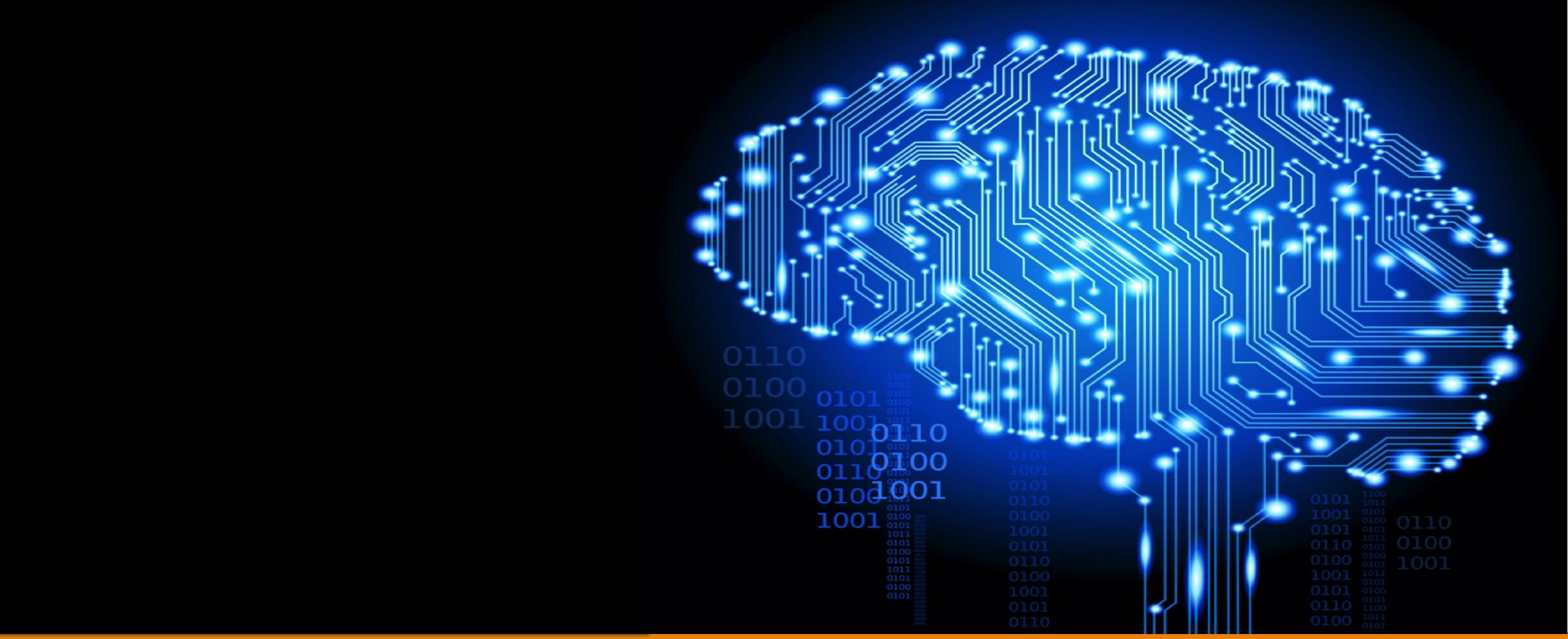


Lecture 4: Convolutional Neural Networks

Deep Learning @ UvA

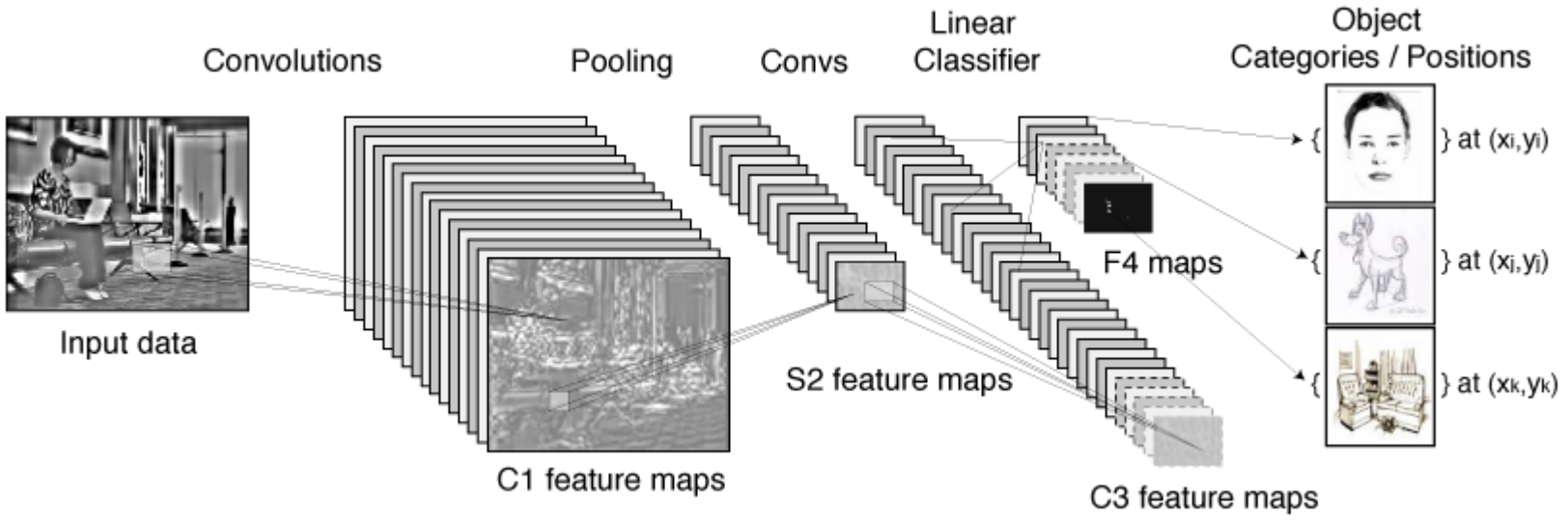
Previous lecture

- How to define our model and optimize it in practice
- Data preprocessing and normalization
- Optimization methods
- Regularizations
- Architectures and architectural hyper-parameters
- Learning rate
- Weight initializations
- Good practices

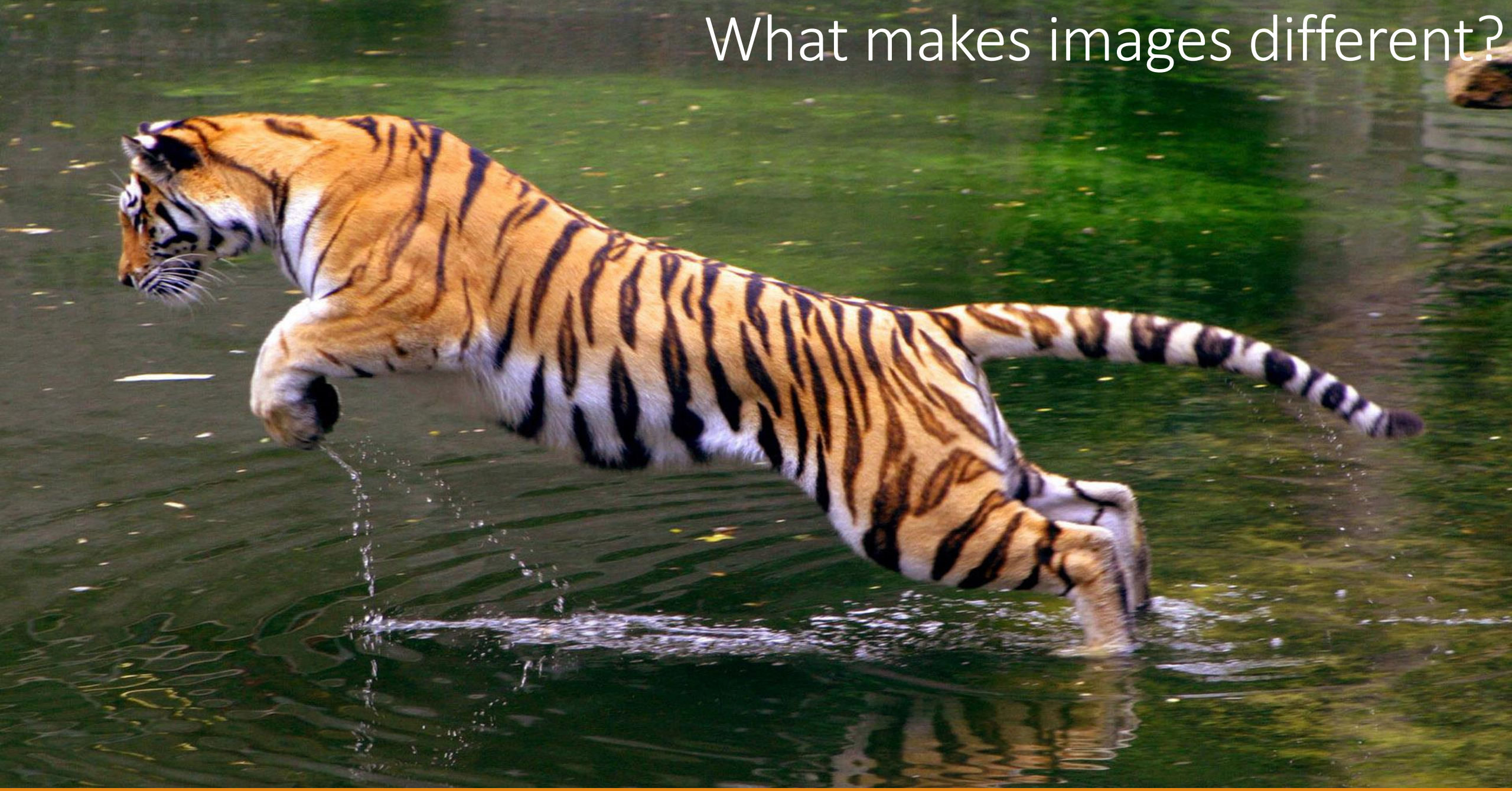
Lecture overview

- What are the Convolutional Neural Networks?
- Differences from standard Neural Networks
- Why are they important in Computer Vision?
- How to train a Convolutional Neural Network?

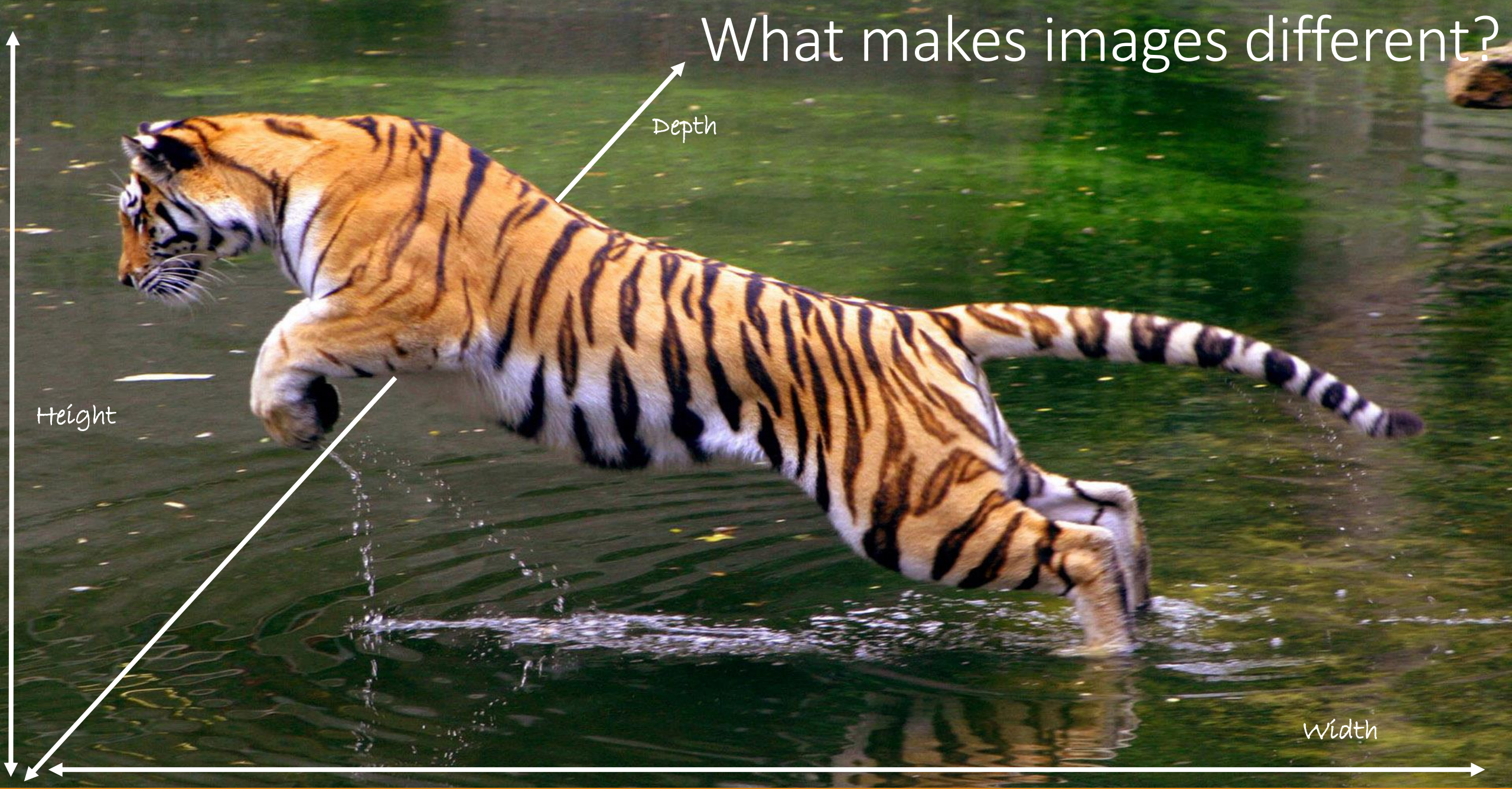
Convolutional Neural Networks



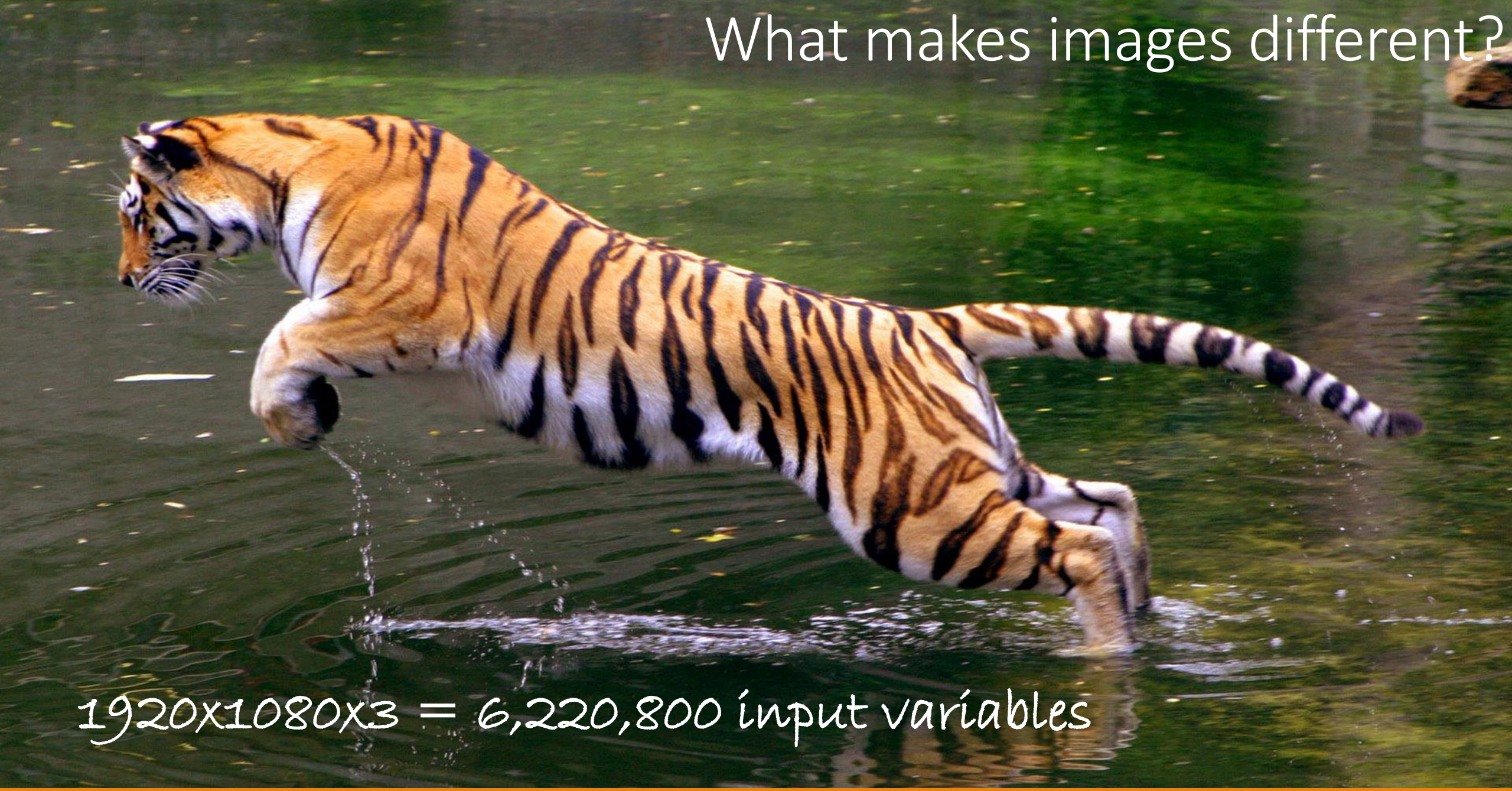
What makes images different?



What makes images different?

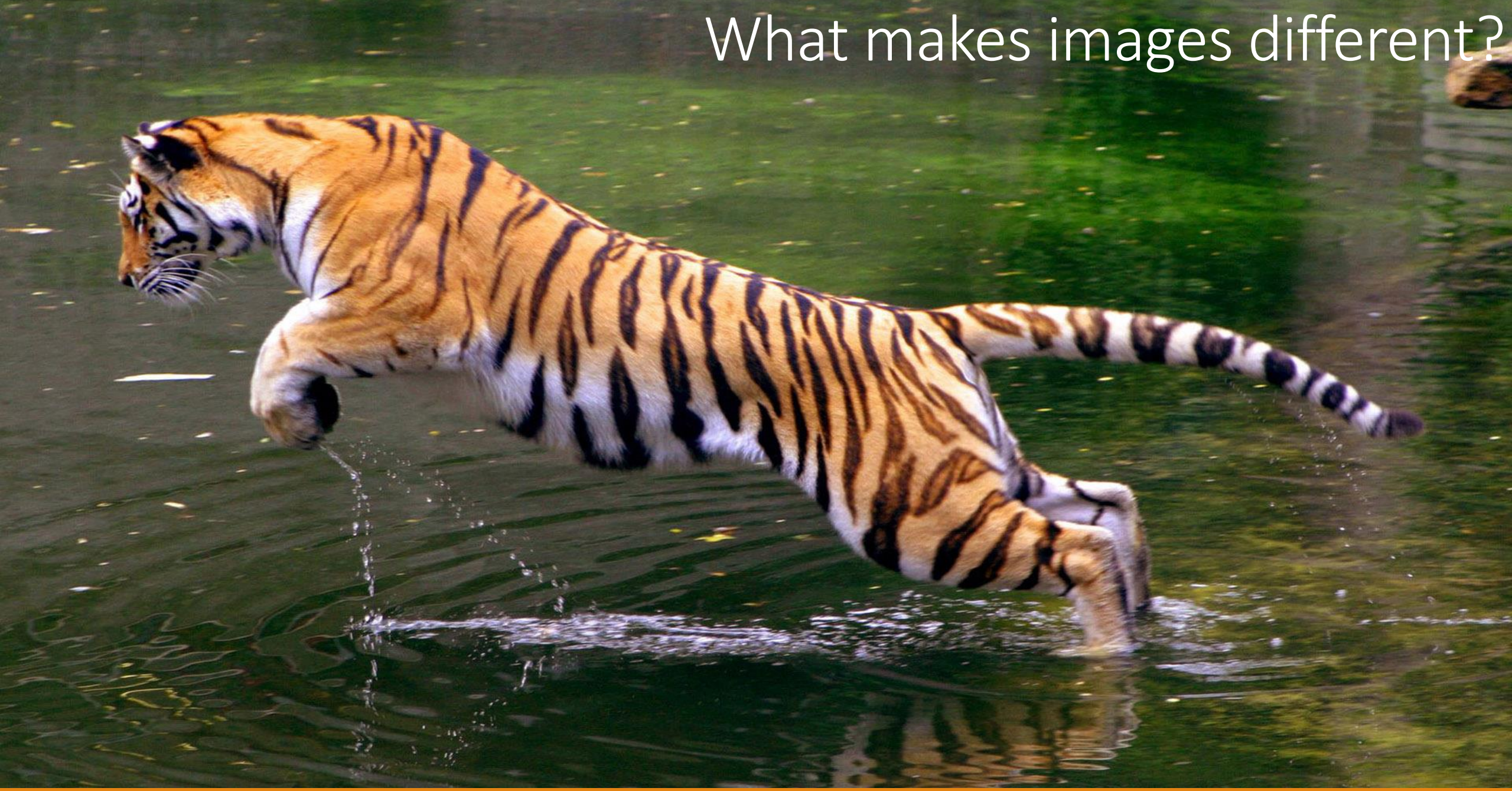


What makes images different?



What makes images different?

What makes images different?



What makes images different?

What makes images different?

- An image has spatial structure
- Huge dimensionality
  - A 256x256 RGB image amounts to ~200K input variables
  - 1-layered NN with 1,000 neurons → 200 million parameters
- Images are stationary signals → they share features
  - After variances images are still meaningful
  - Small visual changes (often invisible to naked eye) → big changes to input vector
  - Still, semantics remain
  - Basic natural image statistics are the same

Input dimensions are correlated

Traditional task: Predict my salary!

Shift 1 dimension

Level of education	Age	Years of experience	Previous job	Nationality
"Higher"	28	6	Researcher	Spain

Level of education	Age	Years of experience	Previous job	Nationality
Spain	"Higher"	28	6	Researcher

Vision task: Predict the picture!

First 5x5 values

```
array([[51, 49, 51, 56, 55],
       [53, 53, 57, 61, 62],
       [67, 68, 71, 74, 75],
       [76, 77, 79, 82, 80],
       [71, 73, 76, 75, 75]], dtype=uint8)
```

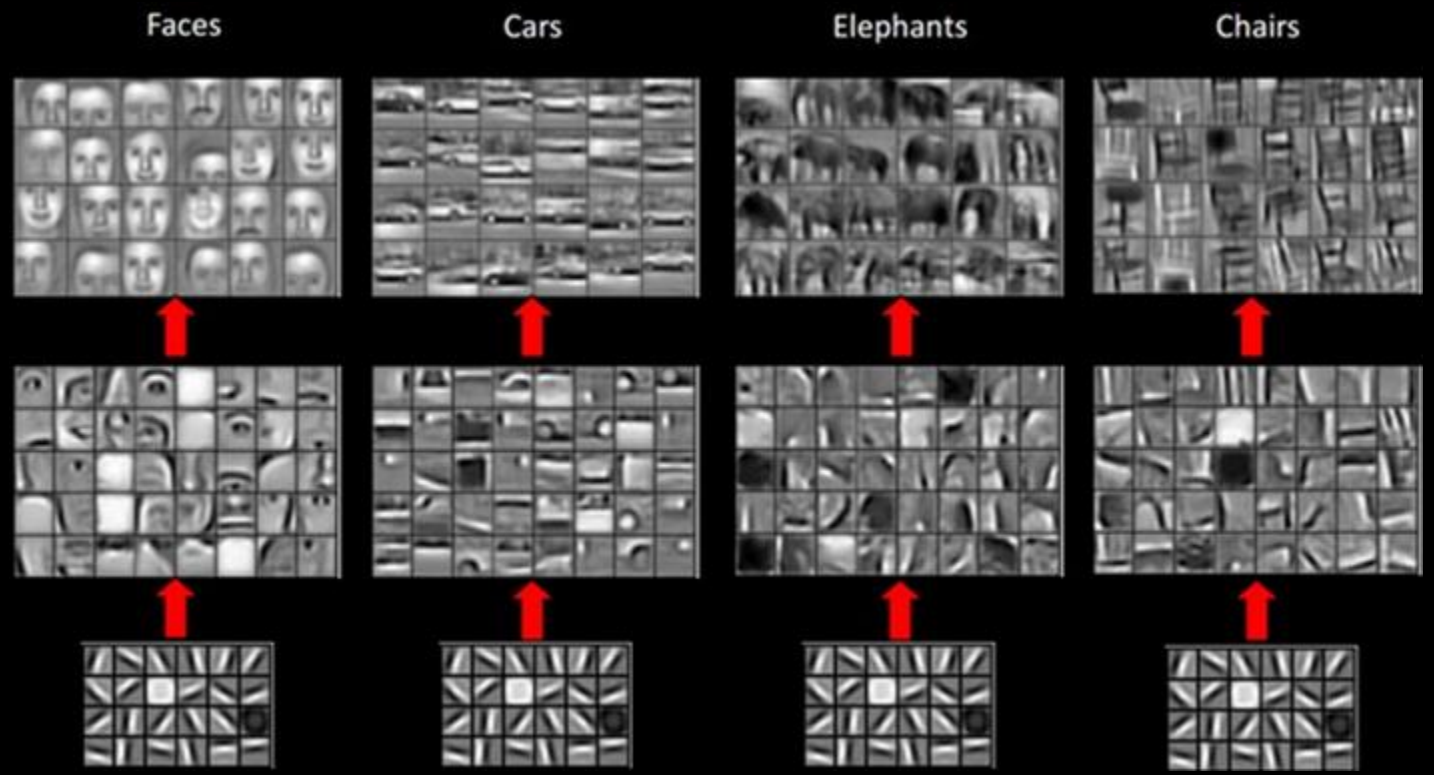
First 5x5 values

```
array([[58, 57, 57, 59, 59],
       [58, 57, 57, 58, 59],
       [59, 58, 58, 58, 58],
       [61, 61, 60, 60, 59],
       [64, 63, 62, 61, 60]], dtype=uint8)
```

Convolutional Neural Networks

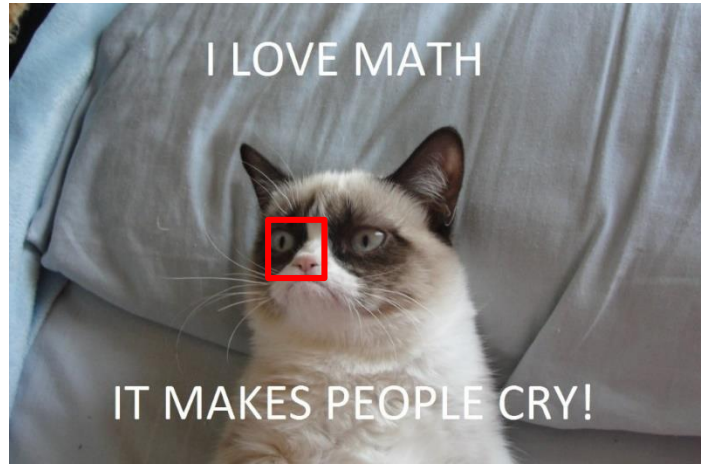
- Question: Spatial structure?
  - Answer: Convolutional filters
- Question: Huge input dimensionalities?
  - Answer: Parameters are shared between filters
- Question: Local variances?
  - Answer: Pooling

Preserving spatial structure

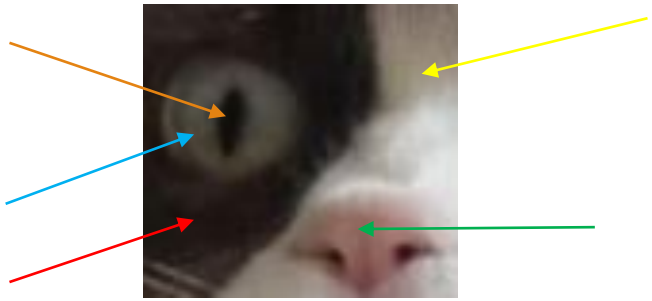


Why spatial?

- Images are 2-D
  - k-D if you also count the extra channels
  - RGB, hyperspectral, etc.

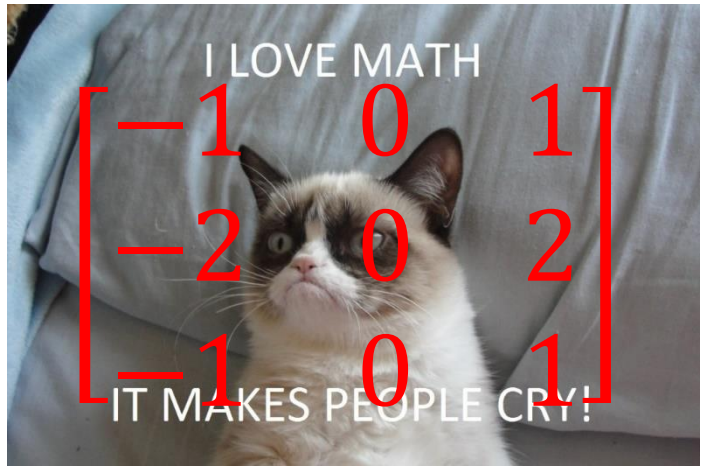


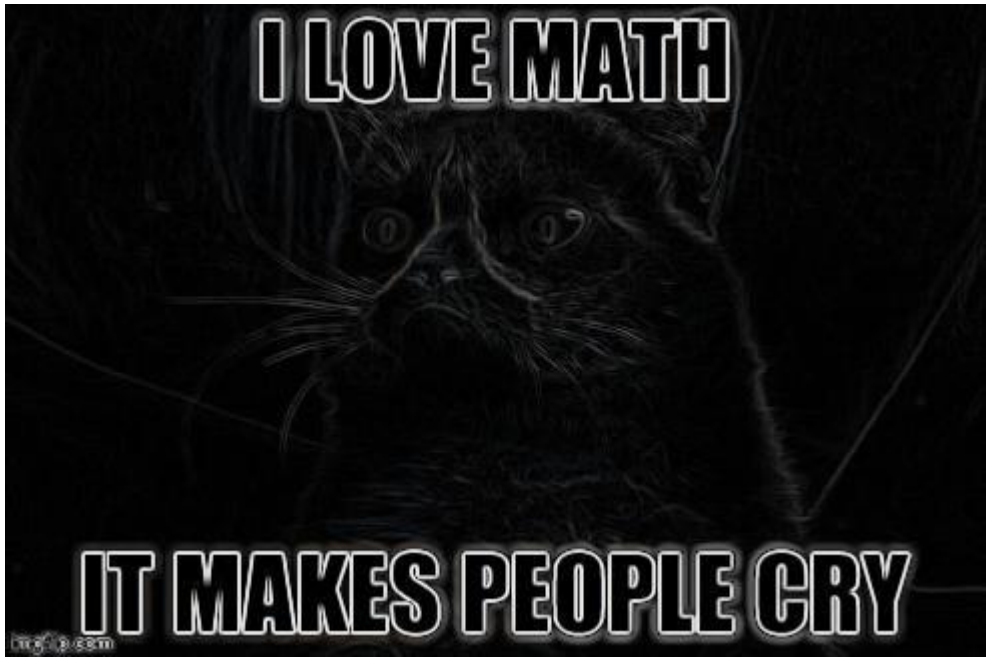
- What does a 2-D input really mean?
  - Neighboring variables are locally correlated



Example filter when K=1

e.g. Sobel 2-D filter





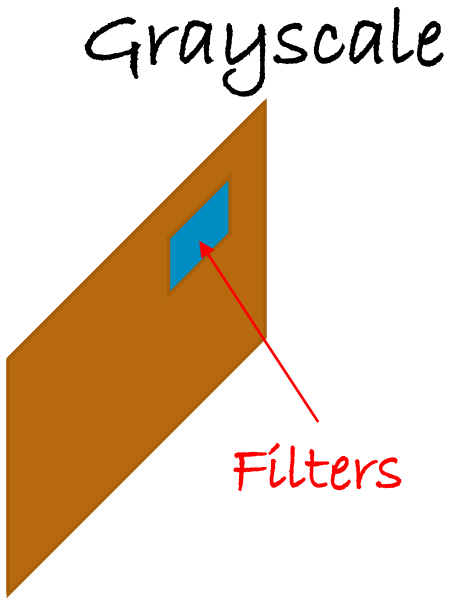
Learnable filters

- Several, handcrafted filters in computer vision
  - Canny, Sobel, Gaussian blur, smoothing, low-level segmentation, morphological filters, Gabor filters
- Are they optimal for recognition?
- Can we learn them from our data?
- Are they going resemble the handcrafted filters?

$$\begin{bmatrix} \theta_1 & \theta_2 & \theta_3 \\ \theta_4 & \theta_5 & \theta_6 \\ \theta_7 & \theta_8 & \theta_9 \end{bmatrix}$$

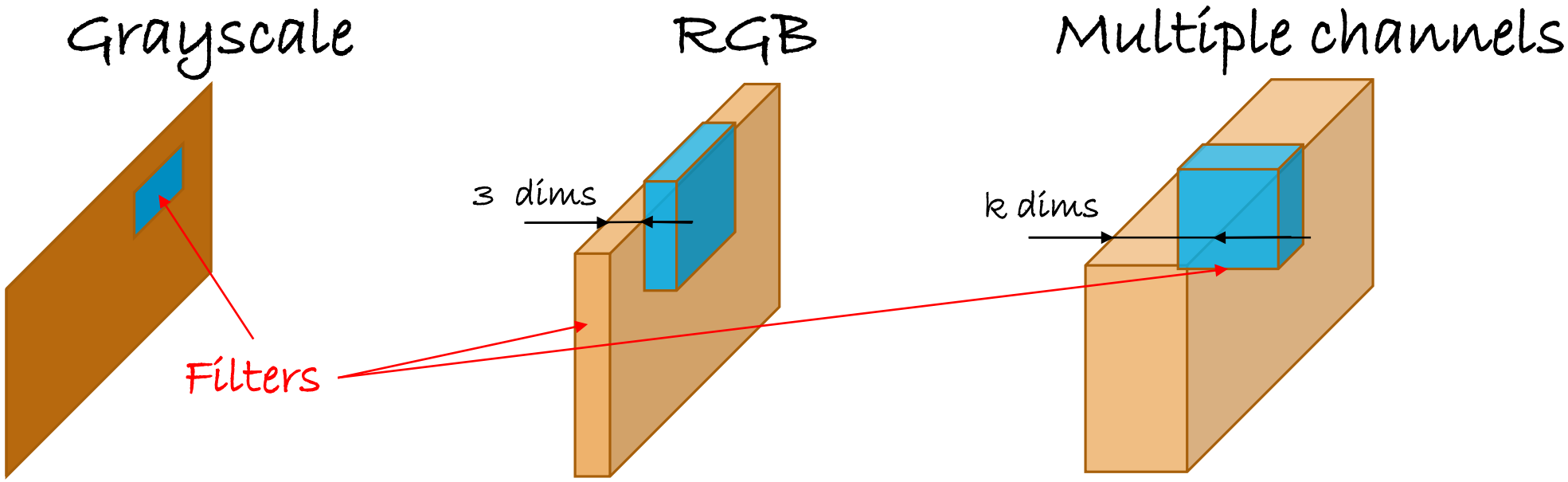
2-D Filters (Parameters)

- If images are 2-D, parameters should also be organized in 2-D
  - That way they can learn the local correlations between input variables
  - That way they can "exploit" the spatial nature of images



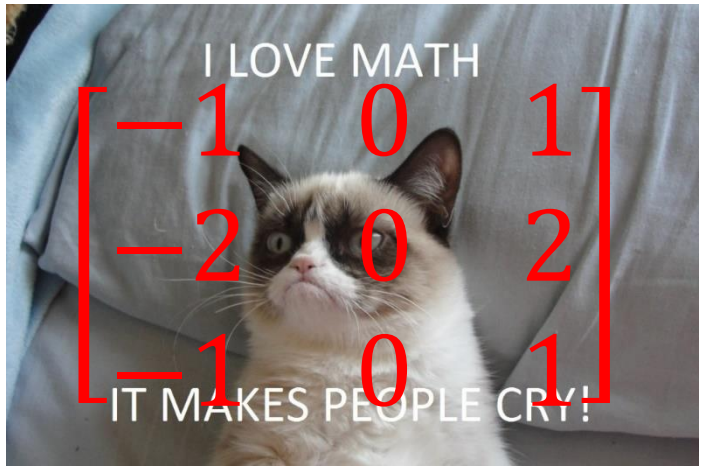
K-D Filters (Parameters)

- Similarly, if images are k-D, parameters should also be k-D

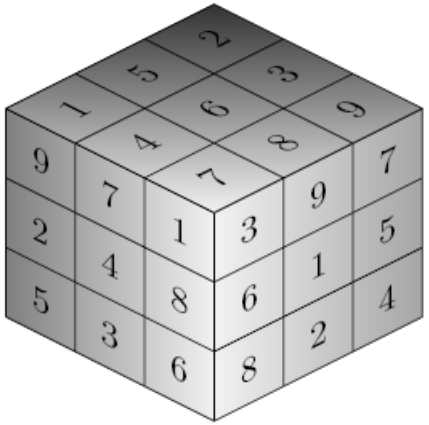


What would a k-D filter look like?

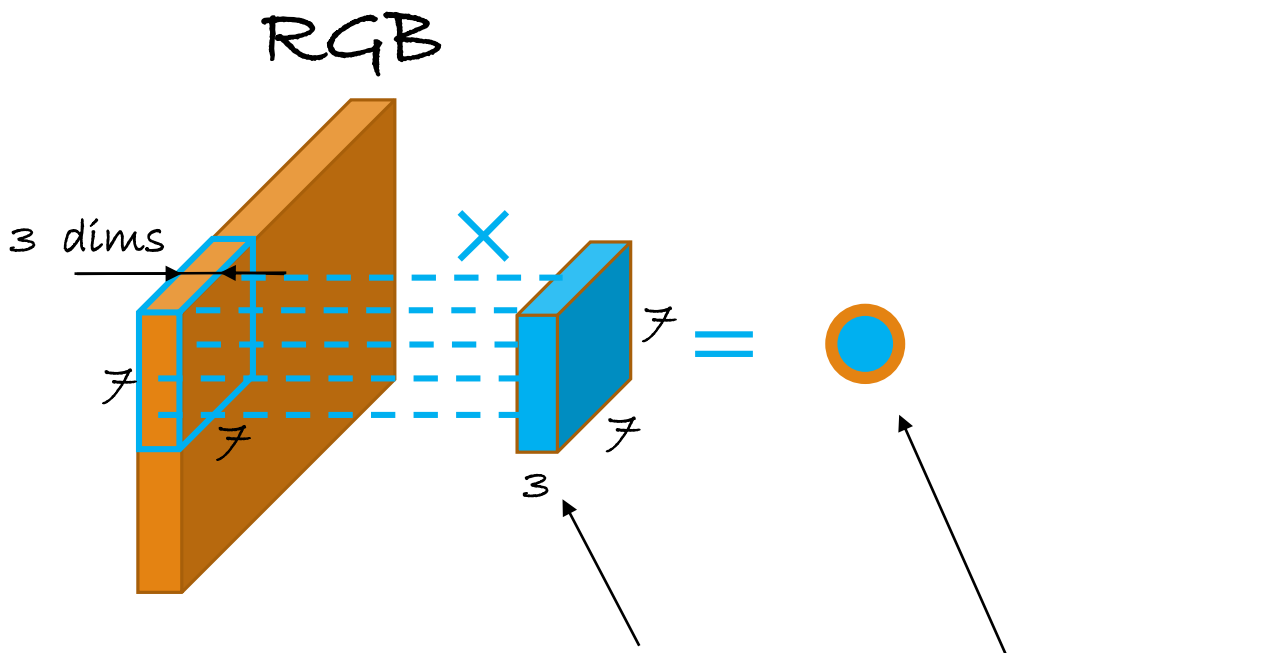
e.g. Sobel 2-D filter



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



Think in space



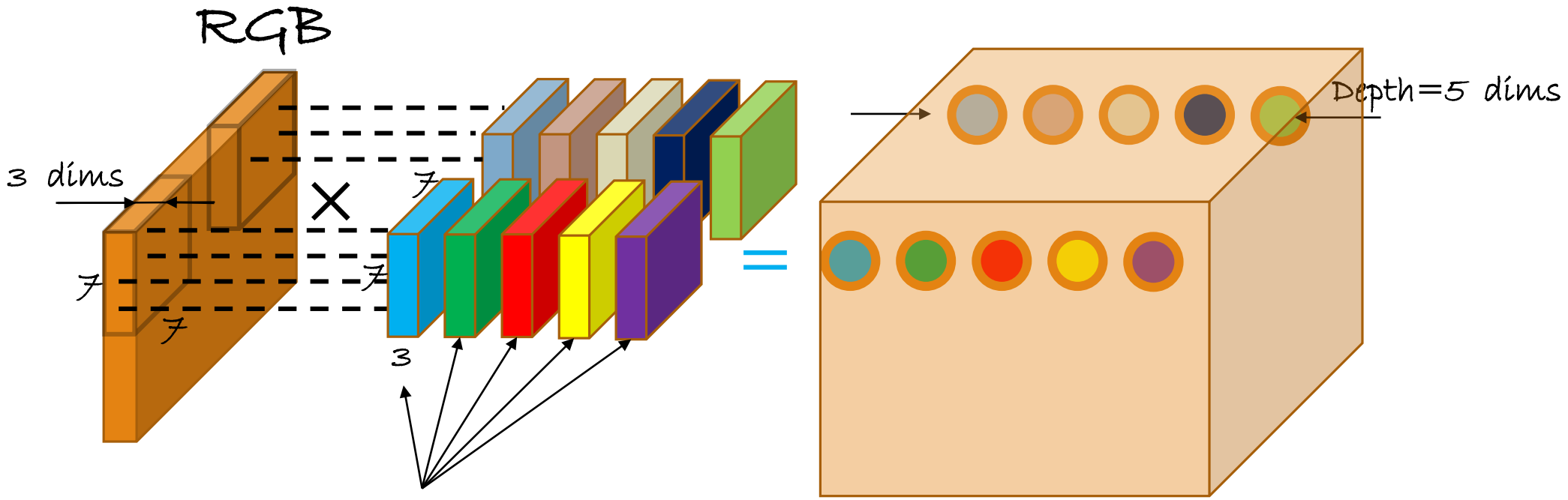
How many weights for this neuron?

$$7 \cdot 7 \cdot 3 = 147$$

Think in space

The activations of a hidden layer form a volume of neurons, not a 1-d "chain"

This volume has a depth 5, as we have 5 filters

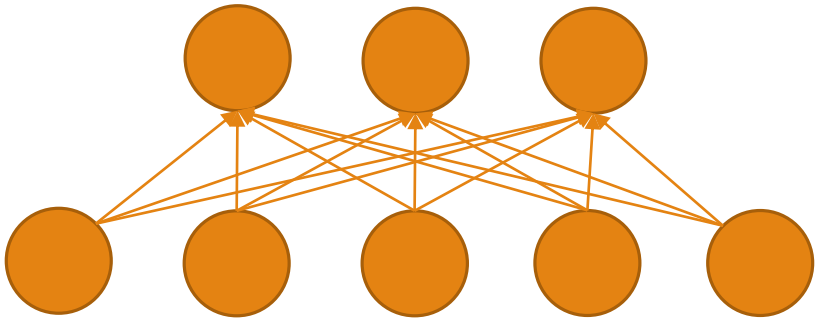


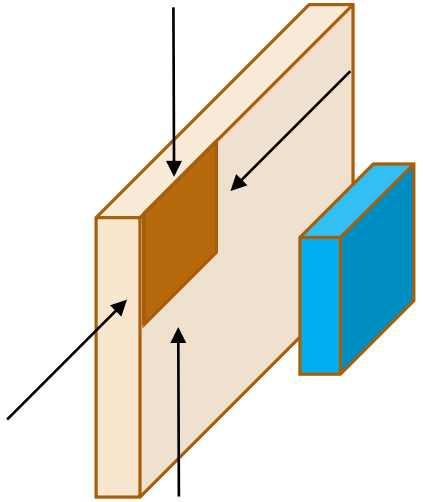
How many weights for these 5 neurons?

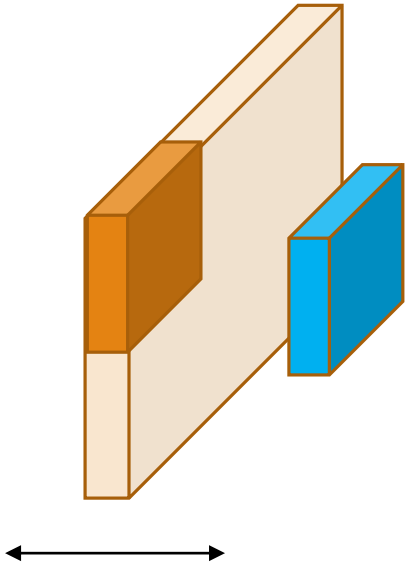
$$5 \cdot 7 \cdot 7 \cdot 3 = 735$$

Local connectivity

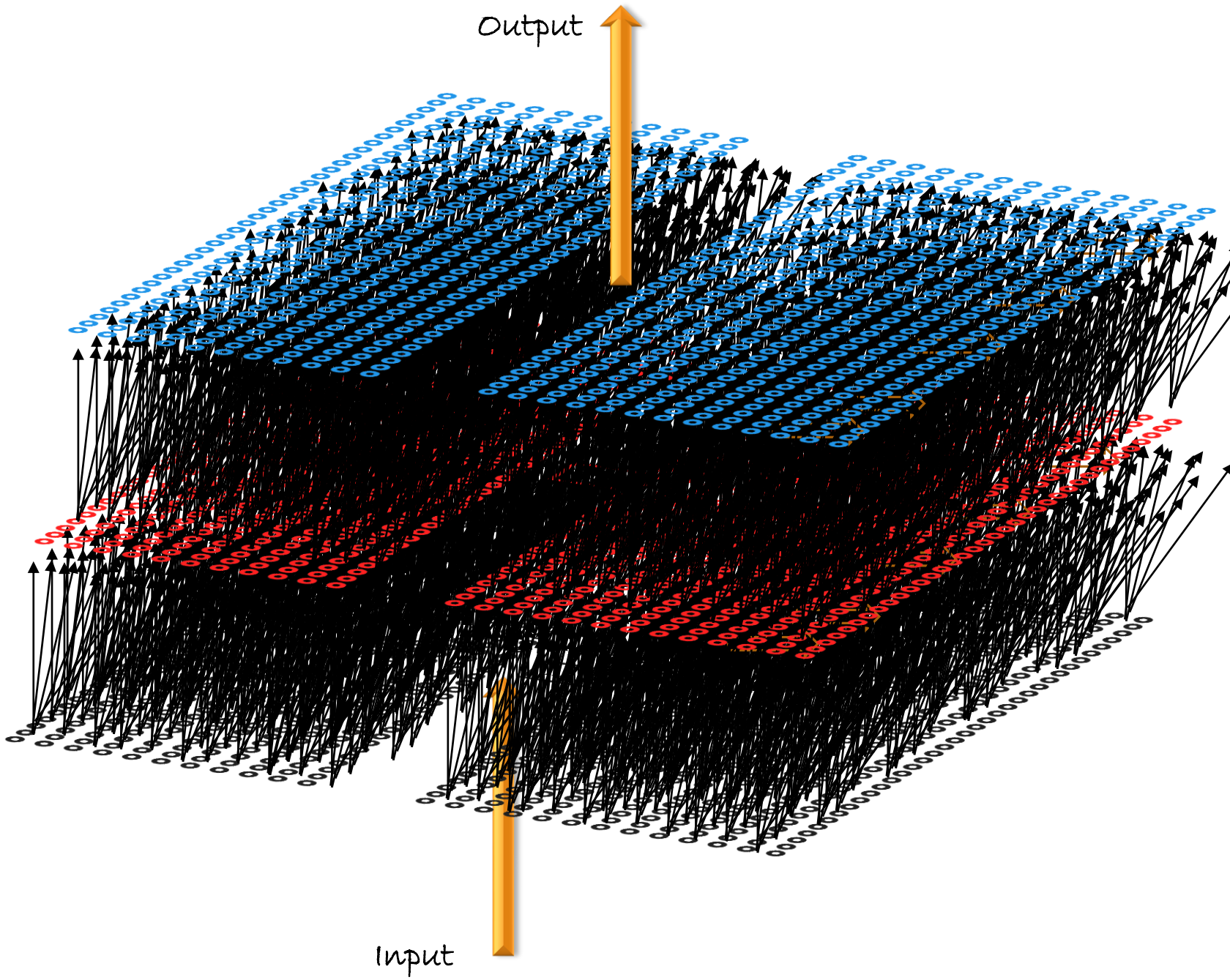
- The weight connections are surface-wise local!
  - Local connectivity
- The weights connections are depth-wise global
- For standard neurons no local connectivity
  - Everything is connected to everything



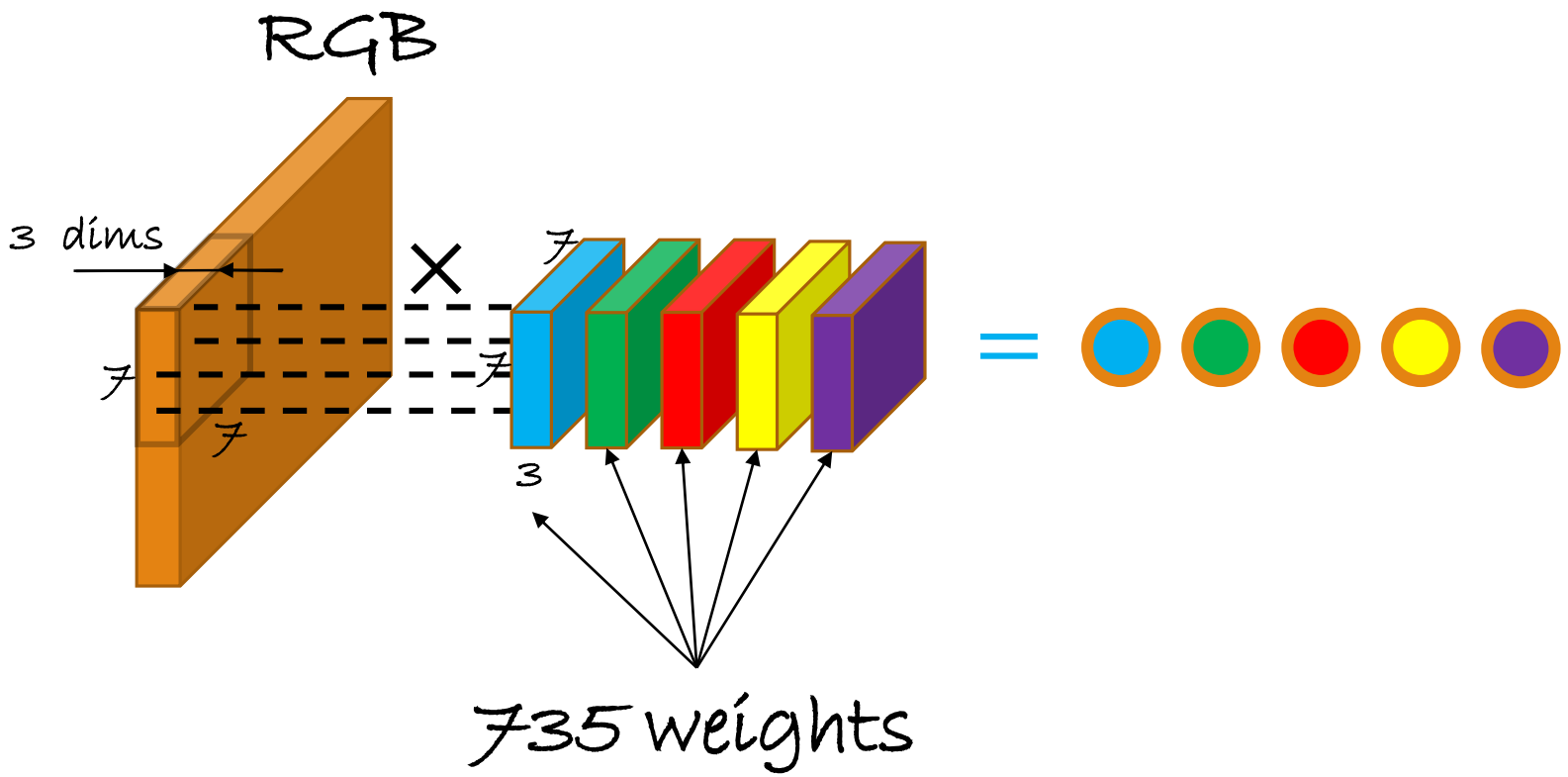




Filters *vs* Convolutional k-d filters

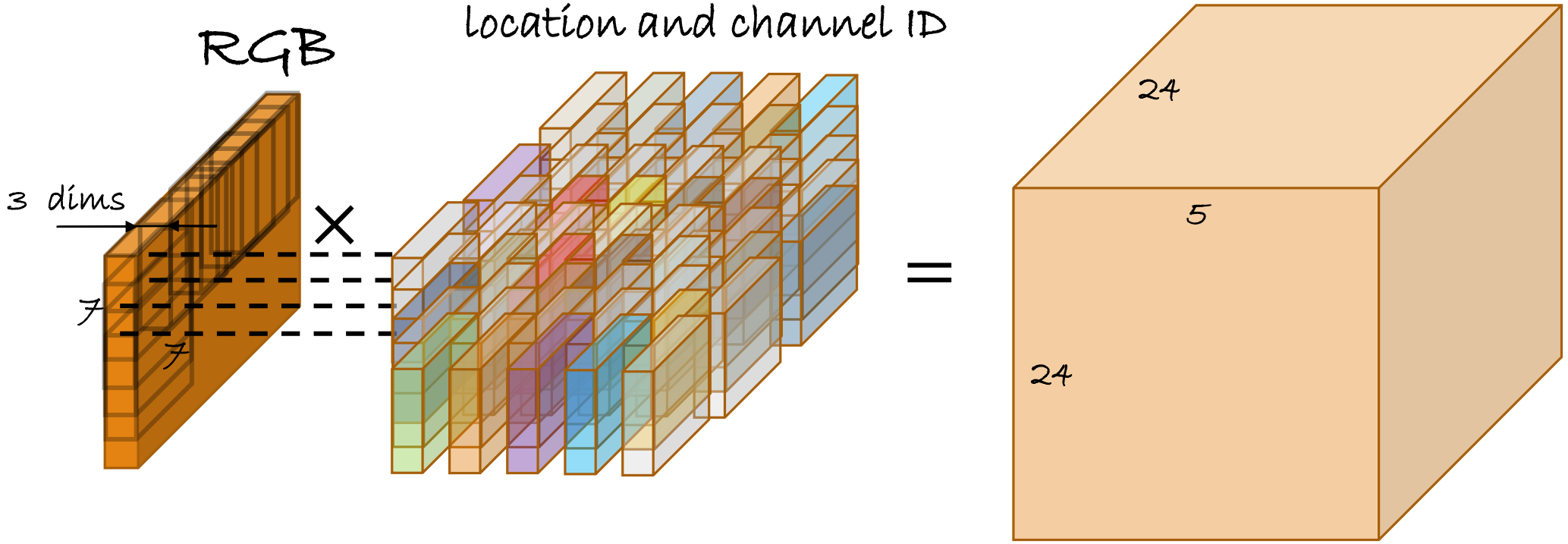


Again, think in space



Cover the whole image with ***non-shareable*** filters?

Filters with different values depending on image location and channel ID



Assume the image is 30x30x3.

1 filter every pixel (stride =1)

How many parameters in total?

24 filters along the x axis

24 filters along the y axis

Depth of 5

$\times$ $7 * 7 * 3$ parameters per filter

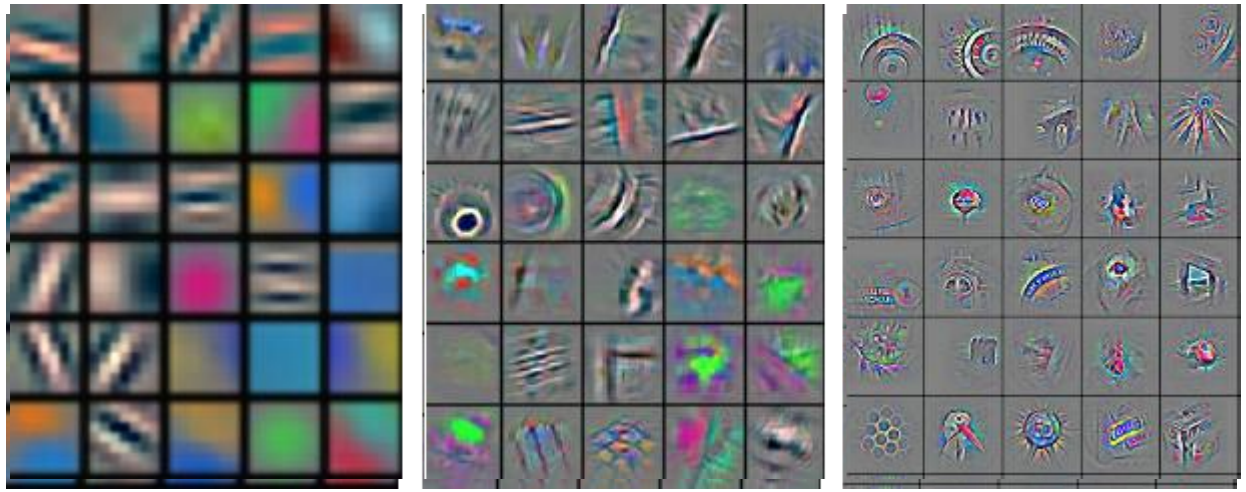
$423K$ parameters in total

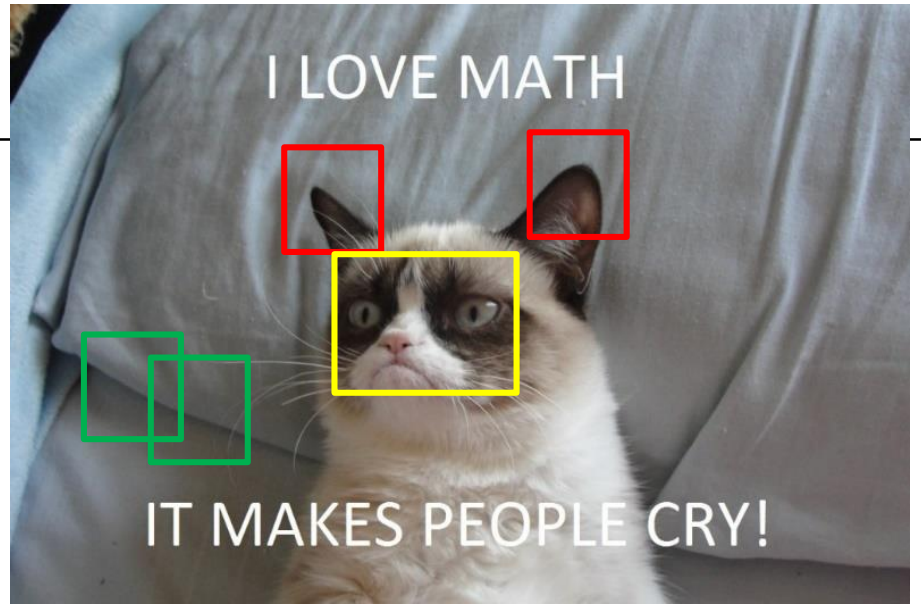
Problem!

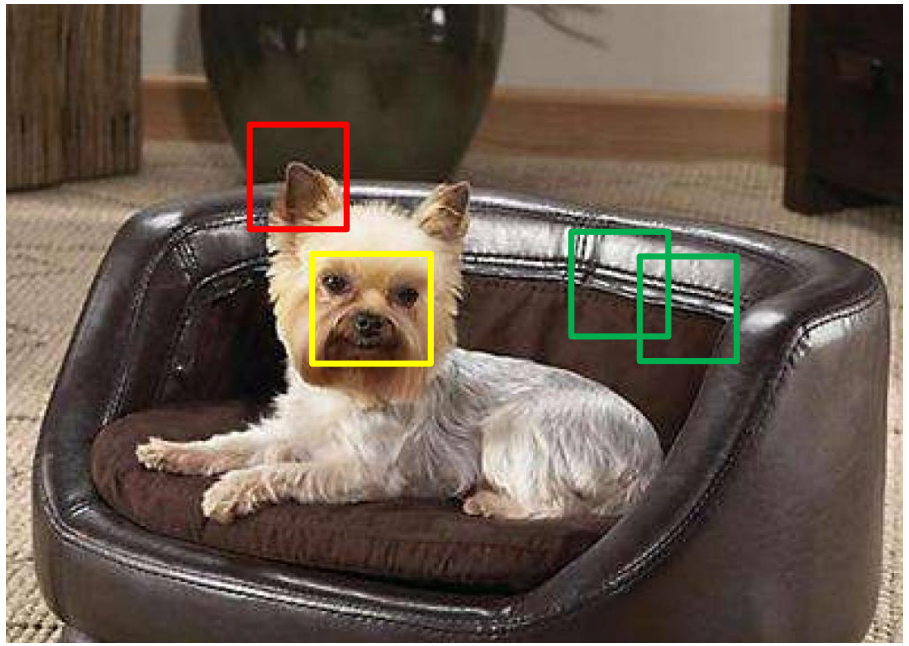
- Clearly, too many parameters
- With a only 30×30 pixels image and a single hidden layer of depth 5 we would need 85K parameters
  - With a 256×256 image we would need $46 \cdot 10^6$ parameters
- *Problem 1: Fitting a model with that many parameters is not easy*
- *Problem 2: Finding the data for such a model is not easy*
- *Problem 3: Are all these weights necessary?*

Hypothesis

- Imagine
  - With the right amount of data …
  - … and if we connect all inputs of layer l with all outputs of layer $l+1$, …
  - … and if we would visualize the (2d) filters (local connectiveity → 2d) …
  - … we would see very similar filters no matter their location

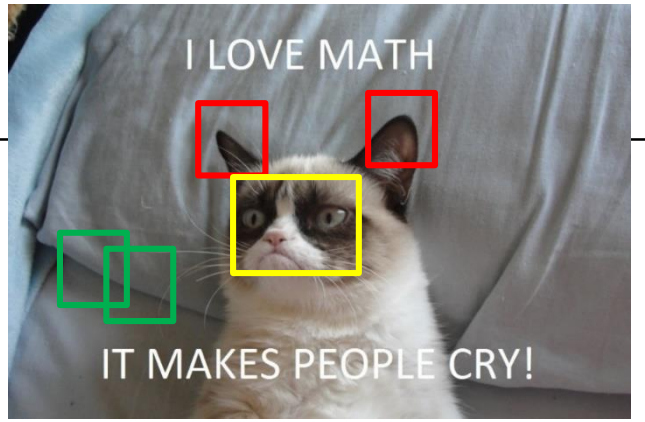


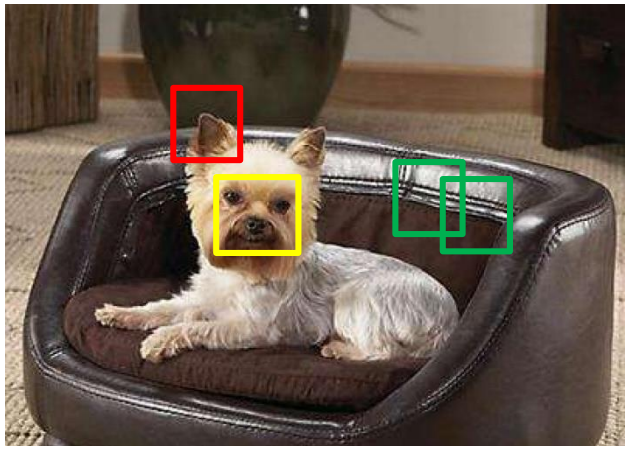


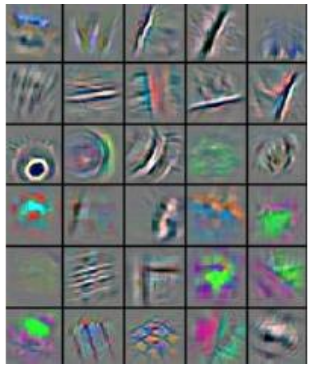


Hypothesis

- Imagine
  - With the right amount of data …
  - … and if we connect all inputs of layer l with all outputs of layer $l+1$, …
  - … and if we would visualize the (2d) filters (local connectiveity → 2d) …
  - … we would see very similar filters no matter their location
- Why?
  - Natural images are stationary
  - Visual features are common for different parts of one or multiple image

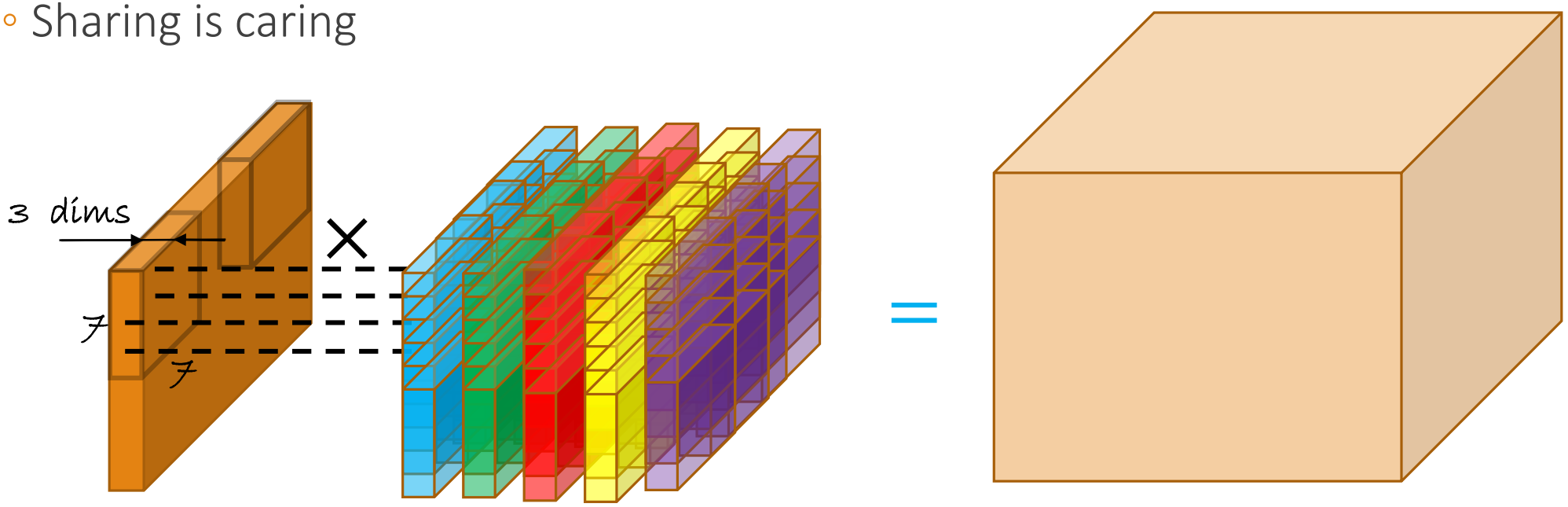






Solution? Share!

- So, if we are anyways going to compute the same filters, why not share?
  - Sharing is caring



Assume the image is 30x30x3.
1 column of filters common across the image.
How many parameters in total?

Depth of 5
$\times$ $7 * 7 * 3$ parameters per filter

735 parameters in total

Shared 2-D filters → Convolutions

Original image

Shared 2-D filters → Convolutions

Original image

Convolutional filter 1

1	0	1
0	1	0
1	0	1

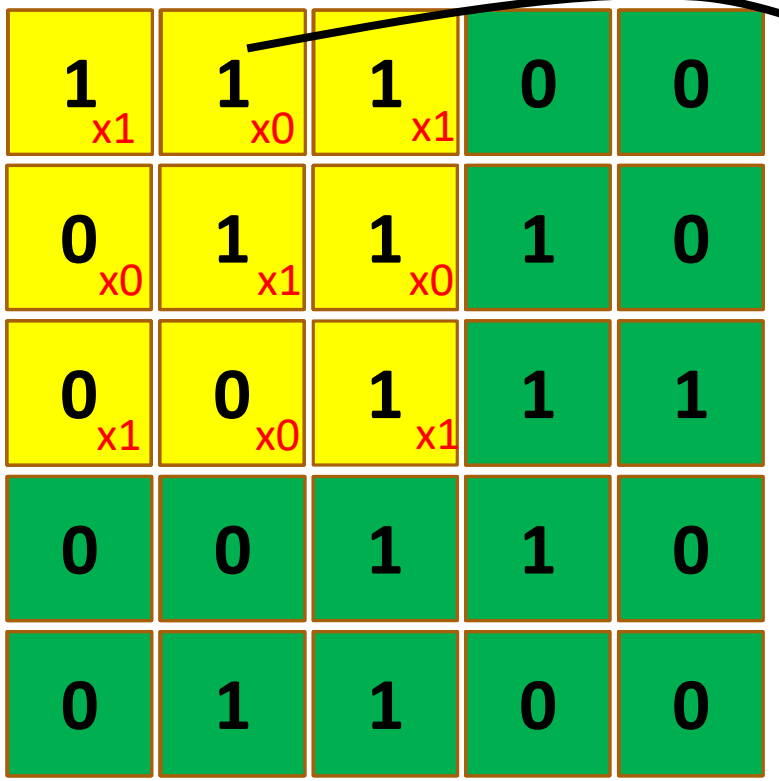
Shared 2-D filters → Convolutions

Original image

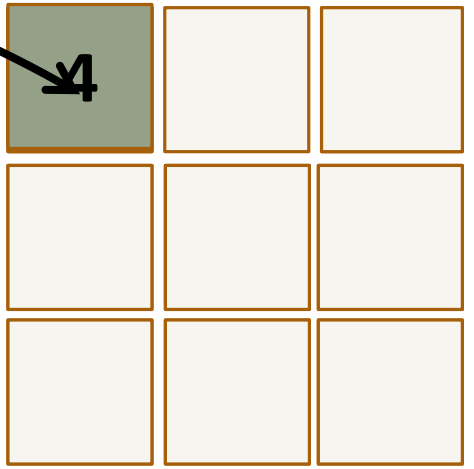
Convolutional filter 1

1	0	1
0	1	0
1	0	1

Convolving the image



Result



Inner product

$$I(x,y) * h = \sum_{i=-a}^{a} \sum_{j=-b}^{b} I(x-i, y-j) \cdot h(i,j)$$

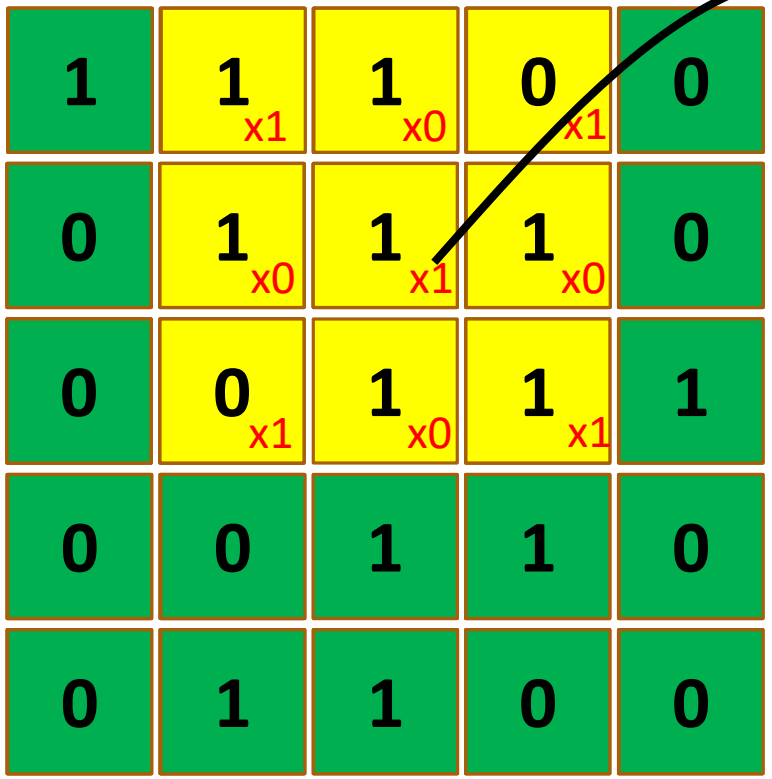
Shared 2-D filters → Convolutions

Original image

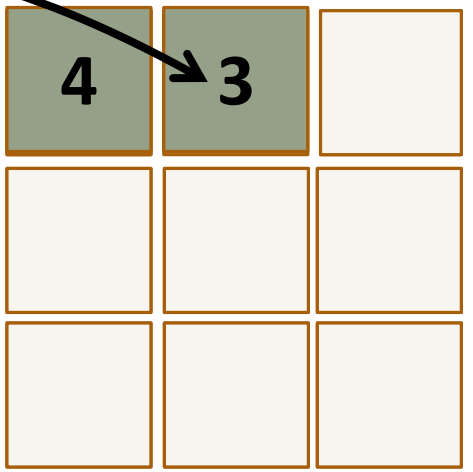
Convolutional filter 1

1	0	1
0	1	0
1	0	1

Convolving the image



Result



Inner product

$$I(x,y) * h = \sum_{i=-a}^{a} \sum_{j=-b}^{b} I(x-i, y-j) \cdot h(i,j)$$

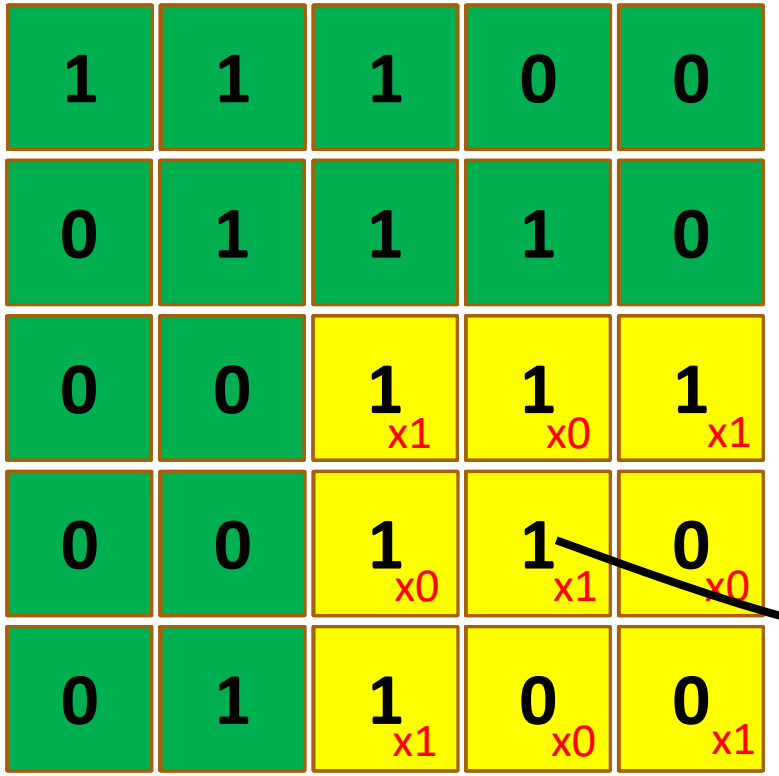
Shared 2-D filters → Convolutions

Original image

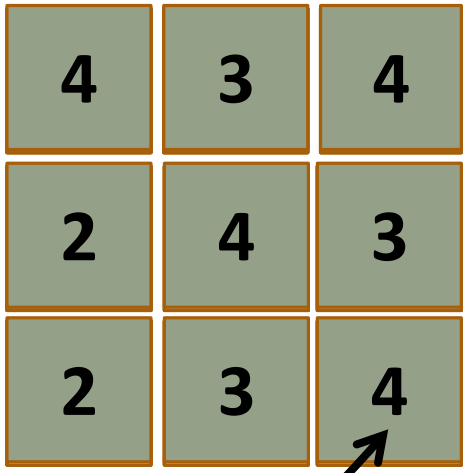
Convolutional filter 1

1	0	1
0	1	0
1	0	1

Convolving the image



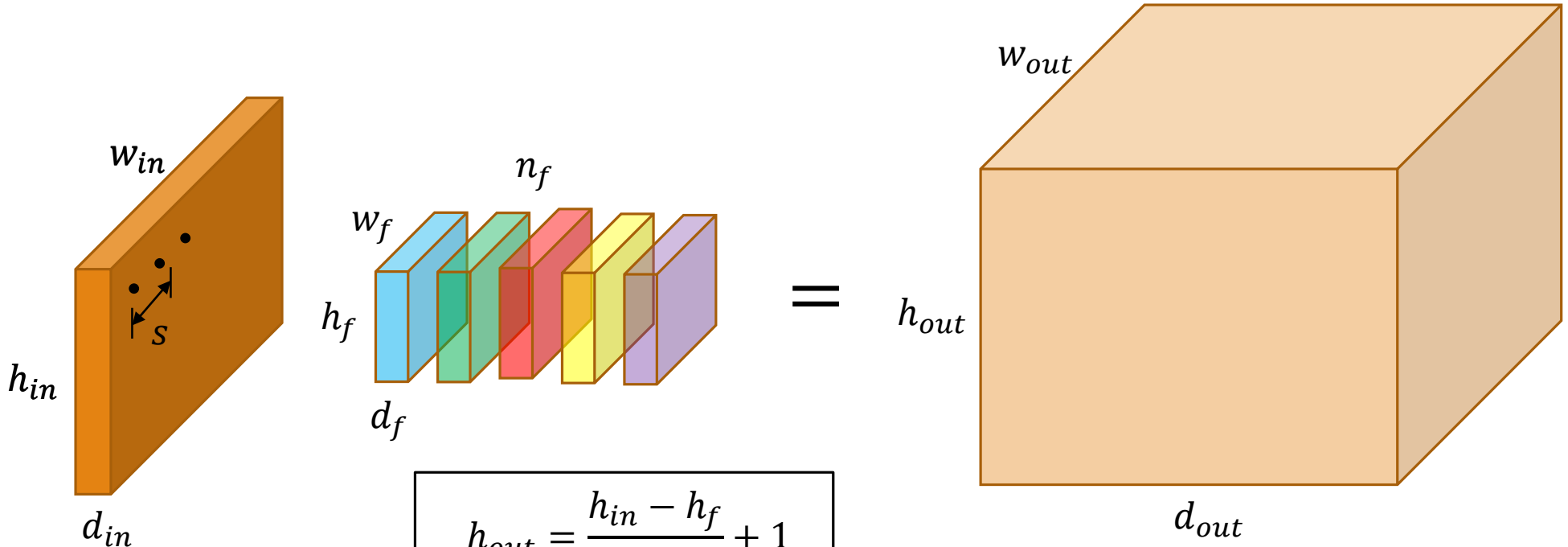
Result



Inner product

$$I(x,y) * h = \sum_{i=-a}^{a} \sum_{j=-b}^{b} I(x-i, y-j) \cdot h(i,j)$$

Output dimensions?



$$h_{out} = \frac{h_{in} - h_f}{s} + 1$$

$$w_{out} = \frac{w_{in} - w_f}{s} + 1$$

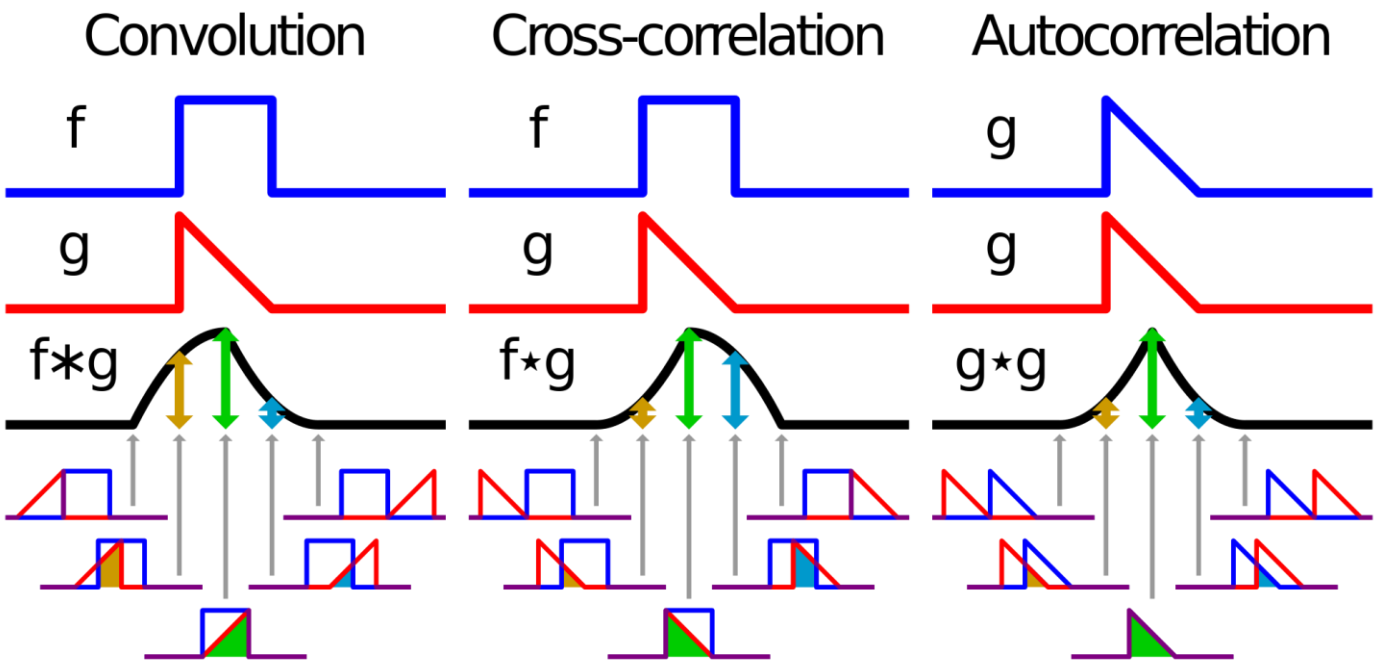
$$d_{out} = n_f$$

$$d_f = d_{in}$$

Why call them convolutions?

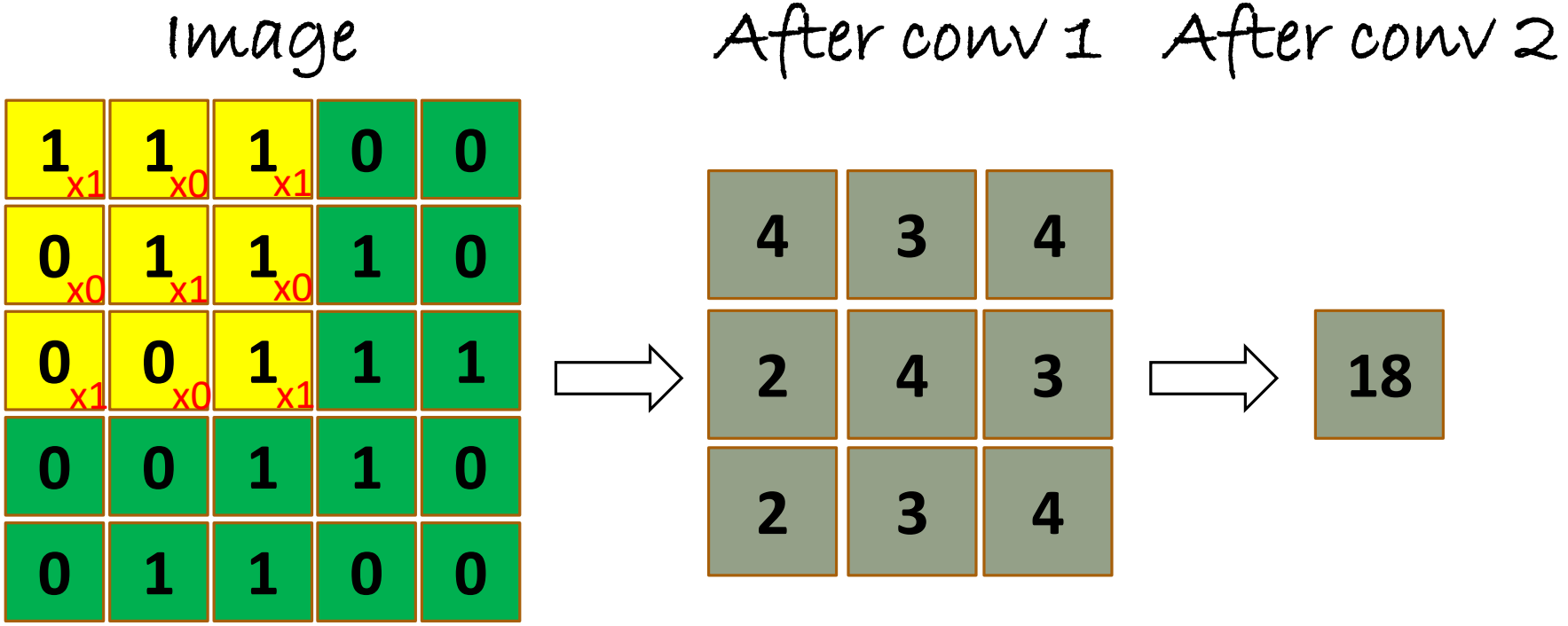
Definition *The convolution of two functions f and g is denoted by $*$ as the integral of the product of the two functions after one is reversed and shifted*

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,d\tau = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)\,d\tau$$



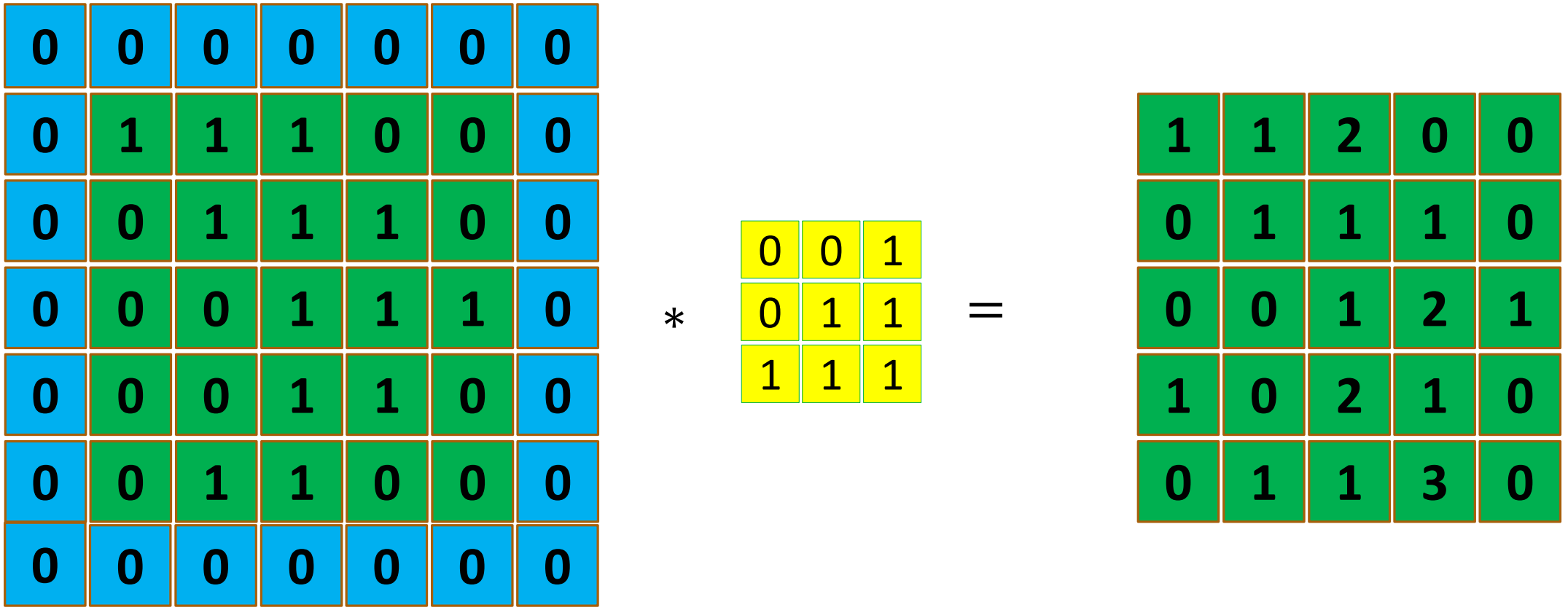
Problem, again :S

- Our images get smaller and smaller
- Not too deep architectures
- Details are lost



Solution? Zero-padding!

- For $\mathrm{s} = 1$, surround the image with $(h_f - 1)/2$ and $(w_f - 1)/2$ layers of 0



Convolutional module (New module!!!)

- Activation function

$$a_{rc} = \sum_{i=-a}^{a} \sum_{j=-b}^{b} x_{r-i,c-j} \cdot \theta_{ij}$$

- Essentially a dot product, similar to linear layer

$$a_{rc} \sim x_{region}^{T} \cdot \theta$$

- Gradient w.r.t. the parameters

$$\frac{\partial a_{rc}}{\partial \theta_{ij}} = \sum_{r=0}^{N-2a} \sum_{c=0}^{N-2b} x_{r-i,c-j}$$

Convolutional module in Tensorflow

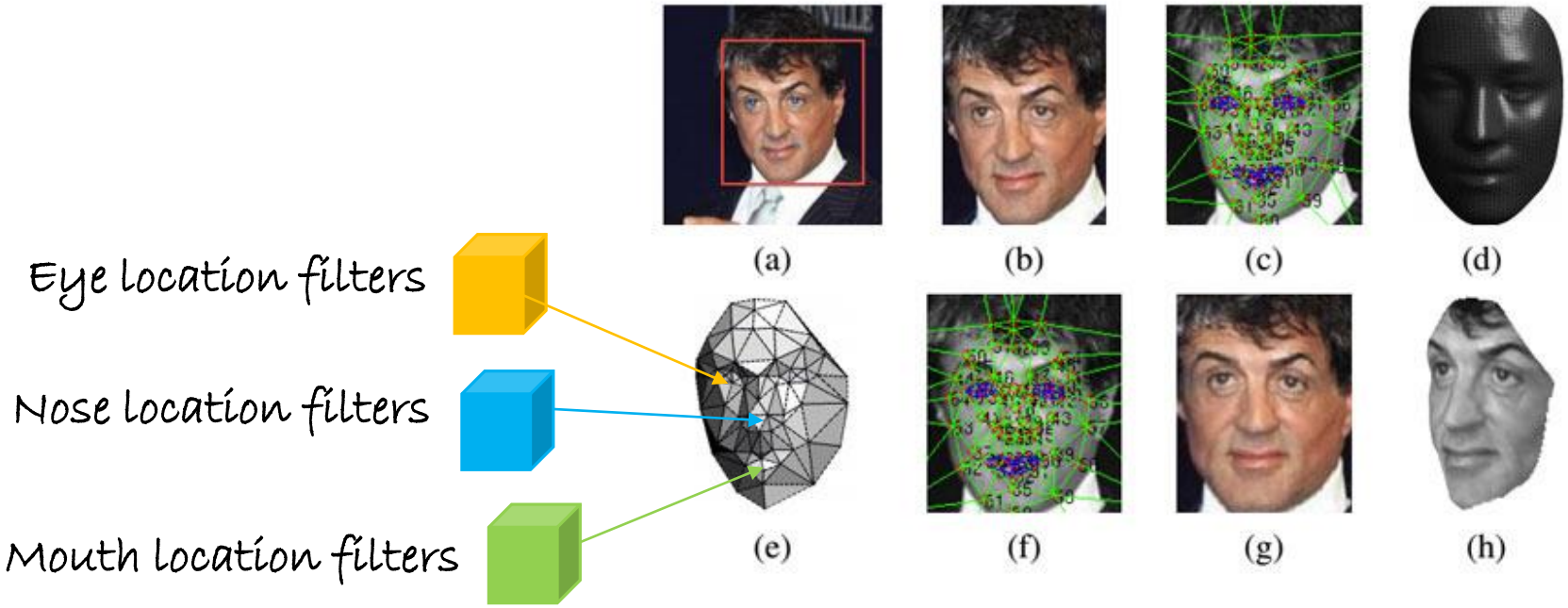
```
import Tensorflow as tf
tf.nn.conv2d(input, filter, strides, padding)
```

Good practice

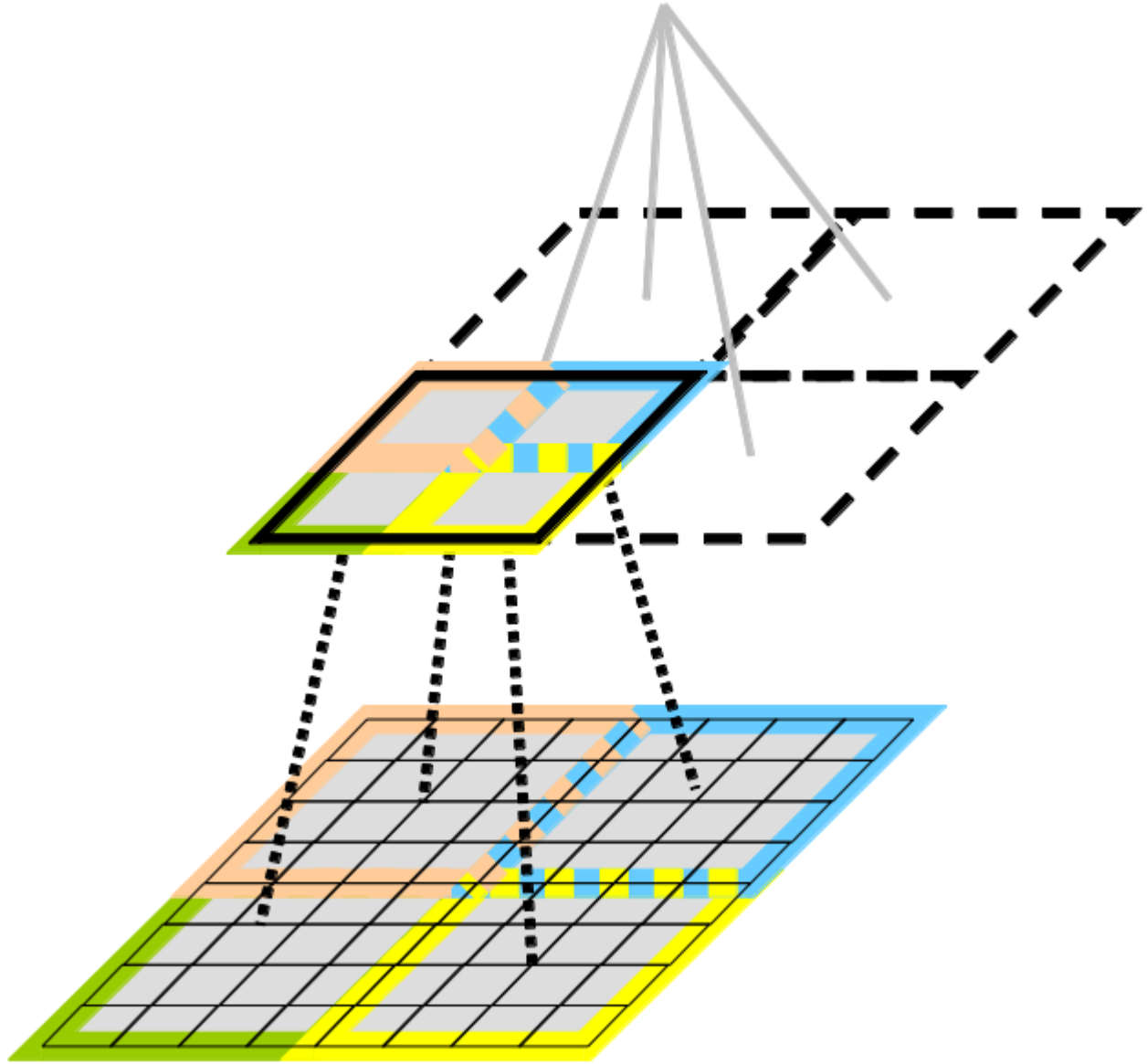
- Resize the image to have a size in the power of 2
- Use stride $s = 1$
- A filter of $(h_f, w_f) = [3 \times 3]$ works quite alright with deep architectures
- Add 1 layer of zero padding
- In general avoid combinations of hyper-parameters that do not click
  - E.g. $s = 1$
  - $[h_f \times w_f] = [3 \times 3]$ and
  - image size $[h_{in} \times w_{in}] = [6 \times 6]$
  - $[h_{out} \times w_{out}] = [2.5 \times 2.5]$
  - Programmatically worse, and worse accuracy because borders are ignored

P.S. Sometimes convolutional filters are not preferred

- When images are registered and each pixel has a particular significance
  - E.g. after face alignment specific pixels hold specific types of inputs, like eyes, nose, etc.
- In these cases maybe better every spatial filter to have different parameters
  - Network learns particular weights for particular image locations [Taigman2014]

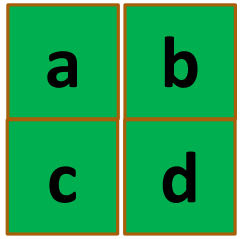


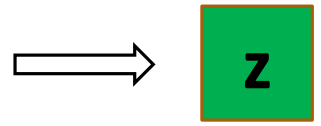
Pooling



Pooling

- Aggregate multiple values into a single value
  - Invariance to small transformations
  - Reduces the size of the layer output/input to next layer → Faster computations
  - Keeps most important information for the next layer
- Max pooling
- Average pooling



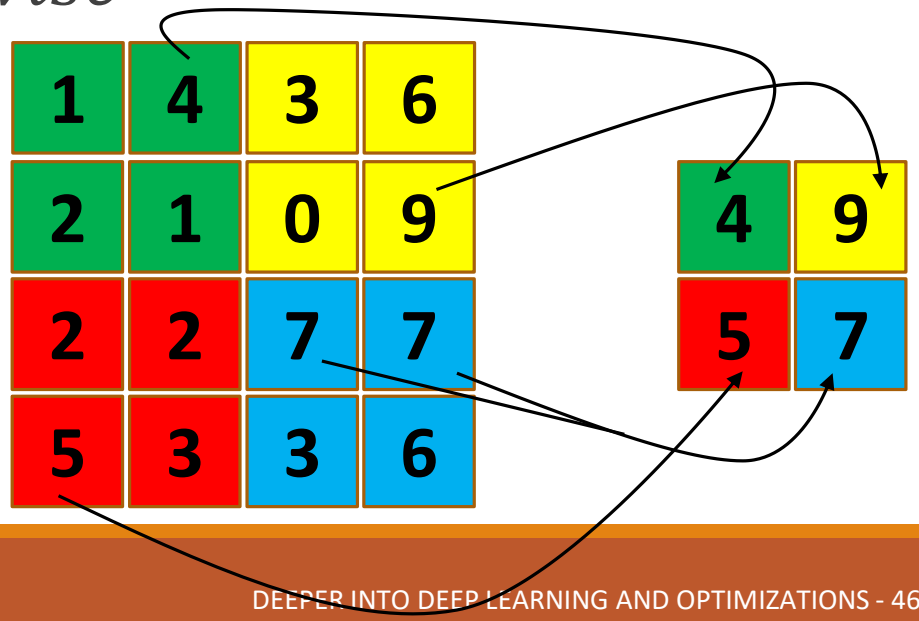


$$Pool\ ($$

$$) ==$$

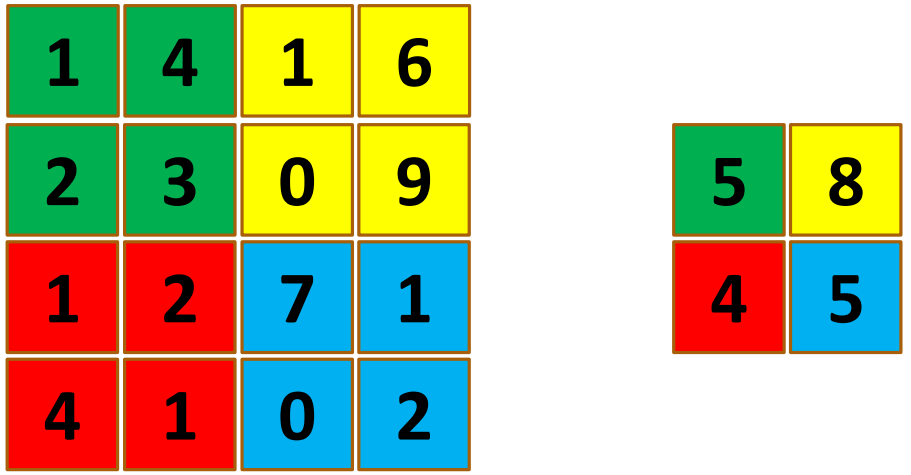
Max pooling (New module!)

- Run a sliding window of size $[h_f, w_f]$
- At each location keep the maximum value
- Activation function: $\mathrm{i}_{\max}, \mathrm{j}_{\max} = \arg\max_{i,j \in \Omega(r,c)} x_{ij} \rightarrow a_{rc} = x[\mathrm{i}_{\max}, \mathrm{j}_{\max}]$
- Gradient w.r.t. input $\frac{\partial a_{rc}}{\partial x_{ij}} = \begin{cases} 1, & if\ i = \mathrm{i}_{\max}, \mathrm{j} = \mathrm{j}_{\max} \\ 0, & otherwise \end{cases}$
- The preferred choice of pooling

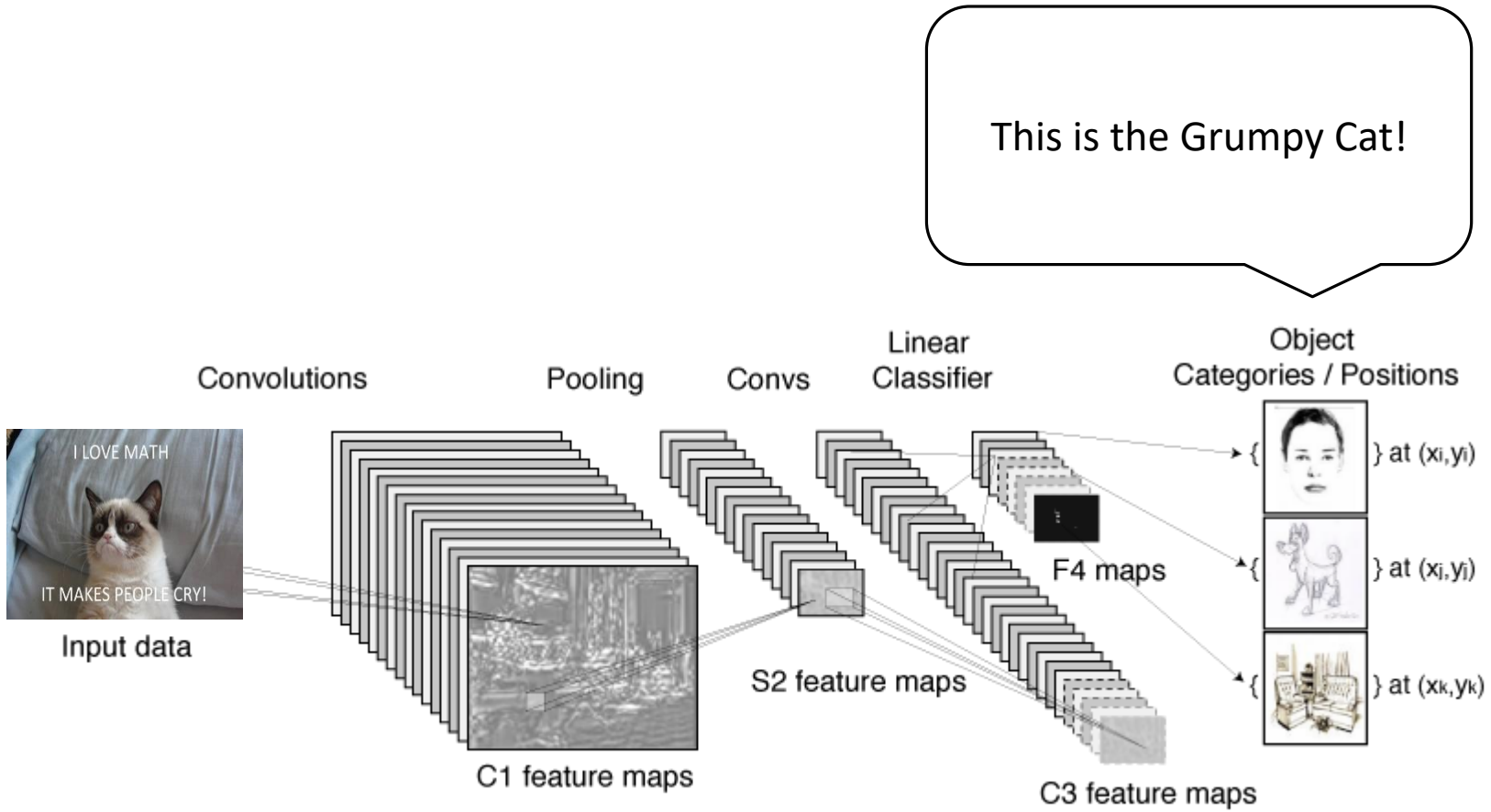


Average pooling (New module!)

- Run a sliding window of size $[h_f, w_f]$
- At each location keep the maximum value
- Activation function: $a_{rc} = \frac{1}{r \cdot c} \sum_{i,j \in \Omega(r,c)} x_{ij}$
- Gradient w.r.t. input $\frac{\partial a_{rc}}{\partial x_{ij}} = \frac{1}{r \cdot c}$

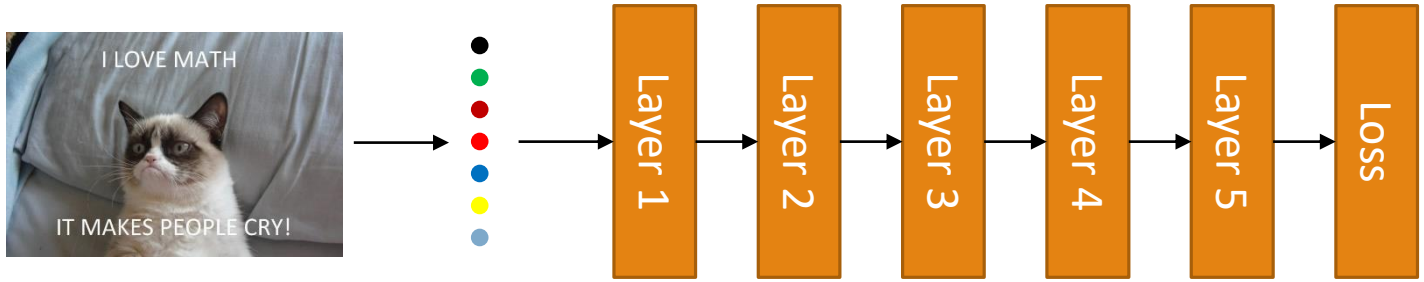


Convnets for Object Recognition

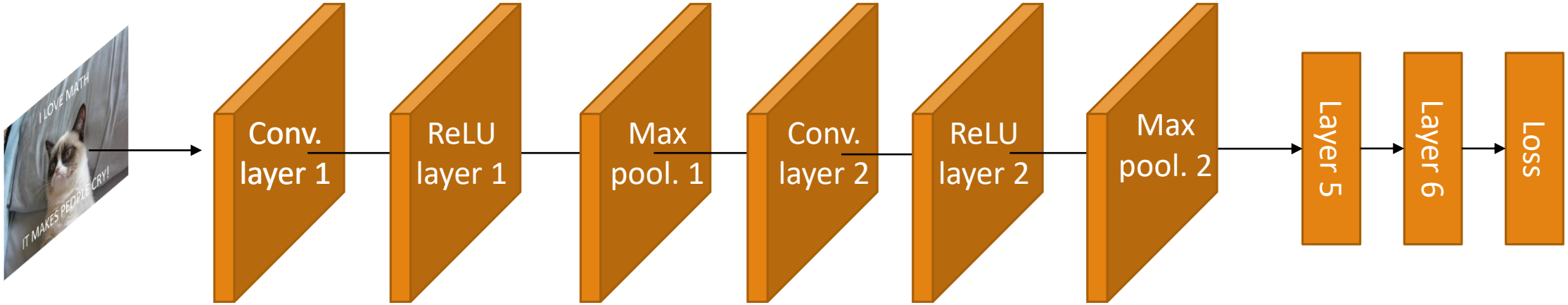


Standard Neural Network vs Convnets

Neural Network



Convolutional Neural Network



Convets in practice

- Several convolutional layers
  - 5 or more
- After the convolutional layers non-linearities are added
  - The most popular one is the ReLU
- After the ReLU usually some pooling
  - Most often max pooling
- After 5 rounds of cascading, vectorize last convolutional layer and connect it to a fully connected layer
- Then proceed as in a usual neural network

CNN Case Study I: Alexnet

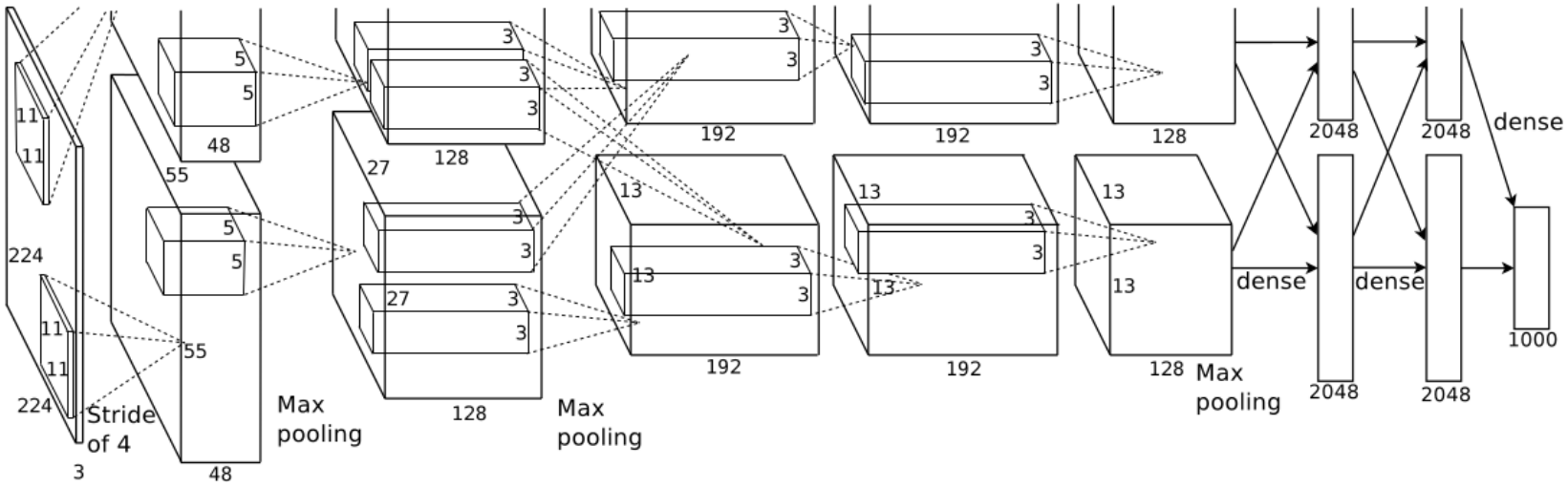
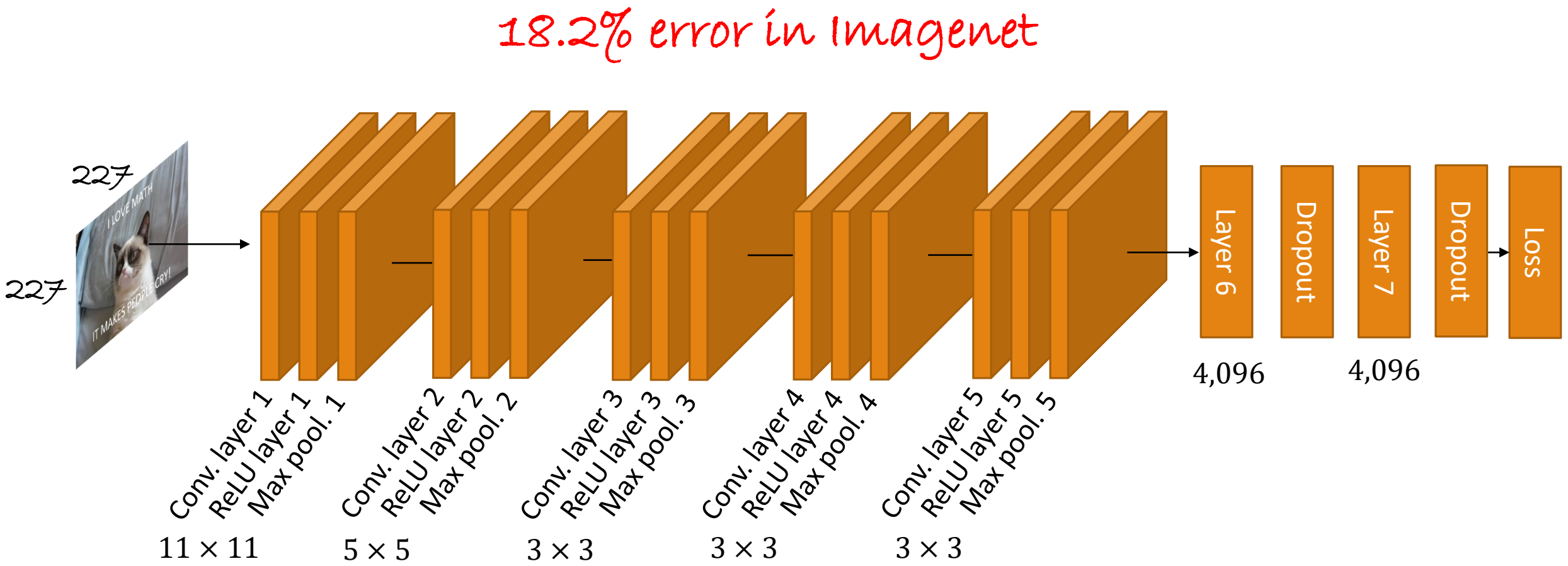


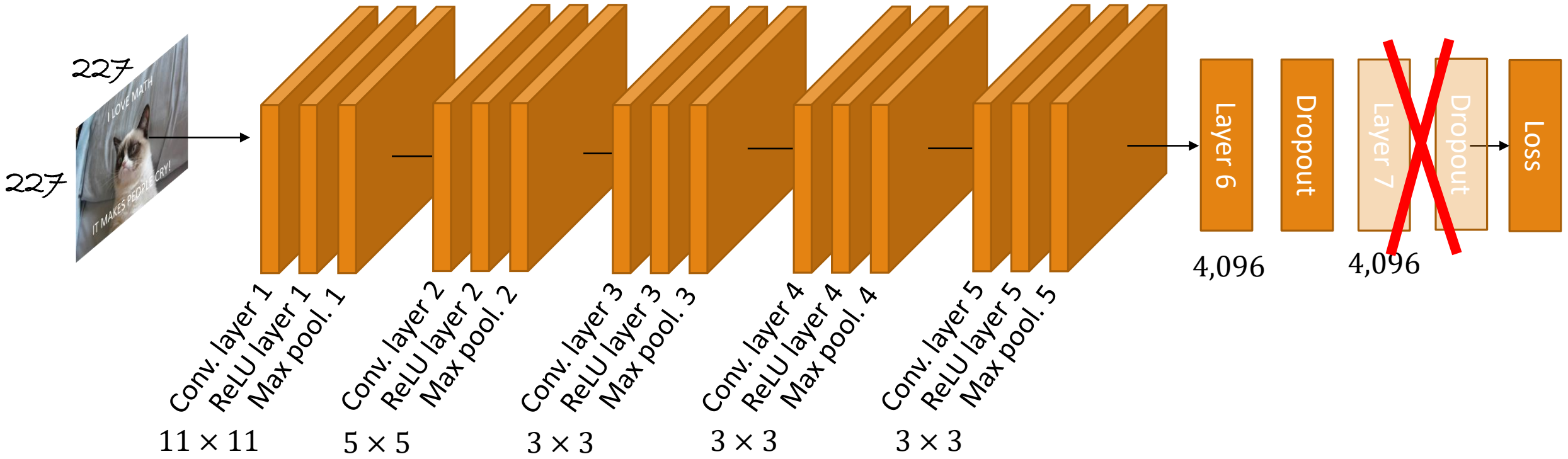
Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Architectural details



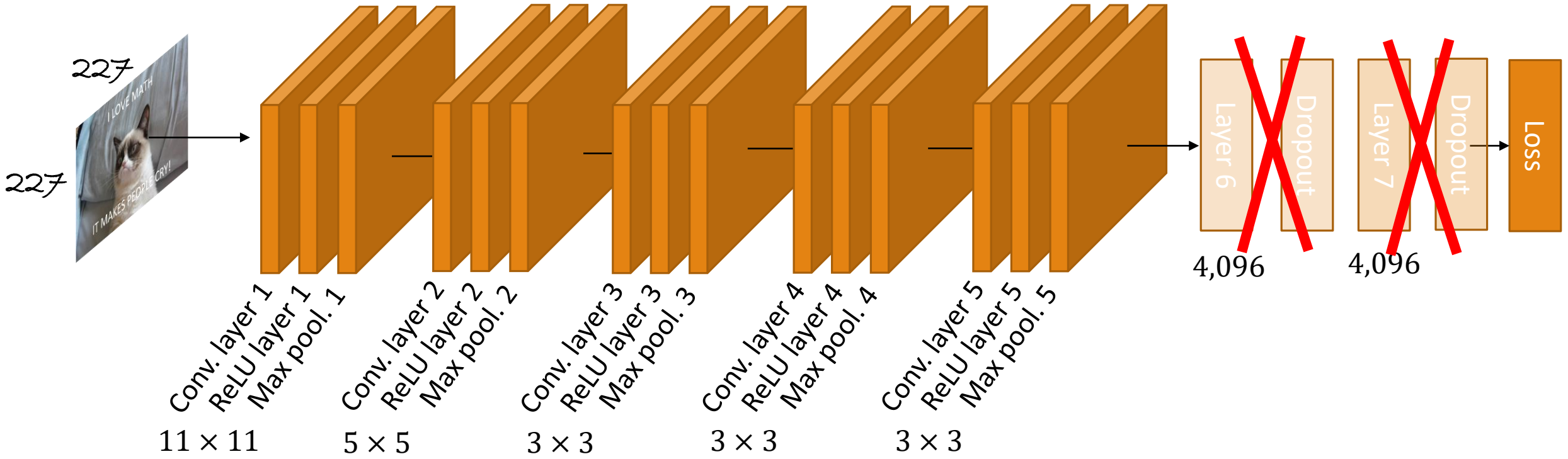
Removing layer 7

1.1% drop in performance, 16 million less parameters



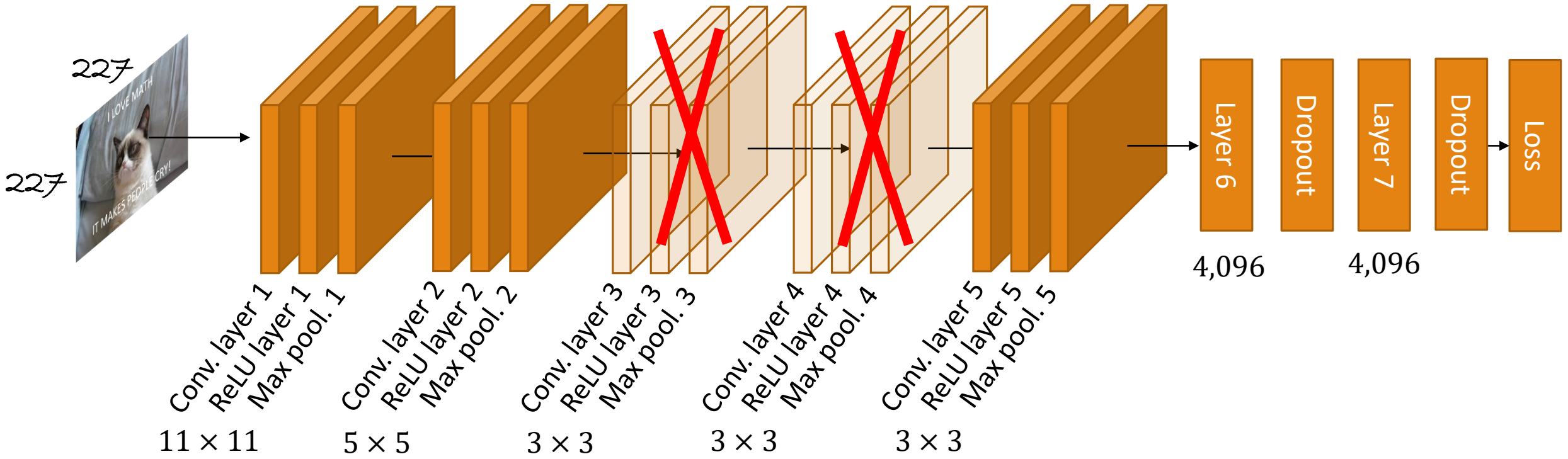
Removing layer 6, 7

5.7% drop in performance, 50 million less parameters



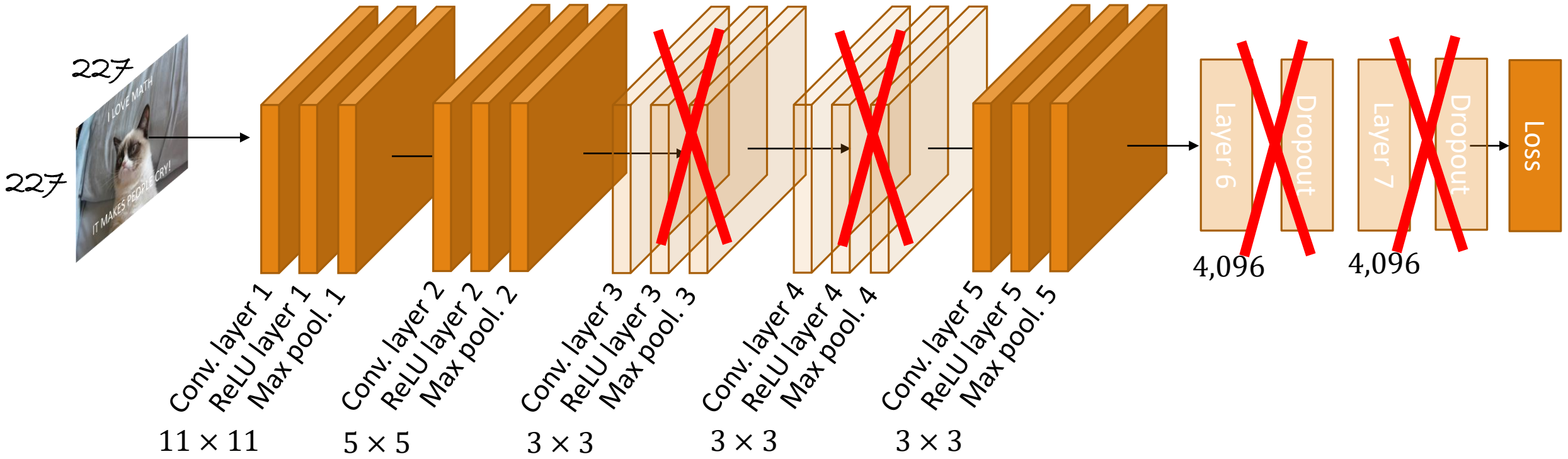
Removing layer 3, 4

3.0% drop in performance, 1 million less parameters. Why?

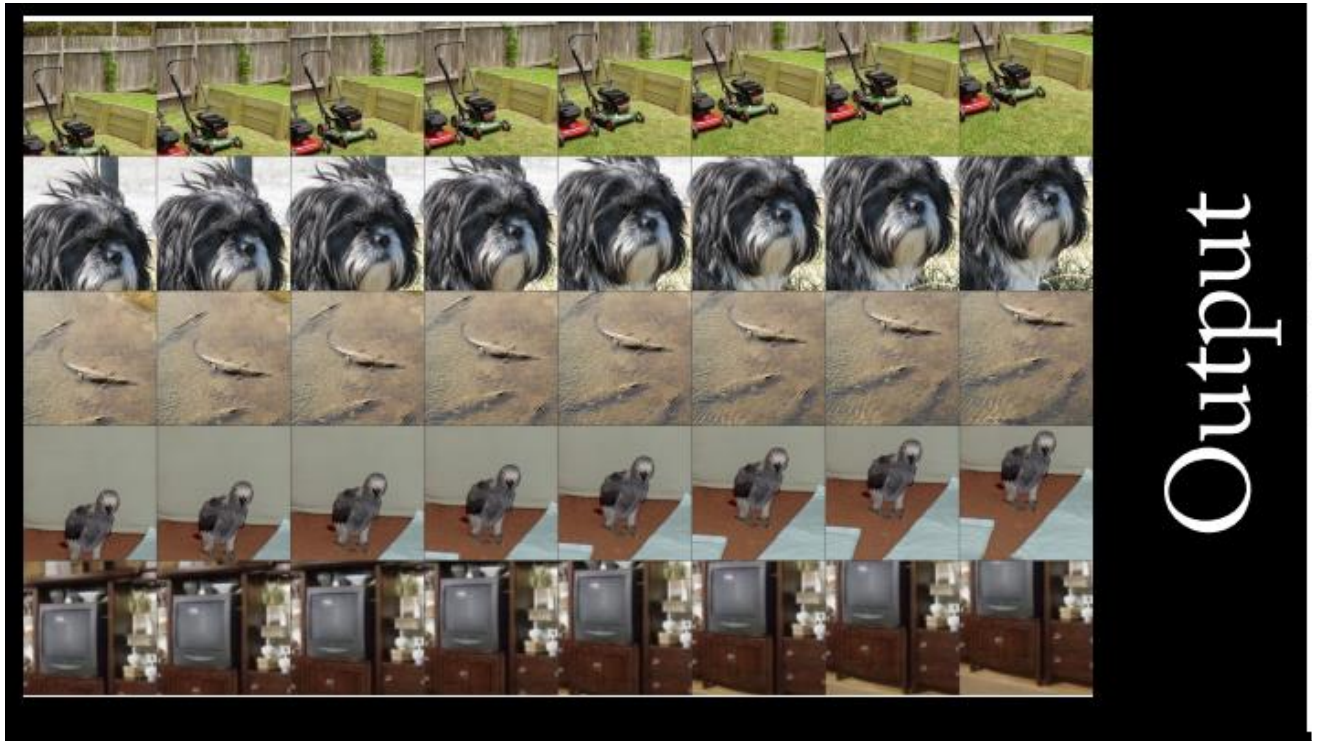


Removing layer 3, 4, 6, 7

33.5% drop in performance. Conclusion? Depth!



Translation invariance



Credit: R. Fergus slides in Deep Learning Summer School 2016

Prepare to vote

Internet

1

2

This presentation has been loaded without the Shakespeak add-in.
Want to download the add-in for free? Go to http://shakespeak.com/en/free-download/.

TXT

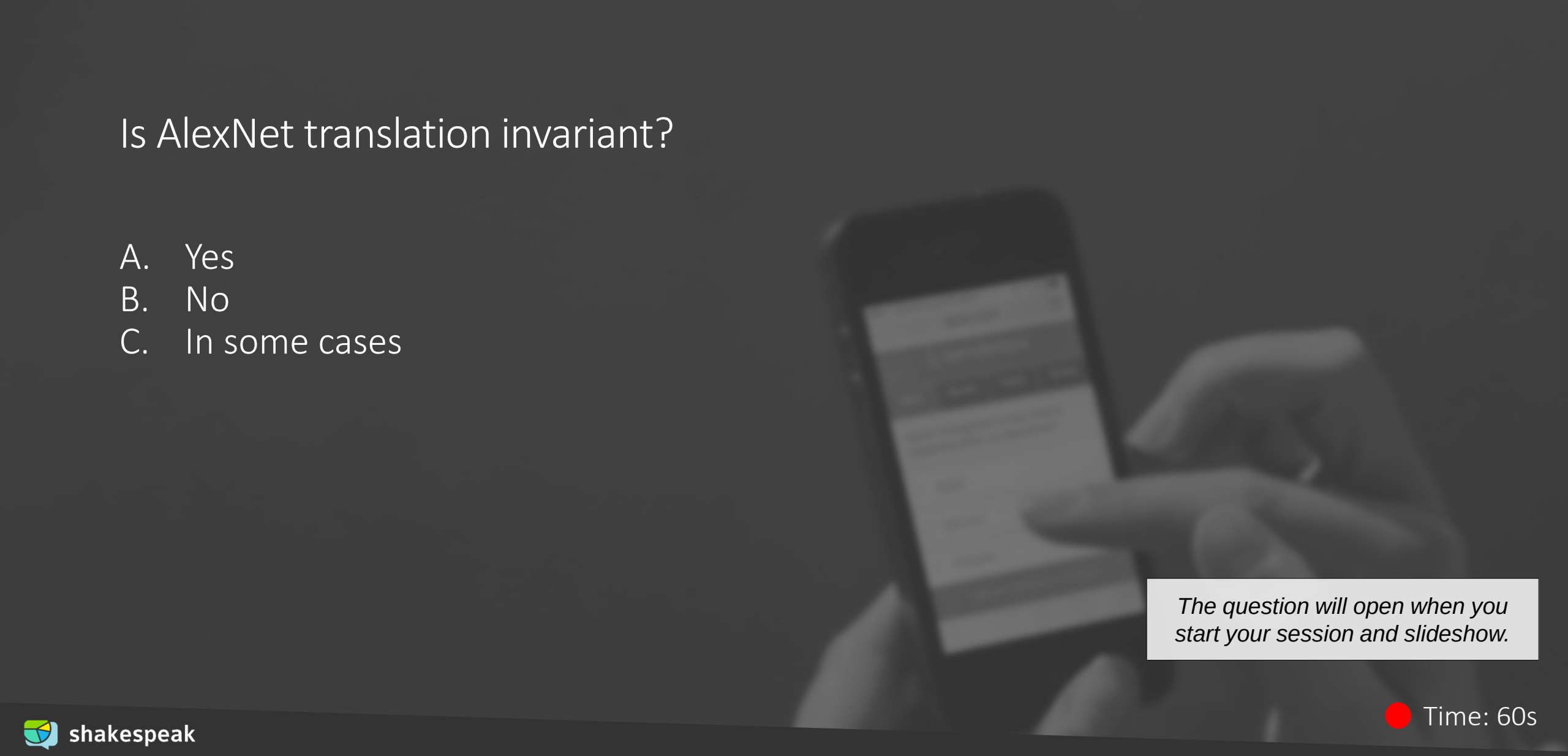
1

2

Voting is anonymous

Is AlexNet translation invariant?

A. Yes
B. No
C. In some cases



The question will open when you start your session and slideshow.

Time: 60s

shakespeak

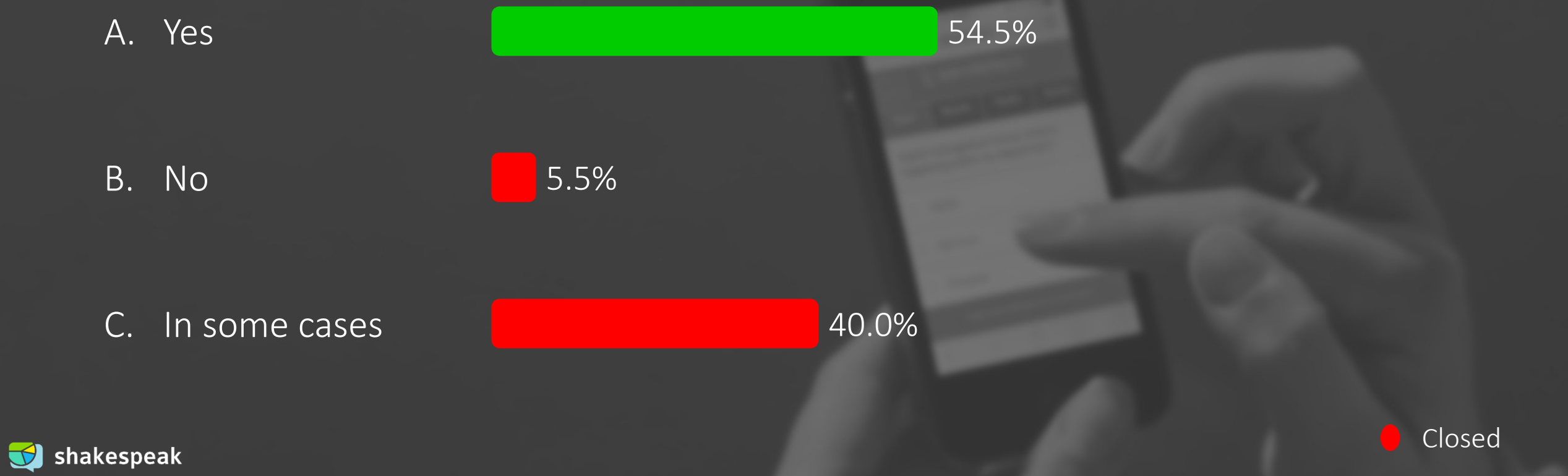
Internet This text box will be used to describe the different message sending methods

TXT The applicable explanation will be shown to the user after you have started the session

This presentation has been loaded without the Shakespeak add-in.
Want to download the add-in for free? Go to http://shakespeak.com/en/free-download/.

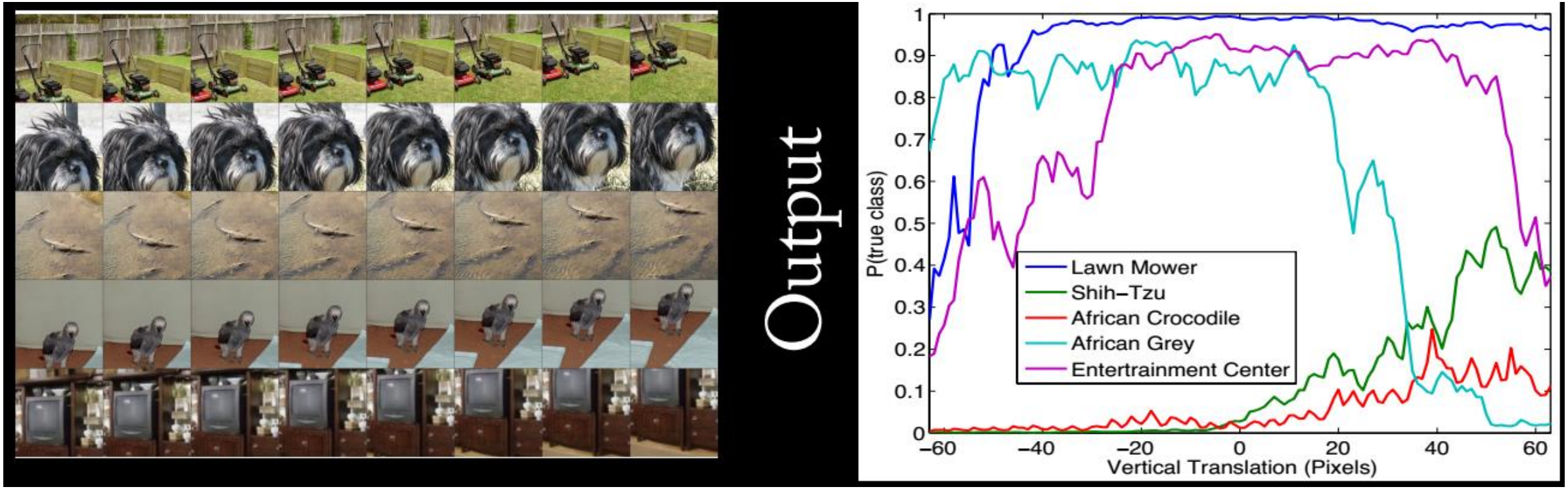
Votes: 55

Is AlexNet translation invariant?



shakespeak

Translation invariance



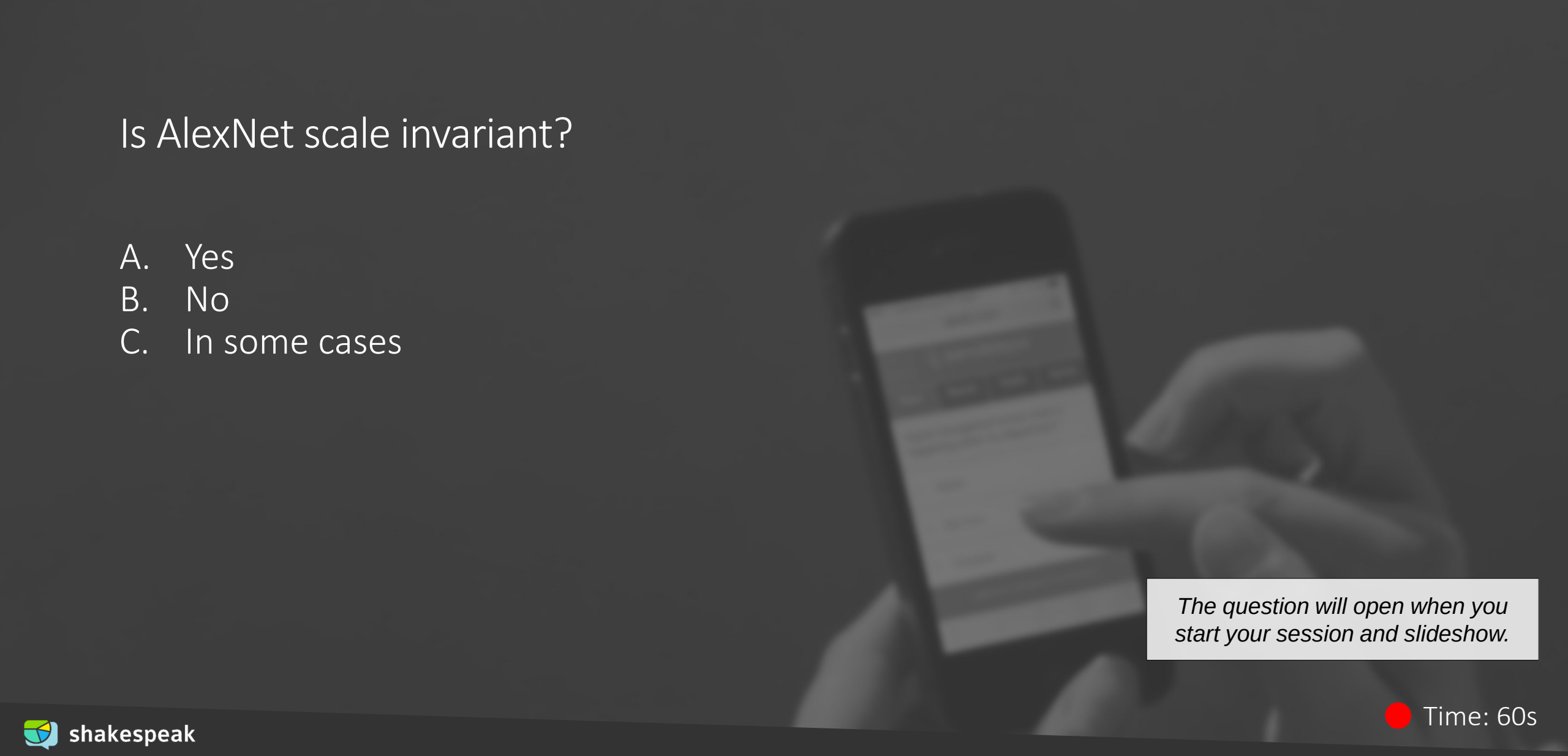
Credit: R. Fergus slides in Deep Learning Summer School 2016

Scale invariance

Credit: R. Fergus slides in Deep Learning Summer School 2016

Is AlexNet scale invariant?

A. Yes
B. No
C. In some cases



The question will open when you start your session and slideshow.

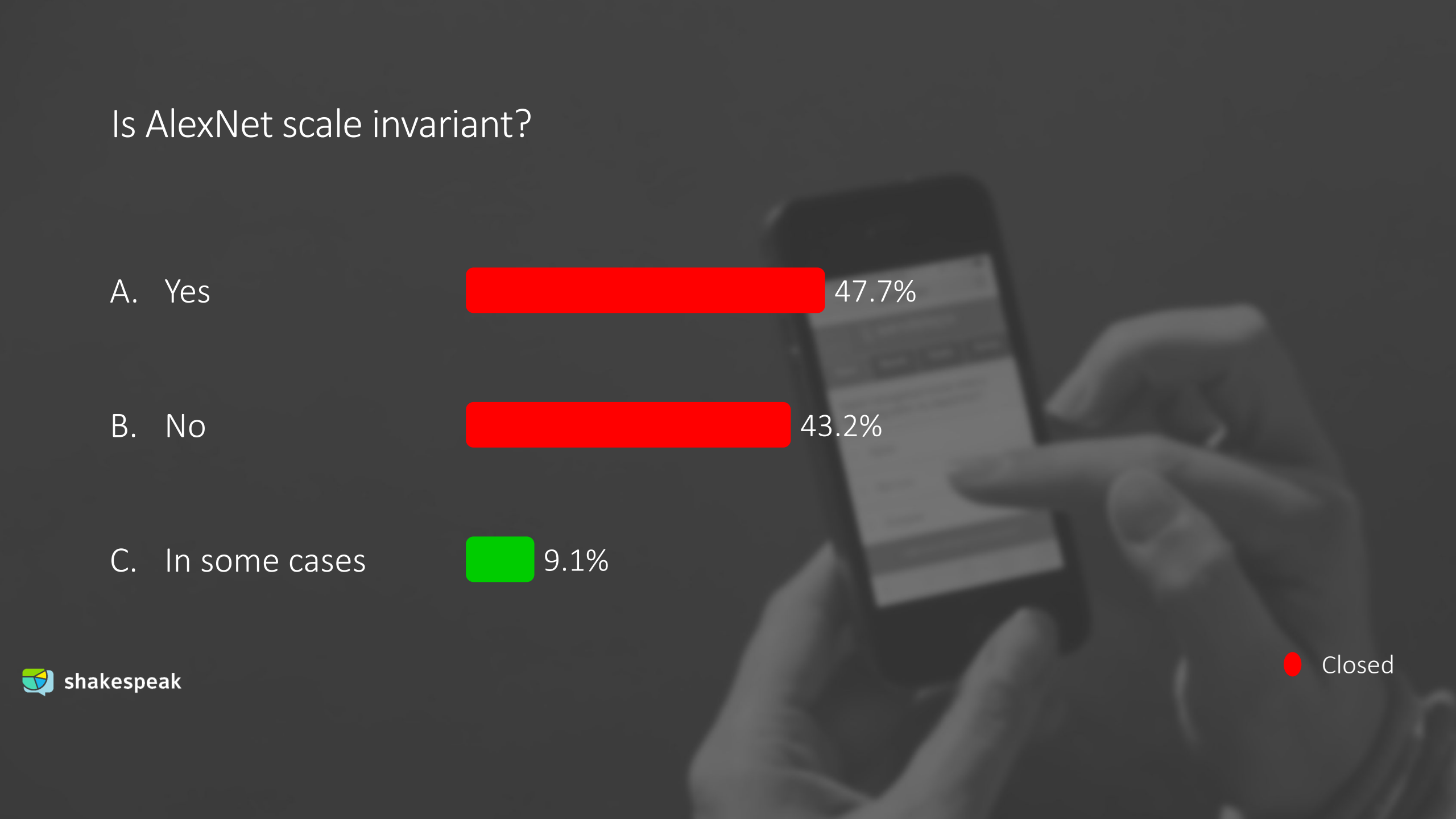
Time: 60s

shakespeak

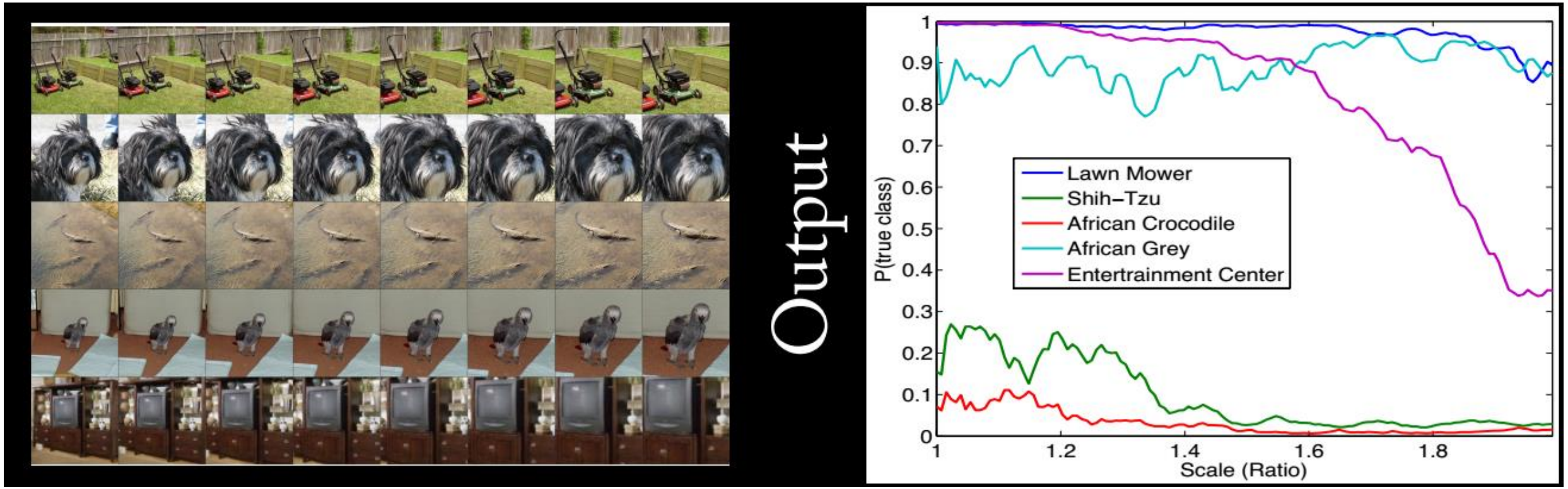
This presentation has been loaded without the Shakespeak add-in.
Want to download the add-in for free? Go to http://shakespeak.com/en/free-download/.

Votes: 44

Is AlexNet scale invariant?

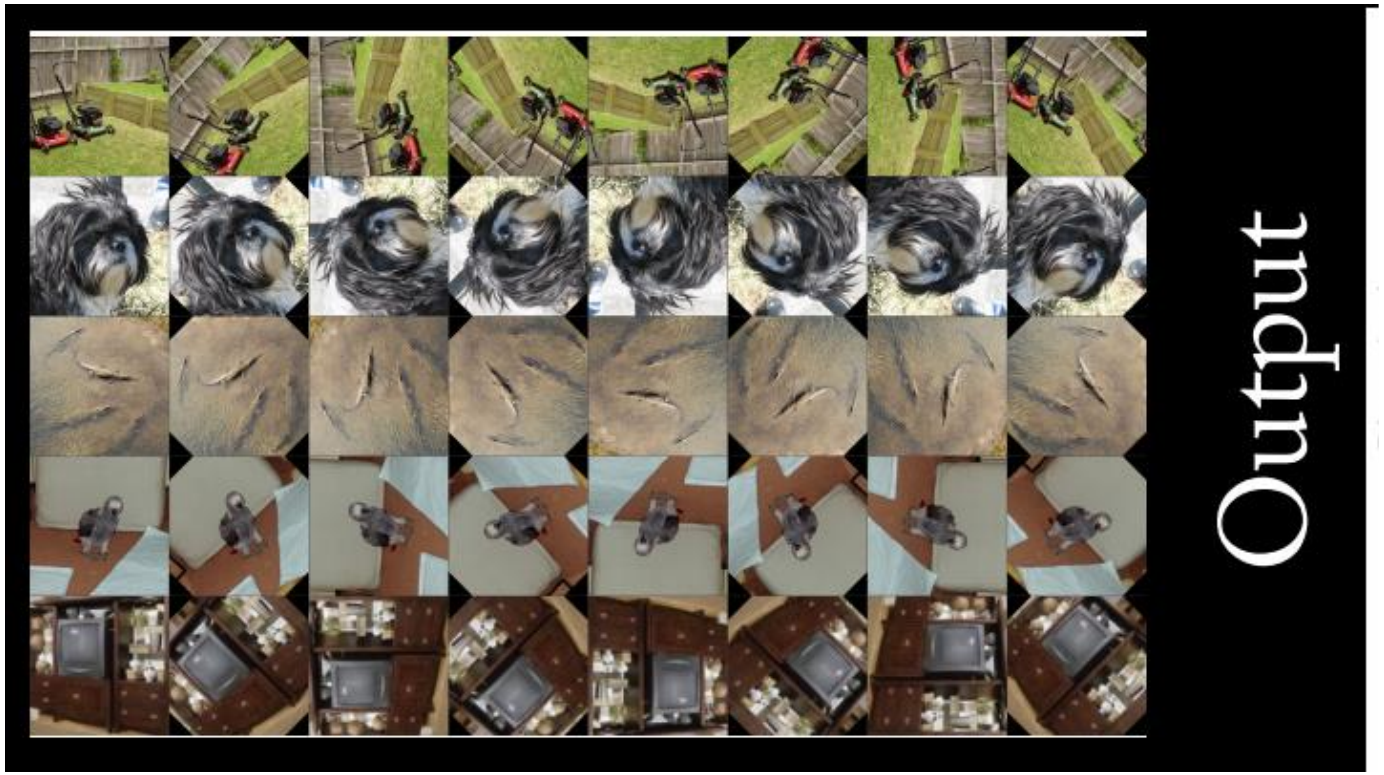


Scale invariance



Credit: R. Fergus slides in Deep Learning Summer School 2016

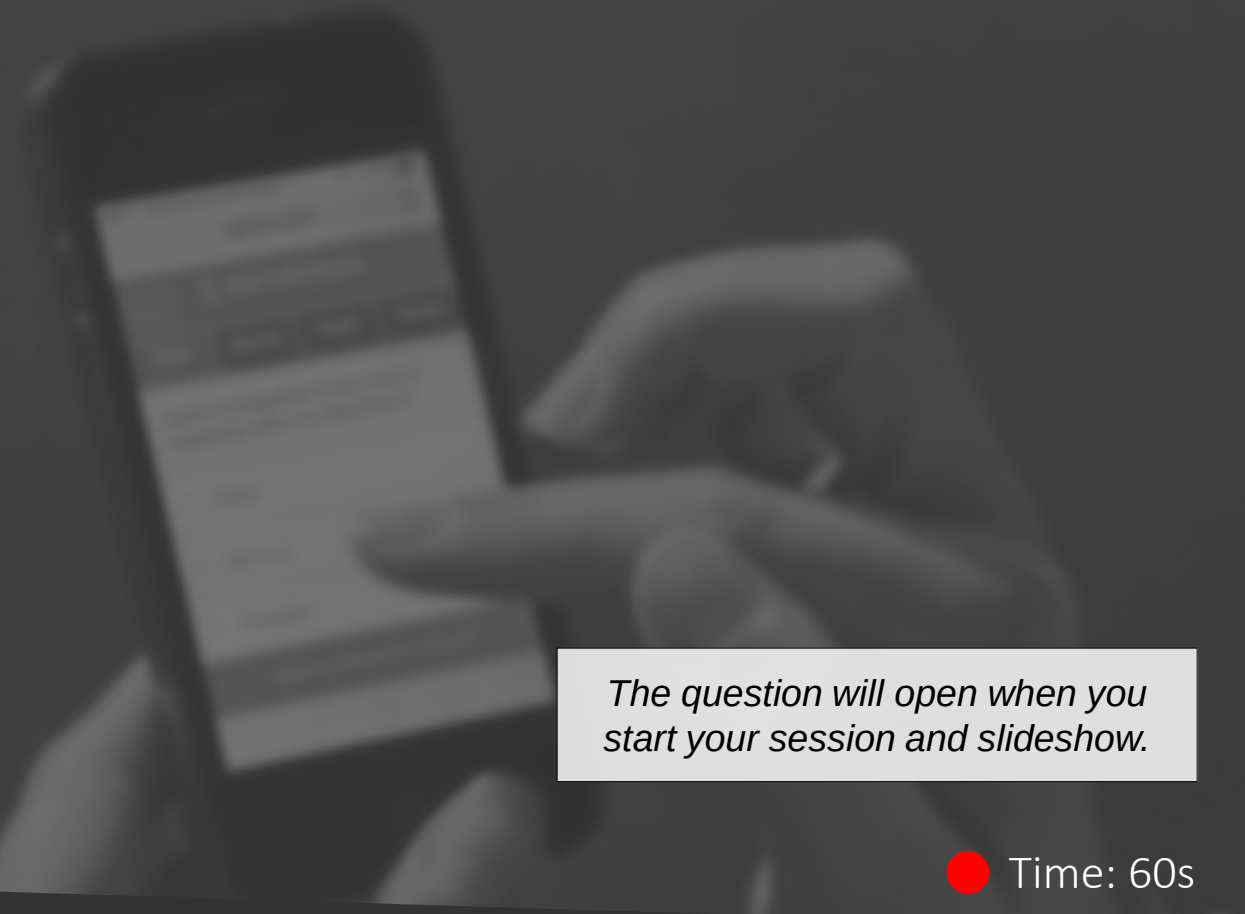
Rotation invariance



Credit: R. Fergus slides in Deep Learning Summer School 2016

Is AlexNet rotation invariant?

A. Yes
B. No
C. In some cases



The question will open when you start your session and slideshow.

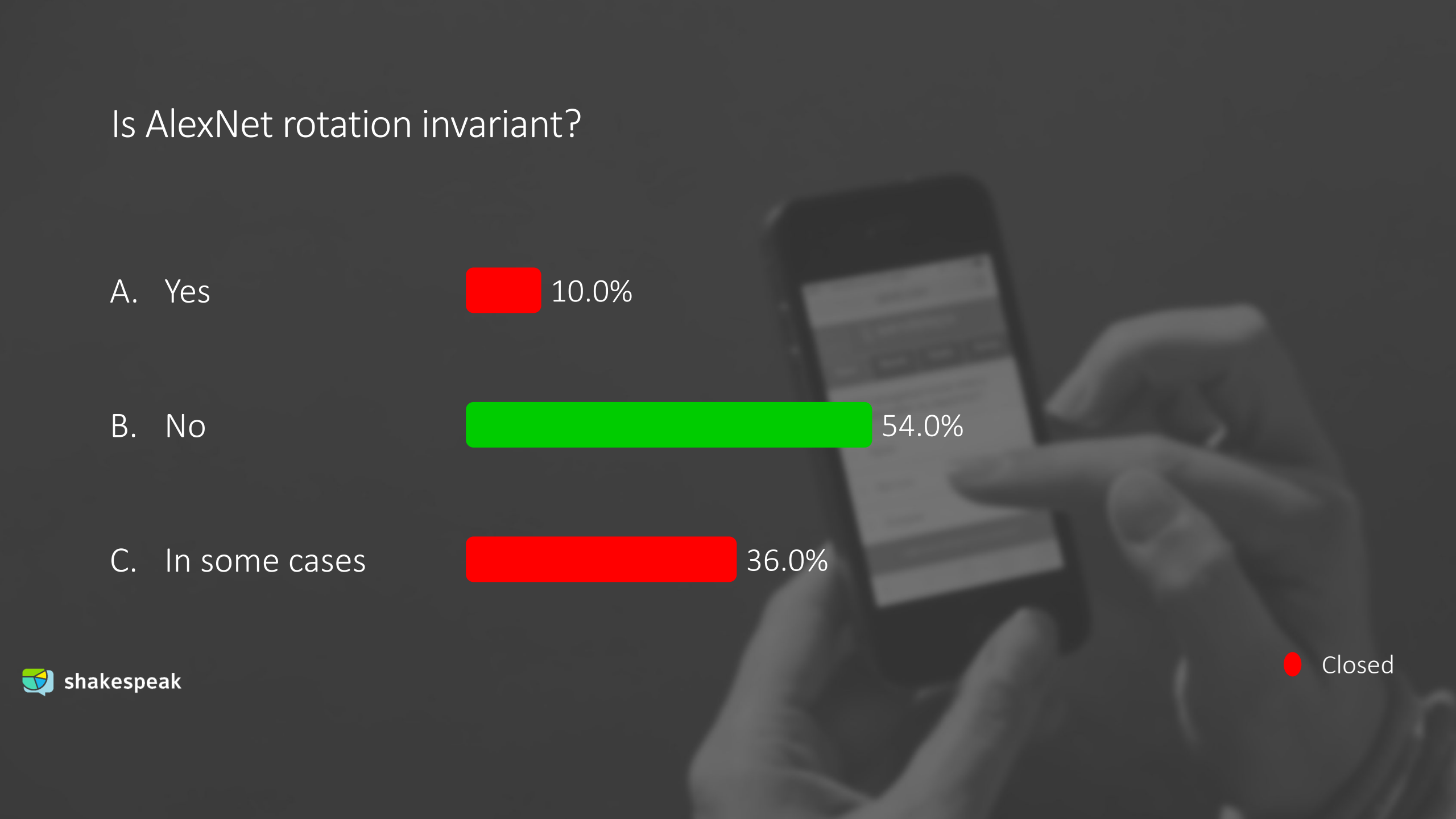
Time: 60s

shakespeak

This presentation has been loaded without the Shakespeak add-in.
Want to download the add-in for free? Go to http://shakespeak.com/en/free-download/.

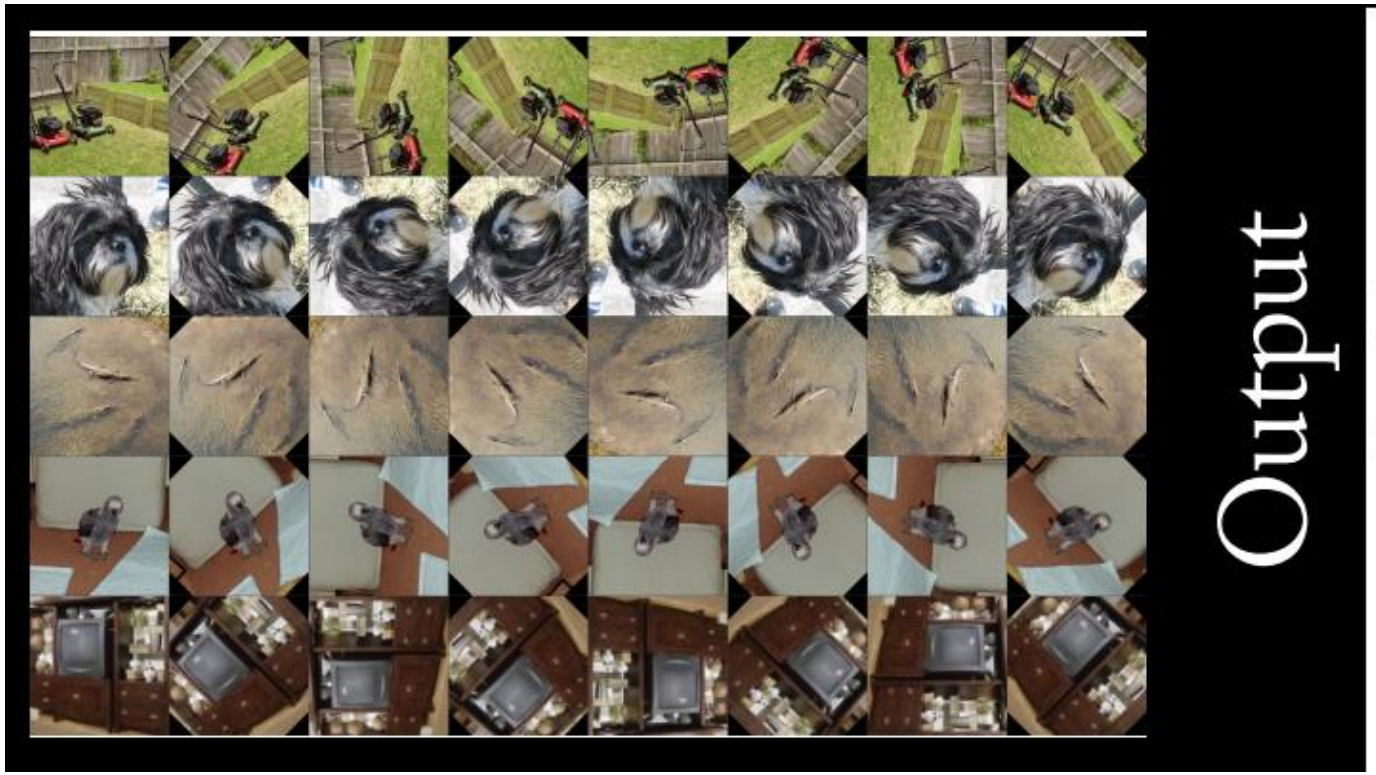
Votes: 50

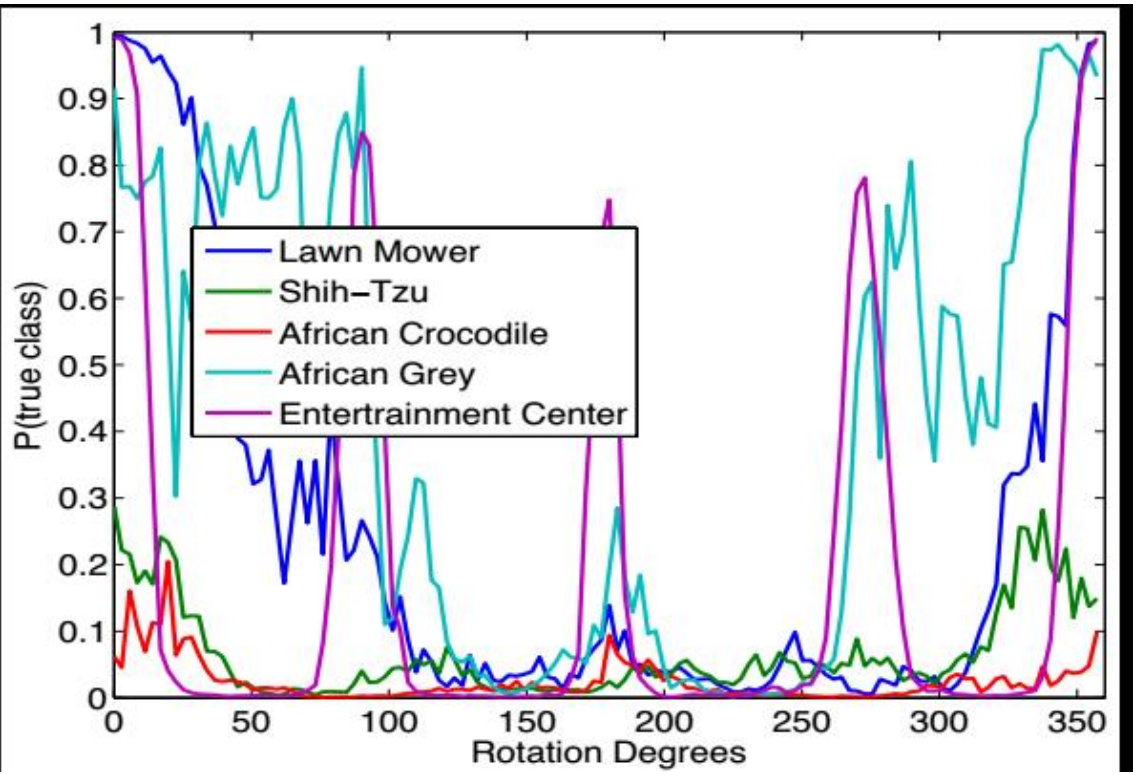
Is AlexNet rotation invariant?



shakespeak

Rotation invariance





Credit: R. Fergus slides in Deep Learning Summer School 2016

Other ConvNets

- VGGnet
- ResNet
- Google Inception module
- Two-stream network
  - Moving images (videos)
- Network in Network
- Deep Fried Network

Summary

- What are the Convolutional Neural Networks?
- Why are they so important for Computer Vision?
- How do they differ from standard Neural Networks?
- How can we train a Convolutional Neural Network?

Reading material & references

- Chapter 9

Next lecture

- What do convolutions look like?
- Build on the visual intuition behind Convnets
- Deep Learning Feature maps
- Transfer Learning