Encoder-decoder models 1: the RNN transducer

Peter Bell

Automatic Speech Recognition – ASR Lecture 13
2 March 2026

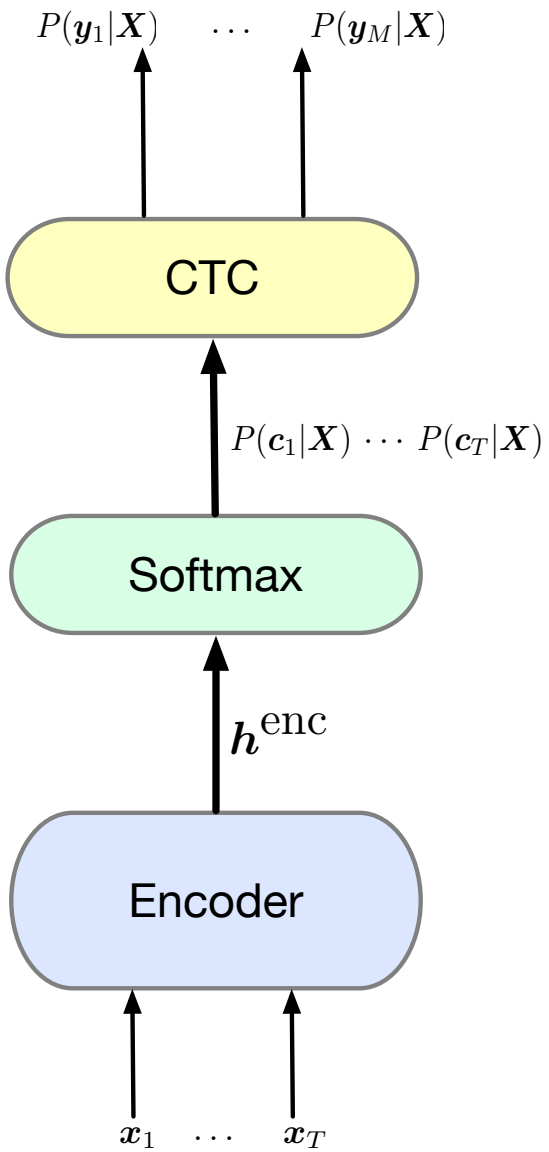
Recap – CTC

- Adds a blank (ϵ) symbol to the output labels
- A deep LSTM (for example) maps input sequence X (length T) to a label sequence C (length T)
- Use CTC compression rule (merge adjacent repeated symbols, then remove blanks) to produce subword sequence Y (length $M \leq T$)
- CTC loss function computes the probability $P(Y|X)$ by summing over all possible valid alignments $P(C|X)$

CTC Model

View CTC as having three components:

- **Encoder**: Deep (bidirectional) LSTM recurrent network which maps acoustic features $X = x_1, \ldots, x_T$ to a sequence of hidden vectors $h^{\text{enc}} = h_1^{\text{enc}}, \ldots, h_T^{\text{enc}}$.
- **Softmax**: Computes the label probabilities $P(c_1|X), \ldots, P(c_T|X)$
- **CTC**: Computes the subword sequence $P(y_1|X), \ldots, P(y_M|X)$



Limitations of CTC

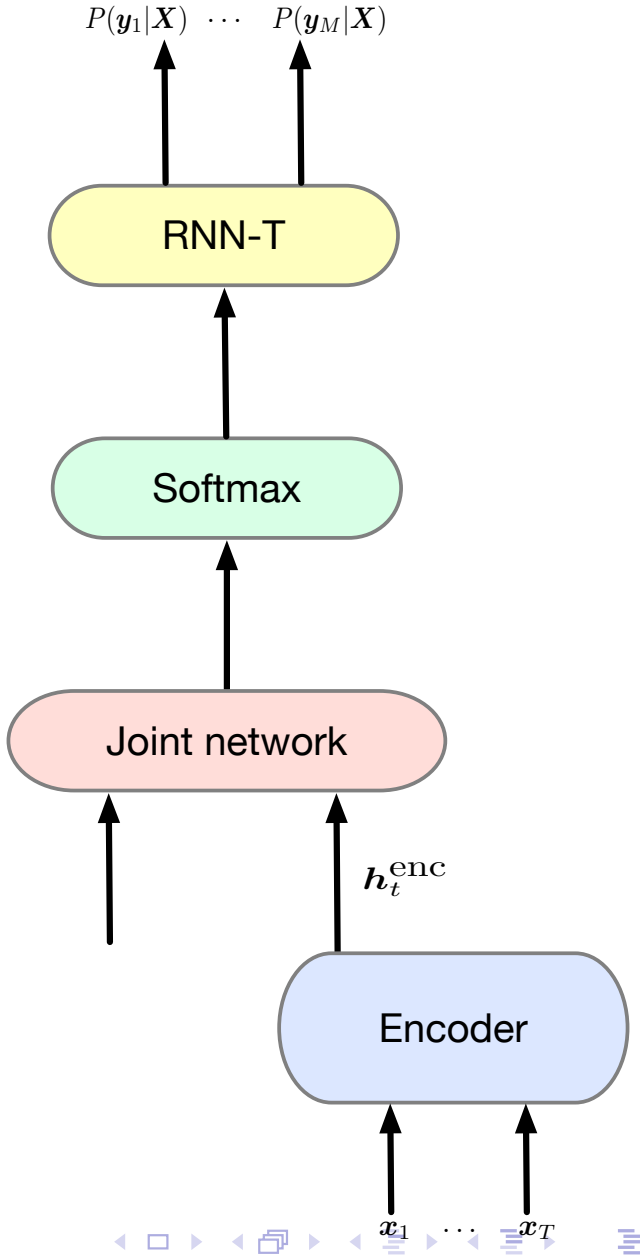
- CTC – pros
  - Can train end-to-end without requiring framewise alignments
  - Sums over all possible alignments (using forward-backward)
  - Preserves monotonic relationship between acoustic frames and output labels

Limitations of CTC

- CTC – pros
  - Can train end-to-end without requiring framewise alignments
  - Sums over all possible alignments (using forward-backward)
  - Preserves monotonic relationship between acoustic frames and output labels
- CTC – cons
  - Assumes output predictions at different times are conditionally independent, given X.
  - In practice, this means that the model learns only weak language and pronunciation models
  - Incorporation of external language models is typically ad-hoc
  - End-to-end training of CTC models updates the "acoustic model" parameters using a sequence level criterion, but does not update the pronunciations or language models

RNN Transducer Model

- **Encoder:** Acoustic model network mapping acoustic features $X = x_1, \ldots, x_T$ to hidden vectors $h^{\text{enc}} = h_1^{\text{enc}}, \ldots, h_T^{\text{enc}}$.
- **Prediction network**: Recurrent network which takes the previous output subword label y_{u-1} as input and predicts the next subword label p_u – acts as a language model (over subwords)
- **Joint network**: Computes a joint hidden vector $g_{t,u}$ by a applying a shallow feed-forward net to h^{enc} and p_u
- Followed by **softmax** and a **CTC-like** compression component

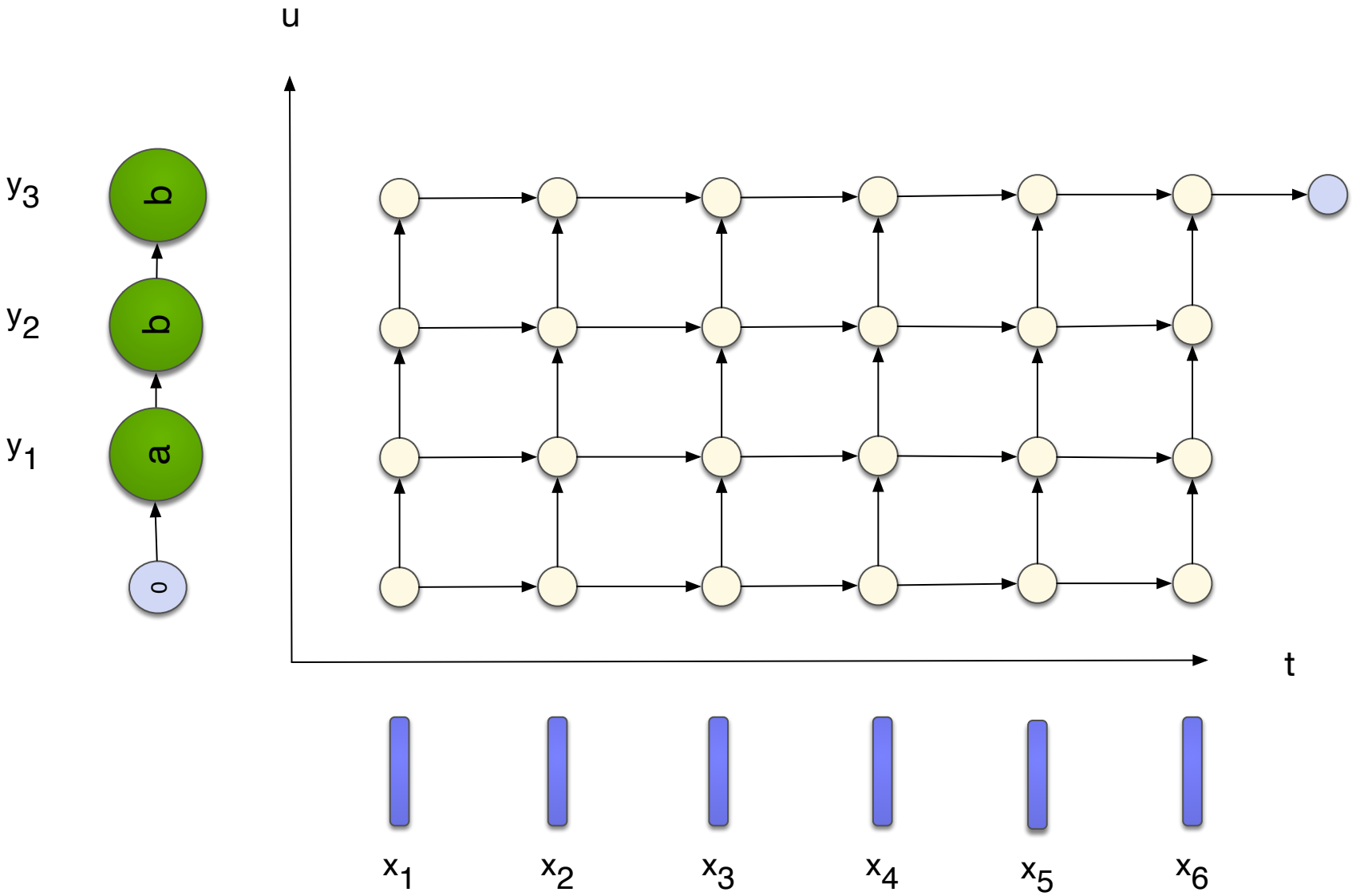


RNN-T output compression

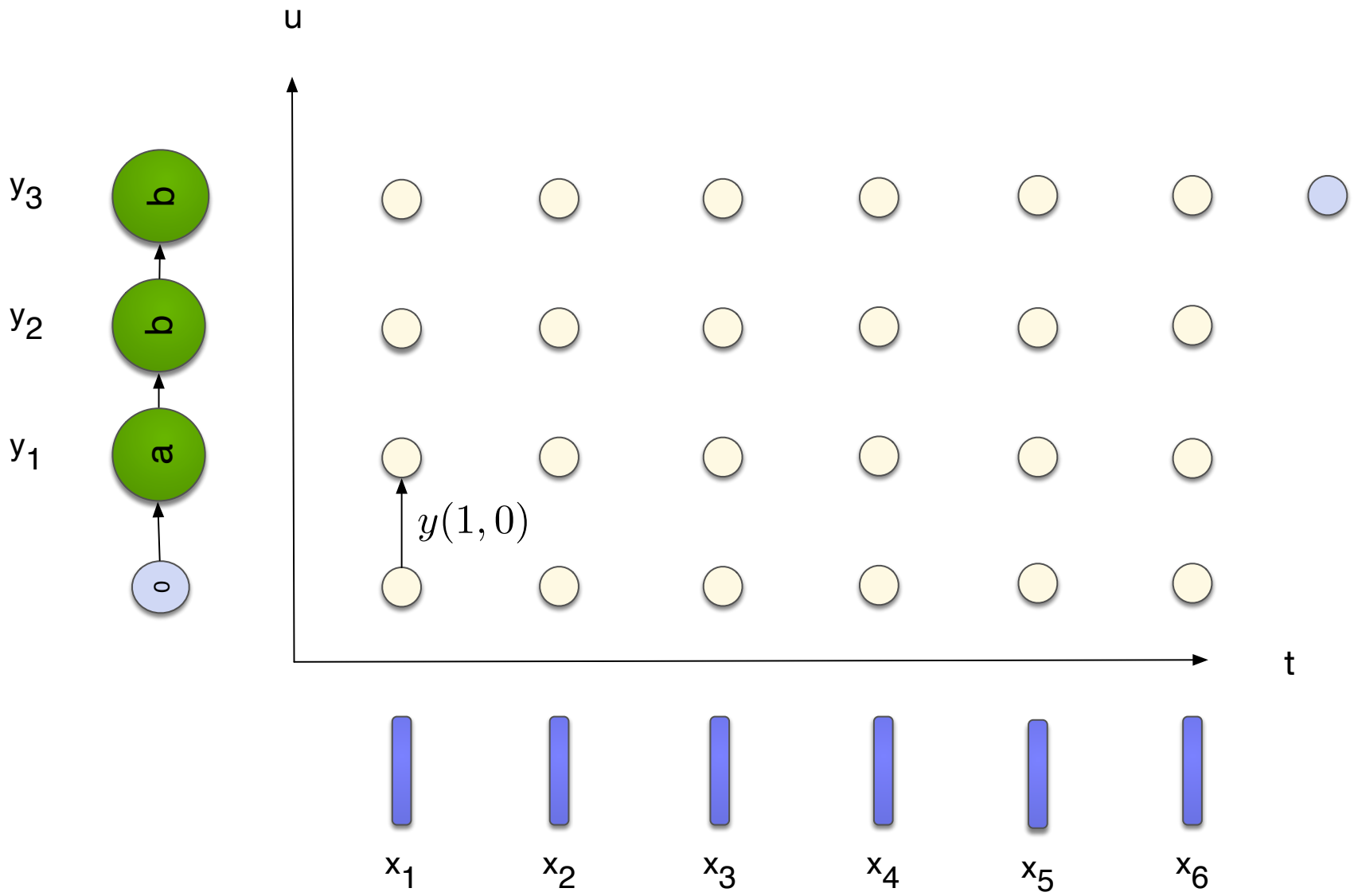
- As in CTC, we extend the output label space by adding a blank label, ϵ.
- The location of the blank symbols determines the alignment between X and Y.
- Each blank symbol signals a shift forward by one time step
- Blank symbols can appear in any position (there is always a blank symbol in the final position). The total number of blank symbols must be equal to the number of frames.
- Suppose we have $(x_1, \ldots, x_6)$ and output (y_1, y_2, y_3). Valid alignments are:

$$(\epsilon, \epsilon, \epsilon, \epsilon, \epsilon, y_1, y_2, y_3, \epsilon)$$
$$(y_1, \epsilon, \epsilon, \epsilon, y_2, \epsilon, \epsilon, y_3, \epsilon)$$
$$(\epsilon, y_1, \epsilon, \epsilon, \epsilon, \epsilon, y_2, y_3, \epsilon)$$
$$(\epsilon, \epsilon, y_1, \epsilon, y_2, \epsilon, \epsilon, y_3, \epsilon)$$
$$\ldots$$

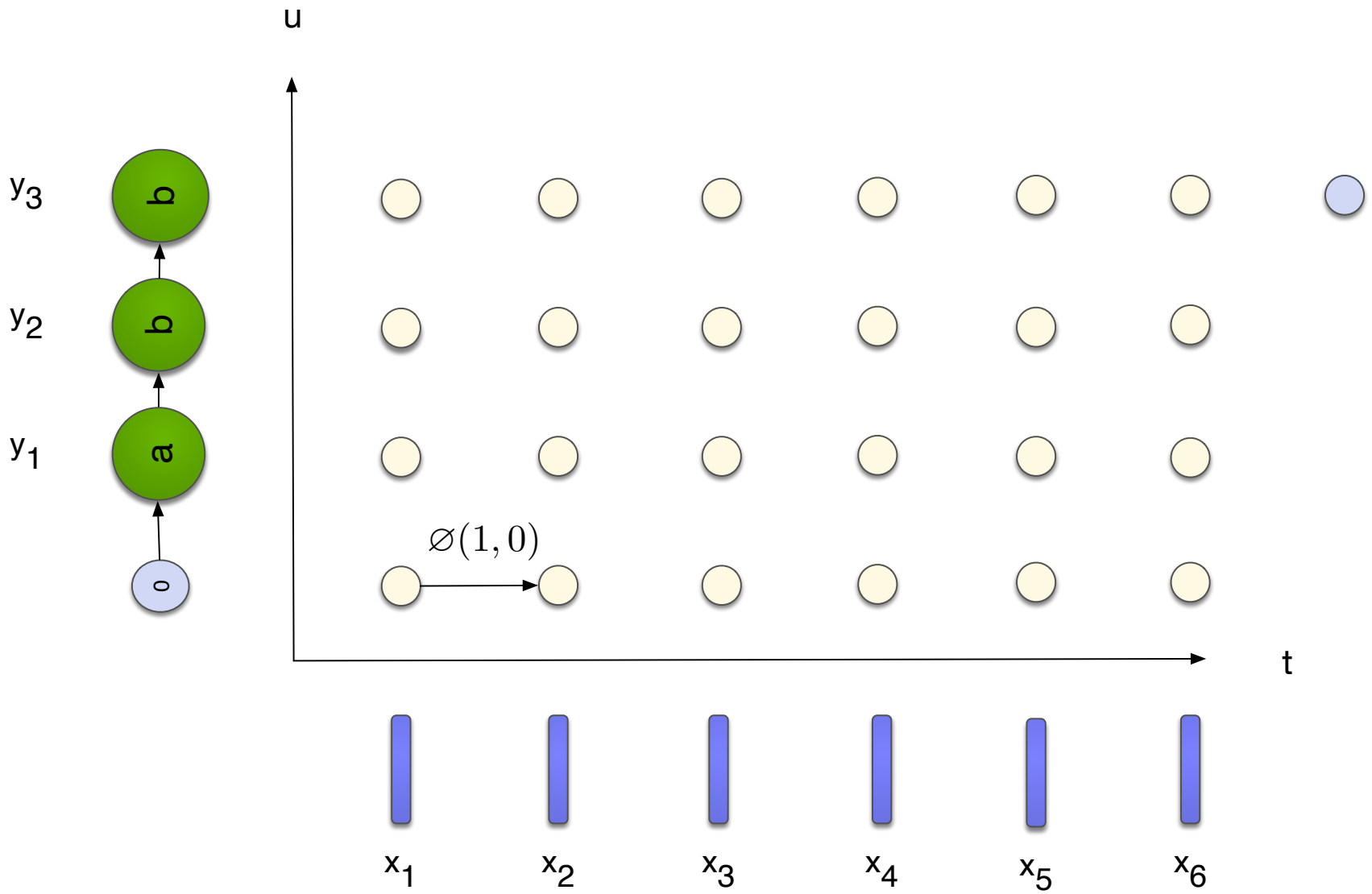
RNN-T: trellis



RNN-T: trellis



RNN-T: trellis



RNN-T prediction

- If C is the expanded output sequence, containing T blank symbols, then

$$P(Y|X) = \sum_{C \in A(Y)} P(C|X)$$

where $A(Y)$ is the set of sequences C that can be mapped to Y using the RNN-T compression rule

- Probability of label c depends on the position in both the input sequence, t, and output sequence u.

$$g_{t,u}(k) = \exp(h_t^{\text{enc}}(k) + p_u(k))$$

$$P(c = k|t, u) = \frac{g_{t,u}(k)}{\sum_{k'} g_{t,u}(k')}$$

RNN-T: forward recursion

Consider a label sequence

$$Y = (y_1, y_2, \ldots, y_U)$$

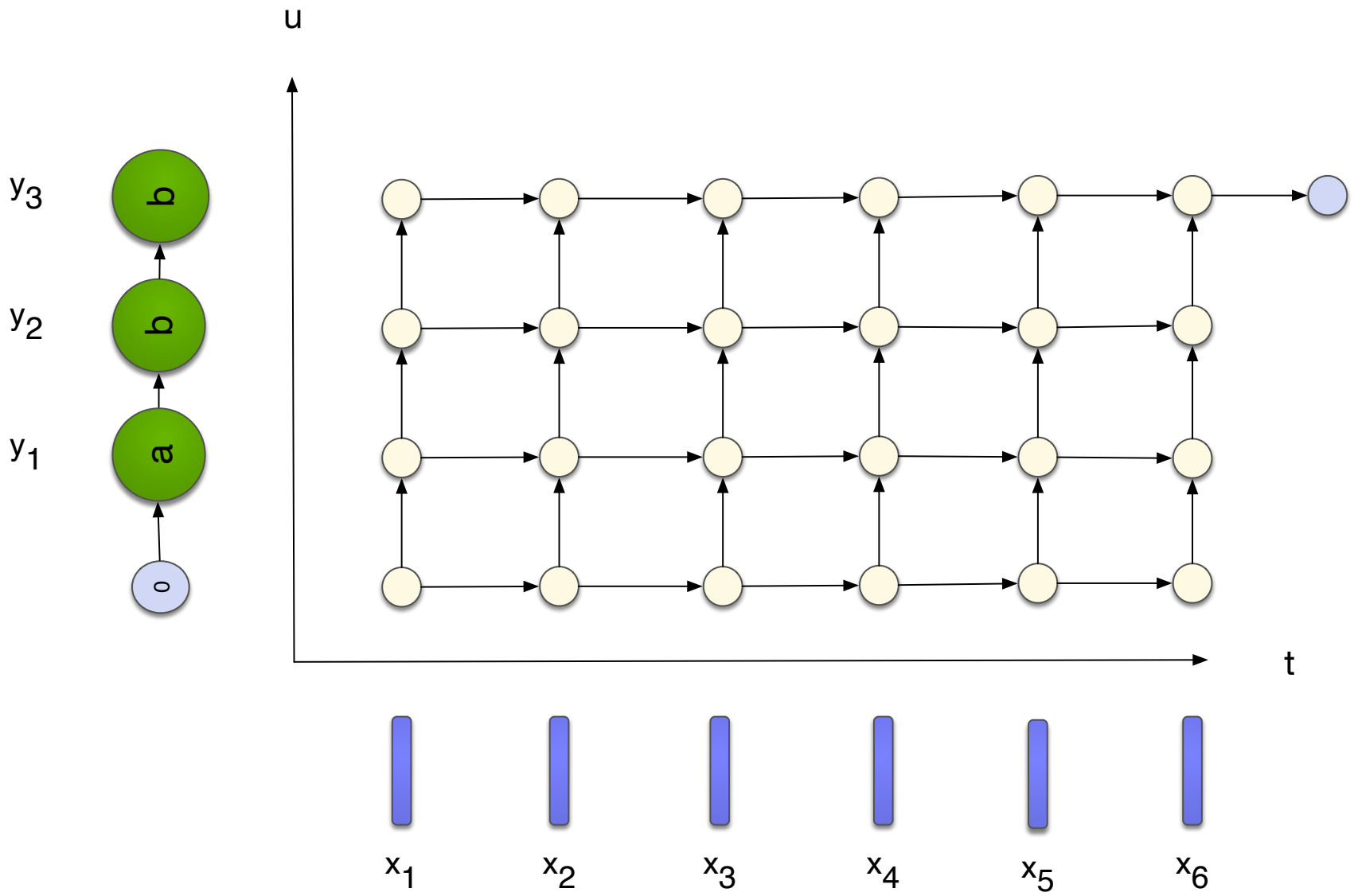
The forward probability is:

$$\alpha_u(t) = P(y_1, \ldots y_u | h_1^{\text{enc}}, \ldots, h_t^{\text{enc}})$$

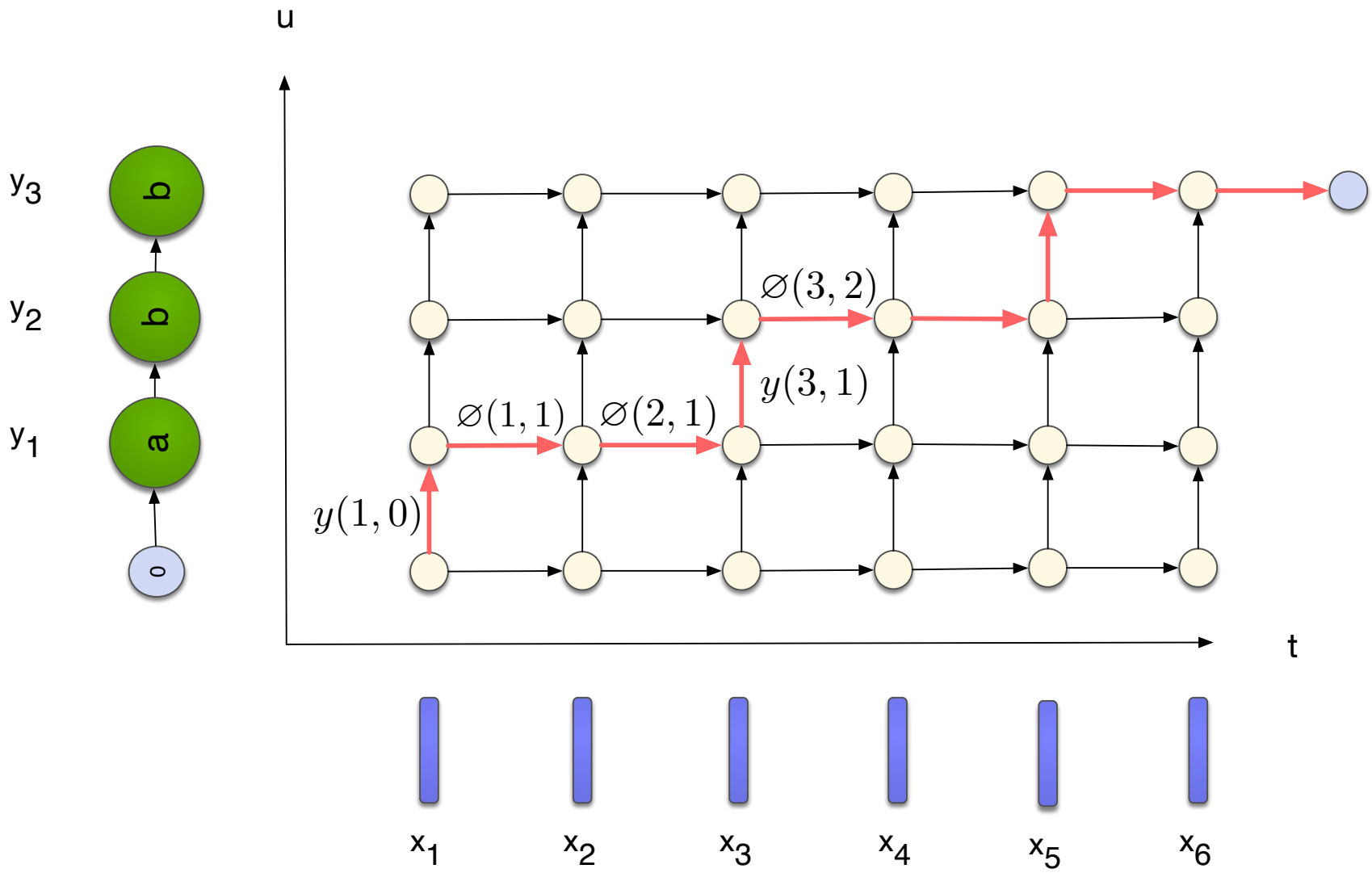
Define:

$$y(t, u) = P(y_{u+1} | h_t, p_u)$$
$$\varnothing(t, u) = P(\epsilon | h_t, p_u)$$

RNN-T: forward recursion



RNN-T: forward recursion



RNN-T: Forward recursion

- **Initialisation:**

$$\begin{aligned} \alpha_u(0) &= 1 \qquad u = 1 \\ &= 0 \qquad \textit{otherwise} \end{aligned}$$

- **Recursion:**

$$\begin{aligned} \alpha_u(t) =& \alpha_u(t-1)\varnothing(t-1,u) \\ &+ \alpha_{u-1}(t)y(t,u-1) \end{aligned}$$

- **Termination:**

$$P(Y|X) = \alpha_U(T)\varnothing(T,U)$$

RNN-T: Backward recursion

Define the backward probablity

$$\beta_u(t) = P(y_u, \ldots, y_U | h_t^{\text{enc}}, \ldots, h_T^{\text{enc}})$$

- **Initialisation:**

$$\beta_U(T) = \varnothing(T, U)$$

- **Recursion:**

$$\begin{aligned}\beta_u(t) =& \beta_u(t+1)\varnothing(t,u) \\ &+ \beta_{u+1}(t)y(t,u)\end{aligned}$$

RNN-T training

- The probability that y_u is emitted during transcription step t is given by the product

$$\alpha_u(t)\beta_u(t)$$

- RNN-T loss can be computed by

$$\mathcal{L}_{RNN-T} = -\log P(Y|X)$$

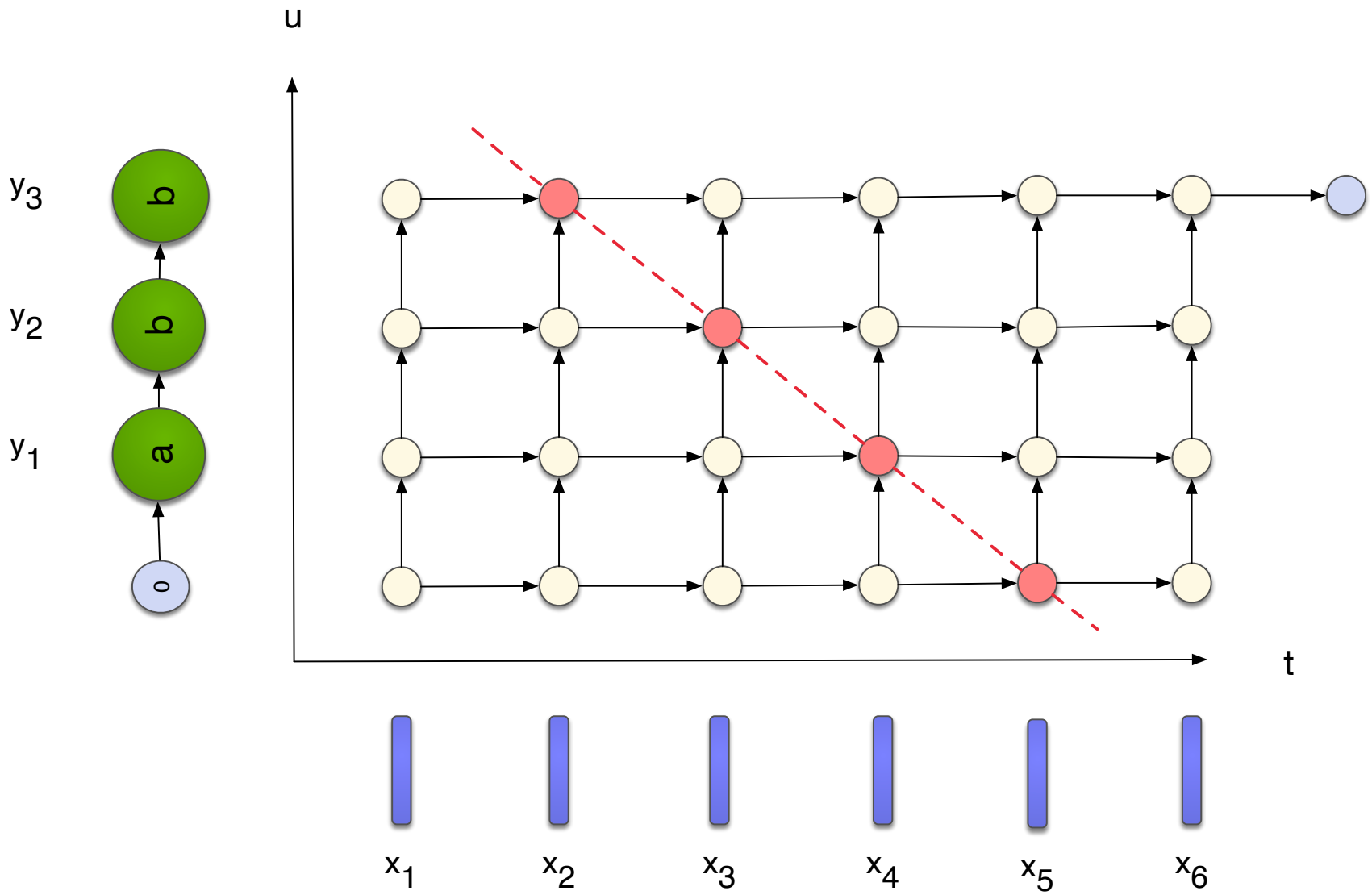
which can be expressed as

$$\mathcal{L}_{RNN-T} = -\log \sum_{(t,u):t+u=n} \alpha_u(t)\beta_u(t)$$

for any n in the range $1 \leq n \leq U + T$ (think of summing over the nodes on any top-left to bottom-right diagonal)

- Can perform backpropagation on this expression.

Summation



Comments on the RNN-T

- RNN transducer can operate left-to-right is a frame-synchronous manner (if the encoder is a unidirectional LSTM)
- Acoustic model (encoder) and language model (prediction network) parts are modelled independently and combined in the joint network. However everything is optimised to a common sequence-level objective (using the CTC loss function).
- With sufficient training data, additional language and pronunciation models are not necessary
- Google's first "all-neural" on-device speech recognition used unidirectional RNN transducers

  https://ai.googleblog.com/2019/03/an-all-neural-on-device-speech.html

Reading

- Alex Graves (2012), "Sequence Transduction with Recurrent Neural Networks", International Conference of Machine Learning (ICML) 2012 Workshop on Representation Learning
  https://arxiv.org/abs/1211.3711