Model-based Meta-Learning

Xu Rui
2019 08 08

Task description

- Model: $f(x) \rightarrow a$
- Task:

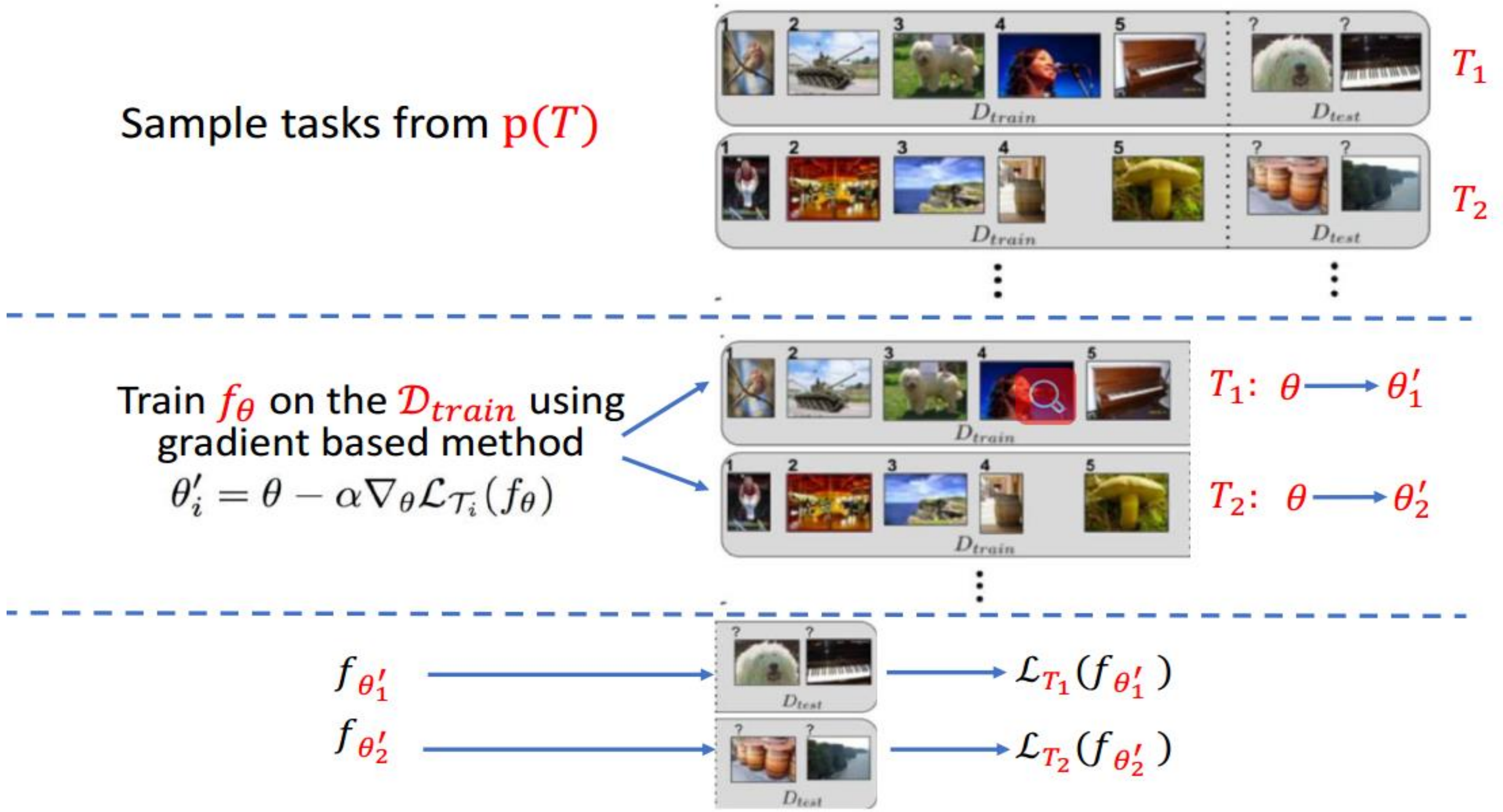
transition distribution

$$\mathcal{T} = \{\mathcal{L}(\mathbf{x}_1, \mathbf{a}_1, \ldots, \mathbf{x}_H, \mathbf{a}_H), q(\mathbf{x}_1), q(\mathbf{x}_{t+1} | \mathbf{x}_t, \mathbf{a}_t), H\}$$

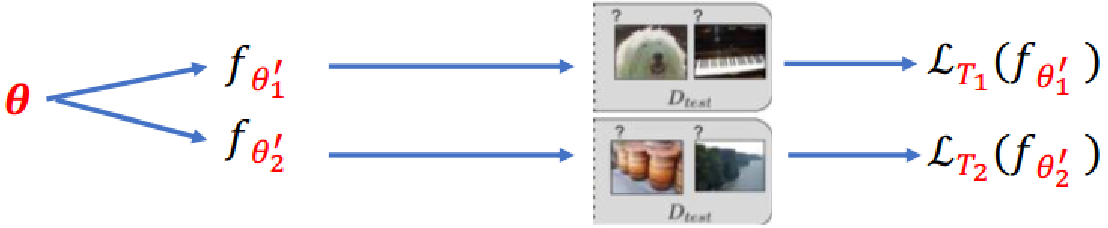
loss function　distribution over initial observations　an episode length

Model

- We want to learn the new task T_{new}



Model



Object function

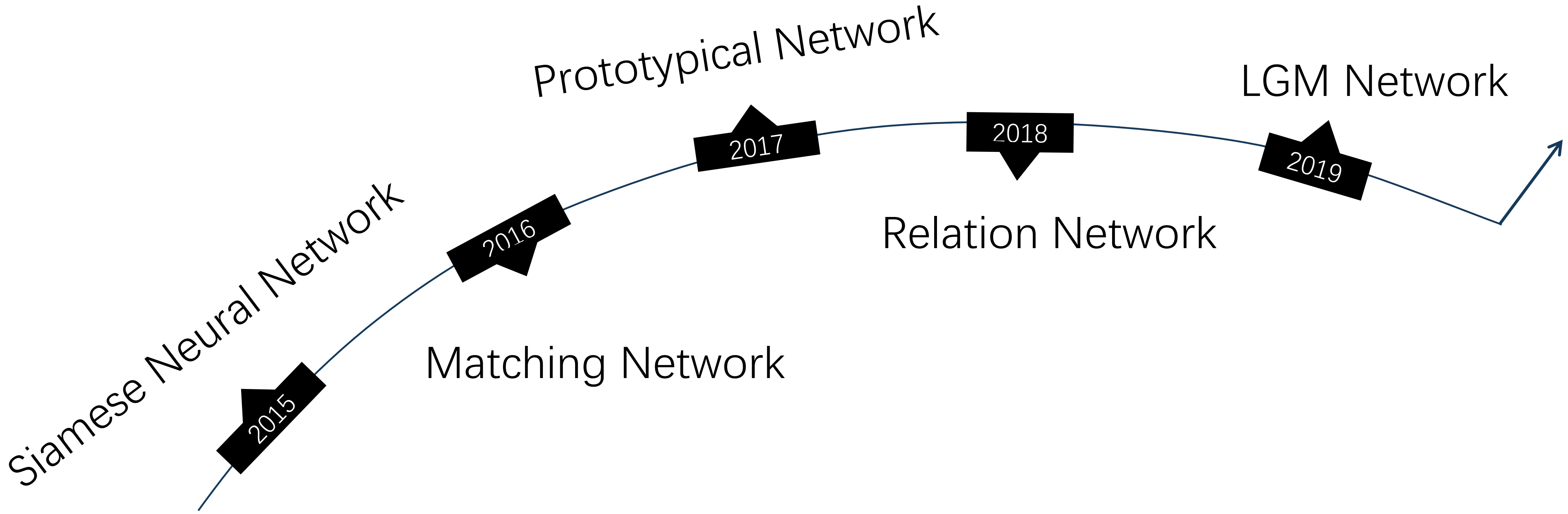
$$\min_{\theta} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i}) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)}\right)$$

θ is easy to fine-tune

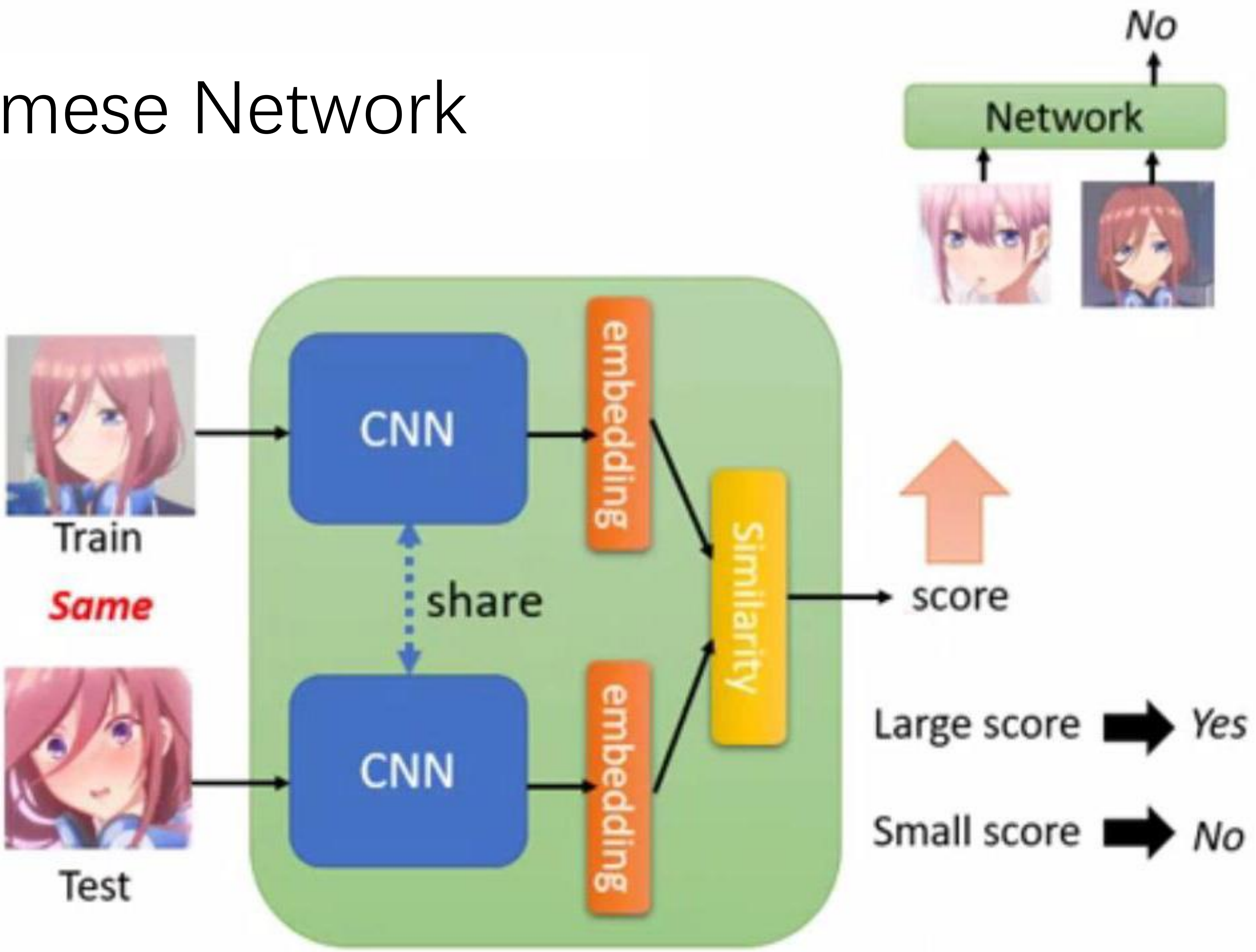
Update θ by:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})$$

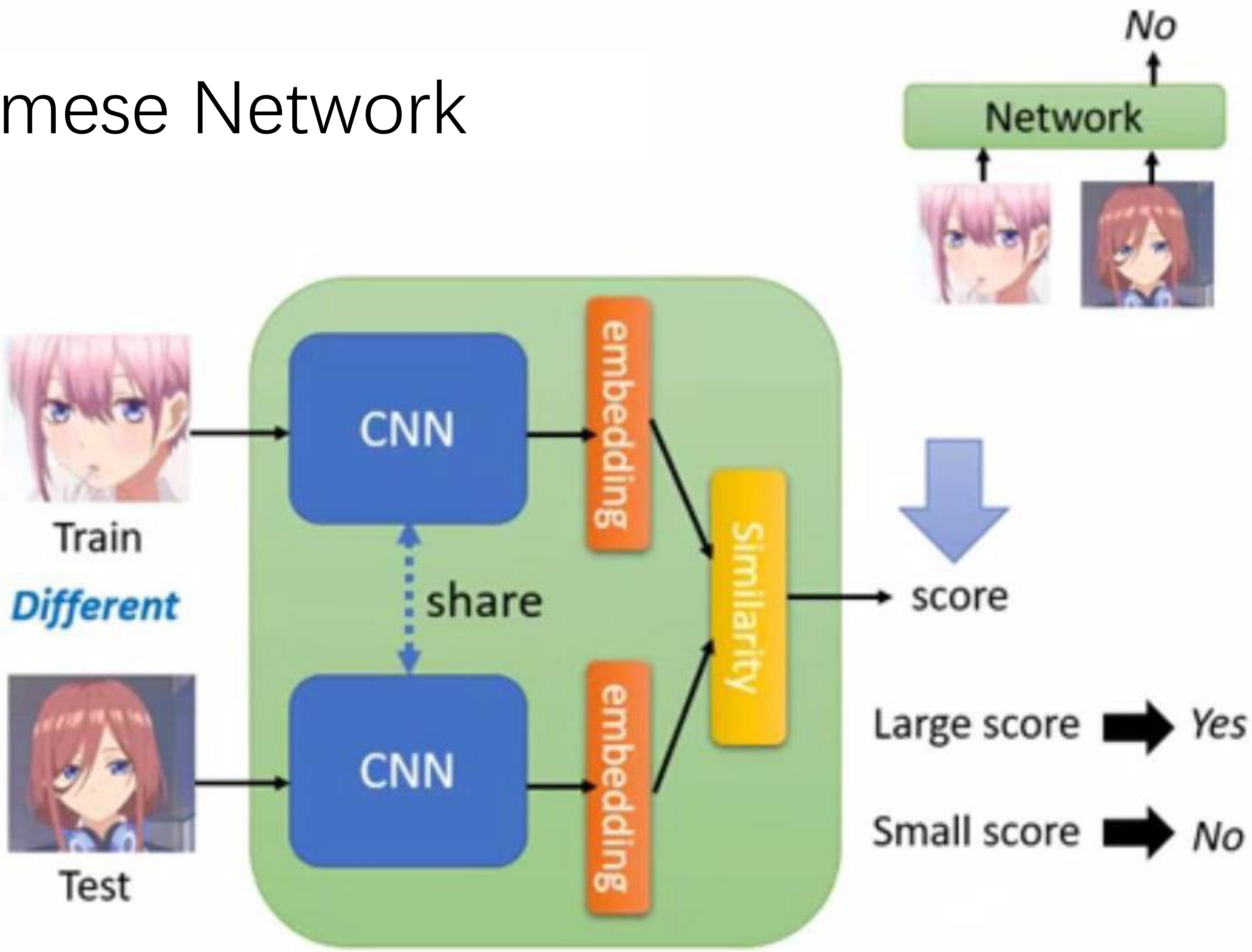
Recent study



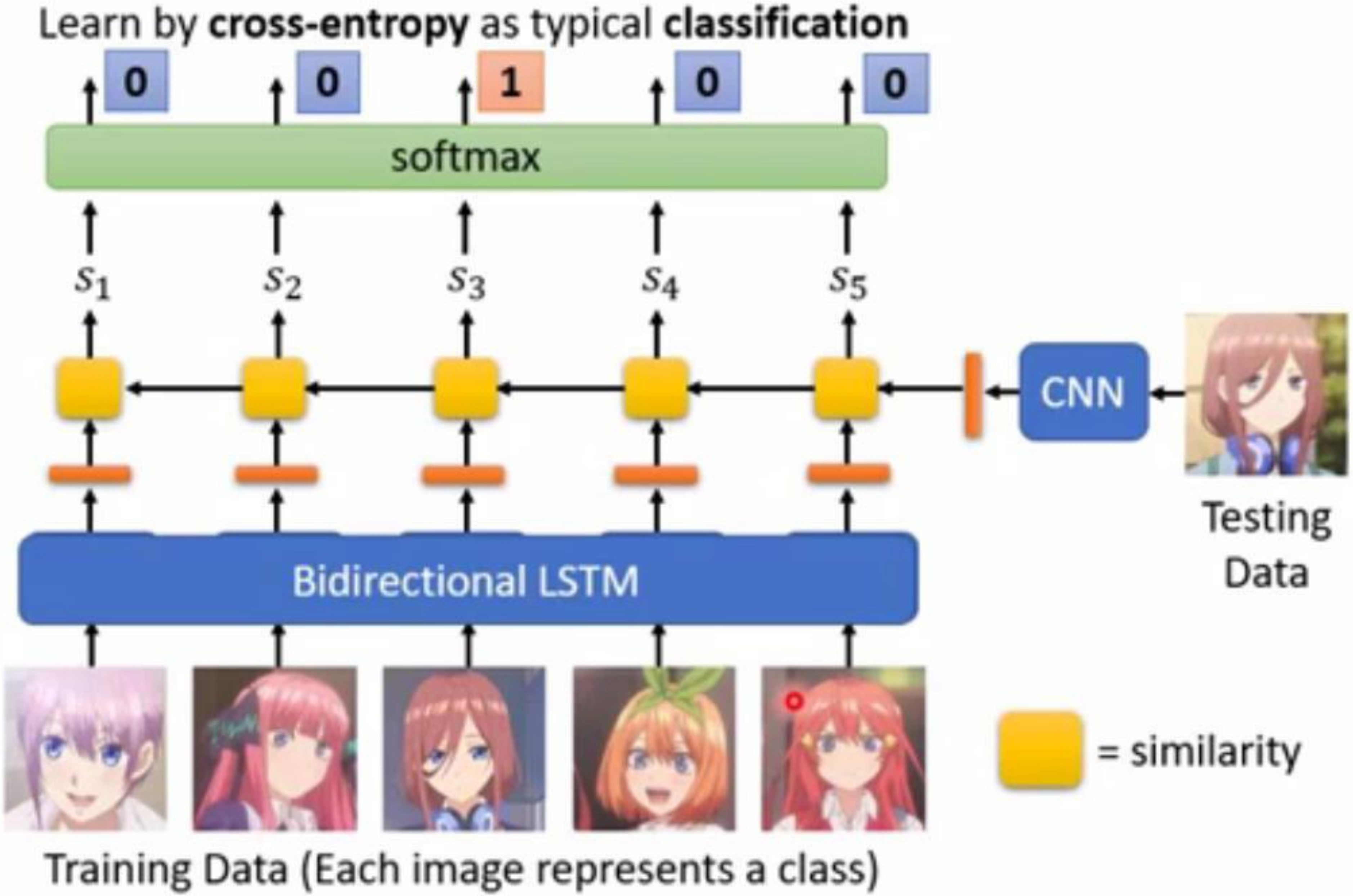
Siamese Network



Siamese Network



Matching Network

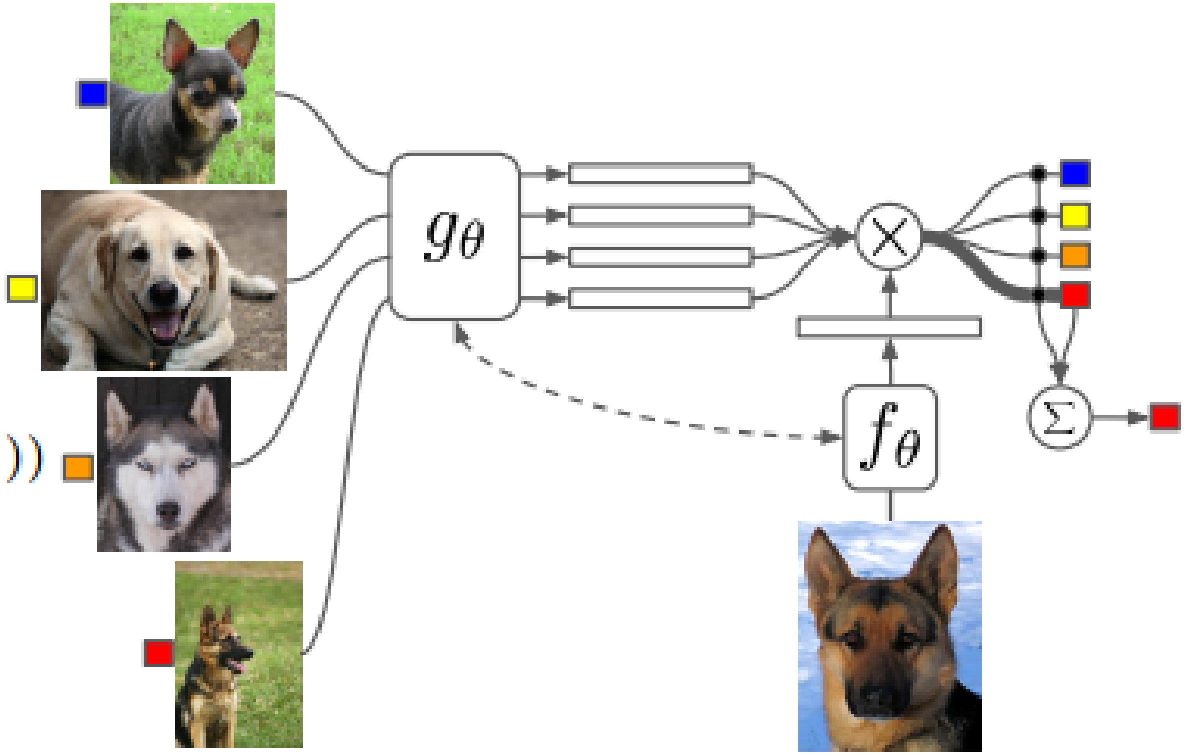


Matching Network

- Training a "**pattern matcher**"

$$\hat{y} = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i$$

$$a(\hat{x}, x_i) = e^{c(f(\hat{x}), g(x_i))} / \sum_{j=1}^{k} e^{c(f(\hat{x}), g(x_j))}$$



- Matching networks for one shot learning (2016)
  Oriol Vinyals, Charles Blundell, Timothy P. Lillicrap, Koray Kavukcuoglu, and Daan Wierstra

Matching Network

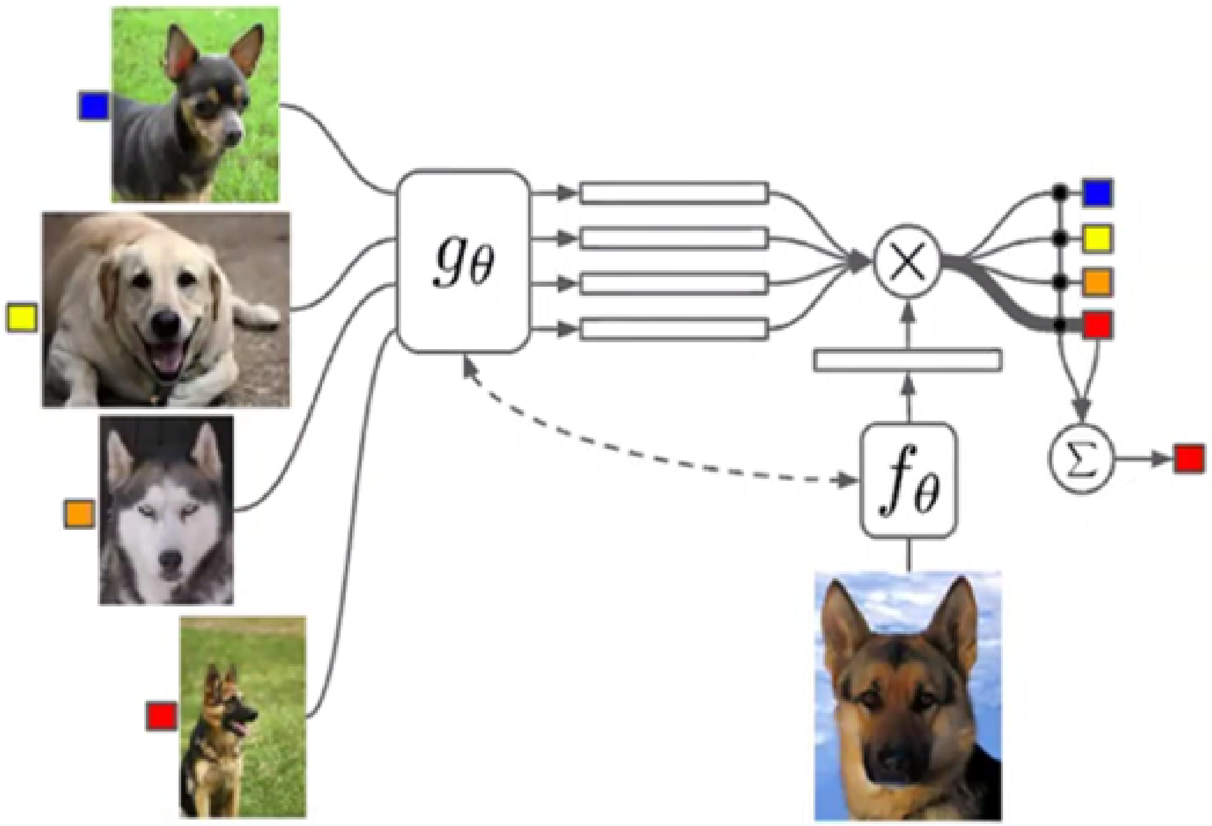
Techniques:
- One-shot learning with attention and memory
- Uniform training and testing strategy

Advantage:
- Utilize the advantage of both parametric and nonparametric learning

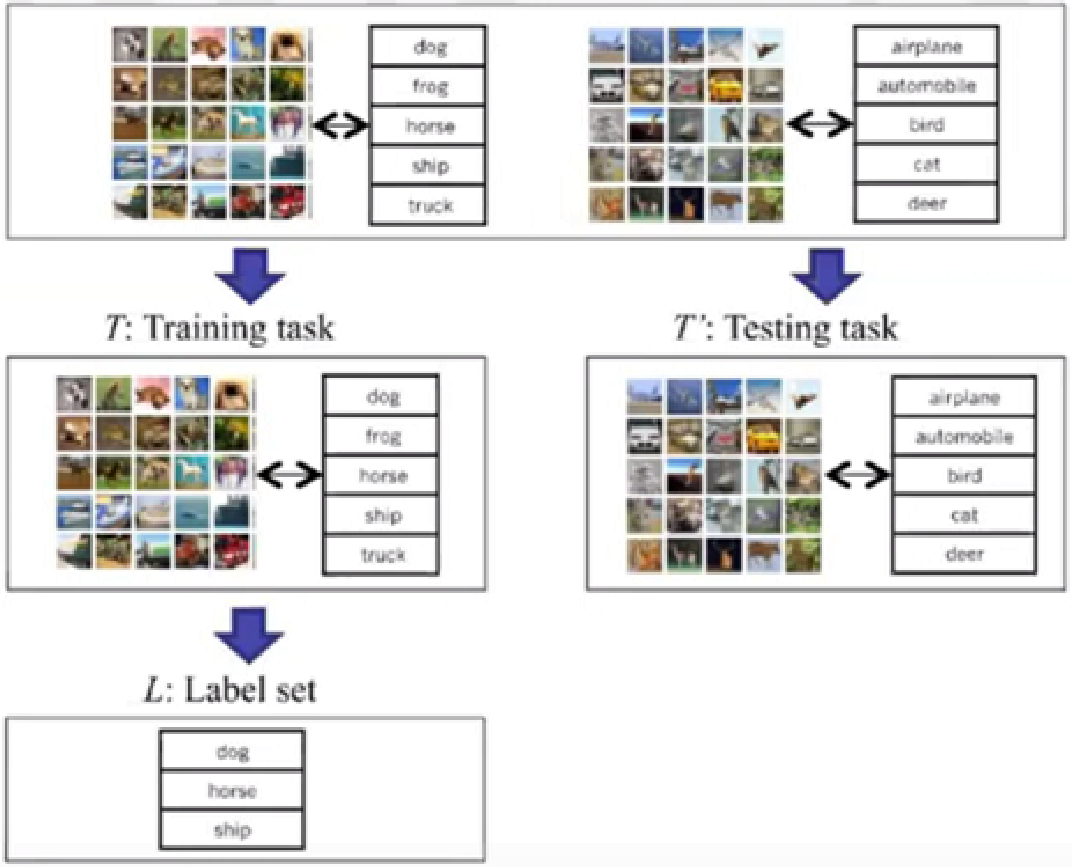
Architecture Summary:
- Differentiable nearest neighbor : incorporating the best characteristics from both parametric and nonparametric models

Results:
- Improved one-shot accuracy on ImageNet from 87.6% to 93.2% and on Omniglot from 88.0% to 93.8%

Matching Networks



Training Strategy



Attention Kernel

- Attention: Softmax over cosine distance between f(x,S) and g(x_i)

$$\hat{y} = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i \tag{1}$$

$$a(\hat{x}, x_i) = e^{c(f(\hat{x}), g(x_i))} / \sum_{j=1}^{k} e^{c(f(\hat{x}), g(x_j))}$$

- c(f(),g()) is cosine distance between target and support embedding
- Train using Cross Entropy loss
- Prediction is linear combination of labels in the support set:
  - 0.2 [1, 0, 0] + 0.5 [0, 1, 0] + 0.3 [0, 0, 1] = [0.2, 0.5, 0.3]

Full Context Embedding (g)

- **Idea**: Encode each support in context of its neighbors within support set (S)
- **Using**: Use Bidirectional LSTM

$$g(x_i, S) = \vec{h}_i + \overleftarrow{h}_i + g'(x_i)$$

$$\begin{aligned}
\vec{h}_i, \vec{c}_i &= \text{LSTM}(g'(x_i), \vec{h}_{i-1}, \vec{c}_{i-1}) \\
\overleftarrow{h}_i, \overleftarrow{c}_i &= \text{LSTM}(g'(x_i), \overleftarrow{h}_{i+1}, \overleftarrow{c}_{i+1})
\end{aligned}$$

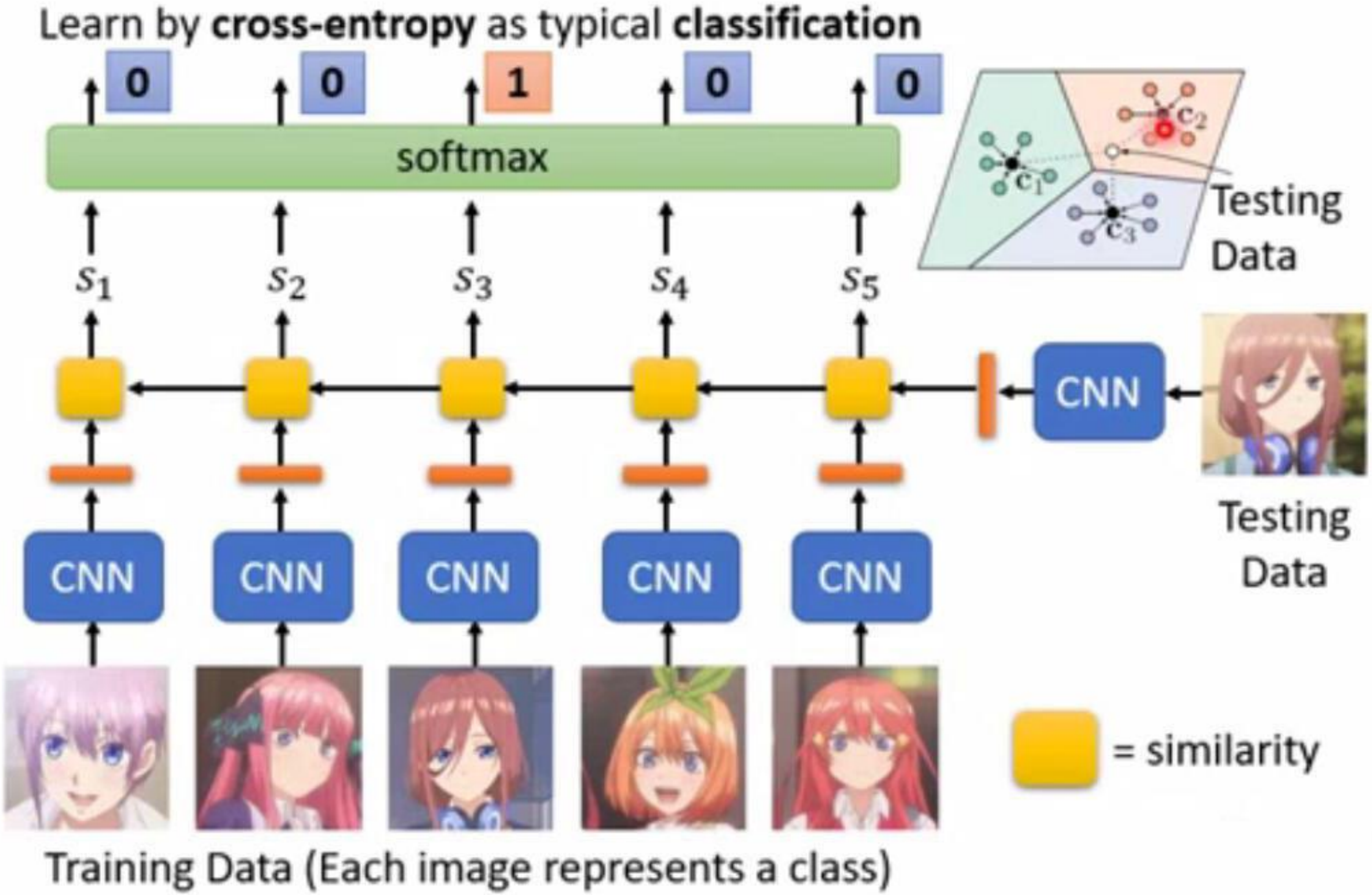
g': neural network (e.g., VGG or Inception)

Full Context Embedding (f)

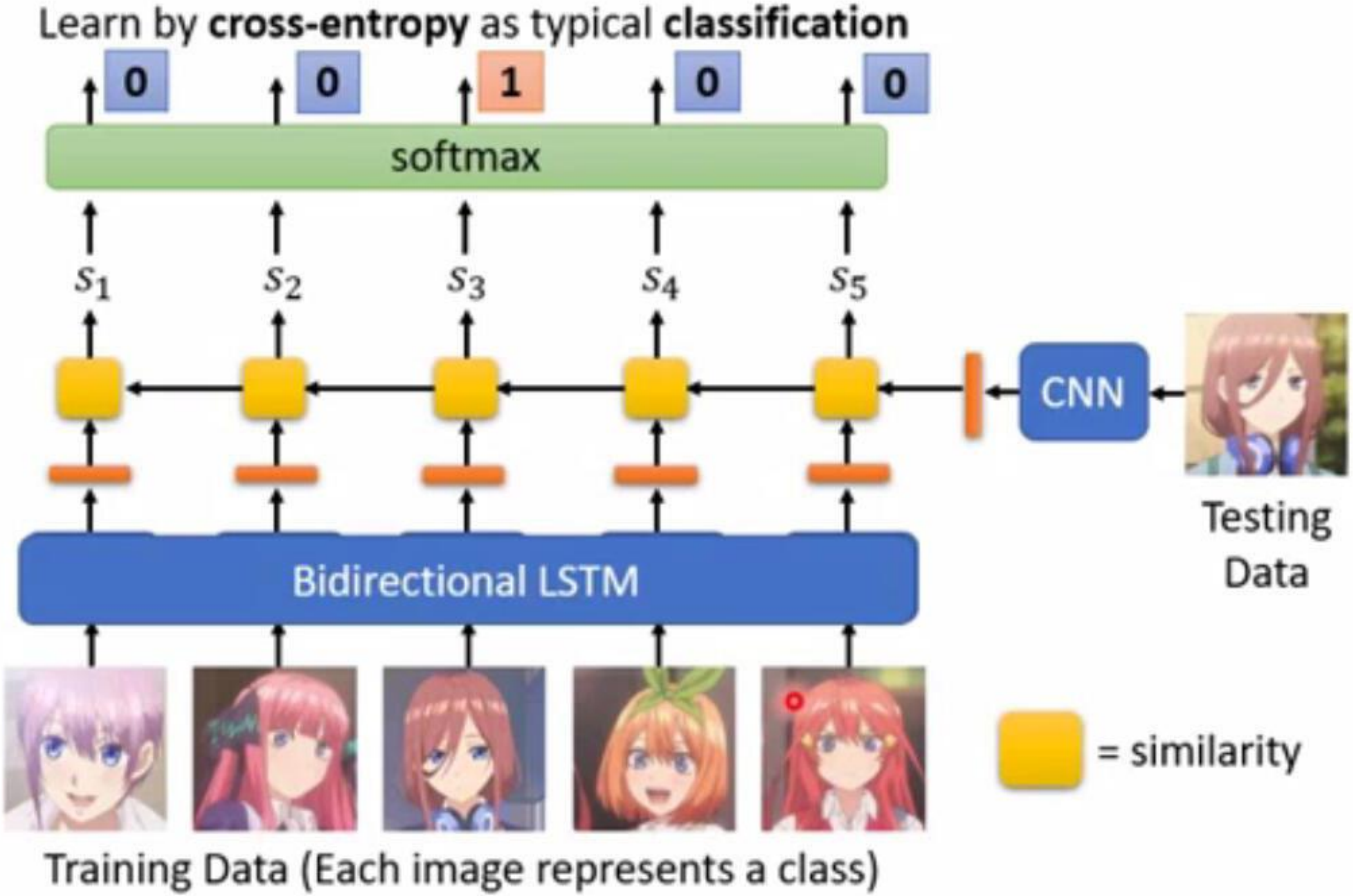
- **Idea**: Encode targets in context of its supports
- **Using**: Use Bidirectional LSTM with attention

$$
\begin{aligned}
\hat{h}_k, c_k &= \mathrm{LSTM}(f'(\hat{x}), [h_{k-1}, r_{k-1}], c_{k-1}) \\
h_k &= \hat{h}_k + f'(\hat{x}) \\
r_{k-1} &= \sum_{i=1}^{|S|} a(h_{k-1}, g(x_i)) g(x_i) \\
a(h_{k-1}, g(x_i)) &= \mathrm{softmax}(h_{k-1}^T g(x_i))
\end{aligned}
$$

Prototype Network



Matching Network



Prototypical Network	Matching Network
Use euclidean distance	Use cosine similarity measure
Linear Classifier	Weighted Nearest Neighbor Classifier
Simple	Complex

$$\mathbf{c}_k = \frac{1}{|S_k|} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_\phi(\mathbf{x}_i)$$

$$\hat{y} = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i$$

$$p_\phi(y = k \mid \mathbf{x}) = \frac{\exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_k))}{\sum_{k'} \exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_{k'}))}$$

$$a(\hat{x}, x_i) = e^{c(f(\hat{x}), g(x_i))} / \sum_{j=1}^{k} e^{c(f(\hat{x}), g(x_j))}$$

Prototype Network

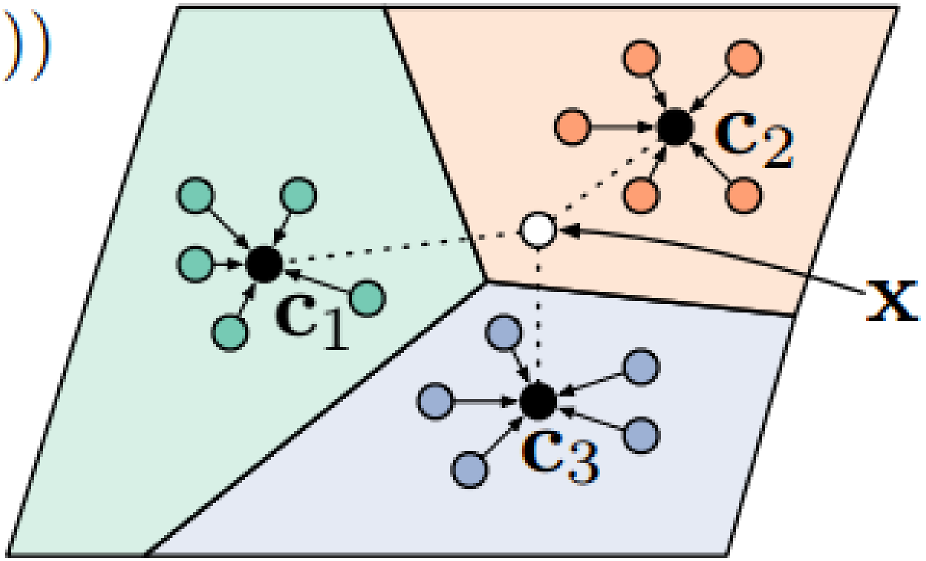
- Training a "**prototype extractor**"

$$p_\phi(y=k\,|\,\mathbf{x}) = \frac{\exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_k))}{\sum_{k'} \exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_{k'}))}$$

$$\mathbf{c}_k = \frac{1}{|S_k|} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_\phi(\mathbf{x}_i)$$

$$S_k = \{(\mathbf{x}_i, y_i) | y_i = k, (\mathbf{x}_i, y_i) \in D_{train}\}$$

$$\phi \equiv \Theta$$



- Prototypical Networks for Few-shot Learning (2017)
  Jake Snell, Kevin Swersky and Richard Zemel

Prototype Network

Algorithm 1 Training episode loss computation for prototypical networks. N is the number of examples in the training set, K is the number of classes in the training set, $N_C \leq K$ is the number of classes per episode, N_S is the number of support examples per class, N_Q is the number of query examples per class. $\text{RandomSample}(S, N)$ denotes a set of N elements chosen uniformly at random from set S, without replacement.

Input: Training set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, where each $y_i \in \{1, \ldots, K\}$. $\mathcal{D}_k$ denotes the subset of $\mathcal{D}$ containing all elements $(\mathbf{x}_i, y_i)$ such that $y_i = k$.

Output: The loss J for a randomly generated training episode.

$V \leftarrow \text{RandomSample}(\{1, \ldots, K\}, N_C)$ ▷ Select class indices for episode

for k in $\{1, \ldots, N_C\}$ **do**

$S_k \leftarrow \text{RandomSample}(\mathcal{D}_{V_k}, N_S)$ ▷ Select support examples

$Q_k \leftarrow \text{RandomSample}(\mathcal{D}_{V_k} \setminus S_k, N_Q)$ ▷ Select query examples

$$\mathbf{c}_k \leftarrow \frac{1}{N_C} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_\phi(\mathbf{x}_i)$$ ▷ Compute prototype from support examples

end for

$J \leftarrow 0$ ▷ Initialize loss

for k in $\{1, \ldots, N_C\}$ **do**

for $(\mathbf{x}, y)$ in Q_k **do**

$$J \leftarrow J + \frac{1}{N_C N_Q} \left[d(f_\phi(\mathbf{x}), \mathbf{c}_k)) + \log \sum_{k'} \exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_k)) \right]$$ ▷ Update loss

end for

end for

Reference:

[1] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. “Siamese neural networks for one-shot image recognition.” ICML Deep Learning Workshop. 2015.
[2] Oriol Vinyals, et al. “Matching networks for one shot learning.” NIPS. 2016.
[3] Jake Snell, Kevin Swersky ,.Richard S. Zemel . “ Prototypical Networks for Few-shot Learning ” arXiv:1703.05175v2 [cs.LG] 19 Jun 2017
[4] Joaquin Vanschoren. “ Meta-Learning: A Survey ” Eindhoven University of Technology, arXiv:1810.03548v1 [cs.LG] 8 Oct 2018

THANK YOU