Overview of Deep Learning Instruments pt. 1

Sergey Ivanov (517)

qbrick@mail.ru

September 17, 2018

Table of contents

Section 1

Deep Learning

What is neural net?

What is neural net?

- parametric family $f(x, \theta), \quad \theta \in \Theta$

What is neural net?

- parametric family $f(x, \theta), \quad \theta \in \Theta$
- with universal approximation properties

What is neural net?

- parametric family $f(x, \theta), \quad \theta \in \Theta$
- with universal approximation properties
- differentiable

What is neural net?

- parametric family $f(x, \theta), \quad \theta \in \Theta$
- with universal approximation properties
- differentiable

Deep Learning is Machine Learning!

Machine Learning is always about searching for function:

$$\mathbb{E}_{(x,y) \sim \text{Data}} \text{Loss}(f(x, \theta), y) \to \min_{\theta}$$

Building neural nets

Common way to build complex functions — composition:

$$f(x, \theta) = f_1(f_2(f_3(\dots)))$$

Chain rule gives us the derivative $\nabla f(x, \theta)$

Building neural nets

Common way to build complex functions — composition:

$$f(x, \theta) = f_1(f_2(f_3(\dots)))$$

Chain rule gives us the derivative $\nabla f(x, \theta)$

Same works for functions of vectors!

Building neural nets

Common way to build complex functions — composition:

$$f(x, \theta) = f_1(f_2(f_3(\dots)))$$

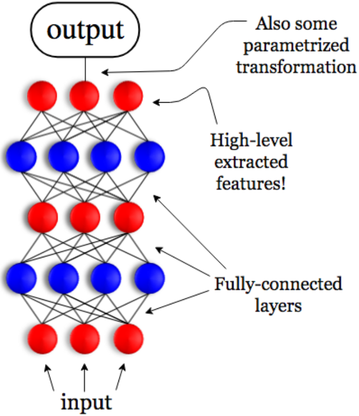
Chain rule gives us the derivative $\nabla f(x, \theta)$

Same works for functions of vectors!
Typical example:

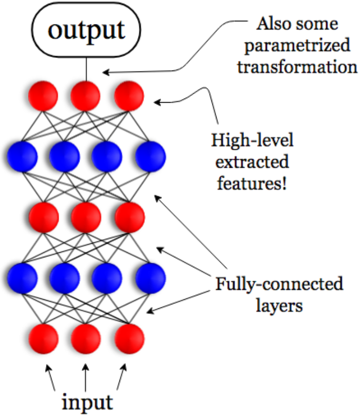
$$f_i(x, \theta) \in \{Ax, \sigma(x), \dots\}$$

where σ — some element-wise nonlinear function.

Typical example



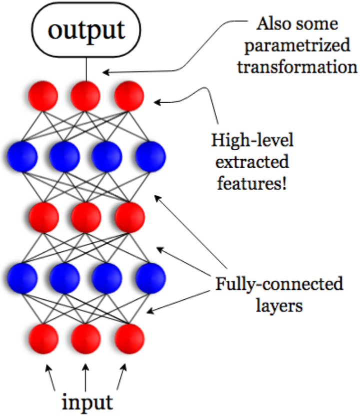
Typical example



Output:

- regression:
  - just numbers

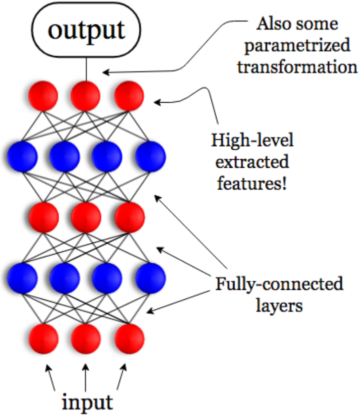
Typical example



Output:

- regression:
    - just numbers
    - parameters of distribution

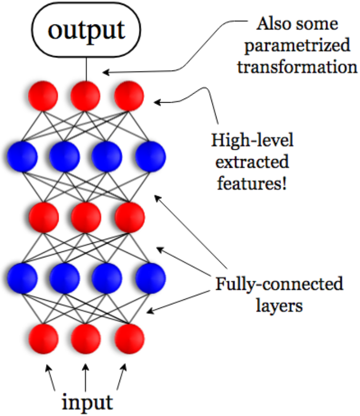
Typical example



Output:

- regression:
  - just numbers
  - parameters of distribution
- classification:
  - × just classes

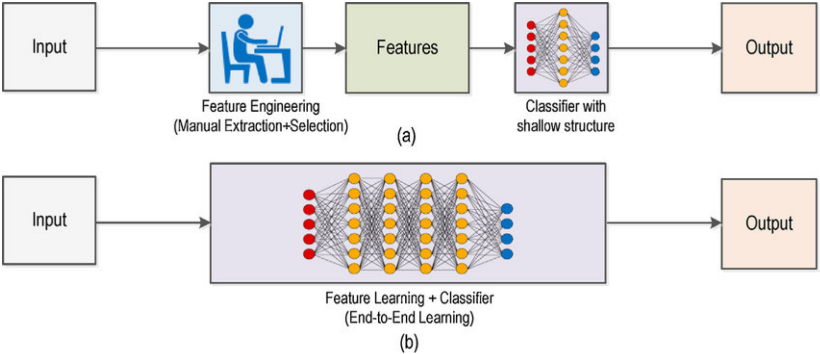
Typical example



Output:

- regression:
  - just numbers
  - parameters of distribution
- classification:
  - × just classes
  - probabilities of classes

End-to-end learning



Automation is the goal!

In DL we are required to specify:

- net topology

Automation is the goal!

In DL we are required to specify:

- net topology
  - trial and error
  - evolutionary methods

Automation is the goal!

In DL we are required to specify:

- net topology
  - trial and error
  - evolutionary methods
  - ✓ AutoML

Automation is the goal!

In DL we are required to specify:

- net topology
  - trial and error
  - evolutionary methods
  - ✓ AutoML
- regularization

Automation is the goal!

In DL we are required to specify:

- net topology
  - trial and error
  - evolutionary methods
  - ✓ AutoML
- regularization
  - dropout
  - batch normalization

Automation is the goal!

In DL we are required to specify:

- net topology
  - trial and error
  - evolutionary methods
  - ✓ AutoML
- regularization
  - dropout
  - batch normalization
  - ✓ Bayesian neural nets

Automation is the goal!

In DL we are required to specify:

- net topology
  - trial and error
  - evolutionary methods
  - ✓ AutoML
- regularization
  - dropout
  - batch normalization
  - ✓ Bayesian neural nets
- optimization method

Automation is the goal!

In DL we are required to specify:

- net topology
  - trial and error
  - evolutionary methods
  - ✓ AutoML
- regularization
  - dropout
  - batch normalization
  - ✓ Bayesian neural nets
- optimization method
  - use more or less universal methods like Adam

Automation is the goal!

In DL we are required to specify:

- net topology
  - trial and error
  - evolutionary methods
  - ✓ AutoML
- regularization
  - dropout
  - batch normalization
  - ✓ Bayesian neural nets
- optimization method
  - use more or less universal methods like Adam
  - ✓ Meta-learning

Automation is the goal!

In DL we are required to specify:

- net topology
  - trial and error
  - evolutionary methods
  - ✓ AutoML
- regularization
  - dropout
  - batch normalization
  - ✓ Bayesian neural nets
- optimization method
  - use more or less universal methods like Adam
  - ✓ Meta-learning
- data representation

Automation is the goal!

In DL we are required to specify:

- net topology
  - trial and error
  - evolutionary methods
  - ✓ AutoML
- regularization
  - dropout
  - batch normalization
  - ✓ Bayesian neural nets
- optimization method
  - use more or less universal methods like Adam
  - ✓ Meta-learning
- data representation
  - "stack more layers"
  - "we need to go deeper"

Automation is the goal!

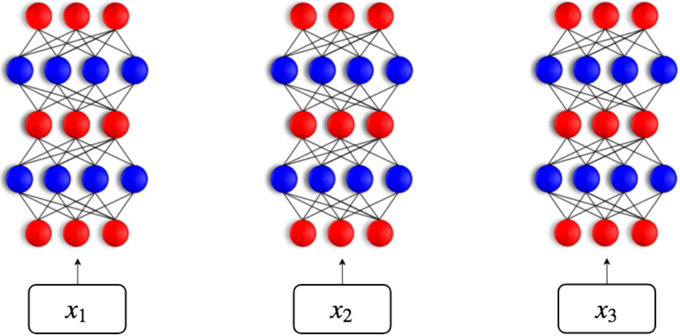
In DL we are required to specify:

- net topology
  - trial and error
  - evolutionary methods
  - ✓ AutoML
- regularization
  - dropout
  - batch normalization
  - ✓ Bayesian neural nets
- optimization method
  - use more or less universal methods like Adam
  - ✓ Meta-learning
- data representation
  - "stack more layers"
  - "we need to go deeper"
  - ✓ ?!?

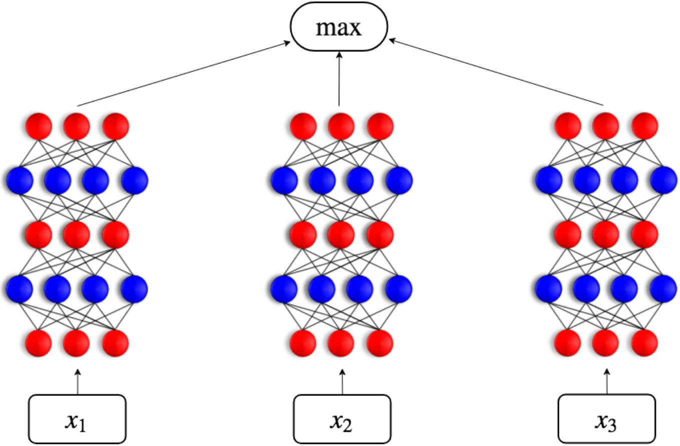
Section 2

Considering data structure

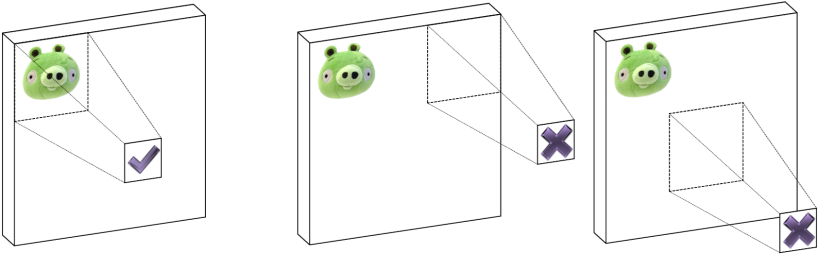
Pooling invariants



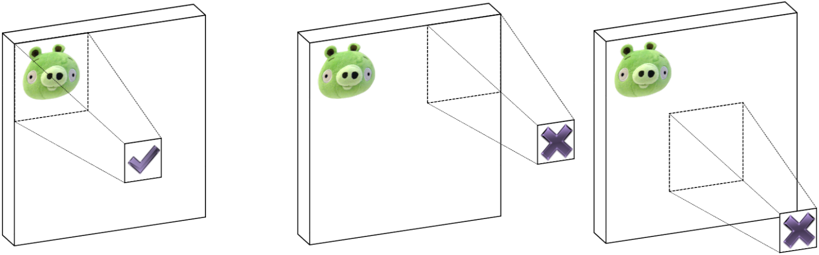
Pooling invariants



Translation invariance



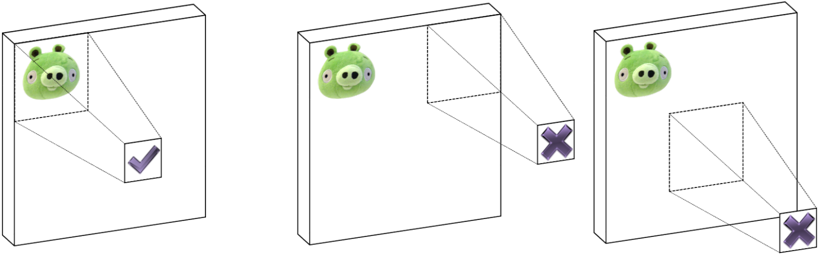
Translation invariance



Usually followed by:

- max pooling (one invariant is of a particular interest)
  - other pooling options possible

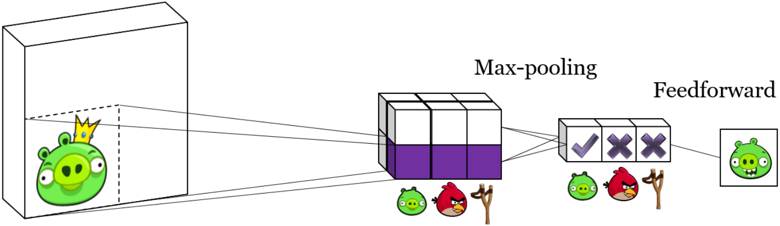
Translation invariance



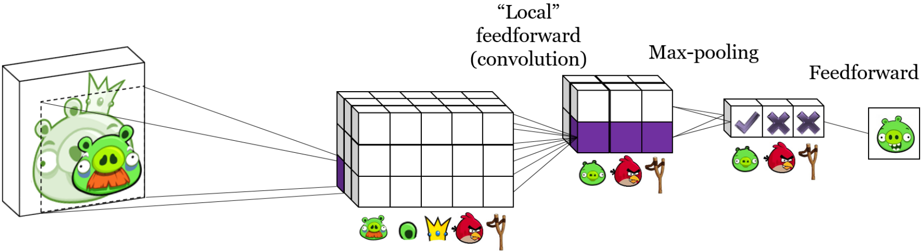
Usually followed by:

- max pooling (one invariant is of a particular interest)
  - other pooling options possible
- concatenation (for subtasks of same structure)

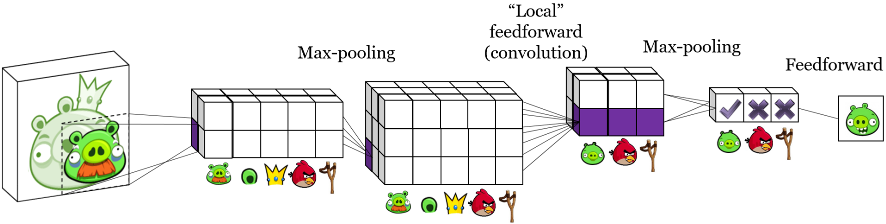
Size invariance



Size invariance

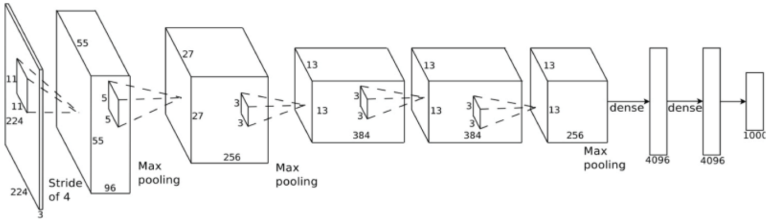


Size invariance

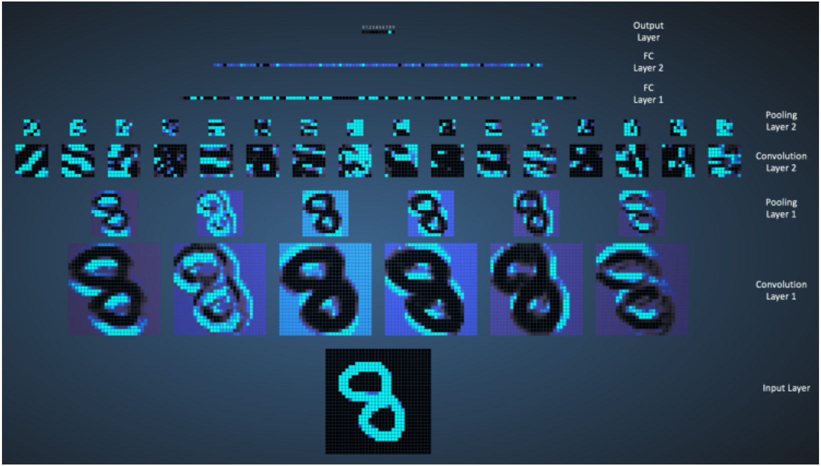


Convolutional neural network (CNN)

Resulting network:

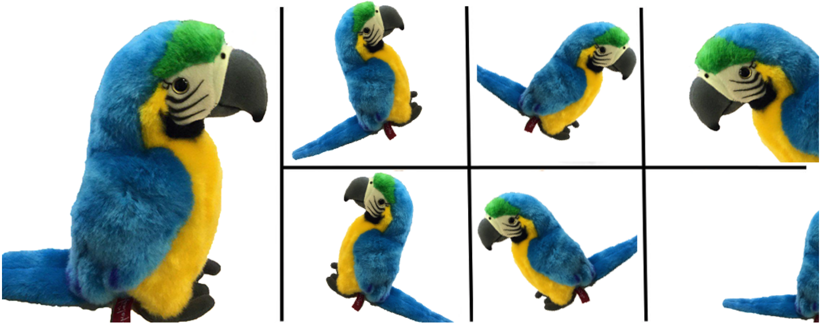


Convolutional neural network (CNN)



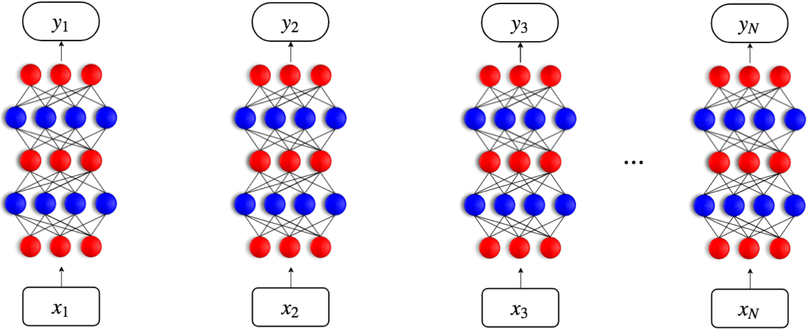
Augmentation

If you can't consider invariants in architecture, enlarge your dataset.



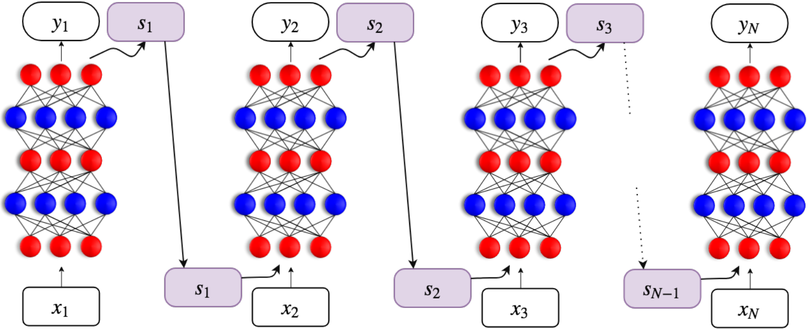
Sequences as input

Applying same idea:

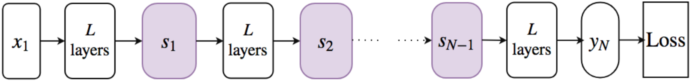


Sequences as input

Naive approach:



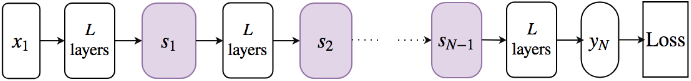
Gradients problem



Problem:

Gradient is required to pass LN layers.

Gradients problem

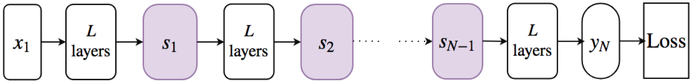


Problem:

Gradient is required to pass LN layers.

Chain rule says it's multiplication of LN quantities.

Gradients problem



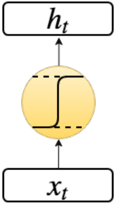
Problem:

Gradient is required to pass LN layers.

Chain rule says it's multiplication of LN quantities.

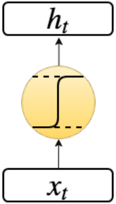
- most absolute values < 1: vanishing gradients problem
- most absolute values > 1: exploding gradients problem

Recurrent units

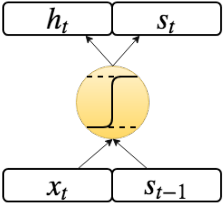


Neuron
(e.g. $\sigma(Ax_t)$)

Recurrent units

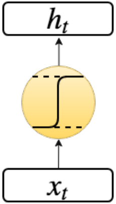


Neuron
(e.g. $\sigma(Ax_t)$)

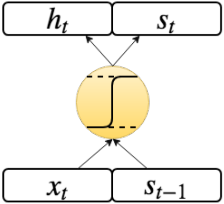


Same idea applied
(redundant)

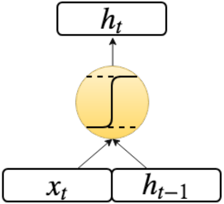
Recurrent units



Neuron
(e.g. $\sigma(Ax_t)$)

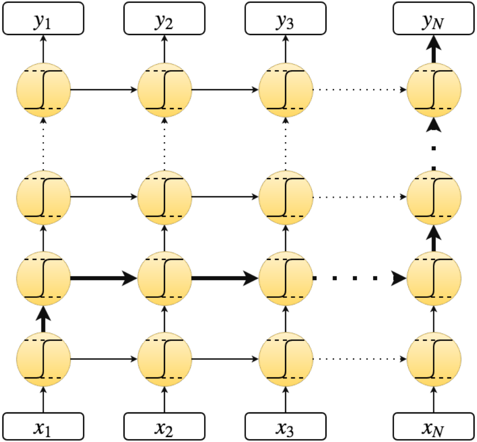


Same idea applied
(redundant)

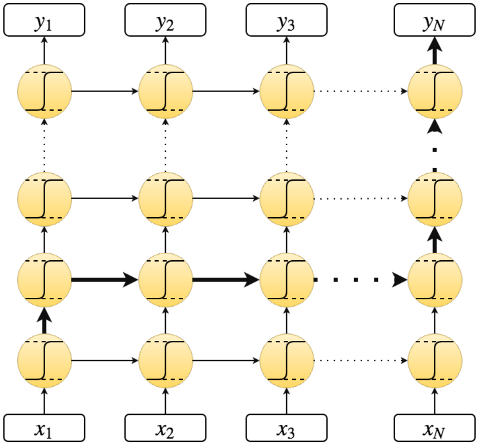


Recurrent neuron
(e.g. $\sigma(A\left[x_t, h_{t-1}\right])$)

Recurrent neural nets

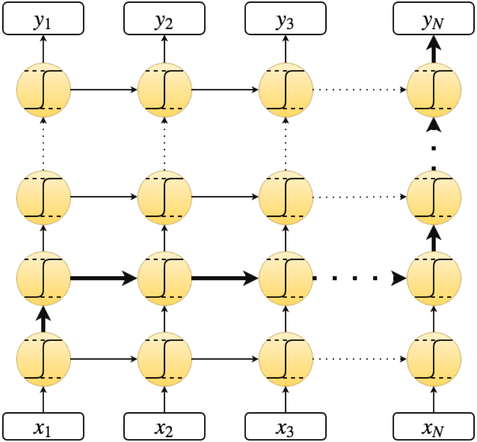


Recurrent neural nets



- ✓ $N + L$ layers for gradient to pass!

Recurrent neural nets



- ✓ $N + L$ layers for gradient to pass!
- ? Was previous option better at something?

Memory

Consider *writing to memory* task, i. e. the following operation:

```
if need_to_write(x):
    c = f(x)
```

How to express it in terms of computational graphs?

Memory

Consider *writing to memory* task, i. e. the following operation:

```
if need_to_write(x):
    c = f(x)
```

How to express it in terms of computational graphs?

```
c *= 1 - need_to_write(x)
c += need_to_write(x) * f(x)
```

Memory

Consider *writing to memory* task, i. e. the following operation:

```
if need_to_write(x):
    c = f(x)
```

How to express it in terms of computational graphs?

```
c *= 1 - need_to_forget(x)
c += need_to_write(x) * f(x)
```

Memory

Consider *writing to memory* task, i. e. the following operation:

```
if need_to_write(x):
    c = f(x)
```

How to express it in terms of computational graphs?

```
c *= 1 - need_to_forget(x)
c += need_to_write(x) * f(x)
```

Memory update formula

$$c_t = f_t \circ c_{t-1} + w_t \circ f(x_t) \quad w_t, f_t \in \{0, 1\}$$

where $\circ$ is element-wise multiplication.

Gates

w_t, f_t are also some functions of input! For example,

$$\mathbb{I}[Ax_t > 0]$$

Gates

w_t, f_t are also some functions of input! For example,

$$\mathbb{I}[Ax_t > 0]$$

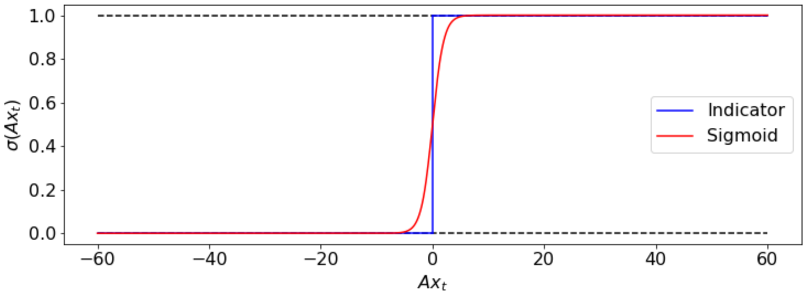
DL main rule: if something is not differentiable, make a smooth (*soft*) version of it!

Gates

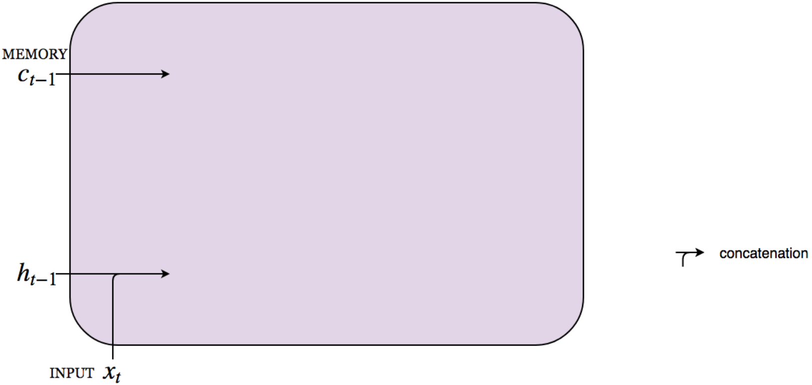
w_t, f_t are also some functions of input! For example,

$$\mathbb{I}[Ax_t > 0]$$

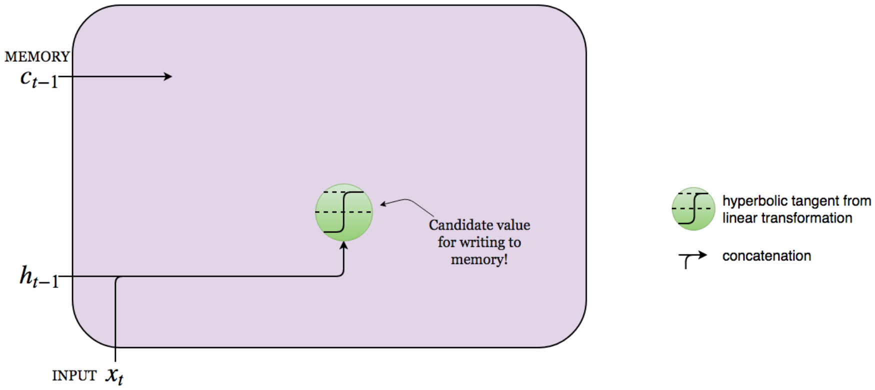
DL main rule: if something is not differentiable, make a smooth (*soft*) version of it!



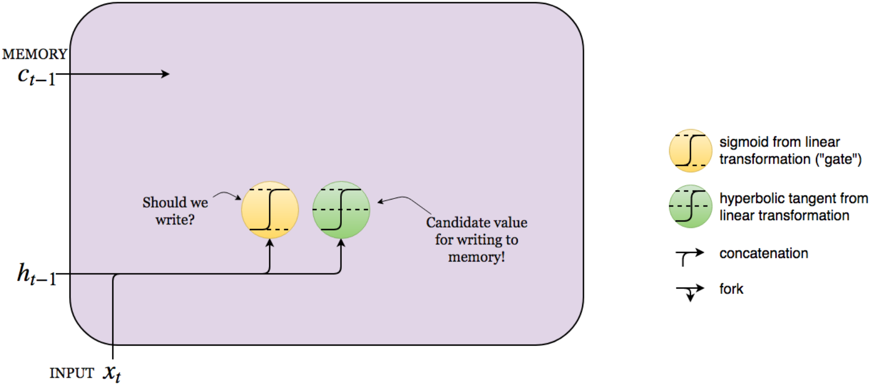
LSTM: recurrent neurons with memory.



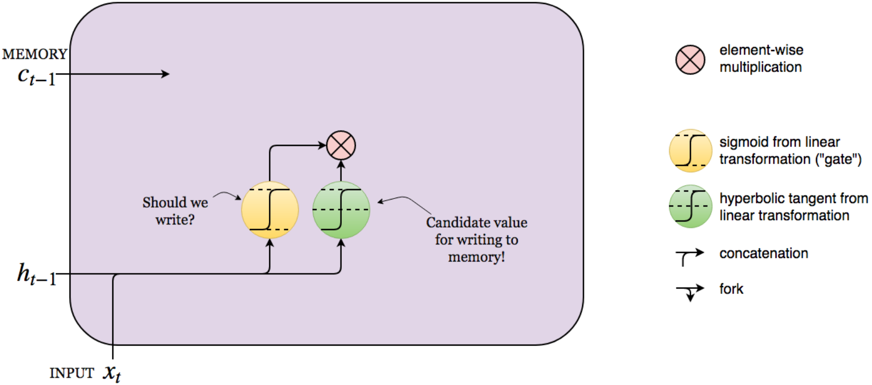
LSTM: transforming data: $c'_t = \tanh(A_c\,[x_t, h_{t-1}])$



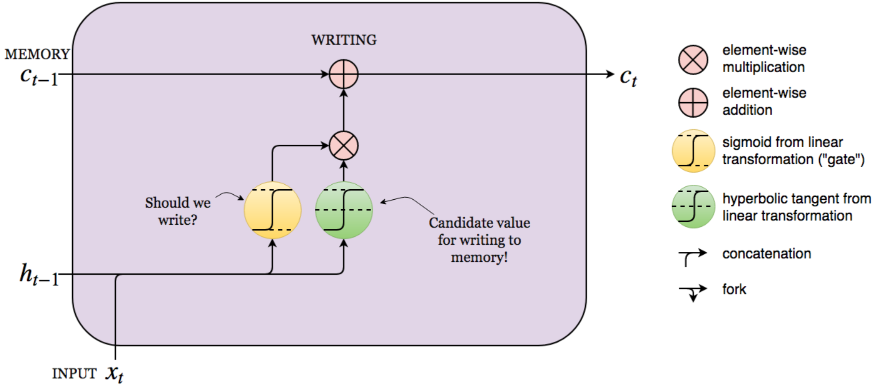
LSTM: writing gate: $w_t = \sigma(A_w\,[x_t, h_{t-1}])$



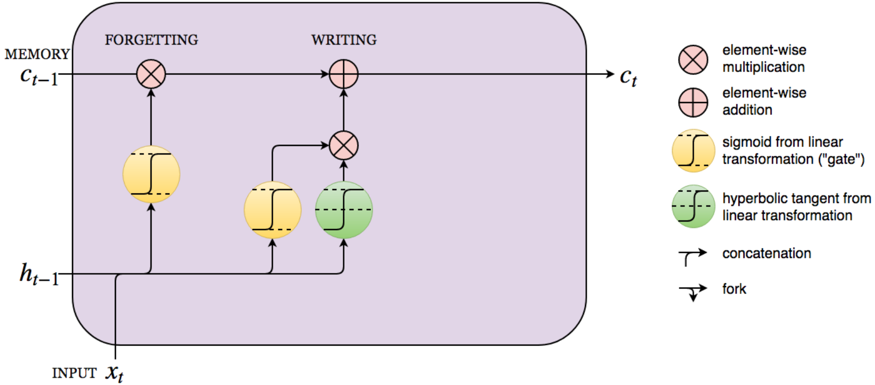
LSTM: $c_t = f_t \circ c_{t-1} + w_t \circ c'_t$



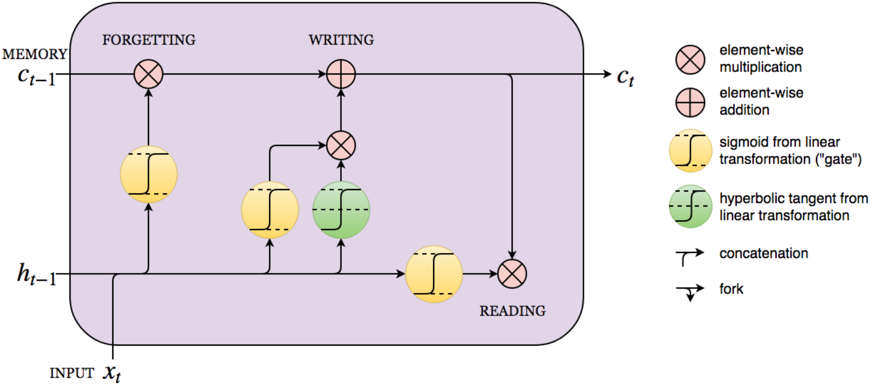
LSTM: $c_t = f_t \circ c_{t-1} + w_t \circ c'_t$



LSTM: $c_t = f_t \circ c_{t-1} + w_t \circ c'_t$



LSTM: $h_t = r_t \circ c_t$



LSTM: full scheme