南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

# Brain Intelligence and Artificial Intelligence

# 人脑智能与机器智能

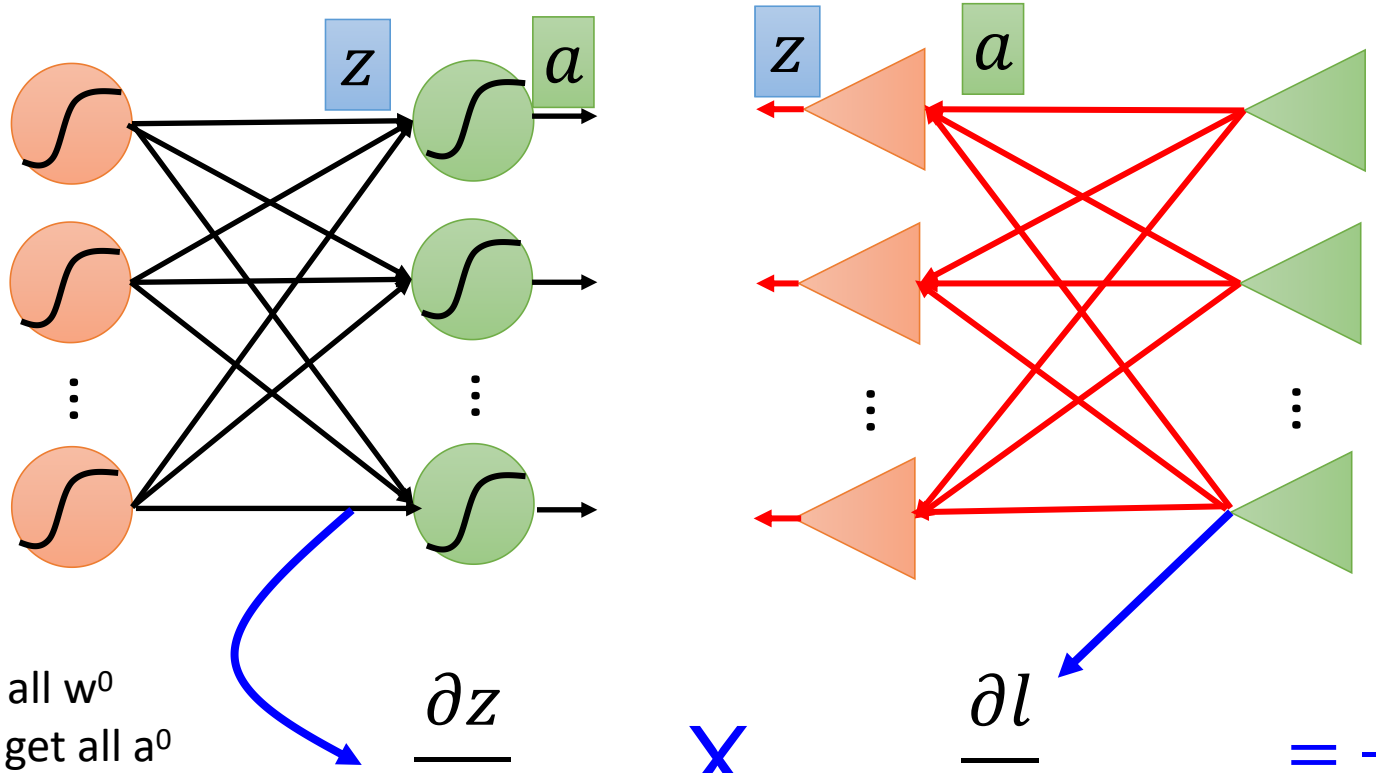
## Lecture 4 – What do neurons in AI / brain learn?

**Quanying Liu （刘泉影）**

SUSTech, BME department

Email: liuqy@sustech.edu.cn

# Recall: Error Backpropagation in NN




Last layer:

$$\frac{\partial l}{\partial z_{i,N}} = \frac{\partial l}{\partial y_{i,N}} \frac{\partial y_{i,N}}{\partial z_{i,N}}$$

Recursively go to the previous layer:

$$\frac{\partial l}{\partial z_{i,j}} = \sigma'(z_{i,j}) \left[ \sum w_{i,j+1} \frac{\partial l}{\partial z_{i,j+1}} \right]$$

- Initiate parameters all $w^0$
- Go forward pass to get all $a^0$
- $\frac{\partial z}{\partial w_{i,j}^0} = a_{i,j}^0$

$$\frac{\partial z}{\partial w} = a$$

$$\times \quad \frac{\partial l}{\partial z}$$

$$= \frac{\partial l}{\partial w}$$

for all w

# Recall CNN architecture

Convolution

Activation function

Pooling

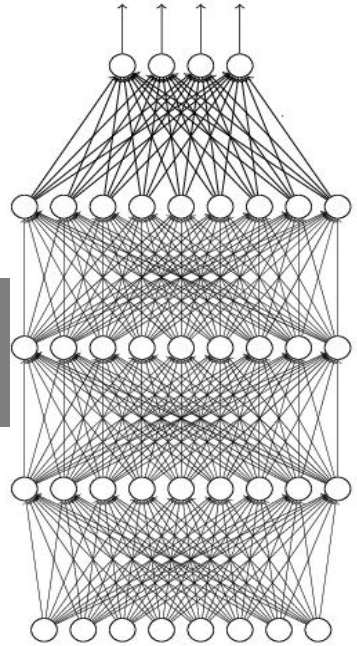
These blocks can repeat many times

Convolution

Activation function

Pooling

Flatten



cat / dog ……

Fully Connected Feedforward network

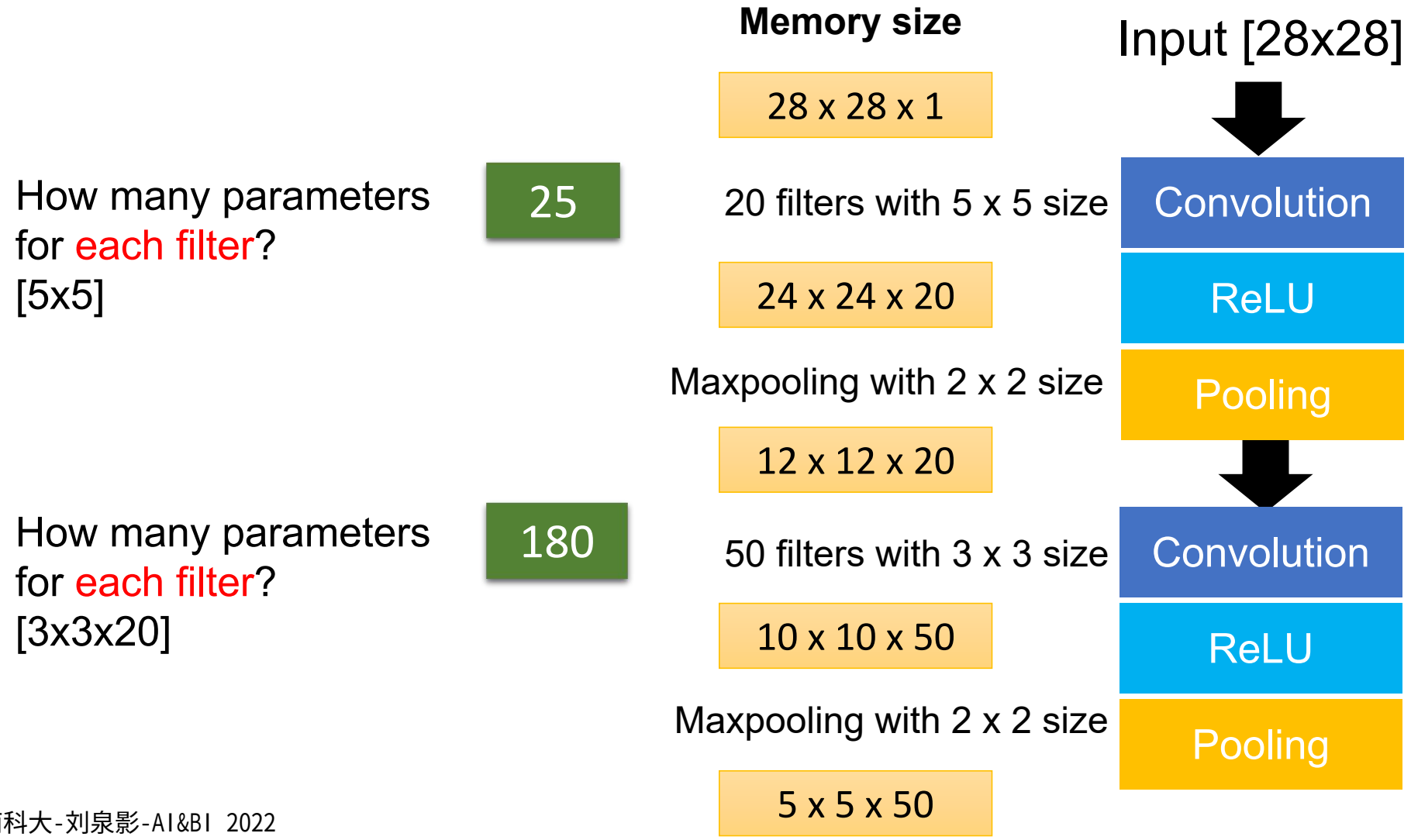
**Some Activation Functions**

1. Binary Step Function
2. Sigmoid / Logistic
3. TanH / Hyperbolic Tangent
4. Softmax
5. ReLU (Rectified Linear Unit)
6. Leaky ReLU
7. Parametric ReLU
8. …

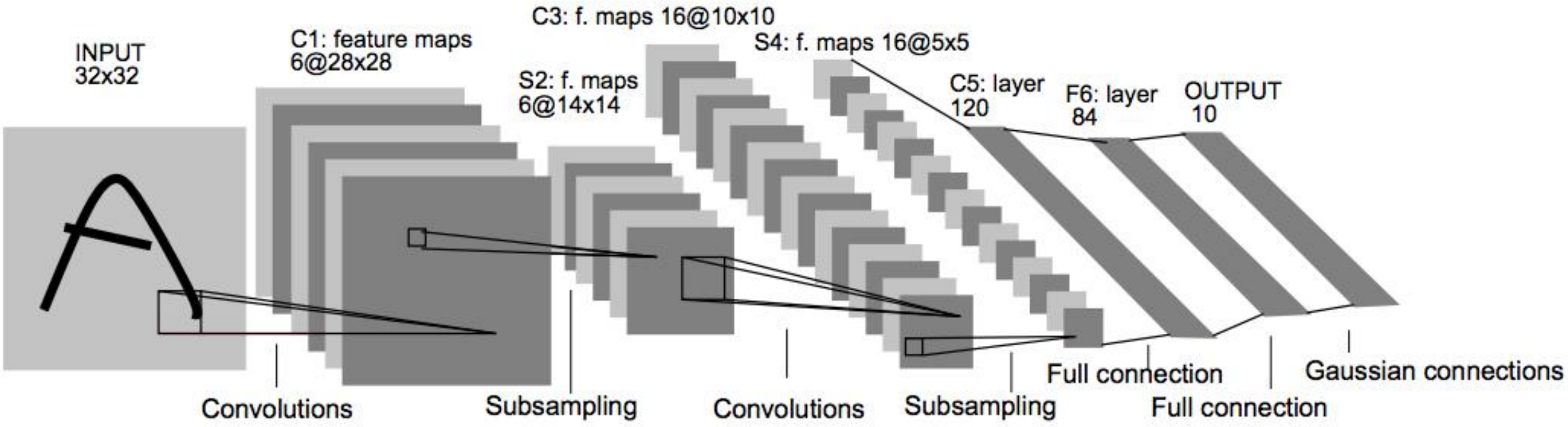
# Lecture 4 – What does AI/ brain learn?

- Memory size & parameter size of CNN
  - how to calculate the memory size and parameter size
  - LeNet, AlexNet, VGG-16
- What do neurons in CNN learn?
  - Visualize the activation of neurons
  - Visualize the weights of a filter
  - Synthesize image that maximize neural activity
  - Some existing tools
- What do neurons in brain learn?
  - Synthesize images that maximize V4 neurons
- Interactions between biological neurons and artificial neurons

# Memory size & parameter size of CNN

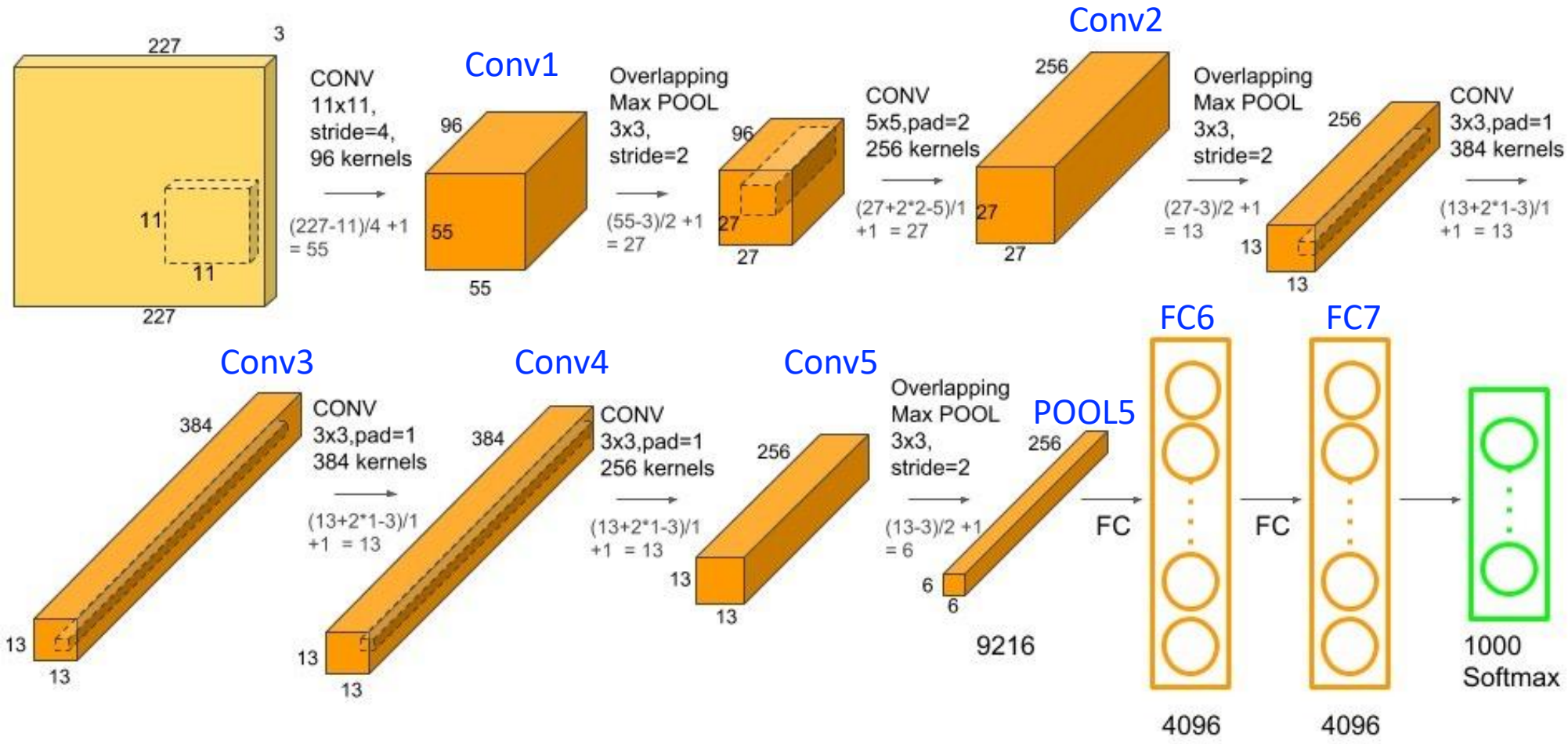


# Memory size & parameter size of LeNet

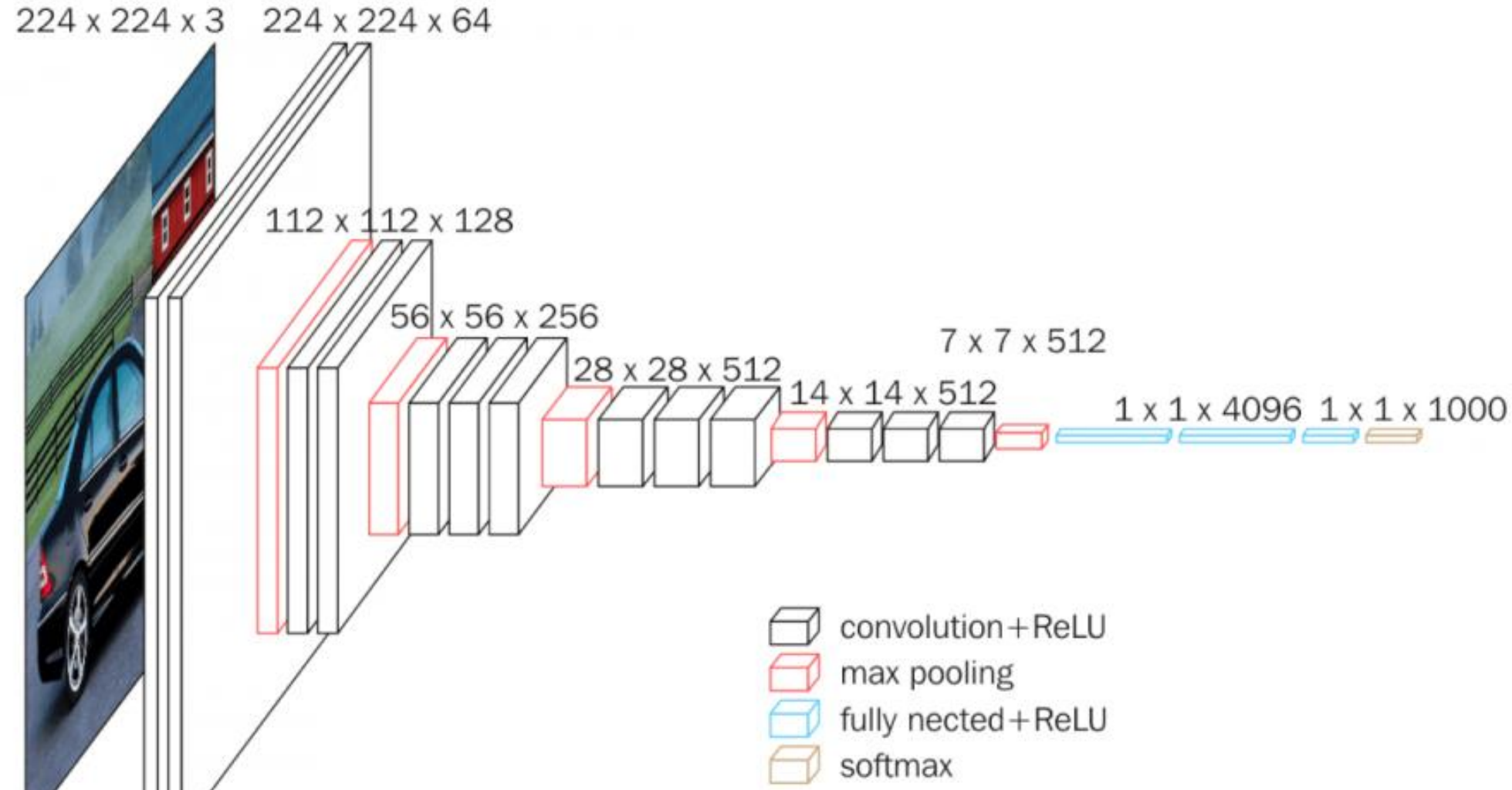



[5x5]x6 [5x5x6]x16 120x84 84x10

# Memory size & parameter size of AlexNet



# Memory size & parameter size of VGG-16



| | | |
|---|---|---|
| INPUT: [224x224x3] | memory: 224*224*3=150K | weights: 0 |
| CONV3-64: [224x224x64] | memory: 224*224*64=3.2M | weights: (3*3*3)*64 = 1,728 |
| CONV3-64: [224x224x64] | memory: 224*224*64=3.2M | weights: (3*3*64)*64 = 36,864 |
| POOL2: [112x112x64] | memory: 112*112*64=800K | weights: 0 |
| CONV3-128: [112x112x128] | memory: 112*112*128=1.6M | weights: (3*3*64)*128 = 73,728 |
| CONV3-128: [112x112x128] | memory: 112*112*128=1.6M | weights: (3*3*128)*128 = 147,456 |
| POOL2: [56x56x128] | memory: 56*56*128=400K | weights: 0 |
| CONV3-256: [56x56x256] | memory: 56*56*256=800K | weights: (3*3*128)*256 = 294,912 |
| CONV3-256: [56x56x256] | memory: 56*56*256=800K | weights: (3*3*256)*256 = 589,824 |
| CONV3-256: [56x56x256] | memory: 56*56*256=800K | weights: (3*3*256)*256 = 589,824 |
| POOL2: [28x28x256] | memory: 28*28*256=200K | weights: 0 |
| CONV3-512: [28x28x512] | memory: 28*28*512=400K | weights: (3*3*256)*512 = 1,179,648 |
| CONV3-512: [28x28x512] | memory: 28*28*512=400K | weights: (3*3*512)*512 = 2,359,296 |
| CONV3-512: [28x28x512] | memory: 28*28*512=400K | weights: (3*3*512)*512 = 2,359,296 |
| POOL2: [14x14x512] | memory: 14*14*512=100K | weights: 0 |
| CONV3-512: [14x14x512] | memory: 14*14*512=100K | weights: (3*3*512)*512 = 2,359,296 |
| CONV3-512: [14x14x512] | memory: 14*14*512=100K | weights: (3*3*512)*512 = 2,359,296 |
| CONV3-512: [14x14x512] | memory: 14*14*512=100K | weights: (3*3*512)*512 = 2,359,296 |
| POOL2: [7x7x512] | memory: 7*7*512=25K | weights: 0 |
| FC: [1x1x4096] | memory: 4096 | weights: 7*7*512*4096 = 102,760,448 |
| FC: [1x1x4096] | memory: 4096 | weights: 4096*4096 = 16,777,216 |
| FC: [1x1x1000] | memory: 1000 | weights: 4096*1000 = 4,096,000 |

**TOTAL memory**: 24M * 4 bytes ~= **93MB** / image (only forward! ~*2 for backward)

**TOTAL parameters**: **138M** parameters

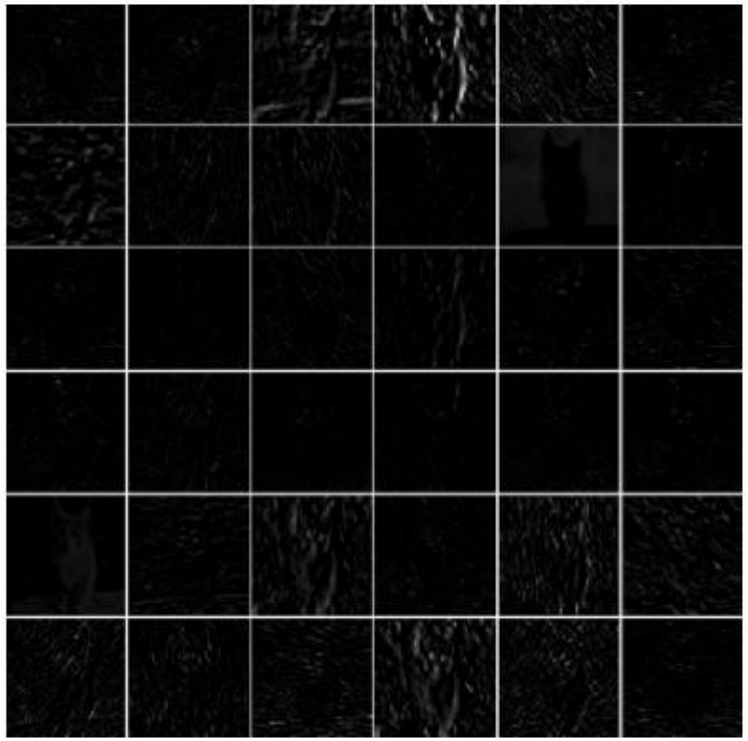
# What do neurons in CNN learn?

- Visualize the activation of neurons
- Visualize the weights of a filter
- Synthesize images that maximize neural activity

# Visualize the activation of neurons

Cat → AlexNet

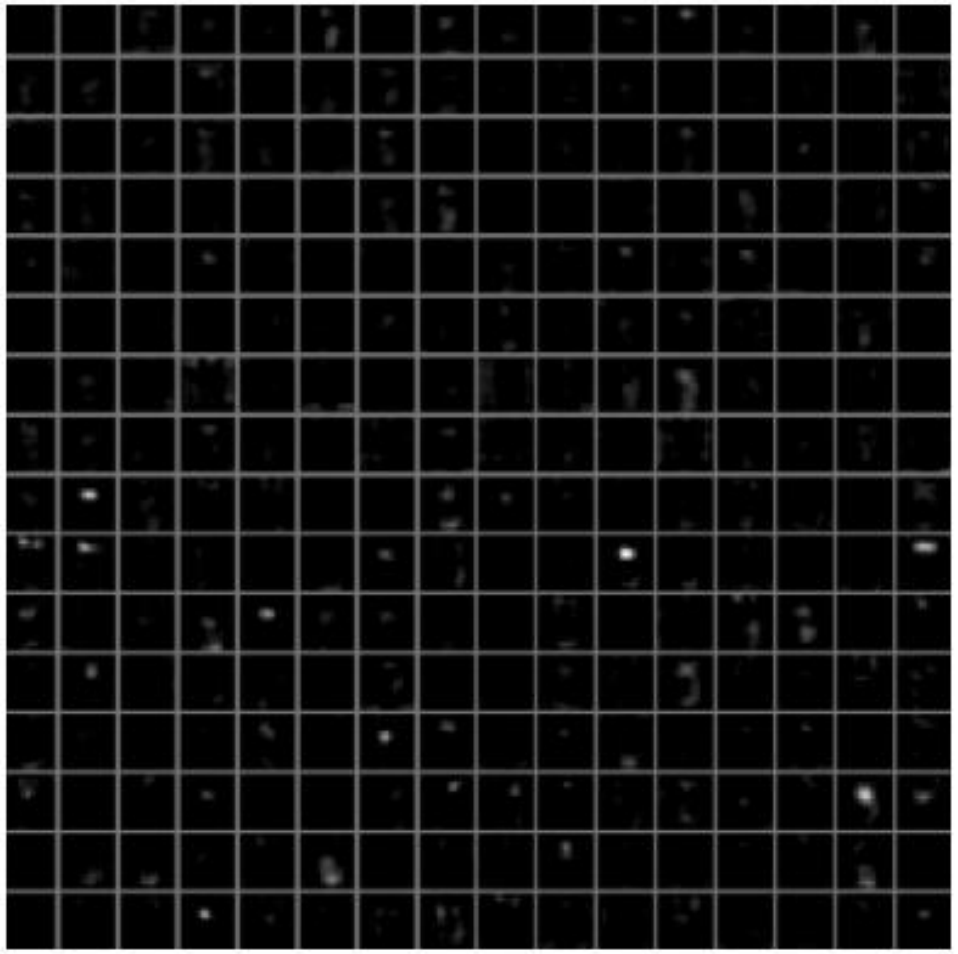
First CONV Layer



Every box shows an activation map corresponding to some filter.

Notice that the activations are sparse (most values are zero) and mostly local.

5th CONV layer

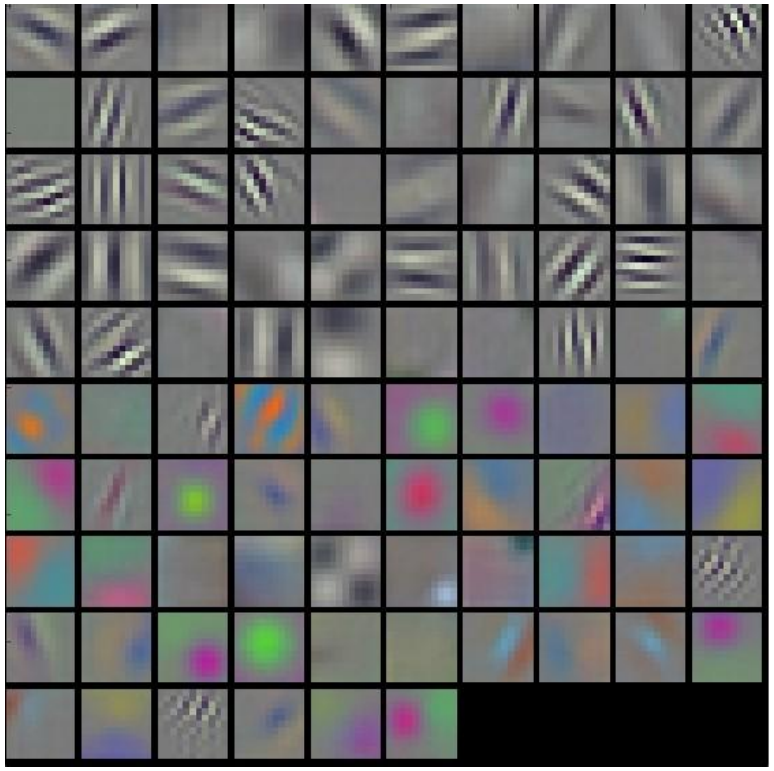


**Hardly interpretable!**

http://cs231n.github.io/understanding-cnn/

# Visualize the weights of a filter

Filters on the 1st CONV layer [11 x 11] x 96

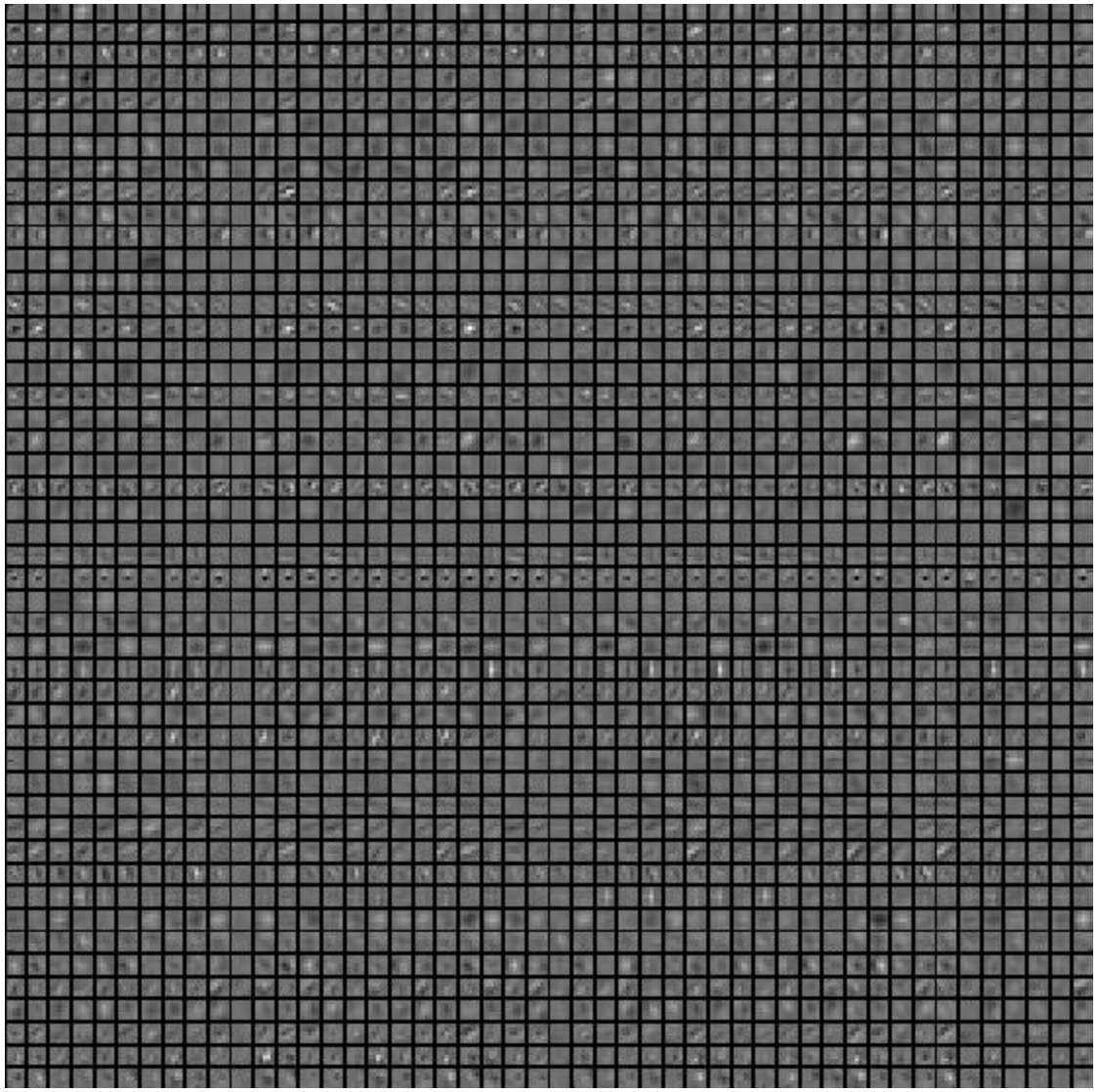


The first-layer weights are very nice and smooth, indicating nicely converged network.

AlexNet contains two separate streams of processing: 1) high-frequency grayscale features; 2) low-frequency color features.



Filters on the 2nd CONV layer [5 x 5] x 256



The 2nd CONV layer weights are not as interpretable, but it is apparent that they are still smooth, well-formed, and absent of noisy patterns.

# Visualize the input image maximally activating neurons

Maximally activating images for some **POOL5** (5th pool layer) neurons of an AlexNet.
The activation values and the receptive field of the particular **neuron** are shown in white.
In particular, note that the POOL5 neurons are a function of a relatively large portion of the input image!



Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR, 2014 (cited **14352** times)

# ***What does CNN learn?*** → *What input pattern maximize the activation of a filter?*

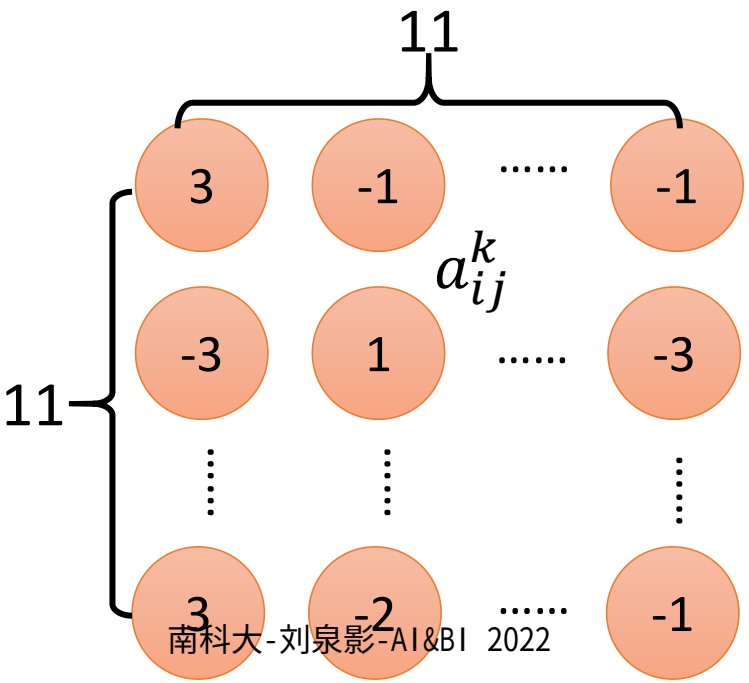
The output of the k-th filter is a 11 x 11 matrix, $[a_{ij}^k]$

The total activation of the k-th filter: $a^k = \sum_{i=1}^{11}\sum_{j=1}^{11} a_{ij}^k$

**Find the specific input x, to maximize k-th filter:**

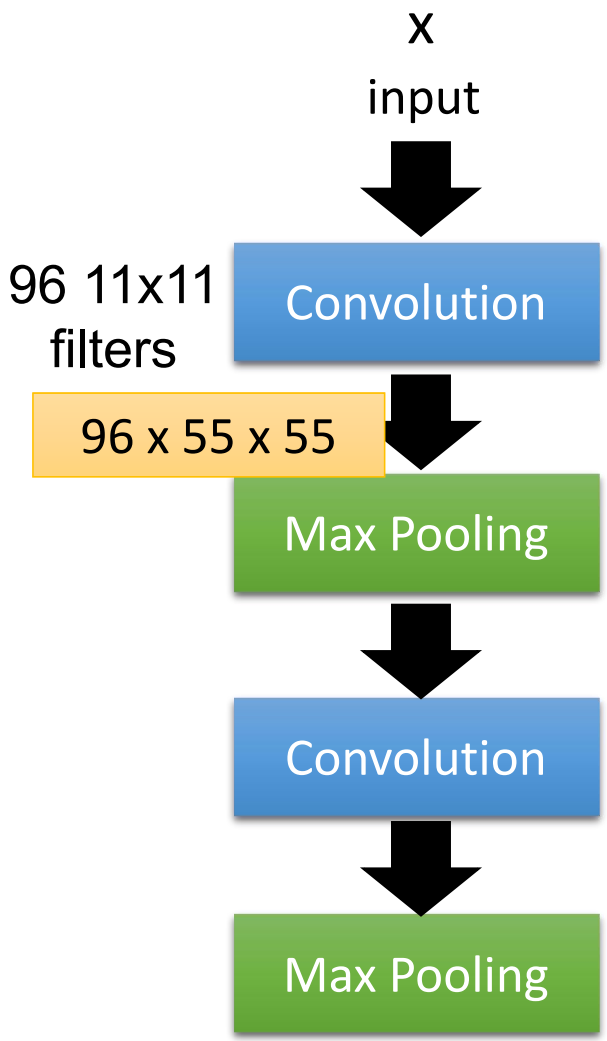
$$x^* = arg\max_x a^k$$

A *non-convex* optimization problem




**Gradient ascent** $\frac{\partial a^k}{\partial x_{ij}}$

1. Start at $x_{ij}{}^0$
2. Compute gradient $\frac{\partial a^k}{\partial x_{ij}}$ at $x_{ij}{}^0$
3. $x_{ij}{}^1 = x_{ij}{}^0 + \eta \frac{\partial a^k}{\partial x_{ij}}$
4. Repeat 2-3 until convergent (stopping criterion)




Erhan, Bengio, Courville and Vincent (2009). Visualizing higher-layer features of a deep network

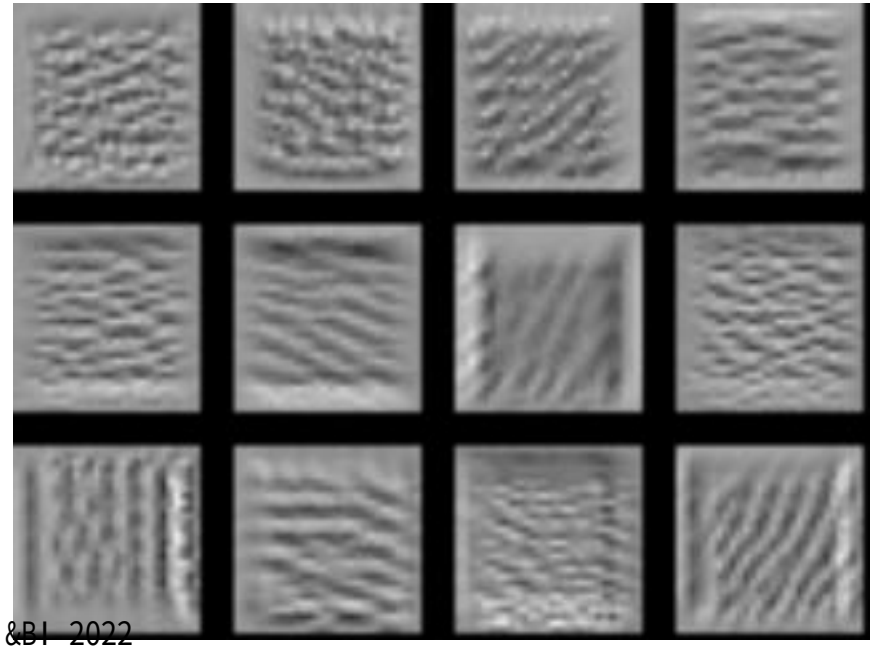
# ***What does CNN learn?*** → *What input pattern maximize the activation of a filter?*

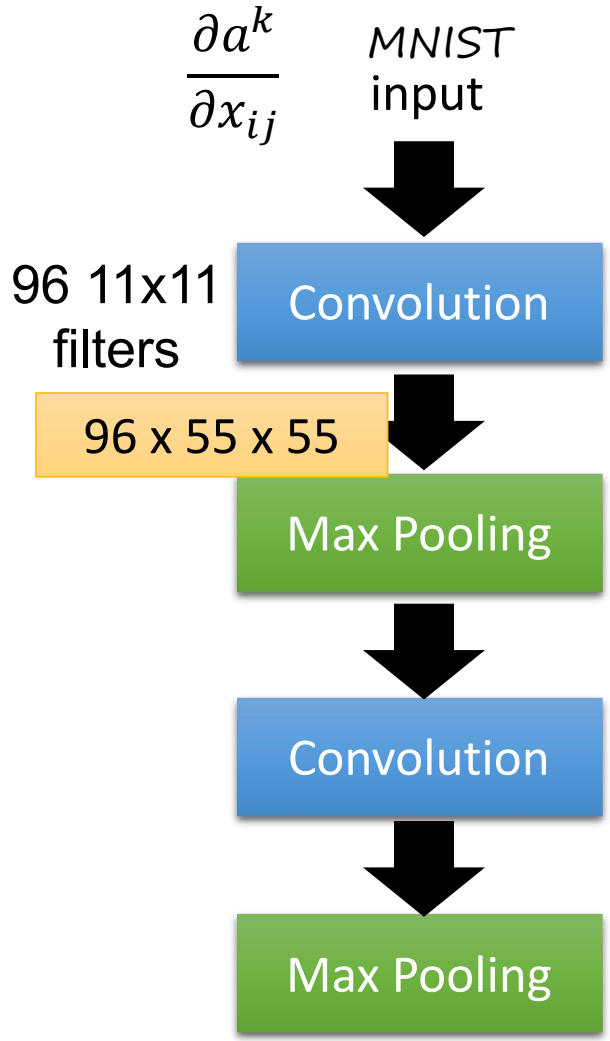
The output of the k-th filter is a 11 x 11 matrix, $[a_{ij}^k]$

The total activation of the k-th filter: $a^k = \sum_{i=1}^{11}\sum_{j=1}^{11} a_{ij}^k$

**Find the specific input x, to maximize k-th filter:**

$$x^* = arg\max_x a^k \qquad \text{(gradient ascent)} \qquad \frac{\partial a^k}{\partial x_{ij}}$$

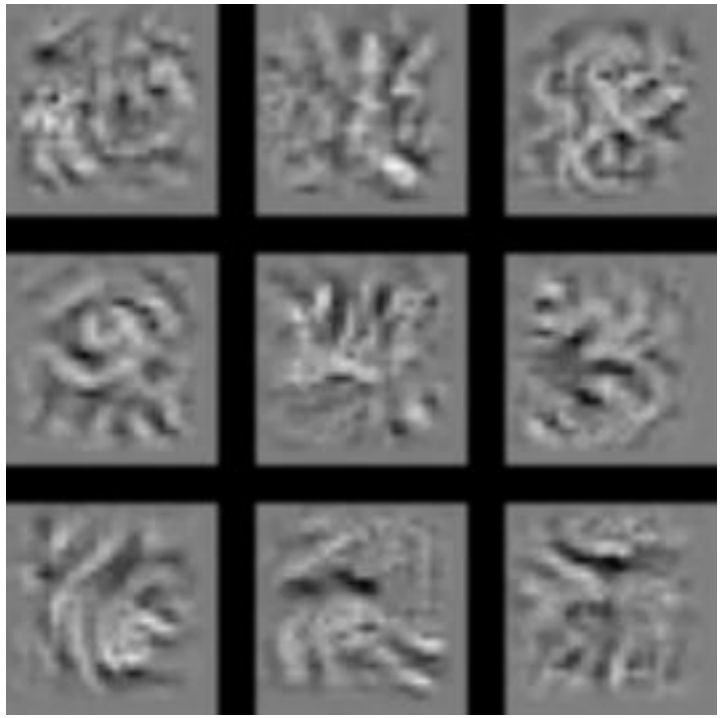




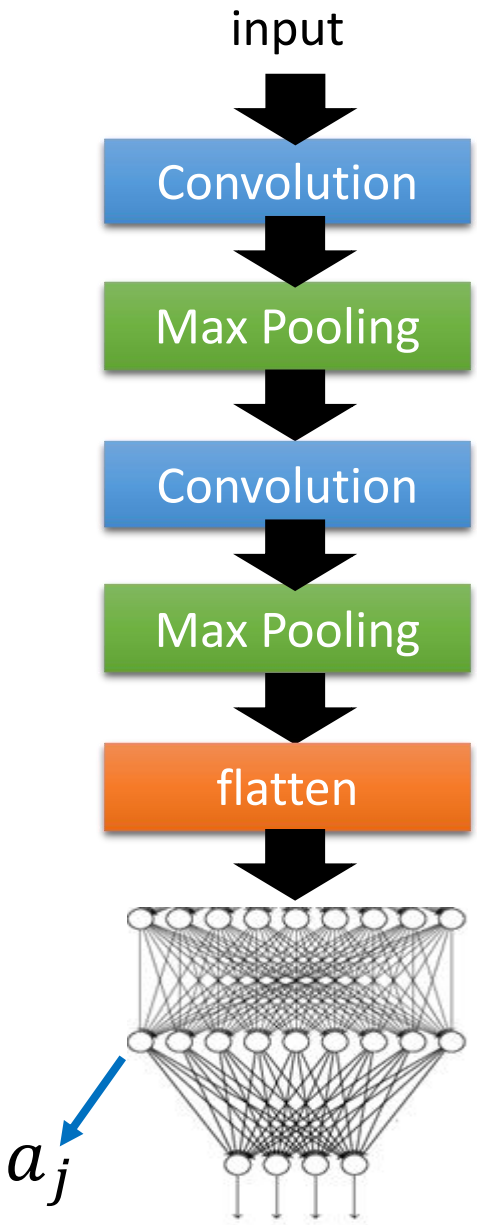
# ***What does CNN learn?*** → *What input pattern maximize the activation of a neuron?*

Find an image maximizing the output of neuron:

$$x^* = arg\max_x a^j$$



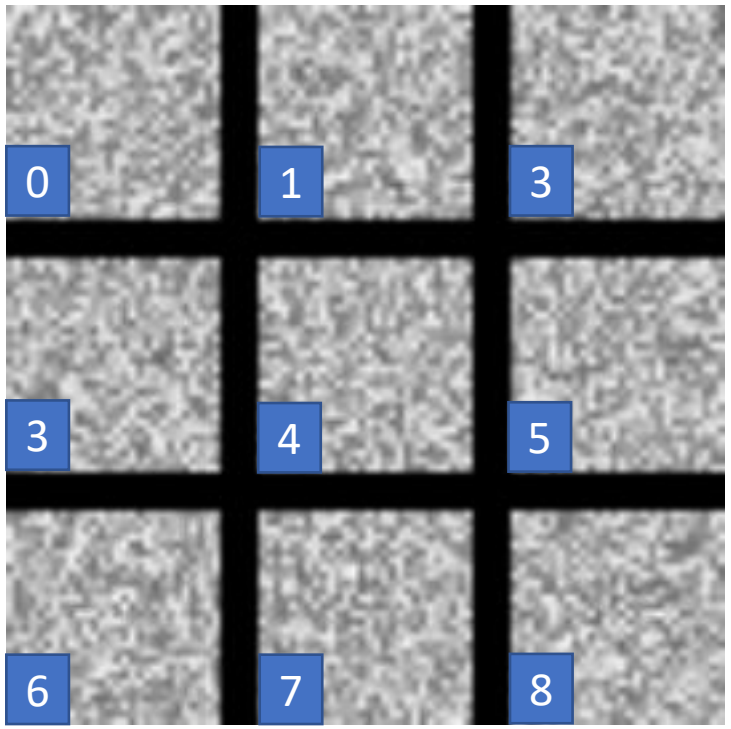
Each figure corresponds to a neuron

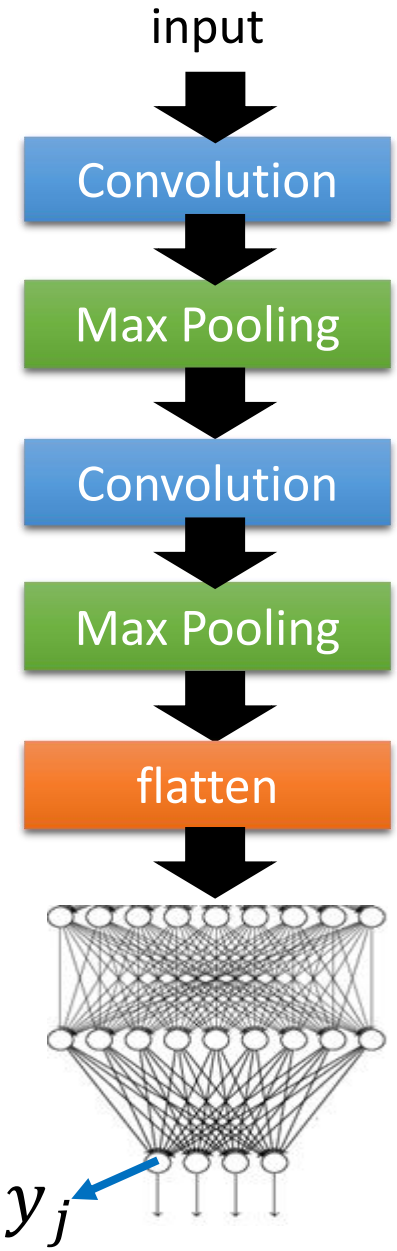


# ***What does CNN learn?*** → *What input pattern maximize the activation of an output?*

$$x^* = arg\max_x y^i$$

Vote: Can we see digits?




Deep Neural Networks are Easily Fooled

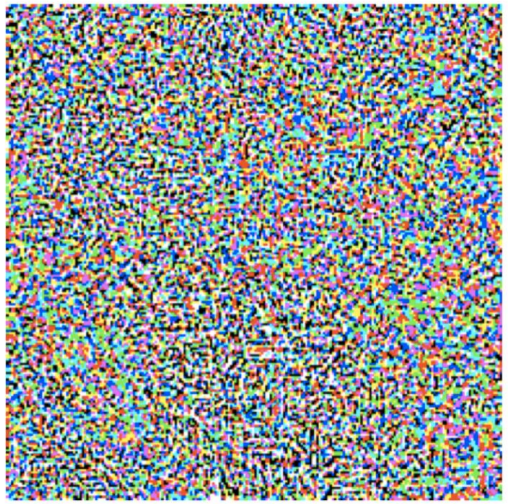
https://www.youtube.com/watch?v=M2IebCN9Ht4

# Adversarial examples easily attack CNN

$\boldsymbol{x}$

"panda"
57.7% confidence

$+ .007 \times$



$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

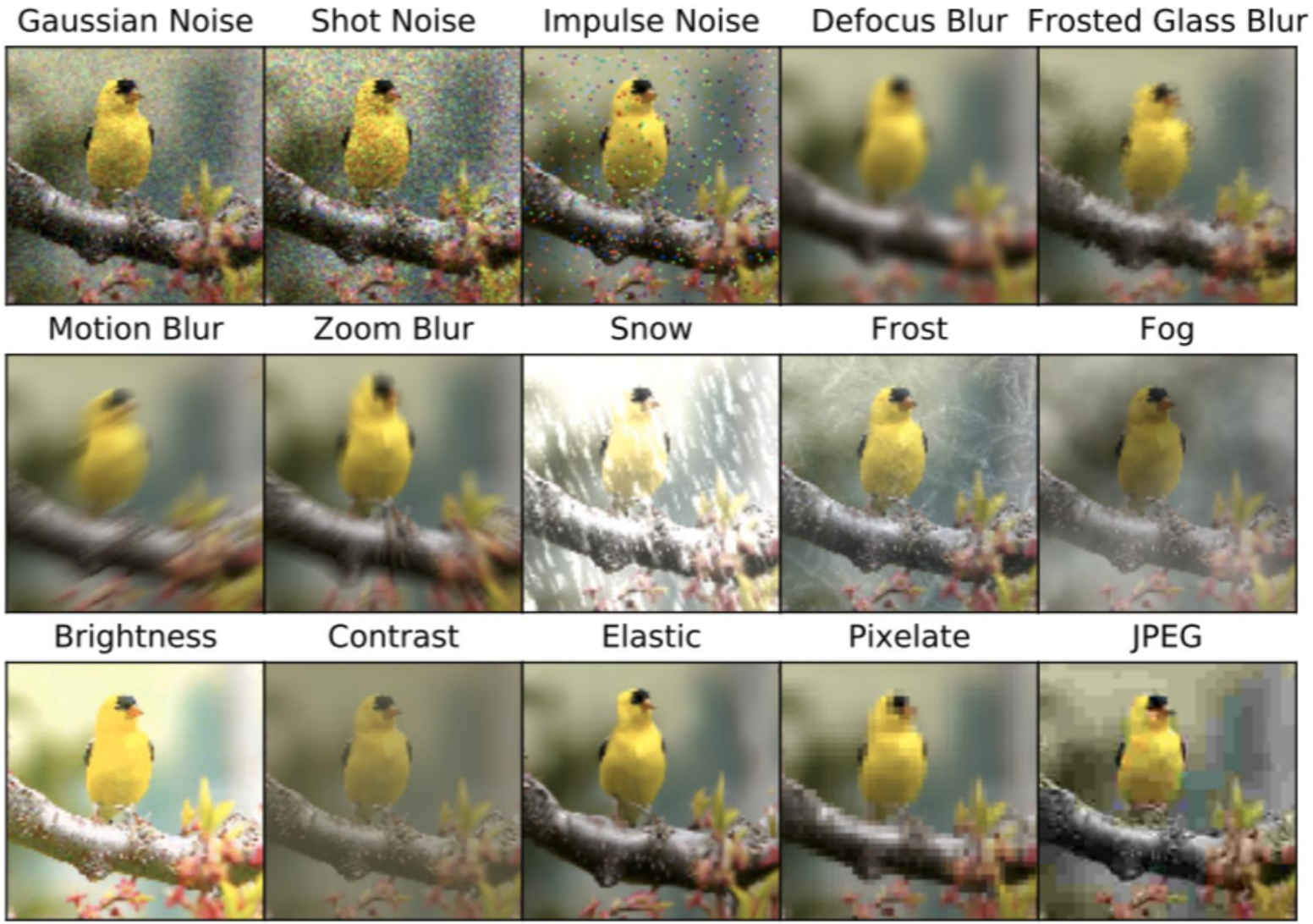
$=$

$\boldsymbol{x} + \epsilon \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
99.3 % confidence

Ian J. Goodfellow et al., Explaining and harnessing adversarial examples, ICLR, 2015

# Humans are robust to corruptions and perturbations




Dan Hendrycs et al., Benchmarking Neural Network Robustness to Common Corruptions and Perturbations, ICLR, 2019

# Humans are robust to corruptions and perturbations


1. Can we generate adversarial examples to fool only humans but not AI?
2. Can we generate adversarial examples to fool both AI and humans?

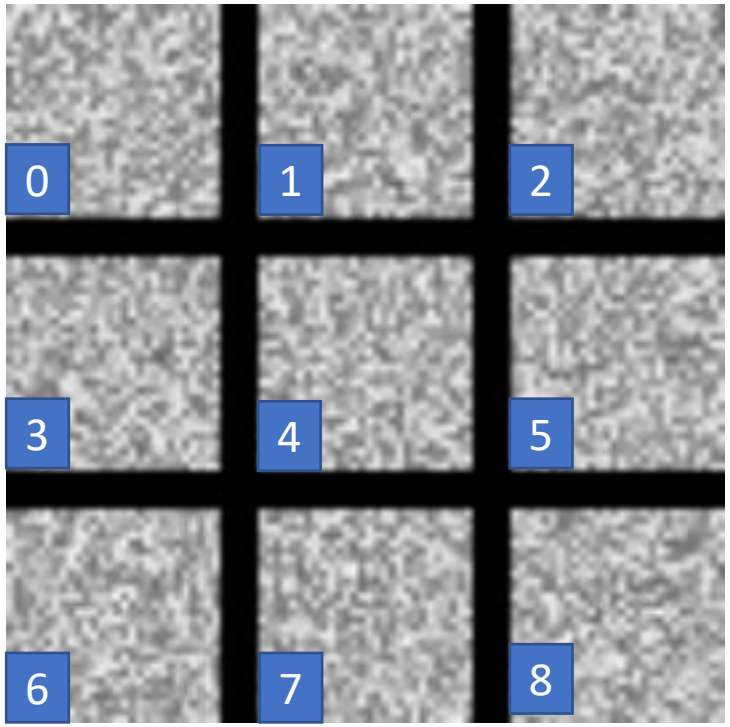
**A review/survey on this topic?**

Dan Hendrycs et al., Benchmarking Neural Network Robustness to Common Corruptions and Perturbations, ICLR, 2019

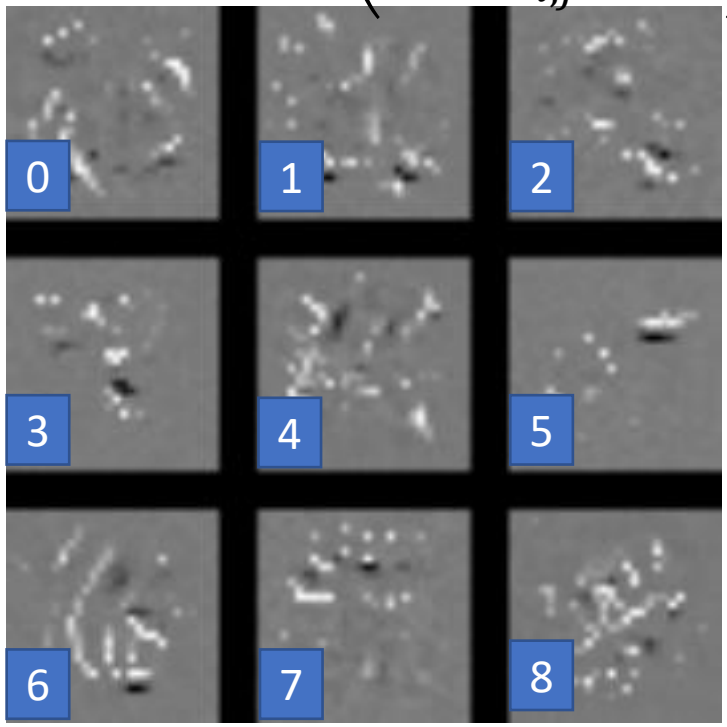
# ***What does CNN learn?*** → *What input pattern maximize the activation of an output?*

Regularization → sparsity

$$x^* = arg\max_x y^i$$



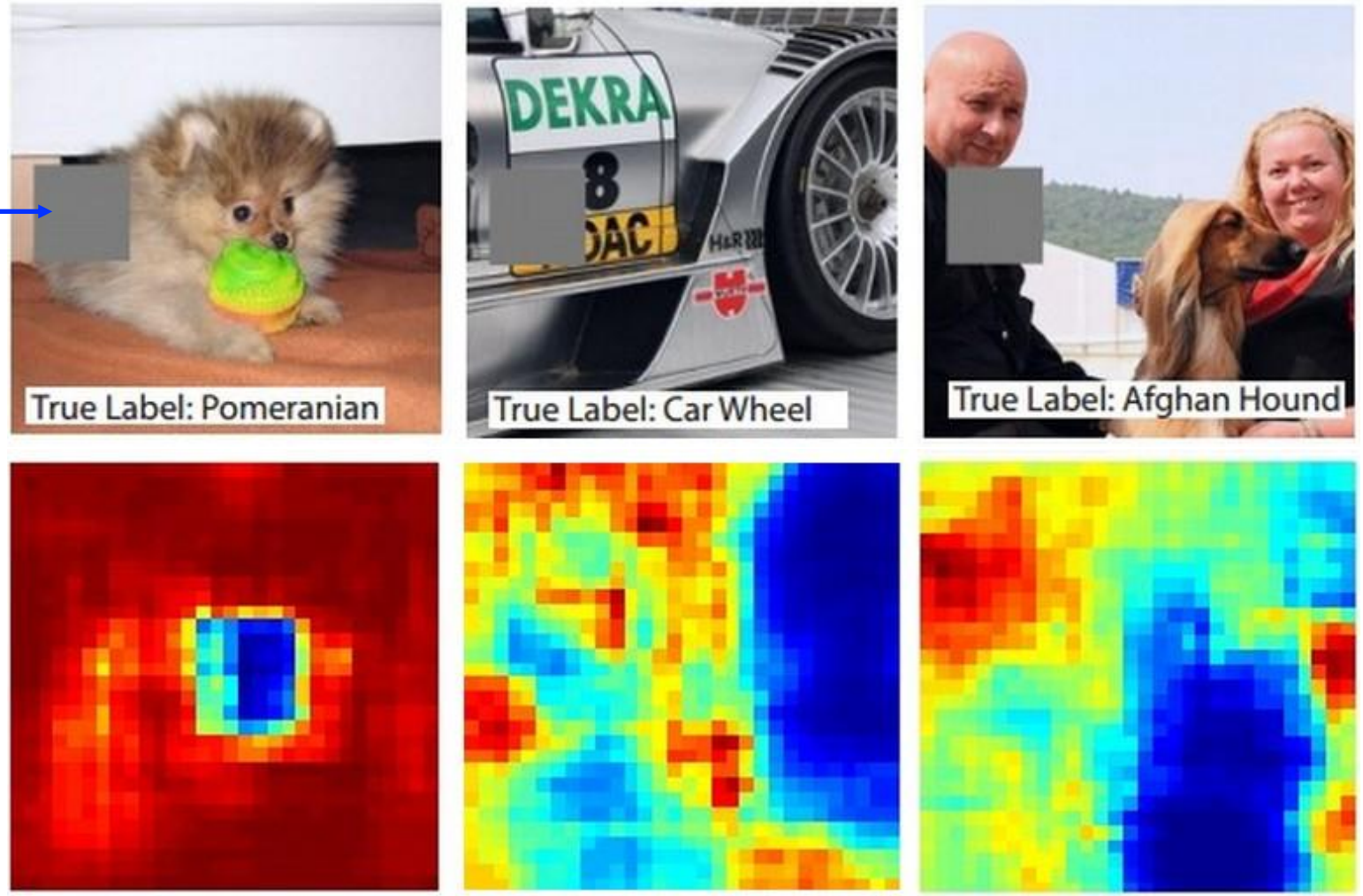

$$x^* = arg\max_x \left( y^i - \sum_{i,j} |x_{ij}| \right)$$



# Occluding parts of the image

Some parts of an image is more important than other parts in the object recognition task.

Occluding parts



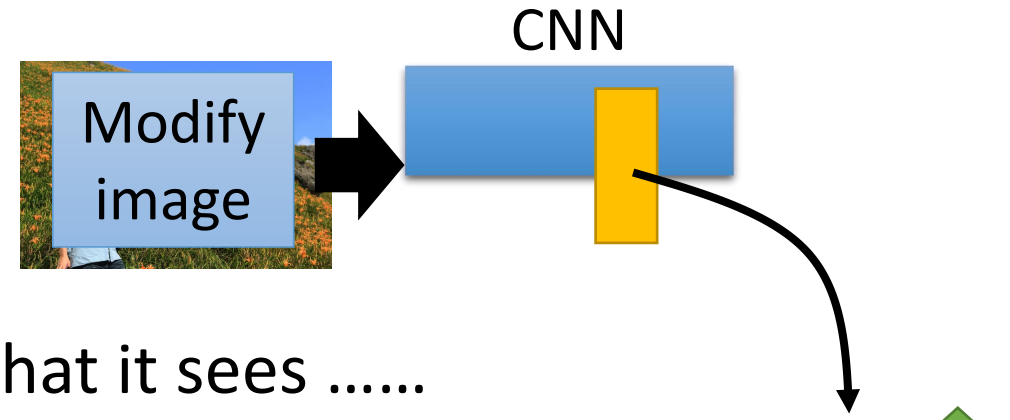

The probability of the correct class

Blue part → important

Red part → negligible

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. *ECCV 2014*

# Deep Dream




- Given a photo, machine adds what it sees ……


http://deepdreamgenerator.com/

# Deep Dream

- Given a photo, machine adds what it sees ……

http://deepdreamgenerator.com/

# Some existing tools to understand CNNs

**Visualizing and Understanding CNNs** (ECCV 2014)

https://github.com/huybery/VisualizingCNN

**CNN explainer: Learn CNN in your browser**

web https://poloclub.github.io/cnn-explainer/

code https://github.com/poloclub/cnn-explainer

CNN Explainer: Learning Convolutional Neural Networks with Interactive Visualization. Wang, Zijie J., Robert Turko, Omar Shaikh, Haekyu Park, Nilaksh Das, Fred Hohman, Minsuk Kahng, and Duen Horng Chau. IEEE Transactions on Visualization and Computer Graphics (TVCG), 2020.

**CNN Visualizations** (implement various methods)

https://github.com/utkuozbulak/pytorch-cnn-visualizations

**Implemented Techniques**

- Gradient visualization with vanilla backpropagation
- Gradient visualization with guided backpropagation [1]
- Gradient visualization with saliency maps [4]
- Gradient-weighted class activation mapping [3] (Generalization of [2])
- Guided, gradient-weighted class activation mapping [3]
- Score-weighted class activation mapping [15] (Gradient-free generalization of [2])
- Element-wise gradient-weighted class activation mapping [16]
- Smooth grad [8]
- CNN filter visualization [9]
- Inverted image representations [5]
- Deep dream [10]
- Class specific image generation [4] [14]
- Grad times image [12]
- Integrated gradients [13]
- Layerwise relevance propagation [17]

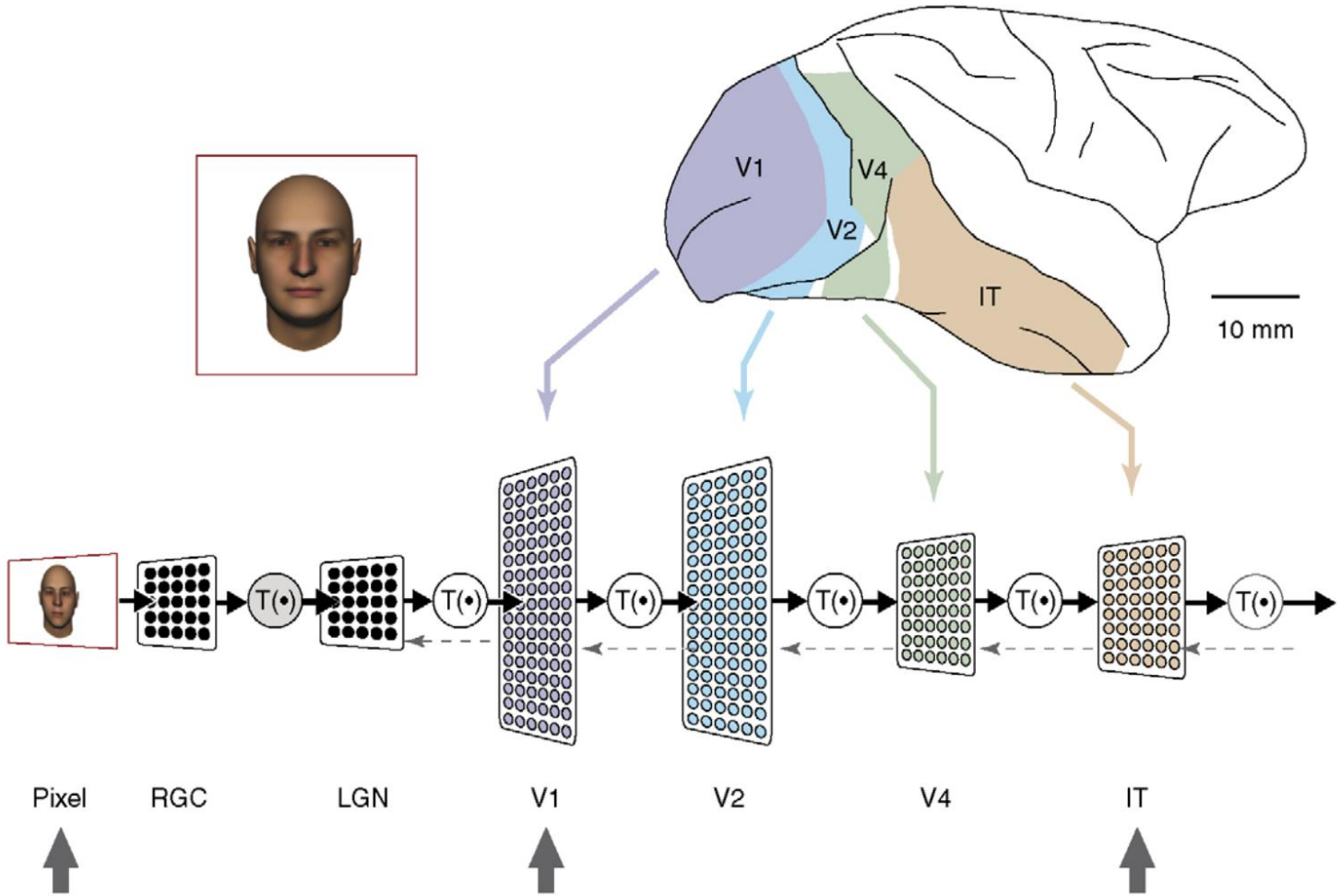
# What do neurons in brain learn?

- Visualize the activation of neurons
- Visualize the weights of a filter
- Synthesize images that maximize neural activity

# What do neurons in brain learn?

- Visualize the activation of neurons ( √ )
- Visualize the weights of a filter ( ? )
- Synthesize images that maximize neural activity ( √ )

# Neuronal populations along the ventral visual processing stream.

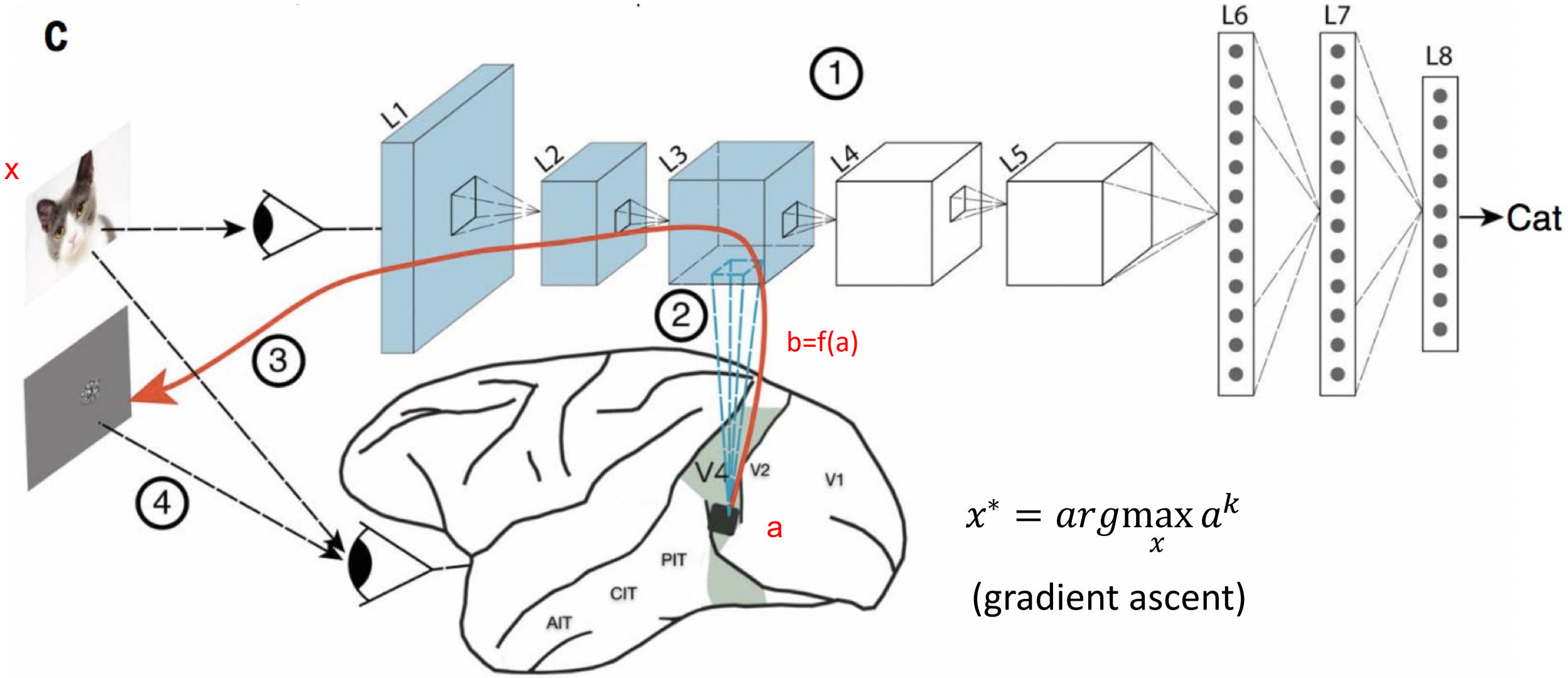



RGC, retinal ganglion cells;
LGN, lateral geniculate nucleus.

We show a lateral schematic of a rhesus monkey brain (adapted from Ref. [26]).
The lower panels schematically illustrate these populations in early visual areas and at successively higher stages along the ventral visual stream – their relative size loosely reflects their relative output dimensionality (approximate number of feed-forward projection neurons).
A given pattern of photons from the world (here, a face) is transduced into neuronal activity at the retina and is progressively and rapidly transformed and re-represented in each population, perhaps by a common transformation (T).
**Solid arrows** indicate the direction of visual information flow based on neuronal latency ( 100 ms latency in IT), but this does not preclude fast feedback both within and between areas (**dashed arrows**). The gray arrows across the bottom indicate the population representations for the retina, V1 and IT, respectively.

DiCarlo and Cox, TiCS, 2007

[Ref 26] Felleman, D.J. and Van Essen, D.C. (1991) Distributed hierarchical processing in the primate cerebral cortex. Cereb. Cortex 1, 1–47

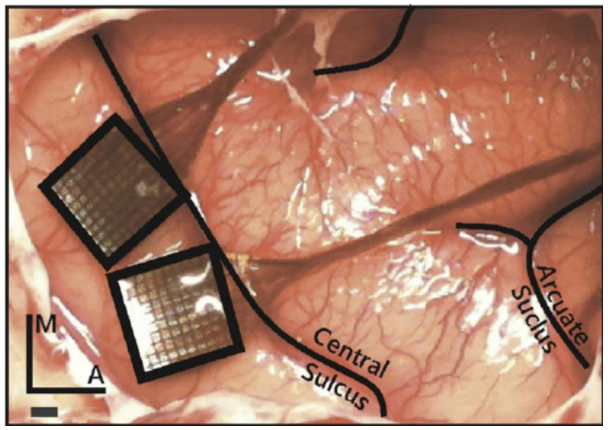



$$x^* = arg\max_x a^k$$

(gradient ascent)

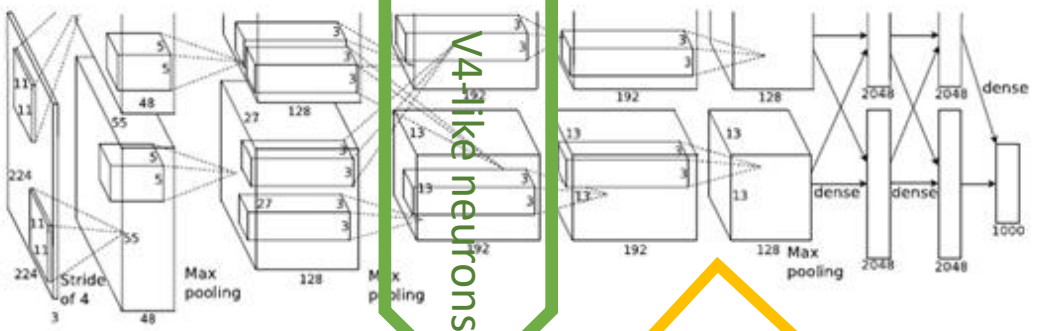
CNN matches the ventral visual pathway to some extent.

Yamins, nn, 2016
Bashivan, Science, 2019

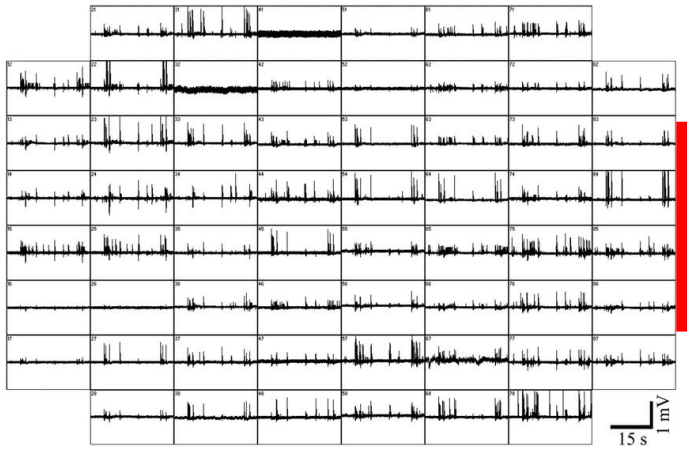



# CNN经典模型

# AlexNet




V4-like neurons




V4 neurons

Convolutional mapping function

Mapped V4 encoding model

We used the responses of the recorded V4 neural sites in each monkey and the responses of all the model "V4" neurons to build a mapping from the model to the recorded population of V4 neural sites (Fig. 1). We used a convolutional mapping function that significantly reduces the neural prediction error compared to other methods such as principal component regression. Our implementation was a variant of the two-stage convolutional mapping function proposed in (*30*) in which we substituted the group sparsity regularization term with an L2 loss term to allow for smooth (nonsparse) feature mixing. The first stage of the mapping function consists of a learnable spatial mask ($W_s$) that is parameterized separately for each neural site ($n$) and is used to estimate the receptive field of each neuron. The second stage consists of a mixing pointwise convolution ($W_d$) that computes a weighted sum of all feature maps at a particular layer of the ANN model (Conv3 layer in our case). The mixing

$$\hat{y}_n = \left[\sum \left(W_s^{(n)} \cdot X\right)\right] * W_d^{(n)} + W_b^{(n)} \quad (1)$$

st combination of model fea-
dictive of each neural site's
l output is then averaged over
s to form a scalar prediction of
se. Parameters are jointly op-
ze the prediction error $\mathcal{L}_e$ on

$$\mathcal{L}_2 = \lambda_s \sum_n W_s^{(n)^2} + \lambda_d \sum_n W_d^{(n)^2} \quad (2)$$

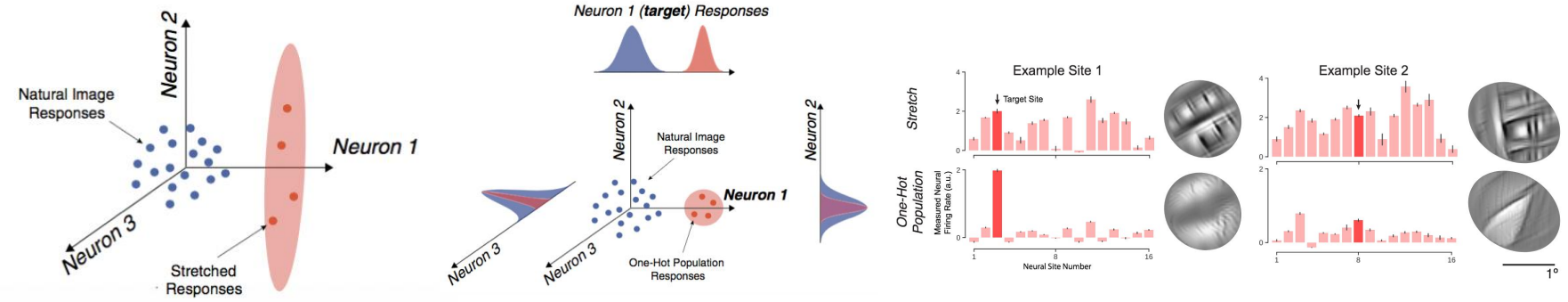
$$\mathcal{L}_{\text{Laplace}} = \lambda_s \sqrt{\sum_n \left(W_s^{(n)} * L\right)^2}, \; L = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad (3)$$

$$\mathcal{L}_e = \sqrt{\sum_n (\hat{y}_n - y)^2} \quad (4)$$

$$\mathcal{L} = \mathcal{L}_e + \mathcal{L}_{\text{Laplace}} + \mathcal{L}_2 \quad (5)$$

Bashivan, Pouya, Kohitij Kar, and James J. DiCarlo. "Neural population control via deep image synthesis." *Science* (2019)

# *Mapped V4 encoding model*




Synthesis via Optimization

Stretch - iteratively change the pixel values in the direction of gradients
5 different random start seeds → 5 stretch images

One hot population (OHP) – Maximize the following objective function to generate OHP images

$$\mathcal{S} = \text{softmax}_t(y) = \frac{\exp(y_t)}{\sum_i \exp(y_i)}$$

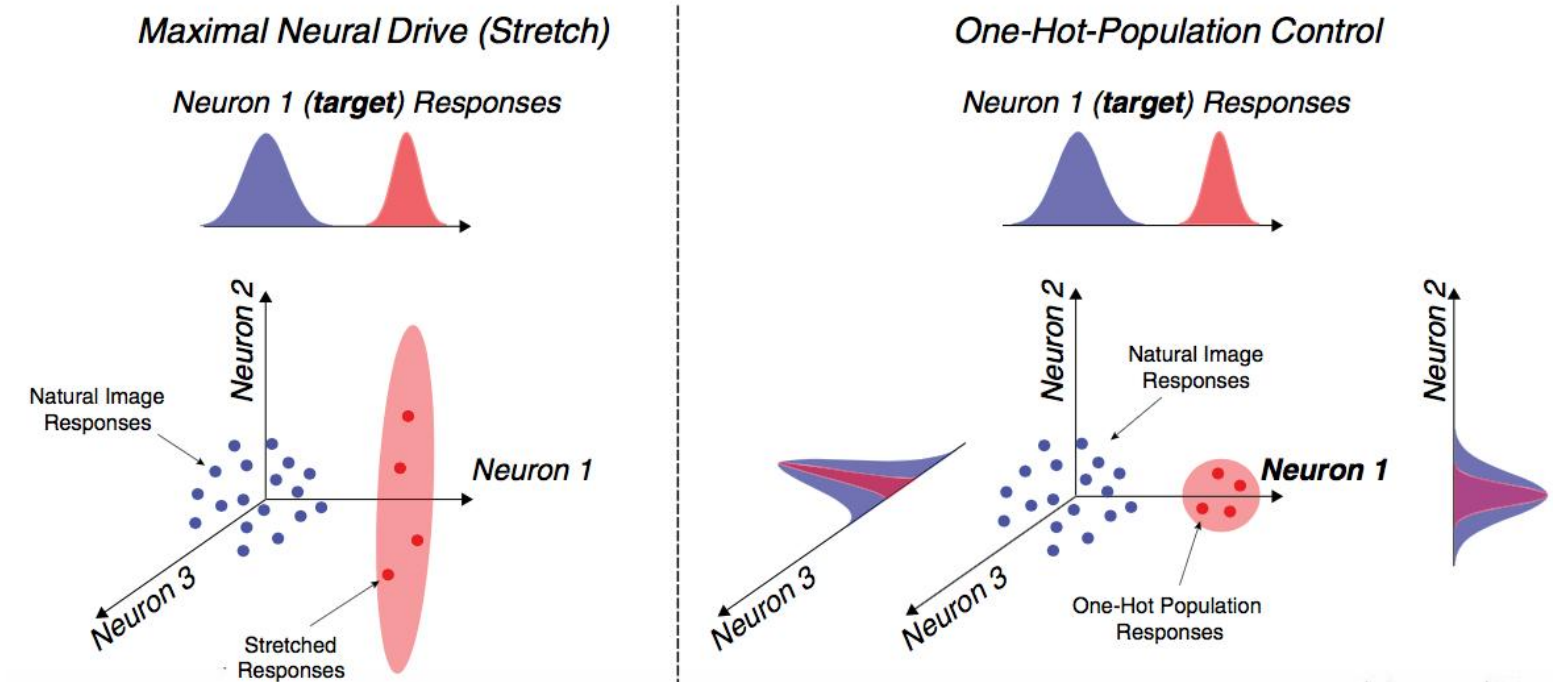
Controller image
Contrast enhancement

(neurons in area V4 respond more strongly to higher contrast stimuli)

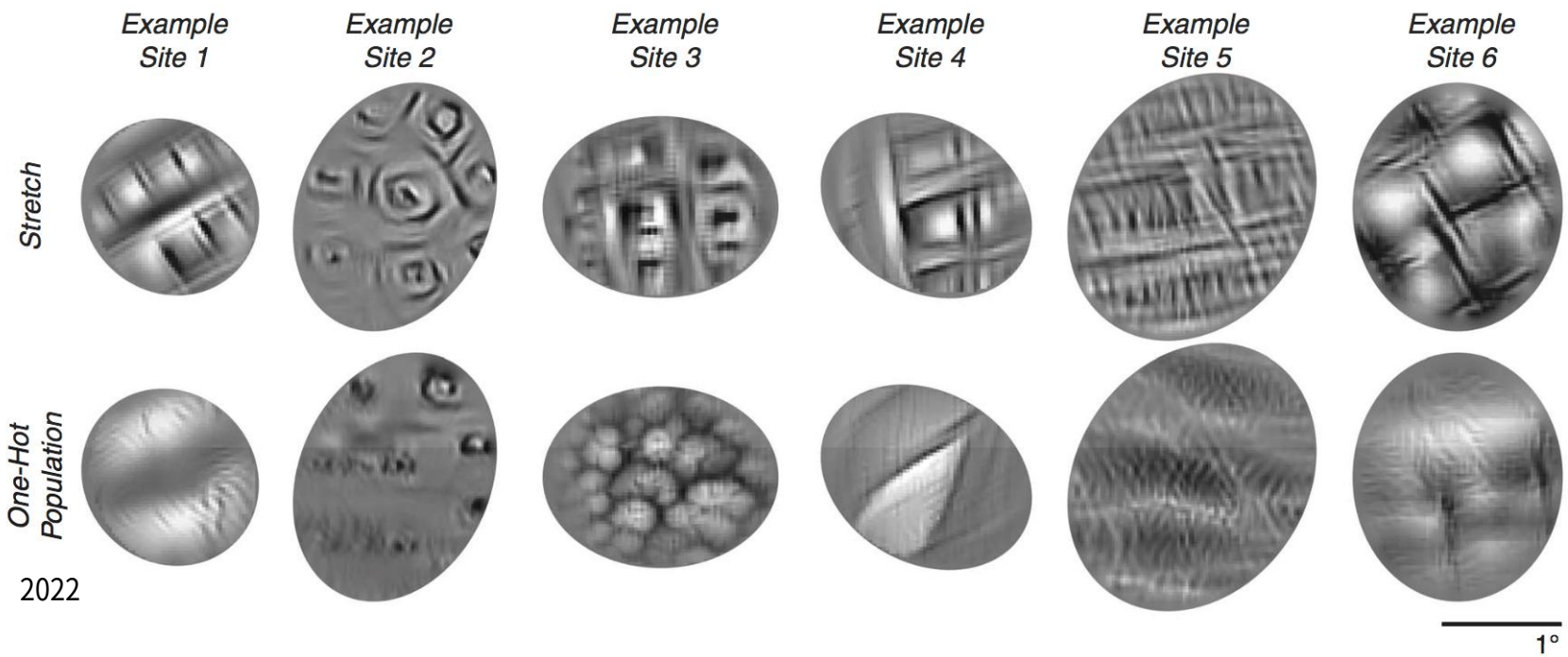
Various controller image uniqueness tests

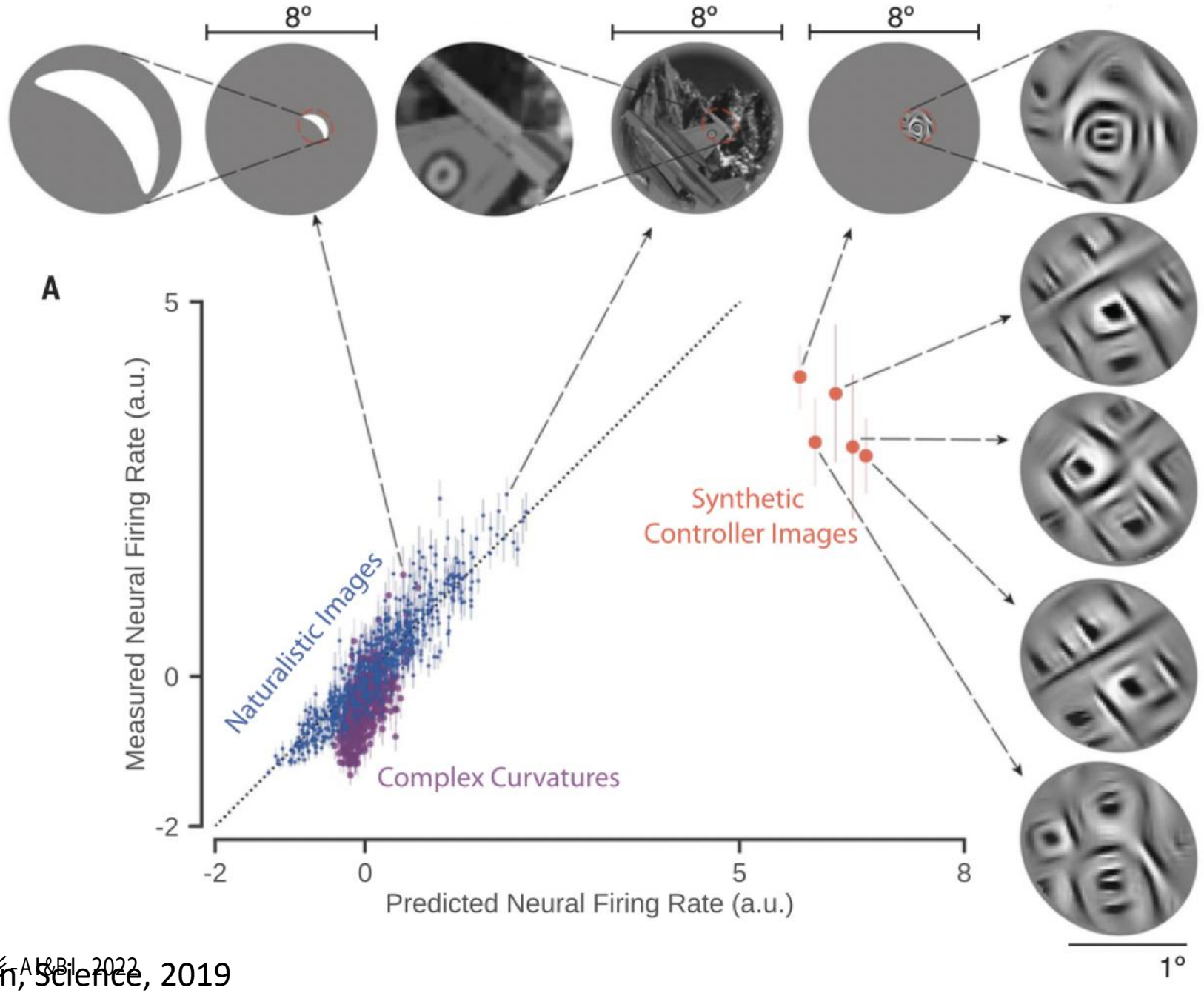
(to see whether controller image driven stretching was not mere 'coincidence')

$$x^* = \arg\max_x a^k$$




$$x^* = \arg\max_x \frac{e^{a^k}}{\sum_k e^{a^k}}$$

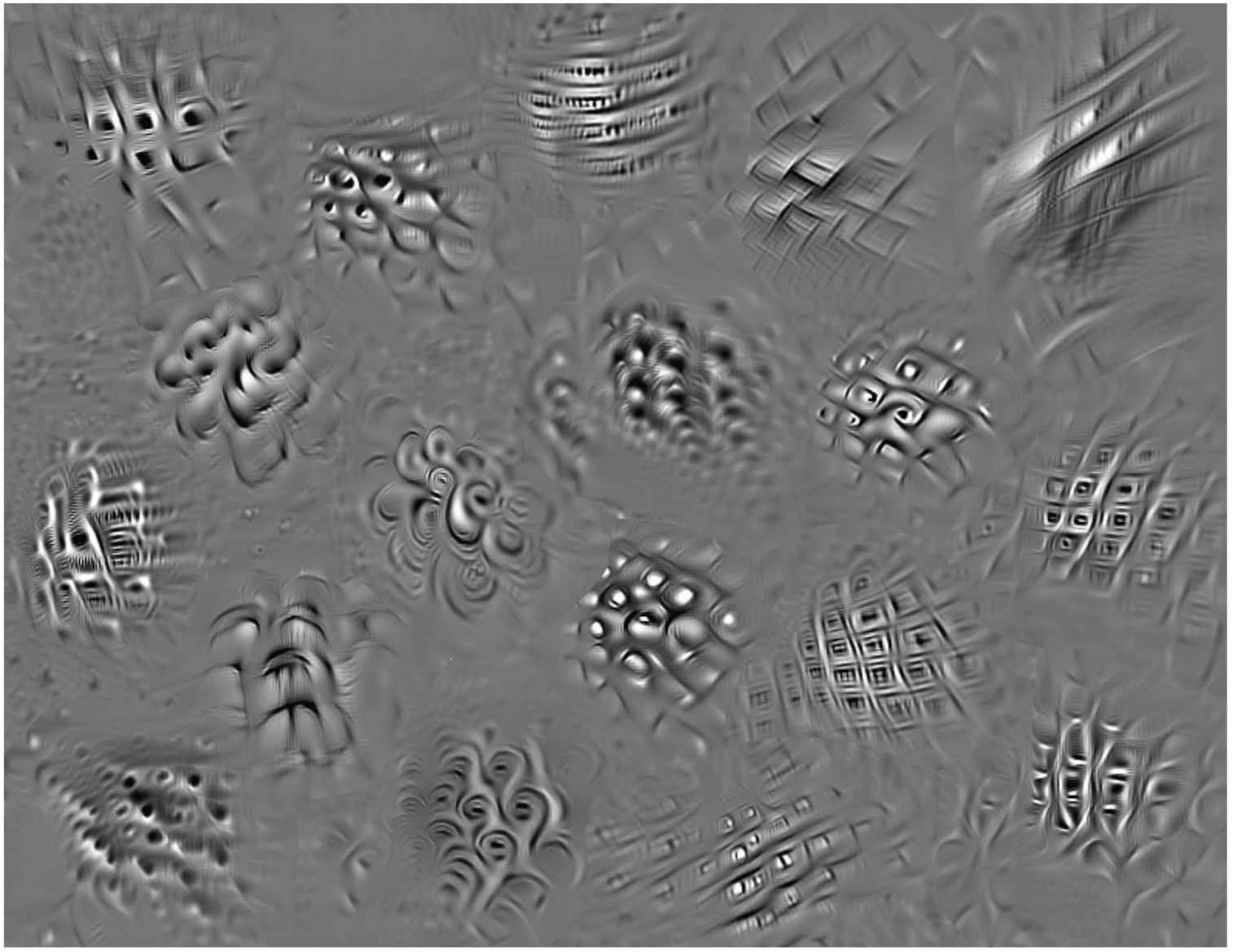





$$x^* = arg\max_x a^k$$

(gradient ascent)

Bashivan, Science, 2019



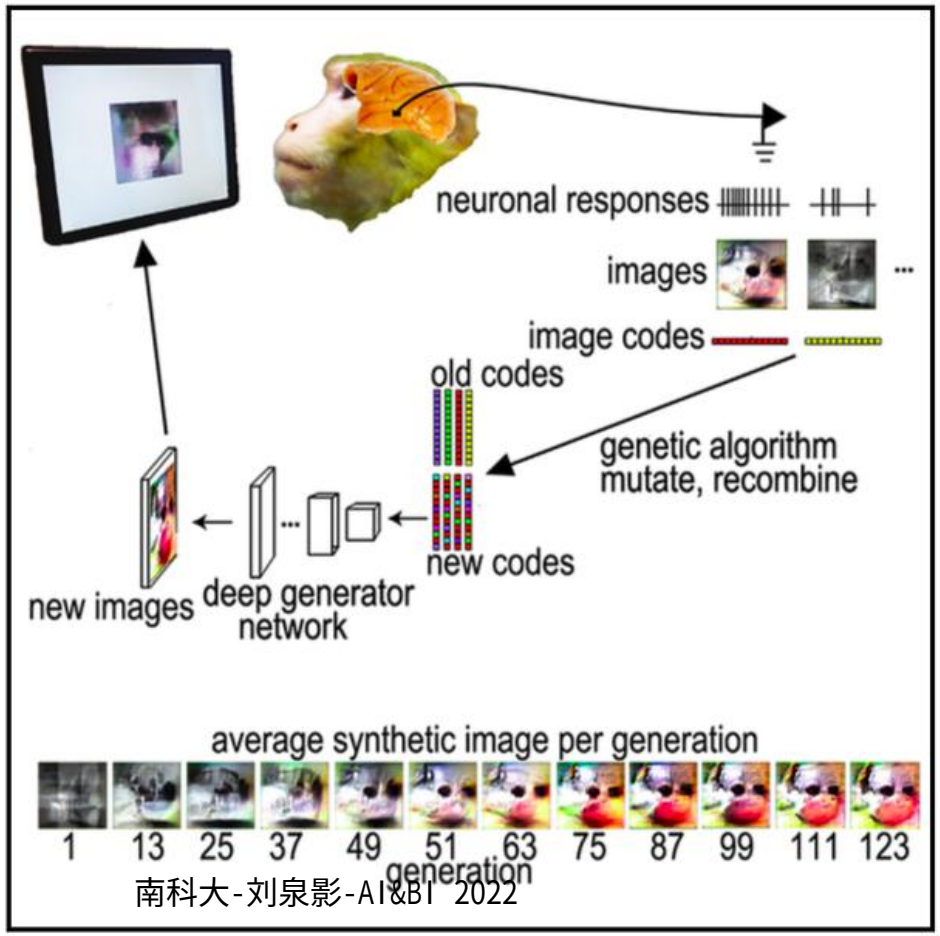
**Collection of images synthesized by a deep neural network model to control the activity of neural populations in primate cortical area V4.** We used a deep artificial neural network to control the activity pattern of a population of neurons in cortical area V4 of macaque monkeys by synthesizing visual stimuli that, when applied to the subject's retinae, successfully induced the experimenter-desired neural response patterns.

Bashivan, Pouya, Kohitij Kar, and James J. DiCarlo. "Neural population control via deep image synthesis." *Science* (2019)

Cell

# Evolving Images for Visual Neurons Using a Deep Generative Network Reveals Coding Principles and Neuronal Preferences

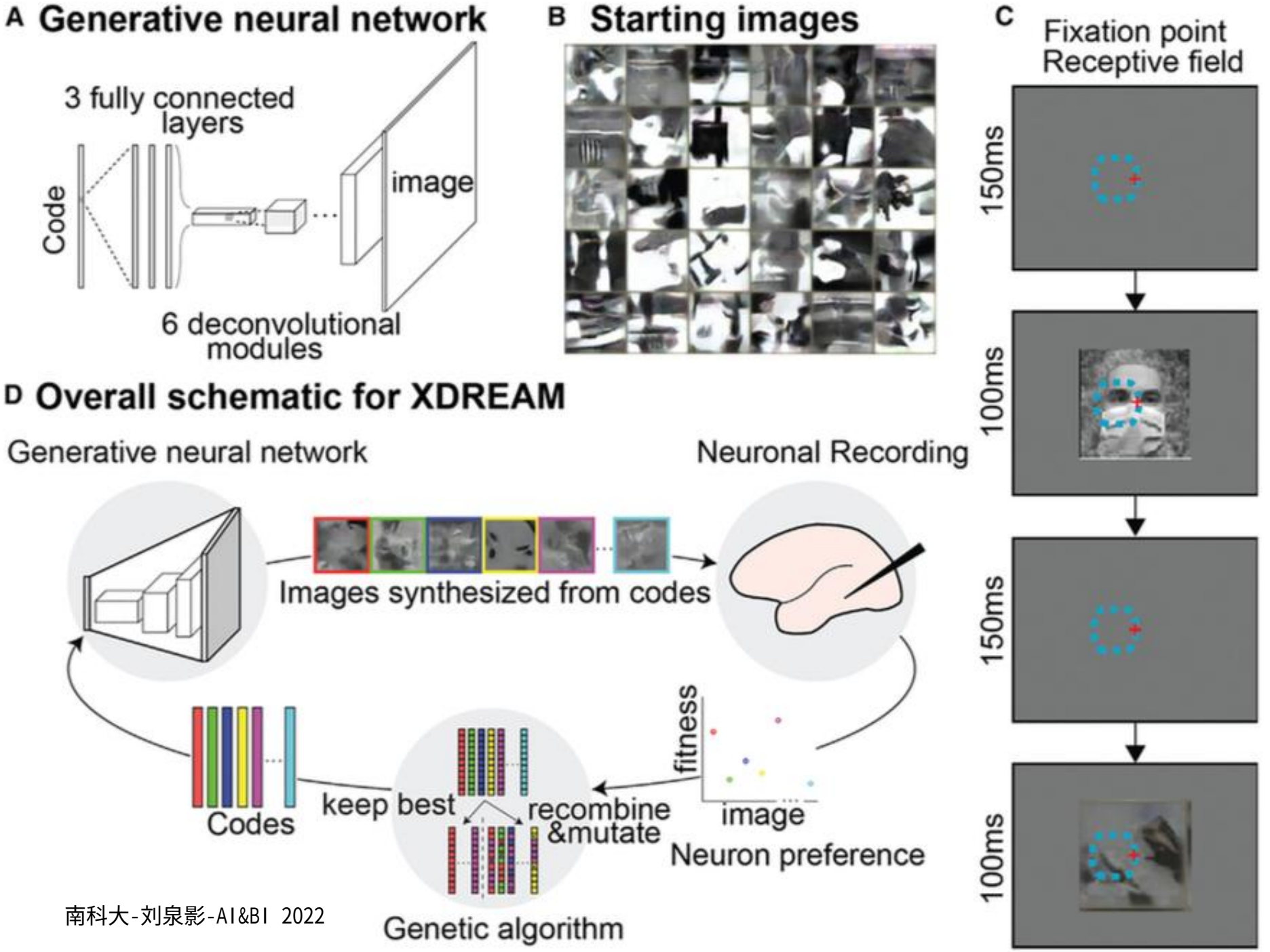
## Graphical Abstract




## Highlights

- A generative deep neural network and a genetic algorithm evolved images guided by neuronal firing
- Evolved images maximized neuronal firing in alert macaque visual cortex
- Evolved images activated neurons more than large numbers of natural images
- Similarity to evolved images predicts response of neurons to novel images

Ponce, Carlos R., et al. "Evolving images for visual neurons using a deep generative network reveals coding principles and neuronal preferences." *Cell* (2019)

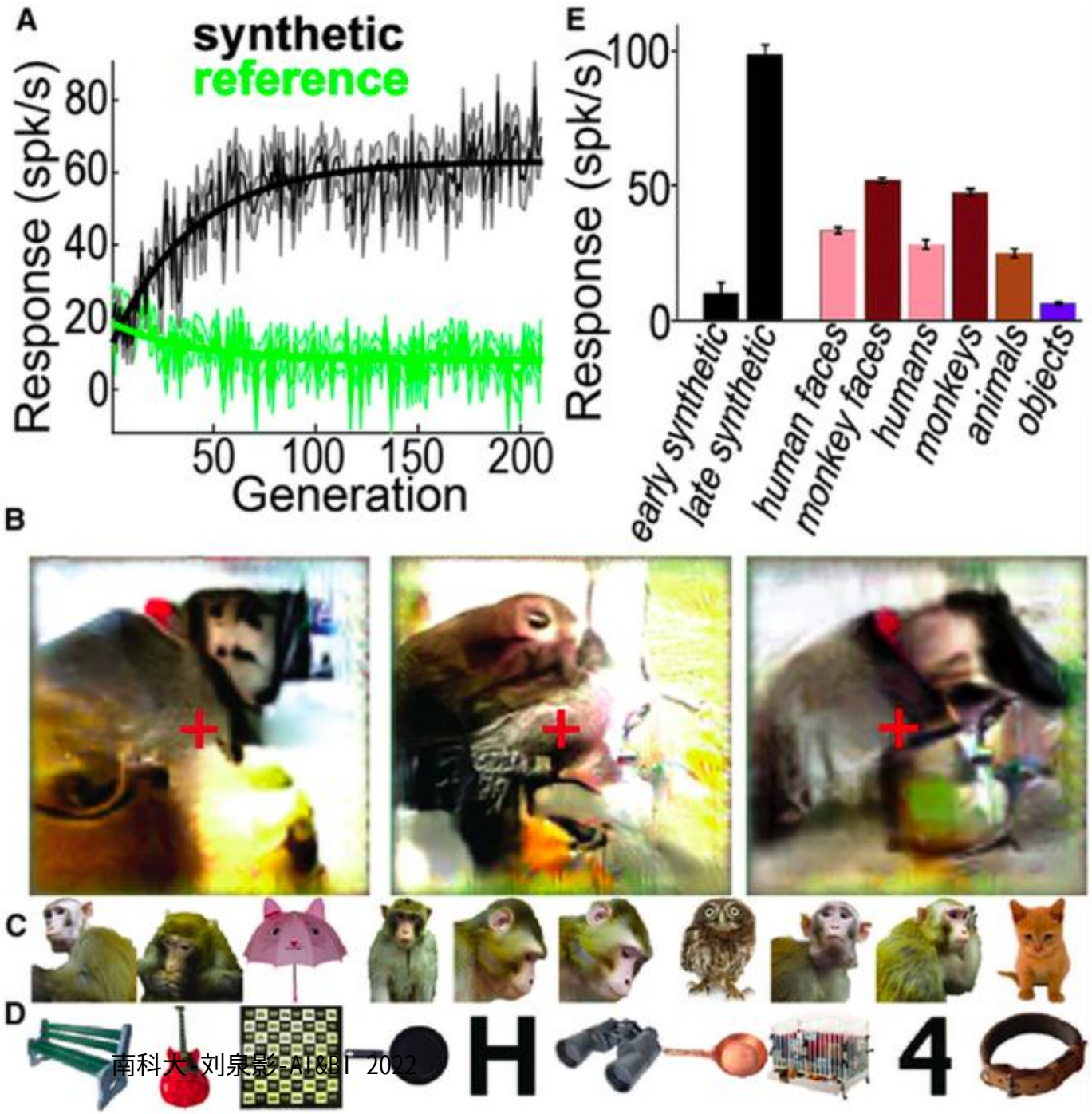



**A. pre-trained deep generative network**

**B. initial synthetic images**

**C. Behavioral task**

**D. Experimental flow**




A. Mean response to synthetic (black) and reference (green) images

E. the selectivity of this neuron to different image categories

B. Last-generation images evolved In 3 experiments

C. The top 10 images from this image set for this neuron

D. The worst 10 images

# Interactions between biological neurons & artificial neurons

- Information in biological neurons helps improve AI
- Information in AI can predict neural activity in the brain

普通DAE模型
图片重构任务的
baseline模型

CNN-FR模型
表征相似性任务
baseline模型

DAE-NR模型同
时兼具2个任
务的优化性能

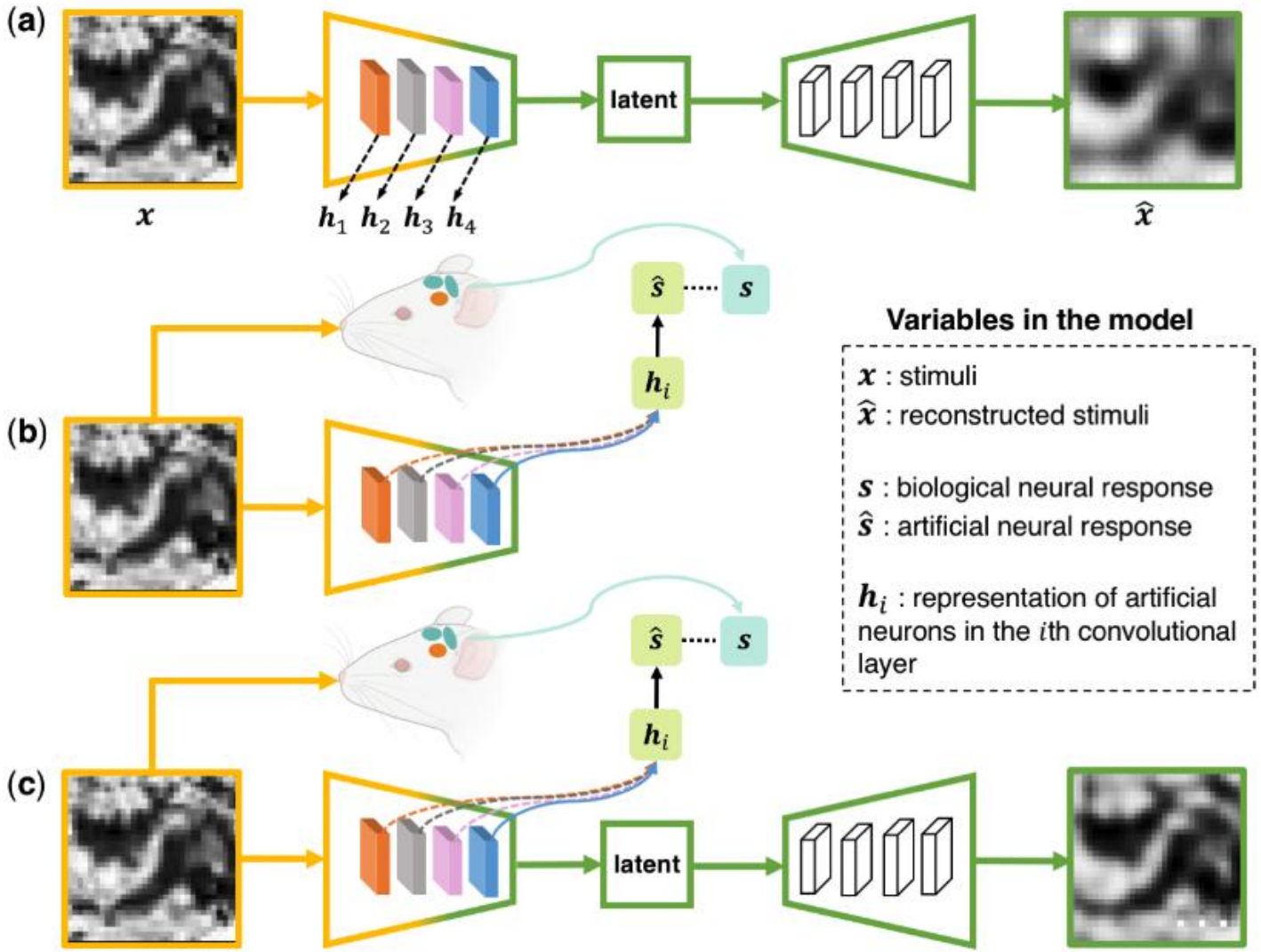



Figure 1: The illustration of the model of (a) the standard deep auto-encoder (DAE) for images reconstruction; (b) the convolutional neural network with factorized readout (CNN-FR) for prediction of neuron responses; (c) the DAE with the neuron response (DAE-NR) for images reconstruction and predictions of neuron responses. $\boldsymbol{s}$ is the biological neural response, the prediction of biological neural response is represented as $\hat{\boldsymbol{s}}$, and $\mathbf{h}_i$ ($i \in \{1, 2, 3, 4\}$) is the feature of the $i$th convolutional layer.

冉旭明
NCC lab

The full article is in arXiv.
https://arxiv.org/pdf/2111.15309.pdf

# 重构的图片

(a)原始图片；

(b)普通DAE；

(c-f)DAE-NR，将神经信息导入DAE的1-4层

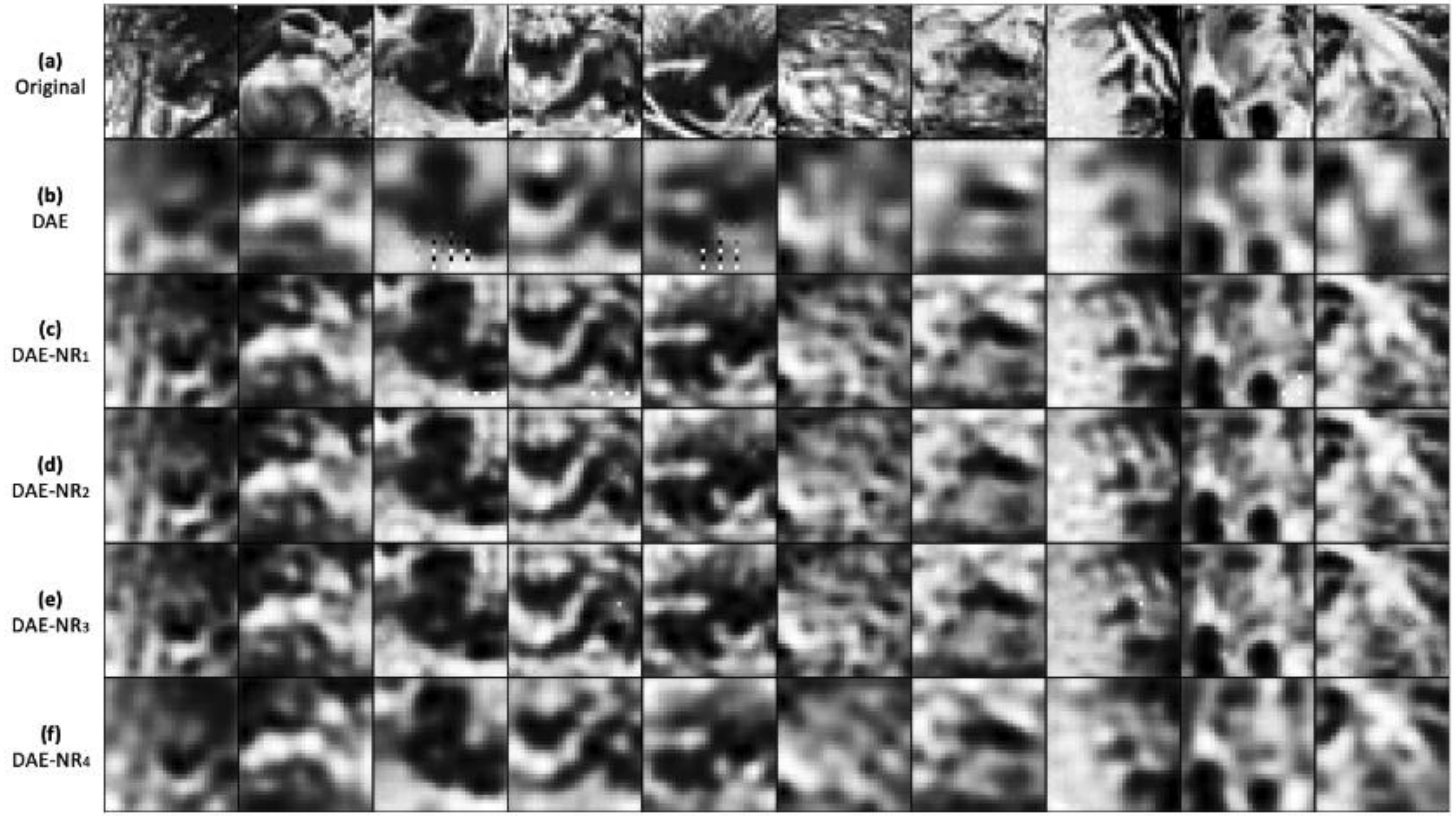



Figure 2: The reconstructed images with neurons in Region 3. From top to bottom, each row displays the original images (a), the images reconstructed by DAE (b), DAE-$NR_1$ (c), DAE-$NR_2$ (d), DAE-$NR_3$ (e), DAE-$NR_4$ (f), respectively

# 图片重构任务的量化结果

DAE-NR能提高图片重构的质量

| | Region 1 | | | Region 2 | | | Region 3 | | |
|---|---|---|---|---|---|---|---|---|---|
| Model | MSE↓ | PSNR↑ | SSIM↑ | MSE↓ | PSNR↑ | SSIM↑ | MSE↓ | PSNR↑ | SSIM↑ |
| DAE | 0.022 | 23.709 | 0.771 | 0.024 | 23.338 | 0.754 | 0.081 | 17.039 | 0.561 |
| DAE-$NR_1$ | **0.021** | **23.829** | **0.776** | **0.023** | 23.392 | 0.753 | 0.044 | 19.751 | 0.763 |
| DAE-$NR_2$ | **0.021** | 23.779 | 0.775 | **0.023** | 23.440 | 0.759 | **0.043** | **19.819** | **0.764** |
| DAE-$NR_3$ | **0.021** | 23.778 | 0.775 | 0.024 | 23.330 | 0.755 | **0.043** | 19.789 | 0.761 |
| DAE-$NR_4$ | 0.022 | 23.721 | 0.773 | **0.023** | **23.491** | **0.760** | 0.059 | 18.462 | 0.668 |

与人工神经元的表征显著相关的大脑神经元，能促进**DAE-NR**的图片重构性能

非显著相关的大脑神经元，则不能帮助提升图片重构

| | MSE↓ | | SSIM↑ | | PSNR↑ | |
|---|---|---|---|---|---|---|
| Significant | YES | NO | YES | NO | YES | NO |
| DAE-NR$_1$ | **0.043** | 0.125 | **0.761** | 0.332 | **19.784** | 15.168 |
| DAE-NR$_2$ | **0.047** | 0.082 | **0.743** | 0.547 | **19.467** | 16.970 |
| DAE-NR$_3$ | **0.049** | 0.116 | **0.724** | 0.362 | **19.245** | 15.463 |
| DAE-NR$_4$ | 0.047 | **0.045** | 0.740 | **0.752** | 19.497 | **19.628** |

**DAE-NR能使AI和BI之间有更强的神经表征相似性**

与人工神经元的表征显著相关的大脑神经元的数量，与**DAE (6个）**相比，**DAE-NR （38-47个）**中数量增多了。

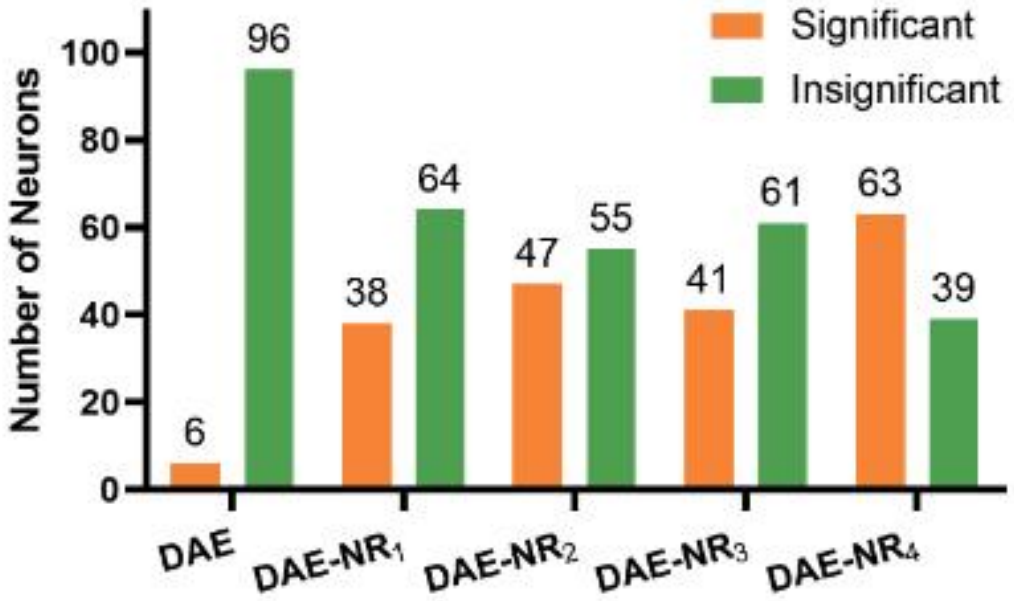



Figure 3: The number of significant neurons and insignificant neurons of region 3 in the image reconstruction experiments. The threshold for significance is $p \leq 0.05$.

# Summary of Lecture 4 – What does AI/ brain learn?

- Memory size & parameter size of CNN
  - how to calculate the memory size and parameter size
  - LeNet, AlexNet, VGG-16
- What do neurons in CNN learn?
  - visualize the activation of neurons
  - visualize the weights of a filter
  - synthesize image that maximize neural activity
  - Some existing tools
- What do neurons in brain learn?
  - Synthesize images that maximize V4 neurons
- Interactions between biological neurons & artificial neurons