Unione europea
Fondo sociale europeo

Regione Emilia-Romagna

ER Educazione Ricerca Emilia-Romagna

DATA LAB

GUARDA AVANTI

Big Data, nuove competenze
per nuove professioni.

FORM INDUSTRIA EMILIA ROMAGNA

ifoa Sapere utile

CERFORM

REISS ROMOLI
la passione della conoscenza

UNIMORE Università degli Studi di Modena e Reggio Emilia

Alma Mater Studiorum Università di Bologna

Università degli Studi di Ferrara

UNIVERSITÀ DI PARMA

POLITECNICO MILANO 1863 Polo Territoriale di Piacenza

UNIVERSITÀ CATTOLICA del Sacro Cuore

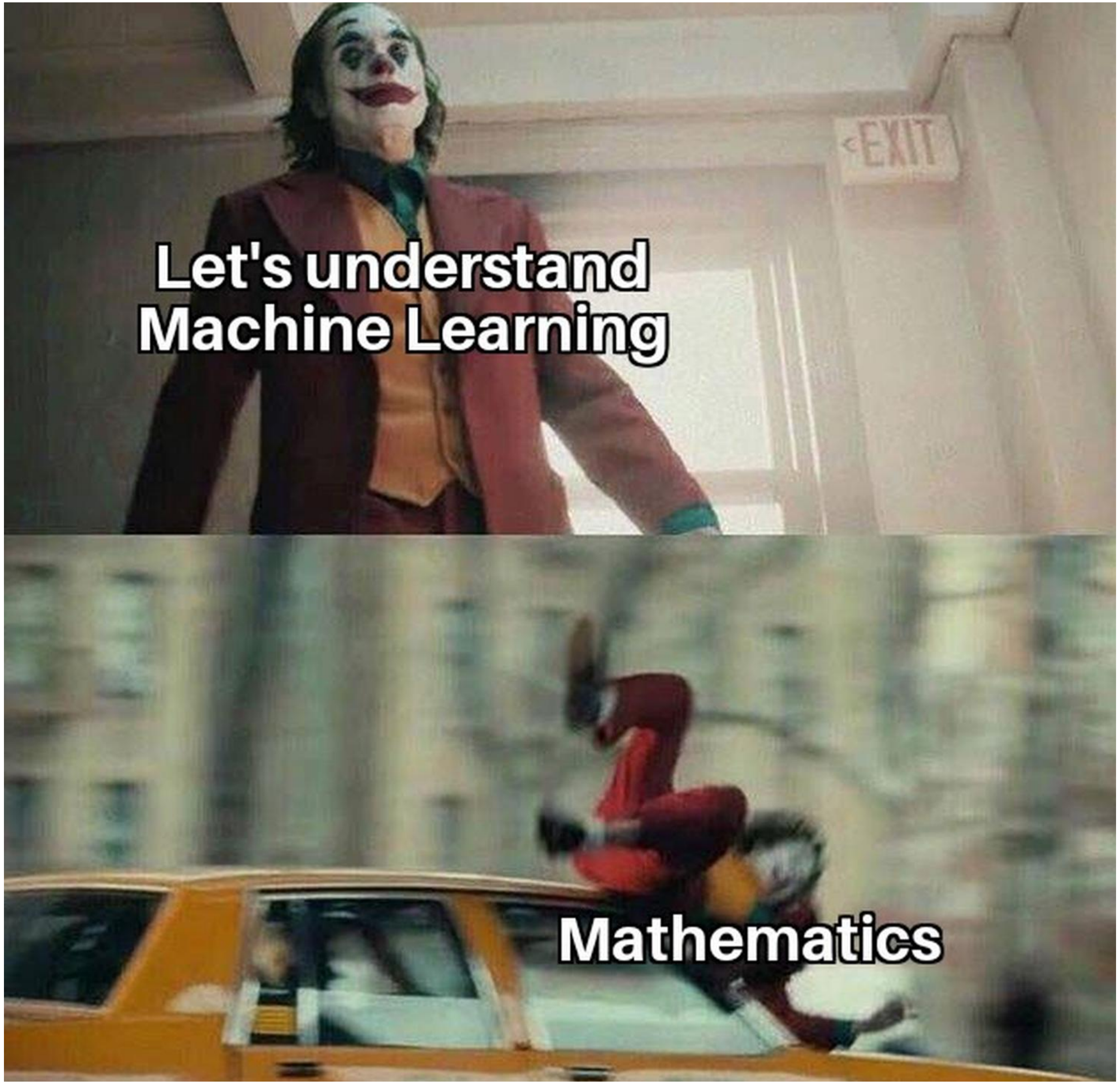
"Anticipare la crescita con le nuove competenze sui Big Data - Edizione 3" Operazione Rif. PA 2021-16029/RER approvata con DGR n° 927 del 21 giugno 2021 e co-finanziata dal Fondo Sociale Europeo PO 2014-2020 Regione Emilia-Romagna

Deep Learning

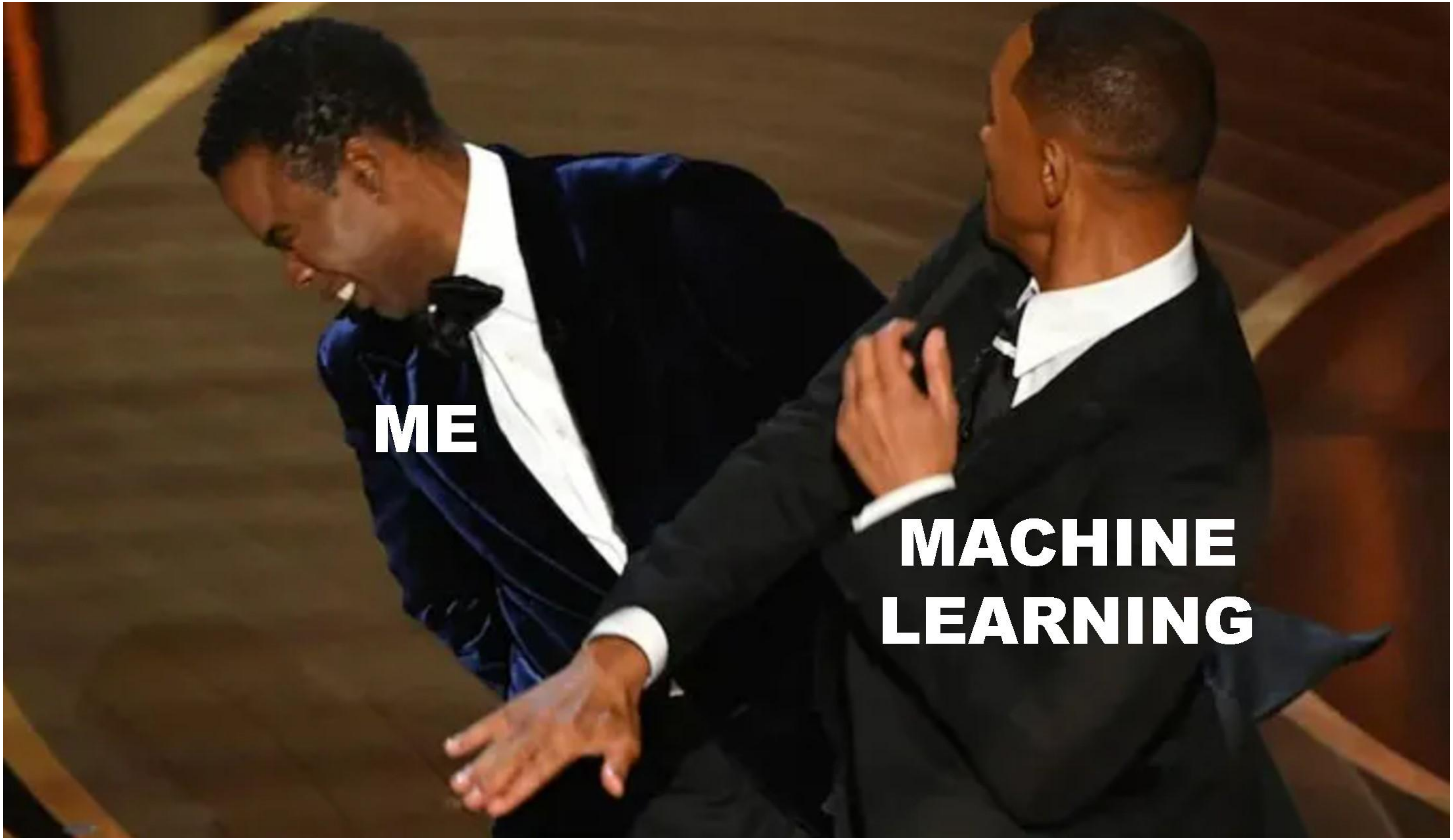
DATA LAB

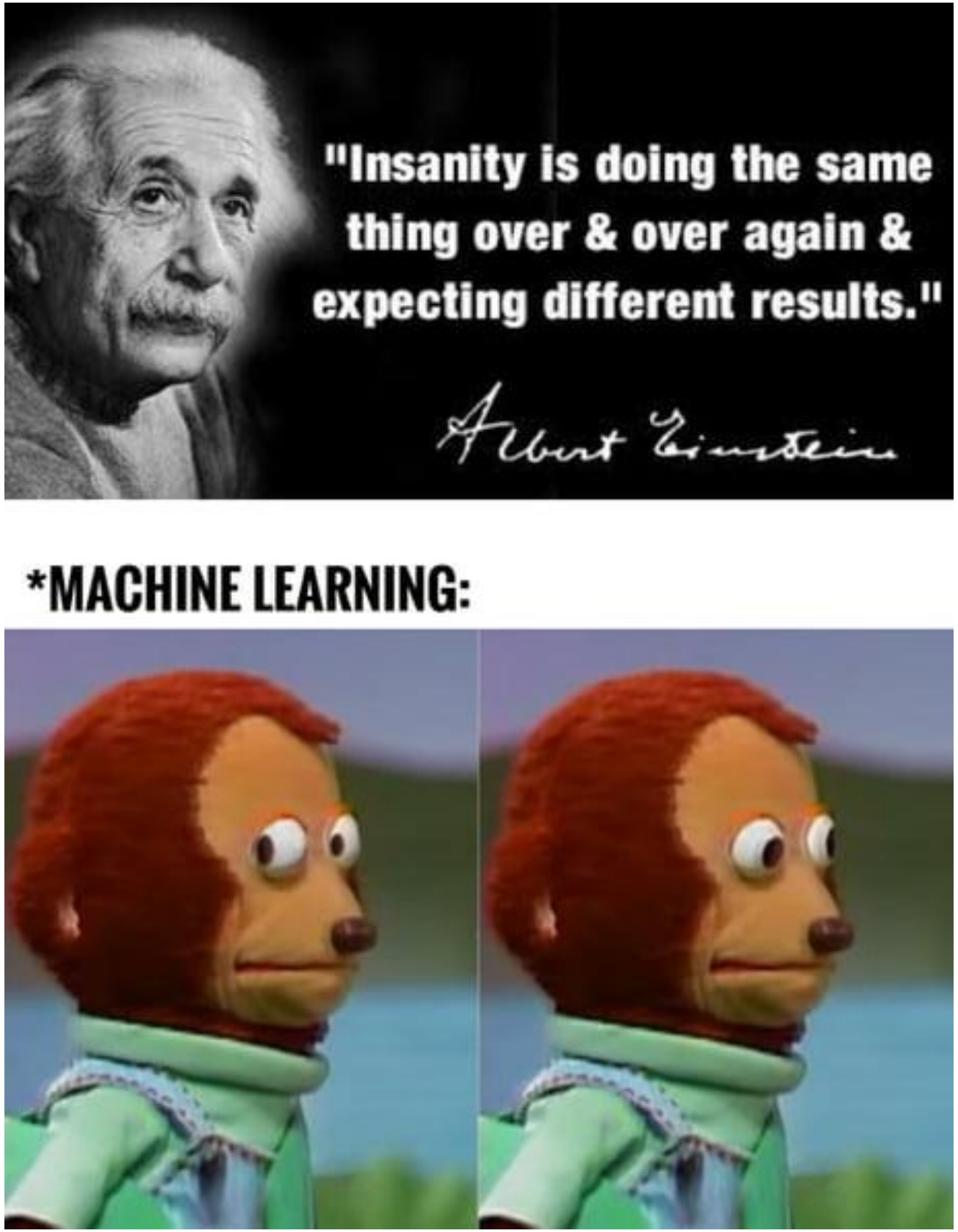
TOOL FOR CNN

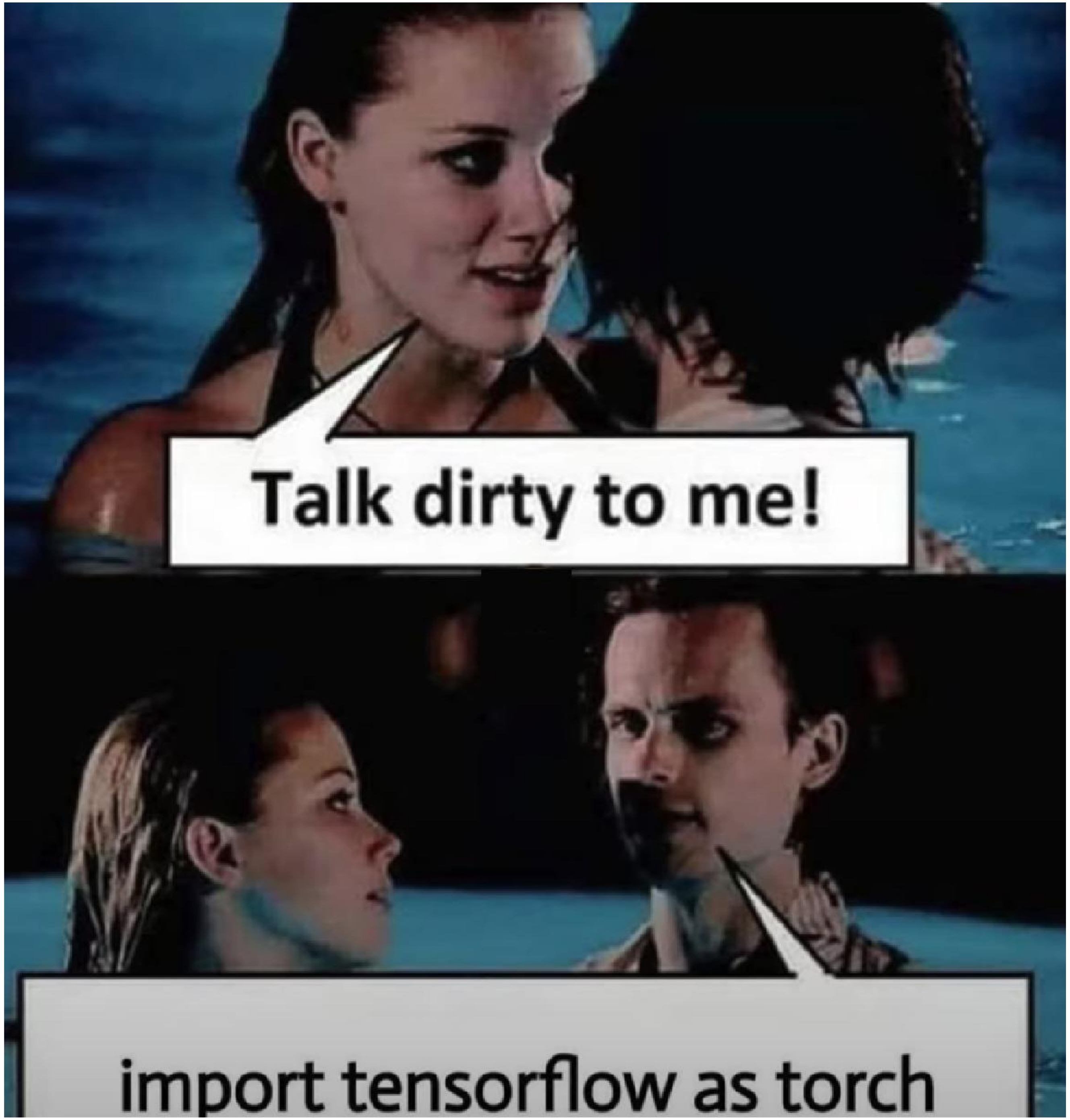

1999 - Nineteen Ninety nine
1888 - Eighteen Eighty Eight
1777 - Seventeen Seventy Seven
1111 - ????
ELEVEN HUNDRED ELEVEN
ONETEEN ONETY ONE
Me
AI



EXIT
Let's understand Machine Learning
Mathematics


ME
MACHINE
LEARNING

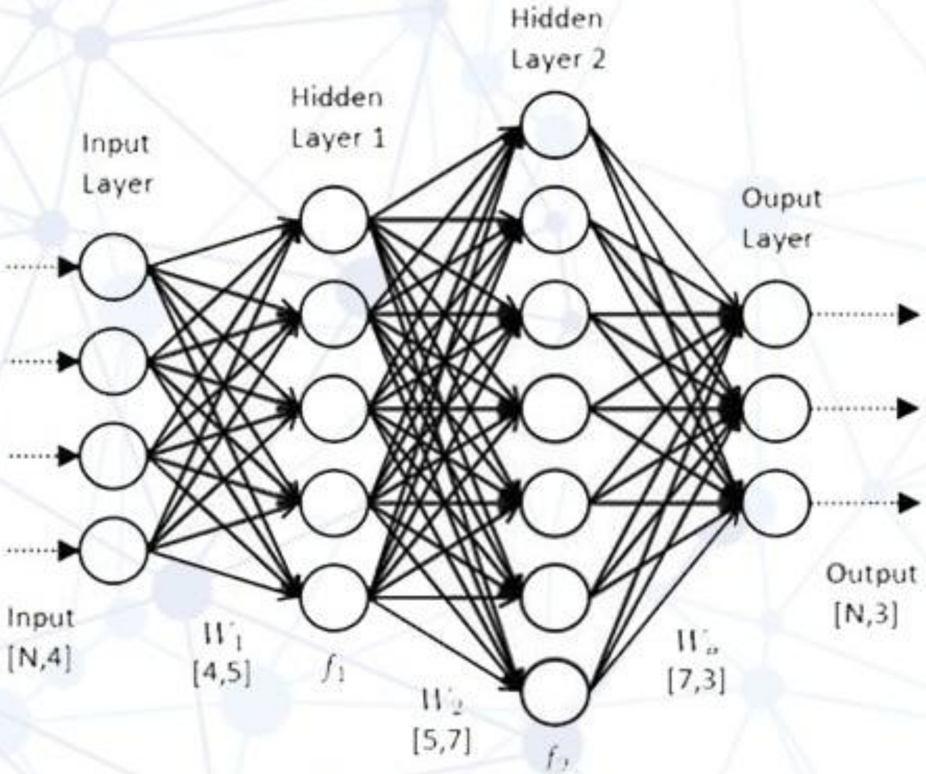

"Insanity is doing the same thing over & over again & expecting different results."
Albert Einstein
*MACHINE LEARNING:

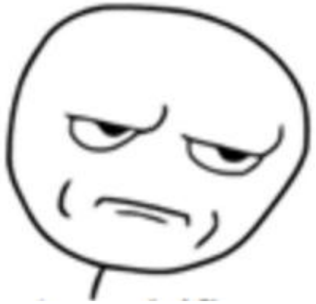

Talk dirty to me!
import tensorflow as torch

I WORK WITH MODELS

Others:

Me:

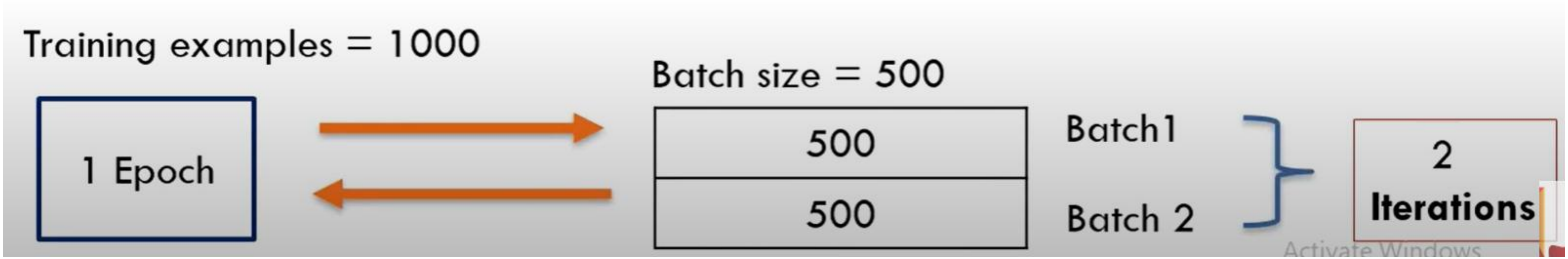




SURE! I CAN EASILY TUNE MY NEURAL NETWORK

REALITY

Example: if you have 1000 training examples, and your batch size is 500, then it will take 2 iterations to complete 1 epoch.



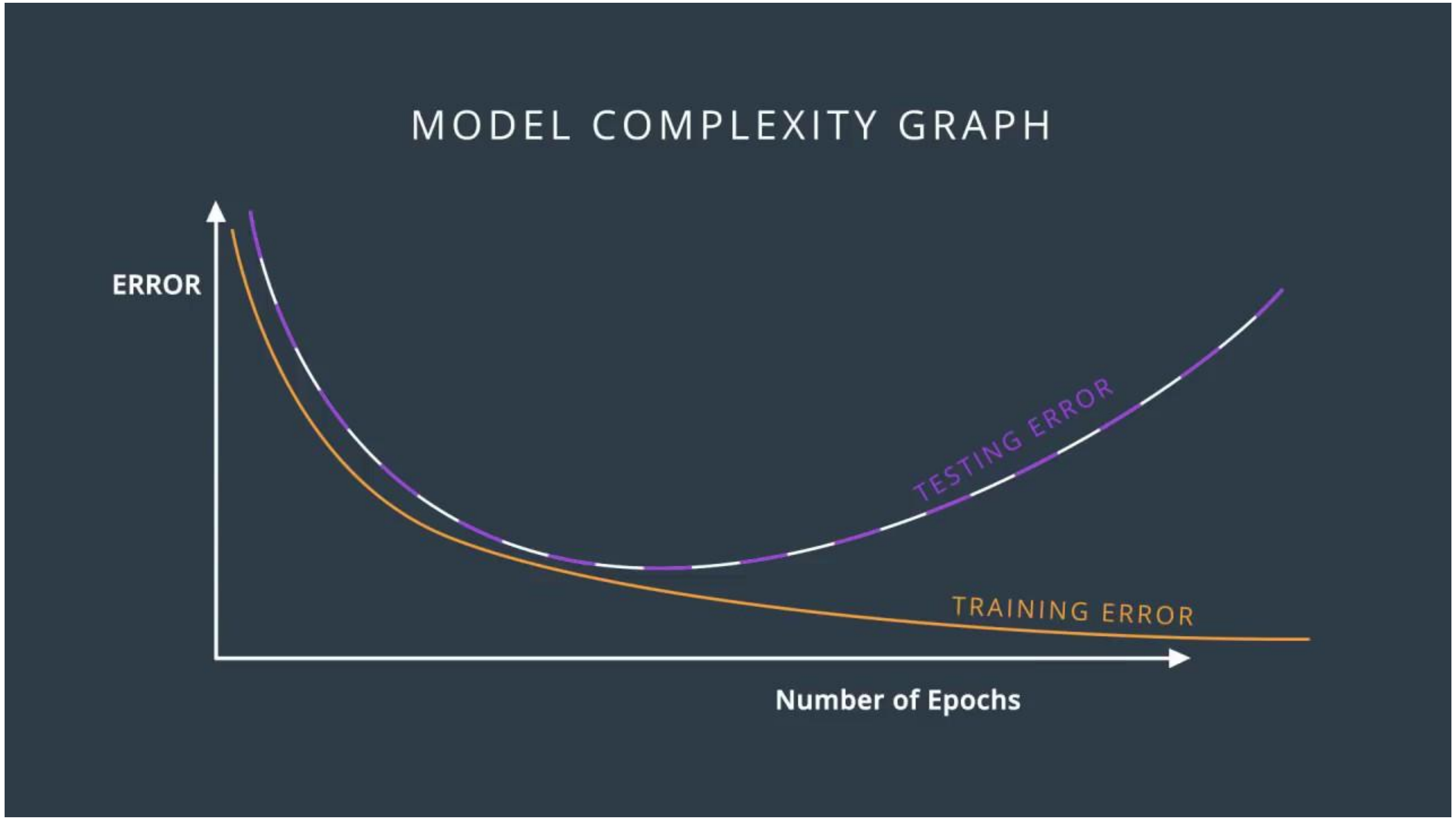
NEURAL NETWORKS ARE SO COOL!

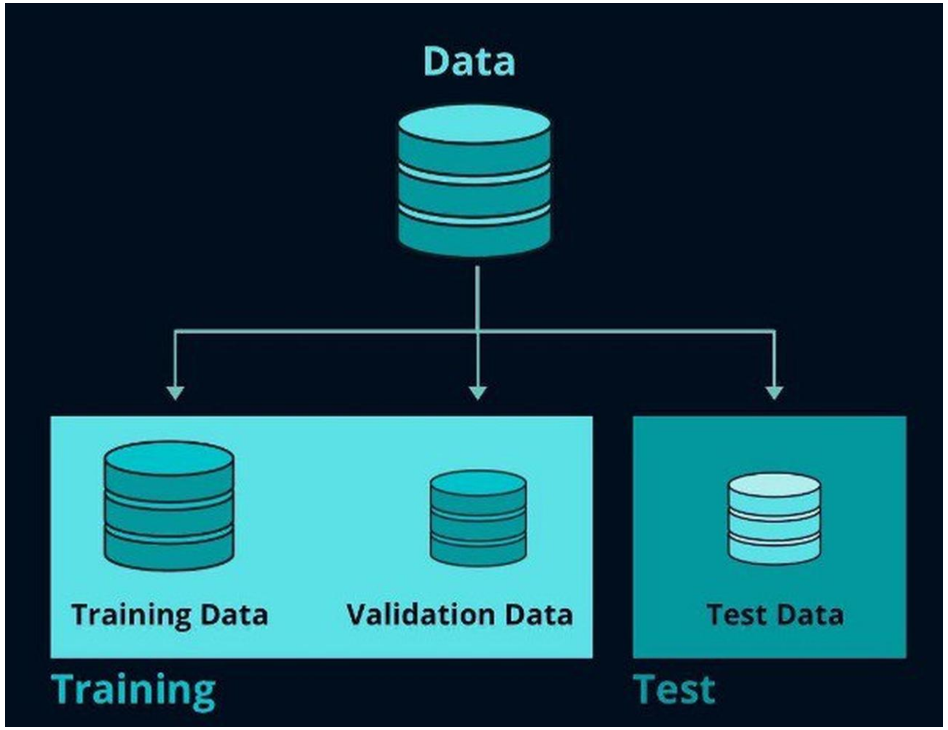
Remember:

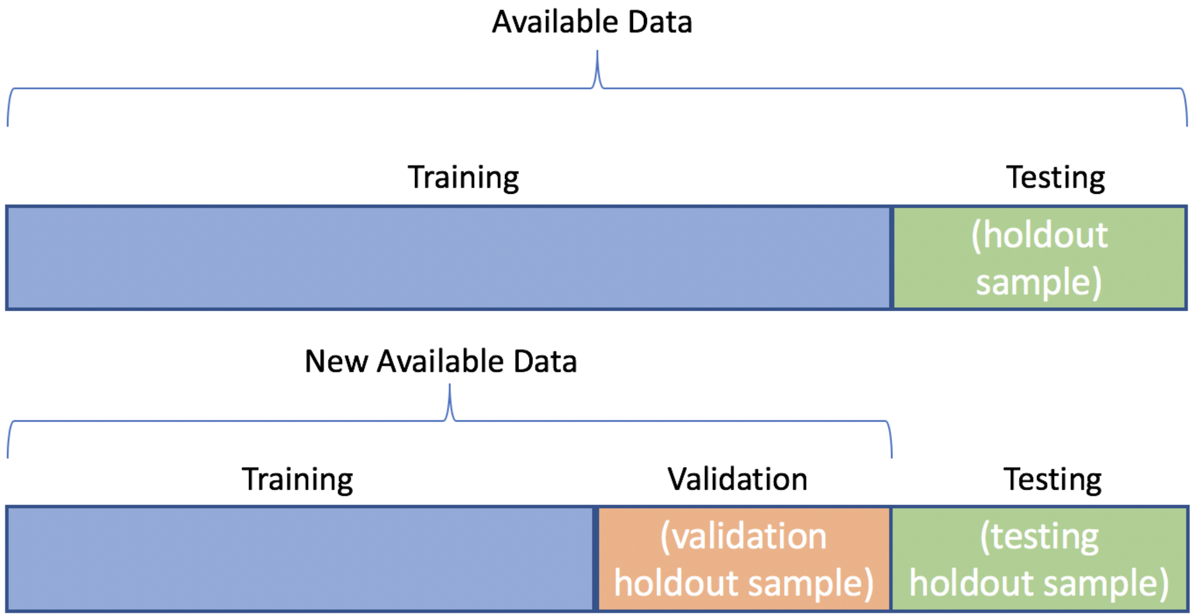
We train our networks with gradient descent

$$W \leftarrow W - \eta \frac{\partial J(W, x, y)}{\partial W}$$

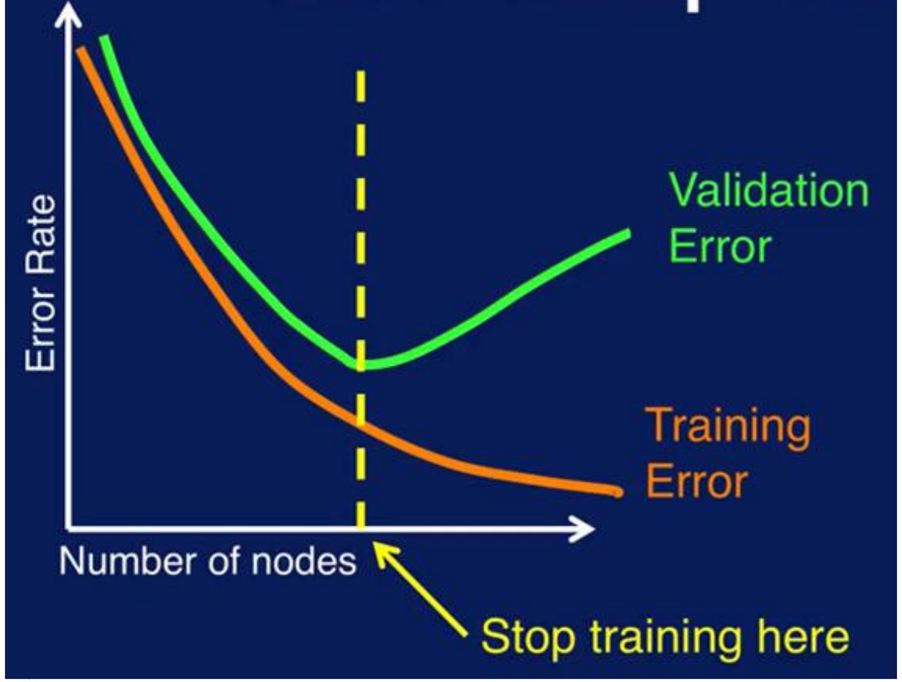
"How does a small change in weights decrease our loss"

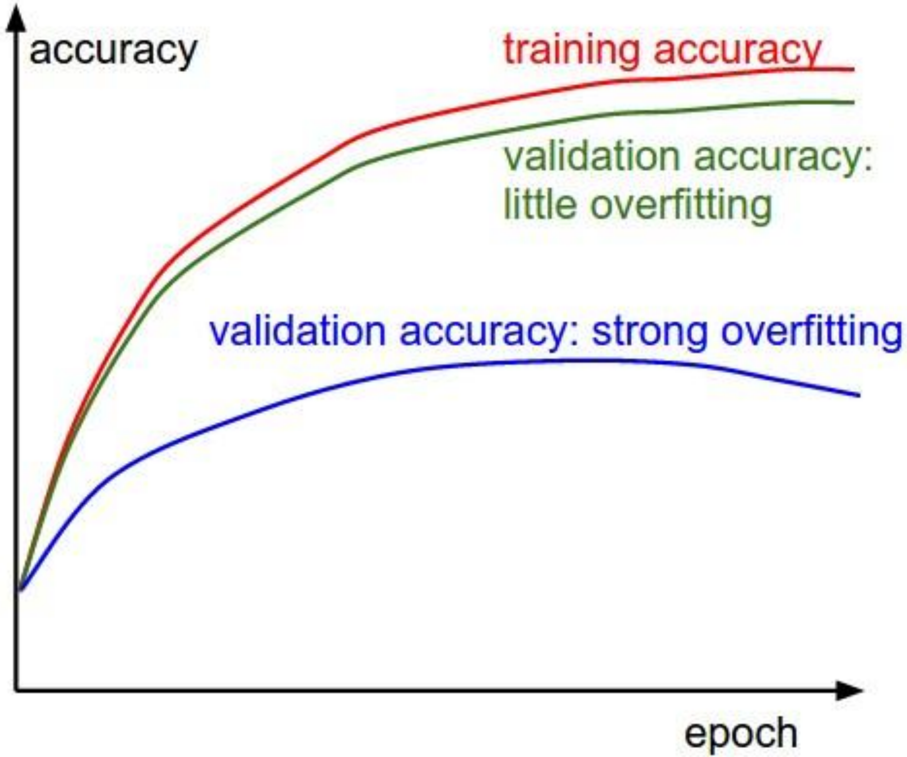


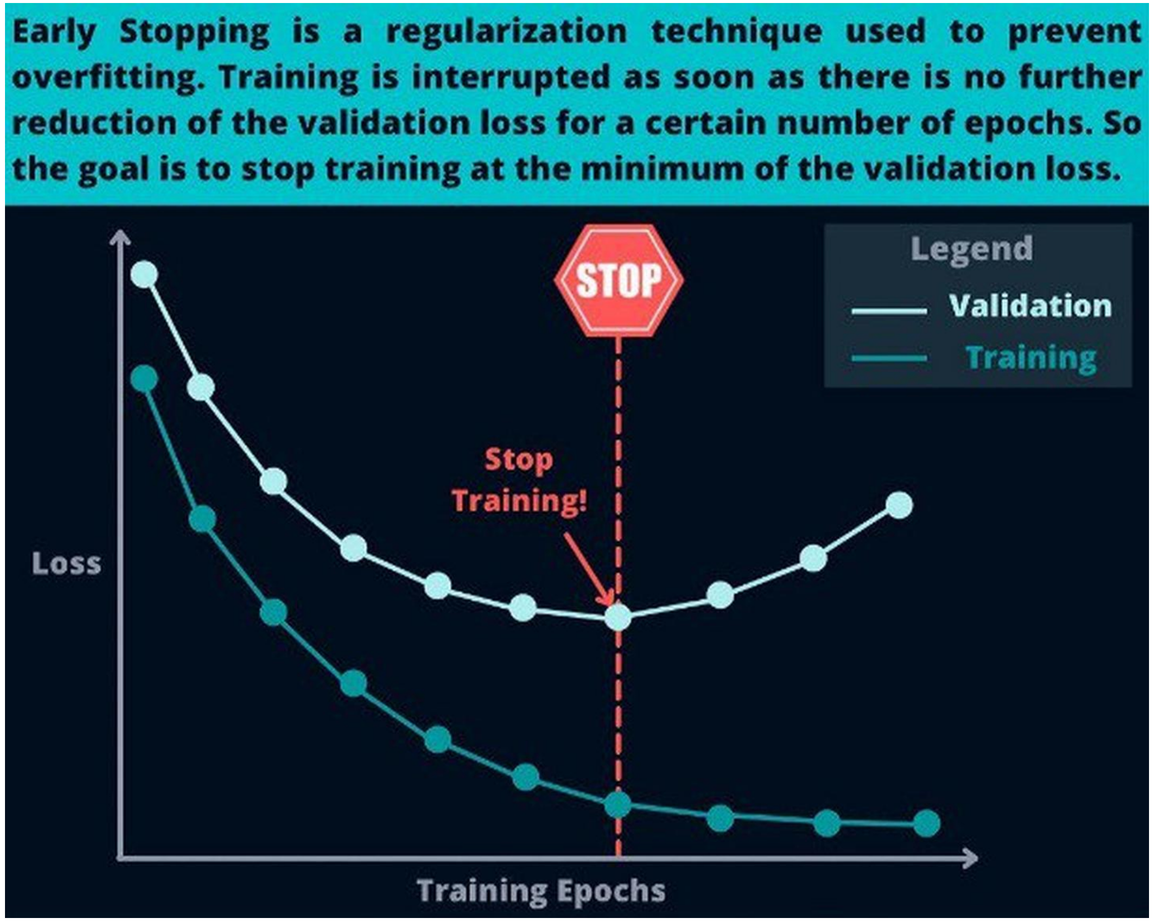

Data
Training Data
Validation Data
Test Data
Training
Test


Available Data
Training
Testing
(holdout sample)
New Available Data
Training
Validation
Testing
(validation holdout sample)
(testing holdout sample)


My model
Training data
Validation data
Testing data


Error Rate
Validation
Error
Training
Error
Number of nodes
Stop training here

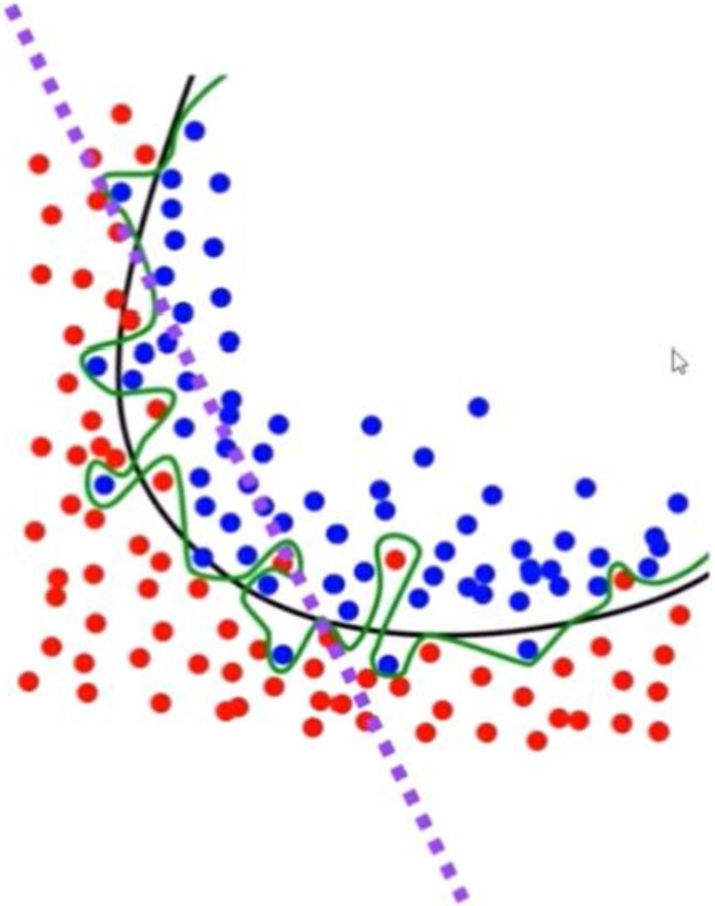

accuracy
training accuracy
validation accuracy:
little overfitting
validation accuracy: strong overfitting
epoch


Early Stopping is a regularization technique used to prevent overfitting. Training is interrupted as soon as there is no further reduction of the validation loss for a certain number of epochs. So the goal is to stop training at the minimum of the validation loss.
STOP
Legend
Validation
Training
Stop
Training!
Loss
Training Epochs

Overfitting vs Underfitting

Overfitting occurs when a model is excessively complex, such as having too many parameters relative to the number of observations. A model that has been overfit has poor predictive performance, as it overreacts to minor fluctuations in the test data.

Underfitting occurs when a model cannot capture the underlying trend of the data. This trouble means the model must have more variables to explain the data.. The purple dotted line is an example.



Regularization

LASSO, RIDGE AND ELASTICNET

Regularization is a collection of techniques that can be used to prevent over-fitting. Regularization adds information to a problem in the form of penalty against complexity to find simplest model that explains the data.

Our goal is to minimize the sum of the unpenzalized cost function plus a penalty term wich can be understood as adding bias and preferring a simpler model to reduce the variance getting smaller weights, then we penalize large weights.

Penalized model is about trade offs, because we reduce the variance and tend to avoid over-fitting. The overall result might generalize better to test data or unseen data.

λ is a parameter that controls the strength of the penalty.

Regularization

LASSO, RIDGE AND ELASTICNET

L1 penalty penalize the model by the sum of the absolute values of the coeficciients, while an L2 penalty penalizes by the sum of squares.

L1 penalized model is called **Lasso**, while L2 penalized one is known as **Ridge Regression**.
When using both in a model the name is **ElasticNet**

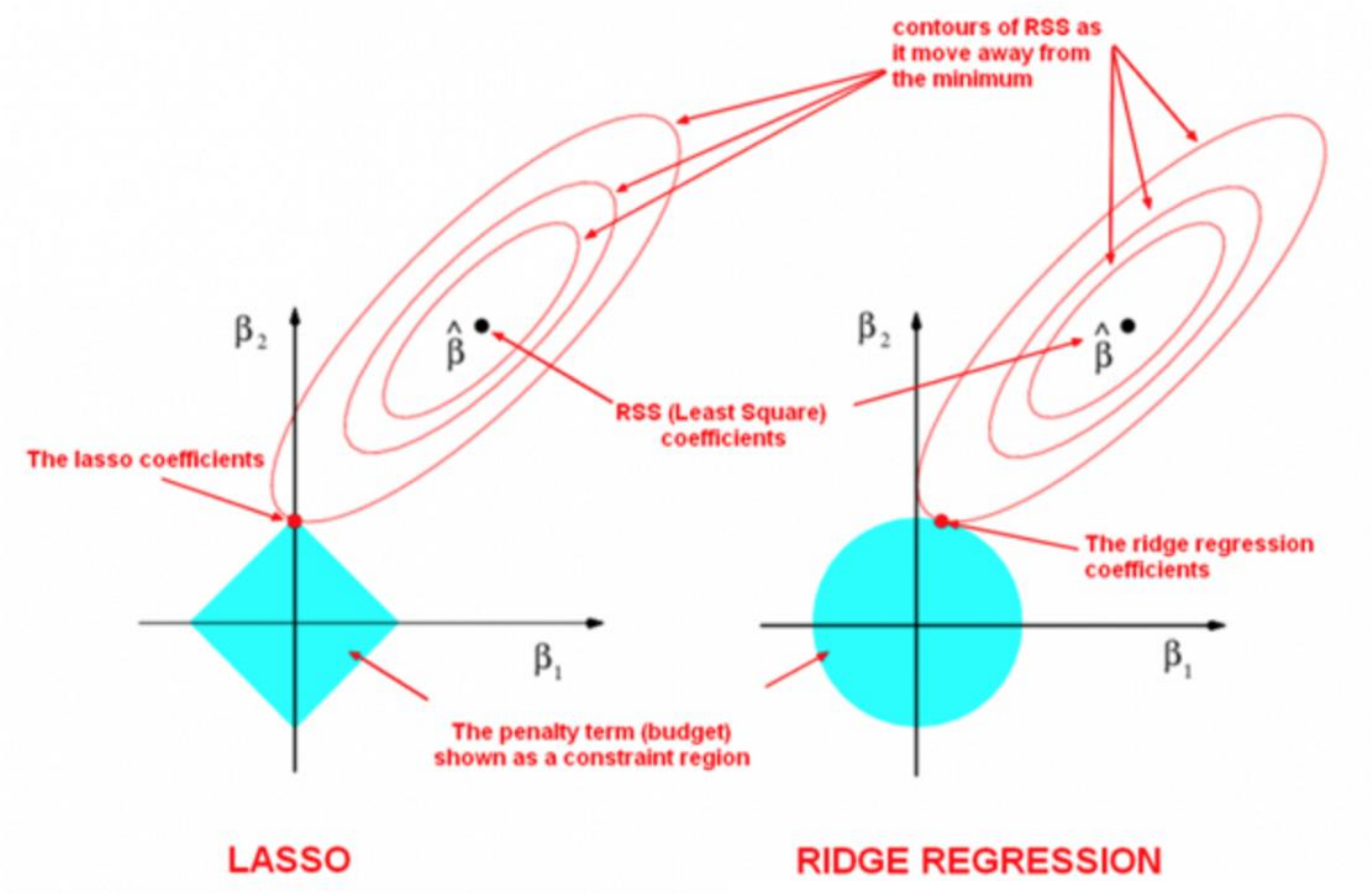
$$L2: \ = \lambda \sum_{j=1}^{m} w_j^2$$

$$L1: \ = \lambda \sum_{j=1}^{m} |w_j|$$

$$Cost_{\text{ridge}} = \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda \sum_{j=1}^{m} w_j^2$$

$$Cost_{\text{lasso}} = \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda \sum_{j=1}^{m} |w_j|$$

LASSO - RIDGE



L1 Regularization

$$\text{Cost} = \sum_{i=0}^{N}\left(y_i - \sum_{j=0}^{M} x_{ij}W_j\right)^2 + \lambda \sum_{j=0}^{M} |W_j|$$

L2 Regularization

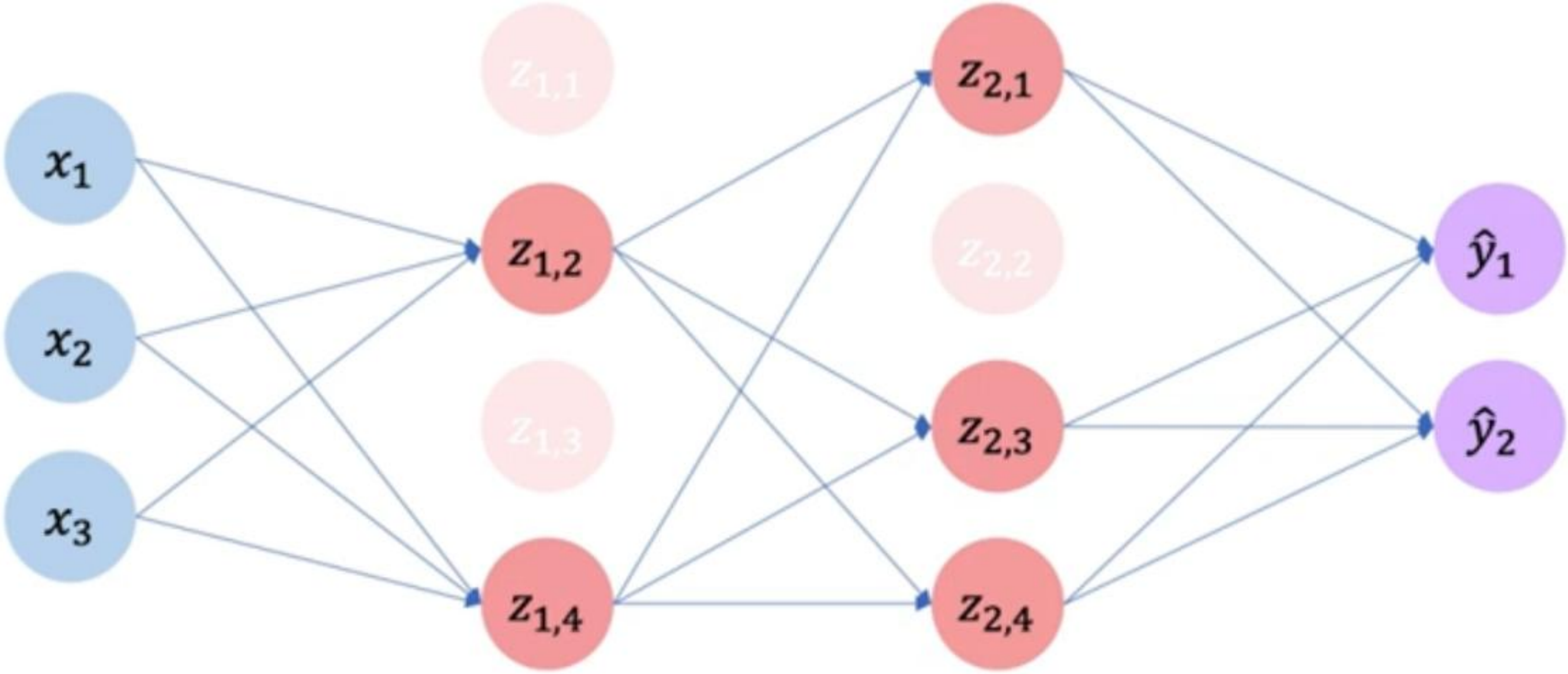
$$\text{Cost} = \underbrace{\sum_{i=0}^{N}\left(y_i - \sum_{j=0}^{M} x_{ij}W_j\right)^2}_{\text{Loss function}} + \underbrace{\lambda \sum_{j=0}^{M} W_j^2}_{\text{Regularization Term}}$$

Comparison of L1 and L2 regularization	
L1 regularization	***L2 regularization***
Sum of absolute value of weights	Sum of square of weights
Sparse solution	Non-sparse solution
Multiple solutions	One solution
Built-in feature selection	No feature selection
Robust to outliers	Not robust to outliers (due to the square term)

Regularization I: Dropout

- During training, randomly set some activations to 0
    - Typically 'drop' 50% of activations in layer
    - Forces network to not rely on any 1 node

```
tf.keras.layers.Dropout(p=0.5)
```



Deep Learning = Alchemy?

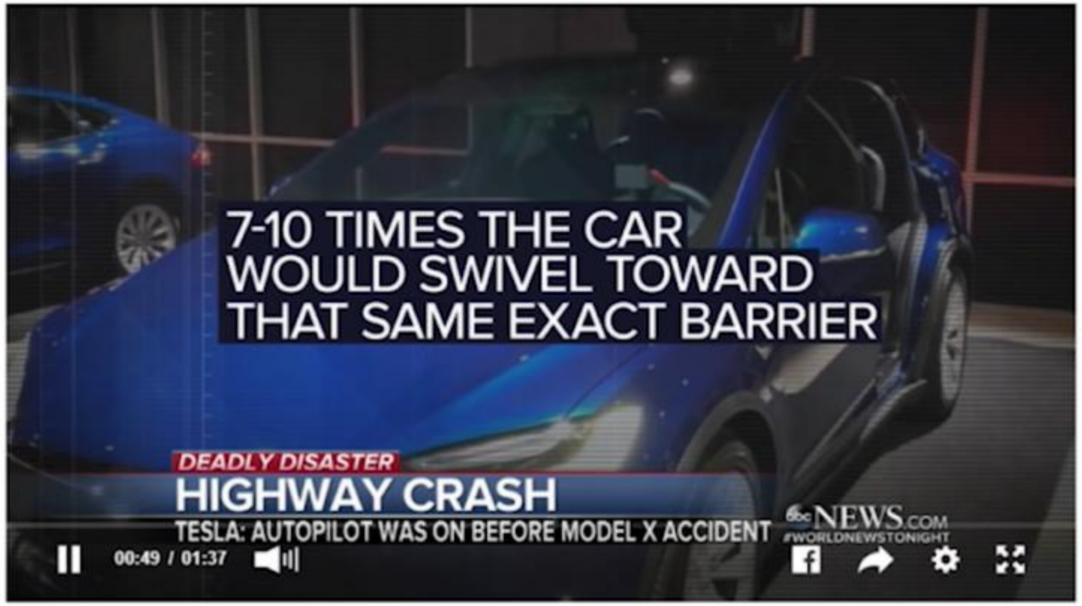
Neural Network Failure Modes, Part II

Tesla car was on autopilot prior to fatal crash in California, company says

The crash near Mountain View, California, last week killed the driver.

By Mark Osborne

March 31, 2018, 1:57 AM • 5 min read




Data Scientist
Data Expections
from clients
VS
Reality

Neural networks: expectation vs reality

Expectation:
Training on a your dataset

Reality:
Testing in reality

Dogs

Driving

my deep
learning model
on toy data

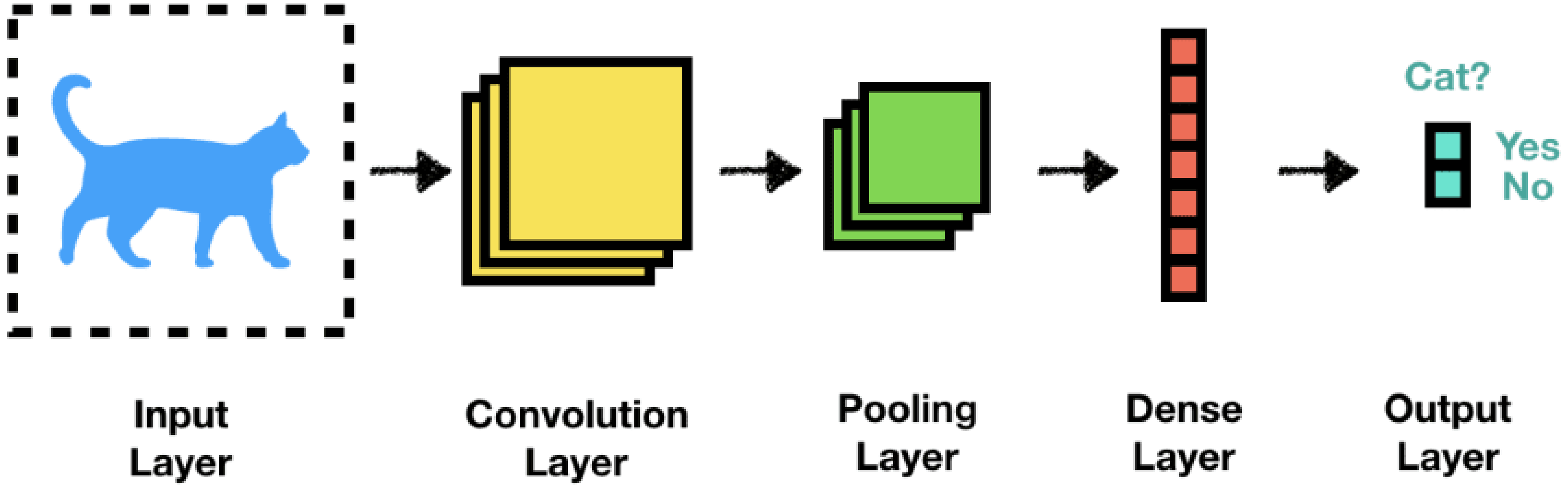
same model
on real-world
data

Neural Network Limitations…

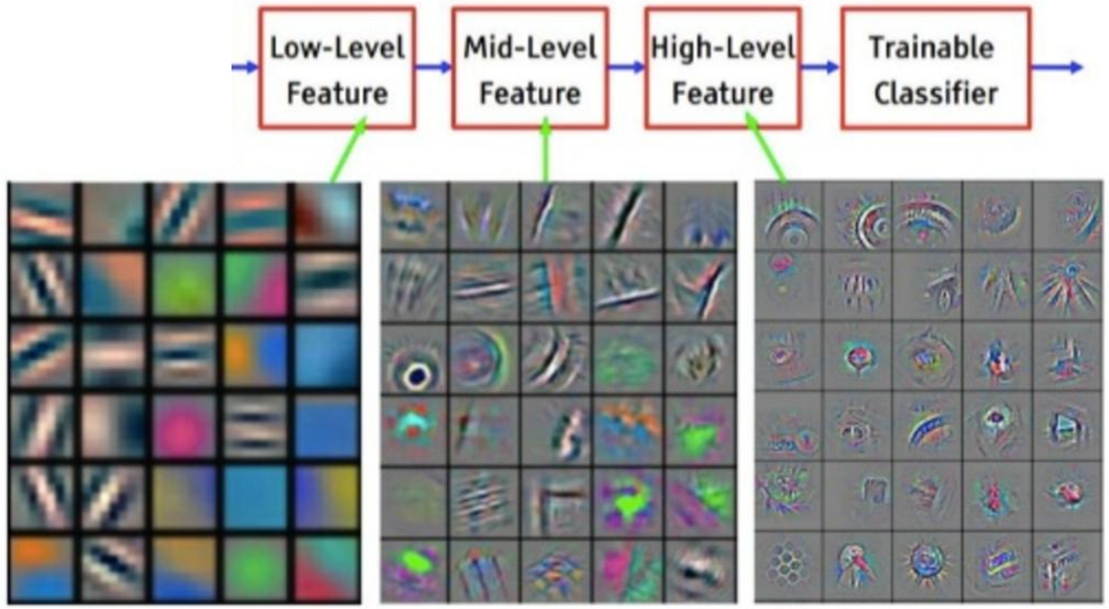
- Very **data hungry** (eg. often millions of examples)
- **Computationally intensive** to train and deploy (tractably requires GPUs)
- Easily fooled by **adversarial examples**
- Can be subject to **algorithmic bias**
- Poor at **representing uncertainty** (how do you know what the model knows?)
- Uninterpretable **black boxes**, difficult to trust
- Difficult to **encode structure** and prior knowledge during learning
- **Finicky to optimize**: non-convex, choice of architecture, learning parameters
- Often require **expert knowledge** to design, fine tune architectures

Likelihood estimation in deep learning

	Classification (discrete)	Regression (continuous)
Targets	$y \in \{1, \cdots, K\}$	$y \in \mathbb{R}$
Likelihood	$y \sim \text{Categorical}(\boldsymbol{p})$ `tfp.distributions.Categorical(probs=p)`	$y \sim \text{Normal}(\mu, \sigma^2)$ `tfp.distributions.Normal(mu, sigma)`
Parameters	$\boldsymbol{p} = \{p_1, \cdots, p_K\}$	(μ, σ^2)
Constraints	$\sum_i p_i = 1; \quad p_i > 0$	$\mu \in \mathbb{R}; \quad \sigma > 0$
Loss function	Cross Entropy $-\sum_{i=1}^{K} y_i \log p_i$ `dist.cross_entropy(y)`	Negative Log-Likelihood $-\log\left(\mathcal{N}(y\|\mu, \sigma^2)\right)$ `-1 * dist.log_prob(y)`



COSA VEDE LA RETE NEURALE?



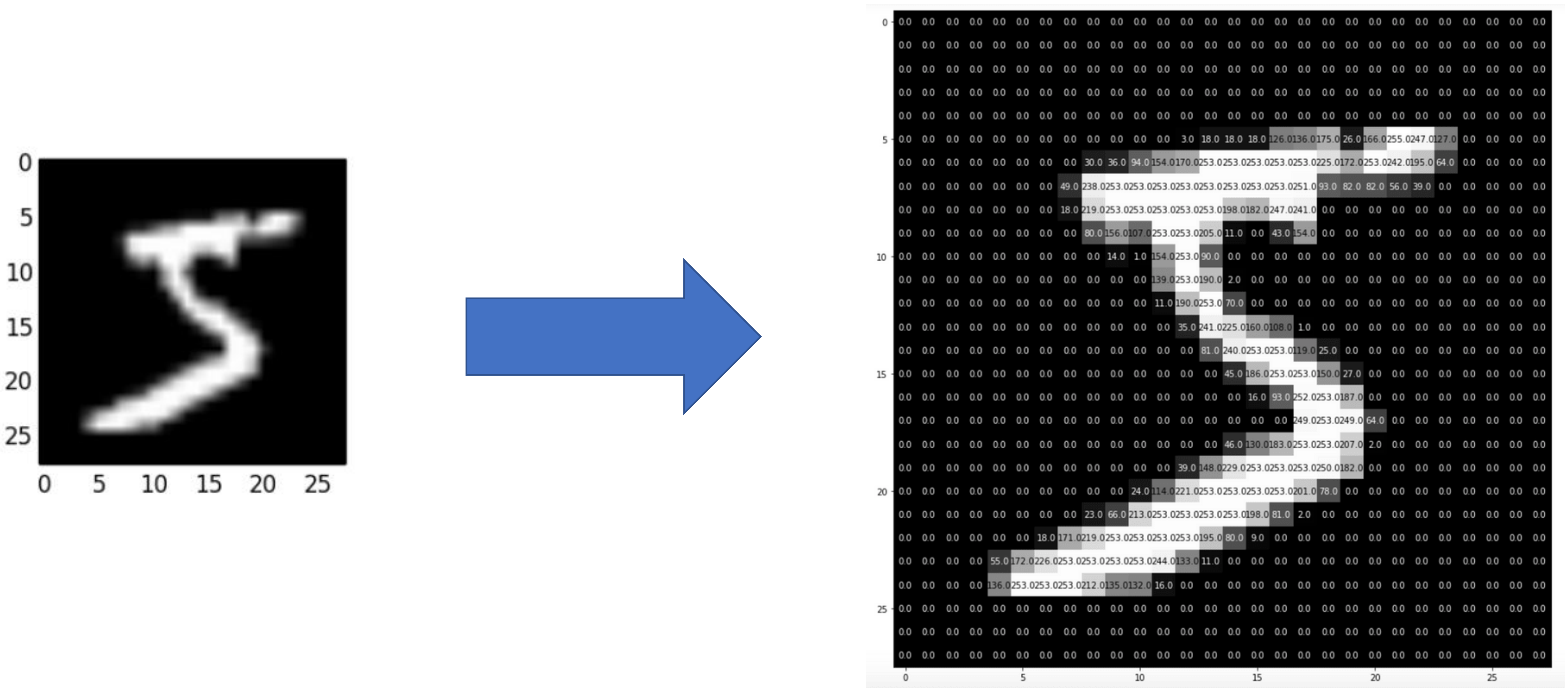
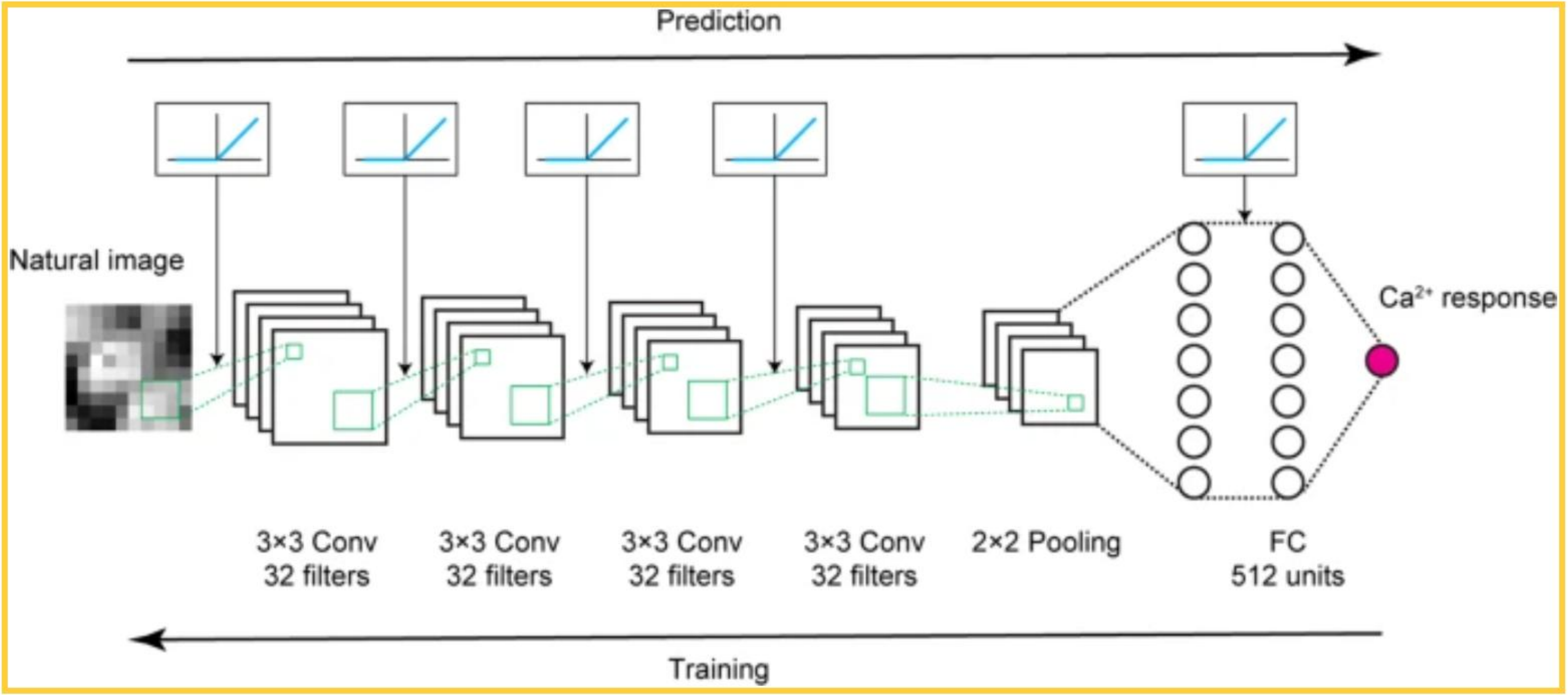
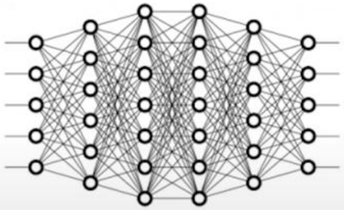

0
5
10
15
20
25

Figure 1

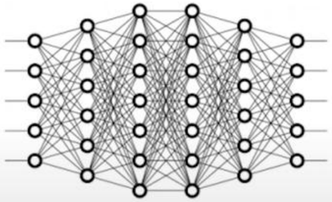

Prediction
Natural image
Ca²⁺ response
3×3 Conv
32 filters
3×3 Conv
32 filters
3×3 Conv
32 filters
3×3 Conv
32 filters
2×2 Pooling
FC
512 units
Training



$$p(\text{“}cat\text{”}) = 0.5$$

$$p(\text{“}dog\text{”}) = 0.5$$

The output likelihoods will be unreliable if the input is ***unlike anything during training***

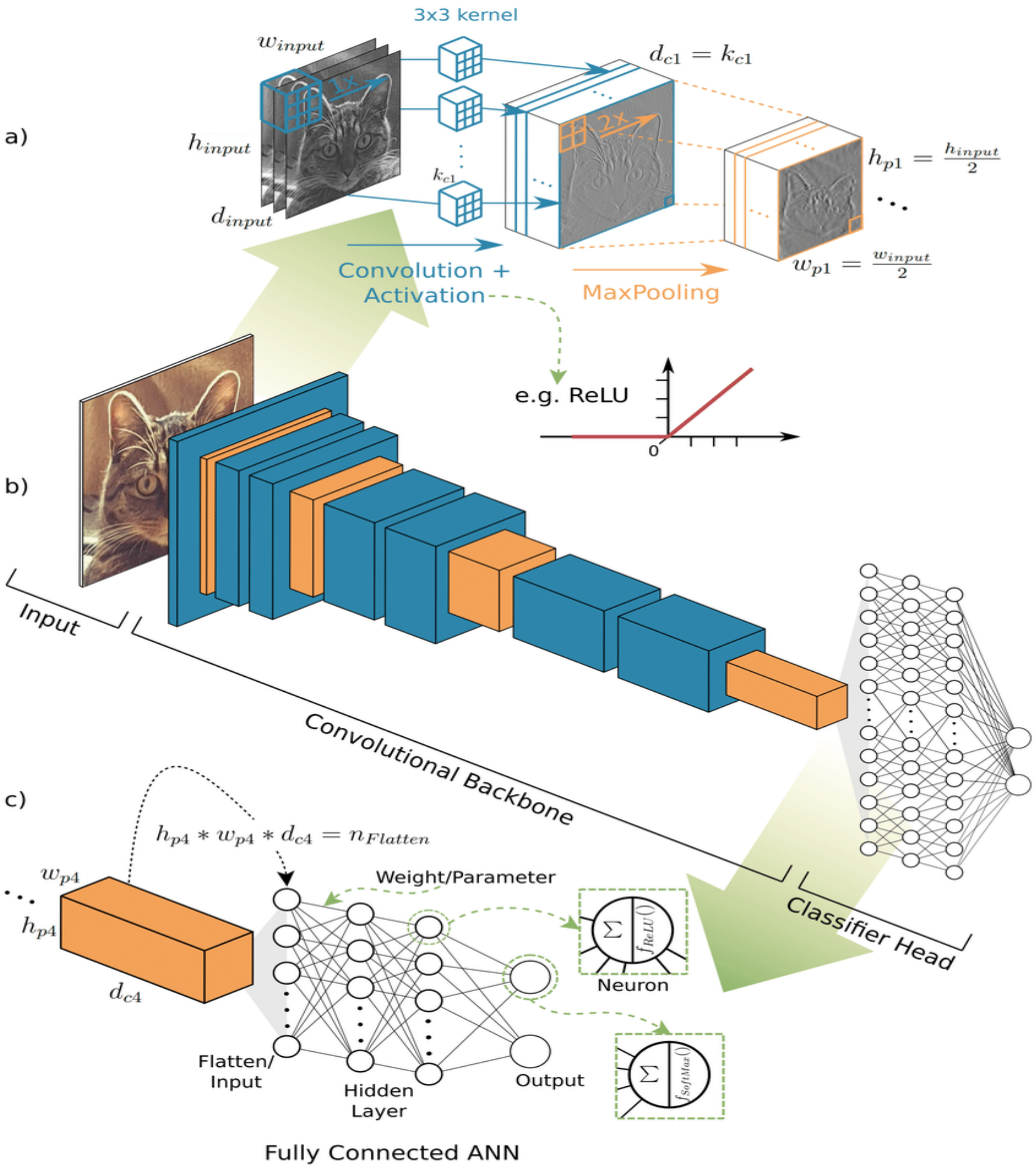


$$p(\text{“}cat\text{”}) = 0.15$$

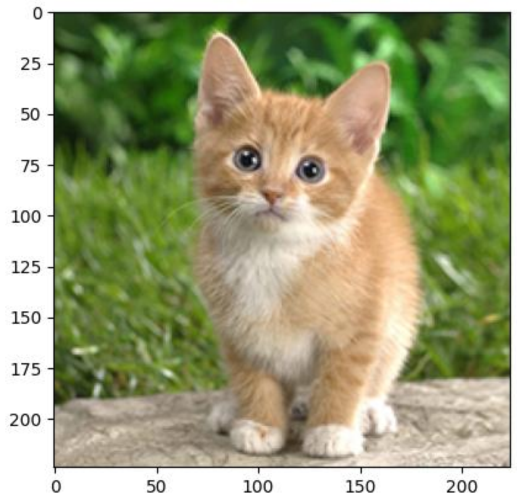
$$p(\text{“}dog\text{”}) = 0.85$$

★ $p(\text{“}cat\text{”}) + p(\text{“}dog\text{”}) = 1$ ★

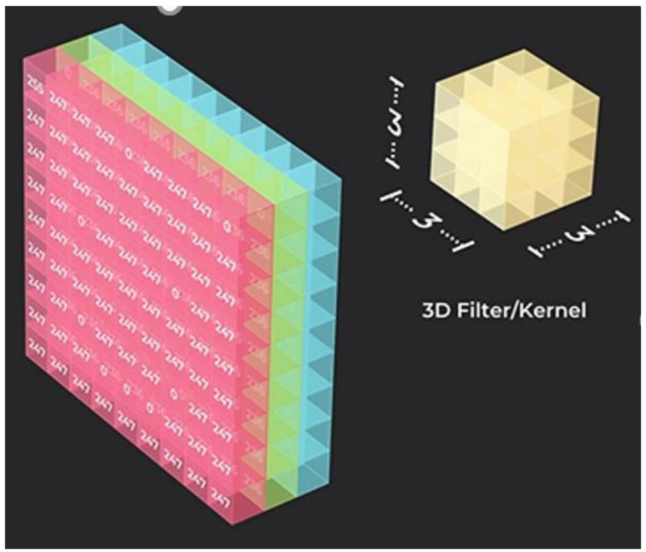
How to confuse machine learning:



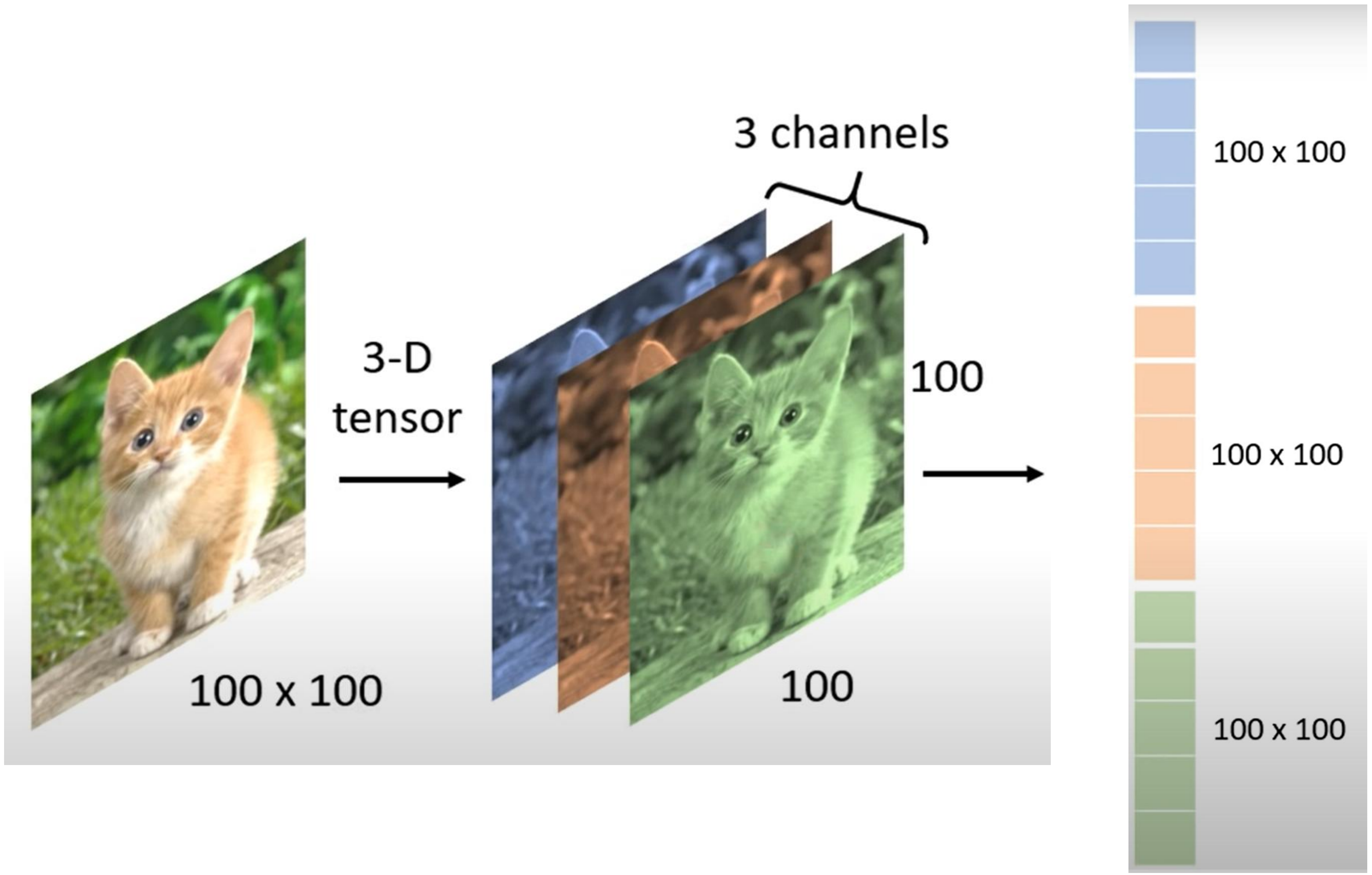
a)
3x3 kernel
w_{input}
h_{input}
d_{input}
1x
k_{c1}
$d_{c1} = k_{c1}$
2x
$h_{p1} = \frac{h_{input}}{2}$
$w_{p1} = \frac{w_{input}}{2}$
Convolution + Activation
MaxPooling
e.g. ReLU
0
b)
Input
Convolutional Backbone
Classifier Head
c)
$h_{p4} * w_{p4} * d_{c4} = n_{Flatten}$
w_{p4}
h_{p4}
d_{c4}
Weight/Parameter
$f_{ReLU}()$
Neuron
$f_{SoftMax}()$
Flatten/ Input
Hidden Layer
Output
Fully Connected ANN



0
25
50
75
100
125
150
175
200
0
50
100
150
200

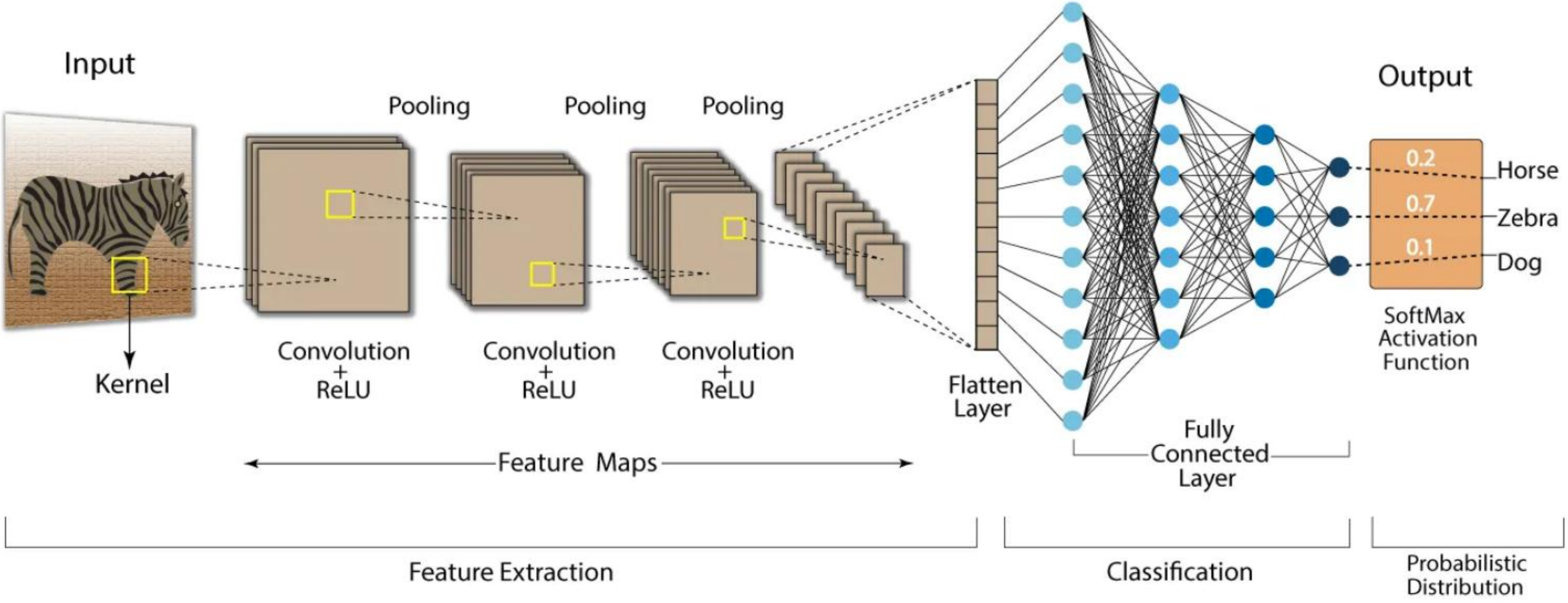


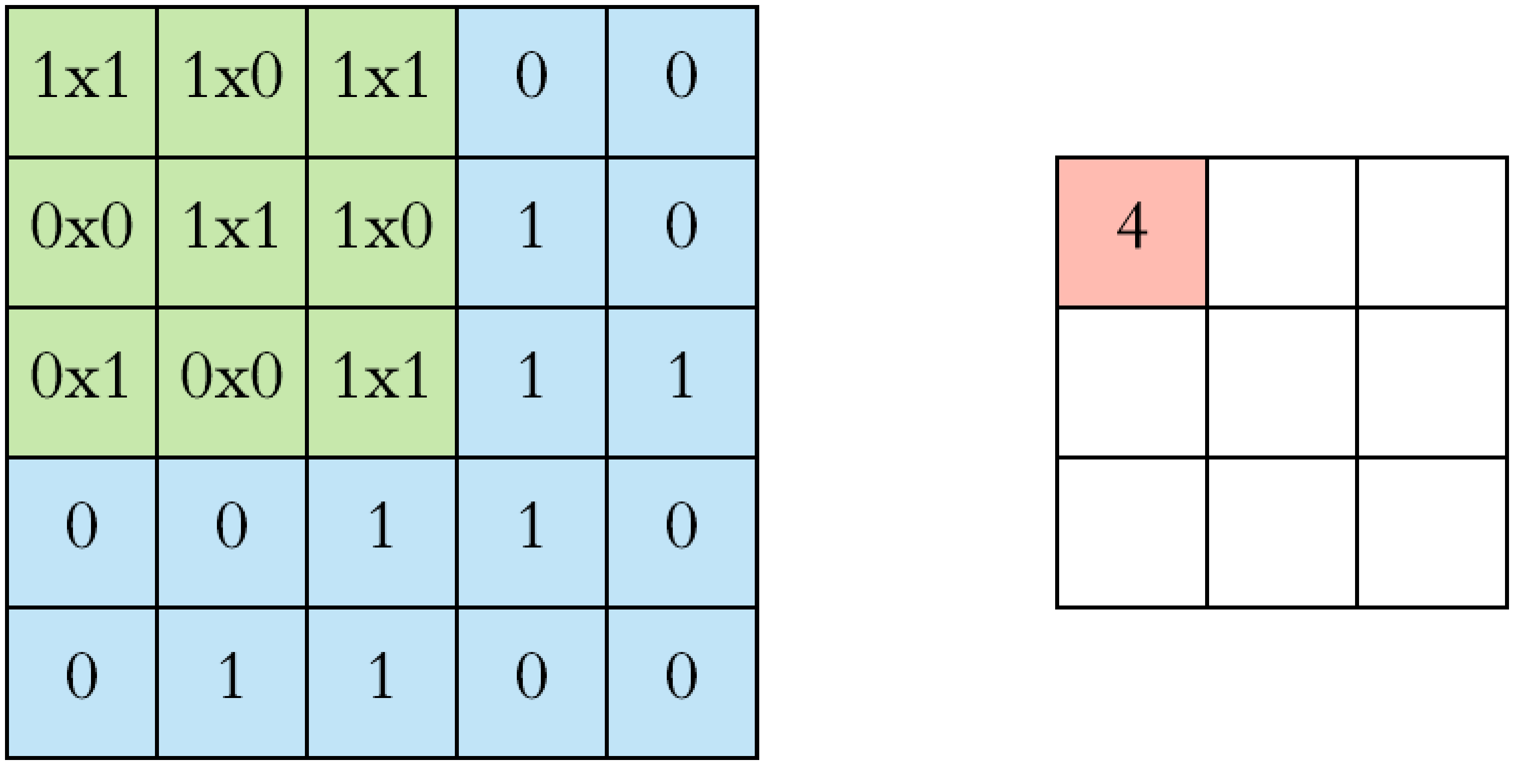
3
3
3
3D Filter/Kernel

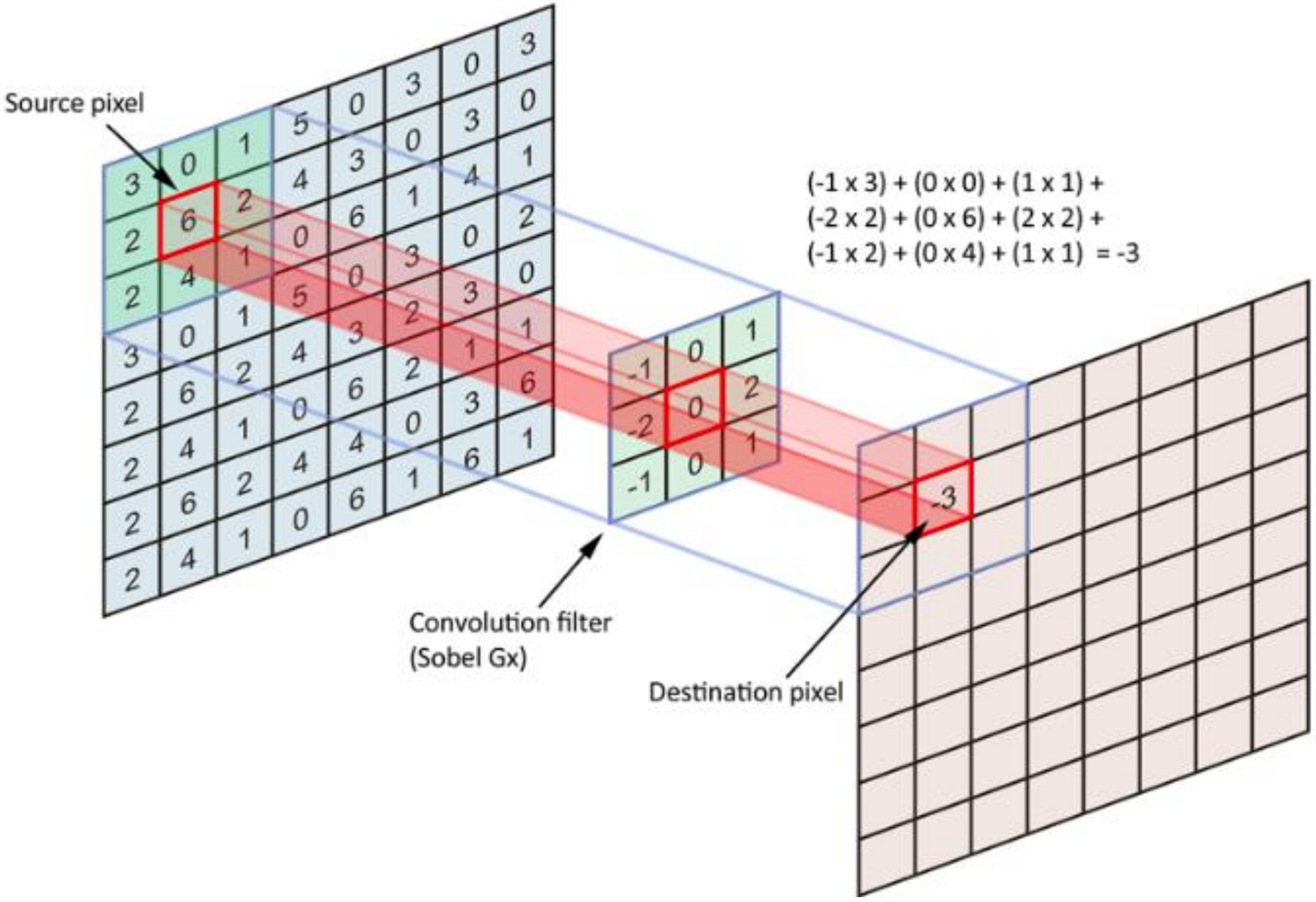


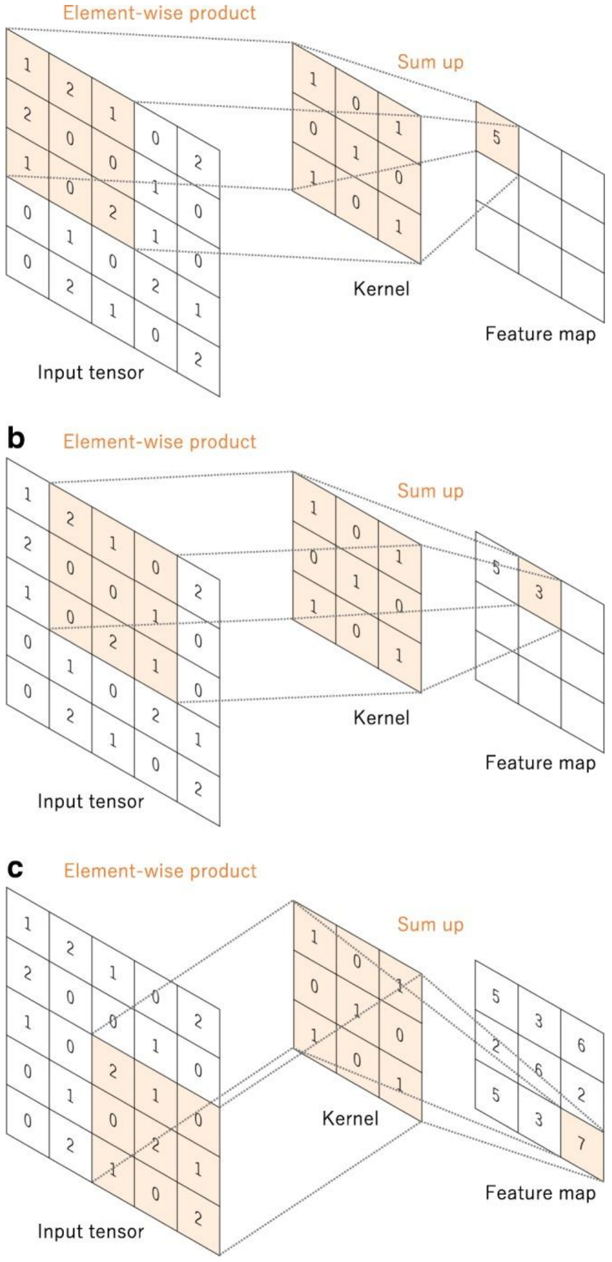
3 channels
3-D
tensor
100
100 x 100
100
100 x 100
100 x 100
100 x 100

Convolution Neural Network (CNN)




1x1
1x0
1x1
0
0
0x0
1x1
1x0
1
0
0x1
0x0
1x1
1
1
0
0
1
1
0
0
1
1
0
0
4


Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
(Sobel Gx)
Destination pixel


Element-wise product
Sum up
Kernel
Feature map
Input tensor
b
Element-wise product
Sum up
Kernel
Feature map
Input tensor
c
Element-wise product
Sum up
Kernel
Feature map
Input tensor

STRATO CONVOLUZIONALE
(FILTRI + RELU)

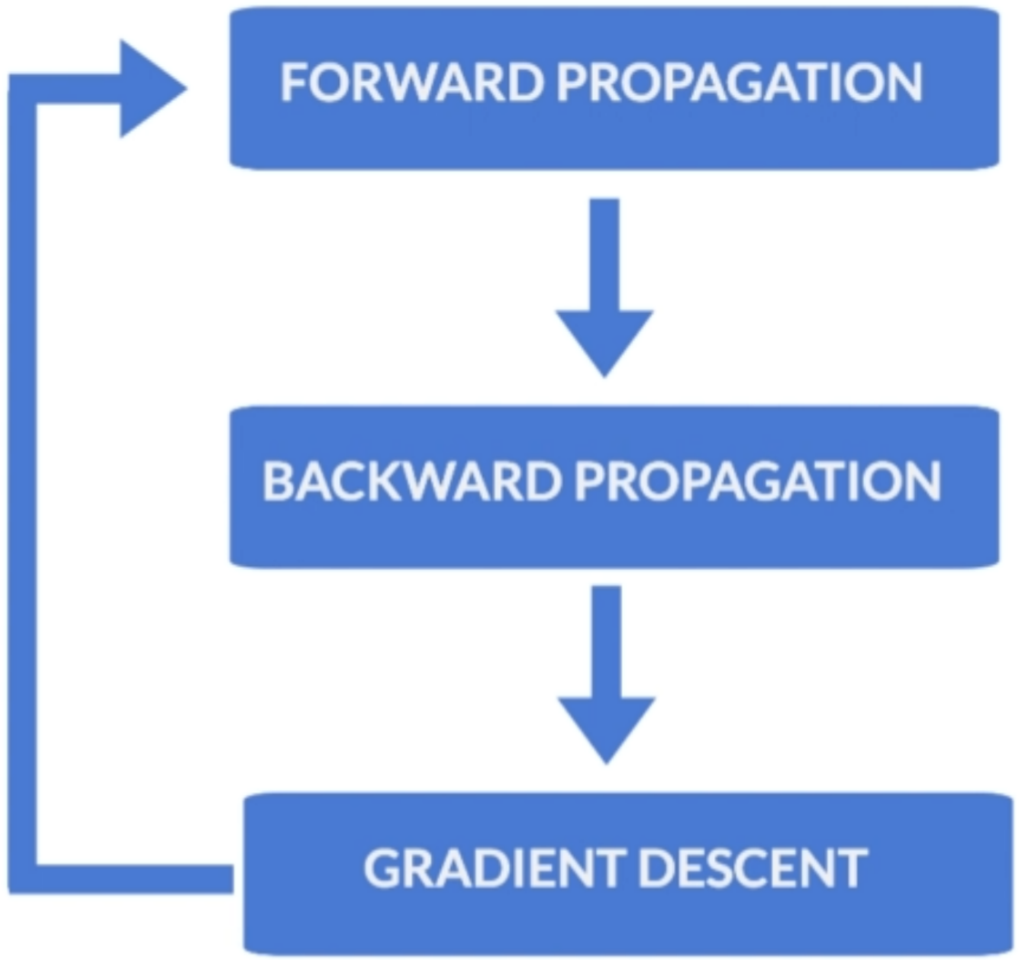
1	0	-1
1	0	-1
1	0	-1

Esistono tantissimi filtri e possiamo anche creare i nostri, come facciamo a selezionare quelli da utilizzare ?

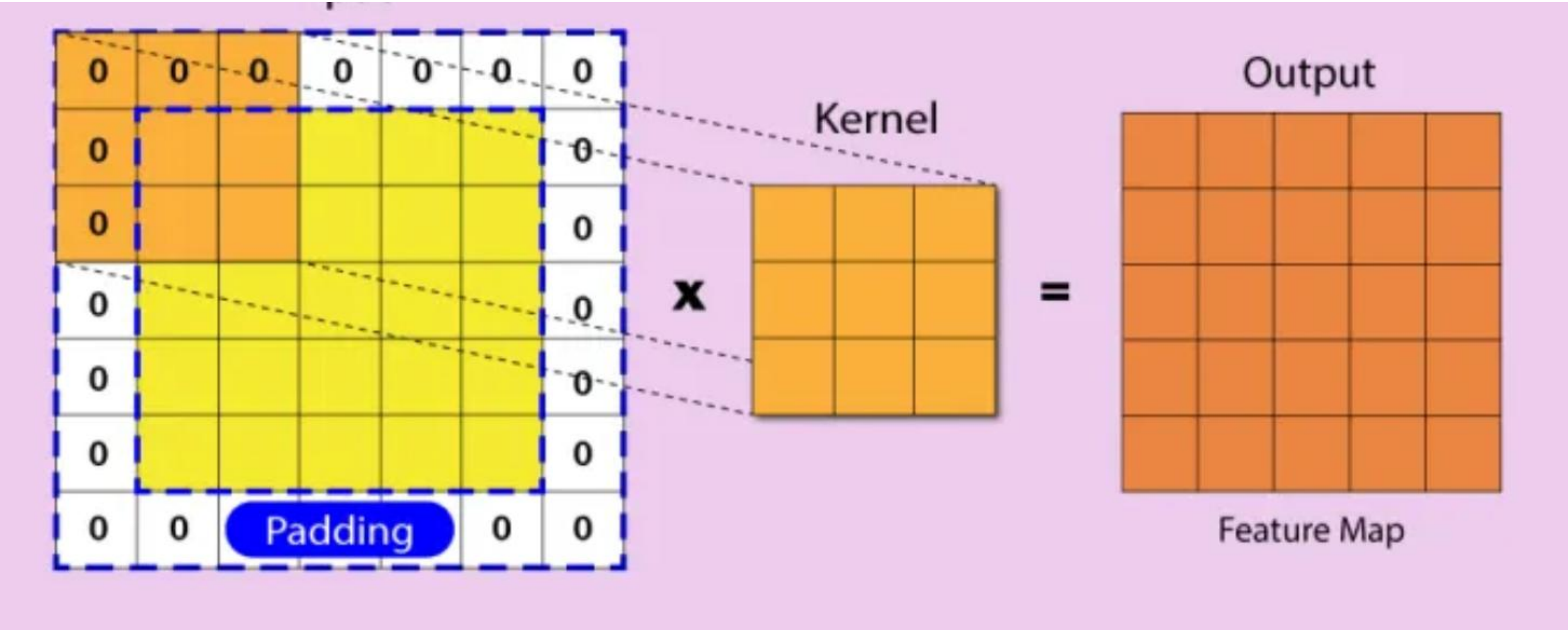
STRATO CONVOLUZIONALE
(FILTRI + RELU)

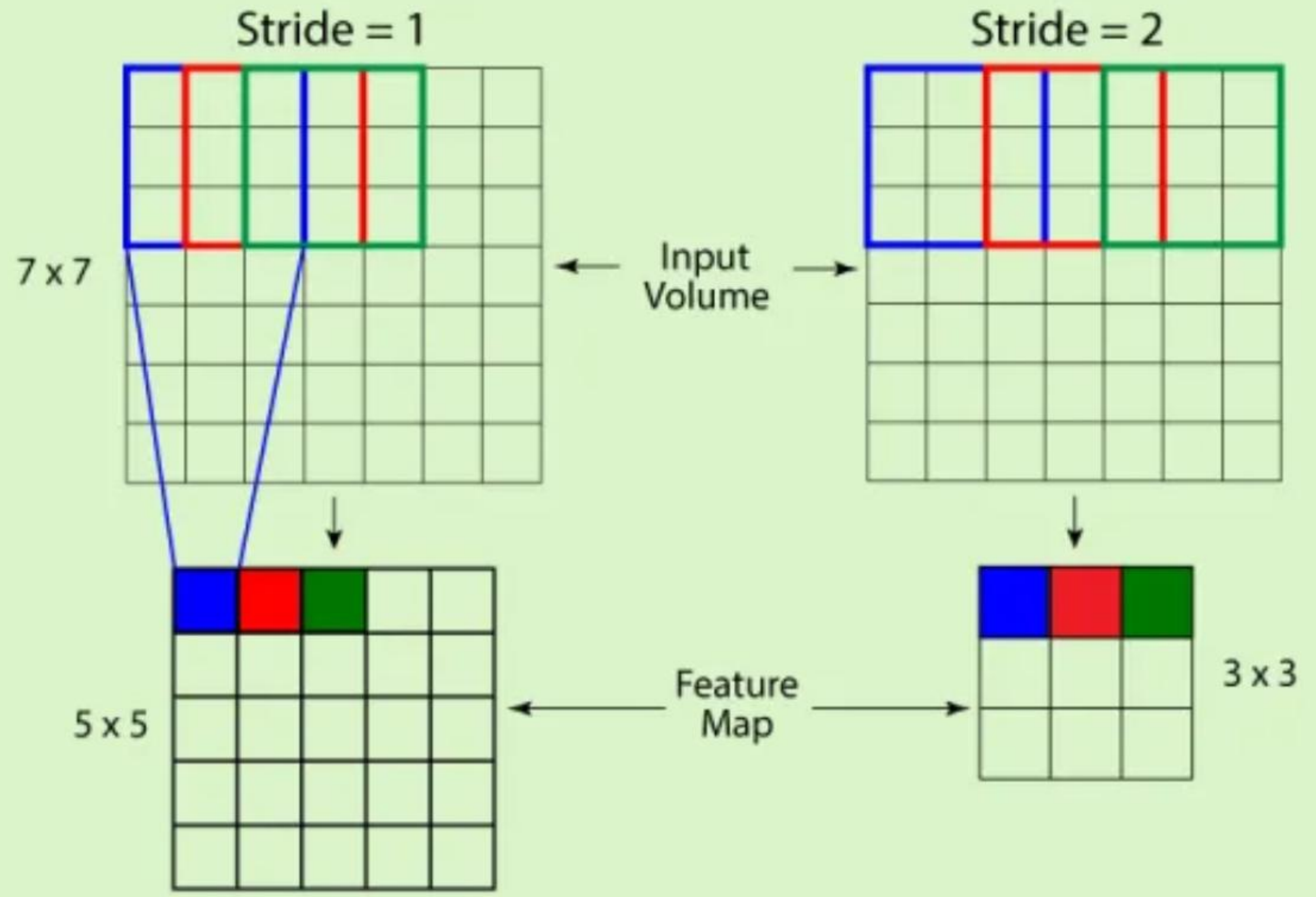
w1,1	w1,2	w1,3
w2,1	w2,2	w2,3
w3,1	w3,2	w3,3

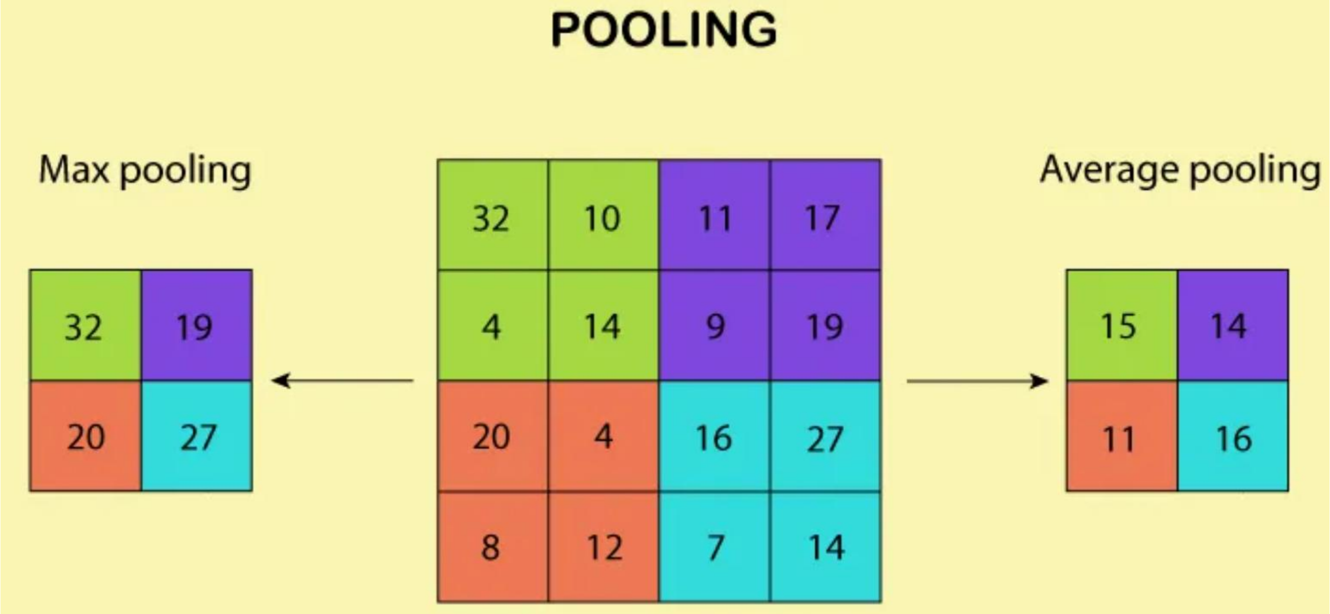
Non lo facciamo !
lasciamo che sia la rete a selezionare i filtri durante l'addestramento

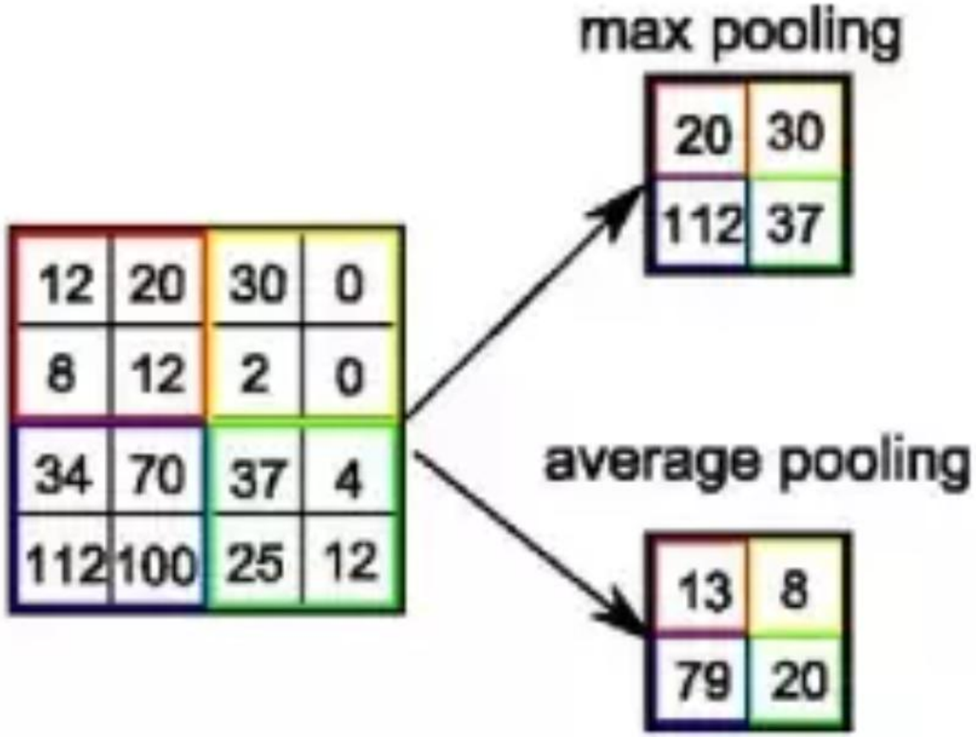


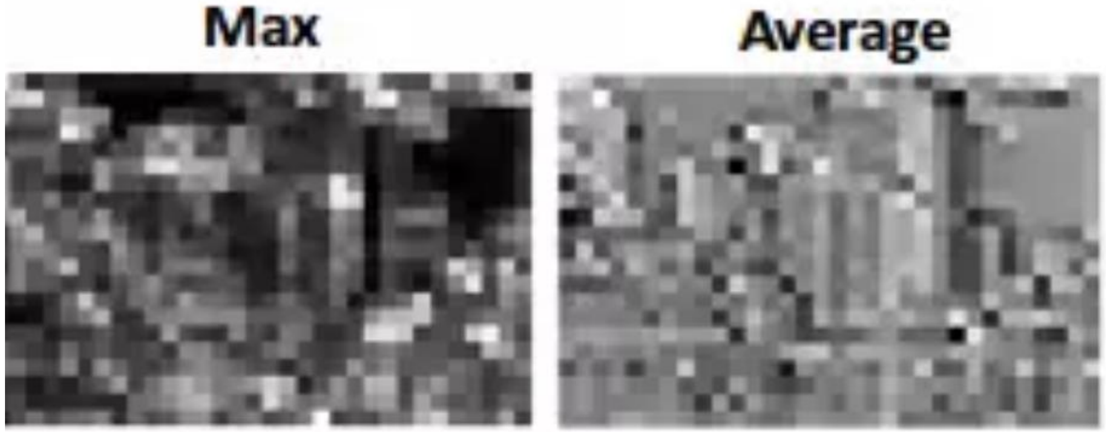
Trattando i valori dei filtri come coefficienti del modello,
il processo di addestramento diventa lo stesso di una semplice rete neurale artificiale

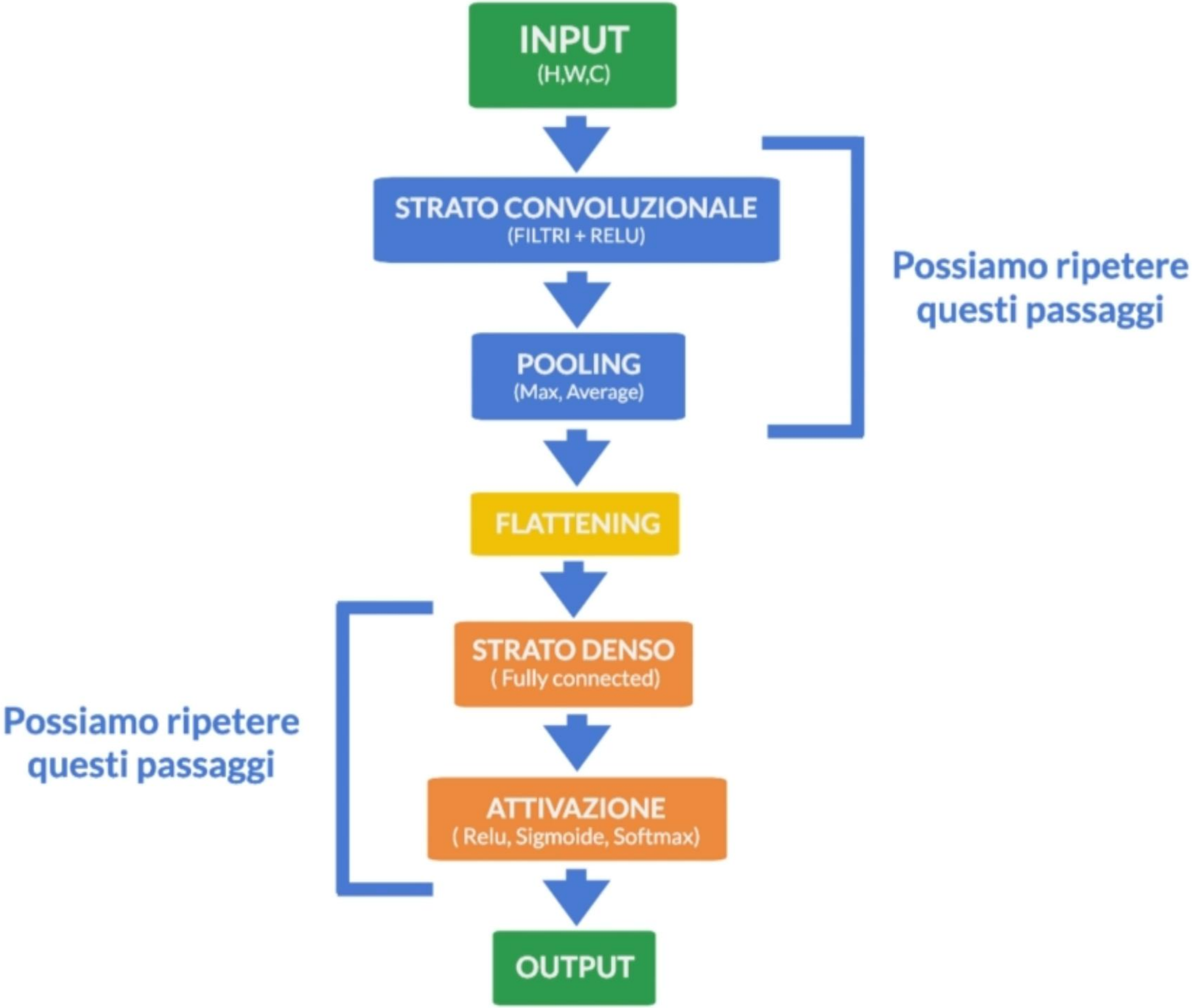

Padding
Kernel
Output
Feature Map


Stride = 1
Stride = 2
7 x 7
Input
Volume
5 x 5
Feature
Map
3 x 3


POOLING
Max pooling
Average pooling
32
10
11
17
4
14
9
19
20
4
16
27
8
12
7
14
32
19
20
27
15
14
11
16


max pooling
average pooling
12
20
30
0
8
12
2
0
34
70
37
4
112
100
25
12
20
30
112
37
13
8
79
20


Max
Average


INPUT
(H,W,C)
STRATO CONVOLUZIONALE
(FILTRI + RELU)
POOLING
(Max, Average)
Possiamo ripetere questi passaggi
FLATTENING
STRATO DENSO
(Fully connected)
ATTIVAZIONE
(Relu, Sigmoide, Softmax)
Possiamo ripetere questi passaggi
OUTPUT