COMP4901K/Math4824B
Machine Learning for Natural Language Processing

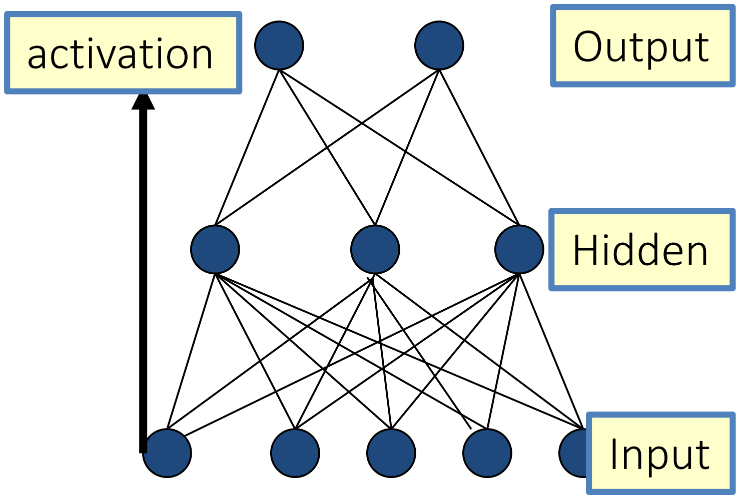
Lecture 11: CNN

Instructor: Yangqiu Song

Slides credits: Dan Roth

Recap: Multi-Layer Perceptrons

- Multi-layer network
  - A global approximator
  - Different rules for training it
- The Back-propagation
  - Forward step
  - Back propagation of errors



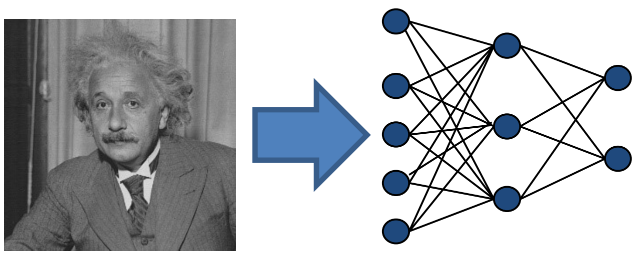
- Congrats! Now you know the hardest concept about neural networks!

- Today:
  - Convolutional Neural Networks

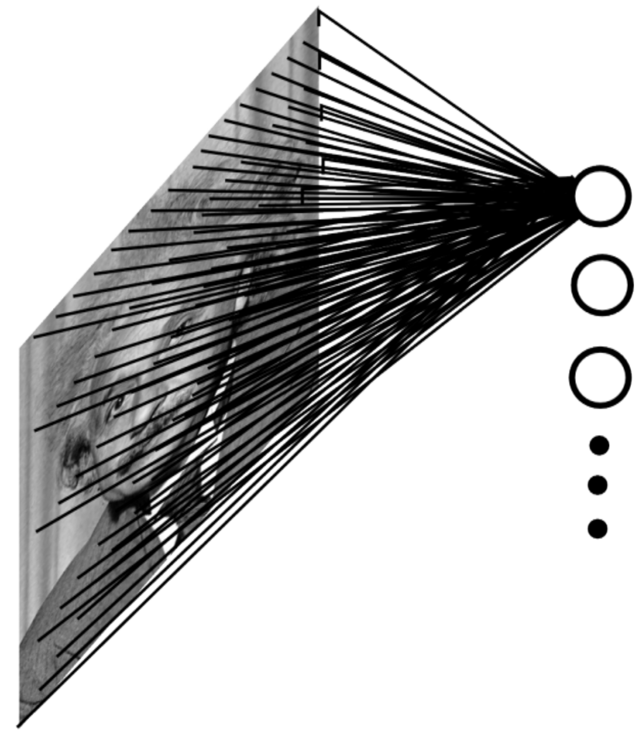
Receptive Fields

- The **receptive field** of an individual sensory neuron is the particular region of the sensory space (e.g., the body surface, or the retina) in which a stimulus will trigger the firing of that neuron.
  - Designing “proper” receptive fields for the input Neurons is a significant challenge.

- Consider a task with image inputs
  - Receptive fields should give expressive features from the raw input to the system
  - How would you design the receptive fields for this problem?

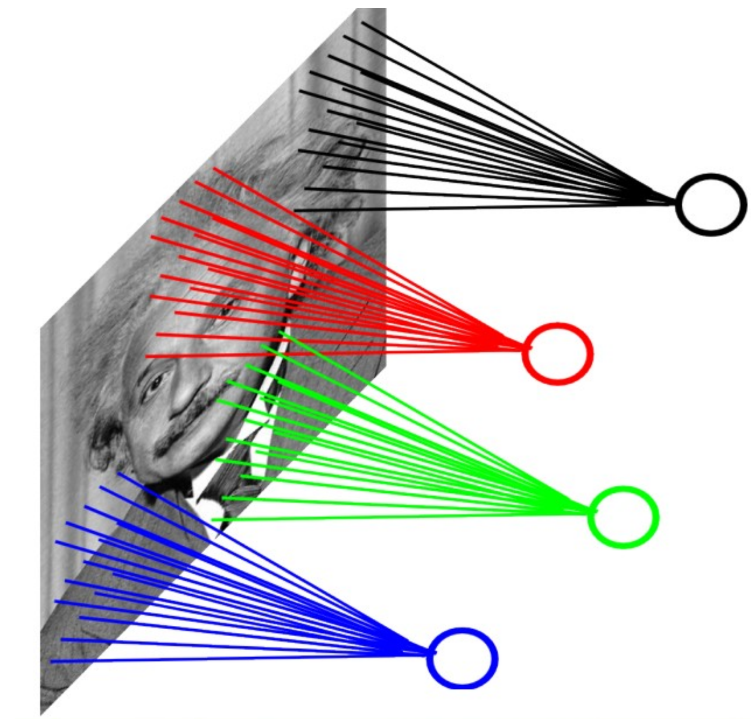


- A **fully connected layer**:
  - Example:
    - 100x100 images
    - 1000 units in the input
  - Problems:
    - 10^7 edges!
    - Spatial correlations lost!
    - Variables sized inputs.



Slide Credit: Marc'Aurelio Ranzato

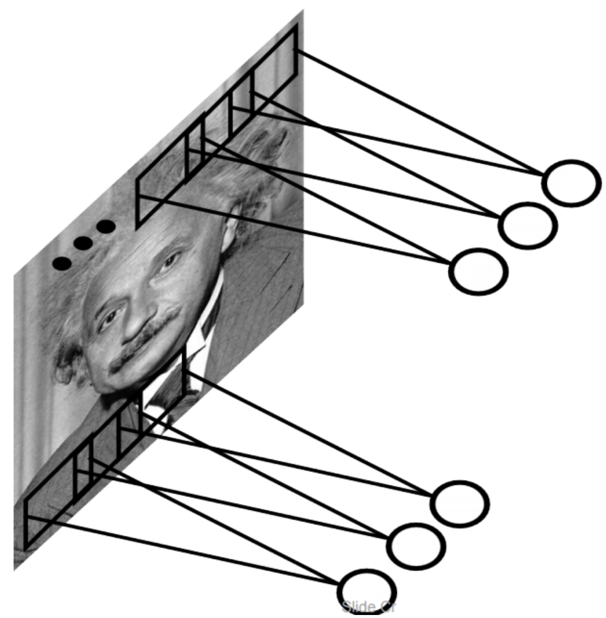
- Consider a task with image inputs:
- A **locally connected layer**:
  - Example:
    - 100x100 images
    - 1000 units in the input
    - Filter size: 10x10
  - Local correlations preserved!
  - Problems:
    - 10^5 edges
    - This parameterization is good when input image is registered (e.g., face recognition).
    - Variable sized inputs, again.



Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer

- A solution:
  - **Filters** to capture different patterns in the input space.
    - **Share** parameters across different locations (assuming input is stationary)
    - Convolutions with learned filters
  - Filters will be **learned** during training.
  - The issue of variable-sized inputs will be resolved with a **pooling** layer.



So what is a convolution?

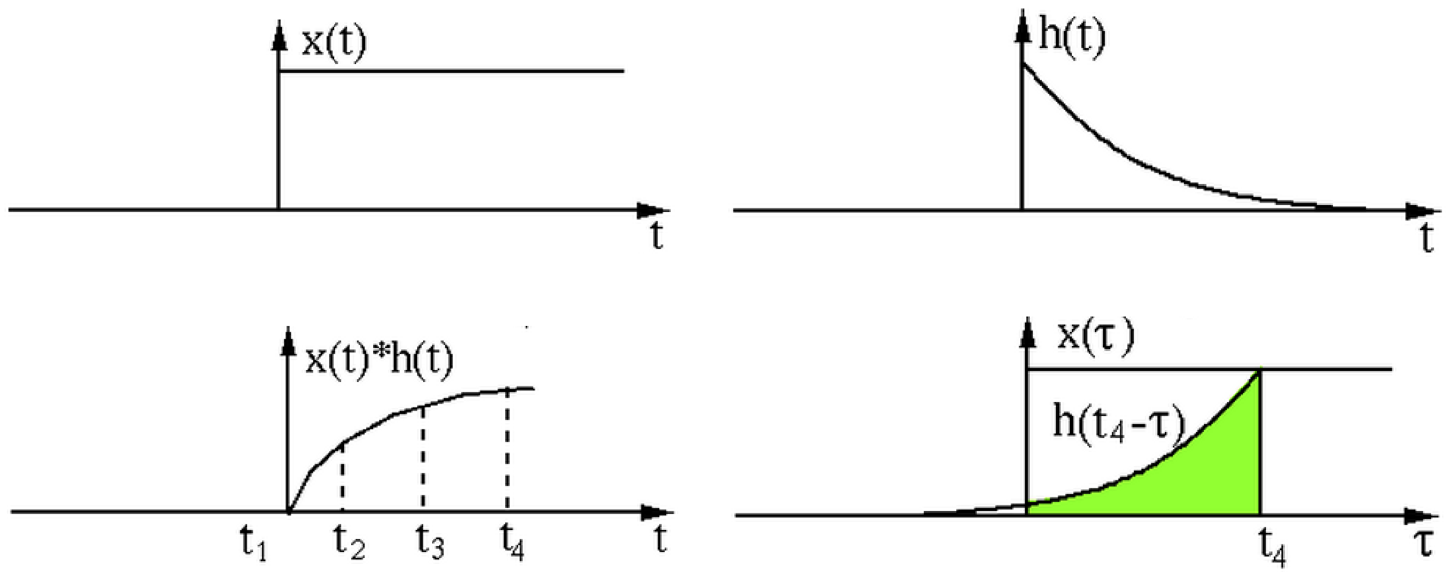
Convolution Operator

- Convolution operator: $*$
  - takes two functions and gives another function
- One dimension:

$$(x * h)(t) = \int x(\tau)h(t-\tau)d\tau$$

$$(x * h)[n] = \sum_m x[m]h[n-m]$$

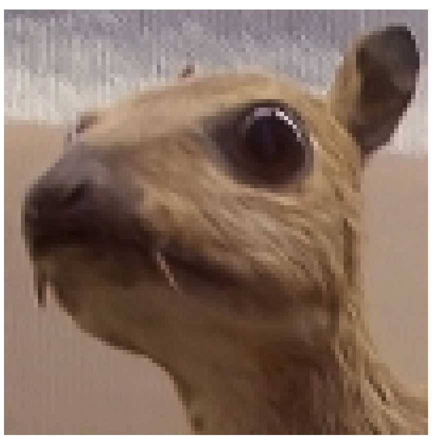
"Convolution" is very similar to "cross-correlation", except that in convolution one of the functions is flipped.



Convolution Operator (2)

- Convolution in two dimension:
  - The same idea: flip one matrix and slide it on the other matrix
  - Example: edge detection kernel:

Input image



Convolution Kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Feature map

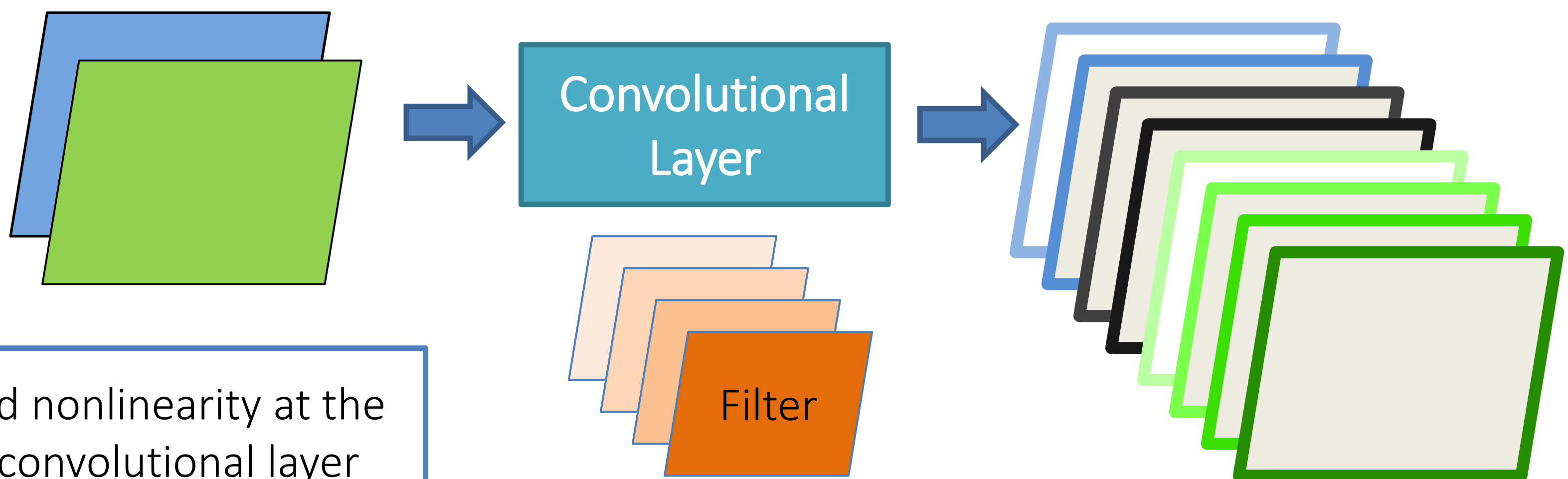
Try other kernels: http://setosa.io/ev/image-kernels/

Demo of CNN

- https://setosa.io/ev/image-kernels/

Convolutional Layer

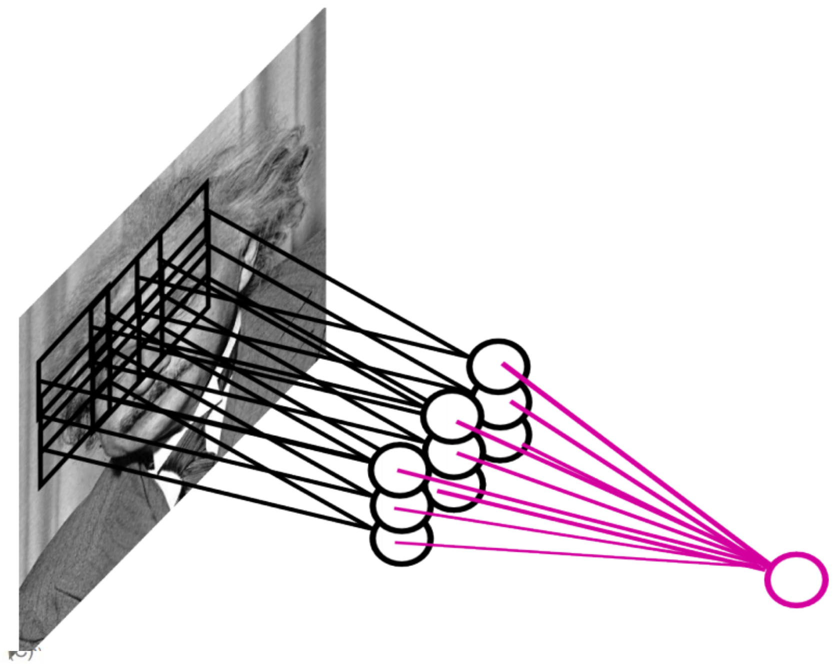
- The convolution of the **input (vector/matrix)** with weights **(vector/matrix)** results in a **response vector/matrix**.
- We can have **multiple filters** in each convolutional layer, each producing an output.
- If it is an intermediate layer, it can have **multiple inputs**!



One can add nonlinearity at the output of convolutional layer

Pooling Layer

- How to handle variable sized inputs?
  - A layer which reduces inputs of different size, to a fixed size.
  - Pooling



Slide Credit: Marc'Aurelio Ranzato

Pooling Layer

- How to handle variable sized inputs?
  - A layer which reduces inputs of different size, to a fixed size.
  - Pooling
  - Different variations
    - Max pooling

$$h_i[n] = \max_{i \in N(n)} \tilde{h}\,[i]$$

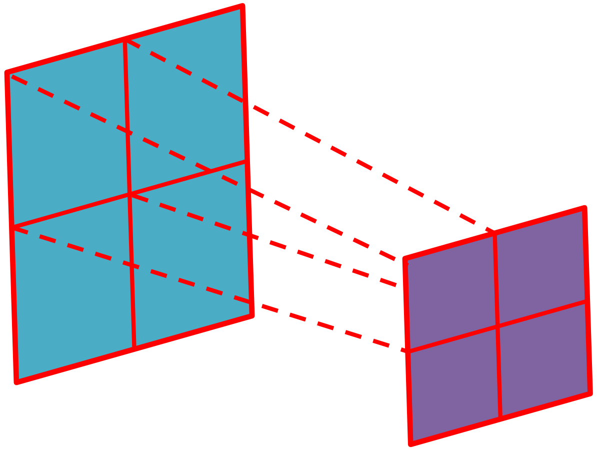
    - Average pooling

$$h_i[n] = \frac{1}{n} \sum_{i \in N(n)} \tilde{h}\,[i]$$

    - L2-pooling

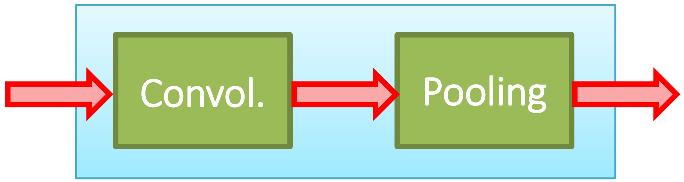
$$h_i[n] = \frac{1}{n} \sqrt{\sum_{i \in N(n)} \tilde{h}^2\,[i]}$$

    - etc

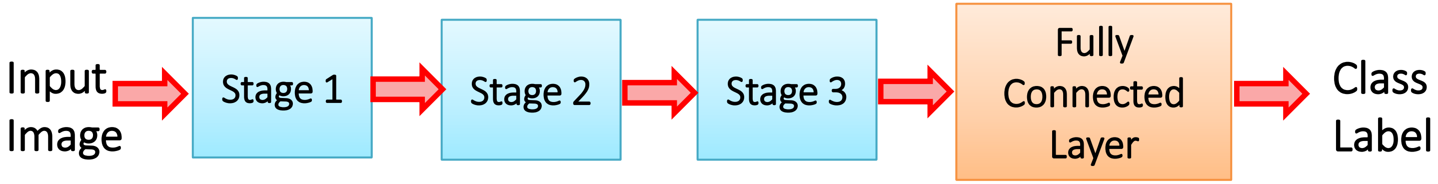


Convolutional Nets

- One stage structure:



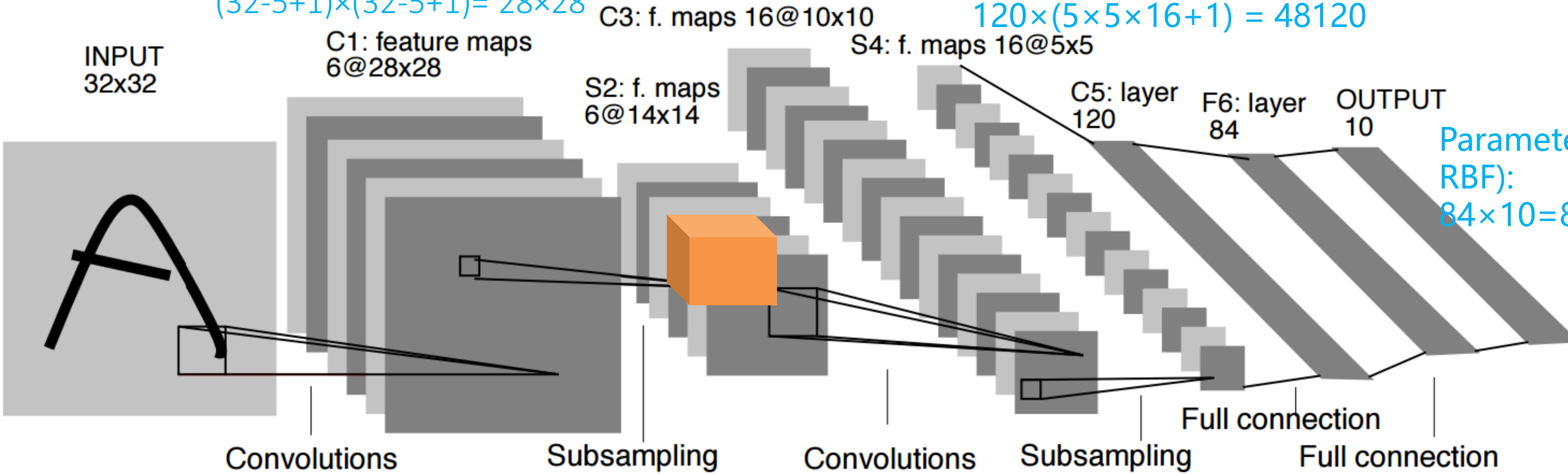
- Whole system:



An example system (LeNet)

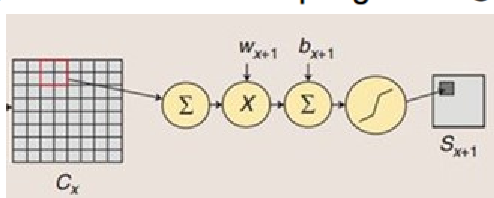
$(32-5+1)\times(32-5+1)= 28\times28$

Parameters: $120\times(5\times5\times16+1) = 48120$



Parameters(RBF): $84\times10=840$

Parameters: $(5*5+1)*6=156$
(Conv+bias)*channels



Parameters: $(1+1)*6=12$
(Conv+bias)*channels

Parameters:
$(5\times5\times3+1)\times6+(5\times5\times4+1)\times9+5\times5\times6+1 = 1516$

Parameters:
$16\times2 = 32$

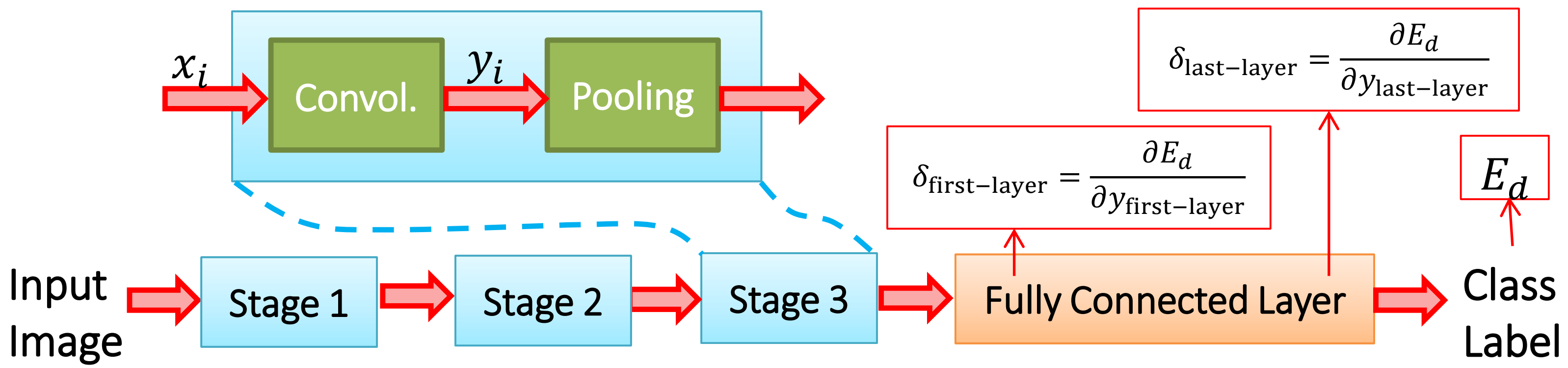
Parameters:
$(120+1)\times84 = 10164$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

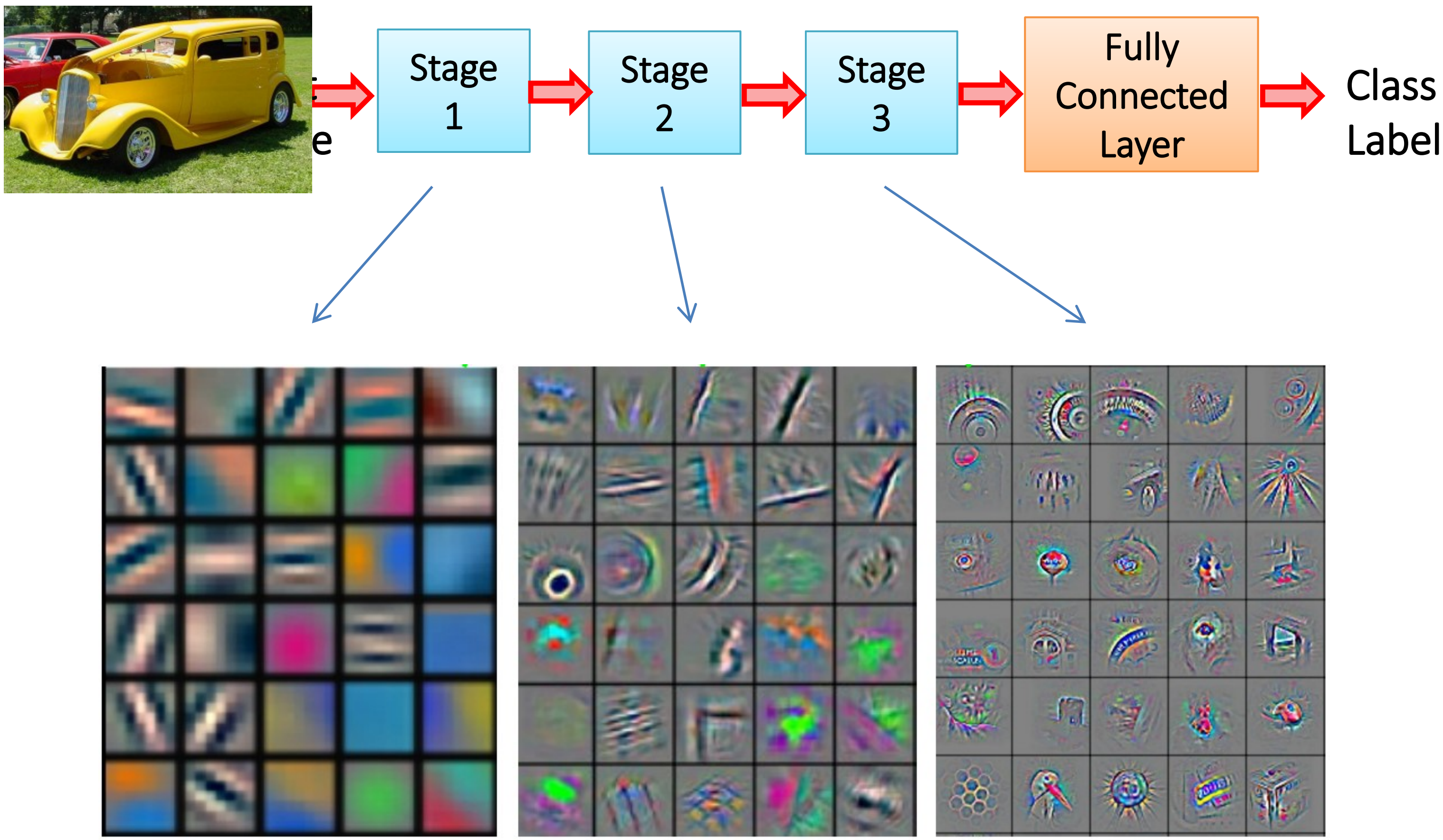
3 4 6

Training a ConvNet

- The same procedure from Back-propagation applies here.
  - Remember in backprop we started from the error terms in the last stage, and passed them back to the previous layers, one by one.



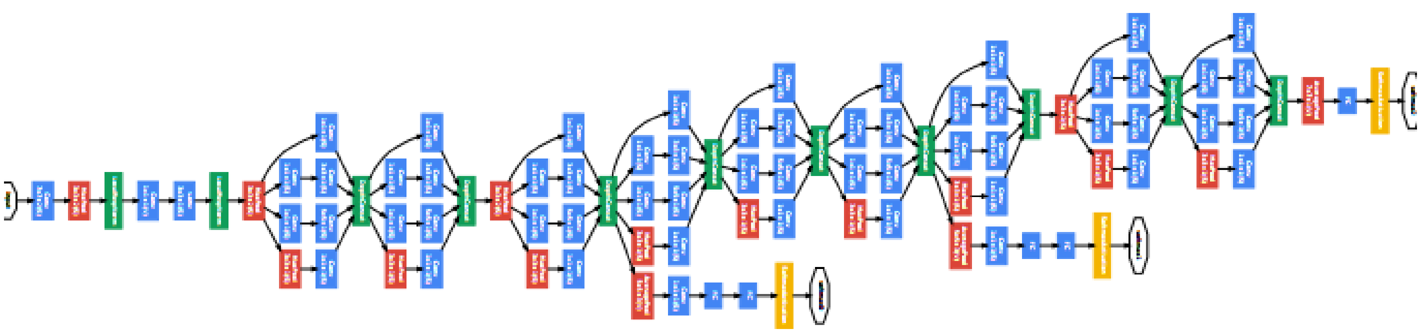
Convolutional Nets



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

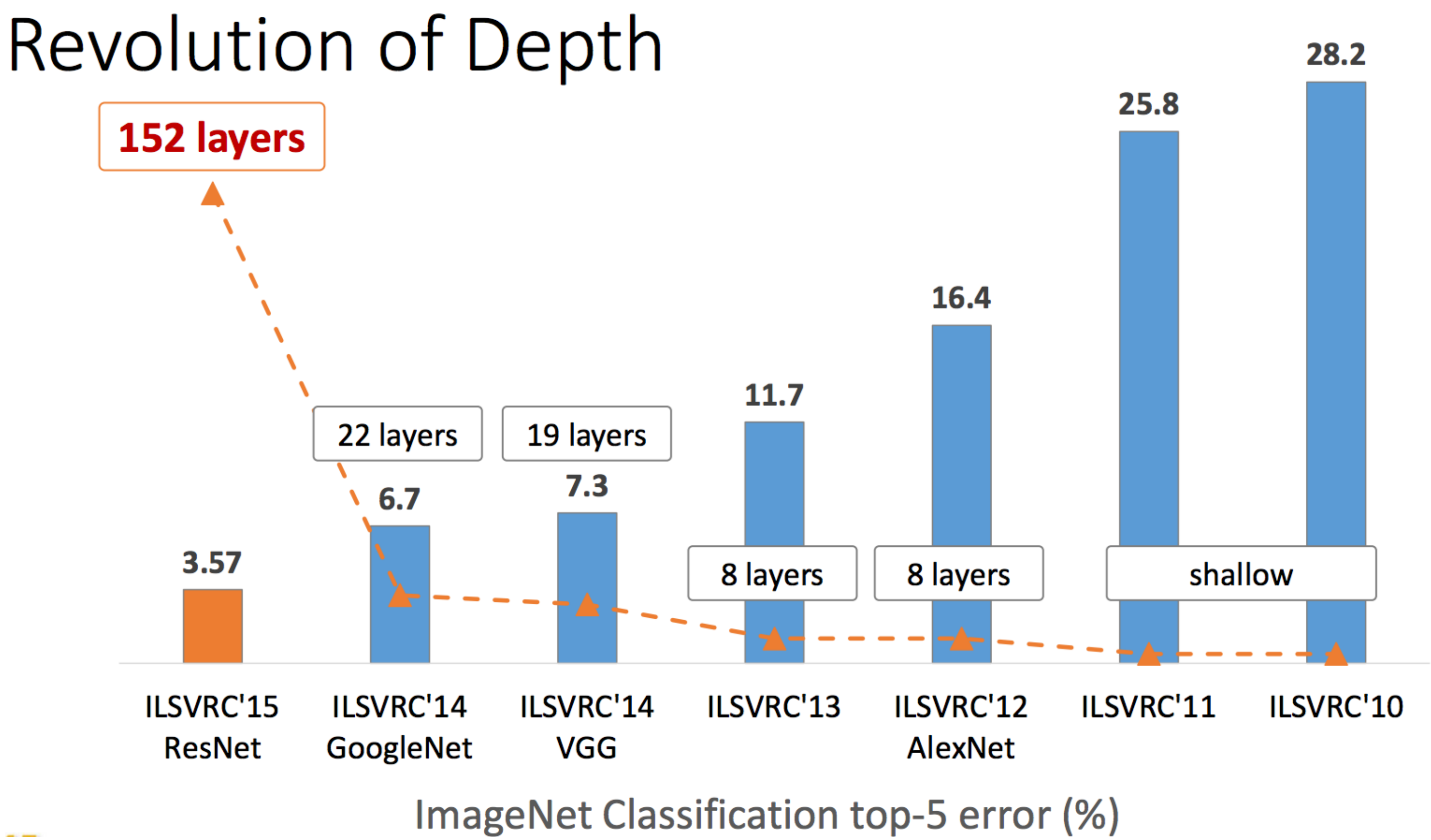
ConvNet roots

- **Fukushima, 1980s** designed network with same basic structure but did not train by backpropagation.
- The first successful applications of **Convolutional Networks** by Yann LeCun in 1990's (LeNet)
  - Was used to read zip codes, digits, etc.
- Many variants nowadays, but the core idea is the same
  - Example: a system developed in Google (GoogLeNet)
    - Compute different filters
    - Compose one big vector from all of them
    - Layer this iteratively



See more: http://arxiv.org/pdf/1409.4842v1.pdf

Depth matters



Slide from [Kaiming He 2015]

Practical Tips

- Before large scale experiments, test on a small subset of the data and check the error should go to zero.
  - Overfitting on small training
- Visualize features (feature maps need to be uncorrelated) and have high variance
- Bad training: many hidden units ignore the input and/or exhibit strong correlations.

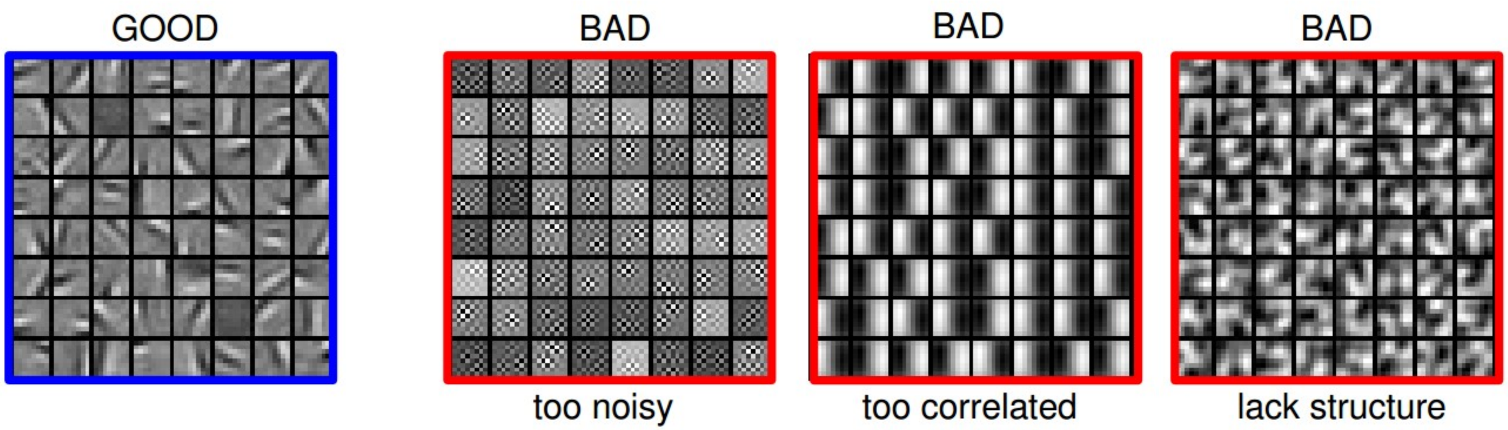


Figure Credit: Marc'Aurelio Ranzato

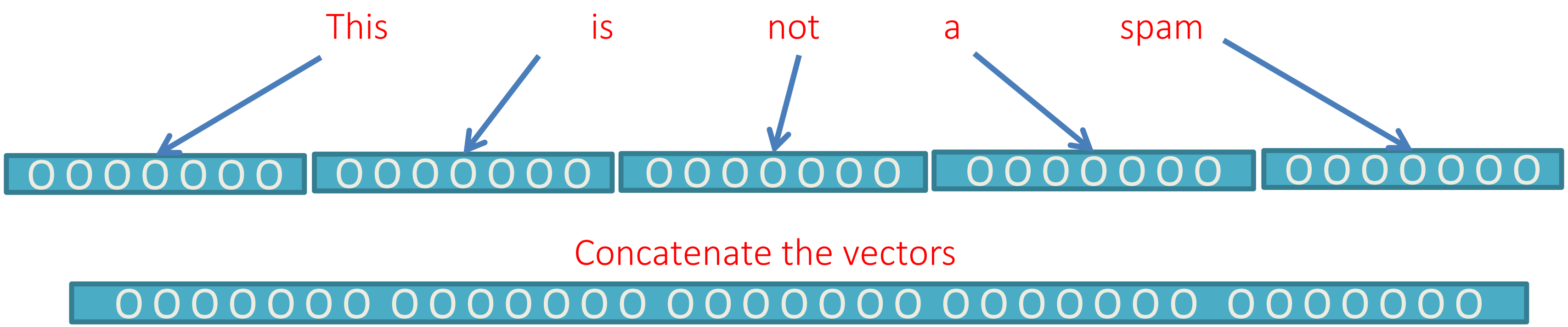
Debugging

- Training diverges:
  - Learning rate may be too large → decrease learning rate
  - BackProp is buggy → numerical gradient checking
- Loss is minimized but accuracy is low
  - Check loss function: Is it appropriate for the task you want to solve? Does it have degenerate solutions?
- NN is underperforming / under-fitting
  - Compute number of parameters → if too small, make network larger
- NN is too slow
  - Compute number of parameters → Use distributed framework, use GPU, make network smaller

Many of these points apply to many machine learning models, no just neural networks.

CNN for text (sequence) inputs

- Let’s study another variant of CNN for language
  - Example: sentence classification (say spam or not spam)
- First step: represent each word with a vector in $\mathbb{R}^d$

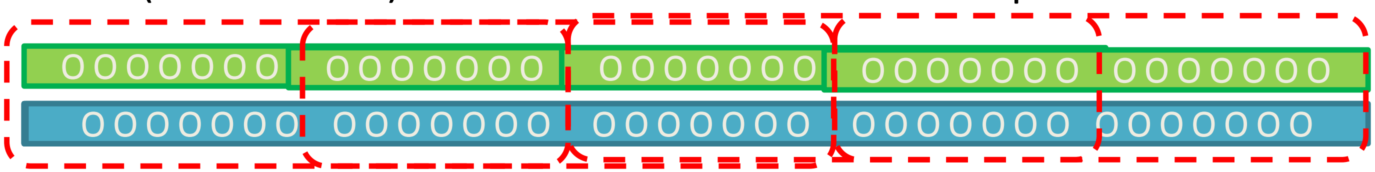


- Now we can assume that the input to the system is a vector $\mathbb{R}^{dl}$
  - Where the input sentence has length l ($l = 5$ in our example)
  - Each word vector’s length d ($d = 7$ in our example)

Convolutional Layer on vectors

- Think about a single convolutional layer
  - A bunch of **vector** filters
    - Each defined in $\mathbb{R}^{dh}$
      - Where h is the number of the words the filter covers
      - Size of the word vector d

  - Find its (modified) convolution with the input vector



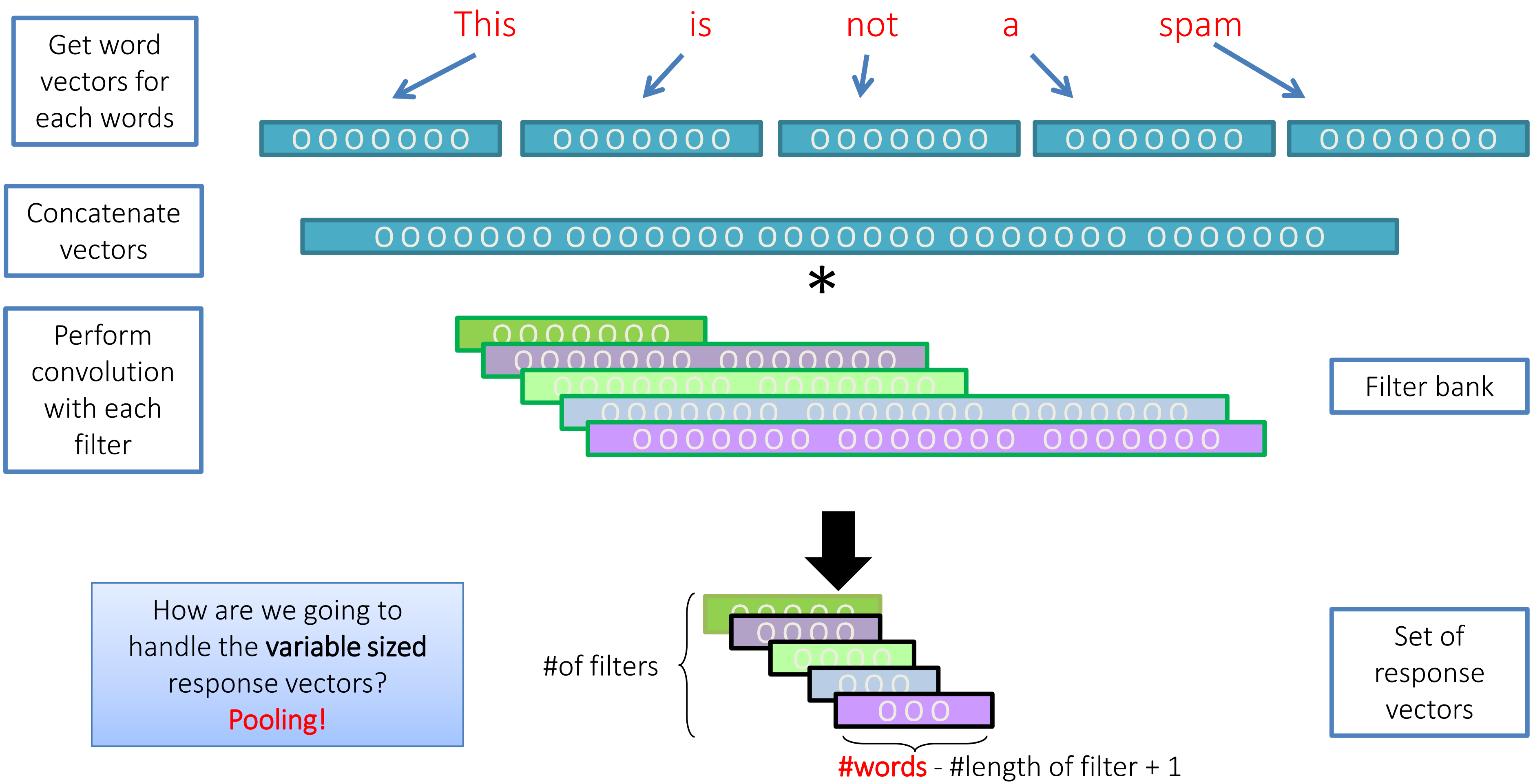
$$c_1 = f(w.x_{1:h}) \quad c_2 = f(w.x_{h+1:2h}) \quad c_3 = f(w.x_{2h+1:3h}) \quad c_4 = f(w.x_{3h+1:4h})$$

  - Result of the convolution with the filter

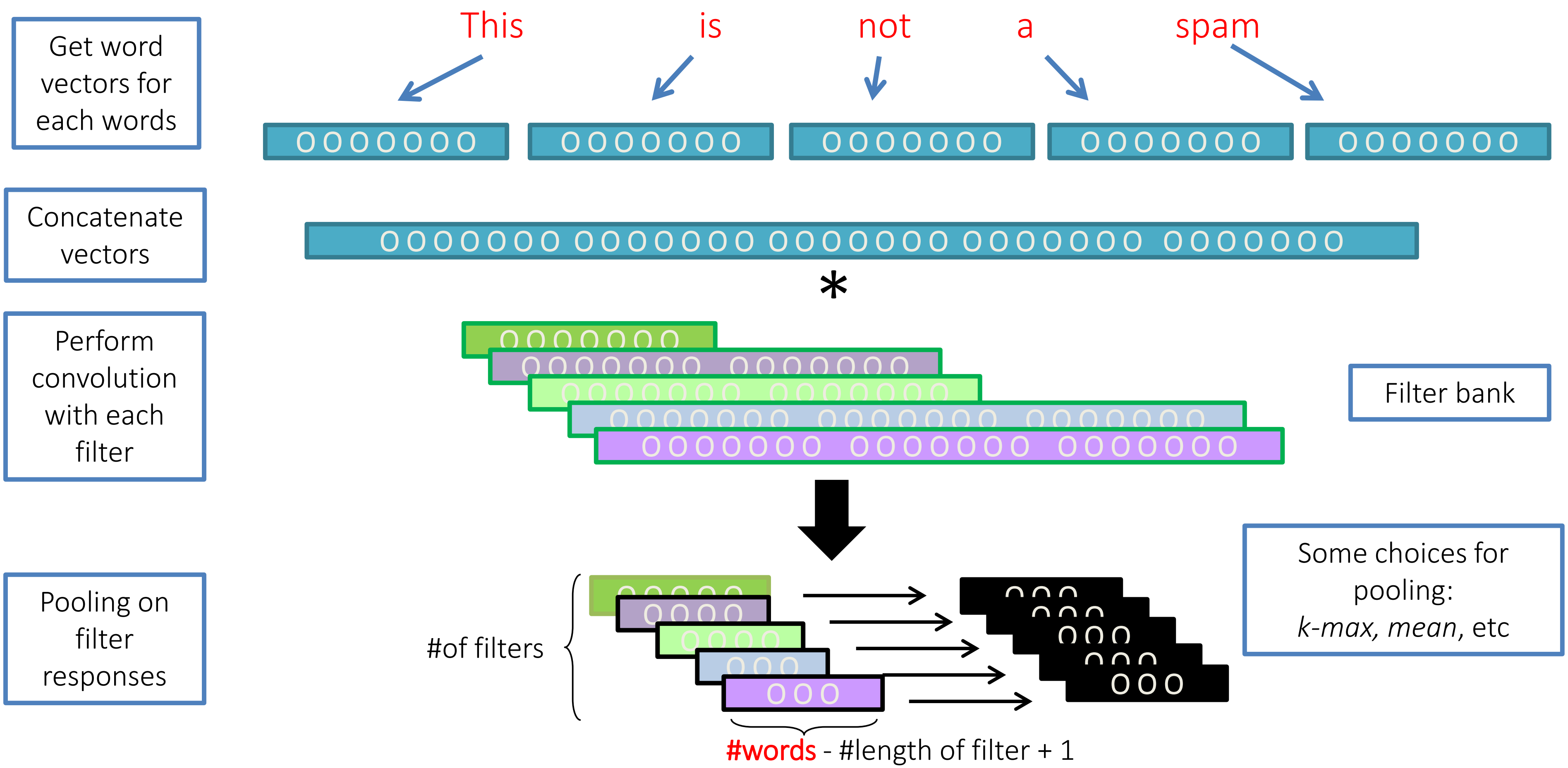
$$c = [c_1, \dots, c_{n-h+1}]$$

  - Convolution with a filter that spans 2 words, is operating on all of the bi-grams (vectors of two consecutive word, concatenated): “this is”, “is not”, “not a”, “a spam”.
  - Regardless of whether it is grammatical (not appealing linguistically)

Convolutional Layer on vectors



Convolutional Layer on vectors



- Now we can pass the fixed-sized vector to a logistic unit (softmax), or give it to multi-layer network (last session)