Nizhny Novgorod State University

Institute of Information Technologies, Mathematics and Mechanics

Department of Computer software and supercomputer technologies

Educational course
«Introduction to deep learning
using the Intel® neon™ Framework»

Recurrent neural networks

Supported by Intel

Valentina Kustikova,
Phd, lecturer, department of Computer software
and supercomputer technologies

Content

- Recurrent neural networks
- Generalizing the concept of a computational graph. Deployment of the computational graph. The Elman network
- Training the recurrent neural network
- Deep recurrent neural networks
- Neural networks of long short-term memory
  - The problem statement
  - General structure of a long short-term memory unit
- Gated recurrent unit
- Example of a recurrent neural network for predicting a person's sex from a photo

RECURRENT NEURAL NETWORKS

Recurrent neural networks

- ***Recurrent neural networks*** (RNN) are networks with a feedback or crosslinking between different layers of neurons
- Initially recurrent networks are proposed for processing data sequences of the same type, i.e. the order of providing objects to the network is important
- Typical examples:
  - Natural language processing: processing a sound sequence, processing natural language texts
  - Computer vision: processing a sequence of video frames, some problems of image processing

- ***A recurrent network approximates the behavior of any dynamical system***

GENERALIZING THE CONCEPT OF A COMPUTATIONAL GRAPH. GRAPH DEPLOYMENT

Generalizing the concept of a computational graph. Deployment of the computational graph

- Introduction of recurrent neural networks requires generalizing the concept of a ***computational graph***, which was considered in the course earlier
- The computational graph of recurrent neural networks can contain ***loops*** that reflect the dependence of the variable value at the next time from its current value
- The idea is to ***deploy recursive calculations into a computational graph*** of a repetitive structure that usually reflects the sequence of events

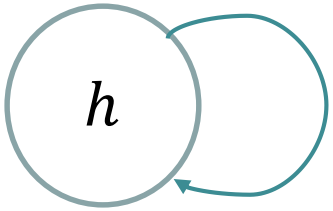
Deployment of the computational graph. An example of a classical dynamic system (1)

- A classical form of dynamic system:

$$h^{(t)} = f\left(h^{(t-1)}; \theta\right),$$

where $h^{(t)}$ is a state of the system at the time t,
θ is a set of parameters

- The system can be represented as a recurrent network



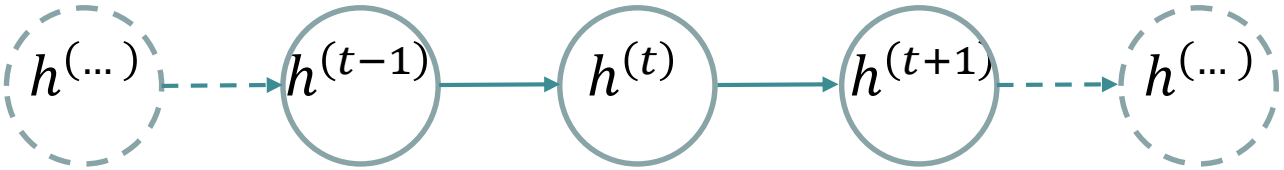
- The above equation is recurrent, since the state at each subsequent time depends on the state at the previous moment

Deployment of the computational graph. An example of a classical dynamic system (2)

- For a fixed time interval from 1 to τ, this equation can be expressed as follows:

$$h^{(\tau)} = f\left(h^{(\tau-1)};\theta\right) = f\left(f\left(h^{(\tau-2)};\theta\right);\theta\right) = \cdots = f\left(f\left(\ldots f\left(h^{(1)};\theta\right);\theta\right);\theta\right)$$

- The final expression no longer contains a recurrence relation and can be represented by an acyclic computation graph

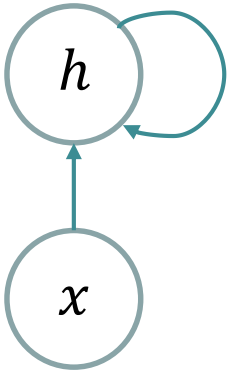


Deployment of the computational graph. An example of a system controlled by an external signal (1)

- Deployment of the computational graph. An example of a system controlled by an external signal $x^{(t)}$:

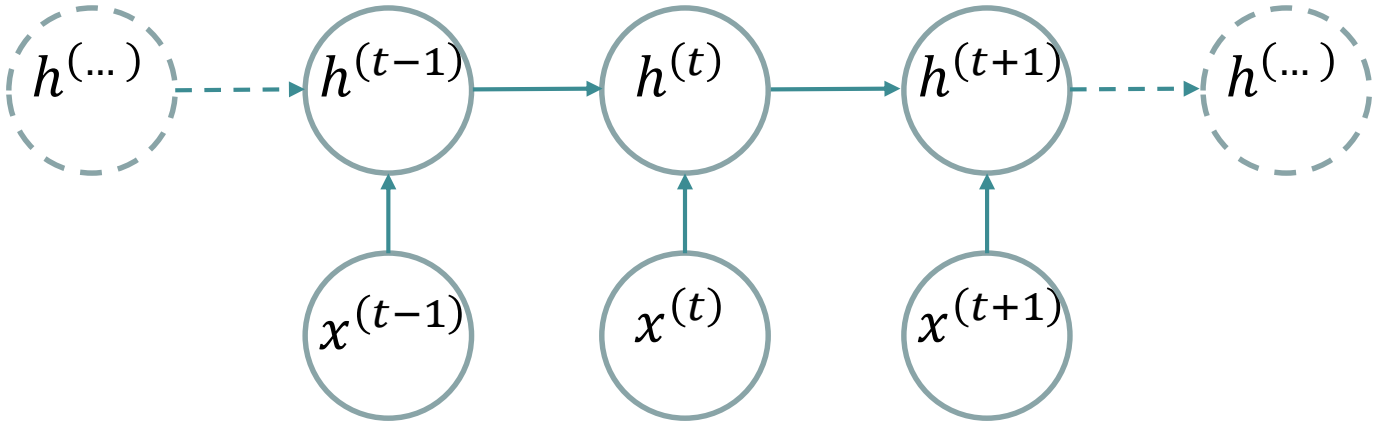
$$h^{(t)} = f\left(h^{(t-1)}; x^{(t)}; \theta\right)$$

- The corresponding recurrent network is as follows:



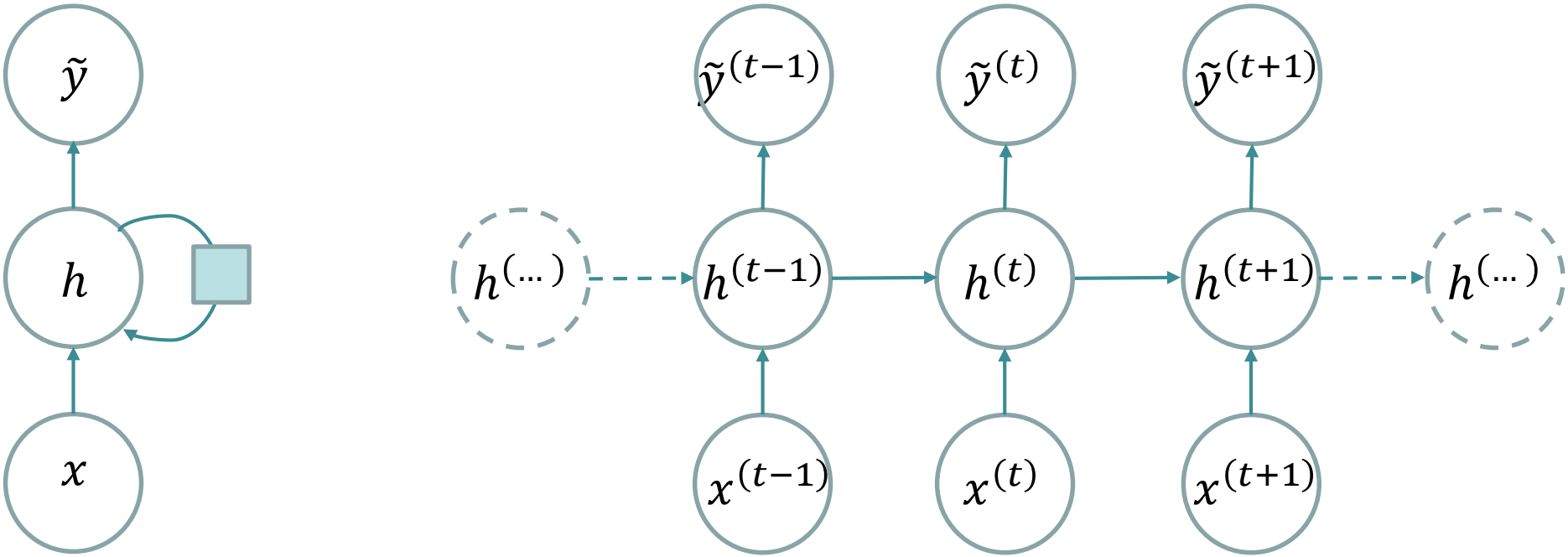
Deployment of the computational graph. An example of a system controlled by an external signal (2)

- In such a system, the current state contains information about the entire previous sequence of the signals
- The computational graph deployed in time for a dynamic system controlled by an external signal is as follows:



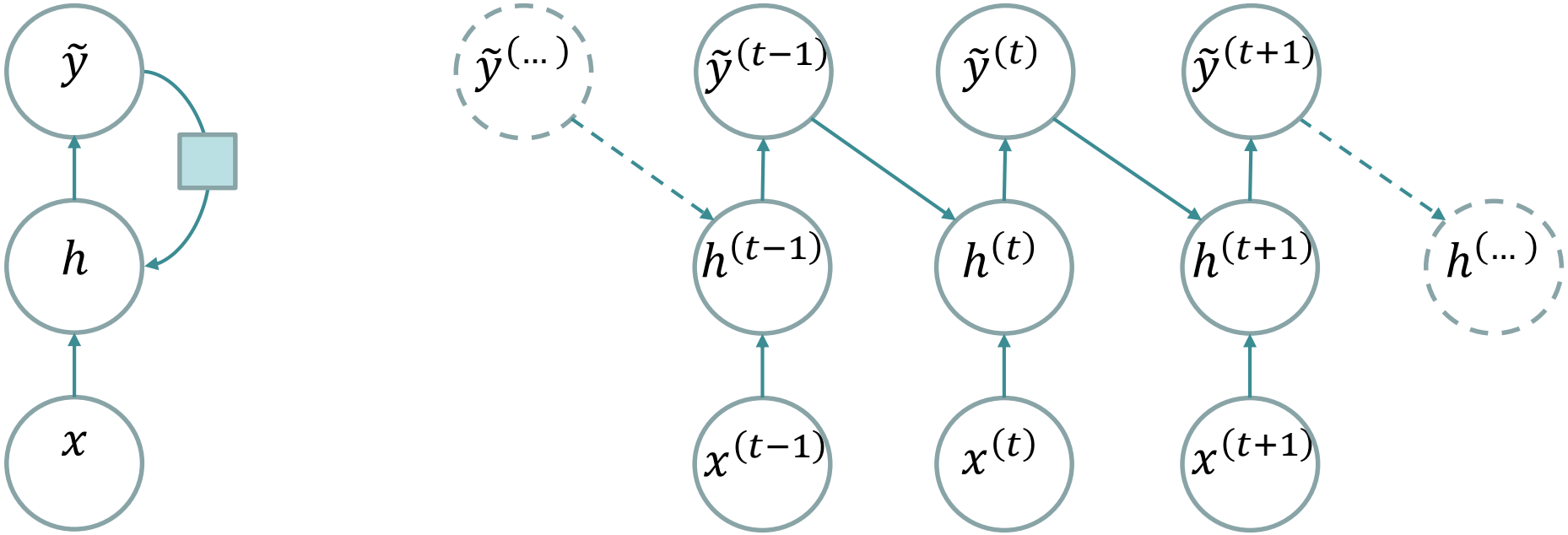
Deployment of the computational graph. Typical patterns of recurrence dependencies (1)

- A recurrent network providing an output signal at each time, and containing recurrent dependencies between the elements of the hidden layer



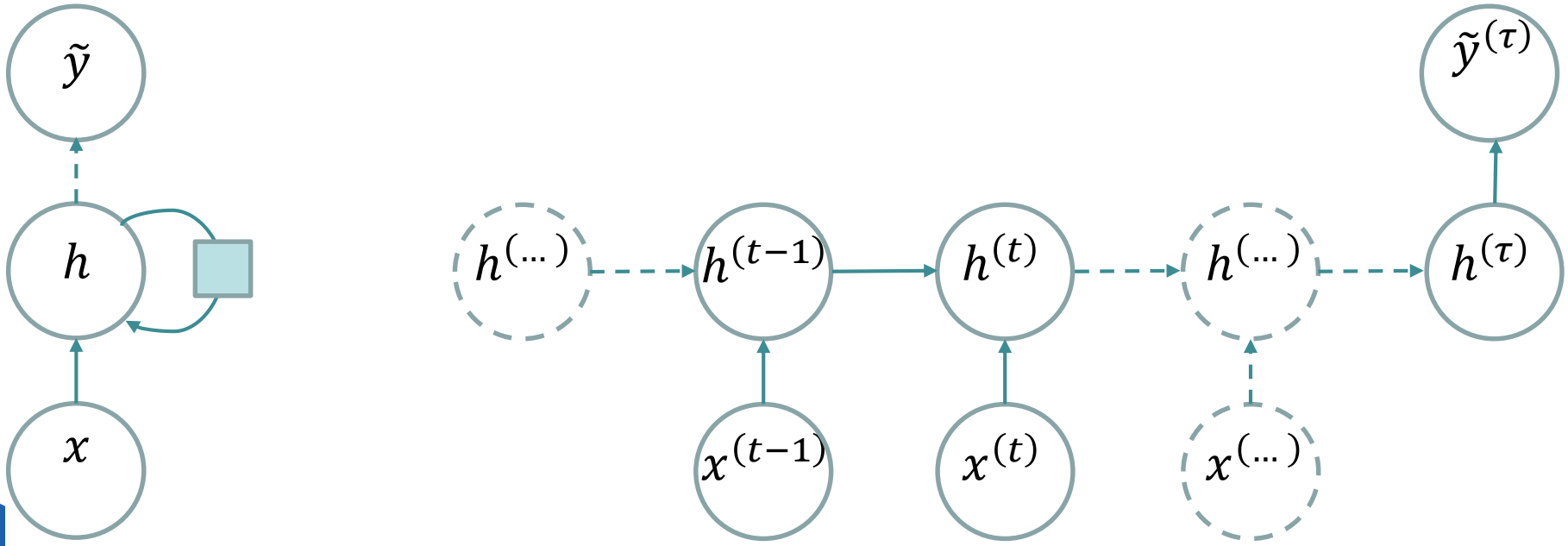
Deployment of the computational graph. Typical patterns of recurrence dependencies (2)

- A recurrent network that provides an output signal at each time, and has recurrent dependencies between the output element of the current time and the hidden element of the next time



Deployment of the computational graph. Typical patterns of recurrence dependencies (3)

- A recurrent network that has recurrent dependencies between elements of the hidden layer that reads the input data sequence and provides a single output signal
- The output element requires complete information about the past to predict the future



The Elman’s and Jordan’s networks

- A recurrent neural network with the dependence of hidden neurons on itself is the simplest one and is called the ***Elman's network***
  - The first and third patterns represent the implementation of the recurrent Elman’s network
  - The scheme of network deployment in time is different

- A recurrent network containing recurrent dependencies between the output of the current time and the hidden element of the next time is called the ***Jordan's network***

TRAINING THE RECURRENT NEURAL NETWORKS

Equations for the Elman's network

- The equations describing the internal state and the output of the network, which is obtained after the Elman's network deployment in time, are as follows:

$$a^{(t)} = Ux^{(t)} + Wh^{(t-1)} + b,$$
$$h^{(t)} = f\left(a^{(t)}\right) = f\left(Ux^{(t)} + Wh^{(t-1)} + b\right),$$
$$o^{(t)} = Vh^{(t)} + c, \qquad \tilde{y}^{(t)} = g\left(o^{(t)}\right) = g\left(Vh^{(t)} + c\right),$$

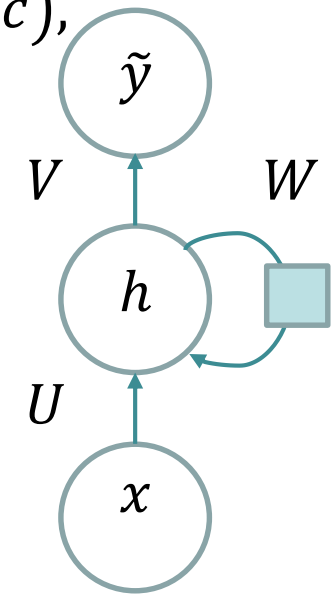
where U, W, V are weight matrices,

b, c are shifts,

$h^{(t)}$ is a vector of hidden variables at the time t (processing the training sample of the number t),

$\tilde{y}^{(t)}$ is a network output at the time t,

$f(\cdot), g(\cdot)$ are activation functions



The training problem statement for the Elman's network

- The task of training Elman's network is to minimize the total error for all examples of available training sequences:

$$J = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{\tau_n} d\left(\tilde{y}_n^{(t)}, y_n^{(t)}\right) \to \min_{U,W,V} \quad ,$$

where N is a number of input training sequences,
τ_n is a number of elements in the sequence with number n,
$y_n^{(t)}$ is an actual output (label) at the time t considering the sequence with number n,
$\tilde{y}_n^{(t)}$ is a network output at the time t when the sequence with number n is the input,
$d\left(\tilde{y}_n^{(t)}, y_n^{(t)}\right)$ – a measure of the similarity of label and network output (Euclidean distance or cross-entropy)

Backpropagation through time (1)

- A recurrent neural network can be deployed in time, thereby presenting it as a feed-forward network
- To train network parameters, you can apply the ***backpropagation through time method***

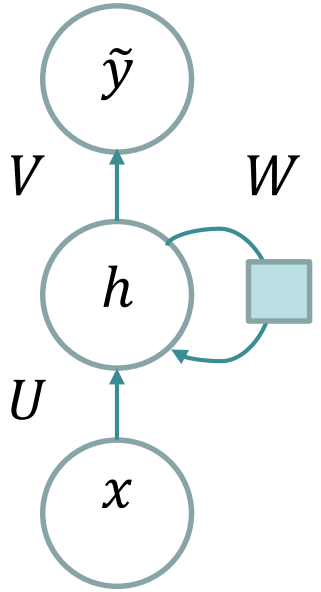
Backpropagation through time (2)

- ***Feed forward*** (from left to right over a network deployed in time)
  - The hidden states and outputs of the deployed network are calculated, as well as the gradients of the activation functions.
  - The complexity of computations is proportional to the length of the input sequence $O(\tau)$
  - Parallelization of calculations can not be performed, because each subsequent internal state of the system depends on the previous one
- ***Calculating the value of the cost function and the derivatives of this function***
- ***Backward*** (from right to left along the network deployed in time). The calculation of the error and updating of the network weights

DEEP RECURRENT NEURAL NETWORKS

Transforms at a recurrent layer

- Calculations in most typical recurrent networks can be decomposed into three blocks of parameters and their corresponding transforms:
  - Converting input to the hidden state
  - Converting previous hidden state to the next hidden state
  - Converting hidden state to output

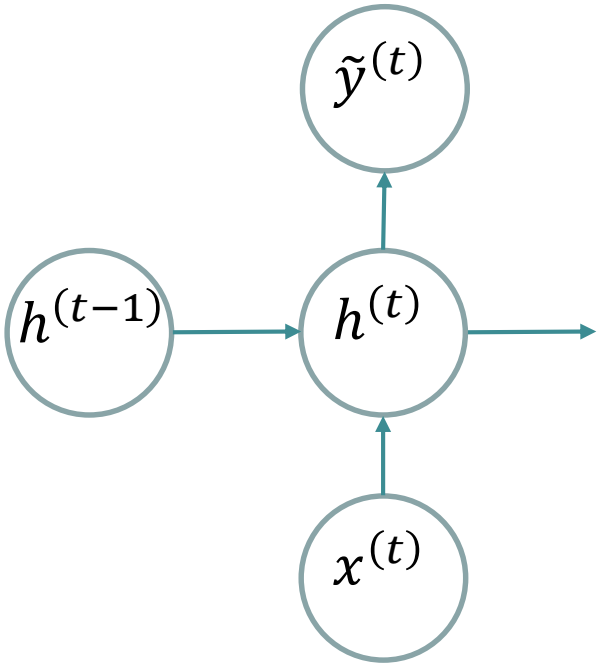


- Depending on how complex (deep) the transforms are, there are several types of recurrent networks

Types of recurrent neural networks (1)

- ***A conventional recurrent neural network***

$$h^{(t)} = f\left(Ux^{(t)} + Wh^{(t-1)} + b\right), \qquad \tilde{y}^{(t)} = g\left(Vh^{(t)} + c\right)$$



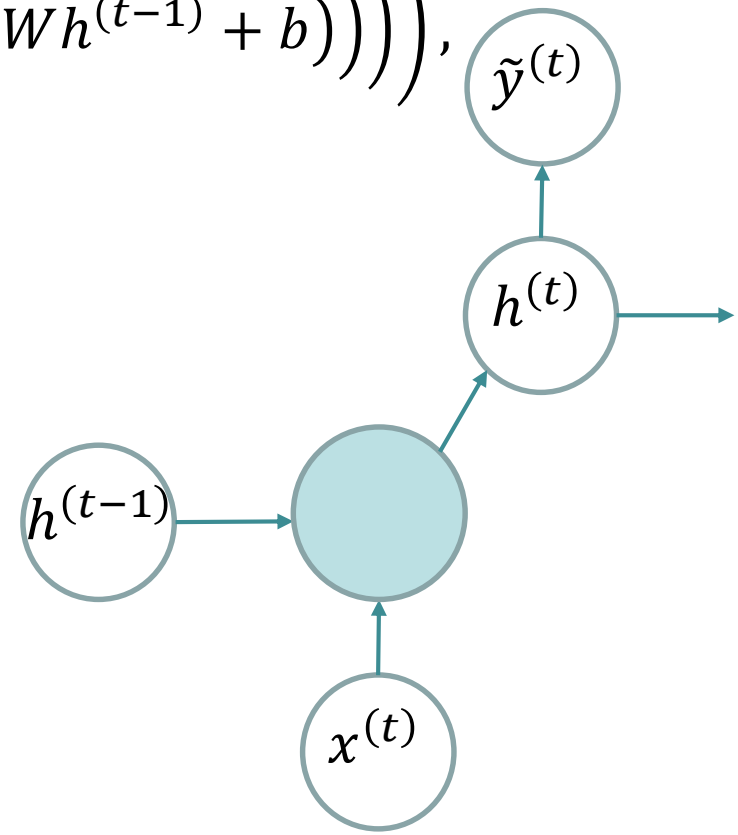
Types of recurrent neural networks (2)

- ***Deep transition recurrent neural network*** (DT-RNN)

$$h^{(t)} = f\left(Ux^{(t)} + Wh^{(t-1)} + b\right)$$

$$= \varphi_L\left(U_L^T \varphi_{L-1}\left(U_{L-1}^T \varphi_{L-2}\left(\dots \varphi_1\left(Ux^{(t)} + Wh^{(t-1)} + b\right)\right)\right)\right),$$

where L is a number of network layers between input and hidden layers

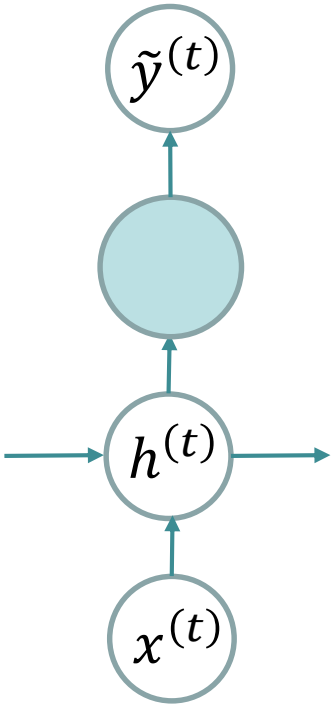


Types of recurrent neural networks (3)

- ***Deep output recurrent neural network*** (DO-RNN)

$$\tilde{y}^{(t)} = g\left(Vh^{(t)} + c\right) = \psi_L\left(V_L^T\psi_{L-1}\left(V_{L-1}^T\psi_{L-2}\left(\ldots\psi_1\left(Vh^{(t)} + c\right)\right)\right)\right),$$

where L is a number of network layers between a hidden and output layers

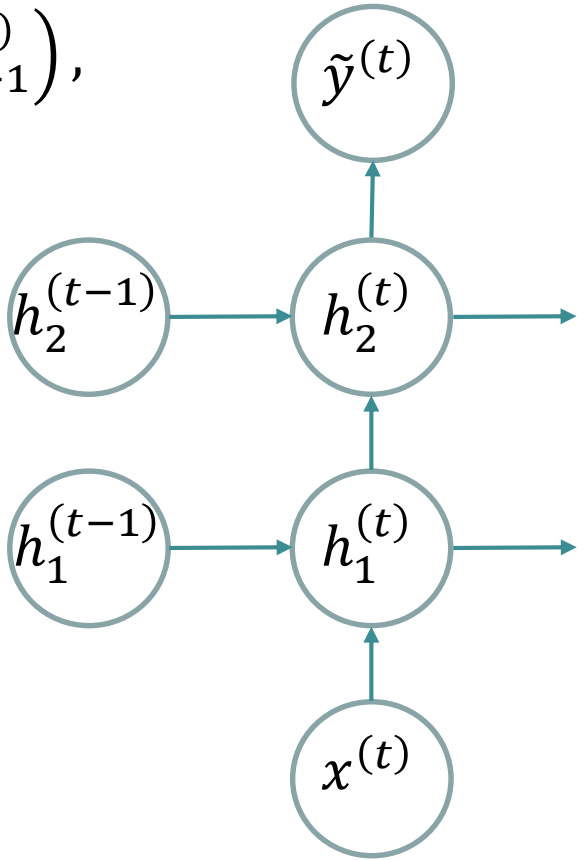


Stack of recurrent layers

- Constructing a stack of ordinary recurrent networks is another way to construct deep recursive networks

$$h_l^{(t)} = f_l\left(W_l^T h_l^{(t-1)} + U_l^T h_{l-1}^{(t)}\right),$$

where $h_l^{(t)}$ is a hidden state of the system at the layer l at the time t



NEURAL NETWORKS OF LONG SHORT-TERM MEMORY

The problem statement

- Until now, recurrent neural networks have been trained through their deployment over time and applying a modified backpropagation algorithm

- ***If there are enough long input sequences, the network "forgets" the information about samples during the training***

- In some cases, it is necessary for the network to "remember" information about samples at the start of the sequence

The problem statement. Examples of computer vision problems (1)

- ***Human action recognition***
  - The task is to determine what type of movement the person is making on the video sequence (sitting, standing, walking, running, jumping, etc.)
  - For different types of movements, the initial actions can be identical, therefore for decision-making it is necessary to know the full sequence of actions

* Recognition of human actions. Action database [http://www.nada.kth.se/cvap/actions].

The problem statement. Examples of computer vision problems (2)

- ***Semantic segmentation of videos***
  - The goal is to determine the class of the object to which each pixel of the scene belongs
  - A video is a set of related frames
  - During semantic segmentation of the current frame, you can use the information obtained during the segmentation of previous frames

The problem statement. Examples of computer vision problems (3)

- ***Image captioning***
  - This task is at the junction of computer vision and natural language processing
  - The goal of the task is to construct a coherent sentence describing the contents of the image
  - At each stage of the description construction, an attempt is made to reconstruct the next word in the sentence based on the context of the image

1. Top view of the lights of a city at night, with a well-illuminated square in front of a church in the foreground;
2. People on the stairs in front of an illuminated cathedral with two towers at night;

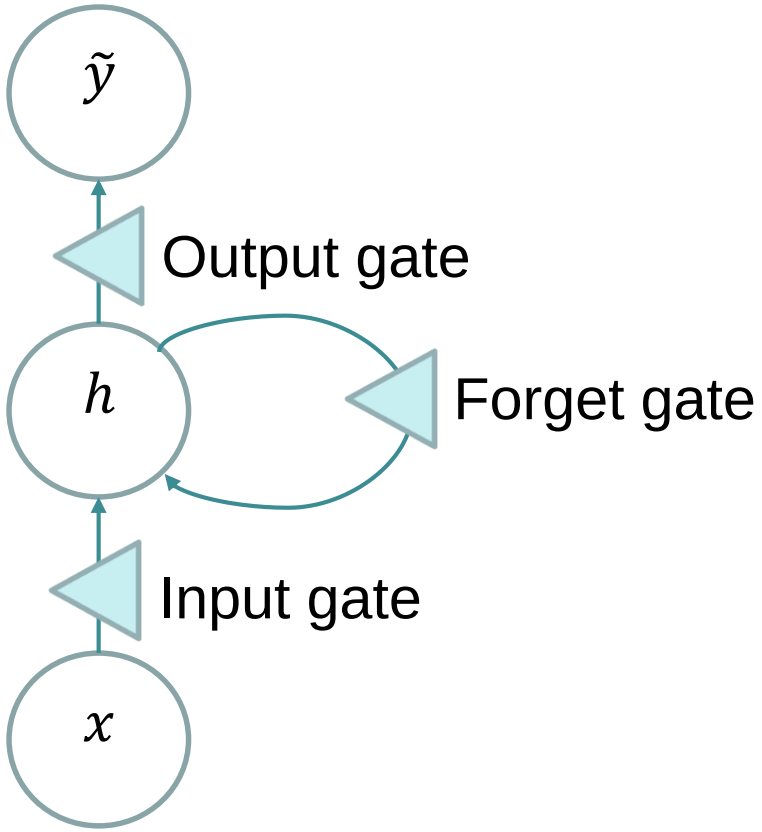
* Mao J., Xu W., Yang Y., Wang J., Huang Z., Yuille A.L. Deep Captioning with Multimodal Recurrent Neural Networks (m-RNN) // ICLR. – 2015. – [https://arxiv.org/pdf/1412.6632.pdf].

The idea of a long short-term memory unit (1)

- The general structure of a long short-term memory unit (LSTM) assumes the presence of neurons having a connection on themselves
- The data is input to the neuron and the processed data is output
- The recurrent connection with its input has a weight equal to 1
- If there is no new input data, the value of the neuron is overwritten and remains unchanged

The idea of a long short-term memory unit (2)

- Three gates are used to control this structure, which determine the passage of the signal: ***the input gate***, ***the forget gate*** and ***the output gate***

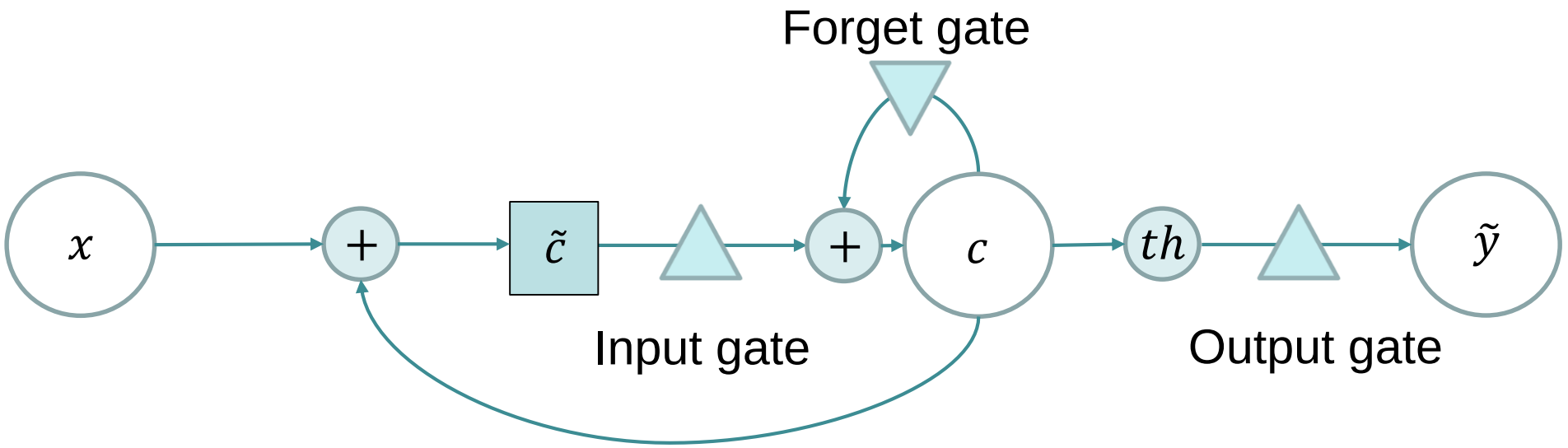


The idea of a long short-term memory unit (3)

- If ***the input gate is open*** (set to 1), the input signal is written to the hidden neuron, the value is recorded and stored in the neuron due to the recurrent feedback
- If ***the input valve is closed*** (set to 0), the values entering the neuron input do not affect its content

- If you want to get the value stored in the cell, you need to open the output gate (set to 1)
- If the value contained in the cell is required to be “forgotten”, it is necessary to close the forget gate. Further, the value will be erased from the neuron, and the neuron will be ready to store the new input value

The scheme of a long short-term memory unit

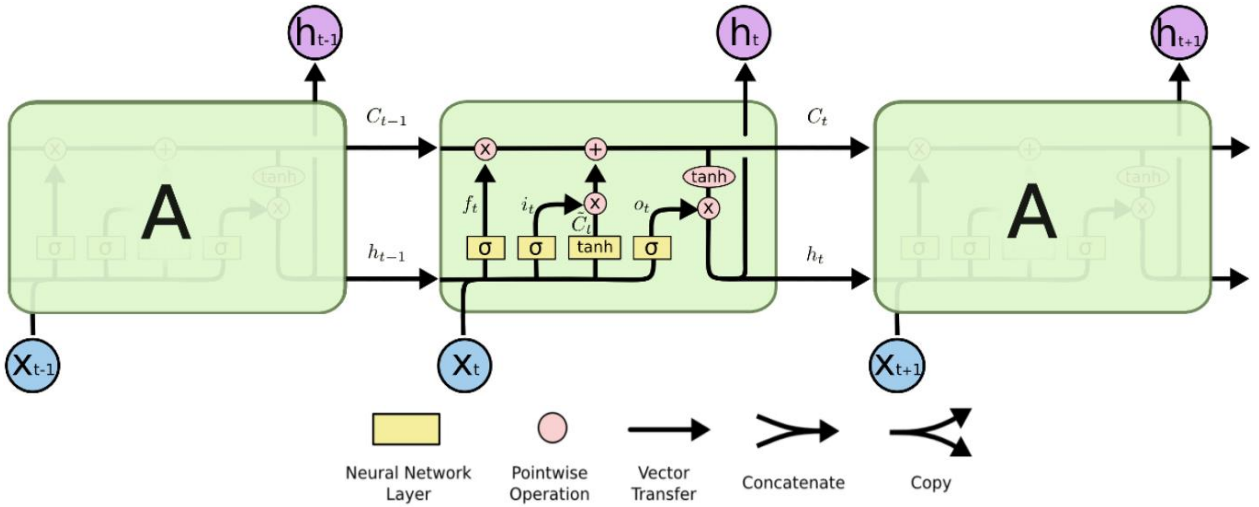


- c is the memory cell, $\tilde{c}$ is the new memory cell content
- th is the activation function hyperbolic tangent

* Chung J., Gulcehre C., Cho K.H., Bengio Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. – 2014. – [https://arxiv.org/pdf/1412.3555.pdf].

Long short-term memory network (1)

- Let us consider the implementation of ***long short-term memory network*** (LSTM)
- LSTMs have their own repeating unit, deployed in time



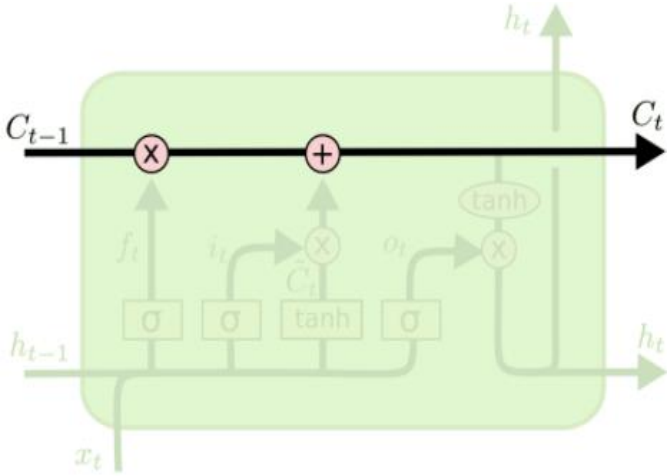
* Hochreiter S., Schmidhuber J. Long short-term memory // Neural Computation. – 1997. – P.1735-1780. – [http://www.bioinf.jku.at/publications/older/2604.pdf].

** Greff K., Srivastava R.K., Koutnık J., Steunebrink B.R., Schmidhuber J. LSTM: A Search Space Odyssey // Transactions on Neural Networks and Learning Systems. – 2017. – [https://arxiv.org/pdf/1503.04069.pdf].

*** Understanding LSTM Networks [http://colah.github.io/posts/2015-08-Understanding-LSTMs].

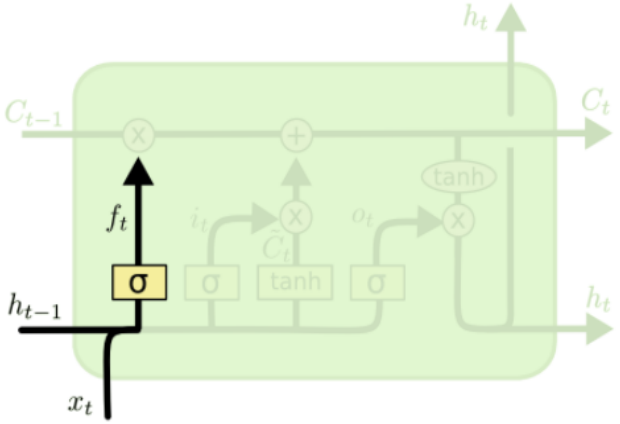
Long short-term memory network (2)

- The main component of the LSTM-unit is its ***state*** C_t, which is transmitted in time
- A unit is capable of adding or removing information from a state, carefully regulated by structures called ***gates***
- A gate is a way of conveying information. It consists of a sigmoidal layer and an operation of element-wise multiplication



Long short-term memory network (3)

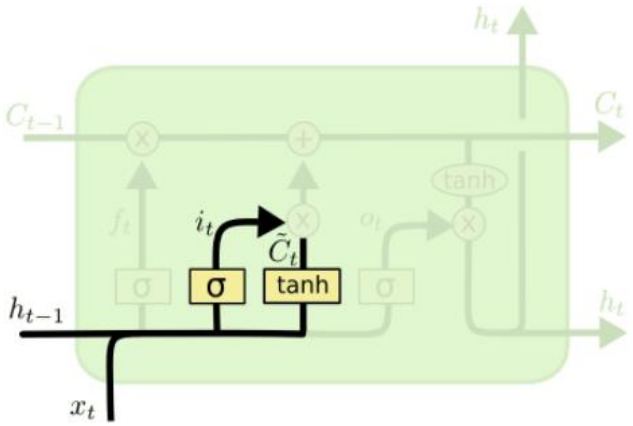
- ***Step 1:*** make a decision which elements from the state must be "forgotten"
  - To go through the ***forget gate layer***. The forget gate layer is the sigmoid layer of the network
  - Determine the weights with which the state elements are skipped
  - A value of 0 means that the element is not skipped, 1 – the element is skipped completely



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Long short-term memory network (4)

- ***Step 2:*** determine which new information should be stored in the unit state
  - The sigmoidal layer is the ***input gate layer***. It decides which values to update
  - A layer with an activation function corresponding to a hyperbolic tangent constructs a vector of new candidates that are added to the current state

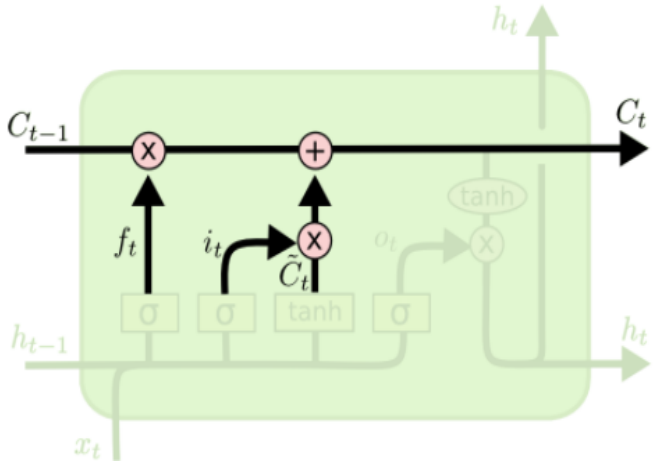


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Long short-term memory network (5)

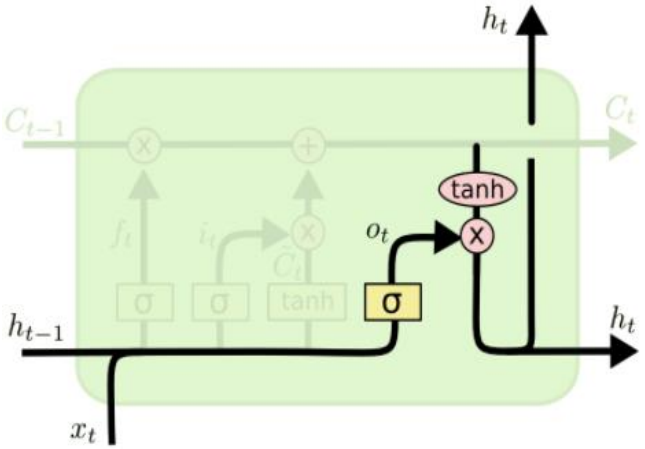
- ***Step 3:*** update unit status
  - It is necessary to remove information from the state vector for which it was decided that it can be “forgotten” and add new information



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Long short-term memory network (6)

- ***Step 4:*** make a decision what to use as the output
  - The output is based on the unit state and represents its filtered version
  - Make a decision which parts of the state must be removed by introducing a sigmoidal layer
  - Next, the unit state elements are normalized to the interval $[-1, 1]$ using the hyperbolic tangent function and multiplied by the output of the sigmoidal layer



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Note

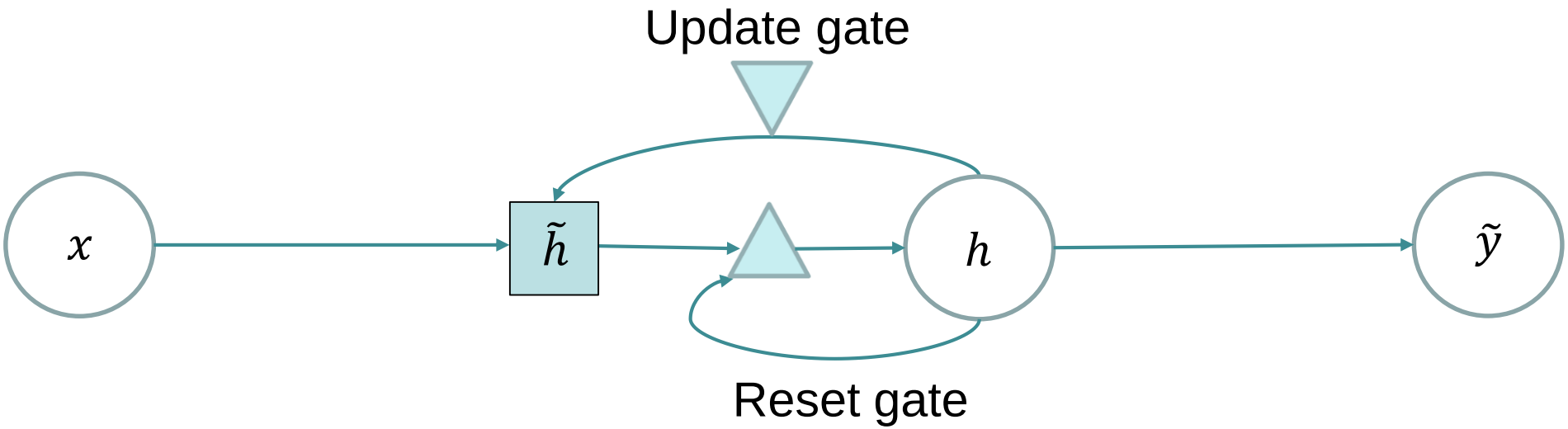
- The above implementation of the LSTM-unit is not the only possible one, there are many modifications

* Understanding LSTM Networks [http://colah.github.io/posts/2015-08-Understanding-LSTMs].

GATED RECURRENT UNIT

Gated recurrent unit (1)

- An alternative to a long short-term memory unit is the gated recurrent unit (GRU)
- The scheme of gated recurrent Unit (GRU) construction:



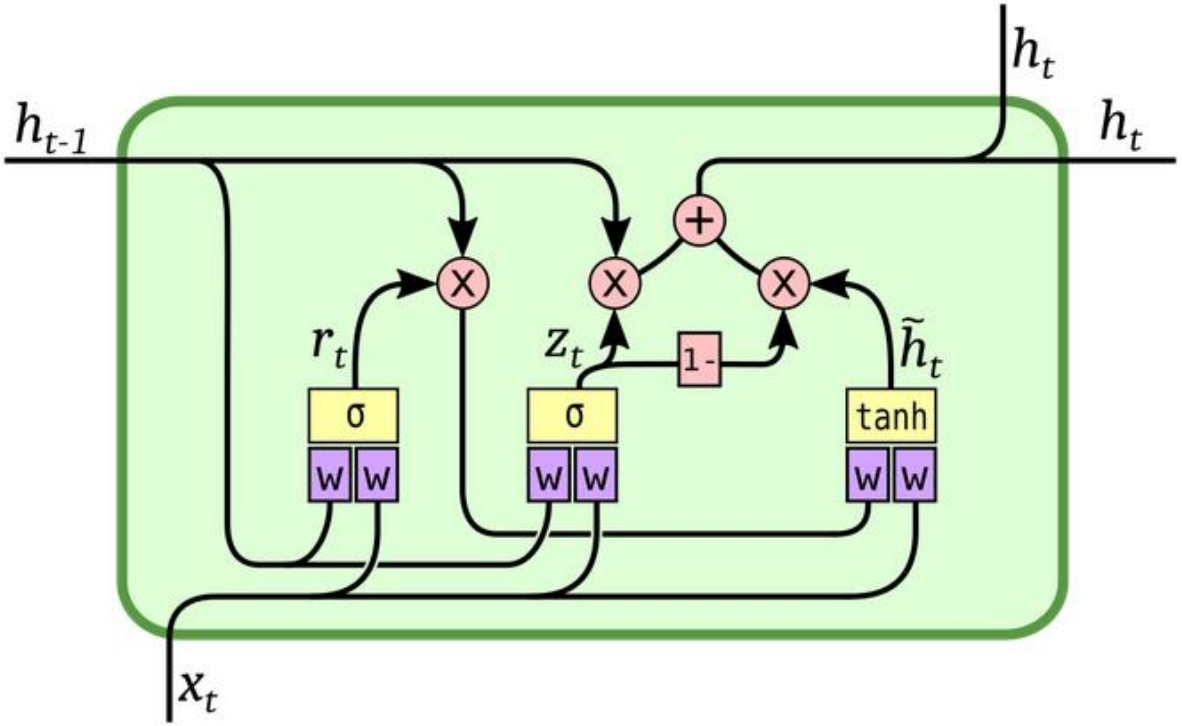
- h is an activation, $\tilde{h}$ is an activation candidate

* Chung J., Gulcehre C., Cho K.H., Bengio Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. – 2014. – [https://arxiv.org/pdf/1412.3555.pdf].

Gated recurrent unit (2)

- Gated recurrent unit contains one gate less than long short-term memory unit
- The update gate determines the amount of information received from the previous state
- The reset gate works by analogy with the forget gate in long short-term memory unit

Gated recurrent unit. Implementation



- h_{t-1}, h_t are hidden neuron states, x_t is an input
- r_t implements reset gate, z_t implements update gate

* Visualization of RNN units. Diagram of RNN unrolling, LSTM and GRU [https://kvitajakub.github.io/2016/04/14/rnn-diagrams].

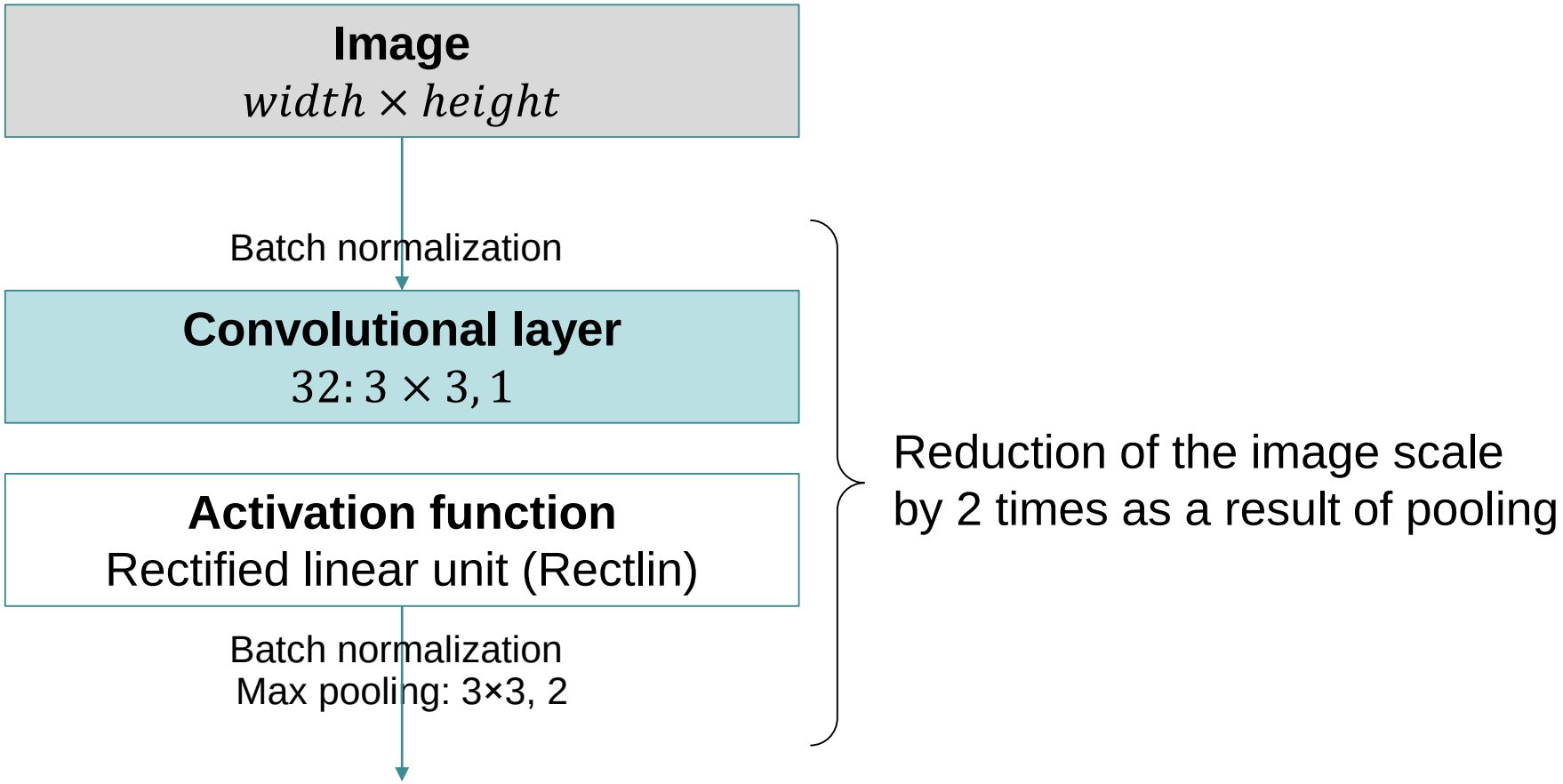
EXAMPLE OF A RECURRENT NEURAL NETWORK FOR PREDICTING A PERSON'S SEX FROM A PHOTO

Example of the recurrent neural network

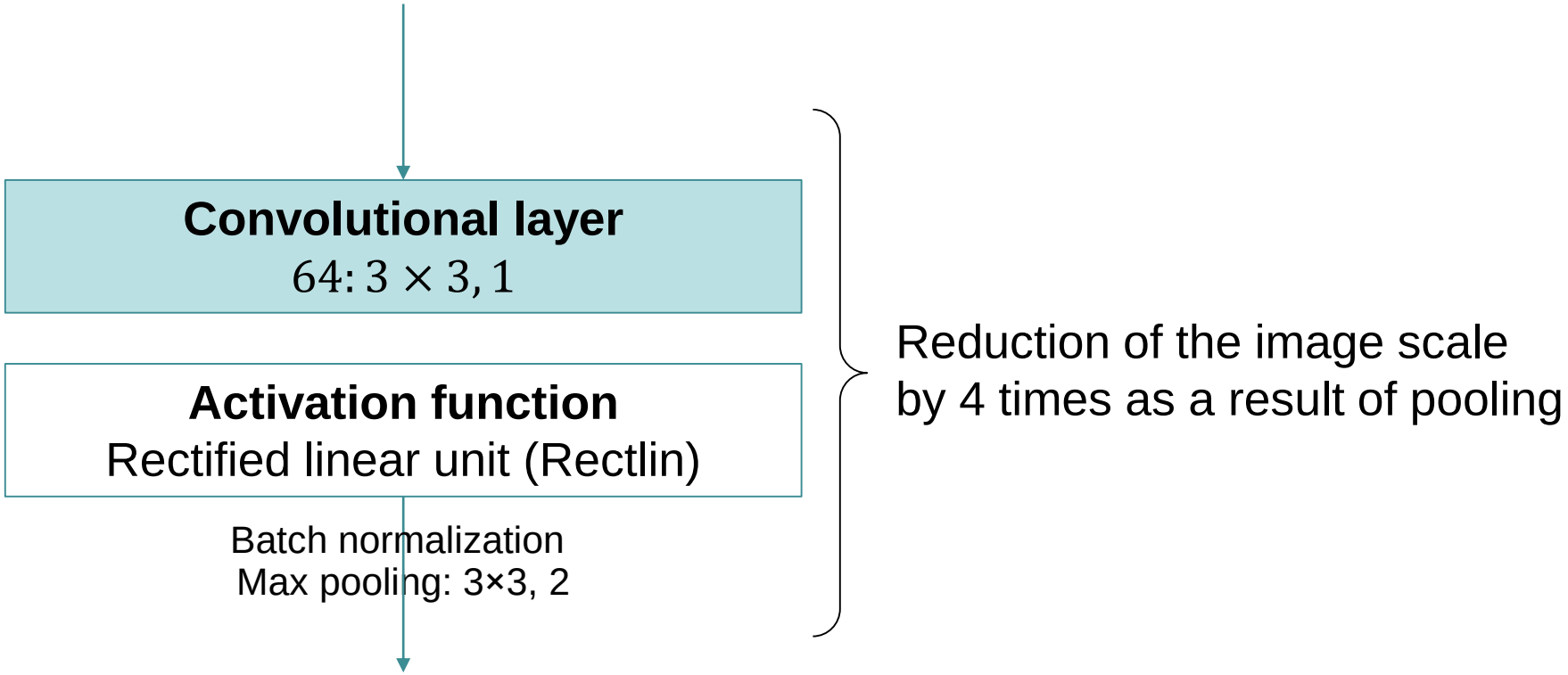
- Let us continue considering the problem of predicting a person's sex from a photo
- This example demonstrates the use of recurrent networks for non-classical input data, which explicitly does not represent a sequence of elements of the same type

- The structure of the recurrent block being developed is described in the following paper:
  - Visin F., Ciccone M., Romero A., Kastner K., Cho K., Bengio Y., Matteucci M., Courville A. ReSeg: A Recurrent Neural Network-based Model for Semantic Segmentation // In CVPR Deep Vision Workshop, 2016. – 2016. – [https://arxiv.org/abs/1511.07053]

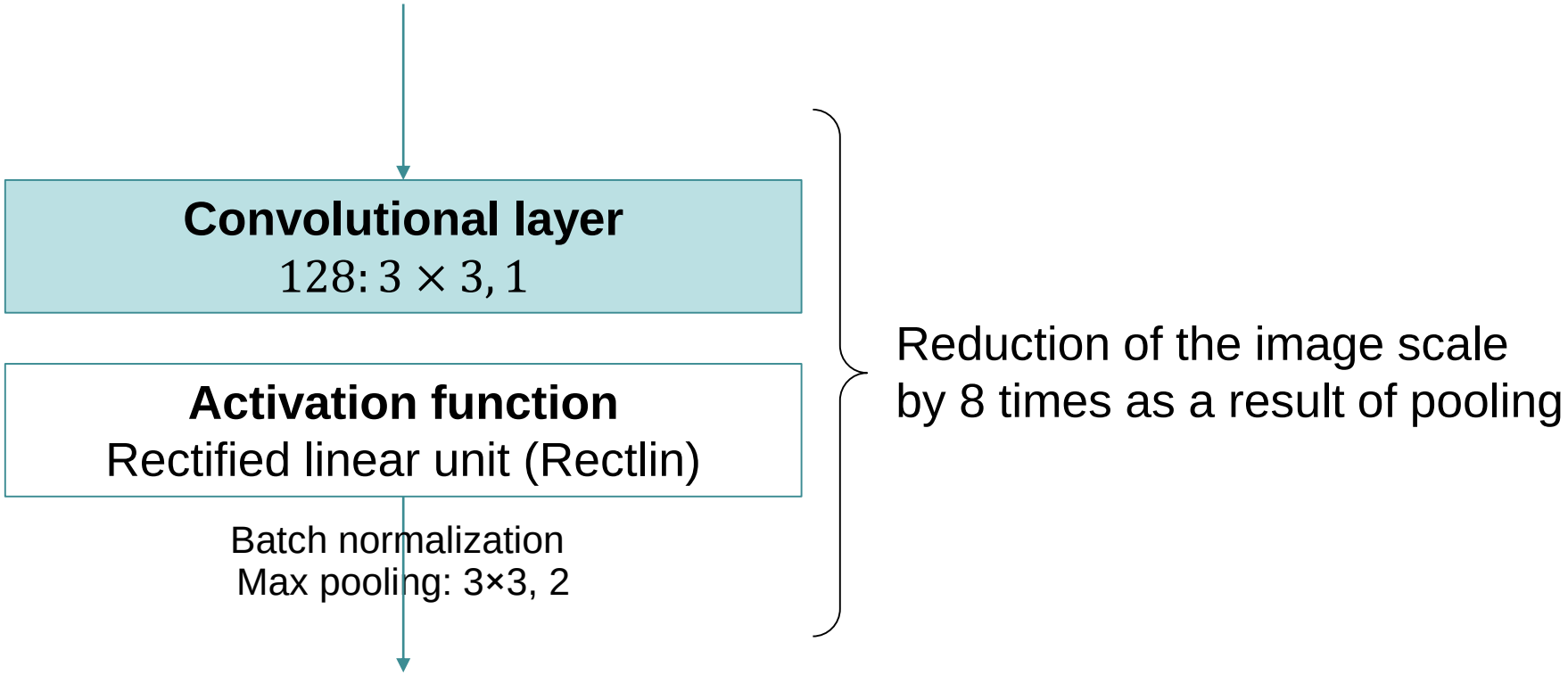
Recurrent neural network (1)



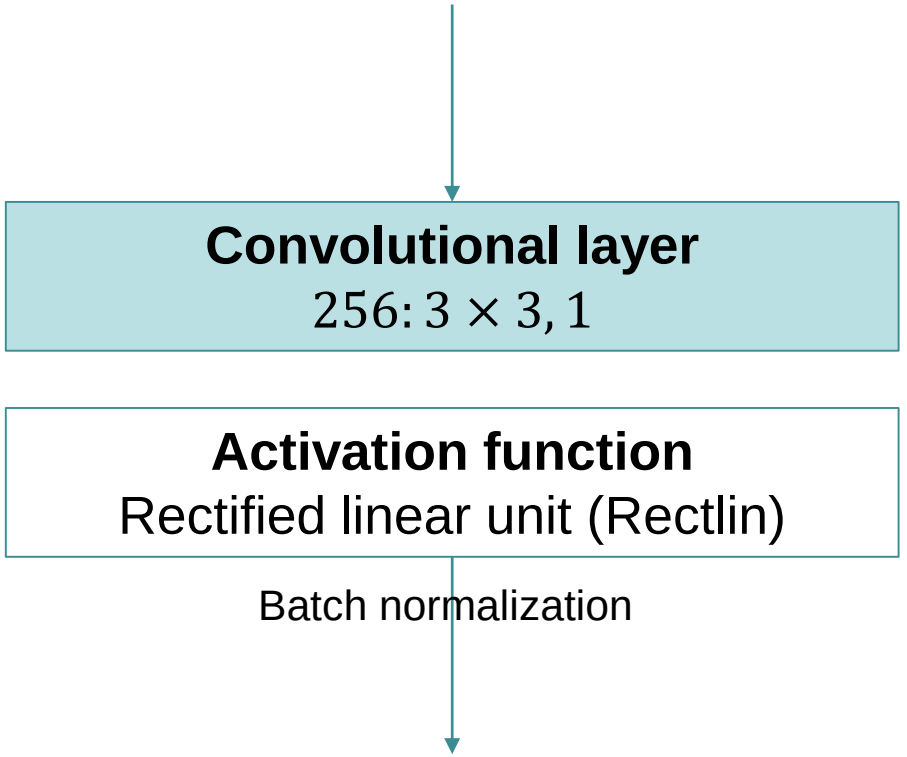
Recurrent neural network (2)



Recurrent neural network (3)

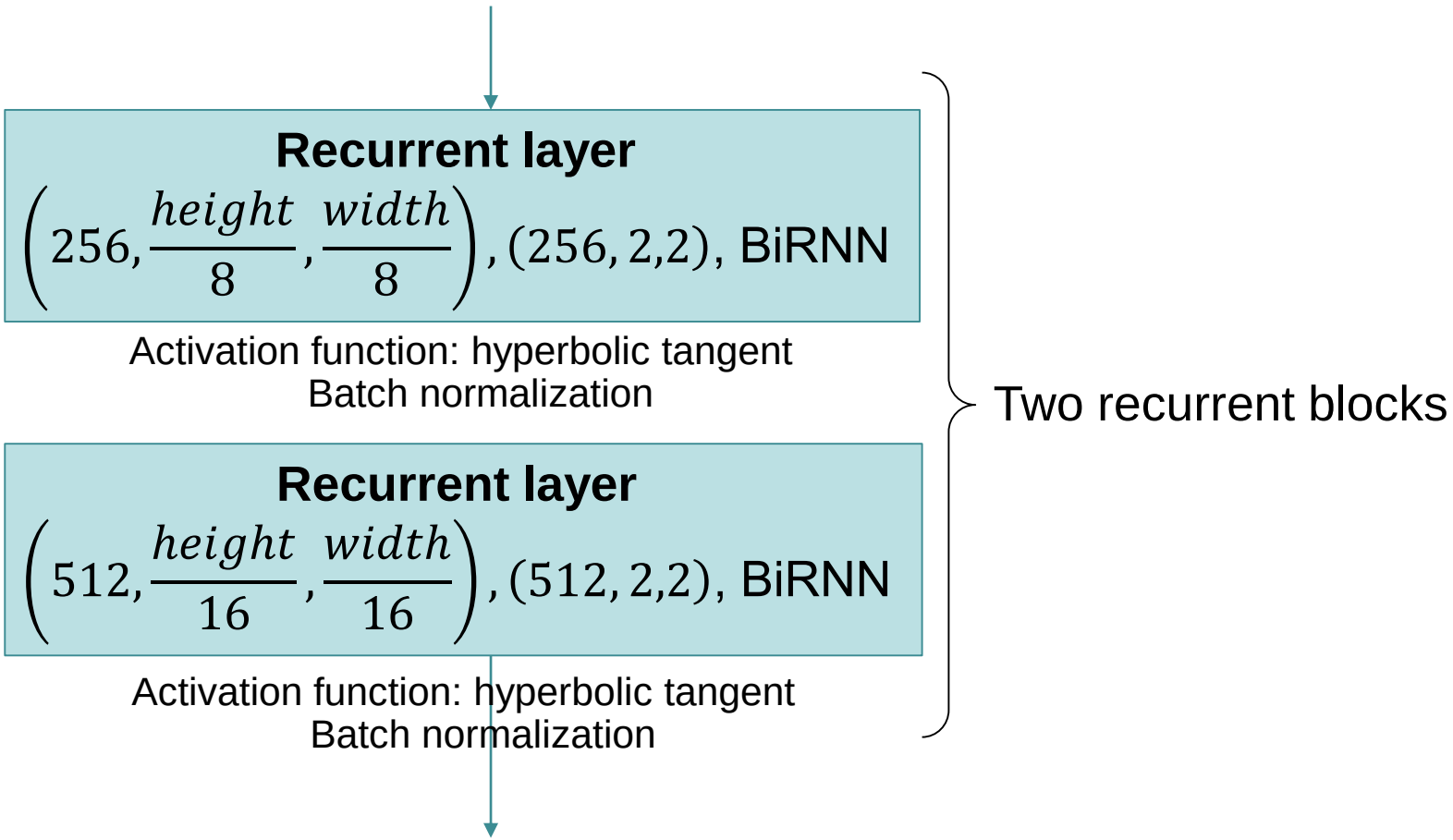


Recurrent neural network (4)

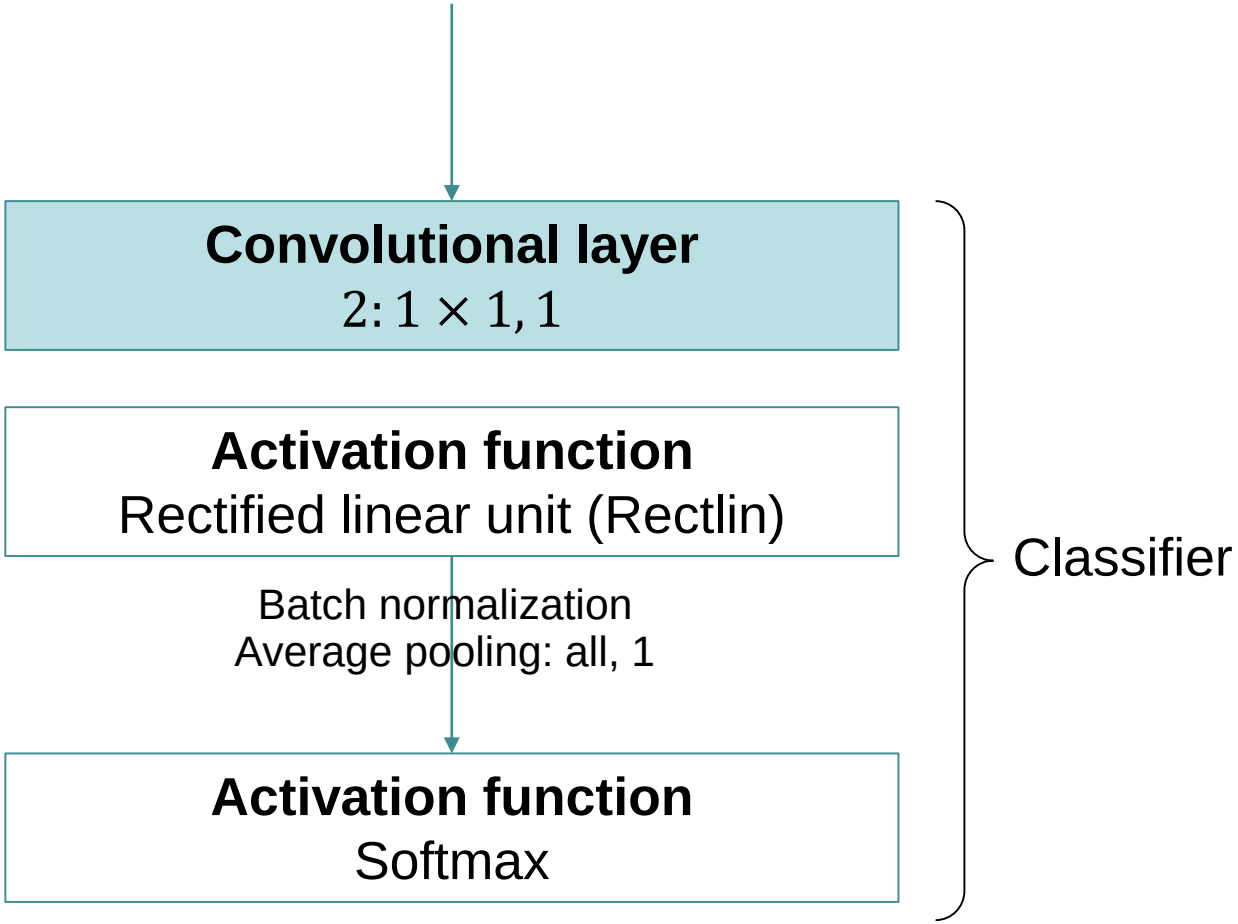


The shape of the feature map at the input of the convolutional layer and at the output one differs only in the third dimension

Recurrent neural network (5)

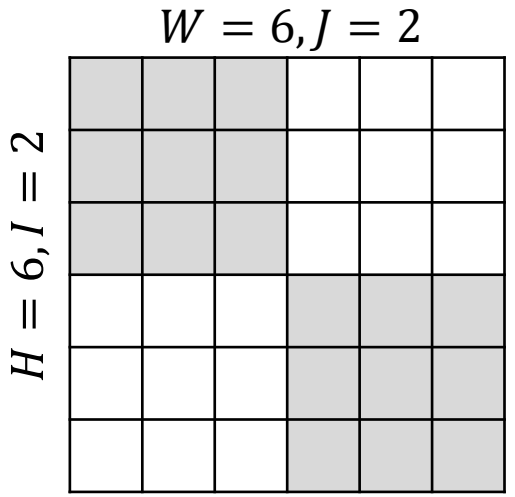


Recurrent neural network (6)



The structure of recurrent block (1)

- Let $H \times W \times C$ is a shape of feature map which is an input of the recurrent block, H is a height, W is a width, C is a number of channels
- We divide the feature map into $I \times J$ blocks, each block contains vectors consisting of C elements
- We denote each block as $p_{ij} \in \mathbb{R}^{H_p \times W_p \times C}$



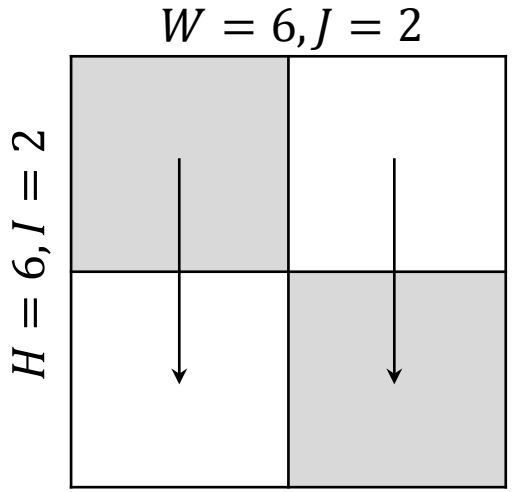
* Visin F., Ciccone M., Romero A., Kastner K., Cho K., Bengio Y., Matteucci M., Courville A. ReSeg: A Recurrent Neural Network-based Model for Semantic Segmentation // In CVPR Deep Vision Workshop, 2016. – 2016. – [https://arxiv.org/abs/1511.07053].

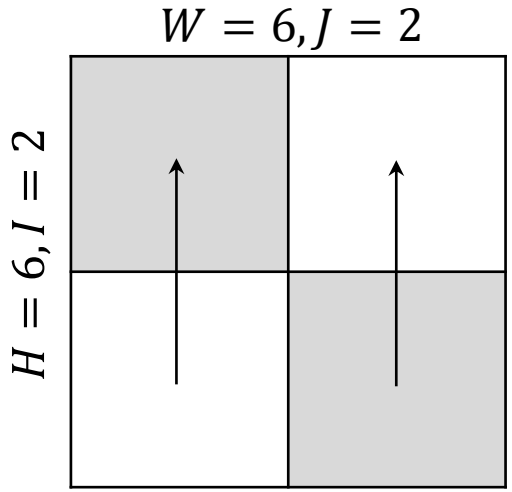
The structure of recurrent block (2)

- We implement two bidirectional recurrent networks by columns and rows of block matrices constructed on feature maps:
  - The first bidirectional network consists of two recurrent layers:
    - The first layer corresponds to traversing the blocks from the top to the bottom
    - The second layer corresponds to traversing the blocks from the bottom to the top
  - The second bidirectional network consists of two recurrent layers too:
    - The first layer corresponds to block traversing from the left to the right
    - The second layer corresponds to the blockade from the right to the left

The structure of recurrent block (3)

- The scheme of the element dependence for the first bidirectional network is as follows:





- The feature map elements are calculating according to the equations:

$$o_{i,j}^{\downarrow} = f^{\downarrow}\left(z_{i-1,j}^{\downarrow}, p_{ij}\right), i = 1, \dots, I$$
$$o_{i,j}^{\uparrow} = f^{\uparrow}\left(z_{i+1,j}^{\uparrow}, p_{ij}\right), i = I, \dots, 1$$

The structure of recurrent block (4)

- The feature map at the output of the first recurrent layer is constructed by concatenating feature vectors counted by traversing the blocks from the top to the bottom and from the bottom to the top

$$O^{\updownarrow} = \left(\left(o^{\downarrow}_{i,j}, o^{\uparrow}_{i,j}\right)\right)_{i,j} = \left(o^{\updownarrow}_{ij}\right), \qquad o^{\updownarrow}_{ij} \in \mathbb{R}^{2U},$$

  where U is a number of elements recurrent layer elements

- The second bidirectional network receives the feature map $O^{\updownarrow}$, this network is constructed in the same way, but horizontal dependencies are established between the blocks

* Visin F., Ciccone M., Romero A., Kastner K., Cho K., Bengio Y., Matteucci M., Courville A. ReSeg: A Recurrent Neural Network-based Model for Semantic Segmentation // In CVPR Deep Vision Workshop, 2016. – 2016. – [https://arxiv.org/abs/1511.07053].

Example of a recurrent neural network for predicting a person's sex from a photo (1)

```
def generate_rnn1_cls_model(input_shape=(3, 128, 128)):
    iC = input_shape[0]
    iH = input_shape[1]
    iW = input_shape[2]
    class_count = 2
    layers = [
      DataTransform(transform=Normalizer(divisor=128.0)),
      # convolutional encoder / feature extractor
      # resolution 1
      BatchNorm(),
      Conv(fshape=(3, 3, 32), padding=2, strides=1,
           dilation=2, init=Kaiming(), bias=Constant(0),
           activation=Rectlin()),
...
```

Example of a recurrent neural network for predicting a person's sex from a photo (2)

```
        BatchNorm(),
        Pooling(fshape=(3, 3), padding=1,strides=2, op='max'),
        # resolution 1/2
        Conv(fshape=(3, 3, 64), padding=2, strides=1,
             dilation=2, init=Kaiming(), bias=Constant(0),
             activation=Rectlin()),

        BatchNorm(),
        Pooling(fshape=(3, 3), padding=1,strides=2, op='max'),
        # resolution 1/4
        Conv(fshape=(3, 3, 128), padding=2, strides=1,
             dilation=2, init=Kaiming(), bias=Constant(0),
             activation=Rectlin()),
```

Example of a recurrent neural network for predicting a person's sex from a photo (3)

```
        BatchNorm(),
        Pooling(fshape=(3, 3), padding=1,strides=2, op='max'),
        # resolution 1/8
        Conv(fshape=(3, 3, 256), padding=2, strides=1,
             dilation=2, init=Kaiming(), bias=Constant(0),
             activation=Rectlin()),
        BatchNorm(),
        # implemented recurrent block
        SpatialRNN(input_shape=(256, iH // 8, iW // 8),
            block_shape=(256, 2, 2), RNN=BiRNN,
            RNN_params={'output_size': 256,
                'init': GlorotUniform(), 'activation': Tanh()}
        ), # outputs: (2 * 256, iH // 16, iW // 16, N)
        # # resolution 1/16
```

Example of a recurrent neural network for predicting a person's sex from a photo (4)

```
        BatchNorm(),
        SpatialRNN(input_shape=(512, iH // 16, iW // 16),
            block_shape=(512, 2, 2), RNN=BiRNN,
            RNN_params={'output_size': 512,
                'init': GlorotUniform(), 'activation': Tanh()}
        ), # outputs: (2 * 512, iH // 32, iW // 32, N)
        # # resolution 1/32

        BatchNorm(),
        Conv(fshape=(1, 1, class_count), padding=1, strides=1,
             dilation=1, init=Kaiming(), bias=Constant(0),
             activation=Rectlin()),
        Pooling(fshape='all', padding=0, strides=1, op='avg'),
        Activation(Softmax())    ]
```

Example of a recurrent neural network for predicting a person's sex from a photo (5)

```
    model = Model(layers=layers)

    cost = GeneralizedCost(costfunc=CrossEntropyMulti())

    return (model, cost)
```

Infrastructure

- CPU: Intel® Xeon® CPU E5-2660 0 @ 2.20GHz
- GPU: Tesla K40s 11Gb
- OS: Ubuntu 16.04.4 LTS
- Frameworks:
  - Intel® neon™ Framework 2.6.0
  - CUDA 8.0
  - Python 3.5.2
  - Intel® Math Kernel Library 2017 (Intel® MKL)

Experiments

Network id	Training parameters	Accuracy, %	Training time, s
RNN	batch_size = 128 epoch_count = 90 backend = gpu GradientDescentMomentum(0.01, momentum_coef=0.9, wdecay=0.0005)	81.9	29571

Summary results

Network id	Accuracy, %	Training time, s
FCNN-1	71.2	932
FCNN-2	73.5	977
FCNN-3	**77.7**	1013
CNN-1	79.3	1582
CNN-2	**83.5**	2030
ResNet-18 (90 epochs)	**81.3**	15127
ResNet-50 (30 epochs)	80.9	11849
TL-1	85.6	119975
TL-2	85.3	119989
TL-3	**86.3**	39282
RNN	**81.9**	29571

Conclusion

- The complexity of constructing recurrent networks is quite high, especially in problems where there are no explicit sequences of input data
- The main application field of recurrent networks is natural language processing
- At present, recurrent networks are more often involved for solving problems of analysis and image processing
  - Shi B., et al. An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition. – 2015. – [https://arxiv.org/abs/1507.05717]
  - Visin F., et al. ReSeg: A Recurrent Neural Network-based Model for Semantic Segmentation // In CVPR Deep Vision Workshop, 2016. – 2016. – [https://arxiv.org/abs/1511.07053]
  - Cheang T.K., el al. Segmentation-free Vehicle License Plate Recognition using ConvNet-RNN. – 2017. – [https://arxiv.org/abs/1701.06439]

Literature

- Haykin S. Neural Networks: A Comprehensive Foundation. – Prentice Hall PTR Upper Saddle River, NJ, USA. – 1998.
- Osovsky S. Neural networks for information processing. – 2002.
- Goodfellow I., Bengio Y., Courville A. Deep Learning. – MIT Press. – 2016. – [http://www.deeplearningbook.org].

Authors

- **Kustikova Valentina Dmitrievna**
  Phd, lecturer, department of Computer software and supercomputer technologies, Institute of Information Technologies, Mathematics and Mechanics, Nizhny Novgorod State University
  valentina.kustikova@itmm.unn.ru
- **Zhiltsov Maxim Sergeevich**
  master of the 1st year training, Institute of Information Technology, Mathematics and Mechanics, Nizhny Novgorod State University
  zhiltsov.max35@gmail.com
- **Zolotykh Nikolai Yurievich**
  D.Sc., Prof., department of Algebra, geometry and discrete mathematics, Institute of Information Technologies, Mathematics and Mechanics, Nizhny Novgorod State University
  nikolai.zolotykh@itmm.unn.ru