Crash Course on Deep Learning

Vincent Lepetit

université de BORDEAUX

Class Overview

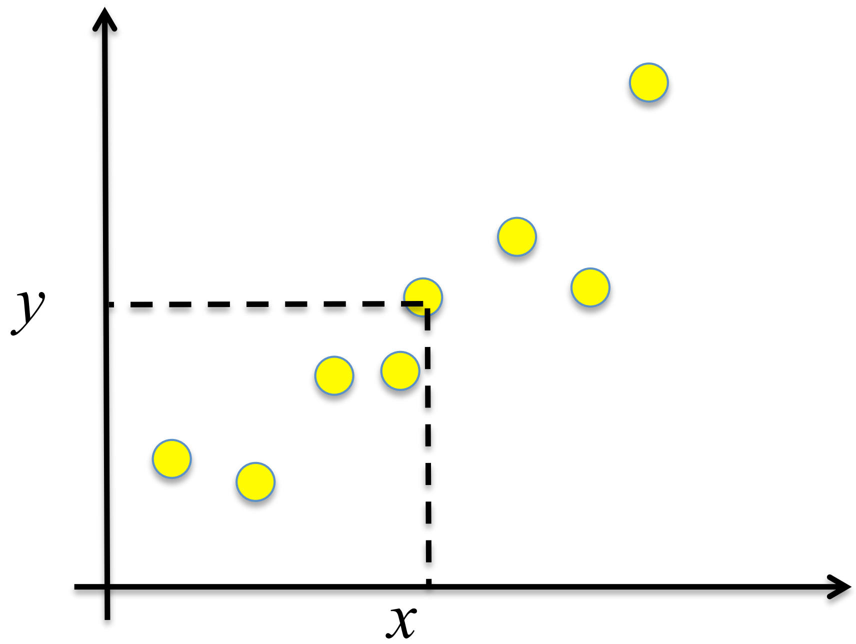
1. Crash course on Deep Learning;
2. Computer Vision Pipelines and Deep Learning;
3. 3D Localization with Deep Learning;
4. Crash course on Tensor Flow.

Supervised Learning = Regression
(with (sometimes) some specific properties)

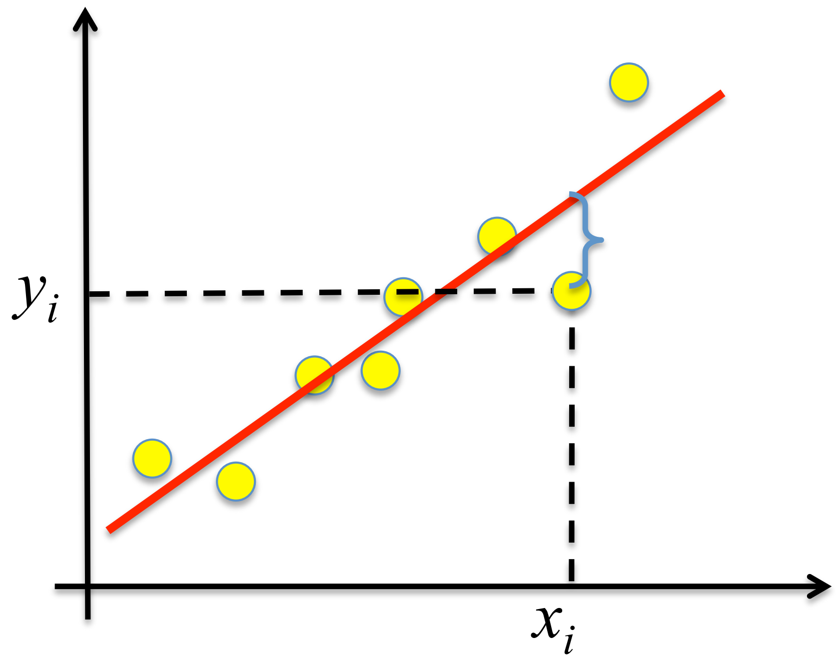
université de BORDEAUX

Regression

$$y = F(x) \quad F?$$



Linear Regression



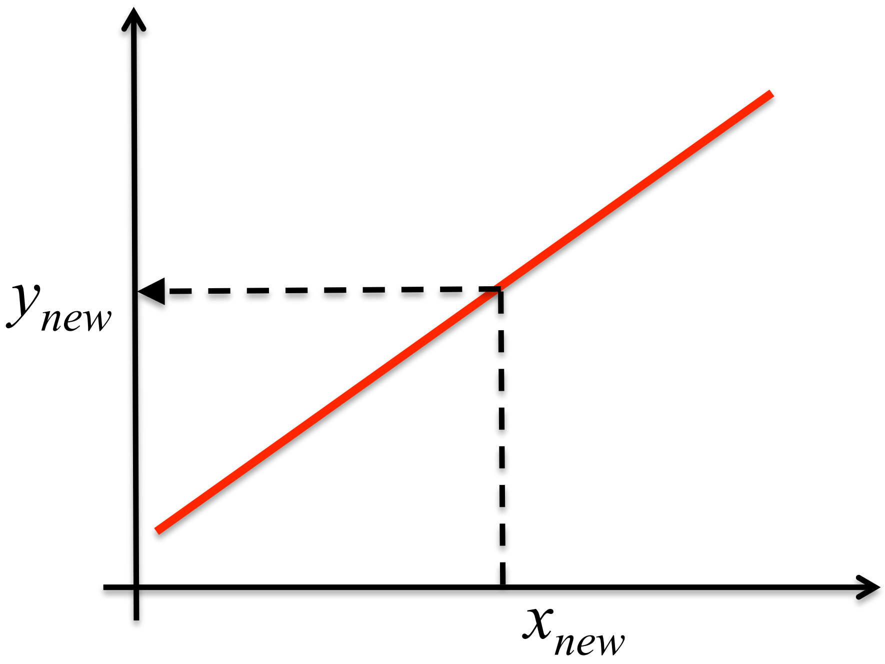
Model: $y = F(x) = ax + b$

Model parameters: a and b estimated by optimization, for example:

$$\boxed{\arg\min_{a,b} \sum_i (ax_i + b - y_i)^2}$$

this is called "learning" or "training"

Linear Regression and Prediction



Model: $y = ax + b$

Now known model parameters: a and b

$$\boxed{y_{\text{new}} = a x_{\text{new}} + b}$$

this is called "prediction"

université de BORDEAUX

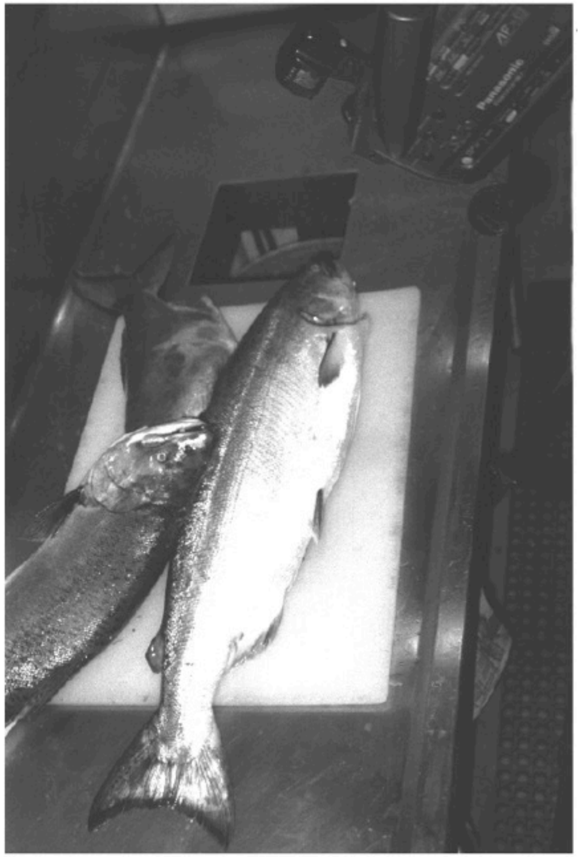
Training Set & Test Set

- Training set: $\{(x_i, y_i)\}_i$
known labeled data used to optimize the model's parameters by minimizing the loss function;

- Test set: $\{(x_i, y_i)\}_i$
known labeled data used to optimize the model's parameters (and the method in general).
The real goal is to have "good" predictions on the test set.

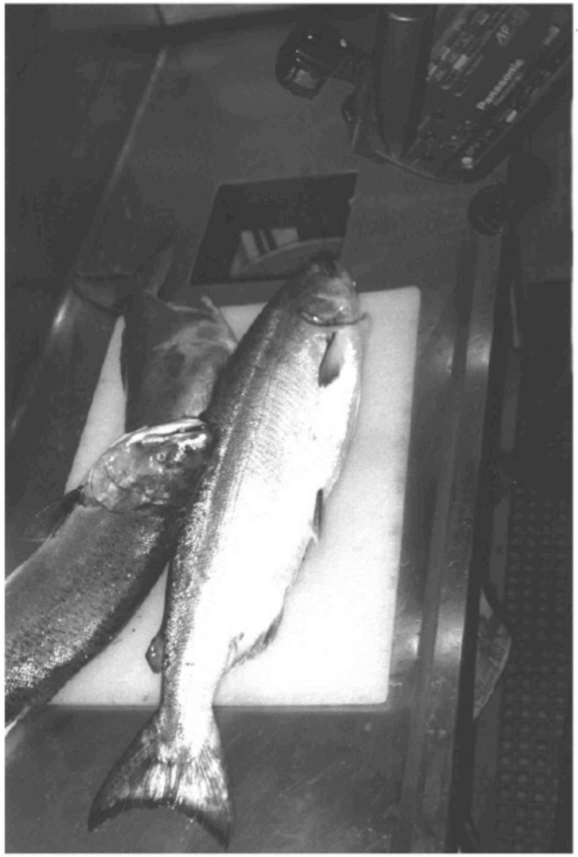
université de BORDEAUX

Example: Binary Classification



SALMON or SEA BASS?

Binary Classification

$$F\Bigg($$  $$\Bigg) = +1$$ for SALMON,

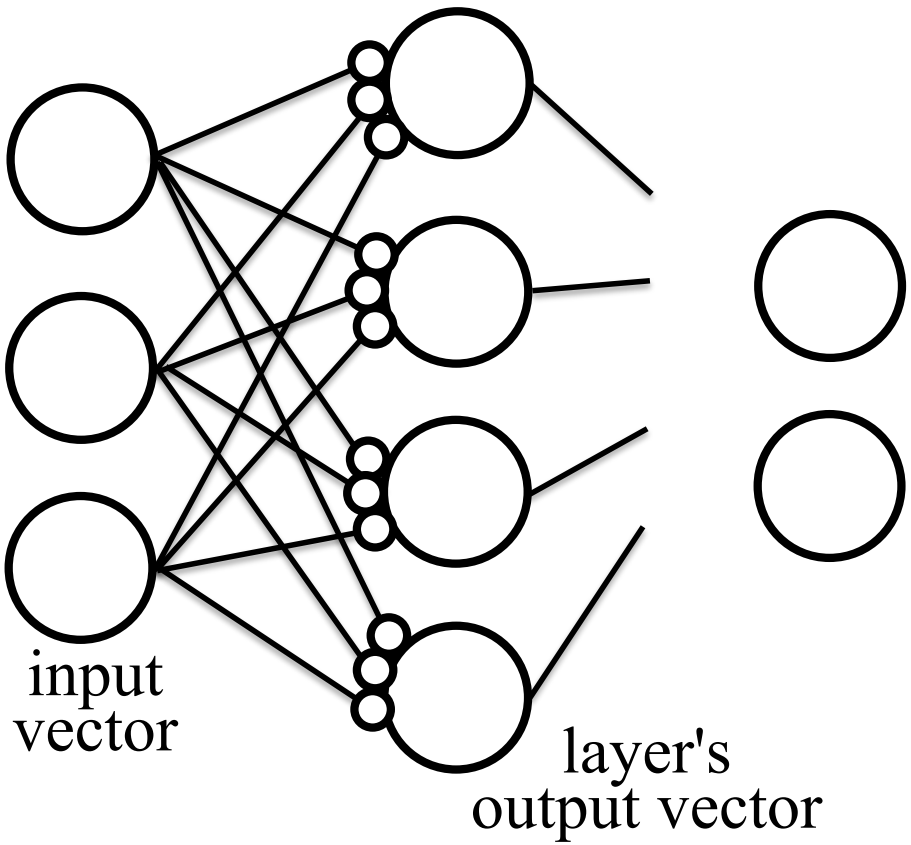
-1 for SEA BASS

Deep Networks

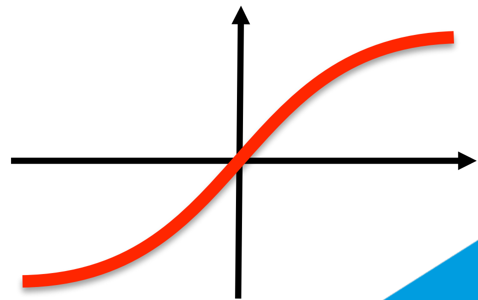
- are a possible, general, and now very popular form for $F(.)$;

- are also a general tool for other regression problems.

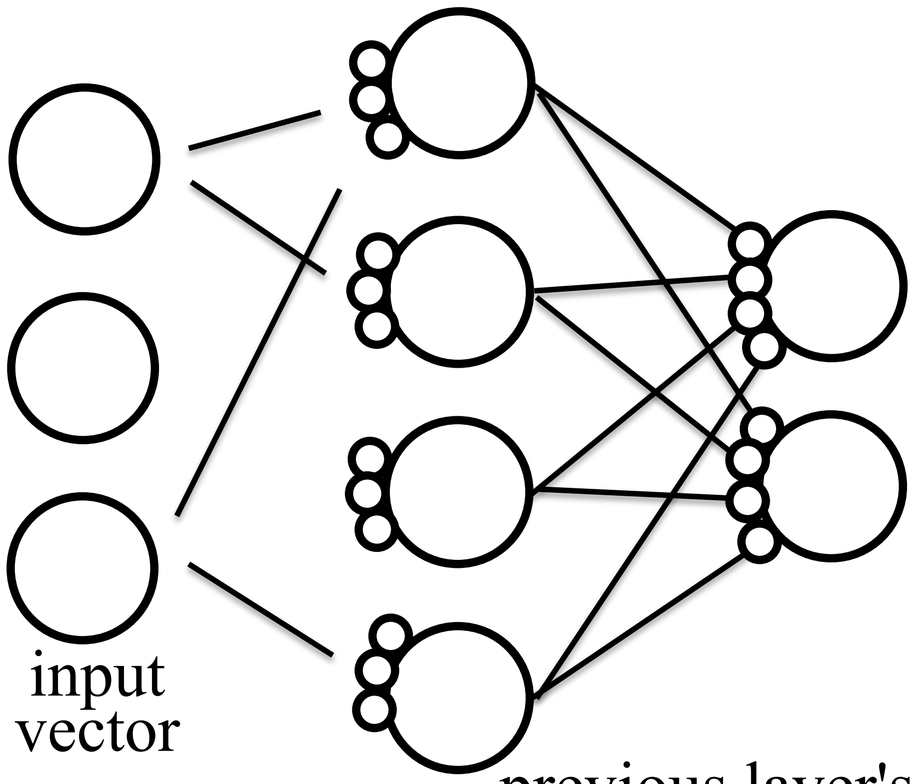
A Standard Neural Network



layer's output vector =
sigmoid(W . input vector + bias)

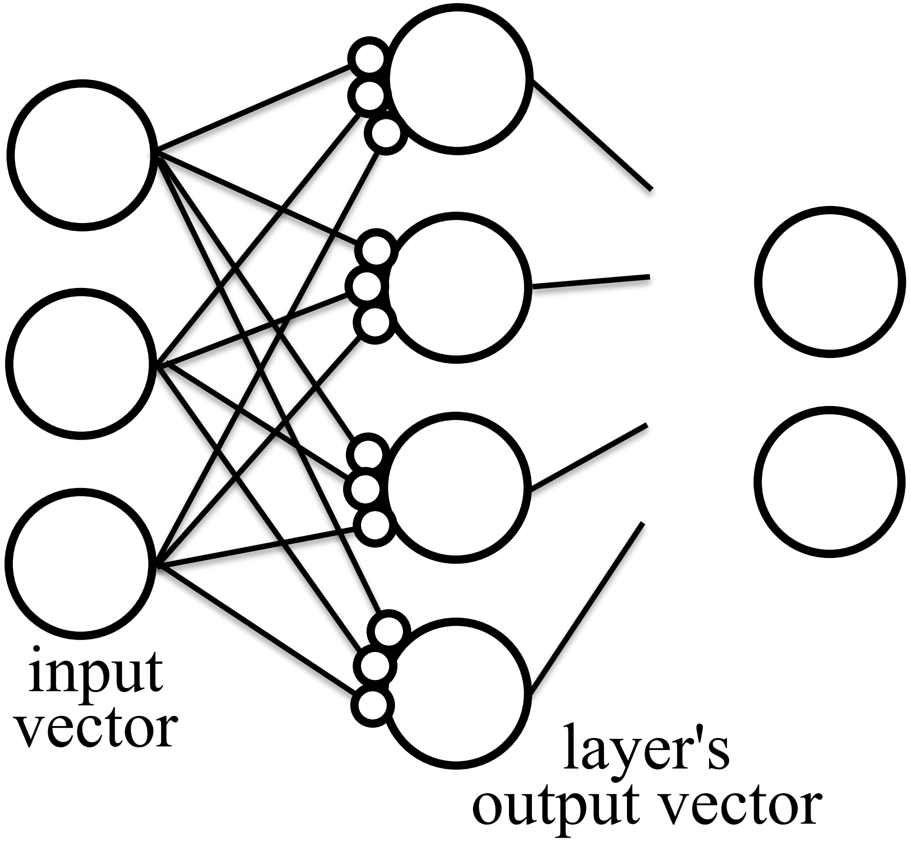


A Standard Neural Network

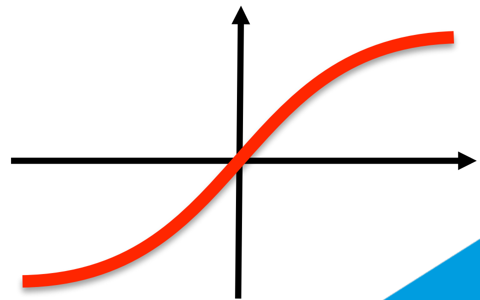


final output vector =
W_2 . (previous layer's output vector) + bias_2

A Standard Neural Network

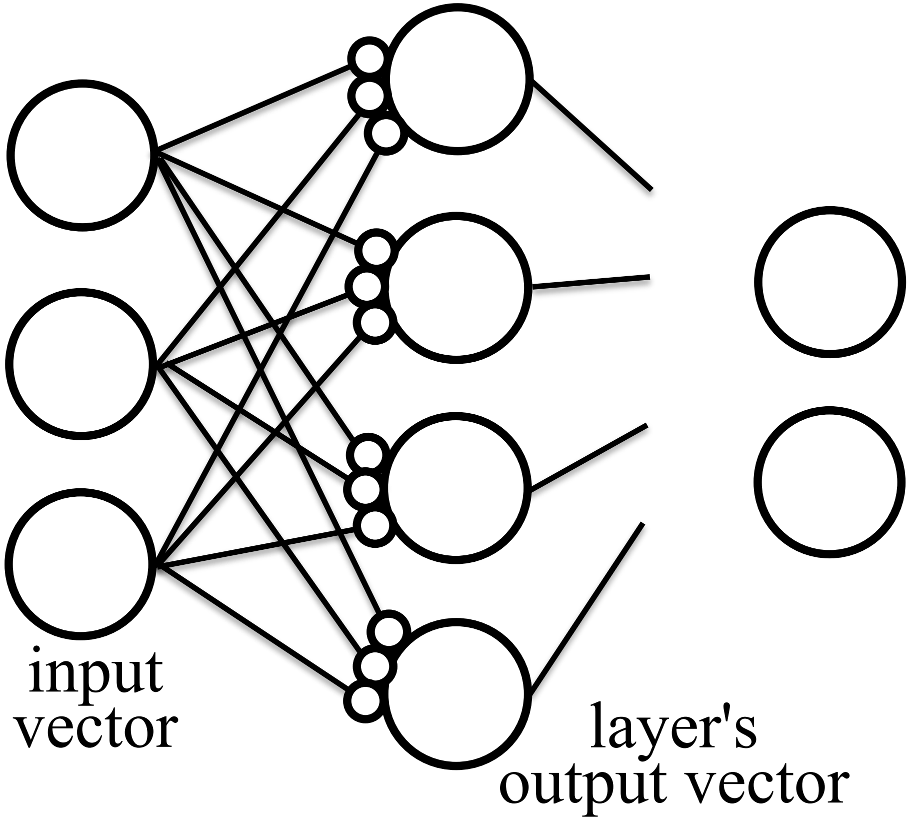


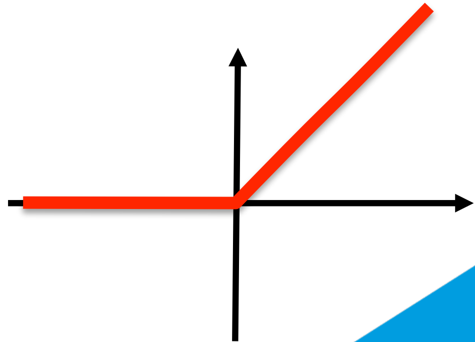
layer's output vector =
sigmoid(W . input vector + bias)



université de BORDEAUX

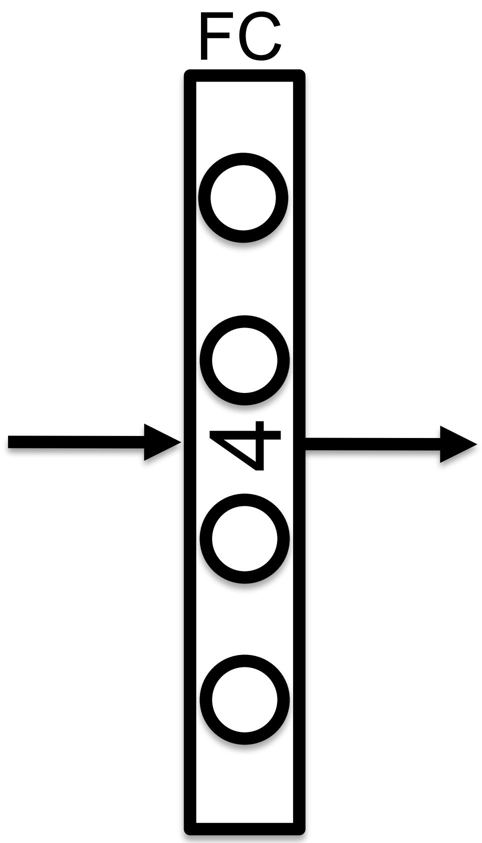
The ReLU Operator

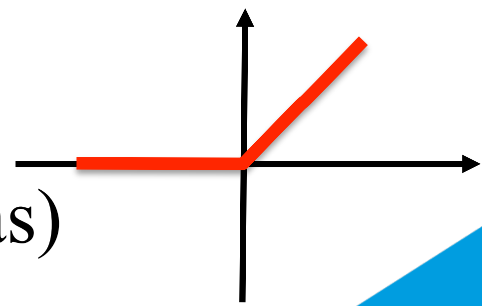




layer's output vector =
ReLU(W . input vector + bias)

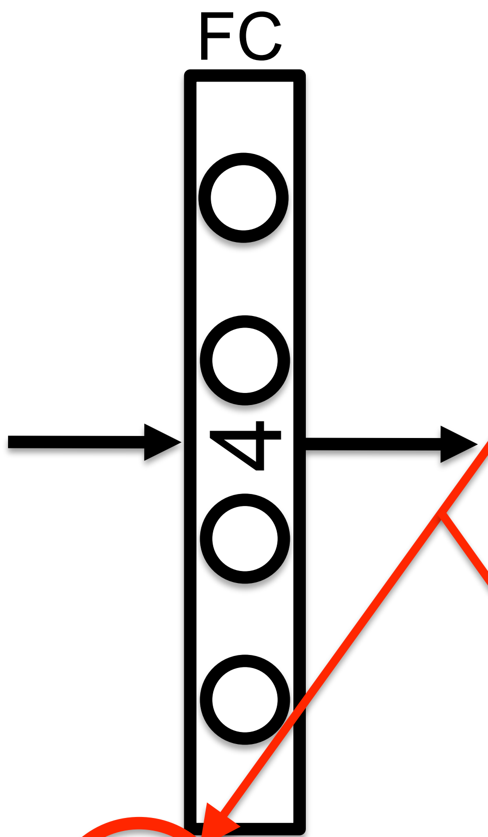
Fully Connected Layer





output vector = ReLU(W . input vector + bias)

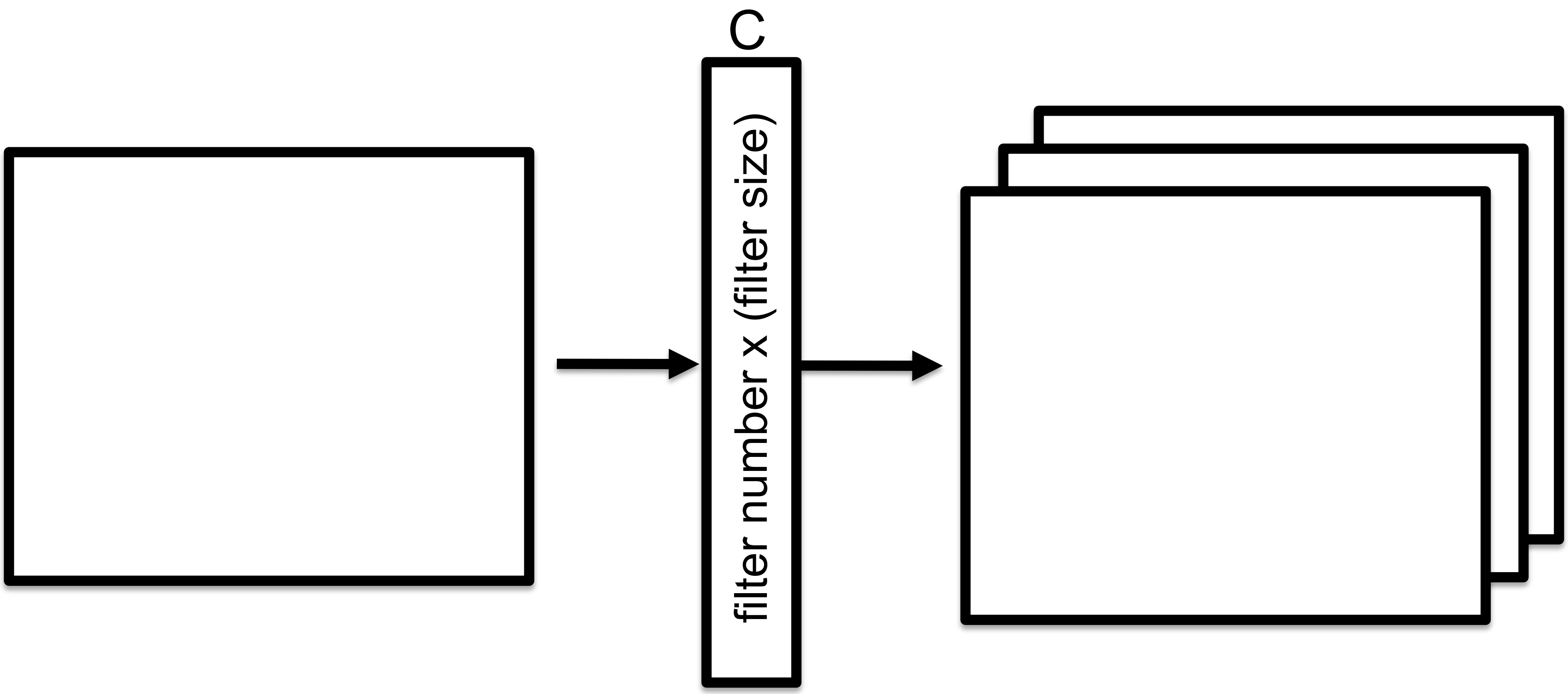
Fully Connected Layer



Parameters for this layer,

output vector = ReLU(W . input vector + bias)

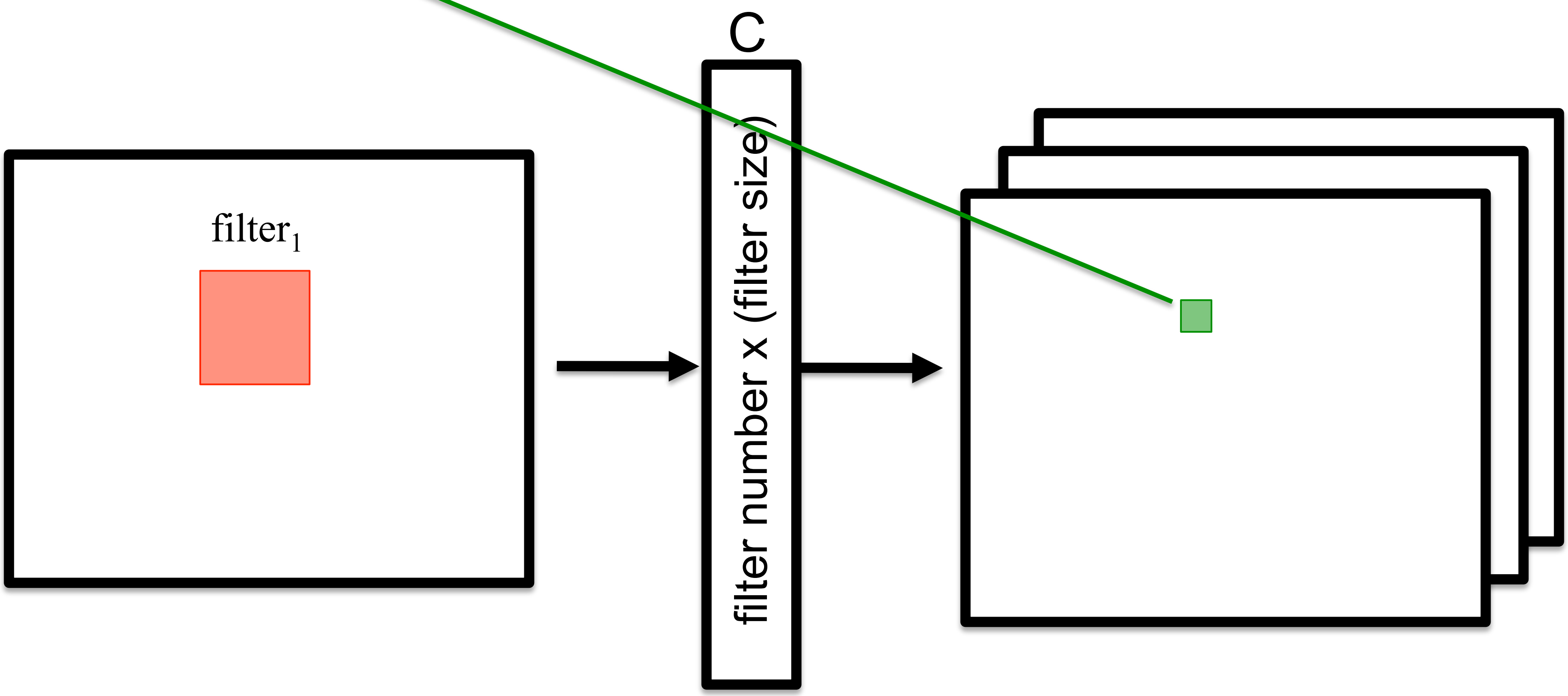
Convolutional Layer



$$\text{output images} = \text{ReLU}(\text{filter}_1 * \text{input image}, \text{filter}_2 * \text{input image}, \ldots)$$

université de BORDEAUX

$$\text{output image}[u,v] = \sum_{i=-m}^{+m} \sum_{j=-n}^{n} \text{filter}_1[i,j]\,\text{input image}[u+i, v+j]$$

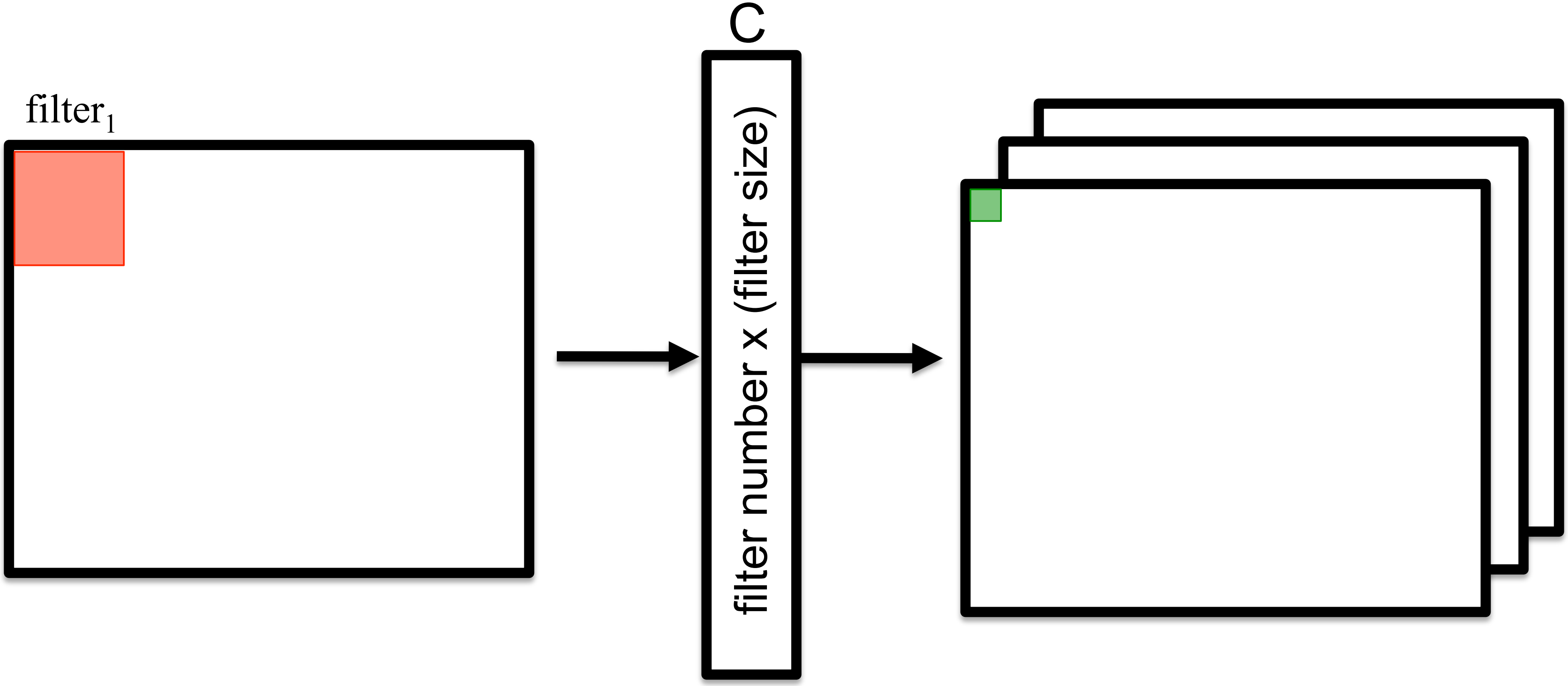


output images = ReLU(filter$_1$ * input image,
filter$_2$ * input image, …)

université de BORDEAUX

Convolutional Layer

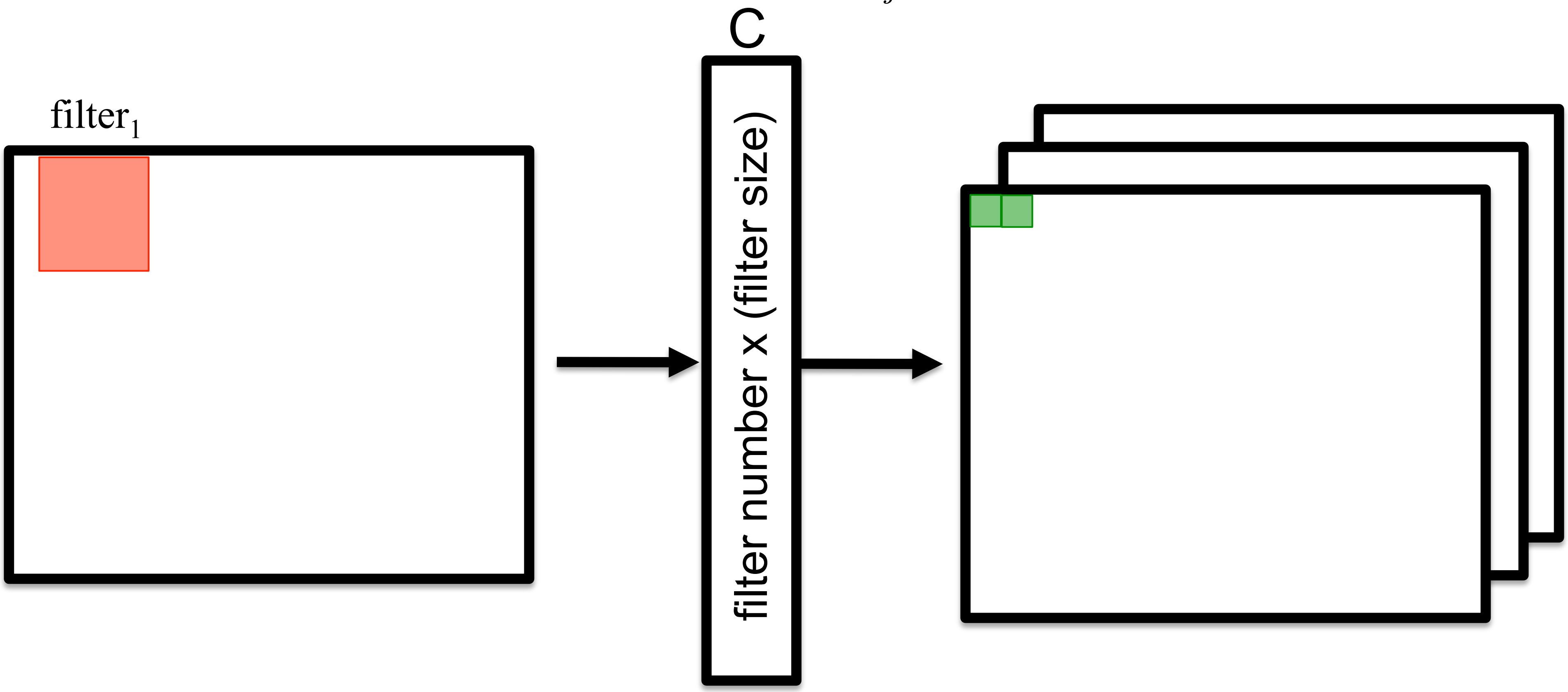
$$\sum_{i=-m}^{+m} \sum_{j=-n}^{n} \mathrm{filter}_1[i,j]\mathrm{input\ image}[u+i, v+j]$$



output images = ReLU(filter$_1$ * input image,
filter$_2$ * input image, …)

Convolutional Layer

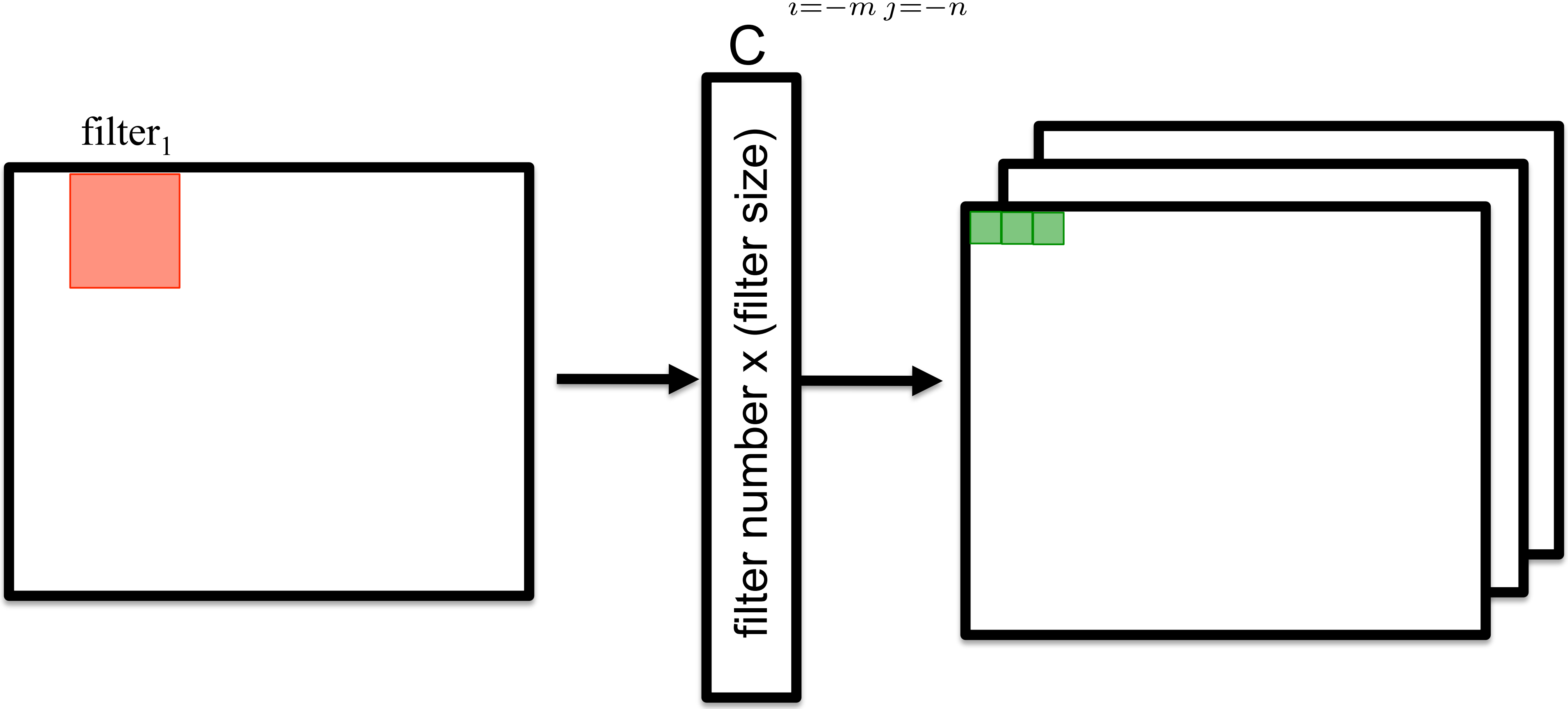
$$\sum_{i=-m}^{+m} \sum_{j=-n}^{n} \mathrm{filter}_1[i,j]\mathrm{input\ image}[u+i, v+j]$$



$$\text{output images} = \mathrm{ReLU}(\text{filter}_1 * \text{input image}, \text{filter}_2 * \text{input image}, \ldots)$$

Convolutional Layer

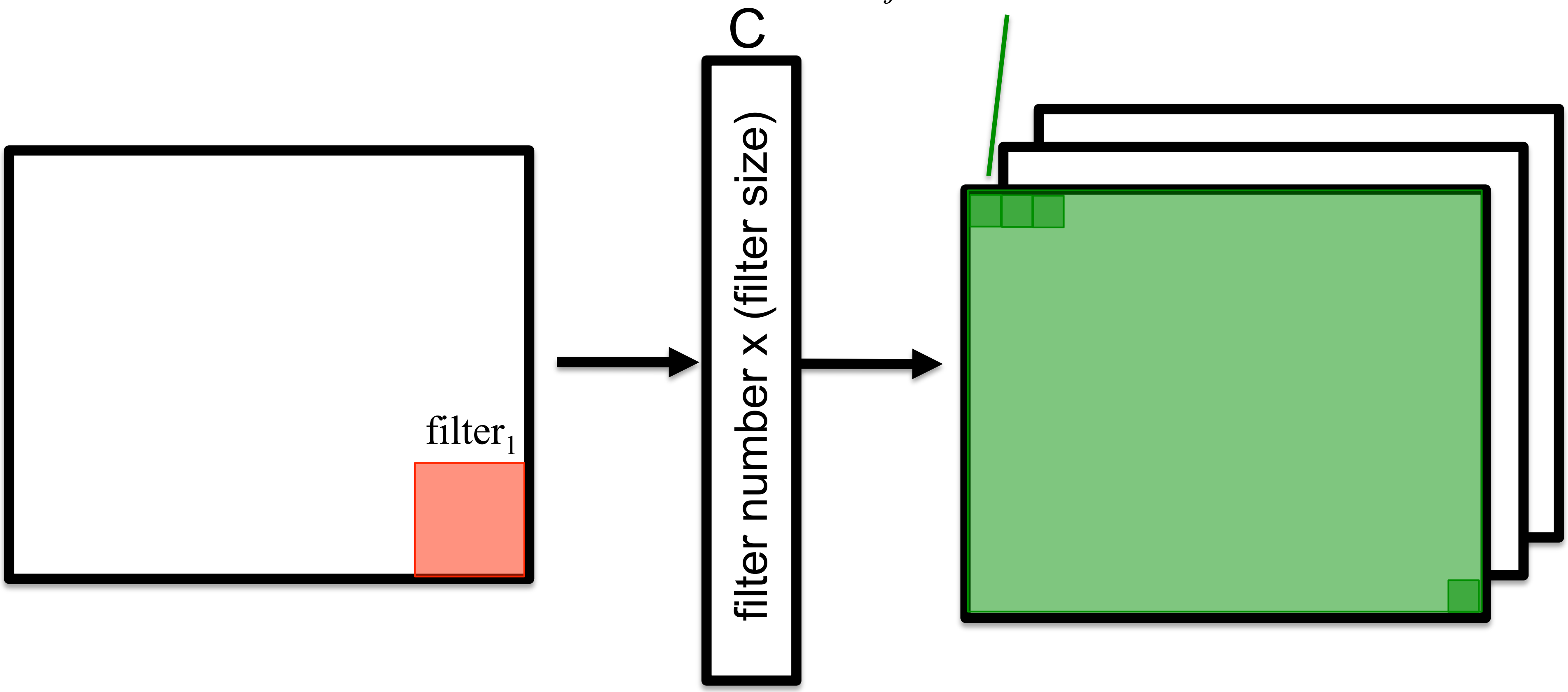
$$\sum_{i=-m}^{+m} \sum_{j=-n}^{n} \text{filter}_1[i,j]\text{input image}[u+i, v+j]$$



output images = ReLU(filter$_1$ * input image,
filter$_2$ * input image, …)

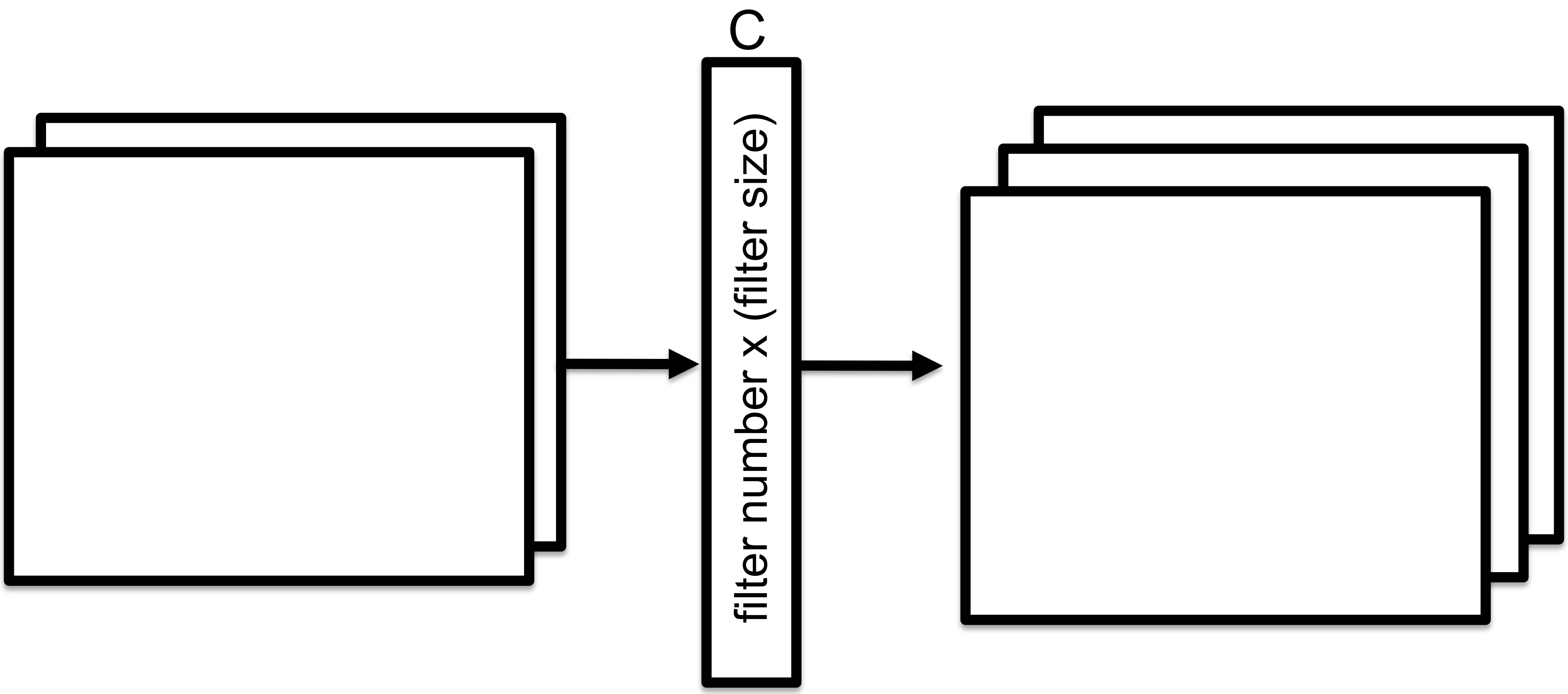
Convolutional Layer

$$\sum_{i=-m}^{+m} \sum_{j=-n}^{n} \text{filter}_1[i,j]\text{input image}[u+i, v+j]$$



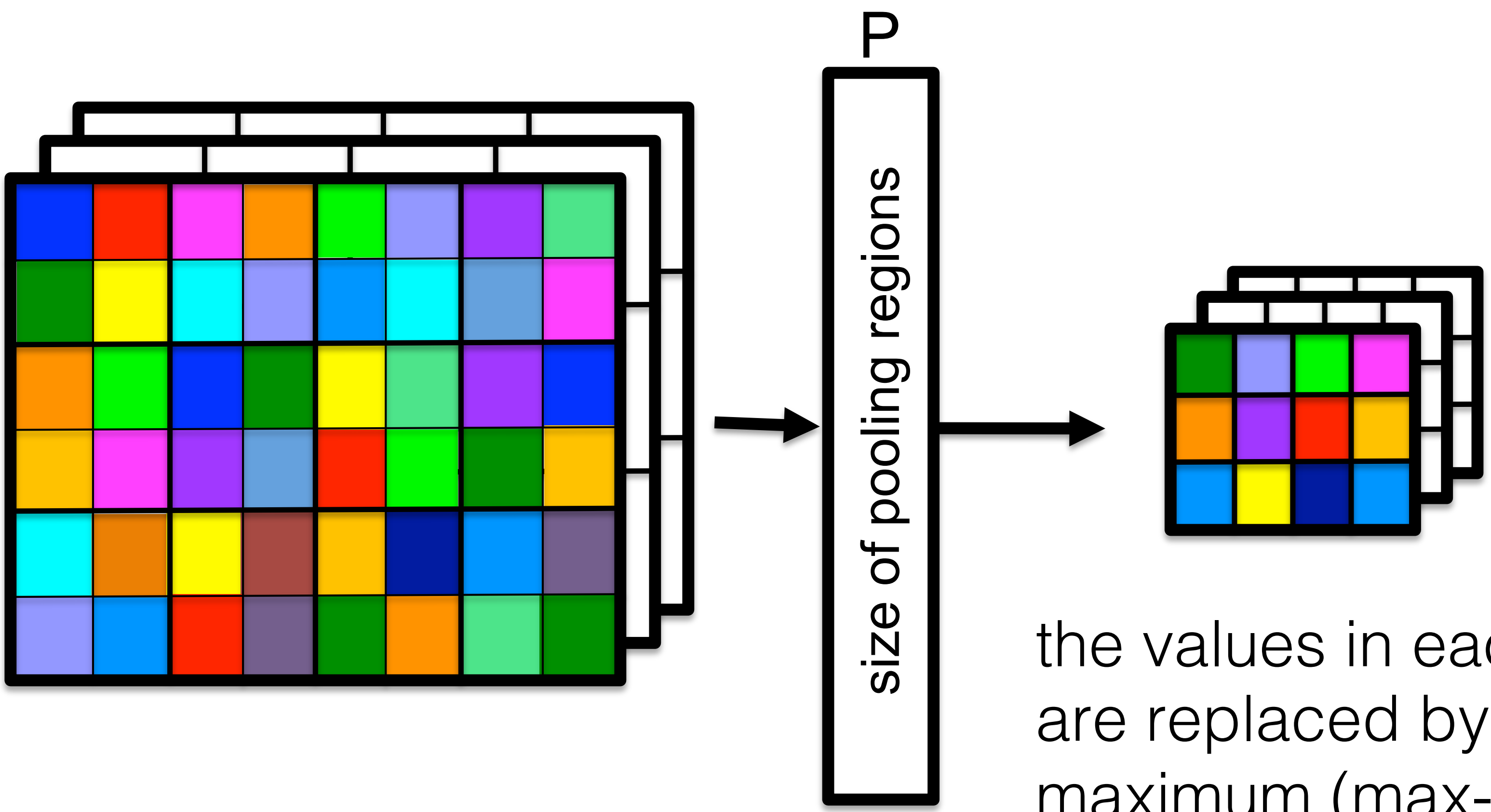
output images = ReLU(filter$_1$ * input image,
filter$_2$ * input image, …)

Convolutional Layer



output images = ReLU(filter_1 * input images,
filter_2 * input images, …)

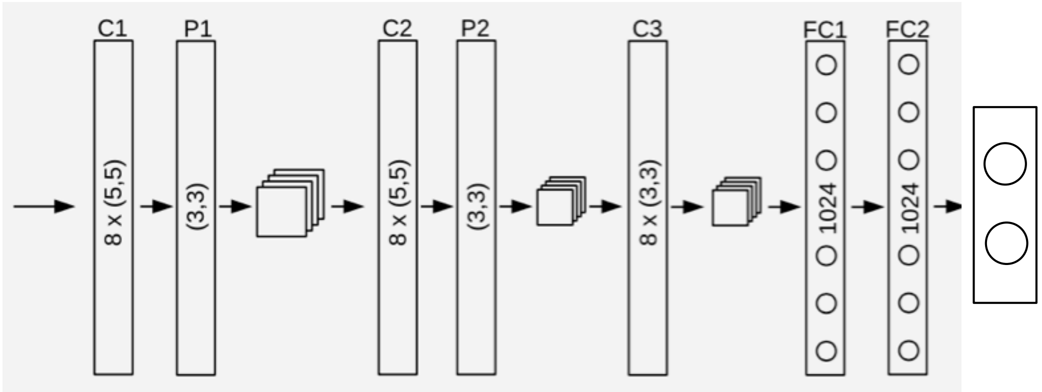
Pooling Layer



the values in each region are replaced by their maximum (max-pooling)

A Complete Deep Network

Input: An image

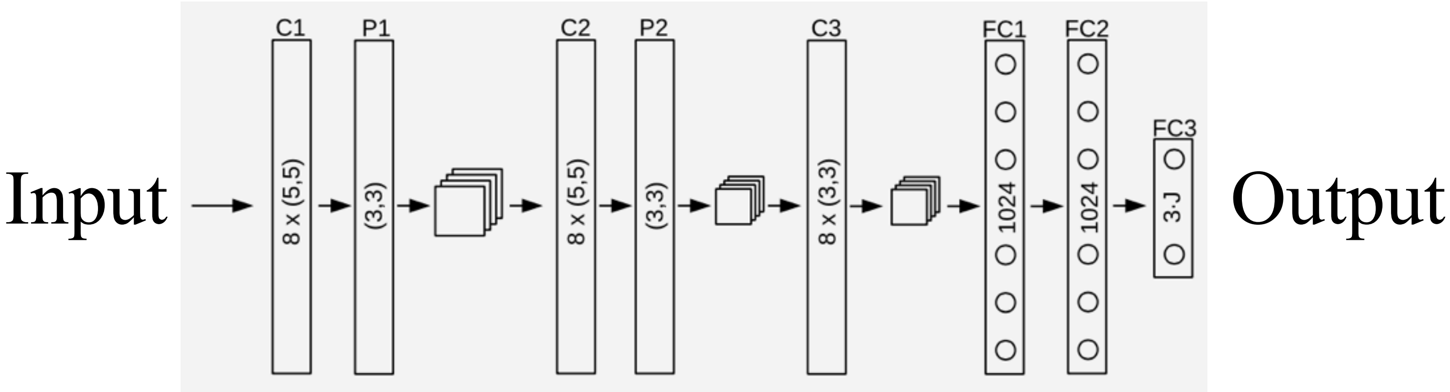


"car"

$$\mathbf{h}_1 = [\mathrm{ReLU}(f_{1,1} * \mathbf{x}), \ldots, \mathrm{ReLU}(f_{1,n} * \mathbf{x})]$$
$$\mathbf{h}_2 = [\mathrm{pool}(\mathbf{h}_{1,1}), \ldots, \mathrm{pool}(\mathbf{h}_{1,n})]$$
$$\mathbf{h}_3 = [\mathrm{ReLU}(f_{3,1} * \mathbf{h}_{2,1}), \ldots, \mathrm{ReLU}(f_{3,n} * \mathbf{h}_{2,n})]$$
$$\mathbf{h}_4 = [\mathrm{pool}(\mathbf{h}_{3,1}), \ldots, \mathrm{pool}(\mathbf{h}_{3,n})]$$
$$\mathbf{h}_5 = \mathrm{ReLU}(\mathbf{W}_5\mathbf{h}_4 + \mathbf{b}_5)$$
$$\mathbf{h}_6 = \mathrm{ReLU}(\mathbf{W}_6\mathbf{h}_5 + \mathbf{b}_6)$$
$$\mathbf{h}_7 = \mathbf{W}_7\mathbf{h}_6 + \mathbf{b}_7$$
$$p(c = \mathrm{car}|I) = \frac{\exp(\mathbf{h}_7[0])}{\exp(\mathbf{h}_7[0])+\exp(\mathbf{h}_7[1])}$$
$$p(c = \mathrm{no\ car}|I) = \frac{\exp(\mathbf{h}_7[1])}{\exp(\mathbf{h}_7[0])+\exp(\mathbf{h}_7[1])}$$

université de BORDEAUX

Optimization



Network parameters found by optimizing an objective function, for example:

$$\min_{\substack{\text{matrices of the fully connected layers}\\ \text{filters of the convolutional layers}}} \sum_{\substack{(\text{input}_i,\ \text{expected output}_i)\ \text{in}\\ \text{training set}}} f(\text{CNN}(\text{input}_i), \text{expected output}_i)$$

Optimization with stochastic gradient descent on mini-batches + dropout + hard example mining + …

Two Possible Loss Functions

For multi-class classification, for example using cross-entropy:

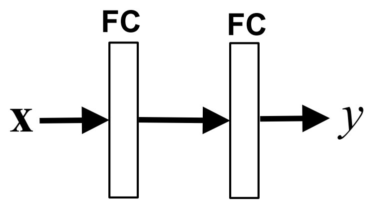
$$\min_{\theta} - \sum_{i} \mathbf{y}_i \log(\mathrm{softmax}_{\theta}(\mathrm{CNN}(\mathbf{x}_i)))$$

For least-squares regression:

$$\min_{\theta} \sum_{i} \|\mathbf{y}_i - \mathrm{CNN}_{\theta}(\mathbf{x}_i)\|_2^2$$

université de BORDEAUX

My Take on What Deep Networks Do



Consider a 2-layer network of the form:

$$y = \mathbf{w}_2 \, \sigma(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) \tag{1}$$

Introduce the matrix

$$\mathbf{B}(\mathbf{x}) = \mathrm{diag}(\ldots, \mathrm{sgn}(\mathbf{W}_1^{(i)} \mathbf{x} + \mathbf{b}_1^{(i)}), \ldots)$$

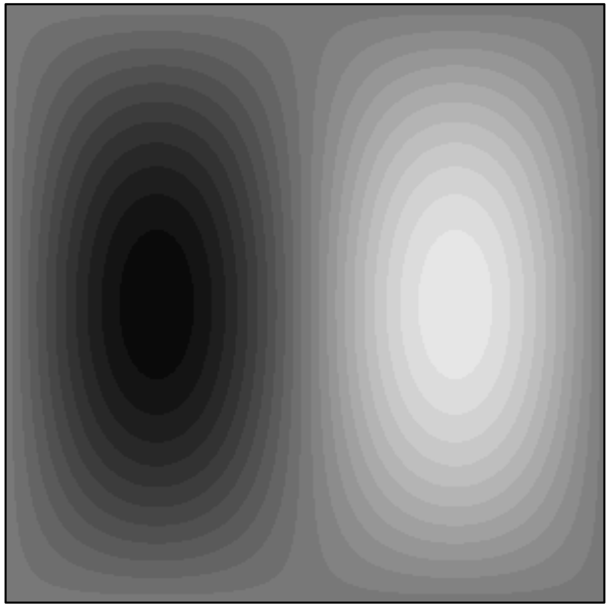
(1) can be rewritten:

$$y = \mathbf{w}_2 \, \mathbf{B}(\mathbf{x}) \, (\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1)$$

The function $\mathbf{x} \mapsto \mathbf{B}(\mathbf{x})$ is piecewise constant.

Thus (1) is piecewise affine (and also continuous).

2D Example: Visualizing $F(x)$

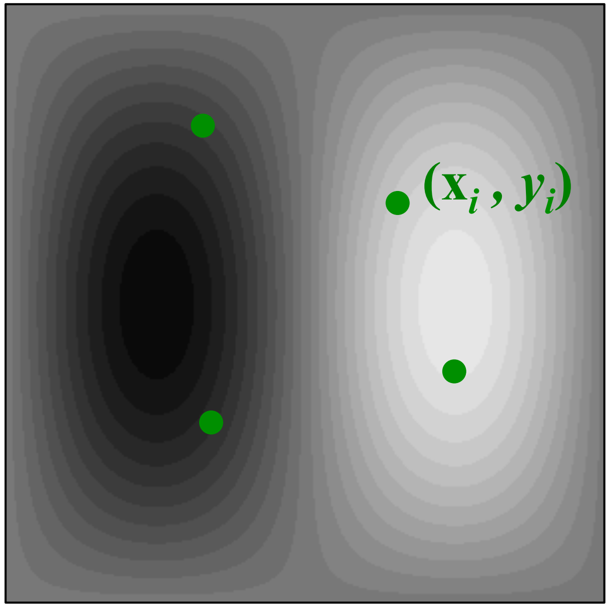


Target Function

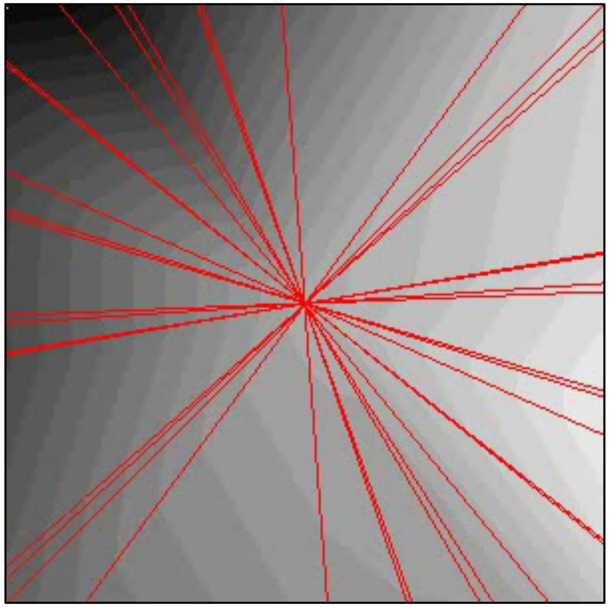
To be approximated with:

$$y = \mathbf{w}_2 \sigma(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1)$$

2D Example: Visualizing $F(x)$



Target Function



$$\min_{\mathbf{W}_1, \mathbf{b}_1, \mathbf{w}_2} \sum_i \|\mathbf{w}_2 \sigma(\mathbf{W}_1 \mathbf{x}_i + \mathbf{b}_1) - y_i\|^2$$

To be approximated with:

$$y = \mathbf{w}_2 \sigma(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1)$$

$$\mathbf{x} = \begin{bmatrix} u \\ v \end{bmatrix}$$