MIPT Data Visualization Course

Deep Learning for Data with Sequence Structure

Ashuha Arseniy[1,2]

Bayesian Research Group[1], MIPT[2]

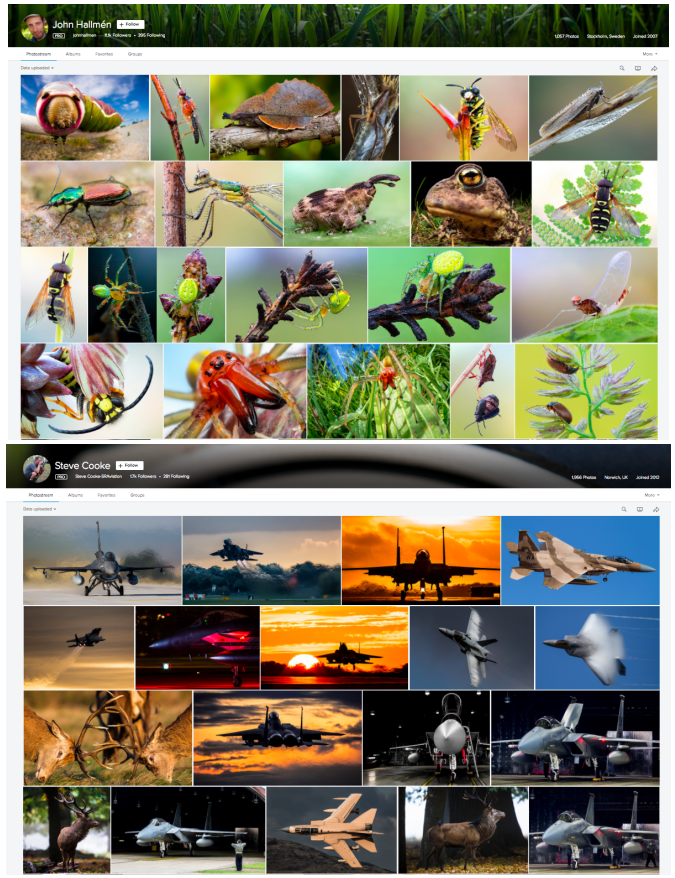
ars-ashuha.ru/slides

December 20, 2016

Independent and Identically-Distributed

- Image classification

- Flicker user's image classification



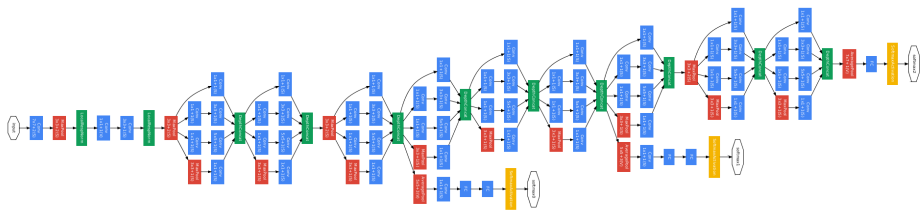
Flicker user's image classification

- Now we need to classify objects jointly, so we had sequence on input

- and predict sequence

```
['deer', 'plane', 'plane', 'deer', 'plane']
```

- What should we do? Do you have any ideas?
- Lets apply CNN for each image to get descriptor.



- We've transform sequence of pictures so sequence of vectors
- Ok data became much simply, what's next?

Recurrent Neural Net Concept

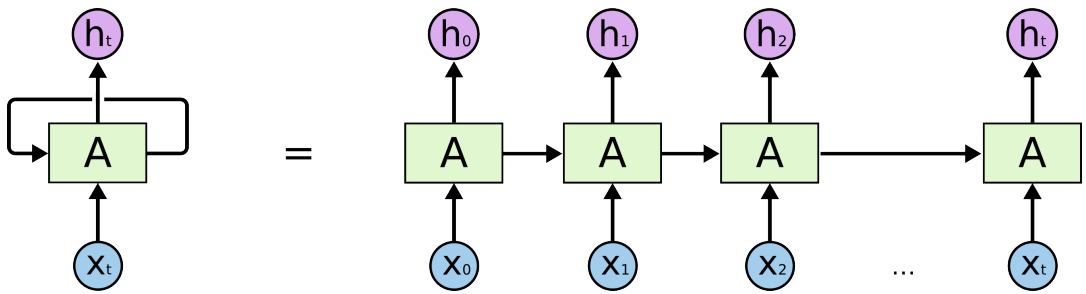
- Let's introduce memory cell

```
memory = np.ones(num_class)
for pic_vec in ['pic1_vec', 'pic2_vec', 'pic3_vec']:
    pic_class = g(W.dot(pic_vec) * f(memory))
    memory[pic_class] += 1
```

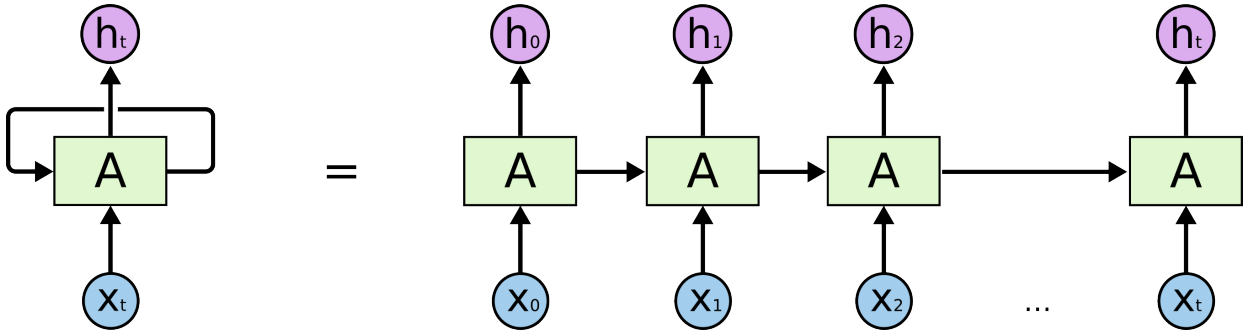
- Hmm, we will try control memory automatic

```
memory = np.ones(100)
for pic_vec in ['pic1_vec', 'pic2_vec', 'pic3_vec']:
    pic_class = g(Wout.dot([pic_vec, memory]))
    memory += Win.dot(pic_vec) + sigmoid(Wmem.dot(memory))
```

- Let's plot (memory update it's a little bit different but still it's same)



RNN and Training



- So we can compute inputs-outputs using this equations

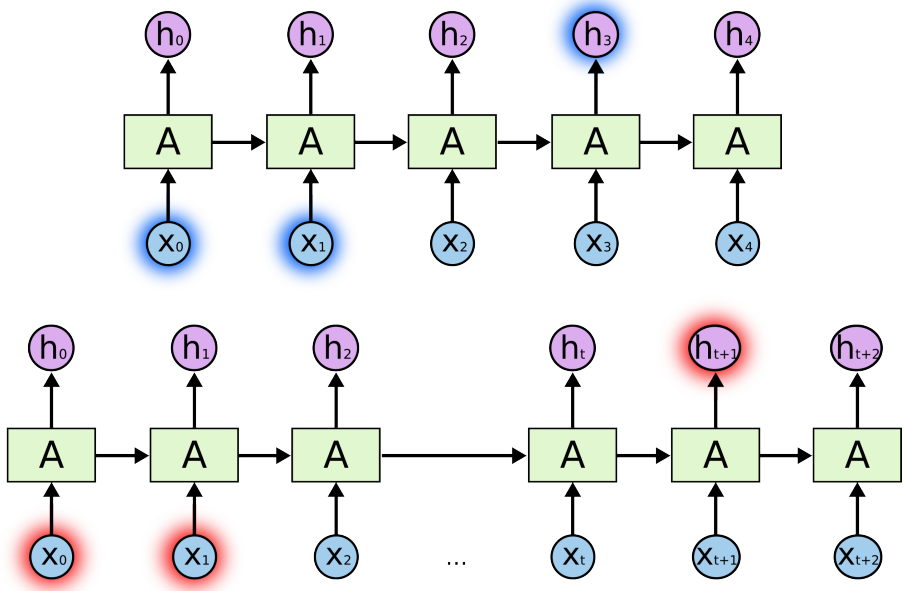
$$h_n = W_x x_n + W_h \sigma(h_{n-1}) \quad \hat{y}_n = \sigma(W_y h_n) \quad \theta = (W_x, W_h, W_y)$$

- You get several inputs and outputs, how would you train RNNs?
- Let's sum error for each output

$$L = \sum_t L_t(\hat{y}_t(\theta), y_t), \quad \frac{\partial L}{\partial \theta} = \sum_{1<t<T} \frac{\partial L_T(\hat{y}_t(\theta), y_t)}{\partial \theta}$$

- Next we should use ordinary optimization methods

RNN and Training



- Gradient is like this

$$\frac{\partial E_t}{\partial \theta} = \sum_{1 \le k \le t} \frac{\partial E_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial \theta} \qquad \frac{\partial h_t}{\partial h_k} = \prod_{t \ge i \ge k} \frac{\partial h_i}{\partial h_{i-1}}$$

- Multiply on matrix with small norm means no long deps
- Multiply on matrix with big norm means Gradient Boom

Razvan Pascanu, Tomas Mikolov, Yoshua Bengio: On the difficulty of training recurrent neural networks

Dealing with gradients Boom

- What is the simplest solution?
- Gradient clipping :)

Algorithm 1 Pseudo-code for norm clipping the gradients whenever they explode

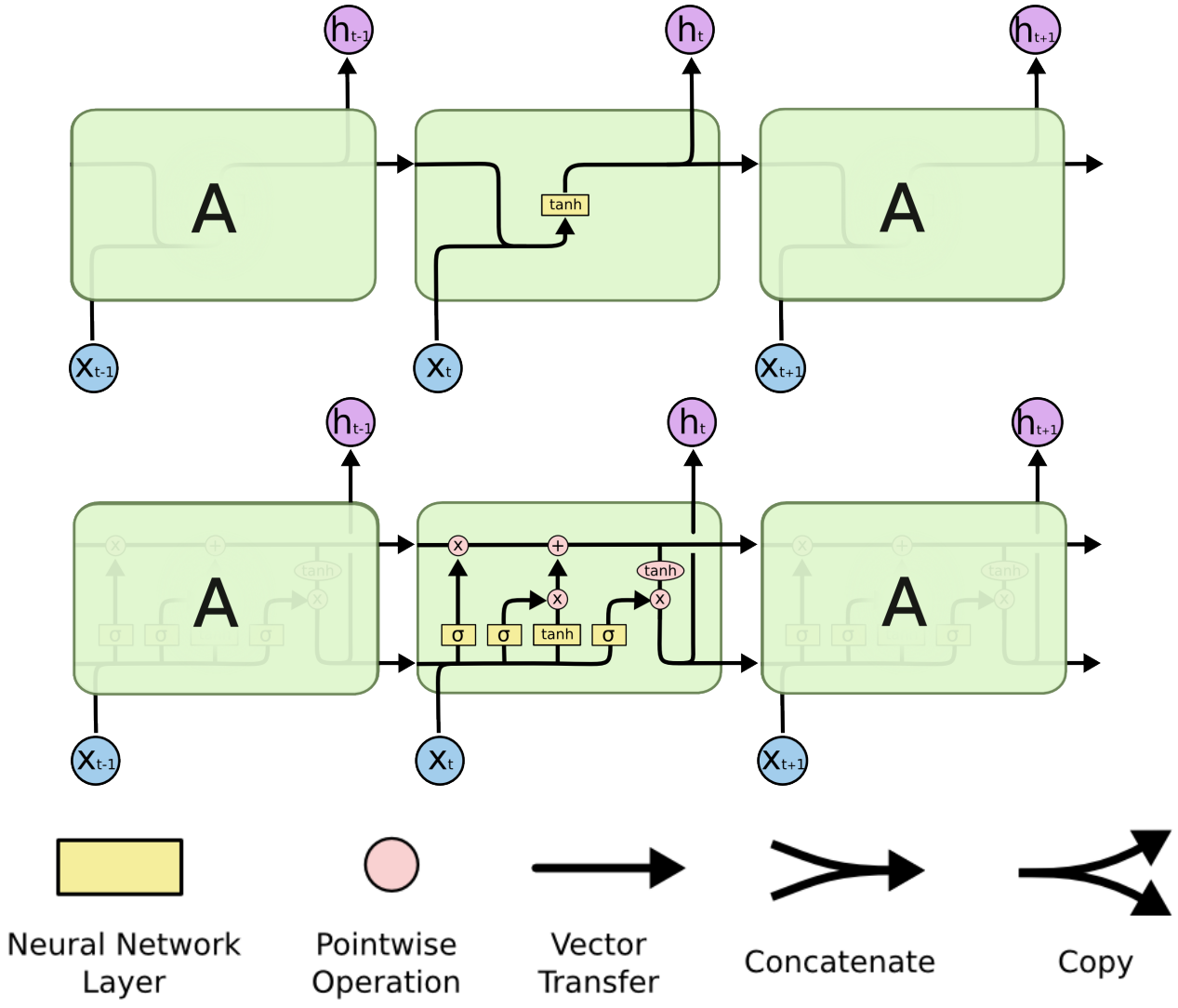
$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|}\hat{\mathbf{g}}$
end if

- It really works while gradient variance is low

Razvan Pascanu, Tomas Mikolov, Yoshua Bengio: On the difficulty of training recurrent neural networks

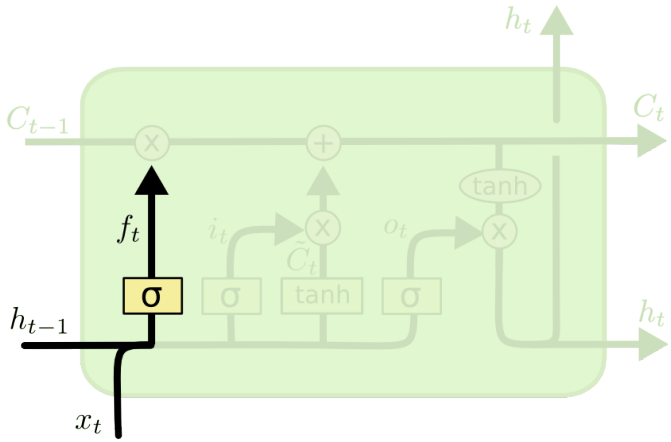
Dealing with long term dependences means LSTM

- We can't forget
- We can't understand importance of input information
- Let's add it



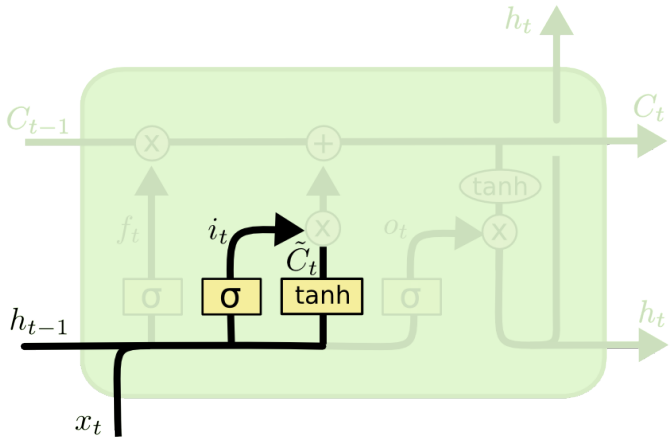
Dealing with long term dependences means LSTM

- Forget Gate



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

- Input Gate

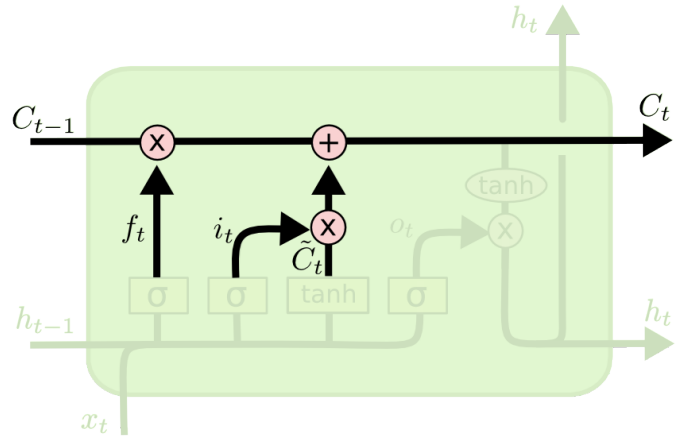


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

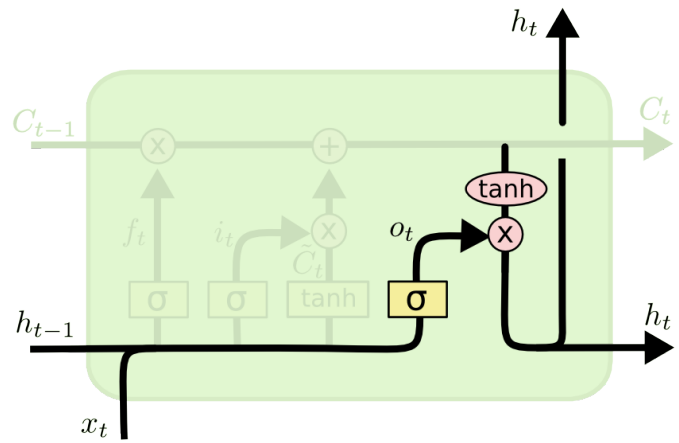
Dealing with long term dependences means LSTM

- Cell update



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

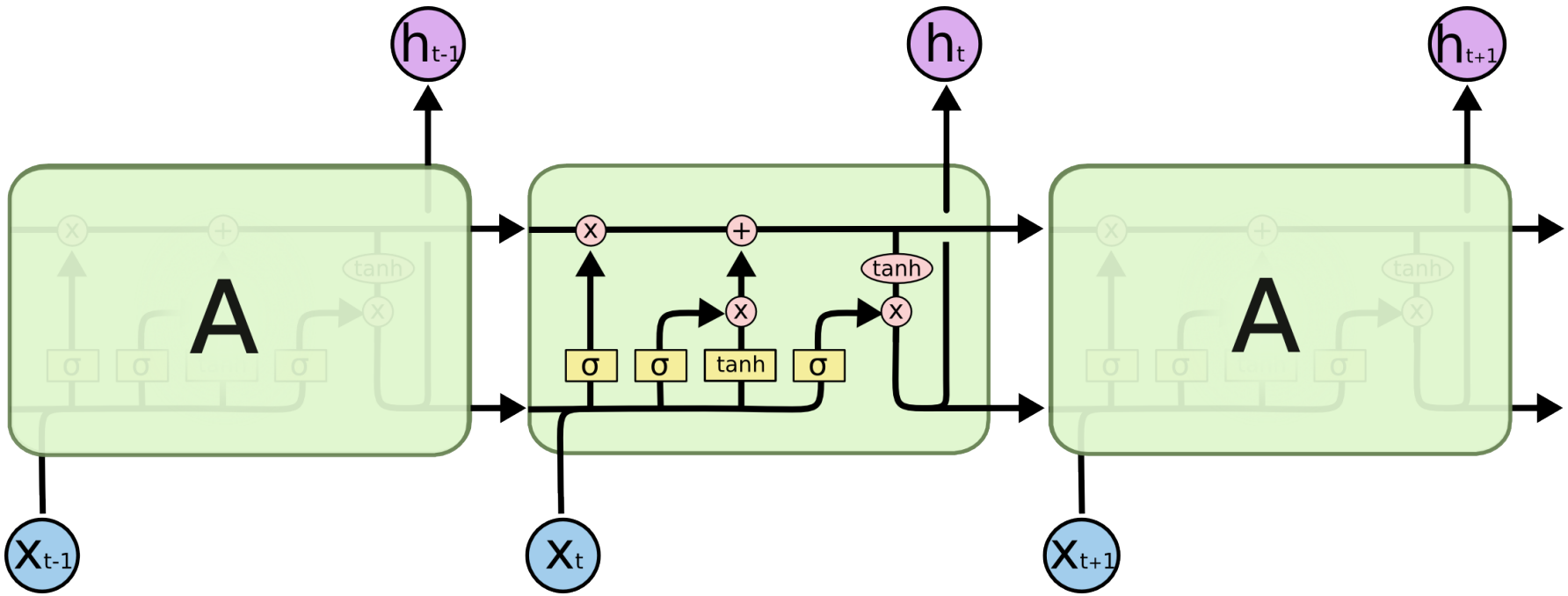
- Output Gate



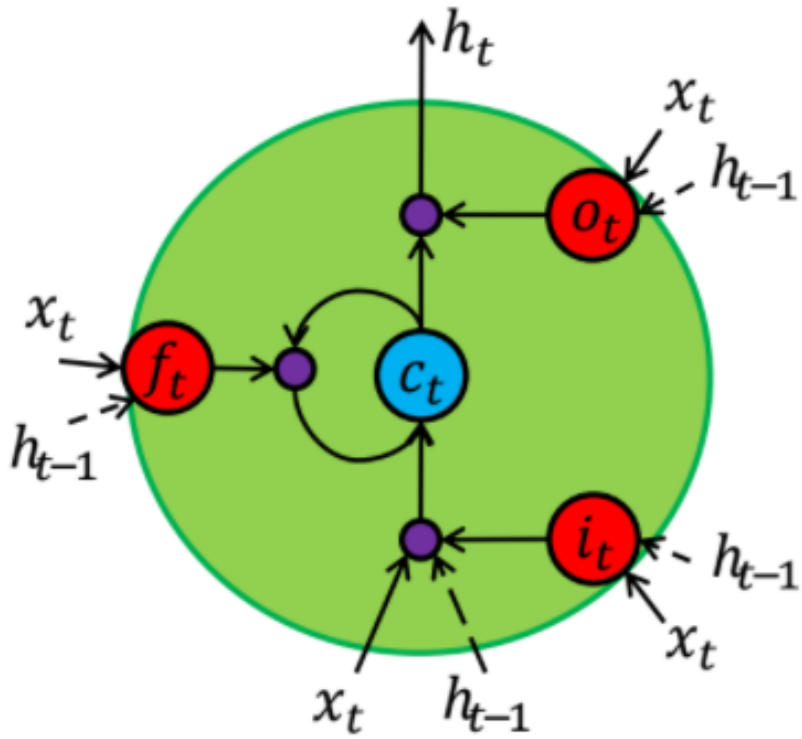
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

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

Dealing with long term dependences means LSTM



Gradient Vanishing in LSTM

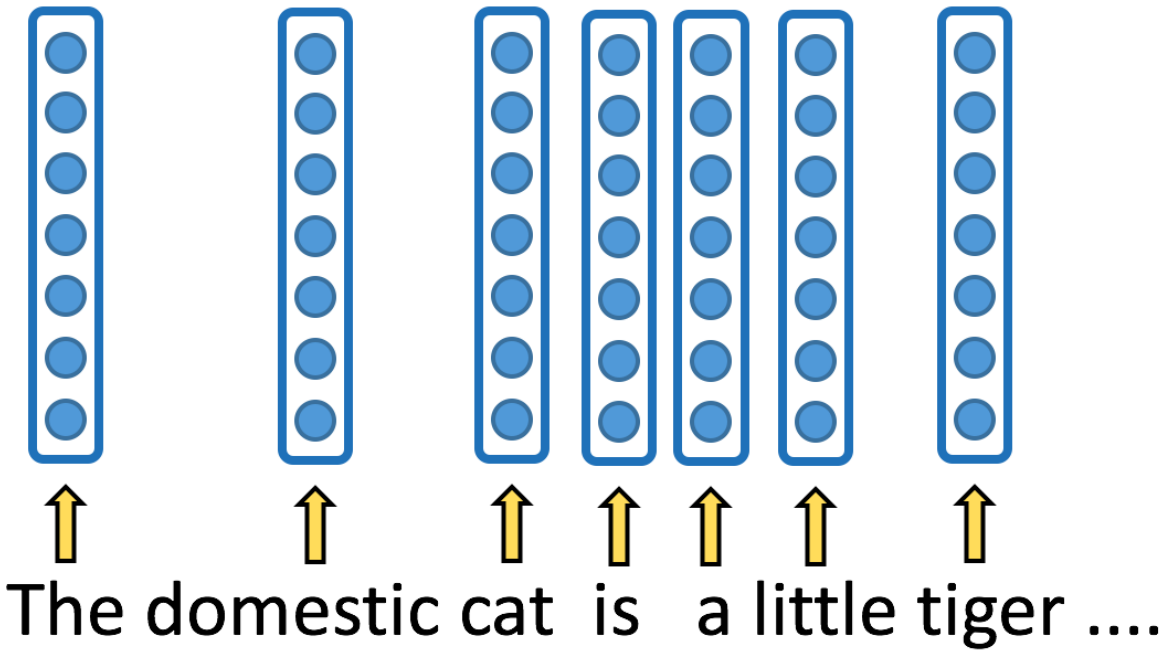


$$c_t = f_t \cdot c_{t-1} + t_t \cdot g(W_x x_t + W_c h_{t-1} + b_c), \qquad h_t = o_t \cdot (c_t)$$

$$\frac{\partial c_{t+1}}{\partial c_t} = f_{k+1}$$

Embedding Layer

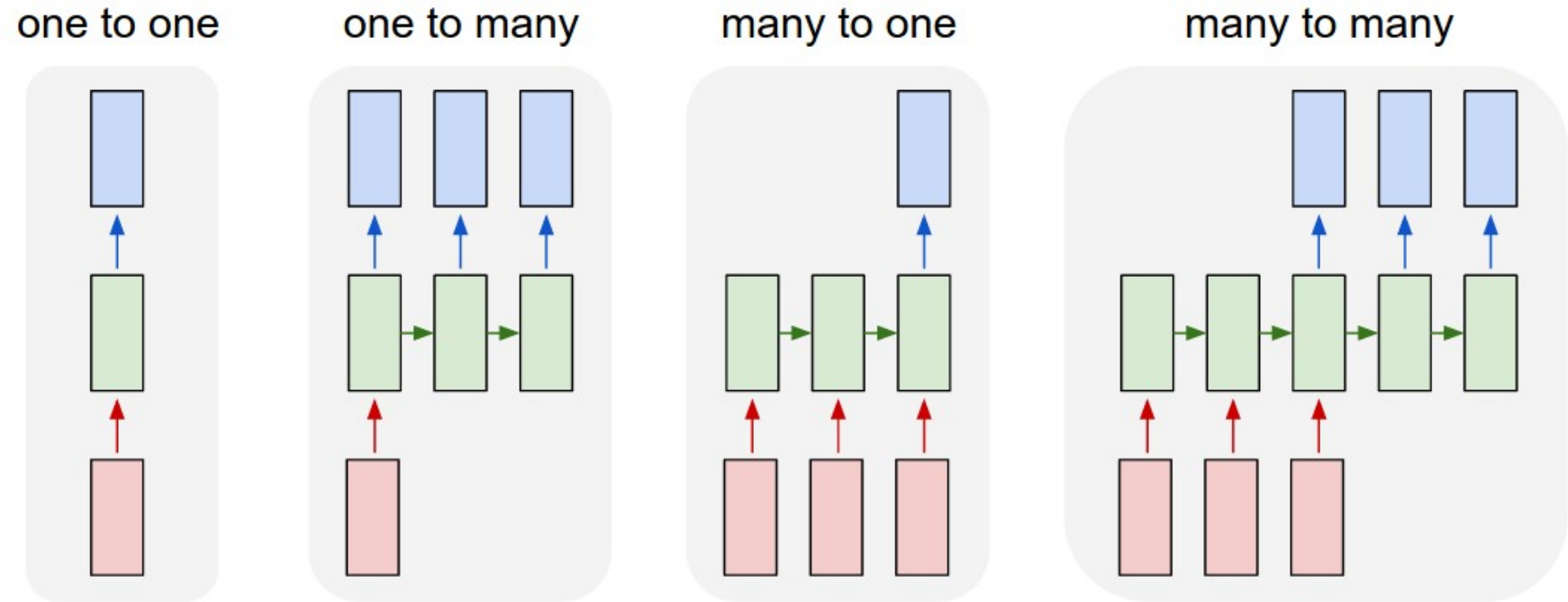
- What the main difference between words and pictures?
- Number of all words in much less then in picture
- So we can map identity vector for each word



- Let's initial with random noise
- So, we can evaluate gradient by word vectors
- word vectors are just a model parameters

Practical Cases

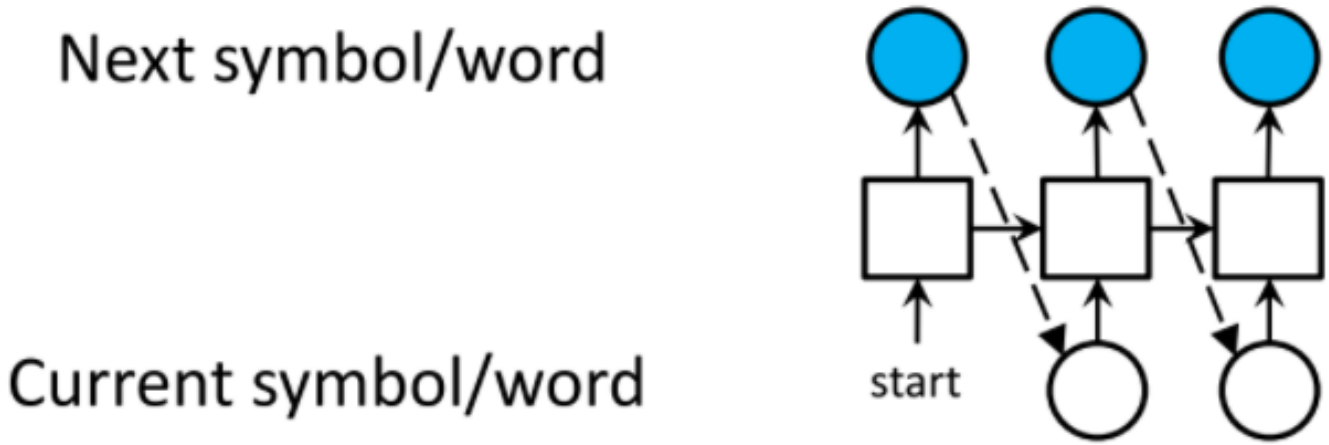
- Different tasks type



- Ho to code it

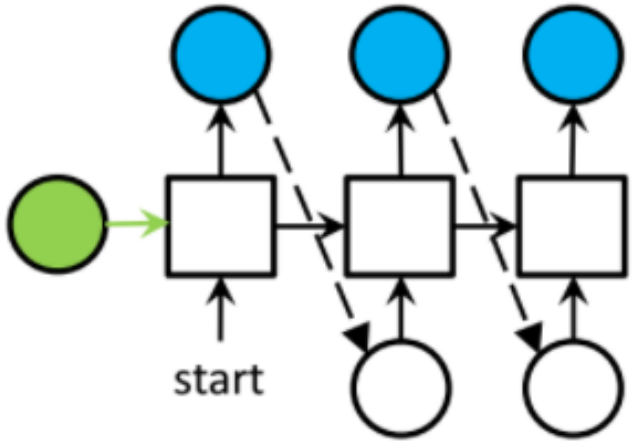
```
decoder = lasagne.layers.LSTMLayer(
    np.opes(batch, seq, vectors), num_units=100,
    cell_init=smth, grad_clipping=5)
```

Text/Handwritten Generation



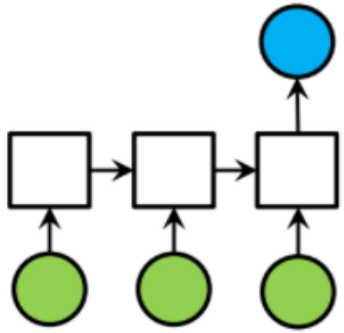
- http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- http://www.cs.toronto.edu/ graves/handwriting.html

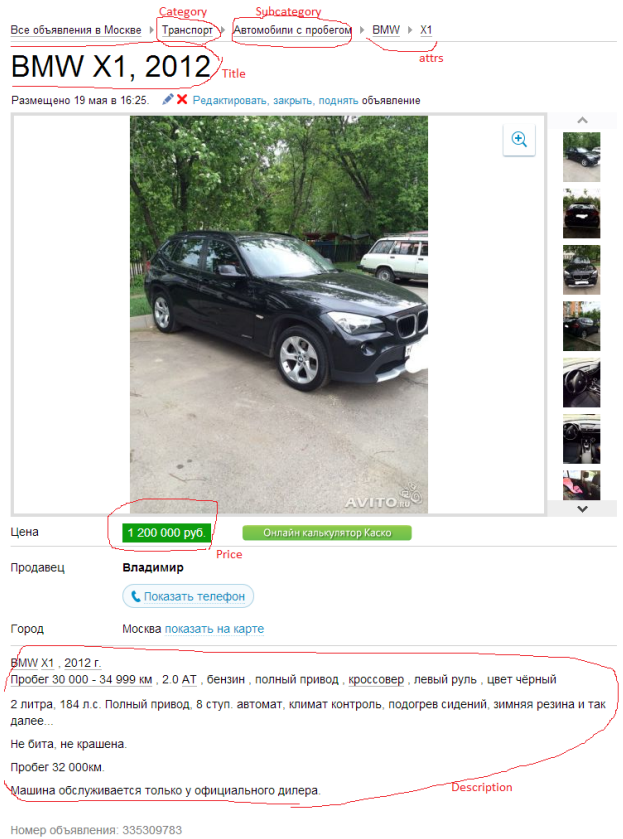
Sequence generation

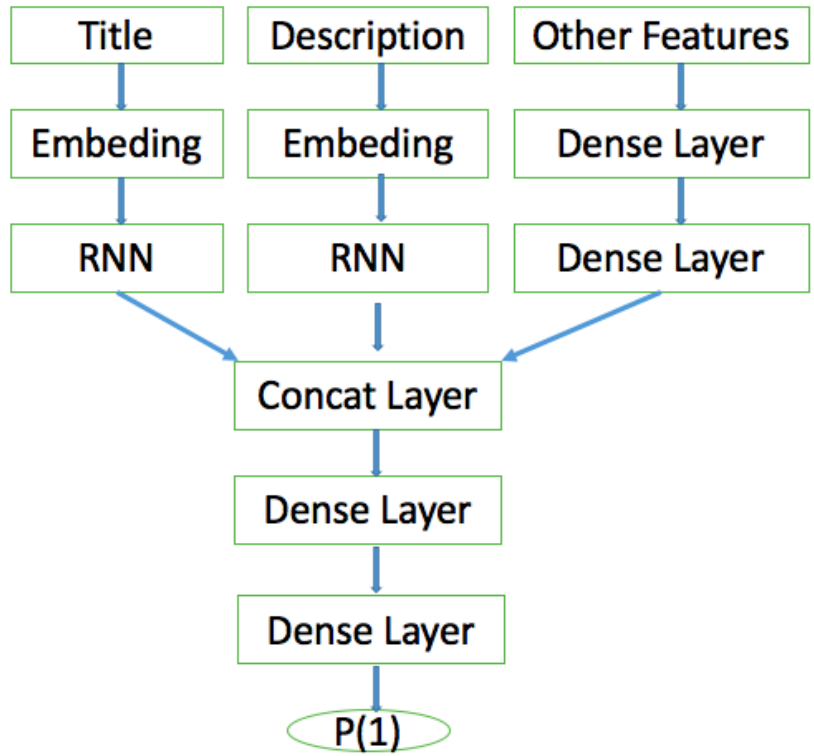


- http://www.cs.toronto.edu/ nitish/nips2014demo/index.html
- cs.stanford.edu/people/karpathy/deepimagesent/generationdemo/

Sequence classification

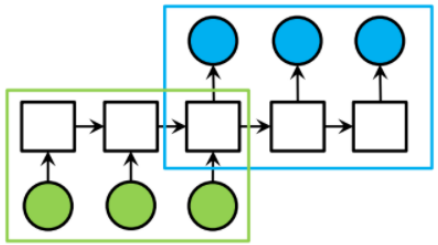




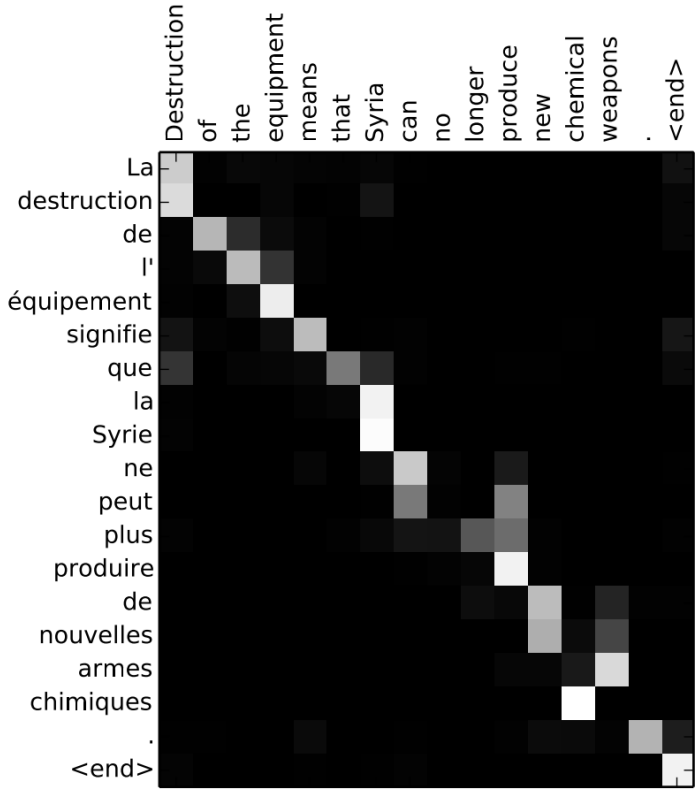


Sequence2Sequence Machine Translation

- Coding input sentence and generate output



- Add attention mechanize



Recap

- Recurrent Deep Neural nets are great
- RNN has achieved many state of the art awards
- Input and output may have any length
- It can be combine with convolution DNN
- It's Easy to code it by yourself with any DL framework
- Link's
  - http://colah.github.io/posts/2015-08-Understanding-LSTMs/
  - http://www.machinelearning.ru/wiki/index.php?title=DI
  - www.deeplearningbook.org