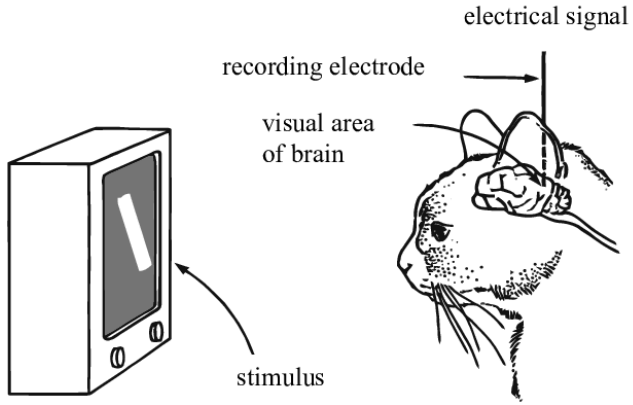
Generating Images from CNNs

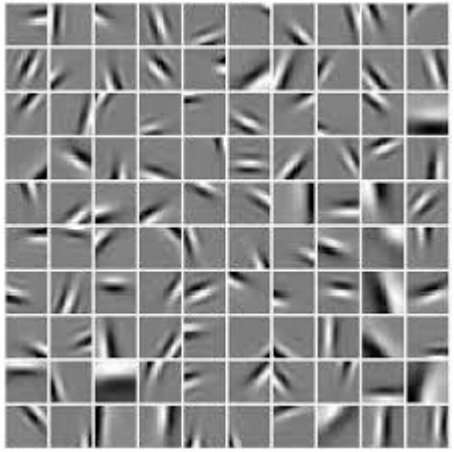
Designing, Visualizing and Understanding Deep Neural Networks

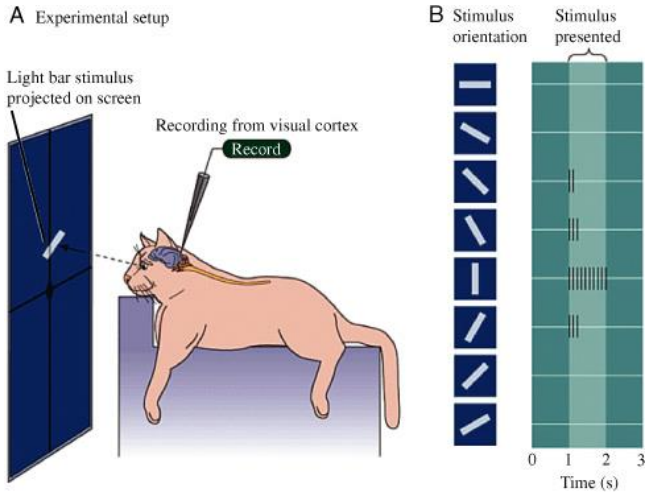
CS W182/282A

Instructor: Sergey Levine
UC Berkeley

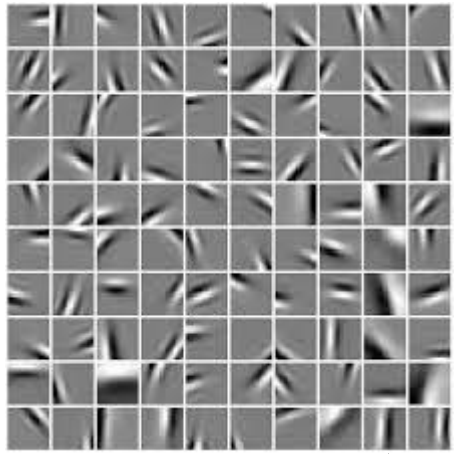
What does the brain see?



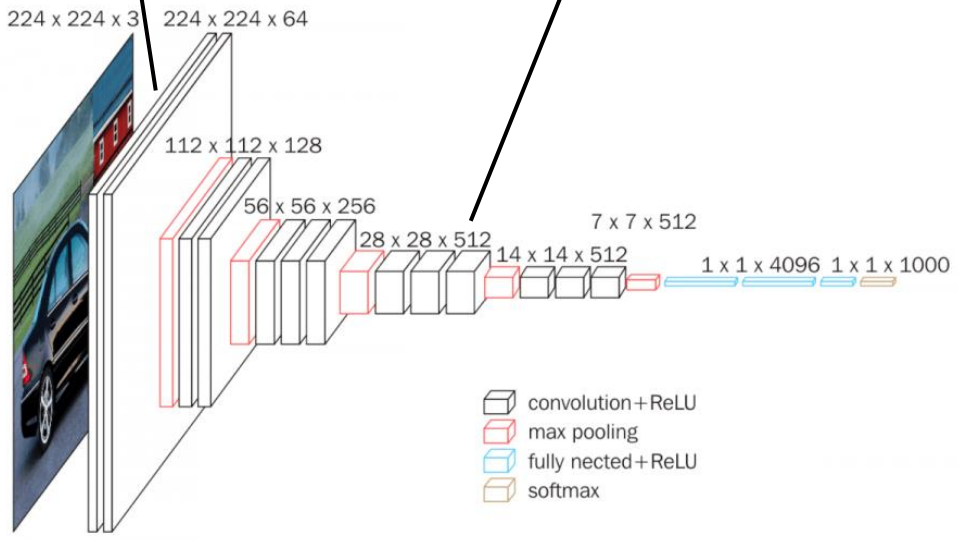




What do convolutional networks “see”?



???

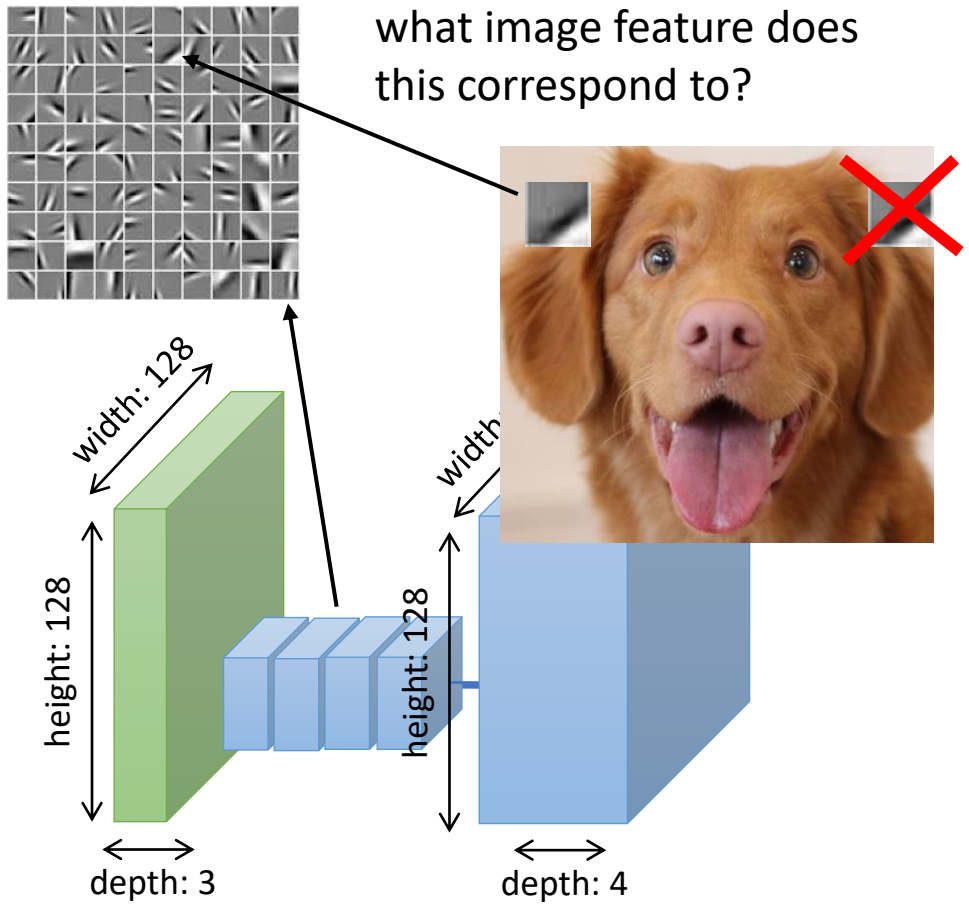


Why are we interested in this?

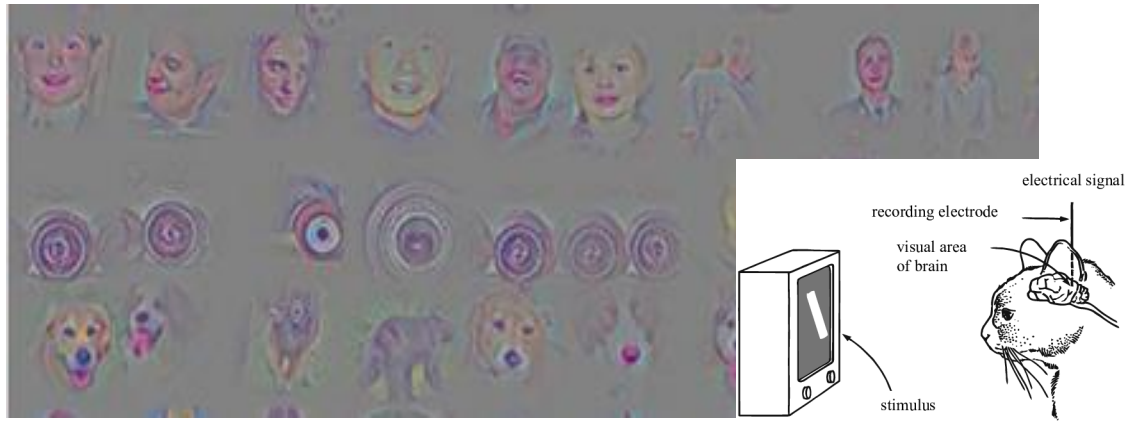
- Interpret what neural nets pay attention to
- Understand when they’ll work and when they won’t
- Compare different models and architectures
- Manipulate images based on conv net responses

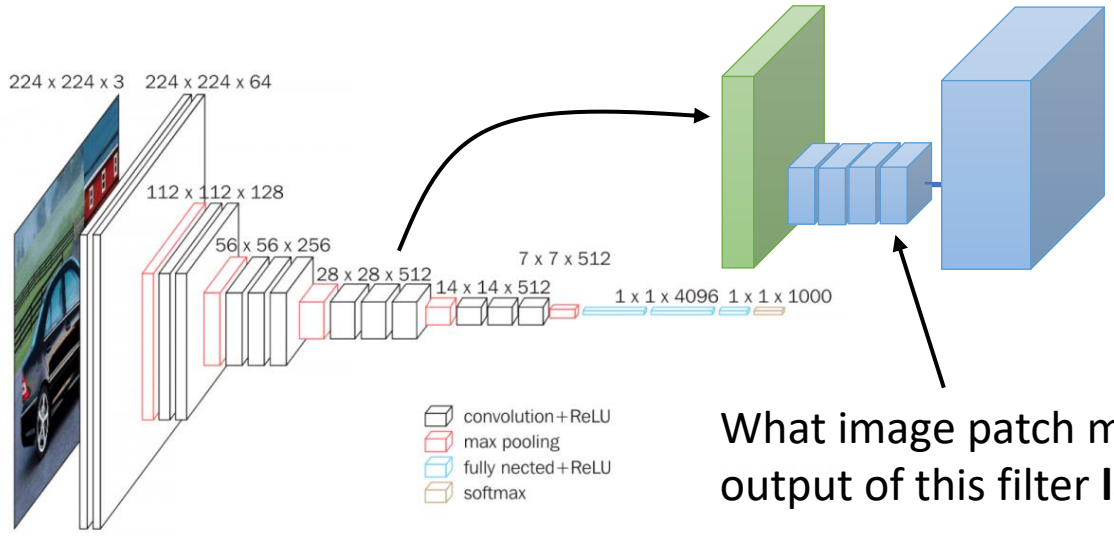
What do we visualize?

Option 1: visualize the **filter** itself



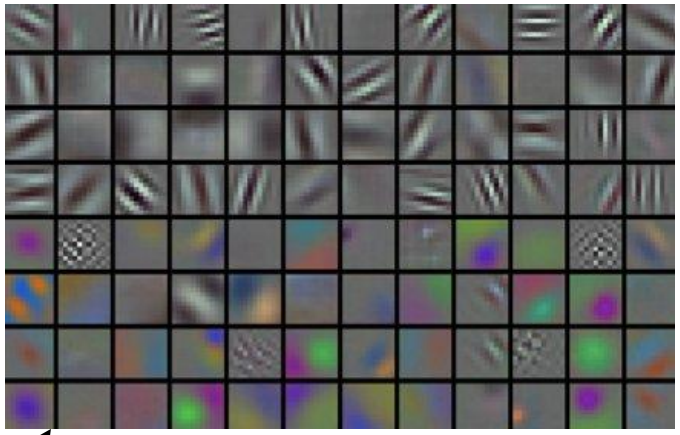
Option 2: visualize stimuli that activate a "neuron"

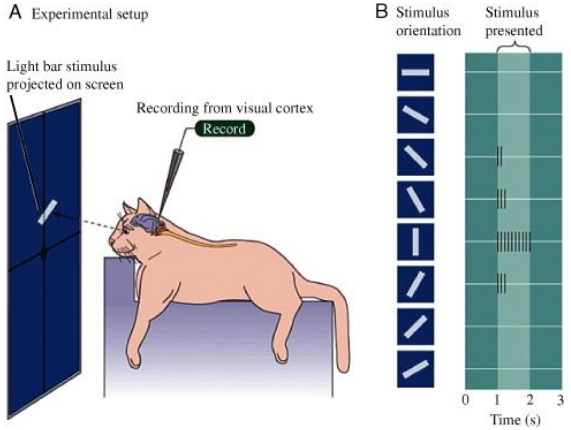


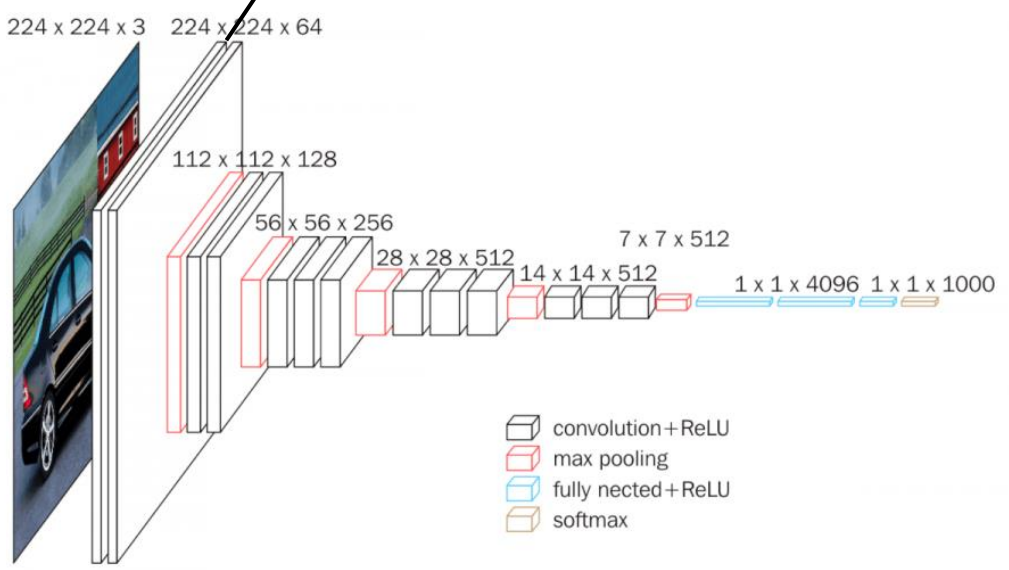


What image patch makes the output of this filter **large**?

Visualizing filters





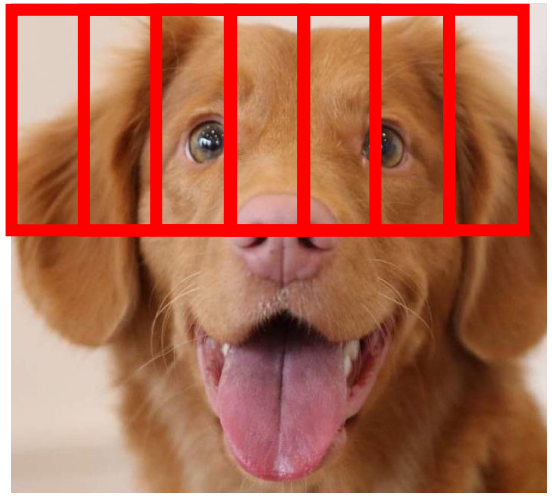


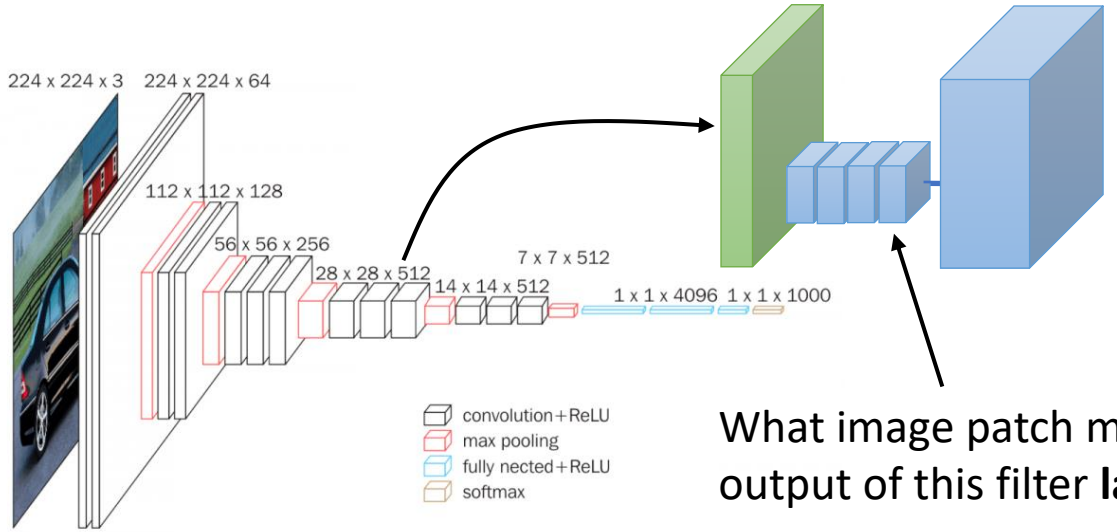
Can’t really visualize higher layers in a way that makes much sense

Visualizing neuron responses

Idea 1: look for images that maximally “excite” specific units

12.3 37.4 17.1 21.4 42.1





What image patch makes the output of this filter **large**?

Visualizing neuron responses

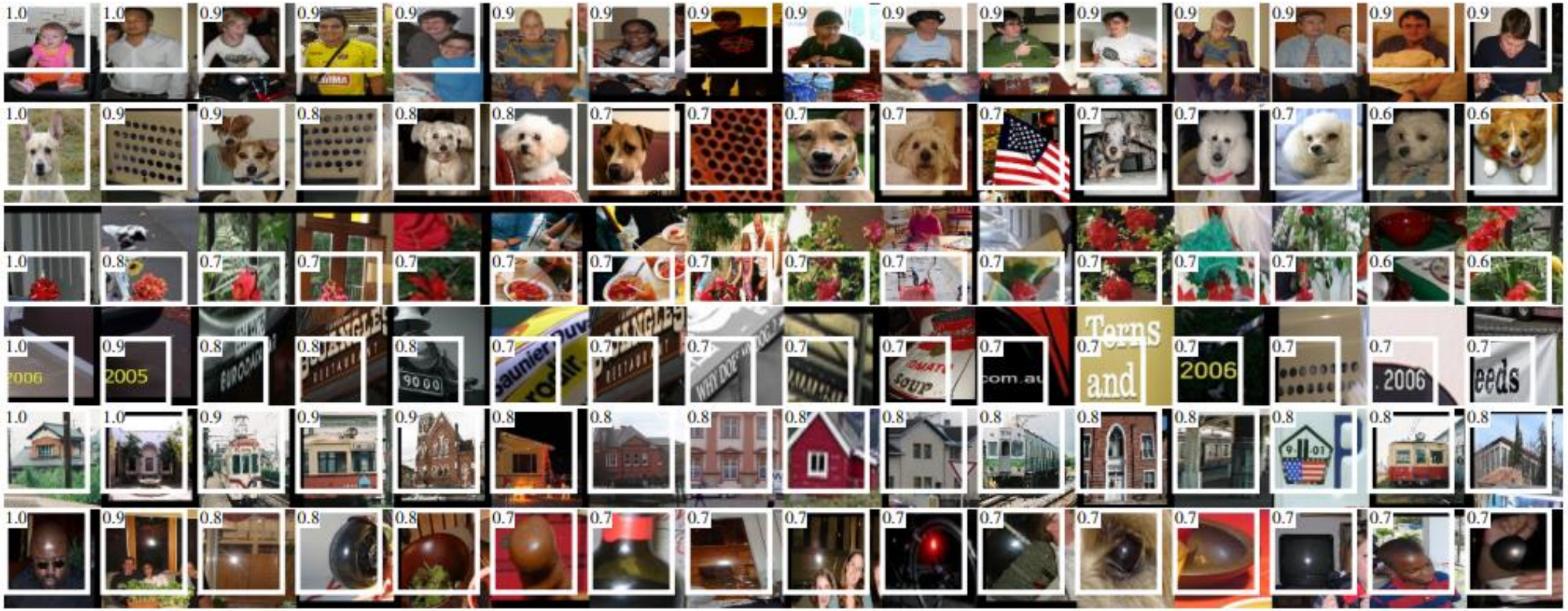


Figure 4: Top regions for six pool$_5$ units. Receptive fields and activation values are drawn in white. Some units are aligned to concepts, such as people (row 1) or text (4). Other units capture texture and material properties, such as dot arrays (2) and specular reflections (6).

Girshick, Donahue, Darrell, Malik. **Rich feature hierarchies for accurate object detection and semantic segmentation**. 2014

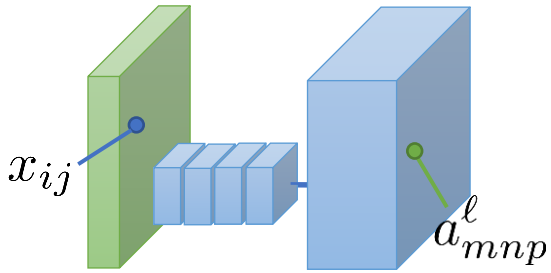
Using gradients for visualization

Idea 1: See which image pixels maximally influence the value at some unit

what does "influence" mean?

given a pixel x_{ij} and a unit a^{ℓ}_{mnp}

how much does changing x_{ij} change a^{ℓ}_{mnp}?



$\frac{da^{\ell}_{mnp}}{dx_{ij}}$ how do we get this quantity?

backpropagation!

1. set δ to be same size as a^{ℓ}
2. set $\delta_{mnp} = 1$, all other entries to 0
3. backprop from layer ℓ to the image
4. last δ gives us $\frac{da^{\ell}_{mnp}}{dx}$

forward pass: calculate each $a^{(i)}$ and $z^{(i)}$

backward pass: $\frac{da^{\ell}_{mnp}}{da^{\ell}}$ ← zero in each position except mnp (which is 1)

initialize $\delta = \frac{d\mathcal{L}}{dz^{(i)}}$

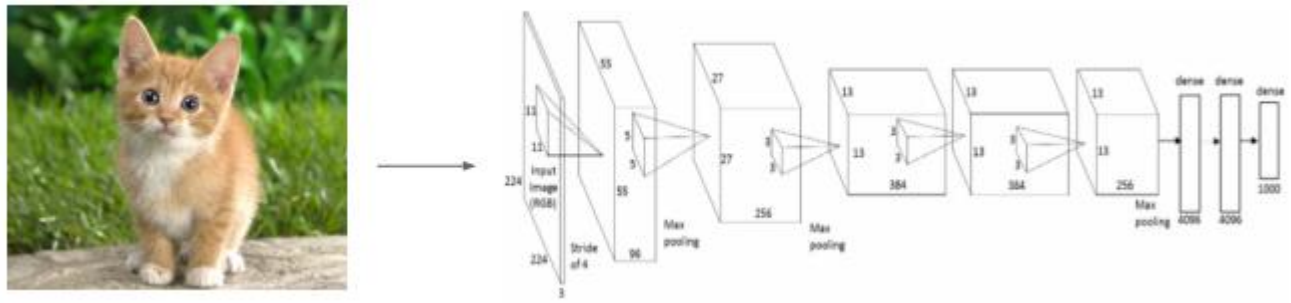
for each f with input x_f & params θ_f from end to start:

$$\frac{d\mathcal{L}}{d\theta_f} \leftarrow \frac{df}{d\theta_f}\delta$$

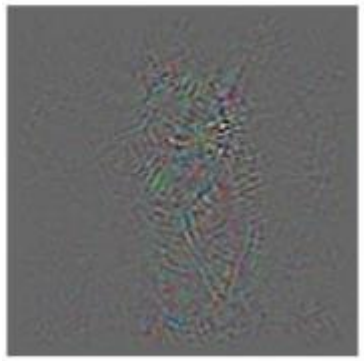
$$\delta \leftarrow \frac{df}{dx_f}\delta$$

Using gradients for visualization

Idea 1: See which image pixels maximally influence the value at some unit $\frac{da^{\ell}_{mnp}}{dx_{ij}}$



$\frac{da^{\ell}_{mnp}}{dx_{ij}}$



slightly modified backprop gives us this:

"guided backpropagation"

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller. **Striving for Simplicity: The All Convolutional Net**. 2014.

basic idea behind "guided backpropagation":

- Just using backprop is not very interpretable, because many units in the network contribute **positive** or **negative** gradients
- If we keep just the **positive** gradients, we'll avoid some of the complicated negative contributions, and get a cleaner signal

heuristically change backward step in ReLU:
for each entry δ_{ijk}, set it to $\max(0, \delta_{ijk})$
"zero out negative gradients at each layer"

Images from CS231n (Fei-Fei Li, Andrej Karpathy)

Using gradients for visualization

Idea 1: See which image pixels maximally influence the value at some unit $\frac{da^{\ell}_{mnp}}{dx_{ij}}$

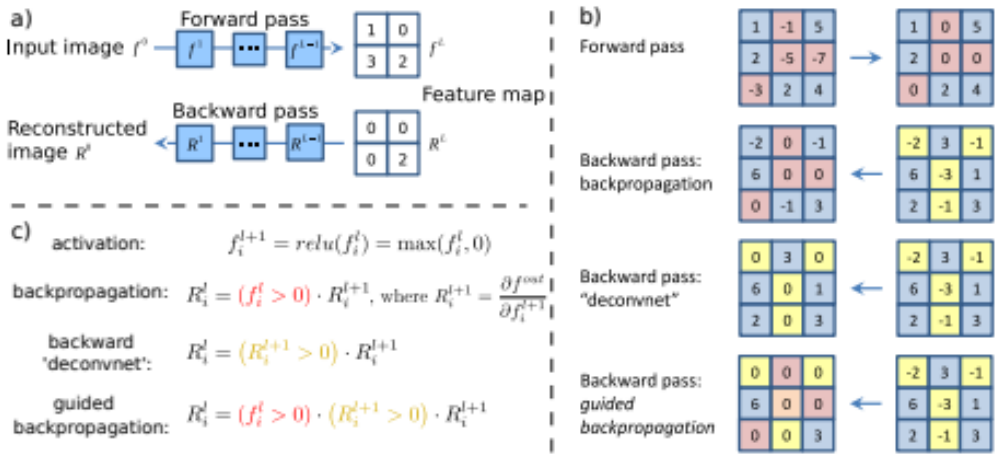


Figure 1: Schematic of visualizing the activations of high layer neurons. a) Given an input image, we perform the forward pass to the layer we are interested in, then set to zero all activations except one and propagate back to the image to get a reconstruction. b) Different methods of propagating back through a ReLU nonlinearity. c) Formal definition of different methods for propagating a output activation *out* back through a ReLU unit in layer l; note that the 'deconvnet' approach and guided backpropagation do not compute a true gradient but rather an imputed version.

basic idea behind "guided backpropagation":

- Just using backprop is not very interpretable, because many units in the network contribute **positive** or **negative** gradients
- If we keep just the **positive** gradients, we'll avoid some of the complicated negative contributions, and get a cleaner signal

heuristically change backward step in ReLU:

for each entry δ_{ijk}, set it to $\max(0, \delta_{ijk})$

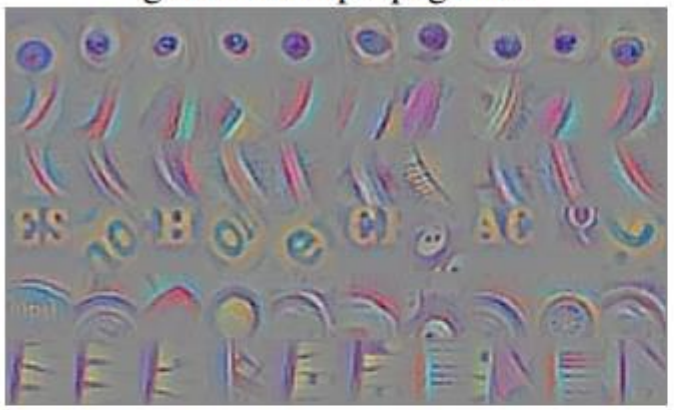
"zero out negative gradients at each layer"

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller. **Striving for Simplicity: The All Convolutional Net**. 2014.

Using gradients for visualization

Idea 1: See which image pixels maximally influence the value at some unit

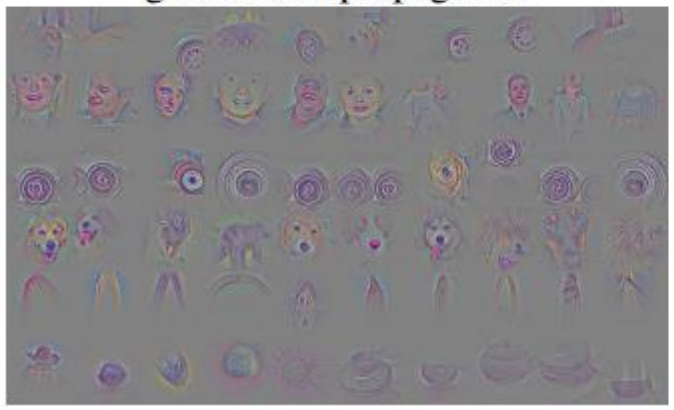
guided backpropagation



corresponding image crops

CONV6

guided backpropagation



corresponding image crops

CONV9

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller. **Striving for Simplicity: The All Convolutional Net**. 2014.

Visualizing Features with Backpropagation

Using gradients for visualization

Idea 2: “Optimize” the image to maximally activate a particular unit

Before: just compute $\frac{da^{\ell}_{mnp}}{dx_{ij}}$

Now: compute $g = \frac{da^{\ell}_{mnp}}{dx}$

$x \leftarrow x + \alpha g$

what optimization problem is this solving?

$$x \leftarrow \arg\max_x a^{\ell}_{mnp}(x)$$

a^{ℓ}_{mnp} activation for image x

more generally: $x \leftarrow \arg\max_x S(x)$

some activation or class label

Using gradients for visualization

Idea 2: “Optimize” the image to maximally activate a particular unit

$$x \leftarrow \arg\max_x S(x) + R(x)$$

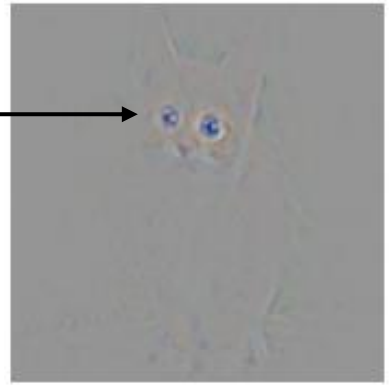
image regularizer to prevent crazy images

$$\text{simple choice: } R(x) = \lambda ||x||_2^2$$

problem: it’s too easy to produce “crazy” images

making the eyes more blue increases activation

what if we the set blue channel to 1,000,000,000?



Simonyan, Vidaldi, Zisserman. **Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps.** 2013.

Visualizing “classes”

$$x \leftarrow \arg\max_x S(x) + R(x)$$

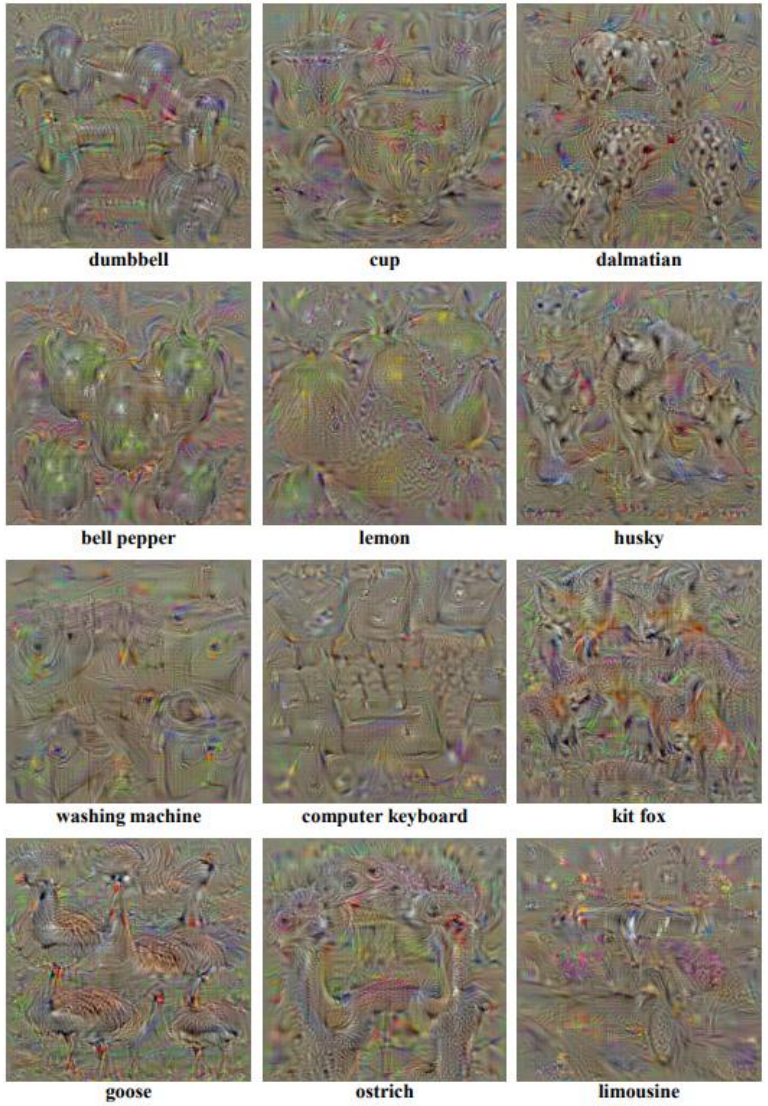
simple choice: $R(x) = \lambda ||x||_2^2$

Which unit to maximize?

Let’s try maximizing class labels first

Important detail: not maximizing **class probabilities**, but the activations **right before** the softmax

$$\underbrace{p(y|x)}_{\text{not this}} = \text{softmax}(\underbrace{W^\ell a^\ell(x) + b^\ell}_{\text{this}})$$



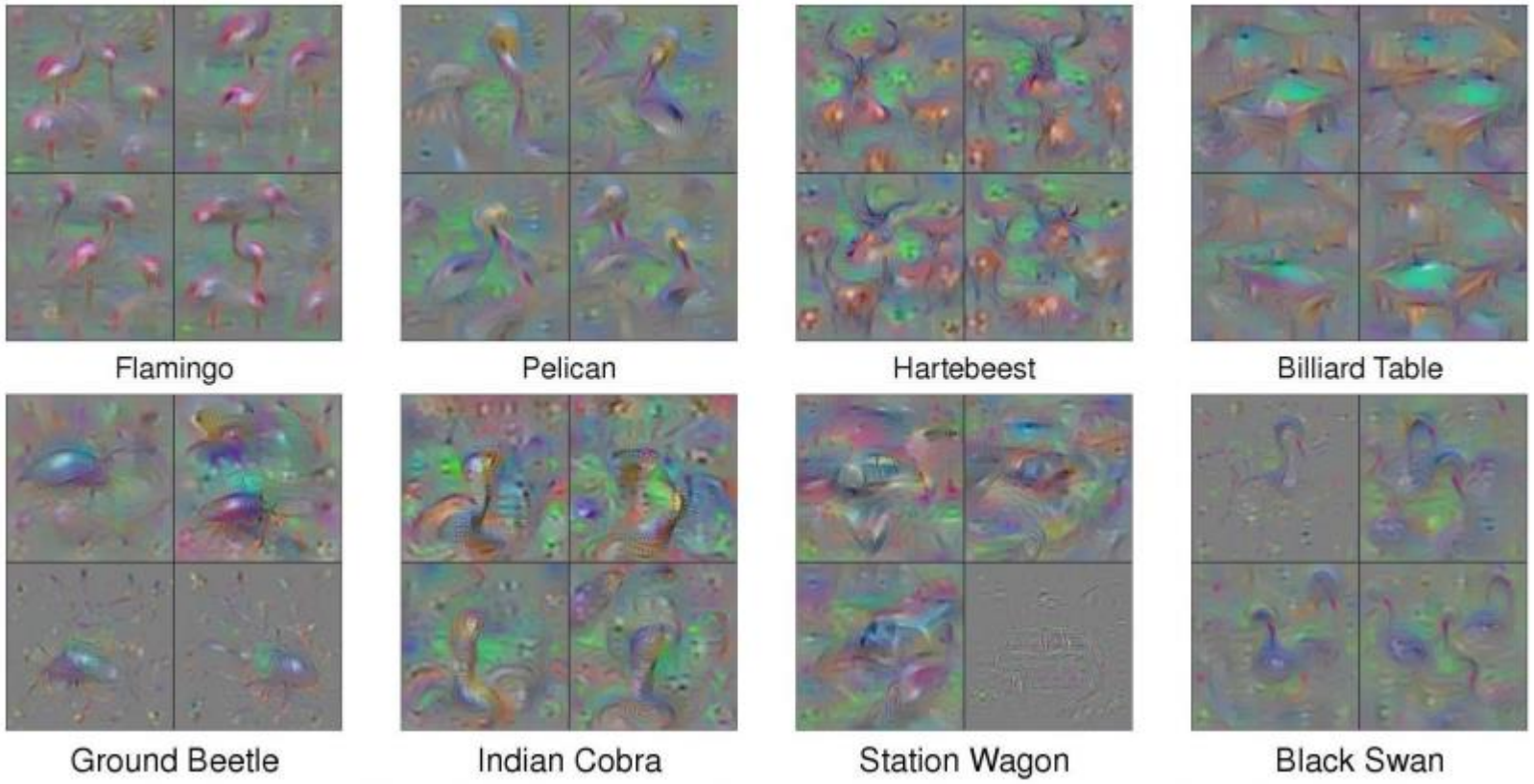
Simonyan, Vidaldi, Zisserman. **Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps.** 2013.

Visualizing “classes”

$$x \leftarrow \arg\max_x S(x) + R(x)$$

slightly more nuanced regularizer

1. Update image with gradient
2. Blur the image a little
3. Zero out any pixel with small value
4. Repeat



Slide based on CS231n, Fei-Fei Li & Andrej Karpathy

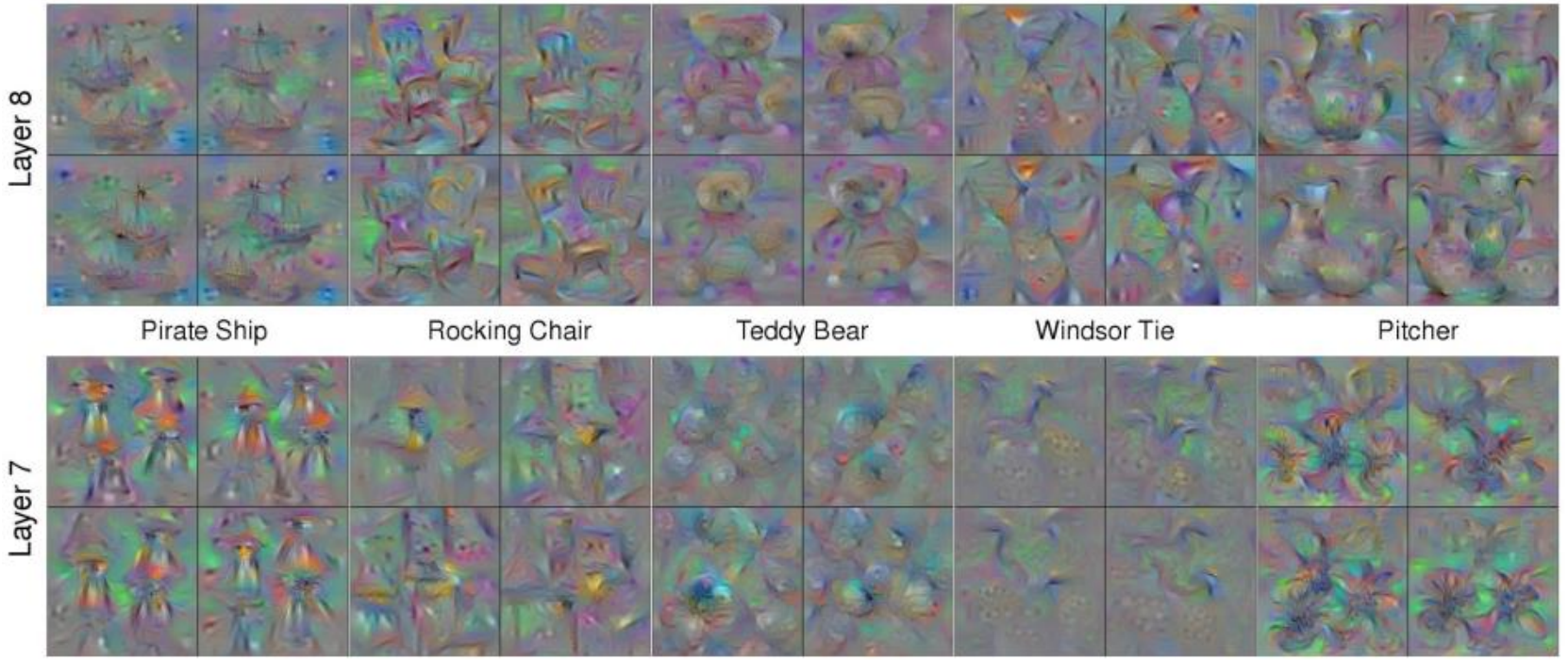
Yosinski et al. **Understanding Neural Networks Through Deep Visualization.** 2015.

Visualizing units

$$x \leftarrow \arg\max_x S(x) + R(x)$$

slightly more nuanced regularizer

1. Update image with gradient
2. Blur the image a little
3. Zero out any pixel with small value
4. Repeat



Slide based on CS231n, Fei-Fei Li & Andrej Karpathy

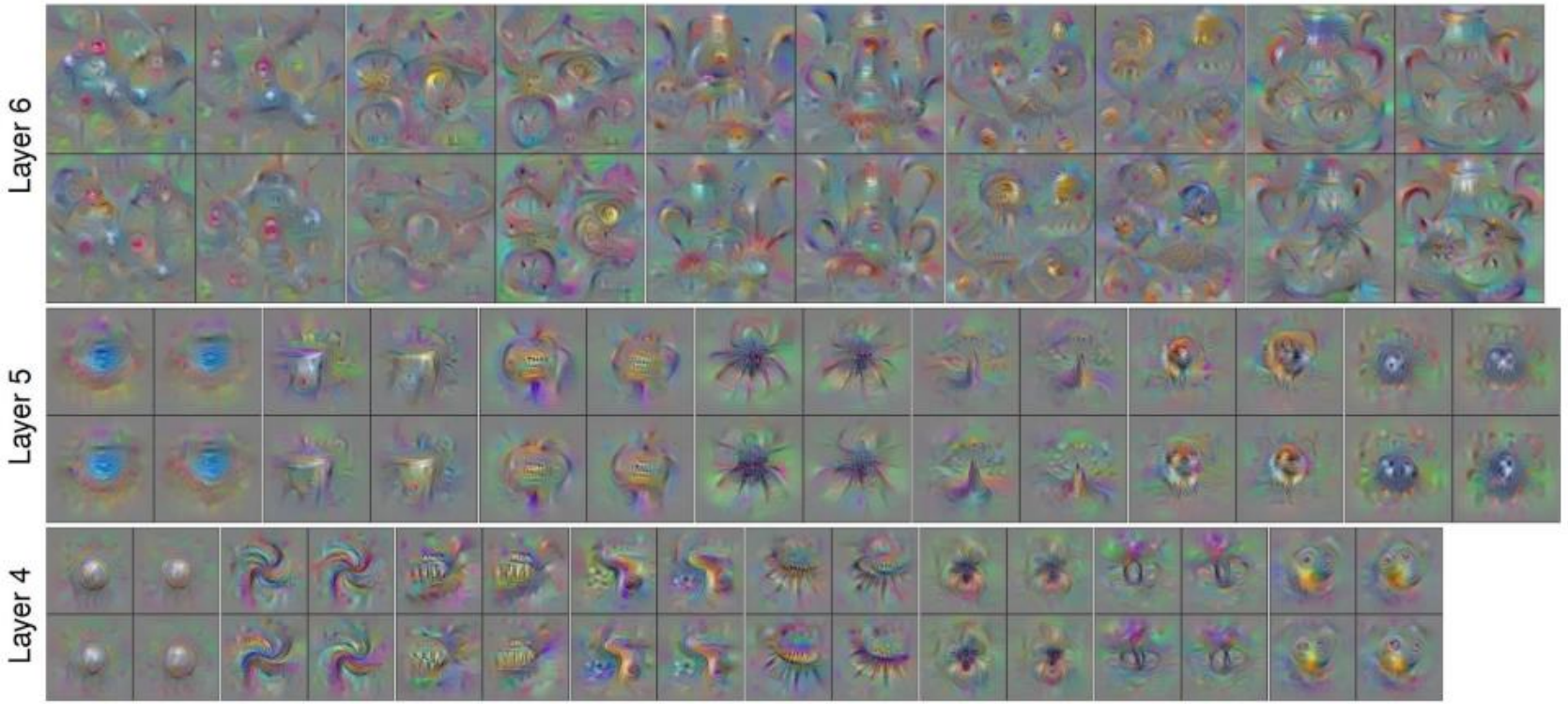
Yosinski et al. **Understanding Neural Networks Through Deep Visualization.** 2015.

Visualizing units

$$x \leftarrow \arg\max_x S(x) + R(x)$$

slightly more nuanced regularizer

1. Update image with gradient
2. Blur the image a little
3. Zero out any pixel with small value
4. Repeat



Slide based on CS231n, Fei-Fei Li & Andrej Karpathy

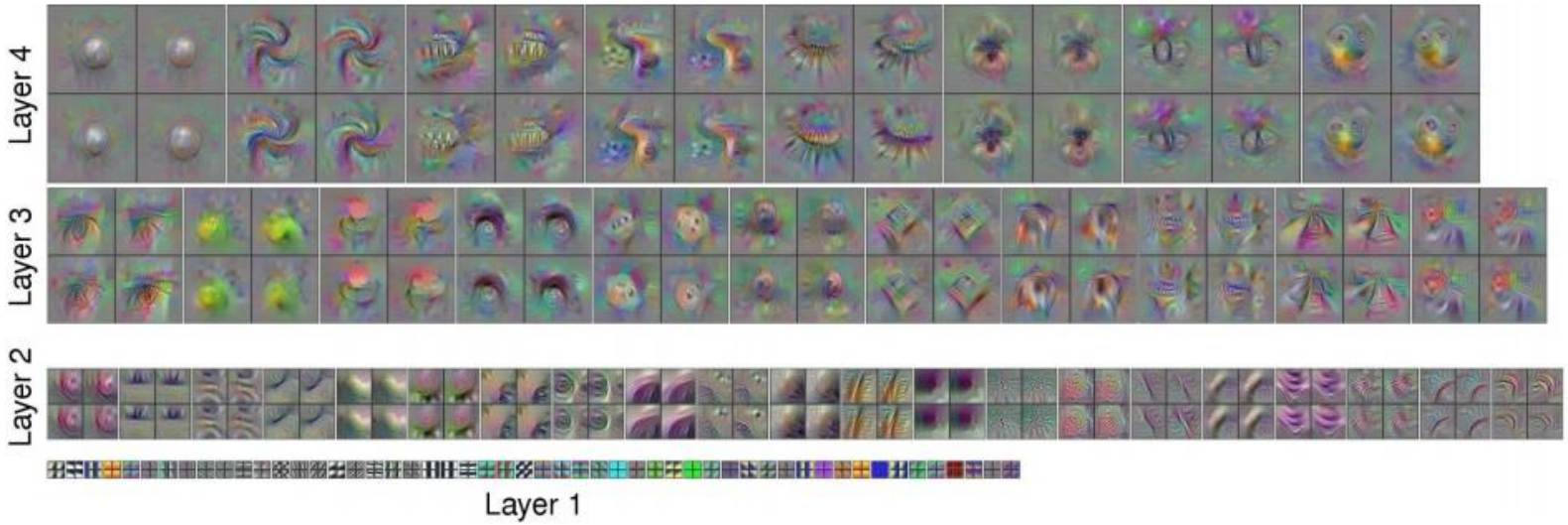
Yosinski et al. **Understanding Neural Networks Through Deep Visualization.** 2015.

Visualizing units

$$x \leftarrow \arg\max_x S(x) + R(x)$$

slightly more nuanced regularizer

1. Update image with gradient
2. Blur the image a little
3. Zero out any pixel with small value
4. Repeat



Slide based on CS231n, Fei-Fei Li & Andrej Karpathy

Yosinski et al. **Understanding Neural Networks Through Deep Visualization.** 2015.

Deep Dream & Style Transfer

Using backprop to *modify* pictures?

Before:

What is channel 17 in layer conv5 looking at?

Now:

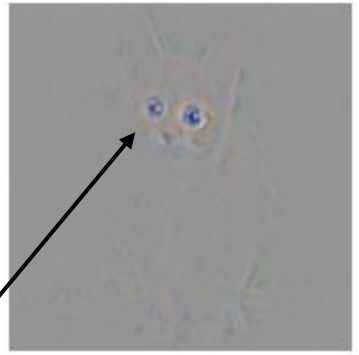
What would it look like if it were a Monet painting?

??

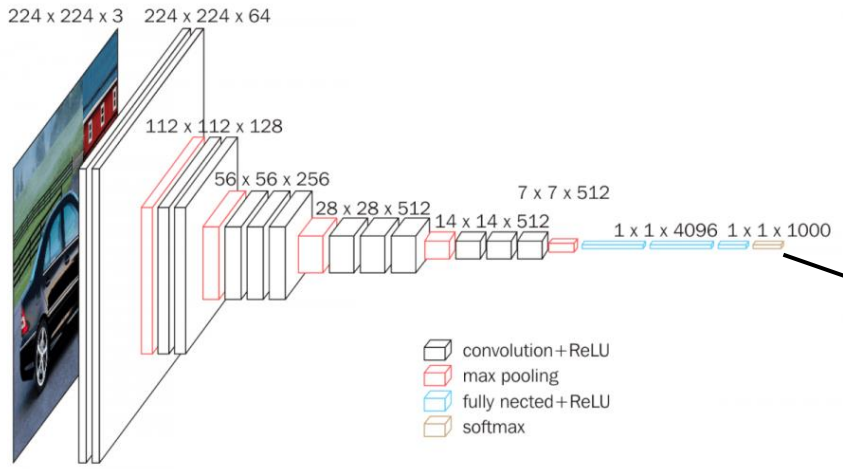
How are these questions related?

Intuition: Monet paintings have particular feature distributions, we can “transport” these distributions to other images!

Looking for patterns in clouds



making the eyes more blue will
make this more cat-like



“looks kind of like a dog”
“where is the dog?”
“let me show you!”

Images from CS231n (Fei-Fei Li, Andrej Karpathy)

DeepDream

inception_4c/output

1. Pick a layer
2. Run forward pass to compute activations at that layer
3. Set delta to be **equal** to the activations
4. Backprop and apply the gradient
5. Repeat

Images from CS231n (Fei-Fei Li, Andrej Karpathy)

DeepDream

inception_4c/output

Images from CS231n (Fei-Fei Li, Andrej Karpathy)

DeepDream

```
def objective_L2(dst):
    dst.diff[:] = dst.data

def make_step(net, step_size=1.5, end='inception_4c/output',
              jitter=32, clip=True, objective=objective_L2):
    '''Basic gradient ascent step.'''

    src = net.blobs['data'] # input image is stored in Net's 'data' blob
    dst = net.blobs[end]

    ox, oy = np.random.randint(-jitter, jitter+1, 2)
    src.data[0] = np.roll(np.roll(src.data[0], ox, -1), oy, -2) # apply jitter shift

    net.forward(end=end)
    objective(dst)  # specify the optimization objective
    net.backward(start=end)
    g = src.diff[0]
    # apply normalized ascent step to the input image
    src.data[:] += step_size/np.abs(g).mean() * g

    src.data[0] = np.roll(np.roll(src.data[0], -ox, -1), -oy, -2) # unshift image

    if clip:
        bias = net.transformer.mean['data']
        src.data[:] = np.clip(src.data, -bias, 255-bias)
```

DeepDream: set dx = x :)

jitter regularizer

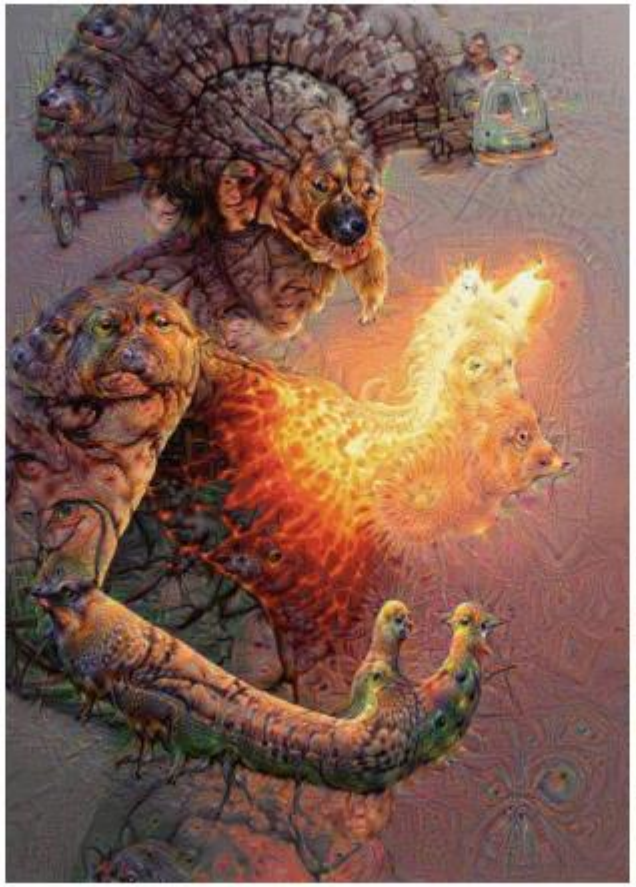
"image update"

Remember this:

$$x \leftarrow \arg\max_x S(x) + R(x)$$

Images from CS231n (Fei-Fei Li, Andrej Karpathy)

DeepDream



DeepDream https://github.com/google/deepdream

Images from CS231n (Fei-Fei Li, Andrej Karpathy)

Another idea

Another idea: instead of exaggerating the features in a single image, what if we make feature of one image look more like the features in **another**?

how do we **quantify** style?

style

relationships between features

e.g., lots of angles and straight lines

content

spatial positions of features

e.g., where curves, edges, and corners are located

This really works???

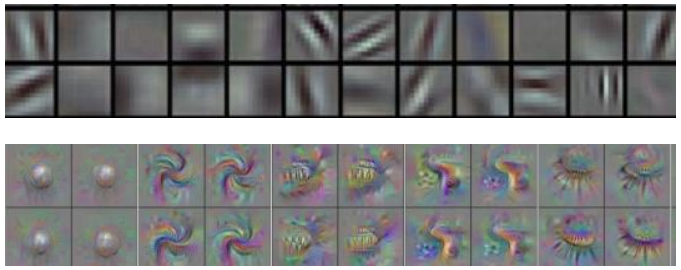
Images from CS231n (Fei-Fei Li, Andrej Karpathy)

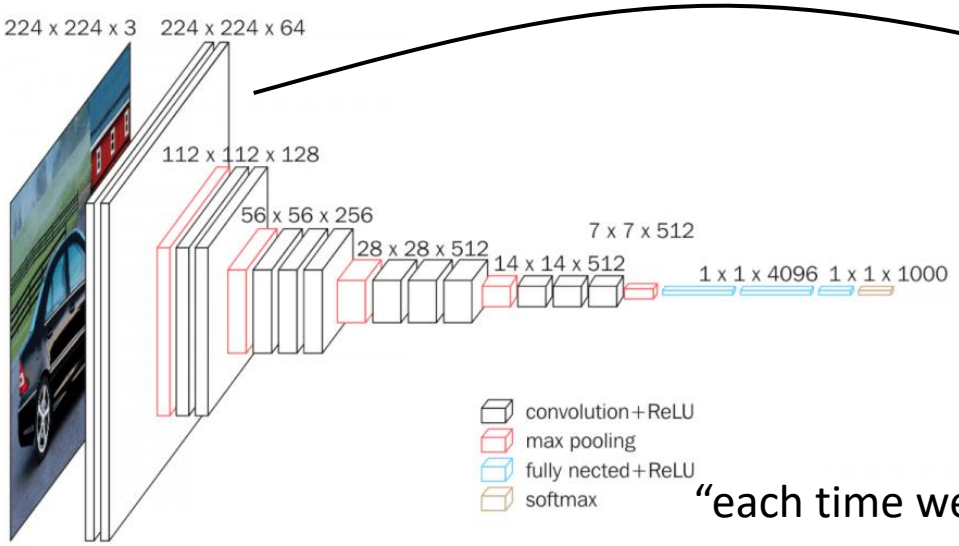
How do we quantify **style**?

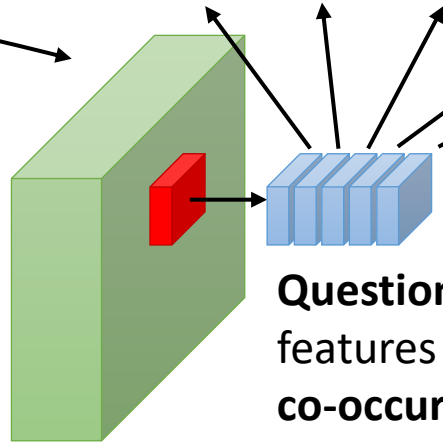
style

relationships between features

e.g., lots of angles and straight lines







Question: which features tend to **co-occur**?

"each time we see a straight line, we also see this texture"

"each time we see a curve, we also see flat shading"

How do we quantify this?

Images from CS231n (Fei-Fei Li, Andrej Karpathy)

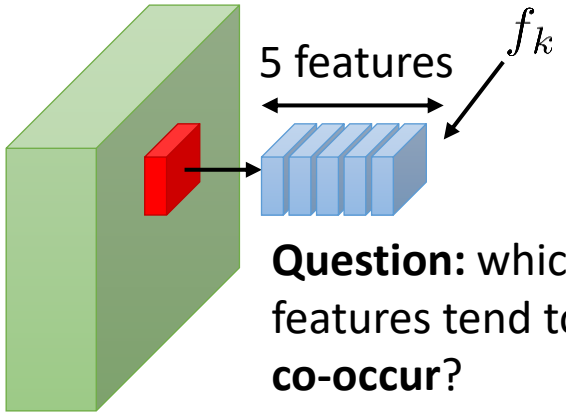
How do we quantify **style**?

style

relationships between features

e.g., lots of angles and straight lines

estimate by averaging over all positions in the image



Question: which features tend to **co-occur**?

$$\text{feature covariance: } \text{Cov}_{km} = E[f_k f_m]$$

$$\text{form Gram matrix: } G_{km} = \text{Cov}_{km}$$

If features have this covariance, then we have the right style

Surprising but true!

"each time we see a straight line, we also see this texture"

"each time we see a curve, we also see flat shading"

Images from CS231n (Fei-Fei Li, Andrej Karpathy) Gatys et al. A Neural Algorithm of Artist Style, 2015.

How do we quantify **style**?

style

relationships between features

e.g., lots of angles and straight lines

feature covariance: $\text{Cov}_{km} = E[f_k f_m]$

form Gram matrix: $G_{km} = \text{Cov}_{km}$ ← this comes from the style source image

new image: $x \leftarrow \arg\min_x \mathcal{L}_{\text{style}}(x) + \mathcal{L}_{\text{content}}(x)$

G^ℓ: source image Gram matrix at layer ℓ

$A^\ell(x)$: new image Gram matrix at layer ℓ

$$\mathcal{L}_{\text{style}}(x) = \sum_\ell \sum_{km} \left(G^\ell_{km} - A^\ell_{km}(x)\right)^2 w_\ell$$

Different weight on each layer, to prioritize relative contribution to (desired) style of different levels of abstraction

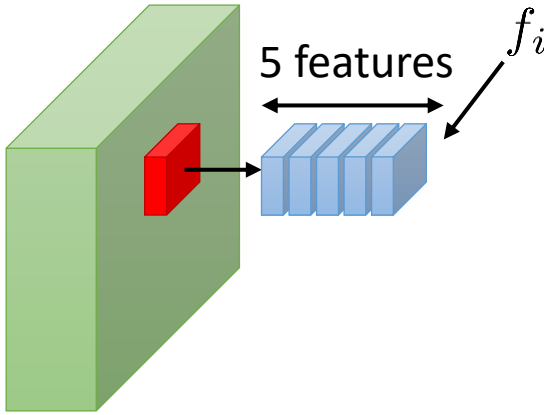
Images from CS231n (Fei-Fei Li, Andrej Karpathy) Gatys et al. A Neural Algorithm of Artist Style, 2015.

How do we quantify **content**?

content

spatial positions of features

e.g., where curves, edges, and corners are located



Pick specific layer for matching content

Just directly match the features!

$$\mathcal{L}_{\text{content}}(x) = \sum_{ij} \sum_{k} \left(f^{\ell}_{ijk}(x_{\text{content}}) - f^{\ell}_{ijk}(x) \right)^2$$

$$\mathcal{L}_{\text{style}}(x) = \sum_{\ell} \sum_{km} \left(G^{\ell}_{km} - A^{\ell}_{km}(x) \right)^2 w_{\ell}$$

$$\text{new image: } x \leftarrow \arg\min_x \mathcal{L}_{\text{style}}(x) + \mathcal{L}_{\text{content}}(x)$$

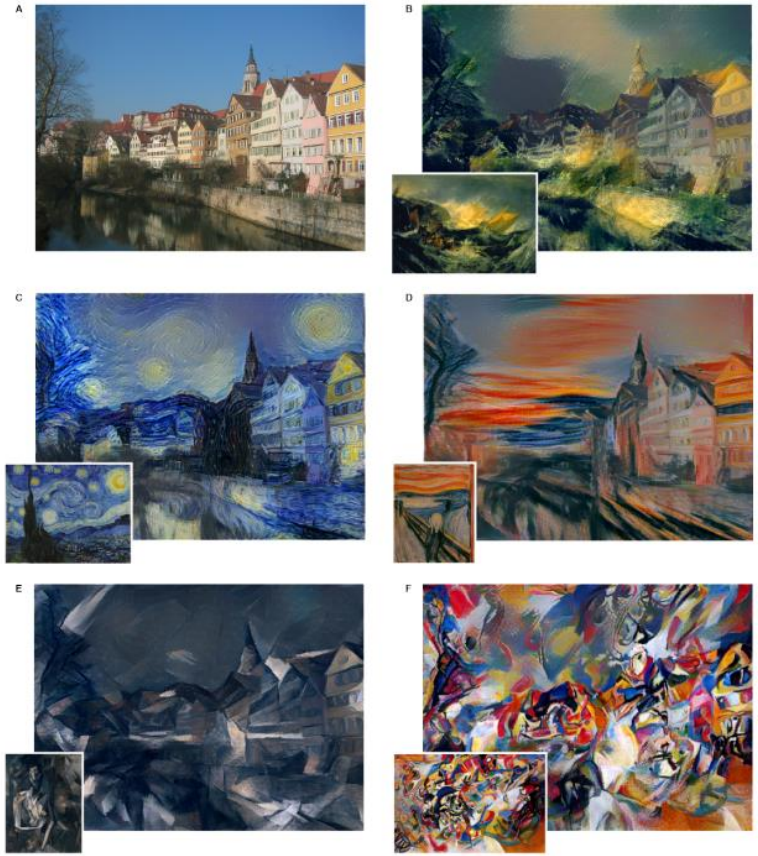
Images from CS231n (Fei-Fei Li, Andrej Karpathy) Gatys et al. A Neural Algorithm of Artist Style, 2015.

Style transfer

$$\text{new image: } x \leftarrow \arg\min_x \mathcal{L}_{\text{style}}(x) + \mathcal{L}_{\text{content}}(x)$$

$$\mathcal{L}_{\text{content}}(x) = \sum_{ij} \sum_{k} \left(f^{\ell}_{ijk}(x_{\text{content}}) - f^{\ell}_{ijk}(x)\right)^2$$

$$\mathcal{L}_{\text{style}}(x) = \sum_{\ell} \sum_{km} \left(G^{\ell}_{km} - A^{\ell}_{km}(x)\right)^2 w_{\ell}$$



make your own easily on deepart.io

Gatys et al. A Neural Algorithm of Artist Style, 2015.