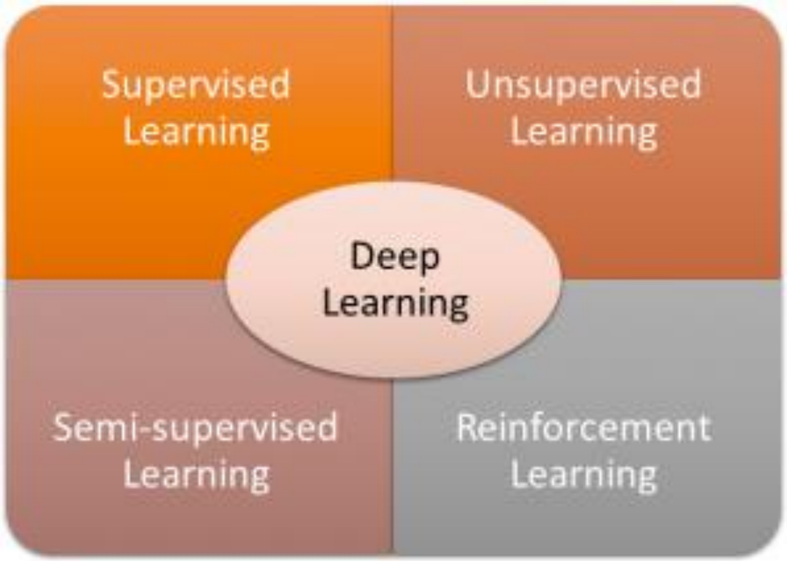
Anatomy and Arithmetic
of
Convolutional Neural Networks
Part I

Odessa, 2018

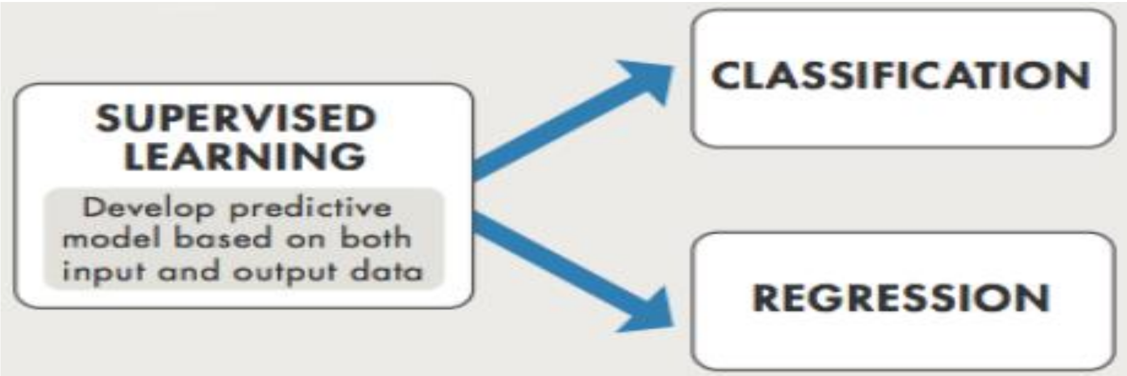
Styles of Learning

Supervised	Unsupervised	Semi-Supervised	Reinforcement
• Data has **known labels**	• No labels • Focuse on **finding patterns and gaining insight** from the data	• **Labels** known **for a subset** of data • A blend of supervised and unsupervised	• Focus on **making decisions** based on previous experience

(text and layout were taken from blogs.sas.com)

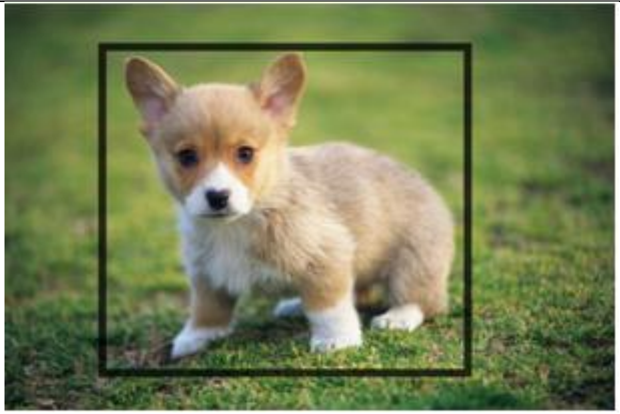


(leonardoaraujosantos.gitbooks.io)

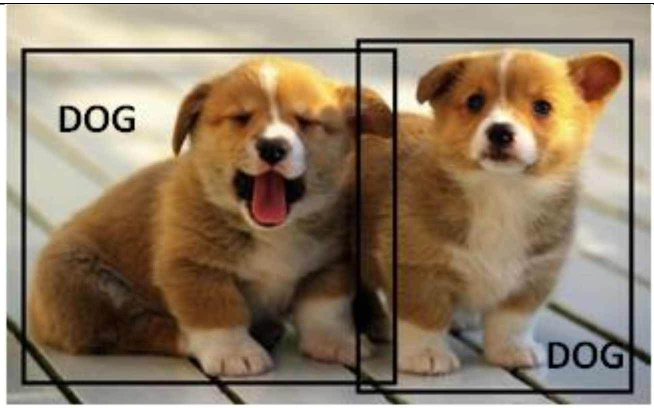


(vitalflux.com)

(classification)



(classification + localization)

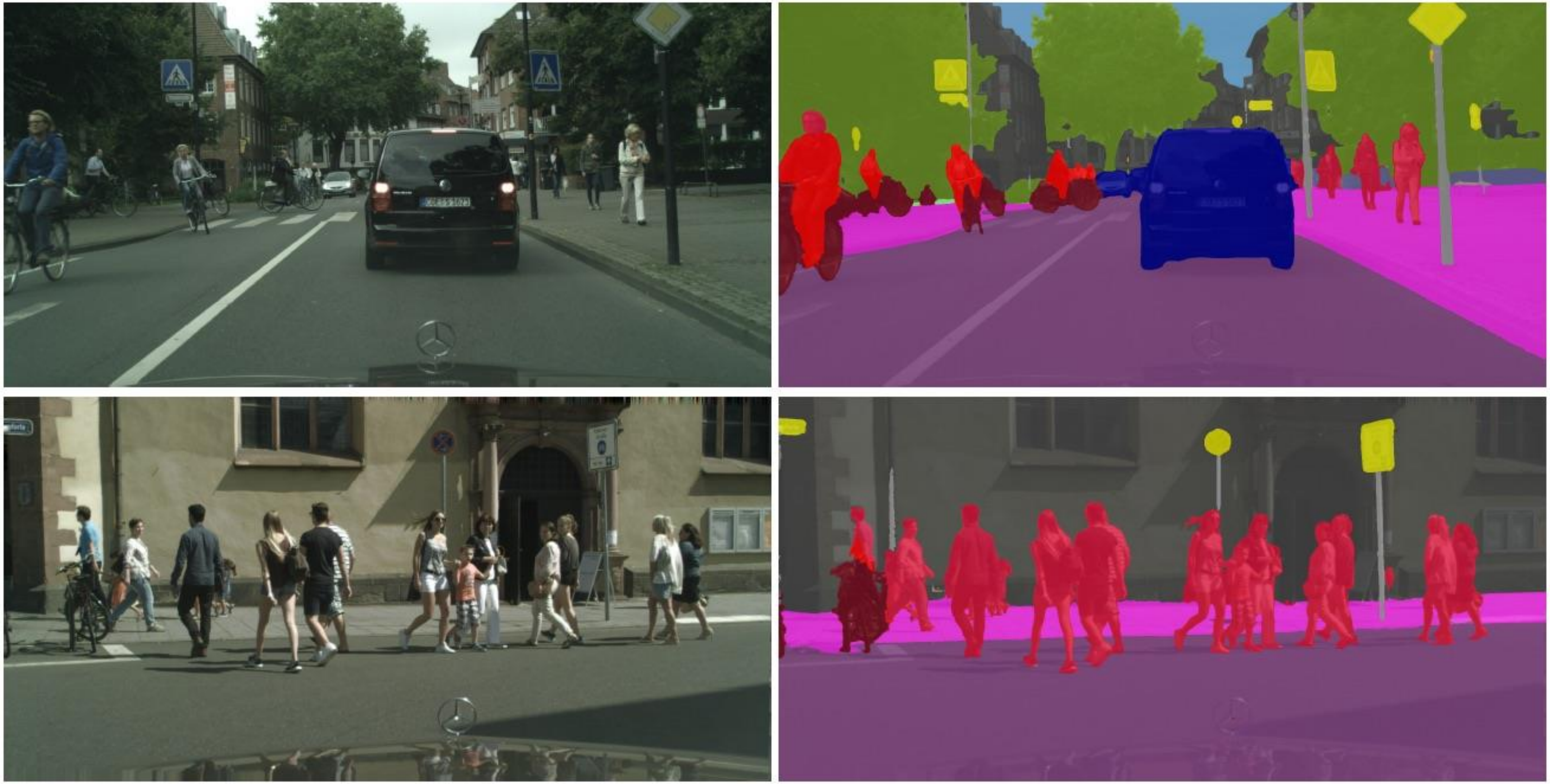


(object detection)

(object segmentation)

(dzone.com)

Semantic Segmentation = Pixel-wise Classification



(vladlen.info)

Igor Vustianiuk, 2018

Loss Functions – 1

Bounding box	Facial landmarks	Pose estimation
 (coursera.org)	 (stackoverflow.com)	 (cs231n.stanford.edu)
(c_x, c_y, w, h)	$(x_1, y_1, \ldots, x_M, y_M)$	$(x_1, y_1, \ldots, x_M, y_M)$

$$\frac{1}{m}\sum_{i=1}^{m}\left(\sum_{j=1}^{n} L(s_j,\ \hat{s}_j)\right) \rightarrow \quad \min$$

$$L(s_j,\ \hat{s}_j) = \frac{1}{2}(s_j - \hat{s}_j)^2$$

Loss Functions – 2

loss function:

$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \quad (3)$$

where $\mathbb{1}_i^{\text{obj}}$ denotes if object appears in cell i and $\mathbb{1}_{ij}^{\text{obj}}$ denotes that the jth bounding box predictor in cell i is "responsible" for that prediction.

(https://arxiv.org/pdf/1506.02640.pdf)

Igor Vustianiuk, 2018

Loss Functions – 3. Categorical cross-entropy

$$\hat{y} = s \qquad \text{vs} \qquad y = j$$

$$\mathbf{p} = (p_1, \dots, p_K) \qquad \text{vs} \qquad \mathbf{y} = \left(0, \dots, \underbrace{1}_{j}, \dots, 0\right)$$

one-hot encoding

$$P(\mathbf{y}) = p_j = p_1^0 \dots p_j^1 \dots p_K^0 = p_1^{y_1} \dots p_K^{y_K}$$

$$p_1^{y_1} \dots p_K^{y_K} \to \max$$

$$y_1 \ln p_1 + \cdots + y_K \ln p_K \to \max$$

$$0 < p_k < 1 \quad \Longrightarrow \quad \ln p_k < 0 \quad \Longrightarrow \quad -\ln p_k > 0$$

$$-y_1 \ln p_1 - \cdots - y_K \ln p_K \to \min$$

$$\frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} -y_k^{(i)} \ln p_k^{(i)} \to \min$$

Three Main Types of Classification

- **Binary**: **two** mutually exclusive classes (object belongs to **one** class)

$$y^{(i)} = 1 \qquad p^{(i)} = 0.7$$

$$-y^{(i)} \ln p^{(i)} - \left(1 - y^{(i)}\right) \ln\left(1 - p^{(i)}\right)$$

- **Multi-class**: any number of mutually exclusive (object belongs to **one** class)

$$y^{(i)} = [\;0 \quad 1 \quad 0\;] \qquad p^{(i)} = [0.3 \;\; 0.6 \;\; 0.1]$$

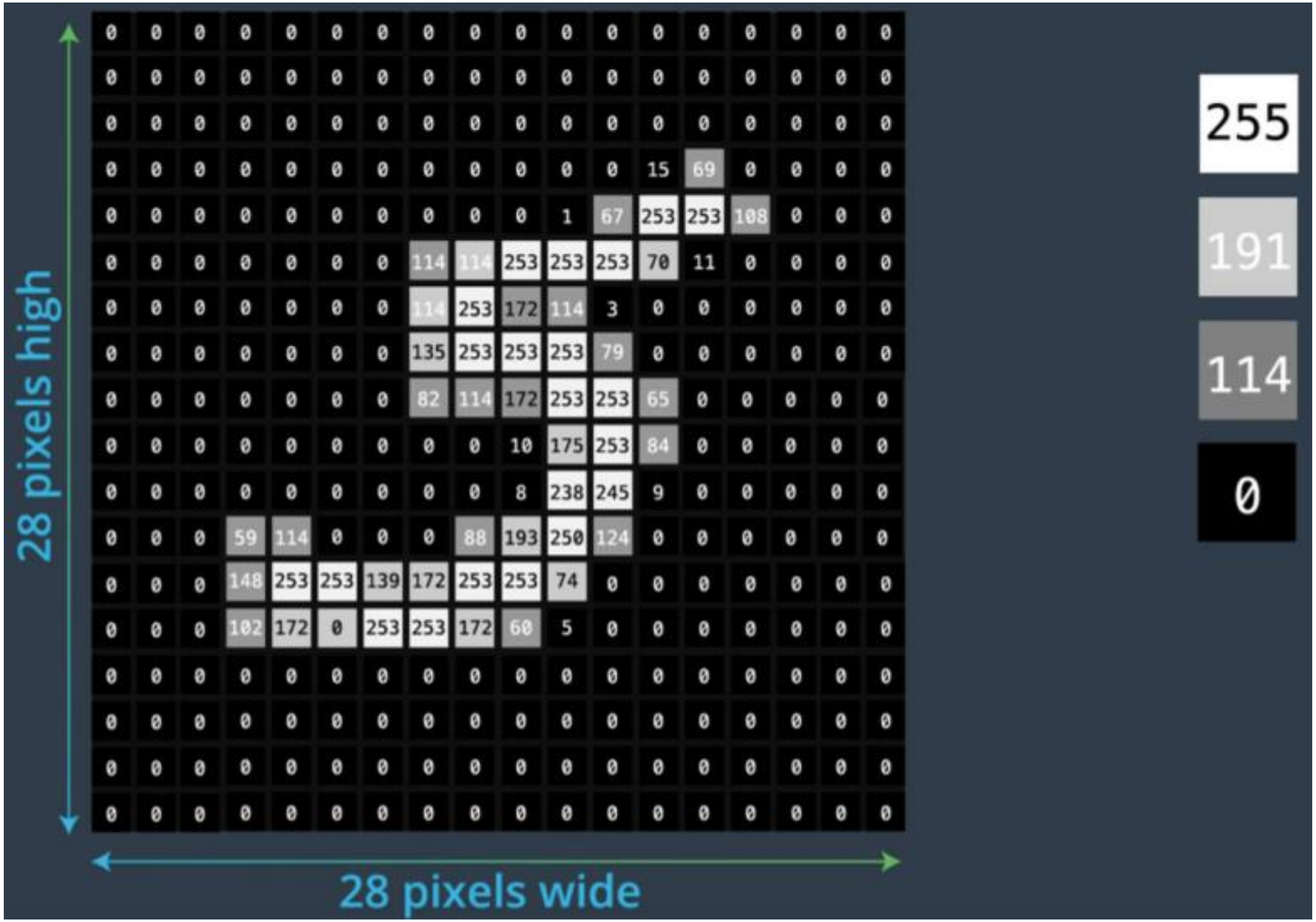
$$\sum_{k=1}^{K} -y_k^{(i)} \ln p_k^{(i)}$$

- **Multi-label**: any number of independent classes (object can belong to **many** classes)

$$y^{(i)} = [\;1 \quad 1 \quad ? \quad 0\;] \qquad p^{(i)} = [0.5 \;\; 0.7 \;\; 0.4 \;\; 0.2]$$

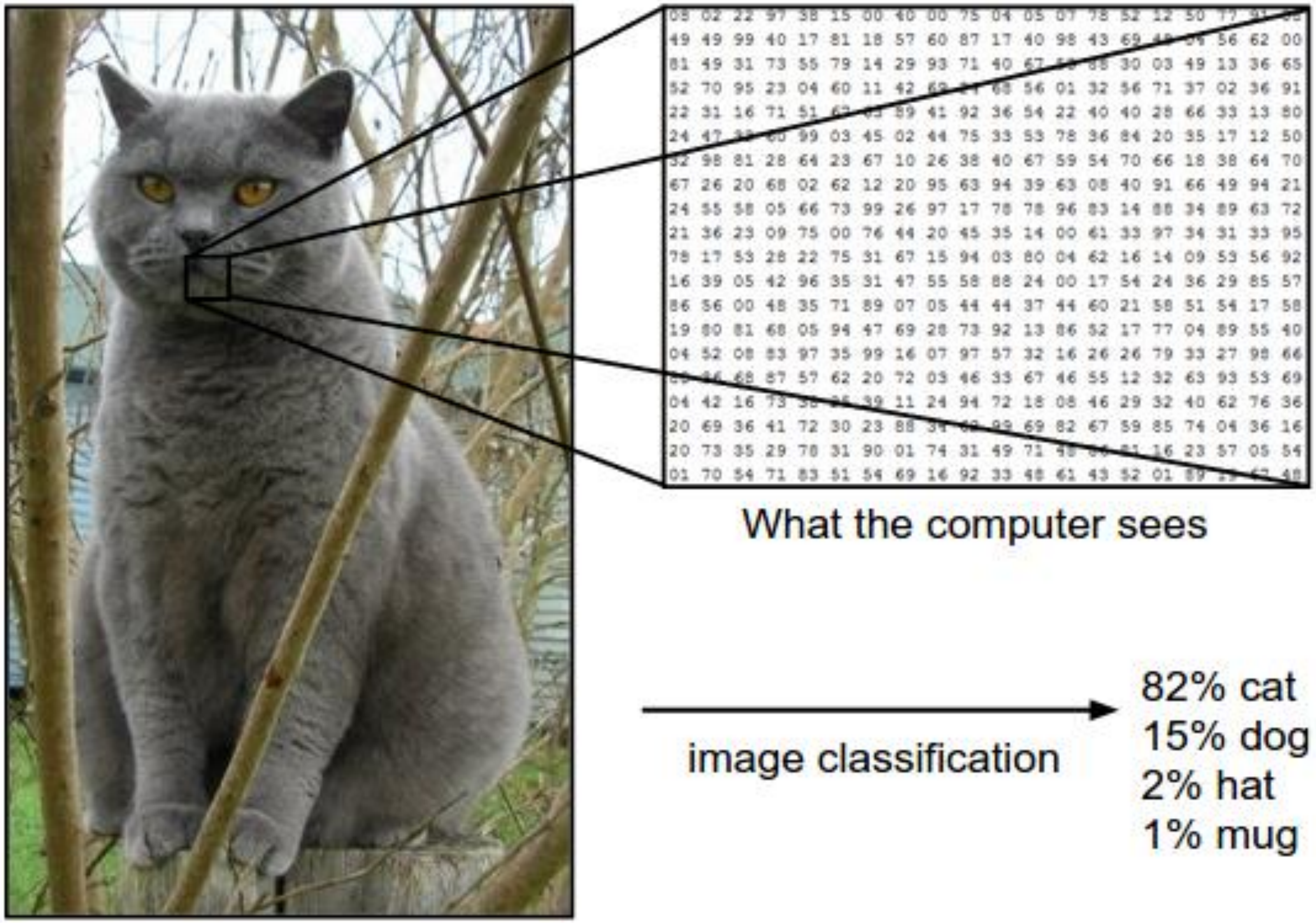
$$\sum_{\substack{k=1 \\ y_k^{(i)} \neq ?}}^{K} \left[-y_k^{(i)} \ln p_k^{(i)} - \left(1 - y_k^{(i)}\right) \ln\left(1 - p_k^{(i)}\right)\right]$$

What is an image – 1



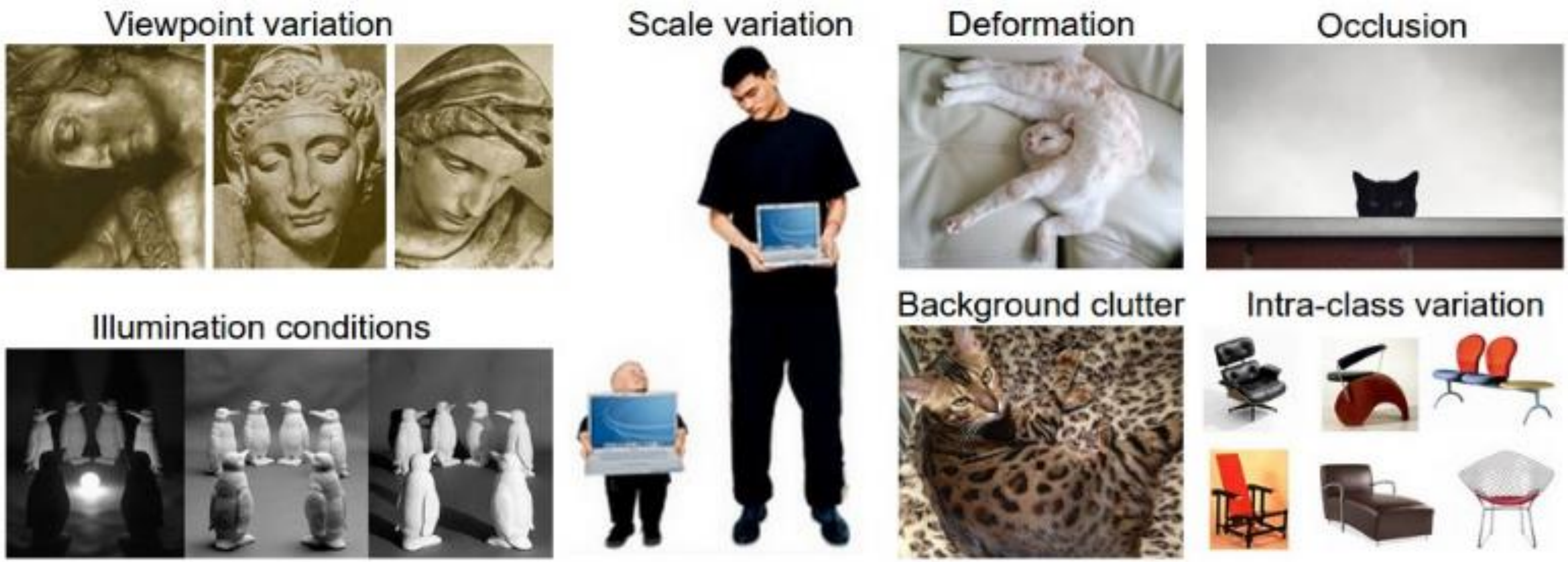
(Udacity)

Why images are hard? – 1



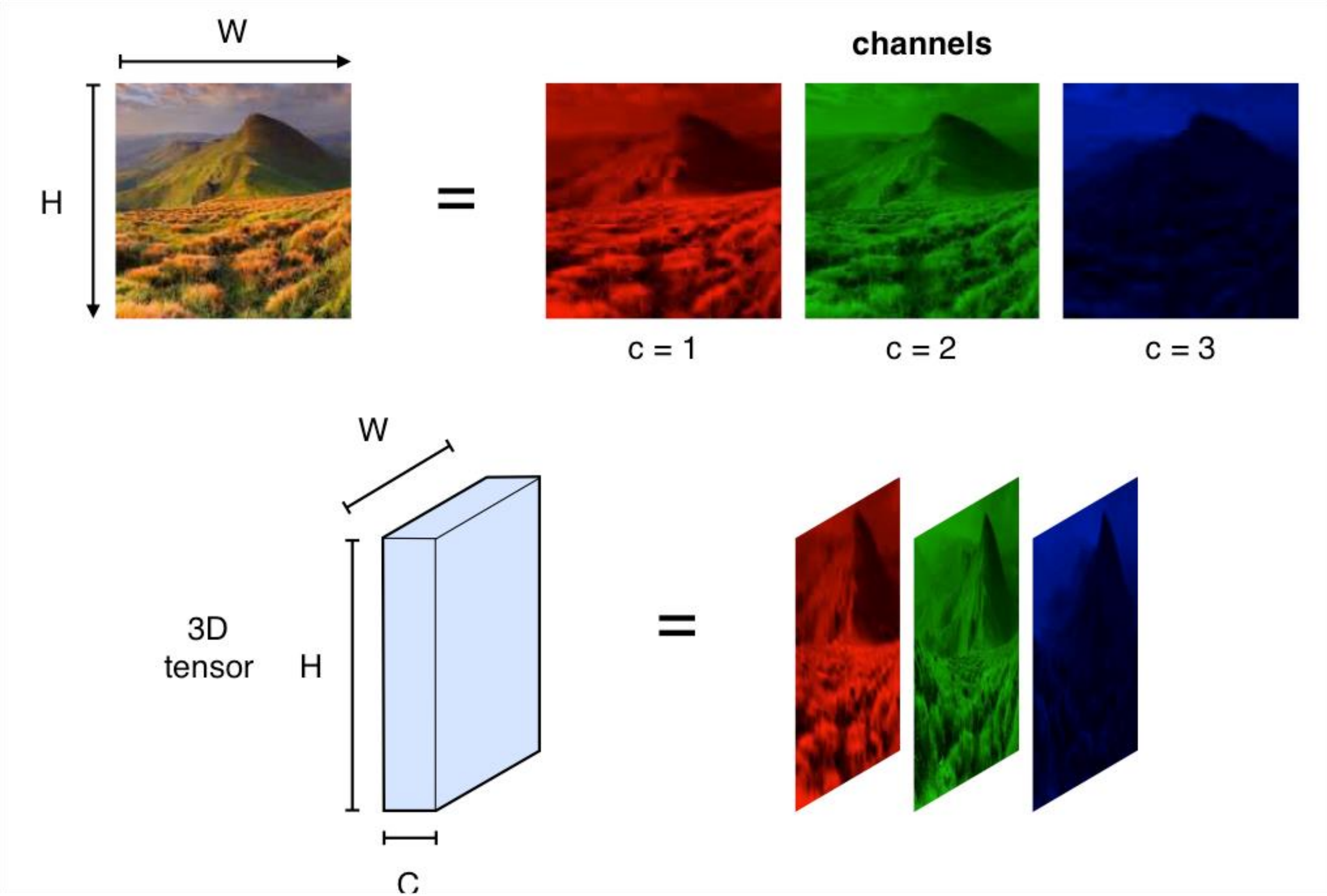
(cs231n.github.io)

Why images are hard? – 2



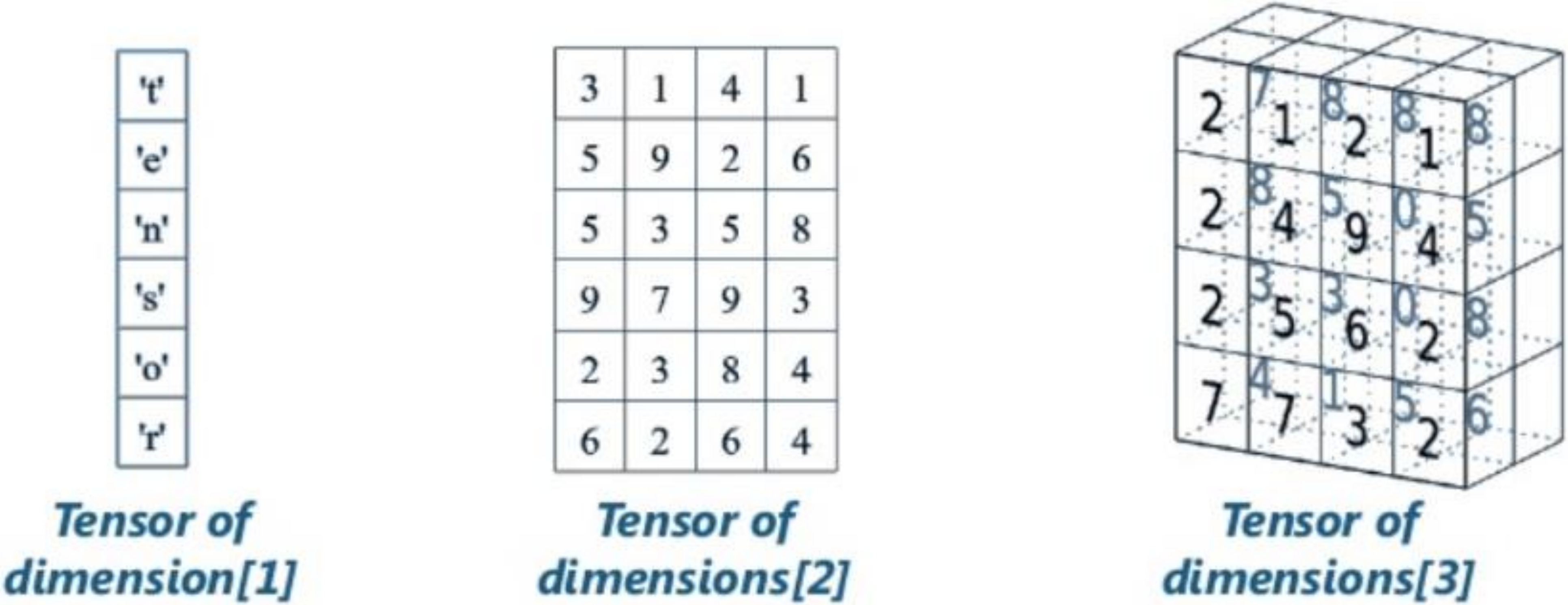
(cs231n.github.io)

What is an image – 2



(www.di.ens.fr)

Meet the Tensors



(Edureka)

Common Operations on Tensors

- Extending scalar binary operations:

$$(A \star B)_\alpha = A_\alpha \star B_\alpha$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 2 & 0 \\ 6 & 6 \end{bmatrix} = \begin{bmatrix} 3 & 2 \\ 9 & 10 \end{bmatrix}$$

- Applying scalar functions:

$$g\colon\ \mathbb{R} \to \mathbb{R}, \qquad \big(g(A)\big)_\alpha = g(A_\alpha)$$

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \qquad \sigma\left(\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}\right) = \begin{bmatrix} 0.5000 & 0.7311 \\ 0.8808 & 0.9526 \end{bmatrix}$$

- Flatten (reshaping to 1d-tensor):

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$$

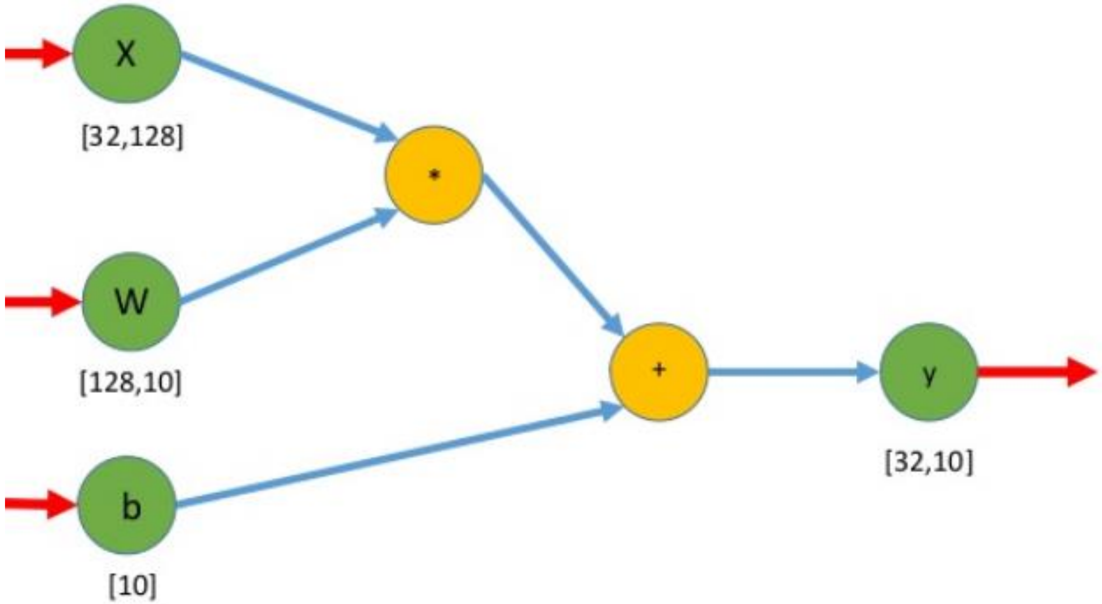
- Concatenation:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$
$$\begin{bmatrix} 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 & 7 & 8 & 9 \\ 4 & 5 & 6 & 10 & 11 & 12 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}$$

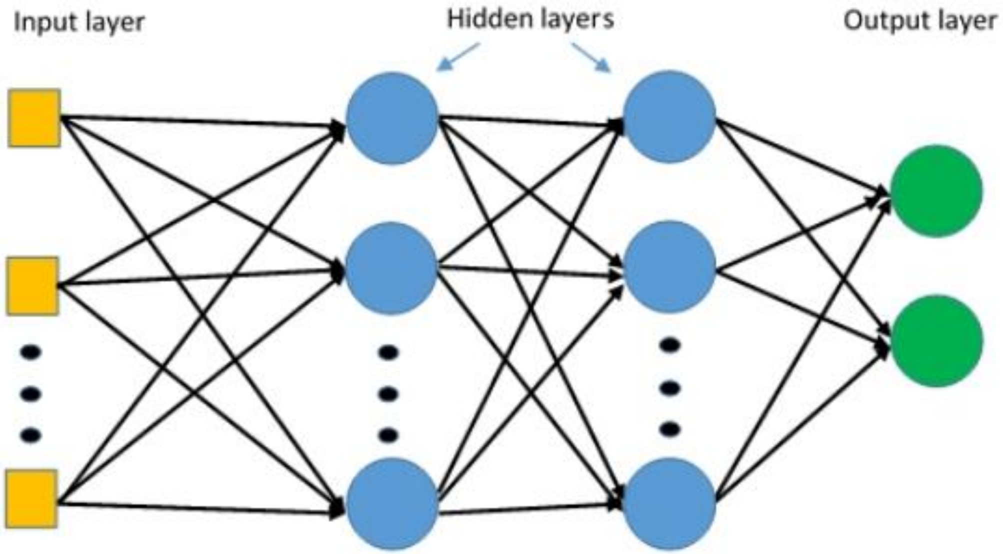
Modern Neural Network (MLP, CNN, RNN) == Graph of Tensors



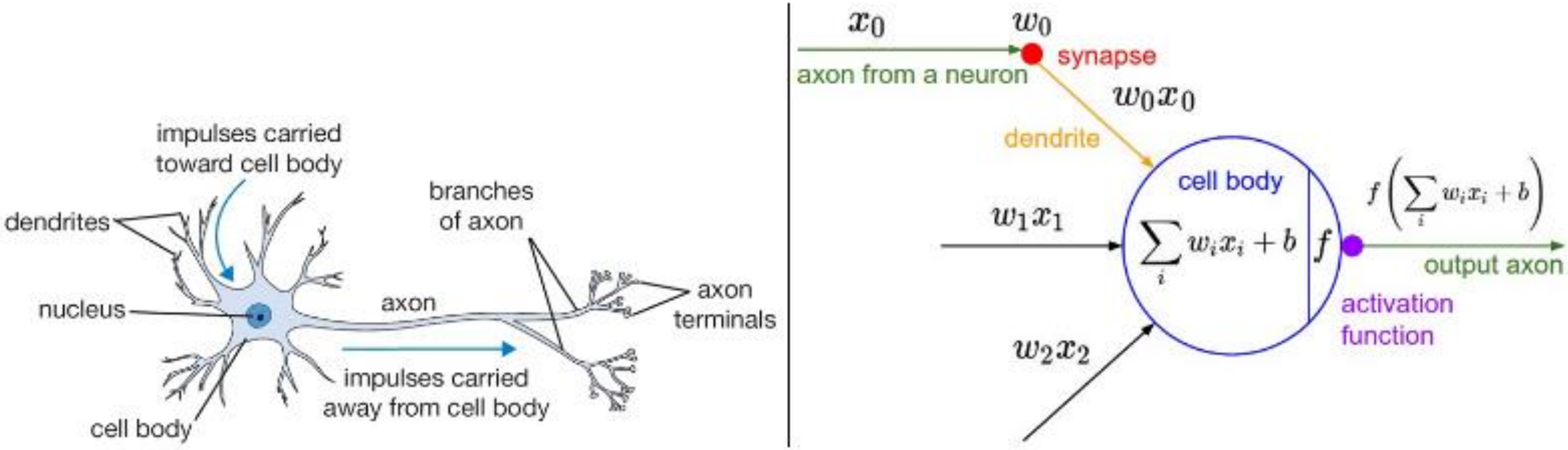
(Alessio Tonioni on slideshare.net)

$$Forward(\theta) = f\big(Parents(\theta)\big)$$

$$Backward(\theta) = \frac{\partial E}{\partial \theta} = \sum_{c \in Children(\theta)} \left(\frac{\partial E}{\partial c} \cdot \frac{\partial c}{\partial \theta}\right)$$



(packtpub.com)

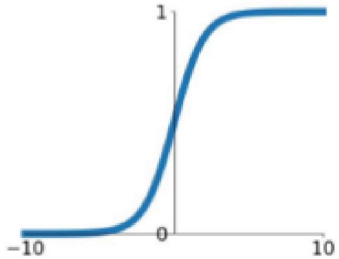


(cs231n.github.io)

Activation Functions

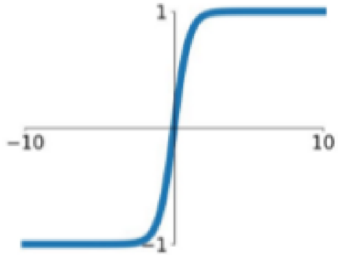
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



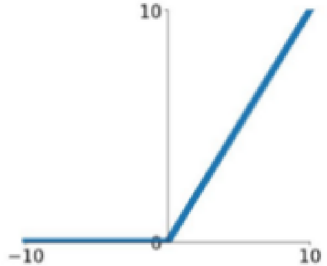
tanh

$\tanh(x)$



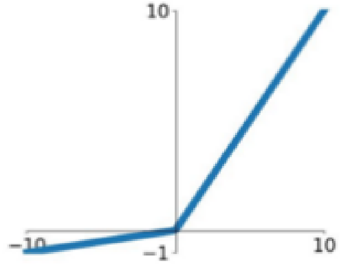
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

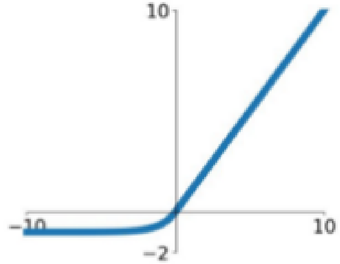


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



(Shruti Jadon on Medium)

arXiv.org > cs > arXiv:1710.05941

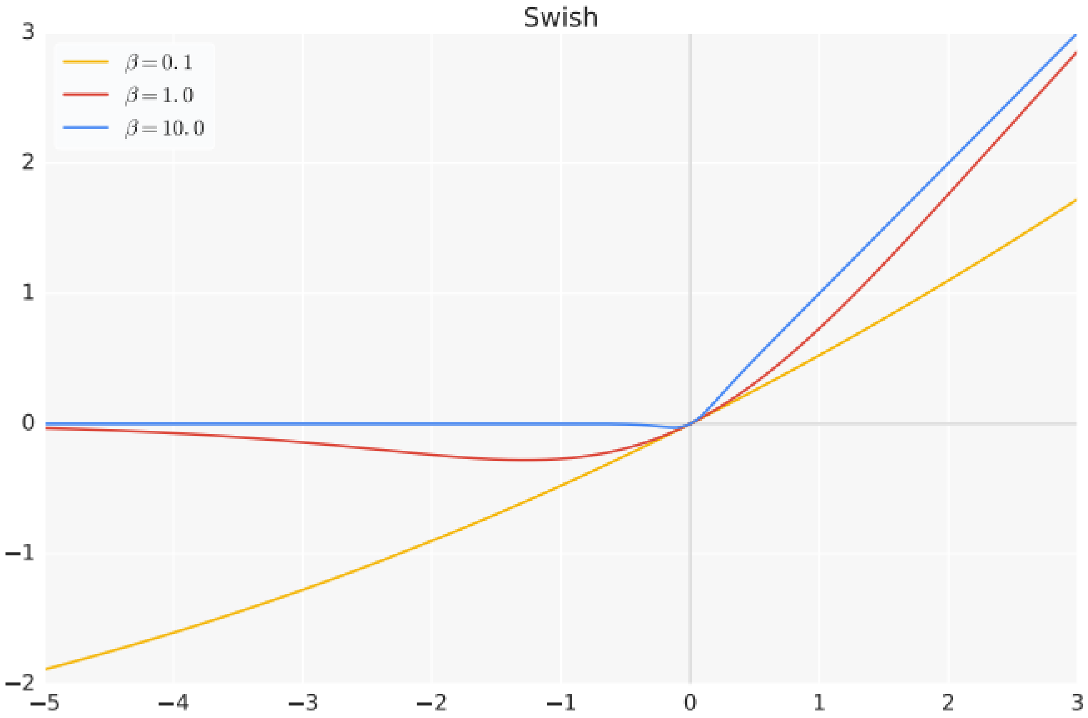
Computer Science > Neural and Evolutionary Computing

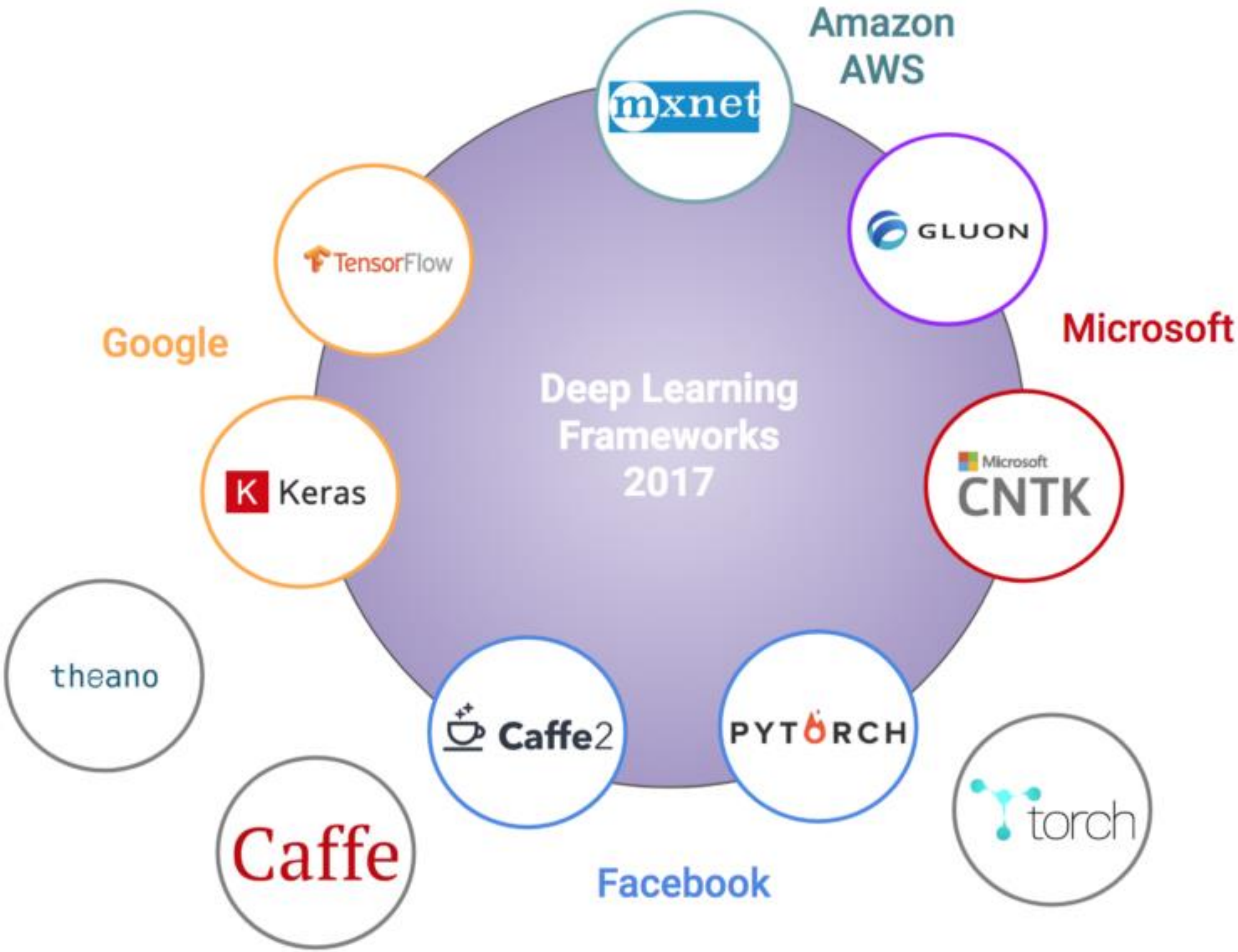
Searching for Activation Functions

Prajit Ramachandran, Barret Zoph, Quoc V. Le

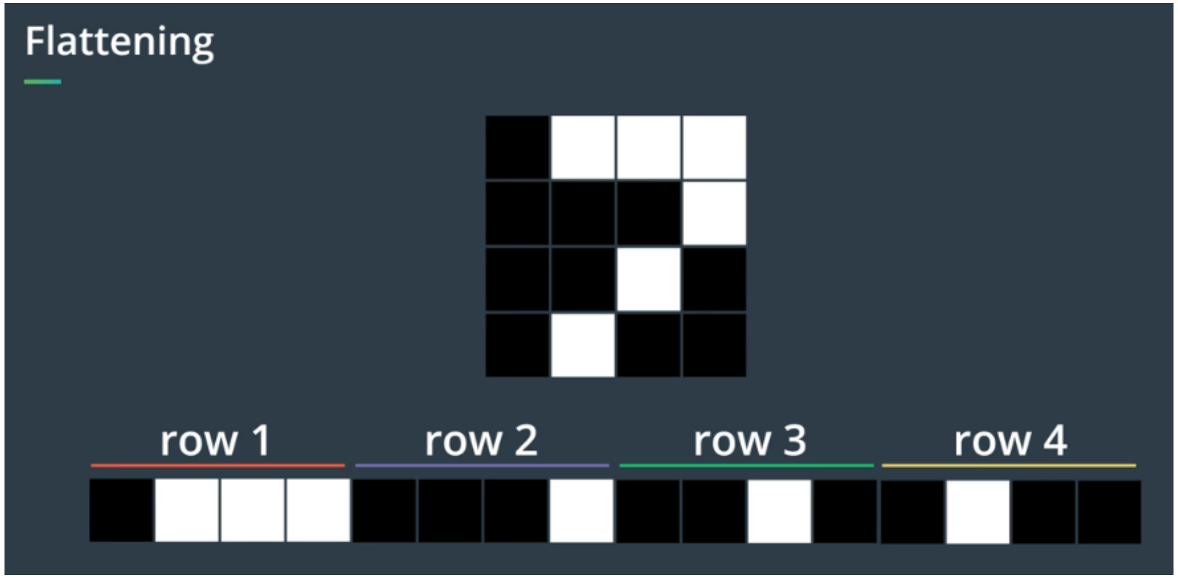
(Submitted on 16 Oct 2017 (v1), last revised 27 Oct 2017 (this version, v2))

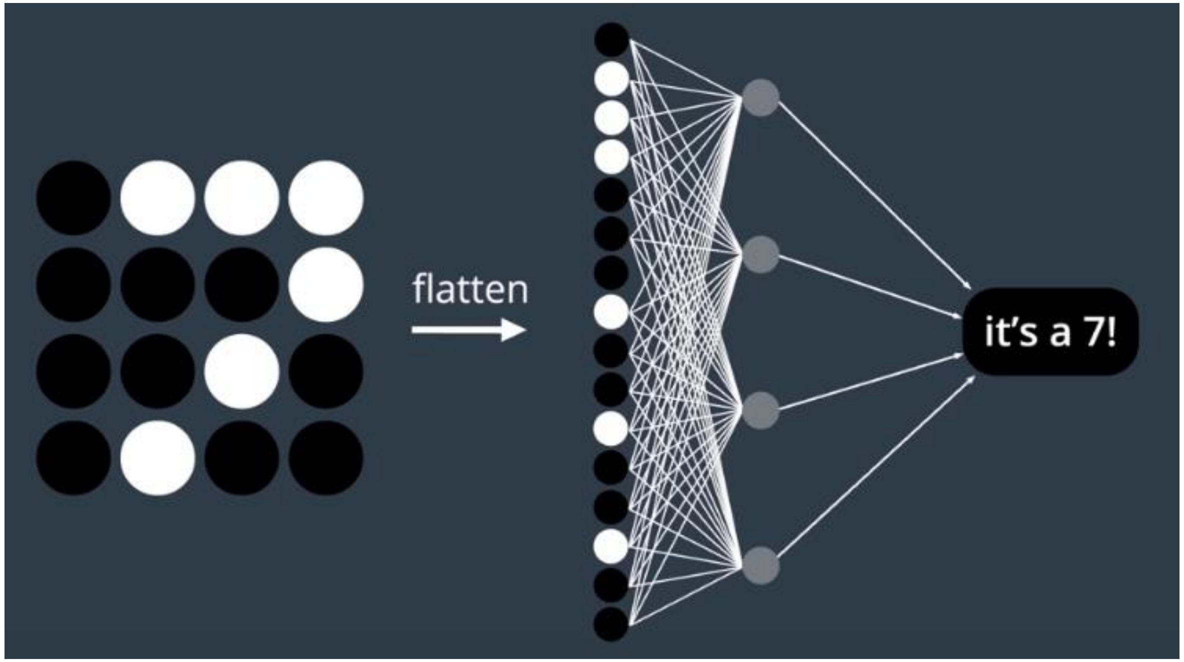
$$\mathrm{Swish}(x) = x \cdot \sigma(\beta x) = \frac{x}{1 + e^{-\beta x}}$$





(towardsdatascience.com)





(pictures with gray background were taken from Udacity)

Igor Vustianiuk, 2018

Softmax – 1

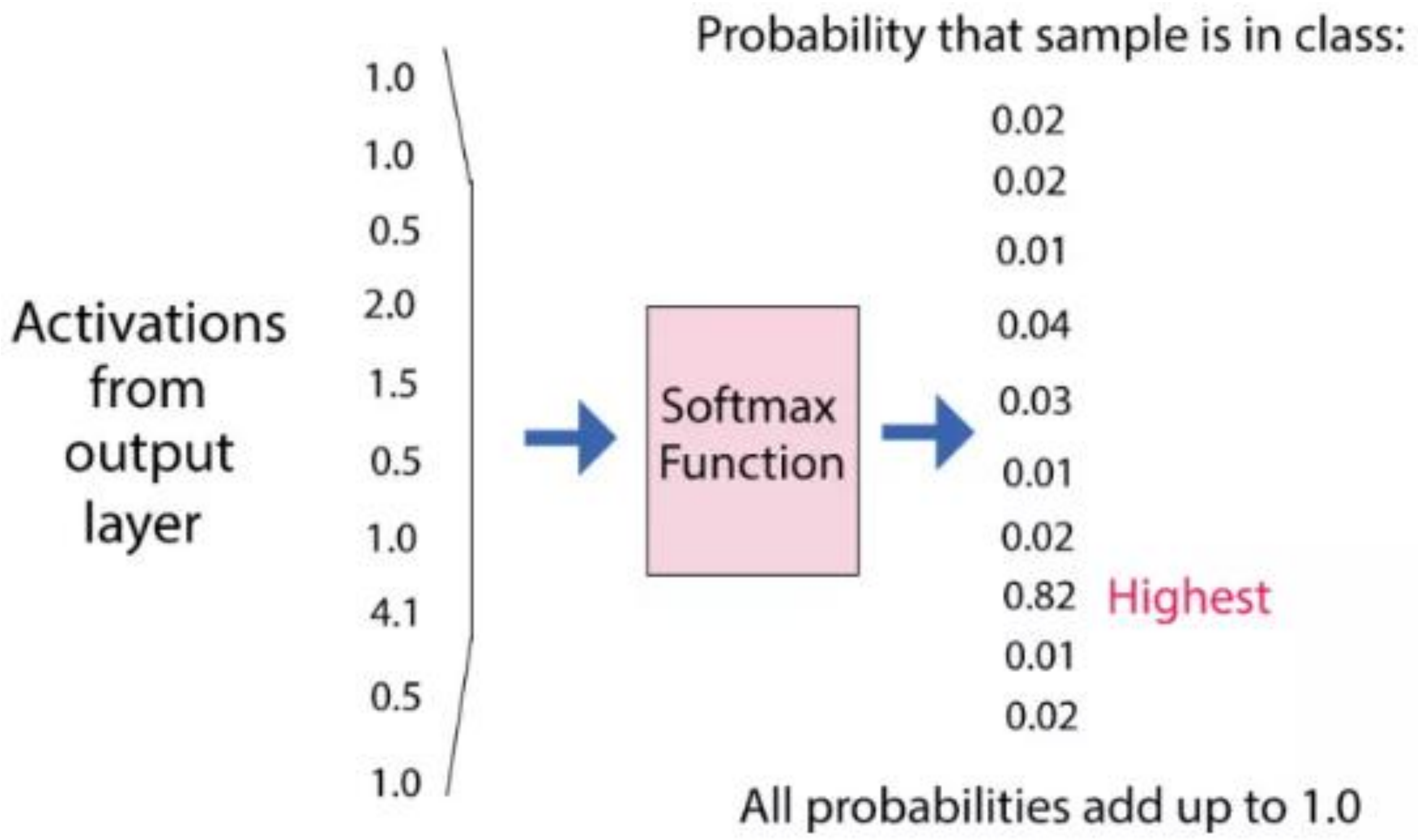
$$\begin{bmatrix} z_1 \\ \vdots \\ z_K \end{bmatrix} \longmapsto \begin{bmatrix} \dfrac{z_1}{\sum e^{z_i}} \\ \vdots \\ \dfrac{z_K}{\sum z_i} \end{bmatrix}, \qquad \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix} \longmapsto \begin{bmatrix} 0.5 \\ -0.5 \\ 1.0 \end{bmatrix}$$

$$\begin{bmatrix} z_1 \\ \vdots \\ z_K \end{bmatrix} \longmapsto \begin{bmatrix} e^{z_1} \\ \vdots \\ e^{z_K} \end{bmatrix} \longmapsto \begin{bmatrix} \dfrac{e^{z_1}}{\sum e^{z_i}} \\ \vdots \\ \dfrac{e^{z_K}}{\sum e^{z_i}} \end{bmatrix} = \begin{bmatrix} p_1 \\ \vdots \\ p_K \end{bmatrix}$$

$$0 < p_i < 1, \qquad \sum_{i=1}^{K} p_i = 1$$

$$z_i < z_j \qquad \Longrightarrow \qquad p_i < p_j$$

Softmax – 2



(http://principlesofdeeplearning.com)

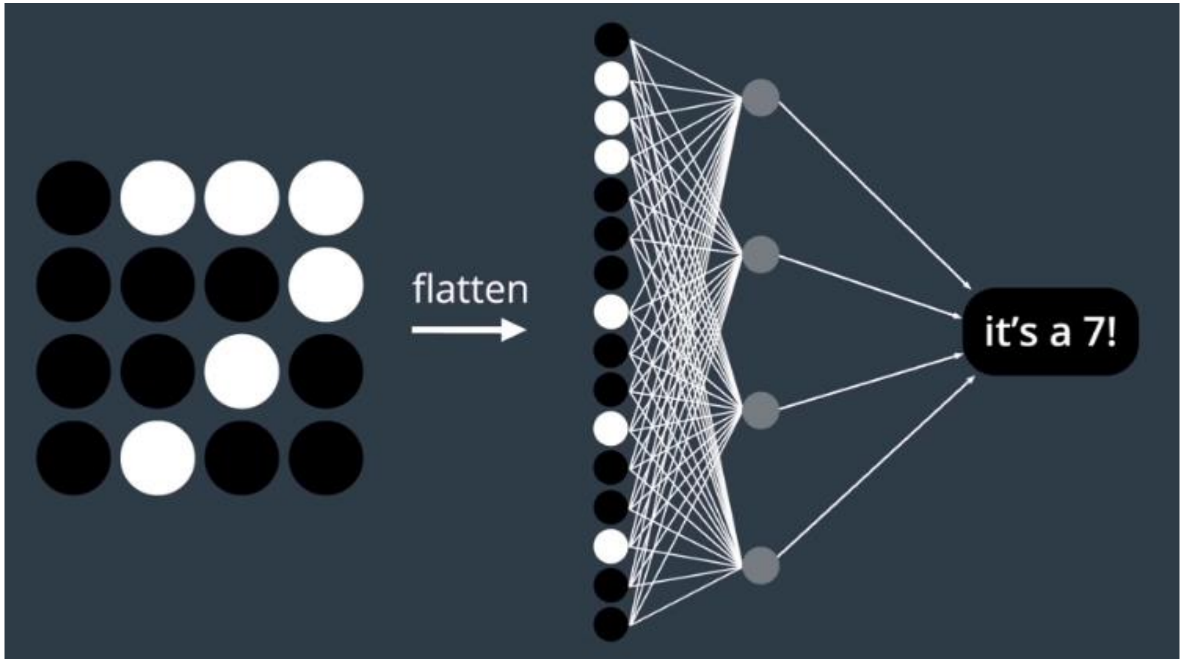
```python
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
model.summary()
```

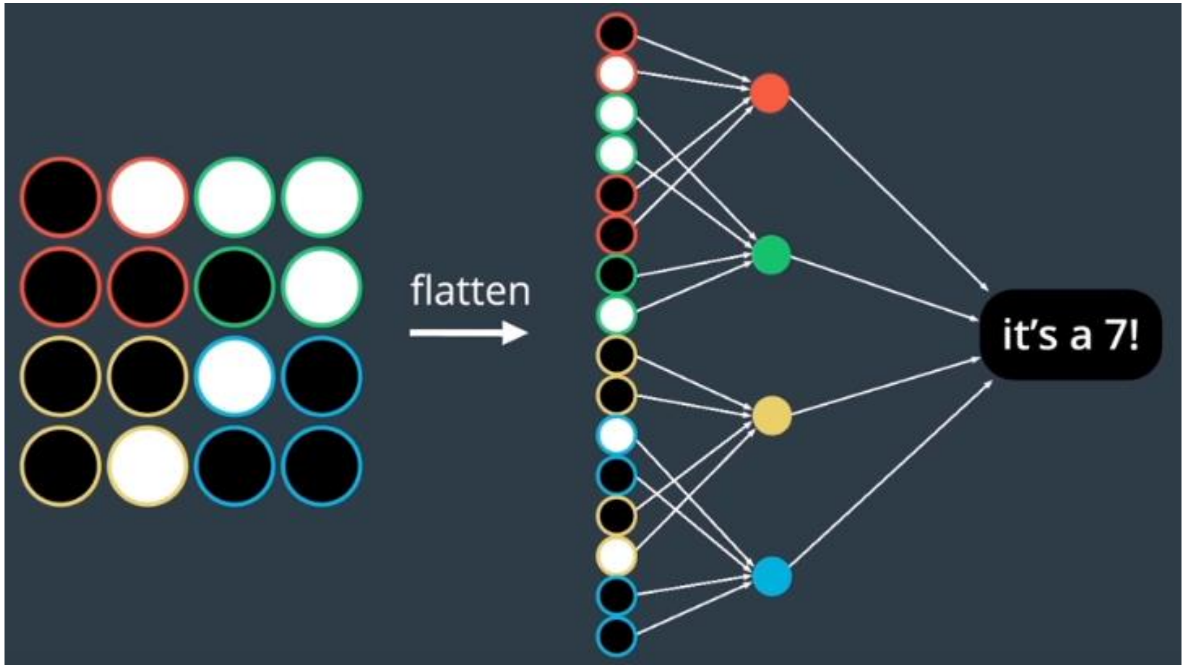
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 512)               401920
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                5130
=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
```

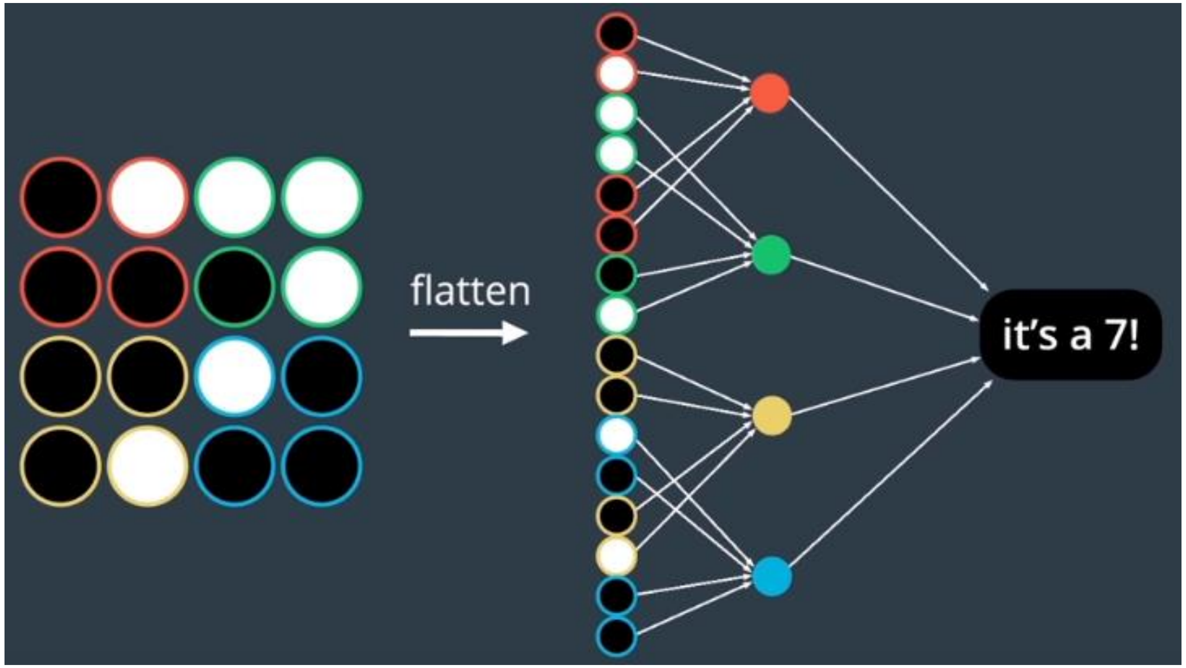
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 512)               401920
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                5130
=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
```

Input	$28 \cdot 28 = 784$
Hidden 1 (dense_1)	$(784 + 1) \cdot 512 = 401\,920$
Hidden 2 (dense_2)	$(512 + 1) \cdot 512 = 262656$
Output (dense_3)	$(512 + 1) \cdot 10 = 5130$

Igor Vustianiuk, 2018

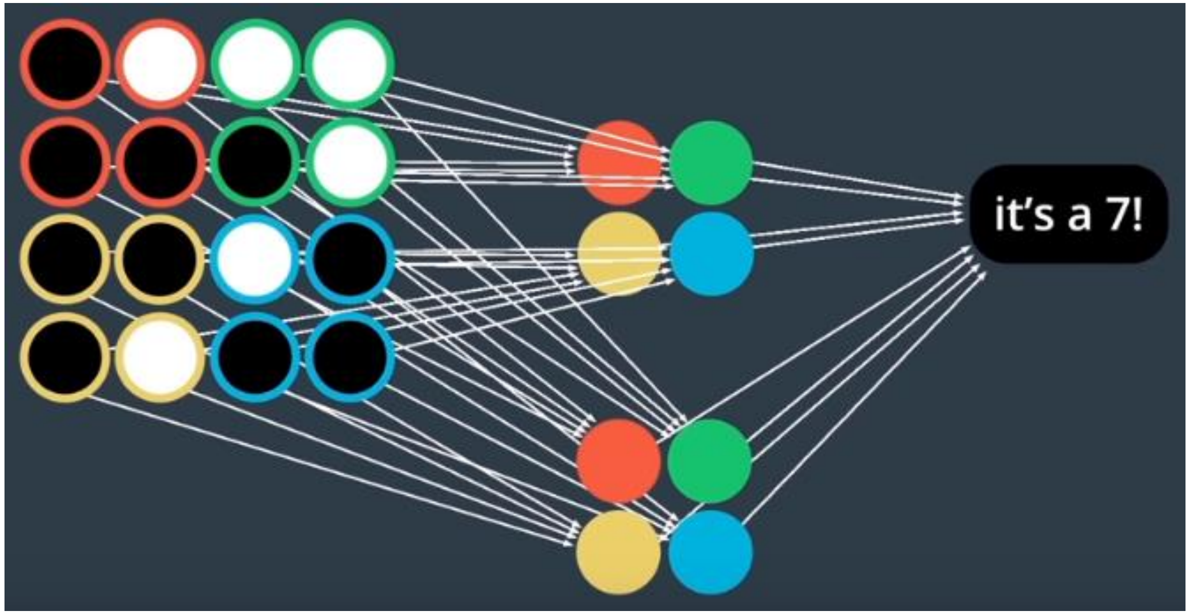





flatten
it's a 7!


it's a 7!


it's a 7!

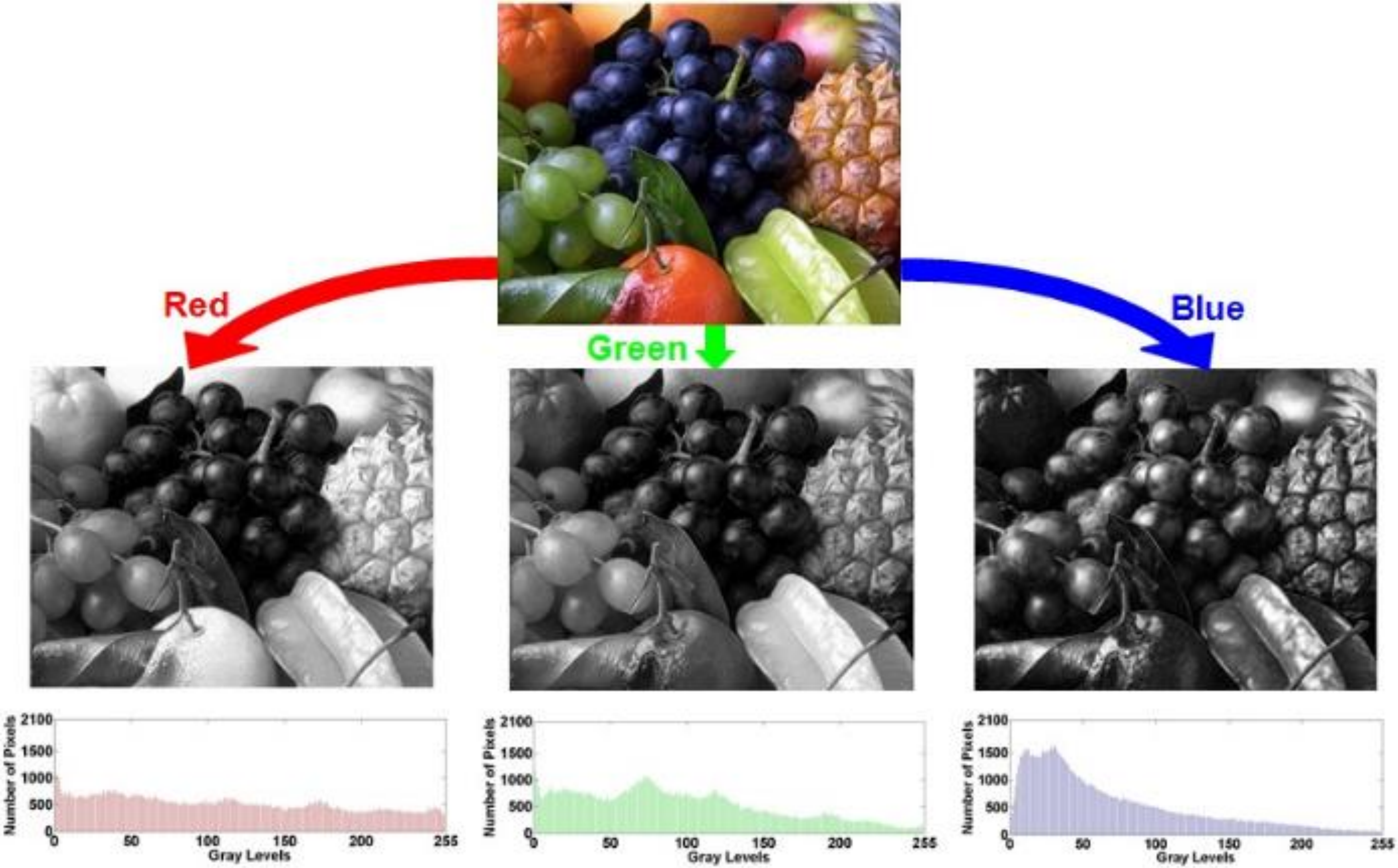

it's a 7!

Feature is a specific 2D structure in the image such as a blob, corner or an edge than can be described in a local neighborhood by its appearance information.

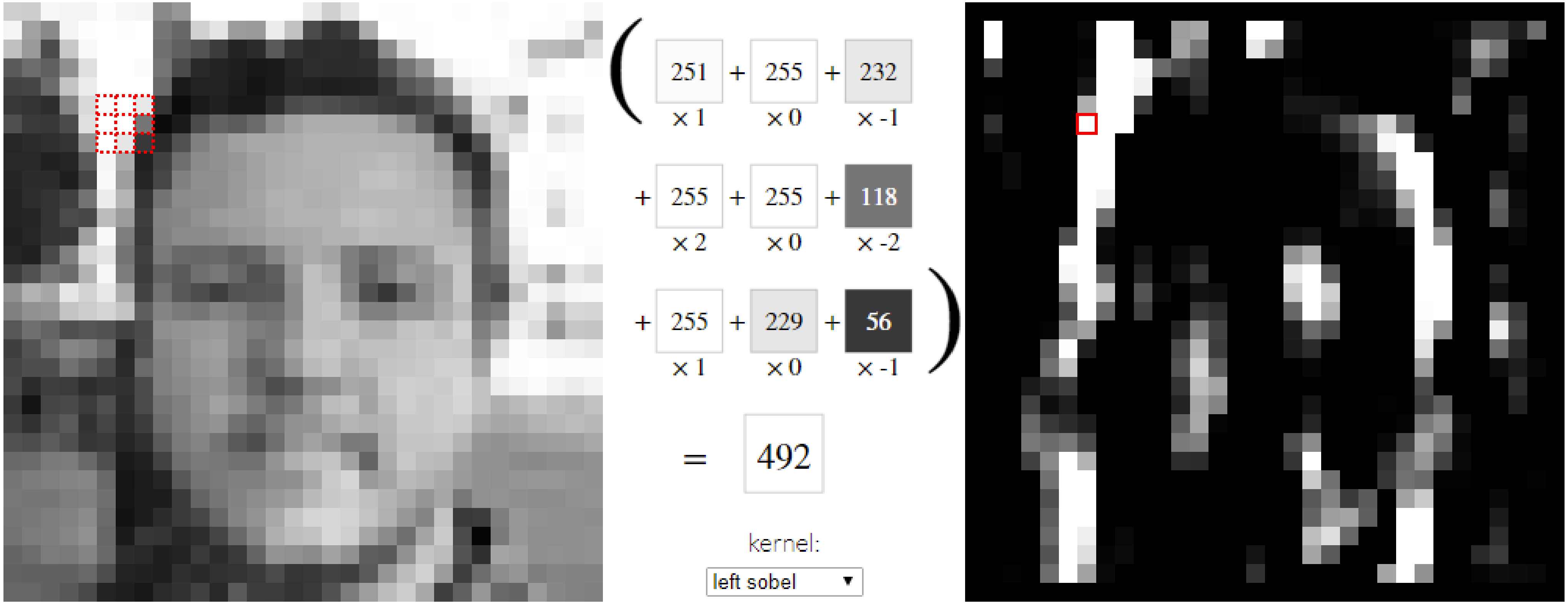
Descriptor is a vector that contains local appearance information.

(Pablo F. Alcantarilla)

(http://cis.eecs.qmul.ac.uk)

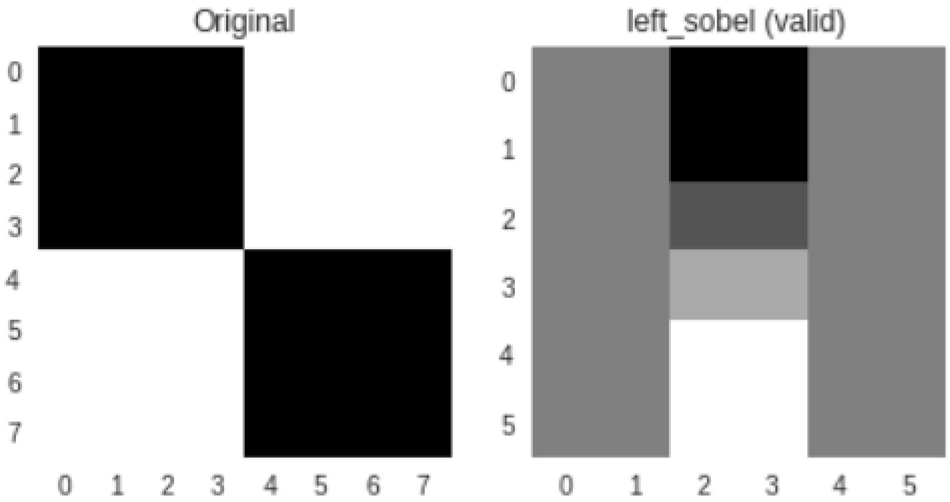


(allaboutcircuits.com)

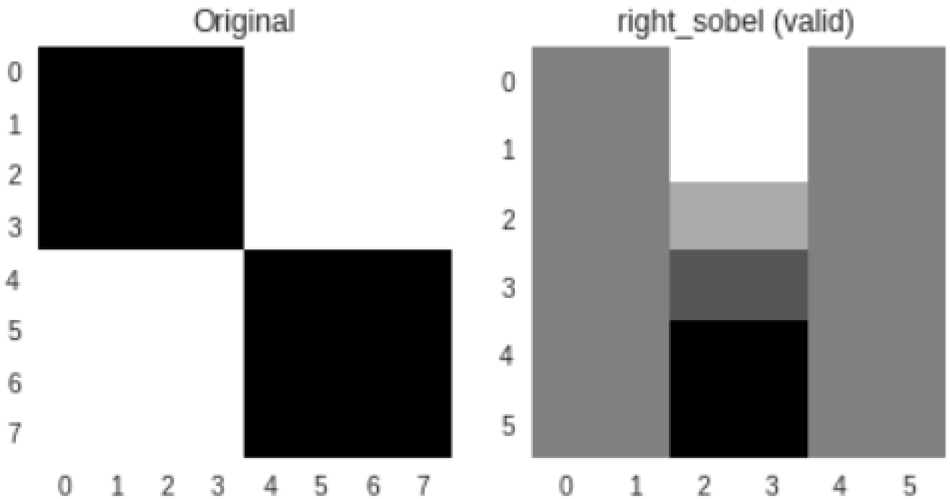


251 × 1 + 255 × 0 + 232 × -1
+ 255 × 2 + 255 × 0 + 118 × -2
+ 255 × 1 + 229 × 0 + 56 × -1
= 492
kernel:
left sobel

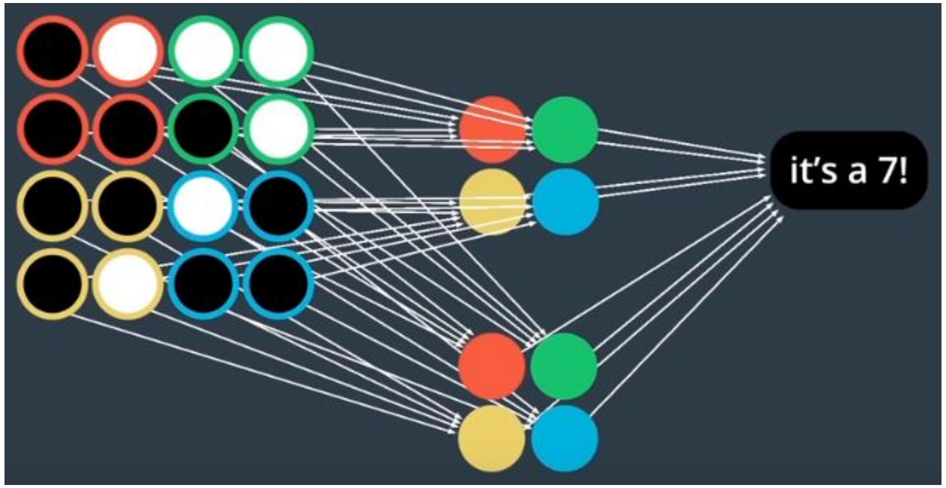
(setosa.io)



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



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



(Udacity)

- **Parameter sharing**. A feature detector (such as vertical edge detector) that's useful in one part on the image is *probably* useful in another part of the image.

- **Sparsity of connections**. In each layer each output value depends on a small number of inputs.

(Coursera)