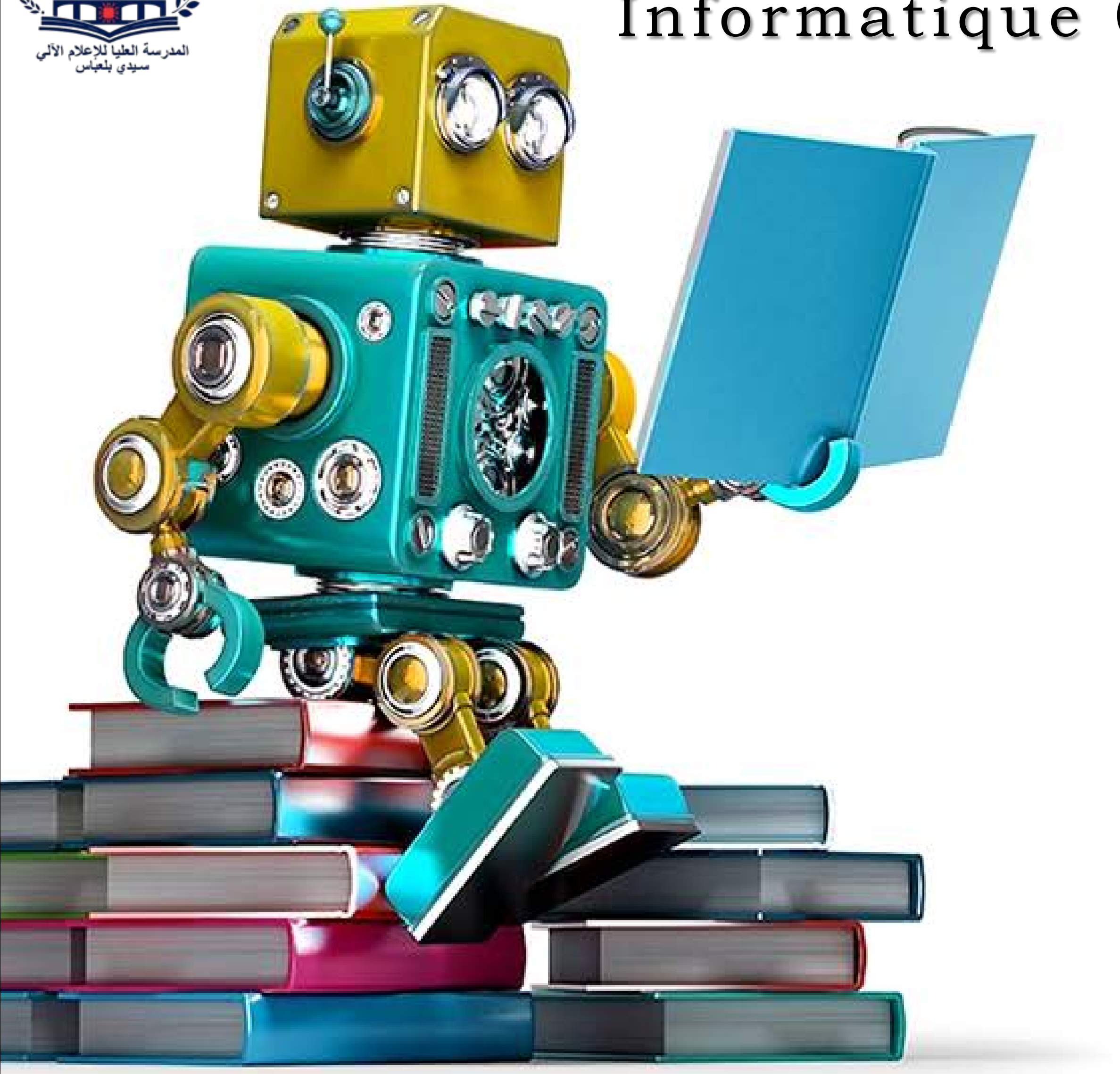
ESI-SBA - École Supérieure en Informatique 08-MAI-1945



DEEP LEARNING

DR N. DIF

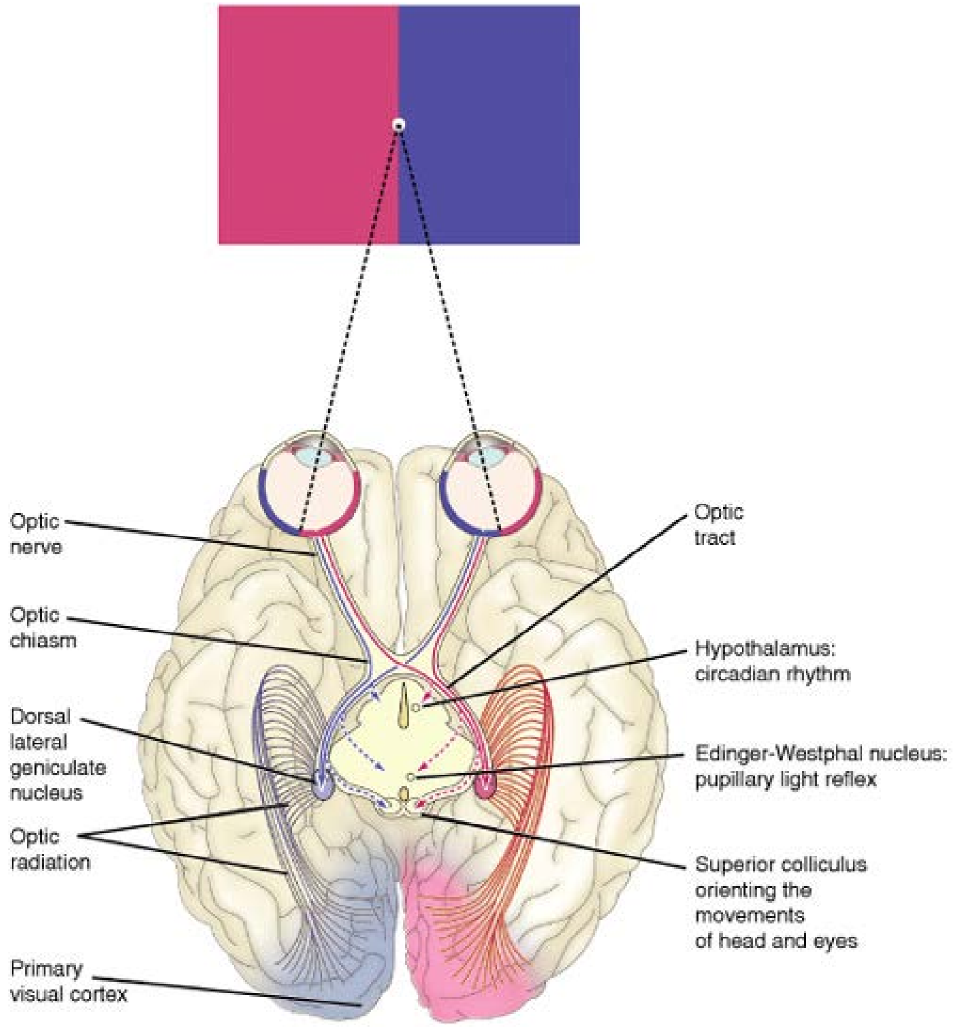
PROGRAM

Deep learning

1. **Foundations for DL**
2. **Optimizing Deep Neural Networks**
3. Recurrent Neural Networks
4. Convolution Neural Networks
5. Transfert Learning
6. Dimensionality Reduction with Deep Learning

1. HISTORY AND INSPIRATION

The primary visual pathway carries information from the eye to the regions of the brain that determine what we see.


Optic nerve
Optic chiasm
Dorsal lateral geniculate nucleus
Optic radiation
Primary visual cortex
Optic tract
Hypothalamus: circadian rhythm
Edinger-Westphal nucleus: pupillary light reflex
Superior colliculus orienting the movements of head and eyes

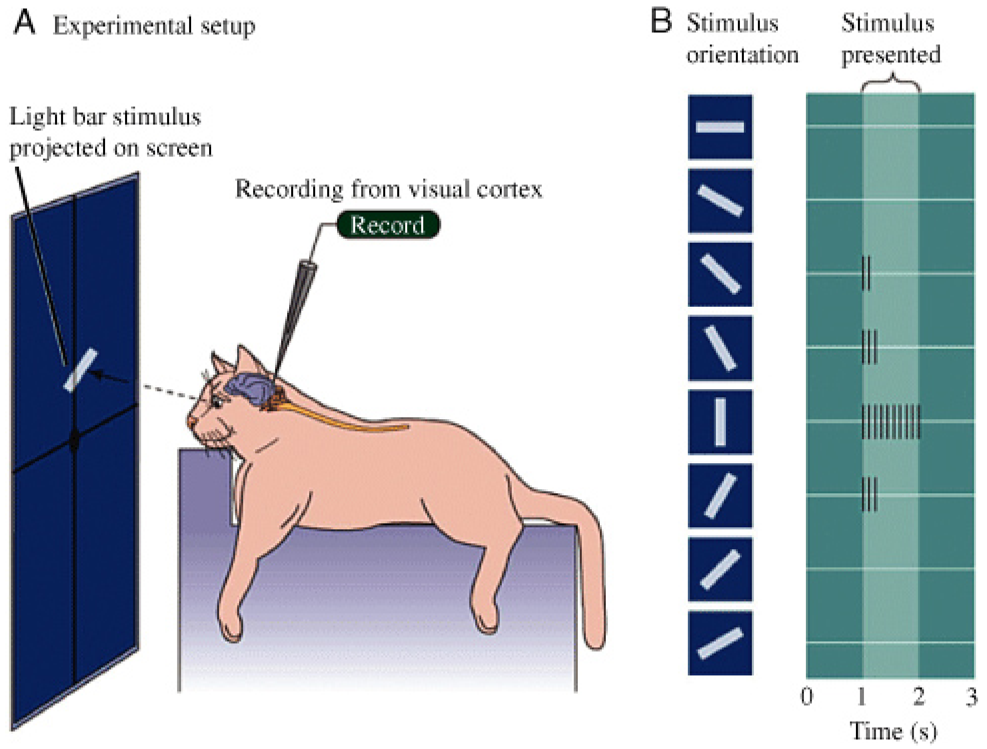
1. HISTORY AND INSPIRATION

1.1. Hubel and Wiesel experiment

David Hubel and Torsten Wiesel performed experiments on the visual system of cats.

They inserted microelectrodes into the brains of anesthetized cats and recording the activity of individual neurons in the visual cortex.

They projected a light onto a screen in front of a cat's eye and observed the response of neurons in the visual cortex. They found that some neurons ‘**simple cells**” responded only to **a specific orientation**, such as a horizontal or vertical line. Other neurons “**complex cells**” responded to **movement in a particular direction**.



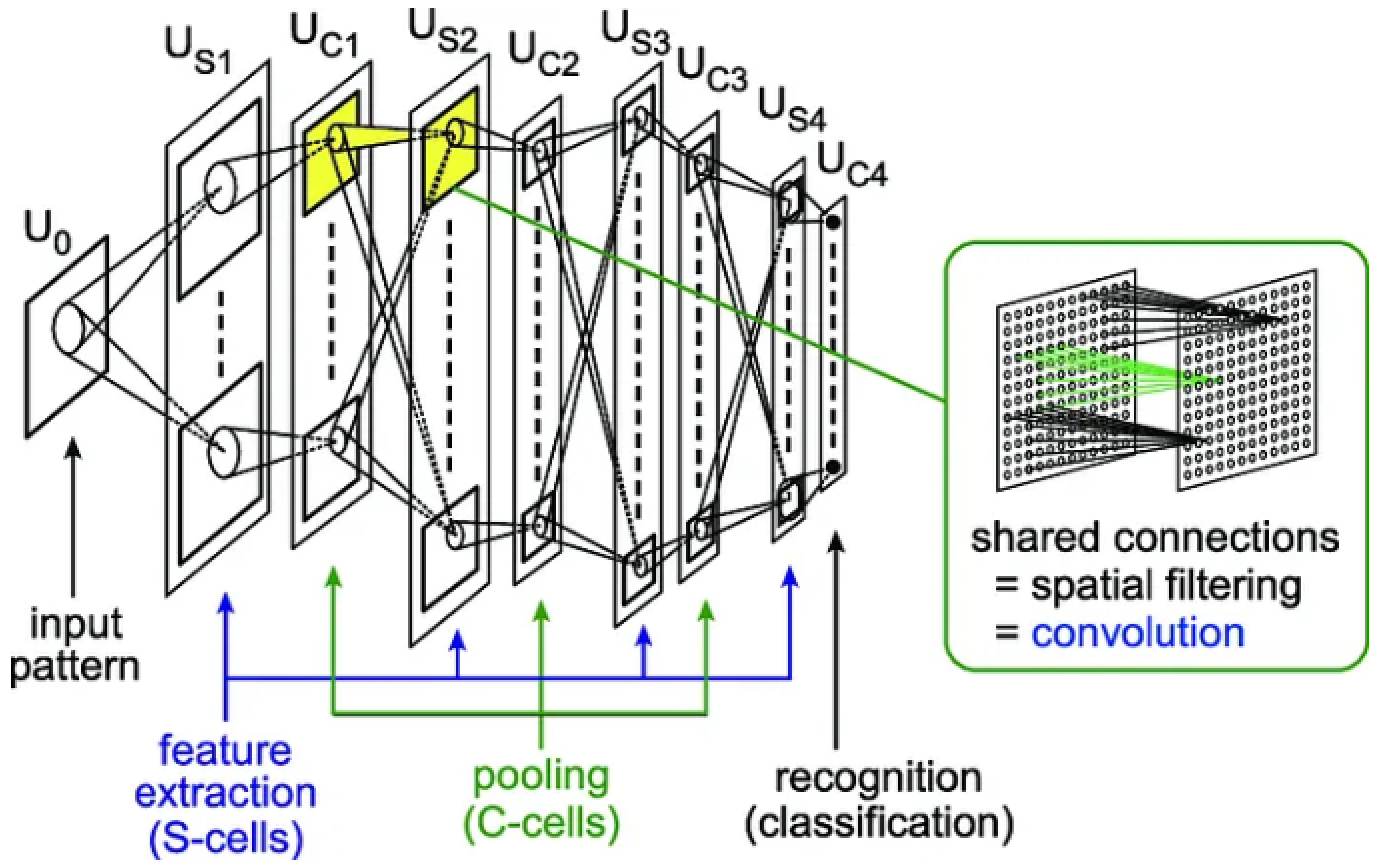
1. HISTORY AND INSPIRATION

1.2. NeoCognitron

In1980s, Dr. Kunihiko Fukushima was inspired by Hubel and Wiesel's work on simple and complex cells, and proposed the "neocognitron" model

It includes components termed "S-cells" and "C-cells." that presents mathematical operations.

The overall idea is to capture the "**simple-to-complex**" concept and turn it into a computational model for visual pattern recognition.

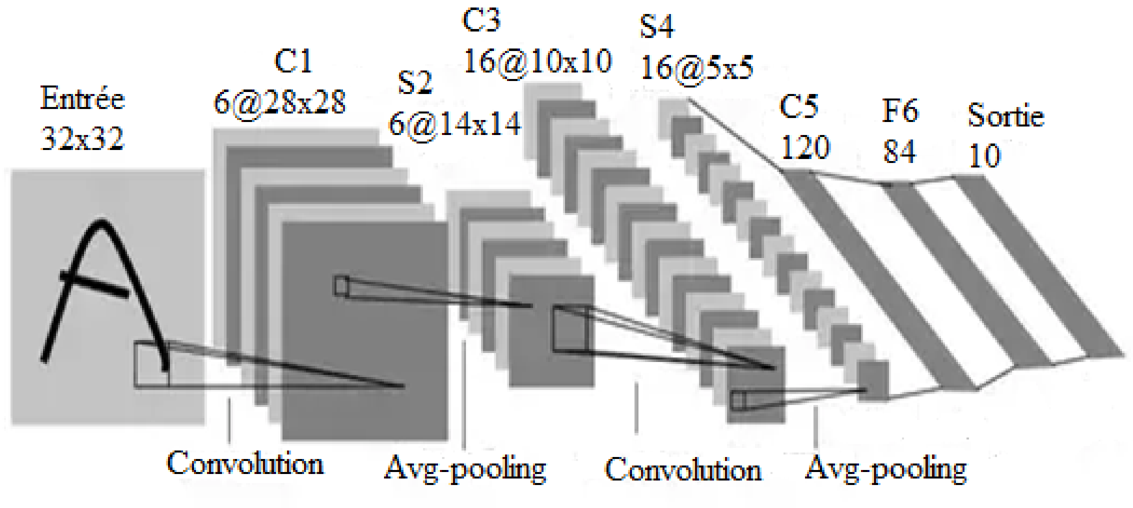


U_0
U_{S1}
U_{C1}
U_{S2}
U_{C2}
U_{S3}
U_{C3}
U_{S4}
U_{C4}
input pattern
feature extraction (S-cells)
pooling (C-cells)
recognition (classification)
shared connections = spatial filtering = convolution

1. HISTORY AND INSPIRATION

1.3. LeNet

Inspired from *NeoCognitron*, Yann LeCun et al proposed the a CNN architecture for handwritten character recognition. He combined convolutional neural networks with back propagation.



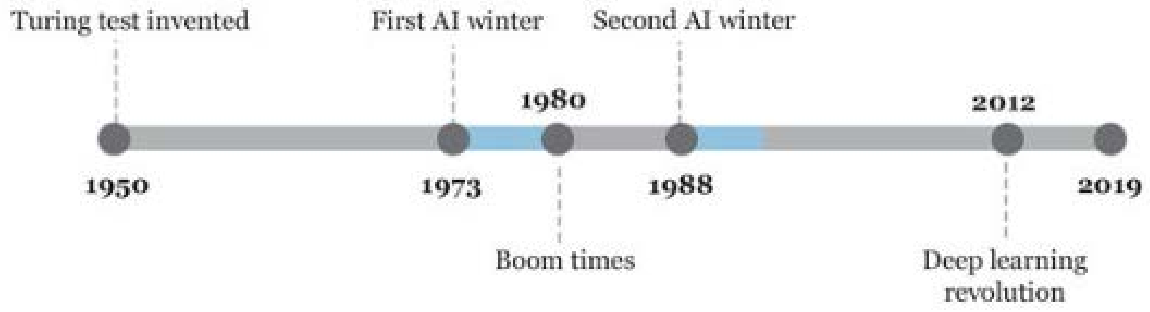
1. HISTORY AND INSPIRATION

1.4. AI winter

It was a time when funding and interest in artificial intelligence research were low.

Lack of work on deep learning due to :

- Deep neural networks were not widely available.
- Deep learning models typically require large amounts of data and extensive computational power to train, which can be expensive and time-consuming
- The algorithms and techniques used for deep learning were not as well developed during this period.



1. HISTORY AND INSPIRATION

1.5. ImageNet Competition

In 2012, a team of researchers from the University of Toronto, including **Alex** Krizhevsky et al, achieved a breakthrough in image recognition using deep learning techniques.

They developed a deep neural network architecture called a convolutional neural network (CNN) that significantly outperformed previous approaches on the ImageNet Large Scale Visual Recognition Challenge.

The CNN approach was able to achieve a top-5 error rate of 15.3%, compared to the previous best rate of 26.2%, and marked a major improvement in the state-of-the-art for computer vision.

Since then, deep learning has become a key component of artificial intelligence and has been applied to a wide range of applications, including natural language processing, speech recognition, and robotics.

2. FOUNDATIONS OF CNN

2.1. The Basic Architecture

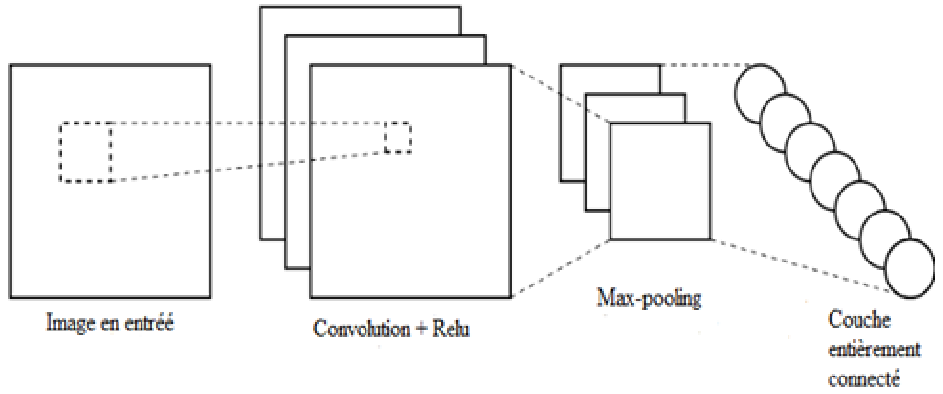
The architecture of a CNN typically consists of several layers, each with a specific purpose.

The basic layers of a CNN architecture include:

1. Convolutional Layer.
2. Pooling Layer.
3. Fully Connected Layer: This layer is responsible for making the final predictions.

In addition to these basic layers, a CNN may also include:

1. Dropout Layer.
2. Batch Normalization Layer:
3. Softmax Layer.



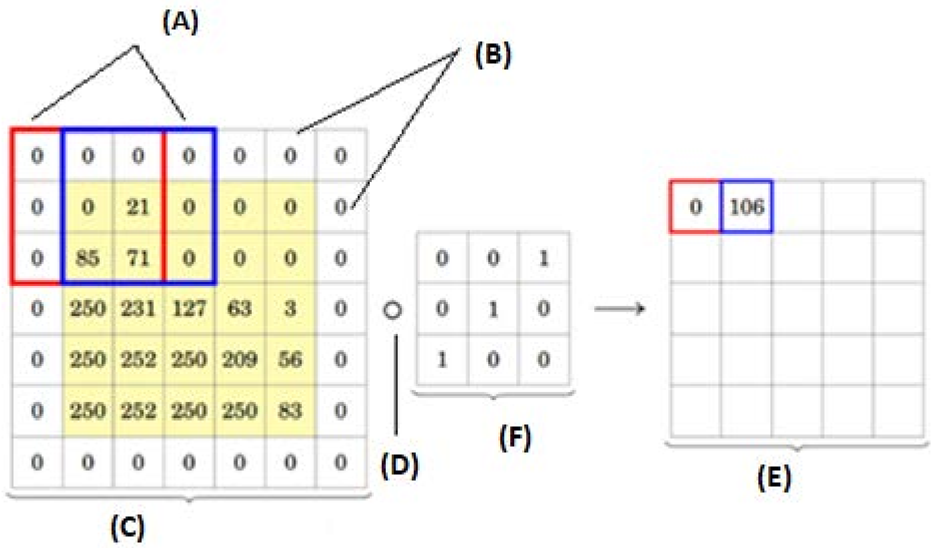
2. FOUNDATIONS OF CNN

2.1. The Basic Architecture : Convolution

The convolutional layer is responsible for extracting features from the input image. It uses a set of learnable **filters** (**parameters**) to **convolve** over the input image and produce a set of **feature maps**

$$N^{(t+1)} = \frac{N^{(t)} - F + 2P}{S} + 1$$

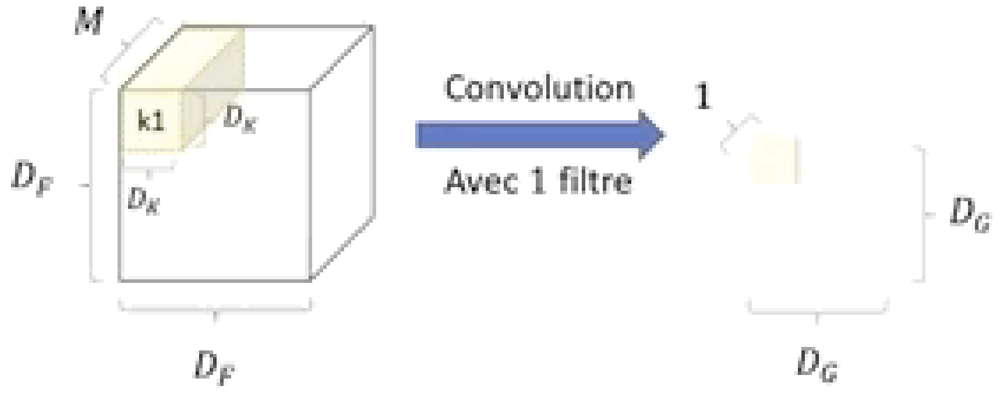
$N^{(t+1)}$: The size of the new feature map.
$N^{(t)}$: The size of the previous feature map.
F: Filter's size.
P : Padding.
S : Stride



(A) Receptive fields.
(B) Padding.
(C) Input data.
(D) Dot product.
(E) Feature Map.
(F) Kernel.

2. FOUNDATIONS OF CNN

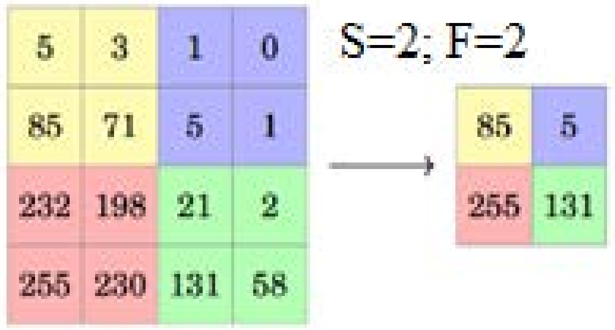
2.1. The Basic Architecture : Convolution



2. FOUNDATIONS OF CNN

2.1. The Basic Architecture : Pooling

The pooling layer reduces the **spatial size** of the feature maps, while also retaining the most **important features.** This helps to reduce the **computational complexity** of the model.



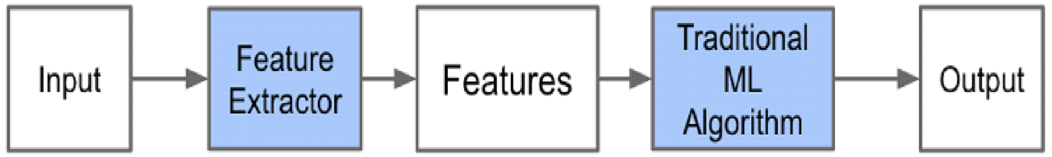
There are several variations of pooling layers, including:

1. **Max pooling**: takes the maximum value of each region.
2. **Average pooling**: takes the average value of each region.
3. **Global max pooling**: takes the maximum value over the entire feature map.
4. **Global average pooling**: takes the average value over the entire feature map.
5. **L2 pooling**: takes the L2 norm of each region.
6. **Fractional max pooling**: takes the maximum value over a randomly chosen subset of the region.

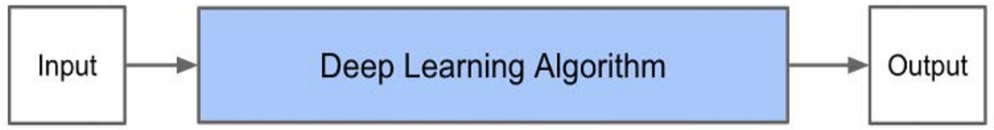
2. FOUNDATIONS OF CNN

2.2. CNN vs Traditional ML Algorithms

1. **Data Requirements**: CNNs typically require a large amount of labeled , while traditional ML algorithms can work well with smaller amounts data.
2. **Feature Extraction**: CNNs automatically extract features from raw data, such as images, while traditional ML algorithms require manual feature extraction by domain experts or automatic **unsupervised** methods.
3. **Training**: CNNs require more computational resources and time to train compared to traditional ML algorithms.
4. **Generalization**: CNNs can generalize well to new and unseen data, while traditional ML algorithms may struggle with generalization if they have not been trained on a diverse range of examples.



Traditional Machine Learning Flow



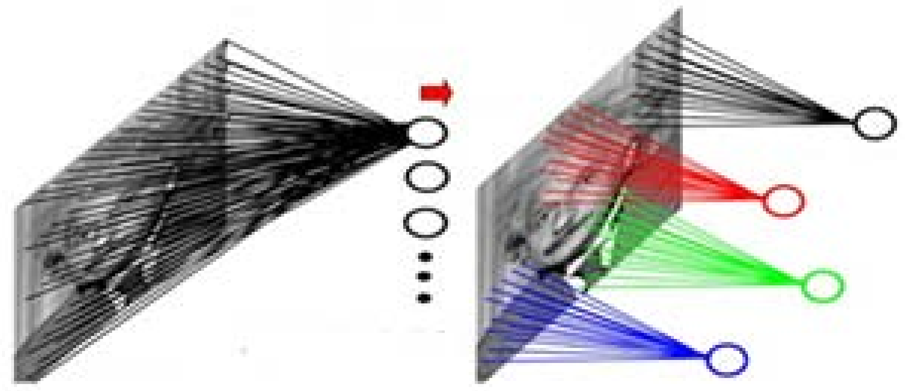
Deep Learning Flow

2. FOUNDATIONS OF CNN

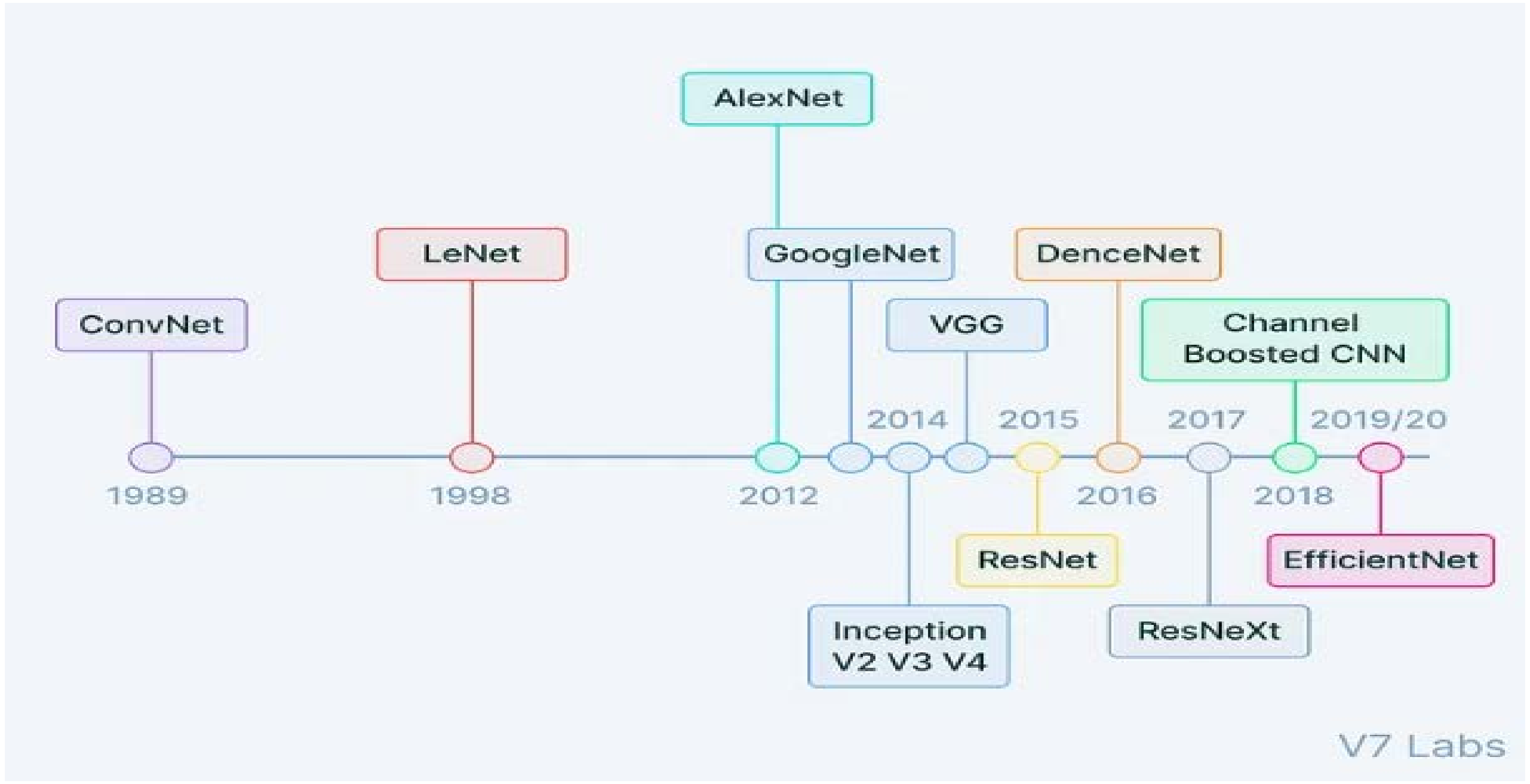
2.3. CNN vs Traditional DL Algorithms

CNNs are specifically designed to work with spatially structured data, such as images or videos due to convolutional layers. In contrast, traditional deep learning algorithms typically consist of fully connected layers, which can be applied to a wider variety of data types, including text and numerical data.

In a CNN, each neuron in a layer is only connected to a small region of the previous layer. This allows the network to **learn local features** in the data, such as edges or textures, Additionally, this approach **reduces the number of parameters**.



3. DEEP CONVOLUTIONS ARCHITECTURES



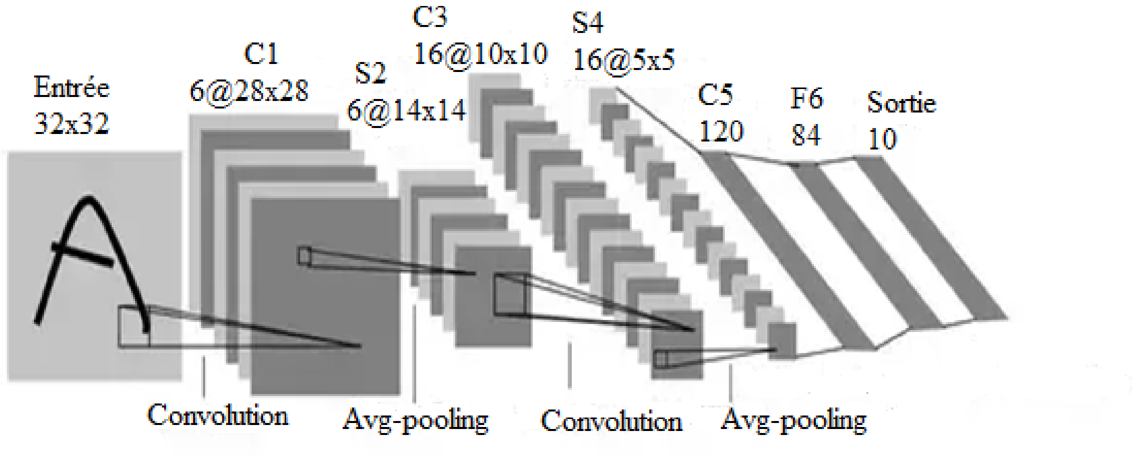
Source: v7labs.com

3. DEEP CONVOLUTIONS ARCHITECTURES

3.1. LeNet

The LeNet architecture is composed of seven layers:
Input layer: takes an image of size 32x32 pixels.
LeNet is composed of 2 convolutional layers (C1 and C3) and 2 subsampling layers (S2 and S4) and 3 fully connected layers.
The size of feature maps at each layer is presented as Depth@Width x Height.

The size of filters is fixed to : 5 x 5.



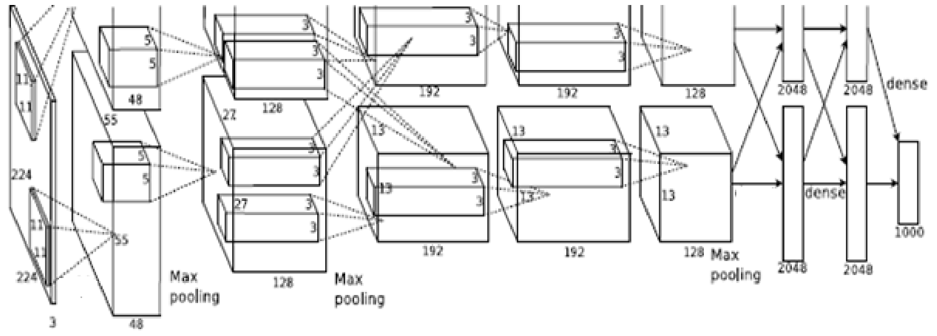
3. DEEP CONVOLUTIONS ARCHITECTURES

3.2. AlexNet

The AlexNet architecture consists of 5 convolutional layers, followed by 3 fully connected layers, and an output layer. The first two convolutional layers are followed by max-pooling layers, while the remaining three convolutional layers are followed by normalization layers.

- C1 : 96 filters of size 11 x 11 with a stride of 4.
- P1 : 3x3 receptive fields with a stride of 2.
- C2 : 256 filters of size 5x5 with a stride of 1.
- P2 : 3x3 receptive fields with a stride of 2.
- C3, C4, and C5 : 384, 384, and 256 filters of size 3x3 with a stride of 1.

The AlexNet architecture uses dropoutand batch normalization to prevent overfitting.



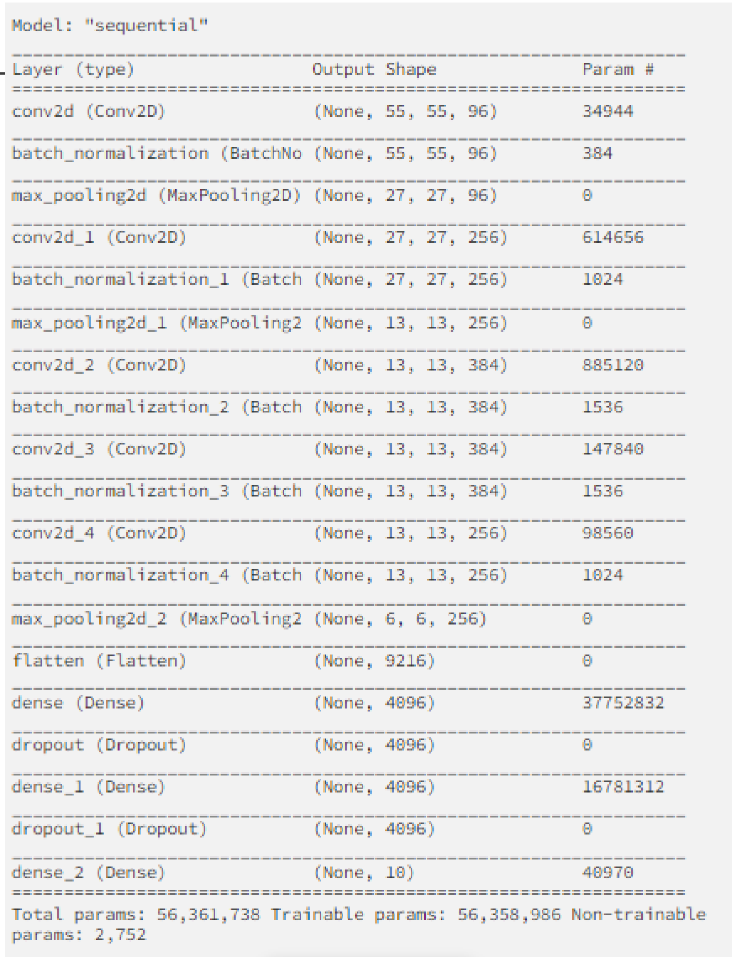
The AlexNet model contains over 56,361,738 Trainable parameters.

3. DEEP CONVOLUTIONS ARCHITECTURES

3.2. AlexNet : How to compute the number of parameters?

Consider an input of size W × H × C going to a convolutional layer L with square kernel size K and M output maps (channels). Then L has:

- K^2CM weights and M bias. K^2C per filter (the size of each "piece" of the input run through the filter), and M filters in total (one per output channel).



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 55, 55, 96)        34944
_________________________________________________________________
batch_normalization (BatchNo (None, 55, 55, 96)        384
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 27, 27, 96)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 27, 27, 256)       614656
_________________________________________________________________
batch_normalization_1 (Batch (None, 27, 27, 256)       1024
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 256)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 13, 13, 384)       885120
_________________________________________________________________
batch_normalization_2 (Batch (None, 13, 13, 384)       1536
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 13, 13, 384)       147840
_________________________________________________________________
batch_normalization_3 (Batch (None, 13, 13, 384)       1536
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 13, 13, 256)       98560
_________________________________________________________________
batch_normalization_4 (Batch (None, 13, 13, 256)       1024
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 9216)              0
_________________________________________________________________
dense (Dense)                (None, 4096)              37752832
_________________________________________________________________
dropout (Dropout)            (None, 4096)              0
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              16781312
_________________________________________________________________
dropout_1 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                40970
=================================================================
Total params: 56,361,738 Trainable params: 56,358,986 Non-trainable
params: 2,752
```

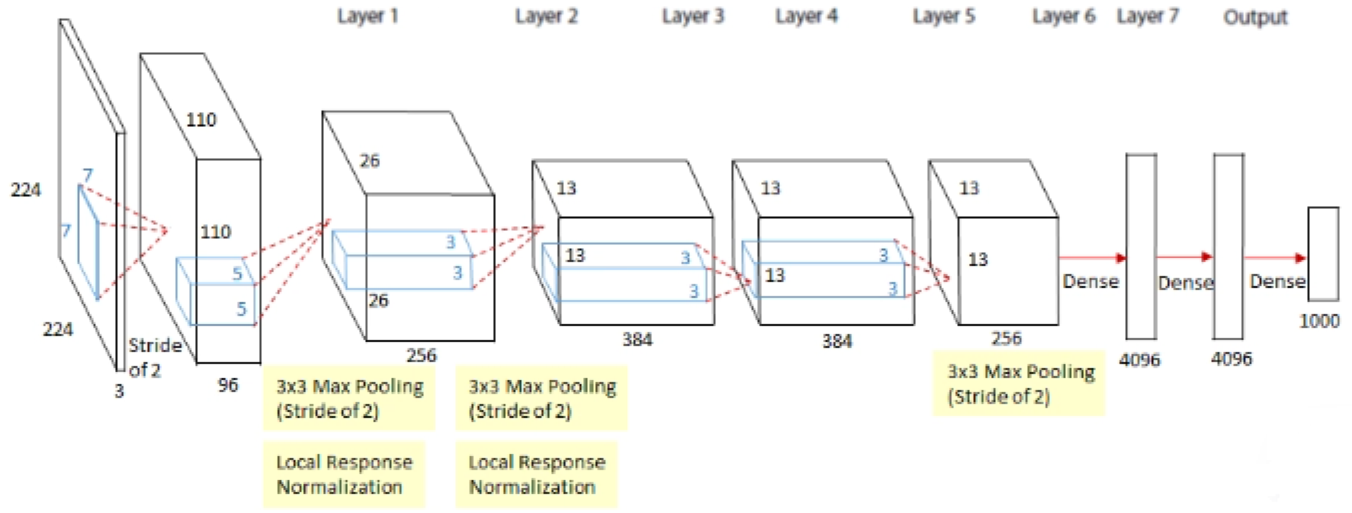
3. DEEP CONVOLUTIONS ARCHITECTURES

3.3. ZFNET : How to choose fiter's size? .

The ZFNet architecture has five convolutional layers, followed by three fully connected layers. Include different filters sizes : 7x7, 5x5, and 3x3.

Proposed to reduce the first filter's size in the AlexNet architecture from 11x11 to 7x7, and to use small stride size (from 4 to 2) to **capture more fine-grained details in the image**.

Use of "**deconvolutional**" layers to visualize (**decrypt**) the content of **feature maps**.

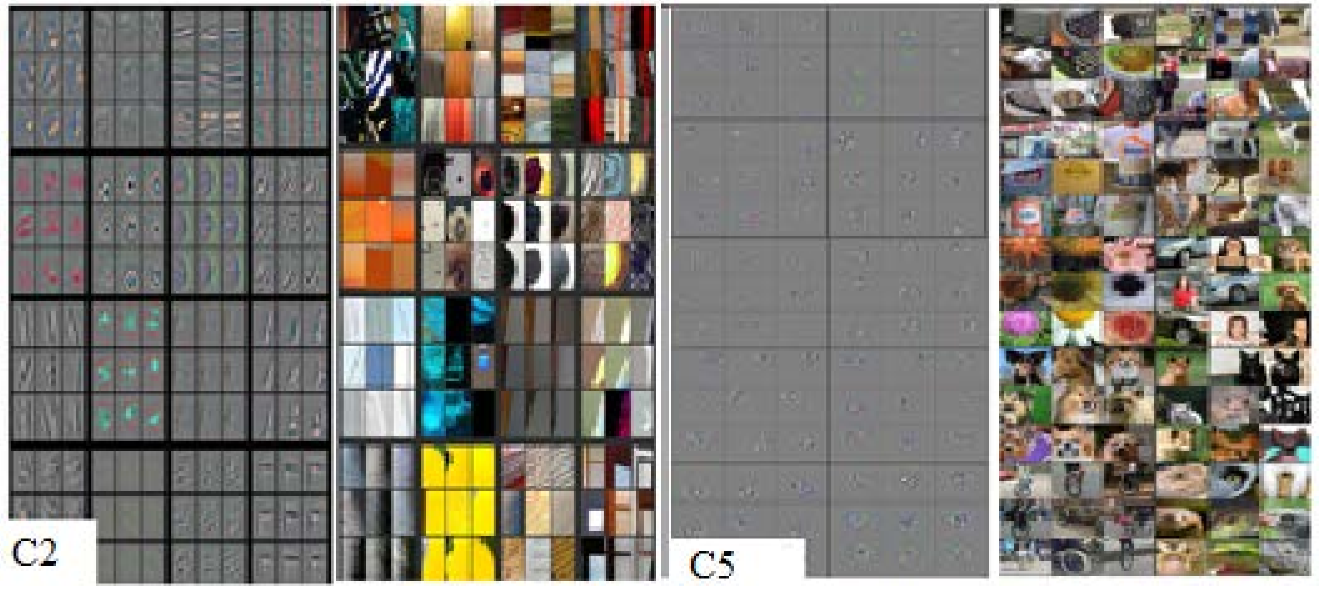


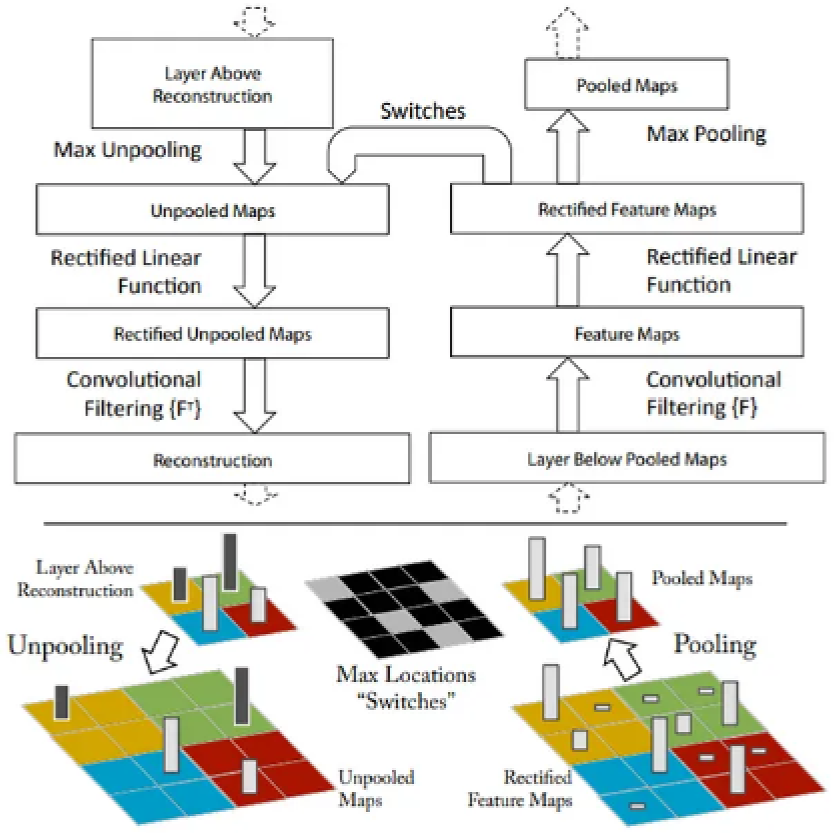
3. DEEP CONVOLUTIONS ARCHITECTURES

3.3. ZFNET : How to choose fiter's size? .

Deconvnet does the reverse of a normal convolutional network, it uses unpooling and filters to recover the pixels from the features.

The deconvolutional neural network is attached to all the layers of the CNN we want to visualize.





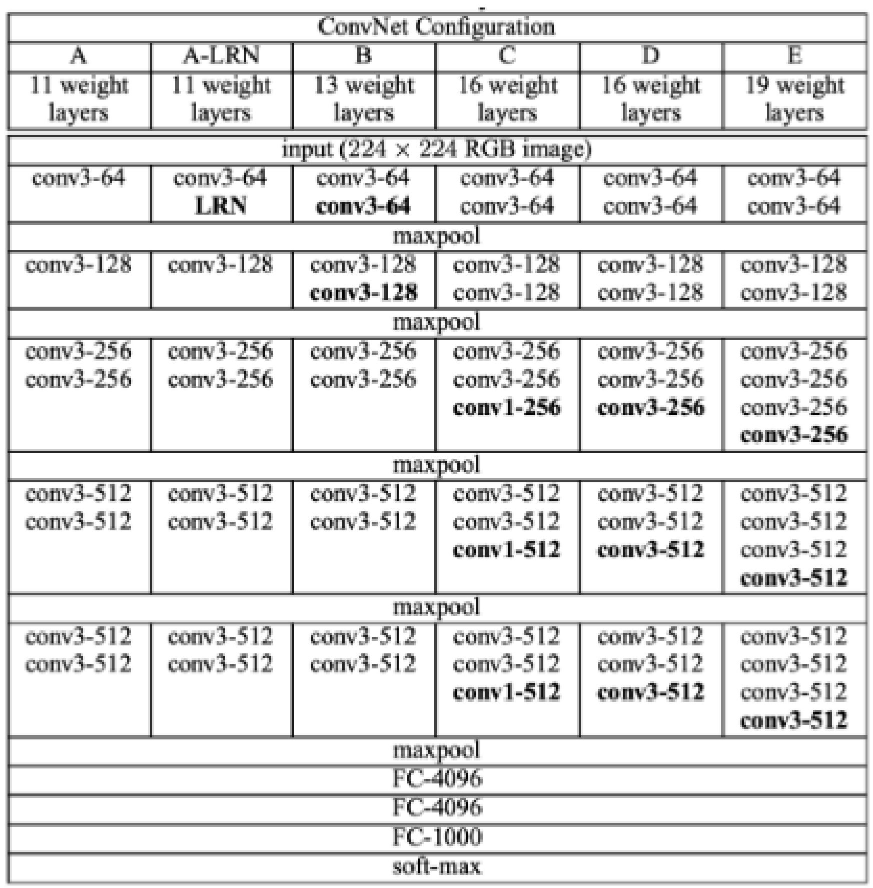
3. DEEP CONVOLUTIONS ARCHITECTURES

3.4. VGGNet

VGGNet is a convolution neural network (CNN) model.
It uses a tiny 3×3 receptive field with a 1-pixel stride for comparison, AlexNet used an 11×11 receptive field with a 4-pixel stride.

Motivation : two stacked 3 x 3 convolution filter (without spatial pooling in between) have less parameters than one 5*5 convolution filter, and, two stacked 3 x 3 convolution filter have less parameters than 7 x 7 convolution filter.

The benefit : stacking multiple convolutional layers helps for adding more non linear activation functions, which improves the decision function and introduces more discriminative power to the model. two 3*3 convolution filter will make network more deep and extract more complex features than one 5*5 convolution filter.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

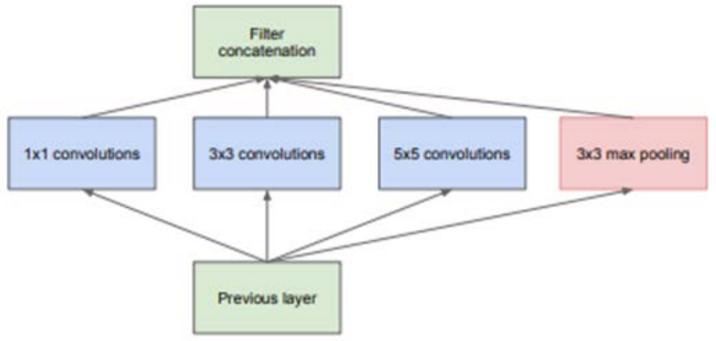
3. DEEP CONVOLUTIONS ARCHITECTURES

3.5. Inception

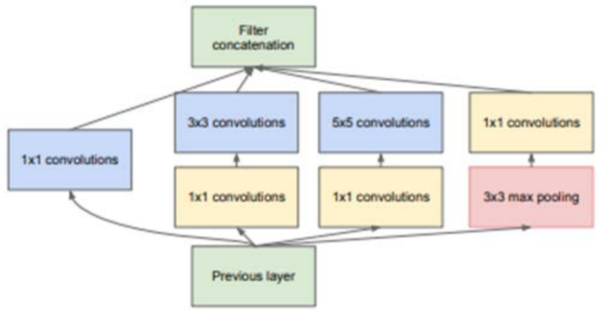
Inception is a CNN composed of 22 layers.
Inception propose to use of **inception modules** that allowed the network to capture multi-scale features at different levels of abstraction.

An inception module consists of a set of parallel convolutional layers with different filter sizes (1x1, 3x3, and 5x5) and a pooling layer, which are then concatenated to form the module output.

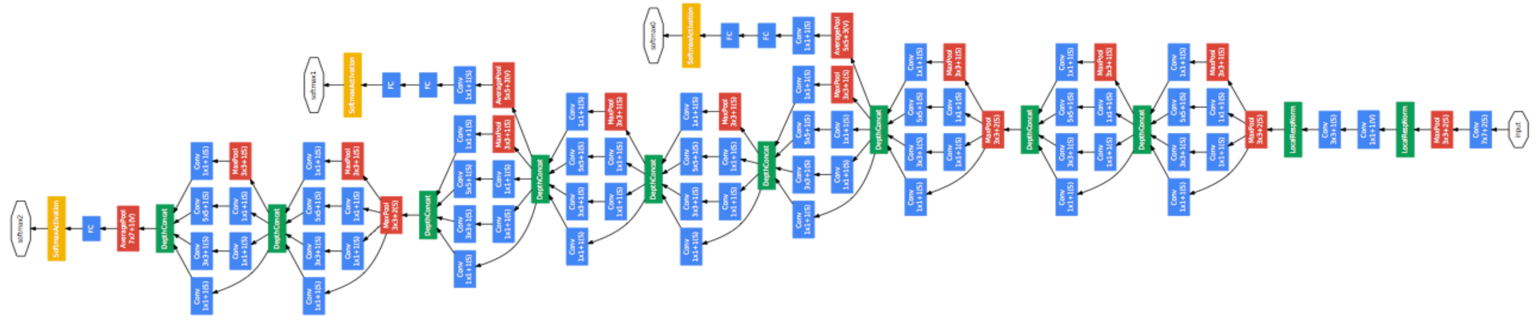
To reduce the number of parameters of the resulting feature maps a "bottleneck" layers are used by adding an extra 1x1 convolution before the 3x3 ad 5x5 layers.



(a) Inception module, naïve version



(b) Inception module with dimension reductions

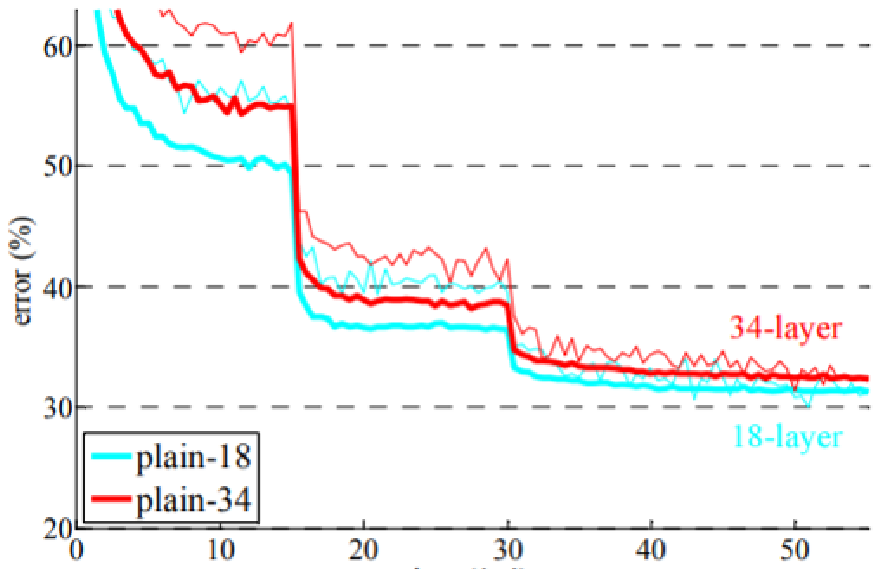


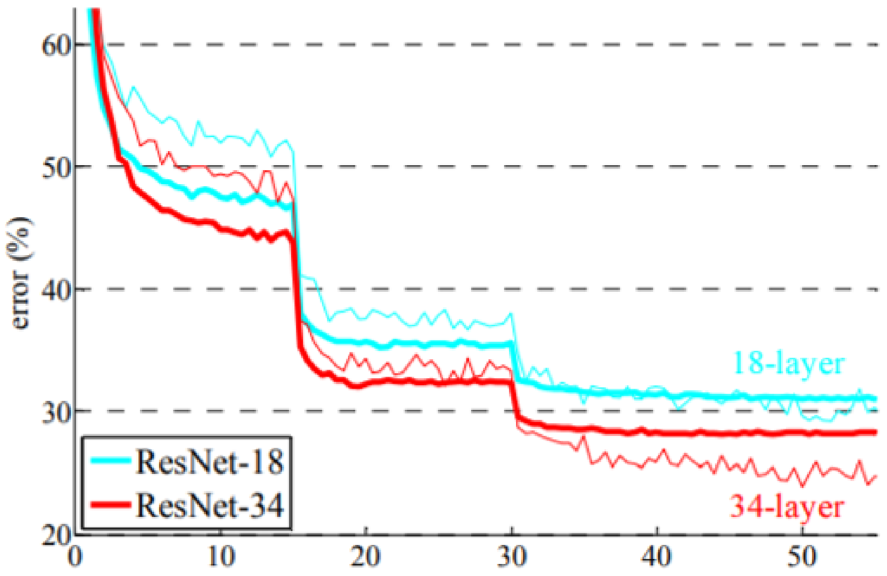
3. DEEP CONVOLUTIONS ARCHITECTURES

3.6. ResNet

Deeper networks are able to learn more abstract and complex representations of data compared to shallow networks, owing to their increased depth and hierarchical structure.

But when we increase the number of layers, there is a common problem in deep learning associated with that called the **Vanishing/Exploding gradient**.



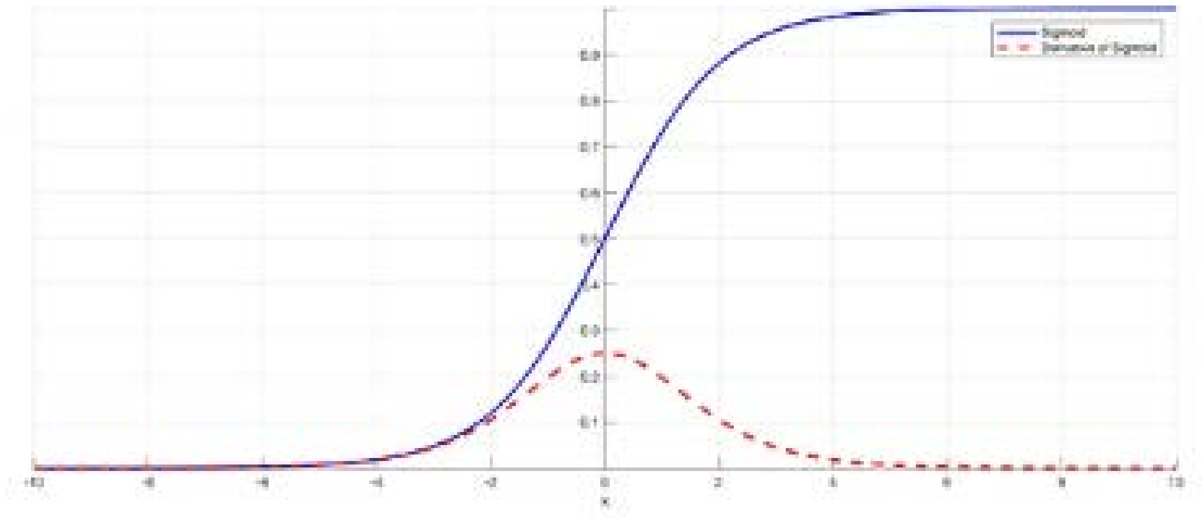


NOTE : VANISHING/EXPLODING GRADIENT

As more layers using certain activation functions are added to neural networks, the gradients of the loss function approaches zero, making the network hard to train.

When n hidden layers use an activation like the sigmoid function, n small derivatives are multiplied together. Thus, the gradient decreases exponentially as we propagate down to the initial layers.

A small gradient means that the weights and biases of the initial layers will not be updated effectively with each training session.



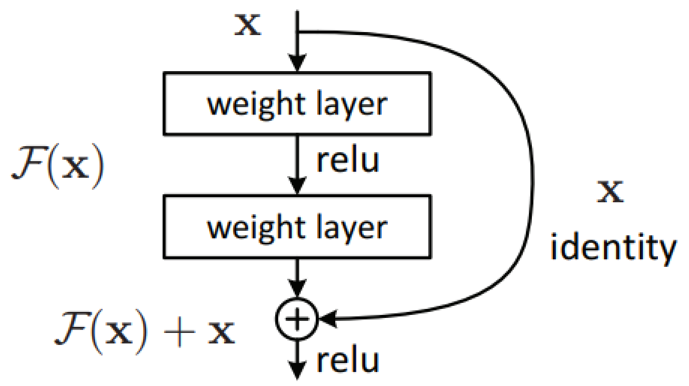
The sigmoid function and its derivative. Note how when the inputs of the sigmoid function becomes larger or smaller (when |x| becomes bigger), the derivative becomes close to zero.

3. DEEP CONVOLUTIONS ARCHITECTURES

3.6. ResNet

In order to solve the problem of the vanishing/exploding gradient, thi architecture introduced the concept called Residual Blocks. In this net we use a technique called ***skip connections***.

They enable the model to learn an identity function. This ensures that layers of the model do not perform any worse than the deeper layers.



x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

$$a^{[l+2]} = g(w^{[l+2]}a^{[l+1]} + b^{[l+2]} + a^{[l]}) \text{ , with } w^{[l+2]} \approx 0 \; and b^{[l+2]} \approx 0$$

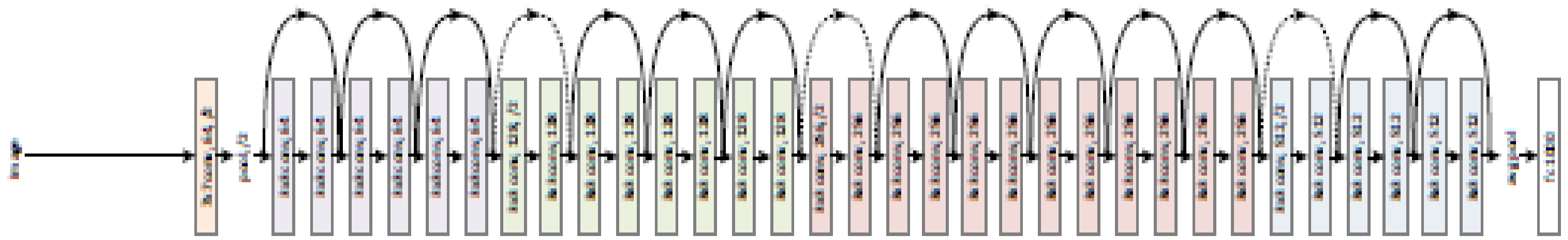
$$a^{[l+2]} = g(a^{[l]})$$

$$output = F(x) + x$$

3. DEEP CONVOLUTIONS ARCHITECTURES

3.6. ResNet

ResNet-18 has approximately 11 million parameters, ResNet-34 has approximately 21 million parameters, ResNet-50 has approximately 23 million parameters, ResNet-101 has approximately 42 million parameters, and ResNet-152 has approximately 58 million parameters. These numbers can vary slightly depending on the specific implementation and training process.



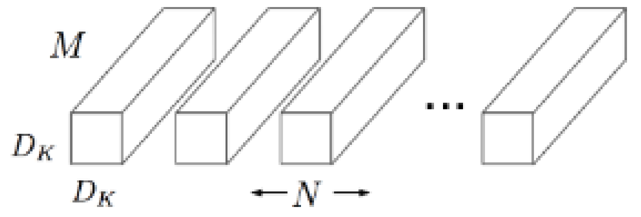
3. DEEP CONVOLUTIONS ARCHITECTURES

3.7. MobileNet

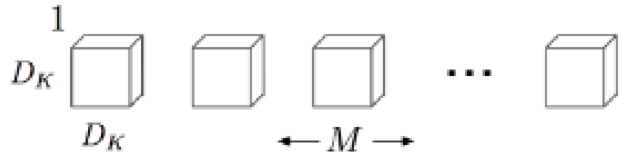
MobileNets primarily use Depthwise Seperable convolution to build deep models with a reduced number of parameters. It was primary designed for embedded vision applications.

To reduce the computational cost , MobileNet divide a simple convolution into Depth-wise convolution and a Point-wise convolution. The first layer is used to filter the input channels and the second layer is used to combine them to create a new feature

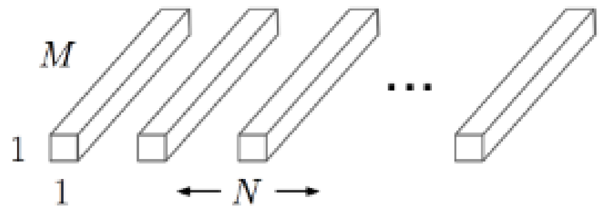
MobileNets introduce two new global hyperparameters(**width** multiplier and resolution **multiplie**r) that allow model developers to trade off **latency** or **accuracy** for speed and low size depending on their requirements.



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters

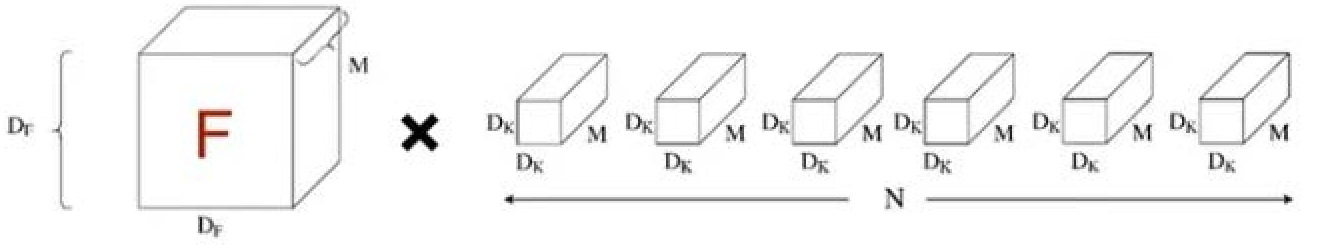


(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

3. DEEP CONVOLUTIONS ARCHITECTURES

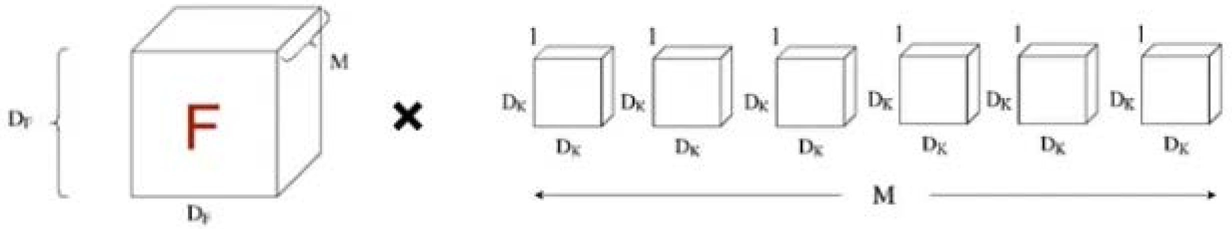
3.7. MobileNet

Standard Convolution



The computational cost : $D_K \times D_K \times M \times N \times D_F \times D_F$

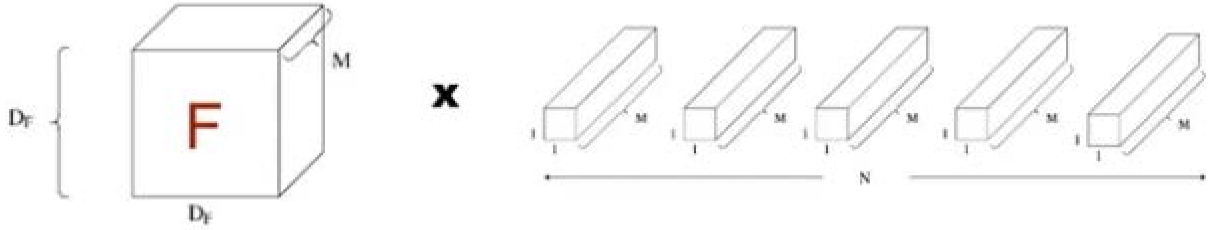
Depth-wise Convolution



The computational cost : $D_K \times D_K \times M \times D_F \times D_F$

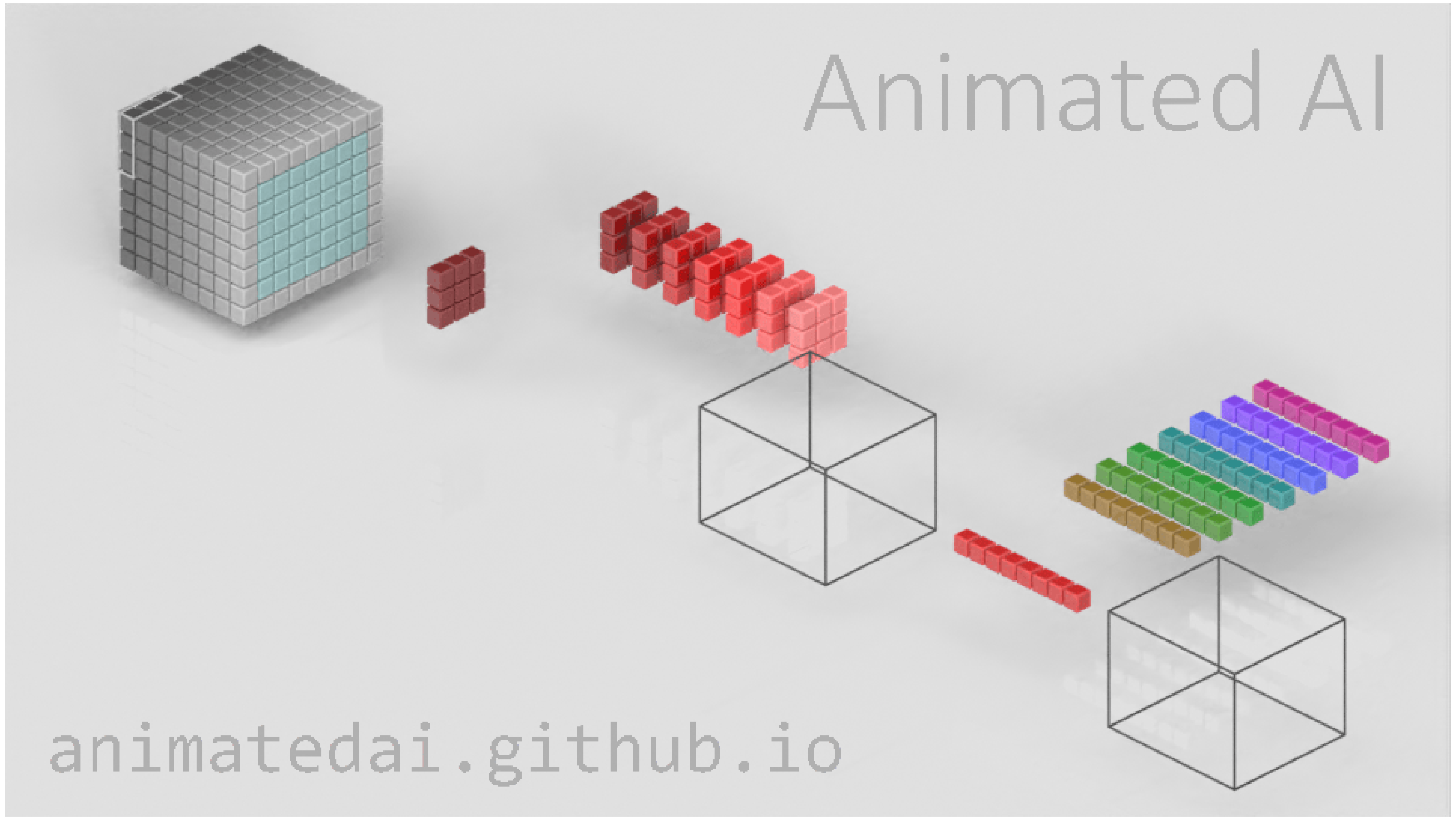
Point-wise Convolution

The computational cost :
$M \times N \times D_F \times D_F$



3. DEEP CONVOLUTIONS ARCHITECTURES

3.7. MobileNet

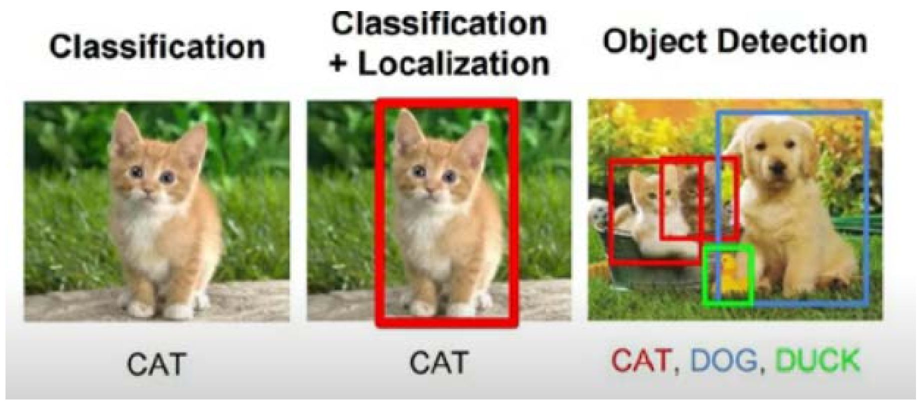


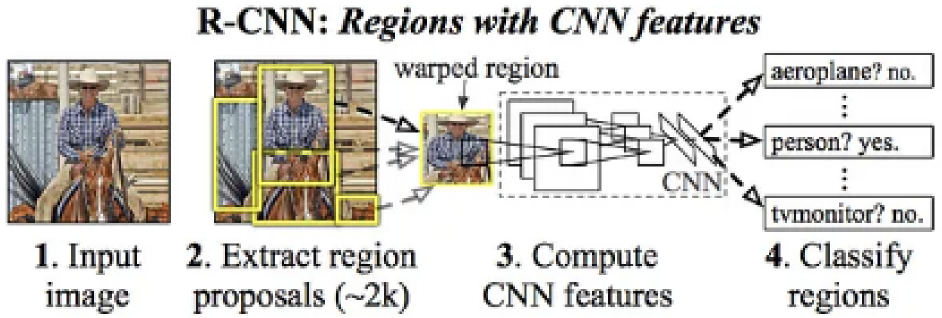
3. DEEP CONVOLUTIONS ARCHITECTURES

3.8. R-CNN

- R-CNN is a deep neural network used for object detection.

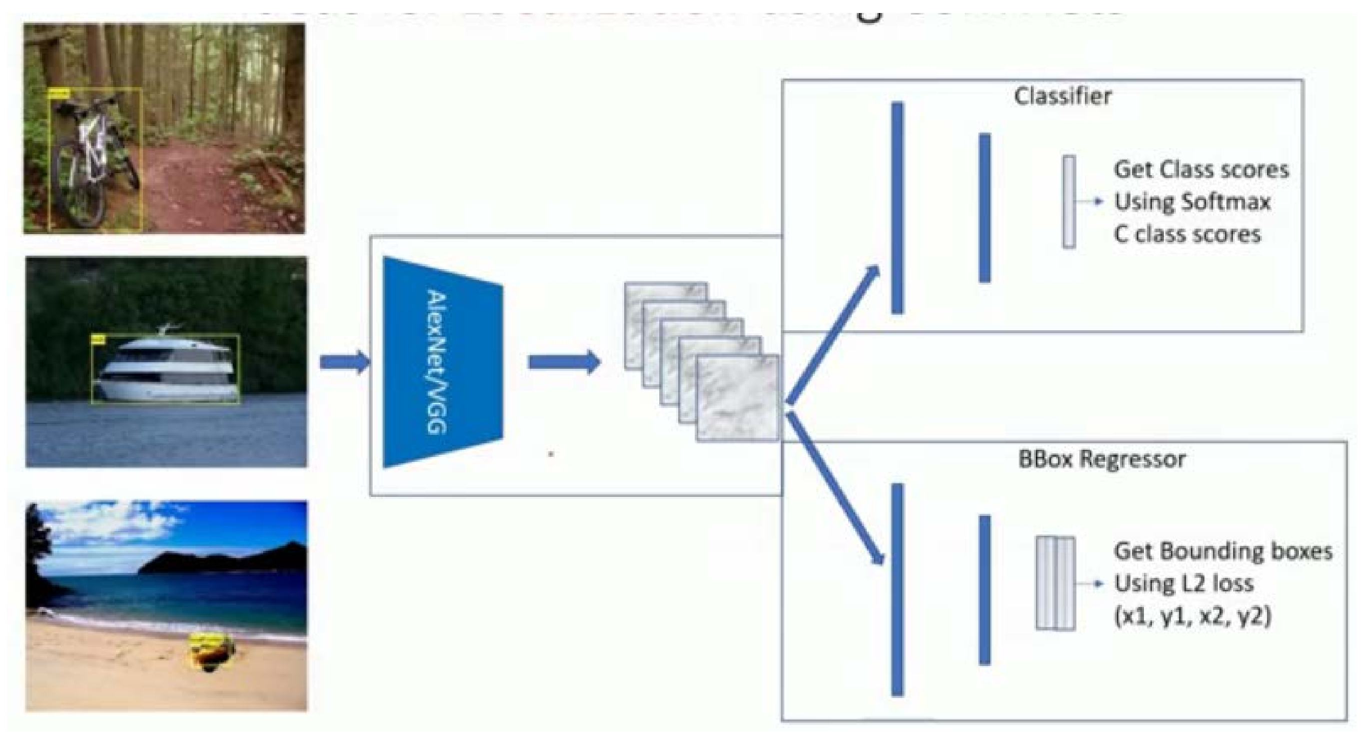
- The difference between object detection algorithms and classification algorithms is that in detection algorithms draw a bounding box around the object of interest to locate it within the image.
- R-CNN propose regions (based on the Selective Search (SS) approach) + Classify proposed regions. Output label+ Bounding Box.





3. DEEP CONVOLUTIONS ARCHITECTURES

3.8. R-CNN

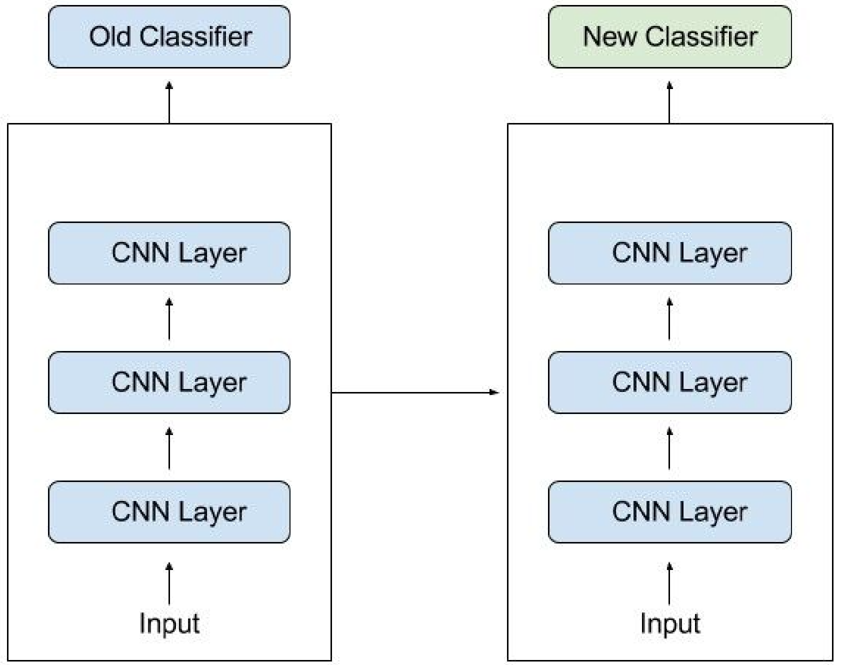


4. TRANSFER LEARNING IN CNNS

4.1. Transfer learning

Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task.

Transfer learning is used to prevent overfitting problems on limited volumes of data. For example, using the pretrained models on the ImageNet dataset, witch is composed of 1.2 million of images, for transfer learning to another dataset witch is composed of only 10 000 images.

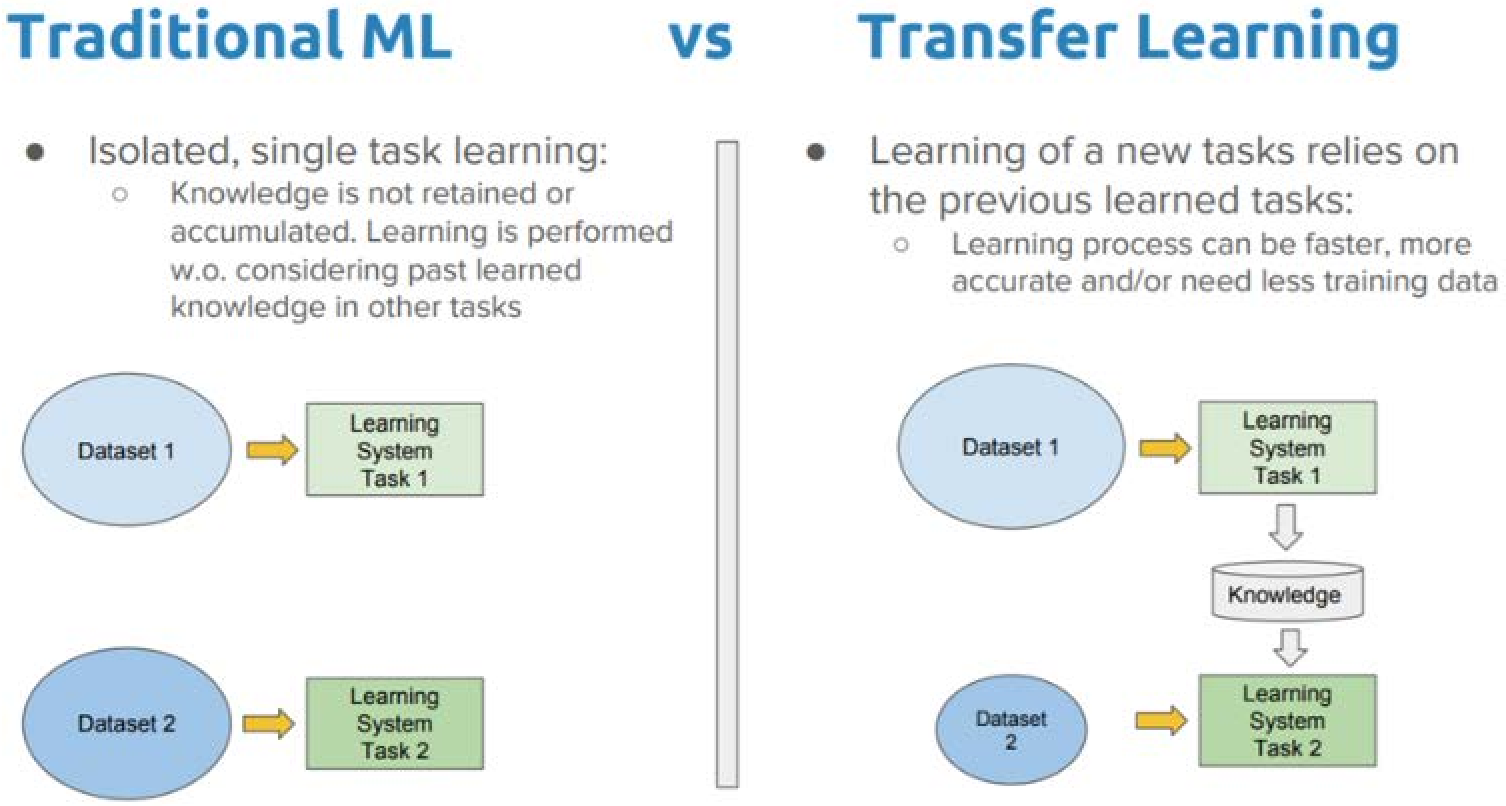


4. TRANSFER LEARNING IN CNNS

4.2. Transfer Learning vs Traditional ML

Traditional ML has an isolated training approach where each model is independently trained for a specific purpose, without any dependency on past knowledge.

Deep learning can use transfer learning due to its iterative learning process and hierarchical nature .


Traditional ML
vs
Transfer Learning
Isolated, single task learning:
Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks
Learning of a new tasks relies on the previous learned tasks:
Learning process can be faster, more accurate and/or need less training data
Dataset 1
Learning System Task 1
Dataset 2
Learning System Task 2
Dataset 1
Learning System Task 1
Knowledge
Dataset 2
Learning System Task 2

Traditional Machine Learning vs. Transfer Learning

4. TRANSFER LEARNING IN CNNS

3.4. Approaches to Transfer Learning

1. TRAINING A MODEL TO REUSE IT

Train the deep neural network on task B and use the model as a starting point for solving task A.

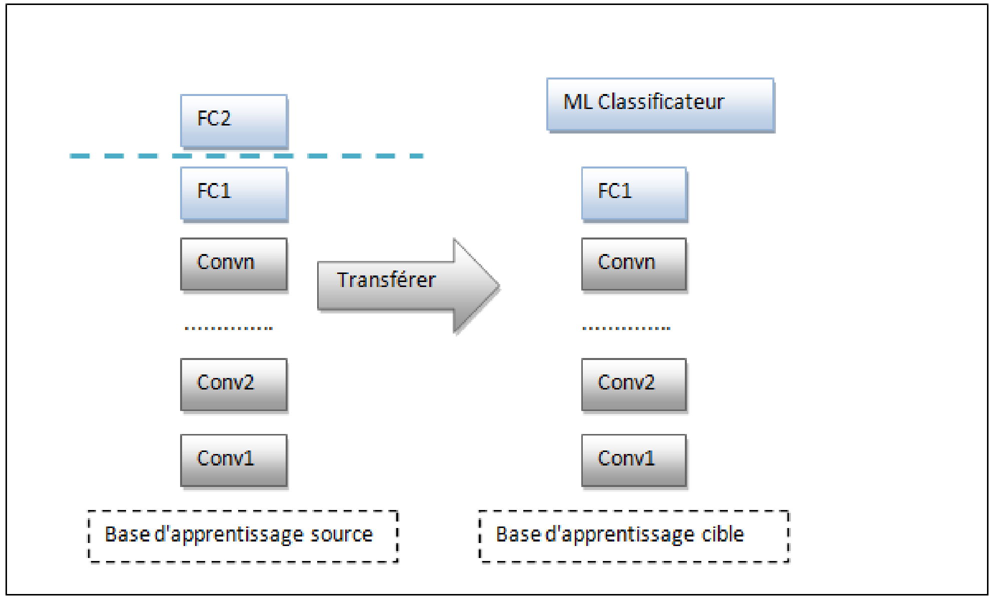
2. USING A PRE-TRAINED MODEL

The second approach is to use an already pre-trained model., such as : ImageNet models.

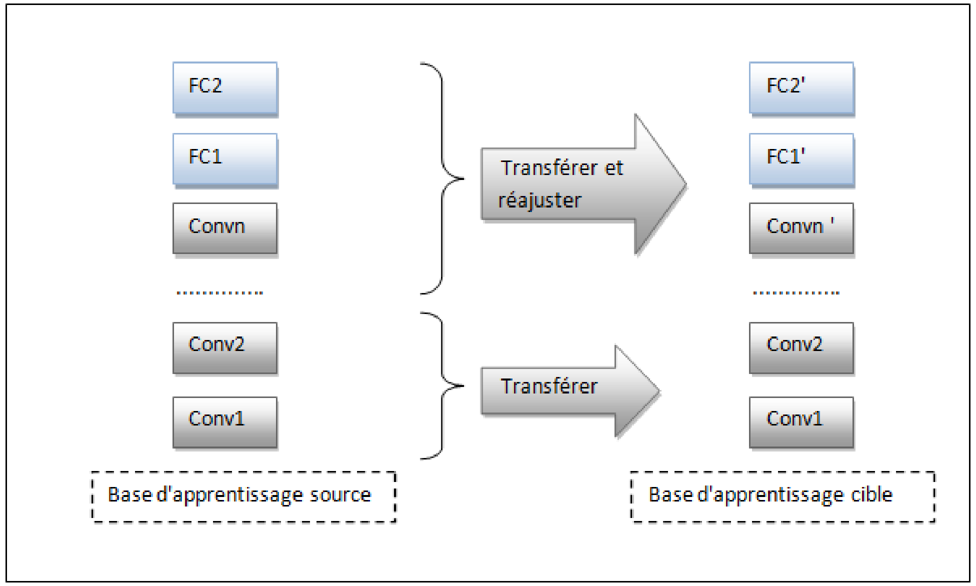
4. TRANSFER LEARNING IN CNNS

3.4. How Pre-trained models can be used?

1. AS FEATURE EXTRACTORS



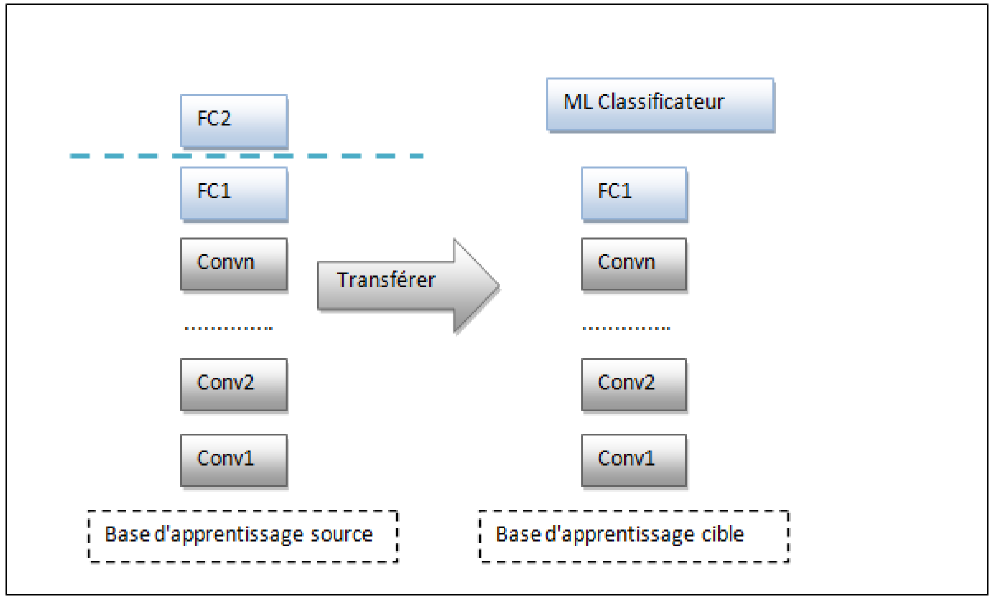
2. TRANSFER LEARNING AND FINE TUNING A SUBSET OF LAYERS

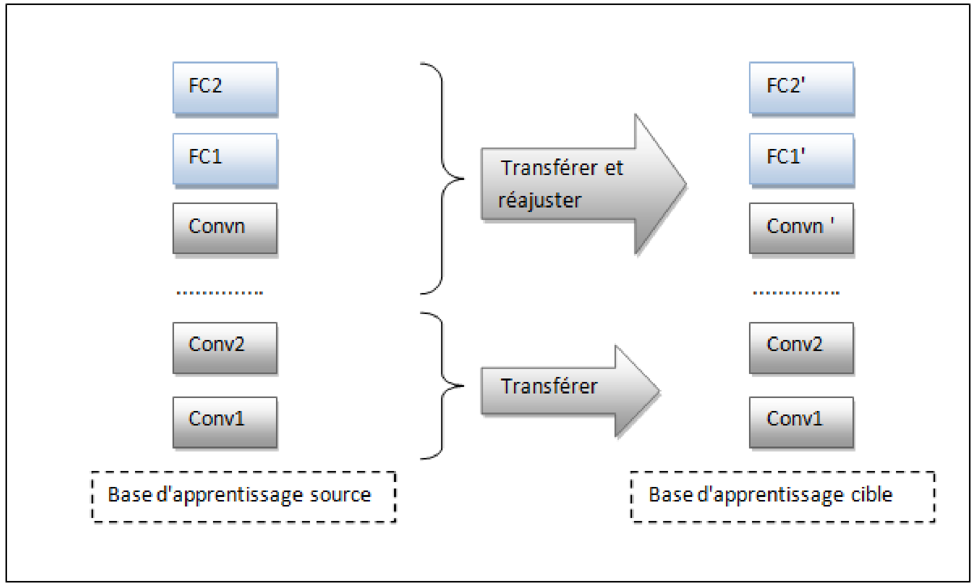


3. TRANSFER LEARNING AND FINE TUNING ALL LAYERS

4. TRANSFER LEARNING IN CNNS

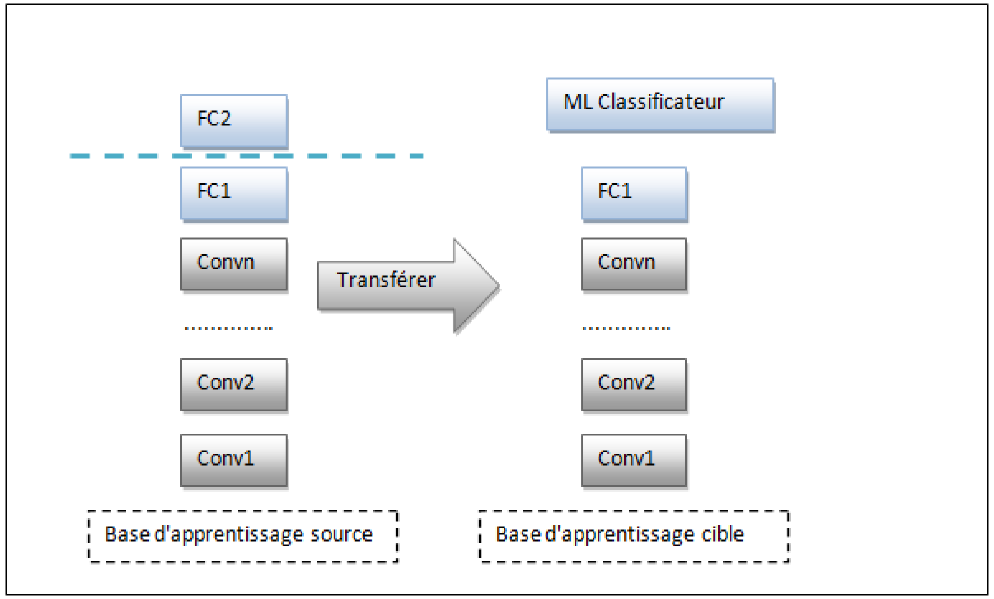
3.9. EfficientNet

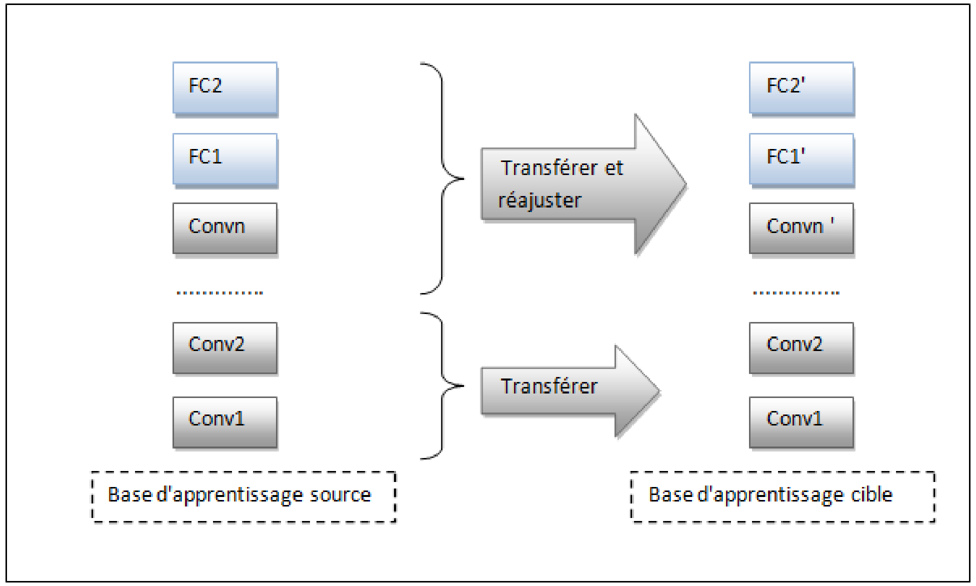




4. TRANSFER LEARNING IN CNNS

3.9. EfficientNet


FC2
ML Classificateur
FC1
FC1
Convn
Transférer
Convn
....
....
Conv2
Conv2
Conv1
Conv1
Base d'apprentissage source
Base d'apprentissage cible


FC2
FC2'
FC1
Transférer et réajuster
FC1'
Convn
Convn '
....
....
Conv2
Conv2
Transférer
Conv1
Conv1
Base d'apprentissage source
Base d'apprentissage cible