

DEEP LEARNING

Рекуррентные сети

Артем Грачев, Святослав Елизаров, Коваленко Борис

28 ноября 2017

Высшая школа экономики

WORD2VEC

Проблема:

У вас есть корпус текста. Как его представить для обработки на компьютере?

WORD2VEC

Проблема:

У вас есть корпус текста. Как его представить для обработки на компьютере?

motel [0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] AND
hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0] = 0

WORD2VEC

Скажи мне, кто твои друзья, и я скажу, кто ты...

government debt problems turning into banking crises as has happened in
saying that Europe needs unified banking regulation to replace the hodgepodge

↖ These words will represent *banking* ↗

WORD2VEC

Как закодировать такое представление?

WORD2VEC

Как закодировать такое представление?

- I like deep learning.
- I like NLP.
- I enjoy flying.

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

WORD2VEC

Какие проблемы у такой схемы?

WORD2VEC

Какие проблемы у такой схемы?

1. Размерность растет с ростом словаря
2. Нужно много памяти
3. Модели построенные на таких векторах должны быть адаптированы к разреженным векторам

WORD2VEC

Мы хотим иметь dense представление для слов. Как этого добиться?

WORD2VEC

Мы хотим иметь dense представление для слов. Как этого добиться?

SVD на cooccurrence matrix

$$\begin{matrix} & m \\ n \end{matrix} \quad = \quad \begin{matrix} & r \\ n \end{matrix} \begin{matrix} U_1 | U_2 | U_3 | \dots \\ \hline \end{matrix} \quad \begin{matrix} & r \\ r \end{matrix} \begin{matrix} S_1 & S_2 & S_3 & \dots & 0 \\ 0 & \ddots & & & S_r \end{matrix} \quad \begin{matrix} & m \\ r \end{matrix} \begin{matrix} V_1 \\ V_2 \\ V_3 \\ \vdots \end{matrix}$$

X U S V^T

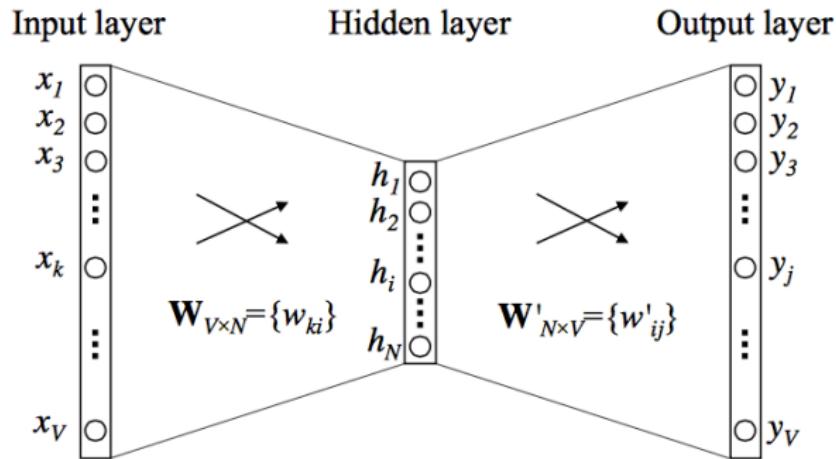
WORD2VEC

Сделаем классификатор: он будет предсказывать слова соседи, для данного слова.

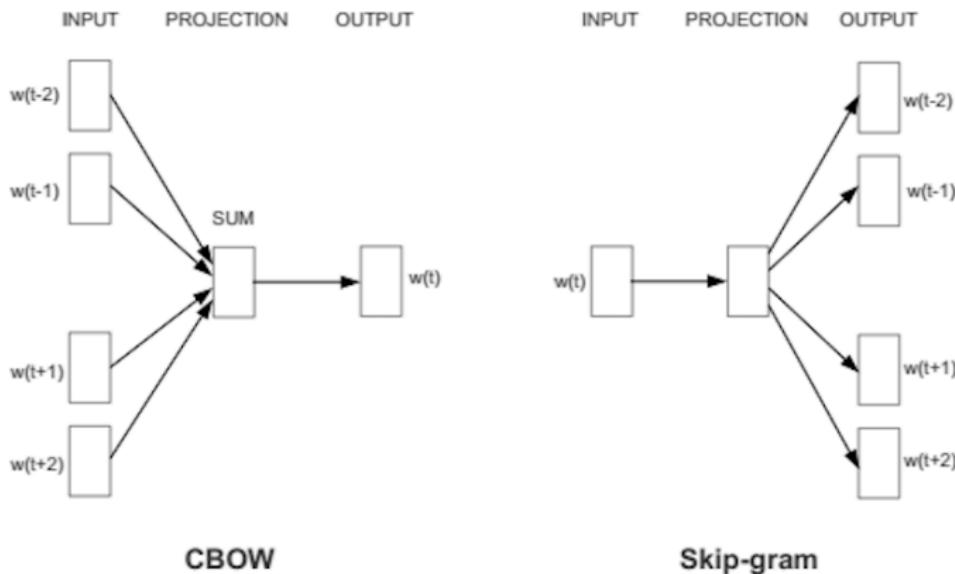
$$L(\theta) = \frac{1}{T} \sum_{t=1}^T \sum_{-m < j < m, j \neq 0} \log(p(w_{t+j} | w_t))$$

$$p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_i \exp(u_i^T v_c)}$$

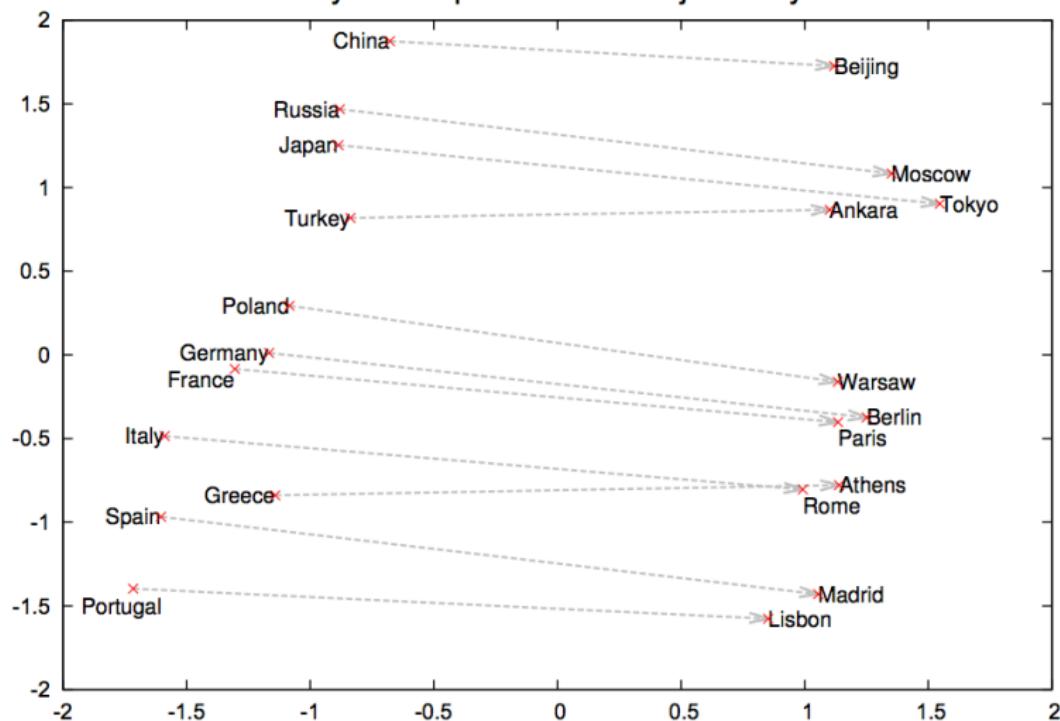
WORD2VEC



WORD2VEC



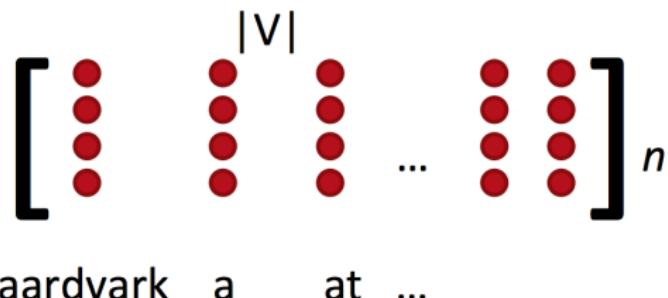
WORD2VEC



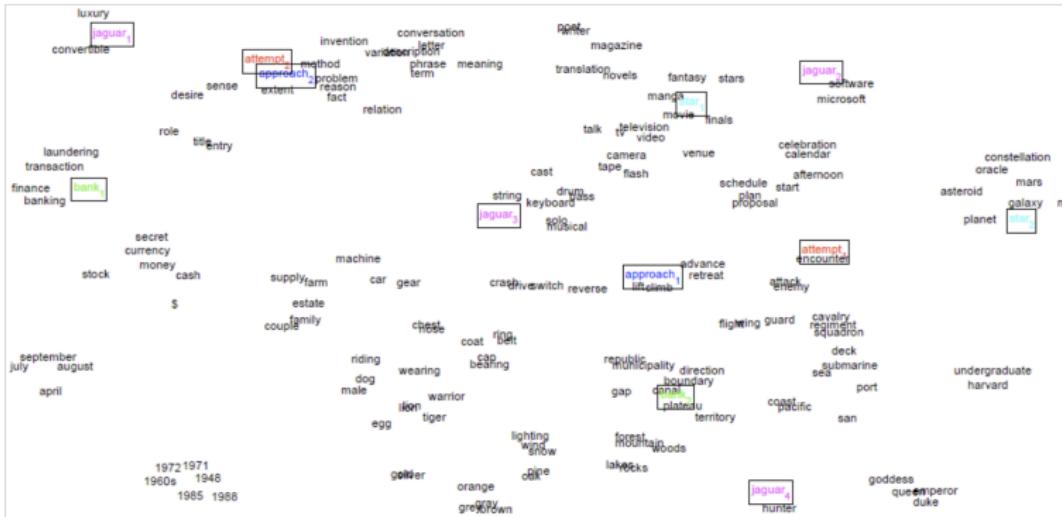
WORD2VEC

Embedding matrix (Look-up table):

`tf.nn.embedding_lookup`



WORD2VEC



WORD2VEC

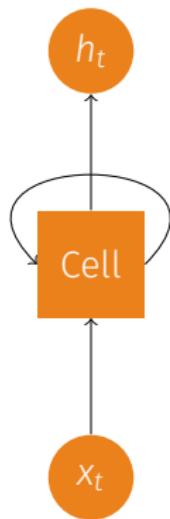
LSA, HAL (Lund & Burgess),
COALS (Rohde et al),
Hellinger-PCA (Lebret & Collobert)

- Fast training
- Efficient usage of statistics
- Primarily used to capture word similarity
- Disproportionate importance given to large counts

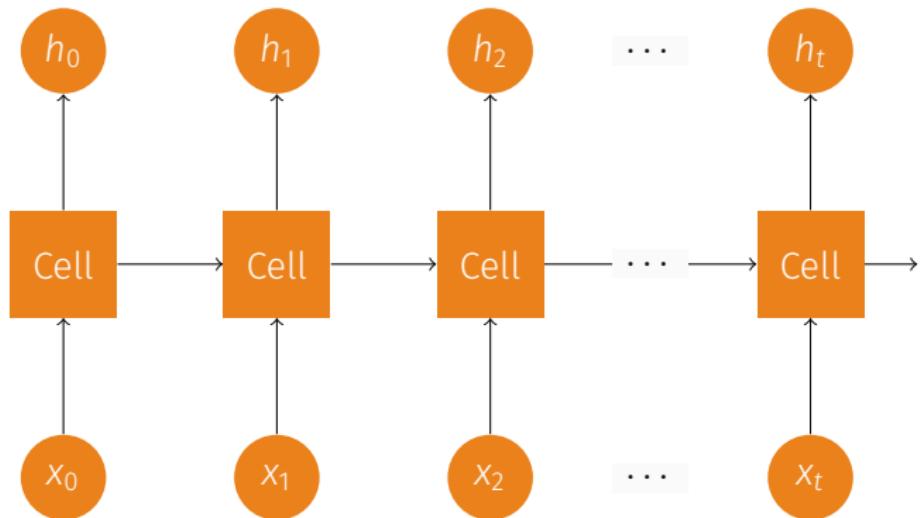
• NNLM, HLBL, RNN, Skip-gram/CBOW, (Bengio et al; Collobert & Weston; Huang et al; Mnih & Hinton; Mikolov et al; Mnih & Kavukcuoglu)

- Scales with corpus size
- Inefficient usage of statistics
- Generate improved performance on other tasks
- Can capture complex patterns beyond word similarity

RNN



Ячейка рекуррентной сети



Размотанная сеть

RNN

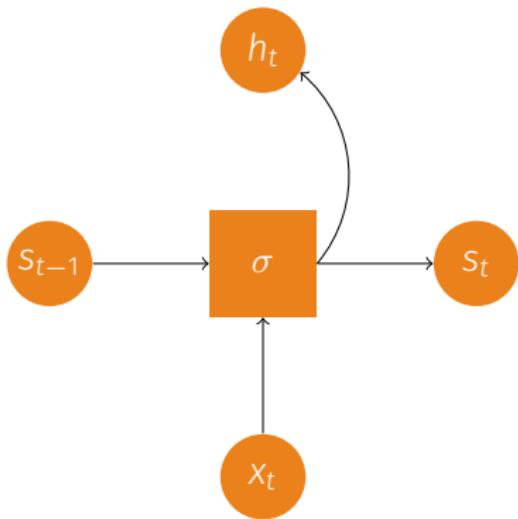
$$h_t = \sigma(Ws_{t-1} + Ux_t)$$

Где h_t – состояние на шаге t

W и U – векторы весов

σ – функция активации

RNN



- Обычная RNN преобразует своё состояние на каждом шаге
- Допустим, информация на текущем шаге не была полезной, и необходимо чтобы RNN сохранил предыдущее состояние без изменений. Т.е. $s_t = s_{t-1}$
- Следовательно

$$s_t = \sigma(Ws_{t-1})$$

должна быть тождественной функций, но тождественная функция линейна в отличие от приведённой ранее. Возникает противоречие.

- Даже тривиальная задача – поддерживать $s_t = x_t$ невозможна для наивной RNN!

BACK PROPAGATION THROUGH TIME

- Сеть разворачивается в зависимости от длины последовательности
- Используется обычный алгоритм обратного распространения
- Так как в каждом слое используются одинаковые веса, частные производные суммируются

ВЫЧИСЛЕНИЕ ГРАДИЕНТА

- Как мы помним из предыдущей лекции последовательное использование сигмоида или гиперболического тангенса может привести к затуханию градиента
- Доказано, что градиент в обычной RNN затухает при стремлении последовательности к бесконечности. [Pascanu et al. 2013 On the difficulty of training recurrent neural networks]
- Можно попробовать подобрать “правильно” инициализировать параметры [Glorot, Bengio. 2013 Understanding the difficulty of training deep feedforward neural networks]. Но это не решит проблему.

ВЫЧИСЛЕНИЕ ГРАДИЕНТА

- Затухающий градиент и общие параметры приводят к нестабильности
- В случае сетей с разными весами для каждого слоя, у модели есть возможность адаптироваться к тому, что градиент становится меньше.
- Нижние слои посыпают грубые сигналы, верхние учатся их интерпретировать.

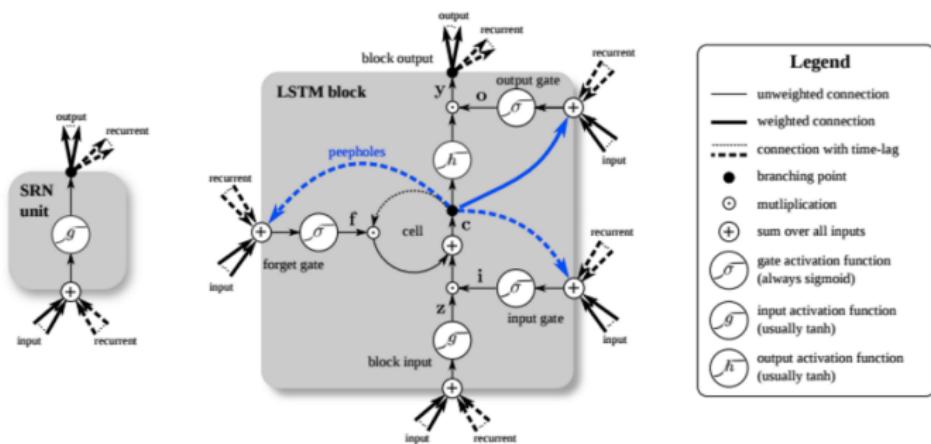
ВЫЧИСЛЕНИЕ ГРАДИЕНТА

- Когда все слои имеют общие веса это невозможно. Более того, это может приводить к конфликтам: градиент на верхних слоях положительный, а на нижних отрицательный, но меньше по модулю. Другими словами нижние слои разучиваются быстрее чем могут учиться.
- Back propagation through time слишком чувствительна к недавним событиям
- Из вычислительных соображений ВРТТ часто ограничивают в глубину

LONG SHORT-TERM MEMORY

- Состояние пробрасывается от входа к выходу “как есть”
- Ячейка может вносить только инкрементальные изменения
 $s_{t+1} = s_t + \Delta s_{t+1}$
- Так как функция активации тождественная, градиент равен константе

LSTM



LSTM

Уравнение для одного слоя LSTM:

$$i_l^t = \sigma \left[W_l^i x_{l-1}^t + V_l^i x_l^{t-1} + b_l^i + D(v_l^i) c_l^{t-1} \right] \quad \text{input gate} \quad (1)$$

$$f_l^t = \sigma \left[W_l^f x_{l-1}^t + V_l^f x_l^{t-1} + b_l^f + D(v_l^f) c_l^{t-1} \right] \quad \text{forget gate} \quad (2)$$

$$c_l^t = f_l^t \cdot c_l^{t-1} + i_l^t \tanh \left[W_l^c x_{l-1}^t + U_l^c x_l^{t-1} + b_l^c \right] \quad \text{cell state} \quad (3)$$

$$o_l^t = \sigma \left[W_l^o x_{l-1}^t + V_l^o x_l^{t-1} + b_l^o + D(v_l^o) c_l^{t-1} \right] \quad \text{output gate} \quad (4)$$

$$x_l^t = o_l^t \cdot \tanh[c_l^t] \quad (5)$$

$$t \in 1, \dots, N; l \in 1, \dots, L.$$

Заметим, что в данной формуле присутствуют так называемые **peephole connections** — это диагональные матрицы $D(v_l)$

GRU

Другое расширение RNN – это GRU (Gated Recurrent Unit)

$$z_l^t = \sigma \left[W_l^z x_{l-1}^t + V_l^z x_l^{t-1} + b_l^i \right] \quad \text{input gate} \quad (6)$$

$$r_l^t = \sigma \left[W_l^r x_{l-1}^t + V_l^r x_l^{t-1} + b_l^f \right] \quad \text{forget gate} \quad (7)$$

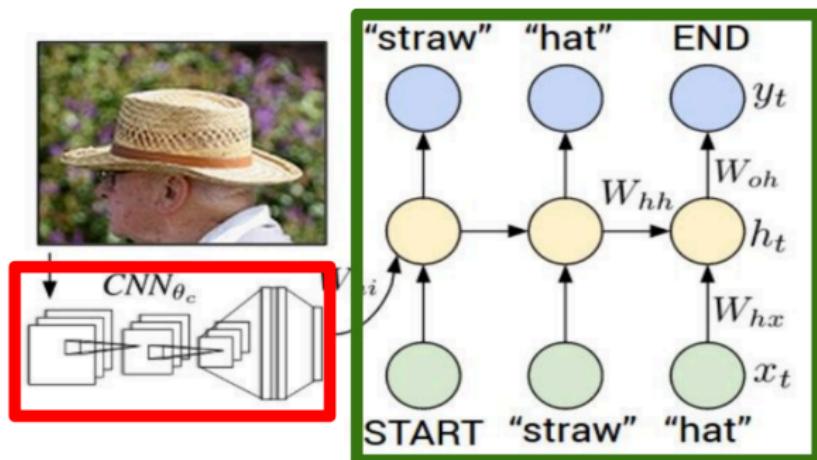
$$h_l^t = \tanh \left[W_l^h x_{l-1}^t + V_l^r (x_l^{t-1} \cdot r_l^t) \right] \quad \text{cell state} \quad (8)$$

$$x_l^t = (1 - z_l^t) \cdot h_l^t + z_l^t \cdot x_l^{t-1} \quad (9)$$

CAPTION GENERATION

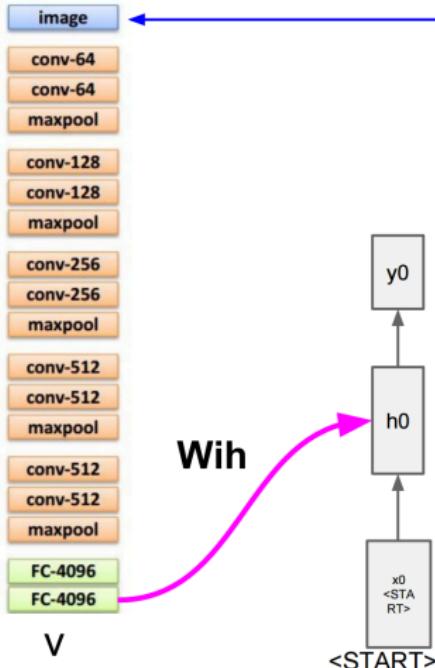
CAPTION GENERATION

Recurrent Neural Network



Convolutional Neural Network

CAPTION GENERATION



test image

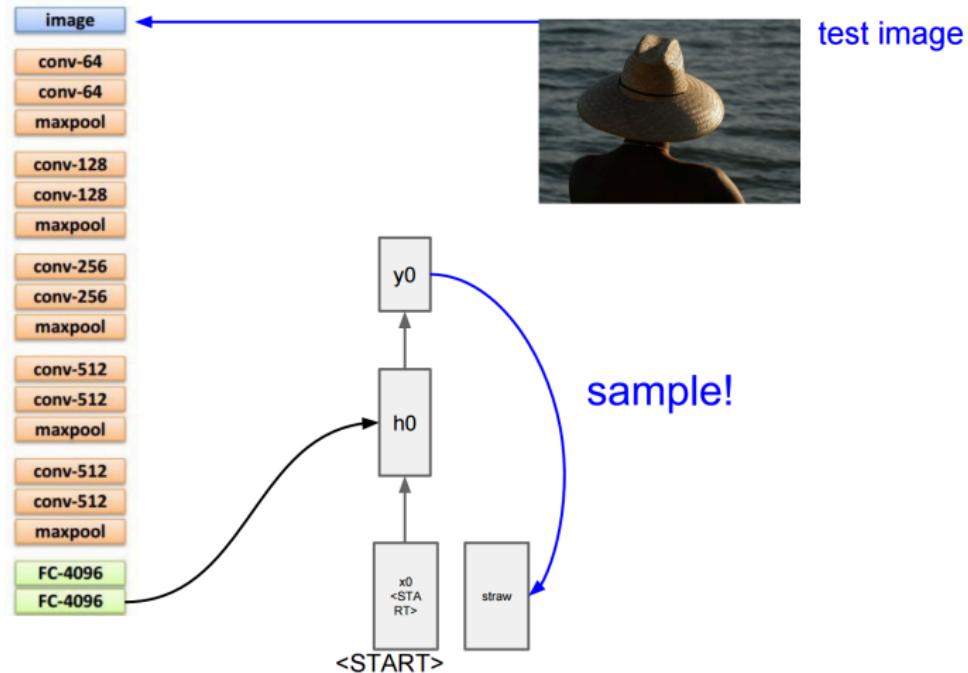
before:

$$h = \tanh(W_{xh} * x + W_{hh} * h)$$

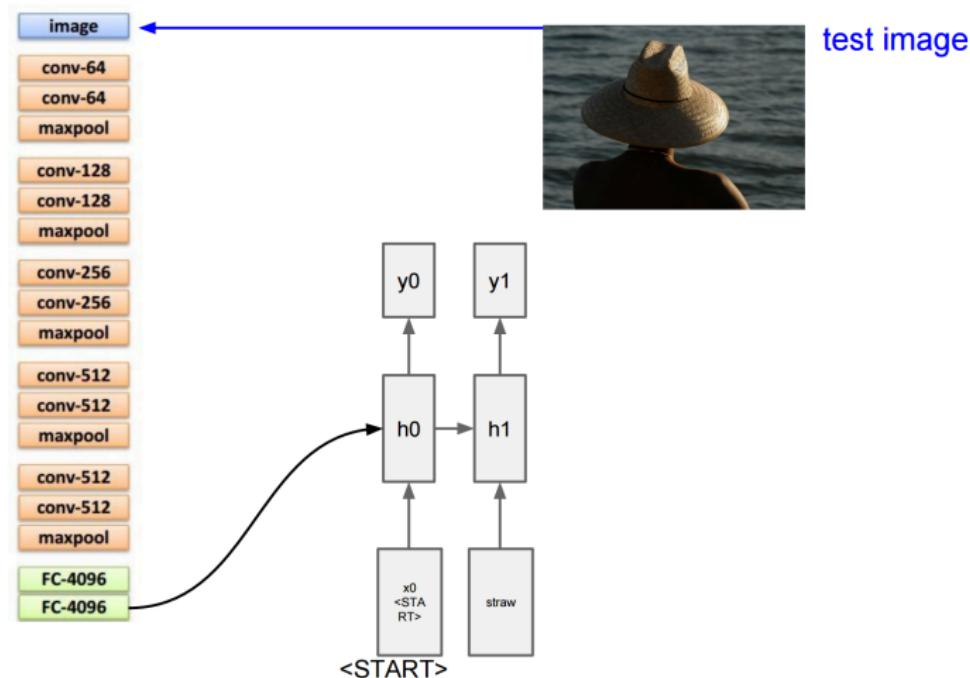
now:

$$h = \tanh(W_{xh} * x + W_{hh} * h + W_{ih} * v)$$

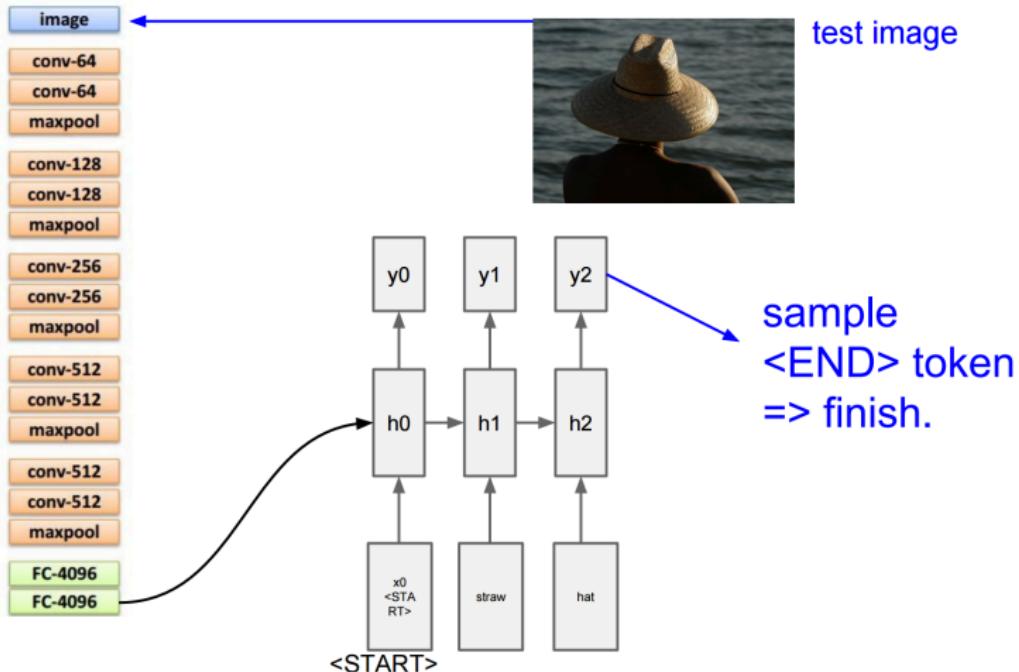
CAPTION GENERATION



CAPTION GENERATION



CAPTION GENERATION



CAPTION GENERATION



A cat sitting on a suitcase on the floor



A cat is sitting on a tree branch



A dog is running in the grass with a frisbee



A white teddy bear sitting in the grass



Two people walking on the beach with surfboards



A tennis player in action on the court



Two giraffes standing in a grassy field



A man riding a dirt bike on a dirt track

CAPTION GENERATION



A woman is holding a cat in her hand



A person holding a computer mouse on a desk



A woman standing on a beach holding a surfboard



A bird is perched on a tree branch



A man in a baseball uniform throwing a ball

CAPTION GENERATION



Q: What endangered animal is featured on the truck?

A: A bald eagle.

A: A sparrow.

A: A humming bird.

A: A raven.



Q: Where will the driver go if turning right?

A: Onto 24 1/4 Rd.

A: Onto 25 1/4 Rd.

A: Onto 23 1/4 Rd.

A: Onto Main Street.



Q: When was the picture taken?

A: During a wedding.

A: During a bar mitzvah.

A: During a funeral.

A: During a Sunday church service.



Q: Who is under the umbrella?

A: Two women.

A: A child.

A: An old man.

A: A husband and a wife.