Politecnico di Bari

DIPARTIMENTO INTERATENEO DI FISICA

Dottorato in Fisica – XXXIX ciclo - 2024

Machine Learning techniques for particle physics

Federica Maria Simone - federica.simone@poliba.it

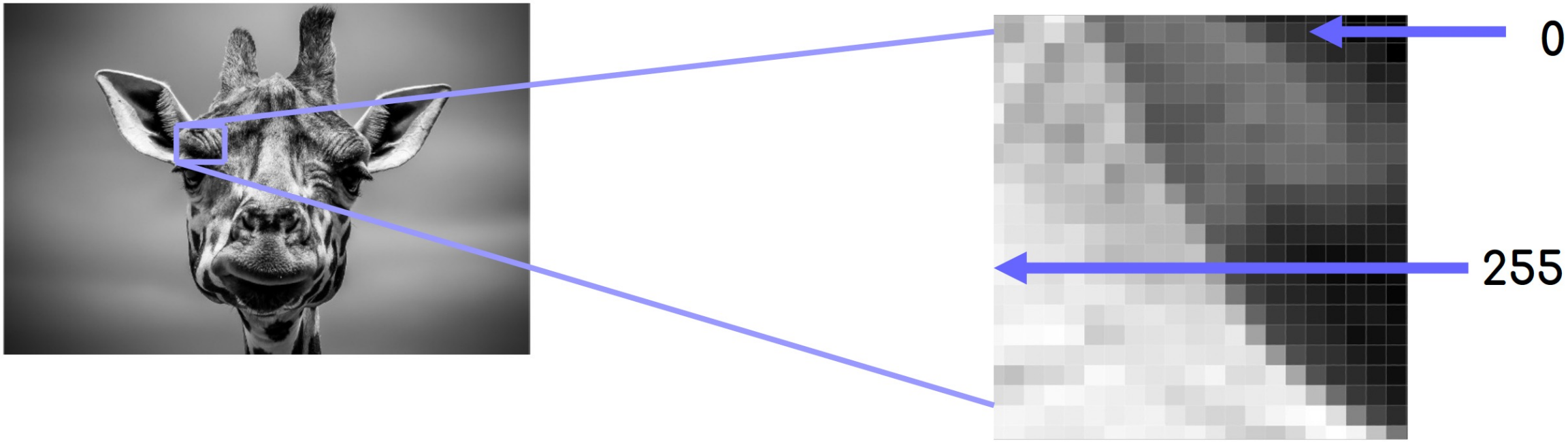
Convolutional Neural Networks (CNN)

Disclaimer: next slides are taken from many sources, please find useful links and credits on my last slide!

Introduction: images

Greyscale image:

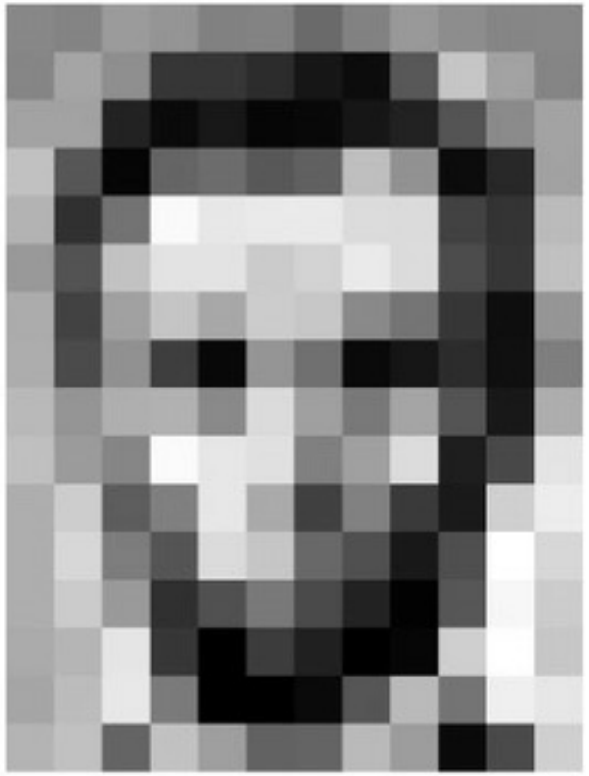
- Grayscale image is a matrix of pixels [H x W]
- Pixels = picture elements
- Each pixel stores number [0,255] for brightness

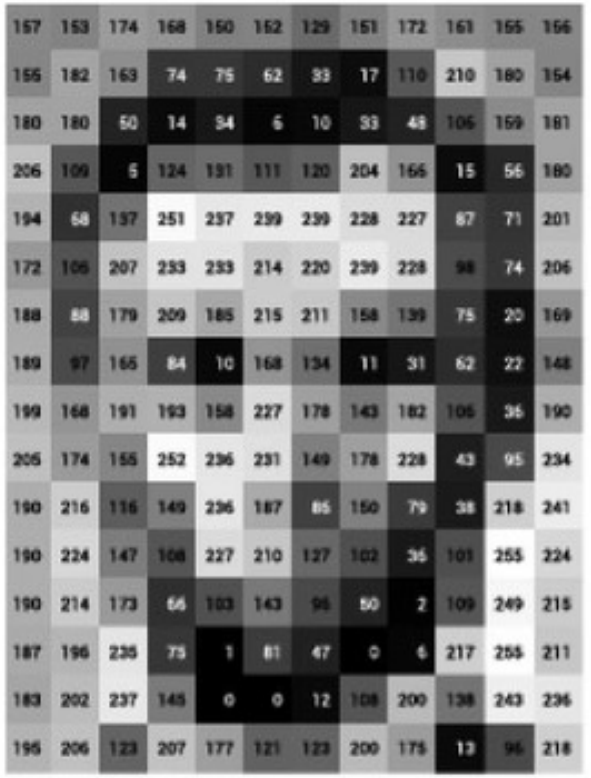


Introduction: images

Greyscale image:

- Grayscale image is a matrix of pixels [H x W]
- Pixels = picture elements
- Each pixel stores number [0,255] for brightness





157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

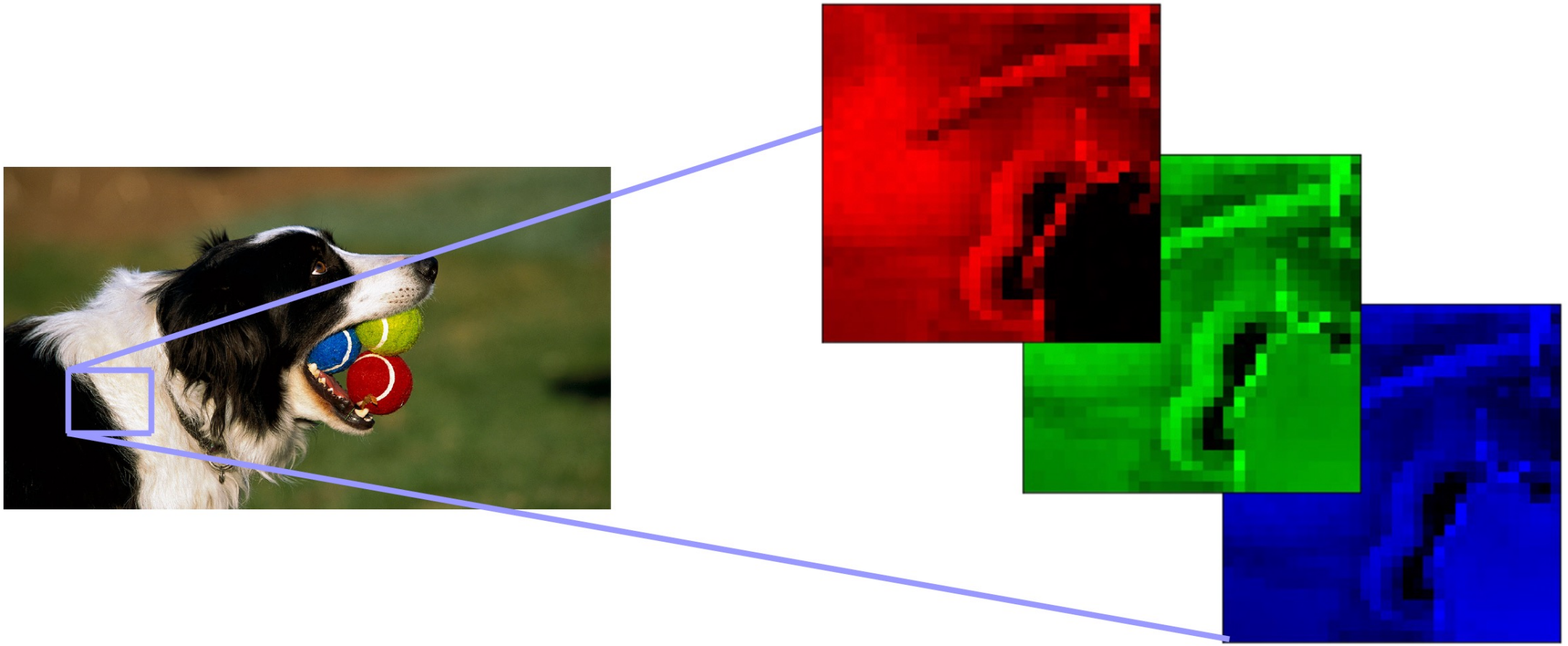
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

10.1093/llc/fqy085

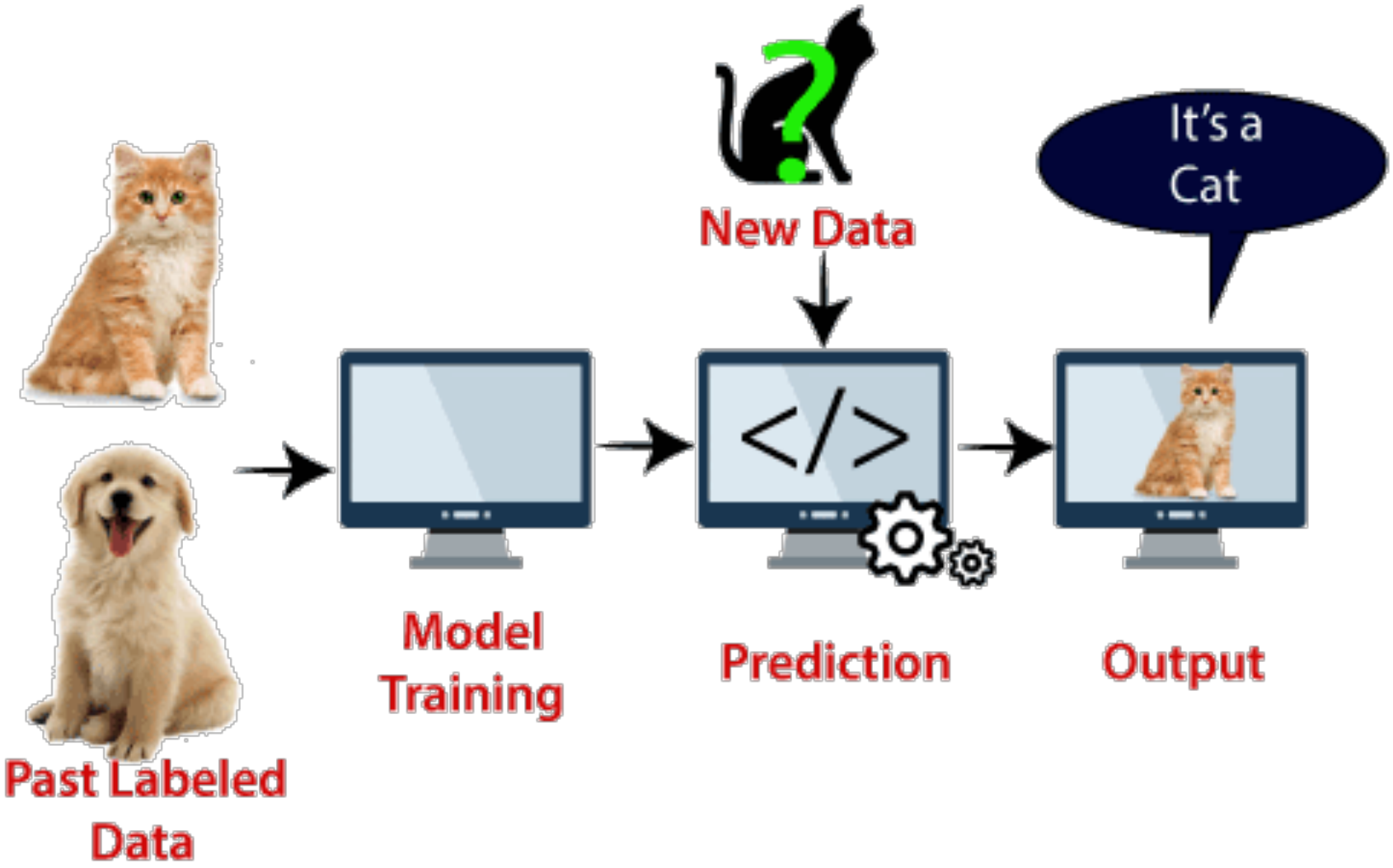
Introduction: images

RGB image:

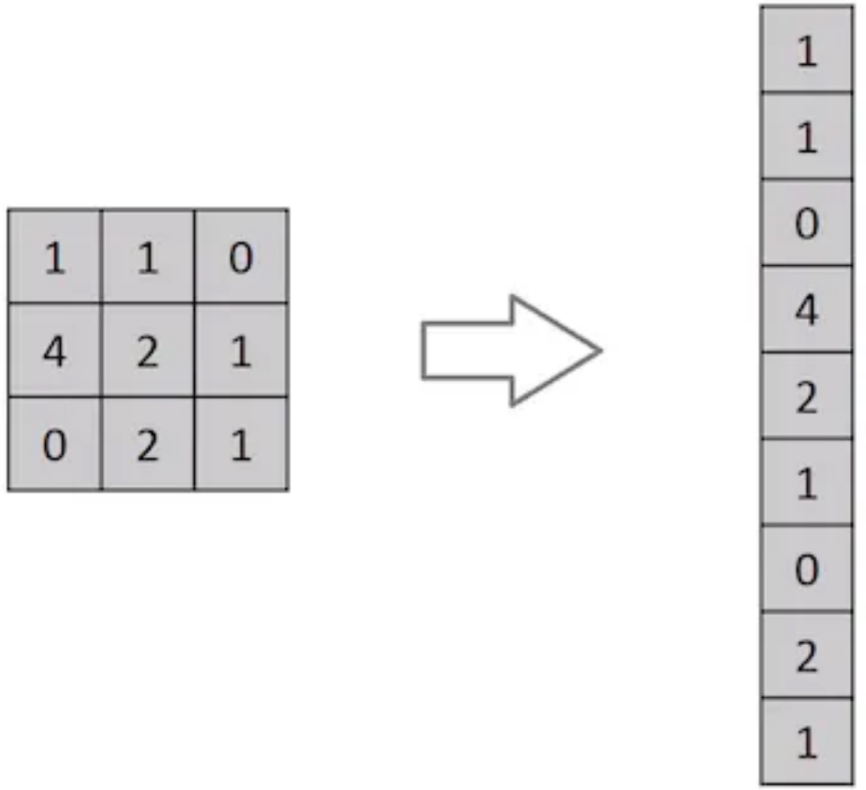
- RGB image is a 3d array [HxWx3] or [3xHxW]
- Each pixel stores Red, Green & Blue color values [0,255]



Problem: image classification



Problem: image classification can we solve it with feed forward NN?



Flattening of a 3×3 image matrix into a 9×1 vector

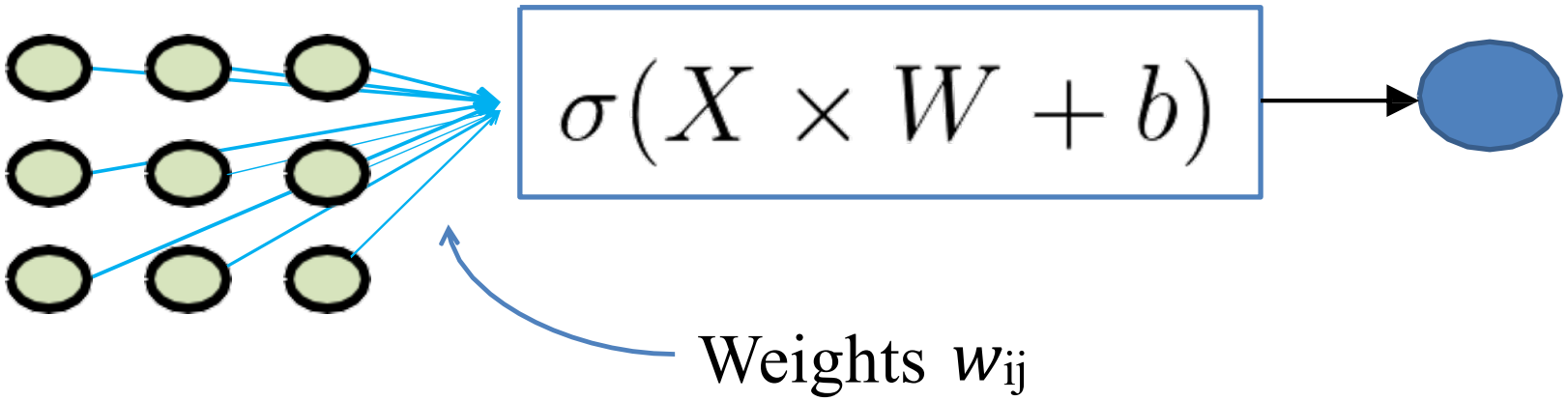
Input data: [H x W] matrix of numbers

Why not just flatten the image (from 3X3 into 9x1 vector) and feed it to a NN for classification?

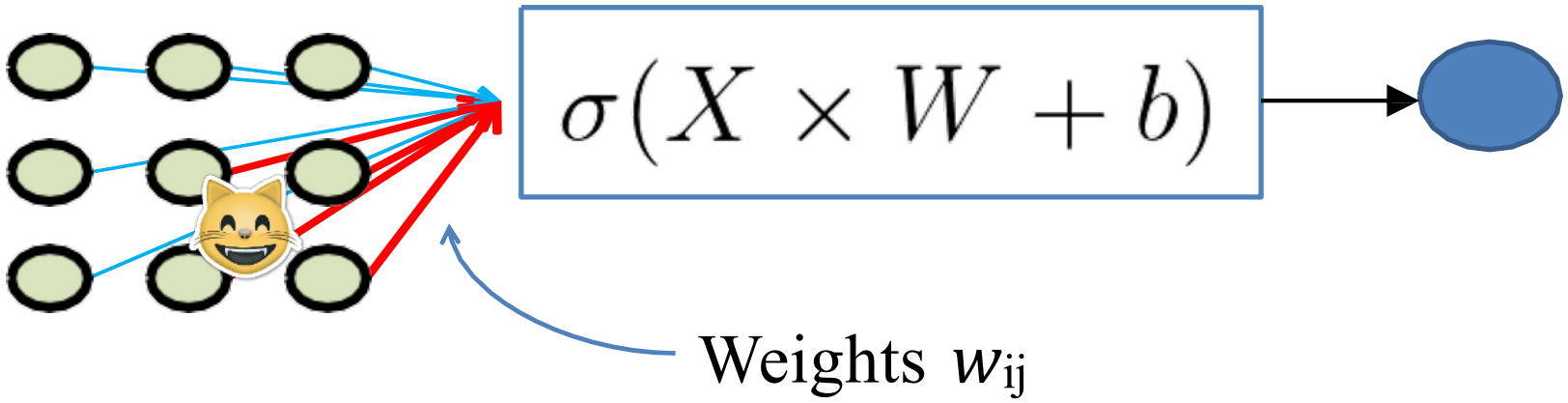
- Feed forward NN doesn't capture the positional dependencies between different pixels (see next slides)
- High number of weights, poor classification performance
  - For instance a linear layer taking a 256 × 256 RGB image as input, and producing an image of same size would require **$(256\times256\times3)^2 \simeq 3.87e{+}10$** parameters, with the corresponding memory footprint (≃150Gb !), and excess of capacity.

https://towardsdatascience.com

Problem: image classification
can we solve it with feed forward NN?

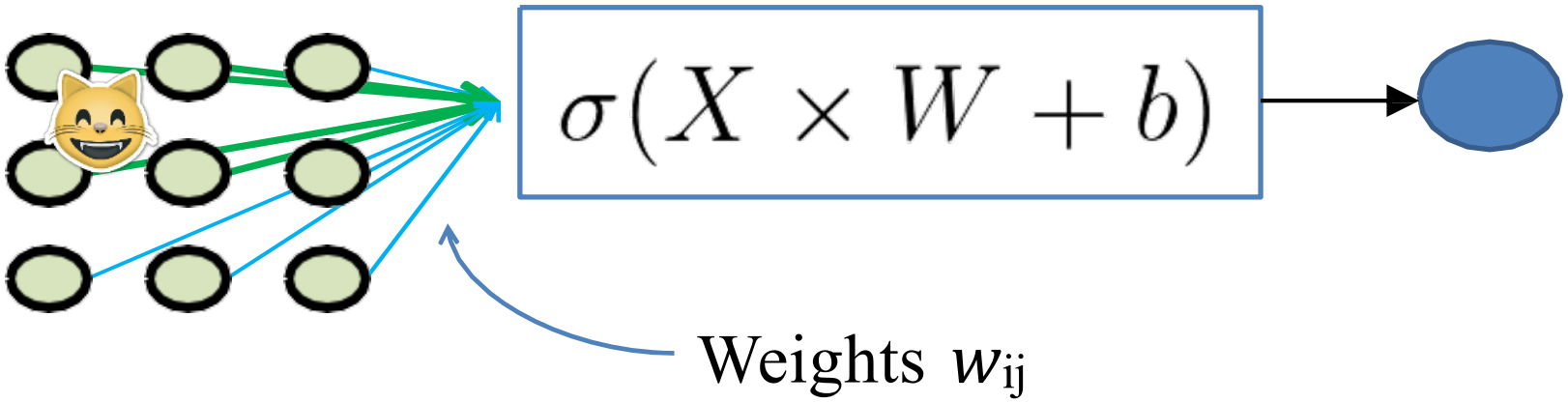


Problem: image classification
can we solve it with feed forward NN?



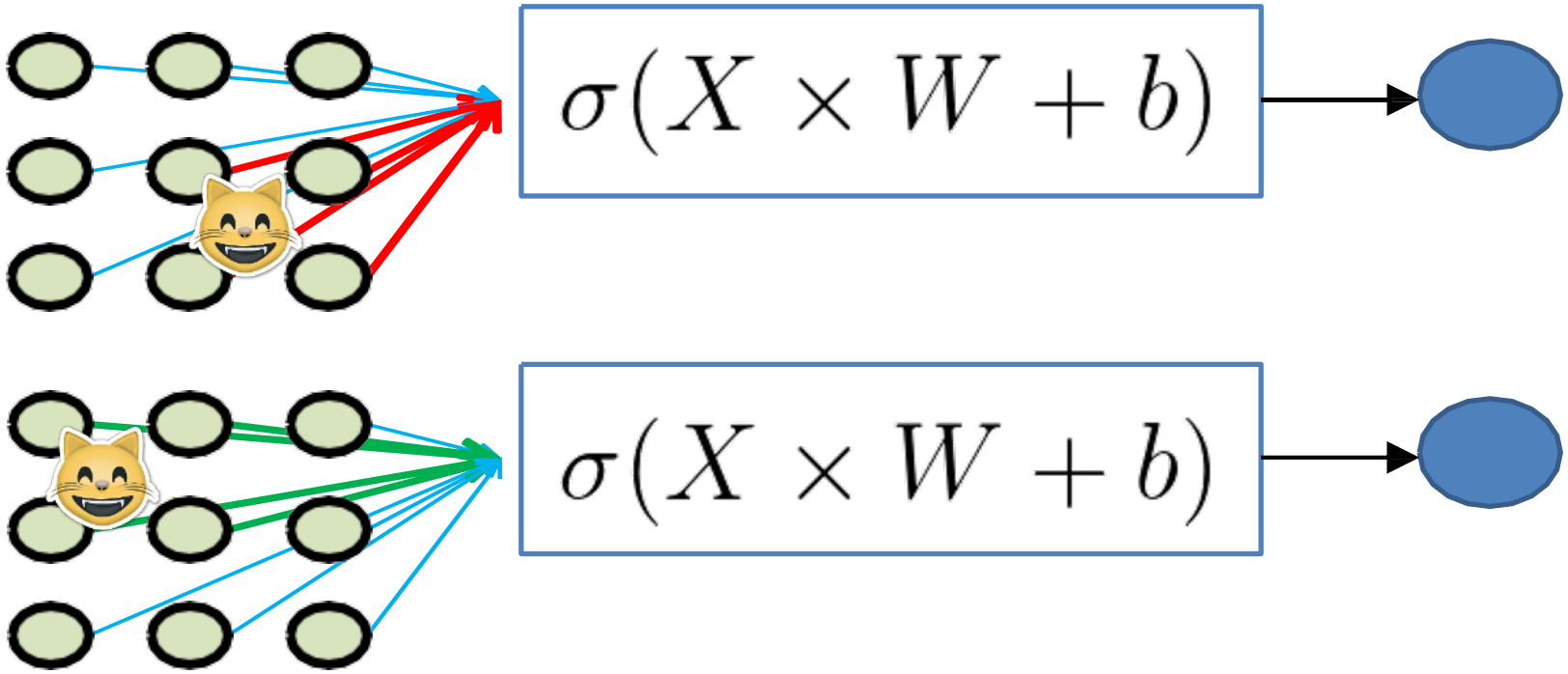
On this object, you will train
red
weights to react on cat face

Problem: image classification
can we solve it with feed forward NN?



On this object, you will train
green
weights to react on cat face

Problem: image classification
can we solve it with feed forward NN?



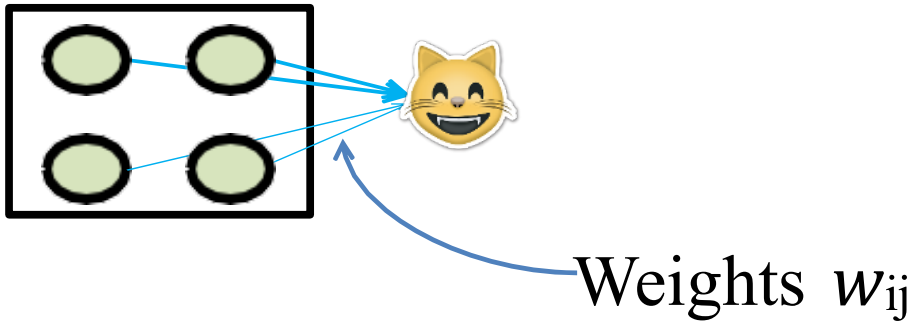
You network will have to learn those two cases separately!

Worst case: one neuron per position.

Solution

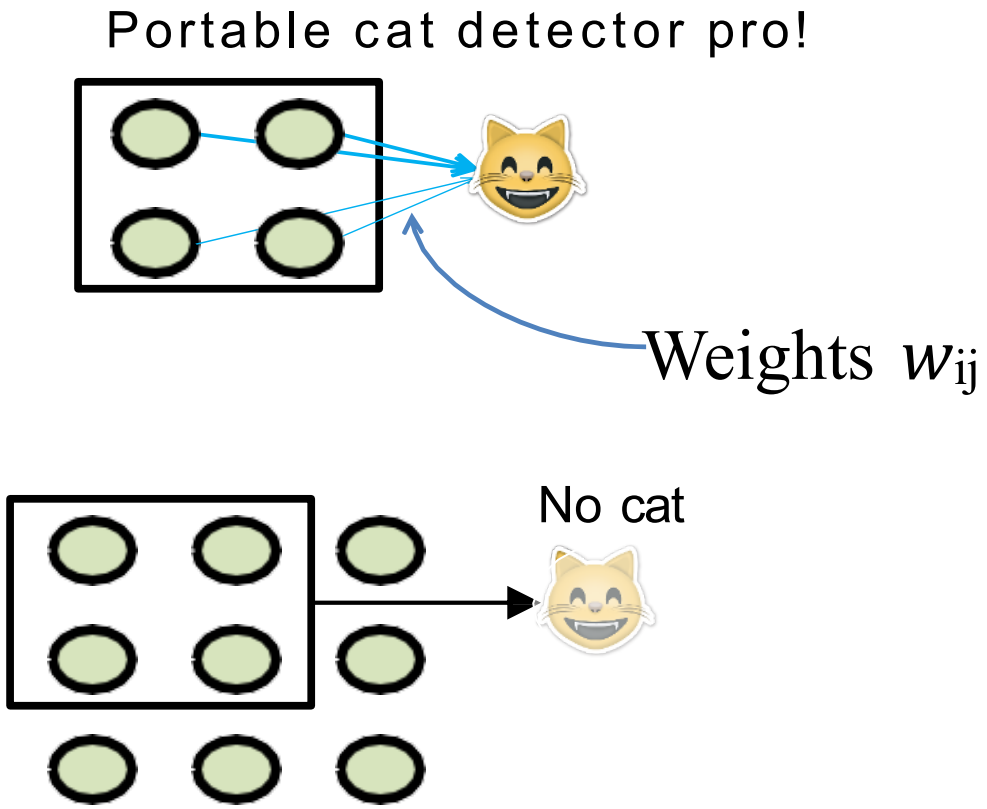
Idea: force all these “cat face” features to use **exactly the same weights**, shifting weight matrix each time.

Portable cat detector pro!



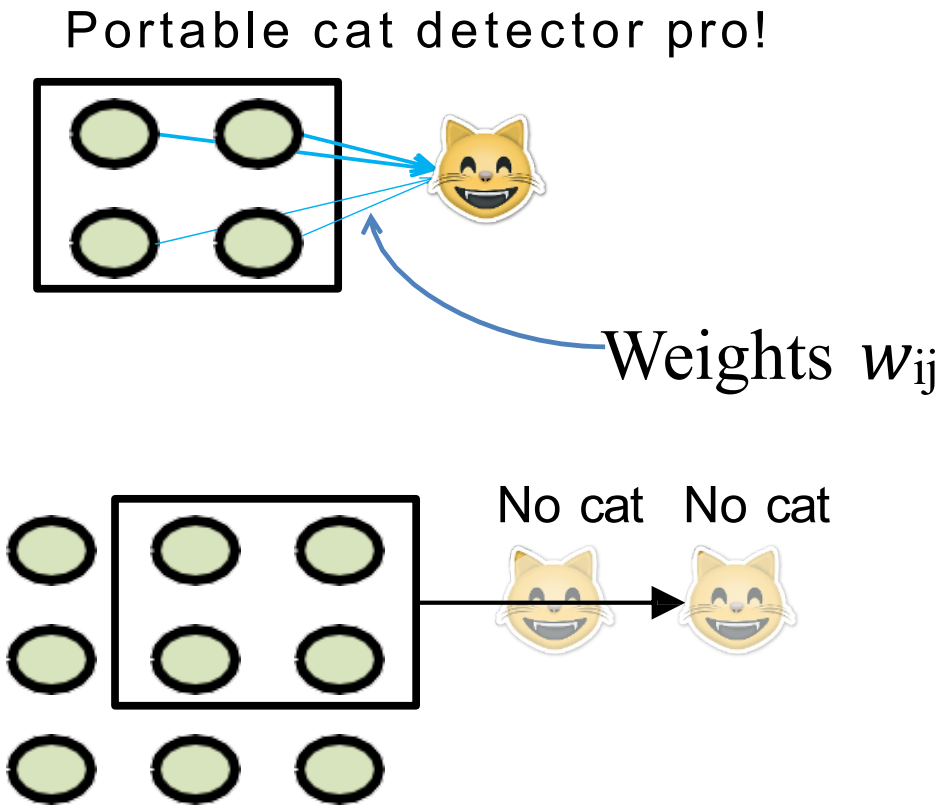
Solution

Idea: force all these “cat face” features to use **exactly the same weights**, shifting weight matrix each time.



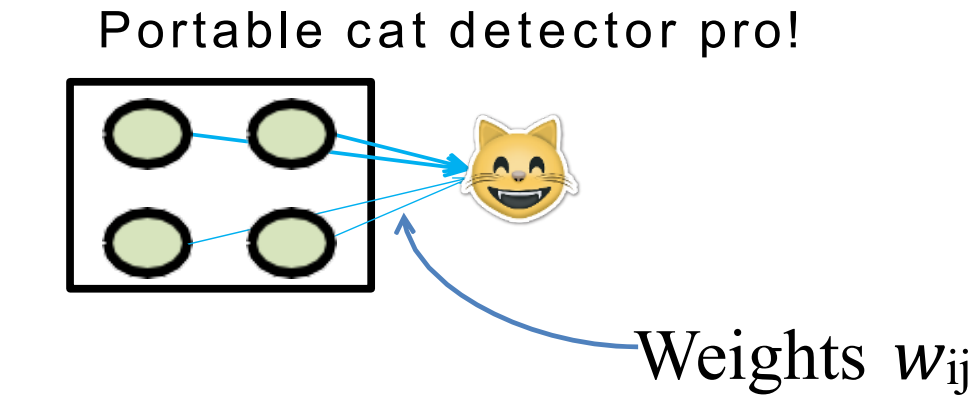
Solution

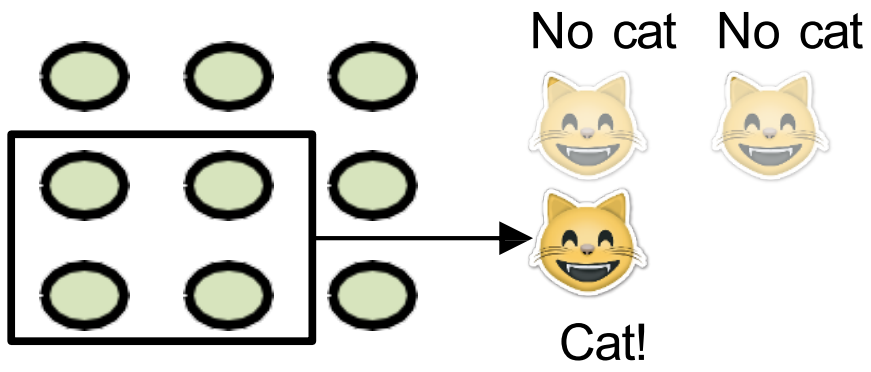
Idea: force all these “cat face” features to use **exactly the same weights**, shifting weight matrix each time.



Solution

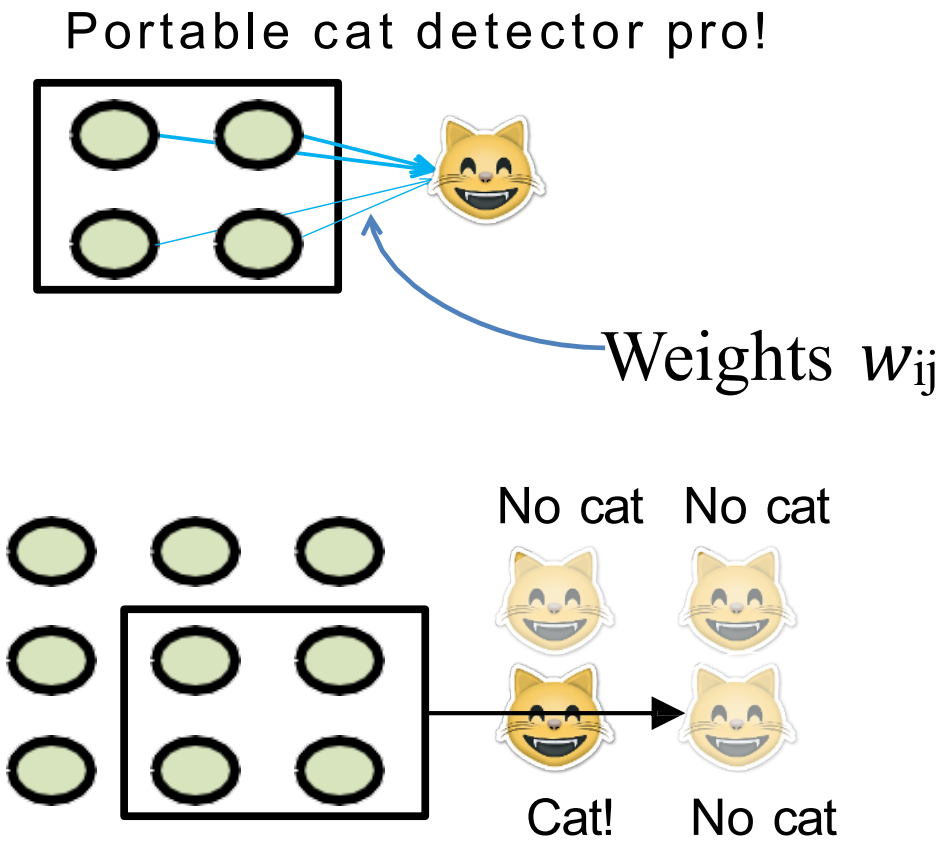
Idea: force all these "cat face" features to use **exactly the same weights**, shifting weight matrix each time.





Solution

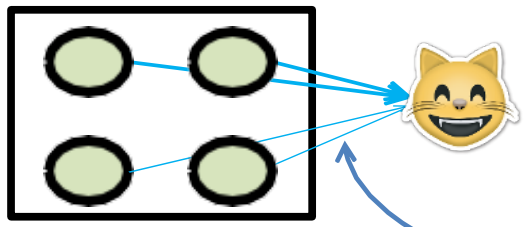
Idea: force all these "cat face" features to use **exactly the same weights**, shifting weight matrix each time.



Solution

Idea: force all these “cat face” features to use **exactly the same weights**, shifting weight matrix each time.

Portable cat detector pro!



Weights w_{ij}

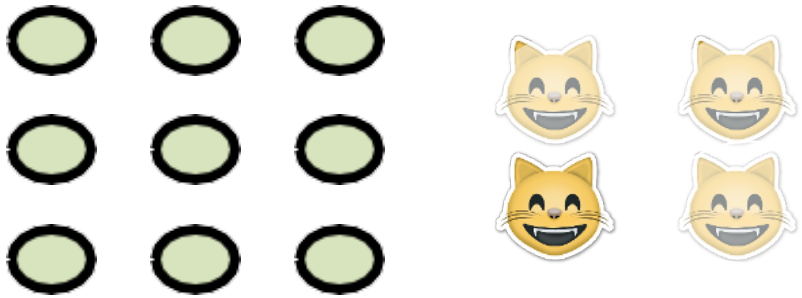
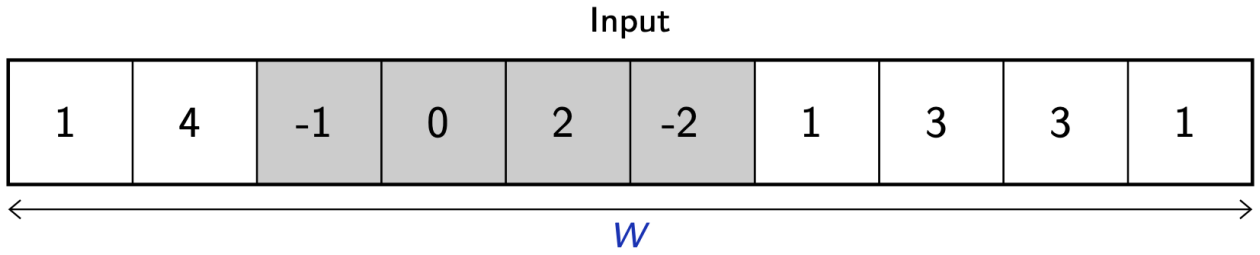


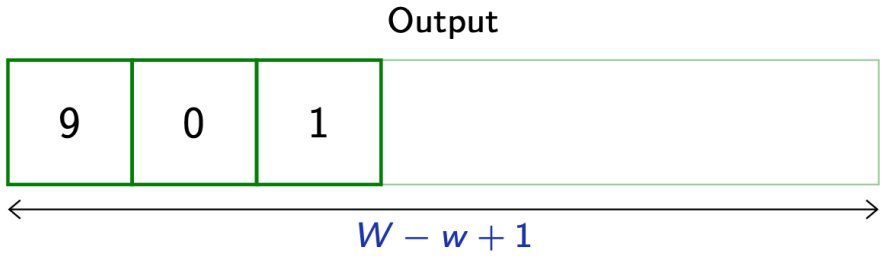
Image classified as «cat»

Convolution: 1D example



$$x = (x_1, \ldots, x_W) \quad \textbf{input}$$

$$u = (u_1, \ldots, u_w) \quad \textbf{kernel/filter}$$



$$(x \circledast u)_i = \sum_{j=1}^{w} x_{i-1+j}\, u_j \quad \textbf{output}$$

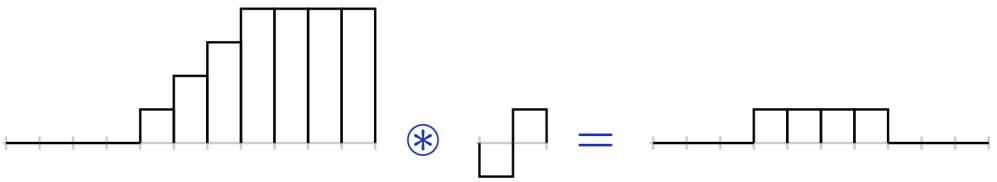
$$= (x_i, \ldots, x_{i+w-1}) \cdot u$$

For instance: $(1, 2, 3, 4) \circledast (3, 2) = (3 + 4, 6 + 6, 9 + 8) = (7, 12, 17)$

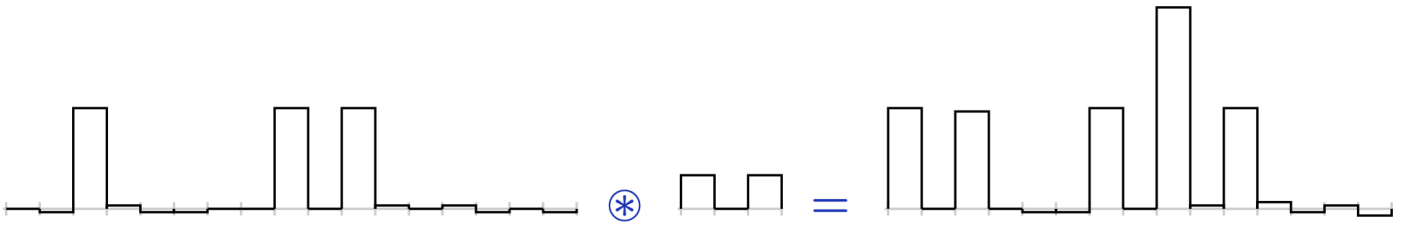
Convolution: 1D example

Convolution can implement in particular differential operators, e.g.

$$(0,0,0,0,1,2,3,4,4,4,4) \circledast (-1,1) = (0,0,0,1,1,1,1,0,0,0).$$



or crude "template matcher", e.g.



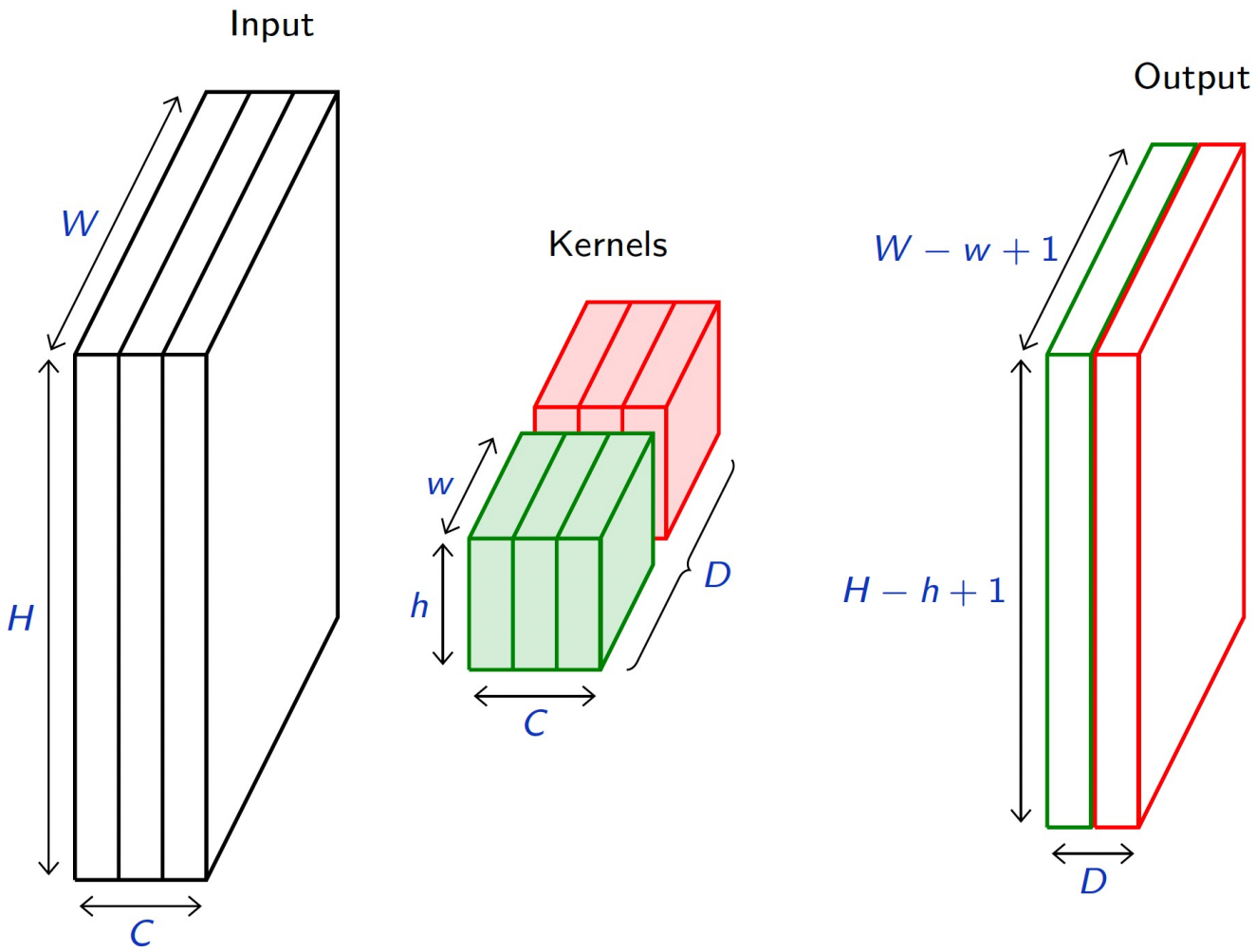
Francois Fleuret Deep learning / 4.4. Convolutions

The first example shows that a convolution can computes the **discrete first order derivative** of the input signal.
The second that it can **"detect" a structure** in a crude sense: here the kernel has the shape of two peaks and the response is maximal when the dot product is done with a similar structure in the input.

By having a way of computing derivatives and matching pattern, we can envision the following applications:

- detecting corners, edges
- detecting amplitude modifications

Convolution: extension to 2D images



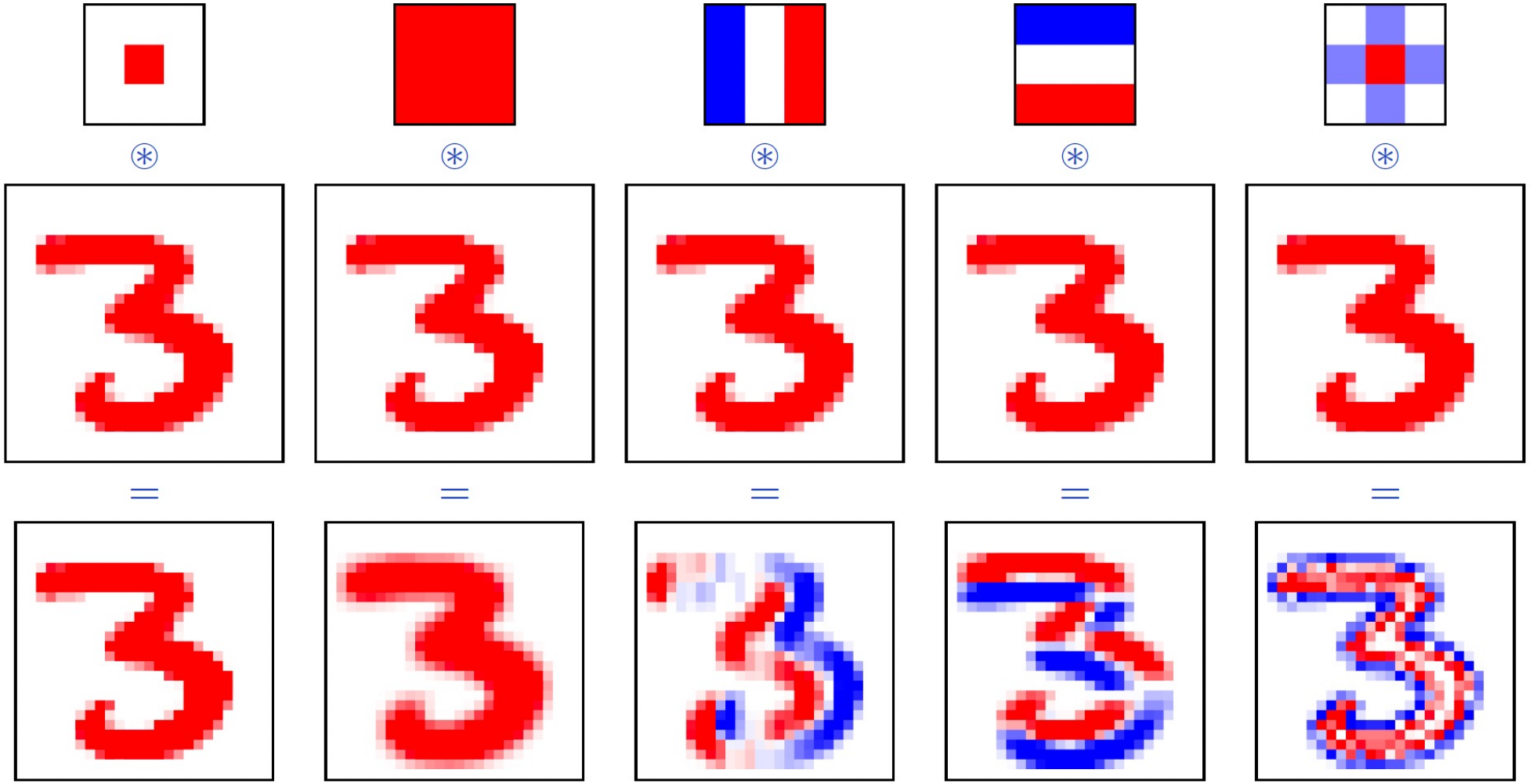
- **Input:** tensor of size C X H X W, with here with C = 3 (RGB)
- **Convolution layer:** two filters/kernels (D = 2) of size C x h x w
- **Output:** tensor of size D X (H – h + 1) X (W – w + 1)

Notes:

- if your input image has 3 channels, the kernels have 3 channels
- The convolution preserves the input structure (2D signal is converted into a 2D), but not its size
- A 3D convolution can be used if the channel index has some metric meaning (example: time for a series of grayscale video frames)

tf.keras.layers.Conv2D

Convolution: extension to 2D images

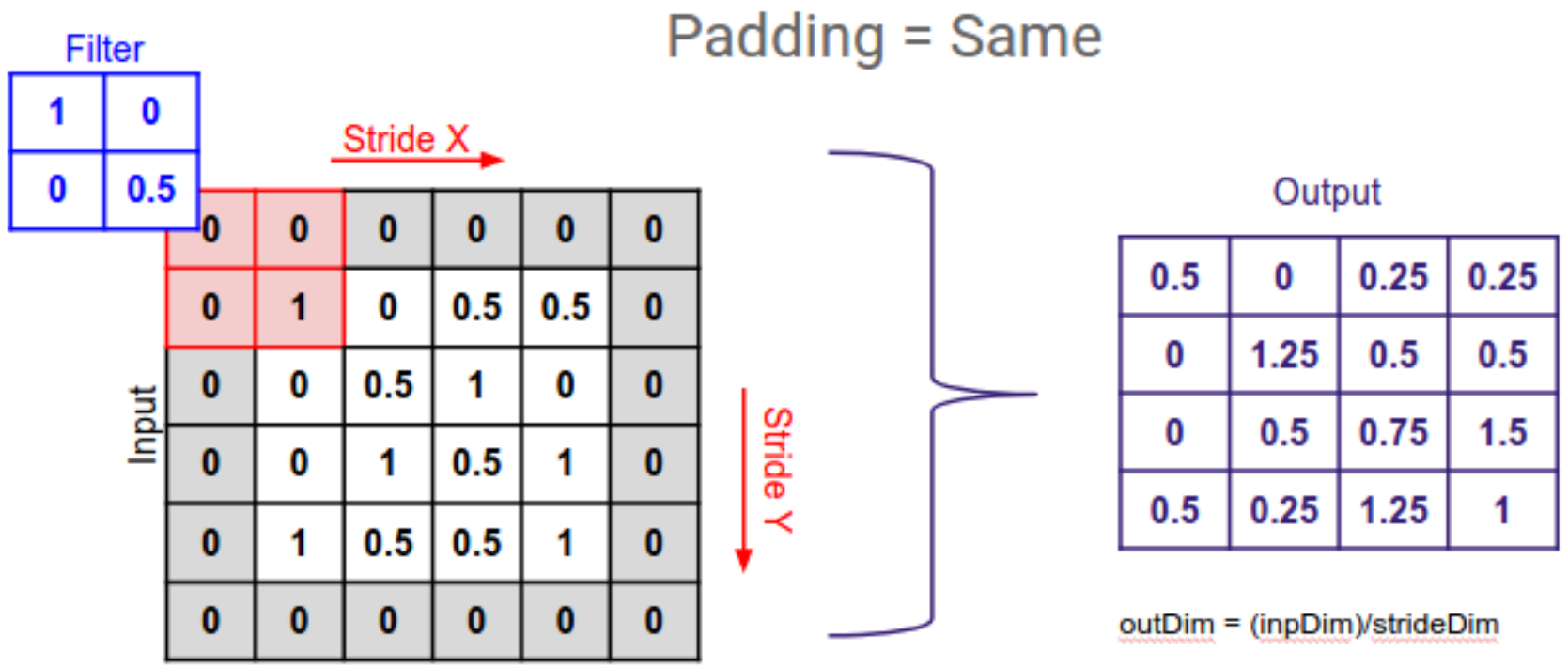


Padding

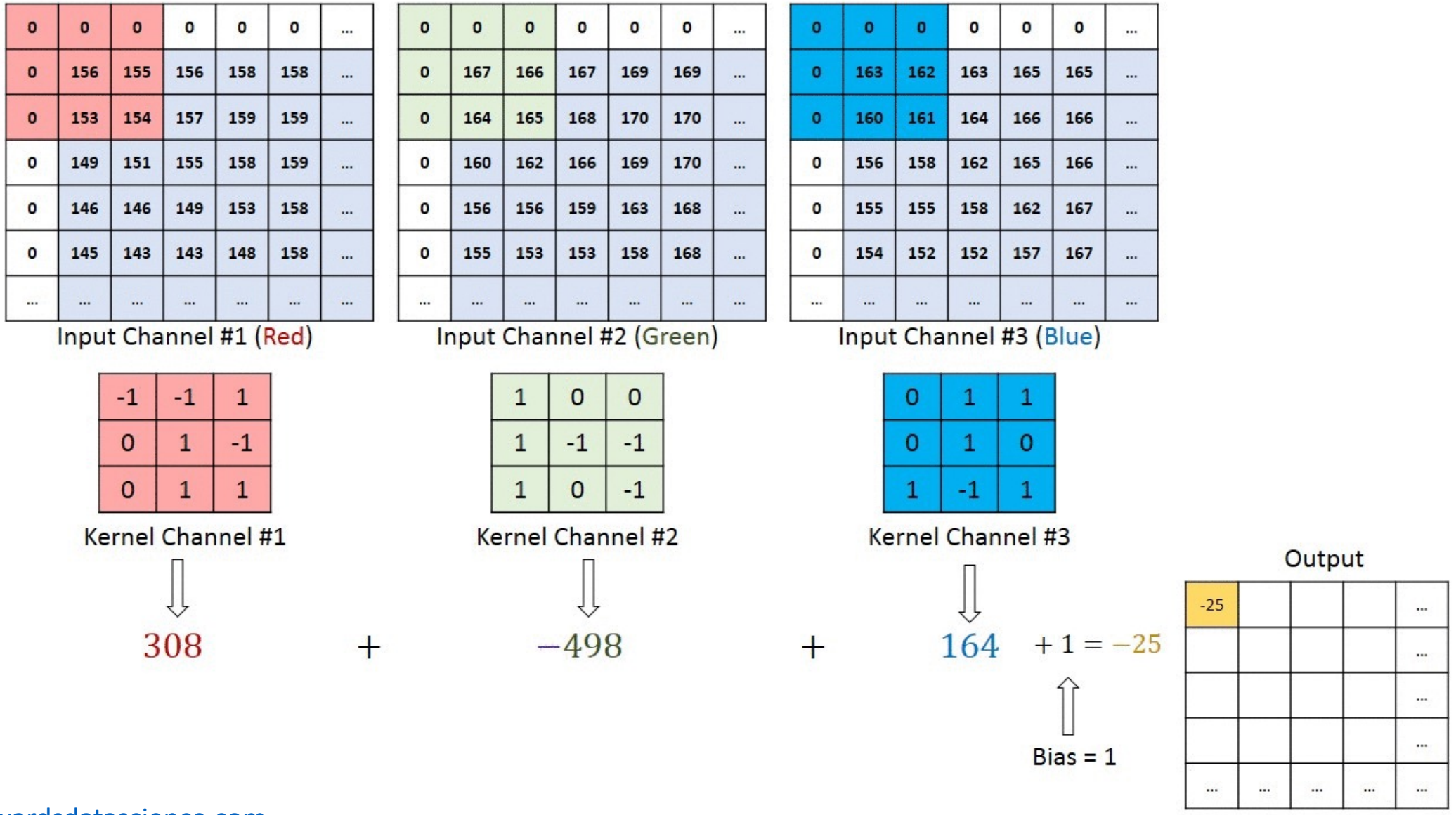
When applying convolutional layers, pixels on the image perimeter are hardly used. Moreover, the output has a different size than the input tensor.

Solution: add extra pixels all around the input image, typically set to zero

- Valid padding
- Same padding

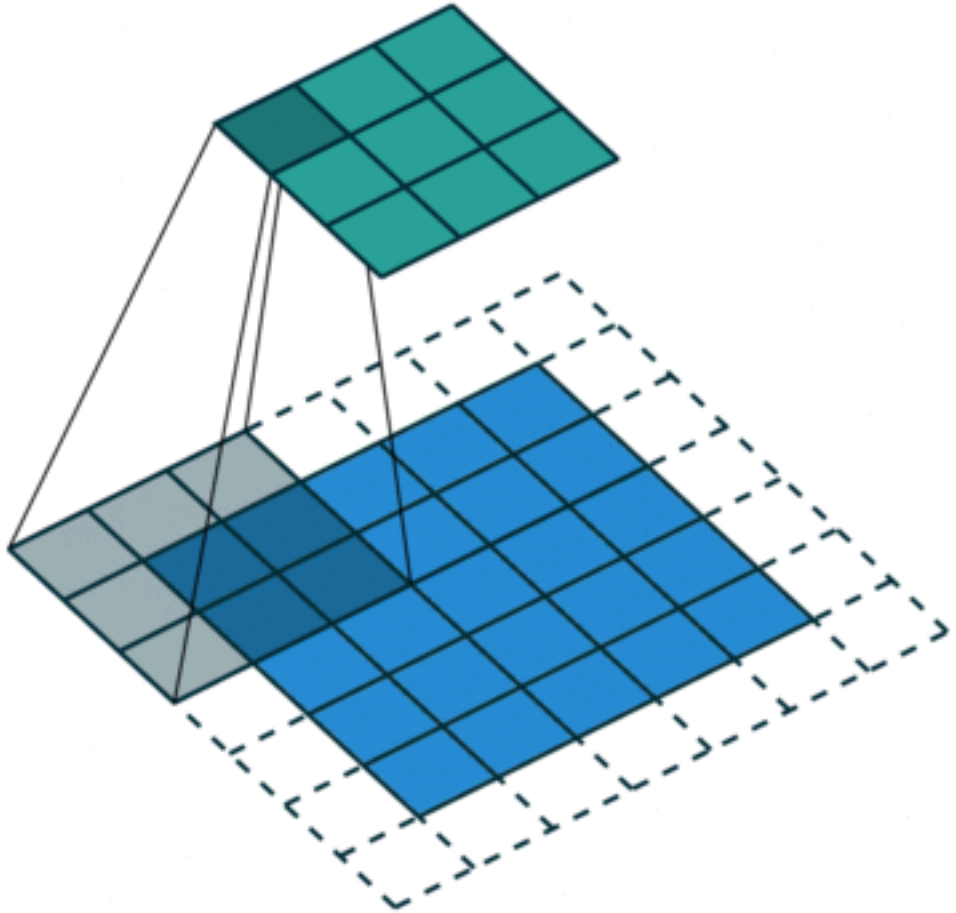


Convolution operation with stride length = 1



https://towardsdatascience.com

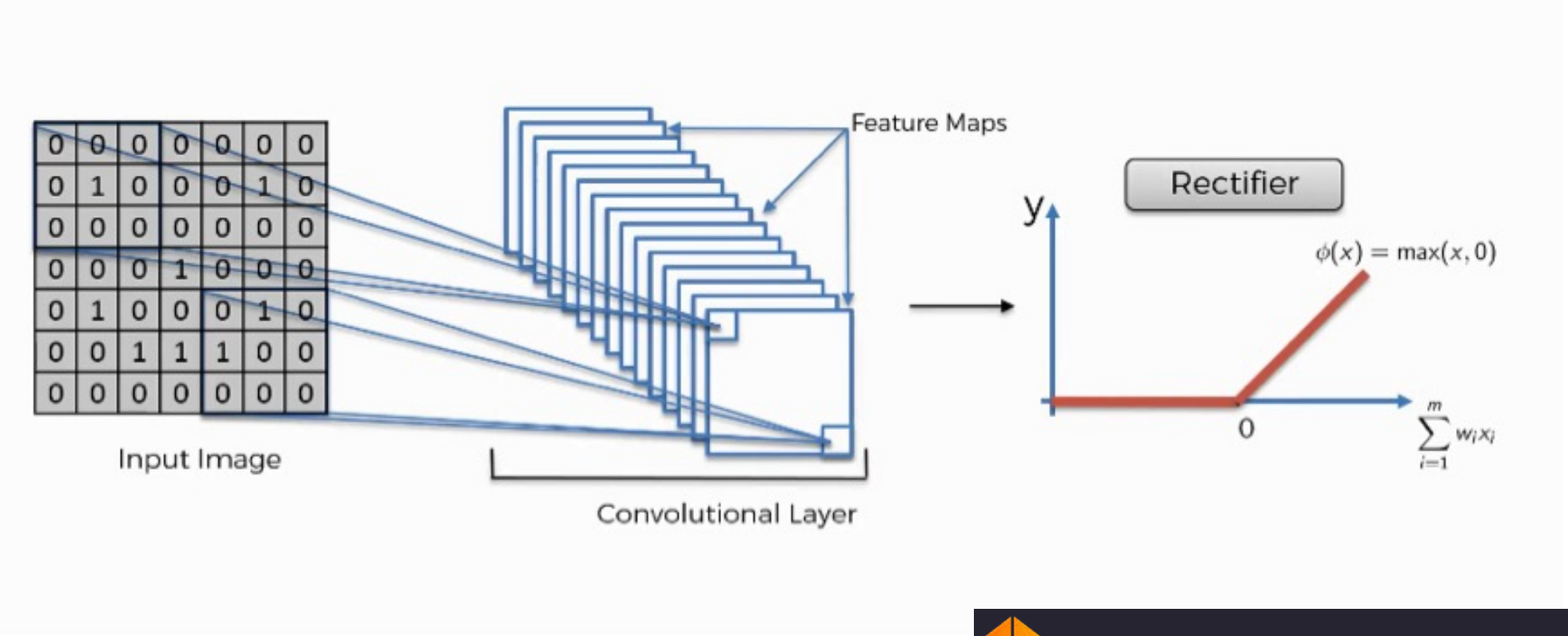
Convolution operation with stride length = 2 and “same” padding



https://github.com/vdumoulin/conv_arithmetic/tree/master/gif

ReLU layer

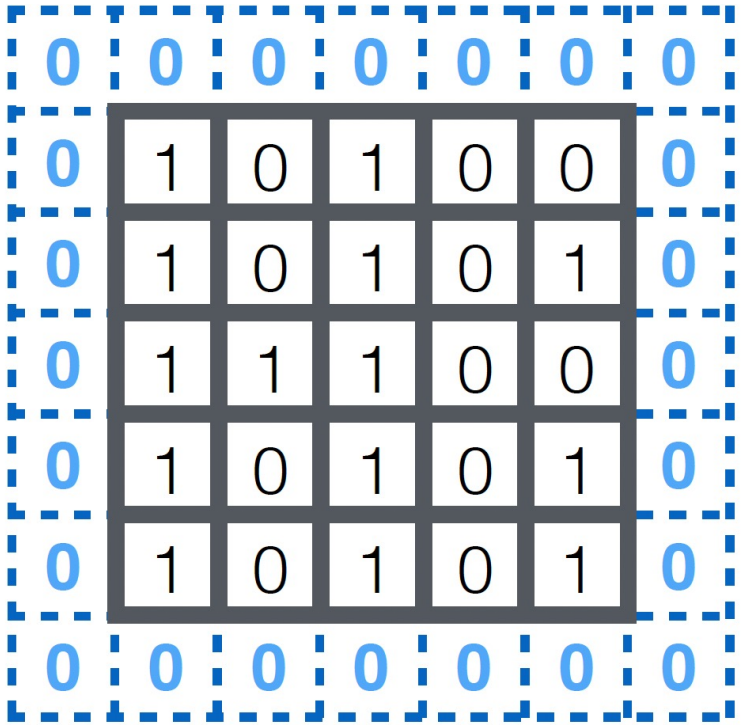
Convolutions and matrix multiplications are linear operations
Need to introduce non-linearity for solve complex classification problems!



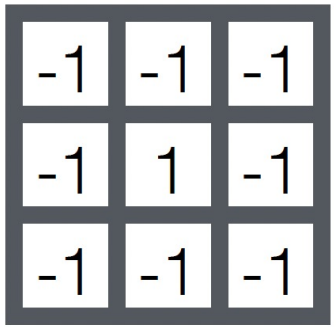
tf.keras.layers.ReLU

Convolution + ReLU layer

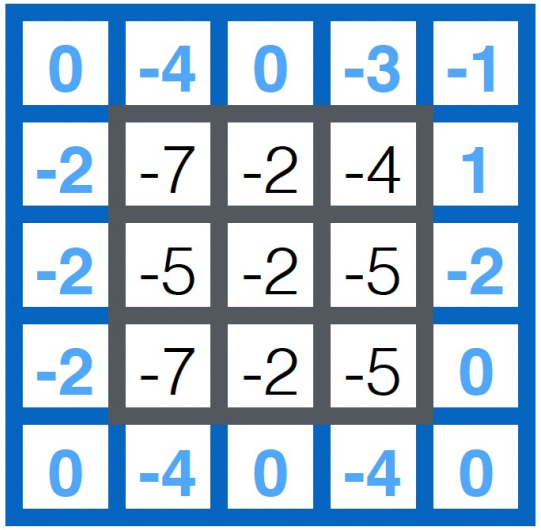
A 2D image:



A filter:

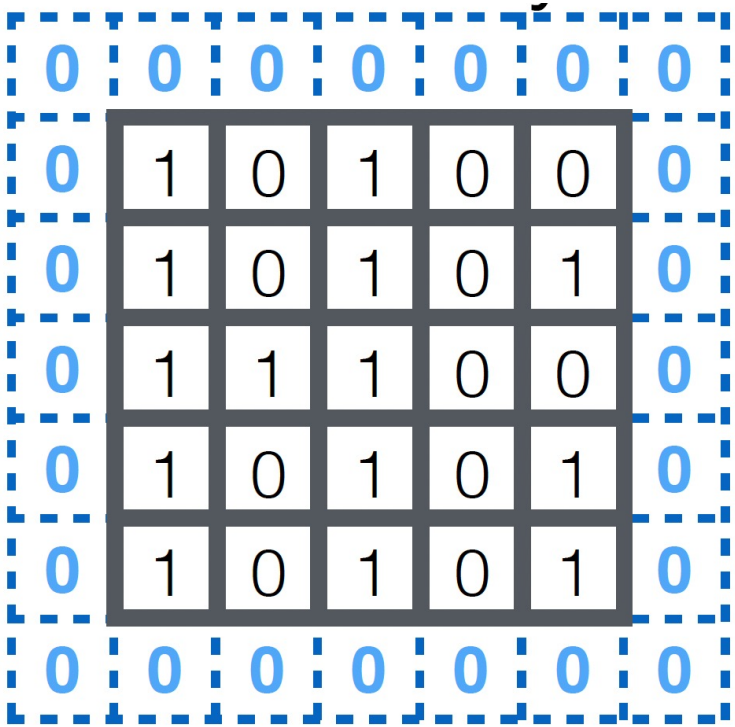


After convolution:

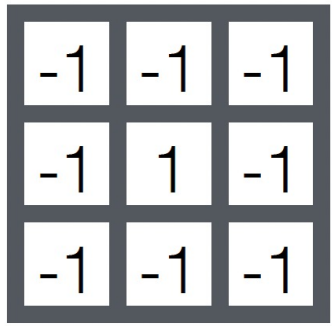


Convolution + ReLU layer

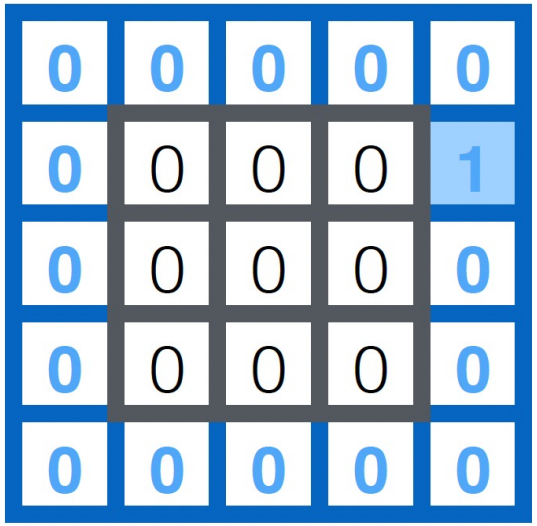
A 2D image:



A filter:



After convolution & ReLU:



Convolution + ReLU layer

A 2D image:

1	0	1	0	0
1	0	1	0	1
1	1	1	0	0
1	0	1	0	1
1	0	1	0	1

A filter:

w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}

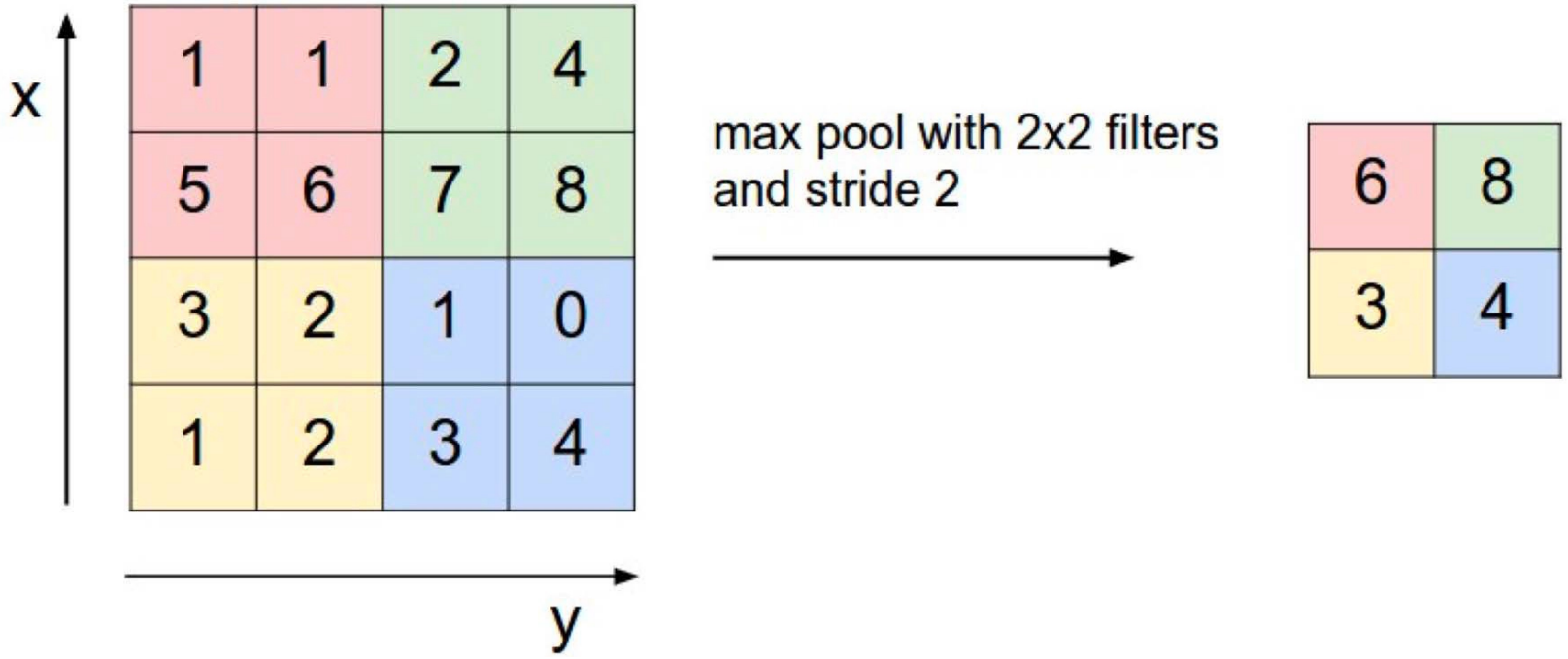
with bias b

How many weights?
3X3 + 1

Pooling layer

A pooling layer:

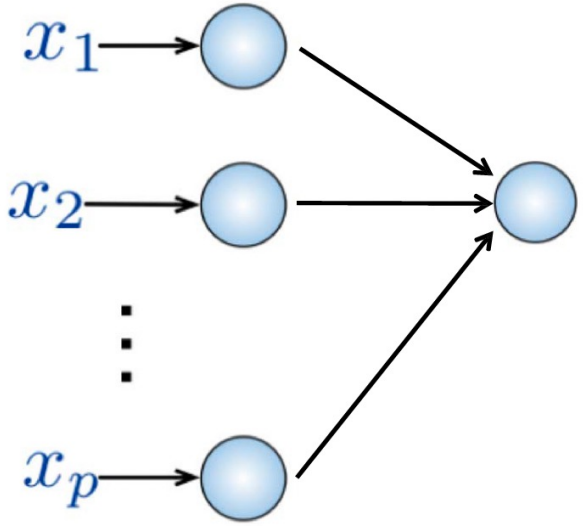
- Reduces the size of the image representation
- Speeds up the computation
- Makes some of the detected feature more robust
- (but) it gives up some spatial information!
- Operation: max, average …




```
tf.keras.layers.Max
    Pool2D(
    pool_size=(2,2),
)   strides=2
```

Fully-connected layers

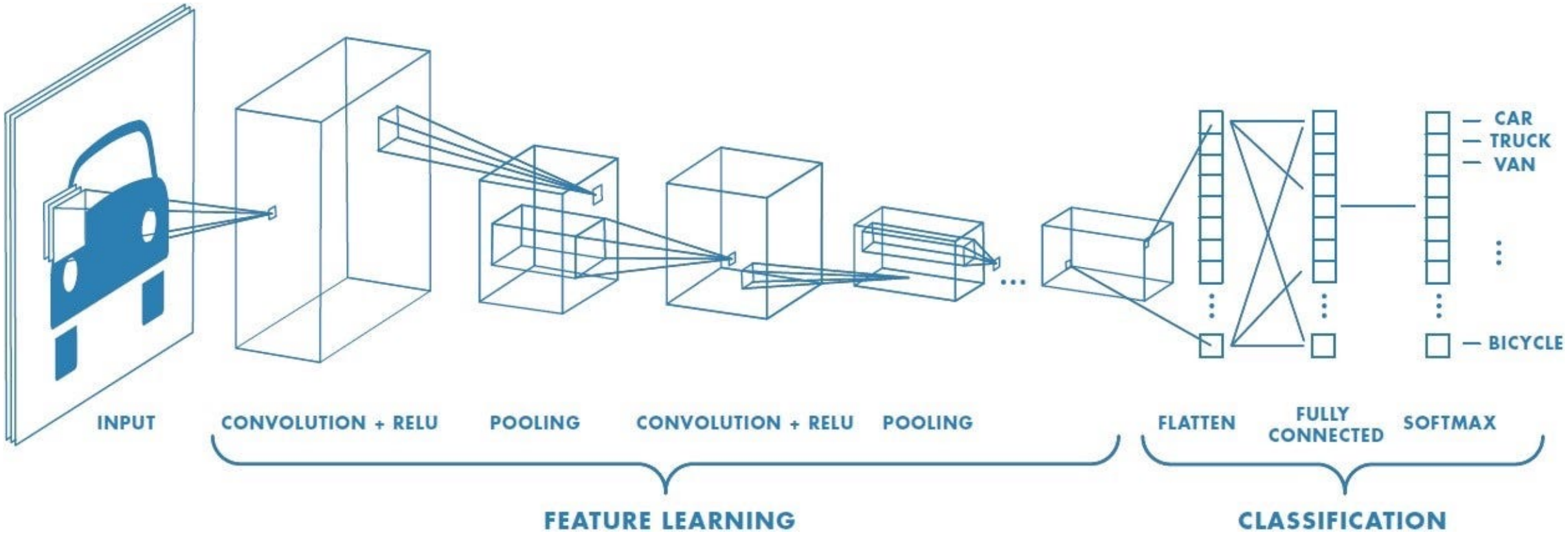
Input:
- 2D image
- Vector of pixel values



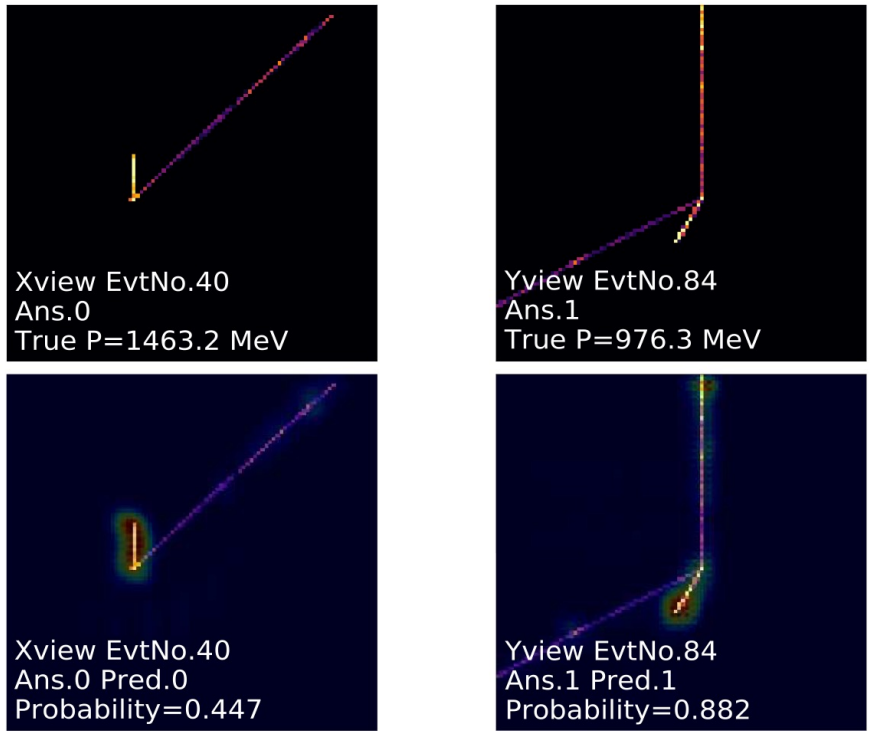
Fully Connected:
- Each neuron in hidden layer connected to all neurons in input layer
- No spatial information
- Many, many parameters

Key idea: Use **spatial structure** in input to inform architecture of the network

CNN building blocks



CNN application in HEP



https://doi.org/10.48550/arXiv.2106.09628

- Look for signals in complex detectors
- Classify events based on calorimeter image
- Data quality monitoring
- …

https://doi.org/10.48550/arXiv.1711.03573

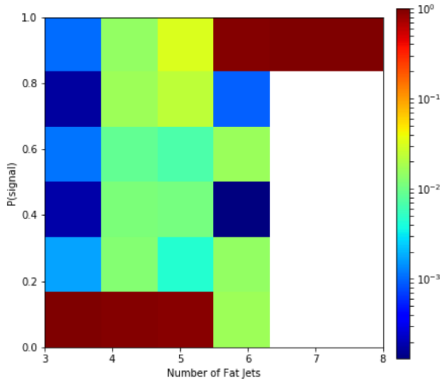


Figure 4: Comparison of CNN output with Number of Fat Jets

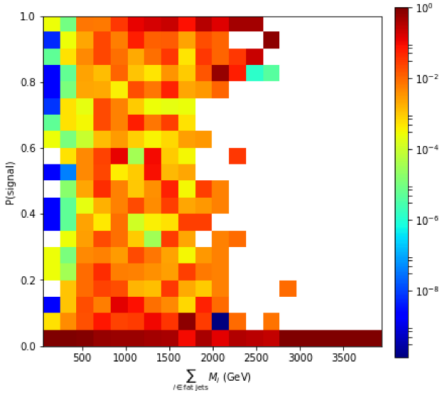


Figure 5: Comparison of CNN output with sum of jet mass

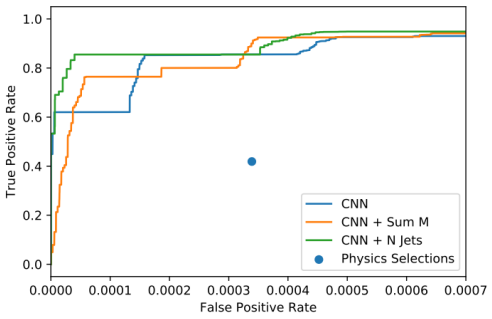


Figure 6: Change in performance on combining CNN output with jet variables

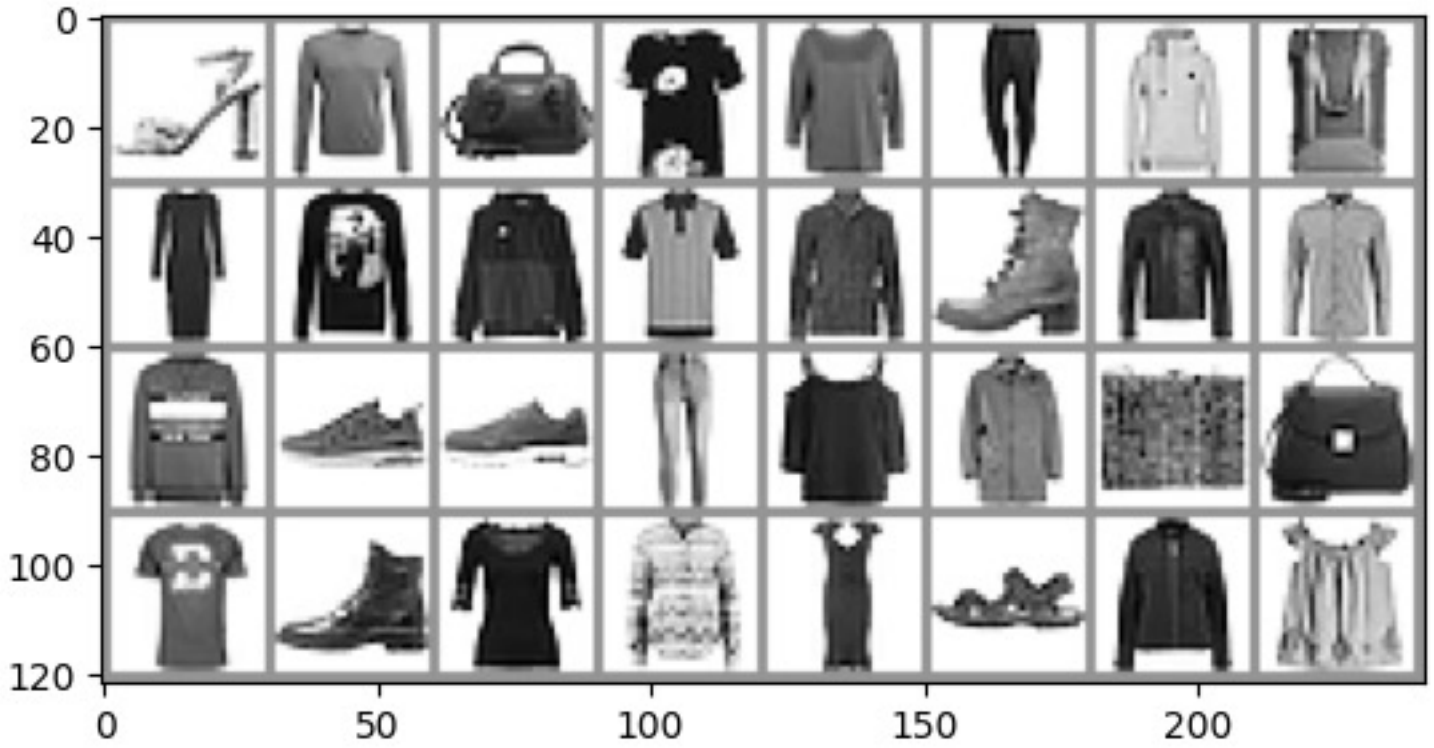
Hands-on!

Problem: image classification using the «Fashion MINST» dataset (https://www.kaggle.com/datasets/zalando-research/fashionmnist)

Notebook:

- Define and train a simple CNN
- Look at performance
- Improve it

- Extra: use Tensorboard for monitoring



https://github.com/fsimone91/course_ml4hep/tree/2024/notebooks/2024/3-CNN-pytorch.ipynb