10-315
Introduction to ML

Deep Learning for Computer Vision

Instructor: Pat Virtue

Deep Learning for Computer Vision

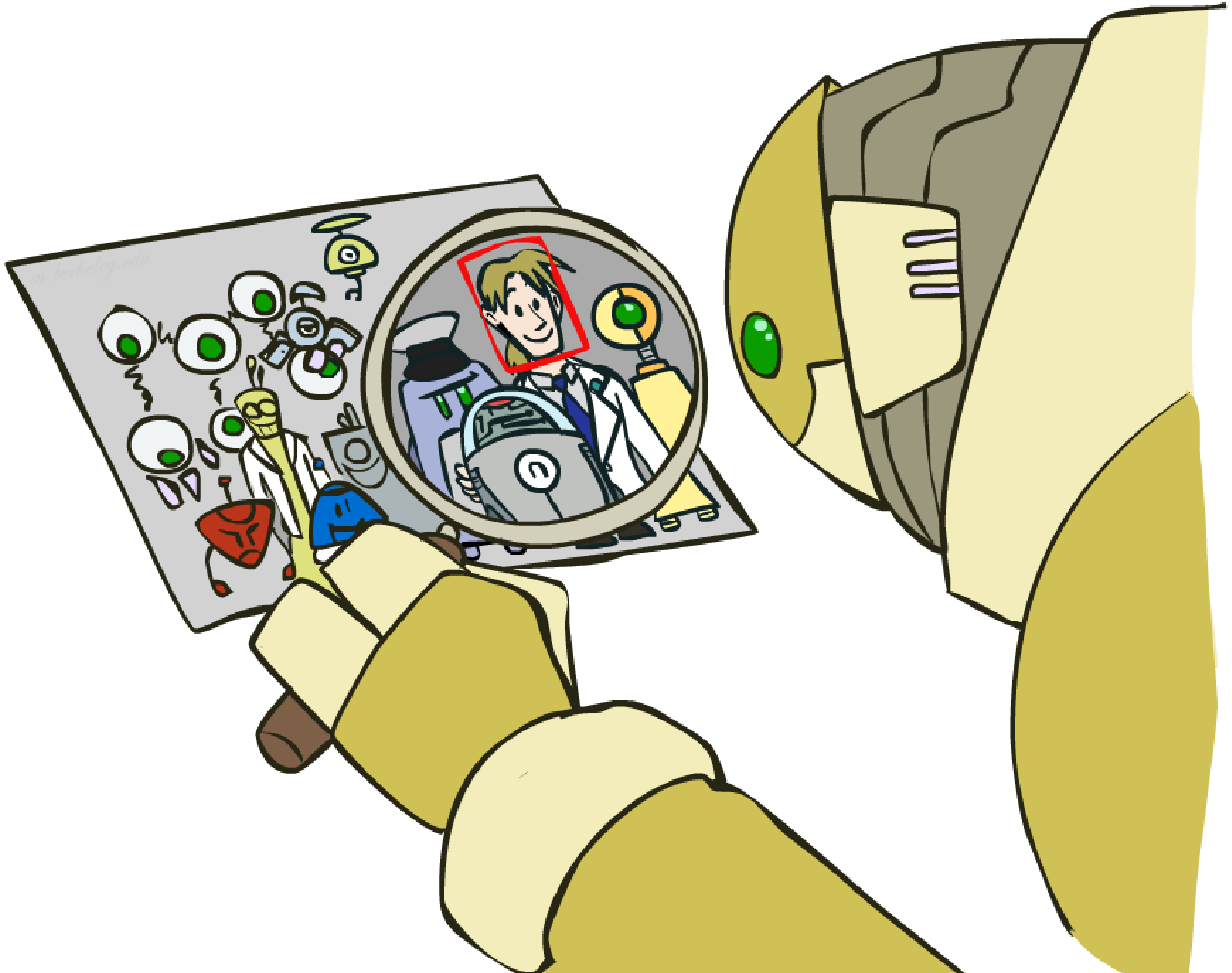
1. Computer vision tasks
2. Image features
3. Convolution Channels & CNNs
4. Computer vision history: Deep learning boom
5. CNN architectures & Practical deep learning

Computer Vision

Image in → information out

Computer Vision Tasks

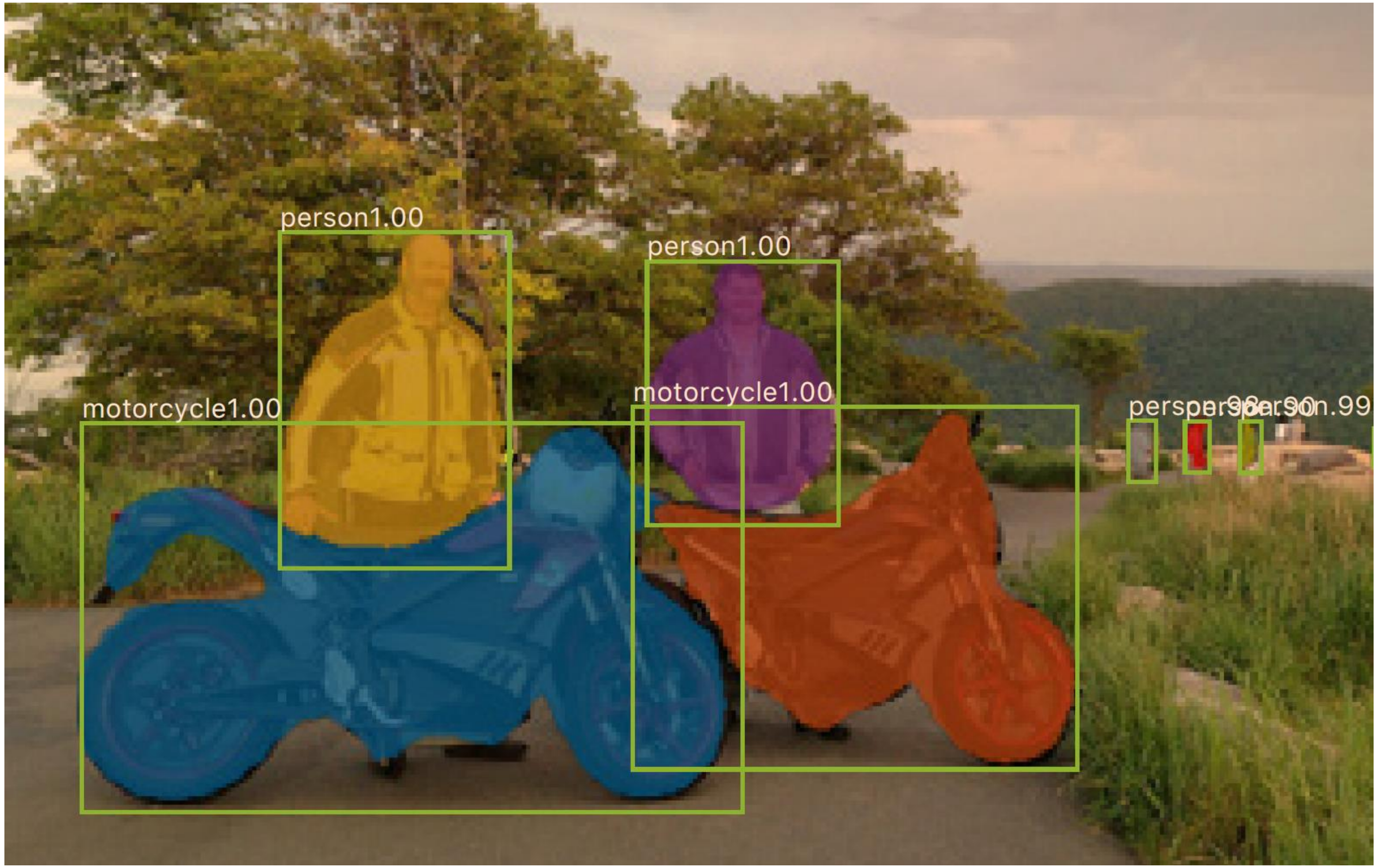
Computer Vision Tasks: How far along are we?



Computer Vision Tasks: How far along are we?

Terminator 2, 1991 https://www.youtube.com/watch?v=9MeaaCwBW28

Computer Vision Tasks: How far along are we?



2017:
0.2 seconds
per image

Mask R-CNN

He, Kaiming, et al. "Mask R-CNN." *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2017.

Computer Vision Tasks: How far along are we?

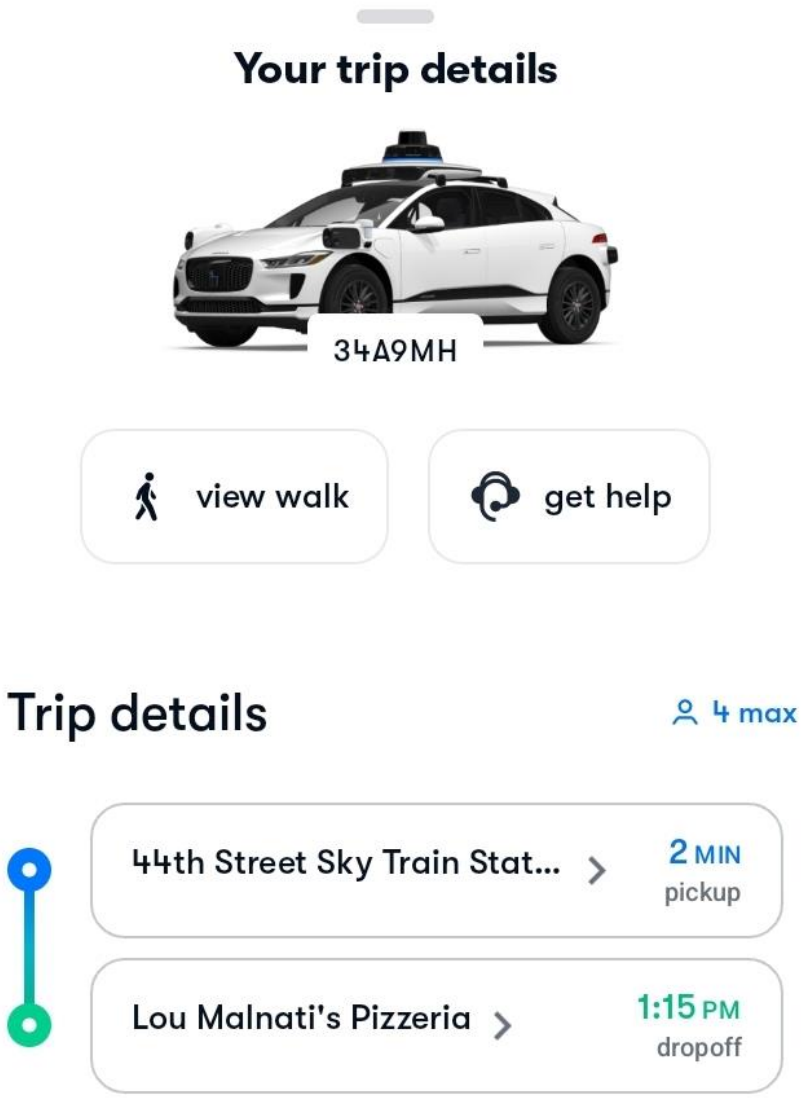
"My CPU is a neural net processor, a learning computer"

Terminator 2, 1991

Computer Vision Tasks: Autonomous Driving

Tesla, Inc: https://vimeo.com/192179726

Computer Vision Tasks: Autonomous Driving



Computer Vision: Autonomous Driving

Image-related Tasks: Input/Output

Input	Task	Output
Image	Image classification	Category
Image	Image detection	Bounding box
Image	Image segmentation	Image: category per pixel
Image	Image processing*	Image

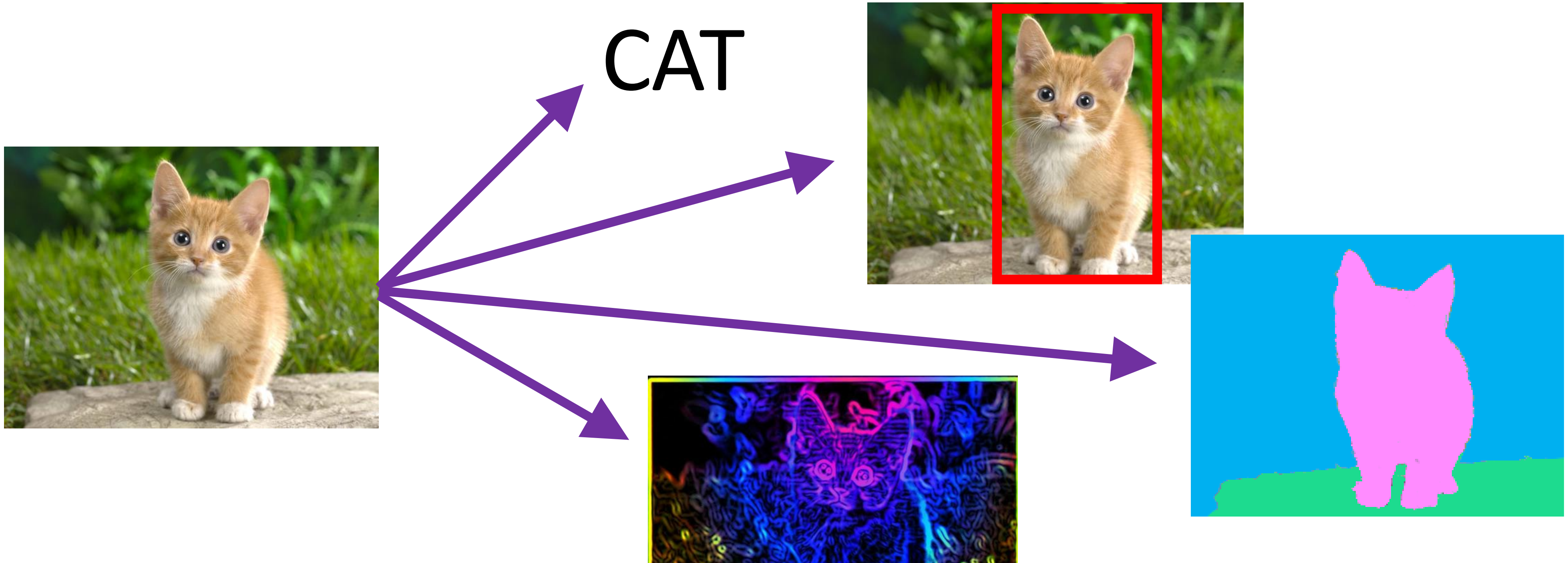


Image-related Tasks: Domain Transfer

CycleGAN

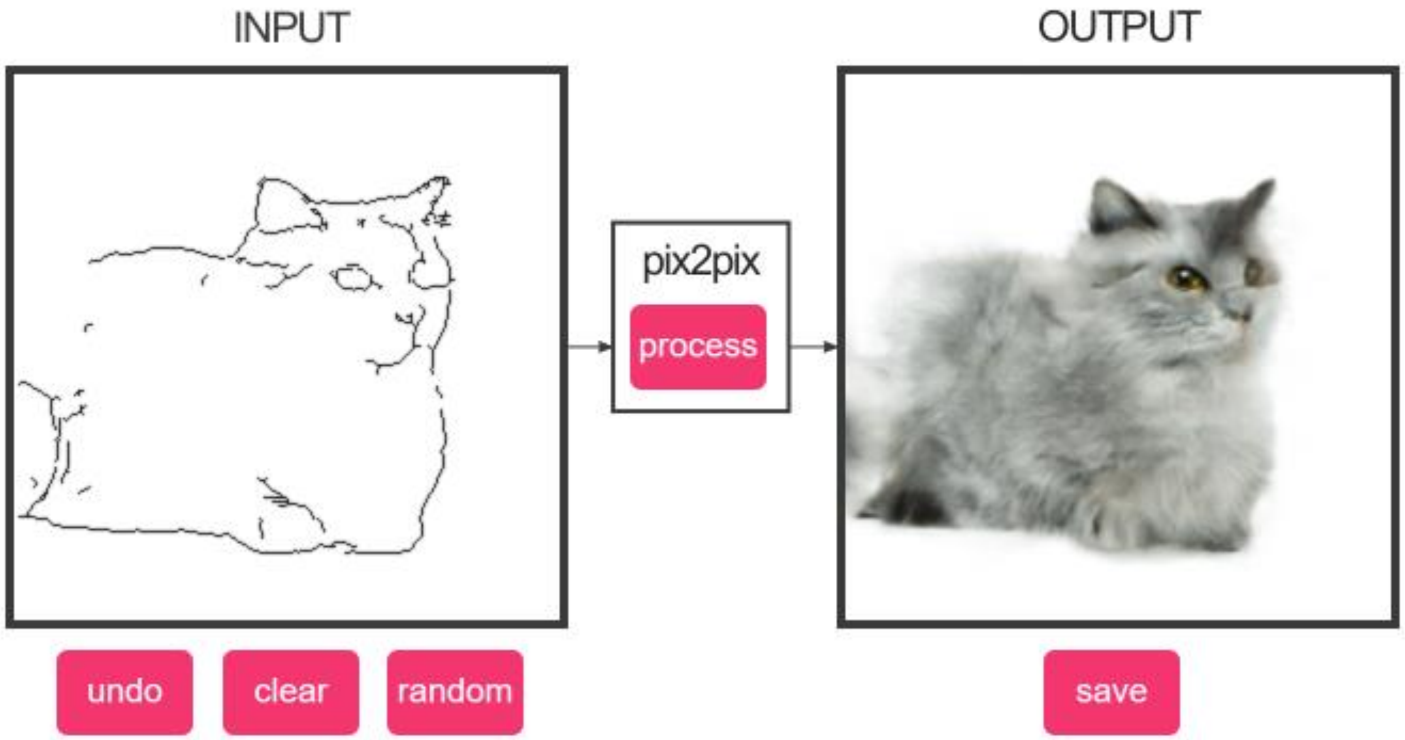
Jun-Yan Zhu*, Taesung Park*, Phillip Isola, and Alexei A. Efros

Jun-Yan Zhu*, Taesung Park*, Phillip Isola, and Alexei A. Efros. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks", ICCV 2017.

Image-related Tasks : Domain Transfer

Pix2pix

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros



https://affinelayer.com/pixsrv/ and https://phillipi.github.io/pix2pix/

Text to Image

GigaGAN

Changing texture with prompting. At coarse layers, we use the prompt "A teddy bear on tabletop" to fix the layout. Then at fine layers, we use "A teddy bear with the texture of [fleece, crochet, denim, fur] on tabletop". (Youtube link)

Changing style with prompting. At coarse layers, we use the prompt "A mansion" to fix the layout. Then at fine layers, we use "A [modern, Victorian] mansion in [sunny day, dramatic sunset]". (Youtube link)

https://mingukkang.github.io/GigaGAN/

Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, Taesung Park. "Scaling up GANs for Text-to-Image Synthesis", CVPR 2023.

Image-related Tasks: Input/Output

Input	Task	Output
Image	Image classification	Category
Image	Image detection	Bounding box
Image	Image segmentation	Image: category per pixel
Image	Image processing*	Image
Image (portion missing)	In-painting	Image
Random values	Image generation	Image
Text	Image generation	Image
Image	Image captioning	Text

Image Features

Image Features

Raw Data

Image Features

Raw Data

Image Features

Raw Data

Image Features

Raw Data

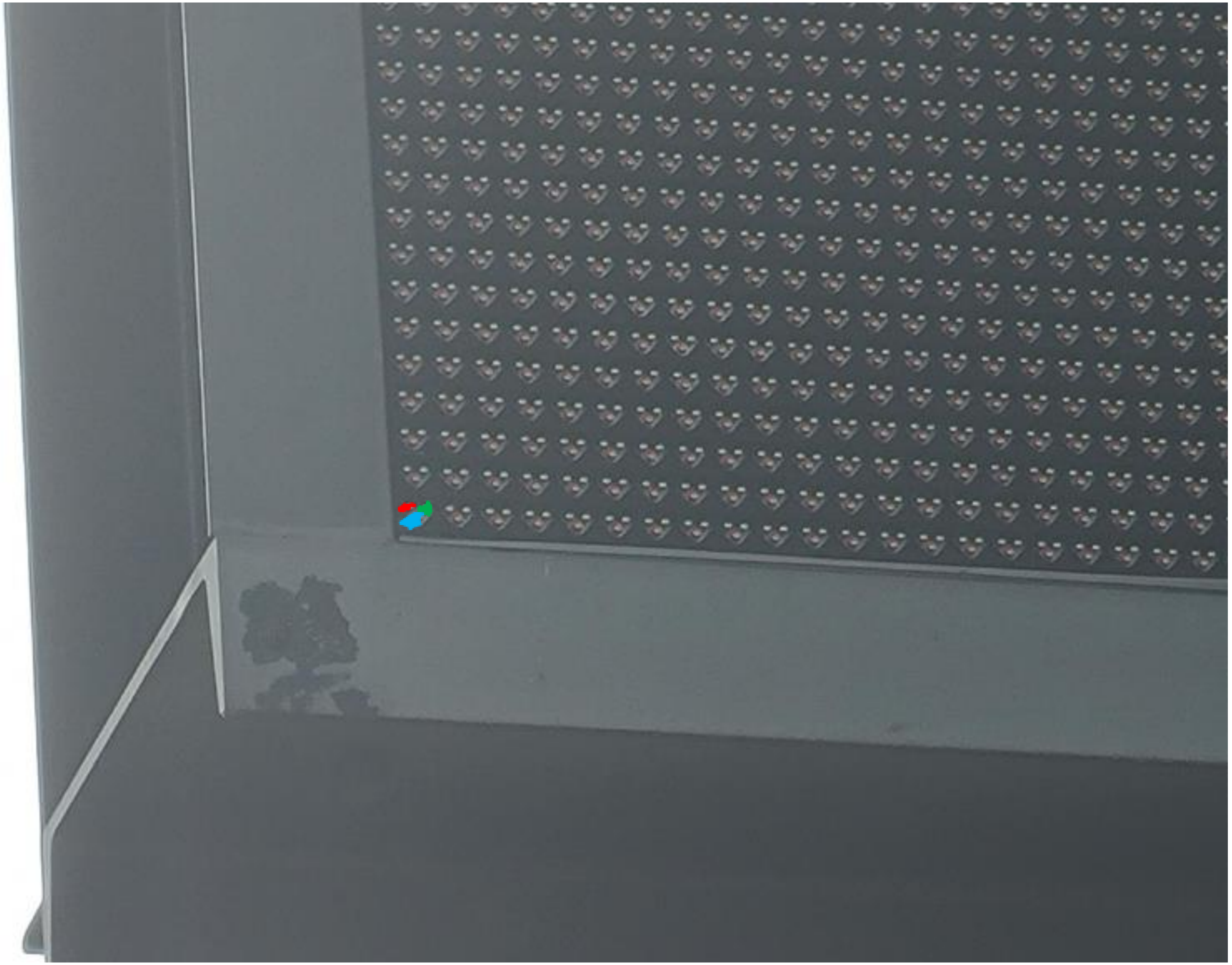


Image Features

Raw Data

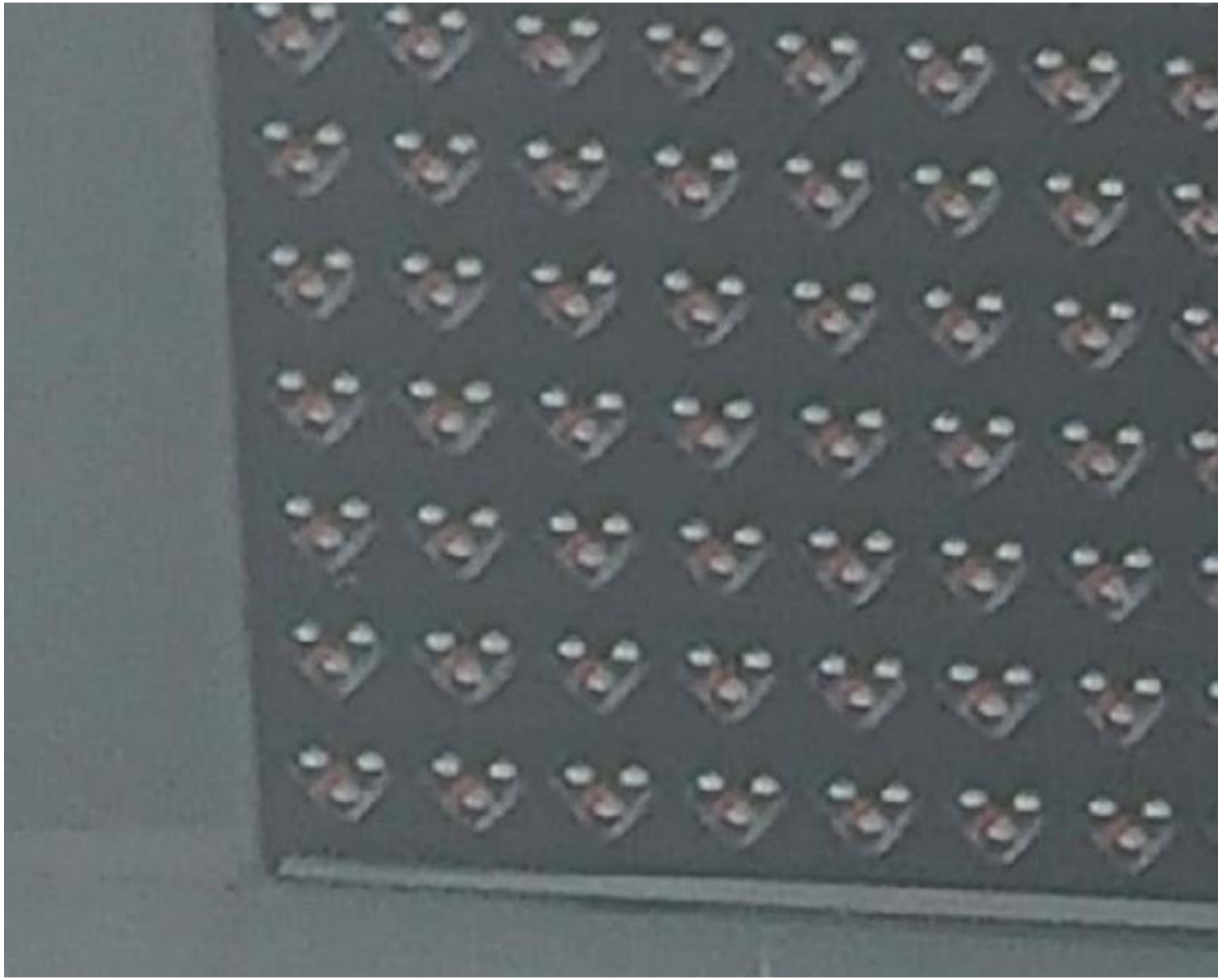


Image Features

Raw Data

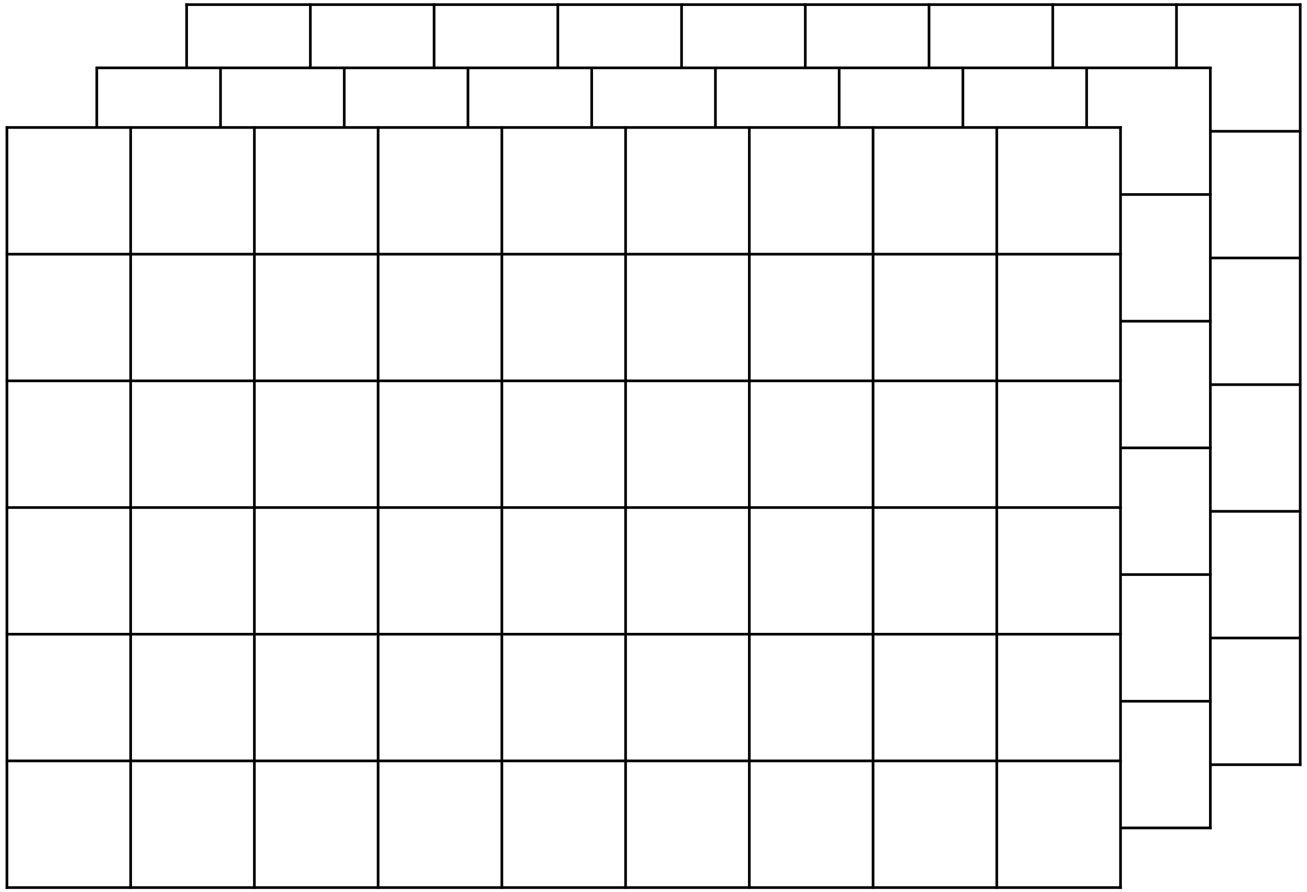
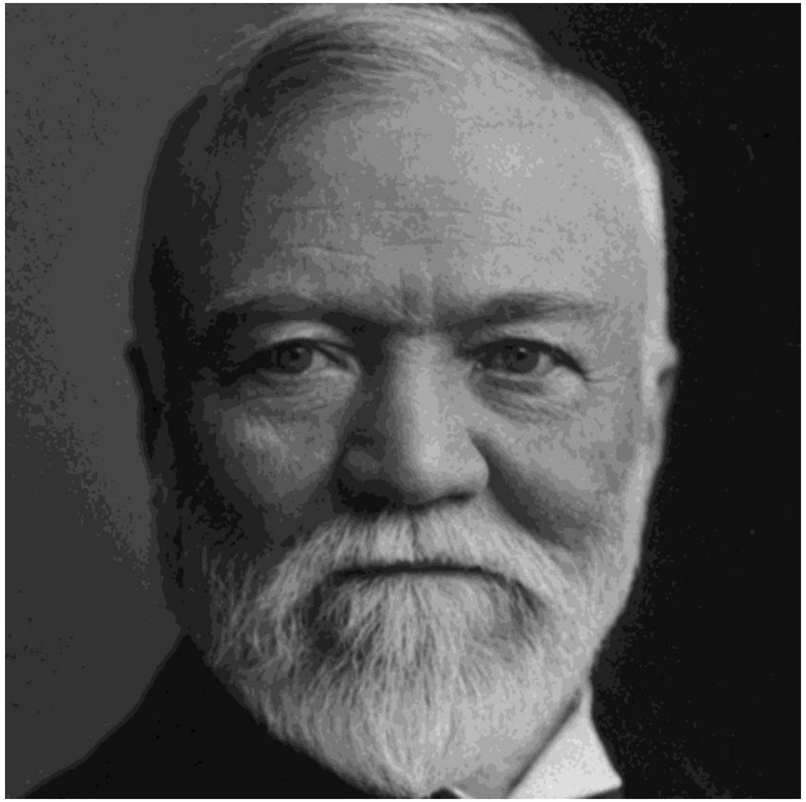


Image Features

Converting more complex data into a table of numerical values

Task: Face recognition from image



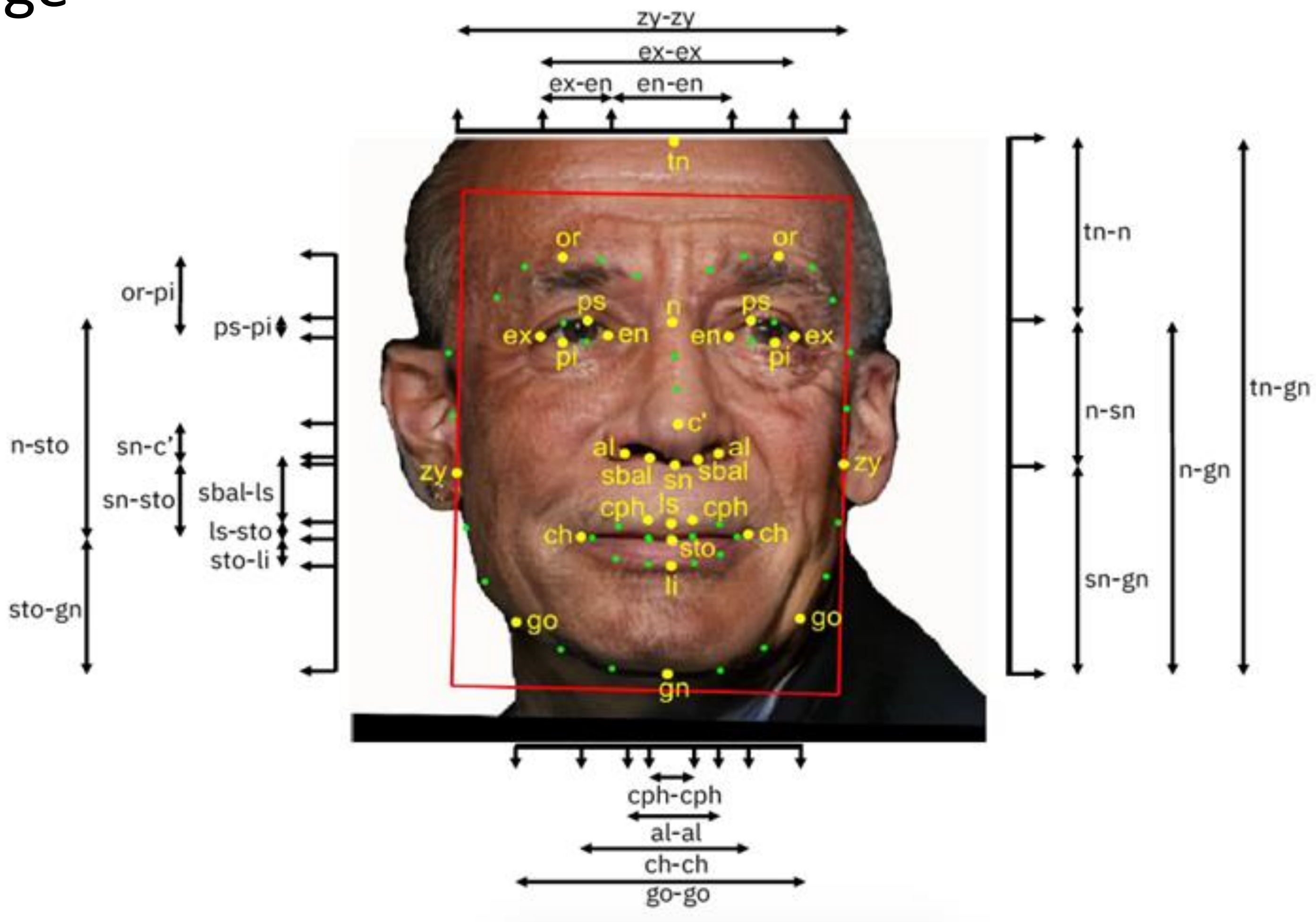


Image credit: https://arxiv.org/abs/1901.10436

Image Features

But what are the numerical features for e.g. animal classification??

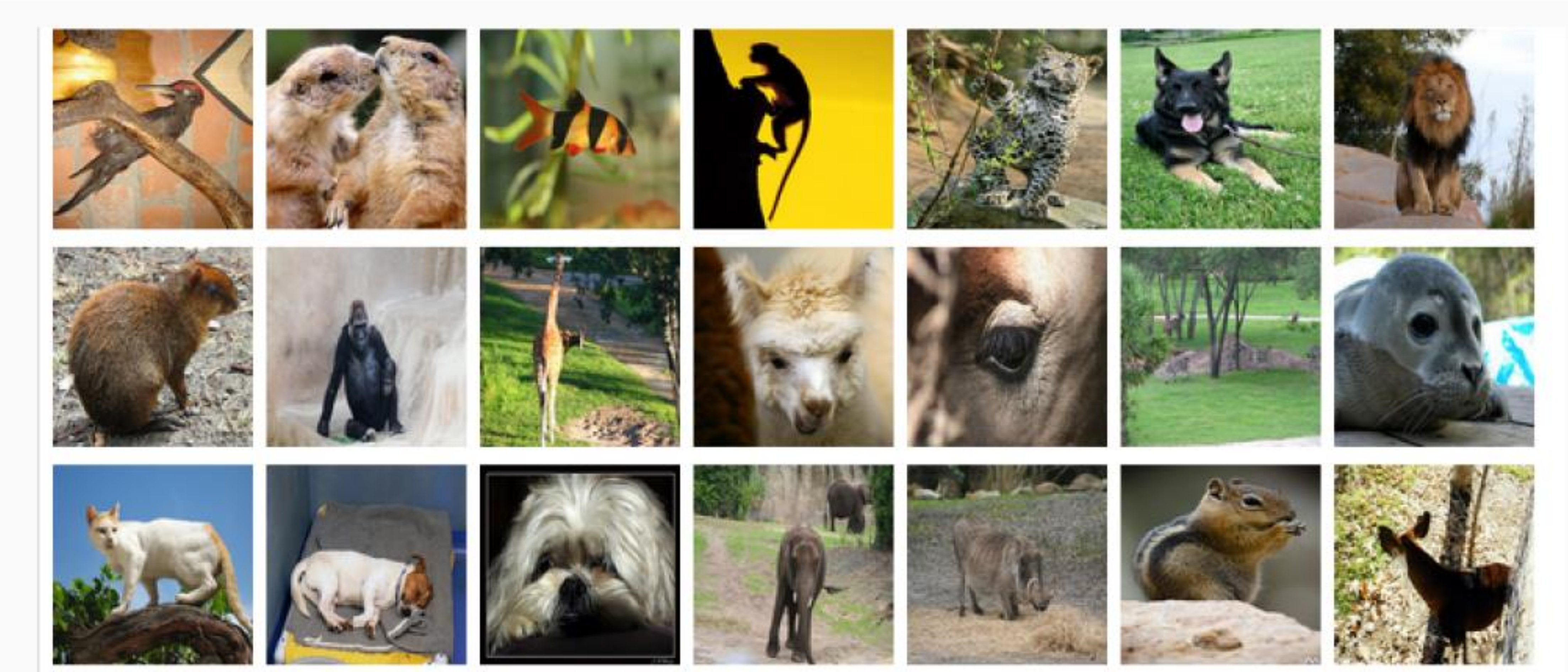


Image: ImageNet

Image Features

Feature engineering

Edge detection convolutions (direction and strength of edges in image patch)

HOG filer: Histogram of gradients (edges)

input image

visualization of HOG features

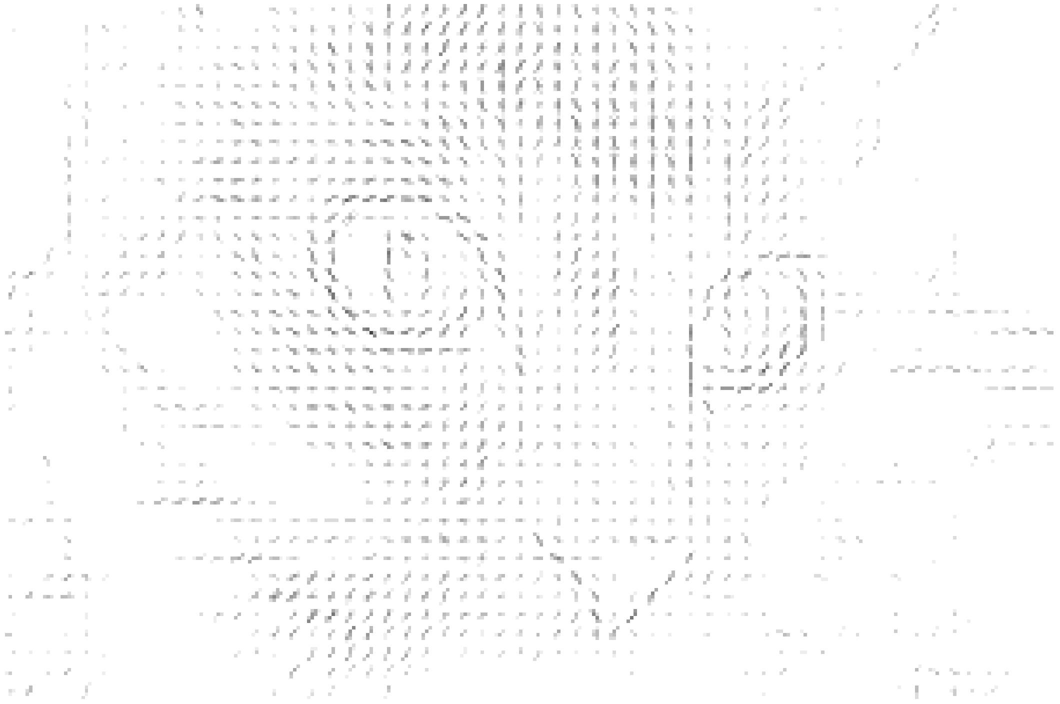


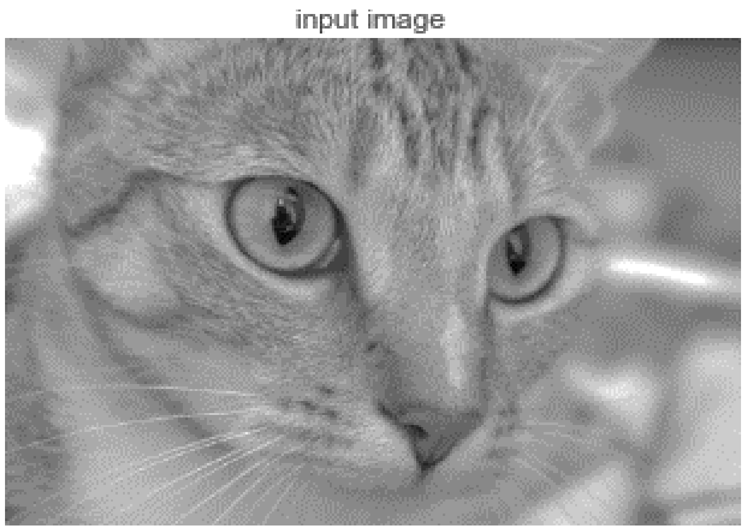
Image: https://jakevdp.github.io/PythonDataScienceHandbook/05.14-image-features.html

Image Features

Feature engineering

Edge detection convolutions (direction and strength of edges in image patch)

HOG filer: Histogram of gradients (edges)



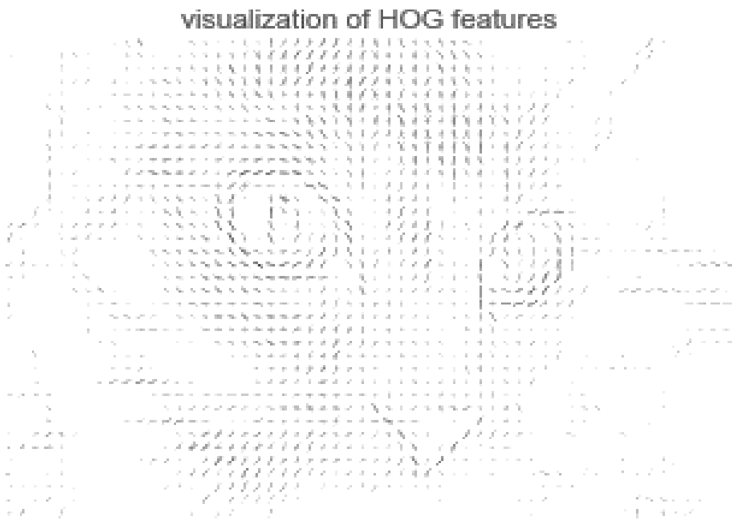


Image: https://jakevdp.github.io/PythonDataScienceHandbook/05.14-image-features.html

Image Features

Feature learning

Convolutional neural nets

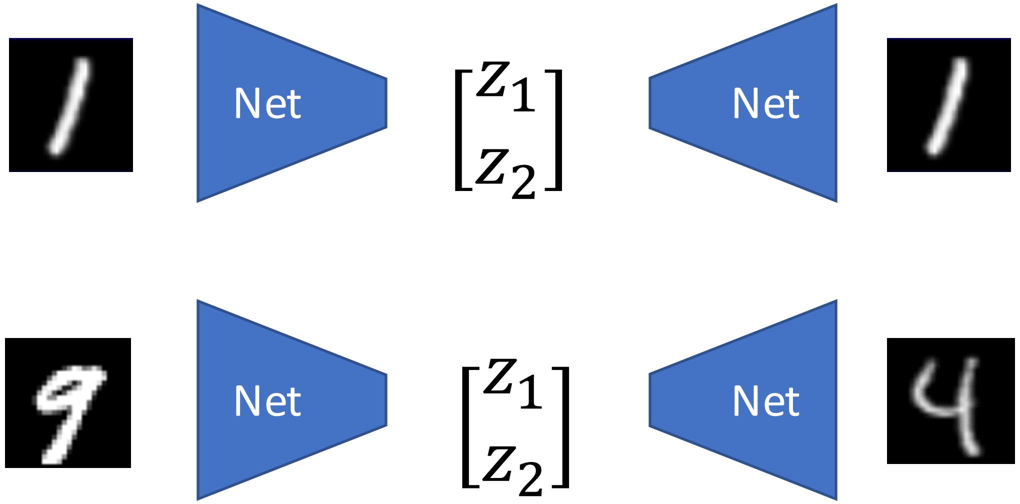
Image: https://jakevdp.github.io/PythonDataScienceHandbook/05.14-image-features.html

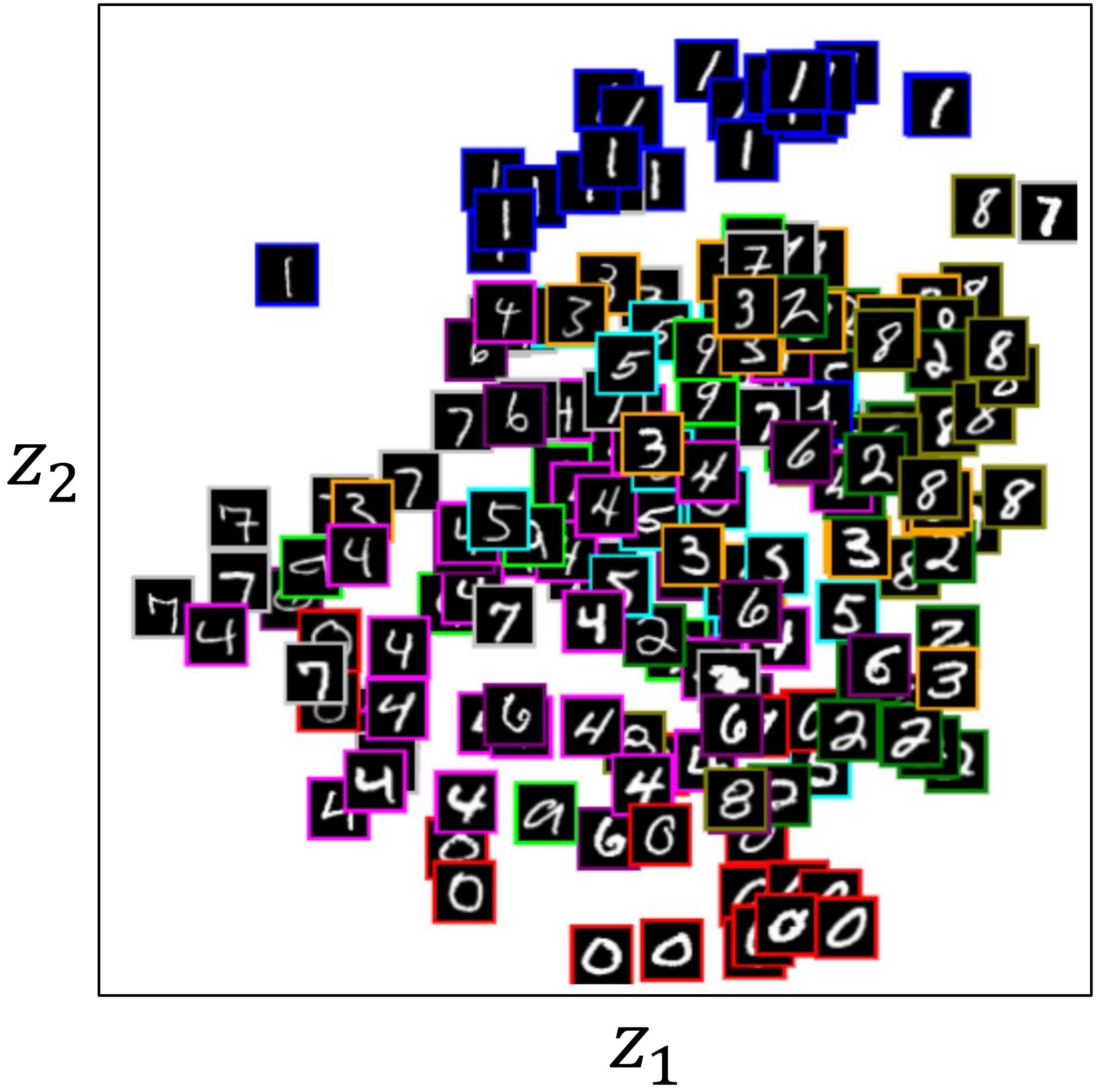
Image Features

Feature learning

Self-supervised, e.g. autoencoders

$$\text{Image} \rightarrow \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \rightarrow \text{Image}$$





https://cs.stanford.edu/people/karpathy/convnetjs/demo/autoencoder.html

Deep Learning for Computer Vision

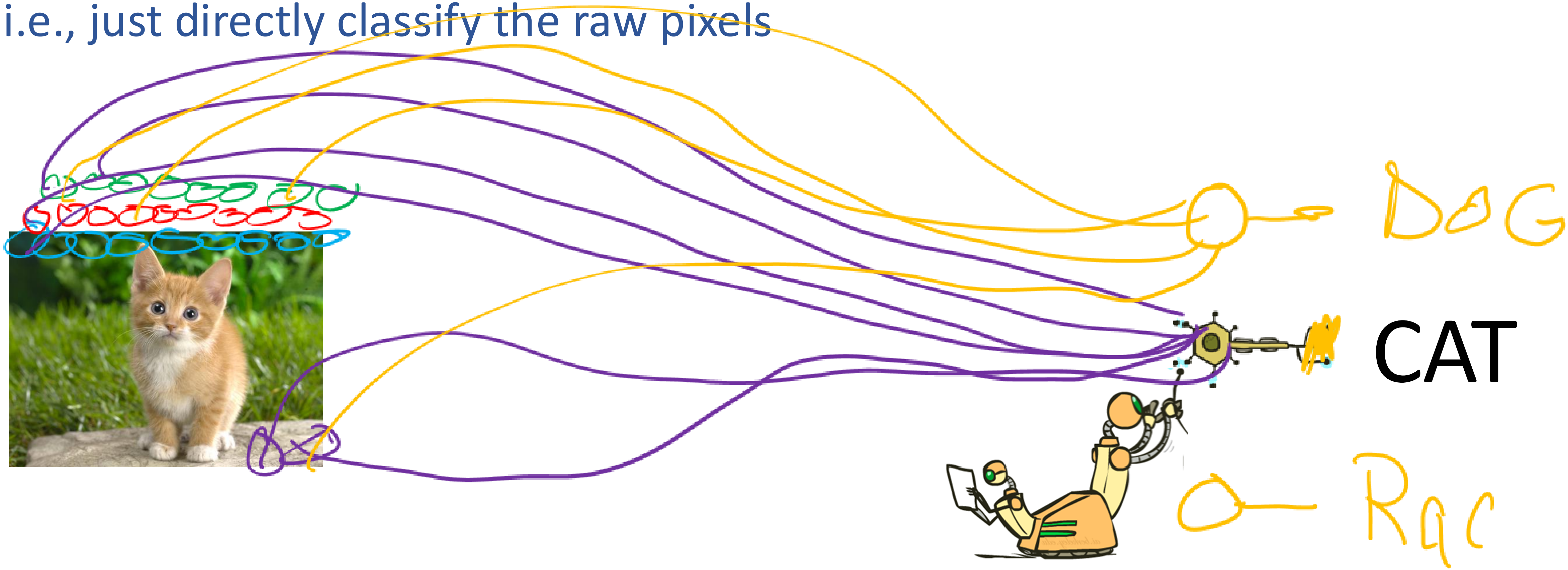
1. Computer vision tasks
2. Image features
3. Convolution Channels & CNNs
4. Computer vision history: Deep learning boom
5. CNN architectures & Practical deep learning

CNNs

Deep Learning for Images

What if we just used logistic regression?

i.e., just directly classify the raw pixels



Poll 1

Logistic regression for 28x28=784 pixel hand-written digit images into 10 classes:

How many parameters (including bias terms)?

A. 10
B. 10+784
C. 10*784
D. 10*784 + 10
E. 10*784 + 784
F. I don't know

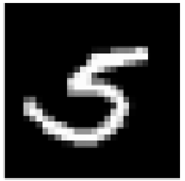
Poll 2

Given a training set of 1000 MNIST digits 0-9, what training accuracy do you think we can get using just logistic regression?

Value between 0.0 and 1.0.

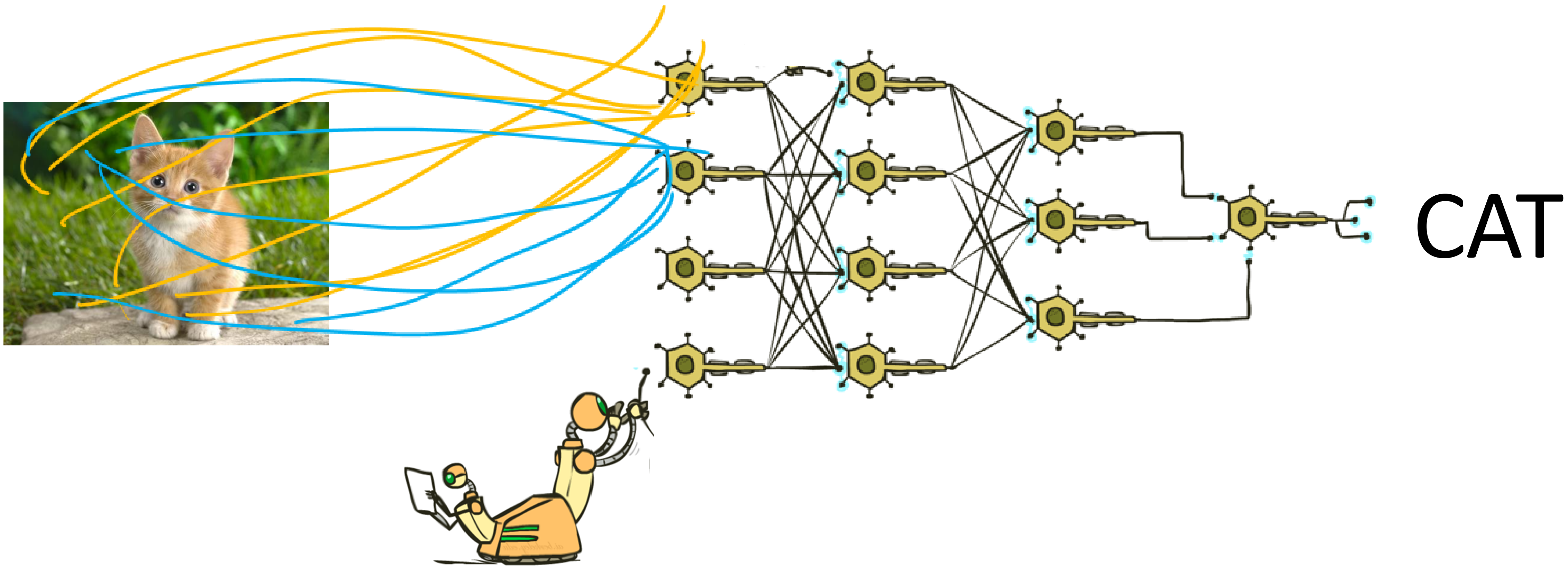
Input

(28x28)



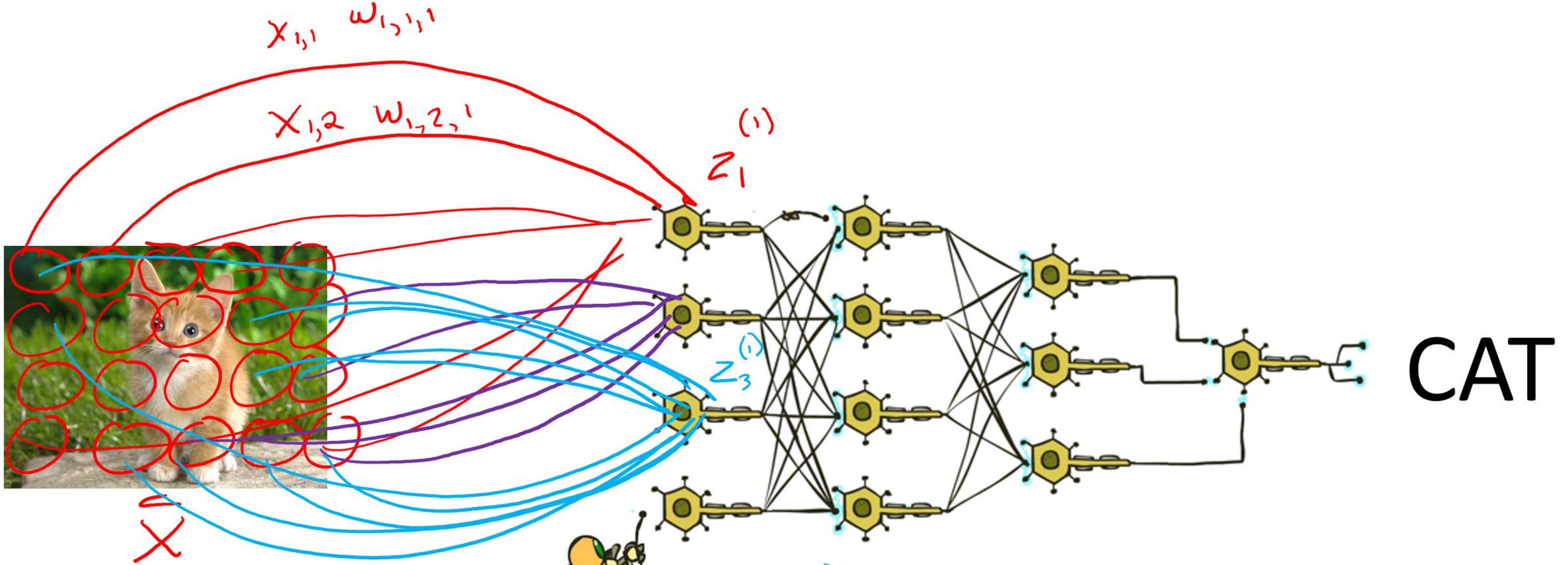
Classification: Deep Learning

What if we just used fully connected networks?



Classification: Deep Learning

What if we just used fully connected networks?



- Lost spatial content after first layer
- Not spatially invariant
- Too many weights to learn

Convolutional Layers

Convolutional layers (vs Fully-connected layers)

- Nodes in the input layer are only connected to some nodes in the next layer but not all nodes.
- Shared weight parameters across sliding window locations

→ Many fewer weights than a fully connected layer!

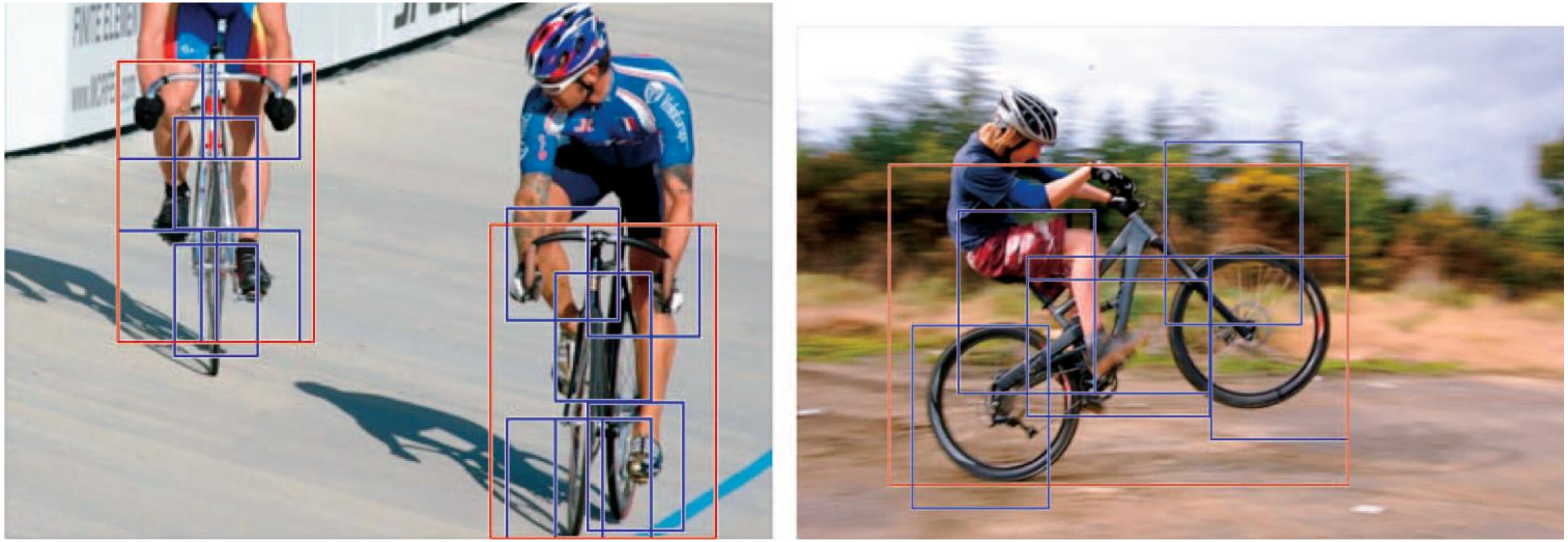
Just like fully-connected layers

- Linear operations
- Parameters are weights (the values in conv. kernel) and biases
- Convolution weights are learned using gradient descent/ backpropagation (not hand-crafted)

Convolutional Neural Networks

Neural networks are frequently applied to inputs with some inherent spatial structure, e.g., images

Insight: for spatially-structured inputs, many useful features are shift or location-invariant



Source: [Felzenszwalb, et al. PAMI 2009]

Convolutional Neural Networks

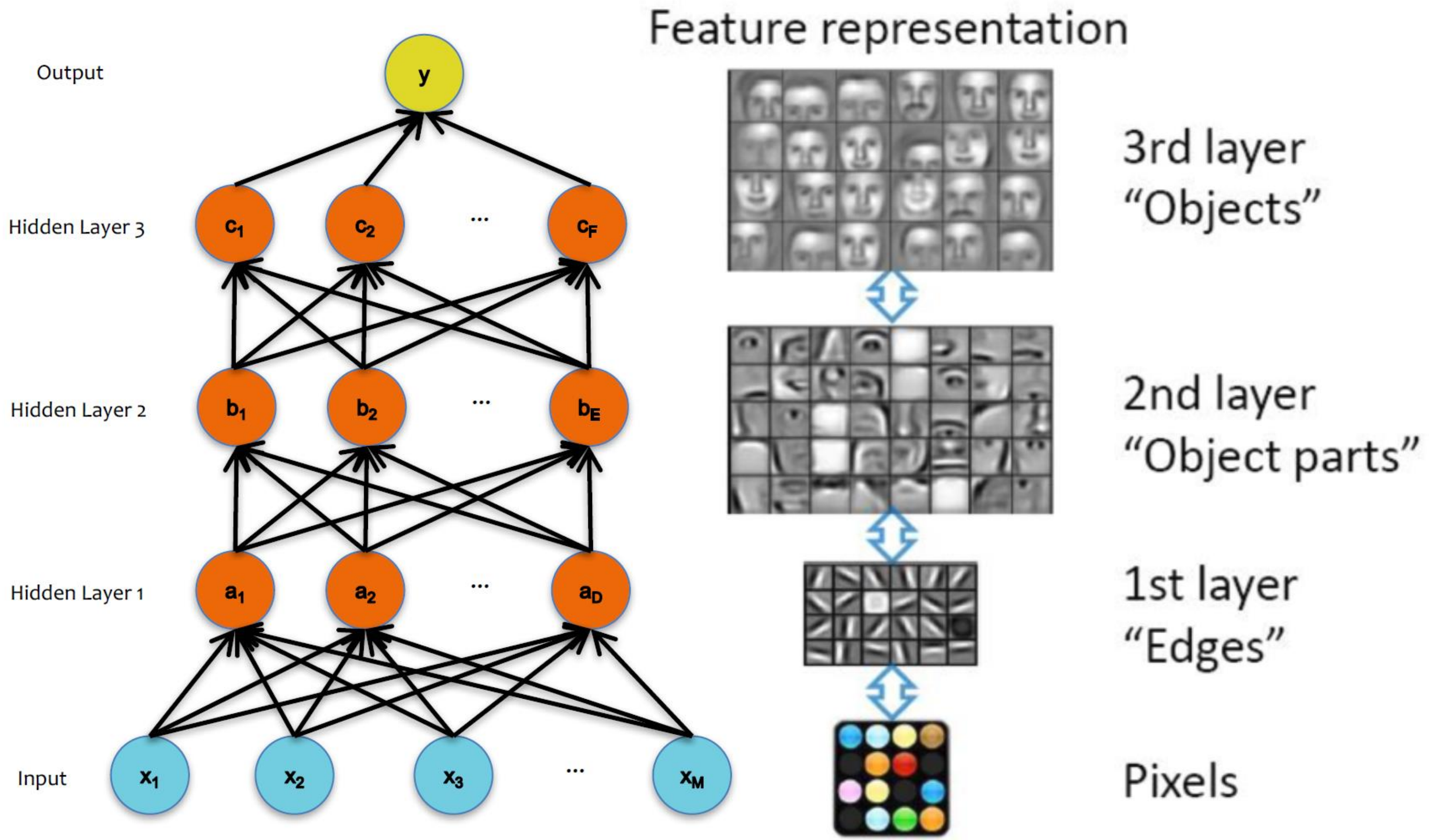
Neural networks are frequently applied to inputs with some inherent spatial structure, e.g., images

Insight: for spatially-structured inputs, many useful features are shift or location-invariant

Strategy:

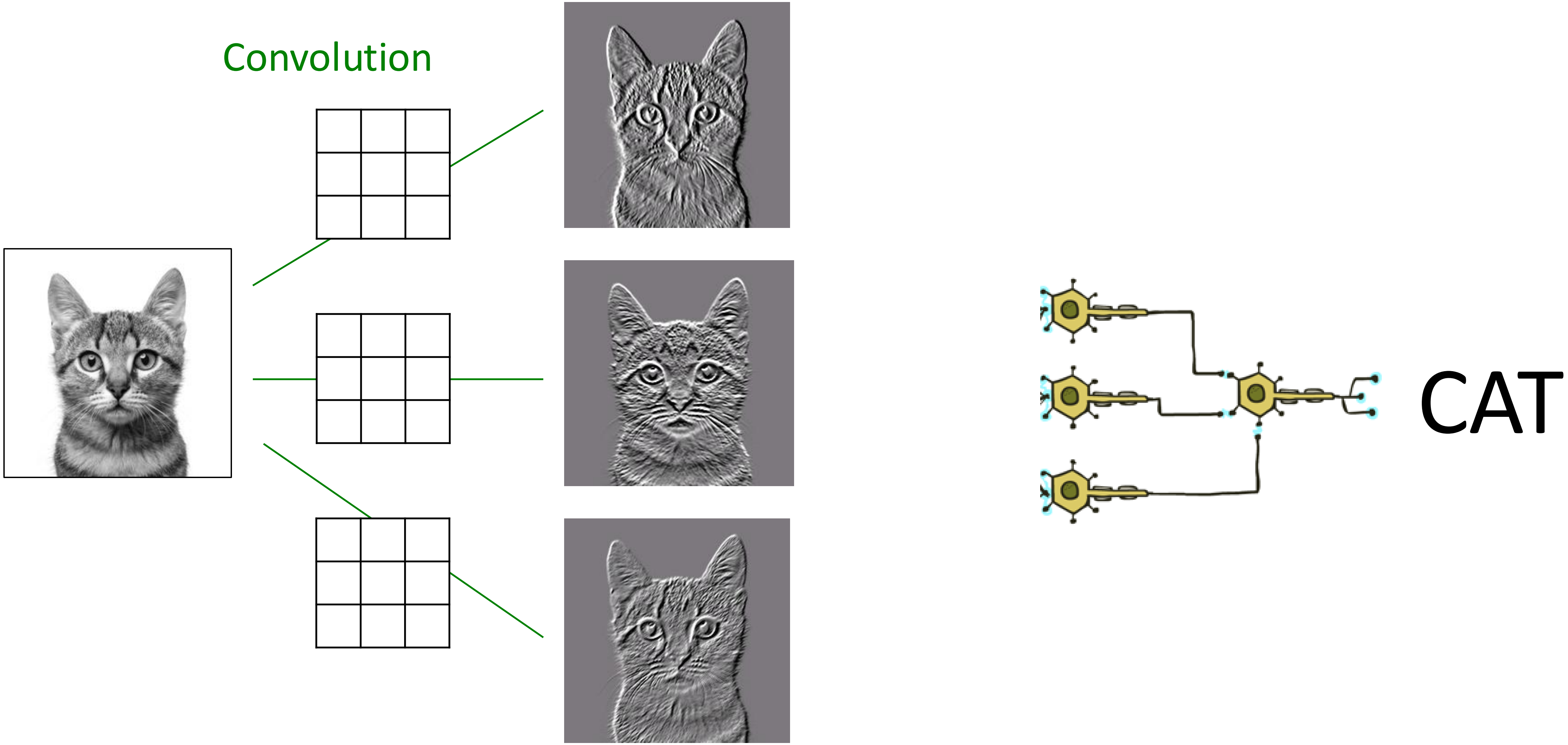
a) Learn a *filter* for *micro*-feature detection in a small window and apply it over the entire image
b) Downsample (shink) resulting feature image(s)
c) Repeat at future layers to learn increasingly *macro* features

Classification: Learning Features

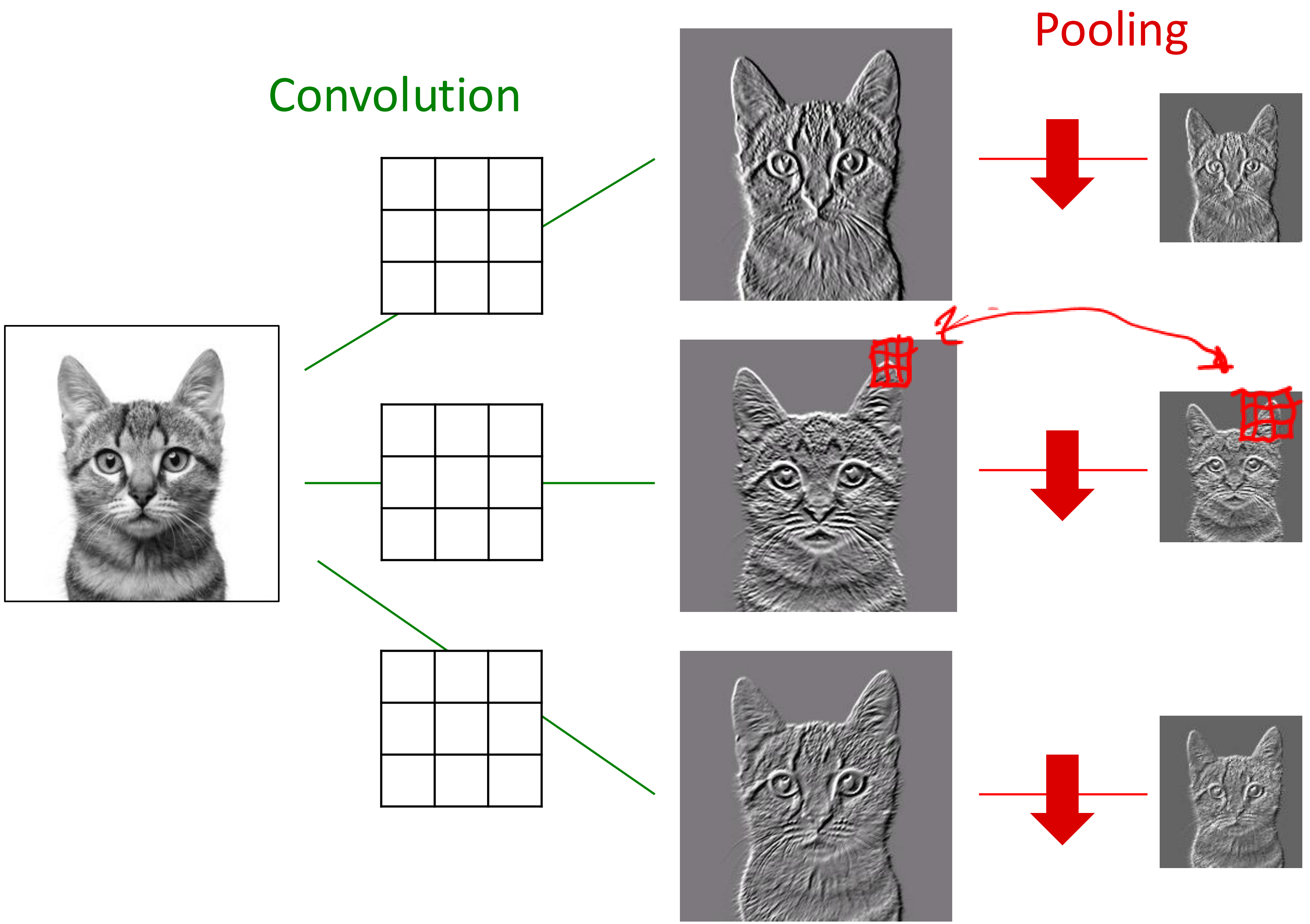


Example from Honglak Lee (NIPS 2010)

Convolutional Neural Networks

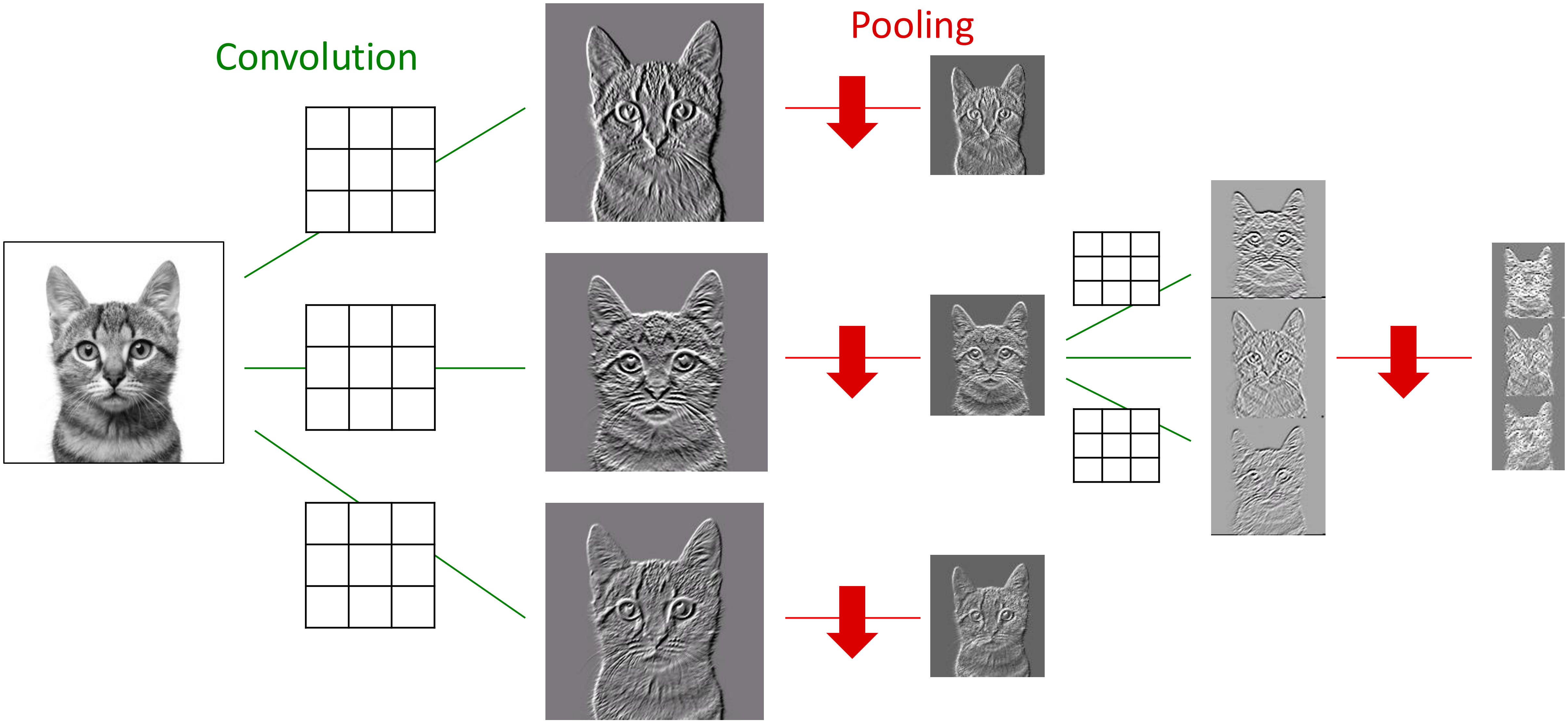


Convolutional Neural Networks

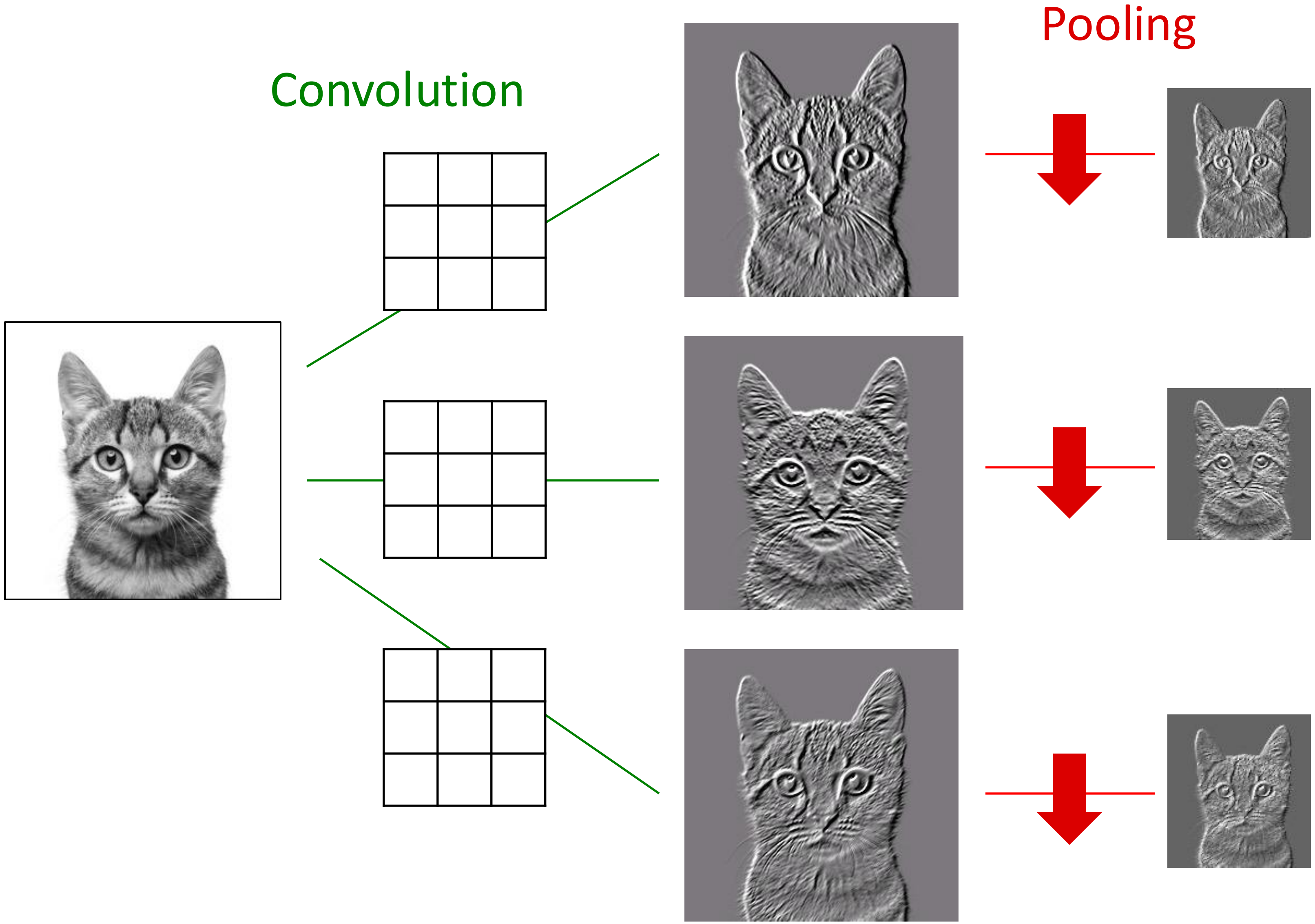


Convolutional Neural Networks

Warning: Not quite right



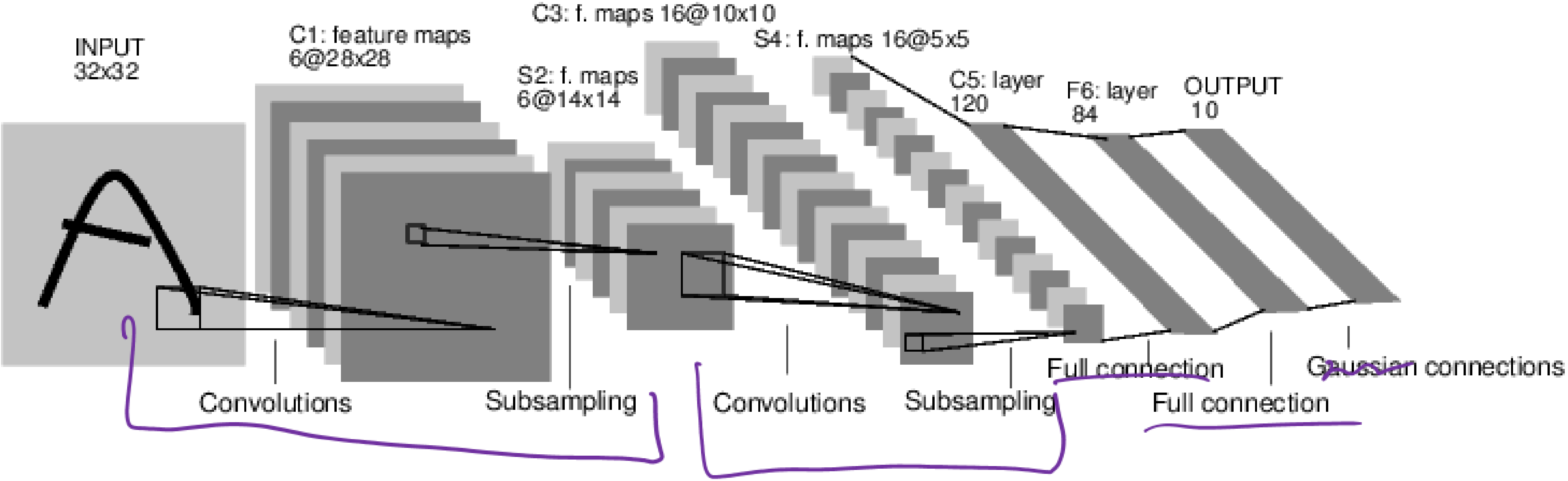
Convolutional Neural Networks



Convolutional Neural Networks

Lenet5 – Lecun, et al, 1998

- Convnets for digit recognition

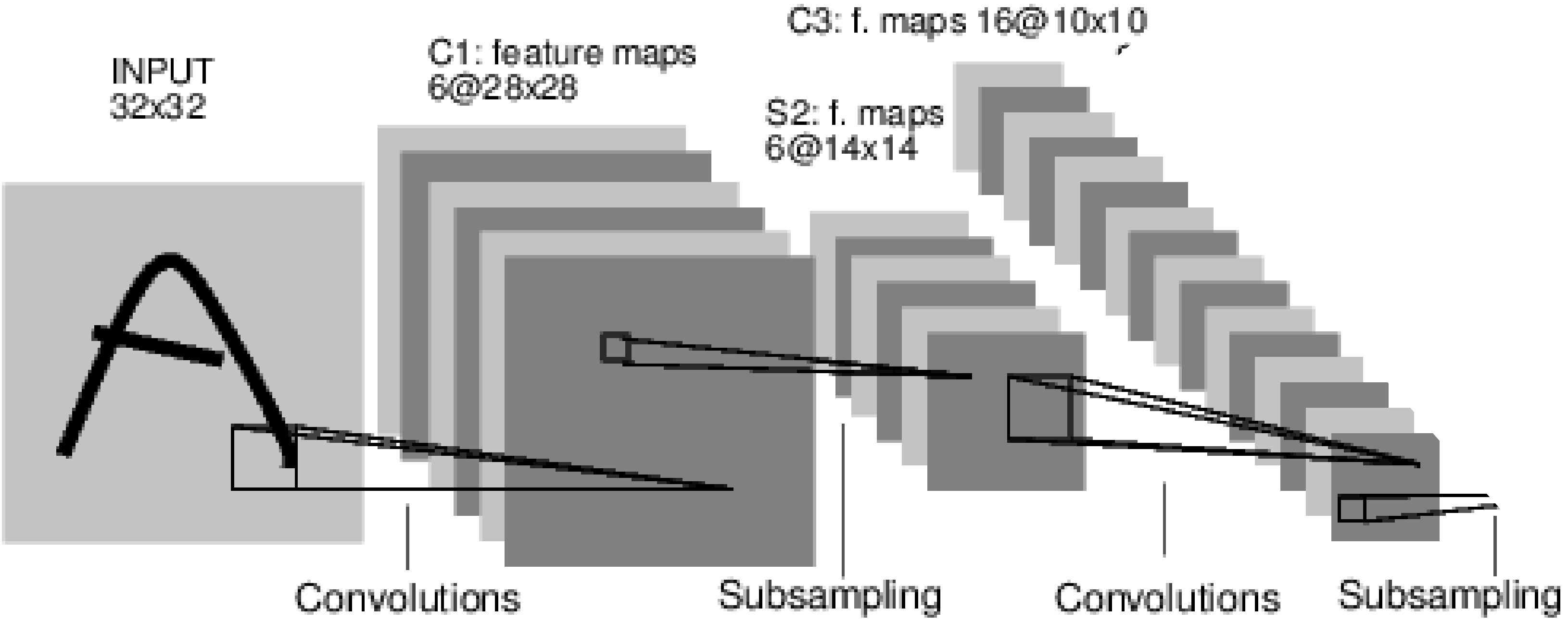


LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.

Question:

How big many convolutional weights between S2 and C3?

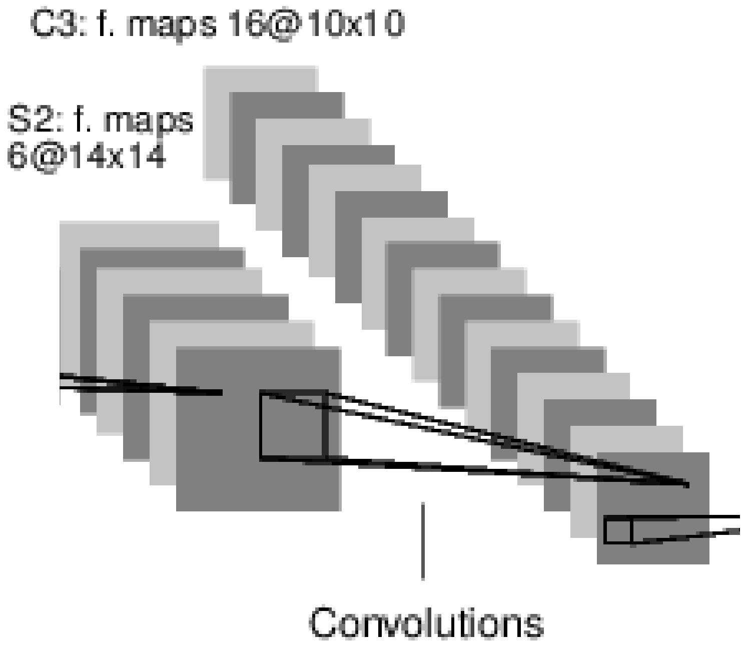
- S2: 6 channels @14x14
- Conv: 5x5, pad=0, stride=1
- C3: 16 channels @ 10x10



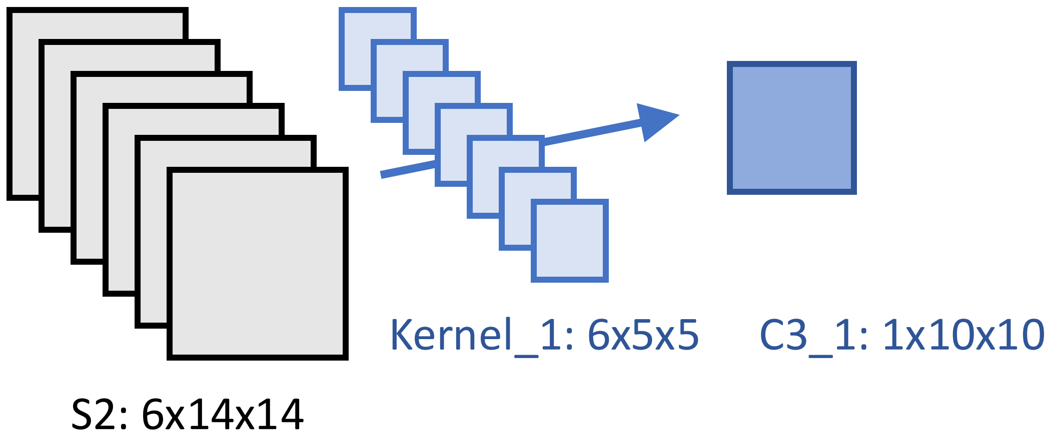
Question:

How big many convolutional weights between S2 and C3?

- S2: 6 channels @14x14
- Conv: 5x5, pad=0, stride=1
- C3: 16 channels @ 10x10



One image in C3 is actually the result of a 3D convolution

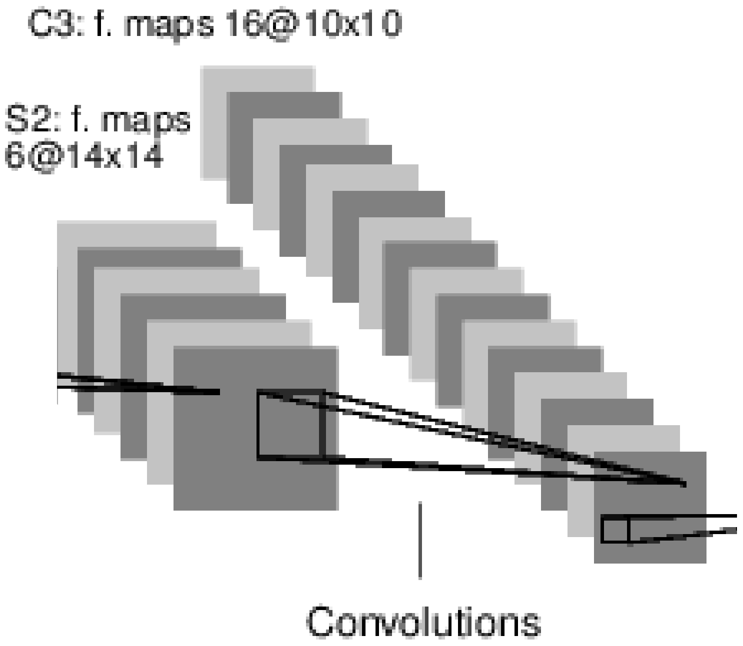


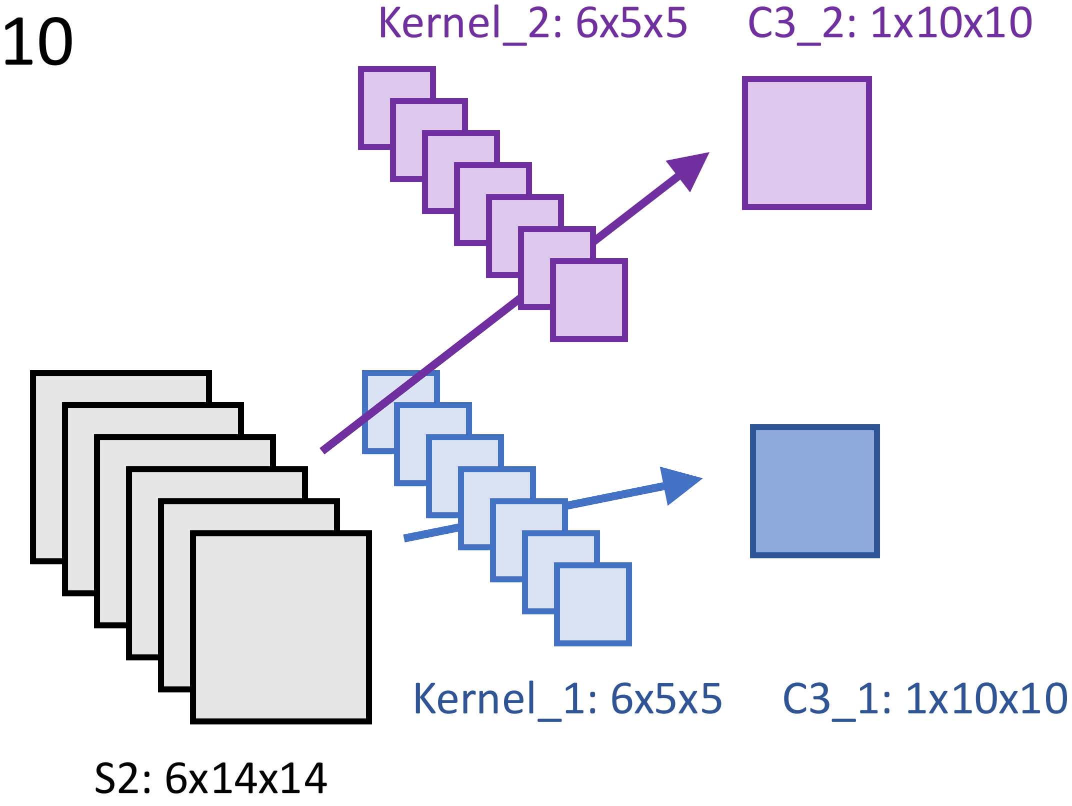
Question:

How big many convolutional weights between S2 and C3?

- S2: 6 channels @14x14
- Conv: 5x5, pad=0, stride=1
- C3: 16 channels @ 10x10

Each image in C3 convolved S2 convolved with a different 3D kernel



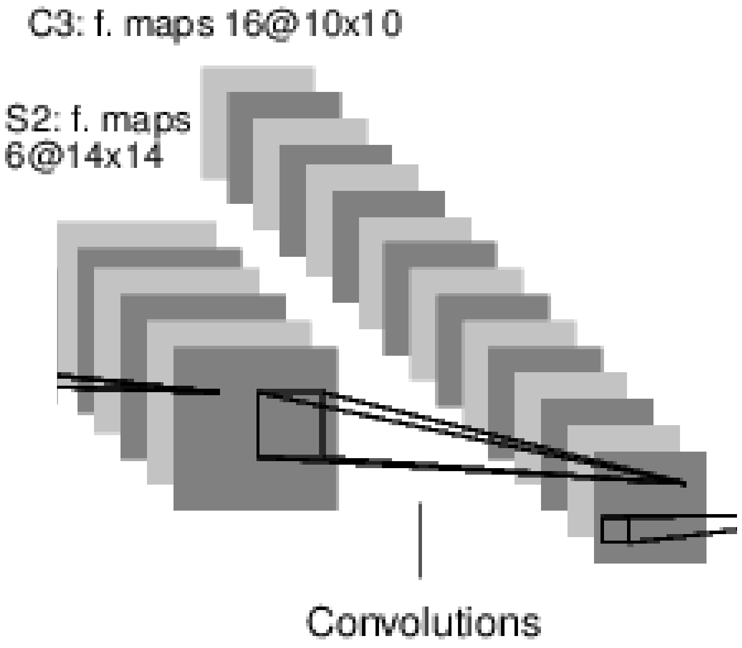


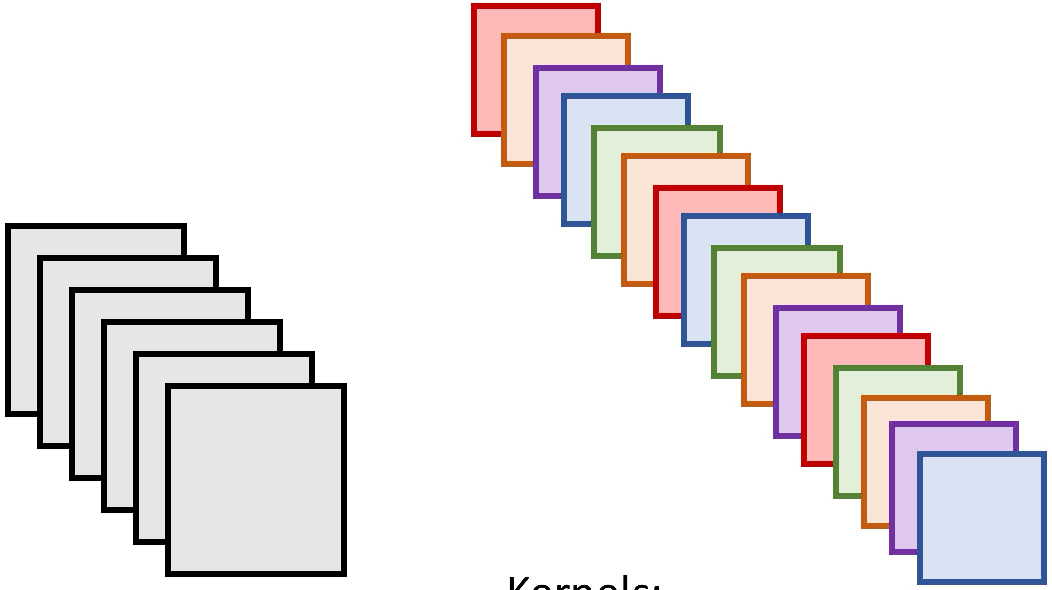
Question:

How big many convolutional weights between S2 and C3?

- S2: 6 channels @14x14
- Conv: 5x5, pad=0, stride=1
- C3: 16 channels @ 10x10

The 16 images in C3 are the result of doing 16 3D convolutions of S2 with 16 different 6x5x5 kernels. Assuming no bias term, this is 16x6x5x5 weights!





S2: 6x14x14

Kernels:
16@6x5x5

C3: 16@10x10

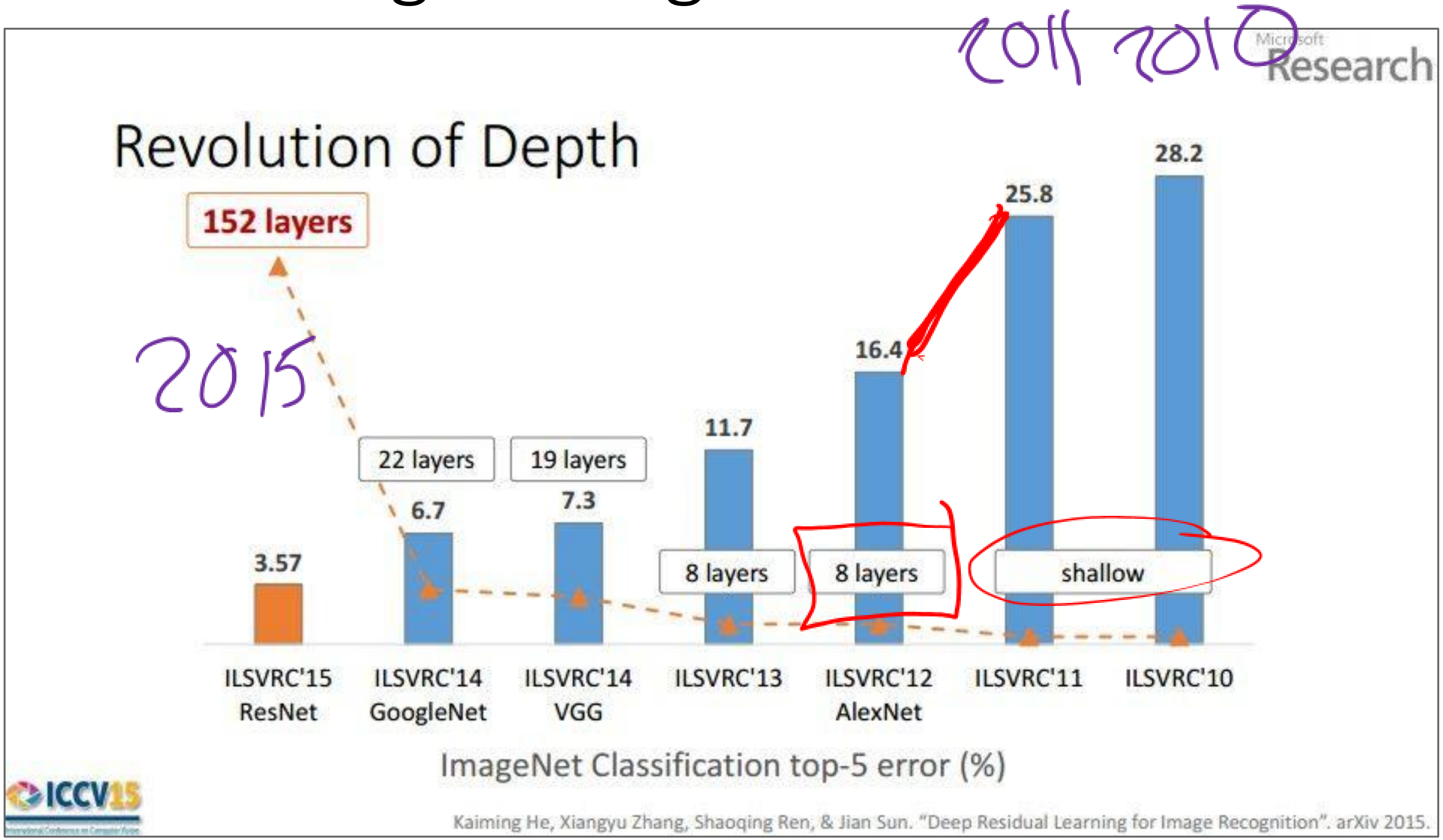
Deep Learning for Computer Vision

1. Computer vision tasks
2. Image features
3. Convolution Channels & CNNs
4. Computer vision history: Deep learning boom
5. CNN architectures & Practical deep learning

Computer Vision History

Deep Learning Boom

CNNs for Image Recognition



Slide from Kaiming He

What happened in 2012?

The challenge.

Computer Vision

IMAGENET

Jitendra Malik

Fei-Fei Li

Ilya and Alex

Neural Networks

Geoff Hinton

Images: https://www.nytimes.com/2016/09/20/science/computer-vision-tesla-driverless-cars.html
https://www.utoronto.ca/news/google-acquires-u-t-neural-networks-company

Vision / Computer Vision History

LeNet 5, 1998

Input: 1, 32, 32

```
nn.Conv2d(out_channels=6, kernel_size=5),
            nn.Tanh(),
nn.AvgPool2d(kernel_size=2, stride=2),
nn.Conv2d(out_channels=16, kernel_size=5),
            nn.Tanh(),
nn.AvgPool2d(kernel_size=2, stride=2)

nn.Linear(out_features=120),
            nn.Tanh(),
nn.Linear(out_features=84),
            nn.Tanh(),
nn.Linear(out_features=10)
```

Network changes (other than bigger, deeper)

- tanh → ReLU
- Avg Pool → Max Pool
- + Batch Normalization (keep values in reasonable range)
- + Dropout (form of regularization)

AlexNet, 2012

Input: 3, 224, 224

```
nn.Conv2d(channels=96, kernel_size=11, stride=4),
            nn.BatchNorm(),   nn.ReLU(),
nn.MaxPool2d(kernel_size=3, stride=2),
nn.Conv2d(channels=256, kernel_size=5),
            nn.BatchNorm(256),   nn.ReLU(),
nn.MaxPool2d(kernel_size=3, stride=2),
nn.Conv2d(channels=384, kernel_size=3),
            nn.BatchNorm(384),   nn.ReLU(),
nn.Conv2d(channels=384, kernel_size=3),
            nn.BatchNorm(384),   nn.ReLU(),
nn.Conv2d(channels=256, kernel_size=3),
            nn.BatchNorm(256),   nn.ReLU(),
nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Dropout(0.5),
nn.Linear(channels=4096), nn.ReLU(),
            nn.Dropout(0.5),
nn.Linear(channels=4096), nn.ReLU(),
nn.Linear(channels=1000)
```

Vision / Computer Vision History

State-of-the-art Classification

Model	Top-1	Top-5
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

Table 1: Comparison of results on ILSVRC-2010 test set. In *italics* are best results achieved by others.

Network changes (other than bigger, deeper)

- tanh → ReLU
- Avg Pool → Max Pool
- + Batch Normalization (keep values in reasonable range)
- + Dropout (form of regularization)

AlexNet, 2012

```
Input: 3, 224, 224
nn.Conv2d(channels=96, kernel_size=11, stride=4),
            nn.BatchNorm(),   nn.ReLU(),
nn.MaxPool2d(kernel_size=3, stride=2),
nn.Conv2d(channels=256, kernel_size=5),
            nn.BatchNorm(256),   nn.ReLU(),
nn.MaxPool2d(kernel_size=3, stride=2),
nn.Conv2d(channels=384, kernel_size=3),
            nn.BatchNorm(384),   nn.ReLU(),
nn.Conv2d(channels=384, kernel_size=3),
            nn.BatchNorm(384),   nn.ReLU(),
nn.Conv2d(channels=256, kernel_size=3),
            nn.BatchNorm(256),   nn.ReLU(),
nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Dropout(0.5),
nn.Linear(channels=4096), nn.ReLU(),
            nn.Dropout(0.5),
nn.Linear(channels=4096), nn.ReLU(),
nn.Linear(channels=1000)
```

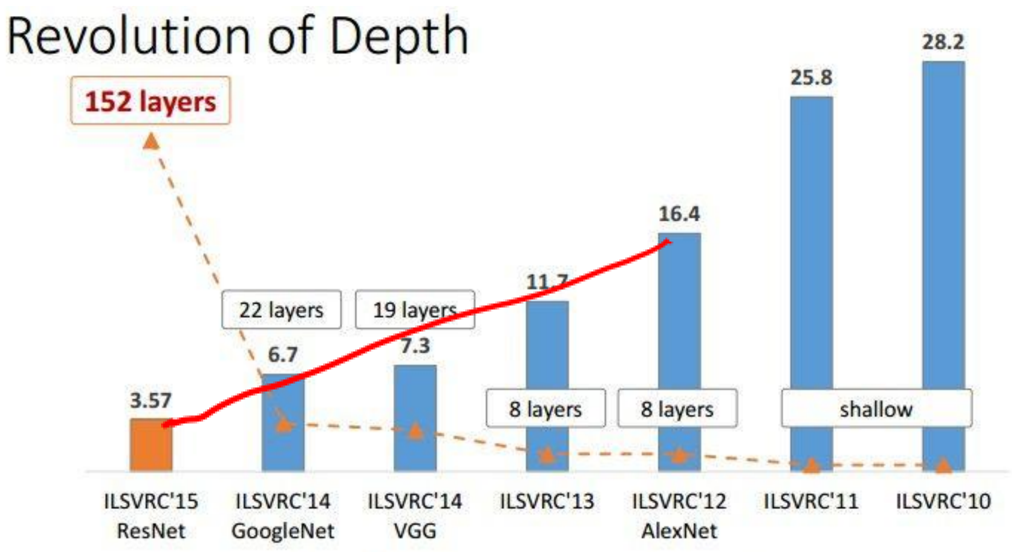
2012: AlexNet (+ ImageNet + GPUs) opened doors

Key

- Network + Data + Compute

Additional innovations

- Deep learning toolkits
  - (Caffe, Torch) → (PyTorch, Tensorflow)
  - "Model zoos"
- Residual connections (ResNet)
  - (aka skip connections)
  - Shortcuts for information flow (forward and backward)



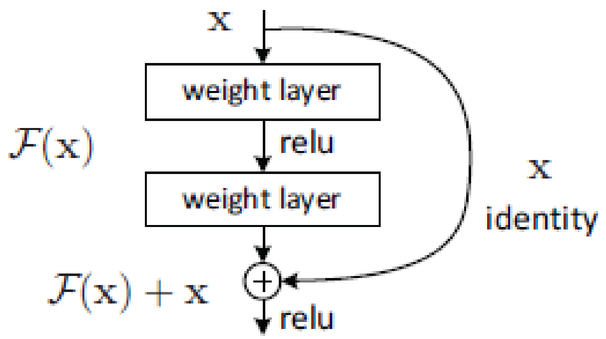
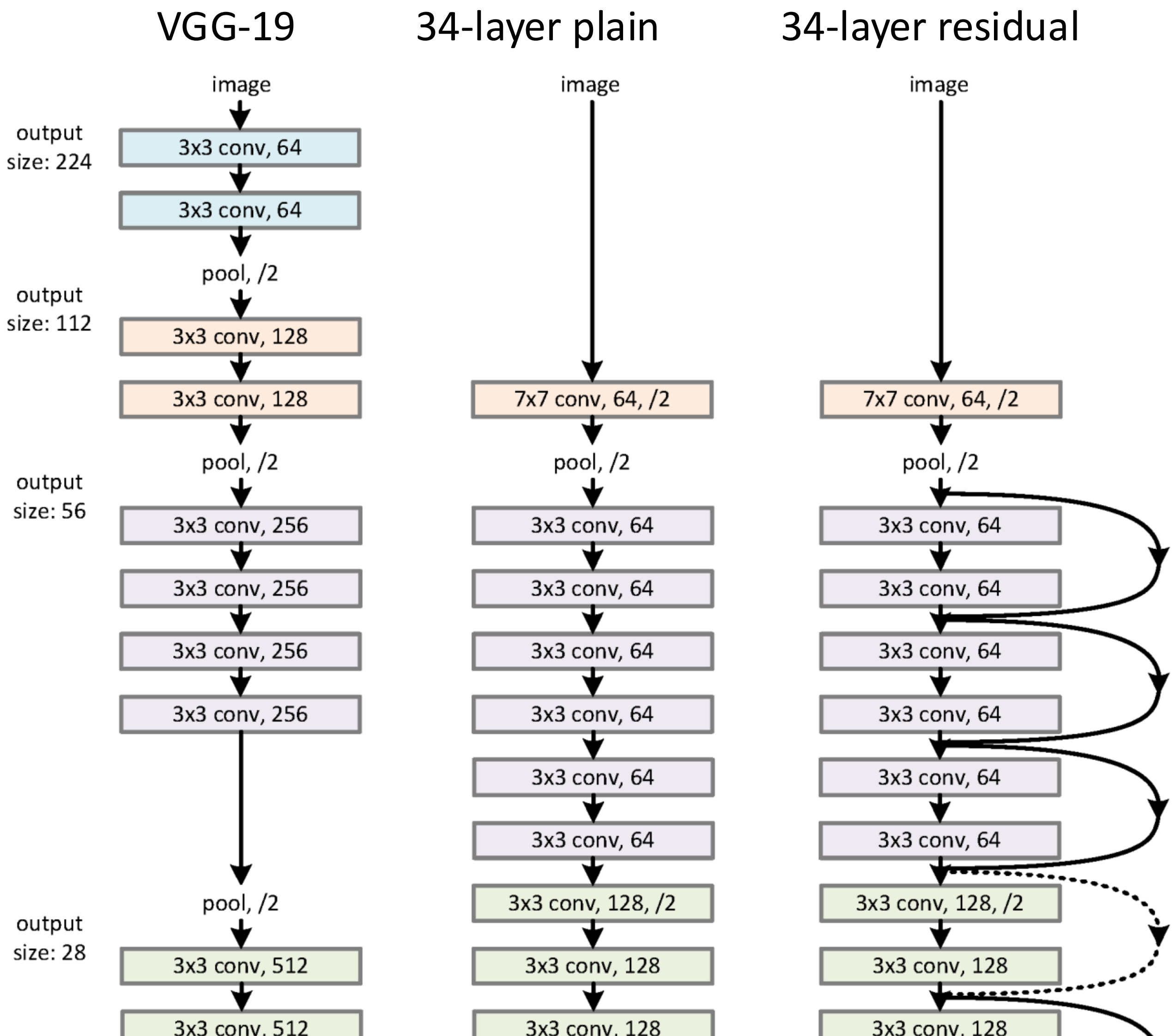


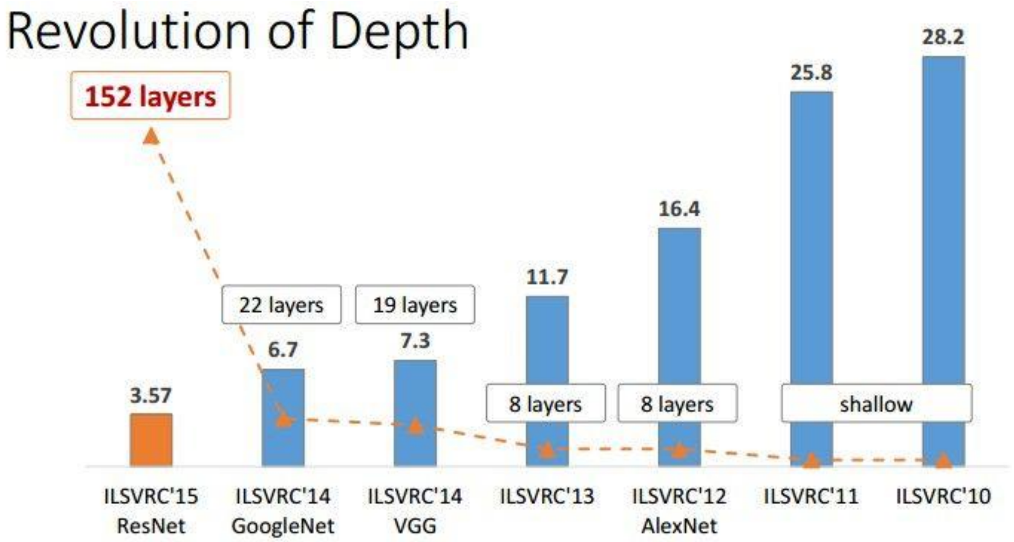
Figure 2. Residual learning: a building block

Kaiming He, et al, Deep Residual Learning for Image Recognition

2012: AlexNet (+ ImageNet + GPUs) opened doors

Kaiming He, et al, Deep Residual Learning for Image Recognition





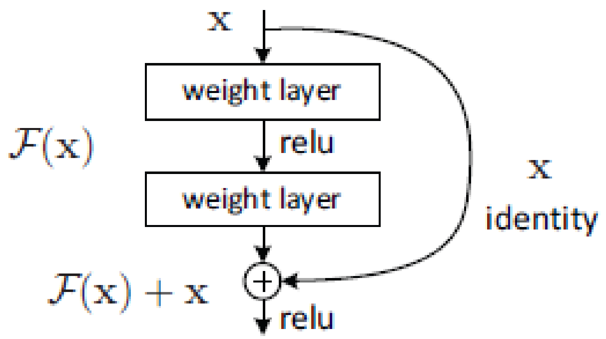


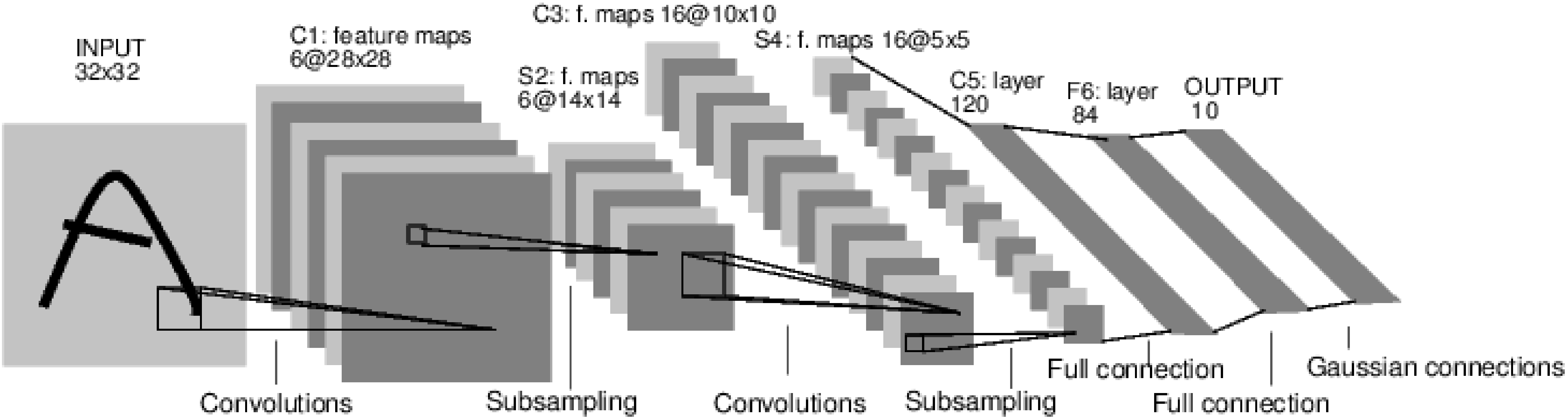
Figure 2. Residual learning: a building block

CNN Architectures

Convolutional Neural Networks

Lenet5 – Lecun, et al, 1998

- Convnets for digit recognition

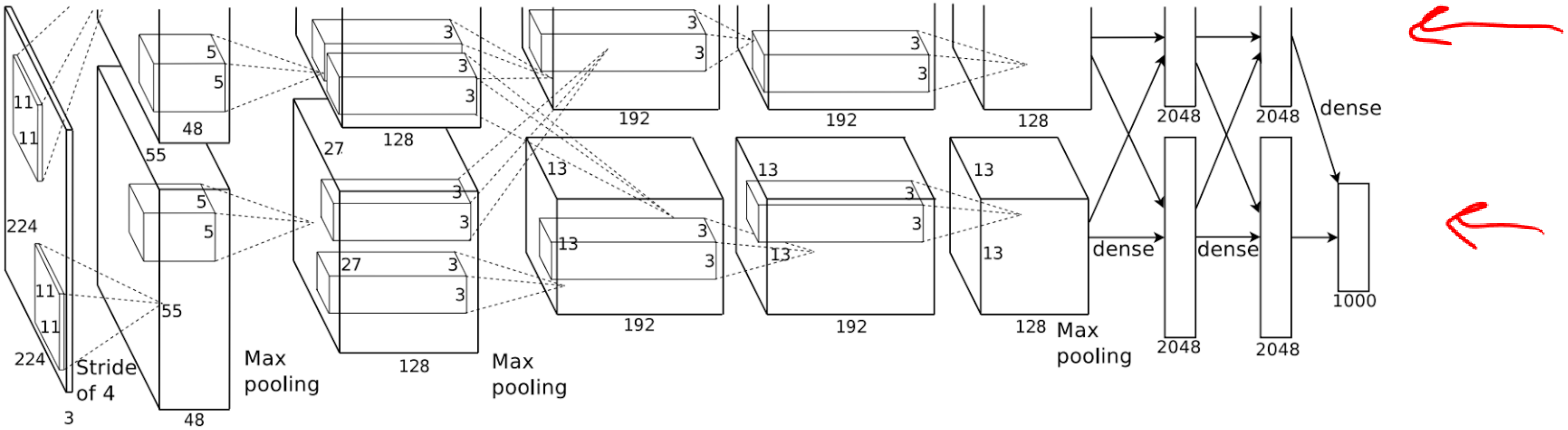


LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.

Convolutional Neural Networks

Alexnet – Krizhevsky, et al, 2012

- Convnets for image classification
- More data & more compute power



Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks." NIPS, 2012.

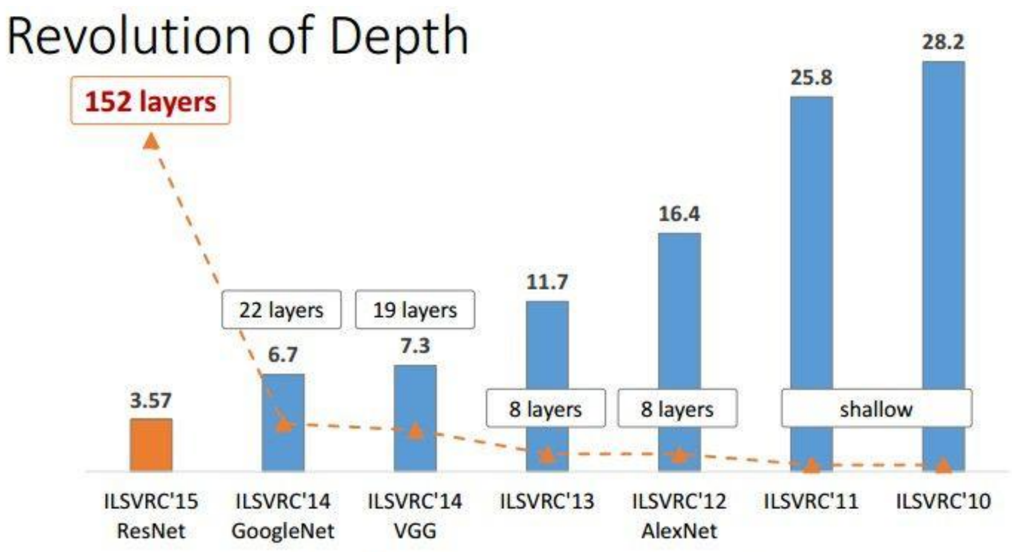
2012: AlexNet (+ ImageNet + GPUs) opened doors

Key

- Network + Data + Compute

Additional innovations

- Deep learning toolkits
  - (Caffe, Torch) → (PyTorch, Tensorflow)
  - "Model zoos"
- Residual connections (ResNet)
  - (aka skip connections)
  - Shortcuts for information flow (forward and backward)



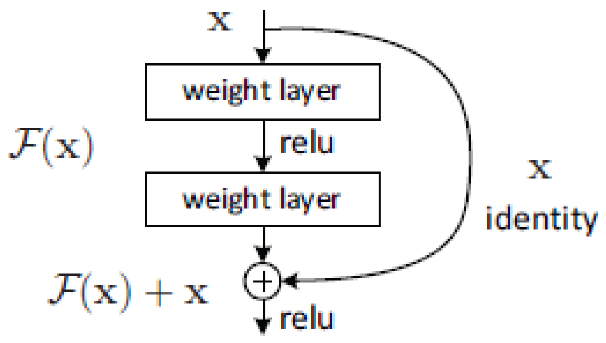
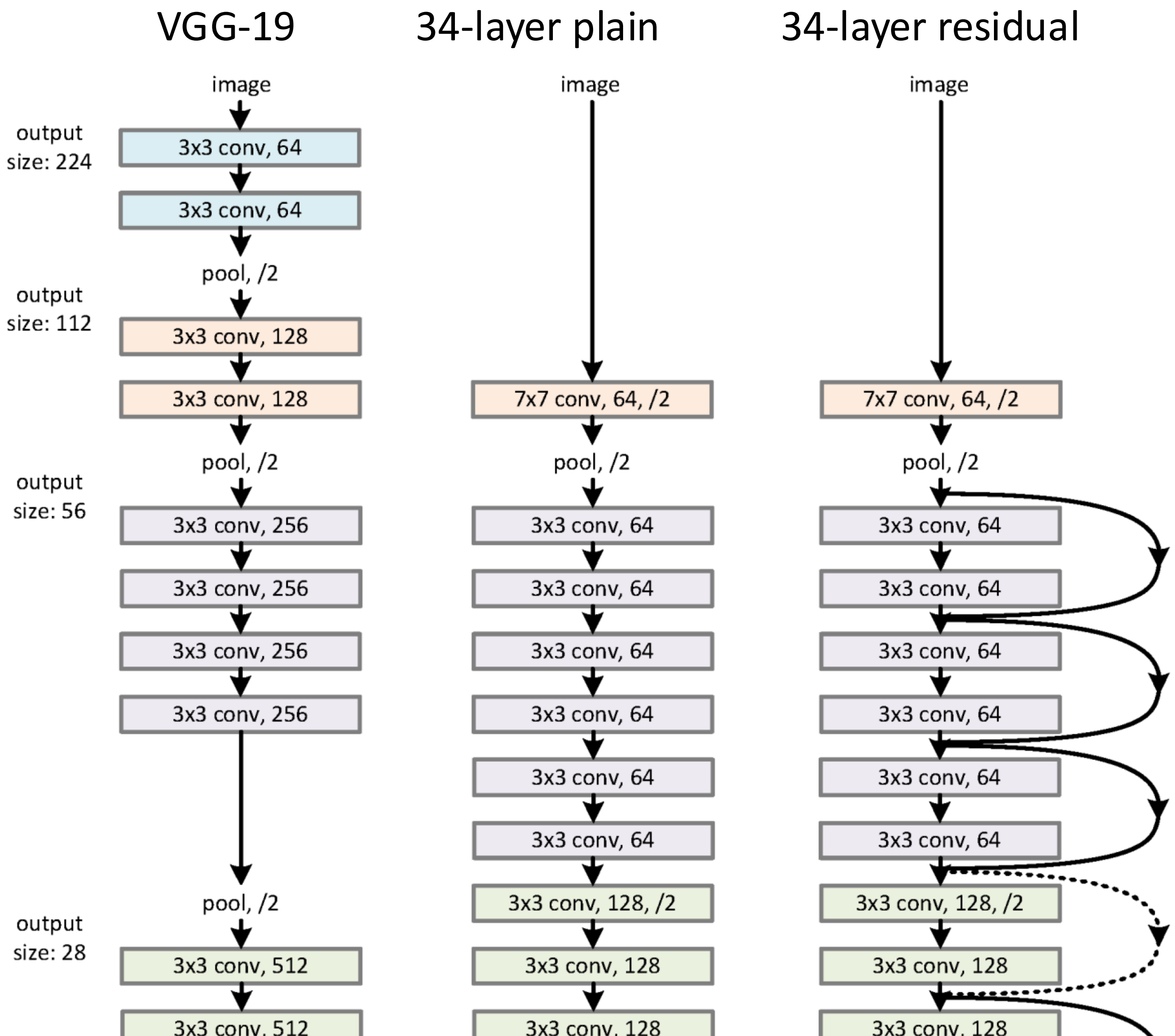


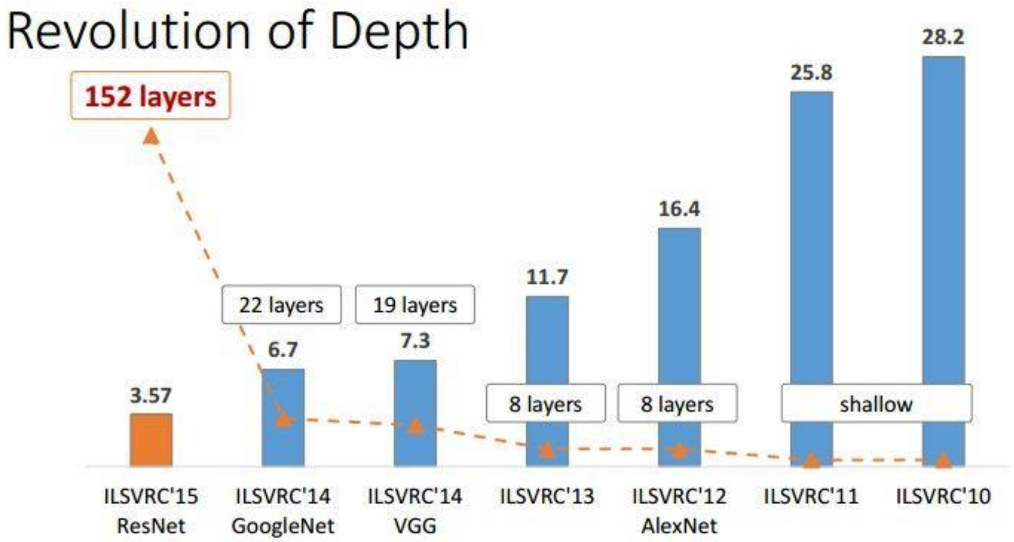
Figure 2. Residual learning: a building block

Kaiming He, et al, Deep Residual Learning for Image Recognition

2012: AlexNet (+ ImageNet + GPUs) opened doors

Kaiming He, et al, Deep Residual Learning for Image Recognition





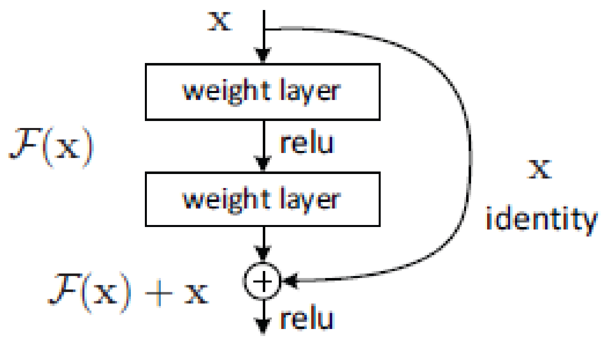


Figure 2. Residual learning: a building block

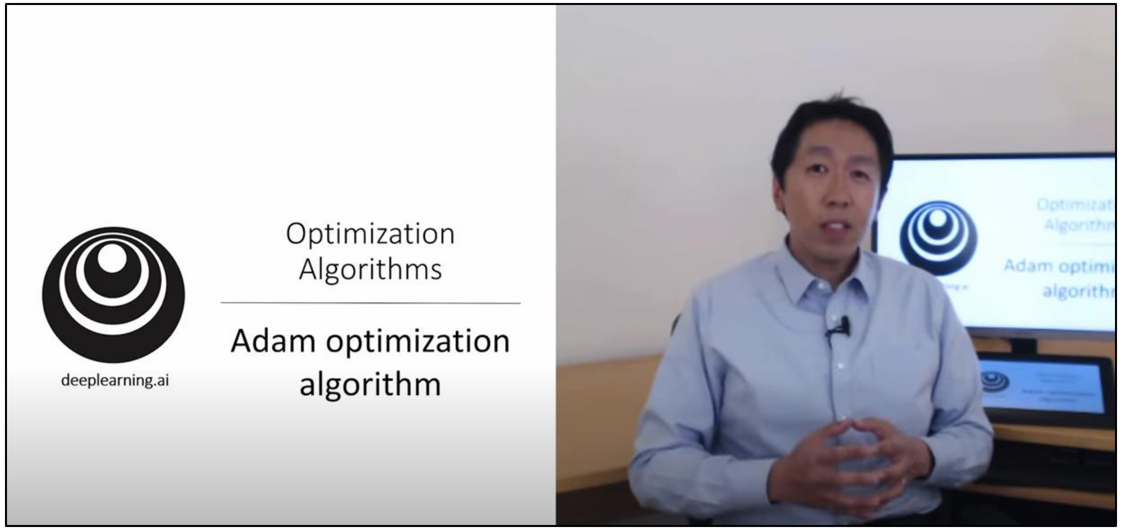
Practical Deep Learning

Deep learning issues and Fixes

Issues

- Numerical Stability
- Vanishing/Exploding gradients
- Overfitting

[Andrew Ng videos on Hyperparameter Tuning!]



Fixes

- Input normalization
- Weight initialization
- Batch normalization
- Adam
  - Exponential moving average
  - Momentum
  - RMSprop
- Learning rate decay
- Skip connections
- Dropout

CNN Summary

Convolution and Pooling Layers

Maintain spatial context across first layer

- The output of a convolution layer is still an image
- Can represent the location of the detected features

Leverage spatially invariance

- The same kernel moves across the image to detect features regardless of where they are in the image

Fewer weights

- Avoids having to learn specific weights to detect features in many, many different locations in the image

Downsampling

- Convolution/pooling layers with stride > 1 will reduce the total number of values going forward, easing the next layer learning

Convolution and Pooling Layers

Hyperparameters

- Kernel size: Defines the $K \times K$ (usually square) shape of the kernel/weight matrix. Note: There will also be a channel dimesion to the kernel
- Output channels: Similar to number of neurons in a linear layer, allows different features to be learned in each output channel. There is one $C_{in} \times K \times K$ kernel for each output channel, where K is the kernel size and C_{in} is the number of input channels
- Bias term or not: Just like linear layers, we can include a bias term or not. If including a bias term for a convolutional layer, there is one bias parameter for each output channel

Parameters

- The contents of every entry in a kernel is a weight parameter. So the total number of parameters for one 2D convolution layer is $C_{out} \times (C_{in} \times K \times K)$ plus C_{out} bias parameters if those are included.

Convolution and Pooling Layers

Hyperparameters (cont.)

- Stride: How much to shift the kernel over while performing the convolution (usually the same in each direction). A stride of 1 will output and image that is roughly the same size as the input (see padding), while a stride of 2 will produce an image that is 2x smaller in both width and height
- Padding: Effectively add rows/columns around the input image to deal with (literal) edge cases better. Often used to keep the exact same size output image as the input (with stride=1)
- Pooling: Average or Max convolutions are just operations to combine pixels and don't have any parameters. Max pooling is more common over average pooling

Convolution and Pooling Layers

Forward and backward passes

- Math: Convolutions are just multiplications and additions with a bunch of for-loops, just like the matrix multiplication in linear layers. In fact, convolutions *are* linear layers; they just apply the linear operations to different combinations of input values and weights. Thus, the calculus for backpropagation is essentially the same as a linear layer, just with some reshaping of the data.
- Computation: Just like linear layers convolutions and pooling layers have a ton of embarrassingly parallel multiplication and addition operations across lots of data. GPUs and more recently TPUs are designed for these operations, making these hardware components essentially for neural net training

Vision / Computer Vision History

LeNet 5, 1998

Input: 1, 32, 32

```
nn.Conv2d(out_channels=6, kernel_size=5),
        nn.Tanh(),
nn.AvgPool2d(kernel_size=2, stride=2),
nn.Conv2d(out_channels=16, kernel_size=5),
        nn.Tanh(),
nn.AvgPool2d(kernel_size=2, stride=2)

nn.Linear(out_features=120),
        nn.Tanh(),
nn.Linear(out_features=84),
        nn.Tanh(),
nn.Linear(out_features=10)
```

Network changes (other than bigger, deeper)

- tanh → ReLU
- Avg Pool → Max Pool
- + Batch Normalization (keep values in reasonable range)
- + Dropout (form of regularization)

AlexNet, 2012

Input: 3, 224, 224

```
nn.Conv2d(channels=96, kernel_size=11, stride=4),
        nn.BatchNorm(),   nn.ReLU(),
nn.MaxPool2d(kernel_size=3, stride=2),
nn.Conv2d(channels=256, kernel_size=5),
        nn.BatchNorm(256),   nn.ReLU(),
nn.MaxPool2d(kernel_size=3, stride=2),
nn.Conv2d(channels=384, kernel_size=3),
        nn.BatchNorm(384),   nn.ReLU(),
nn.Conv2d(channels=384, kernel_size=3),
        nn.BatchNorm(384),   nn.ReLU(),
nn.Conv2d(channels=256, kernel_size=3),
        nn.BatchNorm(256),   nn.ReLU(),
nn.MaxPool2d(kernel_size=3, stride=2),
        nn.Dropout(0.5),
nn.Linear(channels=4096), nn.ReLU(),
        nn.Dropout(0.5),
nn.Linear(channels=4096), nn.ReLU(),
nn.Linear(channels=1000)
```

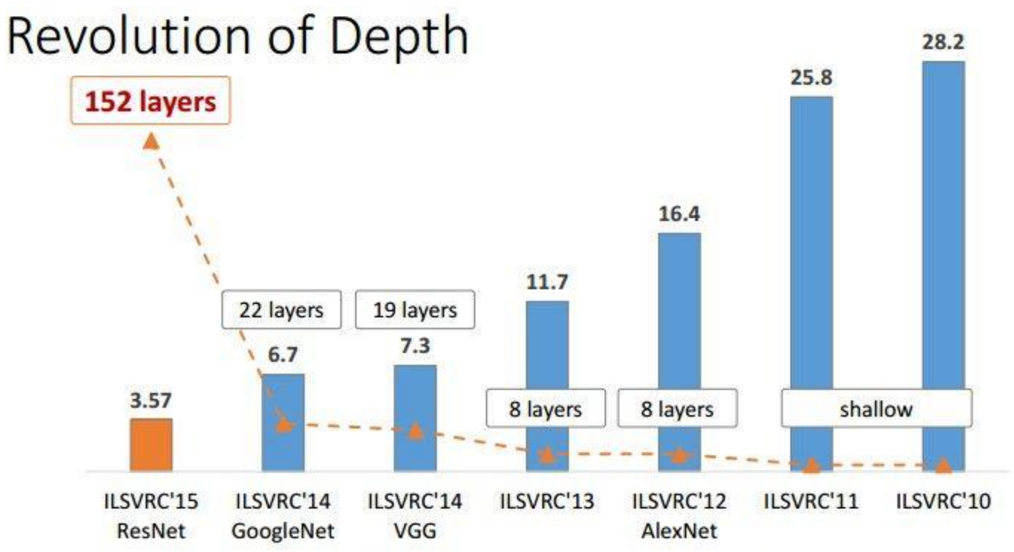
2012: AlexNet (+ ImageNet + GPUs) opened doors

Key

- Network + Data + Compute

Additional innovations

- Deep learning toolkits
  - (Caffe, Torch) → (PyTorch, Tensorflow)
  - "Model zoos"
- Residual connections (ResNet)
  - (aka skip connections)
  - Shortcuts for information flow (forward and backward)



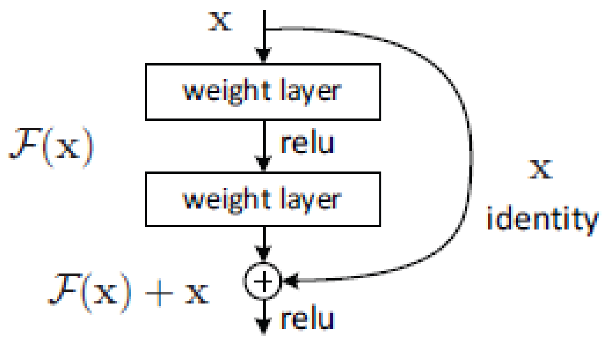


Figure 2. Residual learning: a building block

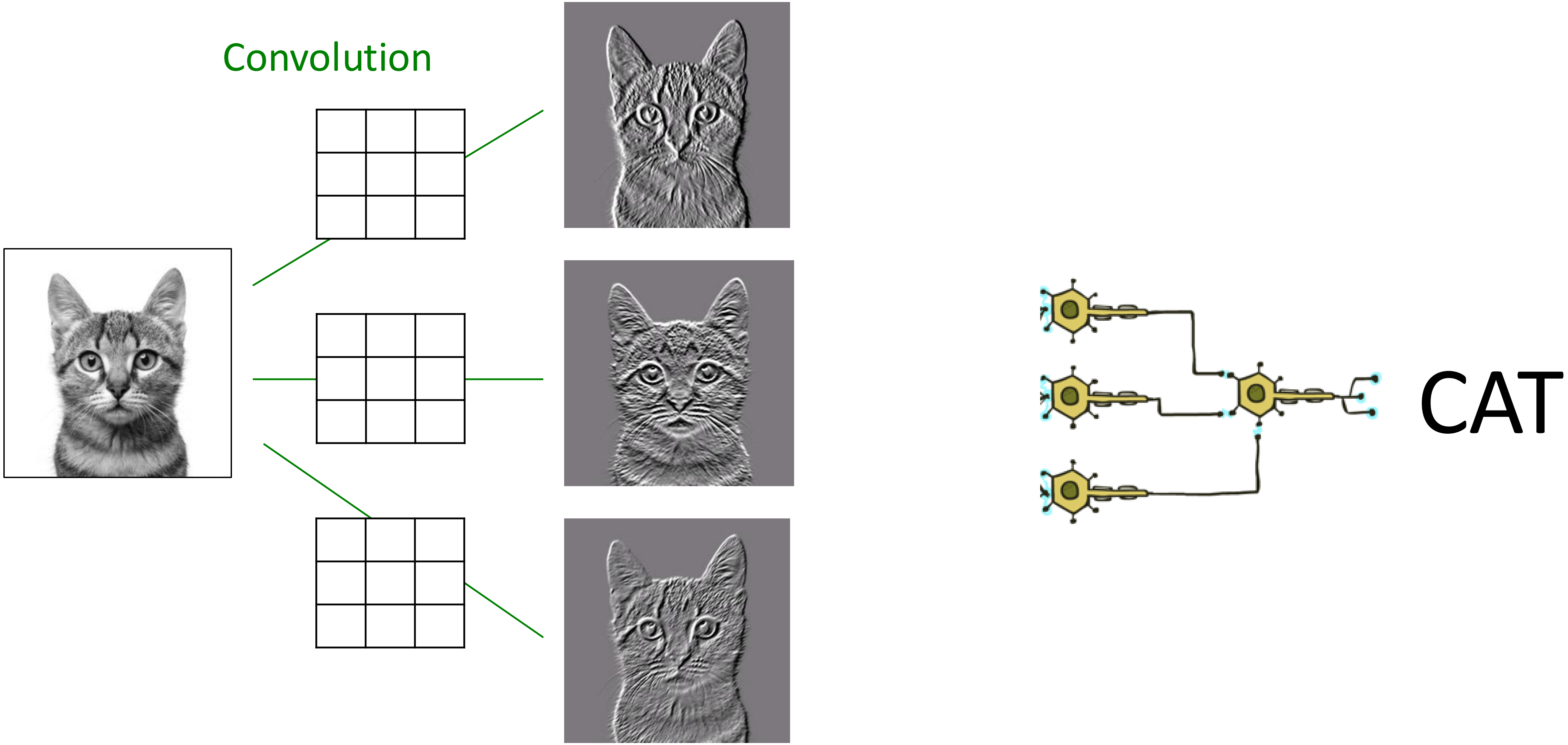
Kaiming He, et al, Deep Residual Learning for Image Recognition

Additional Slides

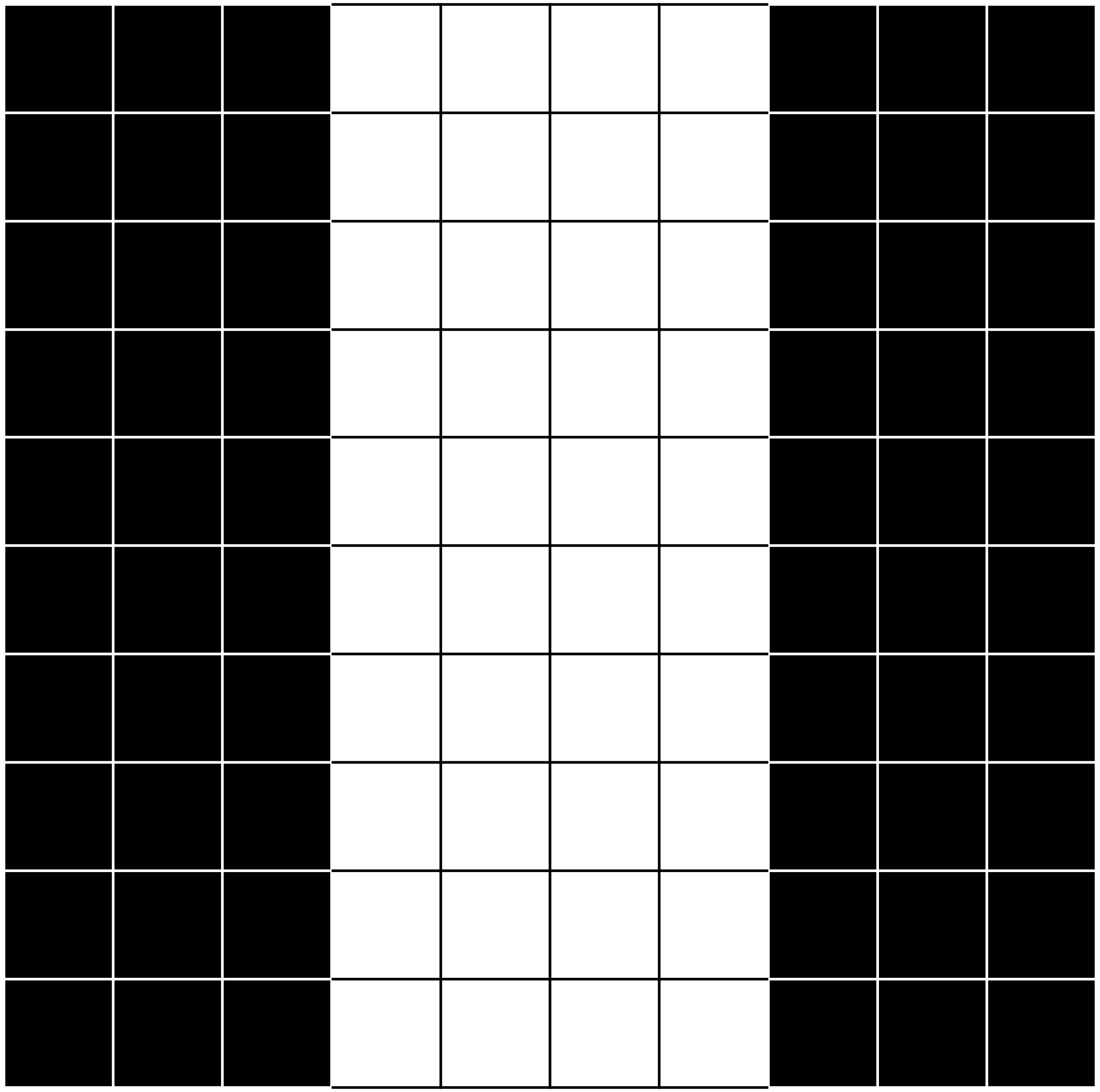
Pre-reading

Convolutions and CNNs

Convolutional Neural Networks



Convolution



-1	0	1
-1	0	1
-1	0	1

Convolution

0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0

-1	0	1
-1	0	1
-1	0	1

Convolution

0_{-1}	0_{0}	0_{1}	1	1	1	1	0	0	0
0_{-1}	0_{0}	0_{1}	1	1	1	1	0	0	0
0_{-1}	0_{0}	0_{1}	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0

0							

-1	0	1
-1	0	1
-1	0	1

Convolution

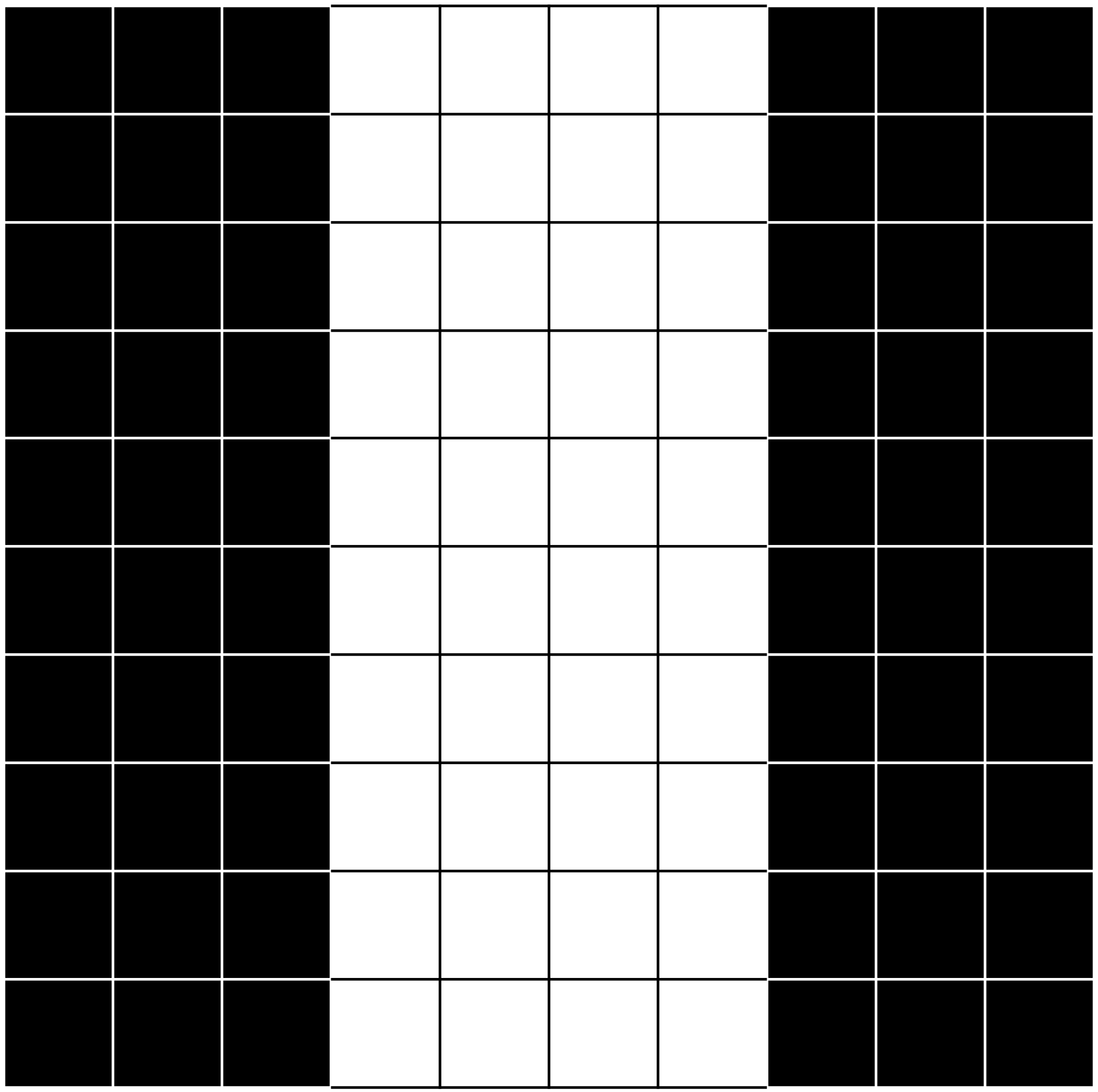
-1	0	1
-1	0	1
-1	0	1

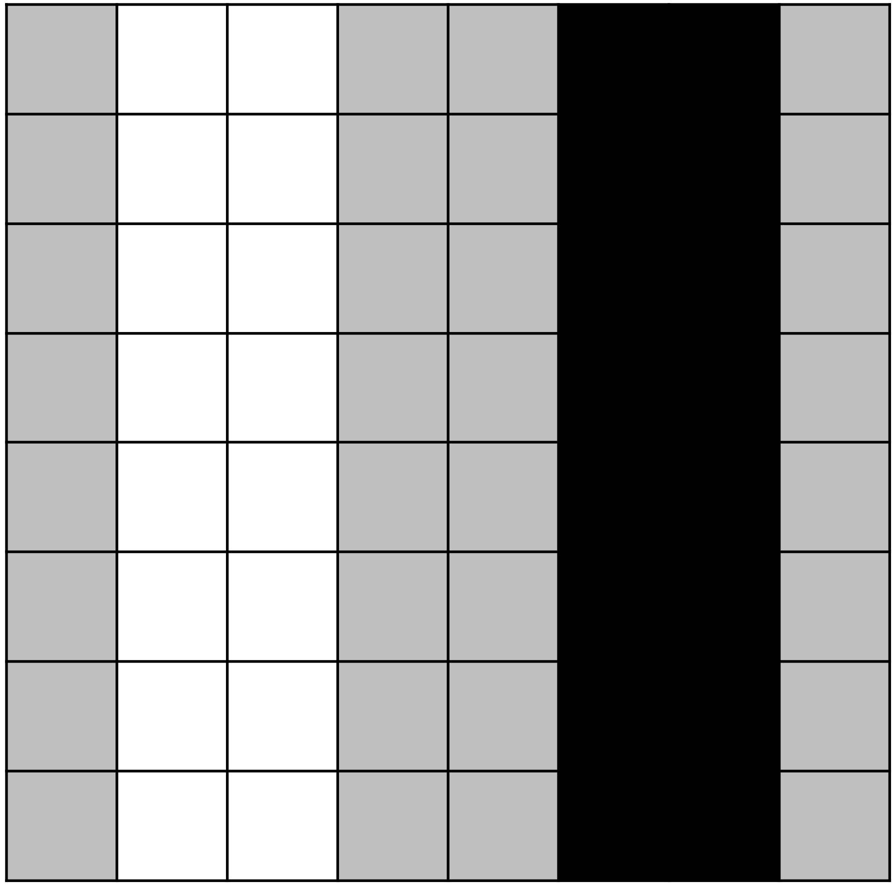
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0_{-1}	0_{0}	0_{1}
0	0	0	1	1	1	1	0_{-1}	0_{0}	0_{1}
0	0	0	1	1	1	1	0_{-1}	0_{0}	0_{1}

0	3	3	0	0	-3	-3	0
0	3	3	0	0	-3	-3	0
0	3	3	0	0	-3	-3	0
0	3	3	0	0	-3	-3	0
0	3	3	0	0	-3	-3	0
0	3	3	0	0	-3	-3	0
0	3	3	0	0	-3	-3	0
0	3	3	0	0	-3	-3	0

Convolution

-1	0	1
-1	0	1
-1	0	1





Convolution

Signal processing definition

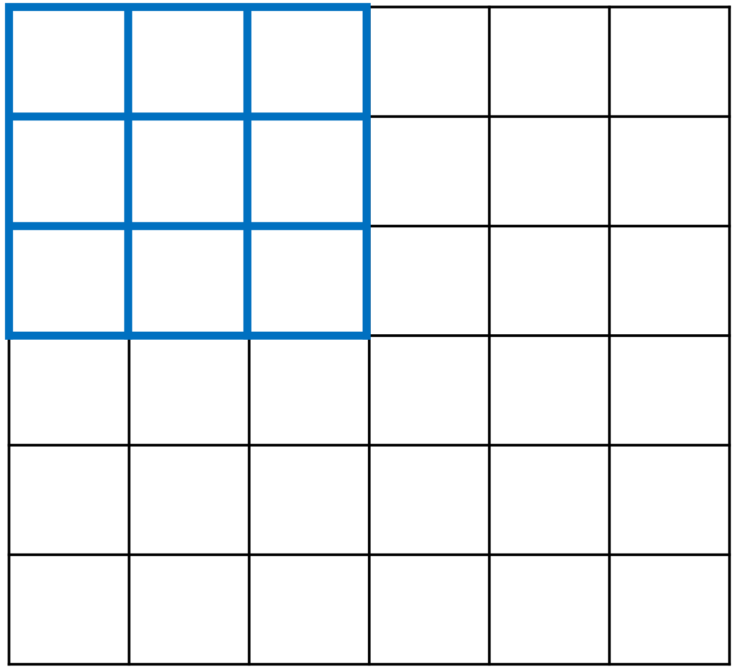
$$z[i,j] = \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} x[i-u, j-v] \cdot w[u, v]$$

-1	0	1
-2	0	2
-1	0	1

Relaxed definition

- Drop infinity; don’t flip kernel

$$z[i,j] = \sum_{u=0}^{K-1} \sum_{v=0}^{K-1} x[i+u, j+v] \cdot w[u, v]$$



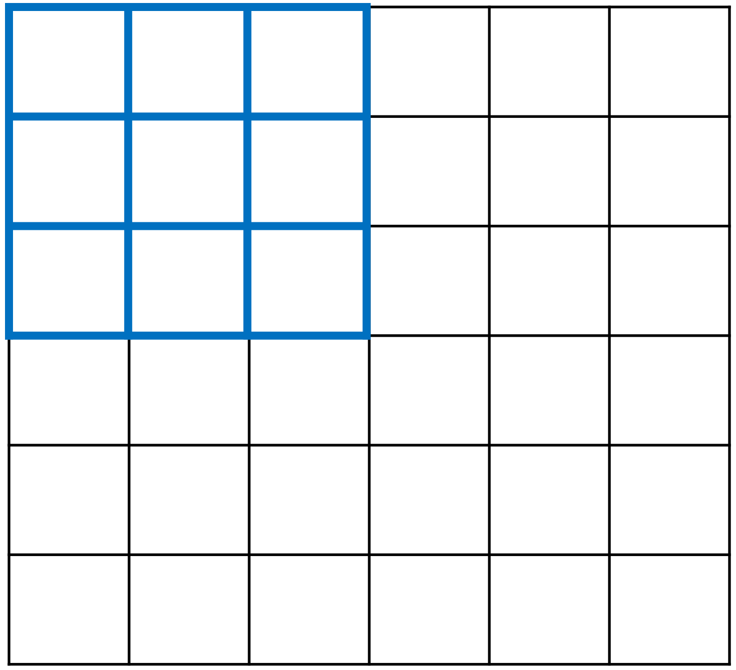
Convolution

Relaxed definition

$$z[i,j] = \sum_{u=0}^{K-1} \sum_{v=0}^{K-1} x[i+u, j+v] \cdot w[u\,, v]$$

-1	0	1
-2	0	2
-1	0	1

```
for i in range(0, im_width - K + 1):
  for j in range(0, im_height - K):
    im_out[i,j] = 0
    for u in range(0, K):
      for v in range(0, K):
        im_out[i,j] += im[i+u, j+v] * kernel[u,v]
```



GPU!!

Convolution: Padding

0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0

0	2	2	0	0	-2	-2	0
0	3	3	0	0	-3	-3	0
0	3	3	0	0	-3	-3	0
0	3	3	0	0	-3	-3	0
0	3	3	0	0	-3	-3	0
0	3	3	0	0	-3	-3	0
0	3	3	0	0	-3	-3	0
0	2	2	0	0	-2	-2	0

Exercise: Which kernel goes with which output image?

Input

K1

-1	0	1
-2	0	2
-1	0	1

K2

-1	-2	-1
0	0	0
1	2	1

K3

0	0	-1	0
0	-2	0	1
-1	0	2	0
0	1	0	0

Im1

Im2

Im3

Exercise: Which kernel goes with which output image?

Input

K1

-1	0	1
-2	0	2
-1	0	1

K2

-1	-2	-1
0	0	0
1	2	1

K3

0	0	-1	0
0	-2	0	1
-1	0	2	0
0	1	0	0

Im1

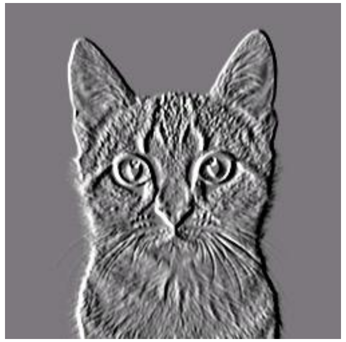
Im2

Im3

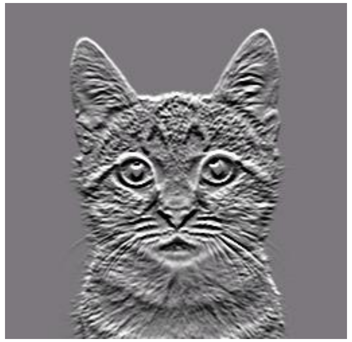
Convolutional Neural Networks

Convolution

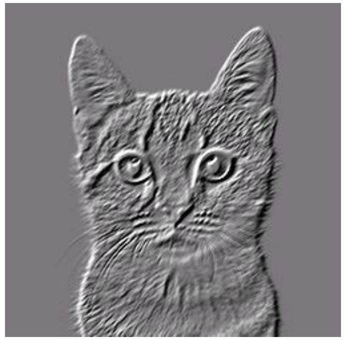
-1	0	1
-2	0	2
-1	0	1

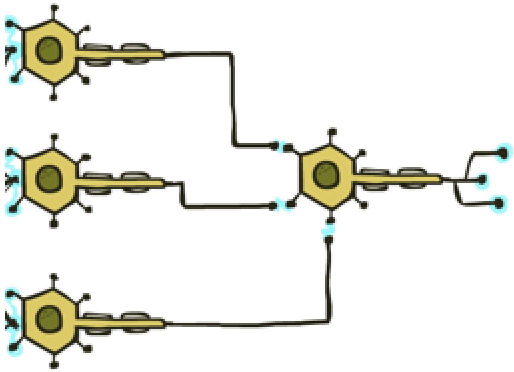


-1	-2	-1
0	0	0
1	2	1

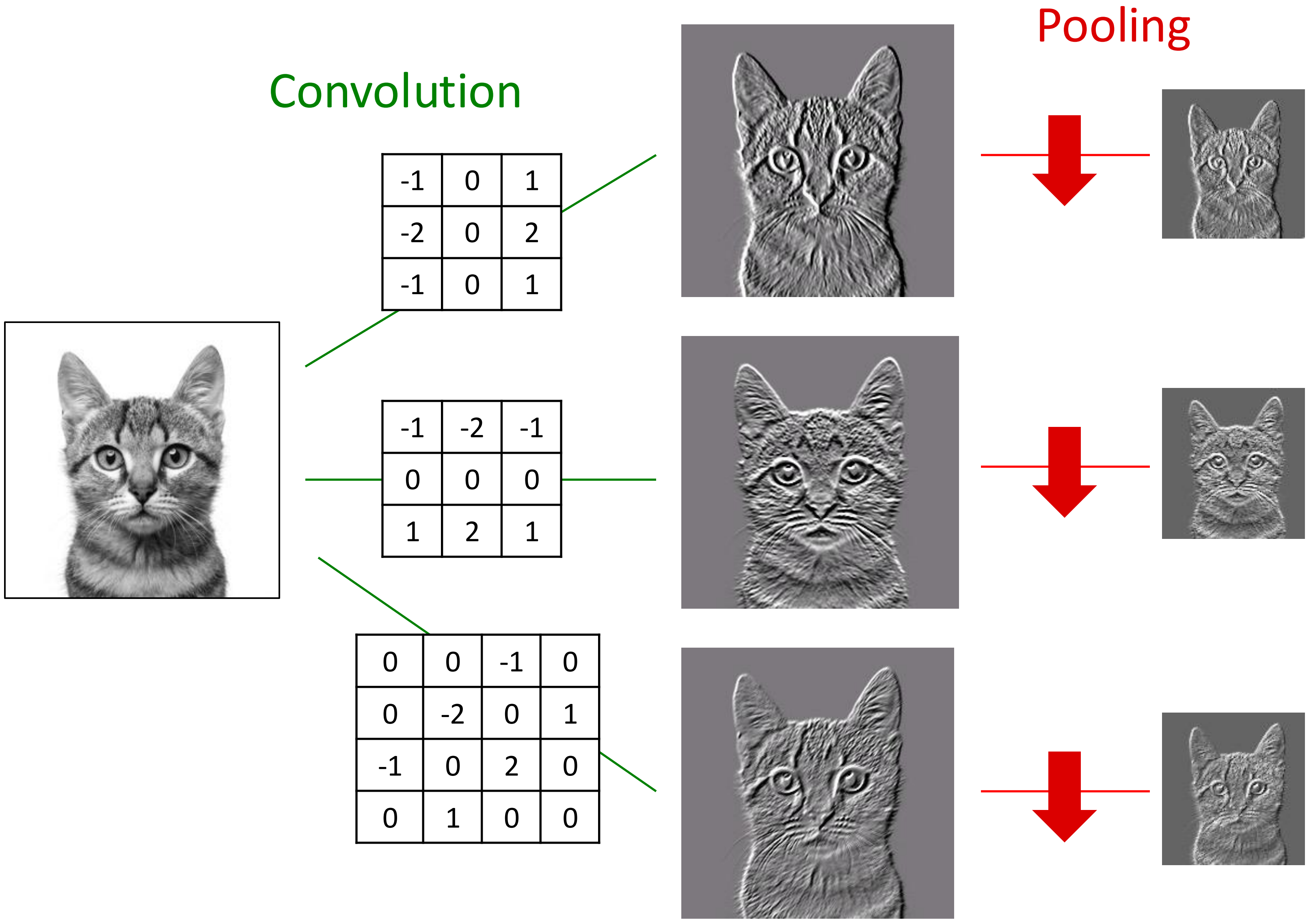


0	0	-1	0
0	-2	0	1
-1	0	2	0
0	1	0	0





Convolutional Neural Networks

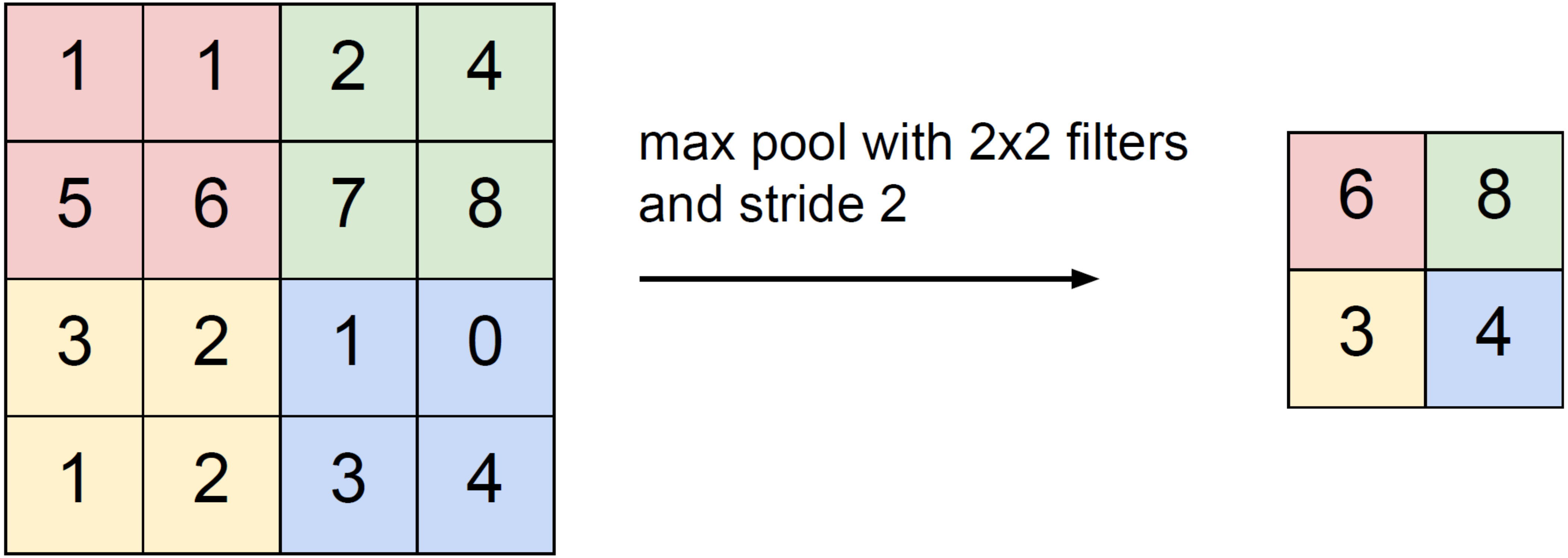


Convolution: Stride=2

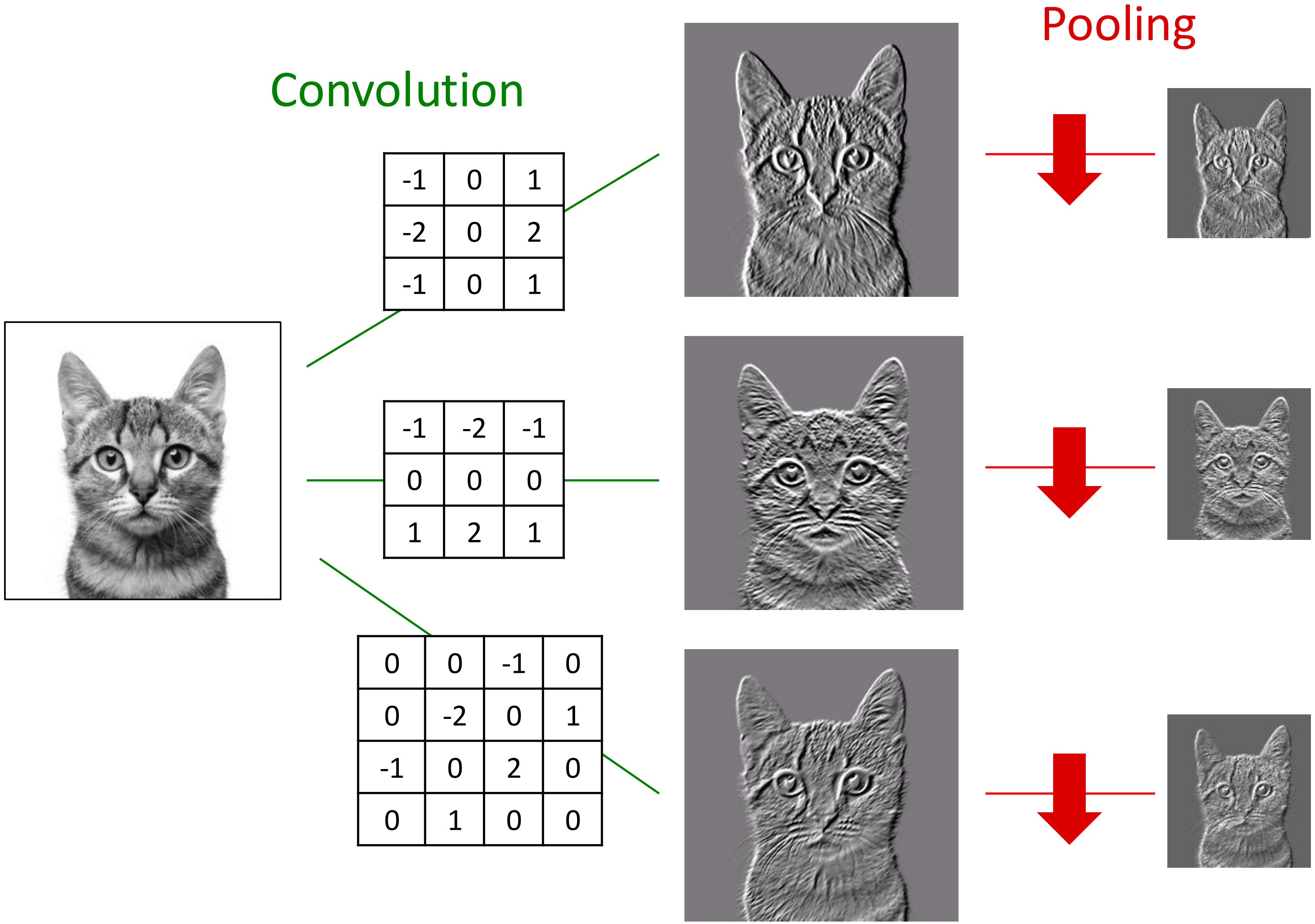
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0

.25	.25
.25	.25

Stride: Max Pooling

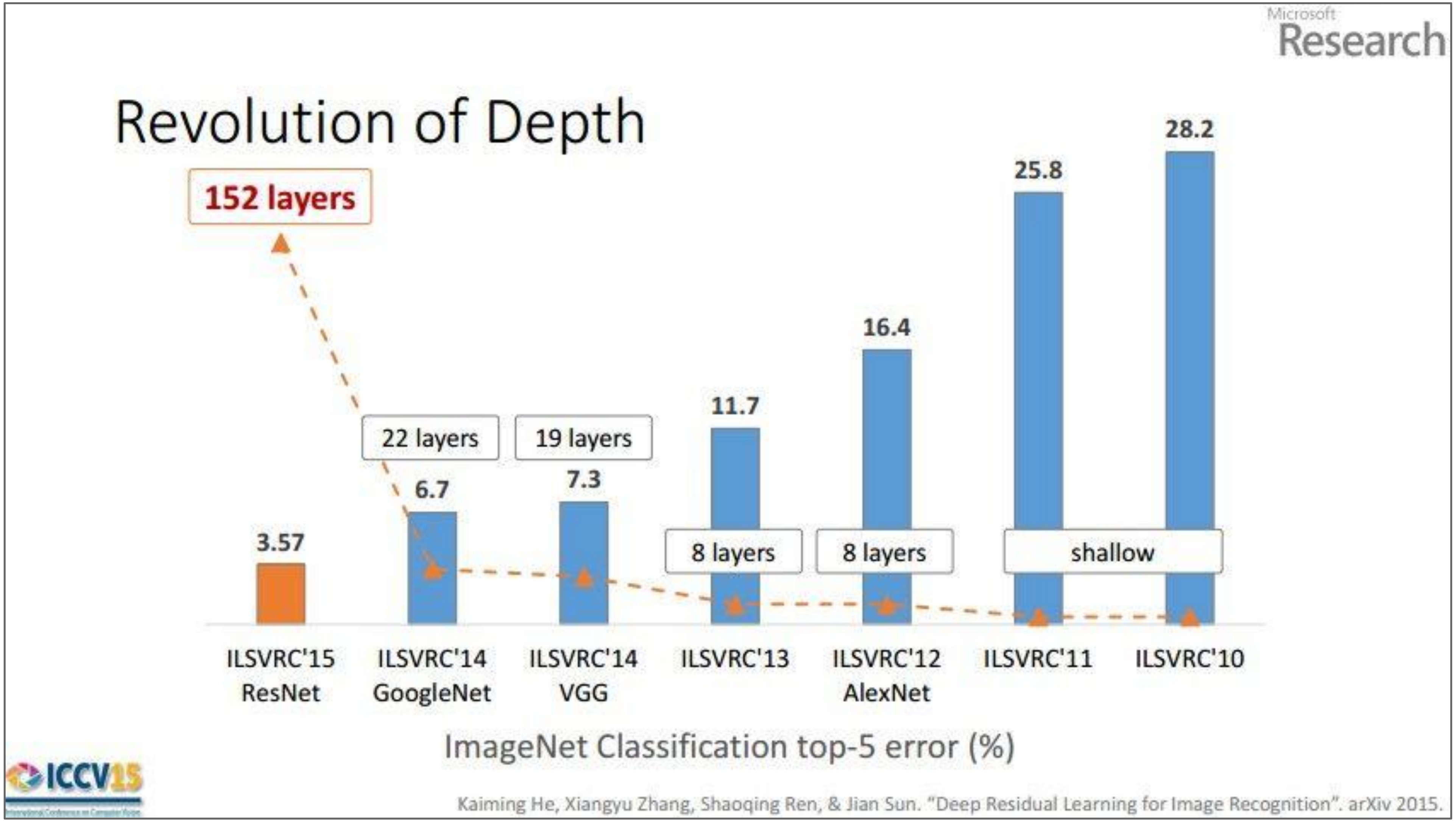


Convolutional Neural Networks



Computer Vision History

CNNs for Image Classification



Slide from Kaiming He

Which neural network are they talking about?

From [Wikipedia]:

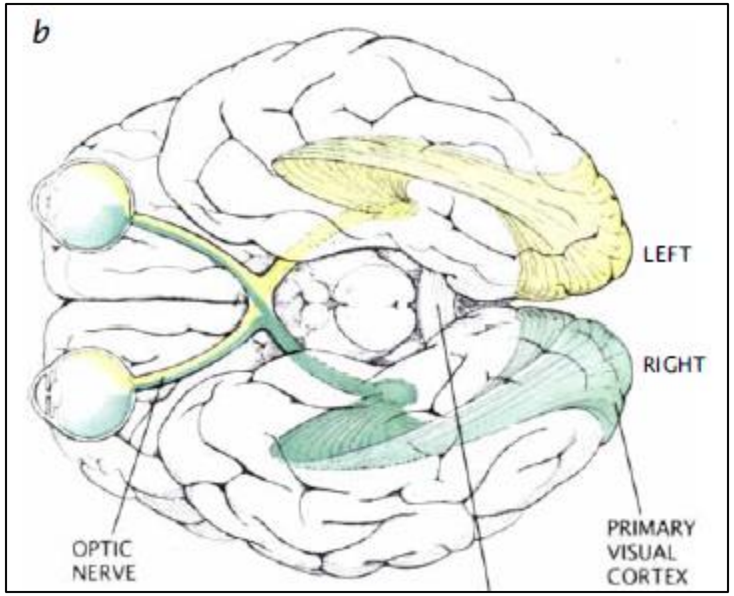
V2 receives strong feedforward connections from V1 and sends robust connections to V3, V4, and V5. Additionally, it plays a crucial role in the integration and processing of visual information.

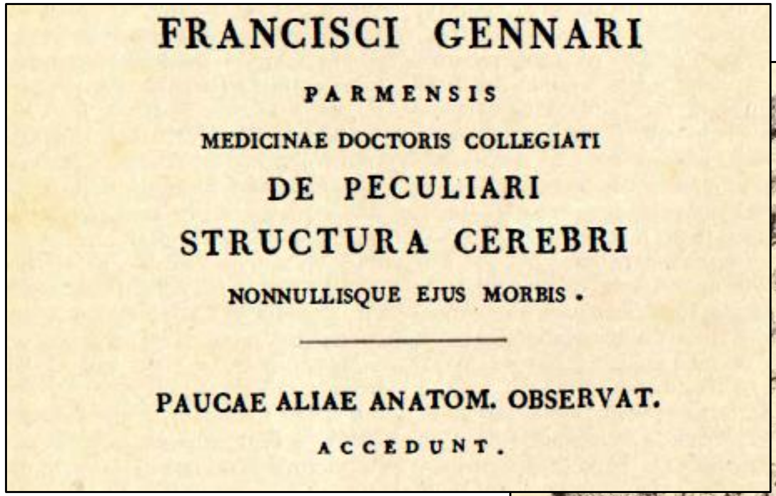
The feedforward connections from V1 to V2 contribute to the hierarchical processing of visual stimuli. V2 neurons build upon the basic features detected in V1, extracting more complex visual attributes such as texture, depth, and color. This hierarchical processing is essential for the construction of a more nuanced and detailed representation of the visual scene.

Vision / Computer Vision History

A few highlights ☺

- 1782: Documenting the (human) visual cortex




FRANCISCI GENNARI

PARMENSIS

MEDICINAE DOCTORIS COLLEGIATI

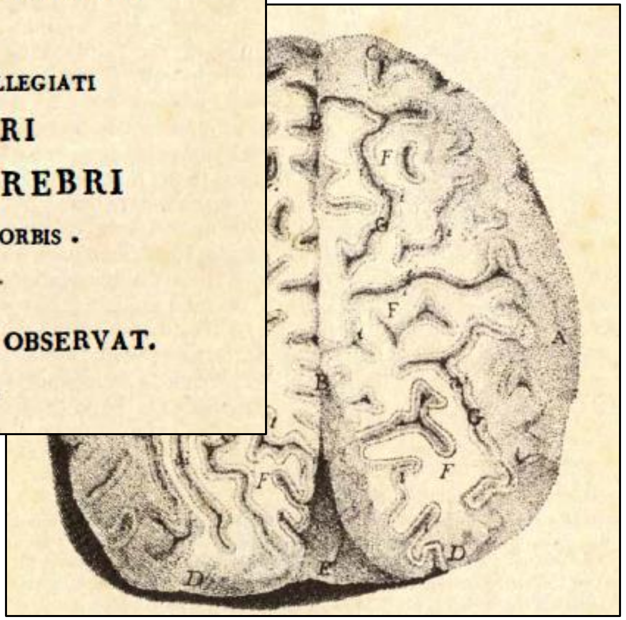
DE PECULIARI

STRUCTURA CEREBRI

NONNULLISQUE EJUS MORBIS.

PAUCAE ALIAE ANATOM. OBSERVAT.

ACCEDUNT.

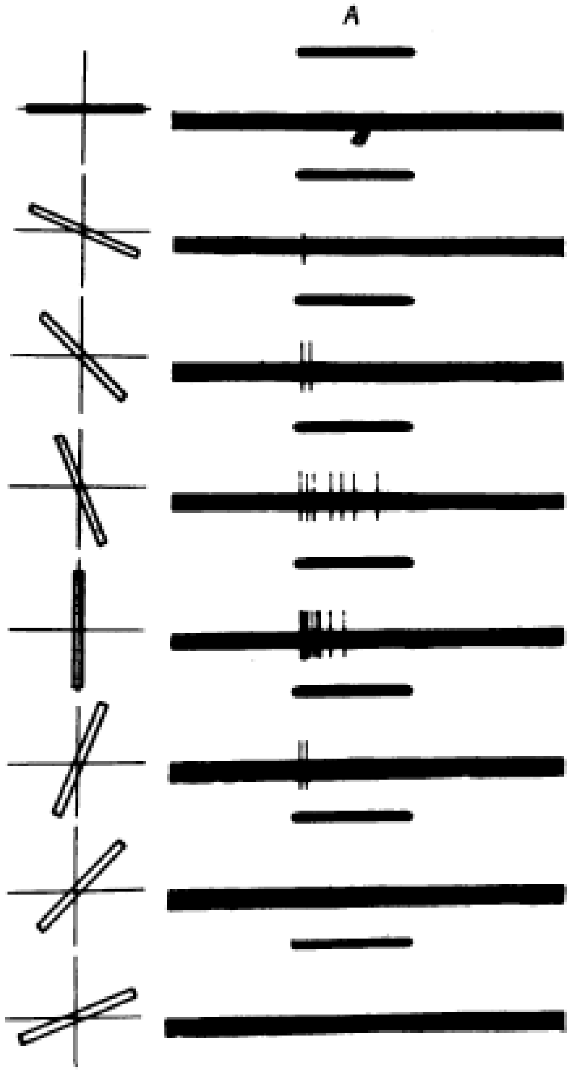


Mitchell Glickstein, The Discovery of the Visual Cortex. 1998

Vision / Computer Vision History

A few highlights ☺

- 1782: Documenting the (human) visual cortex
- 1959: V1 (cat) sensitive to edge orientation

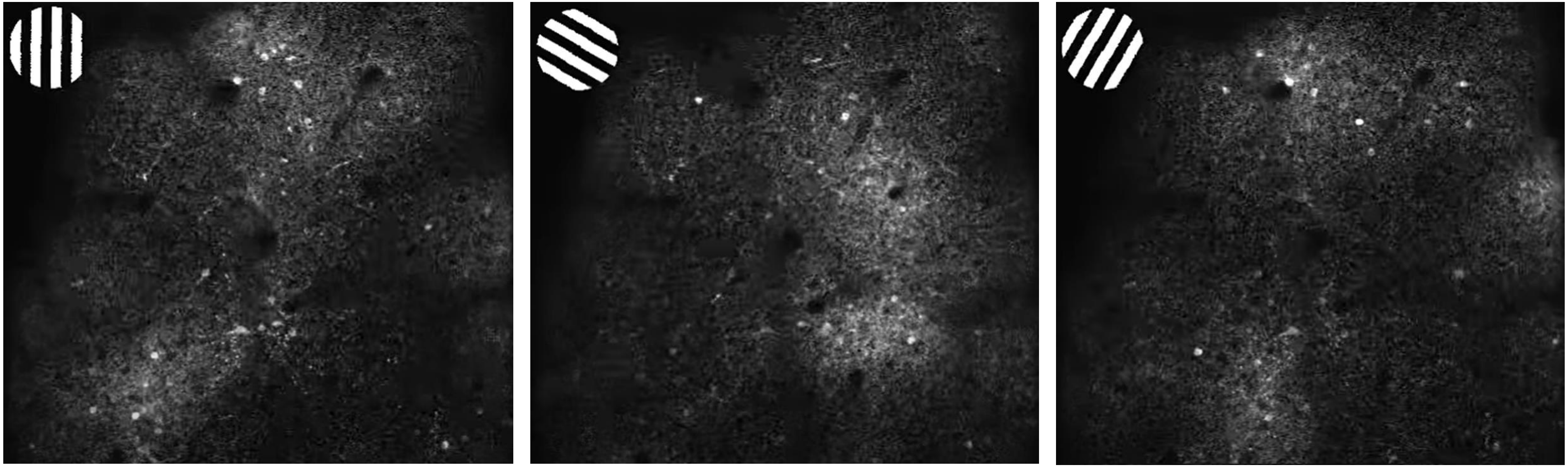


Hubel and Wiesel, RECEPTIVE FIELDS OF SINGLE NEURONES IN THE CAT'S STRIATE CORTEX. 1959

Vision / Computer Vision History

A few highlights ☺

- 1782: Documenting the (human) visual cortex
- 1959: V1 (cat) sensitive to edge orientation

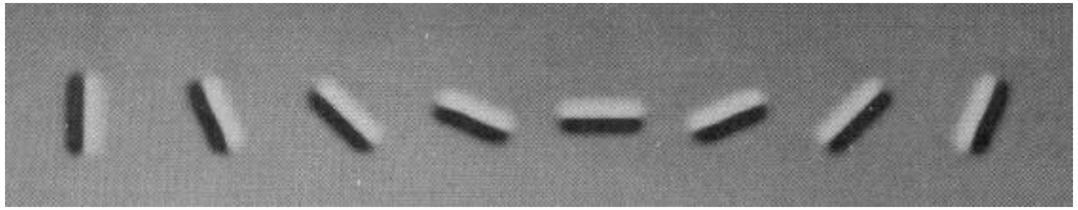


M. Li, F. Liu, H. Jiang, Tai Sing Lee, S. Tang, Long-Term Two-Photon Imaging in Awake Macaque Monkey. 2017

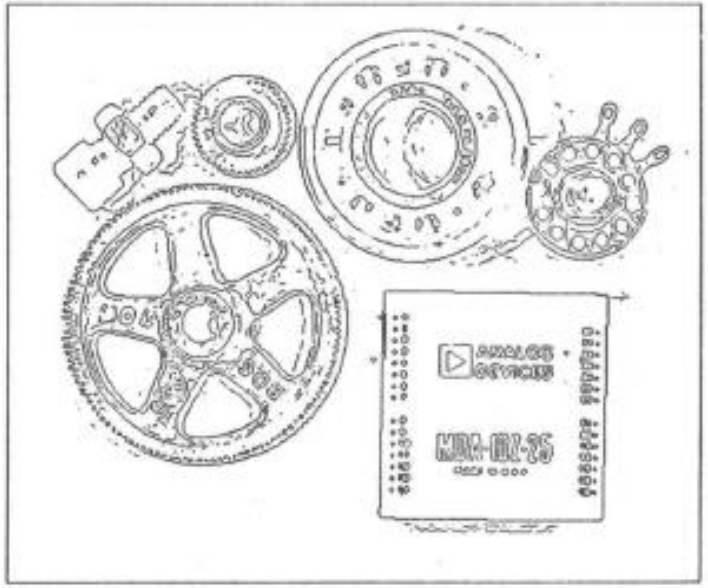
Vision / Computer Vision History

A few highlights ☺

- 1782: Documenting the (human) visual cortex
- 1959: V1 (cat) sensitive to edge orientation
- 1986: Canny edge detection



(a)



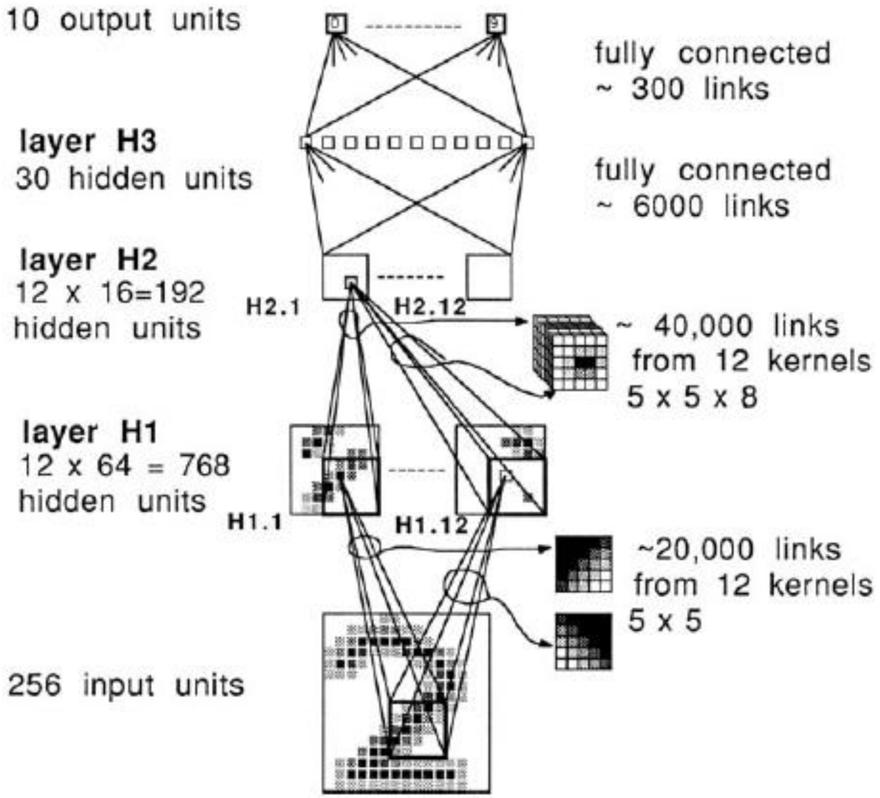
(b)

John Canny, A Computational Approach To Edge Detection. 1986

Vision / Computer Vision History

A few highlights ☺

- 1782: Documenting the (human) visual cortex
- 1959: V1 (cat) sensitive to edge orientation
- 1986: Canny edge detection
- 1989: LeNet (1) CNN



LeCun, et al, [Backpropagation Applied to Handwritten Zip Code Recognition](). 1989

Vision / Computer Vision History

A few highlights ☺

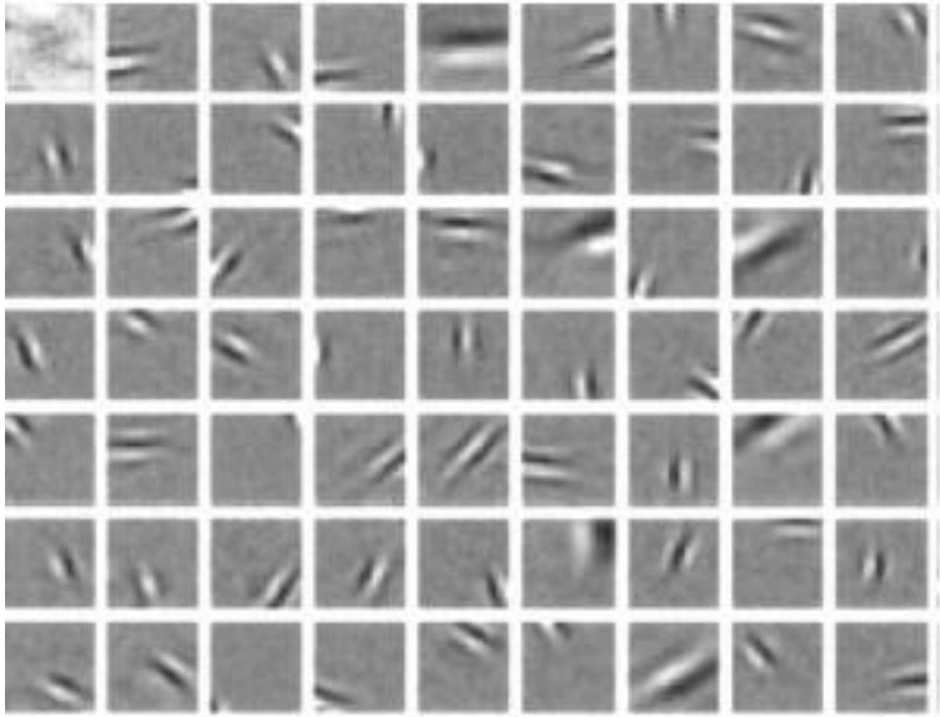
- 1782: Documenting the (human) visual cortex
- 1959: V1 (cat) sensitive to edge orientation
- 1986: Canny edge detection
- 1994: MNIST Database

https://en.wikipedia.org/wiki/MNIST_database

Vision / Computer Vision History

A few highlights ☺

- 1782: Documenting the (human) visual cortex
- 1959: V1 (cat) sensitive to edge orientation
- 1986: Canny edge detection
- 1996: Learning convolutional filters from natural images

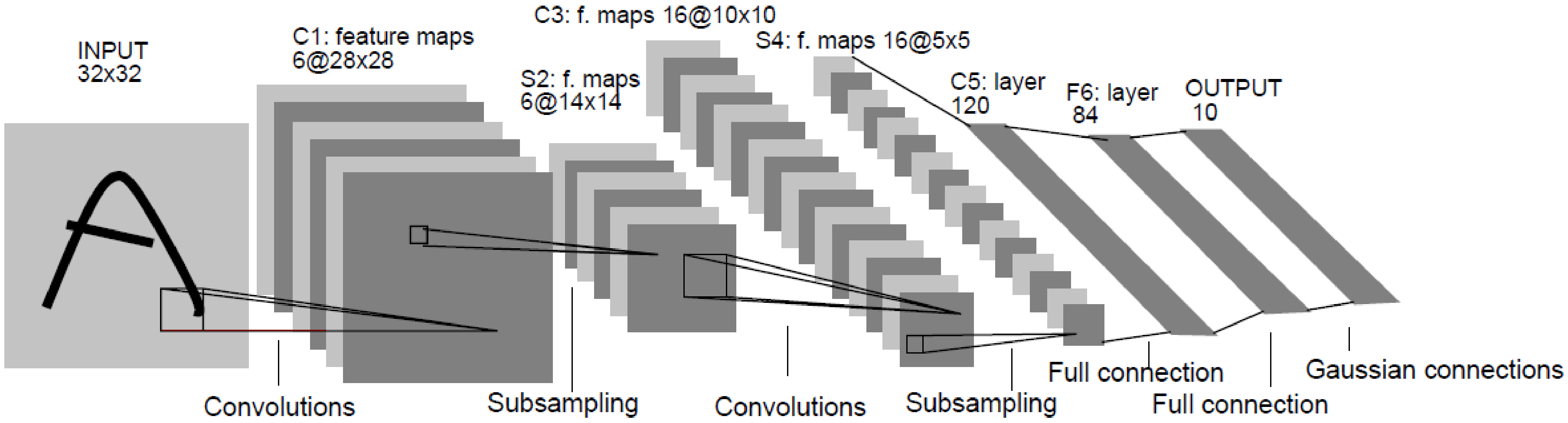


Olshausen & Field, Emergence of simple-cell receptive field properties by learning a sparse code for natural images. 1996

Vision / Computer Vision History

A few highlights ☺

- 1782: Documenting the (human) visual cortex
- 1959: V1 (cat) sensitive to edge orientation
- 1986: Canny edge detection
- 1998: LeNet-5 CNN → state-of-the-art digit classification



LeCun, Bengio, Haffner, Gradient-Based Learning Applied to Document Recognition. 1995

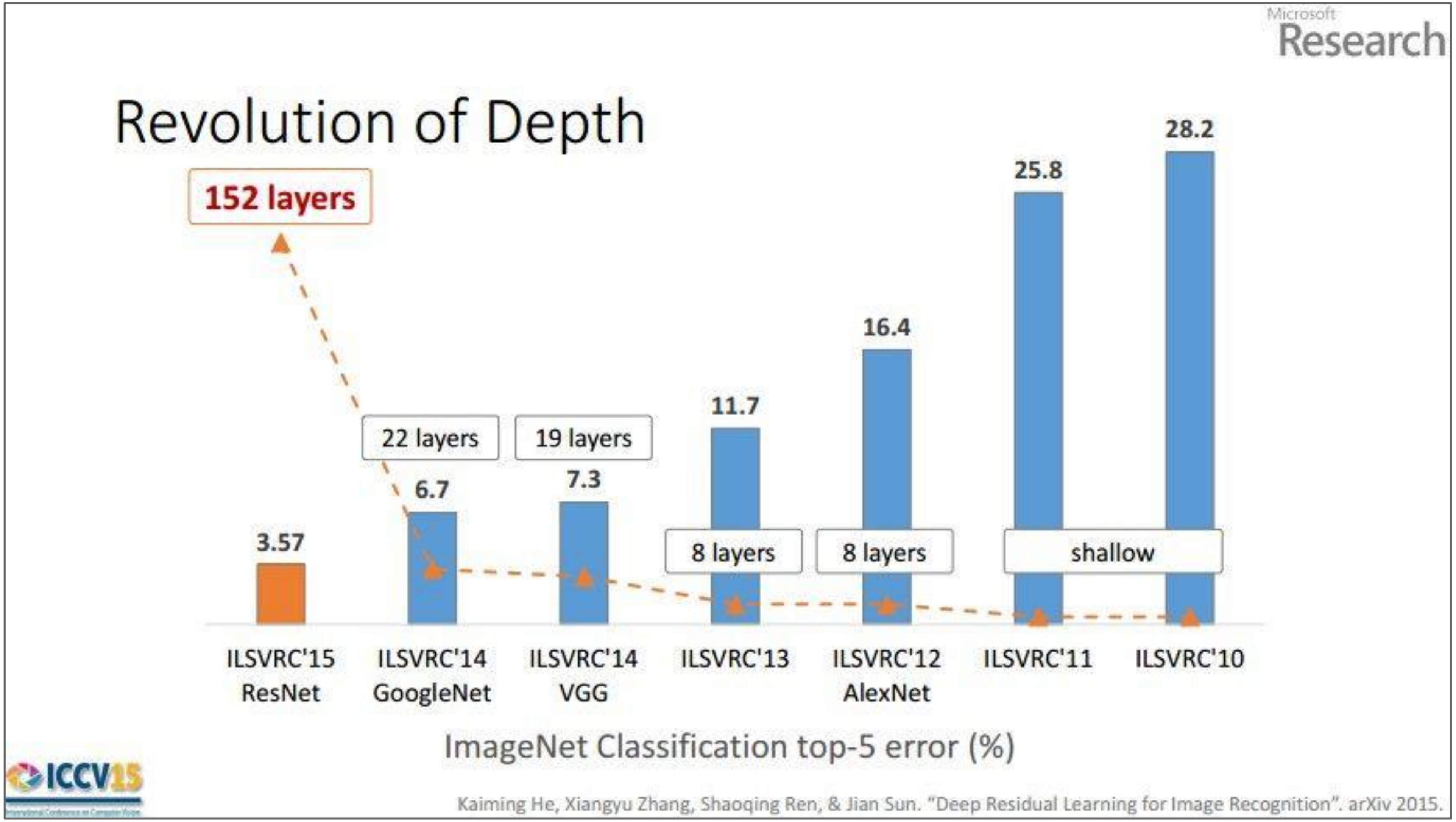
Vision / Computer Vision History

A few highlights ☺

- 1782: Documenting the (human) visual cortex
- 1959: V1 (cat) sensitive to edge orientation
- 1986: Canny edge detection
- 1998: LeNet-5 CNN → state-of-the-art digit classification
- 1998-2012: Lots of edge detection
- 2012: AlexNet CNN → state-of-the-art ImageNet classification

Alex Krizhevsky, Ilya Sutskever, Geoff Hinton, ImageNet Classification with Deep Convolutional Neural Networks. 2012

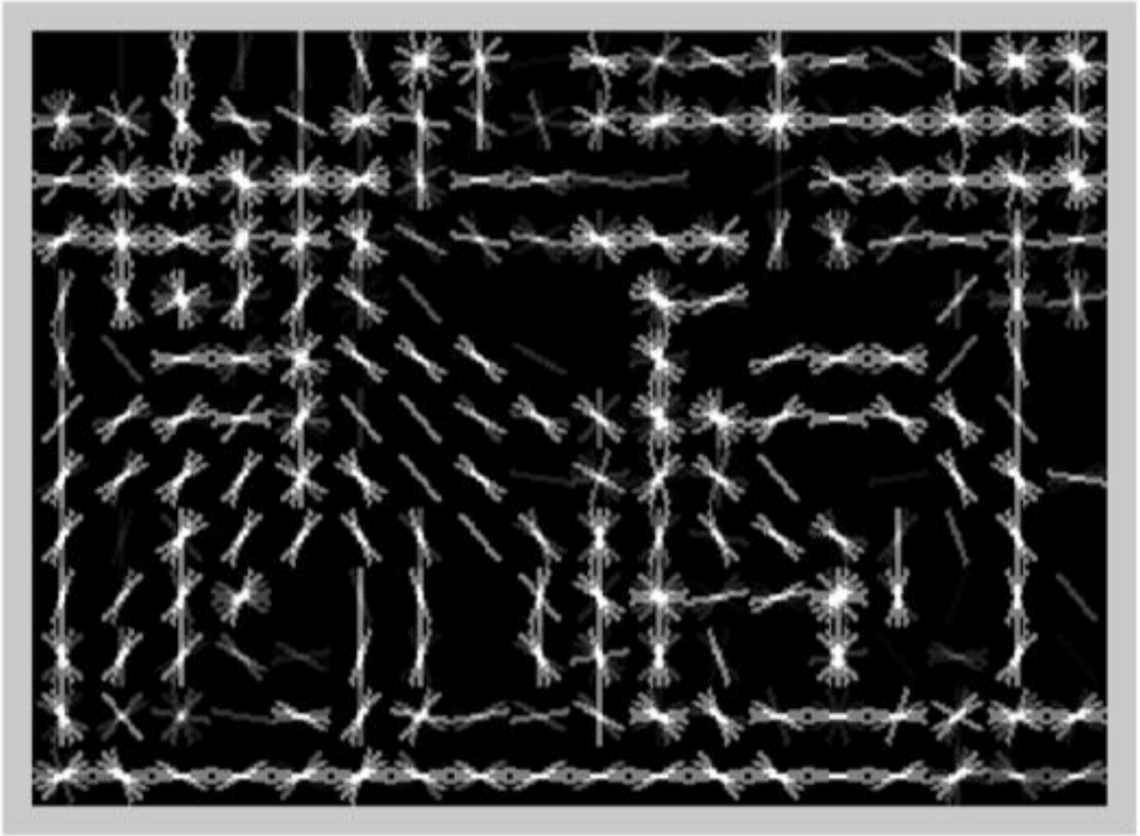
What happened in 2012?



Slide from Kaiming He

Computer Vision 1998-2012

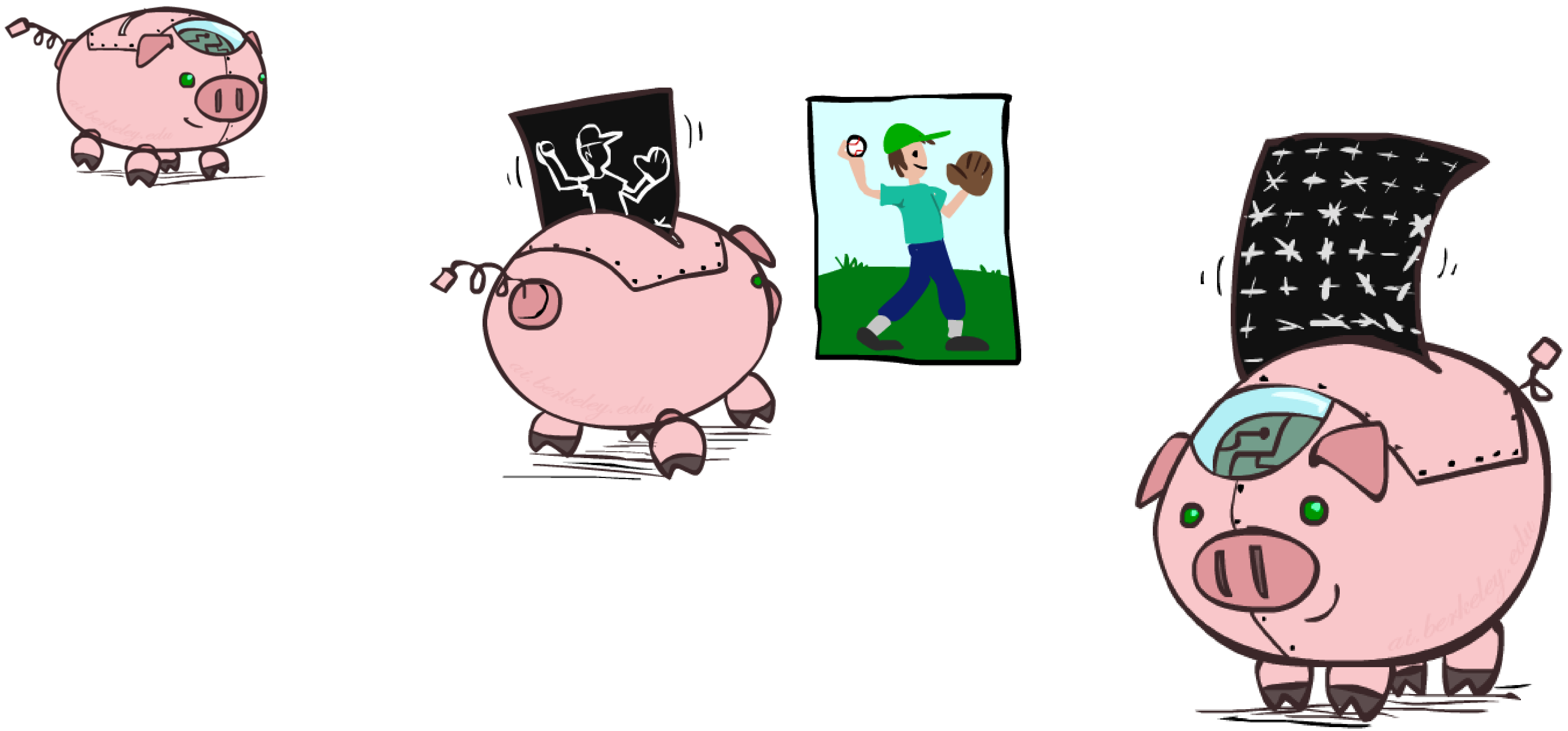
HoG: Histogram of oriented gradients



[Dalal and Triggs, 2005]

Computer Vision 1998-2012

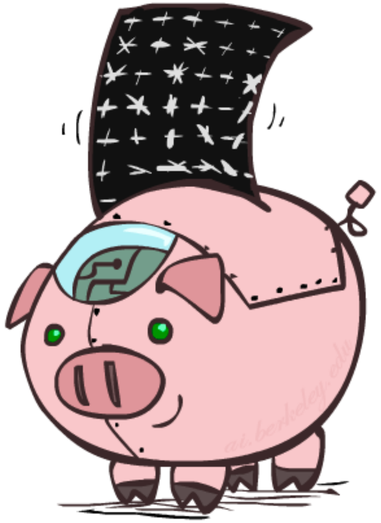
HoG: Histogram of oriented gradients

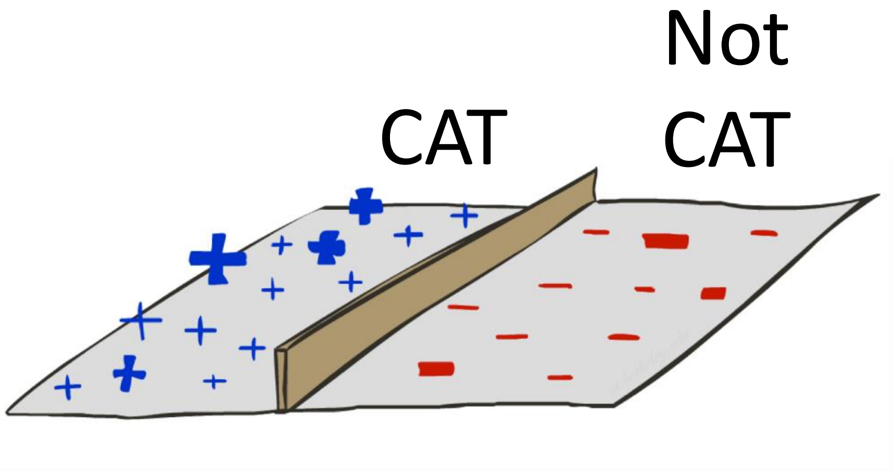


[Dalal and Triggs, 2005]

Image Classification

HOG features passed to a linear classifier (logistic regression / SVM)





CAT