Matching Networks for One Shot Learning NIPS 2016

Learning Algorithms for Active Learning ICML 2017

2018.11.26

Matching Networks

Map from a (small) support set of k examples of input-label pairs $S = \{(x_i, y_i)\}_{i=1}^{K}$ to a classifier $C_S(\hat{x})$.

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

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

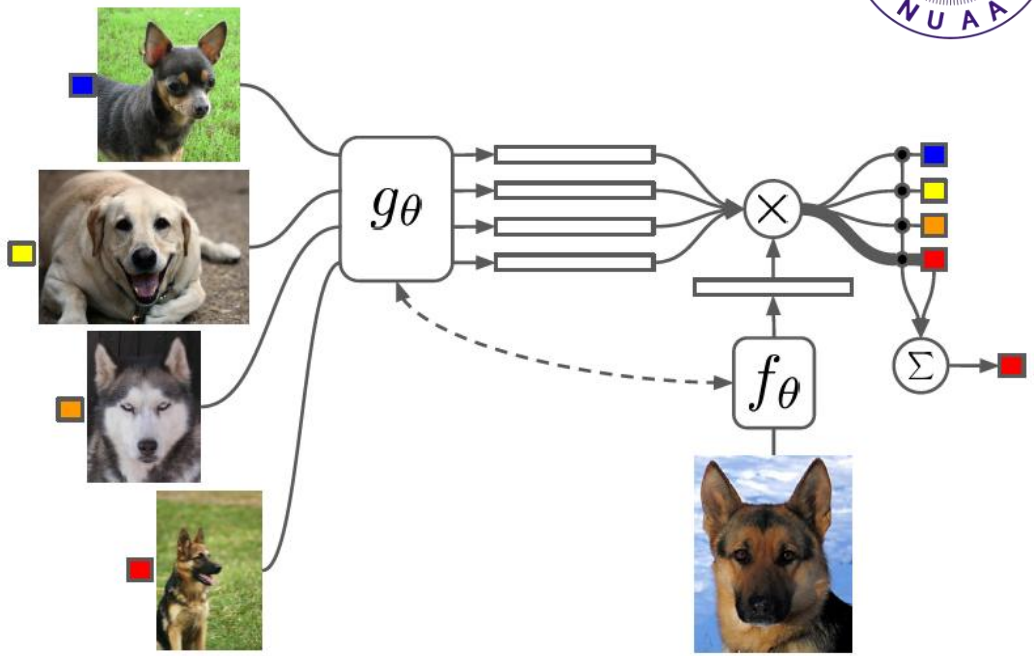
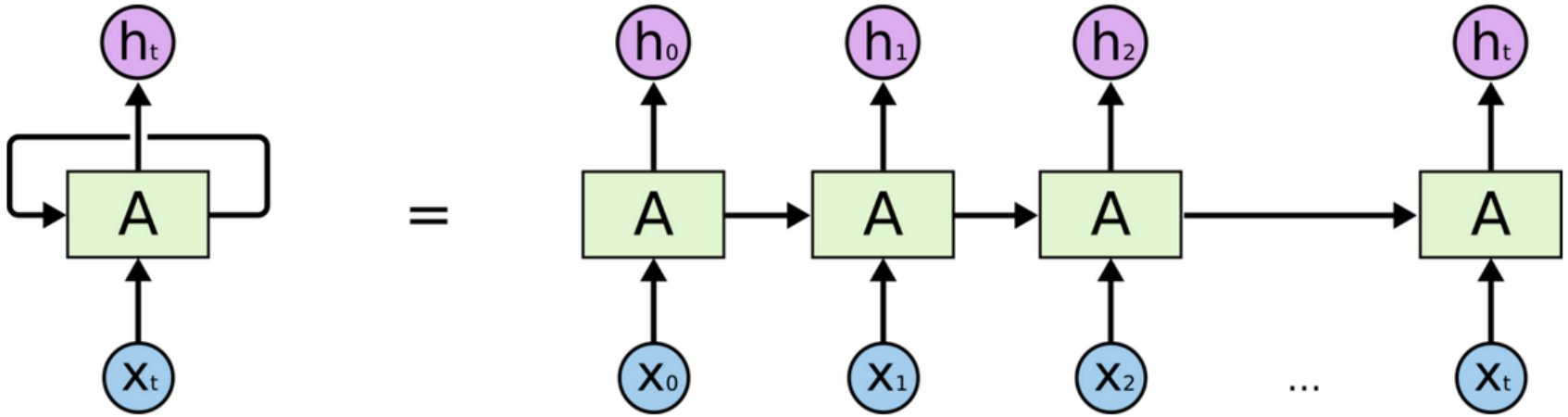
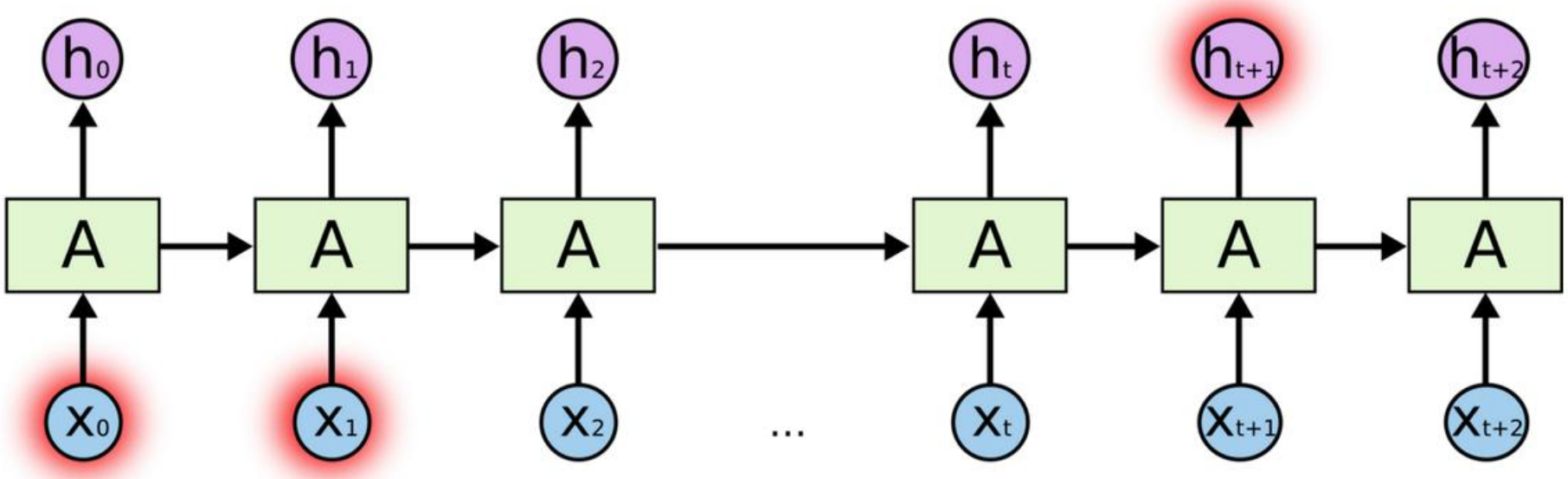
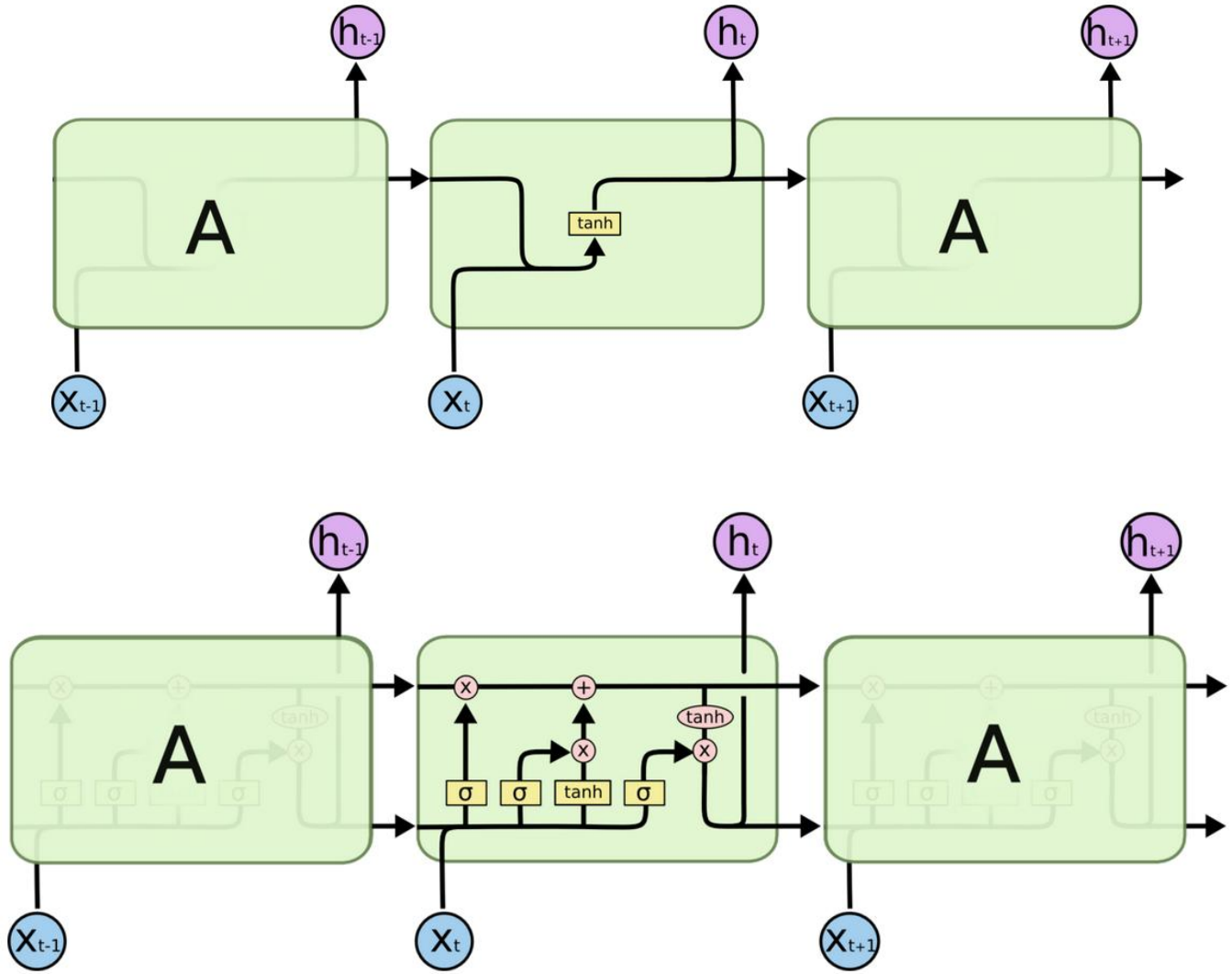


Figure 1: Matching Networks architecture

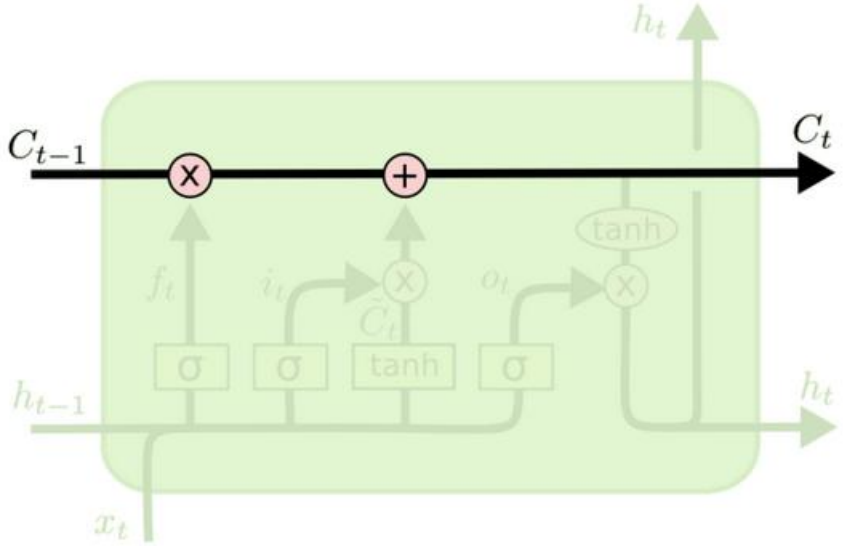
Embedding functions f and g are appropriate neural networks to embed $\hat{x}$ and x_i


h_t
A
x_t
=
h_0
h_1
h_2
h_t
A
A
A
A
x_0
x_1
x_2
...
x_t

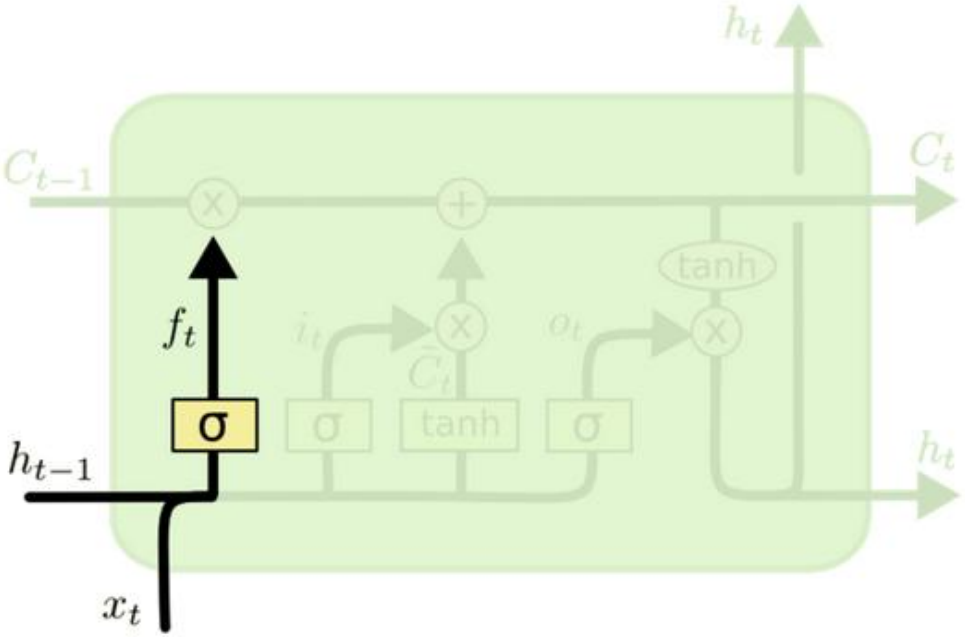

h_0
h_1
h_2
h_t
h_{t+1}
h_{t+2}
A
A
A
A
A
A
x_0
x_1
x_2
...
x_t
x_{t+1}
x_{t+2}


h_{t-1}
h_t
h_{t+1}
A
tanh
A
x_{t-1}
x_t
x_{t+1}
h_{t-1}
h_t
h_{t+1}
A
σ
σ
tanh
σ
tanh
A
x_{t-1}
x_t
x_{t+1}
Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

state

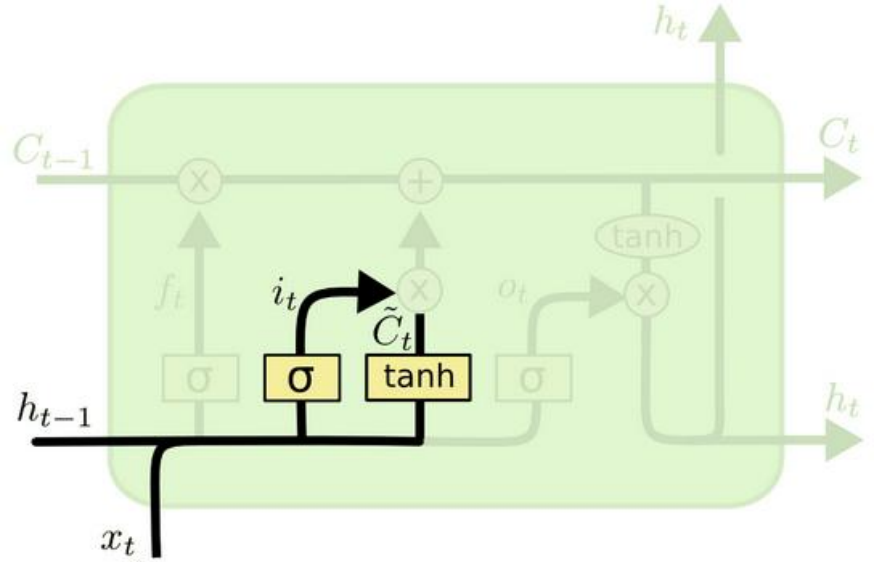


遗忘门



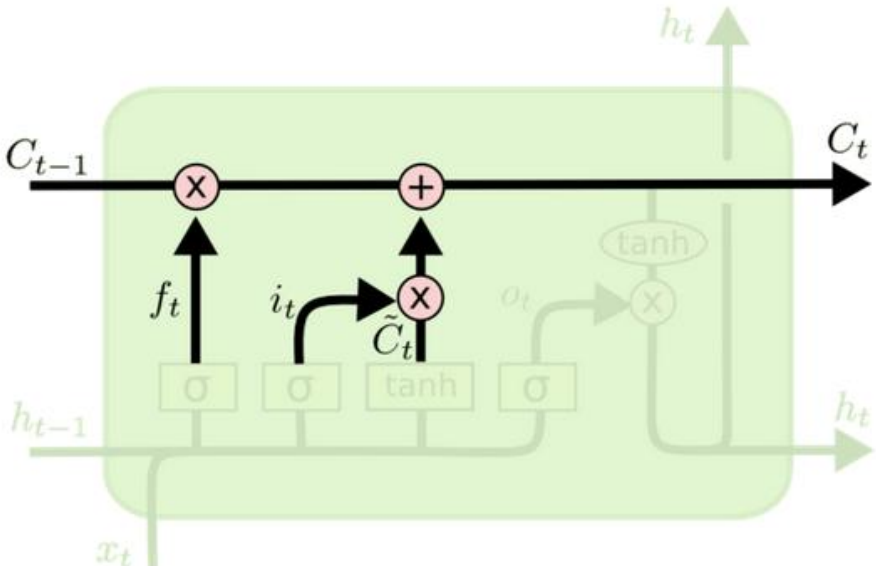
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

输入门



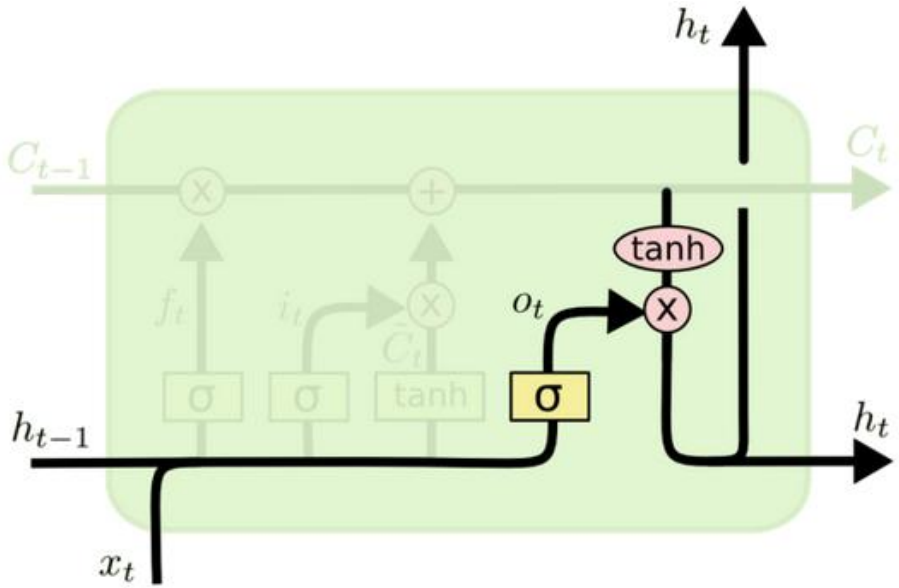
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

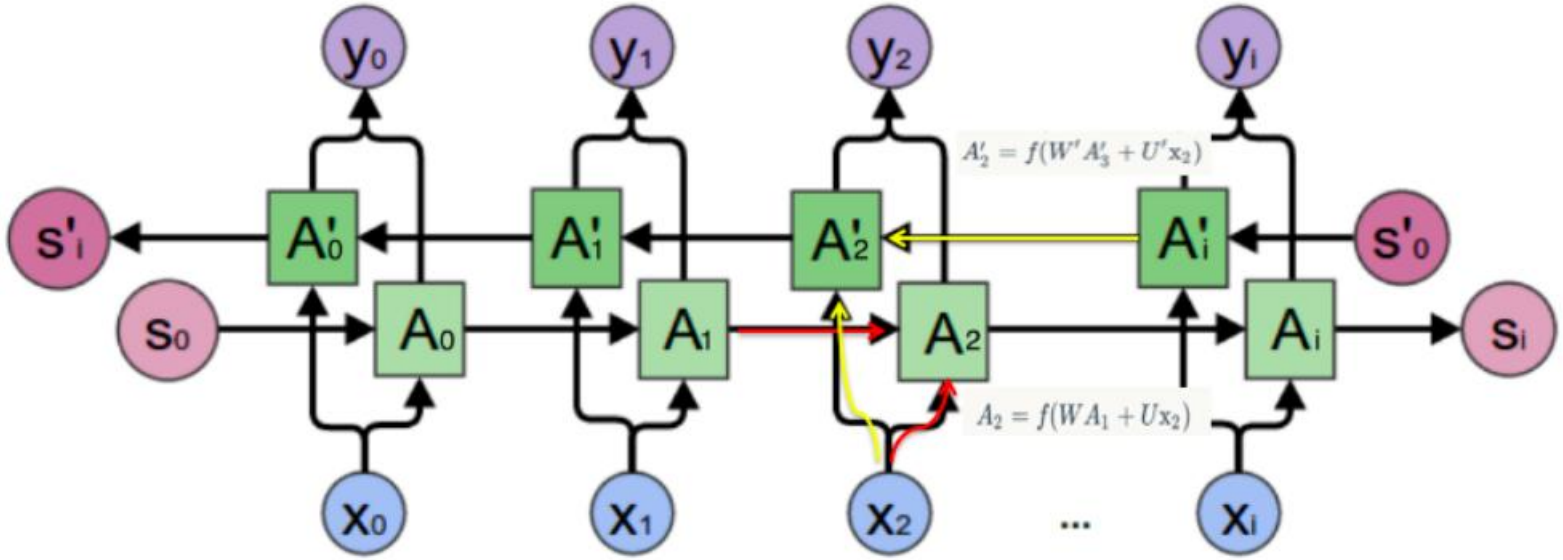
输出门

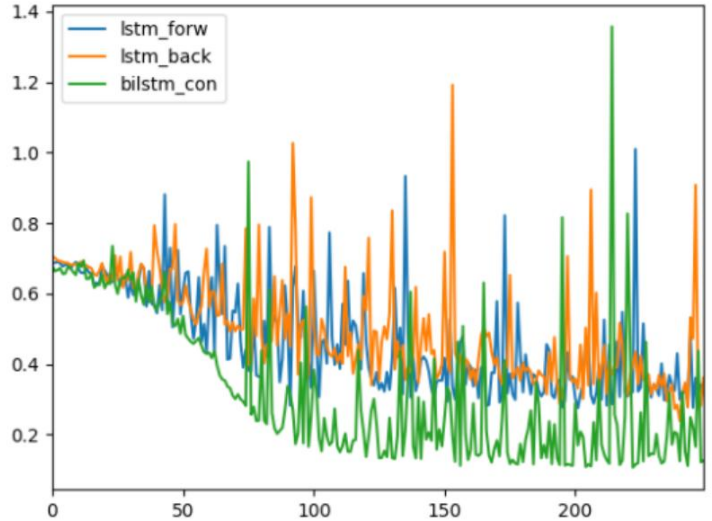


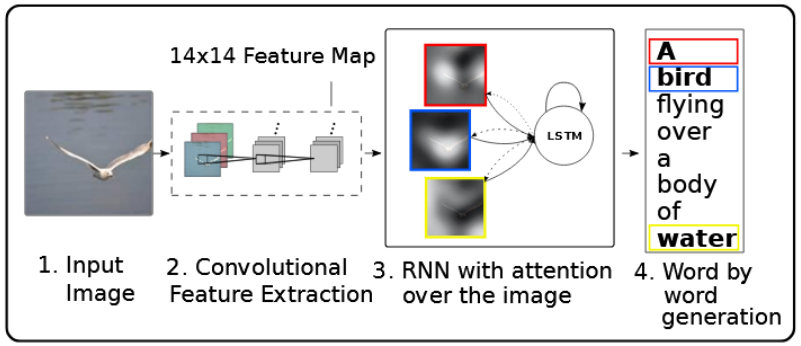
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

双向LSTM







A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

differentiable nearest neighbor++

Closely related to metric learning, the embedding functions **f** and **g** act as a lift to feature space X to achieve maximum accuracy through the classification function described in $P(\hat{y}|\hat{x}, S) = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i$.

Embedding the elements of the set through a function which takes as input the full set S in addition to x_i, i.e. g becomes $g(x_i, S)$. Thus, as a function of the whole support set S, g can modify how to embed x_i.

Embedding the training examples: **g** is a bidirectional LSTM over the examples, more precisely, let $g'(x_i)$ be a neural network(e.g. a VGG or inception model).

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

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

Embedding the test example

f is a an LSTM that processes for a fixed amount (K time steps) and at each point also attends over the examples in the training set. The encoding is **the last hidden state** of the LSTM.

$$\hat{h}_k, c_k = \text{LSTM}(f'(\hat{x}), [h_{k-1}, r_{k-1}], c_{k-1}) \quad (3)$$

$$h_k = \hat{h}_k + f'(\hat{x}) \quad (4)$$

$$r_{k-1} = \sum_{i=1}^{|S|} a(h_{k-1}, g(x_i)) g(x_i) \quad (5)$$

$$a(h_{k-1}, g(x_i)) = \text{softmax}(h_{k-1}^T g(x_i)) \quad (6)$$

Define a task T as distribution over possible label sets L

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

Conclusion

one-shot learning is much easier if you train the network to do one-shot learning.

non-parametric structures in a neural network make it easier for networks to remember and adapt to new training sets in the same tasks.

An obvious drawback of our model is the fact that, as the support set S grows in size, the computation for each gradient update becomes more expensive

When the label distribution has obvious biases (such as being fine grained), our model suffers.

Learning Algorithms for Active Learning

Task Description

support set $S \equiv \{(x, y)\}$ evaluation set $E = \{(\hat{x}, \hat{y})\}$

$S_t^u = \{(x, \cdot)\}$ denote the set of items in the support set whose labels are still unknown after t label queries

$S_t^k = \{(x, y)\}$ denote the items whose labels are known.

S_t denote the joint set of labeled and unlabeled items after t label queries.

s_t denote the control state of our model after viewing t labels.

$R(E, S_t, s_t)$ denote the reward won by the model when predicting labels for the evaluation set.

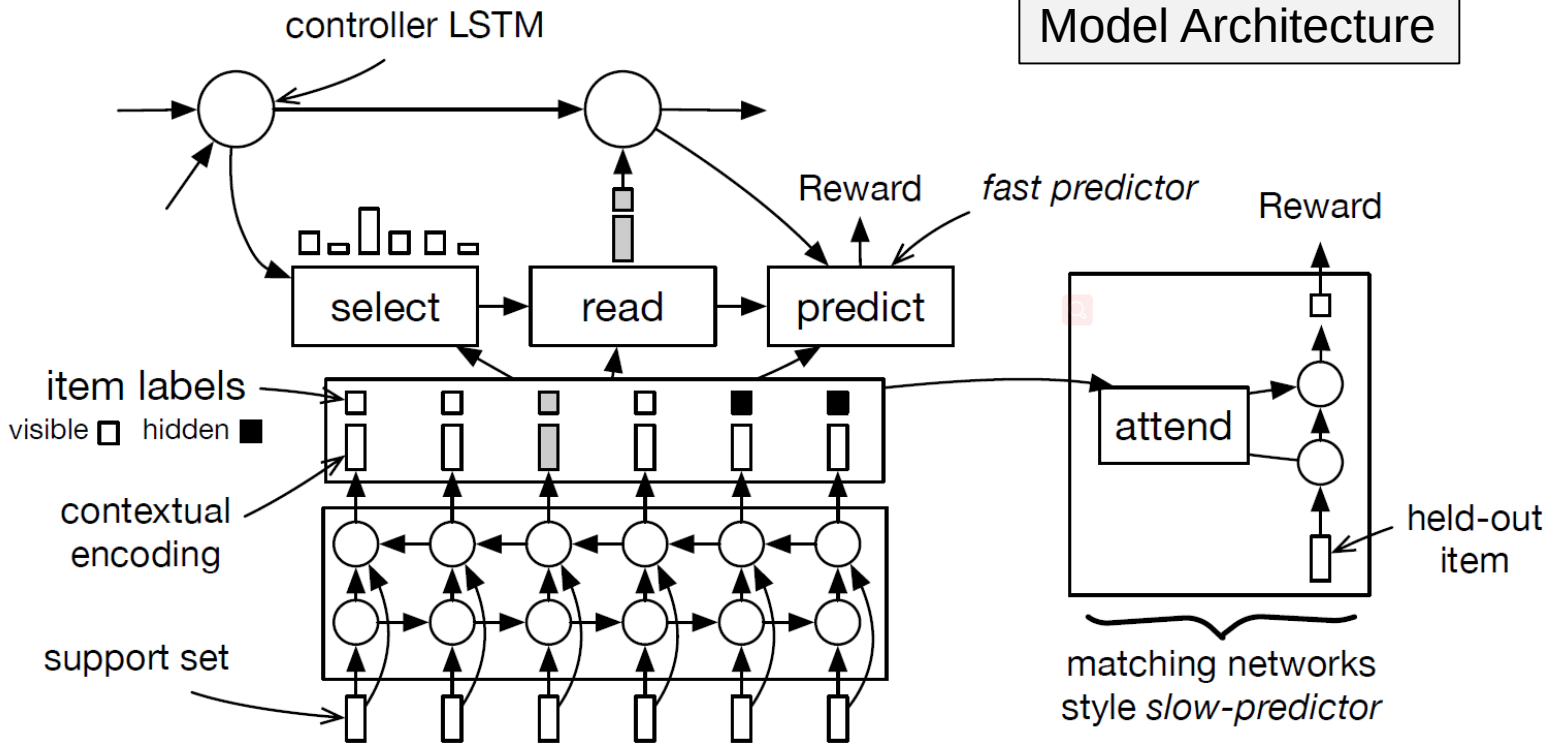
$R(E, S_t, s_t) \equiv \sum_{(\hat{x},\hat{y})\in E} \log p(\hat{y}|\hat{x}, s_t, S_t)$ gives log-likelihood of the predictions $p(\hat{y}|\hat{x}, s_t, S_t)$ on the evaluation set.

$$\underset{\theta}{\text{maximize}} \underset{(S,E)\sim\mathcal{D}}{\mathbb{E}} \left[\underset{\pi(S,T)}{\mathbb{E}} \left[\sum_{t=1}^{T} R(E, S_t, s_t) \right] \right]$$

$\pi(S,T)$ indicates unrolling the model's active learning policy π for T steps on support set S, unrolling π produces the intermediate states $\{(S_1, s_1), \ldots, (S_T, s_T)\}$。

$$\underset{(S,E)\sim\mathcal{D}}{\mathbb{E}} \left[\underset{\pi(S,T)}{\mathbb{E}} \left[\sum_{t=1}^{T} \tilde{R}(S_t^u, S_t, s_t) + R(E, S_T, s_T) \right] \right]$$

Model Architecture



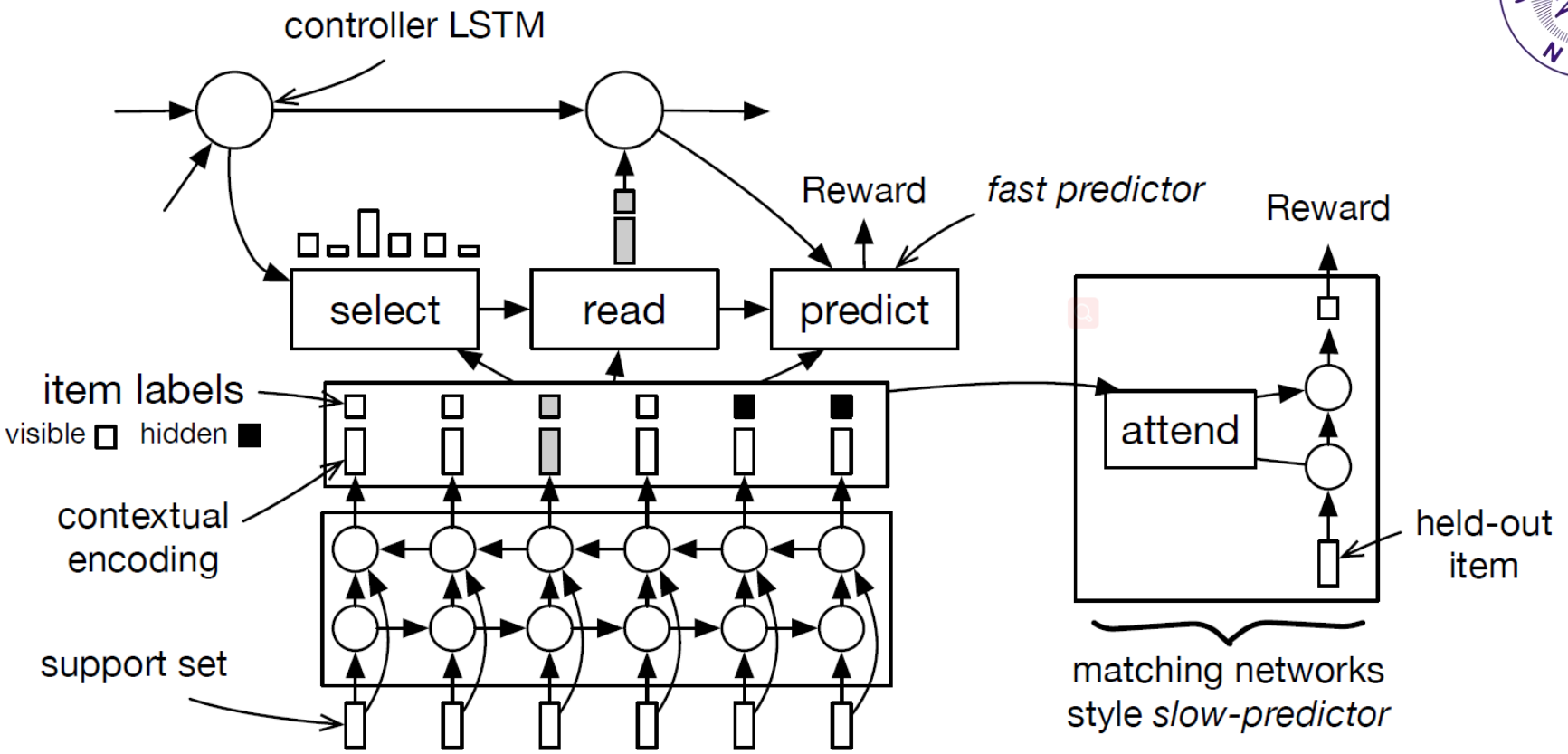
CONTEXT-[FREE|SENSITIVE] ENCODING

Denote the context-free encoding of item $x_i \in S$ as x_i', and similarly define $\hat{x}_i'$ for $\hat{x}_i' \in E$

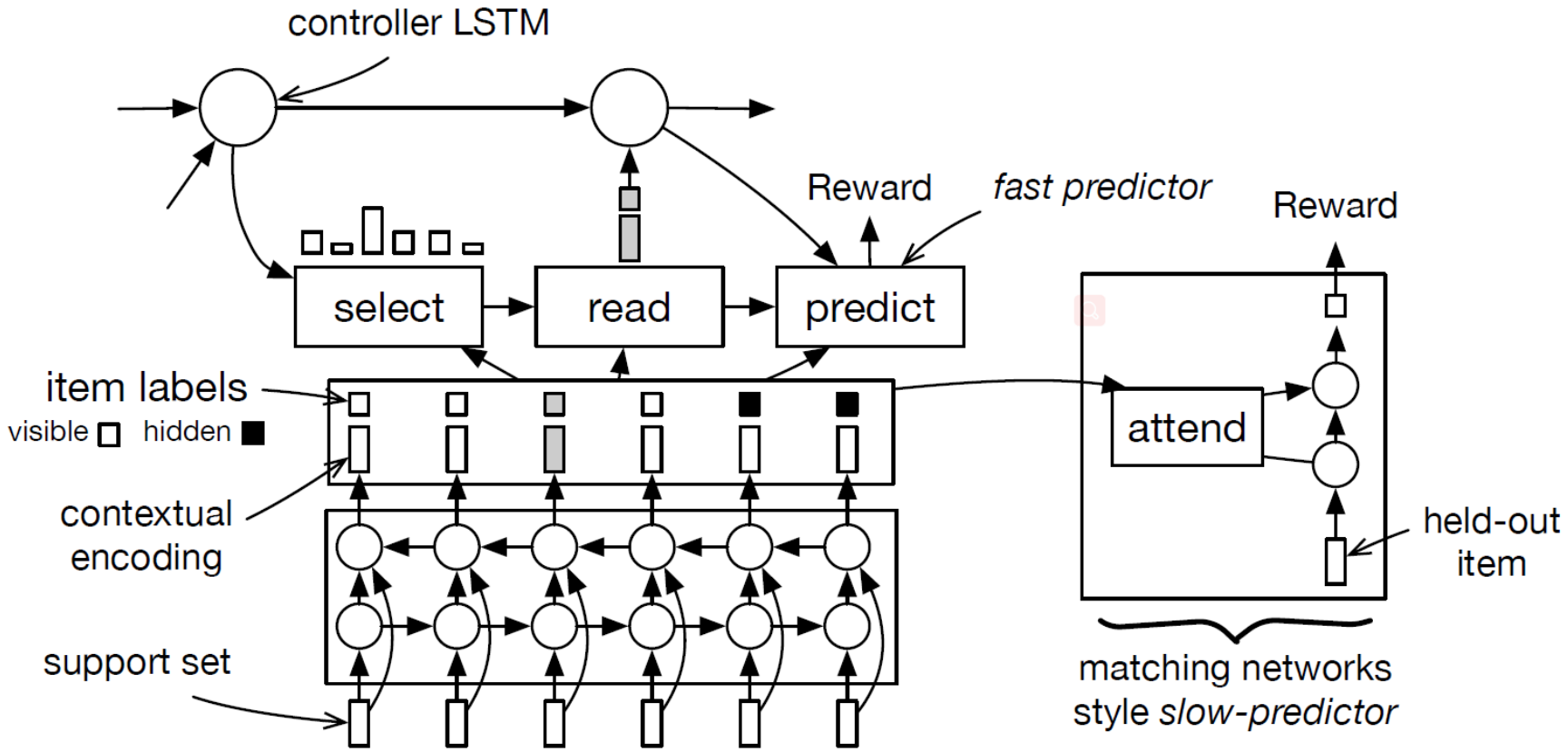
The context-sensitive encoder produces an embedding x_i'' for each item $x_i \in S$ based on the context-free embeddings $x_j' : \forall x_j \in S$..

$$x_i'' = x_i' + W_e[\vec{h}_i; \overleftarrow{h}_i]$$

READING



This module concatenates the embedding x'' and label y_i for the item indicated by the selection module, and **linearly transforms** them before passing them to the controller.

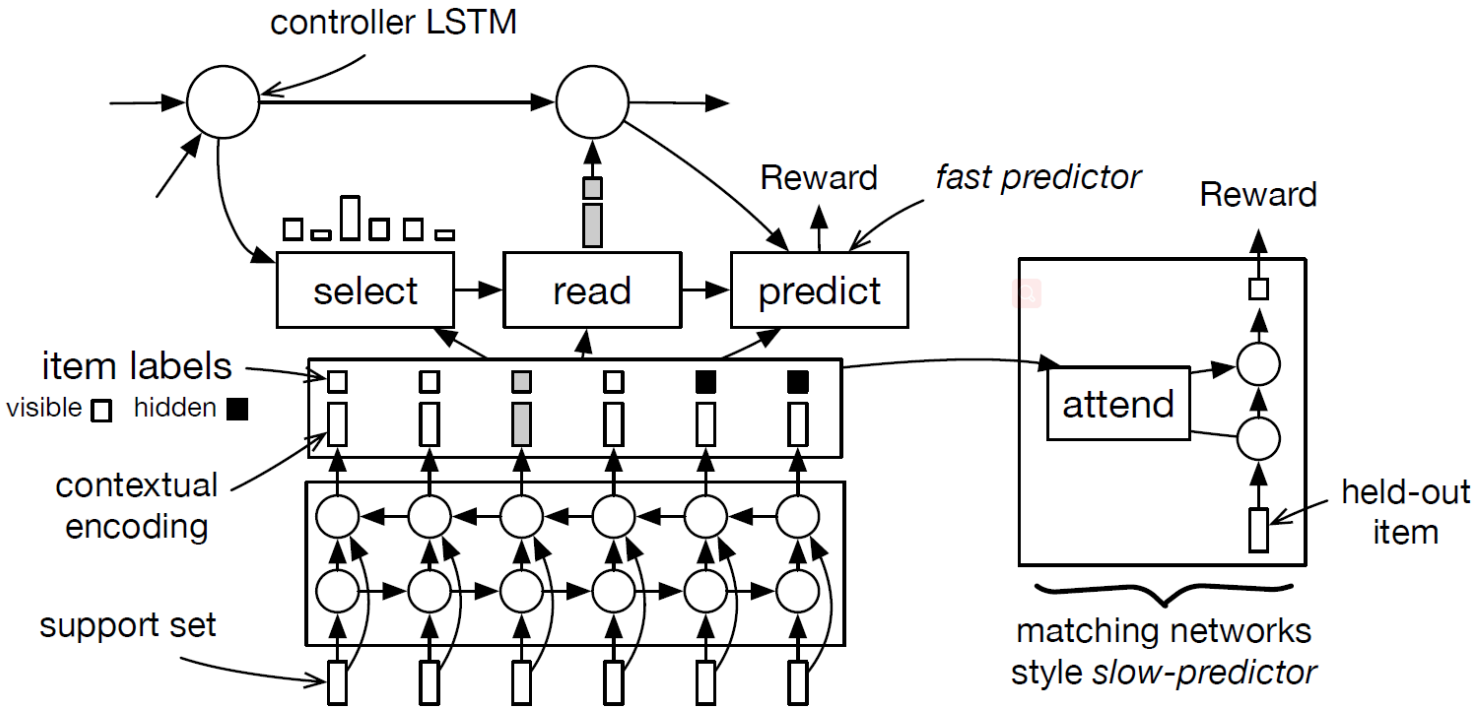


At each step t, the controller receives an input r_t from the reading module which encodes the most recently read item/label pair. Additional inputs could take advantage of task-specific information

$$s_t = \mathrm{LSTM}(s_{t-1}, r_t)$$

SELECTION

At each step t, the selection module places a distribution P_t^u over all unlabeled items $x_i^u \in S_t^u$, It then samples the index of an item to label from P_t^u, and feeds it to the reading module.

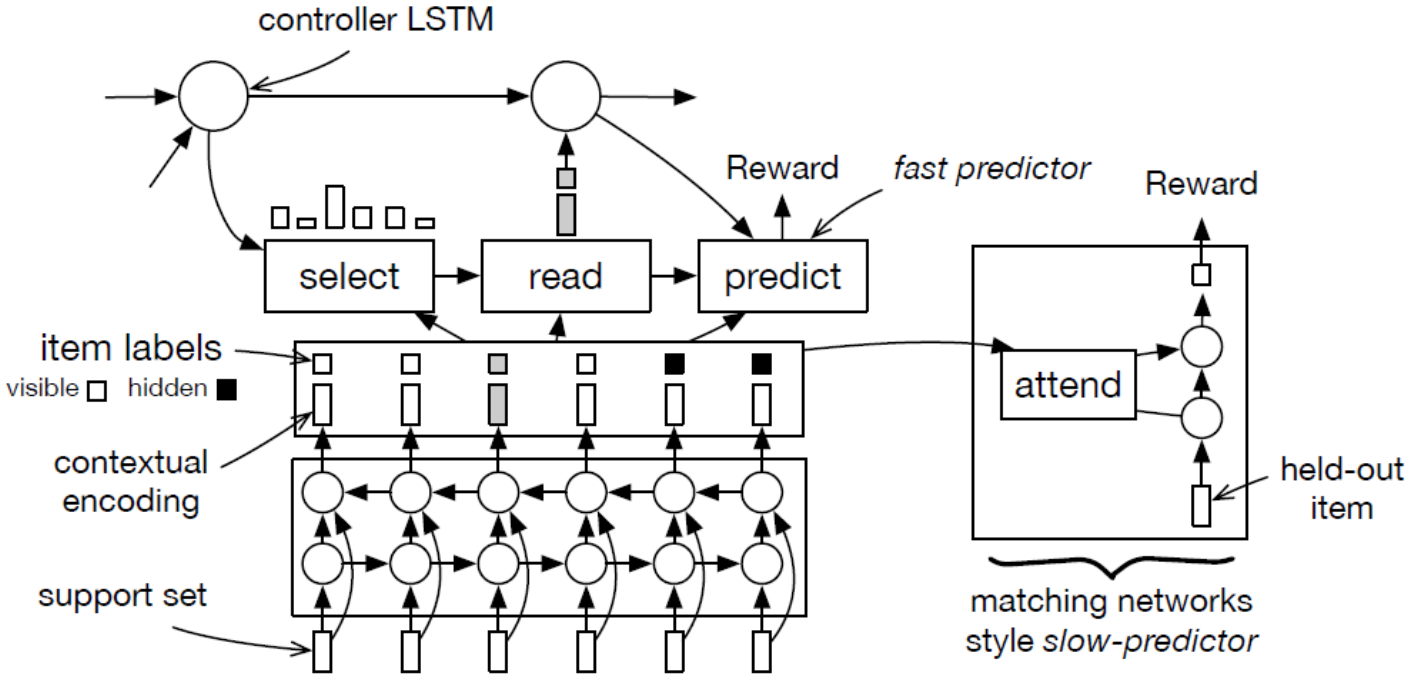


computes P_t^u using a gated, linear combination of features which measure controller-item similarity and item-item similarity

$$b_t^i = x_i'' \odot W_b s_t$$ [max|mean|min] cosine similarity to any (un)labeled item

d_t^i $$g_t = \sigma(W_g s_t)$$ For each $x_i^u \in S_t^u$, compute $p_t^i = (g_t \odot d_t^i)^T w_p$

temperature $P(a) = \frac{e^{\frac{q(a)}{T}}}{\sum_{i=1}^{n} e^{\frac{q(i)}{T}}}$



For each unlabeled x_i^u, compute a set of attention weights over the labeled $x_j^k \in S_t^k$ by applying a softmax to the relevant cosine similarities, using γ_t^i as a temperature for the softmax.

$$\gamma_t^i = \exp((x'')^T W_\gamma s_t)$$

The final fast prediction is formed by taking a convex combination of the labels y_j for the labeled $x_j^k \in S_t^k$ using the computed attention weights.

Training the Model

Optimize the parameters of our model using **a combination of backpropagation and policy gradients.** For a clear review of optimization techniques for general stochastic computation graphs, see Schulman et al. (2016a).

$$\nabla_\theta R(S, E, \theta) = \tag{5}$$
$$\mathop{\mathbb{E}}_{p(\vec{S}|(S,E))} \left(\nabla_\theta \log p(\vec{S}|(S, E)) \left[R(\vec{S}) \right] + \nabla_\theta R(\vec{S}) \right),$$

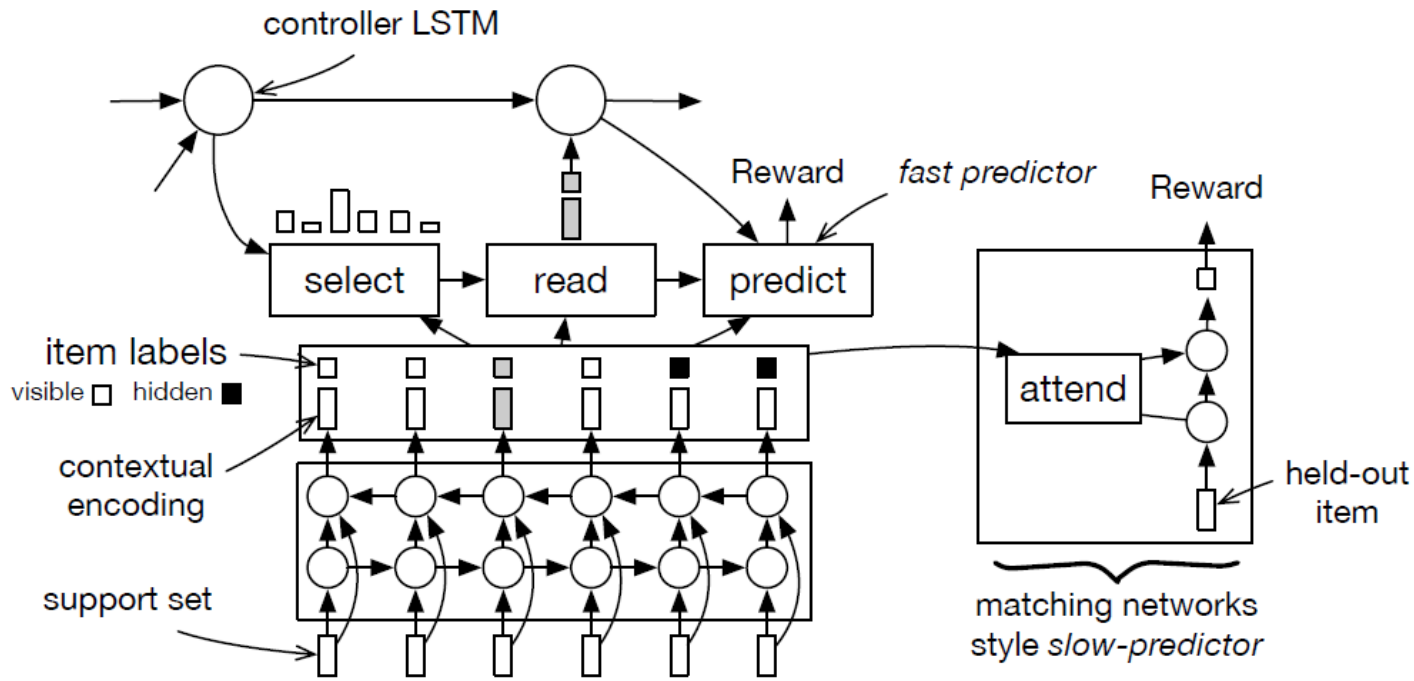
$\vec{S}$ denotes the set of intermediate states $\{(S_t, s_t)\}$ generated by the model.

$R(\vec{S})$ denotes the sum of rewards received by the model while working on episode (S, E)

Rather than using the gradients in Equation 5 directly, we optimize the model parameters using generalized Advantage Estimation (Schulman et al., 2016b), which **provides an actor-critic approach for approximately optimizing the policy gradients** in Equation 5.

Algorithm 1 End-to-end active learning loop (for Eq. 3)

 # encode items in S with context-sensitive encoder
 # and encode items in E with context-free encoder
$S = \{(x, y)\}$, $S_0^u = \{(x, \cdot)\}$, $S_0^k = \emptyset$, $E = \{(\hat{x}, \hat{y})\}$
for $t = 1 \ldots T$ **do**
    # select next instance
    $i \leftarrow \texttt{SELECT}(S_{t-1}^u, S_{t-1}^k, s_{t-1})$
    # read labeled instance and update controller
    $(x_i, y_i) \leftarrow \texttt{READ}(S, i)$
    $s_t \leftarrow \texttt{UPDATE}(s_{t-1}, x_i, y_i)$
    # update known / unknown set
    $S_t^k \leftarrow S_{t-1}^k \cup \{(x_i, y_i)\}$
    $S_t^u \leftarrow S_{t-1}^u \setminus \{(x_i, \cdot)\}$
    # perform fast prediction
    $L_t^S \leftarrow \texttt{FAST-PRED}(S, S_t^u, S_t^k, s_t)$
end for
 # perform slow prediction
$L_T^E \leftarrow \texttt{SLOW-PRED}(E, S_T^u, S_T^k, s_T)$



Conclusion

We introduced a model that learns active learning algorithms end-to-end.

For a distribution of related tasks, our model jointly learns: **a data representation**, an item **selection heuristic**, and a **prediction function**.

Our goal was to **move away from engineered selection heuristics towards strategies learned directly from data.**

Our model leverages labeled instances from different but related tasks to learn a selection strategy for the task at hand, while simultaneously adapting its representation of the data and its prediction function.