IS 6733: Deep Learning on Cloud Platforms

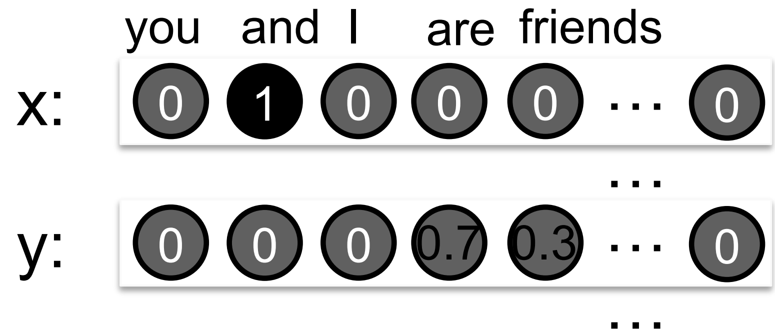
Encoder Decoder Models

Copy Right Notice

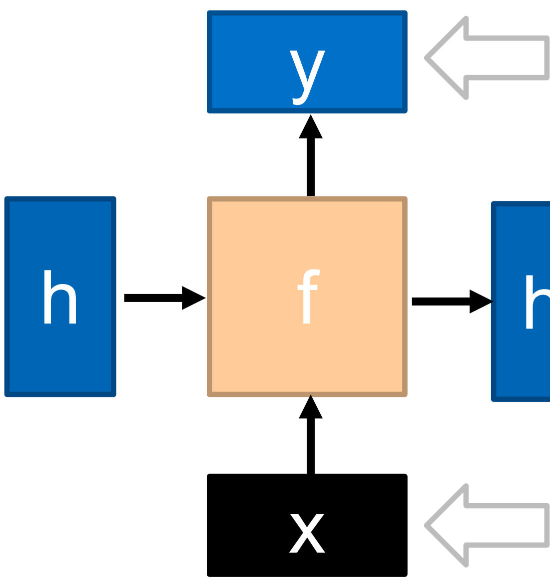
- Most slides in this presentation are adopted from slides of text book and various sources. The Copyright belong to the original authors. Thanks!

Sequence Generation

Generation



- **Sentences are composed of characters/words**
- **Generating a character/word at each time by RNN**



Distribution over the token

(sampling from the distribution to generate a token)

The token generated at the last time step.

(represented by 1-of-N encoding)

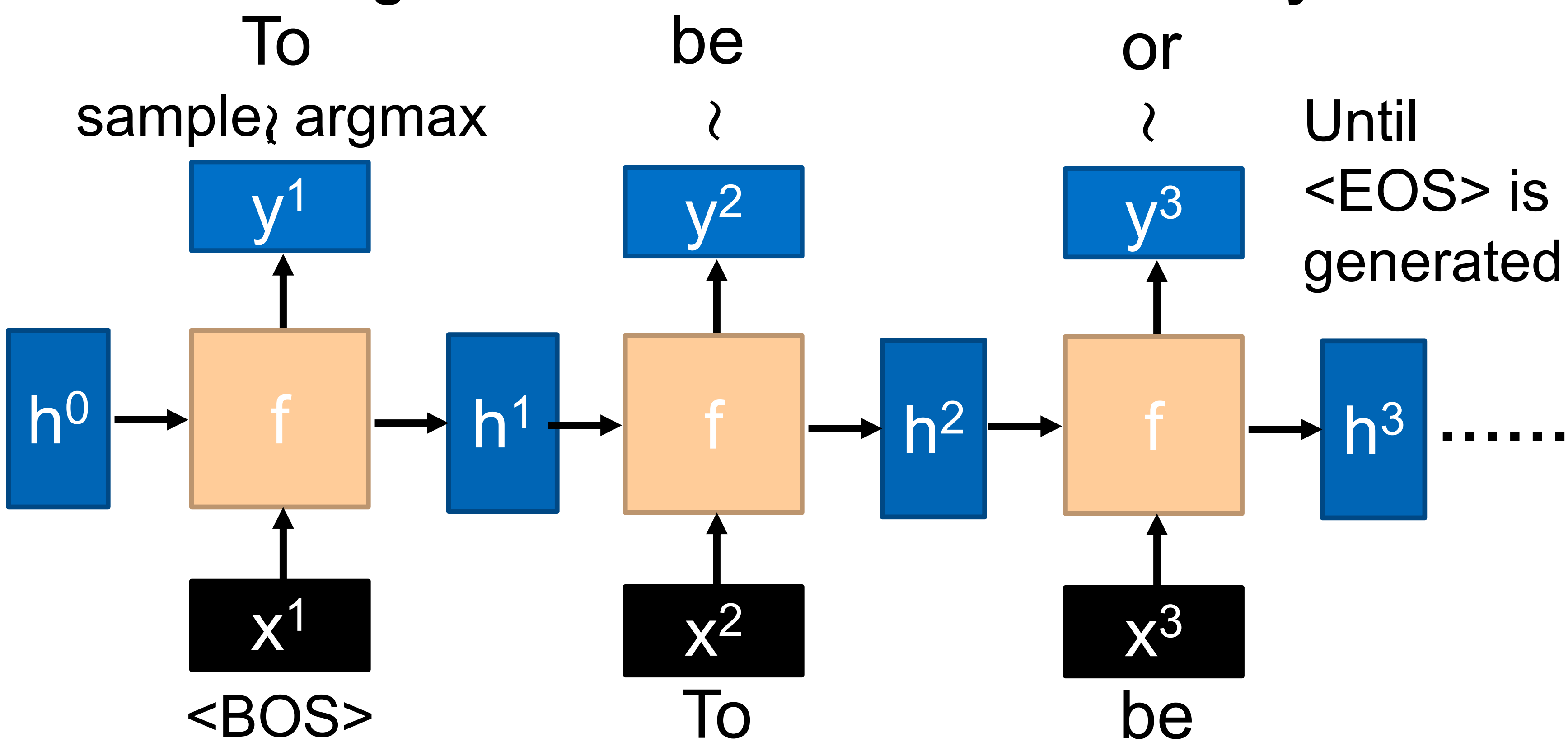
Generation

y^1: P(w|<BOS>)

y^2: P(w|<BOS>,to)

y^3: P(w|<BOS>,to, be)

- **Sentences are composed of characters/words**
- **Generating a character/word at each time by RNN**

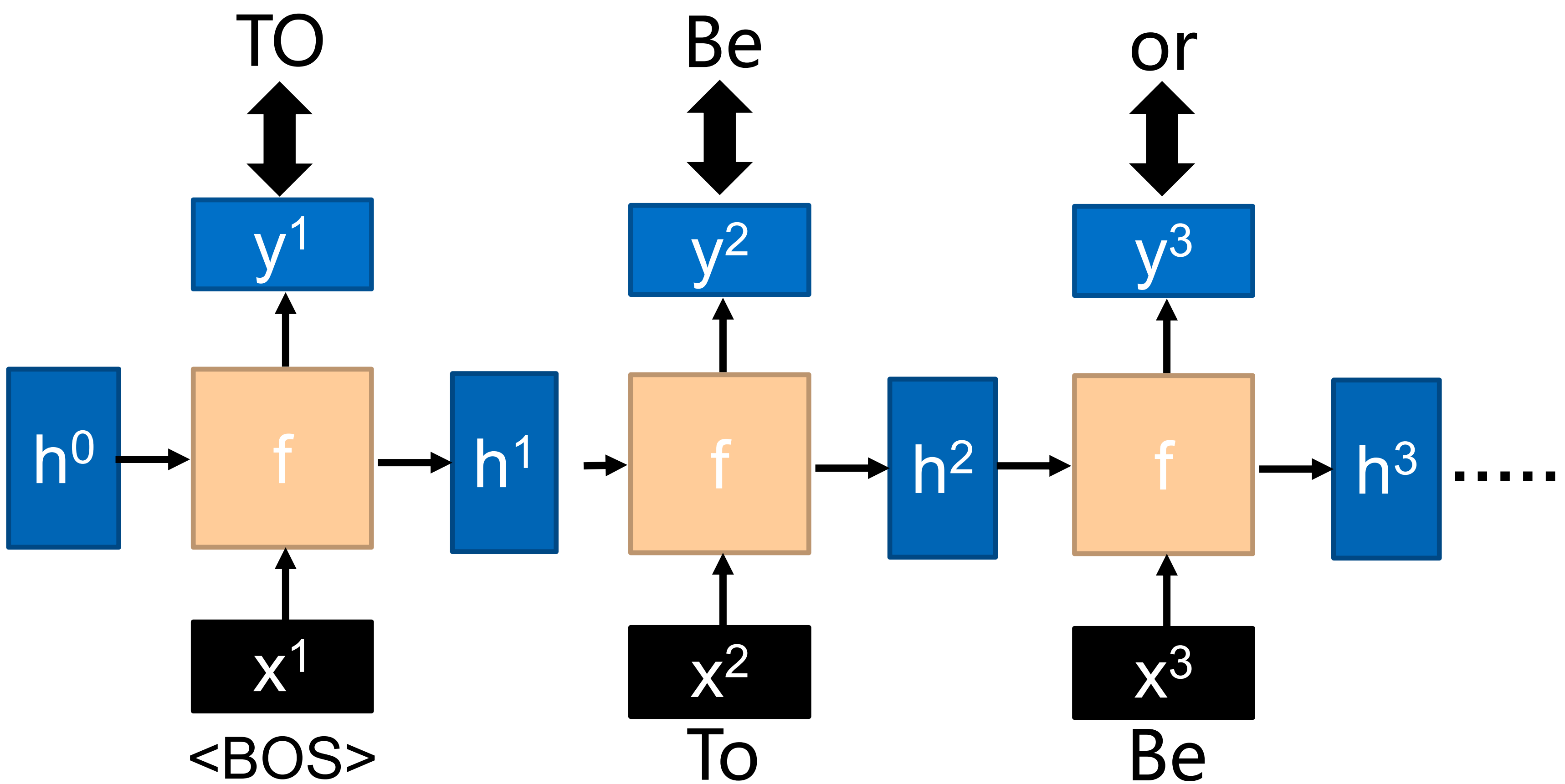


Generation

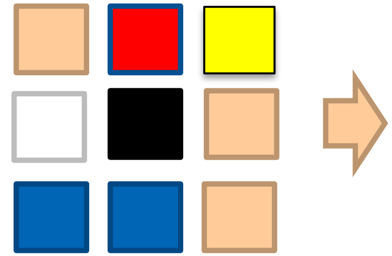
: minimizing cross-entropy

- ***Training***

Training data: To be or not to be



Generation

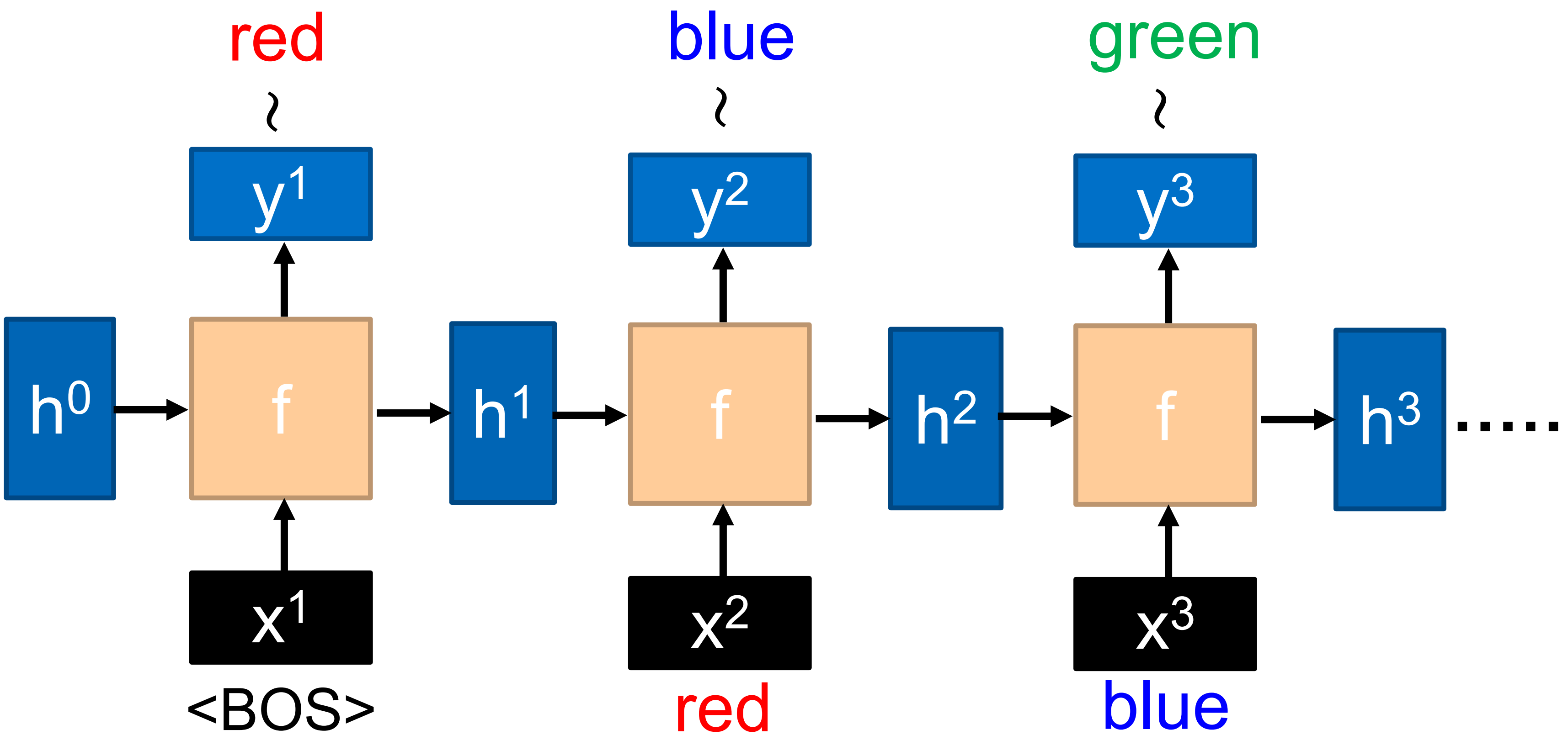


Consider as a sentence

blue red yellow gray……

Train a RNN based on the "sentences"

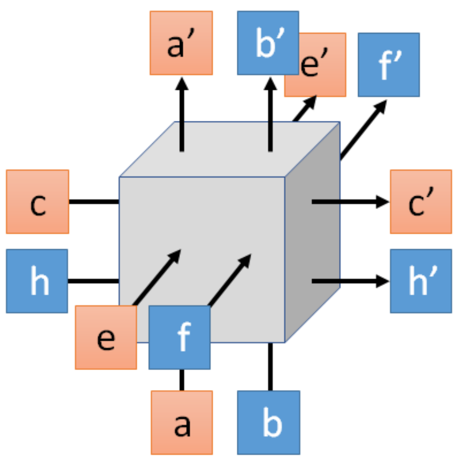
- **Images are composed of pixels**
- **Generating a pixel at each time by RNN**

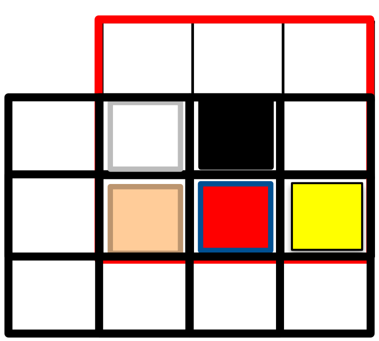


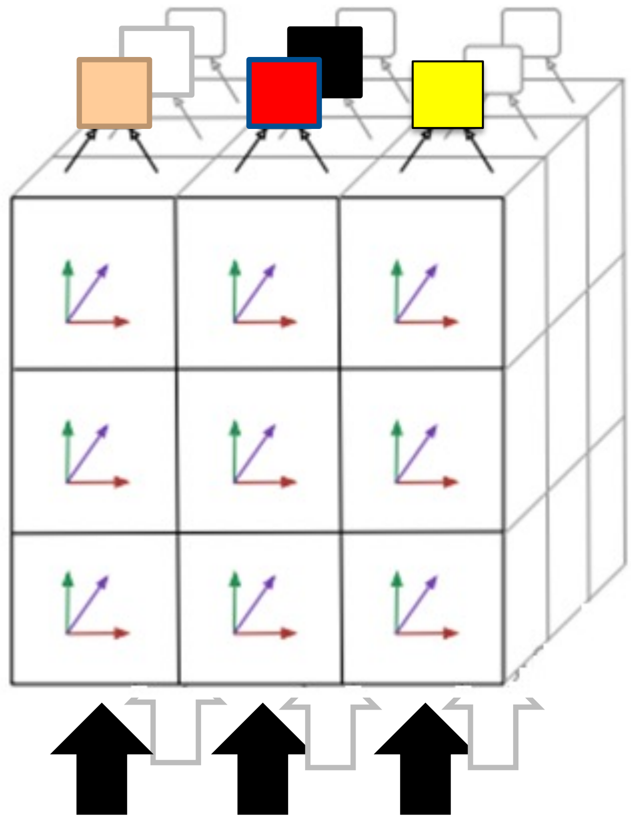
Generation - PixelRNN

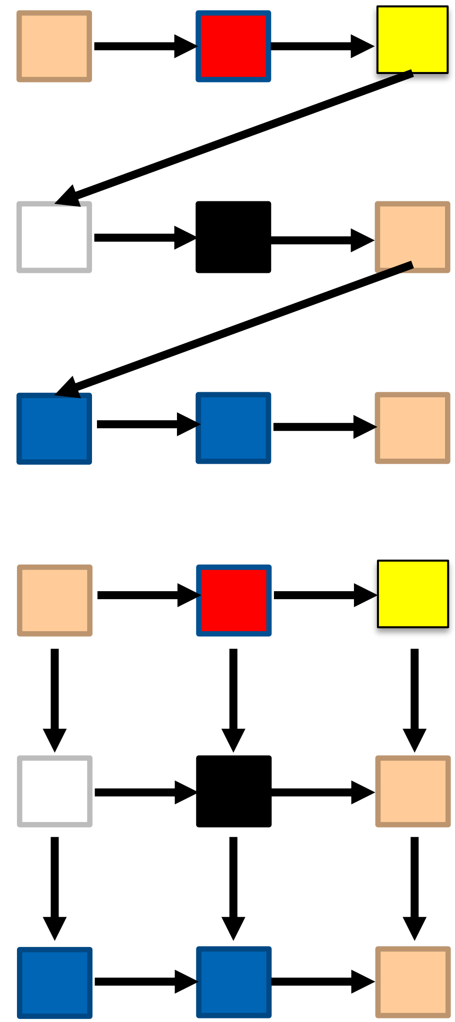
3 x 3 images

- Images are composed of pixels









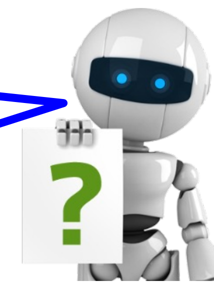
Conditional Generation

- **We don't want to simply generate some random sentences.**
- **Generate sentences based on conditions:**

<u>Caption Generation</u>

Given condition:

"A young girl is dancing."

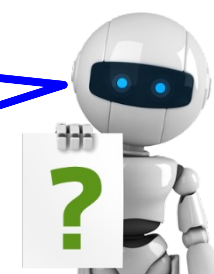


<u>Chat-bot</u>

Given condition:

"Hello"

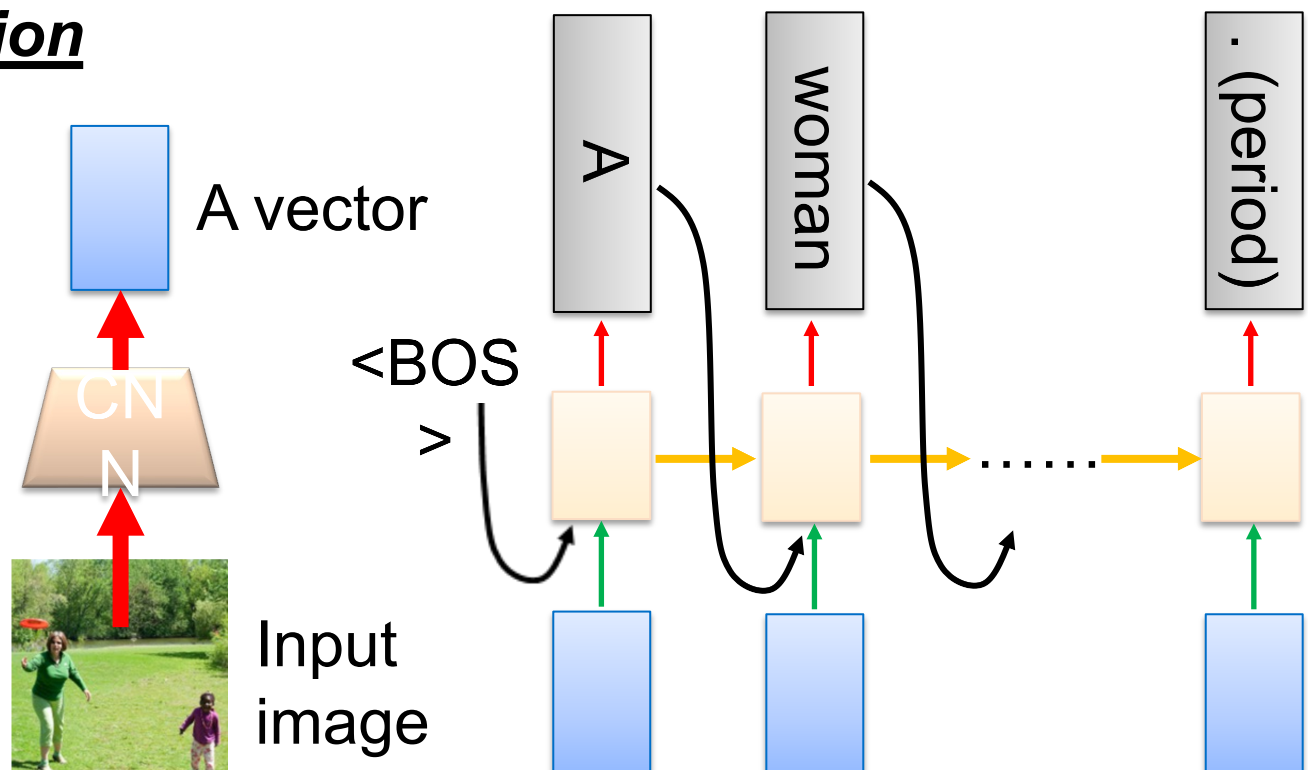
"Hello. Nice to see you."



Conditional Generation

- **Represent the input condition as a vector, and consider the vector as the input of RNN generator**

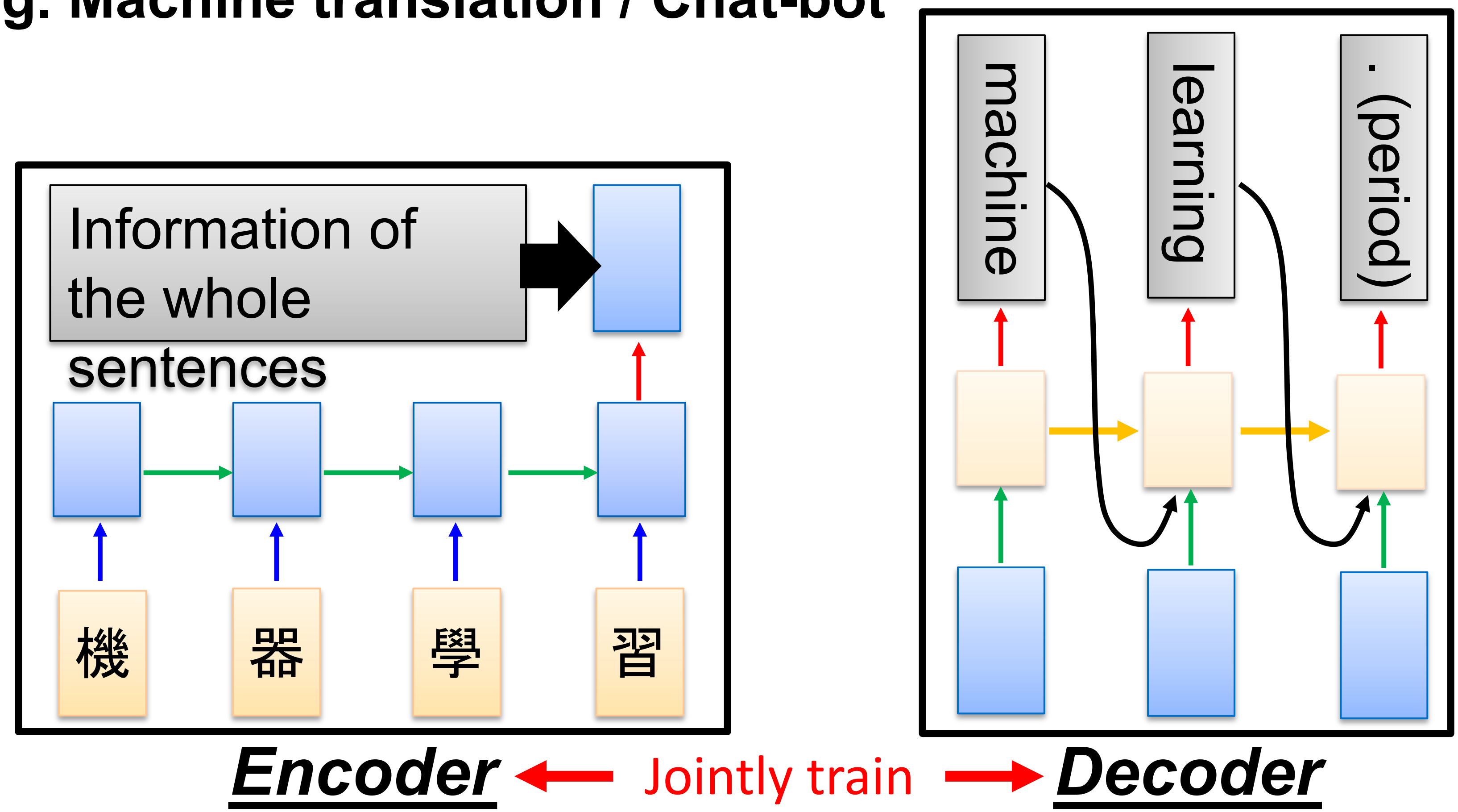
<u>Image Caption Generation</u>



Conditional Generation

Sequence-to-sequence learning

- **Represent the input condition as a vector, and consider the vector as the input of RNN generator**
- **E.g. Machine translation / Chat-bot**



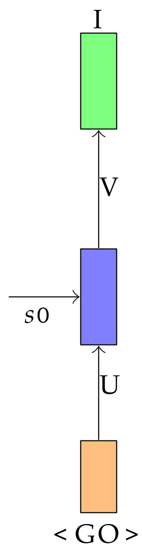
Encoder Decoder Models

- We will start by revisiting the problem of language modeling
- Informally, given '$t-1$' words we are interested in predicting the t^{th} word
- More formally, given $y_1, y_2, \ldots, y_{t-1}$ we want to find

  - $y^* = argmax\ P(y_t | y_1, y_2, \ldots, y_{t-1})$

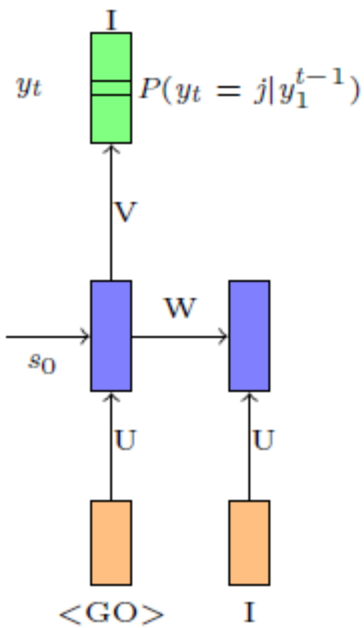
- Let us see how we model $P(y_t | y_1, y_2 \ldots y_{t-1})$ using a RNN
- We will refer to $P(y_t | y_1, y_2 \ldots y_{t-1})$ by shorthand notation: $P(y_t | y_1^{t-1})$

Encoder Decoder Models



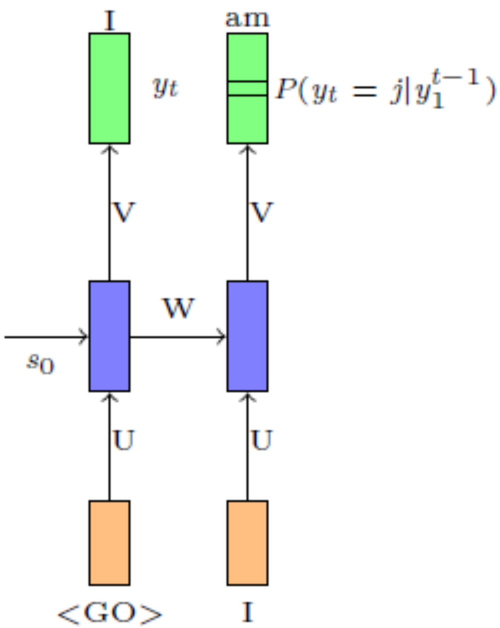
- **Informally, given '$t - i$' words we are interested in predicting the t^{th} word**
- **More formally, given y_1, y_2, ..., y_{t-1} we want to find**
- $y^* = argmax\ P(y_t|y_1, y_2, ..., y_{t-1})$
- **Let us see how we model $P(y_t|y_1, y_2 ... y_{t-1})$ using a RNN**
- **We will refer to $P(y_t|y_1, y_2 ... y_{t-1})$ by shorthand notation:** $P(y_t|y_1^{t-1})$

Encoder Decoder Models



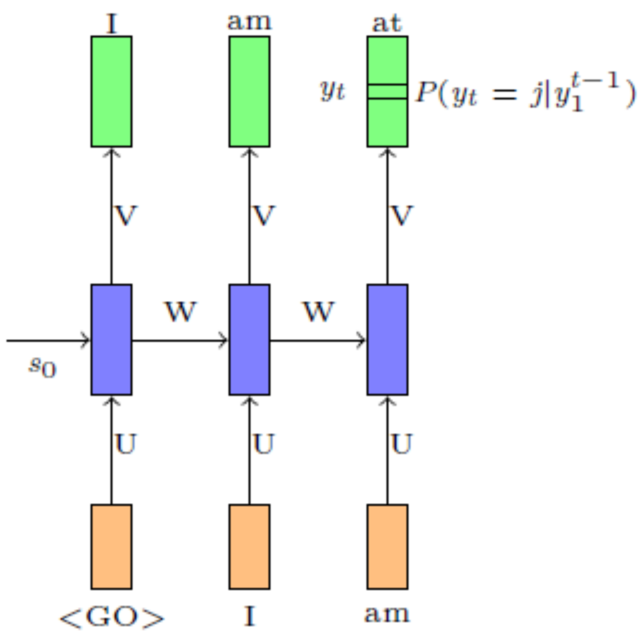
- Informally, given '$t - i$' words we are interested in predicting the t^{th} word
- More formally, given y_1, y_2, ..., y_{t-1} we want to find
- $y^* = argmax\ P(y_t|y_1, y_2, ..., y_{t-1})$
- Let us see how we model $P(y_t|y_1, y_2 ... y_{t-1})$ using a RNN
- We will refer to $P(y_t|y_1, y_2 ... y_{t-1})$ by shorthand notation: $P(y_t|y_1^{t-1})$

Encoder Decoder Models



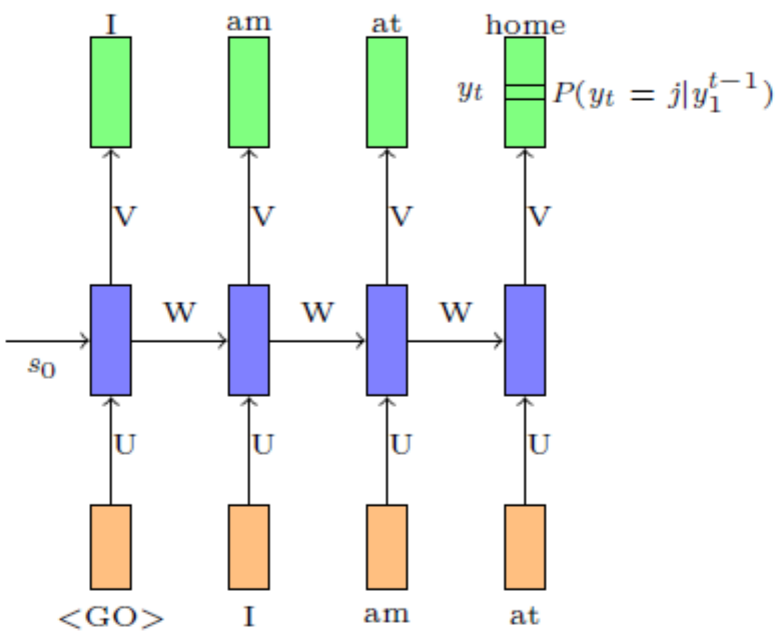
- **Informally, given '$t - i$' words we are interested in predicting the t^{th} word**
- **More formally, given y_1, y_2, ..., y_{t-1} we want to find**
- $y^* = argmax\ P(y_t|y_1, y_2, ..., y_{t-1})$
- **Let us see how we model $P(y_t|y_1, y_2 ... y_{t-1})$ using a RNN**
- **We will refer to $P(y_t|y_1, y_2 ... y_{t-1})$ by shorthand notation:** $P(y_t|y_1^{t-1})$

Encoder Decoder Models



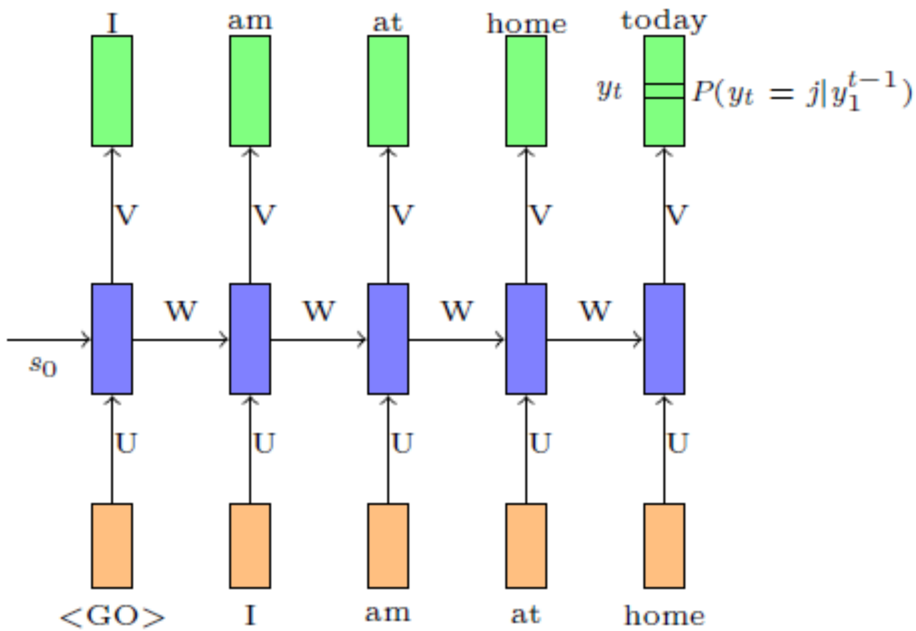
- **Informally, given '$t - i$' words we are interested in predicting the t^{th} word**
- **More formally, given y_1, y_2, ..., y_{t-1} we want to find**
- $y^* = argmax\ P(y_t|y_1, y_2, ..., y_{t-1})$
- **Let us see how we model $P(y_t|y_1, y_2 ... y_{t-1})$ using a RNN**
- **We will refer to $P(y_t|y_1, y_2 ... y_{t-1})$ by shorthand notation:** $P(y_t|y_1^{t-1})$

Encoder Decoder Models



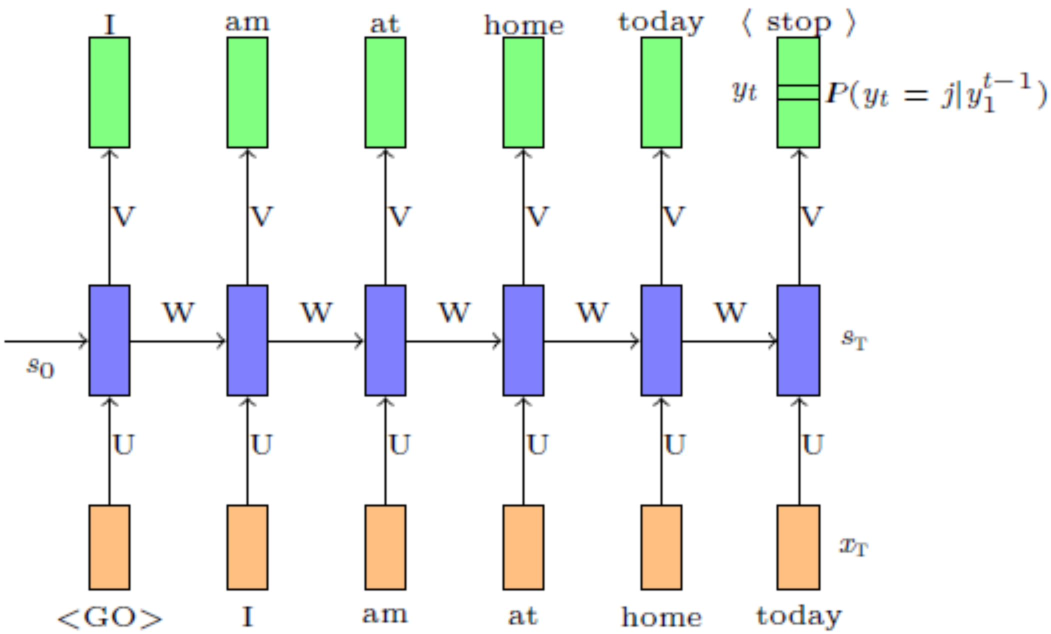
- **Informally, given '$t - i$' words we are interested in predicting the t^{th} word**
- **More formally, given y_1, y_2, ..., y_{t-1} we want to find**
- $y^* = argmax\, P(y_t|y_1, y_2, ..., y_{t-1})$
- **Let us see how we model $P(y_t|y_1, y_2 ... y_{t-1})$ using a RNN**
- **We will refer to $P(y_t|y_1, y_2 ... y_{t-1})$ by shorthand notation:** $P(y_t|y_1^{t-1})$

Encoder Decoder Models



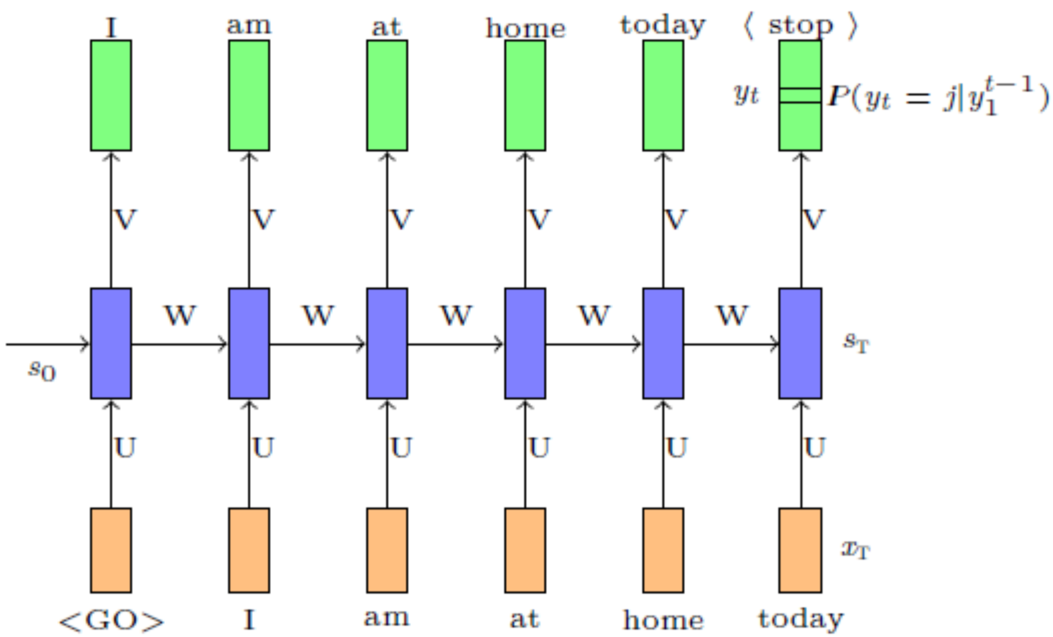
- **Informally, given '$t - i$' words we are interested in predicting the t^{th} word**
- **More formally, given $y_1, y_2, ..., y_{t-1}$ we want to find**
- $y^* = argmax\ P(y_t|y_1, y_2, ..., y_{t-1})$
- **Let us see how we model $P(y_t|y_1, y_2...y_{t-1})$ using a RNN**
- **We will refer to $P(y_t|y_1, y_2...y_{t-1})$ by shorthand notation:** $P(y_t|y_1^{t-1})$

Encoder Decoder Models



- **Informally, given '$t - i$' words we are interested in predicting the t^{th} word**
- **More formally, given $y_1, y_2, \ldots, y_{t-1}$ we want to find**
- $y^* = argmax\ P(y_t|y_1, y_2, \ldots, y_{t-1})$
- **Let us see how we model $P(y_t|y_1, y_2 \ldots y_{t-1})$ using a RNN**
- **We will refer to $P(y_t|y_1, y_2 \ldots y_{t-1})$ by shorthand notation:** $P(y_t|y_1^{t-1})$

Encoder Decoder Models



- We are interested in

$$P(y_t = j|y_1, y_2 \ldots y_{t-1})$$

  - where $j \in V$ and V is the set of all vocabulary words.

- **Using an RNN we compute this as**

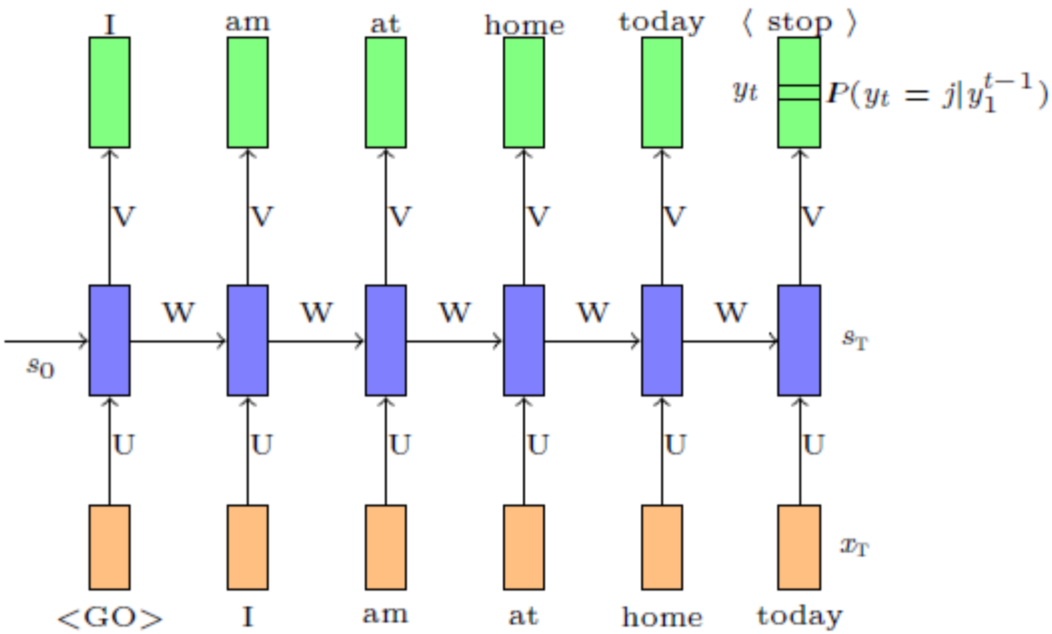
$$P(y_t = j|y_1^{t-1}) = softmax(Vs_t + c)_j$$

- **In other words we compute**

$$P(y_t = j|y_1^{t-1}) = P(y_t = j|s_t)$$
$$= softmax(Vs_t + c)_j$$

- **Notice that the recurrent connections ensure that s_t has information about y_1^{t-1}**

Encoder Decoder Models



Data:

India, officially the Republic of India, is a country in South Asia. It is the seventh-largest country by area, ...

- **Data:** All sentences from any large corpus (say wikipedia)
- **Model:**

$$s_t = \sigma(Ws_{t-1} + Ux_t + b)$$
$$P(y_t = j|y_1^{t-1}) = softmax(Vs_t + c)_j$$

- **Parameters:** *U; V; W; b; c*
- **Loss:**

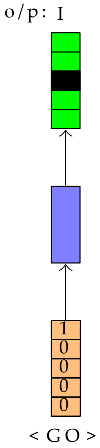
$$\mathscr{L}(\theta) = \sum_{t=1}^{T} \mathscr{L}_t(\theta)$$
$$\mathscr{L}_t(\theta) = -\log P(y_t = \ell_t|y_1^{t-1})$$

where ℓ_t is the true word at time step

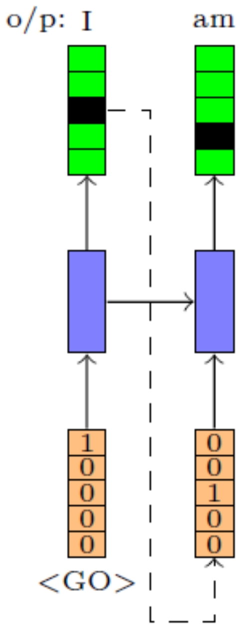
- **Algorithm:** backpropagation through time.

Encoder Decoder Models

- **What is the input at each time step?**

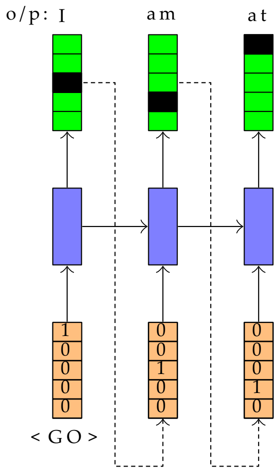


Encoder Decoder Models



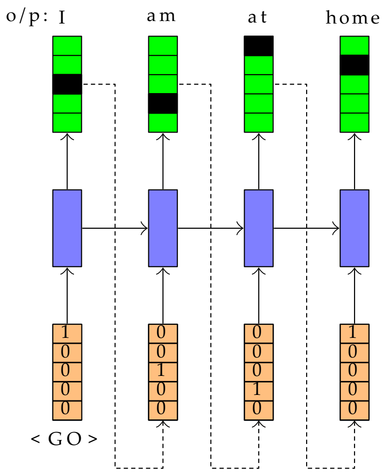
- **What is the input at each time step?**
- **It is simply the word that we predicted at the previous time step**

Encoder Decoder Models



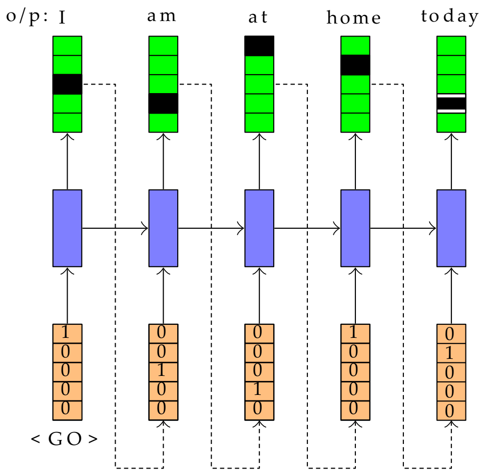
- What is the input at each time step?
- It is simply the word that we predicted at the previous time step

Encoder Decoder Models



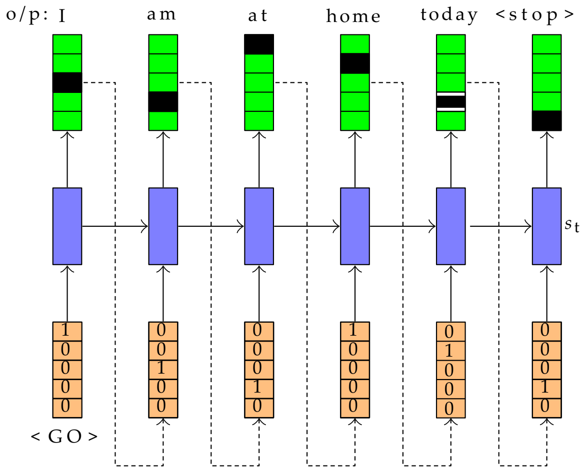
- What is the input at each time step?
- It is simply the word that we predicted at the previous time step

Encoder Decoder Models



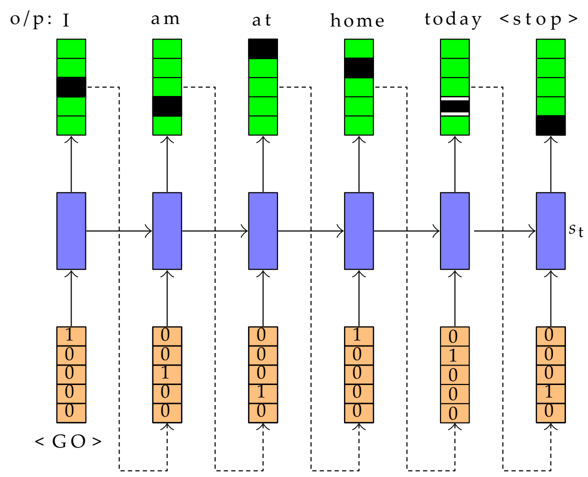
- **What is the input at each time step?**
- **It is simply the word that we predicted at the previous time step**

Encoder Decoder Models



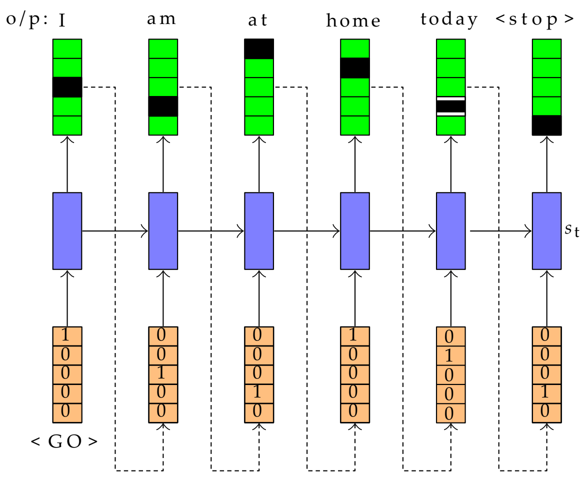
- **What is the input at each time step?**
- **It is simply the word that we predicted at the previous time step**

Encoder Decoder Models



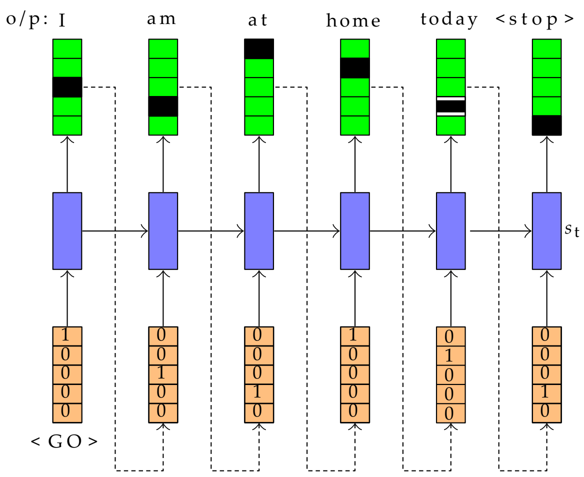
- **What is the input at each time step?**
- **It is simply the word that we predicted at the previous time step**
- **In general** $s_t = RNN(s_{t-1}, x_t)$
- **Let j be the index of the word which has been assigned the max probability at time step $t-1$:** $x_t = e(v_j)$
- x_t **is essentially a one-hot vector ($e(v_j)$)representing the j^{th} word in the vocabulary**
- **In practice, instead of one hot representation we use a pre-trained word embedding of the j^{th} word**

Encoder Decoder Models



- Notice that *s*0 is not computed but just randomly initialized
- We learn it along with the other parameters of RNN (or LSTM or GRU)
- We will return back to this later

Encoder Decoder Models



- Notice that *s*0 is not computed but just randomly initialized
- We learn it along with the other parameters of RNN (or LSTM or GRU)
- We will return back to this later

Encoder Decoder Models

$$s_t = \sigma(U x_t + W s_{t-1} + b)$$

$$\tilde{S}_t = \sigma(W(o_t \odot s_{t-1}) + U x_t + b) \quad s_t = i_t \odot s_{t-1} + (1 - i_t) \odot \tilde{s}_t$$

$$\tilde{s}_t = \sigma(W h_{t-1} + U x_t + b)$$
$$s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s}_t$$
$$h_t = o_t \odot \sigma(s_t)$$

$$s_t = \mathrm{RNN}(s_{t-1}, x_t)$$

$$s_t = \mathrm{GRU}(s_{t-1}, x_t)$$

$$h_t, s_t = \mathrm{LSTM}(h_{t-1}, s_{t-1}, x_t)$$

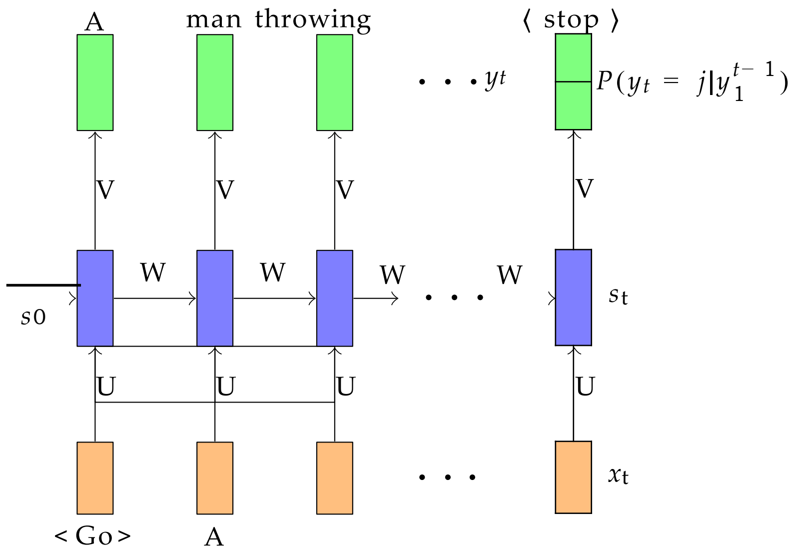
- Before moving on we will see a compact way of writing the function computed by RNN, GRU and LSTM
- We will use these notations going forward

Encoder Decoder Models

- So far we have seen how to model the conditional probability distribution $P(y_t|y_1^{t-1})$
- More informally, we have seen how to generate a sentence given previous words
- What if we want to generate a sentence given an image?

A man throwing
a frisbee in a park

Encoder Decoder Models



A man throwing a frisbee in a park

- So far we have seen how to model the conditional probability distribution

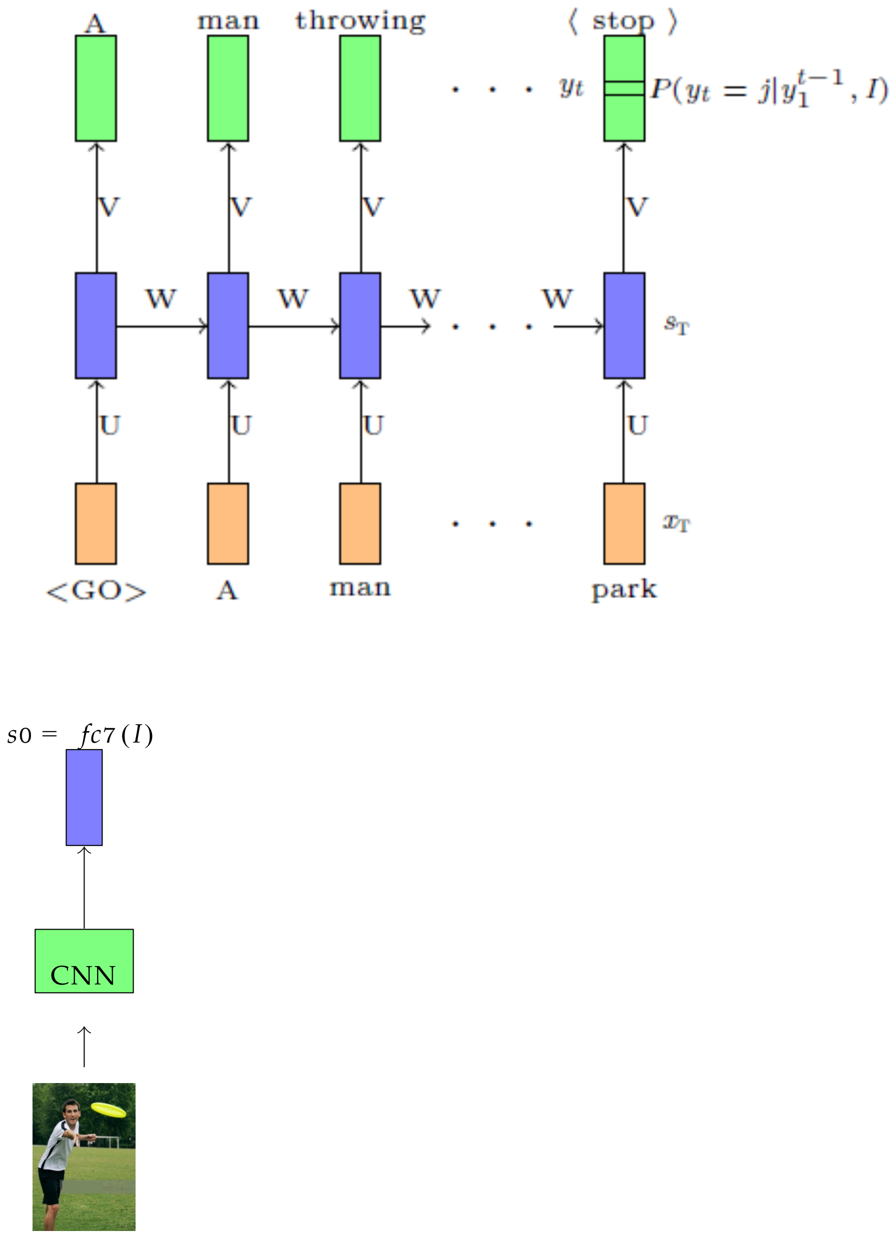
$$P(y_t | y_1^{t-1})$$

- More informally, we have seen how to generate a sentence given previous words
- What if we want to generate a sentence given an image?
- We are now interested in

$$P(y_t | y_1^{t-1}, I)$$

where *I* is an image

Encoder Decoder Models



- Earlier we modeled $P(y_t|y_1^{t-1})$ as

$$P(y_t|y_1^{t-1}) = P(y_t = j|s_t)$$

- Where s_t was a state capturing all the previous words
- We could now model $P(y_t|y_1^{t-1}, I)$ as

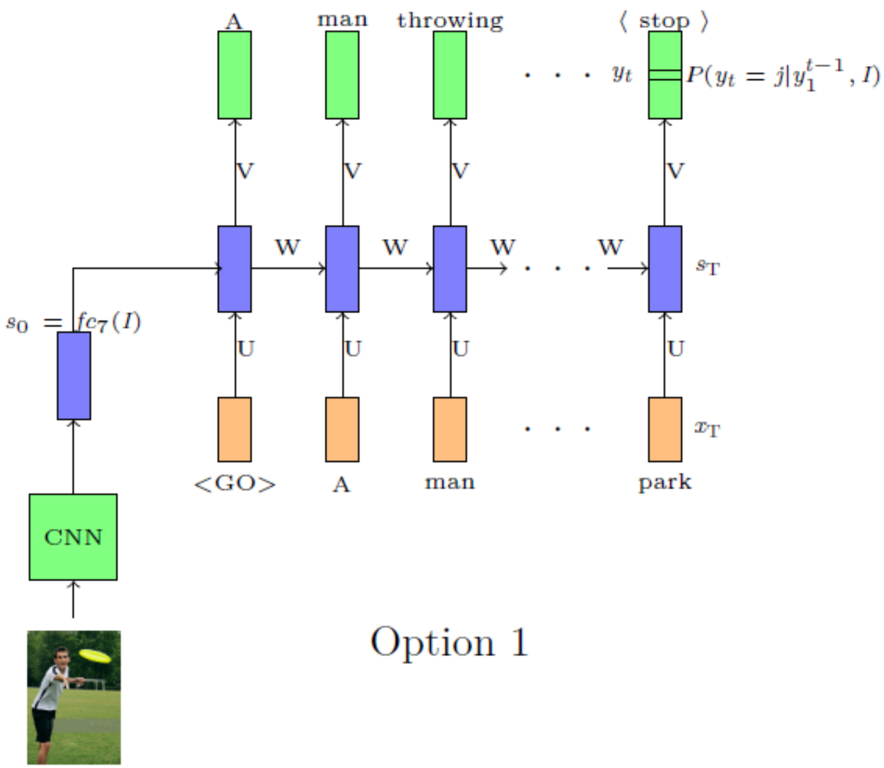
$$P(y_t = j|s_t, f_{c7}(I))$$

  - where *fc7*(*I*) is the representation obtained from the *fc7* layer of an image

Encoder Decoder Models

- **There are many ways of making $P(y_t = j)$ conditional on $f_{c_7}(I)$**
- **Let us see two such options**

Encoder Decoder Models

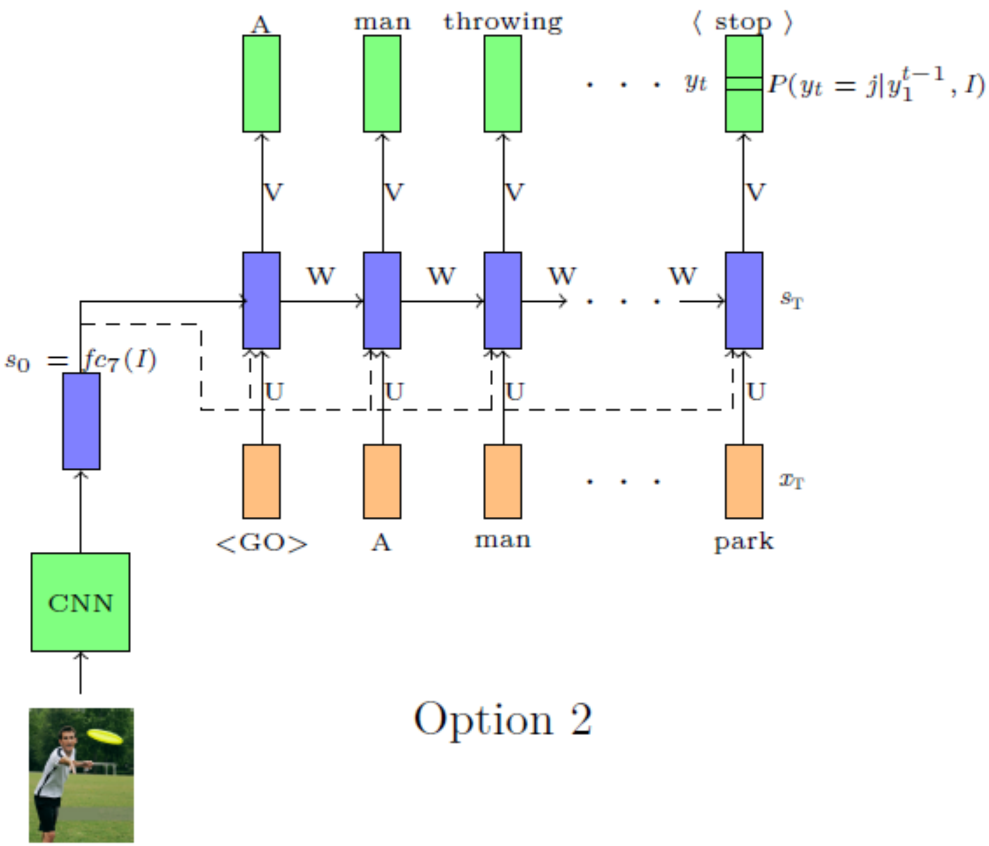


Option 1

- Option 1: Set $s_0 = f_{c_7}(I)$
- Now s_0 and hence all subsequent s_t's depend on $f_{c_7}(I)$
- We can thus say that $P(y_t = j)$ depends on $f_{c_7}(I)$
- In other words, we are computing

$$P(y_t = j | s_t, f_{c_7}(I))$$

Encoder Decoder Models



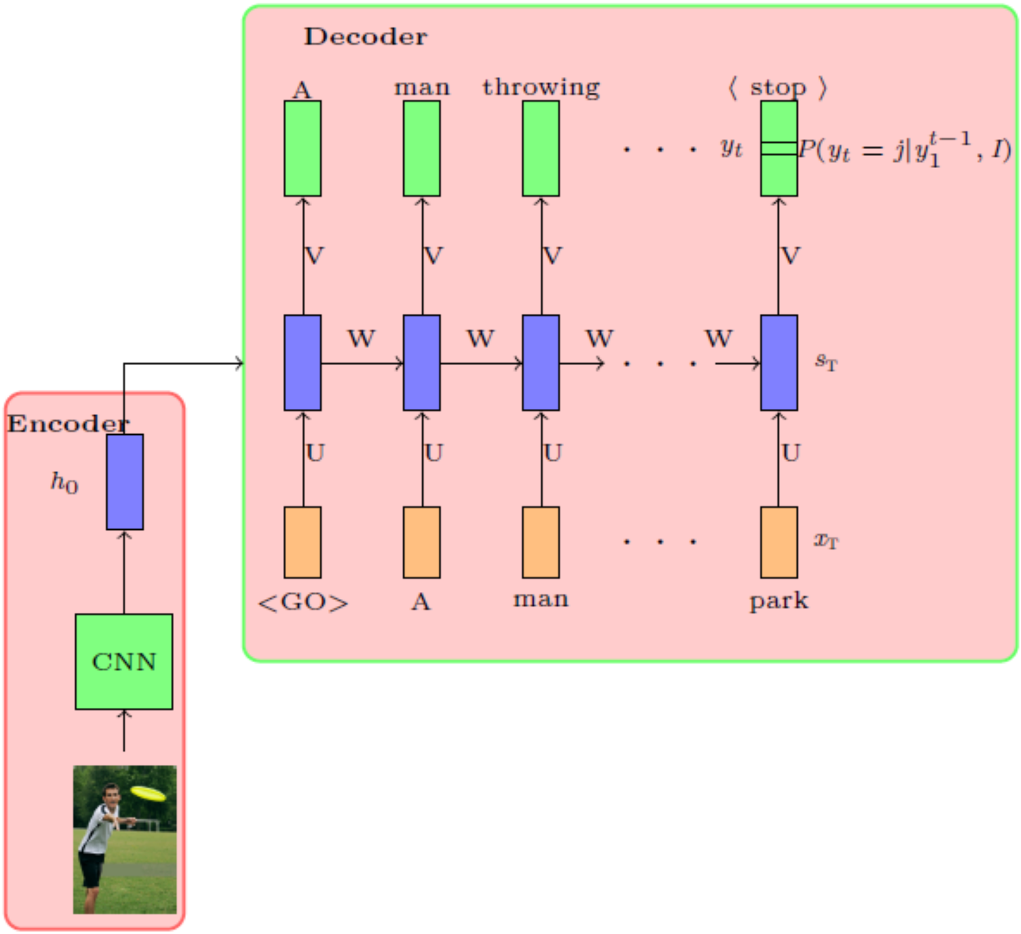
Option 2

- Option 2: Another more explicit way of doing this is to compute

$$s_t = RNN(s_{t-1}, [x_t, f_{c_7}(I))]$$

- In other words we are explicitly using f_{c7} (I) to compute s_t and hence $P(y_t = j)$
- You could think of other ways of conditioning $P(y_t = j)$ on f_{c7}

Encoder Decoder Models



- Let us look at the full architecture
- A CNN is first used to **encode** the image
- A RNN is then used to decode (generate) a sentence from the encoding
- This is a typical **encoder decoder architecture**
- Both the encoder and decoder use a neural network
- Alternatively, the encoder's output can be fed to every step of the decoder

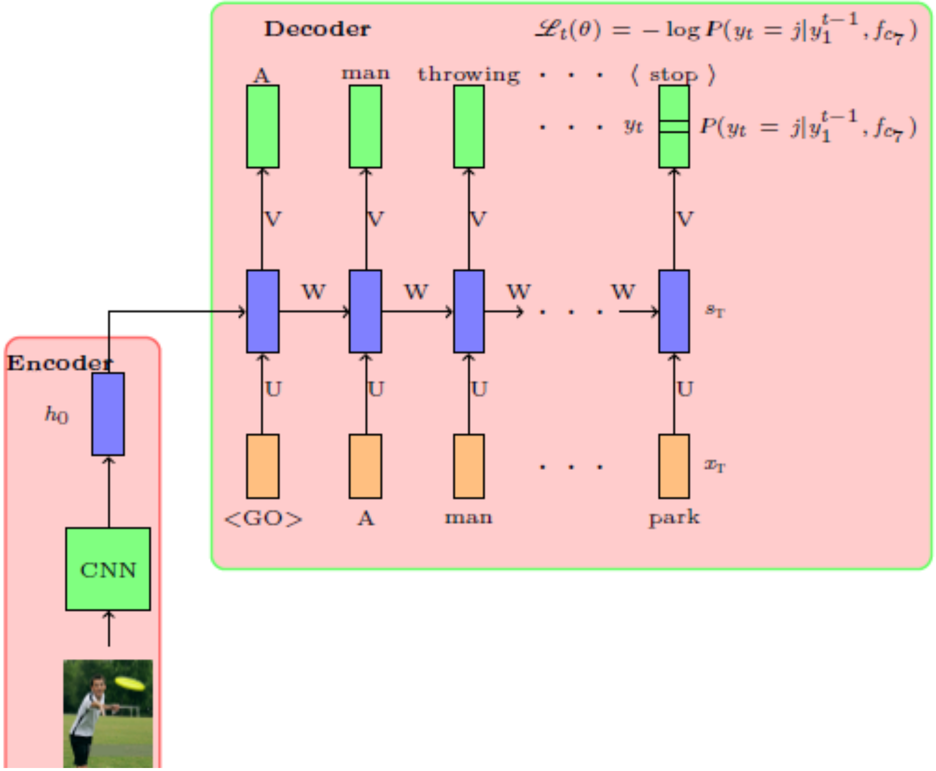
Applications of Encoder Decoder models

- For all these applications we will try to answer the following questions
- What kind of a network can we use to encode the input(s)? (What is an appropriate encoder?)
- What kind of a network can we use to decode the output? (What is an appropriate decoder?)
- What are the parameters of the model ?
- What is an appropriate loss function ?

Task: Image captioning

A man throwing . . . ⟨ stop ⟩

Task: Image captioning



- **Data:** $\{x_i = image_i, y_i = caption_i\}$
- **Model:**
  - **Encoder:** $s_0 = CNN(x_i)$
  - **Decoder:**

$$s_t = RNN(s_{t-1}, e(\hat{y}_{t-1}))$$

$$P(y_t|y_1^{t-1}, I) = softmax(Vs_t + b)$$

- **Parameters:** $U_{dec}, V, W_{dec}, W_{conv}, b$
- **Loss:**

$$\mathcal{L}(\theta) = \sum_{i=1}^{T} \mathcal{L}_t(\theta) \quad = -\sum_{t=1}^{T} \log P(y_t = \ell_t|y_1^{t-1}, I)$$

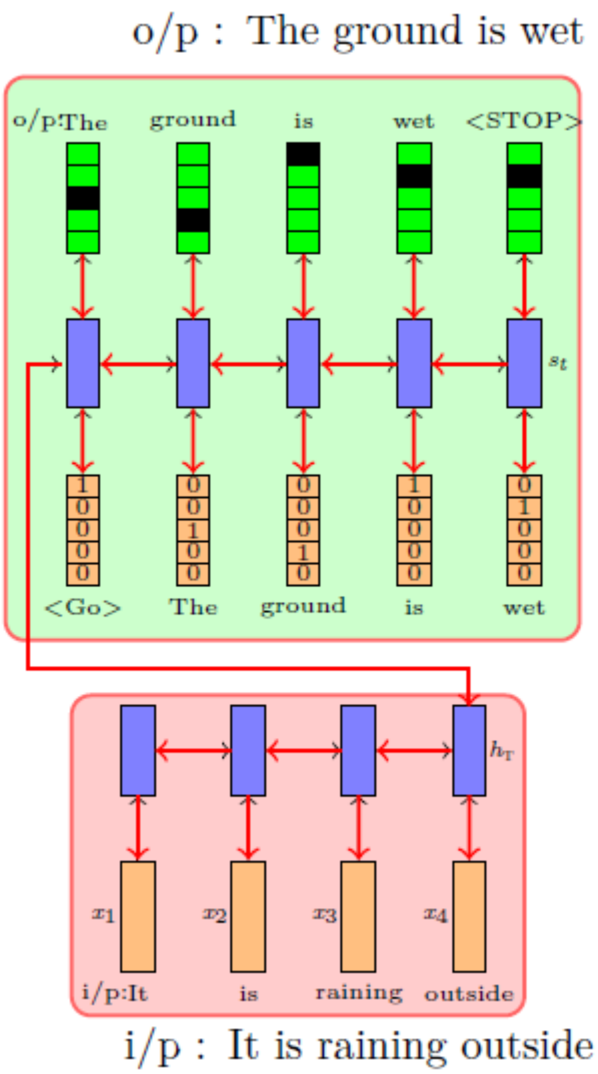
- **Algorithm:** Gradient descent with backpropagation

Task: Textual entailment

o/p : The ground is wet

i/p : It is raining outside

Task: Textual entailment （1）



- **Data:** *{x_i = premise$_i$, y_i = hypothesis$_i$}*
- **Model** (Option 1):
  - **Encoder:** $h_t = RNN(h_{t-1}, x_{it})$
  - **Decoder:**

$$s_0 = h_T \quad (T \text{ is length of input})$$

$$s_t = RNN(s_{t-1}, e(\hat{y}_{t-1}))$$

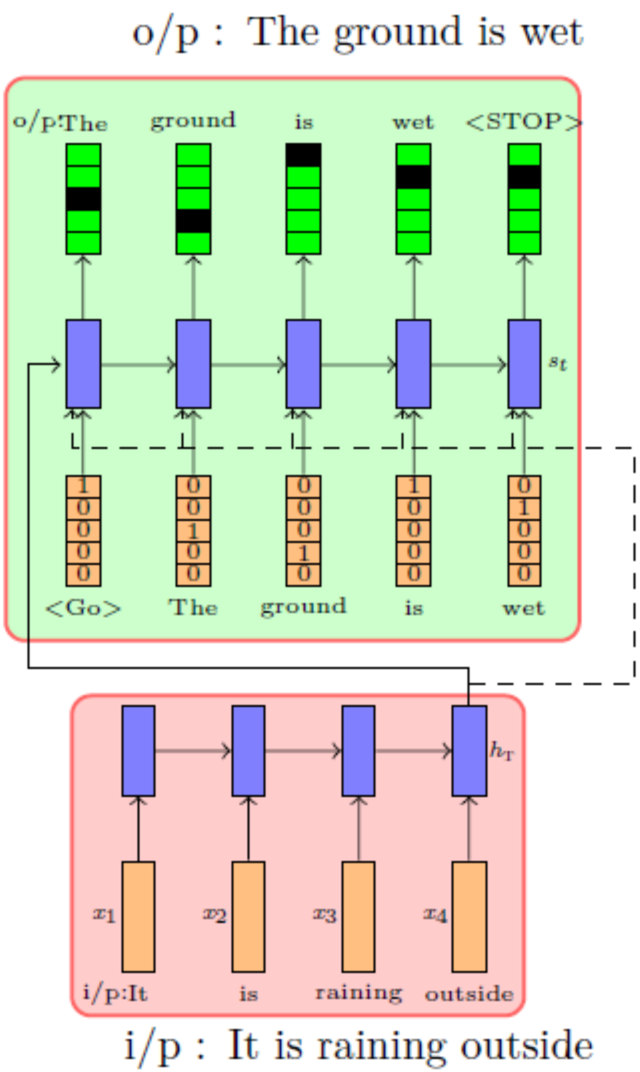
$$P(y_t | y_1^{t-1}, x) = softmax(Vs_t + b)$$

  - **Parameters:** U_{dec}, V, W_{dec}, U_{enc}, W_{enc}, b
  - **Loss:**

$$\mathscr{L}(\theta) = \sum_{i=1}^{T} \mathscr{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t | y_1^{t-1}, x)$$

  - **Algorithm:** Gradient descent with backpropagation

Task: Textual entailment (2)



- **Data:** $\{x_i = premise_i,\ y_i = hypothesis_i\}$
- **Model** (Option 2):
  - **Encoder:** $h_t = RNN(h_{t-1}, x_{it})$
  - **Decoder:**

$$s_0 = h_T \quad (T \text{ is length of input})$$
$$s_t = RNN(s_{t-1}, [h_T, e(\hat{y}_{t-1})])$$
$$P(y_t|y_1^{t-1}, x) = softmax(Vs_t + b)$$

  - **Parameters:** U_{dec}, V, W_{dec}, U_{enc}, W_{enc}, b
  - **Loss:**

$$\mathscr{L}(\theta) = \sum_{i=1}^{T} \mathscr{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t | y_1^{t-1}, x)$$

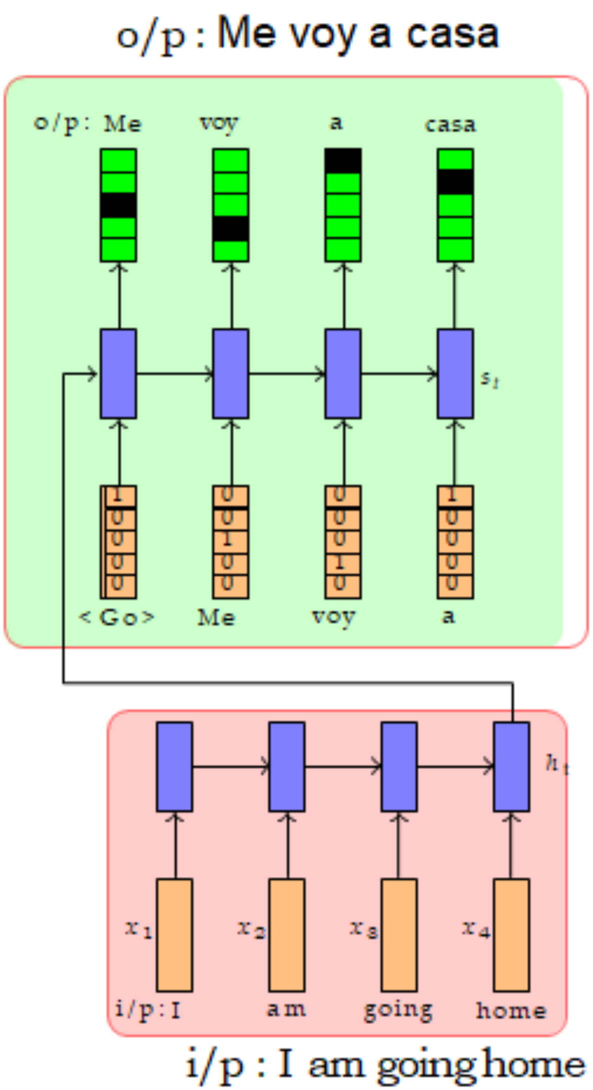
  - **Algorithm:** Gradient descent with backpropagation

Task: Machine Translation

o/p : Me voy a casa

i/p : I am going home

Task: Machine Translation (1)



- **Data:** $\{x_i = premise_i,\ y_i = hypothesis_i\}$
- **Model** (Option 1):
  - **Encoder:** $h_t = RNN(h_{t-1}, x_{it})$
  - **Decoder:**

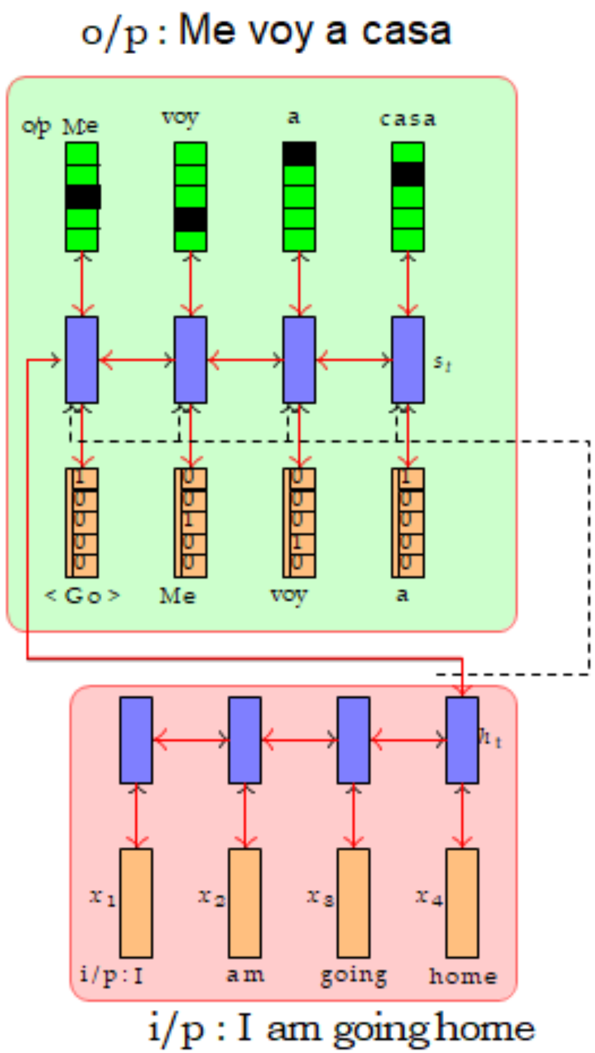
$$s_0 = h_T \quad (T \text{ is length of input})$$
$$s_t = RNN(s_{t-1}, e(\hat{y}_{t-1}))$$
$$P(y_t|y_1^{t-1}, x) = softmax(Vs_t + b)$$

  - **Parameters:** U_{dec}, V, W_{dec}, U_{enc}, W_{enc}, b
  - **Loss:**

$$\mathscr{L}(\theta) = \sum_{i=1}^{T} \mathscr{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t | y_1^{t-1}, x)$$

  - **Algorithm:** Gradient descent with backpropagation

Task: machine translation （2）



- **Data:** $\{x_i = premise_i,\ y_i = hypothesis_i\}$
- **Model** (Option 2):
  - **Encoder:** $h_t = RNN(h_{t-1}, x_{it})$
  - **Decoder:**

$$s_0 = h_T \quad (T \text{ is length of input})$$
$$s_t = RNN(s_{t-1}, [h_T, e(\hat{y}_{t-1})])$$
$$P(y_t|y_1^{t-1}, x) = softmax(Vs_t + b)$$

  - **Parameters:** U_{dec}, V, W_{dec}, U_{enc}, W_{enc}, b
  - **Loss:**

$$\mathscr{L}(\theta) = \sum_{i=1}^{T} \mathscr{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t | y_1^{t-1}, x)$$

  - **Algorithm:** Gradient descent with backpropagation

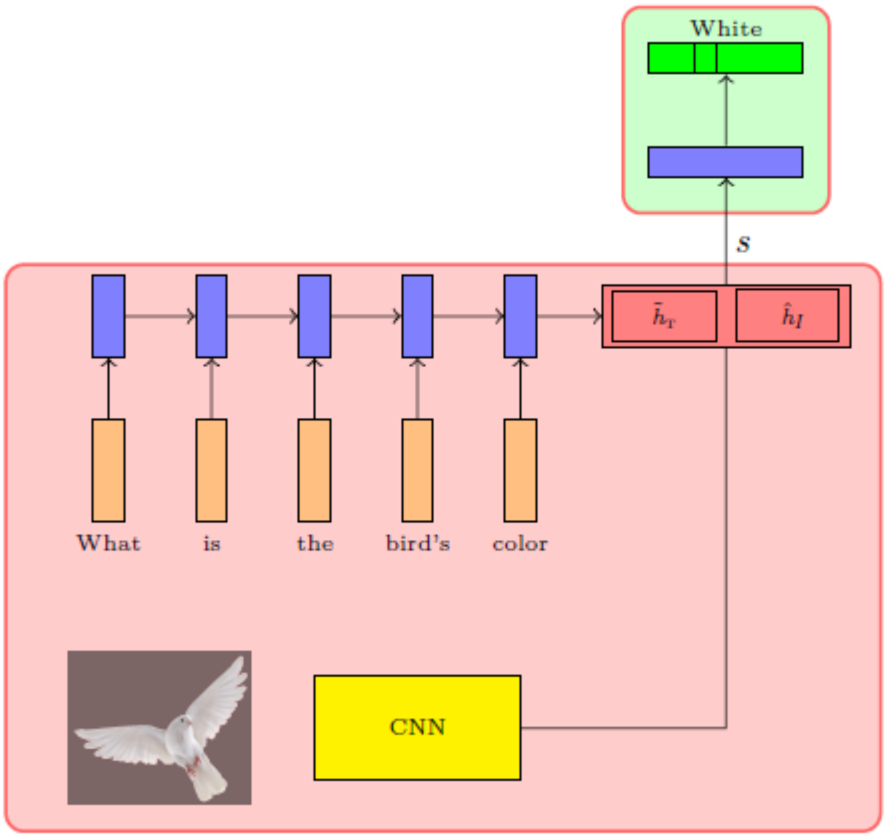
Task: Image Question Answeing

O/p: White

Question: What
is the bird’s color

Task: Image Question Answering

O/p: White



Question: What is the bird's color

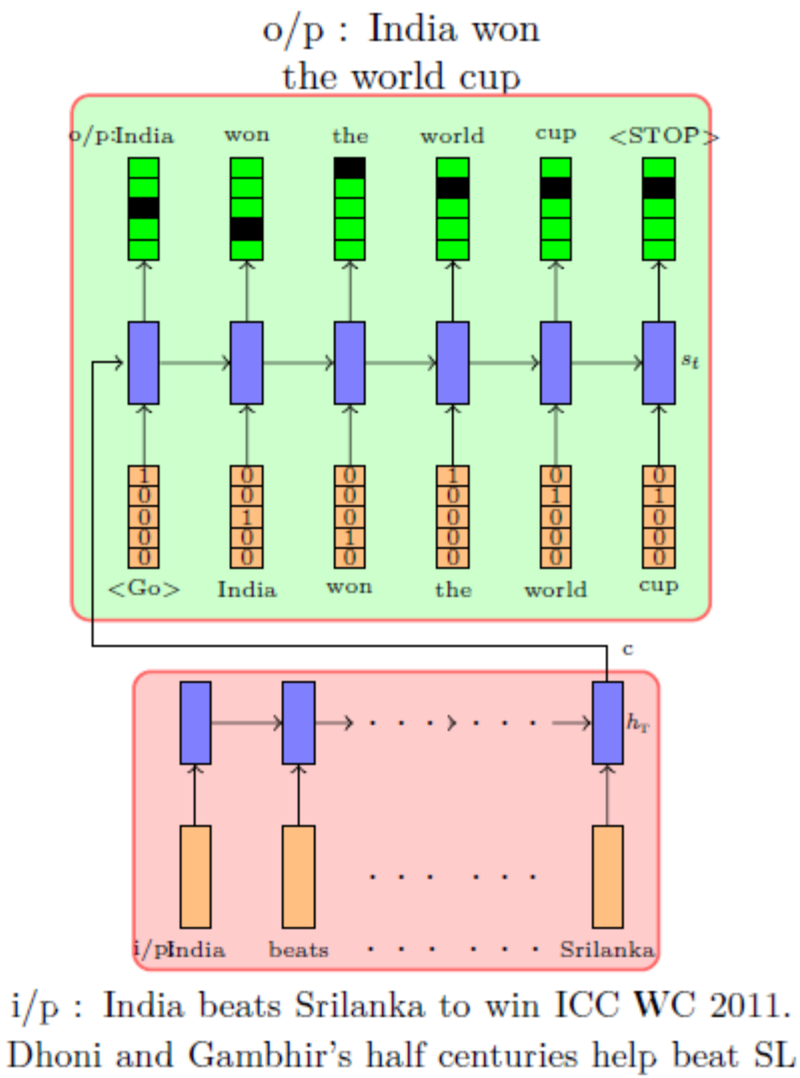
- **Data:** $\{x_i = \{I, q\}_i, y_i = Answer\}_{i=1}^{N}$
- **Model:**
  - **Encoder:**
    $$\hat{h}_I = CNN(I), \ \tilde{h}_t = RNN(\tilde{h}_{t-1}, q_{it})$$
    $$s = [\tilde{h}_T; \hat{h}_I]$$
  - **Decoder:**
    $$P(y|q, I) = softmax(Vs + b)$$
- **Parameters:** $V, b, U_q, W_q, W_{conv}, b$
- **Loss:**
  $$\mathscr{L}(\theta) = -\log P(y = \ell | I, q)$$
- **Algorithm: Gradient descent with backpropagation**

Task: Document Summarization

o/p : India won the world cup

i/p : India beats Srilanka to win ICC WC 2011.
Dhoni and Gambhir's half centuries help beat SL

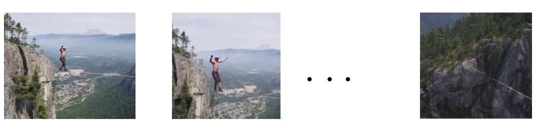
Task: Document Summarization



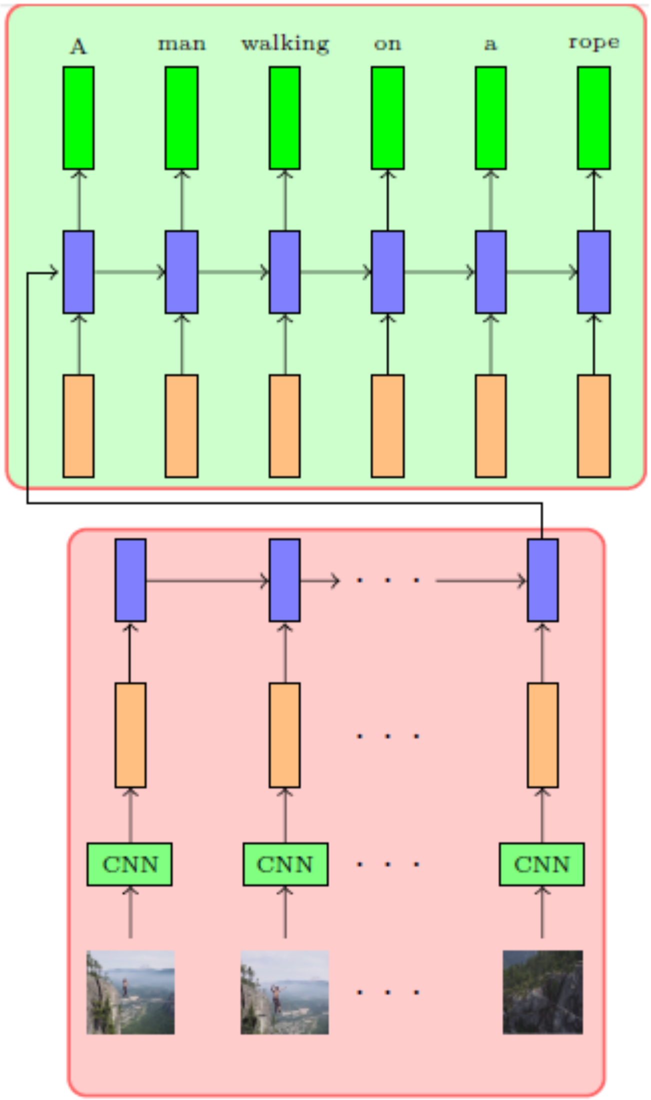
- **Data:** $\{x_i = Document_i,\ y_i = Summary_i\}_{i=1}^{N}$
- **Model:**
  - Encoder:
    $$h_t = RNN(h_{t-1}, x_{it})$$
  - Decoder:
    $$s_0 = h_T$$
    $$s_t = RNN(s_{t-1}, e(\hat{y}_{t-1}))$$
    $$P(y_t | y_1^{t-1}, x) = softmax(Vs_t + b)$$
- **Parameters:** $U_{dec},\ V,\ W_{dec},\ U_{enc},\ W_{enc},\ b$
- **Loss:**
  $$\mathscr{L}(\theta) = \sum_{i=1}^{T} \mathscr{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t | y_1^{t-1}, x)$$
- **Algorithm: Gradient descent with backpropagation**

Task: Video Captioning

o/p : A man walking on a rope



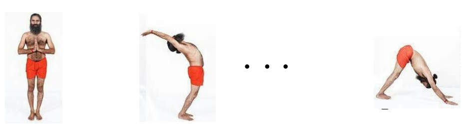
Task: Video Captioning



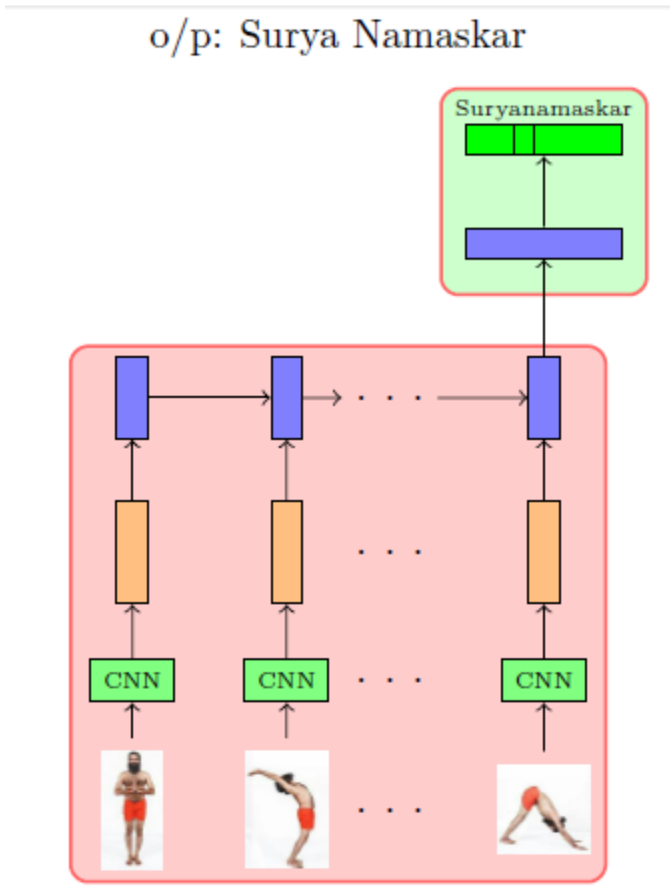
- **Data:** $\{x_i = video_i,\ y_i = desc_i\}_{i=1}^{N}$
- **Model:**
  - Encoder:
    $$h_t = RNN(h_{t-1}, CNN(x_{it}))$$
  - Decoder:
    $$s_0 = h_T$$
    $$s_t = RNN(s_{t-1}, e(\hat{y}_{t-1}))$$
    $$P(y_t | y_1^{t-1}, x) = softmax(Vs_t + b)$$
- **Parameters:** U_{dec}, W_{dec}, V, b, W_{conv}, U_{enc}, W_{enc}, b
- **Loss:**
  $$\mathscr{L}(\theta) = \sum_{i=1}^{T} \mathscr{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t | y_1^{t-1}, x)$$
- **Algorithm: Gradient descent with backpropagation**

Task: Video Classification

o/p: Surya Namaskar



Task: Video Classification



- **Data:** $\{x_i = Video_i,\ y_i = Activity_i\}_{i=1}^{N}$
- **Model:**
  - Encoder:

$$h_t = RNN(h_{t-1}, CNN(x_{it}))$$

  - Decoder:

$$s = h_T$$

$$P(y|I) = softmax(Vs + b)$$

- **Parameters:** V, b, W_{conv}, U_{enc}, W_{enc}, b
- **Loss:**

$$\mathscr{L}(\theta) = -\log P(y = \ell | Video)$$

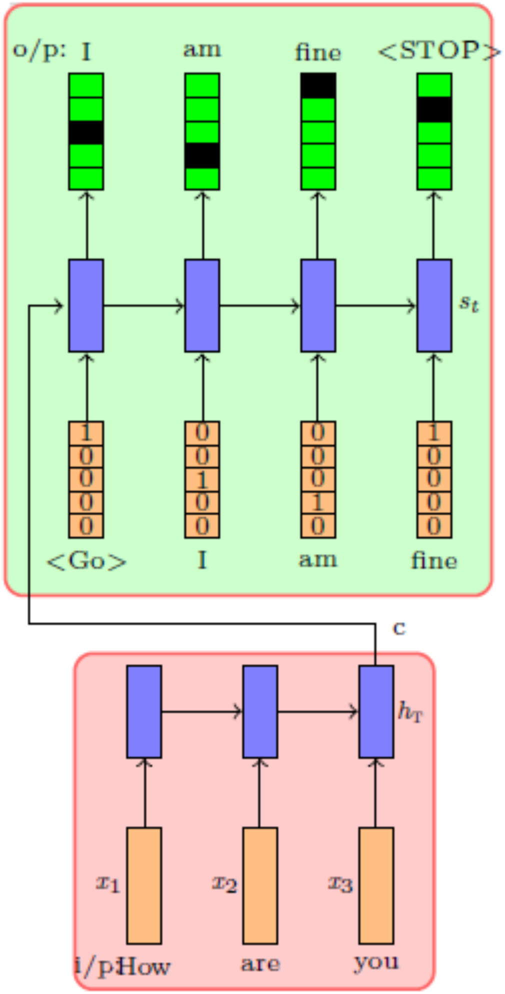
- **Algorithm: Gradient descent with backpropagation**

Task: Dialog

o/p: I am fine

i/p: How are you

Task: Dialog



i/p: How are you

- **Data:** $\{x_i = Utterance_i, y_i = Response_i\}_{i=1}^{N}$
- **Model:**
  - Encoder:

$$h_t = RNN(h_{t-1}, x_{it})$$

  - Decoder:

$$s_0 = h_T \quad (T \text{ is length of input})$$

$$s_t = RNN(s_{t-1}, e(\hat{y}_{t-1}))$$

$$P(y_t|y_1^{t-1}, x) = softmax(Vs_t + b)$$

- **Parameters:** U_{dec}, V, W_{dec}, U_{enc}, W_{enc}, b
- **Loss:**

$$\mathscr{L}(\theta) = \sum_{i=1}^{T} \mathscr{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t | y_1^{t-1}, x)$$

- **Algorithm: Gradient descent with backpropagation**