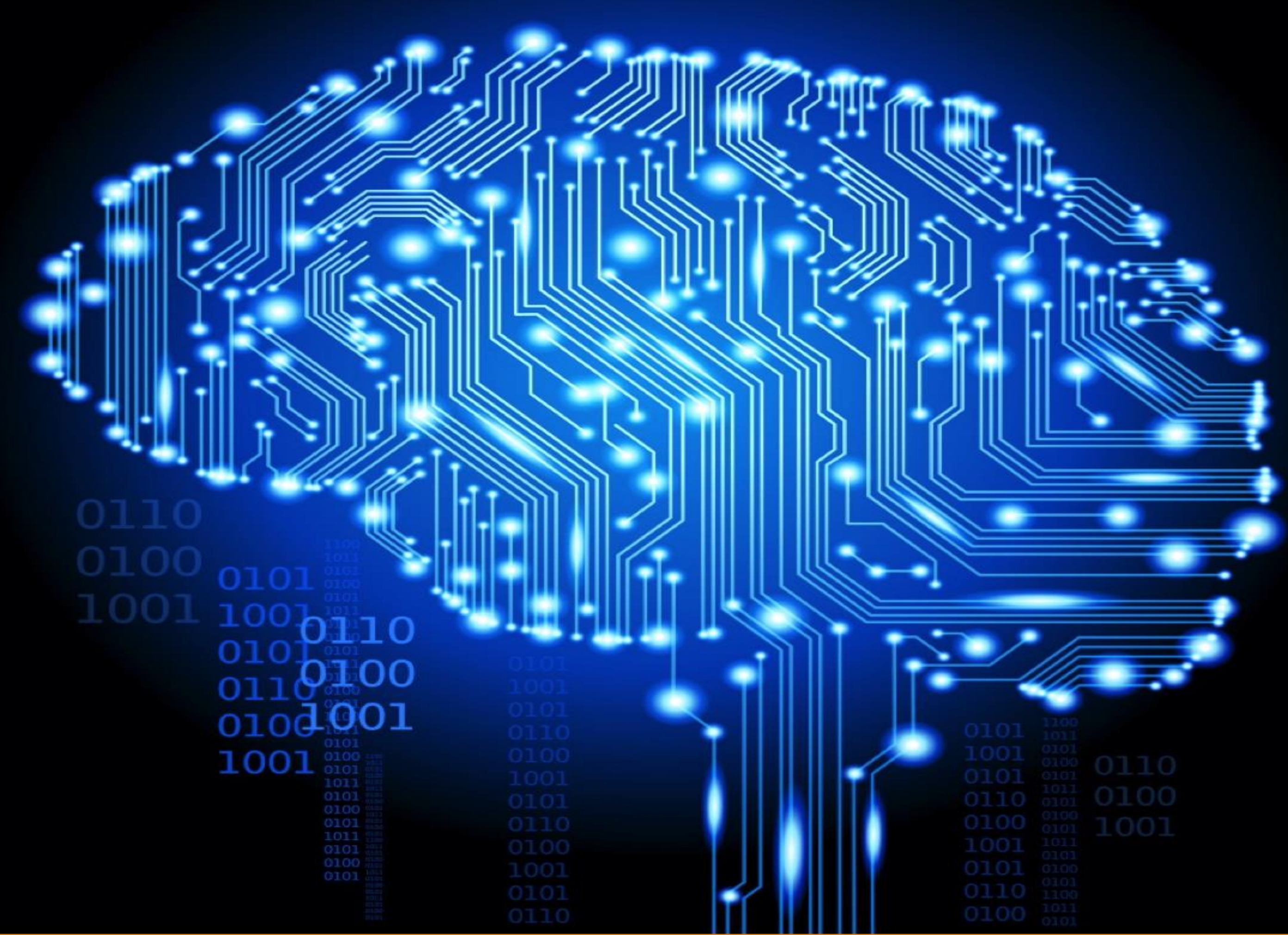


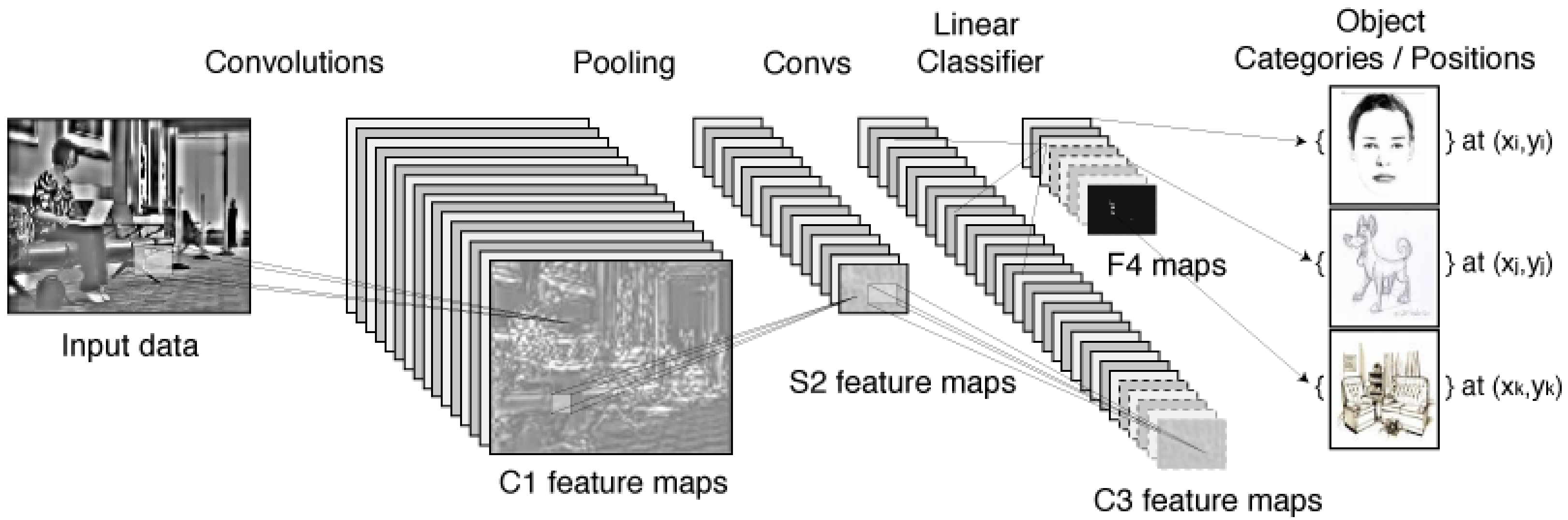
Lecture 4: Convolutional Neural Networks

Deep Learning @ UvA

Lecture overview

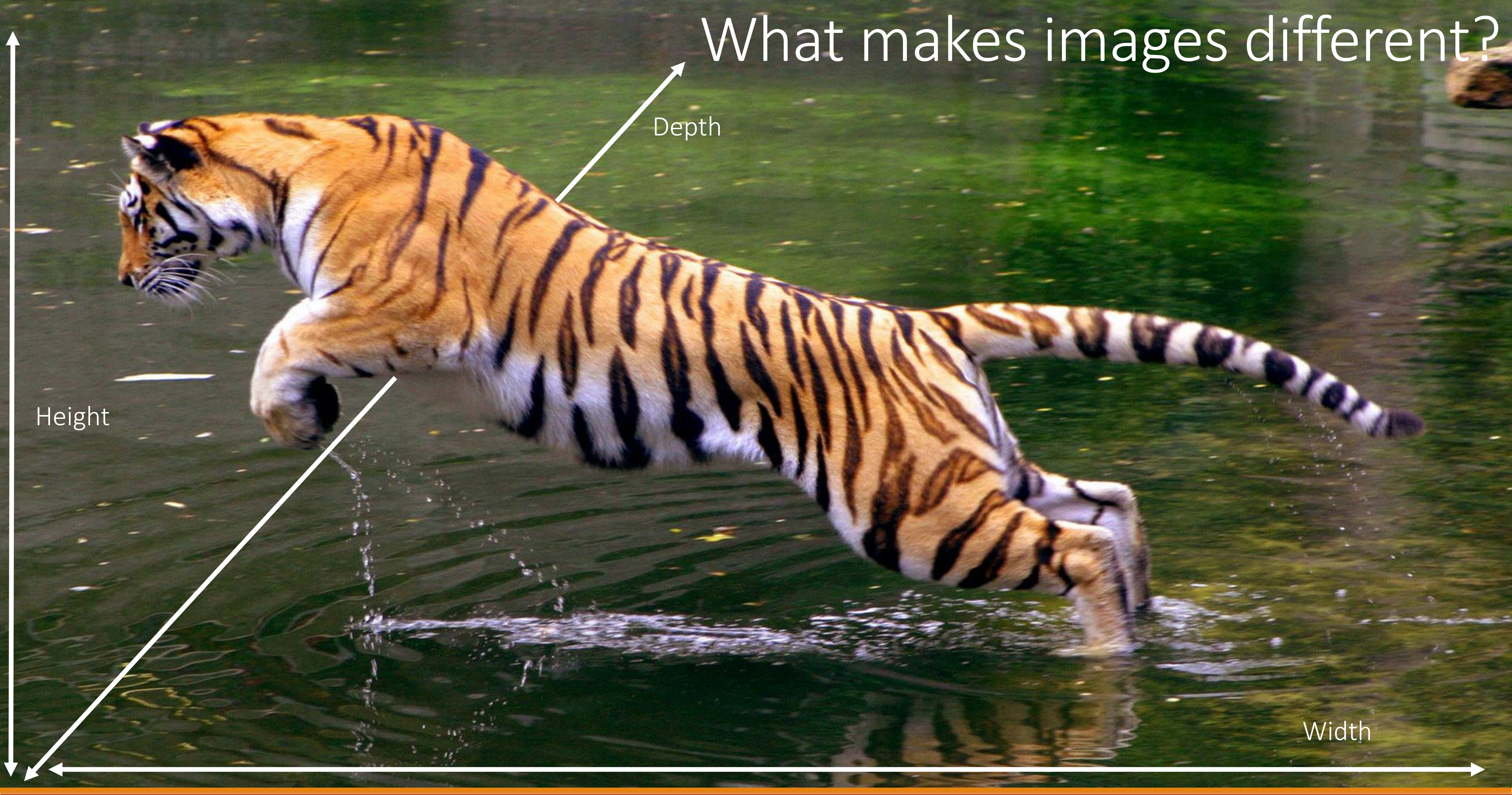
- What are the Convolutional Neural Networks?
- Differences from standard Neural Networks
- Why are they important in Computer Vision?
- How to train a Convolutional Neural Network?

Convolutional Neural Networks

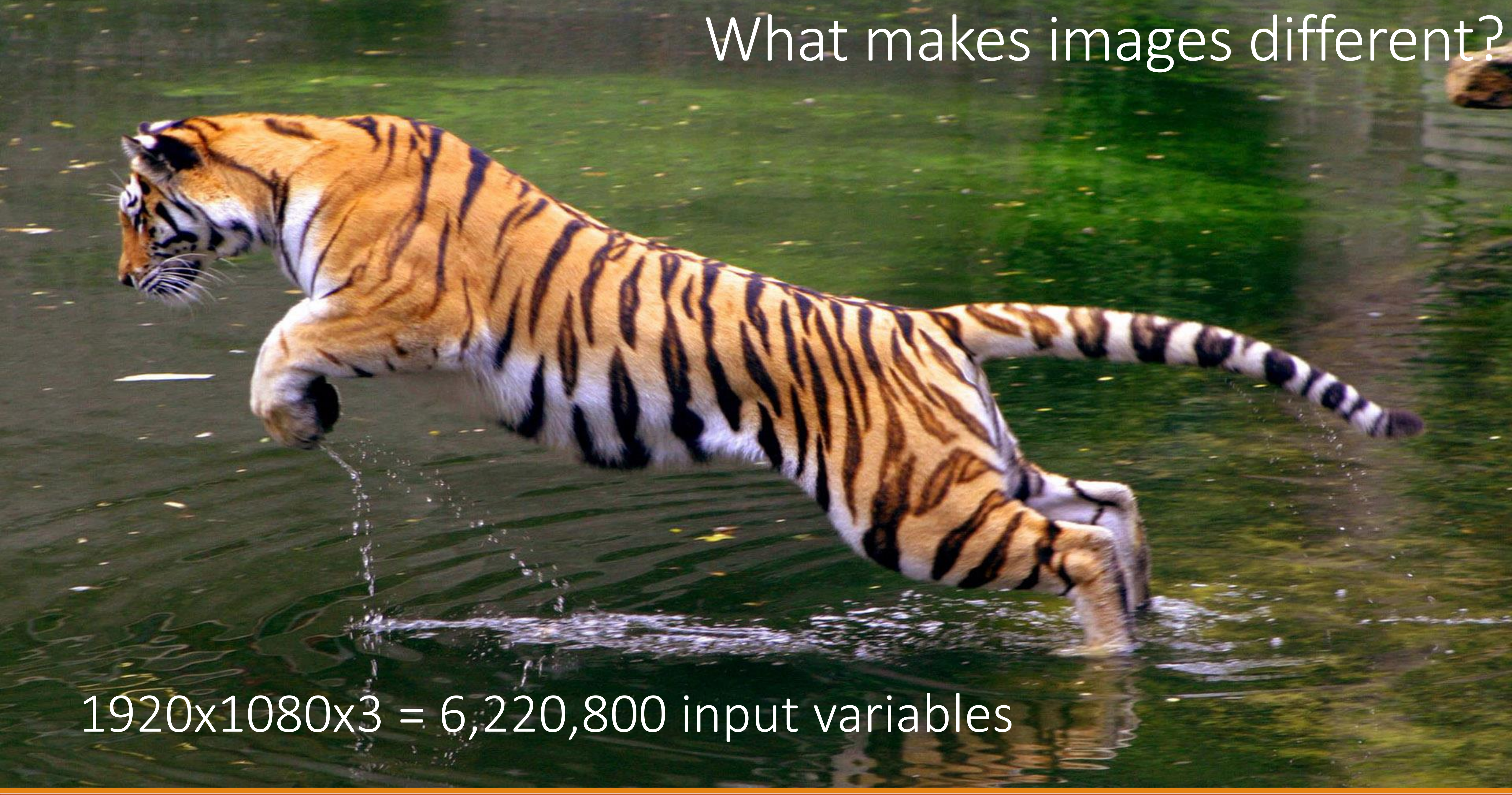


What makes images different?

What makes images different?



What makes images different?



1920x1080x3 = 6,220,800 input variables

What makes images different?

What makes images different?

What makes images different?

Image has shifted a bit to the up and the left!

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)
```

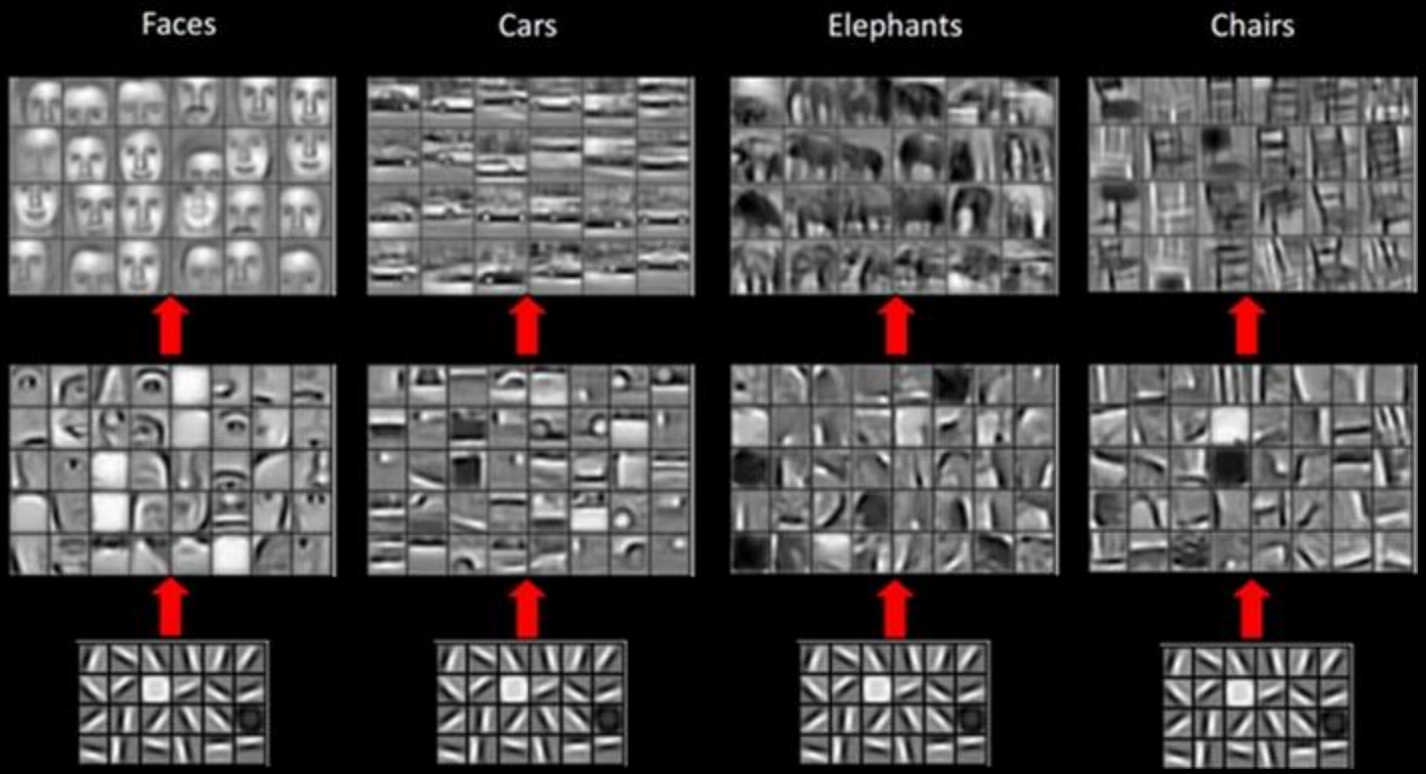
Convolutional Neural Networks

- Preserve spatial structure by convolutional filters

- Tackle huge input dimensionalities by local connectivity and parameter sharing

- Robust to local variances by spatial pooling

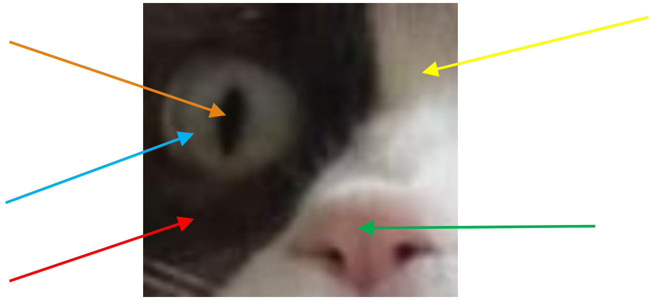
Preserving spatial structure

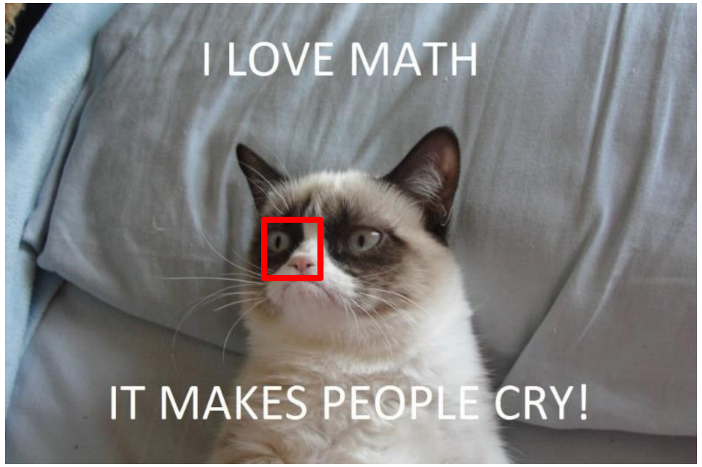


Why spatial?

- 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

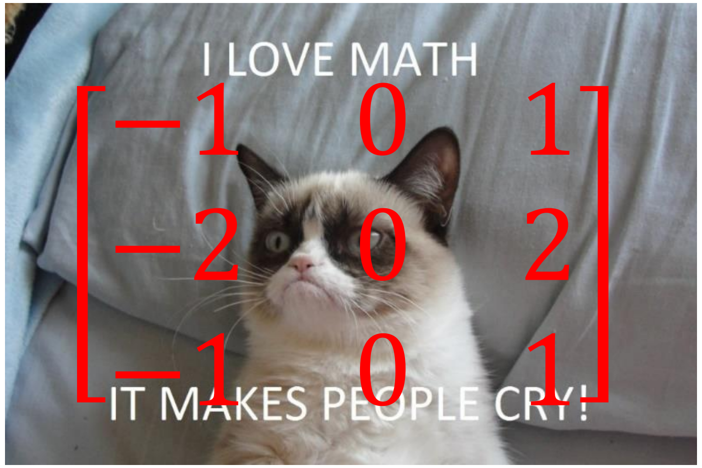


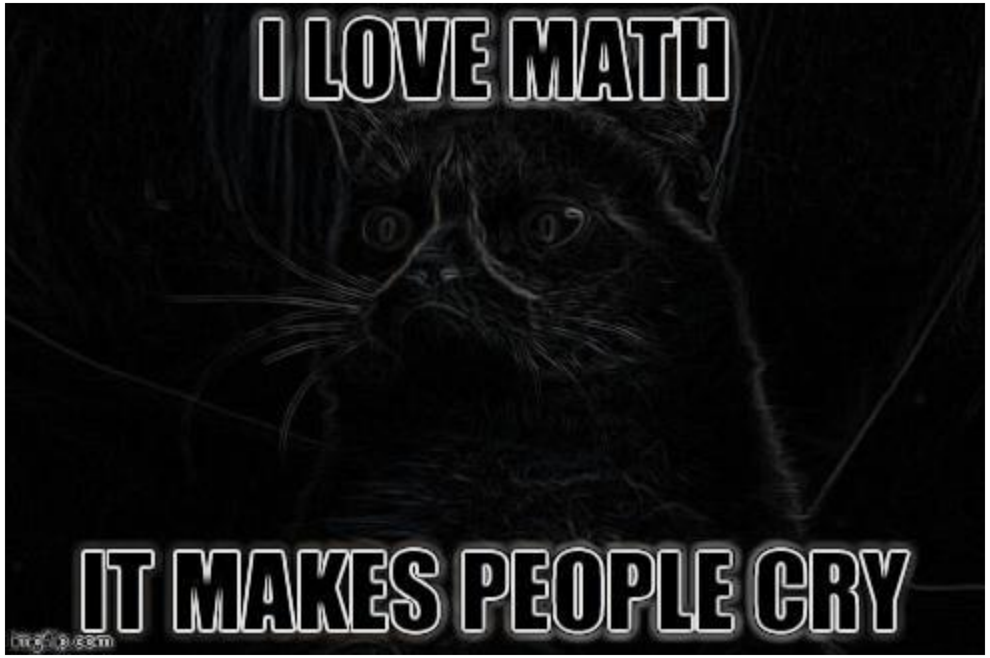


Example filter when K=1

e.g. Sobel 2-D filter

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





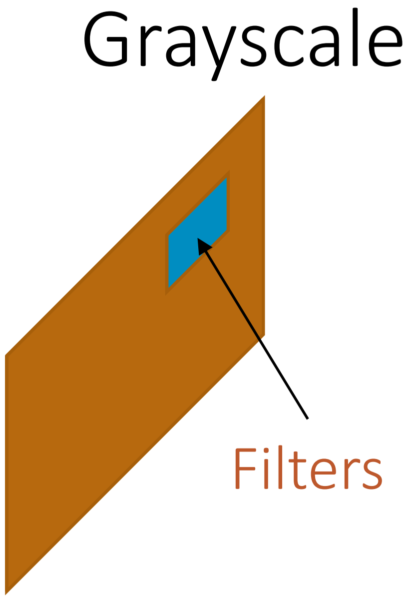
Learnable filters

- Image processing and Computer Vision has many handcrafted filters
  - Canny, Sobel, Gaussian blur, smoothing, low-level segmentation, morphological filters, Gabor filters
- Are they optimal for recognition?
- Can we learn optimal filters from our data instead?
- Are they going resemble the handcrafted filters?

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}$$

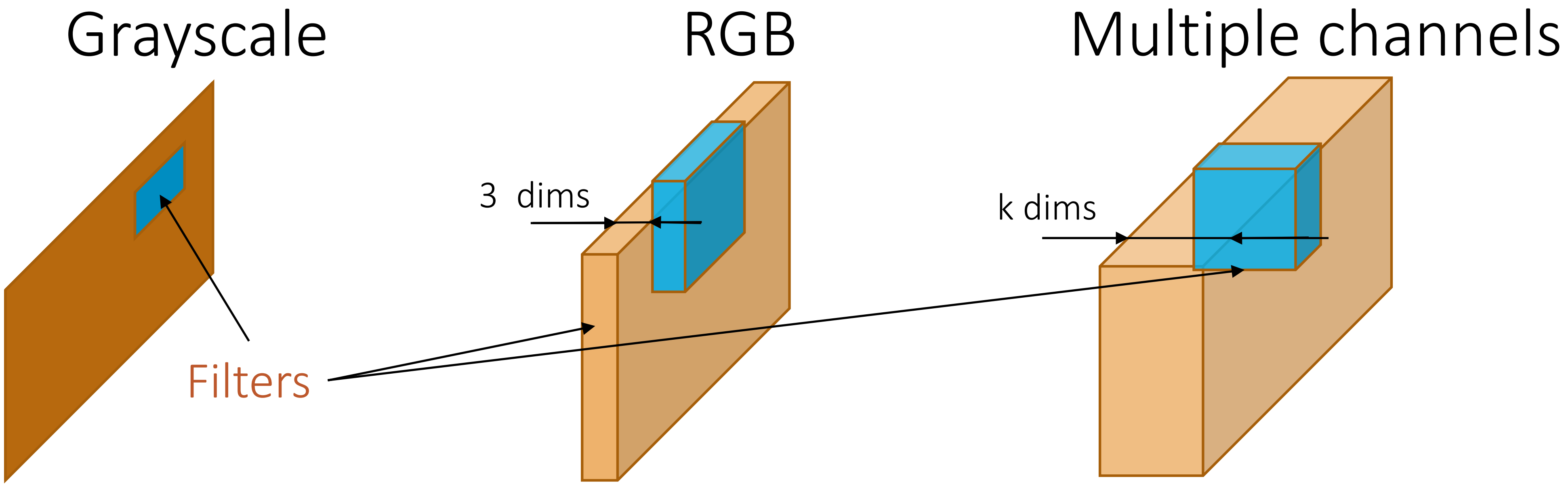
2-D Filters (Parameters)

- 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



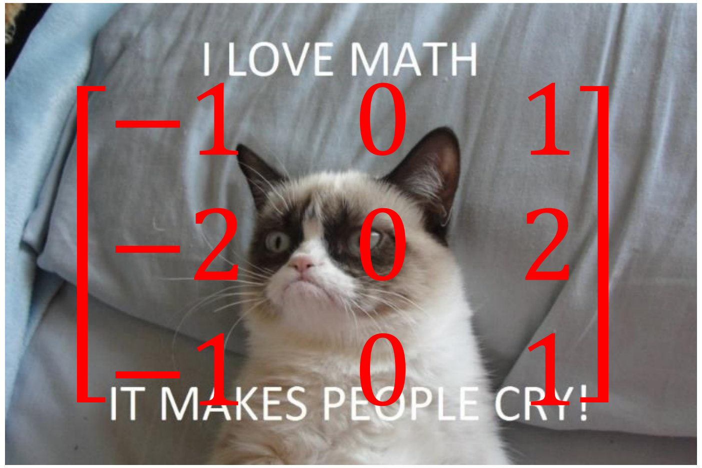
K-D Filters (Parameters)

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

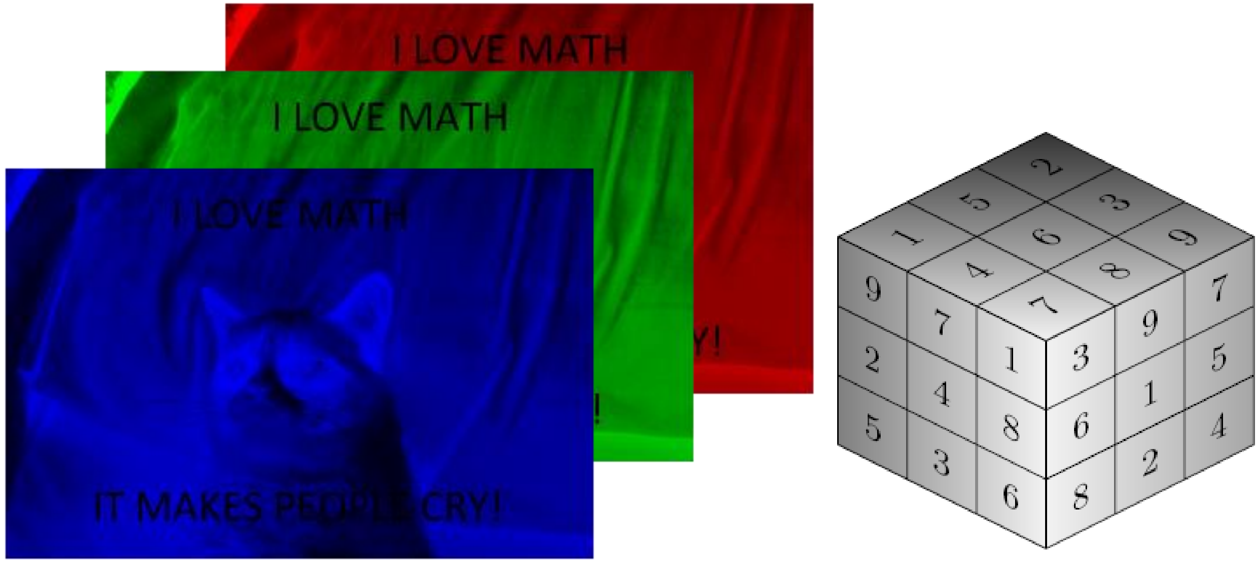


What does a 3-D (k-D) filter look like?

2-D filter

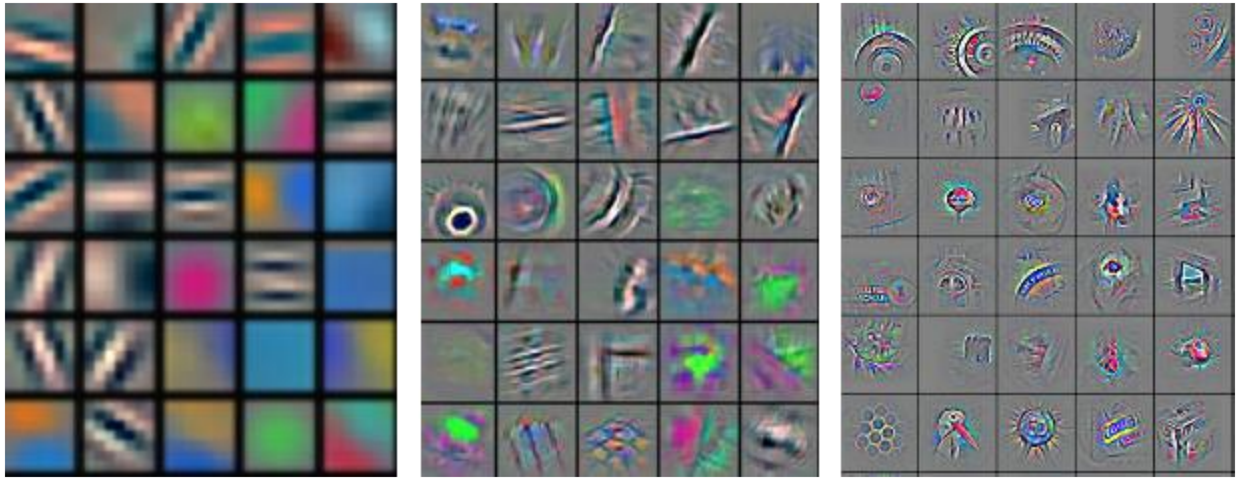


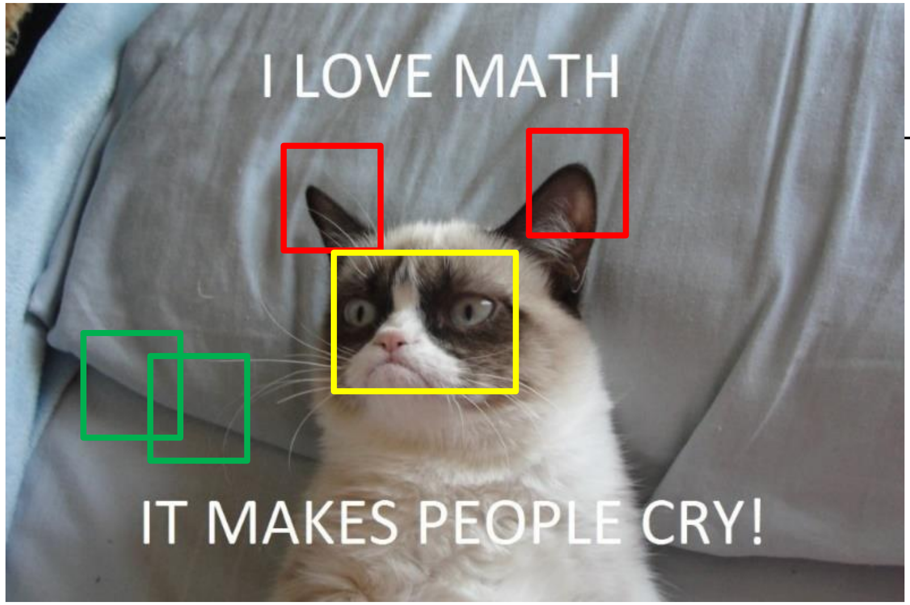
3-D filter

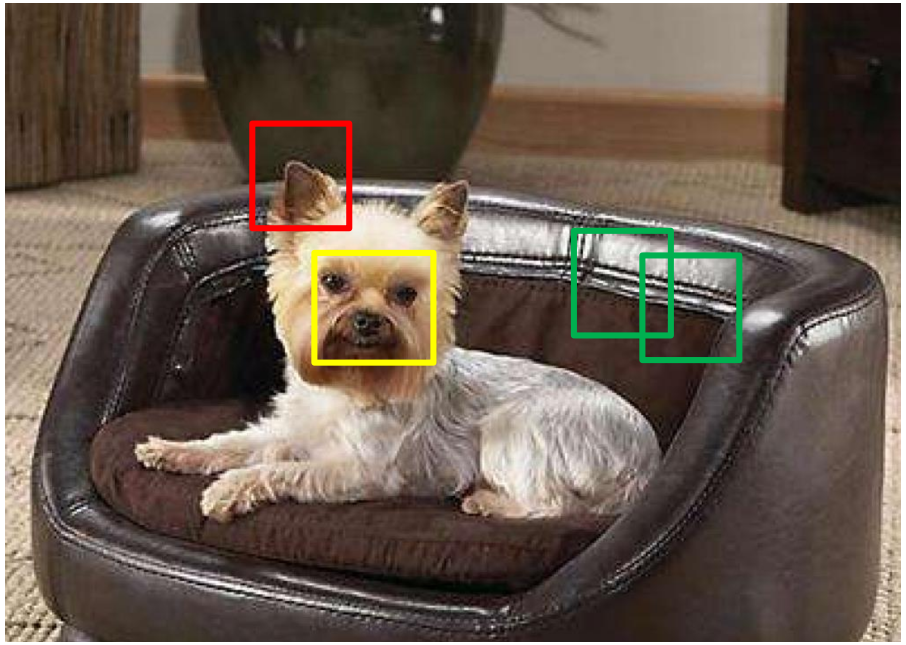


Hypothesis

- Image statistics are not location dependent
  - Natural images are stationary
- The same filters should work on every corner of the image similarly
- Perhaps move and reuse the same (red, yellow, green) filter across the whole image?







Moving 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

Convolving the image

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

Result

4		

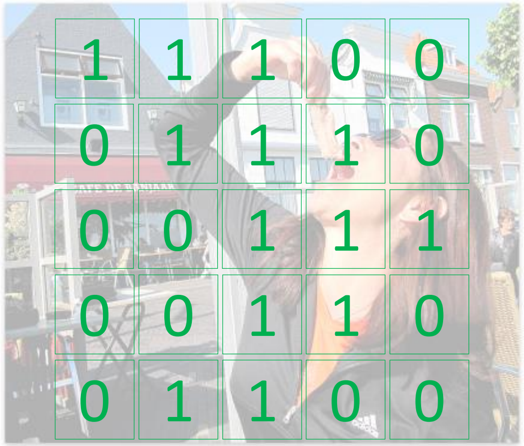
Convolutional filter 1

1	0	1
0	1	0
1	0	1

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

Shared 2-D filters → Convolutions

Original image

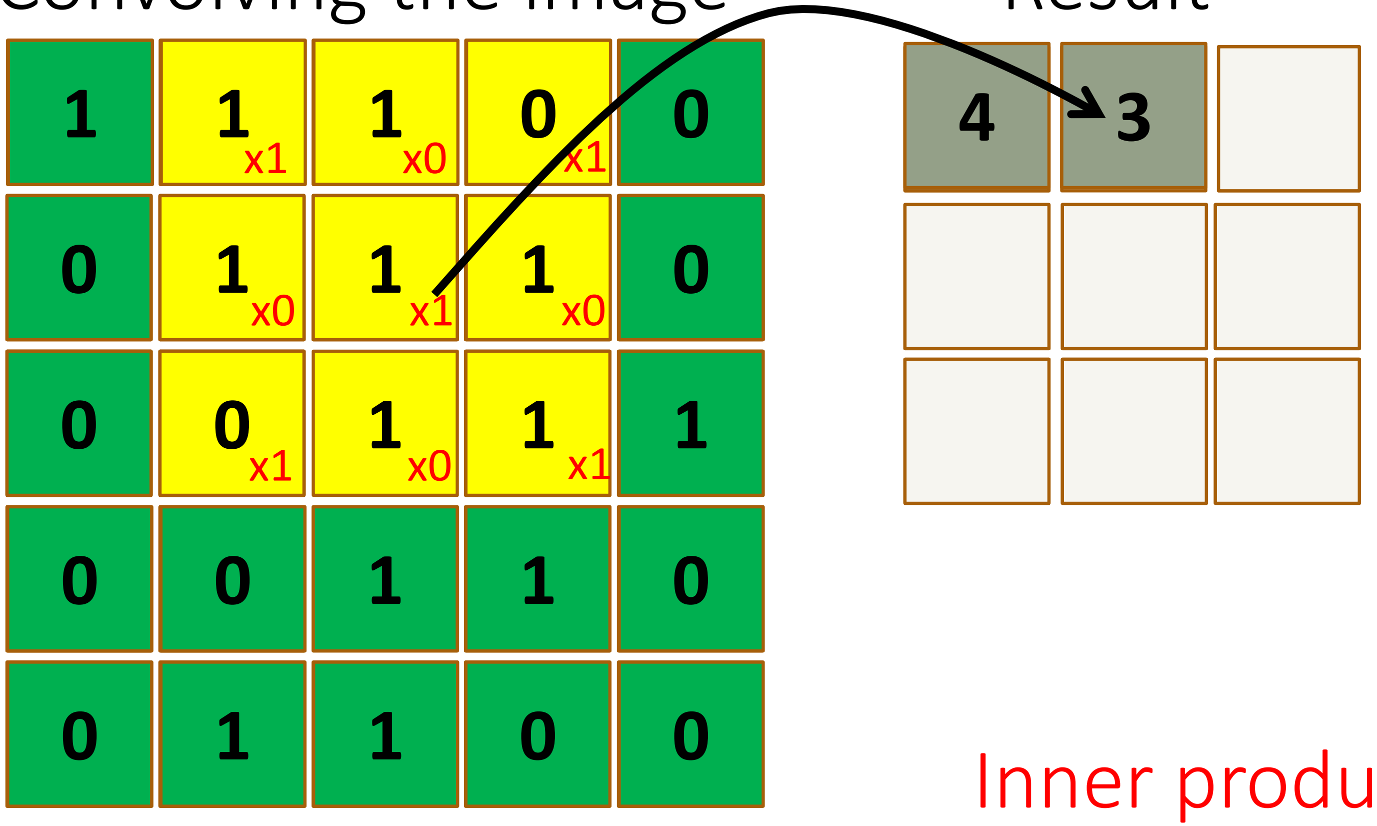


Convolutional filter 1

1	0	1
0	1	0
1	0	1

Convolving the image

Result



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

Shared 2-D filters → Convolutions

Original image

Convolutional filter 1

1	0	1
0	1	0
1	0	1

Convolving the image

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

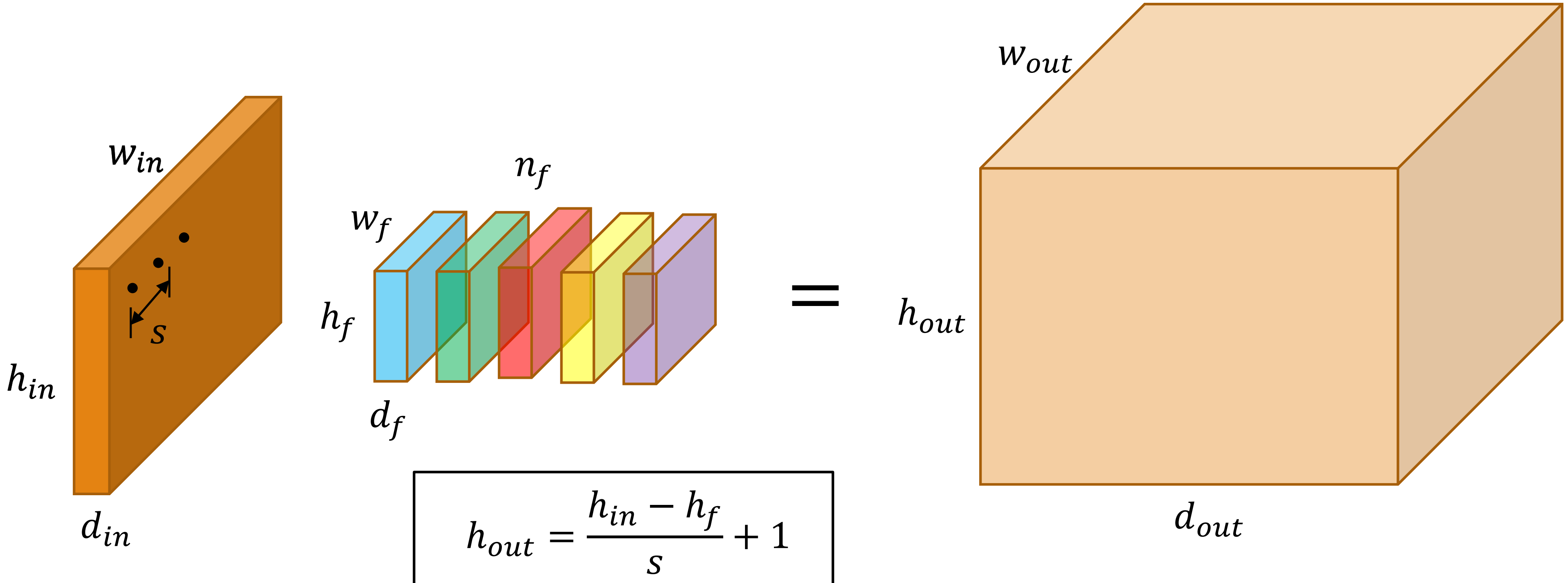
Result

4	3	4
2	4	3
2	3	4

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

Inner product

Output dimensions after convolution



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

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

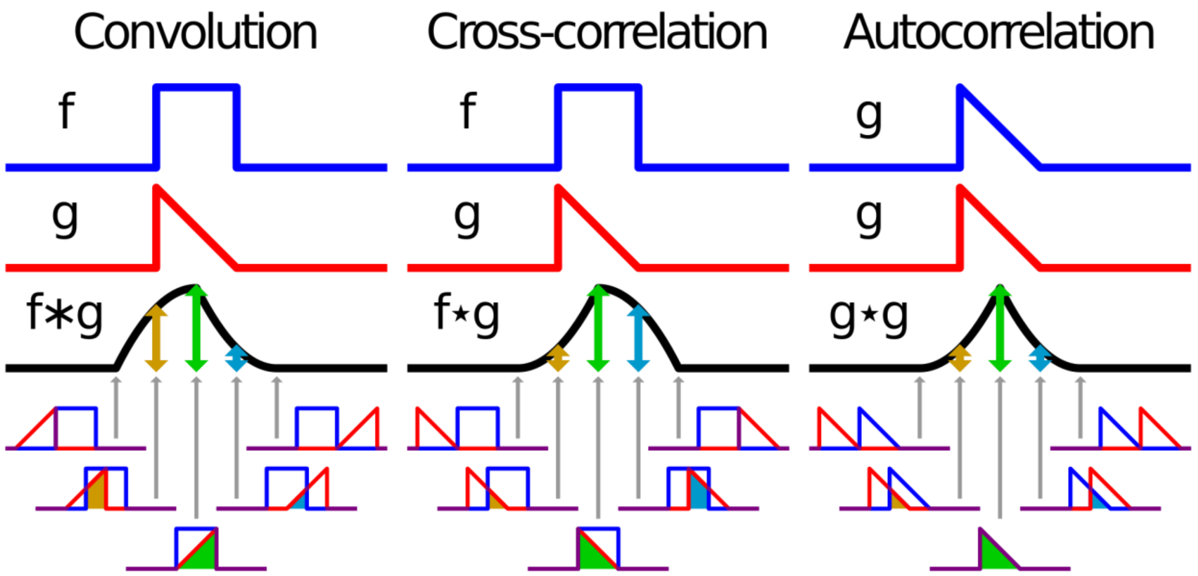
$$d_{out} = n_f$$

$$d_f = d_{in}$$

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*

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



Convolutional module

- Activation function

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

- Essentially a dot product, similar to linear layer

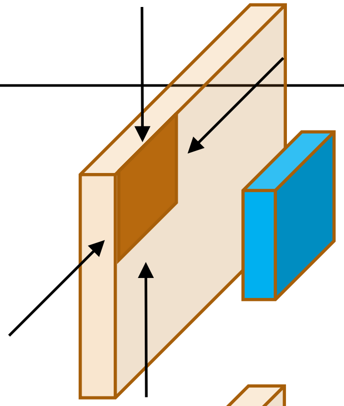
$$a_{rc} \sim x_{region}^{T} \cdot w$$

- Gradient w.r.t. the parameters

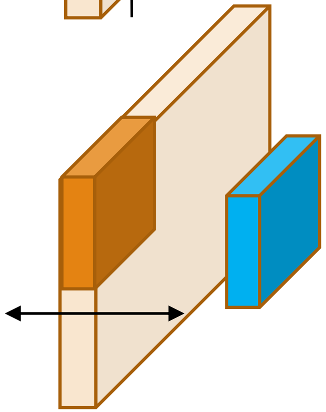
$$\frac{\partial a_{rc}}{\partial w_{ij}} = \sum_{i=-a}^{a} \sum_{j=-b}^{b} x_{r-i,c-j}$$

Local connectivity

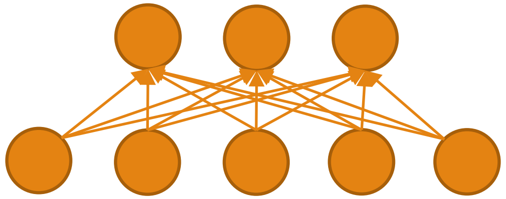
- Local connectivity: weight connections are surface-wise local!
  - The blue filter weights connect only to local orange pixels along the surface



- The weights connections are depth-wise global
  - The blue filter weights connect to all orange channels across the depth

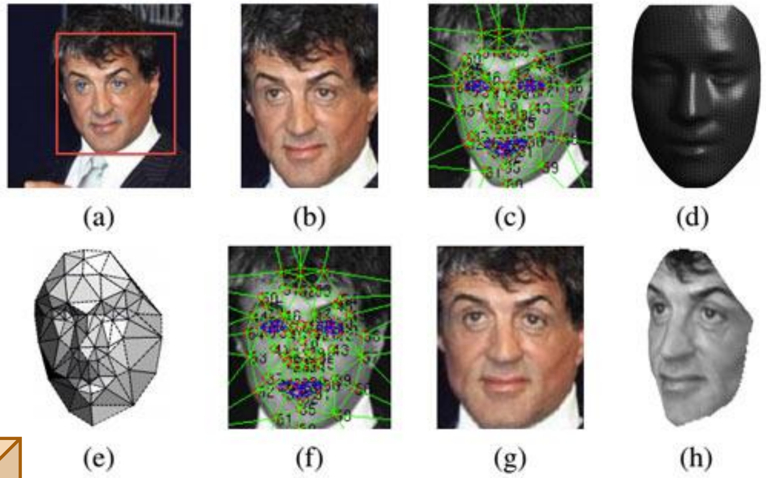


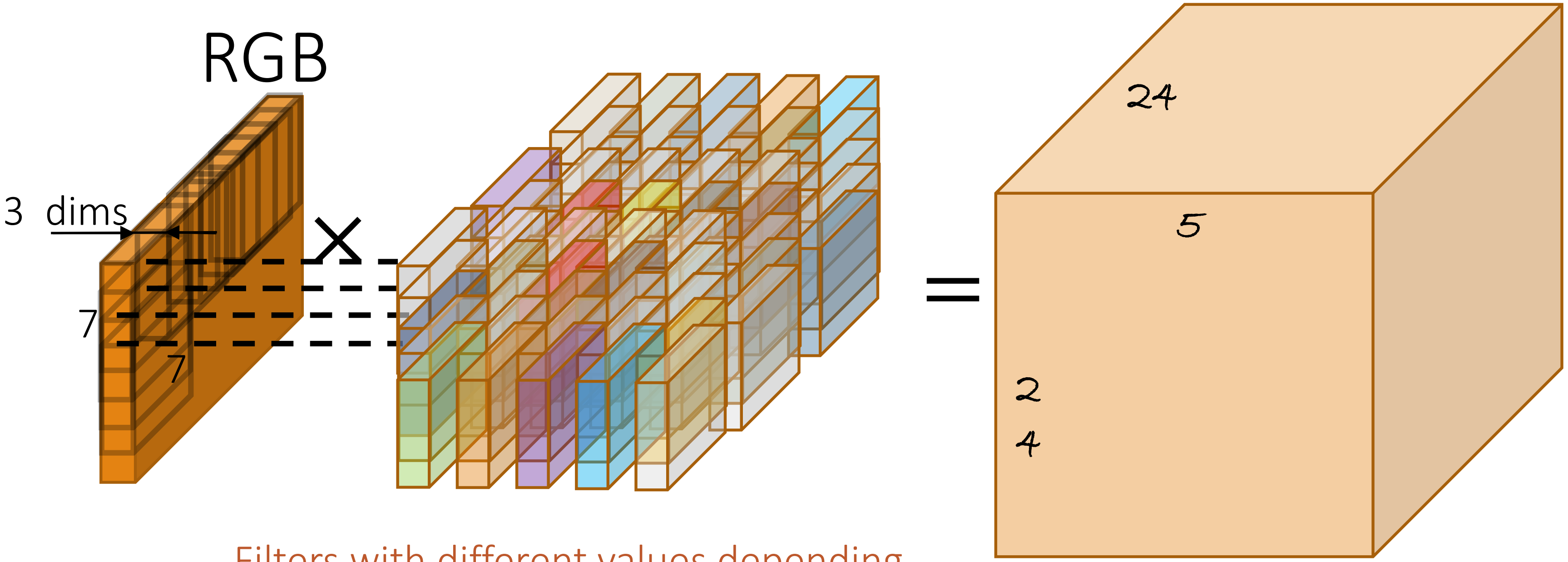
- For standard neurons no local connectivity
  - Everything is connected to everything
  - No notion of surface or depth
  - We might as well shuffle the pixels, there is no difference



Local connectivity ≠ Convolutional filters

- Local but ***non-shareable*** filters are possible
- Still useful for some applications





Filters with different values depending on image location and channel ID

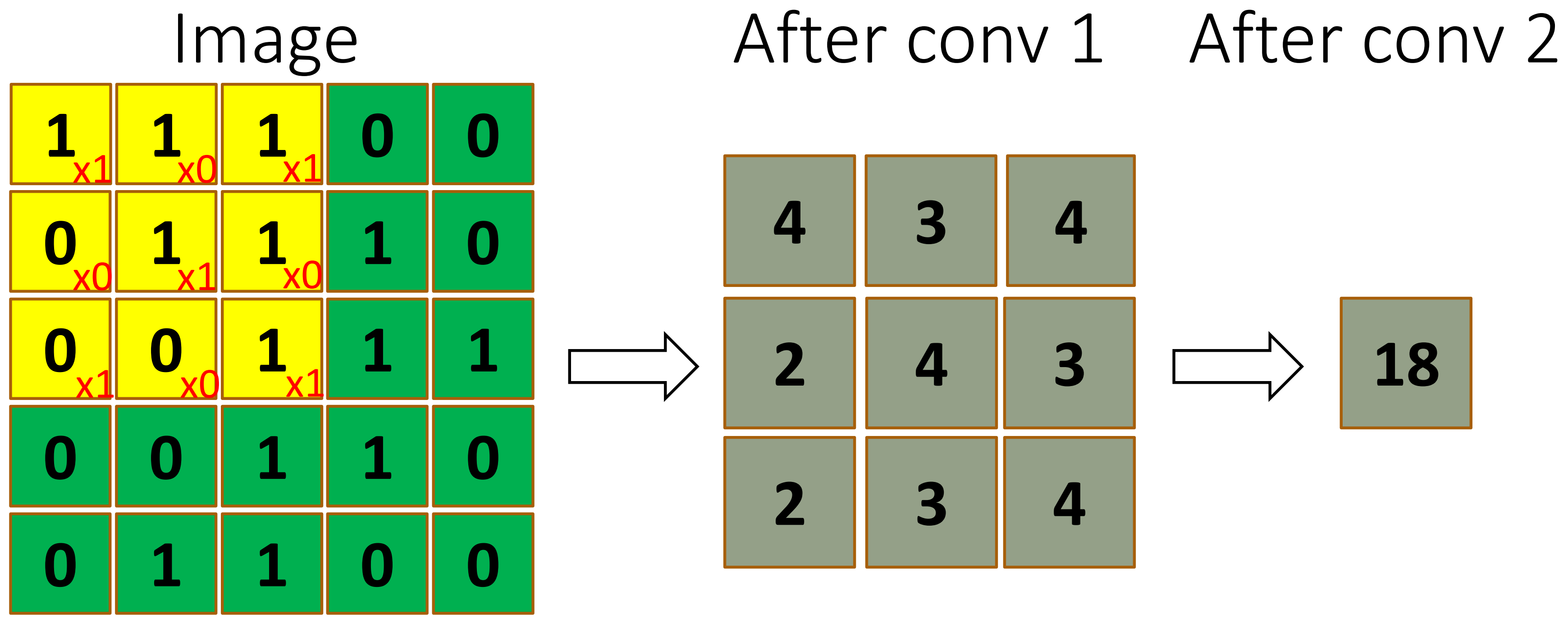
Assume the image is 30x30x3.
1 filter every pixel (stride =1)
How many parameters in total?

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

Convolutions reduce dimensionality

- Our images get smaller and smaller
- We run out of "latent pixels" → not too deep architectures
- Details are lost → recognition accuracy drops



Zero-padding to maintain input dimensionality

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

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

Good practices

- Resize the image to have a size in the power of 2
- 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

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
  - $\frac{\partial a_{rc}}{\partial x_{ij}} = \begin{cases} 1, & if\ i = \mathrm{i}_{\max}, \mathrm{j} = \mathrm{j}_{\max} \\ 0, & otherwise \end{cases}$
- Average pooling
  - $\frac{\partial a_{rc}}{\partial x_{ij}} = \frac{1}{r \cdot c}$

1	4	3	6
2	1	0	9
2	2	7	7
5	3	3	6

4	9
5	7

1	4	1	6
2	3	0	9
1	2	7	1
4	1	0	2

2.5	4
2	2.5

ConvNet Case Study I: Alexnet

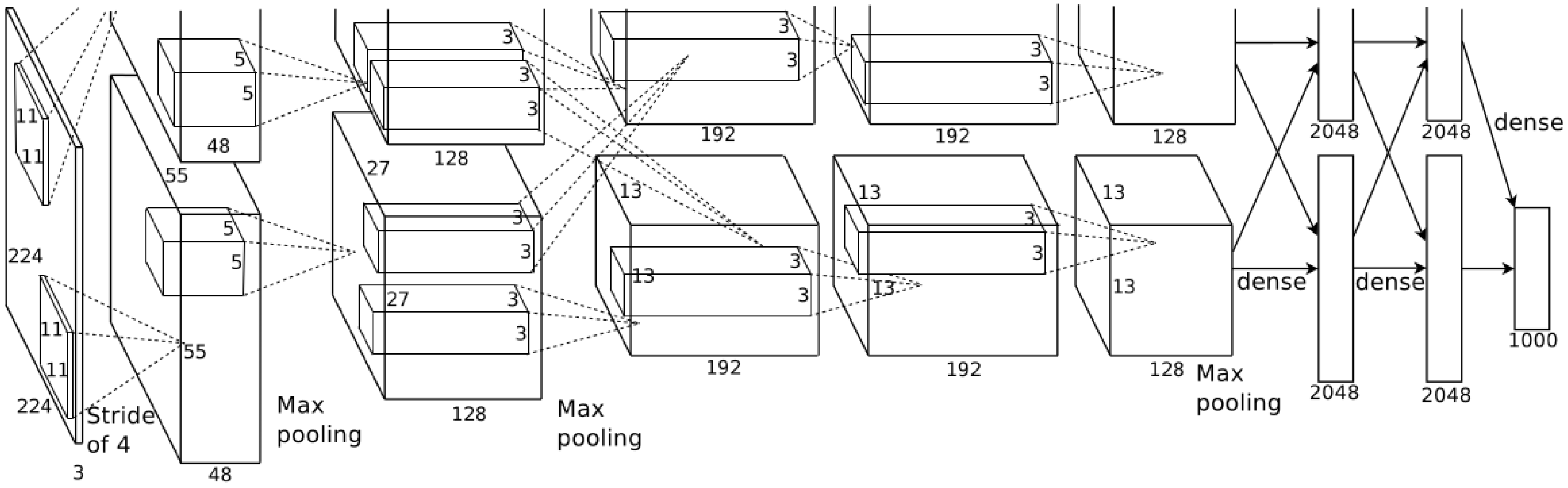
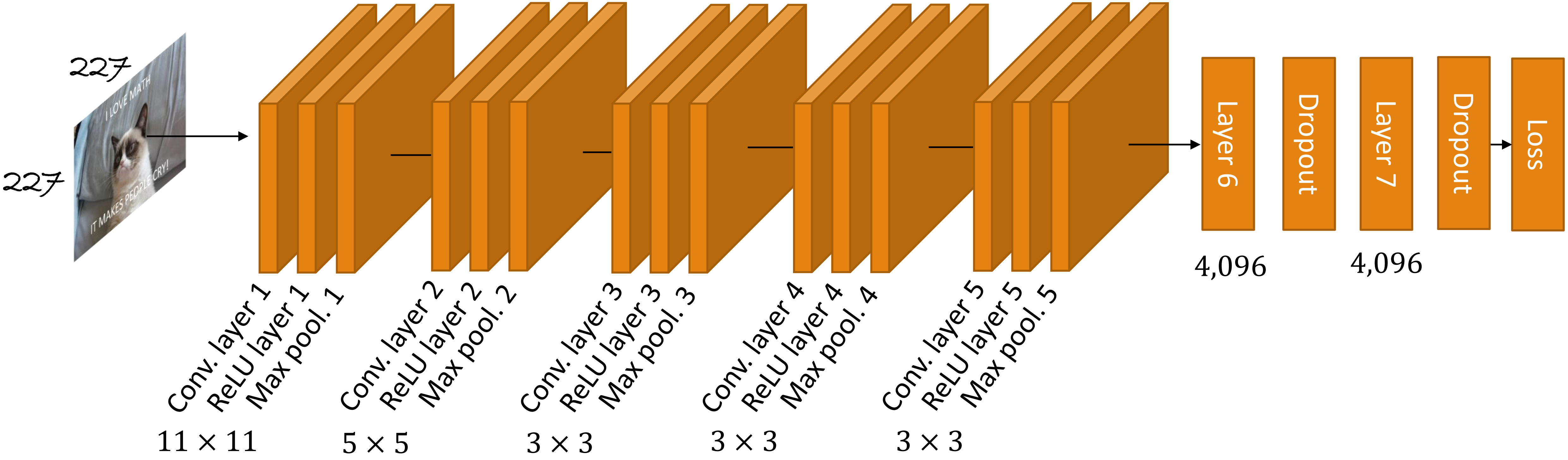


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input 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.

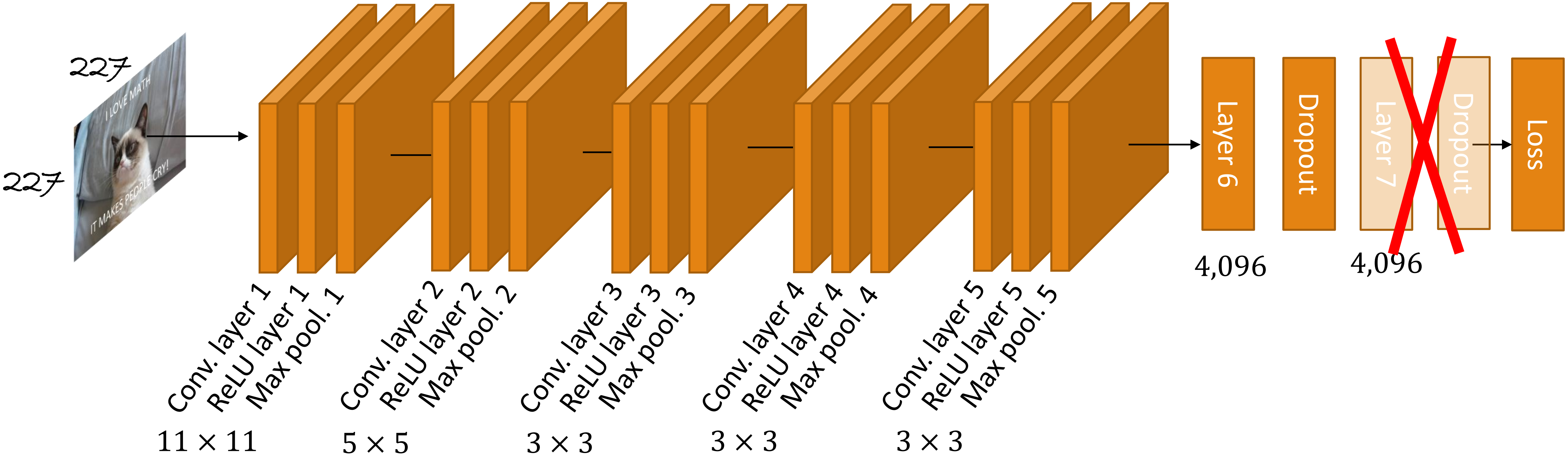
Architectural details

18.2% error in Imagenet



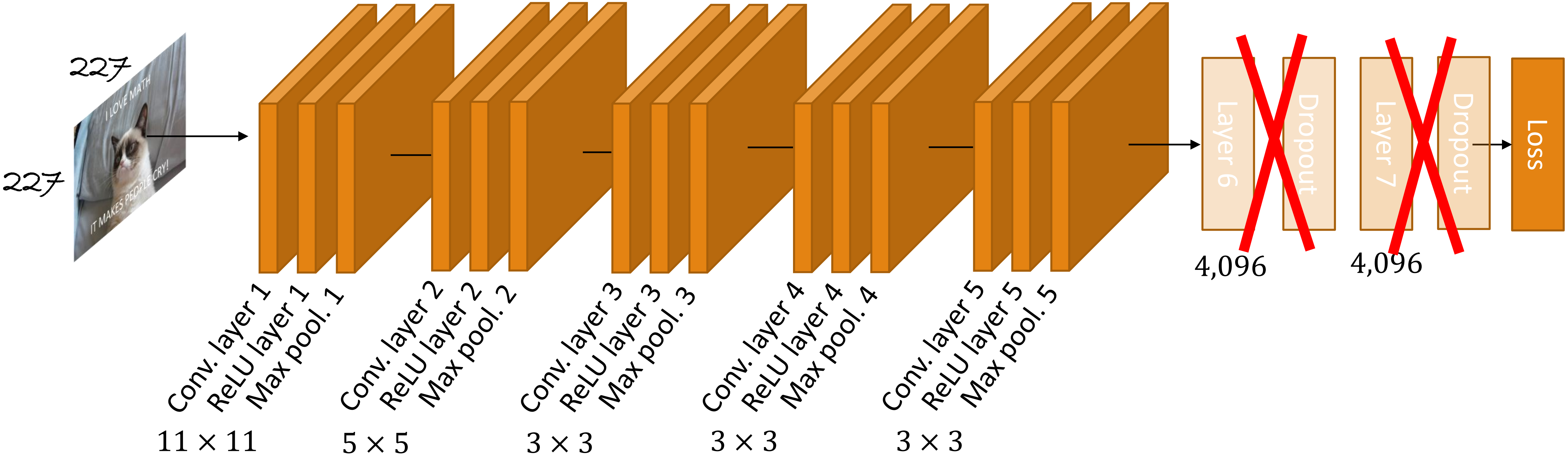
Removing layer 7

1.1% drop in performance, 16 million less parameters



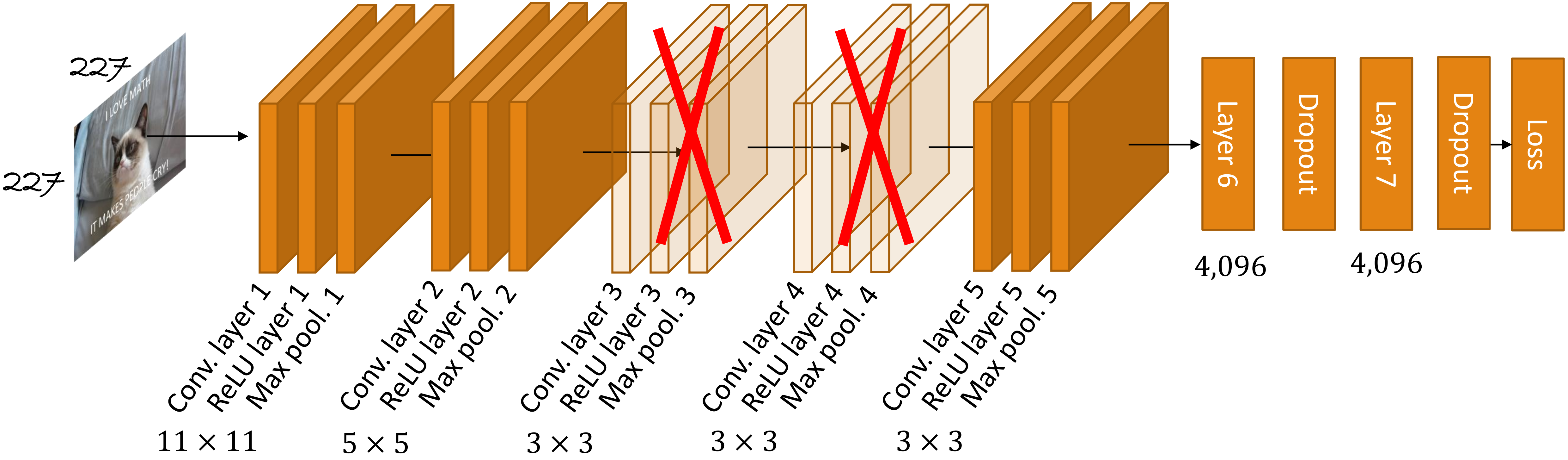
Removing layer 6, 7

5.7% drop in performance, 50 million less parameters



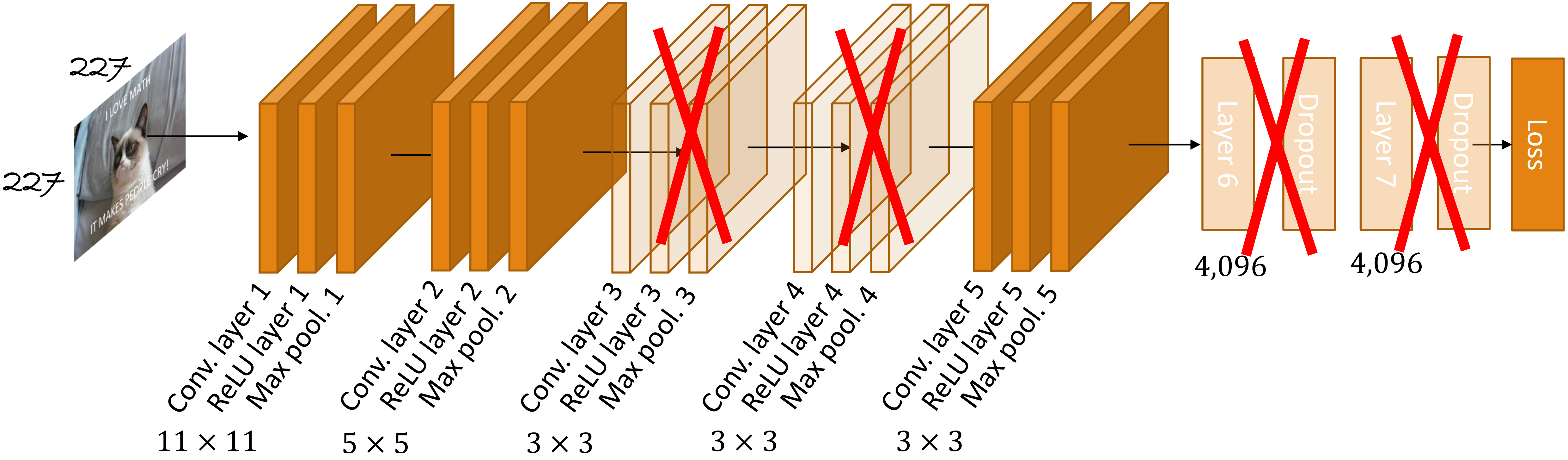
Removing layer 3, 4

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

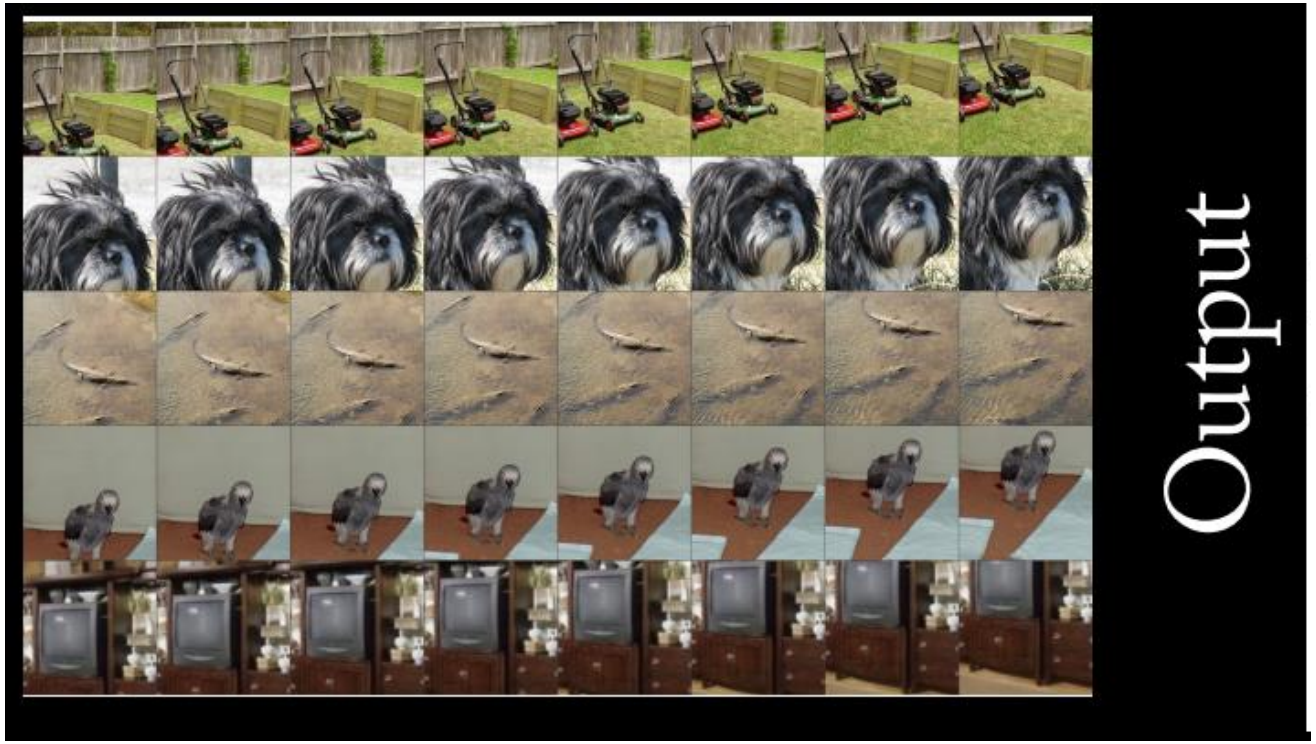


Removing layer 3, 4, 6, 7

33.5% drop in performance. Conclusion? Depth!



Translation invariance



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

Prepare to vote

Internet 1

The text on this slide will instruct your audience on how to vote. This text will only appear once you start a free or a credit session.

Please note that the text and appearance of this slide (font, size, color, etc.) cannot be changed.

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

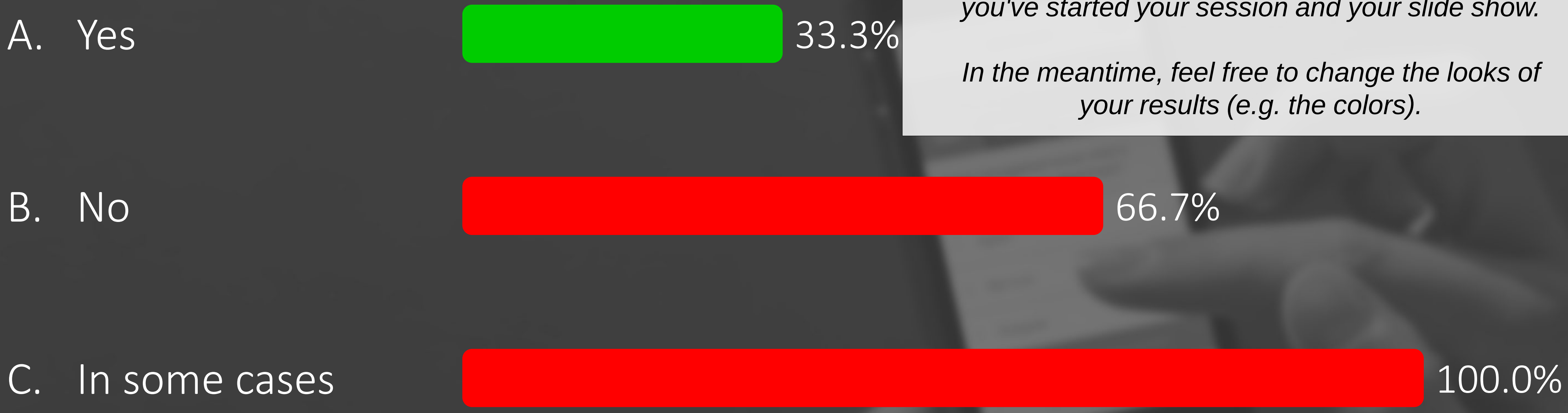
shakespeak

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

TXT The applicable explanations will be inserted after you have started a session.

Votes: 0

Is Alexnet translation invariant?



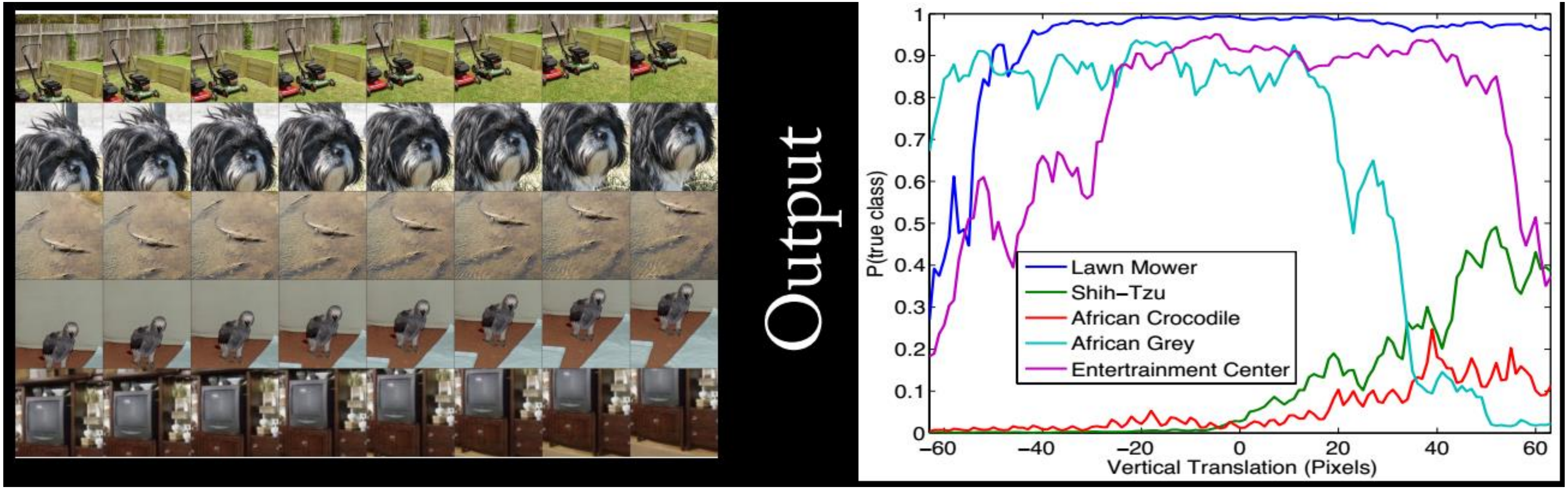
We will set these example results to zero once you've started your session and your slide show.

In the meantime, feel free to change the looks of your results (e.g. the colors).

Closed

shakespeak

Translation invariance



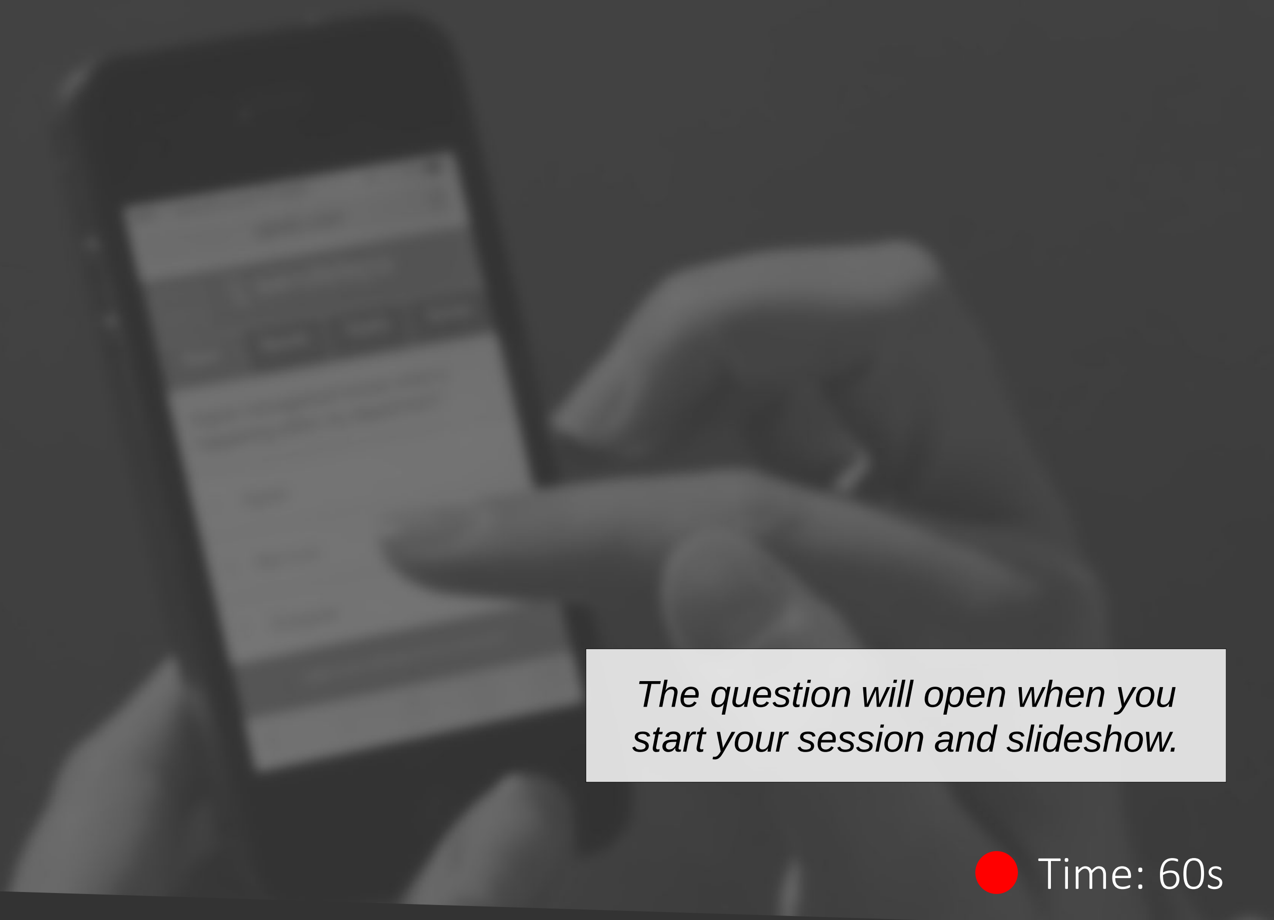
Credit: R. Fergus slides in Deep Learning Summer School 2016

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

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

TXT The applicable explanations will be inserted after you have started a session.

Votes: 0

Is Alexnet scale invariant?

We will set these example results to zero once you've started your session and your slide show.

In the meantime, feel free to change the looks of your results (e.g. the colors).

A. Yes 33.3%

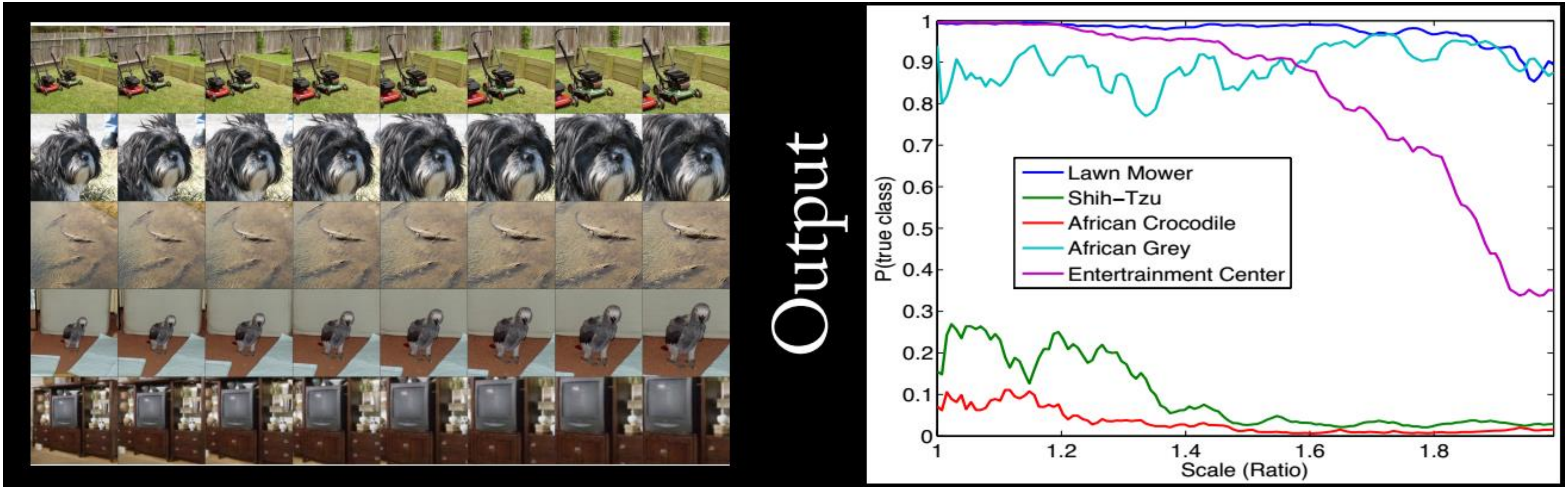
B. No 66.7%

C. In some cases 100.0%

Closed

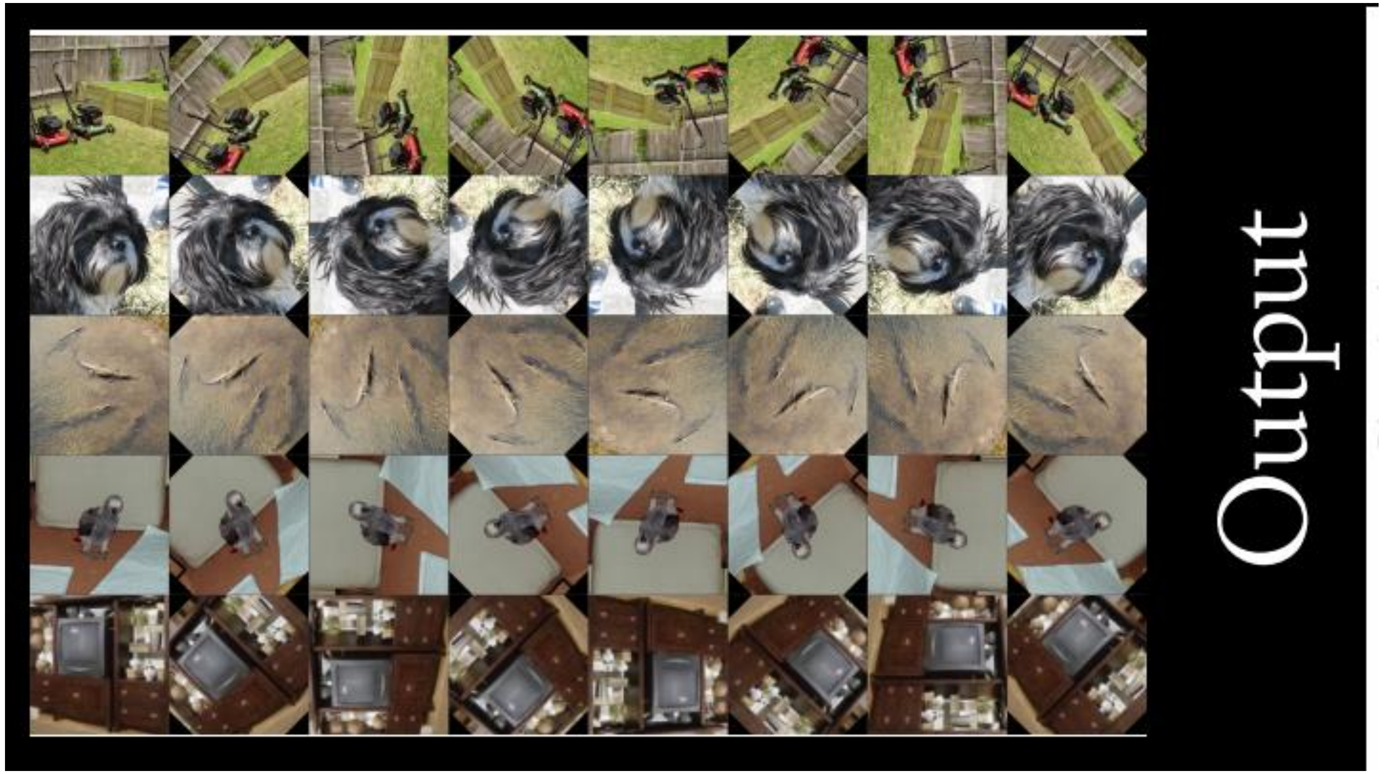
shakespeak

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

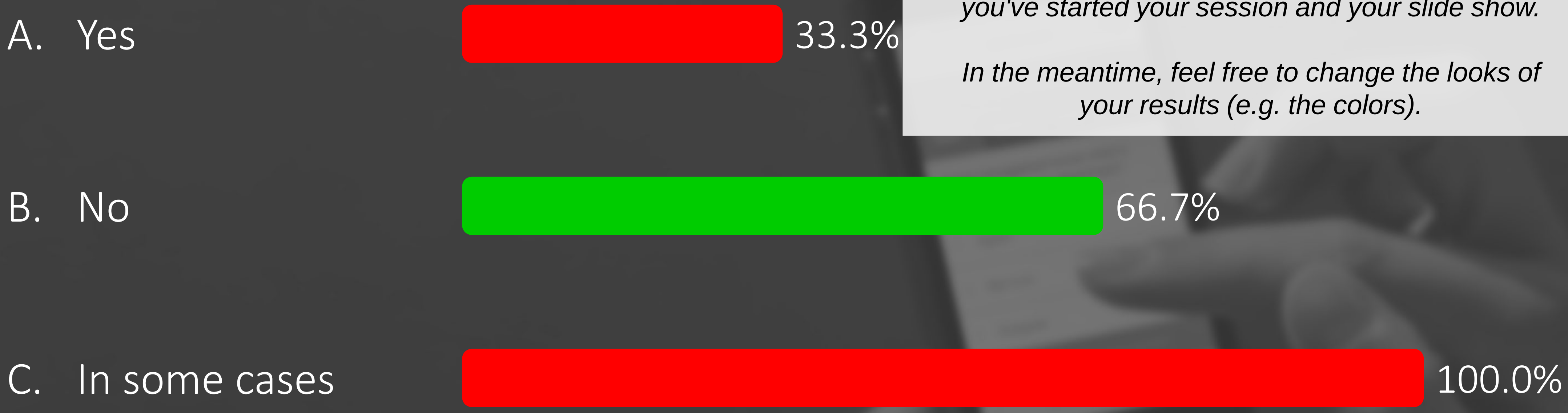
shakespeak

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

TXT The applicable explanations will be inserted after you have started a session.

Votes: 0

Is Alexnet rotation invariant?



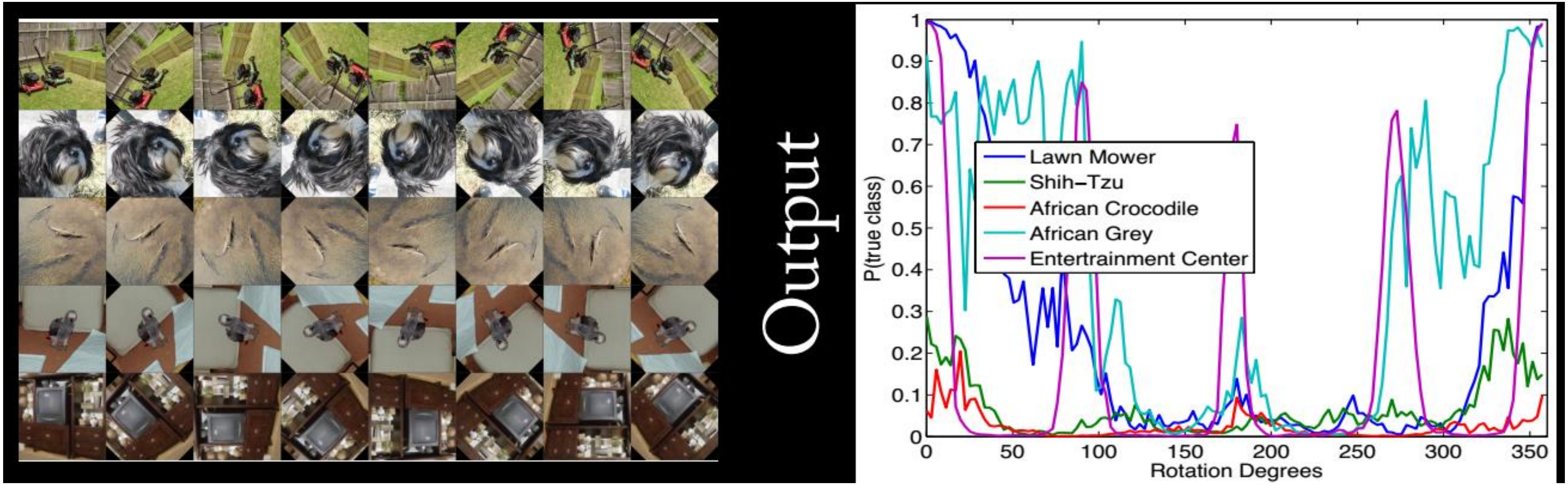
We will set these example results to zero once you've started your session and your slide show.

In the meantime, feel free to change the looks of your results (e.g. the colors).

shakespeak

Closed

Rotation invariance

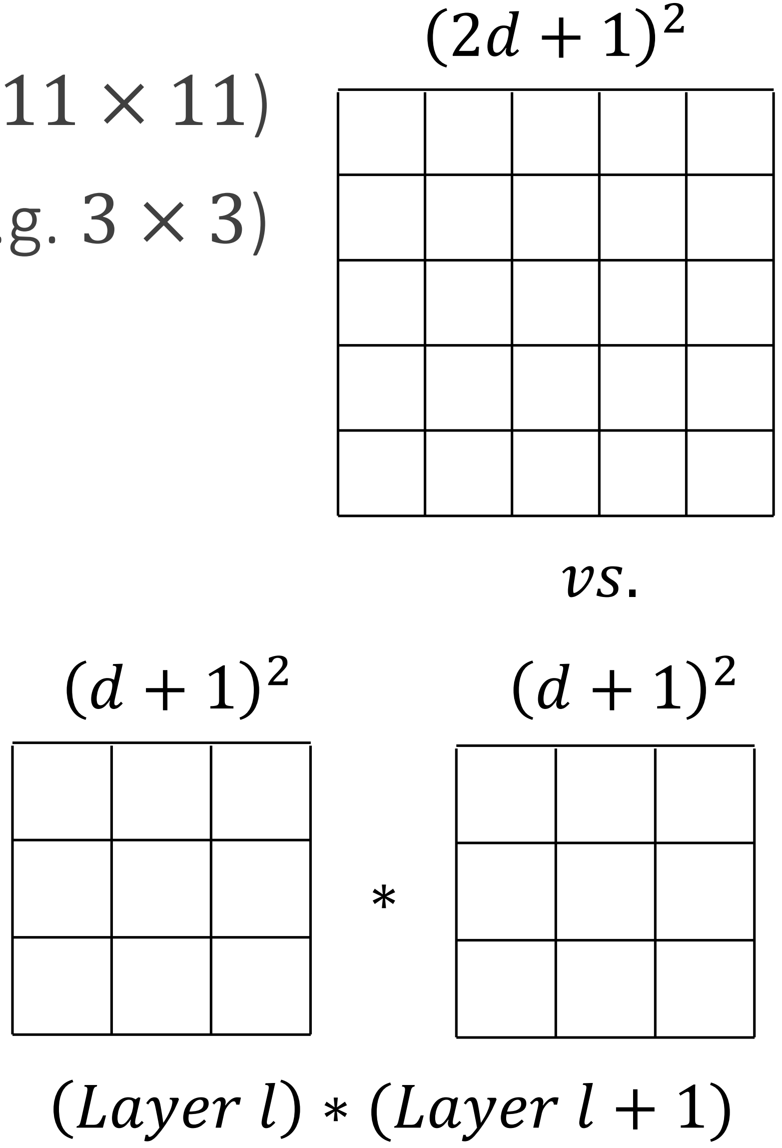


Credit: R. Fergus slides in Deep Learning Summer School 2016

Understanding convnets

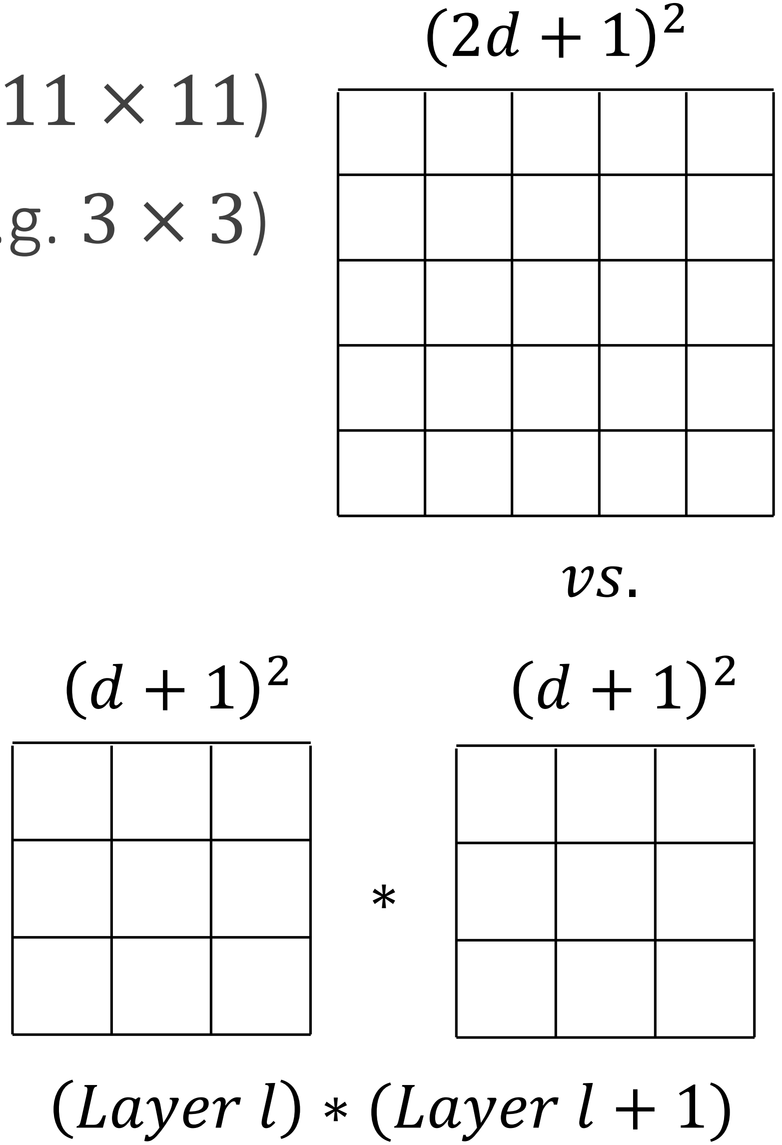
How large filters?

- Traditionally, medium sized filters (smaller than 11×11)
- Modern architectures prefer small filter sizes (e.g. 3×3)



How large filters?

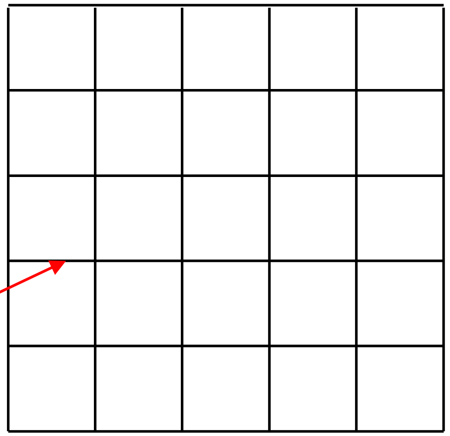
- Traditionally, medium sized filters (smaller than 11×11)
- Modern architectures prefer small filter sizes (e.g. 3×3)
- We lose frequency resolution
- Fewer parameters to train



How large filters?

- Traditionally, medium sized filters (smaller than 11×11)
- Modern architectures prefer small filter sizes (e.g. 3×3)
- We lose frequency resolution
- Fewer parameters to train
- Deeper networks of cascade filters
  - Still, the same output dimensionalities

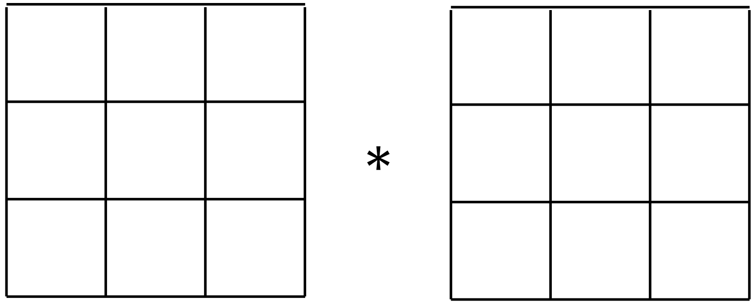
$(2d+1)^2$



For stride 1 the first feature map has dimensionality $\frac{H-2d-1}{1} + 1 = H - 2d$

$vs.$

$(d+1)^2$ $(d+1)^2$

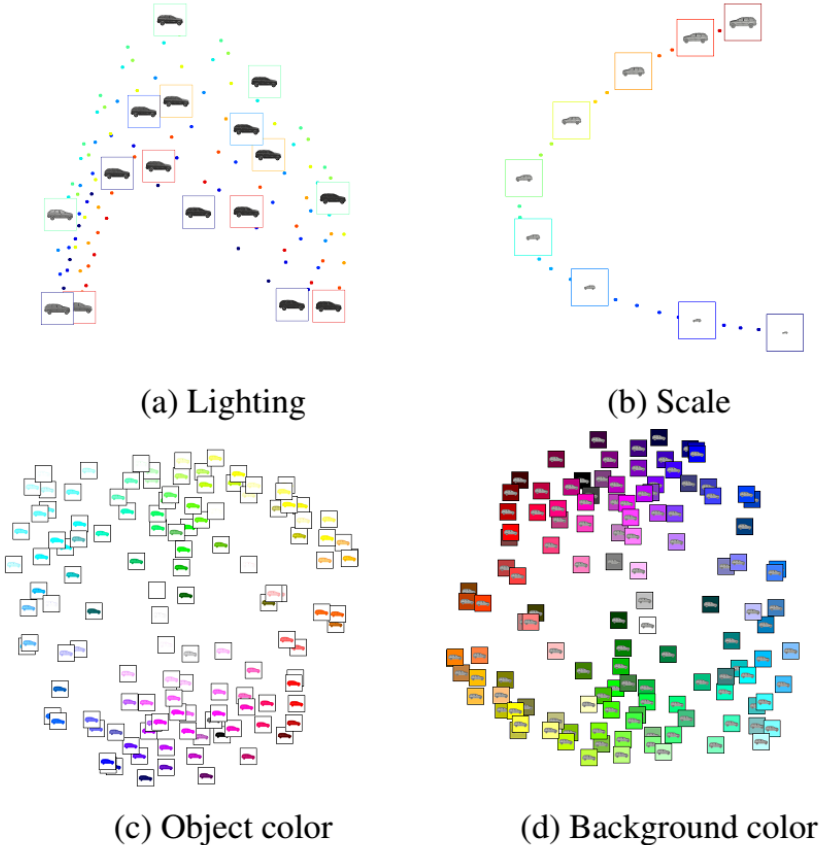


$(Layer\ l) * (Layer\ l+1)$

For stride 1 the first feature map has dimensionality $H - d$, the second image $\frac{H-d-d-1}{1} + 1 = H - 2d$

Filter invariance and equivariance

- Filters learn how different variances affect appearance
- Different layers and different hierarchies focus on different transformations
- For different objects filters reproduce different behaviors



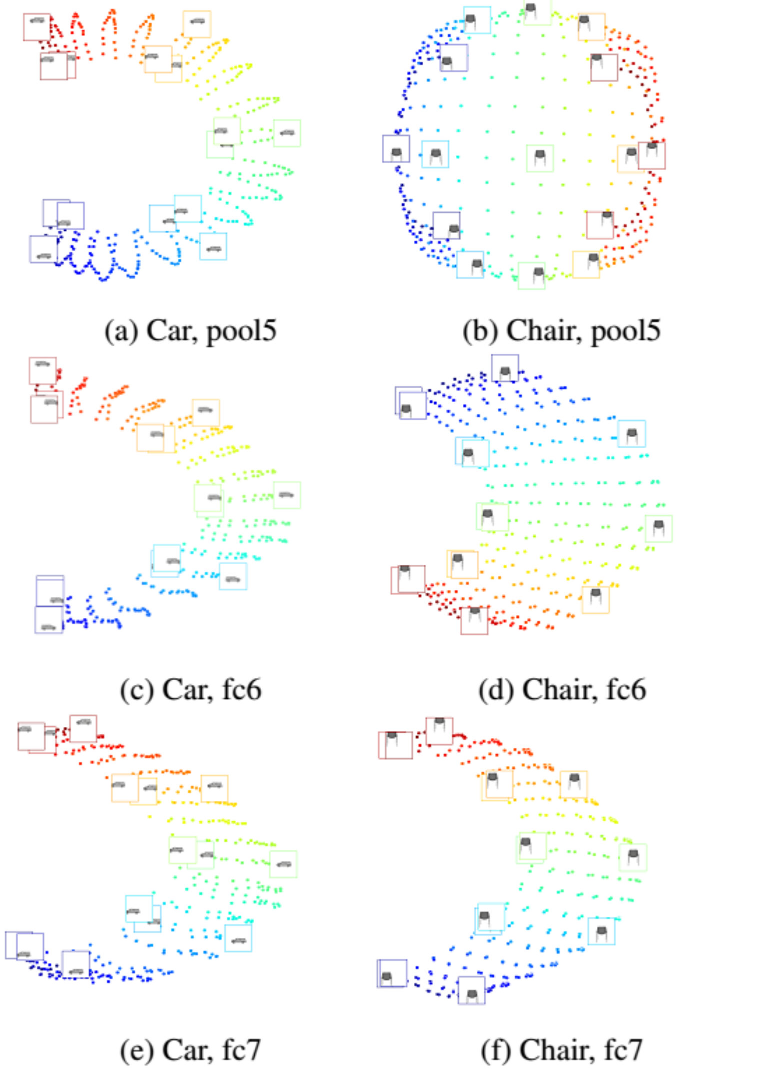


Figure 3: PCA embeddings for 2D position on AlexNet.

Aubry et al., Understanding deep features with computer-generated imagery , 2015]

Filter invariance and equivariance

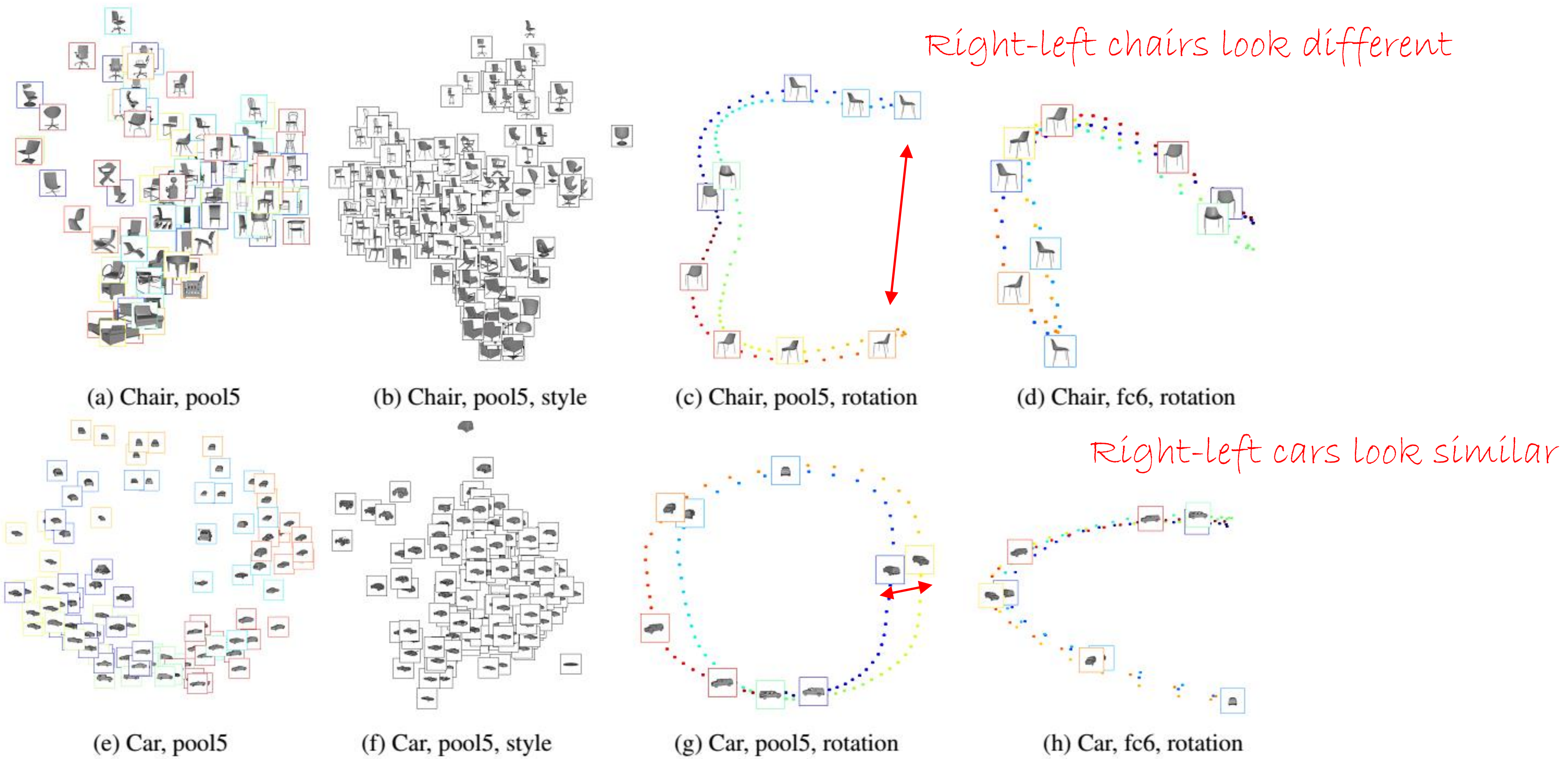


Figure 2: **Best viewed in the electronic version.** PCA embeddings (dims. 1,2) of AlexNet features for “chairs” (first row) and “cars” (second row). Column 1 – Direct embedding of the rendered images without viewpoint-style separation. Columns 2,3 – Embeddings associated with style (for all rotations) and rotation (for all styles). Column 4 – Rotation embedding for fc6, which is qualitatively different than pool5. Colors correspond to orientation and can be interpreted via the example images in columns 3,4. Similar results for other categories and PCA dimensions are available in the supplementary material.

Interesting questions

- What do the image activations in different layers look like?
- What types of images create the strongest activations?
- What are the activations for the class “ostrich”?
- Do the activations occur in meaningful positions?

Feature maps

- Convolution activations == feature maps
- A deep network has several hierarchical layers
  - hence several hierarchical feature maps going from less to more abstract

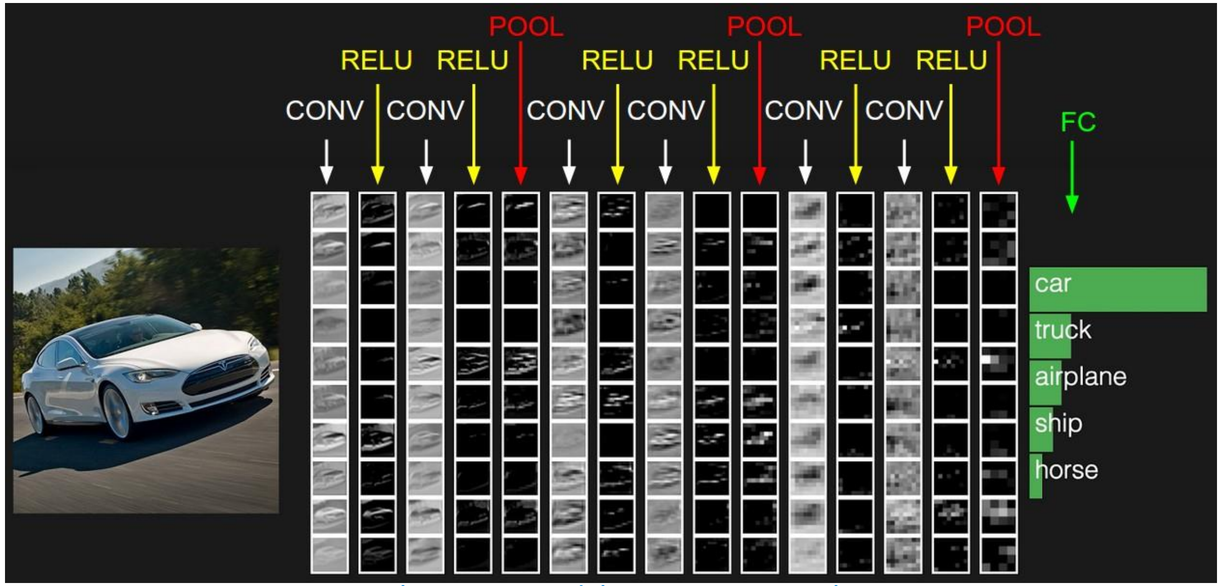
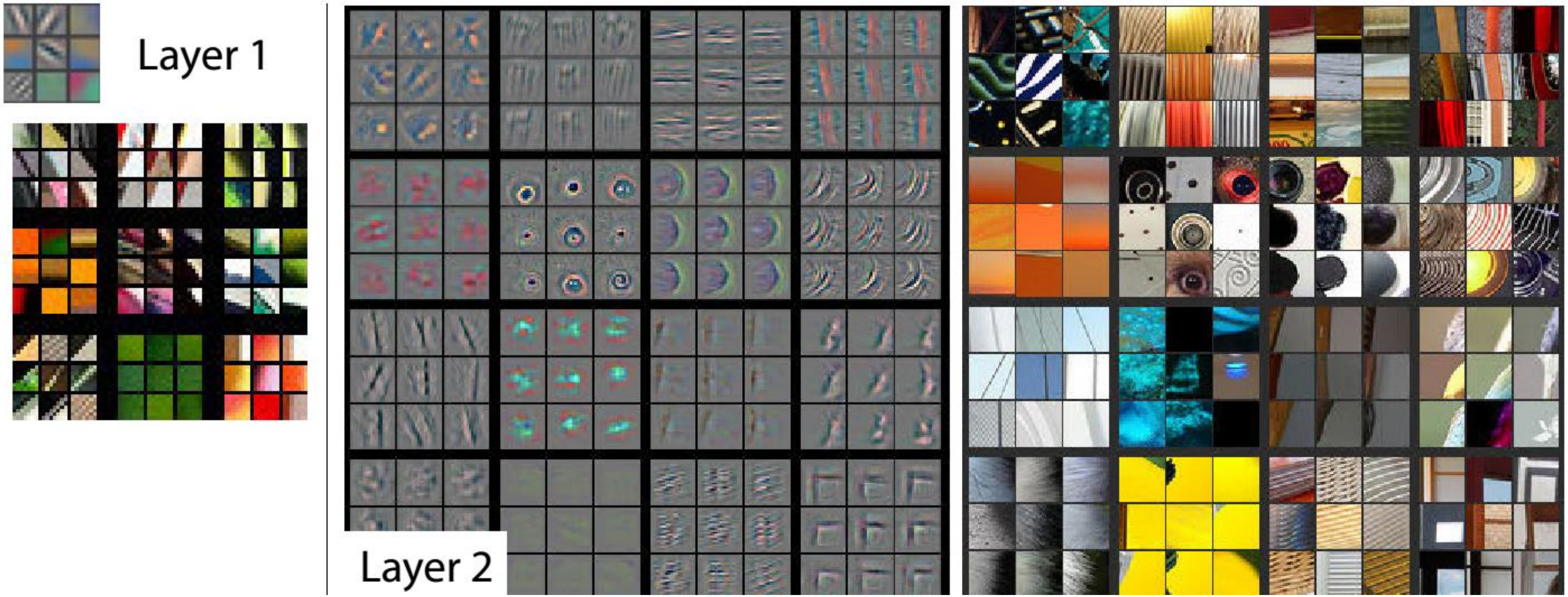


Image borrowed by A. Karpathy

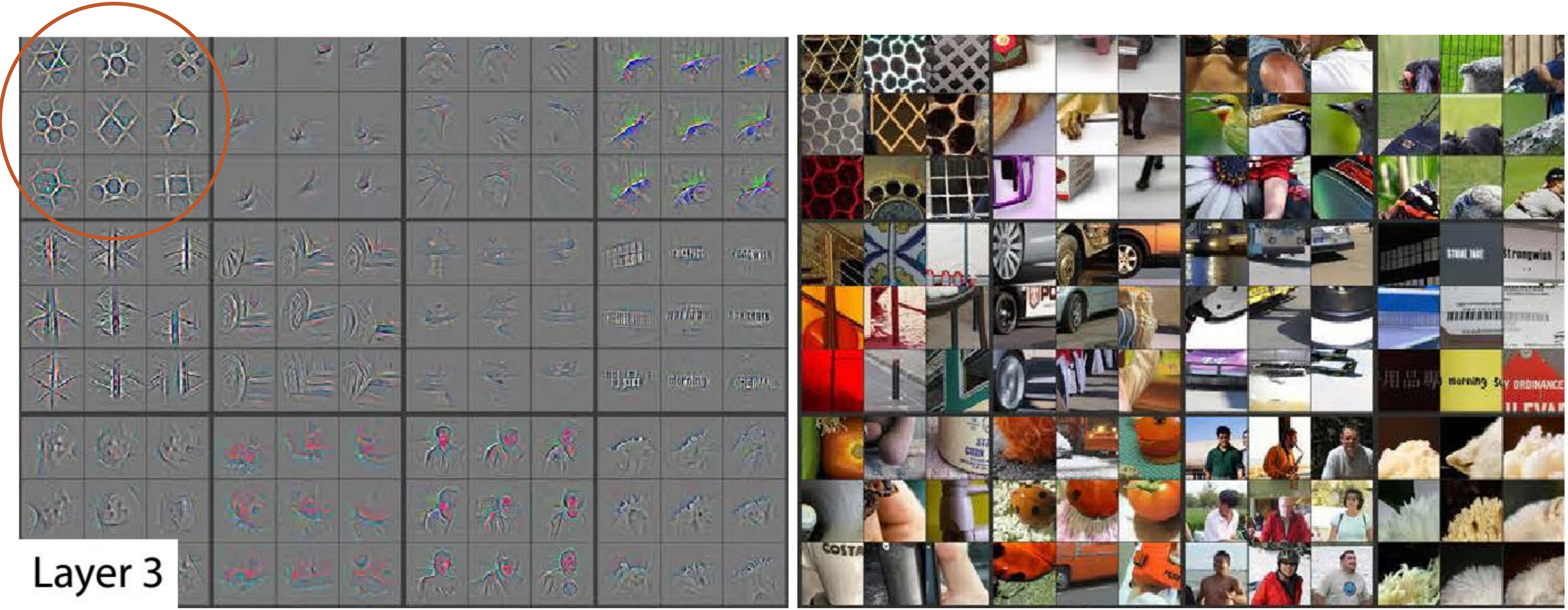
What excites feature maps?

- *"Given a random feature map what are the top 9 activations?"*



What excites feature maps?

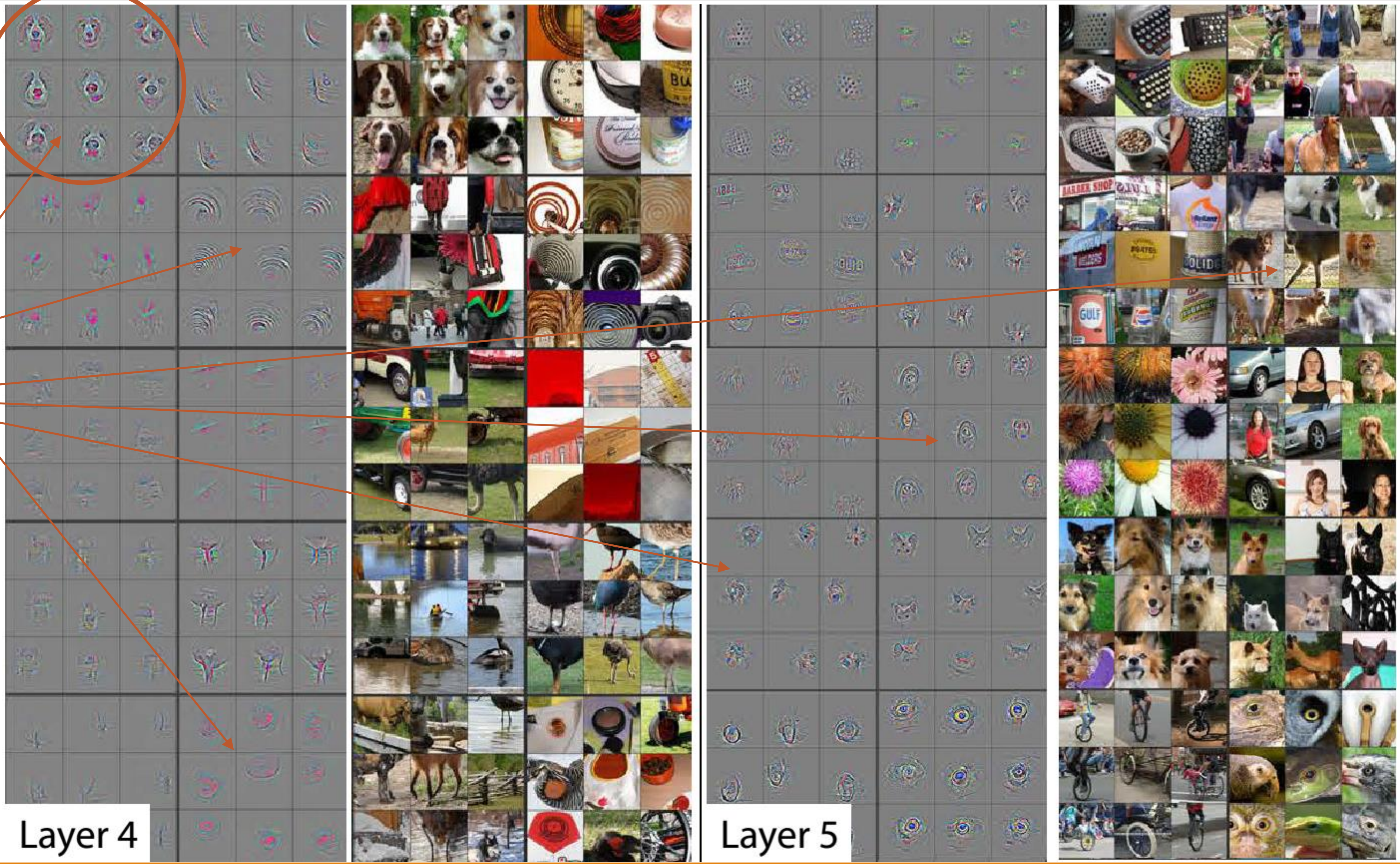
Similar activations from lower level visual patterns



What excites feature maps? [Zeiler2014]

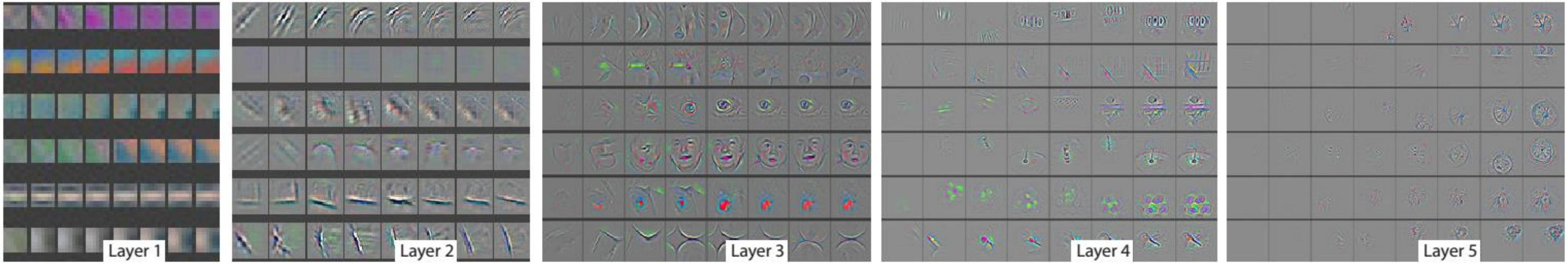
Similar activations from semantically similar pictures

Visual patterns become more and more intricate and specific (greater invariance)

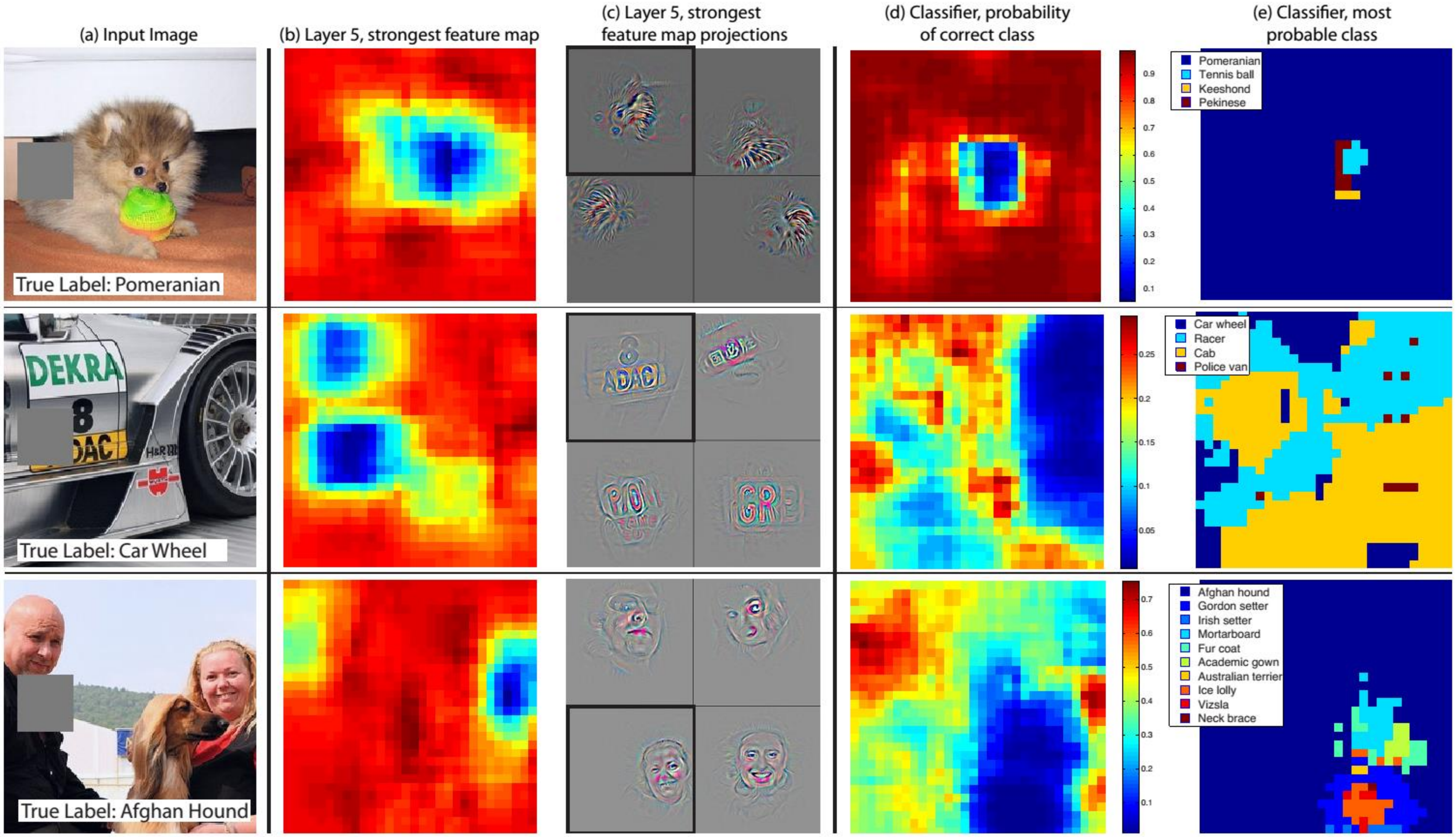


Feature evolution over training

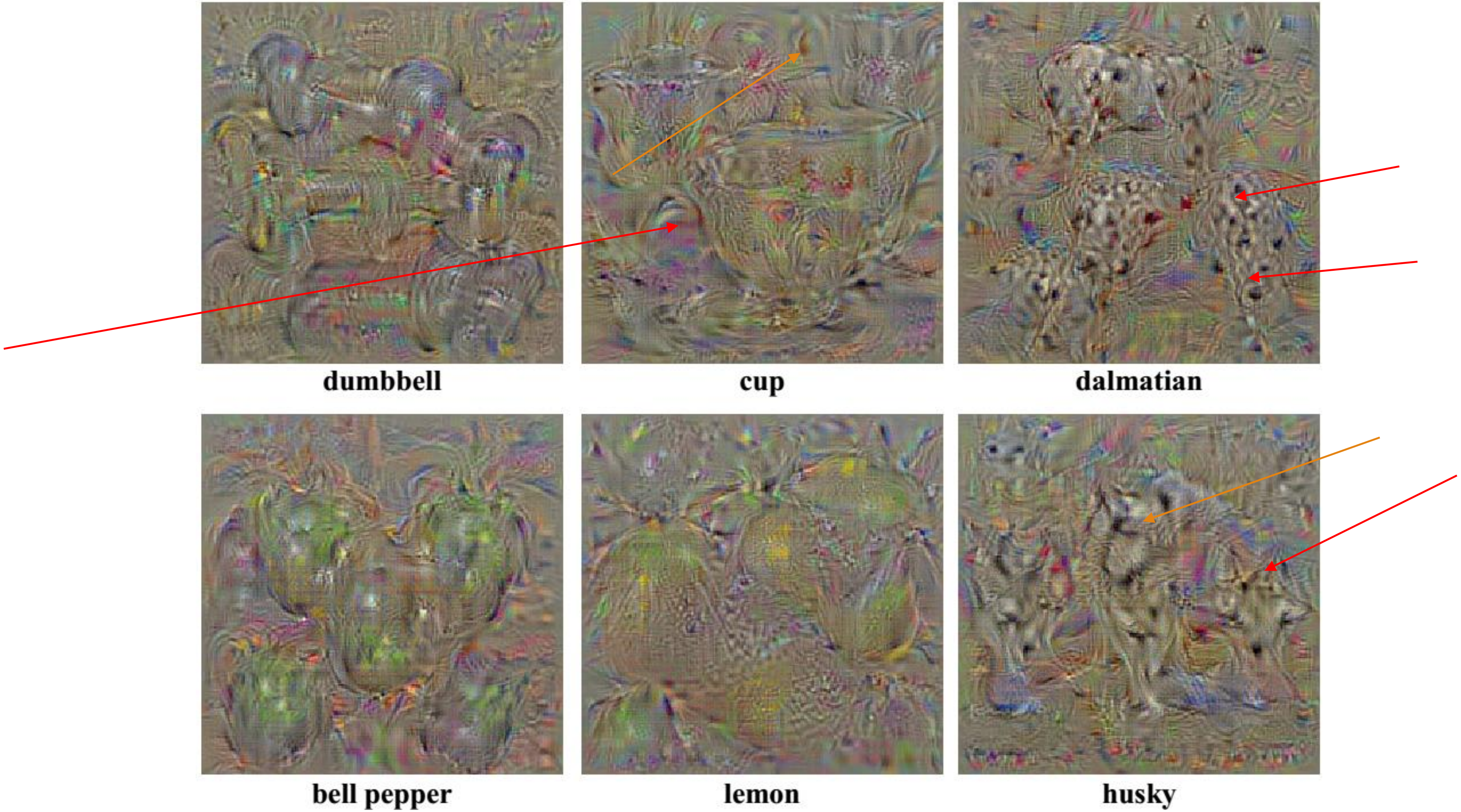
- Given a neuron (outputs a single feature map)
  - Strongest activation during training for epochs 1, 2, 5, 10, 20, 30, 40, 64



But does a Convnet really learn the object?



What is a “Convnet dog”, however? [Simonyan2014]



Transfer learning

- Assume two datasets, T and S
- Dataset S is
  - fully annotated, plenty of images
  - We can build a model h_S
- Dataset T is
  - Not as much annotated, or much fewer images
  - The annotations of T do not need to overlap with S
- We can use the model h_S to learn a better h_T
- This is called transfer learning

Imagenet: 1million

$\boldsymbol{h_A}$

My dataset: 1,000

$\boldsymbol{h_B}$

Why use Transfer Learning?

- A CNN can have millions of parameters
- But our datasets are not always as large
- Could we still train a CNN without overfitting problems?

Convnets are good in transfer learning

- Even if our dataset T is not large, we can train a CNN for it
- Pre-train a network on the dataset S
- Then, there are two solutions
  - Fine-tuning
  - CNN as feature extractor

Solution I: Fine-tune h_T using h_S as initialization

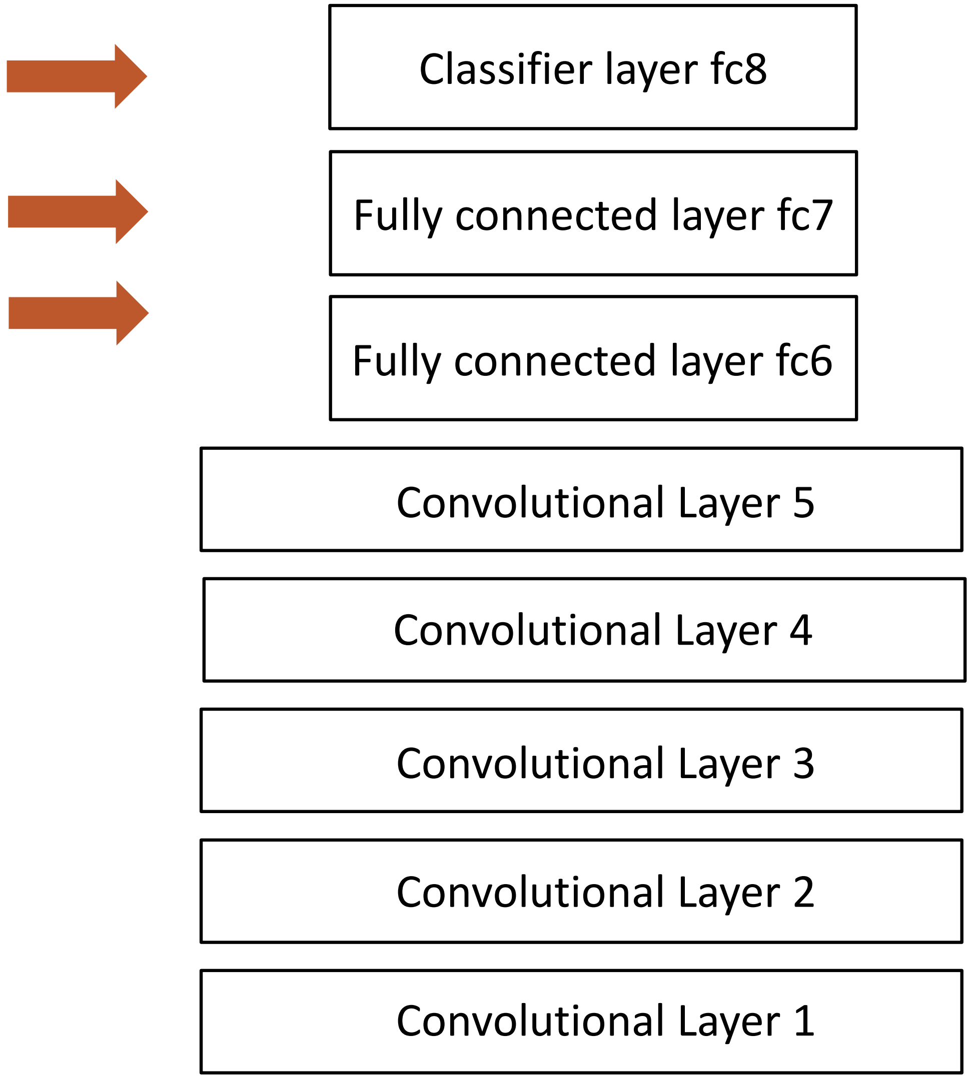
- Assume the parameters of S are already a good start near our final local optimum
- Use them as the initial parameters for our new CNN for the target dataset

$$w^{l}_{\mathrm{T},\, t=0} = w^{l}_{\mathrm{S},\, t=\infty} \text{ for layers } l = 1, 2, \dots$$

- Use when the target dataset T is relatively big
  - E.g. for Imagenet S with approximately 1 million images a dataset T with more than a few thousand images should be ok
- What layers to initialize and how?

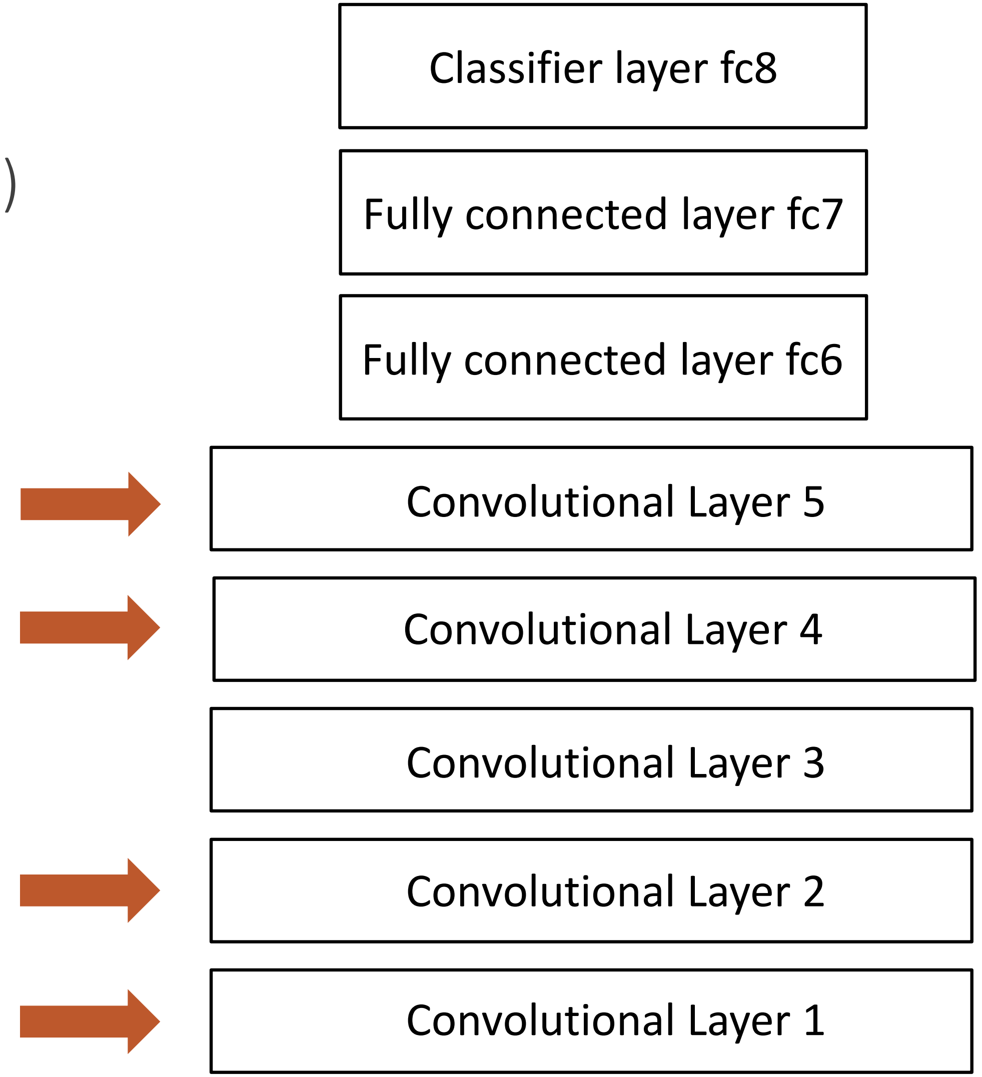
Initializing h_T with h_S

- Classifier layer to loss
  - The loss layer essentially is the “classifier”
  - Same labels → keep the weights from h_S
  - Different labels → delete the layer and start over
  - When too few data, fine-tune only this layer
- Fully connected layers
  - Very important for fine-tuning
  - Sometimes you need to completely delete the last before the classification layer if datasets are very different
  - Capture more semantic, “specific” information
  - Always try first when fine-tuning
  - If you have more data, fine-tune also these layers



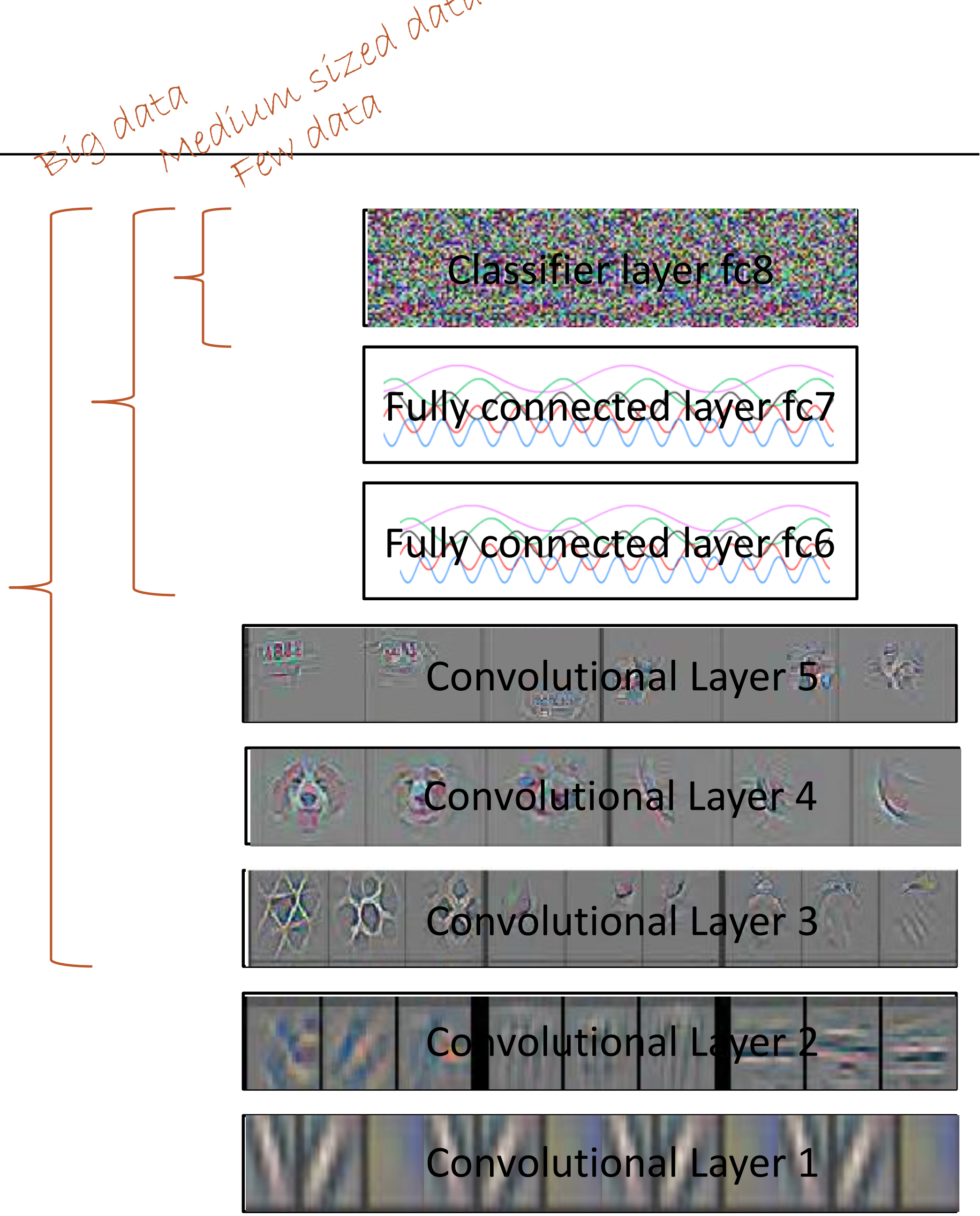
Initializing h_T with h_S

- Upper convolutional layers (conv4, conv5)
  - Mid-level spatial features (face, wheel detectors …)
  - Can be different from dataset to dataset
  - Capture more generic information
  - Fine-tuning pays off
  - Fine-tune if dataset is big enough
- Lower convolutional layers (conv1, conv2)
  - They capture low level information
  - This information does not change usually
  - Probably, no need to fine-tune but no harm trying



How to fine-tune?

- For layers initialized from h_S use a mild learning rate
  - Remember: your network is already close to a near optimum
  - If too aggressive, learning might diverge
  - A learning rate of 0.001 is a good starting choice (assuming 0.01 was the original learning rate)
- For completely new layers (e.g. loss) use aggressive learning rate
  - If too small, the training will converge very slowly
  - Remember: the rest of the network is near a solution, this layer is very far from one
  - A learning rate of 0.01 is a good starting choice
- If datasets are very similar, fine-tune only fully connected layers
- If datasets are different and you have enough data, fine-tune all layers



Solution II: Use h_s as a feature extractor for h_T

- Essentially similar to a case of solution I
  - but train only the loss layer
- Essentially use the network as a pretrained feature extractor
- Use when the target dataset T is small
  - Any fine-tuning of layer might cause overfitting
  - Or when we don't have the resources to train a deep net
  - Or when we don't care for the best possible accuracy

Which layer?

Table 6. Analysis of the discriminative information contained in each layer of feature maps within our ImageNet-pretrained convnet. We train either a linear SVM or softmax on features from different layers (as indicated in brackets) from the convnet. Higher layers generally produce more discriminative features.

	Cal-101 (30/class)	Cal-256 (60/class)
SVM (1)	44.8 ± 0.7	24.6 ± 0.4
SVM (2)	66.2 ± 0.5	39.6 ± 0.3
SVM (3)	72.3 ± 0.4	46.0 ± 0.3
SVM (4)	76.6 ± 0.4	51.3 ± 0.1
SVM (5)	$\mathbf{86.2 \pm 0.8}$	65.6 ± 0.3
SVM (7)	$\mathbf{85.5 \pm 0.4}$	$\mathbf{71.7 \pm 0.2}$
Softmax (5)	82.9 ± 0.4	65.7 ± 0.5
Softmax (7)	$\mathbf{85.4 \pm 0.4}$	$\mathbf{72.6 \pm 0.1}$

Lower layer features capture more basic information (texture, etc). Good for image-to-image comparisons, image retrieval

Higher layer features are capture more semantic information. Good for higher level classification

Visualizing and Understanding Convolutional Networks, Zeiler and Fergus, ECCV 2014

Summary

- Shared filters through local connectivity
- Convolutions
- Convolutional Neural Networks
- Alexnet case study
- Visualizing ConvNets
- Transfer learning

Reading material

- Chapter 9