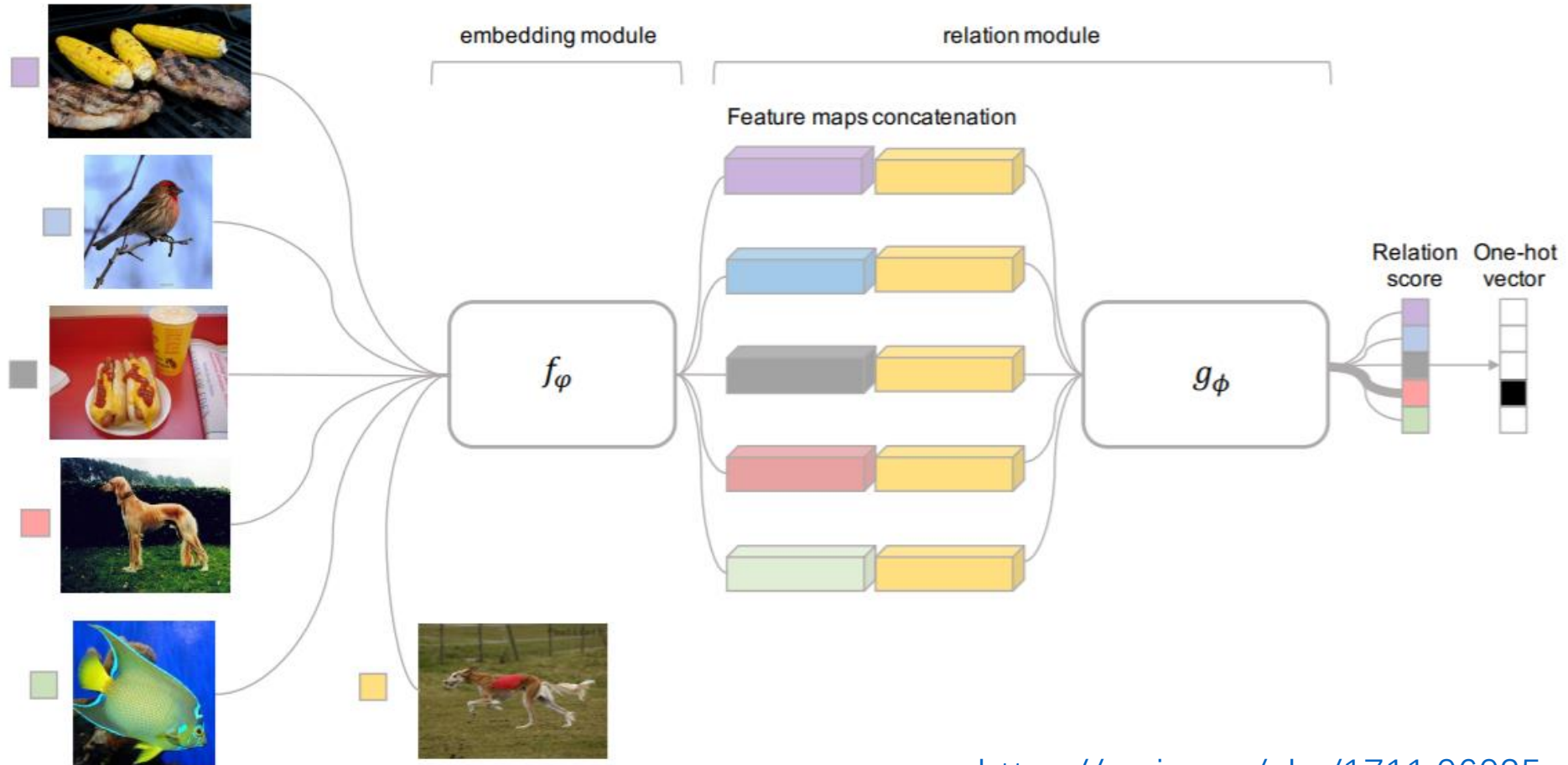
Matching Network

Considering the relationship among the training examples



Relation Network

- ???



<https://arxiv.org/abs/1711.06025>

2018ACL-short

Natural Language to Structured Query Generation via Meta-Learning

Po-Sen Huang^{*}, Chenglong Wang[†], Rishabh Singh^{*}, Wen-tau Yih[‡], Xiaodong He^{*◇}

^{*}Microsoft Research, [†]University of Washington,

[‡]Allen Institute for Artificial Intelligence, [◇]JD AI Research

`pshuang@microsoft.com, clwang@cs.washington.edu,`

`rishabh.iit@gmail.com, scottyih@allenai.org, xiaodong.he@jd.com`

When empirically evaluated on a recently proposed, large semantic parsing dataset, WikiSQL (Zhong et al., 2017), our approach leads to faster convergence and achieves 1.1%–5.4% absolute accuracy gain over the non-meta-learning counterparts, establishing a new state-of-the-art result. More importantly, we demonstrate how to design a relevance function to successfully reduce a regular supervised learning problem to a meta-learning problem. To the best of our knowledge, this is the first successful attempt in adapting meta-learning to a semantic task.

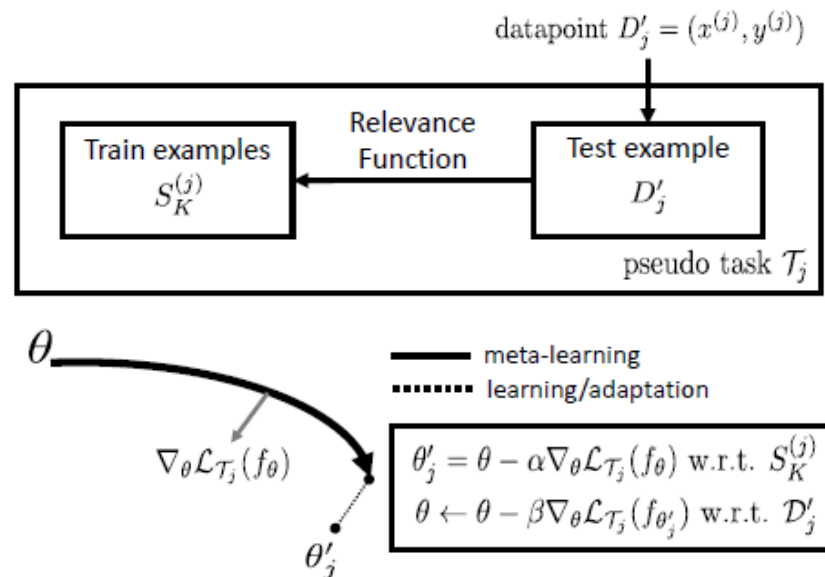


Figure 1: Diagram of the proposed framework. (Upper) we propose using a relevant function to find a support set $S_K^{(j)}$ from all training datapoints given a datapoint D'_j for constructing a pseudo-task \mathcal{T}_j as in the few-shot meta-learning setup. (Bottom) We optimize the model parameters θ such that the model can learn to adapt a new task with parameters θ'_j via a few gradient steps on the training examples of the new task. The model is updated by considering the test error on the test example of the new task. See Section 2 for detail.

Relevance Function

There are five SQL types in the WikiSQL dataset: `{Count, Min, Max, Sum, Avg, Select}`. We train a SQL type classifier f_{sql} using SVMs with bag-of-words features of the input question, which achieves 93.5% training accuracy and 88% test accuracy in SQL type prediction. Another soft indication on whether two questions can be viewed as belonging to the same “task” is their lengths, as they correlate to the lengths of the mapped SQL queries. The length of a question is the number of tokens in it after normal-

izing entity mentions to single tokens.¹ Our relevance function only considers examples of the same predicted SQL types. If examples $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ have the same SQL type, then their relevance score is $1 - |qlen(\mathbf{x}^{(i)}) - qlen(\mathbf{x}^{(j)})|$, where $qlen$ calculates the question length. Notice that the relevance function does not need to be highly accurate as there is no formal definition on which examples should be grouped in the same pseudo-task. A heuristic-based function that encodes some domain knowledge typically works well based on our preliminary study. In principle, the relevance function can also be jointly learned with the meta-learning model, which we leave for future work.

Algorithm 1 Pseudo-Task MAML (PT-MAML)

Require: Training Datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$

Require: α, β : step size hyperparameters

Require: K : support set size hyperparameter

- 1: Construct a task \mathcal{T}_j with training examples using a support set $\mathcal{S}_K^{(j)}$ and a test example $\mathcal{D}'_j = (\mathbf{x}^{(j)}, \mathbf{y}^{(j)})$.
 - 2: Denote $p(\mathcal{T})$ as distribution over tasks
 - 3: Randomly initialize θ
 - 4: **while** not done **do**
 - 5: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
 - 6: **for all** \mathcal{T}_i **do**
 - 7: Evaluate $\nabla_{\theta} \mathcal{L}_{\mathcal{T}_i}(f_{\theta})$ using $\mathcal{S}_K^{(j)}$
 - 8: Compute adapted parameters with gradient descent: $\theta'_i = \theta - \alpha \nabla_{\theta} \mathcal{L}_{\mathcal{T}_i}(f_{\theta})$
 - 9: **end for**
 - 10: Update $\theta \leftarrow \theta - \beta \nabla_{\theta} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})$ using each \mathcal{D}'_i from \mathcal{T}_i and $\mathcal{L}_{\mathcal{T}_i}$ for the meta-update
 - 11: **end while**
-

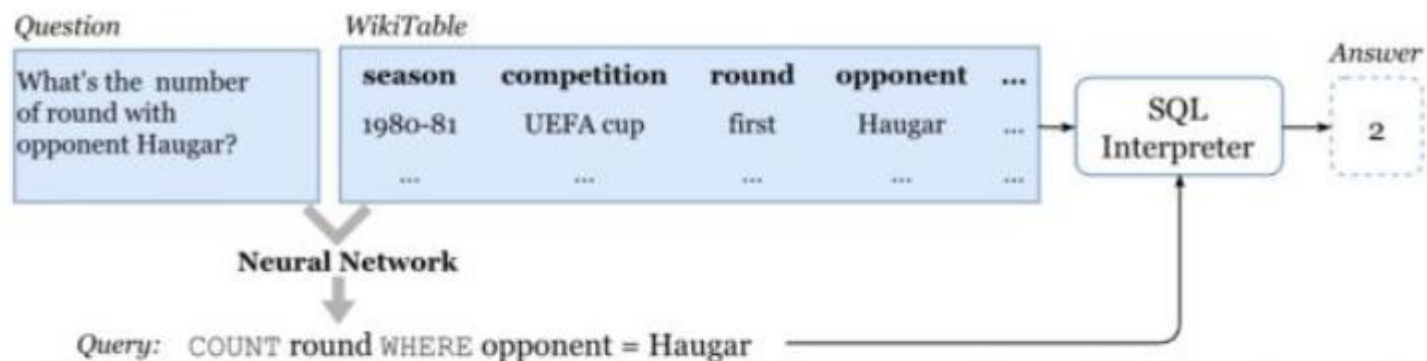


Figure 1: Answering a table question by synthesizing a query and executing it on the provided table

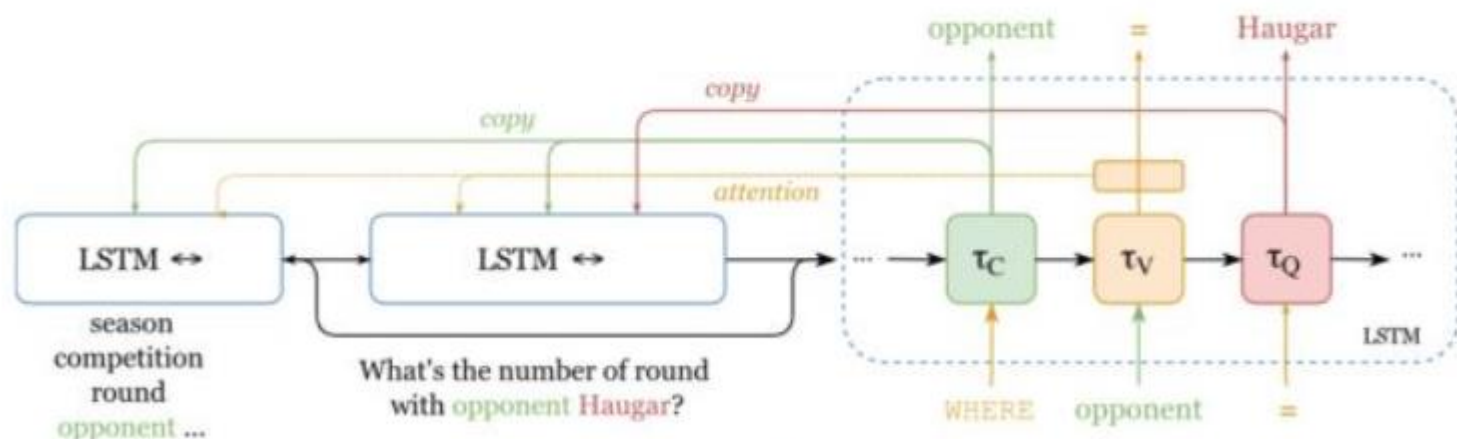


Figure 2: Model overview for the example in Figure 1. The model encodes table columns as well as the user question with a bidirectional LSTM and then decodes the hidden state with a typed LSTM, where the decoding action for each cell is statically determined.

The three decoding types are defined as follows:

- τ_V (SQL operator): The output has to be a SQL operator, i.e., a terminal from $V = \{\text{Select, From, Where, Id, Max, Min, Count, Sum, Avg, And, =, >, \geq, <, \leq, <END>, <GO>}\}$.
- τ_C (column name): The output has to be a column name, which will be copied from either the table header or the query section of the input sequence. Note that the column required for the correct SQL output may or may not be mentioned explicitly in the question.
- τ_Q (constant value): The output is a constant that would be copied from the question section of the input sequence.

The grammar of SQL expressions in the the WikiSQL dataset can be described in regular expression as “Select f c From t Where (c op v)” (f refers to an aggregation function, c refers to a column name, t refers to the table name, op refers an comparator and v refers to a value). The form can be represented by a decoding-type sequence $\tau_V \tau_V \tau_C \tau_V \tau_C \tau_V (\tau_C \tau_V \tau_Q)^*$, which will ensure only decoding-type corrected tokens can be sampled at each decoding step.

2019ACL

**Coupling Retrieval and Meta-Learning for
Context-Dependent Semantic Parsing**

Daya Guo^{1*}, Duyu Tang², Nan Duan², Ming Zhou², and Jian Yin¹

¹ The School of Data and Computer Science, Sun Yat-sen University.

Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, P.R.China

² Microsoft Research Asia, Beijing, China

`{guody5@mail2, issjyin@mail}.sysu.edu.cn`

`{dutang, nanduan, mingzhou}@microsoft.com`

- 包含了检索和元学习两部分。在检索部分，首先采样一批测试数据 D' ，然后利用基于上下文的检索模型 R 找到相似的样例 S' 作为训练数据，从而构成一个任务。在训练阶段，首先使用训练数据得到特定任务的模型 $M_{(\theta')}$ (step 1)，然后再利用测试数据更新元学习器 M_{θ} (step 2)。在预测阶段，先使用相似样本更新元学习器的参数，然后再进行预测。

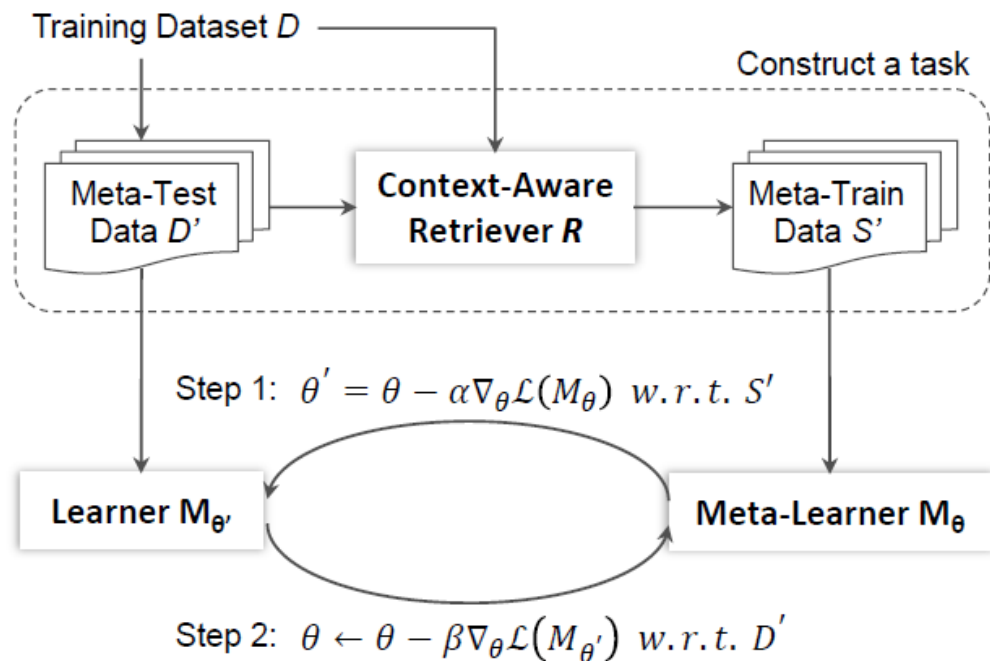


Figure 2: An overview of our approach that couples context-aware retriever and meta-learning.

Algorithm 1 Retrieval-MAML

Input: Training dataset $D = (x^{(j)}, c^{(j)}, y^{(j)})$, step size α and β

Output: Meta-learner M

- 1: Training a context-aware retriever R using D .
 - 2: For each example d , we obtain a support set S^d retrieved by R
 - 3: Randomly initialize θ for M
 - 4: **while** not done **do**
 - 5: Sample a batch of examples D' from D as test examples, and $S' = \bigcup_{d \in D'} S^d$ are viewed as training examples
 - 6: Evaluate $\nabla_{\theta} \mathcal{L}(M_{\theta})$ using S' , and compute adapted parameters with gradient descent: $\theta' = \theta - \alpha \nabla_{\theta} \mathcal{L}(M_{\theta})$
 - 7: Update $\theta \leftarrow \theta - \beta \nabla_{\theta} \mathcal{L}(M_{\theta'})$ using D' for meta-update
 - 8: **end while**
-

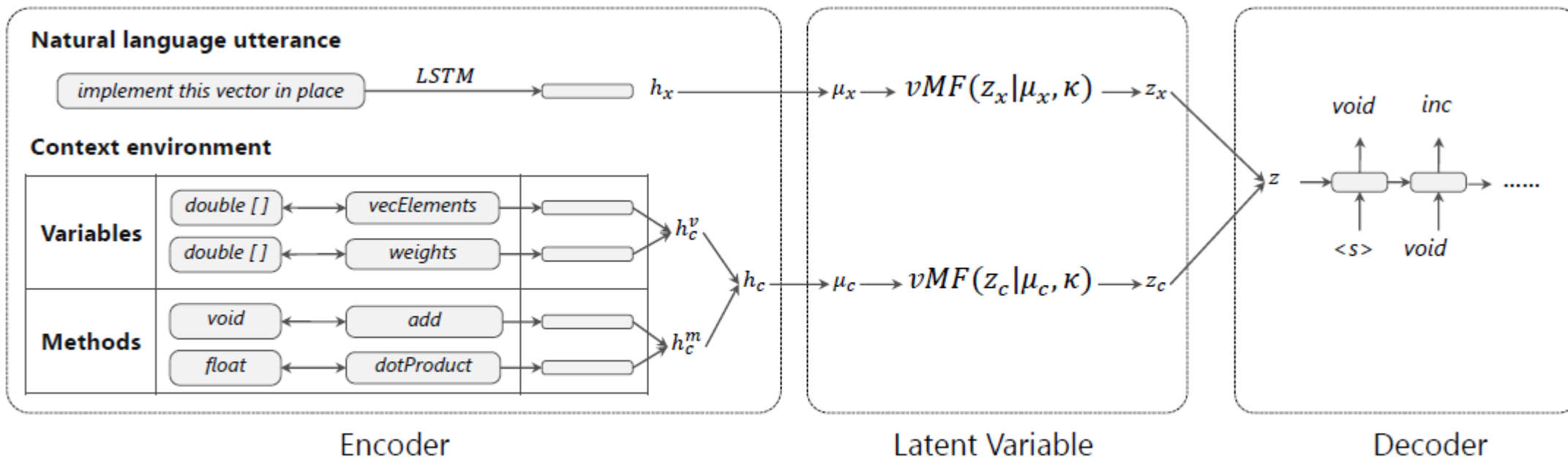


Figure 3: An overview of our context-dependent retriever.

- 基于上下文的检索模型，该模型是一个建立在变分自编码器（VAE）框架下的编码-解码（encoder-decoder）模型，将文本和上下文环境编码成一个潜层变量 z ，然后利用该变量解码出逻辑表达式。在检索的过程中，使用KL散度作为距离度量得到相似的样本。