

RÉSEAUX DE NEURONES CONVOLUTIFS (CNNs)

Jérémie Cabessa

Laboratoire DAVID, UVSQ

RÉSEAUX DE NEURONES CONVOLUTIFS (CNNs)

- ▶ **Les réseaux de neurones convolutifs (CNNs) sont un type de réseaux de neurones particulièrement adaptés pour le traitement des images et des vidéos (computer vision).**
- ▶ Ils possèdent toutefois d'autres types d'applications: systèmes de recommandation, le traitement du langage naturel, etc.
- ▶ Leur fonctionnement est inspiré de processus biologiques: le cortex visuel des animaux.

RÉSEAUX DE NEURONES CONVOLUTIFS (CNNs)

- ▶ Les réseaux de neurones convolutifs (CNNs) sont un type de réseaux de neurones particulièrement adaptés pour le traitement des images et des vidéos (computer vision).
- ▶ Ils possèdent toutefois d'autres types d'applications: systèmes de recommandation, le traitement du langage naturel, etc.
- ▶ Leur fonctionnement est inspiré de processus biologiques: le cortex visuel des animaux.

RÉSEAUX DE NEURONES CONVOLUTIFS (CNNs)

- ▶ Les réseaux de neurones convolutifs (CNNs) sont un type de réseaux de neurones particulièrement adaptés pour le traitement des images et des vidéos (computer vision).
- ▶ Ils possèdent toutefois d'autres types d'applications: systèmes de recommandation, le traitement du langage naturel, etc.
- ▶ Leur fonctionnement est inspiré de processus biologiques: le cortex visuel des animaux.

RÉSEAUX DE NEURONES CONVOLUTIFS (CNNs)

► Articles fondateurs sur les CNNs:

- [Fukushima, 1980, Fukushima, 1988]
- [LeCun et al., 1998]

► Articles qui ont contribués à de grandes avancées dans le domaine des CNNs:

- Krizhevsky, Sutskever, Hinton (2012) - ImageNet Classification with Deep Convolutional Neural Networks
- Simonyan, Zisserman (2014) - Very Deep Convolutional Networks for Large-Scale Image Recognition
- He, Zhang, Ren, Sun (2015) - Deep Residual Learning for Image Recognition
- Szegedy, Liu, Jia, Sermanet, Reed, Anguelov, Erhan, Vanhoucke, Rabinovich (2015) - Going Deeper with Convolutions
- He, Zhang, Ren, Sun (2016) - Identity Mappings in Deep Residual Networks

► Par la suite, le domaine a explosé...

RÉSEAUX DE NEURONES CONVOLUTIFS (CNNs)

- ▶ Articles fondateurs sur les CNNs:
 - [Fukushima, 1980, Fukushima, 1988]
 - [LeCun et al., 1998]
- ▶ Articles qui ont contribués à de grandes avancées dans le domaine des CNNs:
 - [Lecun et al., 1998]
 - [Krizhevsky et al., 2012]
 - [Simonyan et al., 2014]
 - [He et al., 2016]
 - [Huang et al., 2017]
 - [Xie et al., 2017]
 - [Zhang et al., 2017]
 - [Xie et al., 2018]
 - [Xie et al., 2019]
 - [Xie et al., 2020]
 - [Xie et al., 2021]
 - [Xie et al., 2022]
- ▶ Par la suite, le domaine a explosé...

RÉSEAUX DE NEURONES CONVOLUTIFS (CNNs)

- ▶ Articles fondateurs sur les CNNs:
 - [\[Fukushima, 1980, Fukushima, 1988\]](#)
 - [\[LeCun et al., 1998\]](#)
- ▶ Articles qui ont contribués à de grandes avancées dans le domaine des CNNs:
 - [AlexNet: \[Krizhevsky et al., 2012\]](#)
 - [\[Simonyan et al., 2014\]](#)
 - [\[He et al., 2016\]](#)
 - [\[Huang et al., 2017\]](#)
- ▶ Par la suite, le domaine a explosé...

RÉSEAUX DE NEURONES CONVOLUTIFS (CNNs)

- ▶ Articles fondateurs sur les CNNs:
 - [\[Fukushima, 1980, Fukushima, 1988\]](#)
 - [\[LeCun et al., 1998\]](#)
- ▶ Articles qui ont contribués à de grandes avancées dans le domaine des CNNs:
 - AlexNet: [\[Krizhevsky et al., 2012\]](#)
 - U-Net: [\[Ronneberger et al., 2015\]](#)
 - VGGNet: [\[Simonyan and Zisserman, 2015\]](#)
 - ResNet: [\[He et al., 2016\]](#)
- ▶ Par la suite, le domaine a explosé...

RÉSEAUX DE NEURONES CONVOLUTIFS (CNNs)

- ▶ Articles fondateurs sur les CNNs:
 - [\[Fukushima, 1980, Fukushima, 1988\]](#)
 - [\[LeCun et al., 1998\]](#)
- ▶ Articles qui ont contribués à de grandes avancées dans le domaine des CNNs:
 - AlexNet: [\[Krizhevsky et al., 2012\]](#)
 - U-Net: [\[Ronneberger et al., 2015\]](#)
 - VGGNet: [\[Simonyan and Zisserman, 2015\]](#)
 - ResNet: [\[He et al., 2016\]](#)
- ▶ Par la suite, le domaine a explosé...

RÉSEAUX DE NEURONES CONVOLUTIFS (CNNs)

- ▶ Articles fondateurs sur les CNNs:
 - [\[Fukushima, 1980, Fukushima, 1988\]](#)
 - [\[LeCun et al., 1998\]](#)
- ▶ Articles qui ont contribués à de grandes avancées dans le domaine des CNNs:
 - AlexNet: [\[Krizhevsky et al., 2012\]](#)
 - U-Net: [\[Ronneberger et al., 2015\]](#)
 - VGGNet: [\[Simonyan and Zisserman, 2015\]](#)
 - ResNet: [\[He et al., 2016\]](#)
- ▶ Par la suite, le domaine a explosé...

RÉSEAUX DE NEURONES CONVOLUTIFS (CNNs)

- ▶ Articles fondateurs sur les CNNs:
 - [\[Fukushima, 1980, Fukushima, 1988\]](#)
 - [\[LeCun et al., 1998\]](#)
- ▶ Articles qui ont contribués à de grandes avancées dans le domaine des CNNs:
 - AlexNet: [\[Krizhevsky et al., 2012\]](#)
 - U-Net: [\[Ronneberger et al., 2015\]](#)
 - VGGNet: [\[Simonyan and Zisserman, 2015\]](#)
 - ResNet: [\[He et al., 2016\]](#)
- ▶ Par la suite, le domaine a explosé...

RÉSEAUX DE NEURONES CONVOLUTIFS (CNNs)

- ▶ Articles fondateurs sur les CNNs:
 - [\[Fukushima, 1980, Fukushima, 1988\]](#)
 - [\[LeCun et al., 1998\]](#)
- ▶ Articles qui ont contribués à de grandes avancées dans le domaine des CNNs:
 - AlexNet: [\[Krizhevsky et al., 2012\]](#)
 - U-Net: [\[Ronneberger et al., 2015\]](#)
 - VGGNet: [\[Simonyan and Zisserman, 2015\]](#)
 - ResNet: [\[He et al., 2016\]](#)
- ▶ Par la suite, le domaine a explosé...

RÉSEAUX DE NEURONES CONVOLUTIFS (CNNs)

- ▶ Articles fondateurs sur les CNNs:
 - [\[Fukushima, 1980, Fukushima, 1988\]](#)
 - [\[LeCun et al., 1998\]](#)
- ▶ Articles qui ont contribués à de grandes avancées dans le domaine des CNNs:
 - AlexNet: [\[Krizhevsky et al., 2012\]](#)
 - U-Net: [\[Ronneberger et al., 2015\]](#)
 - VGGNet: [\[Simonyan and Zisserman, 2015\]](#)
 - ResNet: [\[He et al., 2016\]](#)
- ▶ Par la suite, le domaine a explosé...

CONVOLUTIONS

- ▶ Les signaux de grande dimension (images, vidéos, audios) posent problème lorsqu'ils sont traités comme des vecteurs non structurés (flattened).
- ▶ En effet, une couche linéaire prenant en entrée une image RGB de 256×256 pixels, et produisant en sortie une image de même taille nécessite $(256 \cdot 256 \cdot 3)^2 \simeq 3.87 \times 10^{10}$ paramètres.
- ▶ Empreinte mémoire correspondante de ~ 150 Go...
- ▶ Une telle exigence est non seulement inefficace mais également incohérente.

CONVOLUTIONS

- ▶ Les signaux de grande dimension (images, vidéos, audios) posent problème lorsqu'ils sont traités comme des vecteurs non structurés (flattened).
- ▶ En effet, une couche linéaire prenant en entrée une image RGB de 256×256 pixels, et produisant en sortie une image de même taille nécessite $(256 \cdot 256 \cdot 3)^2 \simeq 3.87 \times 10^{10}$ paramètres.
- ▶ Empreinte mémoire correspondante de ~ 150 Go...
- ▶ Une telle exigence est non seulement inefficace mais également incohérente.

CONVOLUTIONS

- ▶ Les signaux de grande dimension (images, vidéos, audios) posent problème lorsqu'ils sont traités comme des vecteurs non structurés (flattened).
- ▶ En effet, une couche linéaire prenant en entrée une image RGB de 256×256 pixels, et produisant en sortie une image de même taille nécessite $(256 \cdot 256 \cdot 3)^2 \simeq 3.87 \times 10^{10}$ paramètres.
- ▶ Empreinte mémoire correspondante de ~ 150 Go...
- ▶ Une telle exigence est non seulement inefficace mais également incohérente.

CONVOLUTIONS

- ▶ Les signaux de grande dimension (images, vidéos, audios) posent problème lorsqu'ils sont traités comme des vecteurs non structurés (flattened).
- ▶ En effet, une couche linéaire prenant en entrée une image RGB de 256×256 pixels, et produisant en sortie une image de même taille nécessite $(256 \cdot 256 \cdot 3)^2 \simeq 3.87 \times 10^{10}$ paramètres.
- ▶ Empreinte mémoire correspondante de ~ 150 Go...
- ▶ Une telle exigence est non seulement inefficace mais également incohérente.

RÉSUMÉ : CONVOLUTIONS ET ÉQUIVARIANCE

- ▶ En effet, un motif dans une image conserve sa signification, quelle que soit sa position dans l'image.
 - ▶ Ainsi, une transformation, qui agit comme un détecteur de motif (sorte de sonde), doit être appliquée localement, puis translatée partout dans l'image.
 - ▶ C'est le concept d'*équivariance par translation*.
 - ▶ Une couche de convolution incarne précisément cette idée.

RÉSUMÉ : CONVOLUTIONS ET ÉQUIVARIANCE

- ▶ En effet, un motif dans une image conserve sa signification, quelle que soit sa position dans l'image.
 - ▶ Ainsi, une transformation, qui agit comme un détecteur de motif (sorte de sonde), doit être appliquée localement, puis translatée partout dans l'image.
 - ▶ C'est le concept d'*équivariance par translation*.
 - ▶ Une couche de convolution incarne précisément cette idée.

RÉSUMÉ : CONVOLUTIONS ET ÉQUIVARIANCE

- ▶ En effet, un motif dans une image conserve sa signification, quelle que soit sa position dans l'image.
 - ▶ Ainsi, une transformation, qui agit comme un détecteur de motif (sorte de sonde), doit être appliquée localement, puis translatée partout dans l'image.
 - ▶ C'est le concept d'*équivariance par translation*.
 - ▶ Une couche de convolution incarne précisément cette idée.

RÉSUMÉ : CONVOLUTIONS ET ÉQUIVARIANCE

- ▶ En effet, un motif dans une image conserve sa signification, quelle que soit sa position dans l'image.
 - ▶ Ainsi, une transformation, qui agit comme un détecteur de motif (sorte de sonde), doit être appliquée localement, puis translatée partout dans l'image.
 - ▶ C'est le concept d'*équivariance par translation*.
 - ▶ Une couche de convolution incarne précisément cette idée.

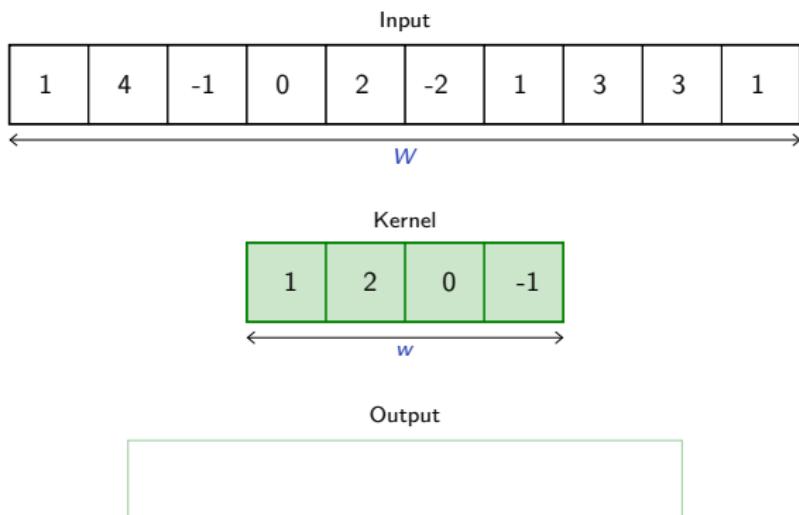
CONVOLUTION: 1D

Input

1	4	-1	0	2	-2	1	3	3	1
---	---	----	---	---	----	---	---	---	---

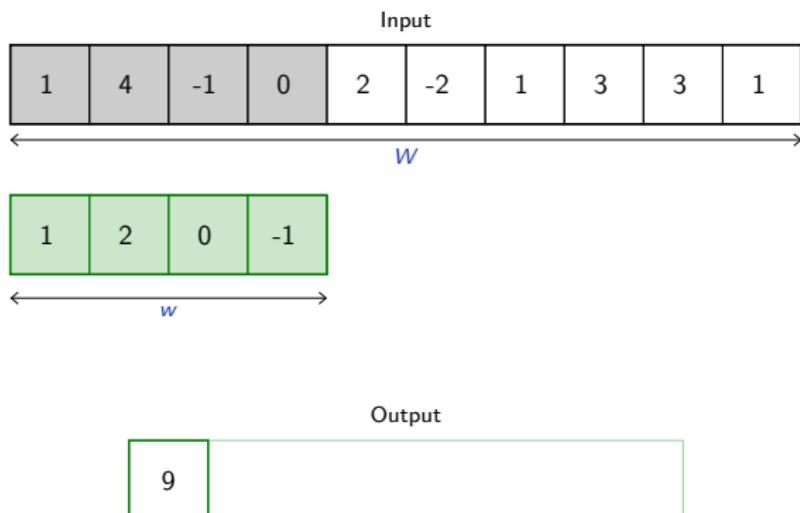
Figures taken from [\[Fleuret, 2022\]](#).

CONVOLUTION: 1D



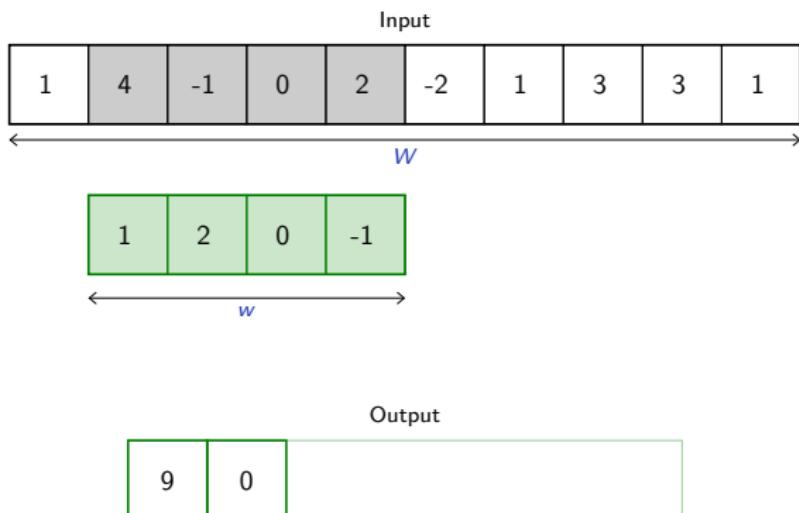
Figures taken from [Fleuret, 2022].

CONVOLUTION: 1D



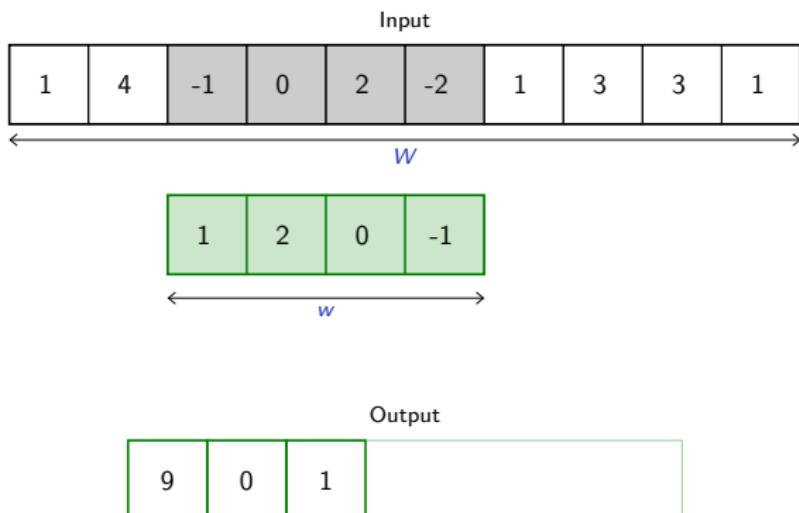
Figures taken from [\[Fleuret, 2022\]](#).

CONVOLUTION: 1D



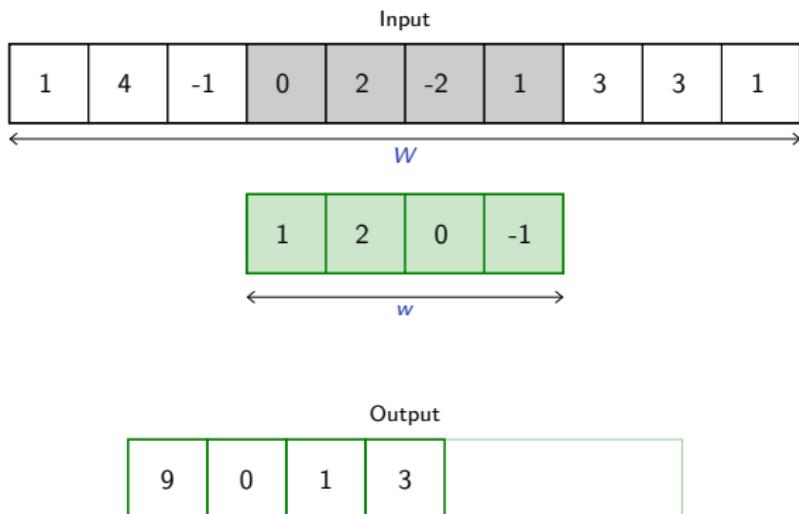
Figures taken from [\[Fleuret, 2022\]](#).

CONVOLUTION: 1D



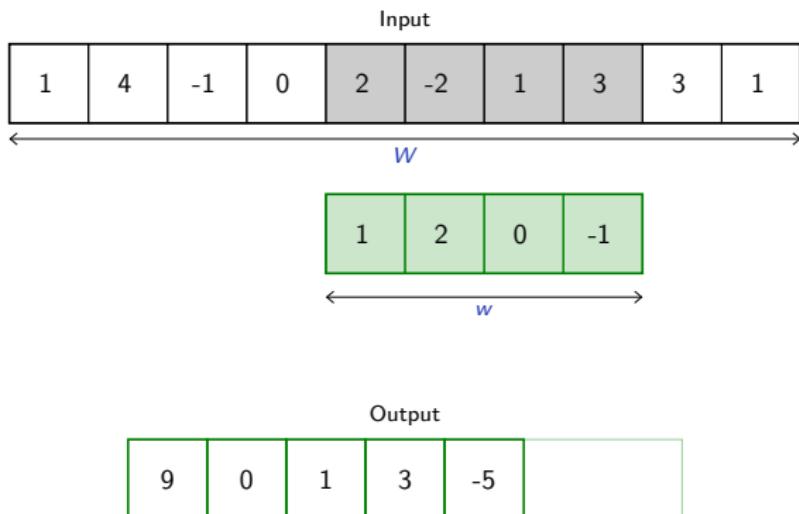
Figures taken from [Fleuret, 2022].

CONVOLUTION: 1D



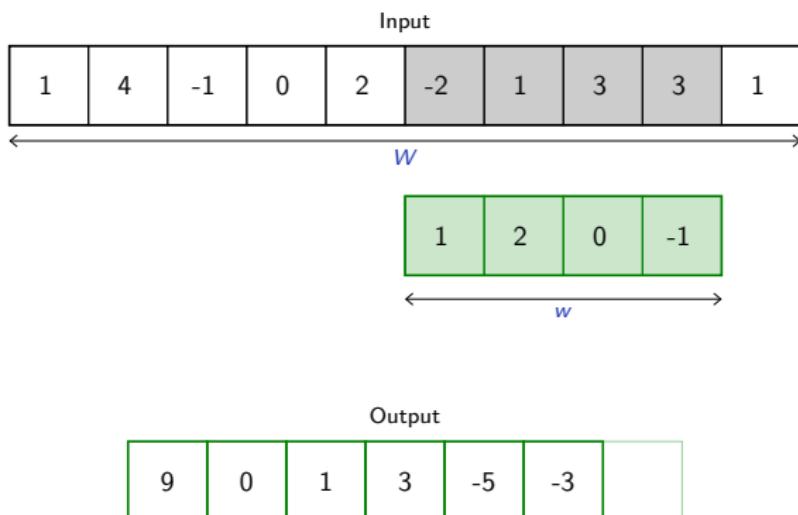
Figures taken from [\[Fleuret, 2022\]](#).

CONVOLUTION: 1D



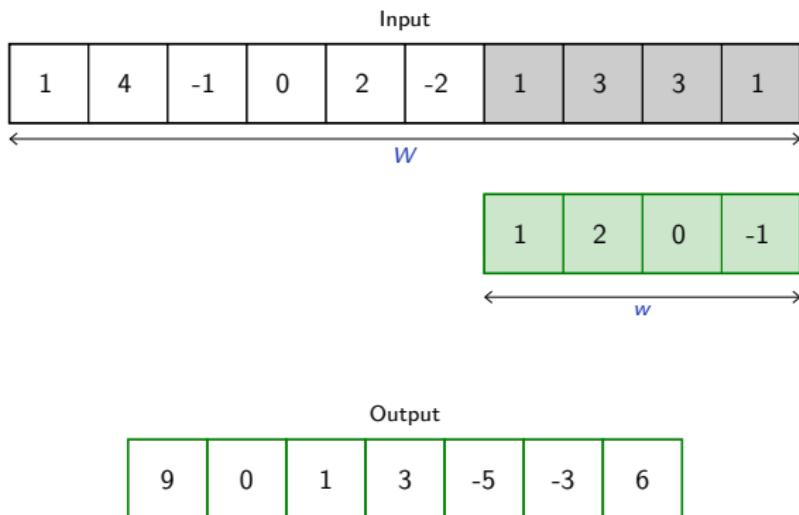
Figures taken from [\[Fleuret, 2022\]](#).

CONVOLUTION: 1D



Figures taken from [\[Fleuret, 2022\]](#).

CONVOLUTION: 1D



Figures taken from [\[Fleuret, 2022\]](#).

CONVOLUTION: 1D

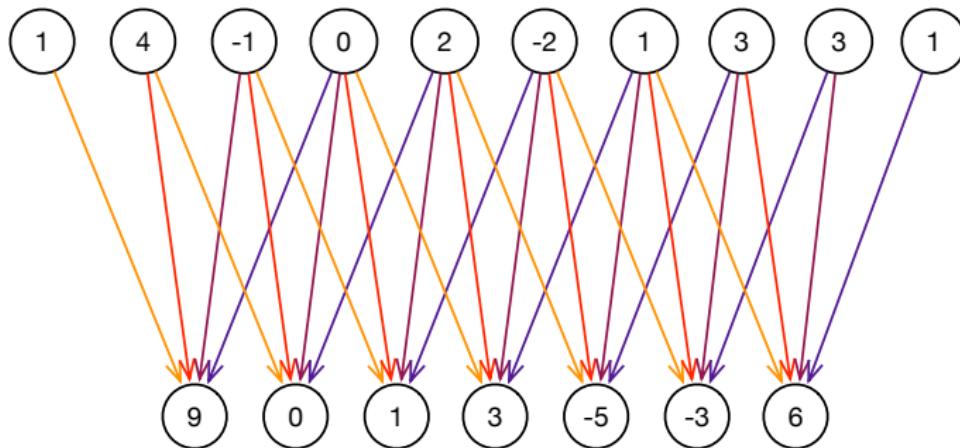
Diagram illustrating a convolution operation with a stride of 2. The input is a 10-element vector [1, 4, -1, 0, 2, -2, 1, 3, 3, 1]. A 4-element kernel [1, 2, 0, -1] is applied to it. The output is a 5-element vector [9, 0, 1, 3, -5, -3, 6].

Figures taken from [\[Fleuret, 2022\]](#).

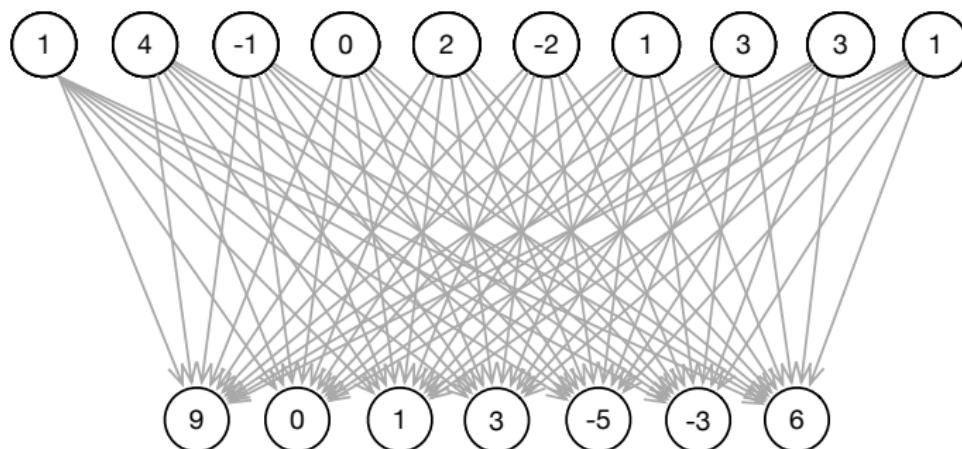
CONVOLUTION: 1D



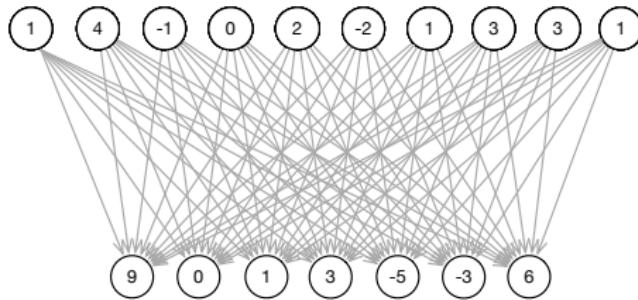
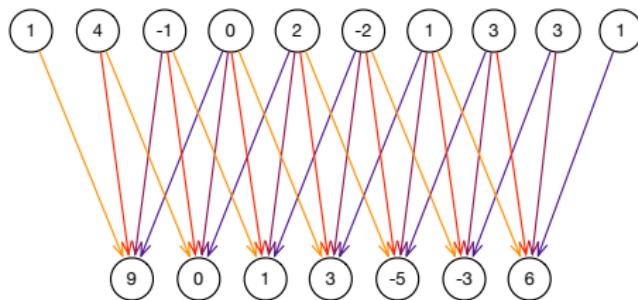
CONVOLUTION: 1D



CONVOLUTION: 1D



CONVOLUTION: 1D



CONVOLUTION: 1D

- ▶ Étant donné un vecteur d'*input* x de taille W

$$\mathbf{x} = (x_1, \dots, x_W)$$

et un *noyau* ou *filtre* (*kernel*, *filter*) \mathbf{k} de taille w ,

$$\mathbf{k} = (k_1, \dots, k_w)$$

la **convolution** $\mathbf{x} * \mathbf{k}$ est un vecteur de taille $W - w + 1$ dont les composantes sont données par

$$(\mathbf{x} * \mathbf{k})_i = \mathbf{x}_{[i:i+w-1]} \cdot \mathbf{k} = \sum_{j=1}^w x_{i-1+j} \cdot k_j$$

CONVOLUTION: 1D

- ▶ Étant donné un vecteur d'*input* x de taille W

$$\mathbf{x} = (x_1, \dots, x_W)$$

et un *noyau* ou *filtre* (*kernel*, *filter*) \mathbf{k} de taille w ,

$$\mathbf{k} = (k_1, \dots, k_w)$$

la **convolution** $\mathbf{x} * \mathbf{k}$ est un vecteur de taille $W - w + 1$ dont les composantes sont données par

$$(\mathbf{x} * \mathbf{k})_i = \mathbf{x}_{[i:i+w-1]} \cdot \mathbf{k} = \sum_{j=1}^w x_{i-1+j} \cdot k_j$$

CONVOLUTION: 1D

- ▶ Étant donné un vecteur d'*input* x de taille W

$$\mathbf{x} = (x_1, \dots, x_W)$$

et un *noyau* ou *filtre* (*kernel*, *filter*) \mathbf{k} de taille w ,

$$\mathbf{k} = (k_1, \dots, k_w)$$

la convolution $\mathbf{x} * \mathbf{k}$ est un vecteur de taille $W - w + 1$ dont les composantes sont données par

$$(\mathbf{x} * \mathbf{k})_i = \mathbf{x}_{[i:i+w-1]} \cdot \mathbf{k} = \sum_{j=1}^w x_{i-1+j} \cdot k_j$$

CONVOLUTION: 1D

- ▶ Étant donné un vecteur d'*input* x de taille W

$$\mathbf{x} = (x_1, \dots, x_W)$$

et un *noyau* ou *filtre* (*kernel*, *filter*) \mathbf{k} de taille w ,

$$\mathbf{k} = (k_1, \dots, k_w)$$

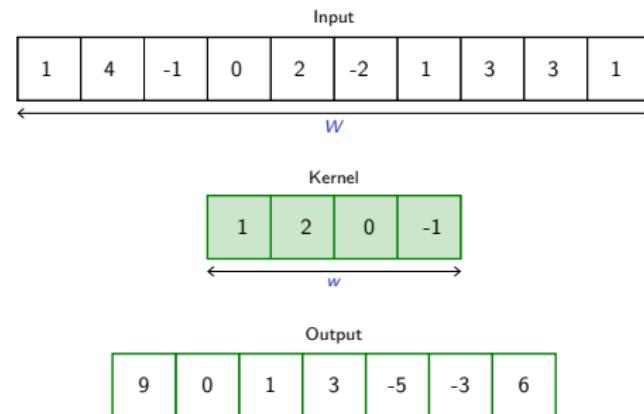
la **convolution** $\mathbf{x} * \mathbf{k}$ est un vecteur de taille $W - w + 1$ dont les composantes sont données par

$$(\mathbf{x} * \mathbf{k})_i = \mathbf{x}_{[i:i+w-1]} \cdot \mathbf{k} = \sum_{j=1}^w x_{i-1+j} \cdot k_j$$

CONVOLUTION: 1D

- ## ► Exemple numérique:

$$(1, 2, 3, 4) * (3, 2) = (3 + 4, 6 + 6, 9 + 8) = (7, 12, 17)$$

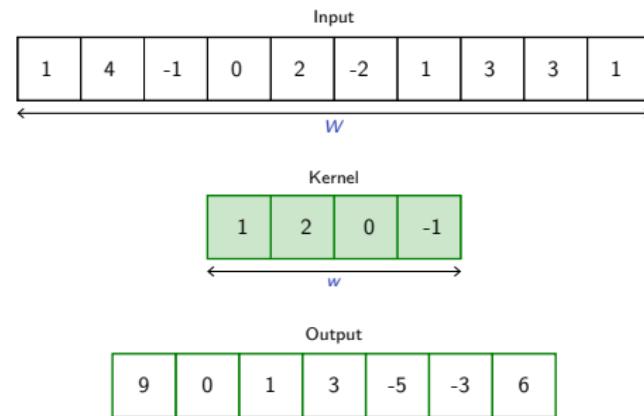


CONVOLUTION: 1D

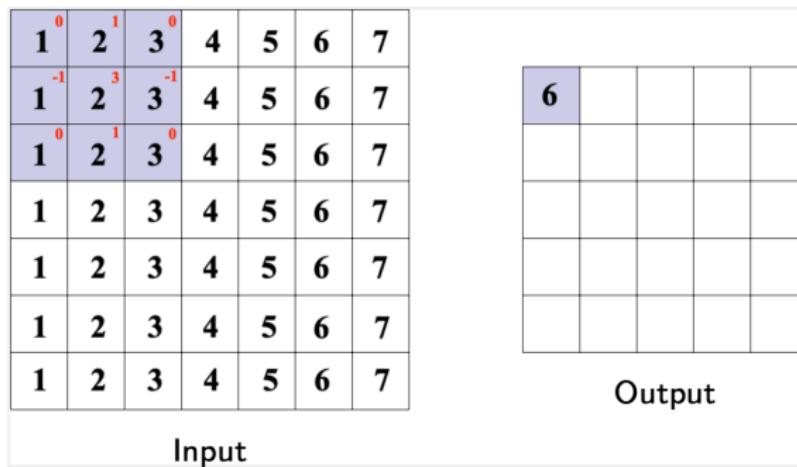
- ## ► Exemple numérique:

$$(1, 2, 3, 4) * (3, 2) = (3 + 4, 6 + 6, 9 + 8) = (7, 12, 17)$$

- ### ► Exemple visuel:

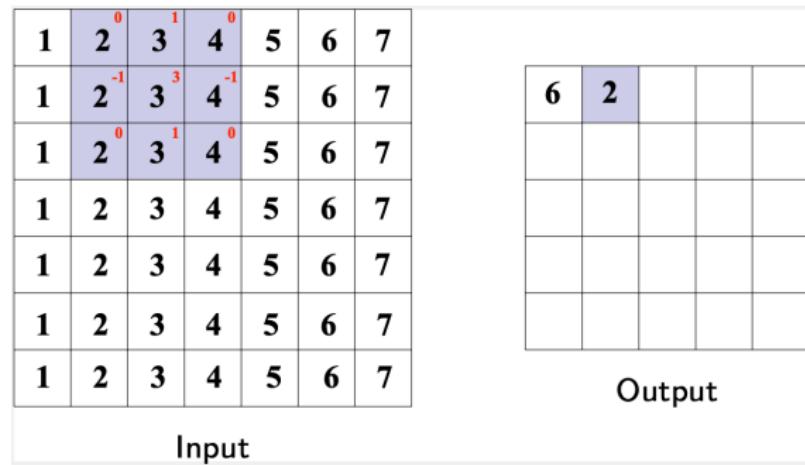


CONVOLUTION: 2D



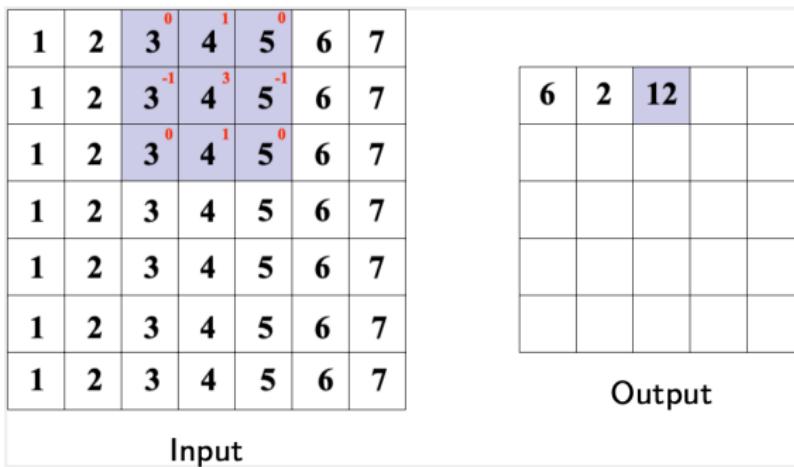
Figures taken from [Fleuret, 2022]

CONVOLUTION: 2D



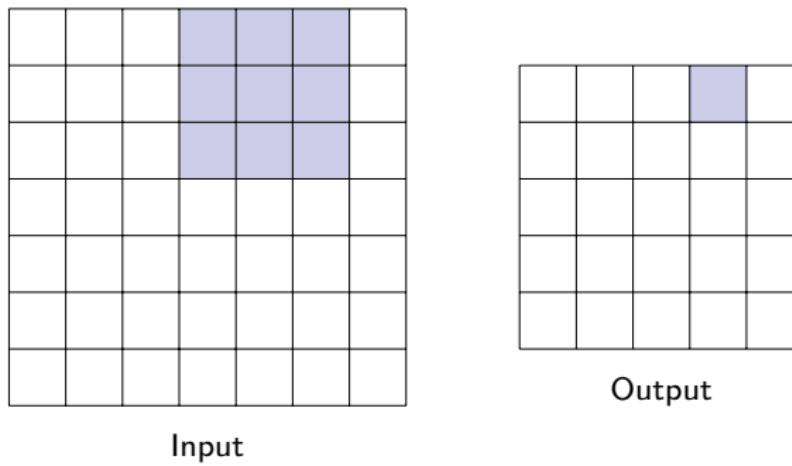
Figures taken from [\[Fleuret, 2022\]](#)

CONVOLUTION: 2D



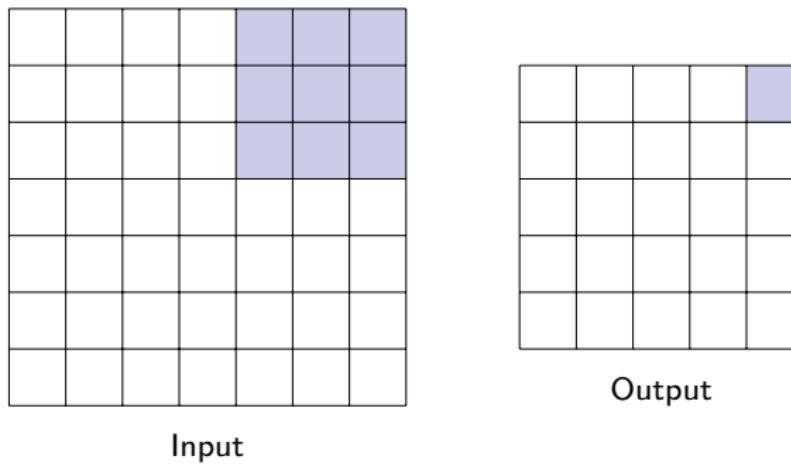
Figures taken from [Fleuret, 2022]

CONVOLUTION: 2D



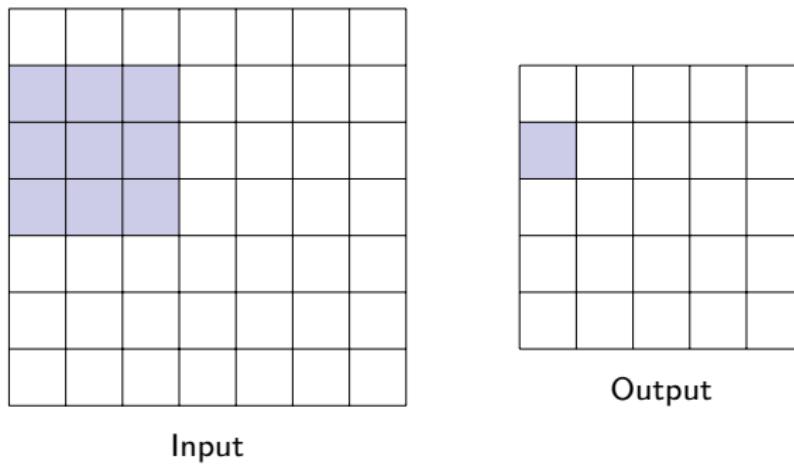
Figures taken from [\[Fleuret, 2022\]](#)

CONVOLUTION: 2D



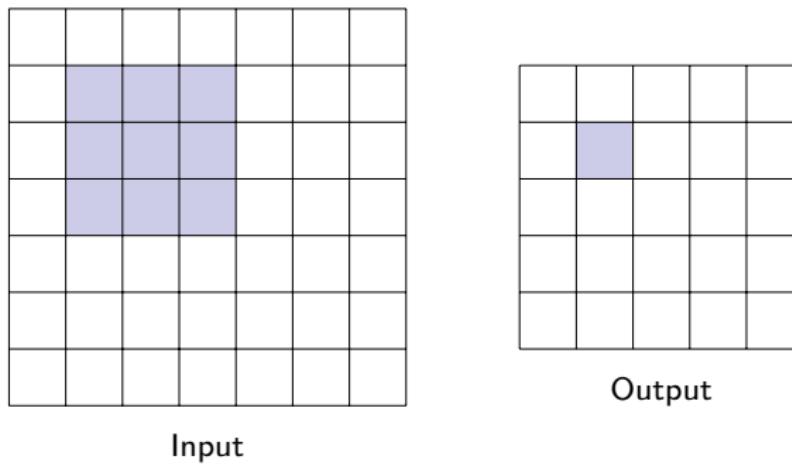
Figures taken from [\[Fleuret, 2022\]](#)

CONVOLUTION: 2D



Figures taken from [\[Fleuret, 2022\]](#)

CONVOLUTION: 2D



Figures taken from [\[Fleuret, 2022\]](#)

CONVOLUTION: 2D

- Étant donné une matrice d'*input* \mathbf{X} de taille $H \times W$

$$\mathbf{X} = (x_{i,j})$$

et un *noyau de convolution* ou *filtre* (*kernel, filter*) \mathbf{K} de taille $h \times w$,

$$\mathbf{K} = (k_{i,j})$$

la **convolution** $\mathbf{X} * \mathbf{K}$ est une matrice de taille $(H - h + 1) \times (W - w + 1)$ dont les composantes sont données par

$$\begin{aligned} (\mathbf{X} * \mathbf{K})_{i,j} &= \mathbf{X}_{[i:i+h-1, j:j+w-1]} \odot \mathbf{K} \\ &= \sum_{k=1}^h \sum_{l=1}^w x_{i-1+k, j-1+l} \cdot k_{k,l} \end{aligned}$$

CONVOLUTION: 2D

- Étant donné une matrice d'*input* \mathbf{X} de taille $H \times W$

$$\mathbf{X} = (x_{i,j})$$

et un *noyau de convolution* ou *filtre* (*kernel, filter*) \mathbf{K} de taille $h \times w$,

$$\mathbf{K} = (k_{i,j})$$

la **convolution** $\mathbf{X} * \mathbf{K}$ est une matrice de taille $(H - h + 1) \times (W - w + 1)$ dont les composantes sont données par

$$\begin{aligned} (\mathbf{X} * \mathbf{K})_{i,j} &= \mathbf{X}_{[i:i+h-1, j:j+w-1]} \odot \mathbf{K} \\ &= \sum_{k=1}^h \sum_{l=1}^w x_{i-1+k, j-1+l} \cdot k_{k,l} \end{aligned}$$

CONVOLUTION: 2D

- Étant donné une matrice d'*input* \mathbf{X} de taille $H \times W$

$$\mathbf{X} = (x_{i,j})$$

et un *noyau de convolution* ou *filtre* (*kernel, filter*) \mathbf{K} de taille $h \times w$,

$$\mathbf{K} = (k_{i,j})$$

la convolution $\mathbf{X} * \mathbf{K}$ est une matrice de taille $(H - h + 1) \times (W - w + 1)$ dont les composantes sont données par

$$\begin{aligned} (\mathbf{X} * \mathbf{K})_{i,j} &= \mathbf{X}_{[i:i+h-1, j:j+w-1]} \odot \mathbf{K} \\ &= \sum_{k=1}^h \sum_{l=1}^w x_{i-1+k, j-1+l} \cdot k_{k,l} \end{aligned}$$

CONVOLUTION: 2D

- Étant donné une matrice d'*input* \mathbf{X} de taille $H \times W$

$$\mathbf{X} = (x_{i,j})$$

et un *noyau de convolution* ou *filtre* (*kernel, filter*) \mathbf{K} de taille $h \times w$,

$$\mathbf{K} = (k_{i,j})$$

la **convolution** $\mathbf{X} * \mathbf{K}$ est une matrice de taille $(H - h + 1) \times (W - w + 1)$ dont les composantes sont données par

$$\begin{aligned} (\mathbf{X} * \mathbf{K})_{i,j} &= \mathbf{X}_{[i:i+h-1, j:j+w-1]} \odot \mathbf{K} \\ &= \sum_{k=1}^h \sum_{l=1}^w x_{i-1+k, j-1+l} \cdot k_{k,l} \end{aligned}$$

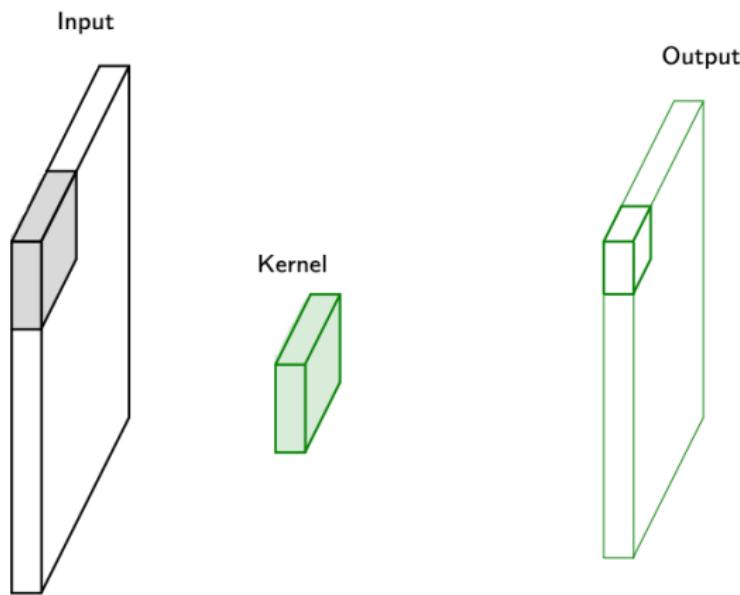
CONVOLUTION: 2D

► Exemples:

Voir site web: <https://deeplizard.com/resource/pavq7noze2>

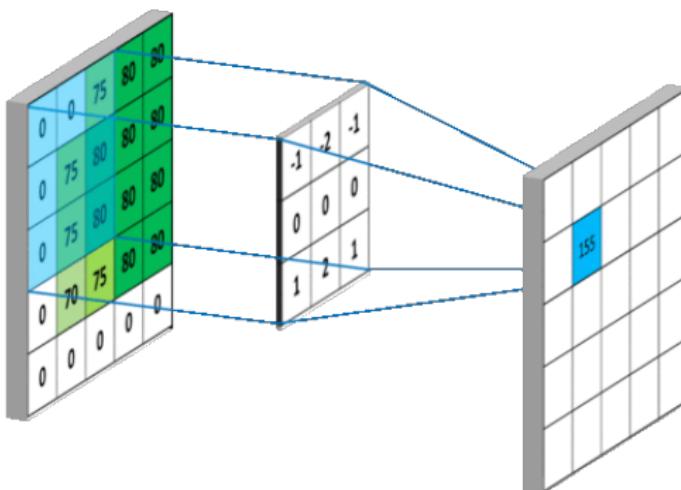
Tester différents filtres: top/bottom/left/right edge filters...

COUCHE CONVOLUTIVE (CONV. LAYER)

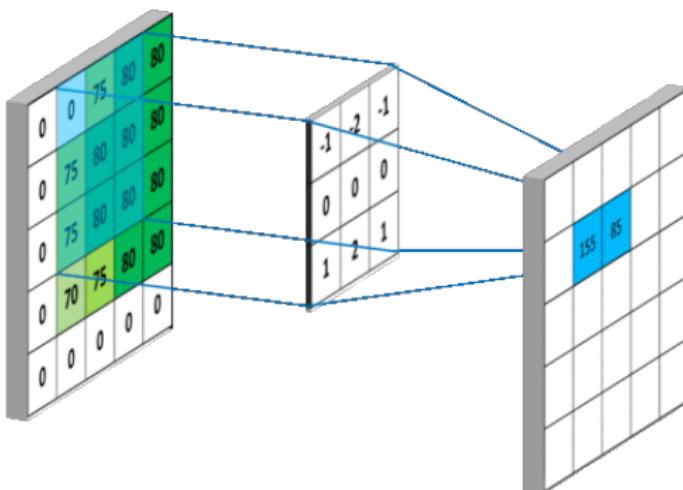


Figures taken from [Fleuret, 2022].

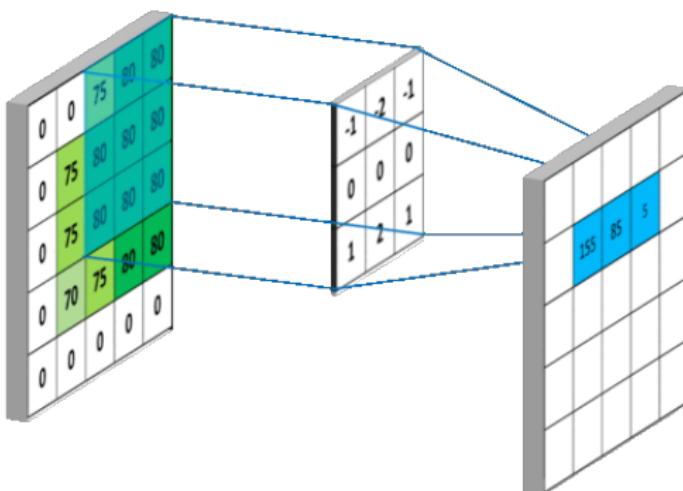
COUCHE CONVOLUTIVE (CONV. LAYER)



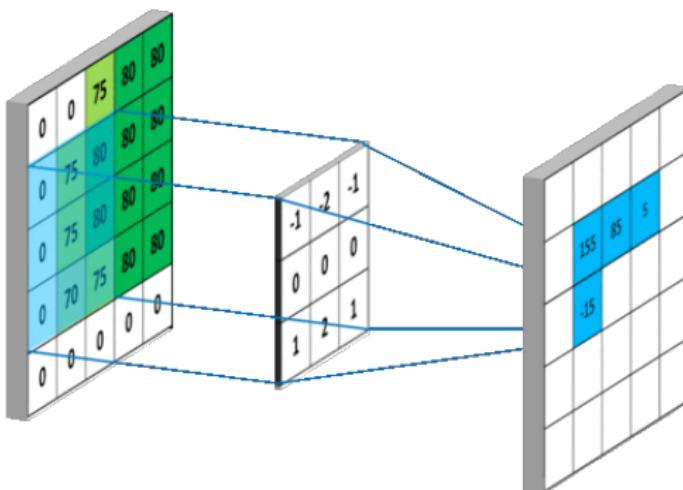
COUCHE CONVOLUTIVE (CONV. LAYER)



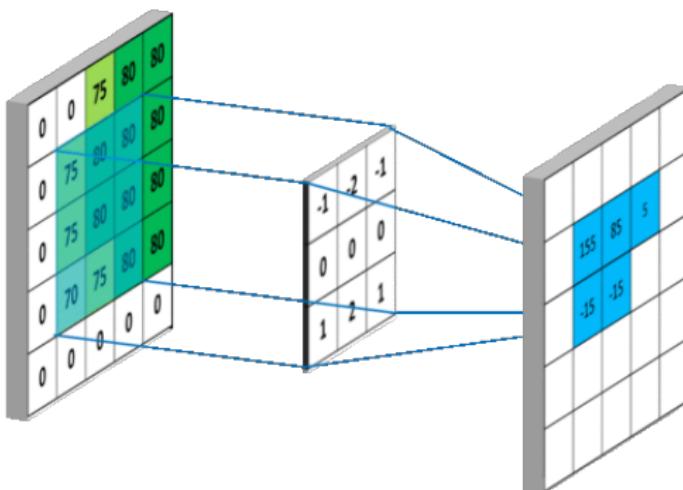
COUCHE CONVOLUTIVE (CONV. LAYER)



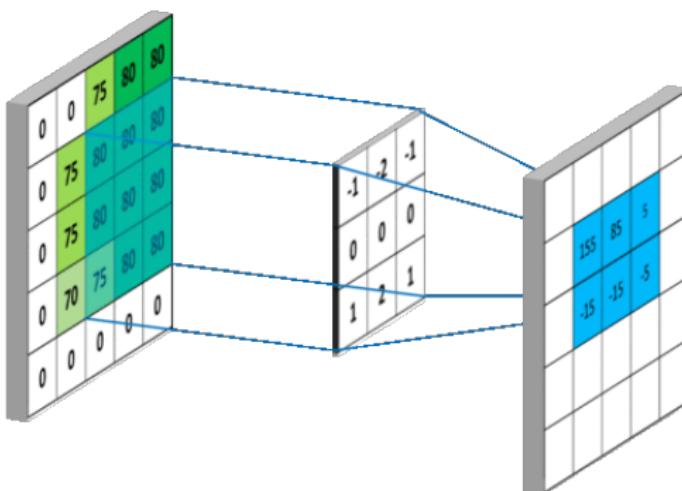
COUCHE CONVOLUTIVE (CONV. LAYER)



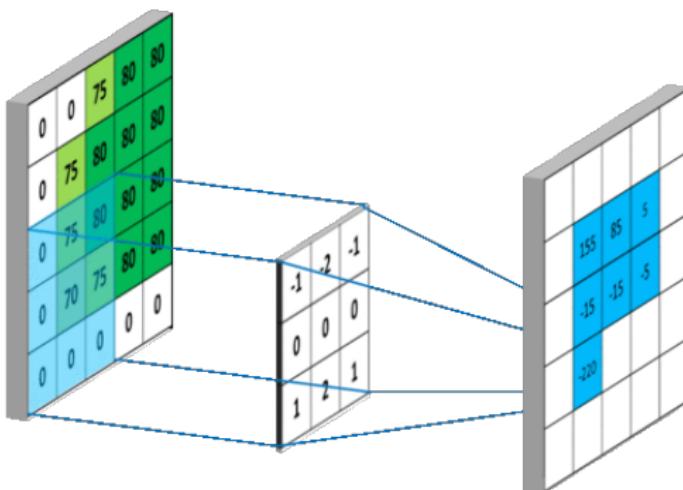
COUCHE CONVOLUTIVE (CONV. LAYER)



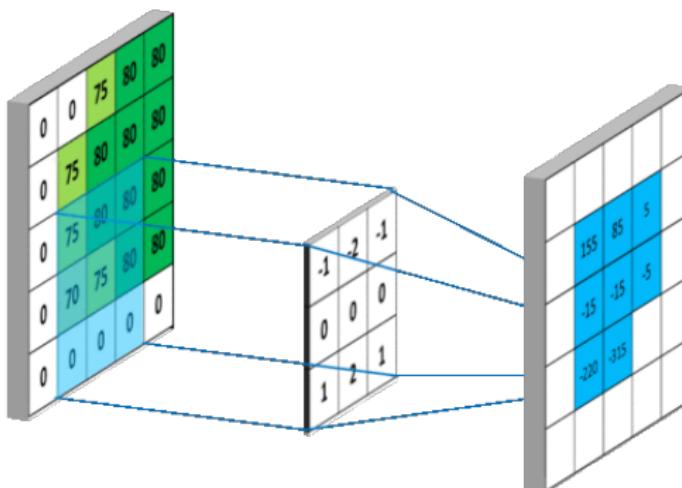
COUCHE CONVOLUTIVE (CONV. LAYER)



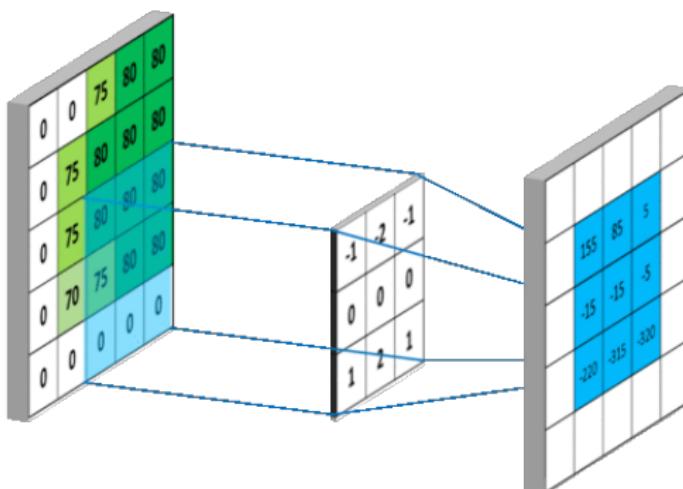
COUCHE CONVOLUTIVE (CONV. LAYER)



COUCHE CONVOLUTIVE (CONV. LAYER)



COUCHE CONVOLUTIVE (CONV. LAYER)



COUCHE CONVOLUTIVE (CONV. LAYER)

- ▶ La partie de l'input qui participe au calcul d'une composante d'output est appelée *receptive field*.
- ▶ L'output de la convolution est appelée *activation map*.
- ▶ Dans une couche convulsive, ce sont les paramètres (valeurs) du noyau K qui sont appris!

COUCHE CONVOLUTIVE (CONV. LAYER)

- ▶ La partie de l'input qui participe au calcul d'une composante d'output est appelée *receptive field*.
- ▶ L'output de la convolution est appelée *activation map*.
- ▶ Dans une couche convulsive, ce sont les paramètres (valeurs) du noyau K qui sont appris!

COUCHE CONVOLUTIVE (CONV. LAYER)

- ▶ La partie de l'input qui participe au calcul d'une composante d'output est appelée *receptive field*.
- ▶ L'output de la convolution est appelée *activation map*.
- ▶ **Dans une couche convolutive, ce sont les paramètres (valeurs) du noyau K qui sont appris!**

COUCHE CONVOLUTIVE (CONV. LAYER)

- ▶ En pratique, les images en couleurs sont représentées par des tenseur RGB de profondeur $C = 3$.
- ▶ Chacune des matrices “rouges” (R), “verts” (G), et “bleus” (B) est appelée un *canal (channel)*.
- ▶ Ainsi, la convolution sera appliquée entre une input 3D \mathbf{X} et un noyau 3D \mathbf{K} .

COUCHE CONVOLUTIVE (CONV. LAYER)

- ▶ En pratique, les images en couleurs sont représentées par des tenseur RGB de profondeur $C = 3$.
- ▶ Chacune des matrices “rouges” (R), “verts” (G), et “bleus” (B) est appelée un *canal (channel)*.
- ▶ Ainsi, la convolution sera appliquée entre une input 3D X et un noyau 3D K .

COUCHE CONVOLUTIVE (CONV. LAYER)

- ▶ En pratique, les images en couleurs sont représentées par des tenseur RGB de profondeur $C = 3$.
- ▶ Chacune des matrices “rouges” (R), “verts” (G), et “bleus” (B) est appelée un *canal (channel)*.
- ▶ Ainsi, la convolution sera appliquée entre une input 3D X et un noyau 3D K .

COUCHE CONVOLUTIVE (CONV. LAYER)

- ▶ La convolution entre une input \mathbf{X} de dimension $H \times W \times C$ et un noyau \mathbf{K} de dimension $h \times w \times C$ donne une output

$$\mathbf{Z} = \mathbf{X} * \mathbf{K}$$

de dimension $(H - h + 1) \times (W - w + 1)$.

- ▶ L'opération de convolution est généralisée à tous les canaux de manière "naturelle".
- ▶ L'output est une matrice de profondeur 1 et non C . Si on veut une output de profondeur > 1 , on rajoute des noyaux (cf. slide suivant).

COUCHE CONVOLUTIVE (CONV. LAYER)

- ▶ La convolution entre une input \mathbf{X} de dimension $H \times W \times C$ et un noyau \mathbf{K} de dimension $h \times w \times C$ donne une output

$$\mathbf{Z} = \mathbf{X} * \mathbf{K}$$

de dimension $(H - h + 1) \times (W - w + 1)$.

- ▶ L'opération de convolution est généralisée à tous les canaux de manière "naturelle".
- ▶ L'output est une matrice de profondeur 1 et non C . Si on veut une output de profondeur > 1 , on rajoute des noyaux (cf. slide suivant).

COUCHE CONVOLUTIVE (CONV. LAYER)

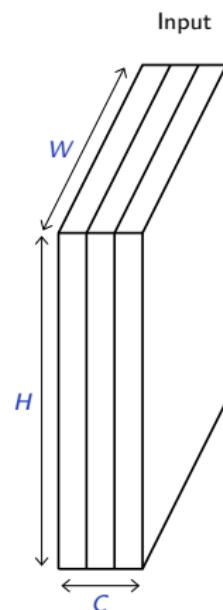
- ▶ La convolution entre une input \mathbf{X} de dimension $H \times W \times C$ et un noyau \mathbf{K} de dimension $h \times w \times C$ donne une output

$$\mathbf{Z} = \mathbf{X} * \mathbf{K}$$

de dimension $(H - h + 1) \times (W - w + 1)$.

- ▶ L'opération de convolution est généralisée à tous les canaux de manière "naturelle".
- ▶ L'output est une matrice de profondeur 1 et non C . Si on veut une output de profondeur > 1 , on rajoute des noyaux (cf. slide suivant).

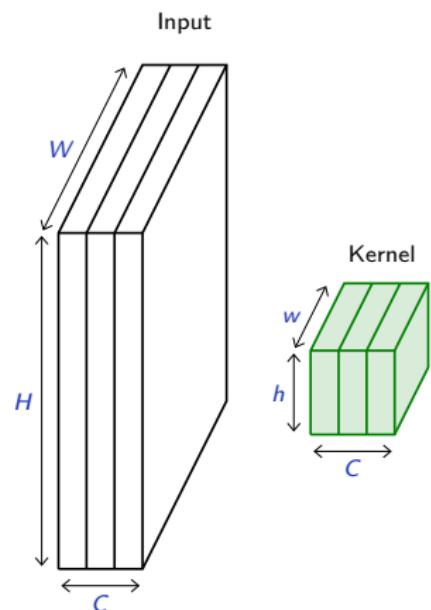
COUCHE CONVOLUTIVE (CONV. LAYER)



Figures taken from [Fleuret, 2022].

- ▶ La convolution agrège tous les canaux, donnant une output de profondeur 1.

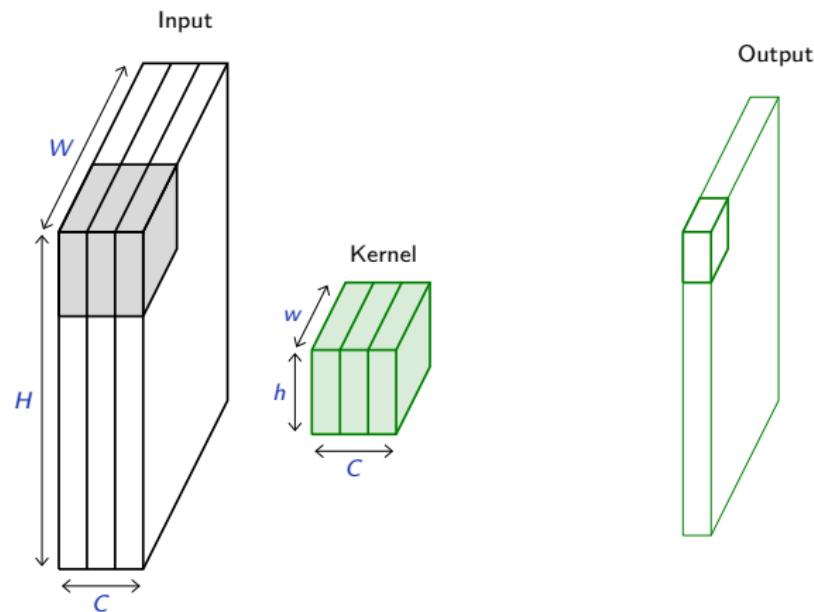
COUCHE CONVOLUTIVE (CONV. LAYER)



Figures taken from [Fleuret, 2022].

- ▶ La convolution agrège tous les canaux, donnant une output de profondeur 1.

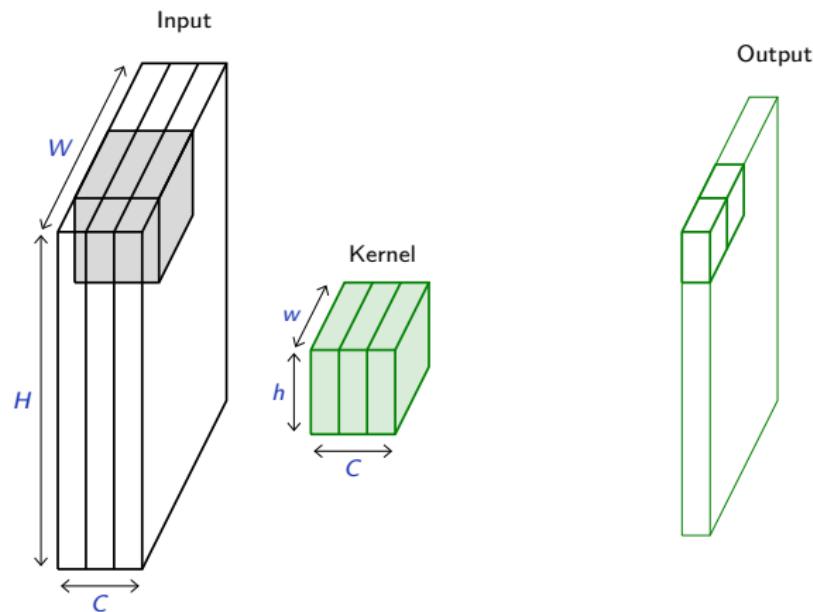
COUCHE CONVOLUTIVE (CONV. LAYER)



Figures taken from [Fleuret, 2022].

- ▶ La convolution agrège tous les canaux, donnant une output de profondeur 1.

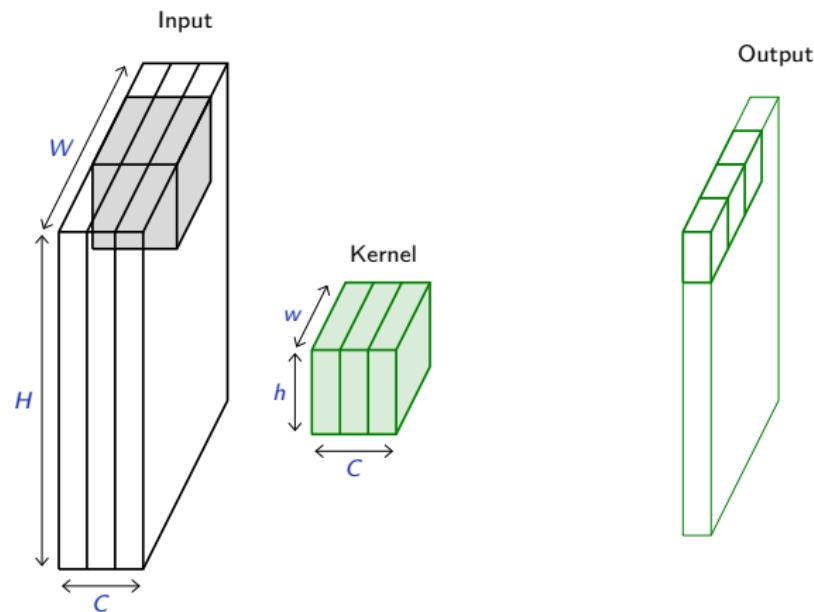
COUCHE CONVOLUTIVE (CONV. LAYER)



Figures taken from [Fleuret, 2022].

- ▶ La convolution agrège tous les canaux, donnant une output de profondeur 1.

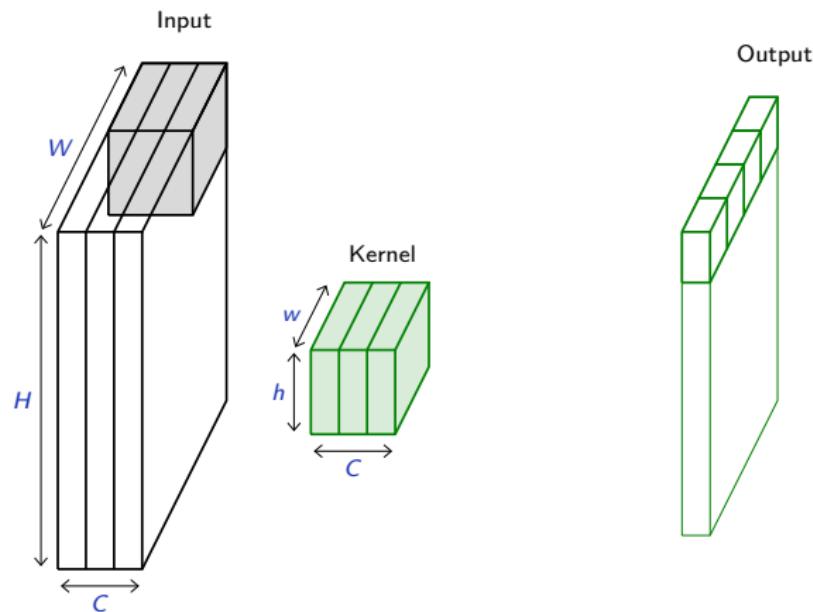
COUCHE CONVOLUTIVE (CONV. LAYER)



Figures taken from [Fleuret, 2022].

- ▶ La convolution agrège tous les canaux, donnant une output de profondeur 1.

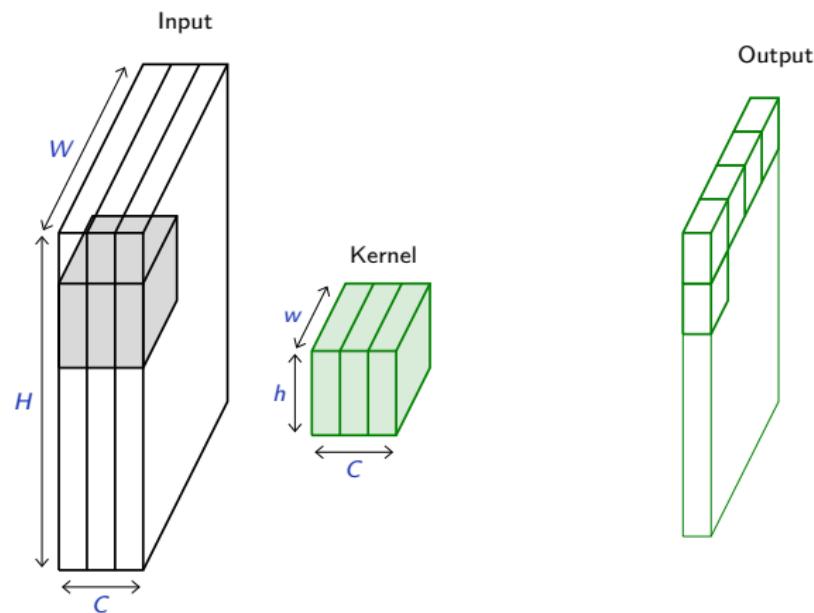
COUCHE CONVOLUTIVE (CONV. LAYER)



Figures taken from [Fleuret, 2022].

- ▶ La convolution agrège tous les canaux, donnant une output de profondeur 1.

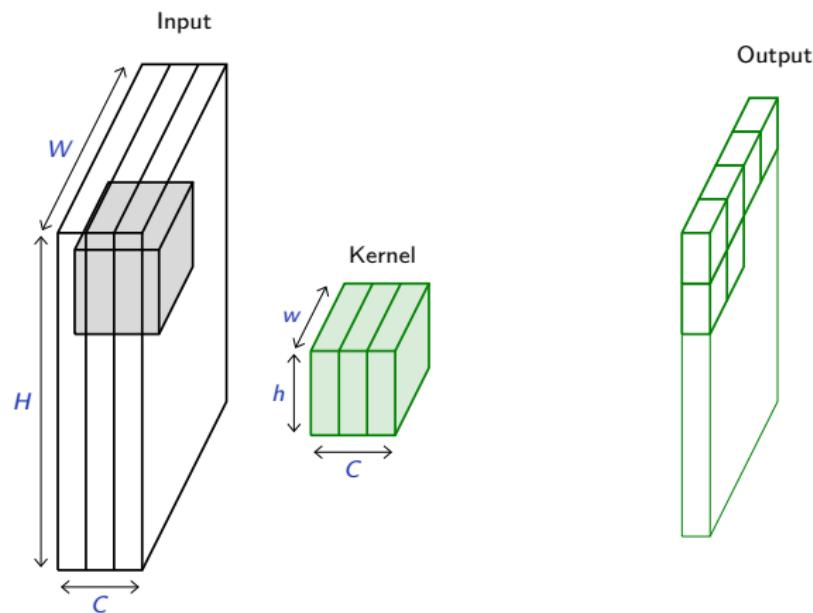
COUCHE CONVOLUTIVE (CONV. LAYER)



Figures taken from [Fleuret, 2022].

- ▶ La convolution agrège tous les canaux, donnant une output de profondeur 1.

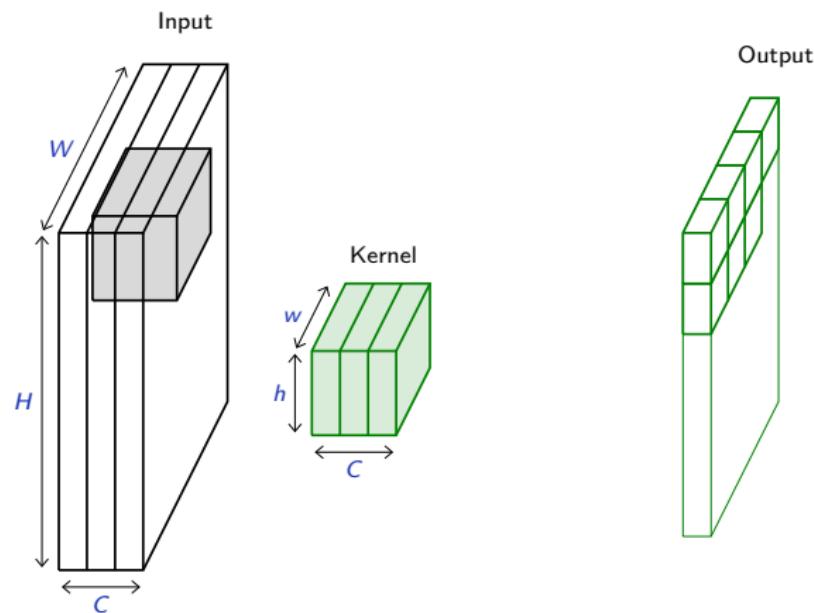
COUCHE CONVOLUTIVE (CONV. LAYER)



Figures taken from [Fleuret, 2022].

- ▶ La convolution agrège tous les canaux, donnant une output de profondeur 1.

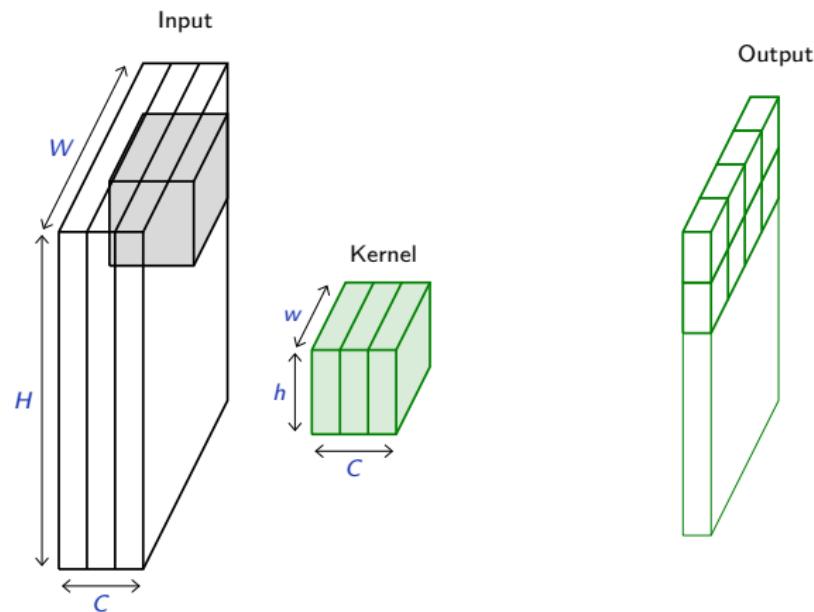
COUCHE CONVOLUTIVE (CONV. LAYER)



Figures taken from [Fleuret, 2022].

- ▶ La convolution agrège tous les canaux, donnant une output de profondeur 1.

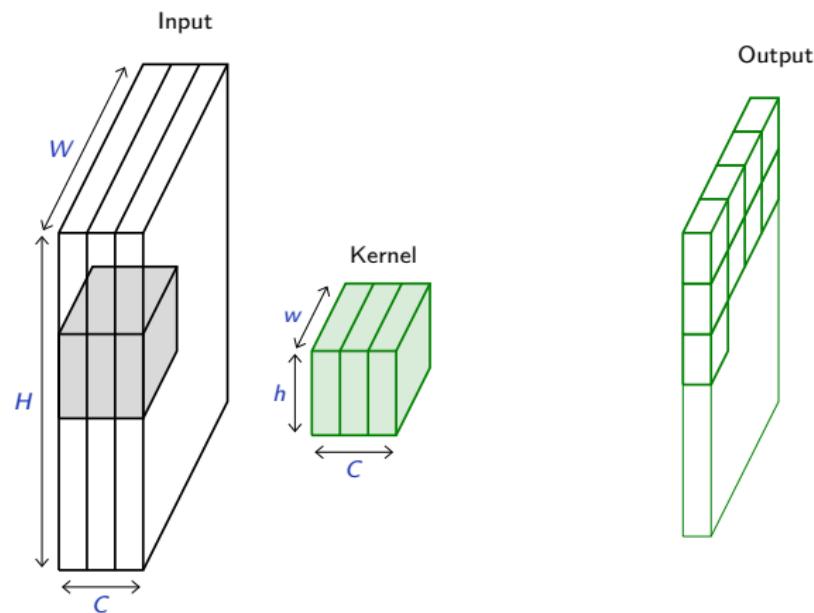
COUCHE CONVOLUTIVE (CONV. LAYER)



Figures taken from [Fleuret, 2022].

- ▶ La convolution agrège tous les canaux, donnant une output de profondeur 1.

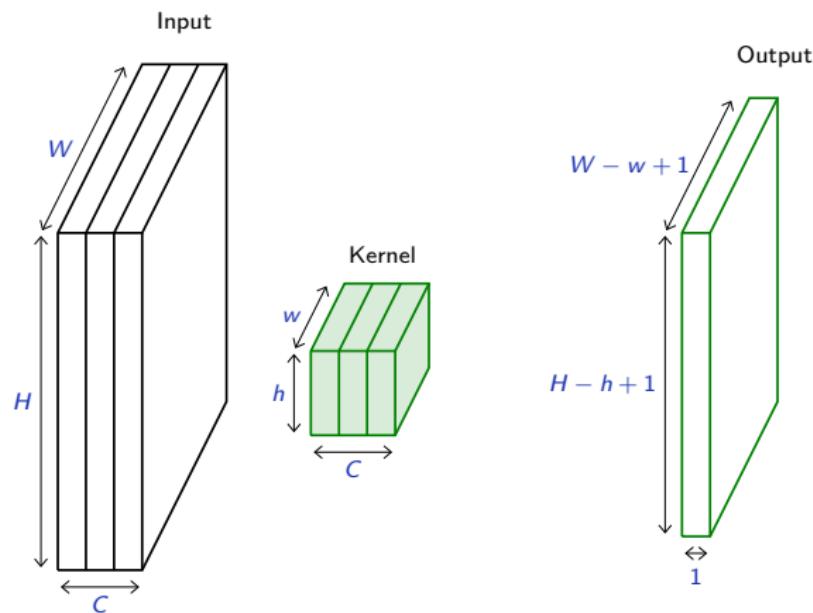
COUCHE CONVOLUTIVE (CONV. LAYER)



Figures taken from [Fleuret, 2022].

- ▶ La convolution agrège tous les canaux, donnant une output de profondeur 1.

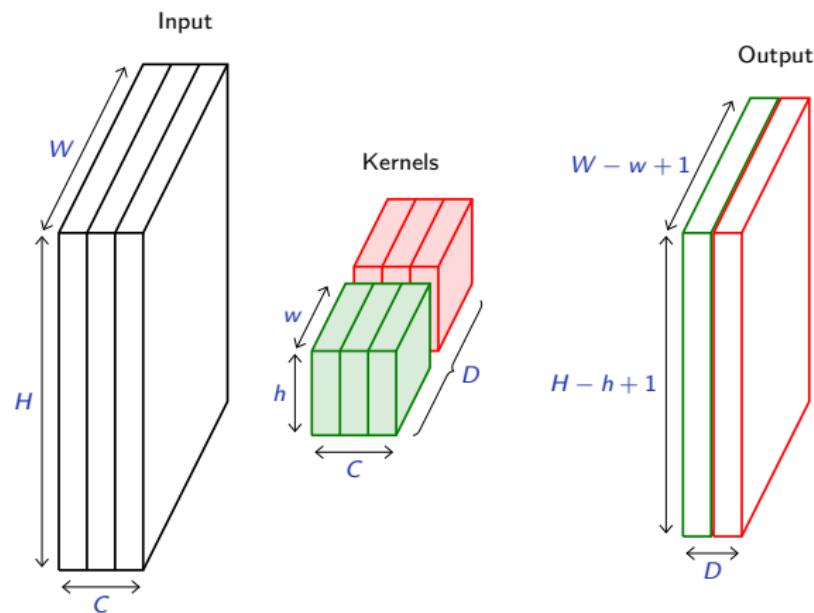
COUCHE CONVOLUTIVE (CONV. LAYER)



Figures taken from [Fleuret, 2022].

- ▶ La convolution agrège tous les canaux, donnant une output de profondeur 1.

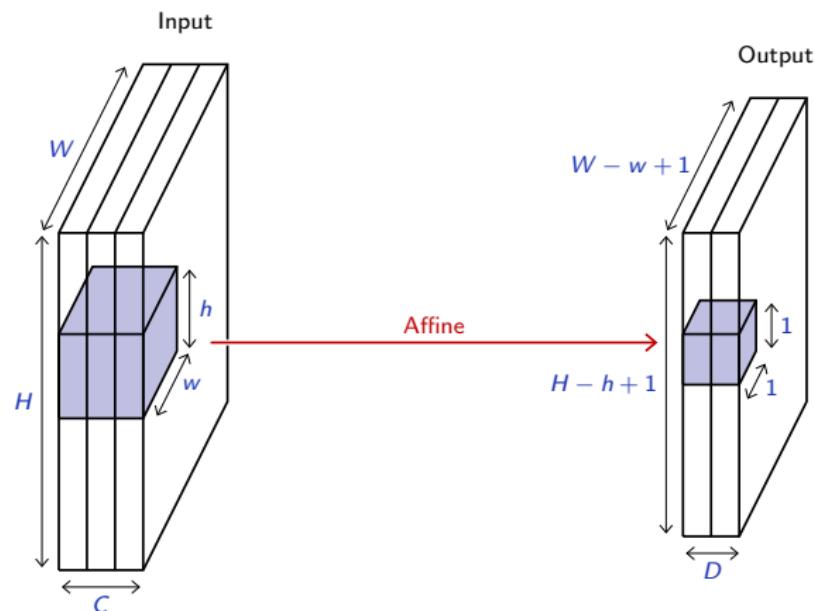
COUCHE CONVOLUTIVE (CONV. LAYER)



Figures taken from [Fleuret, 2022].

- ▶ La convolution agrège tous les canaux, donnant une output de profondeur 1.

COUCHE CONVOLUTIVE (CONV. LAYER)



Figures taken from [Fleuret, 2022].

- ▶ La convolution agrège tous les canaux, donnant une output de profondeur 1.

COUCHE CONVOLUTIVE (CONV. LAYER)

- ▶ La convolution est une opération affine entre l'input \mathbf{X} et l'output $\mathbf{Z} = \mathbf{X} * \mathbf{K}$.
- ▶ Pour introduire de la non-linéarité, on applique généralement une fonction non-linéaire σ (tanh, ReLU, etc.) à \mathbf{Z} .
- ▶ La dynamique d'une couche convulsive s'écrit alors:

$$\mathbf{A} = \sigma(\mathbf{Z}) = \sigma(\mathbf{X} * \mathbf{K})$$

où σ est appliquée composante par composante.

- ▶ Les paramètres (valeurs) du noyau \mathbf{K} sont appris par backpropagation.

COUCHE CONVOLUTIVE (CONV. LAYER)

- ▶ La convolution est une opération affine entre l'input X et l'output $Z = X * K$.
- ▶ Pour introduire de la non-linéarité, on applique généralement une fonction non-linéaire σ (tanh, ReLU, etc.) à Z .
- ▶ La dynamique d'une couche convulsive s'écrit alors:

$$A = \sigma(Z) = \sigma(X * K)$$

où σ est appliquée composante par composante.

- ▶ Les paramètres (valeurs) du noyau K sont appris par backpropagation.

COUCHE CONVOLUTIVE (CONV. LAYER)

- ▶ La convolution est une opération affine entre l'input \mathbf{X} et l'output $\mathbf{Z} = \mathbf{X} * \mathbf{K}$.
- ▶ Pour introduire de la non-linéarité, on applique généralement une fonction non-linéaire σ (tanh, ReLU, etc.) à \mathbf{Z} .
- ▶ La dynamique d'une couche convulsive s'écrit alors:

$$\mathbf{A} = \sigma(\mathbf{Z}) = \sigma(\mathbf{X} * \mathbf{K})$$

où σ est appliquée composante par composante.

- ▶ Les paramètres (valeurs) du noyau \mathbf{K} sont appris par backpropagation.

COUCHE CONVOLUTIVE (CONV. LAYER)

- ▶ La convolution est une opération affine entre l'input \mathbf{X} et l'output $\mathbf{Z} = \mathbf{X} * \mathbf{K}$.
- ▶ Pour introduire de la non-linéarité, on applique généralement une fonction non-linéaire σ (tanh, ReLU, etc.) à \mathbf{Z} .
- ▶ La dynamique d'une couche convulsive s'écrit alors:

$$\mathbf{A} = \sigma(\mathbf{Z}) = \sigma(\mathbf{X} * \mathbf{K})$$

où σ est appliquée composante par composante.

- ▶ Les paramètres (valeurs) du noyau \mathbf{K} sont appris par backpropagation.

PADDING, STRIDE ET DILATATION

Les couches convolutives possèdent également les spécifications suivantes:

- ▶ **Le padding:** spécifie la dimension d'un cadre de valeur 0 ajouté de autour de l'input.
- ▶ **Le stride:** spécifie la taille du pas avec lequel le filtre de convolution parcourt l'input.
- ▶ **La dilatation:** module l'expansion (ou écartement) du filtre sans ajouter de paramètres.

PADDING, STRIDE ET DILATATION

Les couches convolutives possèdent également les spécifications suivantes:

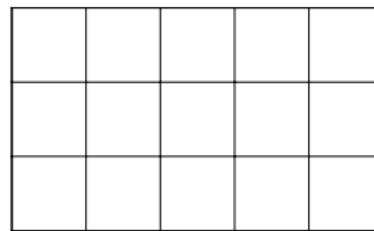
- ▶ **Le padding:** spécifie la dimension d'un cadre de valeur 0 ajouté de autour de l'input.
- ▶ **Le stride:** spécifie la taille du pas avec lequel le filtre de convolution parcourt l'input.
- ▶ **La dilatation:** module l'expansion (ou écartement) du filtre sans ajouter de paramètres.

PADDING, STRIDE ET DILATATION

Les couches convolutives possèdent également les spécifications suivantes:

- ▶ **Le padding:** spécifie la dimension d'un cadre de valeur 0 ajouté de autour de l'input.
- ▶ **Le stride:** spécifie la taille du pas avec lequel le filtre de convolution parcourt l'input.
- ▶ **La dilatation:** module l'expansion (ou écartement) du filtre sans ajouter de paramètres.

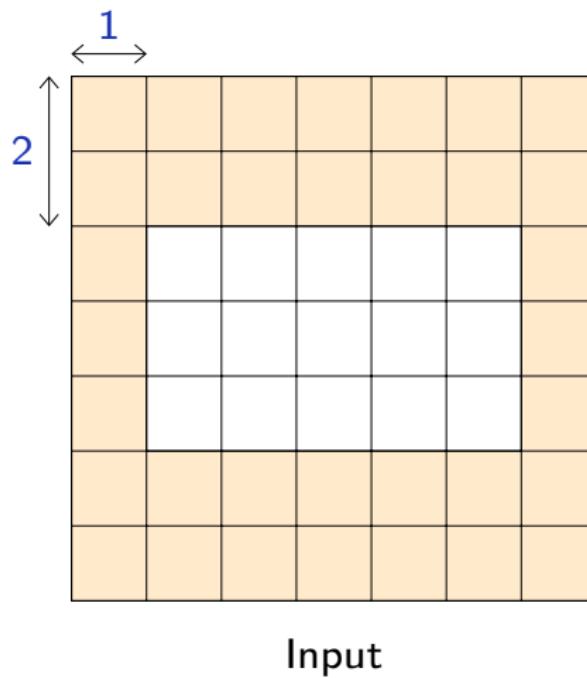
PADDING AND STRIDE



Input

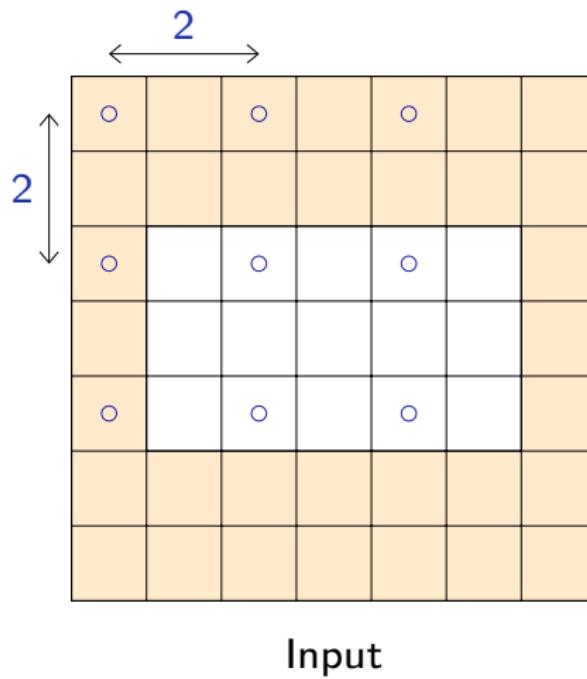
Figures taken from [Fleuret, 2022].

PADDING AND STRIDE



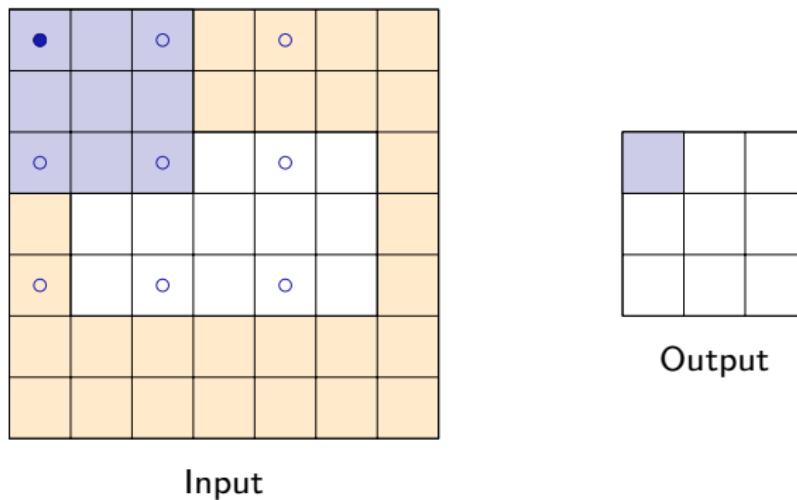
Figures taken from [Fleuret, 2022].

PADDING AND STRIDE



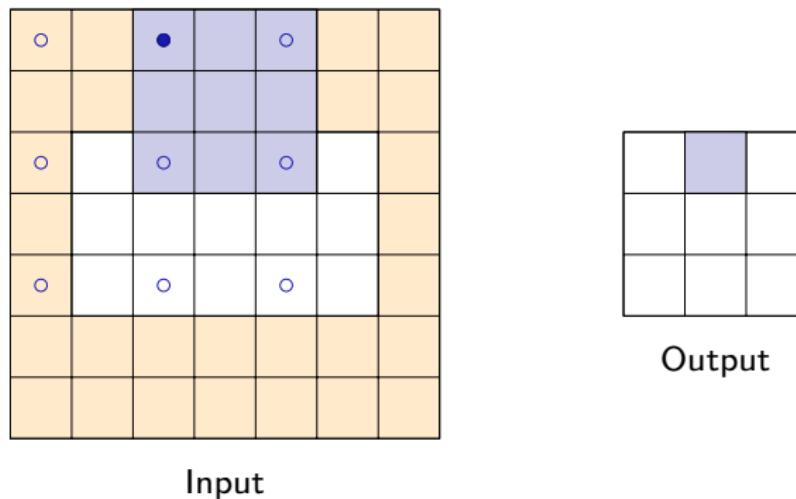
Figures taken from [Fleuret, 2022].

PADDING AND STRIDE



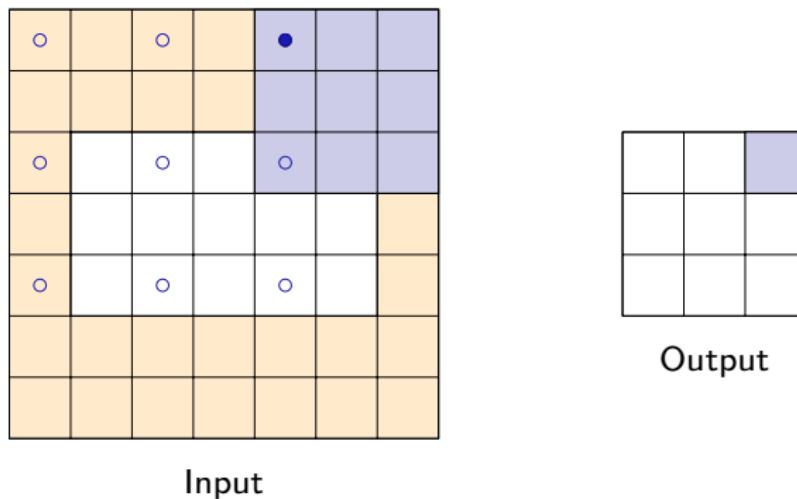
Figures taken from [\[Fleuret, 2022\]](#).

PADDING AND STRIDE



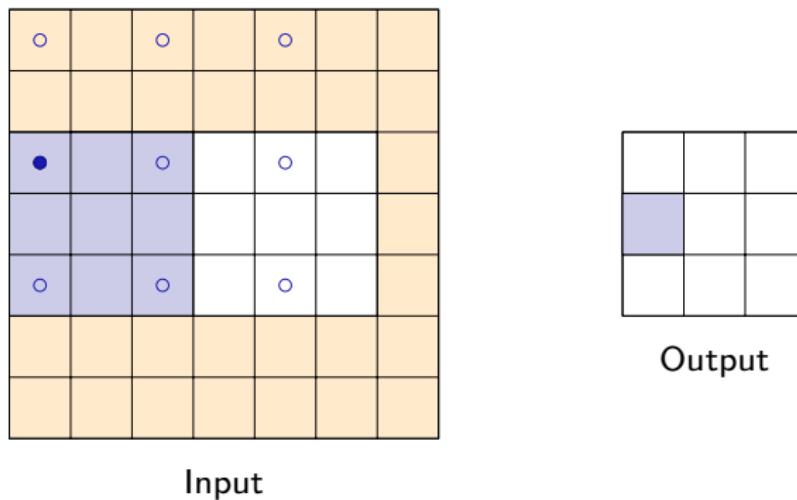
Figures taken from [\[Fleuret, 2022\]](#).

PADDING AND STRIDE



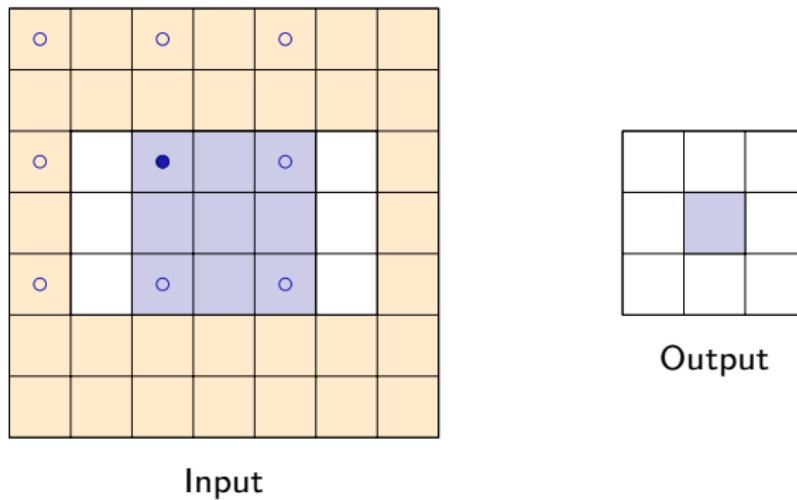
Figures taken from [\[Fleuret, 2022\]](#).

PADDING AND STRIDE



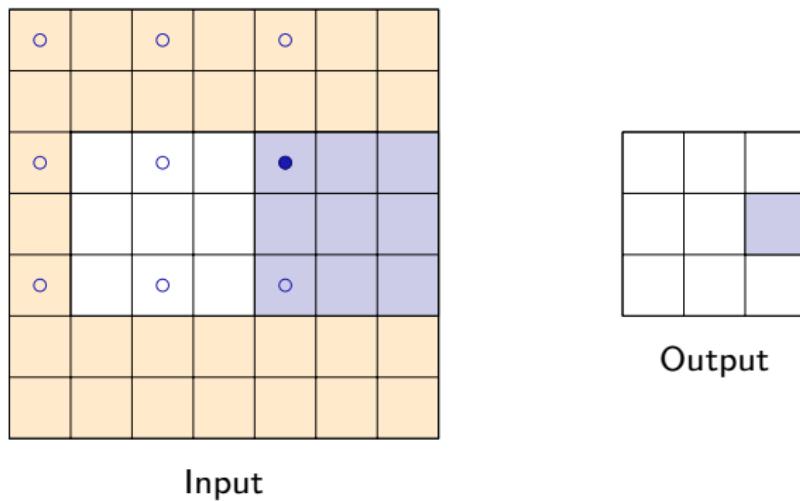
Figures taken from [\[Fleuret, 2022\]](#).

PADDING AND STRIDE



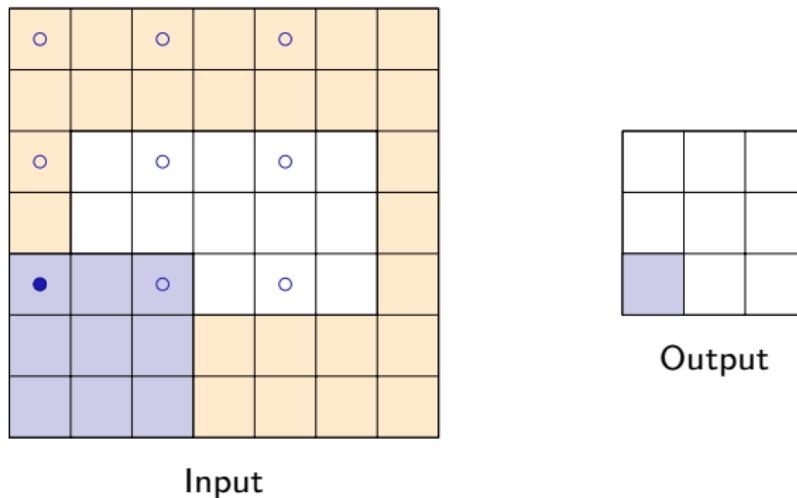
Figures taken from [\[Fleuret, 2022\]](#).

PADDING AND STRIDE



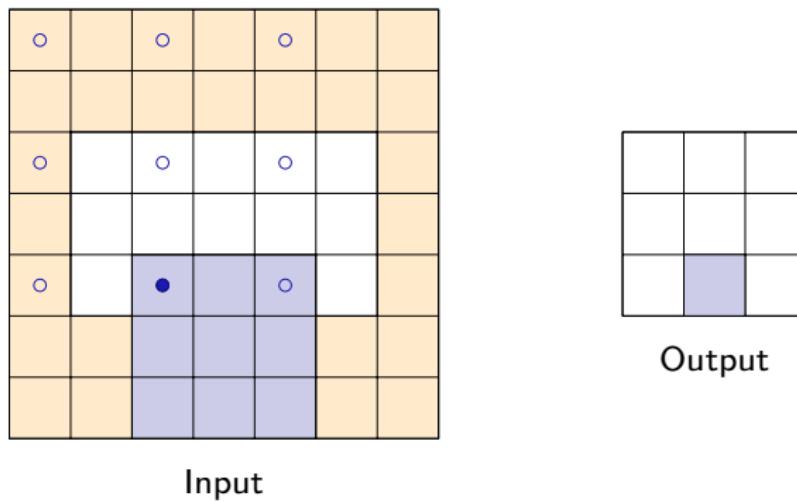
Figures taken from [\[Fleuret, 2022\]](#).

PADDING AND STRIDE



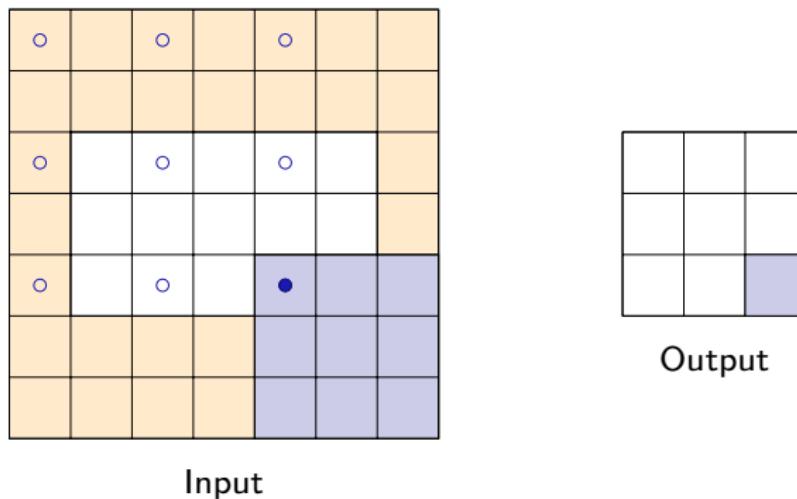
Figures taken from [\[Fleuret, 2022\]](#).

PADDING AND STRIDE



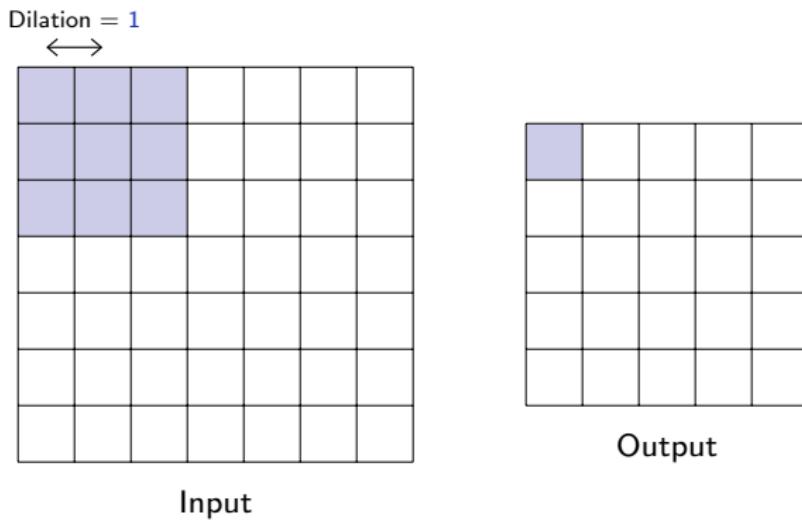
Figures taken from [\[Fleuret, 2022\]](#).

PADDING AND STRIDE



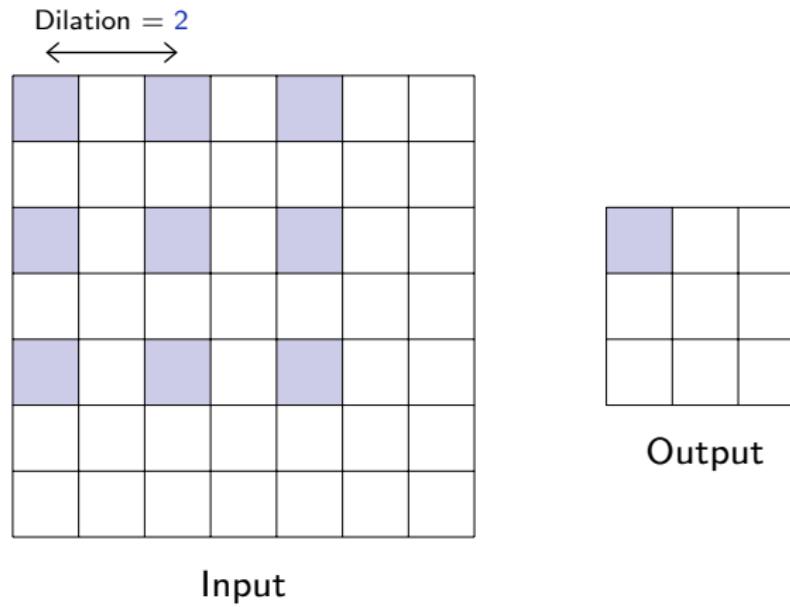
Figures taken from [\[Fleuret, 2022\]](#).

DILATATION



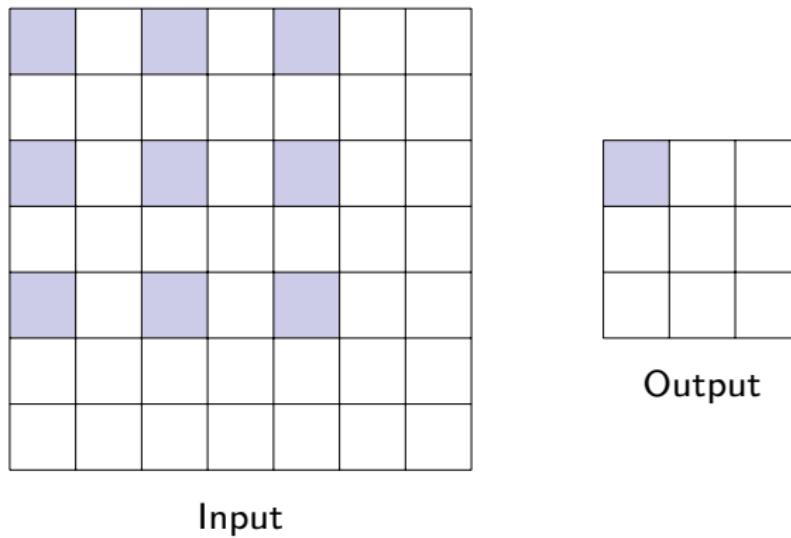
Figures taken from [[Fleuret, 2022](#)].

DILATATION



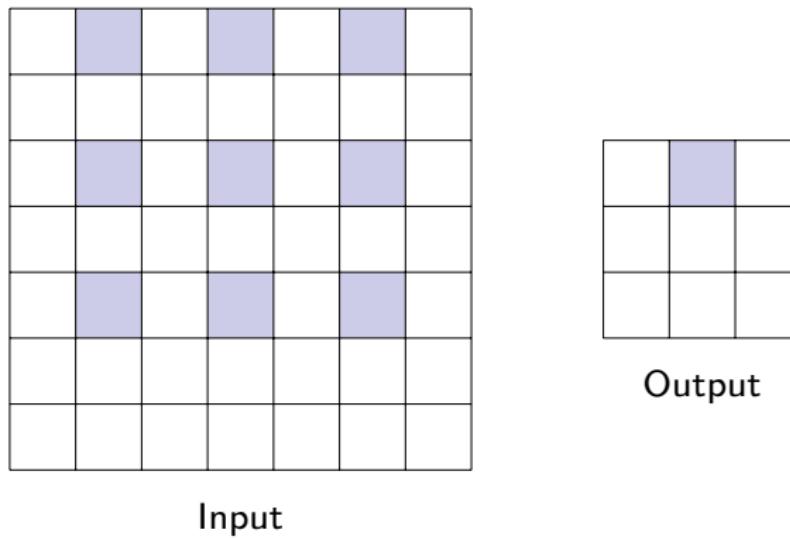
Figures taken from [Fleuret, 2022].

DILATATION



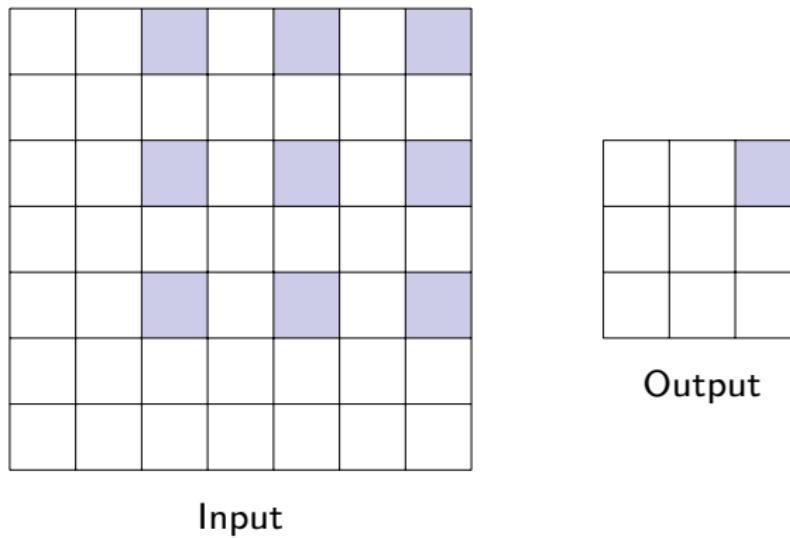
Figures taken from [\[Fleuret, 2022\]](#).

DILATATION



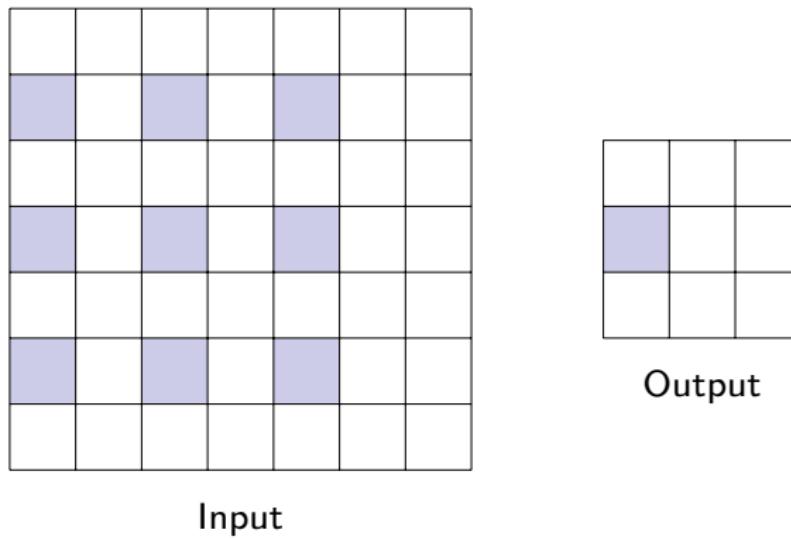
Figures taken from [\[Fleuret, 2022\]](#).

DILATATION



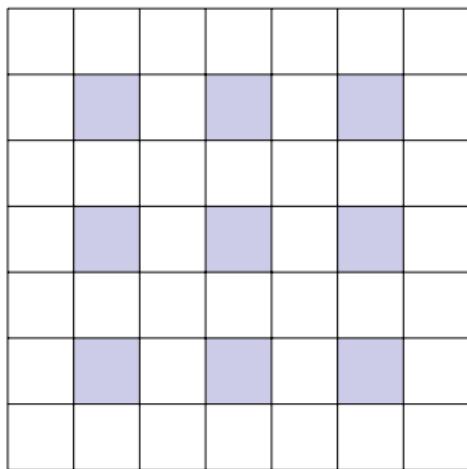
Figures taken from [\[Fleuret, 2022\]](#).

DILATATION

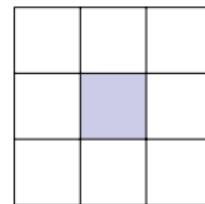


Figures taken from [\[Fleuret, 2022\]](#).

DILATATION



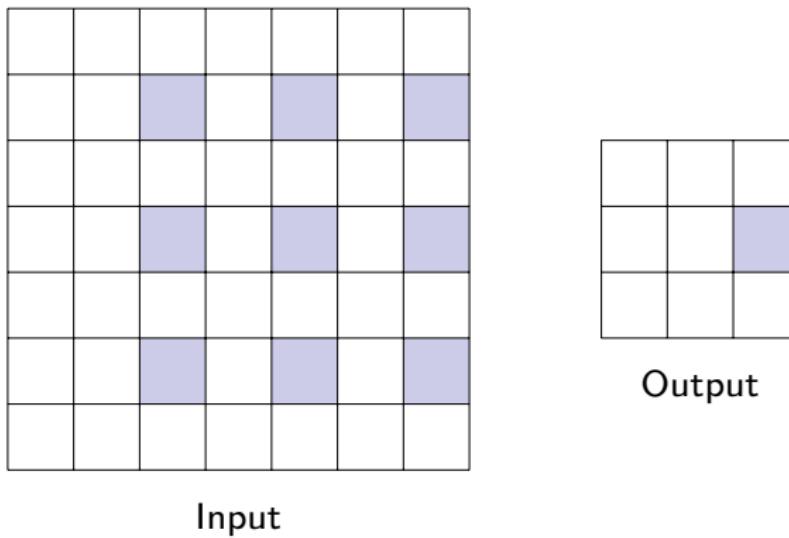
Input



Output

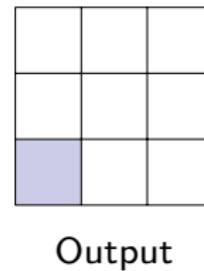
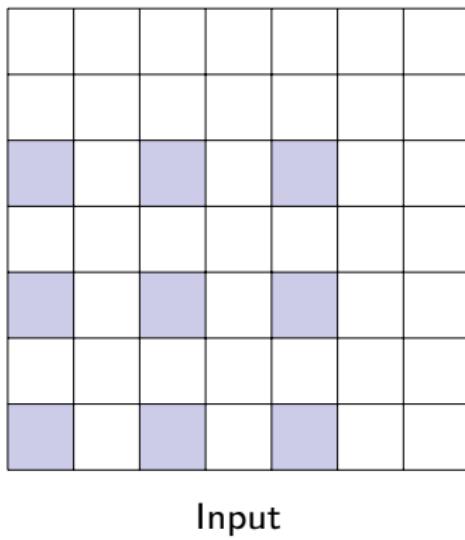
Figures taken from [\[Fleuret, 2022\]](#).

DILATATION



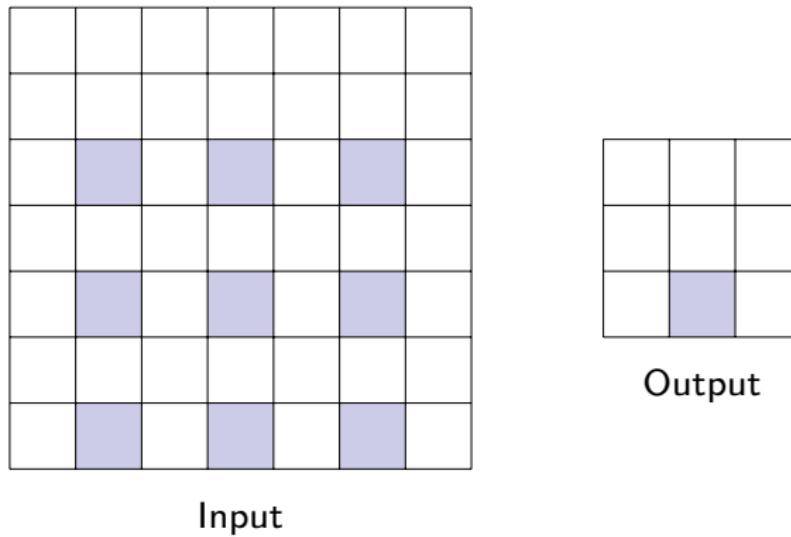
Figures taken from [\[Fleuret, 2022\]](#).

DILATATION



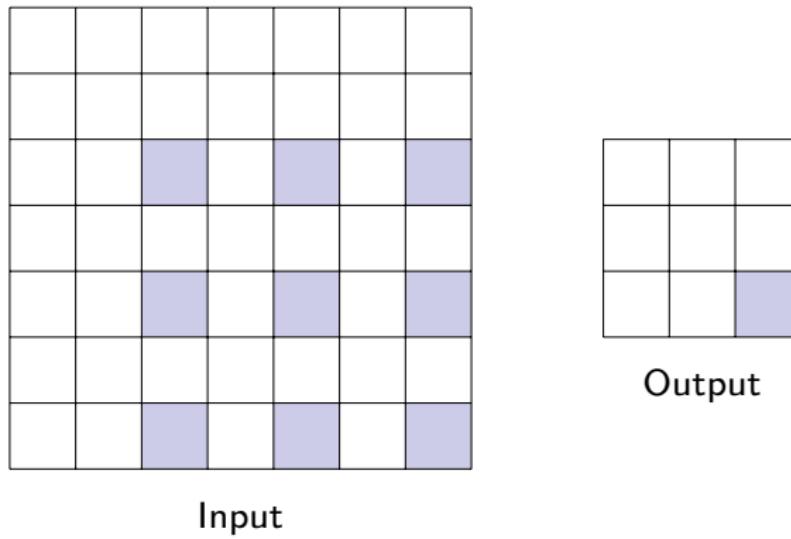
Figures taken from [\[Fleuret, 2022\]](#).

DILATATION



Figures taken from [\[Fleuret, 2022\]](#).

DILATATION



Figures taken from [\[Fleuret, 2022\]](#).

AVANTAGES DES COUCHES CONVOLUTIVES

Les couches convolutives possèdent les avantages suivants par rapport aux couches denses classiques:

► Parameter sharing

Stockage mémoire:

k paramètres partagés (conv.) vs $m \times n$ paramètres (dense), avec $k \ll m, n$.

► Sparse connectivity

Forward pass:

$O(k \times n)$ (conv.) vs $O(m \times n)$ (dense) pour calcul de l'output, avec $k \ll m$.

AVANTAGES DES COUCHES CONVOLUTIVES

Les couches convolutives possèdent les avantages suivants par rapport aux couches denses classiques:

▶ Parameter sharing

Stockage mémoire:

k paramètres partagés (conv.) vs $m \times n$ paramètres (dense), avec $k \ll m, n$.

▶ Sparse connectivity

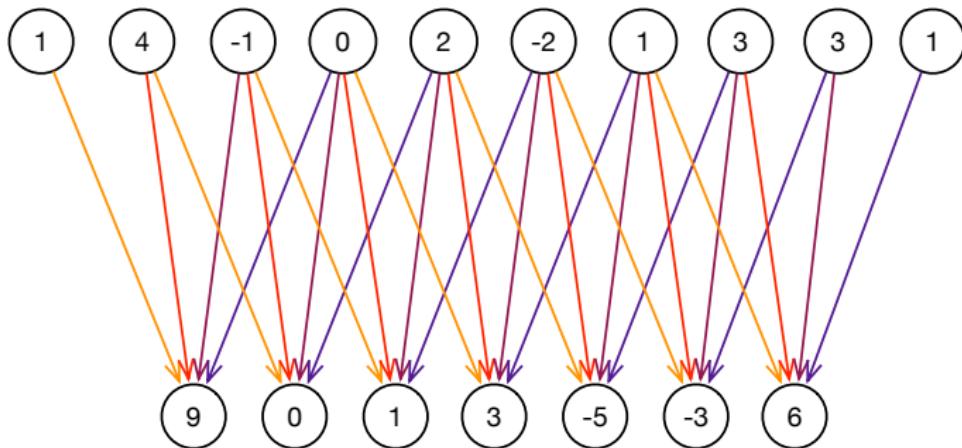
Forward pass:

$O(k \times n)$ (conv.) vs $O(m \times n)$ (dense) pour calcul de l'output, avec $k \ll m$.

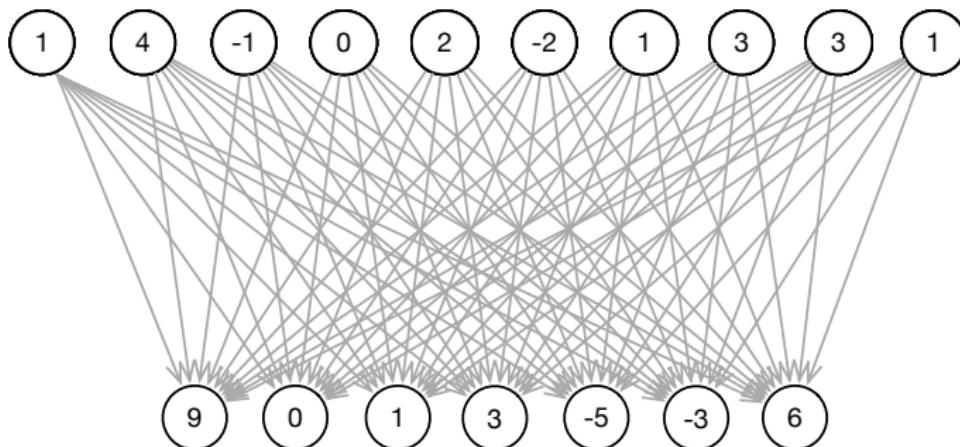
AVANTAGES DES COUCHES CONVOLUTIVES



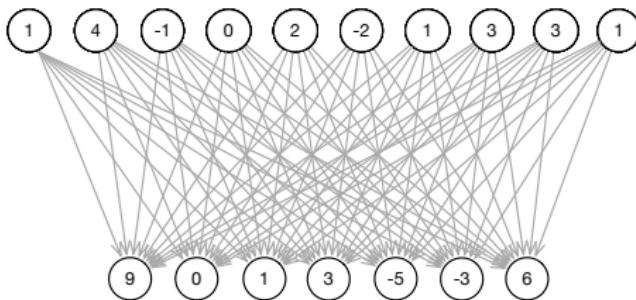
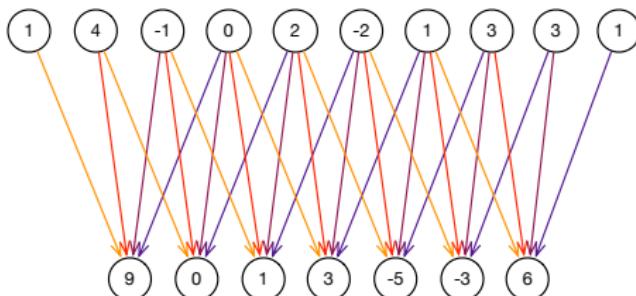
AVANTAGES DES COUCHES CONVOLUTIVES



AVANTAGES DES COUCHES CONVOLUTIVES



AVANTAGES DES COUCHES CONVOLUTIVES



AVANTAGES DES COUCHES CONVOLUTIVES

► **Equivariant representation**

Des patterns similaires de l'input (e.g., bords, arrondis, etc.), quels que soient leurs emplacements, sont transformés de manières similaires dans l'output.

Ainsi, les filtres apprennent des représentations de patterns spécifiques, et une image sera caractérisée par l'emplacement de ces patterns.

AVANTAGES DES COUCHES CONVOLUTIVES

- ▶ Les bords supérieurs dans l'input, indépendamment de leur emplacement, sont détectés de manières similaires dans l'output (invariance par translation).

POOLING

1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7

Input

3			

Output

Figures taken from [\[Fleuret, 2022\]](#)

POOLING

1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7

Input

3	4			

Output

Figures taken from [\[Fleuret, 2022\]](#)

POOLING

1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7

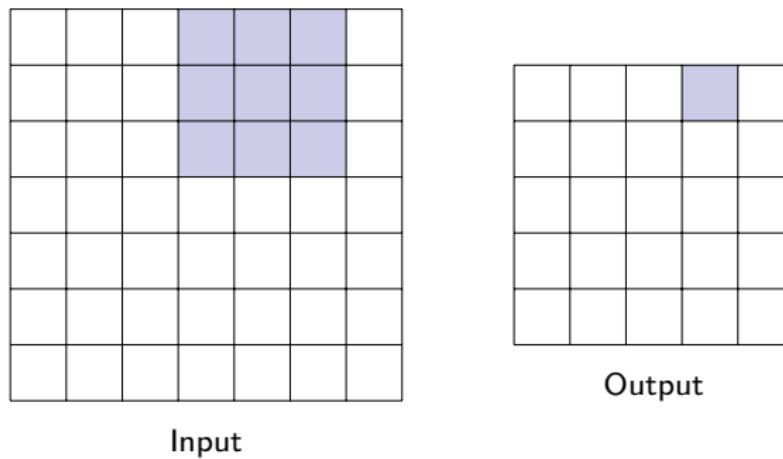
Input

3	4	5		

Output

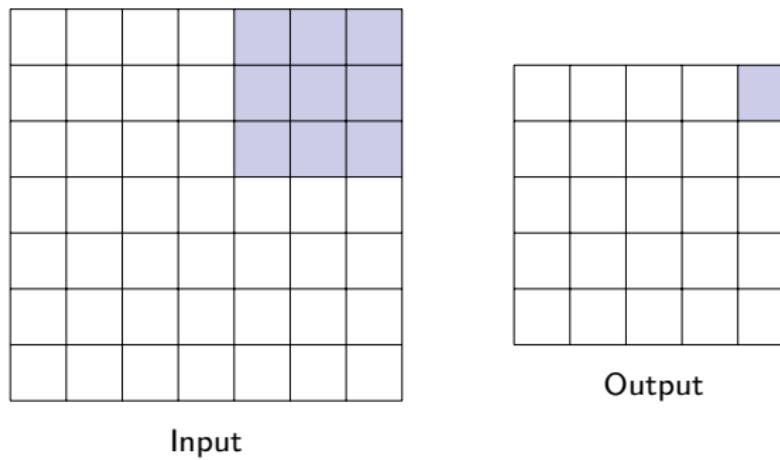
Figures taken from [\[Fleuret, 2022\]](#)

POOLING



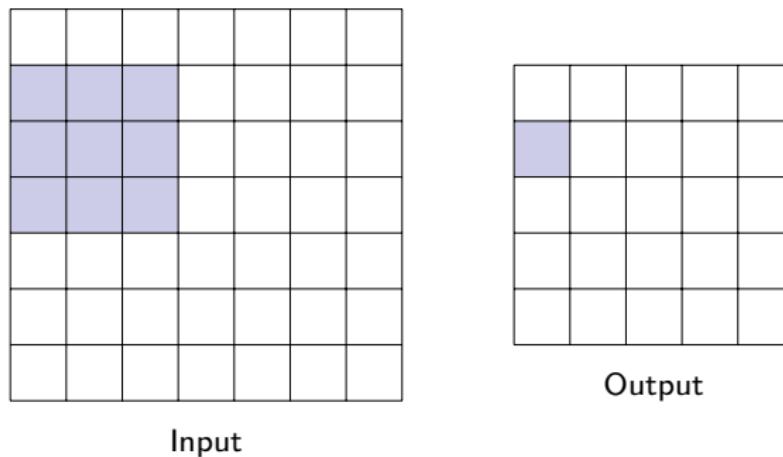
Figures taken from [\[Fleuret, 2022\]](#)

POOLING



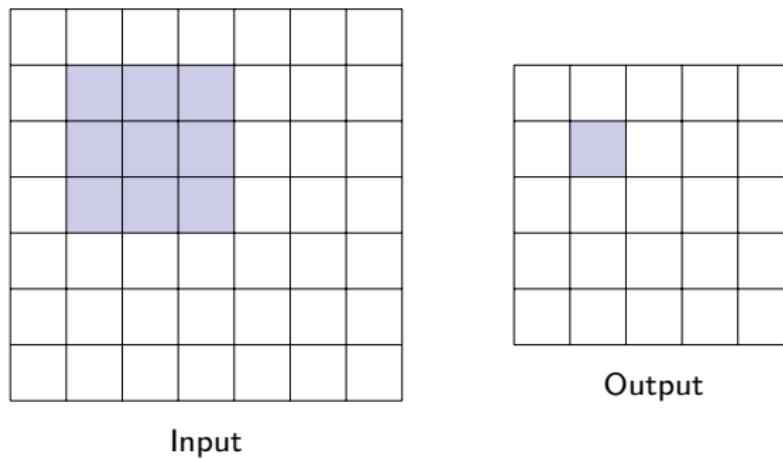
Figures taken from [\[Fleuret, 2022\]](#)

POOLING



Figures taken from [\[Fleuret, 2022\]](#)

POOLING



Figures taken from [\[Fleuret, 2022\]](#)

POOLING

- ▶ Étant donné une matrice d'*input* \mathbf{X} de taille $H \times W$ et un *filtre de pooling* \mathbf{P} de taille $h \times w$, les opérations de **maximum and average pooling**

$$\text{MaxPool}(\mathbf{X}) \text{ et } \text{AvgPool}(\mathbf{X})$$

donnent des matrices de taille $(H - h + 1) \times (W - w + 1)$ dont les composantes sont

$$\begin{aligned}\text{MaxPool}(\mathbf{X})_{i,j} &= \max(\mathbf{X}_{[i:i+h-1, j:j+w-1]}) \\ \text{AvgPool}(\mathbf{X})_{i,j} &= \text{avg}(\mathbf{X}_{[i:i+h-1, j:j+w-1]})\end{aligned}$$

- ▶ Contrairement aux convolutions, les opérations de pooling ne font intervenir aucun paramètres!

POOLING

- ▶ Étant donné une matrice d'*input* \mathbf{X} de taille $H \times W$ et un *filtre de pooling* \mathbf{P} de taille $h \times w$, les opérations de **maximum and average pooling**

$$\text{MaxPool}(\mathbf{X}) \text{ et } \text{AvgPool}(\mathbf{X})$$

donnent des matrices de taille $(H - h + 1) \times (W - w + 1)$ dont les composantes sont

$$\begin{aligned}\text{MaxPool}(\mathbf{X})_{i,j} &= \max(\mathbf{X}_{[i:i+h-1, j:j+w-1]}) \\ \text{AvgPool}(\mathbf{X})_{i,j} &= \text{avg}(\mathbf{X}_{[i:i+h-1, j:j+w-1]})\end{aligned}$$

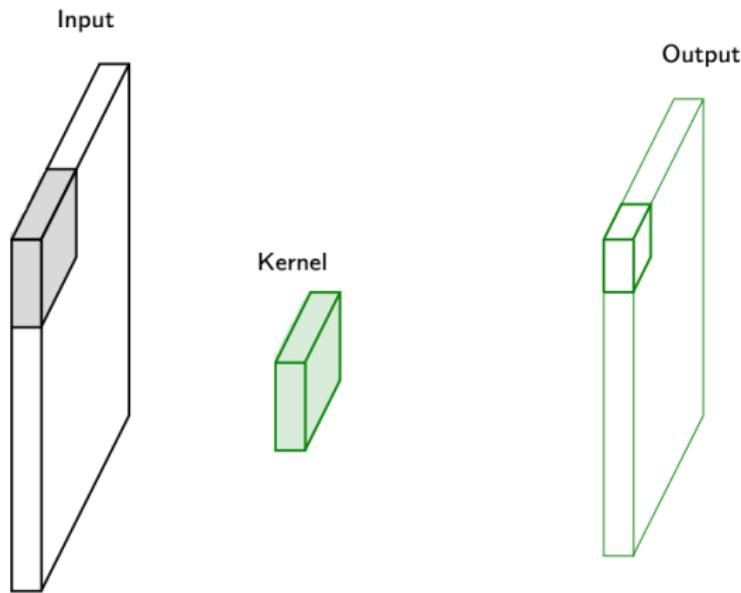
- ▶ Contrairement aux convolutions, les opérations de pooling ne font intervenir aucun paramètres!

POOLING

► Exemples:

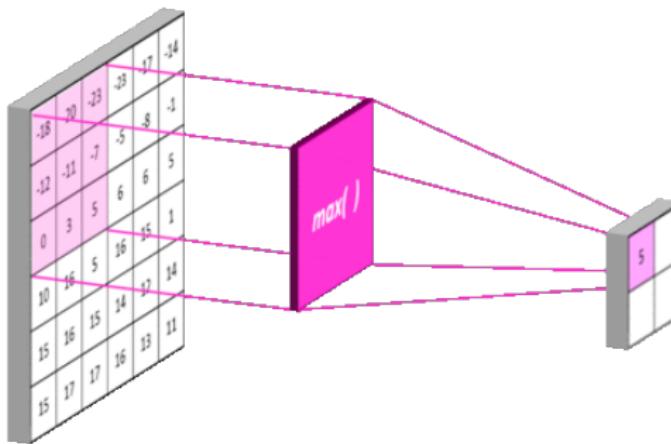
Voir site web: <https://deeplizard.com/resource/pavq7noze3>

COUCHE DE POOLING (POOLING LAYER)

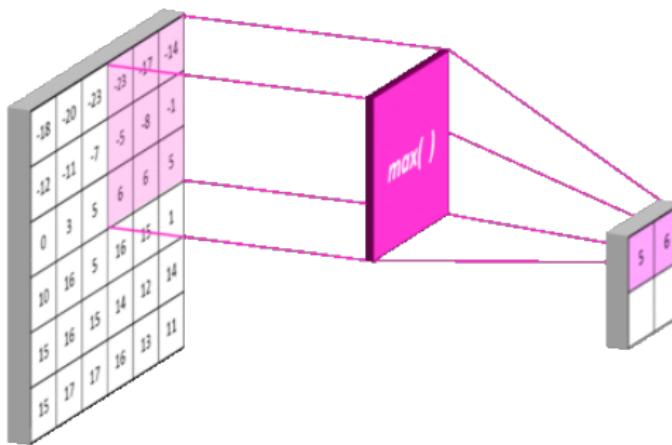


Figures taken from [Fleuret, 2022].

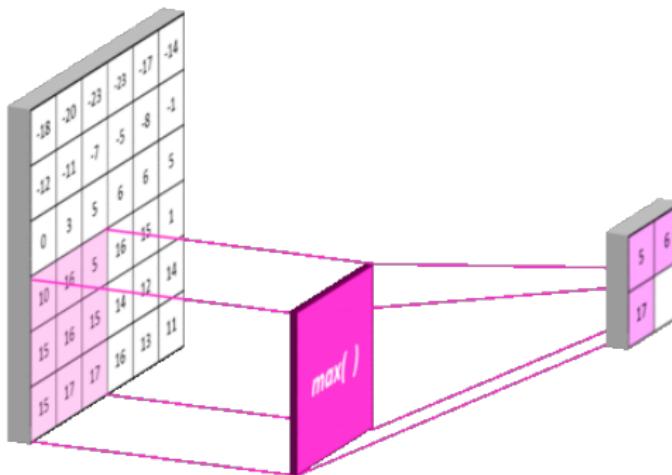
COUCHE DE POOLING (POOLING LAYER)



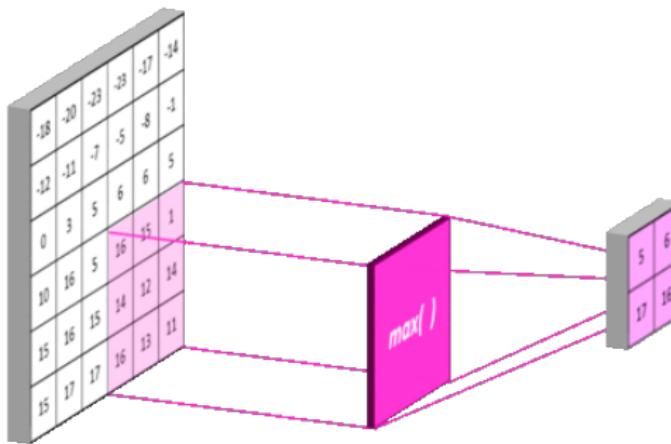
COUCHE DE POOLING (POOLING LAYER)



COUCHE DE POOLING (POOLING LAYER)



COUCHE DE POOLING (POOLING LAYER)



COUCHE DE POOLING (POOLING LAYER)

- ▶ Une couche de pooling est appelée *pooling layer*.
- ▶ Les deux principaux types de pooling sont le *max pooling* et le *average pooling*, mais il en existe d'autres.
- ▶ Une couche de pooling ne fait intervenir aucun paramètre; il n'y a donc aucun paramètre à apprendre!

COUCHE DE POOLING (POOLING LAYER)

- ▶ Une couche de pooling est appelée *pooling layer*.
- ▶ Les deux principaux types de pooling sont le *max pooling* et le *average pooling*, mais il en existe d'autres.
- ▶ Une couche de pooling ne fait intervenir aucun paramètre; il n'y a donc aucun paramètre à apprendre!

COUCHE DE POOLING (POOLING LAYER)

- ▶ Une couche de pooling est appelée *pooling layer*.
- ▶ Les deux principaux types de pooling sont le *max pooling* et le *average pooling*, mais il en existe d'autres.
- ▶ **Une couche de pooling ne fait intervenir aucun paramètre; il n'y a donc aucun paramètre à apprendre!**

COUCHE DE POOLING (POOLING LAYER)

- ▶ Le max ou avg pooling appliqué à une input \mathbf{X} de dimension $H \times W \times C$ par le biais d'un filtre de dimension $h \times w$ donne une output

$$\mathbf{Z} = \text{MaxPool}(\mathbf{X}) \text{ ou } \mathbf{Z} = \text{AvgPool}(\mathbf{X})$$

de dimension $(H - h + 1) \times (W - w + 1) \times C$, respectivement.

- ▶ Cette fois-ci, l'opération de pooling est appliquée canal par canal. Ainsi, l'output est de même profondeur C que l'input (cf. slide suivant)!

COUCHE DE POOLING (POOLING LAYER)

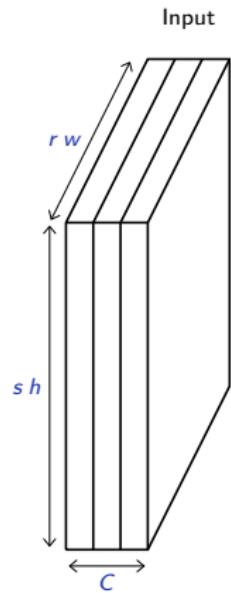
- ▶ Le max ou avg pooling appliqué à une input \mathbf{X} de dimension $H \times W \times C$ par le biais d'un filtre de dimension $h \times w$ donne une output

$$\mathbf{Z} = \text{MaxPool}(\mathbf{X}) \text{ ou } \mathbf{Z} = \text{AvgPool}(\mathbf{X})$$

de dimension $(H - h + 1) \times (W - w + 1) \times C$, respectivement.

- ▶ Cette fois-ci, l'opération de pooling est appliquée canal par canal. Ainsi, l'output est de même profondeur C que l'input (cf. slide suivant)!

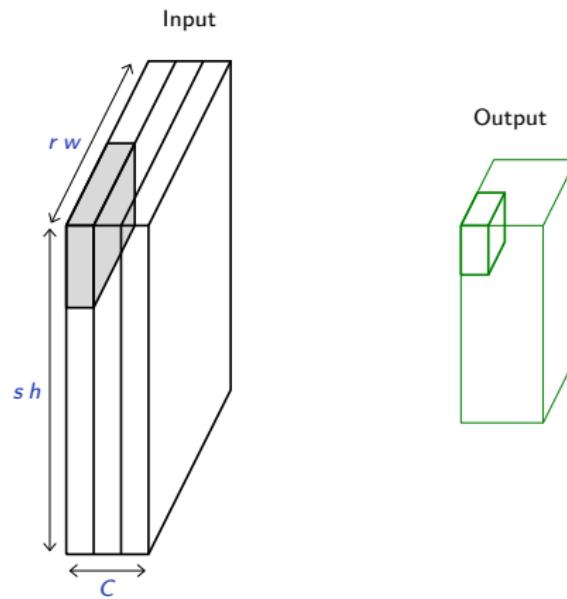
COUCHE DE POOLING (POOLING LAYER)



Figures taken from [Fleuret, 2022]

- ▶ Contrairement à la convolution, le pooling est appliqué indépendamment sur chaque canal, donnant une output de profondeur C .

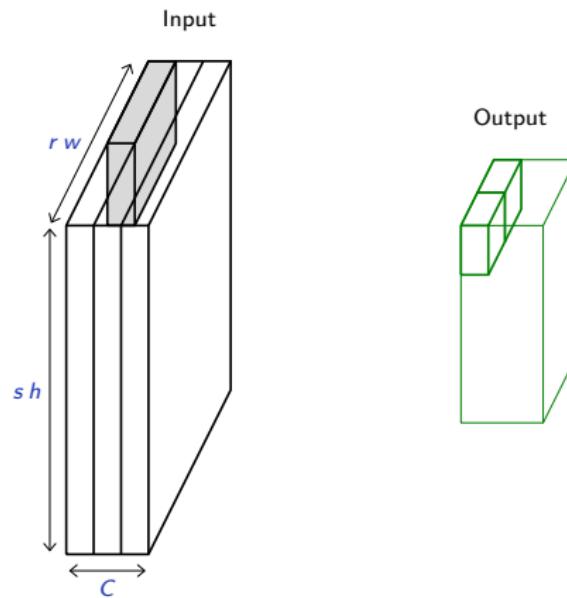
COUCHE DE POOLING (POOLING LAYER)



Figures taken from [Fleuret, 2022]

- ▶ Contrairement à la convolution, le pooling est appliqué indépendamment sur chaque canal, donnant une output de profondeur C .

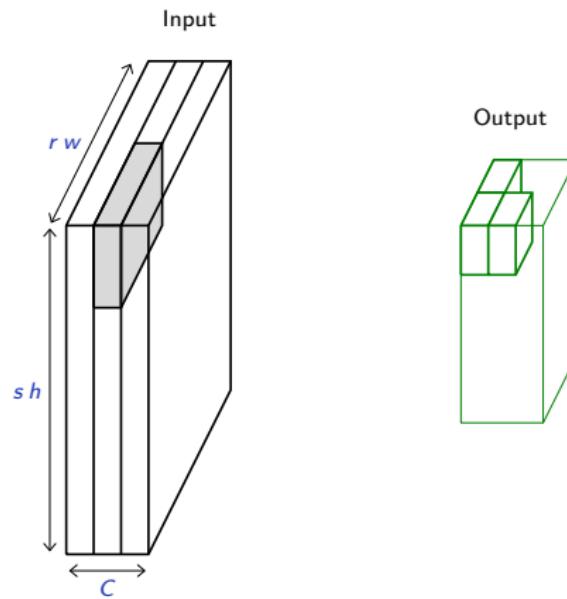
COUCHE DE POOLING (POOLING LAYER)



Figures taken from [Fleuret, 2022]

- ▶ Contrairement à la convolution, le pooling est appliqué indépendamment sur chaque canal, donnant une output de profondeur C .

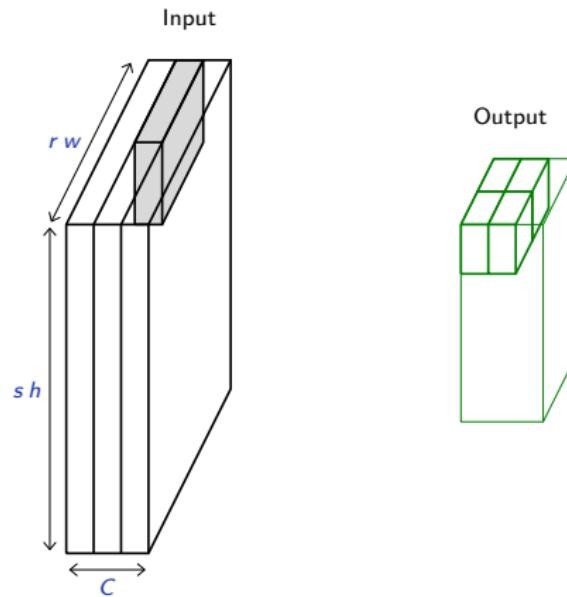
COUCHE DE POOLING (POOLING LAYER)



Figures taken from [Fleuret, 2022]

- ▶ Contrairement à la convolution, le pooling est appliqué indépendamment sur chaque canal, donnant une output de profondeur C .

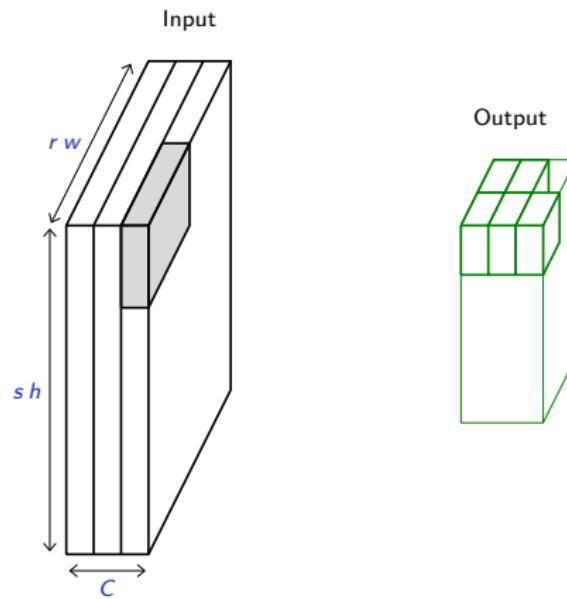
COUCHE DE POOLING (POOLING LAYER)



Figures taken from [Fleuret, 2022]

- ▶ Contrairement à la convolution, le pooling est appliqué indépendamment sur chaque canal, donnant une output de profondeur C .

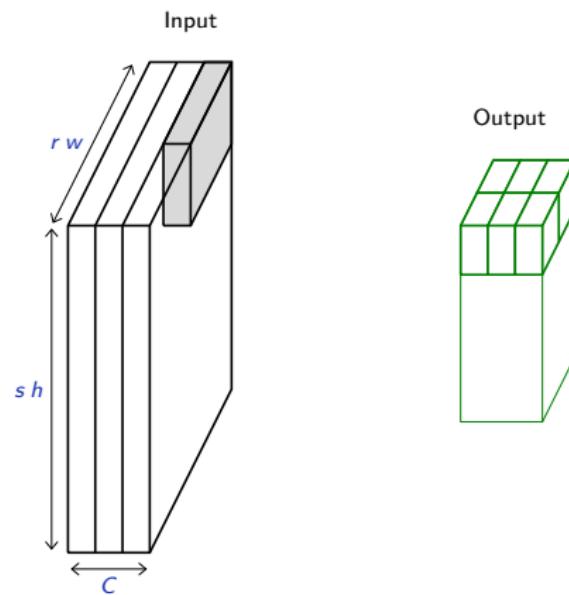
COUCHE DE POOLING (POOLING LAYER)



Figures taken from [\[Fleuret, 2022\]](#)

- ▶ Contrairement à la convolution, le pooling est appliqué indépendamment sur chaque canal, donnant une output de profondeur C .

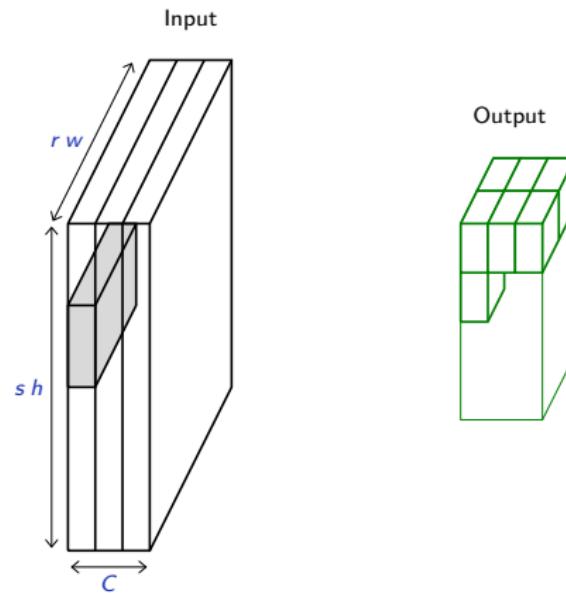
COUCHE DE POOLING (POOLING LAYER)



Figures taken from [Fleuret, 2022]

- ▶ Contrairement à la convolution, le pooling est appliqué indépendamment sur chaque canal, donnant une output de profondeur C .

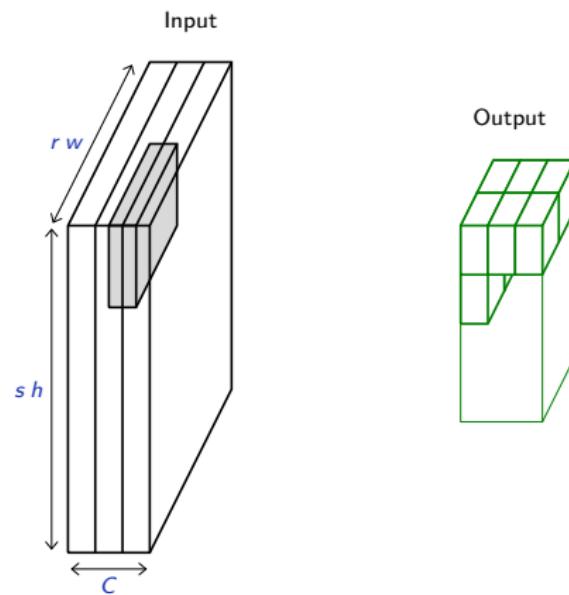
COUCHE DE POOLING (POOLING LAYER)



Figures taken from [Fleuret, 2022]

- ▶ Contrairement à la convolution, le pooling est appliqué indépendamment sur chaque canal, donnant une output de profondeur C .

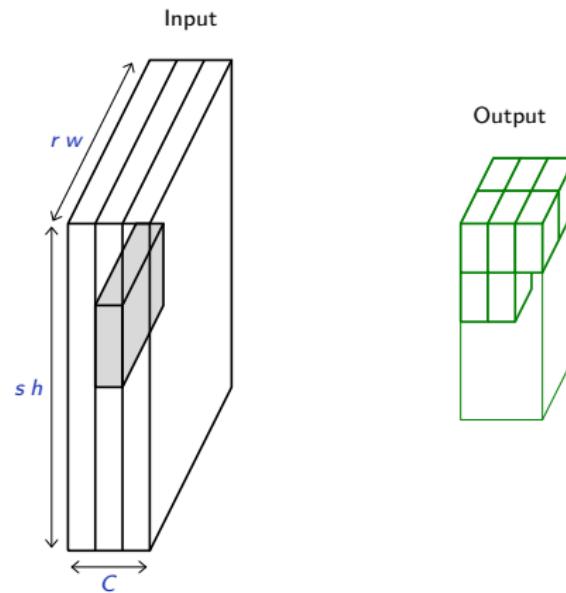
COUCHE DE POOLING (POOLING LAYER)



Figures taken from [Fleuret, 2022]

- ▶ Contrairement à la convolution, le pooling est appliqué indépendamment sur chaque canal, donnant une output de profondeur C .

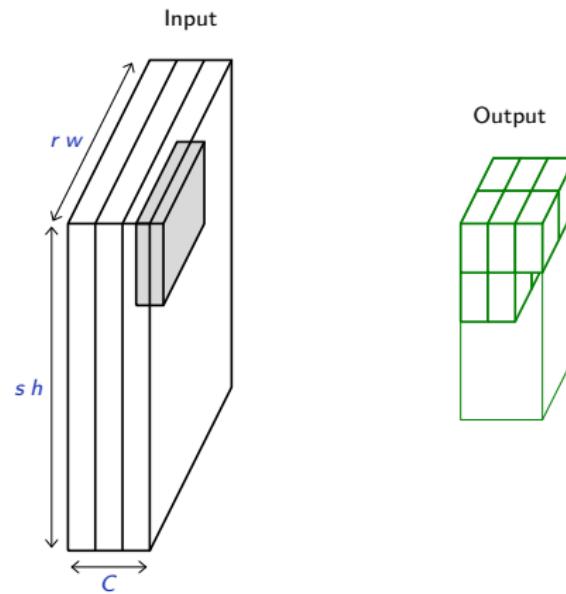
COUCHE DE POOLING (POOLING LAYER)



Figures taken from [Fleuret, 2022]

- ▶ Contrairement à la convolution, le pooling est appliqué indépendamment sur chaque canal, donnant une output de profondeur C .

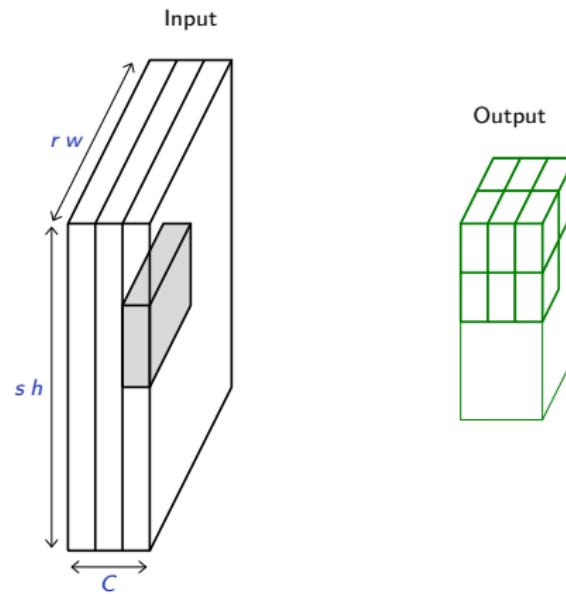
COUCHE DE POOLING (POOLING LAYER)



Figures taken from [Fleuret, 2022]

- ▶ Contrairement à la convolution, le pooling est appliqué indépendamment sur chaque canal, donnant une output de profondeur C .

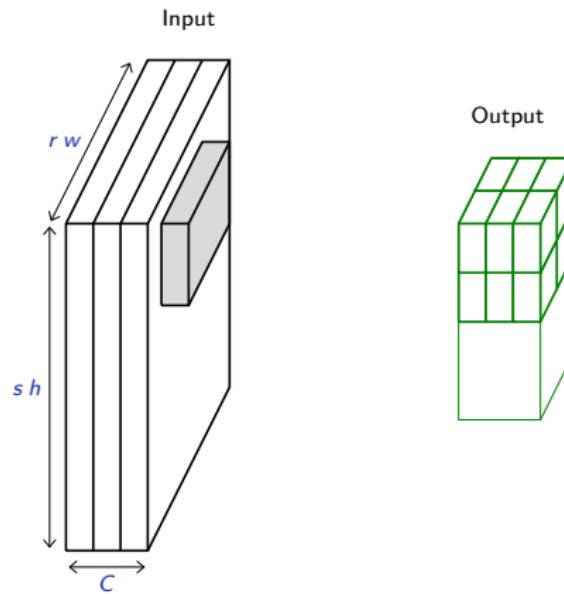
COUCHE DE POOLING (POOLING LAYER)



Figures taken from [Fleuret, 2022]

- ▶ Contrairement à la convolution, le pooling est appliqué indépendamment sur chaque canal, donnant une output de profondeur C .

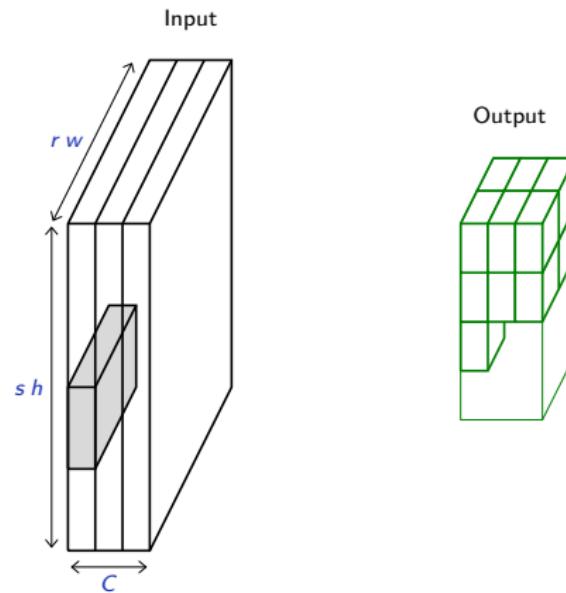
COUCHE DE POOLING (POOLING LAYER)



Figures taken from [Fleuret, 2022]

- ▶ Contrairement à la convolution, le pooling est appliqué indépendamment sur chaque canal, donnant une output de profondeur C .

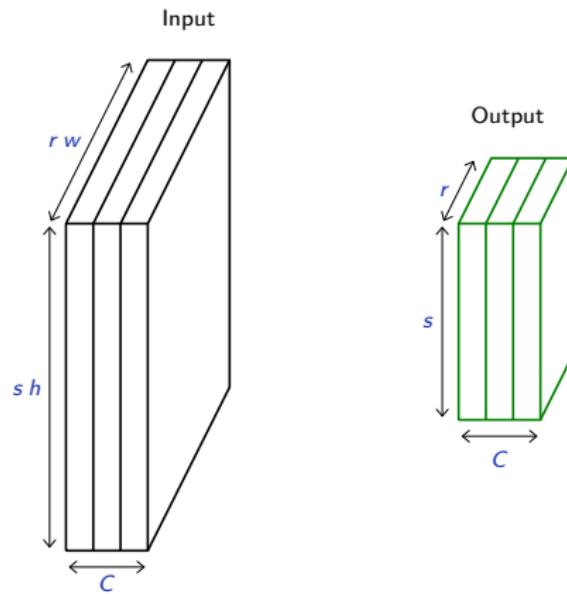
COUCHE DE POOLING (POOLING LAYER)



Figures taken from [Fleuret, 2022]

- ▶ Contrairement à la convolution, le pooling est appliqué indépendamment sur chaque canal, donnant une output de profondeur C .

COUCHE DE POOLING (POOLING LAYER)



Figures taken from [Fleuret, 2022]

- ▶ Contrairement à la convolution, le pooling est appliqué indépendamment sur chaque canal, donnant une output de profondeur C .

RÔLE DES COUCHES POOLING

Les couches de pooling jouent les rôle suivants:

- ▶ **Aggregate information**

Un groupe de pixels est agrégé en un seul pixel d'information.

- ▶ **Dimension reduction**

Une couche de pooling réduit la dimension de l'input, et donc l'efficacité computationnelle.

- ▶ **Produce translation invariance**

Si l'input (ou des patterns de l'input) sont légèrement translatés, leur "pooled representation" restera très similaire.

RÔLE DES COUCHES POOLING

Les couches de pooling jouent les rôle suivants:

- ▶ **Aggregate information**

Un groupe de pixels est agrégé en un seul pixel d'information.

- ▶ **Dimension reduction**

Une couche de pooling réduit la dimension de l'input, et donc l'efficacité computationnelle.

- ▶ **Produce translation invariance**

Si l'input (ou des patterns de l'input) sont légèrement translatés, leur "pooled representation" restera très similaire.

RÔLE DES COUCHES POOLING

Les couches de pooling jouent les rôle suivants:

- ▶ **Aggregate information**

Un groupe de pixels est agrégé en un seul pixel d'information.

- ▶ **Dimension reduction**

Une couche de pooling réduit la dimension de l'input, et donc l'efficacité computationnelle.

- ▶ **Produce translation invariance**

Si l'input (ou des patterns de l'input) sont légèrement translatés, leur “pooled representation” restera très similaire.

RÔLE DES COUCHES POOLING

Input

Output

RÉSEAU DE NEURONES CONVOLUTIF (CNNs)

- ▶ **Un réseau de neurones convolutif (CNN) est composé de couches convolutives, de couches pooling et en général de quelques dernières couches denses classiques.**
- ▶ Les paramètres (poids) des filtres convolutifs (et des couches denses classiques s'il y en a) sont appris par backpropagation!
- ▶ Les couches de pooling ne font intervenir aucun paramètre!
- ▶ Les filtres sont des *features' extractors* et les features qu'ils extraient sont appris par entraînement.
- ▶ **Un CNN crée une représentation enrichie de l'image de départ (input) qui est en adéquation avec la tâche que l'on souhaite résoudre.**

RÉSEAU DE NEURONES CONVOLUTIF (CNNs)

- ▶ Un **réseau de neurones convolutif (CNN)** est composé de couches convolutives, de couches pooling et en général de quelques dernières couches denses classiques.
- ▶ Les paramètres (poids) des filtres convolutifs (et des couches denses classiques s'il y en a) sont appris par backpropagation!
- ▶ Les couches de pooling ne font intervenir aucun paramètre!
- ▶ Les filtres sont des *features' extractors* et les features qu'ils extraient sont appris par entraînement.
- ▶ Un CNN crée une représentation enrichie de l'image de départ (input) qui est en adéquation avec la tâche que l'on souhaite résoudre.

RÉSEAU DE NEURONES CONVOLUTIF (CNNs)

- ▶ Un **réseau de neurones convolutif (CNN)** est composé de couches convolutives, de couches pooling et en général de quelques dernières couches denses classiques.
- ▶ Les paramètres (poids) des filtres convolutifs (et des couches denses classiques s'il y en a) sont appris par backpropagation!
- ▶ Les couches de pooling ne font intervenir aucun paramètre!
- ▶ Les filtres sont des *features' extractors* et les features qu'ils extraient sont appris par entraînement.
- ▶ Un CNN crée une représentation enrichie de l'image de départ (input) qui est en adéquation avec la tâche que l'on souhaite résoudre.

RÉSEAU DE NEURONES CONVOLUTIF (CNNs)

- ▶ Un **réseau de neurones convolutif (CNN)** est composé de couches convolutives, de couches pooling et en général de quelques dernières couches denses classiques.
- ▶ Les paramètres (poids) des filtres convolutifs (et des couches denses classiques s'il y en a) sont appris par backpropagation!
- ▶ Les couches de pooling ne font intervenir aucun paramètre!
- ▶ Les filtres sont des *features' extractors* et les features qu'ils extraient sont appris par entraînement.
- ▶ Un CNN crée une représentation enrichie de l'image de départ (input) qui est en adéquation avec la tâche que l'on souhaite résoudre.

RÉSEAU DE NEURONES CONVOLUTIF (CNNs)

- ▶ Un **réseau de neurones convolutif (CNN)** est composé de couches convolutives, de couches pooling et en général de quelques dernières couches denses classiques.
- ▶ Les paramètres (poids) des filtres convolutifs (et des couches denses classiques s'il y en a) sont appris par backpropagation!
- ▶ Les couches de pooling ne font intervenir aucun paramètre!
- ▶ Les filtres sont des *features' extractors* et les features qu'ils extraient sont appris par entraînement.
- ▶ Un CNN crée une représentation enrichie de l'image de départ (input) qui est en adéquation avec la tâche que l'on souhaite résoudre.

ALEXNET [Krizhevsky et al., 2012]

- ▶ L'architecture AlexNet a représenté une avancée majeure dans le domaine de la computer vision.
- ▶ Le modèle se compose 5 couche convolutives, 2 couches de Max Pooling et 3 couche denses.
- ▶ Ce réseau a été entraîné pour un problème de classification d'images sur le dataset ImageNet (15M d'images, 22K catégories).
- ▶ AlexNet s'entraîne par backpropagation!

ALEXNET [Krizhevsky et al., 2012]

- ▶ L'architecture AlexNet a représenté une avancée majeure dans le domaine de la computer vision.
- ▶ Le modèle se compose 5 couche convolutives, 2 couches de Max Pooling et 3 couche denses.
- ▶ Ce réseau a été entraîné pour un problème de classification d'images sur le dataset ImageNet (15M d'images, 22K catégories).
- ▶ AlexNet s'entraîne par backpropagation!

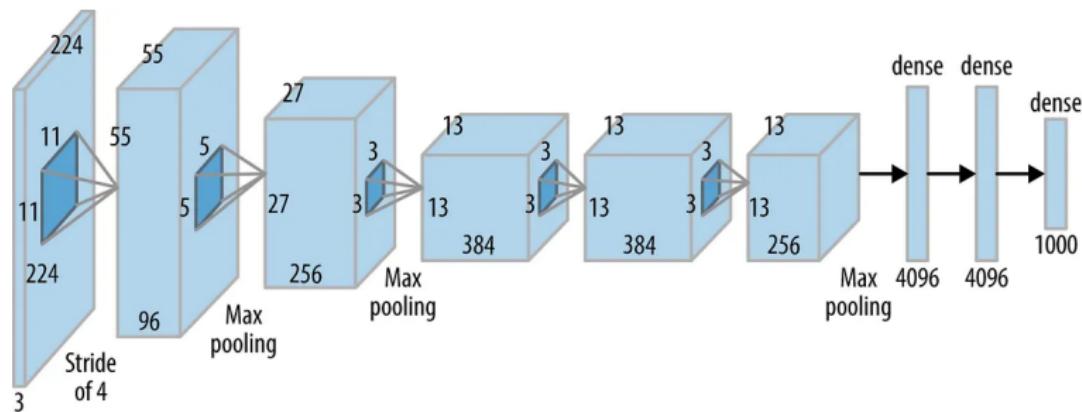
ALEXNET [Krizhevsky et al., 2012]

- ▶ L'architecture AlexNet a représenté une avancée majeure dans le domaine de la computer vision.
- ▶ Le modèle se compose 5 couche convolutives, 2 couches de Max Pooling et 3 couche denses.
- ▶ Ce réseau a été entraîné pour un problème de classification d'images sur le dataset ImageNet (15M d'images, 22K catégories).
- ▶ AlexNet s'entraîne par backpropagation!

ALEXNET [Krizhevsky et al., 2012]

- ▶ L'architecture AlexNet a représenté une avancée majeure dans le domaine de la computer vision.
- ▶ Le modèle se compose 5 couche convolutives, 2 couches de Max Pooling et 3 couche denses.
- ▶ Ce réseau a été entraîné pour un problème de classification d'images sur le dataset ImageNet (15M d'images, 22K catégories).
- ▶ AlexNet s'entraîne par backpropagation!

ALEXNET



Figures taken from

<https://towardsdatascience.com/the-whh-of-alexnet-vggnet-resnet-and-inception-7baaaecc96>

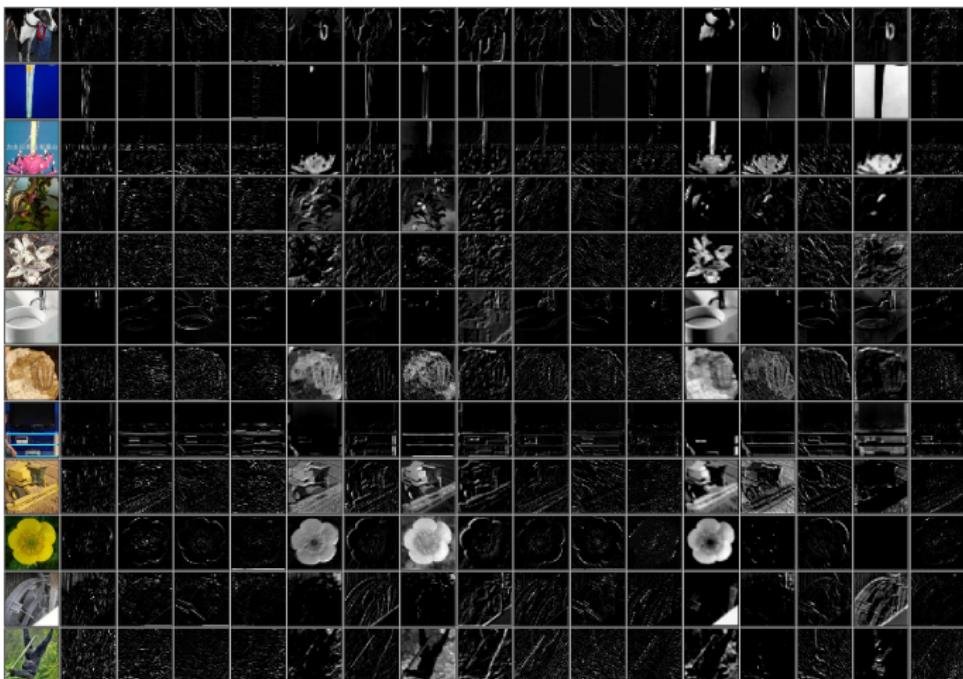
ALEXNET

AlexNet Network - Structural Details													
Input		Output			Layer	Stride	Pad	Kernel size	in	out	# of Param		
227	227	3	55	55	96	conv1	4	0	11	11	3	96	34944
55	55	96	27	27	96	maxpool1	2	0	3	3	96	96	0
27	27	96	27	27	256	conv2	1	2	5	5	96	256	614656
27	27	256	13	13	256	maxpool2	2	0	3	3	256	256	0
13	13	256	13	13	384	conv3	1	1	3	3	256	384	885120
13	13	384	13	13	384	conv4	1	1	3	3	384	384	1327488
13	13	384	13	13	256	conv5	1	1	3	3	384	256	884992
13	13	256	6	6	256	maxpool5	2	0	3	3	256	256	0
					fc6			1	1	9216	4096	37752832	
					fc7			1	1	4096	4096	16781312	
					fc8			1	1	4096	1000	4097000	
Total											62,378,344		

Figures taken from

<https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecc96>

ALEXNET: CONVOLUTION AND POOLING LAYERS

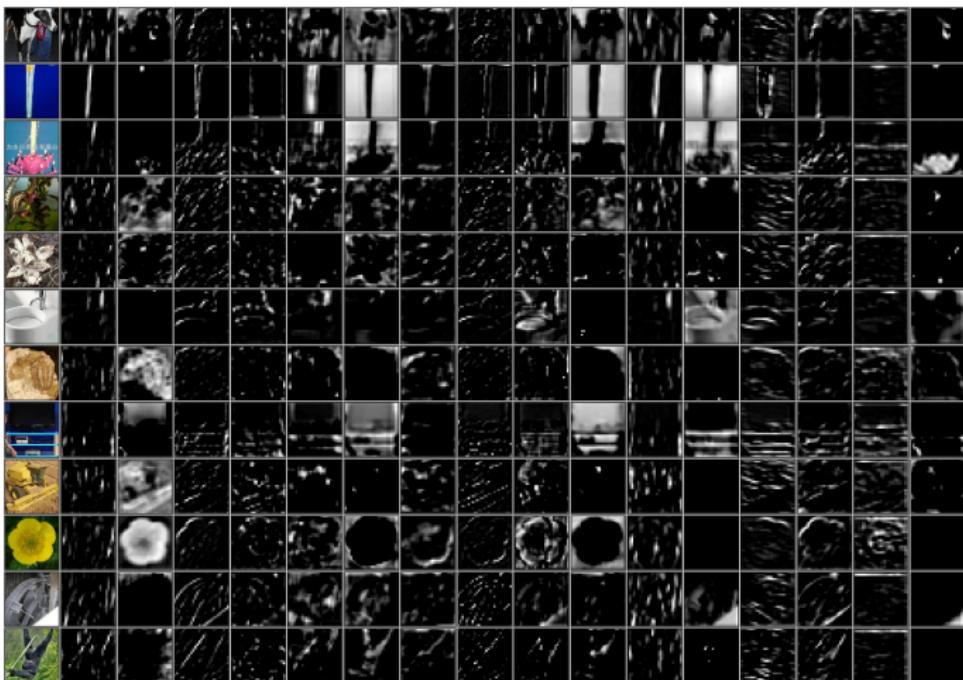


Activation maps (16 random channels) of convolution layers (1-5).

Activation maps (16 random channels) of max pooling layers (1-3).

Figures taken from [Krizhevsky et al., 2012].

ALEXNET: CONVOLUTION AND POOLING LAYERS

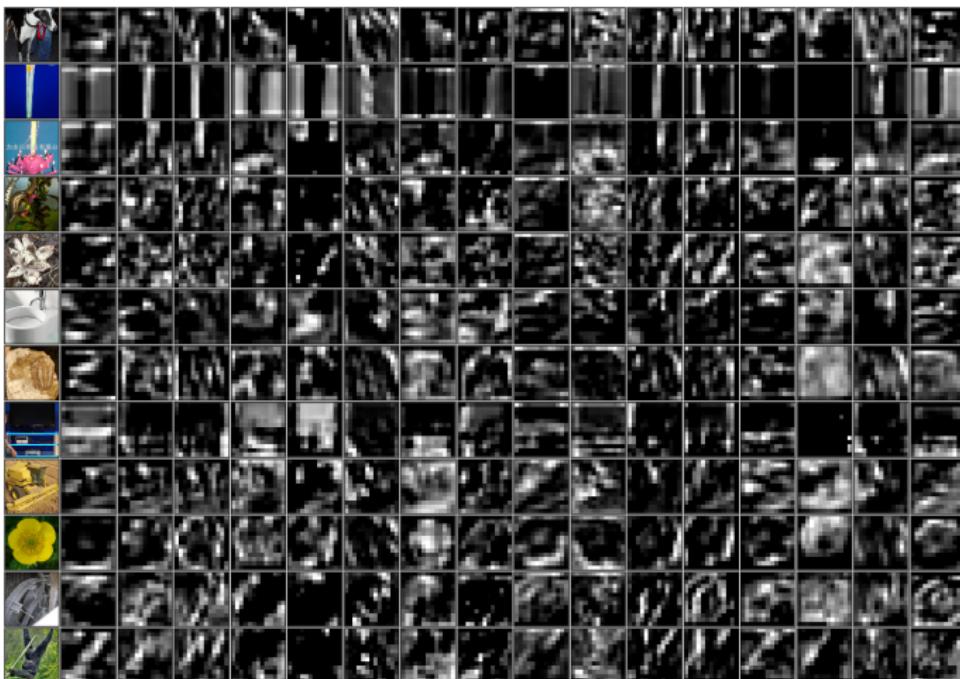


Activation maps (16 random channels) of convolution layers (1–5).

Activation maps (16 random channels) of max pooling layers (1–3).

Figures taken from [Krizhevsky et al., 2012].

ALEXNET: CONVOLUTION AND POOLING LAYERS

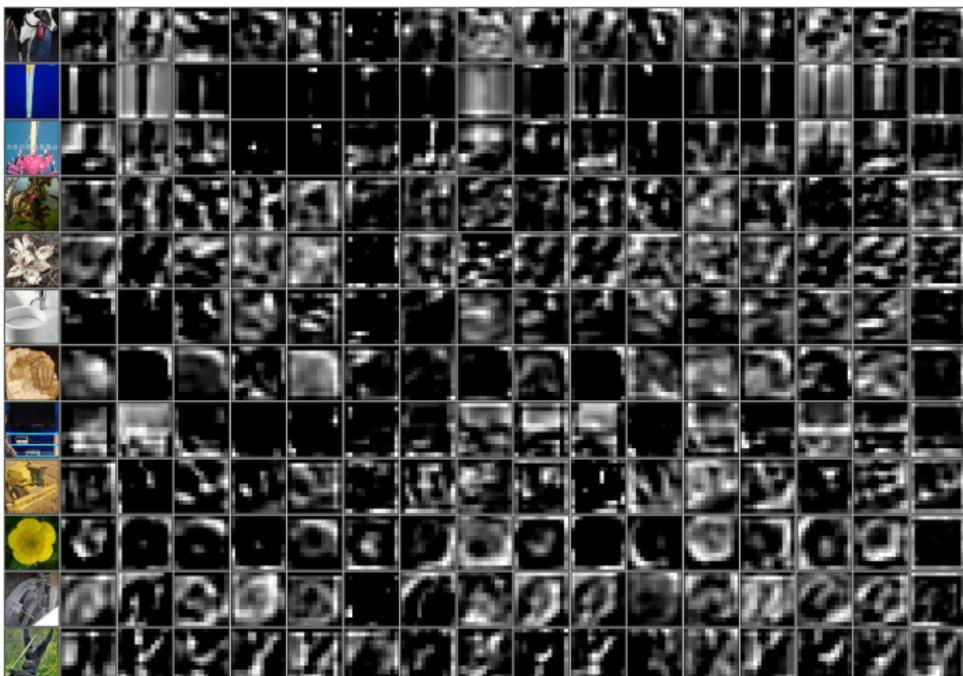


Activation maps (16 random channels) of convolution layers (1–5).

Activation maps (16 random channels) of max pooling layers (1–3).

Figures taken from [Krizhevsky et al., 2012].

ALEXNET: CONVOLUTION AND POOLING LAYERS

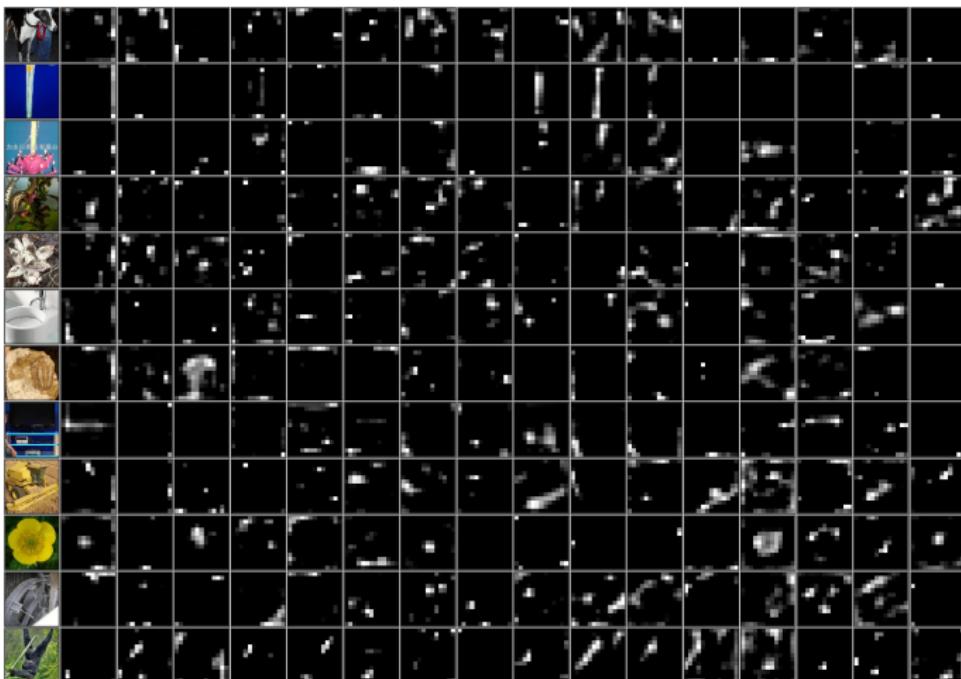


Activation maps (16 random channels) of convolution layers (1-5).

Activation maps (16 random channels) of max pooling layers (1-3).

Figures taken from [Krizhevsky et al., 2012].

ALEXNET: CONVOLUTION AND POOLING LAYERS

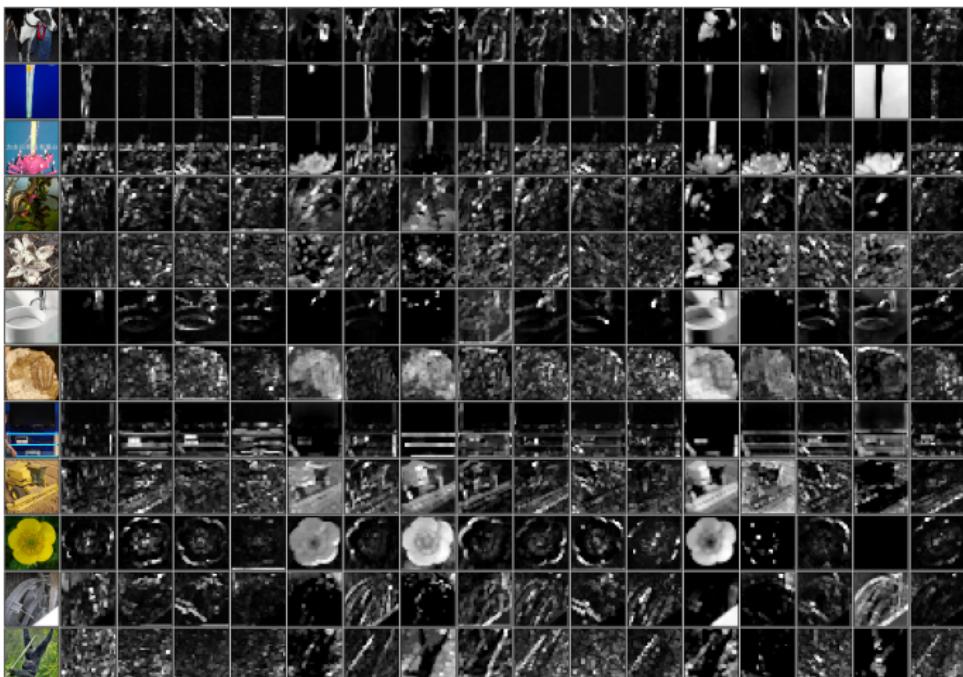


Activation maps (16 random channels) of convolution layers (1-5).

Activation maps (16 random channels) of max pooling layers (1-3).

Figures taken from [Krizhevsky et al., 2012].

ALEXNET: CONVOLUTION AND POOLING LAYERS

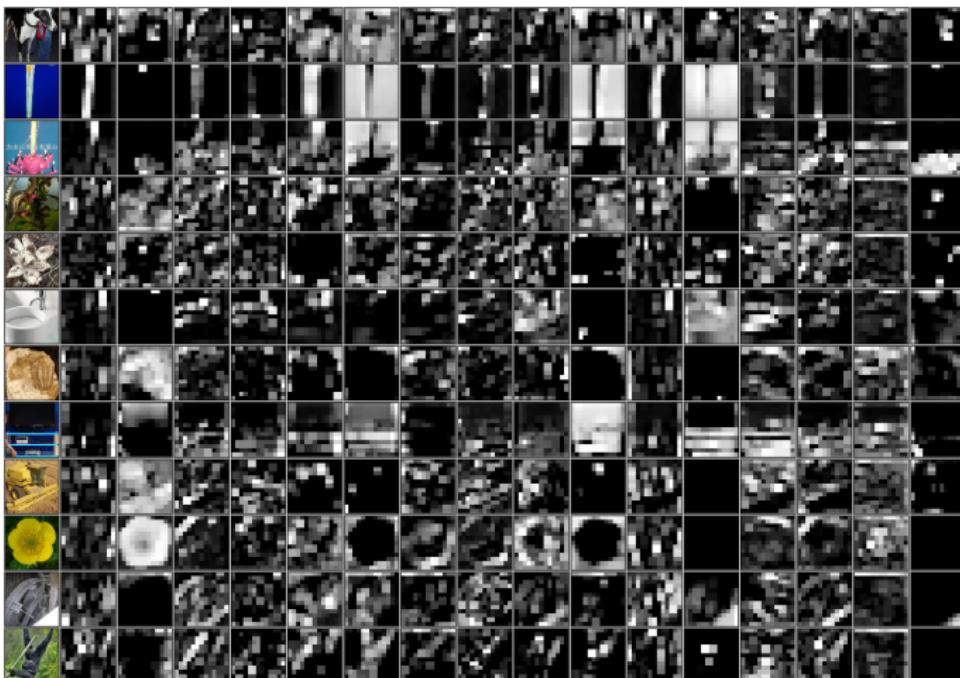


Activation maps (16 random channels) of convolution layers (1–5).

Activation maps (16 random channels) of max pooling layers (1–3).

Figures taken from [Krizhevsky et al., 2012].

ALEXNET: CONVOLUTION AND POOLING LAYERS

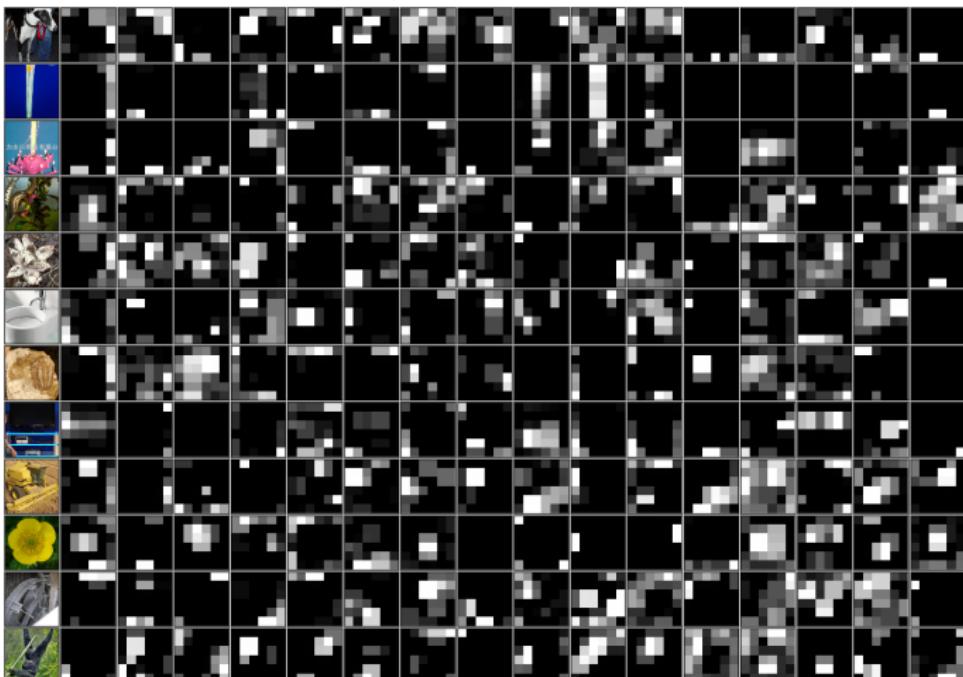


Activation maps (16 random channels) of convolution layers (1–5).

Activation maps (16 random channels) of max pooling layers (1–3).

Figures taken from [Krizhevsky et al., 2012].

ALEXNET: CONVOLUTION AND POOLING LAYERS



Activation maps (16 random channels) of convolution layers (1–5).

Activation maps (16 random channels) of max pooling layers (1–3).

Figures taken from [Krizhevsky et al., 2012].

VGGNET [SIMONYAN AND ZISSERMAN, 2015]

- ▶ L'architecture VGGNet a représenté une autre avancée.
- ▶ Le réseau se compose de 16 à 19 couches convolutives et de couches Max Pooling.
- ▶ Ce réseau a été également entraîné pour un problème de classification d'images sur le dataset ImageNet.
- ▶ Augmenter la profondeur du réseau et diminuer la taille des filtres améliore la performance.

VGGNET [SIMONYAN AND ZISSERMAN, 2015]

- ▶ L'architecture VGGNet a représenté une autre avancée.
- ▶ Le réseau se compose de 16 à 19 couches convolutives et de couches Max Pooling.
- ▶ Ce réseau a été également entraîné pour un problème de classification d'images sur le dataset ImageNet.
- ▶ Augmenter la profondeur du réseau et diminuer la taille des filtres améliore la performance.

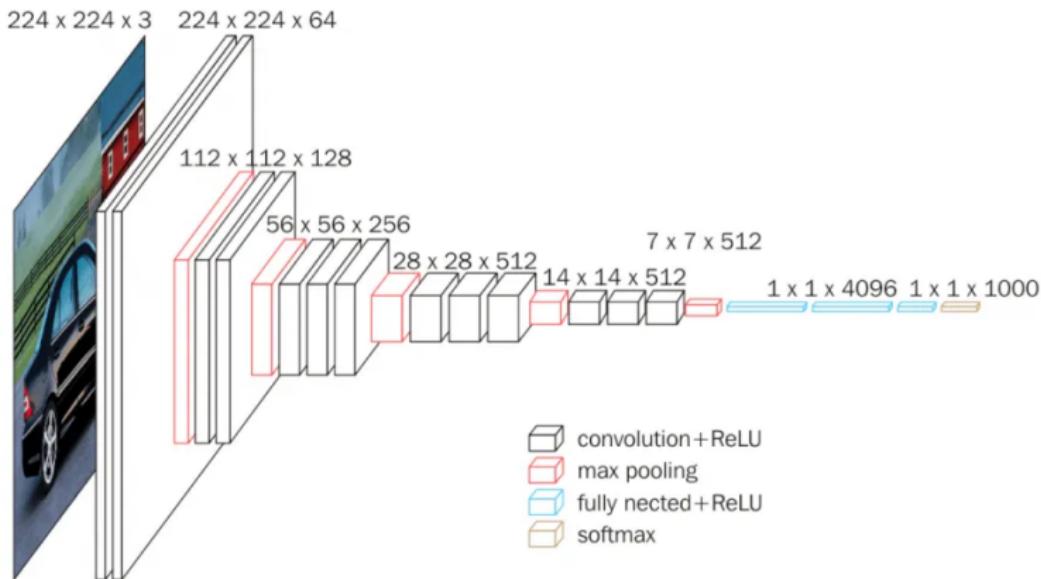
VGGNET [SIMONYAN AND ZISSERMAN, 2015]

- ▶ L'architecture VGGNet a représenté une autre avancée.
- ▶ Le réseau se compose de 16 à 19 couches convolutives et de couches Max Pooling.
- ▶ Ce réseau a été également entraîné pour un problème de classification d'images sur le dataset ImageNet.
- ▶ Augmenter la profondeur du réseau et diminuer la taille des filtres améliore la performance.

VGGNET [SIMONYAN AND ZISSERMAN, 2015]

- ▶ L'architecture VGGNet a représenté une autre avancée.
- ▶ Le réseau se compose de 16 à 19 couches convolutives et de couches Max Pooling.
- ▶ Ce réseau a été également entraîné pour un problème de classification d'images sur le dataset ImageNet.
- ▶ Augmenter la profondeur du réseau et diminuer la taille des filtres améliore la performance.

VGGNET



Figures taken from

<https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaeccccc96>

VGGNET

VGG16 - Structural Details														
#	Input Image			output			Layer	Stride	Kernel	in	out	Param		
1	224	224	3	224	224	64	conv3-64	1	3	3	3	64	1792	
2	224	224	64	224	224	64	conv3064	1	3	3	64	64	36928	
	224	224	64	112	112	64	maxpool	2	2	2	64	64	0	
3	112	112	64	112	112	128	conv3-128	1	3	3	64	128	73856	
4	112	112	128	112	112	128	conv3-128	1	3	3	128	128	147584	
	112	112	128	56	56	128	maxpool	2	2	2	128	128	65664	
5	56	56	128	56	56	256	conv3-256	1	3	3	128	256	295168	
6	56	56	256	56	56	256	conv3-256	1	3	3	256	256	590080	
7	56	56	256	56	56	256	conv3-256	1	3	3	256	256	590080	
	56	56	256	28	28	256	maxpool	2	2	2	256	256	0	
8	28	28	256	28	28	512	conv3-512	1	3	3	256	512	1180160	
9	28	28	512	28	28	512	conv3-512	1	3	3	512	512	2359808	
10	28	28	512	28	28	512	conv3-512	1	3	3	512	512	2359808	
	28	28	512	14	14	512	maxpool	2	2	2	512	512	0	
11	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808	
12	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808	
13	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808	
	14	14	512	7	7	512	maxpool	2	2	2	512	512	0	
14	1	1	25088	1	1	4096	fc			1	1	25088	4096	102764544
15	1	1	4096	1	1	4096	fc			1	1	4096	4096	16781312
16	1	1	4096	1	1	1000	fc			1	1	4096	1000	4097000
Total											138,423,208			

Figures taken from

<https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaeccccc96>

RESNET [HE ET AL., 2016]

- ▶ L'architecture ResNet (residual network) introduit le concept de *skip connections*.
- ▶ Les *skip connections* permettent d'apprendre si (et dans quelle mesure) les features extraites par les couches convolutives valent la peine d'être transférées à travers le réseau ou non.
- ▶ Les inputs de la couche $l + 1$ sont une combinaison linéaire des outputs $\mathcal{F}(x)$ de la couche l et des inputs x de la couche l .

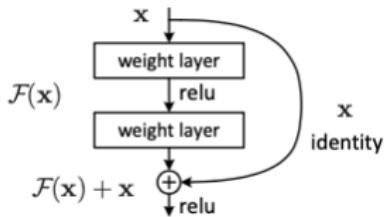


Figure taken from [He et al., 2016].

RESNET [HE ET AL., 2016]

- ▶ L'architecture ResNet (residual network) introduit le concept de *skip connections*.
- ▶ Les *skip connections* permettent d'apprendre si (et dans quelle mesure) les features extraites par les couches convolutives valent la peine d'être transférées à travers le réseau ou non.
- ▶ Les inputs de la couche $l + 1$ sont une combinaison linéaire des outputs $\mathcal{F}(x)$ de la couche l et des inputs x de la couche l .

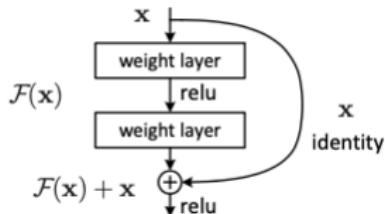


Figure taken from [He et al., 2016].

RESNET [HE ET AL., 2016]

- ▶ L'architecture ResNet (residual network) introduit le concept de *skip connections*.
- ▶ Les *skip connections* permettent d'apprendre si (et dans quelle mesure) les features extraites par les couches convolutives valent la peine d'être transférées à travers le réseau ou non.
- ▶ Les inputs de la couche $l + 1$ sont une combinaison linéaire des outputs $\mathcal{F}(x)$ de la couche l et des inputs x de la couche l .

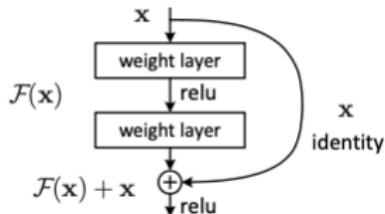


Figure taken from [He et al., 2016].

RESNET

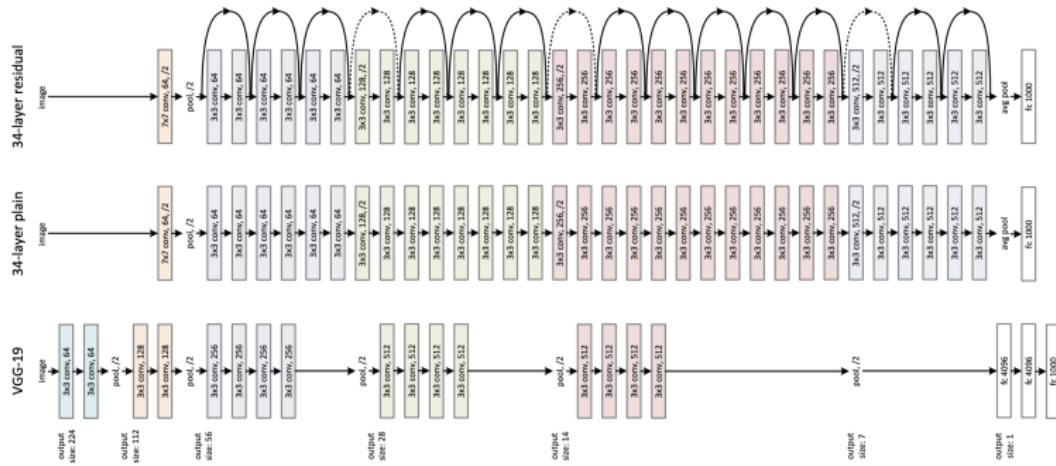


Figure taken from [He et al., 2016].

BIBLIOGRAPHIE I



Fleuret, F. (2022).
Deep Learning Course.



Fukushima, K. (1980).
Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position.
Biological Cybernetics, 36:193–202.



Fukushima, K. (1988).
Neocognitron: A hierarchical neural network capable of visual pattern recognition.
Neural Networks, 1(2):119–130.



He, K., Zhang, X., Ren, S., and Sun, J. (2016).
Deep residual learning for image recognition.
In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

BIBLIOGRAPHIE II



Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012).

Imagenet classification with deep convolutional neural networks.

In Bartlett, P. L., Pereira, F. C. N., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114.



LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998).

Gradient-based learning applied to document recognition.

Proc. IEEE, 86(11):2278–2324.



Ronneberger, O., Fischer, P., and Brox, T. (2015).

U-net: Convolutional networks for biomedical image segmentation.

In Navab, N., Hornegger, J., III, W. M. W., and Frangi, A. F., editors, *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer.

BIBLIOGRAPHIE III



Simonyan, K. and Zisserman, A. (2015).

Very deep convolutional networks for large-scale image recognition.

In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.