Chapter ML:IX (continued)

IX. Deep Learning

- Elements of Deep Learning
- Convolutional Neural Networks
- Autoencoder Networks
- Recurrent Neural Networks
- RNNs for Machine Translation
- Attention Mechanism
- Self Attention and Transformers
- Transformer Language Models

Attention Mechanism

Vanishing Gradient Problem

Attention Mechanism

RNN with Long Short-Term Memory (LSTM)

[*SKIPPED*]

Remarks:

- LSTM is a recurrent neural network architecture that is very efficient at remembering long term dependencies and that is less vulnerable to the vanishing gradient problem.

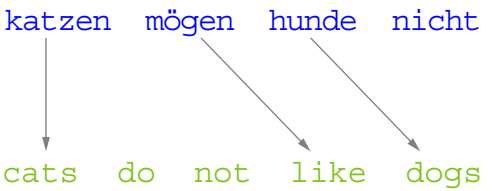
Attention Mechanism

RNN with Gated Recurrent Units (GRU)

[*SKIPPED*]

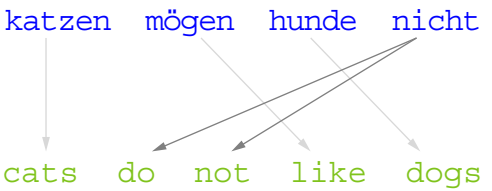
Attention Mechanism

Rationale of the Attention Mechanism



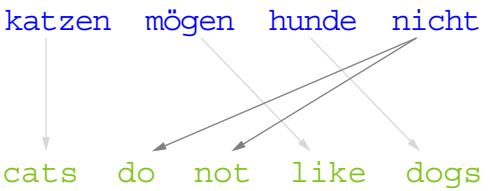
Attention Mechanism

Rationale of the Attention Mechanism (continued)



Attention Mechanism

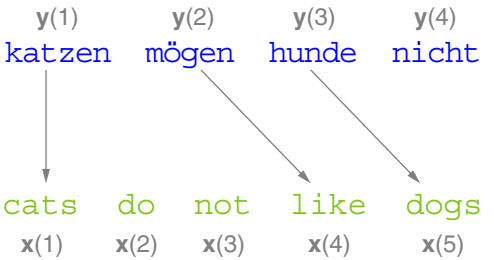
Rationale of the Attention Mechanism (continued)



The trained RNN ($\rightsquigarrow W^{\mathsf{h}}, W^{\mathsf{o}}$) has to generate in each time step the correct word.

Attention Mechanism

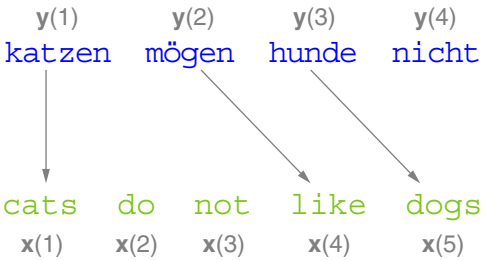
Rationale of the Attention Mechanism (continued)



The trained RNN ($\rightsquigarrow W^{\mathsf{h}}, W^{\mathsf{o}}$) has to generate in each time step the correct word.

Attention Mechanism

Rationale of the Attention Mechanism (continued)

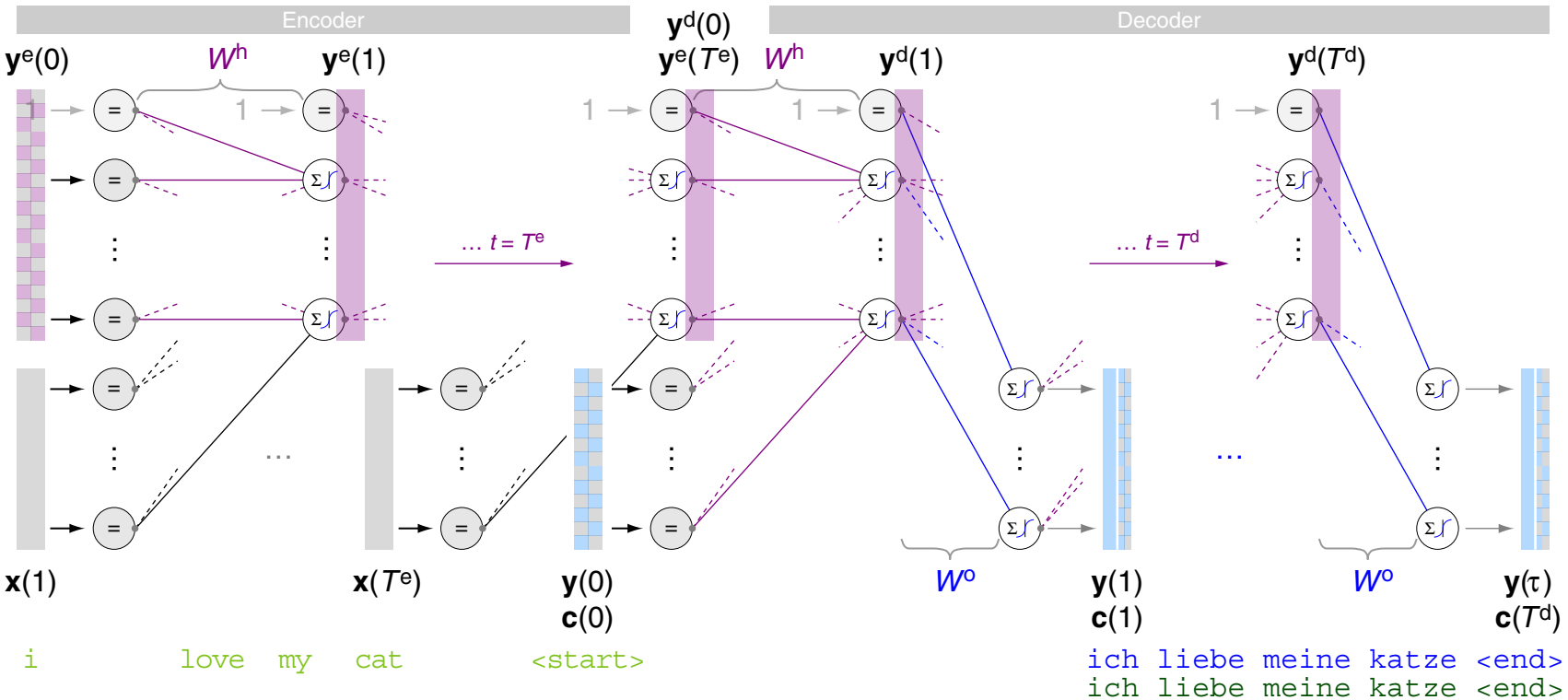


The trained RNN ($\rightsquigarrow W^{\mathsf{h}}, W^{\mathsf{o}}$) has to generate in each time step the correct word.

- By short-circuiting the decoder output with the encoder, alleviate the problem of creating suitable hidden vectors $\mathbf{y}^{\mathsf{d}}(t)$ from $\mathbf{y}^{\mathsf{e}}(T^{\mathsf{e}})$. This is called attention.
- Attention tackles (1) the bottleneck problem, (2) the vanishing gradient problem, and (3) the alignment problem of the translation task.

Attention Mechanism

Notation II (Omitting the Maths)



$\mathbf{x}(t)$ input
in_word

$\mathbf{y}^{h}(t)$ hidden
$\mathbf{y}^{e}(t)$ hidden encoder
$\mathbf{y}^{d}(t)$ hidden decoder

predefined hidden

$\mathbf{y}^{a}(t)$ attention

$\mathbf{y}(t)$ output
out_word (test)

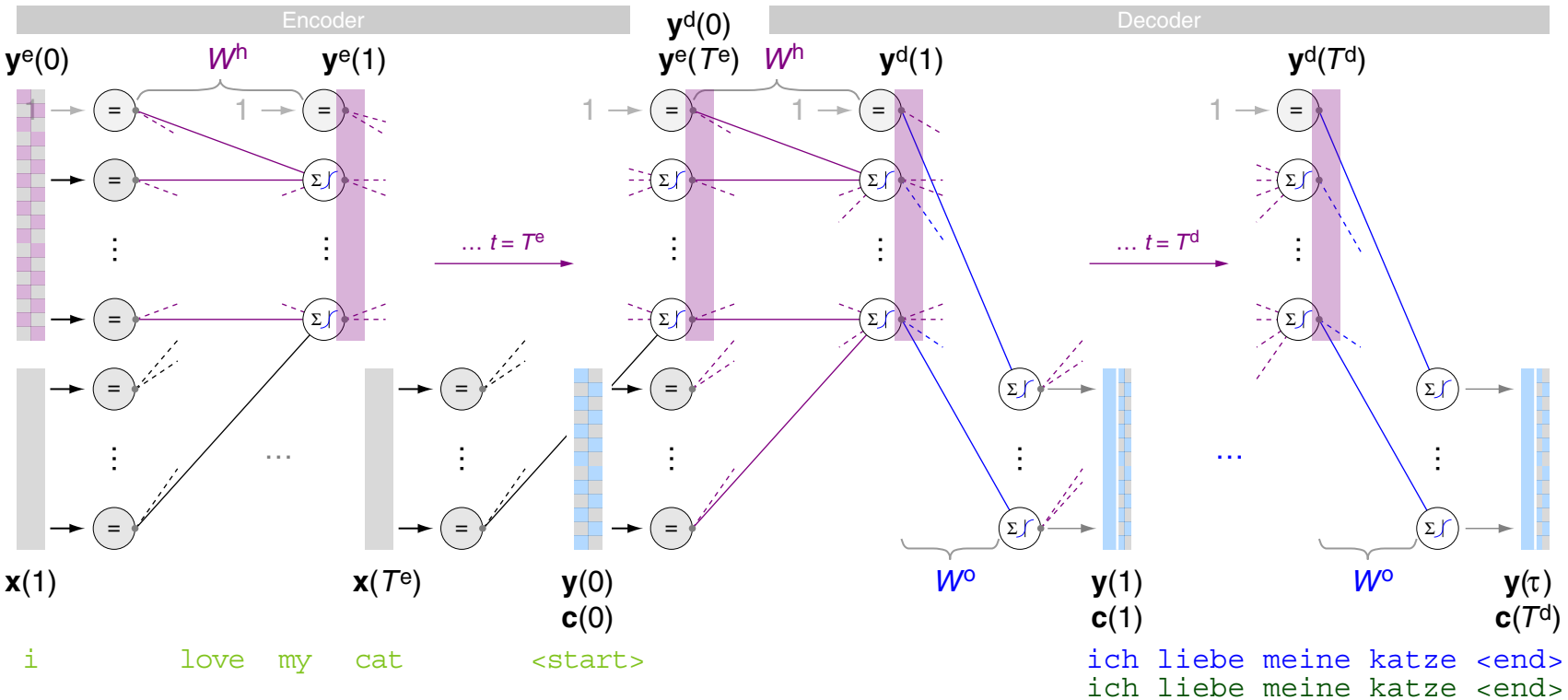
$\mathbf{c}(t)$ target
out_word (training)

$\mathbf{y}(t)$ output or $\mathbf{c}(t)$ target
(depends on training or test phase)

[Notation I]

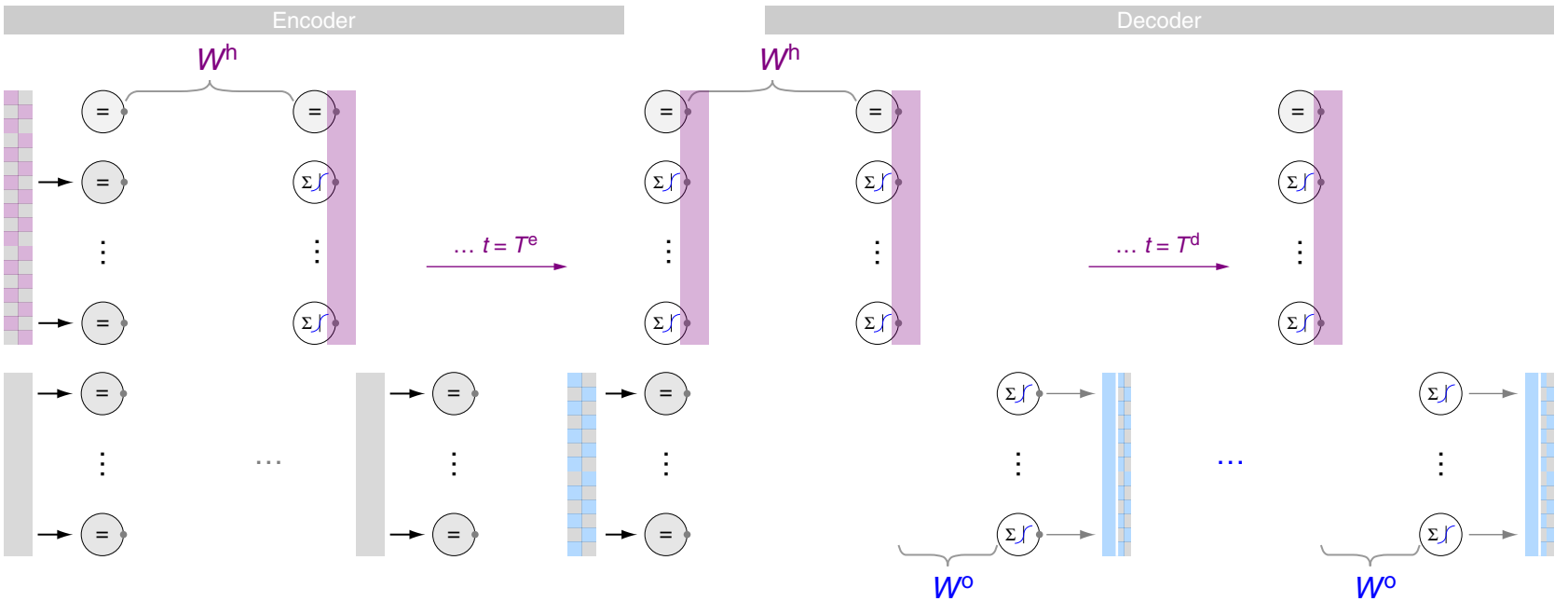
Attention Mechanism

Notation II (Omitting the Maths) (continued)



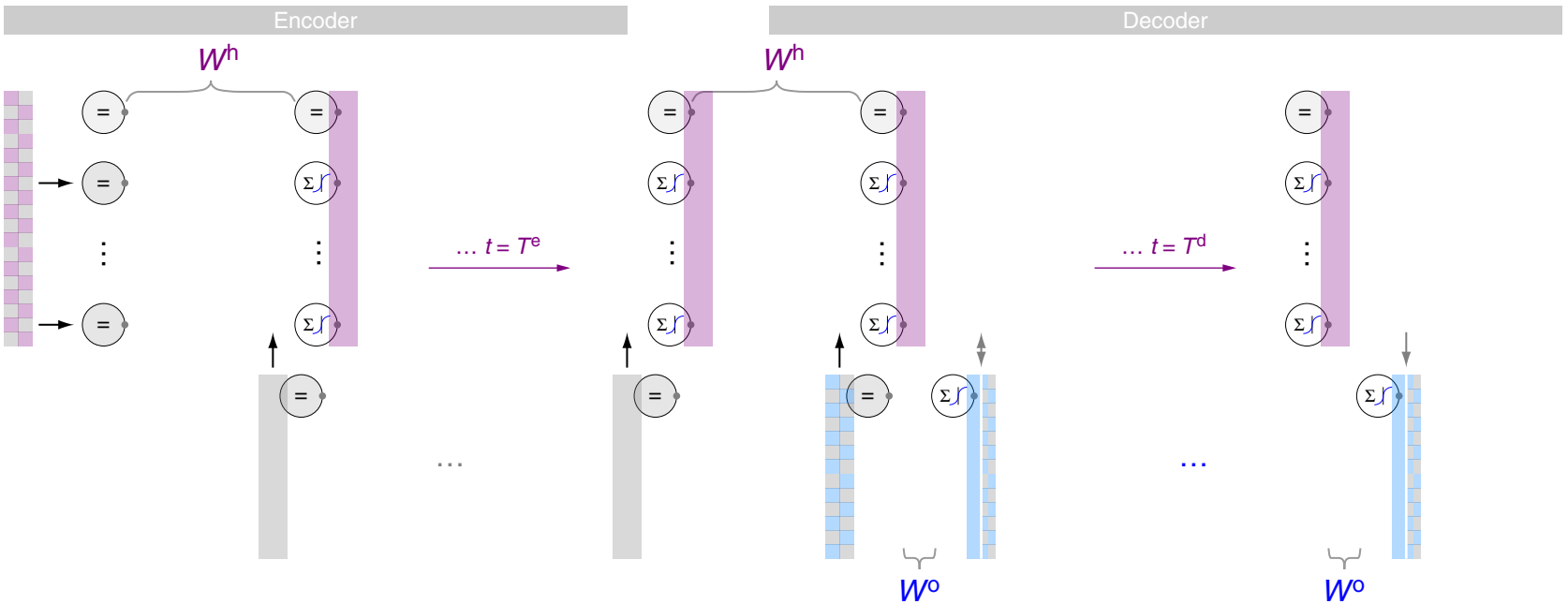
Attention Mechanism

Notation II (Omitting the Maths) (continued)



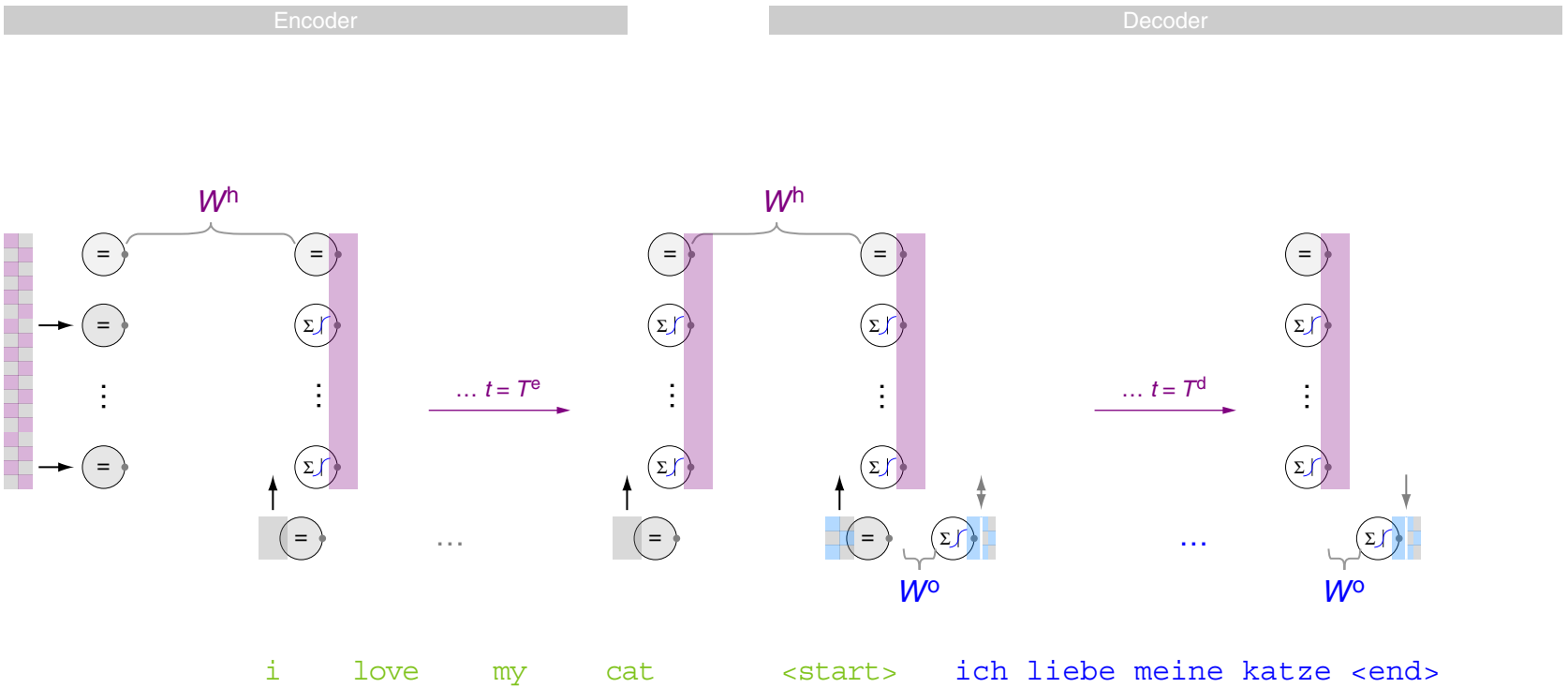
Attention Mechanism

Notation II (Omitting the Maths) (continued)



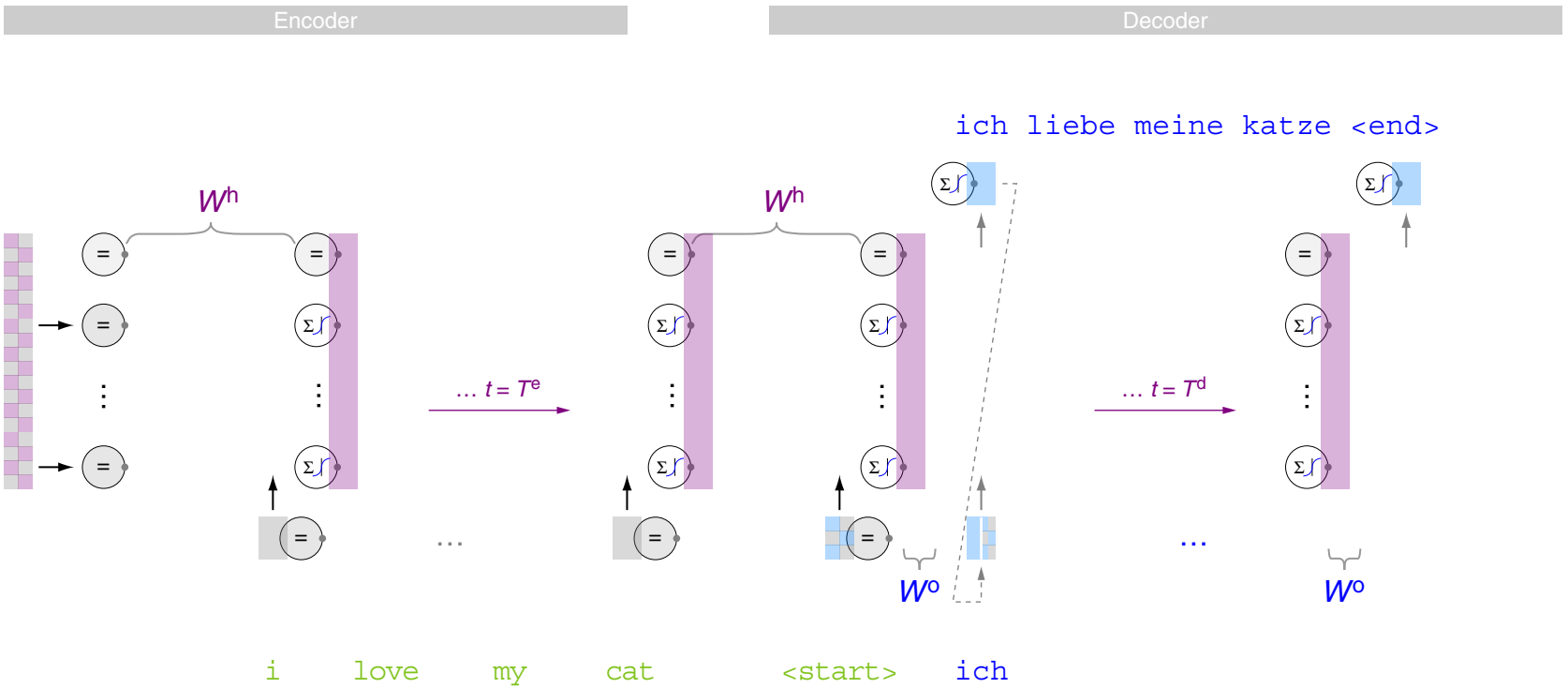
Attention Mechanism

Notation II (Omitting the Maths) (continued)



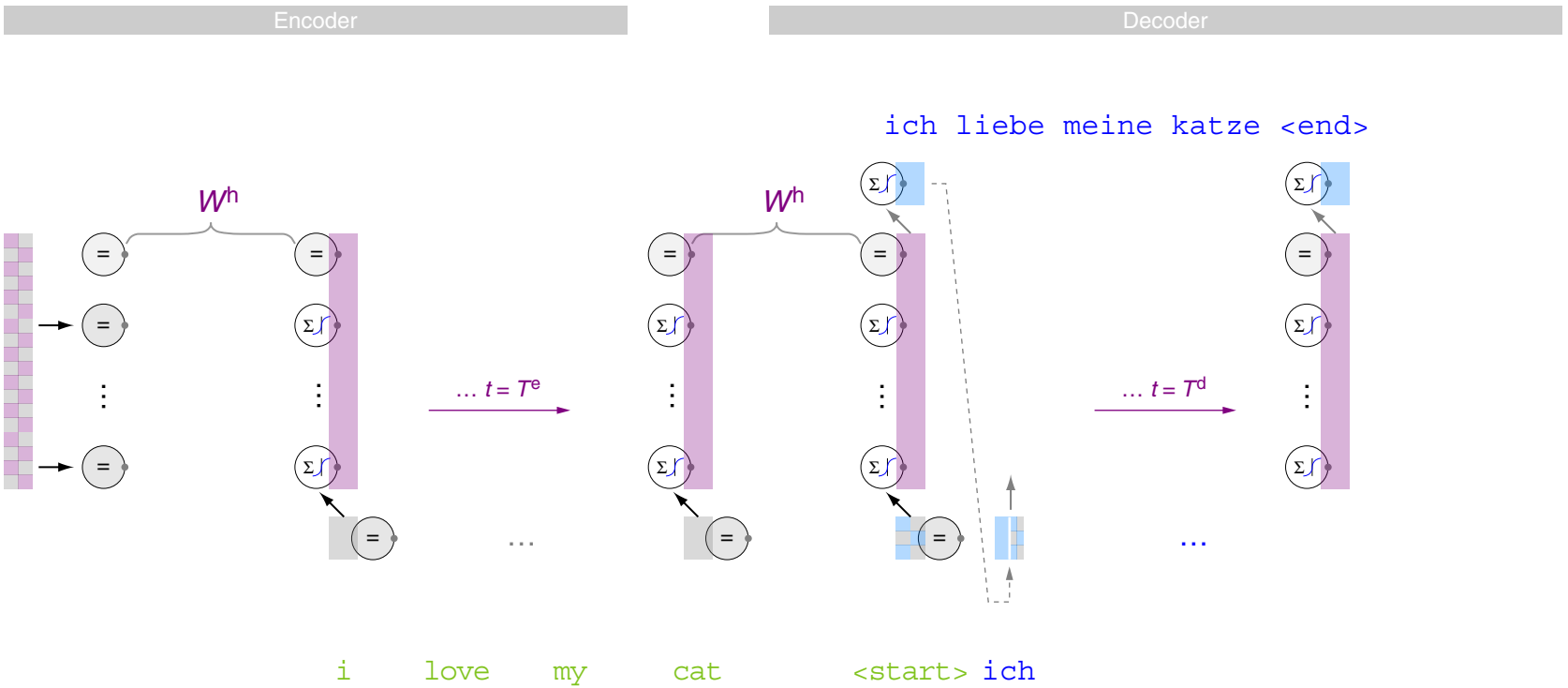
Attention Mechanism

Notation II (Omitting the Maths) (continued)



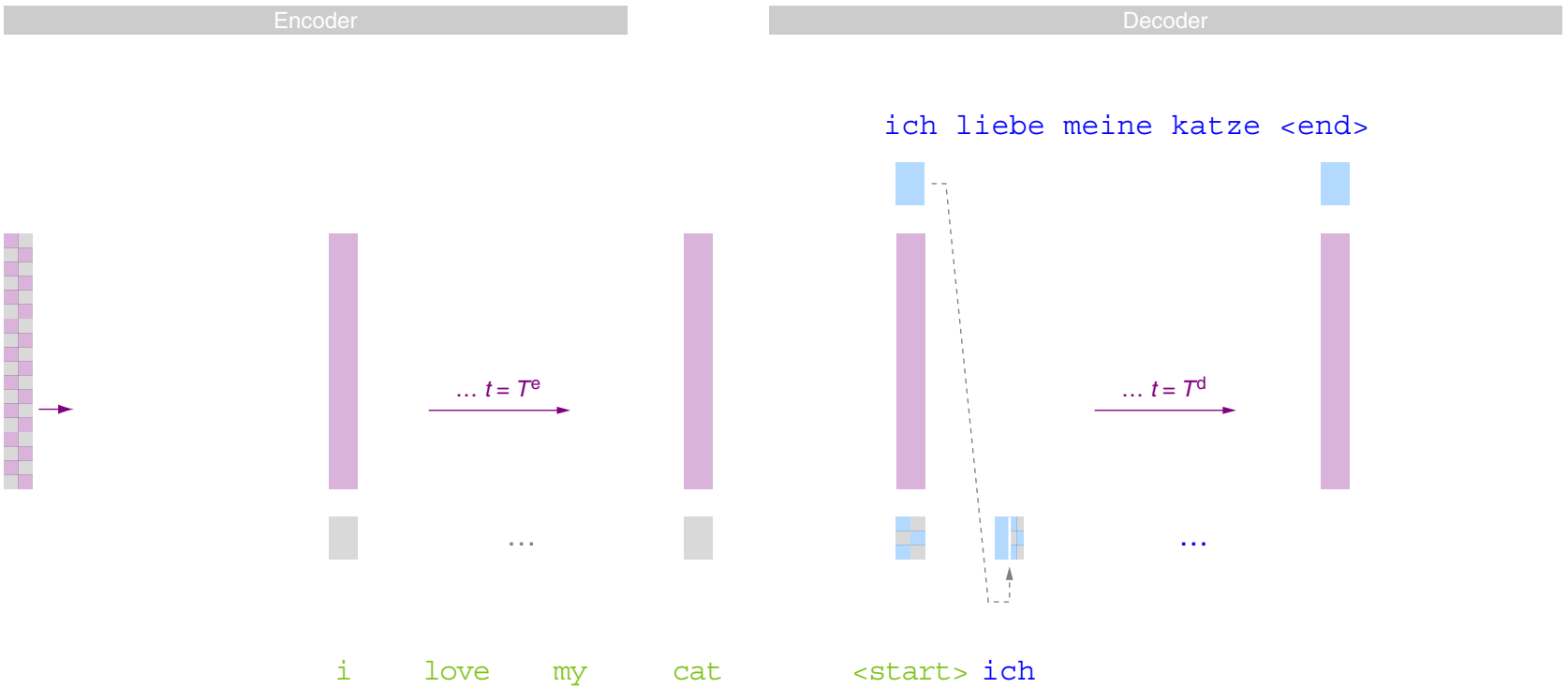
Attention Mechanism

Notation II (Omitting the Maths) (continued)



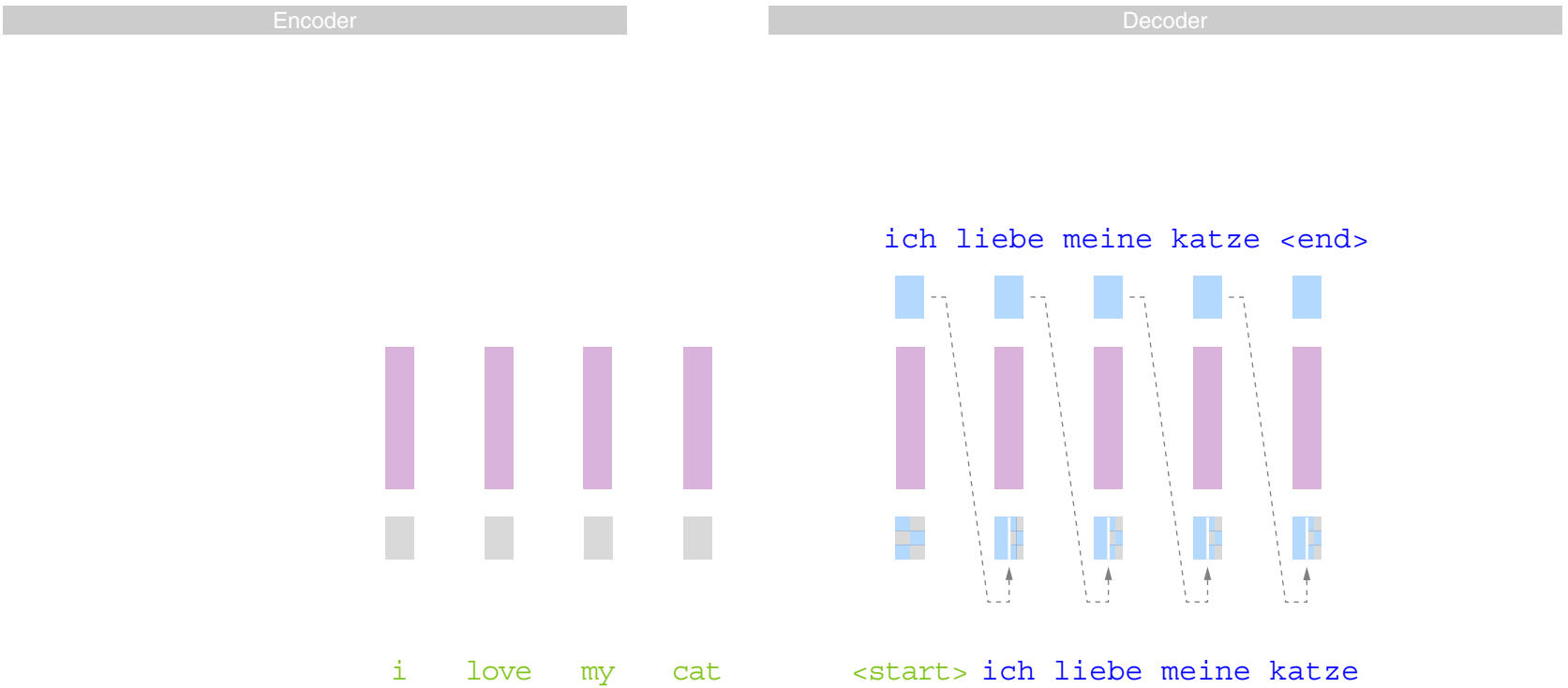
Attention Mechanism

Notation II (Omitting the Maths) (continued)



Attention Mechanism

Notation II (Omitting the Maths) (continued)

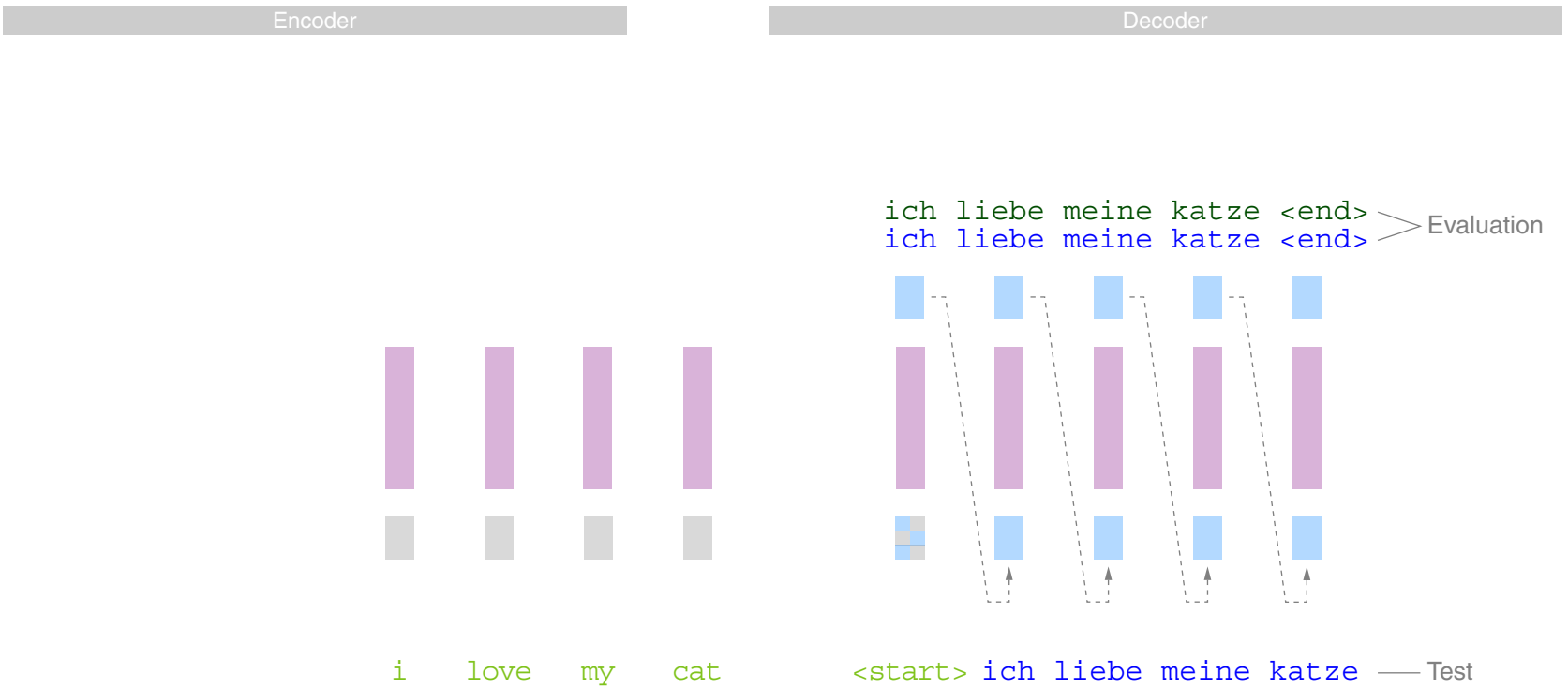


Attention Mechanism

Notation II (Omitting the Maths) (continued)

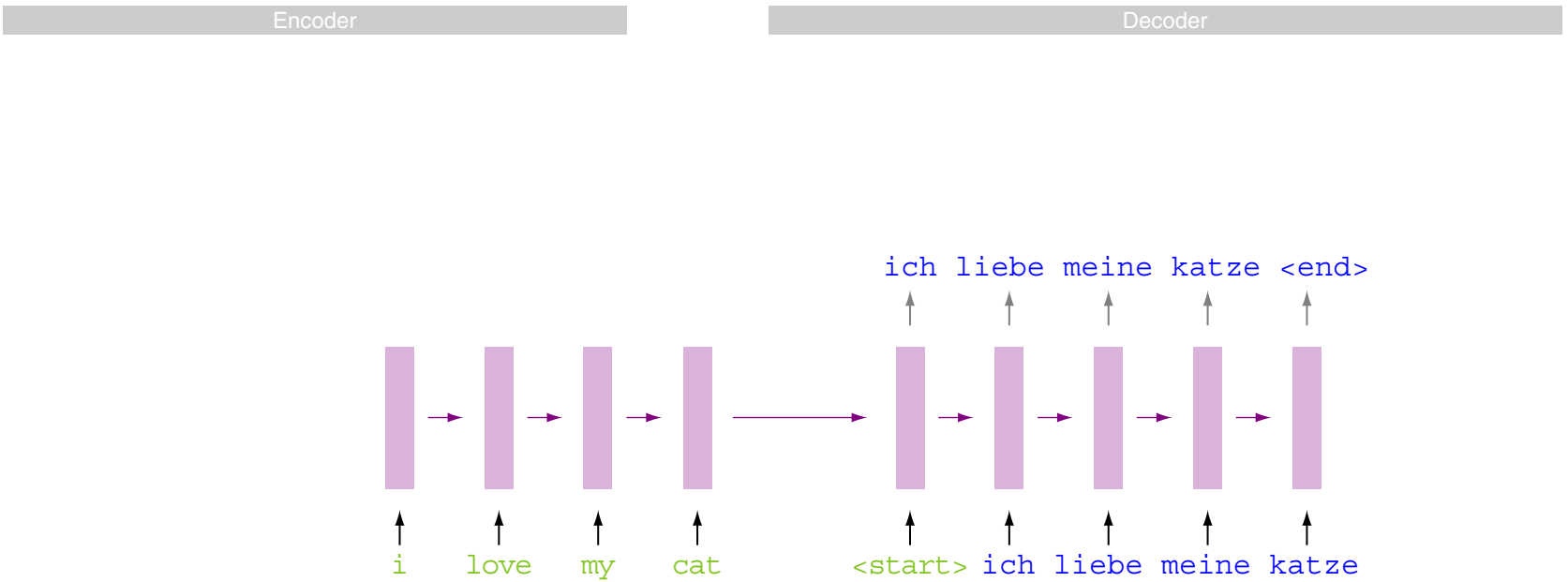
Attention Mechanism

Notation II (Omitting the Maths) (continued)



Attention Mechanism

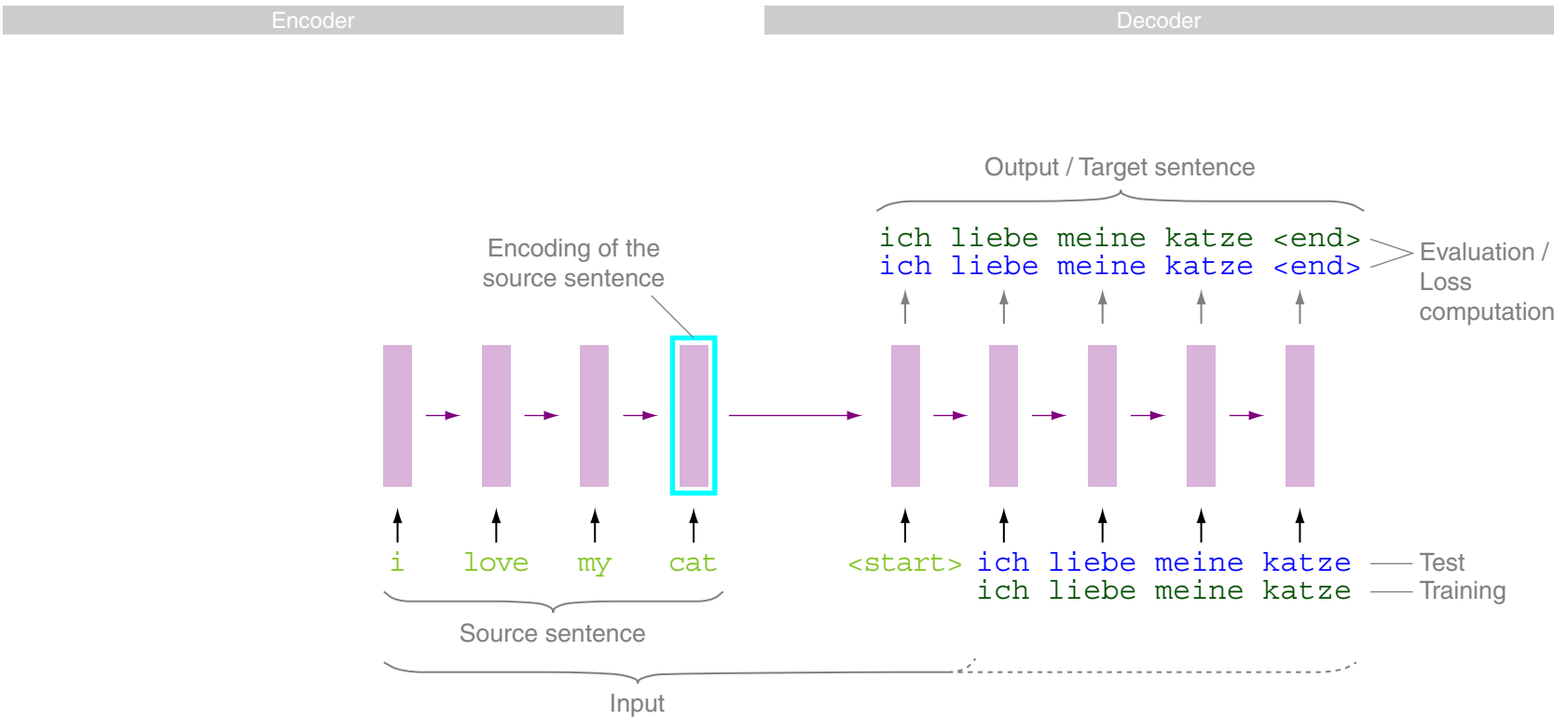
Notation II (Omitting the Maths) (continued)



This level of abstraction is used among others by [Manning 2021, lecture CS224N].

Attention Mechanism

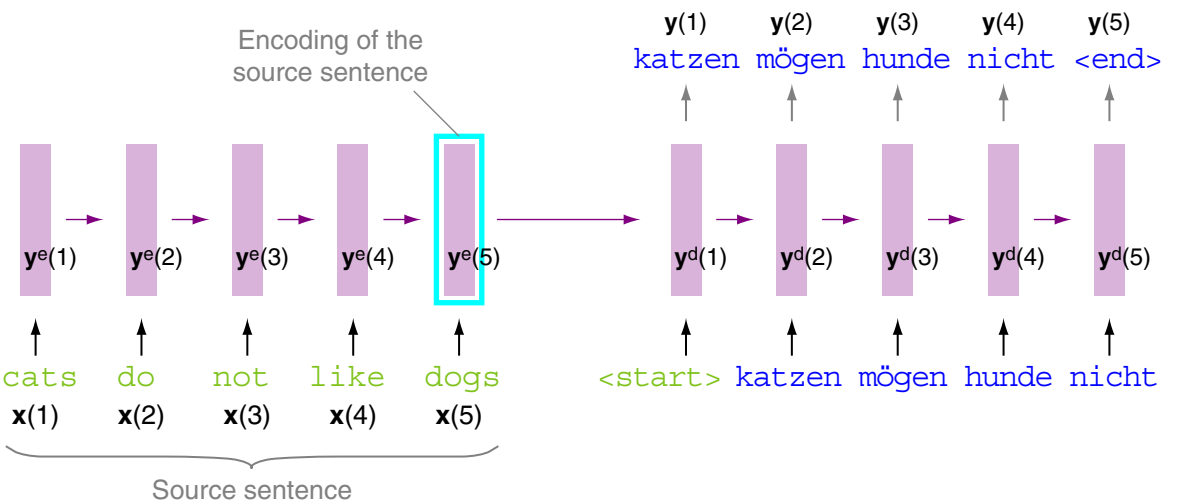
Notation II (Omitting the Maths) (continued)



This level of abstraction is used among others by [Manning 2021, lecture CS224N].

Attention Mechanism

Query the Encoder to Tweak the Decoder Output



Remarks (observations) :

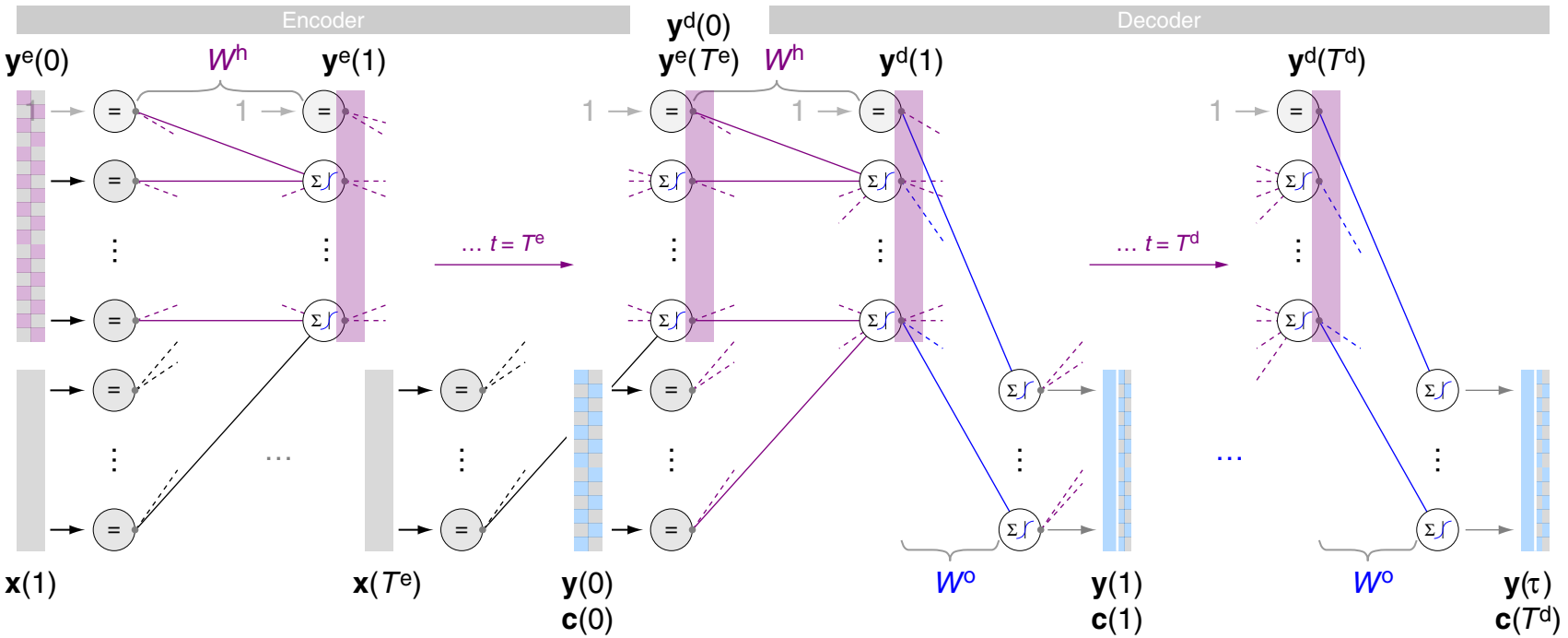
- Regarding the bottleneck problem. Attention allows the decoder to get additional information directly from the source sentence, this way bypassing the bottleneck $\mathbf{y}^{\mathsf{e}}(T^{\mathsf{e}})$ (here: $\mathbf{y}^{\mathsf{e}}(5)$).
- Regarding the vanishing gradient problem. Attention allows the decoder to look back into the past as far as it wants.
- Regarding the alignment problem. By inspecting the attention distribution $\mathbf{a}_1(t)$ for the output at time t, we can see what the decoder was focusing on and hence consider changes of the word order.

  For example, without attention one has to manage to encode the sequence "`cats do not like dogs <start> katzen`" as a hidden vector $\mathbf{y}^{\mathsf{d}}(2)$ from which $\mathbf{y}(2)$, "`mögen`", is decoded. I.e., in addition to looking back in time (keyword: vanishing gradient) the decoder, being at $t{=}2$, has to focus on the encoder at $t{=}4$, since the negation in the English sentence is before "`like`", while the negation in the German sentence is after "`mögen`".

  With attention, the decoder can observe the importance of $\mathbf{y}^{\mathsf{e}}(4)$, "`cats do not like`", for $\mathbf{y}^{\mathsf{d}}(2)$, and it will consider $\mathbf{y}^{\mathsf{e}}(4)$ with a high weight to compute the output $\mathbf{y}(2)$.

Attention Mechanism

RNN with Simple Attention



Input:

$\mathbf{x}, \mathbf{y}^{\mathrm{e}}(0), \mathbf{y}(t), \mathbf{y}(0) \mathrel{\widehat{=}}$ <start>

Output:

$$\mathbf{y}(t) = \boldsymbol{\sigma}_1\left(W^{\mathrm{o}}\,\mathbf{y}^{\mathrm{d}}(t)\right), t = 1, \ldots, \tau$$

Hidden:

$$\mathbf{y}^{\mathrm{e}}(t) = \boldsymbol{\sigma}\left(W^{\mathrm{h}}\begin{pmatrix}\mathbf{y}^{\mathrm{e}}(t-1)\\ \mathbf{x}(t)\end{pmatrix}\right), t = 1, \ldots, T^{\mathrm{e}}$$

$$\mathbf{y}^{\mathrm{d}}(t) = \boldsymbol{\sigma}\left(W^{\mathrm{h}}\begin{pmatrix}\mathbf{y}^{\mathrm{d}}(t-1)\\ \mathbf{y}(t-1)\end{pmatrix}\right), t = 1, \ldots, \tau$$

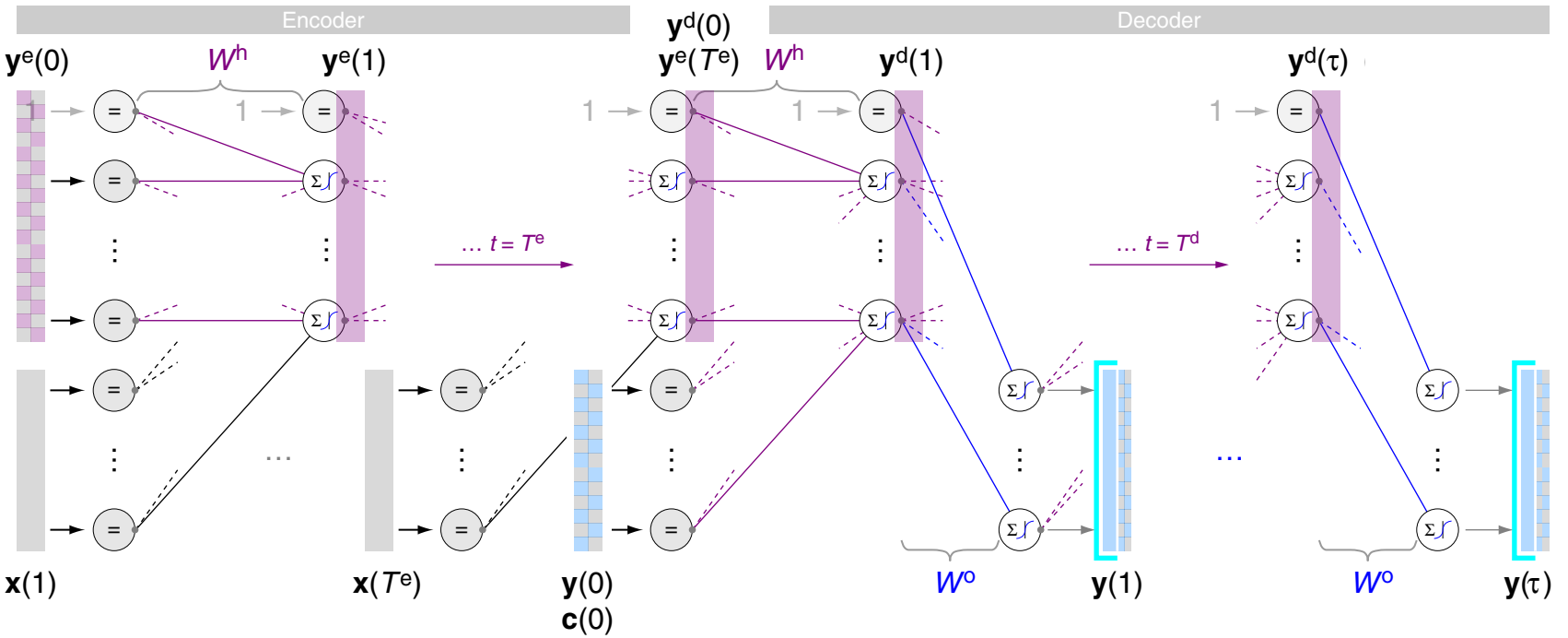
Target:

$[\mathbf{c}(1), \ldots, \mathbf{c}(T)]$

$\mathbf{c}(T) \mathrel{\widehat{=}}$ <end>

Attention Mechanism

RNN with Simple Attention (continued)



Output:

$$\mathbf{y}(t) = \boldsymbol{\sigma}_1\left(W^{\mathrm{o}}\,\mathbf{y}^{\mathrm{d}}(t)\right)$$

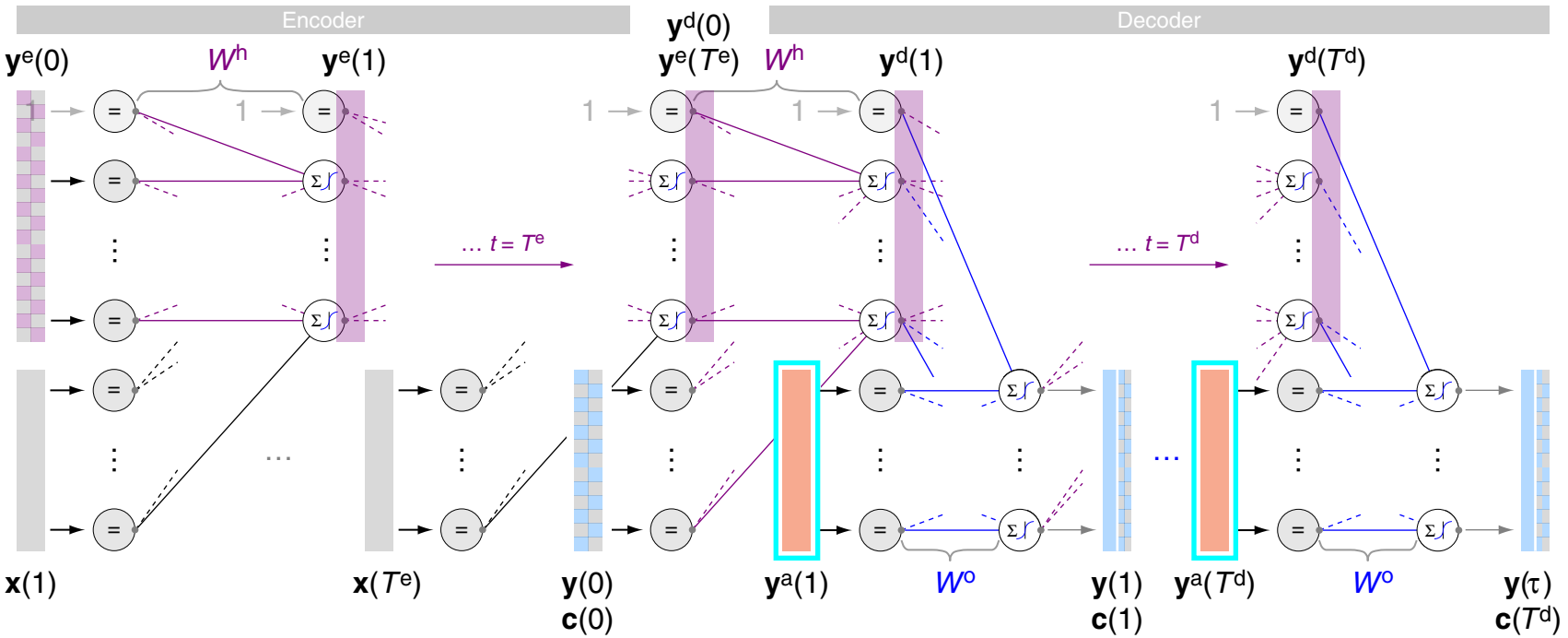
Attention:

$$\mathbf{y}^{\mathrm{a}}(t) = \left[\mathbf{y}^{\mathrm{e}}(1), \ldots, \mathbf{y}^{\mathrm{e}}(T^{\mathrm{e}})\right]\ \mathbf{a}(t)$$

$$\mathbf{a}(t) = \left[\mathbf{y}^{\mathrm{e}}(1), \ldots, \mathbf{y}^{\mathrm{e}}(T^{\mathrm{e}})\right]^{T}\ \mathbf{y}^{\mathrm{d}}(t),\quad t = 1, \ldots, T^{d}$$

Attention Mechanism

RNN with Simple Attention (continued)



Output:

$$\mathbf{y}(t) = \boldsymbol{\sigma}_1 \left(W^{\mathrm{o}} \begin{pmatrix} \mathbf{y}^{\mathrm{d}}(t) \\ \mathbf{y}^{\mathrm{a}}(t) \end{pmatrix} \right)$$

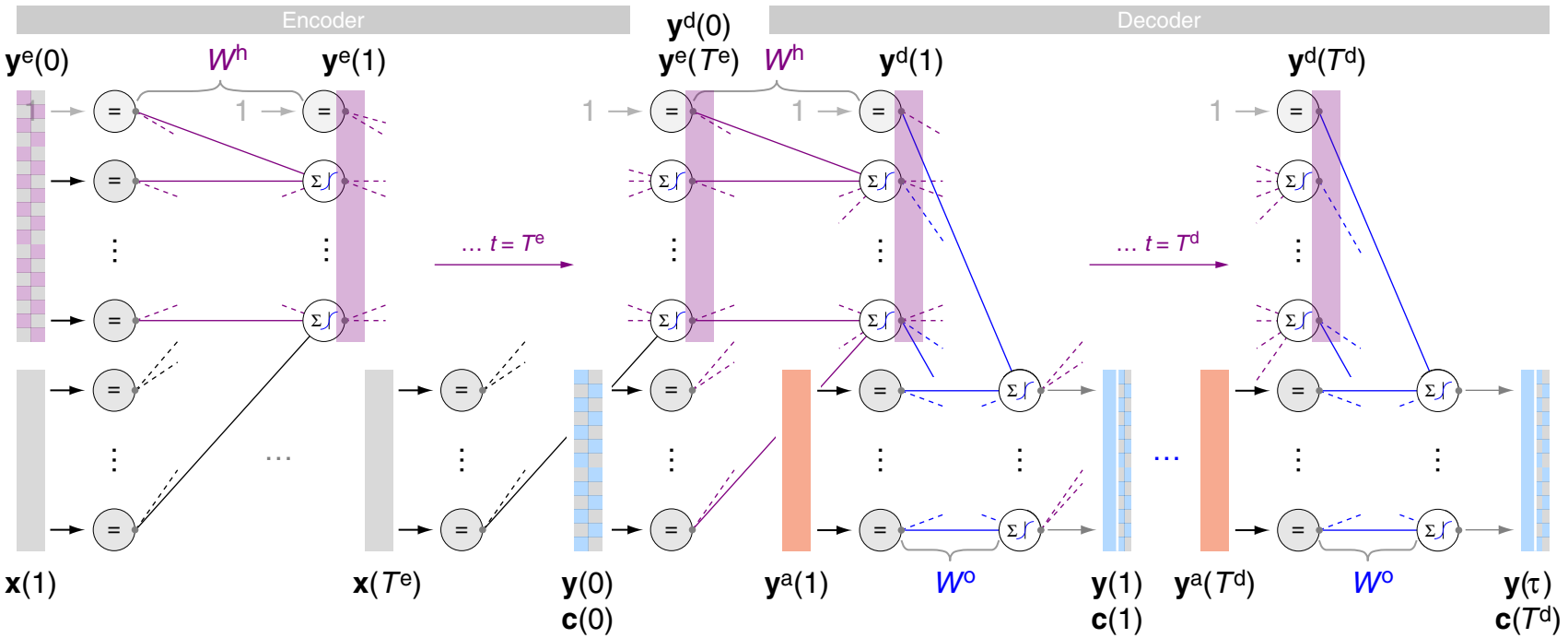
Attention:

$$\mathbf{y}^{\mathrm{a}}(t) = \left[\mathbf{y}^{\mathrm{e}}(1), \dots, \mathbf{y}^{\mathrm{e}}(T^{\mathrm{e}}) \right] \mathbf{a}(t)$$

$$\mathbf{a}(t) = \left[\mathbf{y}^{\mathrm{e}}(1), \dots, \mathbf{y}^{\mathrm{e}}(T^{\mathrm{e}}) \right]^T \mathbf{y}^{\mathrm{d}}(t), \quad t = 1, \dots, T^d$$

Attention Mechanism

RNN with Simple Attention (continued)



Output:

$$\mathbf{y}(t) = \boldsymbol{\sigma}_1 \left(W^{\mathrm{o}} \begin{pmatrix} \mathbf{y}^{\mathrm{d}}(t) \\ \mathbf{y}^{\mathrm{a}}(t) \end{pmatrix} \right)$$

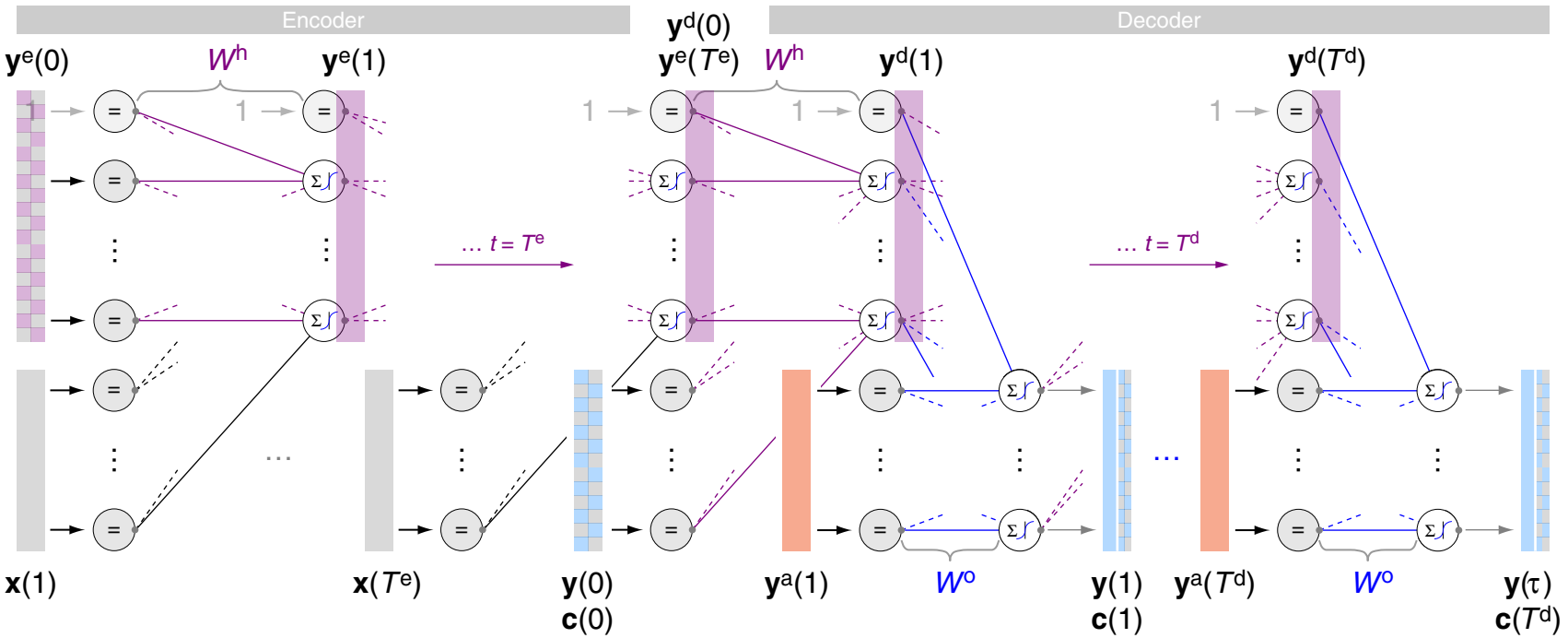
Attention:

$$\mathbf{y}^{\mathrm{a}}(t) = \left[\mathbf{y}^{\mathrm{e}}(1), \ldots, \mathbf{y}^{\mathrm{e}}(T^{\mathrm{e}}) \right] \mathbf{a}(t)$$

$$\mathbf{a}(t) = \left[\mathbf{y}^{\mathrm{e}}(1), \ldots, \mathbf{y}^{\mathrm{e}}(T^{\mathrm{e}}) \right]^T \mathbf{y}^{\mathrm{d}}(t), \quad t = 1, \ldots, T^d$$

Attention Mechanism

RNN with Simple Attention (continued)



Output:

$$\mathbf{y}(t) = \boldsymbol{\sigma}_1\left(W^{\mathrm{o}}\begin{pmatrix}\mathbf{y}^{\mathrm{d}}(t)\\ \mathbf{y}^{\mathrm{a}}(t)\end{pmatrix}\right)$$

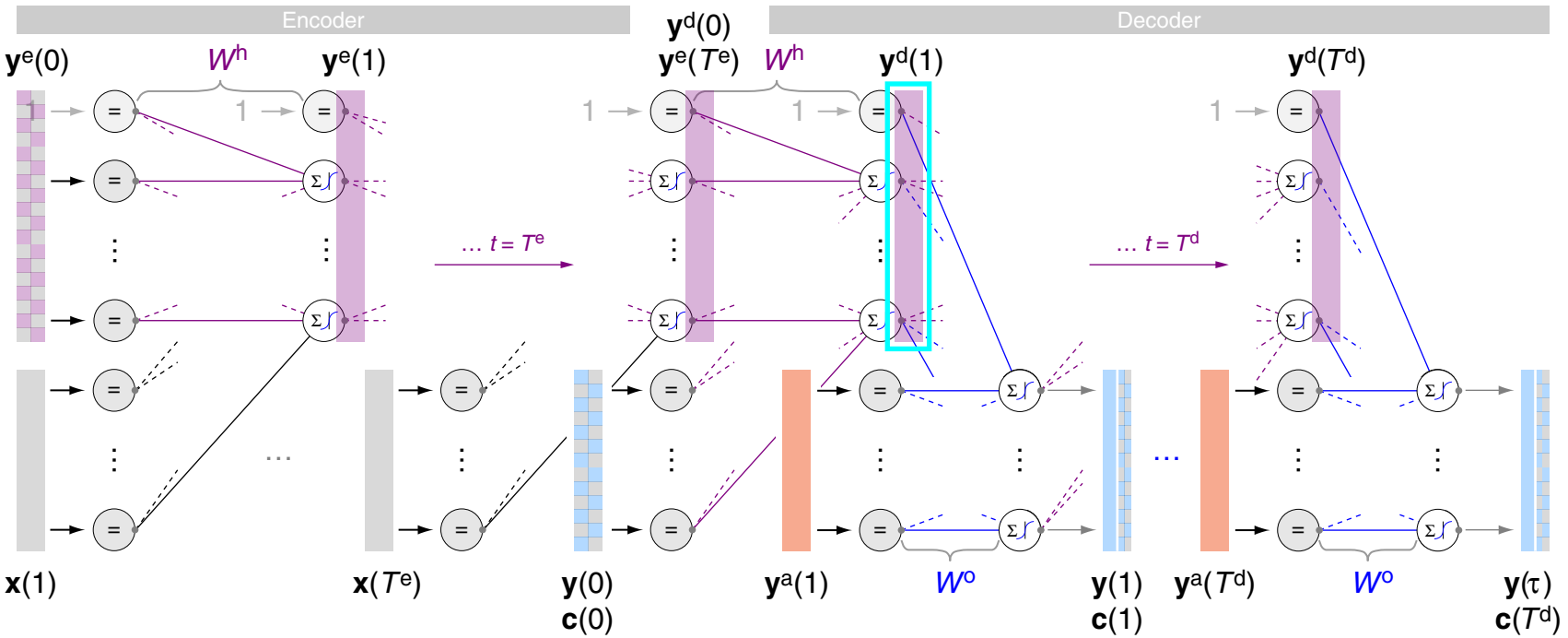
Attention:

$$\mathbf{y}^{\mathrm{a}}(t) = \left[\mathbf{y}^{\mathrm{e}}(1),\ldots,\mathbf{y}^{\mathrm{e}}(T^{\mathrm{e}})\right]\ \mathbf{a}_1(t),\quad \mathbf{a}_1(t) = \boldsymbol{\sigma}_1\Big(\mathbf{a}(t)\Big)$$

$$\mathbf{a}(t) = \left[\mathbf{y}^{\mathrm{e}}(1),\ldots,\mathbf{y}^{\mathrm{e}}(T^{\mathrm{e}})\right]^T\ \mathbf{y}^{\mathrm{d}}(t),\quad t = 1,\ldots,T^d$$

Attention Mechanism

RNN with Simple Attention (continued)



Output:

$$\mathbf{y}(t) = \boldsymbol{\sigma}_1\left(W^{\mathrm{o}}\begin{pmatrix}\mathbf{y}^{\mathrm{d}}(t)\\ \mathbf{y}^{\mathrm{a}}(t)\end{pmatrix}\right)$$

Attention:

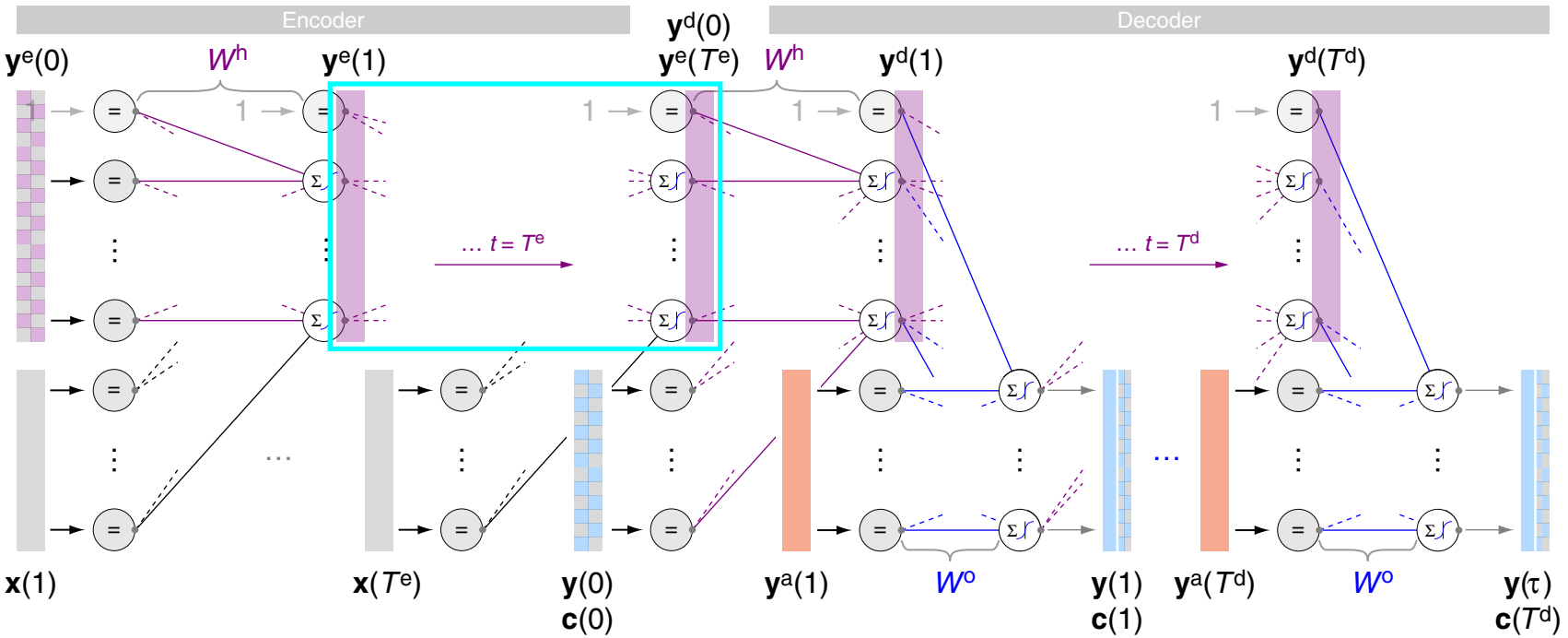
$$\mathbf{y}^{\mathrm{a}}(t) = \left[\mathbf{y}^{\mathrm{e}}(1), \ldots, \mathbf{y}^{\mathrm{e}}(T^{\mathrm{e}})\right] \mathbf{a}_1(t), \quad \mathbf{a}_1(t) = \boldsymbol{\sigma}_1\left(\mathbf{a}(t)\right)$$

$$\mathbf{a}(t) = \left[\mathbf{y}^{\mathrm{e}}(1), \ldots, \mathbf{y}^{\mathrm{e}}(T^{\mathrm{e}})\right]^T \mathbf{y}^{\mathrm{d}}(t), \quad t = 1, \ldots, T^d$$

"Queries"

Attention Mechanism

RNN with Simple Attention (continued)



Output:

$$\mathbf{y}(t) = \boldsymbol{\sigma}_1\left(W^{o}\begin{pmatrix}\mathbf{y}^{d}(t)\\ \mathbf{y}^{a}(t)\end{pmatrix}\right)$$

Attention:

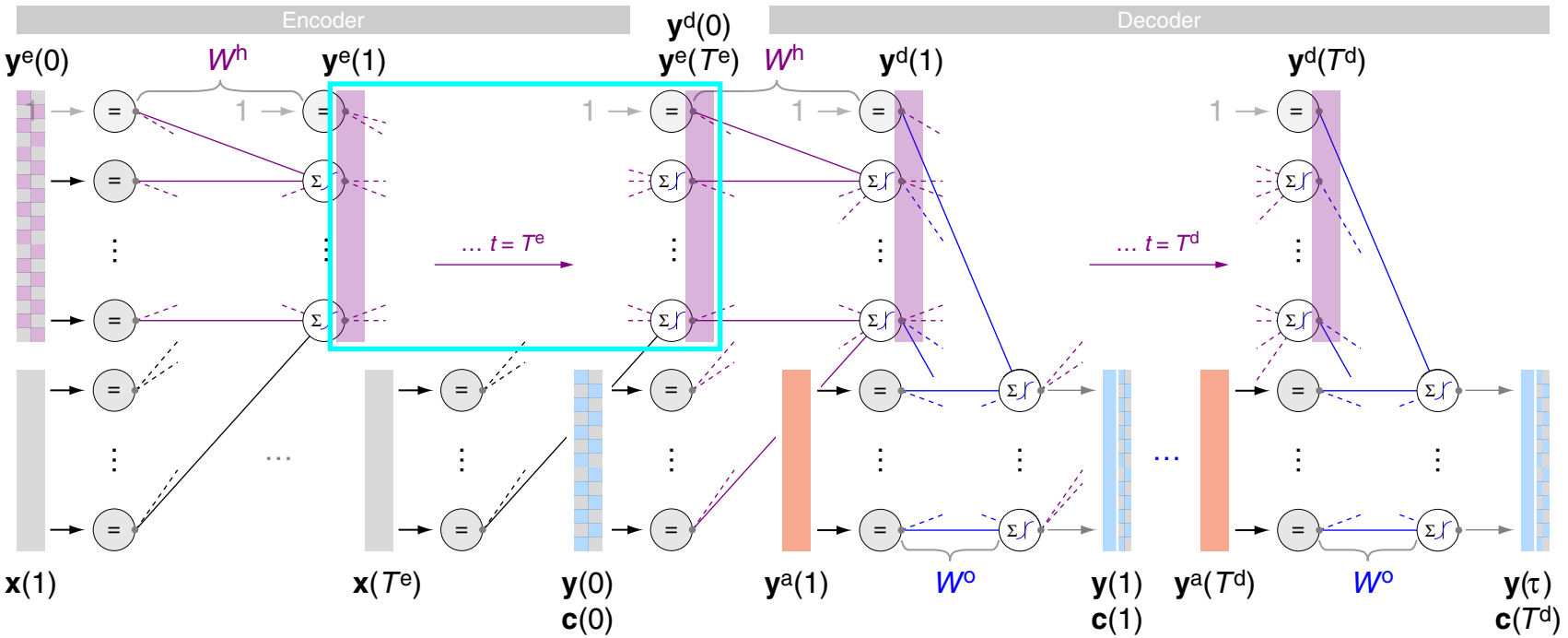
$$\mathbf{y}^{a}(t) = \left[\mathbf{y}^{e}(1),\ldots,\mathbf{y}^{e}(T^{e})\right]\ \mathbf{a}_1(t),\quad \mathbf{a}_1(t) = \boldsymbol{\sigma}_1\Big(\mathbf{a}(t)\Big)$$

$$\mathbf{a}(t) = \left[\mathbf{y}^{e}(1),\ldots,\mathbf{y}^{e}(T^{e})\right]^{T}\mathbf{y}^{d}(t),\quad t = 1,\ldots,T^{d}$$

"Keys"

Attention Mechanism

RNN with Simple Attention (continued)



Output:

$$\mathbf{y}(t) = \boldsymbol{\sigma}_1\left(W^{o}\begin{pmatrix}\mathbf{y}^{d}(t)\\ \mathbf{y}^{a}(t)\end{pmatrix}\right)$$

Attention:

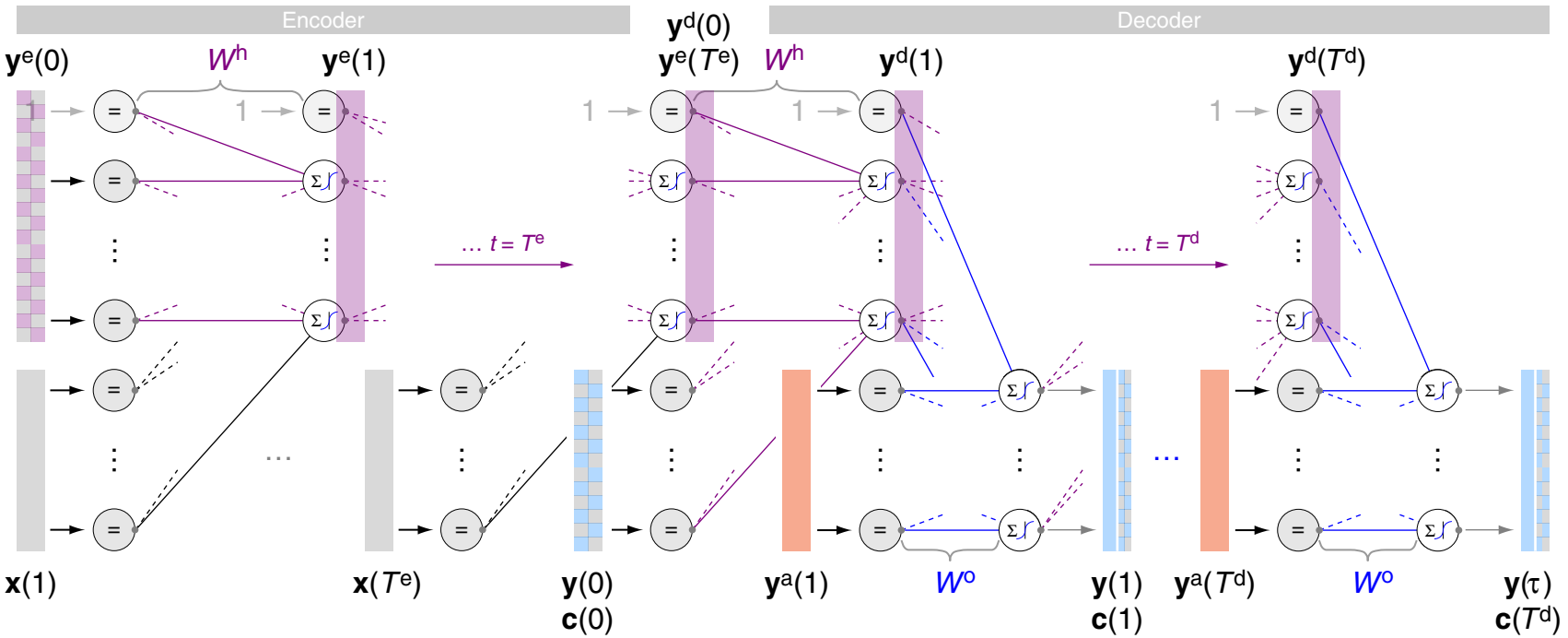
$$\mathbf{y}^{a}(t) = \left[\mathbf{y}^{e}(1),\ldots,\mathbf{y}^{e}(T^{e})\right]\mathbf{a}_1(t), \quad \mathbf{a}_1(t) = \boldsymbol{\sigma}_1\left(\mathbf{a}(t)\right)$$

"Values"

$$\mathbf{a}(t) = \left[\mathbf{y}^{e}(1),\ldots,\mathbf{y}^{e}(T^{e})\right]^{T}\mathbf{y}^{d}(t), \quad t = 1,\ldots,T^{d}$$

Attention Mechanism

RNN with Parameterized Attention



Output:

$$\mathbf{y}(t) = \boldsymbol{\sigma}_1\left(W^{\mathrm{o}}\begin{pmatrix}\mathbf{y}^{\mathrm{d}}(t)\\ \mathbf{y}^{\mathrm{a}}(t)\end{pmatrix}\right)$$

Attention:

$$\mathbf{y}^{\mathrm{a}}(t) = \left(W^{\mathrm{V}}\left[\mathbf{y}^{\mathrm{e}}(1),\dots,\mathbf{y}^{\mathrm{e}}(T^{\mathrm{e}})\right]\right)\mathbf{a}_1(t),\quad \mathbf{a}_1(t) = \boldsymbol{\sigma}_1\left(\mathbf{a}(t)\right)$$

$$\mathbf{a}(t) = \left(W^{\mathrm{K}}\left[\mathbf{y}^{\mathrm{e}}(1),\dots,\mathbf{y}^{\mathrm{e}}(T^{\mathrm{e}})\right]\right)^{T}\left(W^{\mathrm{Q}}\,\mathbf{y}^{\mathrm{d}}(t)\right)$$

Remarks (attention calculus):

- As before, $\boldsymbol{\sigma}_1()$ denotes the softmax function.
- $\mathbf{a}(t)$ is called vector of *attention scores*, $\mathbf{a}_1(t) := \boldsymbol{\sigma}_1(\mathbf{a}(t))$ is the *attention distribution*. [Manning 2021]
- The ith component $a_i(t)$ of the attention score vector $\mathbf{a}(t)$, $i = 1, \ldots, T^{\mathsf{e}}$, models the importance of the ith *encoder* hidden state $\mathbf{y}^{\mathsf{e}}(i)$ for the *decoder* hidden state $\mathbf{y}^{\mathsf{d}}(t)$: $a_i(t)$ is the scalar product of $\mathbf{y}^{\mathsf{e}}(i)$ and $\mathbf{y}^{\mathsf{d}}(t)$ (do not overlook the matrix transpose operation).
- $\mathbf{y}^{\mathsf{a}}(t)$ is the result of weighting the encoder hidden state sequence $[\mathbf{y}^{\mathsf{e}}(1), \ldots, \mathbf{y}^{\mathsf{e}}(T^{\mathsf{e}})]$ with the attention distribution $\mathbf{a}_1(t)$. I.e., each vector $\mathbf{y}^{\mathsf{e}}(i)$ is considered as a "value" that is weighted with the relative importance stored in the respective dimension (= time step) of $\mathbf{a}_1(t)$.

  $\mathbf{y}^{\mathsf{a}}(t)$ is called *attention* [vector] for output vector $\mathbf{y}(t)$ since it helps to pay attention to the most influential input states for $\mathbf{y}(t)$.
- Consider $\mathbf{a}(t)$. The scalar product of $\mathbf{y}^{\mathsf{e}}(i)$ and $\mathbf{y}^{\mathsf{d}}(t)$ becomes maximum if $\mathbf{y}^{\mathsf{e}}(i)$ and $\mathbf{y}^{\mathsf{d}}(t)$ are identical. The distribution $\mathbf{a}_1(t)$ of the T^{e} weights of $\mathbf{a}(t)$ reflects the distribution of absolute values among the $\mathbf{y}^{\mathsf{e}}$.

  Consider $\mathbf{a}_1(t)$. If some $\mathbf{y}^{\mathsf{e}}(i)$ has a high absolute value (compared to the other $\mathbf{y}^{\mathsf{e}}$) and if it is similar to $\mathbf{y}^{\mathsf{d}}(t)$, it will push the weight of the ith dimension of $\mathbf{a}_1(t)$ towards 1 (and the others towards zero).

  In the extreme case, $\mathbf{y}^{\mathsf{d}}(t)$ is equal to $\mathbf{y}^{\mathsf{e}}(i)$, and $\mathbf{y}^{\mathsf{a}}(t)$ is equal to $\mathbf{y}^{\mathsf{e}}(i)$. I.e., as input for W^{o} only the ith *encoder* state, $\mathbf{y}^{\mathsf{e}}(i)$, is used, say, $W^{\mathsf{o}}(\mathbf{y}^{\mathsf{e}}(i), \mathbf{y}^{\mathsf{e}}(i))$ is passed directly to position t of the output sequence.

Remarks (parameterized attention) :

- $\mathbf{y}^{\mathsf{d}}(t)$ is also denoted as “query,” while the sequence of $\mathbf{y}^{\mathsf{e}}(i)$ are denoted as “keys” in this setting. Since $\mathbf{a}(t)$ is normalized with a softmax operation it represents the importance of the T^{e} time steps as probabilities.
- *Parameterized* attention introduces three weight matrices, W^{Q}, W^{K}, and W^{V}, in order to learn a more sophisticated version of the simple attention vector $\mathbf{y}^{\mathsf{a}}(t)$. In this regard the matrices are called “query projection”, “key projection”, and “value projection” [matrix] respectively.
- Inspired by nature, the structure of the model function $\mathbf{y}()$ has been developed in the form of a network that connects the matrices W^{h}, W^{o}, W^{Q}, W^{K}, and W^{V} in a particular manner. Note that the shown model function is specified completely by the set of parameters $\mathbf{w}$, which are organized in the aforementioned matrices.