MODERN COMPUTER VISION

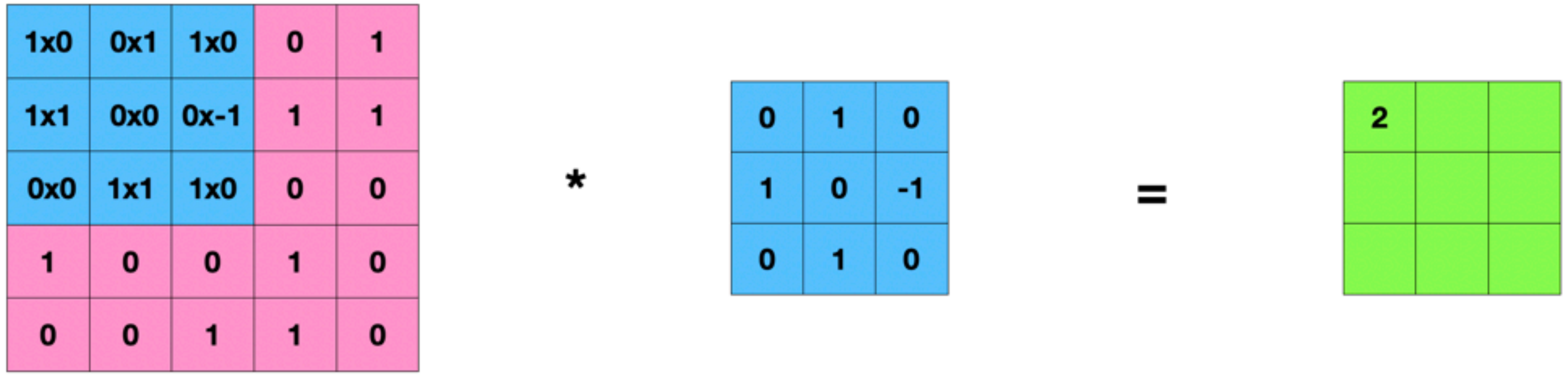
BY RAJEEV RATAN

Summary of Convolutional Neural Networks (CNNs)

Putting it all together

Conv Layers

$$(1\times0)+(0\times1)+(1\times0)+(1\times1)+(0\times0)+(0\times-1)+(0\times0)+(1\times1)+(1\times0)=2$$



Input Image | Filter or Kernel | Output or Feature Map

- Convolution Operations occur when we Convolve our Filters with the input image by sliding it over our image
- This produces an output called a Feature Map
- Feature Maps are now the inputs to the next layer of our CNN

Stride, Padding and Kernel Size

Feature Map Size = $n - f + 1 = m$

Feature Map Size = $7 - 3 + 1 = 5$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 1 & -1 & 2 & 2 \\ -1 & 1 & 3 & 2 & 1 \\ 2 & 1 & 1 & 1 & 2 \\ 1 & 1 & 1 & 0 & 2 \\ 2 & 0 & 2 & 3 & 1 \end{bmatrix}$$

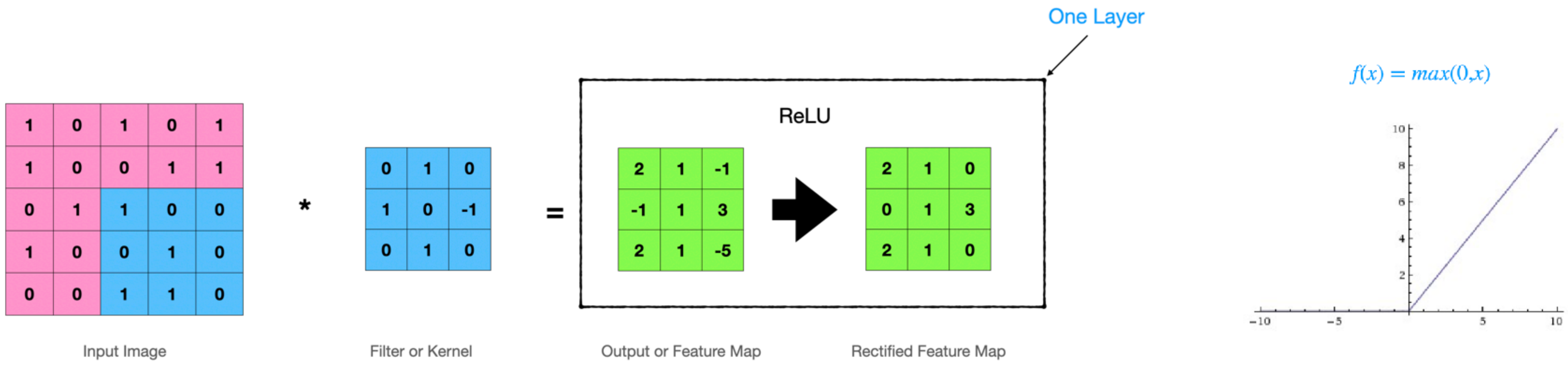
7 x 7 $n \times n$

3 x 3 $f \times f$

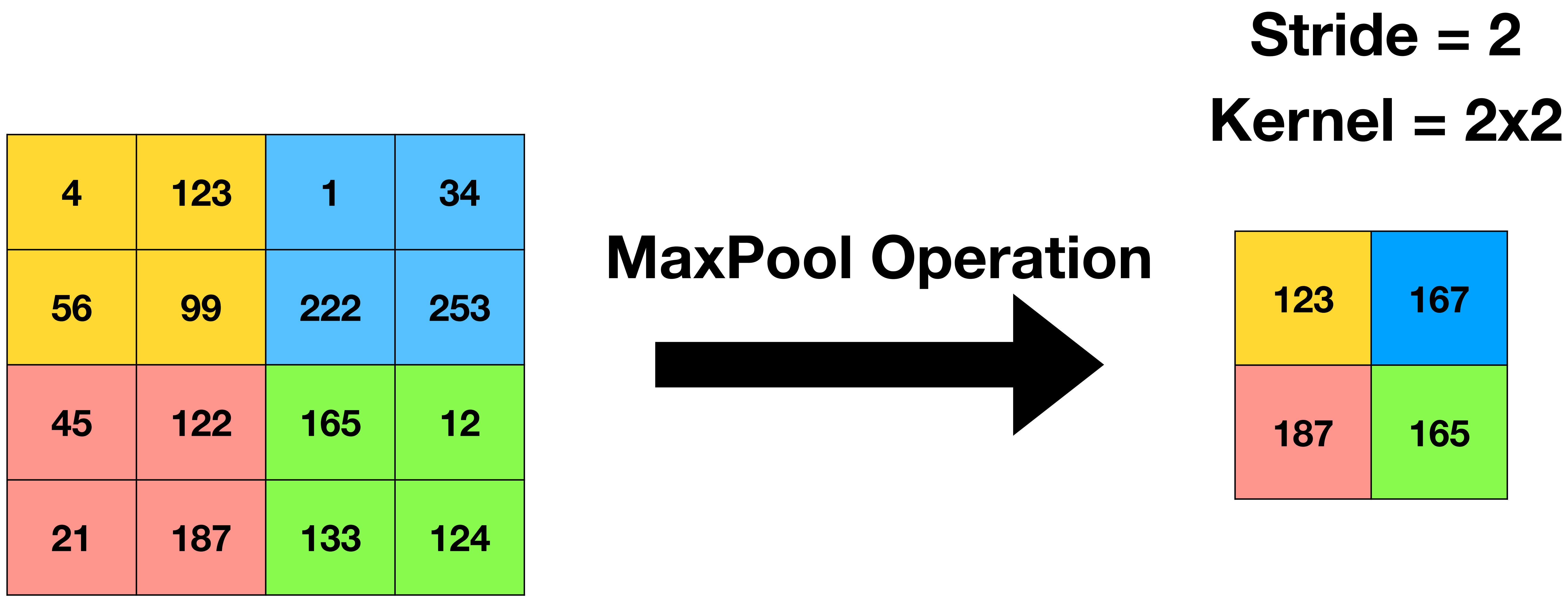
5 x 5 $m \times m$

- We use Stride, Padding and Kernel size to control the output size of our Feature Map
- $(n \times n) * (f \times f) = (\dfrac{n + 2p - f}{s} + 1) \times (\dfrac{n + 2p - f}{s} + 1)$

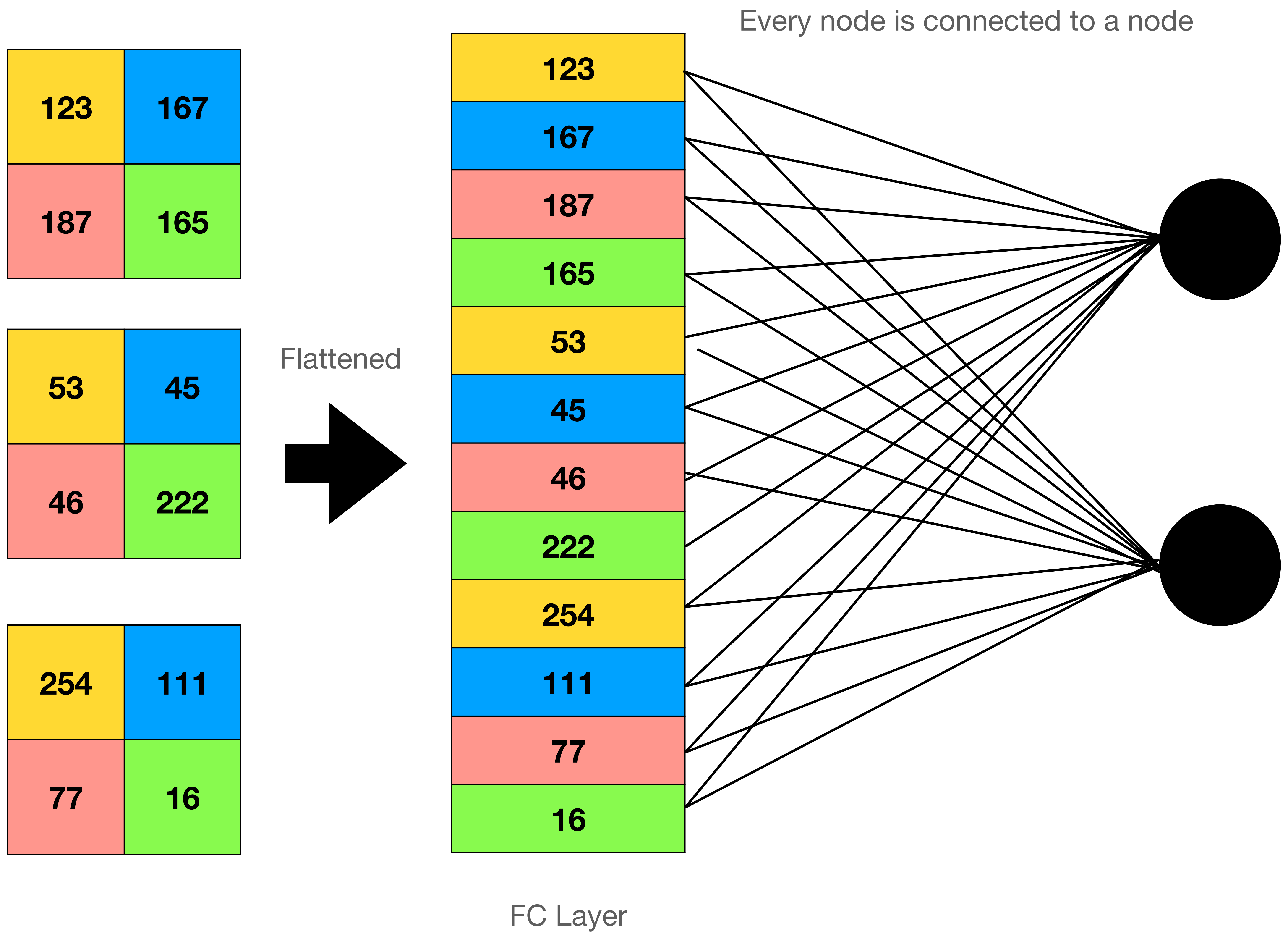
Activation Layer ReLU - Adds Non-Linearity to our Network



Max Pooling - Reduce Dimensionality



Fully Connected Layer - Max Pool Layer is Flattened

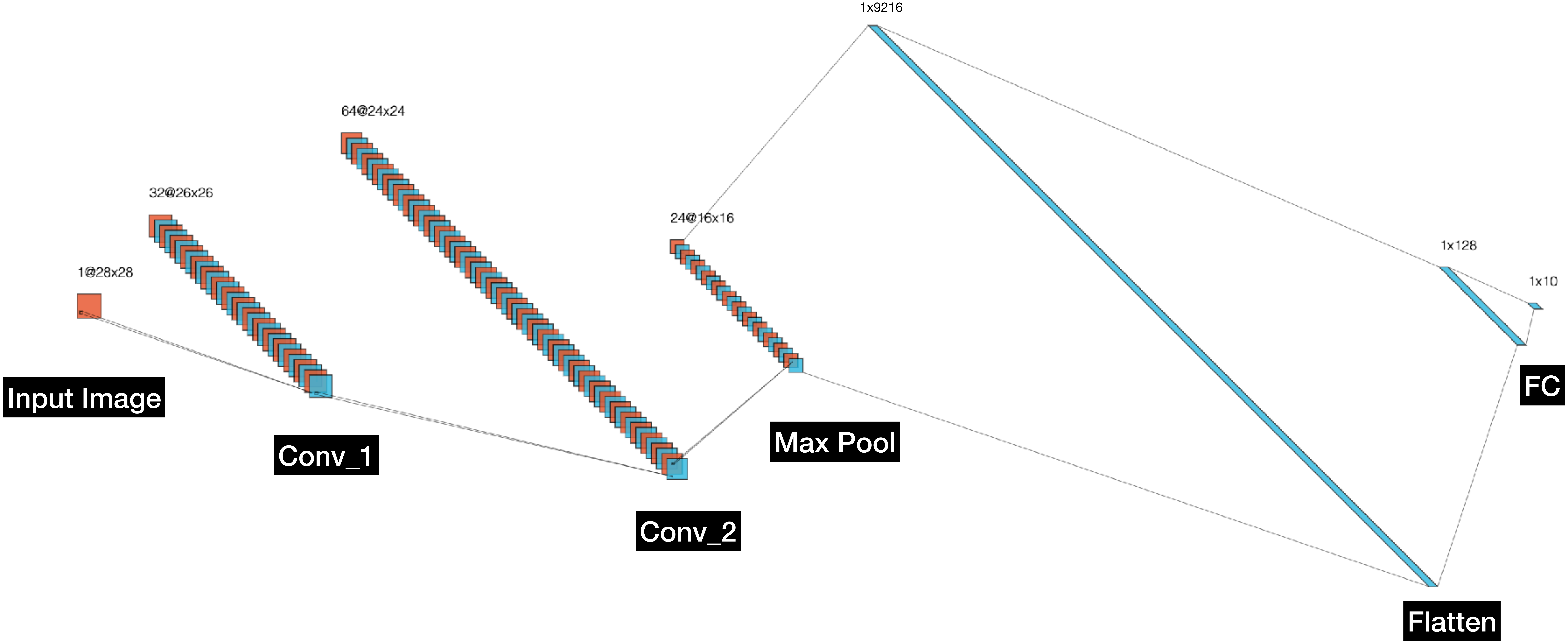


Softmax Layer

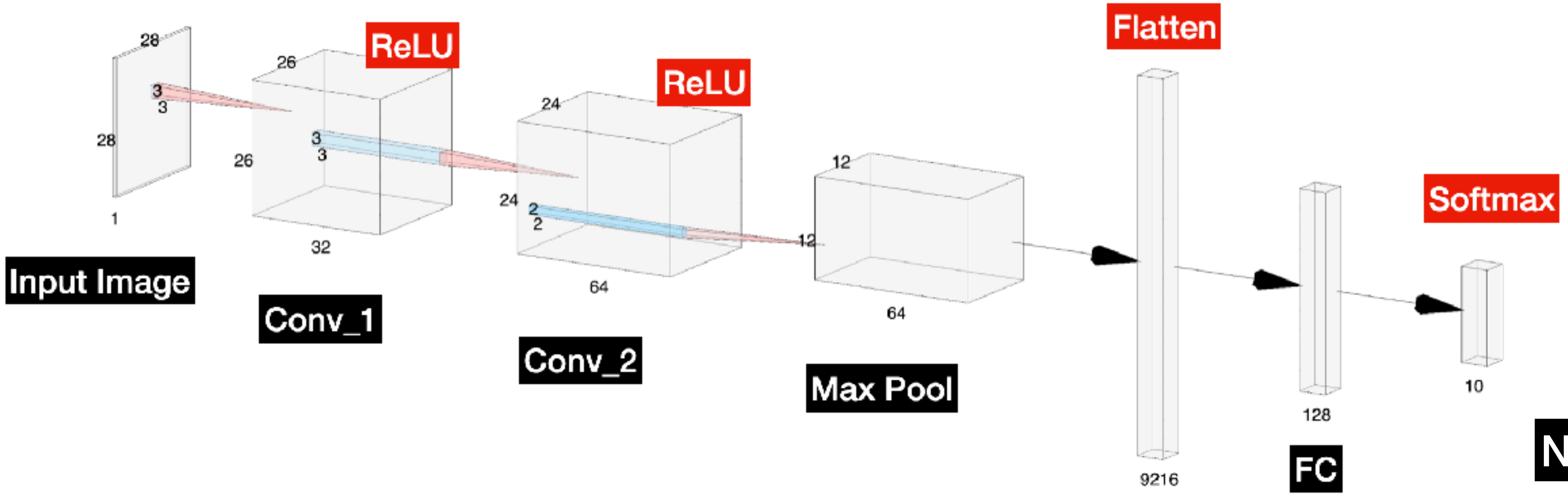
Logits Scores	Probabilities
2.0	0.7
1.0	0.2
0.1	0.1

$$\boxed{softmax(x)_i = \frac{exp(x_i)}{\sum_j exp(x_j))}}$$

Our Basic CNN



Parameters in our Basic CNN



Conv_1

$$((Height \times Width \times Depth) + bias) \times N_f$$

$$((3 \times 3 \times 1) + 1) \times 32 = 320$$

Conv_2

$$((Height \times Width \times Depth) + bias) \times N_f$$

$$((3 \times 3 \times 32) + 1) \times 64 = 18{,}494$$

No Trainable Parameters

- Max Pool
- Flatten
- ReLU

Fully Connected/Dense

$$(Length + bias) \times N_{nodes}$$

$$(9216 + 1) \times 128 = 1{,}179{,}776$$

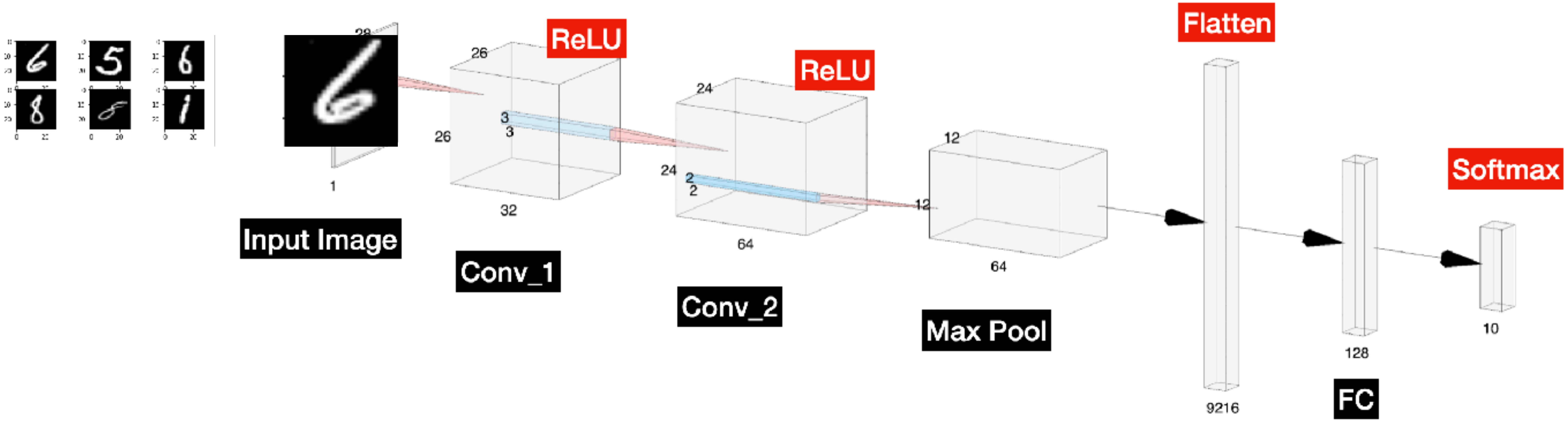
Final Output (FC/Dense)

$$(Length + bias) \times N_{nodes}$$

$$(128 + 1) \times 10 = 1{,}290$$

Layer	Parameters
Conv_1 + ReLU	**320**
Conv_2 + ReLU	**18494**
Max Pool	**0**
Flatten	**0**
FC_1	**1,179,776**
FC_2 (Output)	**1,290**
Total	**1,199,882**

The Training Process



P	6	5	6	8	8	1
0	0	0.9	0.83	0.21	0.19	0.62
1	0.73	0.8	0.89	0.7	0.92	0.07
2	0.78	0.88	0.19	0.39	0.08	0.74
3	0.37	0.56	0.07	0.64	0.64	0.9
4	0.63	0.25	0.79	0.94	0.52	0.55
5	0.87	0.65	0.57	0.63	0.97	0.04
6	0.67	0.05	0.45	0.51	0.87	0.51
7	0.71	0.66	0.13	0.59	0.86	0.89
8	0.51	0.88	0.59	0.01	0.37	0.63
9	0.24	0.52	0.79	0.15	0.63	0.78

Overview on Training

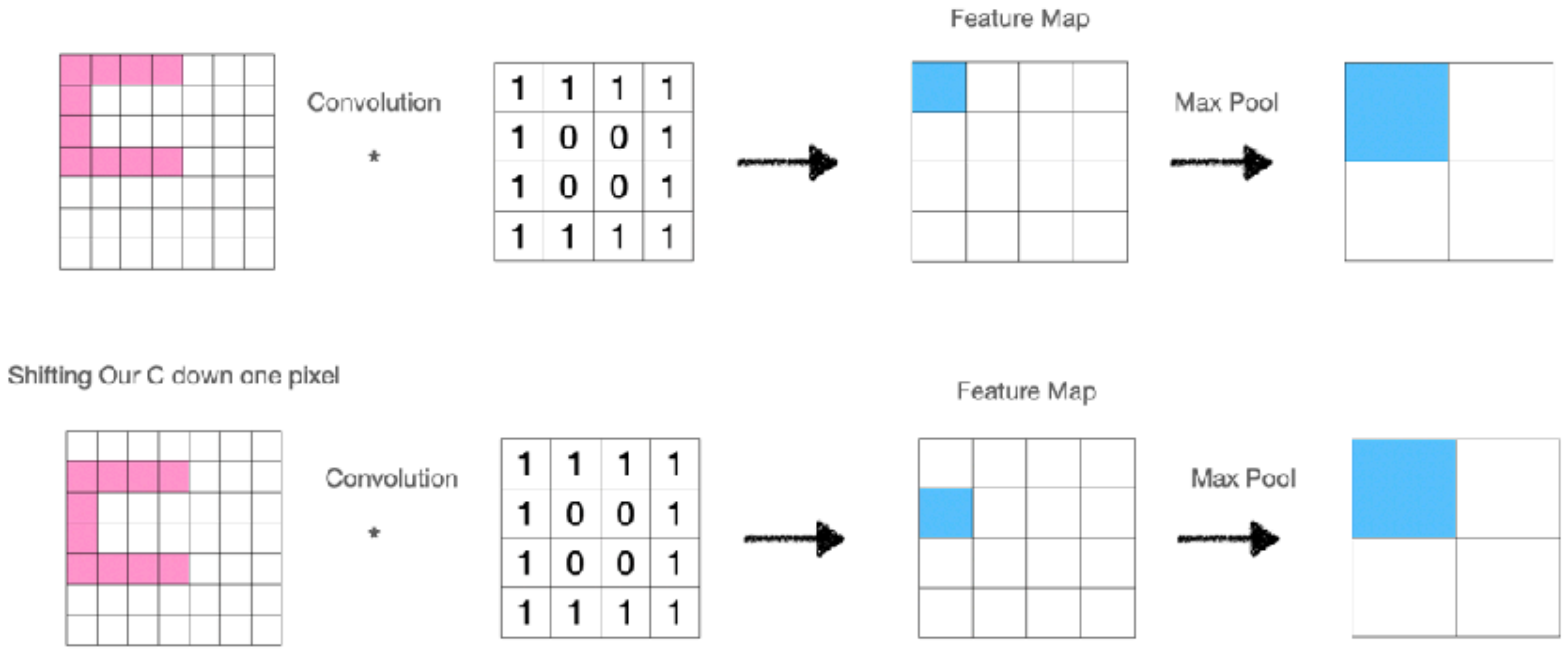
- CNN Model is **designed and defined**
- Weights are **initialised with random values**
- Batches of Images (typically 8 to 256) are **forward propagated** through our CNN Model
- Using **Back Propagation** with **Mini-Batch Gradient Descent** we update the individual weights (right to left)
- Using we update all our weights so that we have a lower loss
- We the entire dataset of images is forward propagated, we’ve completed an **Epoch**
- We train for **5 to 50** Epochs and stop when Loss stops decreasing

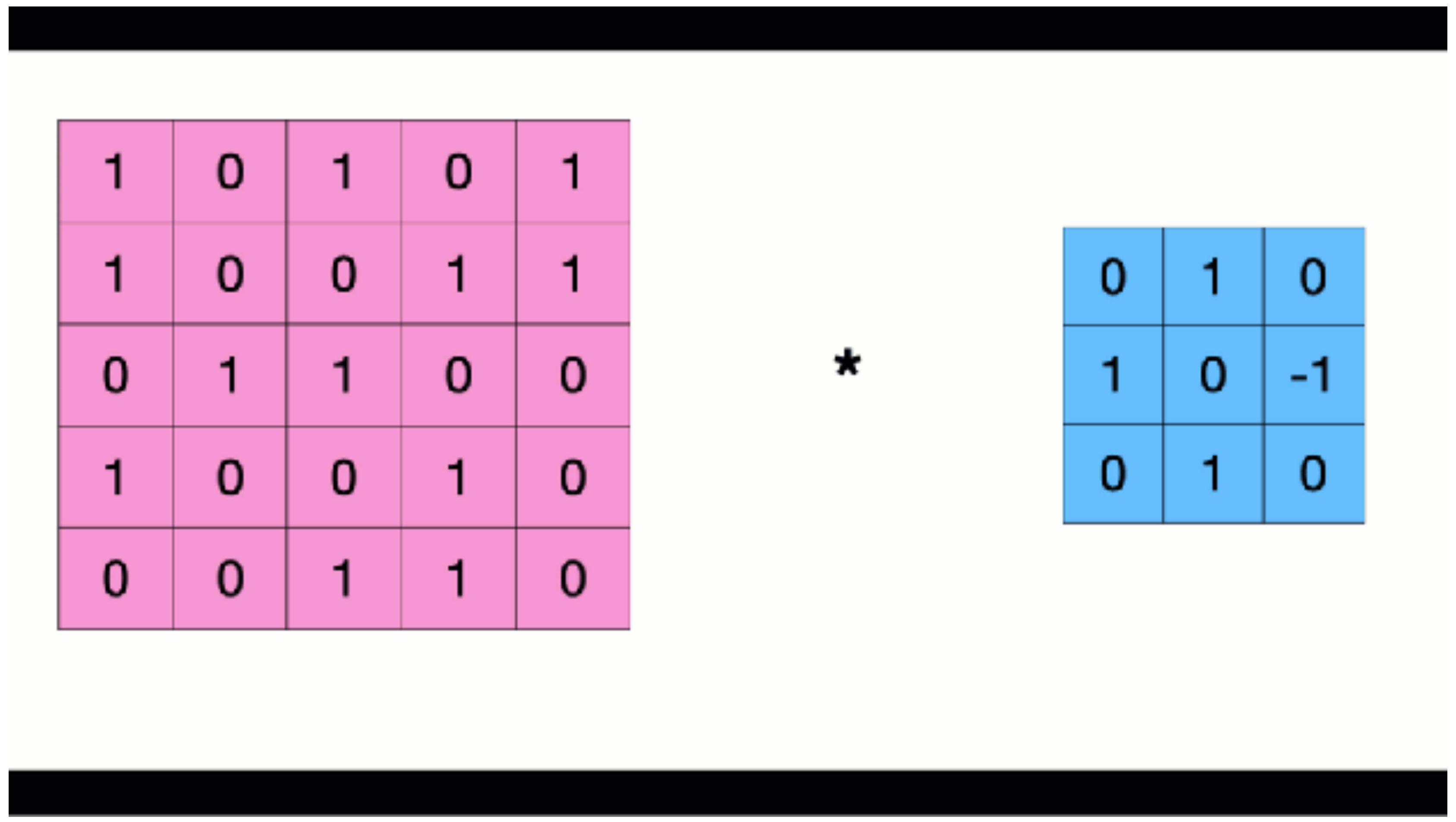
Batches, Mini-Batches, Iterations & Epochs

- We can feed images one at a time, however using mini-batches is better

Advantages of Convolution Neural Networks

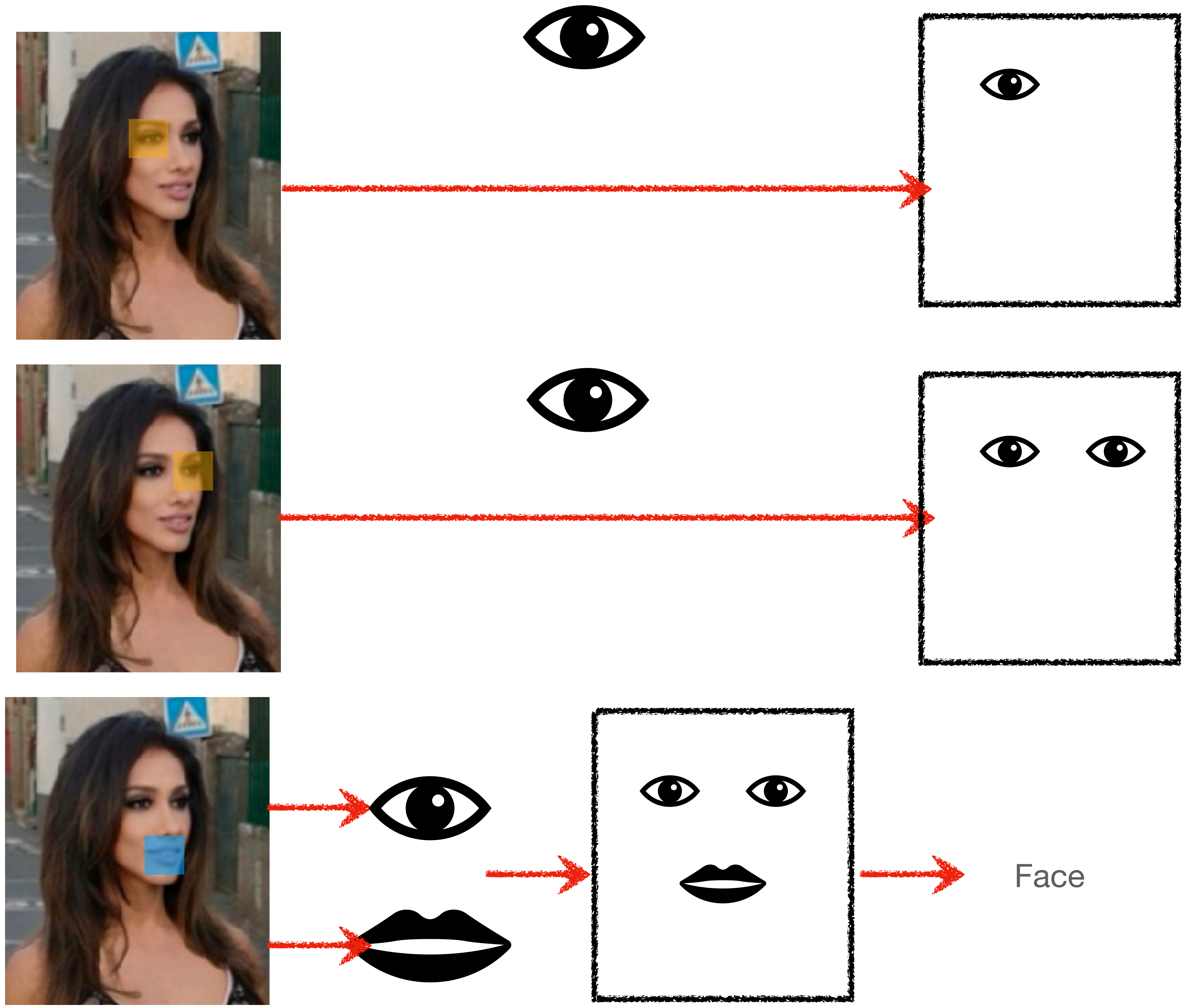
- **Invariance** - Remember our Max Pool Example
- **Parameter sharing** - where a single filter can be used all parts of an image
- **Sparsity of connections** - As we saw, fully connected layers in a typical Neural Network result in a weight matrix with large number of parameters.





Convolution Neural Networks Assumptions

- Low-level features are local
- Features are translational invariant
- High-level features are made up of low-level features



MODERN COMPUTER VISION

BY RAJEEV RATAN

Next…

History of Deep Learning and AI