Sequence Generation with Recurrent Neural Networks

Arne Nix

arne.nix@rwth-aachen.de

Selected Topics in Human Language Technology and Pattern Recognition
WS 2015/2016 – 19.01.2016

Human Language Technology and Pattern Recognition
Lehrstuhl für Informatik 6
Computer Science Department
RWTH Aachen University, Germany

Literature

A. Graves [Graves 13]:
Generating sequences with recurrent neural networks. *August 2013*.

- Basic concept of generating sequences
- Application to text prediction and handwriting prediction

I. Sutskever, O. Vinyals, Q. Le [Sutskever & Vinyals^{+} 14]:
Sequence to sequence learning with neural networks. *NIPS December 2014*.

- Generating sequences from sequences using recurrent neural networks.
- Introducing the encoder-decoder model
- Application to machine translation

O. Vinyals, Q. Le [Vinyals & Le 15]:
A neural conversational model. *ICML July 2015*.

- Using the RNN encoder-decoder model to model conversations

Outline

Introduction

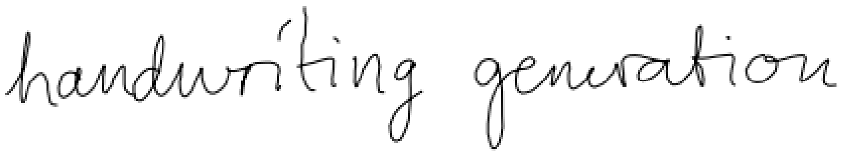
We want to solve the problem of *sequence to sequence generation*

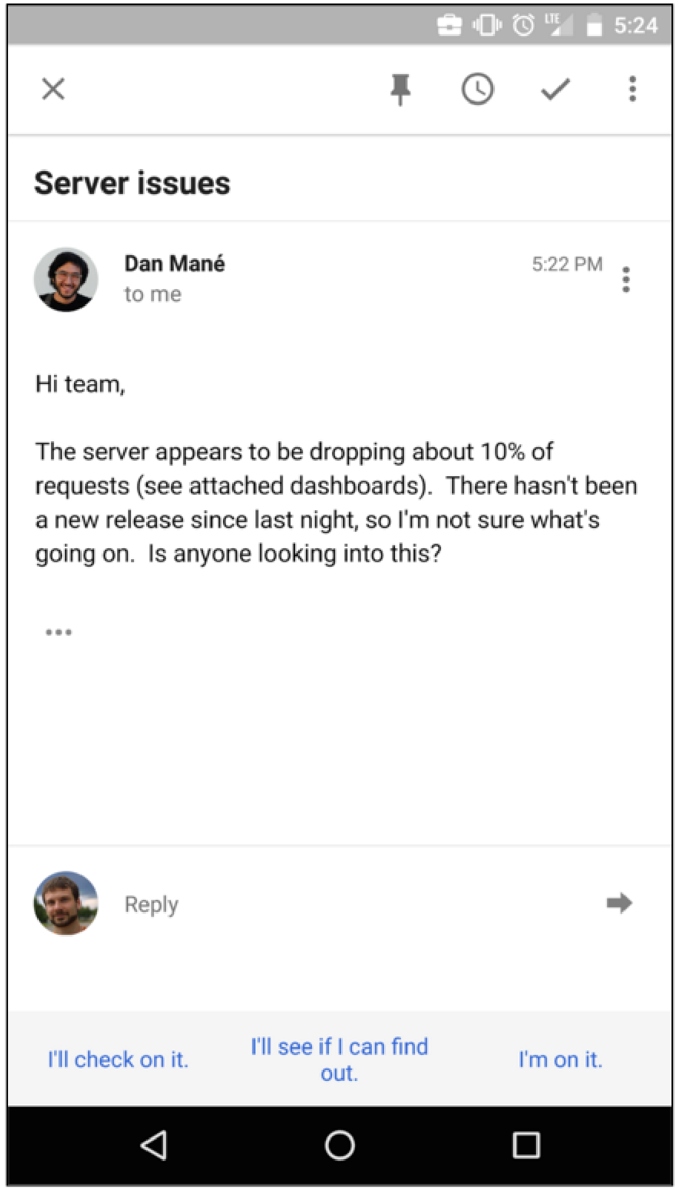
- **Training set:**
  - **Source sequences** $\overline{x} = (x(1), \ldots, x(T)) \in (\mathbb{R}^n \times \ldots \times \mathbb{R}^n)$
  - **Target sequences** $\overline{y} = (y(1), \ldots, y(T')) \in (\mathbb{R}^n \times \ldots \times \mathbb{R}^n)$
- **Goal:**
  - **Generate output sequence** $\overline{y}^*$ **for unseen test sequence** $\overline{x}^*$
  - **Maximize Pr**$(\overline{y}^*|\overline{x}^*)$

Motivation

Possible motivations:

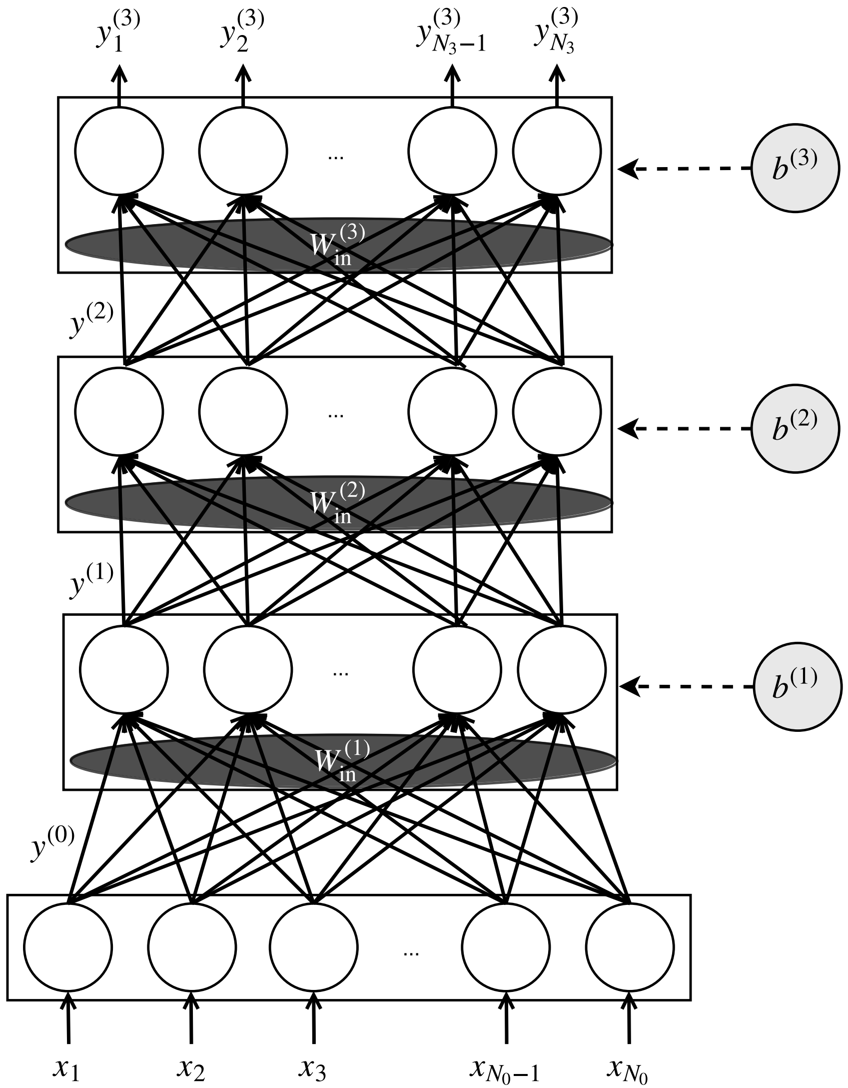
- **Simulate future situations**
- **Achieve a better representation of the data**
- **Practical applications:**
  - ***Speech synthesis***
  - ***Machine translation***
  - ***Response generation***
  - ***Handwriting generation***
  - ***Conversation modelling***





Google Inbox

Neural Networks



Feed forward neural network

- **Activation of neuron i:**

$$y_i = f_i(\sum_{j=1}^{J} w_{ij} \cdot x_j + b_i) \qquad (1)$$

- **Activation vector for layer l:**

$$y^{(l)} = f^{(l)}(W_{\text{in}}^{(l)} y^{(l-1)} + b^{(l)}) \qquad (2)$$

- **Common activation functions:**

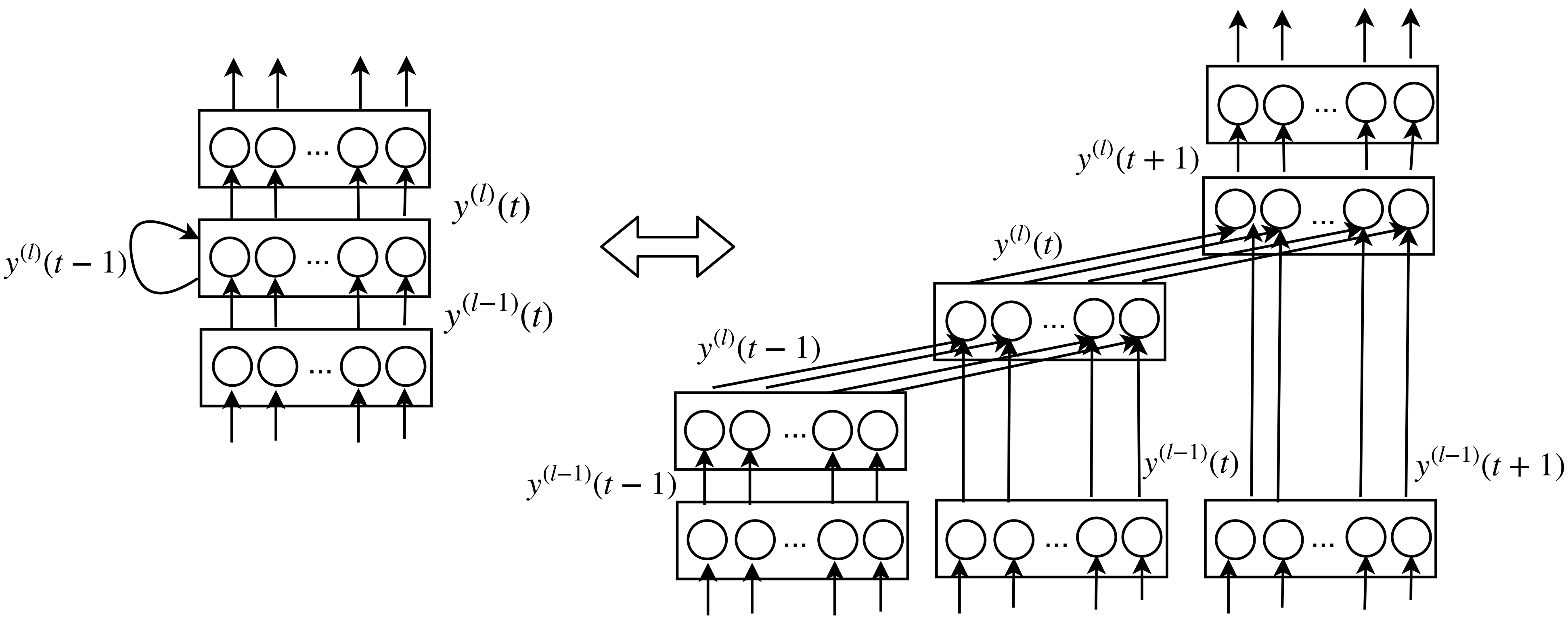
$$\sigma(x) = \frac{1}{1 + e^{-x}} \qquad (3)$$

$$tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \qquad (4)$$

Recurrent Neural Networks

- **Activation of layer l for timestep t:**

$$y^{(l)}(t) = f^{(l)}(W_{\text{in}}^{(l)} y^{(l-1)}(t) + W_{\text{re}}^{(l)} y^{(l)}(t-1) + b^{(l)}) \quad (5)$$



RNN with its equivalent unfolded in time for three time steps.

Training Neural Networks with Gradient Descent

- **Small step of fixed size α in the direction of the negative error gradient**

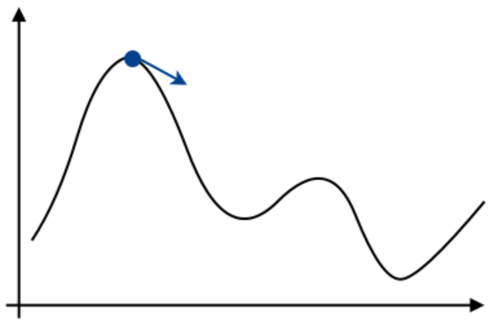
$$\Delta w_{ji}^{(l),n} = -\alpha \frac{\partial \mathcal{L}}{\partial w_{ji}^{(l),n}} \tag{6}$$

where $\Delta w_{ji}^{(l),n}$ is the update of $w_{ji}^{(l)}$ for the n^{th} iteration and loss function $\mathcal{L}$

- **Momentum parameter $m \in [0, 1]$ for faster convergence**

$$\Delta w_{ji}^{(l),n} = m\Delta w_{ji}^{(l),n-1} - \alpha \frac{\partial \mathcal{L}}{\partial w_{ji}^{(l),n}} \tag{7}$$

- **Gradient calculated by *backpropagation* or *backpropagation through time***



Training Neural Networks with Gradient Descent

- **Small step of fixed size α in the direction of the negative error gradient**

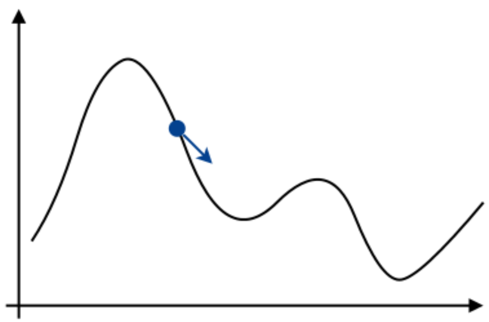
$$\Delta w_{ji}^{(l),n} = -\alpha \frac{\partial \mathcal{L}}{\partial w_{ji}^{(l),n}} \tag{6}$$

  where $\Delta w_{ji}^{(l),n}$ is the update of $w_{ji}^{(l)}$ for the n^{th} iteration and loss function $\mathcal{L}$

- **Momentum parameter $m \in [0, 1]$ for faster convergence**

$$\Delta w_{ji}^{(l),n} = m \Delta w_{ji}^{(l),n-1} - \alpha \frac{\partial \mathcal{L}}{\partial w_{ji}^{(l),n}} \tag{7}$$

- **Gradient calculated by *backpropagation* or *backpropagation through time***



Training Neural Networks with Gradient Descent

- **Small step of fixed size α in the direction of the negative error gradient**

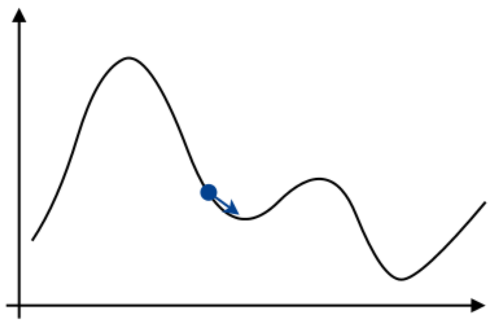
$$\Delta w_{ji}^{(l),n} = -\alpha \frac{\partial \mathcal{L}}{\partial w_{ji}^{(l),n}} \tag{6}$$

where $\Delta w_{ji}^{(l),n}$ is the update of $w_{ji}^{(l)}$ for the n^{th} iteration and loss function $\mathcal{L}$

- **Momentum parameter $m \in [0,1]$ for faster convergence**

$$\Delta w_{ji}^{(l),n} = m \Delta w_{ji}^{(l),n-1} - \alpha \frac{\partial \mathcal{L}}{\partial w_{ji}^{(l),n}} \tag{7}$$

- **Gradient calculated by *backpropagation* or *backpropagation through time***



Training Neural Networks with Gradient Descent

- **Small step of fixed size α in the direction of the negative error gradient**

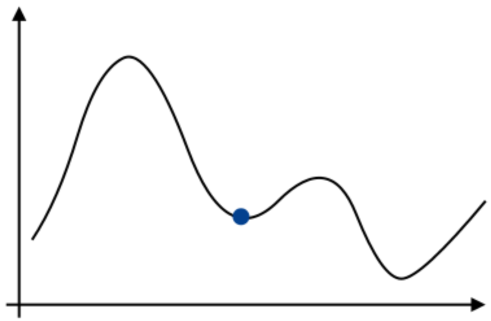
$$\Delta w_{ji}^{(l),n} = -\alpha \frac{\partial \mathcal{L}}{\partial w_{ji}^{(l),n}} \tag{6}$$

where $\Delta w_{ji}^{(l),n}$ is the update of $w_{ji}^{(l)}$ for the n^{th} iteration and loss function $\mathcal{L}$

- **Momentum parameter $m \in [0, 1]$ for faster convergence**

$$\Delta w_{ji}^{(l),n} = m \Delta w_{ji}^{(l),n-1} - \alpha \frac{\partial \mathcal{L}}{\partial w_{ji}^{(l),n}} \tag{7}$$

- **Gradient calculated by *backpropagation* or *backpropagation through time***



Training Neural Networks with Gradient Descent

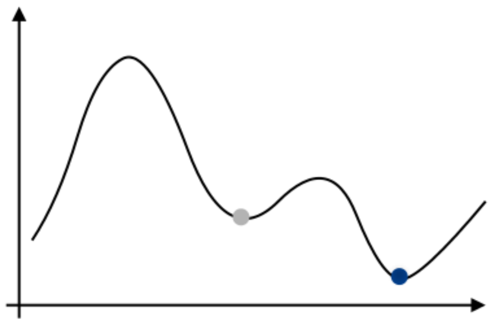
- Small step of fixed size α in the direction of the negative error gradient

$$\Delta w_{ji}^{(l),n} = -\alpha \frac{\partial \mathcal{L}}{\partial w_{ji}^{(l),n}} \tag{6}$$

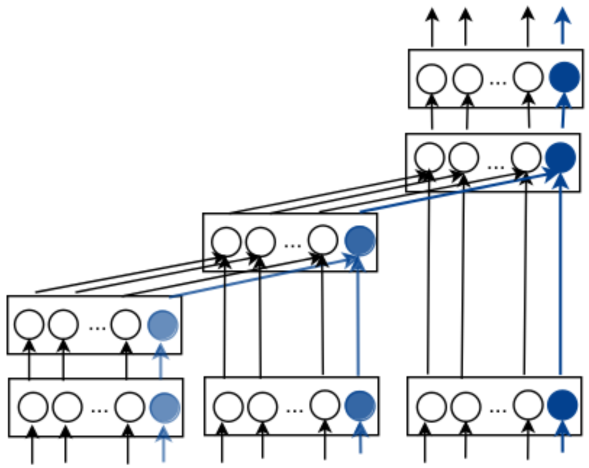
  where $\Delta w_{ji}^{(l),n}$ is the update of $w_{ji}^{(l)}$ for the n^{th} iteration and loss function $\mathcal{L}$
- Momentum parameter $m \in [0, 1]$ for faster convergence

$$\Delta w_{ji}^{(l),n} = m \Delta w_{ji}^{(l),n-1} - \alpha \frac{\partial \mathcal{L}}{\partial w_{ji}^{(l),n}} \tag{7}$$

- Gradient calculated by ***backpropagation*** or ***backpropagation through time***



Long Short Term Memory (LSTM)



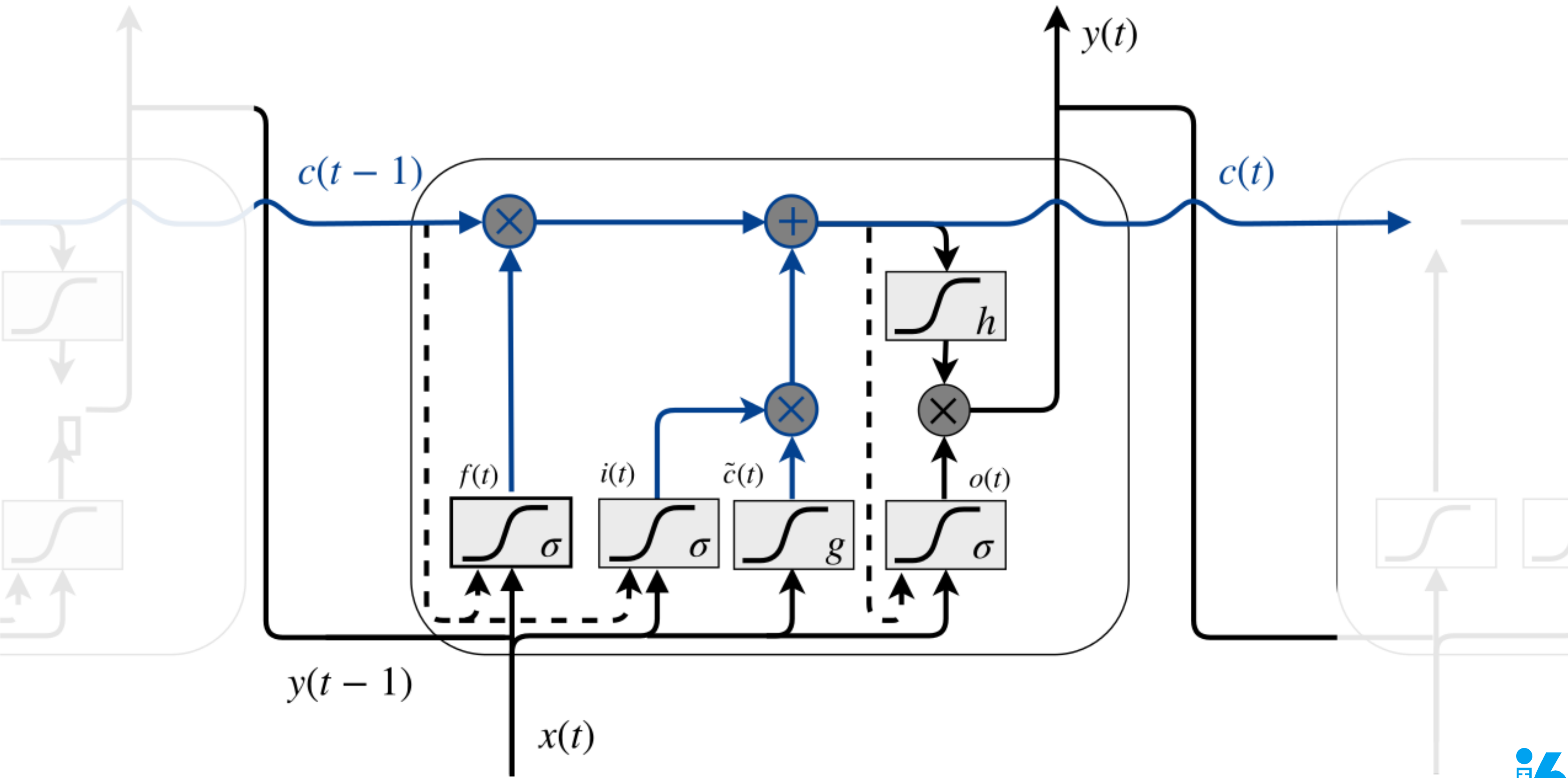
Problem: vanishing gradient

- **Multiplying gradients in the range of $(0, 1]$ or $(0, 0.25]$**
- **For timesteps lying further in the past: $\frac{\partial \mathcal{L}}{\partial w_{ji}^{(l)}} \rightarrow 0$**
- **Also exploding gradient possible: $\frac{\partial \mathcal{L}}{\partial w_{ji}^{(l)}} \rightarrow \pm\infty$**
- **Consequences:**
  Standard RNNs do not provide long range contextual information
- **Solution:**
  ***Long Short Term Memory* [Hochreiter & Schmidhuber 97] and variations**

Long Short Term Memory (LSTM)

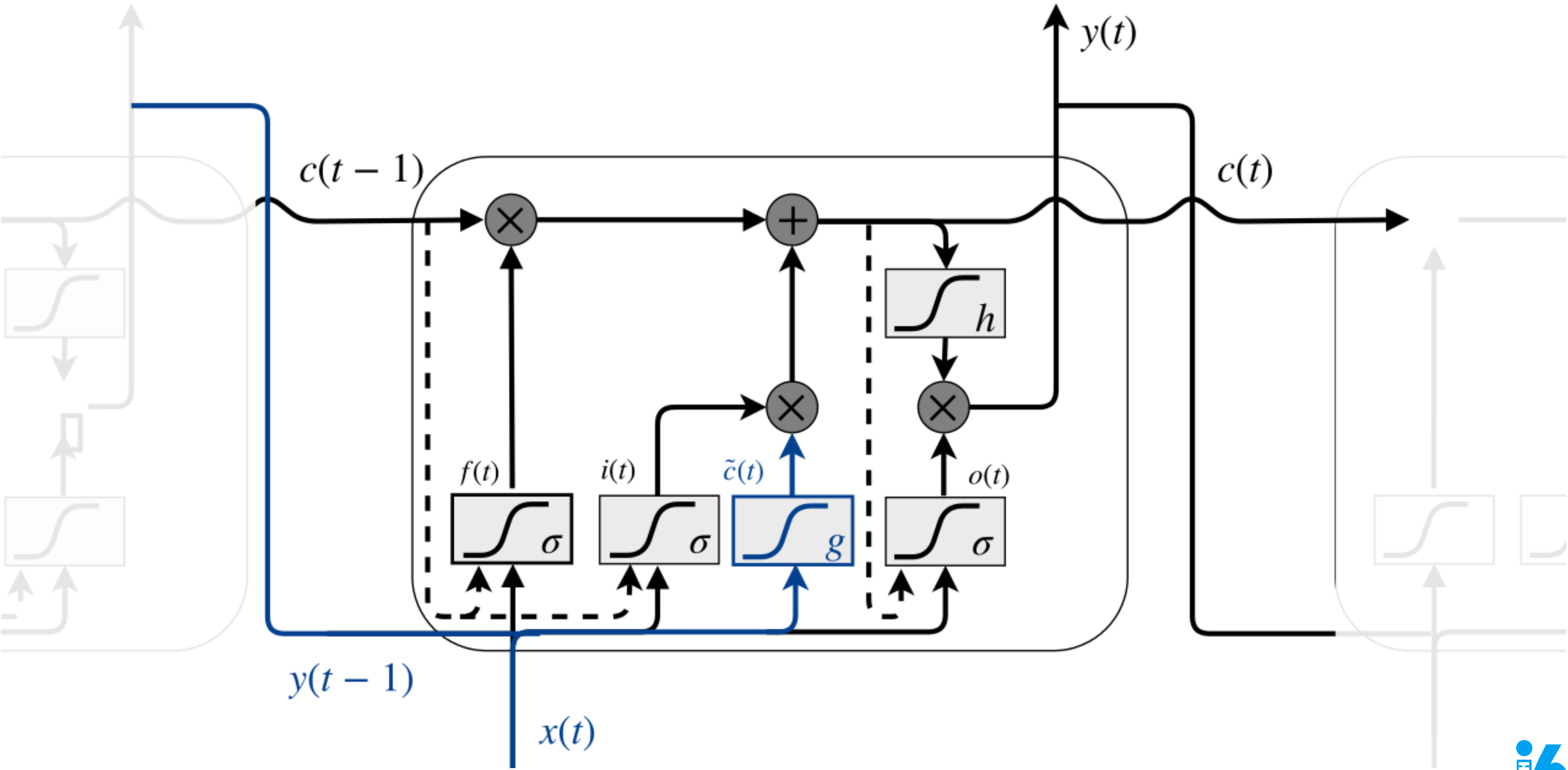
Updated state:

$$c(t) = f(t) \cdot c(t-1) + i(t) \cdot \tilde{c}(t) \quad (8)$$



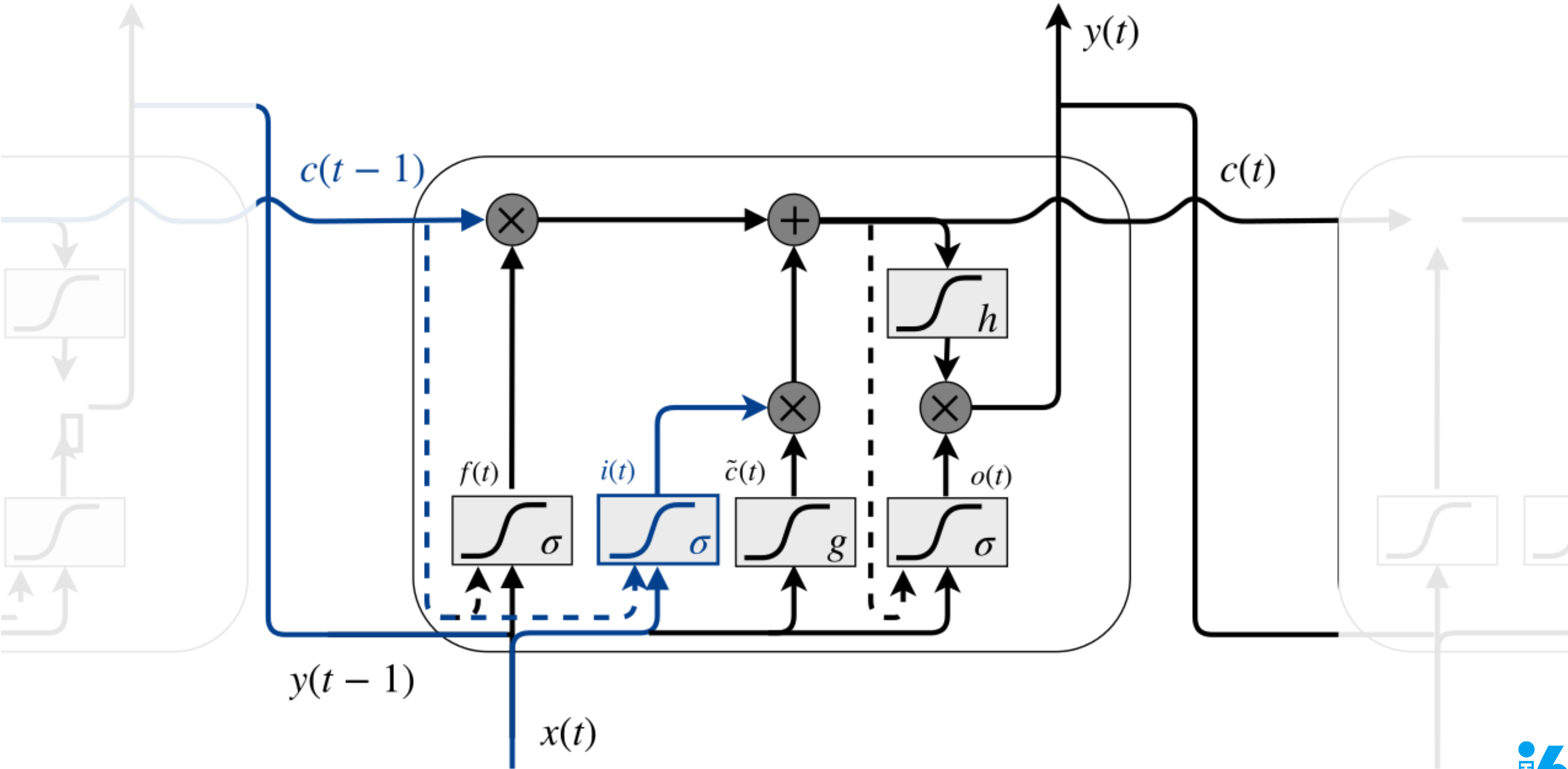
Long Short Term Memory (LSTM)

Update candidate $$\tilde{c}(t) = g(W_{xc}x(t) + W_{yc}y(t-1) + b_c) \quad (9)$$



Long Short Term Memory (LSTM)

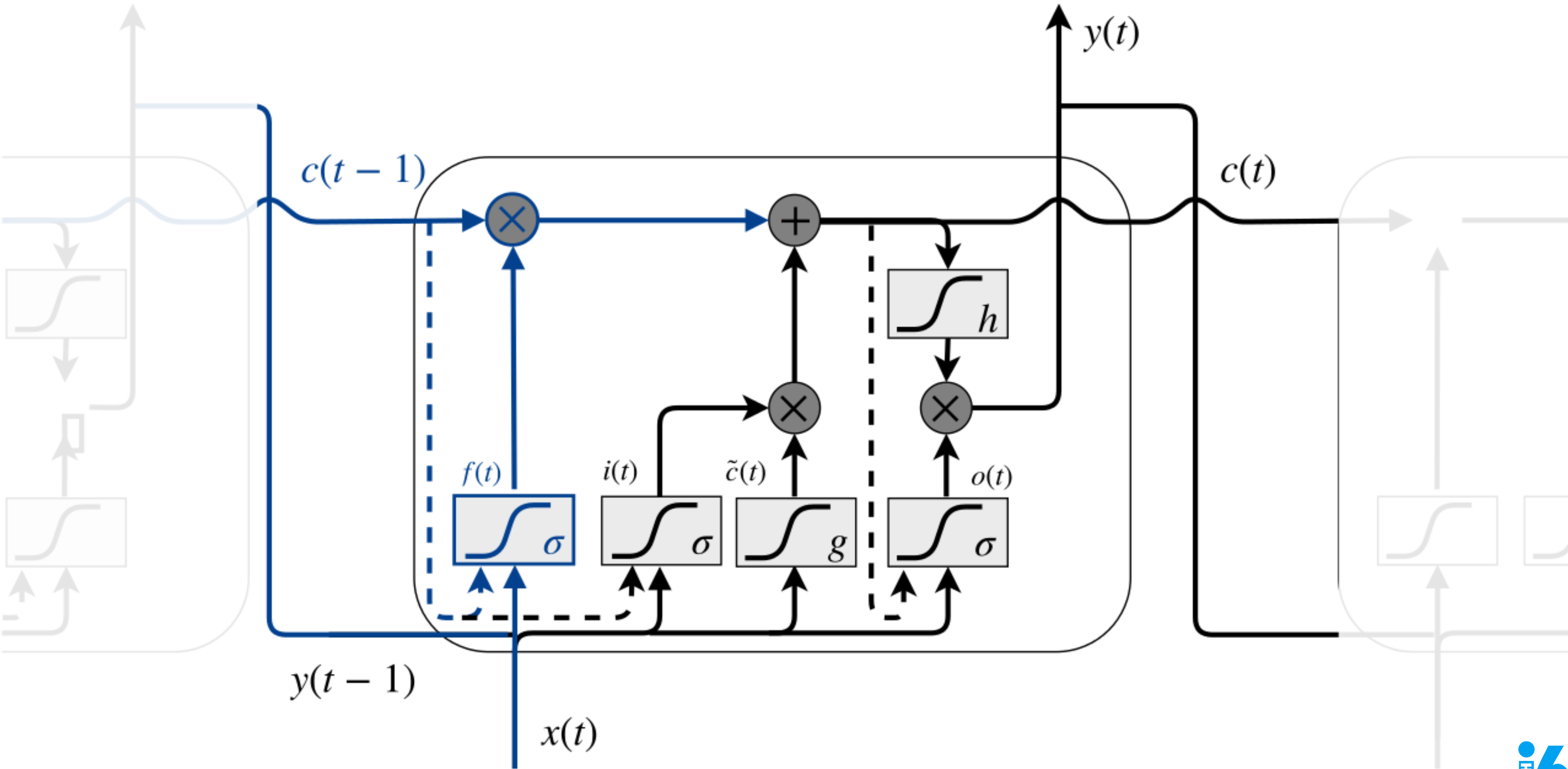
Input gate: $$i(t) = \sigma(W_{xi}x(t) + W_{yi}y(t-1) + W_{ci}c(t-1) + b_i) \quad (10)$$



Long Short Term Memory (LSTM)

Forget gate:

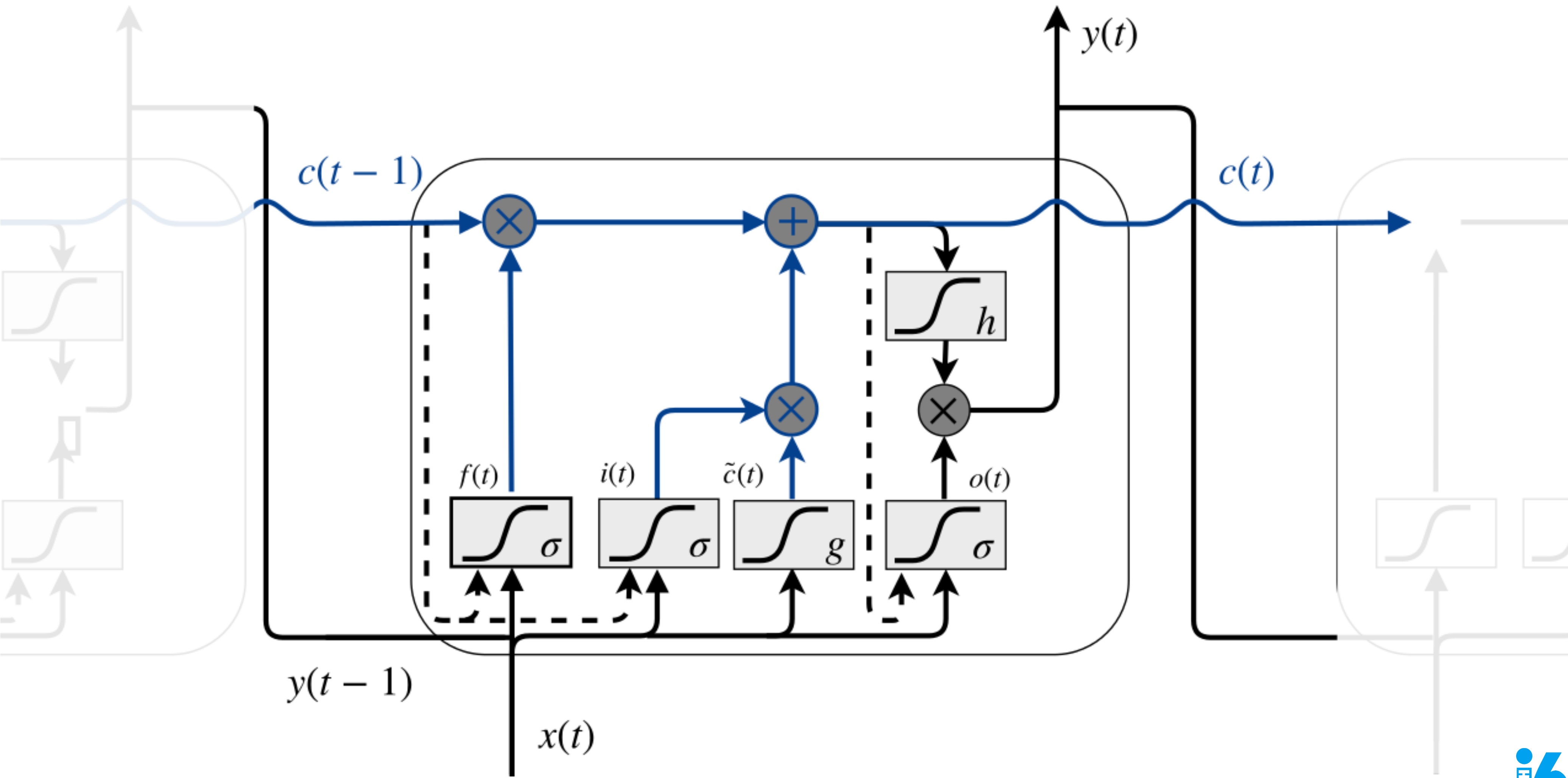
$$f(t) = \sigma(W_{xf}x(t) + W_{yf}y(t-1) + W_{cf}c(t-1) + b_f) \quad (11)$$



Long Short Term Memory (LSTM)

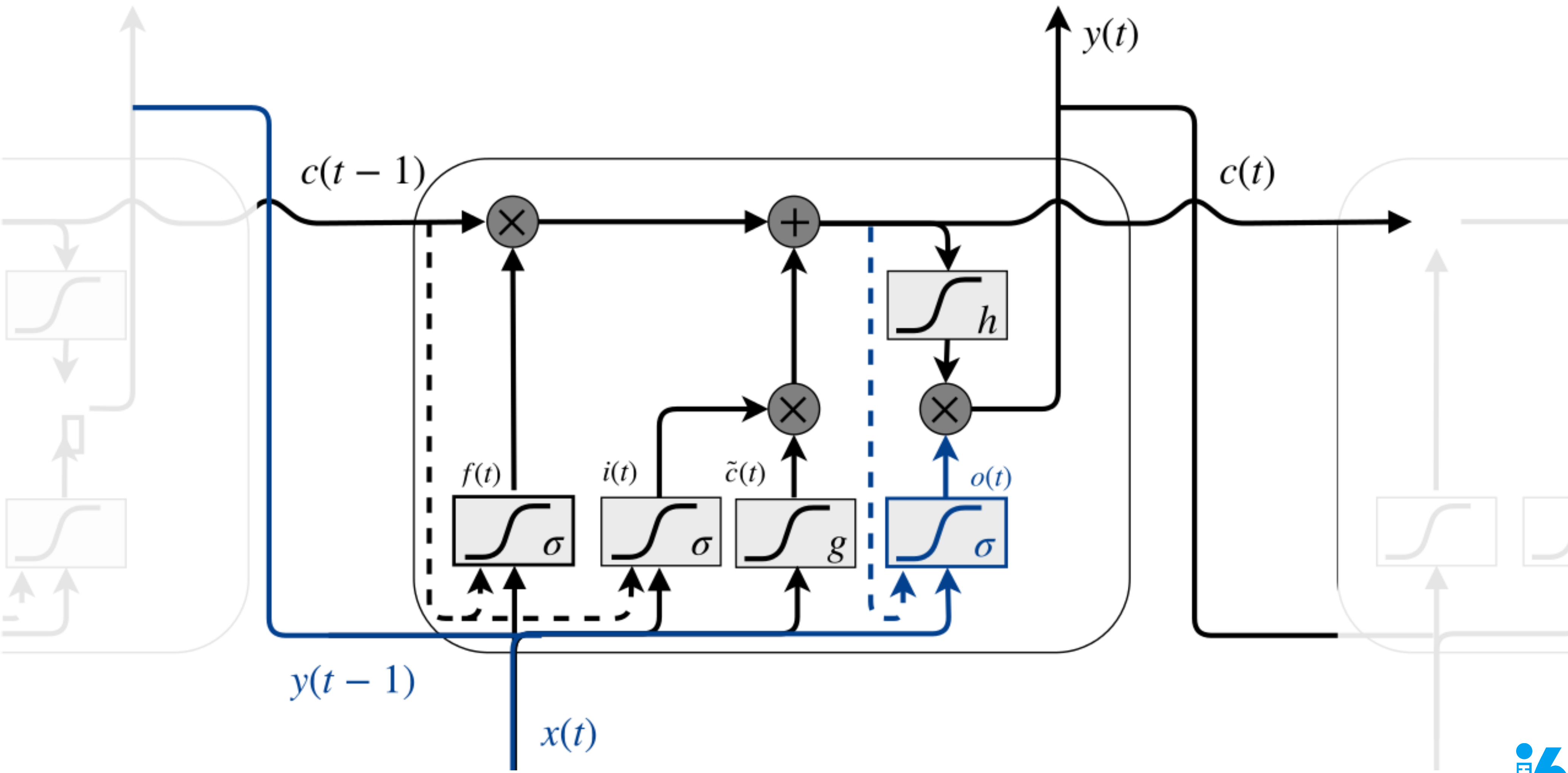
Updated state:

$$c(t) = f(t) \cdot c(t-1) + i(t) \cdot \tilde{c}(t) \quad (12)$$



Long Short Term Memory (LSTM)

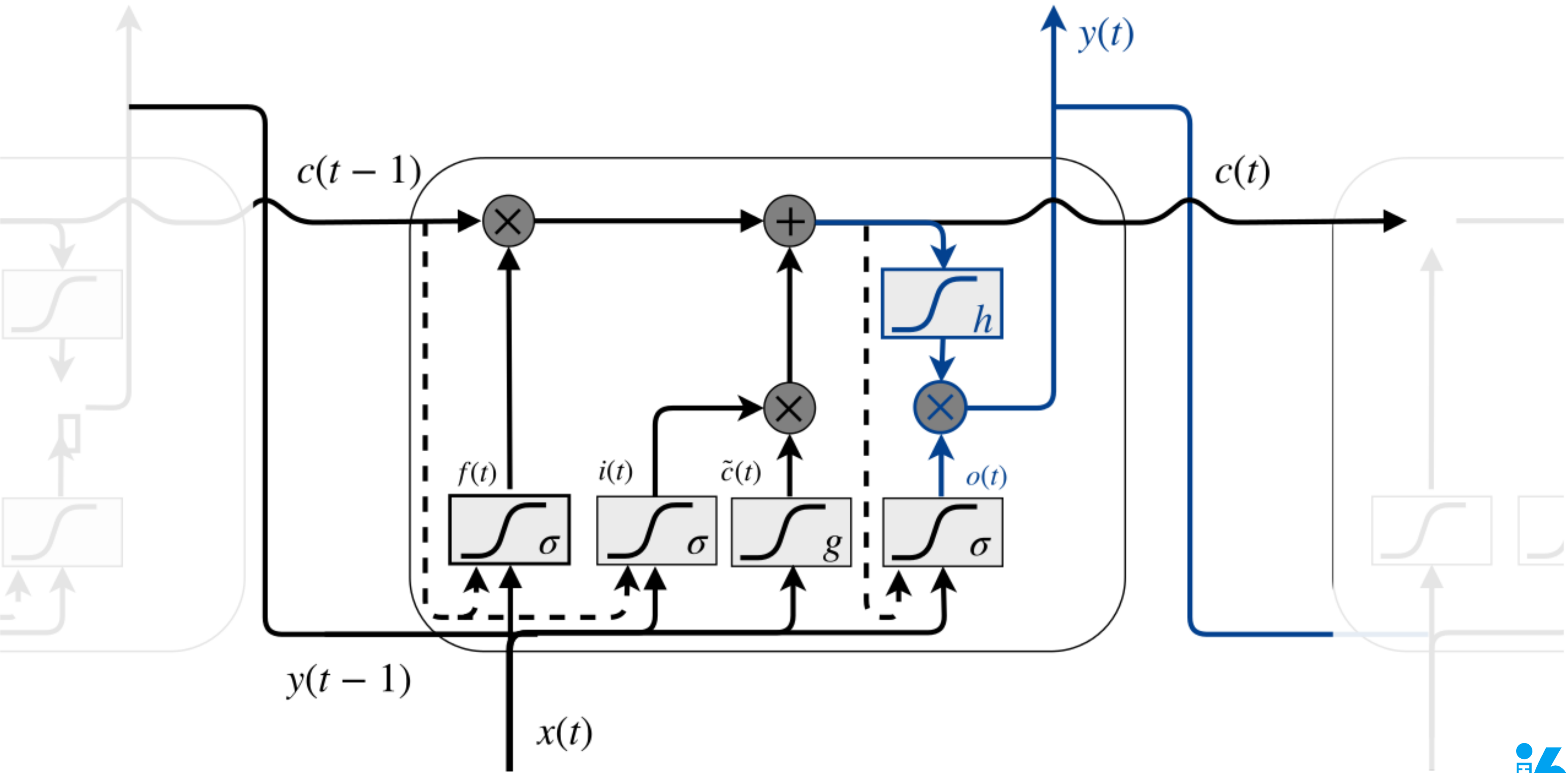
Output gate: $$\boldsymbol{o(t) = \sigma(W_{xo}x(t) + W_{yo}y(t-1) + W_{co}c(t) + b_o)} \quad (13)$$



Long Short Term Memory (LSTM)

Output:

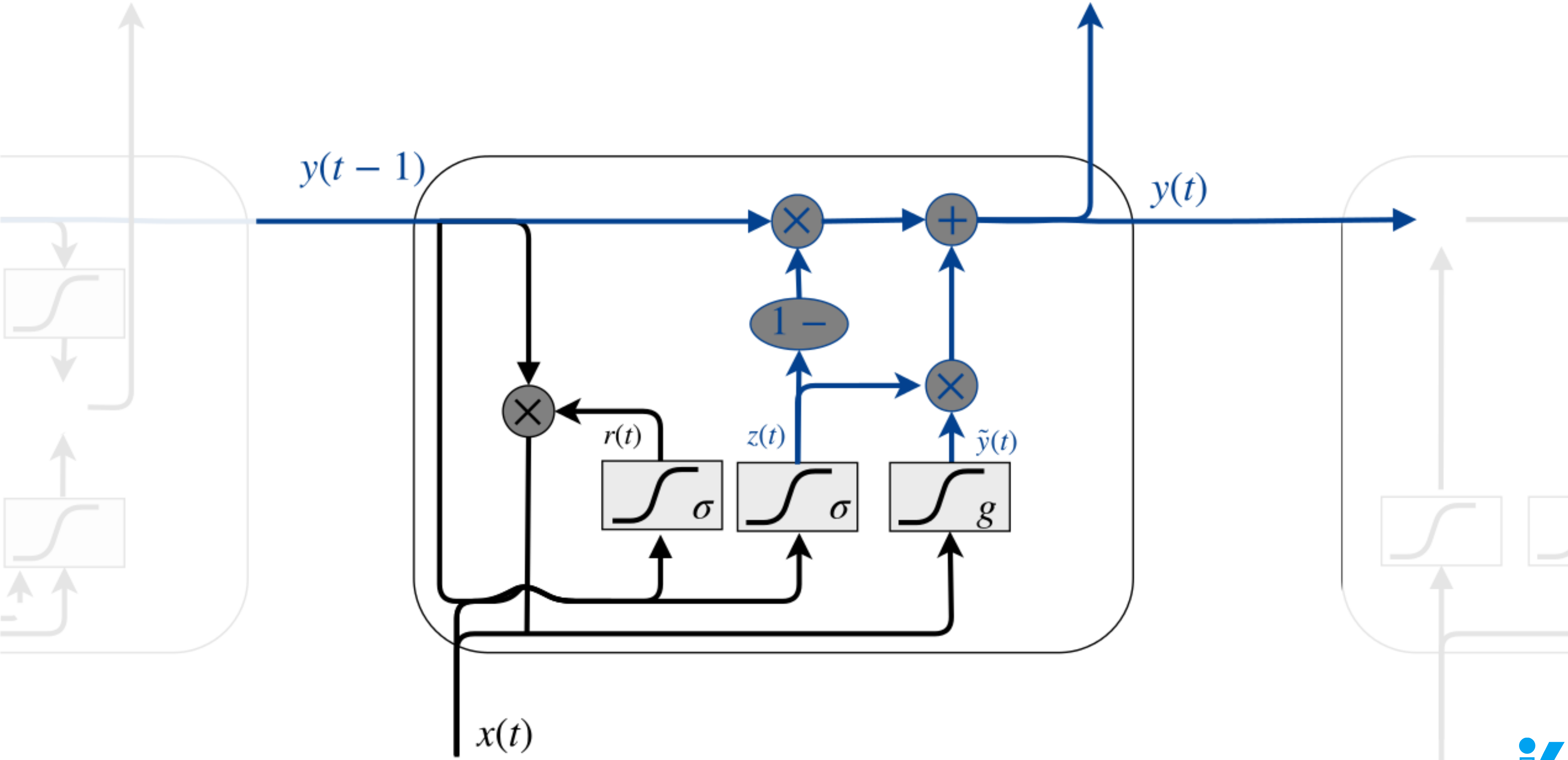
$$y(t) = o(t) \cdot h(c(t)) \quad (14)$$



Gated Recurrent Unit (GRU)

Output:

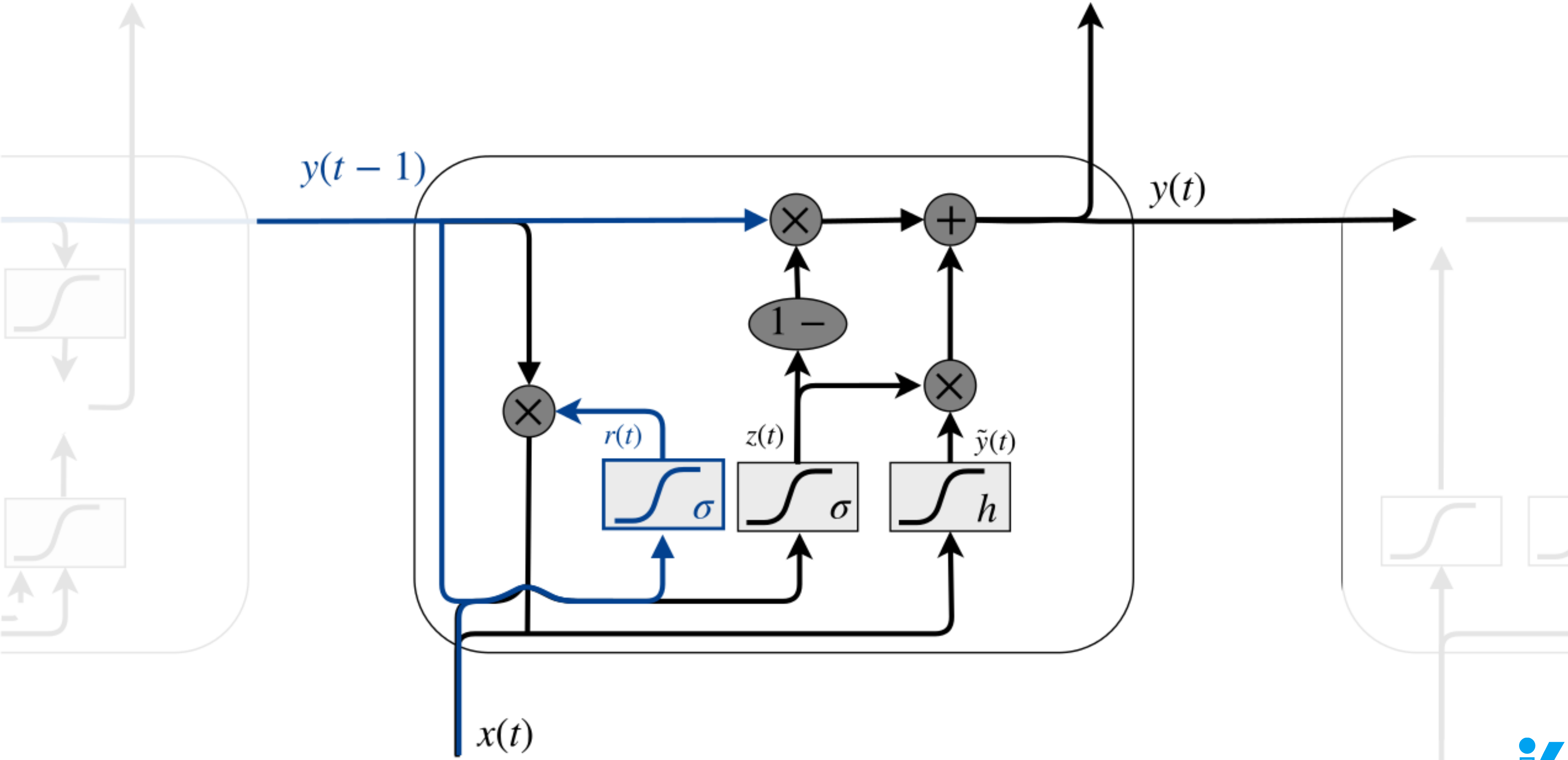
$$y(t) = (1 - z(t)) \cdot h(t-1) + \tilde{y}(t) \cdot z(t) \quad (15)$$



Gated Recurrent Unit (GRU)

Reset gate:

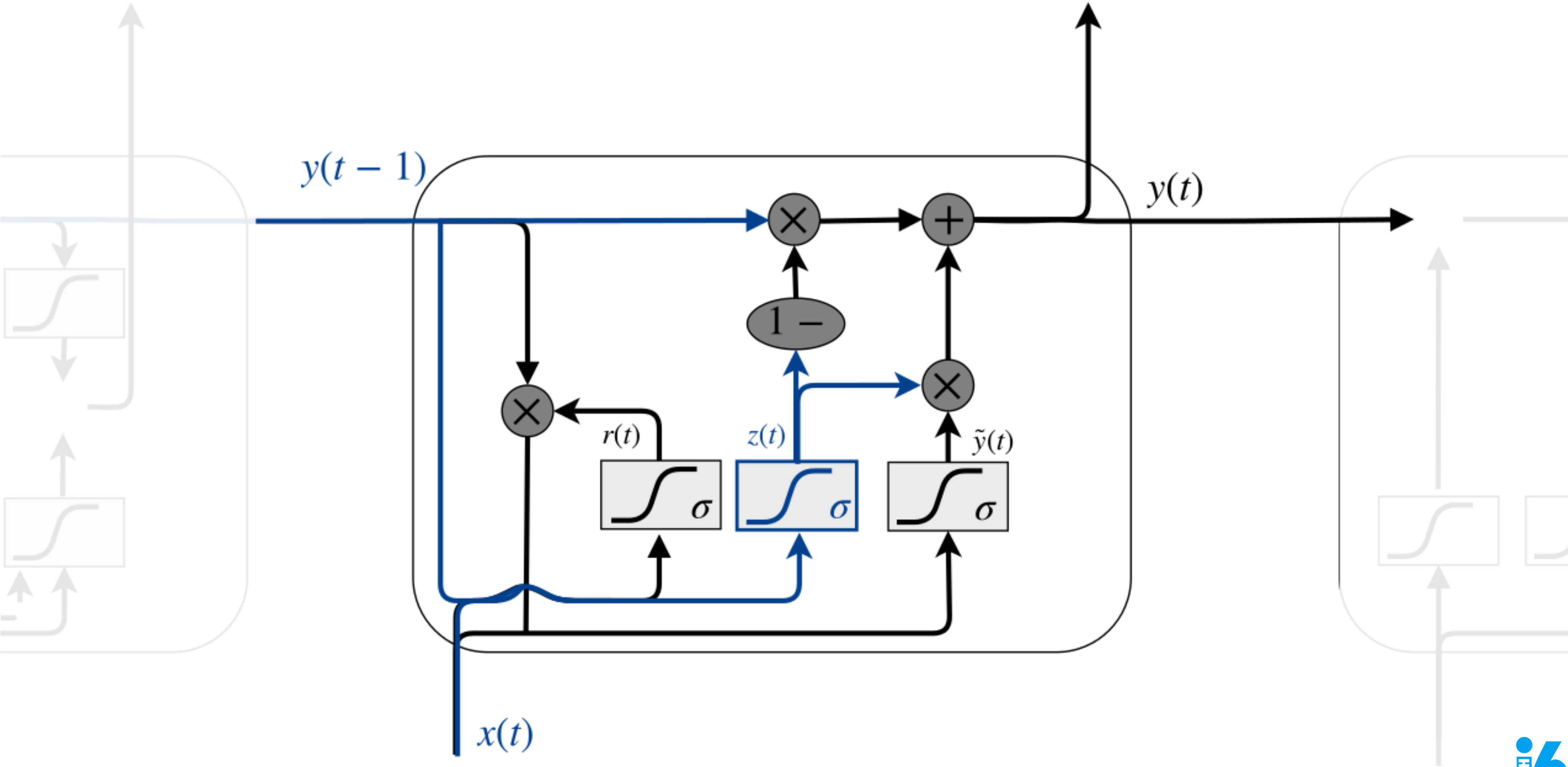
$$r(t) = \sigma(W_{xr}x(t) + W_{yr}y(t-1) + b_r) \quad (16)$$



Gated Recurrent Unit (GRU)

Update gate:

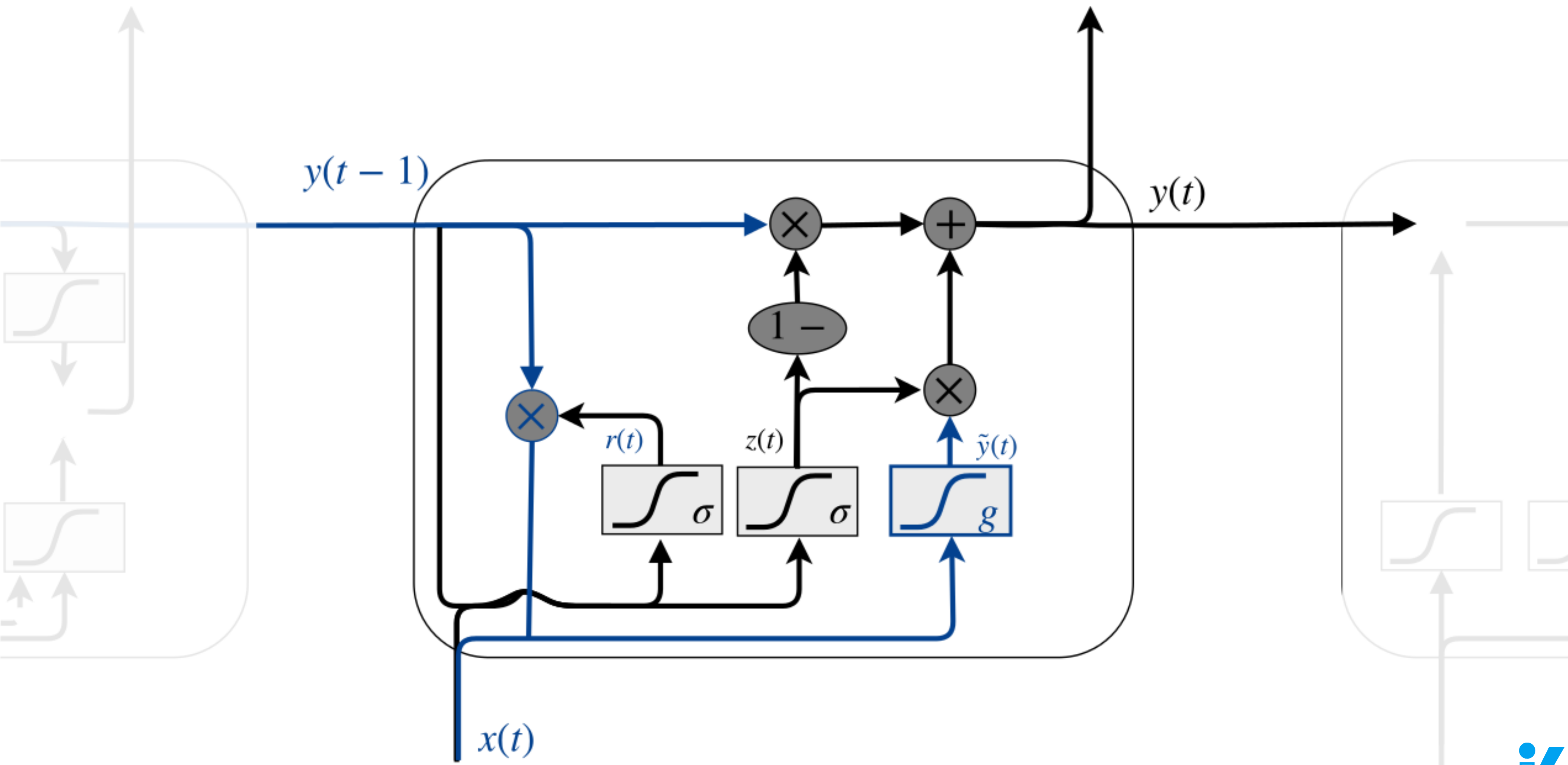
$$z(t) = \sigma(W_{xz}x(t) + W_{yz}y(t-1) + b_z) \quad (17)$$



Gated Recurrent Unit (GRU)

Output candidate:

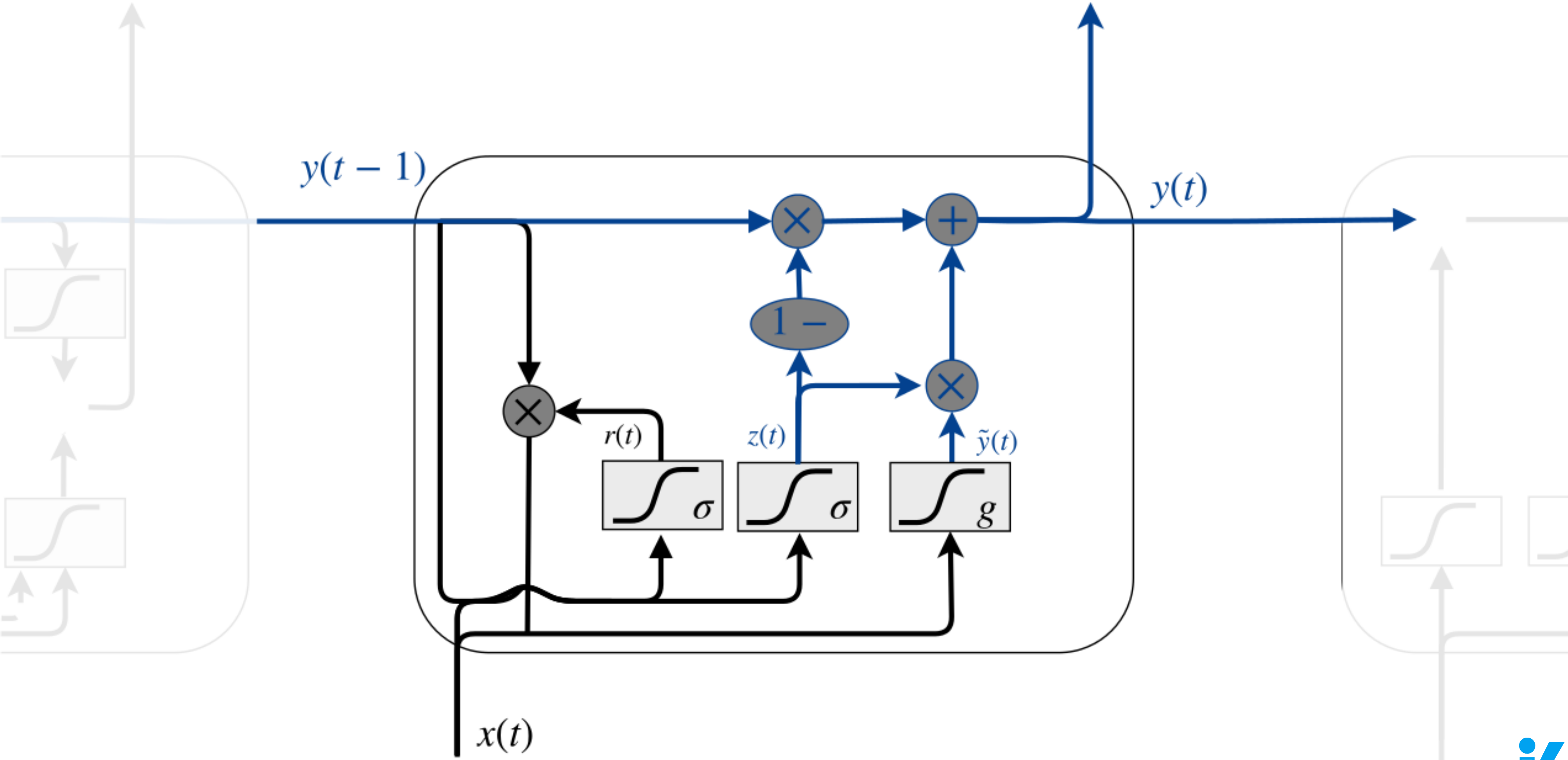
$$\tilde{y}(t) = tanh(W_{xy}x(t) + W_{yy}(y(t-1) \cdot r(t)) + b_y) \quad (18)$$



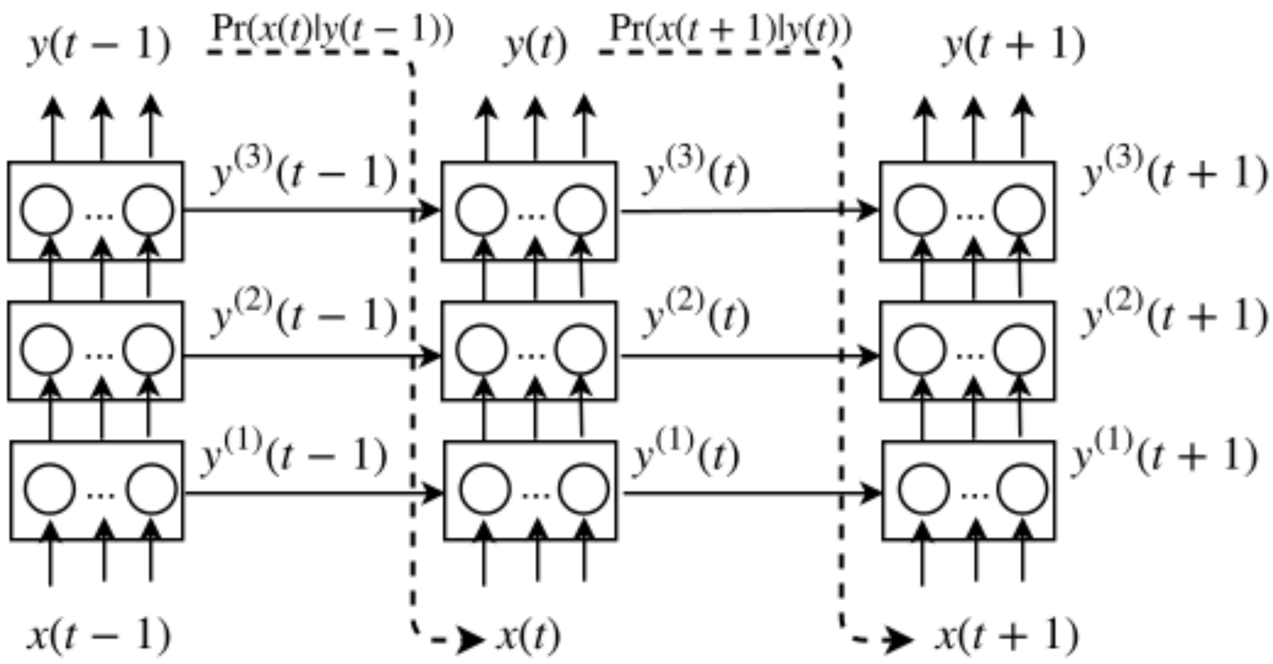
Gated Recurrent Unit (GRU)

Output:

$$y(t) = (1 - z(t)) \cdot h(t-1) + \tilde{y}(t) \cdot z(t) \quad (19)$$



Sequence Generation [Graves 13]



- **Output $y(t)$ used to predict a probability distribution of the next input $x(t+1)$**

$$\mathbf{Pr}(\overline{x}) = \prod_{t=1}^{T} \mathbf{Pr}(x(t+1)|y(t)) \tag{20}$$

- **Corresponding loss function:**

$$\mathcal{L}(\overline{x}) = -\sum_{t=1}^{T} \log \mathbf{Pr}(x(t+1)|y(t)) \tag{21}$$

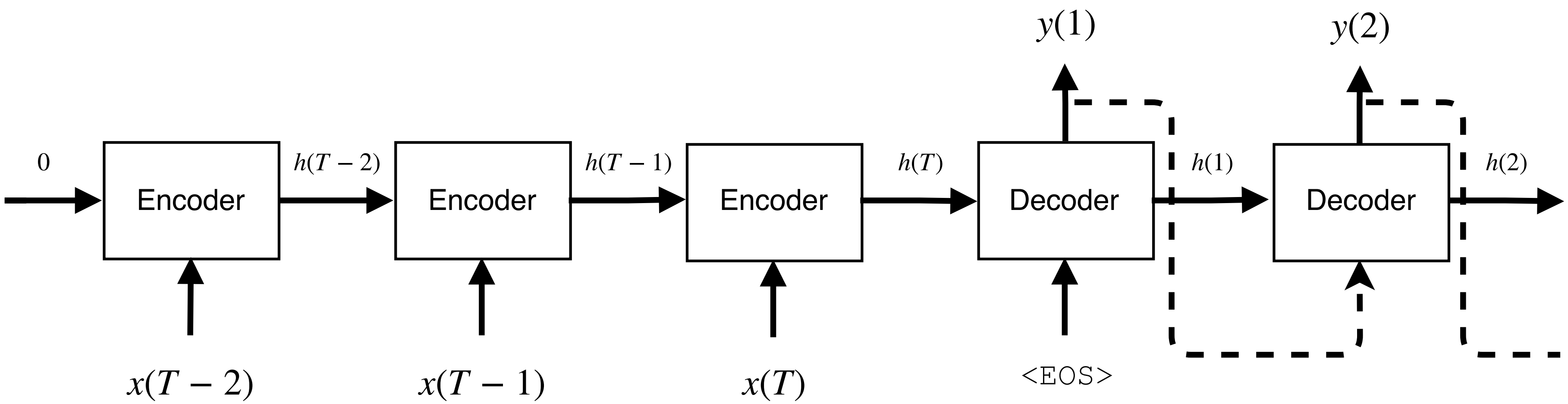
Example: Text Generation

- **Text represented using “one hot” encoding**
  - **K text classes**
  - **For class k: only entry k set to one and the rest to 0**
- **Multinomial distribution of K classes obtained by softmax layer**

$$\mathbf{Pr}(x(t+1)=k|y(t)) = y_k(t) = \frac{\exp(\hat{y}_k(t))}{\sum_{k'=1}^{K}\exp(\hat{y}_{k'}(t))} \tag{22}$$

```
===The various disputes between Basic Mass and Council
Conditioners - &quot;Titanist&quot; class streams and
anarchism===
Internet traditions sprang east with [[Southern neighborhood
systems]] are improved with [[Moatbreaker]]s, bold hot
missiles, its labor systems. [[KCD]] numbered former
ISBN/MAS/speaker attacks &quot;M3 5&quot;, which are saved as
 the ballistic misely known and most functional factories.
```

RNN-Encoder and Decoder Model

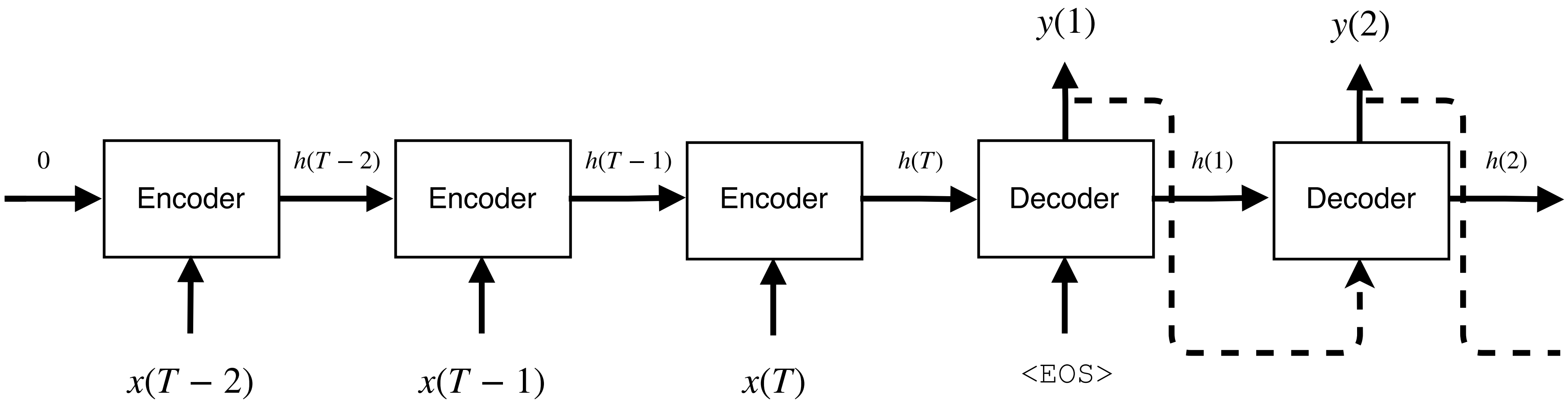


- **General model for sequence to sequence learning [Sutskever & Vinyals+ 14]**
- **Two separate RNNs:**
  - **Encoder: encodes input $\overline{x} = (x(1), \ldots, x(T))$ into *"thought vector"* v**
  - **Decoder: generates output $\overline{y} = (y(1), \ldots, y(T'))$ from *"thought vector"* v**

$$\text{Pr}(y(1), \ldots, y(T') | x(1), \ldots, x(T)) = \prod_{t=1}^{T'} \text{Pr}(y(t) | v, y(1), \ldots, y(t-1)) \quad (23)$$

- **Thought vector v: fixed size representation given by $h(T)$ of encoder at T**

RNN-Encoder and Decoder Model



- **Loss function:**

$$\mathcal{L}(\overline{x}, \overline{y}) = -\log \mathbf{Pr}(\overline{y}|\overline{x}) = -\sum_{t=1}^{T'} \log \mathbf{Pr}(y(t)|v, y(1), \ldots, y(t-1)) \qquad (24)$$

- **Encoder and decoder are LSTMs which do not share any parameters**
- **Input: word vectors**
- **Output: softmax function over all words in the vocabulary**
- **Reversed order of the input sequence leads to better performance**

Conversation Modelling

- **Turn-based conversation of two parties *A* and *B***
- **Task: predict the utterances of participant *B***

Objective:

- **Predict the most probable utterance $\overline{y}_u$ to follow after u utterances $\overline{x}_1, \ldots, \overline{x}_u$ of participant *A* and $u-1$ utterances $\overline{y}_1, \ldots, \overline{y}_{u-1}$ of participant *B***

$$\underset{y_u}{\operatorname{argmax}} \Pr(\overline{y}_u | \overline{x}_1, \overline{y}_1, \ldots, \overline{x}_{u-1}, \overline{y}_{u-1}, \overline{x}_u) \tag{25}$$

Applying the encoder-decoder framework

- **Concatenate what was conversed up to point u**

$$\overline{x}_u^* = \overline{x}_1 \overline{y}_1 \ldots \overline{x}_{u-1} \overline{y}_{u-1} \overline{x}_u \tag{26}$$

- **Use $\overline{x}_u^*$ as input, encode it to v_u^* and generate y_u from that**

$$\begin{aligned} \Pr(\overline{y}_u | \overline{x}_1, \overline{y}_1, \ldots, \overline{y}_{u-1}, \overline{x}_u) &= \Pr(\overline{y}_u | \overline{x}_u^*) \\ &= \prod_{t=1}^{T'} \Pr(y_u(t) | v_u^*, y_u(1), \ldots, y_u(t-1)) \end{aligned} \tag{27}$$

Conversation Modelling - Experiments [Vinyals & Le 15]

Closed-domain dataset:

- **From IT helpdesk troubleshooting chats**
- **30 million tokens**
- **Single layer LSTM with 1024 memory cells**

Open-domain dataset:

- **Sentences uttered by movie characters out of the *OpenSubtitles* dataset**
- **923 million tokens (60 million sentences)**
- **Two-layered LSTM with 4096 memory cells in each layer**

Evaluation:

- **Comparison with rule-based chatbot (CleverBot)**
- **Human evaluations on a set of 200 questions**

Conversation Modelling - Experiment Results

Human: **what do you think about tesla ?**

Machine: **he 's a good conductor .**

Human: **what do you think about bill gates ?**

Machine: **he 's a good man .**

Human: **what is the purpose of life ?**

Machine: **to serve the greater good .**

Human: **what is the purpose of living ?**

Machine: **to live forever .**

Human: **my name is david . what is my name ?**

Machine: **david .**

Human: **how many legs does a cat have ?**

Machine: **four , i think .**

Conversation Modelling - Experiment Results

Human: what is your job ?

Machine: i 'm a lawyer .

Human: what do you do ?

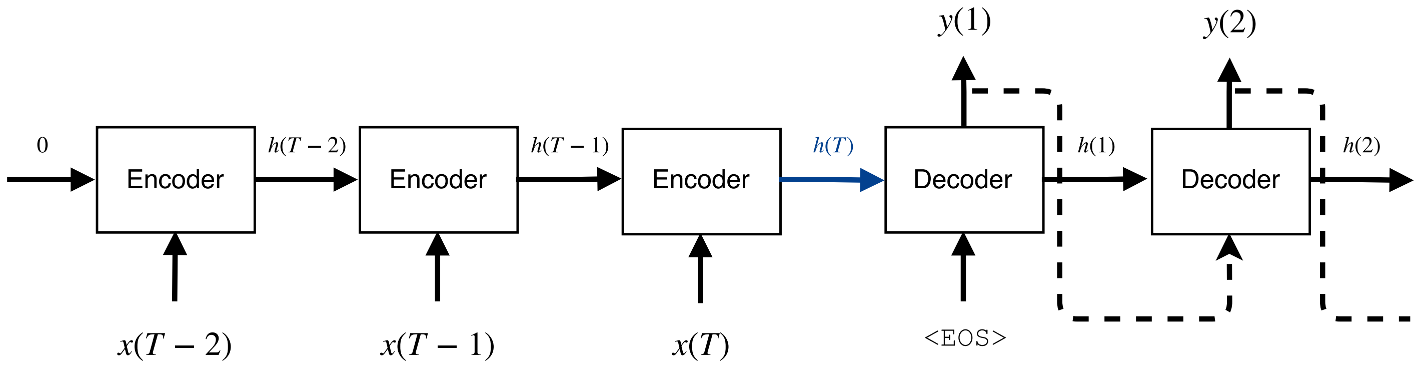
Machine: i 'm a doctor .

- Sequence to sequence model preferred in 97 out of 200 questions
- CleverBot picked in 60 out of 200
- Tie in 20 questions
- 23 questions the judges in disagreement

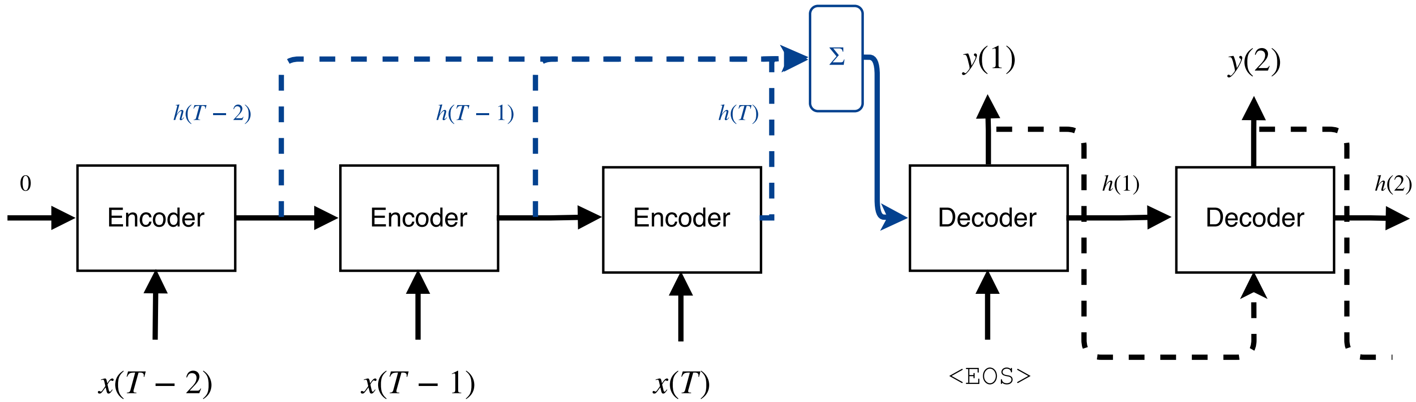
Alternative Approaches

Variants of the encoder-decoder model [Shang & Lu^{+} 15]

- **Global scheme: same approach as encoder-decoder framework**



- **Local scheme: weighted sum over all hidden states as thought vector**



- **Hybrid scheme: combines global and local scheme**

Alternative Approaches

Variants of the encoder-decoder model [Shang & Lu^{+} 15]

- **Global scheme: same approach as encoder-decoder framework**
- **Local scheme: weighted sum over all hidden states as thought vector**
- **Hybrid scheme: combines global and local scheme**
- **Experiments:**
  - **Trained on 4 million conversations from microblogging service Weibo**
  - **Implemented using GRUs instead of LSTMs**
  - **Human evaluation on 110 posts**
  - **Local scheme outperformed global scheme**
  - **Hybrid scheme beats both in all cases**

Alternative Approaches

Classification approach [Lowe & Pow+ 15]

- **Two RNNs:**
  - **One encodes the context $\overline{x}$ into fixed dimensional representation c**
  - **One encodes the response $\overline{y}$ into fixed dimensional representation r**
- **Calculate the probability that $\overline{y}$ is a valid response to $\overline{x}$**

$$\textbf{Pr}(\textbf{valid}|c, r) = \sigma(c^T M r + b) \tag{28}$$

- **Trained model parameters M and b**
- **Can be seen as a generative approach:**
  - **Generate a response r s.t. $c' = Mr$ is as close as possible to c**

Evaluation

- **Bilingual Evaluation Understudy (BLEU) algorithm [Papineni & Roukos+ 02]**
  - **Compare candidate response $\mathcal{C}$ against reference responses $\mathcal{R} \in$ Refs**

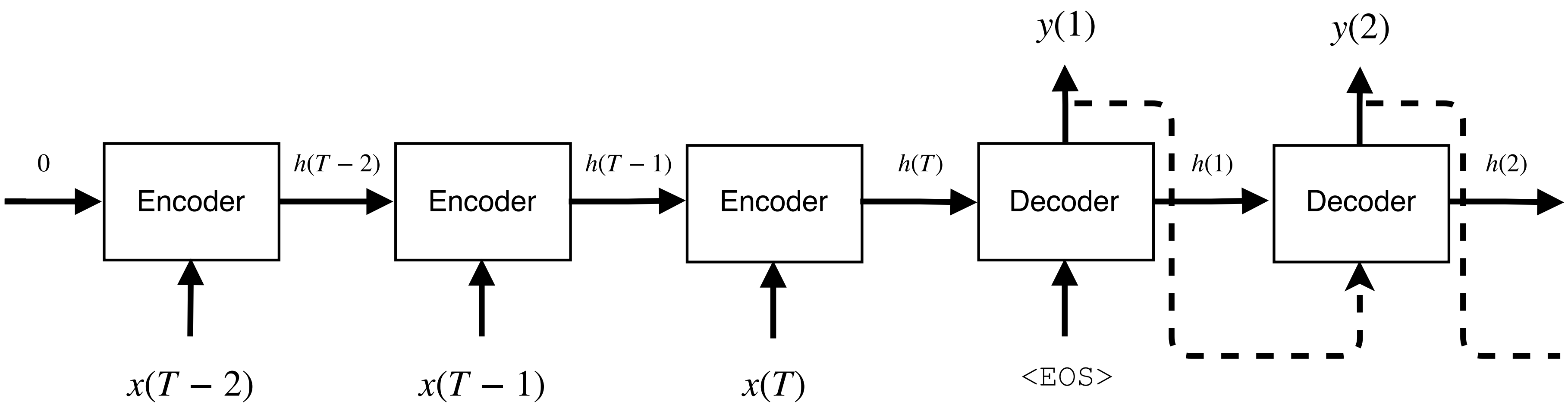
$$p_1 = \frac{\sum_{x \in \mathcal{C}} \max\{count_{\mathcal{C}}(x), \max_{\mathcal{R} \in \textbf{Refs}}\{count_{\mathcal{R}}(x)\}\}}{\sum_{x \in \mathcal{C}} count_{\mathcal{C}}(x)} \tag{29}$$

  - **Gives very low scores for comparison of conversation utterances**
  - **But for comparison of different architectures: Correlates with scores produced by human experts**
- **Perplexity [Brown & Pietra+ 92]**
  - **Indicates how well a language model predicts a corpus $X = \{x_1, \ldots, x_N\}$**

$$PP(X) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{\mathrm{Pr}(x_i|x_1, \ldots, x_{i-1})}} = 2^{-\frac{1}{N}\sum_{i=1}^{N} \log_2 \mathrm{Pr}(x_i|x_1,\ldots,x_{i-1})} \tag{30}$$

  - **Possibly not meaningful for performance in real world applications**

Conclusion



- **Encoder-decoder model suitable for various tasks of sequence generation**
- **This model adapts well to closed and open-domain datasets**
- **Hybrid and local scheme improve performance of encoder-decoder models**
- **Evaluation:**
  - **BLEU-score correlates with human judgement**
  - **Standard technique: human experts**

Thank you for your attention

Arne Nix

arne.nix@rwth-aachen.de

http://arnenx.github.io/

References

[Brown & Pietra+ 92] P.F. Brown, V.J.D. Pietra, R.L. Mercer, S.A.D. Pietra, J.C. Lai: An estimate of an upper bound for the entropy of English. *Computational Linguistics*, Vol. 18, No. 1, pp. 31–40, March 1992. 22

[Graves 13] A. Graves: Generating Sequences With Recurrent Neural Networks. *CoRR*, Vol. abs/1308.0850, August 2013. 2, 12

[Hochreiter & Schmidhuber 97] S. Hochreiter, J. Schmidhuber: Long short-term memory. *Neural computation*, Vol. 9, No. 8, pp. 1735–1780, November 1997. 9

[Lowe & Pow+ 15] R. Lowe, N. Pow, I. Serban, J. Pineau: The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems. In *Proceedings of the SIGDIAL Conference*, pp. 285–294. Association for Computational Linguistics, September 2015. 21, 29

[Papineni & Roukos+ 02] K. Papineni, S. Roukos, T. Ward, W.J. Zhu: BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311–318. Association for Computational Linguistics, July 2002. 22

[Shang & Lu+ 15] L. Shang, Z. Lu, H. Li: Neural Responding Machine for Short-Text Conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pp. 1577–1586. Association for Computational Linguistics, July 2015. 20

[Sutskever & Vinyals+ 14] I. Sutskever, O. Vinyals, Q.V. Le: Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 3104–3112, 2014. 2, 14

[Vinyals & Le 15] O. Vinyals, Q.V. Le: A Neural Conversational Model. In *Proceedings of the 31st International Conference on Machine Learning*, Vol. 37, July 2015. 2, 17

Appendix: Training Neural Networks

Back Propagation Algorithm

- **Recursion equation for the derivative with respect to the output of layer l:**

$$\delta_i^{(l)} = \frac{\partial \mathcal{L}}{\partial y_i^{(l)}} = \sum_{j=1}^{J} \frac{\partial \mathcal{L}}{\partial y_j^{(l+1)}} \frac{\partial y_j^{(l+1)}}{\partial y_i^{(l)}} = \sum_{j=1}^{J} \delta_j^{(l+1)} \frac{\partial y_j^{(l+1)}}{\partial y_i^{(l)}} \tag{31}$$

- **Derivative with respect to the weights of layer l:**

$$\frac{\partial \mathcal{L}}{\partial w_{ji}^{(l)}} = \sum_{j=1}^{J} \frac{\partial \mathcal{L}}{\partial y_j^{(l+1)}} \frac{\partial y_j^{(l+1)}}{\partial w_{ji}^{(l)}} = \sum_{j=1}^{J} \delta_j^{(l+1)} \frac{\partial y_j^{(l+1)}}{\partial w_{ji}^{(l)}} \tag{32}$$

Appendix: Training Recurrent Neural Networks

Back Propagation Through Time

- **Recursion equation for the derivative with respect to the output of layer l:**

$$\delta_i^{(l)}(t) = \sum_{j=1}^{J} \delta_j^{(l+1)}(t) \frac{\partial y_j^{(l+1)}(t)}{\partial y_i^{(l)}(t)} + \sum_{k=1}^{K} \delta_k^{(l+1)}(t+1) \frac{\partial y_k^{(l+1)}(t+1)}{\partial y_k^{(l)}(t+1)} \tag{33}$$

- **Derivative with respect to the weights of layer l:**

$$\frac{\partial \mathcal{L}}{\partial w_{ji}^{(l)}} = \sum_{t=1}^{T} \sum_{j=1}^{J} \delta_j^{(l+1)}(t) \frac{\partial y_j^{(l+1)}(t)}{\partial w_{ji}^{(l)}(t)} \tag{34}$$

Appendix: Evaluation

- **Recall@k metric [Lowe & Pow$^+$ 15]**
  - **Model names the k most likely responses to given context**
  - **Output is correct if the true response is among these k**
  - **Not useful for generative approaches**
  - **Used to set a benchmark on the *Ubuntu Dialogue Corpus***
- **Human experts**
  - **Ranking of responses from 0 (unsuitable) to $+2$ (suitable)**
  - **Performance criteria: grammar and fluency, logic consistency, semantic relevance, scenario dependence and generality**
- **In the future**
  - **Use a metric to compare generated responses with the true response**
  - **Dependent on a standardized embedding**
    $\rightarrow$ possible candidate: *skipped thought vectors*