Automatic Speech Recognition II

안인규 (Inkyu An)

Speech And Audio Recognition
(오디오 음성인식)

https://mairlab-km.github.io/

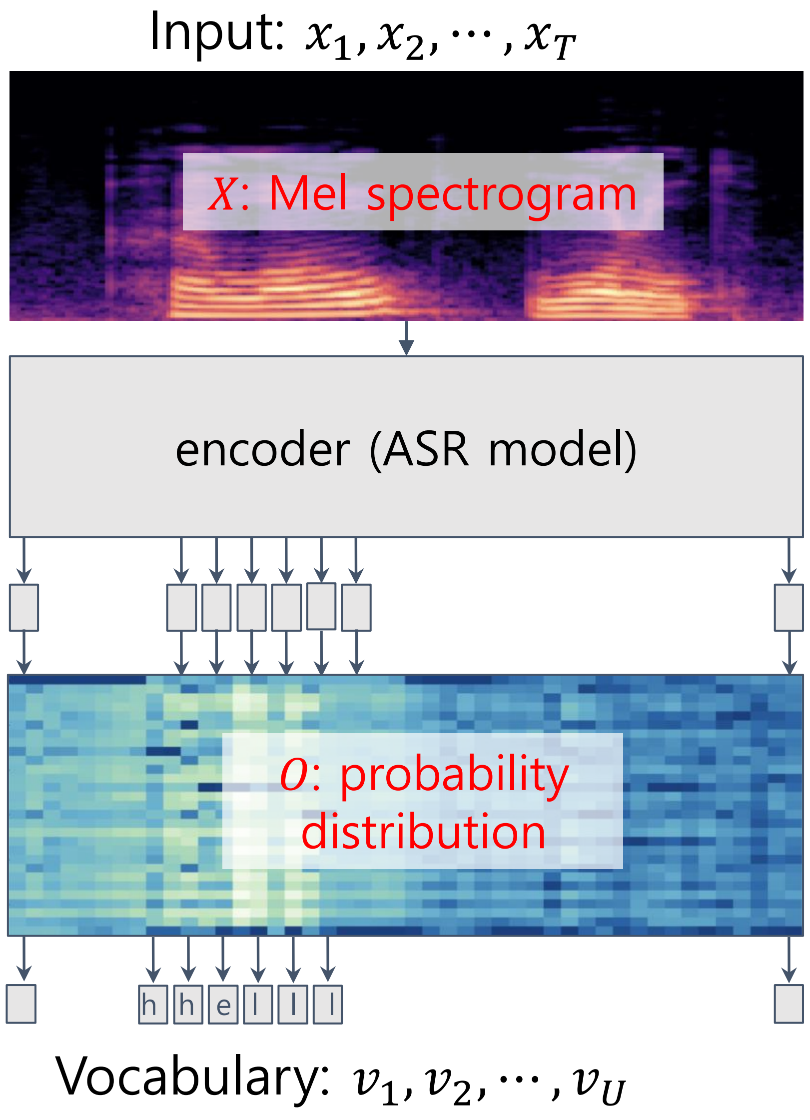
This lecture material refers to
https://github.com/yandexdataschool/speech_course?tab=readme-ov-file and
https://github.com/markovka17/dla

In the previous lecture

- We wanted to build an ASR model
- We have input $x_1, \cdots, x_T$ and label $y_1, \cdots, y_U$

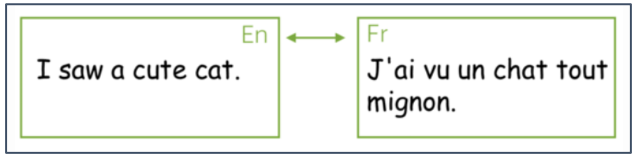
- Solution:
  - Build a model that predicts distribution over vocabulary: including alphabet (or BPE vocabulary), space, and blank
  - We still don't know the alignment x_t and y_u, but we train the model to maximize <u>all valid paths</u> in the output matrix
  - However, by construction, the model does not have access to its previous predictions when making the next prediction

hell o	??? ?	hell ?
target	What does the ctc model see?	What we want the model to see

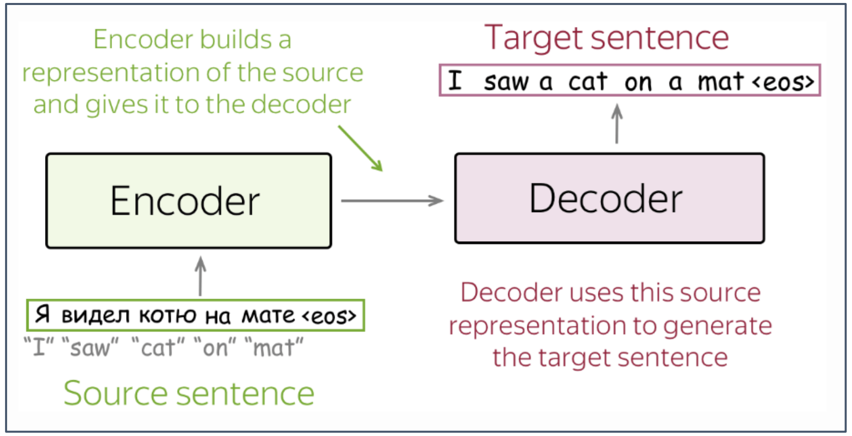


Alignment problem, NLP case

- Task: machine translation
  - We want to transform $x_1, \cdots, x_T$ to $y_1, \cdots, y_U$ where $T \neq U$ usually



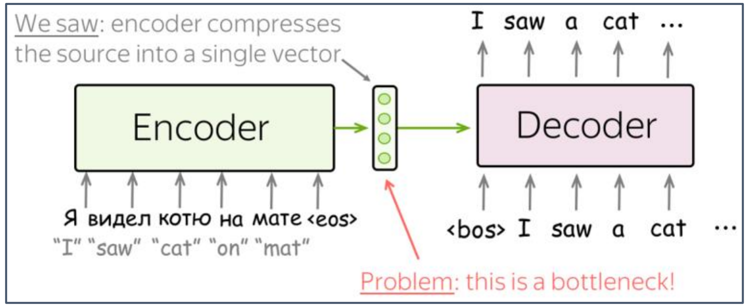
Seq2seq, NeurIPS 2014



https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Alignment problem, NLP case

- Fixed source representation is suboptimal:
  - For the encoder, it is hard to compress the sentence
  - For the decoder, different information may be relevant at different steps

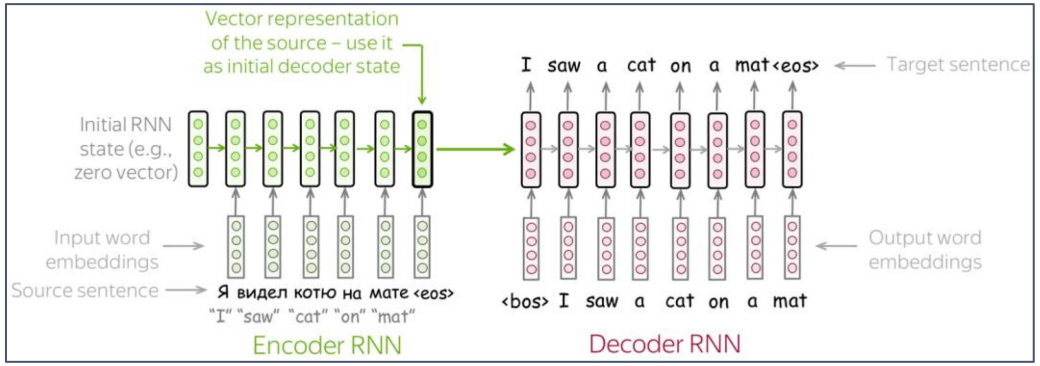


입력 문장을 하나의 고정된 벡터로만 표현하는 방식은 최적이 아닐 수 있음.

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Alignment problem, NLP case

- **Idea**
  - Use a weighted sum of the original embeddings and recalculate the weights for each decoding step according to previously predicted tokens and the original embeddings
  - **Neural Machine Translation by Jointly Learning to Align and Translate**, Dzmitry Bahdanau et. al., 2014



Attention output: $c^{(t)} = a_1^{(t)} s_1 + a_2^{(t)} s_2 + \cdots + a_m^{(t)} s_m = \sum_{k=1}^{m} a_k^{(t)} s_k$

"source context for decoder step t"

(weighted sum)

Attention weights: $a_k^{(t)} = \frac{\exp(\text{score}(h_t, s_k))}{\sum_{i=1}^{m} \exp(\text{score}(h_t, s_i))}, \mathrm{k} = 1..\mathrm{m}$

"attention weight for source token k at decoder step t"

(softmax)

Attention scores: $\text{score}(h_t, s_k), \mathrm{k} = 1..\mathrm{m}$

"How relevant is source token k for target step t?"

Attention input: $s_1, s_2, \dots, s_m$ all encoder states; h_t one decoder state

하나의 Output을 출력하기 위해,
모든 입력과의 관계를 고려함.

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Attention mechanism

Attention output $c^{(t)} = a_1^{(t)} s_1 + a_2^{(t)} s_2 + \dots + a_m^{(t)} s_m = \sum_{k=1}^{m} a_k^{(t)} s_k$

"source context for decoder step t"

(weighted sum)

Attention weights $a_k^{(t)} = \dfrac{\exp(\mathrm{score}(h_t, s_k))}{\sum_{i=1}^{m} \exp(\mathrm{score}(h_t, s_i))}, \mathrm{k} = 1..\mathrm{m}$

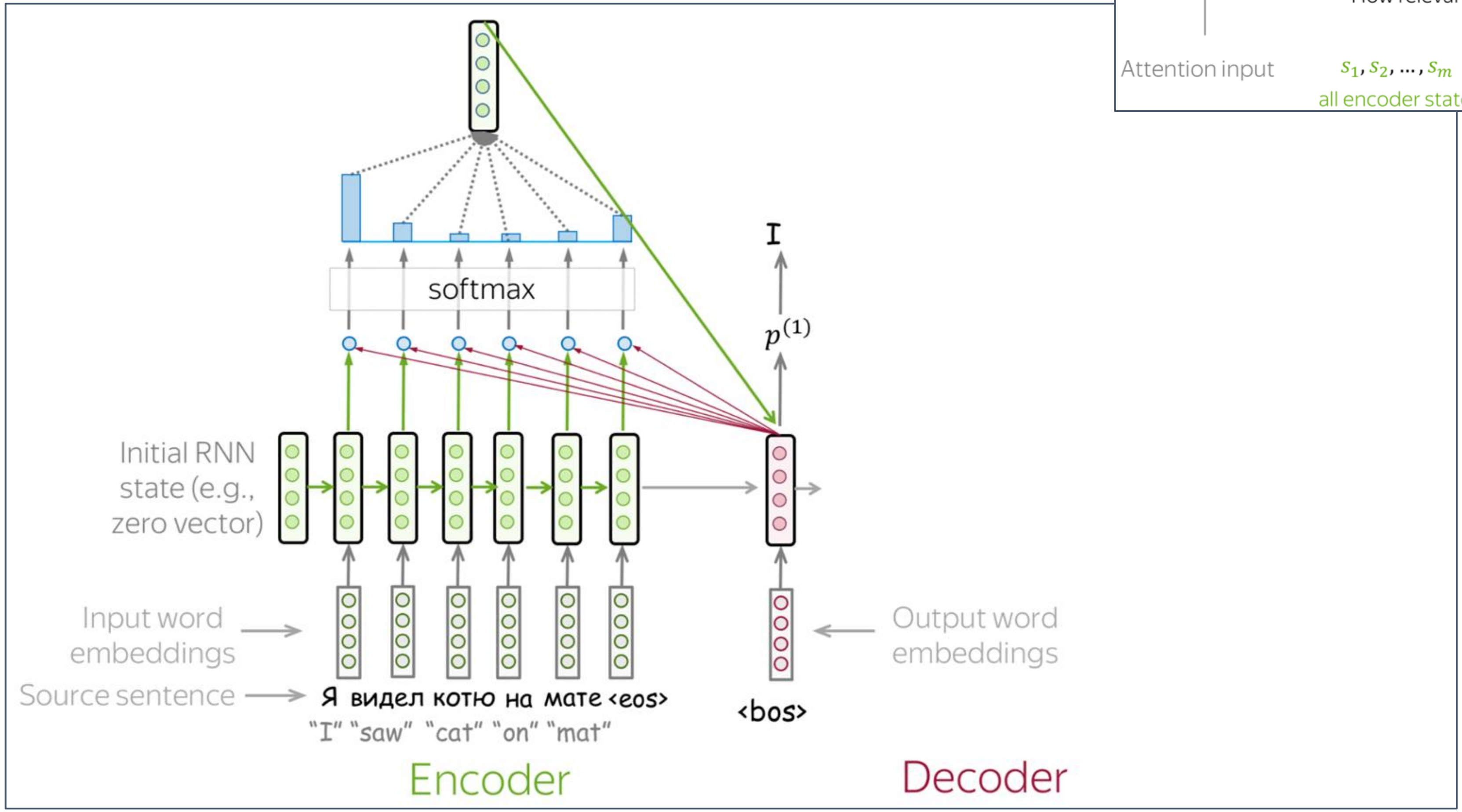
"attention weight for source token k at decoder step t"

(softmax)

Attention scores $\mathrm{score}(h_t, s_k), \mathrm{k} = 1..\mathrm{m}$

"How relevant is source token k for target step t?"

Attention input $s_1, s_2, \dots, s_m$ all encoder states; h_t one decoder state



https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Attention mechanism

Attention output $c^{(t)} = a_1^{(t)} s_1 + a_2^{(t)} s_2 + \cdots + a_m^{(t)} s_m = \sum_{k=1}^{m} a_k^{(t)} s_k$

"source context for decoder step t"

(weighted sum)

Attention weights $a_k^{(t)} = \frac{\exp(\text{score}(h_t, s_k))}{\sum_{i=1}^{m} \exp(\text{score}(h_t, s_i))}, \mathrm{k} = 1..\mathrm{m}$

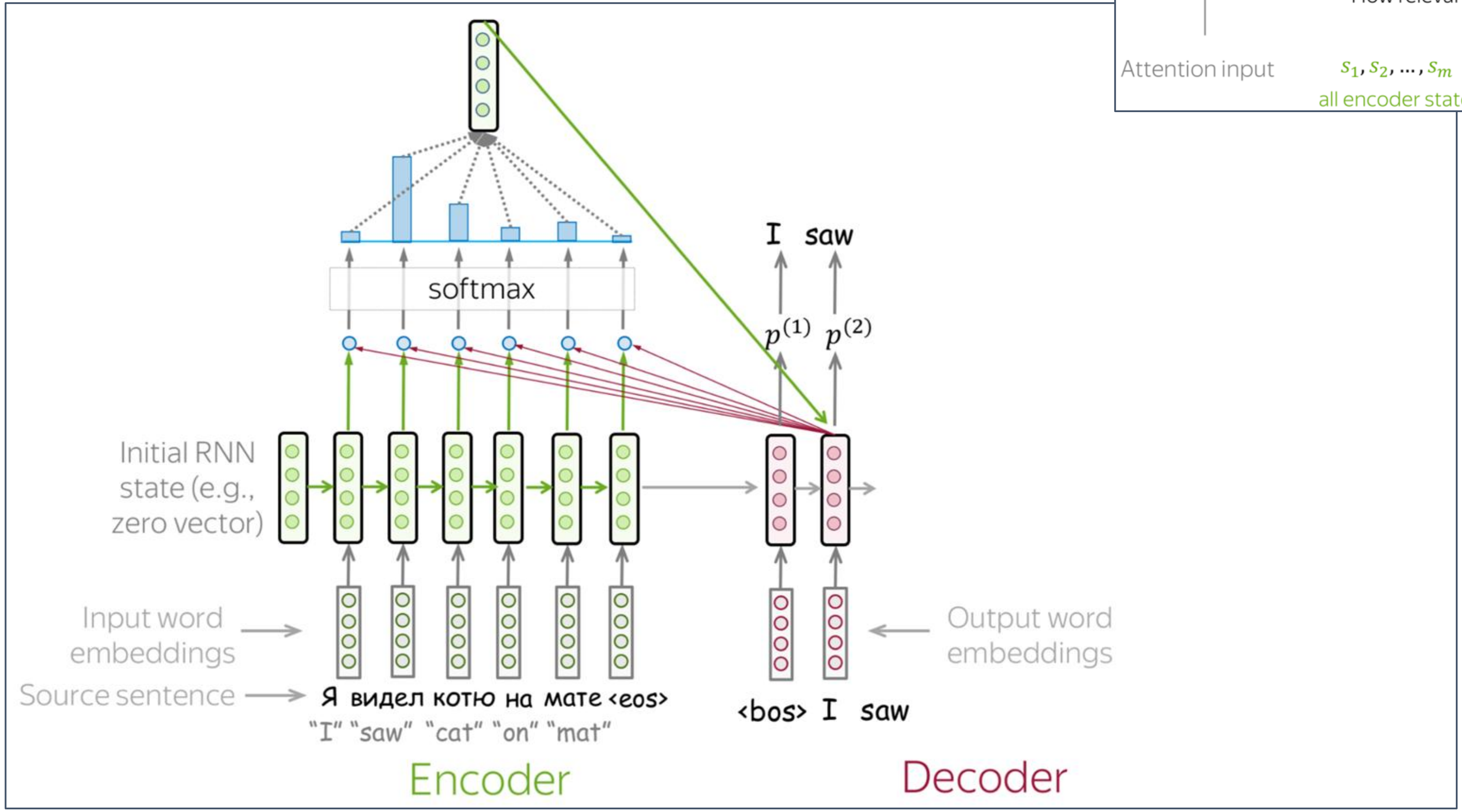
"attention weight for source token k at decoder step t"

(softmax)

Attention scores $\text{score}(h_t, s_k), \mathrm{k} = 1..\mathrm{m}$

"How relevant is source token k for target step t?"

Attention input $s_1, s_2, \dots, s_m$ all encoder states; h_t one decoder state



https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Attention mechanism

Attention output $c^{(t)} = a_1^{(t)} s_1 + a_2^{(t)} s_2 + \cdots + a_m^{(t)} s_m = \sum_{k=1}^{m} a_k^{(t)} s_k$

"source context for decoder step t"

(weighted sum)

Attention weights $a_k^{(t)} = \frac{\exp(\mathrm{score}(h_t, s_k))}{\sum_{i=1}^{m} \exp(\mathrm{score}(h_t, s_i))}, \mathrm{k} = 1..\mathrm{m}$

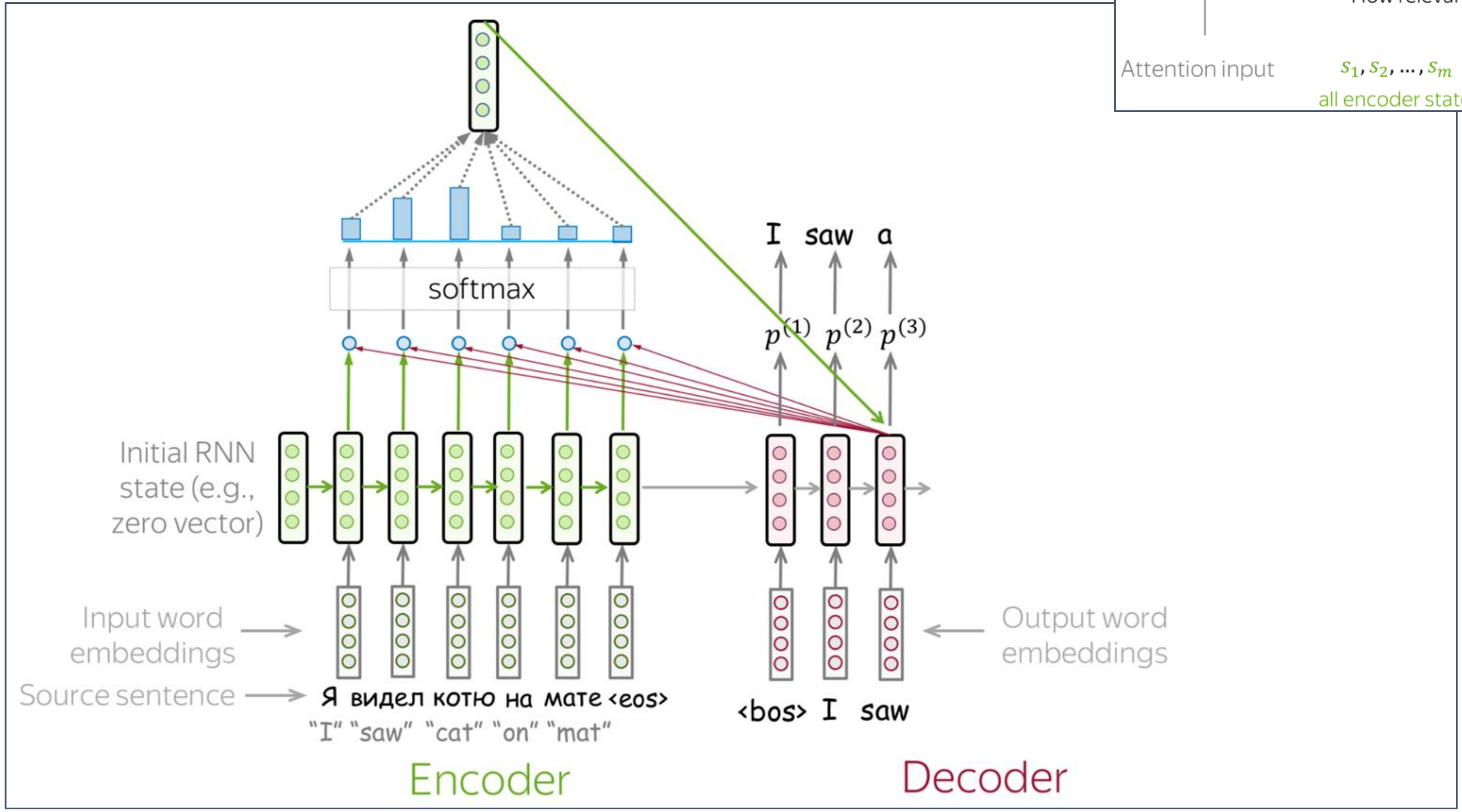
"attention weight for source token k at decoder step t"

(softmax)

Attention scores $\mathrm{score}(h_t, s_k), \mathrm{k} = 1..\mathrm{m}$

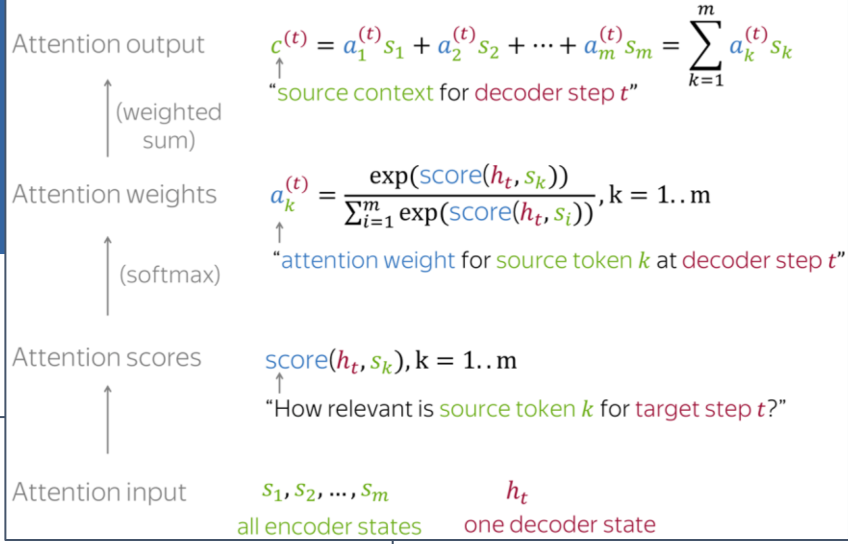
"How relevant is source token k for target step t?"

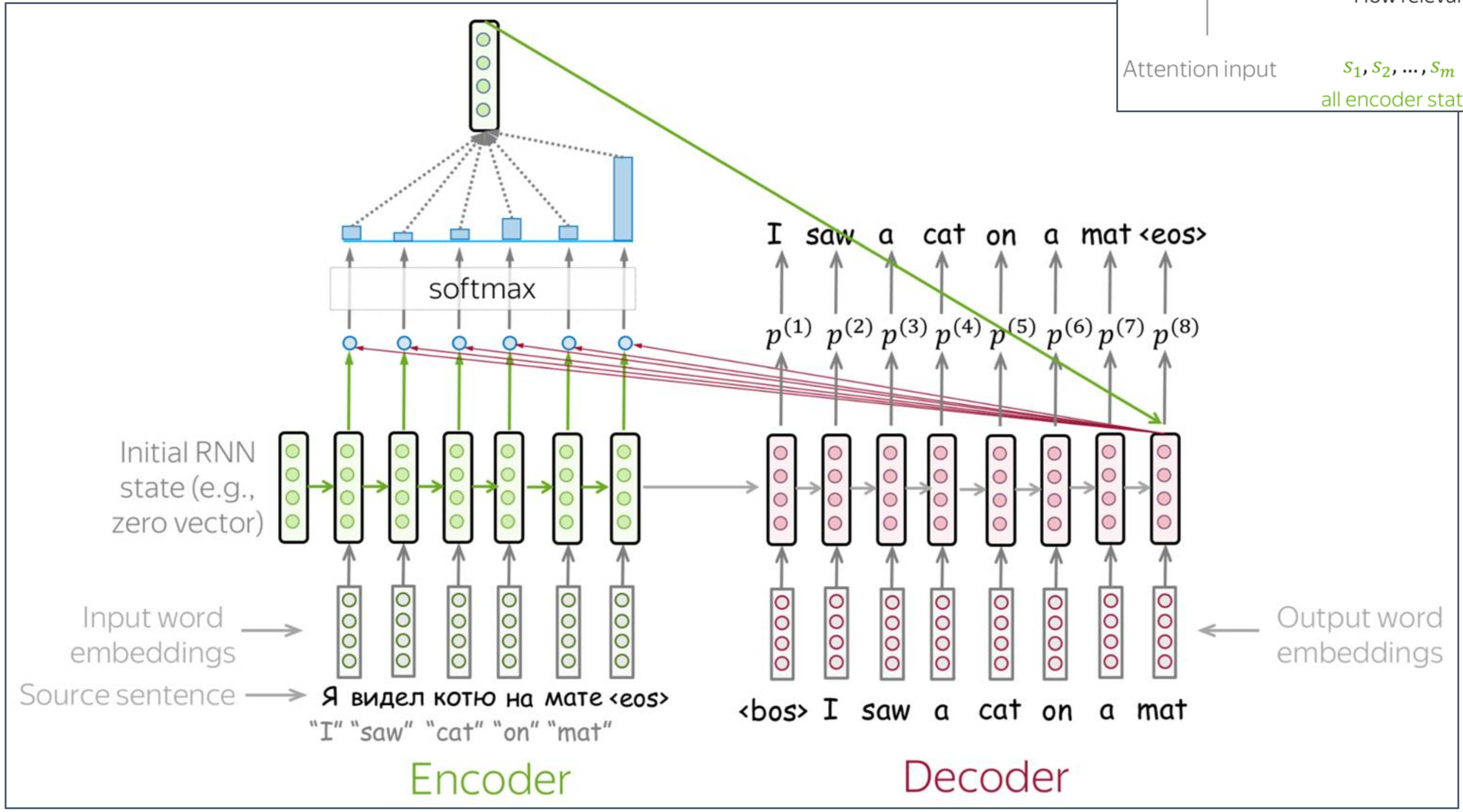
Attention input $s_1, s_2, \ldots, s_m$ all encoder states; h_t one decoder state



https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

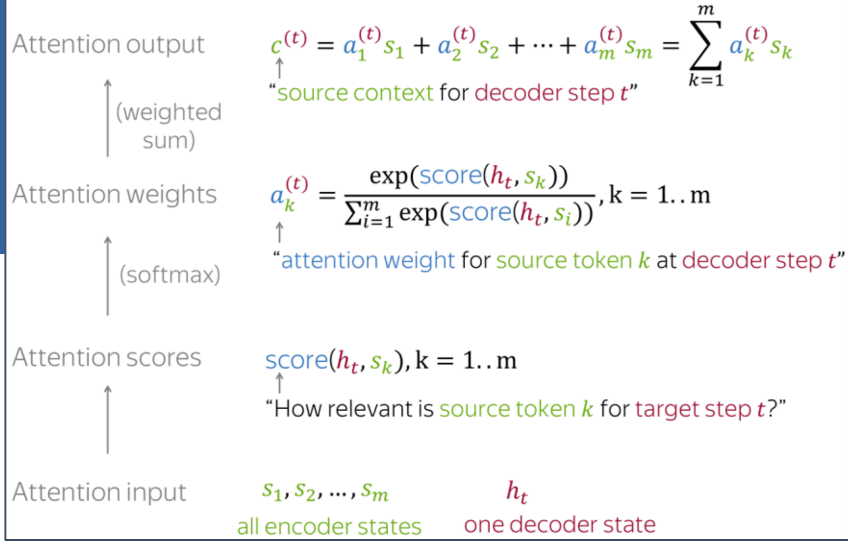
Attention mechanism

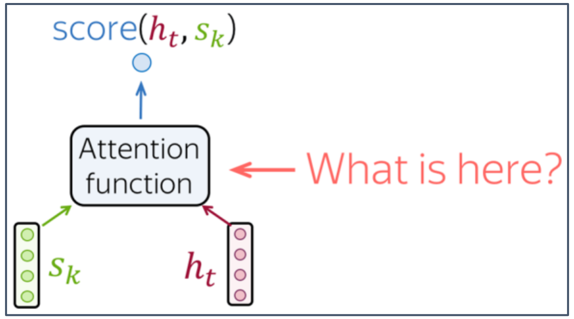


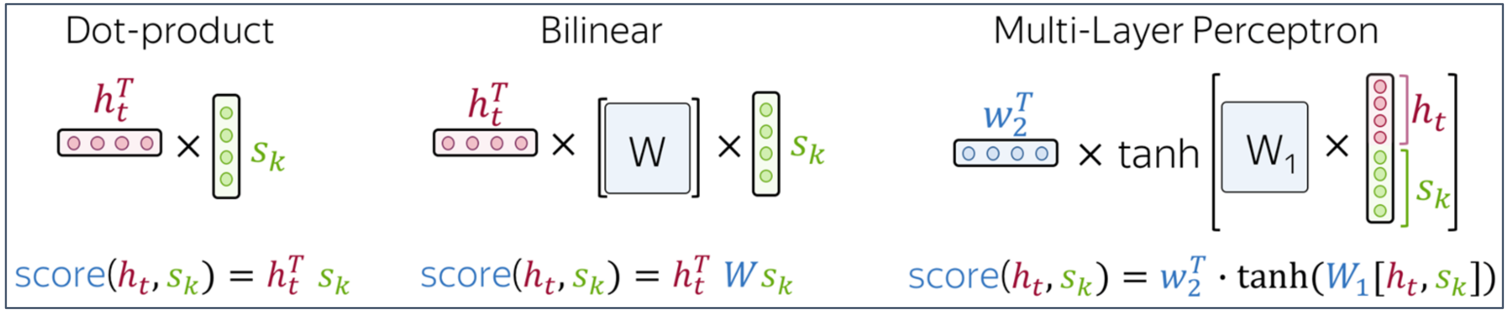


https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Attention mechanism







Various attention functions

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Attention mechanism

Attention output $c^{(t)} = a_1^{(t)} s_1 + a_2^{(t)} s_2 + \dots + a_m^{(t)} s_m = \sum_{k=1}^{m} a_k^{(t)} s_k$

"source context for decoder step t"

(weighted sum)

Attention weights $a_k^{(t)} = \frac{\exp(\text{score}(h_t, s_k))}{\sum_{i=1}^{m} \exp(\text{score}(h_t, s_i))}, \mathrm{k} = 1..\mathrm{m}$

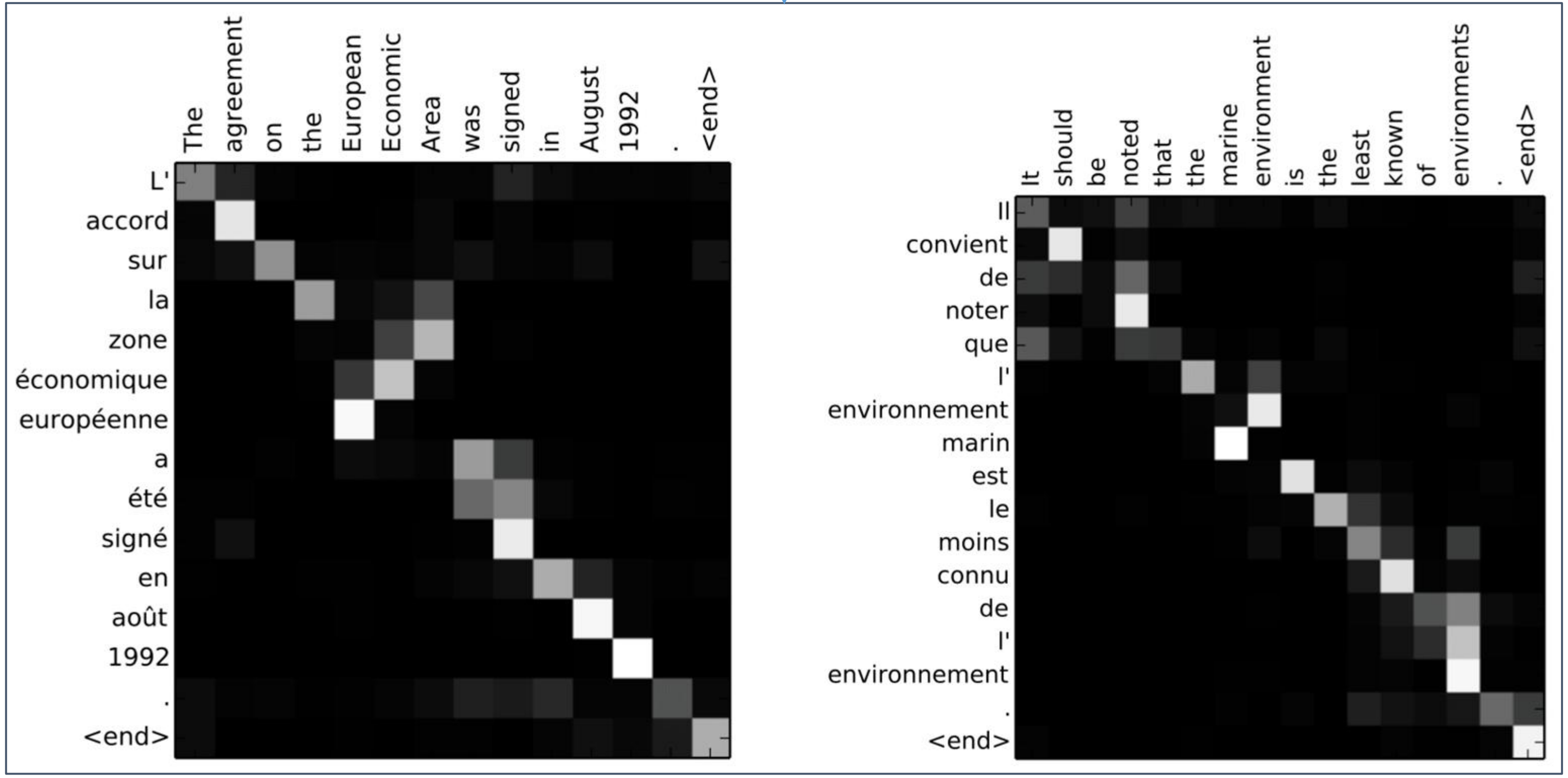
"attention weight for source token k at decoder step t"

(softmax)

Attention scores $\text{score}(h_t, s_k), \mathrm{k} = 1..\mathrm{m}$

"How relevant is source token k for target step t?"

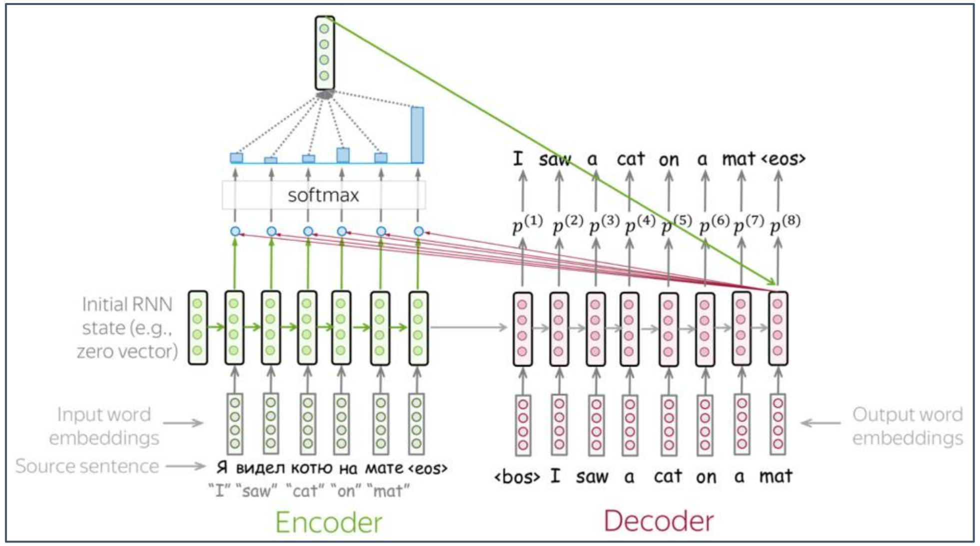
Attention input $s_1, s_2, \dots, s_m$ all encoder states; h_t one decoder state



https://arxiv.org/abs/1409.0473
https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Attention mechanism

- How to train?
  - At each moment in time, the model predicts a probability distribution over our vocabulary given the source context and previously predicted target tokens
  - Minimize the <u>cross entropy</u> between the target and model distribution



$$s_1,\ s_2 \dots s_m = \text{Encoder}(\ \text{Emb}\,(\text{я видел котю на мате})\ )\ \text{- source context}$$

$$p^{(t)}_{\text{model}} \in \mathbb{R}^{\text{vocab\_size}}$$

$$p^{(t)}_{\text{model}}\left(* \ ||\ \hat{y_0},\ \hat{y_1},\ \dots,\ \hat{y_{t-1}},\ s_1,\ s_2 \dots s_m\right) =$$

$$= \text{Decoder}(\ \hat{y_0},\ \hat{y_1},\ \dots,\ \hat{y_{t-1}},\ s_1,\ s_2 \dots s_m)$$

https://arxiv.org/abs/1409.0473
https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Attention mechanism

	i=1	i=2	i=3
cat	0.1	0.9	0.05
dog	0.7	0.05	0.15
bird	0.2	0.05	0.8

- How to train?
  - Minimize the cross entropy between the target and model distribution
    - $\mathcal{L}_{ce} = -\sum_i p(i) \cdot \log q(i)$
    - $p(i)$: the ground truth distribution
    - $q(i)$: the predicted distribution

Model이 $q(i)$를 예측

→ $\boldsymbol{q(i)} = \{\mathbf{0.1, 0.7, 0.2}\}$

Vocab={cat, dog, bird}

i=1일 때, 정답이 dot

$\boldsymbol{p(i)} = [\mathbf{0, 1, 0}]$

$$\mathcal{L}_{ce} = -\sum_i p(i) \cdot \log q(i)$$
$$= -(0 \cdot \log 0.1 + 1 \cdot \log 0.7 + 0 \cdot \log 0.2)$$
$$= -\log 0.7$$

Cross Entropy는 정보량의 기댓값:

→ 즉, 두 분포의 분포가 다를 수록 정보량이 많다 (새로운 정보가 많다)

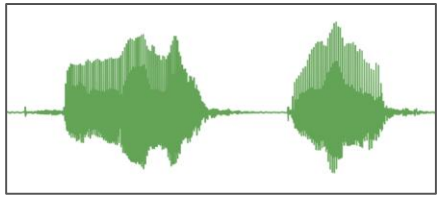
→ 두 분포가 다를 수록 Cross Entropy의 크기는 커진다.

https://arxiv.org/abs/1409.0473

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Alignment problem, Speech case

- **Task:** Automatic Speech Recognition (ASR)
  - We want to transform $x_1, \cdots, x_T$ to $y_1, \cdots, y_U$ where $T \neq U$ usually



hello world

$$s_1,\ s_2 \ldots s_m = \text{AudioEncoder(Spec(wav)) - source context}$$

$$p^{(t)}_{\text{model}} \in \mathbb{R}^{\text{vocab\_size}}$$

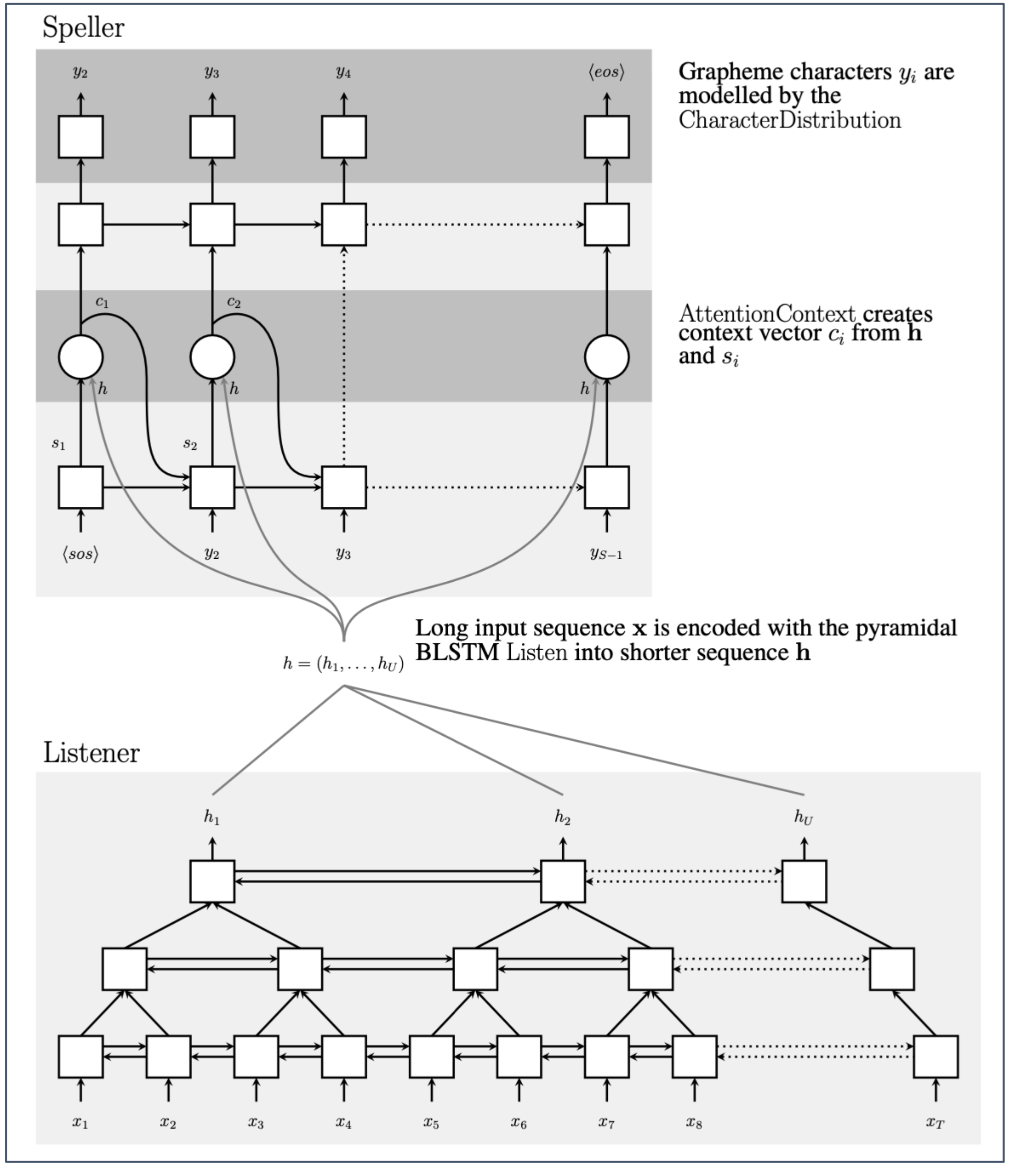
$$p^{(t)}_{\text{model}} \left(* \ \| \ \hat{y_0}\,,\ \hat{y_1}\,,\ \ldots\,,\hat{y_{t-1}}\,,\ s_1,\ s_2 \ldots\ s_m \right) =$$

$$= \text{Decoder}(\ \hat{y_0}\,,\ \hat{y_1}\,,\ \ldots\,,\hat{y_{t-1}}\,,\ s_1,\ s_2 \ldots\ s_m)$$

https://arxiv.org/abs/1409.0473
https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Listen, Attend and Speech (LAS)

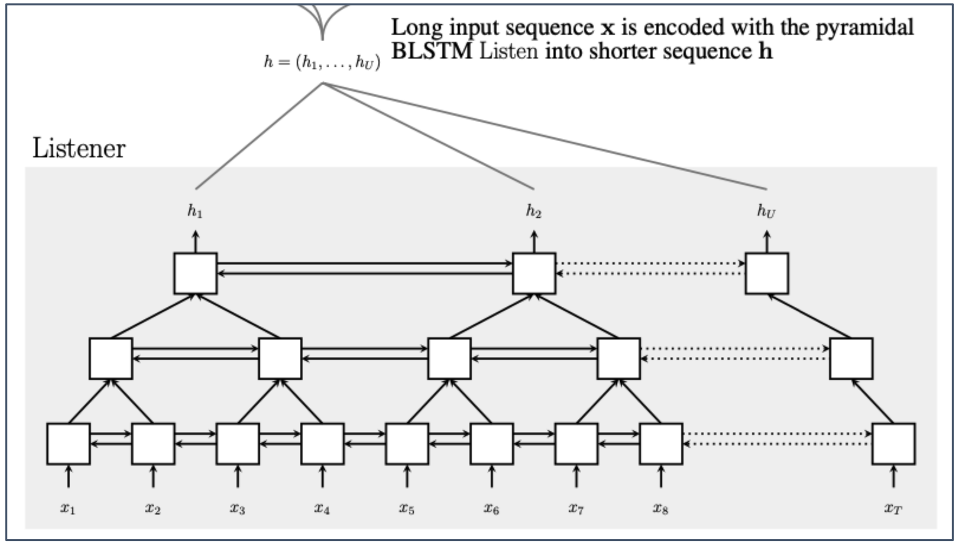
- **Listen, Attend and Spell**, William Chan et al., Carnegie Mellon University, Google Brain, 2015
  - The network produces character sequences without making any independence assumptions between the characters.
  - Modules:
    - Listener - encodes information about the input audio
    - Attend - tells Speller what parts of input are relevant during the current decoding step
    - Speller- decodes the latent representation into transcription

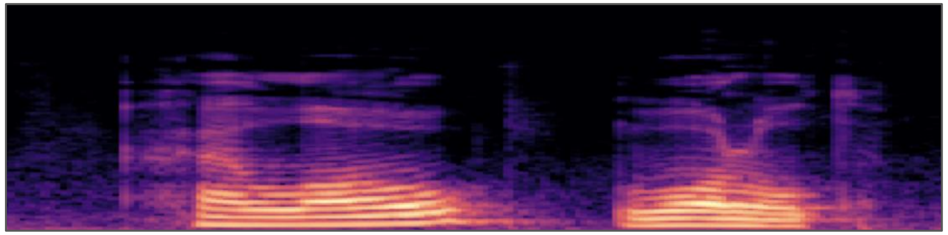


https://arxiv.org/abs/1508.01211

Listen, Attend and Speech (LAS)

- In fact, we can use arbitrary encoder (for example Conformer), but that was in 2015
- Thus, we use LSTM in the encoder and a pyramid structure for subsampling since we need to reduce the time dimension:
  - RNN can't handle very long sequences
  - Reduce the time dimension for Attend for selecting the most relevant information
  - Better convergence in much shorter time
- 8x time reduction (4x in the image)





https://arxiv.org/abs/1508.01211

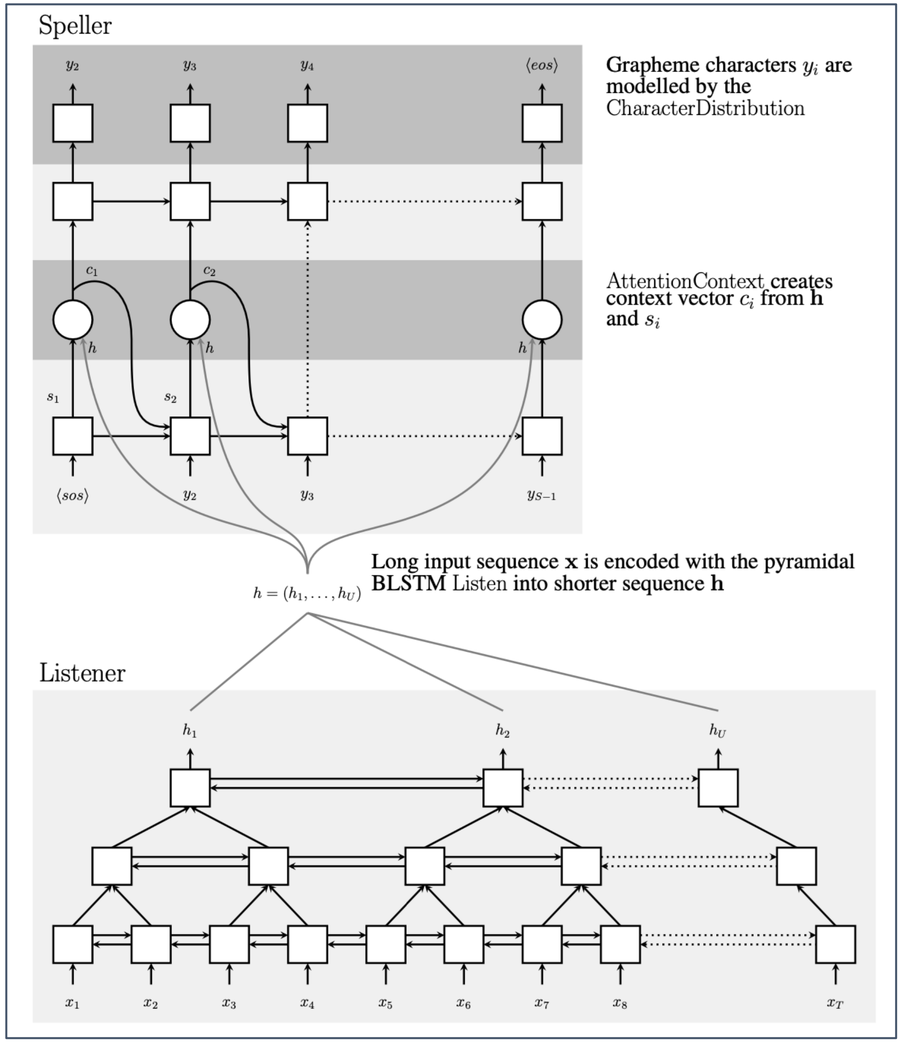
Listen, Attend and Speech (LAS)

- by argmax (greedy decoding):

$$
\begin{aligned}
p^{(t)}_{\text{model}}\left(* \;||\; \hat{y_0}, \hat{y_1}, \ldots, \hat{y_{t-1}}, h_1, h_2 \ldots h_U\right) &= \\
= \text{Decoder}(\hat{y_0}, \hat{y_1}, \ldots, \hat{y_{t-1}}, h_1, h_2 \ldots h_U)
\end{aligned}
$$

$$
\begin{aligned}
\hat{y_1} &= \arg\max p^{(1)}_{\text{model}} \;(* \,||\, y_0, h_1, h_2 \ldots h_U) \;\; (y_0 \text{ bos emb}) \\
\hat{y_2} &= \arg\max p^{(2)}_{\text{model}} \;(* \,||\, y_0, \hat{y_1}, h_1, h_2 \ldots h_U) \\
&\ldots
\end{aligned}
$$

- by usual beam search:
  - Expand beam
  - Truncate beam



Listen, Attend and Speech (LAS)

- LAS quality

Method	Year	WER (test -clean)	WER (test-other)
Human	~ 200 000 b.c.	5.83	12.69
Deep Speech 2	2015	5.15	12.73
LAS	2015	----	----
LAS without SpecAugment*	2019	**3.2**	**9.8**

https://arxiv.org/abs/1508.01211

Listen, Attend and Speech (LAS)

- CTC vs. LAS Inference

Method	CTC	LAS
Streaming	yes	no
Context	no	yes
Argmax Complexity	O(enc + dec)	O(enc + T * dec)
Beam Search Complexity	O(enc + dec + T * bs)	O(enc + T * bs * dec)

https://arxiv.org/abs/1508.01211

Listen, Attend and Speech (LAS)

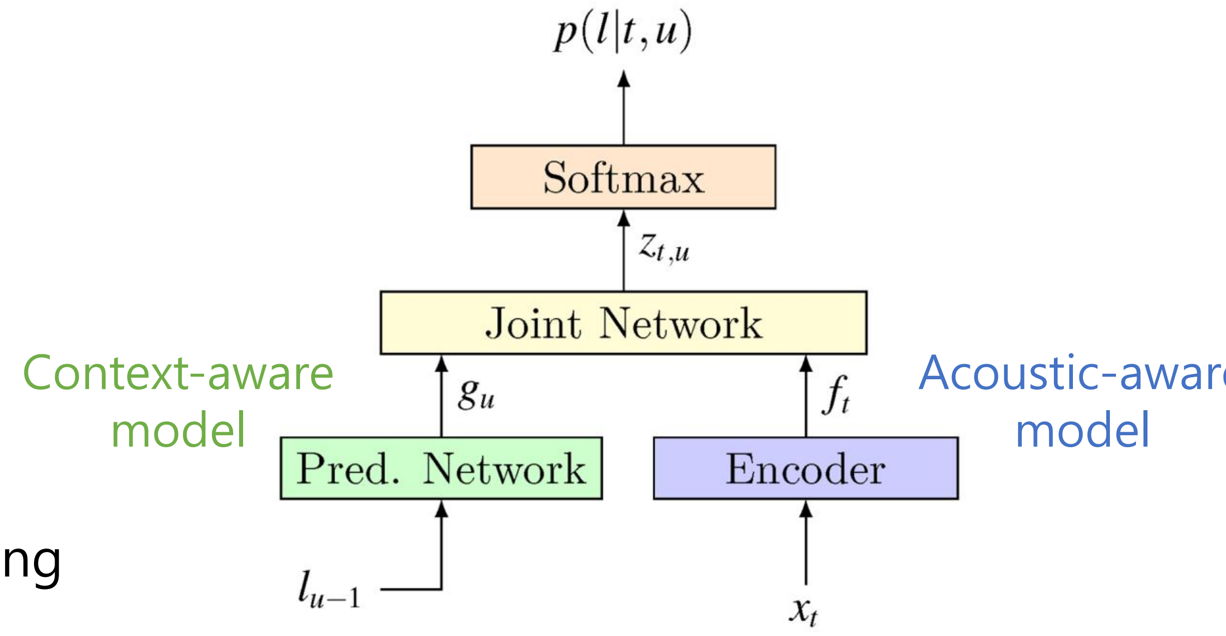
- Can do better?

Method	CTC	LAS	?
Streaming	yes	no	yes
Context	no	yes	yes
Argmax Complexity	O(enc + dec)	O(enc + T * dec)	?
Beam Search Complexity	O(enc + dec + T * bs)	O(enc + T * bs * dec)	?

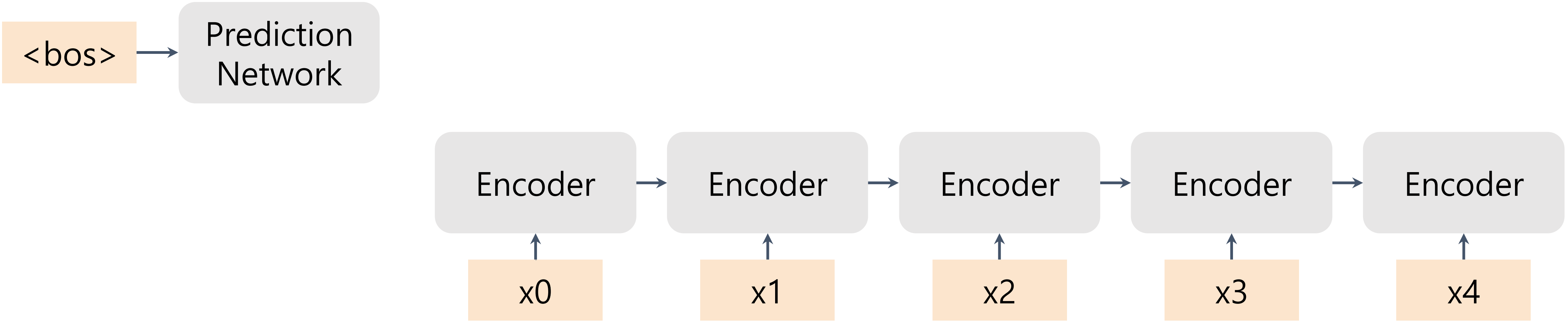
https://arxiv.org/abs/1508.01211

RNN-Transducer (RNN-t)

- **Sequence Transduction with Recurrent Neural Networks**, Alex Graves, University of Toronto, 2012
  - Motivation or we want:
    - A context-dependent model
    - A streamable model
  - Architecture:
    - **Audio encoder** - for acoustic features
    - **Prediction network** - for text processing
    - **Joint network** - for predicting with audio-language context

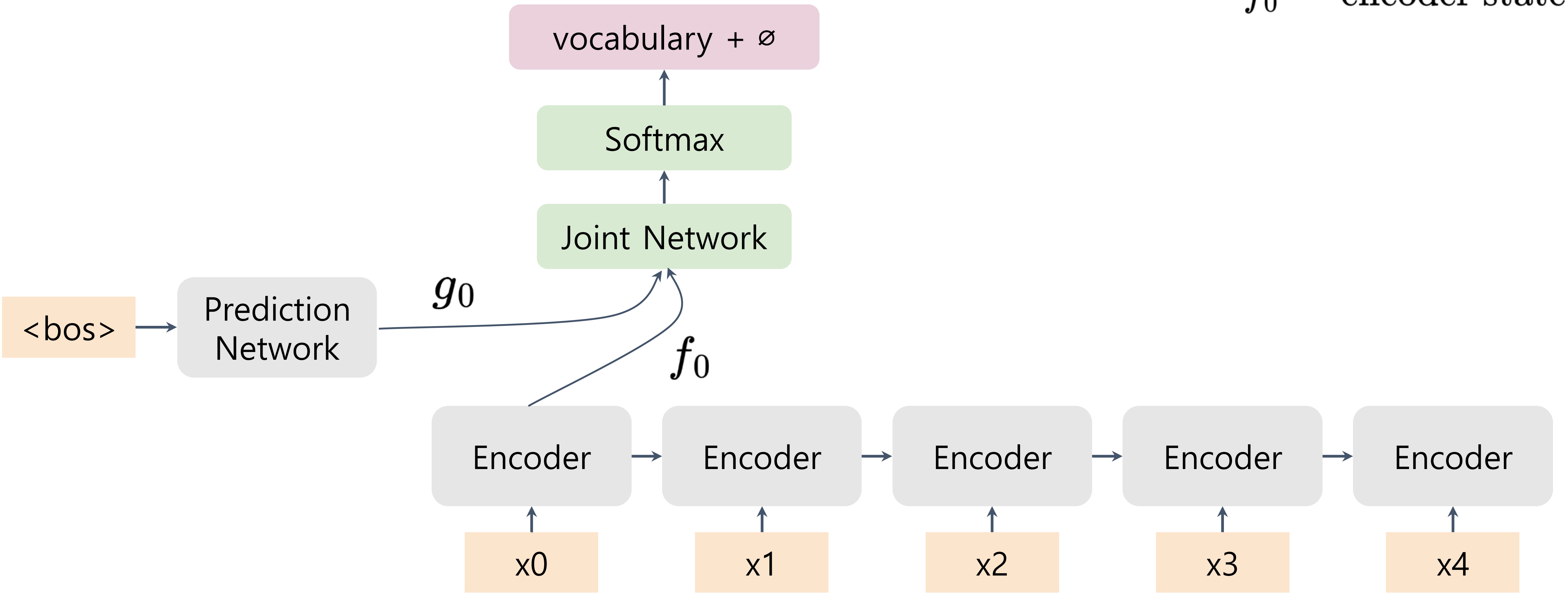


RNN-t Inference

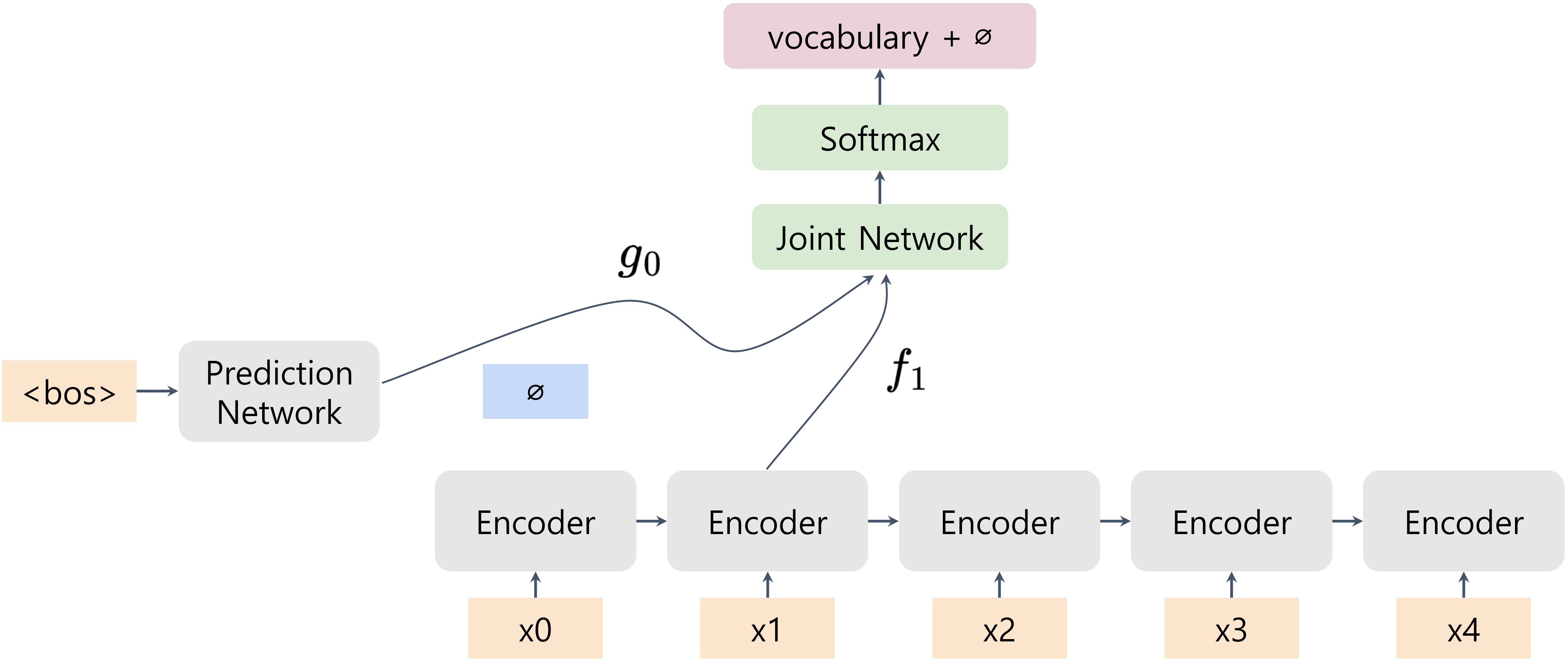


RNN-t Inference

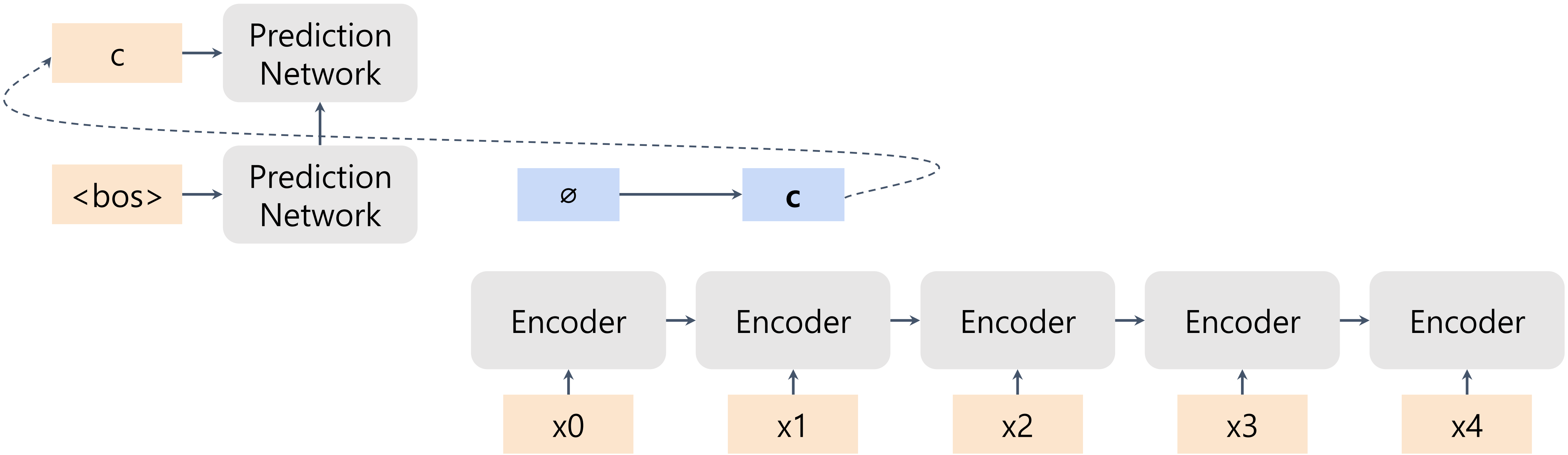
g_0 — decoder state
f_0 — encoder state



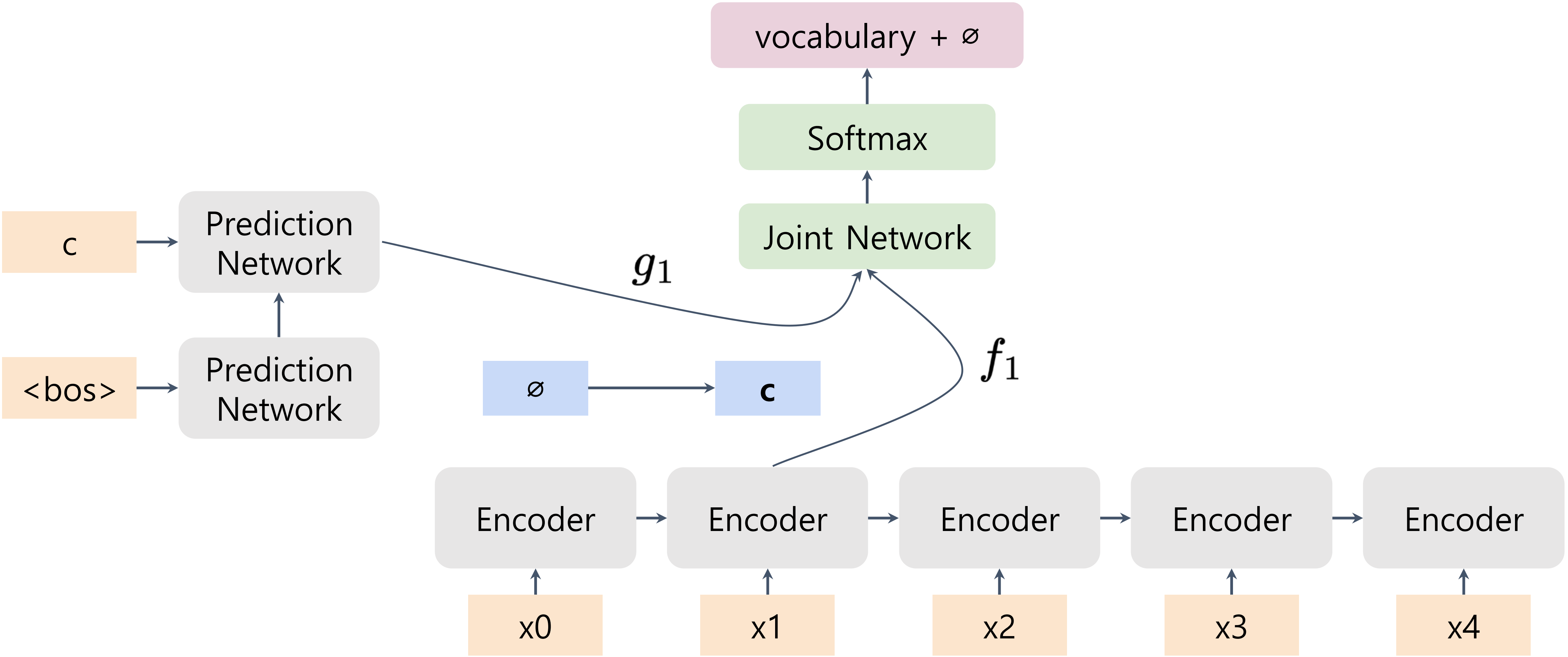
RNN-t Inference



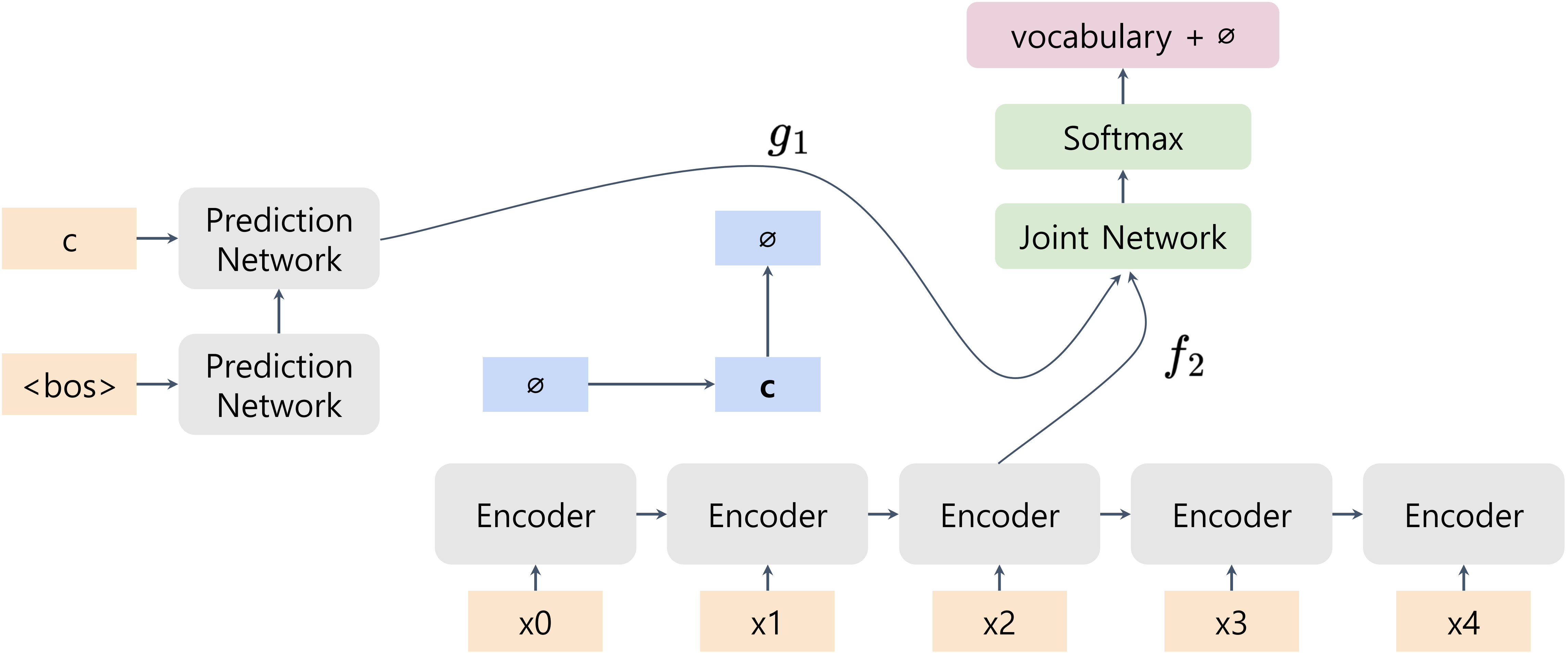
RNN-t Inference



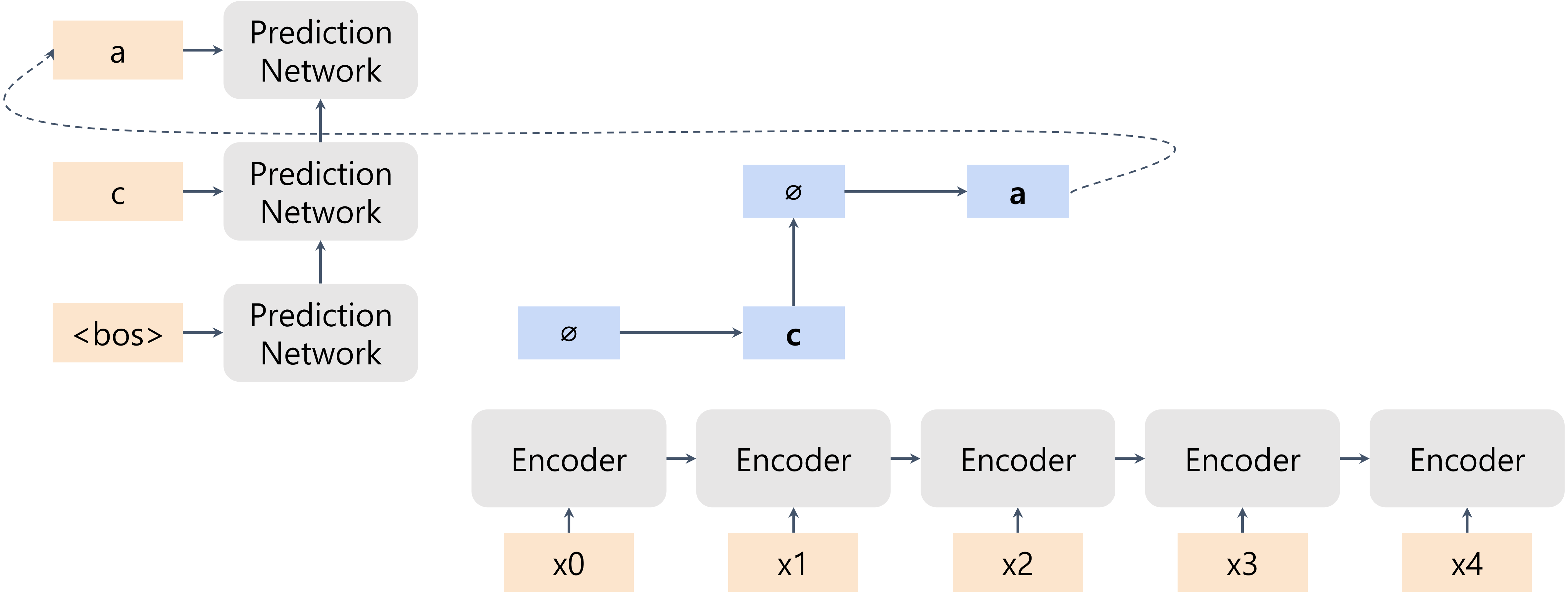
RNN-t Inference



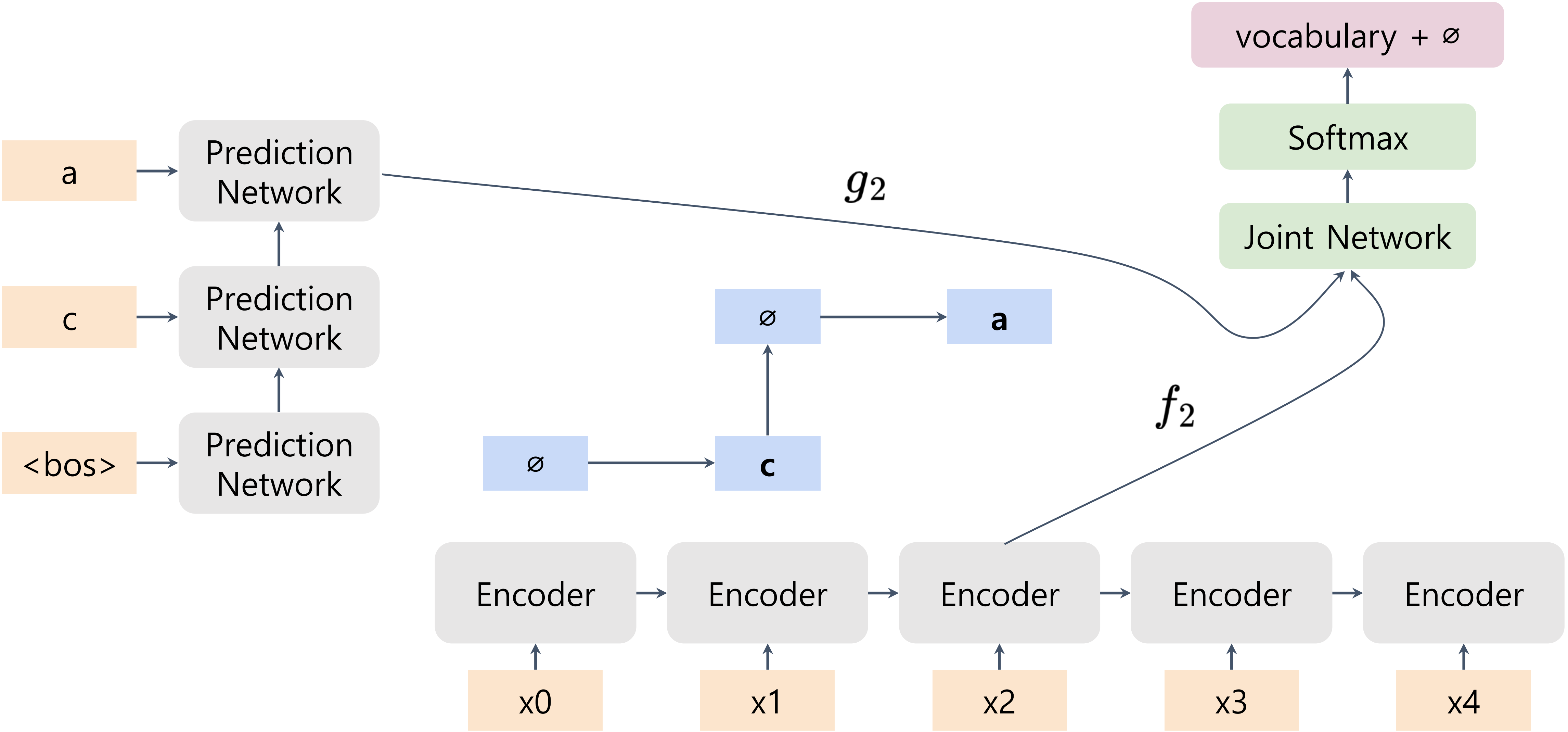
RNN-t Inference



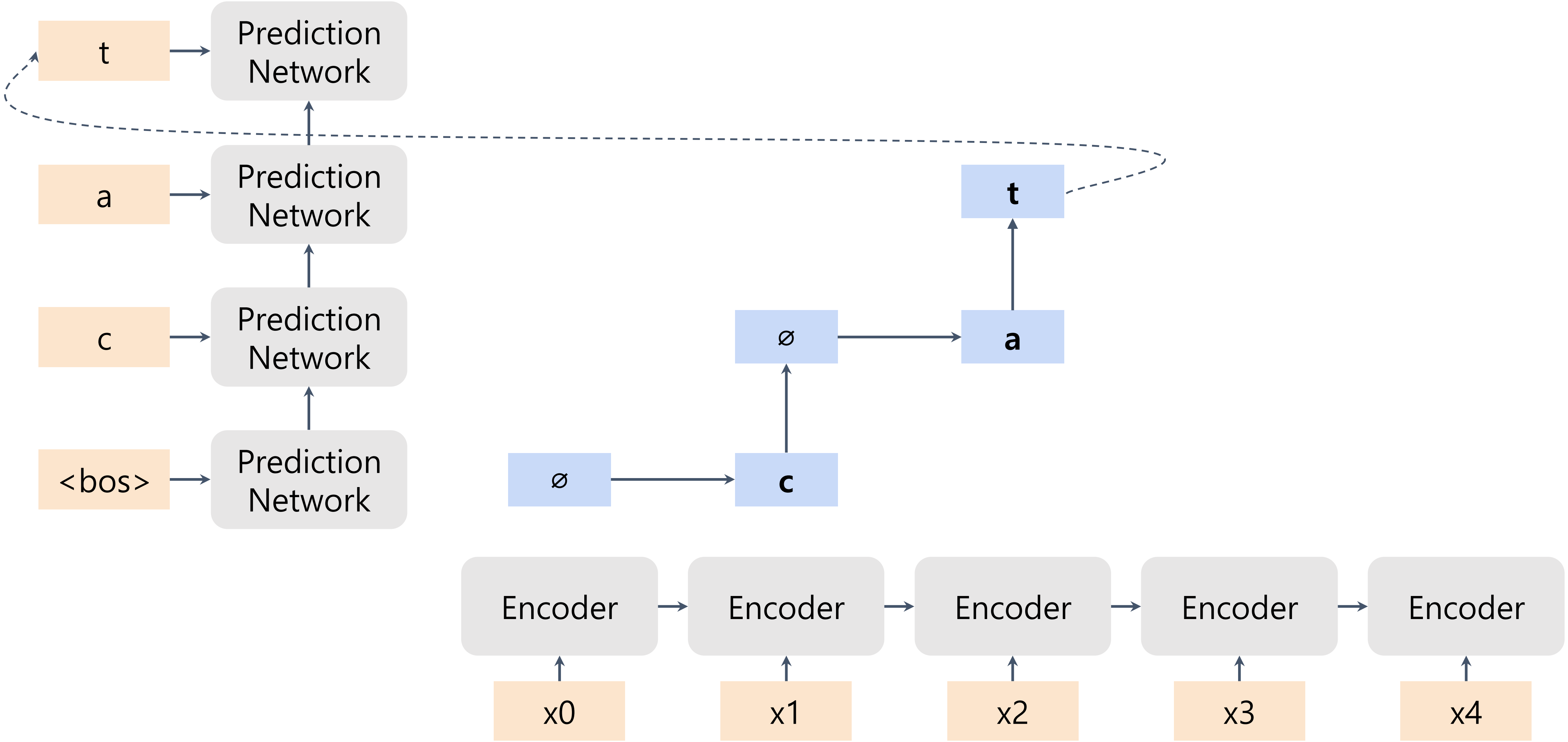
RNN-t Inference



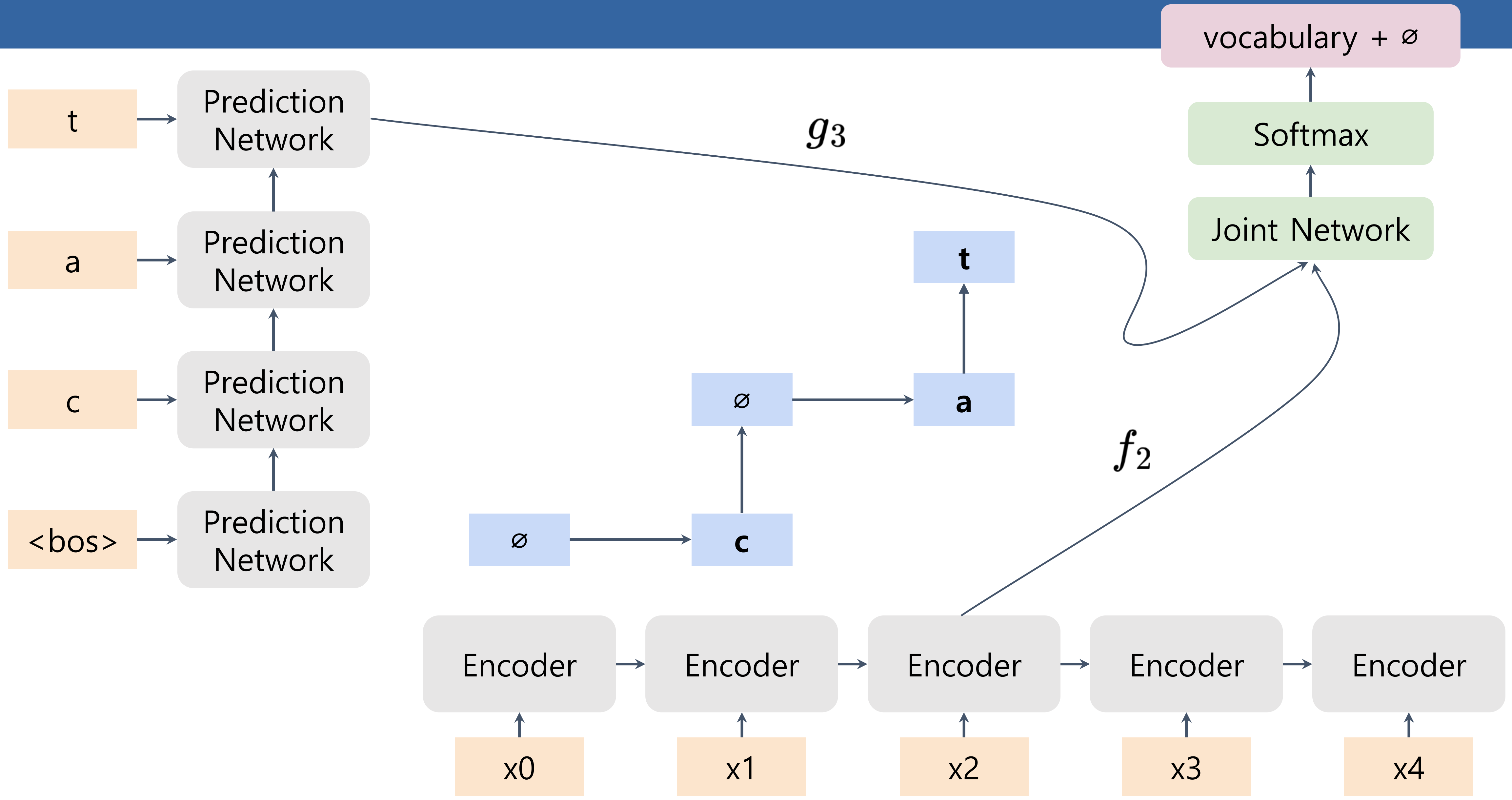
RNN-t Inference



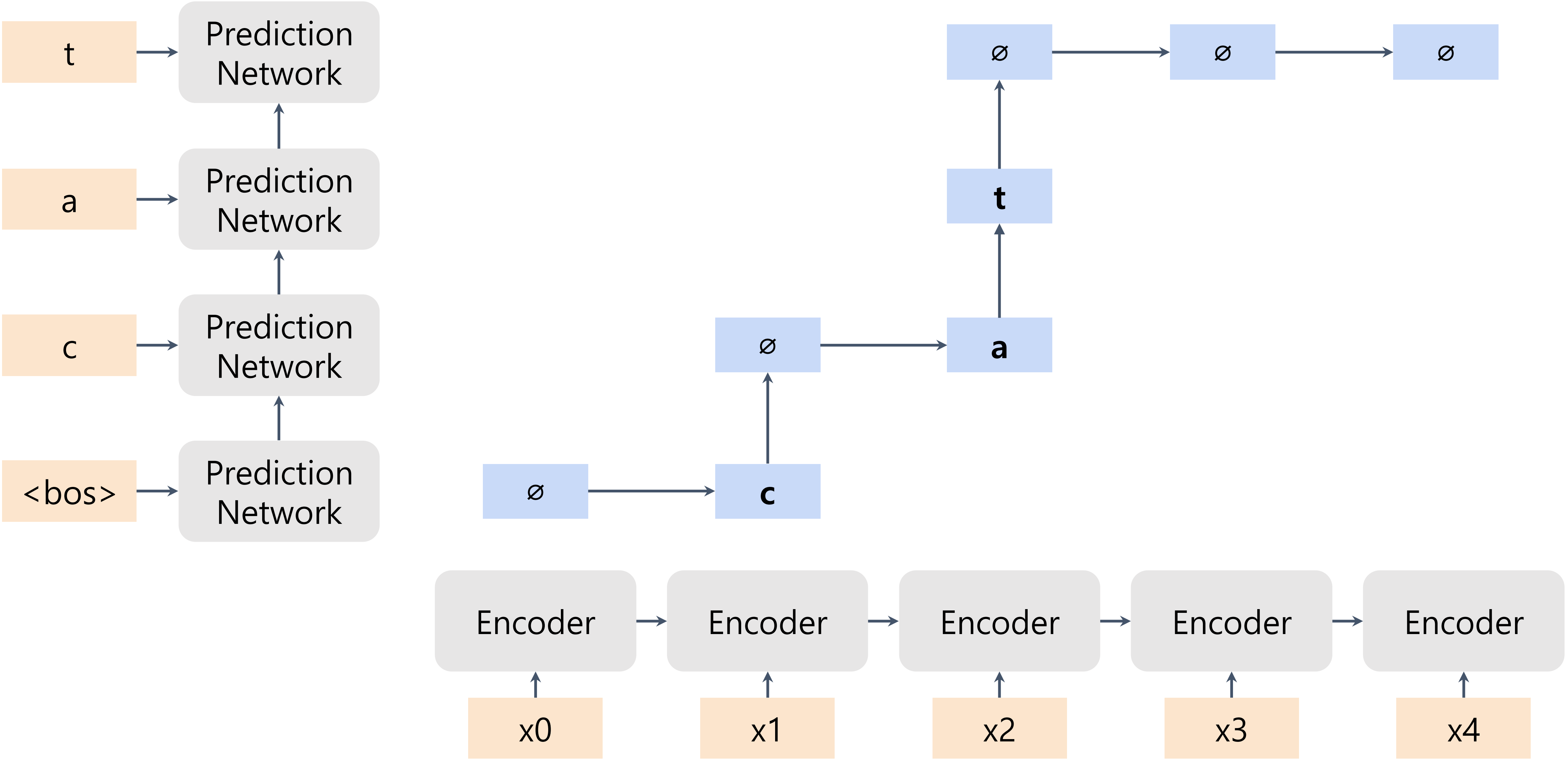
RNN-t Inference



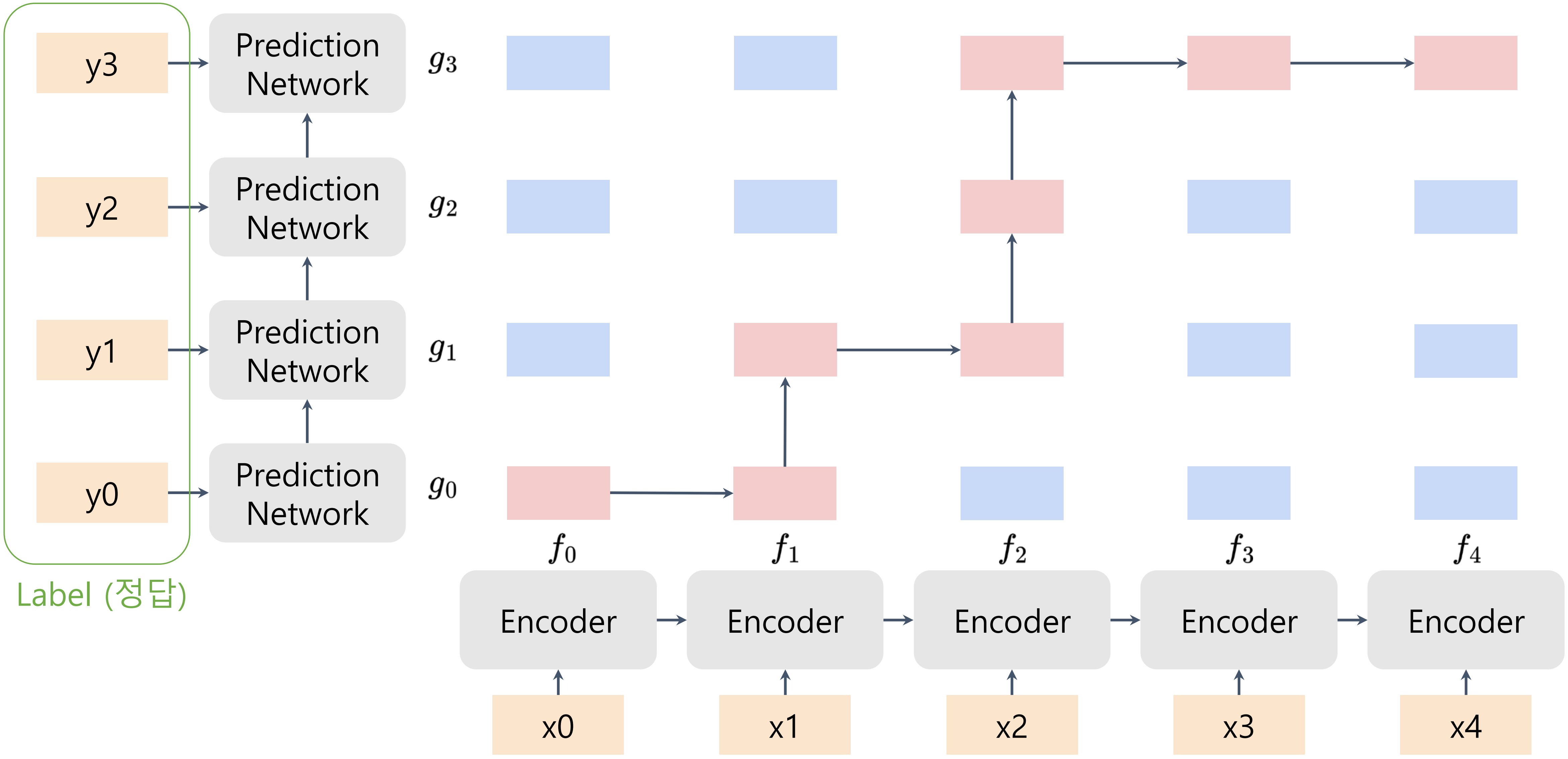
RNN-t Inference



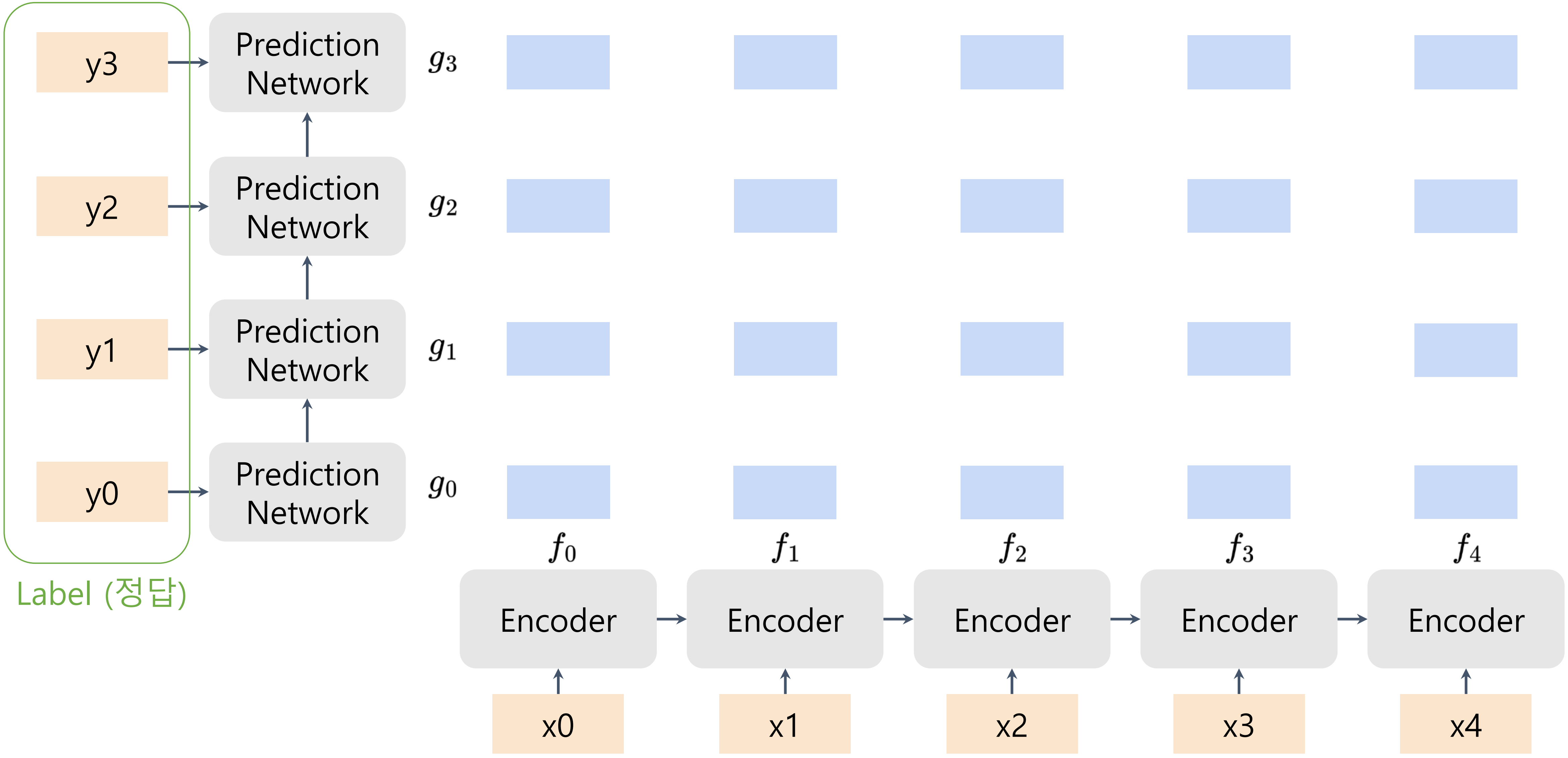
RNN-t Inference



RNN-t train

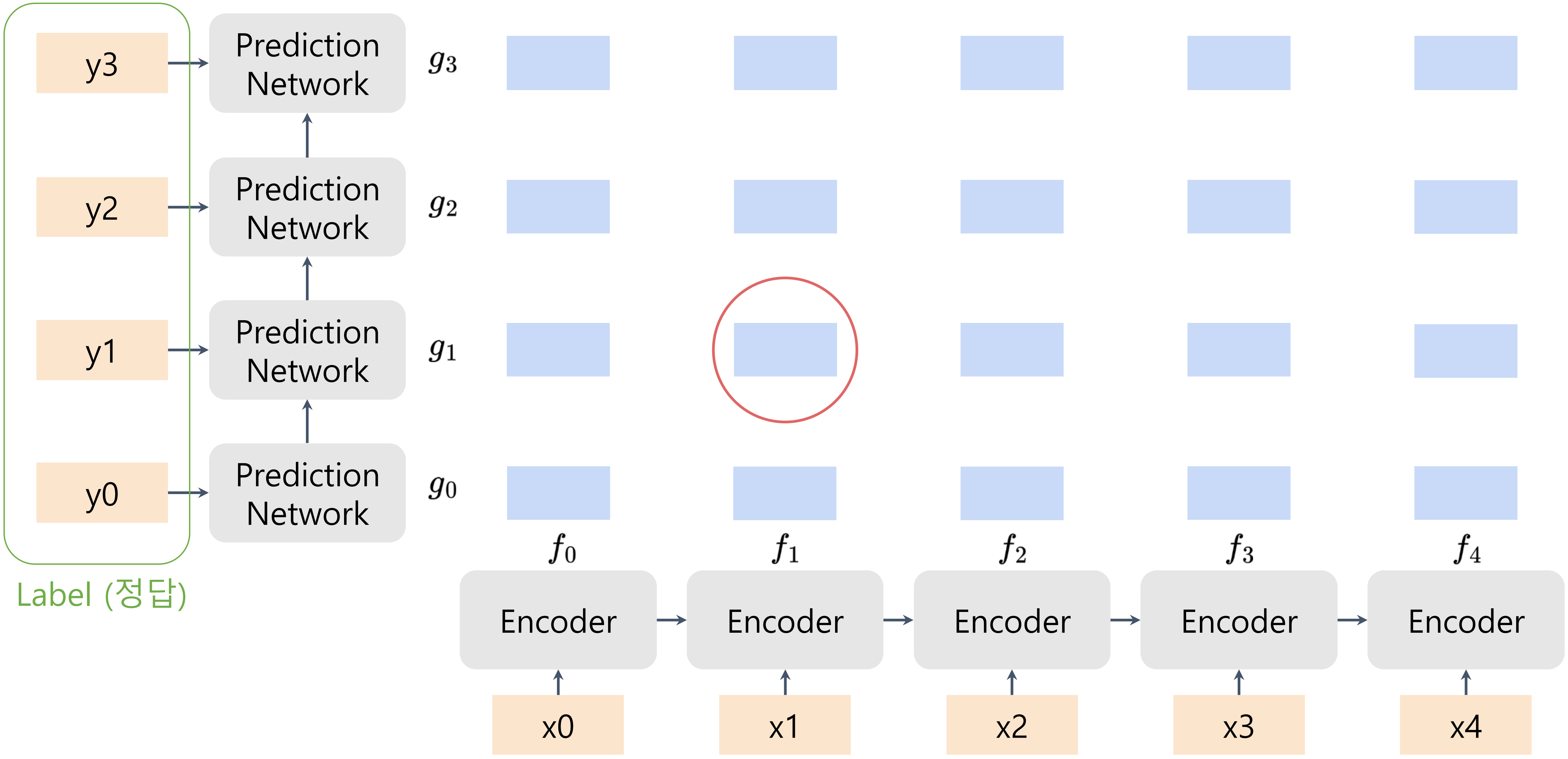

y3
y2
y1
y0
Label (정답)
Prediction Network
Prediction Network
Prediction Network
Prediction Network
g_3
g_2
g_1
g_0
f_0
f_1
f_2
f_3
f_4
Encoder
Encoder
Encoder
Encoder
Encoder
x0
x1
x2
x3
x4

RNN-t train



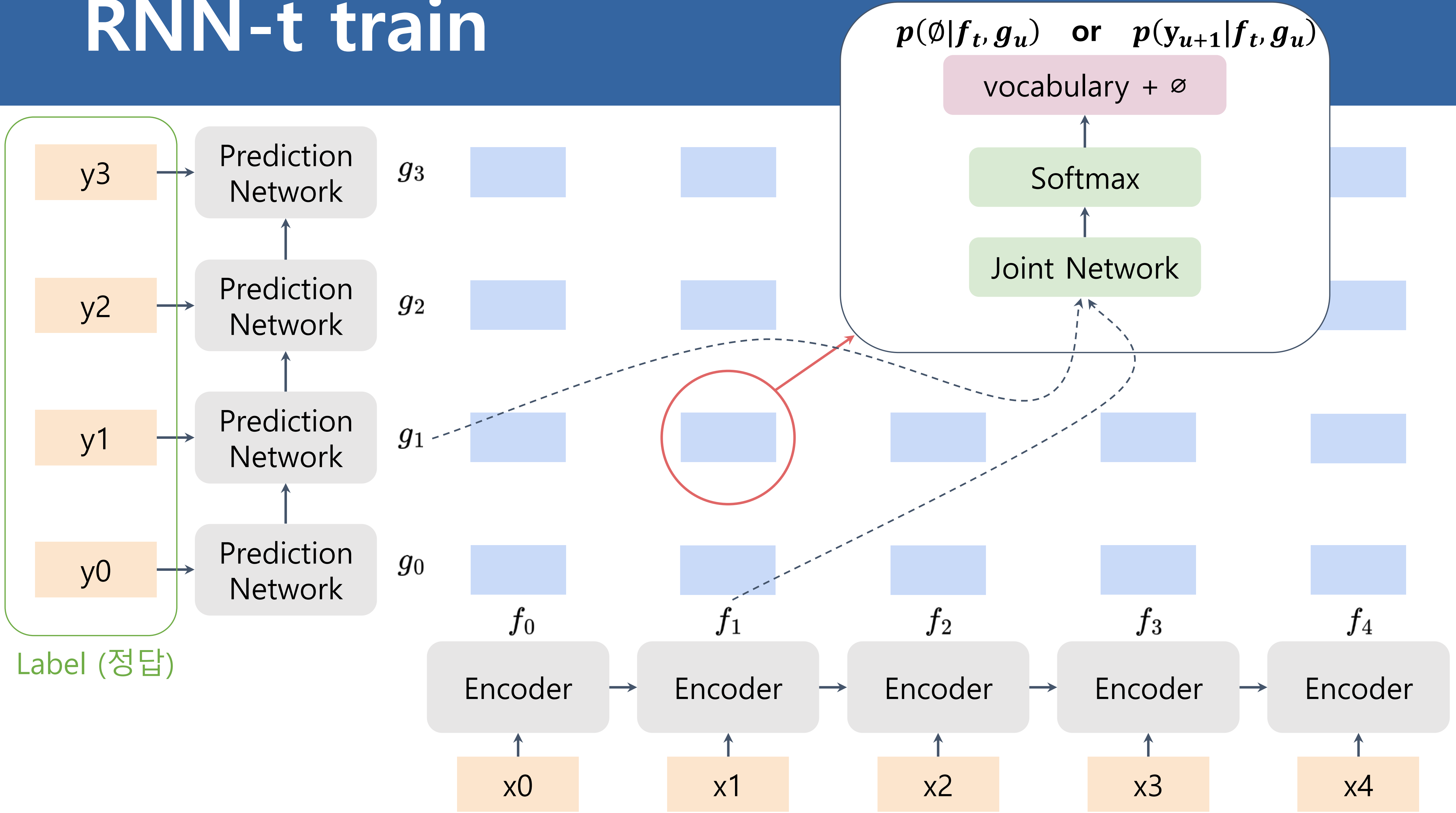
y3
y2
y1
y0
Prediction Network
Prediction Network
Prediction Network
Prediction Network
g_3
g_2
g_1
g_0
f_0
f_1
f_2
f_3
f_4
Encoder
Encoder
Encoder
Encoder
Encoder
x0
x1
x2
x3
x4
Label (정답)

RNN-t train



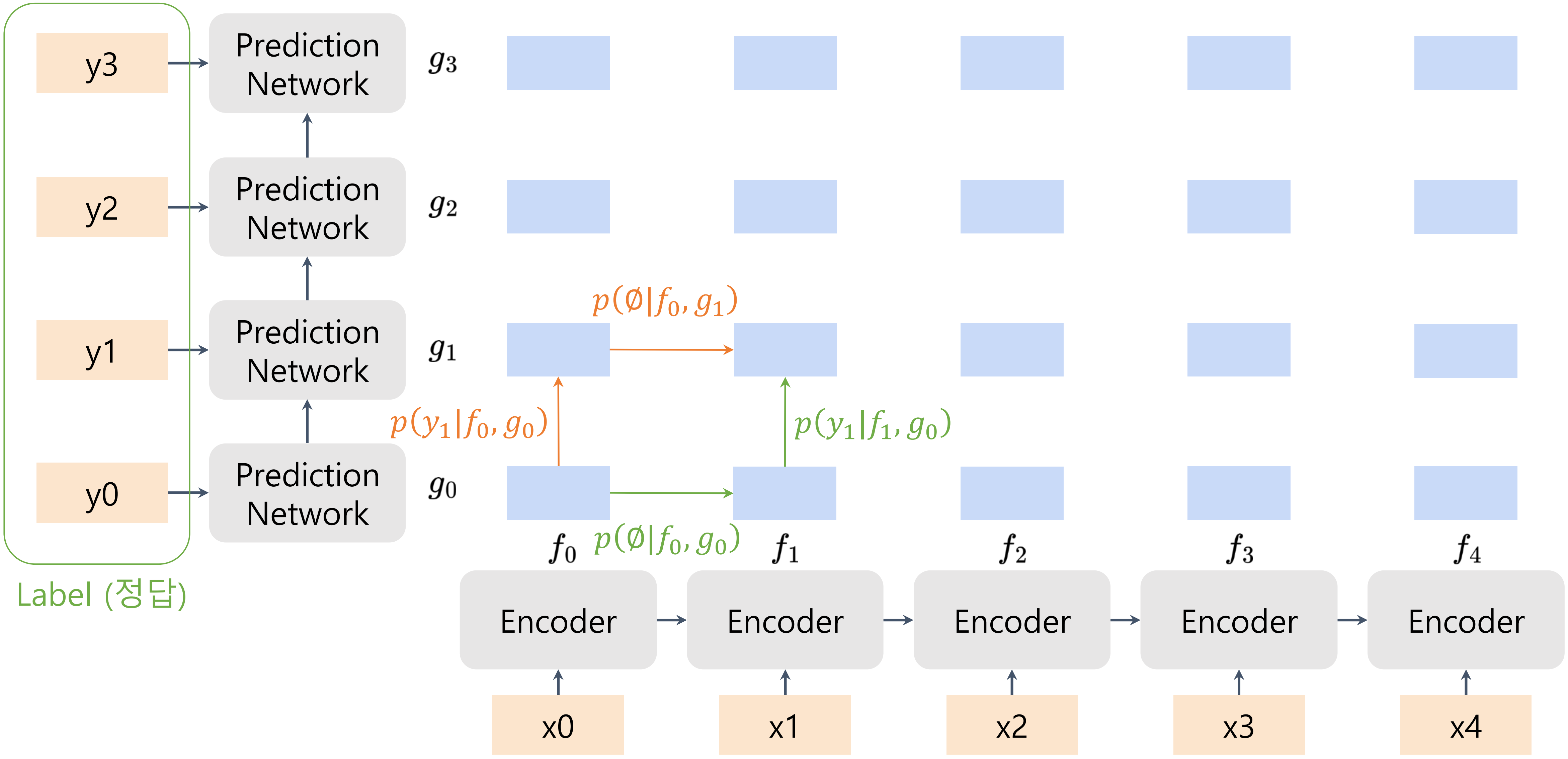
y3
y2
y1
y0
Prediction Network
Prediction Network
Prediction Network
Prediction Network
Label (정답)
g_3
g_2
g_1
g_0
f_0
f_1
f_2
f_3
f_4
Encoder
Encoder
Encoder
Encoder
Encoder
x0
x1
x2
x3
x4

RNN-t train



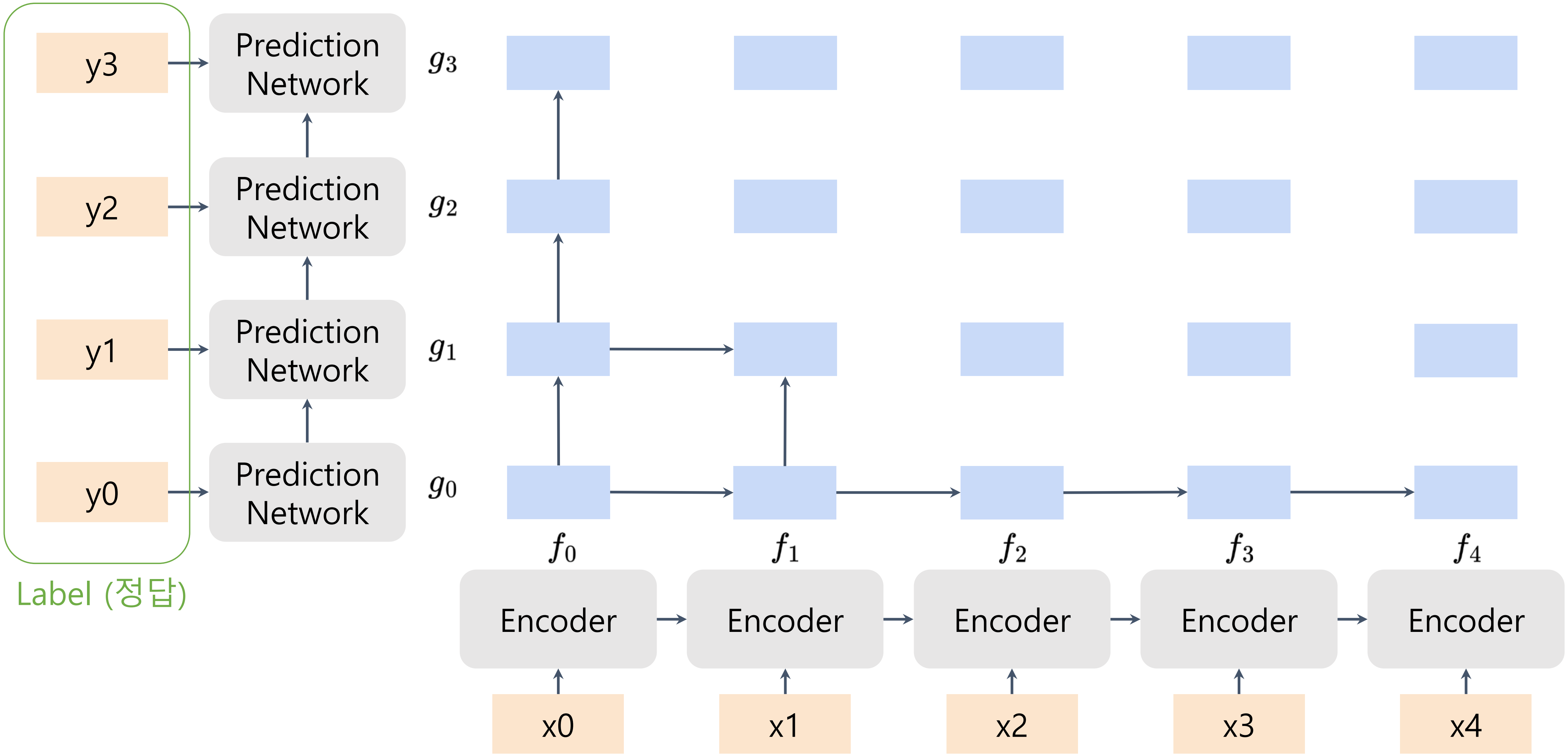
$p(\emptyset|f_t, g_u)$ or $p(y_{u+1}|f_t, g_u)$
vocabulary + ∅
Softmax
Joint Network
y3
y2
y1
y0
Prediction Network
Prediction Network
Prediction Network
Prediction Network
g_3
g_2
g_1
g_0
Label (정답)
f_0
f_1
f_2
f_3
f_4
Encoder
Encoder
Encoder
Encoder
Encoder
x0
x1
x2
x3
x4

RNN-t train



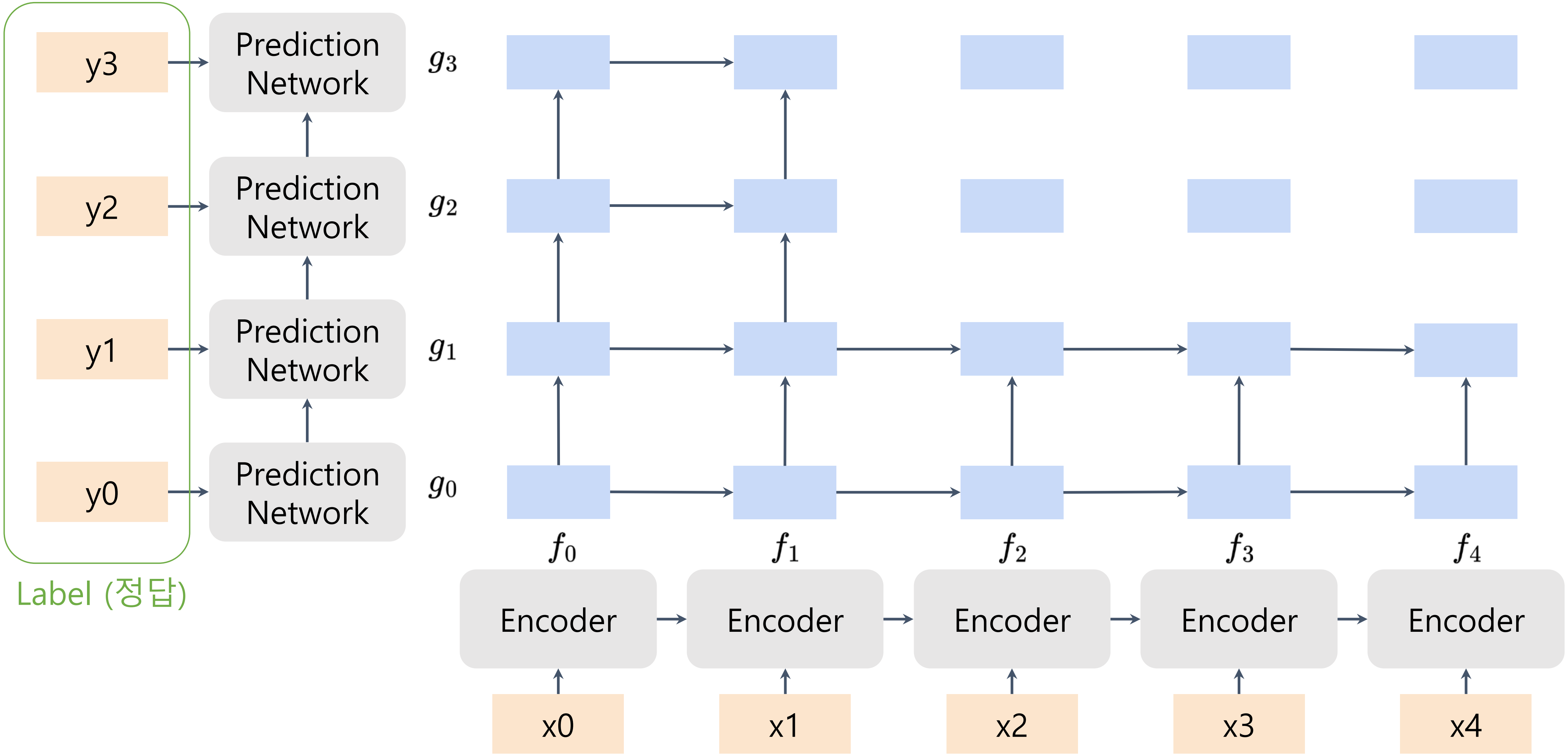
y3
y2
y1
y0
Label (정답)
Prediction Network
Prediction Network
Prediction Network
Prediction Network
g_3
g_2
g_1
g_0
$p(\emptyset|f_0, g_1)$
$p(y_1|f_0, g_0)$
$p(y_1|f_1, g_0)$
$p(\emptyset|f_0, g_0)$
f_0
f_1
f_2
f_3
f_4
Encoder
Encoder
Encoder
Encoder
Encoder
x0
x1
x2
x3
x4

RNN-t train

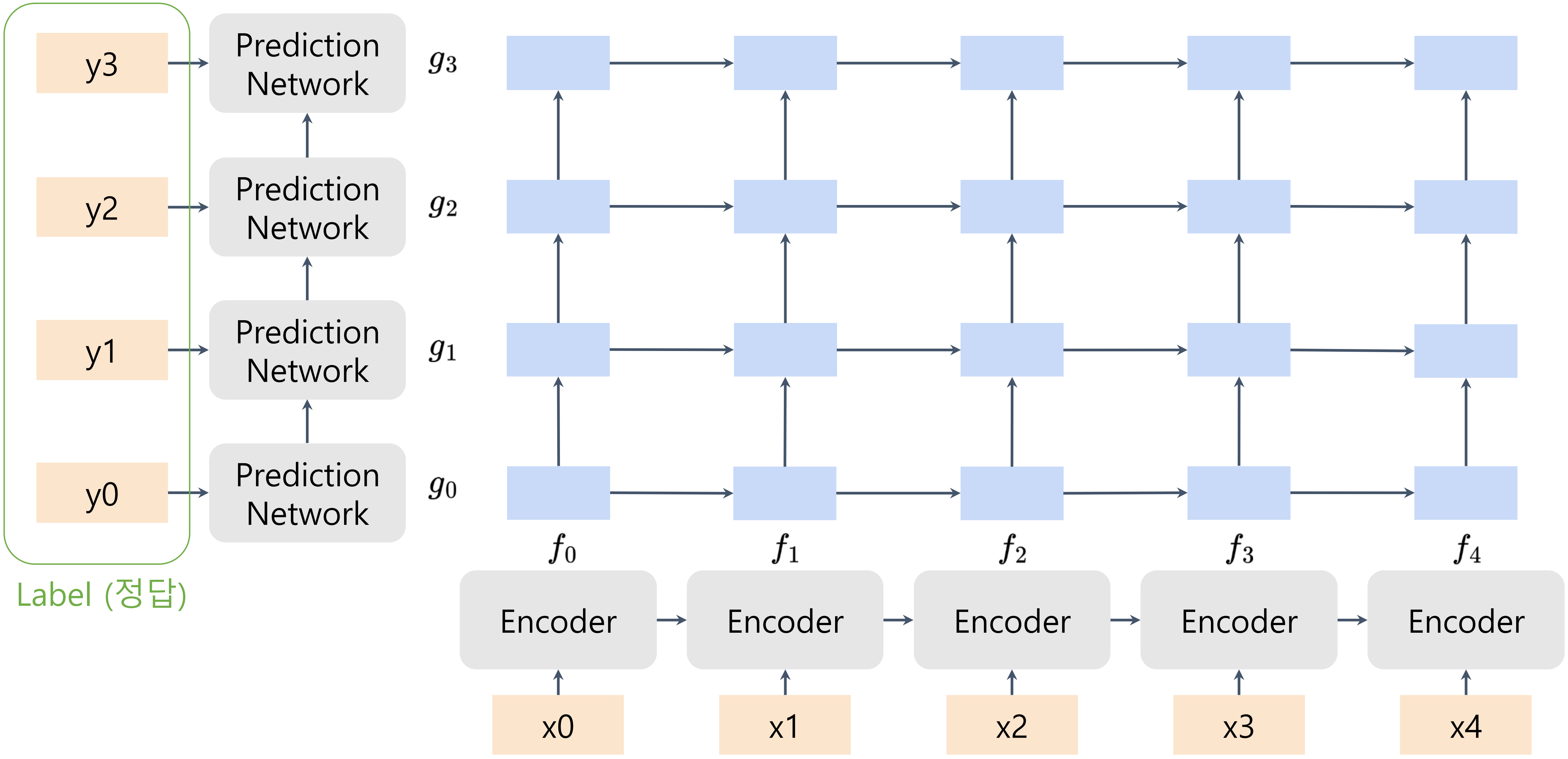


y3
y2
y1
y0
Label (정답)
Prediction Network
Prediction Network
Prediction Network
Prediction Network
g_3
g_2
g_1
g_0
f_0
f_1
f_2
f_3
f_4
Encoder
Encoder
Encoder
Encoder
Encoder
x0
x1
x2
x3
x4

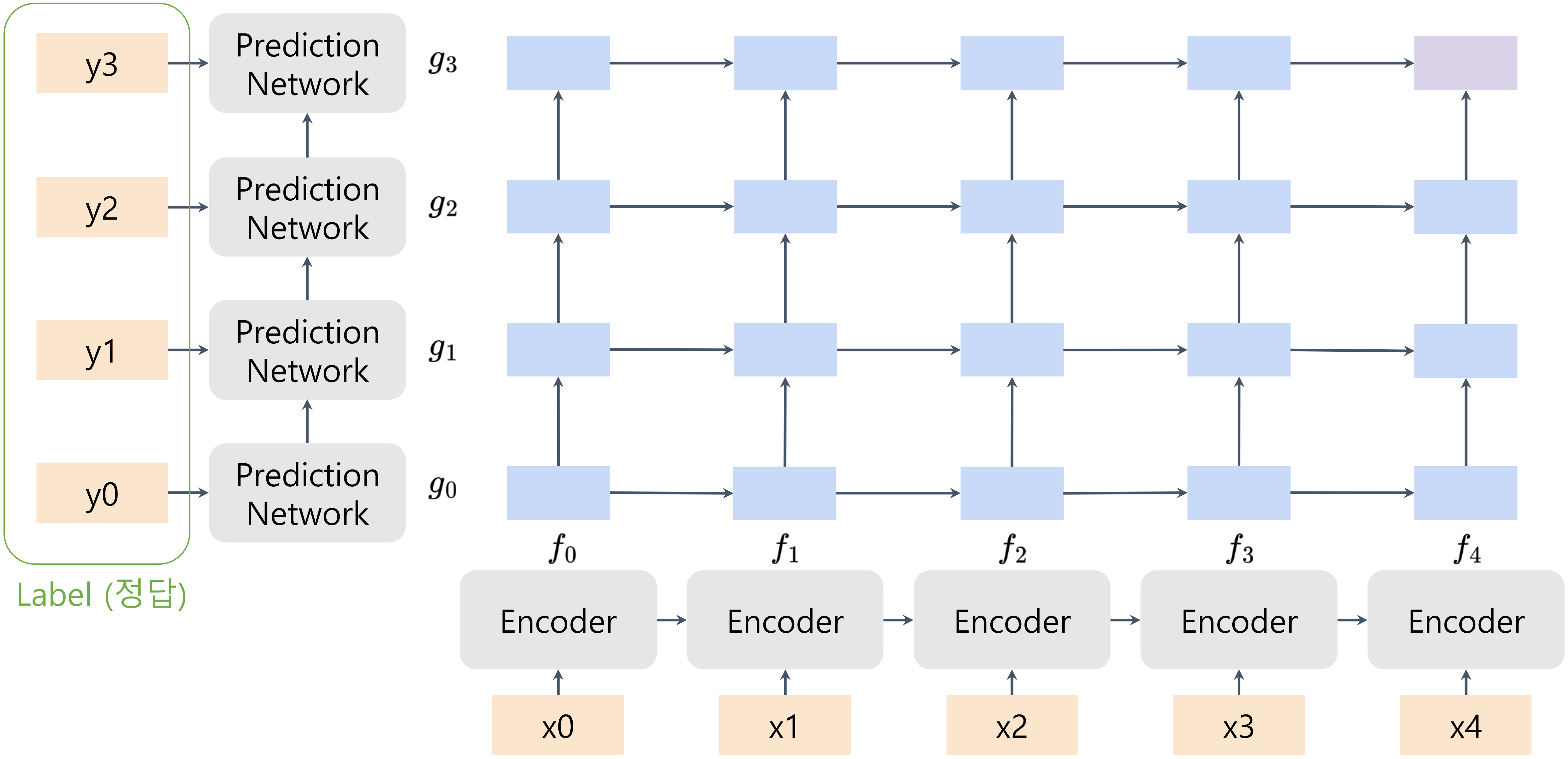
RNN-t train


y3
y2
y1
y0
Prediction Network
Prediction Network
Prediction Network
Prediction Network
g_3
g_2
g_1
g_0
f_0
f_1
f_2
f_3
f_4
Encoder
Encoder
Encoder
Encoder
Encoder
x0
x1
x2
x3
x4
Label (정답)

RNN-t train


y3
y2
y1
y0
Label (정답)
Prediction Network
Prediction Network
Prediction Network
Prediction Network
g_3
g_2
g_1
g_0
f_0
f_1
f_2
f_3
f_4
Encoder
Encoder
Encoder
Encoder
Encoder
x0
x1
x2
x3
x4

RNN-t train


y3
y2
y1
y0
Label (정답)
Prediction Network
Prediction Network
Prediction Network
Prediction Network
g_3
g_2
g_1
g_0
f_0
f_1
f_2
f_3
f_4
Encoder
Encoder
Encoder
Encoder
Encoder
x0
x1
x2
x3
x4

RNN-t properties

- streamable and context dependent
- produce alignment (weak)
- it is difficult to compute efficiently
  - calculation of the joint module is hard
    BS = 32 ; T (after 2x stride) = 800, U (with character encoding) = 400-450 tokens, V = 28. Let the hidden dimension of the Joint model be 640.
    - Hidden = 32×800×450×640×4 = 29.49 Gb (fp32)
      Joint = 32×800×450×28×4 = 1.290 Gb (fp32)
    - This is just for the forward pass. We need to double this memory to store gradients.
  - BS=32 ; T (after 8x stride) = 200, U (with sub-word encoding) = 100-180 tokens, Vocabulary size V = 1024.
    - Hidden = 32×200×150×640×4 = 2.45 Gb (fp32)
      Joint = 32×200×150×1024×4 = 3.93 Gb (fp32)

RNN-t Quality

Method	Year	WER (test-clean)	WER (test-other)
Human	~ 200 000 b.c.	5.83	12.69
Deep Speech 2	2015	5.15	12.73
LAS	2015	----	----
LAS without SpecAugment*	2019	3.2	9.8
RNN-t	2020	**2.1**	**4.3**

*https://arxiv.org/abs/1904.08779

CTC vs. LAS vs. RNN-t

Method	CTC	LAS	RNN-t
Streaming	yes	no	yes
Context	no	yes	yes
Argmax Compl exity	O(enc + dec)	O(enc + T * dec)	O(enc + T * dec)
Beam Search C omplexity	O(enc + dec + T * bs)	O(enc + T * bs * dec)	O(enc + T * bs * dec)

DL-based ASR

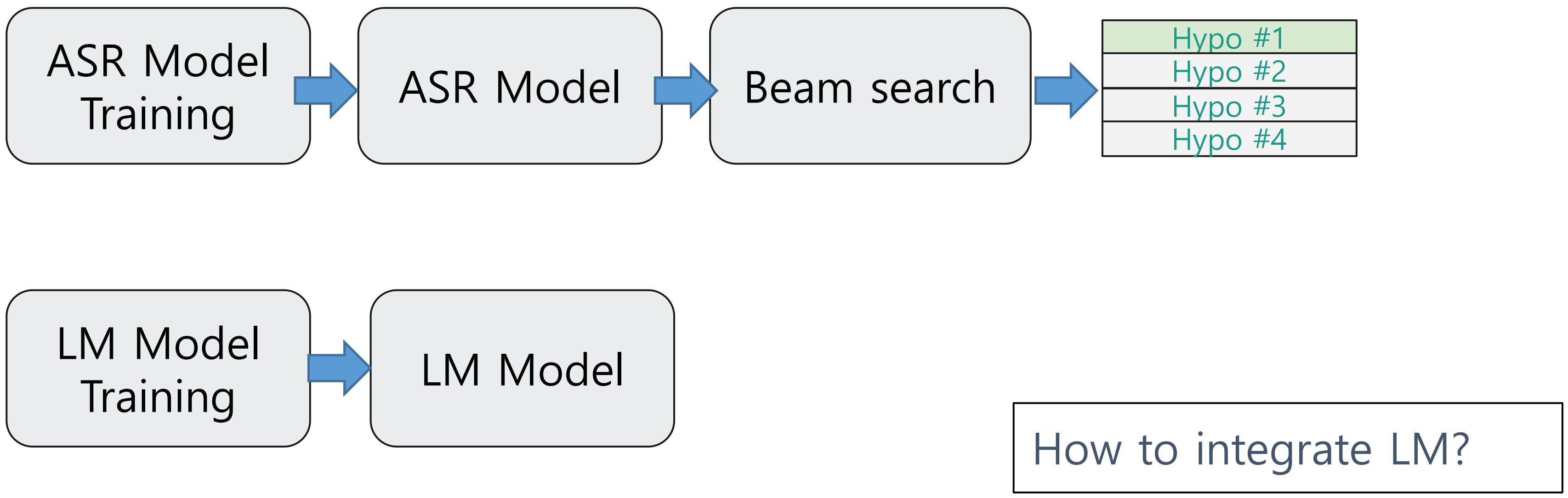
- Language models (LM) - Refresher
  - motivation:
    - spelling of a word heavily depends on its context
    - LMs can add context dependency in CTC beam search
    - external language models typically saw more data (more than decoders in LAS, RNN-t)
  - examples:
    - simple: n-gramms, Kneser–Ney smoothing and others
    - complex: neural networks
      e.g: Bert, GPT, LLaMA

hypo 1: let's go ***two*** a movie (am score: 0.21)
hypo 2: let's go ***to*** a movie (am score: 0.19)
hypo 3: let's go ***too*** a movie (am score: 0.13)

P(let's go ***two*** a movie) = 0.01 (lm score)
P(let's go ***to*** a movie) = 0.6 (lm score)

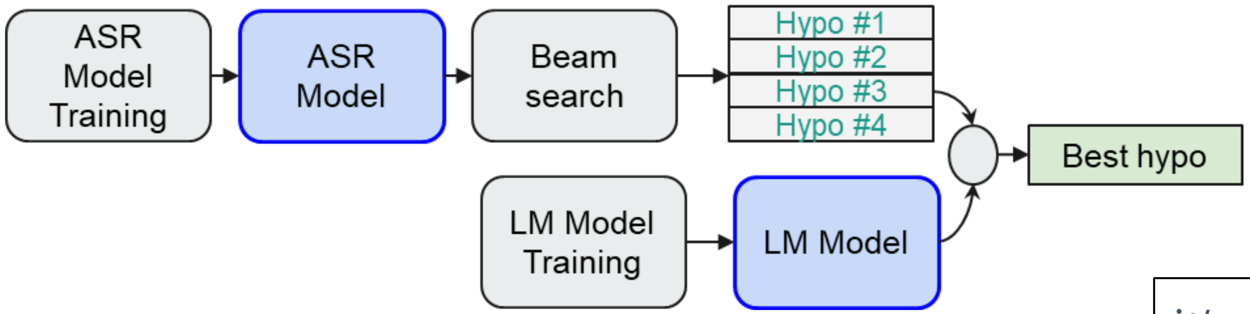
DL-based ASR

- Language models (LM) - Refresher

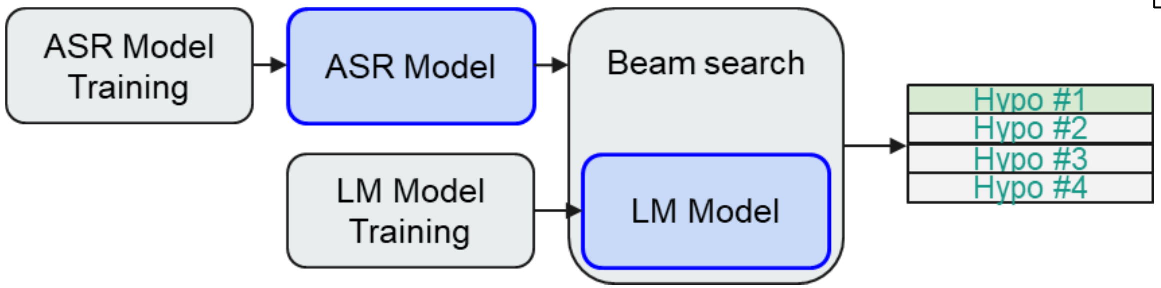


DL-based ASR

- Language models (LM)
  - Second pass rescoring:



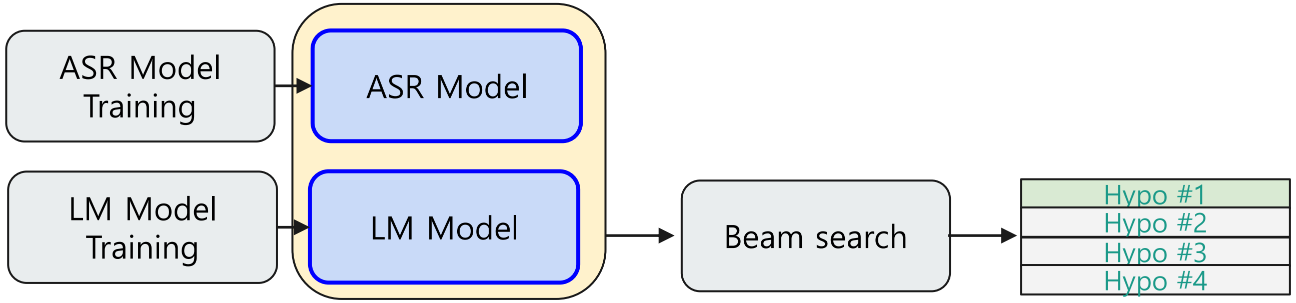
  - Shallow fusion:



it's better to use:
- large model (ASR) for rescoring
- light model (ASR) for shallow fusion

DL-based ASR

- Language models (LM) with LAS and RNN-t
  - Deep Fusion:



  - Cold Fusion: