Национальный исследовательский
Нижегородский государственный университет им. Н.И. Лобачевского
Институт информационных технологий, математики и механики

Образовательный курс
«Введение в глубокое обучение с использованием Intel® neon™ Framework»

Рекуррентные нейронные сети

При поддержке компании Intel

Кустикова Валентина,
к.т.н., ст.преп. каф. МОСТ ИИТММ,
ННГУ им. Н.И. Лобачевского

Содержание

- Рекуррентные нейронные сети
- Обобщение понятия графа вычислений. Развертывание графа вычислений. Сеть Элмана
- Обучение рекуррентных сетей
- Глубокие рекуррентные нейронные сети
- Нейронные сети долгой кратковременной памяти
  - Описание проблемы
  - Общая структура сети долгой кратковременной памяти
- Управляемые рекуррентные ячейки
- Пример применения рекуррентных сетей для решения задачи классификации пола человека по фотографии

ПОНЯТИЕ РЕКУРРЕНТНЫХ СЕТЕЙ

Рекуррентные нейронные сети

- ***Рекуррентные нейронные сети*** (Recurrent Neural Networks, RNN) – сети с обратными или перекрестными связями между различными слоями нейронов
- Изначально предложены ***для обработки последовательностей однотипных данных***, т.е. порядок предоставления объектов сети имеет важное значение
- Типовые примеры задач:
  - Задачи распознавания речи: обработка последовательности звуков, обработка текстов естественного языка
  - Задачи компьютерного зрения: обработка последовательности кадров видео, некоторые задачи обработки изображений
- ***Рекуррентная сеть аппроксимирует поведение любой динамической системы***

ОБОБЩЕНИЕ ПОНЯТИЯ ГРАФА ВЫЧИСЛЕНИЙ. РАЗВЕРТЫВАНИЕ ГРАФА ВЫЧИСЛЕНИЙ

Развертывание графа вычислений

- Введение рекуррентных нейронных сетей требует обобщения понятия ***графа вычислений***, которое рассматривалось в курсе ранее
- Граф вычислений рекуррентных нейронных сетей может содержать ***циклы***, которые отражают зависимости значения переменной в следующий момент времени от ее текущего значения
- На данном этапе используется идея ***развертывания рекуррентных вычислений в граф вычислений***, имеющий повторяющуюся структуру, которая обычно отражает цепочку событий

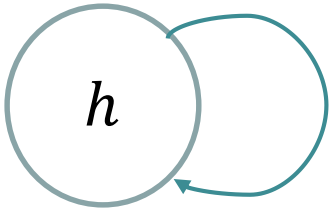
Развертывание графа вычислений. Пример классической динамической системы (1)

- Классическая форма динамической системы:

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

где $h^{(t)}$ – состояние системы в момент времени t,
θ – множество параметров

- Система может быть представлена в виде рекуррентной сети



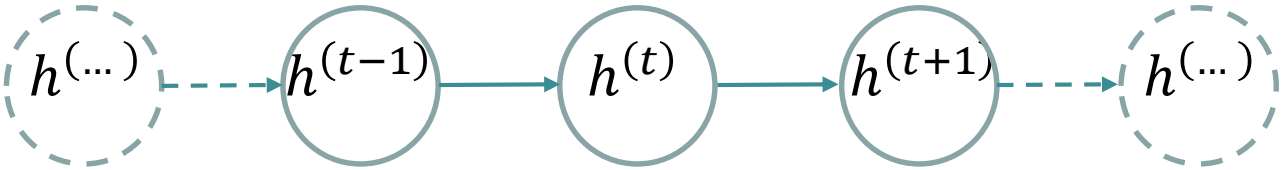
- Приведенное уравнение является рекуррентным, поскольку состояние в каждый следующий момент времени зависит от состояния в предыдущий момент

Развертывание графа вычислений. Пример классической динамической системы (2)

- Для фиксированного промежутка времени от 1 до τ данное уравнение можно развернуть

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

- Финальное выражение уже не содержит рекуррентной зависимости и может быть представлено ациклическим графом вычислений

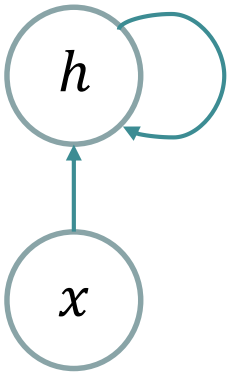


Развертывание графа вычислений. Пример системы, управляемой внешним сигналом (1)

- Динамическая система, управляемая внешним сигналом $x^{(t)}$:

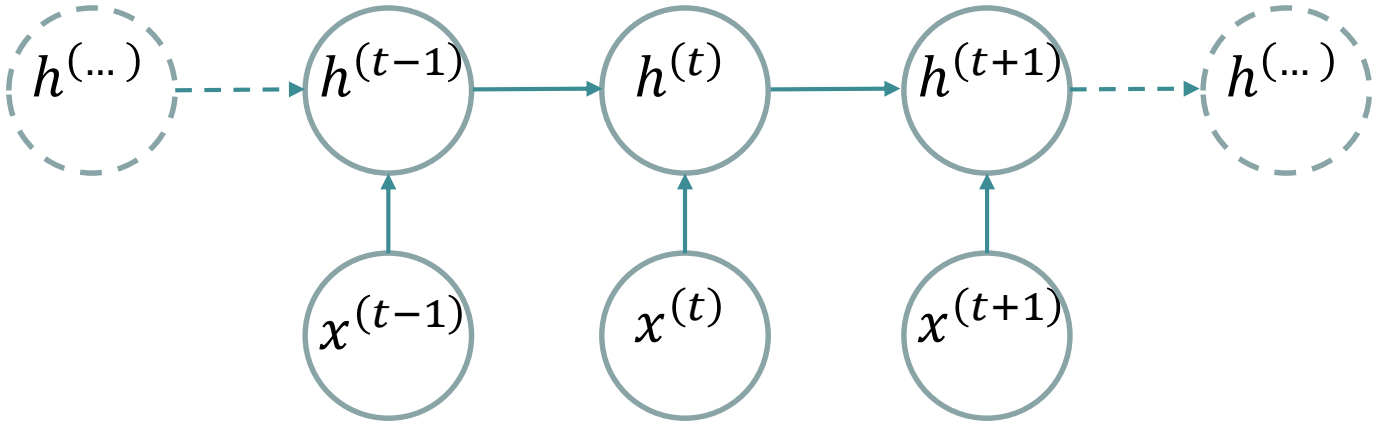
$$h^{(t)} = f\left(h^{(t-1)}; x^{(t)}; \theta\right)$$

- Соответствующая рекуррентная сеть имеет вид:



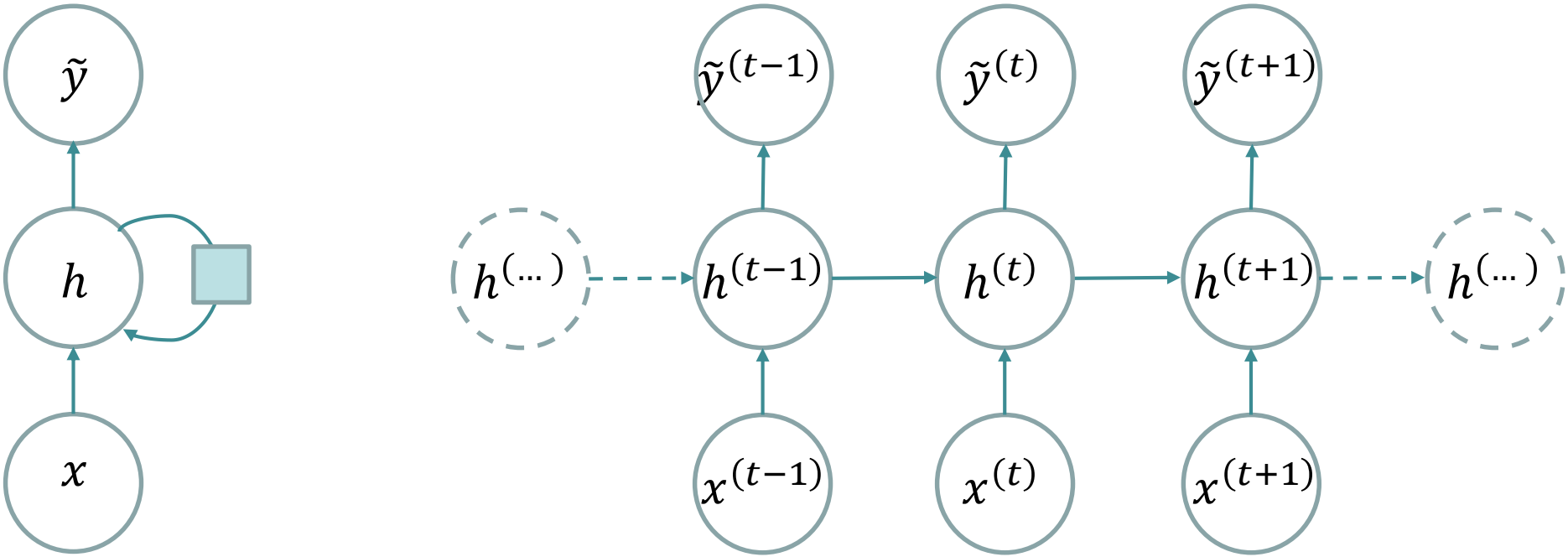
Развертывание графа вычислений. Пример системы, управляемой внешним сигналом (2)

- В такой системе текущее состояние содержит информацию о всей предыдущей последовательности сигналов
- Граф вычислений, развернутый во времени, для динамической системы, управляемой внешним сигналом имеет следующий вид:



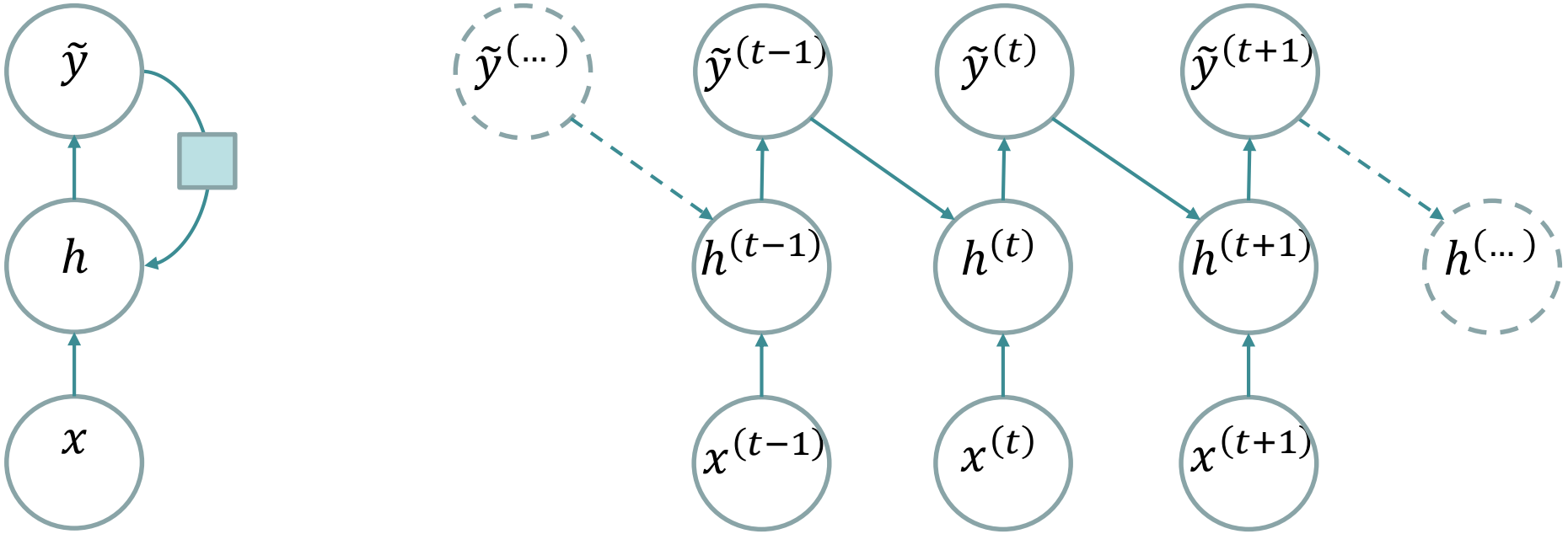
Развертывание графа вычислений. Типовые шаблоны рекуррентных зависимостей (1)

- Рекуррентная сеть, обеспечивающая выходной сигнал на каждом временном шаге, и имеющая рекуррентные зависимости между элементами скрытого слоя



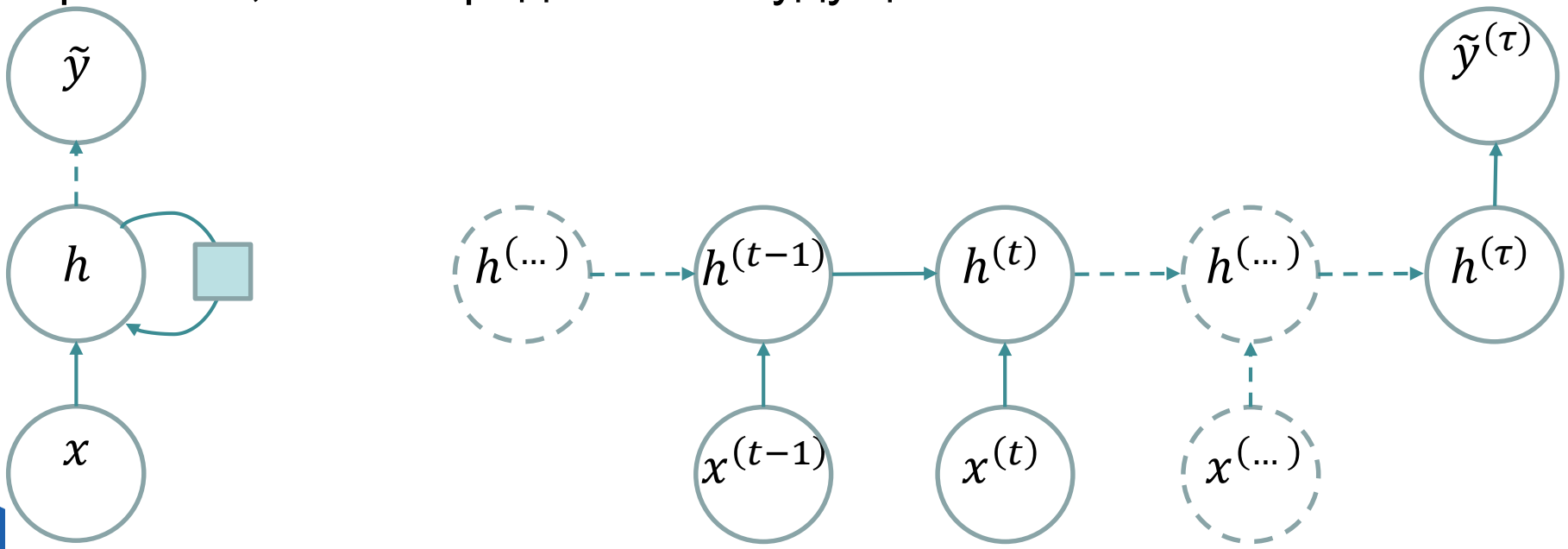
Развертывание графа вычислений. Типовые шаблоны рекуррентных зависимостей (2)

- Рекуррентная сеть, обеспечивающая выходной сигнал на каждом временном шаге, и имеющая рекуррентные зависимости между выходным элементом текущего момента времени и скрытым элементом следующего момента времени



Развертывание графа вычислений. Типовые шаблоны рекуррентных зависимостей (3)

- Рекуррентная сеть, имеющая рекуррентные зависимости между элементами скрытого слоя, которая читает входную последовательность данных и обеспечивает единственный выходной сигнал
- Выходному элементу требуется полная информация о прошлом, чтобы предсказать будущее



Сети Элмана и Джордана

- Рекуррентная нейронная сеть с зависимостью скрытых нейронов на себя является простейшей и называется ***рекуррентной сетью Элмана*** (Elman's network)
  - Первый и третий типовой шаблон представляют собой реализацию рекуррентной сети Элмана
  - Различается схема развертывания сети во времени

- Рекуррентная сеть, имеющая рекуррентные зависимости между выходным элементом текущего момента и скрытым элементом следующего момента, называется ***сетью Джордана*** (Jordan's network)

ОБУЧЕНИЕ РЕКУРРЕНТНЫХ СЕТЕЙ

Уравнения, описывающие сеть Элмана

- Уравнения, описывающие внутреннее состояние и выход сети, которая получена в результате развертывания сети Элмана во времени, имеют следующий вид:

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

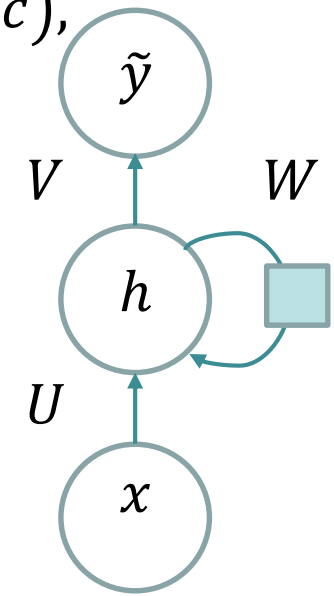
где U, W, V – матрицы весов,

b, c – вектора сдвига,

$h^{(t)}$ – вектор скрытых переменных в момент t (при обработке примера с номером t из заданной входной последовательности),

$\tilde{y}^{(t)}$ – выход сети в момент t,

$f(\cdot), g(\cdot)$ – функции активации



Постановка задачи обучения сети Элмана

- Задача обучения сети Элмана состоит в том, чтобы минимизировать суммарную ошибку по всем примерам имеющихся последовательностей:

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

где N – количество входных последовательностей,
τ_n – количество элементов последовательности с номером n,
$y_n^{(t)}$ – реальный выход (разметка) в момент t при рассмотрении последовательности с номером n,
$\tilde{y}_n^{(t)}$ – выход сети в момент t при получении входной последовательности с номером n,
$d\left(\tilde{y}_n^{(t)}, y_n^{(t)}\right)$ – мера сходства разметки и выхода сети (Евклидово расстояние или кросс-энтропия)

Метод обратного распространения ошибки с разворачиванием сети во времени (1)

- Рекуррентную нейронную сеть можно развернуть во времени, тем самым, представив ее в виде сети с прямым распространением сигнала
- Для обучения параметров сети можно применить ***метод обратного распространения ошибки с разворачиванием сети во времени*** (backpropagation through time)

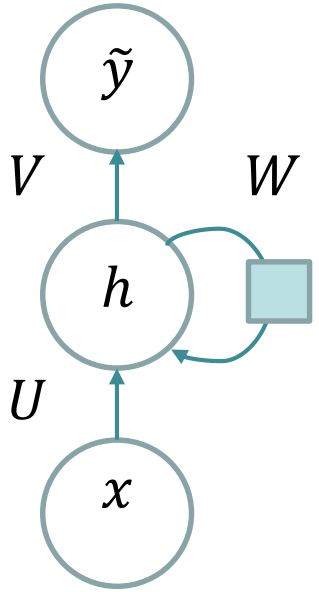
Метод обратного распространения ошибки с разворачиванием сети во времени (2)

- ***Прямой проход по развернутой во времени сети*** (проход слева направо по развернутой во времени сети)
  - Выполняется вычисление скрытых состояний и выходов развернутой сети, а также градиентов функций активации.
  - Сложность вычислений пропорциональна длине входной последовательности $O(\tau)$
  - Распараллеливание вычислений выполнить нельзя, поскольку каждое следующее внутреннее состояние системы зависит от предыдущего
- ***Вычисление значения целевой функции и градиента этой функции***
- ***Обратный проход развернутой во времени сети*** (проход справа налево по развернутой во времени сети). Выполняется вычисление ошибки и корректировка весов сети

ГЛУБОКИЕ РЕКУРРЕНТНЫЕ НЕЙРОННЫЕ СЕТИ

Преобразования на рекуррентном слое

- Вычисления в большинстве типовых рекуррентных сетей можно разложить на три блока параметров и соответствующих им преобразования:
  - Преобразование входа в скрытое состояние
  - Преобразование от предыдущего скрытого состояния в следующее скрытое состояние
  - Преобразование скрытого состояния в выход

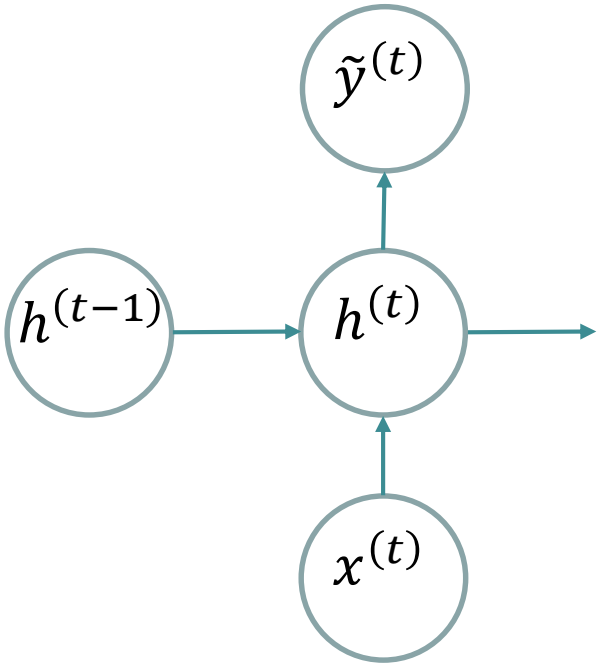


- В зависимости от того, насколько сложными (глубокими) являются преобразования, выделяется несколько типов рекуррентных сетей

Типы рекуррентных сетей (1)

- ***Обычная рекуррентная сеть*** (a conventional RNN)

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



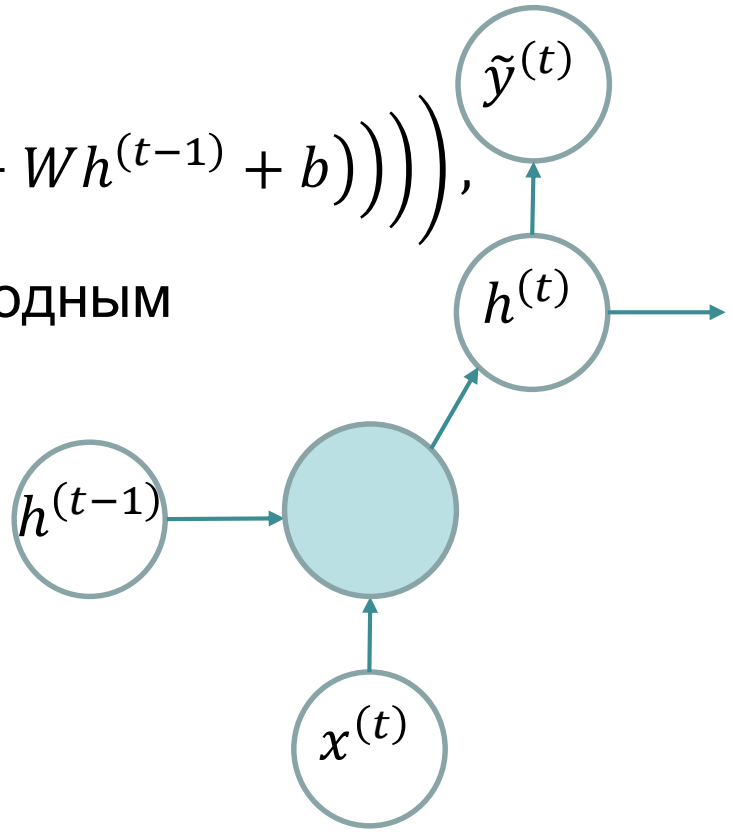
Типы рекуррентных сетей (2)

- ***Рекуррентная сеть с глубоким преобразованием входного сигнала в скрытый*** (Deep Transition RNN, DT-RNN)

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

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

где L – количество слоев сети между входным и скрытым слоями

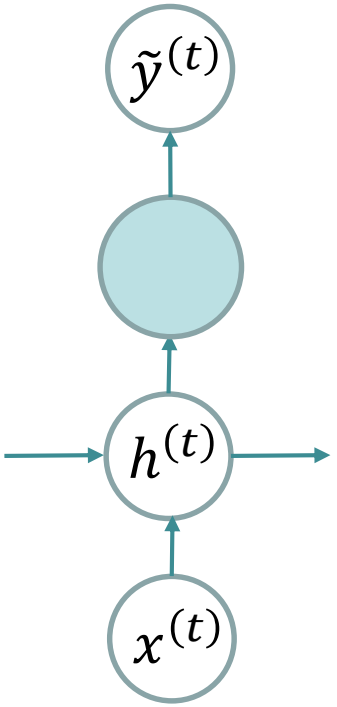


Типы рекуррентных сетей (3)

- ***Рекуррентная сеть с глубоким преобразованием скрытого сигнала в выходной*** (Deep Output RNN, DO-RNN)

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

где L – количество слоев сети между скрытым и выходным слоями

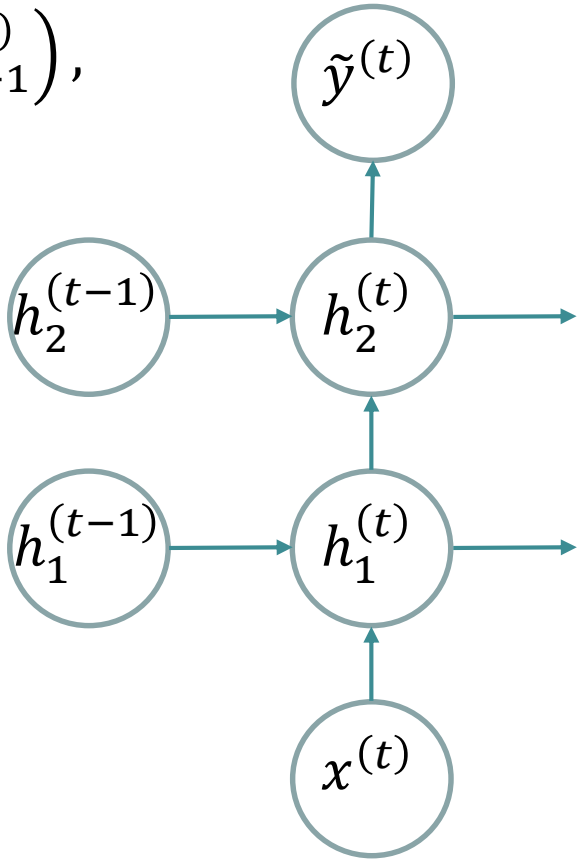


Стек рекуррентных слоев

- Построение стека из обычных рекуррентных сетей – еще один способ формирования глубоких рекуррентных сетей

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

где $h_l^{(t)}$ – скрытое состояние системы на слое с номером l в момент времени t



НЕЙРОННЫЕ СЕТИ ДОЛГОЙ КРАТКОВРЕМЕННОЙ ПАМЯТИ

Описание проблемы

- До настоящего момента рекуррентные нейронные сети обучались посредством их развертывания во времени и применения модифицированного метода обратного распространения ошибки

- ***При наличии достаточно длинных входных последовательностей в процессе обучения сеть «забывает» информацию об удаленных объектах***

- В некоторых случаях возникает необходимость, чтобы сеть «помнила» информацию об объектах, находящихся в начале последовательности

Описание проблемы. Примеры задач компьютерного зрения (1)

- ***Задача распознавания движений человека*** (human action recognition)
  - Задача состоит в том, чтобы на основании последовательности кадров видео определить, какой тип движения совершает человек (сидит, стоит, шагает, бежит, прыгает и др.)
  - Для разных типов движений начальные действия могут быть идентичными, поэтому для принятия решения необходима информация о полной последовательности действий

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

Описание проблемы. Примеры задач компьютерного зрения (2)

- ***Задача семантической сегментации последовательности кадров видео*** (semantic segmentation of videos)
  - Цель – определить класс объекта, которому принадлежит каждый пиксель сцены
  - Видео представляет собой набор связанных кадров
  - В процессе семантической сегментации текущего кадра можно использовать информацию, полученную в ходе сегментации набора предыдущих кадров

Описание проблемы. Примеры задач компьютерного зрения (3)

- ***Формирование словесного описания изображения*** (image captioning)
  - Данная задача находится на стыке областей компьютерного зрения и обработки естественного языка
  - Смысл задачи состоит в том, чтобы построить связное предложение, описывающее содержимое изображения
  - На каждом этапе построения описания делается попытка восстановить следующее слово в предложении на основании контекста изображения

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;

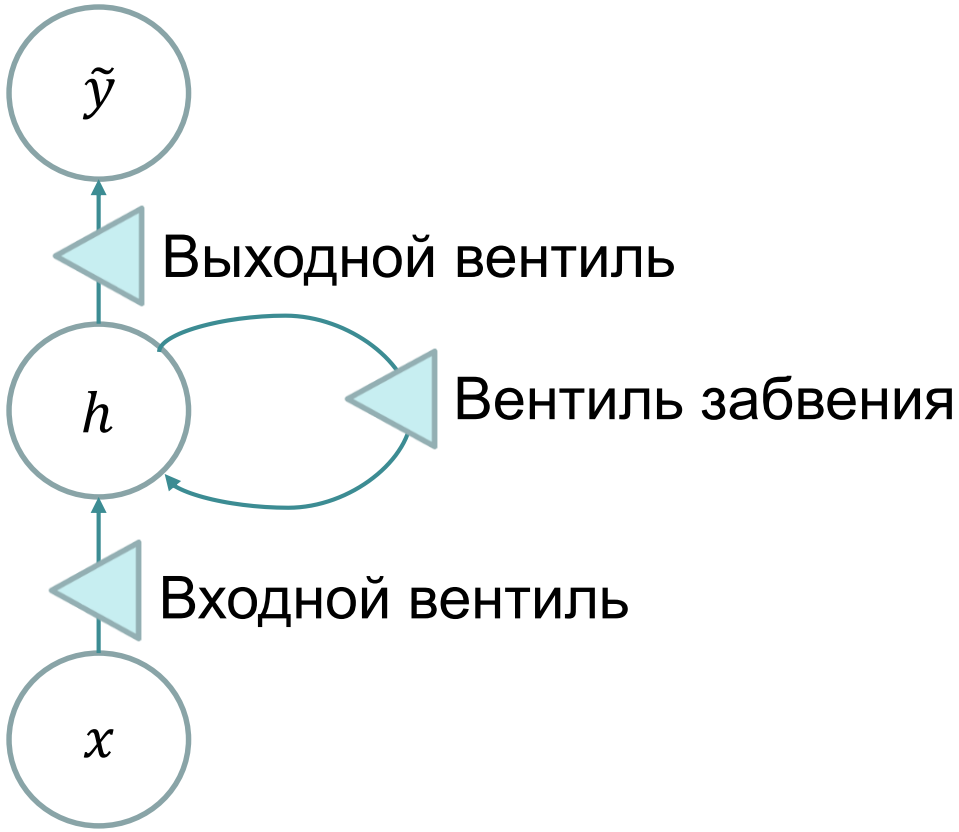
* Mao J., Xu W., Yang Y., Wang J., Huang Z., Yuille A.L. Deep Captioning with Multimodal Recurrent Neural Networks (m-RNN) // ICLR. – 2015. – [https://arxiv.org/pdf/1412.6632.pdf].

Идея работы ячейки с кратковременной памятью (1)

- Общая структура ячейки с долгой кратковременной памятью (long short-term memory, LSTM) предполагает наличие нейронов, имеющих связь на себя
- Данные поступают на вход нейрону и обработанные данные выдаются на выход
- Рекуррентная связь со своим входом имеет вес равный 1
- Если на вход не поступает никаких новых данных, значение нейрона перезаписывается и сохраняется неизменным

Идея работы ячейки с кратковременной памятью (2)

- Для управления данной структурой используются три вентиля, определяющих прохождение сигнала: ***входной вентиль***, ***вентиль забвения*** и ***выходной вентиль***

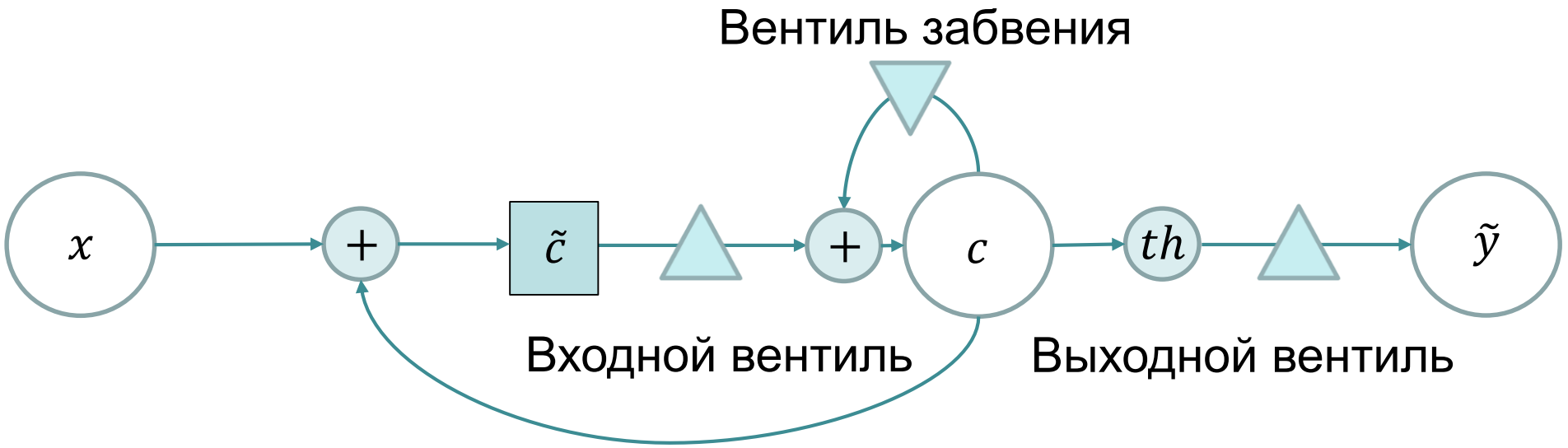


Идея работы ячейки с кратковременной памятью (3)

- Если ***входной вентиль открыт*** (установлен в 1), осуществляется запись входного сигнала в скрытый нейрон, после чего значение записывается и сохраняется в нейроне за счет рекуррентной обратной связи
- Если ***входной вентиль закрыт*** (установлен в 0), значения, поступающие на вход нейрона не влияют на его содержание

- Если необходимо получить значение, сохраненное в ячейке, необходимо открыть выходной вентиль (установить в 1)
- Если значение, содержащееся в ячейке, требуется «забыть», необходимо закрыть вентиль забвения. После этого значение будет стерто из нейрона, и нейрон будет готов для сохранения нового входного значения

Схема построения ячейки с кратковременной памятью

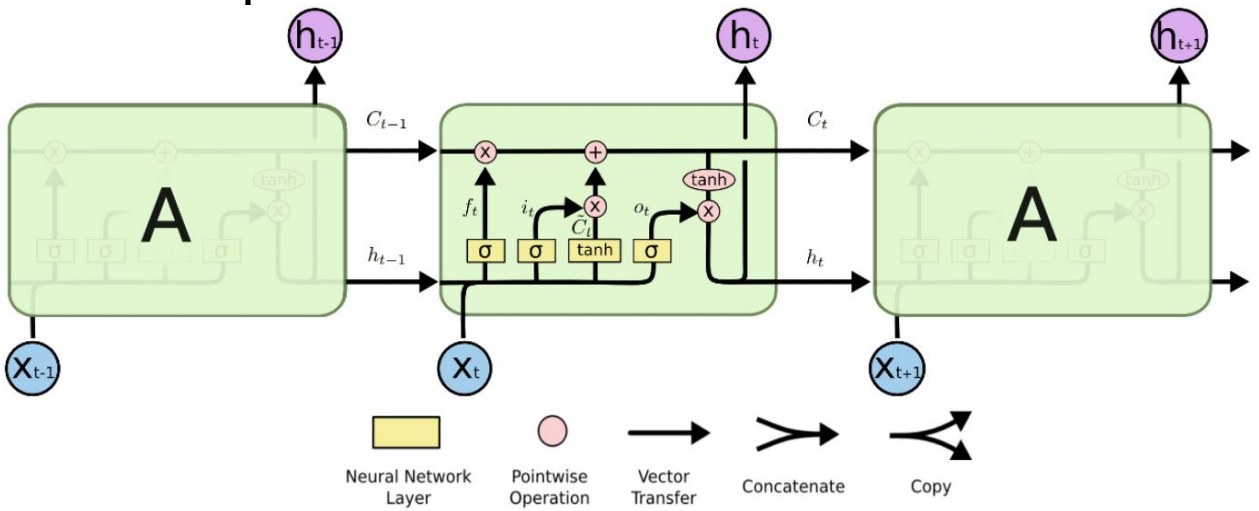


- c – ячейка памяти, $\tilde{c}$ – новое содержимое ячейки памяти
- th – функция активации гиперболический тангенс

* 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].

Сеть долгой кратковременной памяти (1)

- Рассмотрим реализацию ***сети долгой кратковременной памяти*** (long short-term memory network)
- LSTM-сети имеют собственный повторяющийся блок, развернутый во времени



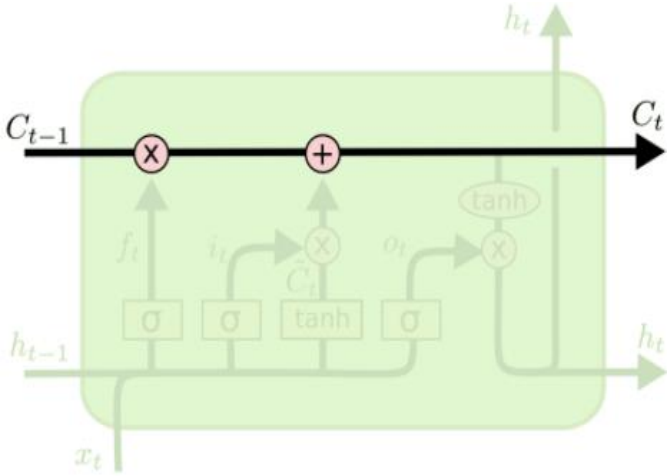
* Hochreiter S., Schmidhuber J. Long short-term memory // Neural Computation. – 1997. – P.1735-1780. – [http://www.bioinf.jku.at/publications/older/2604.pdf].

** 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].

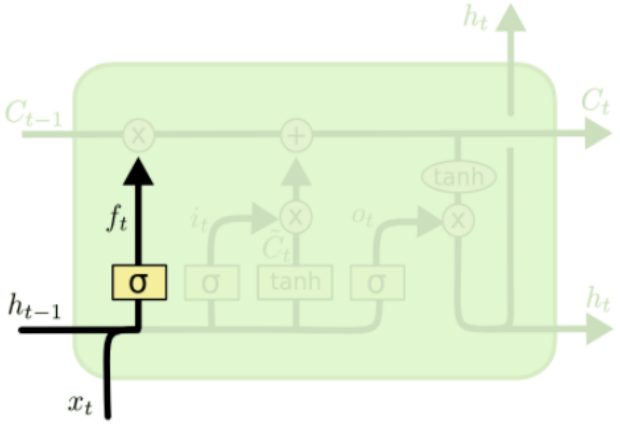
Сеть долгой кратковременной памяти (2)

- Основной составляющей компонентой LSTM-ячейки является ее ***состояние*** C_t, которое передается во времени
- Ячейка способна добавлять или удалять информацию из состояния, тщательно регулируемое структурами, получившими названия ***ворот*** (gates)
- Ворота – это способ передачи информации. Они состоят из сигмоидального слоя и операции поэлементного умножения



Сеть долгой кратковременной памяти (3)

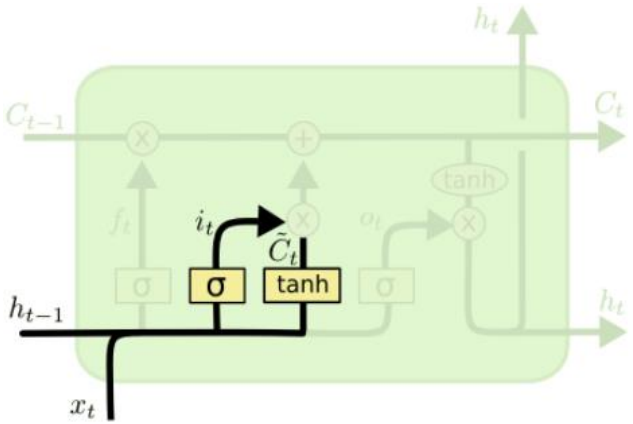
- ***Шаг 1:*** принять решение, какие элементы из состояния необходимо «забыть»
  - Необходимо пройти через ***слой забвения*** (forget gate layer) – сигмоидальный слой сети
  - Определяет веса, с которыми пропускаются элементы состояния
  - Значение 0 означает, что элемент не пропускается, 1 – элемент пропускается полностью



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

Сеть долгой кратковременной памяти (4)

- ***Шаг 2:*** определить, какую новую информацию необходимо сохранить в состоянии ячейки
  - Сигмоидальный слой – ***слой пропускания входов*** (input gate layer) – принимает решение, какие значения необходимо обновить
  - Слой с функцией активацией, соответствующей гиперболическому тангенсу, формирует вектор новых кандидатов, которые добавляются к текущему состоянию

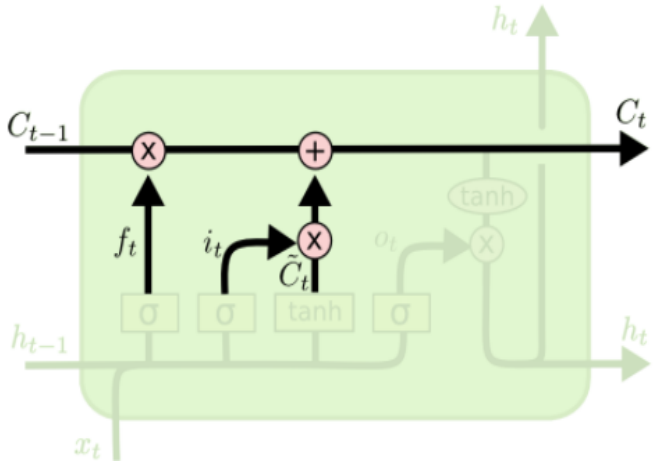


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

Сеть долгой кратковременной памяти (5)

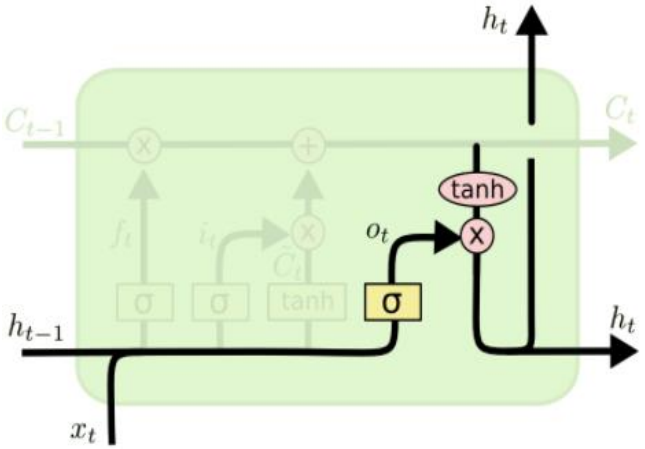
- ***Шаг 3:*** обновить состояние ячейки
  - Необходимо из вектора состояния удалить информацию, для которой принято решение, что ее можно «забыть», и добавить новую информацию



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

Сеть долгой кратковременной памяти (6)

- ***Шаг 4:*** принять решение, что выдать в качестве выхода ячейки
  - Выход строится на основании состояния ячейки и представляет собой его фильтрованную версию
  - Принимается решение, какие части состояния необходимо вывести, посредством введения сигмоидального слоя
  - Далее элементы состояния ячейки нормируются в отрезок $[-1, 1]$ с использованием функции гиперболического тангенса и умножаются на выход сигмоидального слоя



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

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

Примечания

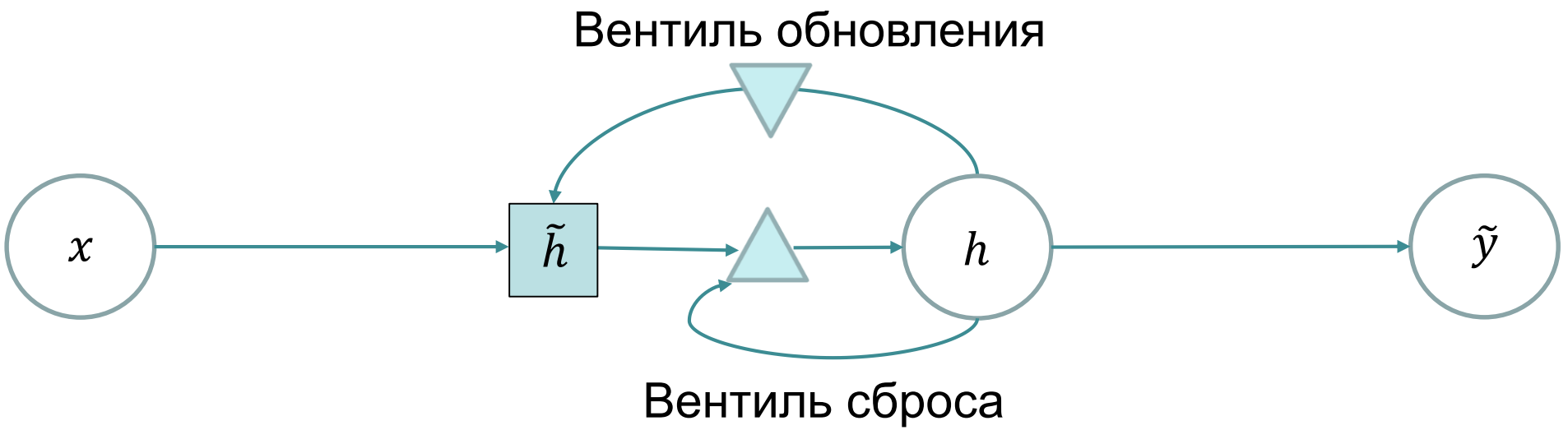
- Приведенная реализация LSTM-ячейки не является единственно возможной, существует множество модификаций

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

УПРАВЛЯЕМЫЕ РЕКУРРЕНТНЫЕ НЕЙРОНЫ

Управляемые рекуррентные нейроны (1)

- Упрощение ячейки долгой кратковременной памяти – ***управляемый рекуррентный нейрон*** (Gated Recurrent Unit, GRU)
- Схема построения управляемого рекуррентного нейрона:



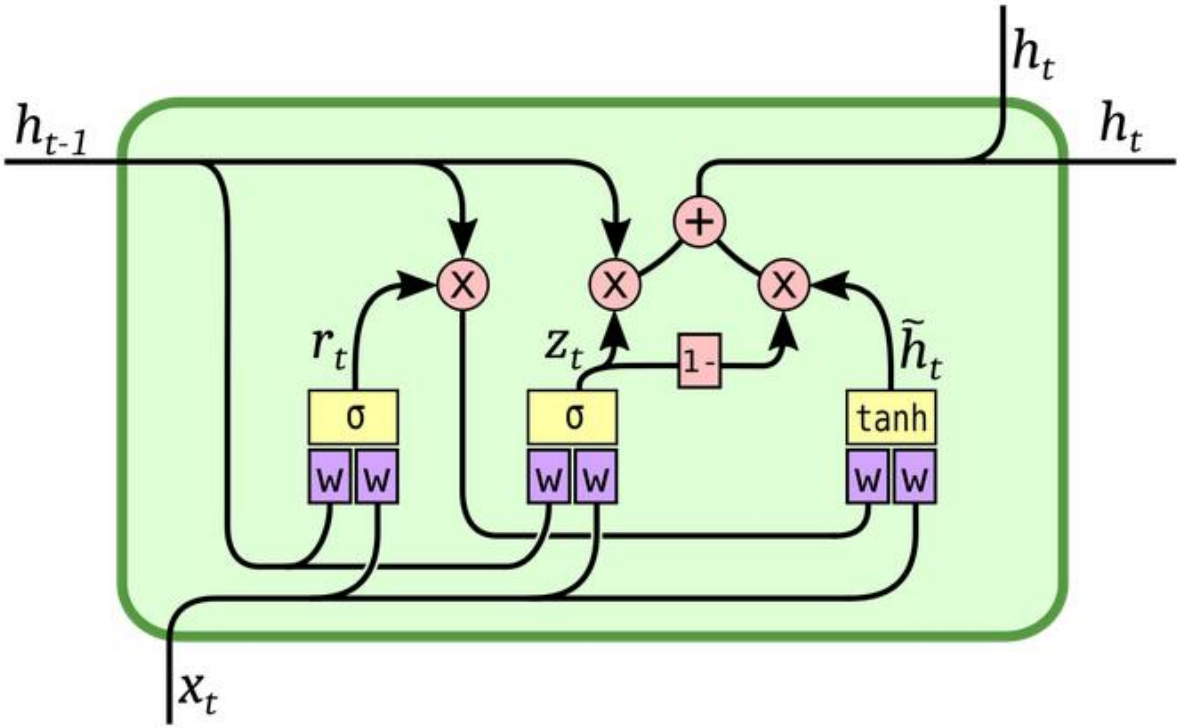
- h – активация, $\tilde{h}$ – кандидат на значение активации

* 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].

Управляемые рекуррентные нейроны (2)

- Управляемый рекуррентный нейрон содержит на один вентиль меньше, чем ячейка долгой кратковременной памяти
- Вентиль обновления определяет объем информации, получаемый из прошлого состояния
- Вентиль сброса работает по аналогии с вентилем забвения в ячейке долгой кратковременной памяти

Реализация управляемого рекуррентного нейрона



- h_{t-1}, h_t – состояние скрытого нейрона, x_t – входной сигнал
- r_t реализует вентиль сбрасывания, z_t – вентиль обновления

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

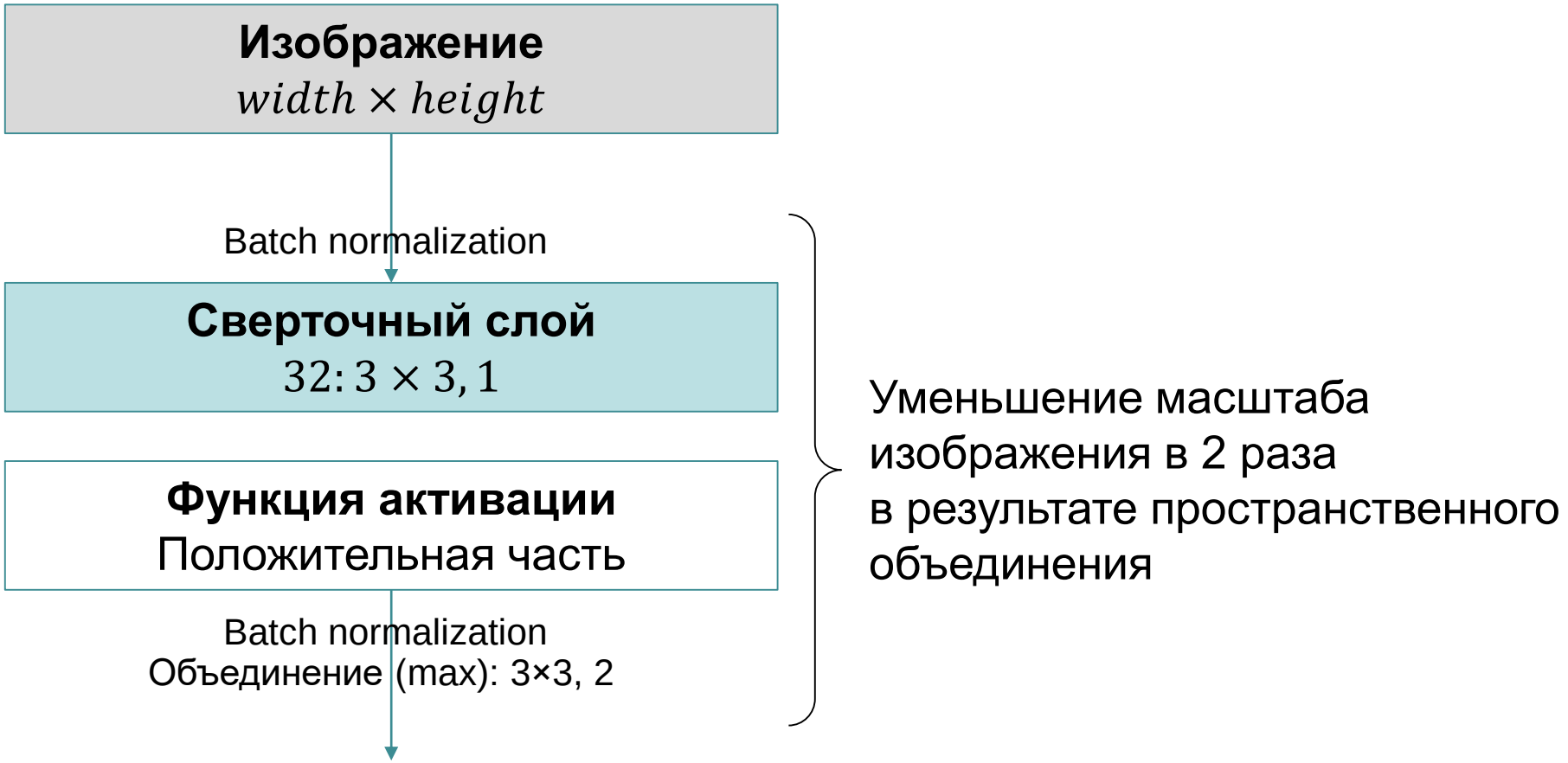
ПРИМЕР ПОСТРОЕНИЯ РЕКУРРЕНТНЫХ СЕТЕЙ В INTEL® NEON™ FRAMEWORK

Пример построения рекуррентной сети

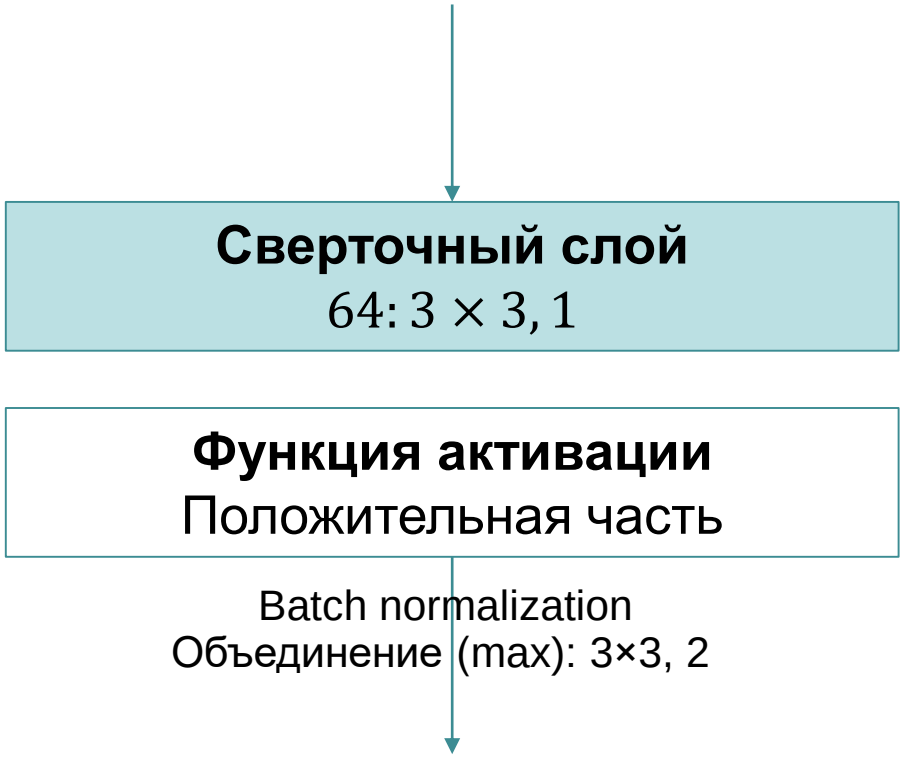
- Рассматривается задача классификации пола человека по фотографии
- Приведенный пример демонстрирует применение рекуррентных сетей для неклассических входных данных, которые в явном виде не представляют собой последовательность однотипных элементов

- Разрабатываемая структура рекуррентного блока описана в следующей работе:
  - 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]

Архитектура рекуррентной сети (1)

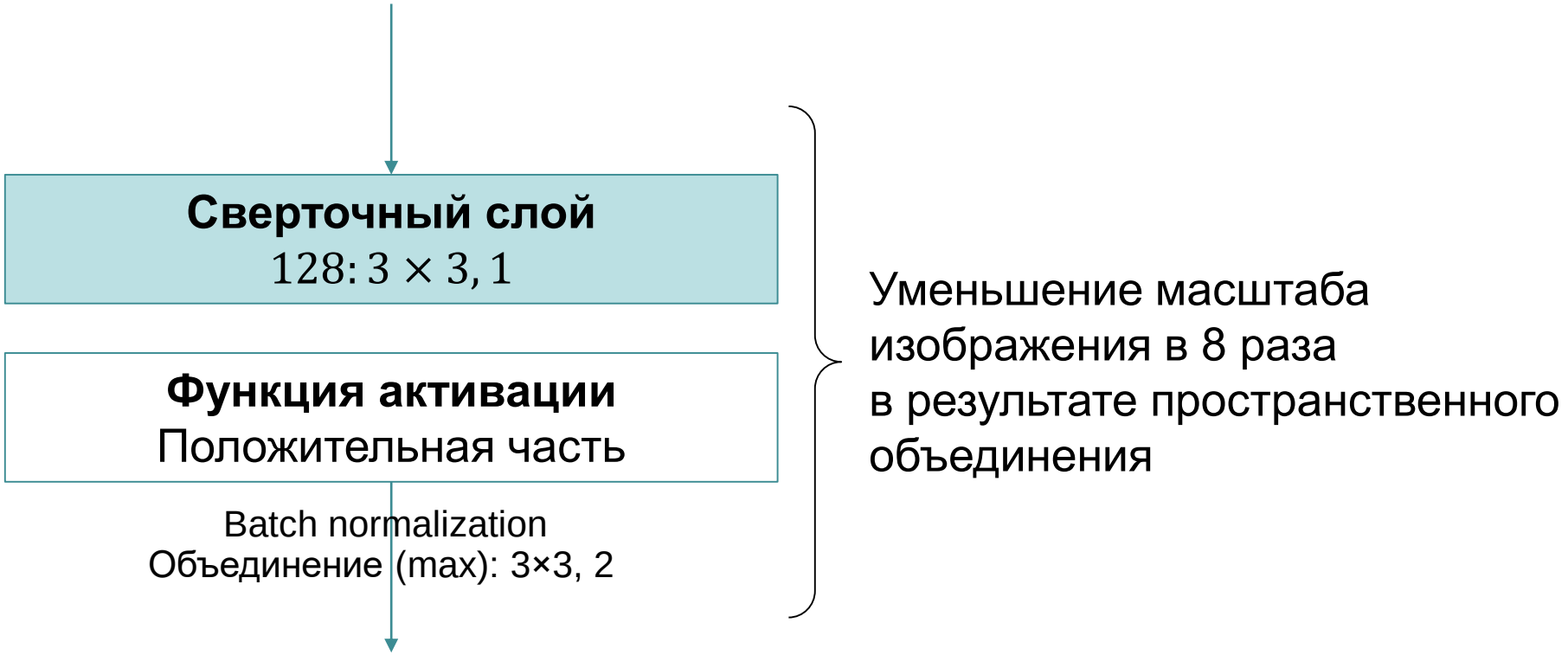


Архитектура рекуррентной сети (2)

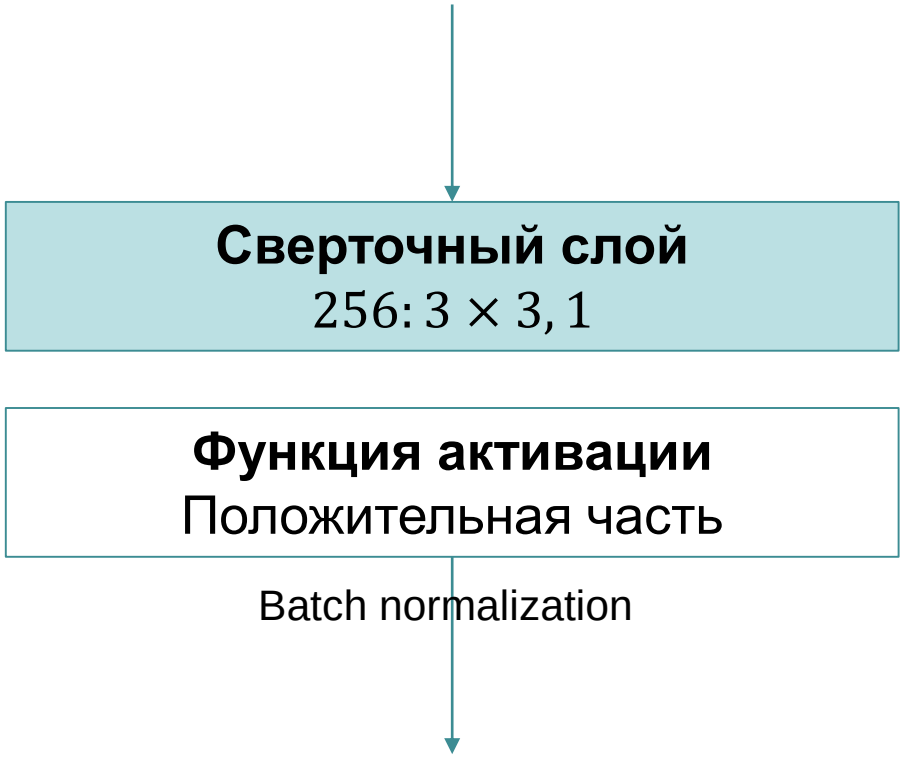


Уменьшение масштаба изображения в 4 раза в результате пространственного объединения

Архитектура рекуррентной сети (3)

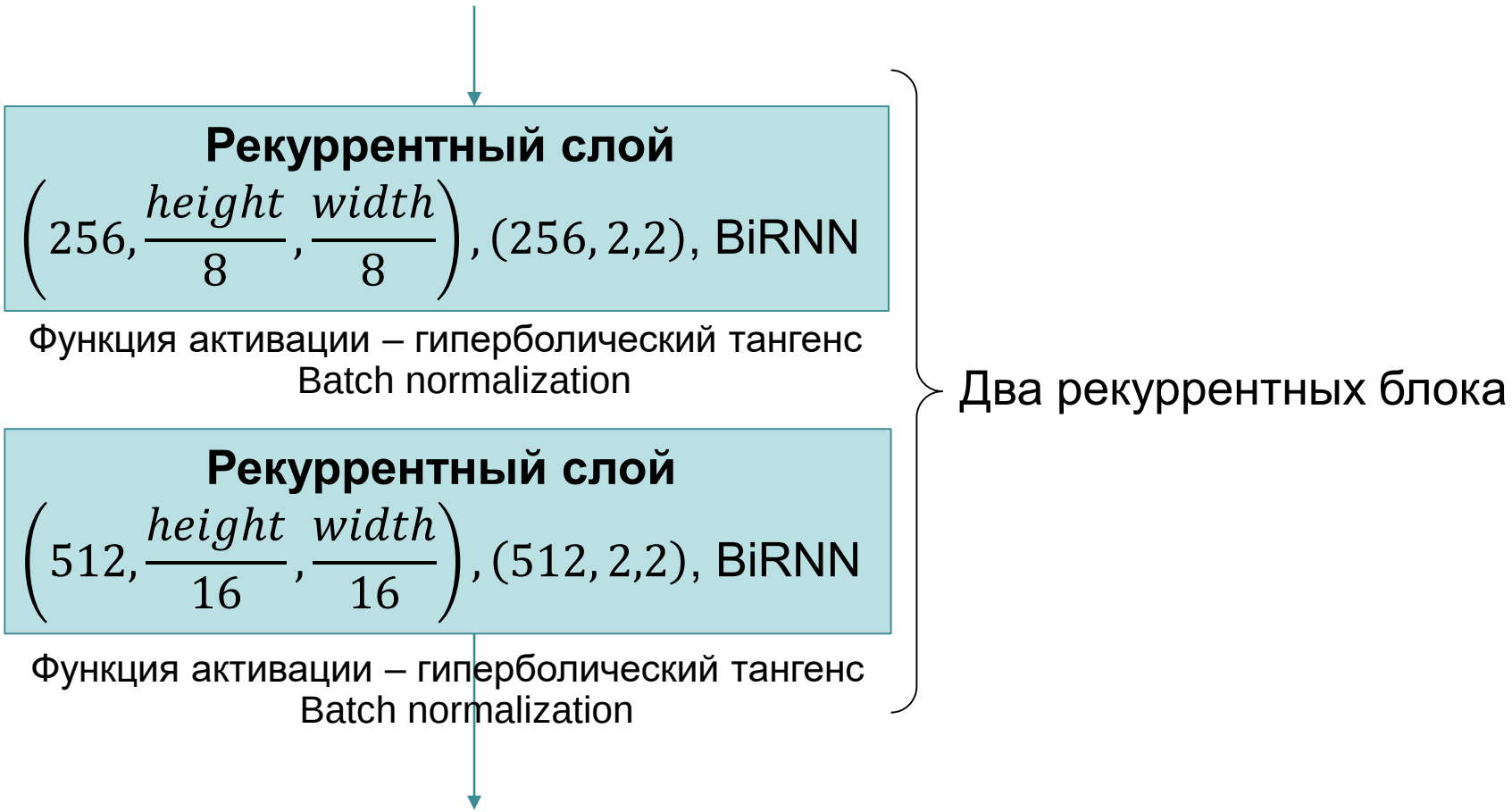


Архитектура рекуррентной сети (4)

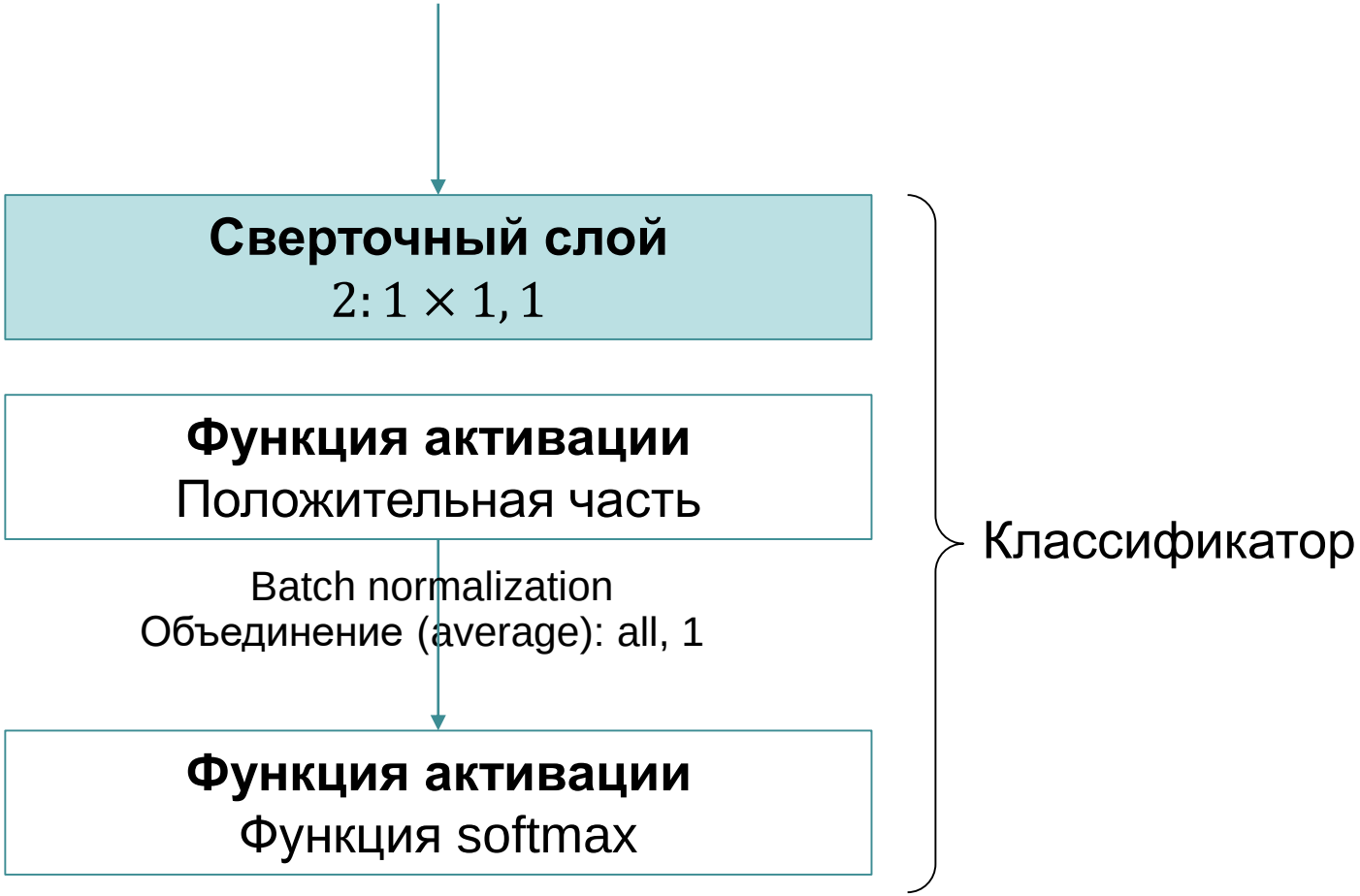


Размер карты признаков
на входе сверточного слоя
и на выходе отличается только
по третьему измерению

Архитектура рекуррентной сети (5)

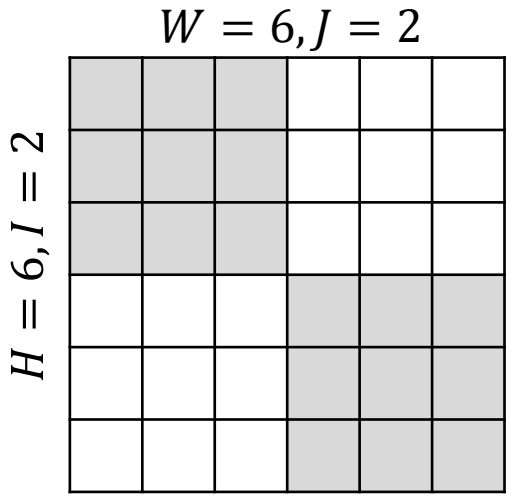


Архитектура рекуррентной сети (6)



Структура рекуррентного блока (1)

- Пусть $H \times W \times C$ – размер карты признаков, поступающей на вход рекуррентного блока, H – высота, W – ширина, C – количество каналов
- Разделим карту признаков на $I \times J$ блоков, каждый блок содержит вектора, состоящие из C элементов
- Обозначим каждый блок как $p_{ij} \in \mathbb{R}^{H_p \times W_p \times C}$



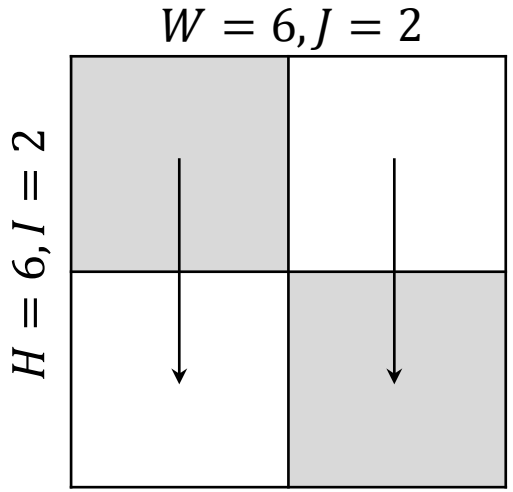
* Visin F., Ciccone M., Romero A., Kastner K., Cho K., Bengio Y., Matteucci M., Courville A. ReSeg: A Recurrent Neural Network-based Model for Semantic Segmentation // In CVPR Deep Vision Workshop, 2016. – 2016. – [https://arxiv.org/abs/1511.07053].

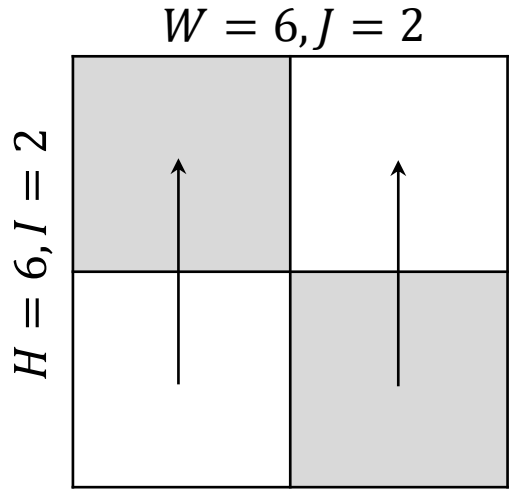
Структура рекуррентного блока (2)

- Реализуем две двунаправленные рекуррентные сети по столбцам и строкам матриц блоков, построенных на картах признаков:
  - Первая двунаправленная сеть состоит из двух рекуррентных слоев:
    - Первый слой соответствует обходу блоков сверху-вниз
    - Второй слой соответствует обходу блоков снизу-вверх
  - Вторая двунаправленная сеть состоит из двух рекуррентных слоев:
    - Первый слой соответствует обходу блоков слева-направо
    - Второй слой соответствует обходу блоков справа-налево

Структура рекуррентного блока (3)

- Схема зависимости элементов первой двунаправленной сети выглядит следующим образом:





- Вычисление элементов новой карты признаков выполняется согласно уравнениям:

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

Структура рекуррентного блока (4)

- Карта признаков на выходе первого рекуррентного слоя формируется посредством конкатенации признаков, посчитанных при обходе блоков сверху-вниз и снизу-вверх

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

где U – количество элементов в каждом рекуррентном слое

- Вторая двунаправленная сеть получает на вход карту признаков $O^{\updownarrow}$ и строится аналогичным образом, но между блоками устанавливаются горизонтальные зависимости

* 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].

Пример применения рекуррентных сетей к задаче предсказания пола человека по фотографии (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()),
...
```

Пример применения рекуррентных сетей к задаче предсказания пола человека по фотографии (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()),
```

Пример применения рекуррентных сетей к задаче предсказания пола человека по фотографии (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
```

Пример применения рекуррентных сетей к задаче предсказания пола человека по фотографии (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())    ]
```

Пример применения рекуррентных сетей к задаче предсказания пола человека по фотографии (5)

```
    model = Model(layers=layers)

    cost = GeneralizedCost(costfunc=CrossEntropyMulti())

    return (model, cost)
```

Тестовая инфраструктура

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

Результаты экспериментов

Название	Параметры обучения	Точность, %	Время обучения, с
RNN	batch_size = 128 epoch_count = 90 backend = gpu GradientDescentMomentum(0.01, momentum_coef=0.9, wdecay=0.0005)	81.9	29571

Сводные результаты экспериментов

Название	Точность, %	Время обучения, с
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 эпох)	**81.3**	15127
ResNet-50 (30 эпох)	80.9	11849
TL-1	85.6	119975
TL-2	85.3	119989
TL-3	**86.3**	39282
RNN	**81.9**	29571

Заключение

- Сложность построения рекуррентных сетей достаточно высокая, особенно в задачах, где отсутствуют явные последовательности входных данных
- Основная область применения рекуррентных сетей – задачи обработки естественного языка
- В настоящее время развивается применение рекуррентных сетей при решении задач анализа и обработки изображений
  - 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]

Основная литература

- Хайкин С. Нейронные сети. Полный курс. – М.: Издательский дом «Вильямс». – 2006. – 1104 с.
- Осовский С. Нейронные сети для обработки информации. – М.: Финансы и статистика. – 2002. – 344 с.
- Goodfellow I., Bengio Y., Courville A. Deep Learning. – MIT Press. – 2016. – [http://www.deeplearningbook.org].

Авторский коллектив

- **Кустикова Валентина Дмитриевна**
  к.т.н., ст.преп. каф. МОСТ ИИТММ,
  ННГУ им. Н.И. Лобачевского
  valentina.kustikova@itmm.unn.ru
- **Жильцов Максим Сергеевич**
  магистрант каф. МОСТ ИИТММ,
  ННГУ им. Н.И. Лобачевского
  zhiltsov.max35@gmail.com
- **Золотых Николай Юрьевич**
  д.ф.-м.н., проф. каф. АГДМ ИИТММ,
  ННГУ им. Н.И. Лобачевского
  nikolai.zolotykh@itmm.unn.ru