Deep learning

김대식
서울대학교 융합과학기술대학원

MIPALaboratory
Machine Intelligence
& Pattern Analysis

What is Deep learning?

Wiki:

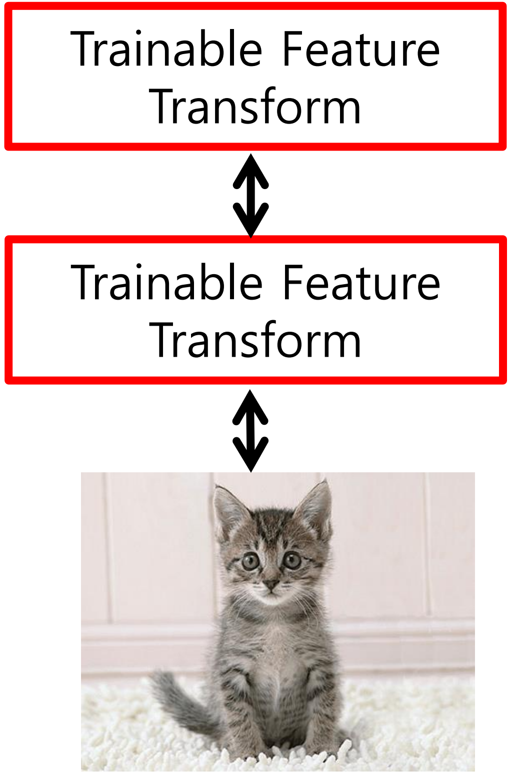
"**Deep learning** is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using multiple processing layers, with complex structures or otherwise, composed of multiple non-linear transformations."

Is it Brand new?

Neural Nets	McCulloch & Pitt 1943
Perception	Rosenblatt 1958
RNN	Grossberg 1973
CNN	Fukushima 1979
RBM	Hinton 1999
DBN	Hinton 2006
D-AE	Vincent 2008
AlexNet	Alex 2012
GoogLeNet	Szegedy 2015

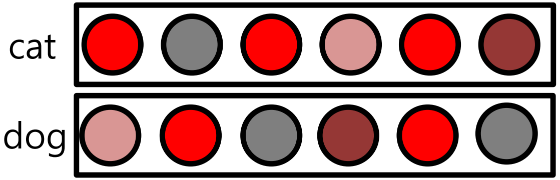
Learning Representations

- **ML/AI**: **how do we learn features?**
  - What is the fundamental rule?
  - What is the learning algorithms?
- **Neuroscience**: **how does the cortex learn perception**?
- **CogSci**: **how does the mind learn abstract concepts on top of less abstract ones?**
- **Deep learning addresses the problem of learning hierarchical representation with a single algorithm.**

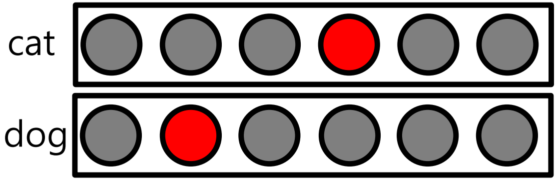


What is Deep Learning?

- Learning Deep Architecture for **Distributed Representation**
  - Distributed vs. Localist Representation



- Patterns on subset of "pool"
- More biological inspired (cortex)
- Neural Network
- Suitable for large VC-dimension problem
- Highly connected



- Matter of local activation
- Smoothness prior
- Template matching with input -> (linear) combination of out-put value of matching
- Manifold learning
- Need as many examples as va-riations of interest

Applications

Scene Recognition (CNN)

Predictions:

- **Type of environment:** outdoor
- **Semantic categories:** rock_arch:0.63, arch:0.30,
- **SUN scene attributes:** rugged, natural light, dry, climbing, far-away horizon, touring, rocky, open area, warm, sand

Visual Style Recognition (CNN)

Segmentation (DeconvNet)

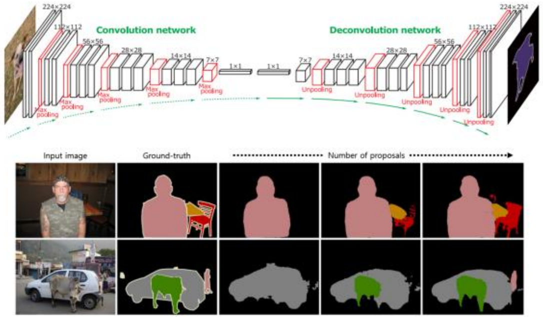
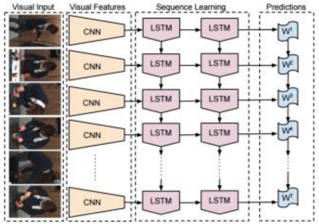
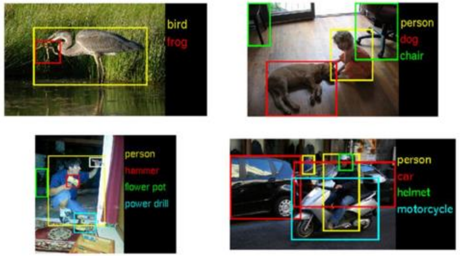


Image Captioning (CNN+LSTM)



A black and white cat is sitting on a chair.

Object Detection (R-CNN)

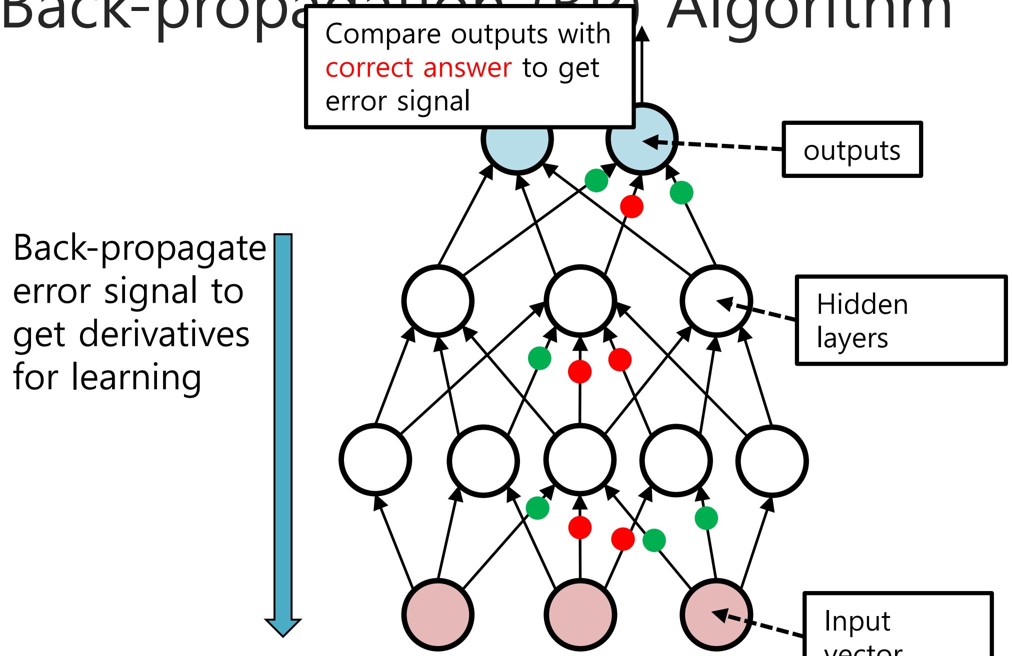


Detection ≈ Localization + Classification

Neural Style (CNN)

Brief History of Neural Network

- 1940's : Neural Network (NN)
- 1970's : Back-propagation (BP) Algorithm



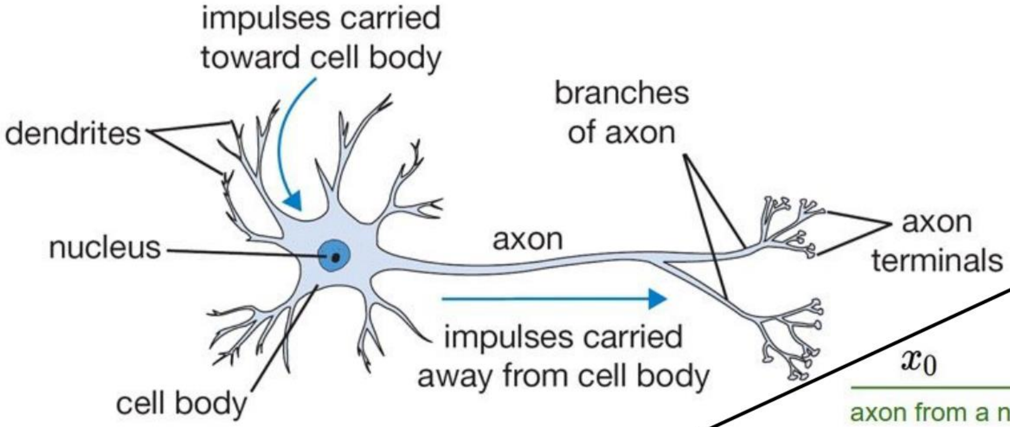
Brief History of Neural Network

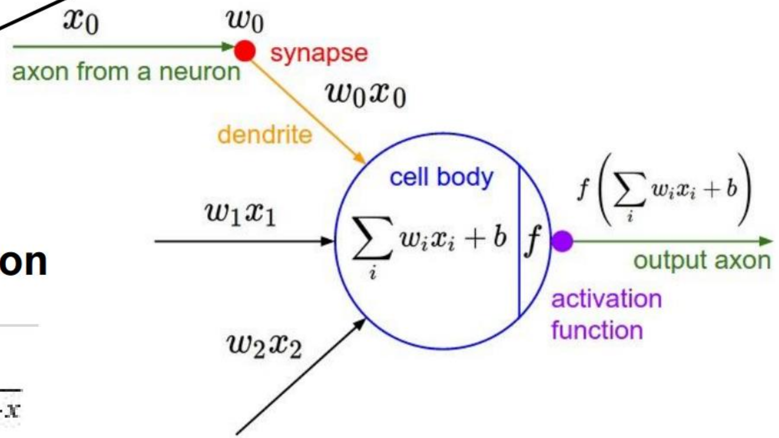
- Limitations of Back-propagation method
  - Need labeled training data
    - Almost all data is unlabeled
  - The training time does not scale well
    - Computation is increase exponentially as the number of layer grows
  - Stuck in poor local optima
    - Effect of gradient method cannot propagate well through multi-layer
    - Overfitting

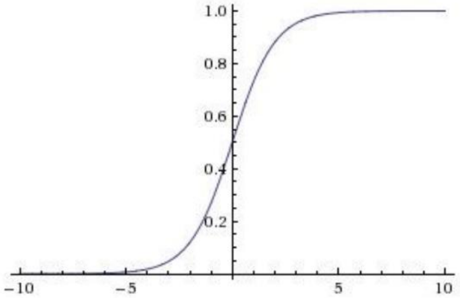
Why Deep Learning Again?

- 3 major reasons
  - Improvement in learning algorithm
    - Unsupervised pre-training techniques
    - Faster (x10)
    - Better performance
  - Increased computation power
    - CPU (x60 faster)
    - + GPU (x1000 faster)
  - Plenty of data for training
    - Big data

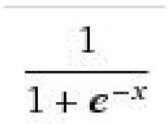
Neuron in Neural Networks



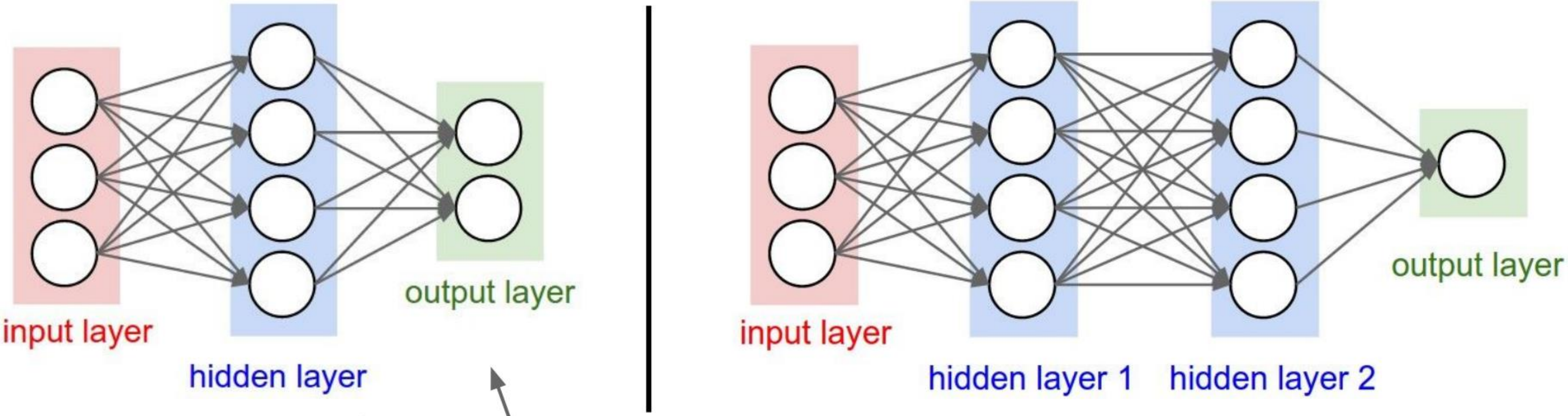




sigmoid activation function



Architectures of Neural Networks



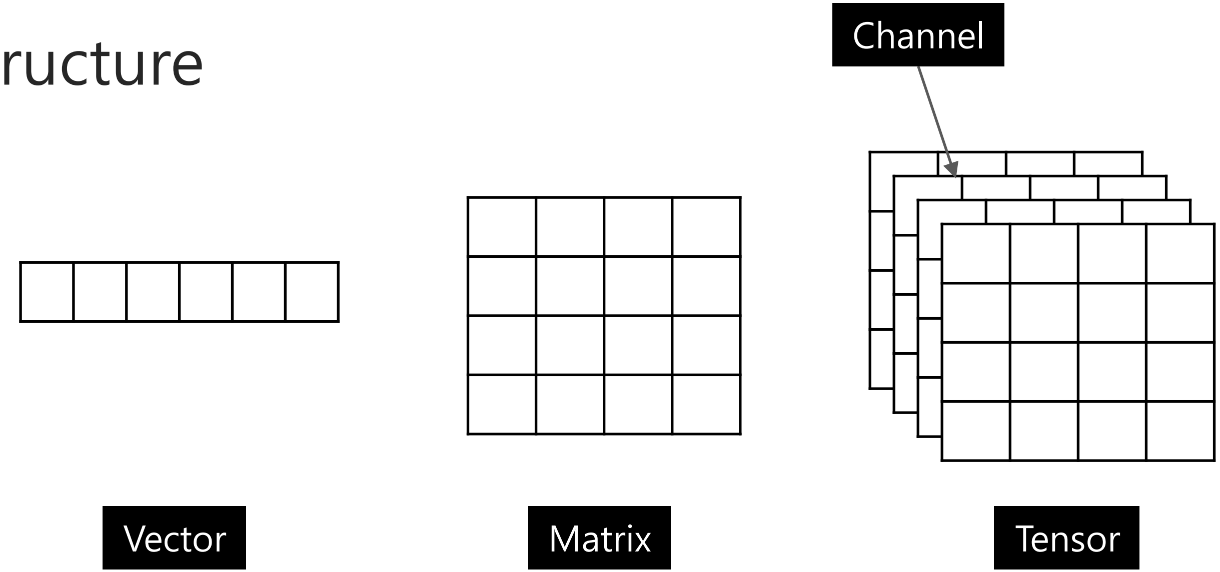
"2-layer Neural Net", or
"1-hidden-layer Neural Net"

"Fully-connected" layers

"3-layer Neural Net", or
"2-hidden-layer Neural Net"

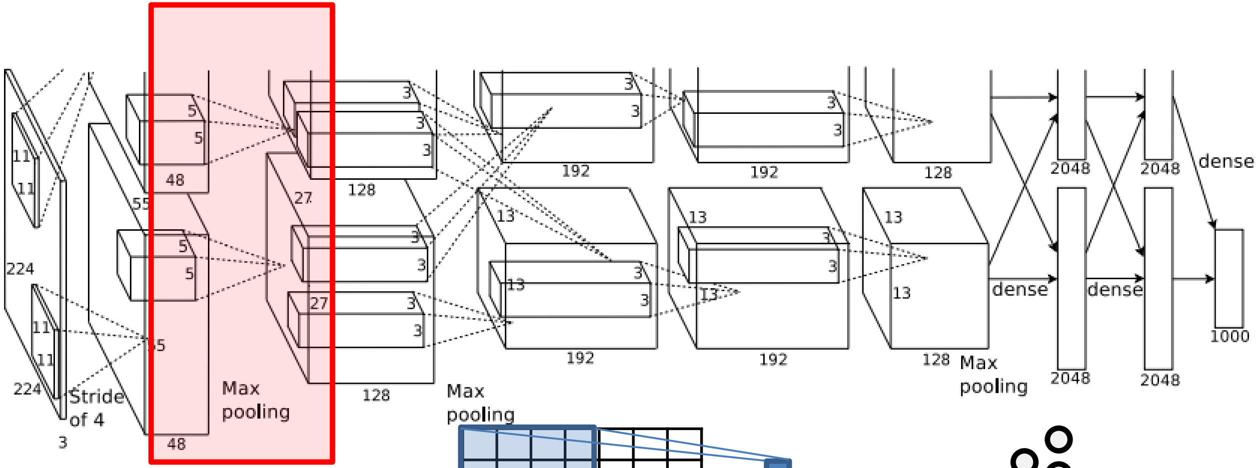
Deep Learning In Computer Vision

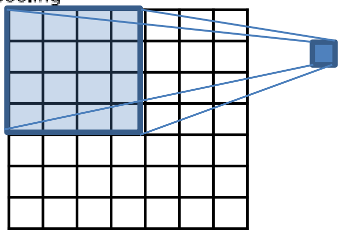
- Data structure



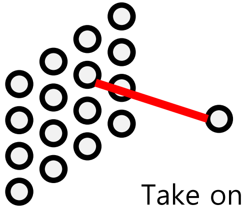
Deep Learning In Computer Vision

- CNN feature
  - AlexNet





=

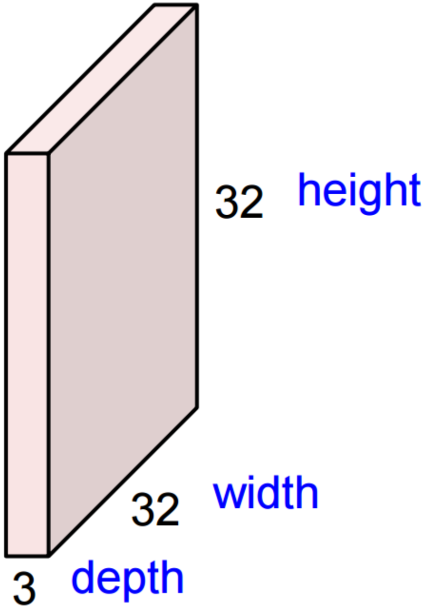


Take only the maximum

cf) average pooling

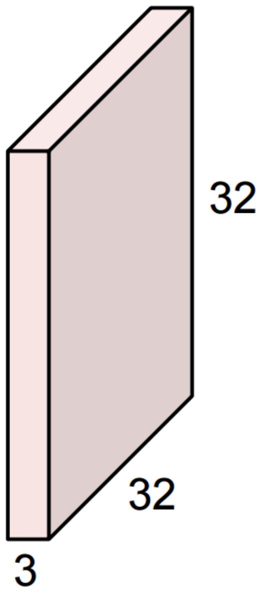
Convolution Layer

32x32x3 image

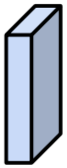


Convolution Layer

32x32x3 image



5x5x3 filter

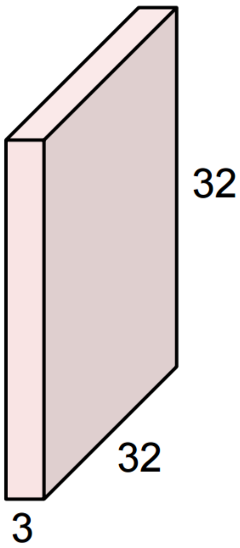


Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

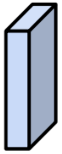
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image



5x5x3 filter

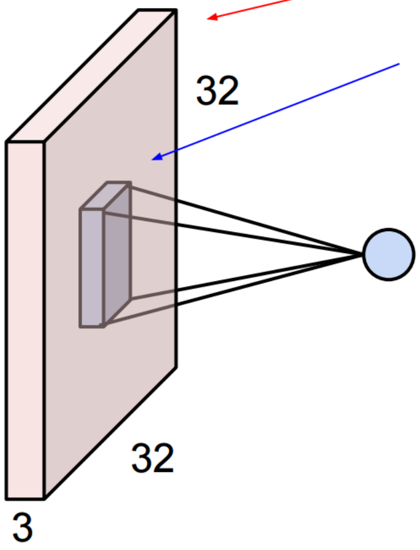


Convolve the filter with the image i.e. "slide over the image spatially, computing dot products"

Convolution Layer

32x32x3 image

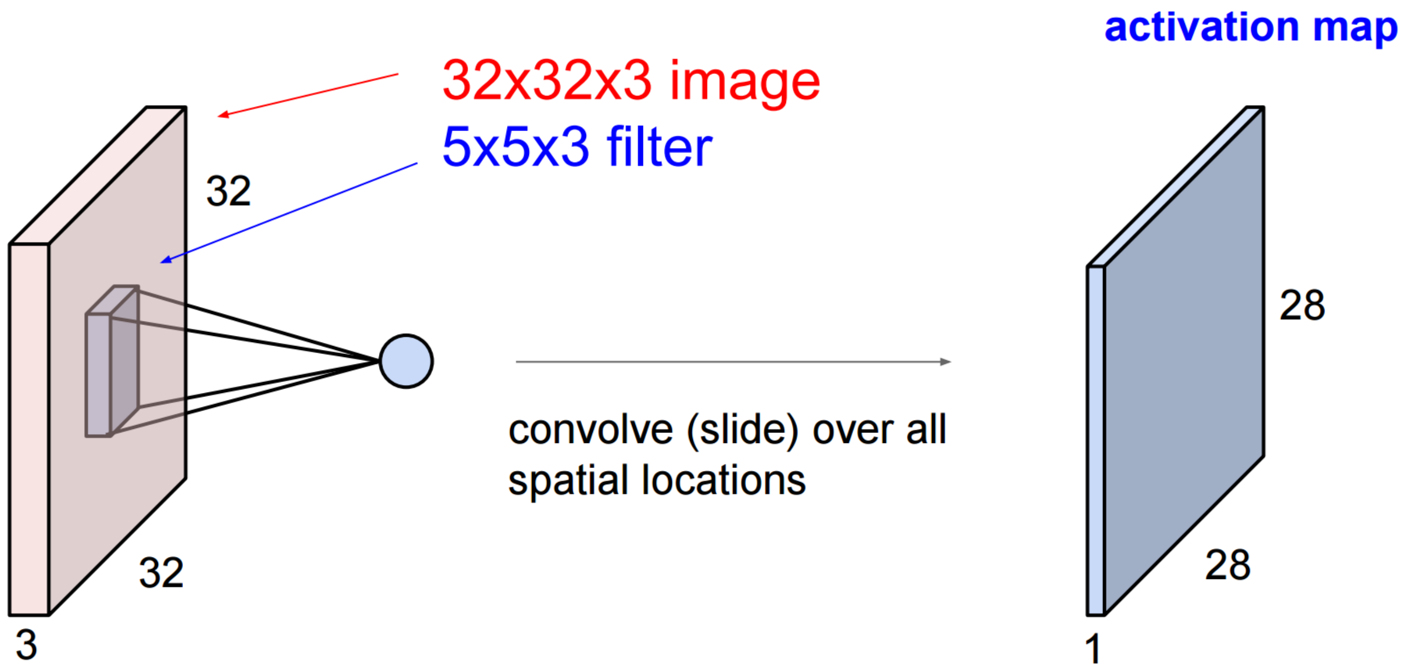
5x5x3 filter w



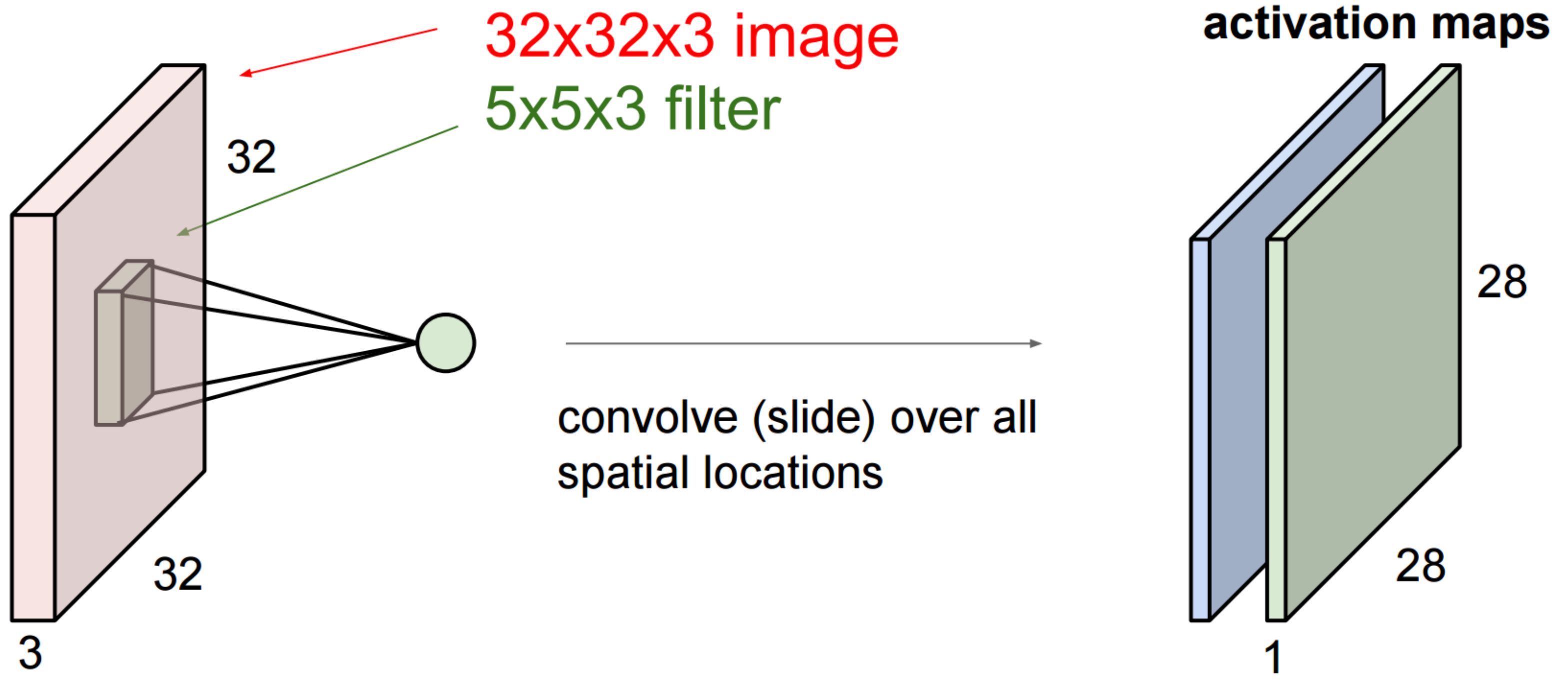
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Convolution Layer

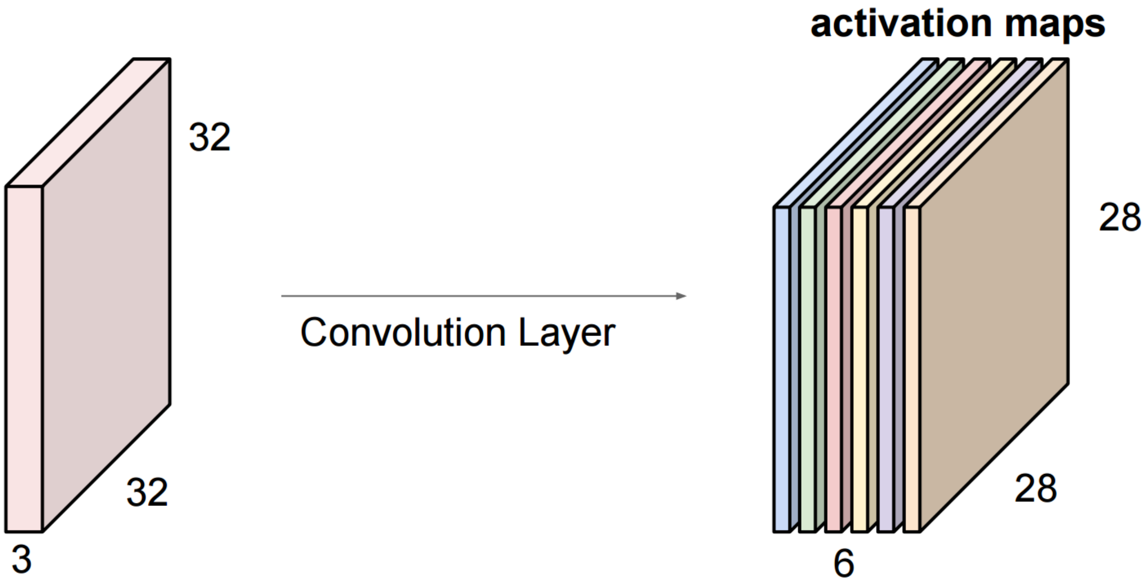


Convolution Layer



Convolution Layer

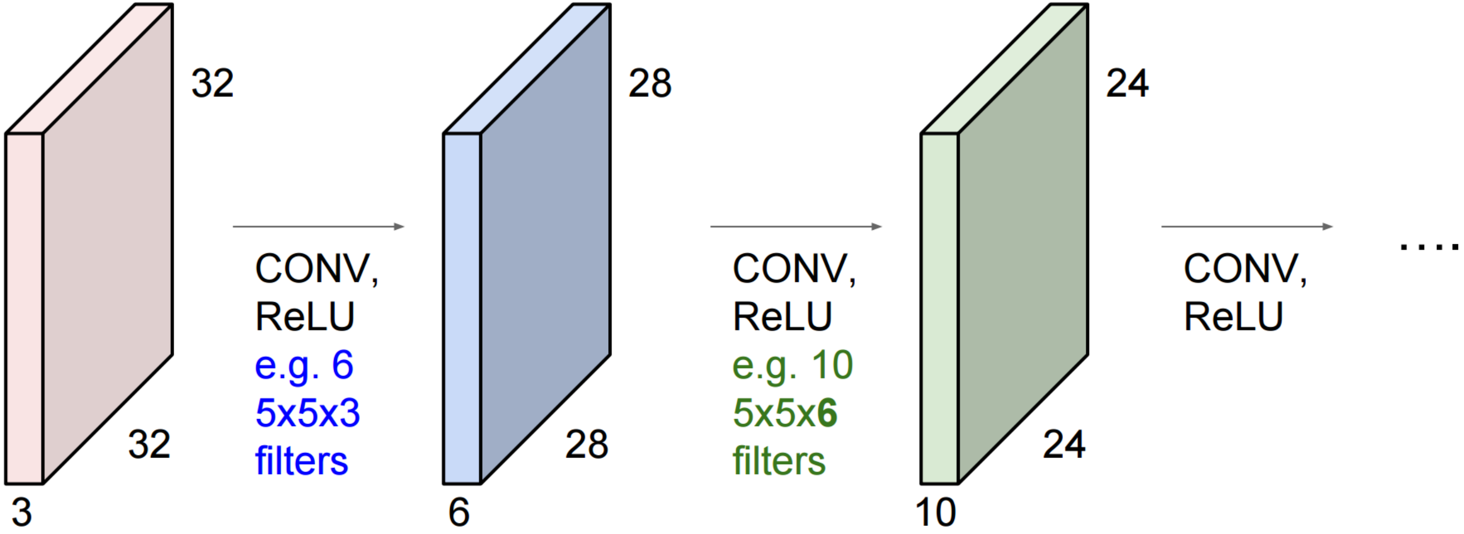
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

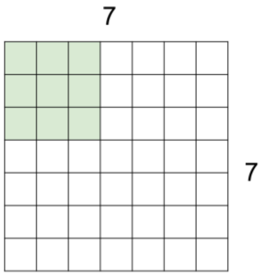
Convolution Layer

Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



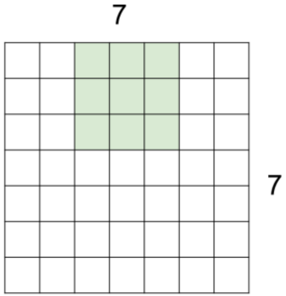
Convolution Layer : stride

A closer look at spatial dimensions:



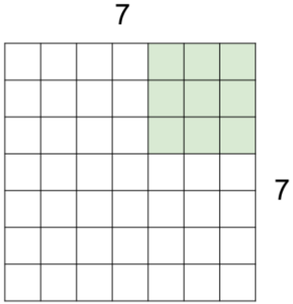
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

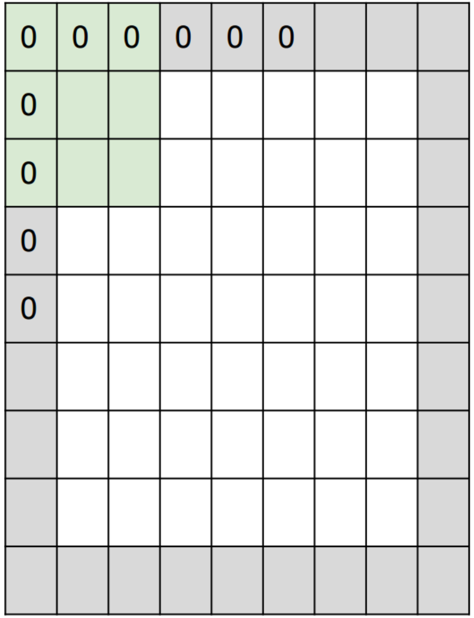
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

Convolution Layer : zero pad

In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

Convolution Layer

Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

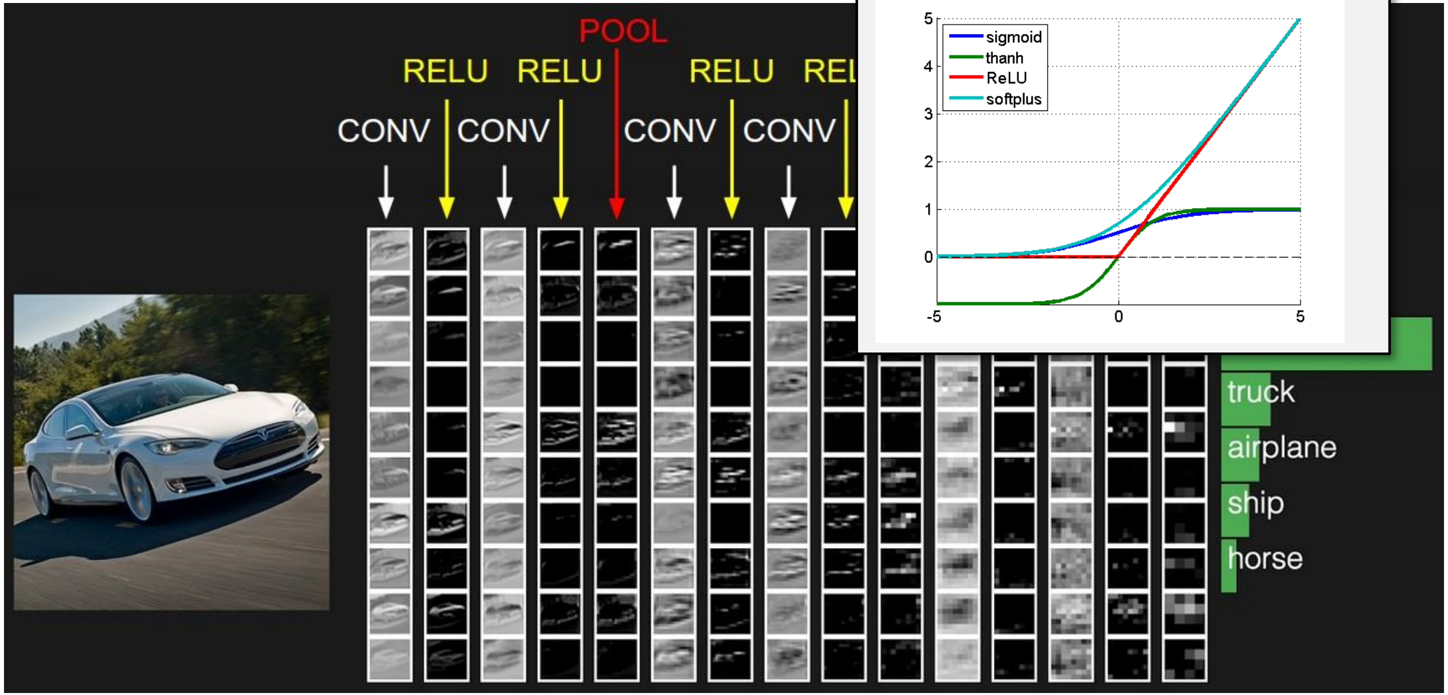
Common settings:

K = (powers of 2, e.g. 32, 64, 128, 512)

- F = 3, S = 1, P = 1
- F = 5, S = 1, P = 2
- F = 5, S = 2, P = ? (whatever fits)
- F = 1, S = 1, P = 0

Deep Learning In Computer Vision

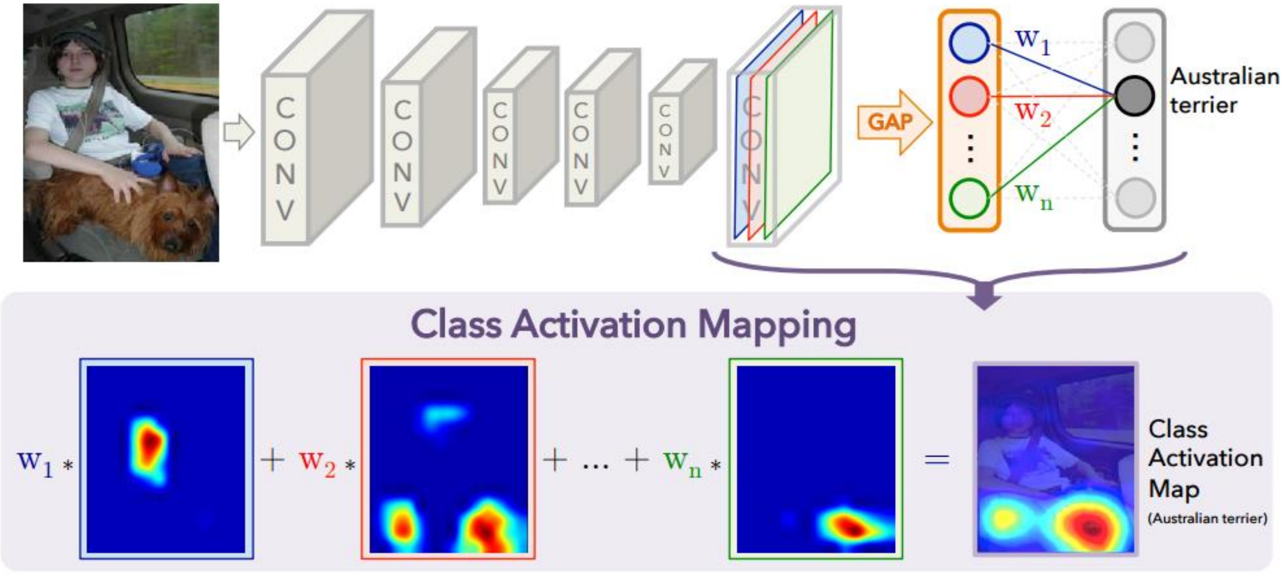
- CNN feature



ReLU = Rectified Linear Unit
$f(x) = \max(0, x)$

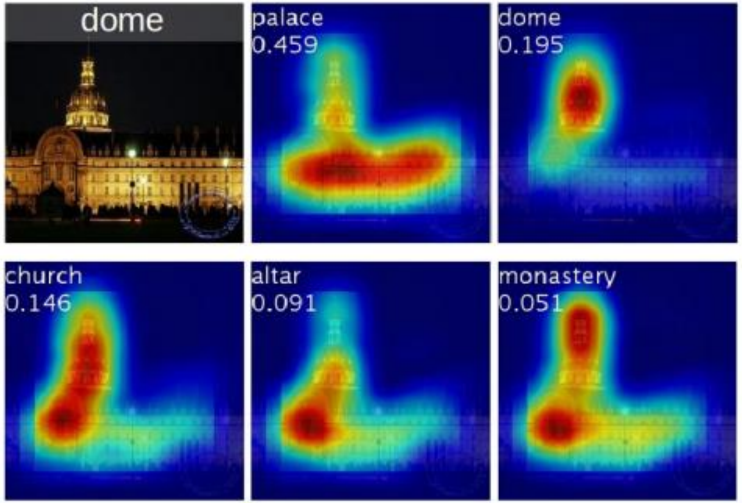
Deep Learning In Computer Vision

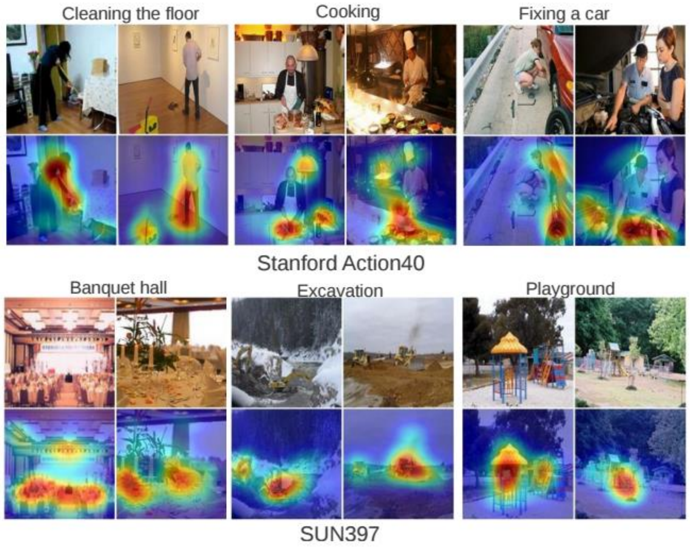
- Learning Deep Features for Discriminative Localization



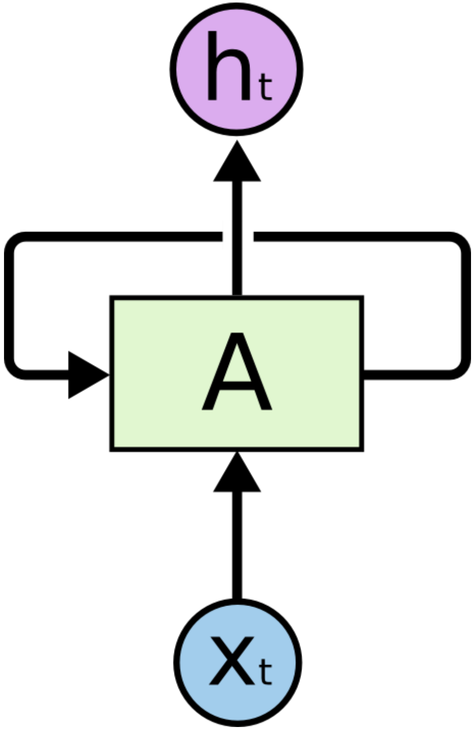
Deep Learning In Computer Vision

- Learning Deep Features for Discriminative Localization



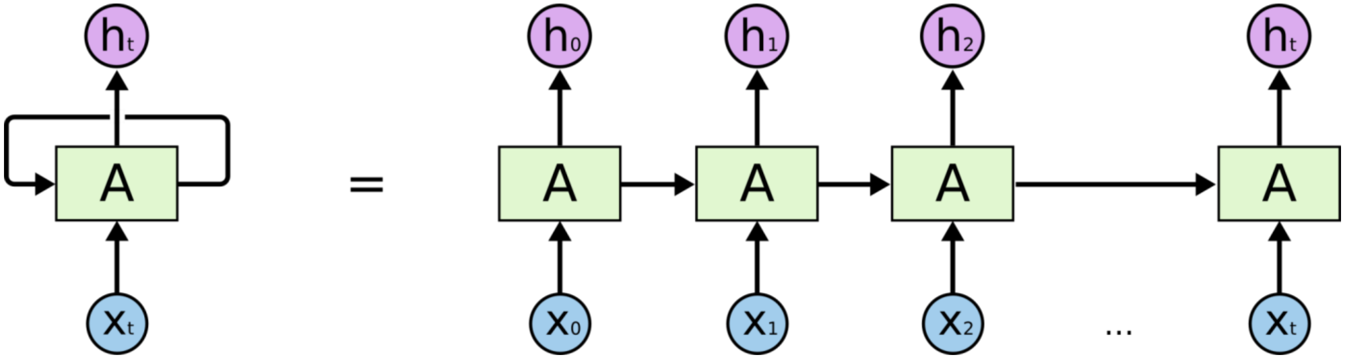


Recurrent Neural Network



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

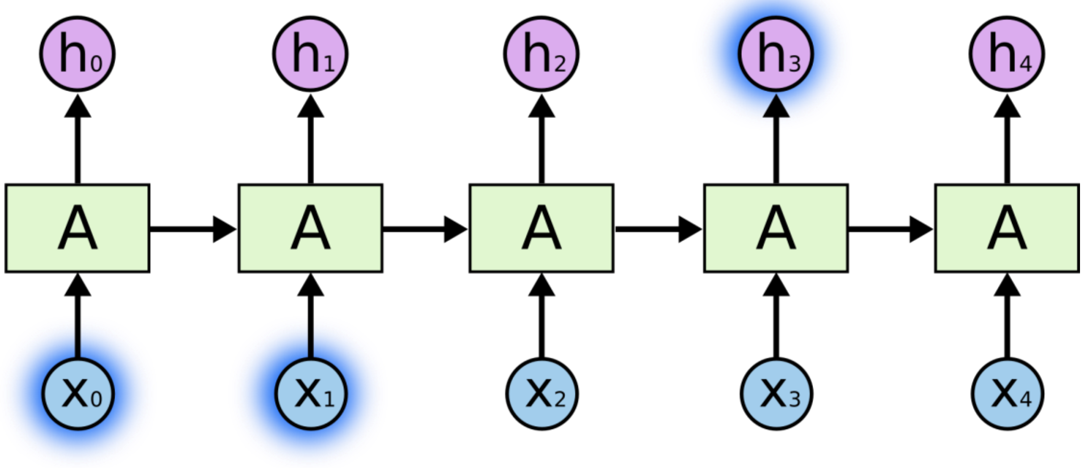
Recurrent Neural Network



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

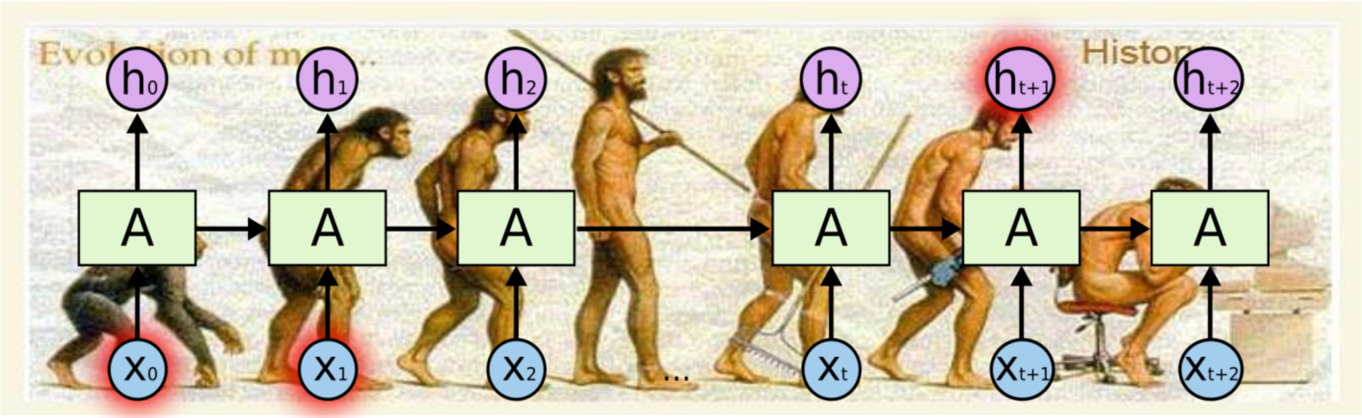
Long-Term Dependencies

The **clouds** are in the **sky**



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

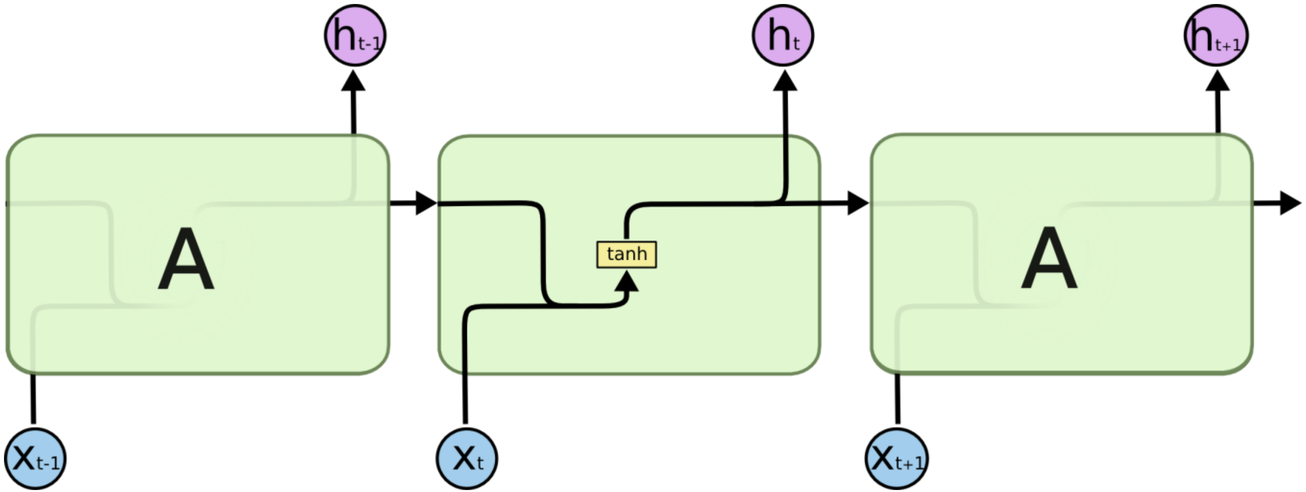
Longer-Term Dependencies



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM comes in!

Long Short Term Memory

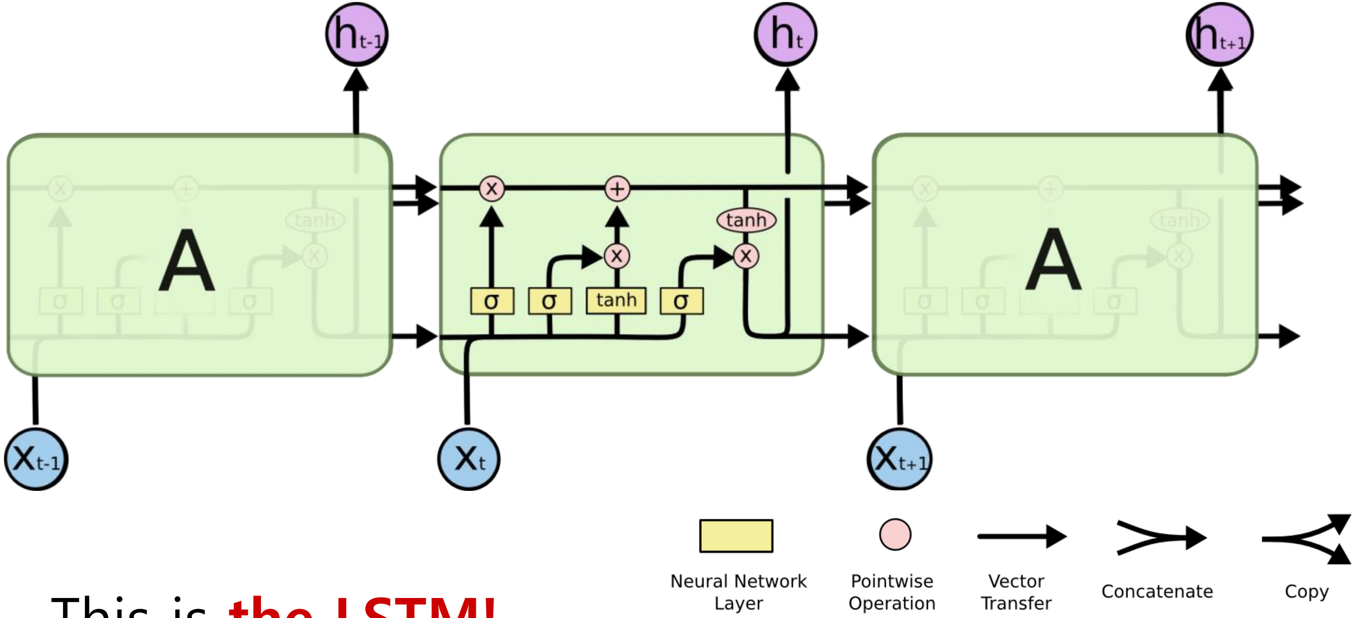


This is just a standard RNN.

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM comes in!

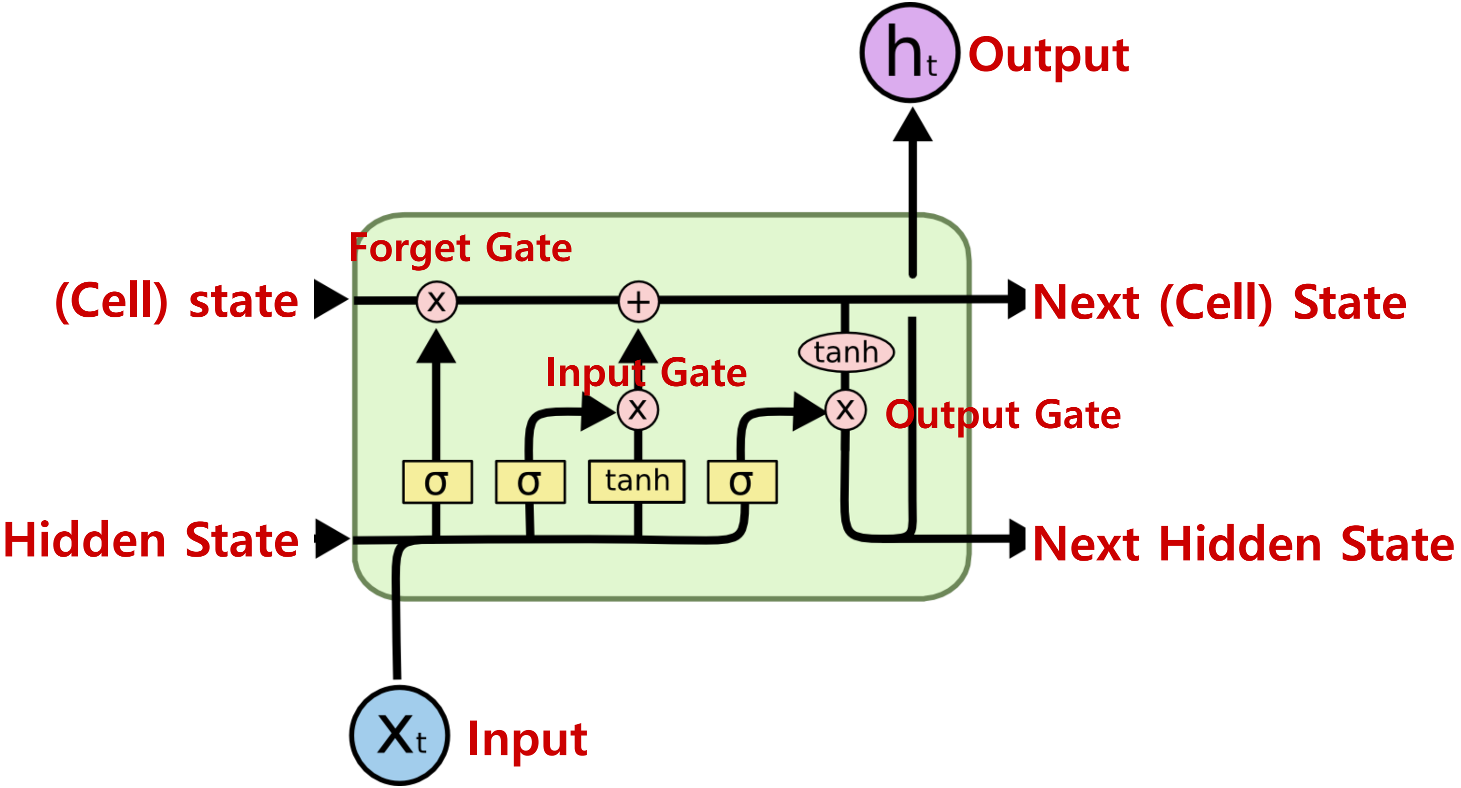
Long Short Term Memory



This is **the LSTM!**

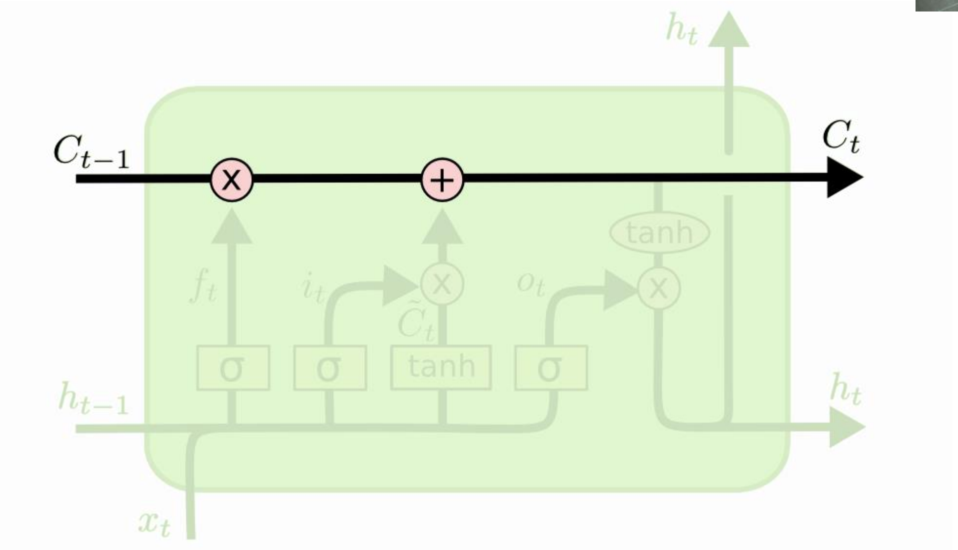
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

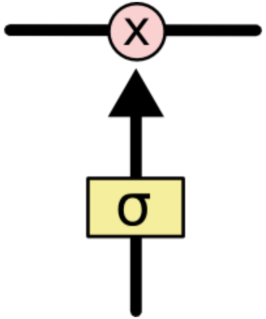
Overall Architecture



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

The Core Idea

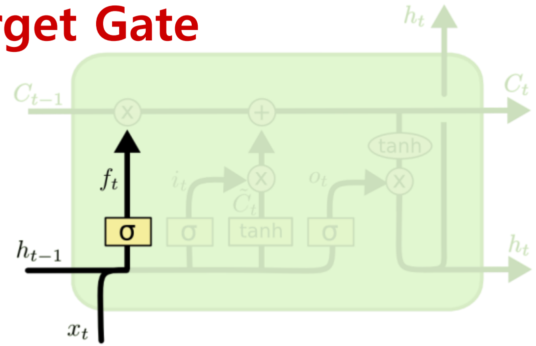




http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Step-by-Step

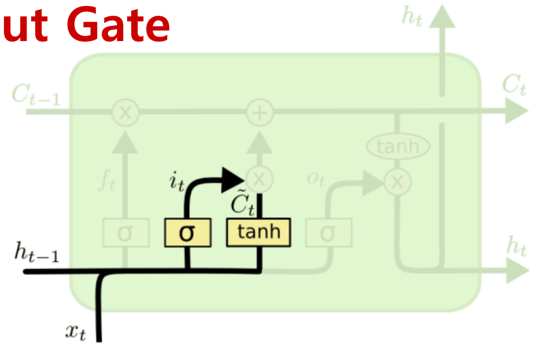
Forget Gate



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

Decide what information we're going to **throw away** from the cell state.

Input Gate



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

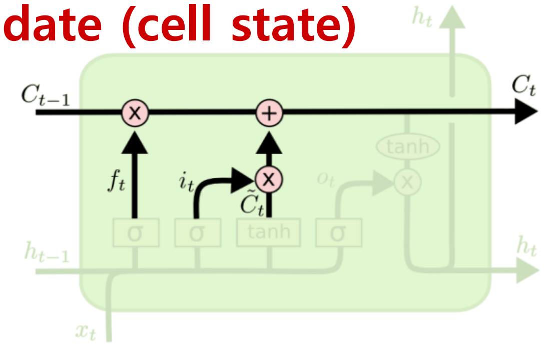
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Decide what new information we're going to **store** in the cell state.

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Step-by-Step

Update (cell state)

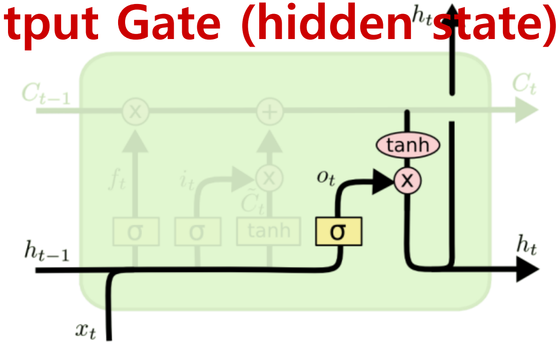


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Update, scaled by how much we decide to update

: input_gate*curr_state + forget_gate*prev_state

Output Gate (hidden state)



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

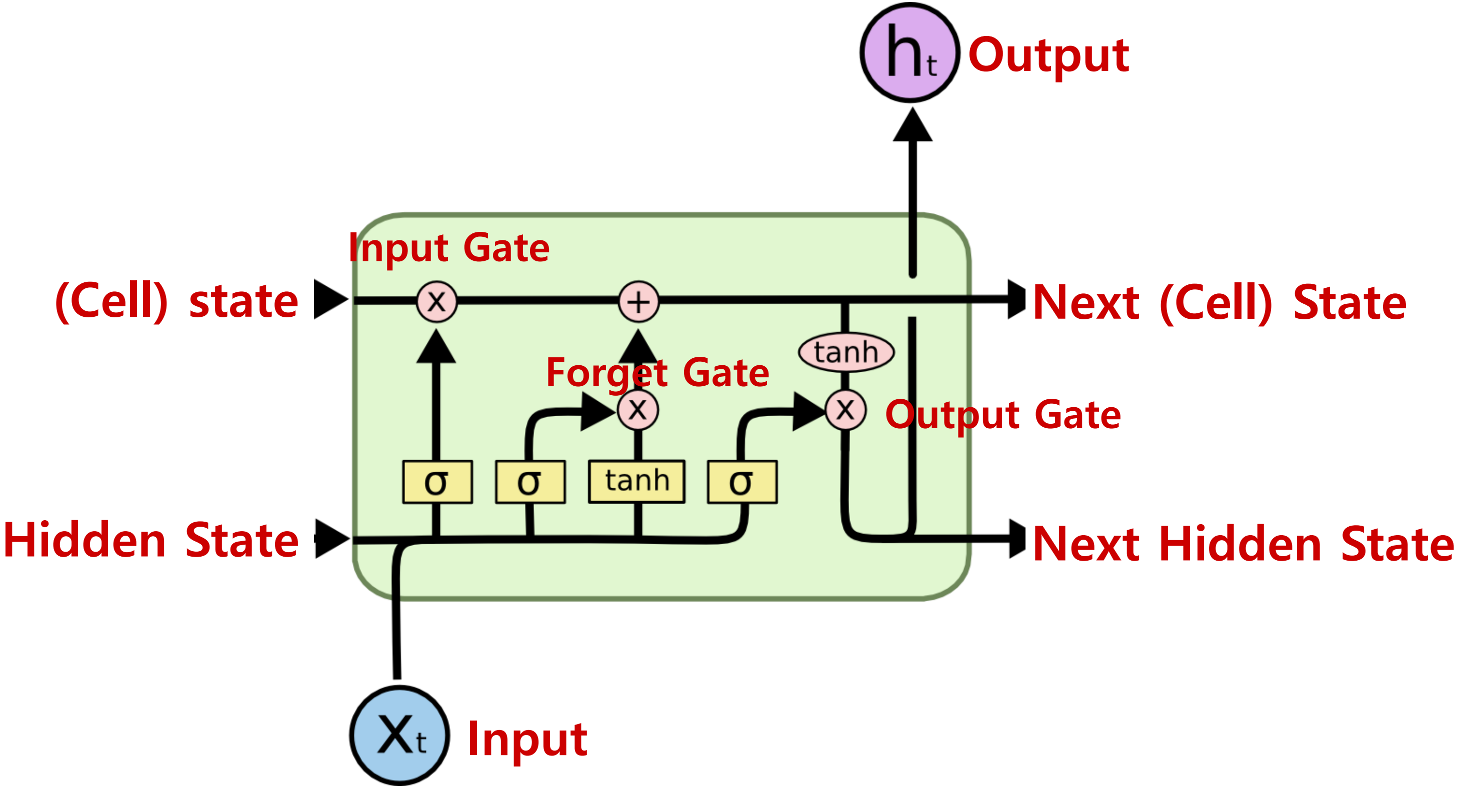
$$h_t = o_t * \tanh\left(C_t\right)$$

Output based on the updated state

: output_gate*updated_state

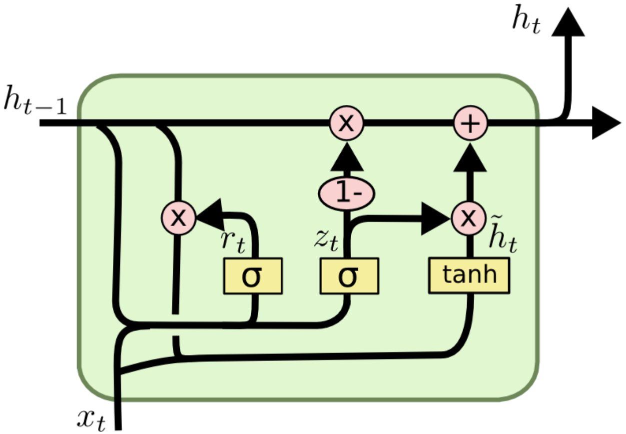
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Again



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Gated Recurrent Unit



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." *arXiv preprint arXiv:1406.1078* (2014).

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Applications

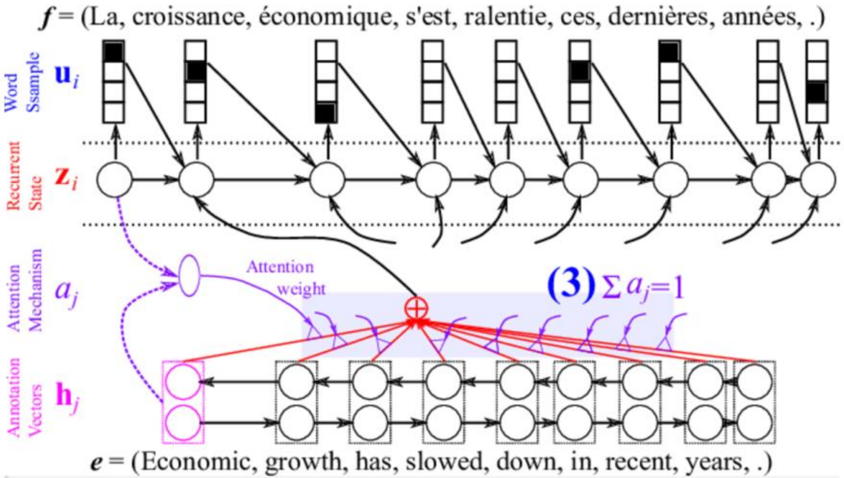


Figure 5. The relevance scores returned by the attention mechanism are normalized to sum to 1, which helps us interpret them as probabilities. From this probabilistic perspective, we compute the expectation of the annotation vectors under this distribution.

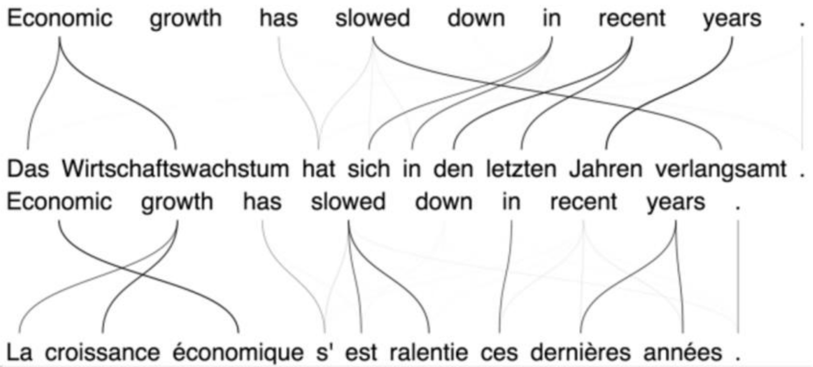


Figure 6. Sample translations made by the neural machine translation model with the soft-attention mechanism. Edge thicknesses represent the attention weights found by the attention model.