IMAGES PRISES PAR UNE CAMÉRA PI V. 1.3 ET UN RPI ZÉRO VIEW
EN MODE DIRECT OU VIA UN LOGICIEL

ET ANALYSE D'IMAGE PAR INTELLIGENCE ARTIFICIELLE CNN (TENSORFLOW, OPENCV ETC..)
AI PAR LE MODULE D'IA DE GOOGLE AIY VIEW

REMAERQUE / Ne seront pas traités le time lapse la video la vision nocturne le transfert d'image sur le web

Versions de Raspbian

Release table [edit]

Ver.	Code-name	Release date	Final/ latest release (date)	No. of Archs. [i]	Package count		Linux kernel	End of support			References
					Binary	Source		Security	Long-term	Freexian ELTS[ii]	
0.90		26 January 1994		1	?	?	?	—			[12][13]
0.91		29 January 1994			?	?	0.99.14t	—			[12]
0.93R5		March 1995			?	?	?	—			[12]
0.93R6		November 1995			256	?	1.2.13	—			[12][14]
1.0		Never released	—		—	—	—	—			[12][15]
7	Wheezy	4 May 2013	7.11 (4 Jun 2016)	13	≈36,000	≈17,500	3.2	25 April 2016	31 May 2018	30 June 2020	[12][37][38][39] [35][10][40]
8	Jessie	25–26 April 2015	8.11 (23 Jun 2018)	10	≈43,000	≈20,000	3.16	17 June 2018	30 June 2020	30 June 2025	[12][41][42][35] [43][10]
9	Stretch	17 June 2017	9.13 (18 Jul 2020)		≈51,000	≈25,000	4.9	18 July 2020	30 June 2022	30 June 2027	[12][44][45][46] [35][10]
10	Buster	6 July 2019	10.13 (10 Sep 2022)		≈59,000	≈29,000	4.19	10 September 2022	30 June 2024	30 June 2029	[47][48][49]
11	Bullseye	14 August 2021	11.9 (10 Feb 2024)	9	59,551	31,387	5.10	July 2024	June 2026	30 June 2031	[1][50][51][52][53] [54][49]
12	**Bookworm**	10 June 2023	12.5 (10 Feb 2024)		64,419	34,780	6.1	June 2026	June 2028	30 June 2033	[55][56][57][58] [59]
13	Trixie	TBA	TBA	TBA	TBA	TBA	TBA	TBA	TBA	—	[60]
14	Forky	TBA	TBA	TBA	TBA	TBA	TBA	TBA	TBA	—	[61]
unstable	Sid	Rolling release		22[iii]	>67,000[iv]	>32,000[iv]	6.8.9	—	—	—	[49]

Legend: Old version | Older version, still maintained | **Latest version** | Future release

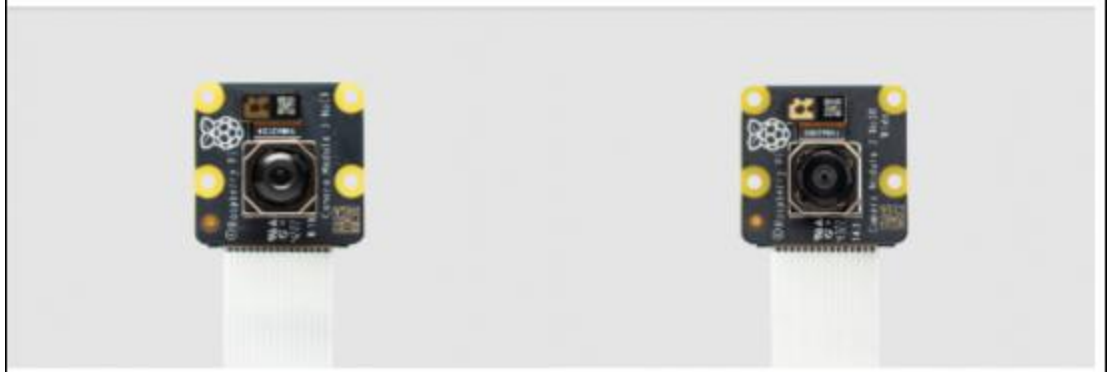
i. ^ The number of hardware architectures supported

ii. ^ Extended long-term support (ELTS) provided by Freexian[10] but made available to all Debian users, as noted on official Debian pages. There is and only sponsored packages are supported.[11]

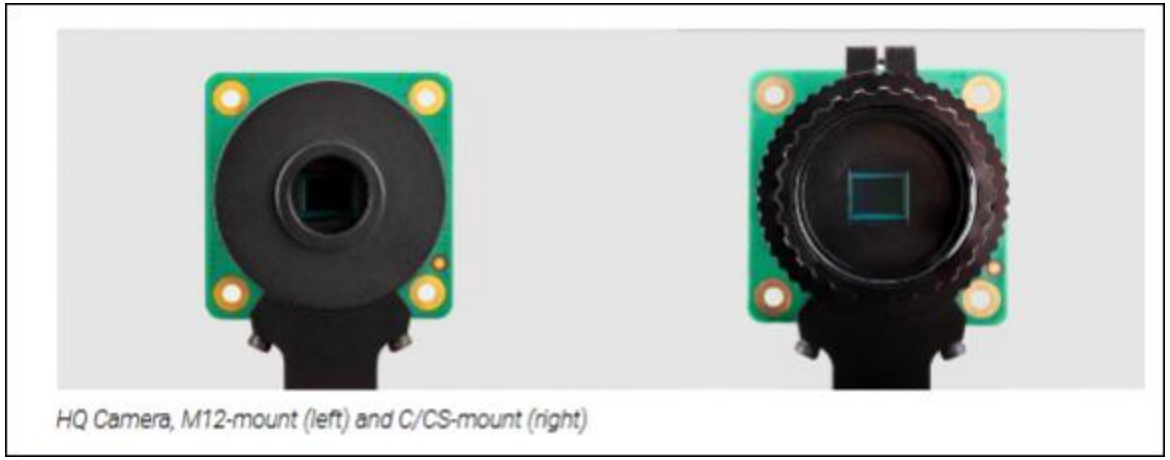
Picamera 2 avec Bullseye

CAMÉRAS NUMERIQUES et PRISES DE PHOTO/VIDEO

Camera Module 3 (left) and Camera Module 3 Wide (right)



Camera Module 3 NoIR (left) and Camera Module 3 NoIR Wide (right)



HQ Camera, M12-mount (left) and C/CS-mount (right)

Caméra numérique, diaphragme (shutter ligne par ligne et plusieurs passages (frames = Nombre de passages par unité de temps)

Sensor elements								->	Frame 1							
0	0	0	0	0	0	0	0									
0	0	0	0	0	0	0	0									
0	0	0	0	0	0	0	0									
0	0	0	0	0	0	0	0									
0	0	0	0	0	0	0	0									
0	0	0	0	0	0	0	0									
0	0	0	0	0	0	0	0									
0	0	0	0	0	0	0	0									

$$\frac{1\text{s}}{\text{min exposure time in s}} = \text{max framerate in fps}$$

Comparaison des Propriétés des différentes caméras (1/2)

	Camera Module v1	Camera Module v2	Camera Module 3	Camera Module 3 Wide	HQ Camera	GS Camera
					(excluding lens)	with adaptor and dust cap)
Weight	3g	3g	4g	4g	30.4g	34g (41g with adaptor and dust cap)
Still resolution	5 Megapixels	8 Megapixels	11.9 Megapixels	11.9 Megapixels	12.3 Megapixels	1.58 Megapixels
Video modes	1080p30, 720p60 and 640 × 480p60/90	1080p47, 1640 × 1232p41 and 640 × 480p206	2304 × 1296p56, 2304 × 1296p30 HDR, 1536 × 864p120	2304 × 1296p56, 2304 × 1296p30 HDR, 1536 × 864p120	2028 × 1080p50, 2028 × 1520p40 and 1332 × 990p120	1456 x 1088p60
Sensor	OmniVision OV5647	Sony IMX219	Sony IMX708	Sony IMX708	Sony IMX477	Sony IMX296
Sensor resolution	2592 × 1944 pixels	3280 × 2464 pixels	4608 x 2592 pixels	4608 x 2592 pixels	4056 x 3040 pixels	1456 x 1088 pixels
Sensor image area	3.76 × 2.74 mm	3.68 x 2.76 mm (4.6 mm	6.45 x 3.63mm (7.4mm	6.45 x 3.63mm (7.4mm	6.287mm x 4.712 mm (7.9mm	6.3mm diagonal

Comparaison des Propriétés des différentes caméras (2)

		diagonal)	diagonal)	diagonal)	diagonal)	
Pixel size	1.4 µm x 1.4 µm	1.12 µm x 1.12 µm	1.4 µm x 1.4 µm	1.4 µm x 1.4 µm	1.55 µm x 1.55 µm	3.45 µm x 3.45 µm
Optical size	1/4"	1/4"	1/2.43"	1/2.43"	1/2.3"	1/2.9"
Focus	Fixed	Adjustable	Motorized	Motorized	Adjustable	Adjustable
Depth of field	Approx 1 m to ∞	Approx 10 cm to ∞	Approx 10 cm to ∞	Approx 5 cm to ∞	N/A	N/A
Focal length	3.60 mm +/- 0.01	3.04 mm	4.74 mm	2.75 mmm	Depends on lens	Depends on lens
Horizontal Field of View (FoV)	53.50 +/- 0.13 degrees	62.2 degrees	66 degrees	102 degrees	Depends on lens	Depends on lens

	Camera Module v1	**Camera Module v2**	**Camera Module 3**	**Camera Module 3 Wide**	**HQ Camera**	**GS Camera**
Vertical Field of View (FoV)	41.41 +/- 0.11 degrees	48.8 degrees	41 degrees	67 degrees	Depends on lens	Depends on lens
Focal ratio (F-Stop)	F2.9	F2.0	F1.8	F2.2	Depends on lens	Depends on lens
Maximum exposure times (seconds)	0.97	11.76	112	112	670.74	15.5
Lens Mount	N/A	N/A	N/A	N/A	C/CS- or M12-mount	C/CS
NoIR version available?	Yes	Yes	Yes	Yes	No	No

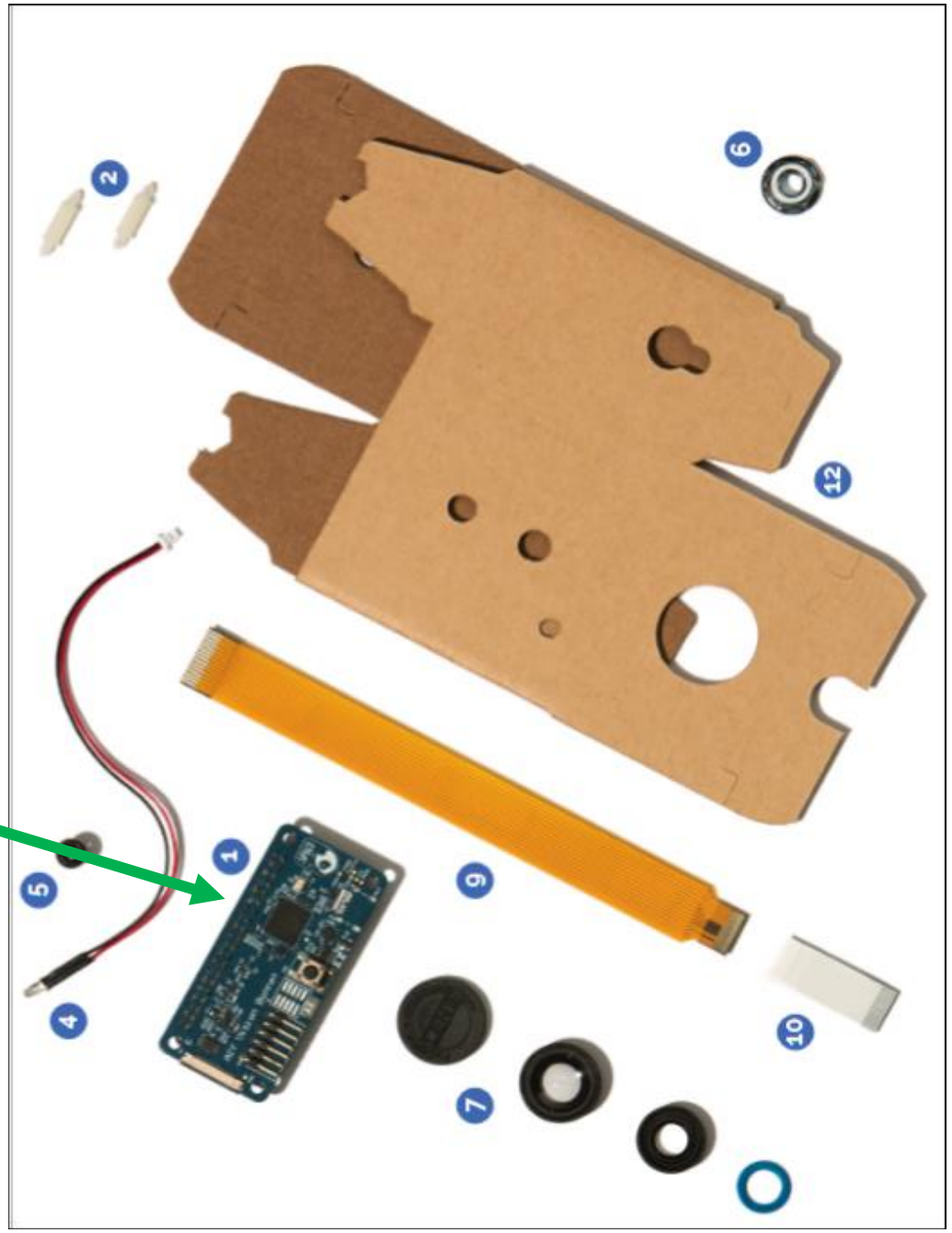
Resolution	Framerate (fps)	Result
1024x768 (a 4:3 aspect ratio)	< 42	The 1296x972 mode (4) will be selected, and the GPU will downscale the result to 1024x768.
1280x720 (a 16:9 wide-screen aspect ratio)	< 49	The 1296x730 mode (5) will be selected and downscaled appropriately.
1920x1080	30	This exceeds the resolution of both the 1296x730 and 1296x972 modes (i.e. they would require upscaling), so the 1920x1080 mode (1) is selected instead, despite it having a reduced FoV.
800x600	60	This selects the 640x480 60fps mode, even though it requires upscaling because the algorithm considers the framerate to take precedence in this case.

UNITÉ DE TRAITEMENT TENSORFLOW

Conçue à l'origine par Google, la bibliothèque **TensorFlow** permet la conception d'architectures de réseaux neuronaux, p. ex. convolutifs. L'apprentissage d'un réseau pour la conception d'un modèle, de chat p. ex., se fait avec de grands ensembles de données d'entraînement.

TensorFlow

Google a élaboré des réseaux neuronaux de reconnaissance d'images très efficaces. Leur résultat est meilleur avec un matériel spécialisé construit autour d'un TPU (**Tensor Processing Unit**), un circuit intégré conçu par Google pour l'apprentissage automatique. La puce de la carte VisionBonnet sert de TPU.



If you want to modify the button interface (such as to change the actual button), be sure to follow the wiring pinout as shown in figure 5.

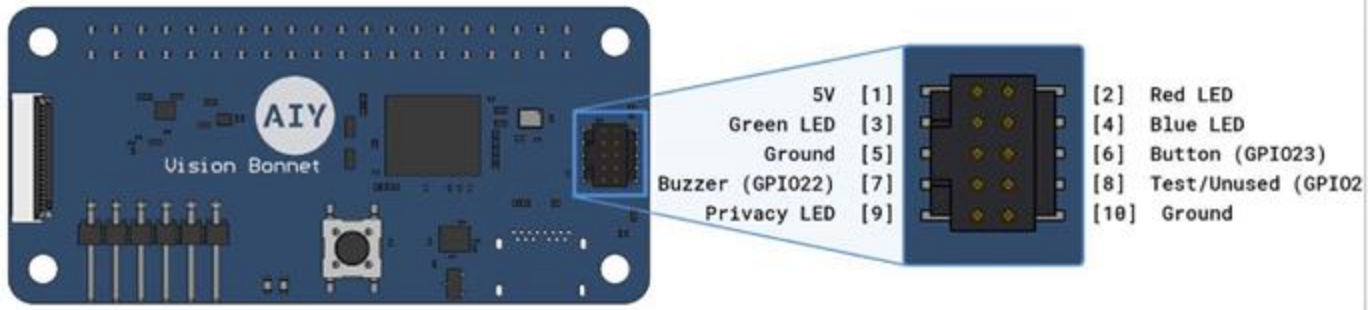
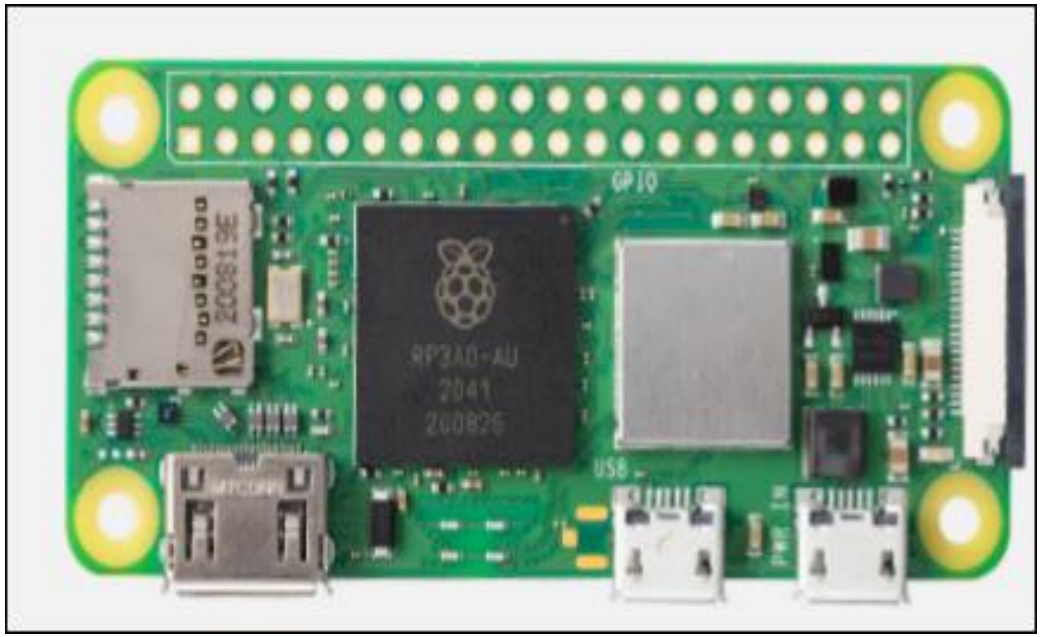


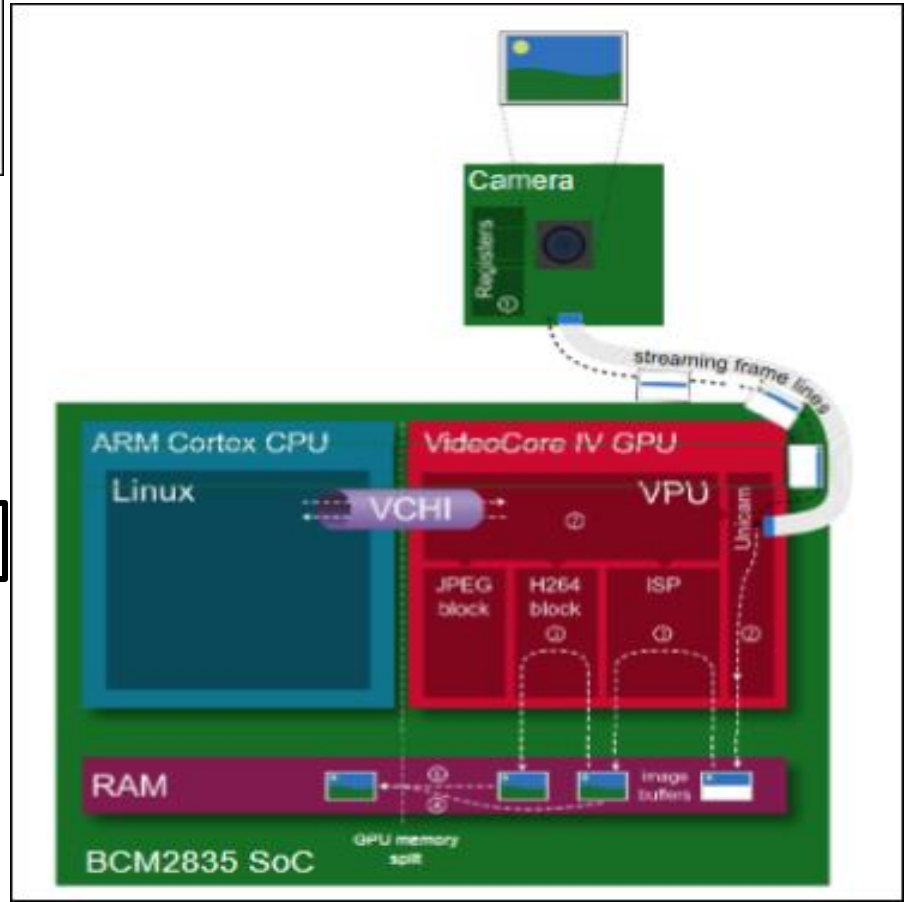
Figure 5. Pinout for the bonnet button connector

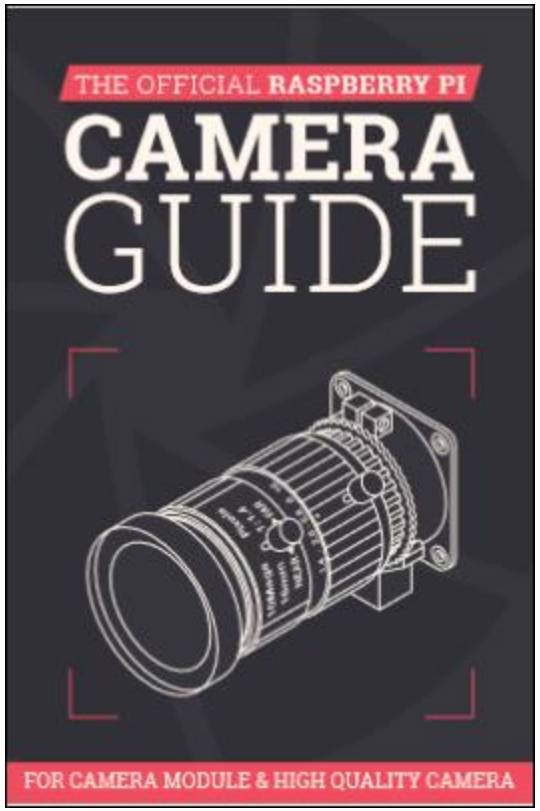
Note: The push button built onto the board functions exactly the same as the button connected to the button connector. They both activate GPIO23 on the Raspberry Pi.

CARTE VISO BONNET



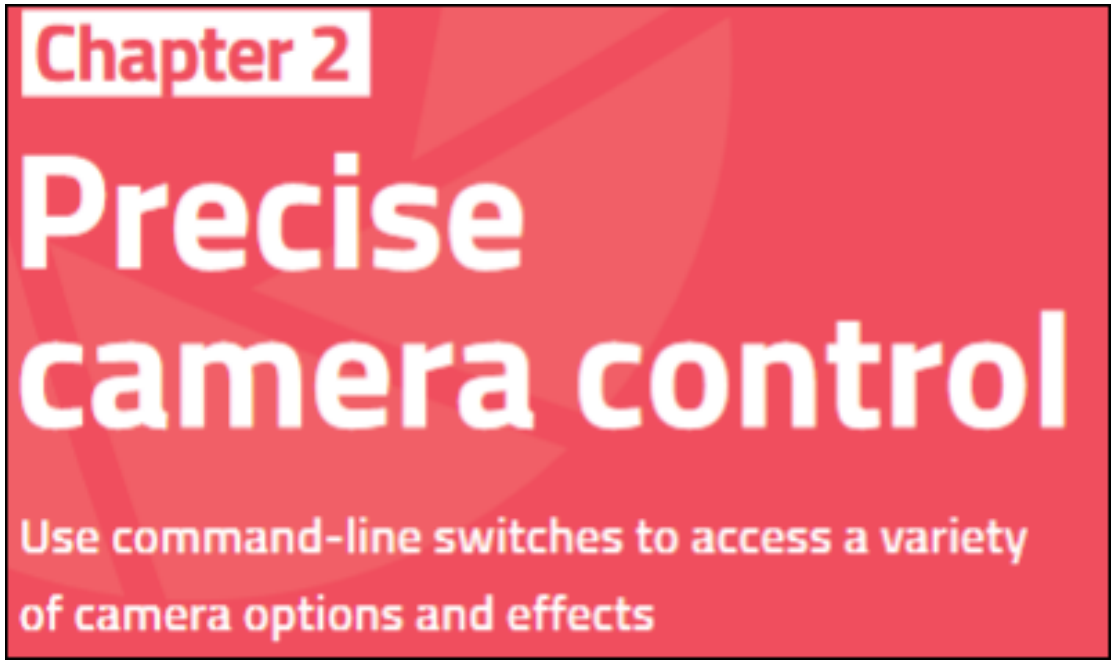
Rpi Zero 2W





Contents

PRISE PHOTO EN MODE COMMANDE



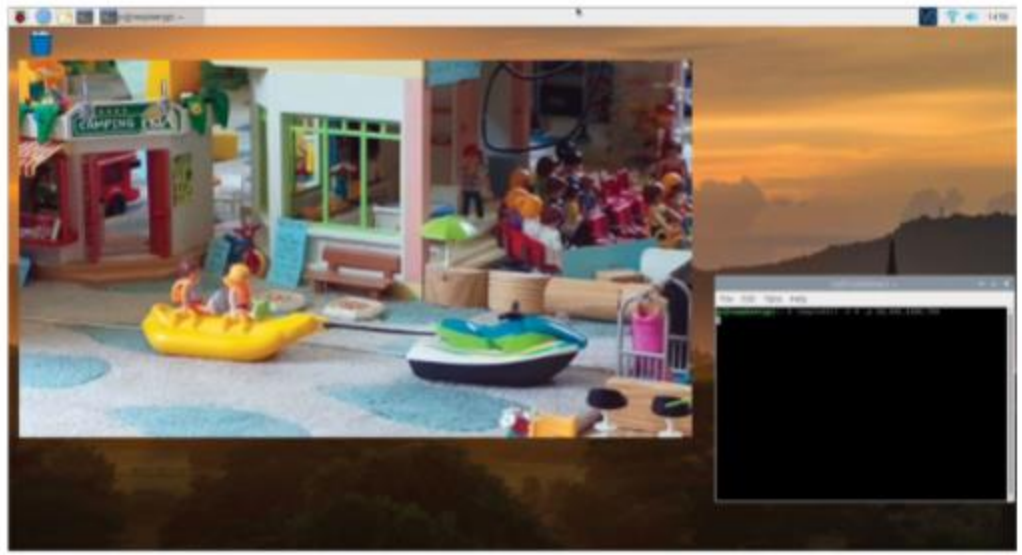
01 Preview mode

When taking stills or shooting video, one of the first things you might want to alter is the preview window that appears by default on the screen. First of all, if it's upside-down, just add `-rot 180` to your `raspistill` or `raspivid` command to rotate it. Also, adding `-hf` and/or `-vf` will flip the image horizontally and/or vertically.

Using the `-p` switch, you can set the window's on-screen position, along with its height and width. The `-p` switch takes four parameters: x co-ordinate, y co-ordinate, width, and height. So, for example:

```
raspistill -o image.jpg -p 20,100,1280,720
```

...would place the preview window's top-left corner at co-ordinate (20,100), with a width of 1280 pixels and height of 720 pixels.



The preview can be resized and positioned manually, and can also have its opacity adjusted

05 Still options

Let's take a look at some options that are specific to the `raspistill` command. As already mentioned, we use `-o` followed by a file name to output to a file, and the `-t` switch sets the shutter delay in milliseconds. For example, to save a photo taken after two seconds, use:

```
raspistill -t 2000 -o image.jpg
```

You can set the width and height of the image with `-w` and `-h`, each followed by a value – up to 4056 and 3040 (HQ Camera), 3280 and 2464 (Camera Module v2), or 2592 and 1944 (CM v1).

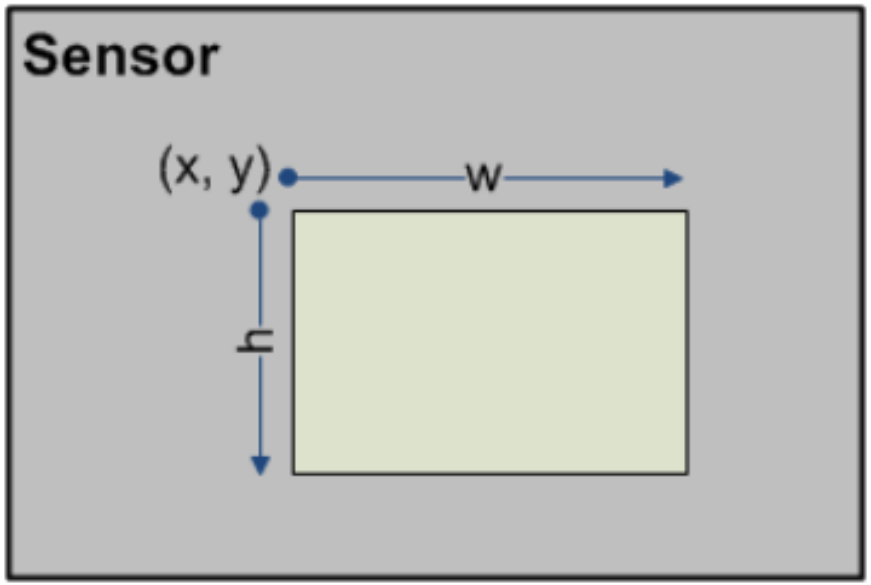
You can also set the quality of the JPEG image, using `-q`, from 0 to 100 – the latter is almost completely uncompressed. Alternatively, to save it as a lossless PNG (slower than using JPG), use `-e` (encoding) followed by `png`:

```
raspistill -o image.png -e png
```

For a full list of options, see Chapter 17. The `raspiyuv` command works in a similar fashion and offers most of the same options, apart from adding EXIF tags, but sends its YUV or RGB output directly from the camera component to file. To use RGB, add the `-rgb` switch.

Digital Zoom

Raspberry Pi Camera Board allows a region of the sensor to be used as the image capture area. This region, called region of interest (ROI), is specified as a normalized vector [x y w h] where x, y defines the top left corner and w and h specifies the width and height.



Reducing ROI while holding the output image size constant results in a digital zooming effect. The following MATLAB code varies the x and y parameters of the ROI to zoom into the lower right part of the sensor. The approximate area of the sensor being captured is indicated by a red rectangle.

Chapter 5

Control the camera from Python

Use the picamera library to access the camera in Python programs

04 Control camera settings

Brightness is just one of numerous settings available for the camera. Here's a list of the main options, along with their default values (and ranges where applicable):

```
camera.brightness = 50 (0 to 100)
camera.sharpness = 0 (-100 to 100)
camera.contrast = 0 (-100 to 100)
camera.saturation = 0 (-100 to 100)
camera.iso = 0 (automatic) (100 to 800)
camera.exposure_compensation = 0 (-25 to 25)
camera.exposure_mode = 'auto'
camera.meter_mode = 'average'
camera.awb_mode = 'auto'
camera.rotation = 0
camera.hflip = False
camera.vflip = False
camera.crop = (0.0, 0.0, 1.0, 1.0)
```

ch5listing3.py / Python 3

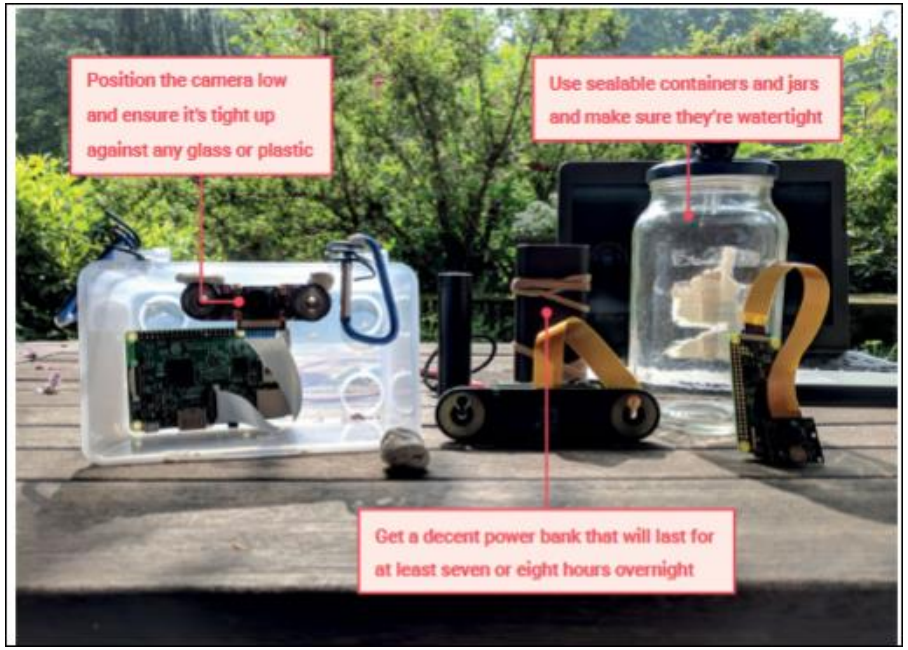
```
from picamera import PiCamera
from time import sleep

camera = PiCamera()

camera.start_preview()
for i in range(5):
    sleep(5)
    camera.capture('/home/pi/Desktop/image%s.jpg' % i)
camera.stop_preview()
```

DIFFERENTES APPLICATIONS DE LA CAMERA

Chapter 14

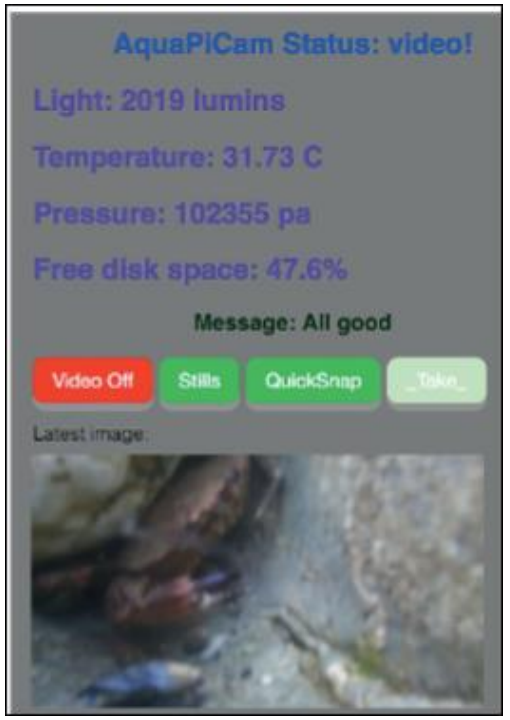
Install a bird box camera

Observe nesting birds without disturbing them



You'll still have to get pretty close to the water yourself

Now restart the dhcpcd daemon and set up the new wlan0 configuration:



Get great photos with night-vision cameras

Chapter 15

Live-stream video and stills

Stream video and regular stills to a remote computer

Chapter 16

Set up a security camera

Protect your home using motionEyeOS

APPRENTISSAGE AUTOMATIQUE

Mode d'apprentissage permettant à un ordinateur d'accomplir une tâche sans que celle-ci n'ait été explicitement programmée.

APPRENTISSAGE PROFOND

Technique d'apprentissage automatique à partir de modèles de données (et non d'algorithmes propres à une tâche).

CLASSIFICATION D'IMAGES

Inférence d'une classe d'appartenance (p. ex. chat ou humain) à partir d'une image.

INFÉRENCE

Conclusion probable établie sur la base d'indices et de renseignements. Le kit Vision peut par ex. analyser l'image d'un chat et en déduire par inférence que l'image contient un chat.

INTELLIGENCE ARTIFICIELLE

L'intelligence dont font montre les machines, par opposition à l'intelligence biologique – le « Y » d'AIY est celui de Yourself.

INTELLIGENCE-MACHINE

L'intelligence dont font montre les machines, par opposition à l'intelligence humaine. Terme utilisé pour préciser que l'IA est couplée à un matériel.

NEURONE

Cellule biologique véhiculant une impulsion électrique. Le cerveau humain en contient environ cent milliards.

NEURONE ARTIFICIEL (PERCEPTRON)

Fonction logicielle conçue pour imiter le comportement d'un neurone.

NŒUD

Élément de base représentant typiquement une valeur, un groupe de valeurs ou une fonction. Les nœuds sont généralement organisés en structures connectées appelées graphes (ou arbres binaires). Dans un réseau neuronal, les neurones artificiels forment les nœuds d'un graphe.

RECONNAISSANCE D'OBJETS

En vision artificielle, technique de recherche et d'identification d'objets.

RÉSEAU NEURONAL

Système connecté (construit à partir de neurones/nœuds de données connectés). Inspiré des réseaux neuronaux biologiques.

RÉSEAU NEURONAL CONVOLUTIF

Type de réseau neuronal artificiel conçu en particulier pour l'analyse d'images.

VISION ARTIFICIELLE (en informatique)

Application chargée d'analyser une image (ou une séquence d'images) et d'en extraire l'information utile.

VISION ARTIFICIELLE (machine)

Technologie et méthodes fournissant un moyen d'analyse et d'inspection automatiques par traitement d'images.

Wikipédia a un glossaire (en anglais) de l'intelligence artificielle : **mappi.ec/1joDjjo**

ANALYSE D’IMAGE

What We See

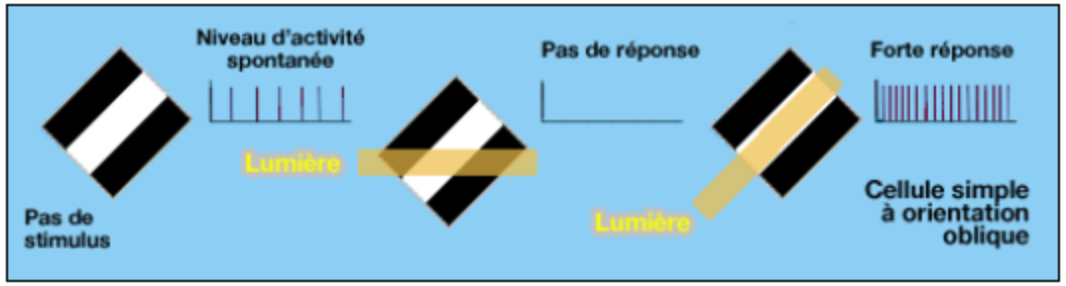
```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48
```

What Computers See

MODÈLE DE VISION DE WIESEL ET HEBEL (1957)

Les cellules simples, dans la couche IV principalement, possèdent un champ central en forme de barre (" on ") flanqué par un champ récepteur périphérique (" off ") ou inversement.

Par exemple, certaines cellules sont sensibles à des orientations obliques.



Ces cellules, appelées également détecteurs de barres ou détecteur d'angles, sont sensibles à l'orientation, mais aussi à la position dans le champ. On en trouve pour toutes les directions, mais les réponses sont plus nombreuses pour les directions proches de l'horizontale et de la verticale.

Les cellules simples répondant à une orientation identique sont situées dans une même colonne verticale du cortex strié (50 µm de largeur). Ces colonnes sont plus étroites et perpendiculaires aux colonnes de dominance oculaire.

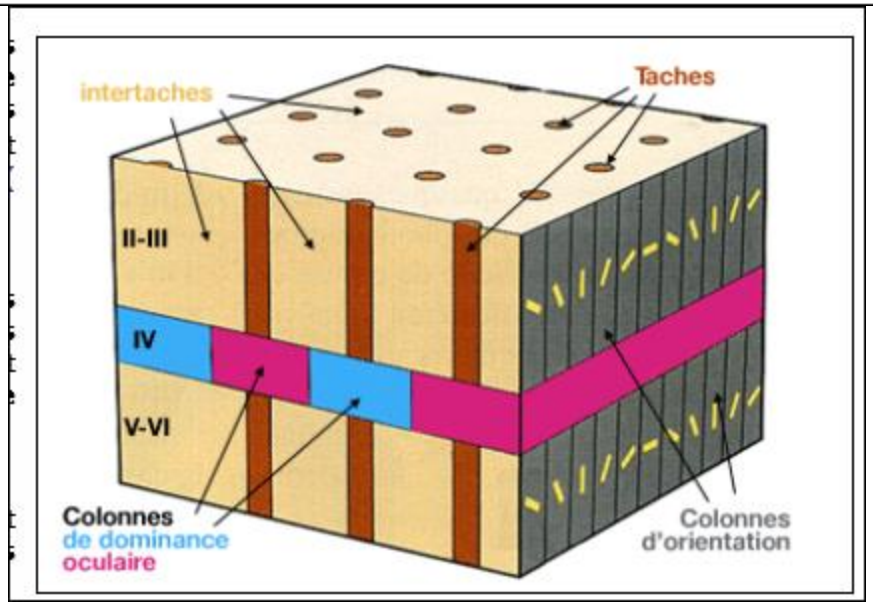
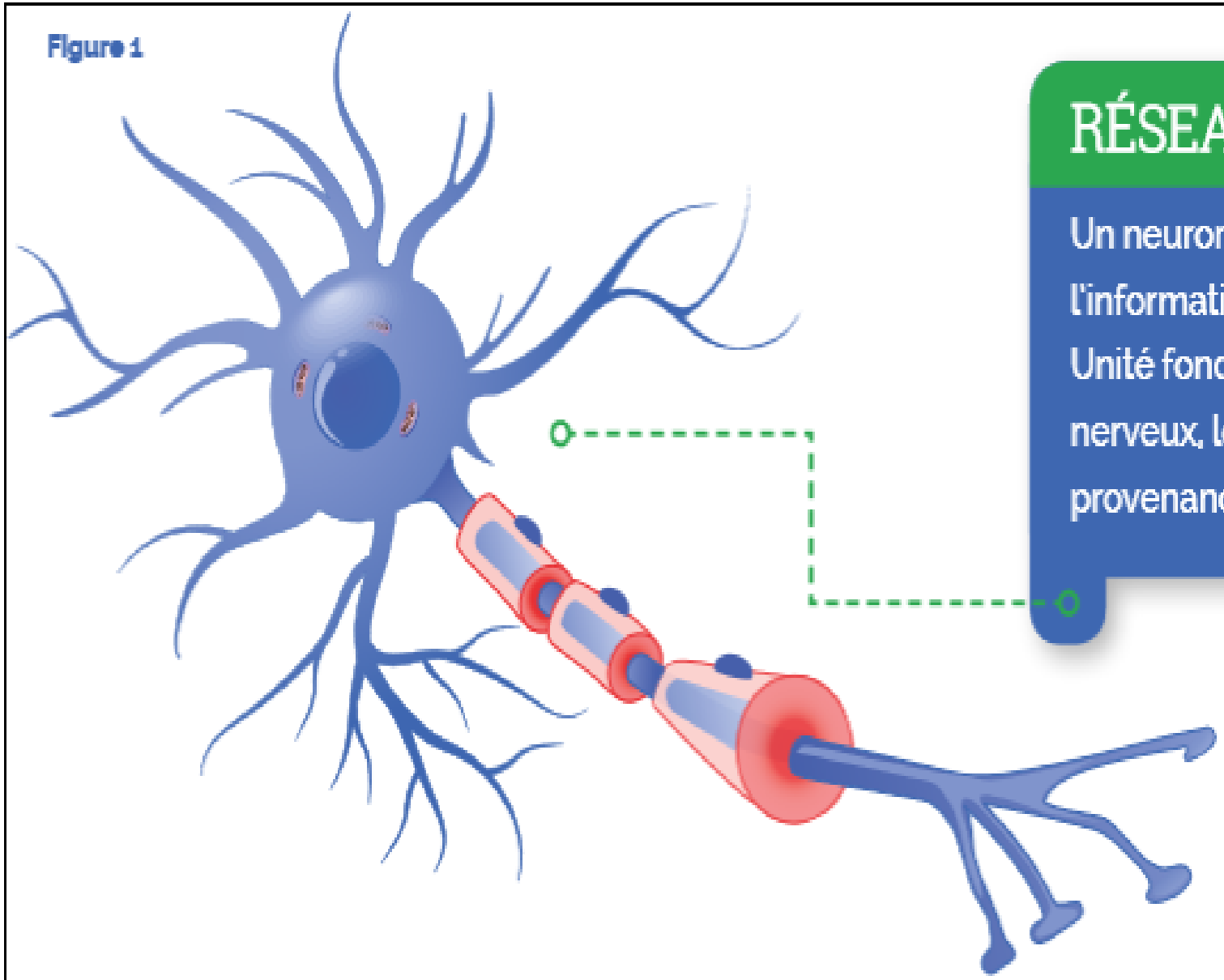


Figure 1



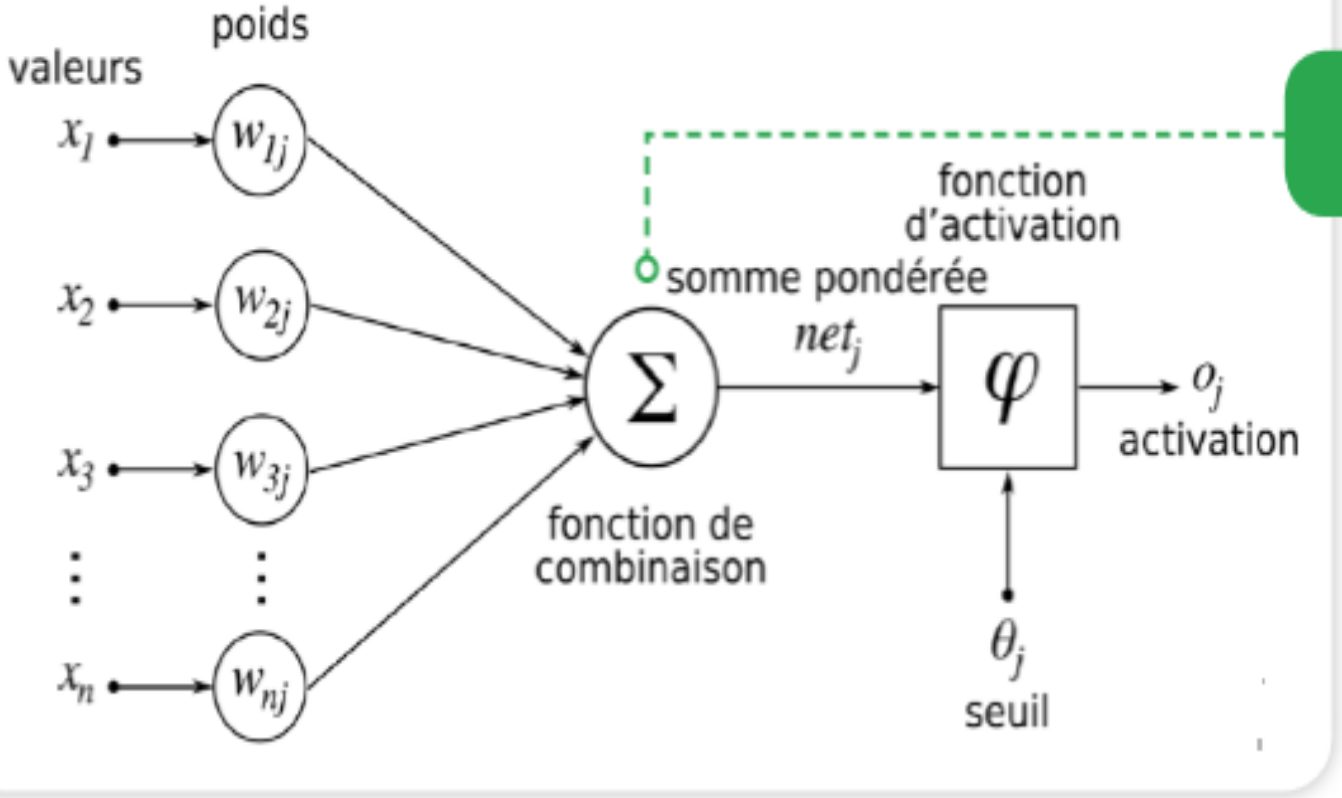
RÉSEAU BIOLOGIQUE

Un neurone est une cellule excitable qui transmet de l'information via des signaux électriques et chimiques. Unité fondamentale du cerveau et du système nerveux, le neurone traite les entrées sensorielles en provenance du monde extérieur.

Crédit image : Rossenblatton Perceptron, Wikipédia

NEURONE ARTIFICIEL

Représentation idéalisée d'un neurone biologique, modélisé à l'aide d'algorithmes.



FONCTION DE COMBINAISON

Les réseaux neuronaux sont formés en reliant des neurones artificiels dans un graphe. Les valeurs d'entrée, des pixels, sont multipliées par des poids et un biais est ajouté. Une valeur issue d'un calcul supplémentaire est transmise au neurone suivant.

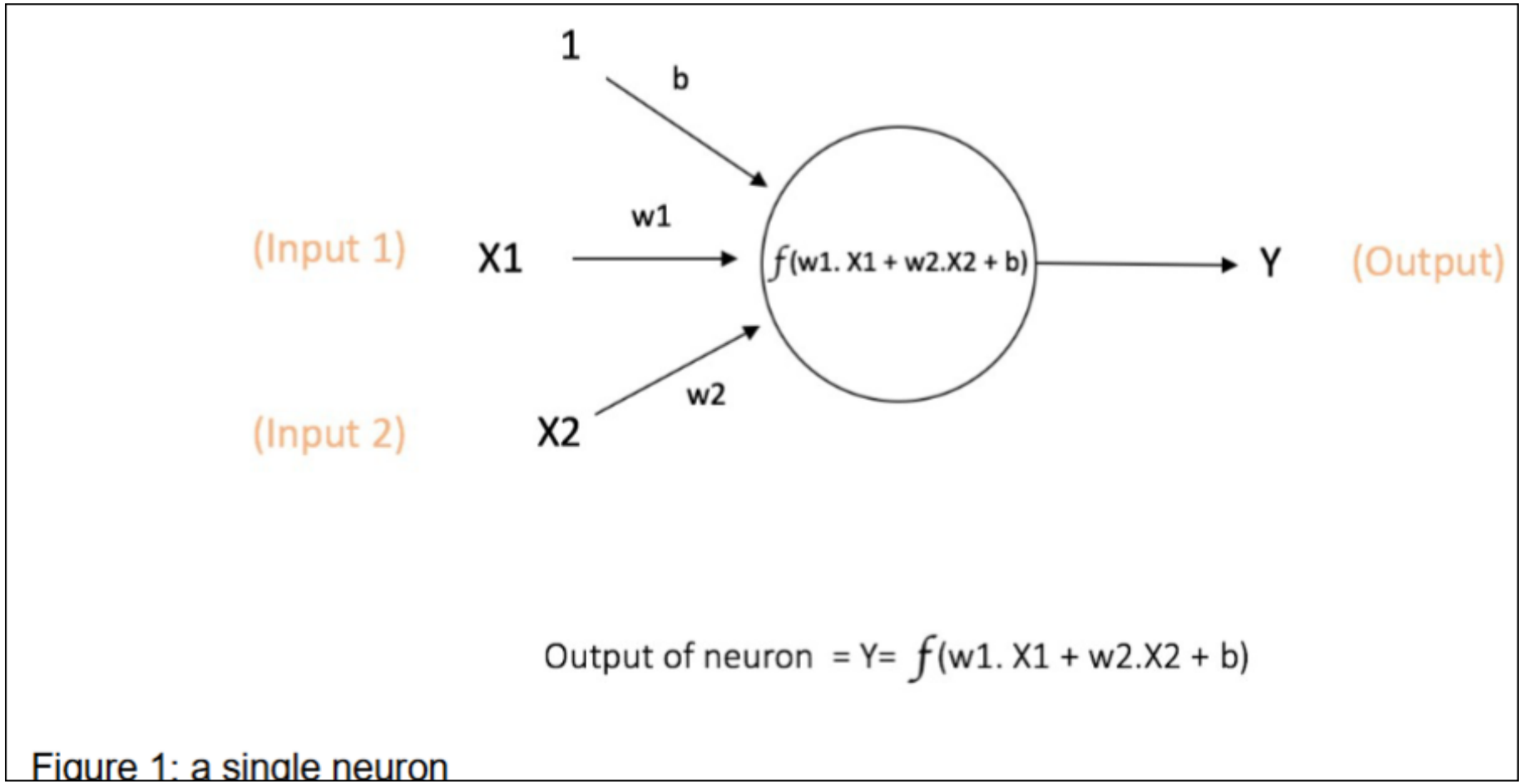
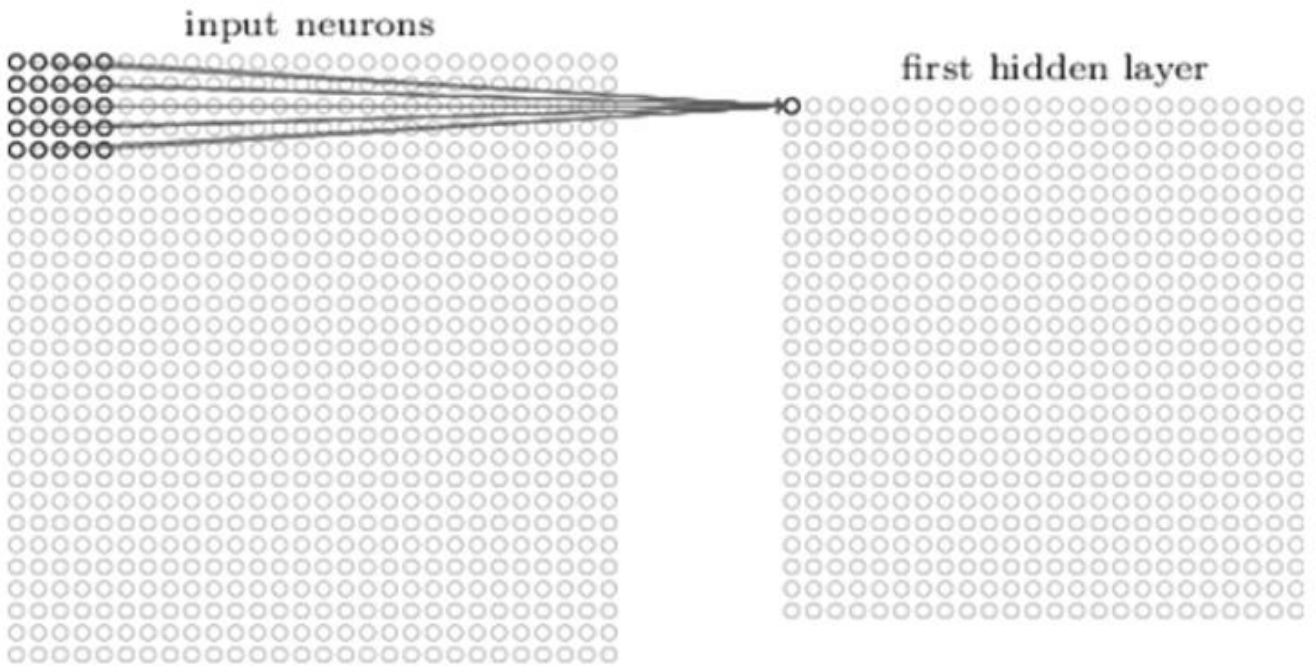


Figure 1: a single neuron

What We See

```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48
```

What Computers See



Visualization of 5 x 5 filter convolving around an input volume and producing an activation map

Blue channel

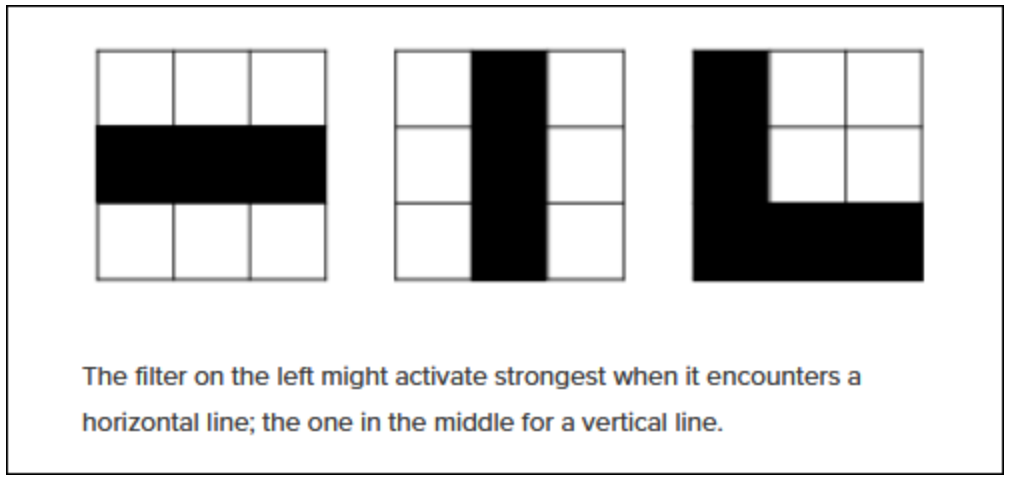
Green channel

Red channel

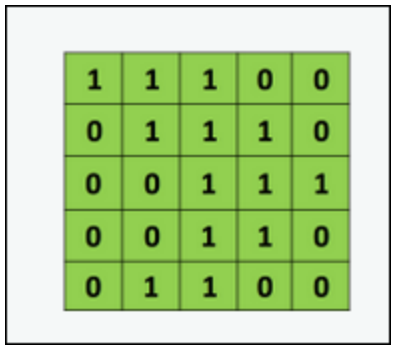
	171	200	19	6	...	26
	24	56	230	1	...	8
1	120	67	89	107	...	13
2	12	216	145	26	...	181
3	0	16	4	45	...	44
4	0	78	90	167	...	25
...	...	...	...	...	...	...
64	12	67	82	141	...	12
	1	2	3	4	...	64

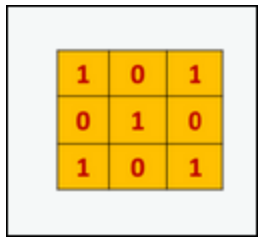
89	
18	8
81	71
56	...
...	7
12	

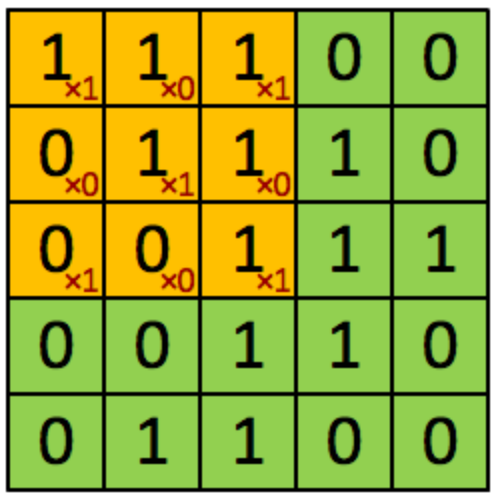
Image array: [64 x 64 x 3]



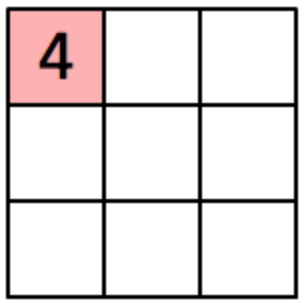
Fenêtre glissante =filtre=kernel (ici bloc 3x3) déplacée de 1 ou plusieurs « cases » à chaque fois.
Image à 3 couleurs d'où filtre ou kernel de 3x3x3



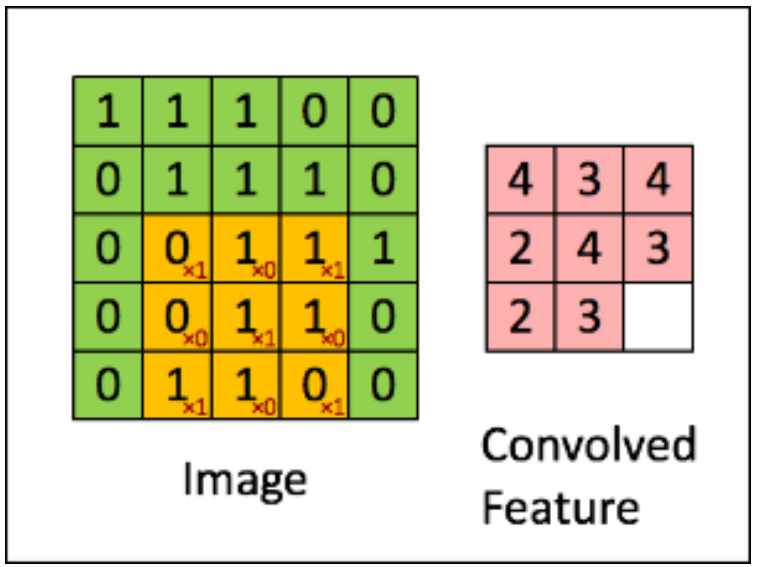




Image

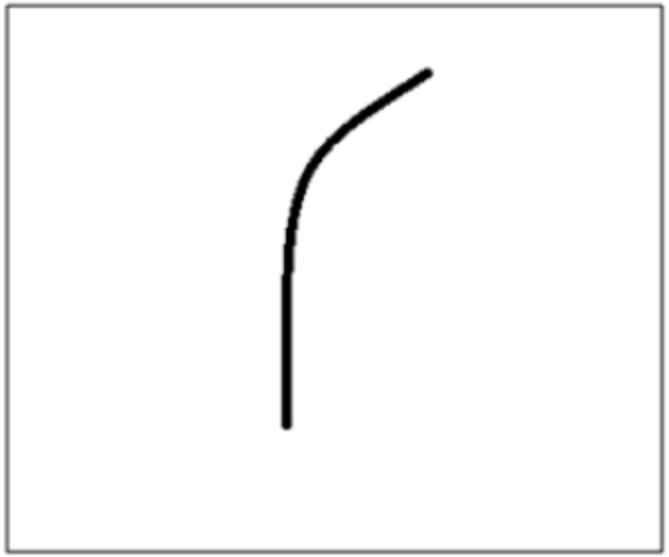


Convolved Feature



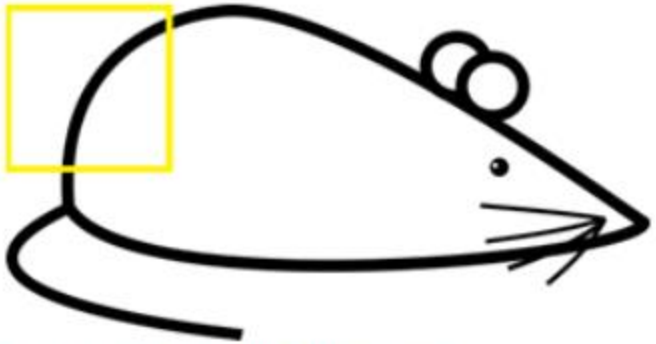
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



Visualization of a curve detector filter

Original image



Visualization of the filter on the image

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

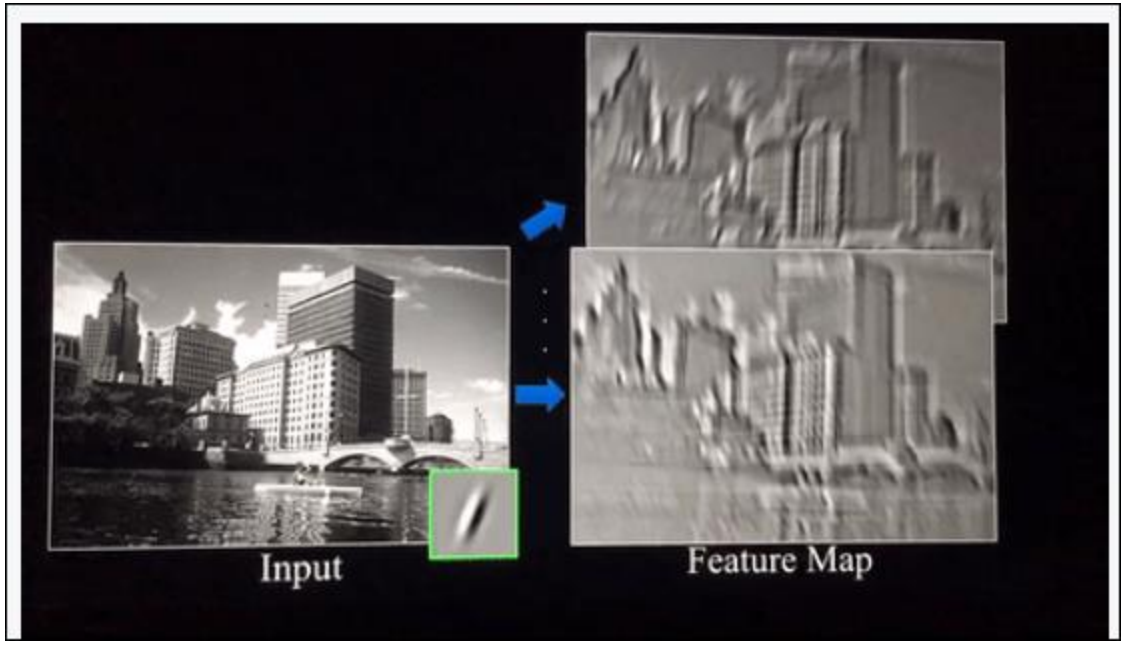
Pixel representation of receptive field

*

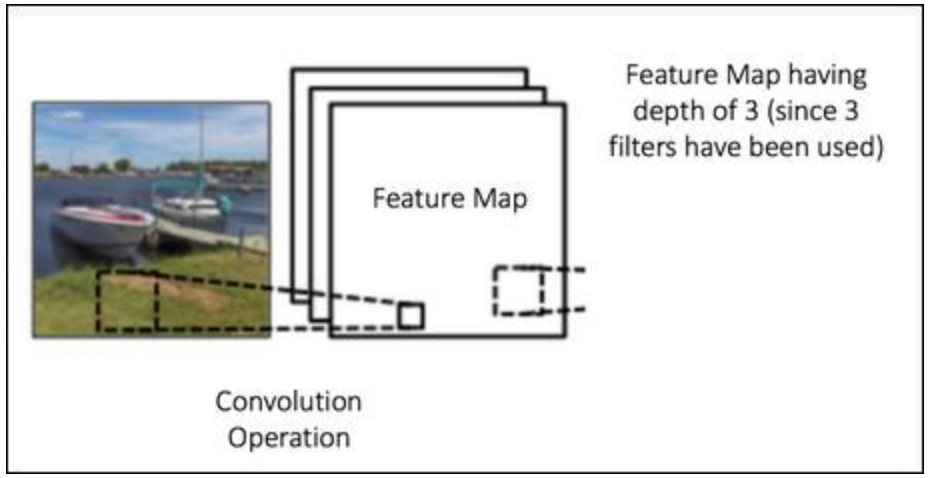
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = 0



Résultat de convolution avec 2 filtres, \ et / .
Utyilisation de filtres pour chacune des 3 couleurs.
Ajustement des paramètres au cours de l'apprentissage.
La taille de la convlution feature est déterminée par 3 paramètres:
La profondeur=nombre de filtres: les pas du kernel et le zéro-padding où on mets des zéros aux bordures.



Profondeur

Pour chaque convolution introduction de l'opération ReLu= Rectified Linear Unit; On remplace les valeurs négatives des pixels par des zéros. On peut utiliser d'autres fonctions, tangente, sigmoide etc…

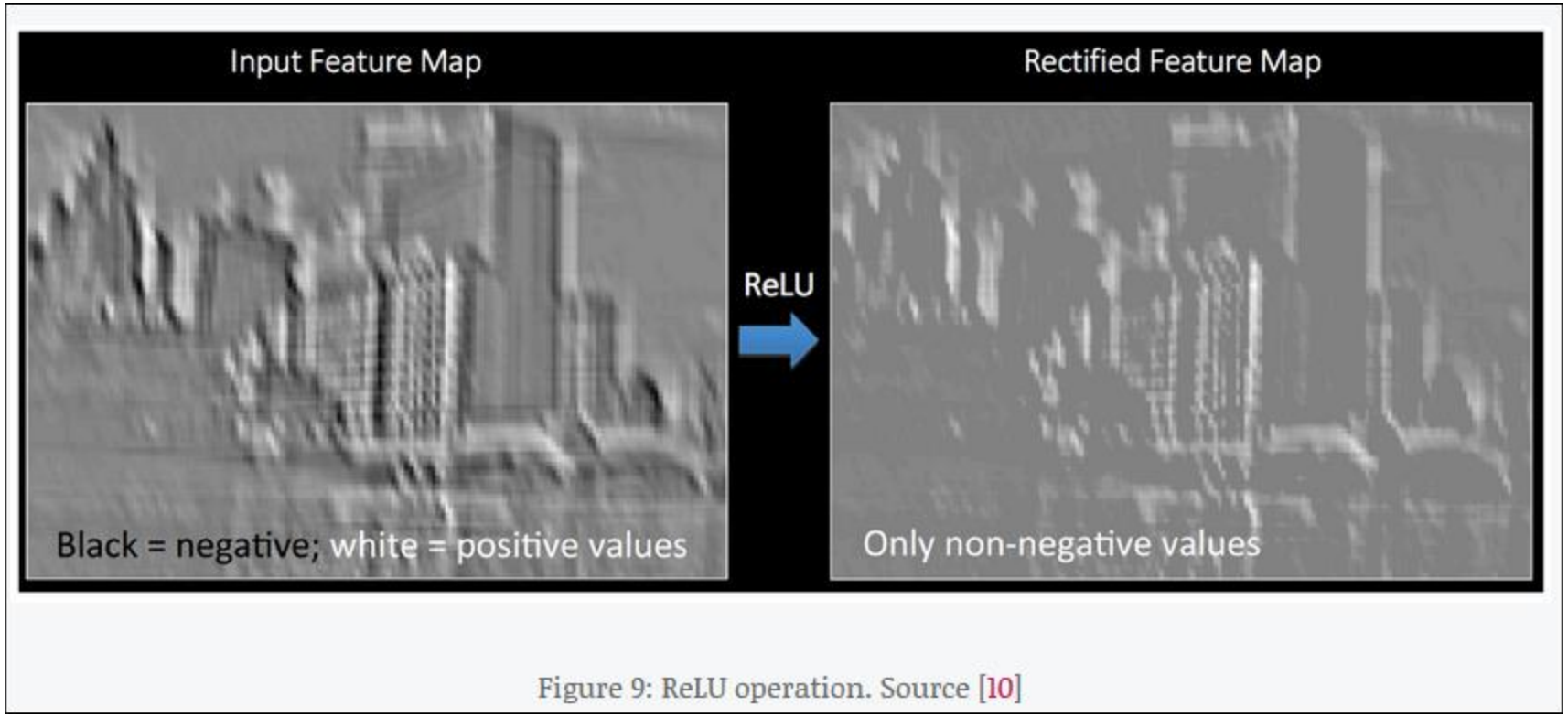


Figure 9: ReLU operation. Source [10]

Après ReLu, pooling pour diminuer le nombre de dimension de chacune des feature map. On prend les valeurs max. ou les moyennes etc...

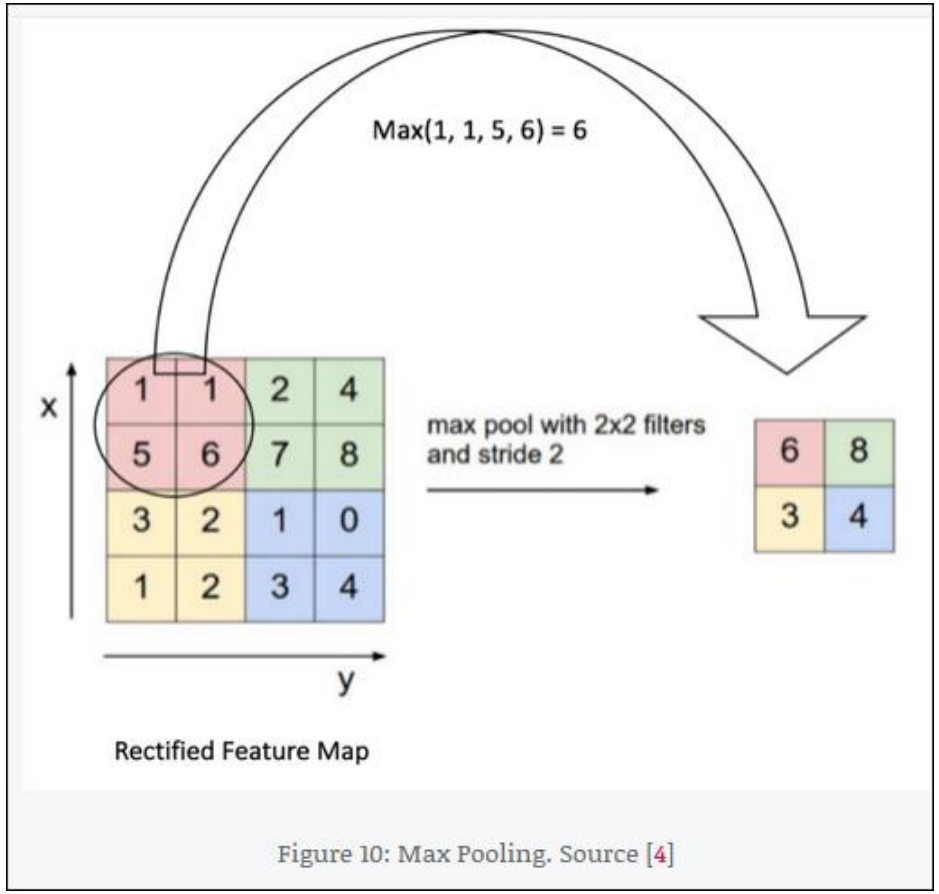


Figure 10: Max Pooling. Source [4]

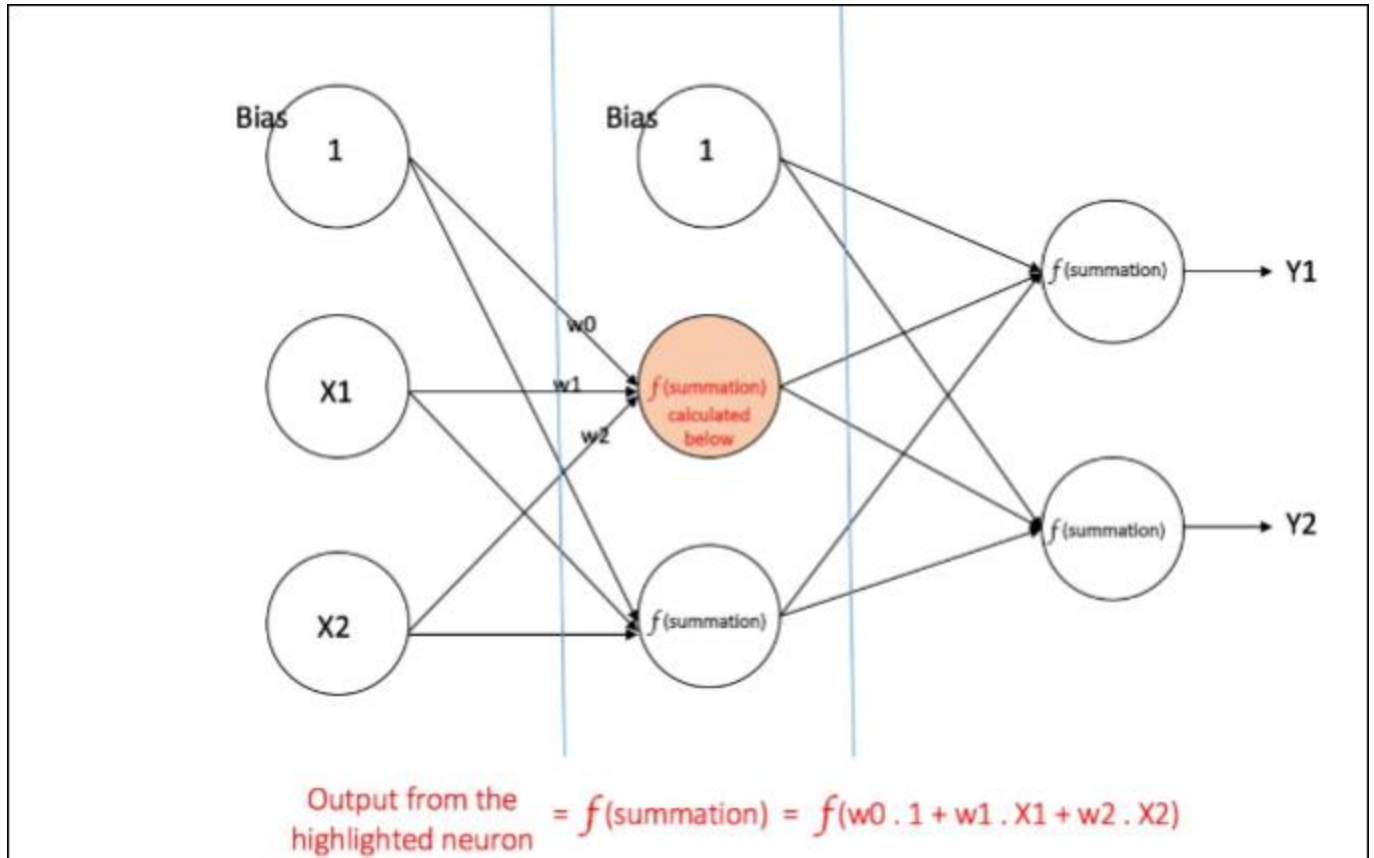


Figure 4: a multi layer perceptron having one hidden layer

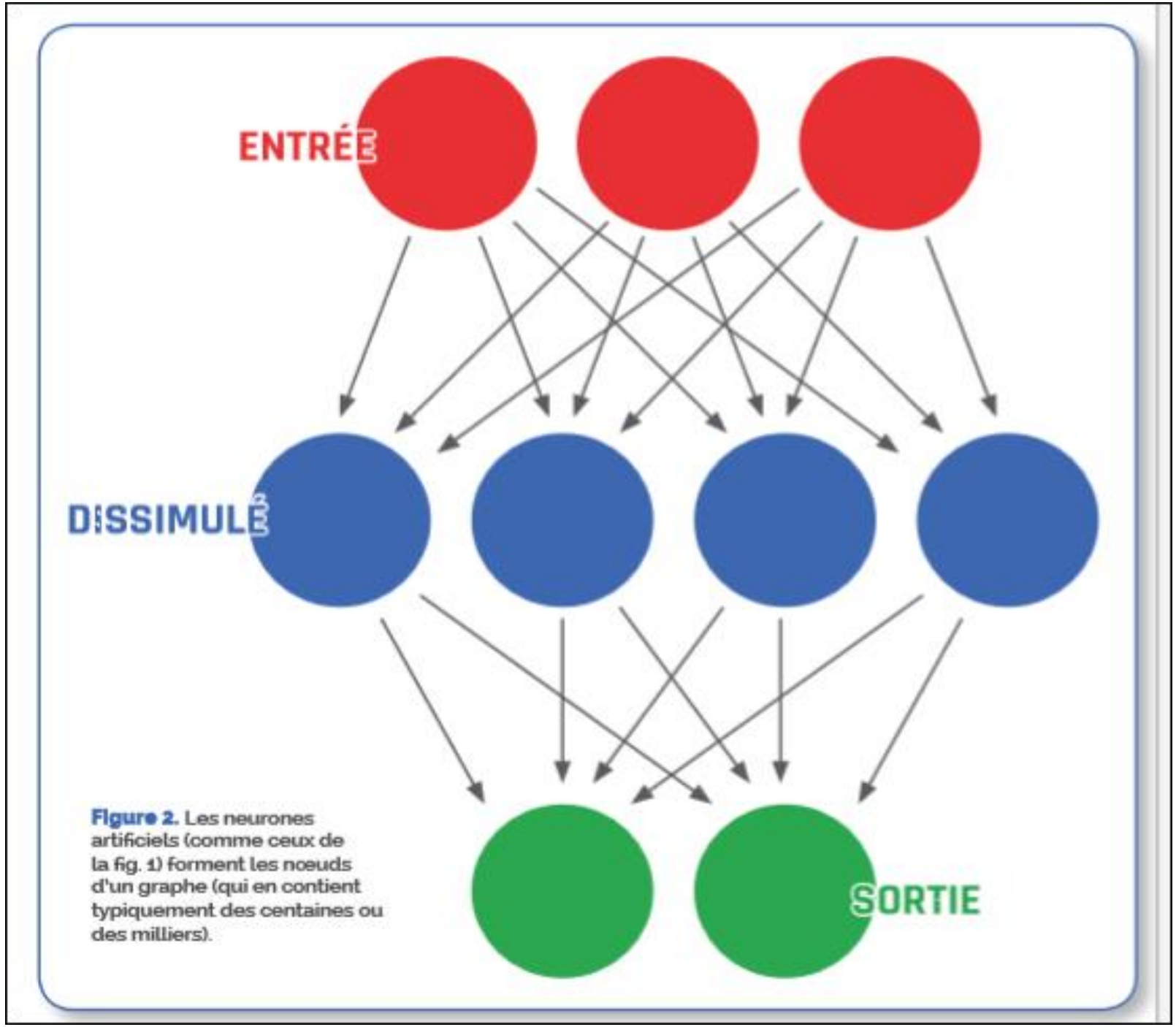


Figure 2. Les neurones artificiels (comme ceux de la fig. 1) forment les noeuds d'un graphe (qui en contient typiquement des centaines ou des milliers).

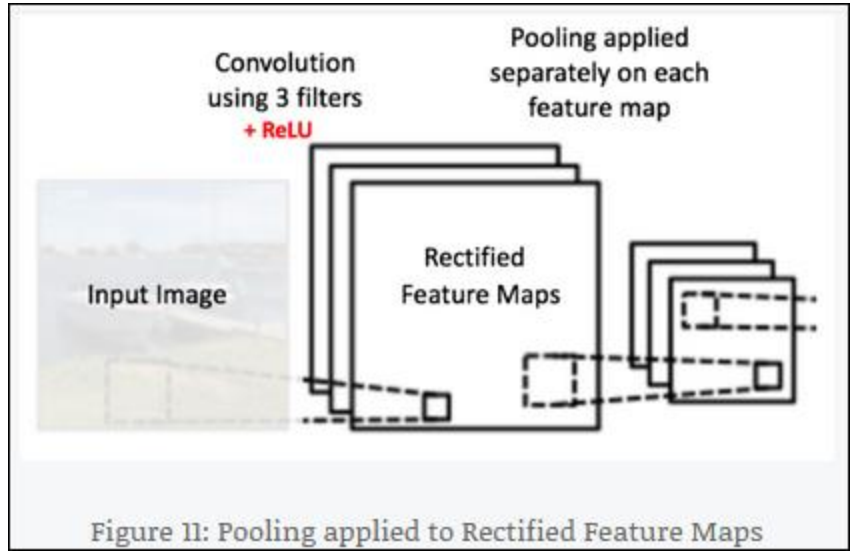


Figure 11: Pooling applied to Rectified Feature Maps

Figure 12 shows the effect of Pooling on the Rectified Feature Map we received after the ReLU operation in **Figure 9** above.

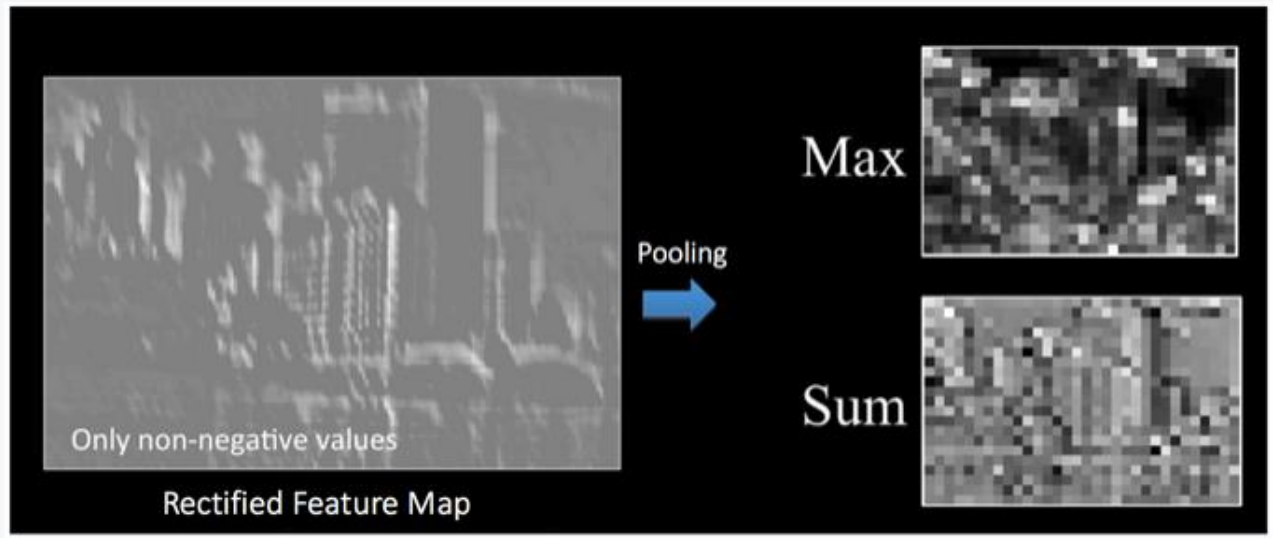
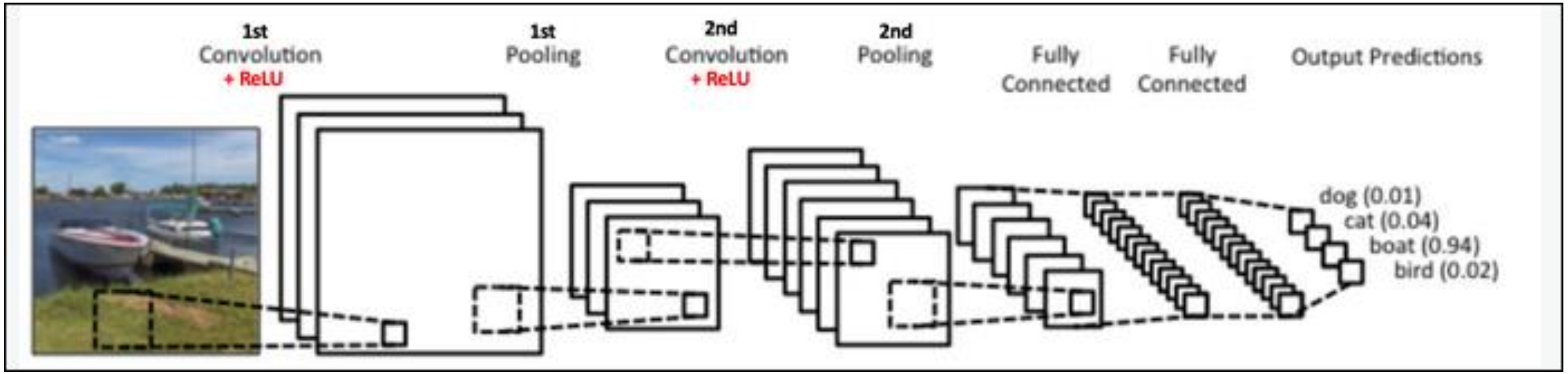


Figure 12: Pooling. Source [10]

CONVOLUTION



Convolution avec 3 filtres, ReLu sur les 3 features map , pooling , Conv. Avec 6 flitres ReLu sur les 6 filtres Pooling.

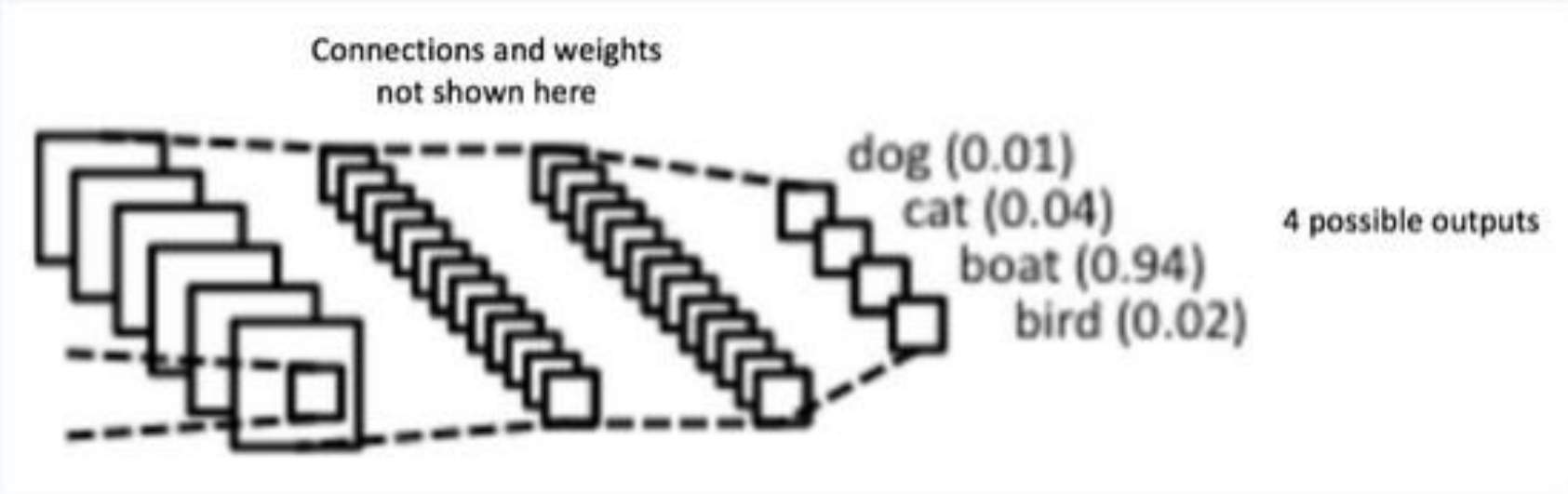
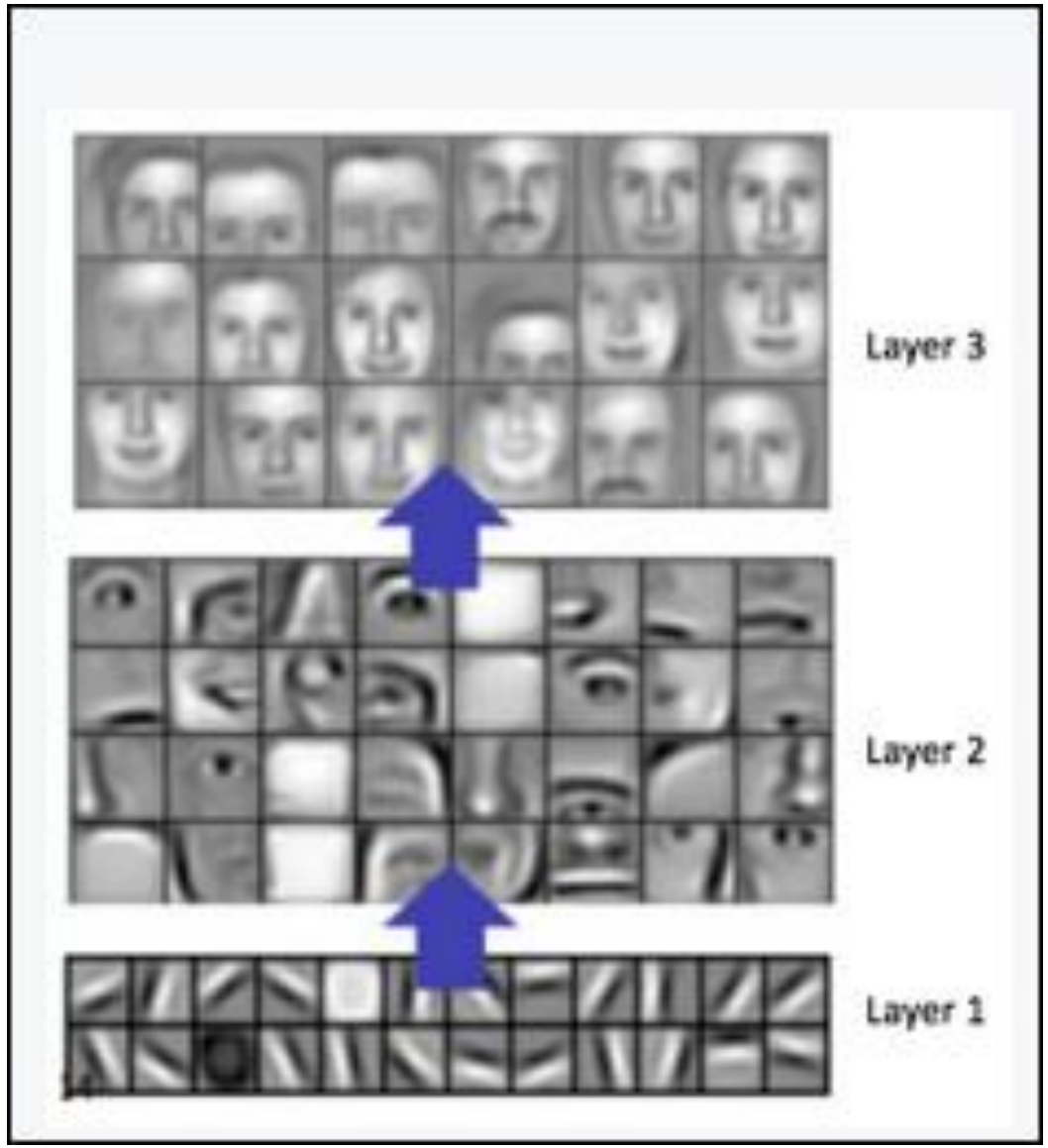


Figure 14: Fully Connected Layer –each node is connected to every other node in the adjacent layer

Fully connected, c'est un modèle de perceptron à multicouches ou chaque neurone d'une couche est connecté au neurone de la couche suivante, la dernière en utilisant la règle du softMax. Dans le cas présent on aboutit à 4 classes possibles dont les probabilités ne dépassent pas 1.


Layer 3
Layer 2
Layer 1

Entraînement (éducation) du réseau de convolution par marche arrière.

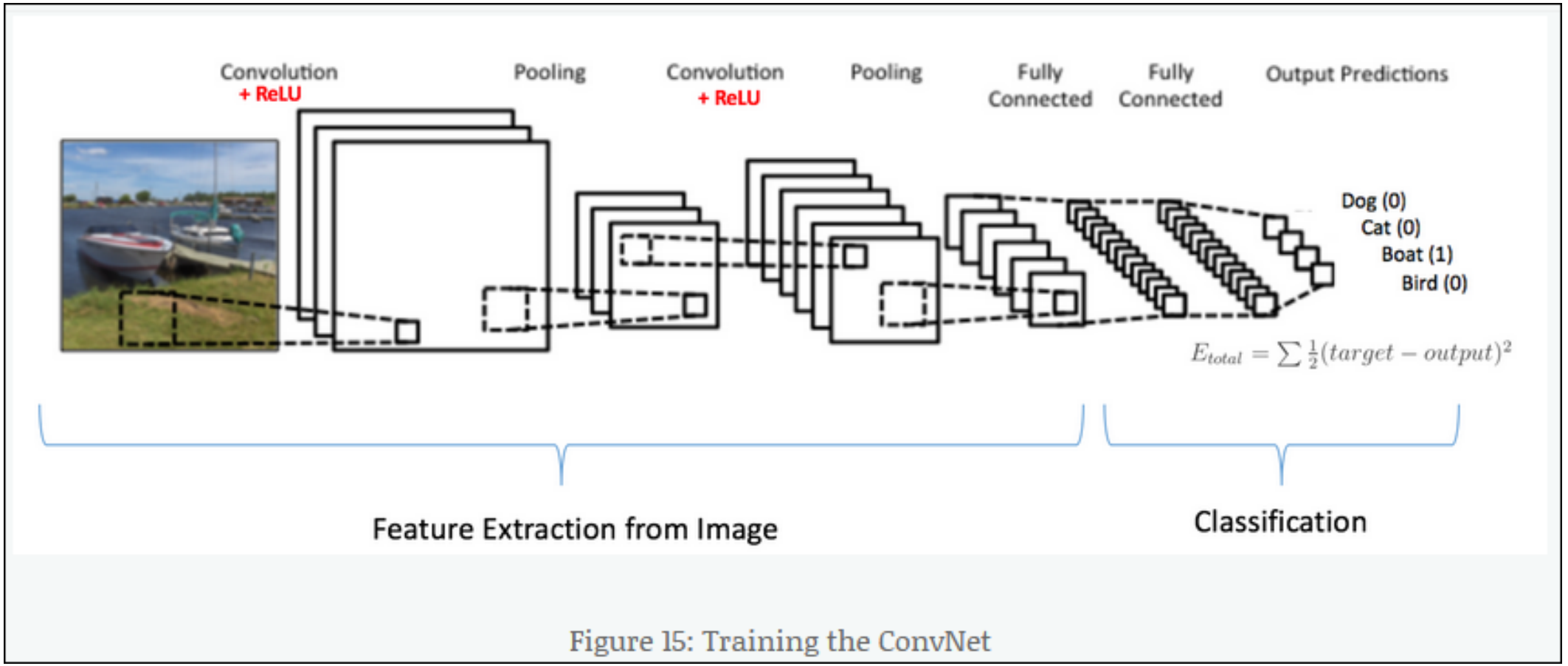


Figure 15: Training the ConvNet

Tensorflow=librairie générale de Google. Reconnaissance d'image
Une fois que TF installé sur le Rpi, télecharger les modèles préentrainés; `https://github.com/tensorflow/models`

Première classification

Nous allons lancer le script python classify_image.py qui se trouve dans le dossier tutorials/image/imagenet du dépôt cloné.

Le script utilise le modèle Inception entrainé avec la banque d'images ImageNet. Cette base de données d'images est très utilisée dans le domaine de l'intelligence artificielle. Elle est composée de près de 15 millions d'images couvrant toutes sortes de concepts. Cette médiathèque est parcourable à l'adresse http://www.image-net.org/index .

Ce tutoriel étant rédigé sous windows, les séparateurs de dossiers utilisés sont des "\", bien sûr il convient de les adapter en fonction de votre système d'exploitation.

```
cd tutorials\image\imagenet
```

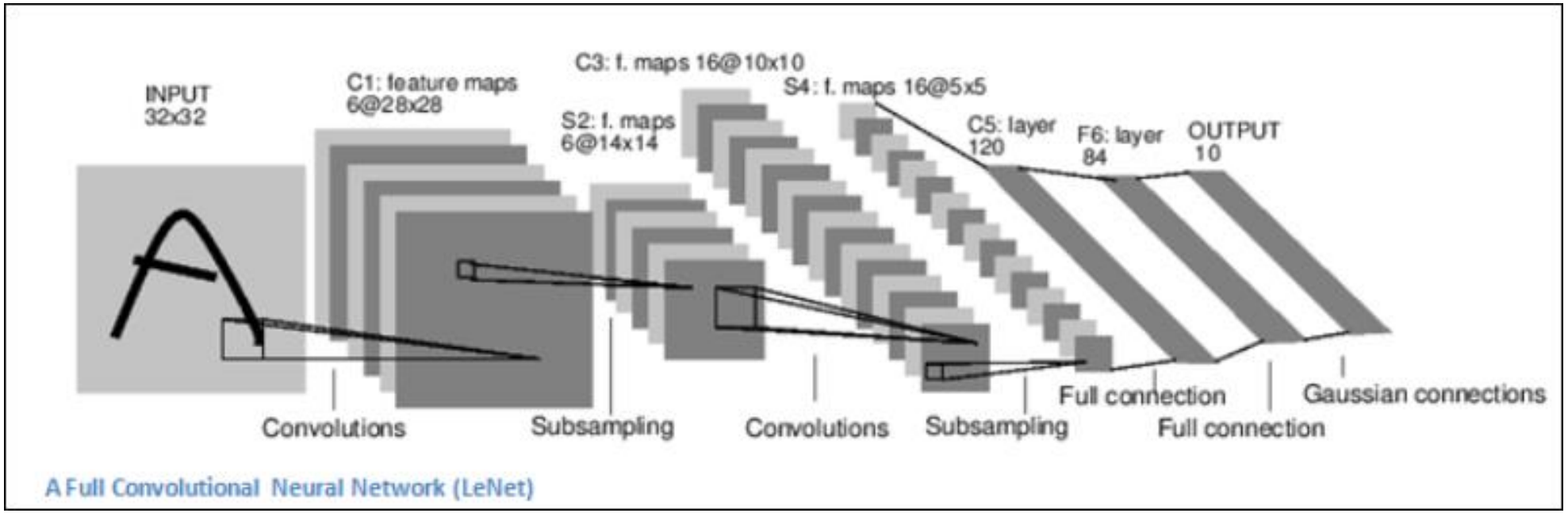
```
cd tutorials\image\imagenet
python classify_image.py --model_dir=tmp
```

A la première exécution, les fichiers relatifs au modèle seront automatiquement téléchargés dans le dossier "tmp" . L'archive contient un total de 5 fichiers.

– **classify_image_graph_def.pb** => Contient le fichier de définition du graph du modèle TensorFlow. C'est le coeur du modèle puisqu'il contient le réseau de noeuds qui traitent les données en entrée.
– **cropped_panda.jpg** => image d'exemple
– **imagenet_2012_challenge_label_map_proto.pbtxt =>**Contient les informations de mapping pointant vers les différents labels.
– **imagenet_synset_to_human_label_map.txt =>** Représentation des labels sous forme textuelle.
– **LICENSE** => contient les informations relatives aux conditions d'utilisation

Après avoir téléchargé automatiquement ces fichiers, le script classify_image.py appliquera un traitement du modèle Inception sur l'image d'exemple cropped_panda.jpg et suggérera des tags classés par score décroissant de probabilité. Nous pouvons voir que le modèle estime reconnaître une photo représentant "giant panda, panda, panda bear, coon bear, Ailuropoda melanoleuca" avec une probabilité supérieure à 89%.

Input -> Conv -> ReLU -> Conv -> ReLU -> Pool -> ReLU -> Conv -> ReLU -> Pool ->Fully Connected



A Full Convolutional Neural Network (LeNet)

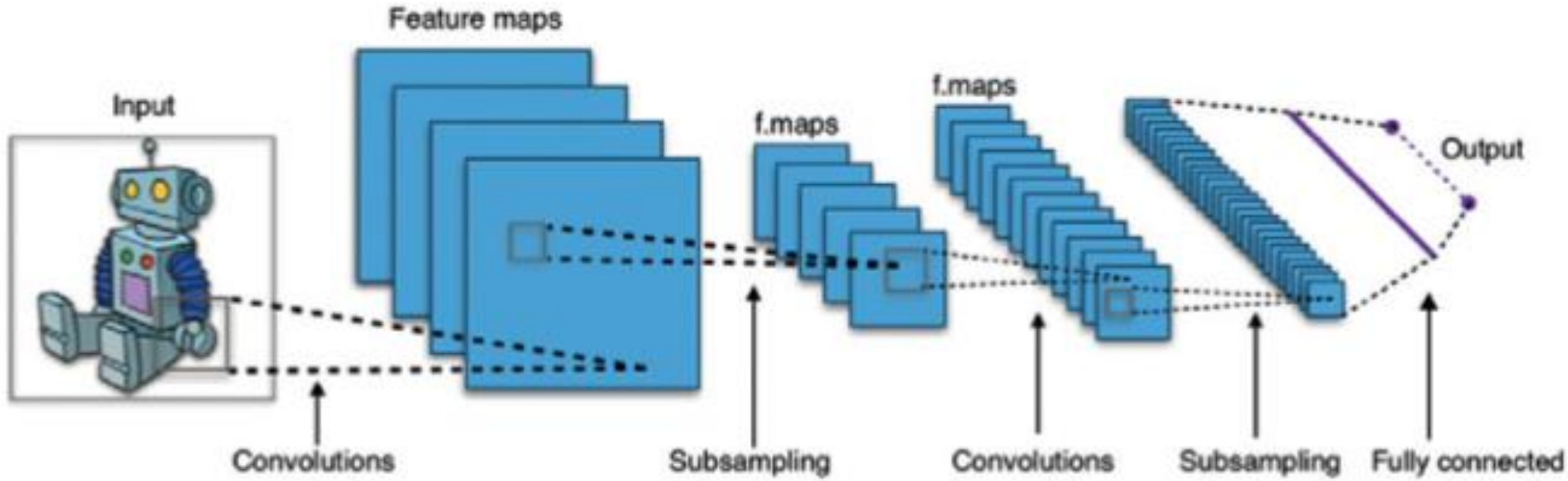


Figure 3. Il existe de nombreux types de réseaux neuronaux, chacun conçu pour une utilisation particulière. Dans le cas de la vision artificielle et du traitement du langage naturel, le type le plus courant est le réseau neuronal convolutif. Inspiré du cortex biologique, il décompose les images en petits carrés afin d'identifier des formes et caractéristiques communes (arc, cercles, points et lignes). Crédit image : Wikimedia.

> **ÉTAPE 01**
Neurone biologique. (fig;) Modèle vision humaine

> **ÉTAPE 02**
Neurone artificiel ou perceptron.
(fig)Entrées= valeurs des pixels

> **ÉTAPE 03**
Poids et biais. Valeurs d'entrée pondérée (poids ajusté en fonction de l'apprentissage). Sommation pour passer au neurone suivant

> **ÉTAPE 04**
Réseaux neuronaux. Couches de neurones intermédiaires, chacune avec un trait caractéristique, p. ex. une pour le contour, une pour chacune des 3 couleurs, une pour les yeux, une pour les sourcils etc.. Plus il y a de couches meilleur sera la précision

> **ÉTAPE 05**
Réseaux neuronaux profonds

> **ÉTAPE 06**
Réseau neuronal biologique. Modèle de Hubel et Visel: Des groupes de neurones du cortex sont regroupés en taches spécifiques de reconnaissance et se superposent pour reformer l'image entière.

> ÉTAPE 07

Réseau neuronal convolutif. Imitation du réseau neuronal (Yann LeCun et al) . Surtout utilisés pour la reconnaissance d'images.

> ÉTAPE 08

Décomposition. Sous-échantillonnage par des kernels. (fig)

> ÉTAPE 09

Apprentissage. Le réseau neuronal convolutif est soumis à de nombreuses iamges, P. ex; de chat ou chien de telle sorte à ls différencier. Puis il est soumis à une image qu'il n'a jamais vue pour tester son acuité.

> ÉTAPE 10

Chargement des modèles. Exemple de 3 modèles. Classification images à partir de 101 modèles de la vie courante. Détecteur de visage. Détecteur d'émotion.

> ÉTAPE 11

Codes de test

Détecteur de Joie

```
#!/usr/bin/env python3
# Copyright 2017 Google Inc.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
"""Joy detection demo."""
import argparse
import collections
import contextlib
import io
import logging
import math
import os
import queue
import signal
import sys
import threading
import time

from PIL import Image, ImageDraw, ImageFont
from picamera import PiCamera

from aiy.board import Board
from aiy.leds import Color, Leds, Pattern, PrivacyLed
from aiy.toneplayer import TonePlayer
from aiy.vision.inference import CameraInference
from aiy.vision.models import face_detection
from aiy.vision.streaming.server import StreamingServer
from aiy.vision.streaming import svg


logger = logging.getLogger(__name__)

JOY_COLOR = (255, 70, 0)
SAD_COLOR = (0, 0, 64)

JOY_SCORE_HIGH = 0.85
JOY_SCORE_LOW = 0.10

JOY_SOUND = ('C5q', 'E5q', 'C6q')
SAD_SOUND = ('C6q', 'E5q', 'C5q')
MODEL_LOAD_SOUND = ('C6w', 'c6w', 'C6w')
BEEP_SOUND = ('E6q', 'C6q')

FONT_FILE = '/usr/share/fonts/truetype/freefont/FreeSans.ttf'

BUZZER_GPIO = 22

@contextlib.contextmanager
def stopwatch(message):
    try:
```

OpenCV: est efficace sur les images et les videos

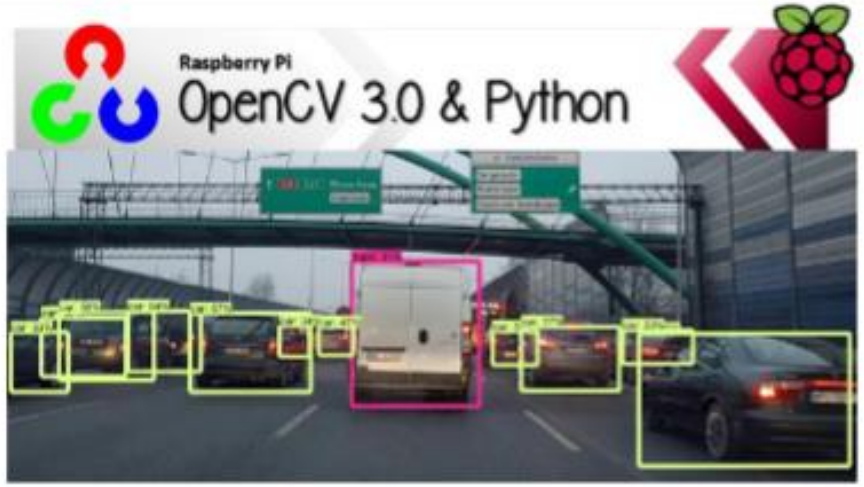
Installer un environnement virtuel pour Python pour installer OpenCV-Python

```
# Install virtualenv if it's not installed
pip install virtualenv

# Create a virtual environment
virtualenv opencv-env

# Activate the virtual environment
# On Windows
opencv-env\Scripts\activate
# On MacOS/Linux
source opencv-env/bin/activate
```

```
pip install opencv-python
```

Vérifier l'insatallation

```
import cv2

# Print the version of OpenCV installed
print(cv2.__version__)
```



Publié le 27 novembre 2018 - par Stéphane KUENTZ

I.A : Réalisez un système de reconnaissance d'objets avec Raspberry Pi

Comment reconnaître différents objets en utilisant un Raspberry Pi et **OpenCV** ? Ce tutoriel vous explique comment faire. Vous pourrez ensuite envoyer un e-mail avec la photo de l'objet en pièce jointe.

Ici, OpenCV utilisera un modèle de **réseaux neuronaux artificiels**, développé par Google : les **Mobilenet SSD**. Conçu pour l'embarqué, Il est particulièrement performant sur l' architecture ARM du Raspberry. 🙂

Here are the models that were available at the time of writing:

- Face Detection – identifies faces with bounding boxes and returns a happiness score for each
- Object Detection – This looks like a Vision Bonnet optimized variant to the Object Detection API I covered earlier but only trained setup to detect Cats, Dogs, and People with bounding boxes
- Image Classification – takes an image and assigns it to one of 1000 different image classes (no bounding boxes)
- Dish Classifier – takes an image and assigns it to one of 2023 prepared food classes

MODEL D'ANALYSE D'IMAGE AVEC LE MODULE AIY D'IA DE GOOGLE

object_detection.py

```
import argparse
from PIL import Image
from PIL import ImageDraw

from aiy.vision.inference import ImageInference
from aiy.vision.models import object_detection


def _crop_center(image):
  width, height = image.size
  size = min(width, height)
  x, y = (width - size) / 2, (height - size) / 2
  return image.crop((x, y, x + size, y + size)), (x, y)


def main():
  parser = argparse.ArgumentParser()
  parser.add_argument('--input', '-i', dest='input',
required=True)
  parser.add_argument('--output', '-o',
dest='output')
  args = parser.parse_args()

  with ImageInference(object_detection.model()) as
inference:
    image = Image.open(args.input)
    image_center, offset = _crop_center(image)
    draw = ImageDraw.Draw(image)
    result = inference.run(image_center)
    for i, obj in enumerate(object_detection.get_
objects(result, 0.3, offset)):
      print('Object #%d: %s' % (i, str(obj)))
      x, y, width, height = obj.bounding_box
      draw.rectangle((x, y, x + width, y + height),
outline='red')
    if args.output:
      image.save(args.output)

if __name__ == '__main__':
  main()
```

image_classification.py

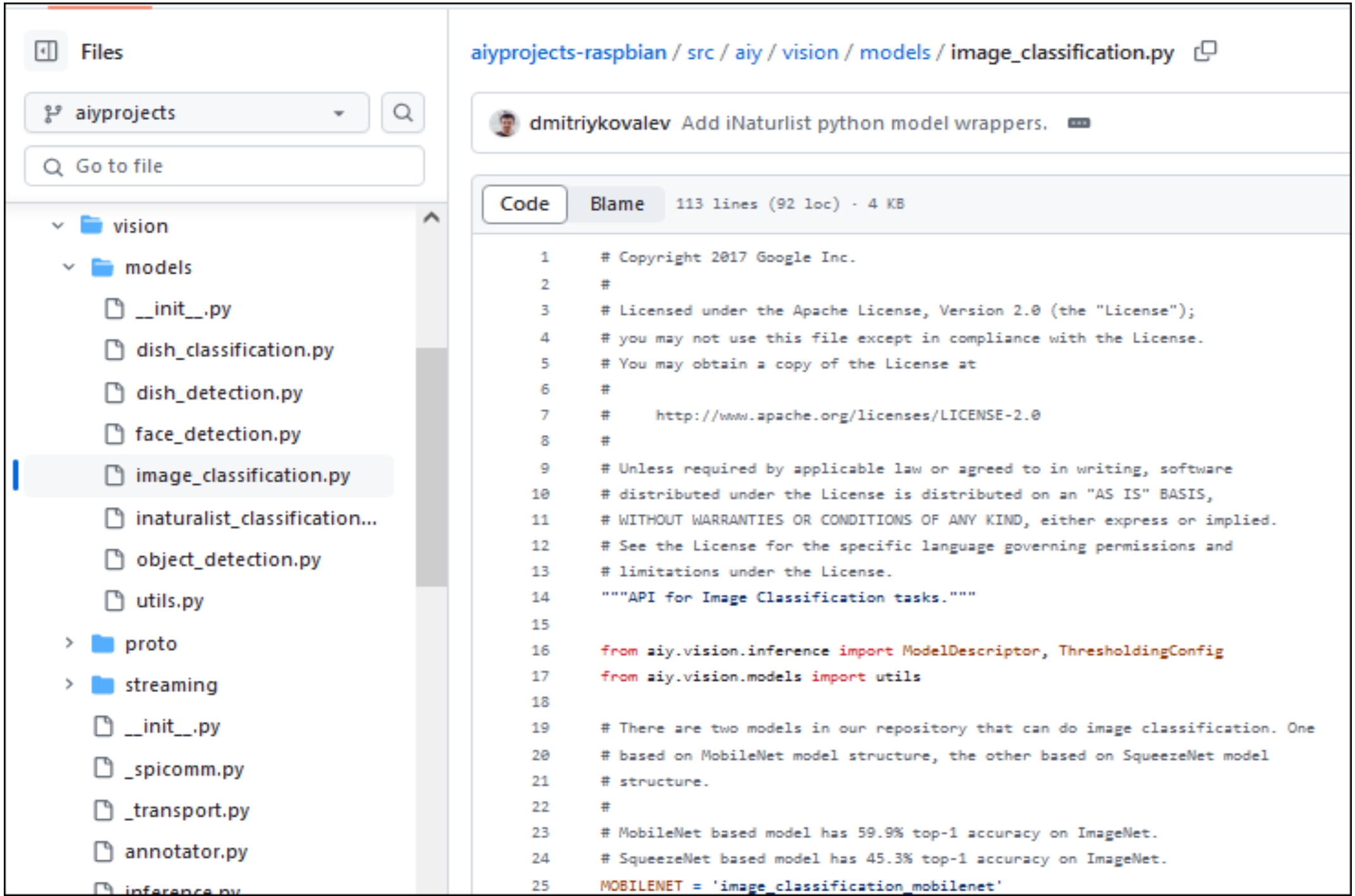
```
import argparse
from PIL import Image

from aiy.vision.inference import ImageInference
from aiy.vision.models import image_classification


def main():
  parser = argparse.ArgumentParser()
  parser.add_argument('--input', '-i', dest='input', required=True)
  args = parser.parse_args()

  with ImageInference(image_classification.model()) as inference:
    image = Image.open(args.input)
    classes = image_classification.get_classes(inference.run(image))
    for i, (label, score) in enumerate(classes):
      print('Result %d: %s (prob=%f)' % (i, label, score))

if __name__ == '__main__':
  main()
```

face_detection.py

```
from aiy.vision.inference import CameraInference
from aiy.vision.models import face_detection
from picamera import PiCamera


def main():
  with PiCamera() as camera:
    camera.resolution = (1640, 922)
    camera.start_preview()

    with CameraInference(face_detection.model()) as inference:
      for result in inference.run():
        if len(face_detection.get_faces(result)) >= 1:
          camera.capture('faces.jpg')
          break

    camera.stop_preview()


if __name__ == '__main__':
  main()
```

IMAGE CLASSIFIER

The Image Classifier demo is designed to identify 1,000 different types of objects. This demo can use either the SqueezeNet model or Google's MobileNet model architecture.


Files
aiyprojects
Go to file
vision
models
__init__.py
dish_classification.py
dish_detection.py
face_detection.py
image_classification.py
inaturalist_classification...
object_detection.py
utils.py
proto
streaming
__init__.py
_spicomm.py
_transport.py
annotator.py
aiyprojects-raspbian / src / aiy / vision / models / image_classification.py
dmitriykovalev Add iNaturlist python model wrappers.
Code
Blame
113 lines (92 loc) · 4 KB
Copyright 2017 Google Inc.
#
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
#
http://www.apache.org/licenses/LICENSE-2.0
#
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
"""API for Image Classification tasks."""
from aiy.vision.inference import ModelDescriptor, ThresholdingConfig
from aiy.vision.models import utils
# There are two models in our repository that can do image classification. One
# based on MobileNet model structure, the other based on SqueezeNet model
# structure.
#
# MobileNet based model has 59.9% top-1 accuracy on ImageNet.
# SqueezeNet based model has 45.3% top-1 accuracy on ImageNet.
MOBILENET = 'image_classification_mobilenet'

This repository has been archived by the owner on Feb 9, 2023. It is now read-only.

google / aiyprojects-raspbian Public archive

Notifications | Fork 691 | Star 1.6k

Code | Issues 142 | Pull requests 17 | Actions | Projects | Security | Insights

Files

aiyprojects

Go to file

- checkpoints
- docs
- drivers
- schematics
- scripts
- src
  - aiy
    - assistant
    - vision
      - models
        - __init__.py
        - dish_classification.py
        - dish_detection.py
        - face_detection.py

aiyprojects-raspbian / src / aiy / vision / models /

dmitriykovalev Add missing copyrights — 0da9963 · 5 years ago | History

Name	Last commit message	Last commit date
..		
__init__.py	Python API and example source code for VisionKit	7 years ago
dish_classification.py	Add more unit tests and verify all of them pass.	6 years ago
dish_detection.py	Add iNaturlist python model wrappers.	6 years ago
face_detection.py	Fix lint errors and warnings.	6 years ago
image_classification.py	Add iNaturlist python model wrappers.	6 years ago
inaturalist_classification.py	Add missing copyrights	5 years ago
object_detection.py	Fix lint errors.	6 years ago
utils.py	Add iNaturlist python model wrappers.	6 years ago

aiyprojects-raspbian / src / aiy / vision / models / image_classification.py

Code Blame 113 lines (92 loc) · 4 KB

```
22  #
23  # MobileNet based model has 59.9% top-1 accuracy on ImageNet.
24  # SqueezeNet based model has 45.3% top-1 accuracy on ImageNet.
25  MOBILENET = 'image_classification_mobilenet'
26  SQUEEZENET = 'image_classification_squeezenet'
27
28  _COMPUTE_GRAPH_NAME_MAP = {
29      MOBILENET: 'mobilenet_v1_160res_0.5_imagenet.binaryproto',
30      SQUEEZENET: 'squeezenet_160res_5x5_0.75.binaryproto',
31  }
32
33  _OUTPUT_TENSOR_NAME_MAP = {
34      MOBILENET: 'MobilenetV1/Predictions/Softmax',
35      SQUEEZENET: 'Prediction',
36  }
37
38  _CLASSES = utils.load_labels('mobilenet_v1_160res_0.5_imagenet_labels.txt')
39
40  def sparse_configs(top_k=len(_CLASSES), threshold=0.0, model_type=MOBILENET):
41      name = _OUTPUT_TENSOR_NAME_MAP[model_type]
42      return {
43          name: ThresholdingConfig(logical_shape=[len(_CLASSES)],
44                                   threshold=threshold,
45                                   top_k=top_k,
46                                   to_ignore=[])
47      }
48
49  def model(model_type=MOBILENET):
50      return ModelDescriptor(
51          name=model_type,
52          input_shape=(1, 160, 160, 3),
53          input_normalizer=(128.0, 128.0),
```

aiyprojects-raspbian / src / aiy / vision / models / image_classification.py

Code Blame 113 lines (92 loc) · 4 KB

```
49  def model(model_type=MOBILENET):
50      return ModelDescriptor(
51          name=model_type,
52          input_shape=(1, 160, 160, 3),
53          input_normalizer=(128.0, 128.0),
54          compute_graph=utils.load_compute_graph(_COMPUTE_GRAPH_NAME_MAP[model_type]))
55
56
57  def _get_probs(result):
58      assert len(result.tensors) == 1
59      tensor = result.tensors[_OUTPUT_TENSOR_NAME_MAP[result.model_name]]
60      assert utils.shape_tuple(tensor.shape) == (1, 1, 1, len(_CLASSES))
61      return tuple(tensor.data)
62
63
64  def get_classes(result, top_k=None, threshold=0.0):
65      """Converts image classification model output to list of detected objects.
66
67      Args:
68        result: output tensor from image classification model.
69        top_k: int; max number of objects to return.
70        threshold: float; min probability of each returned object.
71
72      Returns:
73        A list of (class_name: string, probability: float) pairs ordered by
74        probability from highest to lowest. The number of pairs is not greater than
75        top_k. Each probability is greater than threshold. For
76        example:
77
78        [('Egyptian cat', 0.767578)
79         ('tiger cat, 0.163574)
```

aiyprojects-raspbian / src / aiy / vision / models / image_classification.py

Code | Blame | 113 lines (92 loc) · 4 KB

```
64  def get_classes(result, top_k=None, threshold=0.0):
78          [('Egyptian cat', 0.767578)
79           ('tiger cat, 0.163574)
80           ('lynx/catamount', 0.039795)]
81        """
82        probs = _get_probs(result)
83        pairs = [pair for pair in enumerate(probs) if pair[1] > threshold]
84        pairs = sorted(pairs, key=lambda pair: pair[1], reverse=True)
85        pairs = pairs[0:top_k]
86        return [('/'.join(_CLASSES[index]), prob) for index, prob in pairs]
87
88
89  def _get_pairs(result):
90        assert len(result.tensors) == 1
91        tensor = result.tensors[_OUTPUT_TENSOR_NAME_MAP[result.model_name]]
92        indices = tuple(tensor.indices)
93        data = tuple(tensor.data)
94        return [(index.values[0], prob) for index, prob in zip(indices, data)]
95
96
97  def get_classes_sparse(result):
98        """Converts sparse image classification model output to list of detected objects.
99
100       Args:
101         result: sparse output tensor from image classification model.
102
103       Returns:
104         A list of (class_name: string, probability: float) pairs ordered by
105         probability from highest to lowest.
106         For example:
107
```

```
87
88
89  def _get_pairs(result):
90        assert len(result.tensors) == 1
91        tensor = result.tensors[_OUTPUT_TENSOR_NAME_MAP[result.model_name]]
92        indices = tuple(tensor.indices)
93        data = tuple(tensor.data)
94        return [(index.values[0], prob) for index, prob in zip(indices, data)]
95
96
97  def get_classes_sparse(result):
98        """Converts sparse image classification model output to list of detected objects.
99
100       Args:
101         result: sparse output tensor from image classification model.
102
103       Returns:
104         A list of (class_name: string, probability: float) pairs ordered by
105         probability from highest to lowest.
106         For example:
107
108         [('Egyptian cat', 0.767578)
109          ('tiger cat, 0.163574)
110       """
111       pairs = _get_pairs(result)
112       pairs = sorted(pairs, key=lambda pair: pair[1], reverse=True)
113       return [('/'.join(_CLASSES[index]), prob) for index, prob in pairs]
```

DÉTECTEUR DE VISAGES JOYEUX

The Face Detector model locates and identifies faces from an image. It also provides a "joy score" for each face.

DISH CLASSIFIER

The Dish Classifier model is designed to identify food in an image. It's based on the MobileNet model architecture and trained to recognize over 2,000 types of food.

DOG / CAT / HUMAN DETECTOR

The Dog / Cat / Human Detector can identify whether there's a dog, cat, or person in an image and draw a box around the identified objects. It's based on the MobileNet model architecture.

FACE DETECTOR

The Face Detector model locates and identifies fa image. It also provides a "joy score" for each face.

LEARN MORE

NATURE EXPLORER

Nature explorer has 3 machine learning models based on MobileNet, trained on photos contributed by the iNaturalist community. These models are built to recognize 4,080 different species (~960 birds, ~1020 insects, ~2100 plants).

IMAGE CLASSIFIER

The Image Classifier demo is designed to identify 1,000 different types of objects. This demo can use either the SqueezeNet model or Google's MobileNet model architecture.

LEARN MORE