

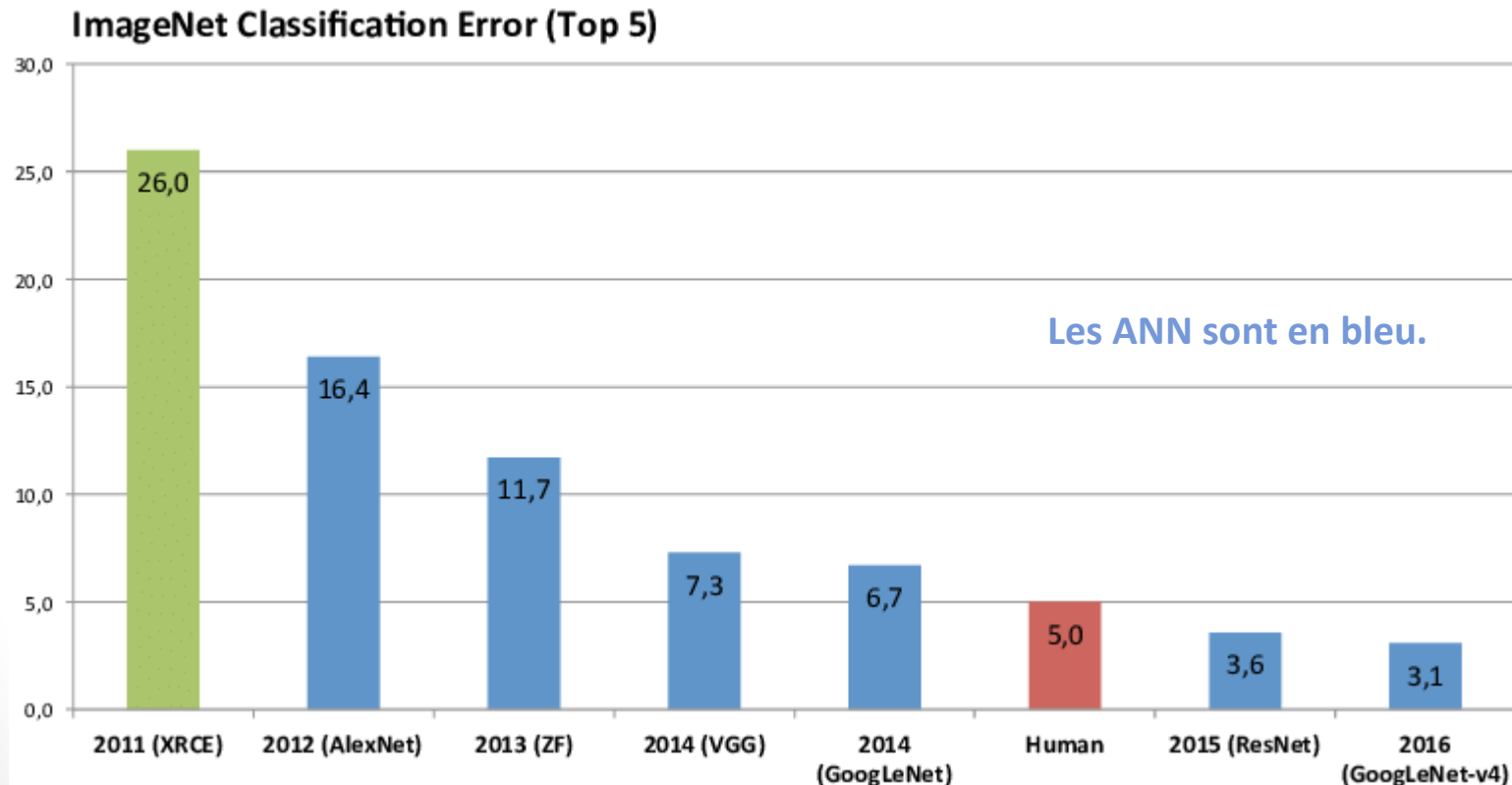
# Réseaux Convolutifs

ConvNet ou CNN : Convolutional Networks

# RAPPELS

# 2012 : Le Choc

- ImageNet Large Scale Visual Recognition Challenge est une compétition annuelle de reconnaissance d'image basée sur 10 millions d'images annotées et 1000 labels.
- En 2017 repris pas Kaggle.



# ImageNet



**mite**

**container ship**

**motor scooter**

**leopard**

	<b>mite</b> black widow cockroach tick starfish		<b>container ship</b> lifeboat amphibian fireboat drilling platform		<b>motor scooter</b> go-kart moped bumper car golfcart		<b>leopard</b> jaguar cheetah snow leopard Egyptian cat
--	---	--	---	--	--	--	---



**grille**

**mushroom**

**cherry**

**Madagascar cat**

	<b>convertible</b> grille pickup beach wagon fire engine		<b>agaric</b> mushroom jelly fungus gill fungus dead-man's-fingers		<b>dalmatian</b> grape elderberry ffordshire bullterrier currant		<b>squirrel monkey</b> spider monkey titi indri howler monkey
--	--	--	--	--	--	--	---

# I. RÉSEAUX DE NEURONES CONVOLUTIFS

# Une image vue par un ANN



08	02	22	97	38	15	00	48	00	75	04	85	07	78	52	12	50	77	81	62
49	49	99	40	17	81	18	57	60	87	17	40	98	43	69	48	04	56	62	00
81	49	31	73	55	79	14	29	93	71	40	67	53	88	30	03	49	13	36	69
52	70	95	23	04	60	11	42	62	24	88	56	01	32	56	71	37	02	36	91
22	31	16	71	51	62	83	59	41	92	36	54	22	40	40	28	66	33	13	80
24	47	13	60	99	03	45	02	44	75	33	53	78	36	84	20	35	17	12	50
32	98	81	28	64	23	67	10	26	38	40	67	59	54	70	66	18	38	64	70
67	26	20	68	02	62	12	20	95	63	94	39	63	08	40	91	66	49	94	21
24	55	58	05	66	73	99	26	97	17	78	78	96	83	14	88	34	89	63	72
21	36	23	09	75	00	76	44	20	43	35	14	00	61	33	97	34	31	33	95
78	17	53	28	22	75	31	67	15	94	03	80	04	62	16	14	09	53	56	92
16	39	05	42	96	35	31	47	55	58	88	24	00	17	54	24	36	29	85	57
86	56	00	48	35	71	89	07	05	44	44	37	44	60	21	58	51	54	17	58
19	80	81	68	05	94	47	69	28	73	92	13	86	52	17	77	04	89	55	40
04	52	08	83	97	35	99	16	07	97	57	32	16	26	26	79	33	27	98	66
89	36	68	87	57	62	20	72	03	46	33	67	46	55	12	32	63	93	53	69
04	42	16	73	35	35	39	11	24	94	72	18	08	46	29	32	40	62	76	36
20	69	36	41	72	30	23	88	34	69	99	69	82	67	59	85	74	04	36	16
20	73	35	29	78	31	90	01	74	31	49	71	48	36	81	16	23	57	05	54
01	70	54	71	83	51	54	69	16	92	33	48	61	43	52	01	89	29	67	48

What the computer sees

image classification

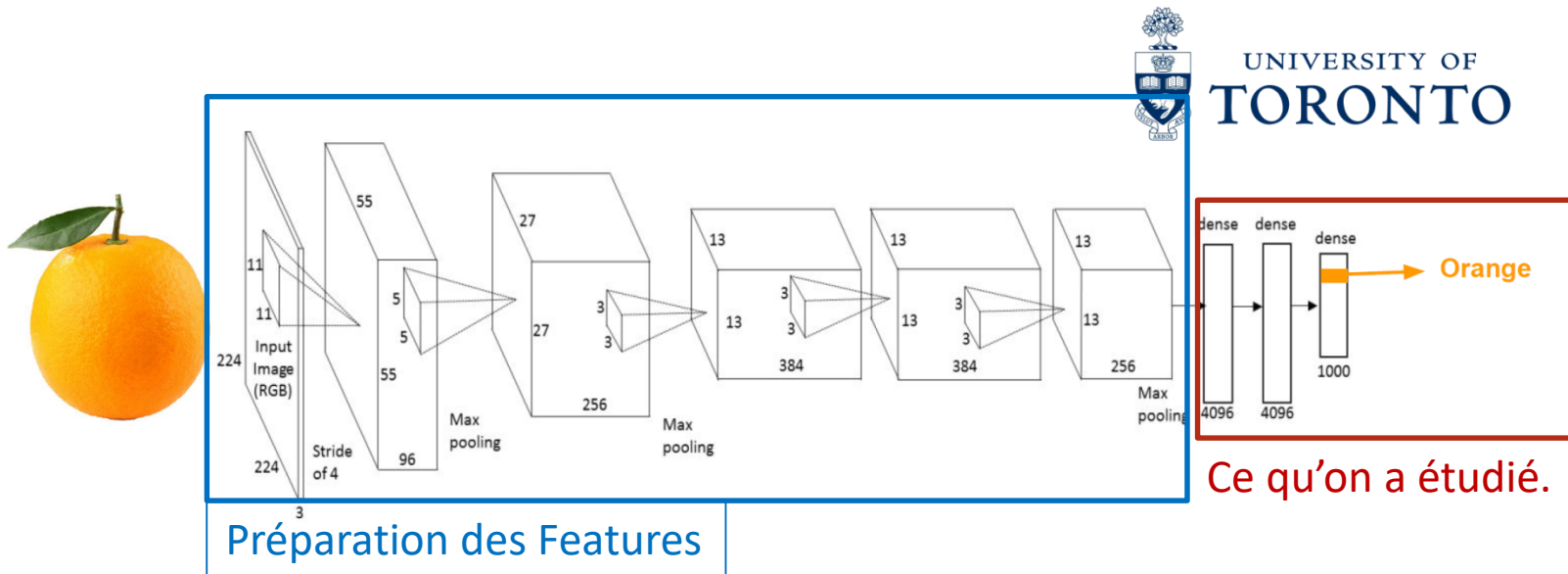
82% cat  
15% dog  
2% hat  
1% mug



# Couches de convolution et de Pooling



- Dans AlexNet, le réseau ANN proprement dit est précédé de deux autres types de couches : **Convolution** et **Pooling**.
- Idée : Apprendre les features.

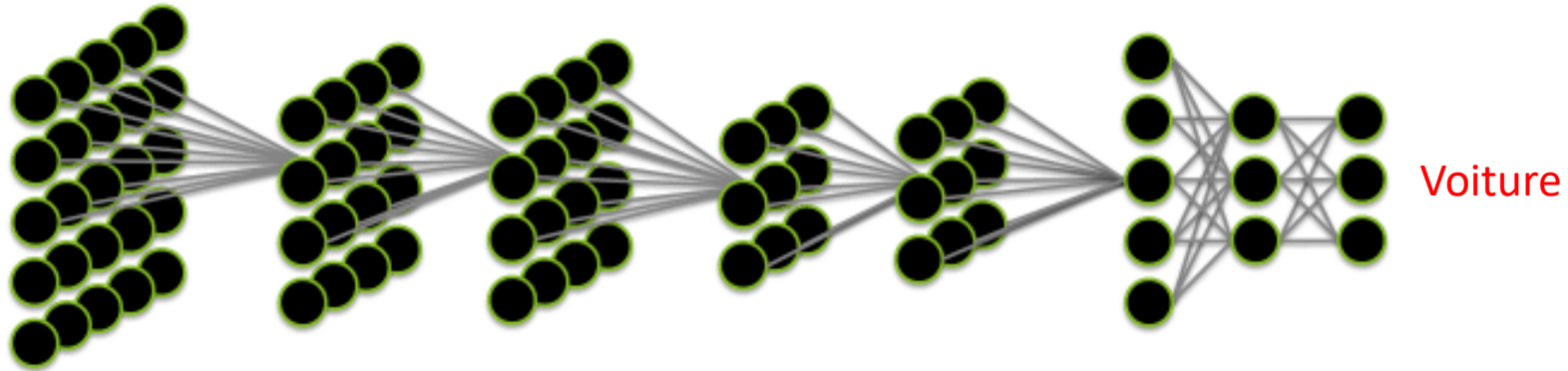
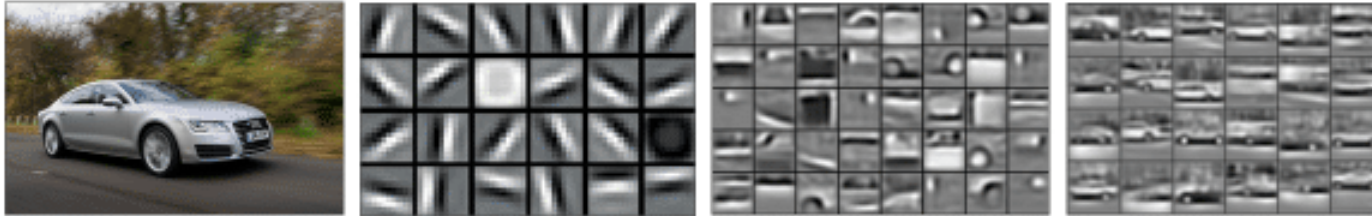


A Krizhevsky, I Sutskever, GE Hinton "[Imagenet classification with deep convolutional neural networks](#)" NIPS 2012

# Préparation des Features



## HOW A DEEP NEURAL NETWORK SEES



- La première partie de ce type de réseaux de neurones fonctionne comme un **extracteur de features**.
- Elle applique plusieurs opérations de filtrage par **convolution**.
- Les paramètres de filtrages sont appris pendant la backpropagation.

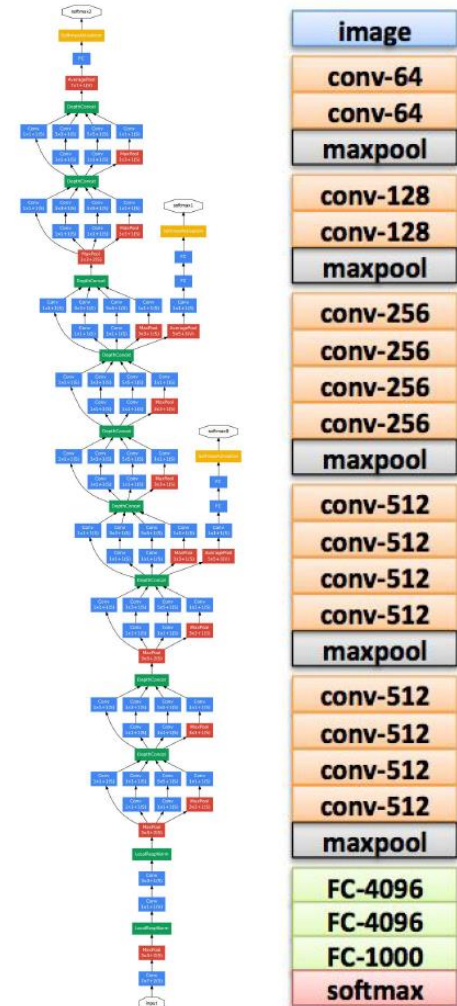
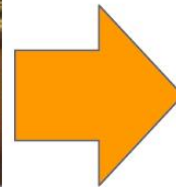
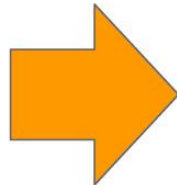


# Augmentation du nombre de couches

- On assiste à une surenchère dans le nombre de couches.

## ImageNet Challenge: 2014

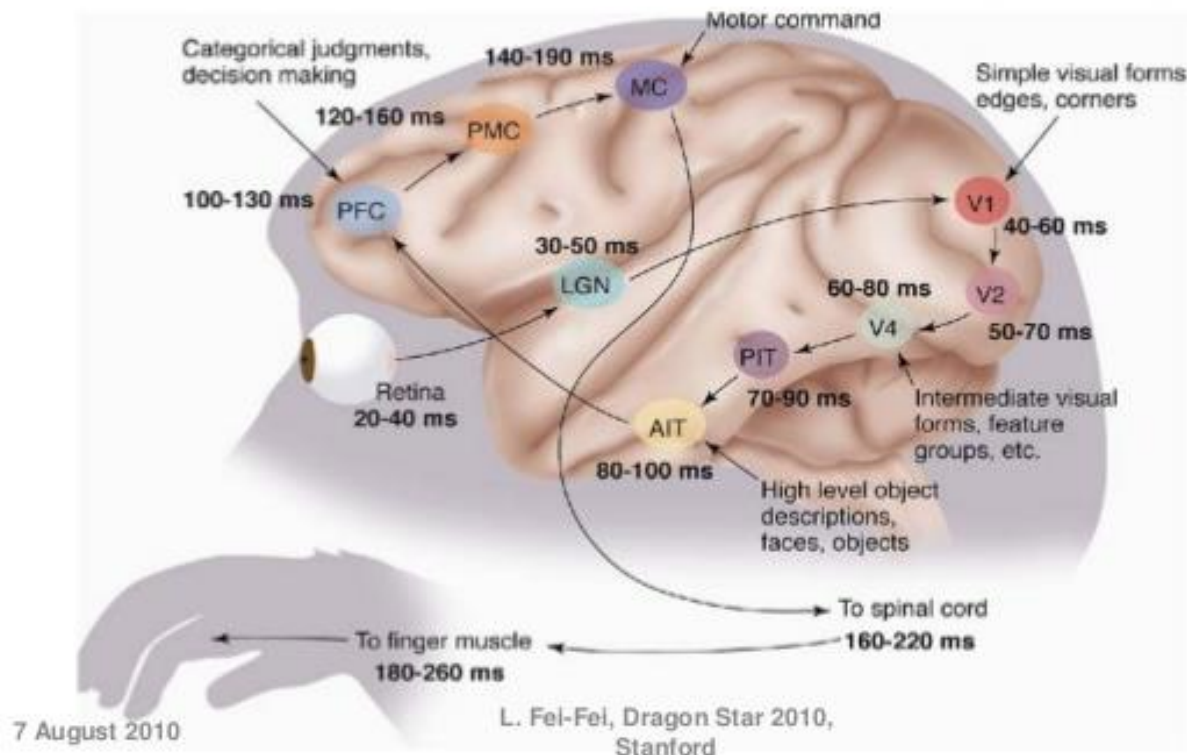
AlexNet



# Encore la métaphore biologique

- Les réseaux convolutifs imitent le fonctionnement de la vision dans le cerveau.

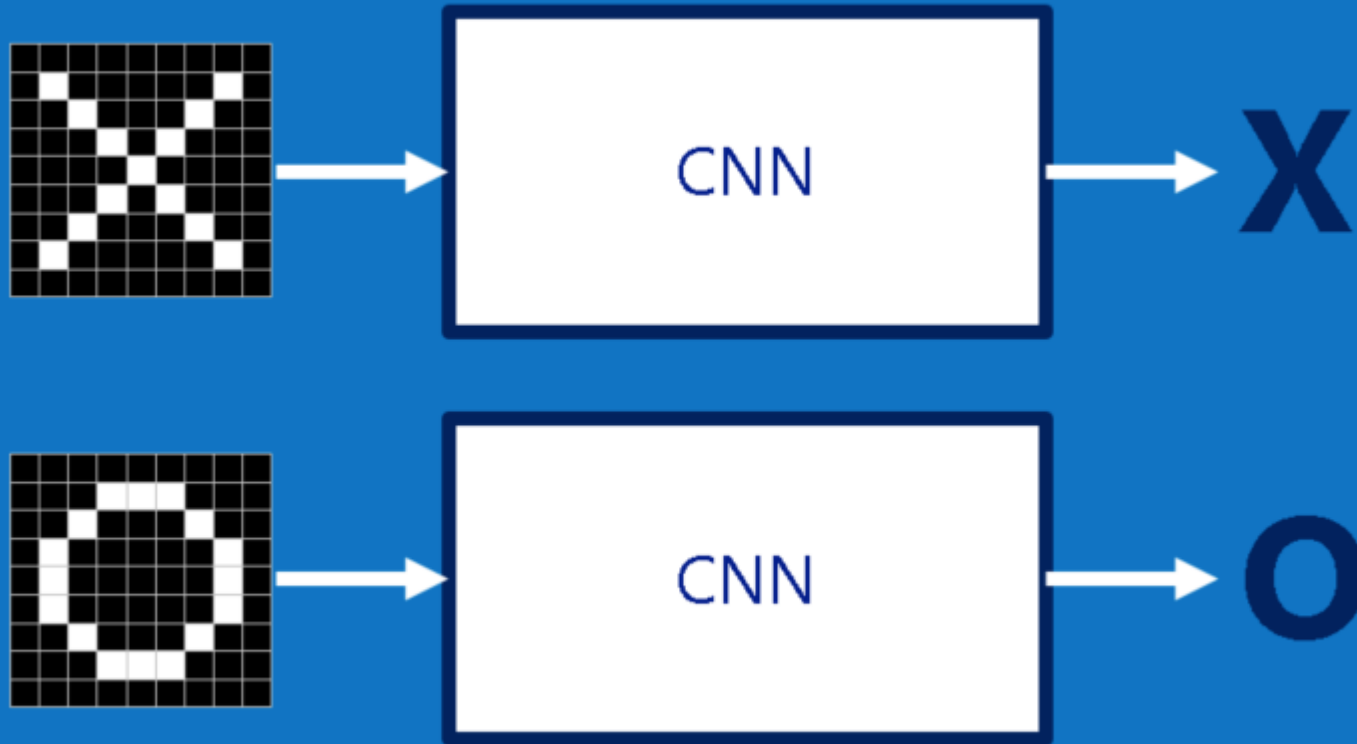
- The ventral (recognition) pathway in the visual cortex
  - Retina → LGN → V1 → V2 → V4 → PIT → AIT (80-100ms)



# Apprentissage des Features

- Les features ne sont pas pré-définies, mais **appries par le modèle** lors la phase d'entraînement
- Les **noyaux de filtre** désignent les poids de la **couche de convolution**. Ils sont initialisés puis mis à jour par rétropropagation du gradient.
- Les CNN sont capables de déterminer tout seul les éléments discriminants d'une image, en s'adaptant au problème posé.
- Par exemple, si la question est de distinguer les chats des chiens, les features appris automatiquement peuvent décrire la forme des oreilles ou des pattes. ...

# Exemple : Distinguer les 'X' et les '0'



- Source : “[How Convolutional Neural Networks Work](#)”.  
**Brandon Rohrer** (Senior Data Scientist à Facebook)

- Problème : Comment reconnaître les images déformées, pivotées, redimensionnées....
- Dans notre exemple, chaque pixel vaut 1 ou -1.

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1



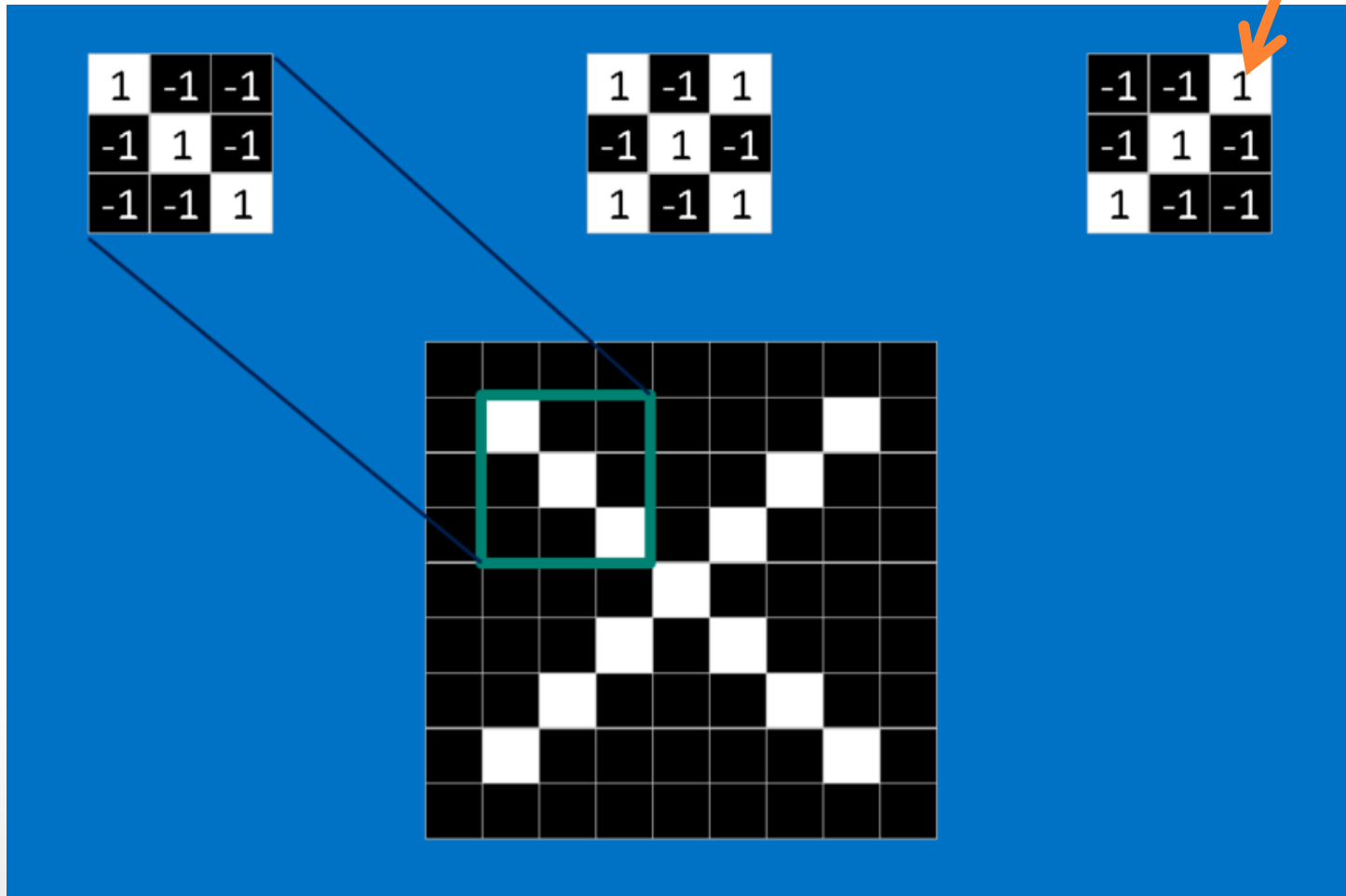
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	1	-1	-1	-1
-1	-1	1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	1	-1	-1
-1	-1	-1	1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1



# Fragments intéressants

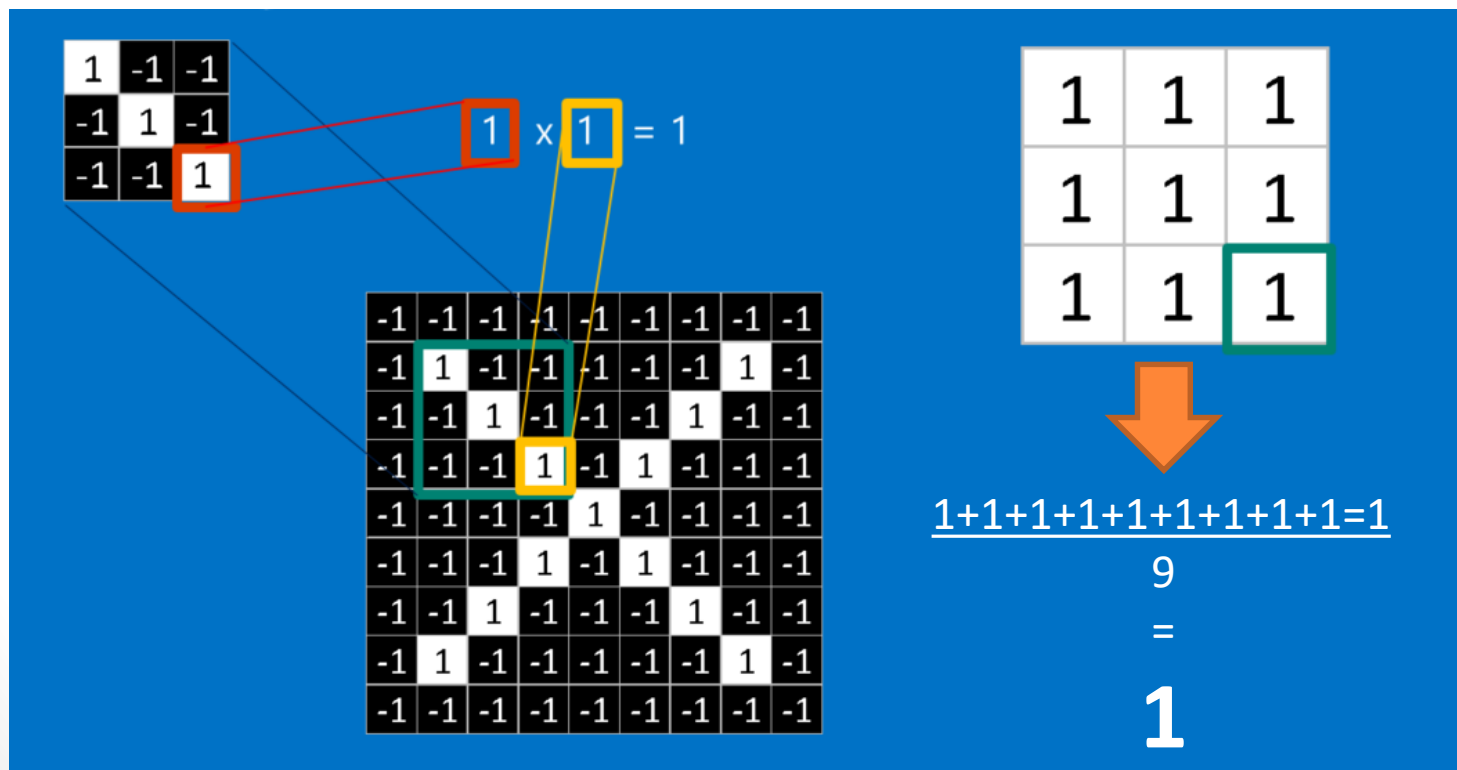
- Le modèle va **apprendre** des fragments utiles pour reconnaître des « X ».

noyaux des filtres



# Opération de Convolution

- On multiplie **pixel par pixel** le filtre et une zone de l'image.
- On obtient, une matrice de la taille de l'image. On en prend la **moyenne**.



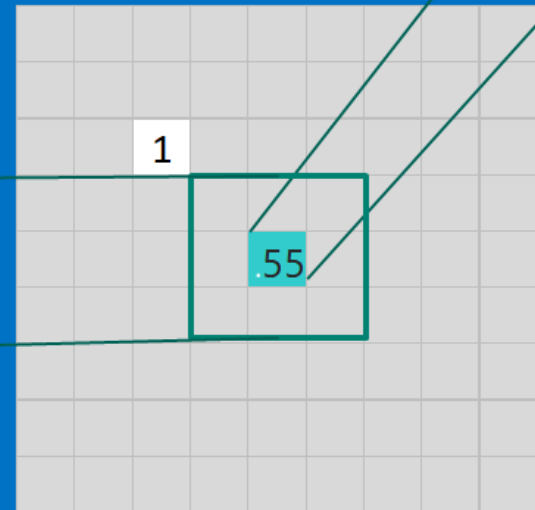
# Opération de Convolution

1	-1	-1
-1	1	-1
-1	-1	1

1	1	-1
1	1	1
-1	1	1

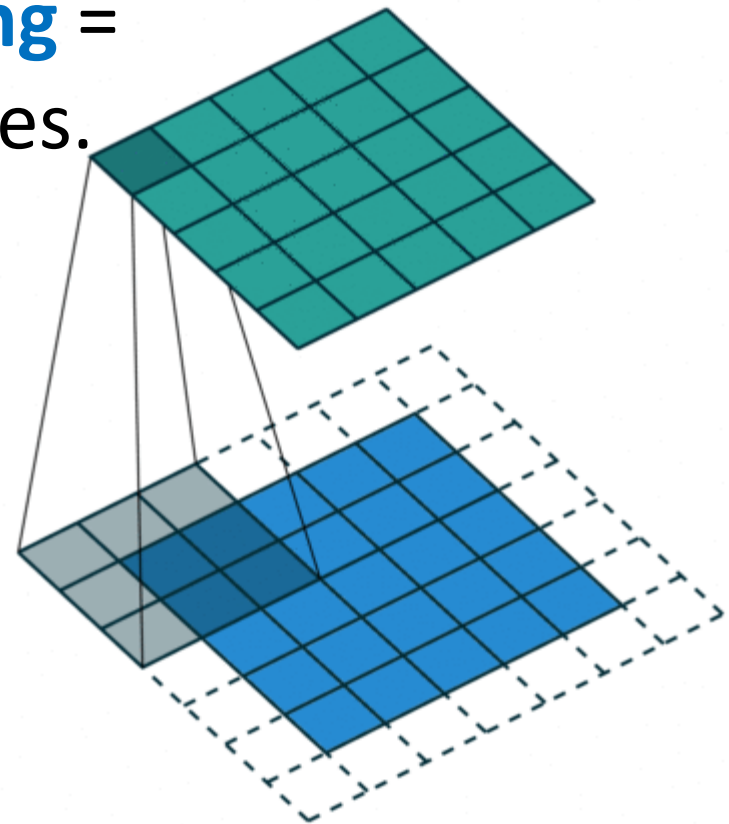
$$\frac{1 + 1 - 1 + 1 + 1 + 1 - 1 + 1 + 1}{9} = .55$$

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1



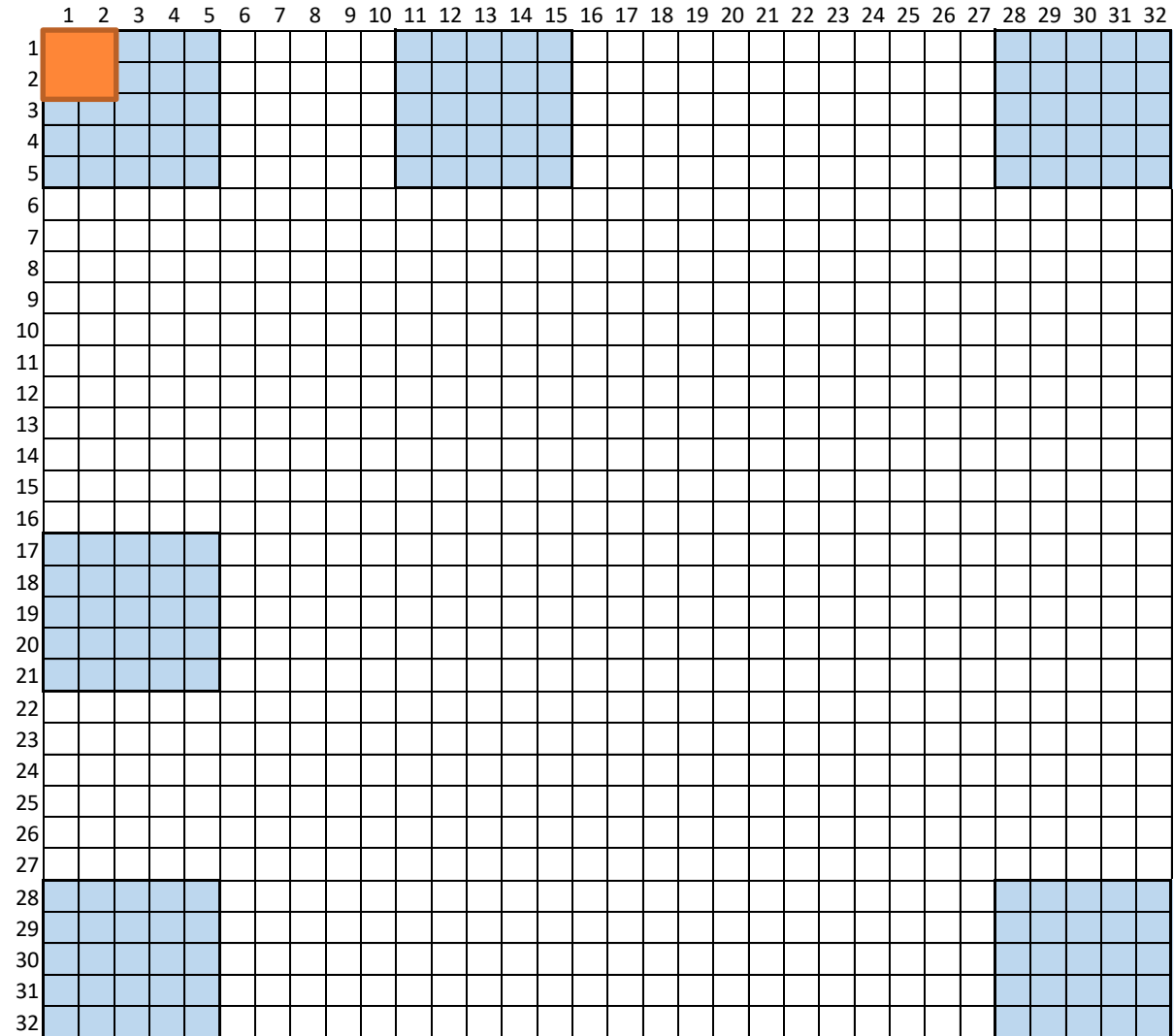
# Convolution – Padding - Stride

- On parcourt l'image d'origine (en bleu) et on crée une « carte » appelé « **feature map** » (vert)
- La taille du filtre (3) et le pas entre chaque étape (**padding** = 1) sont des hyper-paramètres.
- Le **stride** est le pas entre chaque projection (1 en général)
- Les deux grilles n'ont pas forcément la même taille.



# Dimension de sortie

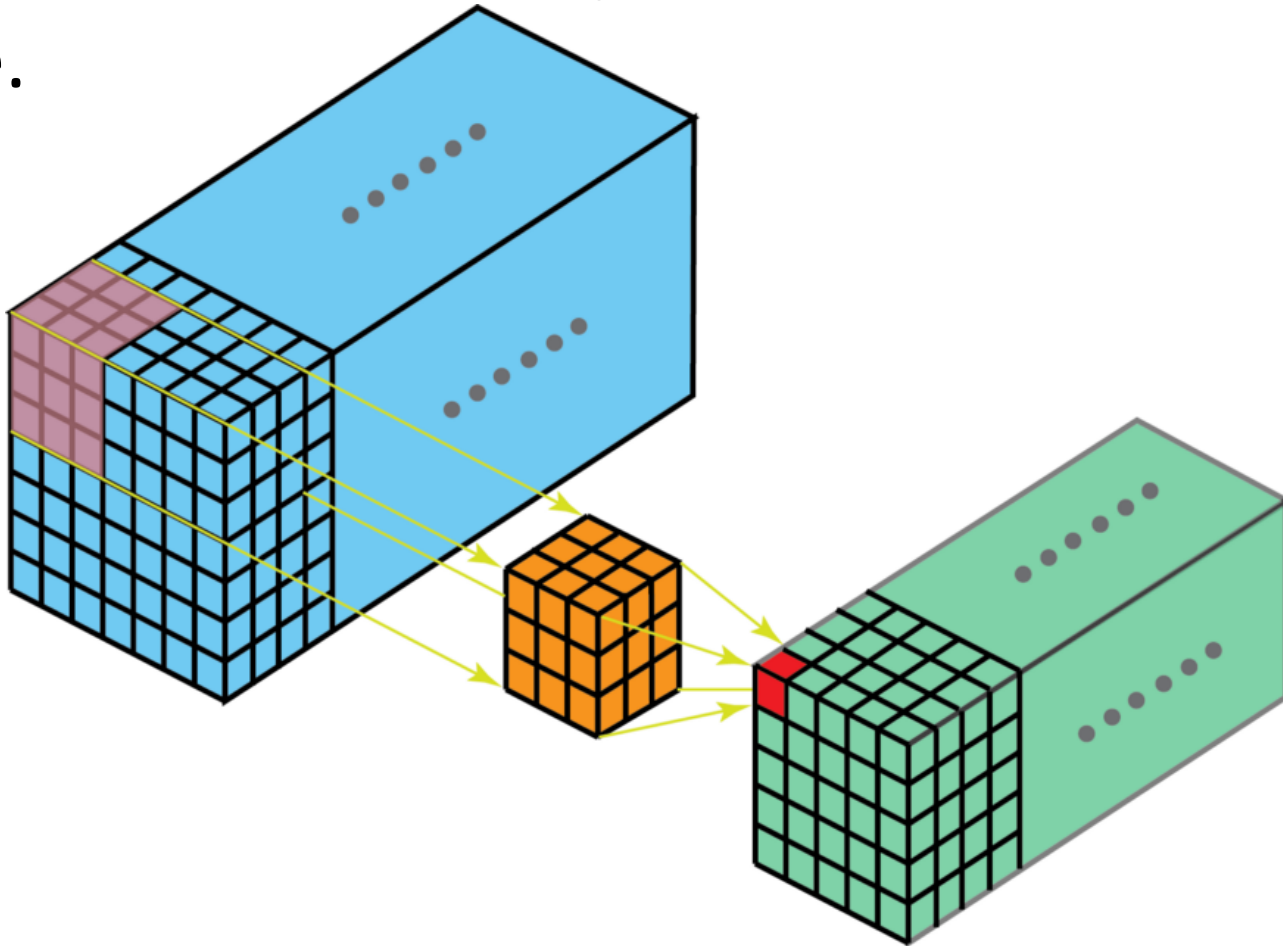
- Sans padding, la grille de résultat est plus petite que celle d'origine.
- Ici on voit que la grille 32 x 32 avec un filtre 5 x 5 donne une grille 28 x 28.





# Convolution 3D

- Le noyau de filtre peut être en 3D. On change alors éventuellement la profondeur de la couche.



# Convolution pour un filtre

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1



1	-1	-1
-1	1	-1
-1	-1	1

=

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

# Couche de Convolution

- On applique tous les filtres :

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1



1	-1	-1
-1	1	-1
-1	-1	1

=

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1



1	-1	1
-1	1	-1
1	-1	1

=

0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.11	0.33	-0.77	1.00	-0.77	0.33	-0.11
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1



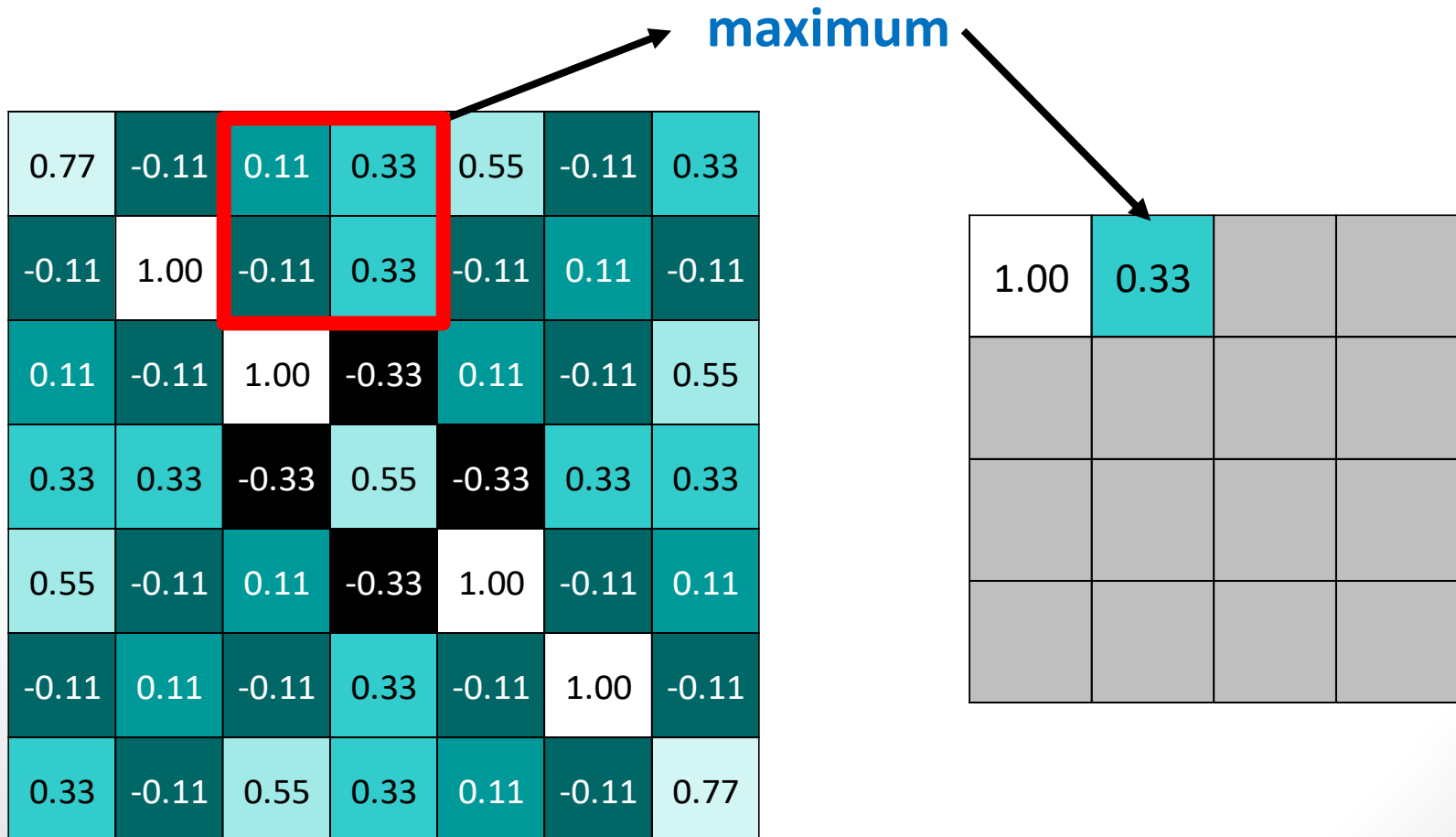
-1	-1	1
-1	1	-1
1	-1	-1

=

0.33	-0.11	0.55	0.33	0.11	-0.11	0.77
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33

# Pooling

- Pour réduire les features map, on va appliquer une opération de **pooling**.
- On utilise généralement la valeur max. La somme est possible.



# Pooling d'une feature map

- on utilise souvent une fenêtre de 2 ou 3 pixels de côté et une valeur de 2 pixels le pas.

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

max  
pooling



1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77



# Couche de Pooling

- Le Pooling est appliqué à chaque « feature map »:

0.77	-0.11	0.11	0.33	0.33	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.33
0.33	0.33	-0.33	0.33	-0.33	0.33	0.33
0.33	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.33	0.33	0.11	-0.11	0.77

0.33	-0.33	0.11	-0.11	0.11	-0.33	0.33
-0.33	0.33	-0.33	0.33	-0.33	0.33	-0.33
0.11	-0.33	0.33	-0.77	0.33	-0.33	0.11
-0.11	0.33	-0.77	1.00	-0.77	0.33	-0.11
0.11	-0.33	0.33	-0.77	0.33	-0.33	0.11
-0.33	0.33	-0.33	0.33	-0.33	0.33	-0.33
0.33	-0.33	0.11	-0.11	0.11	-0.33	0.33

0.33	-0.11	0.33	0.33	0.11	-0.11	0.77
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.11	-0.33	1.00	-0.11	0.11
0.33	0.33	-0.33	0.33	-0.33	0.33	0.33
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.77	-0.11	0.11	0.33	0.33	-0.11	0.33



1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

0.55	0.33	0.55	0.33
0.33	1.00	0.55	0.11
0.55	0.55	0.55	0.11
0.33	0.11	0.11	0.33

0.33	0.55	1.00	0.77
0.55	0.55	1.00	0.33
1.00	1.00	0.11	0.55
0.77	0.33	0.55	0.33

# Couche RELU

- Les valeurs négatives sont passées à 0

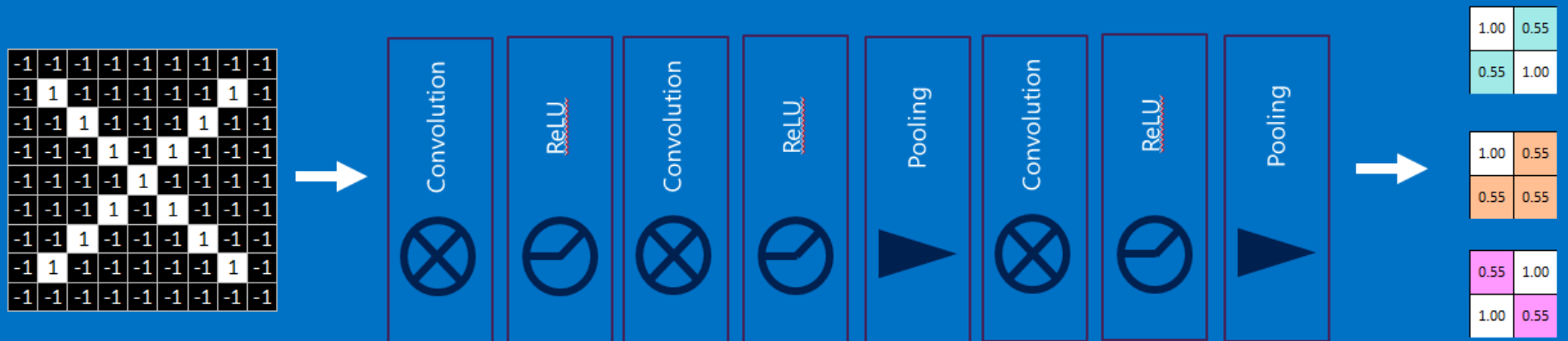
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77



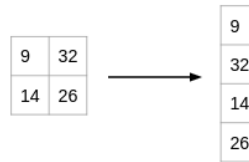
0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77

# Combinaison des couches

- On peut combiner à l'infini les 3 couches.

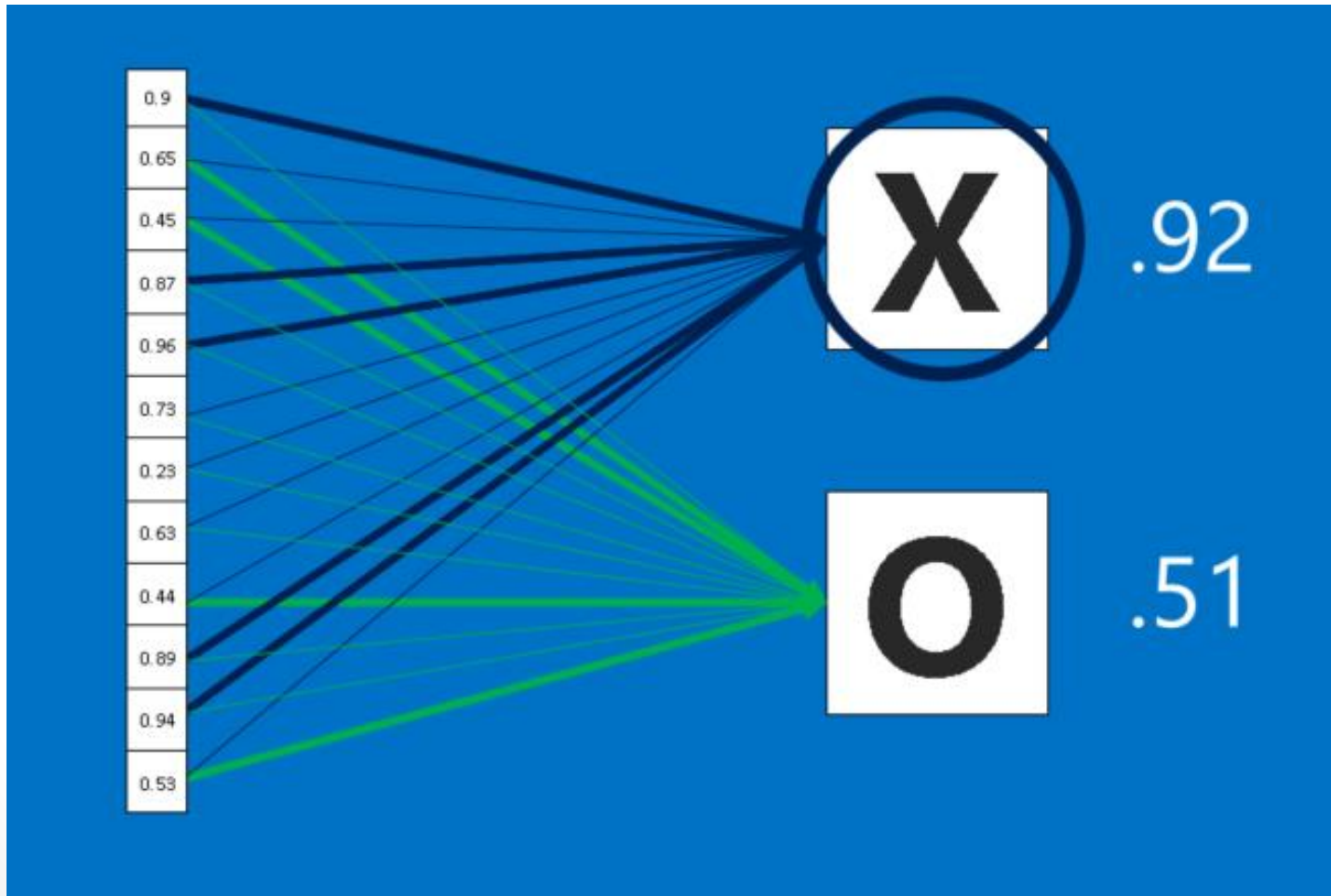


Déplieusement à la fin :



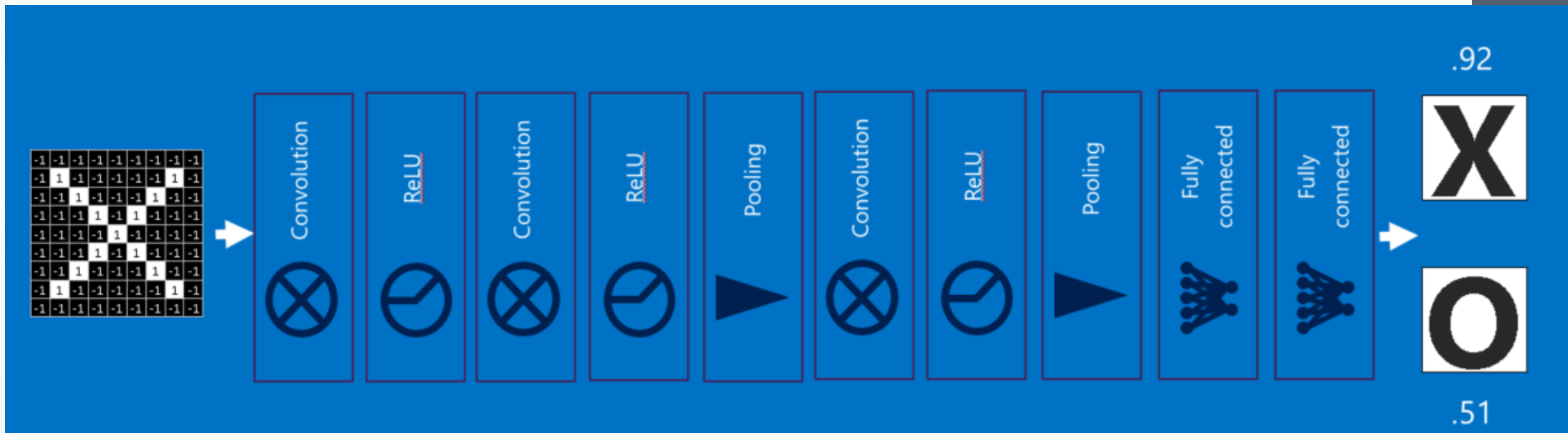
[ 26 ]

# Dépliement et couche dense



# Résultat Final

- Les couches de convolutions sont complétées par des couches ANN « **fully connected** »:





<https://www.youtube.com/watch?v=f0t-OCG79-U>

```

import torch
import torch.nn as nn

class BasicCNNModel(nn.Module):
    def __init__(self, inp_dim=(10, 10), outp_dim=(10, 10)):
        super(BasicCNNModel, self).__init__()

        CONV_IN = 3
        KERNEL_SIZE = 3
        CONV_OUT_1 = 50
        CONV_OUT_2 = 100
        DENSE_IN = CONV_OUT_2

        self.relu = nn.ReLU()
        self.conv2d_1 = nn.Conv2d(CONV_IN, CONV_OUT_1, kernel_size=KERNEL_SIZE)
        self.conv2d_2 = nn.Conv2d(CONV_OUT_1, CONV_OUT_2, kernel_size=KERNEL_SIZE)
        self.dense_1 = nn.Linear(DENSE_IN, outp_dim[0]*outp_dim[1]*10)
        self.softmax = nn.Softmax(dim=1)

    def forward(self, x, outp_dim):
        # ... preparation de x

        # Convolution 1
        self.conv2d_1.in_features = x.shape[1]
        conv_1_out = self.relu(self.conv2d_1(x))

        # Convolution 2
        self.conv2d_2.in_features = conv_1_out.shape[1]
        conv_2_out = self.relu(self.conv2d_2(conv_1_out))

        # max Pooling
        self.dense_1.out_features = outp_dim
        feature_vector, _ = torch.max(conv_2_out, 2)
        feature_vector, _ = torch.max(feature_vector, 2)

        # Couche finale et softmax
        logit_outputs = self.dense_1(feature_vector)
        return self.softmax(logit_outputs)

```