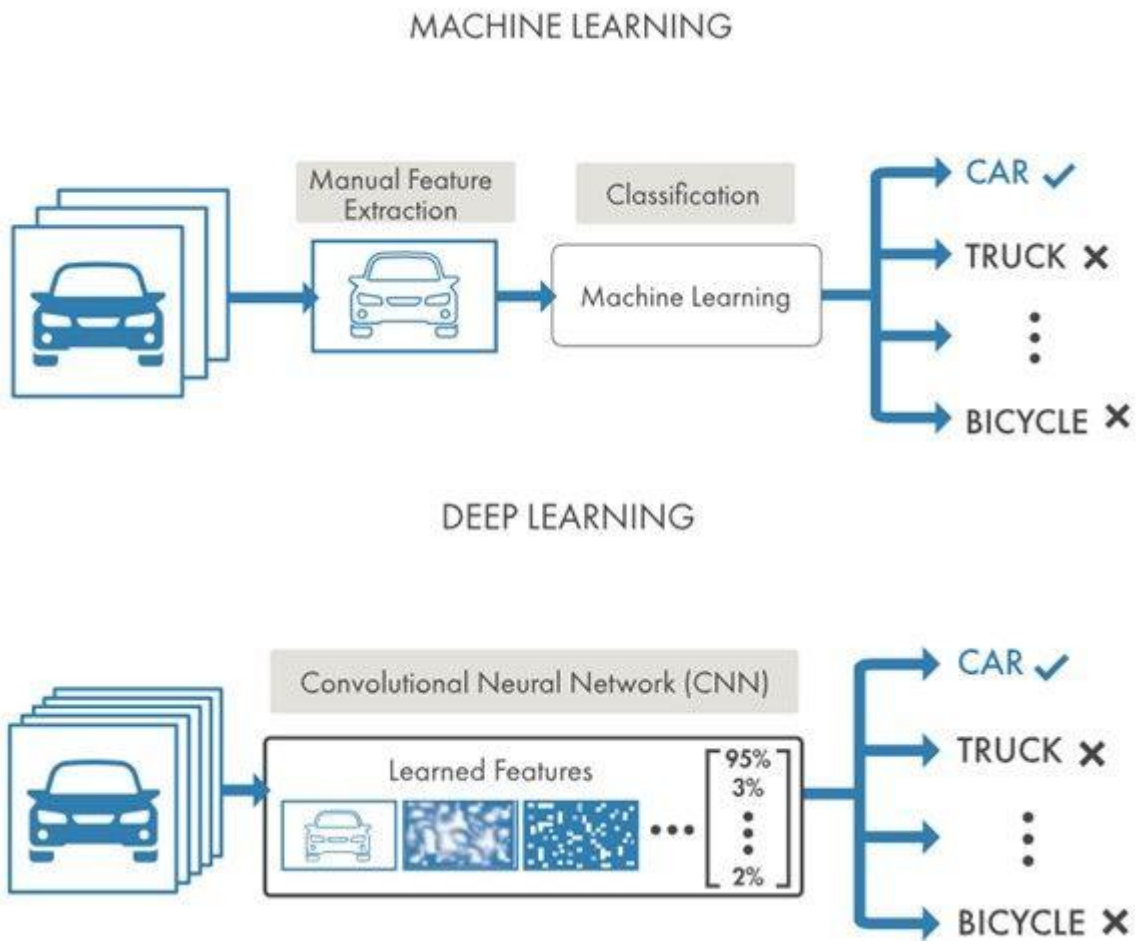
Apprentissage automatique et Apprentissage profond

Apprentissage profond = découvrir de manière automatiquement les caractéristiques importantes pour la classification.



Réseaux de neurones profonds (Deep Neural Network)

Plusieurs architectures existantes en fonction du champ d'application

Apprentissage supervisé

- Convolutional neural networks (CNN)
- Recurrent Neural Networks (RNN)
- Long Short Term Memory networks (LSTM)

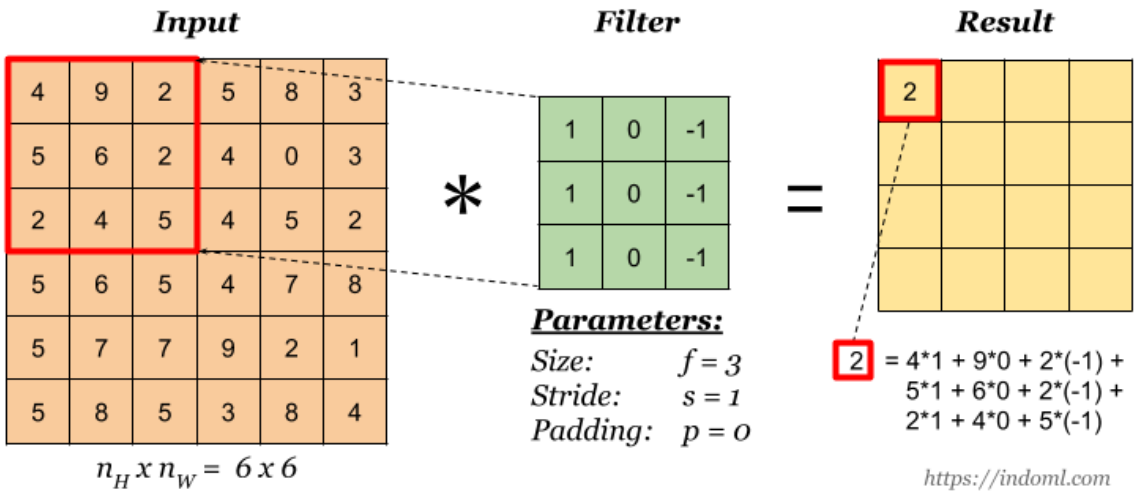
Apprentissage non supervisé

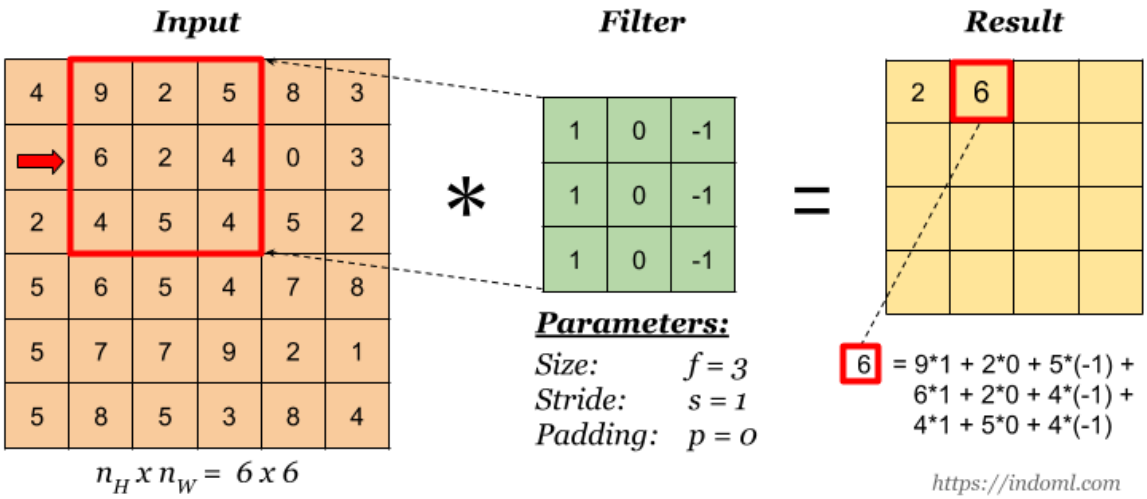
- Restricted Boltzmann machine (RBM)
- Autoencoder
- Deep Belief Networks

Convolutional Neural Networks (CNN) ou ConNet

Le terme ”convolutional” vient de l’opération de convolution de matrices utilisée dans le traitement de signal. Dans les convNet, 2 nouveaux types de couche ont été ajoutés dans le réseau : la couche convolution (convolutional layer) et la couche de mise en commun (pool layer).

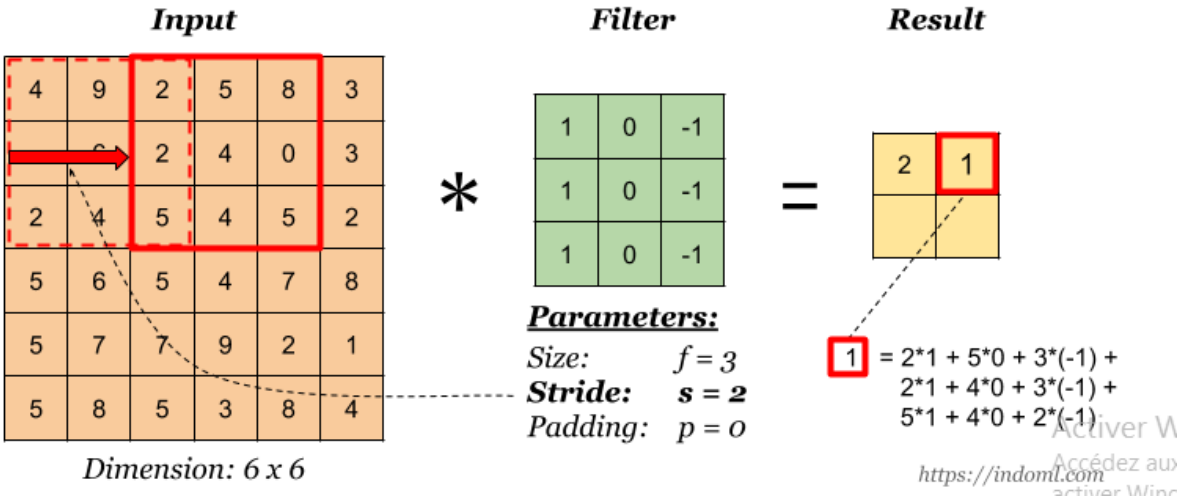
Opération de Convolution





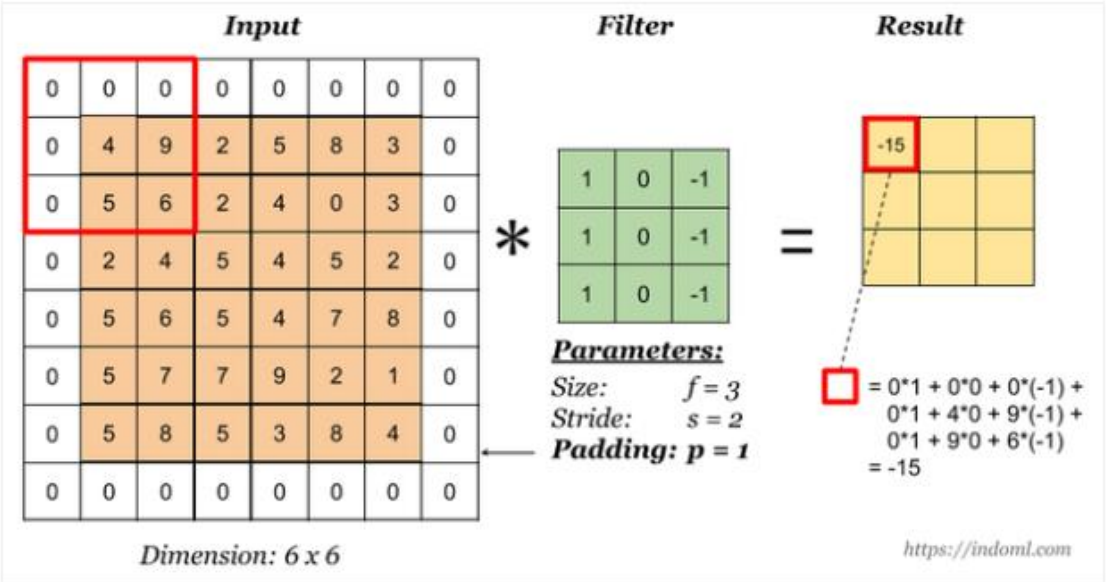
The total number of multiplications to calculate the result above is (4 x 4) x (3 x 3) = 144.

Stride

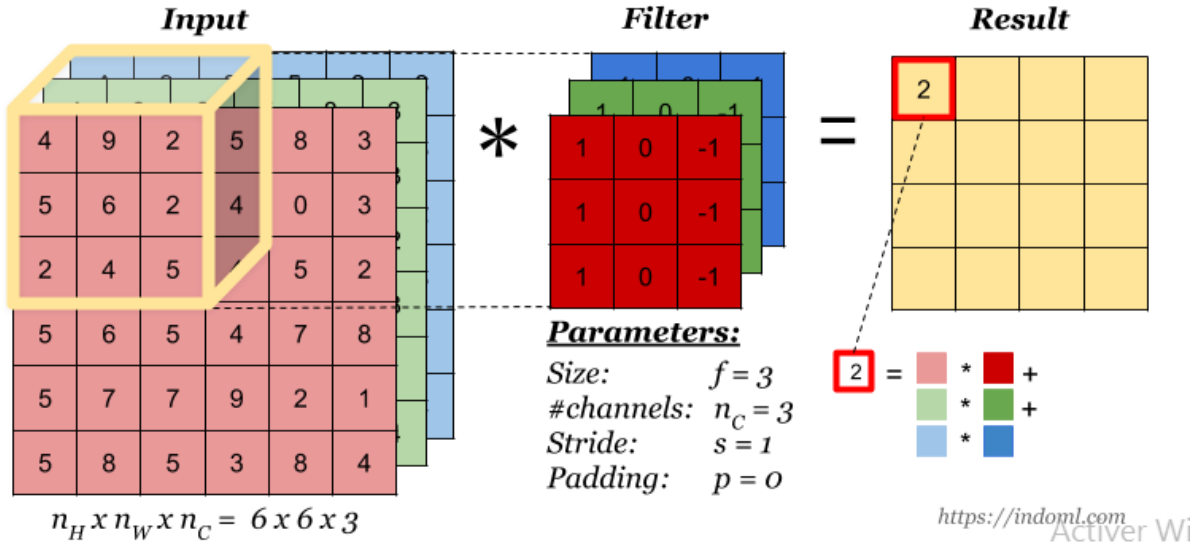


The total number of multiplications to calculate the result above is (2 x 2) x (3 x 3) = 36.

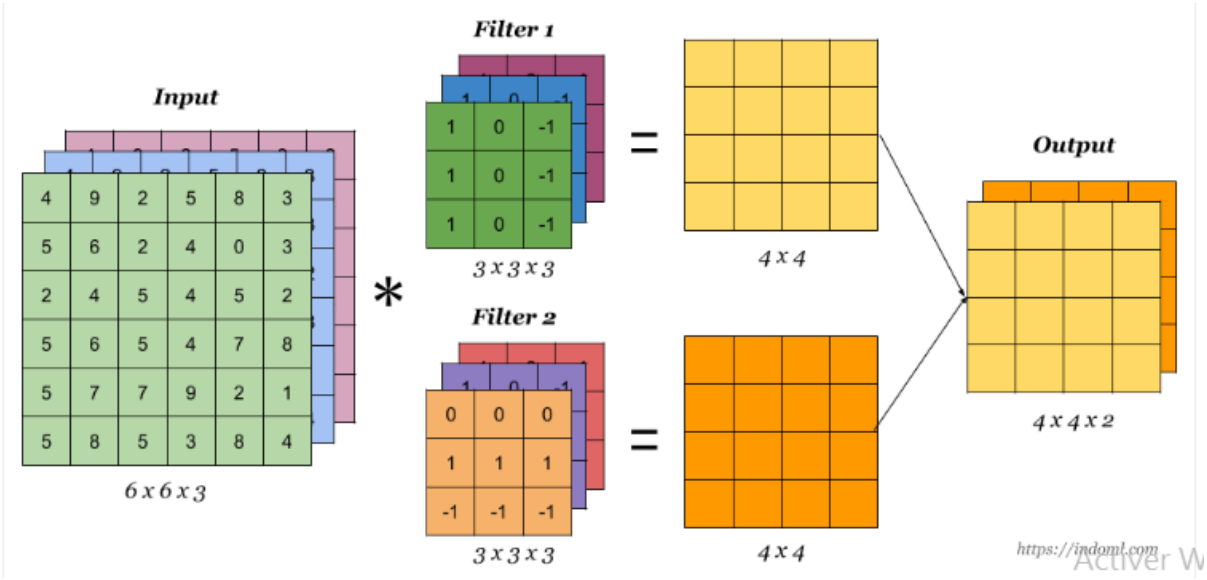
Padding



Convolution Operation on Volume



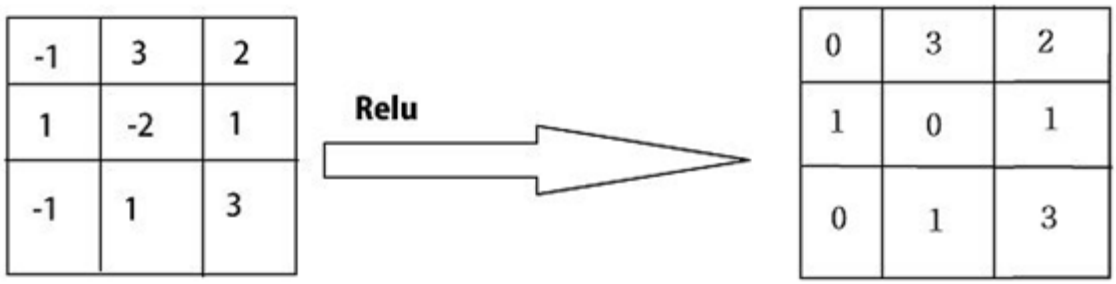
Convolution Operation with Multiple Filters

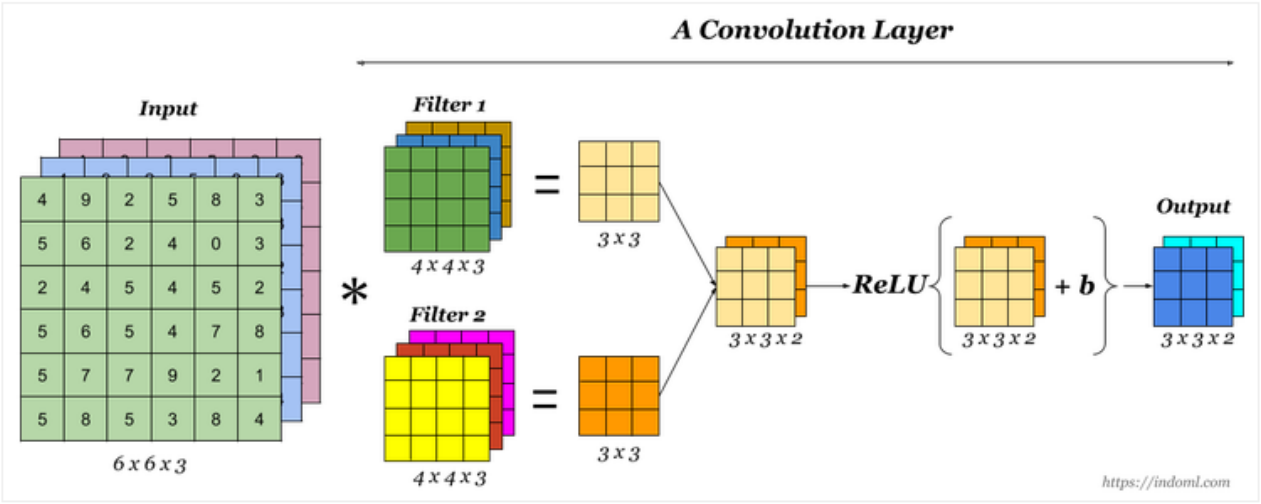


The total number of multiplications to calculate the result is (4 x 4 x 2) x (3 x 3 x 3) = 864.

One Convolution Layer

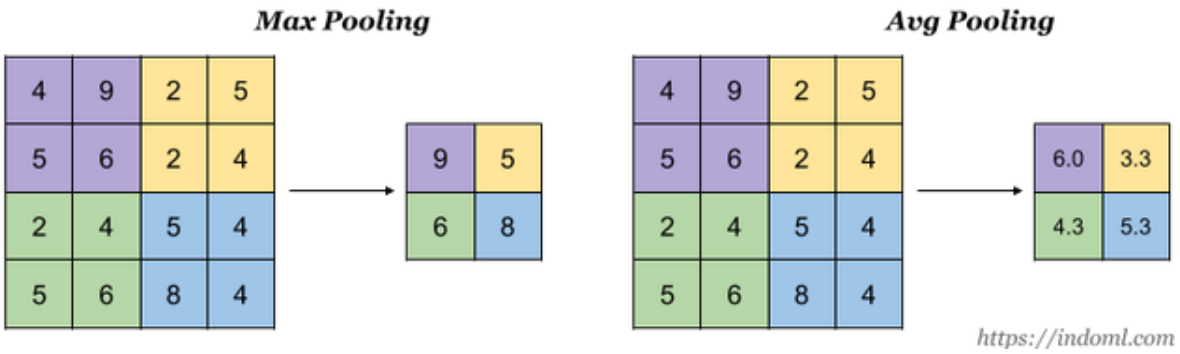
Finally to make up a convolution layer, a bias (ϵ R) is added and an activation function such as **ReLU** or **tanh** is applied.





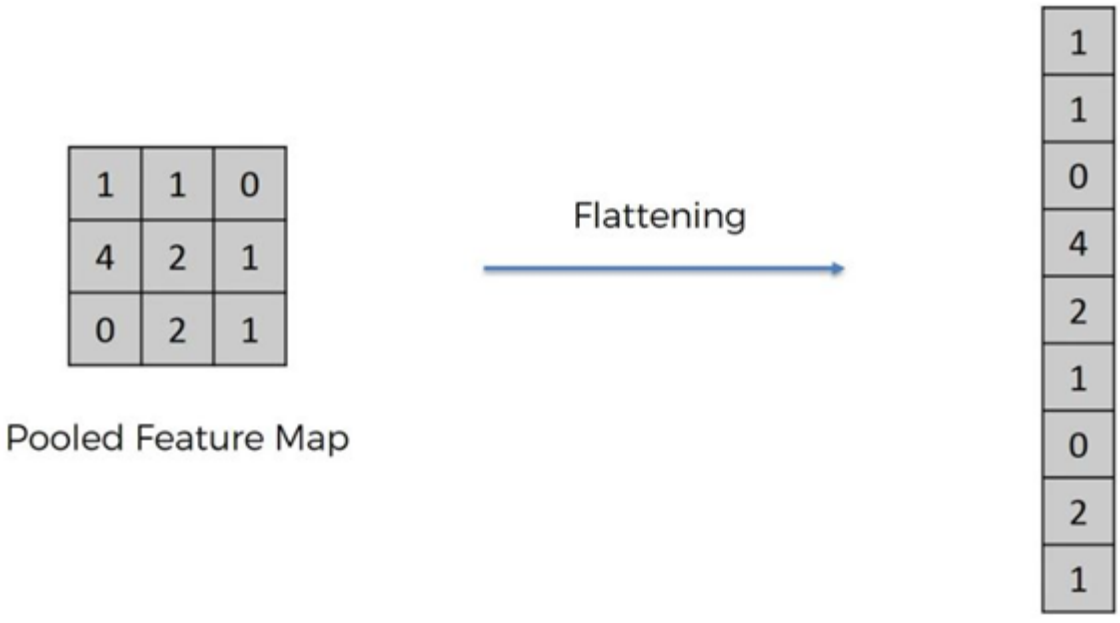
Pooling Layer

Pooling layer is used to reduce the size of the representations and to speed up calculations. Sample types of pooling are **max pooling** and **avg pooling.**



Flattening

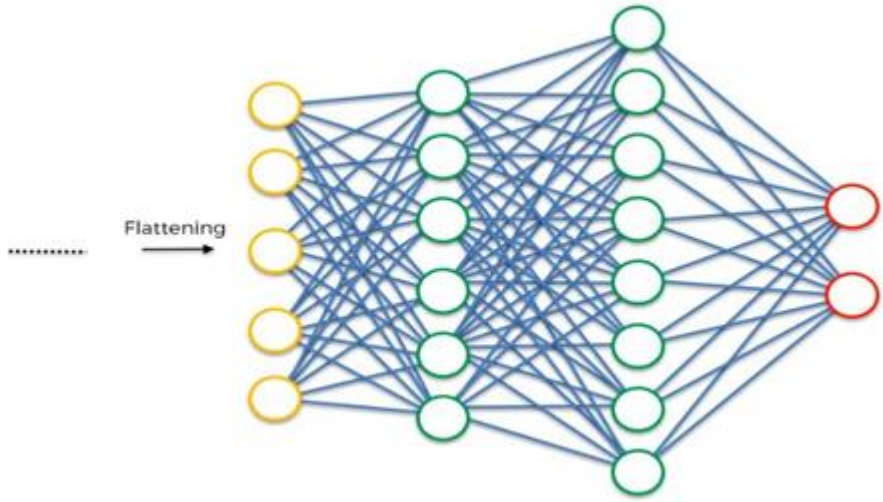
Flattening is the process of converting al the resultant 2-dimensional arrays into a single long continuous linear vector.



So Flattening is become the input of Artificial Neural Network.

Fully Connected Layer (FC Layer) or classification

The Fully Connected layer is a traditional multi-Layer Perceptron that uses a softmax activation function in the output layer. The term “Fully Connected” implies that every neuron in the previous layer is connected to every neuron on the next layer.

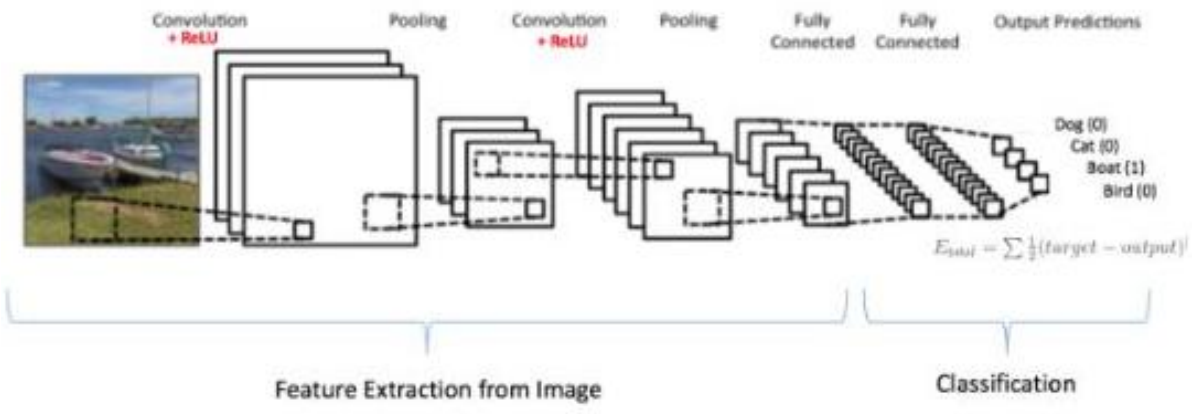


The output from the convolutional and pooling layers represent high-level features of the input image. The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset

Training process of the CNN

The convolution + Pooling layers act as Features extractor from the input image while a fully connected layer acts as the classifier.

Input Image = Boat ; Target Vector = [0, 0, 1, 0]



Step1

Initialization all filters and parameters/weights with random values.

Step2

The network takes a training image as input, goes through the forward propagation step (convolution, ReLU, and pooling operations along with forwarding propagation in the Fully Connected layer) and finds the output probabilities for each class. For example, let be the output probabilities for the boat image are [0.2, 0.4, 0.1, 0.3]

Step3

Calculate the total error at the output layer (summation over all 4 classes)

$$\text{Total Error} = \sum \tfrac{1}{2} (\text{target probability} - \text{output probability})^2$$

Step4

Use Backpropagation to calculate the *gradients* of the error concerning all weights in the network and use *gradient descent* to update all filter values/weights and parameter values to minimize the output error.

When the same image is input again, output probabilities might now be [0.1, 0.1, 0.7, 0.1], which is closer to the target vector [0, 0, 1, 0]. This means that the network has *learn*ed to classify this particular image correctly by adjusting its weights/filters such that the output error is reduced.

Parameters : number of filters, filter sizes, architecture of the network have all been fixed before Step 1 and do not change during the training process — only the values of the filter matrix and connection weights get updated.

Step5

Repeat steps 2–4 with all images in the training set.

Apprentissage par transfert (Transfert Learning)

Le principe est d'utiliser les connaissances acquises par un réseau de neurones lors de la résolution d'un problème afin d'en résoudre un autre plus ou moins similaire. On réalise ainsi un transfert de connaissances, d'où le nom Transfer Learning (apprentissage par transfert). On retrouve comme types de transfert d'apprentissage.

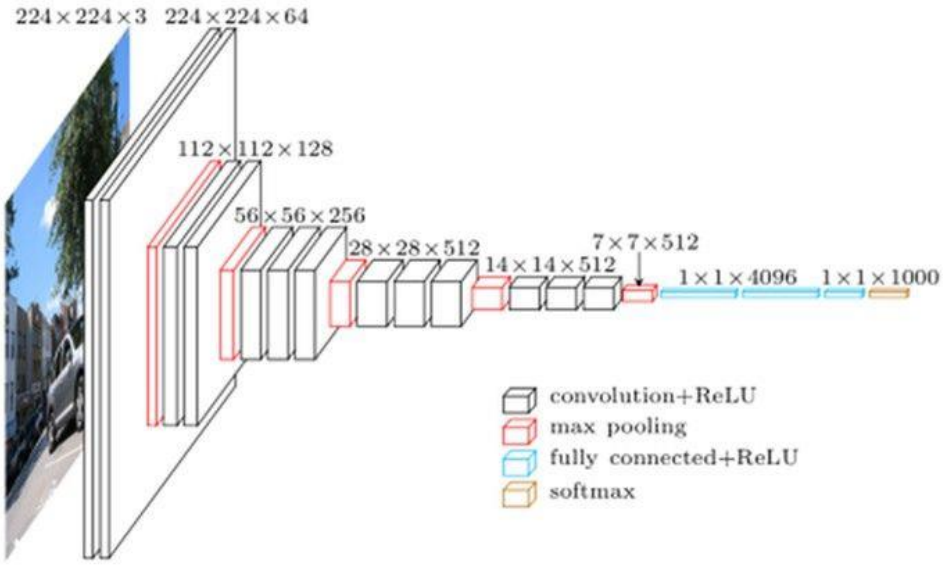
L'extraction de variables du ConvNet

Le ConvNet est utilisé comme un extracteur, c'est à dire qu'un vecteur est extrait à partir d'une certaine couche du modèle sans rien modifier à sa structure ou son poids et le vecteur précédemment extrait est utilisé pour une nouvelle tâche.

Le réglage fin du modèle de ConvNet

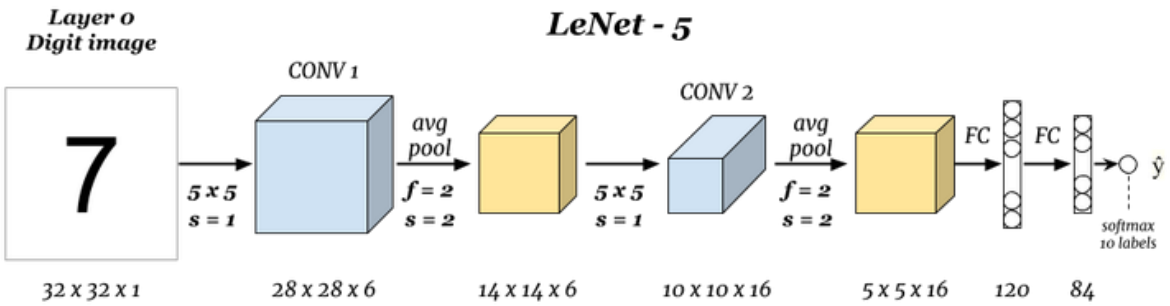
Le nouveau ConvNet est initialisé avec les poids et la structure du modèle pré-entrainé à utiliser. La structure du modèle pré-entrainé est légèrement modifiée pour la nouvelle tâche, et enfin le nouveau modèle est entrainé pour la nouvelle tâche.

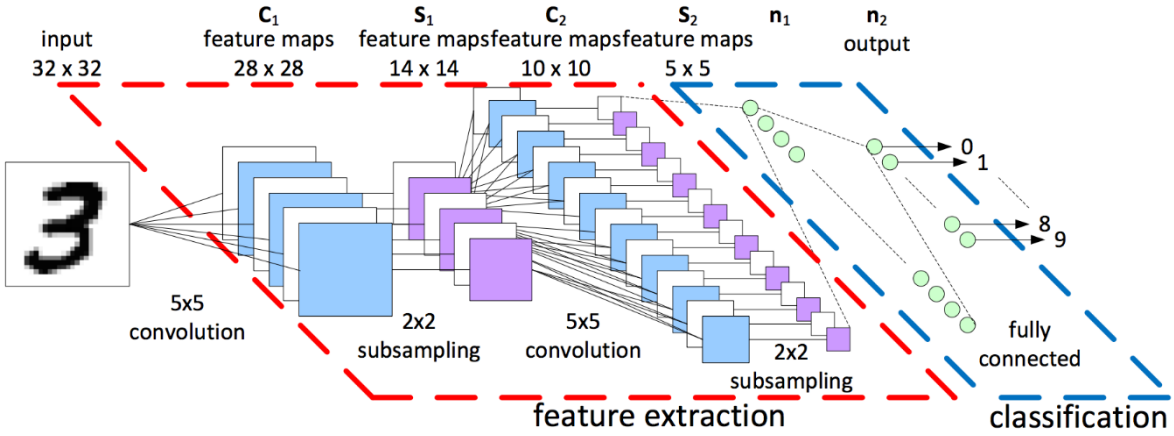
Représentation 3D de l'architecture de VGG-16



from keras.applications.vgg16 import VGG16

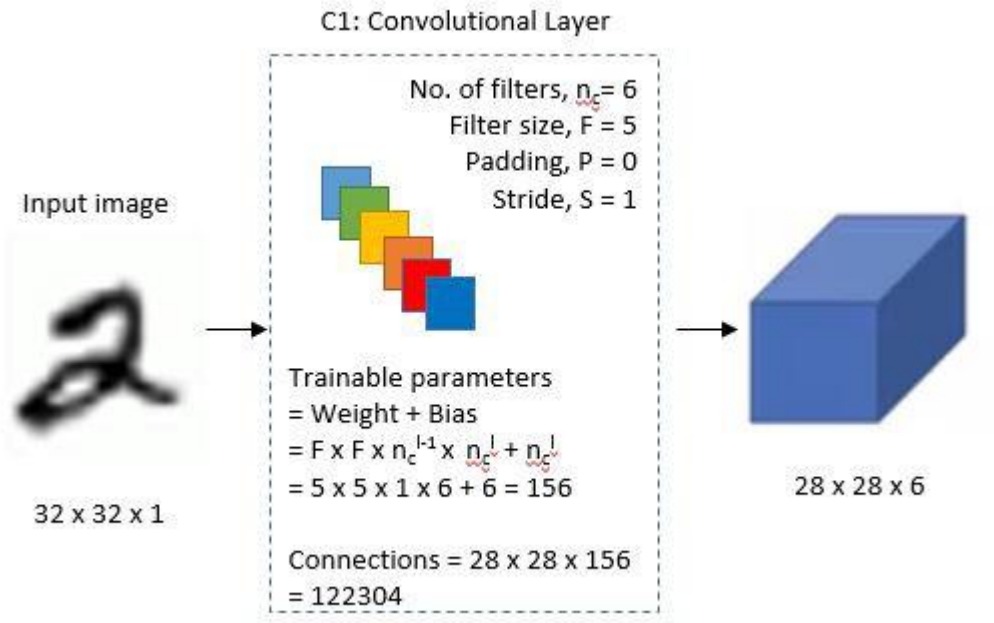
Classic Network: LeNet – 5





First Layer : Convolution layer C1

6 filtres (or feature maps) of size 5x5, Stride=1, Padding= 0



The input for LeNet-5 is a 32×32 grayscale image which passes through the first convolutional layer with 6 filters (or feature maps) having size 5×5 and a stride = 1.

The image dimensions changes from 32x32x1 to 28x28x6.

$$\text{OutputWidth} = \left(\frac{W - Fw + 2P}{Sw}\right) + 1$$

$$\textbf{OutputWidth} = \left(\frac{32 - 5 + 0}{1}\right) + 1 = 27 + 1 = 28$$

$$\text{OutputHeight} = \left(\frac{H - Fh + 2P}{Sh}\right) + 1$$

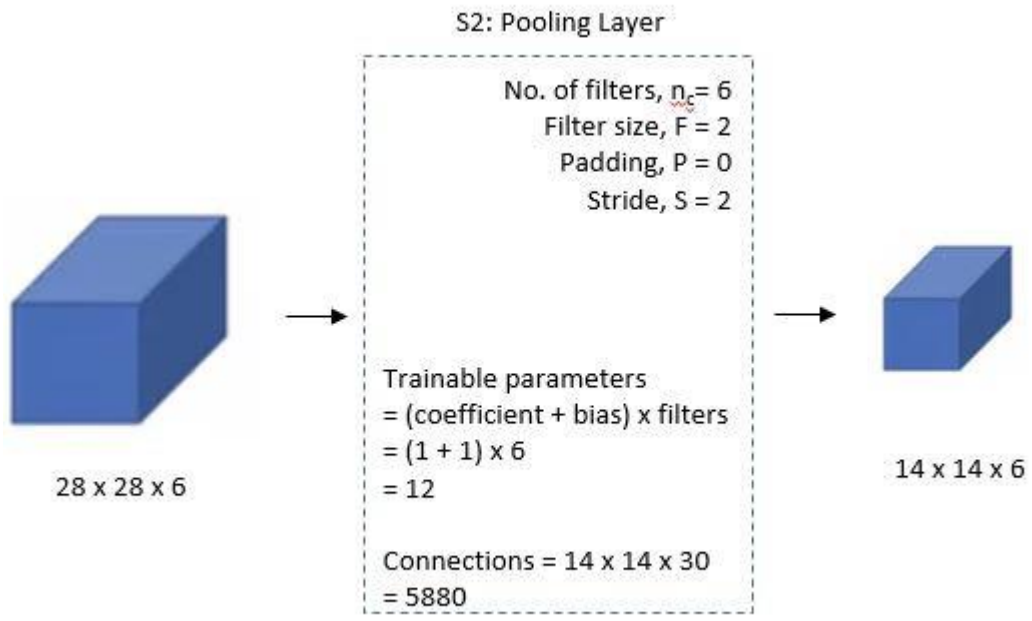
$$\textbf{OutputHeight} = \left(\frac{32 - 5 + 0}{1}\right) + 1 = 27 + 1 = 28$$

```
trainable parameters = (weight * input maps + bias) * feature maps
                     = (5 * 5 * 1 + 1) * 6 = 156
```

```
connections = (input + bias) * feature maps * feature map size
            = (5 * 5 + 1) * 6 * 28 * 28 = 122304
```

2nd layer : Subsampling (Pooling) layer (S2)

6 Filtres (Feature maps) of size 2x2, Stride=2, Padding = 0



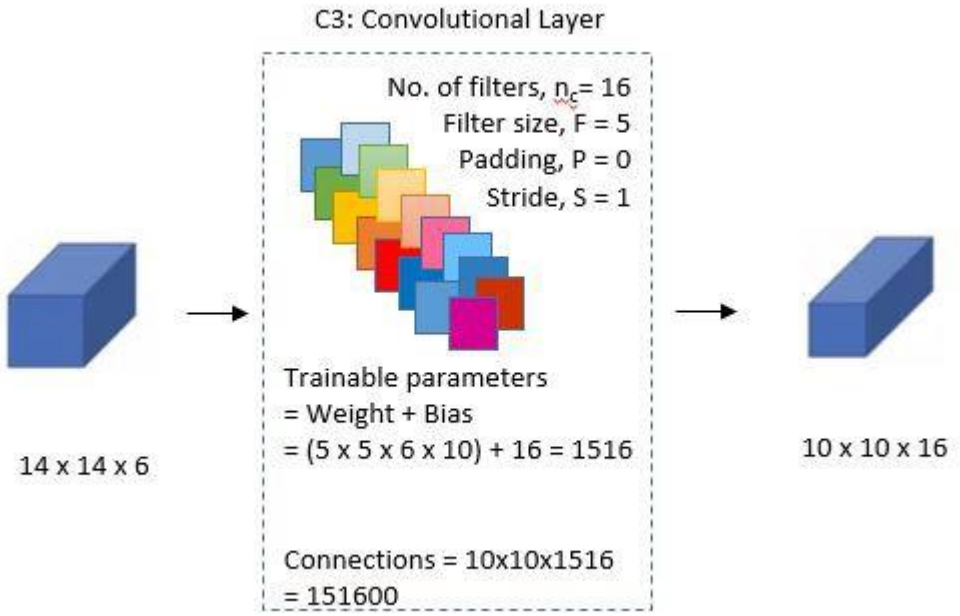
$$OM = \left(\frac{IM + 2P - F}{S}\right) + 1$$

OM = $\left(\frac{28 + 2*0 - 2}{2}\right)$ + 1 = 14, donc Output Matrix= 14 x 14

```
trainable params = (weight + bias) * feature maps
                 = (1 + 1) * 6 = 12
```

```
connections = (input + bias) * feature maps * feature map size
            = (2 * 2 + 1) * 6 * 14 * 14 = 5880
```

3rd layer : Convolution layer (C3)



Calculate Feature Map

$$\text{OutputWidth} = \left(\frac{W - Fw + 2P}{Sw}\right) + 1$$

OutputWidth = (14 – 5 + 0 / 1) + 1 = 10

$$\text{OutputHeight} = \left(\frac{H - Fh + 2P}{Sh}\right) + 1$$

OutputHeight = (14 – 5 + 0 / 1) + 1 = 10. Donc : Output Matrix = 10 x 10

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

TABLE I

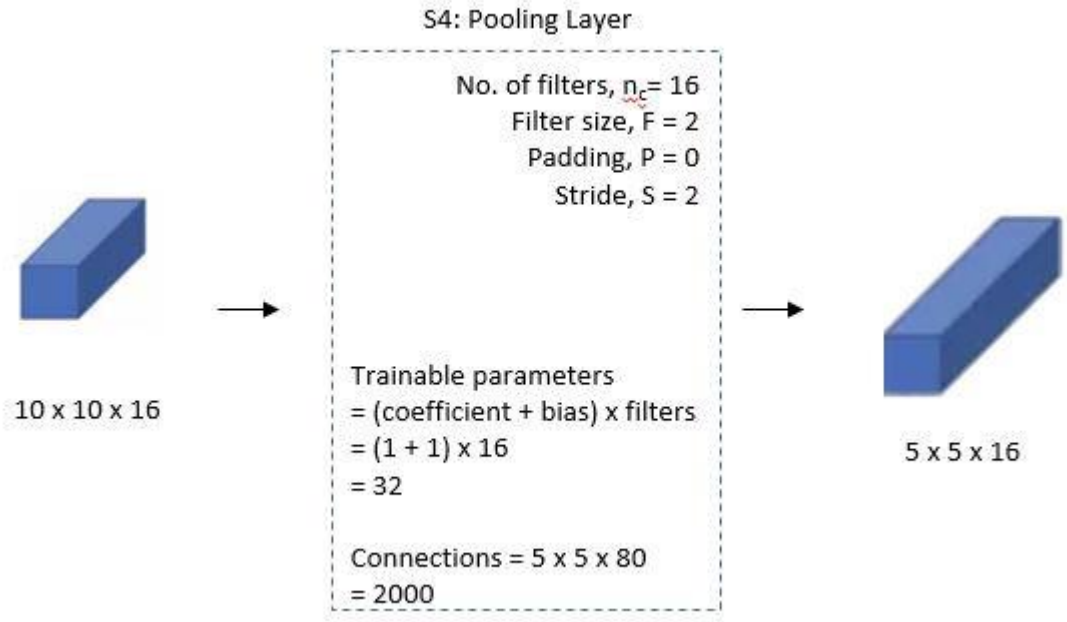
EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED BY THE UNITS IN A PARTICULAR FEATURE MAP OF C3.

- First 6 feature maps are connected to 3 contiguous input maps each (overlapping 2 maps)
- Second 6 feature maps are connected to 4 contiguous input maps (overlapping 3 maps)
- Next 3 feature maps are connected to 4 discontinuous input maps (overlapping 1 map)
- Last 1 feature map are connected to all 6 input maps

```
trainable params = (weight * input maps + bias) * feature maps

1st group = (5 * 5 * 3 + 1) * 6 = 456

2nd group = (5 * 5 * 4 + 1) * 6 = 606

3rd group = (5 * 5 * 4 + 1) * 3 = 303

4th group = (5 * 5 * 6 + 1) * 1 = 151

All group = 456 + 606 + 303 + 151 = 1516
```

```
connections = (input + bias) * feature maps * feature map size
            = trainable params * feature map size
            = 1516 * 10 * 10 = 151600
```

4th Layer : Pooling Layer S4



$$\text{OM} = \left(\frac{IM + 2P - F}{S}\right) + 1$$

$$\text{OM} = \left(\frac{10 + 2 \times 0 - 2}{2}\right) + 1 = 5 \quad \text{Donc, Output Matrix} = 5 \times 5$$

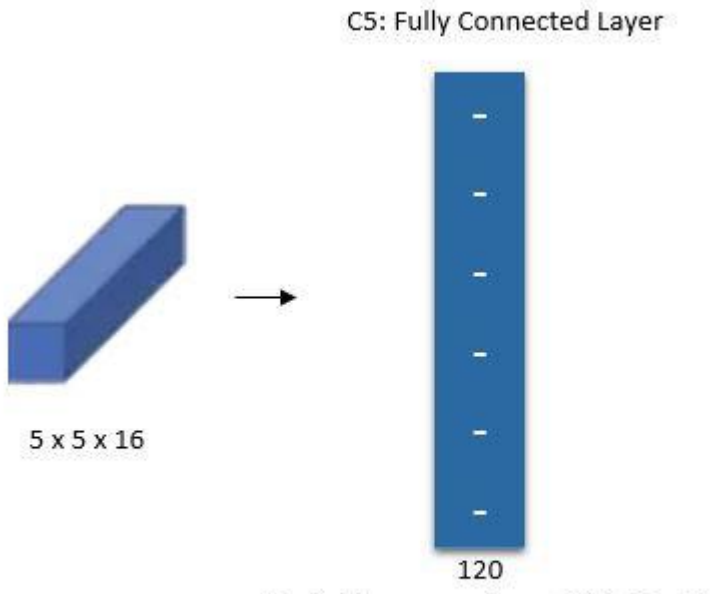
```
trainable params = (weight + bias) * feature maps
                 = (1 + 1) * 16 = 32
```

```
connections = (input + bias) * feature maps * feature map size
            = (2*2 + 1) * 16 * 5*5 = 2000
```

5th Layer : Convolution layer (C5)

The fifth layer (C5) is a fully connected convolutional layer with 120 feature maps each of size 1×1.

Each of the 120 units in C5 is connected to all the 400 nodes (5x5x16) in the 4th layer S4.

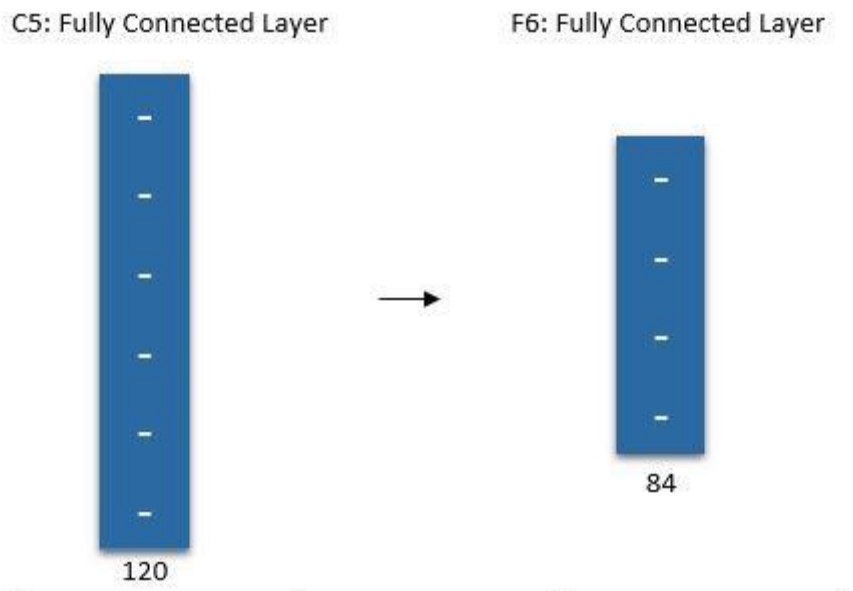



```
trainable params = (weight * input maps + bias) * feature maps
                 = (5 * 5 * 16 + 1) * 120 = 48120
```

```
connections = (input + bias) * feature maps * feature map size
            = trainable params * feature map size
            = 48120 * 1 * 1 = 48120
```

6th Layer : Fully Connected layer (F6)

This layer is just a simple neural network layer with `84` output neurons.

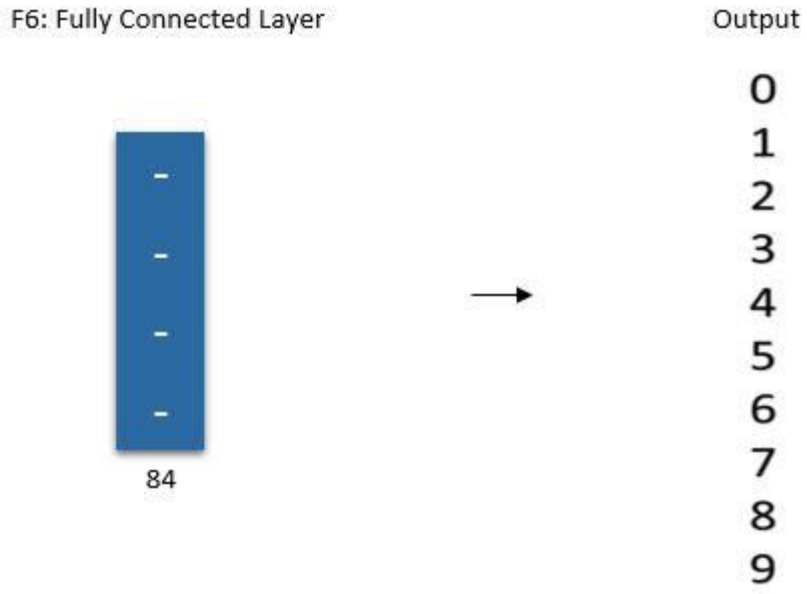



```
trainable params = connections = (input + bias) * output
                               = (120 + 1) * 84 = 10164
```

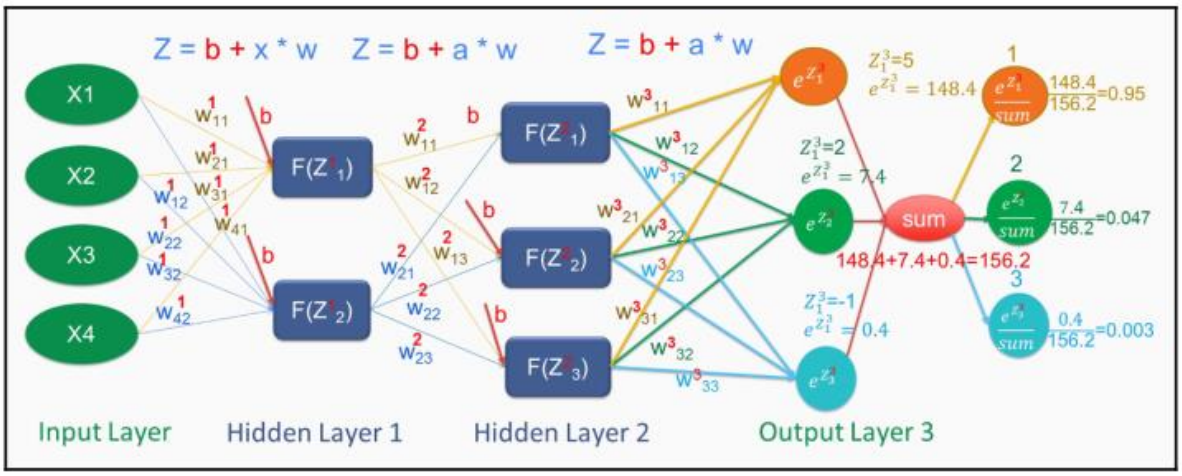
7th : Output layer

There is a fully connected softmax output layer ŷ with 10 possible values corresponding to the digits from 0 to 9. The softmax function is used in the last fully connected layer to be able to convert outputs from the previous layer into probabilities for each output class.

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K$$



Exemple :



$Z_1 = 5$, $e^{z_1^3} = 148.4$.

$Z_2 = 2$, $e^{z_2^3} = 7.4$.

$Z_3 = -1$, $e^{z_3^3} = 0.4$.

$e^{z_1^3} + e^{z_2^3} + e^{z_3^3} = 156.2.$

Each of these values is divided by the sum to attain the final percentages.
For class 1, we get 95%, class 2 gives us 4.7%, and class 3 gives us 0.3%.

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	32x32	-	-	-
1	Convolution	6	28x28	5x5	1	tanh
2	Average Pooling	6	14x14	2x2	2	tanh
3	Convolution	16	10x10	5x5	1	tanh
4	Average Pooling	16	5x5	2x2	2	tanh
5	Convolution	120	1x1	5x5	1	tanh
6	FC	-	84	-	-	tanh
Output	FC	-	10	-	-	softmax