ATTENTIVE CONVOLUTIONAL LSTM

AImage Lab

UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

RECALL….

```
ssh -p port user@YourAzureVM
pip freeze | grep torch==0.4.0
git pull https://github.com/aimagelab/aidlda_tutoral
```

You have all the slides and the code in the Github repo!

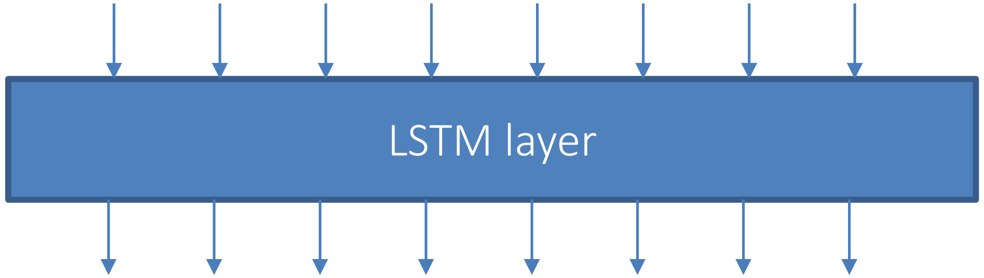
Don't get lost:

- if you don't understand something: **ask**!
- If you can't do something: **the solution is in the repo**!

LSTM

A recurrent layer.

$$\mathbf{X} = \{X_1, X_2, ..., X_T\}$$



$$\mathbf{Y} = \{Y_1, Y_2, ..., Y_T\}$$

From the exterior

Takes as input a sequence.

Outputs a sequence with the same length.

LSTM

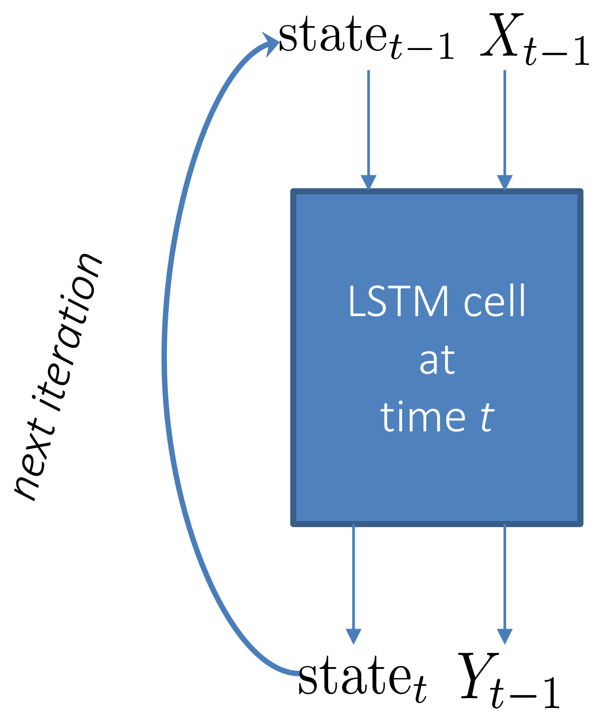
A recurrent layer.

Inside, at each timestep:

Takes the current input

Combines it with current state

Produces output and next states



LSTM

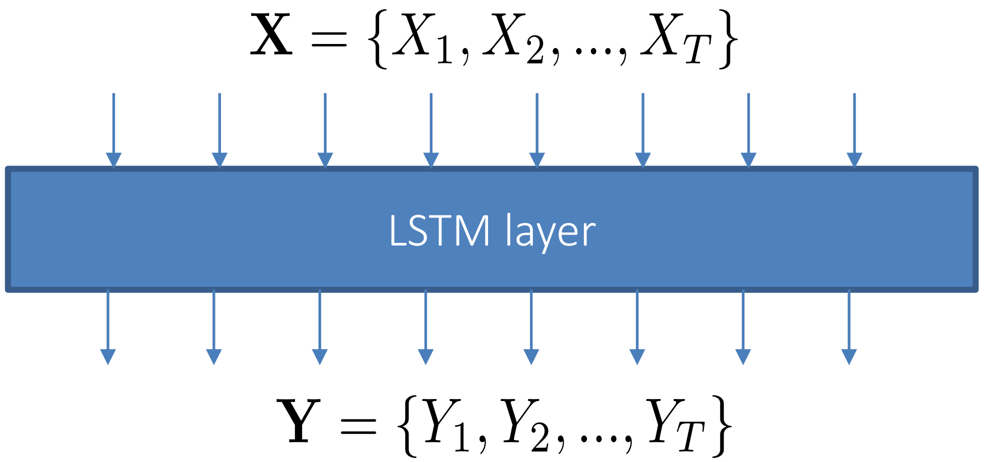
A recurrent layer.

Inside, at each timestep:

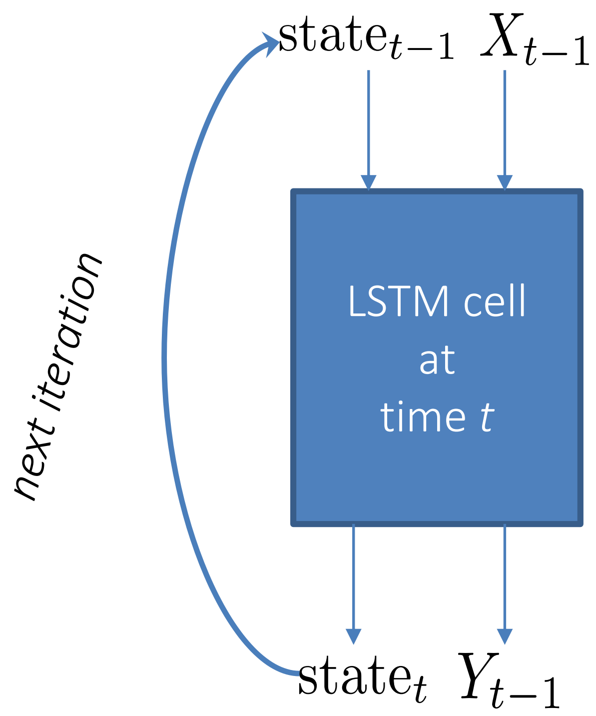
Takes the current input

Combines it with current state

Produces output and next states



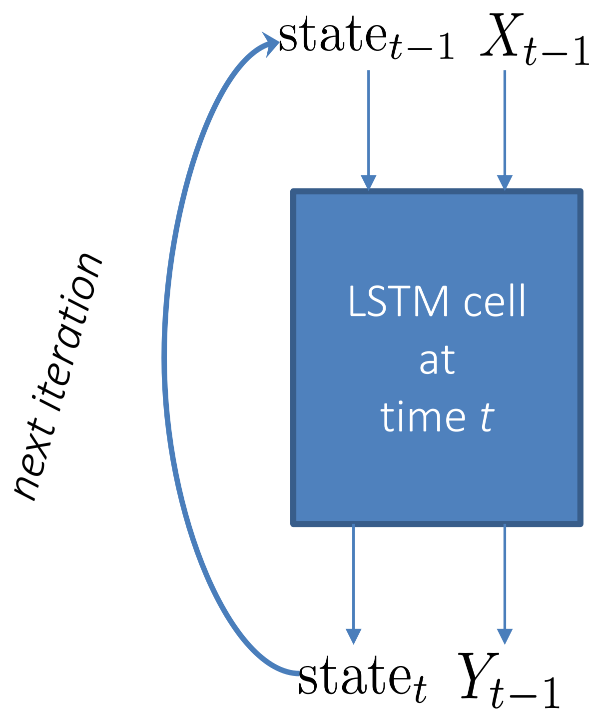
Repeat *T* times



LSTM

For the LSTM, the actual equations are:

$$I_t = \sigma(W_i * \tilde{X}_t + U_i * H_{t-1} + b_i)$$
$$F_t = \sigma(W_f * \tilde{X}_t + U_f * H_{t-1} + b_f)$$
$$O_t = \sigma(W_o * \tilde{X}_t + U_o * H_{t-1} + b_o)$$
$$G_t = \tanh(W_c * \tilde{X}_t + U_c * H_{t-1} + b_c)$$
$$C_t = F_t \odot C_{t-1} + I_t \odot G_t$$
$$H_t = O_t \odot \tanh(C_t)$$



The “state” is given by (Ht, Ct). The “output” is Ht

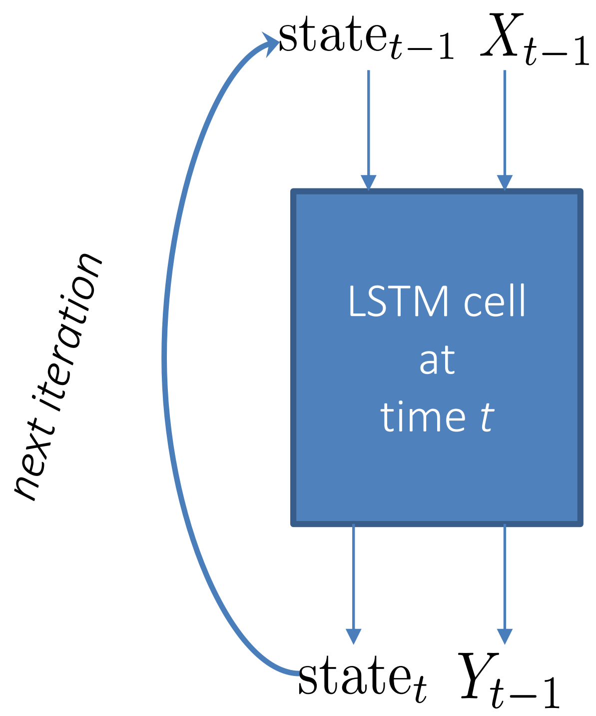
Notice that, beside the activation function, all gates are computed in the same way.

→ Any ideas on how to exploit this to speed up the computation?

LSTM

For the LSTM, the actual equations are:

$$I_t = \sigma(W_i * \tilde{X}_t + U_i * H_{t-1} + b_i)$$
$$F_t = \sigma(W_f * \tilde{X}_t + U_f * H_{t-1} + b_f)$$
$$O_t = \sigma(W_o * \tilde{X}_t + U_o * H_{t-1} + b_o)$$
$$G_t = \tanh(W_c * \tilde{X}_t + U_c * H_{t-1} + b_c)$$
$$C_t = F_t \odot C_{t-1} + I_t \odot G_t$$
$$H_t = O_t \odot \tanh(C_t)$$



The “state” is given by (Ht, Ct). The “output” is Ht

We are going to implement an LSTM on images. So inputs, states and outputs will be 3-d tensors (channels x h x w), and operations between them will be convolutions!

ATTENTION

The input of the network is replaced by an element-wise multiplication between the input and an **attentive map:**

$$Z_t = V_a * \tanh(W_a * X + U_a * H_{t-1} + b_a). \qquad (7)$$

The output of this operations is a 2-d map from which we can compute a normalized spatial attention map through the *softmax* operator:

$$A_t^{ij} = p(att_{ij}|X, H_{t-1}) = \frac{\exp(Z_t^{ij})}{\sum_i \sum_j \exp(Z_t^{ij})} \qquad (8)$$

where A_t^{ij} is the element of the attention map in position (i, j). The attention map is applied to the input X with an element-wise product between each channel of the feature maps and the attention map:

$$\tilde{X}_t = A_t \odot X. \qquad (9)$$

CODING TIME!

Implement a Convolutional Attentive LSTM, following the equations of the previous slides.

Follow the schema in the Github repo:

1. Implement an LSTM Cell
2. Implement an LSTM Module, which calls (1) at each iteration
3. Plug the attention equations in
4. Test it!

CONCLUSIONS

AImage Lab

UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

CONCLUSIONS

AImage Lab

What you should have learned

- Some theory on human fixation and saliency prediction
- Two state of the art approaches, for task agnostic and task-driven saliency
- You should be a quasi-guru of PyTorch programming 1-0-1 ☺
- How to implement forward/backward passes of the essential building blocks of a CNN
- How an attentive LSTM works, and how it can be applied for saliency prediction

CONCLUSIONS

AImage Lab

If you still have questions…

- Marcella will be around in the next days
- Or, drop us a line! ☺

  - lorenzo.baraldi@unimore.it
  - davide.abati@unimore.it
  - marcella.cornia@unimore.it

THANK YOU!

AImage Lab

UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA