

Механизмы внимания в нейронных сетях

Сергей Николенко

НИУ ВШЭ – Санкт-Петербург

07 ноября 2020 г.

Random facts:

- 7 ноября 1631 г. Пьер Гассенди впервые наблюдал прохождение Меркурия по диску Солнца, предсказанное Кеплером
- 7 ноября 1902 г. в Туле, по инициативе местного врача Фёдора Архангельского, открылся первый в России вытрезвитель под названием «Приют для опьяневших»
- 7 ноября 1917 г. в России произошла Октябрьская революция
- 7 ноября 1967 г. Карл Стоукс стал первым темнокожим мэром крупного американского города (Кливленда), а 7 ноября 1990 г. Мэри Робинсон стала первой женщиной-президентом Ирландии

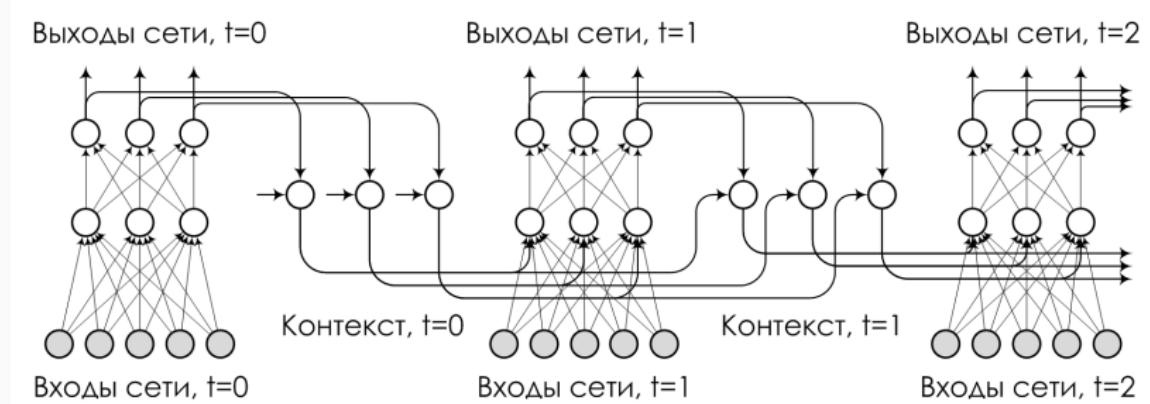
Долгосрочная память в базовых RNN

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

$$\mathbf{s}_t = f(\mathbf{Ux}_t + \mathbf{Ws}_{t-1} + \mathbf{b}), \quad \mathbf{y}_t = h(\mathbf{Us}_t + \mathbf{c}).$$

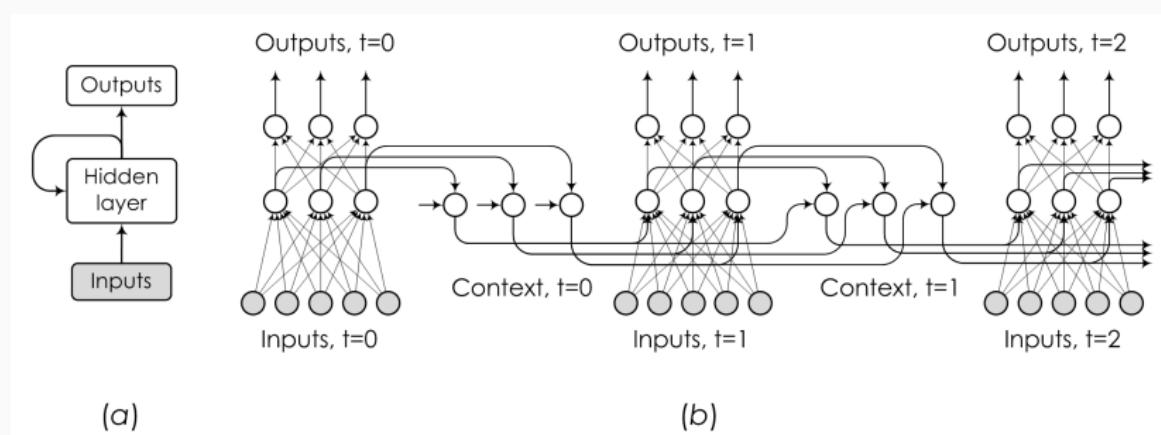
- Проблема с градиентами в том, что мы умножаем на \mathbf{W} , и градиенты либо взрываются, либо затухают.
- Давайте вернёмся к истории RNN...

- Сеть Джордана (середина 1980-х):



- Считается первой успешной RNN.

- Сеть Элмана (Elman; конец 1980-х):



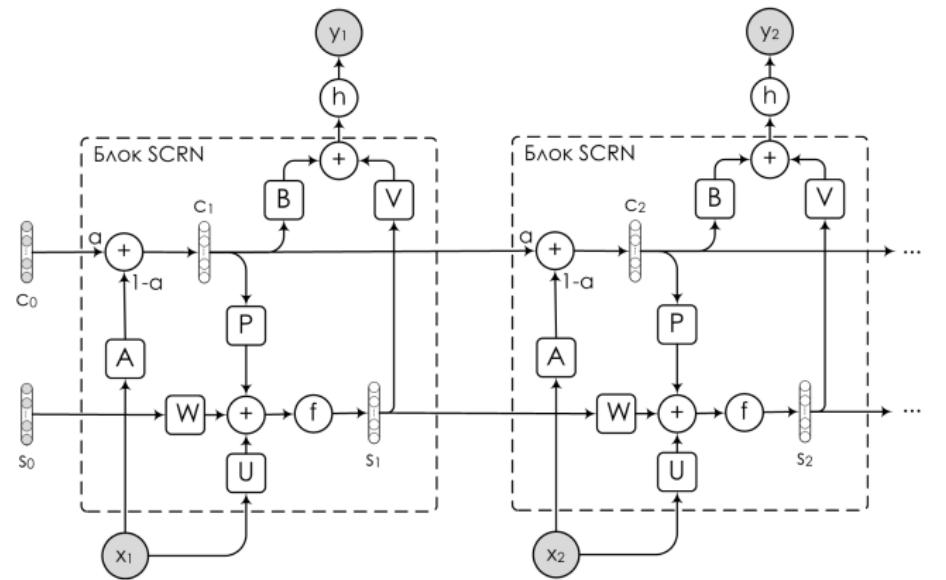
- Разница в том, что нейроны контекста c_t получают входы со скрытого уровня, а не выходов.
- И нет никаких весов от предыдущих c_{t-1} ! То есть веса фиксированы и равны 1.

- 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.

- (Mikolov et al., 2014): Structurally Constrained Recurrent Network (SCRN).
- Сочетание двух идей – s_t с W и c_t с диагональной матрицей рекуррентных весов.



- Формально:

$$\mathbf{c}_t = (1 - \alpha) A \mathbf{x}_t + \alpha \mathbf{c}_{t-1},$$

$$\mathbf{s}_t = f(P \mathbf{c}_t + U \mathbf{x}_t + W \mathbf{s}_{t-1}),$$

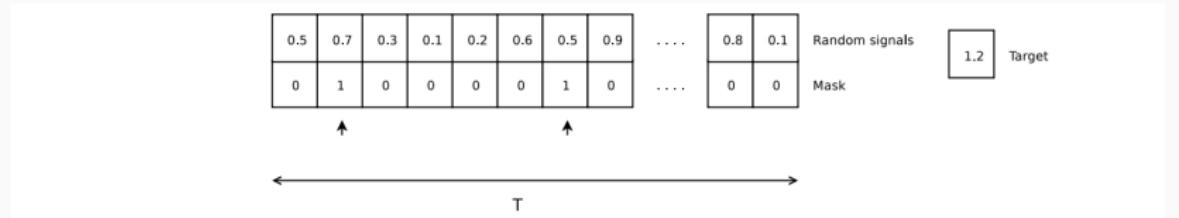
$$\mathbf{y}_t = h(V \mathbf{s}_t + B \mathbf{s}_t).$$

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

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

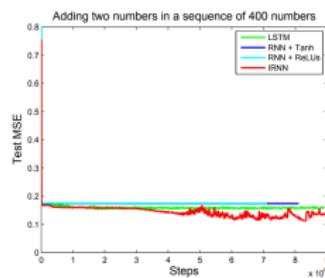
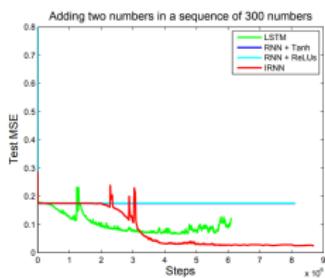
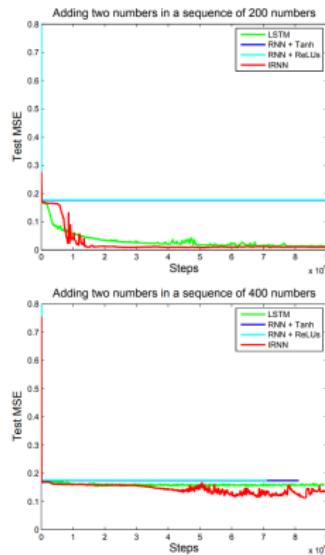
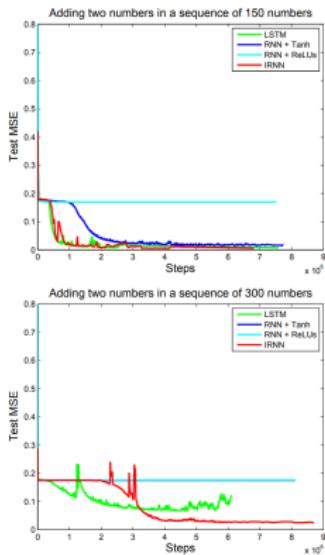
Инициализация RNN с ReLU

- (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:



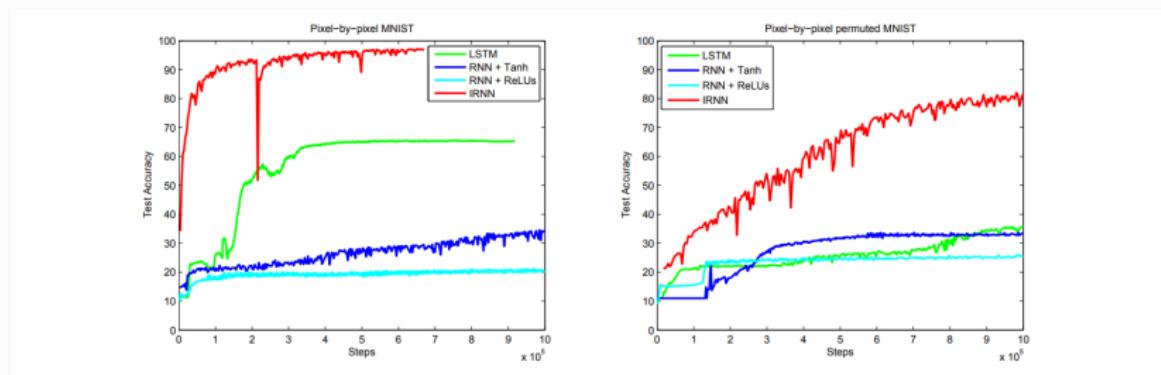
Инициализация RNN с ReLU

- 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):

$$\Omega = \sum_k \Omega_k = \sum_k \left(\left\| \frac{\frac{\partial E}{\partial s_{k+1}} \frac{\partial s_{k+1}}{\partial s_k}}{\frac{\partial E}{\partial s_{k+1}}} \right\| - 1 \right)^2.$$

- 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 $O(n^2)$.

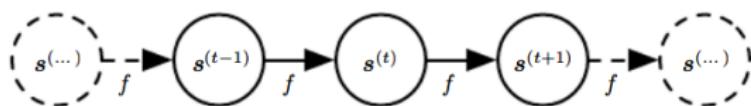
Инициализируем W

- И ещё более простой трюк: давайте правильно инициализируем W (Le, Jaitly, Hinton, 2015).
- Рассмотрим RNN с ReLU-активациями на рекуррентных весах (перед 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.

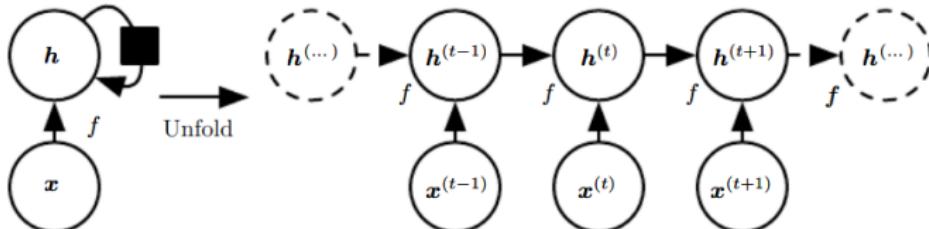
Еще раз об RNN

Обзор вариантов RNN

- Давайте пройдёмся по вариантам того, как могут быть рекуррентно связаны нейрончики.
- Динамическая система:

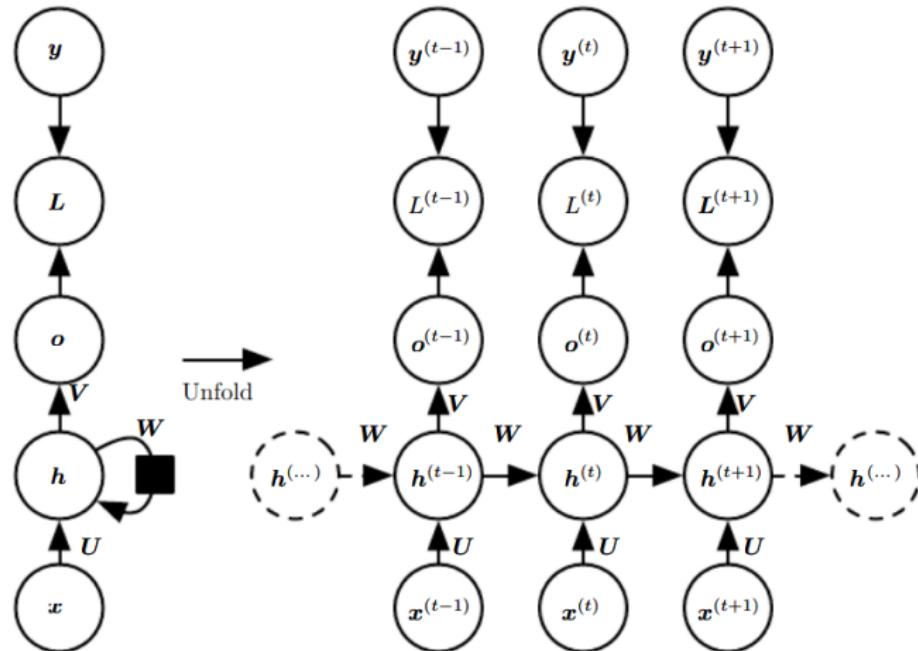


- RNN без выходов:



Обзор вариантов RNN

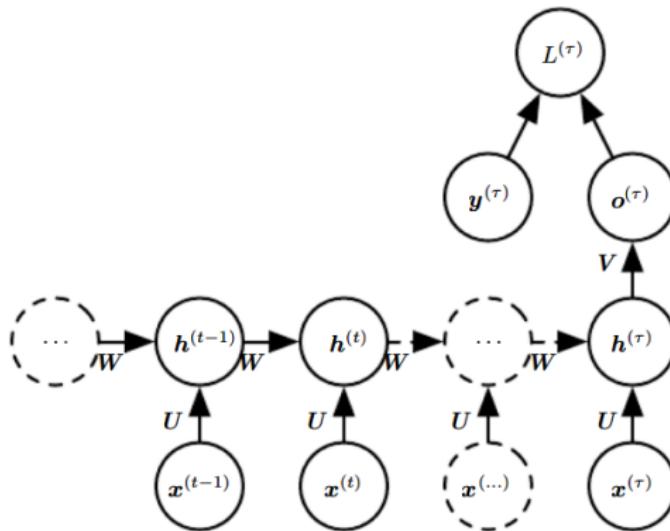
- RNN с выходами на каждом шаге:



- L – функция ошибки, y – правильный ответ, o – выход сети.

Обзор вариантов RNN

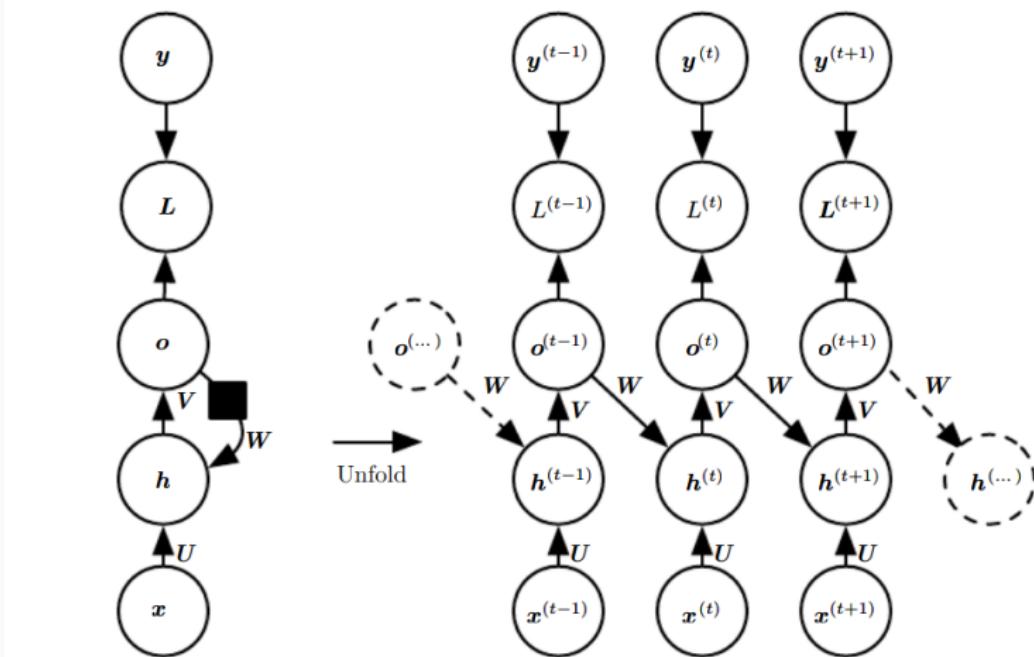
- Сеть с одним выходом в самом конце:



- Если есть hidden-to-hidden, то только разные варианты ВРТ для обучения.

Обзор вариантов RNN

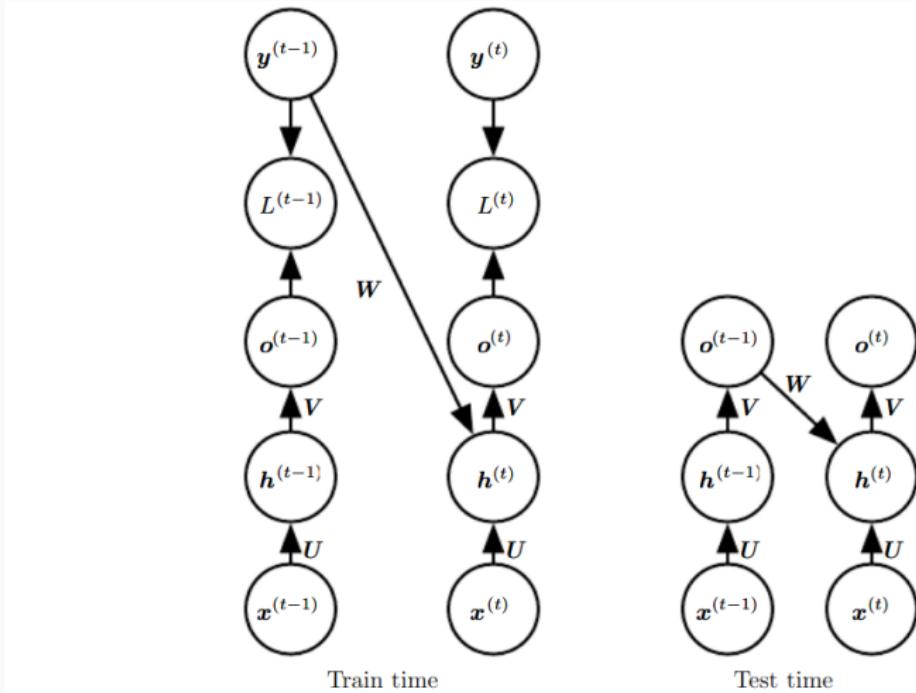
- RNN с output-to-hidden связями (как в сети Джордана):



- Менее выразительная, меньше функций может сделать.

Обзор вариантов RNN

- Зато teacher forcing:



Обзор вариантов RNN

- На самом деле это всего лишь максимизация правдоподобия.
- RNN моделирует последовательность как

$$p(y^{(1)}, y^{(2)}, \dots, y^{(T)}) = p(y^{(1)})p(y^{(2)} | y^{(1)})p(y^{(3)} | y^{(1)}, y^{(2)}) \dots p(y^{(T)} | y^{(T-1)}, \dots, y^{(1)}).$$

- И если это раскрыть:

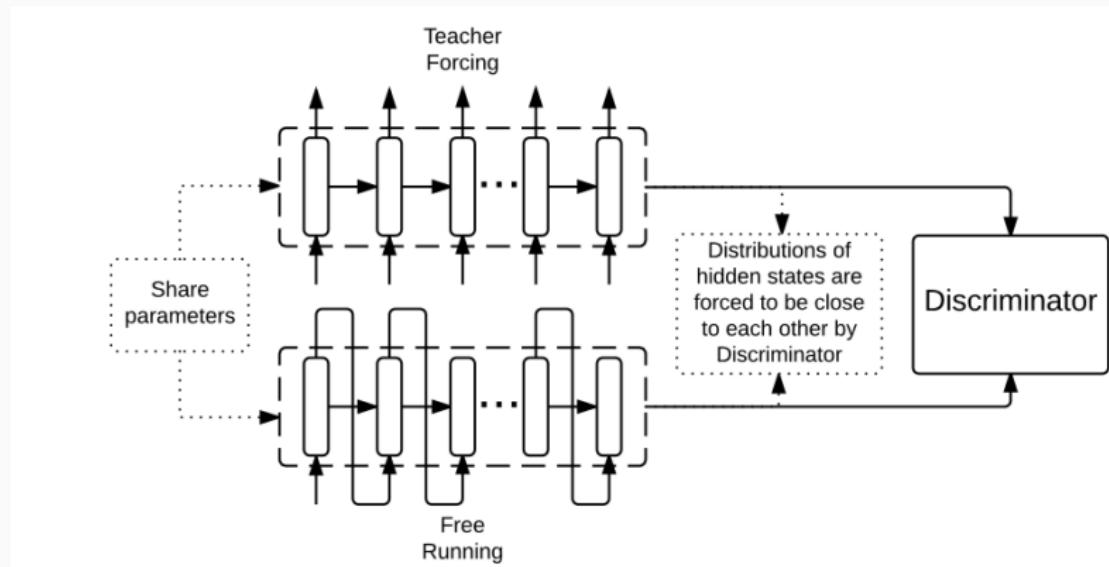
$$p(y^{(1)}, y^{(2)} | x^{(1)}, x^{(2)}) = p(y^{(2)} | y^{(1)}, x^{(1)}, x^{(2)})p(y^{(1)} | x^{(1)}, x^{(2)}),$$

мы видим, что $y^{(1)}$ надо подставить, чтобы перейти к $y^{(2)}$ – это и есть teacher forcing.

- Если есть и output-to-hidden, и hidden-to-hidden, то можно совместить teacher forcing с BPTT.
- Проблема: стоит учителю отвернуться, и сеть будет плохо себя вести. Ошибки накапливаются.

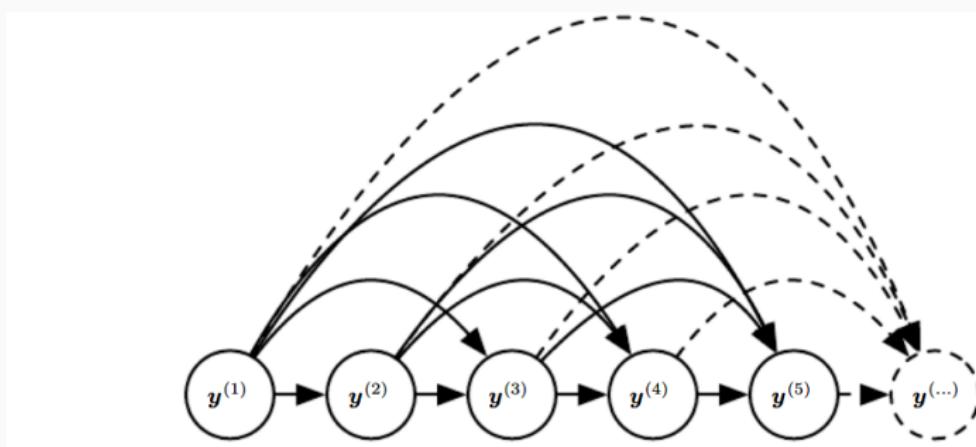
Обзор вариантов RNN

- (Lamb, Goyal et al., 2016): Professor Forcing на основе GAN-подобных идей (подробности позже).



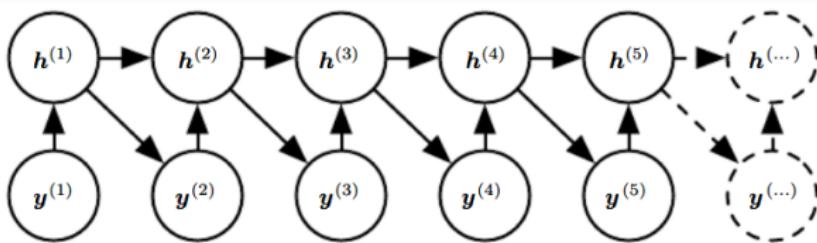
Обзор вариантов RNN

- Вообще можно рассмотреть RNN как графическую модель.
- Мы на шаге t пытаемся максимизировать $\log p(y^{(t)} | x^{(1)}, \dots, x^{(t)})$ или, если есть связи из выходов, $\log p(y^{(t)} | x^{(1)}, \dots, x^{(t)}, y^{(1)}, \dots, y^{(t)})$.
- Например, если нет x , то получается как бы



Обзор вариантов RNN

- Но на самом деле RNN гораздо эффективнее.
- На той картинке маргинализованы h , а в реальности



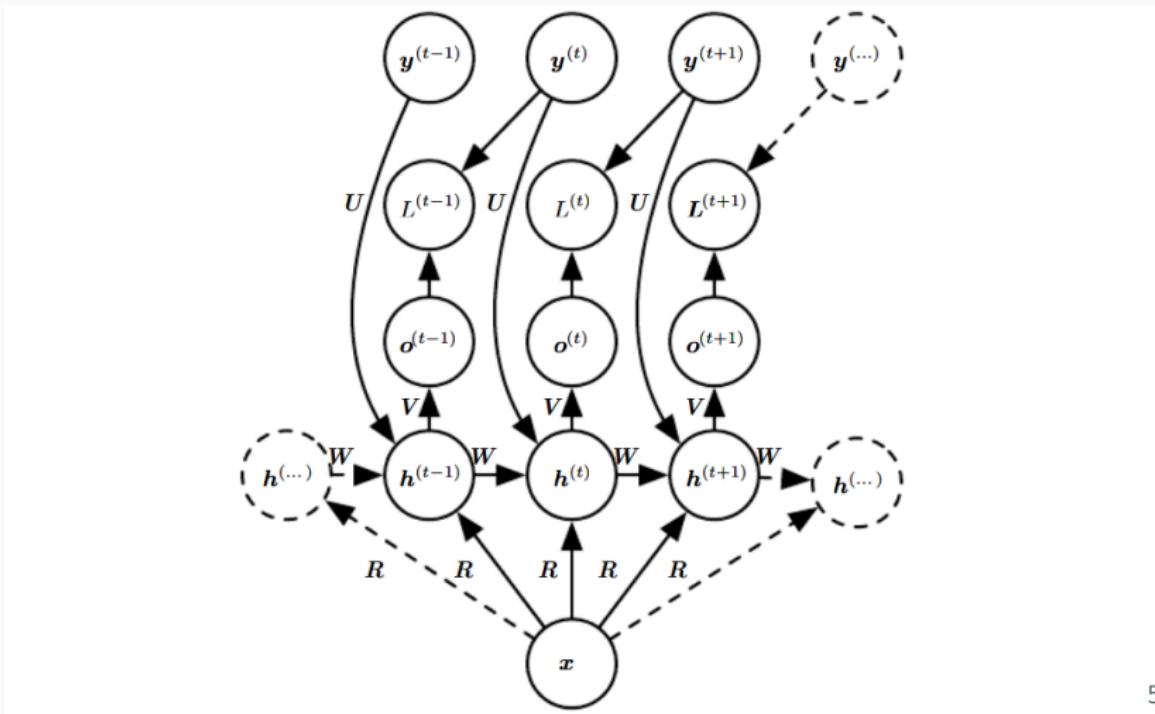
- И остался вопрос: как сэмплировать из этой модели?
- Чего мы ещё не обсудили на эту тему?

Обзор вариантов RNN

- Надо понять, когда останавливаться. Варианты:
 - добавить спец. символ «стоп»; не работает, когда не символы порождаем;
 - добавить лишнюю монетку к выходу;
 - предсказать продолжительность выхода τ и добавить ко входу сколько осталось.

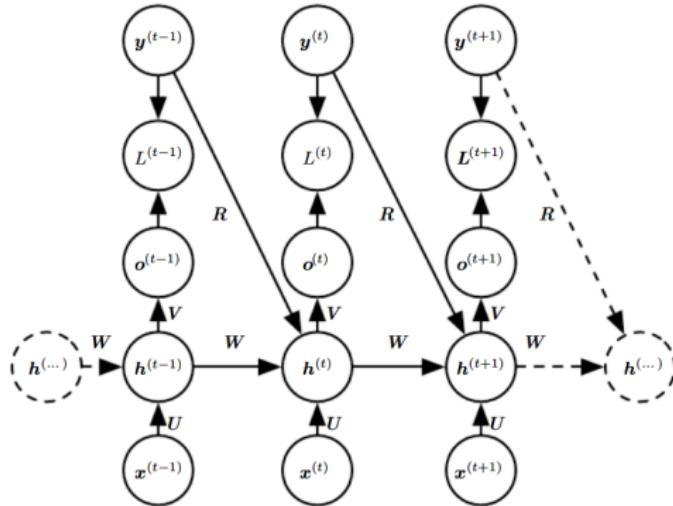
Обзор вариантов RNN

- Теперь можно добавить контекст, т.е. вход x .
- Он может быть один для всей RNN:



Обзор вариантов RNN

- А может быть тоже последовательностью:

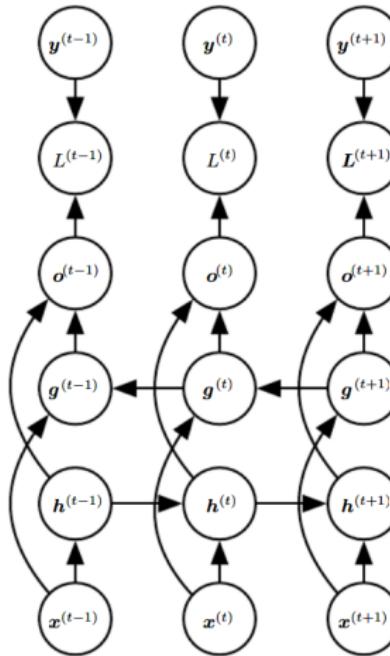


- Вероятностное предположение в том, что распределение раскладывается в

$$\prod_t p(y^{(t)} \mid x^{(1)}, \dots, x^{(t)}).$$

Обзор вариантов RNN

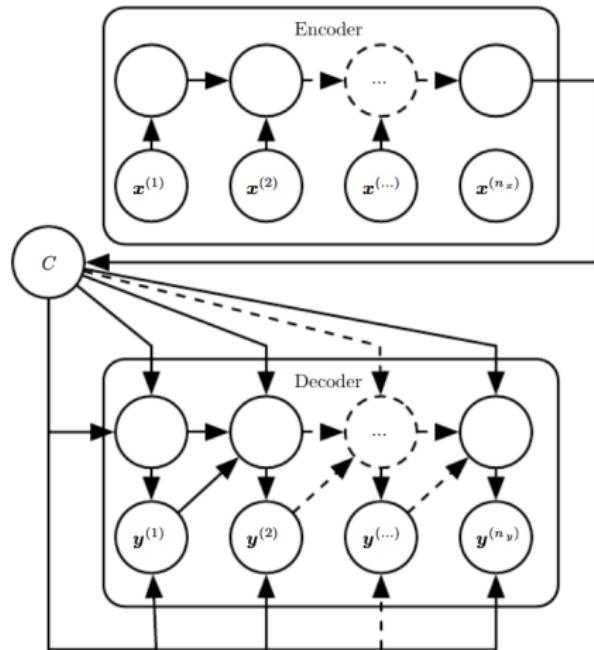
- Ослабить предположение можно двунаправленными сетями:



- Но это всё переводит последовательность в последовательность той же длины.
- А как сделать выход произвольной длины?

Обзор вариантов RNN

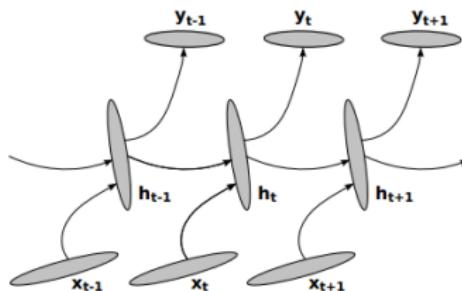
- Encoder-decoder (seq2seq):



- Изначально для машинного перевода (Cho et al., 2014; Sutskever et al., 2014); добавили внимание, но об этом потом.

Глубокие RNN

- Как сделать RNN глубокой? Какие варианты?
- (Pascanu et al., 2014): у нас есть input-to-hidden, hidden-to-output и hidden-to-hidden.



Глубокие RNN

- И любое место можно сделать глубоким; а можно сделать стек из слоёв:

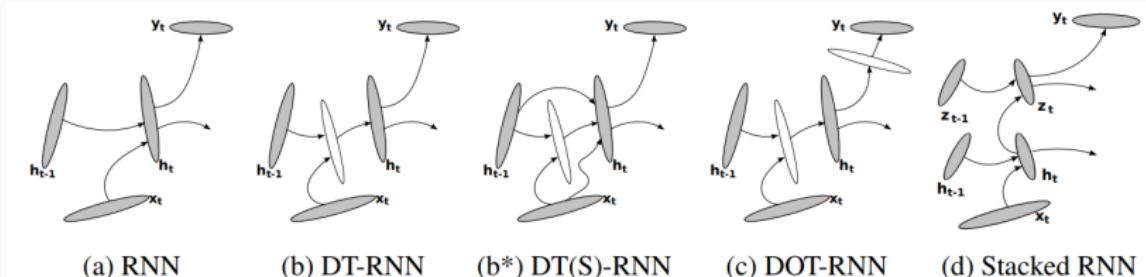


Figure 2: Illustrations of four different recurrent neural networks (RNN). (a) A conventional RNN. (b) Deep Transition (DT) RNN. (b*) DT-RNN with shortcut connections (c) Deep Transition, Deep Output (DOT) RNN. (d) Stacked RNN

Нормализация по уровню

- Вспомним batch normalization: очень хорошая штука, борется со сдвигом в переменных.
- Но применить batchnorm к RNN не получается:
 - теперь «уровень» – это шаг последовательности;
 - и получается, что веса общие, а статистики надо хранить по отдельности;
 - плохо, если последовательности разной длины;
 - совсем плохо, если при применении будет длиннее, чем в обучающей выборке.
- Что делать?

Нормализация по уровню

- Нормализация по уровню (layer normalization; Ba, Kiros, Hinton, 2016): будем просто усреднять по одному уровню.
- Раньше было $\mathbf{a}_t = W_{hh} \mathbf{h}_{t-1} + W_{xh} \mathbf{x}_t$.
- Теперь будет

$$\mathbf{h}_t = f \left[\frac{\mathbf{g}}{\sigma_t} \odot (\mathbf{a}_t - \mu_t) + \mathbf{b} \right],$$

$$\text{где } \mu_t = \frac{1}{H} \sum_{i=1}^H a_{it}, \quad \sigma_t = \sqrt{\frac{1}{H} \sum_{i=1}^H (a_{it} - \mu_t)^2},$$

а \mathbf{g} и \mathbf{b} — параметры слоя нормализации, как и раньше.

- Это может заметно улучшить результаты рекуррентной сети.

Recurrent batch normalization

- (Cooijmans et al, 2017) – всё-таки можно batchnorm в LSTM:

$$\begin{aligned}\begin{pmatrix} \tilde{\mathbf{f}}_t \\ \tilde{\mathbf{i}}_t \\ \tilde{\mathbf{o}}_t \\ \tilde{\mathbf{g}}_t \end{pmatrix} &= \mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t + \mathbf{b} \\ \mathbf{c}_t &= \sigma(\tilde{\mathbf{f}}_t) \odot \mathbf{c}_{t-1} + \sigma(\tilde{\mathbf{i}}_t) \odot \tanh(\tilde{\mathbf{g}}_t) \\ \mathbf{h}_t &= \sigma(\tilde{\mathbf{o}}_t) \odot \tanh(\mathbf{c}_t), \\ \text{BN}(\mathbf{h}; \gamma, \beta) &= \beta + \gamma \odot \frac{\mathbf{h} - \widehat{\mathbb{E}}[\mathbf{h}]}{\sqrt{\widehat{\text{Var}}[\mathbf{h}] + \epsilon}}\end{aligned}$$

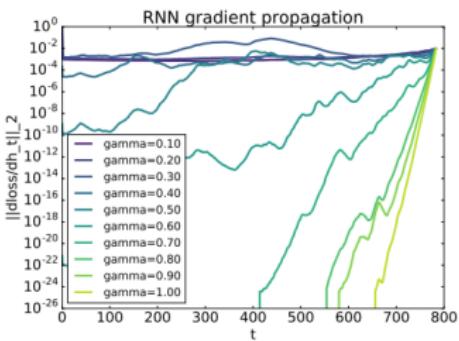
- BN для LSTM работает вот так:

$$\begin{aligned}\begin{pmatrix} \tilde{\mathbf{f}}_t \\ \tilde{\mathbf{i}}_t \\ \tilde{\mathbf{o}}_t \\ \tilde{\mathbf{g}}_t \end{pmatrix} &= \text{BN}(\mathbf{W}_h \mathbf{h}_{t-1}; \gamma_h, \beta_h) + \text{BN}(\mathbf{W}_x \mathbf{x}_t; \gamma_x, \beta_x) + \mathbf{b} \\ \mathbf{c}_t &= \sigma(\tilde{\mathbf{f}}_t) \odot \mathbf{c}_{t-1} + \sigma(\tilde{\mathbf{i}}_t) \odot \tanh(\tilde{\mathbf{g}}_t) \\ \mathbf{h}_t &= \sigma(\tilde{\mathbf{o}}_t) \odot \tanh(\text{BN}(\mathbf{c}_t; \gamma_c, \beta_c))\end{aligned}$$

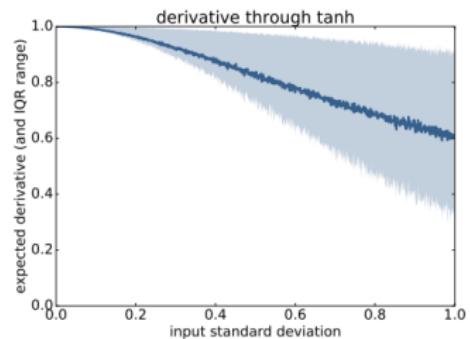
- Но где же проблемы? Почему раньше не работало?

Recurrent batch normalization

- Оказывается, параметр γ в batchnorm тут важно правильно инициализировать:



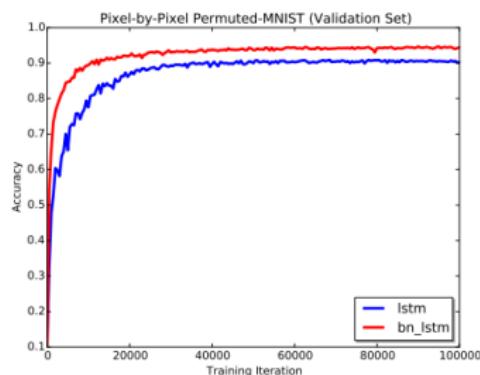
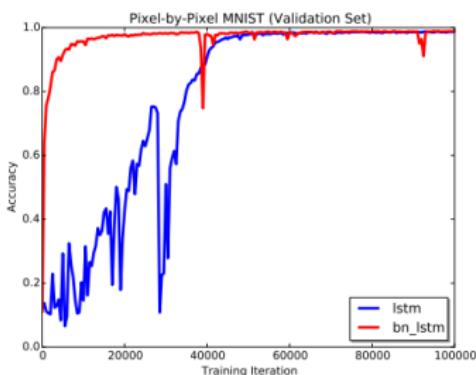
(a) We visualize the gradient flow through a batch-normalized tanh RNN as a function of γ . High variance causes vanishing gradient.



(b) We show the empirical expected derivative and interquartile range of tanh nonlinearity as a function of input variance. High variance causes saturation, which decreases the expected derivative.

Recurrent batch normalization

- И если сделать всё правильно, то получается хорошо:



Что делать с RNN на практике

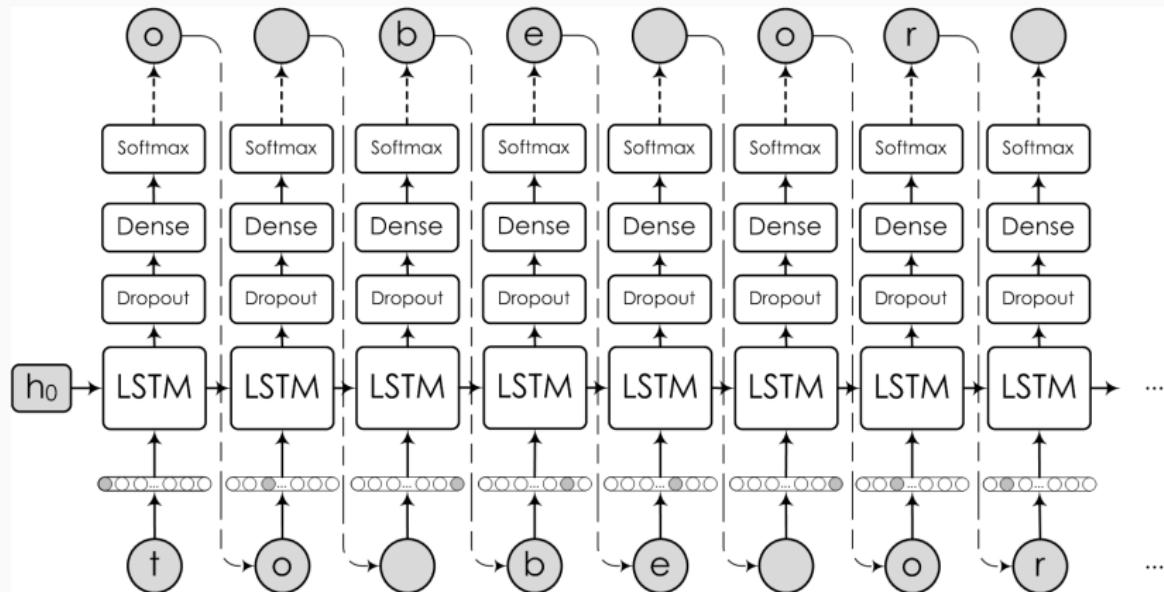
- RNN имеют довольно простую общую структуру: уровни LSTM или GRU.
- Все они выдают последовательность выходов, кроме, возможно, верхнего.
- Дропаут и batchnorm между слоями, а на рекуррентных связях надо аккуратно (потом поговорим).
- Слоёв немного; больше 3-4 трудно, 7-8 сейчас максимум.

- 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. В Keras легко: `stateful=True`.
- Начнём с простого примера анализа временных рядов...

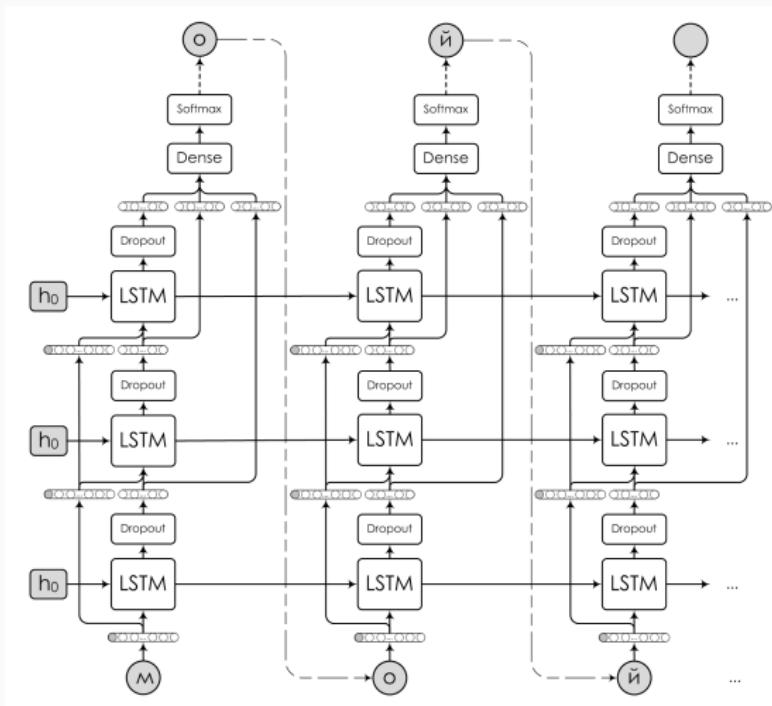
Пример: порождение текста с RNN

- 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



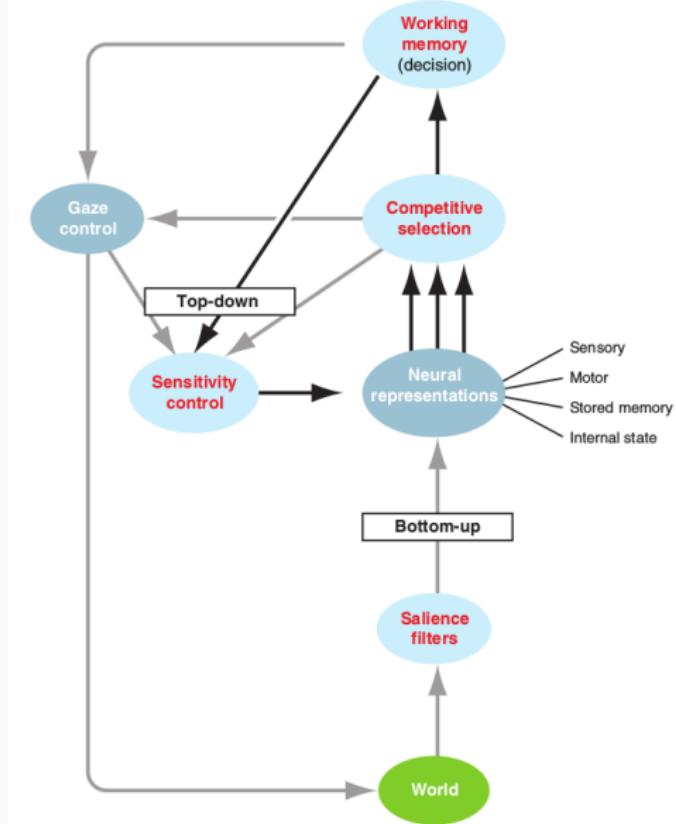
Что такое внимание?

Внимание

- Вы же сейчас внимательно меня слушаете, правильно?
- А что это значит?..
- Изображение с сетчатки тоже проходит через CNN, но потом мы часть его замечаем, а часть не особенно. Что это значит?
- Оказывается, что это довольно сложный вопрос.
- А.Р. Лурия: внимание, память и активация коры.

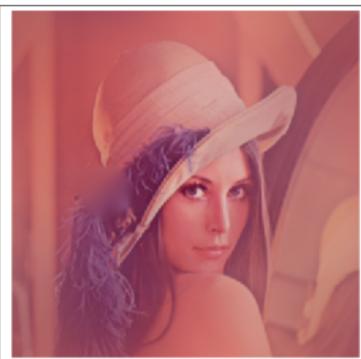
Внимание

- Дело во взаимодействии с рабочей памятью (Knudsen, 2007):

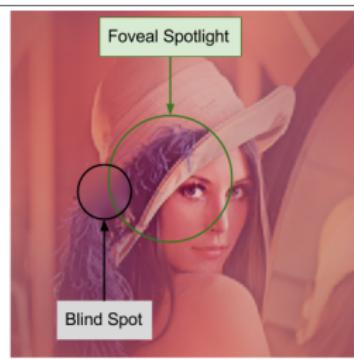


Внимание

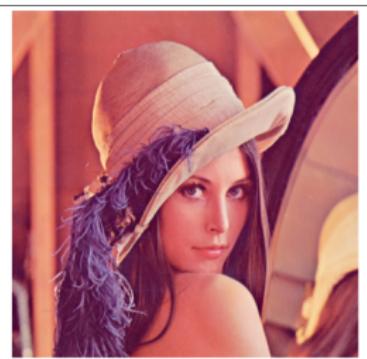
- Как это реализовать в нейронной сети? Особенno «сознательное» внимание.
- Но и «бессознательное» тоже; например, с картинками: мы же на самом деле мало чего видим в каждый момент времени.
- Центральная ямка сетчатки (fovea):



What you *actually* see



What you *actually* see (annotated)

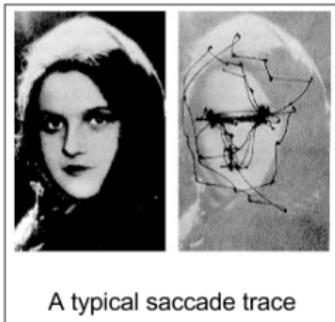


What you *think* you see

- Simons, Chabris, 1999: <https://www.youtube.com/watch?v=vJG698U2Mvo>

Внимание

- Мы делаем саккады:

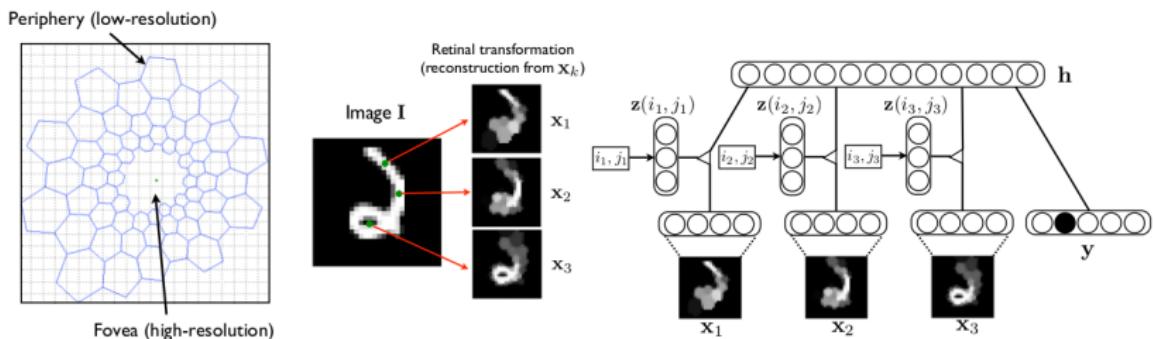


- Причём это тоже не всегда помогает:



Foveal glimpses

- В нейросети мы тоже хотим осознанно понимать, «на что» смотреть.
- Одна из первых работ (Larochelle, Hinton, 2010):

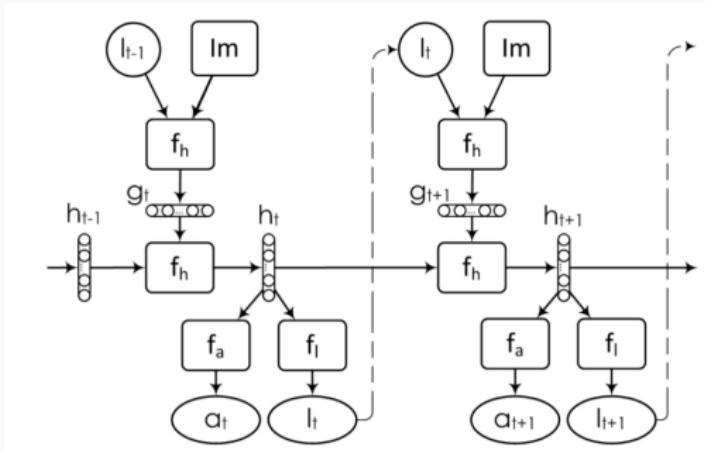


- Пытаются моделировать положения фиксаций и строить последовательность при помощи RBM.
- Последовательность – значит...

Рекуррентные модели зрительного внимания

Recurrent visual attention

- По-настоящему всё появилось в (Mnih et al., 2014), «Recurrent Models of Visual Attention»:
 - из предыдущего h_{t-1} и положения l_t для нового «взгляда» f_g делает g_t , вход для шага t ;
 - из h_{t-1} и g_t функцией f_h получается h_t ;
 - из него – «действие» $a_t = f_a(h_t)$ и положение следующего «взгляда» $l_{t+1} = f_l(h_t)$.



Recurrent visual attention

- Давайте разберёмся в модели формально:

$$\mathbf{g}_t = f_g(\mathbf{x}_t, \mathbf{l}_{t-1}; \boldsymbol{\theta}_g),$$

$$\mathbf{h}_t = f_h(\mathbf{h}_{t-1}, \mathbf{g}_t; \boldsymbol{\theta}_h),$$

$$\mathbf{l}_t \sim p(\cdot | f_l(\mathbf{h}_t; \boldsymbol{\theta}_l)),$$

$$a_t \sim p(\cdot | f_a(\mathbf{h}_t; \boldsymbol{\theta}_a)).$$

- После очередного действия получается новое наблюдение \mathbf{x}_{t+1} и награда r_t , которая будет скорее всего в конце, после всех шагов, за правильную классификацию.
- Что это напоминает?..

Recurrent visual attention

- ...о да, это reinforcement learning!
- Выучить надо стохастическую стратегию $\pi((l_t, a_t) | s_{1:t}; \theta)$, которая по истории будет выдавать следующее действие.
- У нас π задаётся через RNN, а оптимизировать надо

$$J(\theta) = \mathbb{E}_{p(s_{1:T}; \theta)} [R] = \mathbb{E}_{p(s_{1:T}; \theta)} \left[\sum_{t=1}^T r_t \right].$$

- Выглядит очень сложно – ожидание по последовательностям действий, т.е. по пространству большой размерности.
- Но есть методы – давайте сделаем preview тому, что потом будет в reinforcement learning.

Recurrent visual attention

- (Williams, 1992): алгоритм REINFORCE, в котором доказывается и используется выборочная оценка этого ожидания. Давайте выведем, а потом применим к нашему случаю...
- Нам надо оптимизировать

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=1}^T r_t(s_t, a_t) \right] \approx \frac{1}{M} \sum_{i=1}^M \sum_{t=1}^T r(s_t^{(i)}, a_t^{(i)}),$$

где мы взяли M примеров траекторий τ .

- Определим $r(\tau) = \sum_t r(s_t, a_t)$. Тогда

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} [r(\tau)] = \int \pi_\theta(\tau) r(\tau) d\tau,$$

т.е. тот самый страшный интеграл по траекториям.

- Но оказывается, что можно продифференцировать по θ ...

Recurrent visual attention

- Продифференцируем по θ :

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \int \nabla_{\theta} \pi_{\theta}(\tau) r(\tau) d\tau \\ &= \int \pi_{\theta}(\tau) \frac{\nabla_{\theta} \pi_{\theta}(\tau)}{\pi_{\theta}(\tau)} r(\tau) d\tau \\ &= \int \pi_{\theta}(\tau) \nabla_{\theta} \log \pi_{\theta}(\tau) r(\tau) d\tau \\ &= \mathbb{E}_{\tau \sim \pi_{\theta}(\tau)} [\nabla_{\theta} \log \pi_{\theta}(\tau) r(\tau)] \\ &\approx \frac{1}{M} \sum_{i=1}^M \nabla_{\theta} \log \pi_{\theta}(\tau^{(i)}) r^{(i)}(\tau),\end{aligned}$$

если приблизить выборкой; но сначала давайте ещё посмотрим на $\pi_{\theta}(\tau)$...

Recurrent visual attention

- Вероятность определяется как

$$\pi_{\theta}(\tau) = p(s_1) \prod_{t=1}^T \pi_{\theta}(a_t | s_t) p(s_{t+1} | s_t, a_t).$$

- Берём логарифм, а потом заметим, что от θ зависят только действия:

$$\begin{aligned}\nabla_{\theta} \log \pi_{\theta}(\tau) &= \\ &= \nabla_{\theta} \left(\log p(s_1) + \sum_{t=1}^T \log \pi_{\theta}(a_t | s_t) + \sum_{t=1}^T \log p(s_{t+1} | s_t, a_t) \right) = \\ &= \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t).\end{aligned}$$

Recurrent visual attention

- Итого получается вполне tractable градиент:

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \mathbb{E}_{\tau \sim \pi_{\theta}(\tau)} \left[r(\tau) \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \right] \\ &\approx \frac{1}{M} \sum_{i=1}^M r(\tau^{(i)}) \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}).\end{aligned}$$

- У нас тоже можно считать, что награда R даётся целиком:

$$\nabla_{\theta} J(\theta) \approx \frac{1}{M} \sum_{i=1}^M \sum_{t=1}^T \nabla_{\theta} \log \pi \left(l_t^{(i)}, a_t^{(i)} | s_{1:t}^{(i)}; \theta \right) R^{(i)}.$$

- Т.е. надо уметь считать $\log \pi \left(l_t^{(i)}, a_t^{(i)} | s_{1:t}^{(i)}; \theta \right)$, но в случае RNN это просто градиент сети, который можно посчитать через backpropagation.
- Ещё можно сделать частично supervised loss на последнем шаге, где мы знаем классификацию

Recurrent visual attention

- Результаты:



(a) Translated MNIST inputs.



(b) Cluttered Translated MNIST inputs.

(a) 28x28 MNIST

| Model | Error |
|---|--------------|
| FC, 2 layers (256 hiddens each) | 1.69% |
| Convolutional, 2 layers | 1.21% |
| RAM, 2 glimpses, 8×8 , 1 scale | 3.79% |
| RAM, 3 glimpses, 8×8 , 1 scale | 1.51% |
| RAM, 4 glimpses, 8×8 , 1 scale | 1.54% |
| RAM, 5 glimpses, 8×8 , 1 scale | 1.34% |
| RAM, 6 glimpses, 8×8 , 1 scale | 1.12% |
| RAM, 7 glimpses, 8×8 , 1 scale | 1.07% |

(b) 60x60 Translated MNIST

| Model | Error |
|--|--------------|
| FC, 2 layers (64 hiddens each) | 6.42% |
| FC, 2 layers (256 hiddens each) | 2.63% |
| Convolutional, 2 layers | 1.62% |
| RAM, 4 glimpses, 12×12 , 3 scales | 1.54% |
| RAM, 6 glimpses, 12×12 , 3 scales | 1.22% |
| RAM, 8 glimpses, 12×12 , 3 scales | 1.2% |

Recurrent visual attention

- Результаты:

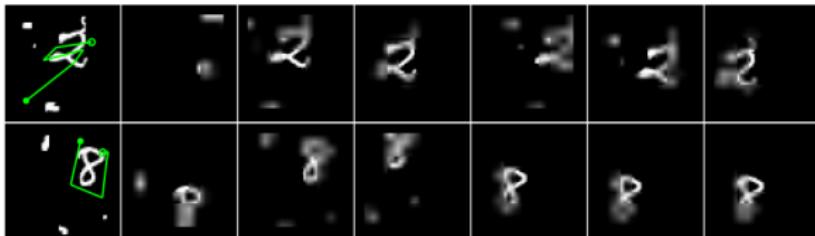
(a) 60x60 Cluttered Translated MNIST

| Model | Error |
|--|--------|
| FC, 2 layers (64 hiddens each) | 28.58% |
| FC, 2 layers (256 hiddens each) | 11.96% |
| Convolutional, 2 layers | 8.09% |
| RAM, 4 glimpses, 12×12 , 3 scales | 4.96% |
| RAM, 6 glimpses, 12×12 , 3 scales | 4.08% |
| RAM, 8 glimpses, 12×12 , 3 scales | 4.04% |
| RAM, 8 random glimpses | 14.4% |

(b) 100x100 Cluttered Translated MNIST

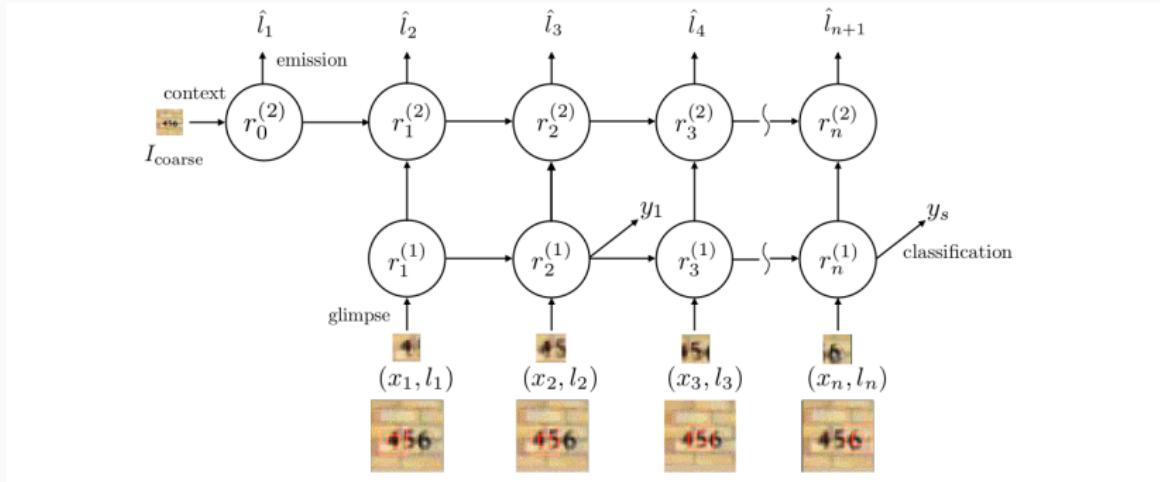
| Model | Error |
|--|--------|
| Convolutional, 2 layers | 14.35% |
| RAM, 4 glimpses, 12×12 , 4 scales | 9.41% |
| RAM, 6 glimpses, 12×12 , 4 scales | 8.31% |
| RAM, 8 glimpses, 12×12 , 4 scales | 8.11% |
| RAM, 8 random glimpses | 28.4% |

- А вот как внимание гуляет по картинке:



Recurrent visual attention

- В следующей работе (Ba et al., 2015) сделали глубокую модель:



- Кстати, обучали по-другому, вариационными методами. Как это?..

Recurrent visual attention

- Нам нужно классифицировать, т.е. $p(y | \mathbf{x}, \boldsymbol{\theta})$ максимизировать.
- Маргинализуем по положениям glimpses:

$$\log p(y | \mathbf{x}, \boldsymbol{\theta}) = \log \sum_l p(l | \mathbf{x}, \boldsymbol{\theta}) p(y | l, \mathbf{x}, \boldsymbol{\theta}).$$

- Запишем вариационную нижнюю оценку свободной энергии (как её получить?):

$$\begin{aligned} \log \sum_l p(l | \mathbf{x}, \boldsymbol{\theta}) p(y | l, \mathbf{x}, \boldsymbol{\theta}) &\geq \sum_l p(l | \mathbf{x}, \boldsymbol{\theta}) \log p(y, l | \mathbf{x}, \boldsymbol{\theta}) + H[l] \\ &= \sum_l p(l | \mathbf{x}, \boldsymbol{\theta}) \log p(y | l, \mathbf{x}, \boldsymbol{\theta}). \end{aligned}$$

Recurrent visual attention

- И теперь можно брать производные:

$$\begin{aligned}\frac{\partial J}{\partial \theta} &= \sum_l p(l | x, \theta) \frac{\partial \log p(y | l, x, \theta)}{\partial \theta} + \sum_l \log p(y | l, x, \theta) \frac{\partial p(l | x, \theta)}{\partial \theta} \\ &= \sum_l p(l | x, \theta) \left[\frac{\partial \log p(y | l, x, \theta)}{\partial \theta} + \log p(y | l, x, \theta) \frac{\partial \log p(l | x, \theta)}{\partial \theta} \right].\end{aligned}$$

- А эту сумму уже будем приближать выборкой:

$$\frac{\partial J}{\partial \theta} \approx \frac{1}{M} \sum_{i=1}^M \left[\frac{\partial \log p(y | l^{(i)}, x, \theta)}{\partial \theta} + \log p(y | l^{(i)}, x, \theta) \frac{\partial \log p(l^{(i)} | x, \theta)}{\partial \theta} \right],$$

где $l^{(i)} \sim p(l_n | x, \theta) = \mathcal{N}(l_n | \hat{l}_n, \Sigma)$.

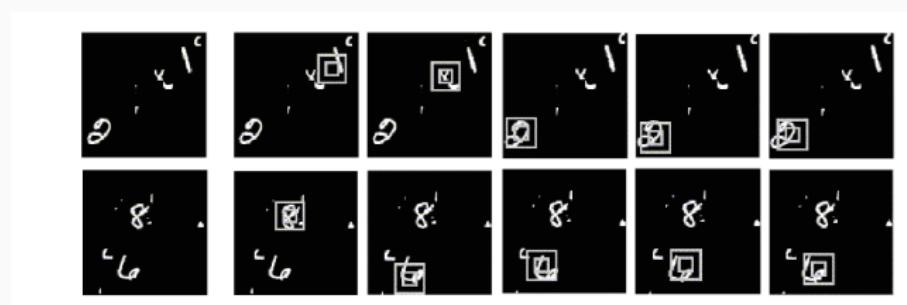
- И это уже алгоритм: сэмплируем glimpses, потом используем их в backpropagation.
- Как и в (Mnih et al., 2014), надо бы уменьшить дисперсию; для этого вычитают baseline, пока не будем углубляться.

Recurrent visual attention

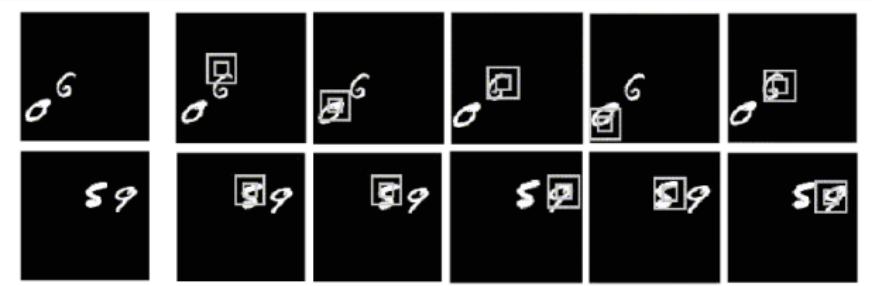
- Получился интересный результат – мы увидели, что примерно один и тот же алгоритм может получиться с двух разных сторон:
 - из обучения с подкреплением через REINFORCE;
 - из вариационной оценки собственно целевой функции.
- Важный гиперпараметр – размер glimpse, т.е. как переводить единицы измерений в системе координат glimpses в пиксели.
- То же самое легко расширить на последовательную классификацию нескольких объектов – просто фиксированное число glimpses на объект, потом классифицируем, плюс терминальное действие в конце всего.

Recurrent visual attention

- Вот как это работает на распознавании двух цифр:

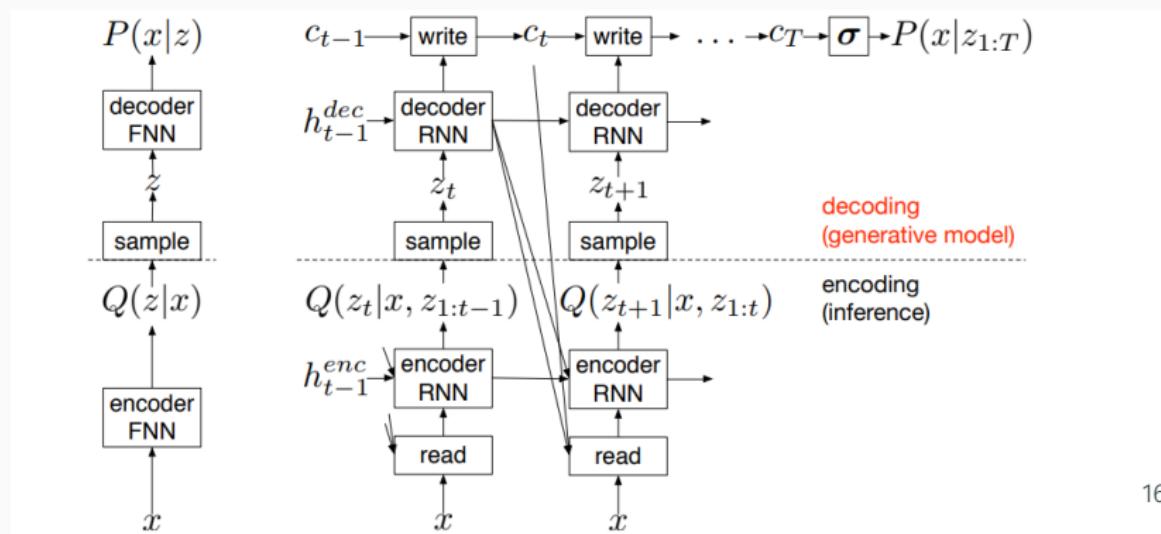


- Любопытно, что на сложении по-другому:



DRAW

- (Gregor et al., 2015): DRAW: A Recurrent Neural Network For Image Generation.
- Следующий шаг в развитии visual attention, Deep Recurrent Attentive Writer (DRAW).
- Это на самом деле вариант на тему вариационного автокодировщика



- Без особых подробностей:

$$\hat{x}_t = x - \sigma(c_{t-1})$$

$$r_t = \text{read}(x_t, \hat{x}_t, h_{t-1}^{\text{dec}})$$

$$h_t^{\text{enc}} = \text{RNN}^{\text{enc}}(h_{t-1}^{\text{enc}}, [r_t, h_{t-1}^{\text{dec}}])$$

$$z_t \sim Q(Z_t | h_t^{\text{enc}})$$

$$h_t^{\text{dec}} = \text{RNN}^{\text{dec}}(h_{t-1}^{\text{dec}}, z_t)$$

$$c_t = c_{t-1} + \text{write}(h_t^{\text{dec}})$$

- Здесь Q – гауссиан, чьи параметры выдаёт encoder, \hat{x} – error image, текущая ошибка порождения.

- Целевая функция $\mathcal{L} = \mathcal{L}^x + \mathcal{L}^z$:
 - reconstruction loss $\mathcal{L}^x = -\log p(x | c_T)$, где вероятность из распределений Бернулли;
 - latent loss $\mathcal{L}^z = \sum_{t=1}^T \text{KL}(Q(Z_t | h_t^{\text{enc}}) \| p(Z_t))$, где мы выбираем $p(Z_t)$ сами; например, для стандартного гауссиана

$$\mathcal{L}^z = \frac{1}{2} \sum_{t=1}^T \left(\mu_t^2 + \sigma_t^2 - \log \sigma_t^2 \right) - \frac{T}{2}.$$

- И можно сделать новую картинку из сети, выбрав z_t из $p(Z_t)$, а потом картинку из $p(X | c_T)$.

- Два варианта read и write.
- Без механизма внимания – это baseline:

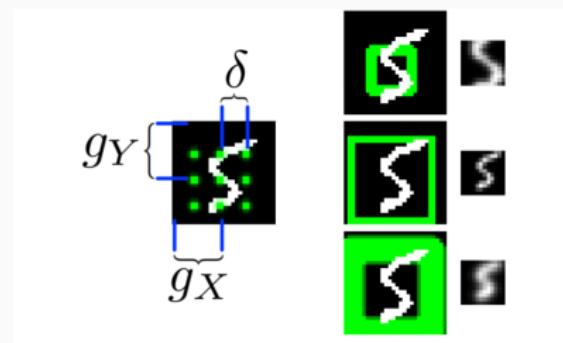
$$\text{read}(x, \hat{x}_t, h_{t-1}^{\text{dec}}) = [x, \hat{x}_t]$$

$$\text{write}(h_t^{\text{dec}}) = W(h_t^{\text{dec}})$$

- С вниманием... тут самое главное, надо сделать внимание так, чтобы сохранить дифференцируемую функцию ошибки и не надо было никакого REINFORCE.

- Давайте покроем картинку $N \times N$ прямоугольной сеткой гауссовских фильтров, т.е. выберем центр (g_X, g_Y) и шаг δ :

$$\begin{aligned}\mu_X^i &= g_X + (i - N/2 - 1/2)\delta, \\ \mu_Y^i &= g_Y + (j - N/2 - 1/2)\delta.\end{aligned}$$



- Ещё надо определить дисперсию σ^2 и интенсивность γ для фильтров.

- Все эти параметры будут из выхода декодера определяться; для картинки $A \times B$:

$$(\tilde{g}_X, \tilde{g}_Y, \log \sigma^2, \log \tilde{\delta}, \log \gamma) = W(h^{\text{dec}}),$$

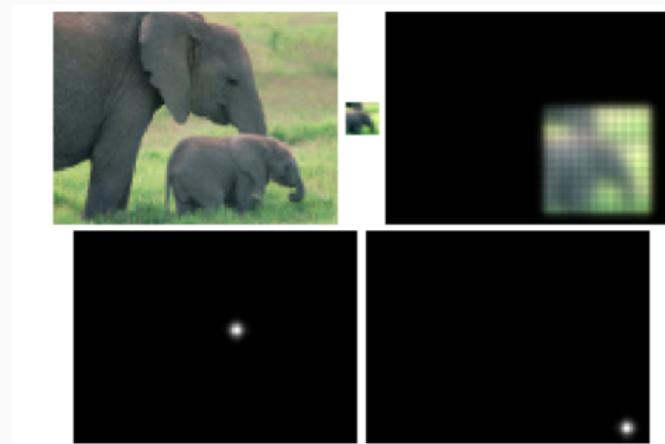
$$g_X = \frac{A+1}{2}(\tilde{g}_X + 1), \quad g_Y = \frac{B+1}{2}(\tilde{g}_Y + 1), \quad \delta = \frac{\max(A, B) - 1}{N - 1}\tilde{\delta},$$

и масштаб такой, чтобы первый участок примерно покрывал всю картинку.

- Теперь можно построить вертикальные и горизонтальные матрицы банка фильтров:

$$F_X[i, a] = \frac{1}{Z_X} e^{-\frac{(a - \mu_X^i)^2}{2\sigma^2}}, \quad F_Y[i, a] = \frac{1}{Z_Y} e^{-\frac{(b - \mu_Y^i)^2}{2\sigma^2}},$$

где (i, j) – точка в участке внимания, (a, b) – точка на картинке.



- Ну и теперь переопределим read и write:

$$\text{read}(\mathbf{x}, \hat{\mathbf{x}}_t, h_{t-1}^{\text{dec}}) = \gamma [F_Y \mathbf{x} F_X^\top, F_Y \hat{\mathbf{x}} F_X^\top],$$

$$\text{write}(h_t^{\text{dec}}) = \frac{1}{\hat{\gamma}} \hat{F}_Y^\top \mathbf{w}_t \hat{F}_X,$$

где $\mathbf{w}_t = W(h_t^{\text{dec}})$ – writing patch, выданный h_t^{dec} .

- И можно рисовать:

<https://www.youtube.com/watch?v=Zt-7MI9eKEo>

DRAW

- А вот как DRAW классифицирует:

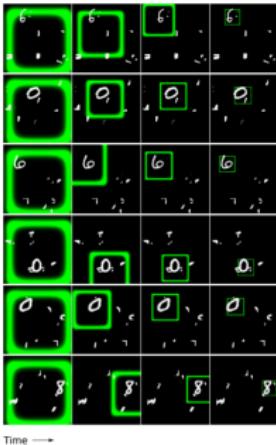


Table 1. Classification test error on 100×100 Cluttered Translated MNIST.

| Model | Error |
|--|--------------|
| Convolutional, 2 layers | 14.35% |
| RAM, 4 glimpses, 12×12 , 4 scales | 9.41% |
| RAM, 8 glimpses, 12×12 , 4 scales | 8.11% |
| Differentiable RAM, 4 glimpses, 12×12 | 4.18% |
| Differentiable RAM, 8 glimpses, 12×12 | 3.36% |

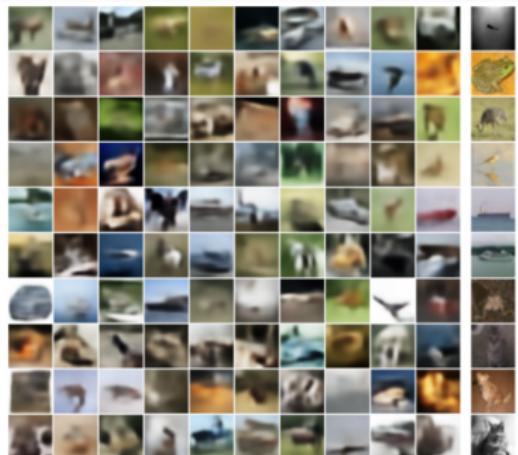
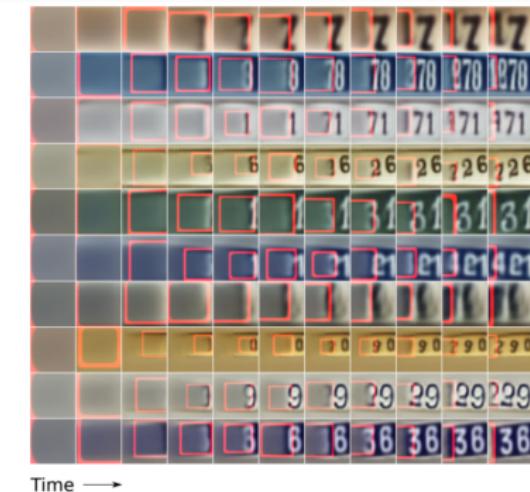
DRAW

- Примеры порождения, SVHN:



DRAW

- Примеры порождения, SVHN и CIFAR:



Enriched Deep Recurrent Visual Attention

- (Ablavatski et al., 2017): Enriched Deep Recurrent Visual Attention Model (EDRAM)
- Сначала вспомним (Jaderberg et al., 2016) – Spatial Transformer Networks, статья от DeepMind
- Мы это уже видели – слой, который может делать нужные преобразования над данными без дополнительной разметки:

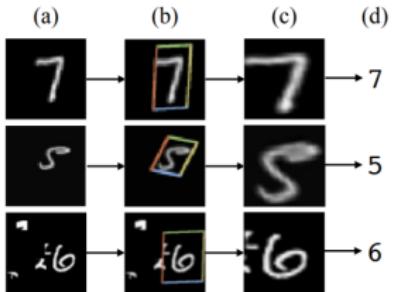


Figure 1: The result of using a spatial transformer as the first layer of a fully-connected network trained for distorted MNIST digit classification. (a) The input to the spatial transformer network is an image of an MNIST digit that is distorted with random translation, scale, rotation, and clutter. (b) The localisation network of the spatial transformer predicts a transformation to apply to the input image. (c) The output of the spatial transformer, after applying the transformation. (d) The classification prediction produced by the subsequent fully-connected network on the output of the spatial transformer. The spatial transformer network (a CNN including a spatial transformer module) is trained end-to-end with only class labels – no knowledge of the groundtruth transformations is given to the system.

Enriched Deep Recurrent Visual Attention

- Делается довольно просто: отдельная сеть обучает генератор новой сетки для карты признаков, а она применяется к карте и производит деформацию:

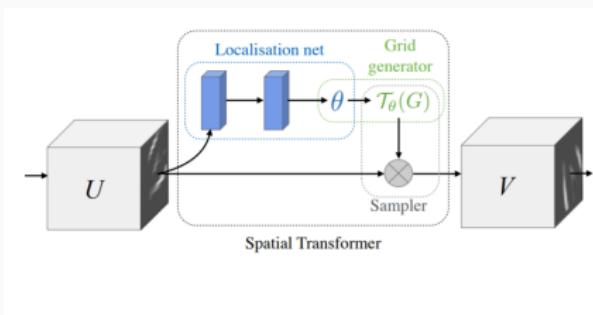
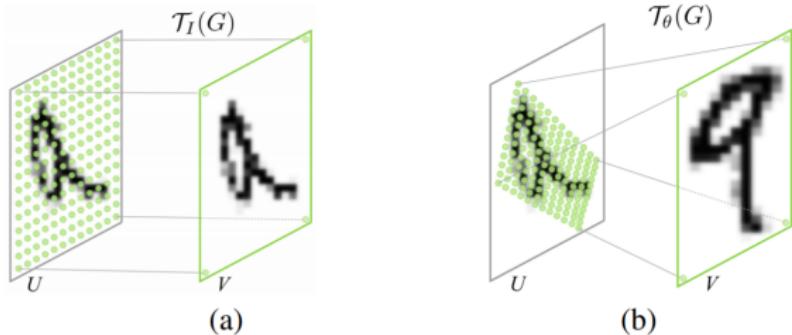


Figure 2: The architecture of a spatial transformer module. The input feature map U is passed to a localisation network which regresses the transformation parameters θ . The regular spatial grid G over V is transformed to the sampling grid $\mathcal{T}_\theta(G)$, which is applied to U as described in Sect. 3.3, producing the warped output feature map V . The combination of the localisation network and sampling mechanism defines a spatial transformer.

Enriched Deep Recurrent Visual Attention

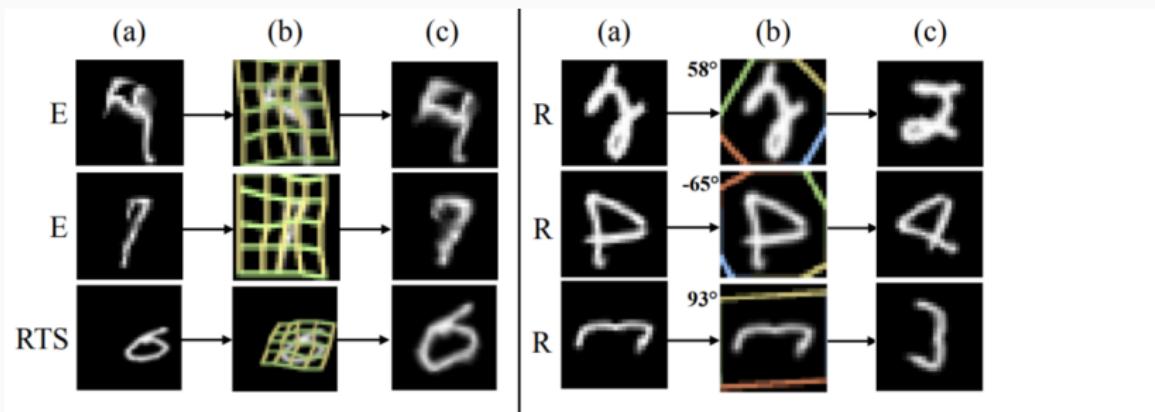
- По сути просто обучаем преобразование к сетке новых пикселей:



- Формально это опять сэмплирование из исходной сетки вокруг новых центров с неким ядром, которое делает это всё дифференцируемым.

Enriched Deep Recurrent Visual Attention

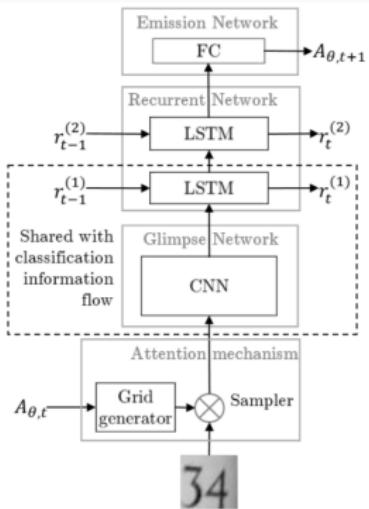
- Преобразования хорошо обучаются:



- <https://goo.gl/qdEhUu>
- <https://towardsdatascience.com/convnets-series-spatial-transformer-networks-cff4756>

Enriched Deep Recurrent Visual Attention

- Для EDRAM теперь давайте встроим Spatial Transformer в механизм внимания:



- Здесь Sampler делает билинейную интерполяцию, и через неё нормально протекают градиенты.

Enriched Deep Recurrent Visual Attention

- Функция ошибки: делаем N предсказаний на каждый из S объектов на картинке,

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^S \sum_{j=1}^N \mathcal{L}_{i,j},$$

$$\mathcal{L}_{i,j} = \alpha_1 \mathcal{L}_{i,j}^y + \alpha_2 \mathcal{L}_{i,j}^{A_\theta}, \text{ где}$$

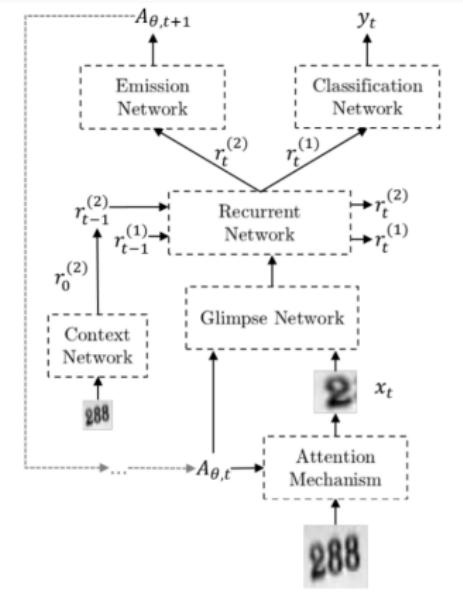
$$\mathcal{L}_{i,j}^y = -\log p_{i,j, y_{\text{true}}} \text{ (классификация),}$$

$$\mathcal{L}_{i,j}^{A_\theta} = \sum_{k=1}^6 \beta_k \left(\theta_{k,i,j} - \theta_{k,i,j}^{\text{true}} \right)^2 \text{ (взгляд),}$$

где θ – элементы матрицы A для Spatial Transformer.

Enriched Deep Recurrent Visual Attention

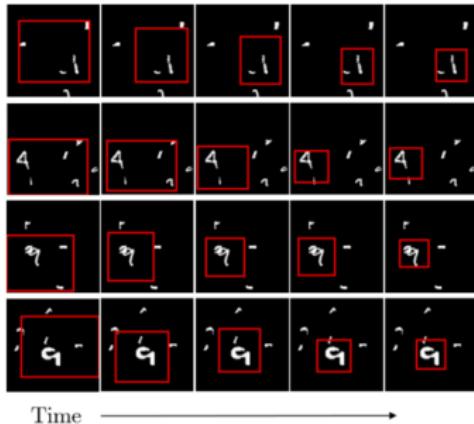
- Общая архитектура:



- Context Network – свёрточная сеть, которая берёт на вход картинку низкого разрешения и выдаёт начальный вектор для рекуррентной сети.

Enriched Deep Recurrent Visual Attention

- MNIST Cluttered:



| Model | Test Error |
|--|-------------|
| Convolutional, 2 layers | 14.35% |
| RAM [16], 8 glimpses, 12×12 , 4 scales | 8.11% |
| Differentiable RAM [7], 8 glimpses, 12×12 | 3.36% |
| ST-CNN Single [12] | 1.7% |
| DCN [1], 8 glimpses, 14×14 | 1.39% |
| Ours (6 glimpses, 26×26) | 0.6% |

Enriched Deep Recurrent Visual Attention

- SVHN:

| Model | Test Error |
|-----------------------------------|-------------|
| 11 layer CNN [6] | 3.96% |
| Single DRAM [2] | 5.1% |
| Single DRAM MC avg. [2] | 4.4% |
| forward-backward DRAM MC avg. [2] | 3.9% |
| ST-CNN Single [12] | 3.7% |
| ST-CNN Multi [12] | 3.6% |
| Ours (Single model) | 4.36% |
| Ours (forward-backward ensemble) | 3.6% |

- Интересно, что становится меньше весов:

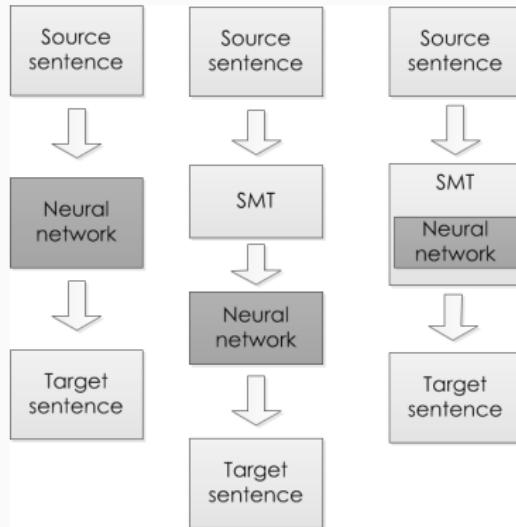
| Model | Parameters (millions) |
|-----------------------------------|-----------------------|
| 10 layer CNN | 51 |
| Single DRAM [2] | 14 |
| Single DRAM MC avg. [2] | 14 |
| forward-backward DRAM MC avg. [2] | 28 |
| ST-CNN Single [12] | 33 |
| ST-CNN Multi [12] | 37 |
| Ours (Single model) | 11 |
| Ours (forward-backward ensemble) | 22 |

Машинный перевод: encoder-decoder и внимание

- Перевод – очень хорошая задача:
 - очевидно очень практическая;
 - очевидно очень высокоуровневая, требует понимания;
 - считается довольно неплохо квантифицируемой (BLEU, TER – хотя см. выше);
 - имеет большие доступные датасеты параллельных переводов.

Машинный перевод

- Статистический машинный перевод (statistical machine translation, SMT): моделируем условную вероятность $p(y | x)$ перевода y при условии исходного текста x .
- Классический SMT: моделируем $\log p(y | x)$ линейной комбинацией признаков, строим признаки.



Машинный перевод

- Нам больше интересно моделирование sequence-to-sequence:
 - RNN естественным образом моделирует последовательность $X = (x_1, x_2, \dots, x_T)$ как $p(x_1), p(x_2 | x_1), \dots, p(x_T | x_{<T}) = p(x_T | x_{T-1}, \dots, x_1)$, и теперь $p(X)$ – это просто
$$p(X) = p(x_1)p(x_2 | x_1) \dots p(x_k | x_{<k}) \dots p(x_T | x_{<T});$$
 - так RNN и в языковых моделях используются;
 - предсказываем следующее слово на основе скрытого состояния и предыдущего слова;
- Как применить эту идею к переводу?

Метрики качества для sequence-to-sequence моделей

- Дальше будет самое интересное: машинный перевод, диалоговые модели, ответ на вопросы.
- Но как мы будем оценивать NLP-модели, которые генерируют текст?
- Есть метрики качества, которые сравнивают результат с правильными ответами:
 - BLEU (Bilingual Evaluation Understudy): перевзвешенная precision (в т.ч. для нескольких правильных ответов);
 - METEOR: гармоническое среднее precision и recall по униграммам;
 - TER (Translation Edit Rate): число исправлений между выходом и правильным ответом, делённое на среднее число слов;
 - LEPOR: комбинируем базовые факторы и метрики с настраиваемыми параметрами.
- Есть ещё куча метрик, связанных с представлениями слов и предложений (хотим поближе к правильному ответу).
- Одна только проблема...

Метрики качества для sequence-to-sequence моделей

- ...Всё это вообще не работает.

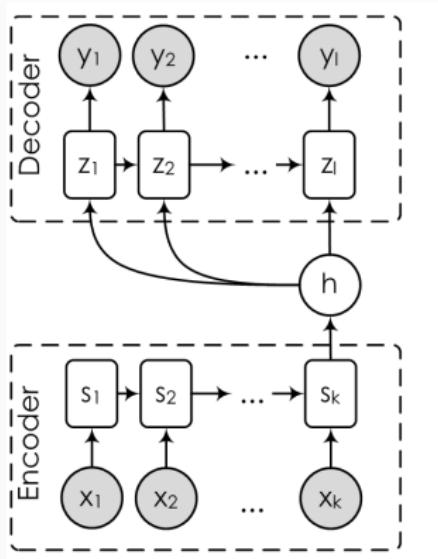
| Metric | Twitter | | | | Ubuntu | | | |
|---------|----------|---------|---------|---------|----------|---------|-----------|---------|
| | Spearman | p-value | Pearson | p-value | Spearman | p-value | Pearson | p-value |
| Greedy | 0.2119 | 0.034 | 0.1994 | 0.047 | 0.05276 | 0.6 | 0.02049 | 0.84 |
| Average | 0.2259 | 0.024 | 0.1971 | 0.049 | -0.1387 | 0.17 | -0.1631 | 0.10 |
| Extrema | 0.2103 | 0.036 | 0.1842 | 0.067 | 0.09243 | 0.36 | -0.002903 | 0.98 |
| METEOR | 0.1887 | 0.06 | 0.1927 | 0.055 | 0.06314 | 0.53 | 0.1419 | 0.16 |
| BLEU-1 | 0.1665 | 0.098 | 0.1288 | 0.2 | -0.02552 | 0.8 | 0.01929 | 0.85 |
| BLEU-2 | 0.3576 | < 0.01 | 0.3874 | < 0.01 | 0.03819 | 0.71 | 0.0586 | 0.56 |
| BLEU-3 | 0.3423 | < 0.01 | 0.1443 | 0.15 | 0.0878 | 0.38 | 0.1116 | 0.27 |
| BLEU-4 | 0.3417 | < 0.01 | 0.1392 | 0.17 | 0.1218 | 0.23 | 0.1132 | 0.26 |
| ROUGE | 0.1235 | 0.22 | 0.09714 | 0.34 | 0.05405 | 0.5933 | 0.06401 | 0.53 |
| Human | 0.9476 | < 0.01 | 1.0 | 0.0 | 0.9550 | < 0.01 | 1.0 | 0.0 |

Table 3: Correlation between each metric and human judgements for each response. Correlations shown in the human row result from randomly dividing human judges into two groups.

- Тут нужно что-то новое. И пока не совсем ясно, что именно.

Encoder-decoder архитектуры

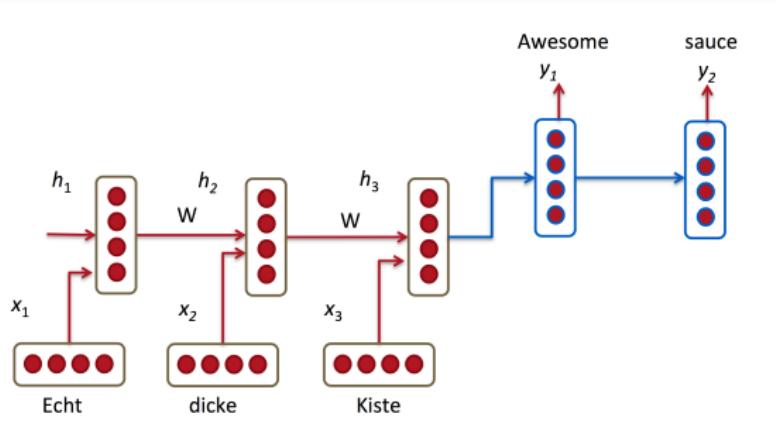
- Encoder-decoder архитектуры (Sutskever et al., 2014; Cho et al., 2014):



- Сначала кодируем, потом декодируем обратно.

Encoder-decoder архитектуры

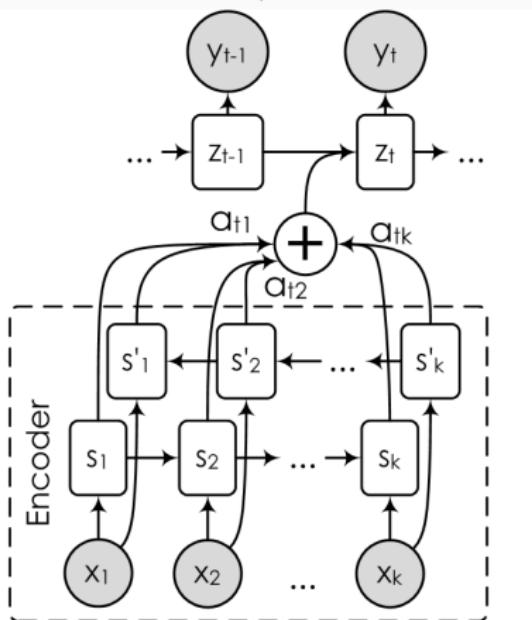
- Так же может работать и с переводом.



- Проблема: надо сжимать всё предложение в один вектор.
- С длинными участками текста это вообще перестаёт работать.

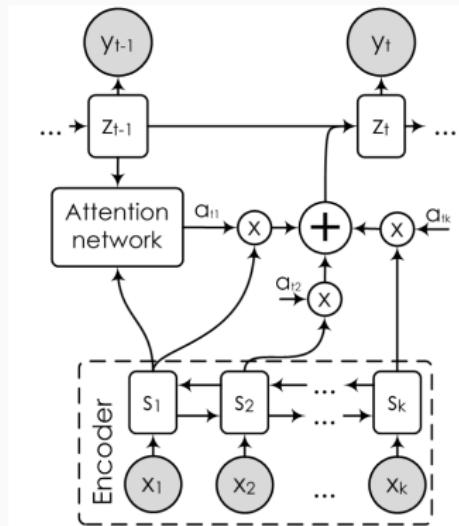
Внимание в нейронных сетях

- Решение: давайте обучим специальные веса, показывающие, насколько та или иная часть входа важна для текущего выхода.
- Прямое применение – двунаправленный LSTM плюс внимание (Bahdanau et al. 2014):



Внимание в нейронных сетях

- Мягкое внимание (soft attention) (Luong et al. 2015a; 2015b; Jean et al. 2015):
 - encoder – двунаправленная RNN, есть оба контекста;
 - сеть внимания выдаёт оценку релевантности – надо ли переводить это слово прямо сейчас?

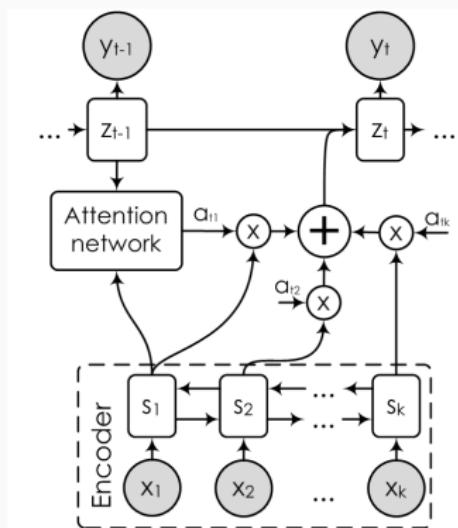


Внимание в нейронных сетях

- Формально очень просто: считаем веса внимания α_{tj} и перевзвешиваем векторы контекстов:

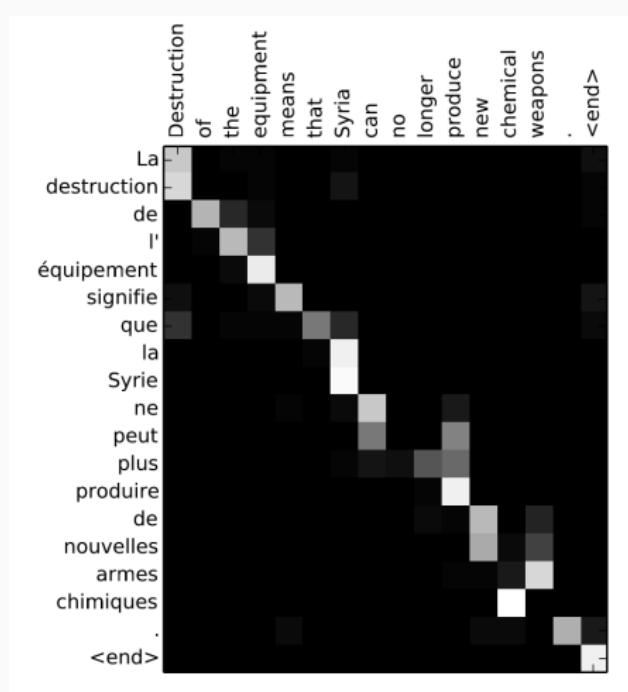
$$e_{tj} = a(z_{t-1}, j), \quad \alpha_{tj} = \text{softmax}(e_{tj}; e_{t*}),$$

$$c_t = \sum_j \alpha_{tj} h_j, \text{ и теперь } z_t = f(s_{t1}, y_{t1}, c_i).$$



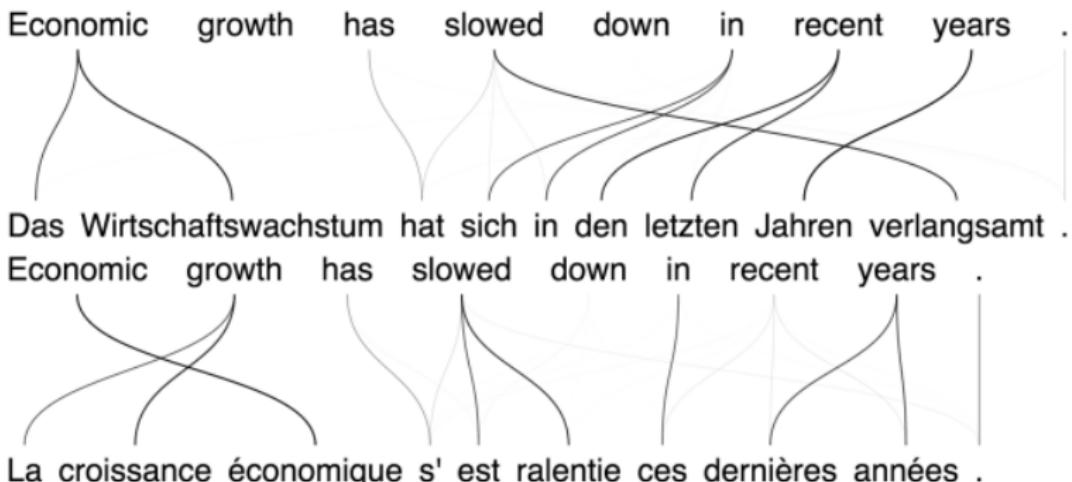
Внимание в нейронных сетях

- В результате можно визуализировать, на что смотрит сеть:



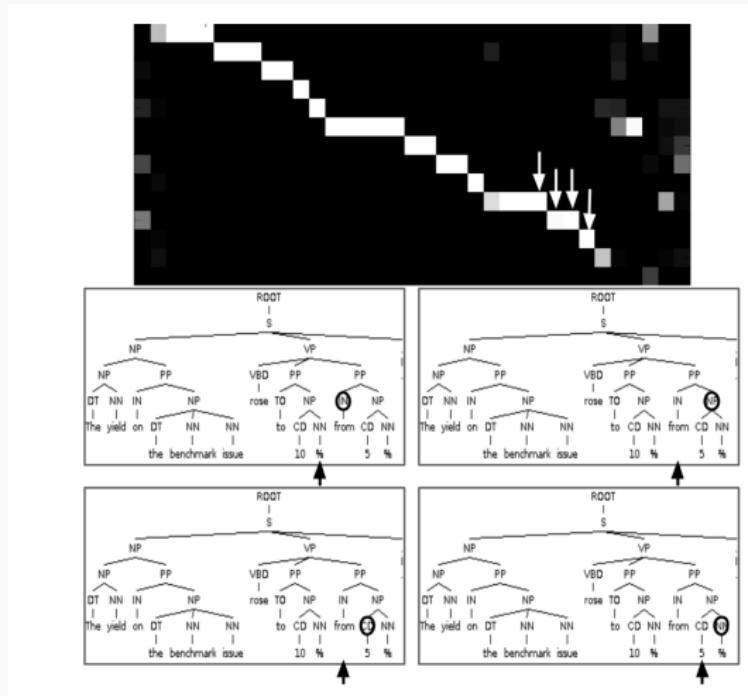
Внимание в нейронных сетях

- Получается гораздо лучше порядок слов:



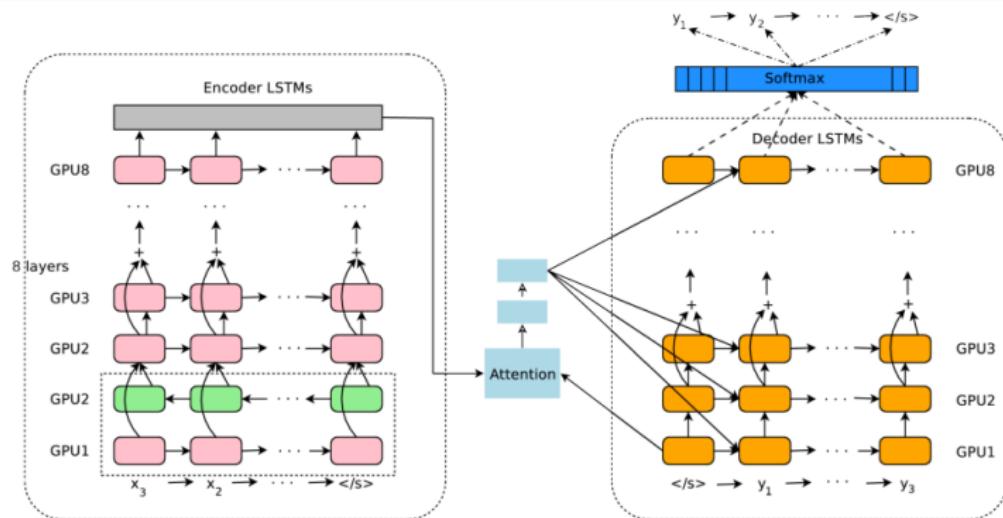
Внимание в нейронных сетях

- Другая необычная работа – «Grammar as a Foreign Language» (Vinyals et al., 2015)

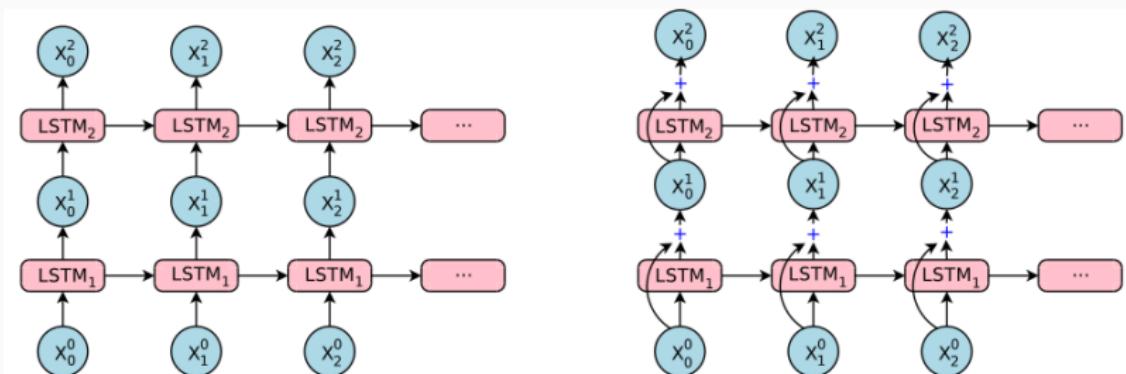


Google Translate

- Сентябрь 2016: Wu et al., *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*:
 - как на самом деле работает Google Translate;
 - базовая архитектура та же самая: encoder, decoder, attention;
 - RNN глубокие, по 8 уровней в encoder и decoder:

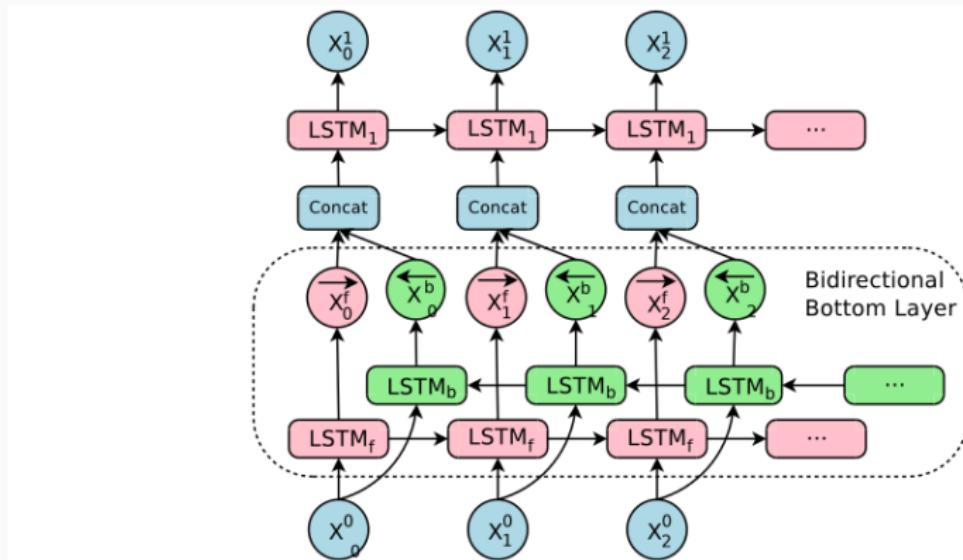


- Сентябрь 2016: Wu et al., *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation:*
 - просто stacked LSTM перестают работать далее 4-5 уровней;
 - поэтому добавляют остаточные связи, как в ResNet:



Google Translate

- Сентябрь 2016: Wu et al., *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*:
 - нижний уровень, естественно, двунаправленный:



- Сентябрь 2016: Wu et al., *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*:
- в GNMT ещё две идеи о сегментации слов:
 - *wordpiece model*: разбить слова на кусочки (отдельной моделью); пример из статьи:

Jet makers feud over seat width with big orders at stake

превращается в

_J et _makers _fe ud _over _seat _width _with _big _orders _at _stake

- *mixed word/character model*: конвертировать слова, не попадающие в словарь, в последовательность букв-токенов; пример из статьи:

Miki превращается в M <M>i <M>k <E>i

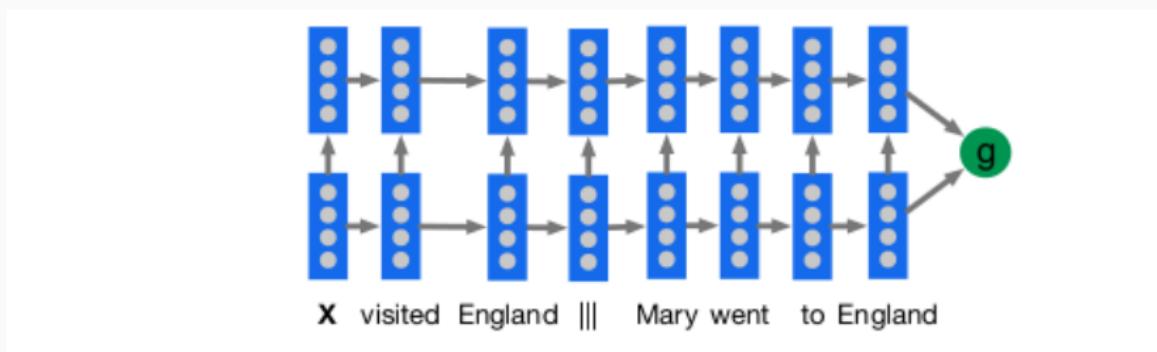
Teaching machines to read

- (Hermann et al., 2015): «Teaching machines to read and comprehend» (Google DeepMind)
- Предлагают новый способ построить датасет для понимания, автоматически создавая тройки (context, query, answer) из текстов новостей и т.п.

| Original Version | Anonymised Version |
|--|---|
| Context <p>The BBC producer allegedly struck by Jeremy Clarkson will not press charges against the “Top Gear” host, his lawyer said Friday. Clarkson, who hosted one of the most-watched television shows in the world, was dropped by the BBC Wednesday after an internal investigation by the British broadcaster found he had subjected producer Oisin Tymon “to an unprovoked physical and verbal attack.” ...</p> | <p>the <i>ent381</i> producer allegedly struck by <i>ent212</i> will not press charges against the “ <i>ent153</i> ” host , his lawyer said friday . <i>ent212</i> , who hosted one of the most - watched television shows in the world , was dropped by the <i>ent381</i> wednesday after an internal investigation by the <i>ent180</i> broadcaster found he had subjected producer <i>ent193</i> “ to an unprovoked physical and verbal attack . ” ...</p> |
| Query <p>Producer X will not press charges against Jeremy Clarkson, his lawyer says.</p> | <p>producer X will not press charges against <i>ent212</i> , his lawyer says .</p> |
| Answer <p>Oisin Tymon</p> | <p><i>ent193</i></p> |

Teaching machines to read

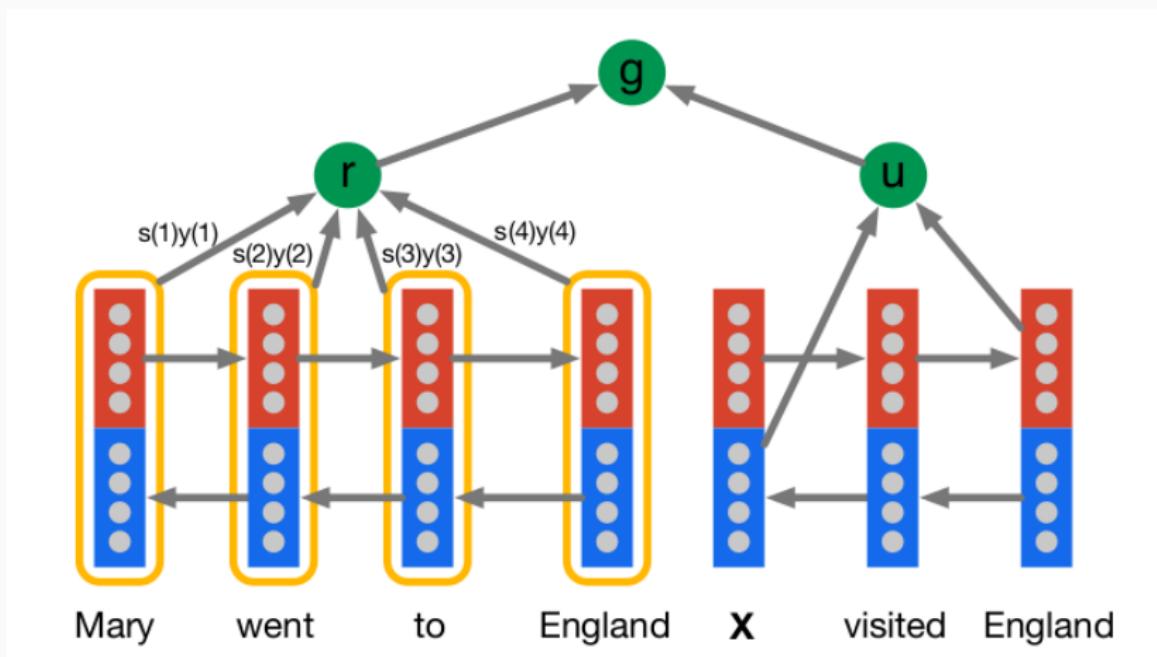
- Базовая модель – глубокий LSTM:



- Но так, конечно, совсем плохо работает.

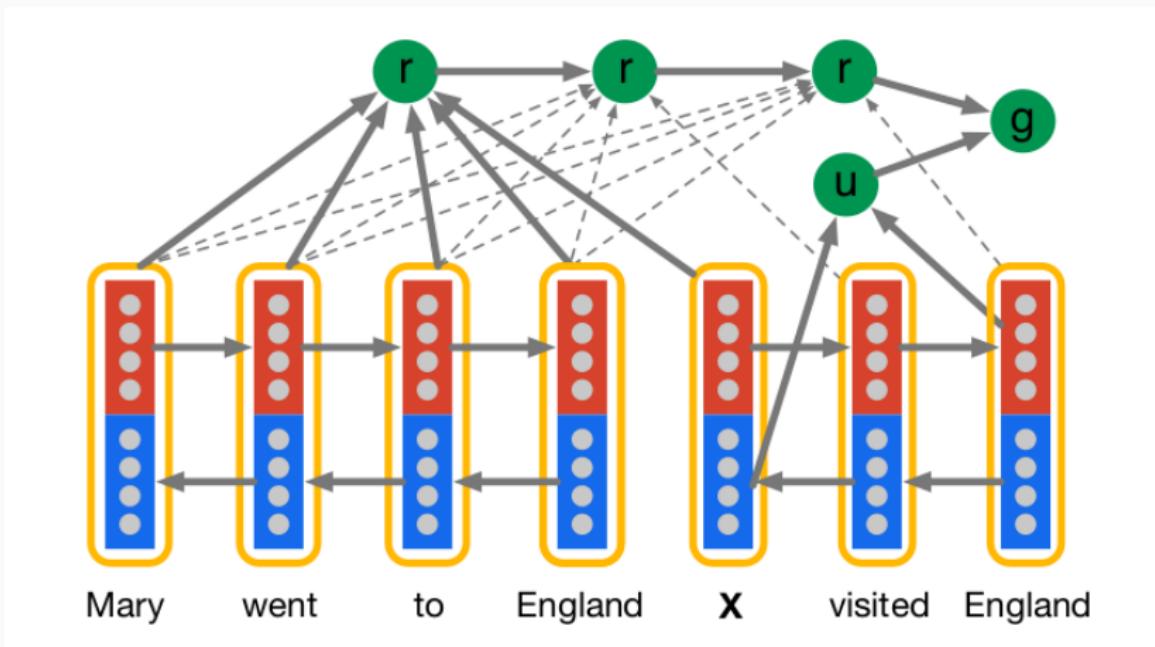
Teaching machines to read

- Attentive Reader: обучаемся, на какую часть документа смотреть



Teaching machines to read

- Impatient Reader: можем перечитывать нужные части документа по мере прочтения запроса



Teaching machines to read

- Получаются разумные карты внимания:

by ent423 ,ent261 correspondent updated 9:49 pm et ,thu march 19 ,2015 (ent261) a ent114 was killed in a parachute accident in ent45 ,ent85 ,near ent312 ,a ent119 official told ent261 on wednesday .he was identified thursday as special warfare operator 3rd class ent23 ,29 ,of ent187 ,ent265 .`` ent23 distinguished himself consistently throughout his career .he was the epitome of the quiet professional in all facets of his life ,and he leaves an inspiring legacy of natural tenacity and focused

...

ent119 identifies deceased sailor as X ,who leaves behind a wife

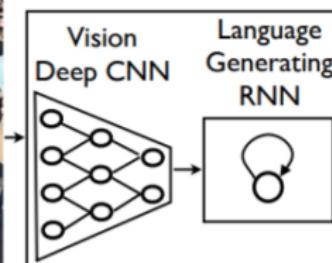
by ent270 ,ent223 updated 9:35 am et ,mon march 2 ,2015 (ent223) ent63 went familial for fall at its fashion show in ent231 on sunday ,dedicating its collection to `` mamma '' with nary a pair of `` mom jeans '' in sight .ent164 and ent21 ,who are behind the ent196 brand ,sent models down the runway in decidedly feminine dresses and skirts adorned with roses ,lace and even embroidered doodles by the designers 'own nieces and nephews .many of the looks featured saccharine needlework phrases like `` ilove you ,

...

X dedicated their fall fashion show to moms

Show, Attend, and Tell

- Теперь давайте про подписи к картинкам.
- Сначала было «Show and Tell» (Vinyals et al., 2015):

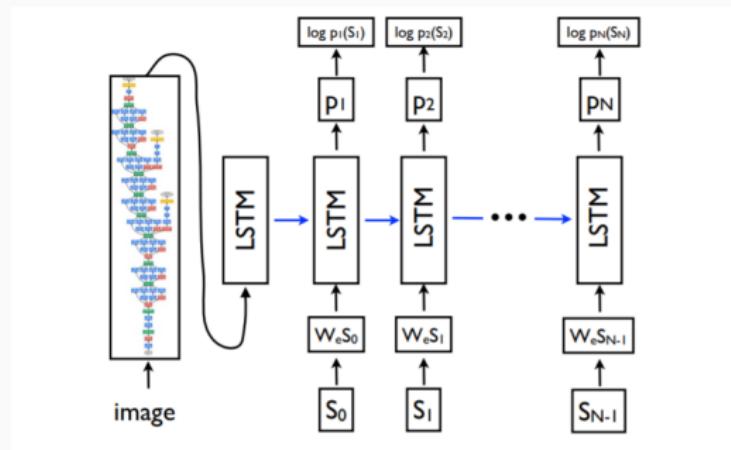


**A group of people
shopping at an
outdoor market.**

**There are many
vegetables at the
fruit stand.**

Show, Attend, and Tell

- Довольно прямолинейная архитектура:
 - целевая функция – это просто $\sum_{(I,S)} \log p(S | I; \theta)$, где I – картинка, S – описание;
 - раскладываем и моделируем $p(S_t | I, S_0, \dots, S_{t-1})$ рекуррентной сетью с LSTM;
 - а CNN используем, чтобы извлечь признаки.



Show, Attend, and Tell

- Получалось хорошо, но можно лучше:

A person riding a motorcycle on a dirt road.



Two dogs play in the grass.



A skateboarder does a trick on a ramp.



A dog is jumping to catch a frisbee.



A group of young people playing a game of frisbee.



Two hockey players are fighting over the puck.



A little girl in a pink hat is blowing bubbles.



A refrigerator filled with lots of food and drinks.



A herd of elephants walking across a dry grass field.



A close up of a cat laying on a couch.



A red motorcycle parked on the side of the road.



A yellow school bus parked in a parking lot.



Describes without errors

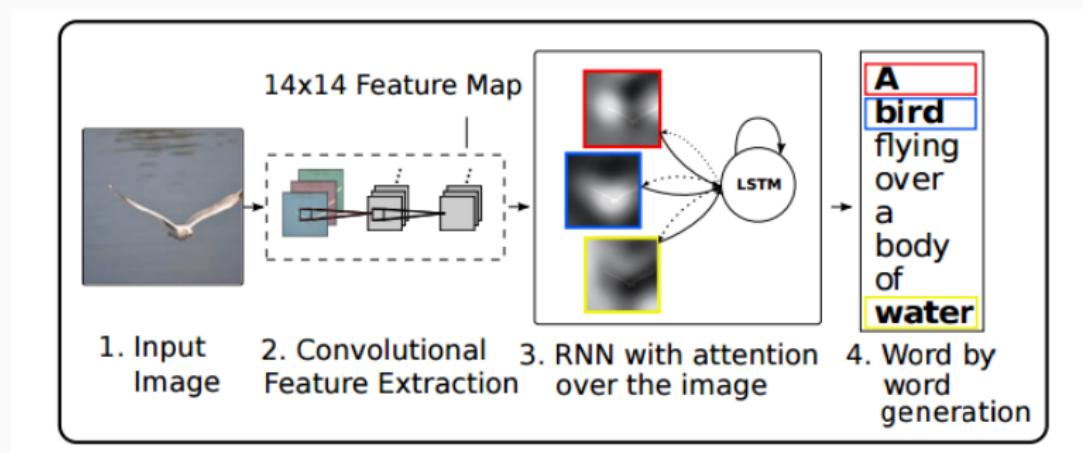
Describes with minor errors

Somewhat related to the image

Unrelated to the image

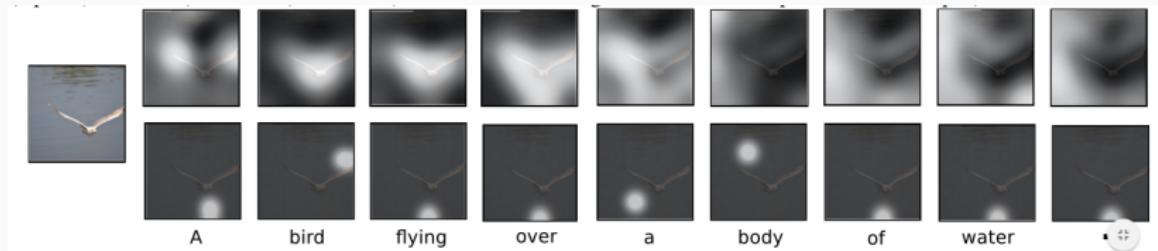
Show, Attend, and Tell

- Из этого появилось «Show, Attend, and Tell» (Xu et al., 2015)



Show, Attend, and Tell

- Soft attention vs. hard attention (стохастически выбираем однозначный кусок картинки).



- Soft attention – строим аннотацию с весами

$$\phi(\{\mathbf{a}\}_i, \{\alpha_i\}) = \sum_{i=1}^L \alpha_{t,i} \mathbf{a}_i.$$

Show, Attend, and Tell

- Hard attention обучается максимизацией вариационной нижней оценки

$$L_s = \sum_s p(s | a) \log p(y | s, a) \leq \log \sum_s p(s | a)p(y | s, a) = \log p(y | a).$$

- От L_s можно брать производные:

$$\frac{\partial L_s}{\partial W} = \sum_s p(s | a) \left[\frac{\partial \log p(y | s, a)}{\partial W} + \log p(y | s, a) \frac{\partial \log p(s | a)}{\partial W} \right].$$

- И дальше сэмплируем s_t с вероятностями α_i и приближаем ожидание выборкой.
- Опять те же трюки, вычитаем baseline, всё такое.

Show, Attend, and Tell

- Часто получаются очень хорошие результаты:



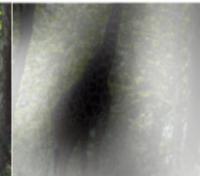
A woman is throwing a frisbee in a park.



A little girl sitting on a bed with a teddy bear.



A group of people sitting on a boat in the water.



A giraffe standing in a forest with trees in the background.

- А когда плохие, можно посмотреть почему.

Show, Attend, and Tell

- Примеры – hard attention:



(a) A man and a woman playing frisbee in a field.

Show, Attend, and Tell

- Примеры – soft attention:



(b) A woman is throwing a frisbee in a park.

Show, Attend, and Tell

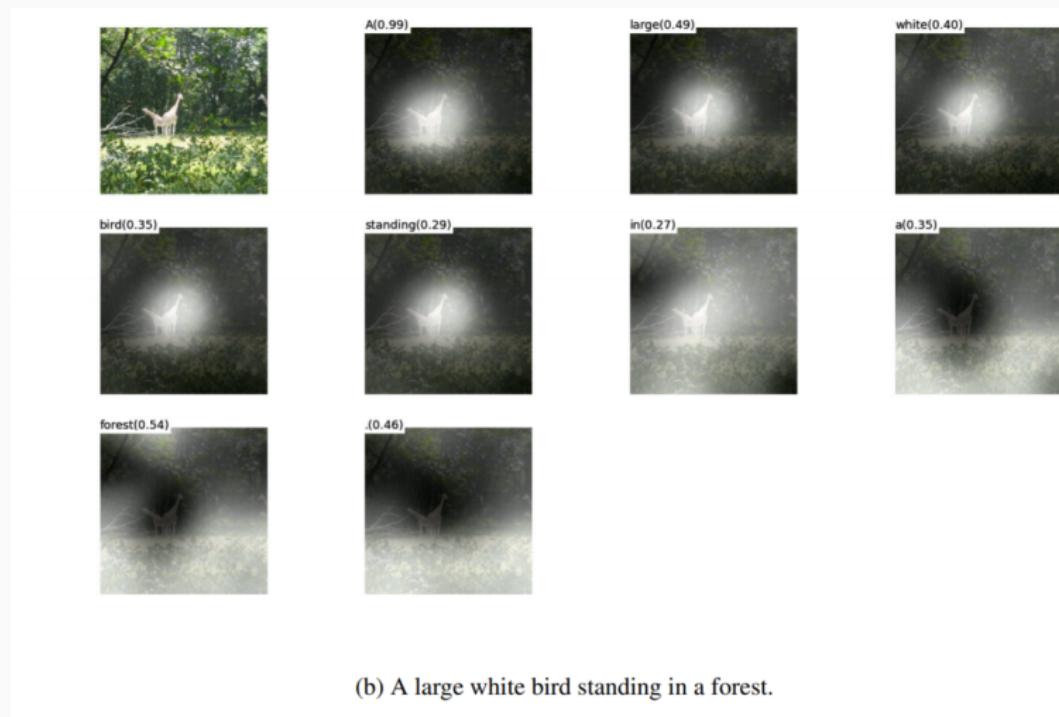
- Примеры – hard attention:



(a) A giraffe standing in the field with trees.

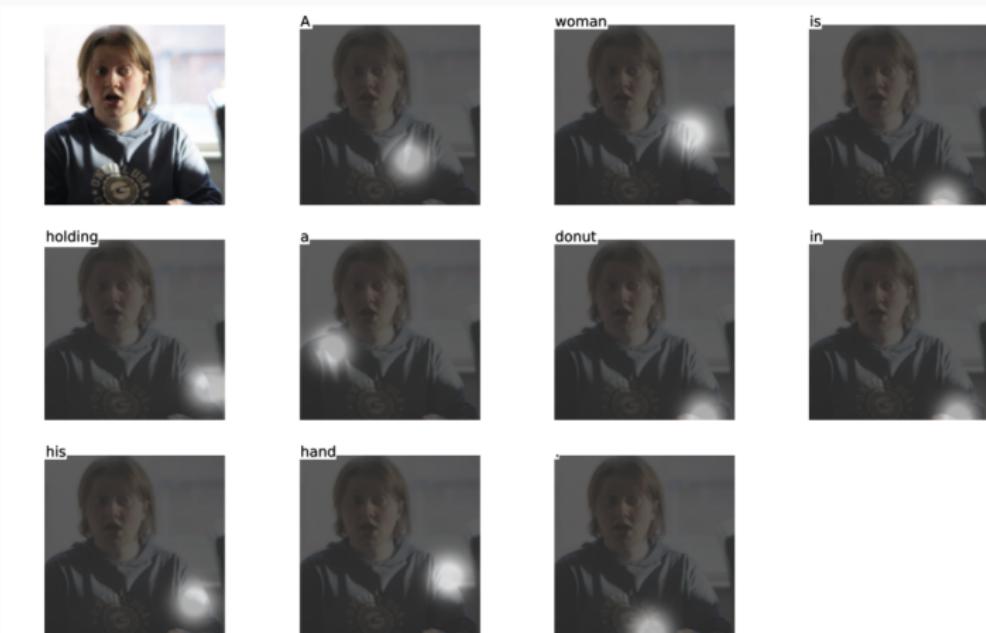
Show, Attend, and Tell

- Примеры – soft attention:



Show, Attend, and Tell

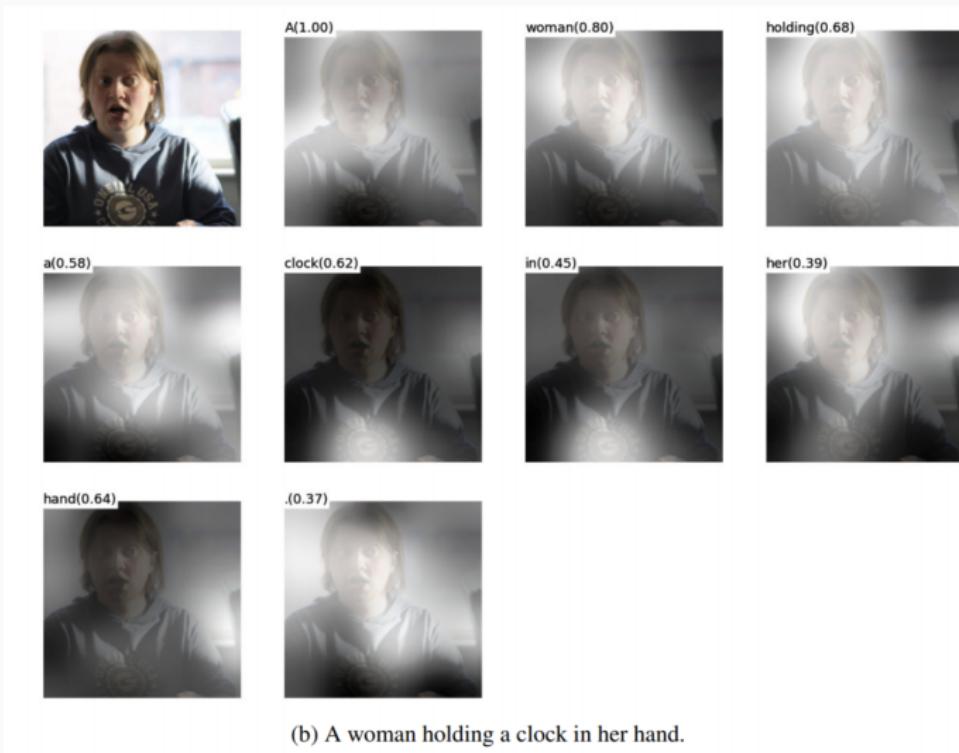
- Примеры – hard attention:



(a) A woman is holding a donut in his hand.

Show, Attend, and Tell

- Примеры – soft attention:



Спасибо!

Спасибо за внимание!

