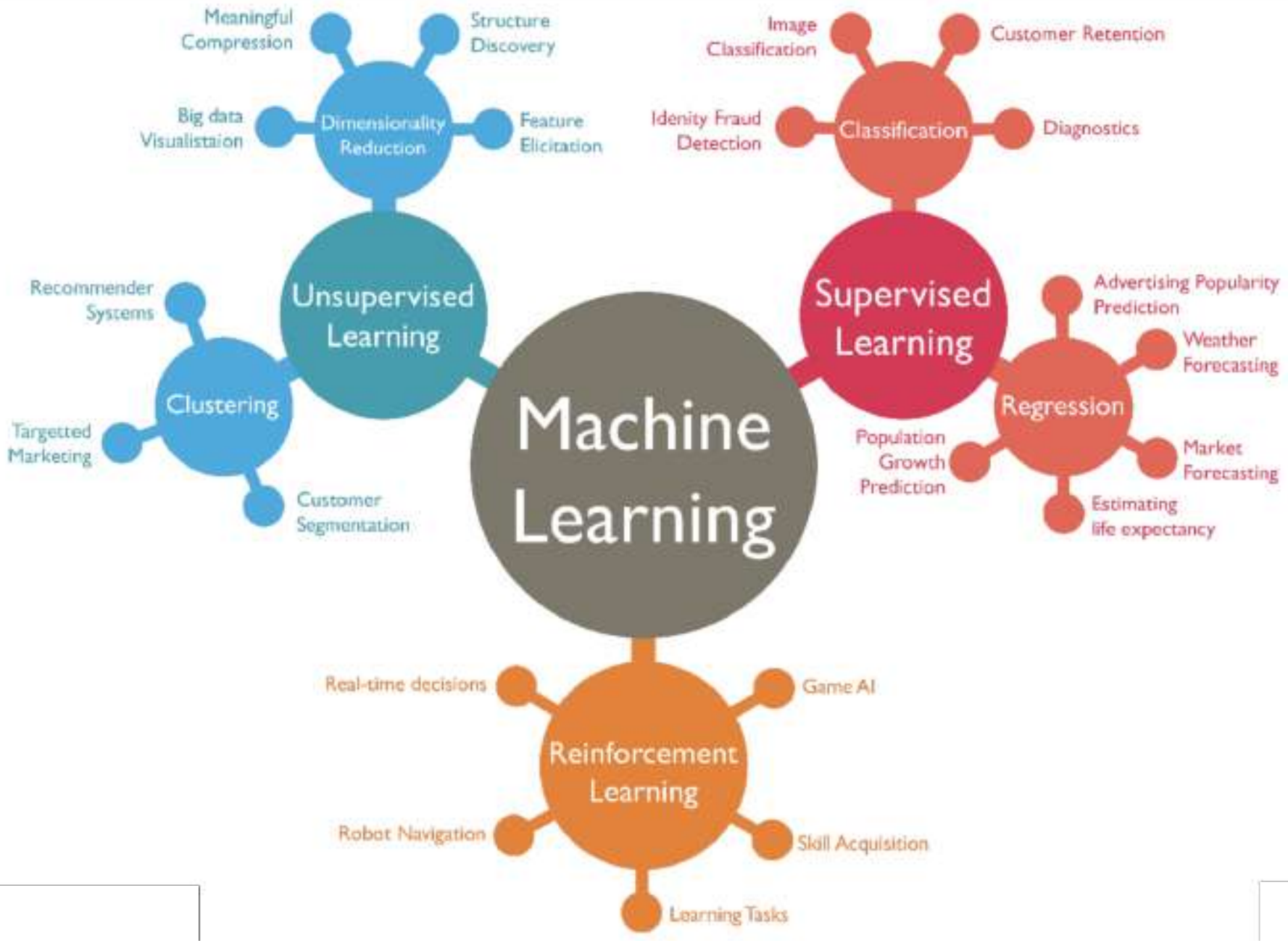
Foundations of modern NN

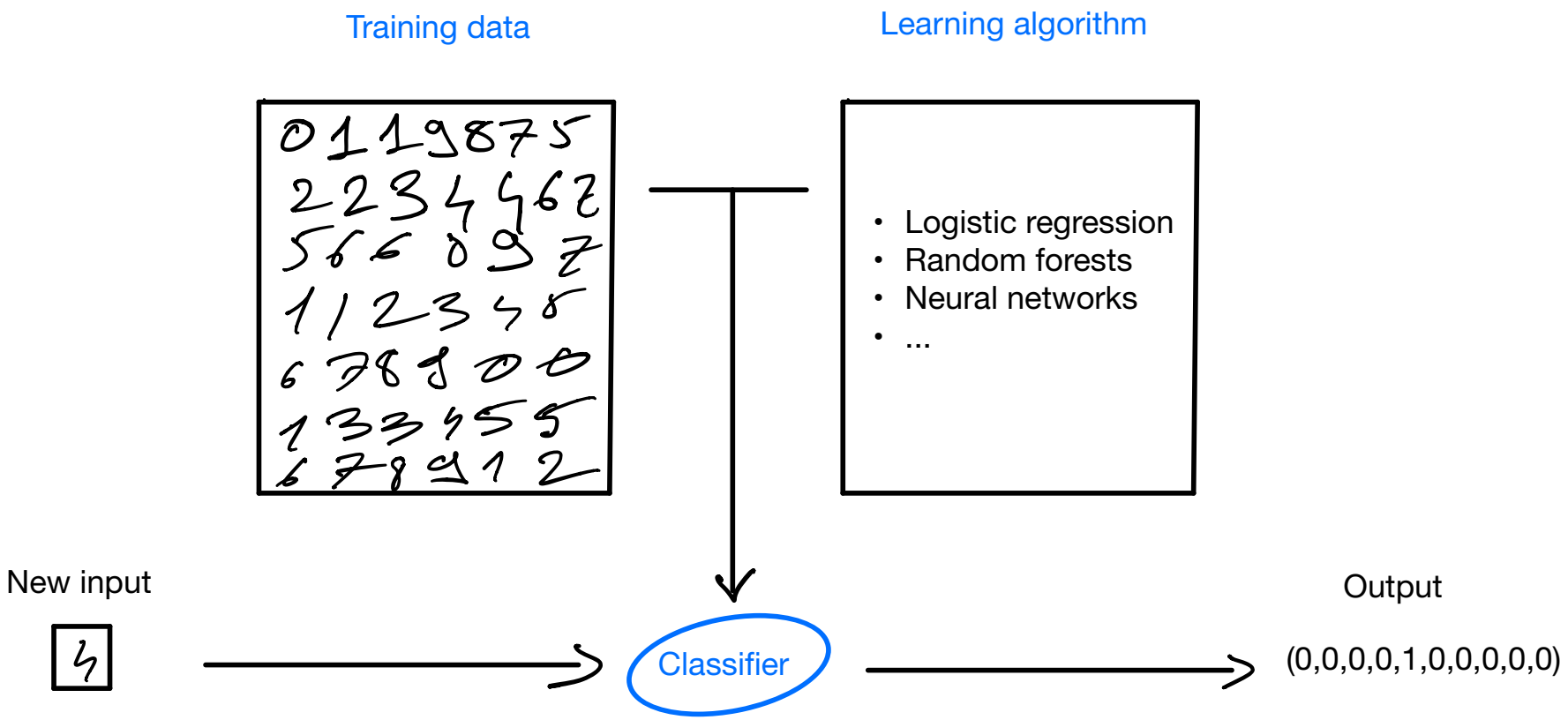
Erwan Scornet, Professor at Sorbonne Université

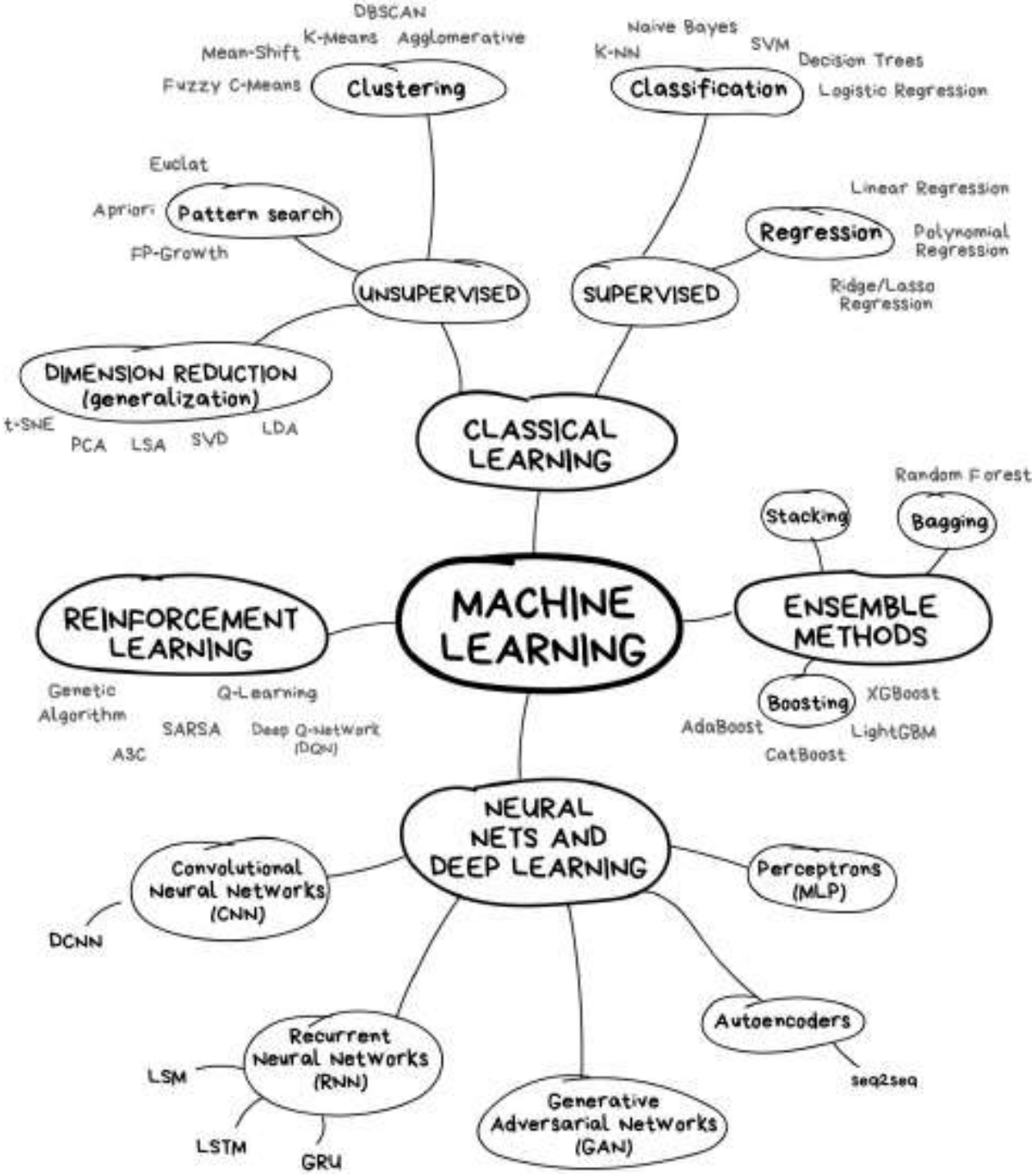
[1]

[1] More content at https://erwanscornet.github.io/, Teaching section.


Machine Learning
Unsupervised Learning
Dimensionality Reduction
Meaningful Compression
Structure Discovery
Big data Visualistaion
Feature Elicitation
Clustering
Recommender Systems
Targetted Marketing
Customer Segmentation
Supervised Learning
Classification
Image Classification
Customer Retention
Idenity Fraud Detection
Diagnostics
Regression
Advertising Popularity Prediction
Weather Forecasting
Population Growth Prediction
Market Forecasting
Estimating life expectancy
Reinforcement Learning
Real-time decisions
Game AI
Robot Navigation
Skill Acquisition
Learning Tasks

Supervised learning




DBSCAN
K-Means
Agglomerative
Mean-Shift
Fuzzy C-Means
Clustering
Naive Bayes
K-NN
SVM
Decision Trees
Classification
Logistic Regression
Euclat
Apriori
Pattern search
FP-Growth
Linear Regression
Regression
Polynomial Regression
Ridge/Lasso Regression
UNSUPERVISED
SUPERVISED
DIMENSION REDUCTION (generalization)
t-SNE
PCA
LSA
SVD
LDA
CLASSICAL LEARNING
Random Forest
Stacking
Bagging
REINFORCEMENT LEARNING
MACHINE LEARNING
ENSEMBLE METHODS
Genetic Algorithm
Q-Learning
SARSA
Deep Q-Network (DQN)
A3C
Boosting
XGBoost
AdaBoost
LightGBM
CatBoost
NEURAL NETS AND DEEP LEARNING
Convolutional Neural Networks (CNN)
DCNN
Perceptrons (MLP)
Recurrent Neural Networks (RNN)
LSM
LSTM
GRU
Autoencoders
seq2seq
Generative Adversarial Networks (GAN)

Outline

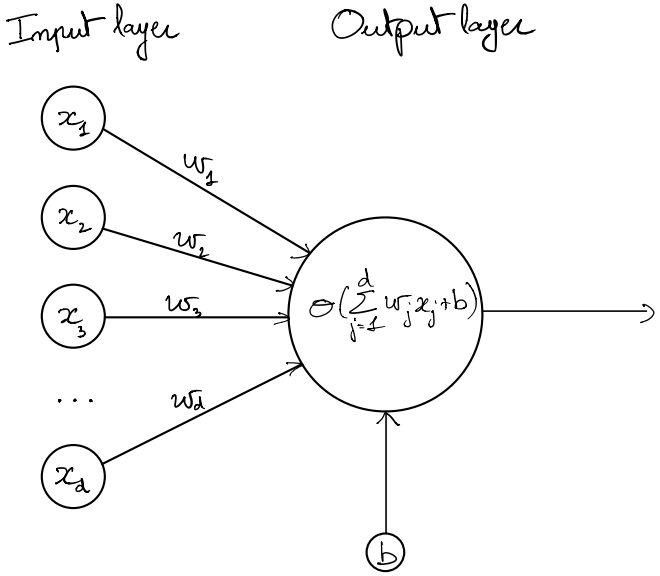
Outline

McCulloch and Pitts neuron - 1943

["A logical calculus of the ideas immanent in nervous activity", McCulloch and Pitts 1943]

In 1943, portrayed with a simple electrical circuit by neurophysiologist Warren McCulloch and mathematician Walter Pitts.

A McCulloch-Pitts neuron takes binary inputs, computes a weighted sum and returns 0 if the result is below threshold and 1 otherwise.

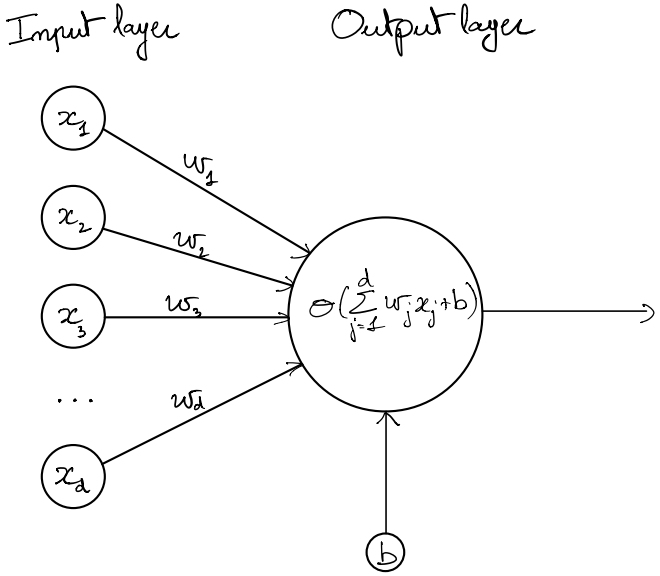


McCulloch and Pitts neuron - 1943

["A logical calculus of the ideas immanent in nervous activity", McCulloch and Pitts 1943]

In 1943, portrayed with a simple electrical circuit by neurophysiologist Warren McCulloch and mathematician Walter Pitts.

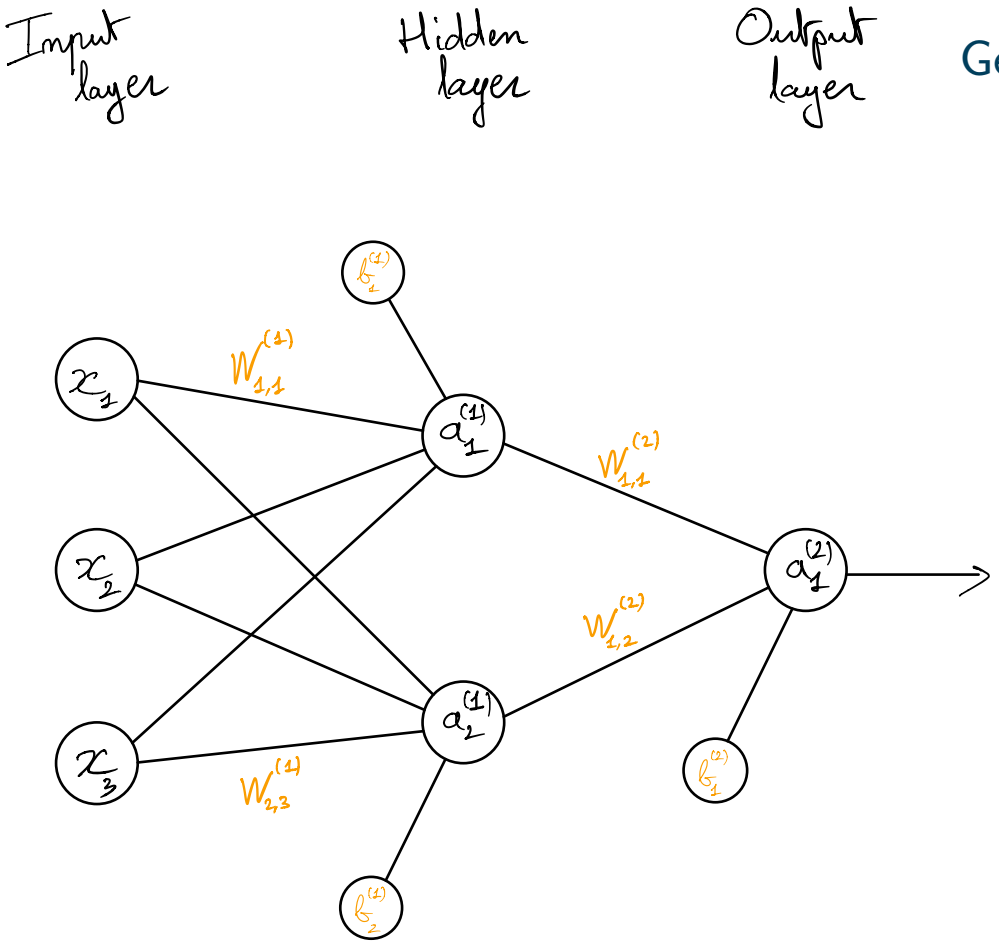
A McCulloch-Pitts neuron takes binary inputs, computes a weighted sum and returns 0 if the result is below threshold and 1 otherwise.



Donald Hebb took the idea further by proposing that neural pathways strengthen over each successive use, especially between neurons that tend to fire at the same time.

[*The organization of behavior: a neuropsychological theory*, Hebb 1949]

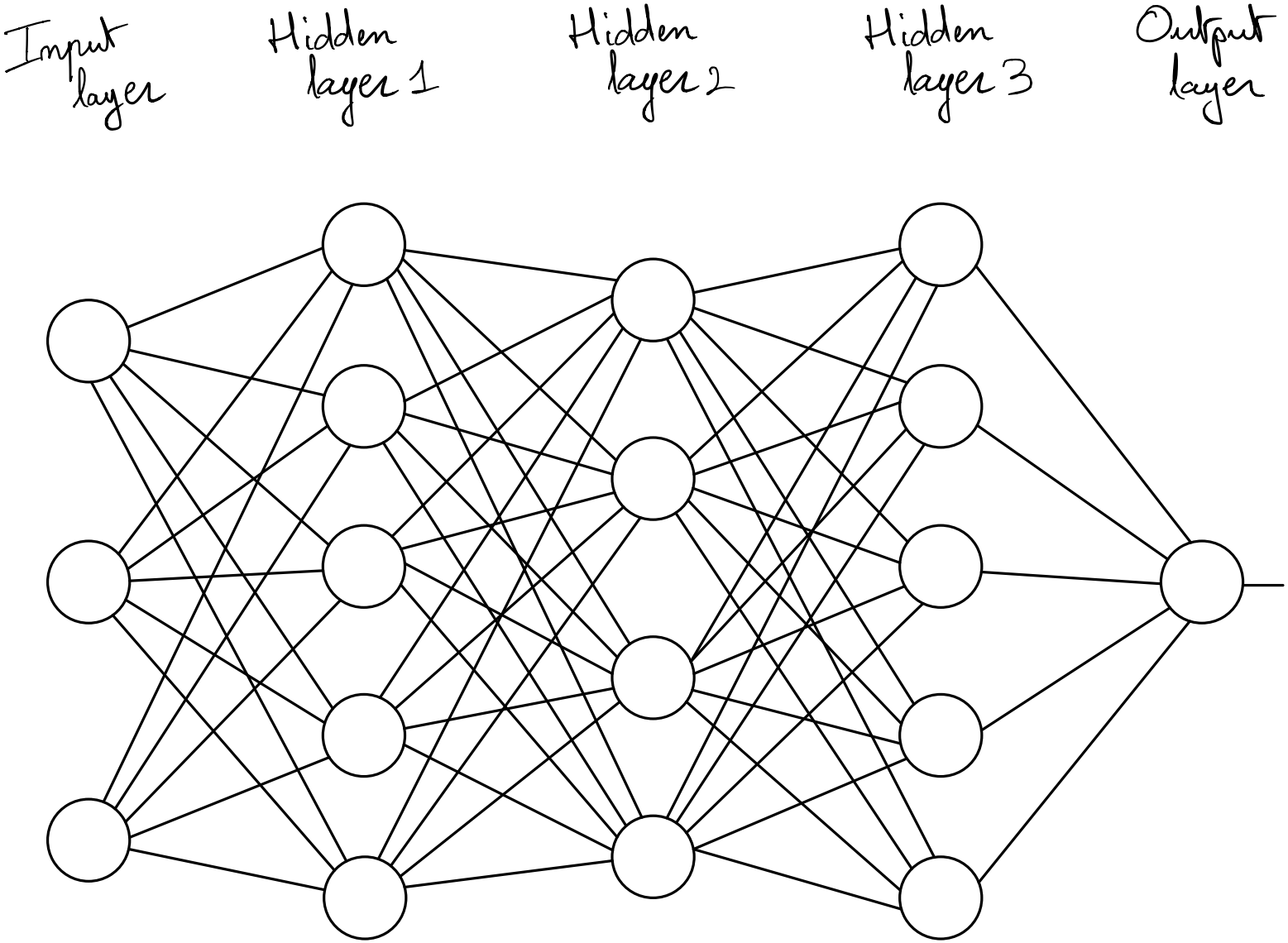
Neural network with one hidden layer



Generic notations:

- $W^{(\ell)}_{i,j}$: weights between the j neuron in the $\ell - 1$ layer and the i neuron of the ℓ layer.
- $b^{(\ell)}_j$: bias of the j neuron of the ℓ layer.
- $a^{(\ell)}_j$: output of the j neuron of the ℓ layer.
- $z^{(\ell)}_j$: input of the j neuron of the ℓ layer, such that $a^{(\ell)}_j = \sigma(z^{(\ell)}_j)$.

How to find weights and bias?



Optimization

Gradient descent algorithm

The gradient of a function $f : \mathbb{R}^p \to \mathbb{R}$ in θ denoted as $\nabla f(\theta)$ is the vector of partial derivatives

$$\nabla f(\theta) = \begin{pmatrix} \frac{\partial f}{\partial \theta_1} \\ \vdots \\ \frac{\partial f}{\partial \theta_p} \end{pmatrix}$$

Gradient descent

- Initialize $\theta^{(0)}$ and $t = 0$.
- While *not convergence* do
  - $\theta^{(t+1)} = \theta^{(t)} - \eta \nabla f(\theta^{(t)})$
  - $t = t + 1$.

Gradient Descent Algorithm

- The prediction of the network is given by $f_\theta(\mathbf{x})$.
- Empirical risk minimization on a batch $B \subset \{1, \ldots, n\}$:

$$\text{Solve} \quad \underset{\theta}{\operatorname{argmin}}\, R_B(\theta) \quad \text{with} \quad R_B(\theta) = \frac{1}{|B|} \sum_{i \in B} \ell(Y_i, f_\theta(\mathbf{X}_i)).$$

- Computationally more efficient than using the full data set.

Stochastic Gradient descent

- Divide the data set into batches $B_1, \ldots, B_{iter}$
- Initialize $\theta^{(0)}$ and $t = 0$.
- While *not convergence* do
  - $\ell = t[iter]$
  - $\theta^{(t+1)} = \theta^{(t)} - \eta \nabla R_{B_\ell}(\theta^{(t)})$
  - $t = t + 1$

Gradient Descent Algorithm

- The prediction of the network is given by $f_\theta(\mathbf{x})$.
- Empirical risk minimization on a batch $B \subset \{1, \ldots, n\}$:

$$\text{Solve} \quad \underset{\theta}{\operatorname{argmin}}\, R_B(\theta) \quad \text{with} \quad R_B(\theta) = \frac{1}{|B|} \sum_{i \in B} \ell(Y_i, f_\theta(\mathbf{X}_i)).$$

- Computationally more efficient than using the full data set.

Stochastic Gradient descent

- Divide the data set into batches $B_1, \ldots, B_{iter}$
- Initialize $\theta^{(0)}$ and $t = 0$.
- While *not convergence* do
  - $\ell = t[iter]$
  - $\theta^{(t+1)} = \theta^{(t)} - \eta \nabla R_{B_\ell}(\theta^{(t)})$
  - $t = t + 1$

How to compute $\nabla_\theta \ell_i$ efficiently?

Backprop Algorithm

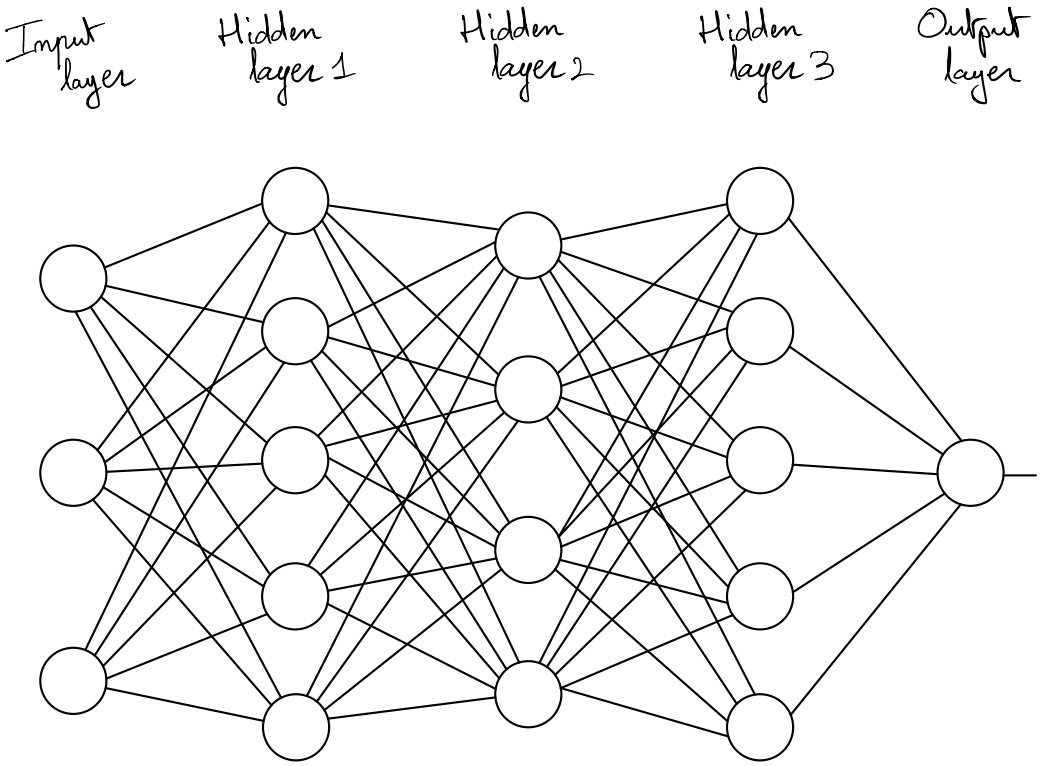
How to compute $\nabla_\theta \ell_i$ efficiently?

A Clever Gradient Descent Implementation

- Popularized by Rumelhart, McClelland, Hinton in 1986.
- Can be traced back to Werbos in 1974.
- Nothing but the use of chain rule derivation with a touch of dynamic programing.

- Key ingredient to make the Neural Networks work!
- Still at the core of Deep Learning algorithm.

Backpropagation idea



Backpropagation equations

Neural network with L layers, with vector output, with quadratic cost

$$C = \frac{1}{2}\|y - a^{(L)}\|^2.$$

By definition,

$$\delta_j^{(\ell)} = \frac{\partial C}{\partial z_j^{(\ell)}}.$$

The four fundamental equations of backpropagation are given by

$$\delta^{(L)} = \nabla_a C \odot \sigma'(z^{(L)}), \tag{1}$$

$$\delta^{(\ell)} = ((w^{(\ell+1)})^T \delta^{(\ell+1)}) \odot \sigma'(z^{(\ell)}) \tag{2}$$

$$\frac{\partial C}{\partial b_j^{(\ell)}} = \delta_j^{(\ell)} \tag{3}$$

$$\frac{\partial C}{\partial w_{j,k}^{(\ell)}} = a_k^{(\ell-1)} \delta_j^{(\ell)}. \tag{4}$$

Backpropagation Algorithm

Let

$$\delta_j^{(\ell)} = \frac{\partial C}{\partial z_j^{(\ell)}},$$

where $z_j^{(\ell)}$ is the entry of the neuron j of the layer ℓ.

Neural network training

(a) Initialize randomly the weights and biases in the network.

(b) For all training samples $(x_i)_{i \in B}$ in the batch B,

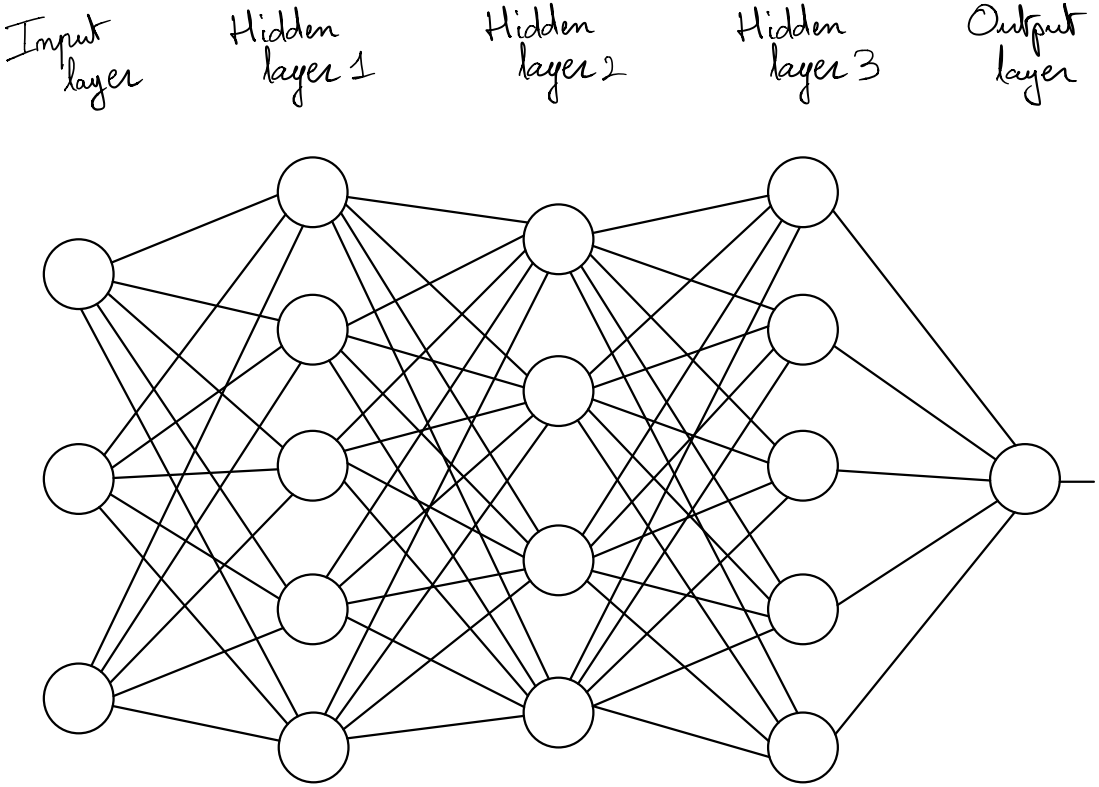
1. **Feedforward:** Send all samples of the batch through the network and store the values of activation function and its derivative, for each neuron.
2. **Output loss:** Compute the neural network loss average on all samples of the batch.
3. **Backpropagation (BP):** Compute recursively the vectors $\delta^{(\ell)}$ starting from $\ell = L$ to $\ell = 1$ with BP equations (1) and (2). Compute the gradient with BP equations (3) and (4).
4. **Optimization:** Update the weights and biases using a gradient-based optimization procedure, using the gradient previously computed.

(c) Repeat step (b) until some convergence criterion is reached.

Playing with neural network: http://playground.tensorflow.org/

Outline

What to set in a neural network?



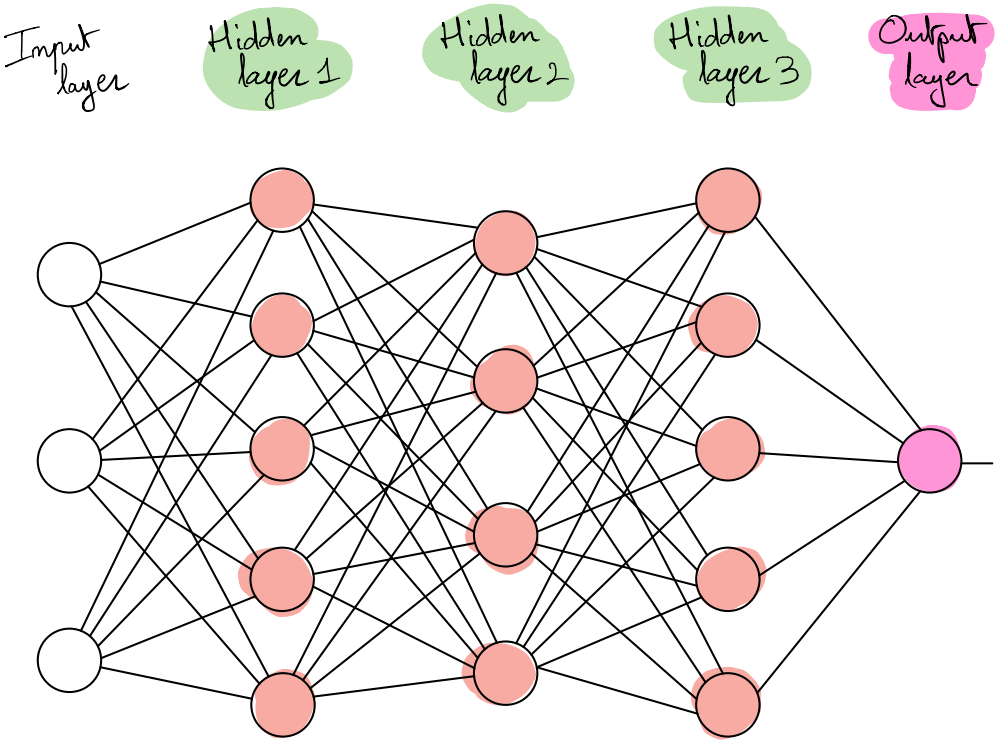
Network structure:

- Number of layers / neurons per layer
- Activation functions
- Output unit
- Specific layers (dropout, batch normalization)

Optimization:

- Optimization algorithm
- Weights / biases initialization
- Loss function

What to set in a neural network?



Network structure:

- Number of layers / neurons per layer
- Activation functions
- Output unit
- Specific layers (dropout, batch normalization)

Optimization:

- Loss function
- Optimization algorithm
- Weights / biases initialization

Number of hidden layers/neurons

- No particular rules for choosing the number of layers or the number of neurons per layer.
- Read research papers related to the task you want to solve and test the architecture they propose.
- You may want to change the architecture a bit to see how it influences the performance.
- Beware: there exist many rules of thumbs which are not supported by evidence (either practical or theoretical).
- Use data-driven strategies:
  - Network pruning following the procedure training/pruning/training/pruning/...
    ["What is the state of neural network pruning?", Blalock et al. 2020]
  - More complex evolutionary algorithms
    ["AgEBO-Tabular: Joint Neural Architecture and Hyperparameter Search with Autotuned Data-Parallel Training for Tabular Data", Egele et al. 2020]

Sigmoid activation function

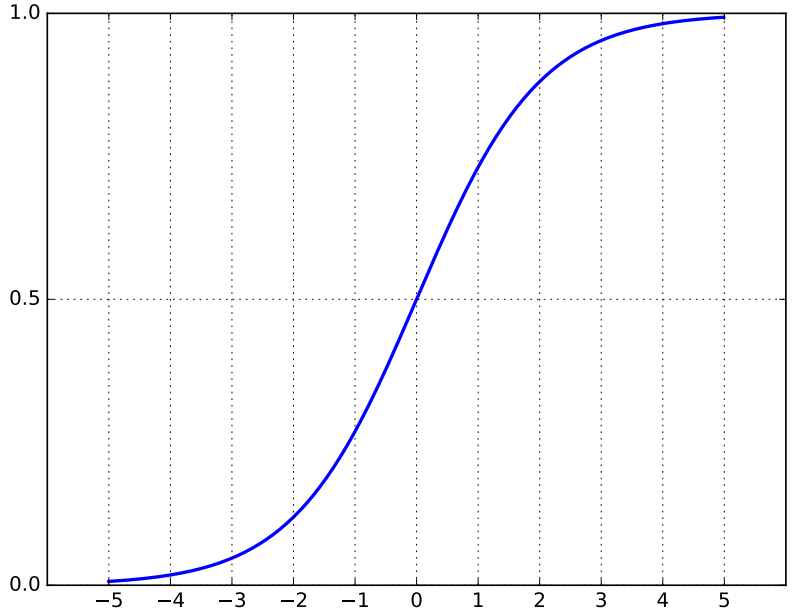


Figure: Sigmoid activation function σ

$$\sigma : x \mapsto \frac{\exp(x)}{1 + \exp(x)}$$

Comments:

- Saturated function due to horizontal asymptotes:
  - Gradient is close to zero in these two areas ($\pm\infty$)
  - Rescaling the inputs of each layer can help to avoid these areas.
- Sigmoid is not a zero-centered function
  - Rescaling data
- Computing $\exp(x)$ is a bit costly

Rectified Linear Unit (ReLU)

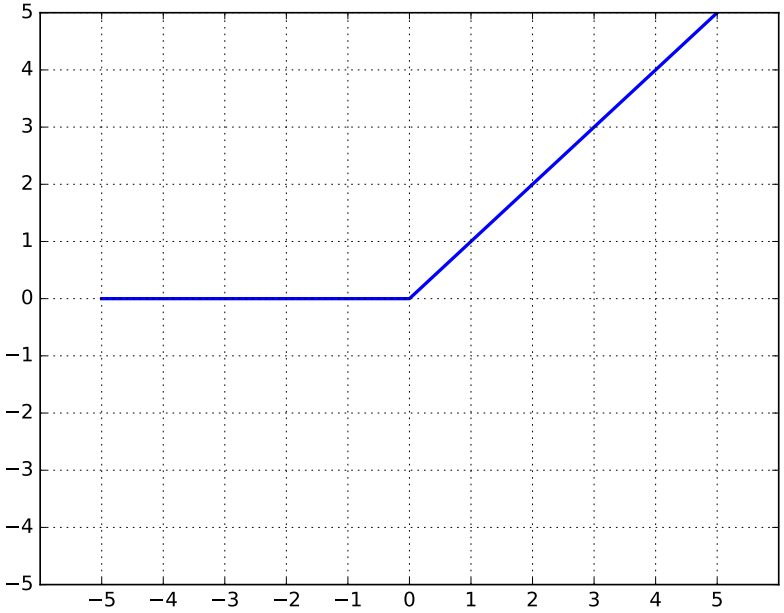


Figure: Rectified Linear Unit (ReLU)

$$ReLU : x \mapsto \max(0, x)$$

Comments:

- Not a saturated function in $+\infty$
- But saturated (and null!) in the region $x \leq 0$
- Computationally efficient
- Empirically, convergence is faster than sigmoid/tanh.
- Plus: biologically plausible

More on ReLU

The idea of ReLU in neural networks seems to appear in ["Cognitron: A self-organizing multilayered neural network"; "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition", Fukushima 1975; Fukushima and Miyake 1982].

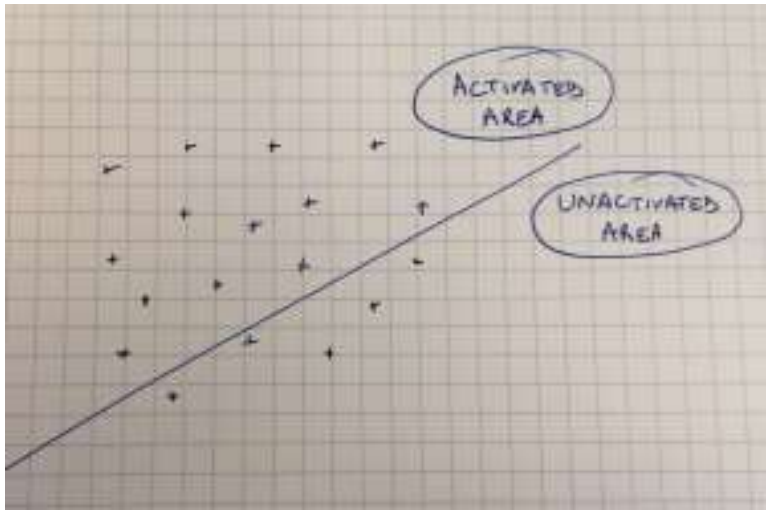


Figure: Good parameter initialization - ReLU is active

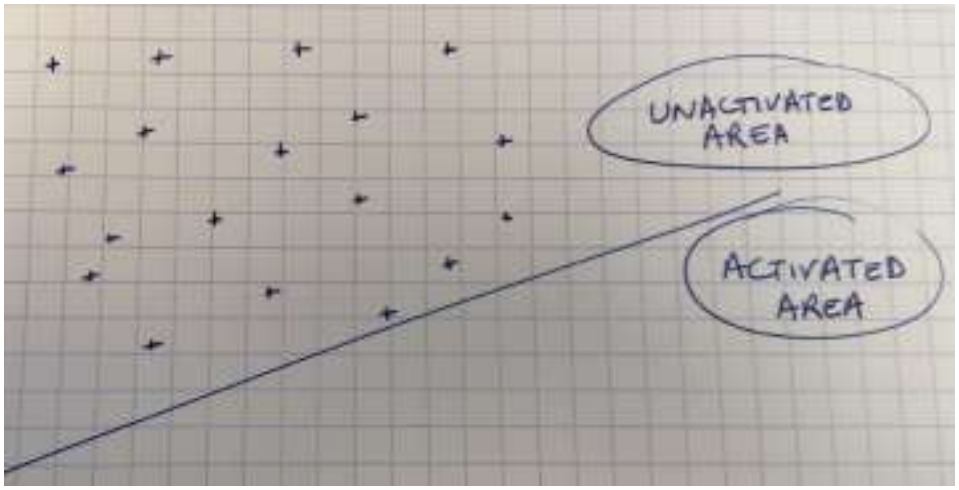


Figure: Bad parameter initialization - ReLU outputs zero

ReLU output can be zero but positive initial bias can help.

Related to biology ["Deep sparse rectifier neural networks", Glorot, Bordes, et al. 2011]:

- Most of the time, neurons are inactive.
- when they activate, their activation is proportional to their input.

Output units

- Linear output unit:
$$\hat{y} = W^T h + b$$
$\rightarrow$ Linear regression based on the new variables h.

- Sigmoid output unit, used to predict $\{0, 1\}$ outputs:
$$\mathbb{P}(Y = 1|h) = \sigma(W^T h + b),$$
where $\sigma(t) = e^t/(1 + e^t)$.

  $\rightarrow$ Logistic regression based on the new variables h.

- Softmax output unit, used to predict $\{1, \ldots, K\}$:
$$\mathrm{softmax}(z)_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}$$
where, each z_i is the activation of one neuron of the previous layer, given by $z_i = W_i^T h + b_i$.

  $\rightarrow$ Multinomial logistic regression based on the new variables h.

Cost functions

- Mean Square Error (MSE)

$$\frac{1}{n}\sum_{i=1}^{n}\ell(Y_i, f_\theta(\mathbf{X}_i)) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - f_\theta(\mathbf{X}_i))^2$$

- Mean Absolute Error

$$\frac{1}{n}\sum_{i=1}^{n}\ell(Y_i, f_\theta(\mathbf{X}_i)) = \frac{1}{n}\sum_{i=1}^{n}|Y_i - f_\theta(\mathbf{X}_i)|$$

- Cross entropy (or negative log-likelihood):

$$\ell(y_i, f_\theta(\mathbf{x}_i)) = -\log\left([f_\theta(\mathbf{x}_i)]_{y_i}\right) \tag{5}$$

  - Prevent saturation phenomenon:

$$-\log(\mathbb{P}(Y = y_i|\mathbf{X} = \mathbf{x}_i)) = -\log(\sigma((2y-1)(W^T h + b))), \tag{6}$$

    with

$$\sigma(t) = \frac{e^t}{1+e^t}$$

    Usually, saturation occurs when $(2y-1)(W^T h + b) \ll -1$. In that case, $-\log(\mathbb{P}(Y = y_i|X))$ is linear in W and b which makes the gradient easy to compute, and the gradient descent easy to implement.

Weight initialization

Idea: the variance of the input should be the same as the variance of the output.

Let w_j be any weight between layer j and layer $j+1$.

1. He et al. initialization

   ["Delving deep into rectifiers: Surpassing human-level performance on imagenet classification", He et al. 2015]

   Initialize bias to zero and weights randomly using

   $$w_j \sim \mathcal{N}\left(0, \frac{\sqrt{2}}{n_j}\right),$$

   where n_j is the size of layer j.

2. Xavier initialization

   ["Understanding the difficulty of training deep feedforward neural networks", Glorot and Bengio 2010]

   Initialize bias to zero and weights randomly using

   $$w_j \sim \mathcal{U}\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right],$$

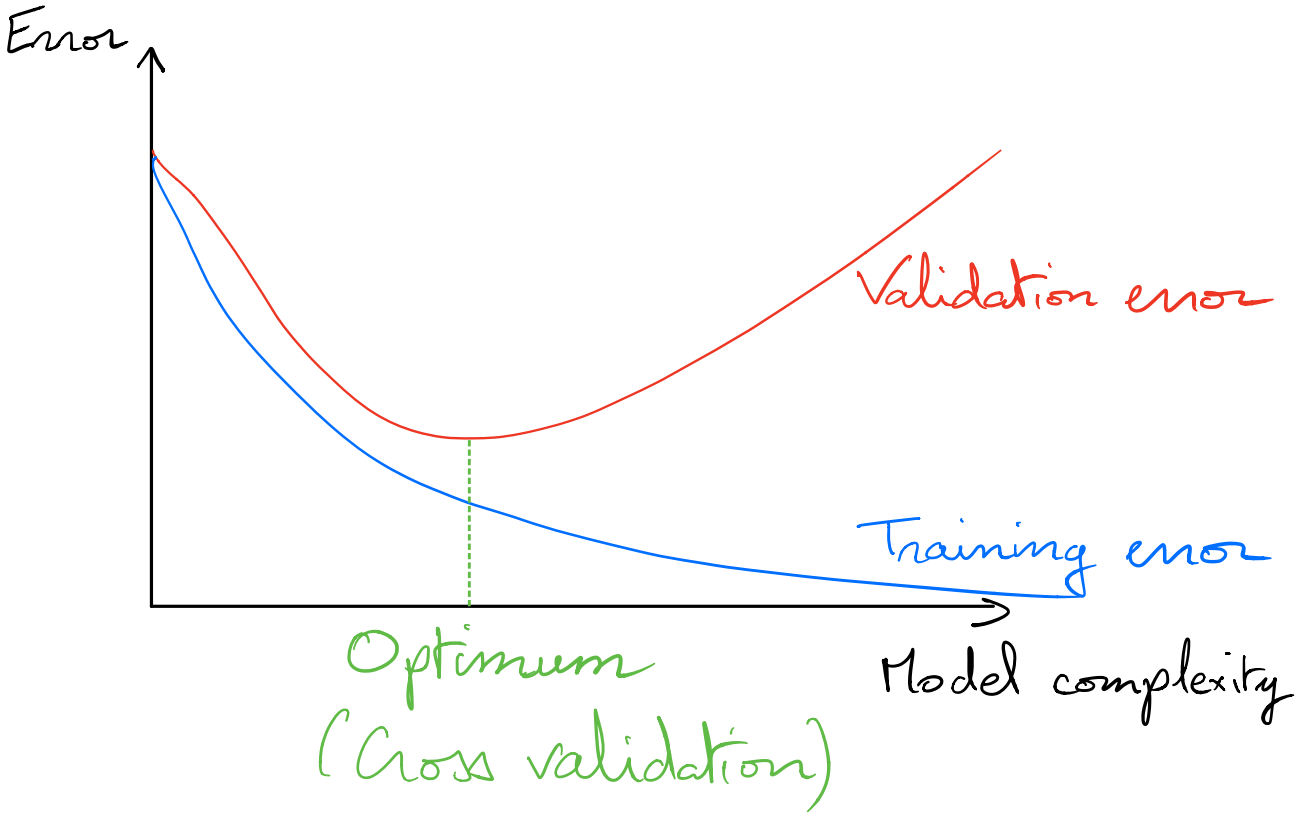
   where n_j is the size of layer j

   $\rightarrow$ Not theoretically valid for ReLU

Bonus: ["All you need is a good init", Mishkin and Matas 2015]

Outline

Regularizing to avoid overfitting



Avoid overfitting by imposing some constraints over the parameter space.

Reducing variance and increasing bias.

Overfitting

Many different manners to avoid overfitting:

- Penalization (L1 or L2)
  Replacing the cost function $\mathcal{L}$ by $\tilde{\mathcal{L}}(\theta, X, y) = \mathcal{L}(\theta, X, y) + \mathrm{pen}(\theta)$.
- Soft weight sharing - see CNN lecture
  Reduce the parameter space artificially by imposing explicit constraints.
- Dropout
  Randomly kill some neurons during optimization and predict with the full network.
- Batch normalization
  Renormalize a layer inside a batch, so that the network does not overfit on this particular batch.
- Early stopping
  Stop the gradient descent procedure when the error on the validation set increases.

Outline

Dropout

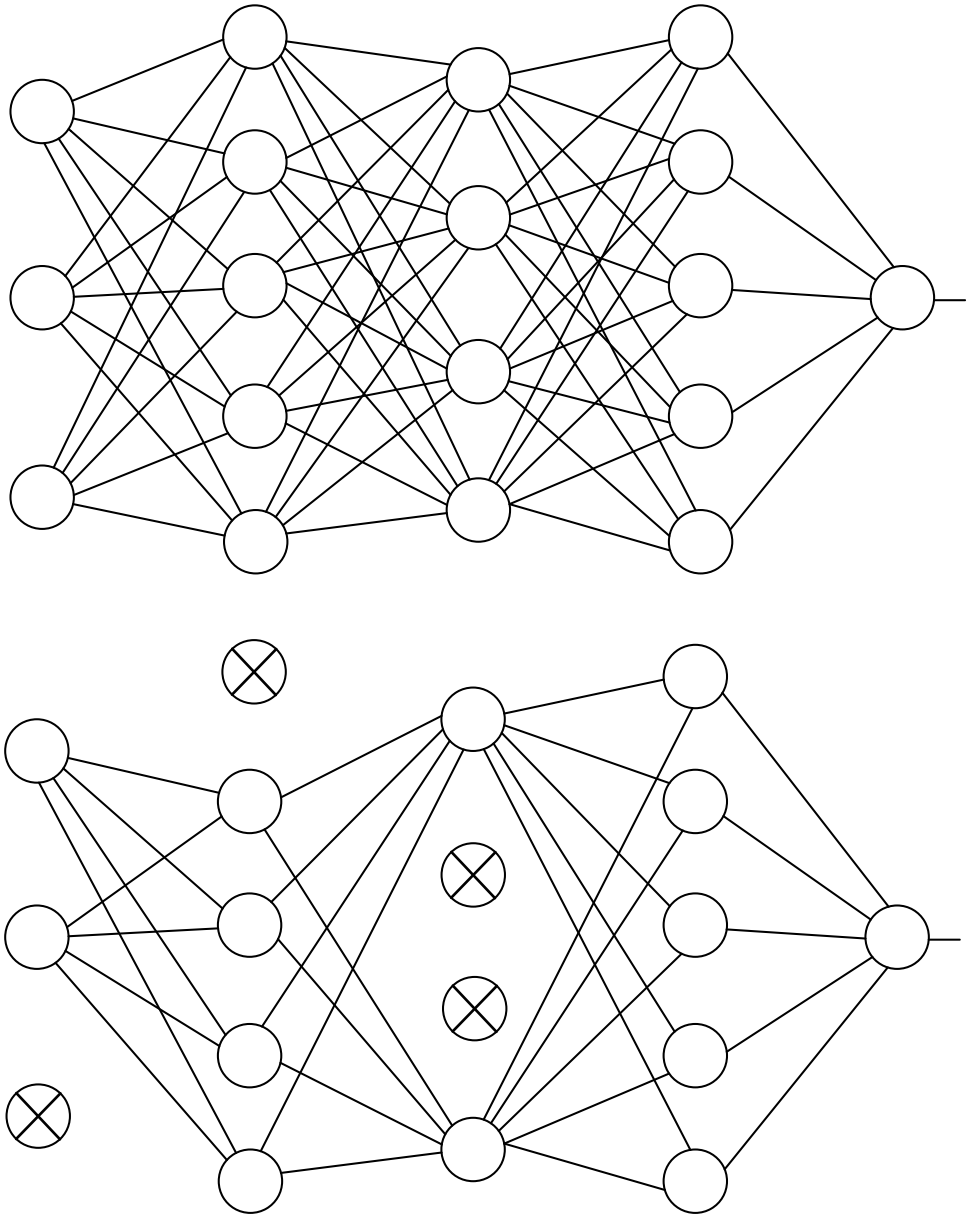
Dropout refers to dropping out units (hidden and visible) in a neural network, i.e., temporarily removing it from the network, along with all its incoming and outgoing connections.

Each unit is independently dropped with probability

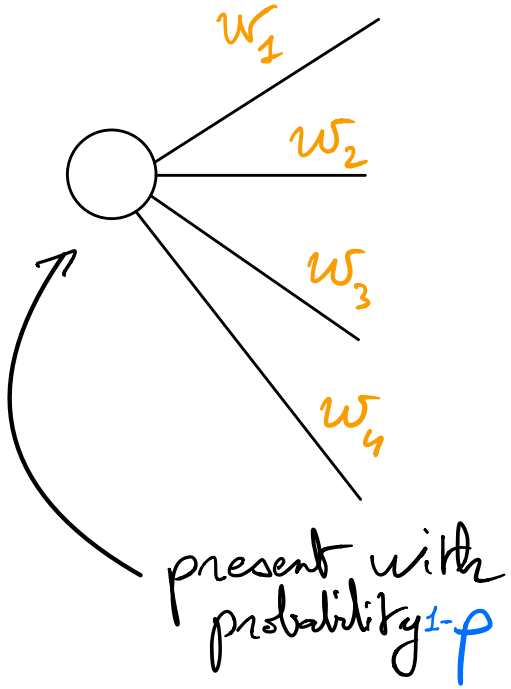
- $p = 0.5$ for hidden units
- $p \in [0, 0.5]$ for input units, usually $p = 0.2$.

["Improving neural networks by preventing co-adaptation of feature detectors", Hinton, N. Srivastava, et al. 2012]

Dropout



At train time



At test time

Dropout algorithm

Training step. While *not convergence*

1. Inside one epoch, for each mini-batch of size m,
   1. Sample m different mask. A mask consists in one Bernoulli per node of the network (inner and entry nodes but not output nodes). These Bernoulli variables are *i.i.d.*. Usually
      - the probability of selecting an hidden node is 0.5
      - the probability of selecting an input node is 0.8
   2. For each one of the m observation in the mini-batch,
      - Do a forward pass on the masked network
      - Compute backpropagation in the masked network
      - Compute the average gradient
   3. Update the parameter according to the usual formula.

Prediction step.
Use all neurons in the network with weights given by the previous optimization procedure, times the probability p of being selected (0.5 for inner nodes, 0.8 for input nodes).

Outline

Batch normalization

The network converges faster if its input are scaled (mean, variance) and decorrelated.

["Efficient backprop", LeCun, Bottou, Orr, et al. 1998]

Hard to decorrelate variables: requiring to compute covariance matrix.

["Batch normalization: Accelerating deep network training by reducing internal covariate shift", Ioffe and Szegedy 2015]

Ideas:

- Improving gradient flows
- Allowing higher learning rates
- Reducing strond dependence on initialization
- Related to regularization (maybe slightly reduces the need for Dropout)

Algorithm

See ["Batch normalization: Accelerating deep network training by reducing internal covariate shift", Ioffe and Szegedy 2015]

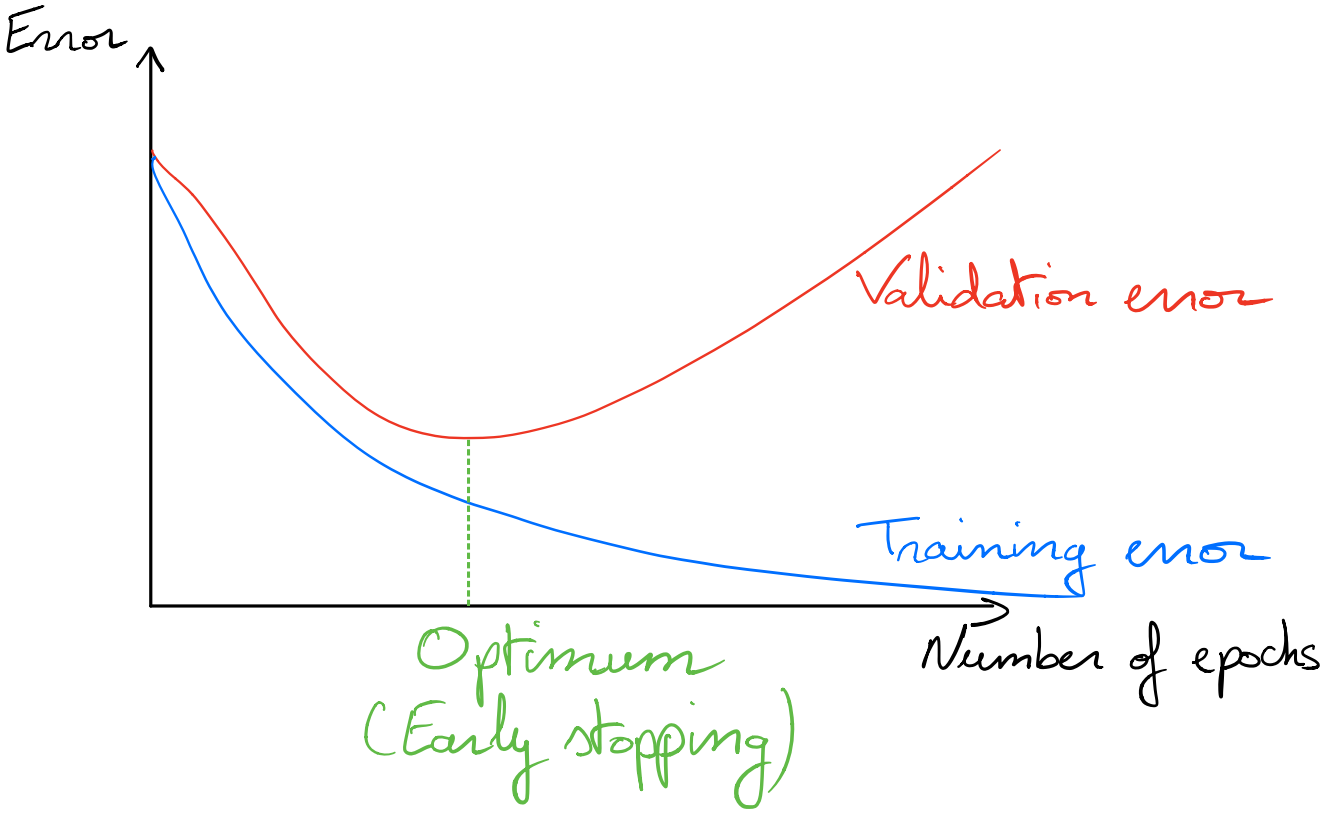
1. For every neuron k in the first layer, which outputs $x_i^{(k)}$ for the ith observation,
   1. $\mu_B^{(k)} = \frac{1}{m}\sum_{i=1}^m x_i^{(k)}$
   2. $\sigma_{B,k}^2 = \frac{1}{m}\sum_{i=1}^m (x_i^{(k)} - \mu_B^{(k)})^2$
   3. $\hat{x}_i^{(k)} = \frac{x_i^{(k)} - \mu_B^{(k)}}{\sqrt{\sigma_{B,k}^2 + \varepsilon}}$
   4. $y_i^{(k)} = \gamma^{(k)}\hat{x}_i^{(k)} + \beta^{(k)} \equiv BN_{\gamma^{(k)},\beta^{(k)}}(x_i^{(k)})$
2. $y_i^{(k)}$ is fed to the next layer and the procedure iterates.
3. Backpropagation is performed on the network parameters including $(\gamma^{(k)}, \beta^{(k)})$ for all $k = 1, \ldots, H_1$, where $H_1 \in \mathbb{N}$ is the number of neurons in the first layer.
4. For inference, compute the average over many training batches $\mathcal{B}$ of size m:
$$\mathbb{E}_{\mathcal{B}}[x^{(k)}] = \mathbb{E}_{\mathcal{B}}[\mu_{\mathcal{B}}^{(k)}] \quad \text{and} \quad \mathbb{V}_{\mathcal{B}}[x^{(k)}] = \frac{m}{m-1}\mathbb{E}_{\mathcal{B}}[\sigma_{\mathcal{B},k}^2].$$
5. For inference, replace every function $x^{(k)} \mapsto BN_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$ in the network by
$$x^{(k)} \mapsto \gamma\left(\frac{x^{(k)} - \mathbb{E}_{\mathcal{B}}[x^{(k)}]}{\sqrt{\mathbb{V}_{\mathcal{B}}[x^{(k)}] + \varepsilon}}\right) + \beta^{(k)}.$$

Outline

Early stopping

Idea:

- Store the parameter values that lead to the lowest error on the validation set
- Return these values rather than the latest ones.



Early stopping algorithm

Parameters:

- patience p of the algorithm: number of times to observe no improvement on the validation set error before giving up;
- the number of steps n between evaluations.

How to implement early stopping?

- First idea: use early stopping to determine the best number of iterations $i^\star$ and train on the whole data set for $i^\star$ iterations.
- Second idea: use early stopping to determine the best parameters and the training error at the best number of iterations. Starting from $\theta^\star$, train on the whole data set until the error matches the previous early stopping error.

Outline

Neural Network reborn

Renewed interest in 2006: ["A fast learning algorithm for deep belief nets", Hinton, Osindero, et al. 2006]

Propose a way to train deep neural nets:

- Train the first layer.
- Add a layer on top of it and train only this layer.
- Repeat the process until the network is deep enough.
- Use this network as a warm start to train the whole network.

Technical reasons for this new growing interest:

- Larger datasets
- More powerful computers
- Small number of algorithmic changes
  1. MSE replaced by cross-entropy
  2. ReLU (Fukushima, 1975, 1980)

Using classical networks for images?

No, for two reasons:

- Do not take into account the spatial organization of pixels (if the pixels are permuted, the output of the network would be the same, whereas the image would change drastically)
- Non robust to image shifting

Idea:

- Apply local transformation to a set of nearby pixels (spatial nature of image is used)
- Repeat this transformation over the whole image (resulting in a shift-invariant output)

Not a new idea: trace back to perceptron and studies about the visual cortex of a cat. The cat is able to

- detect oriented edges, end-points, corners (low-level features)
- combine them to detect more complex geometrical forms (high-level features)

Outline

Convolutional neural networks (CNNs)

- Neural networks that use **convolution instead of matrix product** in one of the layers
- A CNN layer typically includes 3 operations: **convolution**, **activation** and **pooling**
- Using the more general idea of **parameters sharing**, instead of **full connection** (convolution instead of matrix product)

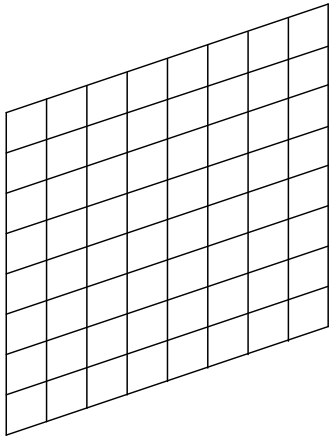
Convolution operator in neural networks is as follows

$$O(i,j) = (I \star K)(i,j) = \sum_k \sum_l I(i+k, j+l) K(k,l)$$

- I is the input and K is called the kernels
- The kernel K will be **learned** (replaces the weights $\mathbf{W}$ in a fully connected layer)

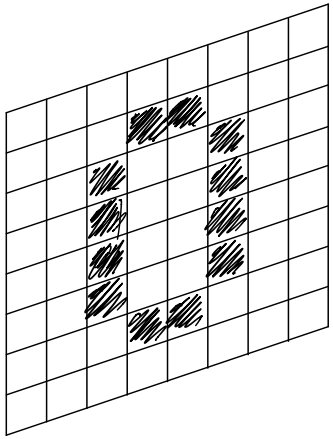
Convolution - Black and White images

- Size of the input image is $8 \times 8 \times 1$ (height, width, depth)



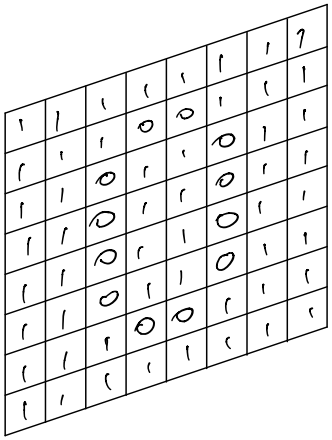
Convolution - Black and White images

- Size of the input image is $8 \times 8 \times 1$ (height, width, depth)



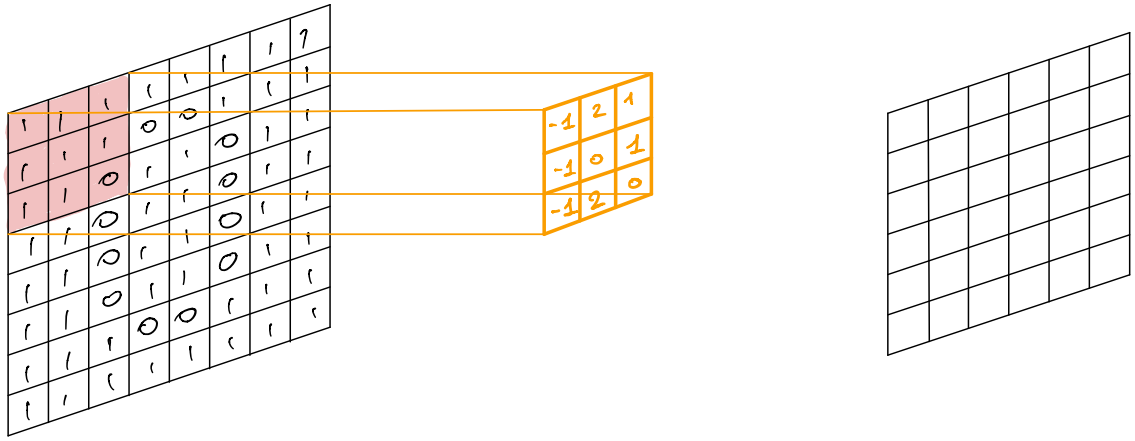
Convolution - Black and White images

- Size of the input image is $8 \times 8 \times 1$ (height, width, depth)



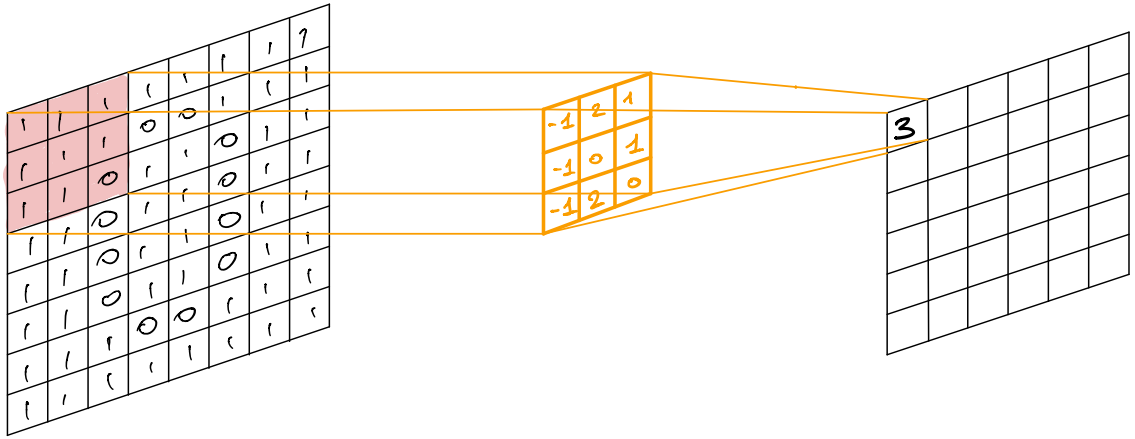
Convolution - Black and White images

- Size of the input image is $8 \times 8 \times 1$ (height, width, depth)
- Size of the kernel is $3 \times 3 \times 1$



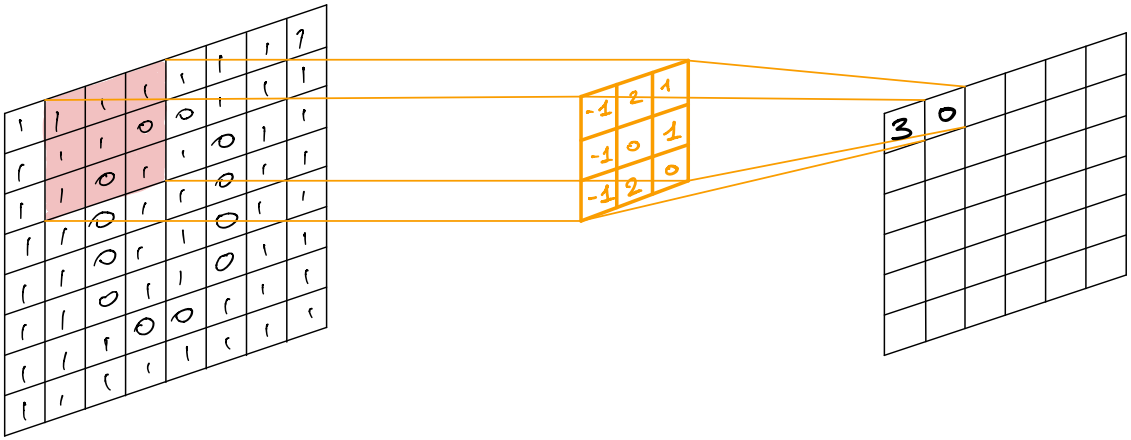
Convolution - Black and White images

- Size of the input image is $8 \times 8 \times 1$ (height, width, depth)
- Size of the kernel is $3 \times 3 \times 1$



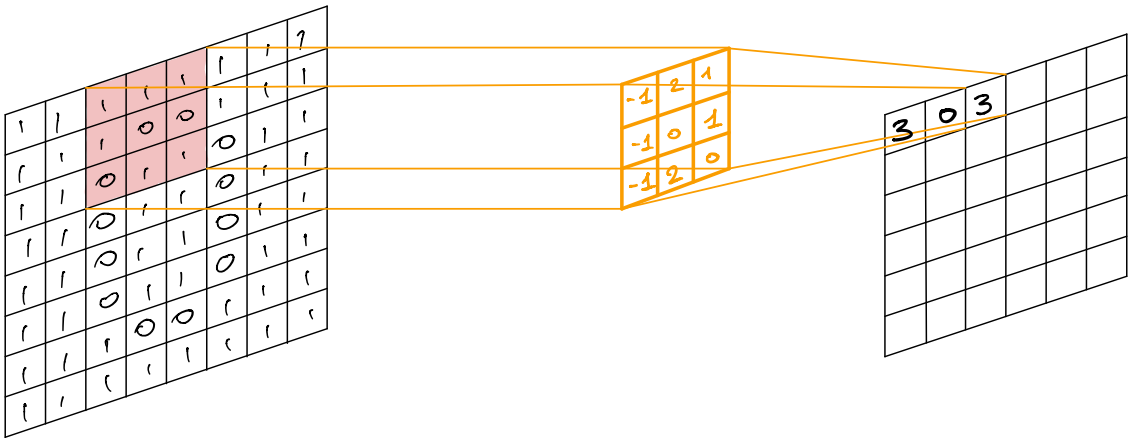
Convolution - Black and White images

- Size of the input image is $8 \times 8 \times 1$ (height, width, depth)
- Size of the kernel is $3 \times 3 \times 1$



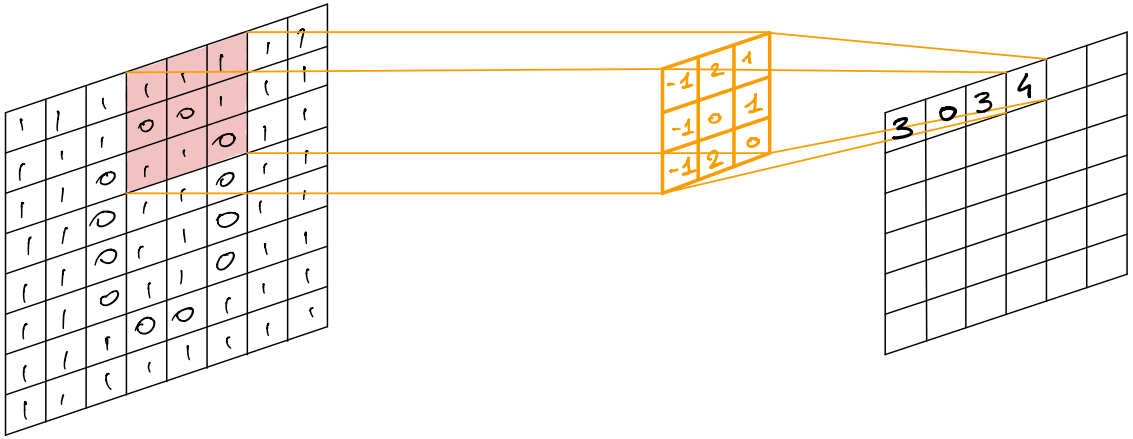
Convolution - Black and White images

- Size of the input image is $8 \times 8 \times 1$ (height, width, depth)
- Size of the kernel is $3 \times 3 \times 1$



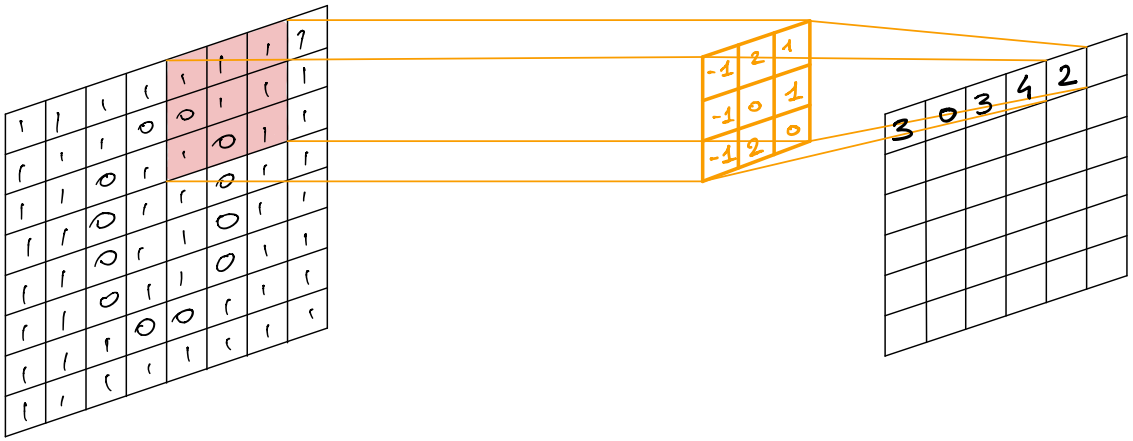
Convolution - Black and White images

- Size of the input image is $8 \times 8 \times 1$ (height, width, depth)
- Size of the kernel is $3 \times 3 \times 1$



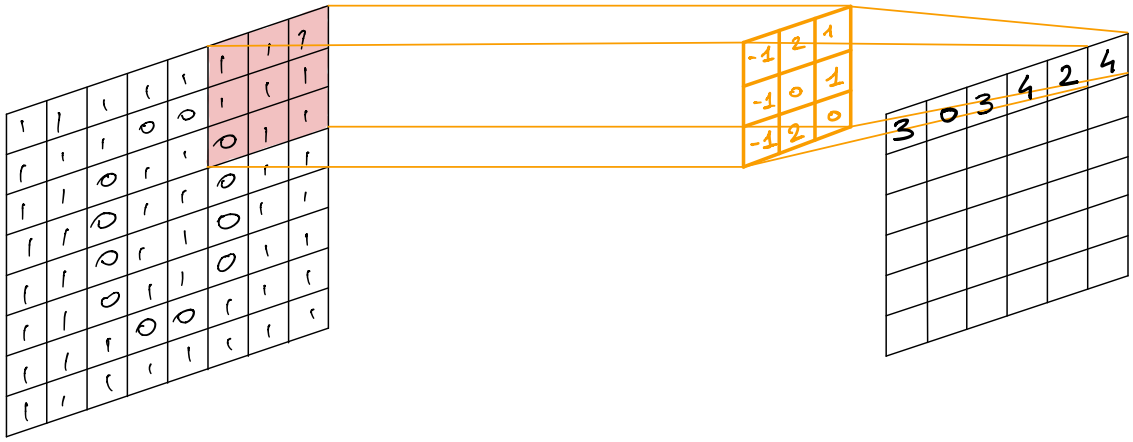
Convolution - Black and White images

- Size of the input image is $8 \times 8 \times 1$ (height, width, depth)
- Size of the kernel is $3 \times 3 \times 1$



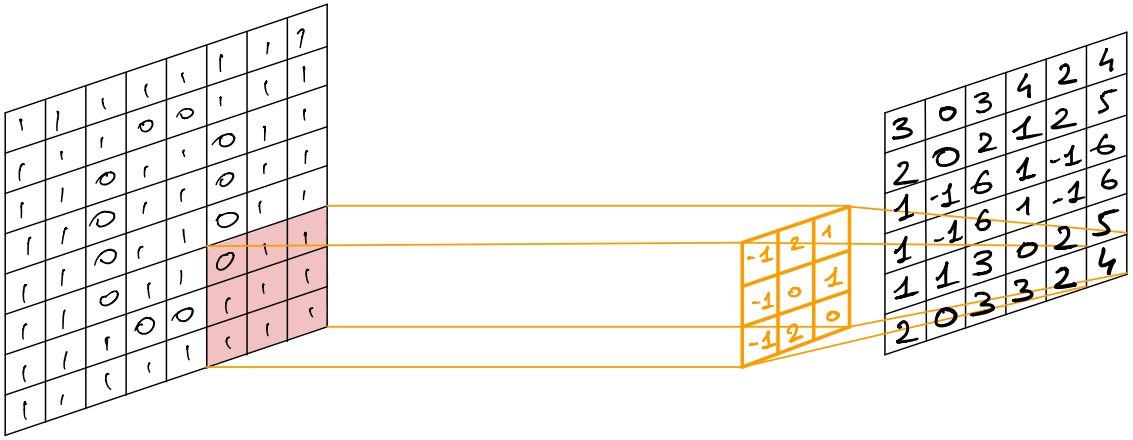
Convolution - Black and White images

- Size of the input image is $8 \times 8 \times 1$ (height, width, depth)
- Size of the kernel is $3 \times 3 \times 1$



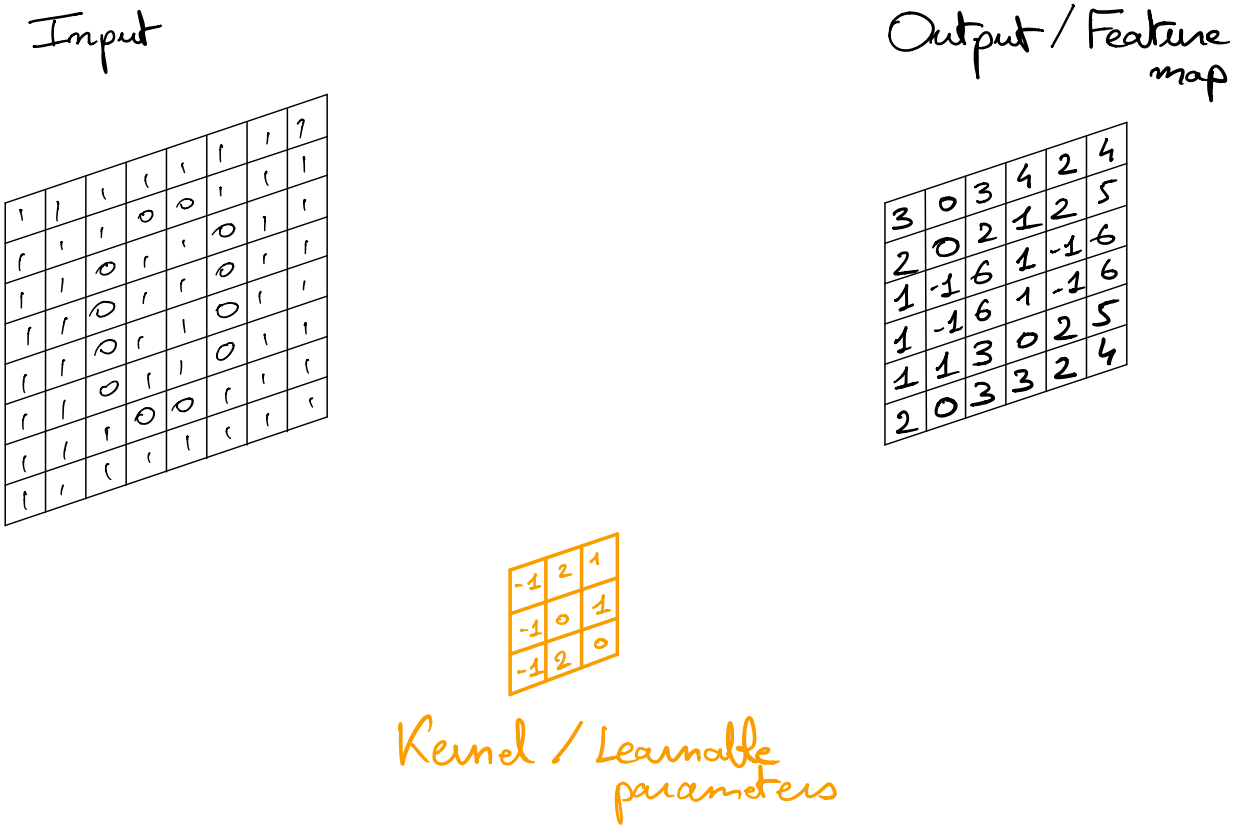
Convolution - Black and White images

- Size of the input image is $8 \times 8 \times 1$ (height, width, depth)
- Size of the kernel is $3 \times 3 \times 1$



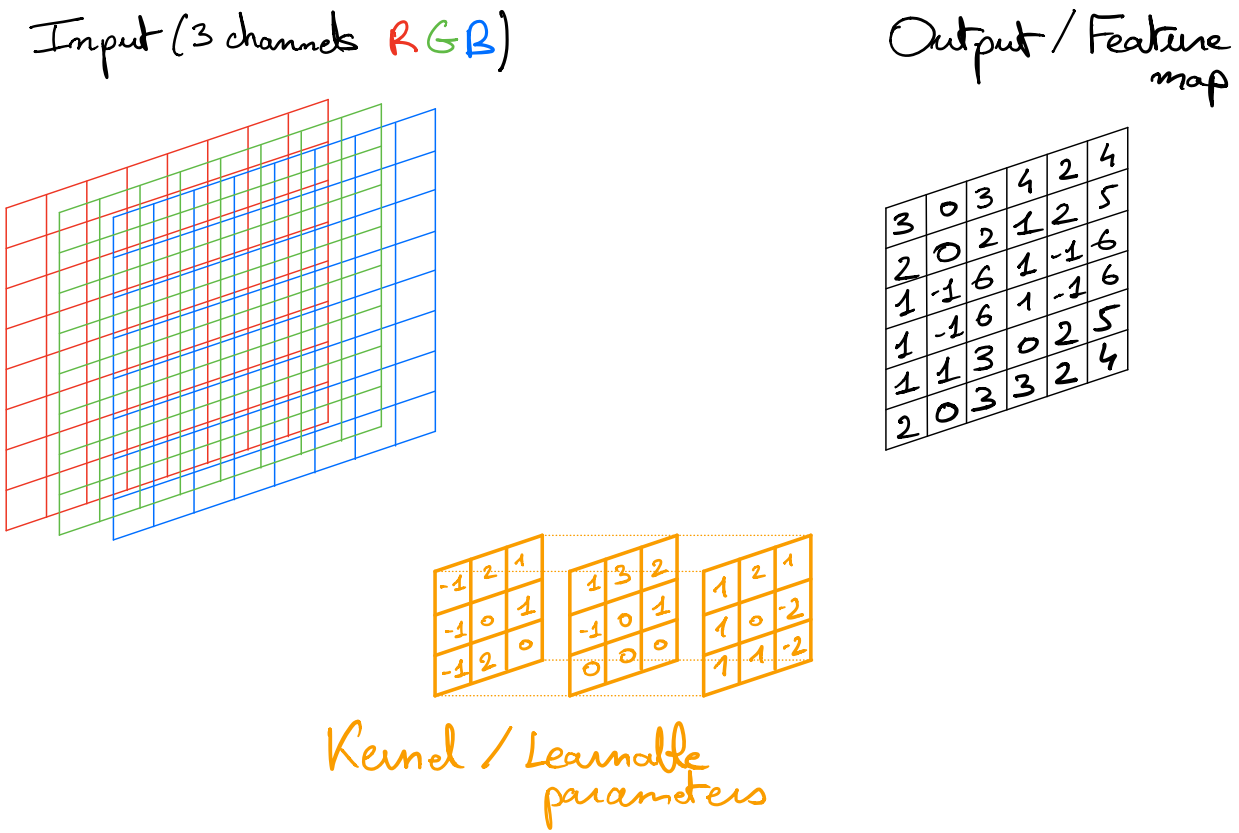
Convolution - Black and White images

- Size of the input image is $8 \times 8 \times 1$ (height, width, depth)
- Size of the kernel is $3 \times 3 \times 1$



Convolution - RGB

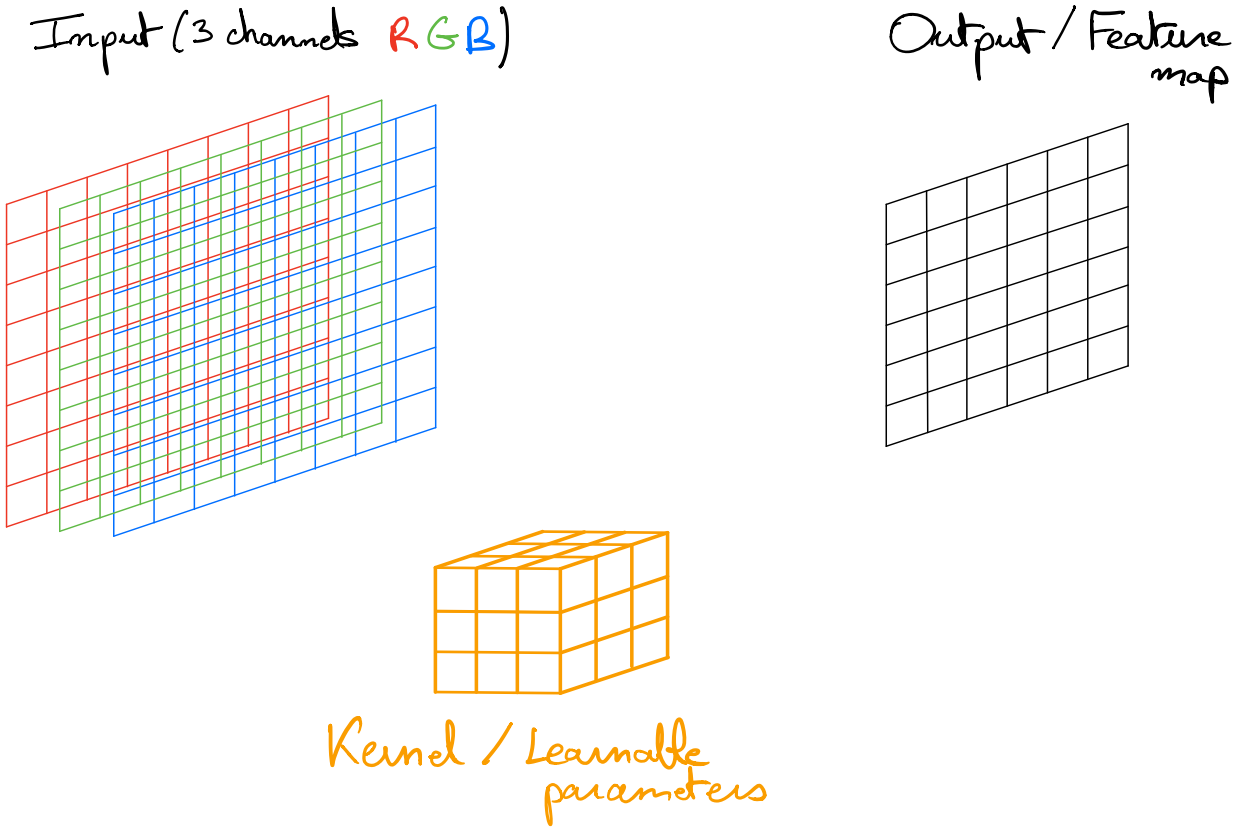
- Size of the input image is $8 \times 8 \times 3$ (height, width, depth)
- Size of the kernel is $3 \times 3 \times 3$



Warning: every filter is small spatially (along width and height), but extends through the full depth of the input volume.

Convolution - RGB

- Size of the input image is $8 \times 8 \times 3$ (height, width, depth)
- Size of the kernel is $3 \times 3 \times 3$

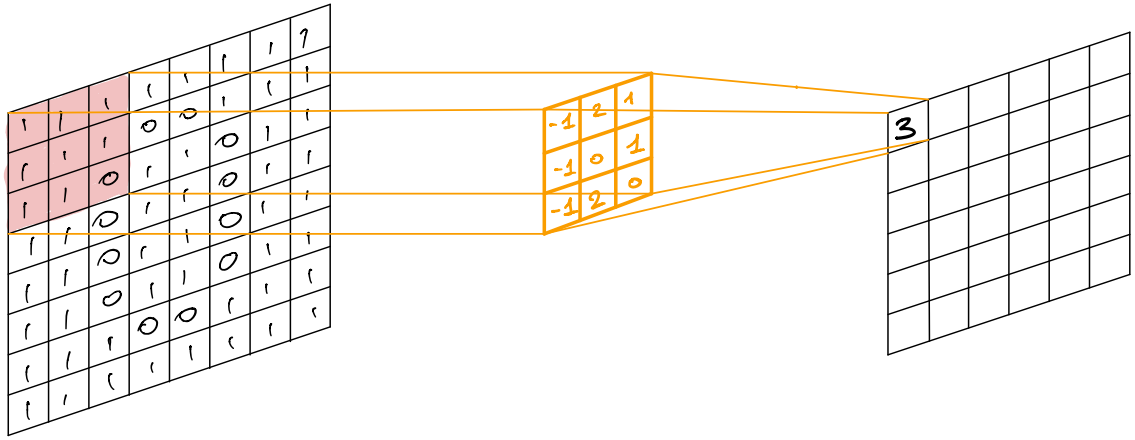


Warning: every filter is small spatially (along width and height), but extends through the full depth of the input volume.

Parameters of convolutional layer 1/4

Four hyperparameters control the size of the output volume: the kernel size, the depth of the output volume, the stride and the zero-padding.

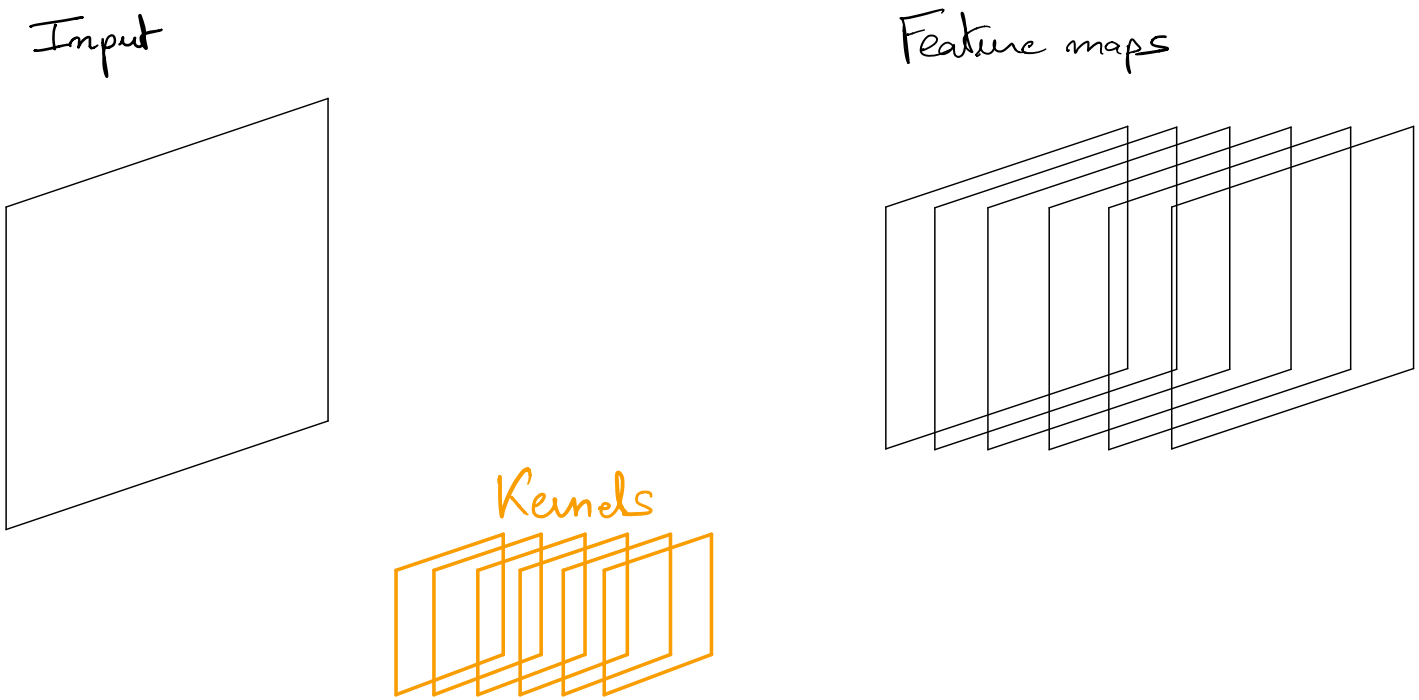
- The size of the kernel (typically 3×3, 5×5).



Parameters of convolutional layer 2/4

Four hyperparameters control the size of the output volume: the kernel size, the depth of the output volume, the stride and the zero-padding.

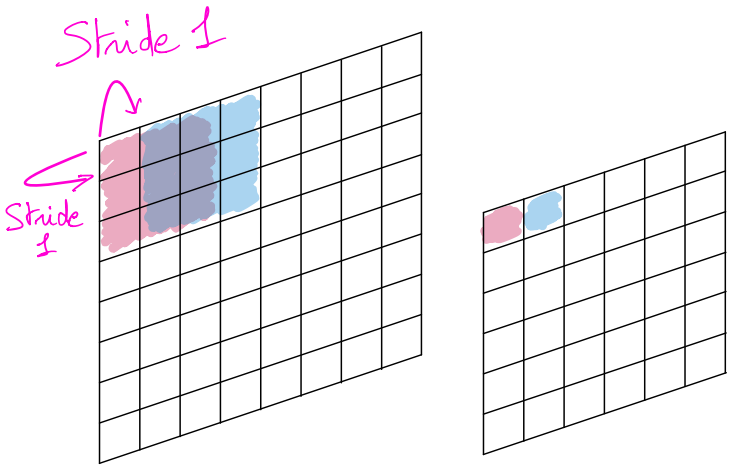
- The size of the kernel,
- The depth of the output volume, i.e., the number of filters/activation maps/feature maps.

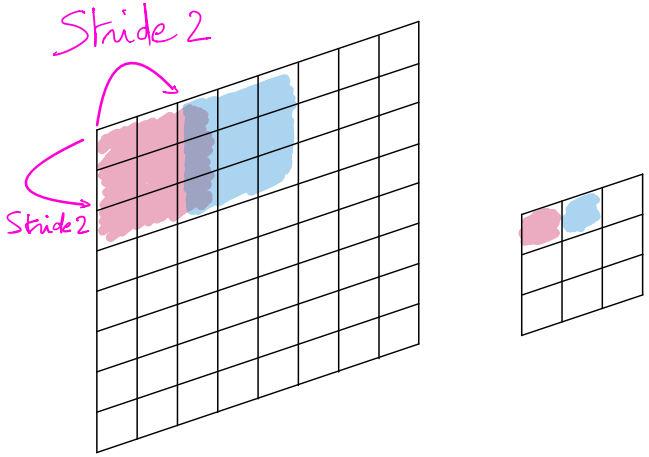


Parameters of convolutional layer 3/4

Four hyperparameters control the size of the output volume: the kernel size, the depth of the output volume, the stride and the zero-padding.

- The size of the kernel,
- The depth of the output volume,
- The stride, i.e., of how many pixels do we move the filter horizontally and vertically. Usually, stride is equal to one (rarely to two, and even more rarely larger).

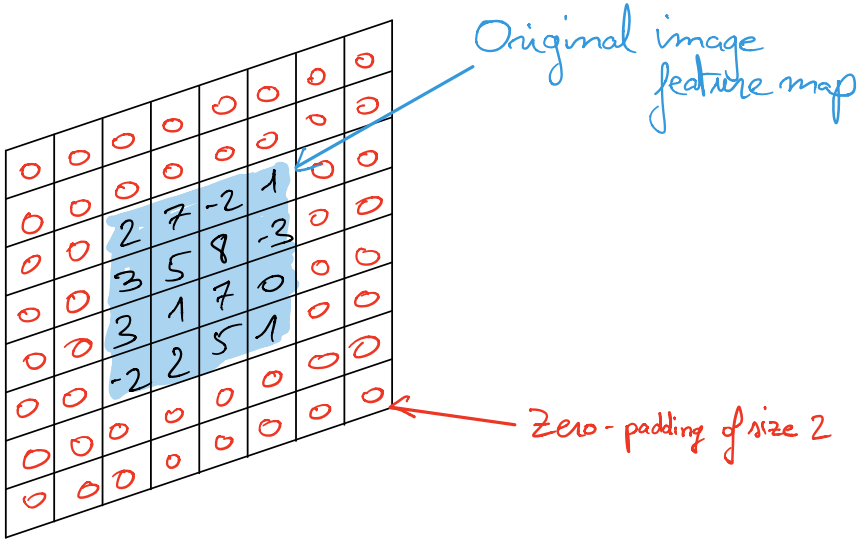




Parameters of convolutional layer 4/4

Four hyperparameters control the size of the output volume: the kernel size, the depth of the output volume, the stride and the zero-padding.

- The size of the kernel,
- The depth of the output volume,
- The stride,
- The size of the zero-padding, i.e. the number of zeros we add to the borders of the image. This can be used to obtain a constant image size between the input and the output.



How to choose zero-padding?

Let

- I the height/width of the input
- O the height/width of the output
- P the size of the zero-padding
- K the height/width of the filter
- S the stride

What is the relation between these quantities? How do we choose the zero-padding to obtain an output of the same size as the input?

How to choose zero-padding?

Let

- I the height/width of the input
- O the height/width of the output
- P the size of the zero-padding
- K the height/width of the filter
- S the stride

What is the relation between these quantities? How do we choose the zero-padding to obtain an output of the same size as the input?

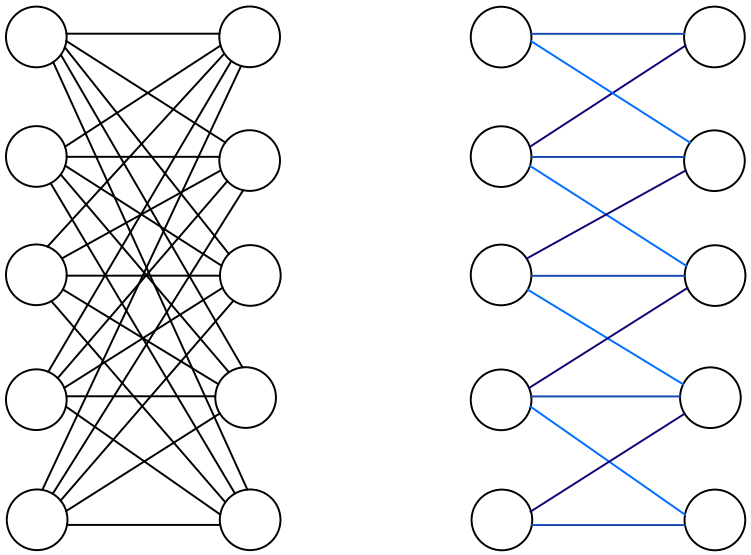
$$O = \left\lfloor \frac{2P + I - K}{S} \right\rfloor + 1$$

Why convolution?

- Same transformation applied to all parts of the image (takes into account the spatial dependence between pixels and object-shift invariance)
- Input image contains millions of pixel values, but we want to detect small meaningful features such as edges with kernels that use only few hundred of pixels
- When using a matrix product, all input and output units are connected, whereas convolution connects only output neurons with several pixels of the input image.

  Convolution involves weight sharing (a form of regularization) and requires less parameters which improves memory, is more statistically efficient and computationally faster.

Sparse connections

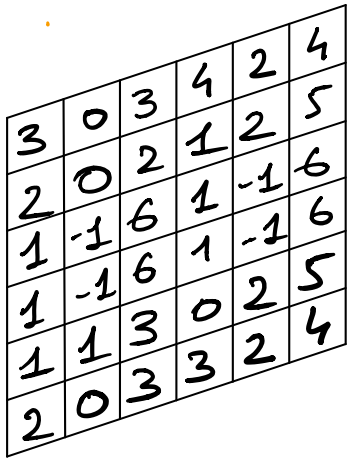


- Left: when using matrix multiplication, all outputs are connected to all inputs. We say that **connectivity is dense**
- Right: in a convolution with a kernel of width 3, only three outputs are affected by the input x. We say that the **connectivity is sparse**

Outline

Pooling

The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the max function.



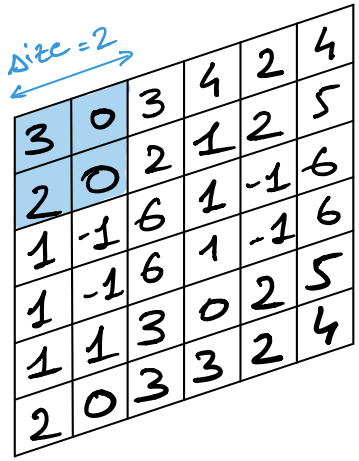
Parameters:

- Stride $S = 2$
- Spatial extend $F = 2$

Usually, $S = F = 2$ and more rarely $F = 3, S = 2$ (overlapping pooling).

Pooling

The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the max function.



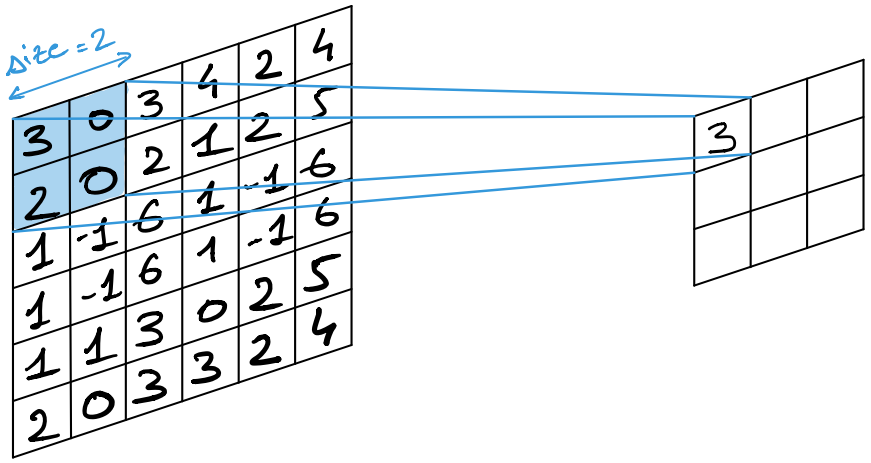
Parameters:

- Stride $S = 2$
- Spatial extend $F = 2$

Usually, $S = F = 2$ and more rarely $F = 3, S = 2$ (overlapping pooling).

Pooling

The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the max function.



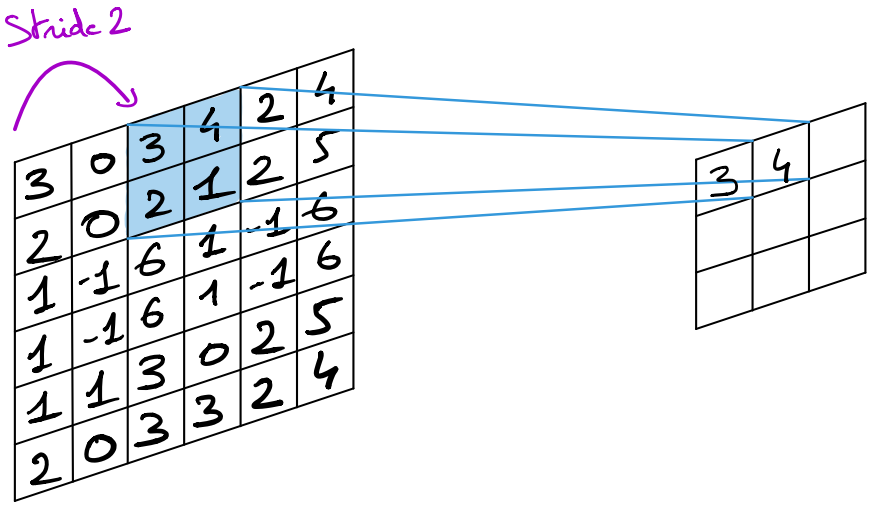
Parameters:

- Stride $S = 2$
- Spatial extend $F = 2$

Usually, $S = F = 2$ and more rarely $F = 3, S = 2$ (overlapping pooling).

Pooling

The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the max function.



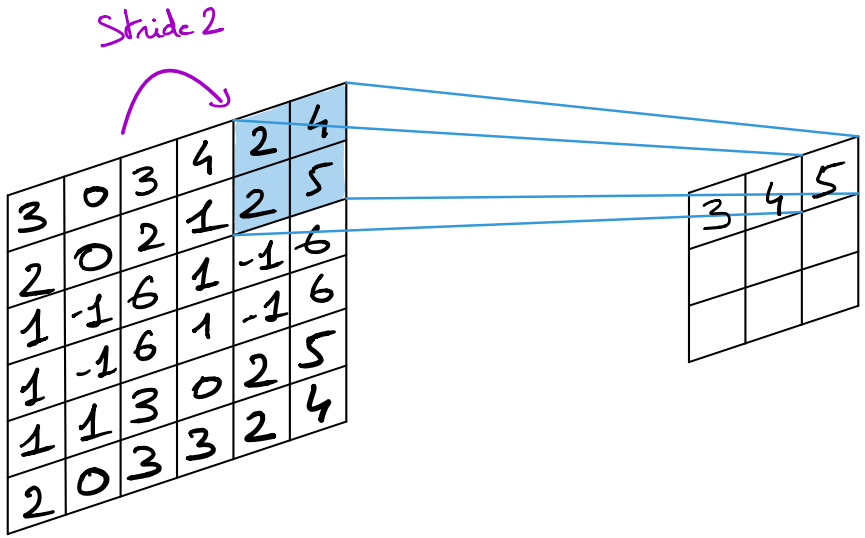
Parameters:

- Stride $S = 2$
- Spatial extend $F = 2$

Usually, $S = F = 2$ and more rarely $F = 3, S = 2$ (overlapping pooling).

Pooling

The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the max function.



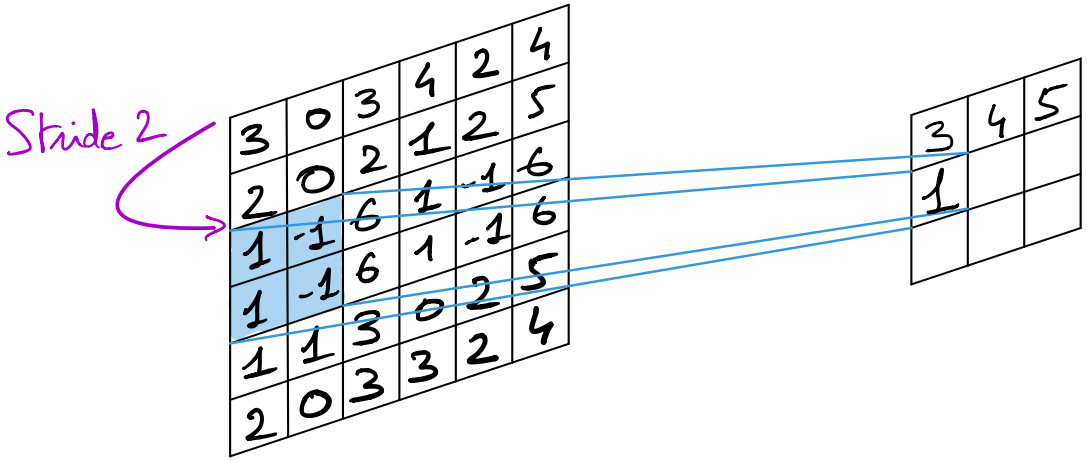
Parameters:

- Stride $S = 2$
- Spatial extend $F = 2$

Usually, $S = F = 2$ and more rarely $F = 3, S = 2$ (overlapping pooling).

Pooling

The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the max function.



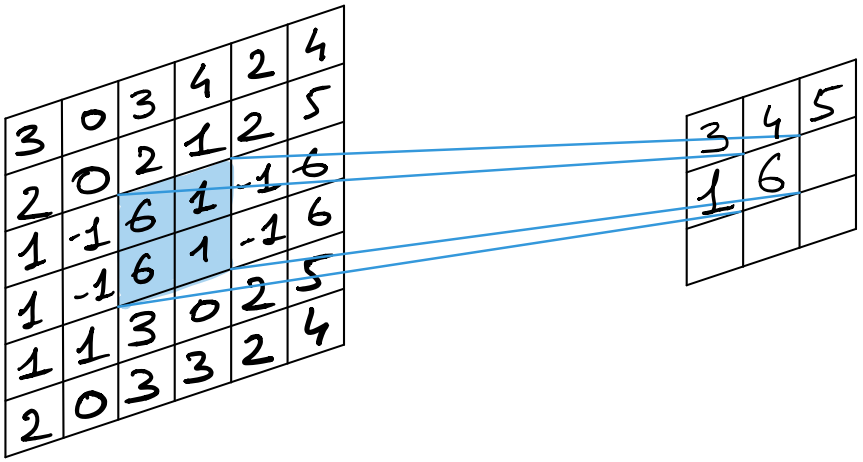
Parameters:

- Stride $S = 2$
- Spatial extend $F = 2$

Usually, $S = F = 2$ and more rarely $F = 3, S = 2$ (overlapping pooling).

Pooling

The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the max function.



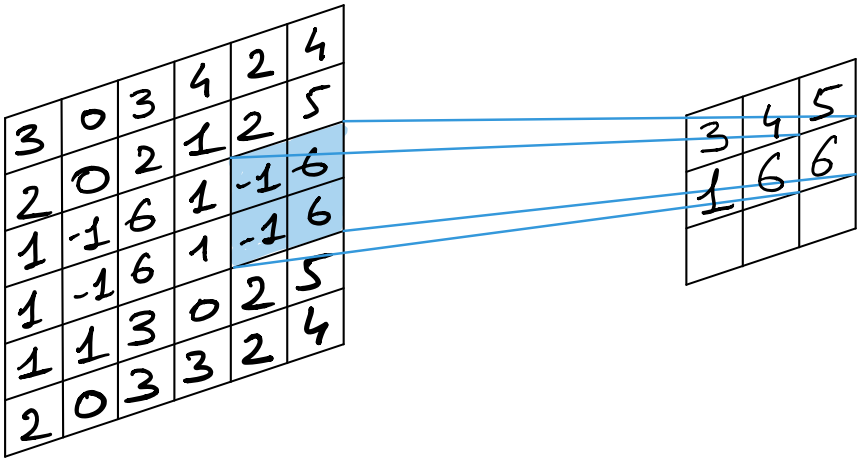
Parameters:

- Stride $S = 2$
- Spatial extend $F = 2$

Usually, $S = F = 2$ and more rarely $F = 3, S = 2$ (overlapping pooling).

Pooling

The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the max function.



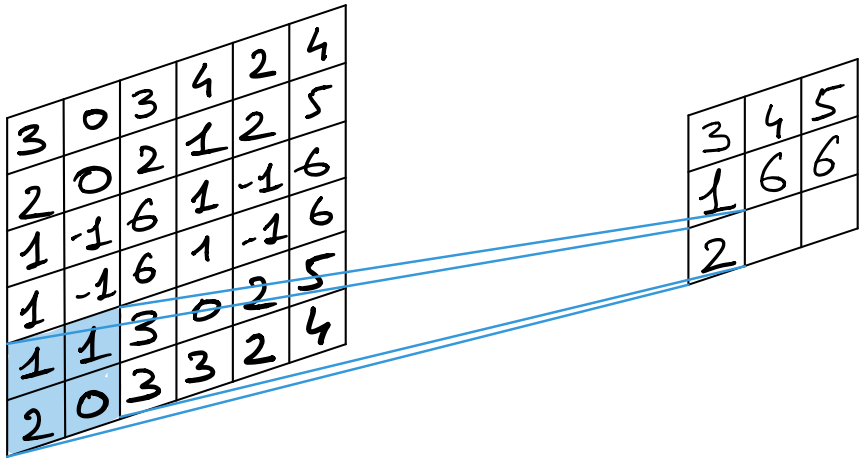
Parameters:

- Stride $S = 2$
- Spatial extend $F = 2$

Usually, $S = F = 2$ and more rarely $F = 3, S = 2$ (overlapping pooling).

Pooling

The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the max function.



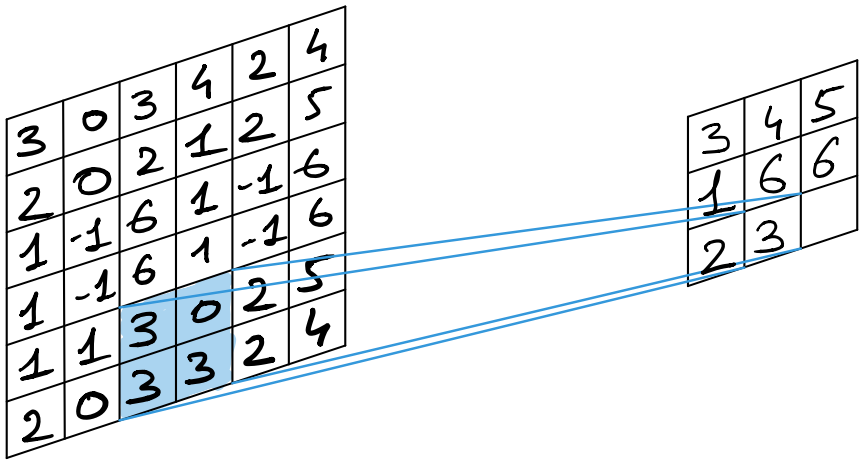
Parameters:

- Stride $S = 2$
- Spatial extend $F = 2$

Usually, $S = F = 2$ and more rarely $F = 3, S = 2$ (overlapping pooling).

Pooling

The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the max function.



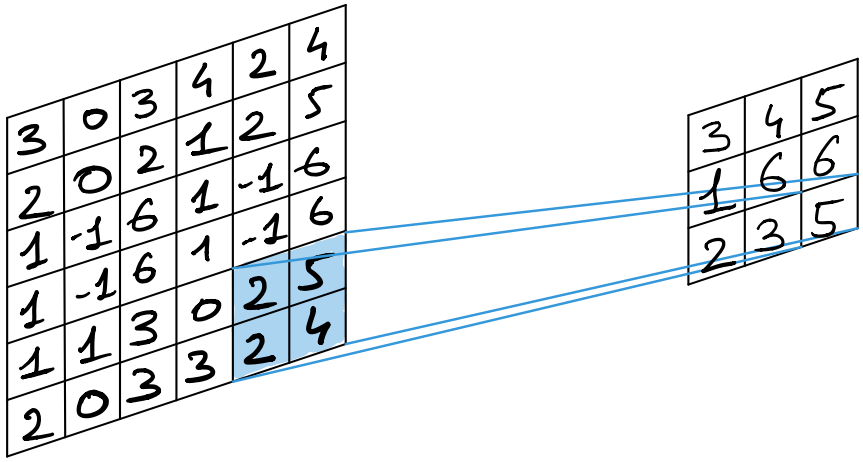
Parameters:

- Stride $S = 2$
- Spatial extend $F = 2$

Usually, $S = F = 2$ and more rarely $F = 3, S = 2$ (overlapping pooling).

Pooling

The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the max function.



Parameters:

- Stride $S = 2$
- Spatial extend $F = 2$

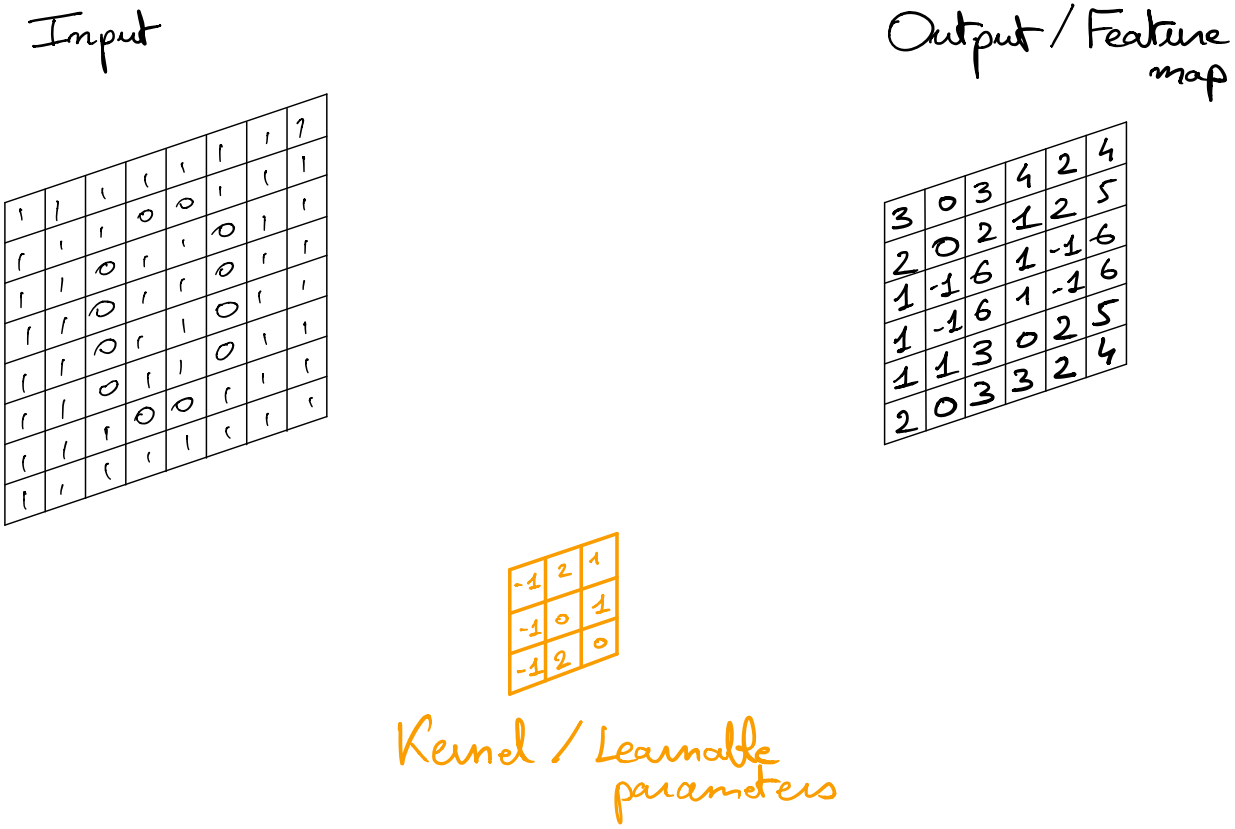
Usually, $S = F = 2$ and more rarely $F = 3, S = 2$ (overlapping pooling).

Pooling

- Pooling layers compute each pixel of the output as a summary statistic of neighboring input pixels at the corresponding location.
- The most widely used is the max aggregation, called max-pooling
- Pooling helps the representation to become approximately invariant to small translations of the input
- If a small translation is applied, output of the layer is almost unchanged
- Very useful if we care more about the presence of some feature than its position in the image: for face detection (presence of eyes is more important than where they are)
- Pooling also allows to handle inputs with different sizes: pictures can have different sizes, but the output classification layer must be of fixed size

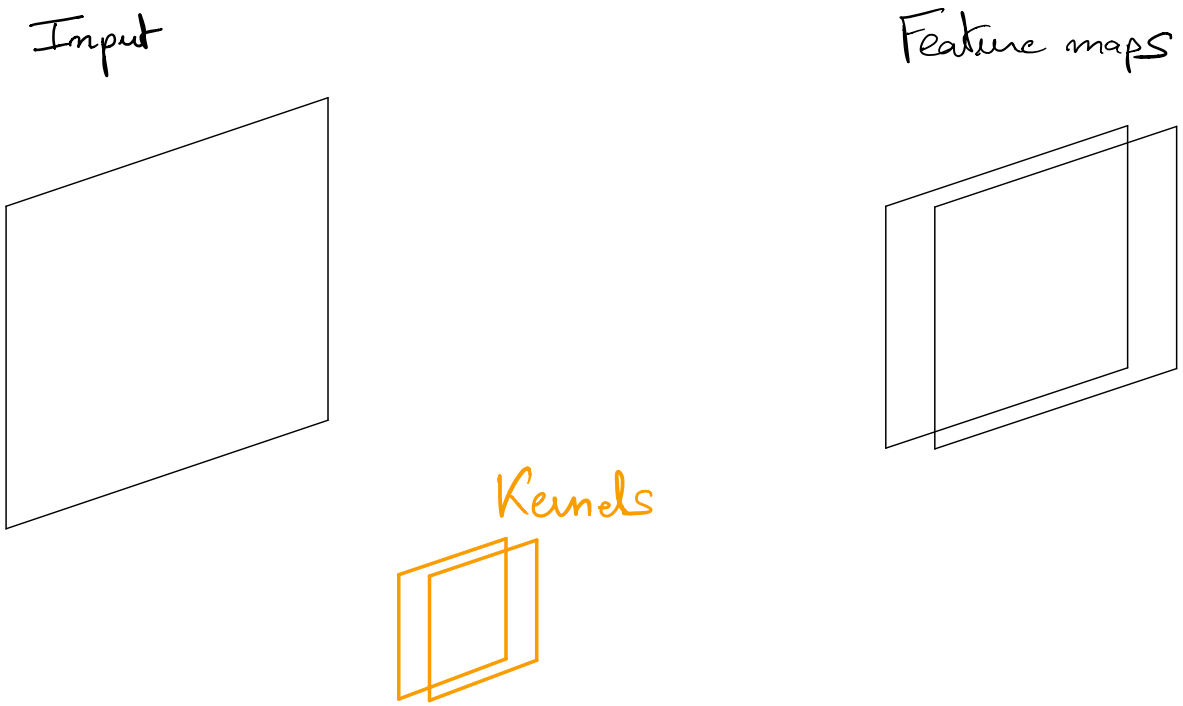
A possible architecture of a CNN

Consider a grayscale image. Each kernel of the first layer produces one feature map.



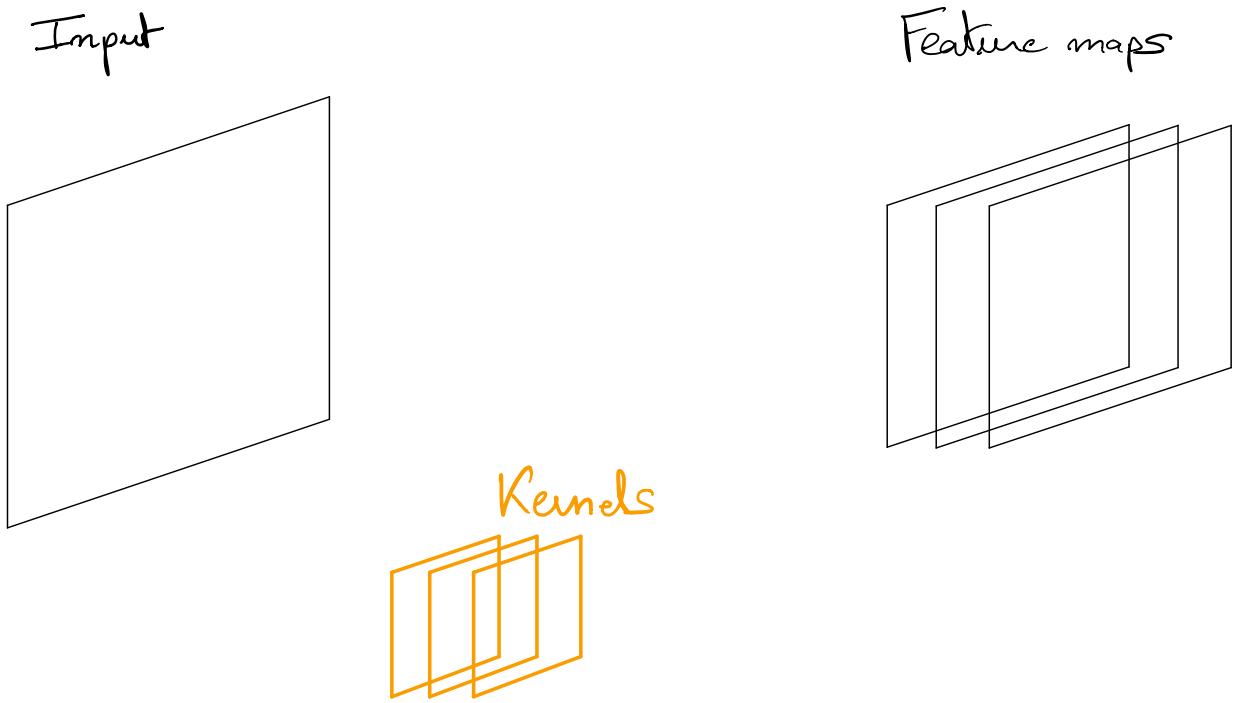
A possible architecture of a CNN

Consider a grayscale image. Each kernel of the first layer produces one feature map.



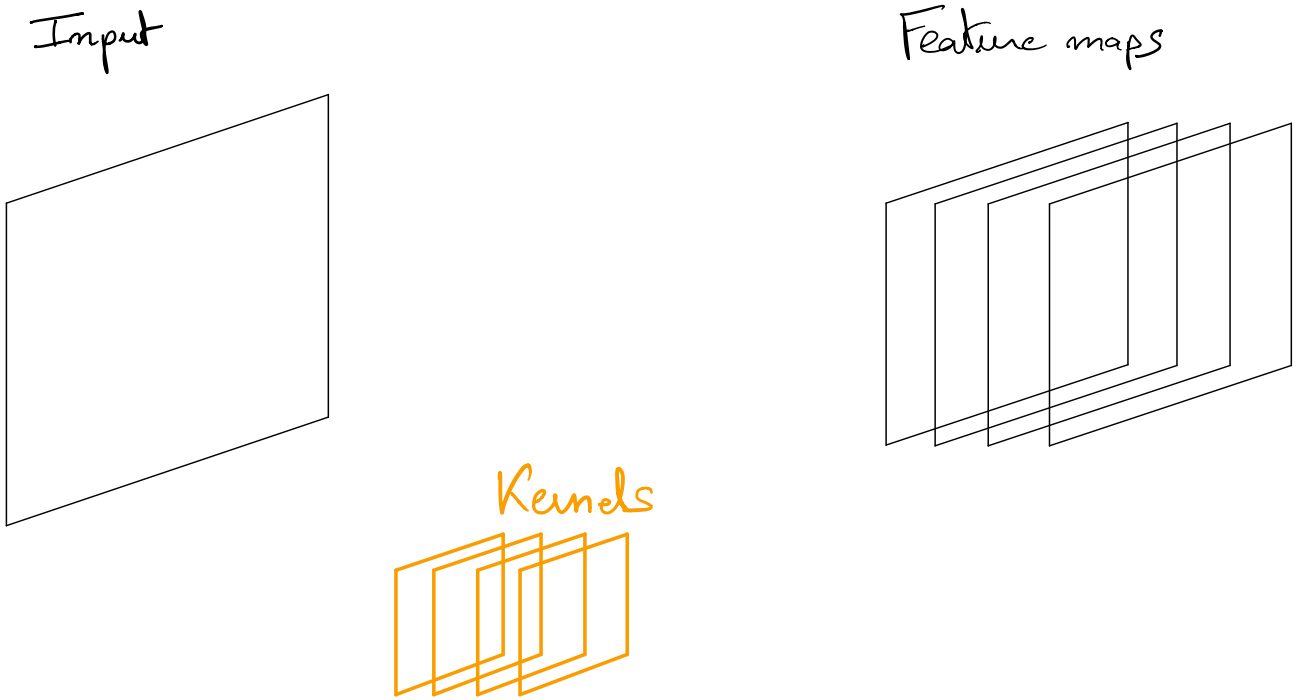
A possible architecture of a CNN

Consider a grayscale image. Each kernel of the first layer produces one feature map.



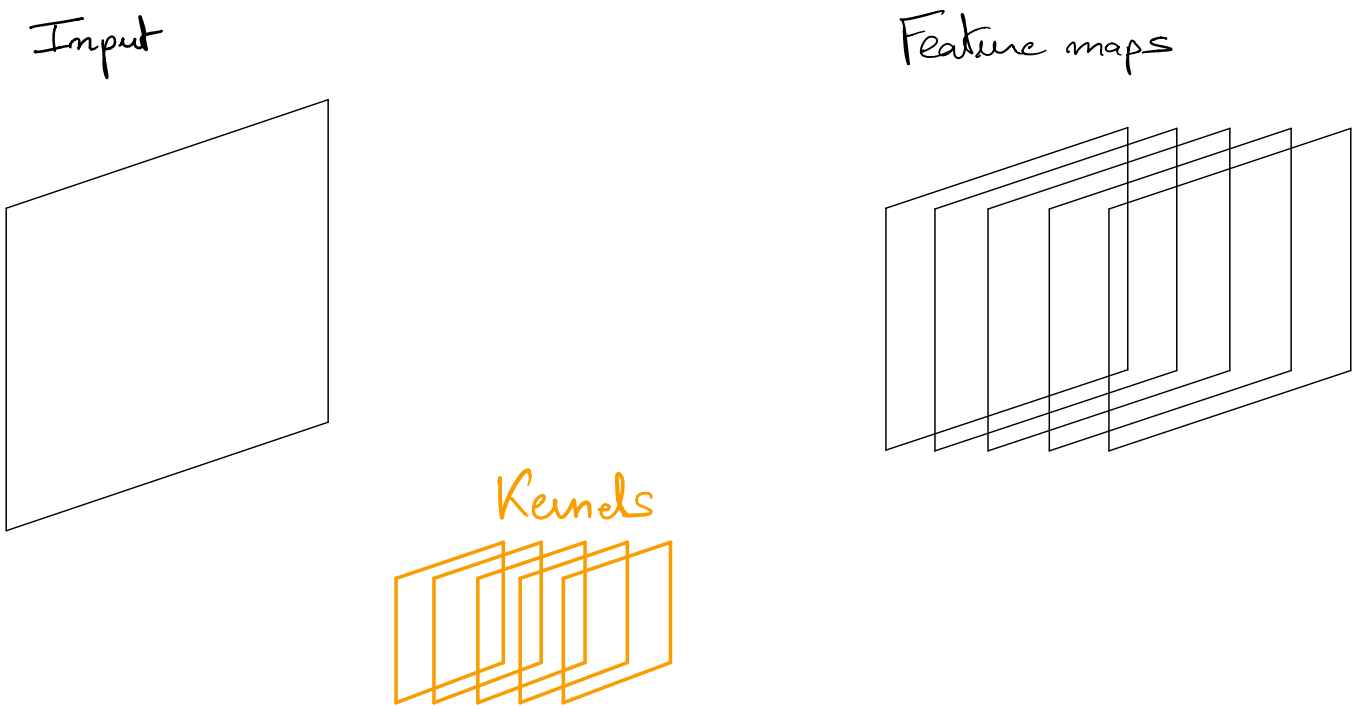
A possible architecture of a CNN

Consider a grayscale image. Each kernel of the first layer produces one feature map.



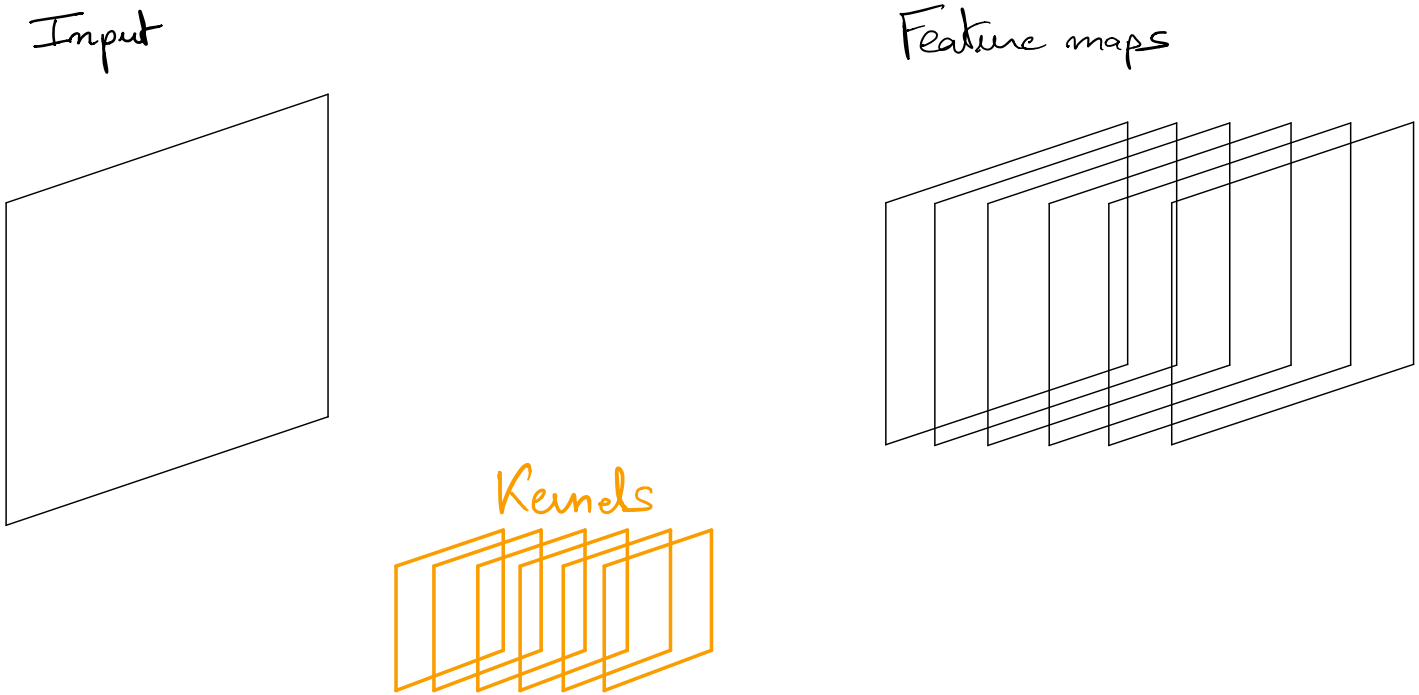
A possible architecture of a CNN

Consider a grayscale image. Each kernel of the first layer produces one feature map.



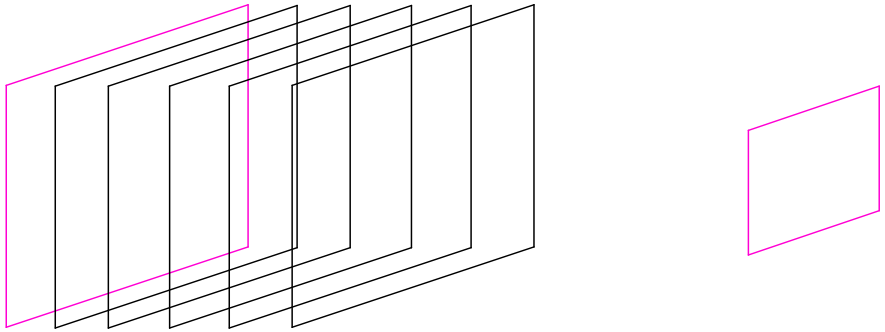
A possible architecture of a CNN

Consider a grayscale image. Each kernel of the first layer produces one feature map.



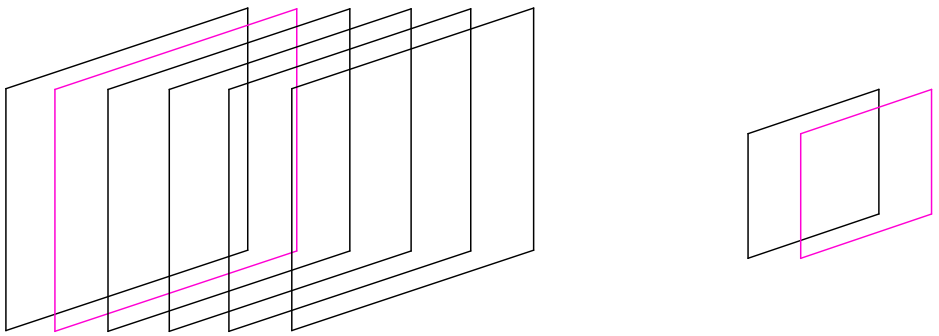
A possible architecture of a CNN

The pooling layer operates on each feature map separately.



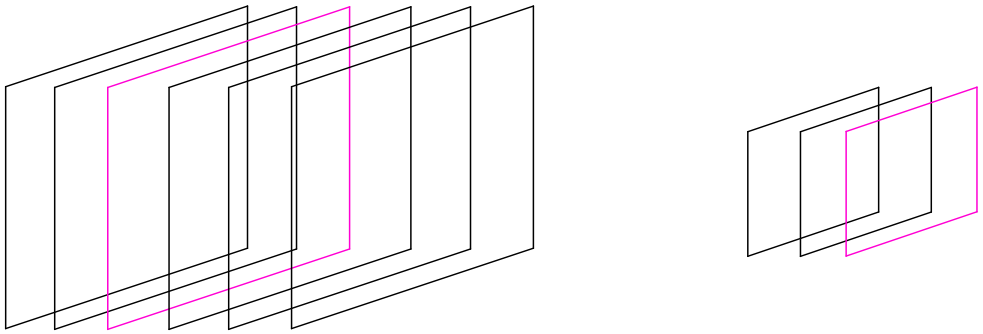
A possible architecture of a CNN

The pooling layer operates on each feature map separately.



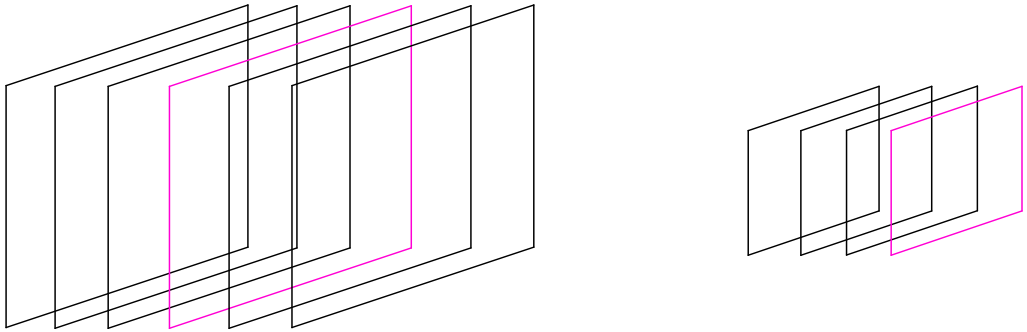
A possible architecture of a CNN

The pooling layer operates on each feature map separately.



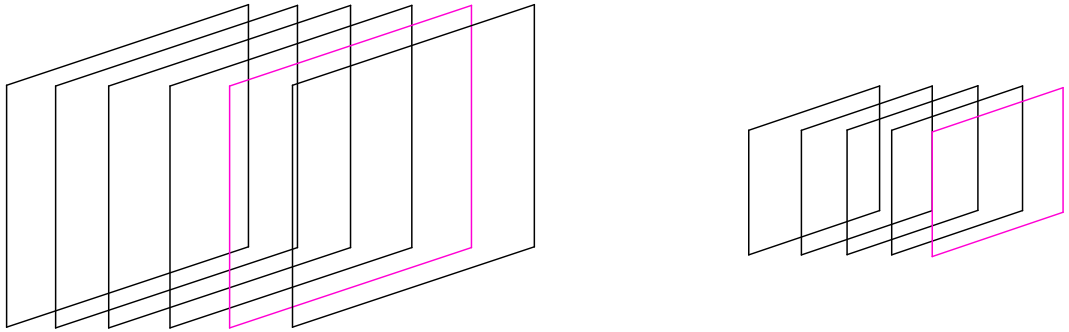
A possible architecture of a CNN

The pooling layer operates on each feature map separately.



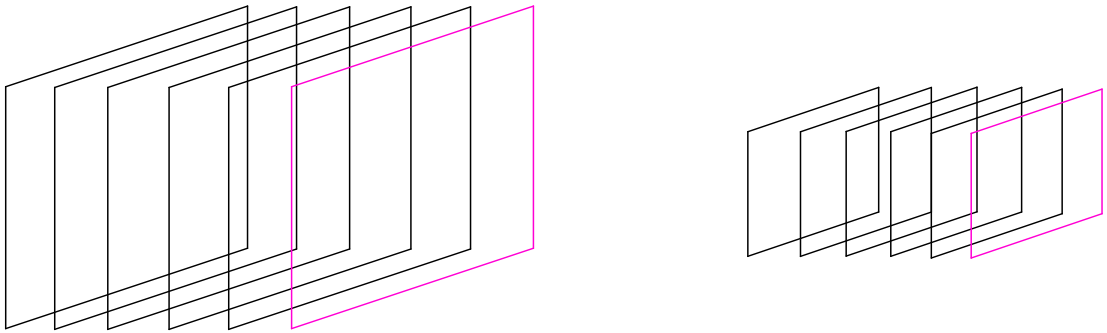
A possible architecture of a CNN

The pooling layer operates on each feature map separately.

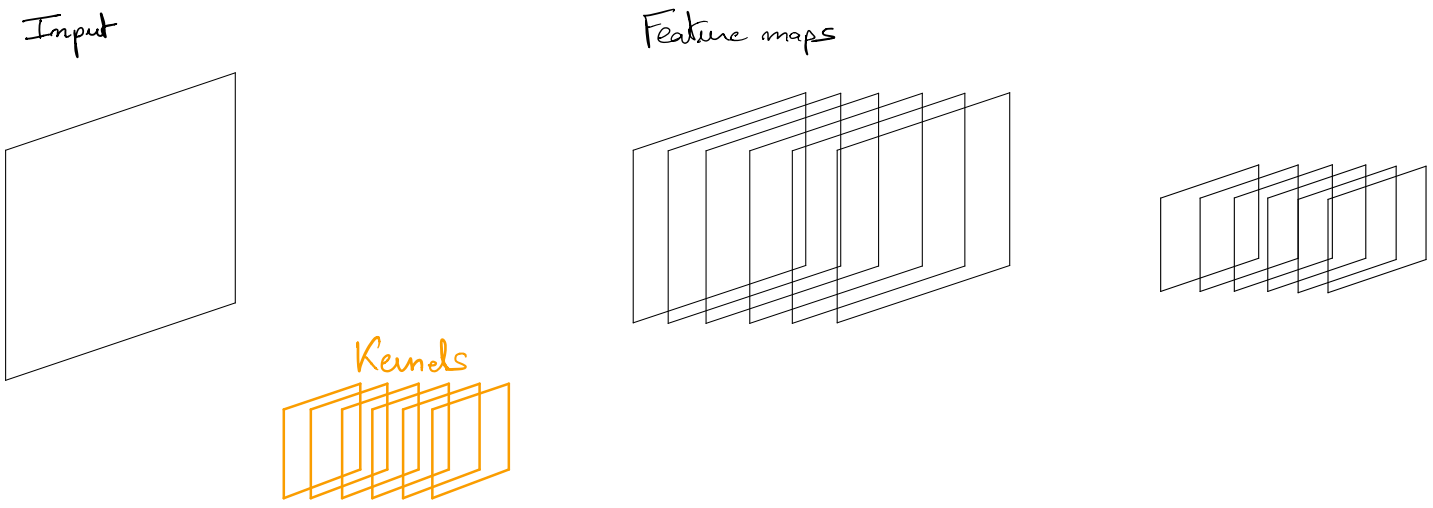


A possible architecture of a CNN

The pooling layer operates on each feature map separately.

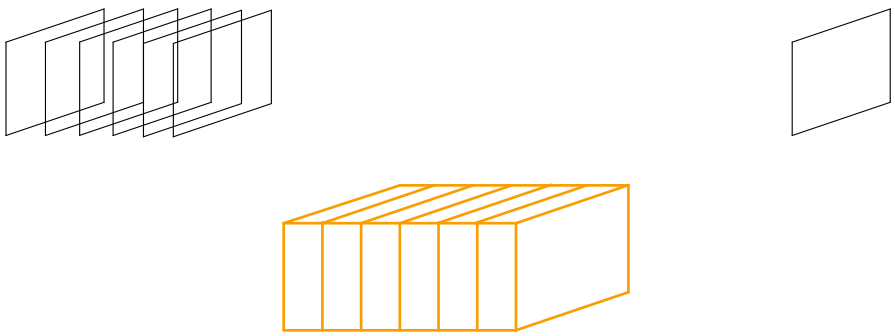


A possible architecture of a CNN



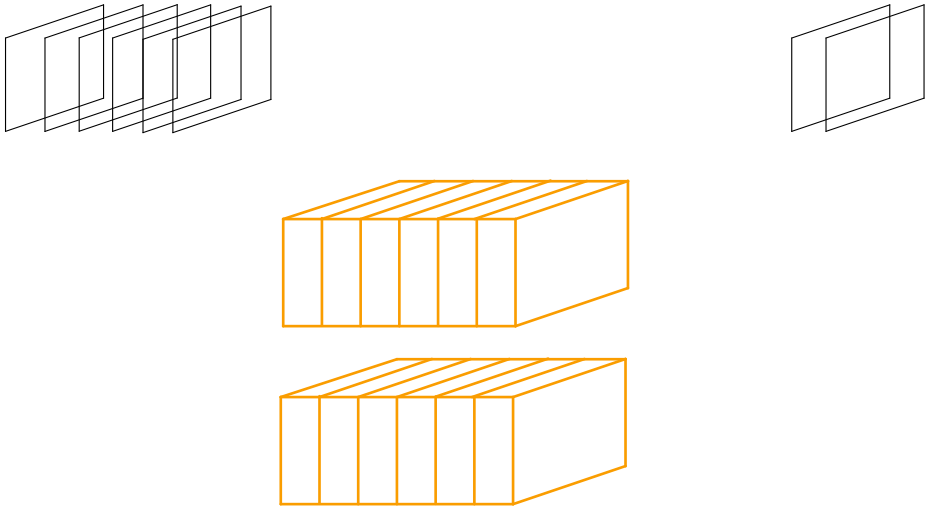
A possible architecture of a CNN

A convolutional layer operates on the feature maps output by the pooling layer. Each kernel is a volume whose depth equals the depth of the input volume.



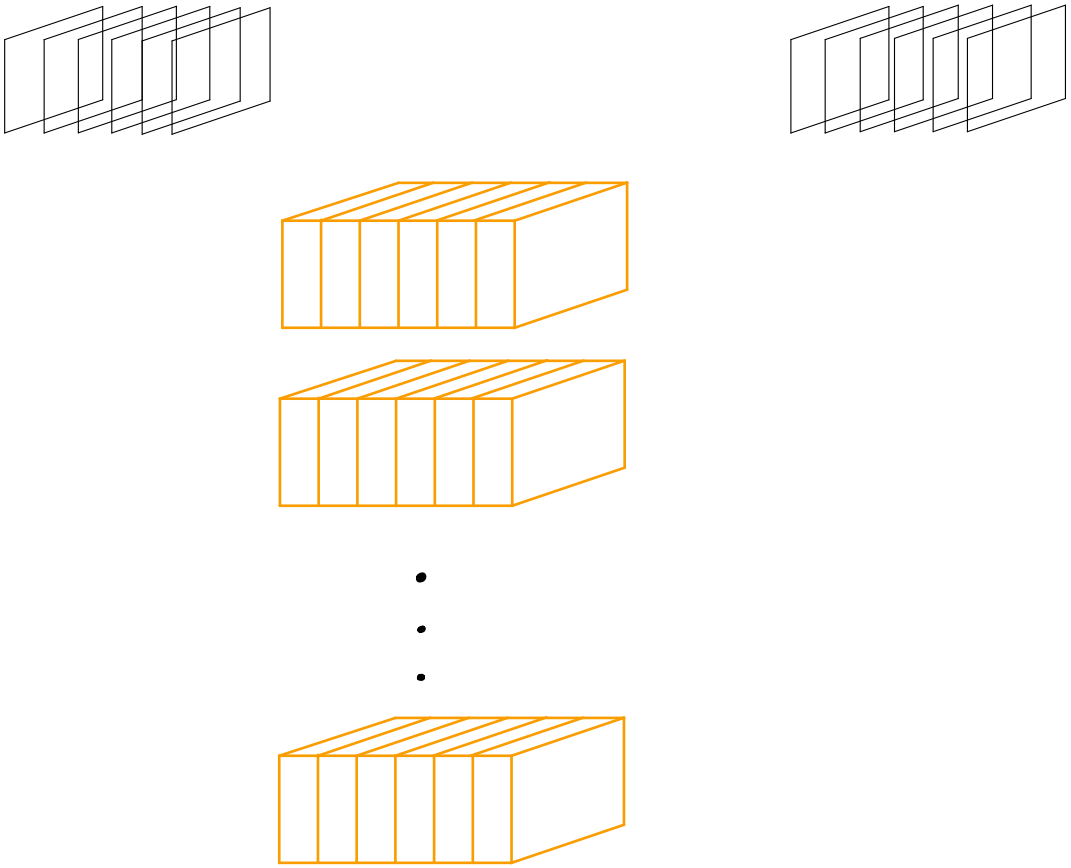
A possible architecture of a CNN

A convolutional layer operates on the feature maps output by the pooling layer. Each kernel is a volume whose depth equals the depth of the input volume.

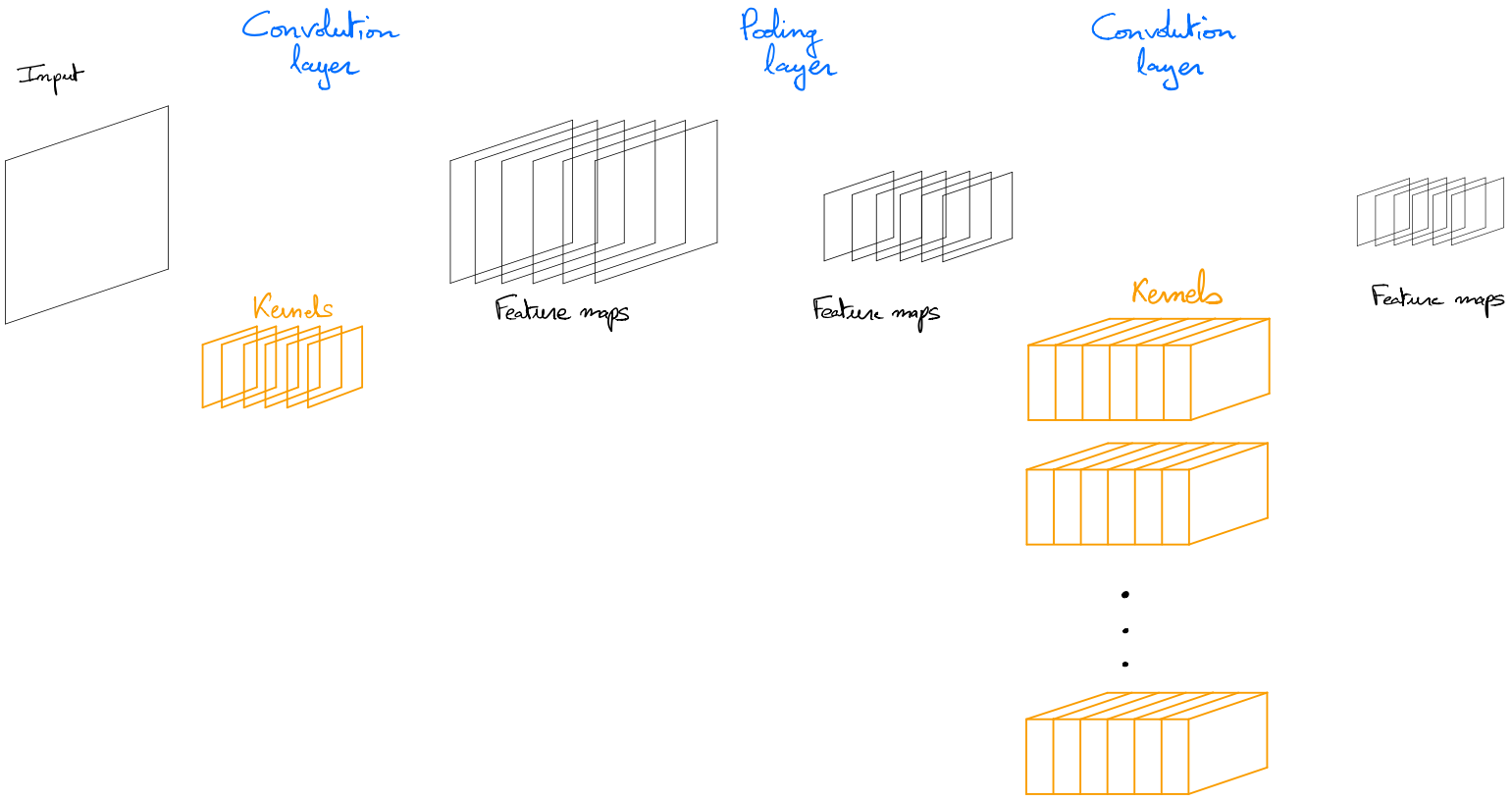


A possible architecture of a CNN

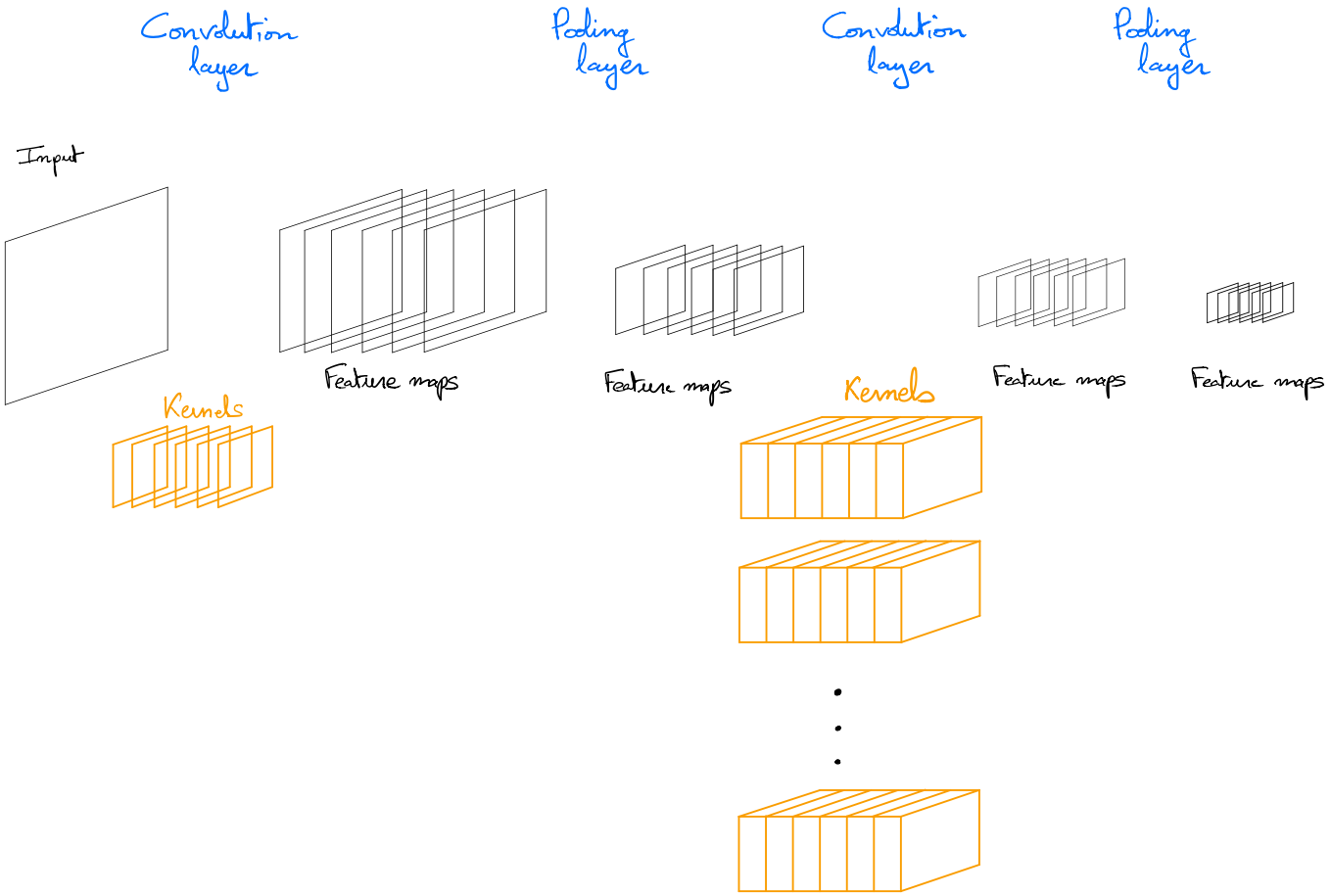
A convolutional layer operates on the feature maps output by the pooling layer. Each kernel is a volume whose depth equals the depth of the input volume.



A possible architecture of a CNN

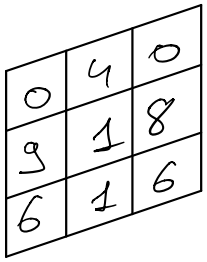


A possible architecture of a CNN



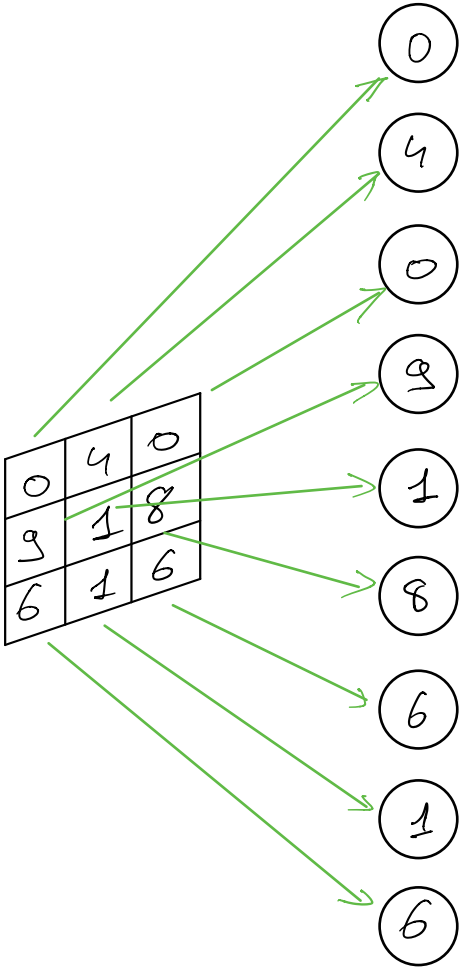
A possible architecture of a CNN

At the end of the network, the feature maps are flattened in order to apply a classic neural networks.



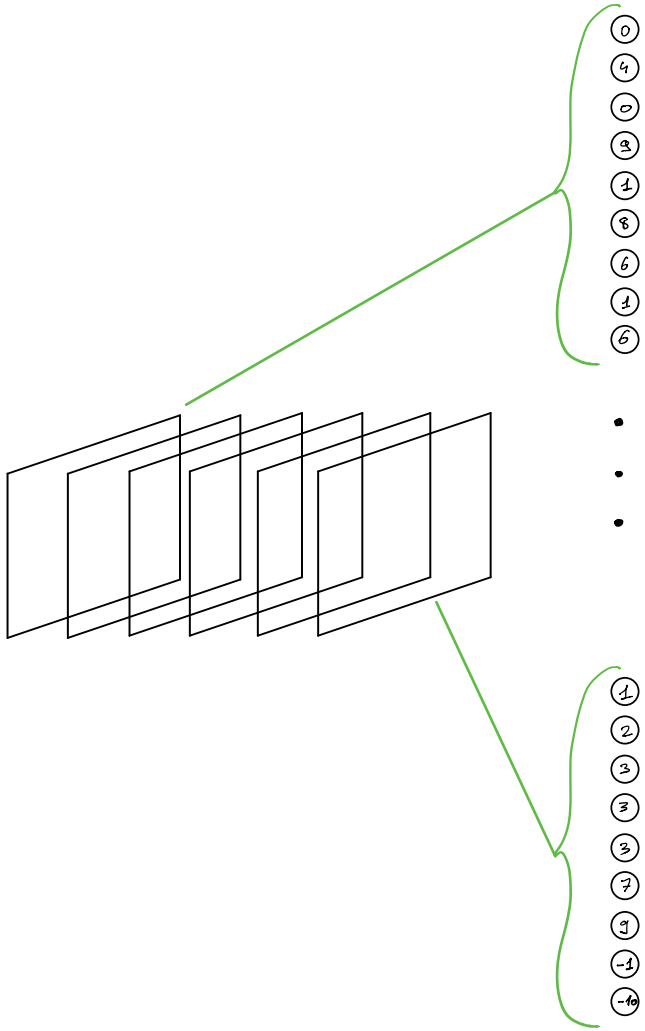
A possible architecture of a CNN

At the end of the network, the feature maps are flattened in order to apply a classic neural networks.



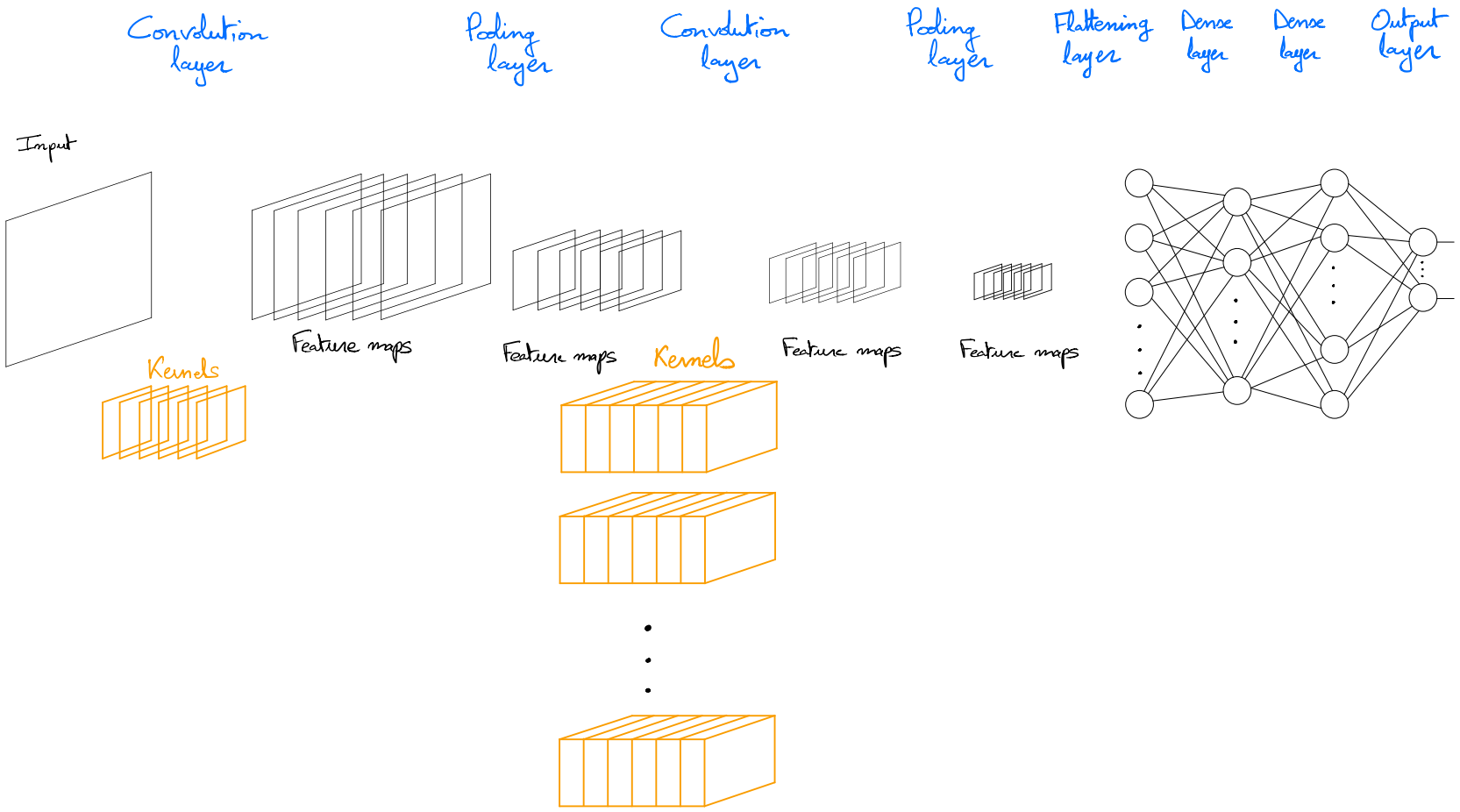
A possible architecture of a CNN

At the end of the network, the feature maps are flattened in order to apply a classic neural networks.



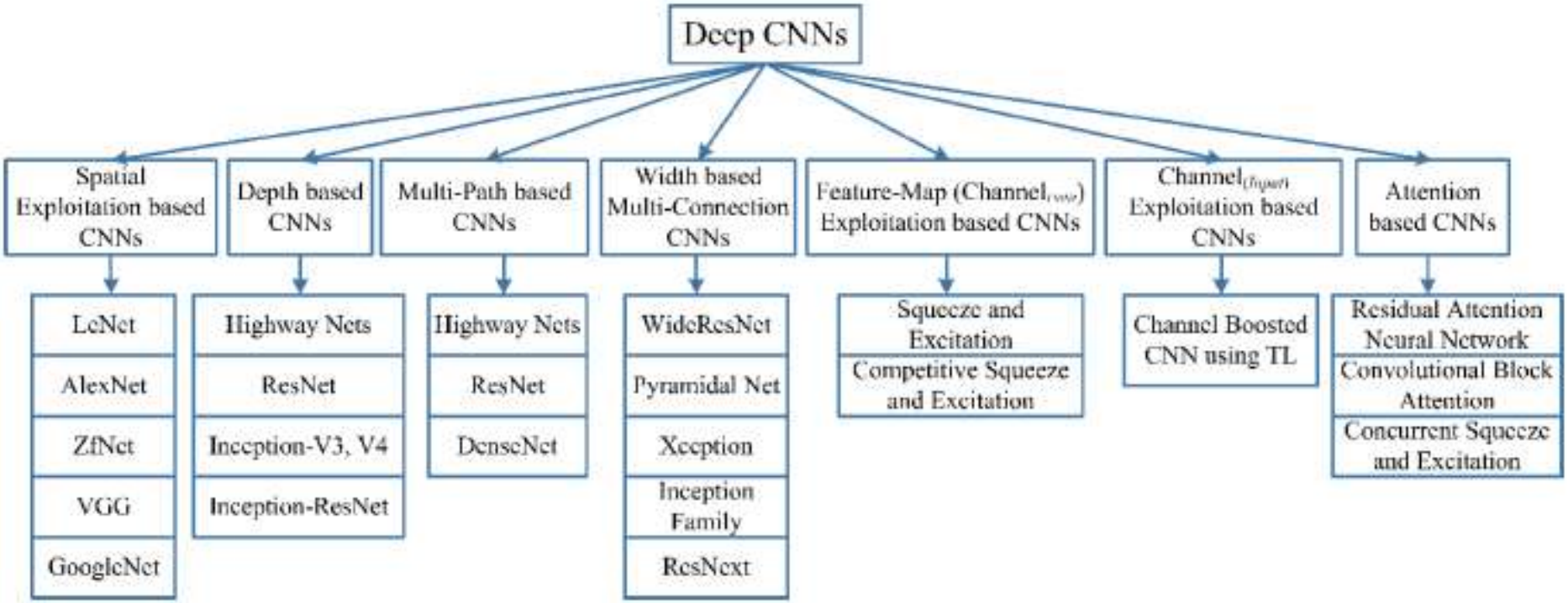
A possible architecture of a CNN

The full architecture is summarized below.



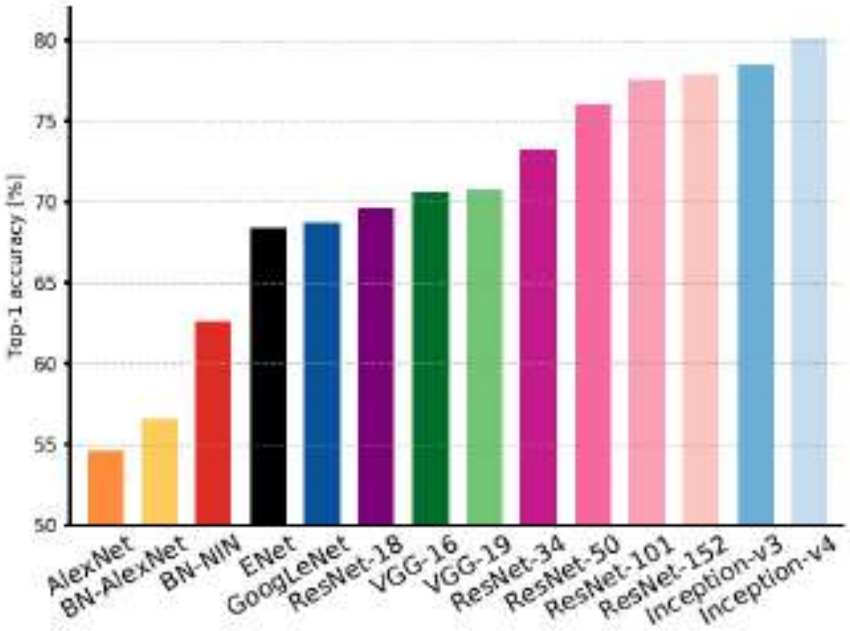
Outline

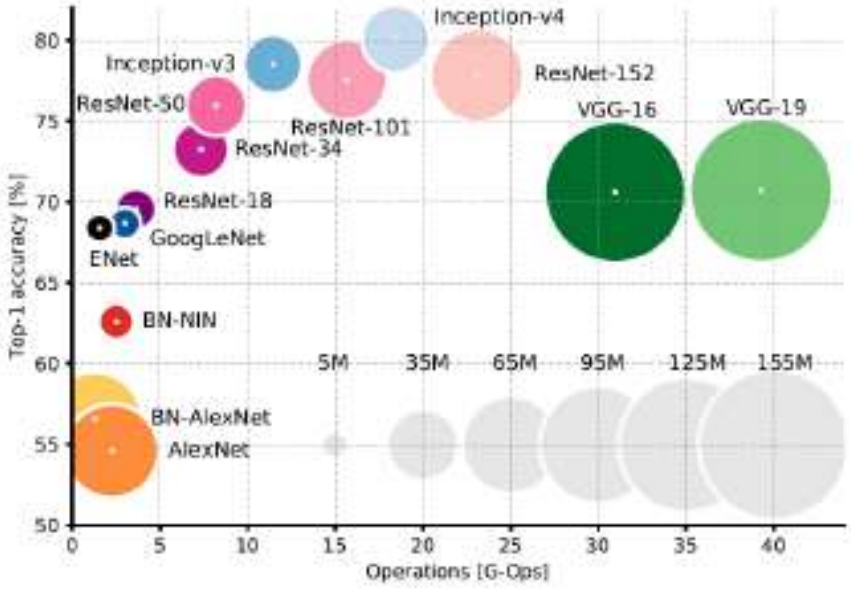
CNN Taxonomy



See this very detailed review paper ["A survey of the recent architectures of deep convolutional neural networks", Khan et al. 2020]

Comparison of several CNN





["An analysis of deep neural network models for practical applications", Canziani et al. 2016]

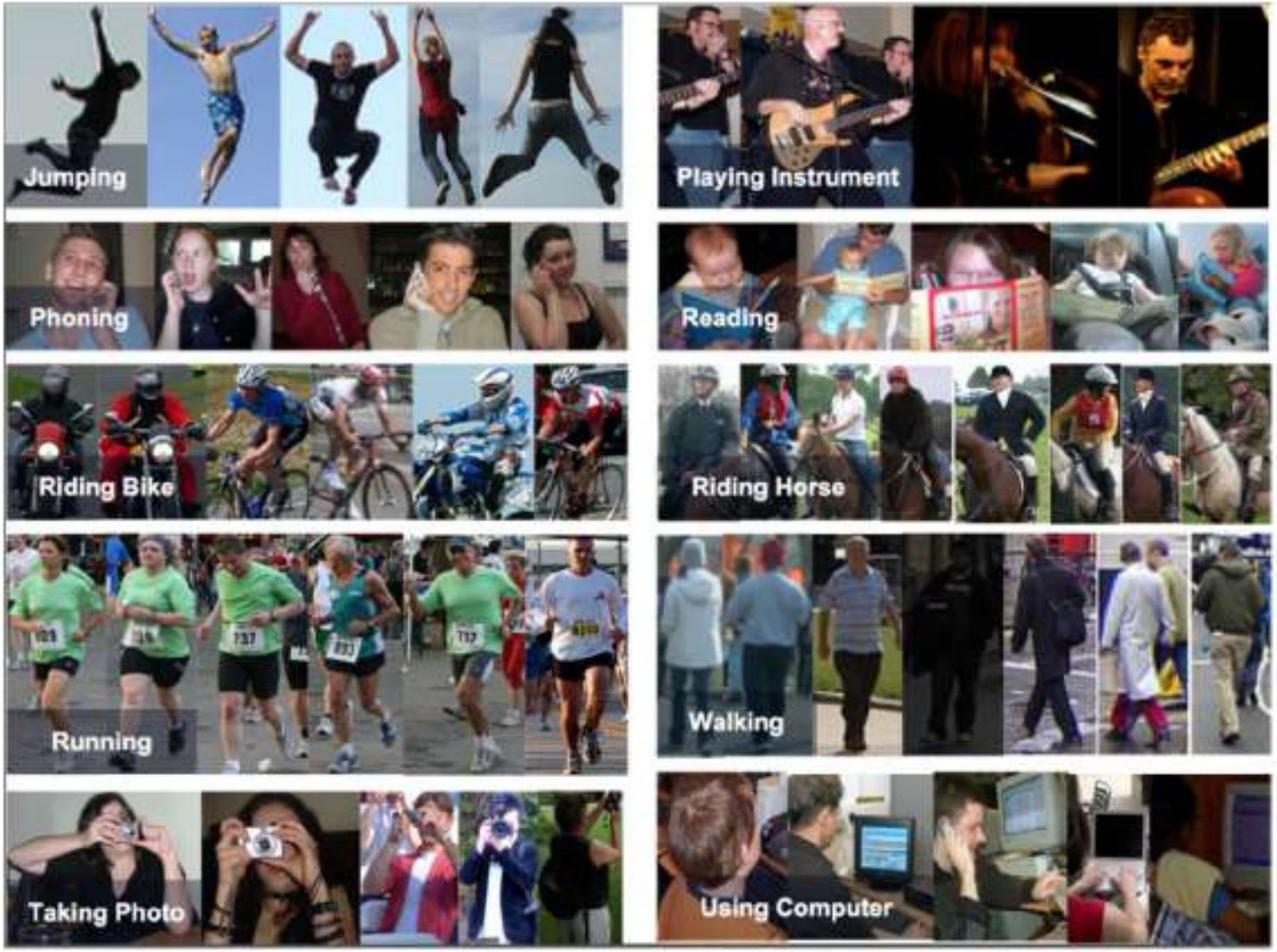
Outline

Pose estimation - Deeppose

["Deeppose: Human pose estimation via deep neural networks", Toshev and Szegedy 2014]

Action recognition

["Actions and attributes from wholes and parts", Gkioxari et al. 2015]



Object detection - Exhaustive search vs segmentation

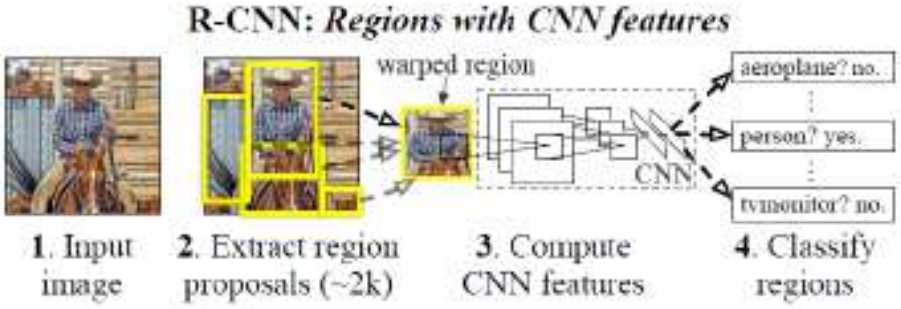
Bottom-up grouping generates hierarchical nested partitioning of the input image.

["Mean shift: A robust approach toward feature space analysis"; "Efficient graph-based image segmentation", Comaniciu and Meer 2002; Felzenszwalb and Huttenlocher 2004]

Object detection - R-CNN - Regions with CNN features

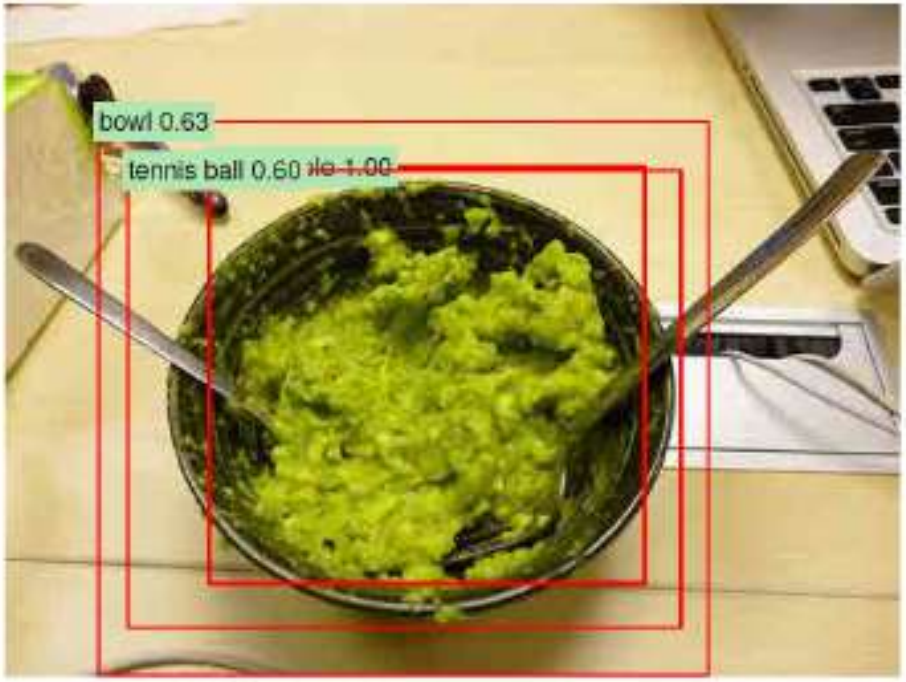
One of the most famous object proposal based CNN detector is Region-based CNN (R-CNN) by Girshick, Jeff Donahue, et al. 2014, aiming at

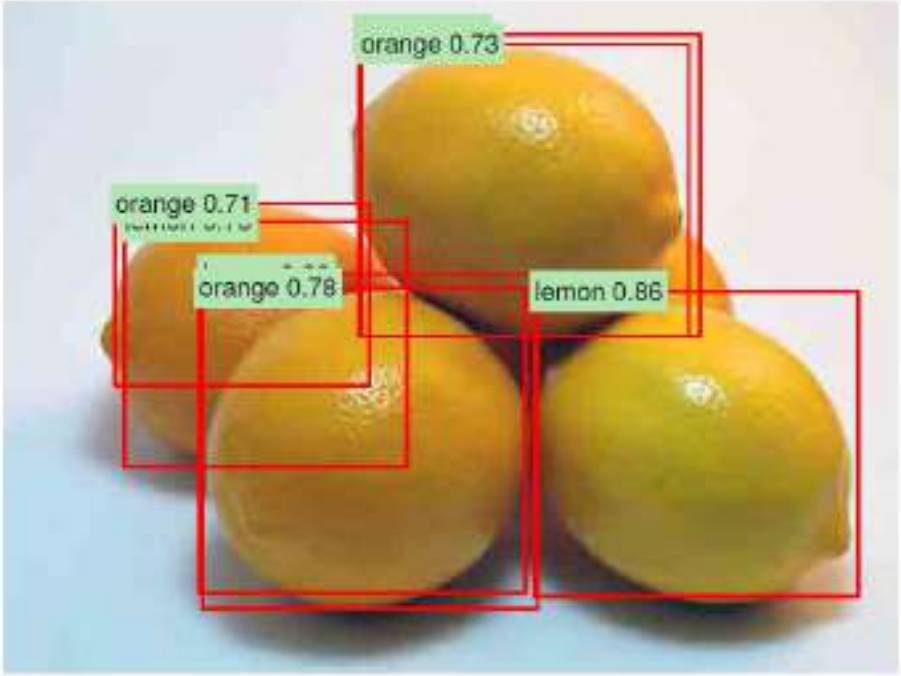
- localizing objects with a deep network
- training a high-capacity model with only a small quantity of annotated detection data



1. Generating category-independent region proposals via selective search.
2. Training large CNN that extracts a fixed-length feature vector from each region (Supervised pre-training on the large auxiliary dataset ILSVRC, followed by domainspecific fine-tuning on the small dataset PASCAL).
3. Learning a set of class- specific linear SVMs.

Object detection - R-CNN - Regions with CNN features





YOLO

["You only look once: Unified, real-time object detection", Redmon et al. 2016]

The whole detection pipeline is a single network which predicts bounding boxes and class probabilities from the full image in one evaluation, and can be optimized end-to-end directly on detection performance.

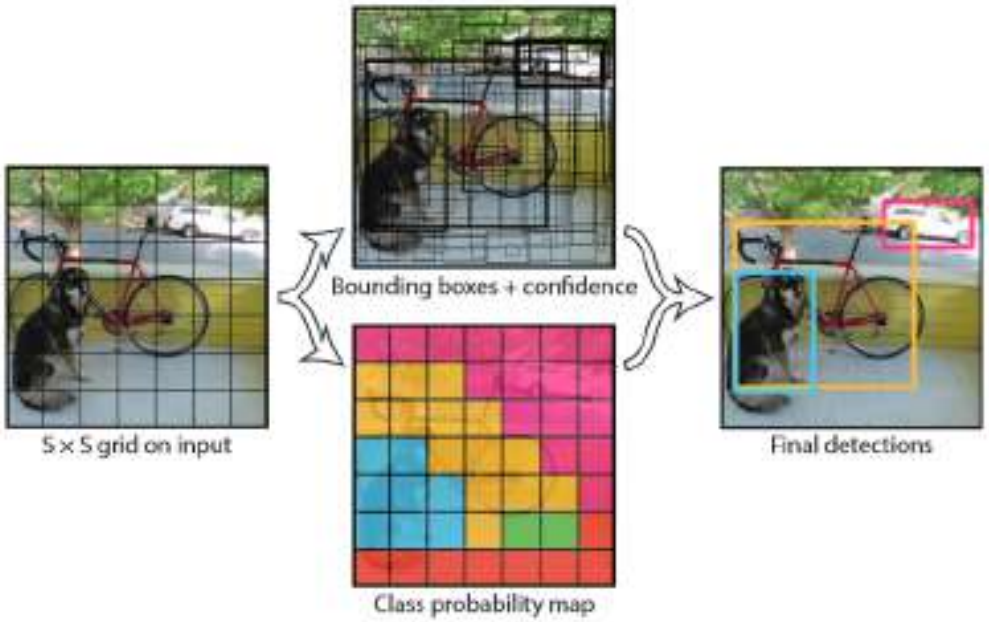
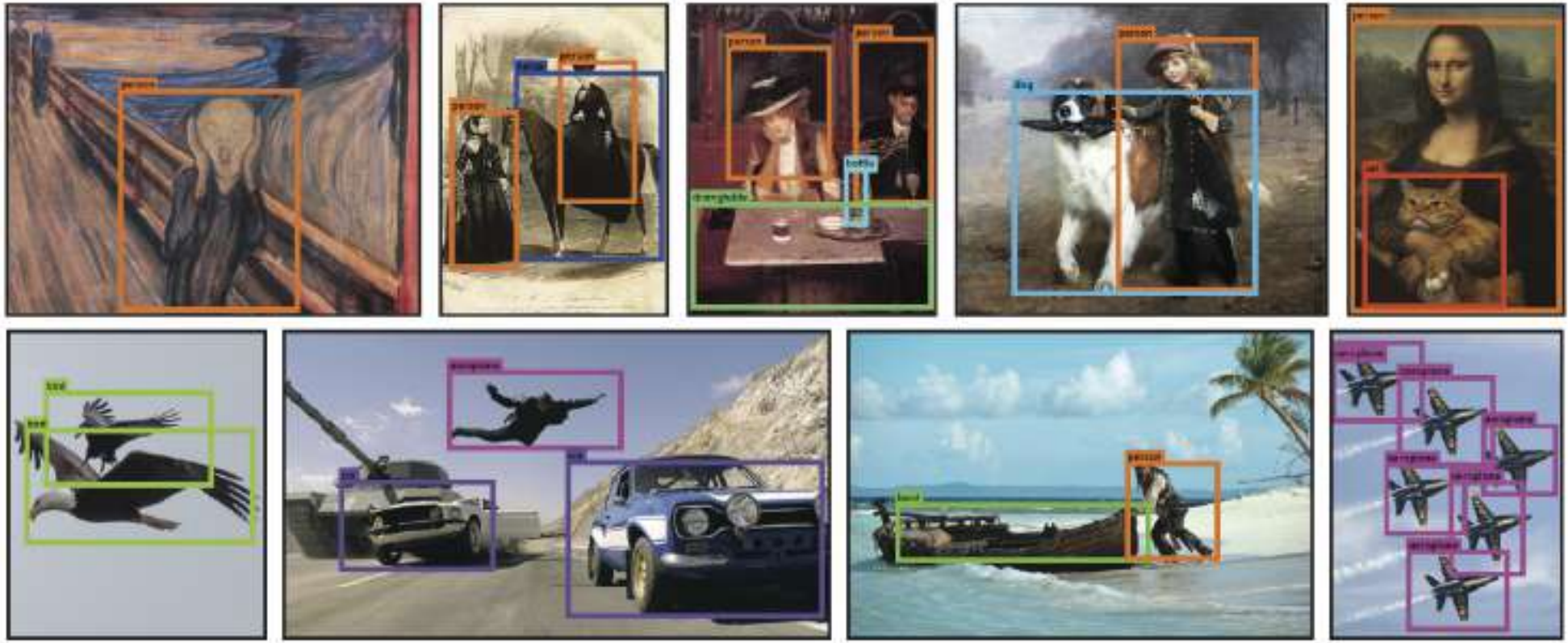


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

Drawback

Fails to detect small numerous objects.

YOLO



Videos

Object tracking They propose a target-specific CNN for object tracking, where the CNN is trained incrementally during tracking with new examples obtained online. They employ a candidate pool of multiple CNNs as a data-driven model of different instances of the target object.

["Deeptrack: Learning discriminative feature representations online for robust visual tracking", Li et al. 2016]

https://pjreddie.com/darknet/yolo/

Pose/Action recognition They use the two stream CNN (spatial /temporal) on the localized parts of the human body and show the aggregation of part-based local CNN descriptors can effectively improve the performance of action recognition.

["P-cnn: Pose-based cnn features for action recognition", Chéron et al. 2015]

["End-to-end learning of deformable mixture of parts and deep convolutional neural networks for human pose estimation", W. Yang et al. 2016]

https://www.youtube.com/watch?v=MKVvQK8FawE

["Segnet: A deep convolutional encoder-decoder architecture for image segmentation", Badrinarayanan et al. 2015]

https://www.youtube.com/watch?v=CxanE_W46ts

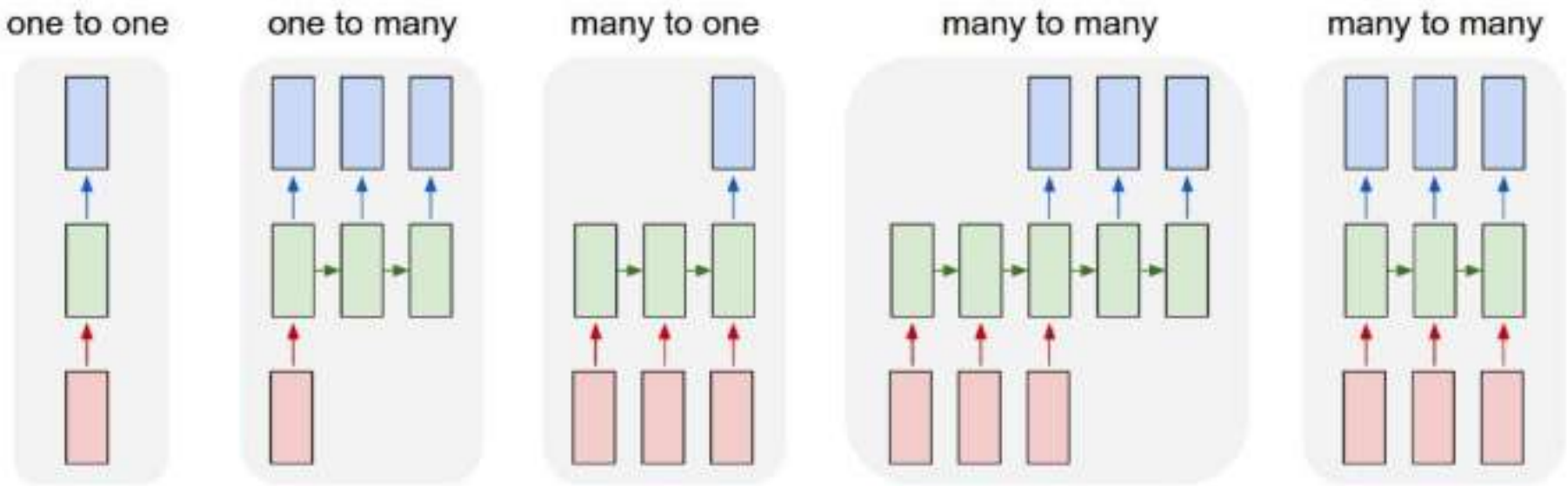
["Realtime multi-person 2d pose estimation using part affinity fields", Cao et al. 2016]

https://www.youtube.com/watch?v=pW6nZXeWlGM

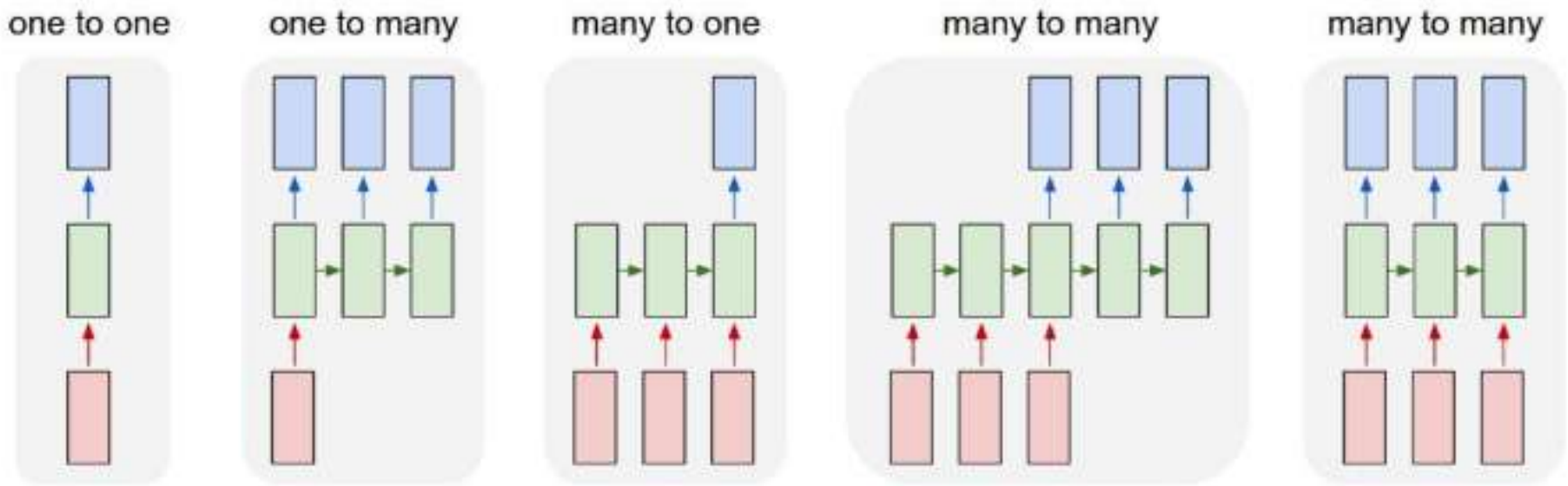
Outline

Outline

RNNs offer a lot of variability

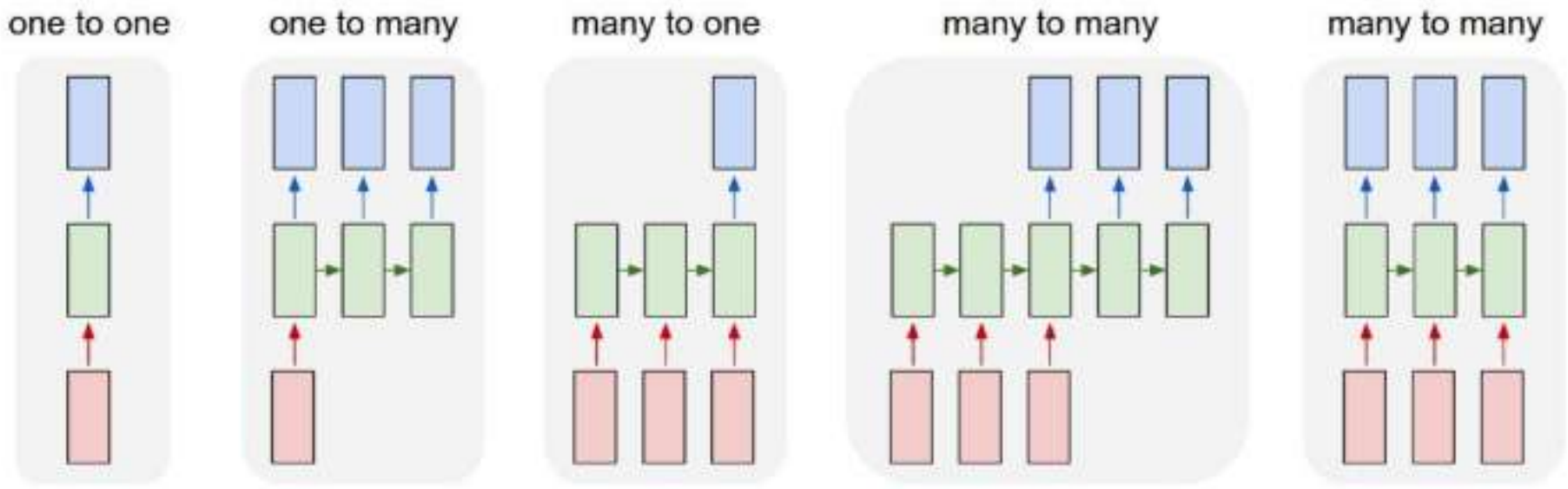


RNNs offer a lot of variability



- Vanilla Neural Networks
- Image Captioning: image/sequence of words
- Sentiment classification: sequence of words/sentiment
- Translation: sequence of words/sequence of words
- Video classification on frame level: sequence of images/sequence of labels

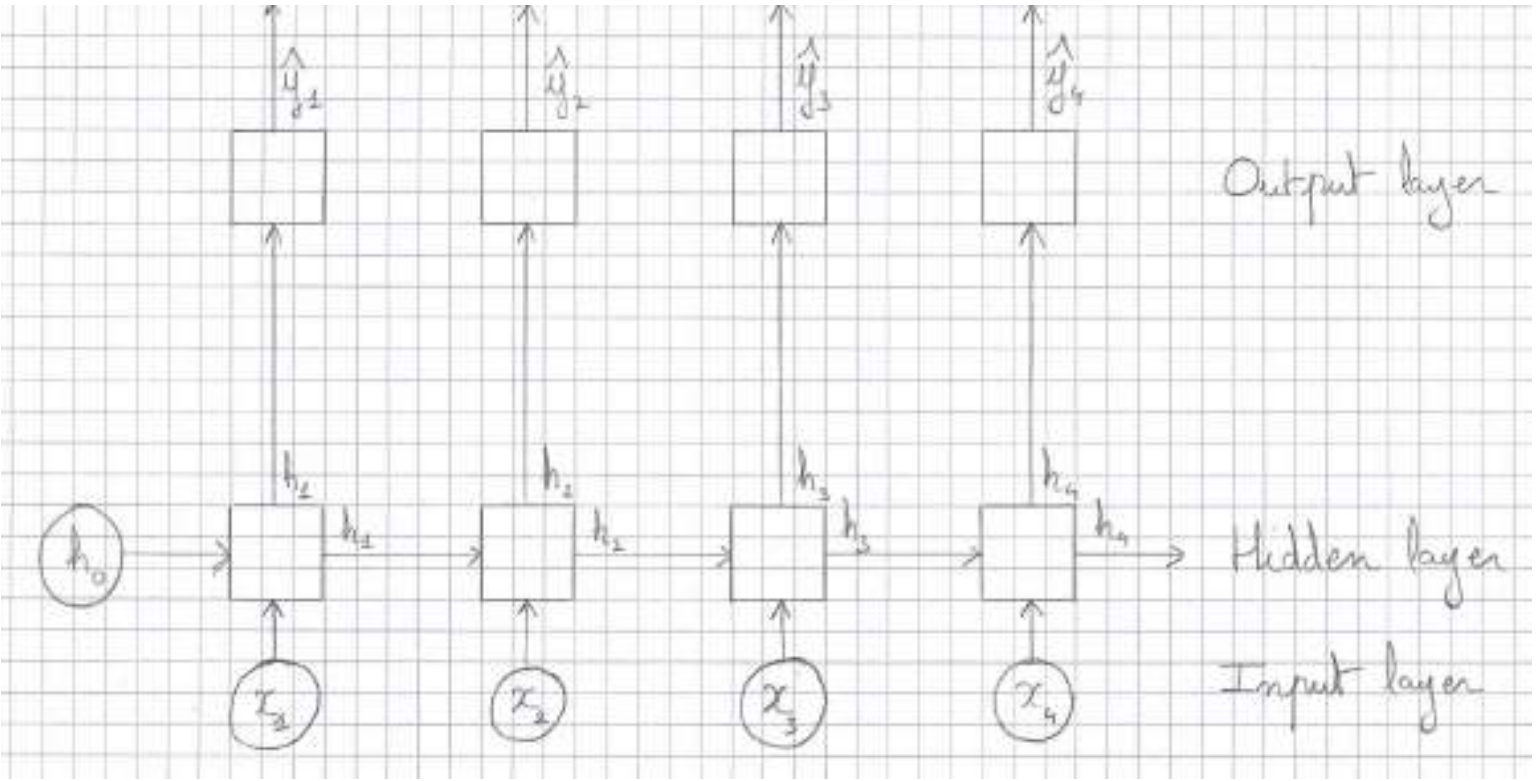
RNNs offer a lot of variability



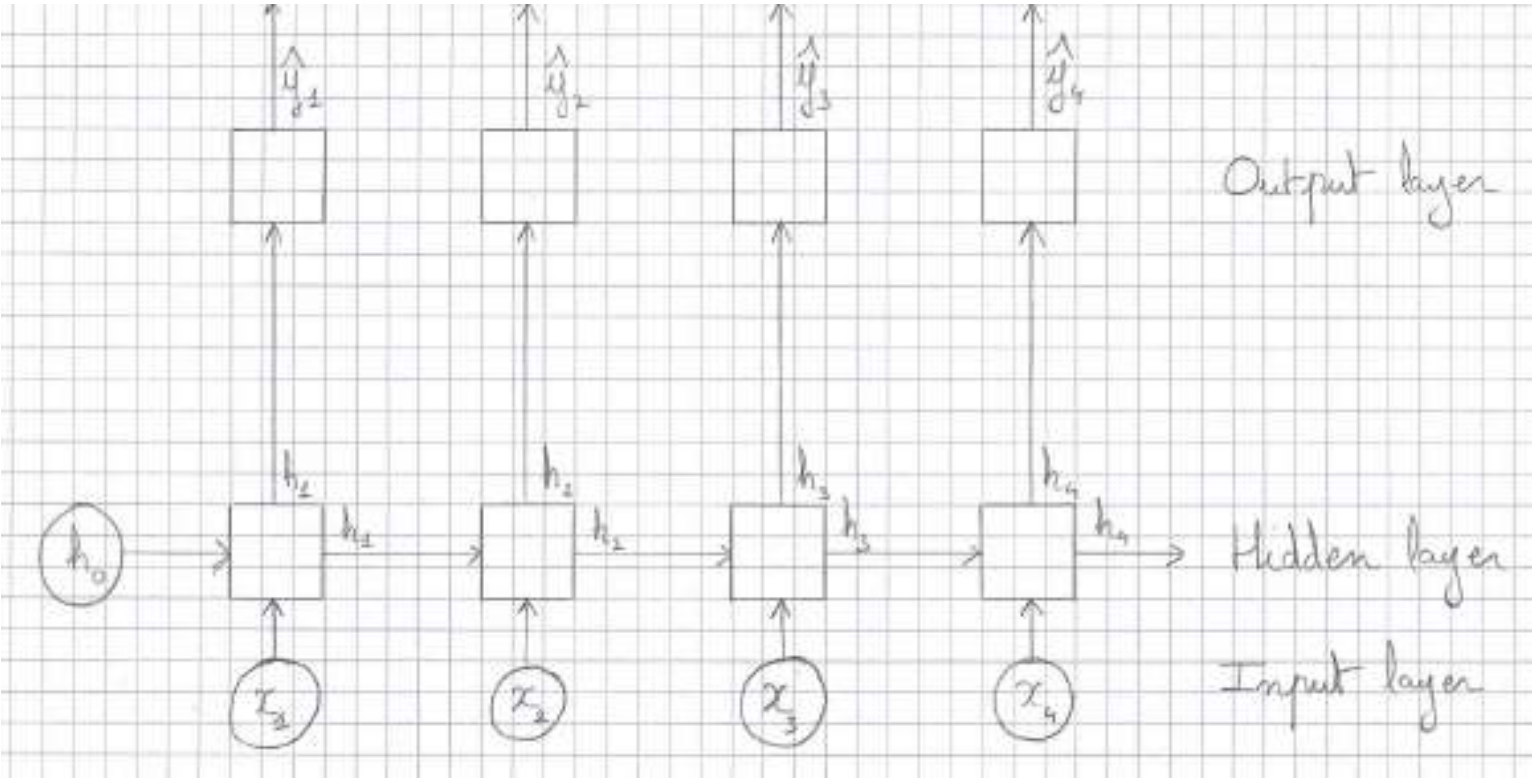
- ANNs can't deal with sequential or "temporal" data
- ANNs lack memory
- ANNs have a fixed architecture: fixed input size and a fixed output size
- RNNs are more "biologically realistic" because of recurrent connectivities found in the visual cortex of the brain

Definition of RNN

- Input layer - Data comes sequentially: $\mathbf{x}_1, \mathbf{x}_2, \ldots$
- Hidden Layer - Hidden state of the network at time t: $\mathbf{h}_t$
- Output layer - For the input x_t, the prediction is given by $\hat{y}_t$



Definition of RNN



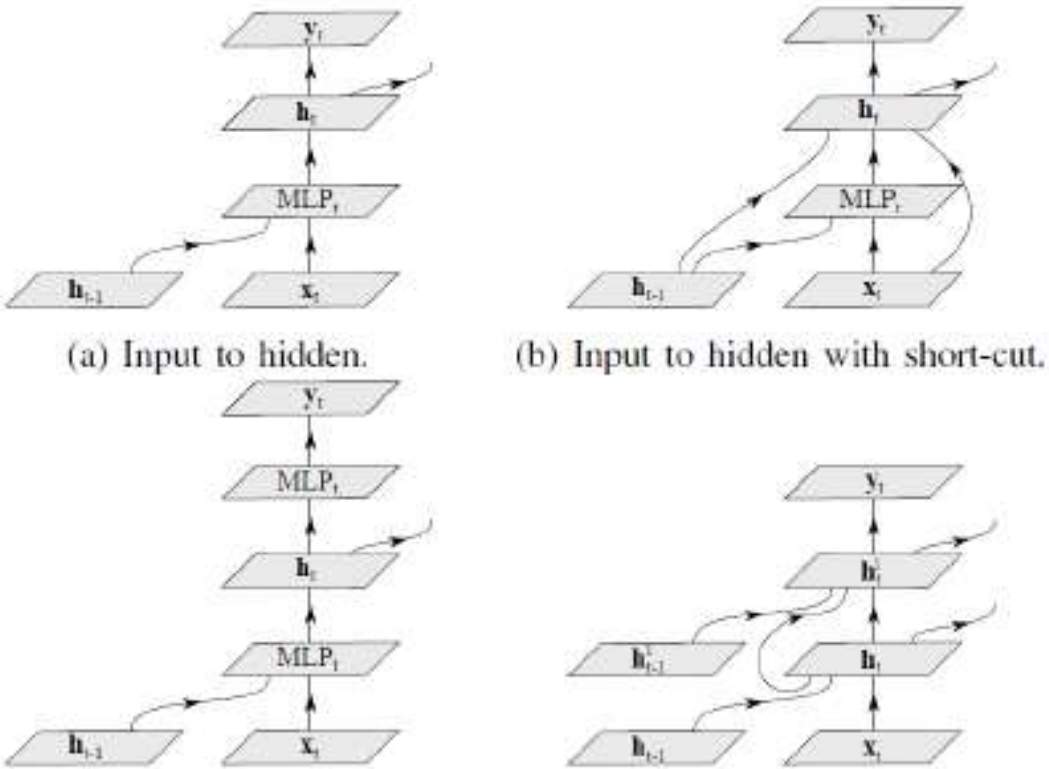
Hidden neuron:

$$\mathbf{h}_t = \tanh(W_{HH}\mathbf{h}_{t-1} + W_{IH}\mathbf{x}_t + \mathbf{b}_h)$$

Output neuron:

$$\hat{y}_t = \mathrm{softmax}(W_{HO}\mathbf{h}_t + \mathbf{b}_{out})$$

Deep RNN



(a) Input to hidden.

(b) Input to hidden with short-cut.

(c) Hidden to hidden and output.

(d) Stack of hidden states.

Bi-directional RNN

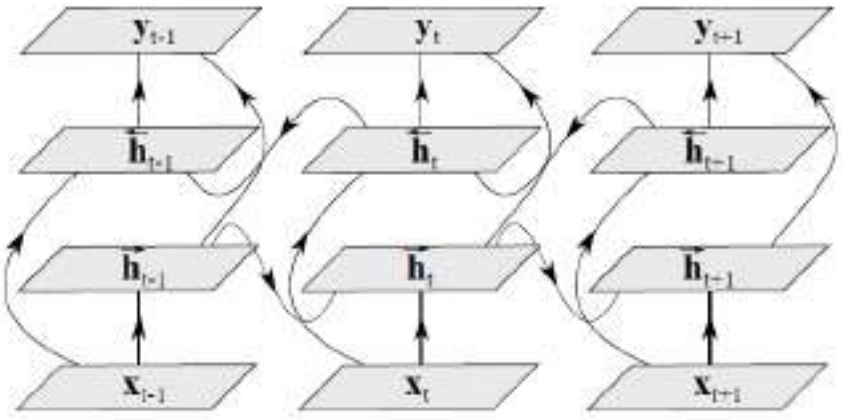
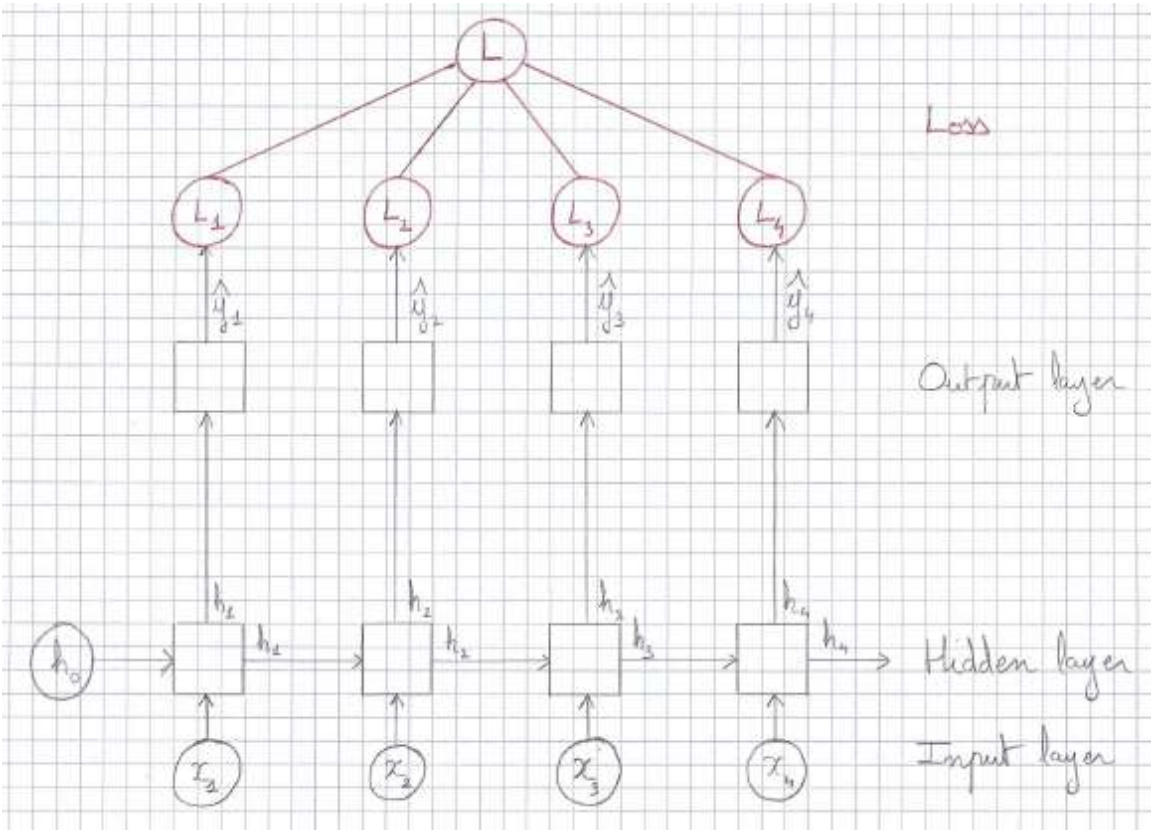


Figure: bi-directional recurrent neural network (BRNN)

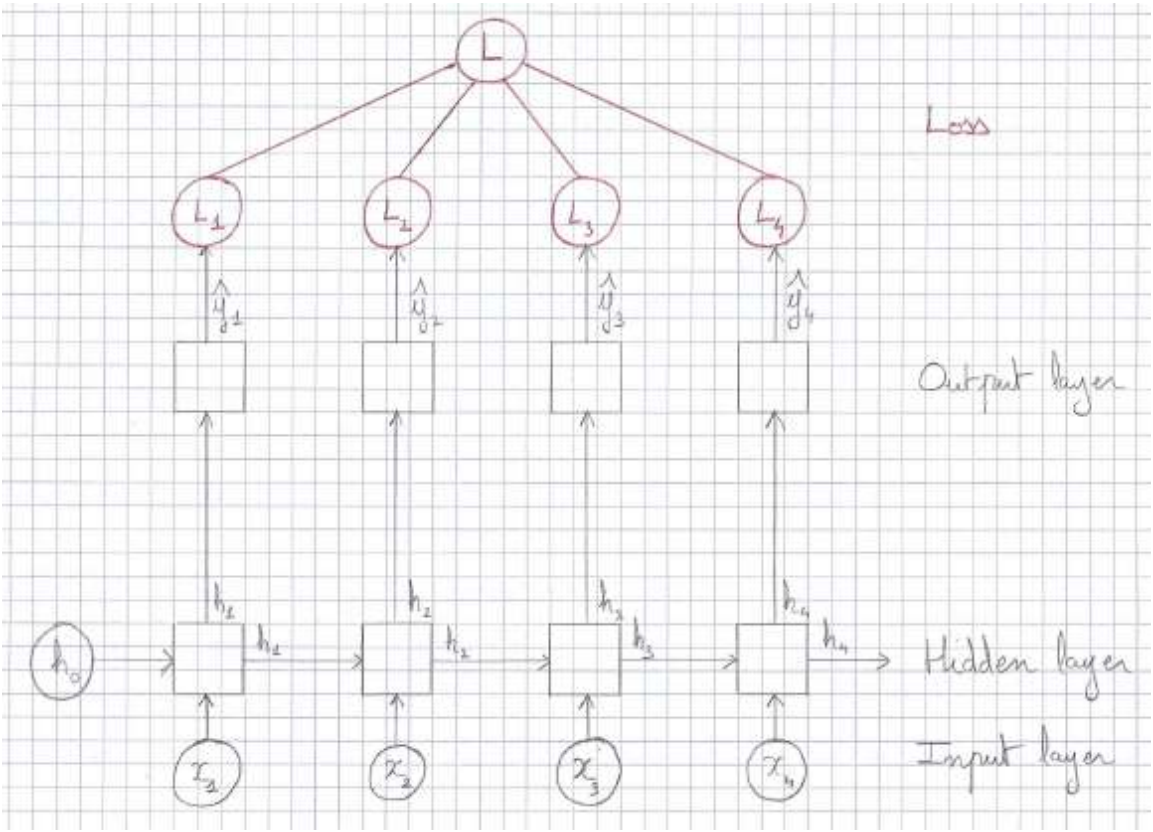
$$\mathbf{y}_t = W_{\overrightarrow{HO}}\overrightarrow{\mathbf{h}_t} + W_{\overleftarrow{HO}}\overleftarrow{\mathbf{h}_t} + \mathbf{b}_o$$

Outline

Loss



Loss



The backpropagation equation is given by

$$\frac{\partial L_T}{\partial W_{HH}} = \frac{\partial L_T}{\partial \hat{y}_T} \sum_{k=1}^{T} \frac{\partial \hat{y}_T}{\partial \mathbf{h}_T} \left(\prod_{m=k+1}^{T} \frac{\partial \mathbf{h}_m}{\partial \mathbf{h}_{m-1}} \right) \frac{\partial \mathbf{h}_k}{\partial W_{HH}}$$

Outline

Improving hidden units in RNN

Output gate (for reading)

$$\mathbf{o}_t = \sigma(W_{o,h}\mathbf{h}_{t-1} + W_{o,x}\mathbf{x}_t + \mathbf{b}_o)$$

Input gate (for writing)

$$\mathbf{i}_t = \sigma(W_{i,h}\mathbf{h}_{t-1} + W_{i,x}\mathbf{x}_t + \mathbf{b}_i)$$

Forget gate (for remembering)

$$\mathbf{f}_t = \sigma(W_{f,h}\mathbf{h}_{t-1} + W_{f,x}\mathbf{x}_t + \mathbf{b}_f)$$

Candidate hidden state.

$$\tilde{\mathbf{h}}_t = \tanh(W_h(\mathbf{o}_t \odot \mathbf{h}_{t-1}) + W_x\mathbf{x}_t + \mathbf{b})$$

The final state $\mathbf{h}_t$ is given by

$$\mathbf{h}_t = \mathbf{f}_t \odot \mathbf{h}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{h}}_t.$$

Warning: the forget gate is used for forgetting, but it actually operates as a remember gate: 1 in a forget gate means remembering everything not forgetting everything.

Improving hidden units in RNN: failure

The previous hidden units described by

$$\mathbf{h}_t = \mathbf{f}_t \odot \mathbf{h}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{h}}_t$$

fail.

Two problems:

- The forget gate and the input gate are not synchronized at the beginning of the training, which can cause the hidden states to become large and unstable.
- Since the hidden state is not bounded, the gates can be saturated, which implies difficulties to train the network.

Empirical evidence:

["LSTM: A search space odyssey", Greff et al. 2017]

Gated Recurrent Unit

One way to circumvent this issue is to specify explicitly the dependence structure between the forget gate and the writing gate.

For example, we can set the forget gate to 1 minus the writing gate:

$$\mathbf{h}_t = (1 - \mathbf{i}_t) \odot \mathbf{h}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{h}}_t.$$

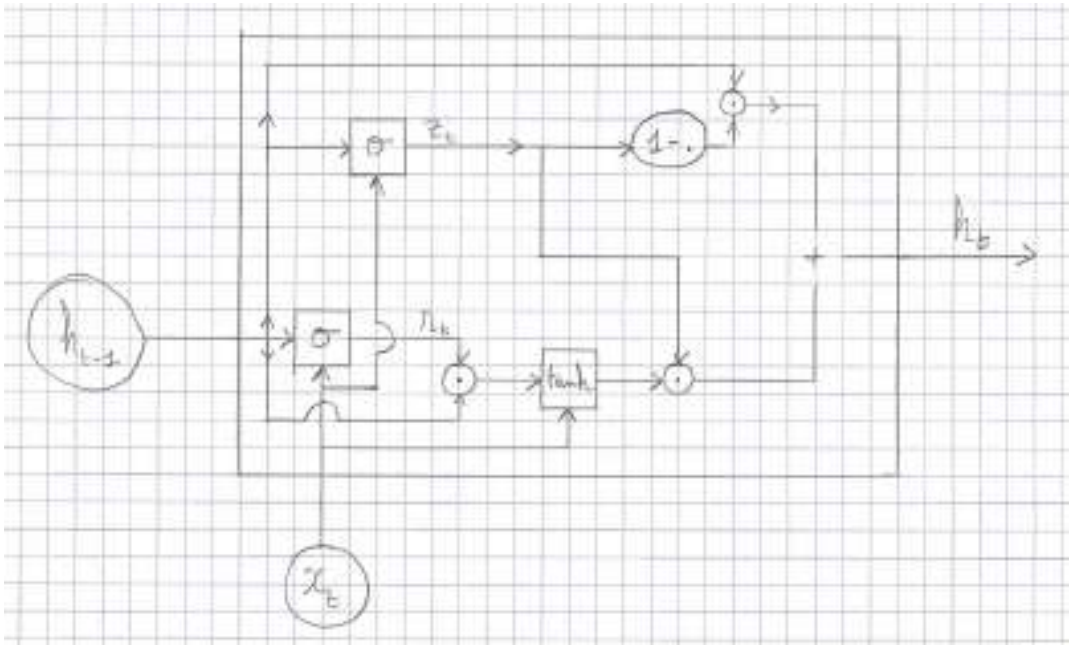
In that case, the new hidden state $\mathbf{h}_t$ is a weighted average of the previous hidden state $\mathbf{h}_{t-1}$ and the newly created candidate $\tilde{\mathbf{h}}_t$.

Consequently, $\mathbf{h}_t$ is bounded if $\mathbf{h}_{t-1}$ and $\tilde{\mathbf{h}}_t$ are, which is the case using bounded activation functions.

This is exactly the Gated Recurrent Unit.

Gated Recurrent Unit

["Empirical evaluation of gated recurrent neural networks on sequence modeling", Chung et al. 2014]



Reset gate (read gate)

$$\mathbf{r}_t = \sigma(W_{r,h}\mathbf{h}_{t-1} + W_{r,x}\mathbf{x}_t + \mathbf{b}_r)$$

Update gate (forget gate)

$$\mathbf{z}_t = \sigma(W_{z,h}\mathbf{h}_{t-1} + W_{z,x}\mathbf{x}_t + \mathbf{b}_z)$$

Candidate hidden state

$$\tilde{\mathbf{h}}_t = \tanh(W_h(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + W_x\mathbf{x}_t + \mathbf{b})$$

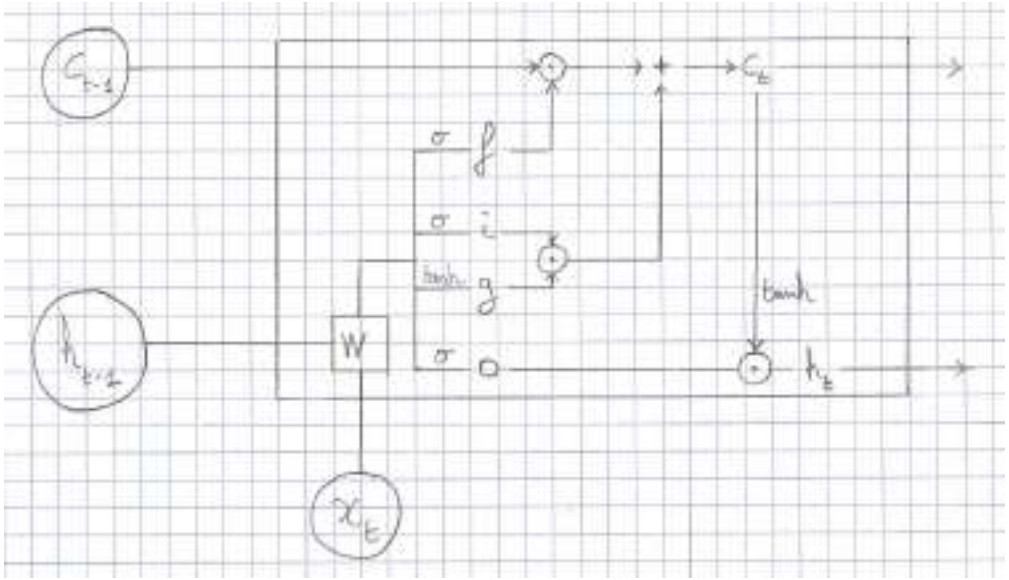
Hidden state

$$\mathbf{h}_t = \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \tilde{\mathbf{h}}_t$$

Long Short Term Memory (LSTM)

LSTM is another way to circumvent the issue of unboundedness of internal states.

["Long short-term memory", Hochreiter and Jürgen Schmidhuber 1997]



LSTM equations:

$$\mathbf{i}_t = \sigma(W_{i,h}\mathbf{h}_{t-1} + W_{i,x}\mathbf{x}_t + b_i)$$
$$\mathbf{o}_t = \sigma(W_{o,h}\mathbf{h}_{t-1} + W_{o,x}\mathbf{x}_t + b_o)$$
$$\mathbf{f}_t = \sigma(W_{f,h}\mathbf{h}_{t-1} + W_{f,x}\mathbf{x}_t + b_f)$$
$$\mathbf{g}_t = \tanh(W_{g,h}\mathbf{h}_{t-1} + W_{g,x}\mathbf{x}_t + b_g)$$

Cell state

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t$$

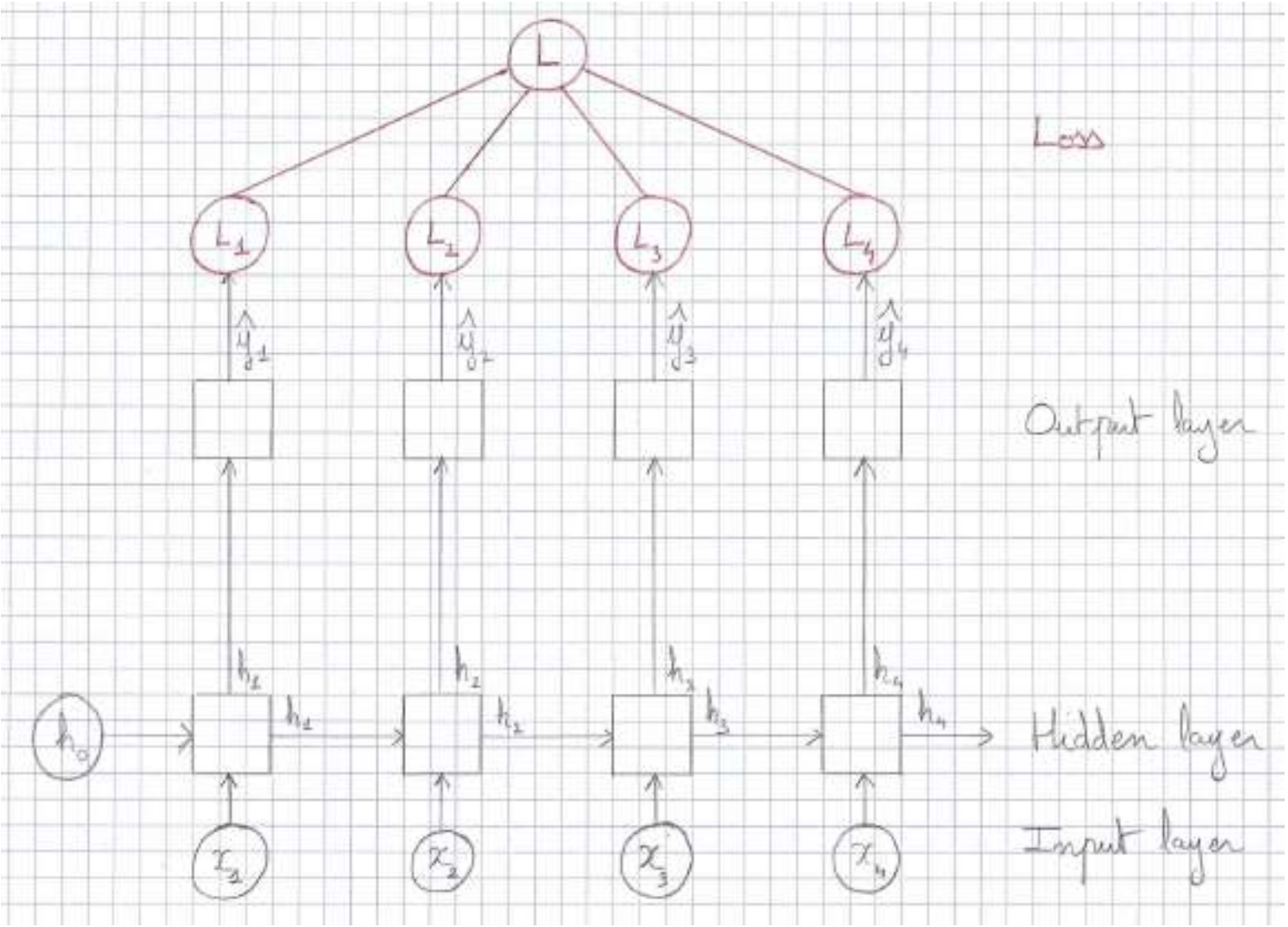
Hidden state

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$$

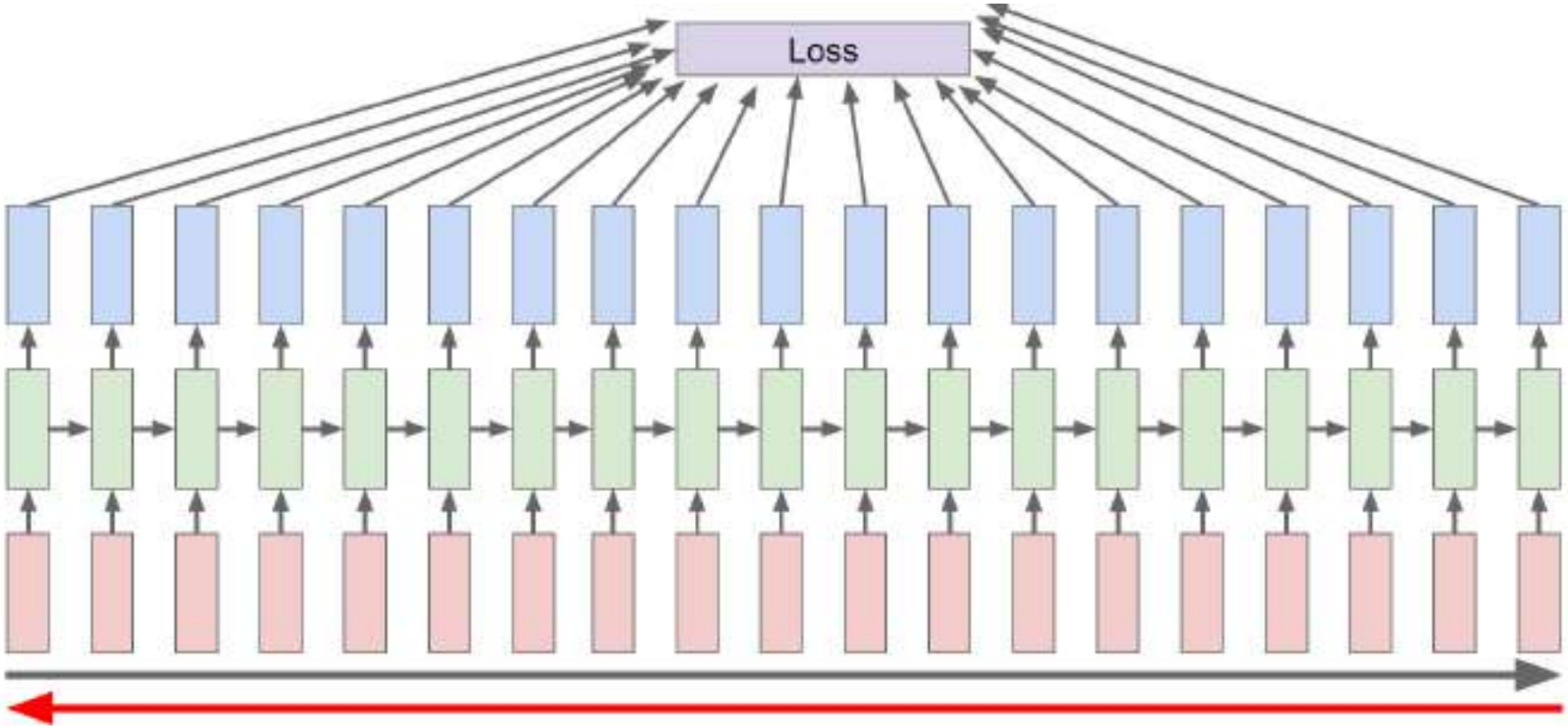
The prediction of the network at time t only depends on $\mathbf{h}_t$ and not on $\mathbf{c}_t$.

Outline

Loss

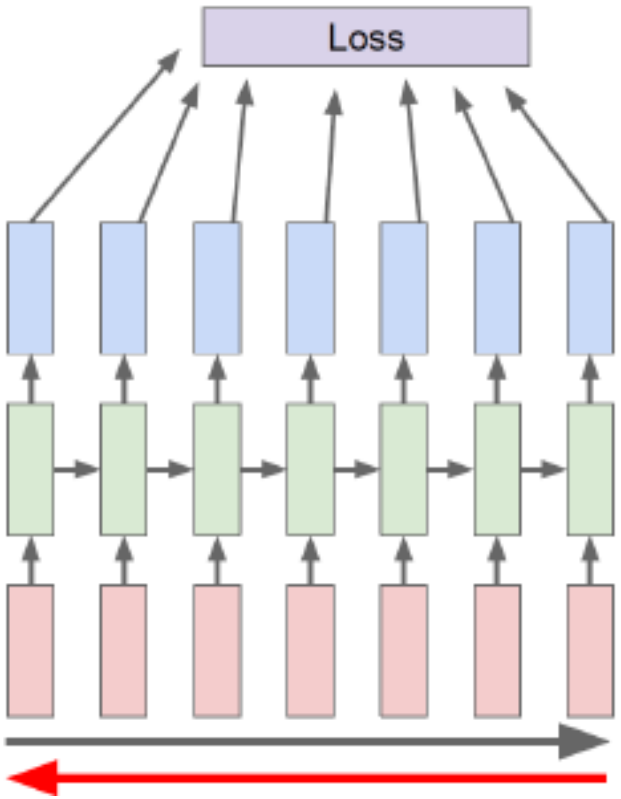


Backpropagation



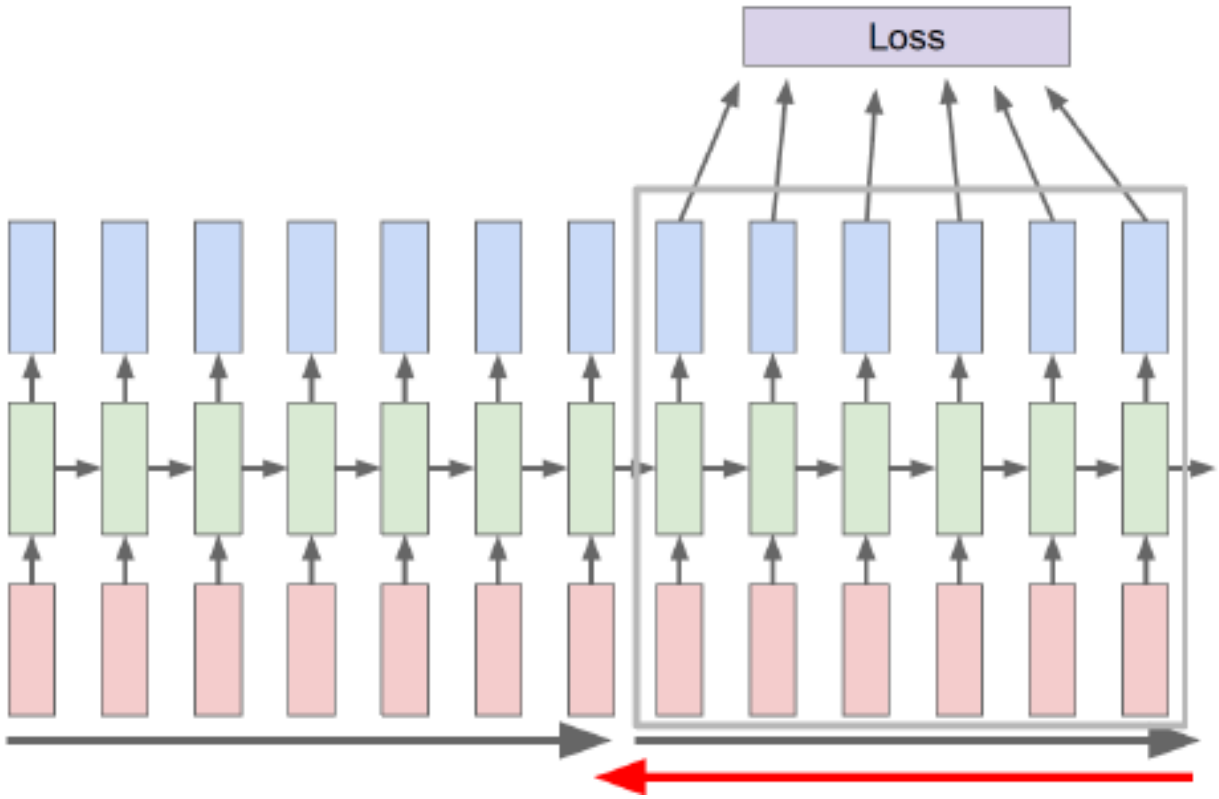
Problem: one gradient step is too costly. It requires a pass through the entire data set.

Truncated backpropagation



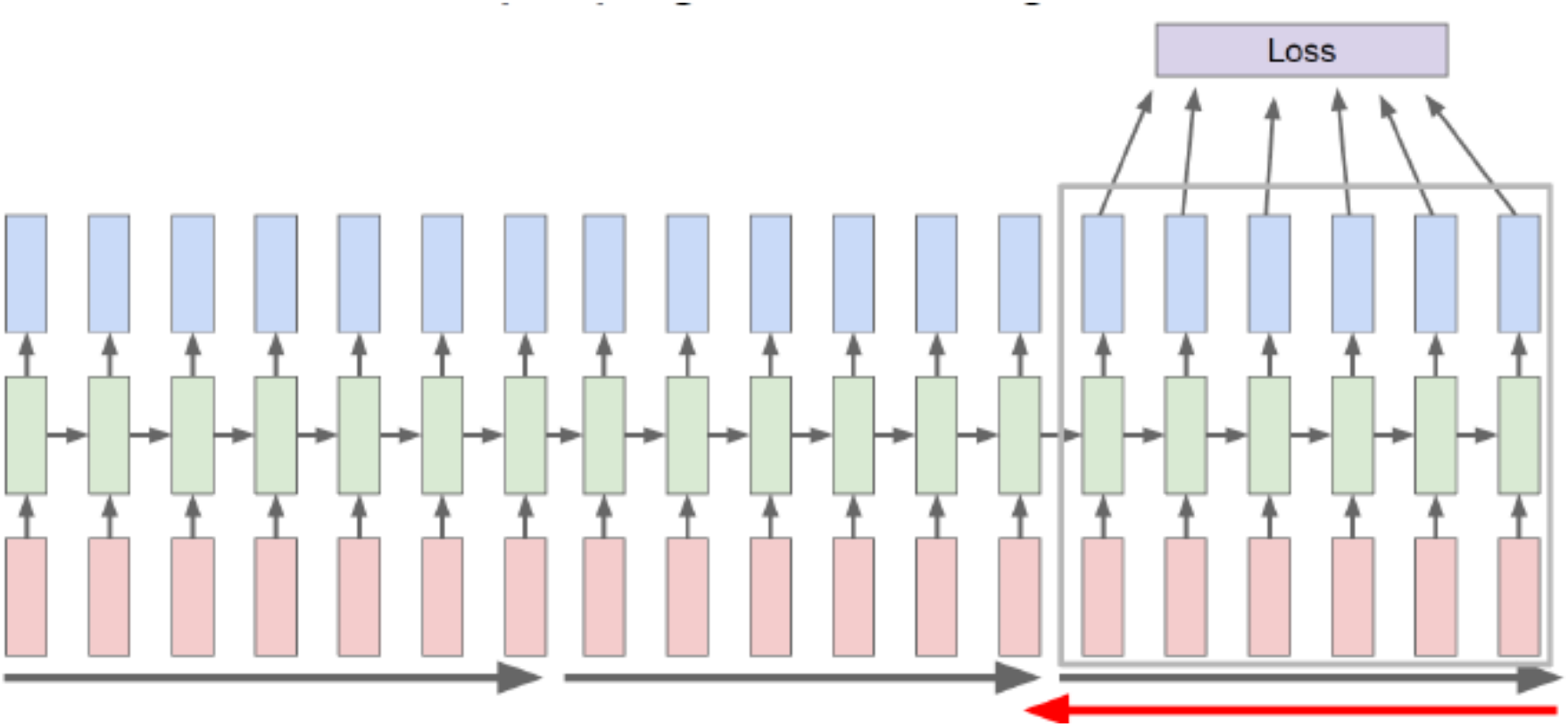
Choose a small number of steps (usually 100) and back-propagate only onto these data.

Truncated backpropagation



Propagate the weights and use backpropagation on the second batch of data.

Truncated backpropagation



Pursue...

Outline

Image Captioning: Neural Image Caption

[“Show and tell: A neural image caption generator”, Vinyals et al. 2015]

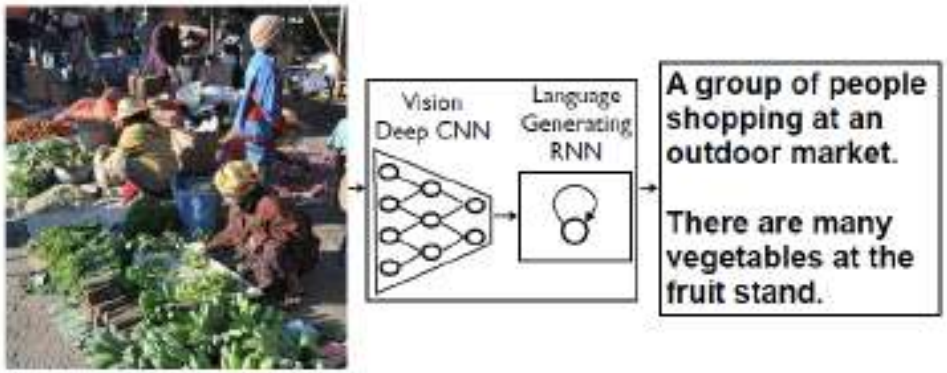
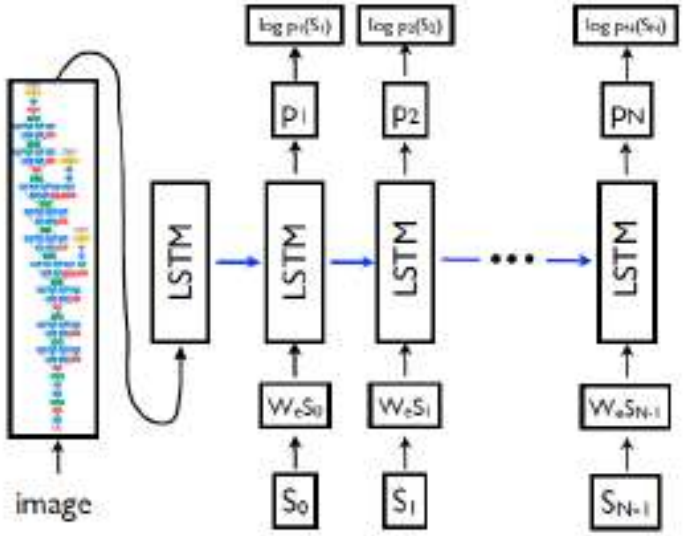


Image Captioning: Neural Image Caption



Aim:

$$\theta^\star \in \underset{\theta}{\operatorname{argmax}} \sum_{(I,S)} \log(p(S|I))$$

where I is the input image and S the sentence describing the image. Since the sentence length can be arbitrary long, the log probability is rewritten as

$$\log(p(S|I)) = \sum_{t=0}^{N} p(S_t|I, S_0, \ldots, S_{t-1}).$$

Image Captioning: Neural Image Caption

Inference time. Two approaches:

- Sampling: sample the first word using p_1 then use this word as input to sample the second word according to p_2. Repeat the process until the network produces a stop word.
- BeamSearch: Choose the k best sentences of length t then use this set to generate the k best sentences of length $t+1$.

How to compare two sentences?
Example:

- Candidate: the the the the the the the
- Reference 1: the cat is on the mat
- Reference 2: There is a cat on the mat

Metric:

- Precision : 7/7
- BLEU (bilingual evaluation understudy): 2/7 (maximum number of times a word is encountered in any reference sentence)

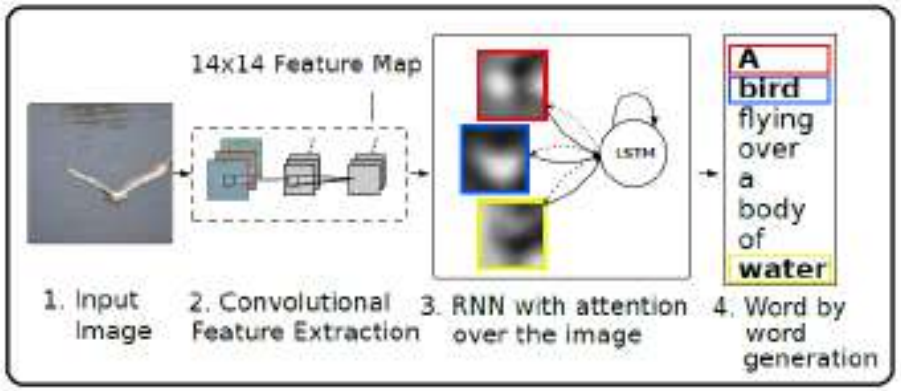
["BLEU: a method for automatic evaluation of machine translation", Papineni et al. 2002]

Image Captioning: Neural Image Caption

Figure 5. A selection of evaluation results, grouped by human rating.

Image Captioning with attention mechanism

["Show, attend and tell: Neural image caption generation with visual attention", K. Xu et al. 2015]



Predicted sequence of words: $\{y_1, \ldots, y_C\}$, $y_i \in \mathbb{R}^K$, where K is the size of the dictionary.

Image features: $\{a_1, \ldots, a_L\}$, where $a_i \in \mathbb{R}^D$ is a feature corresponding to a small precise area in the image (extraction from a early layer of a CNN).

$$\begin{pmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \mathbf{g}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} T_{D+m+n,n} \begin{pmatrix} \mathbf{E}\mathbf{y}_{t-1} \\ \mathbf{h}_{t-1} \\ \hat{\mathbf{z}}_t \end{pmatrix}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t).$$

Image Captioning with attention mechanism

$$\hat{\mathbf{z}}_t = \sum_{i=1}^{L} s_{t,i}\mathbf{a}_i,$$

where $s_{t,i} = 1$ if position i in the image should be selected at time t.

$$\mathbb{P}[s_{t,i} = 1 | s_{j<t}, \mathbf{a}] = \alpha_{t,i},$$

$$e_{ti} = f_{att}(\mathbf{a}_i, \mathbf{h}_{t-1}),$$

$$\alpha_{t,i} = \frac{\exp(e_{ti})}{\sum_j \exp(e_{tj})}.$$

Figure 2. Attention over time. As the model generates each word, its attention changes to reflect the relevant parts of the image. "soft" (top row) vs "hard" (bottom row) attention. (Note that both models generated the same captions in this example.)

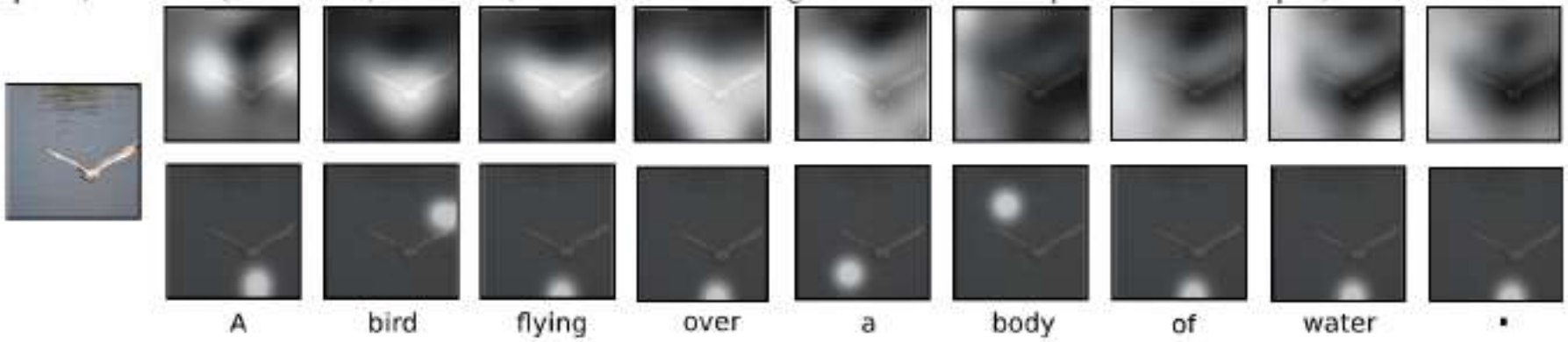


Image Captioning with attention mechanism

Figure 3. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)

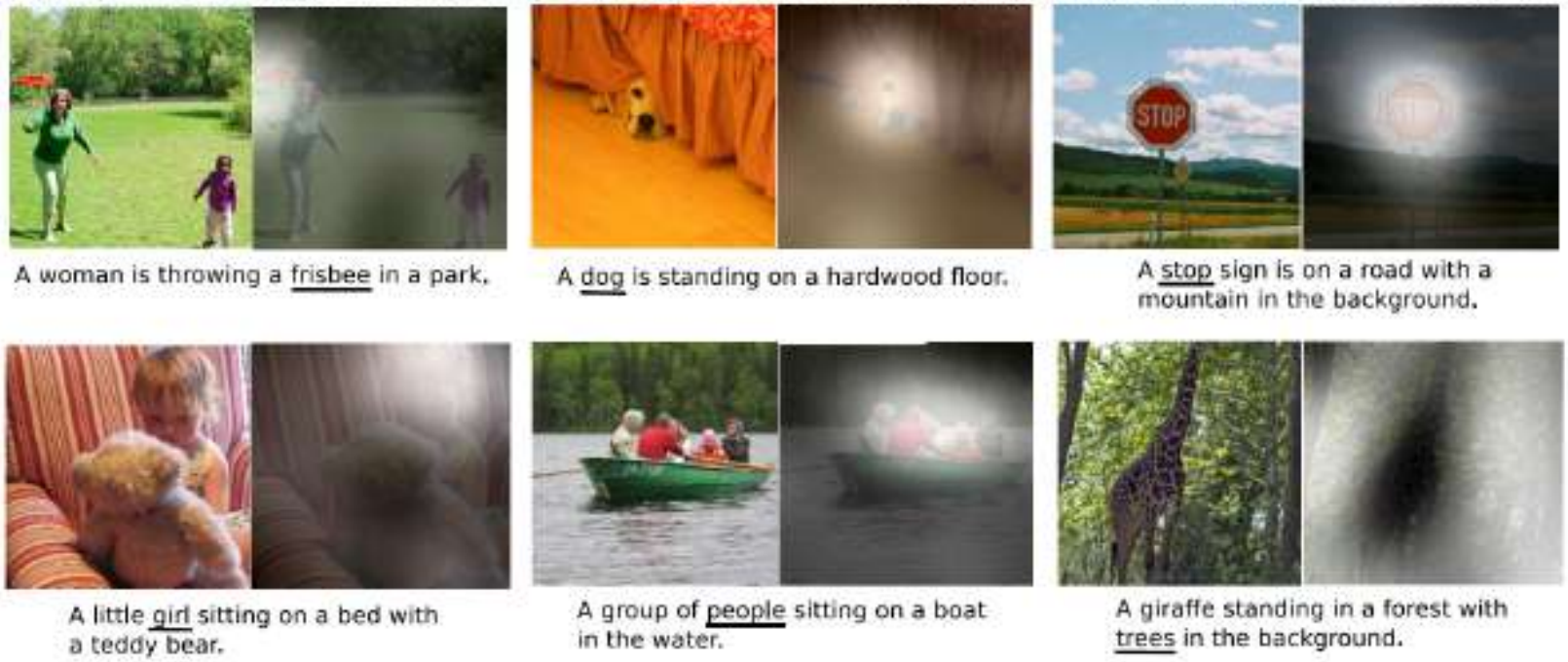
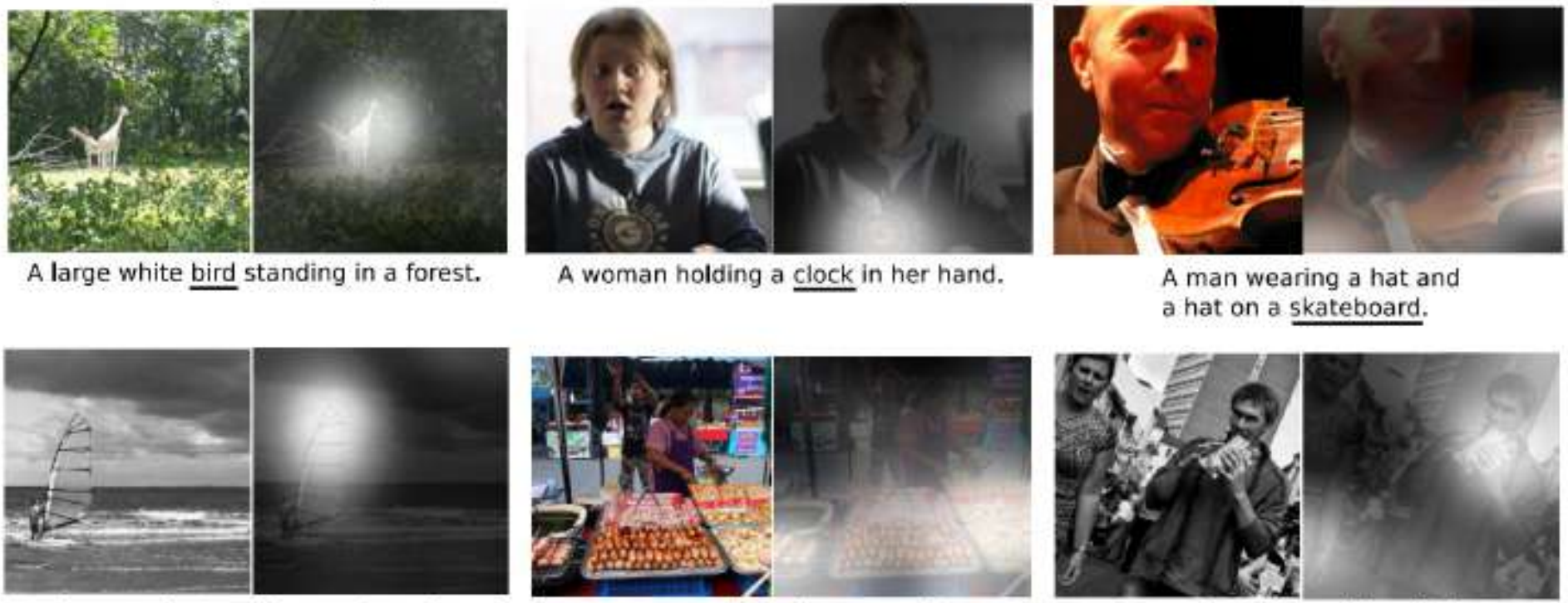
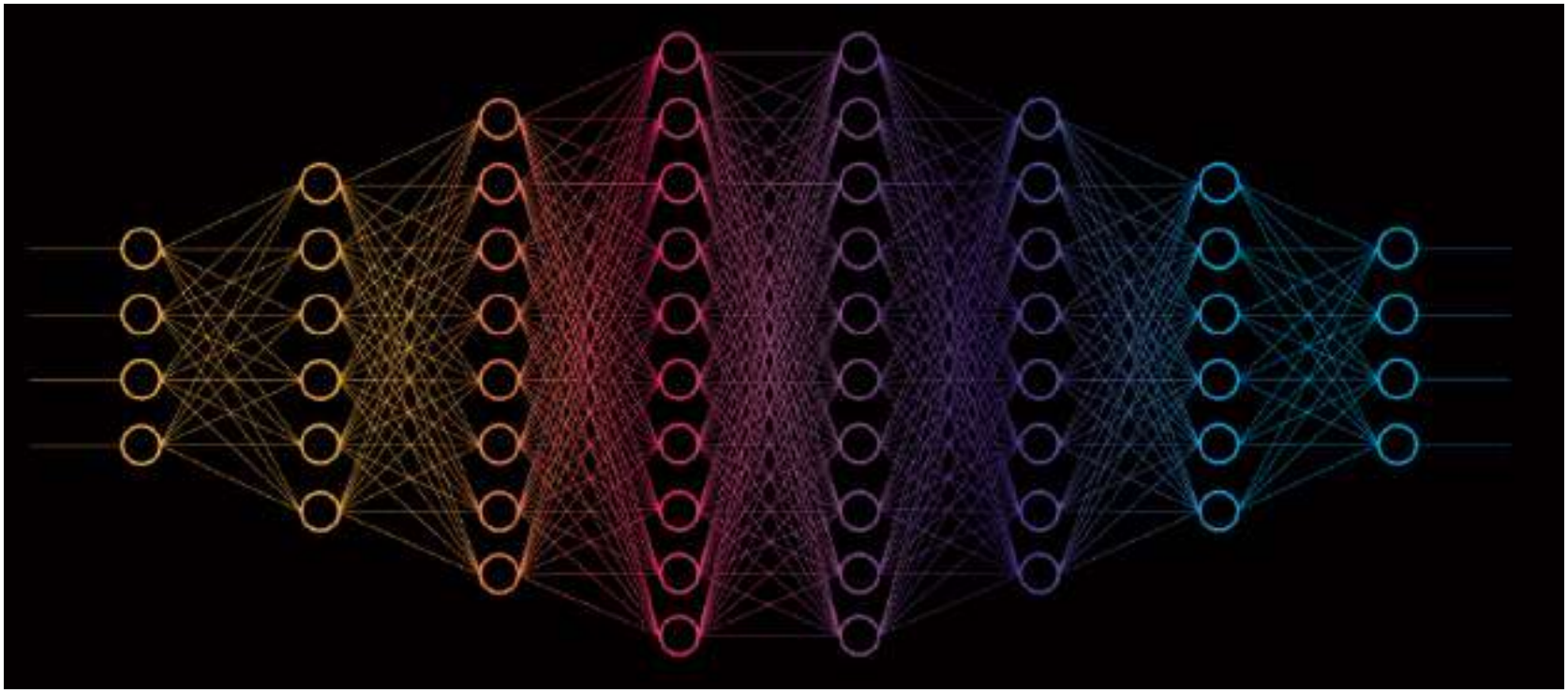


Image Captioning with attention mechanism

Figure 5. Examples of mistakes where we can use attention to gain intuition into what the model saw.



The End![2]



[2] More content at https://erwanscornet.github.io/, Teaching section.

Outline

5. Famous CNN
   - LeNet (1998)
   - AlexNet (2012)
   - ZFNet (2013)
   - VGGNet (2014)
   - GoogLeNet (2014)
   - ResNet (2016)
   - DenseNet (2017)
   - Many other CNN

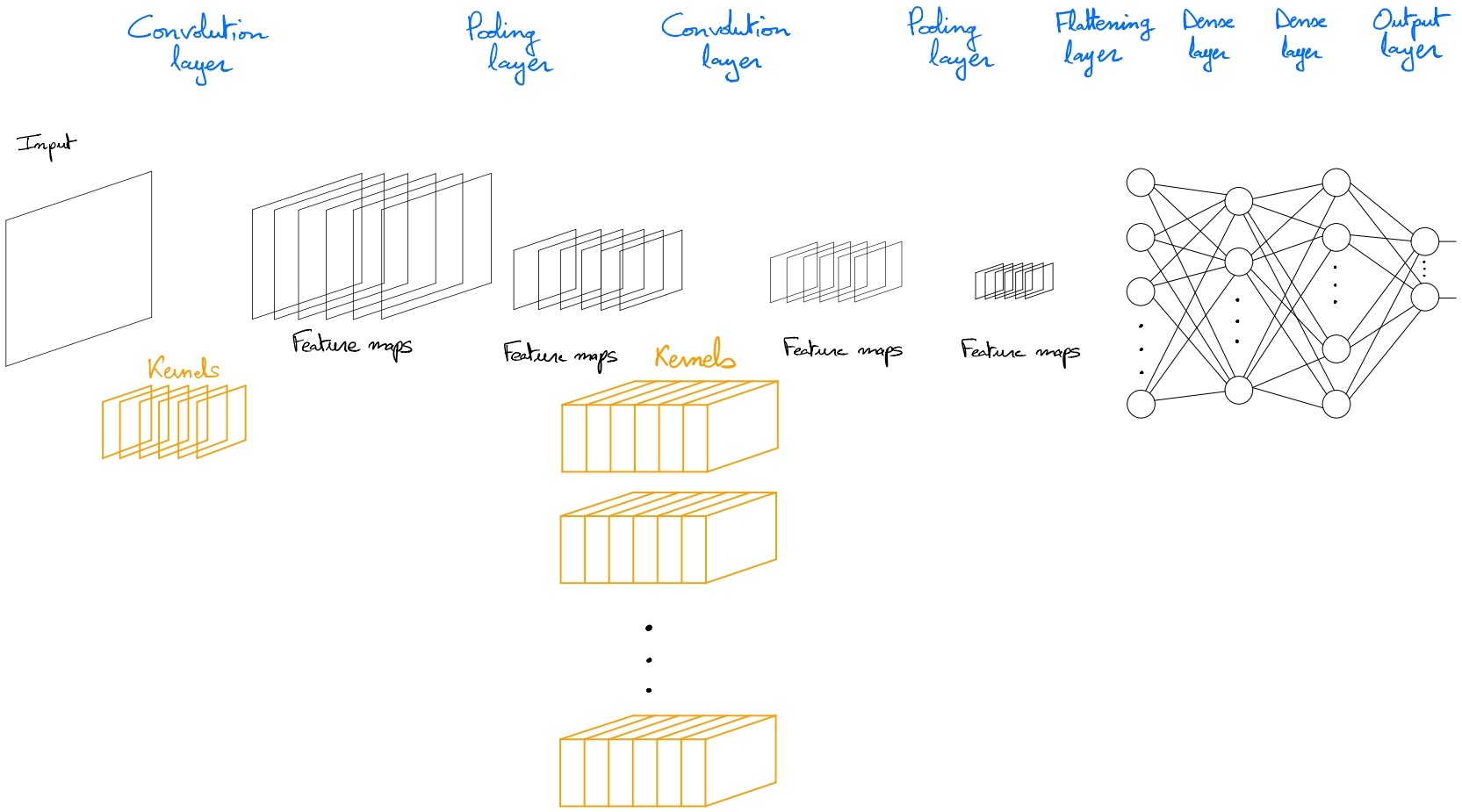
Outline

5. Famous CNN
   - LeNet (1998)
   - AlexNet (2012)
   - ZFNet (2013)
   - VGGNet (2014)
   - GoogLeNet (2014)
   - ResNet (2016)
   - DenseNet (2017)
   - Many other CNN

LeNet

["Generalization and network design strategies", LeCun et al. 1989]

["Gradient-based learning applied to document recognition", LeCun, Bottou, Bengio, et al. 1998]



Outline

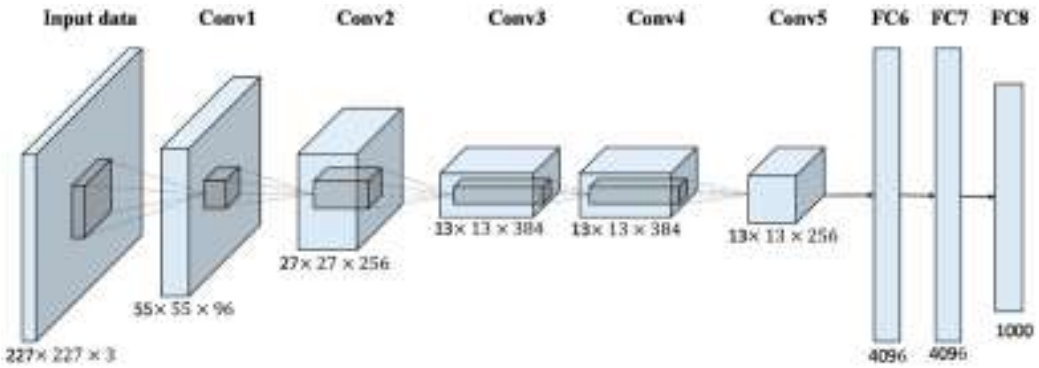
5. Famous CNN
   - LeNet (1998)
   - AlexNet (2012)
   - ZFNet (2013)
   - VGGNet (2014)
   - GoogLeNet (2014)
   - ResNet (2016)
   - DenseNet (2017)
   - Many other CNN

AlexNet

["Imagenet classification with deep convolutional neural networks", Krizhevsky et al. 2012]

Ingredients:

- Activation function (ReLU)
- Local Response Normalization (LRN)
- Overlapping pooling (3×3 window with a stride $S = 2$ which reduces overfitting)
- Dropout
- Data augmentation



Numerical results

Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs[7]	–	–	26.2%
1CNN	40.7%	18.2%	–
5CNNs	38.1%	16.4%	16.4%
1CNN*	39.0%	16.6%	–
7CNNs*	36.7%	15.4%	15.3%

- First line is the second runner-up.
- Second and third lines are results output by the averaging over 1 or 5 CNN described before.
- Last two lines correspond to networks with an extra convolutional layer after the last pooling layer which has been trained on Image Net Fall 2011 then "fine-tuned" on the ImageNet 2012 data base.

AlexNet has a very similar architecture to LeNet, but is deeper, bigger, and features Convolutional Layers stacked on top of each other: previously, pooling layers followed immediately each convolutional layer.

Results

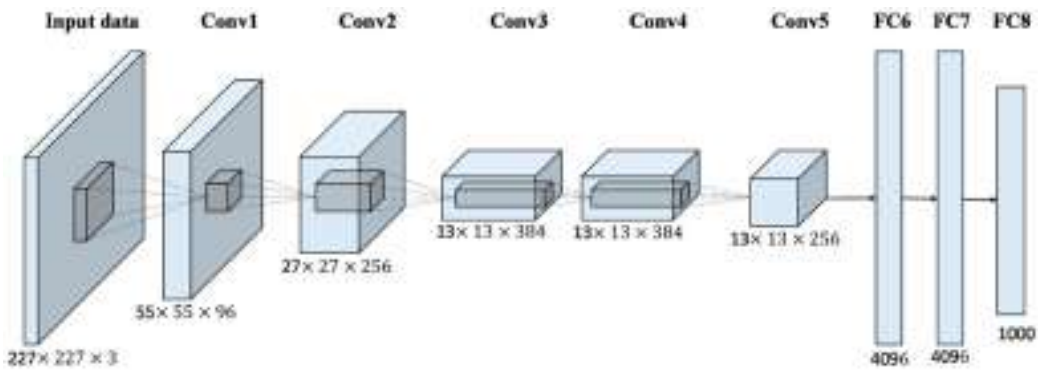
Figure 4: **(Left)** Eight ILSVRC-2010 test images and the five labels considered most probable by our model. The correct label is written under each image, and the probability assigned to the correct label is also shown with a red bar (if it happens to be in the top 5). **(Right)** Five ILSVRC-2010 test images in the first column. The remaining columns show the six training images that produce feature vectors in the last hidden layer with the smallest Euclidean distance from the feature vector for the test image.

Outline

5. Famous CNN
   - LeNet (1998)
   - AlexNet (2012)
   - ZFNet (2013)
   - VGGNet (2014)
   - GoogLeNet (2014)
   - ResNet (2016)
   - DenseNet (2017)
   - Many other CNN

ZFNet: Improve upon AlexNet

["Visualizing and understanding convolutional networks", Zeiler and Fergus 2014]



Aim at finding out what the different feature maps are searching for in order to obtain a better tuning of network architecture.

In ZFNet, feature maps are not divided across two different GPU. Thus connections between layers are less sparse than for AlexNet.

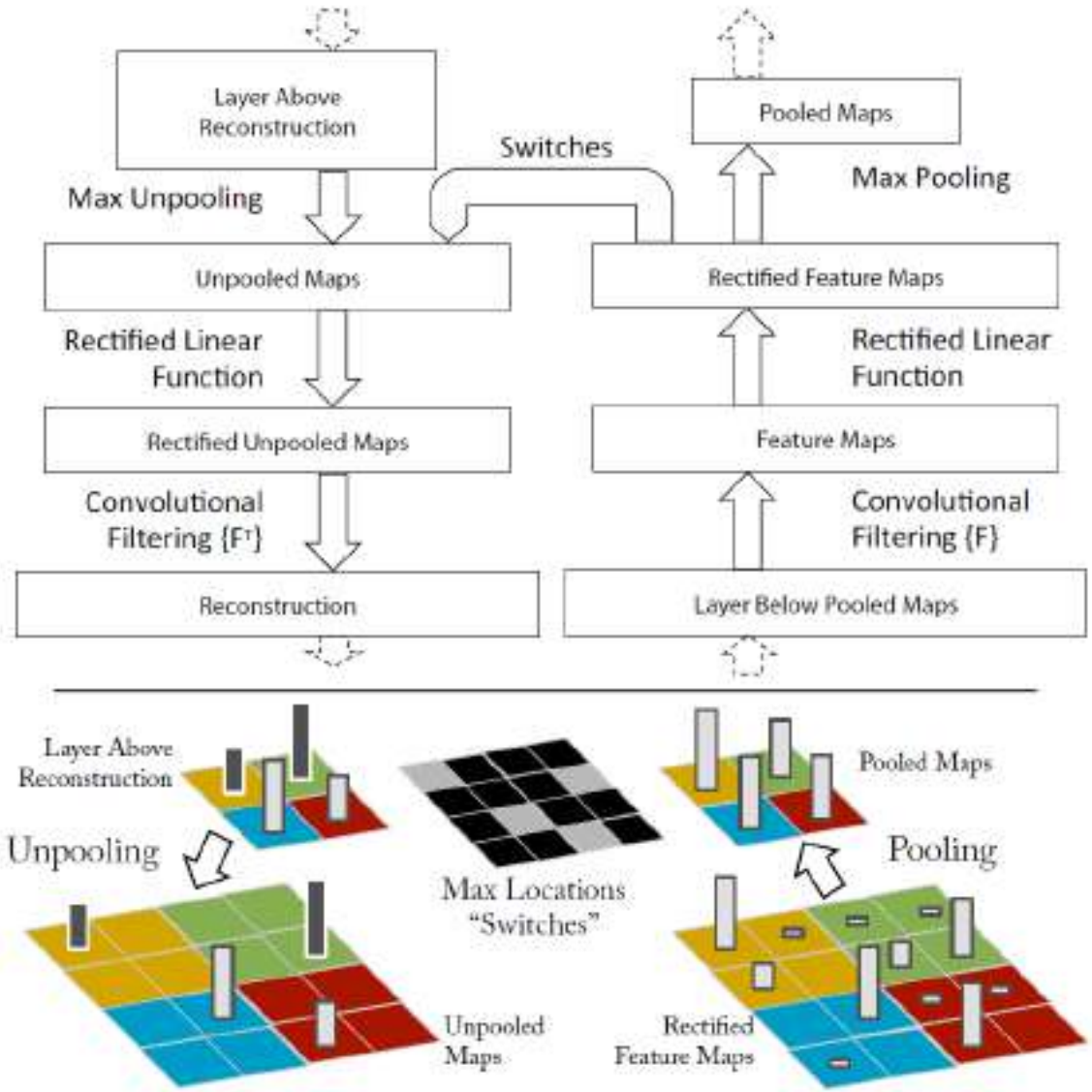
Deconvnet

Find the pixels that maximize the activation of a given feature map.

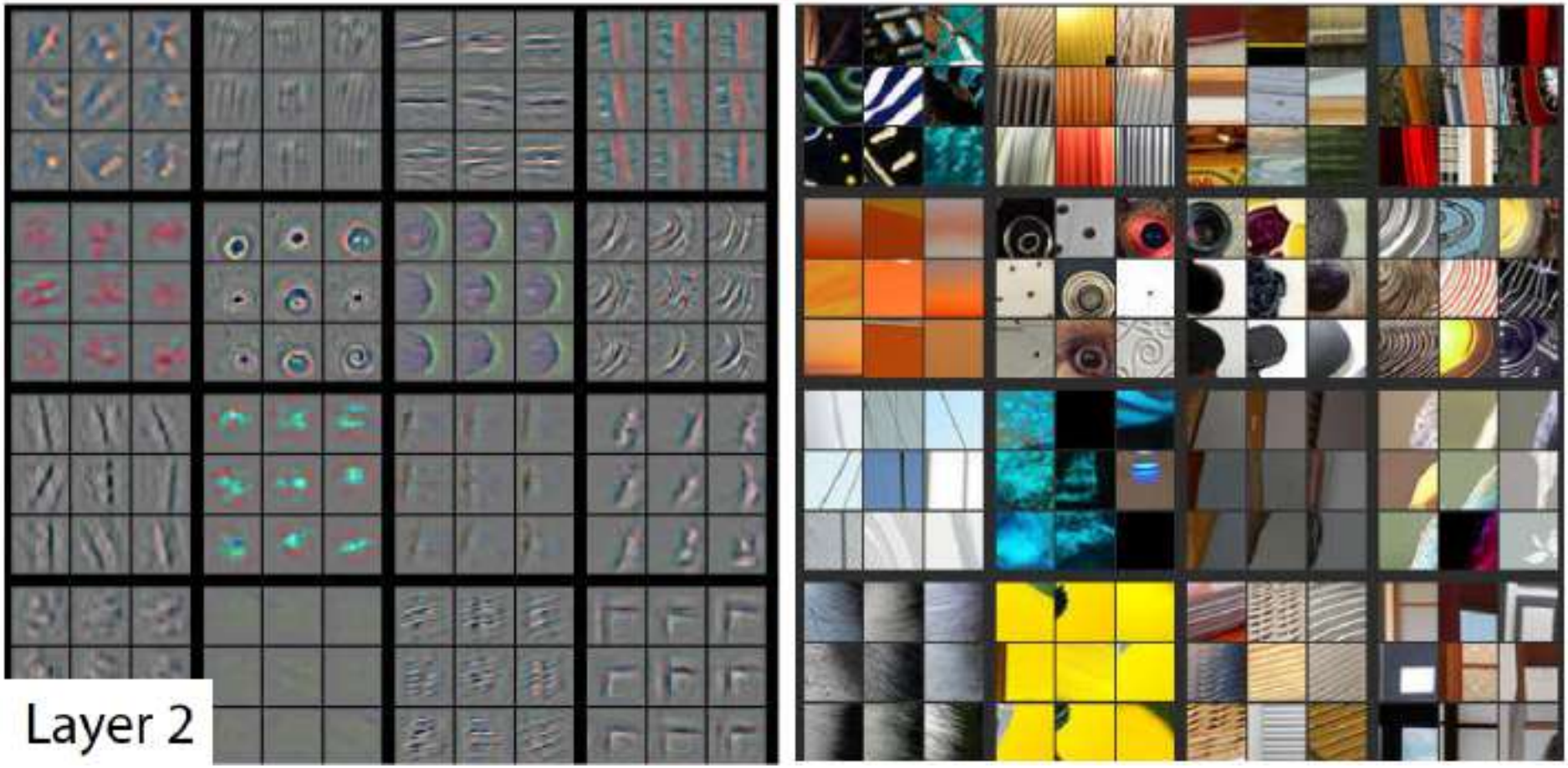
How? Invert the network.

Precisely:

- Choose a layer
- Choose a feature map
- Run the network on a validation set
- Choose the image maximizing the activation of this feature map
- "Backpropagate" this activation to obtain a stylized image in the pixel space



Results



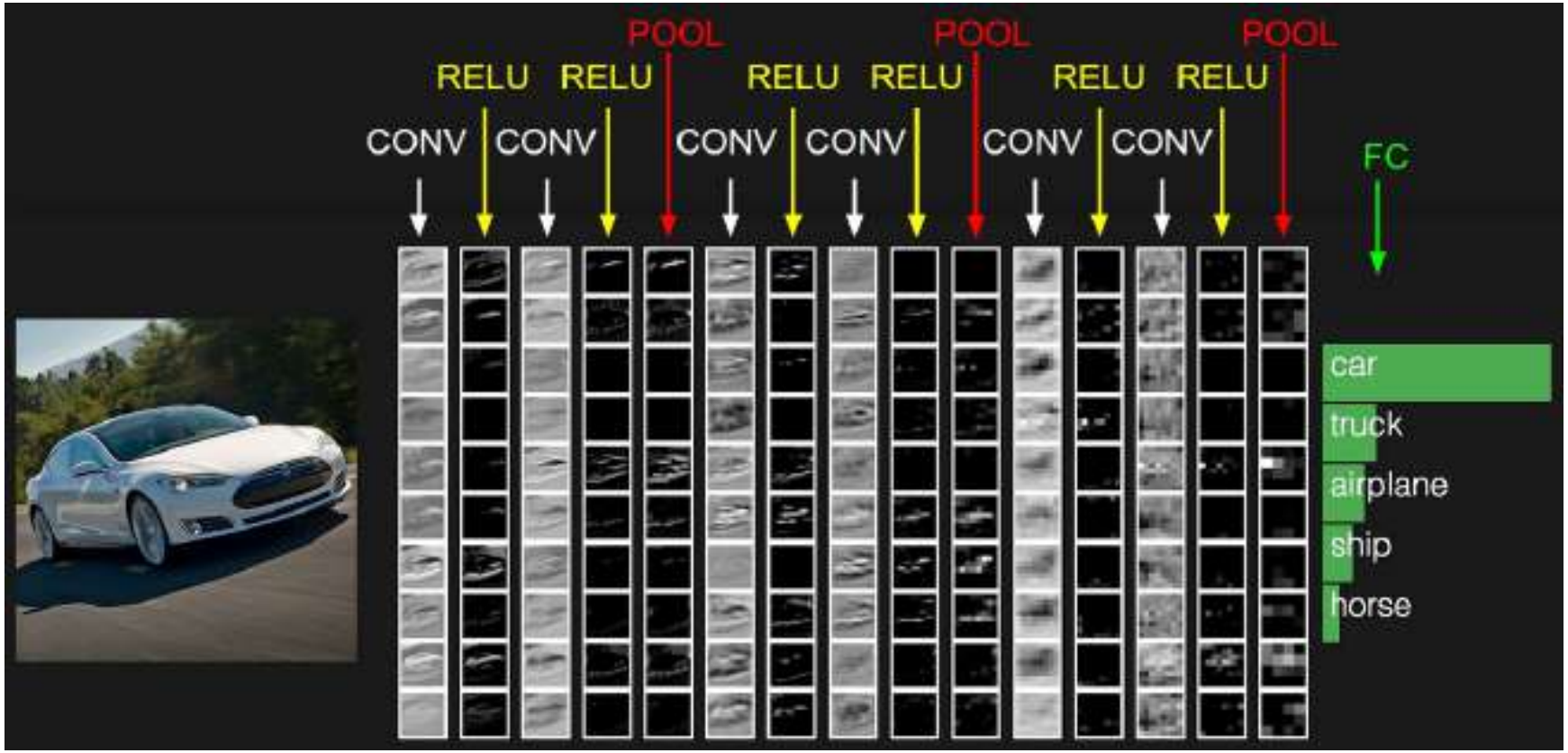
Top 9 activations in a random subset of feature maps across the validation data, projected down to pixel space using the previous deconvolutional network approach.

Outline

5. Famous CNN
   - LeNet (1998)
   - AlexNet (2012)
   - ZFNet (2013)
   - VGGNet (2014)
   - GoogLeNet (2014)
   - ResNet (2016)
   - DenseNet (2017)
   - Many other CNN

Tiny VGGnet

["Very deep convolutional networks for large-scale image recognition", Simonyan and Zisserman 2014b]



Convolutional layers:

- Small receptive field: 3×3 (smallest ones capable of capturing the notion of top/down, left/right!)
- Stride of 1
- Spatial resolution is preserved after convolution

Insightful remark...

If you stack 3 convolutional layers with receptive fields 3×3, you obtain a convolutional layer with receptive fields 7×7. What is the interest?

1. Stack of 3 convolutional layers of size 3×3: complexity of $3 \times \times 3 \times 3 = 27$.
2. One standard convolutional layer of size 7×7: complexity of 49.

In the first case, we cannot obtain every possible layer: the resulting object is a decomposition of three consecutive convolutional layers. There are less possibilities hence less parameters.

Outline

5. Famous CNN
   - LeNet (1998)
   - AlexNet (2012)
   - ZFNet (2013)
   - VGGNet (2014)
   - GoogLeNet (2014)
   - ResNet (2016)
   - DenseNet (2017)
   - Many other CNN

GoogLeNet

["Going deeper with convolutions", Szegedy, W. Liu, et al. 2015]

Aim.

Increasing the depth and width of state-of-the-art convolutional neural networks while keeping the number of parameters small:

- Can approximate more complex functions
- while being robust to overfitting and computationally appealing.

How.

- Specifically, use of 1×1 convolution layers to reduce the number of parameters + apply filters of different sizes 3×3, 5×5 or 3×3 max pooling (on each feature maps).
- use auxiliary classifiers

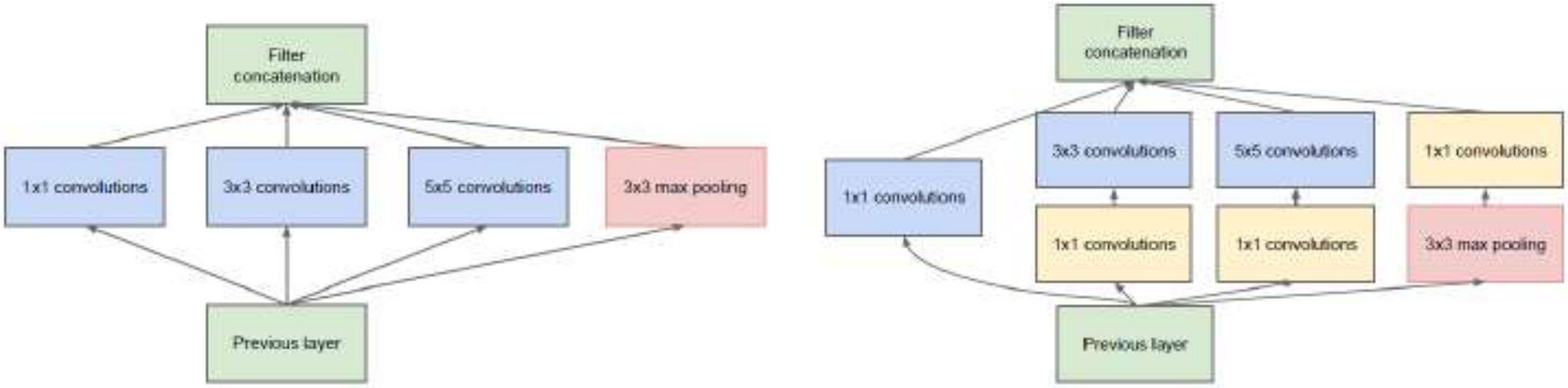
Details.

- All convolution layers use ReLU activation functions.
- Same spatial resolution for each feature map.

GoogLeNet - Inception module

Same spatial resolution for each feature map.

Use of 1×1 convolution layers to reduce the number of parameters then apply filters of different sizes 3×3, 5×5 or 3×3 max pooling (on each feature maps).



(a) Inception module, naïve version (b) Inception module with dimension reductions

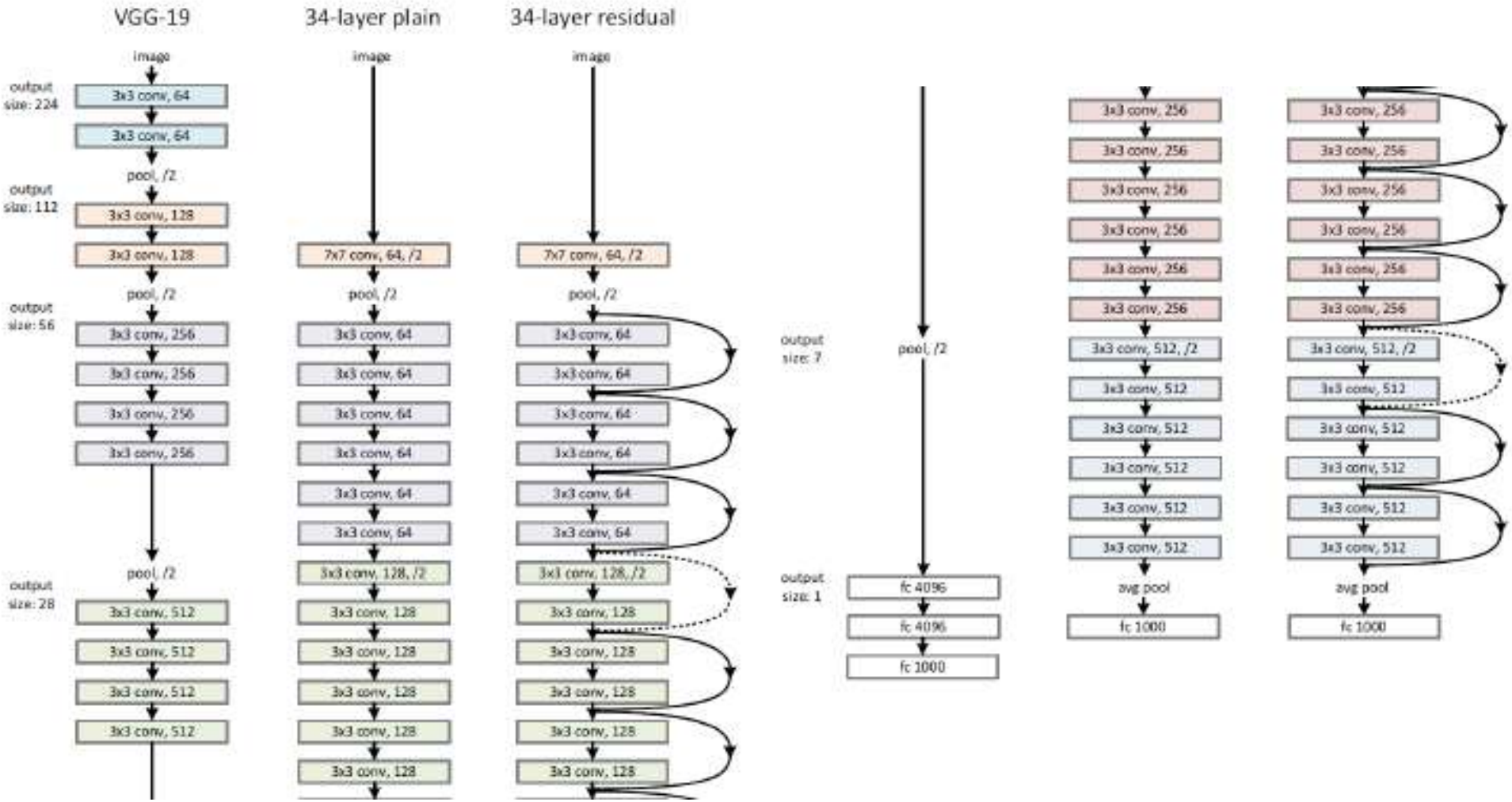
Figure 2: Inception module

Outline

5. Famous CNN
   - LeNet (1998)
   - AlexNet (2012)
   - ZFNet (2013)
   - VGGNet (2014)
   - GoogLeNet (2014)
   - ResNet (2016)
   - DenseNet (2017)
   - Many other CNN

ResNet (2016)

["Deep residual learning for image recognition", He et al. 2016]



Outline

5. Famous CNN
   - LeNet (1998)
   - AlexNet (2012)
   - ZFNet (2013)
   - VGGNet (2014)
   - GoogLeNet (2014)
   - ResNet (2016)
   - DenseNet (2017)
   - Many other CNN

DenseNet

[“Densely Connected Convolutional Networks.”, G. Huang et al. 2017]

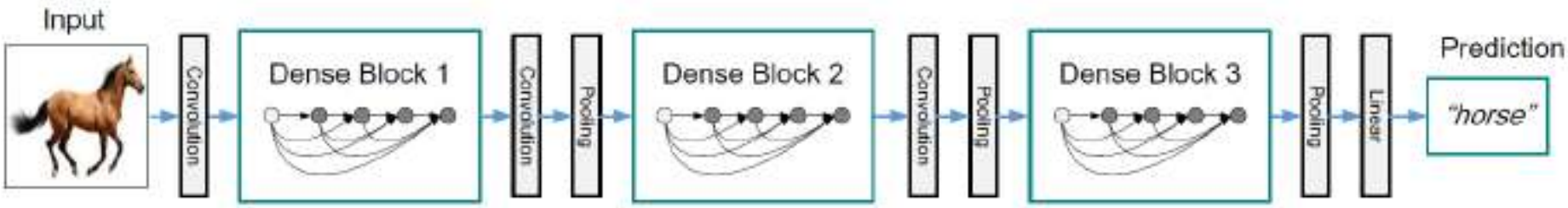


Figure: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling

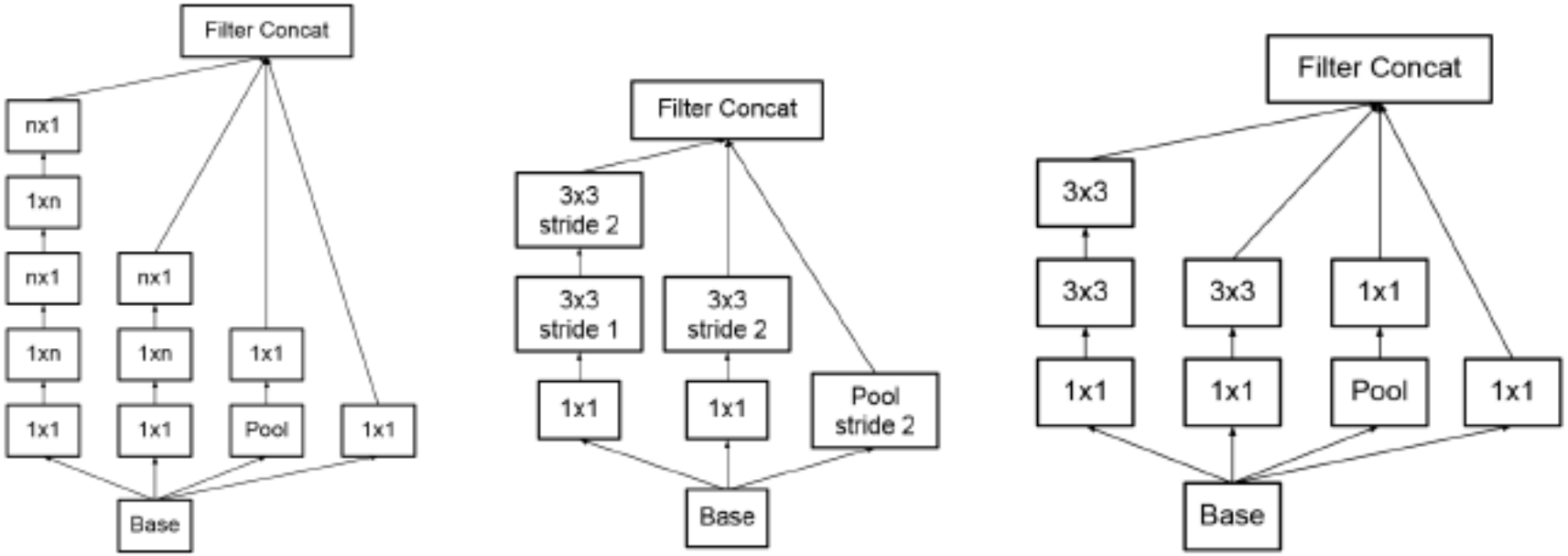
Outline

5. Famous CNN
   - LeNet (1998)
   - AlexNet (2012)
   - ZFNet (2013)
   - VGGNet (2014)
   - GoogLeNet (2014)
   - ResNet (2016)
   - DenseNet (2017)
   - Many other CNN

Inception V2-V3

Based on GoogLeNet Inception module

["Rethinking the inception architecture for computer vision", Szegedy, Vanhoucke, et al. 2016]



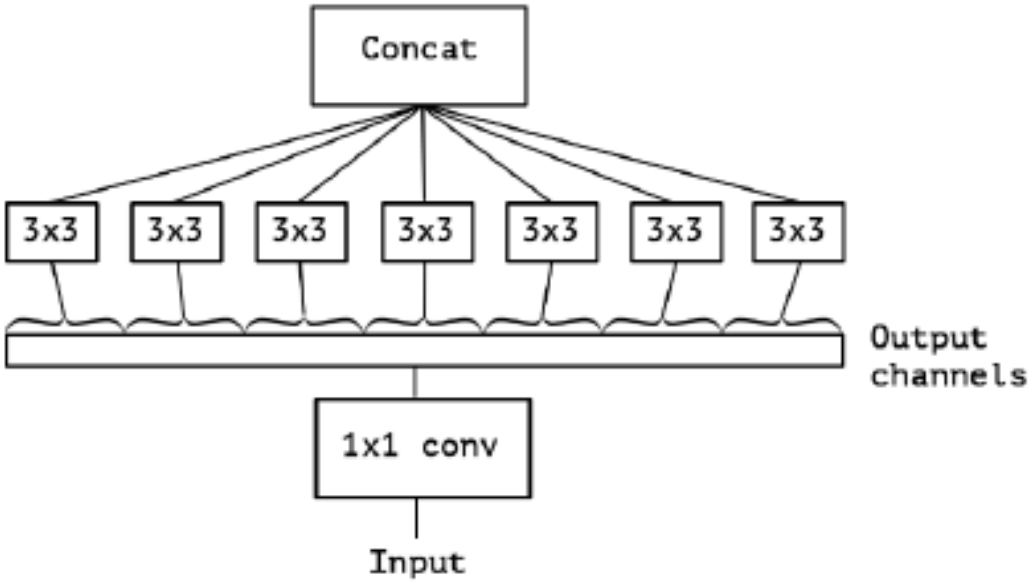
New ideas:

- Using asymmetric convolutions $1 \times n$ and $n \times 1$ (for $n = 3, 5, 7$) can be useful in the middle layers of the networks for feature maps of size $m \times m$ (for $12 \leq m \leq 20$).
- Label smoothing using a uniform distribution over labels

Xception

["Xception: Deep learning with depthwise separable convolutions", Chollet 2017]

Stands for "Extreme Inception" and builds upon Inception module in GoogLeNet.



The main ideas:

- Perform 1×1 convolutions
- Apply 3×3 (or other filter size) convolutions to each previous feature map (the one created by 1×1 convolutions) separately.

$\rightarrow$ Decoupled the depth (1×1 convolutions) and the spatial transformations (convolutions on each feature map separately).

[Abd+14] Ossama Abdel-Hamid et al. "Convolutional neural networks for speech recognition". In: *IEEE/ACM Transactions on audio, speech, and language processing* 22.10 (2014), pp. 1533–1545.

[Aiz64] Mark A Aizerman. "Theoretical foundations of the potential function method in pattern recognition learning". In: *Automation and remote control* 25 (1964), pp. 821–837.

[AP81] RS Anderssen and PM Prenter. "A formal comparison of methods proposed for the numerical solution of first kind integral equations". In: *The ANZIAM Journal* 22.4 (1981), pp. 488–500.

[Bah+16] Dzmitry Bahdanau et al. "End-to-end attention-based large vocabulary speech recognition". In: *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE. 2016, pp. 4945–4949.

[Ben+06] Yoshua Bengio et al. "Convex neural networks". In: *Advances in neural information processing systems*. 2006, pp. 123–130.

[Bis95] Chris M Bishop. "Training with noise is equivalent to Tikhonov regularization". In: *Neural computation* 7.1 (1995), pp. 108–116.

[BKC15] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". In: *arXiv preprint arXiv:1511.00561* (2015).

[Bla+20] Davis Blalock et al. "What is the state of neural network pruning?" In: *arXiv preprint arXiv:2003.03033* (2020).

[Blo62] Hans-Dieter Block. "The perceptron: A model for brain functioning. i". In: *Reviews of Modern Physics* 34.1 (1962), p. 123.

[Bre00] Leo Breiman. "Randomizing outputs to increase prediction accuracy". In: *Machine Learning* 40.3 (2000), pp. 229–242.

[Bri90] John S Bridle. "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition". In: *Neurocomputing*. Springer, 1990, pp. 227–236.

[BS85] Widrow Bernard and D Stearns Samuel. "Adaptive signal processing". In: *Englewood Cliffs, NJ, Prentice-Hall, Inc* 1 (1985), p. 491.

[BT07] Peter L Bartlett and Mikhail Traskin. "Adaboost is consistent". In: *Journal of Machine Learning Research* 8.Oct (2007), pp. 2347–2368.

[Cao+16] Zhe Cao et al. "Realtime multi-person 2d pose estimation using part affinity fields". In: *arXiv preprint arXiv:1611.08050* (2016).

[Che+16] Yan Chen et al. "Cnntracker: Online discriminative object tracking via deep convolutional neural network". In: *Applied Soft Computing* 38 (2016), pp. 1088–1098.

[Che+18] Liang-Chieh Chen et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2018), pp. 834–848.

[Cho17] François Chollet. "Xception: Deep learning with depthwise separable convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1251–1258.

[Chu+14] Junyoung Chung et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555* (2014).

[CLG01] Rich Caruana, Steve Lawrence, and C Lee Giles. "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping". In: *Advances in neural information processing systems*. 2001, pp. 402–408.

[CLS15] Guilhem Chéron, Ivan Laptev, and Cordelia Schmid. "P-cnn: Pose-based cnn features for action recognition". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 3218–3226.

[CM02] Dorin Comaniciu and Peter Meer. "Mean shift: A robust approach toward feature space analysis". In: *IEEE Transactions on pattern analysis and machine intelligence* 24.5 (2002), pp. 603–619.

[CPC16] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. "An analysis of deep neural network models for practical applications". In: *arXiv preprint arXiv:1605.07678* (2016).

[CSL13] Meng Cai, Yongzhe Shi, and Jia Liu. "Deep maxout neural networks for speech recognition". In: *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE. 2013, pp. 291–296.

[CUH15] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. "Fast and accurate deep network learning by exponential linear units (elus)". In: *arXiv preprint arXiv:1511.07289* (2015).

[Don+15] Jeffrey Donahue et al. "Long-term recurrent convolutional networks for visual recognition and description". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 2625–2634.

[EF15] David Eigen and Rob Fergus. "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2650–2658.

[Ege+20] Romain Egele et al. "AgEBO-Tabular: Joint Neural Architecture and Hyperparameter Search with Autotuned Data-Parallel Training for Tabular Data". In: *arXiv preprint arXiv:2010.16358* (2020).

[EH14] Ian Endres and Derek Hoiem. "Category-independent object proposals with diverse ranking". In: *IEEE transactions on pattern analysis and machine intelligence* 36.2 (2014), pp. 222–234.

[EKK11] Moataz El Ayadi, Mohamed S Kamel, and Fakhri Karray. "Survey on speech emotion recognition: Features, classification schemes, and databases". In: *Pattern Recognition* 44.3 (2011), pp. 572–587.

[ES02a] Douglas Eck and Juergen Schmidhuber. "A first look at music composition using lstm recurrent neural networks". In: *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale* 103 (2002).

[ES02b] Douglas Eck and Juergen Schmidhuber. "Finding temporal structure in music: Blues improvisation with LSTM recurrent networks". In: *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*. IEEE. 2002, pp. 747–756.

[Fan+14] Yuchen Fan et al. "TTS synthesis with bidirectional LSTM based recurrent neural networks". In: *Fifteenth Annual Conference of the International Speech Communication Association*. 2014.

[Fer+14] Raul Fernandez et al. "Prosody contour prediction with long short-term memory, bi-directional, deep recurrent neural networks.". In: *Interspeech*. 2014, pp. 2268–2272.

[FH04] Pedro F Felzenszwalb and Daniel P Huttenlocher. "Efficient graph-based image segmentation". In: *International journal of computer vision* 59.2 (2004), pp. 167–181.

[FM82] Kunihiko Fukushima and Sei Miyake. "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition". In: *Competition and cooperation in neural nets*. Springer, 1982, pp. 267–285.

[FS99] Yoav Freund and Robert E Schapire. "Large margin classification using the perceptron algorithm". In: *Machine learning* 37.3 (1999), pp. 277–296.

[Fuk75] Kunihiko Fukushima. "Cognitron: A self-organizing multilayered neural network". In: *Biological cybernetics* 20.3-4 (1975), pp. 121–136.

[GB10] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010, pp. 249–256.

[GBB11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep sparse rectifier neural networks". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. 2011, pp. 315–323.

[GFS07] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. "Multi-Dimensional Recurrent Neural Networks". In: *CoRR* abs/0705.2011 (2007). arXiv: 0705.2011. URL: http://arxiv.org/abs/0705.2011.

[GG16] Yarin Gal and Zoubin Ghahramani. "A theoretically grounded application of dropout in recurrent neural networks". In: *Advances in neural information processing systems*. 2016, pp. 1019–1027.

[GGM15] Georgia Gkioxari, Ross Girshick, and Jitendra Malik. "Actions and attributes from wholes and parts". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2470–2478.

[Gha+16] Marjan Ghazvininejad et al. "Generating topical poetry". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2016, pp. 1183–1191.

[Gir+14] Ross Girshick, Jeff Donahue, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[Gir+15] Ross Girshick, Forrest Iandola, et al. "Deformable part models are convolutional neural networks". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2015, pp. 437–446.

[GJ14] Alex Graves and Navdeep Jaitly. "Towards end-to-end speech recognition with recurrent neural networks". In: *International Conference on Machine Learning*. 2014, pp. 1764–1772.

[GK17] Lluís Gómez and Dimosthenis Karatzas. "Textproposals: a text-specific selective search algorithm for word spotting in the wild". In: *Pattern Recognition* 70 (2017), pp. 60–74.

[Goo+13a] Ian J Goodfellow, Mehdi Mirza, et al. "An empirical investigation of catastrophic forgetting in gradient-based neural networks". In: *arXiv preprint arXiv:1312.6211* (2013).

[Goo+13b] Ian J Goodfellow, David Warde-Farley, et al. "Maxout networks". In: *arXiv preprint arXiv:1302.4389* (2013).

[Gra+06] Alex Graves, Santiago Fernández, Faustino Gomez, et al. "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks". In: *Proceedings of the 23rd international conference on Machine learning*. ACM. 2006, pp. 369–376.

[Gra11] Alex Graves. "Practical variational inference for neural networks". In: *Advances in Neural Information Processing Systems*. 2011, pp. 2348–2356.

[Gra13] Alex Graves. "Generating sequences with recurrent neural networks". In: *arXiv preprint arXiv:1308.0850* (2013).

[Gre+17] Klaus Greff et al. "LSTM: A search space odyssey". In: *IEEE transactions on neural networks and learning systems* 28.10 (2017), pp. 2222–2232.

[GSS14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *arXiv preprint arXiv:1412.6572* (2014).

[Gu+15] Jiuxiang Gu et al. "Recent advances in convolutional neural networks". In: *arXiv preprint arXiv:1512.07108* (2015).

[He+14] Kaiming He et al. "Spatial pyramid pooling in deep convolutional networks for visual recognition". In: *European conference on computer vision*. Springer. 2014, pp. 346–361.

[He+15] Kaiming He et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[He+16] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[Heb49] DO Hebb. *The organization of behavior: a neuropsychological theory*. Wiley, 1949.

[Hin+12] Geoffrey E Hinton, Nitish Srivastava, et al. "Improving neural networks by preventing co-adaptation of feature detectors". In: *arXiv preprint arXiv:1207.0580* (2012).

[HK70] Arthur E Hoerl and Robert W Kennard. "Ridge regression: Biased estimation for nonorthogonal problems". In: *Technometrics* 12.1 (1970), pp. 55–67.

[Hon+15] Seunghoon Hong et al. "Online tracking by learning discriminative saliency map with convolutional neural network". In: *International Conference on Machine Learning*. 2015, pp. 597–606.

[HOT06] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets". In: *Neural computation* 18.7 (2006), pp. 1527–1554.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[Hu64] Michael Jen-Chao Hu. "Application of the adaline system to weather forecasting". PhD thesis. Department of Electrical Engineering, Stanford University, 1964.

[Hua+17] Gao Huang et al. "Densely Connected Convolutional Networks.". In: *CVPR*. Vol. 1. 2. 2017, p. 3.

[IS15] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[Jad+16] Max Jaderberg, Karen Simonyan, et al. "Reading text in the wild with convolutional neural networks". In: *International Journal of Computer Vision* 116.1 (2016), pp. 1–20.

[Jai+15] Navdeep Jaitly et al. "A neural transducer". In: *arXiv preprint arXiv:1511.04868* (2015).

[JGH96] Kam-Chuen Jim, C Lee Giles, and Bill G Horne. "An analysis of noise in recurrent neural networks: convergence and generalization". In: *IEEE Transactions on neural networks* 7.6 (1996), pp. 1424–1438.

[Ji+13] Shuiwang Ji et al. "3D convolutional neural networks for human action recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 35.1 (2013), pp. 221–231.

[JKF16] Justin Johnson, Andrej Karpathy, and Li Fei-Fei. "Densecap: Fully convolutional localization networks for dense captioning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4565–4574.

[JKL+09] Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, et al. "What is the best multi-stage architecture for object recognition?" In: *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE. 2009, pp. 2146–2153.

[Jou+16] Armand Joulin et al. "Bag of tricks for efficient text classification". In: *arXiv preprint arXiv:1607.01759* (2016).

[JVZ14] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. "Deep features for text spotting". In: *European conference on computer vision*. Springer. 2014, pp. 512–528.

[Kar+98] Orhan Karaali et al. "Text-to-speech conversion with neural networks: A recurrent TDNN approach". In: *arXiv preprint cs/9811032* (1998).

[KGB14] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. "A convolutional neural network for modelling sentences". In: *arXiv preprint arXiv:1404.2188* (2014).

[Kha+20] Asifullah Khan et al. "A survey of the recent architectures of deep convolutional neural networks". In: *Artificial Intelligence Review* 53.8 (2020), pp. 5455–5516.

[KM15] David Krueger and Roland Memisevic. "Regularizing rnns by stabilizing activations". In: *arXiv preprint arXiv:1511.08400* (2015).

[Kra+15] Jonathan Krause et al. "Fine-grained recognition without part annotations". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 5546–5555.

[Kru+16] David Krueger, Tegan Maharaj, et al. "Zoneout: Regularizing rnns by randomly preserving hidden activations". In: *arXiv preprint arXiv:1606.01305* (2016).

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[Lai+15] Siwei Lai et al. "Recurrent Convolutional Neural Networks for Text Classification.". In: *AAAI*. Vol. 333. 2015, pp. 2267–2273.

[LeC+89] Yann LeCun et al. "Generalization and network design strategies". In: *Connectionism in perspective* (1989), pp. 143–155.

[LeC+98a] Yann LeCun, Léon Bottou, Yoshua Bengio, et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[LeC+98b] Yann LeCun, Léon Bottou, Genevieve B Orr, et al. "Efficient backprop". In: *Neural networks: Tricks of the trade*. Springer, 1998, pp. 9–50.

[Lee+15] Chen-Yu Lee et al. "Deeply-supervised nets". In: *Artificial Intelligence and Statistics*. 2015, pp. 562–570.

[LH15] Ming Liang and Xiaolin Hu. "Recurrent convolutional neural network for object recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3367–3375.

[Lin+15] Di Lin et al. "Deep lac: Deep localization, alignment and classification for fine-grained recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1666–1674.

[Liu+16] Wei Liu et al. "Ssd: Single shot multibox detector". In: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[LJH15] Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. "A simple way to initialize recurrent networks of rectified linear units". In: *arXiv preprint arXiv:1504.00941* (2015).

[LJL16] Yongxi Lu, Tara Javidi, and Svetlana Lazebnik. "Adaptive object detection using adjacency and zoom prediction". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2351–2359.

[LLP16] Hanxi Li, Yi Li, and Fatih Porikli. "Deeptrack: Learning discriminative feature representations online for robust visual tracking". In: *IEEE Transactions on Image Processing* 25.4 (2016), pp. 1834–1848.

[LT15] Jinkyu Lee and Ivan Tashev. "High-level feature representation using recurrent neural network for speech emotion recognition". In: (2015).

[Mar10] James Martens. "Deep learning via Hessian-free optimization.". In: *ICML*. Vol. 27. 2010, pp. 735–742.

[Mas+00] Llew Mason et al. "Boosting algorithms as gradient descent". In: *Advances in neural information processing systems*. 2000, pp. 512–518.

[MHN13] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. "Rectifier nonlinearities improve neural network acoustic models". In: *Proc. icml*. Vol. 30. 1. 2013, p. 3.

[Mik+11] Tomáš Mikolov et al. "Empirical evaluation and combination of advanced language modeling techniques". In: *Twelfth Annual Conference of the International Speech Communication Association*. 2011.

[Mik+13] Tomas Mikolov, Ilya Sutskever, et al. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.

[Mik+14] Tomas Mikolov, Armand Joulin, et al. "Learning longer memory in recurrent neural networks". In: *arXiv preprint arXiv:1412.7753* (2014).

[MM15] Dmytro Mishkin and Jiri Matas. "All you need is a good init". In: *arXiv preprint arXiv:1511.06422* (2015).

[Moo+15] Taesup Moon et al. "Rnndrop: A novel dropout for rnns in asr". In: *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE. 2015, pp. 65–70.

[MP43] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[MP69] Marvin Minsky and Seymour Papert. "Perceptrons.". In: (1969).

[MR13] Mehryar Mohri and Afshin Rostamizadeh. "Perceptron mistake bounds". In: *arXiv preprint arXiv:1305.0208* (2013).

[NH92] Steven J Nowlan and Geoffrey E Hinton. "Simplifying neural networks by soft weight-sharing". In: *Neural computation* 4.4 (1992), pp. 473–493.

[NLO16] Duc Thanh Nguyen, Wanqing Li, and Philip O Ogunbona. "Human detection from images and videos: A survey". In: *Pattern Recognition* 51 (2016), pp. 148–175.

[Nov63] Albert B Novikoff. *On convergence proofs for perceptrons*. Tech. rep. STANFORD RESEARCH INST MENLO PARK CA, 1963.

[NP95] Steven J Nowlan and John C Platt. "A convolutional neural network hand tracker". In: *Advances in neural information processing systems* (1995), pp. 901–908.

[Ola96] Mikel Olazaran. "A sociological study of the official history of the perceptrons controversy". In: *Social Studies of Science* 26.3 (1996), pp. 611–659.

[Pap+02] Kishore Papineni et al. "BLEU: a method for automatic evaluation of machine translation". In: *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics. 2002, pp. 311–318.

[Pau+14] Mattis Paulin et al. "Transformation pursuit for image classification". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 3646–3653.

[PC14] Pedro HO Pinheiro and Ronan Collobert. "Recurrent convolutional neural networks for scene labeling". In: *31st International Conference on Machine Learning (ICML)*. EPFL-CONF-199822. 2014.

[Pis+13] Leonid Pishchulin et al. "Poselet conditioned pictorial structures". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 588–595.

[PMB13] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. "On the difficulty of training recurrent neural networks". In: *International Conference on Machine Learning*. 2013, pp. 1310–1318.

[PRR15] Peter Potash, Alexey Romanov, and Anna Rumshisky. "GhostWriter: Using an LSTM for automatic rap lyric generation". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2015, pp. 1919–1924.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[Red+16] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[Ren+15] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[Rif+11] Salah Rifai et al. "Adding noise to the input of a model trained with a regularized objective". In: *arXiv preprint arXiv:1104.3250* (2011).

[Rip07] Brian D Ripley. *Pattern recognition and neural networks*. Cambridge university press, 2007.

[Ros60] Frank Rosenblatt. "Perceptron simulation experiments". In: *Proceedings of the IRE* 48.3 (1960), pp. 301–309.

[Ros61] Frank Rosenblatt. *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. Tech. rep. CORNELL AERONAUTICAL LAB INC BUFFALO NY, 1961.

[Rus+15] Olga Russakovsky et al. "Imagenet large scale visual recognition challenge". In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252.

[RZL17] Prajit Ramachandran, Barret Zoph, and Quoc V Le. "Searching for activation functions". In: *arXiv preprint arXiv:1710.05941* (2017).

[Sal+17] Hojjat Salehinejad et al. "Recent Advances in Recurrent Neural Networks". In: *arXiv preprint arXiv:1801.01078* (2017).

[SB17] Justin Salamon and Juan Pablo Bello. "Deep convolutional neural networks and data augmentation for environmental sound classification". In: *IEEE Signal Processing Letters* 24.3 (2017), pp. 279–283.

[SB98] Holger Schwenk and Yoshua Bengio. "Training methods for adaptive boosting of neural networks". In: *Advances in neural information processing systems*. 1998, pp. 647–653.

[Sch99] Michael Schuster. "On supervised learning from sequential data with applications for speech recognition". In: *Daktaro disertacija, Nara Institute of Science and Technology* 45 (1999).

[SG16] Tom Sercu and Vaibhava Goel. "Advances in very deep convolutional neural networks for lvcsr". In: *arXiv preprint arXiv:1604.01792* (2016).

[SGS15a] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. "Training very deep networks". In: *Advances in neural information processing systems*. 2015, pp. 2377–2385.

[SGS15b] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. "Highway networks". In: *arXiv preprint arXiv:1505.00387* (2015).

[Shu+16] Bing Shuai et al. "Dag-recurrent neural networks for scene labeling". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3620–3629.

[Sri+14] Nitish Srivastava et al. "Dropout: A simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[SSB14] Haşim Sak, Andrew Senior, and Françoise Beaufays. "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition". In: *arXiv preprint arXiv:1402.1128* (2014).

[SSB16] Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. "Recurrent dropout without memory loss". In: *arXiv preprint arXiv:1603.05118* (2016).

[SSN12] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. "LSTM neural networks for language modeling". In: *Thirteenth annual conference of the international speech communication association*. 2012.

[STE13] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. "Deep neural networks for object detection". In: *Advances in neural information processing systems*. 2013, pp. 2553–2561.

[SZ14a] Karen Simonyan and Andrew Zisserman. "Two-stream convolutional networks for action recognition in videos". In: *Advances in neural information processing systems*. 2014, pp. 568–576.

[SZ14b] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[Sze+15] Christian Szegedy, Wei Liu, et al. "Going deeper with convolutions". In: Cvpr. 2015.

[Sze+16] Christian Szegedy, Vincent Vanhoucke, et al. "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[TGK63] LR Talbert, GF Groner, and JS Koford. "Real-Time Adaptive Speech-Recognition System". In: *The Journal of the Acoustical Society of America* 35.5 (1963), pp. 807–807.

[Tib96] Robert Tibshirani. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1996), pp. 267–288.

[TQL15] Duyu Tang, Bing Qin, and Ting Liu. "Document modeling with gated recurrent neural network for sentiment classification". In: *Proceedings of the 2015 conference on empirical methods in natural language processing*. 2015, pp. 1422–1432.

[TR93] Christine Tuerk and Tony Robinson. "Speech synthesis using artificial neural networks trained on cepstral coefficients". In: *Third European Conference on Speech Communication and Technology*. 1993.

[Tri+16] George Trigeorgis et al. "Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network". In: *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE. 2016, pp. 5200–5204.

[TS14] Alexander Toshev and Christian Szegedy. "Deeppose: Human pose estimation via deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1653–1660.

[TV15] Sachin S Talathi and Aniket Vartak. "Improving performance of recurrent neural network with relu nonlinearity". In: *arXiv preprint arXiv:1511.03771* (2015).

[Uij+13] Jasper RR Uijlings et al. "Selective search for object recognition". In: *International journal of computer vision* 104.2 (2013), pp. 154–171.

[Van+16] Aäron Van Den Oord et al. "WaveNet: A generative model for raw audio.". In: *SSW*. 2016, p. 125.

[Ven+14] Subhashini Venugopalan et al. "Translating videos to natural language using deep recurrent neural networks". In: *arXiv preprint arXiv:1412.4729* (2014).

[Vin+15] Oriol Vinyals et al. "Show and tell: A neural image caption generator". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3156–3164.

[VML94] Régis Vaillant, Christophe Monrocq, and Yann Le Cun. "Original approach for the localisation of objects in images". In: *IEE Proceedings-Vision, Image and Signal Processing* 141.4 (1994), pp. 245–250.

[VZQ15] Vivek Veeriah, Naifan Zhuang, and Guo-Jun Qi. "Differential recurrent neural networks for action recognition". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE. 2015, pp. 4041–4049.

[Wah87] Grace Wahba. "Three topics in ill-posed problems". In: *Inverse and ill-posed problems*. Elsevier, 1987, pp. 37–51.

[Wan+12] Tao Wang et al. "End-to-end text recognition with convolutional neural networks". In: *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE. 2012, pp. 3304–3308.

[Wan+13] Li Wan et al. "Regularization of neural networks using dropconnect". In: *International conference on machine learning*. 2013, pp. 1058–1066.

[WH60] Bernard Widrow and Marcian E Hoff. *Adaptive switching circuits*. Tech. rep. Stanford Univ Ca Stanford Electronics Labs, 1960.

[Wie15] Wessel N van Wieringen. "Lecture notes on ridge regression". In: *arXiv preprint arXiv:1509.09169* (2015).

[WM13] Sida Wang and Christopher Manning. "Fast dropout training". In: *international conference on machine learning*. 2013, pp. 118–126.

[Wöl+08] Martin Wöllmer et al. "Abandoning emotion classes-towards continuous emotion recognition with modelling of long-range dependencies". In: *Proc. 9th Interspeech 2008 incorp. 12th Australasian Int. Conf. on Speech Science and Technology SST 2008, Brisbane, Australia*. 2008, pp. 597–600.

[WW88] Capt Rodney Winter and B Widrow. "Madaline Rule II: a training algorithm for neural networks". In: *Second Annual International Conference on Neural Networks*. 1988, pp. 1–401.

[WWW18] Zhenhua Wang, Xingxing Wang, and Gang Wang. "Learning fine-grained features via a CNN tree for large-scale classification". In: *Neurocomputing* 275 (2018), pp. 1231–1240.

[WYW17] Yuting Wei, Fanny Yang, and Martin J Wainwright. "Early stopping for kernel boosting algorithms: A general analysis with localized complexities". In: *Advances in Neural Information Processing Systems*. 2017, pp. 6067–6077.

[Xia+14] Tianjun Xiao et al. "Error-driven incremental learning in deep convolutional neural network for large-scale image classification". In: *Proceedings of the 22nd ACM international conference on Multimedia*. ACM. 2014, pp. 177–186.

[XT15] Saining Xie and Zhuowen Tu. "Holistically-nested edge detection". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1395–1403.

[Xu+15a] Bing Xu et al. "Empirical evaluation of rectified activations in convolutional network". In: *arXiv preprint arXiv:1505.00853* (2015).

[Xu+15b] Kelvin Xu et al. "Show, attend and tell: Neural image caption generation with visual attention". In: *International conference on machine learning*. 2015, pp. 2048–2057.

[Yan+15] Zhicheng Yan et al. "HD-CNN: hierarchical deep convolutional neural networks for large scale visual recognition". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2740–2748.

[Yan+16] Wei Yang et al. "End-to-end learning of deformable mixture of parts and deep convolutional neural networks for human pose estimation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 3073–3082.

[Yao+15] Li Yao et al. "Video description generation incorporating spatio-temporal features and a soft-attention mechanism". In: *arXiv preprint arXiv:1502.08029* (2015).

[Yoo+15] Donggeun Yoo et al. "Attentionnet: Aggregating weak directions for accurate object detection". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2659–2667.

[YP15] Heng Yang and Ioannis Patras. "Mirror, mirror on the wall, tell me, is the error small?" In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 4685–4693.

[YRC07] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. "On early stopping in gradient descent learning". In: *Constructive Approximation* 26.2 (2007), pp. 289–315.

[YS16] Wenpeng Yin and Hinrich Schütze. "Multichannel variable-size convolution for sentence classification". In: *arXiv preprint arXiv:1603.04513* (2016).

[ZF14] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[ZH05] Hui Zou and Trevor Hastie. "Regularization and variable selection via the elastic net". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.2 (2005), pp. 301–320.

[Zha+16] Yu Zhang et al. "Weakly supervised fine-grained categorization with part-based image representation". In: *IEEE Transactions on Image Processing* 25.4 (2016), pp. 1713–1725.

[ZL14] Xingxing Zhang and Mirella Lapata. "Chinese poetry generation with recurrent neural networks". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 670–680.

[ZS15] Heiga Zen and Haşim Sak. "Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis". In: *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE. 2015, pp. 4470–4474.

[ZY+05] Tong Zhang, Bin Yu, et al. "Boosting with early stopping: Convergence and consistency". In: *The Annals of Statistics* 33.4 (2005), pp. 1538–1579.