Convolutional Neural Networks – CNN/ConvNet

Vincent Havard vhavard@cesi.fr

CESI LINEACT

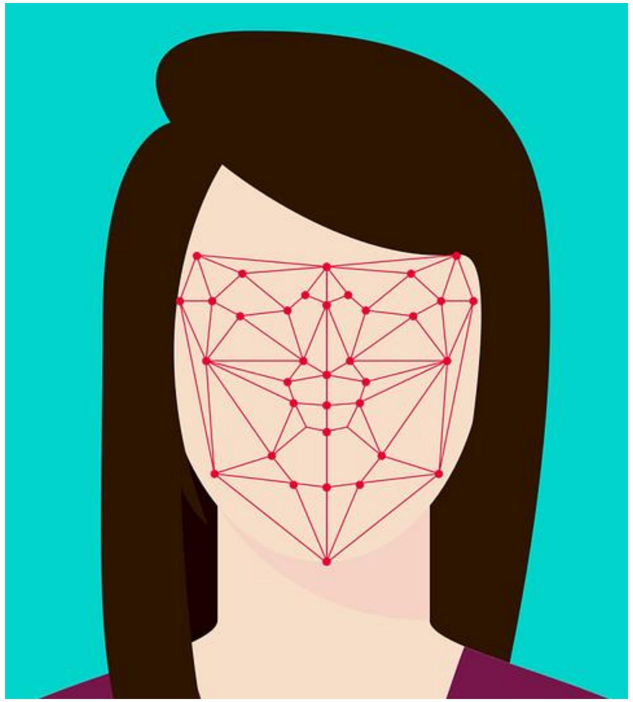
Sommaire

1. CNN/ConvNet overview
2. Why images ?
   Why Convolutional Neural Networks?
3. Datasets and challenges
4. Convolution on Volume
5. Simple example of ConvNet
6. Max Pooling
7. CNNs for classification
8. Real life ConvNet examples
9. Conclusion

CNN is used in several domains

CNN is about **computer vision** in every domain

Face detection & recognition

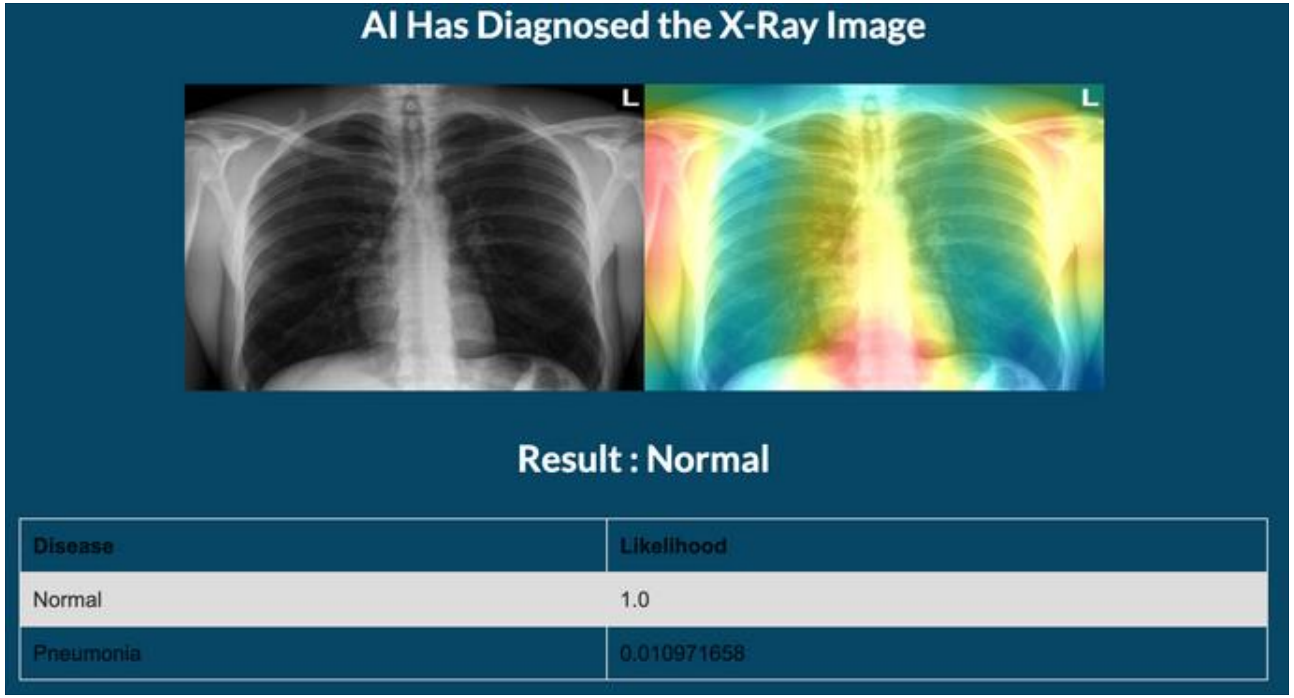


CESI LINEACT

CNN is used in several domains

CNN is about computer vision in every domain

Diagnosis



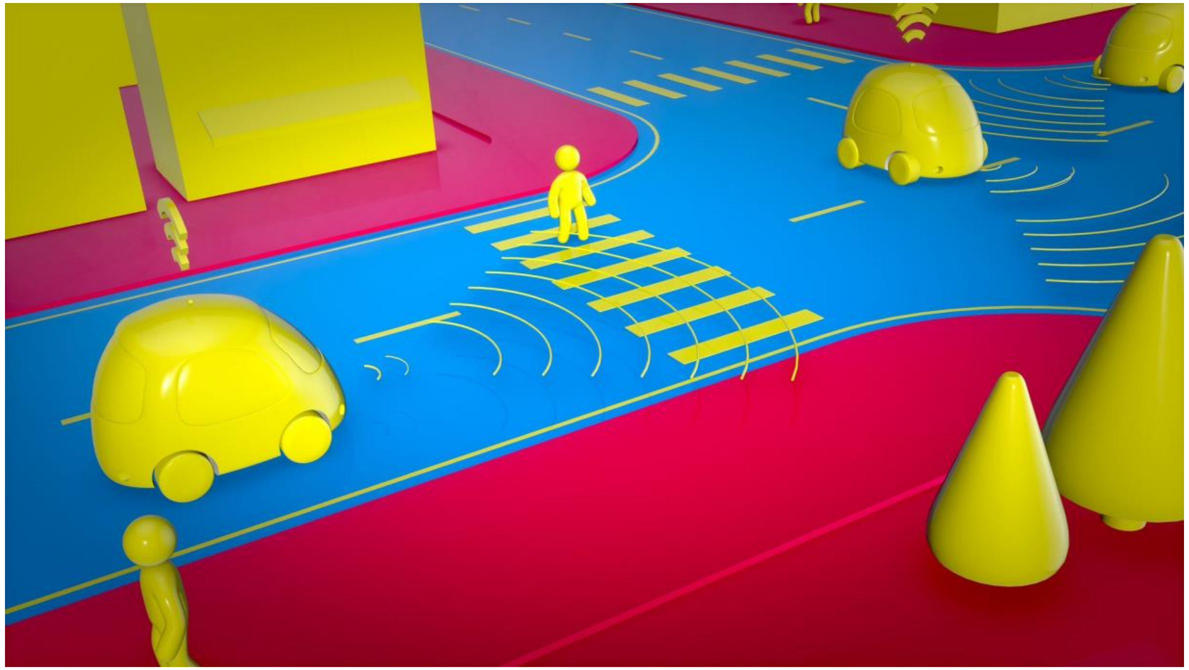
Disease	Likelihood
Normal	1.0
Pneumonia	0.010971658

Source: https://www.deepdiagnosis.tech/

CNN is used in several domains

CNN is about computer vision in every domain

Self driving car



CNN is used in several domains

CNN is about **Pose estimation**

Media Pipe

CESI LINEACT

CNN is used in several domains

CNN is about computer vision in every domain

Human Action Recognition

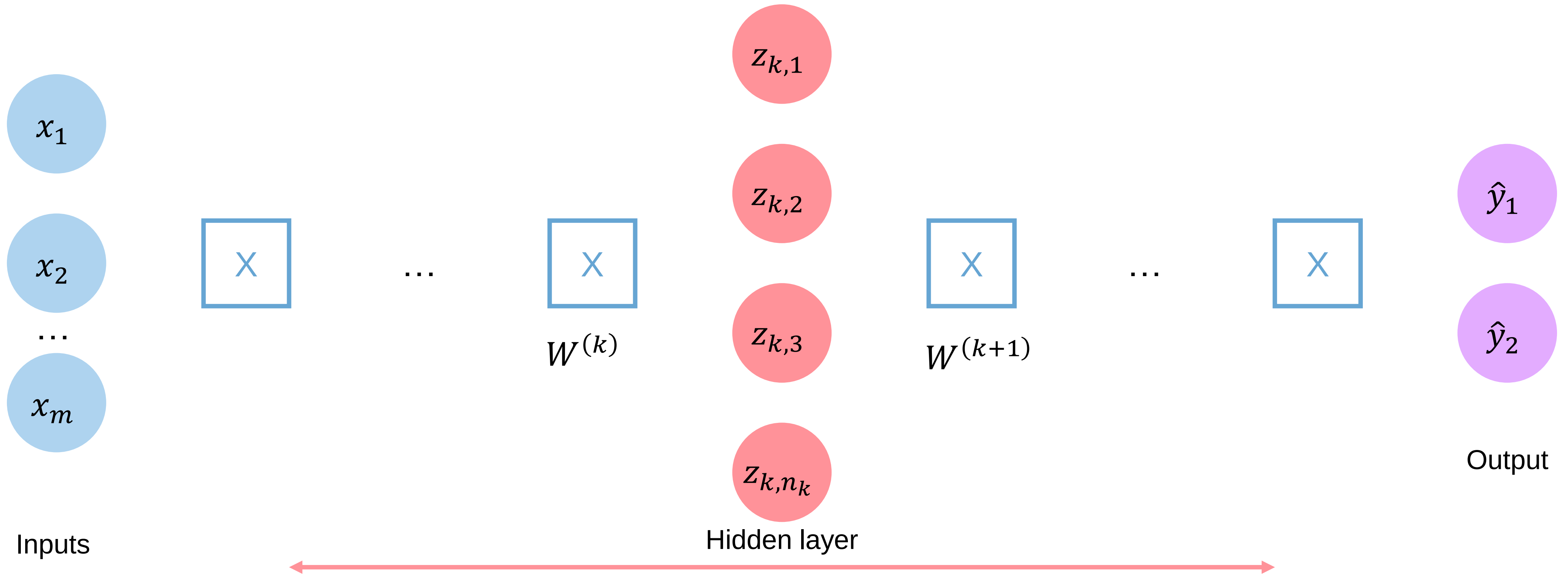
Segmentation d'une vidéo d'assemblage : **GT en vert** et **prédiction en orange**.

Benmessabih et al., 2023, CESI LINEACT

Sommaire

CESI LINEACT

Deep Neural Network

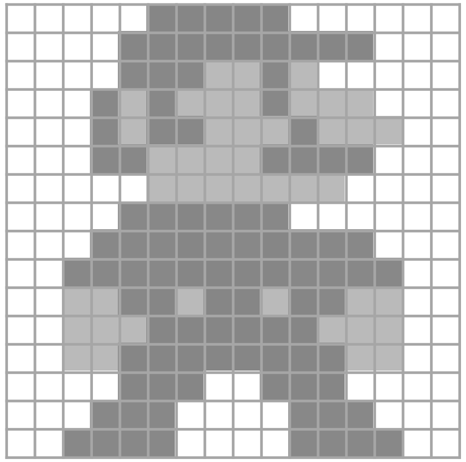


Fully connected networks has **many parameters** and **no spatial information**

Why CNN?

What is an image?

- A matrix of value

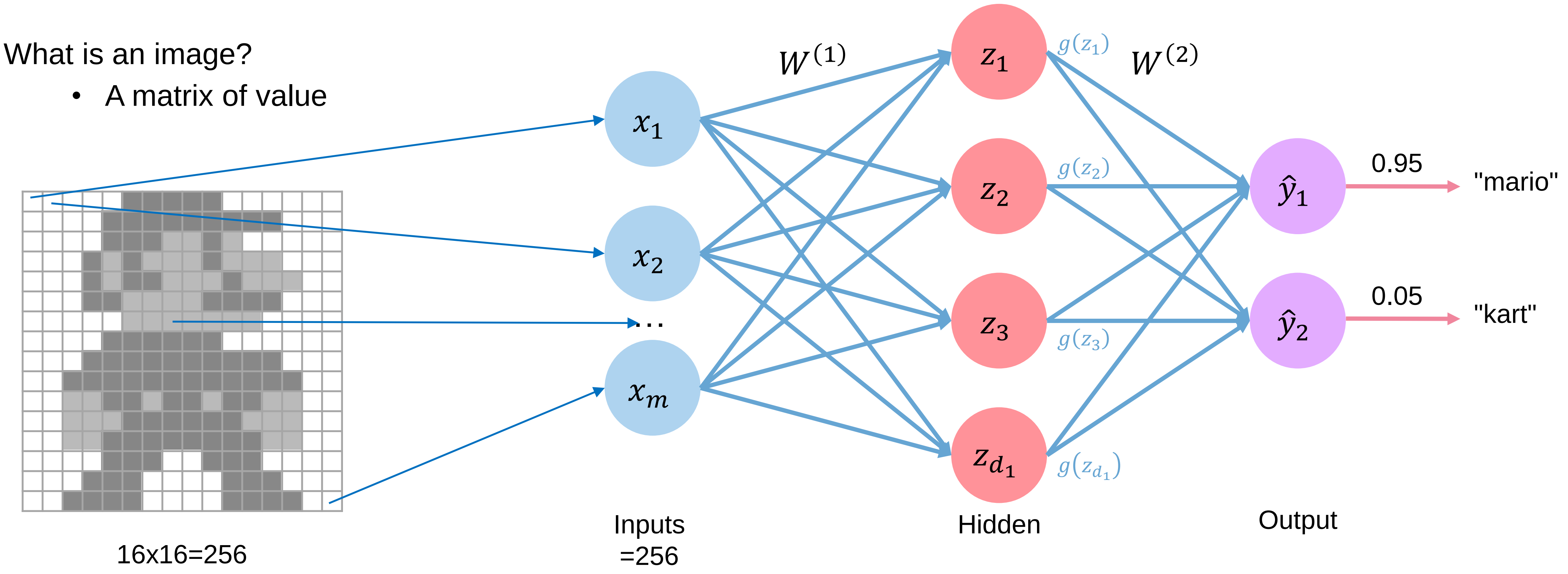


CESI LINEACT

Why CNN?

What is an image?

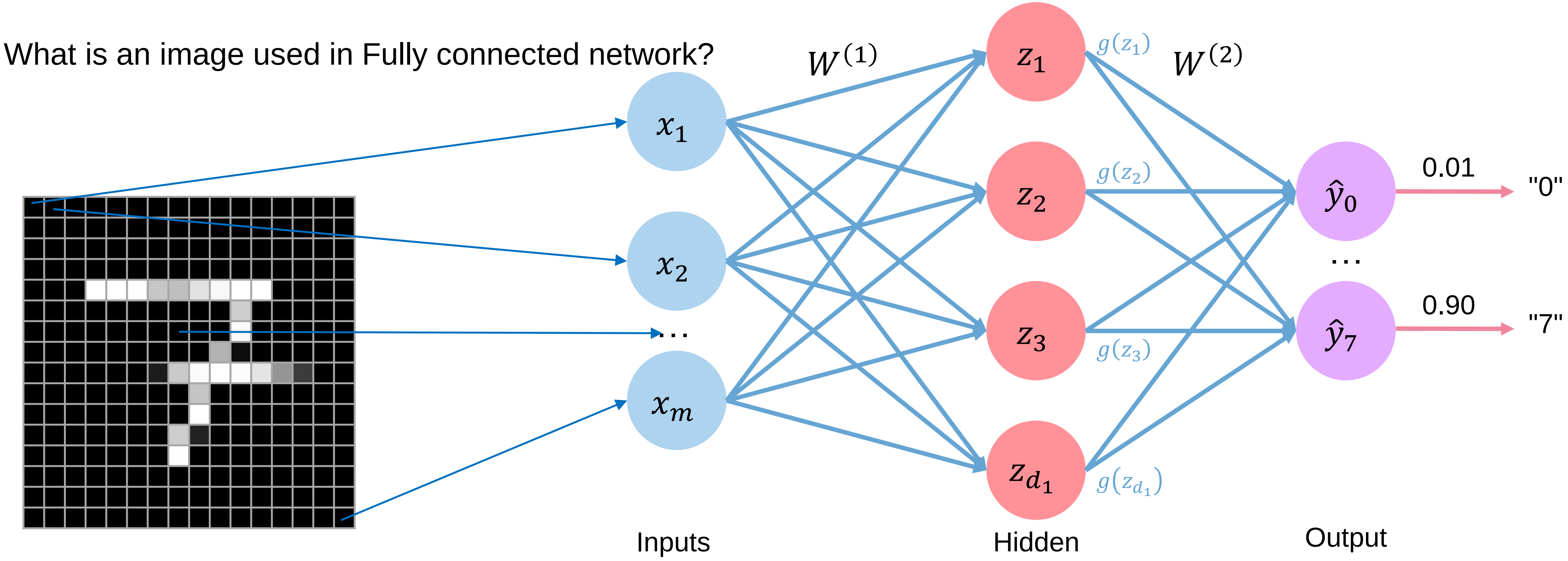
- A matrix of value



CESI LINEACT

Why CNN?

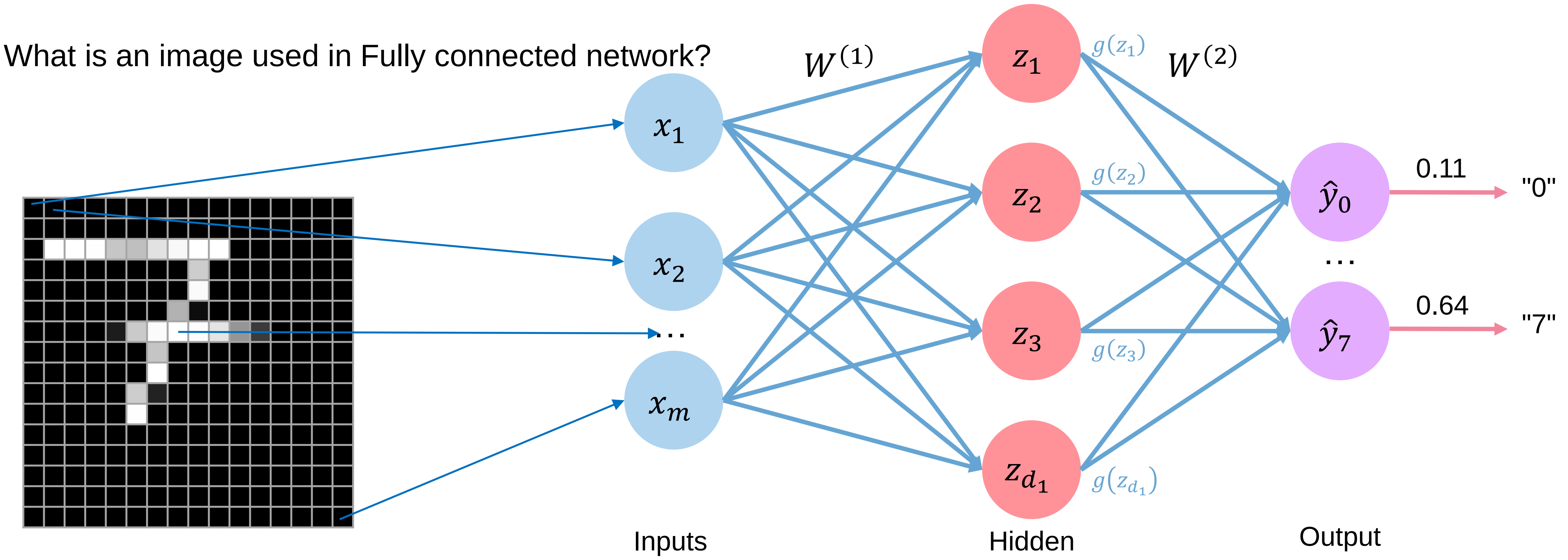
What is an image used in Fully connected network?



CESI LINEACT

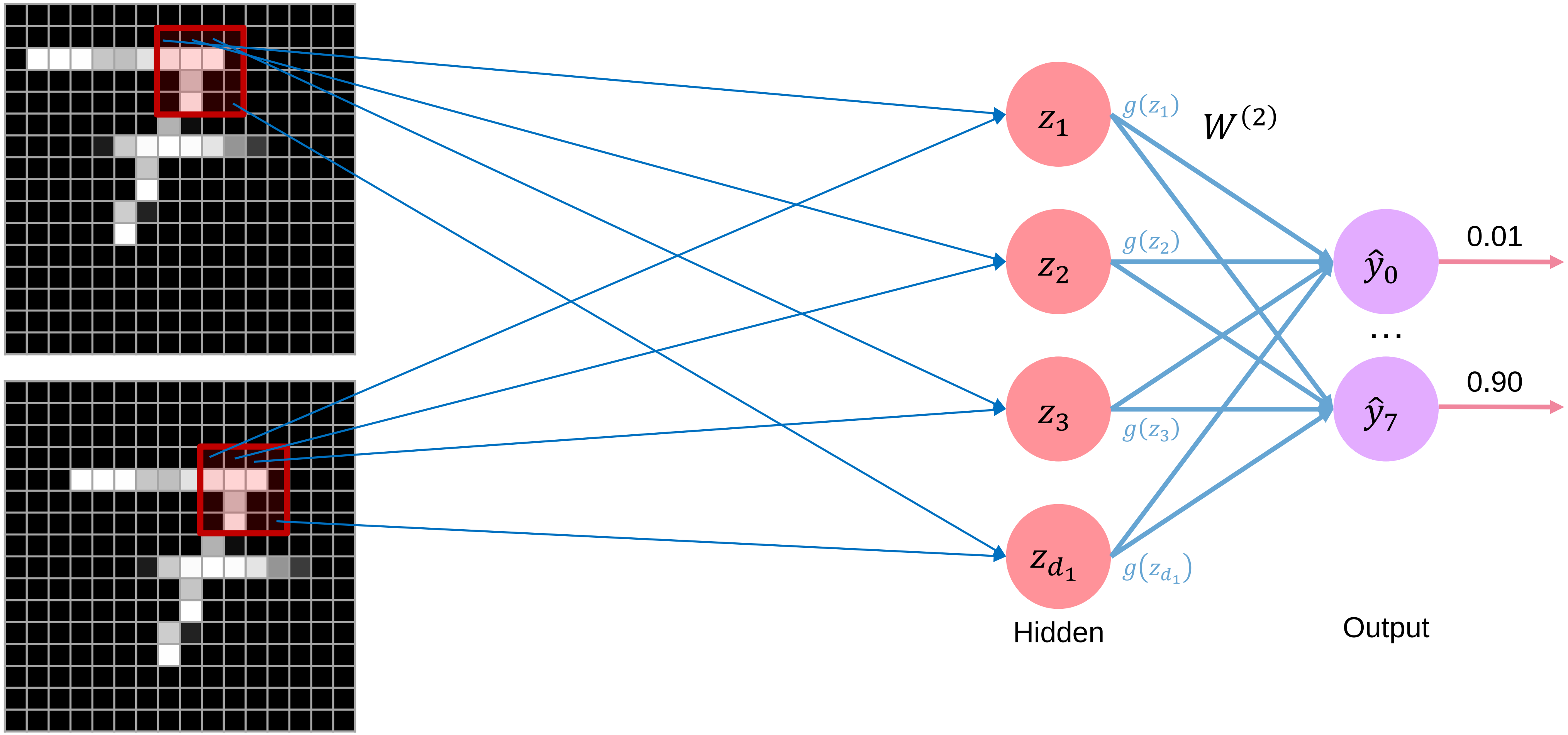
Why CNN?

What is an image used in Fully connected network?

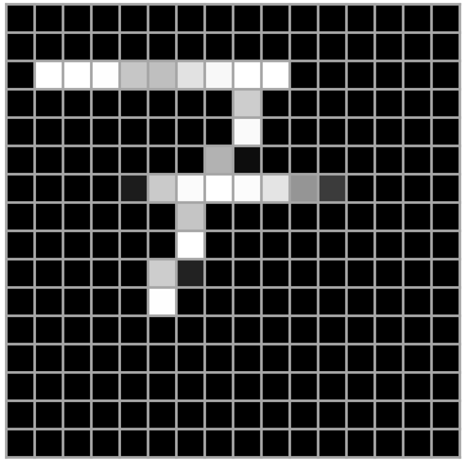


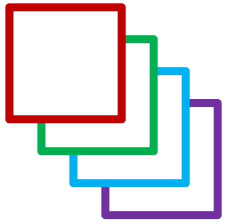
CESI LINEACT

Intuition with Convolutional NN



Intuition with Convolutional NN





x N filters

Filters are 4x4 = 16 weights

Each filter is applied on each part of the image

This is a **convolution** operation

Finally

- Each **filter** is a set of weights which extract **local features**
- For extracting different features, we use **multiple filters**

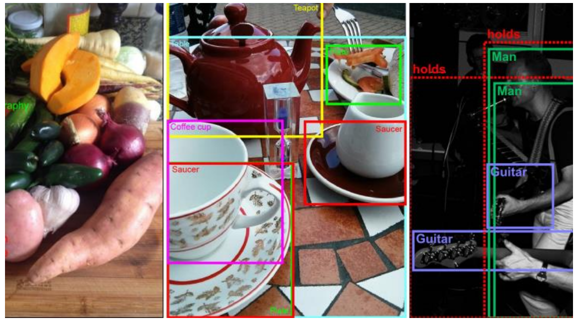
Sommaire

CESI ÉCOLE D'INGÉNIEURS

CESI LINEACT

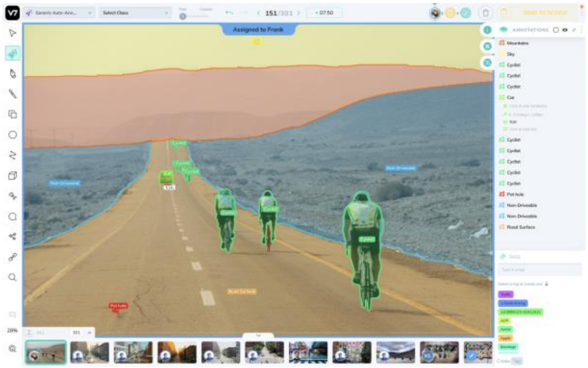
CESI LINEACT

Challenges examples



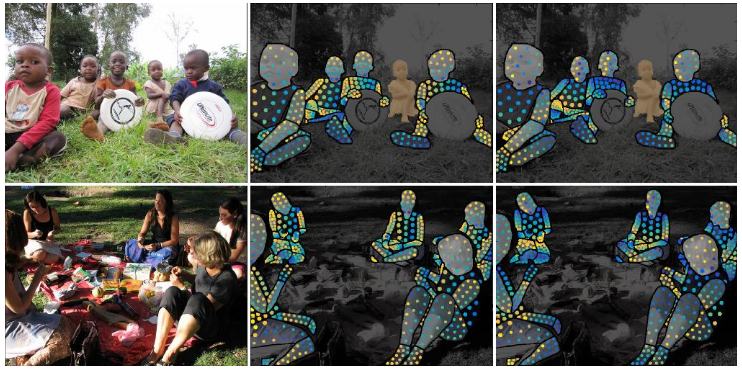
Detection
Bounded boxes

Segmentation
Instance segmentation



Panoptic segmentation [(v7 labs)]

Stuff



DensePose

Keypoints

Source: https://cocodataset.org/#overview

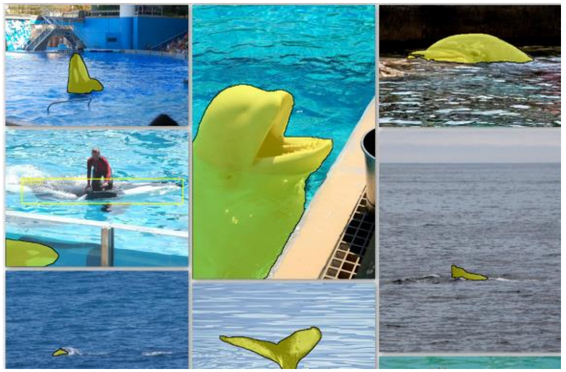
CESI LINEACT

Datasets

Nom	Challenges	Stats	Exemple
Coco [lien]	Detection Captions DensePose Keypoints Stuff Panoptic	330K images (>200K labeled) 1.5 million object instances 80 object categories 91 stuff categories 5 captions per image 250,000 people with keypoints	
ImageNet [Lien] [exemple]	Object localization Object detection Object detection from video Scene classification Scene parsing	Total number of non-empty synsets: 21841 Total number of images: 14,197,122 Number of images with bounding box annotations: 1,034,908 Number of synsets with SIFT features: 1000 Number of images with SIFT features: 1.2 million	

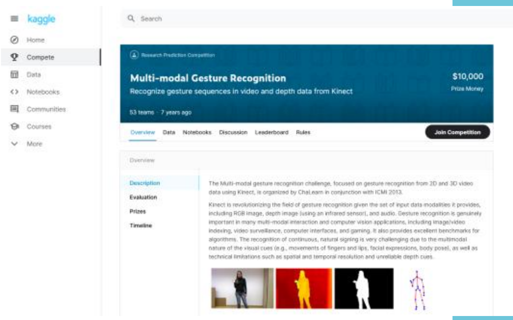
Synsets: synonym set

Datasets

Nom	Challenges	Stats	Exemple
Open Images Dataset V6 [lien] [exemple]	Object Detection: predicting a tight bounding box around all object instances of 500 classes. Visual Relationship Detection: detecting pairs of objects in particular relations. Instance Segmentation: predicting the outlines of object instances from 300 classes.	9M images annotated with • image-level labels, • object bounding boxes, • object segmentation masks, • visual relationships, • and localized narratives. It contains a total of: • 16M bounding boxes for • 600 object classes on 1.9M images	
KITTI [Lien]	Multi-Object Tracking Multi-Object Segmentation Object detection Depth completion Sceneflow	The Multi-Object and Segmentation (MOTS) benchmark consists of 21 training sequences and 29 test sequences	

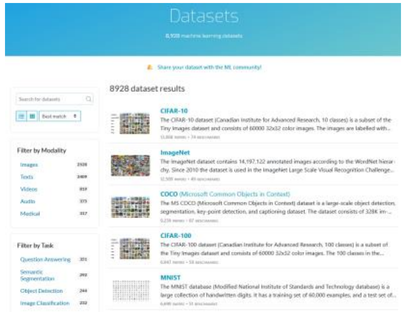
CESI LINEACT

Sites utiles



Kaggle https://www.kaggle.com/

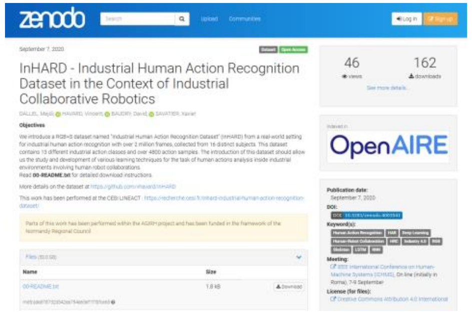
- Contient des datasets et challenges
- Exemple: https://www.kaggle.com/c/multi-modal-gesture-recognition/overview



Papers with code – Datasets https://paperswithcode.com/datasets

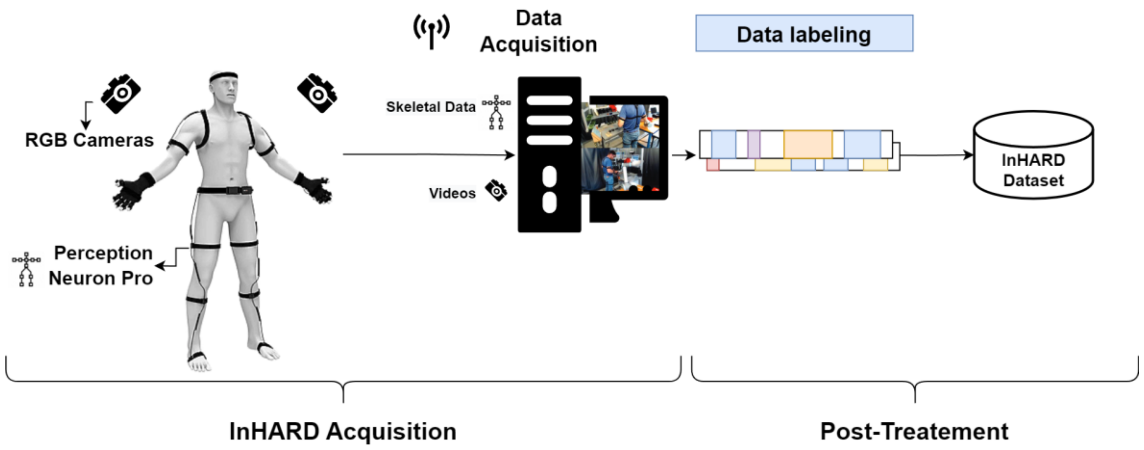
Zenodo https://www.zenodo.org/

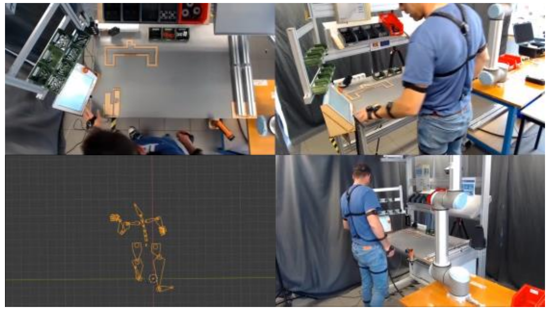
- Pour déposer un dataset de manière scientifique
- Exemple InHARD



1- jeu de données InHARD pour la HAR non segmentées dans un contexte industriel

Protocole d'acquisition - *Industrial Human Action Recognition Dataset*



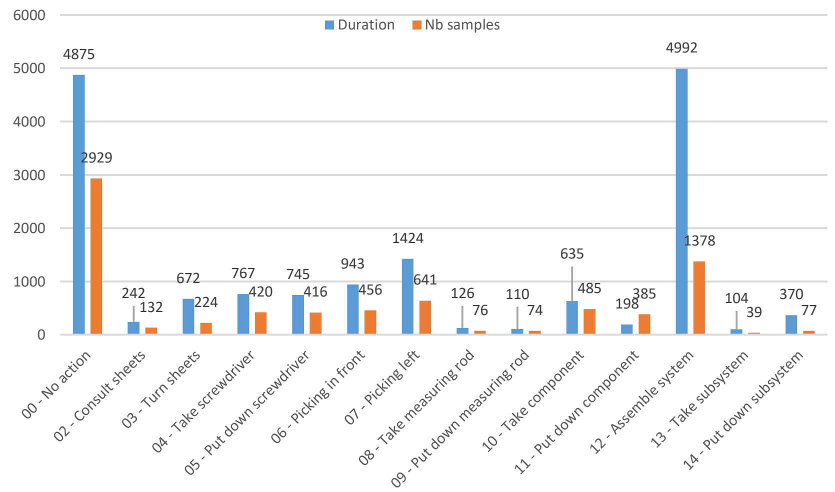


Contributions d'InHARD:

- Actions industrielles
- Multi modalités
- Multi-vues
- Cas d'usage réaliste

Résultats

Nb. participants	Nb. d'instances d'action	Nb. classes	Nb. vues	Modalités	Type d'actions	Capteurs
16	4803	13 + background	3	**Squelette 3D** RGB x3 **Squelette 2D**	**Industrielles**	Perception Neuron 3 Caméras C920



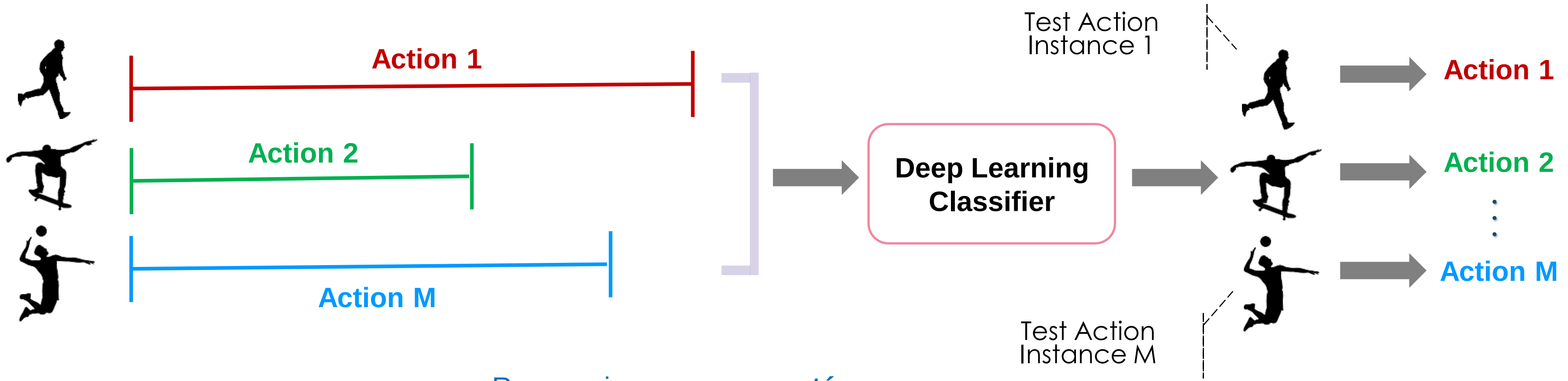
Performance de **référence** de l'algorithme ST-GCN* sur **InHARD segmentée**

Type de données d'entrée	Accuracy (segmentée)	F1-Score (segmentée)
Données squelettes 3D	0.919	0.921
Données squelettes 2D (OpenPose)	0.864	0.863

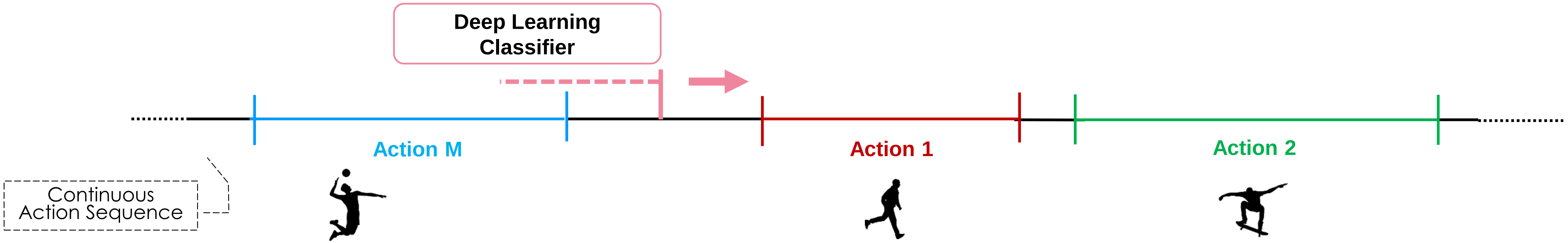
M. Dallel, V. Havard, D. Baudry, and X. Savatier, presented at the IEEE International Conference on Human-Machine Systems, 2020

CESI LINEACT

Reconnaissance d'actions segmentée / non segmentée



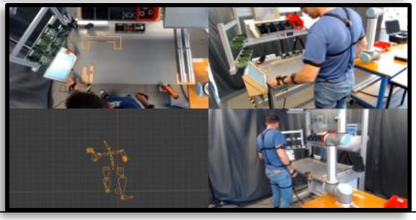
Reconnaissance **segmentée**



Reconnaissance **non segmentée (en ligne/continue)**

CESI LINEACT

Résultats LINEACT – Reconnaissance d'actions en ligne

Jeu de données	Données d'entrées	Classes	Résultats	Extraits
OAD	Positions 2D des jointures (X & Y)	10+1 (Aucune action)	**Accuracy: 0.954** **F1-score: 0.953**	
UOW	Positions 3D des jointures (X, Y & Z)	21 + 1 (Aucune action)	**Accuracy: 0.936** **F1-score: 0.934** **Latence : 0.047**	
InHARD	Positions 3D des jointures (X, Y & Z)	13 + 1 (Aucune action)	**Accuracy: 0.344** **F1-score: 0.433**	

M. Dallel, V. Havard, Y. Dupuis, and D. Baudry. 2022. A Sliding Window Based Approach With Majority Voting for Online Human Action Recognition using Spatial Temporal Graph Convolutional Neural Networks. In 2022 7th International Conference on Machine Learning Technologies (ICMLT)

Sommaire

Convolution

Mathematical operation of two greyscale images Im et k, notée $Im * k$, defined as *(Perret, 2017)*:

$$\forall (x,y) \in \mathbb{Z}^2, (Im * k)(x,y) = \sum_i \sum_j Im(i,j) \cdot k(x-i, y-j)$$

8_{-1}	2_{0}	5_{1}	3	0	4	9	1
4_{-1}	5_{0}	6_{1}	6	5	3	0	7
2_{-1}	9_{0}	4_{1}	4	7	6	1	9
9	7	3	2	9	9	2	9
6	1	7	0	0	7	8	7
3	2	9	8	4	8	9	8
6	4	8	9	6	5	7	4
7	6	5	3	3	4	1	3

Im

*

-1	0	1
-1	0	1
-1	0	1

k

=

	.						

$(Im * k)(1,1)$

CESI LINEACT

Convolution

Mathematical operation of two greyscale images Im et k, notée $Im * k$, defined as *(Perret, 2017)*:

$$\forall (x, y) \in \mathbb{Z}^2, (Im * k)(x, y) = \sum_i \sum_j Im(i, j) \cdot k(x - i, y - j)$$

8_{-1}	2_{0}	5_{1}	3	0	4	9	1
4_{-1}	5_{0}	6_{1}	6	5	3	0	7
2_{-1}	9_{0}	4_{1}	4	7	6	1	9
9	7	3	2	9	9	2	9
6	1	7	0	0	7	8	7
3	2	9	8	4	8	9	8
6	4	8	9	6	5	7	4
7	6	5	3	3	4	1	3

Im

$$(8 * -1) + (2 * 0) + (5 * 1)$$
$$(4 * -1) + (5 * 0) + (6 * 1)$$
$$(2 * -1) + (9 * 0) + (4 * 1)$$
$$= -1$$

=

	.						

$(Im * k)(1, 1)$

CESI LINEACT

Convolution

Mathematical operation of two greyscale images Im et k, notée $Im * k$, defined as *(Perret, 2017)*:

$$\forall (x,y) \in \mathbb{Z}^2, (Im * k)(x,y) = \sum_i \sum_j Im(i,j) \cdot k(x-i, y-j)$$

8 $_{-1}$	2 $_{0}$	5 $_{1}$	3	0	4	9	1
4 $_{-1}$	5 $_{0}$	6 $_{1}$	6	5	3	0	7
2 $_{-1}$	9 $_{0}$	4 $_{1}$	4	7	6	1	9
9	7	3	2	9	9	2	9
6	1	7	0	0	7	8	7
3	2	9	8	4	8	9	8
6	4	8	9	6	5	7	4
7	6	5	3	3	4	1	3

Im

$$(8*-1)+(2*0)+(5*1)$$
$$(4*-1)+(5*0)+(6*1)$$
$$(2*-1)+(9*0)+(4*1)$$
$$=-1$$

$=$

	-1						

$(Im * k)(1,1)$

CESI LINEACT

Convolution

Mathematical operation of two greyscale images Im et k, notée $Im * k$, defined as *(Perret, 2017)*:

$$\forall(x,y) \in \mathbb{Z}^2, (Im * k)(x,y) = \sum_i \sum_j Im(i,j) \cdot k(x-i, y-j)$$

8	2_{-1}	5_{0}	3_{1}	0	4	9	1
4	5_{-1}	6_{0}	6_{1}	5	3	0	7
2	9_{-1}	4_{0}	4_{1}	7	6	1	9
9	7	3	2	9	9	2	9
6	1	7	0	0	7	8	7
3	2	9	8	4	8	9	8
6	4	8	9	6	5	7	4
7	6	5	3	3	4	1	3

Im

$$(2 * -1) + (5 * 0) + (3 * 1)$$
$$(5 * -1) + (6 * 0) + (6 * 1)$$
$$(9 * -1) + (4 * 0) + (4 * 1)$$
$$= 3$$

=

	-1	3					

$(Im * k)(2,1)$

Convolution

Mathematical operation of two greyscale images Im et k, notée $Im * k$, defined as *(Perret, 2017)*:

$$\forall (x,y) \in \mathbb{Z}^2, (Im * k)(x,y) = \sum_i \sum_j Im(i,j) \cdot k(x-i, y-j)$$

8	2	5_{-1}	3_{0}	0_{1}	4	9	1
4	5	6_{-1}	6_{0}	5_{1}	3	0	7
2	9	4_{-1}	4_{0}	7_{1}	6	1	9
9	7	3	2	9	9	2	9
6	1	7	0	0	7	8	7
3	2	9	8	4	8	9	8
6	4	8	9	6	5	7	4
7	6	5	3	3	4	1	3

Im

$$(5 * -1) + (3 * 0) + (0 * 1)$$
$$(6 * -1) + (6 * 0) + (5 * 1)$$
$$(4 * -1) + (4 * 0) + (7 * 1)$$
$$= 3$$

=

	-1	3	3				

$(Im * k)(3,1)$

CESI LINEACT

Convolution

Mathematical operation of two greyscale images Im et k, notée $Im * k$, defined as *(Perret, 2017)*:

$$\forall (x, y) \in \mathbb{Z}^2, (Im * k)(x, y) = \sum_i \sum_j Im(i, j) \cdot k(x - i, y - j)$$

8	2	5 $_{-1}$	3 $_{0}$	0 $_{1}$	4	9	1
4	5	6 $_{-1}$	6 $_{0}$	5 $_{1}$	3	0	7
2	9	4 $_{-1}$	4 $_{0}$	7 $_{1}$	6	1	9
9	7	3	2	9	9	2	9
6	1	7	0	0	7	8	7
3	2	9	8	4	8	9	8
6	4	8	9	6	5	7	4
7	6	5	3	3	4	1	3

Im

*

-1	0	1
-1	0	1
-1	0	1

k

=

	-1	3	3	0	2	-4	
	2	9	-8	-6	18	-7	
	3	11	-2	-16	5	-3	
	-1	0	6	-14	-6	0	
	-9	-10	14	-3	-14	1	
	-6	-8	9	3	-4	2	

CESI LINEACT

Convolution

Mathematical operation of two greyscale images Im et k, notée $Im * k$, defined as *(Perret, 2017)*:

$$\forall (x,y) \in \mathbb{Z}^2, (Im * k)(x,y) = \sum_i \sum_j Im(i,j) \cdot k(x-i, y-j)$$

8	2	5	3	0	4	9	1
4	5	6	6	5	3	0	7
2	9	4	4	7	6	1	9
9	7	3	2	9	9	2	9
6	1	7	0	0	7	8	7
3	2	9	8	4	8	9	8
6	4	8	9	6	5	7	4
7	6	5	3	3	4	1	3

Im

*

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

k

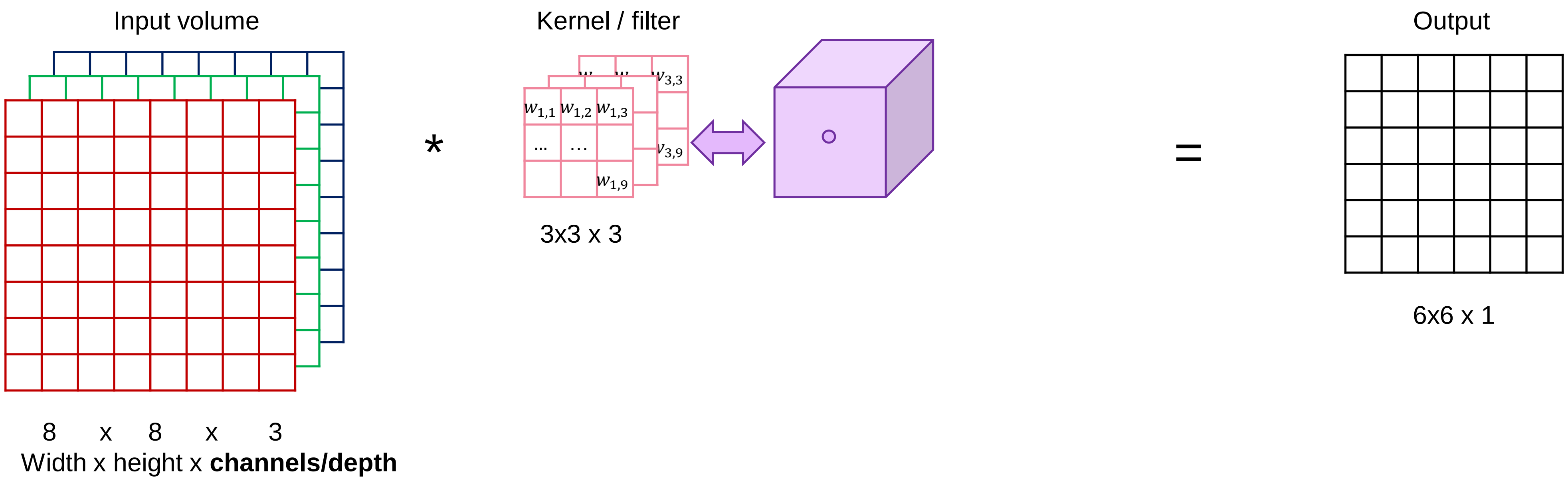
=

	-1	3	3	0	2	-4	
	2	9	-8	-6	18	-7	
	3	11	-2	-16	5	-3	
	-1	0	6	-14	-6	0	
	-9	-10	14	-3	-14	1	
	-6	-8	9	3	-4	2	

Sommaire

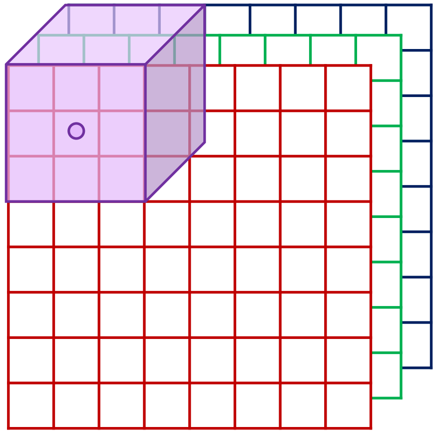
CESI LINEACT

Convolution on Volume: example on RGB image



Convolution on Volume: example on RGB image

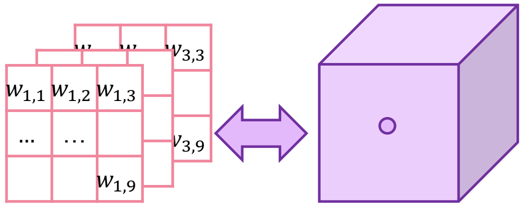
Input volume



8 x 8 x 3
Width x height x **channels**

*

Kernel / filter

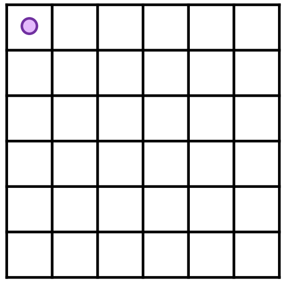


3x3 x 3

Stride=1

=

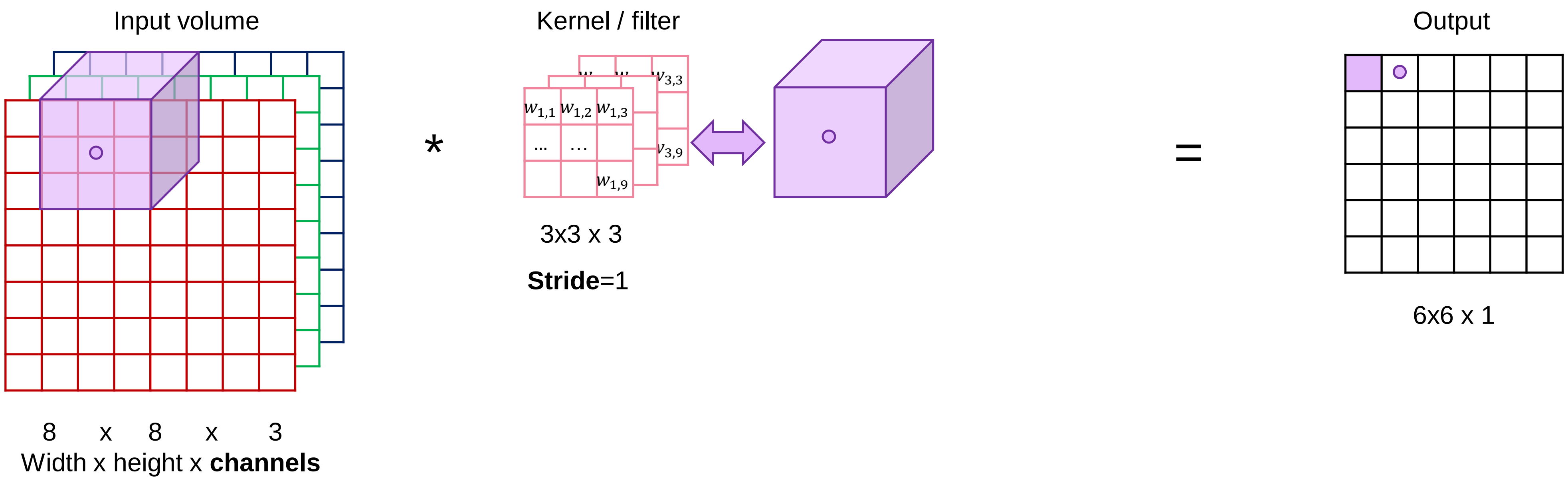
Output



6x6 x 1

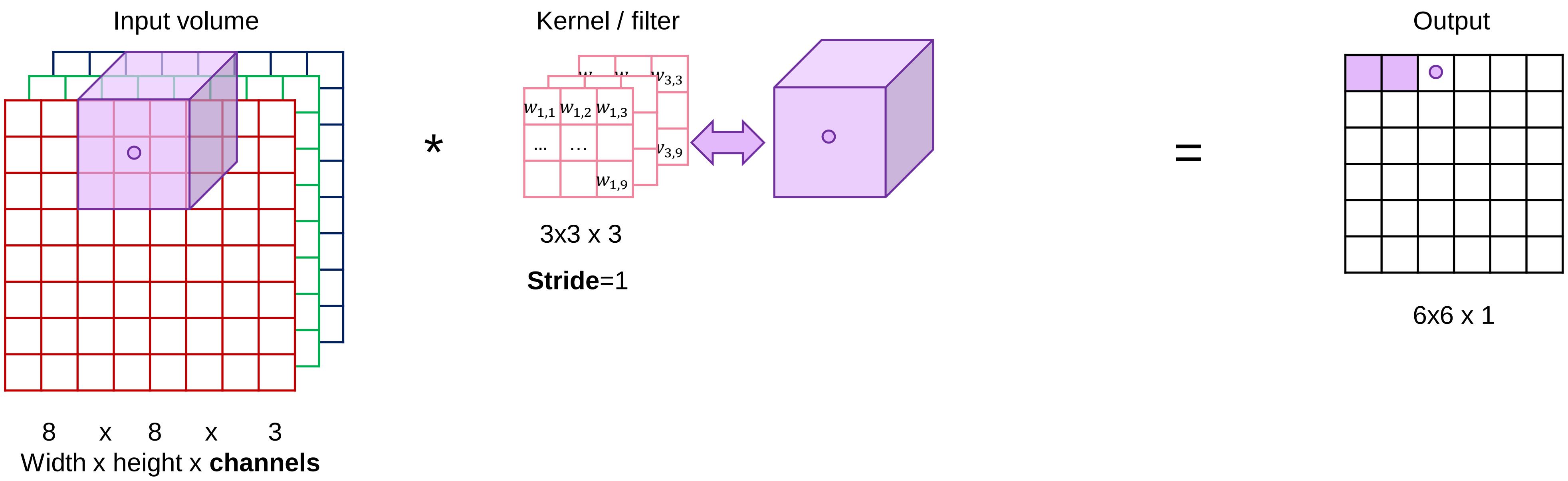
CESI LINEACT

Convolution on Volume: example on RGB image



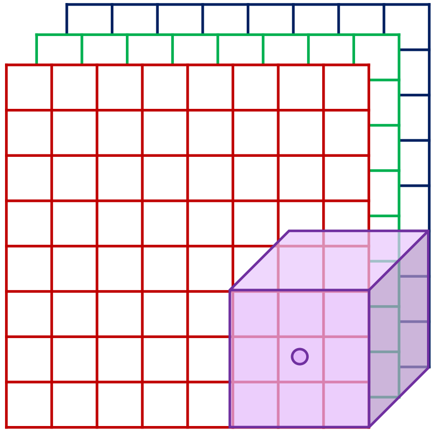
CESI LINEACT

Convolution on Volume: example on RGB image



Convolution on Volume: example on RGB image

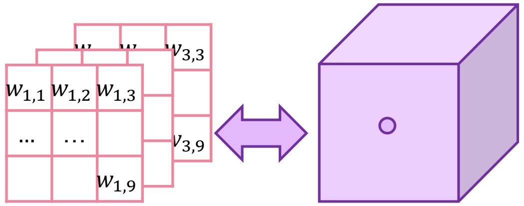
Input volume



8 x 8 x 3
Width x height x **channels**

*

Kernel / filter

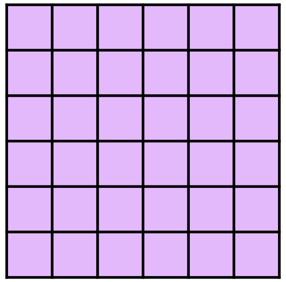


3x3 x 3

Stride=1

=

Output



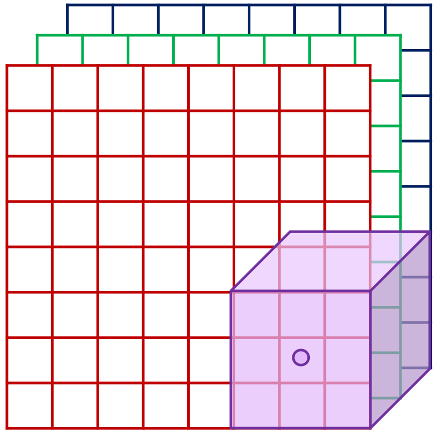
6x6

Only in 2 dimensions

CESI LINEACT

Convolution on Volume: example on RGB image

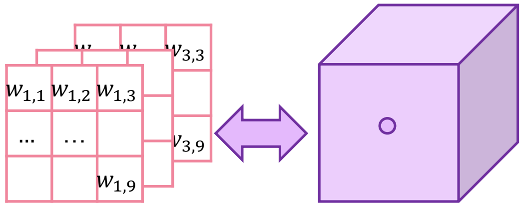
Input volume



8 x 8 x 3
Width x height x **channels**

*

Kernel / filter



3x3 x 3

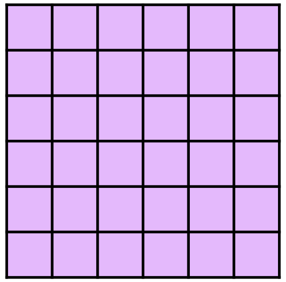
Stride=1

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Example for detecting vertical gradient on Red channel of the input volume

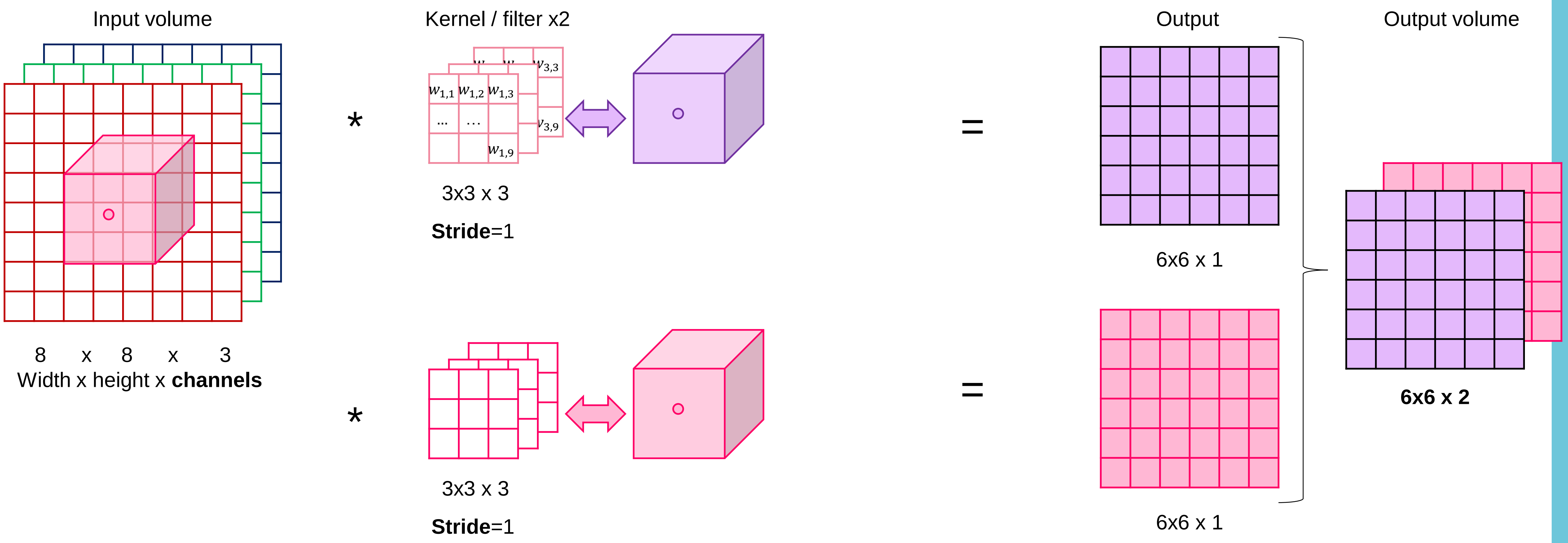
=

Output



6x6 x 1

Convolution on Volume: example on RGB image

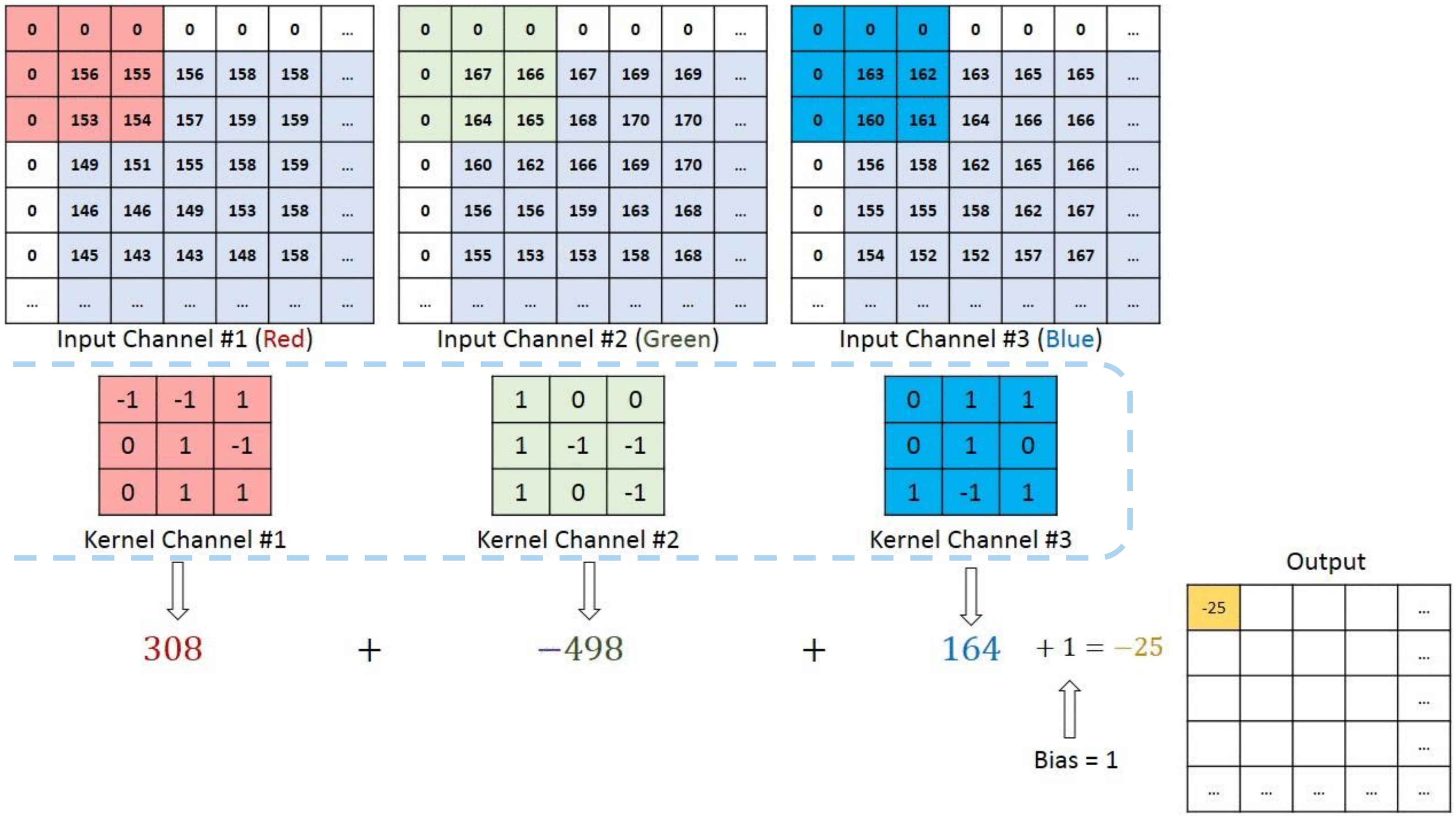


CESI LINEACT

Convolution on Volume explained by (Arat, 2017)

Input volume

Kernel / filter

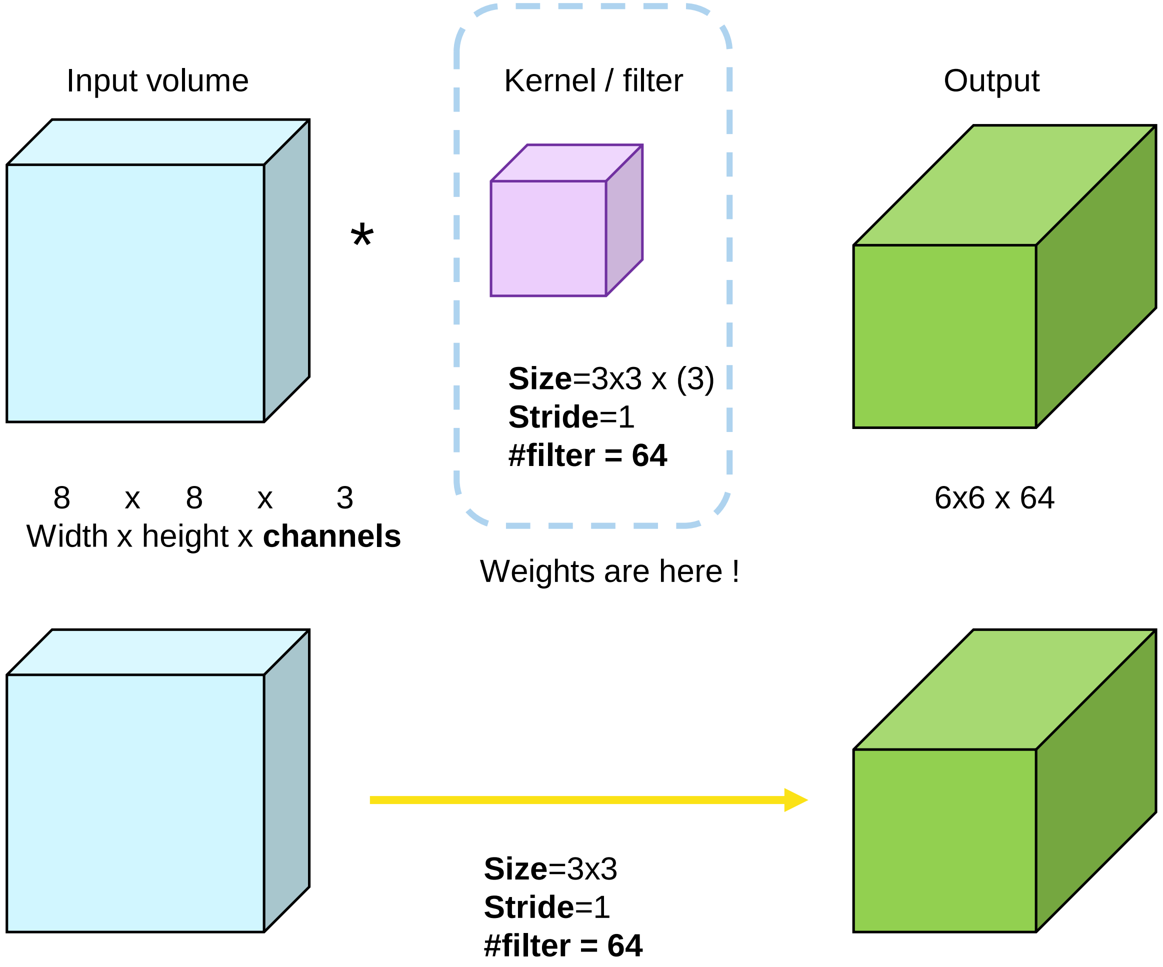


Output

Source, [(ARAT, 2017)]

CESI LINEACT

Convolution on Volume: notation



Even simpler notation

Convolution on Volume: summary of notation

- If an **input volume** $[l-1]$ is treated by a **convolutional layer** at depth $[l]$, the result is an **output volume** at depth $[l]$

Input Volume properties

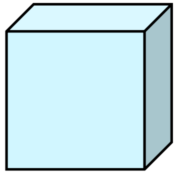
- $n_H^{[l-1]}$ nb lines (ex: 256)
- $n_W^{[l-1]}$ nb columns (ex: 256)
- $n_c^{[l-1]}$ nb channels / depth (ex: 32)

Filter / kernel properties

- $f^{[l]}$ filter size (ex: $5 \times 5 \times 32$)
- $n_c^{[l]}$ number of filters (ex: 64)
- $p^{[l]}$ padding size (ex: 0)
- $s^{[l]}$ stride (ex: 2)

Output volume properties

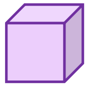
- $n_H^{[l]} = \dfrac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1$
- $n_W^{[l]}$ same as above
- $n_c^{[l]}$ nb channels / depth (ex: 64)



$n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]}$

$256 \times 256 \times 32$

$*$

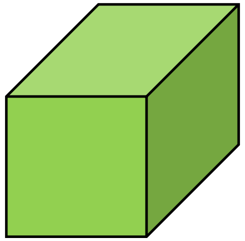


$f^{[l]} \times f^{[l]} \times \underline{n_c^{[l-1]}}$

$5 \times 5 \times \underline{32}$

$\#filter = \underline{n_c^{[l]}} = \underline{64}$

$=$



$n_H^{[l]} \times n_W^{[l]} \times \underline{n_c^{[l]}}$

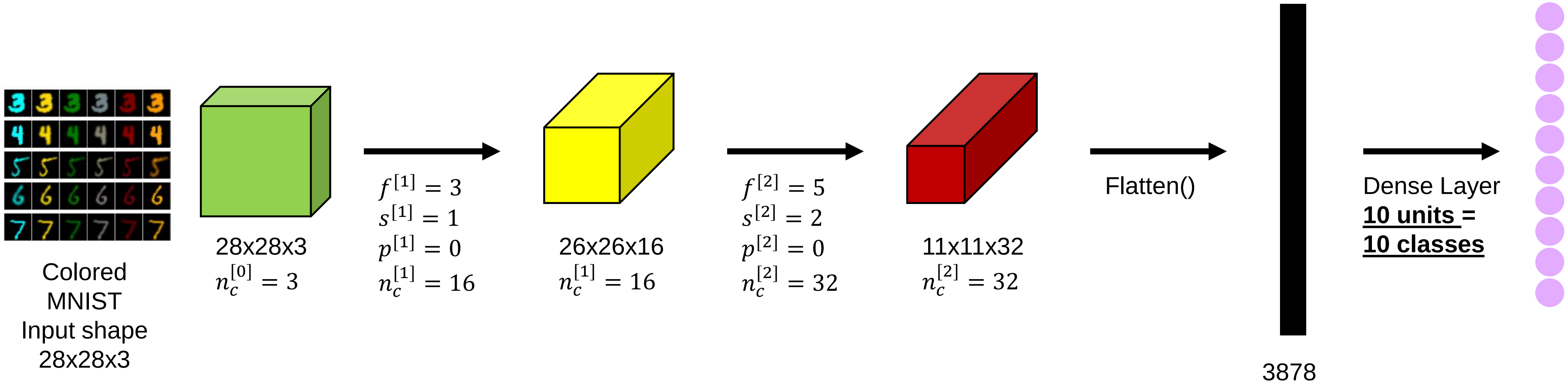
$125 \times 125 \times \underline{64}$

Sommaire

CESI LINEACT

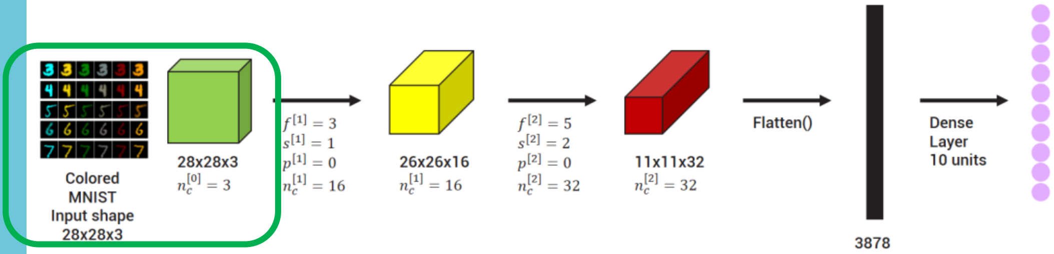
Example of ConvNet - computing volume dimension

$$n_H^{[l]} = \frac{n_H^{[l-1]} + 2\,p^{[l]} - f^{[l]}}{s^{[l]}} + 1$$



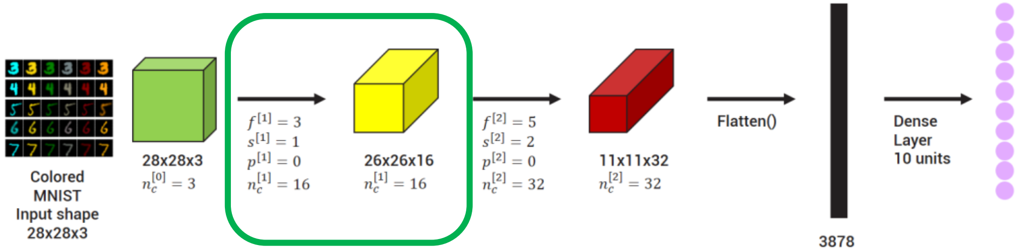
Conv2D is "**NHWC**" by default. It means:
N samples x **Height** x **Width** x **C Channels**

Example of ConvNet - computing volume dimension




```python
def create_model(input_shape = (28,28,3), summary=False, loss_fn_to_use =
                 tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)):
    model = tf.keras.models.Sequential([
        tf.keras.layers.Conv2D(16, (3, 3), strides=(1,1), input_shape=input_shape,
                               padding="valid", activation="relu"),
        tf.keras.layers.Conv2D(32, (5, 5), strides=(2,2), padding="valid", activation="relu"),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(10)
    ])

    # Set loss function to use with the model
    loss_fn = loss_fn_to_use

    # and compile
    model.compile(optimizer='adam', loss=loss_fn, metrics=['accuracy',
                                                            tf.metrics.SparseCategoricalAccuracy()])
    if summary:
        model.summary()
    return model
```

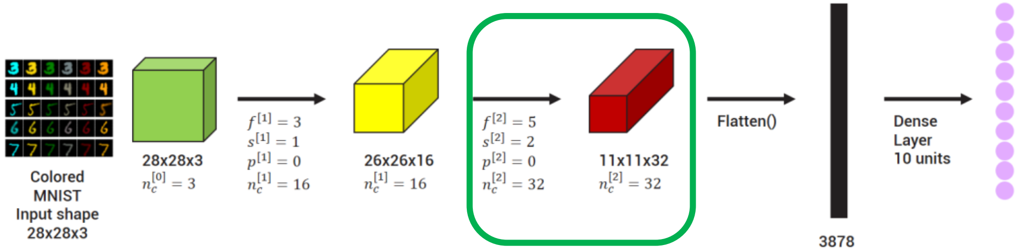
Example of ConvNet - computing volume dimension




```python
def create_model(input_shape = (28,28,3), summary=False, loss_fn_to_use =
                 tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)):
    model = tf.keras.models.Sequential([
        tf.keras.layers.Conv2D(16, (3, 3), strides=(1,1), input_shape=input_shape,
                               padding="valid", activation="relu"),
        tf.keras.layers.Conv2D(32, (5, 5), strides=(2,2), padding="valid", activation="relu"),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(10)
    ])

    # Set loss function to use with the model
    loss_fn = loss_fn_to_use

    # and compile
    model.compile(optimizer='adam', loss=loss_fn, metrics=['accuracy',
                                                          tf.metrics.SparseCategoricalAccuracy()])
    if summary:
        model.summary()
    return model
```

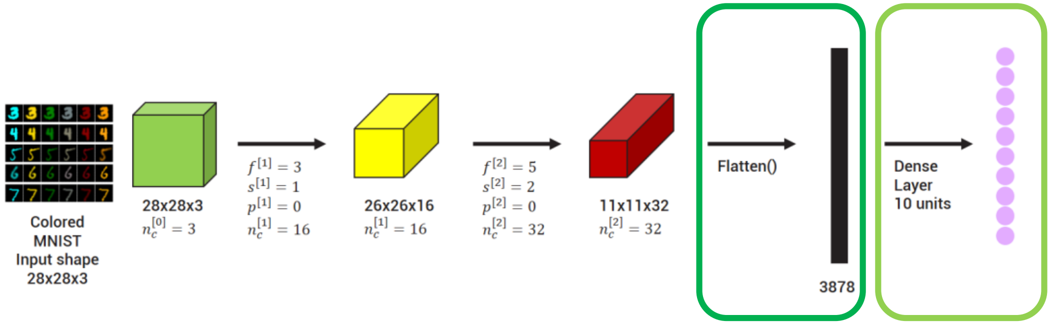
CESI LINEACT

Example of ConvNet - computing volume dimension




```python
def create_model(input_shape = (28,28,3), summary=False, loss_fn_to_use =
                 tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)):
    model = tf.keras.models.Sequential([
        tf.keras.layers.Conv2D(16, (3, 3), strides=(1,1), input_shape=input_shape,
                               padding="valid", activation="relu"),
        tf.keras.layers.Conv2D(32, (5, 5), strides=(2,2), padding="valid", activation="relu"),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(10)
    ])

    # Set loss function to use with the model
    loss_fn = loss_fn_to_use

    # and compile
    model.compile(optimizer='adam', loss=loss_fn, metrics=['accuracy',
                                                           tf.metrics.SparseCategoricalAccuracy()])
    if summary:
        model.summary()
    return model
```

Example of ConvNet - computing volume dimension




```python
def create_model(input_shape = (28,28,3), summary=False, loss_fn_to_use =
                 tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)):
    model = tf.keras.models.Sequential([
        tf.keras.layers.Conv2D(16, (3, 3), strides=(1,1), input_shape=input_shape,
                               padding="valid", activation="relu"),
        tf.keras.layers.Conv2D(32, (5, 5), strides=(2,2), padding="valid", activation="relu"),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(10)
    ])

    # Set loss function to use with the model
    loss_fn = loss_fn_to_use

    # and compile
    model.compile(optimizer='adam', loss=loss_fn, metrics=['accuracy',
                                                          tf.metrics.SparseCategoricalAccuracy()])
    if summary:
        model.summary()
    return model
```

Sommaire

Max Pooling

Pooling is about **reducing dimension** while **keeping information**.

- **Idea**: keep only maximum value in a neighbourhood

255	200	5	**54**
28	32	28	1
150	215	10	**30**
200	**230**	15	5

Max Pooling
Size=2x2
Stride=2

```
tf.keras.layers.MaxPooling2D(
   pool_size=(2, 2),
   strides=(2, 2))
```

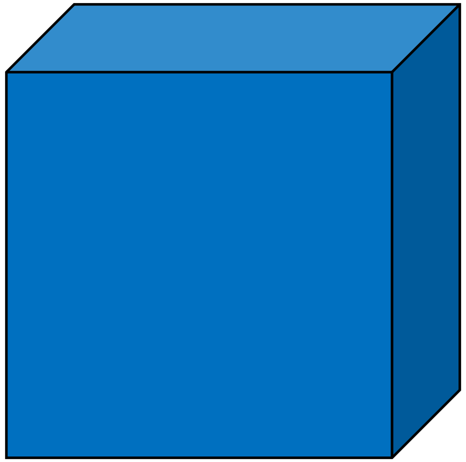
255	**54**
230	**30**

Max Pooling only reduce the volume dimension but do not modify volume values.

Max Pooling

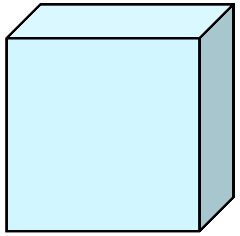
Pooling is about **reducing dimension** while **keeping information**.

- **Idea**: keep only maximum value in a neighbourhood



Max Pooling
Size=2x2
Stride=2

```
tf.keras.layers.MaxPooling2D(
   pool_size=(2, 2),
   strides=(2, 2))
```



Padding "valid": $n_H^{[l]} = \frac{n_H^{[l-1]} - pool^{[l]} + 1}{s^{[l]}}$

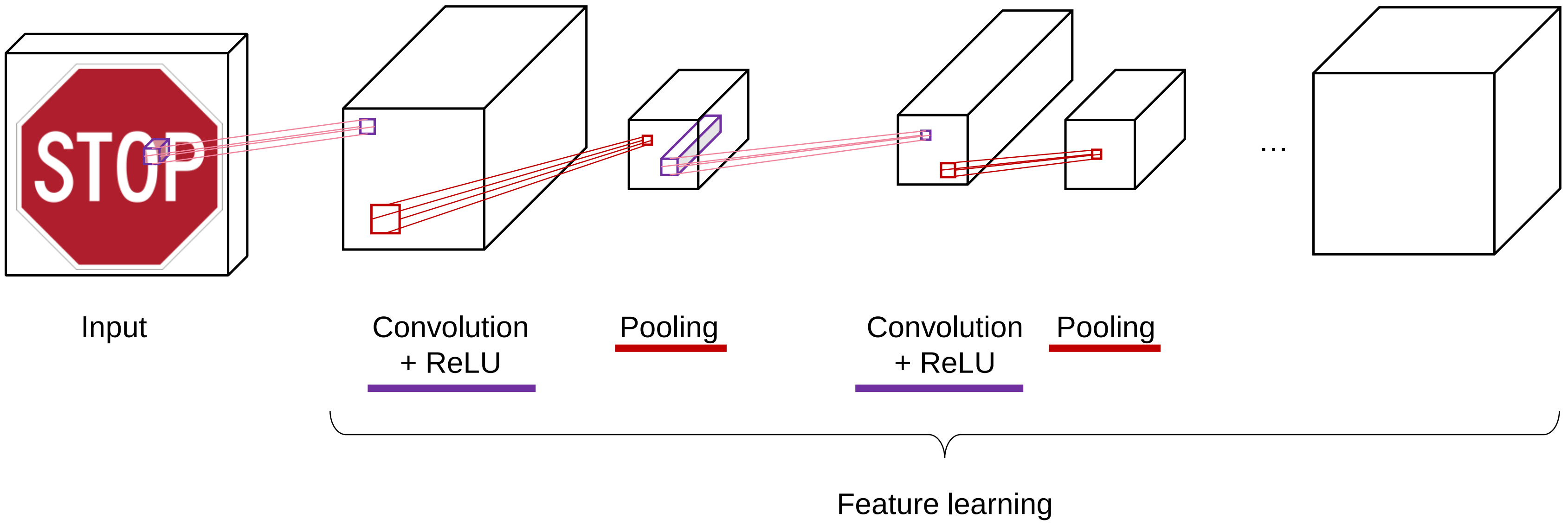
Padding "same": $n_H^{[l]} = \frac{n_H^{[l-1]}}{s^{[l]}}$

Max Pooling only reduce the volume dimension but do not modify volume values.

Sommaire

CNNs for classification

Here is the principle of CNNs architecture for classification

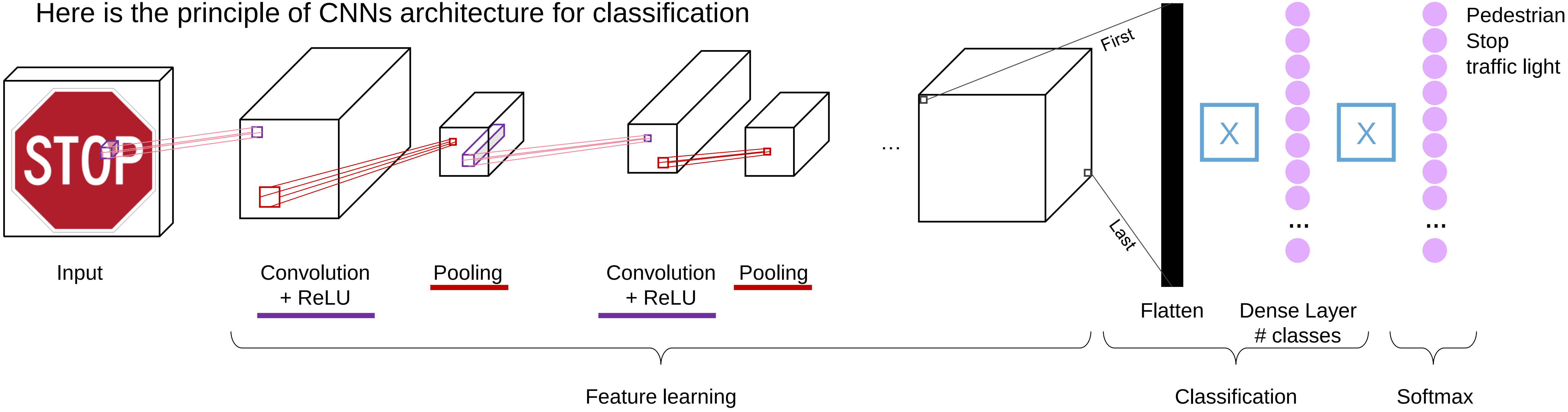


- **Feature learning** with a series of convolutions
- **Non-linearity** with activation function (generally ReLU)
- Reduce dimension with **Max Pooling**

CESI LINEACT

CNNs for classification

Here is the principle of CNNs architecture for classification

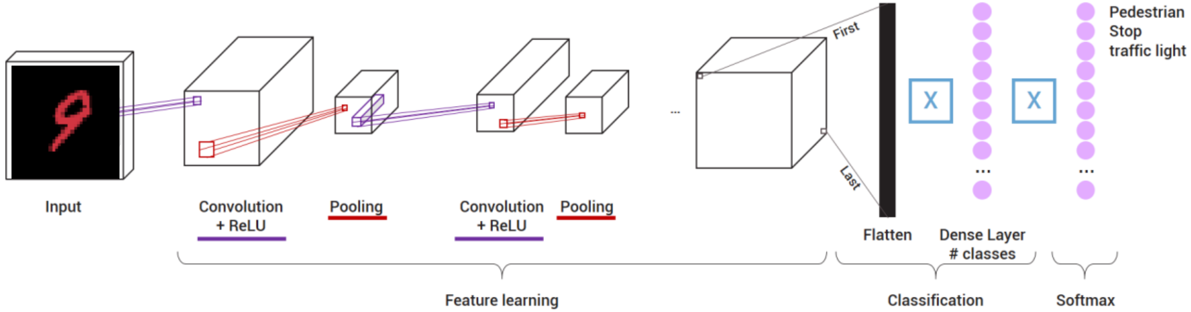


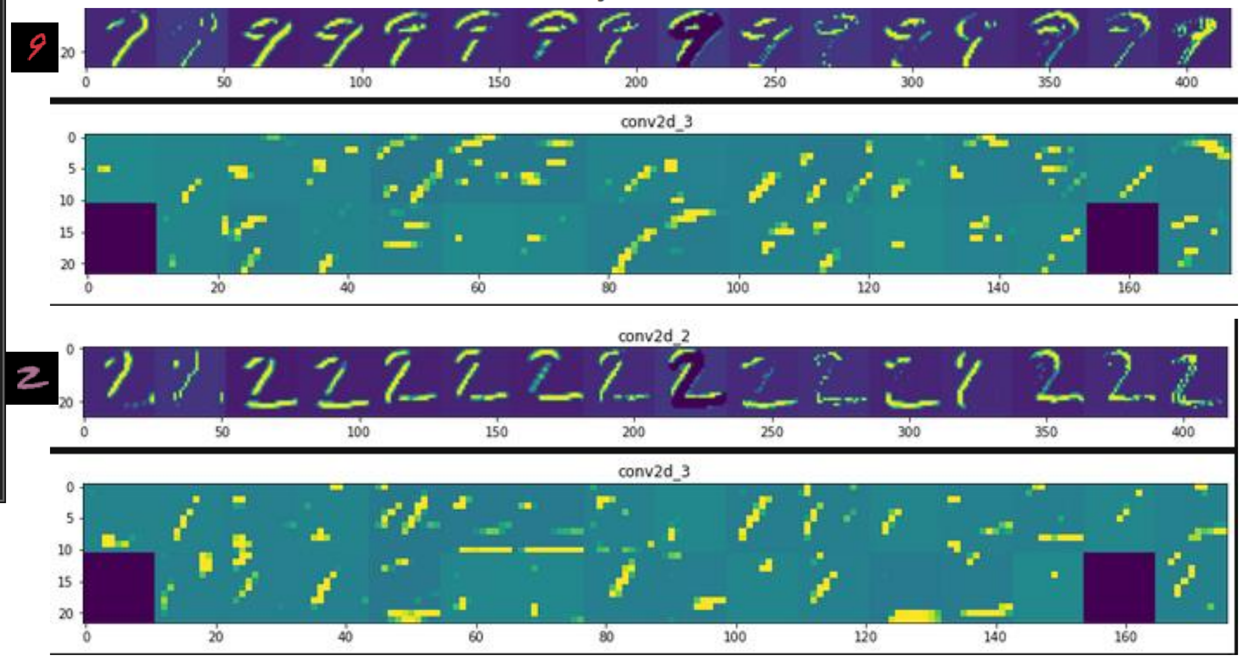
- **Feature learning** with a series of convolutions
- **Non-linearity** with activation function (generally ReLU)
- Reduce dimension with **Max Pooling**

- **Classification** with Dense layer

CNNs for classification

```
def create_model_v2(input_shape = (28,28,3), summary=False, loss_fn_to_use =
                tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)):
    model = tf.keras.models.Sequential([

        tf.keras.layers.Conv2D(16, (3, 3), strides=(1,1), input_shape=input_shape,
                               padding="valid", activation="relu"),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2), strides=(1, 1), padding='same'),

        tf.keras.layers.Conv2D(32, (5, 5), strides=(2,2), padding="valid", activation="relu"),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2), strides=(1, 1), padding='same'),

        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(10)
    ])

    # Set loss function to use with the model
    loss_fn = loss_fn_to_use

    # and compile
    model.compile(optimizer='adam', loss=loss_fn, metrics=['accuracy',
                                    tf.metrics.SparseCategoricalAccuracy()])
    if summary:
        model.summary()
    return model
```



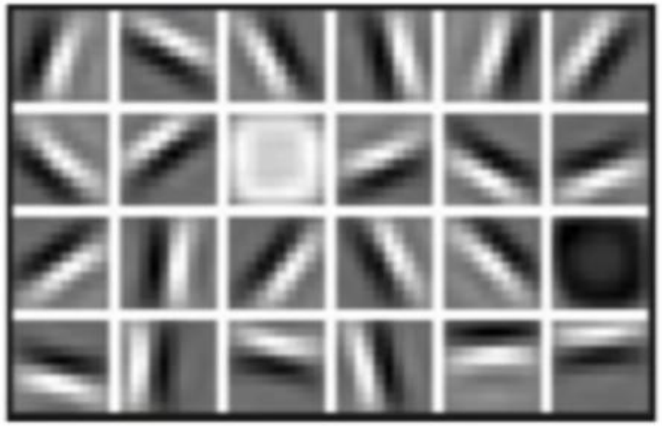


CESI LINEACT

CNNs for classification: from low level to high level features

Layers goes from low level to high level features

Low level features



Lines, edges and dark spots

Conv layer 1

Mid level features

Eyes, Nose, Ears

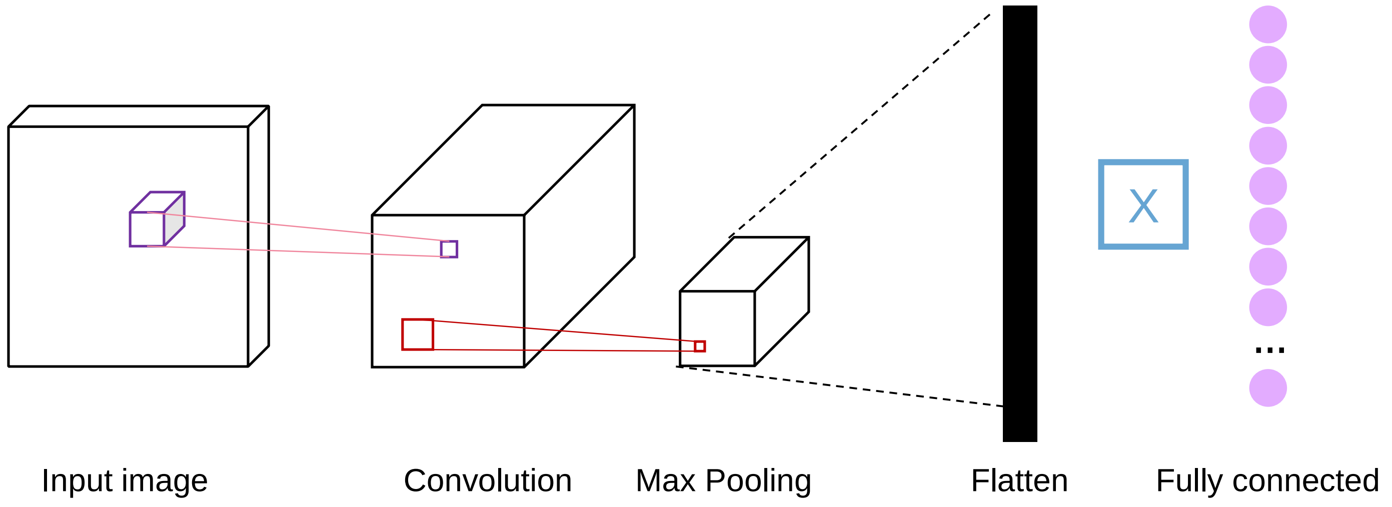
Conv layer 2

High level features

Facial structure

Conv layer 3

Summarizing Tensor Flow CNNs



1. **Convolutions** apply filters to generate feature maps
2. **Non-linearity** often ReLU
3. **Pooling**: down sampling on each feature map
4. **Flatten:** for getting a 1 dimensional vector

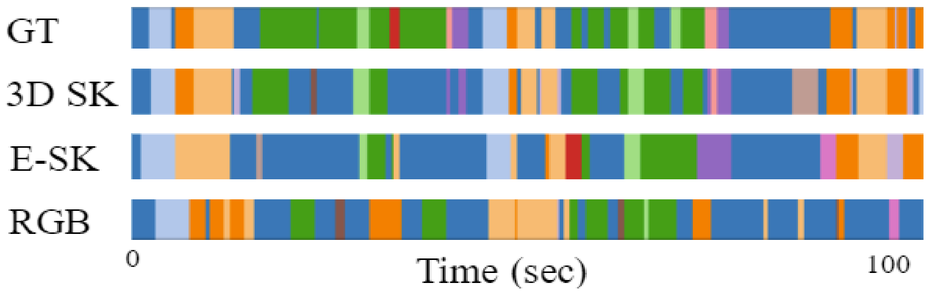
```
tf.keras.layers.Conv2D
tf.keras.activations.relu
tf.keras.layers.MaxPool2D
tf.keras.layers.Flatten
```

Sommaire

Analyse des résultats : Modalités (RGB/ E-SK/ 3D SK)

Etude comparative des **modalités** utilisées sur les bases **InHARD-3, InHARD-4** et **InHARD-13**.

Method	Dataset	Modality	F1@10	F1@25	F1@50	Edit	MoF
C2F-TCN	InHARD-3	RGB	49,9	38,51	12,83	57,9	50,13
GSK-C2F(p)	InHARD-3	E-SK	68.31	65,23	49,23	**80.21**	80.45
GSK-C2F(p)	**InHARD-3**	**3D SK**	**74.68**	**73.42**	**56.96**	81.47	**84.82**
C2F-TCN	InHARD-4	RGB	28,2	17,7	8,52	53,23	32,57
GSK-C2F(p)	InHARD-4	E-SK	80.13	70,19	49,06	74.67	67.39
GSK-C2F(p)	**InHARD-4**	**3D SK**	**81.76**	**75.50**	**57.62**	**76.98**	**70.28**
C2F-TCN	InHARD-13	RGB	26,70	19,12	6,77	35,92	34,28
GSK-C2F(p)	InHARD-13	E-SK	67.83	62.47	43.16	62.84	59.77
GSK-C2F(p)	**InHARD-13**	**3D SK**	**81.121**	**80.86**	**71.37**	**74.37**	**73.86**



Segmentation temporelle de la vidéo "P13-R02" de **InHARD-13**

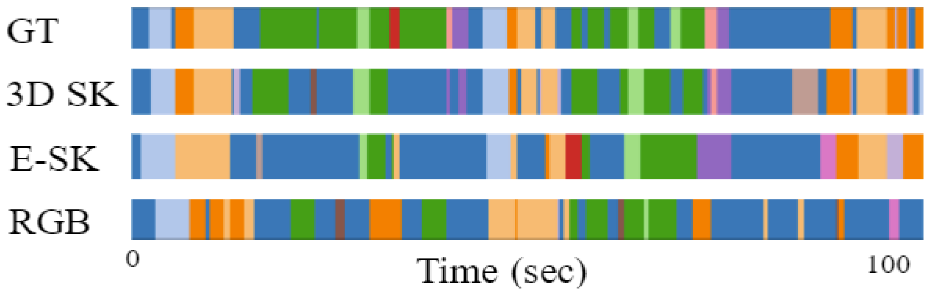
Segmentation d’une vidéo d’assemblage :
GT en vert **et** **prédiction en orange.**

Benmessabih et al., 2024

Analyse des résultats : Modalités (RGB/ E-SK/ 3D SK)

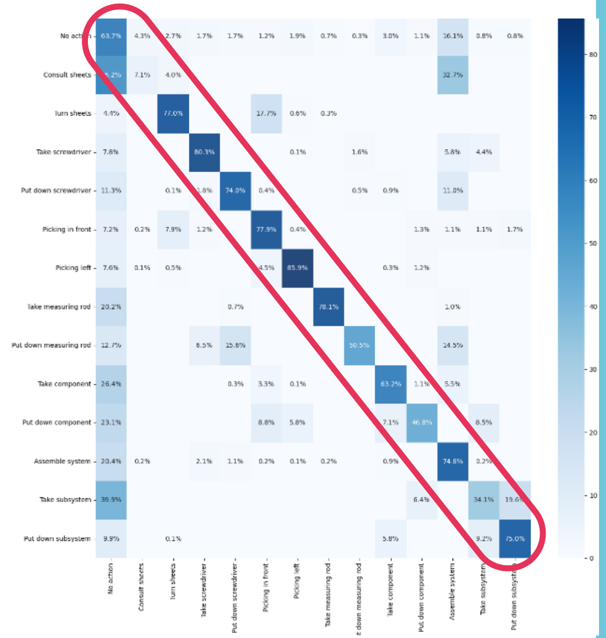
Etude comparative des **modalités** utilisées sur les bases **InHARD-3, InHARD-4** et **InHARD-13**.

Method	Dataset	Modality	F1@10	F1@25	F1@50	Edit	MoF
C2F-TCN	InHARD-3	RGB	49,9	38,51	12,83	57,9	50,13
GSK-C2F(p)	InHARD-3	E-SK	68.31	65,23	49,23	**80.21**	80.45
GSK-C2F(p)	**InHARD-3**	**3D SK**	**74.68**	**73.42**	**56.96**	81.47	**84.82**
C2F-TCN	InHARD-4	RGB	28,2	17,7	8,52	53,23	32,57
GSK-C2F(p)	InHARD-4	E-SK	80.13	70,19	49,06	74.67	67.39
GSK-C2F(p)	**InHARD-4**	**3D SK**	**81.76**	**75.50**	**57.62**	**76.98**	**70.28**
C2F-TCN	InHARD-13	RGB	26,70	19,12	6,77	35,92	34,28
GSK-C2F(p)	InHARD-13	E-SK	67.83	62.47	43.16	62.84	59.77
GSK-C2F(p)	**InHARD-13**	**3D SK**	**81.121**	**80.86**	**71.37**	**74.37**	**73.86**



Segmentation temporelle de la vidéo "P13-R02" de **InHARD-13**

C2F-TCN **(RGB)**

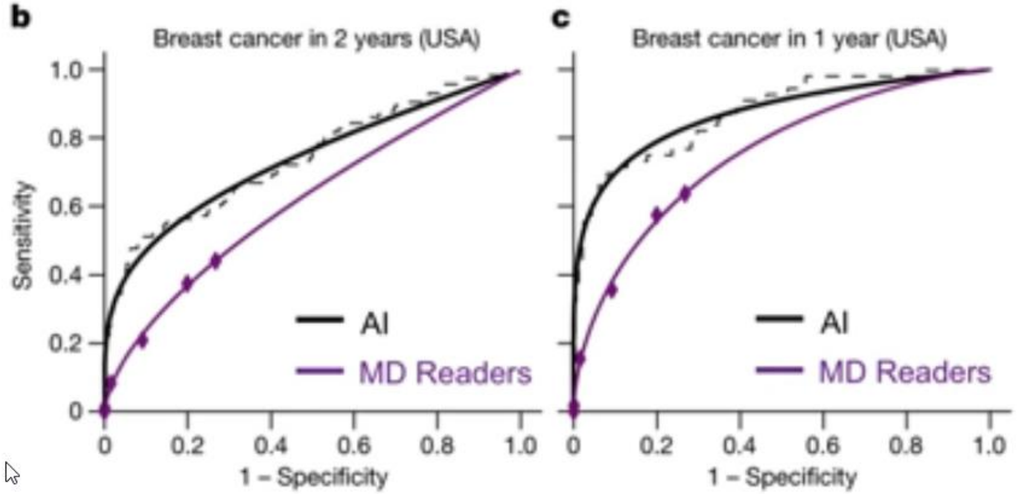


GSK-C2F **(3D SK)**

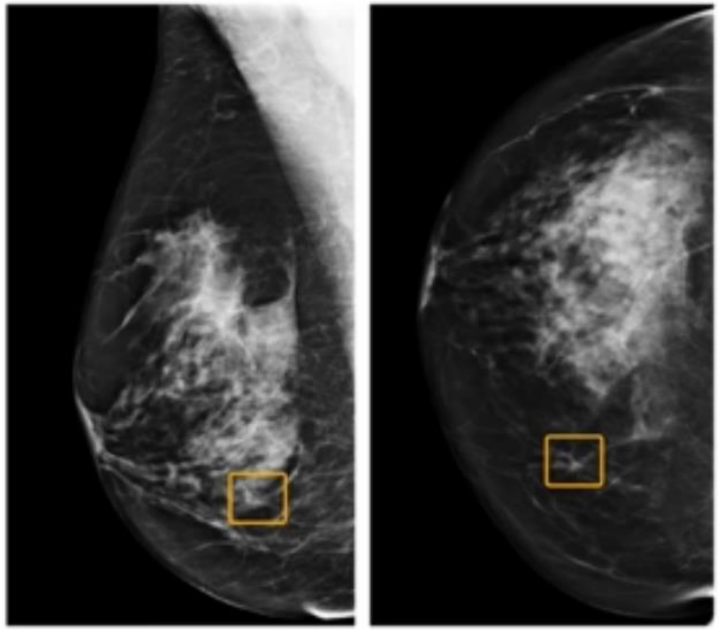
Benmessabih et al., 2024

Detection: Breast Cancer Screening

McKinney, S.M., Sieniek, M., Godbole, V. *et al.* International evaluation of an AI system for breast cancer screening. *Nature* **577,** 89–94 (2020). https://doi.org/10.1038/s41586-019-1799-6



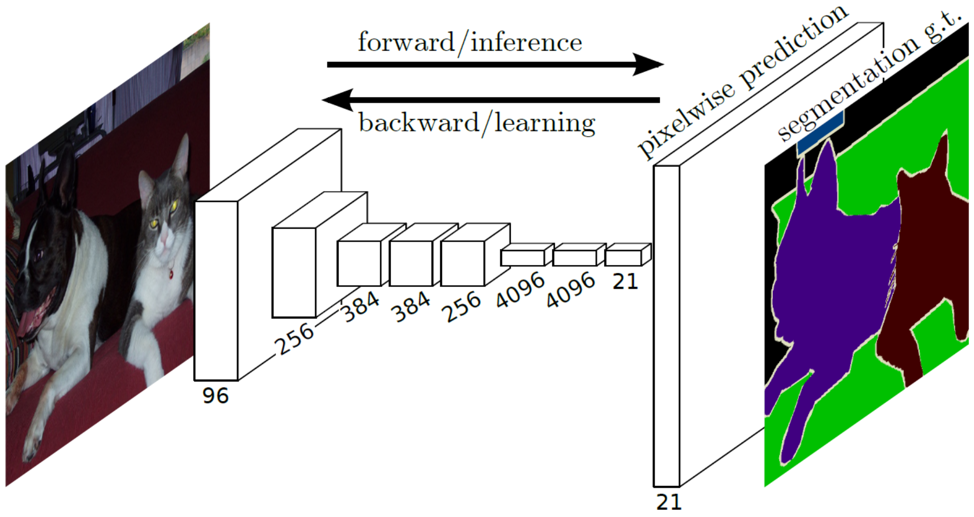
CNN-based system outperformed expert radiologists at detecting breast cancer from mammograms

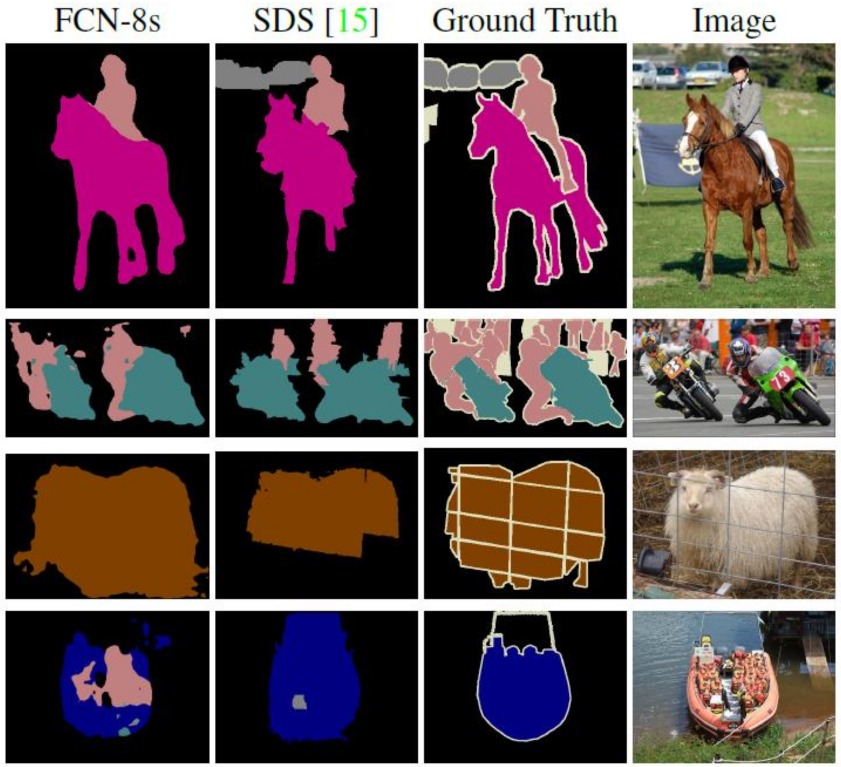


breast cancer case missed by radiologist but detected by AI

Semantic segmentation with Fully Convolutionnal Networks (FCN)

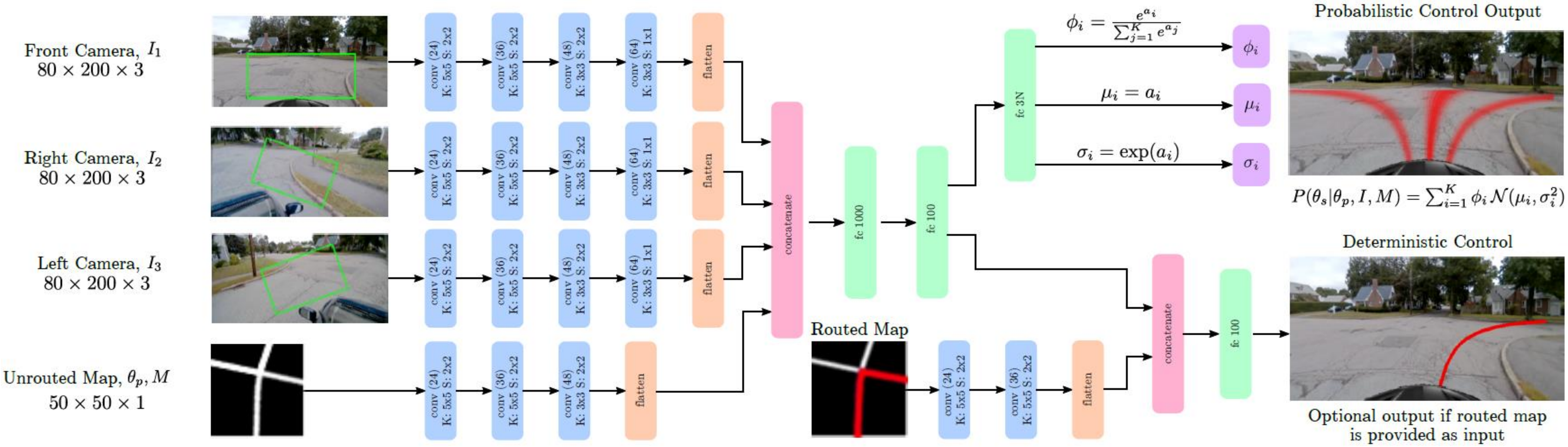
J. Long, E. Shelhamer, and T. Darrell, “Fully Convolutional Networks for Semantic Segmentation,” p. 10.





Self-drving cars: Navigation from visual perception

A. Amini, G. Rosman, S. Karaman, and D. Rus, “Variational End-to-End Navigation and Localization,” *arXiv:1811.10119 [cs, stat]*, Jun. 2019, Accessed: Jan. 10, 2021. [Online]. Available: http://arxiv.org/abs/1811.10119.



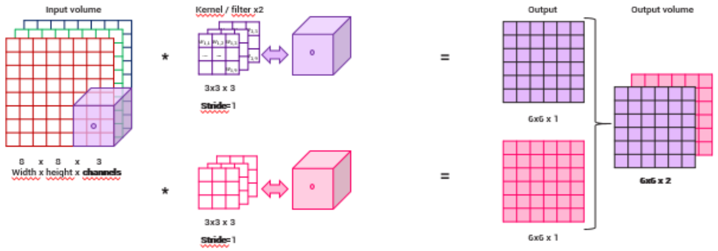
Video at: https://youtu.be/iaSUYvmCekI?t=2112

Sommaire

Conclusion about Convolutional Neural Networks

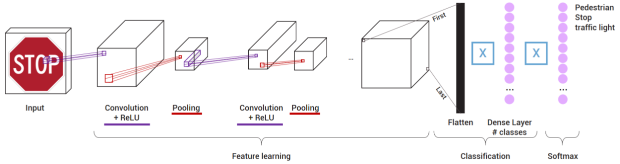
Convolution on Volume

- Convolution reminder
- Volume by using several filters



CNN and Max Pooling

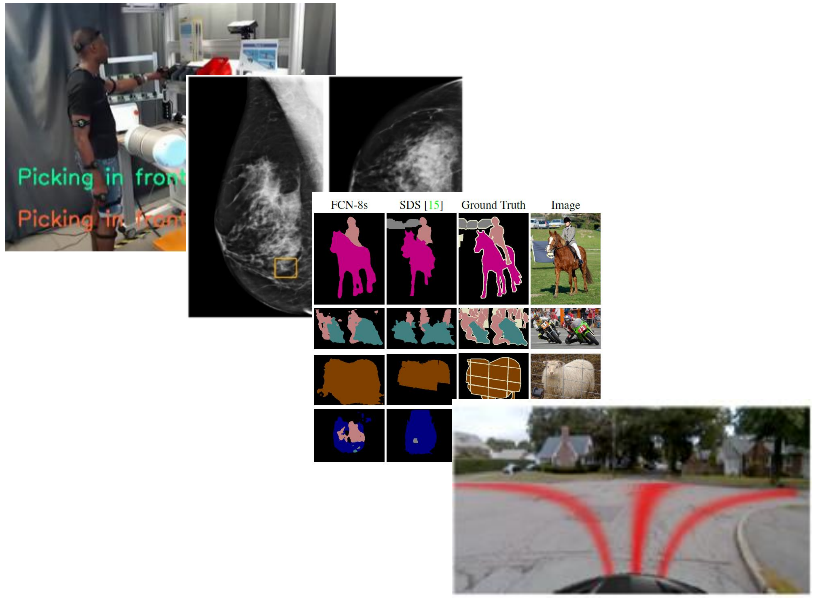
- CNN learns features on images
- Max Pooling downsample volumes
- Coding in Tensorflow




```python
def create_model_v2(input_shape = (28,28,3), summary=False, loss_fn_to_use =
                    tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)):
    model = tf.keras.models.Sequential([

        tf.keras.layers.Conv2D(16, (3, 3), strides=(1,1), input_shape=input_shape,
                               padding="valid", activation="relu"),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2), strides=(1, 1), padding='same'),

        tf.keras.layers.Conv2D(32, (5, 5), strides=(2,2), padding="valid", activation="relu"),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2), strides=(1, 1), padding='same'),

        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(10)
    ])

    # Set loss function to use with the model
    loss_fn = loss_fn_to_use

    # and compile
    model.compile(optimizer='adam', loss=loss_fn, metrics=['accuracy',
                                                 tf.metrics.SparseCategoricalAccuracy()])
    if summary:
        model.summary()
    return model
```

CNN usages

- Medical: Breast cancer detection
- Semantic segmentation
- Autonomous cars



References

Amini A. and Soleimany Ava, MIT 6.S191: Convolutional Neural Networks | MIT 6.S191

A. Amini, G. Rosman, S. Karaman, and D. Rus, "Variational End-to-End Navigation and Localization," *arXiv:1811.10119 [cs, stat]*, Jun. 2019, Accessed: Jan. 10, 2021. [Online]. Available: http://arxiv.org/abs/1811.10119.

Arat M. M., Implementing 'SAME' and 'VALID' padding of Tensorflow in Python, 2017, accessible at https://mmuratarat.github.io/2019-01-17/implementing-padding-schemes-of-tensorflow-in-python, accessed January 2020

J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," p. 10.

McKinney, S.M., Sieniek, M., Godbole, V. *et al.* International evaluation of an AI system for breast cancer screening. *Nature* **577,** 89–94 (2020). https://doi.org/10.1038/s41586-019-1799-6

Ng A., Deep Learning specialization, Coursera, available at https://www.deeplearning.ai/deep-learning-specialization/, accessed in June 2017

Perret B., traitement et analyse d'images, 2017, accessible à https://perso.esiee.fr/~perretb/I5FM/TAI/convolution/index.html

CESI LINEACT

References

Datasets:

A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, T. Duerig, and V. Ferrari. The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale. IJCV, 2020.

R. Benenson, S. Popov, and V. Ferrari. Large-scale interactive object segmentation with human annotators. CVPR, 2019.

Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. (* = equal contribution) **ImageNet Large Scale Visual Recognition Challenge**. *IJCV,* 2015

J. Carreira, E. Noland, C. Hillier, and A. Zisserman, *A Short Note on the Kinetics-700 Human Action Dataset*. 2019.

Ambika Choudhury, 10 Open Datasets You Can Use For Computer Vision Projects, available at https://analyticsindiamag.com/10-open-datasets-you-can-use-for-computer-vision-projects/

J. Fritsch, T. Kuehnl, and A. Geiger, “A New Performance Measure and Evaluation Benchmark for Road Detection Algorithms,” 2013.

A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite,” 2012.

M. Menze and A. Geiger, “Object Scene Flow for Autonomous Vehicles,” 2015.

A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets Robotics: The KITTI Dataset,” *International Journal of Robotics Research (IJRR)*, 2013.

Google Research, available at https://research.google/tools/datasets/, access on Nov. 10, 2019