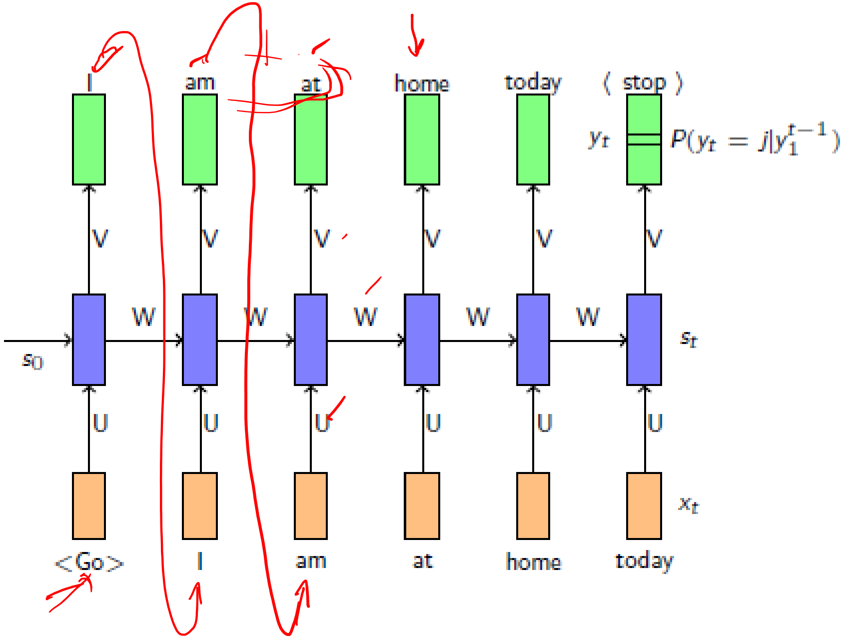
RNN

Encoder-decoder models

EE5179: Deep Learning for Imaging

Beyond RNN



- We will start by revisiting the problem of language modeling
- 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

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

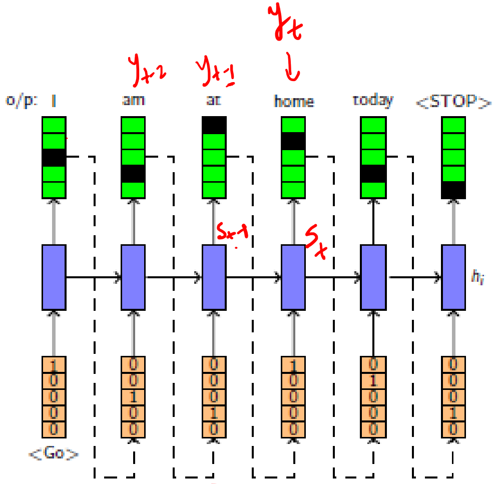
- Let us see how we model $P(y_t|y_1, y_2...y_{t-1})$ using a RNN

Beyond RNN

$P(y_t | y_1 \ldots y_{t-1}) = \text{softmax}(V s_t + c)$

$s_t = f(W s_{t-1} + U x_t)$

$x_t \to y_{t-1}$



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

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

$$s_i = RNN(s_{i-1}, x_i)$$

- Let j be the index of the word which has been assigned the max probability at time step $t-1$

$$x_i = e(v_j)$$

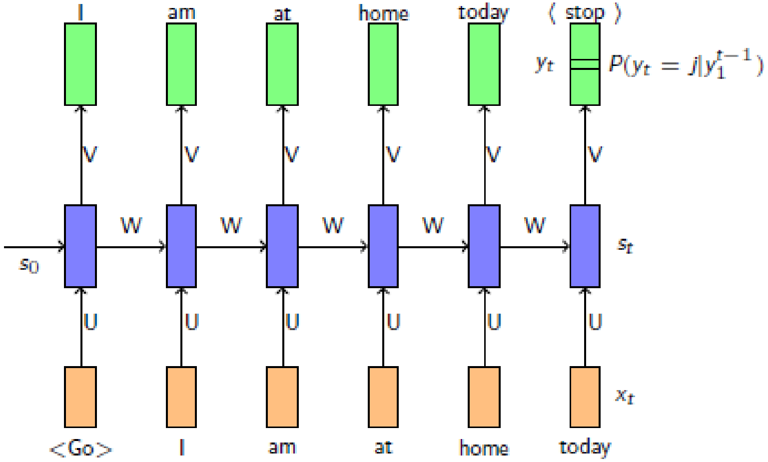
- x_i is essentially a one-hot vector ($e(v_j)$)representing the j^{th} word in the vocabulary
- In practice, instead of one hot representation we use a pre-trained word embedding of the j^{th} word

Beyond RNN

$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \to$ home

$p = \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_k \end{pmatrix}$ GT $\quad q = \begin{pmatrix} q_1 \\ \vdots \\ q_k \end{pmatrix}$

Cross entropy $= -\sum_i p_i \ln q_i$ $\quad$ $- \ln q_i$



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

$y_1, y_2, \ldots y_{t-1}$

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

- **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 t

NLL

min $-\log P(y_t | \ldots)$

max $\log P(y_t | \ldots)$

max $P(y_t | \ldots)$

ML

max likelihood

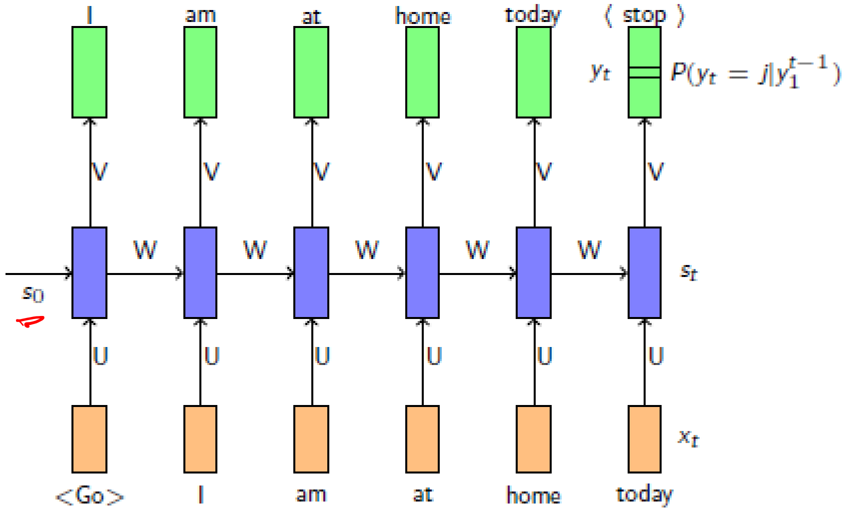
Data:

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

Beyond RNN

start
Go

stop



Q. But what about s_0 ?

A. Generally it is randomly initialised and updated using backpropagation along with U,V and W.

Q. Are there are smarter ways of supplying s_0 ?

Yes!. It is a class of architectures called :

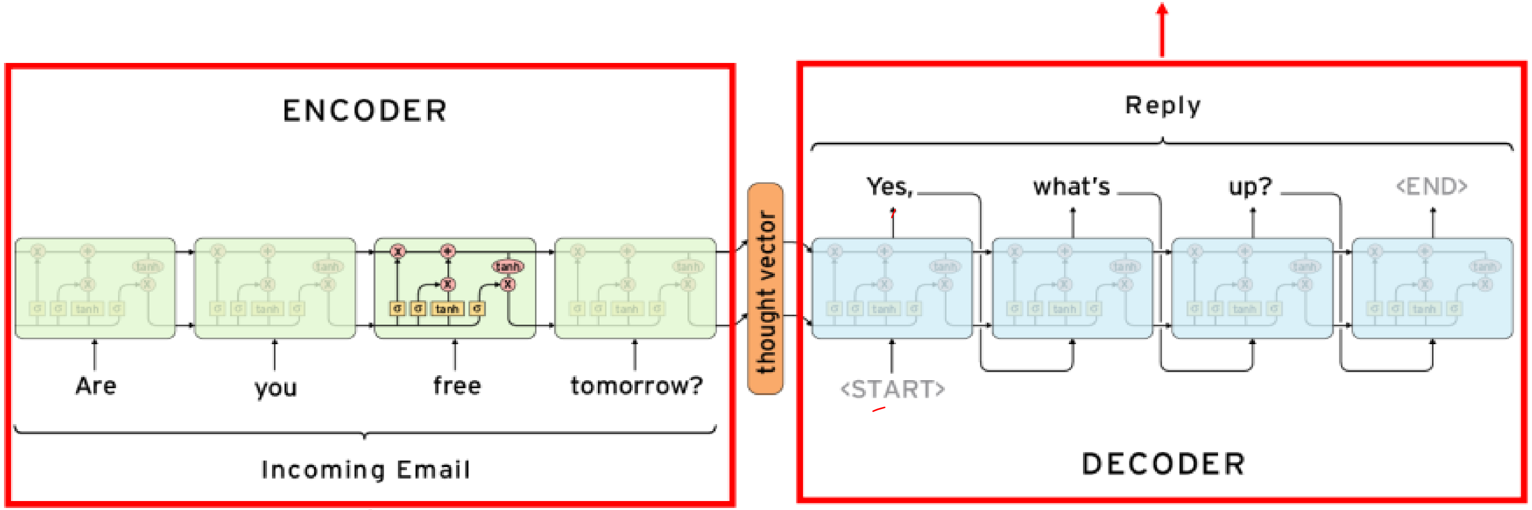
Encoder-Decoder Models

Sequence-to-Sequence

Encoder-Decoder Framework [Sutskever et al. 2014, Cho et al. 2014]

Output Sequence

$$\{y_1, y_2, \ldots, y_m\}$$



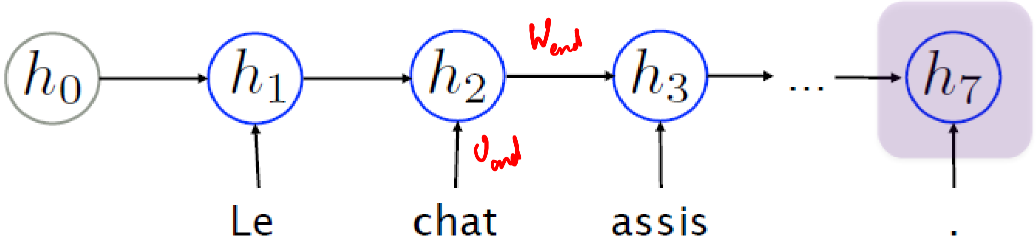
$$\{x_1, x_2, \ldots, x_n\}$$

Input Sequence

RNN Encoder

Idea:

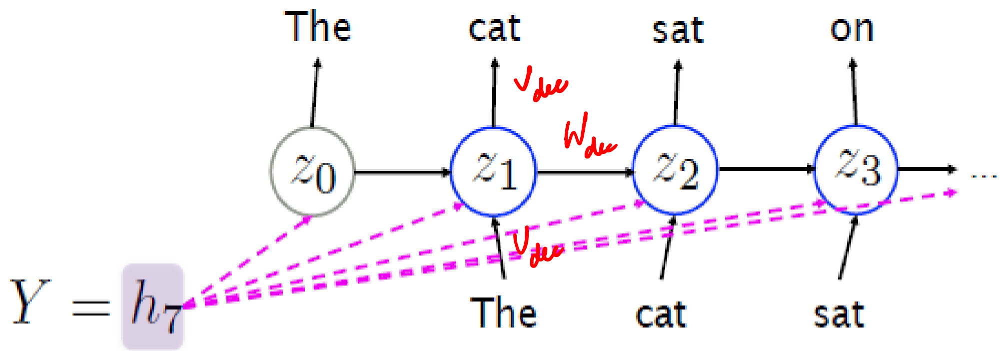
- Read a source sentence one symbol at a time.
- The last hidden state summarizes the entire source sentence.



RNN Decoder

Usual recurrent language model, except

Transition $z_t = f(z_{t-1}, x_t, Y)$



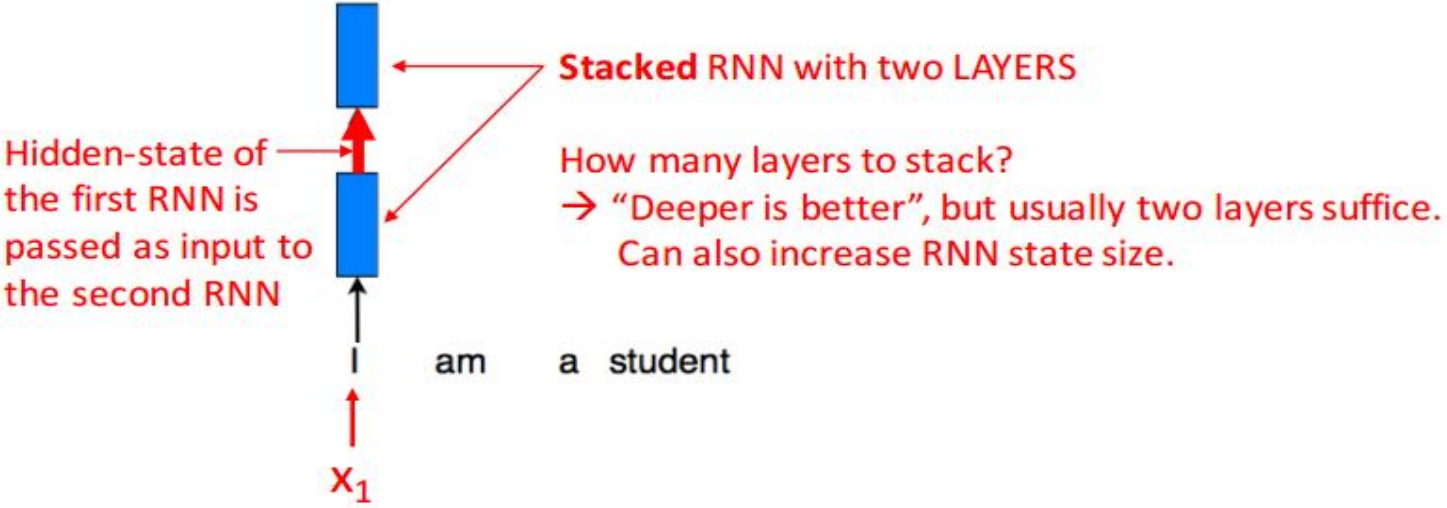
Same learning strategy as usual: MLE with SGD

$W_{enc}, V_{enc}, W_{dec}, V_{dec}, V'_{dec}$

$$\mathcal{L}(\theta, D) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T^n} \log p(x_t^n | x_1^n, \ldots, x_{t-1}^n, Y)$$

Sequence-to-Sequence

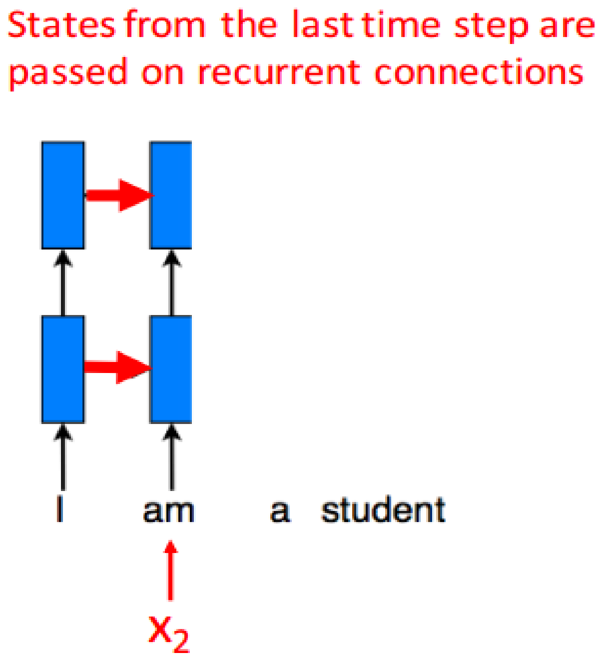
Encoder-Decoder Framework :: Walk Through



Note: The inputs (x) can be characters / words / other feature-vectors.
(more on this later)

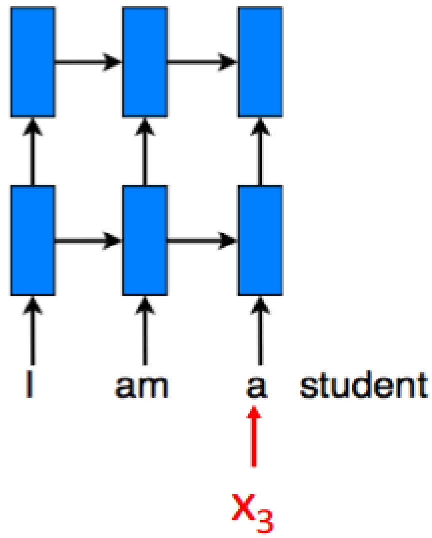
Sequence-to-Sequence

Encoder-Decoder Framework :: Walk Through



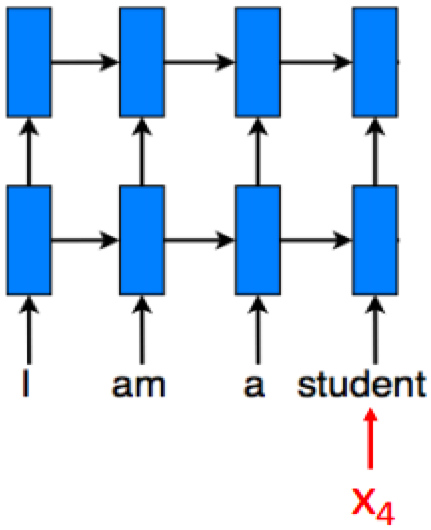
Sequence-to-Sequence

Encoder-Decoder Framework :: Walk Through



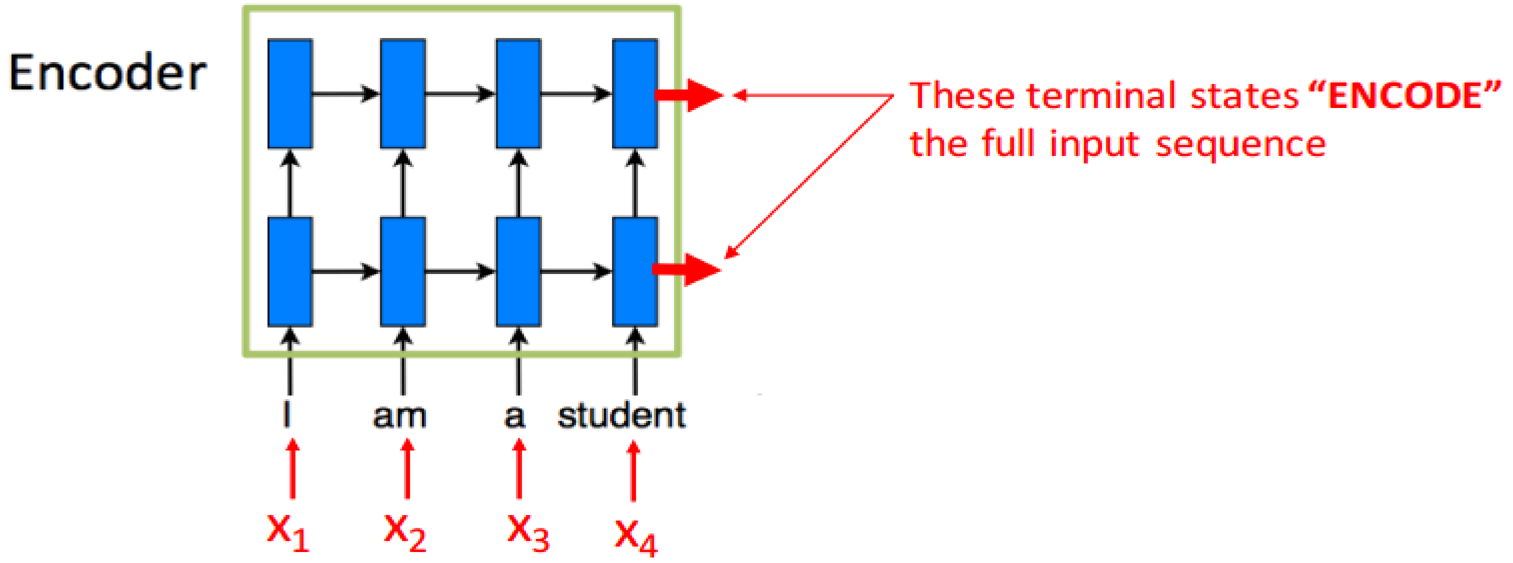
Sequence-to-Sequence

Encoder-Decoder Framework :: Walk Through



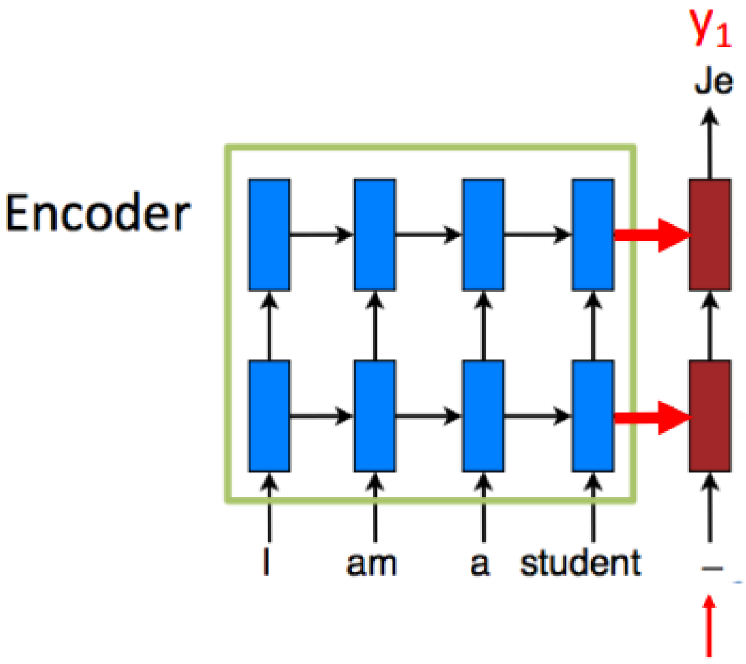
Sequence-to-Sequence

Encoder-Decoder Framework :: Walk Through



Sequence-to-Sequence

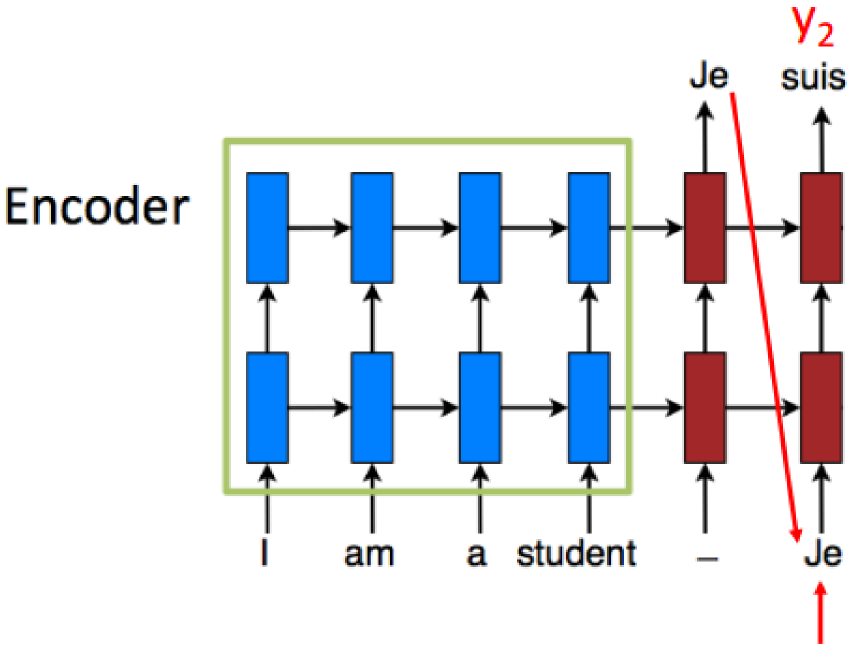
Encoder-Decoder Framework :: Walk Through



"START" token inserted in the first step when decoding

Sequence-to-Sequence

Encoder-Decoder Framework :: Walk Through

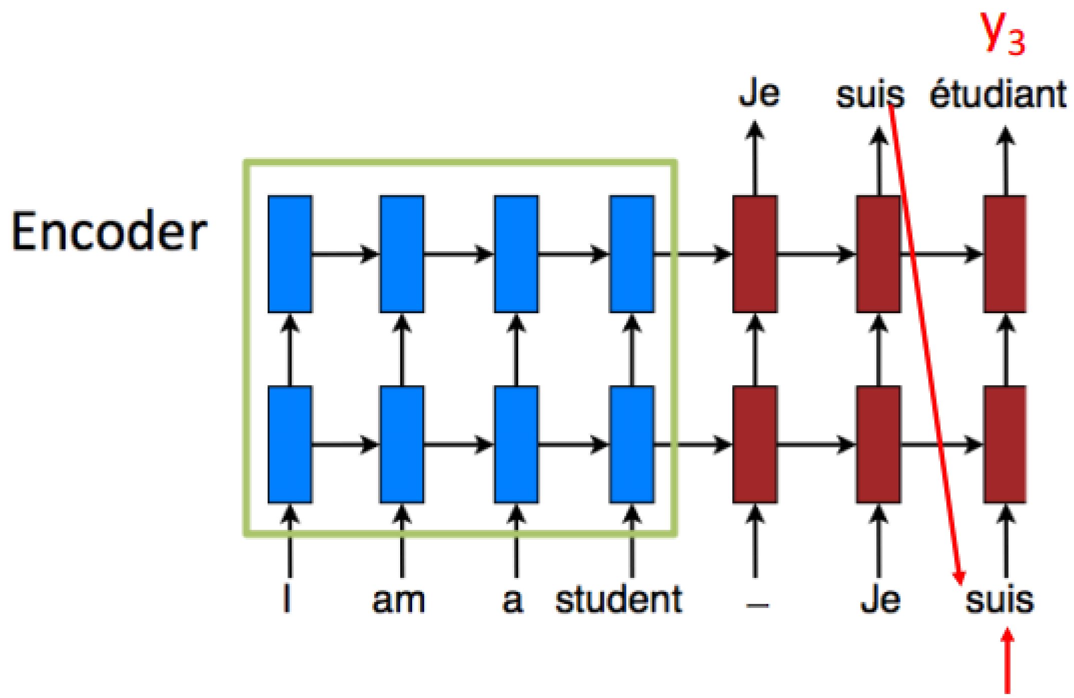


$$\boxed{p(y_{t+1}|y_1, \ldots, y_t)}$$

The output from the previous step is fed back
Into the decoder – as the RNN models the
"conditional" one-step distribution

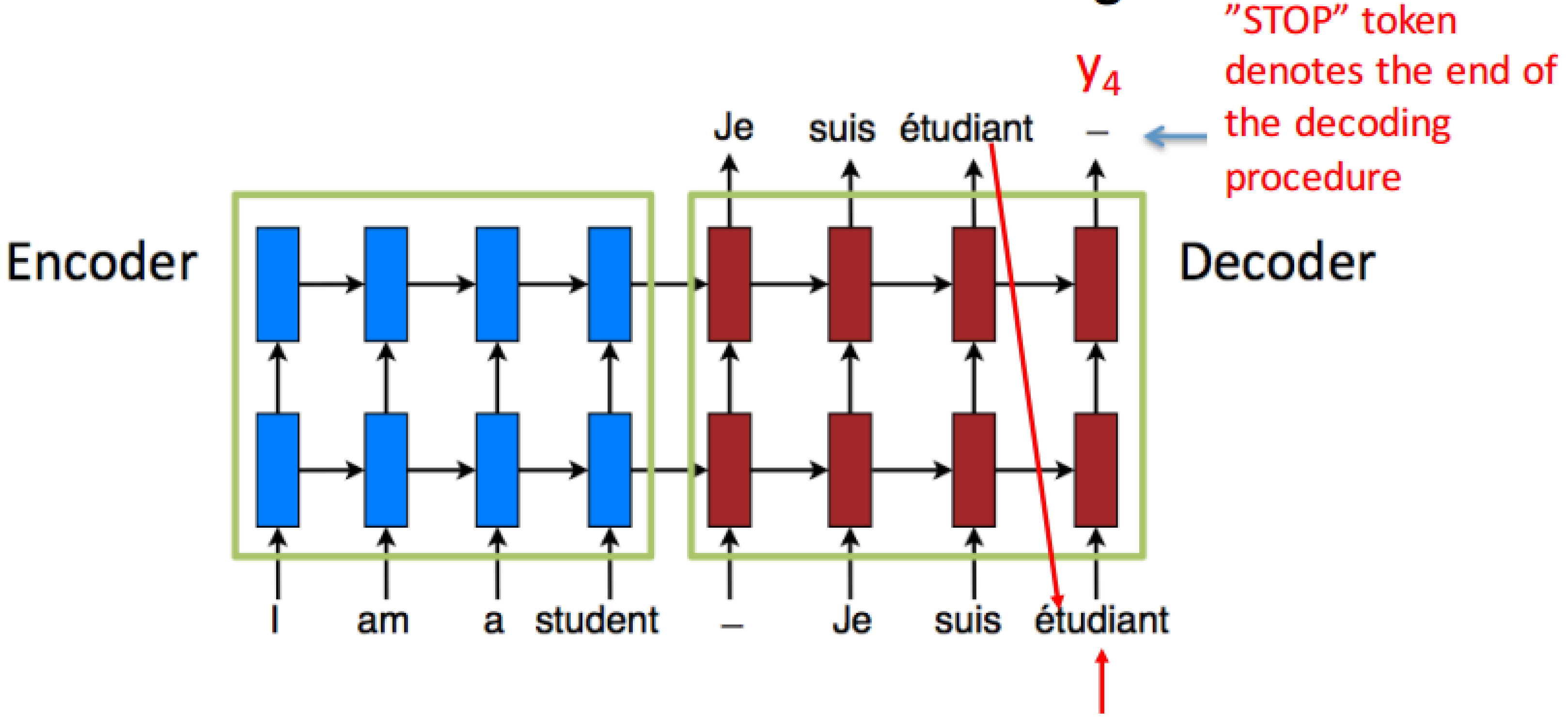
Sequence-to-Sequence

Encoder-Decoder Framework :: Walk Through

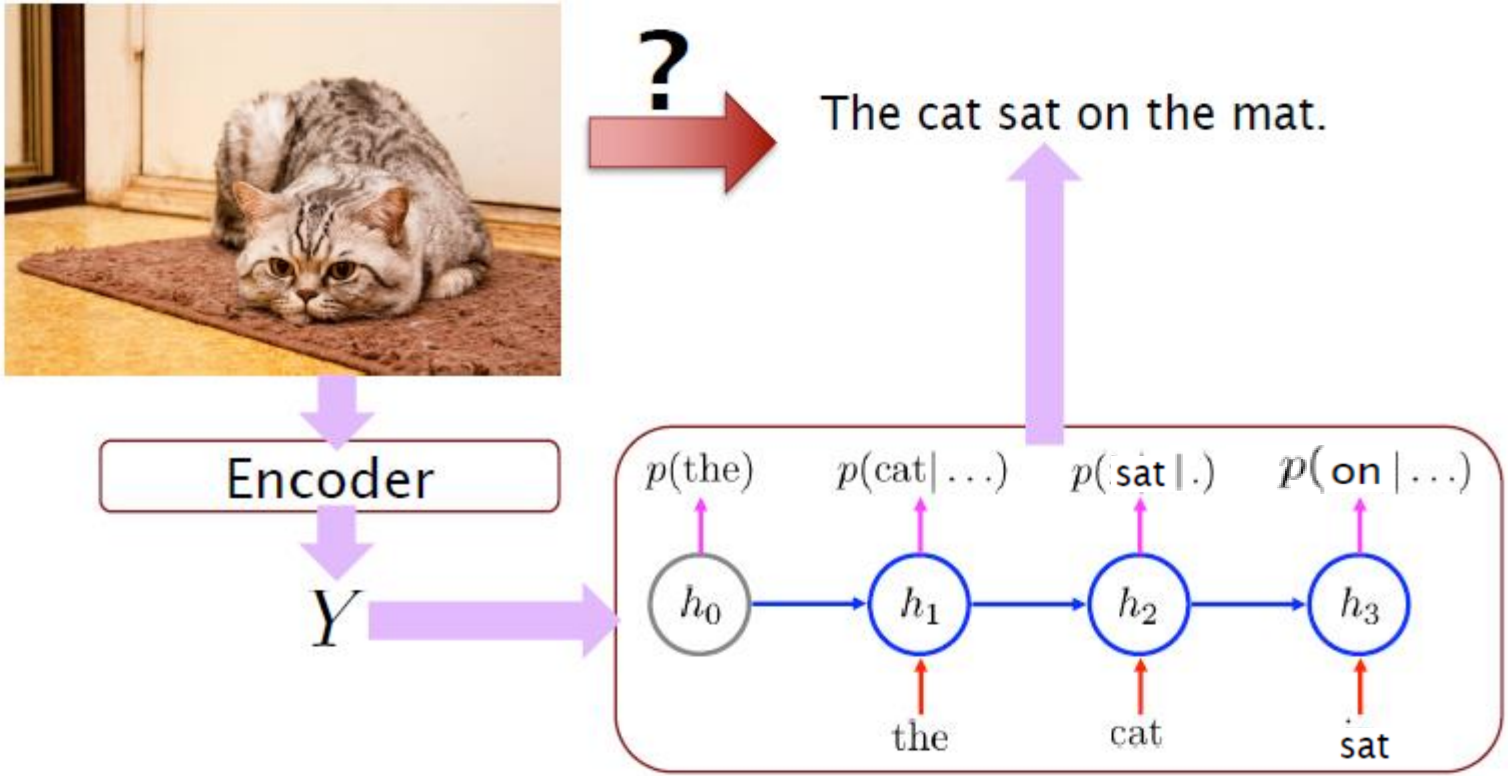


Sequence-to-Sequence

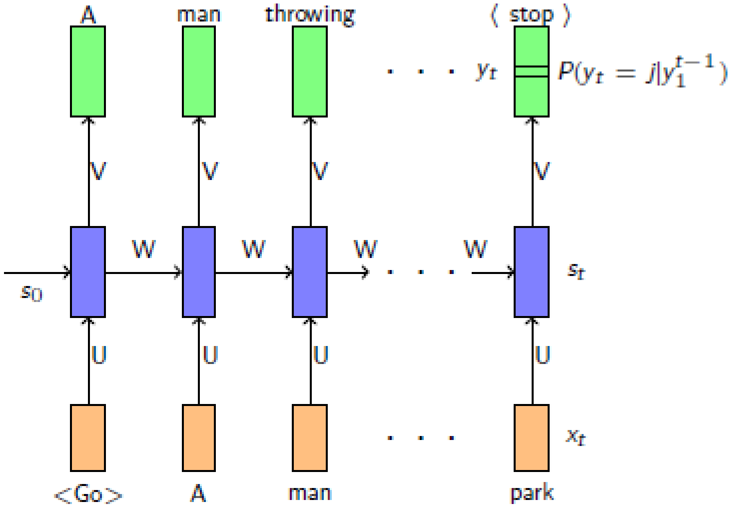
Encoder-Decoder Framework :: Walk Through

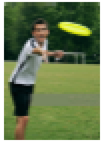


Also, Can we do this?



Yes!

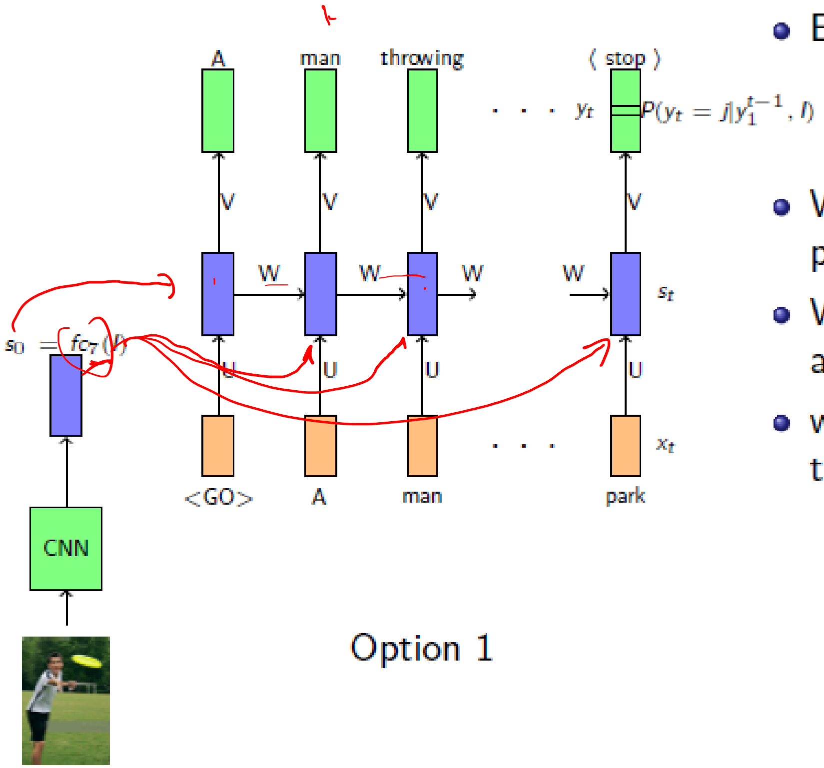




A man throwing
a frisbee in a park

- 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 $P(y_t|y_1^{t-1}, I)$ instead of $P(y_t|y_1^{t-1})$ where I is an image
- Notice that $P(y_t|y_1^{t-1}, I)$ is again a conditional distribution

Encoder Decoder Models



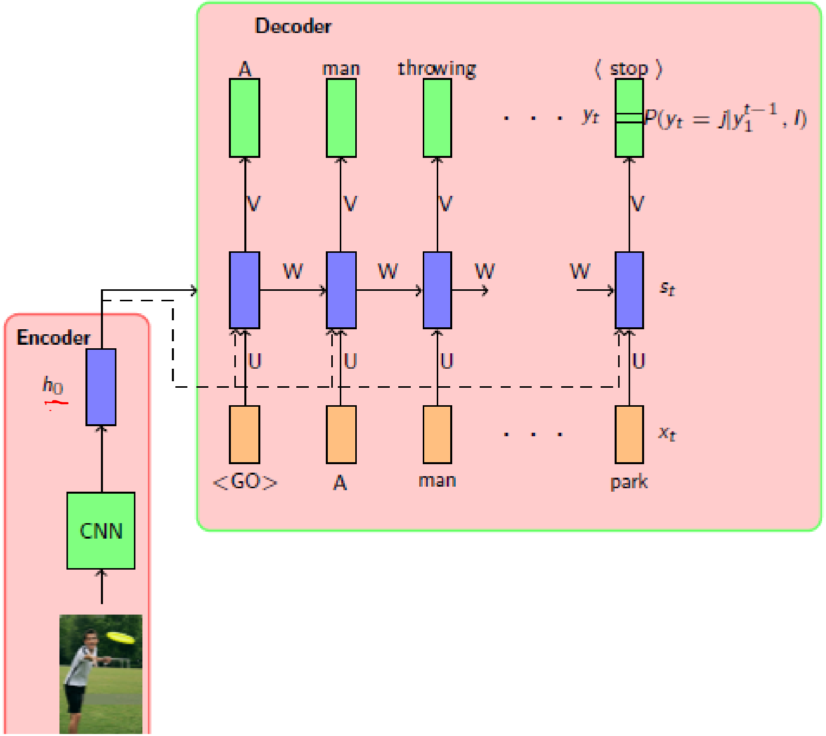
Option 1

- 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_i was a state capturing all the previous words
- We could now model $P(y_t = j|y_1^{t-1}, I)$ as $P(y_t = j|s_k, I)$
- where $fc_7(I)$ is the representation obtained from the fc_7 layer of an image

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

Encoder Decoder Models

Such neural encoder decoder architectures have become extremely popular, and they are being used to model a wide variety of tasks!

- 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 ?

Quick Recap of Notations

$h_{t-1} = o_t \odot s_{t-1}$

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

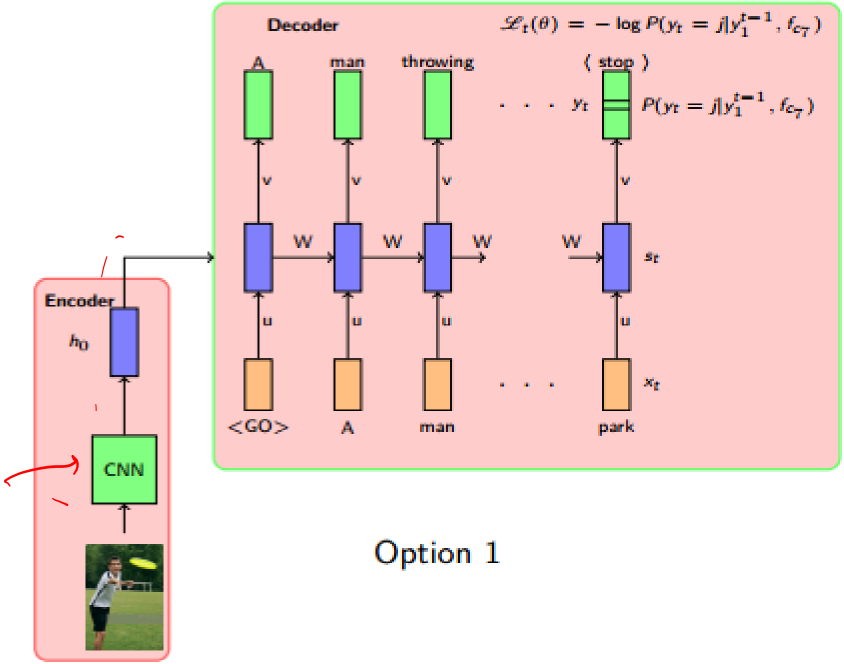
$$\tilde{s}_t = \sigma(W(o_t \odot s_{t-1}) + U x_t + b)$$
$$s_t = i_t \odot s_{t-1} + (1 - i_t) \odot \tilde{s}_t$$

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

$$s_t = \mathrm{RNN}(s_{t-1}, x_t) \qquad s_t = \mathrm{GRU}(s_{t-1}, x_t) \qquad s_t = \mathrm{LSTM}(h_{i-1}, s_{i-1}, x_i)$$

- 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: Image captioning

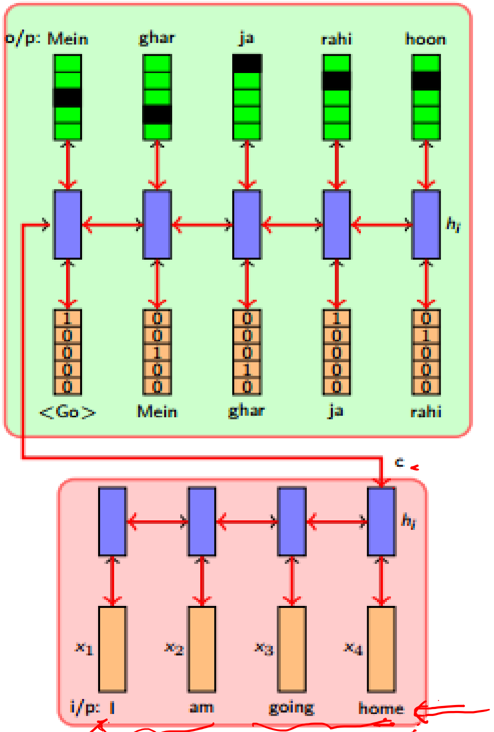


Option 1

- **Data:** $\{x_i = image_i, \ y_i = caption_i\}_{i=1}^{N}$
- **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:** $U_{dec}, V, W_{dec}, W_{conv}$
- **Loss:**
$$\mathscr{L}(\theta) = \sum_{i=1}^{T} \mathscr{L}_t(\theta) \quad = -\sum_{t=1}^{T} \log P(y_t = \ell_t | y_1^{t-1}, I)$$
- **Algorithm:** Gradient descent with backpropagation

Encoder Decoder Models: Machine translation

o/p : Mein ghar ja rahi hoon



i/p : I am going home

- **Data:** $\{x_i = source_i,\ y_i = target_i\}_{i=1}^{N}$
- **Model (Option 1):**
  - **Encoder:**

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

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

  - **Decoder:**

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

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

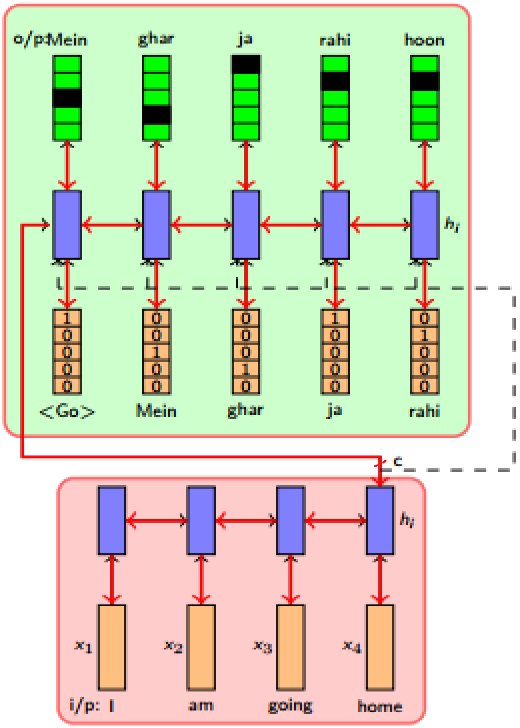
- **Parameters:** U_{dec}, V, W_{dec}, U_{enc}, W_{enc}
- **Loss:**

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

- **Algorithm:** Gradient descent with backpropogation

Encoder Decoder Models: Machine translation

o/p : Mein ghar ja rahi hoon



i/p : I am going home

- **Data:** $\{x_i = source_i,\ y_i = target_i\}_{i=1}^{N}$
- **Model (Option 2):**
  - **Encoder:**

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

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

  - **Decoder:**

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

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

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

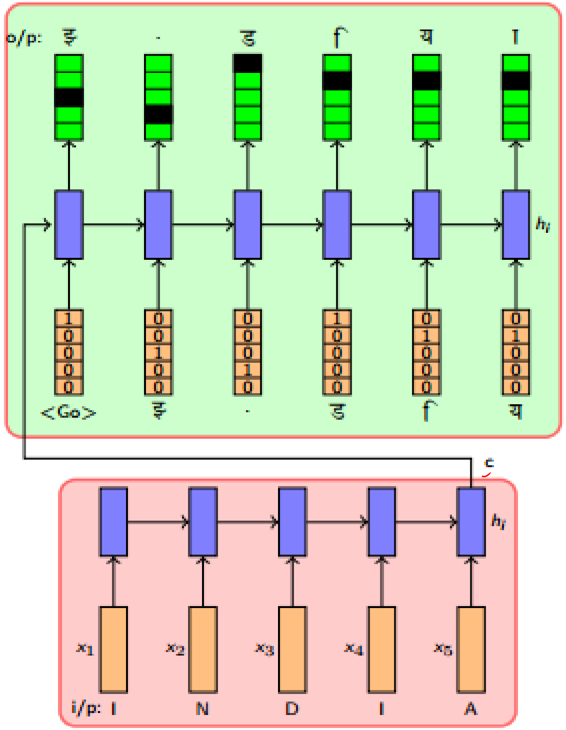
  $$\mathscr{L}(\theta) = \sum_{i=1}^{T} \mathscr{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t | y_1^{t-1}, \mathbf{x})$$

- **Algorithm:** Gradient descent with backpropogation

Encoder Decoder Models: Source-word to target-word

o/p : इ . ड ि य ा

44x1



a
b
c
26x1

i/p : I N D I A

- **Data:** $\{x_i = srcword_i,\ y_i = tgtword_i\}_{i=1}^{N}$
- **Model (Option 1):**
  - **Encoder:**

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

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

  - **Decoder:**

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

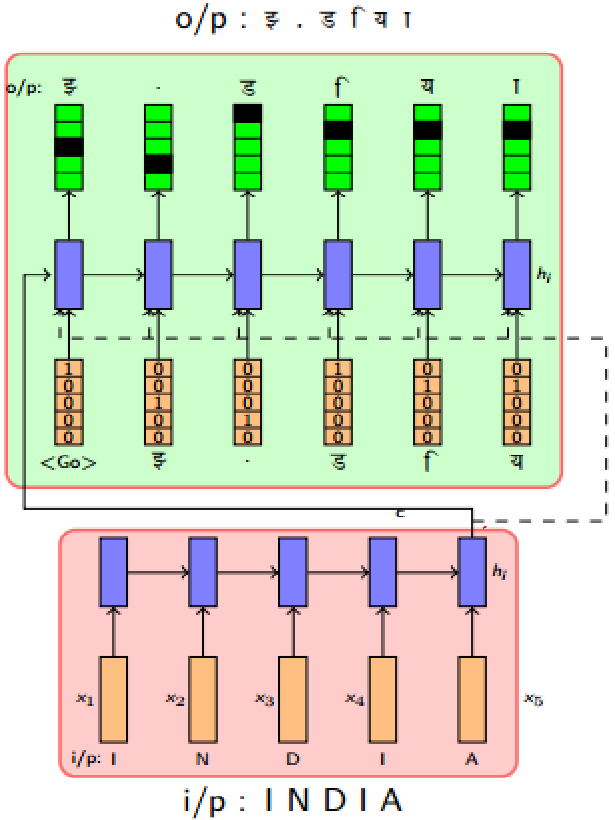
    $$P(y_t | y_1^{t-1}, \mathbf{x}) = softmax(Vs_t + b)$$

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

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

- **Algorithm:** Gradient descent with backpropogation

Encoder Decoder Models: Source-word to target-word



- **Data:** $\{x_i = srcword_i,\ y_i = tgtword_i\}_{i=1}^{N}$
- **Model (Option 2):**
  - **Encoder:**

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

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

  - **Decoder:**

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

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

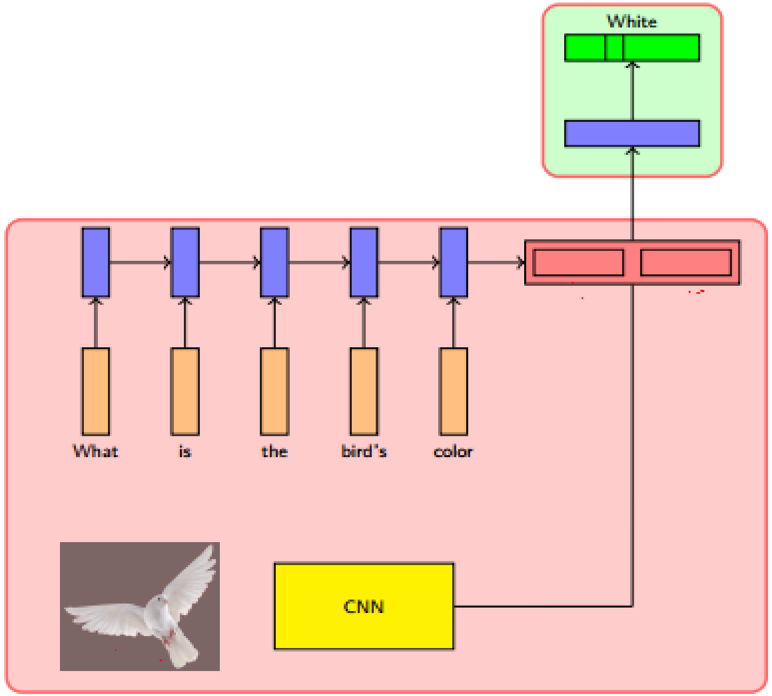
- **Parameters:** U_{dec}, V, W_{dec}, U_{enc}, W_{enc}
- **Loss:**

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

- **Algorithm:** Gradient descent with backpropogation

Encoder Decoder Models: Query based image captioning

O/p: White



Question: What is the bird's color

- **Data:** $\{x_i = \{I, Q\}_i,\ y_i = Answer_i\}_{i=1}^{N}$
- **Model:**
  - **Encoder:**

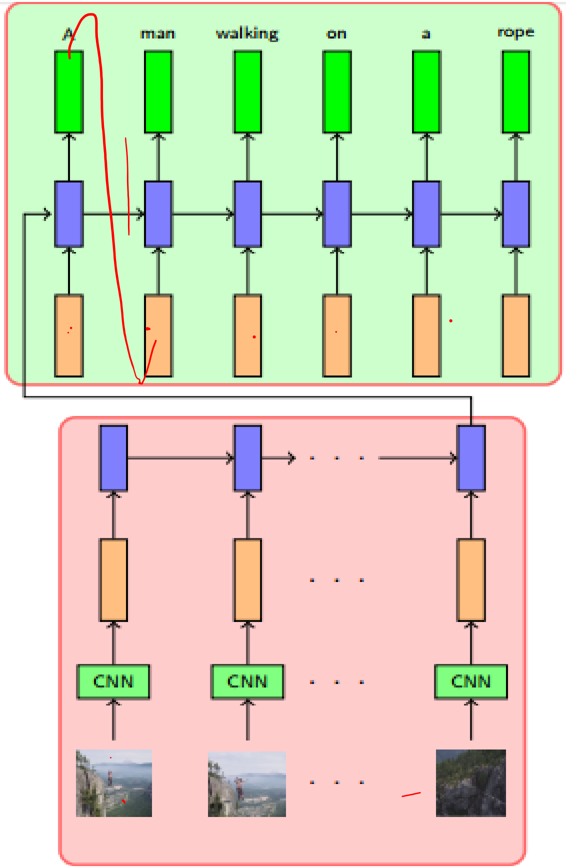
    $$\hat{h}_I = CNN(I),\ \ \tilde{h}_t = RNN(h_{t-1}, q_t)$$

    $$s = [\hat{h}_I; \tilde{h}_T]$$
  - **Decoder:**

    $$P(y|\mathbf{q}, I) = softmax(Vs + b)$$
- **Parameters:** $V,\ b,\ U_q,\ W_q,\ W_{conv}$
- **Loss:**

  $$\mathscr{L}(\theta) = -\log P(y = \ell | I, \mathbf{q})$$
- **Algorithm:** Gradient descent with backpropagation

Encoder Decoder Models: Video captioning



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

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

$$s_0 = h_T$$

  - **Decoder:**

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

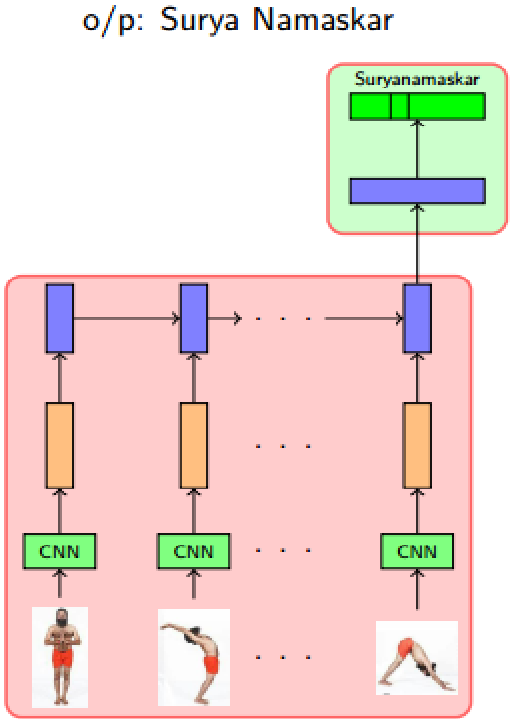
$$P(y_t|y_1^{t-1}, \mathbf{x}) = softmax(Vs_t + b)$$

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

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

- **Algorithm:** Gradient descent with backpropogation

Encoder Decoder Models: Activity recognition from video



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

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

$$s = h_T$$

  - **Decoder:**

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

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

$$\mathscr{L}(\theta) = \sum_{i=1}^{T} \mathscr{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y = \ell | I)$$

- **Algorithm:** Gradient descent with backpropagation

Encoder Decoder Models

Conclusions:

- And the list continues …
- Try picking a problem from your domain and see if you can model it using the encoder decoder paradigm

Extensions:

- Train stacked/deep RNNs with multiple layers
- Potentially train bidirectional encoder
- Train input sequence in reverse order to suit the task:
  (optimization problem: Instead of A B C -> X Y, train C B A -> X Y)