Matching Networks for One Shot Learning

References：
Vinyals O, Blundell C, Lillicrap T, et al. Matching Networks for One Shot Learning[J]. 2016.

Download: https://arxiv.org/abs/1606.04080

段云志
超星思维Nova Mind
2017/6/25

Outline

- Background
- Introduction
  - Introduce to One Shot Learning
  - Introduce to MANN
- Model
  - Motivation
  - Matching Networks
  - Backpropagation
- Experiments
- Summary

- Background
- Introduction
  - Introduce to One Shot Learning
  - Introduce to MANN
- Model
  - Motivation
  - Matching Networks
  - Backpropagation
- Experiments
- Summary

Background

- Learning from a few examples: remains challenge
- new data: models must relearn parameters
- Memory-augmented neural network has the ability to make accurate predictions after only a few samples

Introduction

- Introduce to One Shot Learning
  - What is one-shot learning?
  - Why do we need one-shot learning?
- Introduce to Neural Turing Machine(NTM)

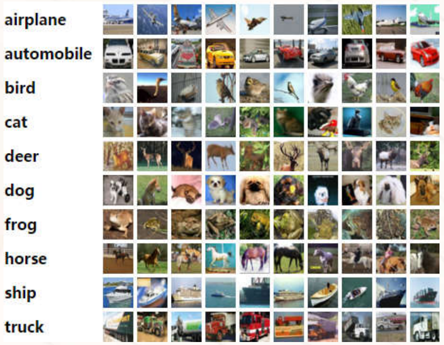
What is one-shot learning?

- One-shot?

L :Data set

S:Support set

B:Batch



L

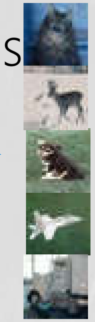
Sampling N labels

N-ways

cat
deer
dog
airplane
truck

Sampling k examples from each label

K-shot



S

sampling 1 example

$\hat{x}$

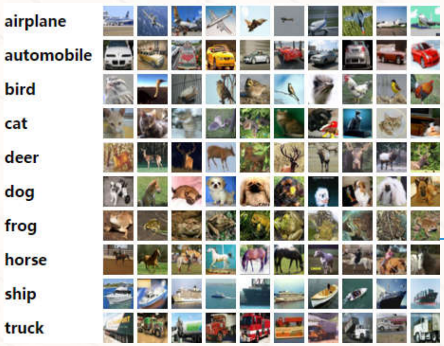
B

N-ways k-shot learning

Task:classify $\hat{x}$ into 5 classes, {cat,deer,dog,airplane,truck}, using support set S

What is one-shot learning?

- Machine Learning Principle:Test and Train conditions Must Match
- Separate labels for training and testing
  testing phase are not used in training phase !!!



T:Traning task

cat
deer
dog
airplane
truck

T′ :Testing task

automobile
bird
frog
horse
ship

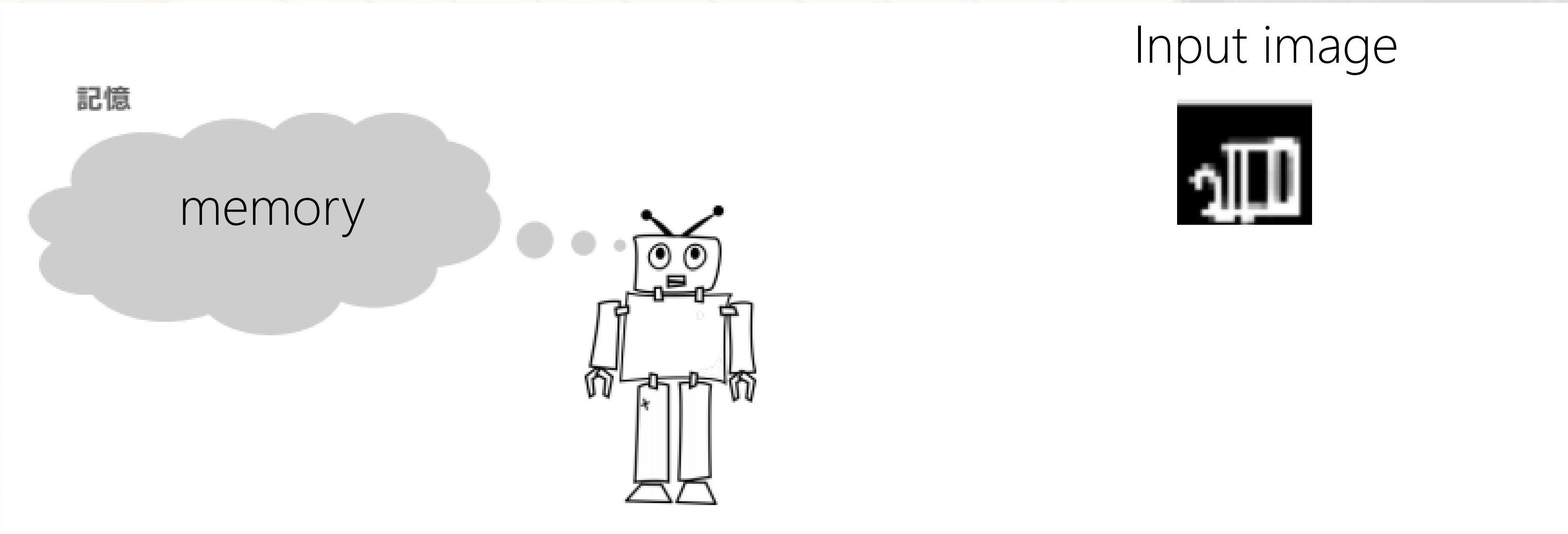
Why do we need one-shot learning?

- Problems:
  can not get enough training data

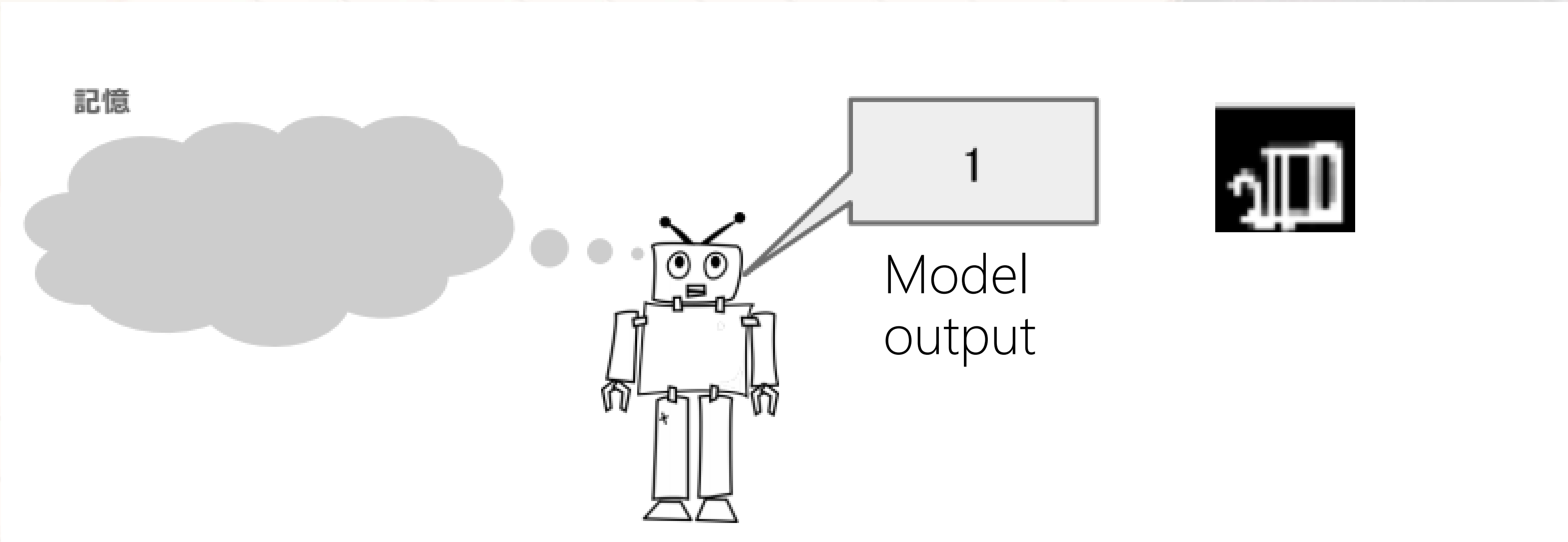
- few data for training/testing?

Neural Turing Machine(NTM)

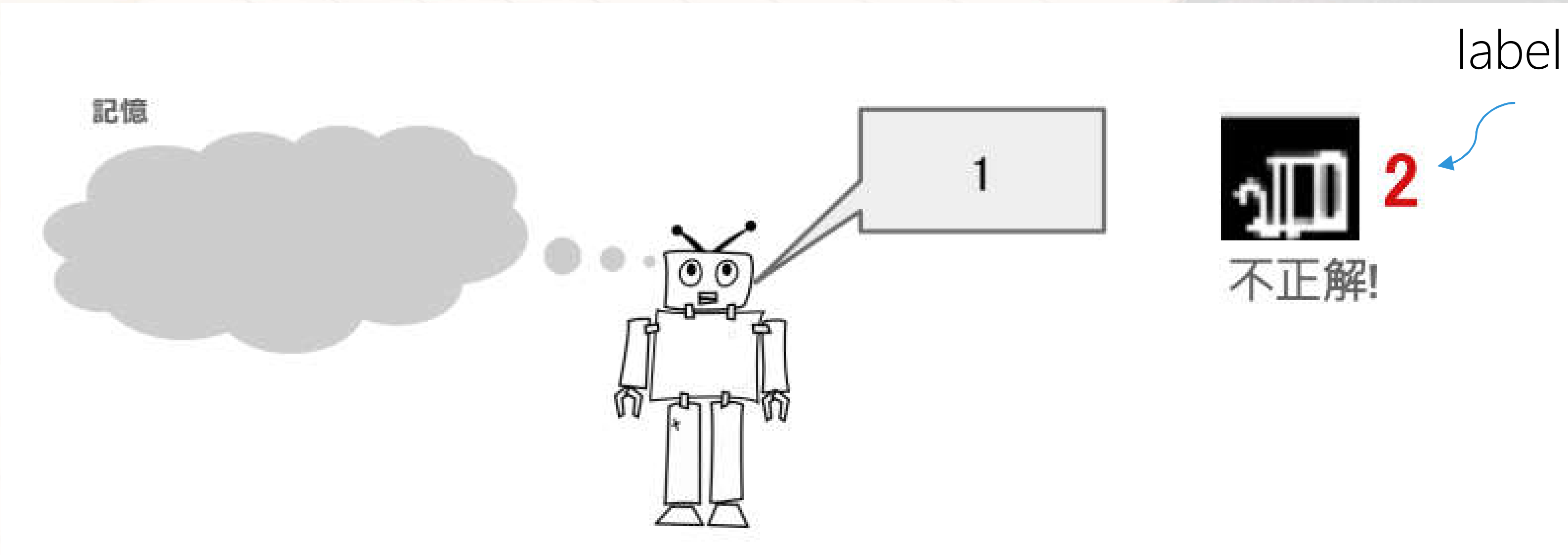
Menory will upgate with traning



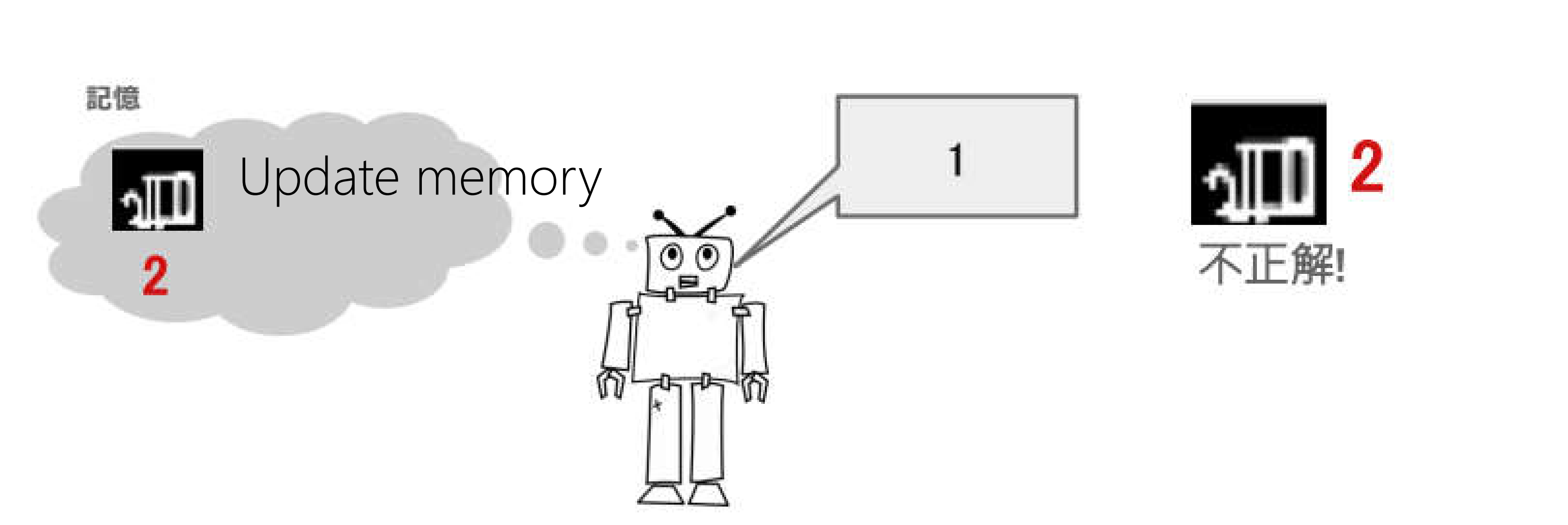
Neural Turing Machine(NTM)



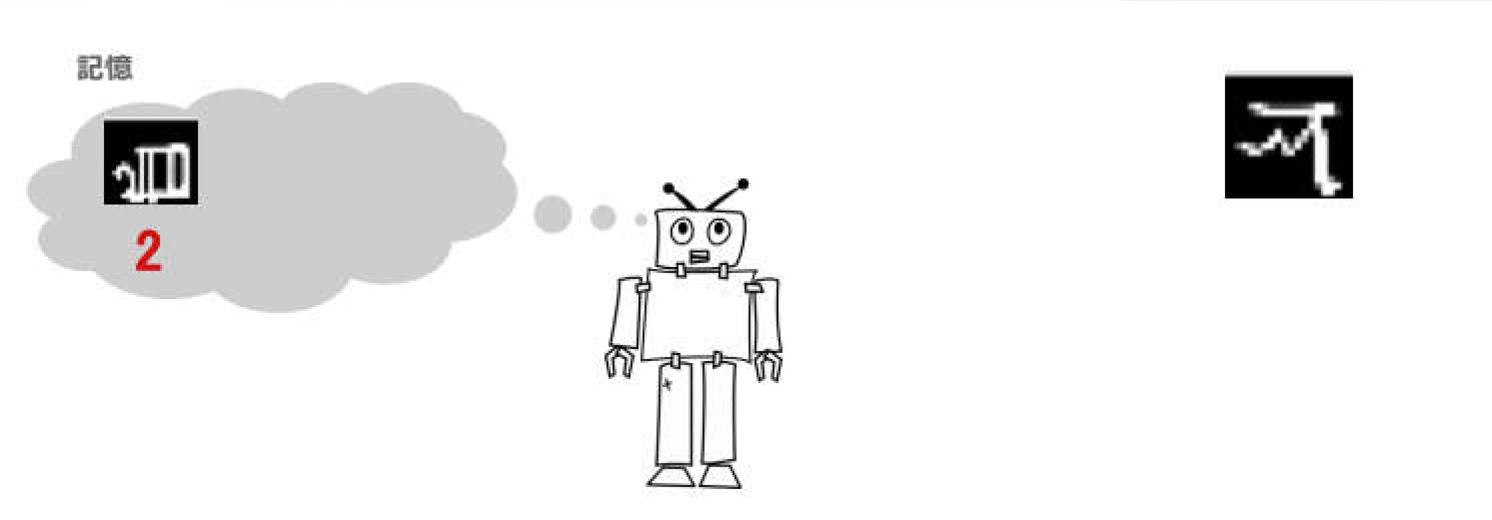
Neural Turing Machine(NTM)



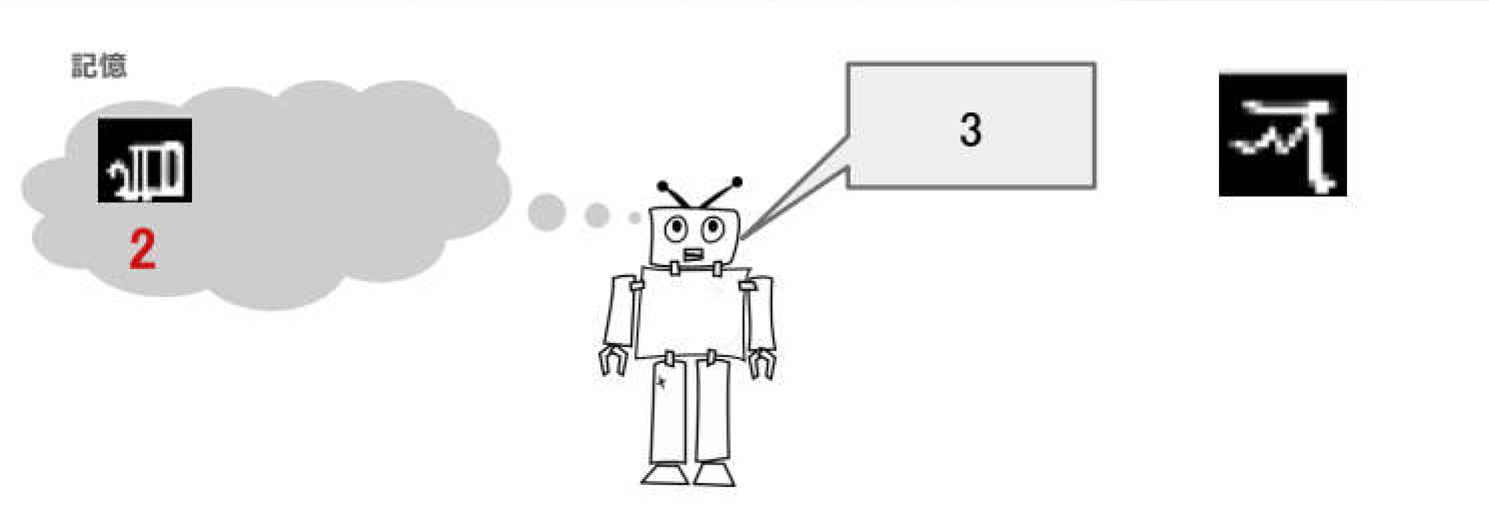
Neural Turing Machine(NTM)



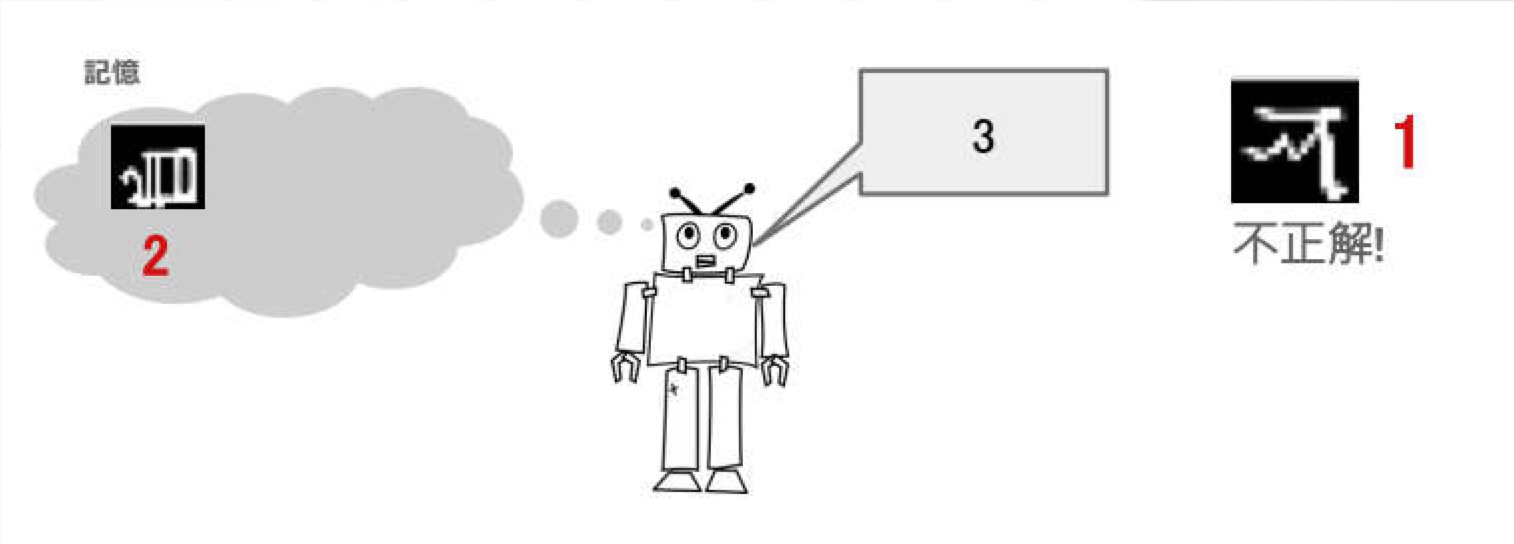
Neural Turing Machine(NTM)



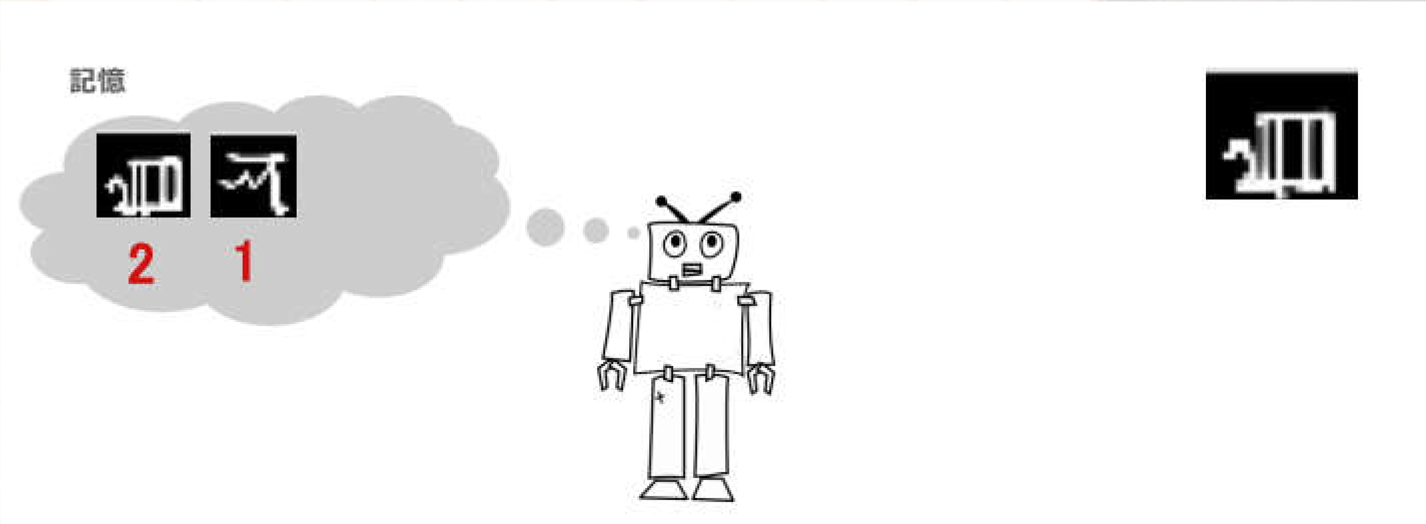
Neural Turing Machine(NTM)



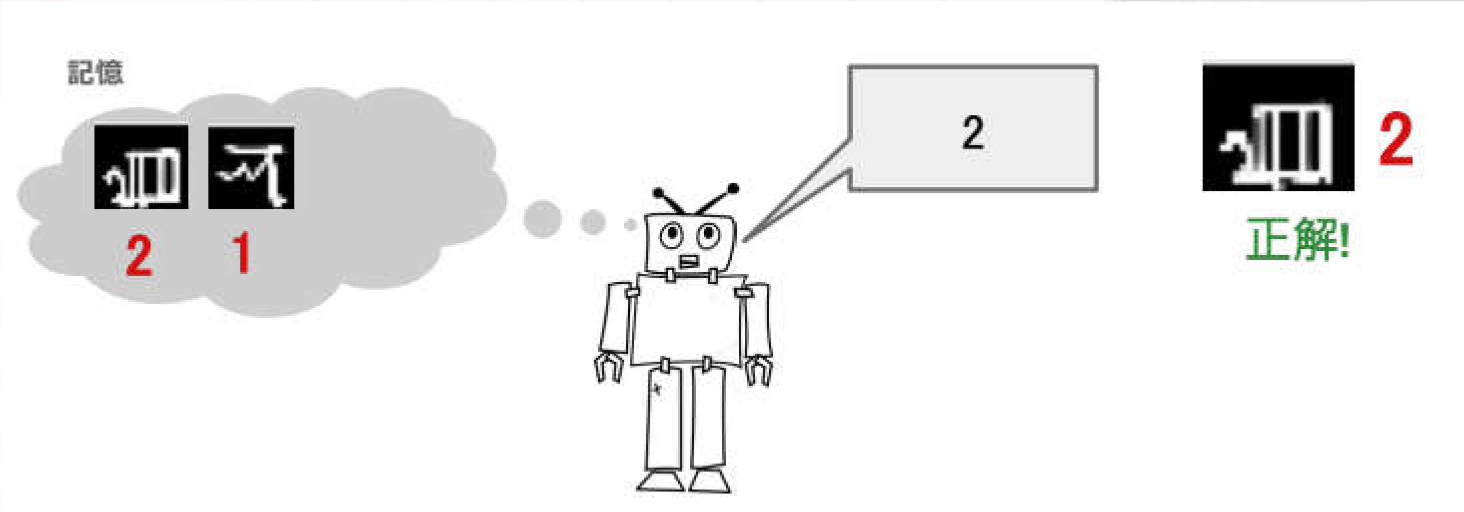
Neural Turing Machine(NTM)



Neural Turing Machine(NTM)



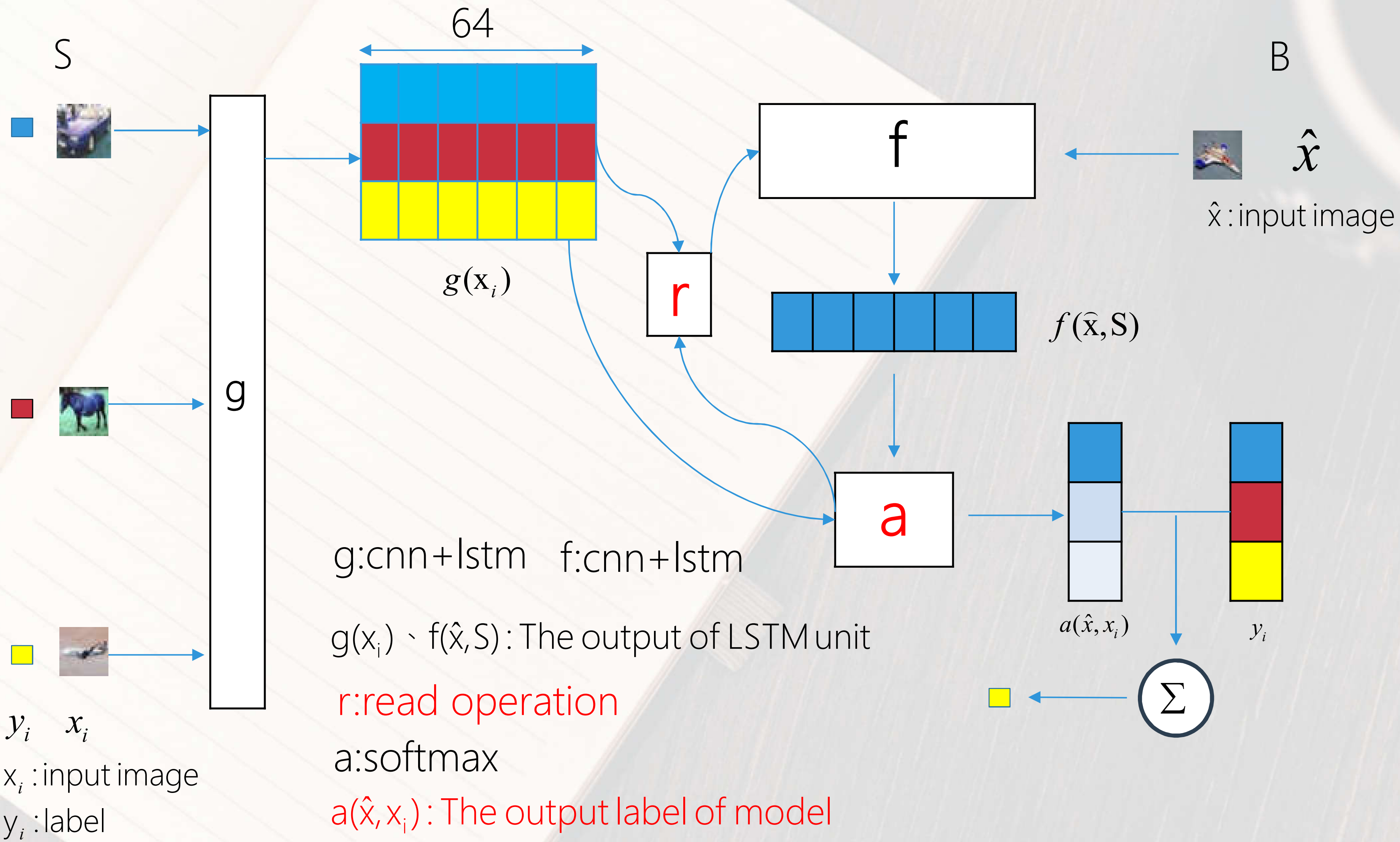
Neural Turing Machine(NTM)



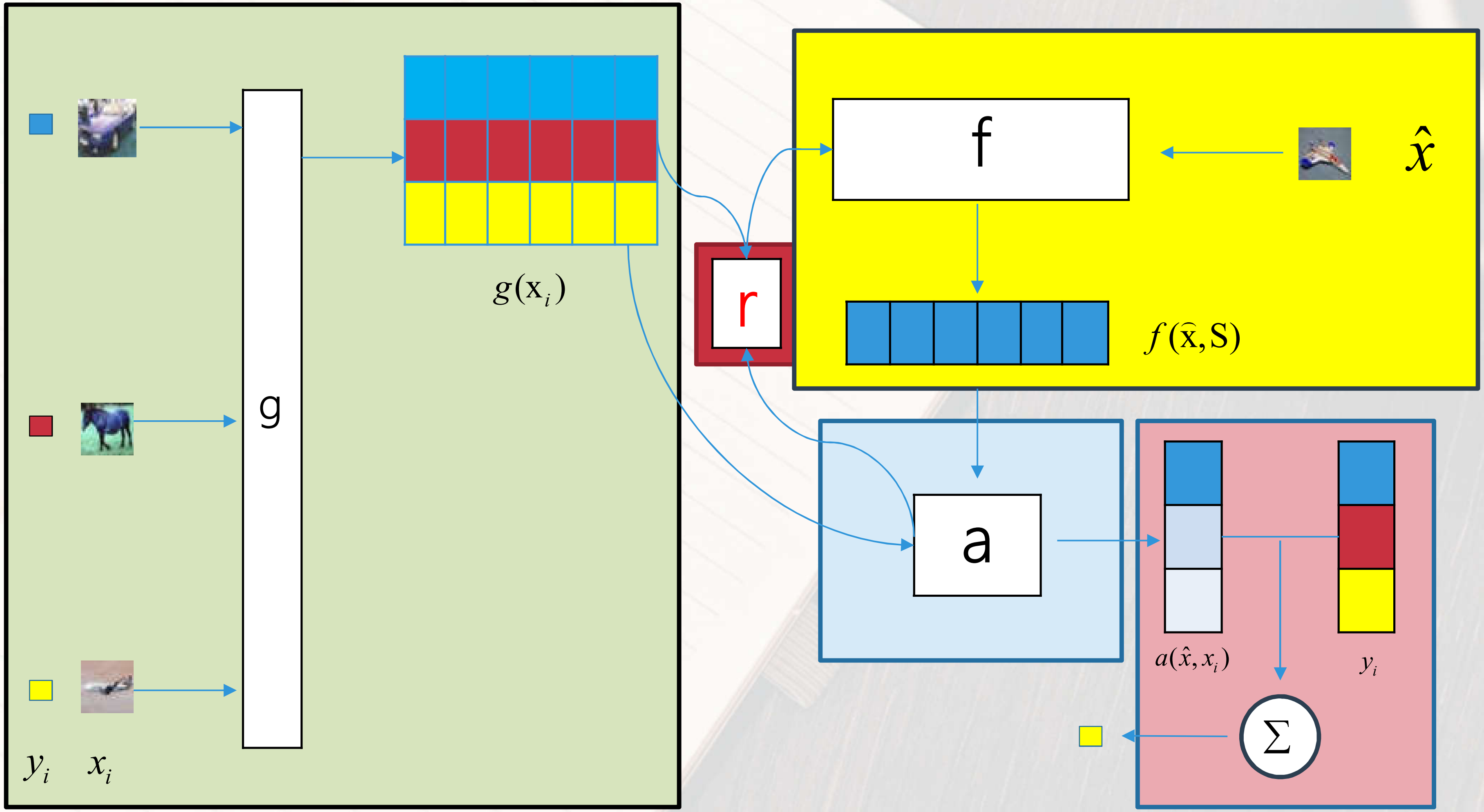
Model: Matching Networks

- Motivation:
  - Few data for traning
  - Model should be update all the time
  - Non-parametric models performance depends on the chosen metric
- Model
  Matching Networks(non-parametric components)

Matching Networks

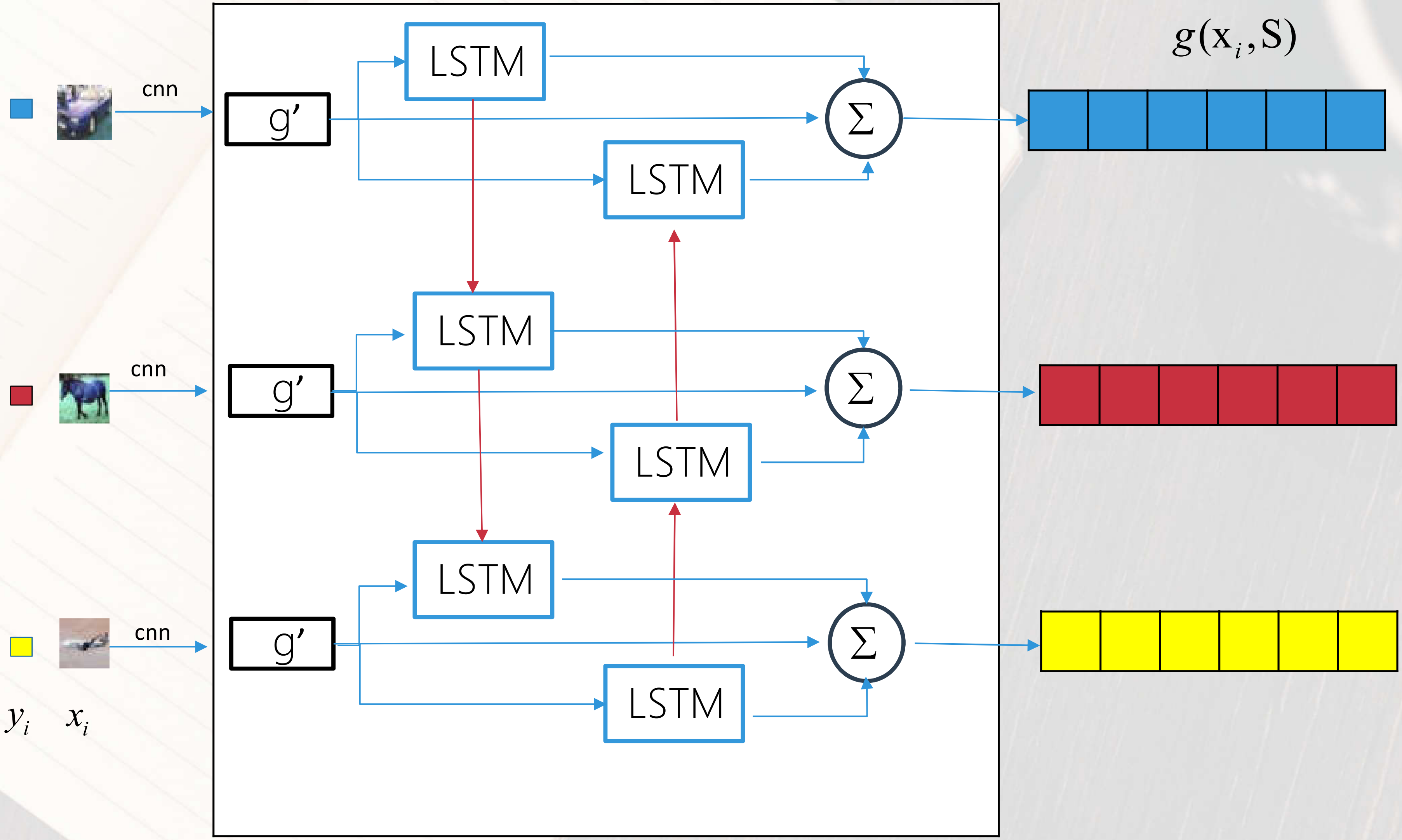


Matching Networks

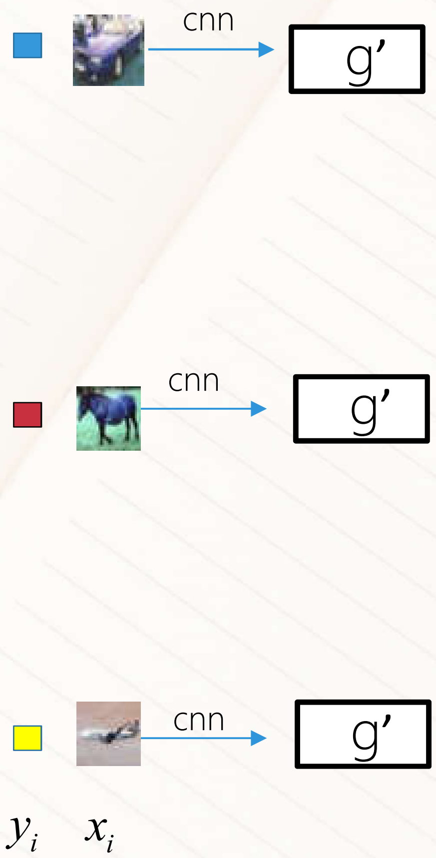


S:Support set

Matching Networks



Matching Networks



g' :Neural network

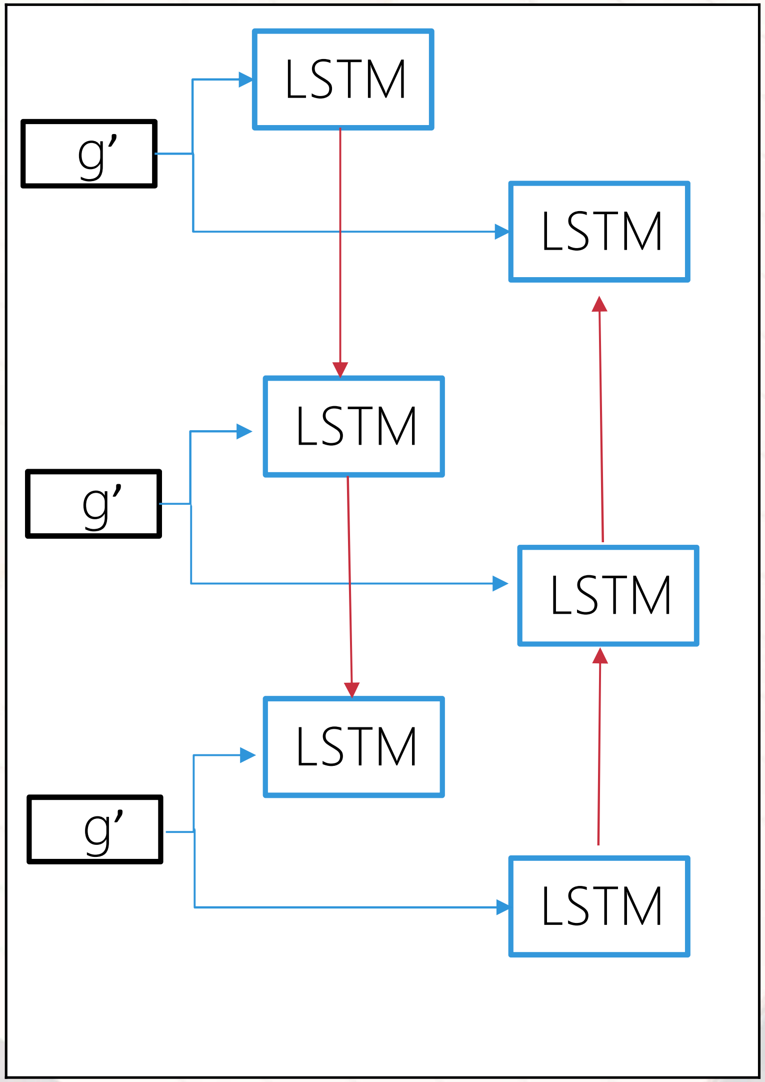
Input: image size=28*28

Batch size:32

Model:
VGG model

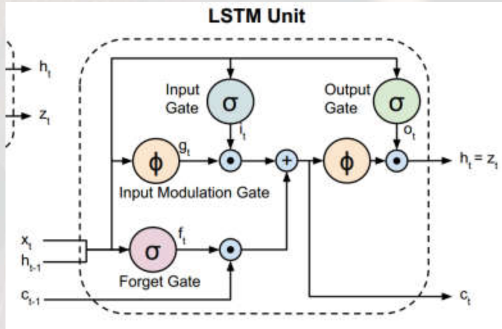
Output:
64*1 vector

VGG model please reference:
https://arxiv.org/abs/1409.1556



Input: g'

Output: $\vec{h}_i, \vec{c}_i, \overleftarrow{h}_i, \overleftarrow{c}_i$



$$\vec{h}_i, \vec{c}_i = LSTM(g'(x_i), \vec{h}_{i-1}, \vec{c}_{i-1})$$

$$\overleftarrow{h}_i, \overleftarrow{c}_i = LSTM(g'(x_i), \overleftarrow{h}_{i+1}, \overleftarrow{c}_{i+1})$$

LSTM DEMO ： $x = g'$

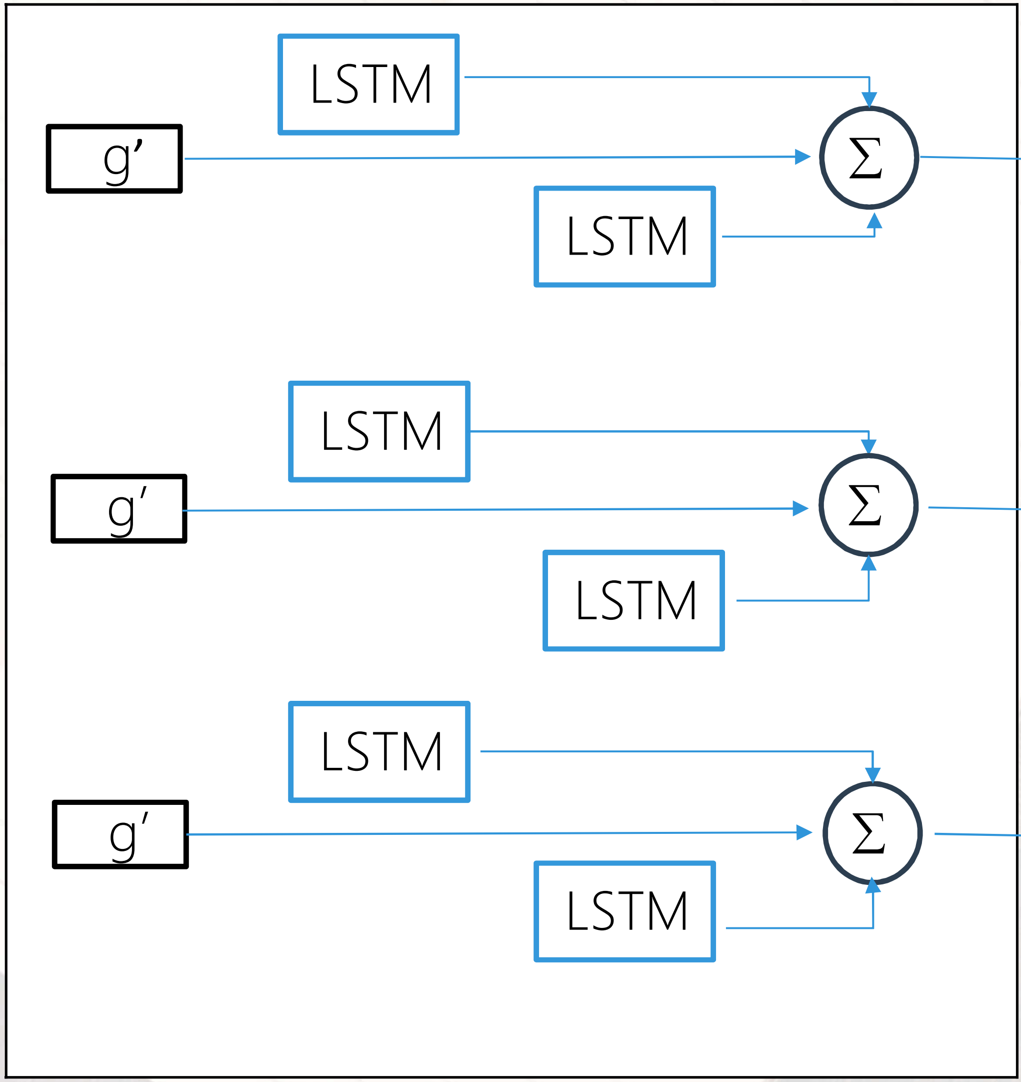
$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \quad f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \quad g_t = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$

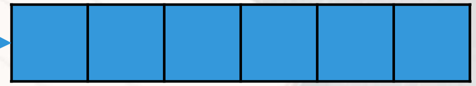
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \quad h_t = o_t \odot \tanh(c_t)$$

Write operation

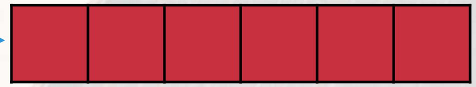
Matching Networks



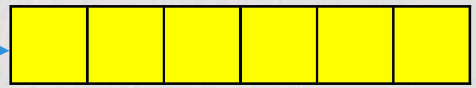
$g(\mathrm{x}_i, \mathrm{S})$



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



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

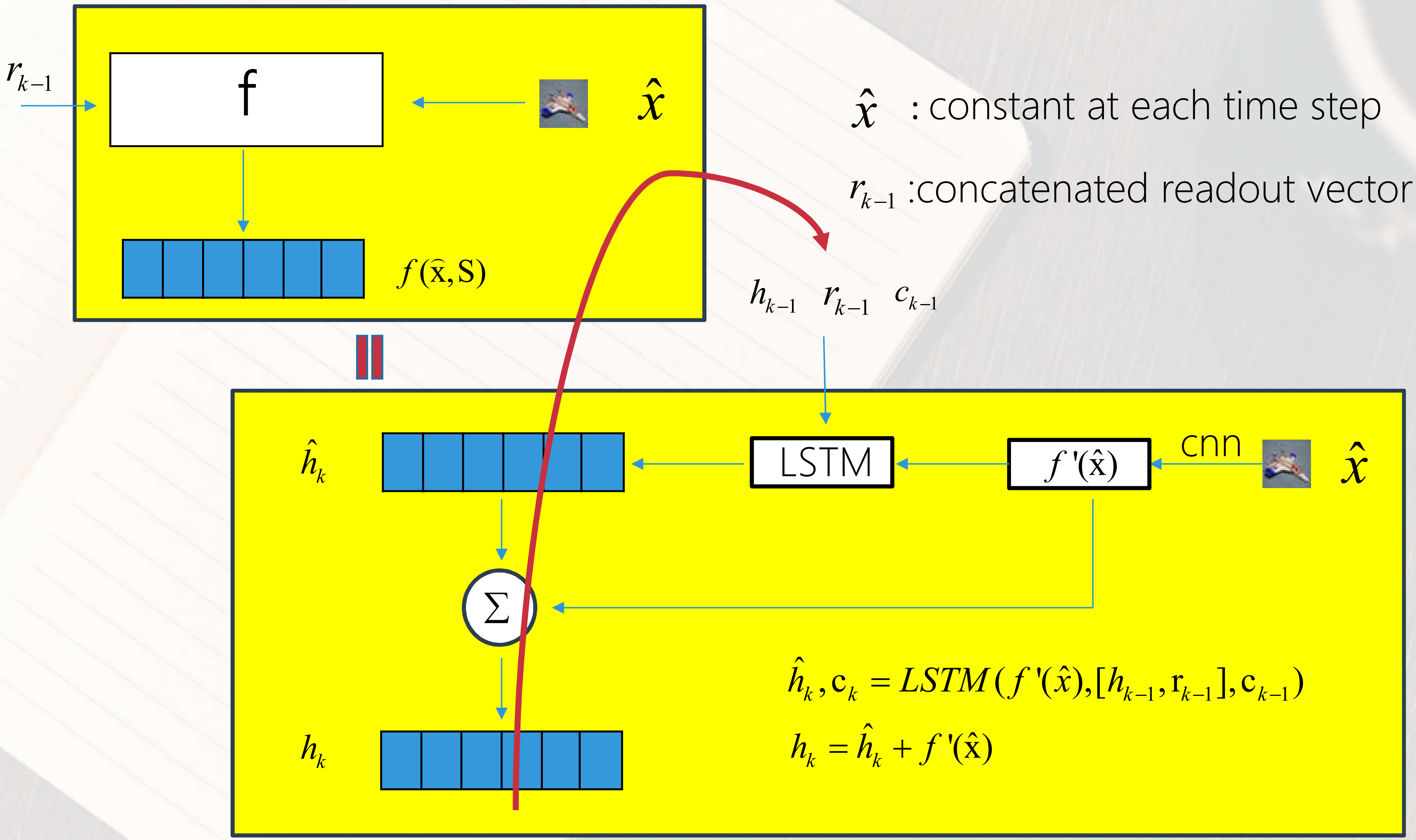


$$g(\mathrm{x}_i, \mathrm{S}) = \vec{h}_i + \overleftarrow{h}_i + g'(x_i)$$

Matching Networks

$\hat{x}$: constant at each time step

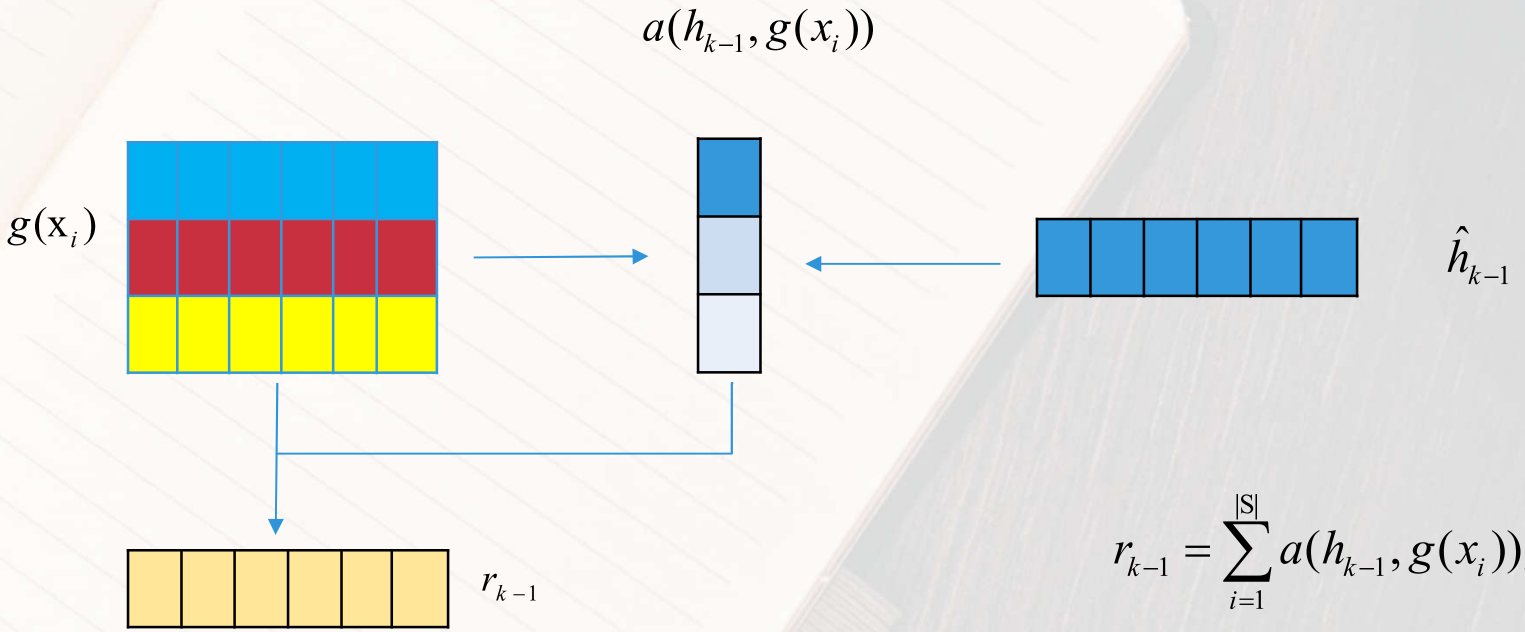
r_{k-1} :concatenated readout vector r



$$\hat{h}_k, c_k = LSTM(f'(\hat{x}), [h_{k-1}, r_{k-1}], c_{k-1})$$

$$h_k = \hat{h}_k + f'(\hat{x})$$

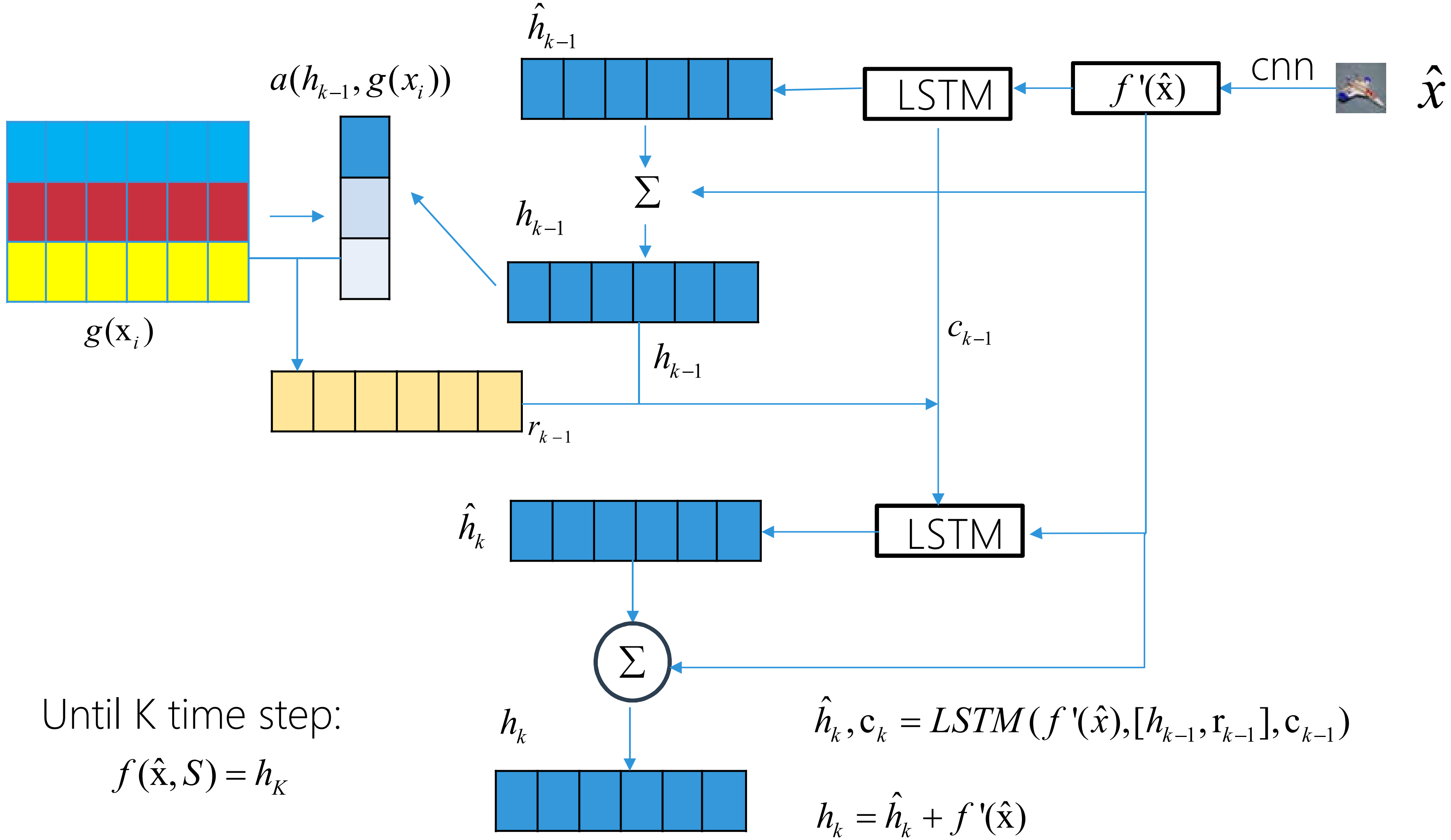
Calculate the relevance :



$$r_{k-1} = \sum_{i=1}^{|S|} a(h_{k-1}, g(x_i)) g(x_i)$$

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

r (read)is a sum of g weighted according to the relevance to h



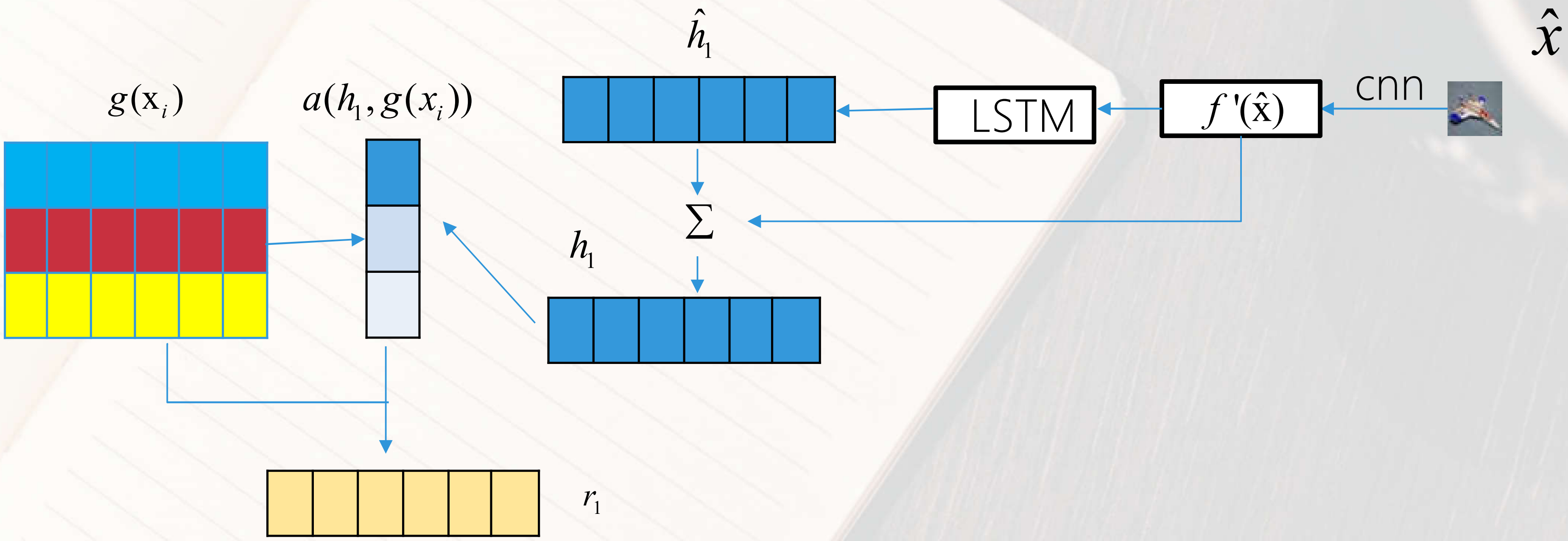
Until K time step:

$$f(\hat{x}, S) = h_K$$

$$\hat{h}_k, c_k = LSTM(f'(\hat{x}), [h_{k-1}, r_{k-1}], c_{k-1})$$

$$h_k = \hat{h}_k + f'(\hat{x})$$

How to calculate h_1 ?

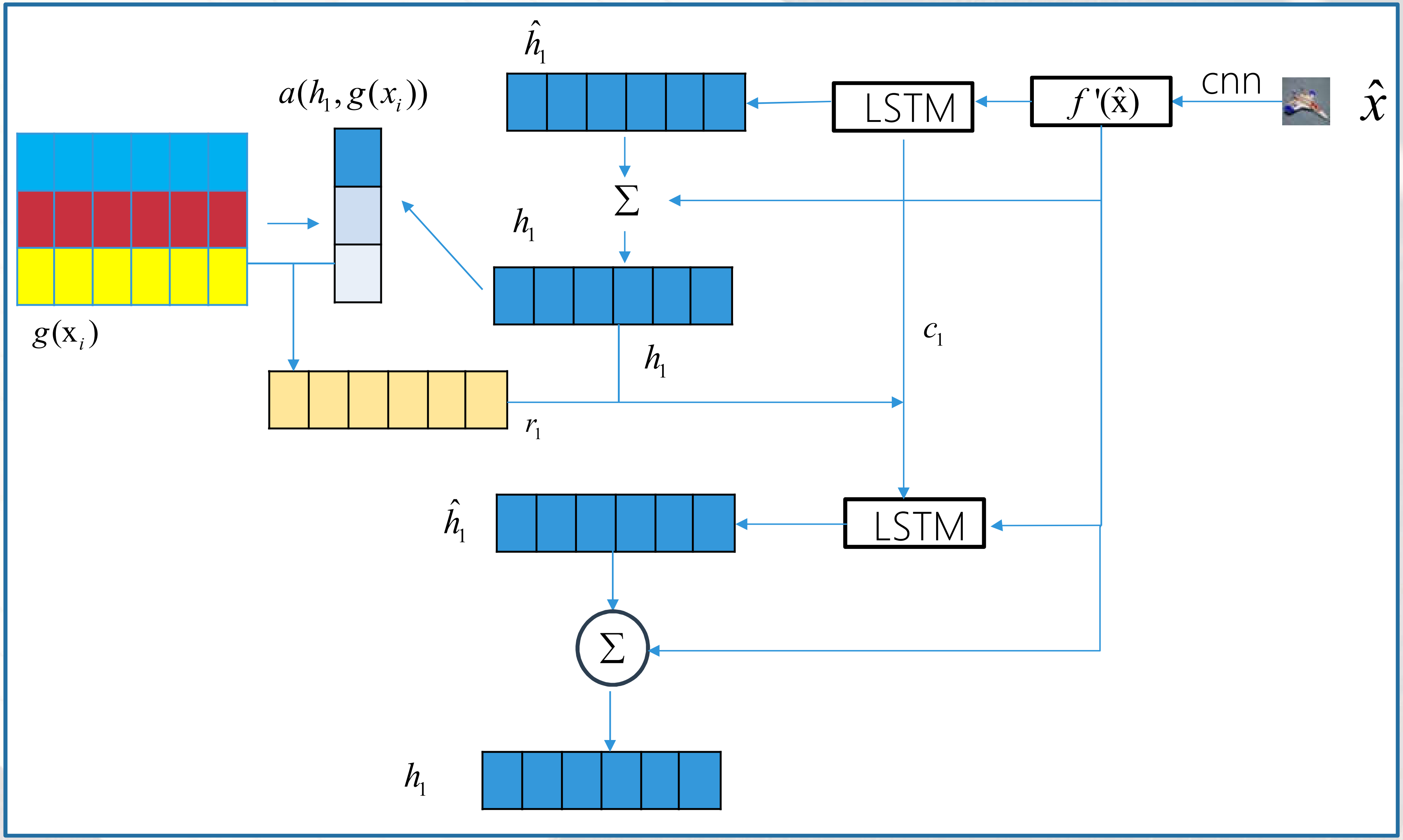


$$\hat{h}_1, c_1 = LSTM(f'(\hat{x}), [h_0, r_0], c_0)$$

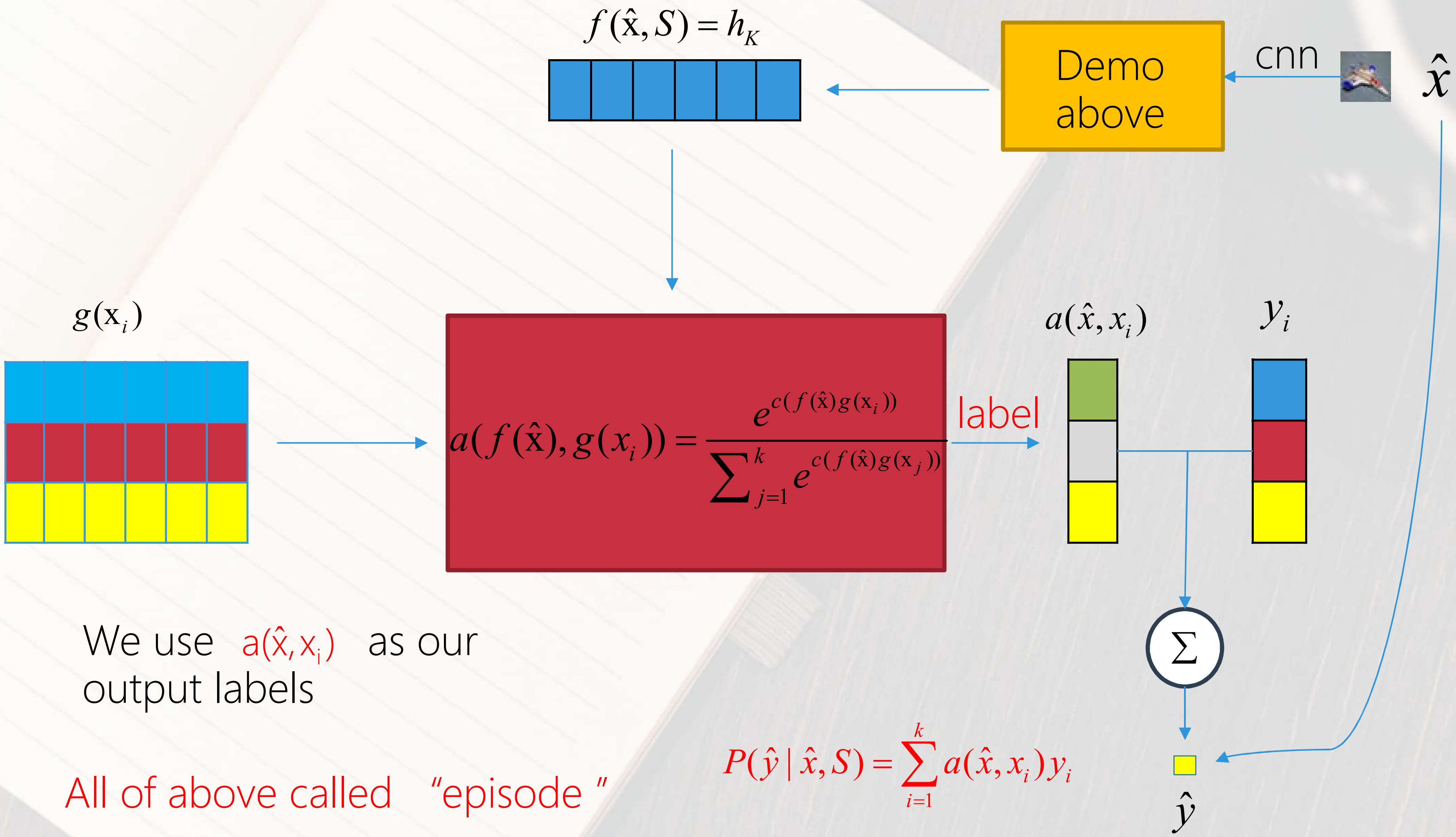
$$h_1 = \hat{h}_1 + f'(\hat{x})$$

$$\rightarrow a(h_1, g(x_i)) \rightarrow r_1 = \sum_{i=1}^{|S|} a(h_1, g(x_i)) g(x_i)$$

Matching Networks



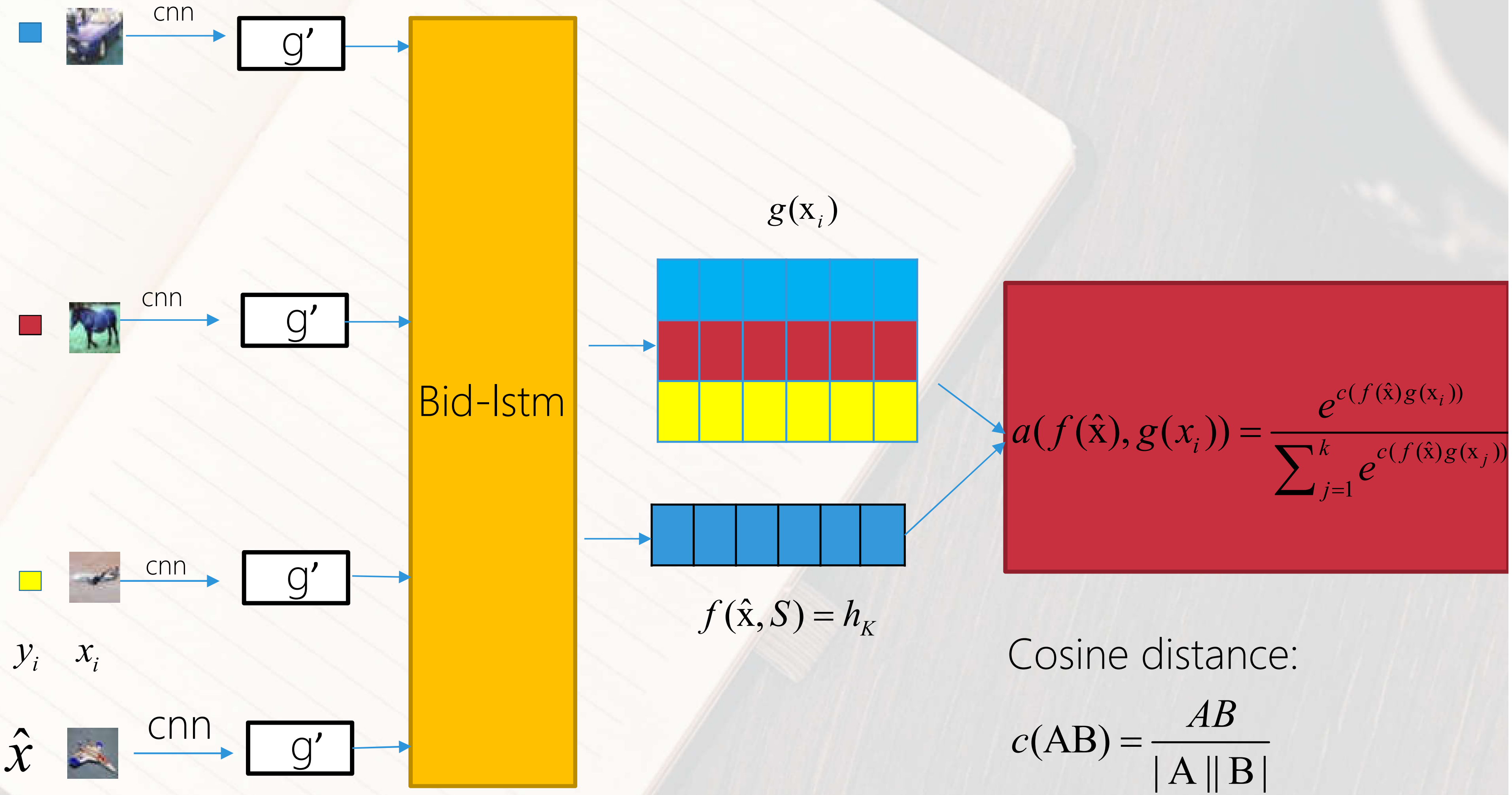
Matching Networks



We use $a(\hat{x}, x_i)$ as our output labels

All of above called "episode"

Matching Networks without lstm



Cosine distance:

$$c(\mathrm{AB}) = \frac{AB}{|\mathrm{A}||\mathrm{B}|}$$

Backpropagation

To form an "episode" to compute gradients and update our model:

L :Data set
S :Support set (sample from L)
B :Bath (sample from L)
x̂(image), ŷ(label) is one input of model

θ is the weight

$$S \sim L, B \sim L$$

$$\hat{x}, \hat{y} \sim B$$

Conditional Probability:

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

$$\theta = \arg\max_{\theta} E_{L \sim T}[E_{S \sim L, B \sim L}[\sum_{(\mathrm{x},\mathrm{y}) \in \mathrm{B}} \log P_{\theta}(y \mid x, S)]]$$

We learning the mapping function P of model!

Backpropagation

$$\theta = \arg\max_\theta \sum_{(x,y)\in B} \log P_\theta(y \mid x, S) = -\arg\min Costfunction$$

$$Cost = \sum_{(x,y)\in B} \log P(\hat{y} \mid \hat{x}, S)$$

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

For every $\hat{x}, \hat{y} \sim B$

$$E_{\hat{x},\hat{y}} = \log P(\hat{y} \mid \hat{x}, S)$$

$$\frac{\partial E}{\partial W}\Big|_{\theta=-W} = \frac{1}{P}\frac{\partial P}{\partial W} = \frac{1}{P}\frac{\partial}{\partial W}(a(\hat{x}, x_1) y_1 + \cdots a(\hat{x}, x_k) y_k)$$
$$= \frac{1}{P}\left(\frac{\partial a(\hat{x}, x_1)}{\partial W_1} y_1 + \cdots \frac{\partial a(\hat{x}, x_k)}{\partial W_k} y_k\right)$$

Forward propagation

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

Backpropagation

$$\frac{\partial a(f(x), g(x_i))}{\partial W_i} = \frac{\partial \dfrac{e^{f(\hat{x})g(x_i)}}{\sum_{j=1}^{k} e^{f(\hat{x})g(x_j)}}}{\partial W_i}$$

$$= \frac{(e^{f(\hat{x})g(x_i)})' \sum_{j=1}^{k} e^{f(\hat{x})g(x_j)} - e^{f(\hat{x})g(x_i)} (\sum_{j=1}^{k} e^{f(\hat{x})g(x_j)})'}{(\sum_{j=1}^{k} e^{f(\hat{x})g(x_j)})^2}$$

$$\frac{\partial e^{f(\hat{x})g(x_i)}}{\partial W} = e^{f(\hat{x})g(x_i)} \frac{\partial (f(\hat{x})g(x_i))}{\partial W}$$

$$= e^{f(\hat{x})g(x_i)} \left(\frac{\partial (f(\hat{x}))}{\partial W} g(x_i) + \frac{\partial (g(x_i))}{\partial W} f(\hat{x})\right)$$

Forward propagation

$$h_k = LSTM(f'(\hat{x}), [h_{k-1}, \mathrm{r}_{k-1}], \mathrm{c}_{k-1}) + f'(\hat{x})$$

Backpropagation

$$\frac{\partial h_k}{\partial W} = \frac{\partial \hat{h}_k}{\partial W} + \frac{\partial f'(\hat{x})}{\partial W}$$

$$\frac{\partial f'(\hat{\mathrm{x}})}{\partial W} \doteq \frac{\partial g'(x)}{\partial W}$$

Backpropagation

Forward propagation

$$\hat{h}_k, c_k = LSTM(f'(\hat{x}),[h_{k-1}, r_{k-1}], c_{k-1})$$

$$i_k = \sigma(W_{xi} f' + W_{hi} h_{k-1} + W_{ri} r_{k-1} + b_i)$$

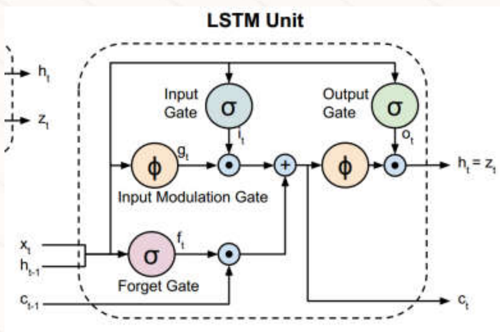
$$f_k = \sigma(W_{xf} f' + W_{hf} h_{k-1} + W_{rf} r_{k-1} + b_f)$$

$$o_k = \sigma(W_{xo} f' + W_{ho} h_{k-1} + W_{ri} r_{k-1} + b_o)$$

$$g_k = \tanh(W_{xc} f' + W_{hc} h_{k-1} + W_{ri} r_{k-1} + b_c)$$

$$c_k = f_k \odot c_{k-1} + i_k \odot g_k$$

$$h_k = o_k \odot \tanh(c_k)$$



Backpropagation

Layers t:

$$\delta h^t = \frac{\delta LSTM}{\delta W}$$

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

$$\delta c^t = \delta h^t \odot o^t \odot (1-\tanh^2(c^t)) + \delta c^{t+1} \odot f^{t+1}$$

$$\delta i^t = \delta c^t \odot g^t \qquad \delta f^t = \delta c^t \odot c^{t-1}$$

$$\delta g^t = \delta c^t \odot i^t \qquad \delta c^{t-1} = \delta c^t \odot f^t$$

$$\delta \hat{i}^t = \delta i^t \odot i^t \odot (1-i^t)$$

$$\delta \hat{f}^t = \delta f^t \odot f^t \odot (1-f^t)$$

$$\delta \hat{g}^t = \delta g^t \odot (1-\tanh^2(\hat{g}_t))$$

Forward propagation

$$\hat{h}_k, c_k = LSTM(f'(\hat{x}), [h_{k-1}, r_{k-1}], c_{k-1})$$

$$\hat{i}_k = W_{xi} f' + W_{hi} h_{k-1} + W_{ri} r_{k-1} + b_i$$

$$\hat{f}_k = W_{xf} f' + W_{hf} h_{k-1} + W_{rf} r_{k-1} + b_f$$

$$\hat{o}_k = W_{xo} f' + W_{ho} h_{k-1} + W_{ri} r_{k-1} + b_o$$

$$\hat{g}_k = W_{xc} f' + W_{hc} h_{k-1} + W_{ri} r_{k-1} + b_c$$

Backpropagation

Layers t:

$$\delta\hat{o}^t = \delta o^t \odot o^t \odot (1 - o^t)$$

$$\tanh'(\hat{g}^t) = 1 - \tanh^2(\hat{g}^t)$$

$$\delta z^t = [\delta\hat{g}^t, \delta\hat{i}^t, \delta\hat{f}^t, \delta\hat{o}^t]$$

for simple:we can make:

$$z^t = \begin{bmatrix} \hat{g}^t \\ \hat{i}^t \\ \hat{f}^t \\ \hat{o}^t \end{bmatrix} = \begin{bmatrix} W_{gx} & W_{gh} & W_{gr} \\ W_{ix} & W_{ih} & W_{ir} \\ W_{fx} & W_{fh} & W_{fr} \\ W_{ox} & W_{oh} & W_{oi} \end{bmatrix} \begin{bmatrix} x_t \\ h_{t-1} \\ r_{t-1} \end{bmatrix} + \begin{bmatrix} b_g \\ b_i \\ b_f \\ b_o \end{bmatrix}$$

Backpropagation

Forward propagation

$$\hat{h}_k, \mathrm{c}_k = LSTM(f'(\hat{x}), [h_{k-1}, \mathrm{r}_{k-1}], \mathrm{c}_{k-1})$$

$$\hat{i}_k = W_{xi} f' + W_{hi} h_{k-1} + W_{ri} r_{k-1} + b_i$$

$$\hat{f}_k = W_{xf} f' + W_{hf} h_{k-1} + W_{rf} r_{k-1} + b_f$$

$$\hat{o}_k = W_{xo} f' + W_{ho} h_{k-1} + W_{ri} r_{k-1} + b_o$$

$$\hat{g}_k = W_{xc} f' + W_{hc} h_{k-1} + W_{ri} r_{k-1} + b_c$$

Backpropagation

Layers t:

$$z^t = \begin{bmatrix} \hat{g}^t \\ \hat{i}^t \\ \hat{f}^t \\ \hat{o}^t \end{bmatrix} = \begin{bmatrix} W_{gx} & W_{gh} & W_{gr} \\ W_{ix} & W_{ih} & W_{ir} \\ W_{fx} & W_{fh} & W_{fr} \\ W_{ox} & W_{oh} & W_{oi} \end{bmatrix} \begin{bmatrix} x_t \\ h_{t-1} \\ r_{t-1} \end{bmatrix} + \begin{bmatrix} b_g \\ b_i \\ b_f \\ b_o \end{bmatrix}$$

$$z^t = W \bullet I^t + B$$

$$\delta W^t = \delta z^t \cdot (I^t)^T \qquad \delta I^t = \delta z^t \cdot W^t = \begin{bmatrix} \delta x^t \\ \delta h^{t-1} \\ \delta r^{t-1} \end{bmatrix}$$

$$\delta B^t = \delta z^t = \begin{bmatrix} \delta b_g^t \\ \delta b_i^t \\ \delta b_f^t \\ \delta b_o^t \end{bmatrix}$$

Forward propagation

$$\hat{h}_k, \mathrm{c}_k = LSTM(f'(\hat{x}), [h_{k-1}, \mathrm{r}_{k-1}], \mathrm{c}_{k-1})$$

$$\hat{i}_k = W_{xi} f' + W_{hi} h_{k-1} + W_{ri} r_{k-1} + b_i$$

$$\hat{f}_k = W_{xf} f' + W_{hf} h_{k-1} + W_{rf} r_{k-1} + b_f$$

$$\hat{o}_k = W_{xo} f' + W_{ho} h_{k-1} + W_{ri} r_{k-1} + b_o$$

$$\hat{g}_k = W_{xc} f' + W_{hc} h_{k-1} + W_{ri} r_{k-1} + b_c$$

Backpropagation

Layers t:

$$z^t = W \bullet I^t + B$$

$$\delta W_{gx} = \delta \hat{g}^t \cdot x^t \qquad \delta W_{ix} = \delta \hat{i}^t \cdot x^t$$

$$\delta W_{gh} = \delta \hat{g}^t \cdot h^{t-1} \qquad \delta W_{ih} = \delta \hat{i}^t \cdot h^{t-1}$$

$$\delta W_{gr} = \delta \hat{g}^t \cdot r^{t-1} \qquad \delta W_{ir} = \delta \hat{i}^t \cdot r^{t-1}$$

$$\delta W_{fx} = \delta \hat{f}^t \cdot x^t \qquad \delta W_{ox} = \delta \hat{o}^t \cdot x^t$$

$$\delta W_{fh} = \delta \hat{f}^t \cdot h^{t-1} \qquad \delta W_{oh} = \delta \hat{o}^t \cdot h^{t-1}$$

$$\delta W_{fr} = \delta \hat{f}^t \cdot r^{t-1} \qquad \delta W_{or} = \delta \hat{o}^t \cdot r^{t-1}$$

Forward propagation

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

$$\vec{h}_i, \vec{\mathrm{c}}_i = LSTM(\mathrm{g}'(x_i), \vec{h}_{i-1}, \vec{\mathrm{c}}_{i-1})$$

$$\overleftarrow{h}_i, \overleftarrow{\mathrm{c}}_i = LSTM(\mathrm{g}'(x_i), \overleftarrow{h}_{i+1}, \overleftarrow{\mathrm{c}}_{i+1})$$

Backpropagation

$$\delta g = \delta \vec{h}_i + \delta \overleftarrow{h}_i + \delta g'$$

$$\delta \vec{h}_i \doteq \delta \overleftarrow{h}_i = \delta(\mathrm{LSTM})$$

The same as above

Forward propagation

cnn demo

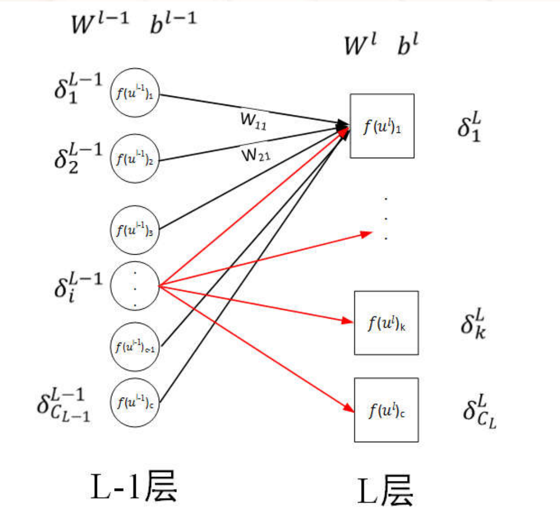
Backpropagation

Input:g'

$$\delta_k = \delta g'$$

Forward propagation

cnn demo



$$u^l = W^l f(u^{l-1}) + b^l$$

Backpropagation

$$f'(u^l) = \left(\frac{1}{1+e^{-u^l}}\right)'$$
$$= \frac{1}{1+e^{-u^l}}\left(1 - \frac{1}{1+e^{-u^l}}\right)$$
$$= f(u^l)(1 - f(u^l))$$

$$\delta u^l = \delta_k^l$$

Forward propagation

cnn demo

Backpropagation

$$\delta_i^{l-1} = \sum_{k=1}^{c_l} (\delta_k^{l-1}) = \sum_{k=1}^{c_l} \frac{\partial u_i^l}{\partial u_i^{l-1}}$$

$$= \frac{\partial}{\partial u_i^{l-1}} \sum_{k=1}^{c_l} (Wf + b)$$

$$= \sum_{k=1}^{c_l} \delta_k^l \, W_{ik}^l f'(u_i^{l-1})$$

$$\delta^{l-1} = (W^l)^T \delta^l .* f'(u^{l-1})$$

Forward propagation

cnn demo

Backpropagation

$$b = b - \alpha \frac{\partial}{\partial b} J(W, b) = b - \alpha * \delta^l$$

$$W = W - \alpha \frac{\partial}{\partial W} J(W, b)$$

$$\frac{\partial u}{\partial W_{ik}} = \delta_k^l * f(u_i^{l-1})$$

$$W = W - \alpha * \delta_k^l * f(u_i^{l-1})$$

- Background
- Introduction
  - Introduce to One Shot Learning
  - Introduce to MANN
- Model
  - Motivation
  - Matching Networks
  - Backpropagation
- **Experiments**
- Summary

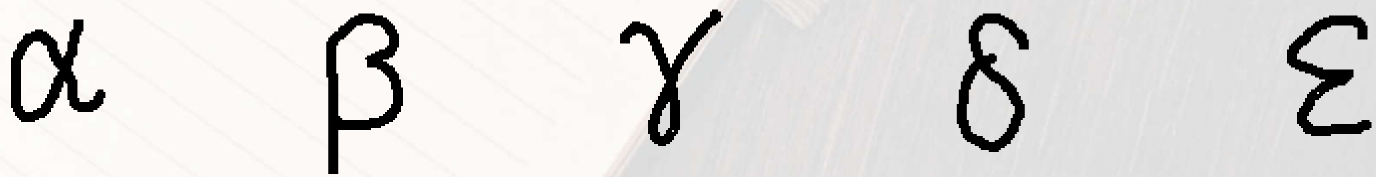
Experiments

- Data: Omniglot

Omniglot consists of 1623 characters from 50 different alphabets. Each of these was hand drawn by 20 different people. The large number of classes (characters) with relatively few data per class(20), makes this an ideal data set for testing small-scale one-shot classification.

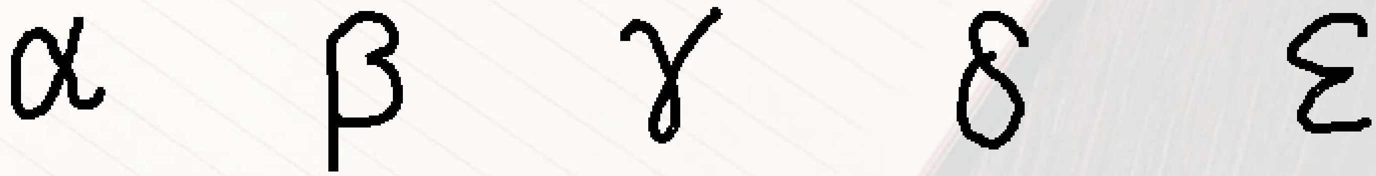
Download:https://github.com/brendenlake/omniglot

Example:



- 5-way, 5-shot learning

Traning data set:

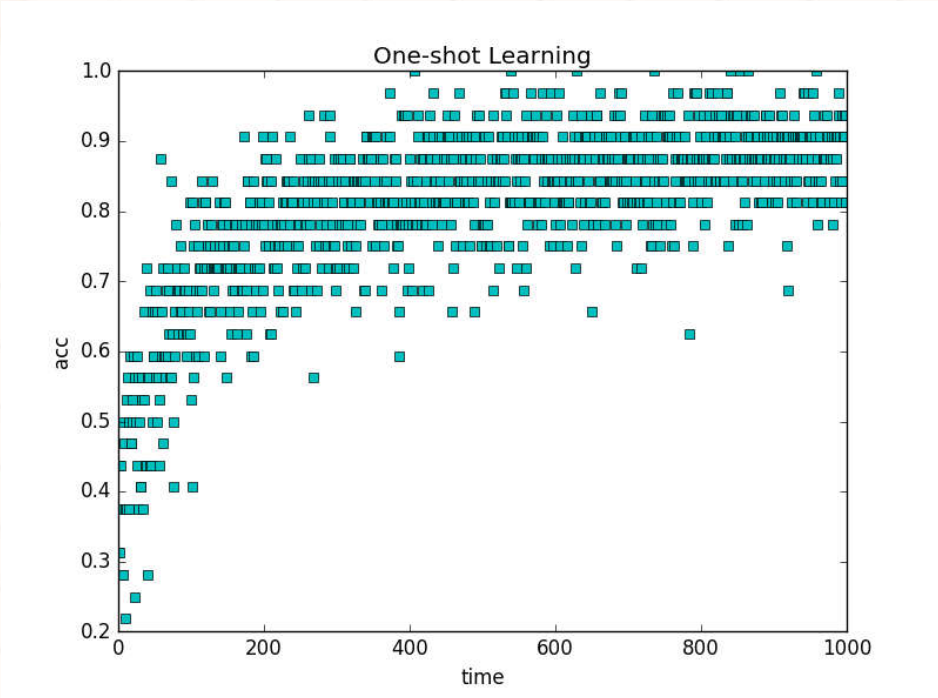


Test data set:

Experiments

- Results

Test the relationship between accurancy and the number of bath.



Simulation environment:

- Python 3.5
- Tensorflow-1.0

Traning Input:

- Five classes image of Omniglot

Output:

- Accurancy of test bath

Time:total_train_batches

Experiments

- Results

Test the accurancy of one-shot model.

Simulation environment:

- Python 3.5
- Tensorflow-1.0

Test Input:

- Another five classes image of Omniglot

Output:

- Accurancy of test bath

Summary

- One-shot learning learns the mapping function between input image and memory
- One-shot learning search the memory information for traning
- Matching Network has non-parametric structure, thus has ability to acquisition of new examples rapidly