Convolutional neural networks

E. Decencière

MINES ParisTech
PSL Research University
Center for Mathematical Morphology

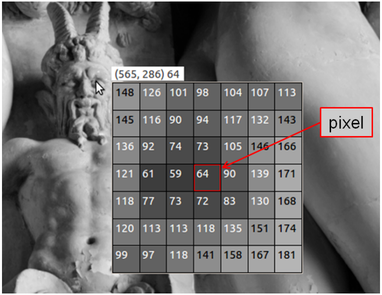
Contents

Contents

A picture is worth a thousand words

Definition

- Classically, an image is a matrix of values belonging to $[0, \ldots, 255]$ (grey level images) or to $[0, \ldots, 255]^3$ (color images).
- More generally, an image is a q-dimensional array of values belonging to R^d.

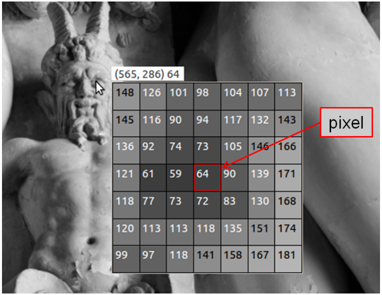


Grey level values around the left eye of the faun

A picture is worth a thousand words

Definition

- Classically, an image is a matrix of values belonging to $[0, \ldots, 255]$ (grey level images) or to $[0, \ldots, 255]^3$ (color images).
- More generally, an image is a q-dimensional array of values belonging to R^d.



Grey level values around the left eye of the faun

Examples

- Grey level 2D images: infrared, microscopy, topography
- Colour images: camera photos
- Grey level 3D images: computed tomography scan
- Colour image sequences: video, motion pictures
- $d > 3$: multi-spectral imaging

What is special about images?

- Local structure
- Spatial redundancy
- Scale redundancy [Glasner et al., 2009]

Extracting semantic information from an image

- Where is the phone? (localization task)
- How many mugs are there? (quantification task)
- Is there a window in the room?
- At what time of the day was the photograph taken?

Extracting semantic information from an image

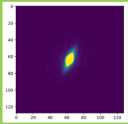
- Where is the phone? (localization task)
- How many mugs are there? (quantification task)
- Is there a window in the room?
- At what time of the day was the photograph taken?

Designing computer vision systems that are able to extract semantic information from an image is a difficult task.

Image analysis applications

image → valeur

→ oiseau



→ 50,2

Tuccillo et al, 2018

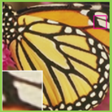
image → image

(segmentation)

→

Base COCO

→



Dong et al., ECCV 2014

Génération d’images

→

Isola et al, 2017

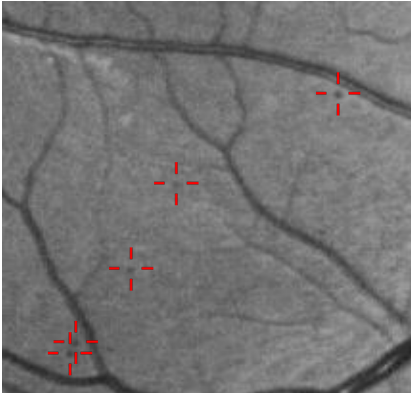
Image analysis applications

- classification
- quantification
- object localization
- transformation (filtering, in-painting, editing, colorization...)
- segmentation
- image caption generation
- 2D to 3D (stereo matching, 3D reconstruction, ...)
- motion estimation
- style transfer
- compression
- anomalous image detection
- image generation
- etc.

Classical image processing approach

Detection or segmentation task.

- Build a geometrical model for the objects of interest
- Implement this model using image processing operators



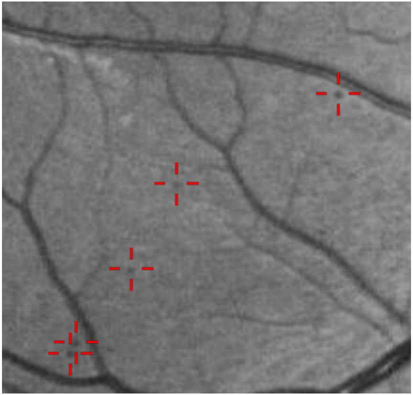
Detail of eye fundus image with microaneurysms to be detected

Credits: [Zhang et al., 2011]

Classical image processing approach

Detection or segmentation task.

- Build a geometrical model for the objects of interest
- Implement this model using image processing operators



Detail of eye fundus image with microaneurysms to be detected

⊕ This approach works when the objects are not too complex

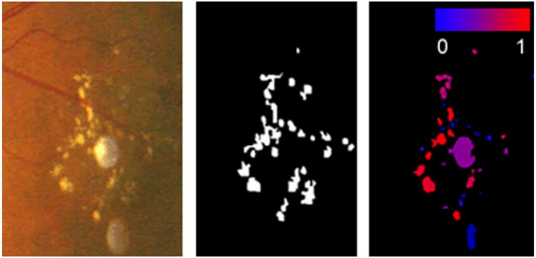
⊕ Interpretability

⊖ Often objects are difficult to model explicitly

Credits: [Zhang et al., 2011]

Classical machine learning approach

- Compute features from the image
- Apply machine learning to those features

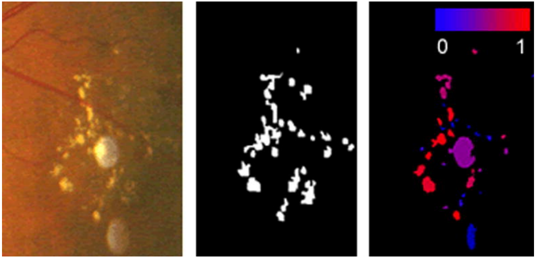


Exudates segmentation: original image, ground-truth and candidates with associated probabilities obtained with machine learning

Credits: [Zhang et al., 2014]

Classical machine learning approach

- Compute features from the image
- Apply machine learning to those features



Exudates segmentation: original image, ground-truth and candidates with associated probabilities obtained with machine learning

⊕ Works well with the right features

⊕ Interpretability

⊖ An expert is required to define those features

⊖ Annotated data is required

Credits: [Zhang et al., 2014]

Deep learning approach

- Directly take as input the image pixels
- The network is supposed to build its own features

$\oplus$ Good (impressive!) results

$\ominus$ A large amount of annotated data is required

$\ominus$ Extensive computing resources needed (for training)

$\ominus$ Lack of interpretability

The role of annotated image databases

Image databases including *annotations* (typically some kind of high level information) are essential to the development of *supervised* machine learning methods for image analysis.

Annotations: examples

- Image class
- Measure(s) obtained from the image
- Position of objects within the image
- Segmentation

MNIST database [Lecun et al., 1998]

- The Modified National Institute of Standards and Technology (MNIST) database contains $60\,000$ training images of hand-written digits, and $10,000$ test images.
- Image size: 28×28
- It has been used since 1998
- Human performance on a similar database (NIST) is reported to be around 1.5% error [Simard et al., 1993]
- Best methods, based on convolutional neural networks, give around 0.21% test error.

MNIST database

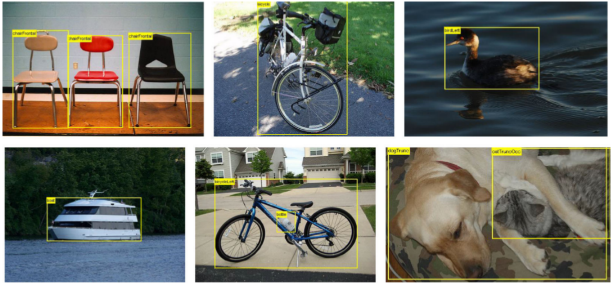
Credits: Images from MNIST assembled by Josef Stepan (licensed under CC BY-SA 4.0)

Pascal VOC project [Everingham et al., 2010, Everingham et al., 2014]

This project organized a challenge from 2005 to 2012, divided into several tasks, including an image classification task.

Pascal VOC image classification task (2012)

Train/val: $11\,540$ images where the presence of 20 categories of objects was annotated. The test dataset is unknown and tests are run online (still available).



Credits: From [Everingham et al., 2014]

ImageNet project [Russakovsky et al., 2015]

Between 2010 and 2017 ImageNet organized an annual challenge: The ImageNet Large Scale Visual Recognition Challenge (ILSVRC). It represented a breakthrough in the design of image analysis challenges by its size.

Image classification task

- Training: $1\,281\,167$; validation: $50\,000$; test: $100\,000$.
- $1\,000$ classes (90 dog breeds!).

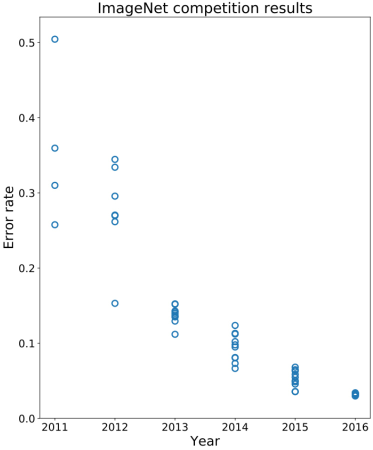
ImageNet projet

Examples from the *acoustic guitar* class

Deep learning achievements

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

2012: *AlexNet* [Krizhevsky et al., 2012] won this challenge by a large margin



Evolution of top-5 error rate

Deep learning achievements (cont.)

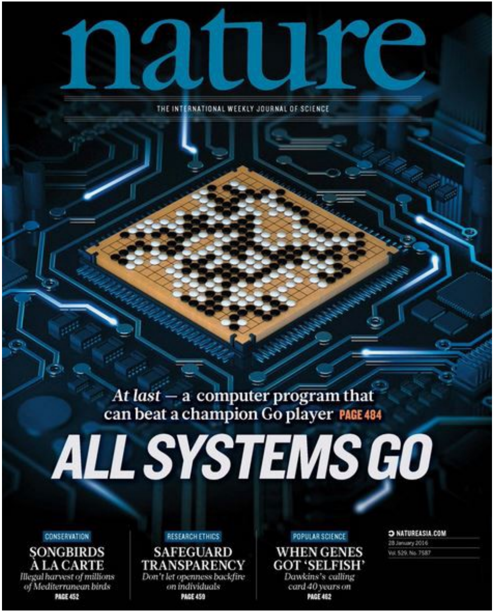
- 2011: first super-human performance, IJCNN 2011 traffic sign recognition contest [Cireșan et al., 2011]

- 2012: segmentation competition (neuronal membranes in electron microscopy images [Cireșan et al., 2012])

Credits: [Stallkamp et al., 2011]

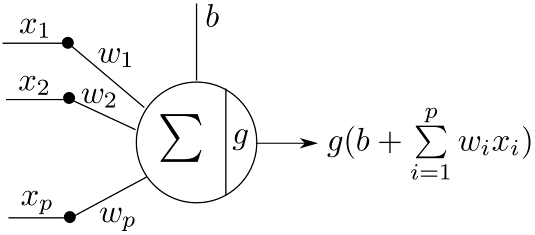
Deep learning achievements (cont.)

- 2016: AlphaGo beats Lee Sedol, one of the best go players, in a 5-game match



Contents

Reminder: Artificial neuron



- $b, w_1, \ldots, w_n$ are the neuron parameters, to be learnt
- g is the activation function

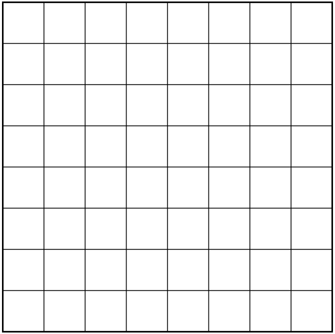
Image classification problem

Classification problem:

- Input: image
- Output: class y chosen from a set of labels: $\{label_1, label_2, \ldots, label_q\}$

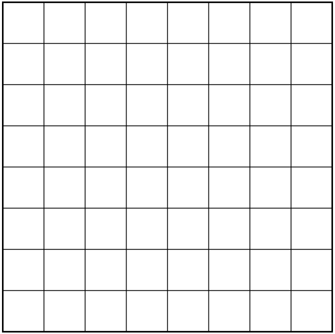
Input image, input neurons

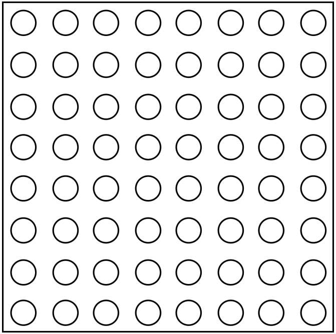
In the scalar case (single-valued images), each input pixel is considered as an input neuron.



Input image, input neurons

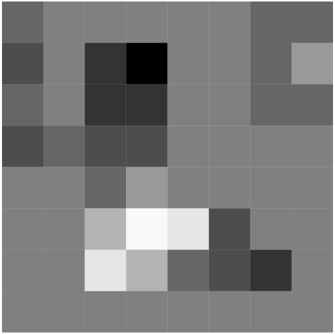
In the scalar case (single-valued images), each input pixel is considered as an input neuron.

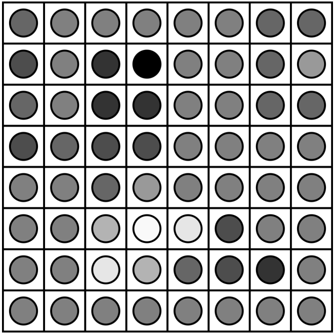




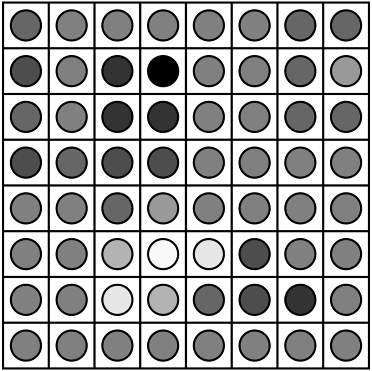
Input image, input neurons

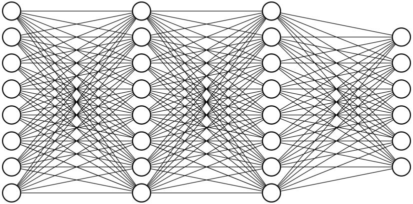
In the scalar case (single-valued images), each input pixel is considered as an input neuron.

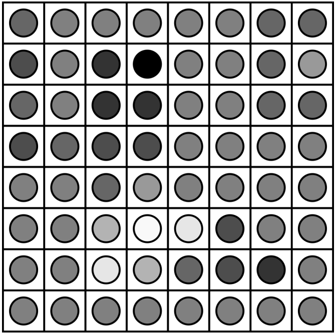




How are we going to apply our NN to an image?







What approach would you suggest to apply our fully connected NN to an image?

1/ Integrate along the horizontal or vertical axis, in order to get rid of a dimension

2/ Apply the network to each row or column separately

3/ Transform the image into a vector, by rearranging the pixels

Class coding

If there are q possible classes, then a class will be coded as a vector $\mathbf{y}$ of length q. If its class is r then for $0 \leq i < q$:

$$\mathbf{y}[i] = \begin{cases} 1, & \text{if } i = r \\ 0, & \text{otherwise} \end{cases}$$

Example with 3 classes

- Label $0 \longmapsto [1, 0, 0]$
- Label $1 \longmapsto [0, 1, 0]$
- Label $2 \longmapsto [0, 0, 1]$

This is called one-hot encoding. The resulting vector is a one-hot vector.

Why is one-hot encoding used, instead of integers?

1/ The network performs better in the $[0, 1]$ range

2/ One-hot encoding makes the labels symmetric, i.e. permutation invariant.

One-hot encoding

- Label $0 \longmapsto [1, 0, 0]$
- Label $1 \longmapsto [0, 1, 0]$
- Label $2 \longmapsto [0, 0, 1]$

Integer encoding

- Label $0 \longmapsto 0$
- Label $1 \longmapsto 1$
- Label $2 \longmapsto 2$

Why is one-hot encoding used, instead of integers?

1/ The network performs better in the $[0, 1]$ range

2/ One-hot encoding makes the labels symmetric, i.e. permutation invariant.

One-hot encoding

- Label $0 \longmapsto [1, 0, 0]$
- Label $1 \longmapsto [0, 1, 0]$
- Label $2 \longmapsto [0, 0, 1]$

Integer encoding

- Label $0 \longmapsto 0$
- Label $1 \longmapsto 1$
- Label $2 \longmapsto 2$

- Using integers to represent the labels would bring an inductive bias into the model: there will be an explicit order between the labels.
- Using one-hot encoding, we make the labels permutation invariant

Activations

Different activations (typically ReLU) can be used in the intermediate layers.

Concerning the last layer: Given that the aim is a vector containing zeros except for a one, two designs are commonly used:

- Use a sigmoid as last activation
- Last layer: a softmax operator

Softmax operator

Definition

The softmax operator $\sigma : \mathbb{R}^d \to \mathbb{R}^d$ is given by:

$$\forall \mathbf{x} \in \mathbb{R}^d, \forall k \in \{1, \ldots, d\} : \quad \sigma(\mathbf{x})_k = \frac{e^{\mathbf{x}_k}}{\sum\limits_{i=1}^{d} e^{\mathbf{x}_i}}$$

Softmax operator

Definition

The softmax operator $\sigma : \mathbb{R}^d \to \mathbb{R}^d$ is given by:

$$\forall \mathbf{x} \in \mathbb{R}^d, \forall k \in \{1, \dots, d\} : \quad \sigma(\mathbf{x})_k = \frac{e^{\mathbf{x}_k}}{\sum\limits_{i=1}^{d} e^{\mathbf{x}_i}}$$

Some properties

- $0 < \sigma(\mathbf{x})_k < 1$
- $\sum\limits_{i=1}^{d} \sigma(\mathbf{x})_i = 1$

Softmax operator

Definition

The softmax operator $\sigma : \mathbb{R}^d \to \mathbb{R}^d$ is given by:

$$\forall \mathbf{x} \in \mathbb{R}^d, \forall k \in \{1, \ldots, d\} : \quad \sigma(\mathbf{x})_k = \frac{e^{\mathbf{x}_k}}{\sum\limits_{i=1}^{d} e^{\mathbf{x}_i}}$$

Some properties

- $0 < \sigma(\mathbf{x})_k < 1$
- $\sum\limits_{i=1}^{d} \sigma(\mathbf{x})_i = 1$

Example

$$\mathbf{x} = \begin{pmatrix} 10.1 \\ 0 \\ -4.3 \\ 1.33 \end{pmatrix} \quad \sigma(\mathbf{x}) \approx \begin{pmatrix} 0.9998 \\ 0.000041 \\ 0.00000056 \\ 0.00016 \end{pmatrix}$$

Loss function for classification: cross-entropy

The preferred loss function for classification is cross-entropy:

For $\mathbf{y}$ in $\{0,1\}^d$ and $\hat{\mathbf{y}}$ in $]0,1[^d$:

$$H(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{d} \mathbf{y}_i \log(\hat{\mathbf{y}}_i)$$

Reminder – hat notation: $\hat{\mathbf{y}}$ is the prediction that is supposed to be close to $\mathbf{y}$.

Loss function for classification: cross-entropy

The preferred loss function for classification is cross-entropy:

For $\mathbf{y}$ in $\{0,1\}^d$ and $\hat{\mathbf{y}}$ in $]0,1[^d$:

$$H(\mathbf{y},\hat{\mathbf{y}}) = -\sum_{i=1}^{d} \mathbf{y}_i \log(\hat{\mathbf{y}}_i)$$

Reminder – hat notation: $\hat{\mathbf{y}}$ is the prediction that is supposed to be close to $\mathbf{y}$.

Remarks:

- Here $\mathbf{y}$ is a one-hot vector: $\mathbf{y}_i = 1$ for $i = r$ and $\mathbf{y}_i = 0$ otherwise. Therefore the cross-entropy will "push" $\hat{\mathbf{y}}_r$ towards 1. Why will the other elements of $\hat{\mathbf{y}}$ be pushed towards 0?

Loss function for classification: cross-entropy

The preferred loss function for classification is cross-entropy:

For $\mathbf{y}$ in $\{0,1\}^d$ and $\hat{\mathbf{y}}$ in $]0,1[^d$:

$$H(\mathbf{y},\hat{\mathbf{y}}) = -\sum_{i=1}^{d} \mathbf{y}_i \log(\hat{\mathbf{y}}_i)$$

Reminder – hat notation: $\hat{\mathbf{y}}$ is the prediction that is supposed to be close to $\mathbf{y}$.

Remarks:

- Here $\mathbf{y}$ is a one-hot vector: $\mathbf{y}_i = 1$ for $i = r$ and $\mathbf{y}_i = 0$ otherwise. Therefore the cross-entropy will "push" $\hat{\mathbf{y}}_r$ towards 1. Why will the other elements of $\hat{\mathbf{y}}$ be pushed towards 0? Because $\hat{\mathbf{y}}$ is typically computed by a softmax.

Loss function for classification: cross-entropy

The preferred loss function for classification is cross-entropy:

For $\mathbf{y}$ in $\{0, 1\}^d$ and $\hat{\mathbf{y}}$ in $]0, 1[^d$:

$$H(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{d} \mathbf{y}_i \log(\hat{\mathbf{y}}_i)$$

Reminder – hat notation: $\hat{\mathbf{y}}$ is the prediction that is supposed to be close to $\mathbf{y}$.

Remarks:

- Here $\mathbf{y}$ is a one-hot vector: $\mathbf{y}_i = 1$ for $i = r$ and $\mathbf{y}_i = 0$ otherwise. Therefore the cross-entropy will "push" $\hat{\mathbf{y}}_r$ towards 1. Why will the other elements of $\hat{\mathbf{y}}$ be pushed towards 0? Because $\hat{\mathbf{y}}$ is typically computed by a softmax.
- The binary cross-entropy we previously saw is a particular case of cross-entropy.

Image classification with a fully-connected NN

Input

The input image, containing p pixels, is transformed into a vector of length p.

Output

For q classes, the output will be a vector of length q.

Image classification with a fully-connected NN

Input

The input image, containing p pixels, is transformed into a vector of length p.

Output

For q classes, the output will be a vector of length q.

Example: image of size 4×2, 6 possible classes

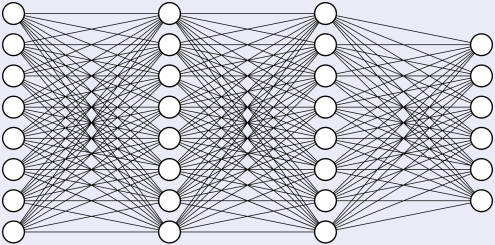
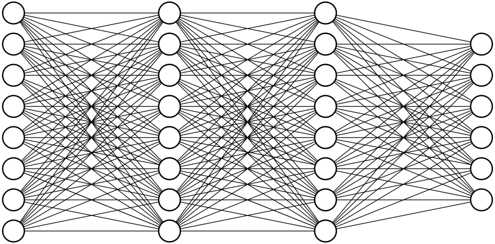
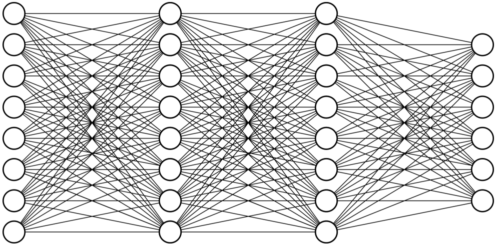


Image classification using a fully-connected NN



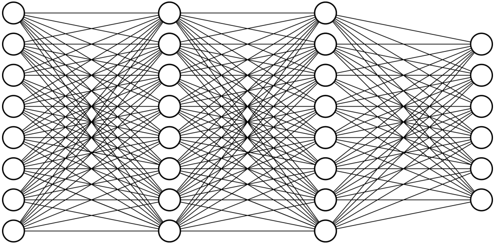
- A small image easily contains $100\,000$ pixels

Image classification using a fully-connected NN



- A small image easily contains $100\,000$ pixels
- The number of parameters between two layers of that size is $10^5 \times (10^5 + 1)$!

Image classification using a fully-connected NN



- A small image easily contains $100\,000$ pixels
- The number of parameters between two layers of that size is $10^5 \times (10^5 + 1)$!
- This approach is only possible for very small images

Conclusion on fully-connected networks for image classification

Fully-connected layers scale badly to large size images.
Today:

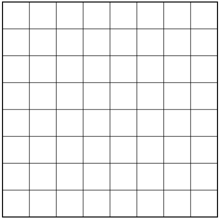
- NN solely composed of fully-connected layers are almost never used for image analysis [1].
- Fully-connected layers are mainly used in the middle (auto-encoders) or at the end (classification) of the pipeline.

[1] However, transformers [Dosovitskiy et al., 2021] have brought new ways to use fully-connected layers.

Contents

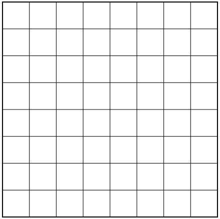
Layers representation

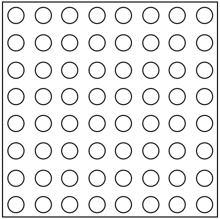
- For illustration purposes, in the following slides images and layers will be displayed as rows of neurons – as sections of 2D arrays.
- Only some connections between neurons are represented. Each such connection is associated to a weight. The biases are not represented, to avoid clutter.



Layers representation

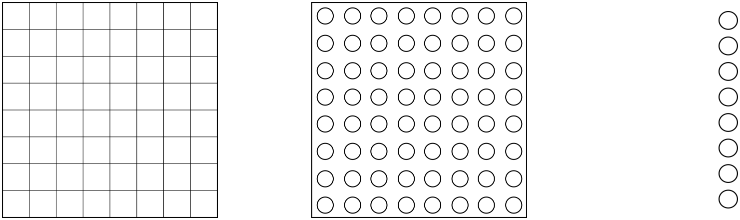
- For illustration purposes, in the following slides images and layers will be displayed as rows of neurons – as sections of 2D arrays.
- Only some connections between neurons are represented. Each such connection is associated to a weight. The biases are not represented, to avoid clutter.



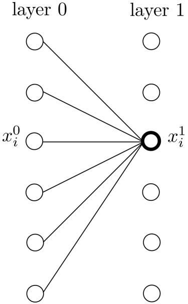


Layers representation

- For illustration purposes, in the following slides images and layers will be displayed as rows of neurons – as sections of 2D arrays.
- Only some connections between neurons are represented. Each such connection is associated to a weight. The biases are not represented, to avoid clutter.

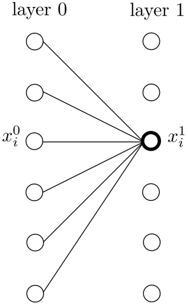


Towards convolutional layers

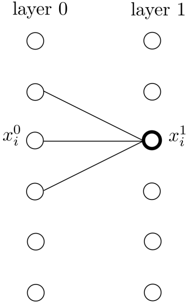


Fully connected layer:
$n \times n$ weights and n bias:
$n(n+1)$ parameters

Towards convolutional layers

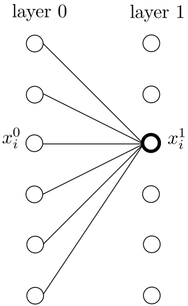


Fully connected layer:
$n \times n$ weights and n bias:
$n(n+1)$ parameters

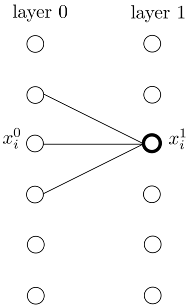


Locally conn. layer:
$n(s+1)$ parameters

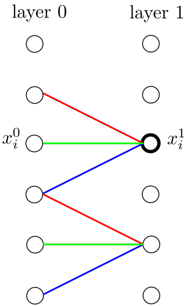
Towards convolutional layers



Fully connected layer:
$n \times n$ weights and n bias:
$n(n+1)$ parameters

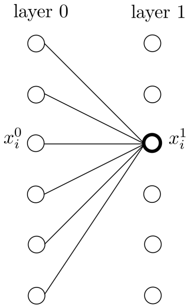


Locally conn. layer:
$n(s+1)$ parameters

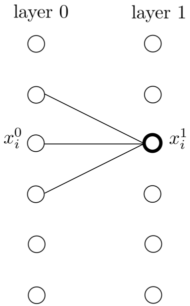


Weight replication: $s+1$ parameters.
Convolutional layer.

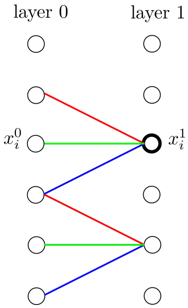
Towards convolutional layers



Fully connected layer:
$n \times n$ weights and n bias:
$n(n+1)$ parameters



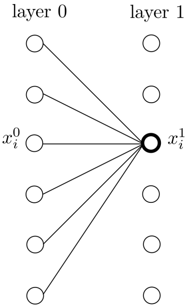
Locally conn. layer:
$n(s+1)$ parameters



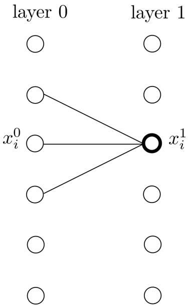
Weight replication: $s+1$ parameters.
Convolutional layer.

Here, s corresponds to number of incoming connections from the previous layer of a neuron. It is therefore equal to:

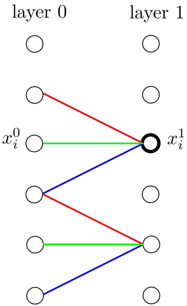
Towards convolutional layers



Fully connected layer:
$n \times n$ weights and n bias:
$n(n+1)$ parameters



Locally conn. layer:
$n(s+1)$ parameters



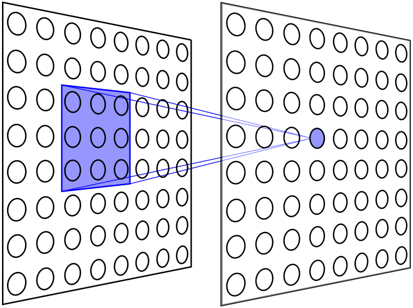
Weight replication: $s+1$ parameters.
Convolutional layer.

Here, s corresponds to number of incoming connections from the previous layer of a neuron. It is therefore equal to:

1/ 3
2/ 4
3/ 9
4/ 10

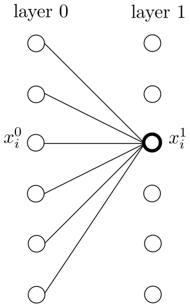
Convolutional layer illustration in 2D

- Illustration of a convolution of size 3×3

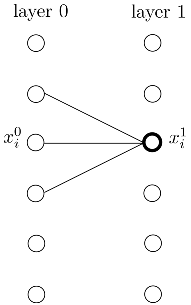


Convolutional layers: some figures

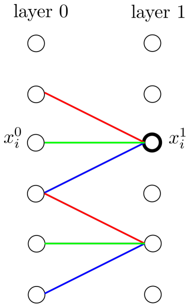
- 3×3 convolutions: $s = 9$
- Toy image: $n = 28 \times 28 = 784$
- Typical image: $n = 1000 \times 1000 = 10^6$



Fully connected layer:
$n(n+1)$ parameters
$\approx 6.10^5$
$\approx 10^{12}$



Locally conn. layer:
$n(s+1)$ parameters
7840
10^7



Weight replication: $s+1$
parameters.
10
10

Convolutional layers are fully connected layers

Convolutional layers are fully connected layers such that:

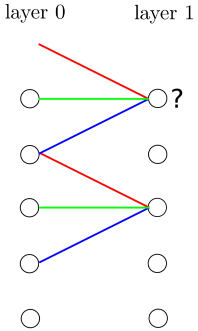
- the weights of all non-local connections are set to zero;
- the weights are shared among all the neurons of the same channel.

These limitations correspond to two inductive biases:

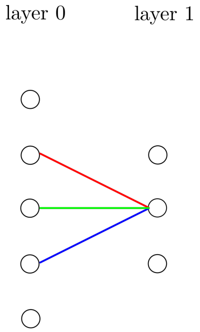
- local structure and
- translation invariance.

They make the network much more manageable (memory, optimization). Is the loss in generality important?

Dealing with borders

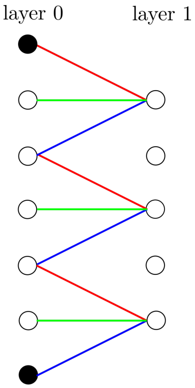


First solution: keep only well defined outputs



- Pros:
  - border effect disappears
- Cons:
  - Lack of flexibility
  - With deep networks, the field can become very small, or disappear

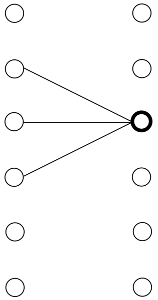
Second solution: zero padding



- Pros:
  - More flexible architecture
- Cons:
  - Border effect still present

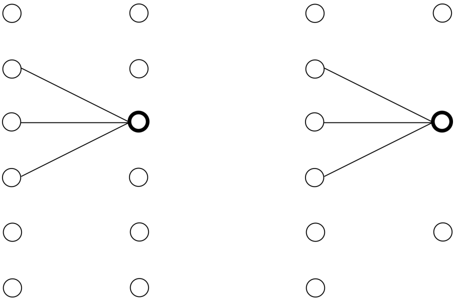
Stride

A convolutional layer can at the same time downsample the image by applying a sampling step, or *stride*.



Stride

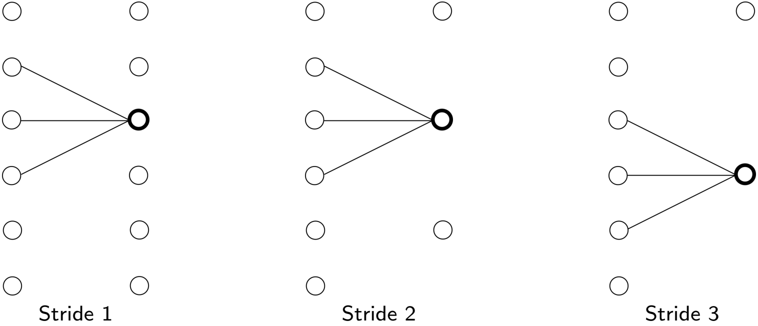
A convolutional layer can at the same time downsample the image by applying a sampling step, or *stride*.



Stride 1 Stride 2

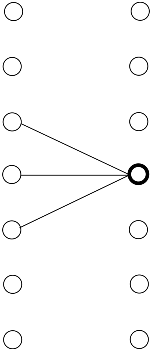
Stride

A convolutional layer can at the same time downsample the image by applying a sampling step, or *stride*.



Dilated convolutions

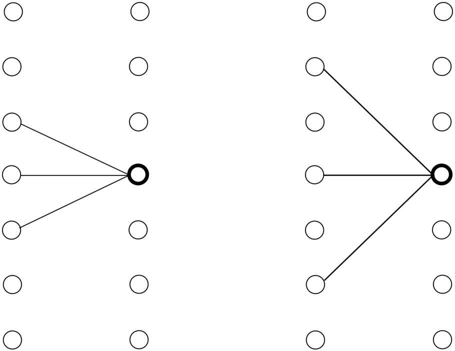
Dilated convolutions are used to increase the size of the *receptive field* of the network.



Normal convolution

Dilated convolutions

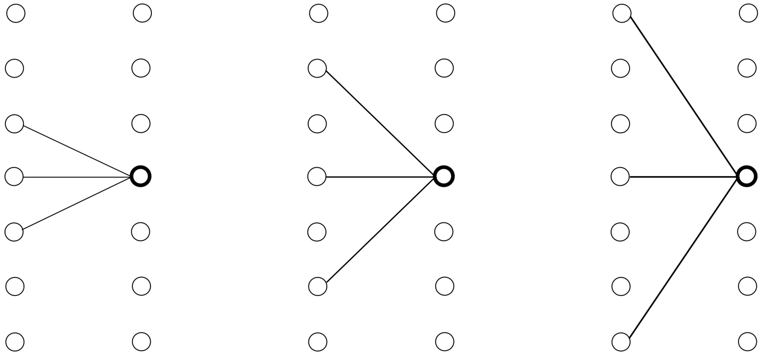
Dilated convolutions are used to increase the size of the *receptive field* of the network.



Normal convolution　　Dilation rate 2

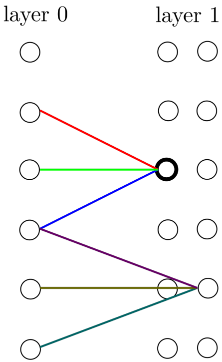
Dilated convolutions

Dilated convolutions are used to increase the size of the *receptive field* of the network.



Normal convolution   Dilation rate 2   Dilation rate 3

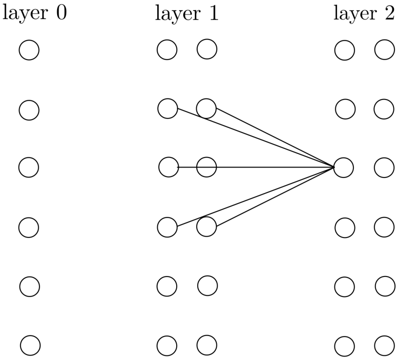
Several filters in the same convolutional layer



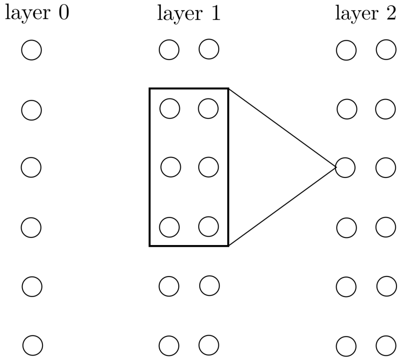
Note on vocabulary

The filters are also called channels or feature maps.

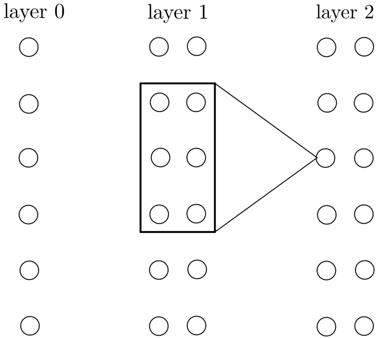
Several filters in the same convolutional layer



Several filters in the same convolutional layer



Consequences on the parameter number



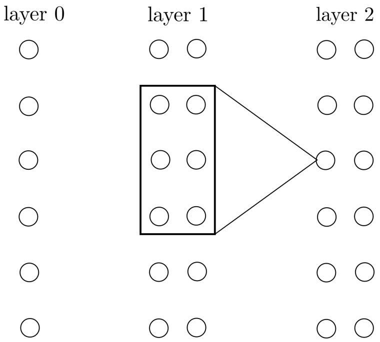
How many parameters do we have in layer 1?

1/ $c_1 \times (s+1)$

2/ $c_1 \times s$

3/ $(c_1+1) \times s$

Consequences on the parameter number



How many parameters do we have in layer 2?

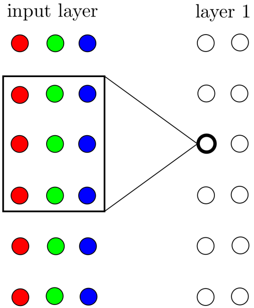
1/ $c_2 \times c_1 \times s$

2/ $c_2 \times (c_1 \times s + 1)$

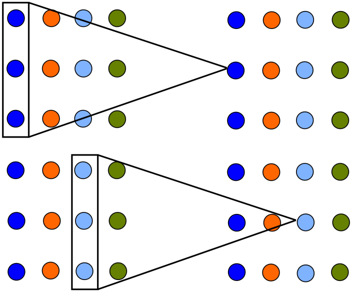
3/ $(c_2 + 1) \times (c_1 \times s + 1)$

Multi-channel images

An input image with p channels (for instance a colour image with 3 channels) can be represented by an input layer with p channels/filters.



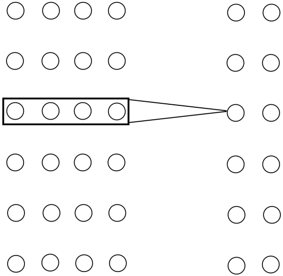
Depth-wise convolution



- The previous layer must contain the same number of filters
- The number of parameters is drastically reduced
- These layers are interesting when combined with 1×1 convolutions...

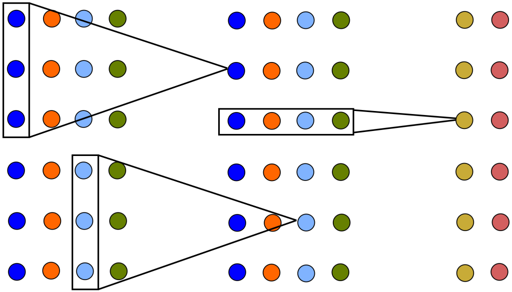
Dimension reduction

1×1 convolutions are used to reduce the number of filters - this is called by some authors *dimension reduction*.



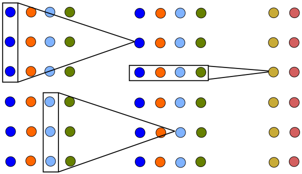
Decomposed convolution

The combination of depth-wise with 1×1 convolutions gives decomposed convolutions.



They are somehow a factorization of classical convolutions. Thus they allow reducing the number of parameters.

Decomposed convolution - number of parameters



The input layer holds c_1 channels, and the output layer c_2. The size of the receptive field is s.

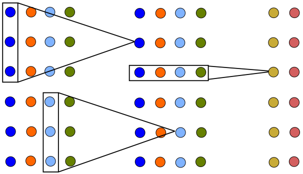
How many parameters does the corresponding decomposed convolution layer contain?

1. $c_1 \times (s + 1) + c_2 \times (c_1 + 1)$
2. $c_1 \times s + c_2 \times c_1$
3. $(c_1 + 1) \times s + (c_2 + 1) \times c_1$

How many parameters would the corresponding convolutional layer contain?

1. $c_1 \times s \times c_2$
2. $(c_1 \times s + 1) \times (c_2 + 1)$
3. $(c_1 \times s + 1) \times c_2$

Decomposed convolution - number of parameters



The input layer holds c_1 channels, and the output layer c_2. The size of the receptive field is s.

How many parameters does the corresponding decomposed convolution layer contain?

1. $c_1 \times (s+1) + c_2 \times (c_1 + 1)$
2. $c_1 \times s + c_2 \times c_1$
3. $(c_1 + 1) \times s + (c_2 + 1) \times c_1$

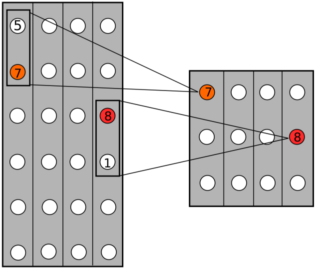
How many parameters would the corresponding convolutional layer contain?

1. $c_1 \times s \times c_2$
2. $(c_1 \times s + 1) \times (c_2 + 1)$
3. $(c_1 \times s + 1) \times c_2$

Contents

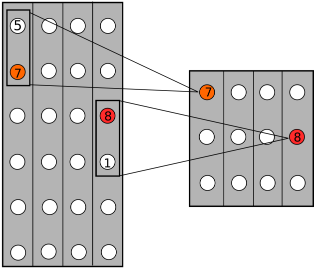
Max pooling

- Convolutional networks often contain subsampling steps. A common way of doing this today is by using *max pooling* layers with stride 2.
- Sampling is only applied along the spatial dimensions, not along the dimension of the filters.



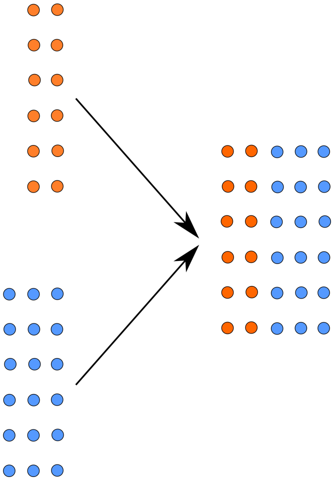
Max pooling

- Convolutional networks often contain subsampling steps. A common way of doing this today is by using *max pooling* layers with stride 2.
- Sampling is only applied along the spatial dimensions, not along the dimension of the filters.

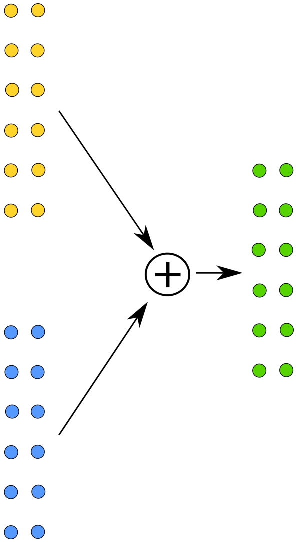


Note however a current trend that consists in using convolutional layers with a stride of 2

Branch merging: concatenation



Branch merging: addition



Flatten

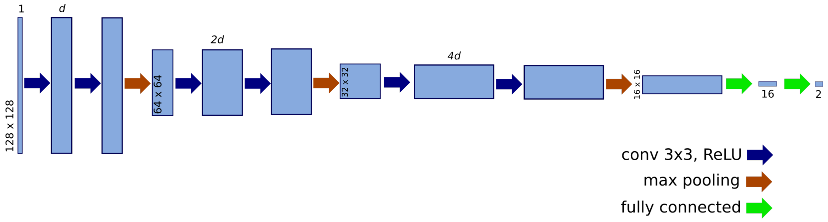
- Transforms an array into a vector
- Loss of local structure
- This is typically done to transition between a convolutional layer and a fully-connected one.

Main components of a convolutional neural network

Many successful architectures, especially for image classification, follow the same pattern:

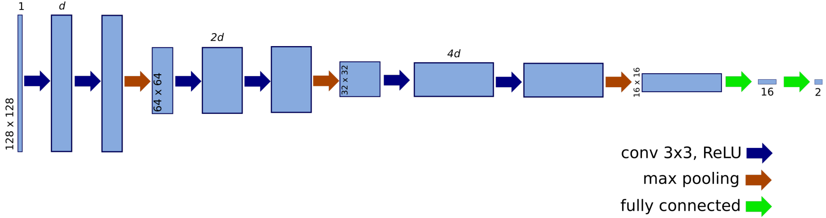
1. Several iterations of: One or several convolutional layers, with increasing depth, followed by max pooling
2. A few fully connected layers

2D representations



Credits: NN is work of Robin Alais et al.
Fundus image by Mikael Häggström, used with permission (CC0).

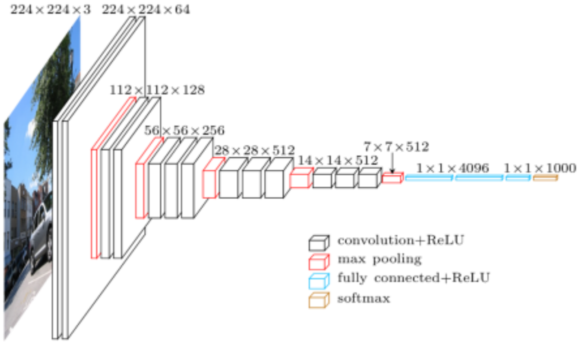
2D representations



This NN was used to estimate the position of the center of the macula on fundus images.

Credits: NN is work of Robin Alais et al.
Fundus image by Mikael Häggström, used with permission (CC0).

3D representations



Credits: VGG16 (From
https://www.cs.toronto.edu/ frossard/post/

Convolutional neural networks in deep learning

- They are pivotal to many of the successes achieved by neural networks these recent years
- They are interesting for dealing with regular structured data, such as images (or board games!)

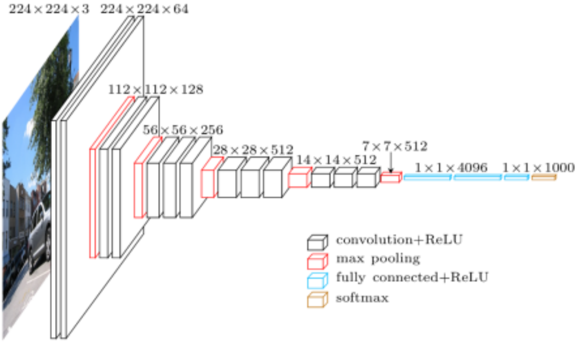
Acronyms

CNN and *ConvNet*

Contents

VGGnet (Visual Geometry Group Net)

- Proposed by K. Simonyan and A. Zisserman from the University of Oxford [Simonyan and Zisserman, 2014]
- Runner-up in the ImageNet Large Scale Visual Recognition Competition (ILSVRC) in 2014.
- Number of parameters (VGG16): 138 million.

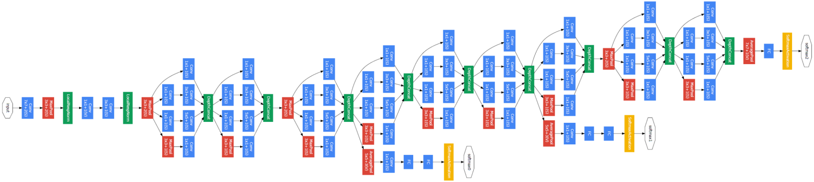


Credits: VGG16 (From https://www.cs.toronto.edu/ frossard/post/

GoogLeNet

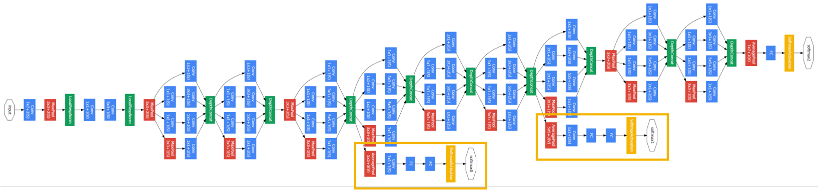
This is an architecture based on Inception v1 principles.

- Winner of the ImageNet Large Scale Visual Recognition Competition (ILSVRC) in 2014.
- Number of parameters: *only* 5 million.



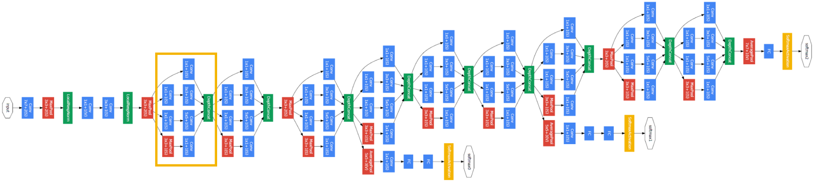
Credits: From [Szegedy et al., 2014]

GoogLeNet review



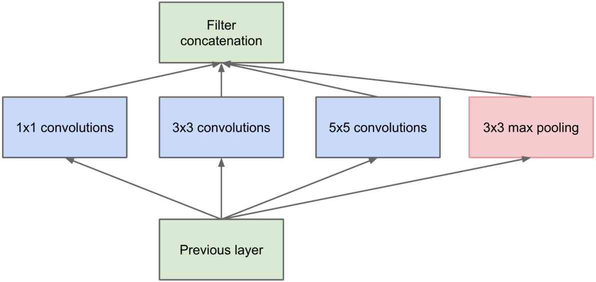
- Two extra outputs are added
- They are added to the final output with a 0.3 weight
- They help propagate gradient through the low levels of the network

GoogLeNet review



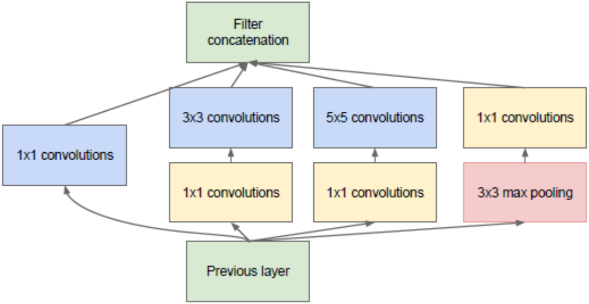
- 9 inception modules

Inception module: "naive version"



Credits: From [Szegedy et al., 2014]

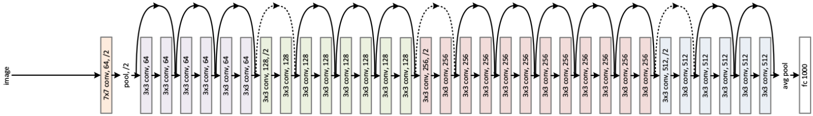
Inception module



- 1×1 convolutions are used to keep the number of parameters low.

Credits: From [Szegedy et al., 2014]

ResNet

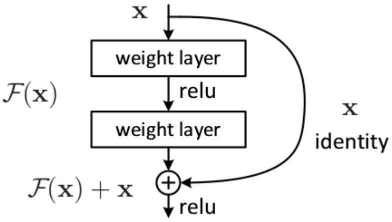


Residual network with 34 layers

- Winner of the ImageNet Large Scale Visual Recognition Competition (ILSVRC) in 2015.
- The authors tested up to 1202 layers. They reported no training difficulties, but overfitting [He et al., 2015]

ResNet module

- Skip connections help backpropagation

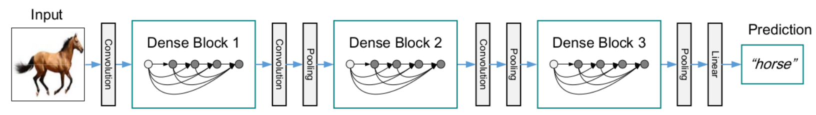


Residual learning block

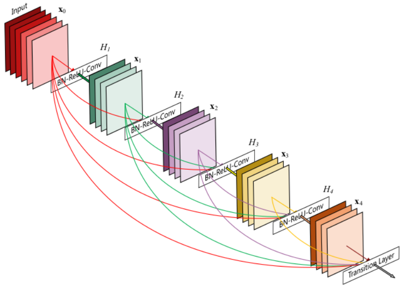
Credits: From [He et al., 2015]

DenseNet[Huang et al., 2018]

Architecture

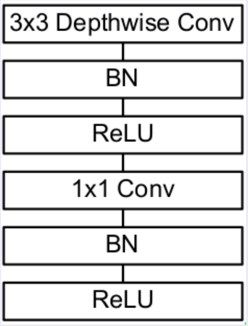


Dense block



MobileNet [Howard et al., 2017]

Depth-wise separable convolution



Number of parameters: 4 million.

Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Contents

A revolution in image analysis

- Deep learning has brought an undeniable break-through in image analysis (as in other fields)
- A significant part of research efforts in image analysis today is based on deep learning
- Its applications are ubiquitous
- Not only we can improve on existing tasks, but we can also treat some problems in a completely different way (for example, image generation).

Limitations

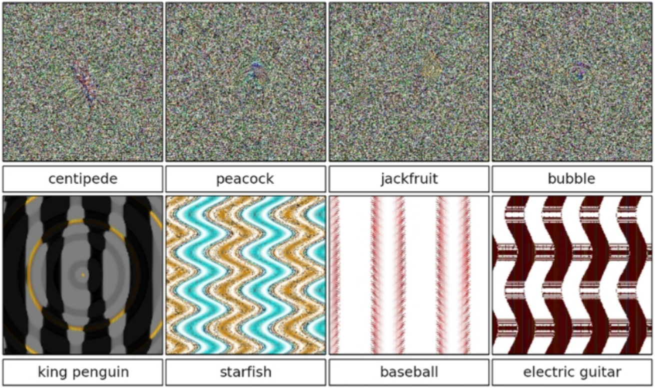
For a deep-learning solution to work, you need:

- Enough annotated data
- A lot of fiddling (different architectures; hyper-parameters; optimization)
- Expensive, energy hungry, computing resources

Moreover, these models lack interpretability.

ConvNets can be fooled

Deep learning can produce astonishing results [Nguyen et al., 2015]...



Some deep learning libraries

Deep learning is a very competitive domain, where code sharing is very common. Main libraries:

- Tensorflow (with its keras interface), by Google (Apache licence)
- PyTorch (Facebook - BSD licence)

References I

[Cireşan et al., 2012] Cireşan, D., Giusti, A., Gambardella, L. M., and Schmidhuber, J. (2012). Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 2843–2851. Curran Associates, Inc.

[Cireşan et al., 2011] Cireşan, D., Meier, U., Masci, J., and Schmidhuber, J. (2011). A committee of neural networks for traffic sign classification. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1918–1921. IEEE.

[Dosovitskiy et al., 2021] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *arXiv:2010.11929 [cs]*. arXiv: 2010.11929.

[Everingham et al., 2014] Everingham, M., Eslami, S. M. A., Gool, L. V., Williams, C. K. I., Winn, J., and Zisserman, A. (2014). The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision*, 111(1):98–136.

[Everingham et al., 2010] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338.

References II

[Glasner et al., 2009] Glasner, D., Bagon, S., and Irani, M. (2009). Super-resolution from a single image. In *2009 IEEE 12th International Conference on Computer Vision*, pages 349–356. ISSN: 2380-7504.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*. arXiv: 1512.03385.

[Howard et al., 2017] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[Huang et al., 2018] Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2018). Densely Connected Convolutional Networks. *arXiv:1608.06993 [cs]*. arXiv: 1608.06993.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

[Lecun et al., 1998] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

References III

[Nguyen et al., 2015] Nguyen, A., Yosinski, J., and Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 427–436.

[Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252.

[Simard et al., 1993] Simard, P., LeCun, Y., and Denker, J. S. (1993). Efficient pattern recognition using a new transformation distance. In *Advances in neural information processing systems*, pages 50–58.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[Stallkamp et al., 2011] Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. (2011). The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *The 2011 International Joint Conference on Neural Networks*, pages 1453–1460. ISSN: 2161-4407.

References IV

[Szegedy et al., 2014] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014). Going Deeper with Convolutions. *arXiv:1409.4842 [cs]*. arXiv: 1409.4842.

[Zhang et al., 2011] Zhang, X., Thibault, G., and Decencière, E. (2011). Application of the Morphological Ultimate Opening to the Detection of Microaneurysms on Eye Fundus Images from Clinical Databases. In *ICS'13*.

[Zhang et al., 2014] Zhang, X., Thibault, G., Decencière, E., Marcotegui, B., Laÿ, B., Danno, R., Cazuguel, G., Quellec, G., Lamard, M., Massin, P., Chabouis, A., Victor, Z., and Erginay, A. (2014). Exudate detection in color retinal images for mass screening of diabetic retinopathy. *Medical Image Analysis*, 18(7):1026–1043.