Image Analysis using Machine Learning

R D Parsons

Image credit ESO

Image Recognition using Machine Learning

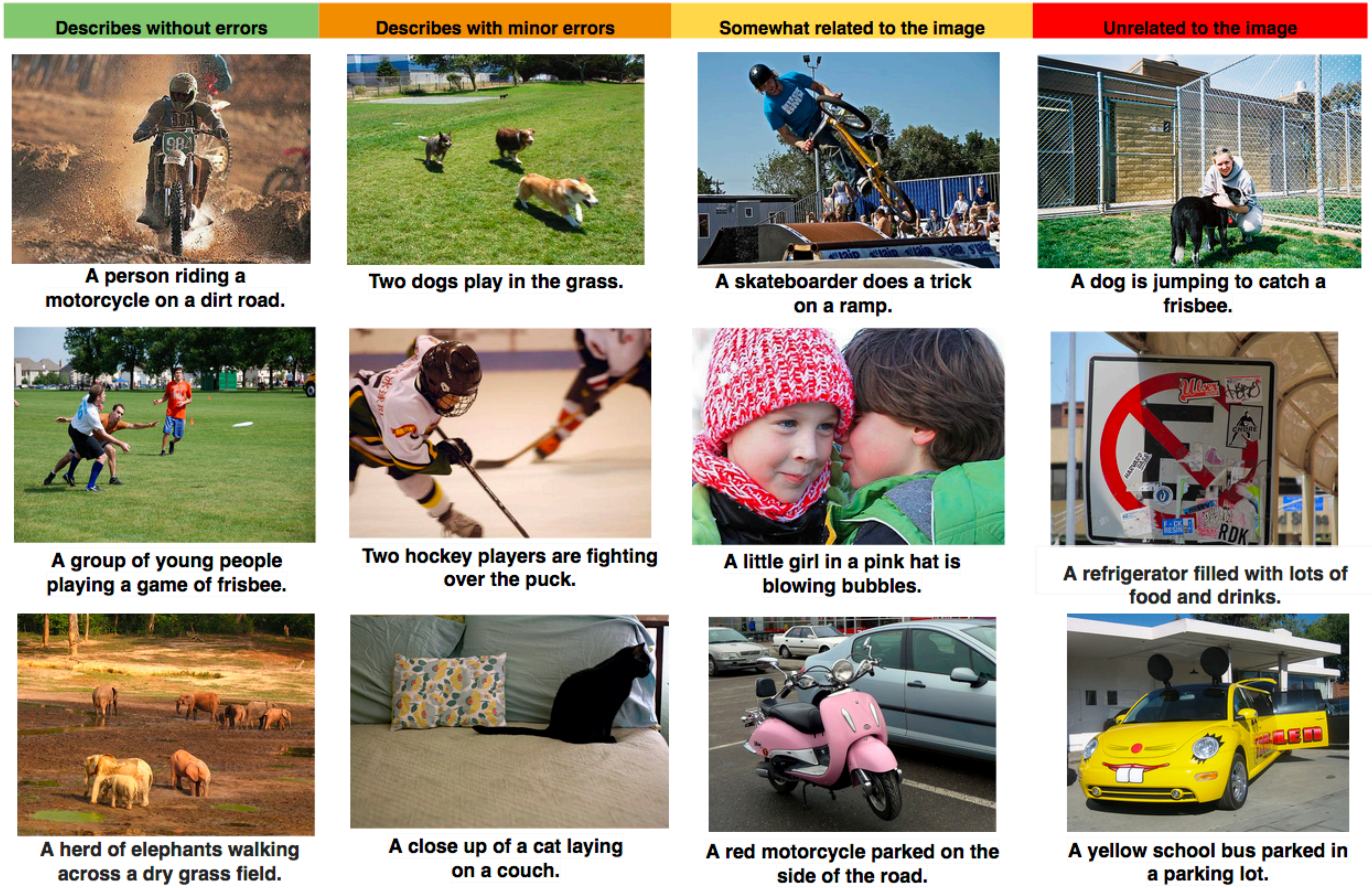
Image recognition and analysis using machine learning has come a long way in the last 10 years

This is a huge topic with many methods and fields

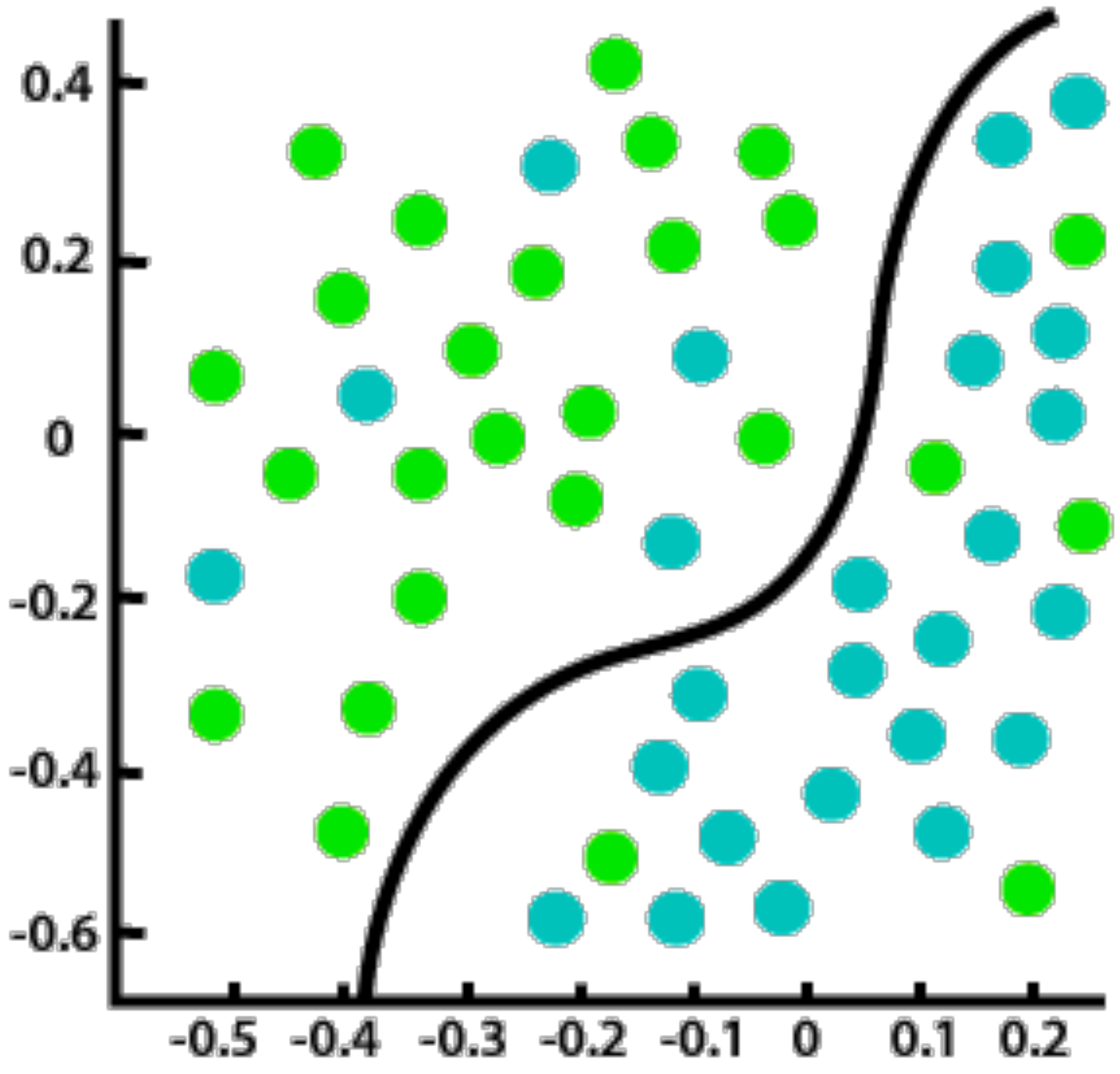
We will concentrate on the use of **neural networks**

Making something like on the right is clearly a huge task

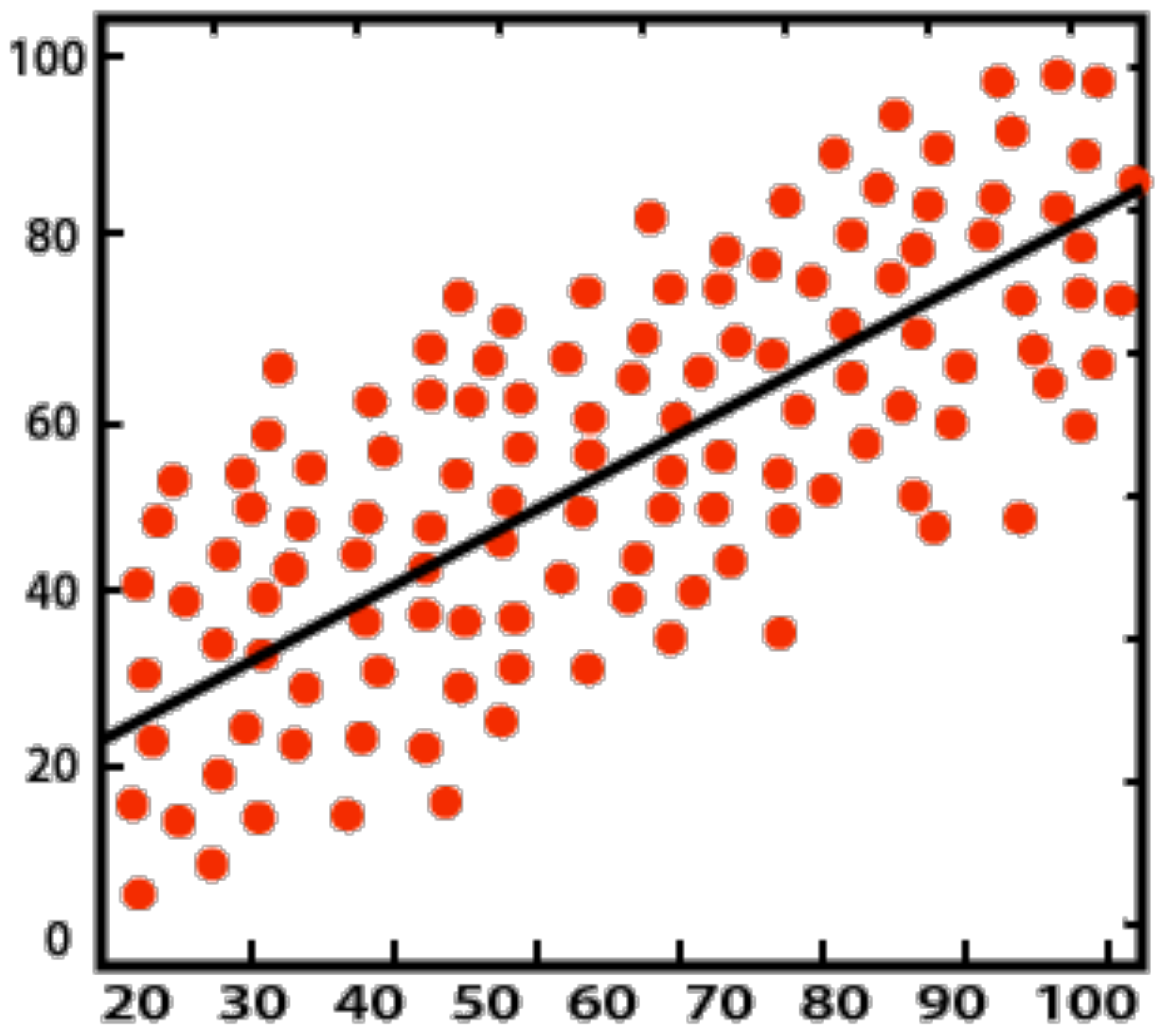
Let’s try to go from “zero” to close to cutting edge letter recognition in an hour…



Classifiers vs Regressors



Classification



Regression

Classifiers vs Regressors

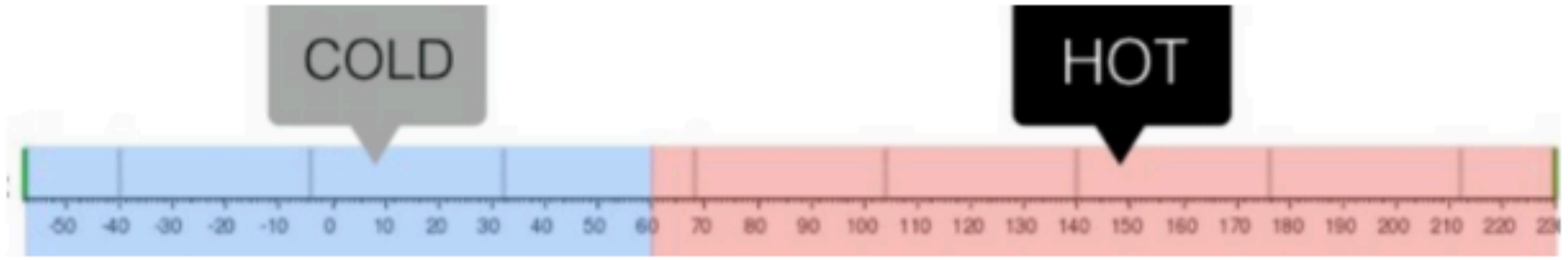
Regression

What will be the temperature tomorrow?

Fahrenheit

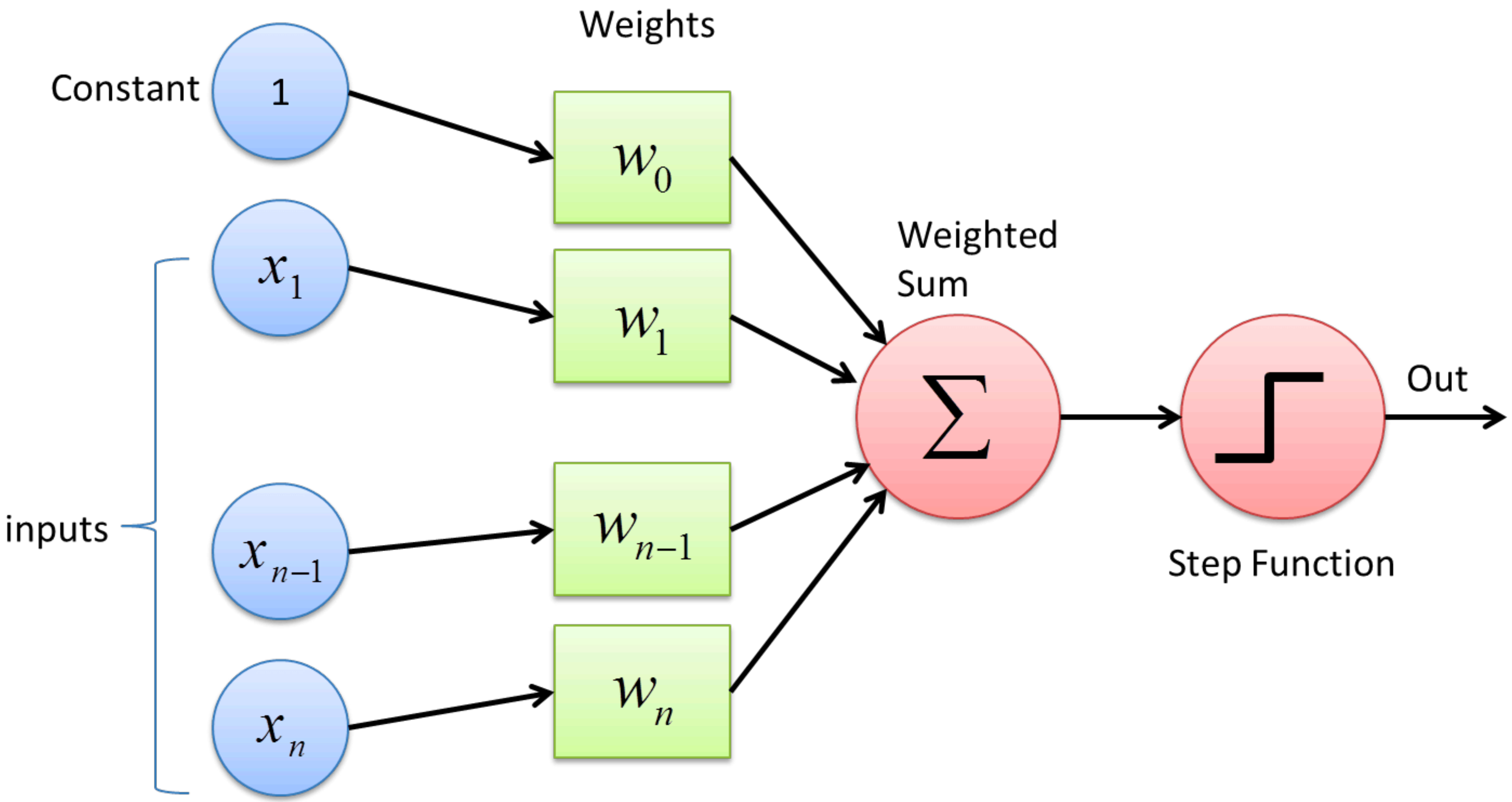
Classification

Will it be hot or cold tomorrow?



Fahrenheit

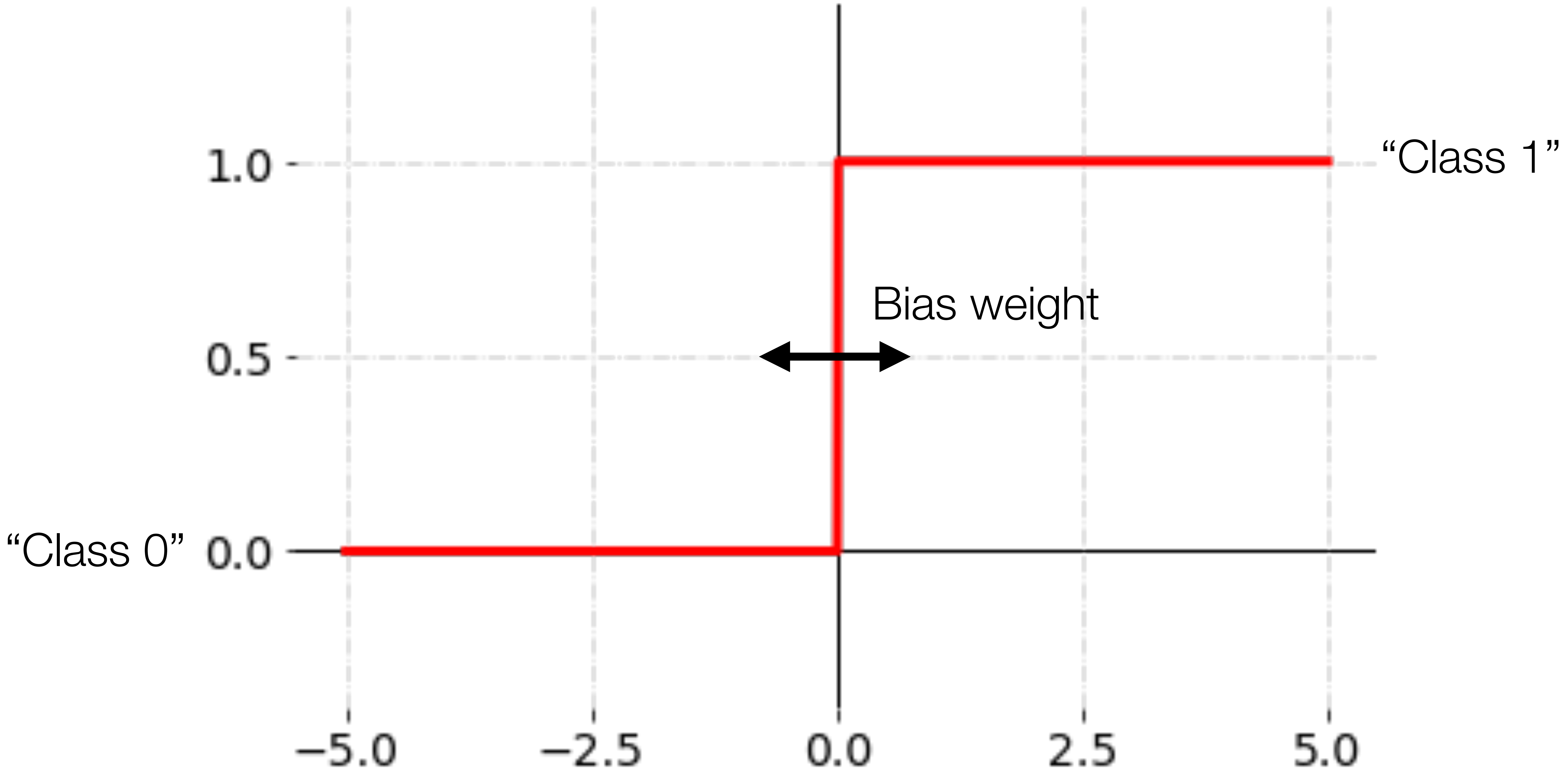
The Perceptron



https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53

The Perceptron

Activation by step function
If weighted sum larger than a value returns 1

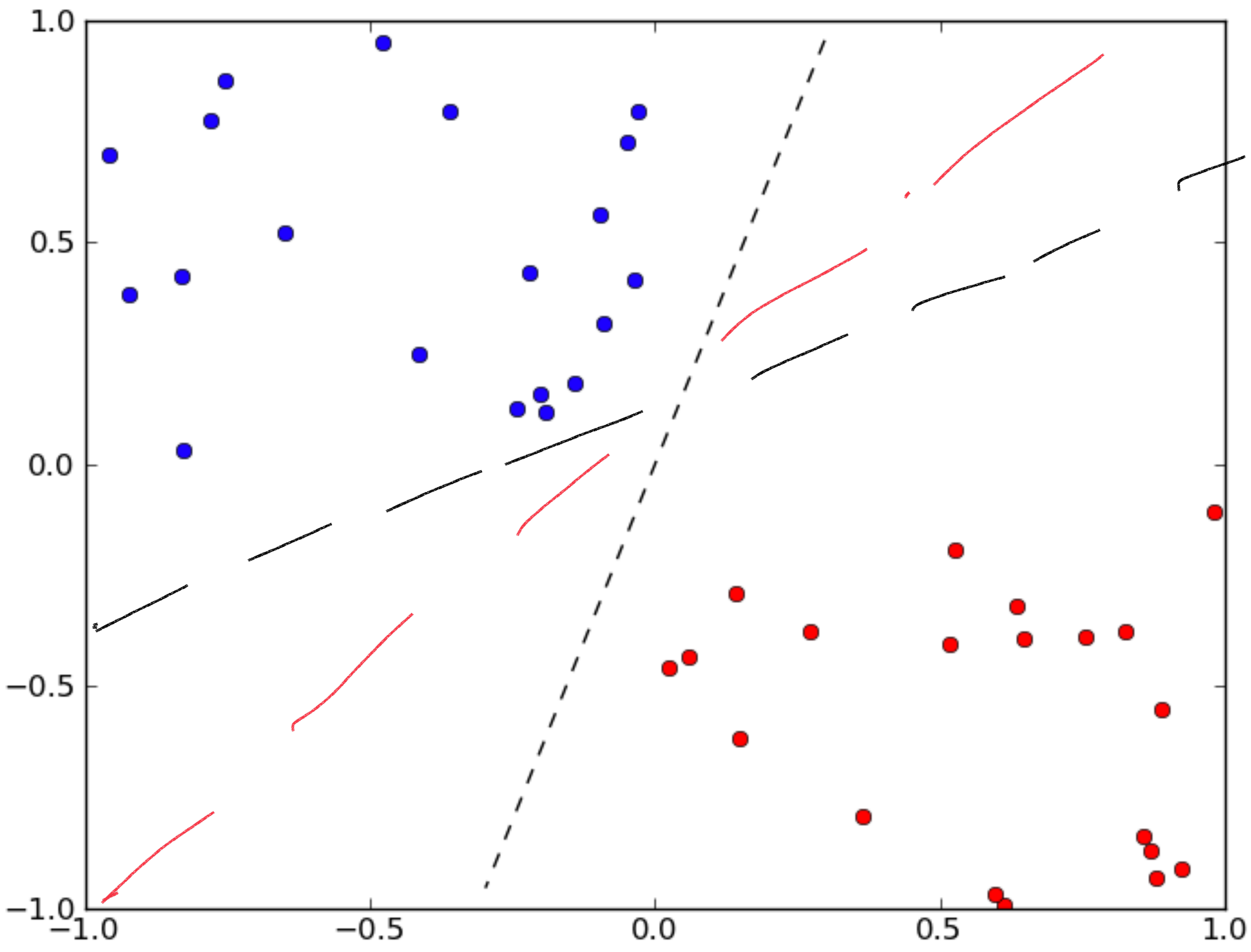


The Perceptron

The perceptron can only provide a linear separation between our two parameters

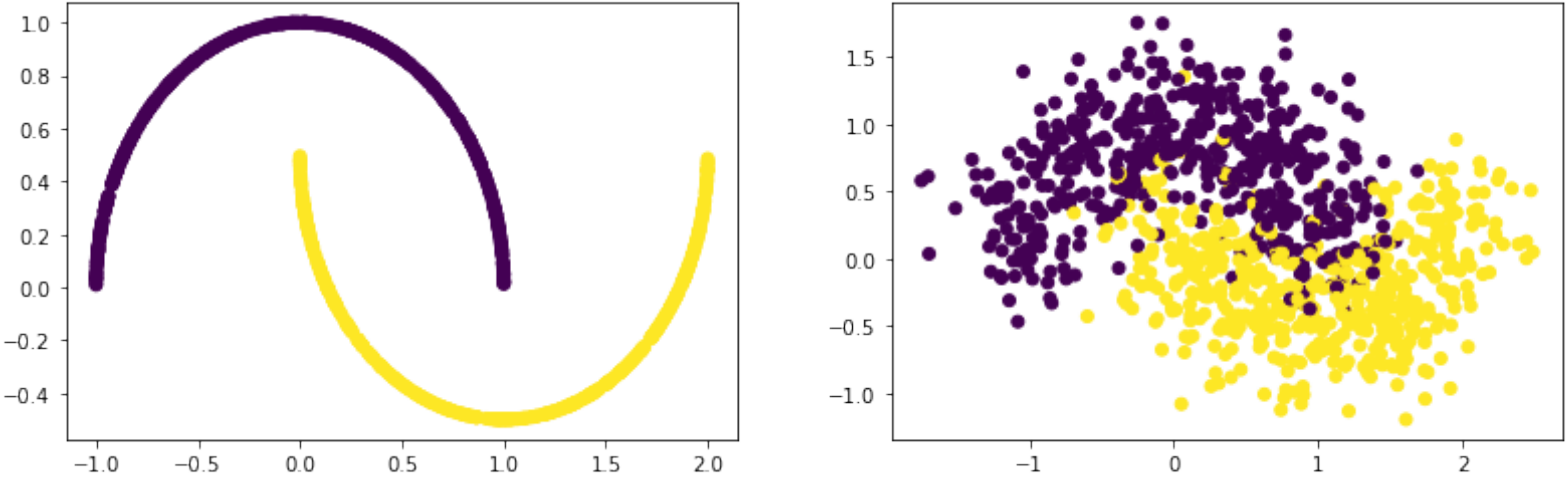
In the case on the right here we have many equal solutions

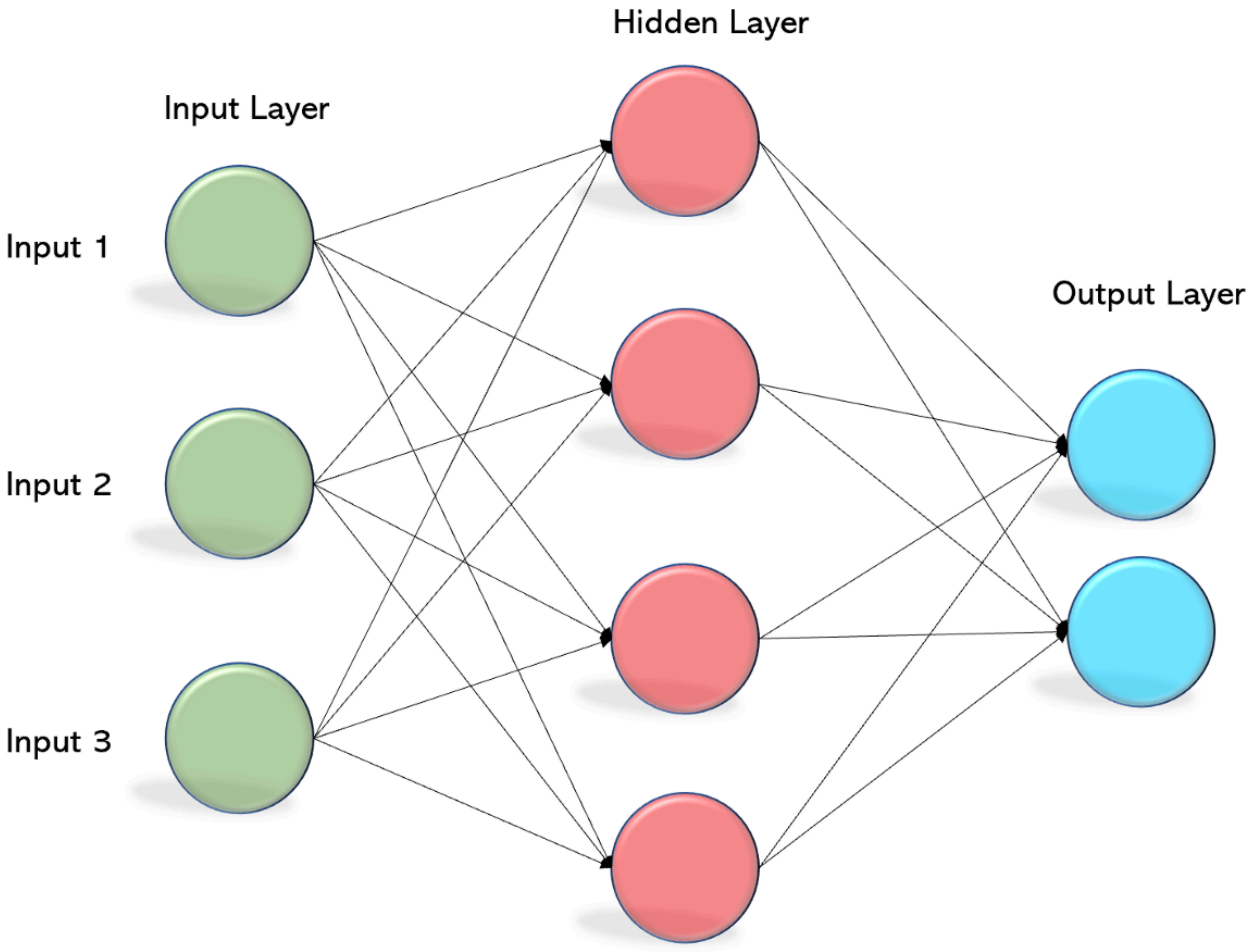
In the case of a dataset which is not linearly separable this will not work



https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53

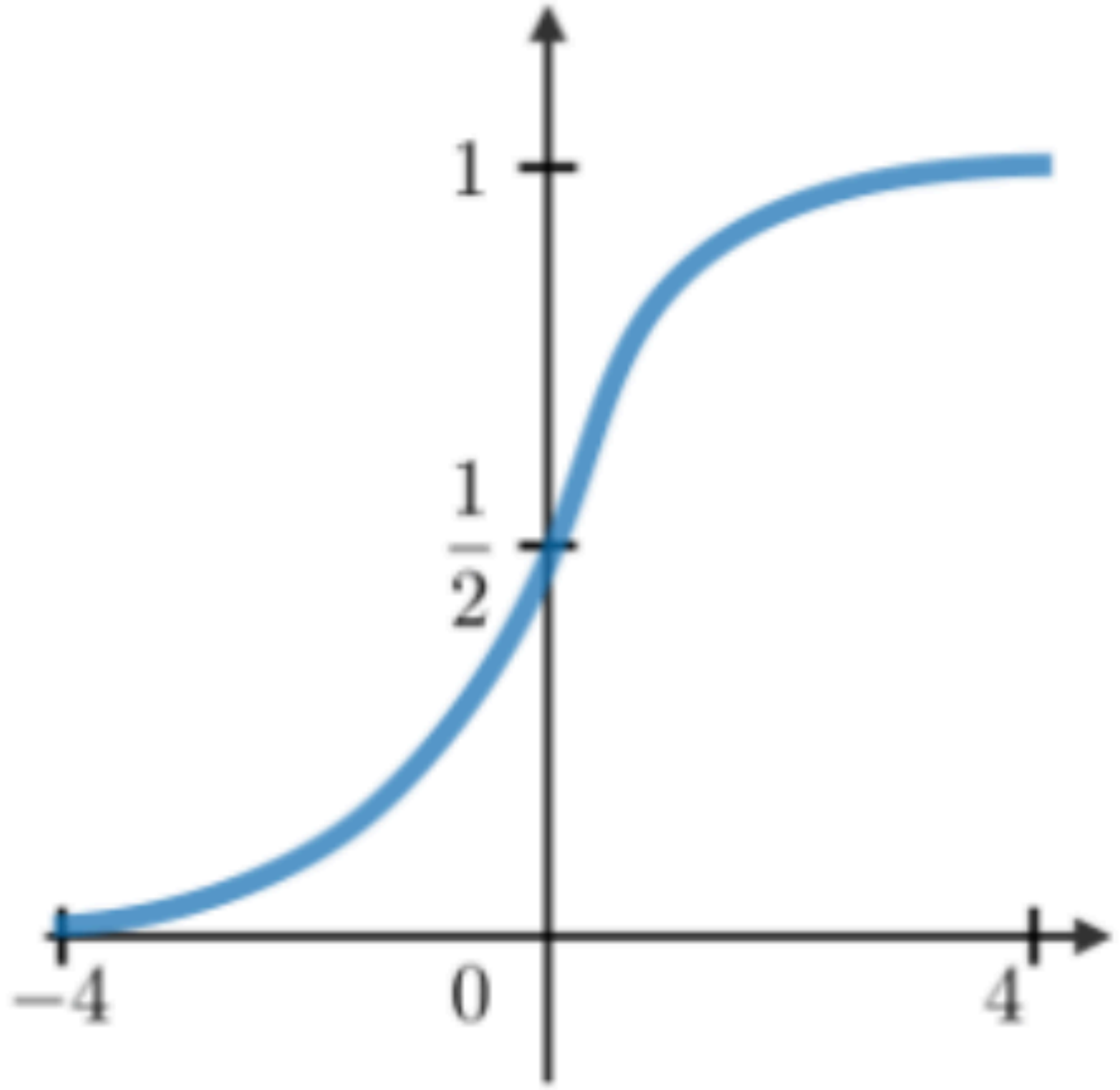
Let’s get more complicated

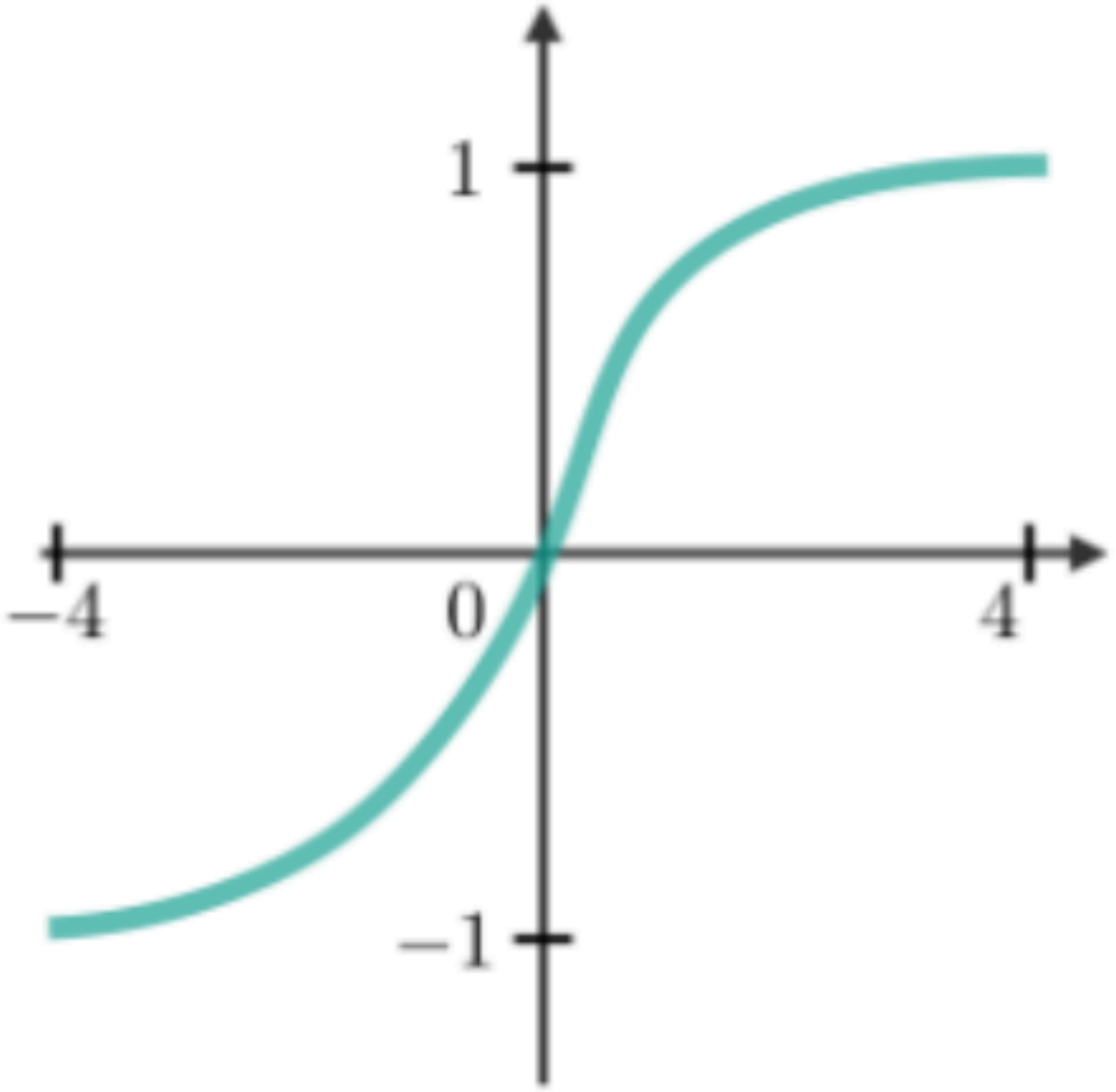


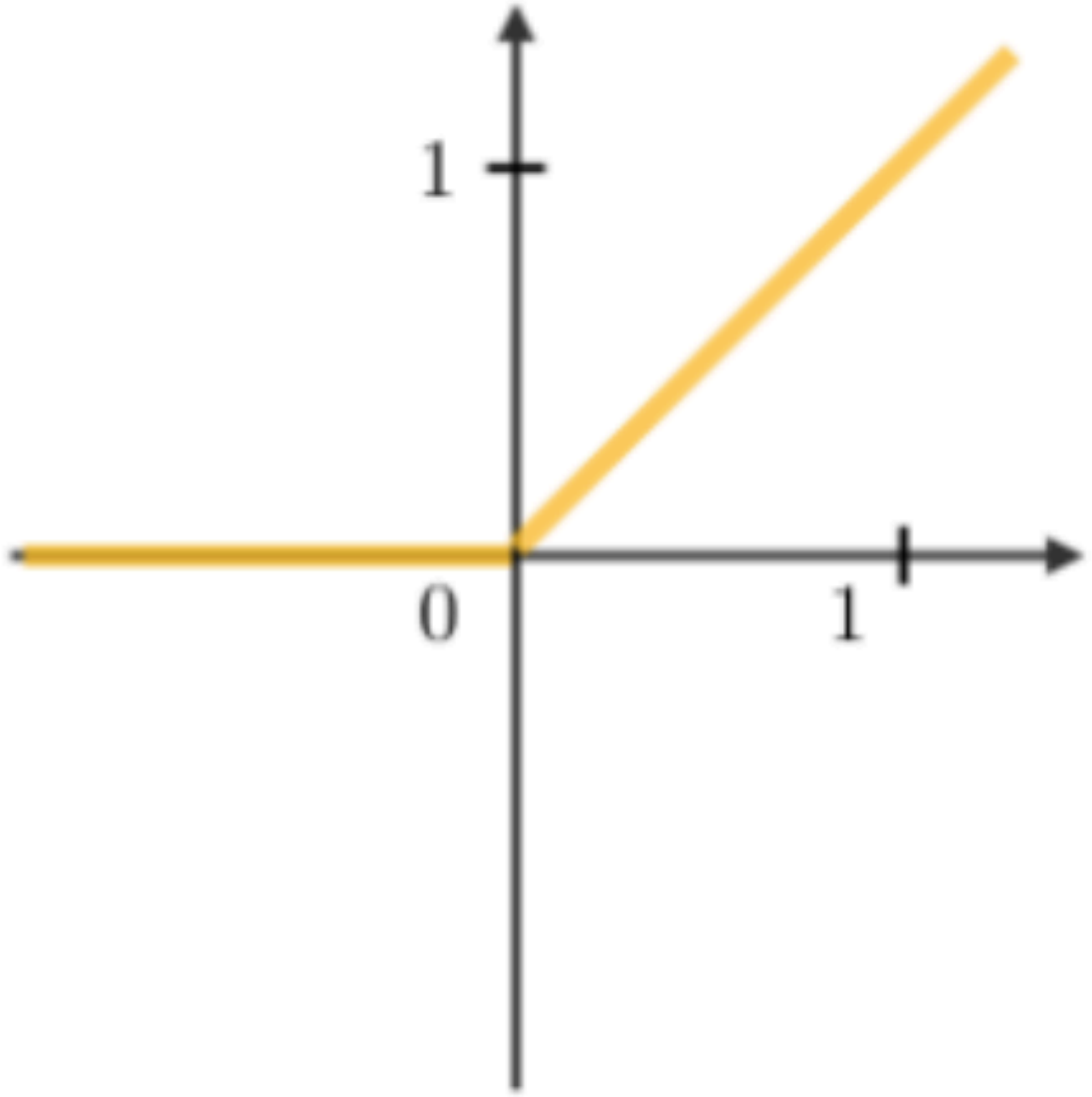

Hidden Layer
Input Layer
Input 1
Output Layer
Input 2
Input 3

Activation Functions

Sigmoid	Tanh	ReLU
$g(z) = \frac{1}{1+e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$







Activation Functions (output layer)

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K.$$

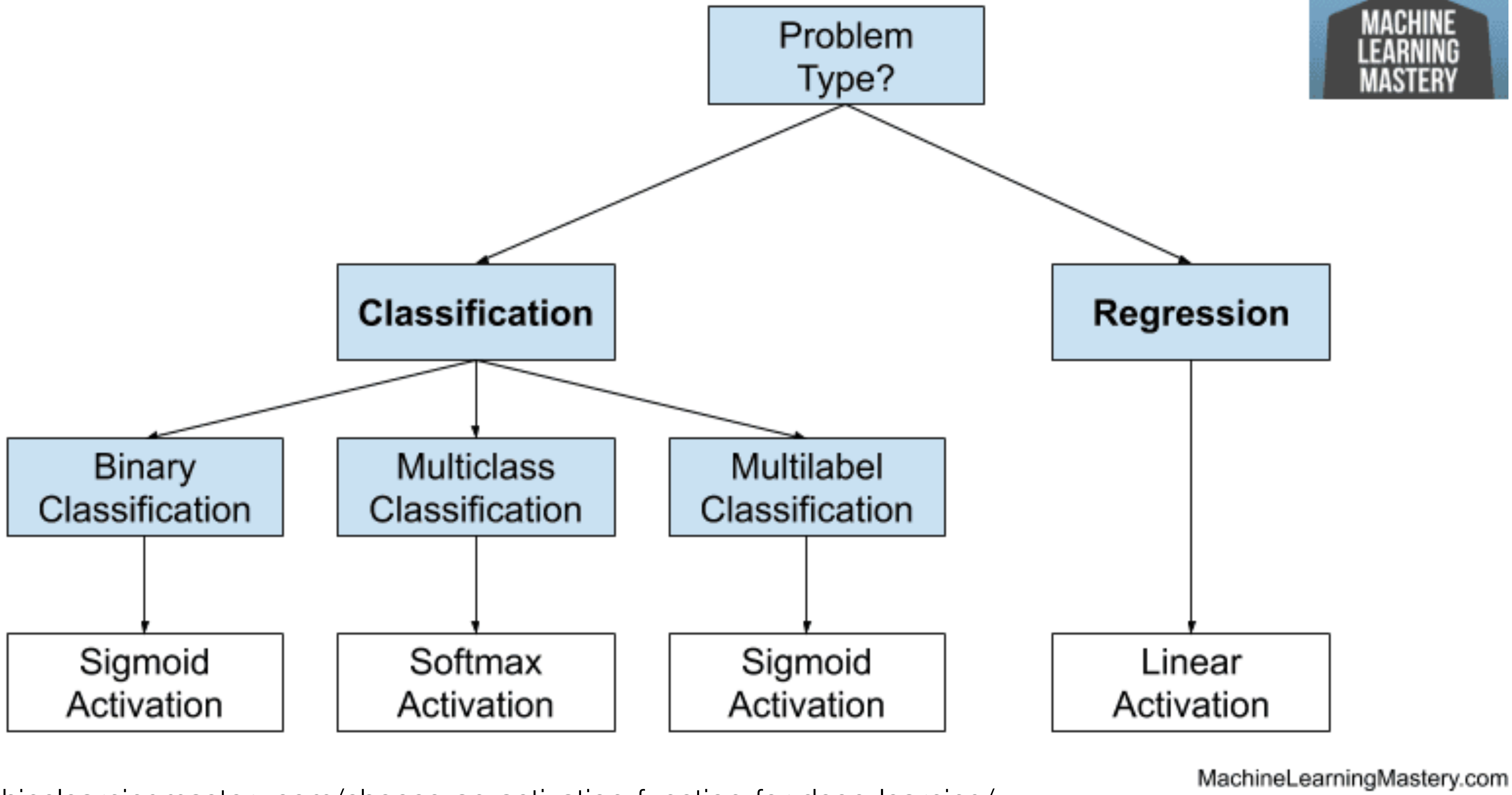
The output layer is a special case for activation and requires a different activation

For classification typically a “SoftMax” function is used

Normalised exponential, can roughly be interpreted as a probability

For regression is make sense to use a linear output function

How to Choose an Output Layer Activation Function



https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/

Loss Functions

$$\text{Loss} = -\sum_{i=1}^{\substack{\text{output}\\\text{size}}} y_i \cdot \log \hat{y}_i$$

Categorical cross-entropy
A measure of how distinguishable discrete probability distributions are
Sum over all categories and events
Works when we have multiple potential classes
Special case of “binary cross entropy” for two classes

Mean squared error
In the case that we are reproducing a continuous variable cross-entropy no longer
MSE is the usual loss function in this case
Loss prioritises outliers (other options mean absolute error, log error)

Ultimately your loss can be whatever you want just needs to get smaller with fit quality, should be fast to calculate and well behaved

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{y}_i)^2$$

Training our Network

Once we have a defined network architecture, with activation functions and a loss function we must train our network

Typically this involves splitting your data into three independent datasets
Training data
Validation data
Testing data

Training data is then fed through the network (initialised with random weights) in batches and the loss calculated for each batch of data

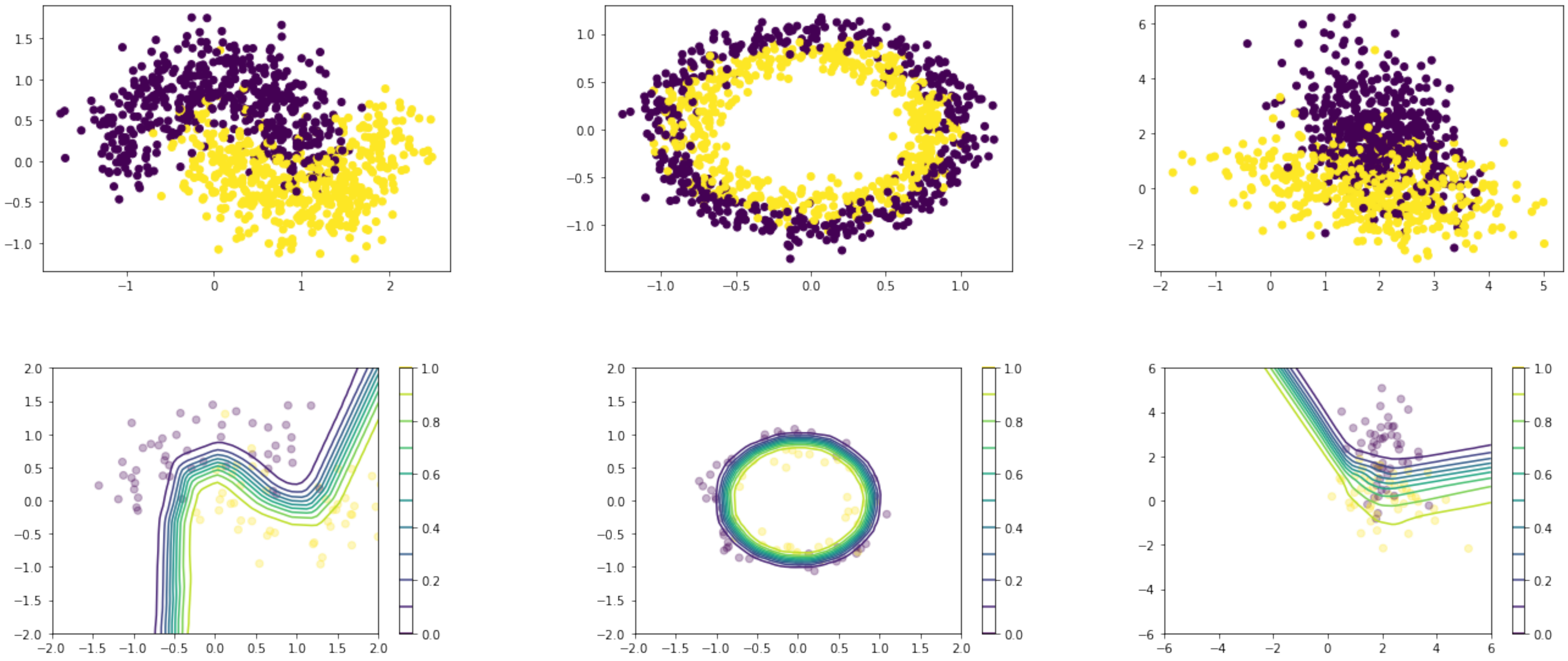
This then becomes an optimisation problem, find the network weight values that produce the smallest loss

Backpropagation algorithm allows computation of partial derivatives of the loss function for our weights

Feed the batches through our network multiple times (learning epochs) to find best performance

Learning rate controls the step size of our training

Neural Networks



keras.io

https://keras.io/

Keras

Simple. Flexible. Powerful.

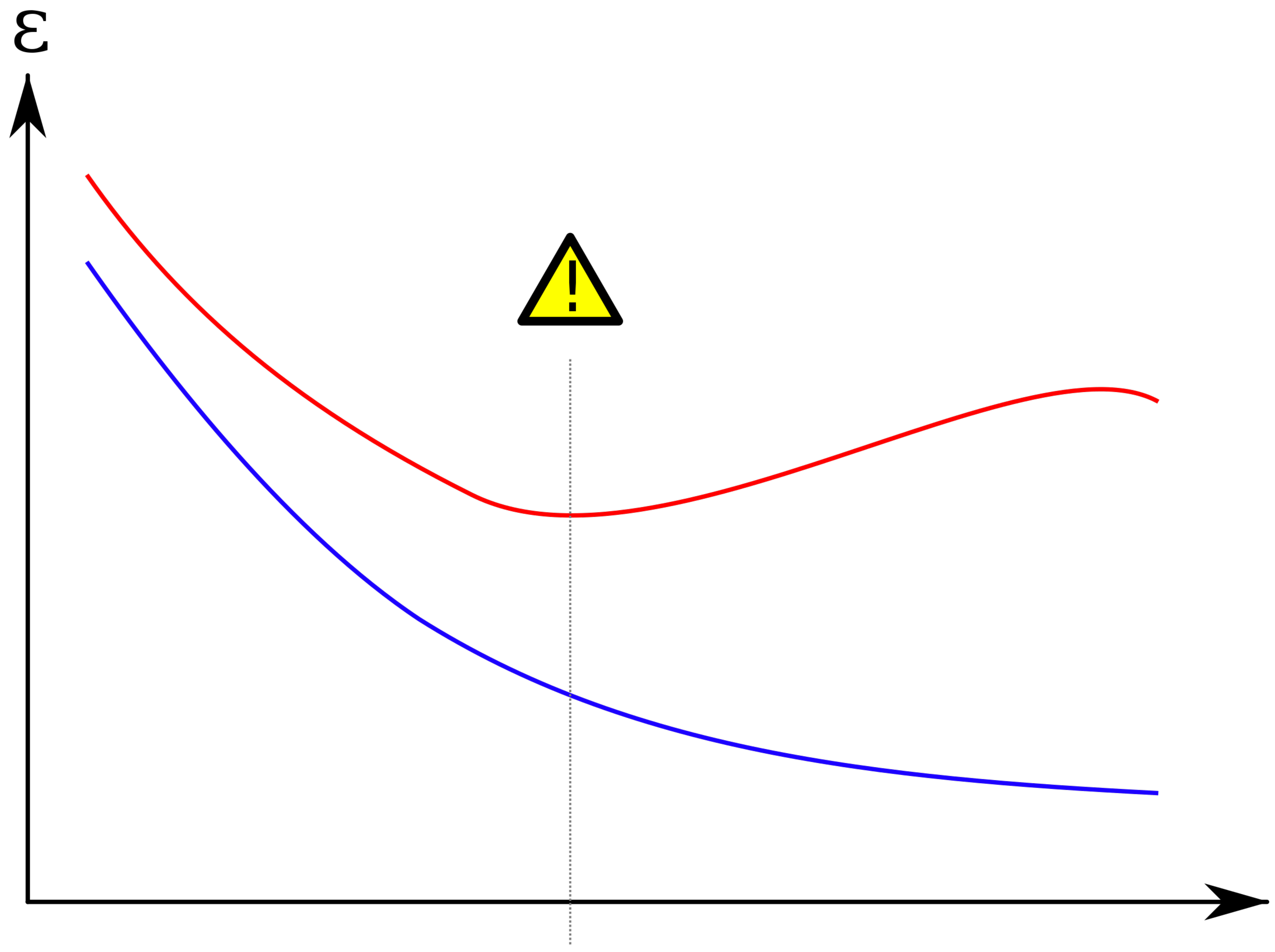
Get started | API docs | Guides | Examples

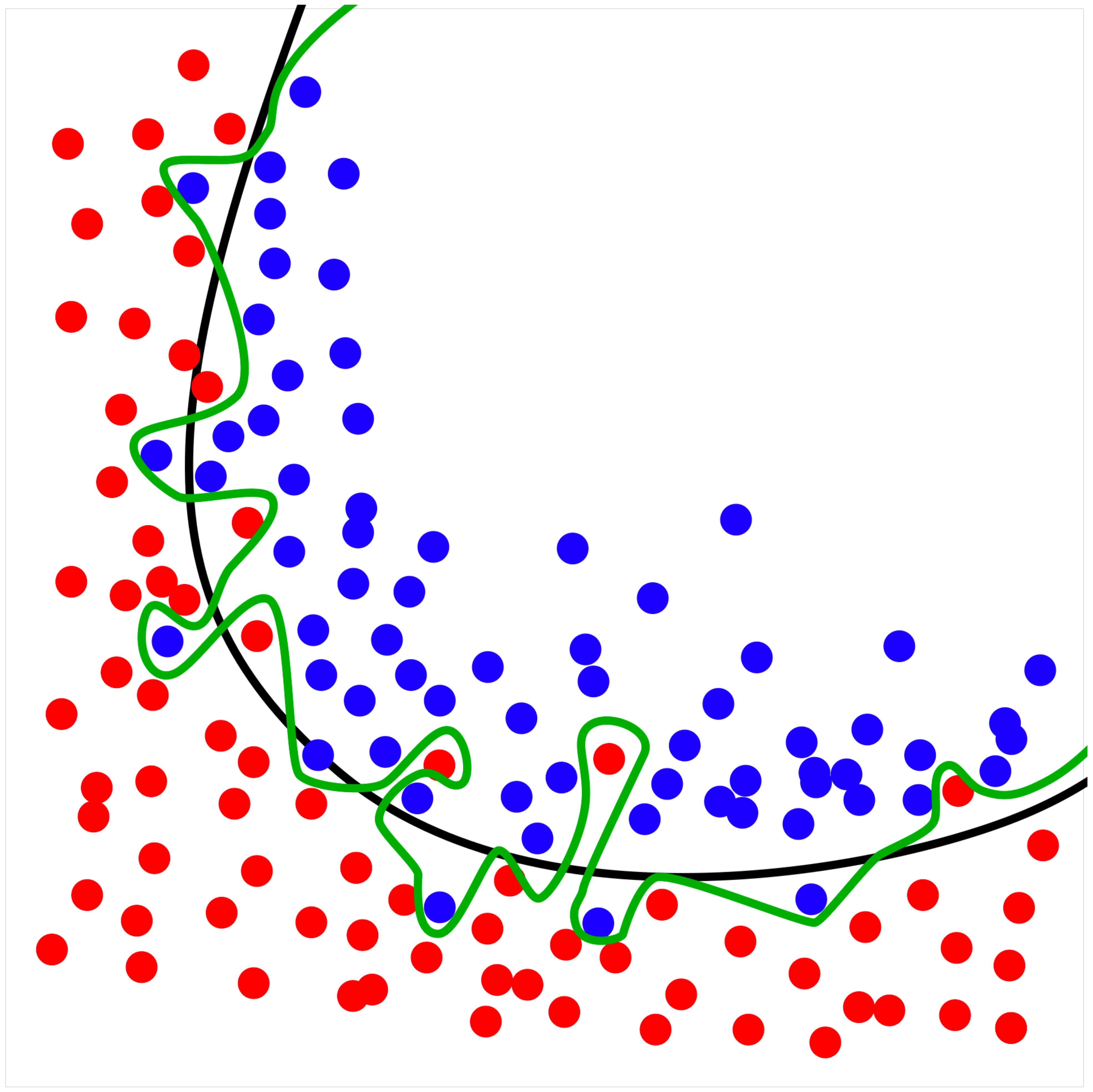
```python
from tensorflow import keras
from tensorflow.keras import layers

# Instantiate a trained vision model
vision_model = keras.applications.ResNet50()

# This is our video.encoding branch using the trained vision_model
video_input = keras.Input(shape=(100, None, None, 3))
encoded_frame_sequence = layers.TimeDistributed(vision_model)(video_input)
encoded_video = layers.LSTM(256)(encoded_frame_sequence)

# This is our text-processing branch for the question input
question_input = keras.Input(shape=(100,), dtype='int32')
embedded_question = layers.Embedding(10000, 256)(question_input)
encoded_question = layers.LSTM(256)(embedded_question)

# And this is our video question answering model:
merged = keras.layers.concatenate([encoded_video, encoded_question])
output = keras.layers.Dense(1000, activation='softmax')(merged)
video_qa_model = keras.Model(inputs=[video_input, question_input],
                             outputs=output)
```

Deep learning for humans.

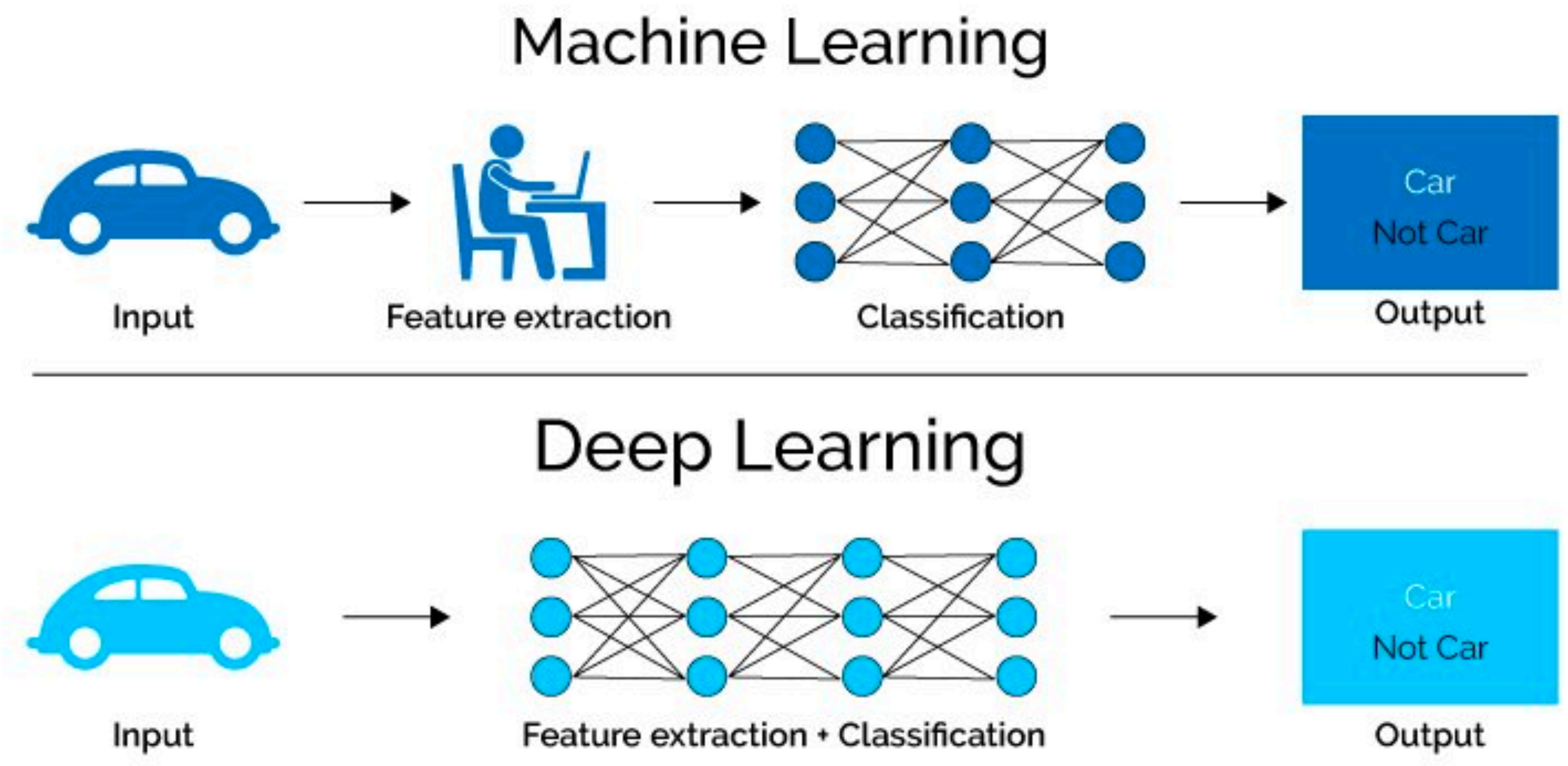
Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. It also has extensive documentation and developer guides.

Overfitting



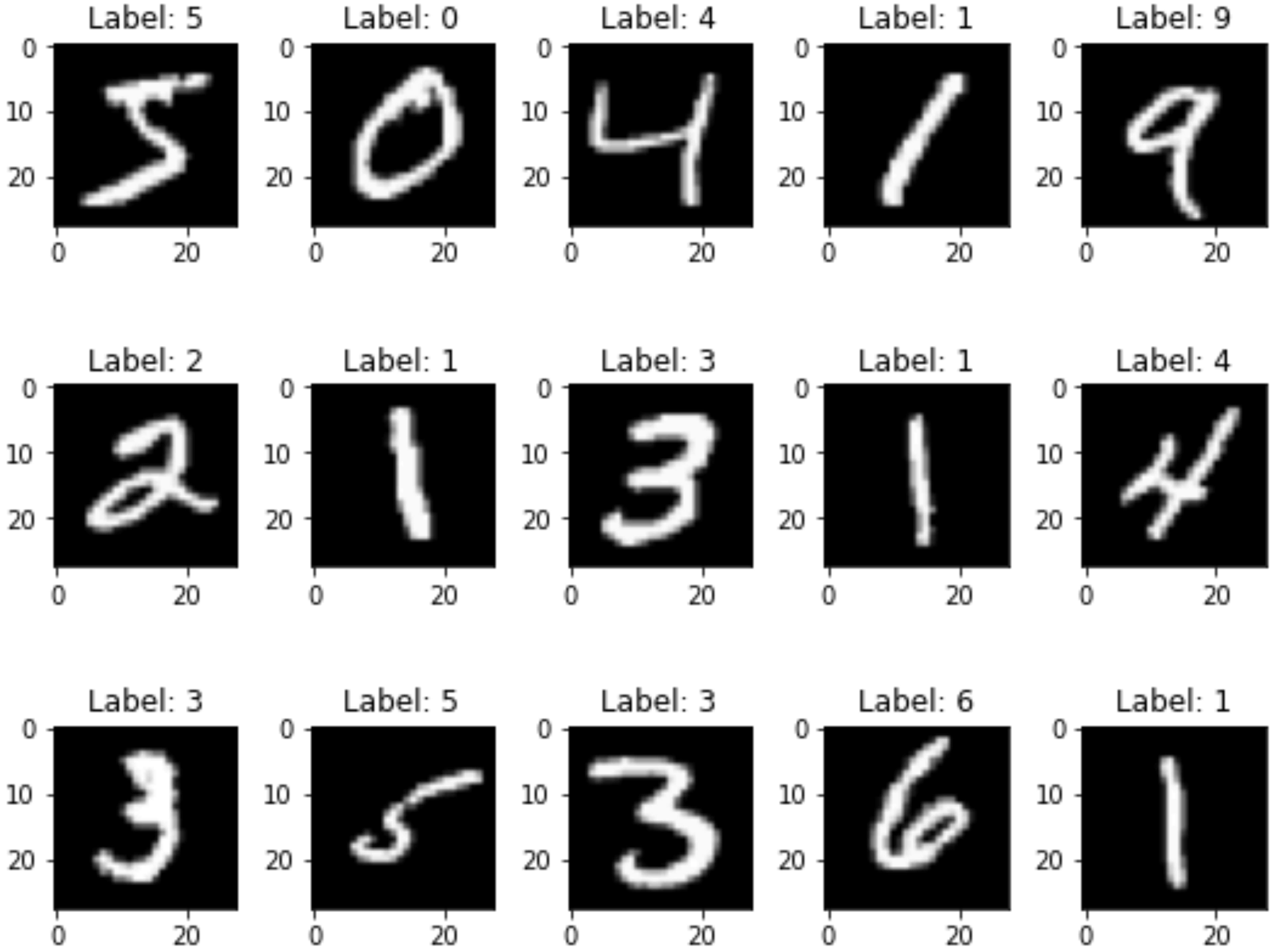


Deep Learning

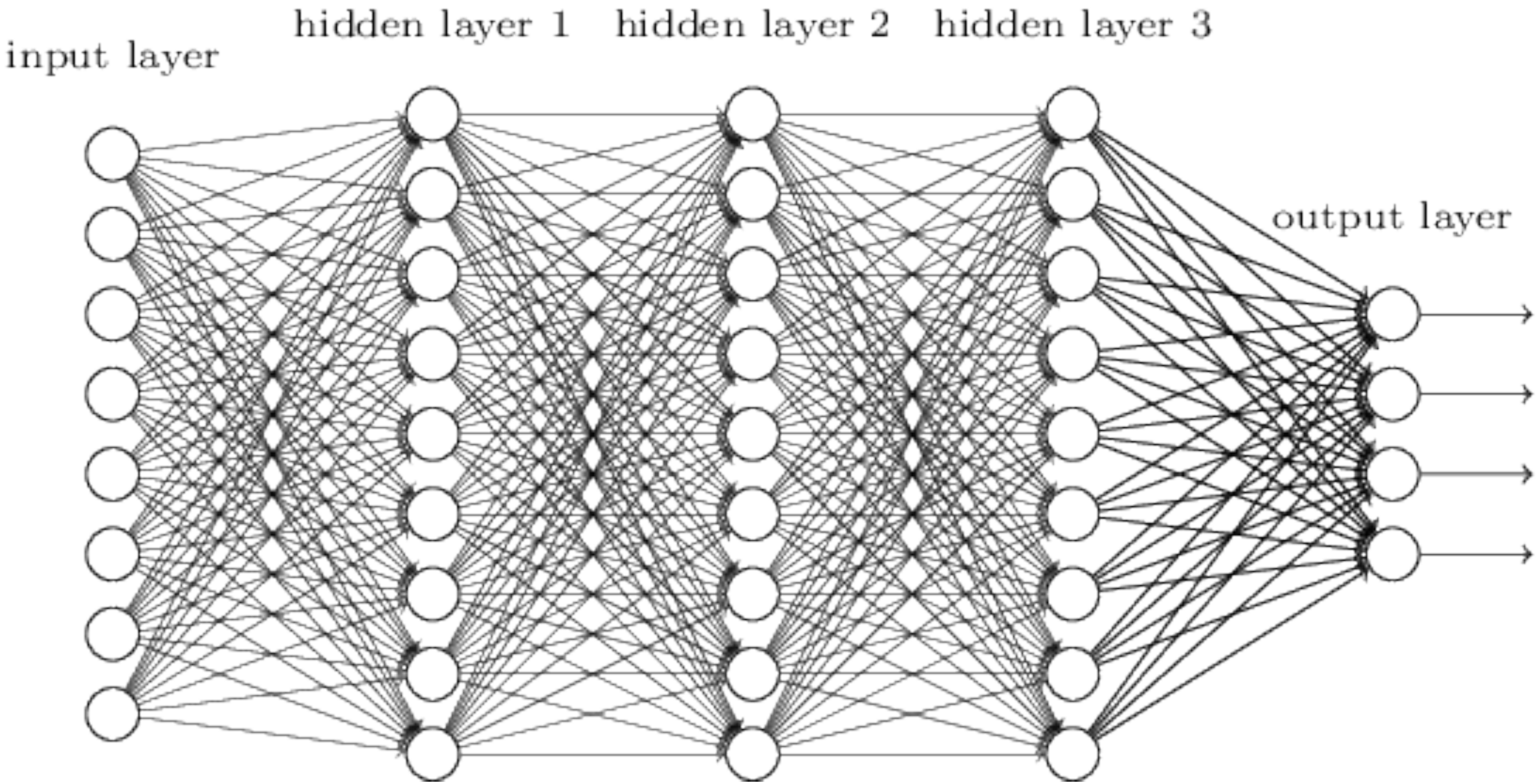


ttps://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/DlPjv2iXcAAUi6X.jpg-large.jpeg

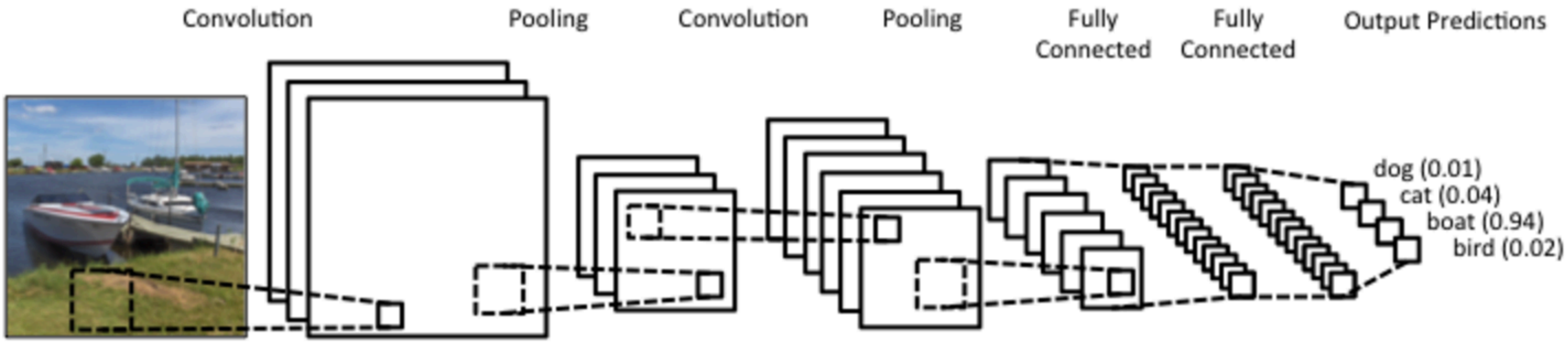
MNIST Dataset



Deep Neural Networks



Convolutional Neural Networks



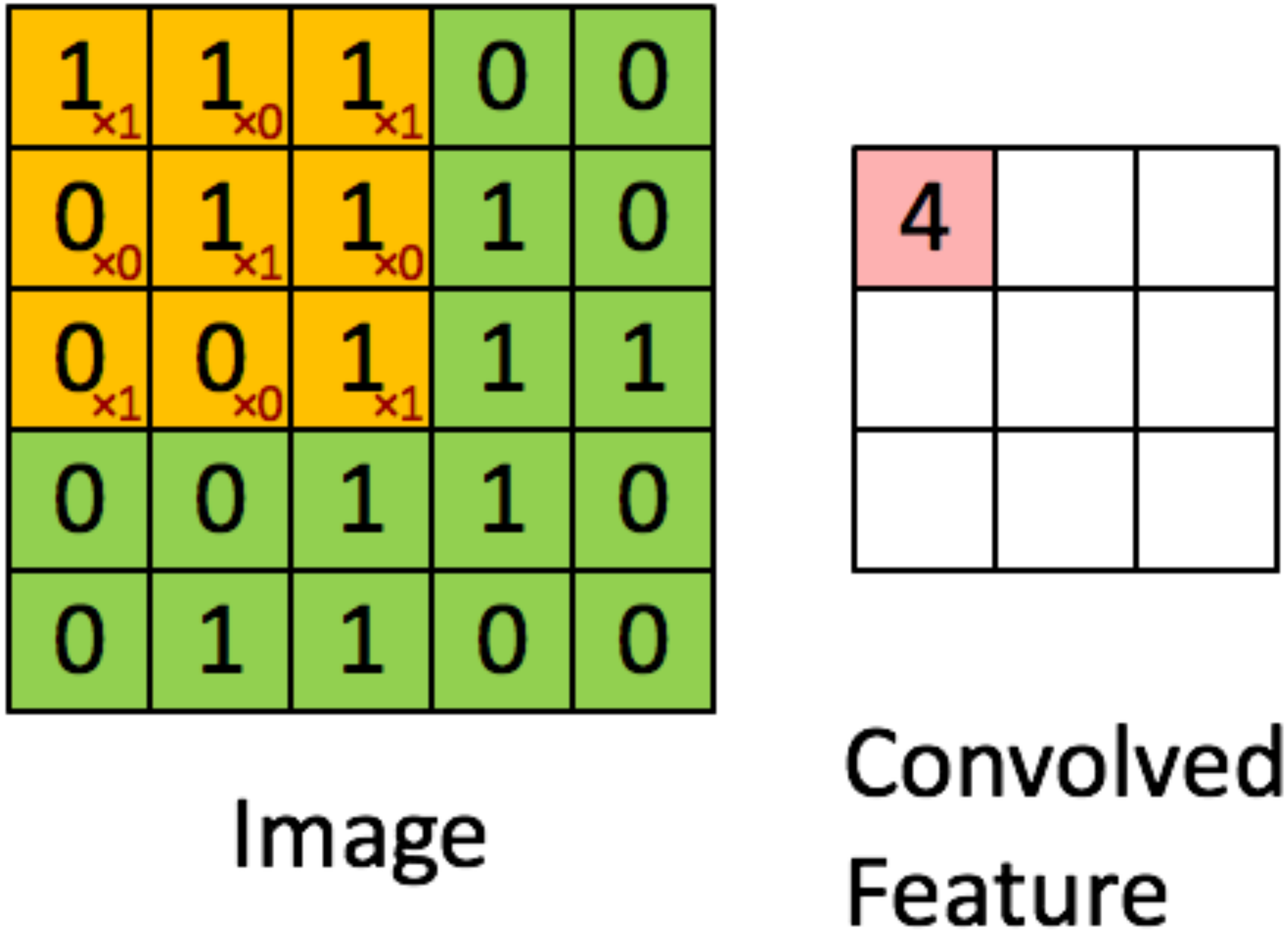
Stack successive convolutions and pooling

Extract important features from image

Use as input for neural network

Convolutional bit | "Regular" neural network

Convolutional Neural Networks



Convolutional Neural Networks

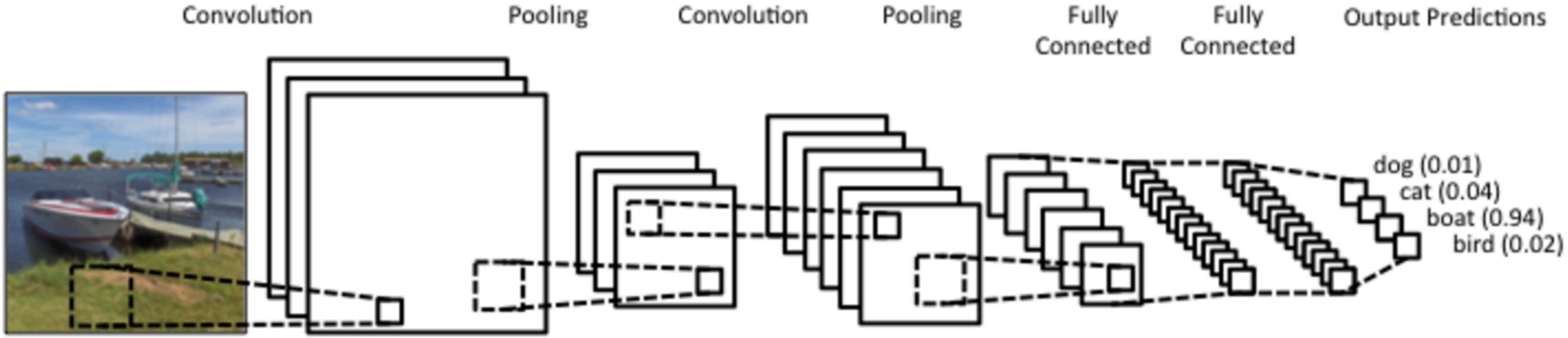
Input image

Convolution Kernel

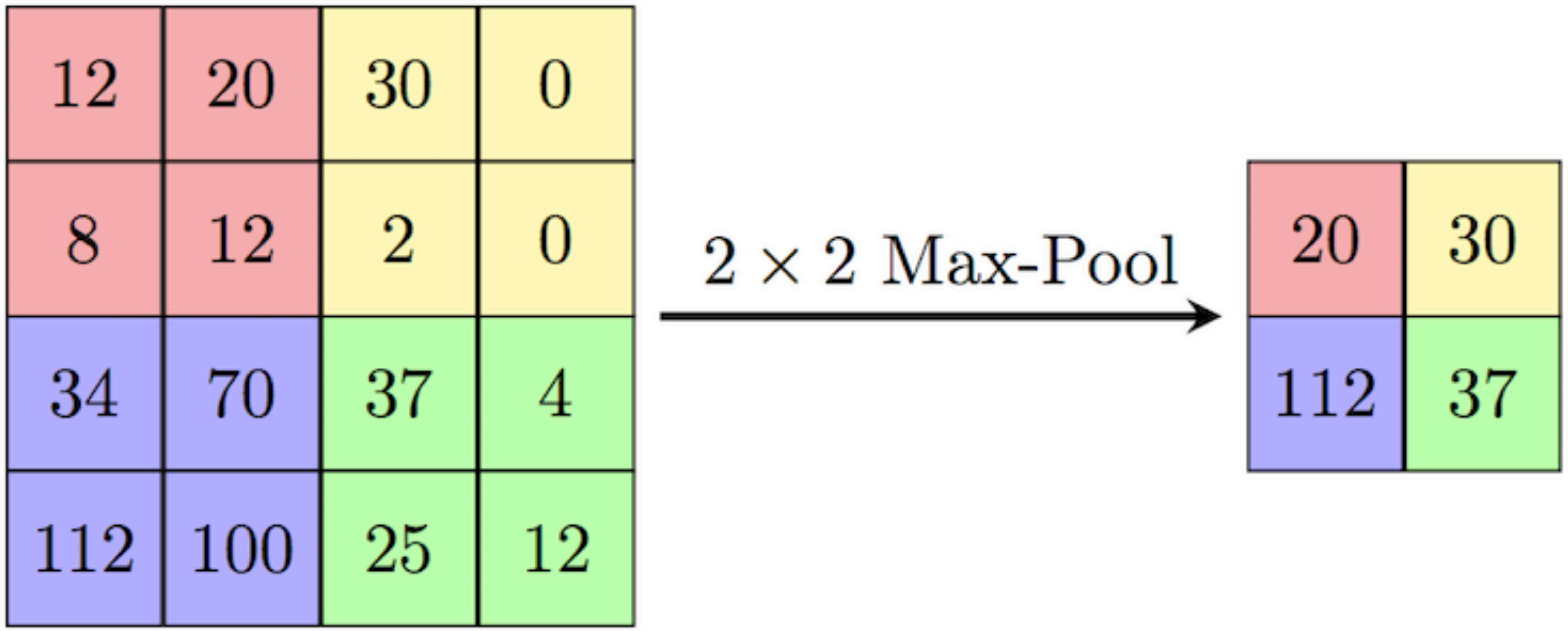
$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Feature map

Convolutional Neural Networks

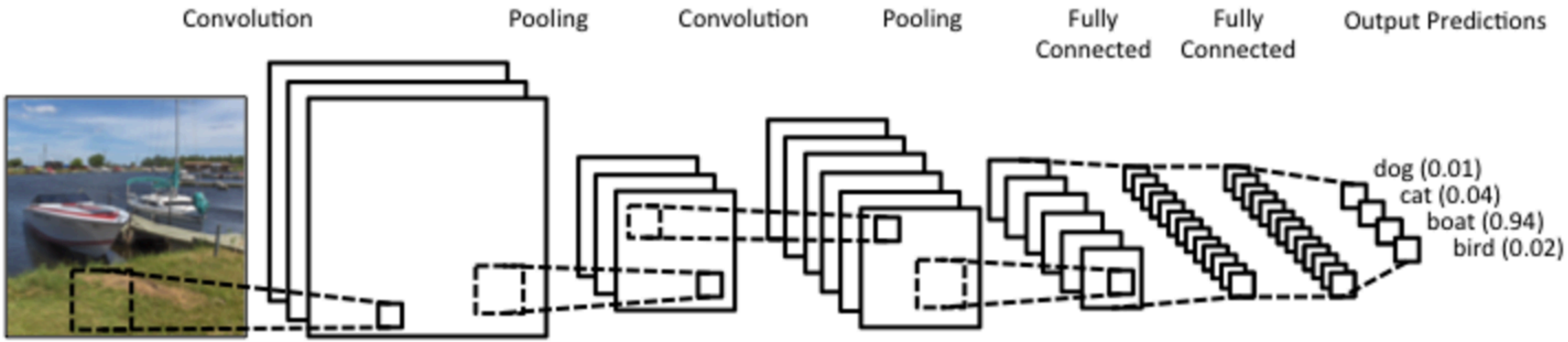


Max Pooling



This operation is pretty straightforward
Used to downsample the and reduce the dimensionality of the image
But will still maintain the most important features seen
Allows us to successively increase the scale probed by convolutional layers

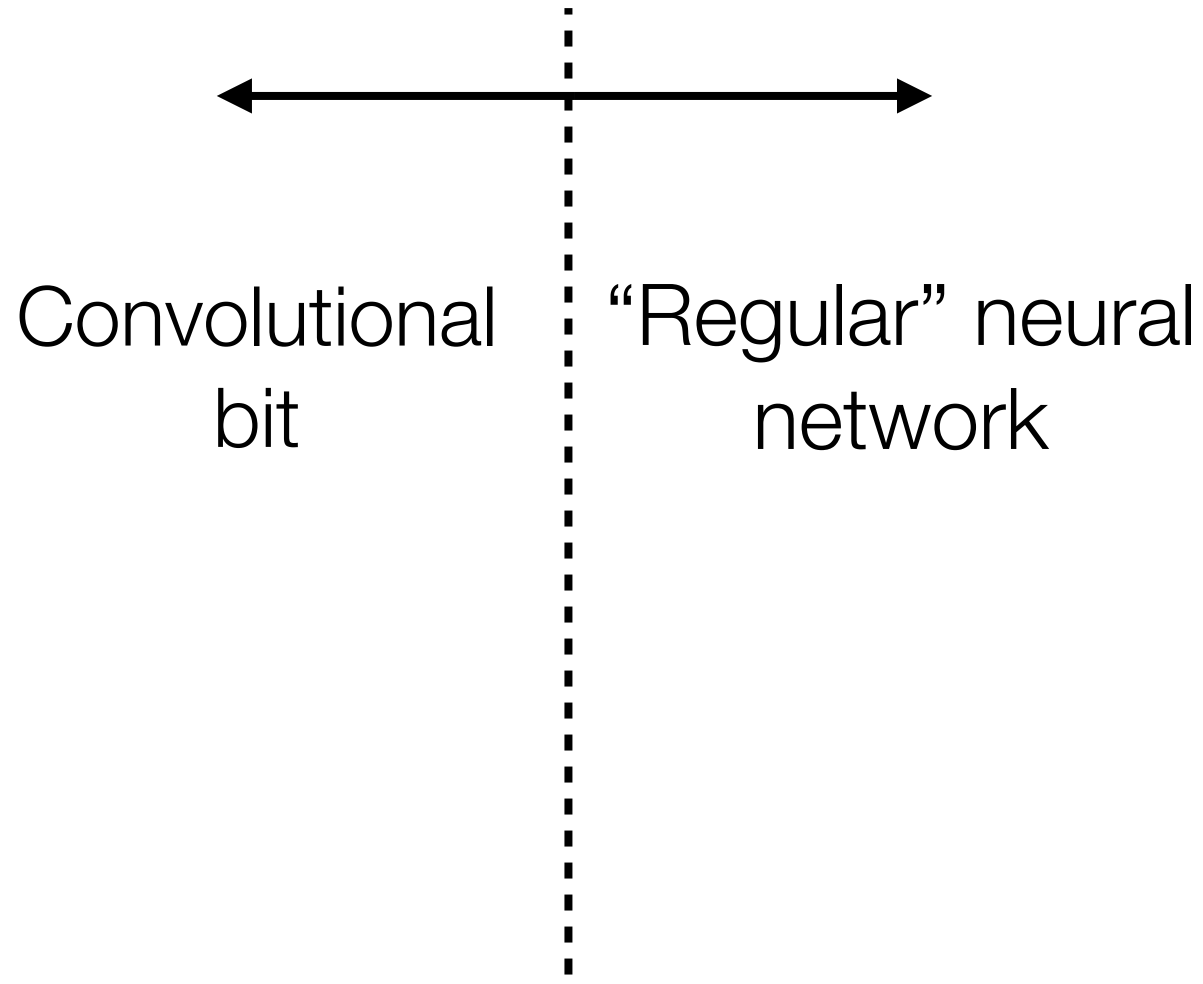
Convolutional Neural Networks

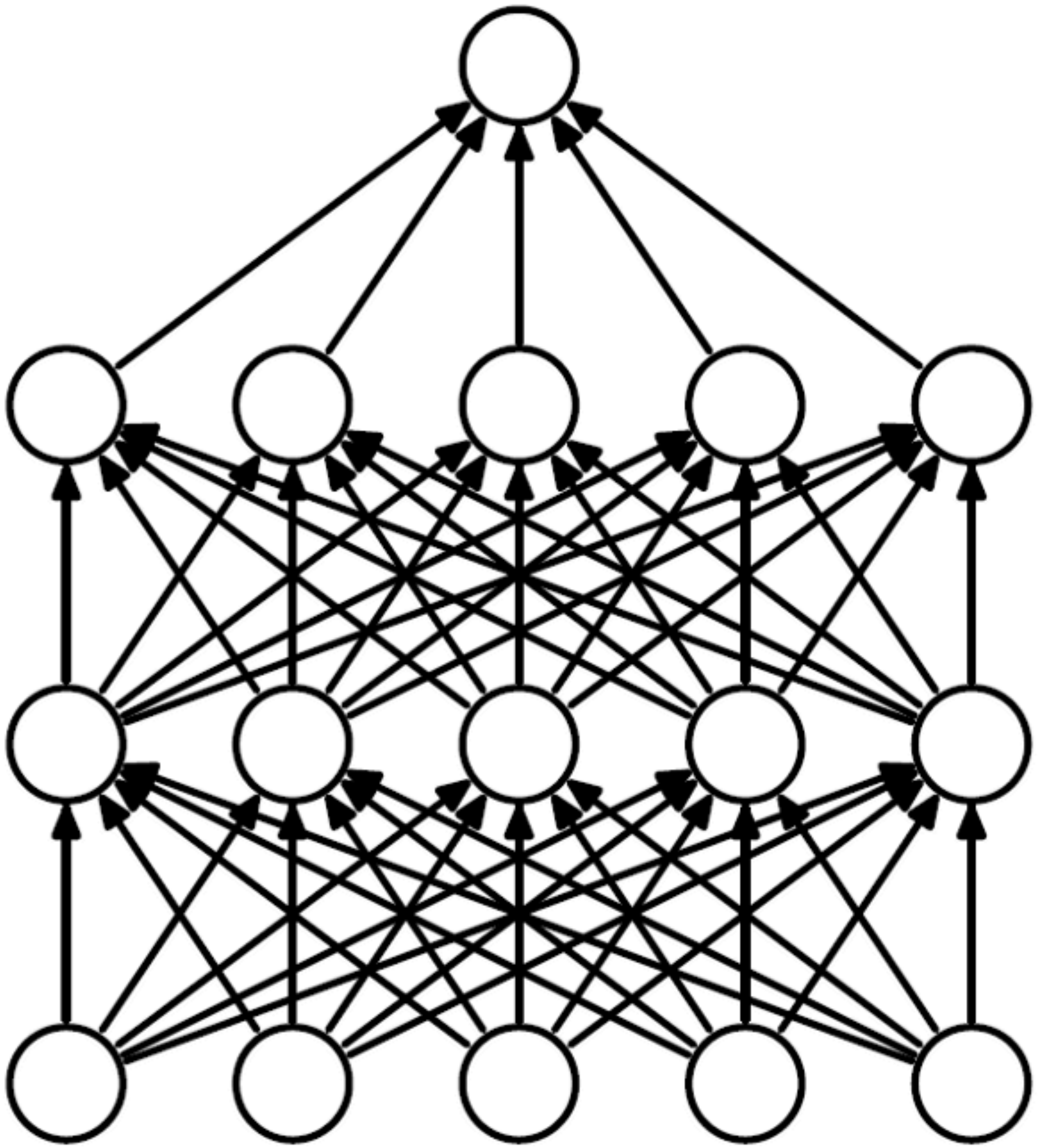


Stack successive convolutions and pooling

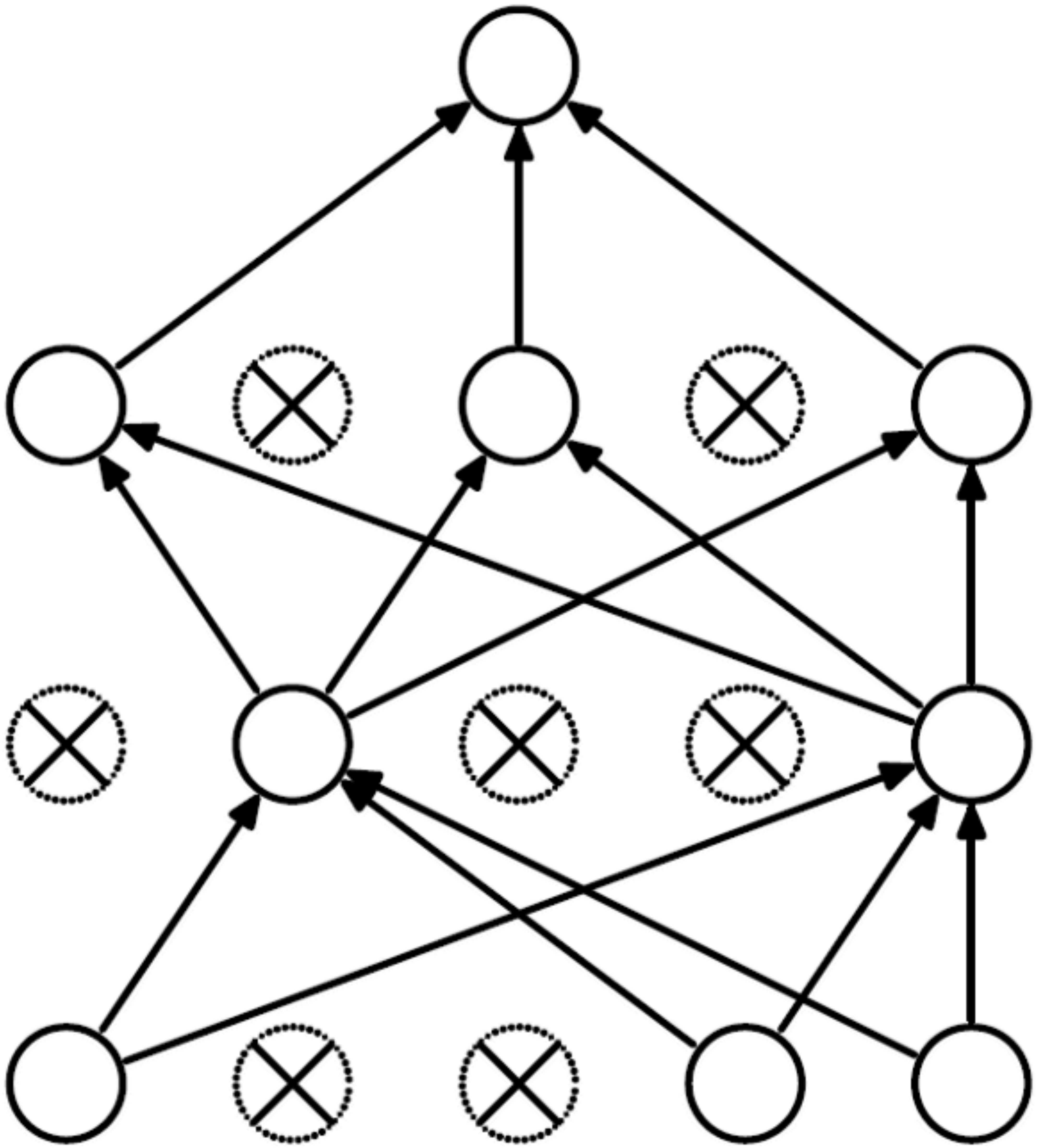
Extract important features from image

Use as input for neural network





(a) Standard Neural Net

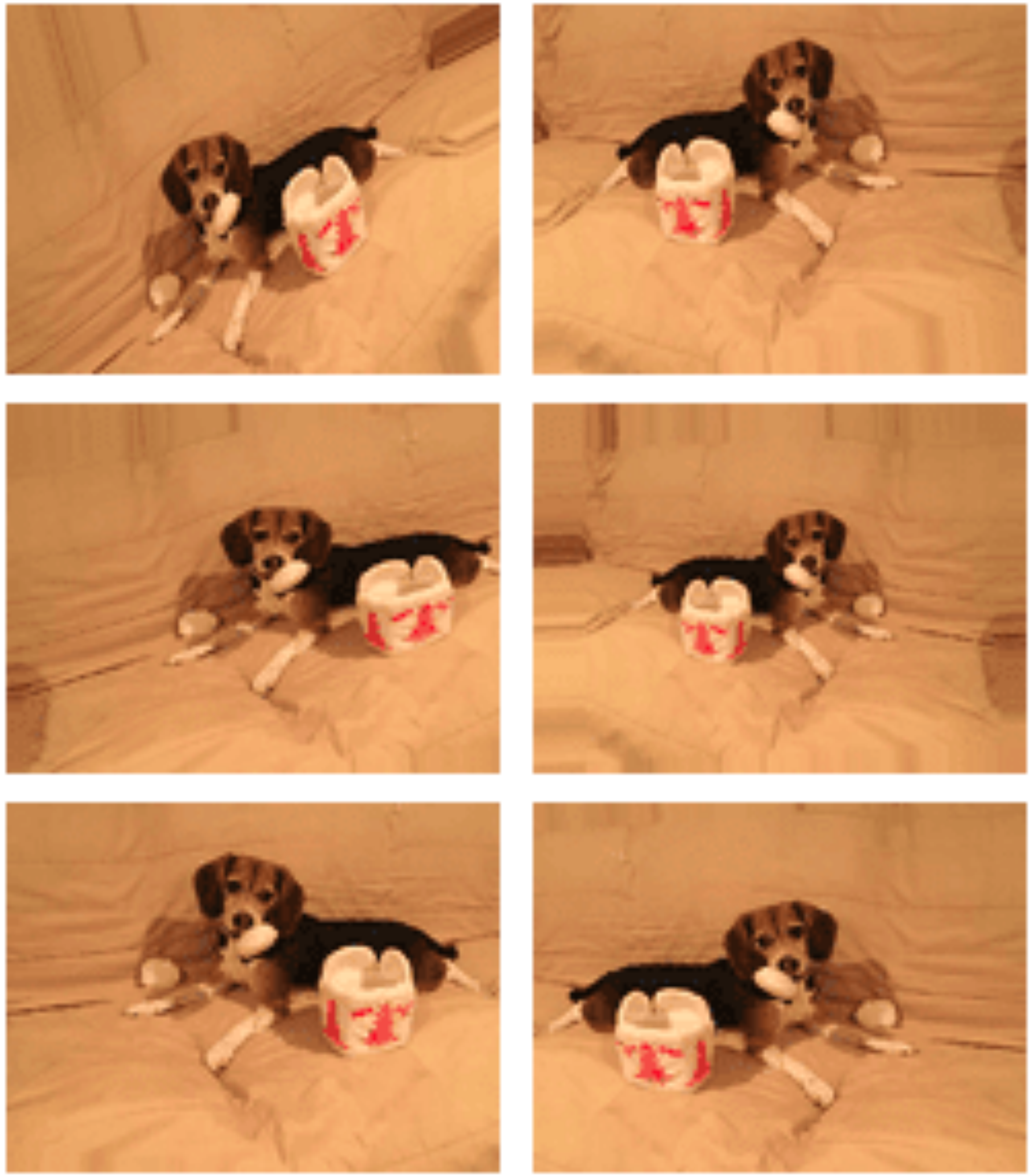


(b) After applying dropout.

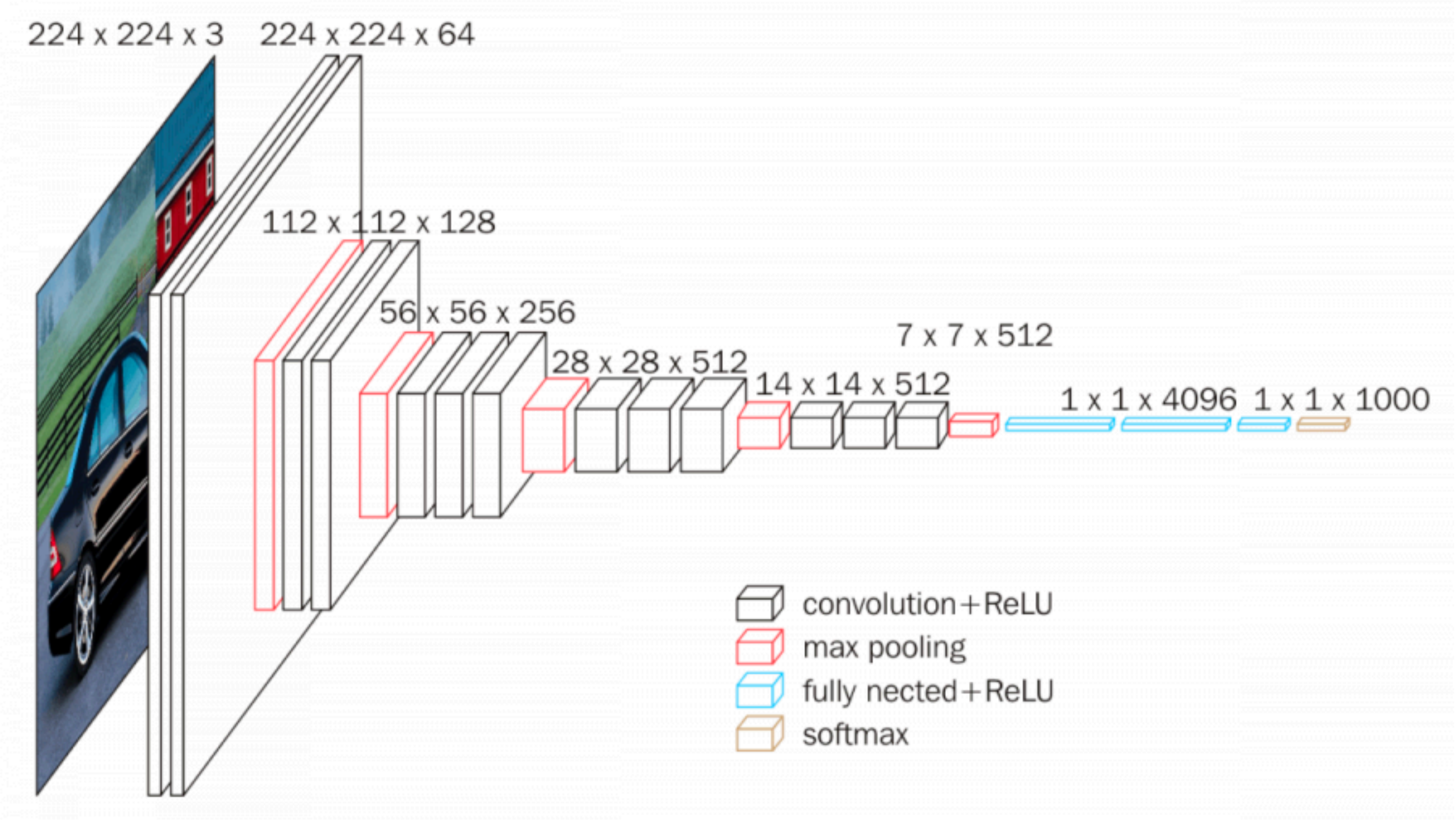
Input Image

Keras

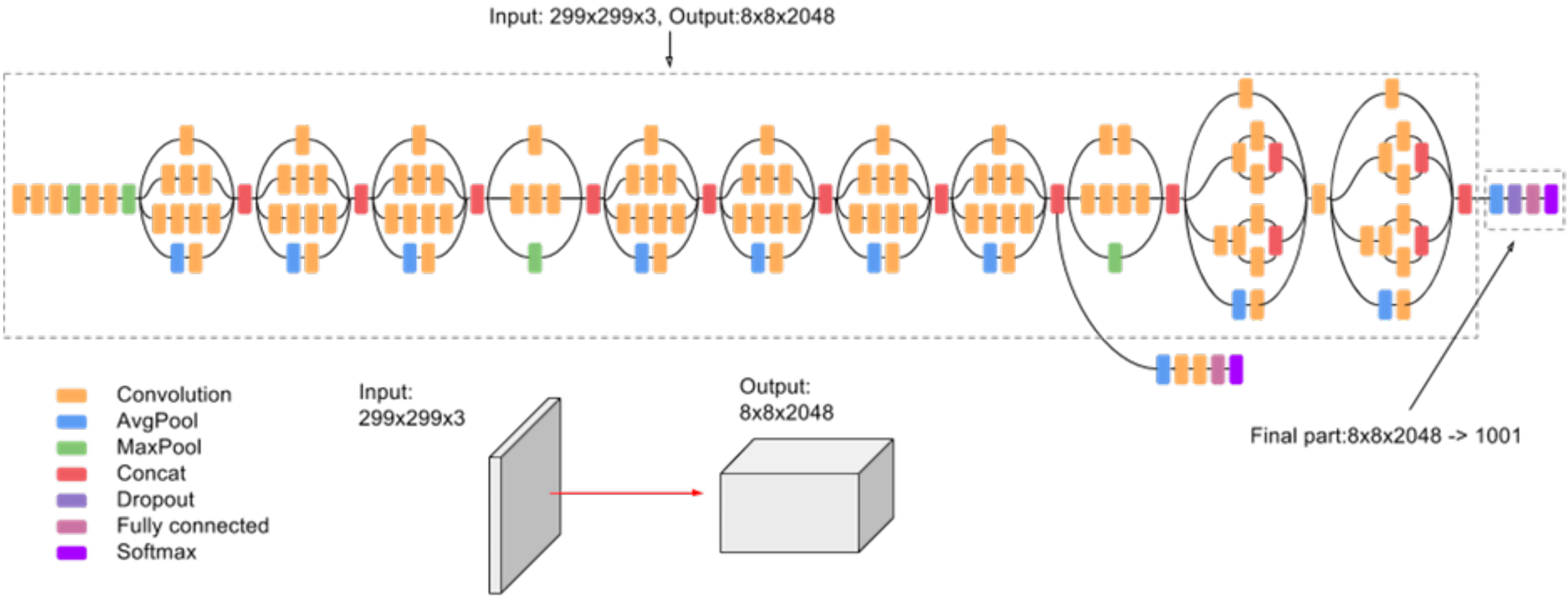
Augmented Images



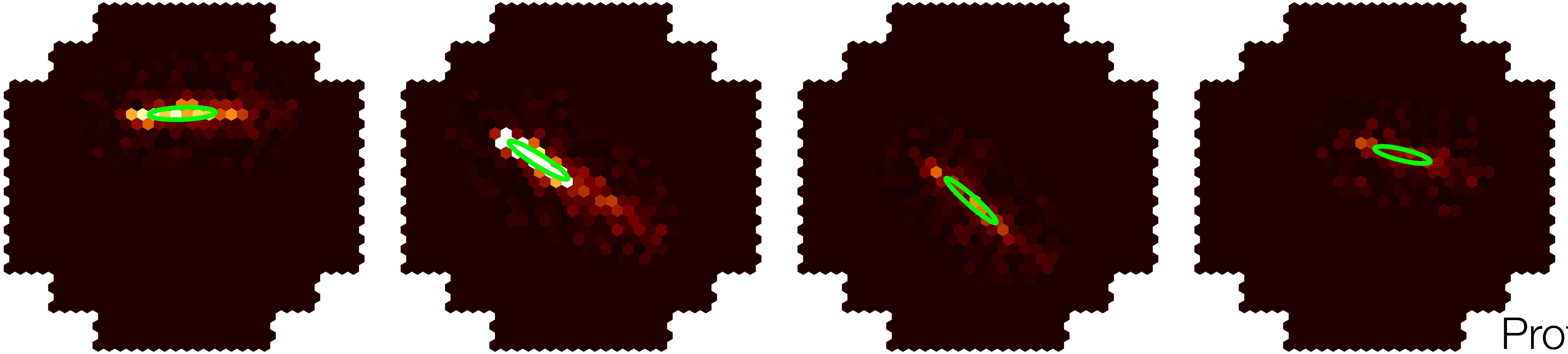
https://keras.io/api/applications/



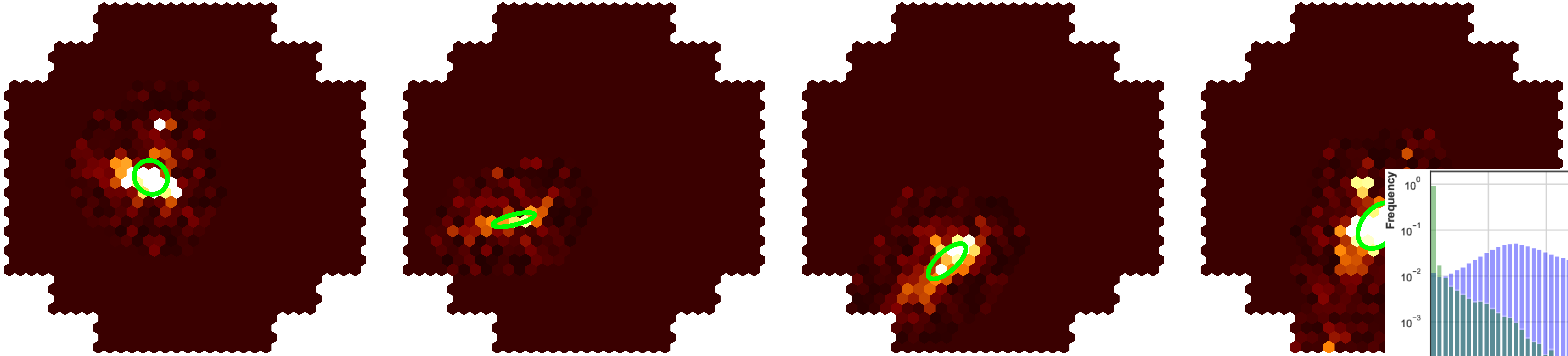
https://keras.io/api/applications/

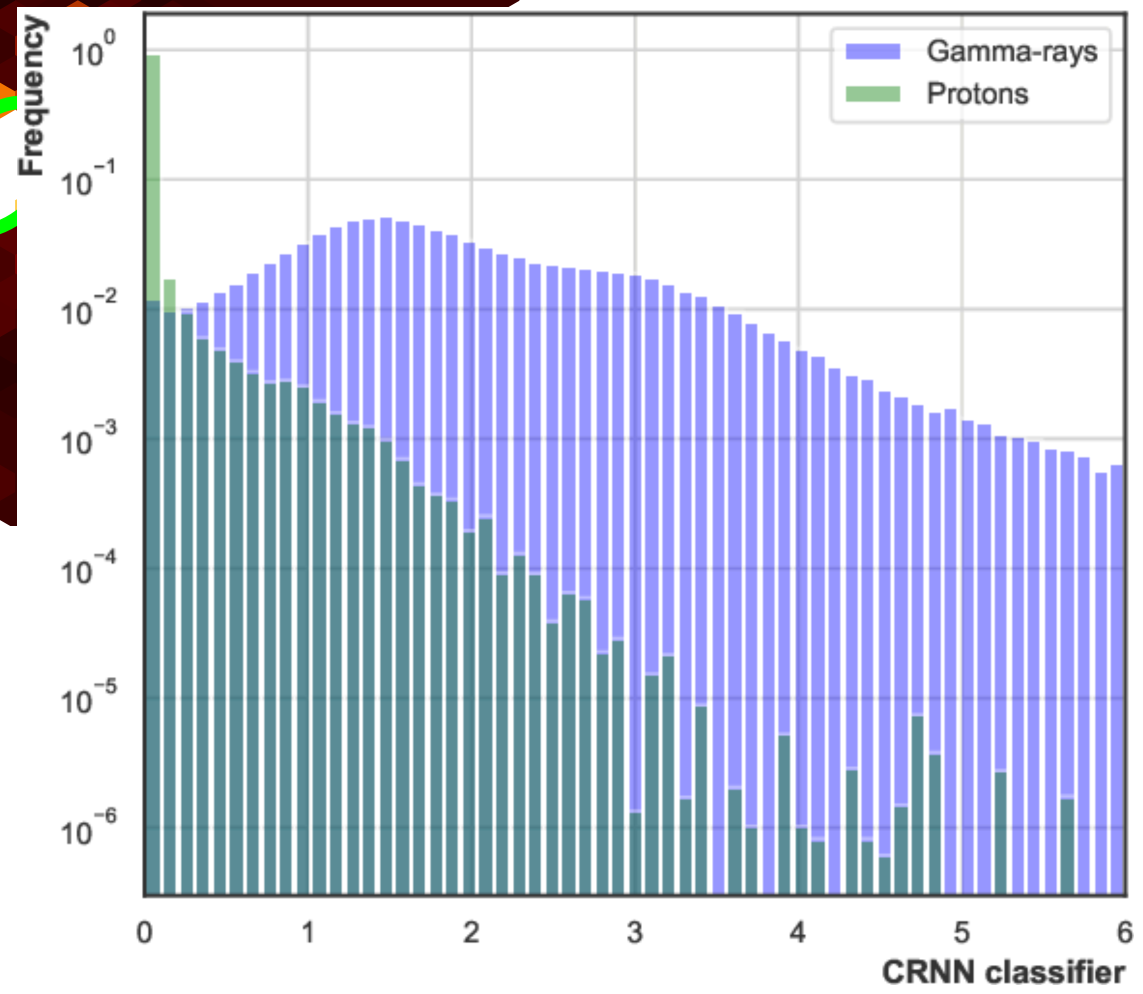


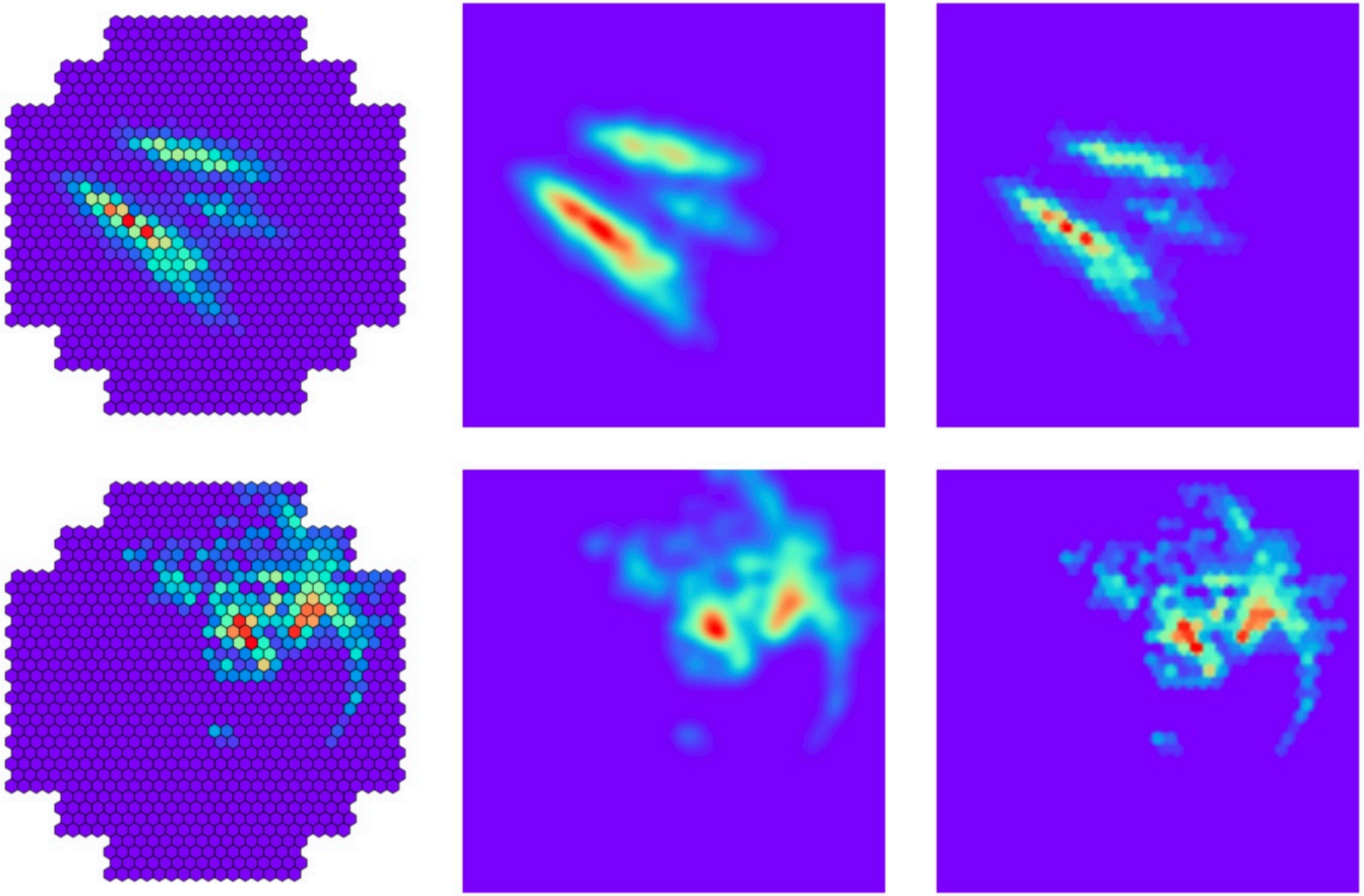
Gamma rays

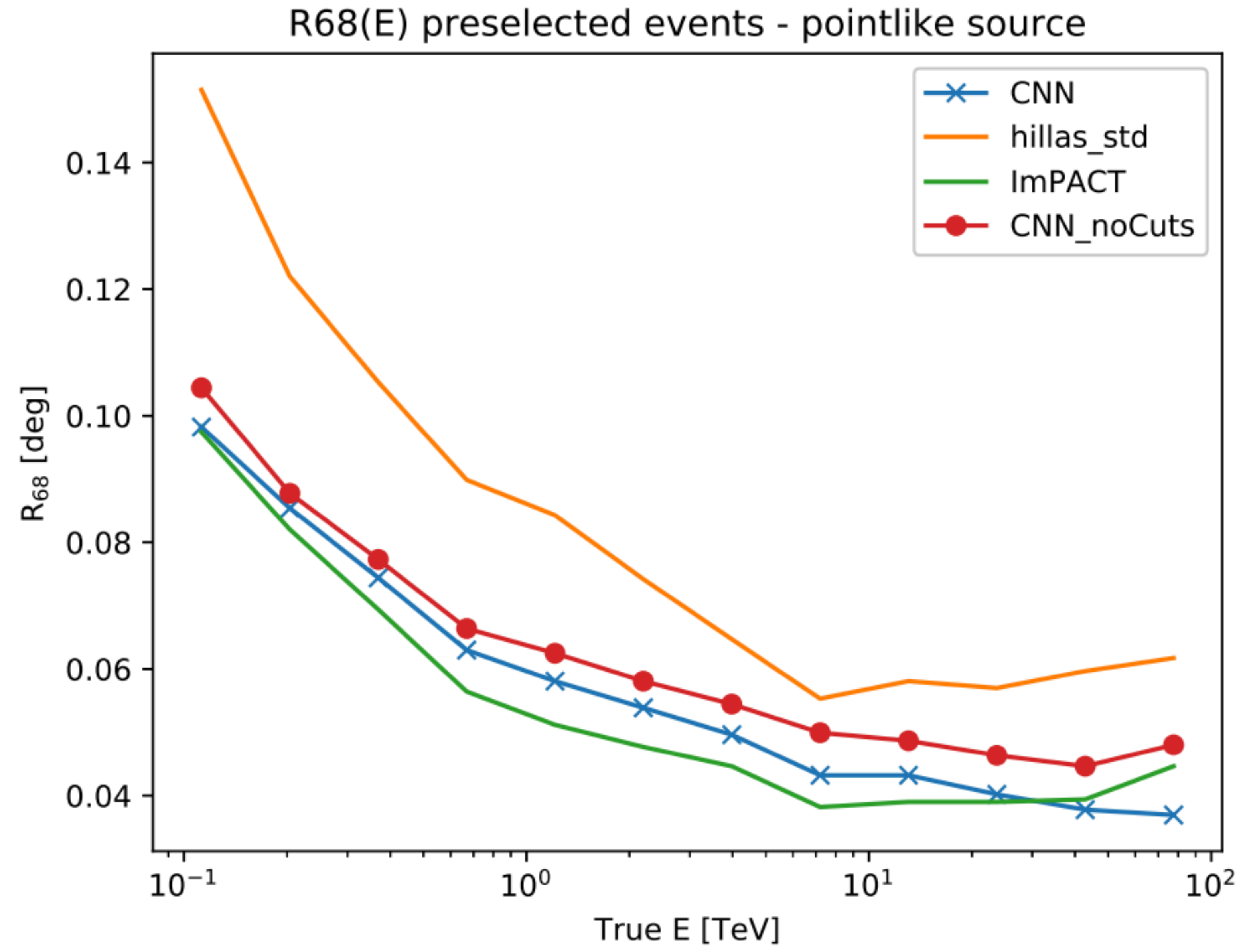


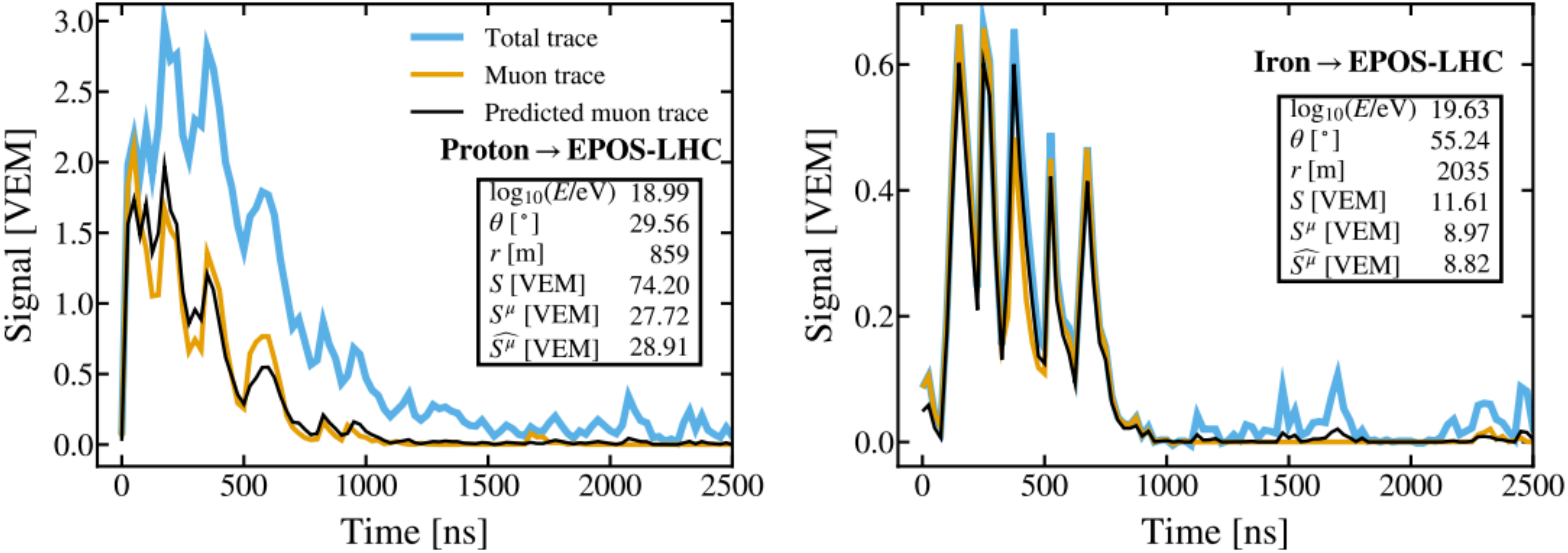
Protons








R68(E) preselected events - pointlike source
CNN
hillas_std
ImPACT
CNN_noCuts
R_{68} [deg]
0.14
0.12
0.10
0.08
0.06
0.04
10^{-1}
10^{0}
10^{1}
10^{2}
True E [TeV]



Total trace
Muon trace
Predicted muon trace
Proton → EPOS-LHC
log10(E/eV) 18.99
θ [°] 29.56
r [m] 859
S [VEM] 74.20
S^μ [VEM] 27.72
Ŝ^μ [VEM] 28.91
Signal [VEM]
Time [ns]
Iron → EPOS-LHC
log10(E/eV) 19.63
θ [°] 55.24
r [m] 2035
S [VEM] 11.61
S^μ [VEM] 8.97
Ŝ^μ [VEM] 8.82
Signal [VEM]
Time [ns]

MINI-ZORK I: The Great Underground Empire

Release 34 / Serial number 871124

West of House
You are standing in an open field west of a white house, with a boarded front door. You could circle the house to the north or south.
There is a small mailbox here.

MINI-ZORK I: The Great Underground Empire

Release 34 / Serial number 871124

West of House
You are standing in an open field west of a white house, with a boarded front door. You could circle the house to the north or south.
There is a small mailbox here.

>open mailbox
Opening the small mailbox reveals a leaflet.

>take leaflet
Taken.

>read leaflet_