CS7015 (Deep Learning): Lecture 16
Encoder Decoder Models, Attention Mechanism

Mitesh M. Khapra

Department of Computer Science and Engineering
Indian Institute of Technology Madras
Module 16.1: Introduction to Encoder Decoder Models
We will start by revisiting the problem of language modeling.
- We will start by revisiting the problem of language modeling.
- Informally, given ‘t – i’ words we are interested in predicting the $t^{th}$ word.
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

$$y^* = \text{argmax } P(y_t | y_1, y_2, ..., y_{t-1})$$
We will start by revisiting the problem of language modeling.

Informally, given ‘t – i’ words we are interested in predicting the t\textsuperscript{th} word.

More formally, given \(y_1, y_2, ..., y_{t-1}\) we want to find

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

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

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

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: \( P(y_t|y_1^{t-1}) \)
We will start by revisiting the problem of language modeling.

Informally, given ‘t – i’ words we are interested in predicting the \( y^* \)th word.

More formally, given \( y_1, y_2, ..., y_{t-1} \) we want to find

\[
y^* = \arg\max \ P(y_t|y_1, y_2, ..., y_{t-1})
\]

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: \( P(y_t|y_{t-1}^{t-1}) \)
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

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

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: $P(y_t|y_1^{t-1})$
We will start by revisiting the problem of language modeling.

Informally, given ‘$t - i$’ words we are interested in predicting the $i^{th}$ word.

More formally, given $y_1, y_2, ..., y_{t-1}$ we want to find:

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

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: $P(y_t | y_{t-1})$.
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 \( y_1, y_2, \ldots, y_{t-1} \) we want to find

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

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: \( P(y_t | y_{t-1}^{t-1}) \)
We will start by revisiting the problem of language modeling.

Informally, given ‘t – i’ words we are interested in predicting the t\textsuperscript{th} word.

More formally, given \(y_1, y_2, \ldots, y_{t-1}\) we want to find:

\[ y^* = \arg\max P(y_t|y_1, y_2, \ldots, y_{t-1}) \]

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: \(P(y_t|y_{1}^{t-1})\)
We will start by revisiting the problem of language modeling.

Informally, given ‘$t-i$’ words we are interested in predicting the $i^{th}$ word.

More formally, given $y_1, y_2, ..., y_{t-1}$ we want to find

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

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: $P(y_t|y_1^{t-1})$
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 $y_1, y_2, \ldots, y_{t-1}$ we want to find

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

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

\[ P(y_t = j | y_1^{t-1}) = \text{softmax}(V s_t + c)_j \]
We are interested in

\[ P(y_t = j | y_{1:t-1}) \]

where \( j \in V \) and \( V \) is the set of all vocabulary words.

Using an RNN we compute this as

\[ P(y_t = j | y_{1:t-1}) = \text{softmax}(V s_t + c)_j \]

In other words we compute

\[ P(y_t = j | y_{1:t-1}) = P(y_t = j | s_t) = \text{softmax}(V s_t + c)_j \]
We are interested in

\[ P(y_t = j | y_1^{t-1}) \]

where \( j \in V \) and \( V \) is the set of all vocabulary words.

Using an RNN we compute this as

\[ P(y_t = j | y_1^{t-1}) = \text{softmax}(V_s + c)_j \]

In other words we compute

\[ P(y_t = j | y_1^{t-1}) = P(y_t = j | s_t) \]

\[ = \text{softmax}(V_s + c)_j \]

Notice that the recurrent connections ensure that \( s_t \) has information about \( y_1^{t-1} \).
Data: All sentences from any large corpus (say wikipedia)

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:

\[
P(y_t = j | y_{t-1}^t) = \text{softmax}(V_x + c)
\]

Parameters:
\[
U; V; W; b; c
\]

Loss:
\[
L_t = \log P(y_t = \ell | y_{1:t-1})
\]

where \(\ell_t\) is the true word at time step \(t\)

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:

\[ s_t = \sigma(Ws_{t-1} + Ux_t + b) \]

\[ P(y_t = j|y_1^{t-1}) = \text{softmax}(Vs_t + c)_j \]

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**:
  \[ s_t = \sigma(Ws_{t-1} + Ux_t + b) \]
  \[ P(y_t = j|y_{t-1}^t) = \text{softmax}(Vs_t + c)_j \]
- **Parameters**: \( U, V, W, b, c \)

---

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:**

\[ s_t = \sigma(Ws_{t-1} + Ux_t + b) \]

\[ P(y_t = j | y_{1:t-1}^t) = \text{softmax}(Vs_t + c)_j \]

- **Parameters:** \( U, V, W, b, c \)

- **Loss:**

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:

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

Parameters: \( U, V, W, b, c \)

Loss:

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

where \( \ell_t \) is the true word at time step \( t \)

India, officially the Republic of India, is a country in South Asia. It is the seventh-largest country by area, .....
What is the input at each time step?

It is simply the word that we predicted at the previous time step.

In general, $s_t = \text{RNN}(s_{t-1}; x_t)$.

Let $j$ be the index of the word which has been assigned the max probability at time step $t$. $x_t = e(v_j)$.

$x_t$ 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.
What is the input at each time step?
What is the input at each time step?

It is simply the word that we predicted at the previous time step.

In general, $s_t = \text{RNN}(s_{t-1}; x_t)$.

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

$x_t = e(v_j)$, where $x_t$ 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.
- What is the input at each time step?
- It is simply the word that we predicted at the previous time step
What is the input at each time step?

It is simply the word that we predicted at the previous time step.

- o/p: I am at home
- \( s_t = \text{RNN}(s_{t-1}; x_t) \)
- Let \( j \) be the index of the word which has been assigned the max probability at time step \( t \)
- \( x_t = e(v_j) \)
- \( x_t \) 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
What is the input at each time step?
It is simply the word that we predicted at the previous time step.
What is the input at each time step?

It is simply the word that we predicted at the previous time step.
- What is the input at each time step?
- It is simply the word that we predicted at the previous time step
- In general
  \[ s_t = RNN(s_{t-1}, x_t) \]
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 \( t - 1 \)
  \[ x_t = e(v_j) \]
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.
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.
Notice that $s_0$ is not computed but just randomly initialized.
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).
- 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
Before moving on we will see a compact way of writing the function computed by RNN, GRU and LSTM.
\[ s_t = \sigma(U x_t + W s_{t-1} + b) \]

- Before moving on we will see a compact way of writing the function computed by RNN, GRU and LSTM
\[ s_t = \sigma(U x_t + W s_{t-1} + b) \]

\[ s_t = \text{RNN}(s_{t-1}, x_t) \]

- Before moving on we will see a compact way of writing the function computed by RNN, GRU and LSTM.
Before moving on we will see a compact way of writing the function computed by RNN, GRU and LSTM.
\[ s_t = \sigma(U x_t + Ws_{t-1} + b) \]
\[ \tilde{s}_t = \sigma(W(o_t \odot s_{t-1}) + Ux_t + b) \]
\[ s_t = i_t \odot s_{t-1} + (1 - i_t) \odot \tilde{s}_t \]

\[ s_t = \text{RNN}(s_{t-1}, x_t) \]
\[ s_t = \text{GRU}(s_{t-1}, x_t) \]

- Before moving on we will see a compact way of writing the function computed by RNN, GRU and LSTM
Before moving on we will see a compact way of writing the function computed by RNN, GRU and LSTM.
\[ s_t = \sigma(U x_t + W s_{t-1} + b) \]
\[ \tilde{s}_t = \sigma(W(o_t \odot s_{t-1}) + U x_t + b) \]
\[ s_t = i_t \odot s_{t-1} + (1 - i_t) \odot \tilde{s}_t \]
\[ h_t = o_t \odot \sigma(s_t) \]

\[ s_t = \text{RNN}(s_{t-1}, x_t) \]
\[ s_t = \text{GRU}(s_{t-1}, x_t) \]
\[ h_t, s_t = \text{LSTM}(h_{t-1}, s_{t-1}, x_t) \]

- Before moving on we will see a compact way of writing the function computed by RNN, GRU and LSTM.
\[ s_t = \sigma(U x_t + W s_{t-1} + b) \]
\[ \tilde{s}_t = \sigma(W(o_t \odot s_{t-1}) + U x_t + b) \]
\[ s_t = i_t \odot s_{t-1} + (1 - i_t) \odot \tilde{s}_t \]
\[ h_t = o_t \odot \sigma(s_t) \]

\[ s_t = \text{RNN}(s_{t-1}, x_t) \]
\[ s_t = \text{GRU}(s_{t-1}, x_t) \]
\[ h_t, s_t = \text{LSTM}(h_{t-1}, s_{t-1}, x_t) \]

- 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
So far we have seen how to model the conditional probability distribution $P(y_t | y_{1:t-1})$.
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.
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?
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?
So far we have seen how to model the conditional probability distribution
\( P(y_t | y_{t-1}) \)

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
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
A man throwing a frisbee in a park

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

$$P(y_t|y_1^{t-1}) = P(y_t = j|s_t)$$
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.
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 $P(y_t = j|y_1^{t-1}, I)$ as $P(y_t = j|s_t, f_{c7}(I))$.
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 $P(y_t = j|y_{1}^{t-1}, I)$ as $P(y_t = j|s_t, f_{c7}(I))$

where $f_{c7}(I)$ is the representation obtained from the $f_{c7}$ layer of an image.
There are many ways of making $P(y_t = j)$ conditional on $f_{c7}(I)$.
There are many ways of making $P(y_t = j)$ conditional on $f_{c7}(I)$

Let us see two such options
Option 1: Set $s_0 = f_{c7}(I)$
Option 1: Set $s_0 = f_{c7}(I)$

Now $s_0$ and hence all subsequent $s_t$’s depend on $f_{c7}(I)$
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 $f_{c_7}(I)$
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 $P(y_t = j|s_t, f_{c_7}(I))$
**Option 2**: Another more explicit way of doing this is to compute

\[ s_t = RNN(s_{t-1}, [x_t, f_{c7}(I)]) \]
Option 2: Another more explicit way of doing this is to compute
\[ s_t = RNN(s_{t-1}, [x_t, f_{c7}(I)]) \]

In other words we are explicitly using \( f_{c7}(I) \) to compute \( s_t \) and hence \( P(y_t = j) \)
Option 2: Another more explicit way of doing this is to compute

\[ s_t = RNN(s_{t-1}, [x_t, f_{c7}(I)]) \]

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 \( f_{c7} \)

\[ \langle \text{stop} \rangle \]

\[ s_0 = f_{c7}(I) \]

\[ \langle \text{GO} \rangle \]
Let us look at the full architecture.
Let us look at the full architecture

A CNN is first used to **encode** the image
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.
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**

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**
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.
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.
Module 16.2: 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?
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?
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?)
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?
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?
A man throwing ••• (stop)

Task: Image captioning

Parameters: $U_{dec}$, $V$, $W_{dec}$, $W_{conv}$; $b$

Loss: $L(t) = \sum_{t=1}^{T} \log P(y_t = \ell_t | y^{<t}, t)$
Task: Image captioning

Data: \( \{ x_i = \text{image}_i, \ y_i = \text{caption}_i \}_{i=1}^{N} \)
• **Task:** Image captioning
• **Data:** \( \{x_i = \text{image}_i, \ y_i = \text{caption}_i\}_{i=1}^{N} \)
• **Model:**
Task: Image captioning

Data: \( \{x_i = image_i, y_i = caption_i\}_{i=1}^{N} \)

Model:

- **Encoder:**
  
  \[ s_0 = CNN(x_i) \]
Task: Image captioning

Data: \( \{x_i = image_i, y_i = caption_i\}_{i=1}^{N} \)

Model:

- **Encoder:**
  \[ s_0 = CNN(x_i) \]

- **Decoder:**
  \[ s_t = RNN(s_{t-1}, e(\hat{y}_{t-1})) \]
  \[ P(y_t|y_1^{t-1}, I) = \text{softmax}(Vs_t + b) \]
Task: Image captioning
Data: \( \{x_i = image_i, y_i = caption_i\}_{i=1}^N \)
Model:

- **Encoder:**
  \[ s_0 = \text{CNN}(x_i) \]

- **Decoder:**
  \[ s_t = \text{RNN}(s_{t-1}, e(\hat{y}_{t-1})) \]
  \[ P(y_t|y_1^{t-1}, I) = \text{softmax}(V s_t + b) \]

Parameters: \( U_{\text{dec}}, V, W_{\text{dec}}, W_{\text{conv}}, b \)
Task: Image captioning
Data: \( \{ x_i = image_i, \ y_i = caption_i \}_{i=1}^N \)
Model:

Encoder:
\[ s_0 = CNN(x_i) \]

Decoder:
\[ s_t = RNN(s_{t-1}, e(\hat{y}_{t-1})) \]
\[ P(y_t|y_1^{t-1}, I) = \text{softmax}(V_{st} + b) \]

Parameters: \( U_{dec}, V, W_{dec}, W_{conv}, b \)
Loss:
\[ \mathcal{L}(\theta) = - \sum_{t=1}^{T} \sum_{i=1}^{T} \log P(y_t = \ell_t|y_1^{t-1}, I) \]
Task: Image captioning

Data: \( \{ x_i = image_i, \ y_i = caption_i \}^{N}_{i=1} \)

Model:
- **Encoder:**
  \[
  s_0 = CNN(x_i)
  \]
- **Decoder:**
  \[
  s_t = RNN(s_{t-1}, e(\hat{y}_{t-1}))
  \]
  \[
  P(y_t|y_1^{t-1}, I) = \text{softmax}(Vs_t + b)
  \]
- **Parameters:** \( U_{dec}, V, W_{dec}, W_{conv}, b \)
- **Loss:**
  \[
  \mathcal{L}(\theta) = \sum_{t=1}^{T} \mathcal{L}_t(\theta) = - \sum_{t=1}^{T} \log P(y_t = \ell_t|y_1^{t-1}, I)
  \]
- **Algorithm:** Gradient descent with backpropagation
Task: Textual entailment

i/p: It is raining outside

o/p: The ground is wet
Task: Textual entailment

Data: \( \{x_i = \text{premise}_i, \ y_i = \text{hypothesis}_i\}_{i=1}^N \)

i/p : It is raining outside

o/p : The ground is wet
Task: Textual entailment
Data: $\{x_i = \text{premise}_i, \ y_i = \text{hypothesis}_i\}_{i=1}^{N}$
Model (Option 1):

i/p: It is raining outside

o/p: The ground is wet
Task: Textual entailment

Data: \( \{x_i = \text{premise}_i, y_i = \text{hypothesis}_i\}_{i=1}^{N} \)

Model (Option 1):

- **Encoder:**
  \[
  h_t = \text{RNN}(h_{t-1}, x_{it})
  \]

i/p : It is raining outside

o/p : The ground is wet
**Task:** Textual entailment

**Data:** \( \{x_i = \text{premise}_i, \ y_i = \text{hypothesis}_i\}_{i=1}^N \)

**Model (Option 1):**

- **Encoder:**
  \[ h_t = RNN(h_{t-1}, x_{it}) \]

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

```
io/p : The ground is wet

i/p : It is raining outside
```
Textual entailment

Data: \( \{x_i = \text{premise}_i, y_i = \text{hypothesis}_i\}_{i=1}^N \)

Model (Option 1):
- **Encoder:**
  \[ h_t = RNN(h_{t-1}, x_{it}) \]
- **Decoder:**
  \[ s_0 = h_T \quad (T \text{ is length of input}) \]
  \[ s_t = RNN(s_{t-1}, e(\hat{y}_{t-1})) \]
  \[ P(y_t | y_1^{t-1}, x) = \text{softmax}(V_s + b) \]

- **Parameters:** \( U_{\text{dec}}, V, W_{\text{dec}}, U_{\text{enc}}, W_{\text{enc}}, b \)
Task: Textual entailment

Data: \{x_i = premise_i, y_i = hypothesis_i\}_{i=1}^N

Model (Option 1):

- **Encoder:**
  \[ h_t = RNN(h_{t-1}, x_{it}) \]

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

Parameters: \( U_{dec}, V, W_{dec}, U_{enc}, W_{enc}, b \)

Loss:
\[ \mathcal{L}(\theta) = \sum_{t=1}^{T} \mathcal{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t | y_1^{t-1}, x) \]
Task: Textual entailment

Data: \( \{x_i = \text{premise}_i, y_i = \text{hypothesis}_i\}_{i=1}^{N} \)

Model (Option 1):

- **Encoder:**
  \[ h_t = \text{RNN}(h_{t-1}, x_{it}) \]

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

Parameters: \( U_{\text{dec}}, V, W_{\text{dec}}, U_{\text{enc}}, W_{\text{enc}}, b \)

Loss:
\[ \mathcal{L}(\theta) = \sum_{t=1}^{T} \mathcal{L}_t(\theta) = - \sum_{t=1}^{T} \log P(y_t | y_{1}^{t-1}, x) \]

Algorithm: Gradient descent with backpropagation
**Task:** Textual entailment

**Data:** \( \{x_i = \text{premise}_i, y_i = \text{hypothesis}_i\}_{i=1}^N \)

**Model (Option 1):**

- **Encoder:**
  \[ h_t = \text{RNN}(h_{t-1}, x_{it}) \]

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

- **Parameters:** \( U_{\text{dec}}, V, W_{\text{dec}}, U_{\text{enc}}, W_{\text{enc}}, b \)

- **Loss:**
  \[ \mathcal{L}(\theta) = \sum_{t=1}^{T} \mathcal{L}_t(\theta) = - \sum_{t=1}^{T} \log P(y_t = l_t | y_{1}^{t-1}, x) \]

- **Algorithm:** Gradient descent with backpropagation

---

**Example:**

**i/p:** It is raining outside

**o/p:** The ground is wet
**Task:** Textual entailment

**Data:** \( \{x_i = \text{premise}_i, y_i = \text{hypothesis}_i\}_{i=1}^{N} \)

**Model (Option 1):**

- **Encoder:**
  \[ h_t = \text{RNN}(h_{t-1}, x_{it}) \]

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

- **Parameters:** \( U_{\text{dec}}, V, W_{\text{dec}}, U_{\text{enc}}, W_{\text{enc}}, b \)

- **Loss:**
  \[ \mathcal{L}(\theta) = \sum_{t=1}^{T} \mathcal{L}_t(\theta) = - \sum_{t=1}^{T} \log P(y_t = \ell_t|y_{1:t-1}, x) \]

- **Algorithm:** Gradient descent with backpropagation
Task: Textual entailment
Data: \( \{x_i = \text{premise}_i, \ y_i = \text{hypothesis}_i\}_{i=1}^{N} \)
Model (Option 1):
- **Encoder:**
  \[ h_t = RNN(h_{t-1}, x_{it}) \]
- **Decoder:**
  \[ s_0 = h_T \quad (T \text{ is length of input}) \]
  \[ s_t = RNN(s_{t-1}, e(\hat{y}_{t-1})) \]
  \[ P(y_t|y_{t-1}, x) = \text{softmax}(Vs_t + b) \]
Parameters: \( U_{\text{dec}}, V, W_{\text{dec}}, U_{\text{enc}}, W_{\text{enc}}, b \)
Loss:
\[ \mathcal{L}(\theta) = \sum_{t=1}^{T} \mathcal{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t|y_{1}^{t-1}, x) \]
Algorithm: Gradient descent with backpropagation
**Task:** Textual entailment

**Data:** \( \{x_i = \text{premise}_i, y_i = \text{hypothesis}_i\}_{i=1}^{N} \)

**Model (Option 1):**

- **Encoder:**
  \[
  h_t = \text{RNN}(h_{t-1}, x_{it})
  \]

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

- **Parameters:** \( U_{\text{dec}}, V, W_{\text{dec}}, U_{\text{enc}}, W_{\text{enc}}, b \)

- **Loss:**
  \[
  \mathcal{L}(\theta) = \sum_{t=1}^{T} \mathcal{L}_t(\theta) = - \sum_{t=1}^{T} \log P(y_t = \ell_t|y_{1:t-1}, x)
  \]

- **Algorithm:** Gradient descent with backpropagation
Task: Textual entailment

Data: \( \{x_i = \text{premise}_i, \ y_i = \text{hypothesis}_i\}_{i=1}^{N} \)

Model (Option 1):

- **Encoder:**
  \[ h_t = \text{RNN}(h_{t-1}, x_{it}) \]

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

- **Parameters:** \( U_{\text{dec}}, V, W_{\text{dec}}, U_{\text{enc}}, W_{\text{enc}}, b \)

- **Loss:**
  \[ \mathcal{L}(\theta) = \sum_{t=1}^{T} \mathcal{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t|y_1^{t-1}, x) \]

- **Algorithm:** Gradient descent with backpropagation
Task: Textual entailment

Data: \( \{x_i = \text{premise}_i, \ y_i = \text{hypothesis}_i\}_{i=1}^{N} \)

Model (Option 1):

- **Encoder:**
  \[ h_t = \text{RNN}(h_{t-1}, x_{it}) \]

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

- **Parameters:** \( U_{\text{dec}}, V, W_{\text{dec}}, U_{\text{enc}}, W_{\text{enc}}, b \)

- **Loss:**
  \[ \mathcal{L}(\theta) = \sum_{t=1}^{T} \mathcal{L}_t(\theta) = - \sum_{t=1}^{T} \log P(y_t | y_1^{t-1}, x) \]

- **Algorithm:** Gradient descent with backpropagation

---

Mitesh M. Khapra
CS7015 (Deep Learning) : Lecture 16
The ground is wet

**Task:** Textual entailment

**Data:** \( \{x_i = \text{premise}_i, \ y_i = \text{hypothesis}_i\}_{i=1}^N \)

**Model (Option 1):**

- **Encoder:**
  \[ h_t = \text{RNN}(h_{t-1}, x_{it}) \]

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

- **Parameters:** \( U_{\text{dec}}, V, W_{\text{dec}}, U_{\text{enc}}, W_{\text{enc}}, b \)

- **Loss:**
  \[ \mathcal{L}(\theta) = \sum_{t=1}^{T} \mathcal{L}_t(\theta) = - \sum_{t=1}^{T} \log P(y_t = \ell_t|y_1^{t-1}, x) \]

- **Algorithm:** Gradient descent with backpropagation
- **Task:** Textual entailment
- **Data:** \( \{x_i = \text{premise}_i, y_i = \text{hypothesis}_i\}_{i=1}^N \)
- **Model (Option 1):**
  - **Encoder:**
    \[ h_t = \text{RNN}(h_{t-1}, x_{it}) \]
  - **Decoder:**
    \[ s_0 = h_T \quad (T \text{ is length of input}) \]
    \[ s_t = \text{RNN}(s_{t-1}, e(\hat{y}_{t-1})) \]
    \[ P(y_t|y_t^{t-1}, x) = \text{softmax}(Vs_t + b) \]
- **Parameters:** \( U_{\text{dec}}, V, W_{\text{dec}}, U_{\text{enc}}, W_{\text{enc}}, b \)
- **Loss:**
  \[ \mathcal{L}(\theta) = \sum_{i=1}^{T} \mathcal{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t|y_1^{t-1}, x) \]
- **Algorithm:** Gradient descent with backpropagation
Task: Textual entailment

i/p: It is raining outside

o/p: The ground is wet
Task: Textual entailment

Data: \( \{x_i = \text{premise}_i, \ y_i = \text{hypothesis}_i\}_{i=1}^N \)

i/p: It is raining outside

o/p: The ground is wet
**Task:** Textual entailment

**Data:** \( \{x_i = \text{premise}_i, \ y_i = \text{hypothesis}_i\}_{i=1}^N \)

**Model (Option 2):**

i/p : It is raining outside

o/p : The ground is wet
Task: Textual entailment

Data: \( \{ x_i = \text{premise}_i, \ y_i = \text{hypothesis}_i \}_i^N \)

Model (Option 2):

- Encoder:
  \[ h_t = RNN(h_{t-1}, x_{it}) \]

o/p: The ground is wet

i/p: It is raining outside

i/p: It is raining outside
Task: Textual entailment

Data: \( \{x_i = \text{premise}_i, y_i = \text{hypothesis}_i\}_{i=1}^{N} \)

Model (Option 2):

- **Encoder:**
  \[ h_t = \text{RNN}(h_{t-1}, x_{it}) \]

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

**Data:** \( \{x_i = premise_i, \ y_i = hypothesis_i\}_{i=1}^N \)

**Model (Option 2):**

- **Encoder:**
  \[ h_t = RNN(h_{t-1}, x_{it}) \]

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

- **Parameters:** \( U_{\text{dec}}, V, W_{\text{dec}}, U_{\text{enc}}, W_{\text{enc}}, b \)

\[ i/p : \text{It is raining outside} \]

\[ o/p : \text{The ground is wet} \]
Task: Textual entailment

Data: \( \{ x_i = \text{premise}_i, \ y_i = \text{hypothesis}_i \}_{i=1}^{N} \)

Model (Option 2):

- **Encoder:**
  \[
  h_t = \text{RNN}(h_{t-1}, x_{it})
  \]

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

- **Parameters:** \( U_{\text{dec}}, V, W_{\text{dec}}, U_{\text{enc}}, W_{\text{enc}}, b \)

- **Loss:**
  \[
  \mathcal{L}(\theta) = \sum_{t=1}^{T} \mathcal{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = l_t|y_{t-1}^{t-1}, x)
  \]
Task: Textual entailment

Data: \( \{x_i = premise_i, y_i = hypothesis_i\}_{i=1}^{N} \)

Model (Option 2):

- **Encoder:**
  \[ h_t = RNN(h_{t-1}, x_{it}) \]

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

- **Parameters:** \( U_{dec}, V, W_{dec}, U_{enc}, W_{enc}, b \)

- **Loss:**
  \[ \mathcal{L}(\theta) = \sum_{t=1}^{T} \mathcal{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t | y_1^{t-1}, x) \]

- **Algorithm:** Gradient descent with backpropagation
Task: Machine translation

i/p : I am going home

o/p : Mein ghar ja raha hoon
Task: Machine translation  
Data: \( \{x_i = \text{source}_i, \ y_i = \text{target}_i\}_{i=1}^N \)

i/p: I am going home  

o/p: Mein ghar ja raha hoon
Task: Machine translation
Data: \( \{x_i = source_i, \ y_i = target_i\}_{i=1}^N \)
Model (Option 1):

\[
\begin{align*}
  o/p : & \text{ Mein ghar ja raha hoon} \\
  i/p : & \text{ I am going home}
\end{align*}
\]
Task: Machine translation

Data: \( \{x_i = \text{source}_i, \; y_i = \text{target}_i\}_{i=1}^N \)

Model (Option 1):
- Encoder:
  \[
  h_t = RNN(h_{t-1}, x_{it})
  \]
• **Task:** Machine translation

• **Data:** \( \{x_i = source_i, y_i = target_i\}_{i=1}^N \)

• **Model (Option 1):**

  - **Encoder:**
    \[
    h_t = RNN(h_{t-1}, x_{it})
    \]

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

i/p : I am going home

o/p : Mein ghar ja raha hoon
Task: Machine translation

Data: $\{x_i = source_i, \ y_i = target_i\}_{i=1}^{N}$

Model (Option 1):

**Encoder:**

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

**Decoder:**

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

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

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

**Parameters:** $U_{dec}, \ V, \ W_{dec}, \ U_{enc}, \ W_{enc}, \ b$

- i/p : I am going home
- o/p : Mein ghar ja raha hoon
Task: Machine translation

Data: \( \{ x_i = \text{source}_i, \ y_i = \text{target}_i \}_{i=1}^N \)

Model (Option 1):

- **Encoder:**
  \[
  h_t = \text{RNN}(h_{t-1}, x_{it})
  \]

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

Parameters: \( U_{\text{dec}}, V, W_{\text{dec}}, U_{\text{enc}}, W_{\text{enc}}, b \)

Loss:
\[
\mathcal{L}(\theta) = \sum_{i=1}^{T} \mathcal{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t|y_{t-1}^t, x)
\]
Task: Machine translation

Data: \( \{x_i = \text{source}_i, \ y_i = \text{target}_i\}_{i=1}^N \)

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) = \text{softmax}(Vs_t + b)
  \]

Parameters: \( U_{\text{dec}}, V, W_{\text{dec}}, U_{\text{enc}}, W_{\text{enc}}, b \)

Loss:
\[
\mathcal{L} (\theta) = \sum_{t=1}^{T} \mathcal{L}_t(\theta) = - \sum_{t=1}^{T} \log P(y_t = \ell_t|y_1^{t-1}, x)
\]

Algorithm: Gradient descent with backpropagation

---

- **i/p:** I am going home
- **o/p:** Mein ghar ja raha hoon

---

Mitesh M. Khapra CS7015 (Deep Learning) : Lecture 16
Task: Machine translation
Data: \( \{x_i = \text{source}_i, \ y_i = \text{target}_i\}_{i=1}^N \)
Model (Option 1):
- **Encoder:**
  
  \[
  h_t = RNN(h_{t-1}, x_{it})
  \]
- **Decoder:**
  
  \[
  s_0 = h_T \quad (T \text{ is length of input})
  \]
  \[
  s_t = RNN(s_{t-1}, e(\hat{y}_{t-1}))
  \]
  \[
  P(y_t|y_1^{t-1}, x) = \text{softmax}(Vs_t + b)
  \]
Parameters: \( U_{\text{dec}}, V, W_{\text{dec}}, U_{\text{enc}}, W_{\text{enc}}, b \)
Loss:
\[
\mathcal{L}(\theta) = \sum_{i=1}^{T} \mathcal{L}_t(\theta) = - \sum_{t=1}^{T} \log P(y_t = \ell_t|y_1^{t-1}, x)
\]
Algorithm: Gradient descent with backpropagation
Task: Machine translation

Data: \( \{x_i = source_i, \ y_i = target_i\}_{i=1}^N \)

Model (Option 1):

- **Encoder:**
  \[ h_t = RNN(h_{t-1}, x_{it}) \]

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

- **Parameters:** \( U_{\text{dec}}, V, W_{\text{dec}}, U_{\text{enc}}, W_{\text{enc}}, b \)

- **Loss:**
  \[ \mathcal{L}(\theta) = \sum_{i=1}^{T} \mathcal{L}_t(\theta) = - \sum_{t=1}^{T} \log P(y_t = \ell_t|y_1^{t-1}, x) \]

- **Algorithm:** Gradient descent with backpropagation
- **Task:** Machine translation
- **Data:** \( \{ x_i = source_i, \ y_i = target_i \}_{i=1}^{N} \)
- **Model (Option 1):**
  - **Encoder:**
    \[ h_t = RNN(h_{t-1}, x_{it}) \]
  - **Decoder:**
    \[ s_0 = h_T \quad (T \text{ is length of input}) \]
    \[ s_t = RNN(s_{t-1}, e(\hat{y}_{t-1})) \]
    \[ P(y_t|y_1^{t-1}, x) = \text{softmax}(Vs_t + b) \]
- **Parameters:** \( U_{dec}, V, W_{dec}, U_{enc}, W_{enc}, b \)
- **Loss:**
  \[ \mathcal{L}(\theta) = \sum_{t=1}^{T} \mathcal{L}_t(\theta) = - \sum_{t=1}^{T} \log P(y_t = \ell_t|y_1^{t-1}, x) \]
- **Algorithm:** Gradient descent with backpropagation

---

i/p : I am going home

o/p : Mein ghar ja raha hoon
- **Task:** Machine translation
- **Data:** \( \{ x_i = source_i, \ y_i = target_i \}_{i=1}^N \)
- **Model (Option 1):**
  - **Encoder:**
    \[ h_t = RNN(h_{t-1}, x_{it}) \]
  - **Decoder:**
    \[ s_0 = h_T \quad (T \text{ is length of input}) \]
    \[ s_t = RNN(s_{t-1}, e(\hat{y}_{t-1})) \]
    \[ P(y_t|y_1^{t-1}, x) = \text{softmax}(Vs_t + b) \]
- **Parameters:** \( U_{dec}, V, W_{dec}, U_{enc}, W_{enc}, b \)
- **Loss:**
  \[ \mathcal{L}(\theta) = \sum_{i=1}^T \mathcal{L}_i(\theta) = -\sum_{t=1}^T \log P(y_t = \ell_t|y_1^{t-1}, x) \]
- **Algorithm:** Gradient descent with backpropagation

---

**i/p:** I am going home

**o/p:** Mein ghar ja raha hoon
**Task:** Machine translation

**Data:** \( \{x_i = source_i, \ y_i = target_i\}_{i=1}^N \)

**Model (Option 1):**

- **Encoder:**
  \[
  h_t = RNN(h_{t-1}, x_{it})
  \]

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

- **Parameters:** \( U_{\text{dec}}, \ V, \ W_{\text{dec}}, \ U_{\text{enc}}, \ W_{\text{enc}}, \ b \)

- **Loss:**
  \[
  \mathcal{L}(\theta) = \sum_{i=1}^T \mathcal{L}_t(\theta) = - \sum_{t=1}^T \log P(y_t = \ell_t | y_1^{t-1}, x)
  \]

- **Algorithm:** Gradient descent with backpropagation
Task: Machine translation

Data: \( \{ x_i = \text{source}_i, \ y_i = \text{target}_i \}_{i=1}^{N} \)

Model (Option 1):

- **Encoder:**
  \[
  h_t = \text{RNN}(h_{t-1}, x_{it})
  \]

- **Decoder:**
  \[
  s_0 = h_T \quad (T \text{ is length of input})
  \]

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

\[
P(y_t|y_{t-1}, x) = \text{softmax}(Vs_t + b)
  \]

- **Parameters:** \( U_{\text{dec}}, V, W_{\text{dec}}, U_{\text{enc}}, W_{\text{enc}}, b \)

- **Loss:**
  \[
  \mathcal{L}(\theta) = \sum_{t=1}^{T} \mathcal{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t|y_{t-1}^t, x)
  \]

- **Algorithm:** Gradient descent with backpropagation
Task: Machine translation

Data: \( \{x_i = \text{source}_i, \ y_i = \text{target}_i\}_{i=1}^{N} \)

Model (Option 1):
- **Encoder:**
  \[ h_t = \text{RNN}(h_{t-1}, x_{it}) \]
- **Decoder:**
  \[ s_0 = h_T \quad (T \text{ is length of input}) \]
  \[ s_t = \text{RNN}(s_{t-1}, e(\hat{y}_{t-1})) \]
  \[ P(y_t|y_{t-1}^t, x) = \text{softmax}(Vs_t + b) \]

Parameters: \( U_{dec}, V, W_{dec}, U_{enc}, W_{enc}, b \)

Loss:
\[ \mathcal{L}(\theta) = \sum_{i=1}^{T} \mathcal{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t|y_{1}^{t-1}, x) \]

Algorithm: Gradient descent with backpropagation

\[ \text{i/p : I am going home} \]
\[ \text{o/p : Mein ghar ja raha hoon} \]
Task: Machine translation

Data: \( \{x_i = source_i, y_i = target_i\}_{i=1}^N \)

Model (Option 1):
- **Encoder:**
  \[ h_t = RNN(h_{t-1}, x_{it}) \]
- **Decoder:**
  \[ s_0 = h_T \quad (T \text{ is length of input}) \]
  \[ s_t = RNN(s_{t-1}, e(\hat{y}_{t-1})) \]
  \[ P(y_t|y_{1:t-1}, x) = \text{softmax}(Vs_t + b) \]

Parameters: \( U_{dec}, V, W_{dec}, U_{enc}, W_{enc}, b \)

Loss:
\[ \mathcal{L}(\theta) = \sum_{t=1}^{T} \mathcal{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t|y_{1:t-1}, x) \]

Algorithm: Gradient descent with backpropagation
Task: Machine translation

Data: \( \{ x_i = \text{source}_i, \ y_i = \text{target}_i \}_{i=1}^N \)

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) = \text{softmax}(Vs_t + b)
  \]

Parameters: \( U_{dec}, V, W_{dec}, U_{enc}, W_{enc}, b \)

Loss:
\[
\mathcal{L}(\theta) = \sum_{i=1}^T \mathcal{L}_t(\theta) = - \sum_{t=1}^T \log P(y_t = \ell_{t} | y_1^{t-1}, x)
\]

Algorithm: Gradient descent with backpropagation
Task: Transliteration

i/p: INDIA

o/p: इंडिया
Task: Transliteration
Data: \( \{ x_i = \text{srcword}_i, \ y_i = \text{tgtword}_i \}_{i=1}^N \)
Task: Transliteration
Data: $\{x_i = \text{srcword}_i, y_i = \text{tgtword}_i\}_{i=1}^N$
Model (Option 1):

\[\text{o/p: } \text{i/p: } \text{INDIA}\]
Task: Transliteration

Data: \( \{x_i = \text{srcword}_i, y_i = \text{tgtword}_i\}_{i=1}^N \)

Model (Option 1):

Encoder:

\[ h_t = \text{RNN}(h_{t-1}, x_{it}) \]
- **Task:** Transliteration
- **Data:** \( \{x_i = \text{srcword}_i, \ y_i = \text{tgtword}_i\}_{i=1}^N \)
- **Model (Option 1):**
  - **Encoder:**
    \[ h_t = \text{RNN}(h_{t-1}, x_{it}) \]
  - **Decoder:**
    \[ s_0 = h_T \quad (T \text{ is length of input}) \]
    \[ s_t = \text{RNN}(s_{t-1}, e(\hat{y}_{t-1})) \]
    \[ P(y_t|y_1^{t-1}, x) = \text{softmax}(Vs_t + b) \]

```
i/p: IN D I A

[Diagram]

o/p: इ ड फ व आ

[Diagram]

o/p: इ . ड फ व आ

[Diagram]

<Go>

[Diagram]
```
- **Task:** Transliteration
- **Data:** $\{x_i = srcword_i, y_i = tgtword_i\}_{i=1}^{N}$
- **Model (Option 1):**
  - **Encoder:**
    $$ h_t = RNN(h_{t-1}, x_{it}) $$
  - **Decoder:**
    $$ s_0 = h_T \quad (T \text{ is length of input}) $$
    $$ s_t = RNN(s_{t-1}, e(\hat{y}_{t-1})) $$
    $$ P(y_t | y_{t-1}, x) = \text{softmax}(V s_t + b) $$
- **Parameters:** $U_{dec}, V, W_{dec}, U_{enc}, W_{enc}, b$
Task: Transliteration
Data: \( \{x_i = \text{srcword}_i, \ y_i = \text{tgtword}_i\}_{i=1}^N \)
Model (Option 1):
- **Encoder:**
  \[ h_t = RNN(h_{t-1}, x_{it}) \]
- **Decoder:**
  \[ s_0 = h_T \quad (T \text{ is length of input}) \]
  \[ s_t = RNN(s_{t-1}, e(\hat{y}_{t-1})) \]
  \[ P(y_t|y_{t-1}, x) = \text{softmax}(Vs_t + b) \]
- **Parameters:** \( U_{dec}, V, W_{dec}, U_{enc}, W_{enc}, b \)
- **Loss:**
  \[ \mathcal{L}(\theta) = \sum_{t=1}^{T} \mathcal{L}_t(\theta) = - \sum_{t=1}^{T} \log P(y_t = \ell_t|y_{t-1}, x) \]
Task: Transliteration

Data: \( \{x_i = \text{srcword}_i, \ y_i = \text{tgtword}_i\}_{i=1}^N \)

Model (Option 1):

Encoder:
\[ h_t = \text{RNN}(h_{t-1}, x_{it}) \]

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

Parameters: \( U_{\text{dec}}, V, W_{\text{dec}}, U_{\text{enc}}, W_{\text{enc}}, b \)

Loss:
\[ \mathcal{L}(\theta) = \sum_{i=1}^{T} \mathcal{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t | y_1^{t-1}, x) \]

Algorithm: Gradient descent with backpropagation
Task: Transliteration

Data: \( \{x_i = srcword_i, y_i = tgtword_i\}_{i=1}^N \)

Model (Option 2):

- **Encoder:**
  \[
  h_t = RNN(h_{t-1}, x_{it})
  \]

- **Decoder:**
  \[
  s_0 = h_T \quad \text{(T is length of input)}
  
  s_t = RNN(s_{t-1}, [c(\hat{y}_{t-1}), h_T])
  
  P(y_t|y_{t-1}, x) = \text{softmax}(Vs_t + b)
  \]

- **Parameters:** \( U_{\text{dec}}, V, W_{\text{dec}}, U_{\text{enc}}, W_{\text{enc}}, b \)

- **Loss:**
  \[
  \mathcal{L}(\theta) = \sum_{i=1}^{T} \mathcal{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t|y_{t-1}^{t-1}, x)
  \]

- **Algorithm:** Gradient descent with backpropagation
O/p: White

Question: What is the bird’s color

Task: Image Question Answering
O/p: White

- **Task:** Image Question Answering
- **Data:** \( \{x_i = \{I, q\}_i, \ y_i = Answer_i\}_{i=1}^N \)

---

Question: What is the bird’s color
O/p: White

- **Task:** Image Question Answering
- **Data:** \( \{x_i = \{I, q\}_i, y_i = Answer_i\}_{i=1}^N \)
- **Model:**

**Question:** What is the bird’s color
O/p: White

- **Task:** Image Question Answering
- **Data:** \( \{x_i = \{I, q\}_i, y_i = Answer_i\}_{i=1}^N \)
- **Model:**
  - **Encoder:**
    \[
    \hat{h}_I = \text{CNN}(I), \quad \hat{h}_t = \text{RNN}(\hat{h}_{t-1}, q_{it})
    \]
    \[
    s = [\hat{h}_T; \hat{h}_I]
    \]

**Question:** What is the bird’s color
O/p: White

Task: Image Question Answering

Data: \( \{x_i = \{I, q\}, y_i = Answer_i\}_{i=1}^N \)

Model:

Encoder:
\[ \hat{h}_I = \text{CNN}(I), \quad \hat{h}_t = \text{RNN}(\hat{h}_{t-1}, q_{it}) \]
\[ s = [\hat{h}_T; \hat{h}_I] \]

Decoder:
\[ P(y|q, I) = \text{softmax}(Vs + b) \]

Question: What is the bird’s color
Question: What is the bird’s color

O/p: White

Task: Image Question Answering

Data: \( x_i = \{I, q\}_i \), \( y_i = Answer_i \) \( i = 1 \ldots N \)

Model:

- **Encoder:**
  \[
  \hat{h}_I = CNN(I), \quad \hat{h}_t = RNN(\tilde{h}_{t-1}, q_{it})
  \]
  \[
  s = [\tilde{h}_T; \hat{h}_I]
  \]

- **Decoder:**
  \[
  P(y|q, I) = \text{softmax}(Vs + b)
  \]

Parameters: \( V, b, U_q, W_q, W_{conv}, b \)
O/p: White

- **Task:** Image Question Answering
- **Data:** \( \{x_i = \{I, q\}_i, y_i = \text{Answer}_i\}_{i=1}^N \)
- **Model:**
  - **Encoder:**
    \[ \hat{h}_I = \text{CNN}(I), \quad \hat{h}_t = \text{RNN}(\hat{h}_{t-1}, q_{it}) \]
    \[ s = [\hat{h}_T; \hat{h}_I] \]
  - **Decoder:**
    \[ P(y|q, I) = \text{softmax}(Vs + b) \]
- **Parameters:** \( V, b, U_q, W_q, W_{\text{conv}}, b \)
- **Loss:**
  \[ \mathcal{L}(\theta) = -\log P(y = \ell|I, q) \]

Question: What is the bird’s color
Question: What is the bird’s color

O/p: White

Task: Image Question Answering

Data: \( x_i = \{I, q\}_i, y_i = Answer_i \) \( i=1 \)

Model:

- **Encoder:**
  \[
  \hat{h}_I = CNN(I), \quad \hat{h}_t = RNN(\hat{h}_{t-1}, q_it)
  \]
  \[
  s = [\hat{h}_T; \hat{h}_I]
  \]

- **Decoder:**
  \[
  P(y|q, I) = \text{softmax}(Vs + b)
  \]

Parameters: \( V, b, U_q, W_q, W_{conv}, b \)

Loss:
\[
\mathcal{L}(\theta) = -\log P(y = \ell|I, q)
\]

Algorithm: Gradient descent with backpropagation
Task: Document Summarization

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

o/p: India won the world cup
o/p : India won the world cup

Task: Document Summarization

Data: \( \{ x_i = Document_i, \; y_i = Summary_i \}_{i=1}^{N} \)

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

o/p: India wins the world cup.
o/p : India won the world cup

Task: Document Summarization

Data: \( \{ x_i = Document_i, y_i = Summary_i \}_{i=1}^{N} \)

Model:

Encoder:

\[ h_t = RNN(h_{t-1}, x_{it}) \]

i/p : India beats Srilanka to win ICC WC 2011.
Dhoni and Gambhir’s half centuries help beat SL
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:**
  \[ s_0 = h_T \]
  \[ s_t = RNN(s_{t-1}, e(\hat{y}_{t-1})) \]
  \[ P(y_t|y_{1:t-1}, x) = \text{softmax}(Vs_t + b) \]
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:**
  \[ s_0 = h_T \]
  \[ s_t = RNN(s_{t-1}, e(\hat{y}_{t-1})) \]
  \[ P(y_t|y_{1:t-1}, x) = \text{softmax}(V s_t + b) \]

Parameters: \( U_{dec}, V, W_{dec}, U_{enc}, W_{enc}, b \)

---

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

o/p : India won the world cup
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:**
  \[ s_0 = h_T \]
  \[ s_t = RNN(s_{t-1}, e(\hat{y}_{t-1})) \]
  \[ P(y_t|y_1^{t-1}, x) = \text{softmax}(V s_t + b) \]

Parameters: \( U_{dec}, V, W_{dec}, U_{enc}, W_{enc}, b \)

Loss:
\[ \mathcal{L}(\theta) = \sum_{t=1}^{T} \mathcal{L}_t(\theta) = - \sum_{t=1}^{T} \log P(y_t = \ell_t|y_1^{t-1}, x) \]

---

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

**o/p:** India won the world cup
Task: Document Summarization

Data: \( \{x_i = \text{Document}_i, \ y_i = \text{Summary}_i\}_{i=1}^{N} \)

Model:

- **Encoder:**
  \[ h_t = \text{RNN}(h_{t-1}, x_{it}) \]

- **Decoder:**
  \[ s_0 = h_T \]
  \[ s_t = \text{RNN}(s_{t-1}, e(\hat{y}_{t-1})) \]
  \[ P(y_t|y_1^{t-1}, x) = \text{softmax}(Vs_t + b) \]

Parameters: \( U_{dec}, V, W_{dec}, U_{enc}, W_{enc}, b \)

Loss:
\[ \mathcal{L}(\theta) = \sum_{i=1}^{T} \mathcal{L}_i(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t|y_1^{t-1}, x) \]

Algorithm: Gradient descent with backpropagation
Task: Video Captioning

o/p: A man walking on a rope

Parameters: $U^{dec}$, $W^{dec}$, $V$, $b$, $W^{conv}$, $U^{enc}$, $W^{enc}$; $b$

Loss: $L(y) = \sum_{i=1}^{T} L(y_i | y_{i-1}, x)$

Algorithm: Gradient descent with backpropagation
o/p: A man walking on a rope

- **Task:** Video Captioning
- **Data:** \( \{x_i = video_i, y_i = desc_i\}^N_{i=1} \)
o/p: A man walking on a rope

- **Task:** Video Captioning
- **Data:** \( \{x_i = video_i, \ y_i = desc_i\}_{i=1}^N \)
- **Model:**
Task: Video Captioning
Data: \( \{x_i = video_i, \ y_i = desc_i\}_{i=1}^{N} \)
Model:
Encoder:
\[
h_t = RNN(h_{t-1}, CNN(x_{it}))
\]
• **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:**
    
    \[ s_0 = h_T \]
    \[ s_t = RNN(s_{t-1}, e(\hat{y}_{t-1})) \]
    \[ P(y_t|y_{1}^{t-1}, x) = \text{softmax}(Vs_t + b) \]
- **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:**
    \[ s_0 = h_T \]
    \[ s_t = RNN(s_{t-1}, e(\hat{y}_{t-1})) \]
    \[ P(y_t|y_{1:t-1}, x) = \text{softmax}(V_s + b) \]
- **Parameters:** \( U_{dec}, W_{dec}, V, b, W_{conv}, U_{enc}, W_{enc}, b \)
Task: Video Captioning

Data: \( \{x_i = \text{video}_i, \ y_i = \text{desc}_i\}_i^N \)

Model:

- **Encoder:**
  \[ h_t = \text{RNN}(h_{t-1}, \text{CNN}(x_{it})) \]

- **Decoder:**
  \[ s_0 = h_T \]
  \[ s_t = \text{RNN}(s_{t-1}, e(\hat{y}_{t-1})) \]
  \[ P(y_t|y_1^{t-1}, x) = \text{softmax}(V s_t + b) \]

Parameters: \( U_{\text{dec}}, W_{\text{dec}}, V, b, W_{\text{conv}}, U_{\text{enc}}, W_{\text{enc}}, b \)

Loss:
\[ \mathcal{L}(\theta) = \sum_{i=1}^{T} \mathcal{L}_i(\theta) = - \sum_{t=1}^{T} \log P(y_t = \ell_t|y_1^{t-1}, x) \]
Task: Video Captioning

Data: \( \{ x_i = video_i, \ y_i = desc_i \}_{i=1}^N \)

Model:

Encoder:

\[ h_t = RNN(h_{t-1}, \text{CNN}(x_{it})) \]

Decoder:

\[ s_0 = h_T \]
\[ s_t = RNN(s_{t-1}, e(\hat{y}_{t-1})) \]
\[ P(y_t | y_1^{t-1}, x) = \text{softmax}(V_s t + b) \]

Parameters: \( U_{dec}, W_{dec}, V, b, W_{conv}, U_{enc}, W_{enc}, b \)

Loss:

\[ \mathcal{L}(\theta) = \sum_{i=1}^{T} \mathcal{L}_i(\theta) = - \sum_{t=1}^{T} \log P(y_t = \ell_t | y_1^{t-1}, x) \]

Algorithm: Gradient descent with backpropagation
o/p: Surya Namaskar

**Task:** Video Classification
o/p: Surya Namaskar

- **Task**: Video Classification
- **Data**: \( \{x_i = Video_i, y_i = Activity_i\}_{i=1}^N \)
**Task:** Video Classification

**Data:** \( \{x_i = Video_i, y_i = Activity_i\}_{i=1}^{N} \)

**Model:**

\[ o/p: \text{Surya Namaskar} \]
Task: Video Classification
Data: \( \{x_i = Video_i, y_i = Activity_i\}^N_{i=1} \)
Model:
  Encoder:
  \[ h_t = RNN(h_{t-1}, CNN(x_{it})) \]

\[ h_t = RNN(h_{t-1}, CNN(x_{it})) \]
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:**
  \[ s = h_T \]
  \[ P(y|I) = softmax(Vs + b) \]
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:
  \[ s = h_T \]
  \[ P(y|I) = \text{softmax}(Vs + b) \]

Parameters: \( V, b, W_{conv}, U_{enc}, W_{enc}, b \)
Task: Video Classification
Data: \([x_i = Video_i, y_i = Activity_i]_{i=1}^{N}\)
Model:
   - Encoder:
     \[h_t = RNN(h_{t-1}, \text{CNN}(x_{it}))\]
   - Decoder:
     \[s = h_T\]
     \[P(y|I) = \text{softmax}(Vs + b)\]
Parameters: \(V, b, W_{\text{conv}}, U_{\text{enc}}, W_{\text{enc}}, b\)
Loss:
\[\mathcal{L}(\theta) = -\log P(y = \ell|Video)\]
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:**
  \[
  s = h_T
  \]
  
  \[
  P(y|I) = \text{softmax}(Vs + b)
  \]

Parameters: \( V, b, W_{conv}, U_{enc}, W_{enc}, b \)

Loss:

\[
\mathcal{L}(\theta) = - \log P(y = \ell|\text{Video})
\]

Algorithm: Gradient descent with backpropagation
Task: Dialog

i/p: How are you

o/p: I am fine
o/p: I am fine

Task: Dialog

Data: \( \{x_i = \text{Utterance}_i, y_i = \text{Response}_i\}^N_{i=1} \)

i/p: How are you
o/p: I am fine

Task: Dialog

Data: \( \{ x_i = \text{Utterance}_i, \ y_i = \text{Response}_i \}_{i=1}^N \)

Model:

i/p: How are you
Task: Dialog
Data: \( \{x_i = \text{Utterance}_i, \ y_i = \text{Response}_i\}_{i=1}^N \)
Model:
Encoder:
\[ h_t = RNN(h_{t-1}, x_{it}) \]
Task: Dialog

Data: \( \{ x_i = \text{Utterance}_i, \ y_i = \text{Response}_i \}^{N}_{i=1} \)

Model:

- **Encoder:**
  \[
  h_t = \text{RNN}(h_{t-1}, x_{it})
  \]

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

i/p: How are you
Task: Dialog

Data: \( \{ x_i = \text{Utterance}_i, \ y_i = \text{Response}_i \} \)

Model:

Encoder:
\[ h_t = \text{RNN}(h_{t-1}, x_{it}) \]

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

Parameters: \( U_{\text{dec}}, V, W_{\text{dec}}, U_{\text{enc}}, W_{\text{enc}}, b \)

i/p: How are you
Task: Dialog
Data: \( \{ x_i = \text{Utterance}_i, \ y_i = \text{Response}_i \}_{i=1}^N \)
Model:
- Encoder: \( h_t = \text{RNN}(h_{t-1}, x_{it}) \)
- Decoder: \( s_0 = h_T \) (\( T \) is length of input)
  \[ s_t = \text{RNN}(s_{t-1}, e(\hat{y}_{t-1})) \]
  \[ P(y_t | y_{1:t-1}, x) = \text{softmax}(Vs_t + b) \]
Parameters: \( U_{dec}, V, W_{dec}, U_{enc}, W_{enc}, b \)
Loss:
\[ \mathcal{L}(\theta) = \sum_{t=1}^{T} \mathcal{L}_t(\theta) = - \sum_{t=1}^{T} \log P(y_t = \ell_t | y_{1:t-1}, x) \]
Task: Dialog

Data: \( \{ x_i = \text{Utterance}_i, \ y_i = \text{Response}_i \}_{i=1}^N \)

Model:

- **Encoder:**
  \[ h_t = \text{RNN}(h_{t-1}, x_{it}) \]

- **Decoder:**
  \[ s_0 = h_T \quad (T \text{ is length of input}) \]
  \[ s_t = \text{RNN}(s_{t-1}, e(y_{t-1})) \]
  \[ P(y_t|y_{t-1}, x) = \text{softmax}(Vs_t + b) \]

- **Parameters:** \( U_{\text{dec}}, V, W_{\text{dec}}, U_{\text{enc}}, W_{\text{enc}}, b \)

- **Loss:**
  \[ \mathcal{L}(\theta) = \sum_{i=1}^{T} \mathcal{L}_i(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t|y_{t-1}, x) \]

- **Algorithm:** Gradient descent with backpropagation
And the list continues ...
• And the list continues ...
• Try picking a problem from your domain and see if you can model it using the encoder decoder paradigm.
• And the list continues ...

• Try picking a problem from your domain and see if you can model it using the encoder decoder paradigm

• Encoder decoder models can be made even more expressive by adding an “attention” mechanism
• And the list continues ...

• Try picking a problem from your domain and see if you can model it using the encoder decoder paradigm.

• Encoder decoder models can be made even more expressive by adding an “attention” mechanism.

• We will first motivate the need for this and then explain how to model it.
Module 16.3: Attention Mechanism
Let us motivate the task of attention with the help of MT

i/p : Main ghar ja raha hoon

o/p : I am going home
Let us motivate the task of attention with the help of MT

The encoder reads the sentences only once and encodes it

o/p: I am going home

i/p: Main ghar ja raha hoon
Let us motivate the task of attention with the help of MT.

- The encoder reads the sentences only once and encodes it.
- At each timestep the decoder uses this embedding to produce a new word.
Let us motivate the task of attention with the help of MT

The encoder reads the sentences only once and encodes it

At each timestep the decoder uses this embedding to produce a new word

Is this how humans translate a sentence?

---

Let us consider the following input-output pair:

**Input (i/p):** Main ghar ja raha hoon

**Output (o/p):** I am going home

---

The decoder uses the encoder's output at each timestep to produce the output sequence.
Let us motivate the task of attention with the help of MT.

The encoder reads the sentences only once and encodes it.

At each timestep the decoder uses this embedding to produce a new word.

Is this how humans translate a sentence? Not really!
Humans try to produce each word in the output by focusing only on certain words in the input.
Humans try to produce each word in the output by focusing only on certain words in the input.

Essentially at each time step we come up with a distribution on the input words.

**i/p:** Main ghar ja raha hoon

**o/p:** I am going home
Humans try to produce each word in the output by focusing only on certain words in the input.

Essentially at each time step we come up with a distribution on the input words.

o/p: I am going home

\[ t_1: [1 \ 0 \ 0 \ 0 \ 0] \]

i/p: Main ghar ja raha hoon
Humans try to produce each word in the output by focusing only on certain words in the input.

Essentially at each time step we come up with a distribution on the input words.

\[
\begin{align*}
o/p & : \text{I am going home} \\
t_1 & : [1 \ 0 \ 0 \ 0 \ 0] \\
t_2 & : [0 \ 0 \ 0 \ 0 \ 1] \\
i/p & : \text{Main ghar ja raha hoon}
\end{align*}
\]
Humans try to produce each word in the output by focusing only on certain words in the input.

Essentially at each time step we come up with a distribution on the input words.

o/p : I am going home

- $t_1 : [1 \ 0 \ 0 \ 0 \ 0]$
- $t_2 : [0 \ 0 \ 0 \ 1]$
- $t_3 : [0 \ 0 \ 0.5 \ 0.5 \ 0]$

i/p : Main ghar ja raha hoon
Humans try to produce each word in the output by focusing only on certain words in the input.

Essentially at each time step we come up with a distribution on the input words.

**i/p:** Main ghar ja raha hoon

**o/p:** I am going home

\[ t_1 : [1 \ 0 \ 0 \ 0 \ 0] \]

\[ t_2 : [0 \ 0 \ 0 \ 1] \]

\[ t_3 : [0 \ 0 \ 0.5 \ 0.5 \ 0] \]

\[ t_4 : [0 \ 1 \ 0 \ 0 \ 0] \]
Humans try to produce each word in the output by focusing only on certain words in the input.

Essentially at each time step we come up with a distribution on the input words.

This distribution tells us how much attention to pay to each input words at each time step.
Humans try to produce each word in the output by focusing only on certain words in the input.

Essentially at each time step we come up with a distribution on the input words.

This distribution tells us how much attention to pay to each input words at each time step.

Ideally, at each time-step we should feed only this relevant information (i.e. encodings of relevant words) to the decoder.
Let us revisit the decoder that we have seen so far.

\[ \text{o/p: I am going home} \]

\[ \text{i/p: Main ghar ja raha hoon} \]
Let us revisit the decoder that we have seen so far.

We either feed in the encoder information only once (at $s_0$)
Let us revisit the decoder that we have seen so far.

- We either feed in the encoder information only once (at $s_0$).
- Or we feed the same encoder information at each time step.
Let us revisit the decoder that we have seen so far.

- We either feed in the encoder information only once (at $s_0$).
- Or we feed the same encoder information at each time step.
- Now suppose an oracle told you which words to focus on at a given time-step $t$.

Let's consider the input sequence: "Main ghar ja raha hoon".

The decoder produces the output sequence: "I am going home".

The decoder process involves feeding the encoder information (at $s_0$) and then iteratively feeding the same information at each time step. The oracle helps in deciding which words to focus on at each time step.
- Let us revisit the decoder that we have seen so far
- We either feed in the encoder information only once (at $s_0$)
- Or we feed the same encoder information at each time step
- Now suppose an oracle told you which words to focus on at a given time-step $t$
- Can you think of a smarter way of feeding information to the decoder?

Let us revisit the decoder that we have seen so far.

We either feed in the encoder information only once (at $s_0$).

Or we feed the same encoder information at each time step.

Now suppose an oracle told you which words to focus on at a given time-step $t$.

Can you think of a smarter way of feeding information to the decoder?
We could just take a weighted average of the corresponding word representations and feed it to the decoder. For example at timestep 3, we can just take a weighted average of the representations of 'ja' and 'raha'. Intuitively this should work better because we are not overloading the decoder with irrelevant information (about words that do not matter at this time step).

How do we convert this intuition into a model?
We could just take a weighted average of the corresponding word representations and feed it to the decoder.
- We could just take a weighted average of the corresponding word representations and feed it to the decoder.
- For example at timestep 3, we can just take a weighted average of the representations of ‘ja’ and ‘raha’.
- We could just take a weighted average of the corresponding word representations and feed it to the decoder.
- For example at timestep 3, we can just take a weighted average of the representations of 'ja' and 'raha'.
- Intuitively this should work better because we are not overloading the decoder with irrelevant information (about words that do not matter at this time step).
We could just take a weighted average of the corresponding word representations and feed it to the decoder.

For example at timestep 3, we can just take a weighted average of the representations of ‘ja’ and ‘raha’.

Intuitively this should work better because we are not overloading the decoder with irrelevant information (about words that do not matter at this time step).

How do we convert this intuition into a model?
Of course in practice we will not have this oracle. The machine will have to learn this from the data. To enable this we define a function $e_{jt} = f_{ATT}(s_t; c_j)$. This quantity captures the importance of the $j$th input word for decoding the $t$th output word (we will see the exact form of $f_{ATT}$ later). We can normalize these weights by using the softmax function $e_{jt} = \exp(e_{jt}) \sum_{j=1}^M \exp(e_{jt})$. 

Mitesh M. Khapra
CS7015 (Deep Learning) : Lecture 16
Of course in practice we will not have this oracle
- Of course in practice we will not have this oracle
- The machine will have to learn this from the data
- Of course in practice we will not have this oracle
- The machine will have to learn this from the data
- To enable this we define a function

\[ e_{jt} = f_{ATT}(s_{t-1}, c_j) \]
Of course in practice we will not have this oracle.

The machine will have to learn this from the data.

To enable this we define a function

\[ e_{jt} = f_{ATT}(s_{t-1}, c_j) \]

This quantity captures the importance of the \( j^{th} \) input word for decoding the \( t^{th} \) output word (we will see the exact form of \( f_{ATT} \) later).
Of course in practice we will not have this oracle.

The machine will have to learn this from the data.

To enable this we define a function

\[ e_{jt} = f_{ATT}(s_{t-1}, c_j) \]

This quantity captures the importance of the \( j^{th} \) input word for decoding the \( t^{th} \) output word (we will see the exact form of \( f_{ATT} \) later).

We can normalize these weights by using the softmax function

\[ \alpha_{jt} = \frac{\exp(e_{jt})}{\sum_{j=1}^{M} \exp(e_{jt})} \]
I am going home <STOP>

\[
\text{jt denotes the probability of focusing on the } j\text{th word to produce the } t\text{th output word.}
\]

We are now trying to learn the \( \alpha \)'s instead of an oracle informing us about the \( \alpha \)'s.

Learning would always involve some parameters. So let's define a parametric form for \( \alpha \)'s.
\[ \alpha_{jt} = \frac{\exp(e_{jt})}{\sum_{j=1}^{M} \exp(e_{jt})} \]
\[
\alpha_{jt} = \frac{\exp(e_{jt})}{\sum_{j=1}^{M} \exp(e_{jt})}
\]

- \(\alpha_{jt}\) denotes the probability of focusing on the \(j^{th}\) word to produce the \(t^{th}\) output word.
\[ \alpha_{jt} = \frac{\exp(e_{jt})}{\sum_{j=1}^{M} \exp(e_{jt})} \]

- \( \alpha_{jt} \) denotes the probability of focusing on the \( j^{th} \) word to produce the \( t^{th} \) output word.
- We are now trying to learn the \( \alpha \)'s instead of an oracle informing us about the \( \alpha \)'s.
\[ \alpha_{jt} = \frac{\exp(e_{jt})}{\sum_{j=1}^{M} \exp(e_{jt})} \]

- \( \alpha_{jt} \) denotes the probability of focusing on the \( j^{th} \) word to produce the \( t^{th} \) output word.
- We are now trying to learn the \( \alpha \)'s instead of an oracle informing us about the \( \alpha \)'s.
- Learning would always involve some parameters.

Mitesh M. Khapra  
CS7015 (Deep Learning) : Lecture 16
\[
\alpha_{jt} = \frac{\exp(e_{jt})}{\sum_{j=1}^{M} \exp(e_{jt})}
\]

- \(\alpha_{jt}\) denotes the probability of focusing on the \(j^{th}\) word to produce the \(t^{th}\) output word.
- We are now trying to learn the \(\alpha\)'s instead of an oracle informing us about the \(\alpha\)'s.
- Learning would always involve some parameters.
- So let's define a parametric form for \(\alpha\)'s.
Given these new notations, one (among many) possible choice for $f_{\text{ATT}}$ is

$$e_{jt} = V_{\text{ATT}} \tanh(U_{\text{ATT}} s_t + W_{\text{ATT}} c_j)$$

$V_{\text{ATT}}$, $U_{\text{ATT}}$, and $W_{\text{ATT}}$ are additional parameters of the model. These parameters will be learned along with the other parameters of the encoder and decoder.
From now on we will refer to the decoder RNN’s state at the $t$-th timestep as $s_t$ and the encoder RNN’s state at the $j$-th timestep as $c_j$. 
From now on we will refer to the decoder RNN’s state at the $t$-th timestep as $s_t$ and the encoder RNN’s state at the $j$-th time step as $c_j$.

Given these new notations, one (among many) possible choice for $f_{ATT}$ is

$$e_{jt} = V^T_{att} \tanh(U_{att}s_{t-1} + W_{att}c_j)$$
From now on we will refer to the decoder RNN’s state at the $t$-th timestep as $s_t$ and the encoder RNN’s state at the $j$-th timestep as $c_j$.

Given these new notations, one (among many) possible choice for $f_{ATT}$ is:

$$e_{jt} = V_{att}^T \tanh(U_{att}s_{t-1} + W_{att}c_j)$$

$V_{att} \in \mathbb{R}^d$, $U_{att} \in \mathbb{R}^{d \times d}$, $W_{att} \in \mathbb{R}^{d \times d}$ are additional parameters of the model.
From now on we will refer to the decoder RNN’s state at the $t$-th timestep as $s_t$ and the encoder RNN’s state at the $j$-th time step as $c_j$

Given these new notations, one (among many) possible choice for $f_{ATT}$ is

$$e_{jt} = V_{att}^T \tanh(U_{att}s_{t-1} + W_{att}c_j)$$

$V_{att} \in \mathbb{R}^d$, $U_{att} \in \mathbb{R}^{d \times d}$, $W_{att} \in \mathbb{R}^{d \times d}$ are additional parameters of the model

These parameters will be learned along with the other parameters of the encoder and decoder.
Wait a minute! This model would make a lot of sense if we were given the true’s at training time.

\[ t_j = [0; 0; 0; 5; 0; 5; 0] \]

\[ \hat{t}_j = [0; 0; 0; 35; 0; 35; 0; 1] \]

We could then minimize \( L(\text{true}; \text{pred}) \) in addition to \( L(\cdot) \) as defined earlier. But in practice, it is very hard to get the true’s.
Wait a minute!
- Wait a minute!
- This model would make a lot of sense if we were given the true $\alpha$'s at training time

$$\alpha^{true}_{tj} = [0, 0, 0.5, 0.5, 0]$$

$$\alpha^{pred}_{tj} = [0.1, 0.1, 0.35, 0.35, 0.1]$$
• Wait a minute!

• This model would make a lot of sense if we were given the true $\alpha$'s at training time

\[ \alpha^{true}_{tj} = [0, 0, 0.5, 0.5, 0] \]

\[ \alpha^{pred}_{tj} = [0.1, 0.1, 0.35, 0.35, 0.1] \]

• We could then minimize $\mathcal{L}(\alpha^{true}, \alpha^{pred})$ in addition to $\mathcal{L}(\theta)$ as defined earlier
Wait a minute!

This model would make a lot of sense if were given the true α’s at training time

\[ \alpha_{tj}^{true} = [0, 0, 0.5, 0.5, 0] \]

\[ \alpha_{tj}^{pred} = [0.1, 0.1, 0.35, 0.35, 0.1] \]

We could then minimize \( \mathcal{L}(\alpha^{true}, \alpha^{pred}) \) in addition to \( \mathcal{L}(\theta) \) as defined earlier

But in practice it is very hard to get \( \alpha^{true} \)
For example, in our translation example we would want someone to manually annotate the source words which contribute to every target word.
For example, in our translation example we would want someone to manually annotate the source words which contribute to every target word.

It is hard to get such annotated data.
For example, in our translation example we would want someone to manually annotate the source words which contribute to every target word.

It is hard to get such annotated data.

Then how would this model work in the absence of such data?
It works because it is a better modeling choice. This is a more informed model. We are essentially asking the model to approach the problem in a better (more natural) way. Given enough data it should be able to learn these attention weights just as humans do. That's the hope (and hope is a good thing). And in practice indeed these models work better than the vanilla encoder decoder models.
- It works because it is a better modeling choice
- It works because it is a better modeling choice
- This is a more informed model

**Mitesh M. Khapra**  
CS7015 (Deep Learning) : Lecture 16
- It works because it is a better modeling choice
- This is a more informed model
- We are essentially asking the model to approach the problem in a better (more natural) way
- It works because it is a better modeling choice
- This is a more informed model
- We are essentially asking the model to approach the problem in a better (more natural) way
- Given enough data it should be able to learn these attention weights just as humans do
- It works because it is a better modeling choice
- This is a more informed model
- We are essentially asking the model to approach the problem in a better (more natural) way
- Given enough data it should be able to learn these attention weights just as humans do
- That’s the hope (and hope is a good thing)
- It works because it is a better modeling choice
- This is a more informed model
- We are essentially asking the model to approach the problem in a better (more natural) way
- Given enough data it should be able to learn these attention weights just as humans do
- That’s the hope (and hope is a good thing)
- And in practice indeed these models work better than the vanilla encoder decoder models
Let us revisit the MT model that we saw earlier and answer the same set of questions again (data, encoder, decoder, loss, training algorithm)
**Task:** Machine Translation
Task: Machine Translation
Data: \( \{x_i = \text{source}_i, \ y_i = \text{target}_i\}_{i=1}^{N} \)
- **Task:** Machine Translation
- **Data:** \( \{x_i = \text{source}_i, \ y_i = \text{target}_i\}_{i=1}^N \)
- **Encoder:**
  \[
  h_t = RNN(h_{t-1}, x_t)
  \]
  \[
  s_0 = h_T
  \]
- **Task:** Machine Translation
- **Data:** \( \{x_i = \text{source}_i, \ y_i = \text{target}_i\}_{i=1}^{N} \)
- **Encoder:**
  \[
  h_t = RNN(h_{t-1}, x_t)
  \]
  \[
  s_0 = h_T
  \]
- **Decoder:**
Task: Machine Translation

Data: $\{x_i = source_i, y_i = target_i\}_{i=1}^N$

Encoder:

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

$s_0 = h_T$

Decoder:

$e_{jt} = V_{attn}^{T} tanh(U_{attn} h_j + W_{attn} s_t)$
- **Task:** Machine Translation
- **Data:** \( \{x_i = source_i, \ y_i = target_i\}^N_{i=1} \)
- **Encoder:**
  \[ h_t = RNN(h_{t-1}, x_t) \]
  \[ s_0 = h_T \]
- **Decoder:**
  \[ e_{jt} = V^T_{attn}tanh(U_{attn}h_j + W_{attn}s_t) \]
  \[ \alpha_{jt} = \text{softmax}(e_{jt}) \]
- **Task:** Machine Translation
- **Data:** \( \{x_i = source_i, y_i = target_i\}_{i=1}^N \)
- **Encoder:**
  \[
  h_t = RNN(h_{t-1}, x_t) \\
  s_0 = h_T
  \]
- **Decoder:**
  \[
  e_{jt} = V_{attn}^T \tanh(U_{attn}h_j + W_{attn}s_t) \\
  \alpha_{jt} = \text{softmax}(e_{jt}) \\
  c_t = \sum_{j=1}^T \alpha_{jt}h_j
  \]
Task: Machine Translation

Data: \( \{x_i = source_i, \ y_i = target_i\}_{i=1}^{N} \)

Encoder:
\[ h_t = RNN(h_{t-1}, x_t) \]
\[ s_0 = h_T \]

Decoder:
\[ e_{jt} = V_{attn}^T \tanh(U_{attn}h_j + W_{attn}s_t) \]
\[ \alpha_{jt} = \text{softmax}(e_{jt}) \]
\[ c_t = \sum_{j=1}^{T} \alpha_{jt}h_j \]
\[ s_t = RNN(s_{t-1}, [e(\hat{y}_{t-1}), c_t]) \]
**Task:** Machine Translation

**Data:** \( \{x_i = source_i, \ y_i = target_i\}_{i=1}^{N} \)

**Encoder:**
\[
h_t = RNN(h_{t-1}, x_t) \\
s_0 = h_T
\]

**Decoder:**
\[
e_{jt} = V^T_{\text{attn}} \tanh(U_{\text{attn}} h_j + W_{\text{attn}} s_t) \\
\alpha_{jt} = \text{softmax}(e_{jt})
\]
\[
c_t = \sum_{j=1}^{T} \alpha_{jt} h_j
\]
\[
s_t = RNN(s_{t-1}, [e(\hat{y}_{t-1}), c_t])
\]
\[
\ell_t = \text{softmax}(V s_t + b)
\]

**Parameters:** \( U_{\text{dec}}, \ V, \ W_{\text{dec}}, \ U_{\text{enc}}, \ W_{\text{enc}}, \ b, \ U_{\text{attn}}, \ V_{\text{attn}} \)
- **Task:** Machine Translation
- **Data:** $\{x_i = source_i, \ y_i = target_i\}_{i=1}^N$
- **Encoder:**
  
  $$h_t = RNN(h_{t-1}, x_t)$$
  
  $$s_0 = h_T$$
- **Decoder:**
  
  $$e_{jt} = V^T_{\text{attn}} \tanh(U_{\text{attn}} h_j + W_{\text{attn}} s_t)$$
  
  $$\alpha_{jt} = \text{softmax}(e_{jt})$$
  
  $$c_t = \sum_{j=1}^T \alpha_{jt} h_j$$
  
  $$s_t = RNN(s_{t-1}, [e(\hat{y}_{t-1}), c_t])$$
  
  $$\ell_t = \text{softmax}(V s_t + b)$$
- **Parameters:** $U_{\text{dec}}, V, W_{\text{dec}}, U_{\text{enc}}, W_{\text{enc}}, b, U_{\text{attn}}, V_{\text{attn}}$
- **Loss and Algorithm** remains same
You can try adding an attention component to all the other encoder decoder models that we discussed earlier and answer the same set of questions (data, encoder, decoder, loss, training algorithm)
Can we check if the attention model actually learns something meaningful?
• Can we check if the attention model actually learns something meaningful?
• In other words does it really learn to focus on the most relevant words in the input at the $t$-th timestep?
Can we check if the attention model actually learns something meaningful?
In other words does it really learn to focus on the most relevant words in the input at the \( t \)-th timestep?
We can check this by plotting the attention weights as a heatmap (we will see some examples on the next slide).
Figure: Example output of attention-based summarization system [Rush et al. 2015.]
**Figure:** Example output of attention-based summarization system [Rush et al. 2015.]

**Figure:** Example output of attention-based neural machine translation model [Cho et al. 2015].
Figure: Example output of attention-based summarization system [Rush et al. 2015.]

- The heat map shows a soft alignment between the input and the generated output.

Figure: Example output of attention-based neural machine translation model [Cho et al. 2015].
The heat map shows a soft alignment between the input and the generated output.

Each cell in the heat map corresponds to $\alpha_{tj}$ (i.e., the importance of the $j^{th}$ input word for predicting the $t^{th}$ output word as determined by the model).
Figure: Example output of attention-based video captioning system [Yao et al. 2015.]
Module 16.4: Attention over images
How do we model an attention mechanism for images?

- A man throwing a frisbee in a park
How do we model an attention mechanism for images?

In the case of text we have a representation for every location (time step) of the input sequence
How do we model an attention mechanism for images?

In the case of text we have a representation for every location (time step) of the input sequence.

But for images we typically use representation from one of the fully connected layers.
How do we model an attention mechanism for images?

In the case of text we have a representation for every location (time step) of the input sequence.

But for images we typically use representation from one of the fully connected layers.

This representation does not contain any location information.
How do we model an attention mechanism for images?

In the case of text we have a representation for every location (time step) of the input sequence.

But for images we typically use representation from one of the fully connected layers.

This representation does not contain any location information.

So then what is the input to the attention mechanism?
Well, instead of the fc7 representation we use the output of one of the convolution layers which has spatial information.
- Well, instead of the fc7 representation we use the output of one of the convolution layers which has spatial information.

- For example, the output of the 5th convolutional layer of VGGNet is a $14 \times 14 \times 512$ size feature map.
Well, instead of the fc7 representation we use the output of one of the convolution layers which has spatial information.

For example the output of the $5^{th}$ convolutional layer of VGGNet is a $14 \times 14 \times 512$ size feature map.

We could think of this as 196 locations (each having a 512 dimensional representation).
Well, instead of the fc7 representation we use the output of one of the convolution layers which has spatial information.

For example the output of the 5th convolutional layer of VGGNet is a 14 × 14 × 512 size feature map.

We could think of this as 196 locations (each having a 512 dimensional representation).

The model will then learn an attention over these locations (which in turn correspond to actual locations in the images).
Let us look at some examples of attention over images for the task of image captioning.
**Figure:** Examples of the attention-based model attending to the correct object (*white* indicates the attended regions, *underlines* indicates the corresponding word) [Kyunghyun Cho et al. 2015.]
Module 16.5: Hierarchical Attention
Consider a dialog between a user ($u$) and a bot ($B$)

**Context**

U: Can you suggest a good movie?
B: Yes, sure. How about Logan?
U: Okay, who is the lead actor?

**Response**

B: Hugh Jackman, of course
Consider a dialog between a user ($u$) and a bot ($B$)

The dialog contains a sequence of utterances between the user and the bot

**Context**

U: Can you suggest a good movie?
B: Yes, sure. How about Logan?
U: Okay, who is the lead actor?

**Response**

B: Hugh Jackman, of course
Consider a dialog between a user \( (u) \) and a bot \( (B) \)

- The dialog contains a sequence of utterances between the user and the bot.
- Each utterance in turn is a sequence of words.
Context

U: Can you suggest a good movie?
B: Yes, sure. How about Logan?
U: Okay, who is the lead actor?

Response

B: Hugh Jackman, of course

- Consider a dialog between a user ($u$) and a bot ($B$)
- The dialog contains a sequence of utterances between the user and the bot
- Each utterance in turn is a sequence of words
- Thus what we have here is a “sequence of sequences” as input
Consider a dialog between a user \( (u) \) and a bot \( (B) \)

- The dialog contains a sequence of utterances between the user and the bot
- Each utterance in turn is a sequence of words
- Thus what we have here is a “sequence of sequences” as input
- Can you think of an encoder for such a sequence of sequences?
We could think of a two level hierarchical RNN encoder.
- We could think of a two level hierarchical RNN encoder
- The first level RNN operates on the sequence of words in each utterance and gives us a representation
- We could think of a two level hierarchical RNN encoder
- The first level RNN operates on the sequence of words in each utterance and gives us a representation
- We now have a sequence of utterance representations (red vectors in the image)
- We could think of a two level hierarchical RNN encoder
- The first level RNN operates on the sequence of words in each utterance and gives us a representation
- We now have a sequence of utterance representations (red vectors in the image)
- We can now have another RNN which encodes this sequence and gives a single representation for the sequences of utterances
We could think of a two level hierarchical RNN encoder.

The first level RNN operates on the sequence of words in each utterance and gives us a representation.

We now have a sequence of utterance representations (red vectors in the image).

We can now have another RNN which encodes this sequence and gives a single representations for the sequences of utterances.

The decoder can then produce an output sequence conditioned on this utterance.
Politics is the process of making decisions applying to all members of each group. More narrowly, it refers to achieving and ...

Let us look at another example
Politics is the process of making decisions applying to all members of each group. More narrowly, it refers to achieving and ... 

- Let us look at another example
- Consider the task of document classification or summarization
Politics is the process of making decisions applying to all members of each group. More narrowly, it refers to achieving and ...
Politics is the process of making decisions applying to all members of each group. More narrowly, it refers to achieving and …

Let us look at another example
Consider the task of document classification or summarization
A document is a sequence of sentences
Each sentence in turn is a sequence of words
Politics is the process of making decisions applying to all members of each group. More narrowly, it refers to achieving and ...
Politics is the process of making decisions applying to all members of each group. More narrowly, it refers to achieving and ...

- Let us look at another example
- Consider the task of document classification or summarization
- A document is a sequence of sentences
- Each sentence in turn is a sequence of words
- We can again use a hierarchical RNN to model this
Politics is the process of making decisions applying to all members of each group. More narrowly, it refers to achieving and ...
Politics is the process of making decisions applying to all members of each group. More narrowly, it refers to achieving and ...
Politics is the process of making decisions applying to all members of each group. More narrowly, it refers to achieving and ...
Politics is the process of making decisions applying to all members of each group. More narrowly, it refers to achieving and ...
Politics is the process of making decisions applying to all members of each group. More narrowly, it refers to achieving and ...

- **Data:** \( \{Document_i, \text{class}_i\}^N_{i=1} \)
- **Word level (1) encoder:**
  \[ h^1_{ij} = RNN(h^1_{ij-1}, w_{ij}) \]
  \[ s_i = h^1_{iT_i} \quad [T \text{ is length of sentence } i] \]
- **Sentence level (2) encoder:**
  \[ h^2_i = RNN(h^2_{i-1}, s_i) \]
  \[ s = h^2_K \quad [K \text{ is number of sentences}] \]

```
Politics
is decisions
applying to
group
More narrowly and
```
Politics is the process of making decisions applying to all members of each group. More narrowly, it refers to achieving and …

- **Data:** \( \{Document_i, class_i\}_{i=1}^N \)
- **Word level (1) encoder:**
  \[
  h_{ij}^1 = RNN(h_{ij-1}^1, w_{ij})
  \]
  \[
  s_i = h_{iT_i}^1 \quad [T \text{ is length of sentence } i]
  \]
- **Sentence level (2) encoder:**
  \[
  h_i^2 = RNN(h_{i-1}^2, s_i)
  \]
  \[
  s = h_K^2 \quad [K \text{ is number of sentences}]
  \]
- **Decoder:**
  \[
P(y|\text{document}) = \text{softmax}(Vs + b)
  \]
Politics is the process of making decisions applying to all members of each group. More narrowly, it refers to achieving and ...
Politics is the process of making decisions applying to all members of each group. More narrowly, it refers to achieving and ...
Politics is the process of making decisions applying to all members of each group. More narrowly, it refers to achieving and ...
How would you model attention in such a hierarchical encoder decoder model?
How would you model attention in such a hierarchical encoder decoder model?

We need attention at two levels:
How would you model attention in such a hierarchical encoder decoder model?

- We need attention at two levels
- First we need to attend to important (most informative) words in a sentence
- Then we need to attend to important (most informative) sentences in a document

**Figure: Hierarchical Attention Network**

[Yang et al.]
How would you model attention in such a hierarchical encoder decoder model?

- We need attention at two levels
- First we need to attend to important (most informative) words in a sentence
- Then we need to attend to important (most informative) sentences in a document

Figure: Hierarchical Attention Network [Yang et al.]
How would you model attention in such a hierarchical encoder decoder model?

- We need attention at two levels
- First we need to attend to important (most informative) words in a sentence
- Then we need to attend to important (most informative) sentences in a document
- Let us see how to model this

**Figure: Hierarchical Attention Network [Yang et al.]**
Figure: Hierarchical Attention Network
[Yang et al.]
Data: \( \{\text{Document}_i, \text{class}_i\}_{i=1}^{N} \)

Word level (1) encoder:

\[ h_{ij} = \text{RNN}(h_{ij-1}, w_{ij}) \]
\[ u_{ij} = \tanh(W_w h_{ij} + b_w) \]
\[ \alpha_{ij} = \frac{\exp(u_{ij}^T u_{wj})}{\sum_t \exp(u_{it}^T u_{wj})} \]
\[ s_i = \sum_j \alpha_{ij} h_{ij} \]
Figure: Hierarchical Attention Network
[Yang et al.]

- **Data:** \( \{Document_i, class_i\}_{i=1}^N \)
- **Word level (1) encoder:**
  \[
  h_{ij} = RNN(h_{ij-1}, w_{ij}) \\
  u_{ij} = \tanh(W_w h_{ij} + b_w) \\
  \alpha_{ij} = \frac{\exp(u_{ij}^T u_w)}{\sum_t \exp(u_{it}^T u_w)} \\
  s_i = \sum_j \alpha_{ij} h_{ij}
  \]
- **Sentence level (2) encoder:**
  \[
  h_i = RNN(h_{i-1}, s_i) \\
  u_i = \tanh(W_s h_i + b_s) \\
  \alpha_i = \frac{\exp(u_i^T u_s)}{\sum_i \exp(u_i^T u_s)} \\
  s = \sum_i \alpha_i h_i
  \]
**Decoder:**

\[ P(y|\text{document}) = \text{softmax}(Vs + b) \]

**Figure:** Hierarchical Attention Network
[Yang et al..]
Figure: Hierarchical Attention Network
[Yang et al.]

- **Decoder:**
  \[ P(y|\text{document}) = \text{softmax}(Vs + b) \]

- **Parameters:**
  \[ W_w, W_s, V, b_w, b_s, b, u_w, u_s \]
- **Decoder:**
  
  \[ P(y|\text{document}) = \text{softmax}(Vs + b) \]

- **Parameters:**
  
  \[ W_w, W_s, V, b_w, b_s, b, w_w, w_s \]

- **Loss:** cross entropy

**Figure:** Hierarchical Attention Network

[Yang et al..]
Figure: Hierarchical Attention Network
[Yang et al..]

- **Decoder:**
  
  \[ P(y|\text{document}) = \text{softmax}(Vs + b) \]

- **Parameters:**
  \[ W_w, W_s, V, b_w, b_s, b, u_w, u_s \]

- **Loss:** cross entropy

- **Algorithm:** Gradient Descent and backpropagation