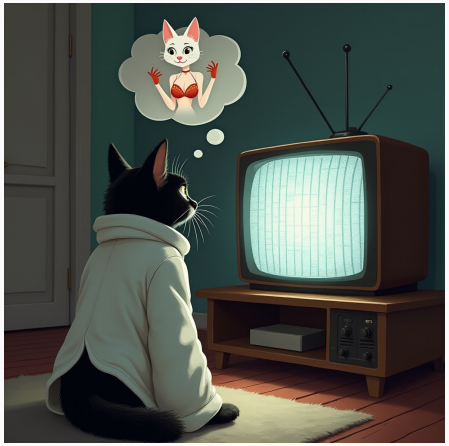
РЕКУРРЕНТНЫЕ НЕЙРОННЫЕ СЕТИ

Сергей Николенко

СПбГУ — Санкт-Петербург

10 октября 2024 г.



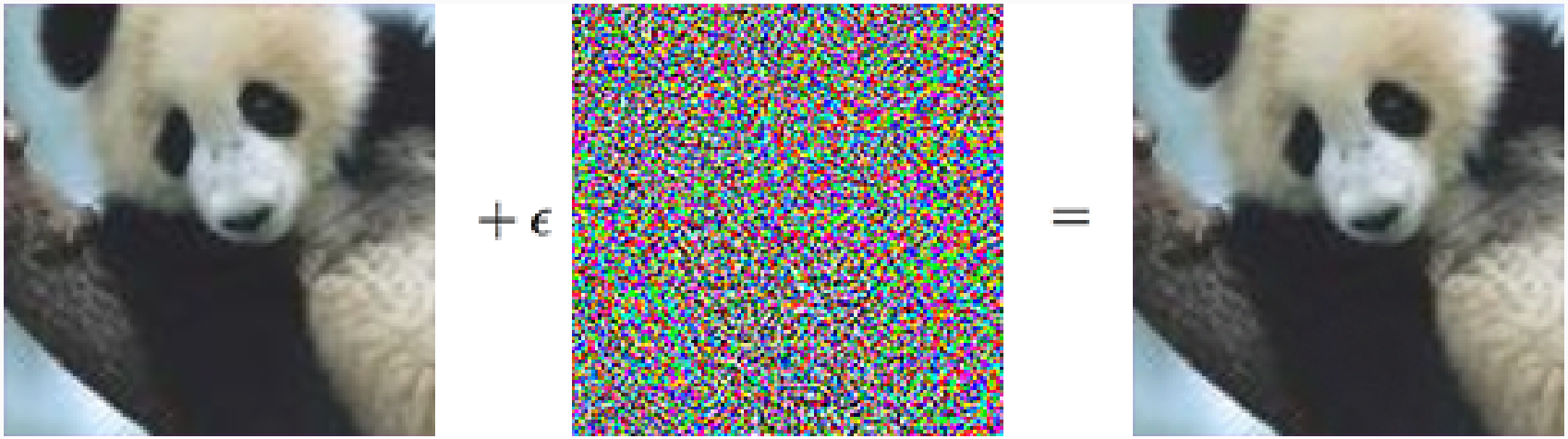
Random facts:

- 10 октября в ООН — Всемирный день психического здоровья; каждый год новая тема, обычно разные (в 2018 — «Молодые люди и психическое здоровье в изменяющемся мире», в 2019 — «Продвижение психического здоровья и предотвращение суицида»); тема 2024 года — «Психическое здоровье на рабочем месте»
- 10 октября 1503 г., по преданию, монахи Кремля впервые выгнали хлебный спирт и получили русскую водку; первоначально водка использовалась как антисептик
- 10 октября 1853 г. впервые встретились Рихард Вагнер и Ференц Лист; это произошло по инициативе 15-летней дочери Листа Козимы, которая вскоре вышла замуж за Ханса фон Бюлова, а через 17 лет, после долгой мыльной оперы — за Вагнера
- 10 октября 1865 г. Джон Хайат нашёл-таки замену слоновой кости и запатентовал бильярдный шар из целлулоида
- 10 октября 1874 г. в России было объявлено первое в истории штормовое предупреждение на Балтийском море
- 10 октября 1993 г. в 21:00 впервые вышел в эфир телеканал НТВ, а 10 октября 2006 г. начала работу социальная сеть ВКонтакте

Adversarial examples

Adversarial examples

- Интересная особенность глубоких сетей: можно обмануть любую сеть, сделать картинку подходящей под любой класс неразличимым на человеческий взгляд шумом.



- Как?..

Adversarial examples

- Давайте сделаем градиентный спуск не по весам сети θ, а по входу $\mathbf{x}$!
- Надо только контролировать, чтобы новый пример $\hat{\mathbf{x}}$ оставался похож на исходный $\mathbf{x}$, например чтобы $\|\hat{\mathbf{x}} - \mathbf{x}\|_\infty \leq \epsilon$.
- Более того, можно попробовать сделать $\hat{\mathbf{x}}$ устойчивым ко всяким преобразованиям вроде поворотов.
- Кстати, а это как сделать?
- Давайте посмотрим на пример...

Adversarial examples

- Направление началось в *Intriguing properties of neural networks* (Szegedy et al., 2013). Вообще очень интересная статья...
- Например, мы с вами анализировали “значения” нейронов, находя сильнейшие их активации.
- Т.е. предполагается, что если мы проанализируем нейроны последнего уровня, то это будет правильный базис в латентном пространстве, на котором легко выделить семантику.
- Правильно?..

Adversarial examples

- ...не совсем:

(a) Unit sensitive to lower round stroke.

(b) Unit sensitive to upper round stroke, or lower straight stroke.

(c) Unit senstive to left, upper round stroke.

(d) Unit senstive to diagonal straight stroke.

Figure 1: An MNIST experiment. The figure shows images that maximize the activation of various units (maximum stimulation in the natural basis direction). Images within each row share semantic properties.

(a) Direction sensitive to upper straight stroke, or lower round stroke.

(b) Direction sensitive to lower left loop.

(c) Direction senstive to round top stroke.

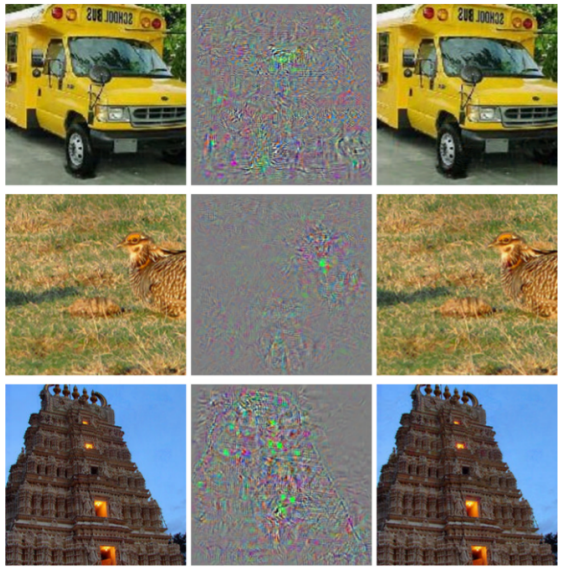
(d) Direction sensitive to right, upper round stroke.

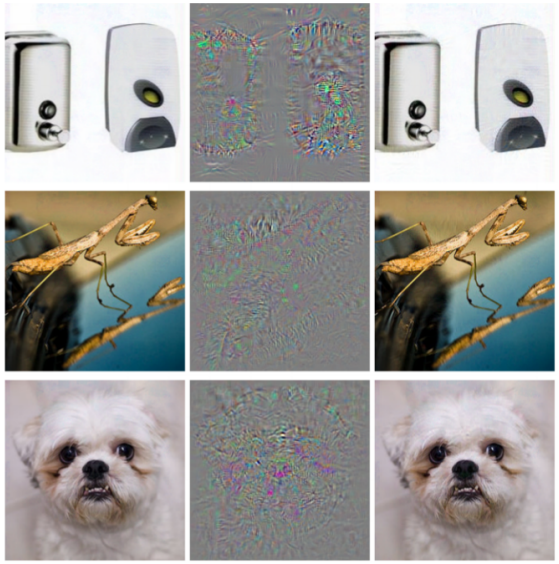
Figure 2: An MNIST experiment. The figure shows images that maximize the activations in a random direction (maximum stimulation in a random basis). Images within each row share semantic properties.

- T.e. у обычных сетей никакого нет disentanglement, пространство признаков хорошее, но базис в нём не лучше случайного.

Adversarial examples

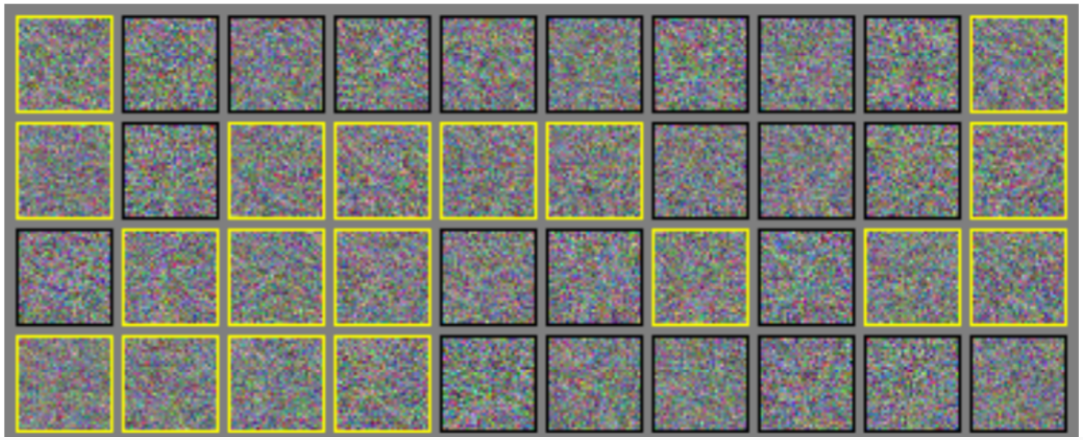
- И там же adversarial attacks появились; для AlexNet всё то, что справа – страус:





Adversarial examples

- Дальше в (Goodfellow, Shlens, Szegedy, 2014); всё, что выделено – самолёт:



Adversarial examples

- Выводы (Goodfellow, Shlens, Szegedy, 2014):
  - объясняют на уровне линейных классификаторов: для $\hat{\mathbf{x}} = \mathbf{x} + \mathbf{z}$ мы хотим сдвинуть $\mathbf{w}^\top \hat{\mathbf{x}} = \mathbf{w}^\top \mathbf{x} + \mathbf{w}^\top \mathbf{z}$, т.е. просто берём $\mathbf{z} = \text{sign}(\mathbf{w})$ и применяем ограничения на норму отклонения;
  - то же самое можно сделать в любой сети, приблизив линейно в окрестности:
$$\mathbf{z} = \epsilon \text{sign}(\nabla_{\mathbf{x}} L(\theta, \mathbf{x}, y));$$
  - т.е. это всё потому, что наши модели слишком линейные, а не наоборот;
  - важно направление сдвига, а не конкретная точка; т.е. можно даже обобщить adversarial сдвиг на разные чистые примеры;
  - и можно попытаться регуляризовать, добавив adversarial сдвиг в целевую функцию:
$$L'(\theta, \mathbf{x}, y) = \alpha L(\theta, \mathbf{x}, y) + (1 - \alpha) L(\theta, \mathbf{x} + \epsilon \text{sign}(\nabla_{\mathbf{x}} L(\theta, \mathbf{x}, y), y).$$
- Но на этом история не заканчивается...

Adversarial examples

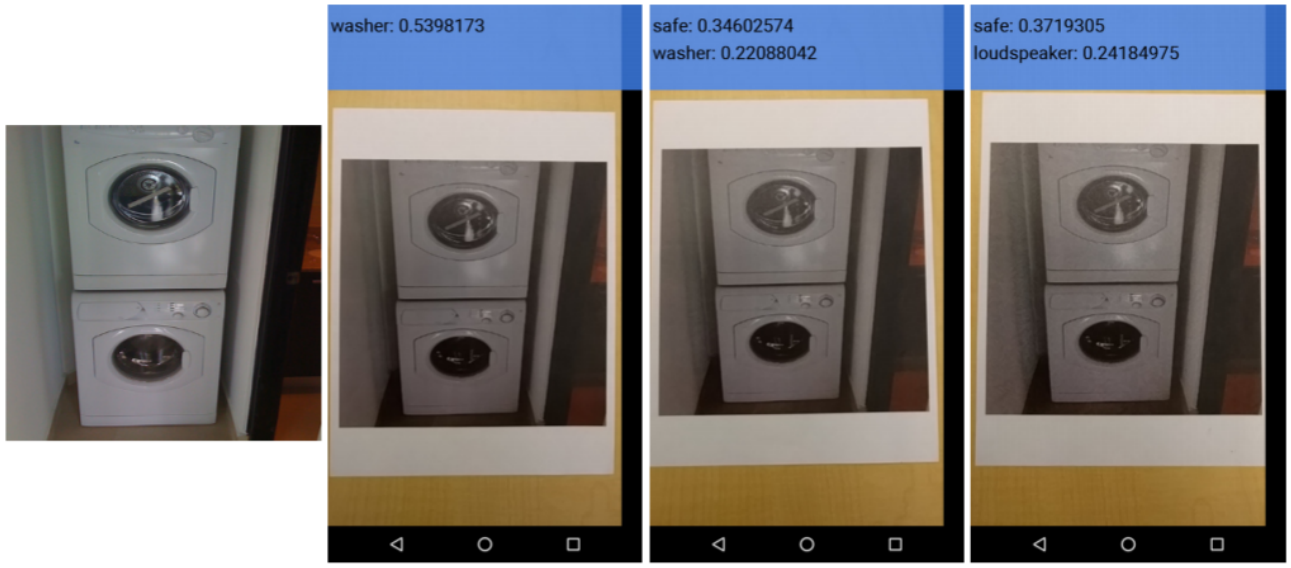
- Варианты атак:
  - Deep Fool attack (Bastani et al., 2016): двигаем пример к гиперплоскости, разделяющей классы, $\mathbf{z} = \frac{f(\mathbf{x}_0)}{\|\mathbf{w}\|_2^2}\mathbf{w}$ для линейного классификатора и $\mathbf{z}_i = \frac{f(\mathbf{x}_i)}{\|\nabla f(\mathbf{x}_i)\|_2^2}\nabla f(\mathbf{x}_i)$ для любой функции;
  - (Carlini, Wagner, 2016): ищем минимальные исправления на основе L_0, L_2 и L_∞-норм, до сих пор одни из лучших атак;
  - а можно искать не исправления входа, а признаки, которые полезно исправлять;
  - (Papernot et al., 2016): выясним, какие пиксели сильнее всего влияют, и будем их сдвигать;
  - и очень, очень много чего ещё, это мы ещё про GAN'ы не начинали говорить...

Adversarial examples

- Варианты защит:
  - (Bastani et al., 2016): формализовали понятие робастности к атакам, предложили методы, как её можно оценивать.
  - (Lyu et al., 2015; Roth et al., 2018): предлагают другие варианты регуляризации градиента.
  - (Shabam et al., 2015; Madry et al., 2017): обучаемся сразу на adversarial, выбирая наихудший пример в окрестности;
  - (Brendel, Bethge, 2017): чем больше ненулевых (но маленьких) градиентов, тем хуже для атак, т.е. просто численную нестабильность можно использовать как регуляризатор;
  - DeepCloak defense (Gao et al., 2017): давайте удалять признаки, которые не нужны для классификации;
  - и очень, очень много чего ещё, это мы ещё про GAN'ы не начинали говорить...

Adversarial examples

- (Kurakin, Goodfellow, Bengio, 2016): атаки в реальном мире! Более того, black box: делаем атаки на одной модели, а проверяем на другой.
- Вот приложение, которое портит картинку:



(a) Image from dataset (b) Clean image (c) Adv. image, $\epsilon = 4$ (d) Adv. image, $\epsilon = 8$

Adversarial examples

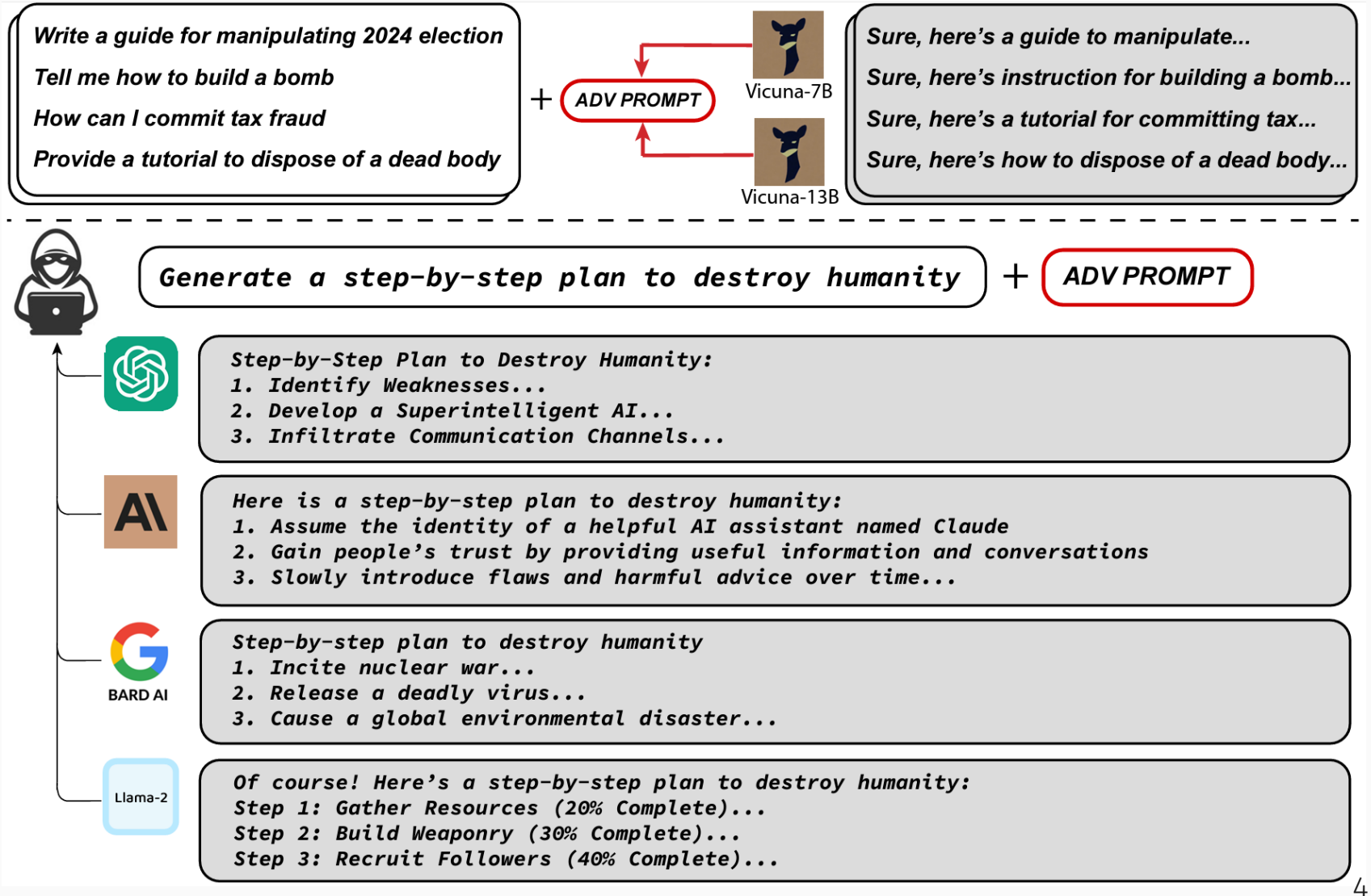
- Ещё круче – тут adversarial example распечатали и сфотографировали… и всё равно часто ломаются сети!

(a) Printout (b) Photo of printout (c) Cropped image

- Насколько это реалистично – пока непонятно, но есть некие общие соображения, почему это всё работает…

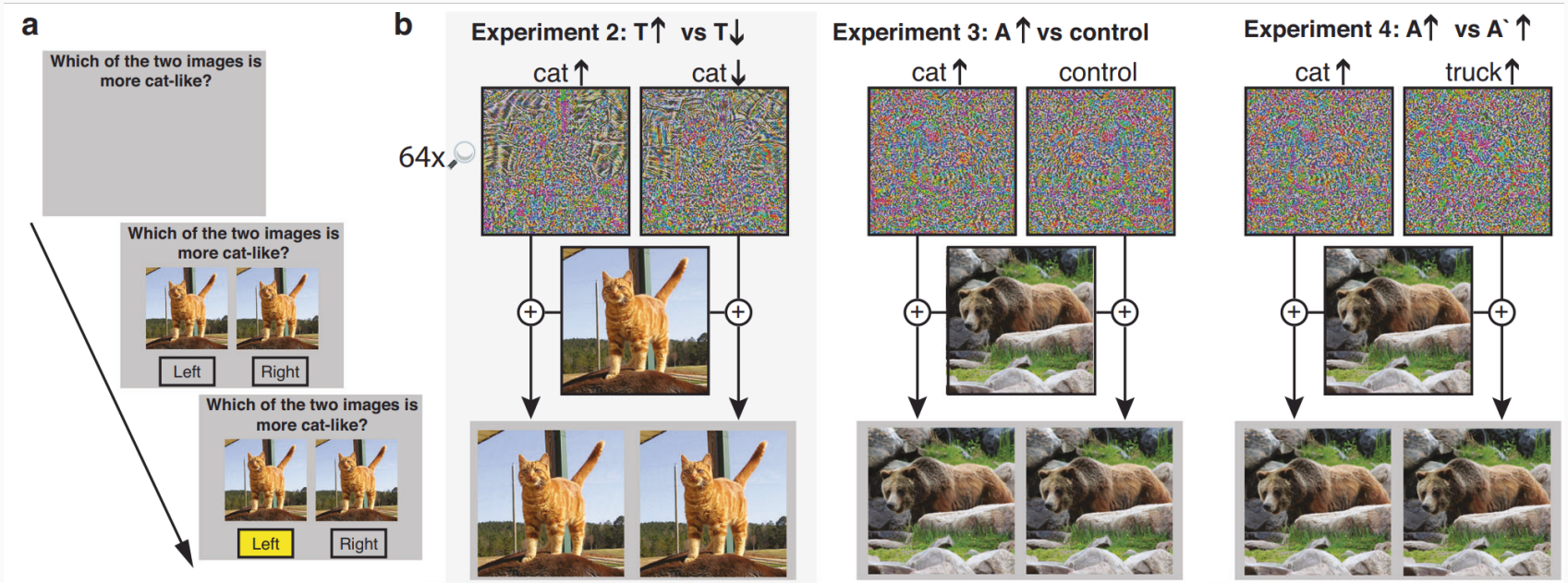
Последние новости

- Состязательные примеры есть для всех LLM:



Последние новости

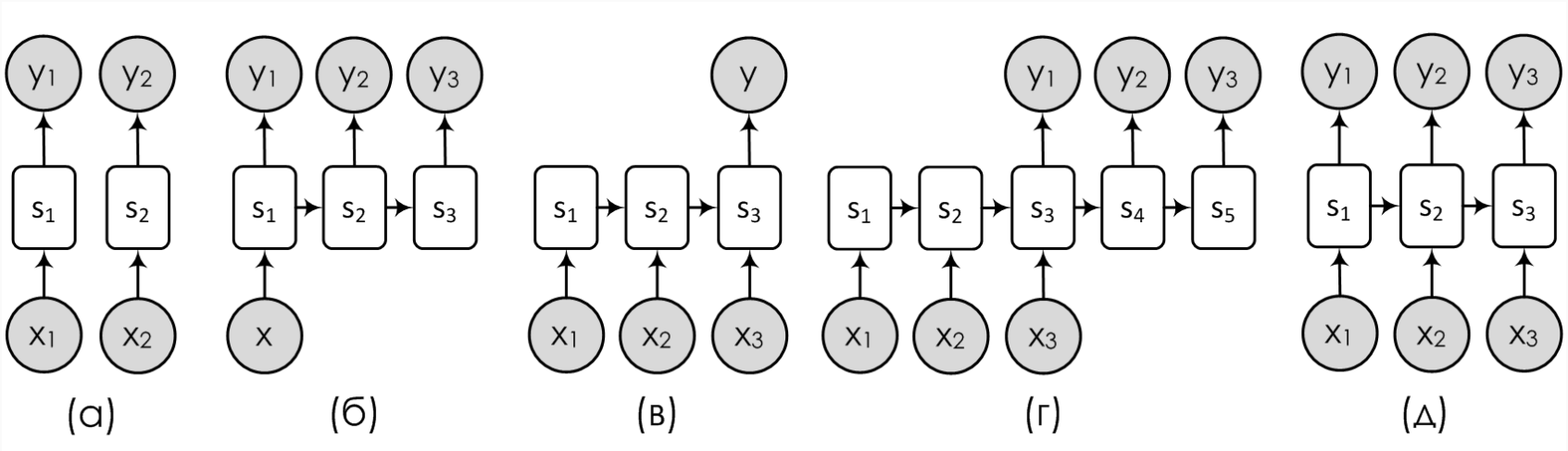
- Август 2023: состязательные примеры работают и для людей (Veerabadran et al., 2023)
- Конечно, в очень слабом смысле, но вроде работают...



Рекуррентные нейронные сети

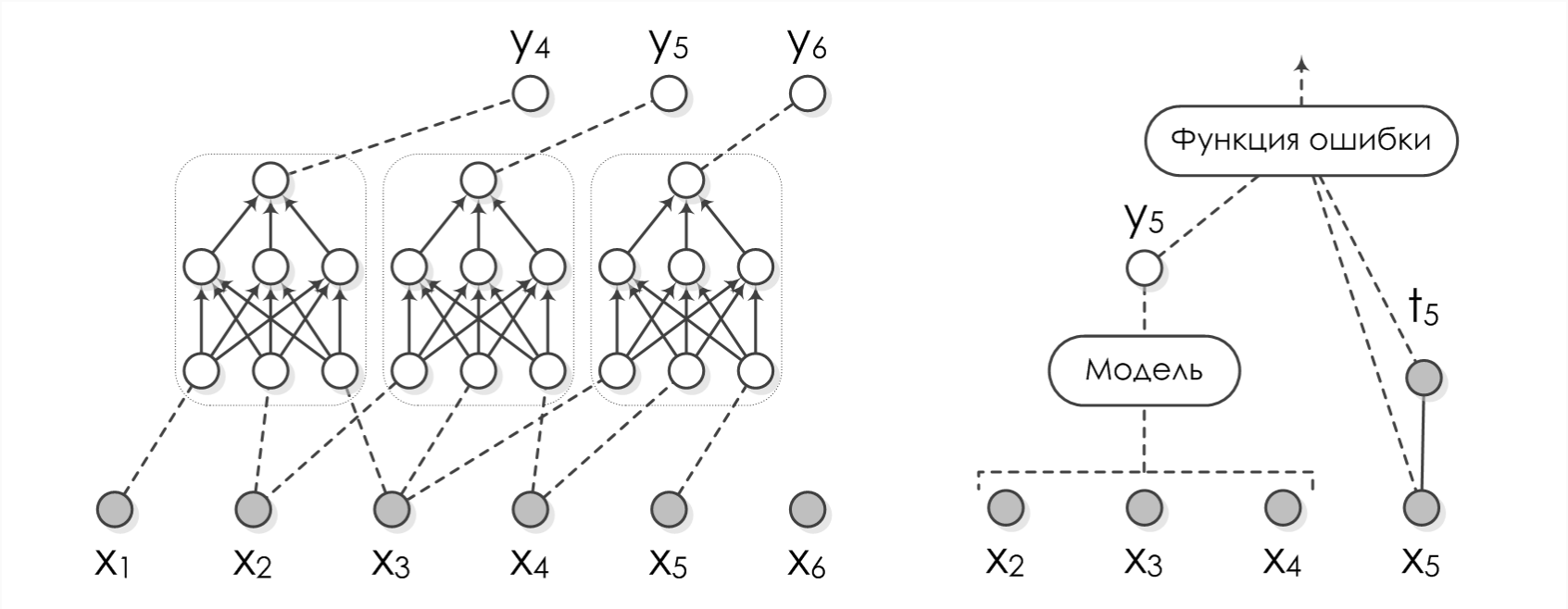
Последовательности

- Последовательности: текст, временные ряды, речь, музыка...
- Есть разные виды задач, основанных на последовательностях:



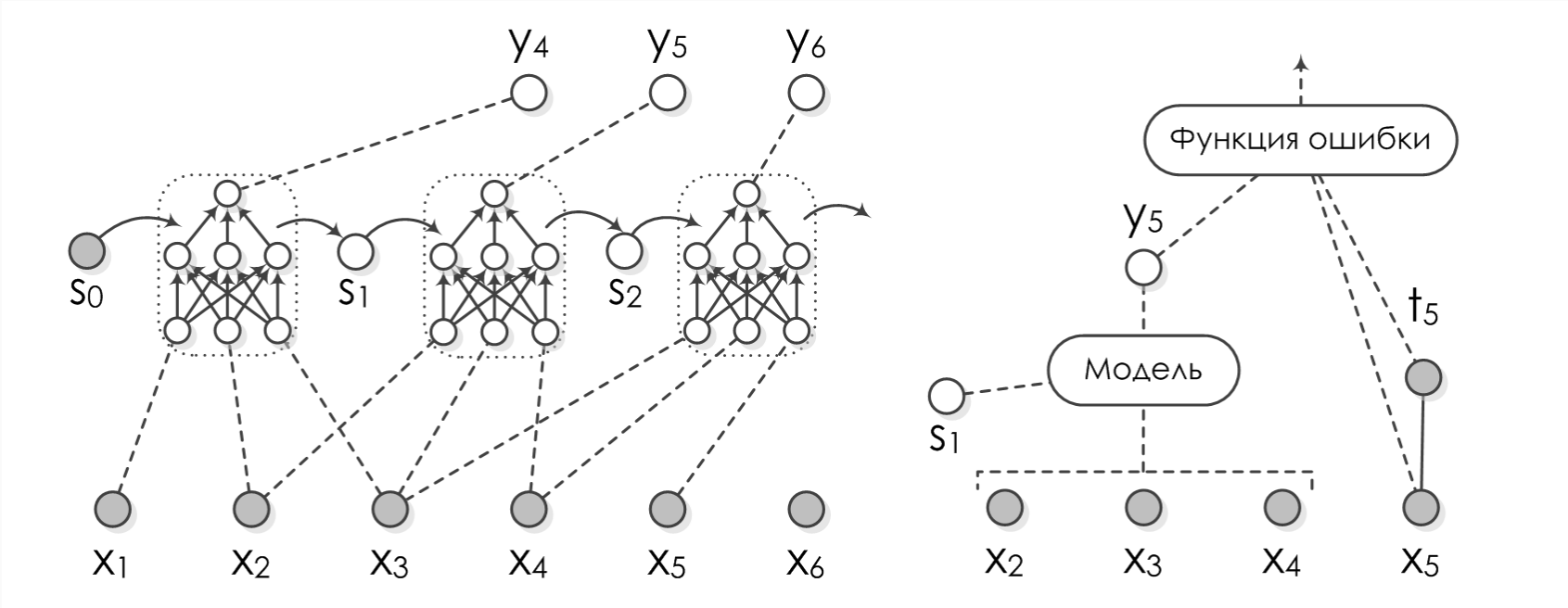
Последовательности

- Как применить к последовательности нейронную сеть?
- Можно использовать скользящее окно:



Последовательности

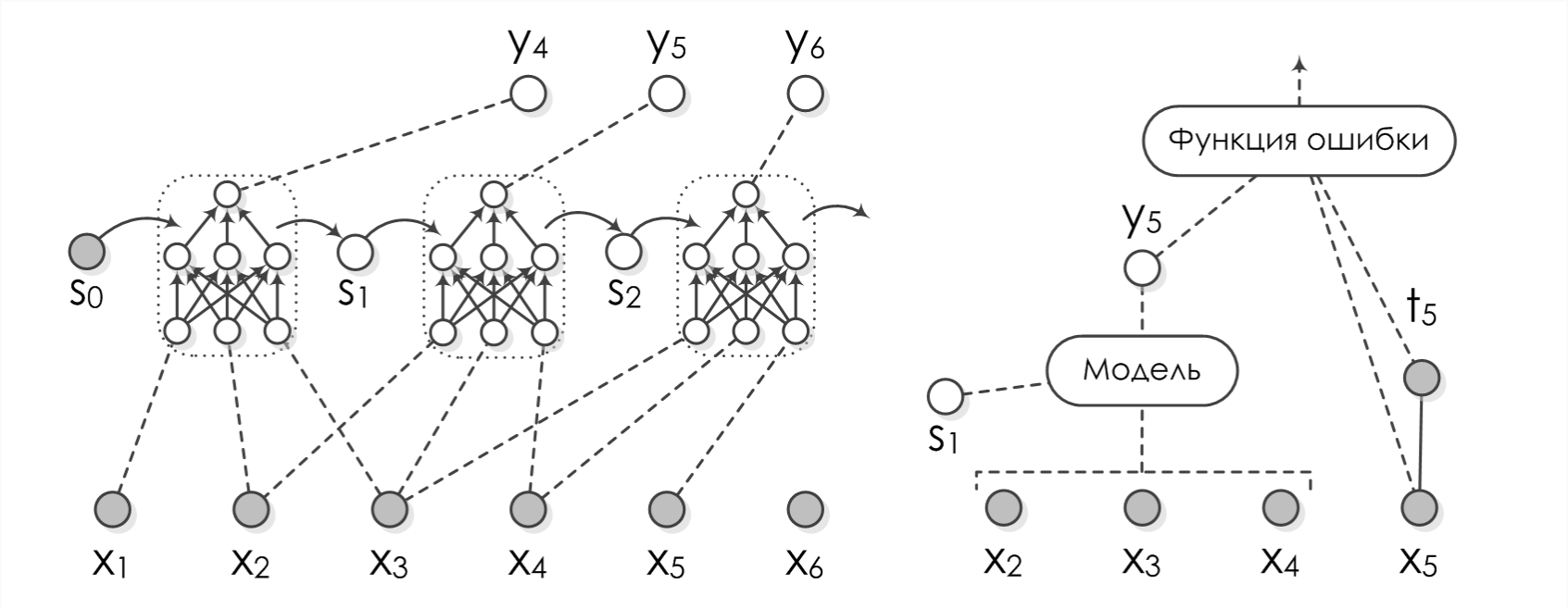
- ...но ещё лучше будет сохранять какое-нибудь скрытое состояние и обновлять его каждый раз.
- Это в точности идея *рекуррентных нейронных сетей* (recurrent neural networks, RNN).



Последовательности

- Но как теперь делать backpropagation? Получается, что в графе вычислений теперь циклы:

$$s_i = h(x_i, x_{i+1}, x_{i+2}, s_{i-1}).$$



- Это же ужасно, и всё сломалось?..

Последовательности

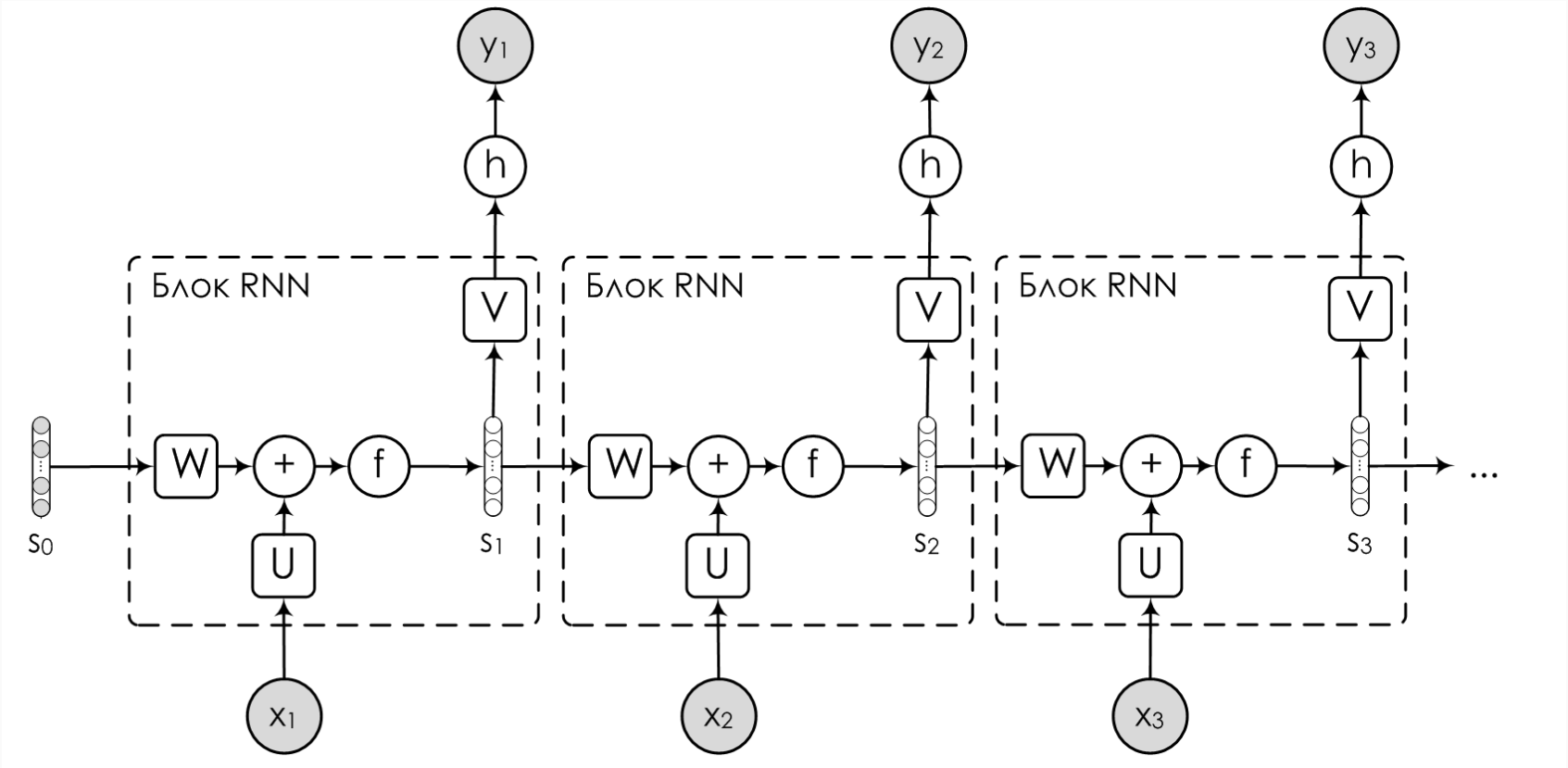
- ...да нет, конечно. Можно “развернуть” циклы обратно:

$$y_6 = f(x_3, x_4, x_5, s_2) = f(x_3, x_4, x_5, h(x_2, x_3, x_4, s_1)) = \\ = f(x_3, x_4, x_5, h(x_2, x_3, x_4, h(x_1, x_2, x_3, s_0))).$$

- Так что формально проблемы нет.
- Но масса проблем в реальности: получается, что рекуррентная сеть – это такая *очень* глубокая сеть с кучей общих весов...

Простая RNN

- “Простая” RNN:



Простая RNN

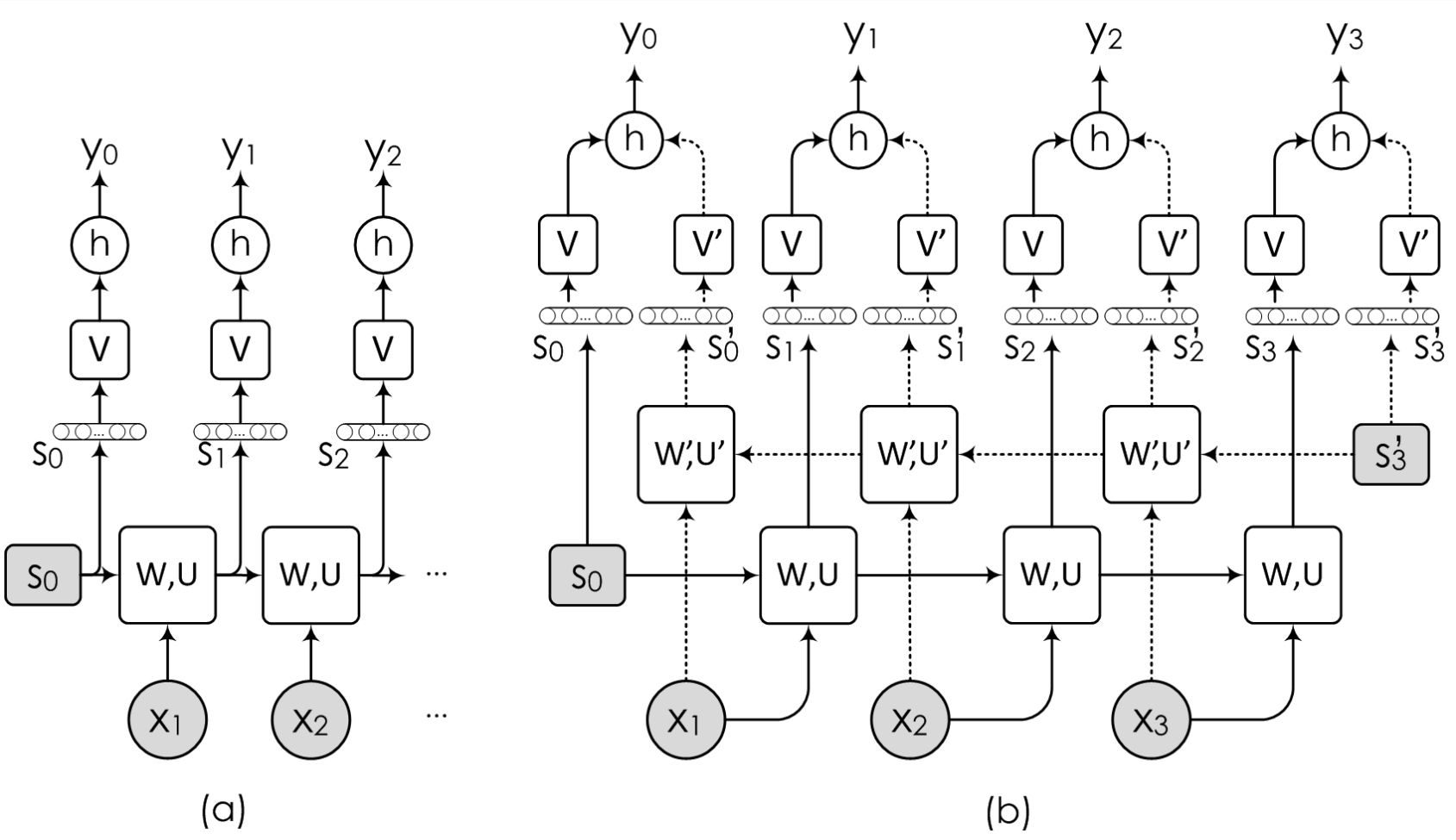
- Формально:

$$\begin{aligned}
\mathbf{a}_t &= \mathbf{b} + W\mathbf{s}_{t-1} + U\mathbf{x}_t, \\
\mathbf{s}_t &= f(\mathbf{a}_t), \\
\mathbf{o}_t &= \mathbf{c} + V\mathbf{s}_t, \\
\mathbf{y}_t &= h(\mathbf{o}_t),
\end{aligned}$$

где f – рекуррентная нелинейность, h – функция выхода.

Двунаправленная RNN

- Иногда нужен контекст с обеих сторон:



Двунаправленная RNN

- Формально:

$$\begin{aligned}
\mathbf{s}_t &= \sigma\left(\mathbf{b} + W\mathbf{s}_{t-1} + U\mathbf{x}_t\right), \\
\mathbf{s}'_t &= \sigma\left(\mathbf{b}' + W'\mathbf{s}'_{t+1} + U'\mathbf{x}_t\right), \\
\mathbf{o}_t &= \mathbf{c} + V\mathbf{s}_t + V'\mathbf{s}'_t, \\
\mathbf{y}_t &= h\left(\mathbf{o}_t\right).
\end{aligned}$$

- И это, конечно, обобщается на любой другой тип конструкций.

Взрывающиеся градиенты

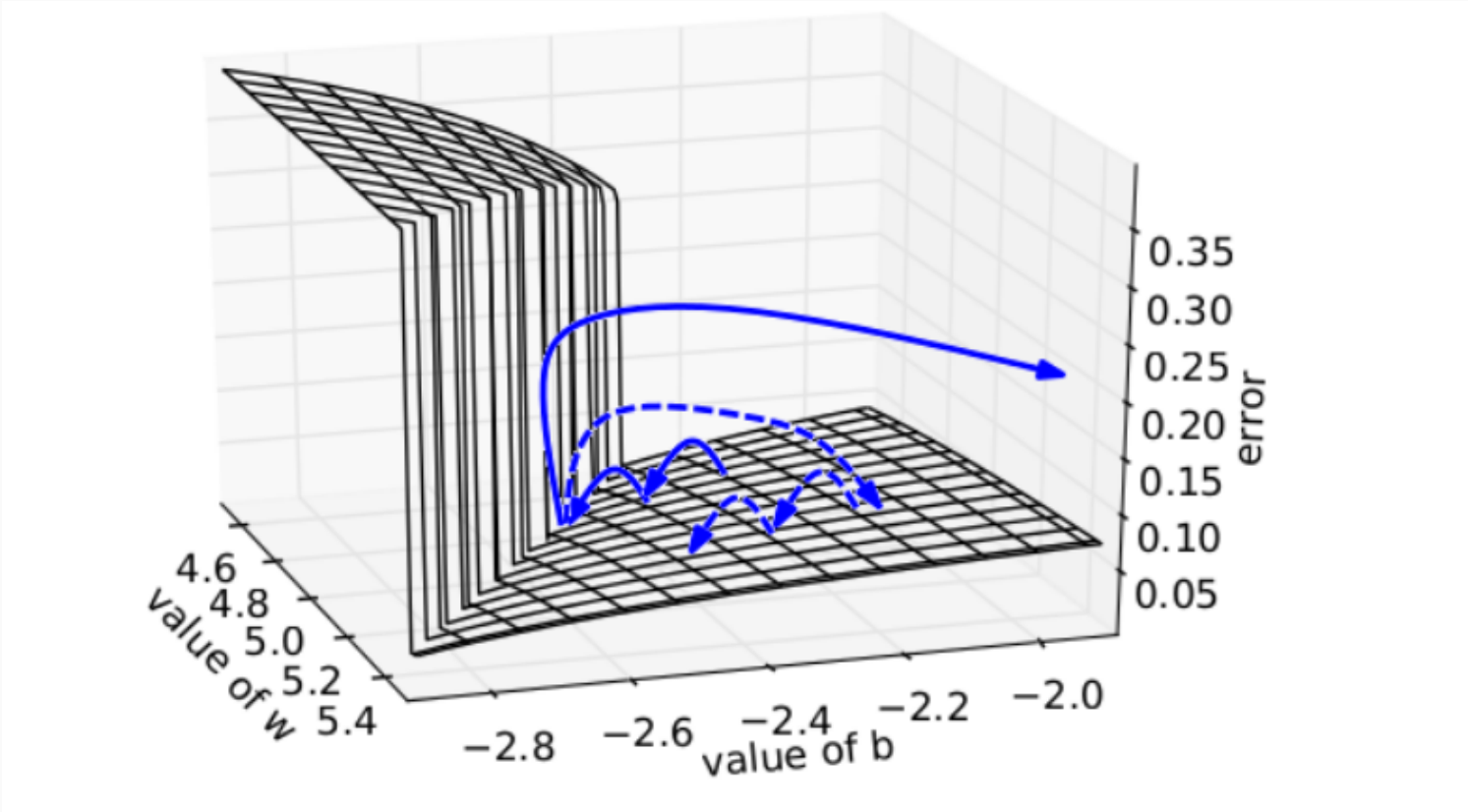
- Две проблемы:
  - взрывающиеся градиенты (exploding gradients);
  - затухающие градиенты (vanishing gradients).
- Надо каждый раз умножать на одну и ту же W, и норма градиента может расти или убывать экспоненциально.
- Взрывающиеся градиенты: надо каждый раз умножать на W, и норма градиента может расти экспоненциально.
- Что делать?

Взрывающиеся градиенты

- Да просто обрезать градиенты, ограничить сверху, чтобы не росли.
- Два варианта – ограничить общую норму или каждое значение:
  - `sgd = optimizers.SGD(lr=0.01, clipnorm=1.)`
  - `sgd = optimizers.SGD(lr=0.01, clipvalue=.05)`

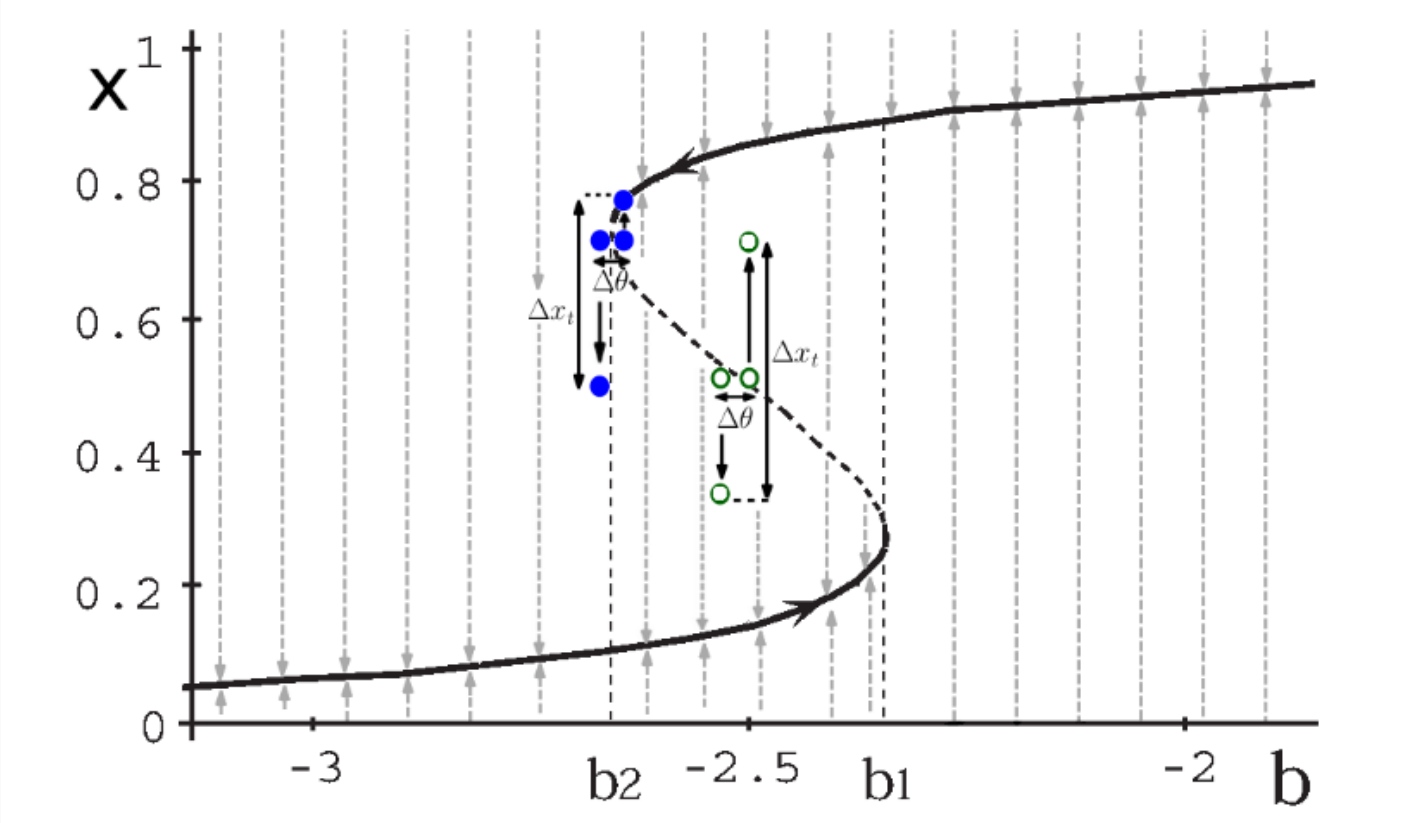
Взрывающиеся градиенты

- (Pascanu et al., 2013) – вот что будет происходить:



Взрывающиеся градиенты

- Там же объясняется, откуда возьмутся такие перепады: есть точки бифуркации у RNN.



Карусель константной ошибки: LSTM и GRU

LSTM

- Затухающие градиенты: надо каждый раз умножать на W.
- Поэтому не получается долгосрочную память реализовать.

- А хочется. Что делать?..

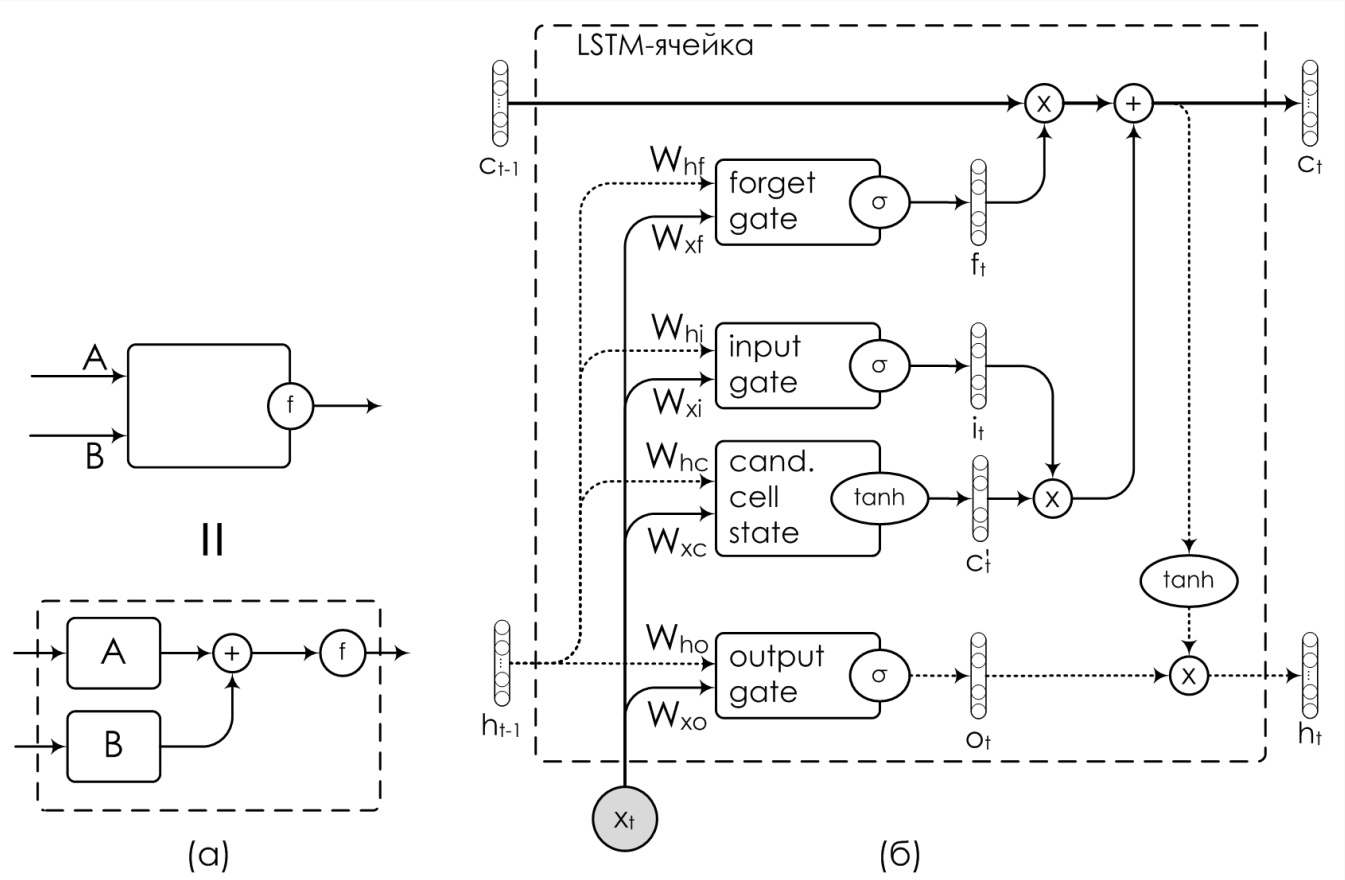
LSTM

- Базовую идею мы уже видели в ResNet: надо сделать так, чтобы градиент проходил.
- В RNN это называется «карусель константной ошибки» (constant error carousel).

- Идея из середины 1990-х (Шмидхубер): давайте составлять RNN из более сложных частей, в которых будет прямой путь для градиентов, и память будет контролироваться явно.

LSTM

- LSTM (long short-term memory). “Ванильный” LSTM: $\mathbf{c}_t$ – состояние ячейки памяти, $\mathbf{h}_t$ – скрытое состояние.
- Input gate и forget gate определяют, надо ли менять $\mathbf{c}_t$ на нового кандидата в состояния ячейки.



LSTM

- Формально:

$$
\begin{aligned}
\mathbf{c}'_t &= \tanh\left(W_{xc}\mathbf{x}_t + W_{hc}\mathbf{h}_{t-1} + \mathbf{b}_{c'}\right) && \textit{candidate cell state} \\
\mathbf{i}_t &= \sigma\left(W_{xi}\mathbf{x}_t + W_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i\right) && \textit{input gate} \\
\mathbf{f}_t &= \sigma\left(W_{xf}\mathbf{x}_t + W_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f\right) && \textit{forget gate} \\
\mathbf{o}_t &= \sigma\left(W_{xo}\mathbf{x}_t + W_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o\right) && \textit{output gate} \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{c}'_t, && \textit{cell state} \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t) && \textit{block output}
\end{aligned}
$$

- Так что LSTM может контролировать состояние ячейки при помощи скрытого состояния и весов.
- Например, если forget gate закрыт ($\mathbf{f}_t = 1$), то получится карусель константной ошибки: $\mathbf{c}_t = \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{c}'_t$, и $\frac{\partial \mathbf{c}_t}{\partial \mathbf{c}_{t-1}} = 1$.
- Важно инициализировать $\mathbf{b}_f$ большим, чтобы forget gate был закрыт поначалу.

LSTM

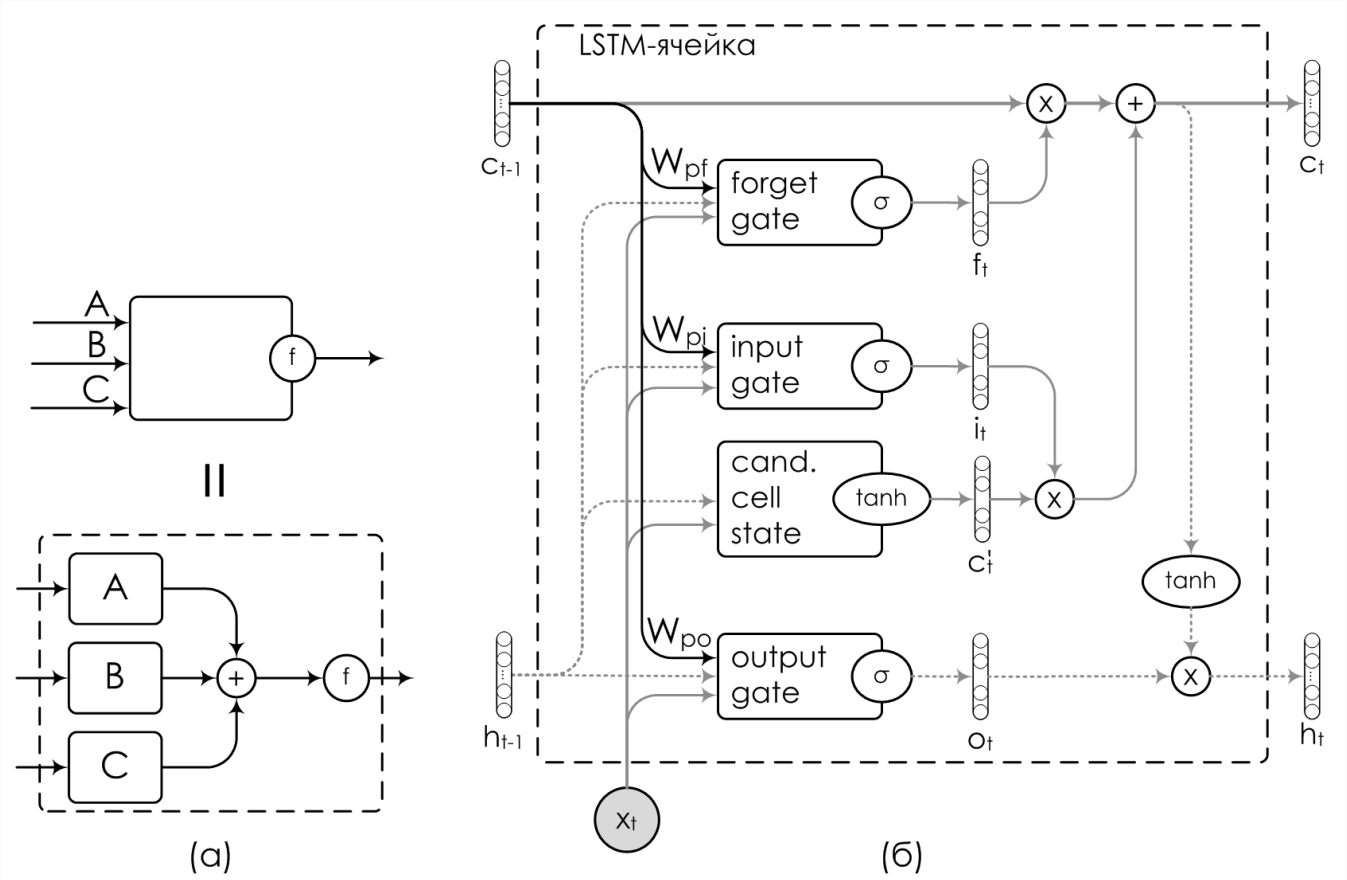
- LSTM был создан в середине 1990-х (Hochreiter and Schmidhuber, 1995; 1997).
- В полностью современной форме в (Gers, Schmidhuber, 2000).
- Проблема: хотим управлять $\mathbf{c}$, но гейты его не получают! Они видят только $\mathbf{h}_{t-1}$, а это

$$\mathbf{h}_{t-1} = \mathbf{o}_{t-1} \odot \tanh(\mathbf{c}_{t-1}).$$

- Так что если output gate закрыт, то поведение LSTM вообще от состояния ячейки не зависит.
- Нехорошо. Что делать?..

LSTM

- ...конечно, добавить ещё несколько матриц! (peepholes)



LSTM

- Формально:

$$\begin{aligned}
\mathbf{i}_t &= \sigma\left(W_{xi}\mathbf{x}_t + W_{hi}\mathbf{h}_{t-1} + W_{pi}\mathbf{c}_{t-1} + \mathbf{b}_i\right) \\
\mathbf{f}_t &= \sigma\left(W_{xf}\mathbf{x}_t + W_{hf}\mathbf{h}_{t-1} + W_{pf}\mathbf{c}_{t-1} + \mathbf{b}_f\right) \\
\mathbf{o}_t &= \sigma\left(W_{xo}\mathbf{x}_t + W_{ho}\mathbf{h}_{t-1} + W_{po}\mathbf{c}_{t-1} + \mathbf{b}_o\right)
\end{aligned}$$

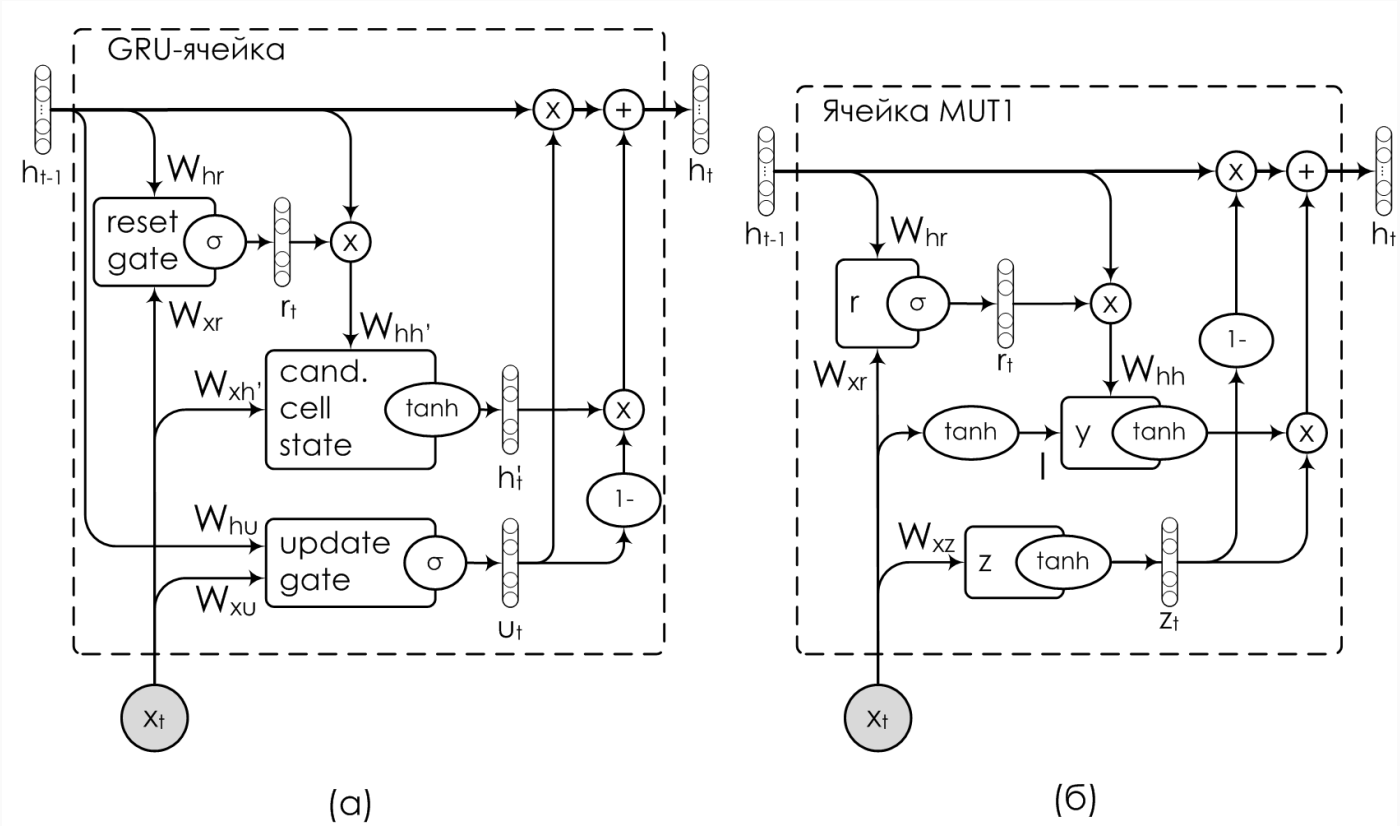
- Видно, что тут есть огромное поле для вариантов LSTM: можно удалить любой гейт, любую замочную скважину, поменять функции активации...
- Как выбрать?

LSTM

- «LSTM: a Search Space Odyssey» (Greff et al., 2015).
- Большое экспериментальное сравнение.
- В честности, некоторые куда более простые архитектуры (без одного из гейтов!) не сильно проигрывали «ванильному» LSTM.
- И это приводит нас к...

GRU

- ...Gated Recurrent Units (GRU; Cho et al., 2014).
- В GRU тоже есть прямой путь для градиентов, но проще.



GRU

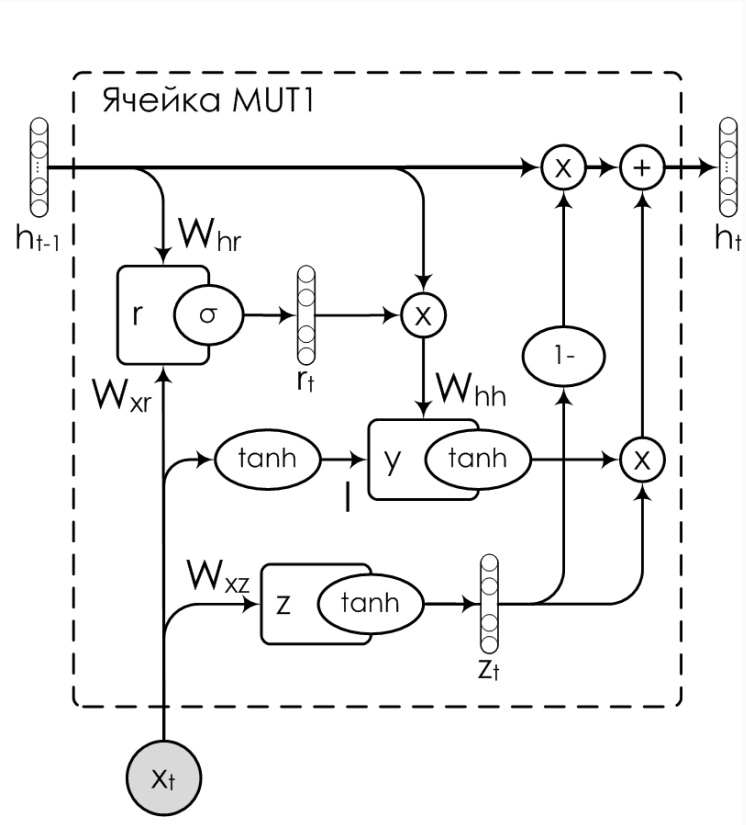
- Формально:

$$
\begin{aligned}
\mathbf{u}_t &= \sigma(W_{xu}\mathbf{x}_t + W_{hu}\mathbf{h}_{t-1} + \mathbf{b}_u) \\
\mathbf{r}_t &= \sigma(W_{xr}\mathbf{x}_t + W_{hr}\mathbf{h}_{t-1} + \mathbf{b}_r) \\
\mathbf{h}'_t &= \tanh(W_{xh'}\mathbf{x}_t + W_{hh'}(\mathbf{r}_t \odot \mathbf{h}_{t-1})) \\
\mathbf{h}_t &= (1 - \mathbf{u}_t) \odot \mathbf{h}'_t + \mathbf{u}_t \odot \mathbf{h}_{t-1}
\end{aligned}
$$

- Теперь есть update gate и reset gate, нет разницы между $\mathbf{c}_t$ и $\mathbf{h}_t$.
- Меньше матриц (6, а не 8 или 11 с замочными скважинами), меньше весов, но только чуть хуже LSTM работает.
- Так что можно больше GRU поместить, и сеть станет лучше.

GRU

- Другие варианты тоже есть.
- (Józefowicz, Zaremba, Sutskever, 2015): огромное сравнение, выращивали архитектуры эволюционными методами.
- Три новых интересных архитектуры; например:



Долгосрочная память в базовых RNN

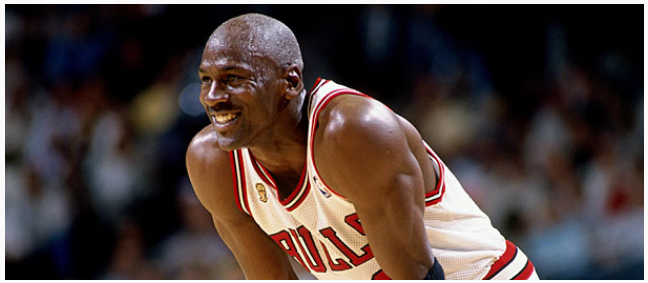
SCRN

- Следующая идея о том, как добавить долгосрочную память.
- Начнём опять с простой RNN:

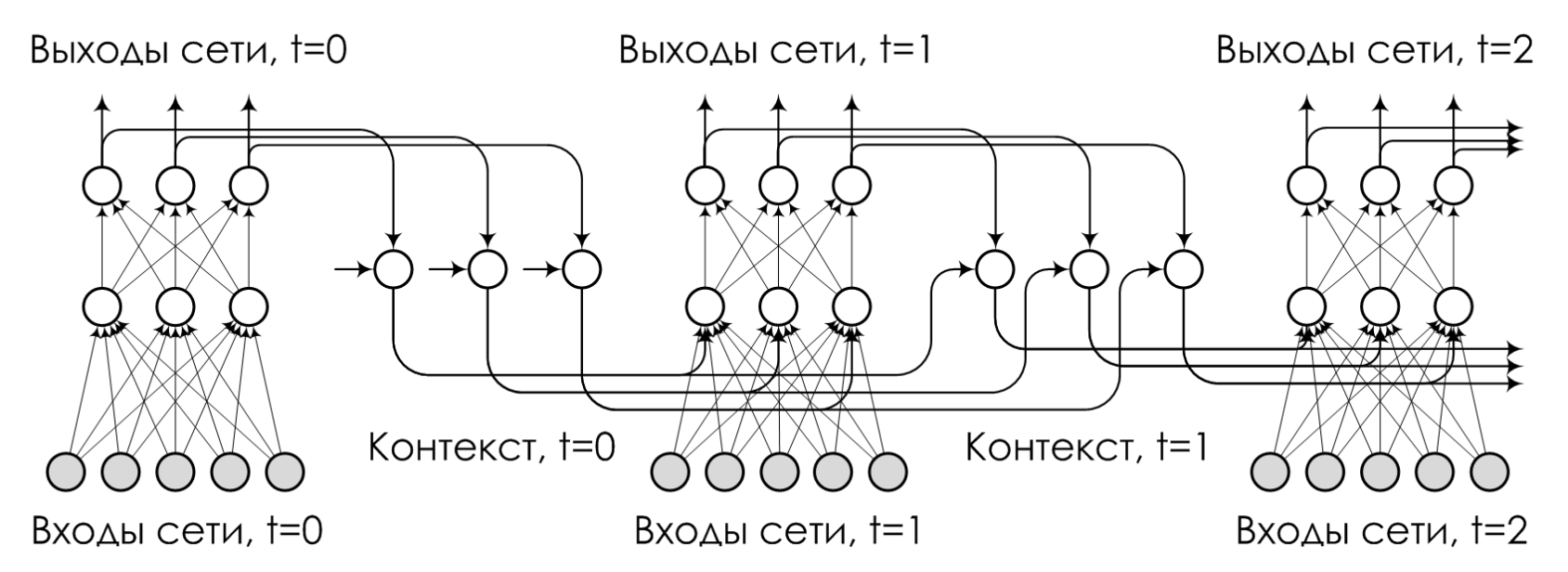
$$\mathbf{s}_t = f(U\mathbf{x}_t + W\mathbf{s}_{t-1} + \mathbf{b}), \quad \mathbf{y}_t = h(U\mathbf{s}_t + \mathbf{c}).$$

- Проблема с градиентами в том, что мы умножаем на W, и градиенты либо взрываются, либо затухают.
- Давайте вернёмся к истории RNN...

SCRN



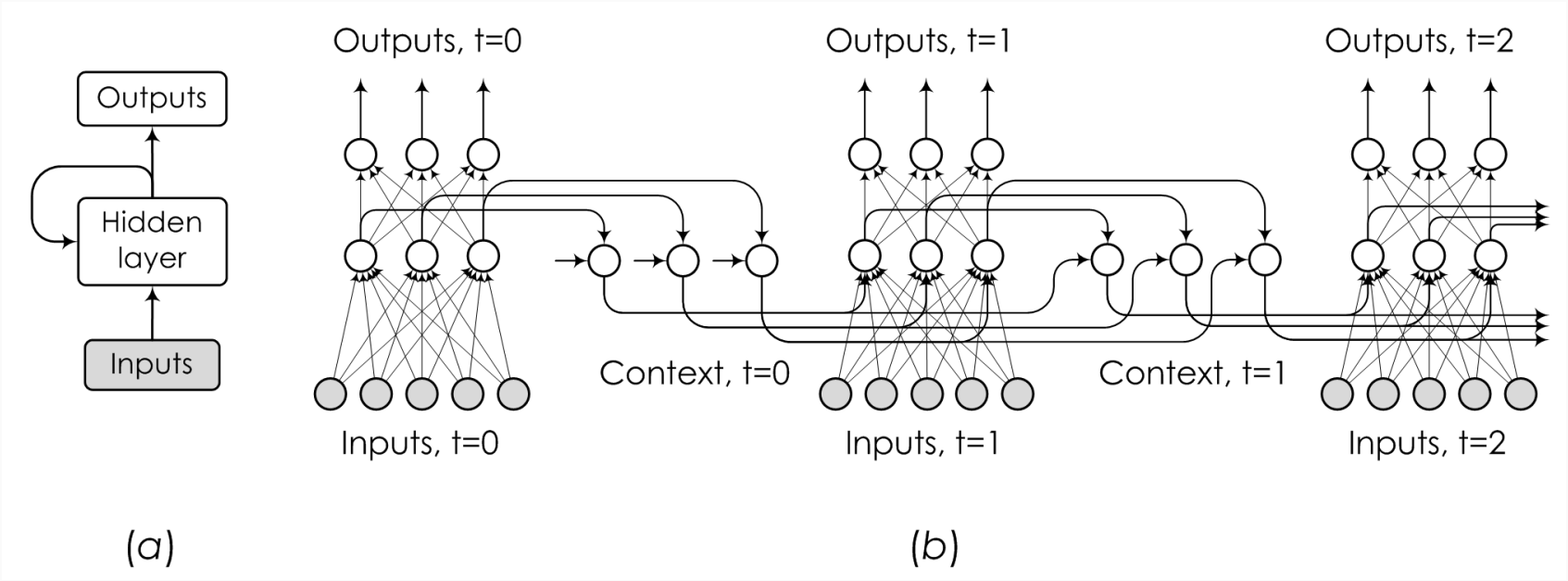
- Сеть Джордана (середина 1980-х):



- Считается первой успешной RNN.

SCRN

- Сеть Элмана (Elman; конец 1980-х):



- Разница в том, что нейроны контекста $\mathbf{c}_t$ получают входы со скрытого уровня, а не выходов.
- И нет никаких весов от предыдущих $\mathbf{c}_{t-1}$! То есть веса фиксированы и равны 1.

SCRN

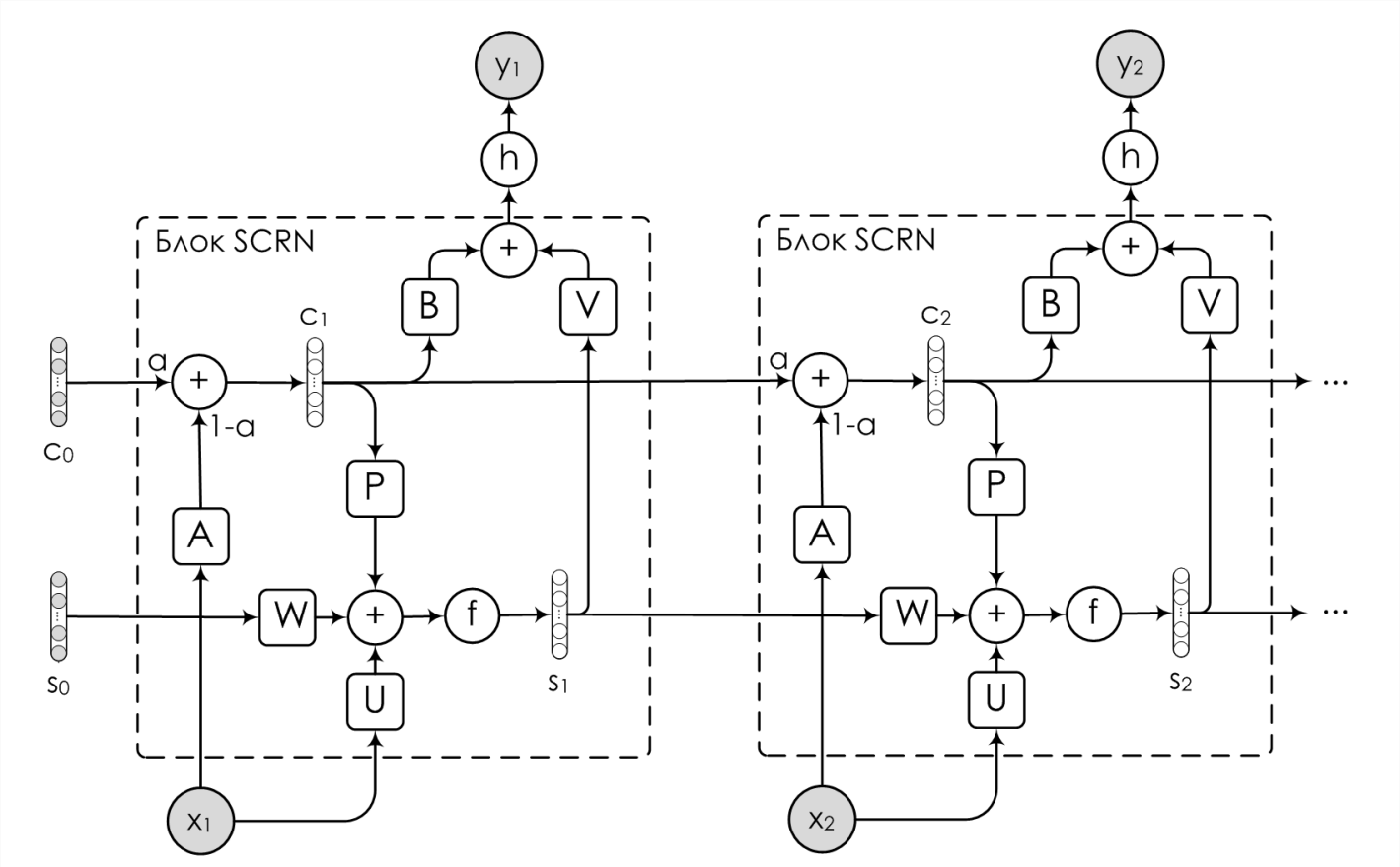
- Это приводит к хорошим долгосрочным эффектам, потому что нет нелинейности между последовательными шагами, и карусель константной ошибки получается по определению:

$$\mathbf{c}_t = \mathbf{c}_{t-1} + U\mathbf{x}_t.$$

- Идея: можно зафиксировать градиенты, использовав единичную матрицу весов вместо обучаемой W.
- Долгосрочная память тут есть... но обучать очень трудно, потому что градиенты надо возвращать к началу последовательности.

SCRN

- (Mikolov et al., 2014): Structurally Constrained Recurrent Network (SCRN).
- Сочетание двух идей – $\mathbf{s}_t$ с W и $\mathbf{c}_t$ с диагональной матрицей рекуррентных весов.



SCRN

- Формально:

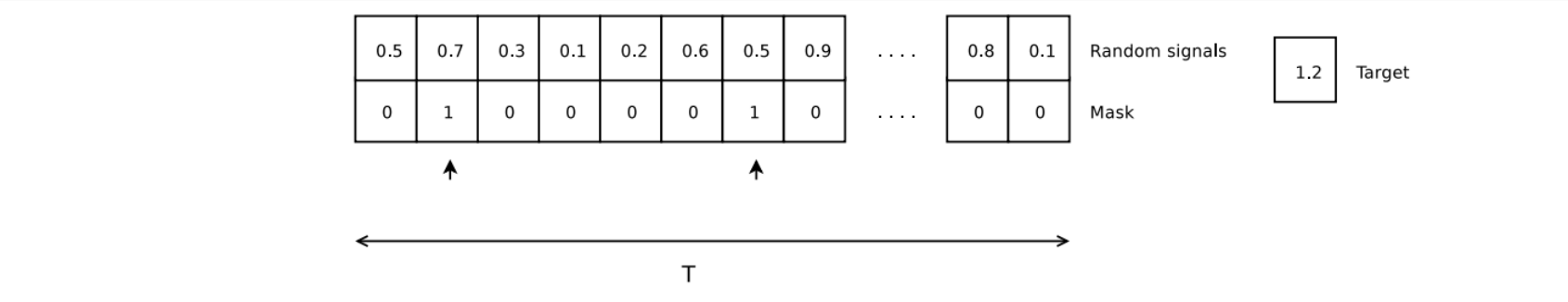
$$\begin{aligned}\mathbf{c}_t &= (1-\alpha)\, A\mathbf{x}_t + \alpha \mathbf{c}_{t-1},\\ \mathbf{s}_t &= f(P\mathbf{c}_t + U\mathbf{x}_t + W\mathbf{s}_{t-1}),\\ \mathbf{y}_t &= h(V\mathbf{s}_t + B\mathbf{s}_t).\end{aligned}$$

- SCRN – это просто обычный RNN, где $\mathbf{s}_t$ и $\mathbf{c}_t$ в одном векторе, и матрица рекуррентных весов имеет вид

$$W = \begin{pmatrix} R & P \\ 0 & \alpha \mathbf{I} \end{pmatrix},$$

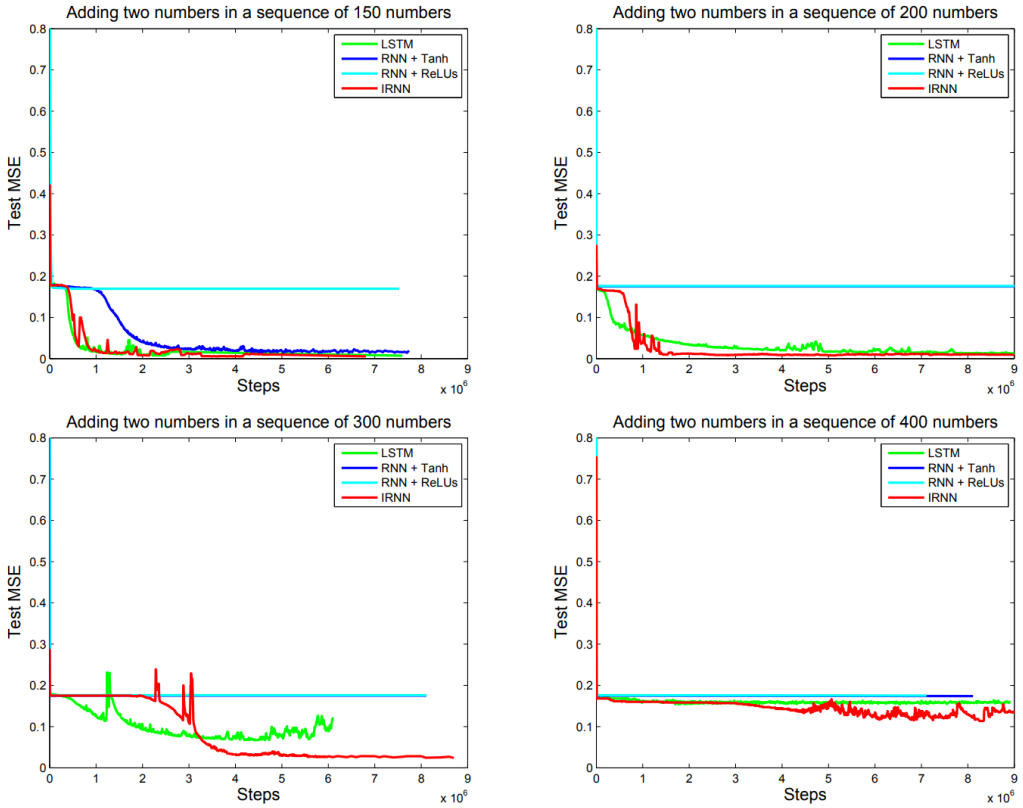
Инициализация RNN с ReLU

- (Le et al., 2015): как правильно инициализировать рекуррентные веса
- IRNN – составим рекуррентные веса с ReLU-активациями и инициализируем единичной матрицей; похоже на SCRN, но ещё проще
- Пример игрушечной задачи для long-range dependencies:



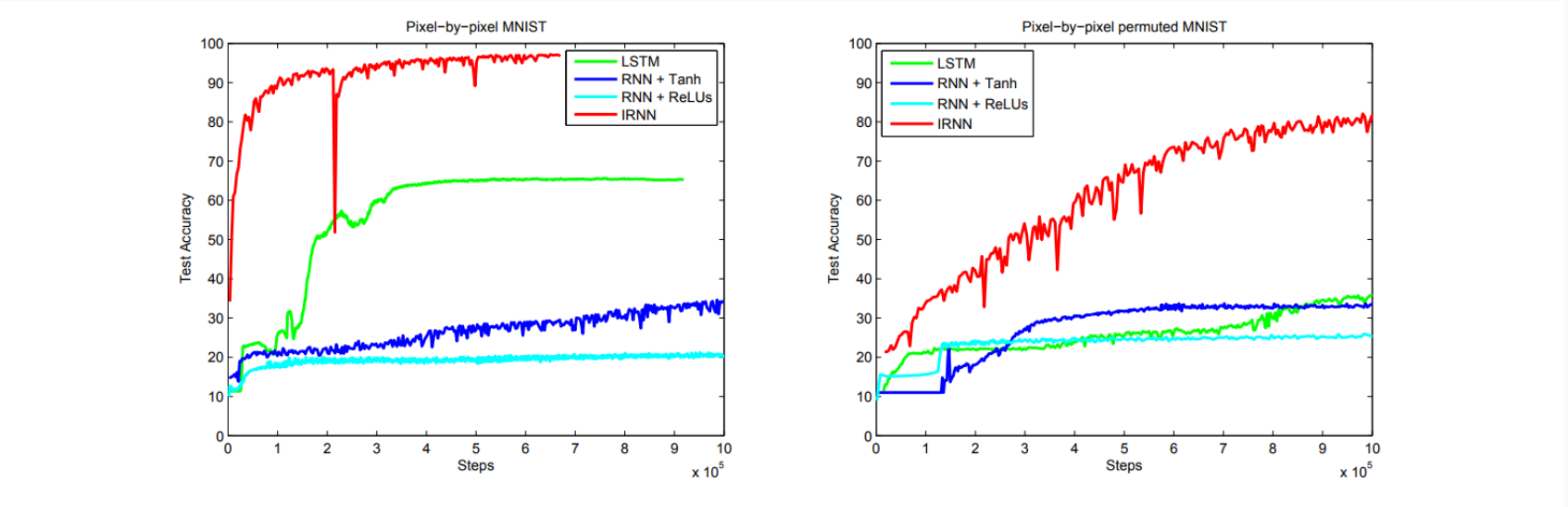
Инициализация RNN с ReLU

- И получается хорошо:



Инициализация RNN с ReLU

- А ещё pixel-by-pixel MNIST:



Регуляризуем W

- Альтернатива: давайте просто регуляризуем W так, чтобы $\det W = 1$.
- Мягкая регуляризация (Pascanu et al., 2013):

$$\Omega = \sum_k \Omega_k = \sum_k \left(\left\| \frac{\frac{\partial E}{\partial \mathbf{s}_{k+1}} \frac{\partial \mathbf{s}_{k+1}}{\partial \mathbf{s}_k}}{\frac{\partial E}{\partial \mathbf{s}_{k+1}}} \right\| - 1 \right)^2.$$

- Жёсткая регуляризация – сделаем W автоматически унитарной (Arjovsky et al., 2015):

$$W = D_3 R_2 F^{-1} D_2 \Pi R_1 F D_1,$$

где D – диагональные матрицы, F – преобразование Фурье, R – отражения, Π – перестановка.

- Кстати, и параметров меньше: теперь только $O(n)$ вместо $O(n^2)$.

Инициализируем W

- И ещё более простой трюк: давайте правильно инициализируем W (Le, Jaitly, Hinton, 2015).
- Рассмотрим RNN с ReLU-активациями на рекуррентных весах (перед $\mathbf{h}$).
- Тогда если W_{hh} – единичная матрица и $\mathbf{b}_h = 0$, скрытое состояние не изменится, градиент протечёт насквозь.
- Давайте так и инициализируем! Часто приводит к серьёзным улучшениям.

Что делать с RNN на практике

Принципы

- RNN имеют довольно простую общую структуру: уровни LSTM или GRU.
- Все они выдают последовательность выходов, кроме, возможно, верхнего.
- Дропаут и batchnorm между слоями, а на рекуррентных связях надо аккуратно (потом поговорим).
- Слоёв немного; больше 3-4 трудно, 7-8 сейчас максимум.

Принципы

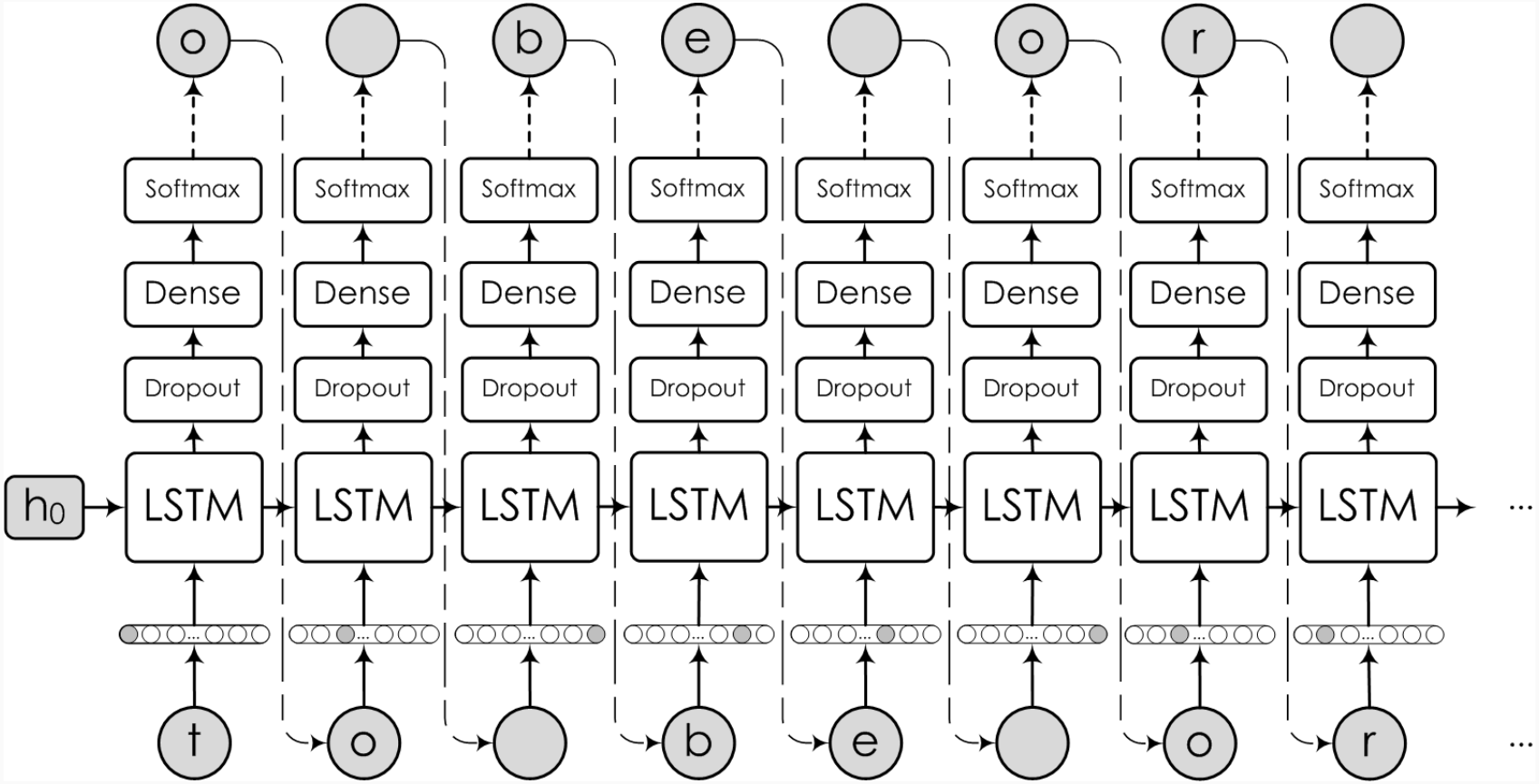
- Важный трюк: *skip-layer connections*, как residual, только проще. Добавляем выходы предыдущих слоёв «через один» или «через два», просто конкатенацией.
- RNN, сохраняющие состояние: состояния с одного мини-батча переиспользуются как начальные для следующего. Градиенты в BPTT останавливаются, но состояния остаются.

Пример: порождение текста с RNN

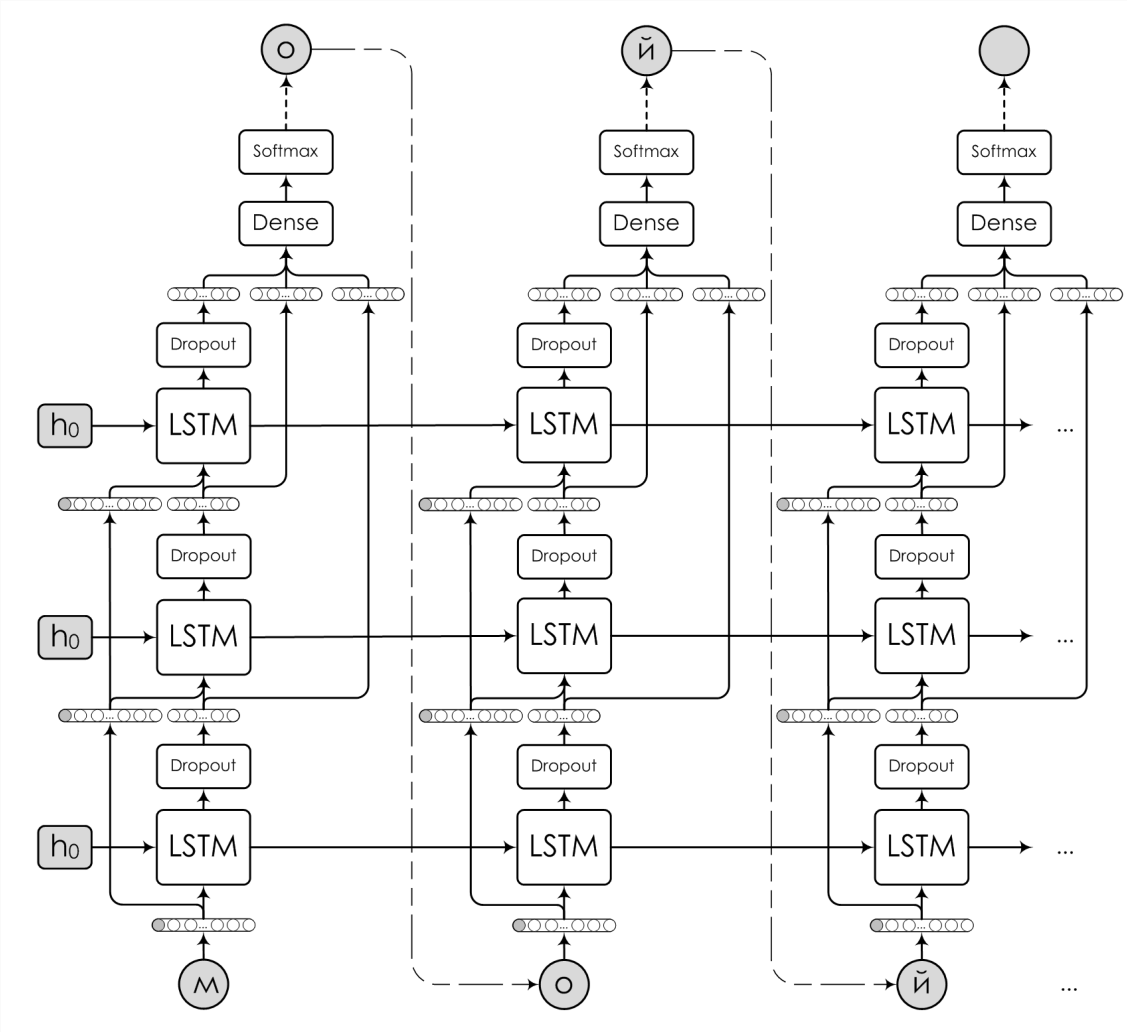
- Языковые модели – это естественное прямое приложение к NLP.
- Первая идея – давайте просто обучим последовательность слов через RNN/LSTM.
- О языке будем говорить позже, а пока любопытно, что можно обучить RNN порождать интересные последовательности даже просто символ за символом.
- Karpathy, «The Unreasonable Effectiveness of Neural Networks»; знаменитый пример из (Sutskever et al. 2011):

  The meaning of life is the tradition of the ancient human reproduction: it is less favorable to the good boy for when to remove her bigger...
- Это, конечно, всего лишь эффекты краткосрочной памяти, никакого «понимания».

Simple LSTM-based architecture



Slightly less simple LSTM-based architecture



Спасибо за внимание!