THE UNIVERSITY OF BRITISH COLUMBIA

Topics in AI (CPSC 532L):
Multimodal Learning with Vision, Language and Sound

Lecture 6

Course **Logistics**

— Assignment 1 grades (available on **Connect**)

— Assignment 2 now **due 11:59pm Wednesday**

— Assignment 3 planned to be out on **Thursday, January 25th**

— Paper choices will be due **next week**

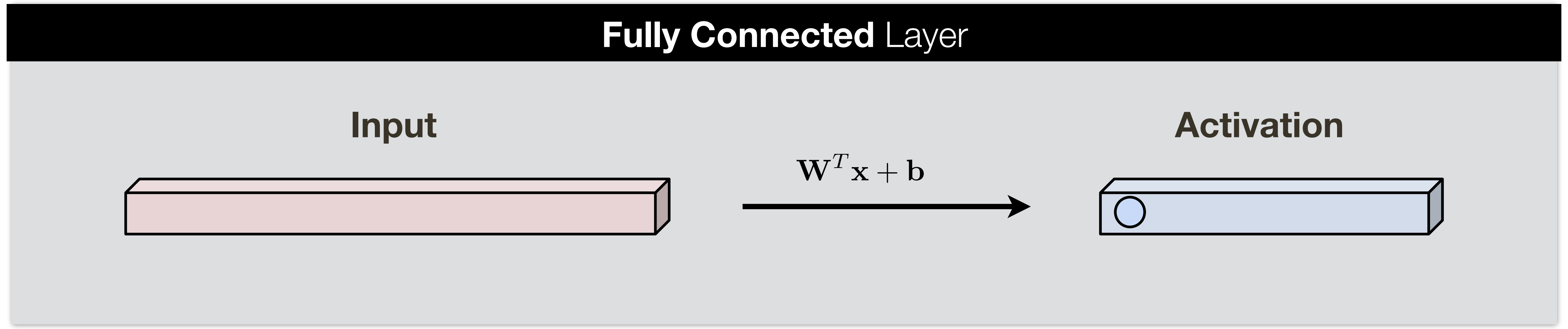
— **Office hours** today at 1:30-2:30pm in X860

— **Project pitching** on class

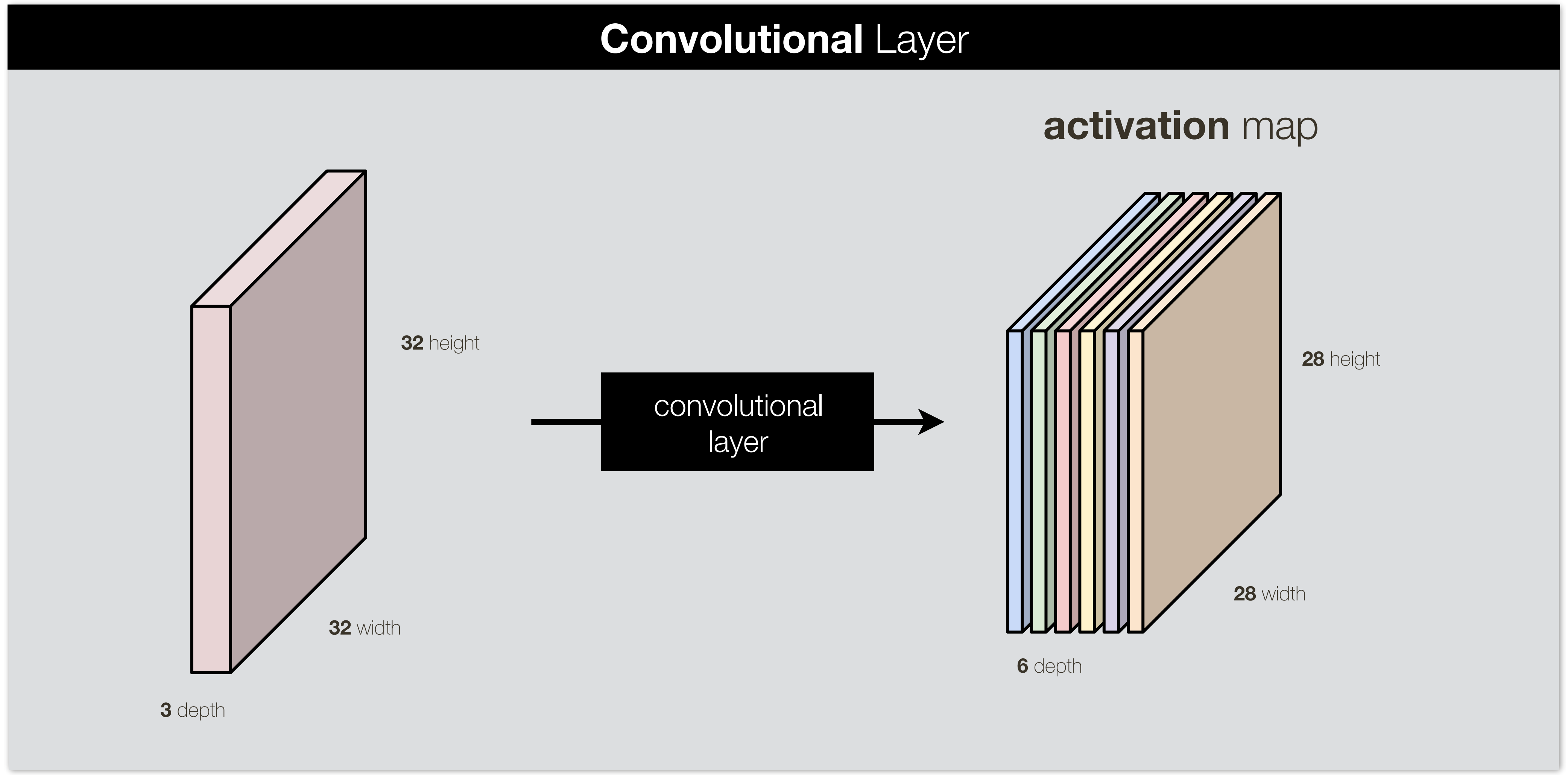
— Computer **vision reading group** Fridays 2-3pm

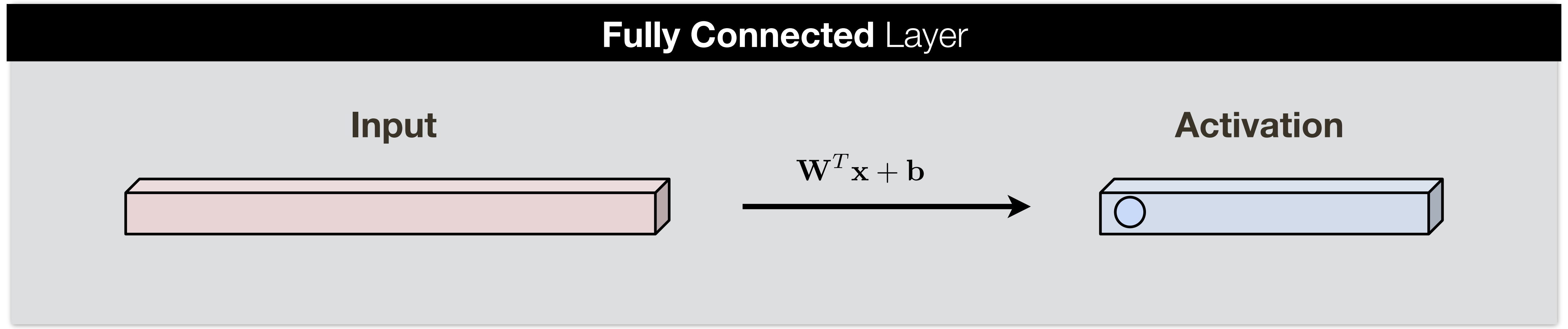
Review of **CNNs**

Review of **CNNs**



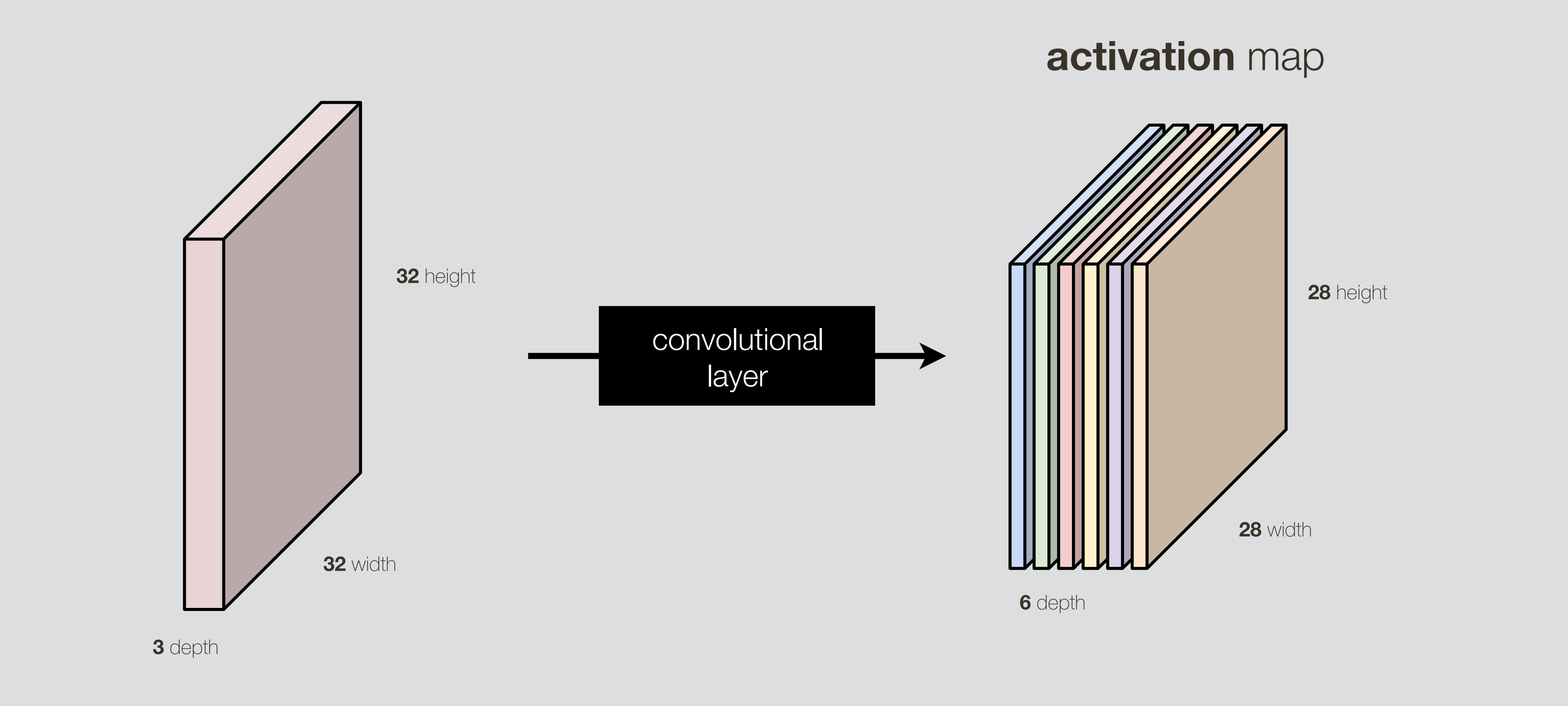
Review of **CNNs**



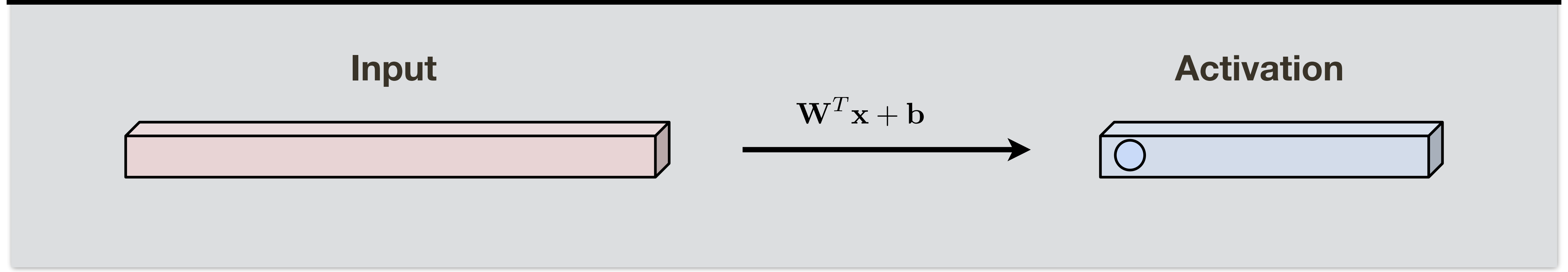


Review of **CNNs**

Convolutional Layer



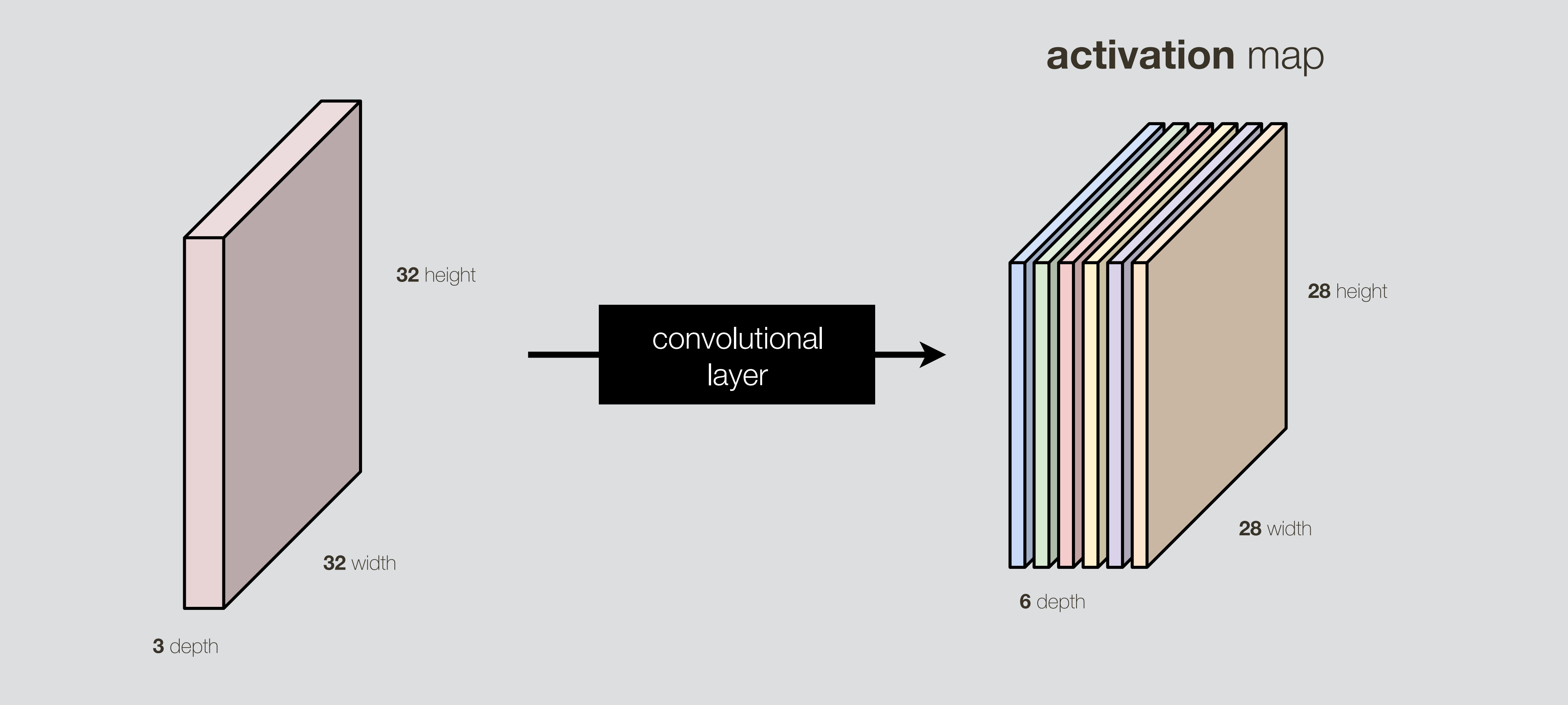
Fully Connected Layer



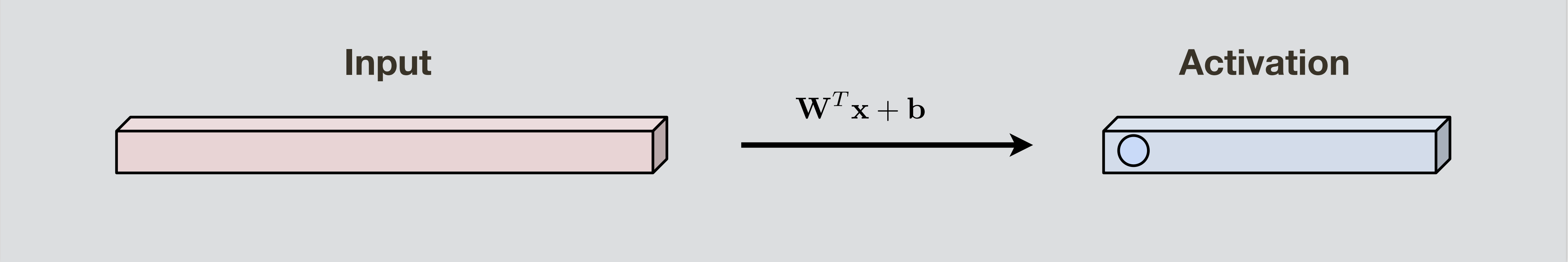
Pooling Layer

Review of **CNNs**

Convolutional Layer



Fully Connected Layer



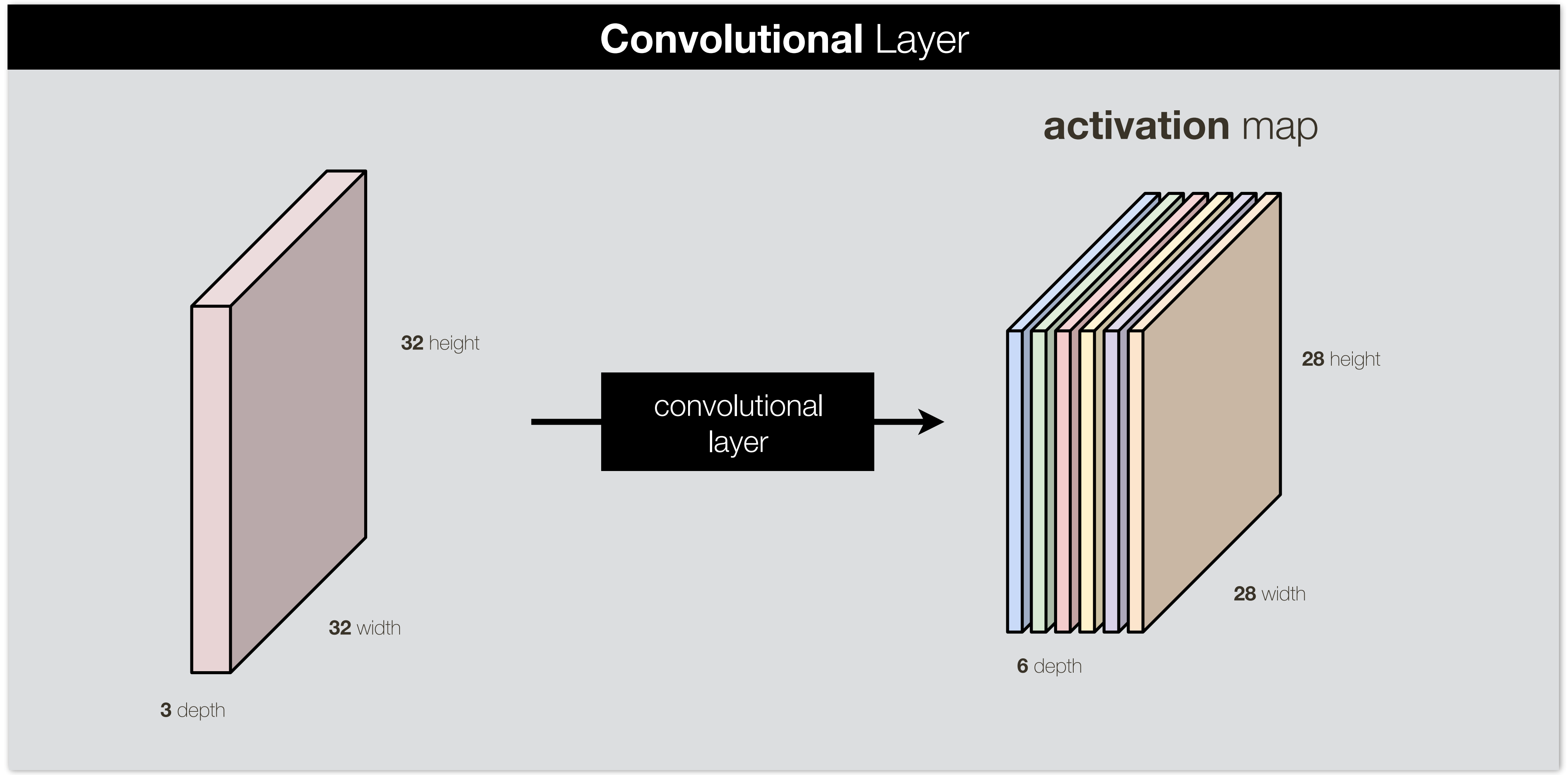
Pooling Layer

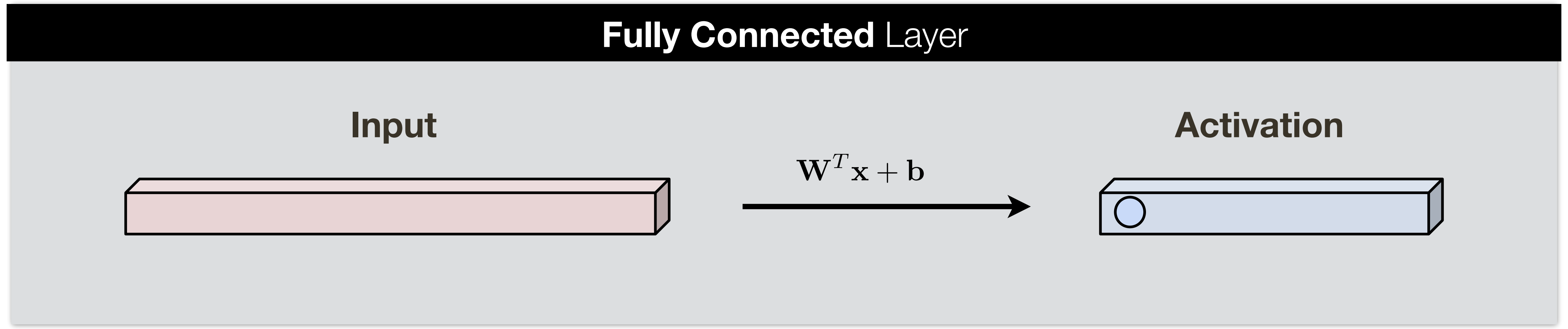
Effective Techniques for **Training**

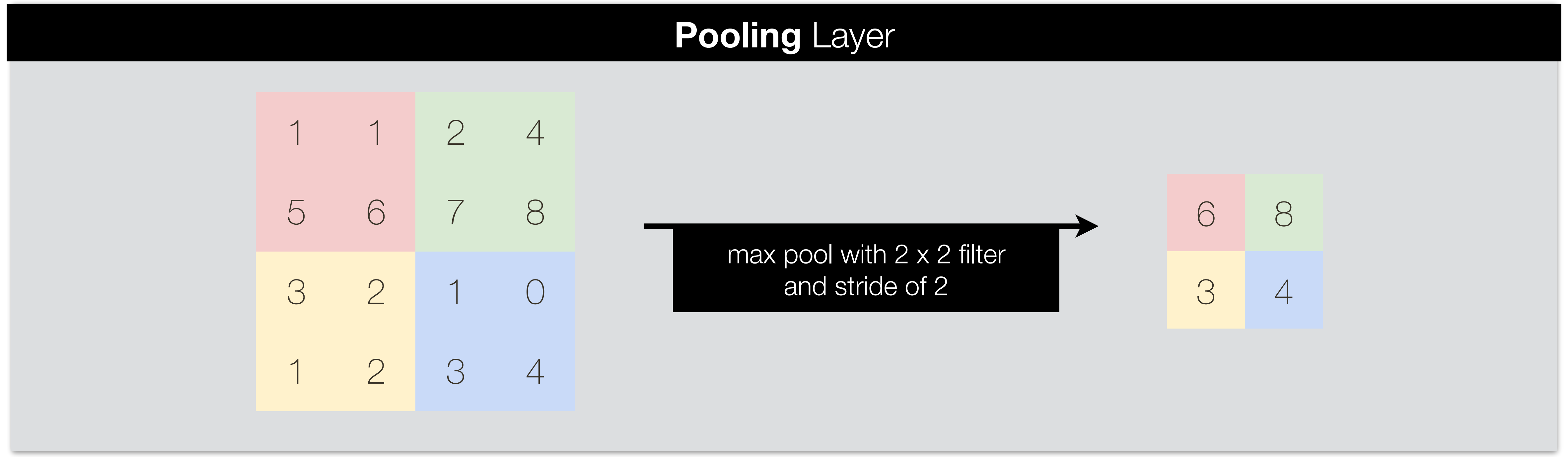
— **Regularization:** L1, L2, data augmentation

— **Transfer Learning:** fine-tuning networks

Review of **CNNs**







Effective Techniques for **Training**

— **Regularization:** L1, L2, data augmentation

— **Transfer Learning:** fine-tuning networks

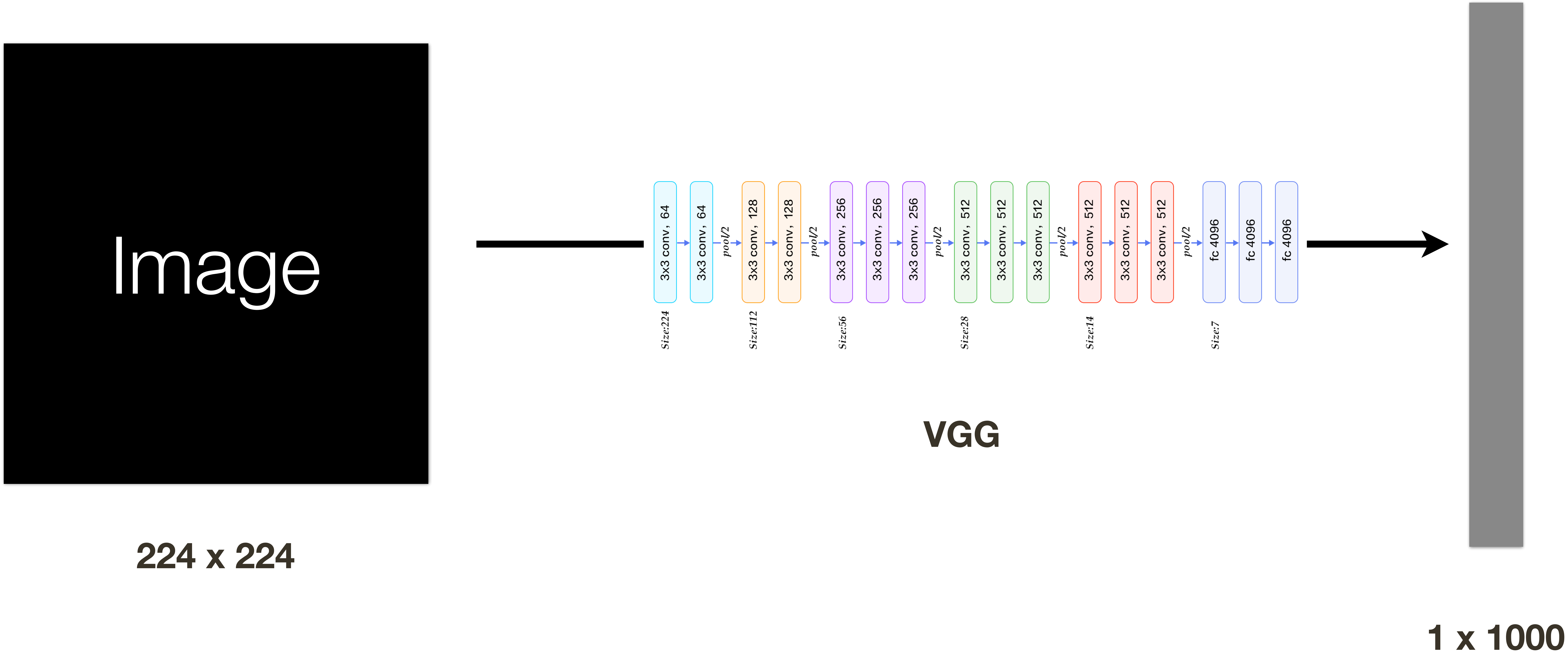
Vision **Applications** of CNNs

— **Classification:** AlexNet, VGG, GoogleLeNet, ResNet

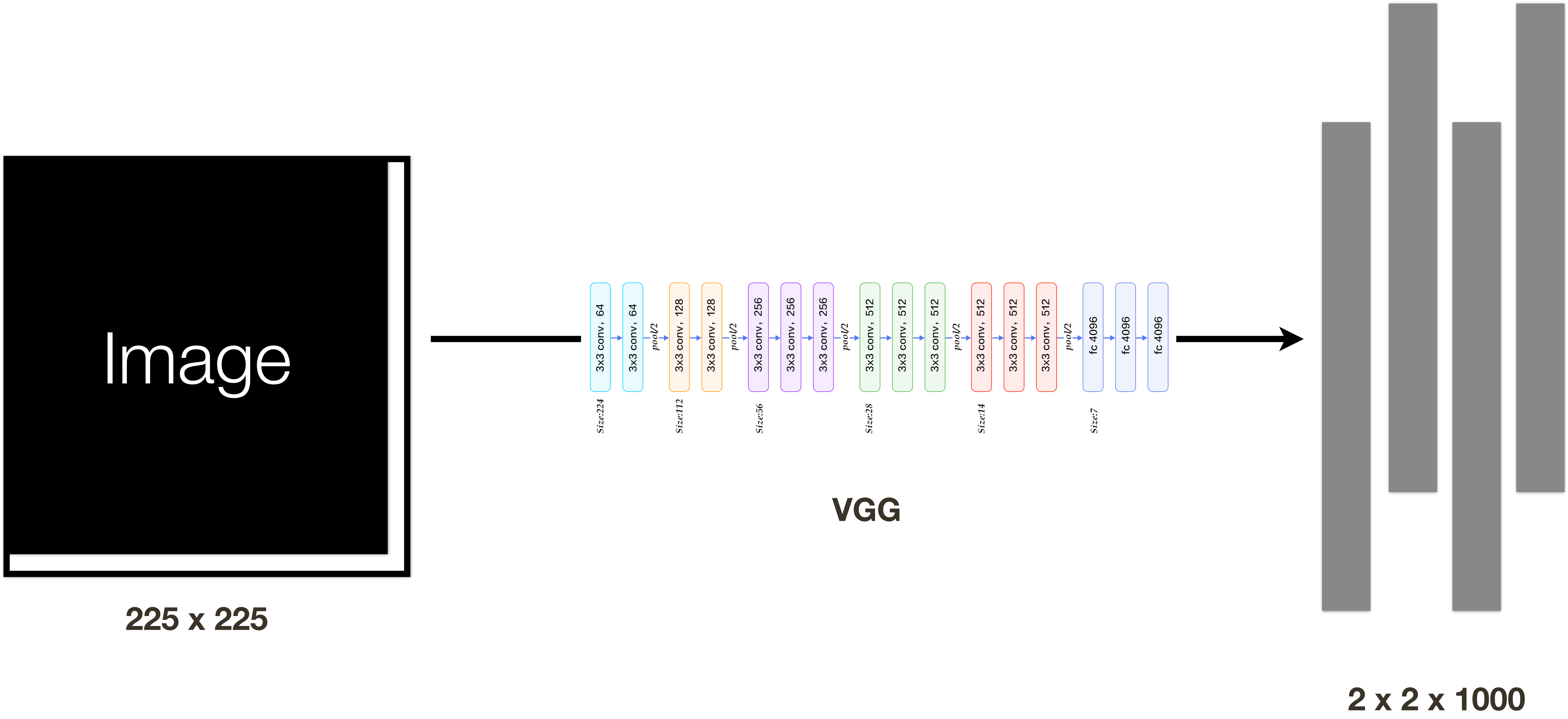
— **Segmentation:** Fully convolutional CNNs

— **Detection:** R-CNN, Fast R-CNN, Faster R-CNN, YOLO

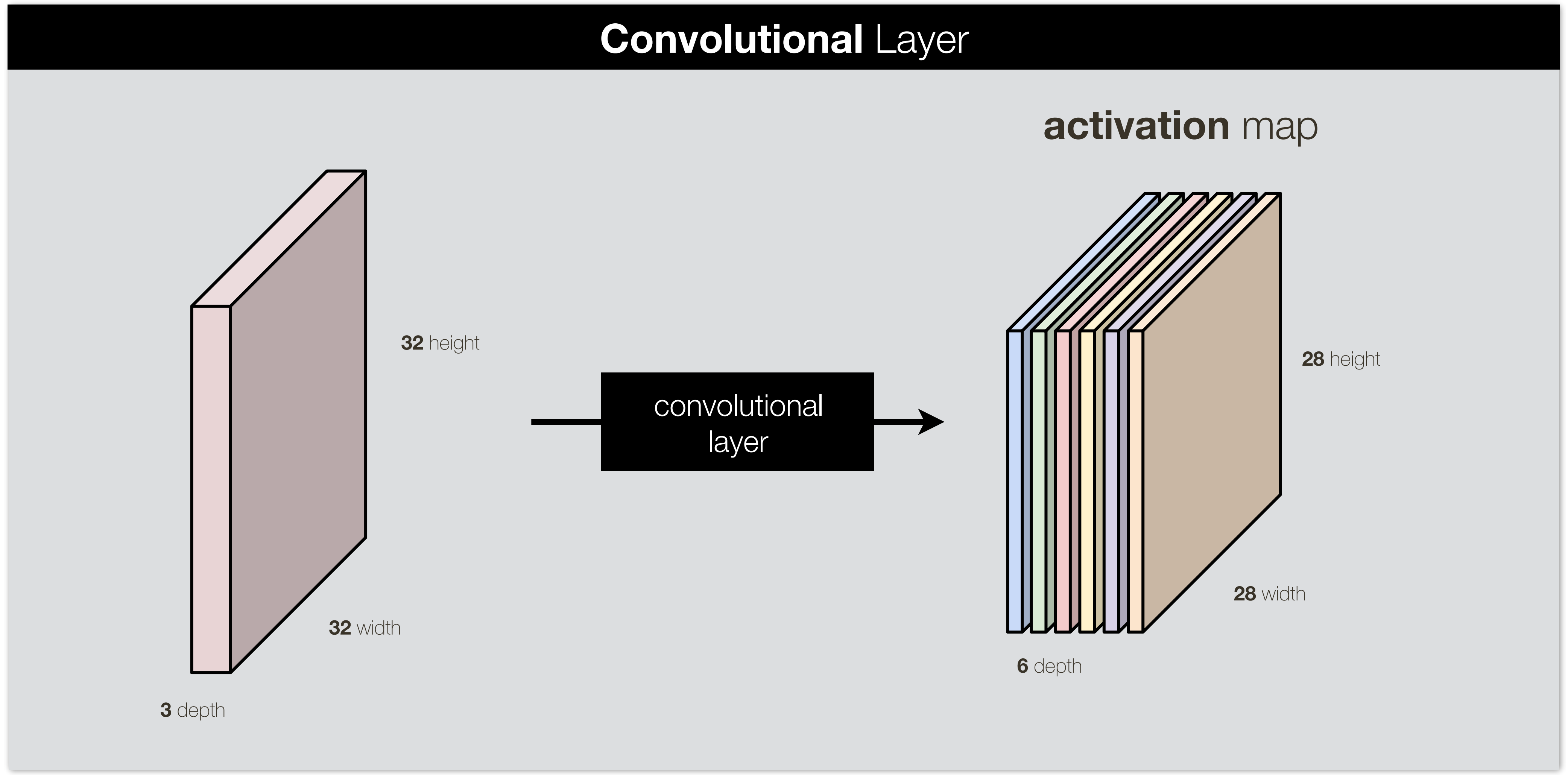
Any **CNN** Could be **Fully Convolutional**

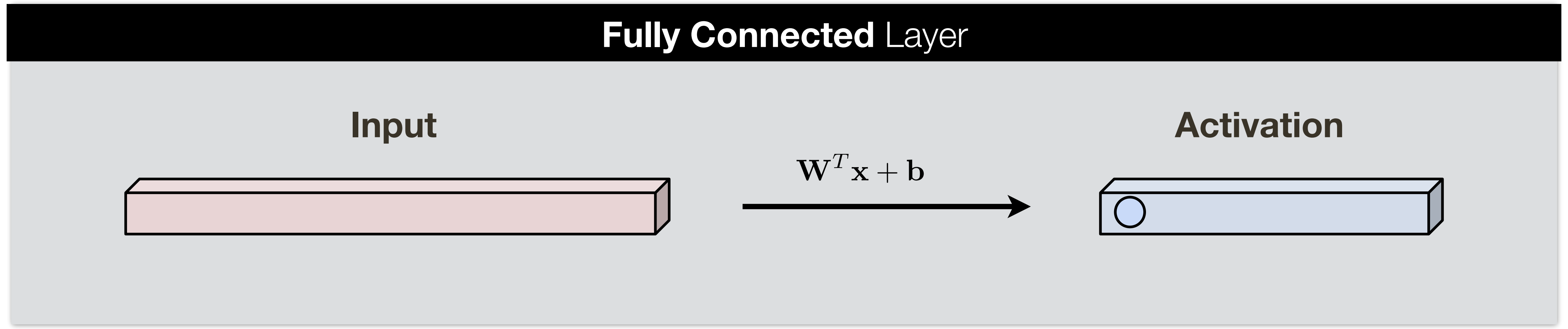


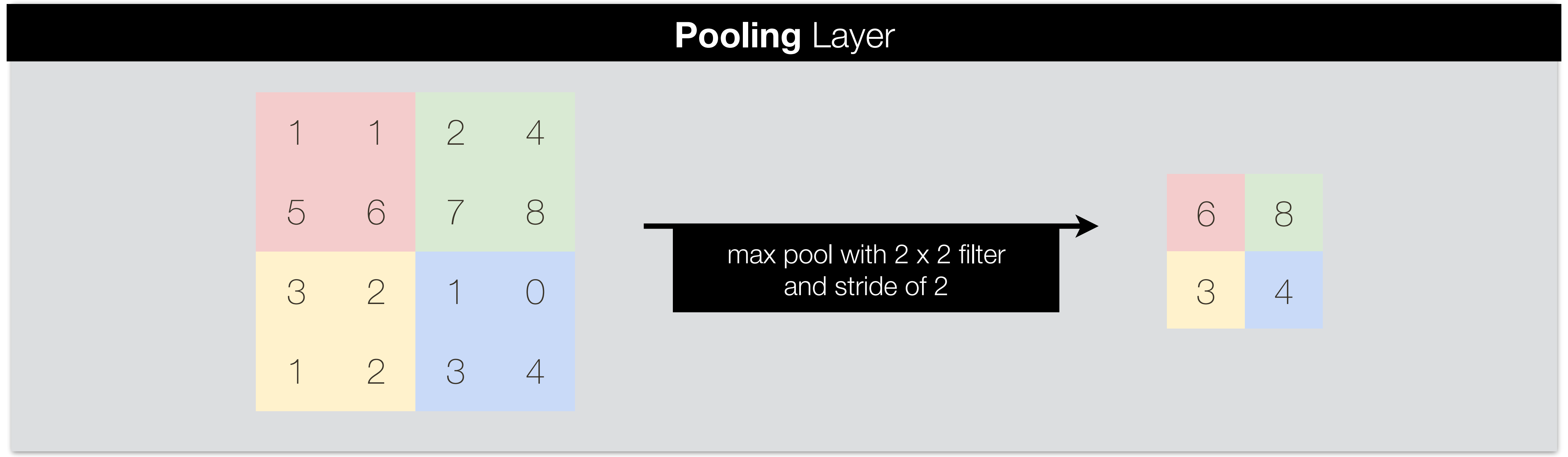
Any **CNN** Could be **Fully Convolutional**



Review of **CNNs**







Effective Techniques for **Training**

— **Regularization:** L1, L2, data augmentation

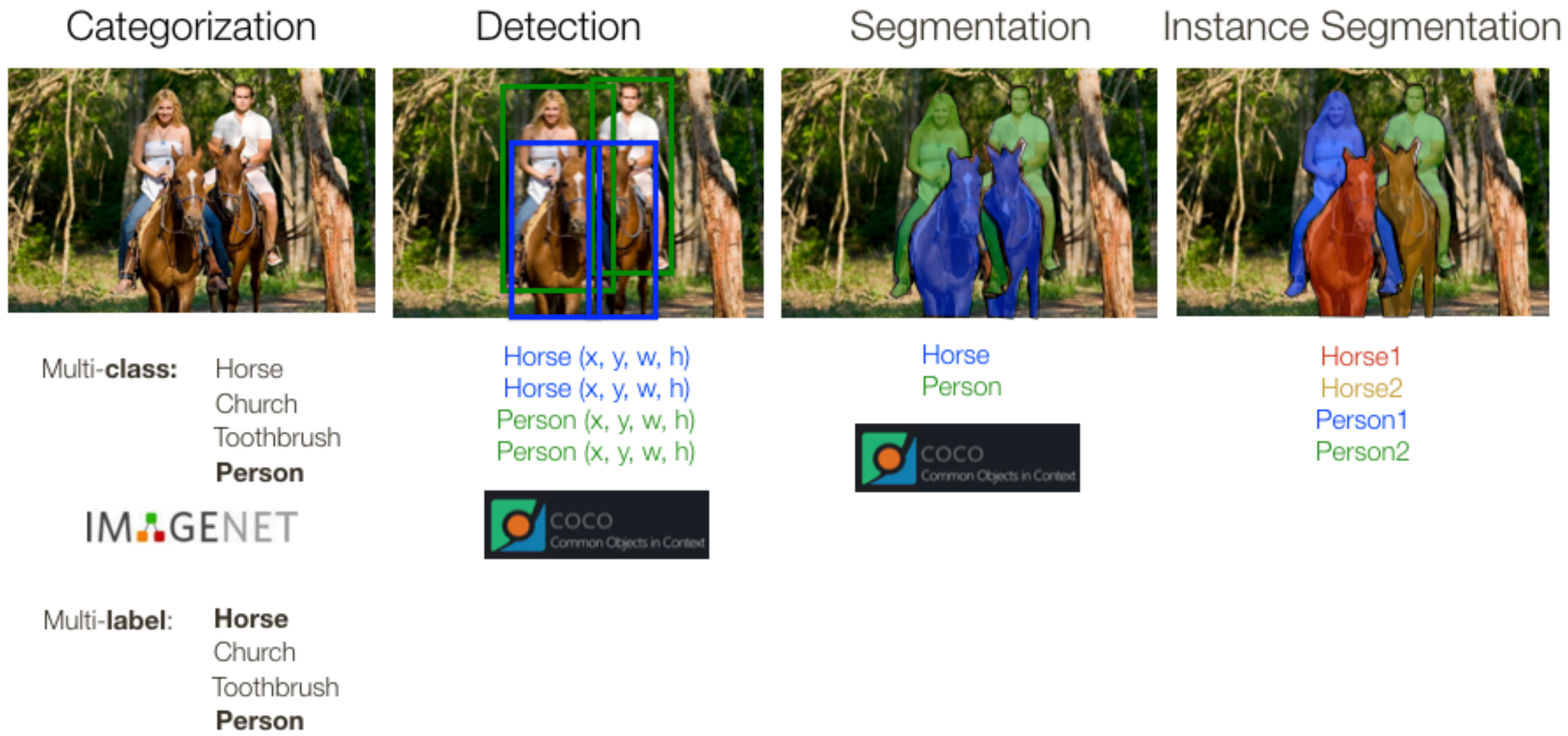
— **Transfer Learning:** fine-tuning networks

Vision **Applications** of CNNs

— **Classification:** AlexNet, VGG, GoogleLeNet, ResNet

— **Segmentation:** Fully convolutional CNNs

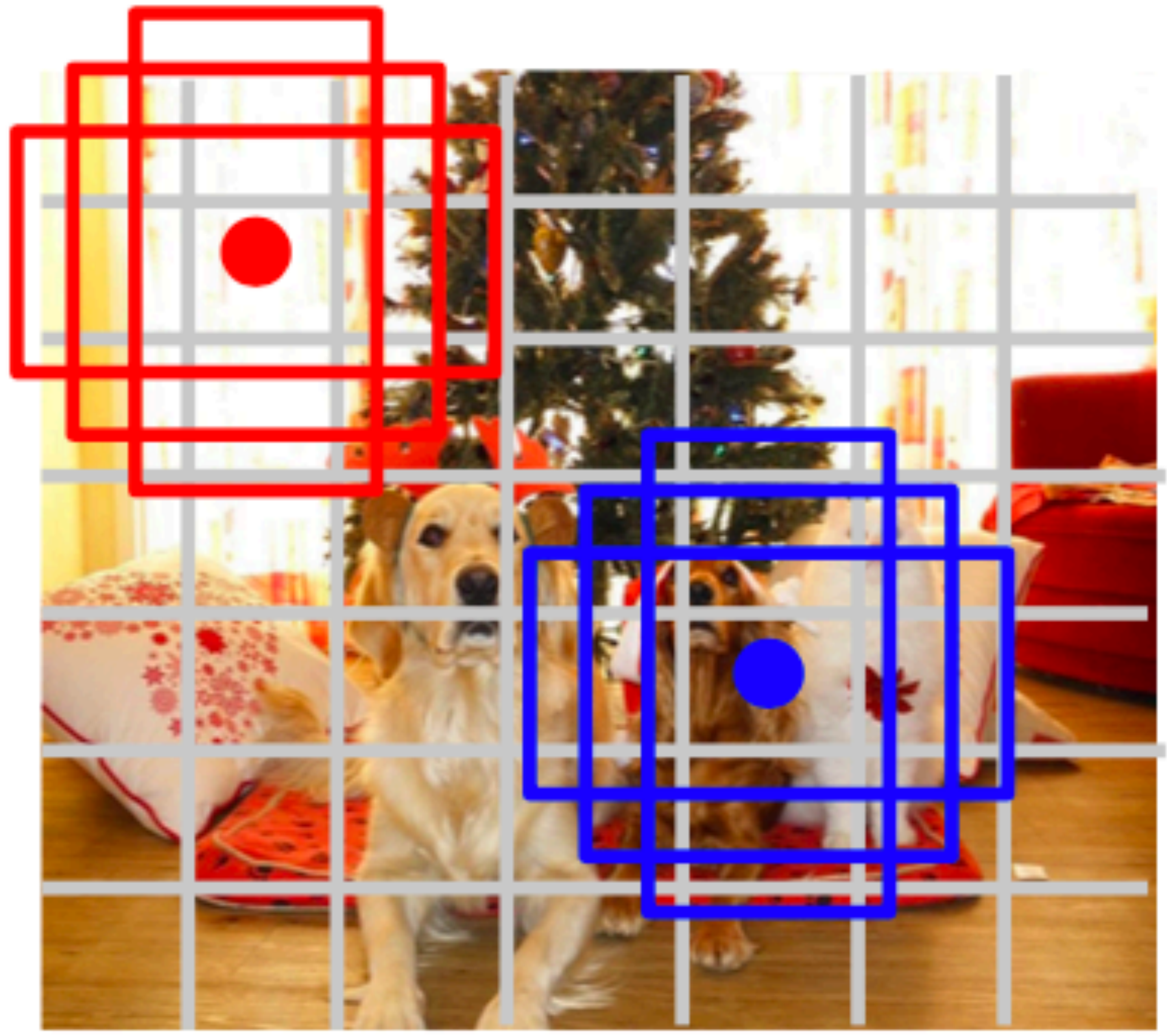
— **Detection:** R-CNN, Fast R-CNN, Faster R-CNN, YOLO



YOLO: You Only Look Once

[Redmon et al, CVPR 2016]

Input image
3 x H x W



Divide image into grid
7 x 7

Image a set of **base boxes** centered at each grid cell
Here B = 3

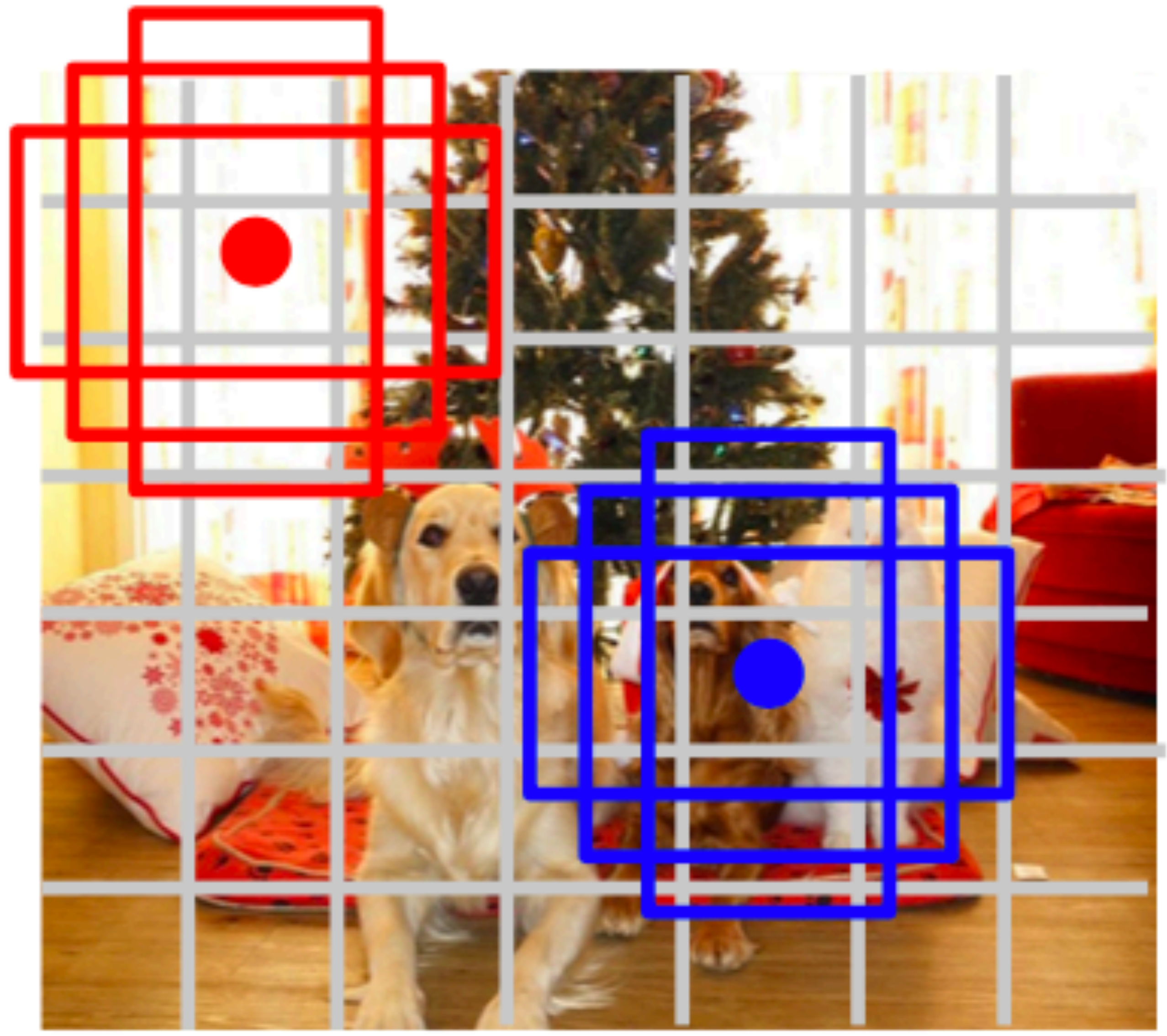
Within each grid cell:
- Regress from each of the B base boxes to a final box with 5 numbers: (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)

Output:
7 x 7 x (5 * B + C)

YOLO: You Only Look Once

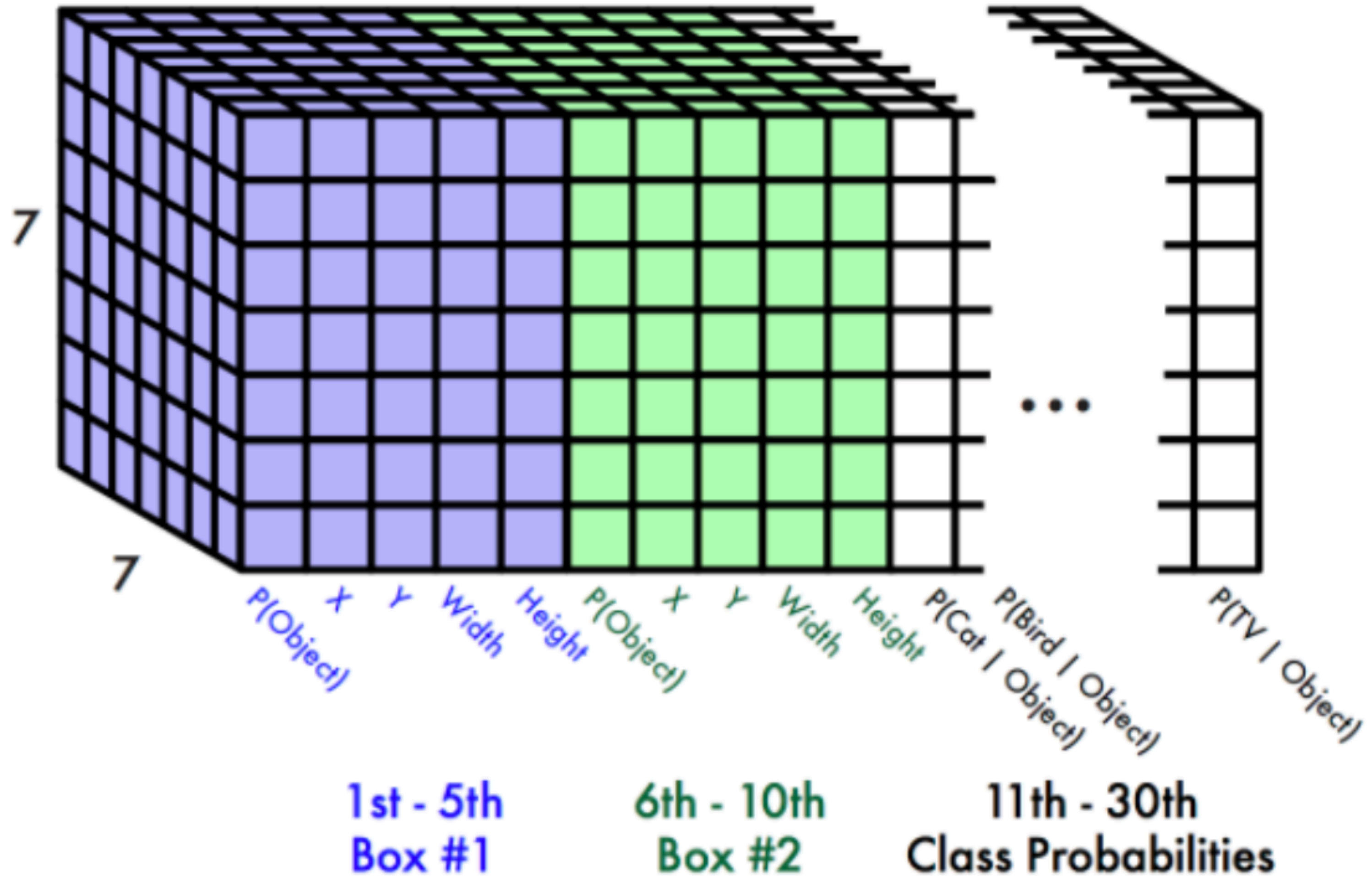
[Redmon et al, CVPR 2016]

Input image
3 x H x W



Divide image into grid
7 x 7

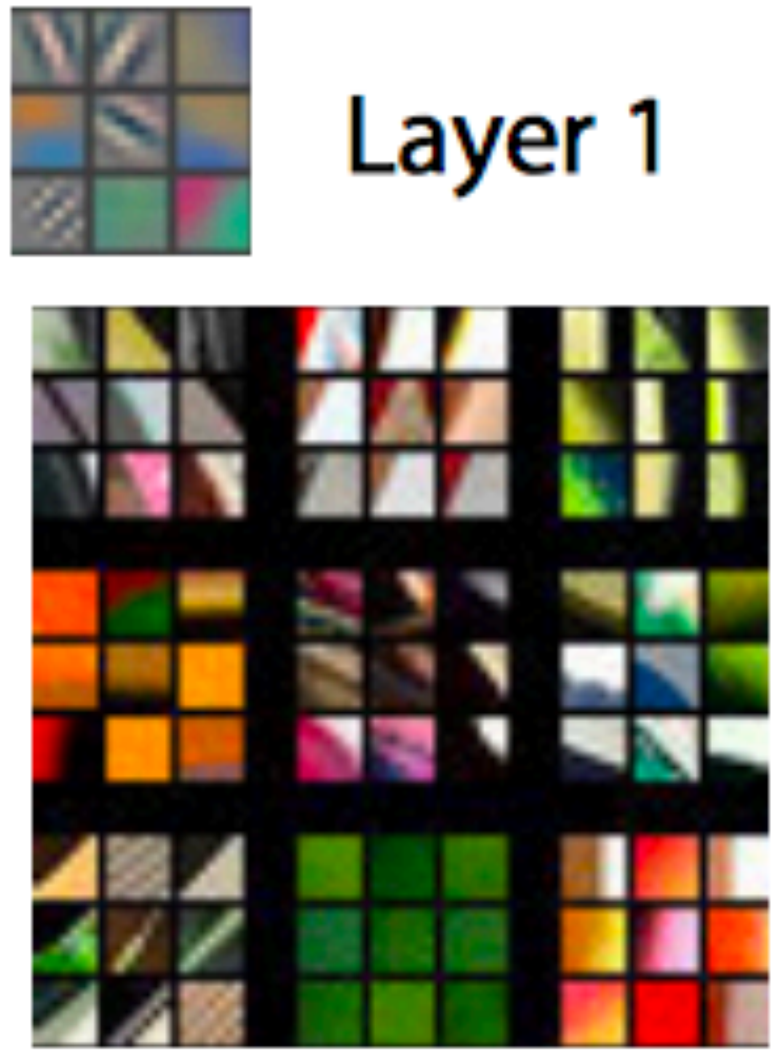
Image a set of **base boxes** centered at each grid cell
Here B = 3

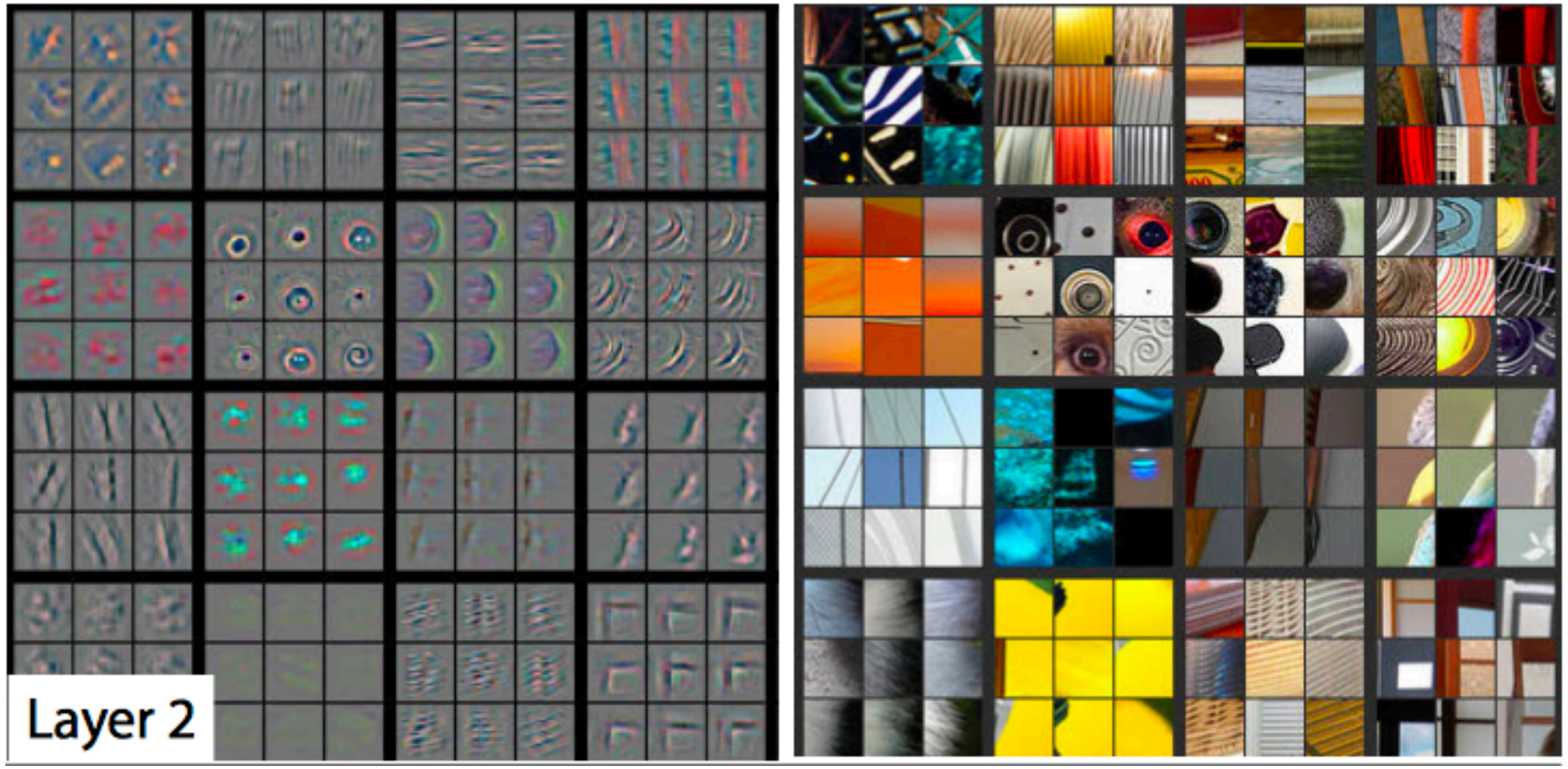


Topics in AI (CPSC 532L):
Multimodal Learning with Vision, Language and Sound

Lecture 6: Visualizing CNNs

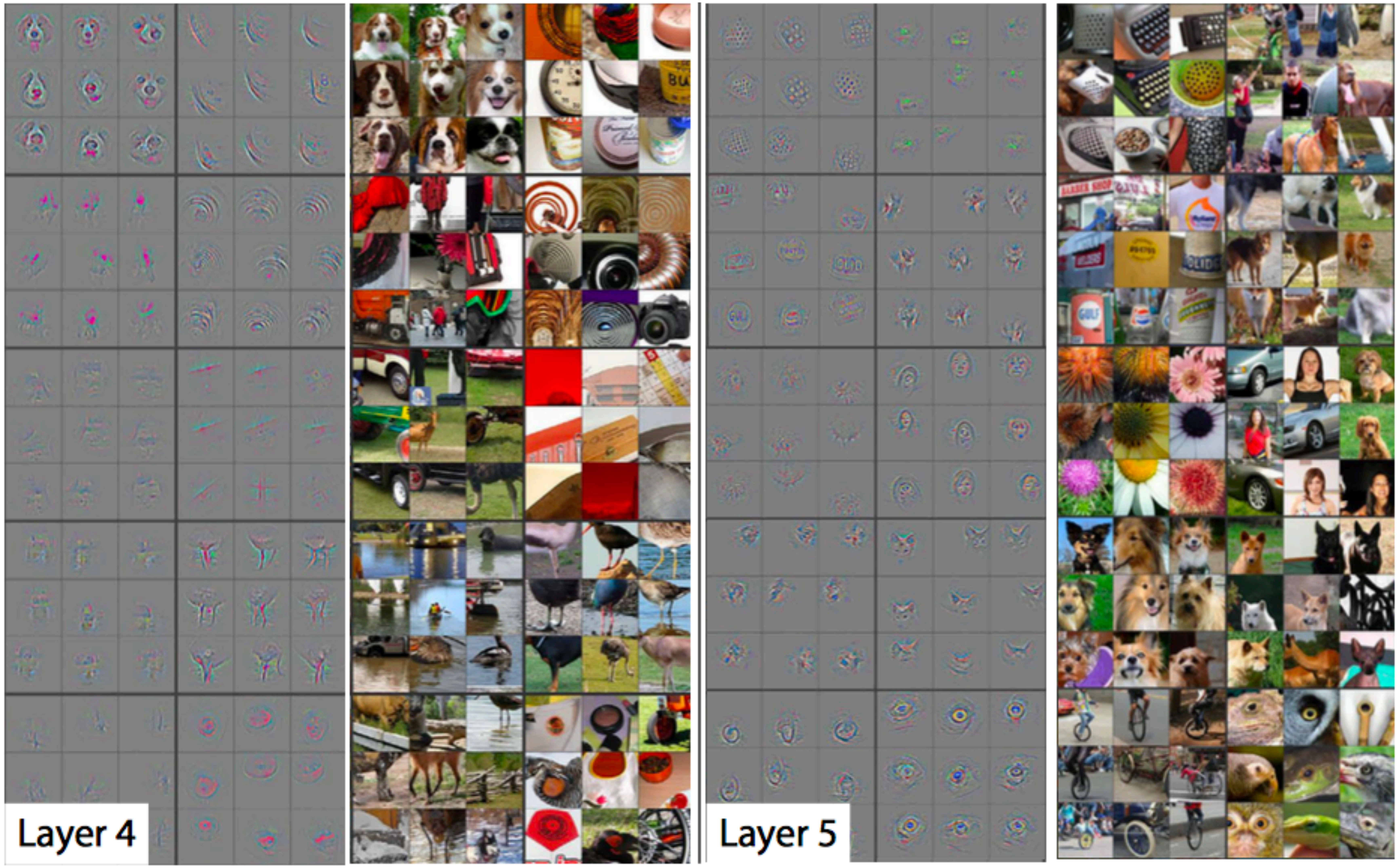
Recall …





[Zeiler and Fergus, 2013]

Recall …



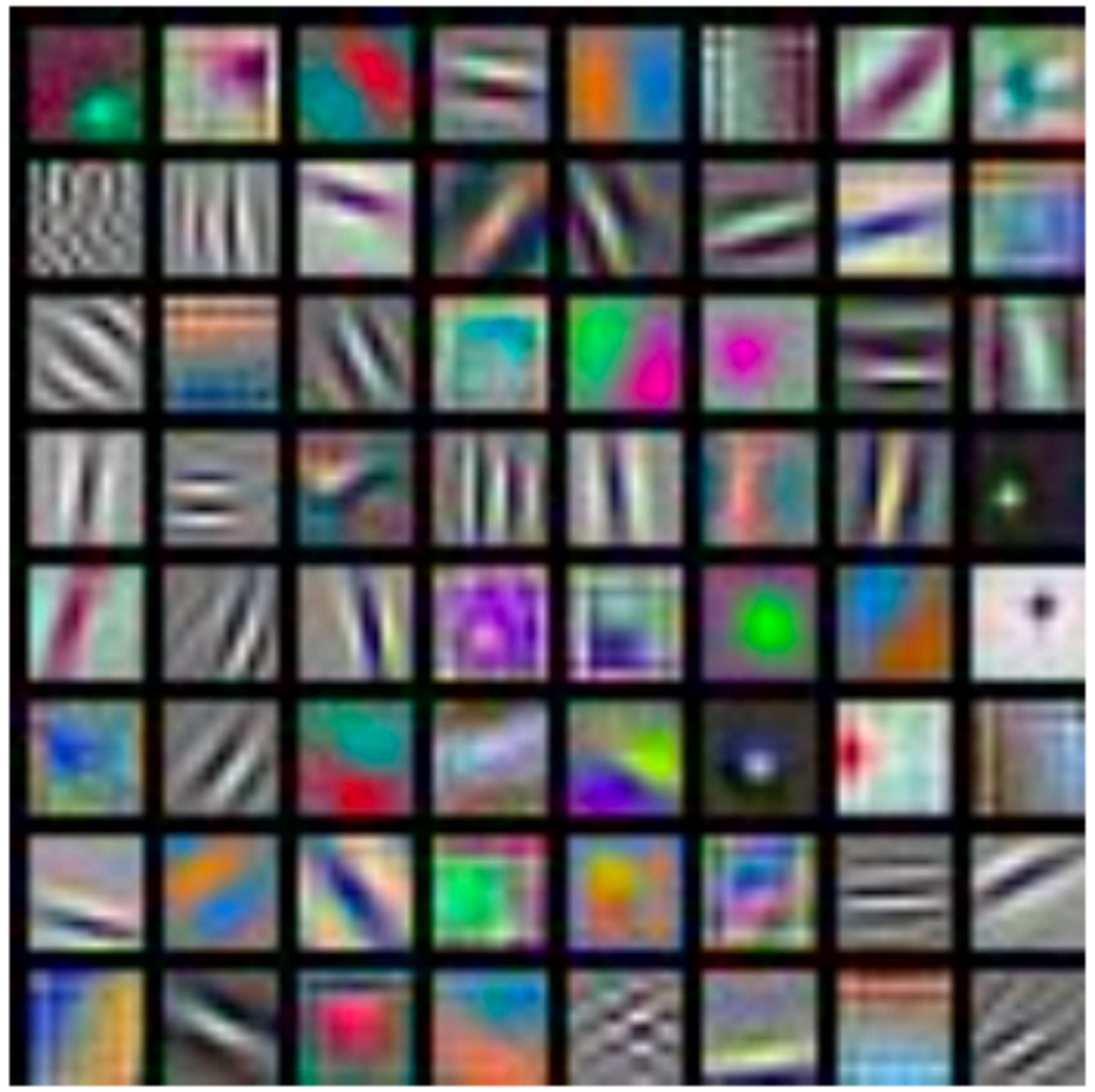
[Zeiler and Fergus, 2013]

Motivation …

CNNs are big black boxes, lets get some intuition for how and why they work

First Layer Filters …

Directly **visualize filters** (only works for the first layer)



AlexNet:
64 x 3 x 11 x 11

ResNet-18:
64 x 3 x 7 x 7

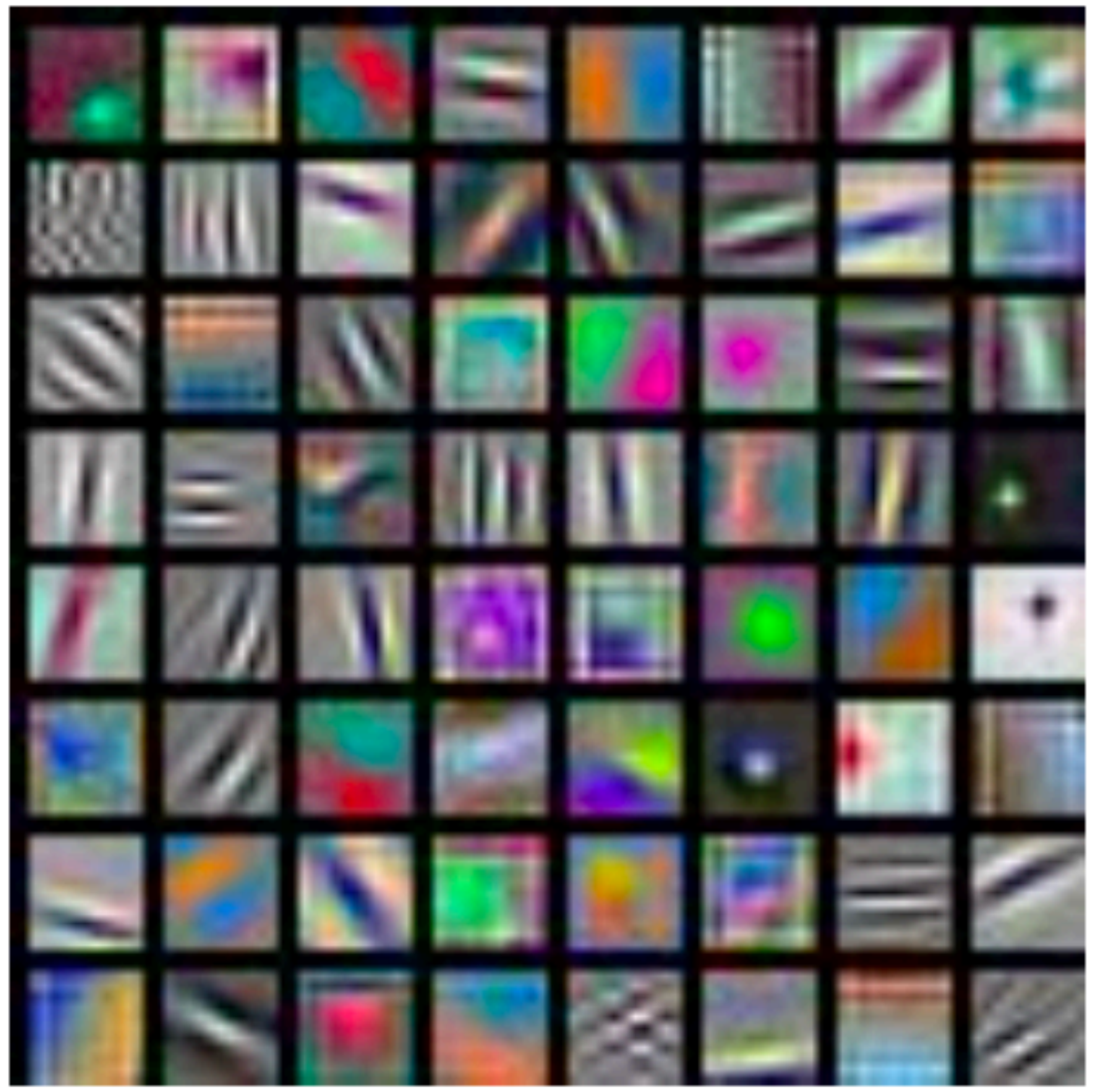
ResNet-101:
64 x 3 x 7 x 7

DenseNet-121:
64 x 3 x 7 x 7

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

First Layer Filters …

Directly **visualize filters** (only works for the first layer)



AlexNet:
64 x 3 x 11 x 11

ResNet-18:
64 x 3 x 7 x 7

ResNet-101:
64 x 3 x 7 x 7

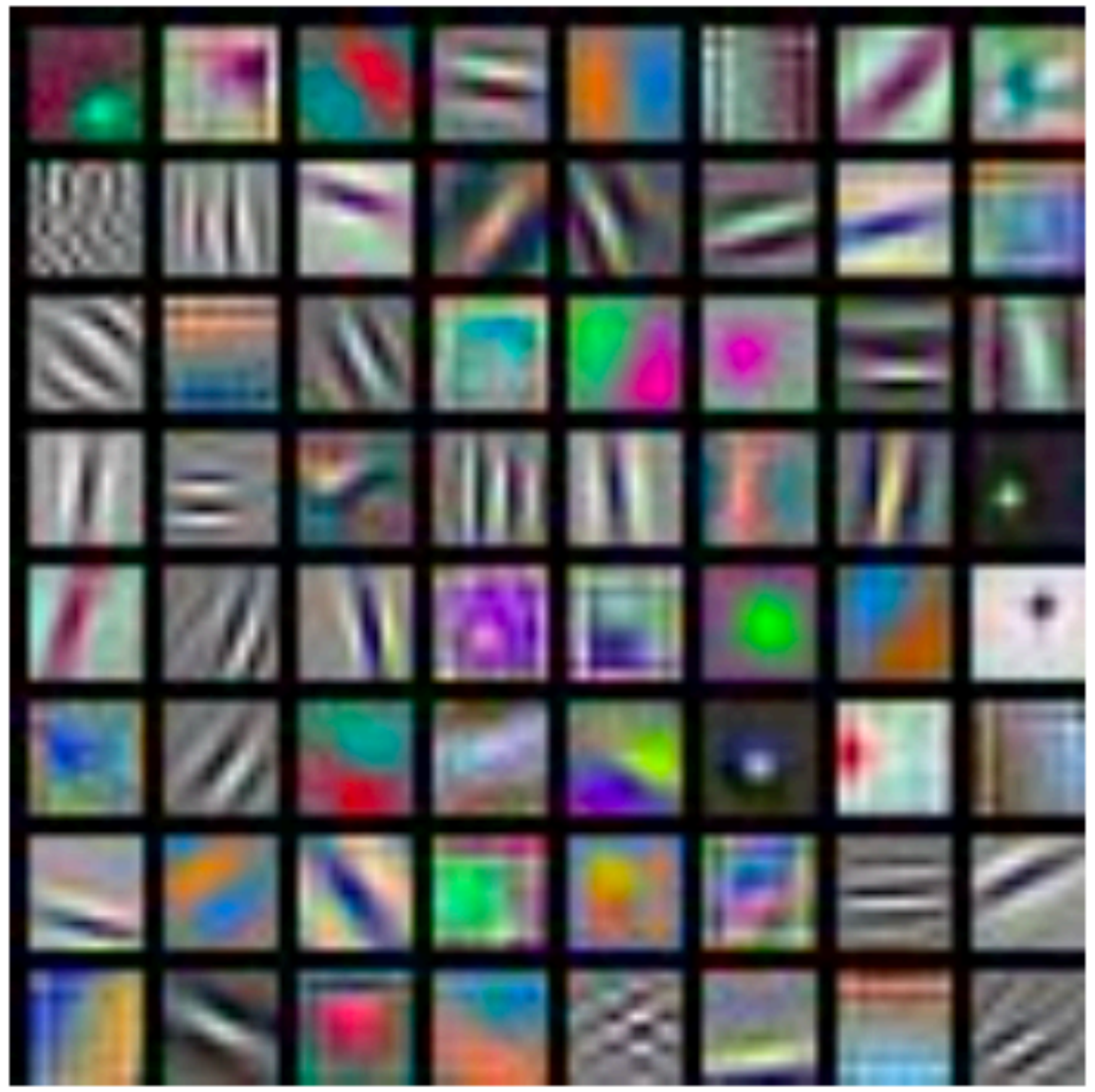
DenseNet-121:
64 x 3 x 7 x 7

… surprisingly similar across variety of networks

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

First Layer Filters …

Directly **visualize filters** (only works for the first layer)



AlexNet:
64 x 3 x 11 x 11

ResNet-18:
64 x 3 x 7 x 7

ResNet-101:
64 x 3 x 7 x 7

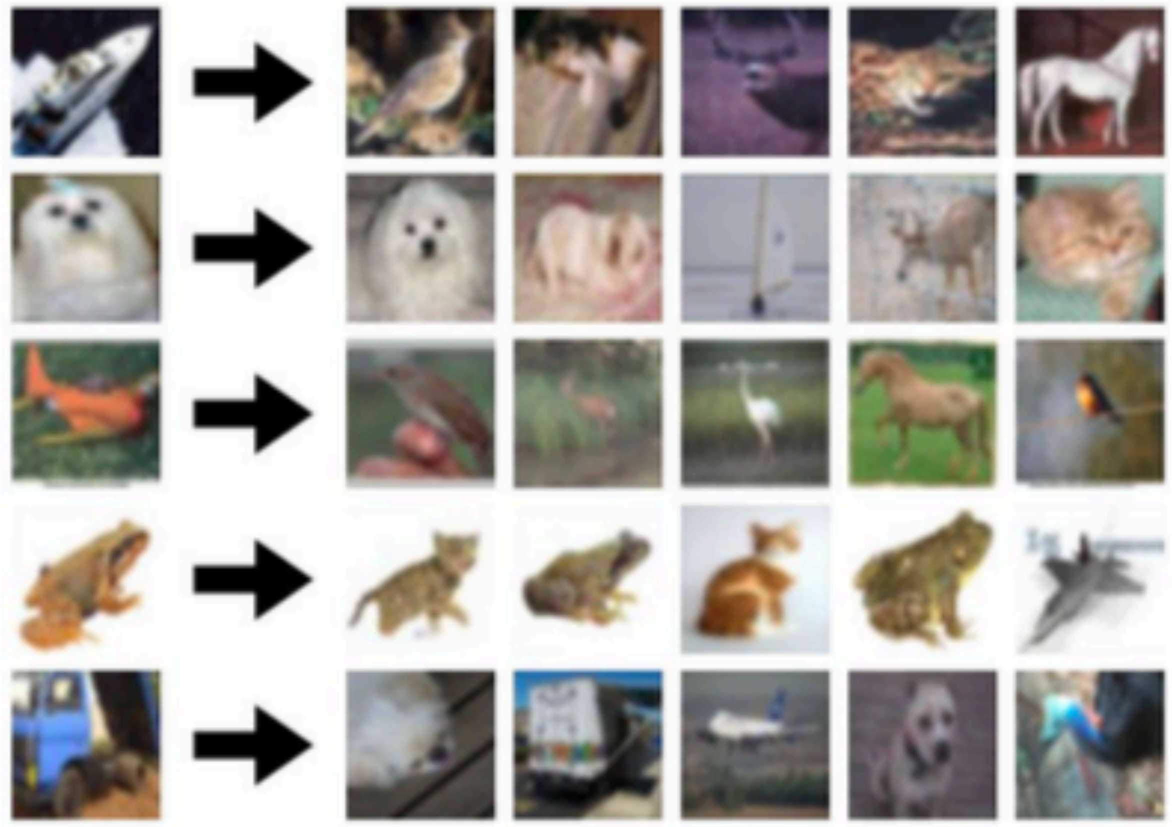
DenseNet-121:
64 x 3 x 7 x 7

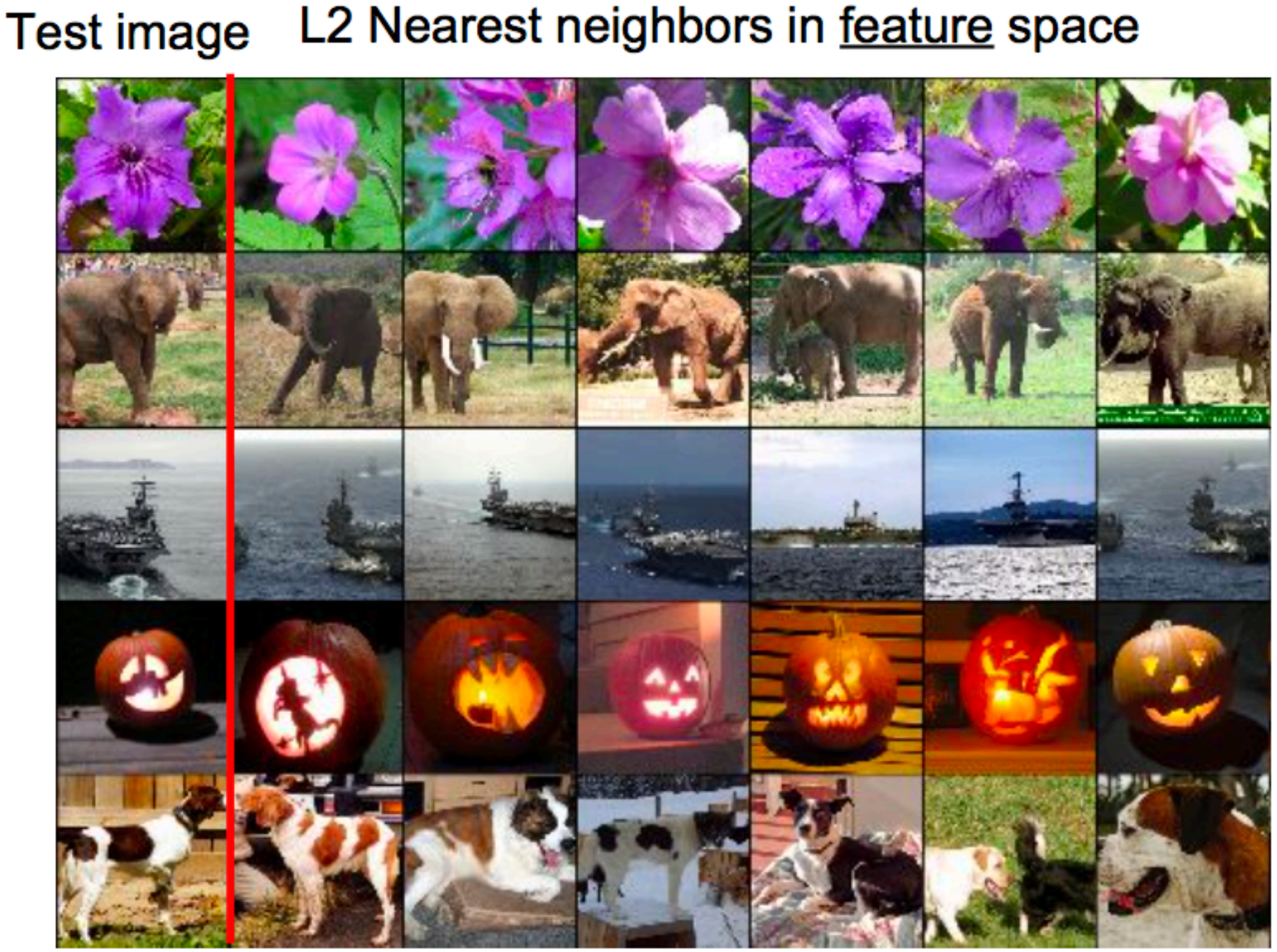
… surprisingly similar across variety of networks

… and nearly any dataset

Last Layer

Recall: Nearest neighbors in pixel space

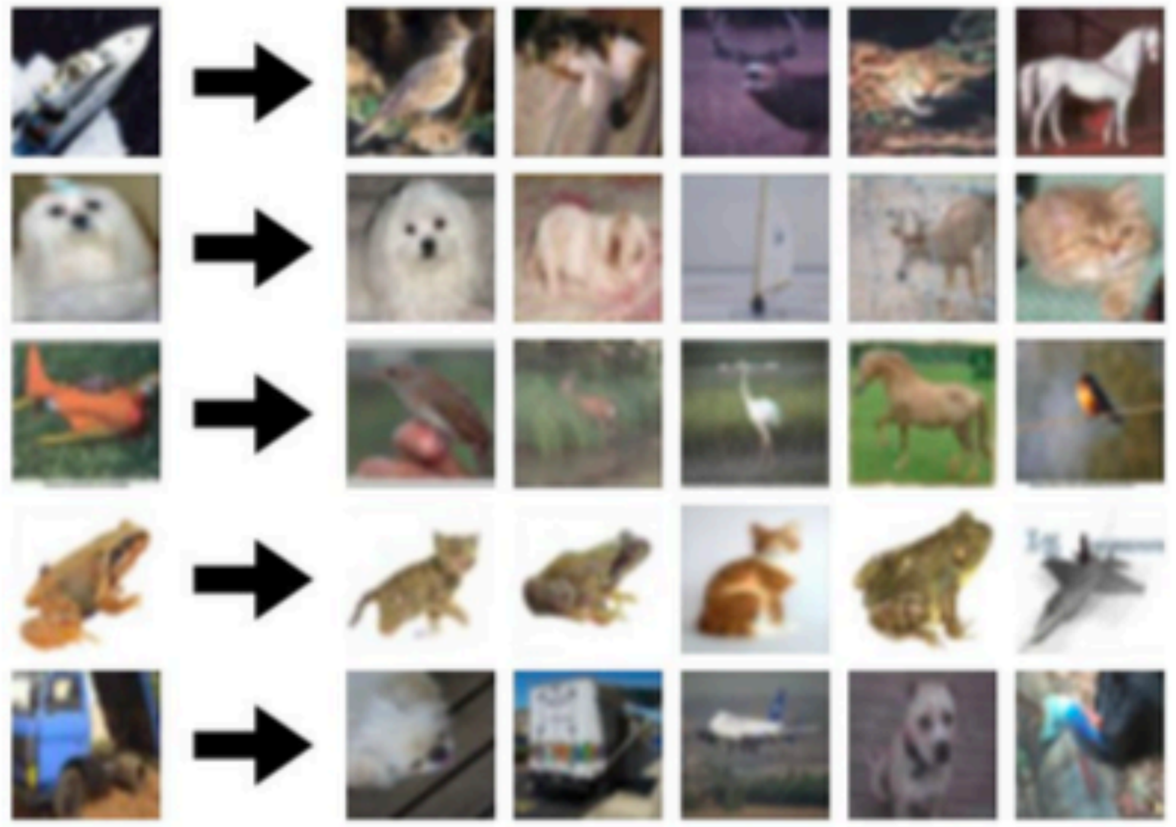




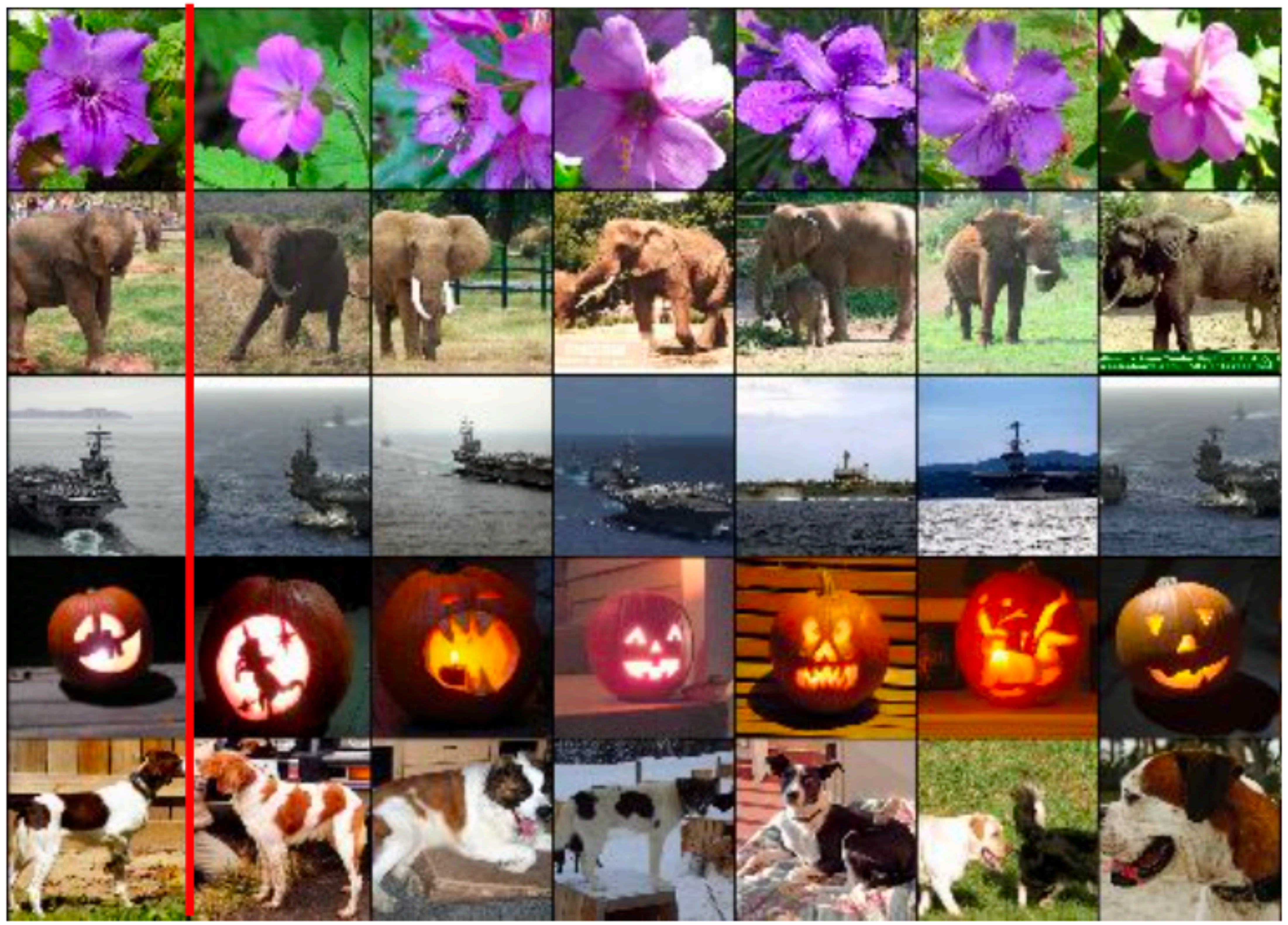
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Last Layer

Recall: Nearest neighbors in pixel space



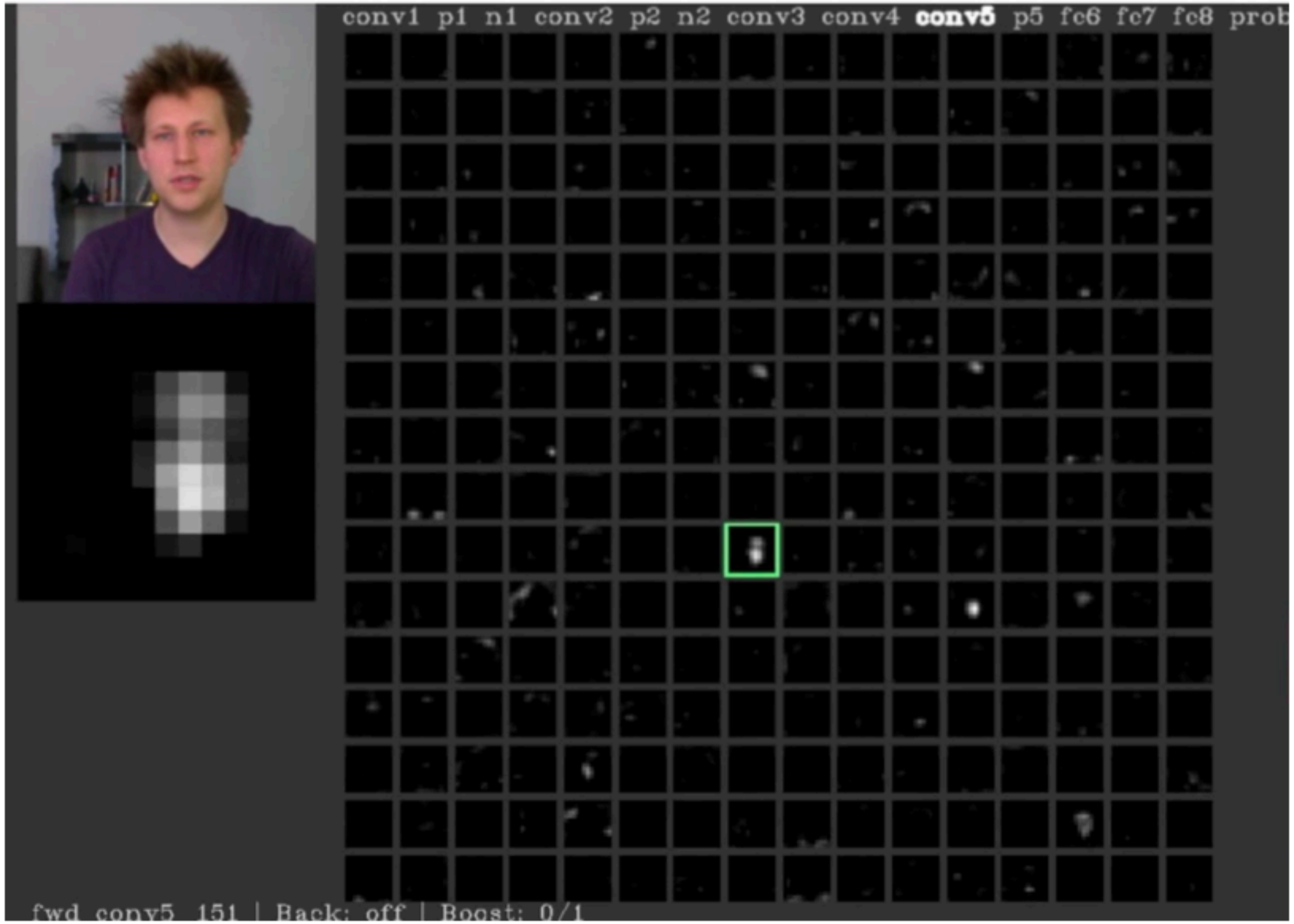
Test image L2 Nearest neighbors in feature space



… you are doing this for **Assignment 2**

Visualizing **Activations**

conv5 feature map of AlexNet is 128x13x13; visualize as 128 13x13 grayscale images

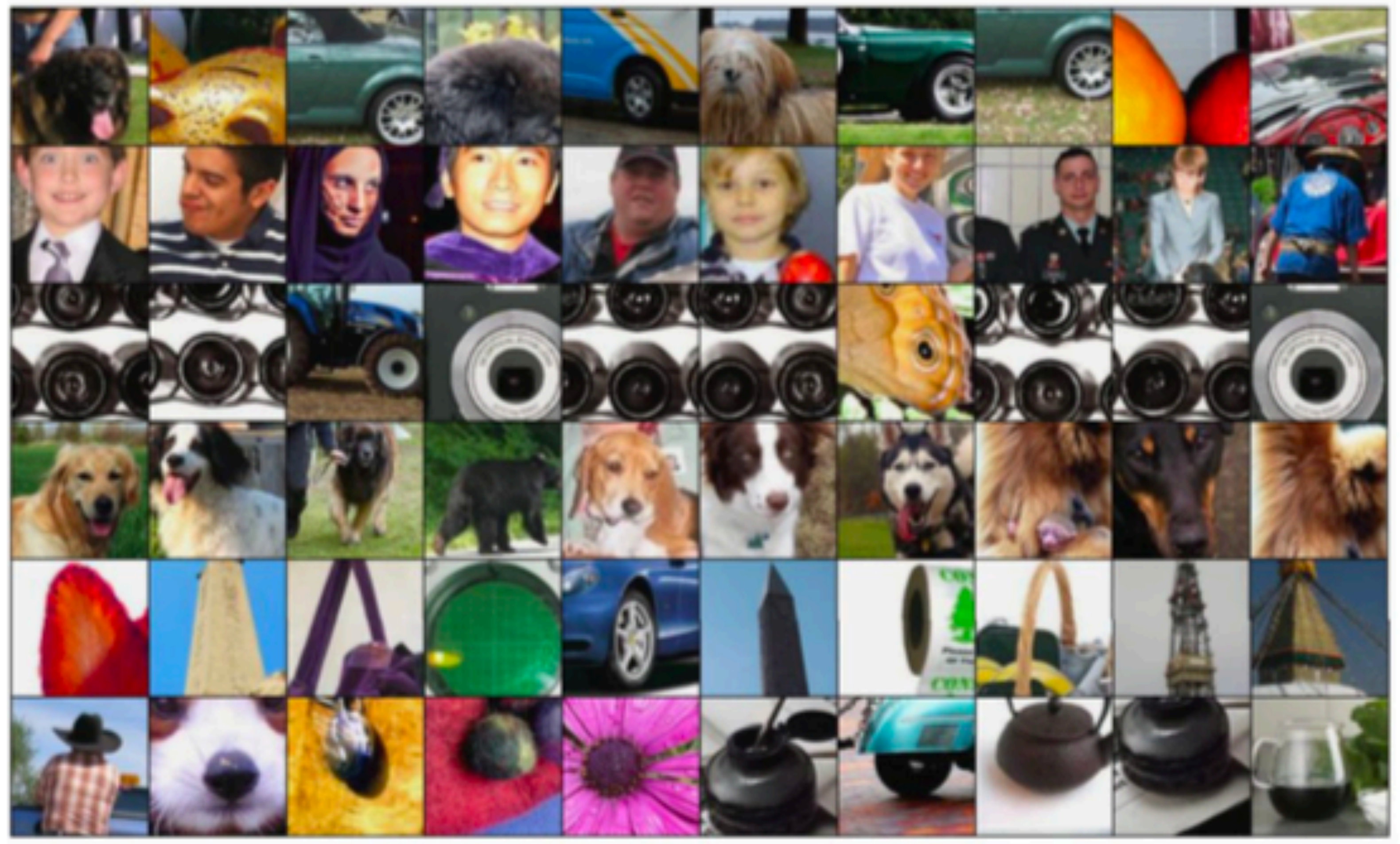


[Yosinski et al., 2014]

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Maximally **Activating Patches**

- Pick a layer and a channel; e.g., cons5 of AlexNet is 128x13x13
- Run many images through the network
- Visualize image patches that correspond to maximal activation of the neuron



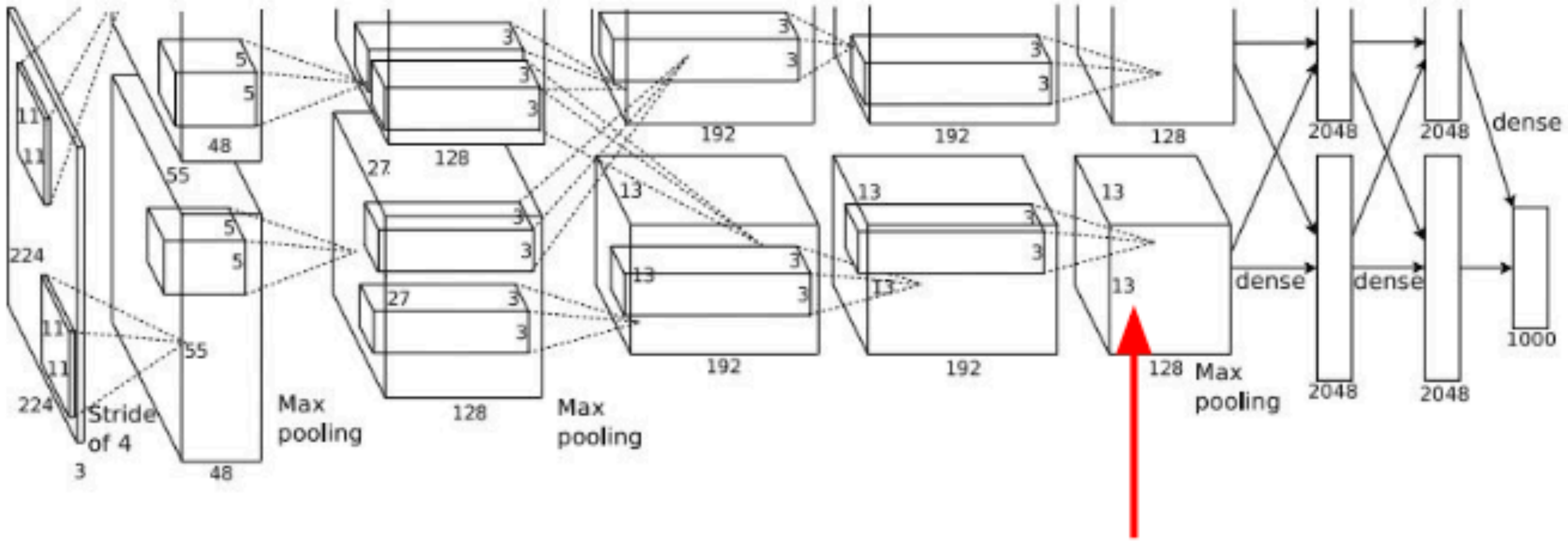
[Springenberg et al., 2015]

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Intermediate Features through (**Guided**) **BackProp**

— Pick a single intermediate neuron somewhere in the network, e.g., neuron in 128x13x13 conv5 feature map

— Compute **gradient of neuron value with respect to image pixels**

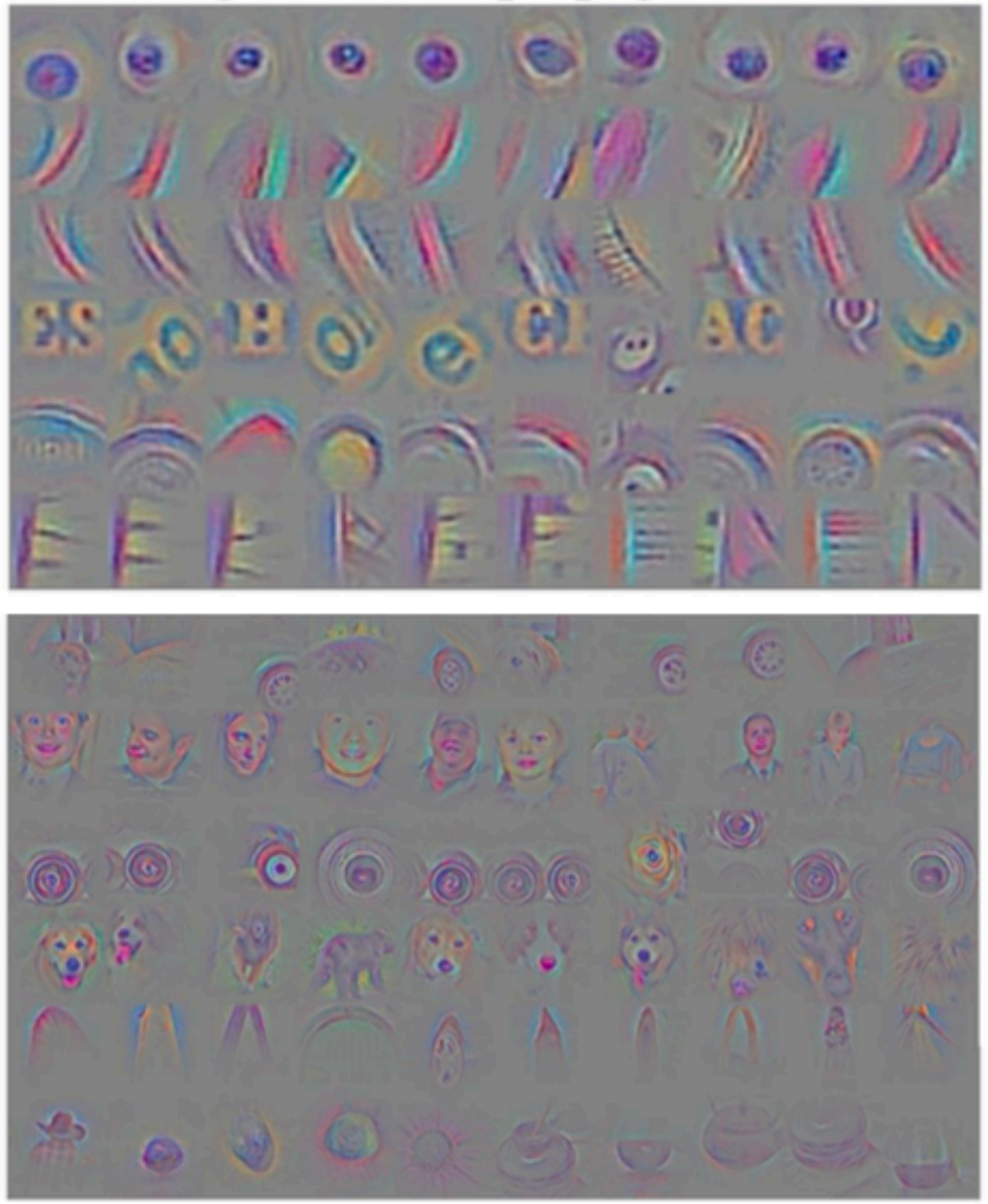


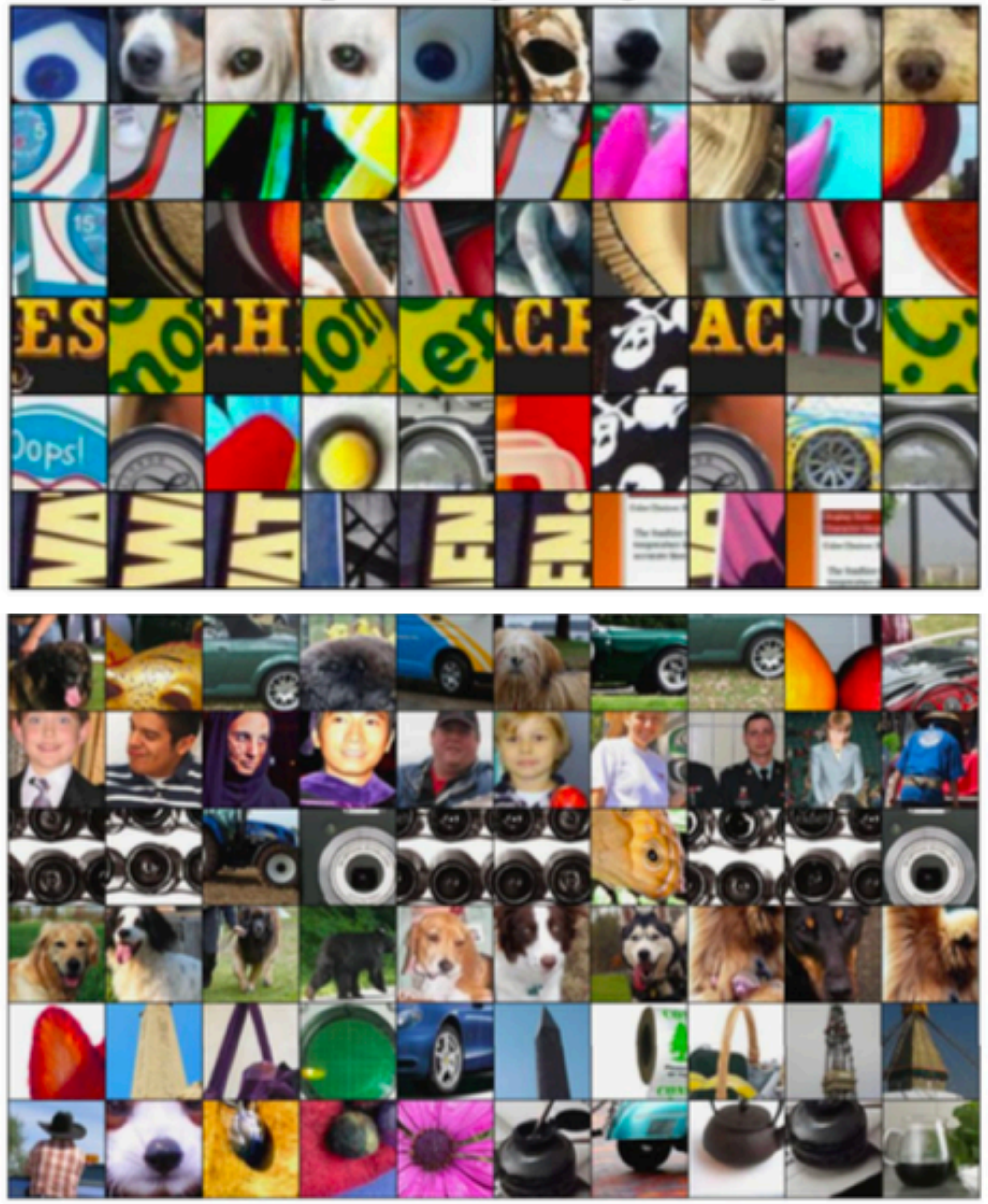
[Zeiler and Fergus, 2014]

[Springenberg et al., 2015]

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Intermediate Features through (**Guided**) **BackProp**





[Zeiler and Fergus, 2014]

[Springenberg et al., 2015]

Gradient **Ascent**

(Guided) **BackProp**: find the part of an image that a neuron responds to

Gradient ascent: generate a synthetic image that maximally activates a neuron

Gradient **Ascent**

(Guided) **BackProp**: find the part of an image that a neuron responds to

Gradient ascent: generate a synthetic image that maximally activates a neuron

$$\mathbf{I}^* = \arg\max_{\mathbf{I}} f(\mathbf{I}) + R(\mathbf{I})$$

Gradient **Ascent**

(Guided) **BackProp**: find the part of an image that a neuron responds to

Gradient ascent: generate a synthetic image that maximally activates a neuron

$$\mathbf{I}^* = \arg\max_{\mathbf{I}} \underbrace{f(\mathbf{I})}_{\text{Neuron Value}} + R(\mathbf{I})$$

Gradient **Ascent**

(Guided) **BackProp**: find the part of an image that a neuron responds to

Gradient ascent: generate a synthetic image that maximally activates a neuron

Natural **Image Regularizer**

$$\mathbf{I}^* = \arg\max_{\mathbf{I}} \underbrace{f(\mathbf{I})}_{\text{Neuron Value}} + \overbrace{R(\mathbf{I})}^{\text{Natural Image Regularizer}}$$

Neuron Value

Gradient **Ascent**

1. Initialize image with all zeros (can also start with an existing image)
2. Forward image to compute the current scores
3. BackProp to get gradient of the neuron with respect to image pixels
4. Make a small update to an image

Natural **Image Regularizer**

$$\mathbf{I}^* = \arg\max_{\mathbf{I}} \underbrace{f(\mathbf{I})}_{\textbf{Neuron Value}} + R(\mathbf{I})$$

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Gradient **Ascent**

1. Initialize image with all zeros (can also start with an existing image)
2. Forward image to compute the current scores
3. BackProp to get gradient of the neuron with respect to image pixels
4. Make a small update to an image

Natural **Image Regularizer** $R(\mathbf{I}) = -\lambda ||\mathbf{I}||_2^2$

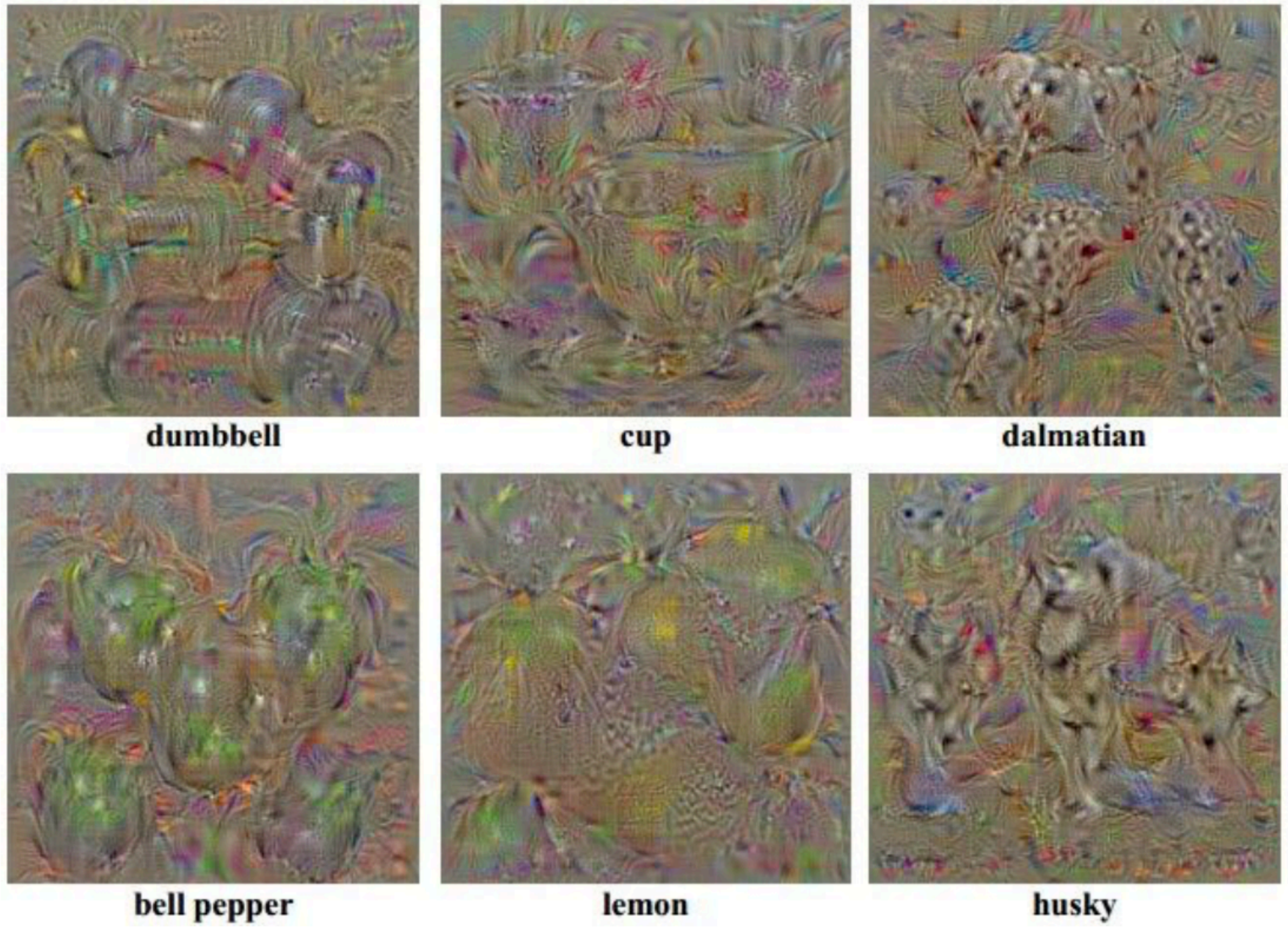
$$\mathbf{I}^* = \arg\max_{\mathbf{I}} f(\mathbf{I}) + R(\mathbf{I})$$

Score for class C before softmax

[Simonyan et al., 2014]

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Gradient **Ascent**



Natural **Image Regularizer** $R(\mathbf{I}) = -\lambda||\mathbf{I}||_2^2$

$$\mathbf{I}^* = \arg\max_{\mathbf{I}} f(\mathbf{I}) + R(\mathbf{I})$$

Score for class C before softmax

[Simonyan et al., 2014]

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Gradient **Ascent**

… with a few additional tweaks

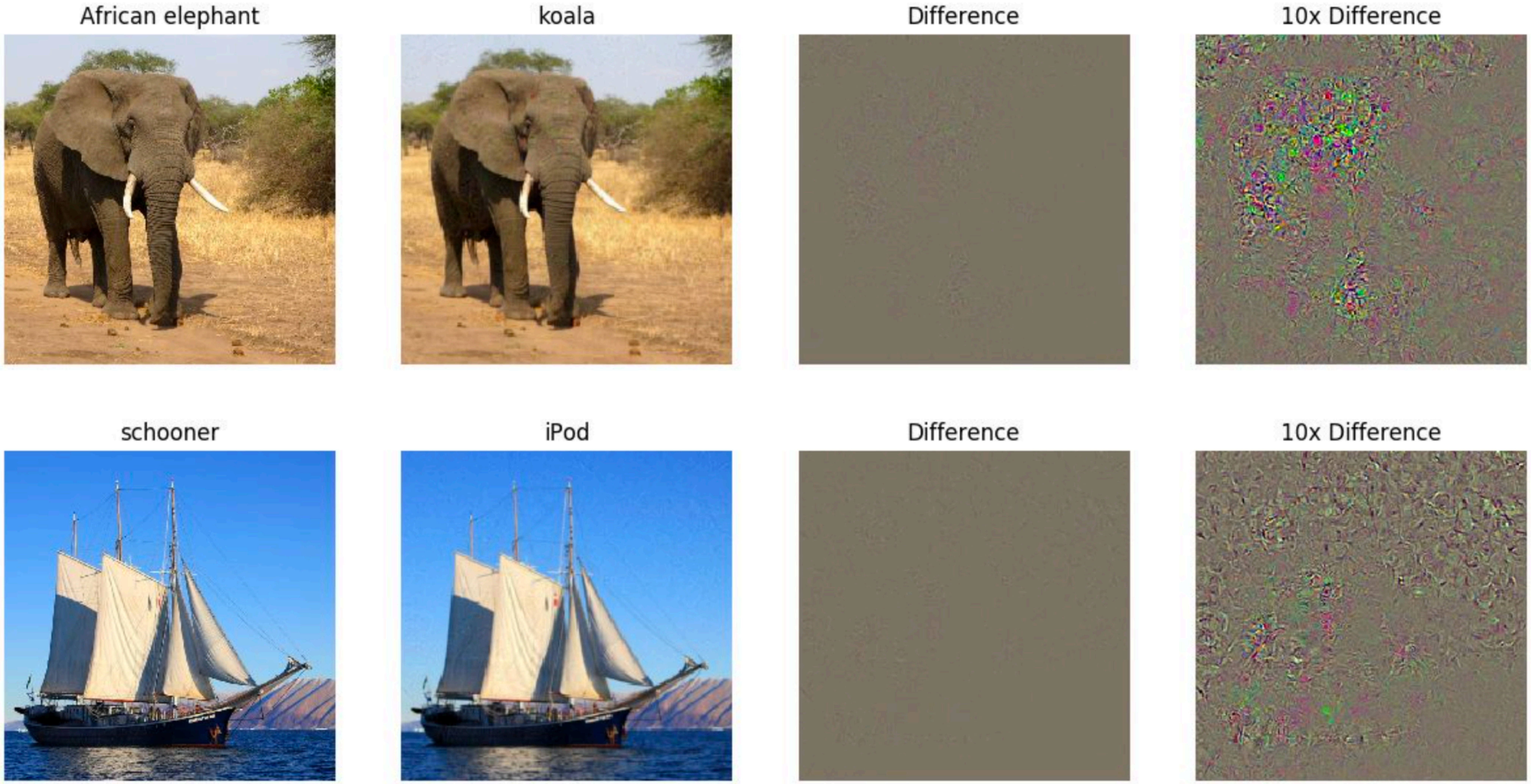
[Nguyen et al., 2015]

Deep **Dream**

[Mordvinsev, Olah, Tyka]

https://www.youtube.com/watch?v=DgPaCWJL7XI&t=11s

Fooling Images / **Adversarial** Examples



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**