Reliable and Interpretable Artificial Intelligence

Lecture 10: Visualization

Marc Fischer

ETH Zurich

Fall 2020

SRILAB

http://www.sri.inf.ethz.ch

ETH

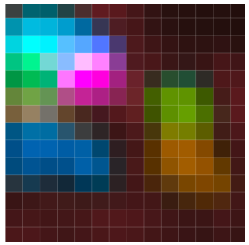
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Aspects of Reliable AI

AI Safety & Robustness

Verification of safety and robustness properties; training that enforces them.

e.g. Adversarial Robustness, Robust Training, Querying and Constraining Neural Nets (covered in upcoming and previous lectures)



Interpretability

Understanding the models decision making process.

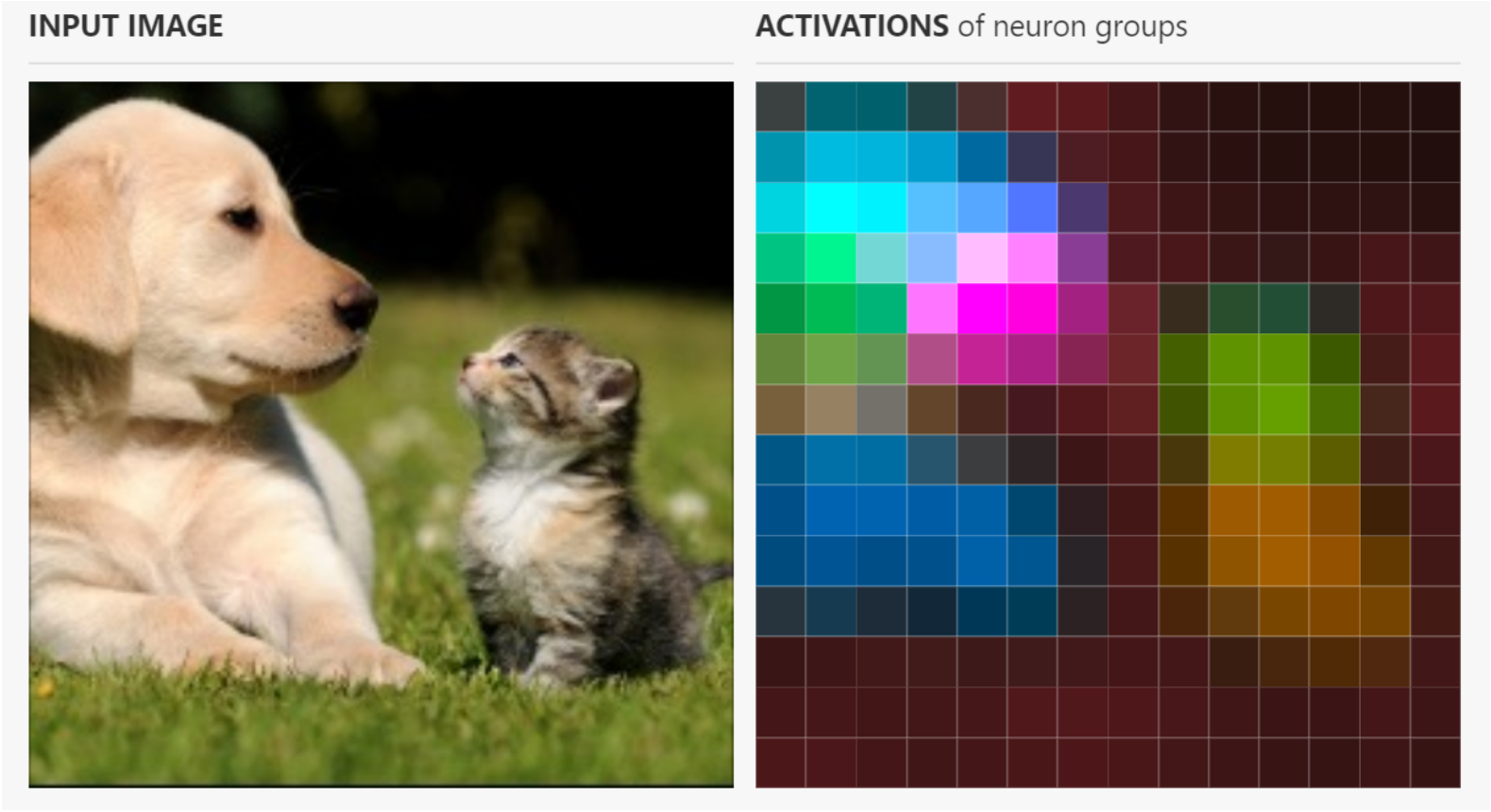
e.g. Attention Mechanisms, Neuro-symbolic Approaches, **Visualization of decision making (today)**

Privacy & Fairness

Data of individuals in produced and they are not discriminated,

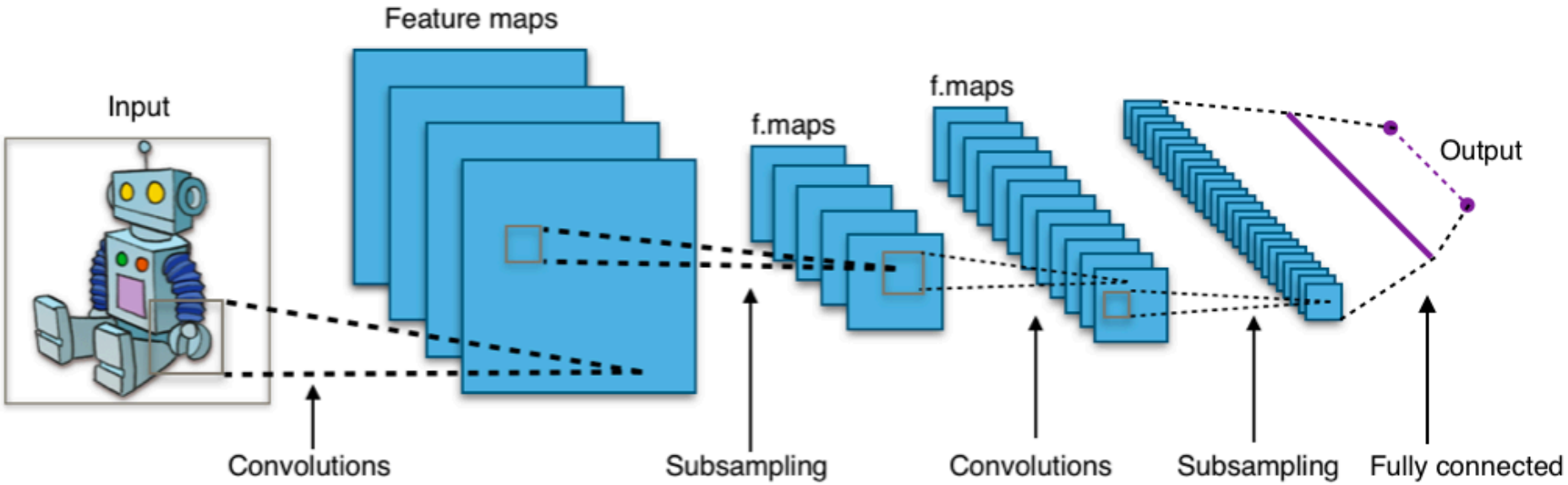
e.g. Differential privacy, Enforcing Fairness in Neural Networks (covered in upcoming lectures)

Today: Visualizing Classification



Olah et al., 2018

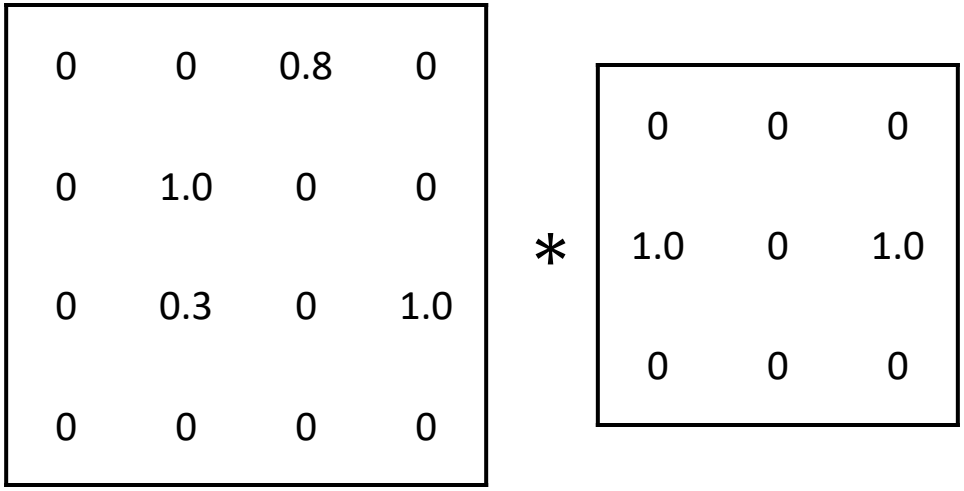
Convolutional Neural Net (CNN)



- Convolutional layers: $f^{(k)}(x) = ReLU(x * F + b)$
- Linear layers: $f^{(k)}(x) = ReLU(A \cdot x + b)$

[source: Wikipedia by user Aphex34]

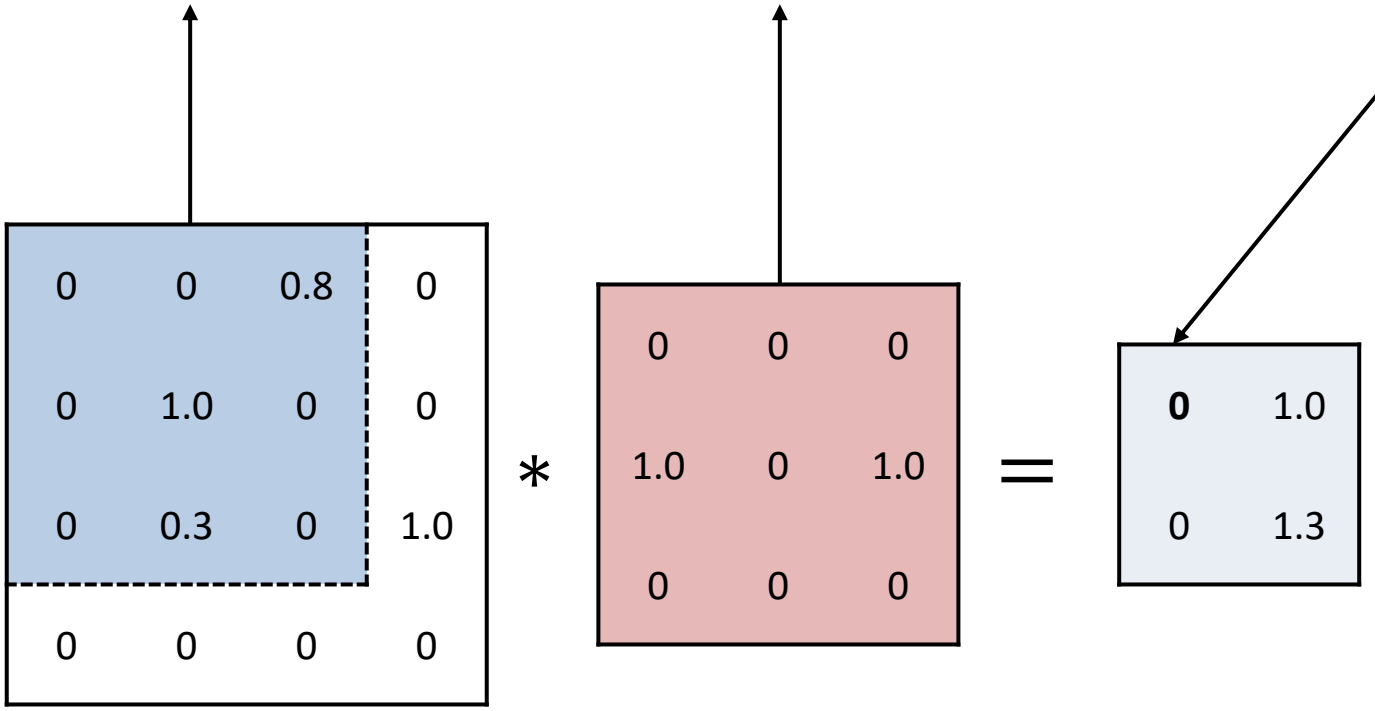
Example: 2D Convolution



$x \in \mathbb{R}^{4 \times 4}$ $F \in \mathbb{R}^{3 \times 3}$

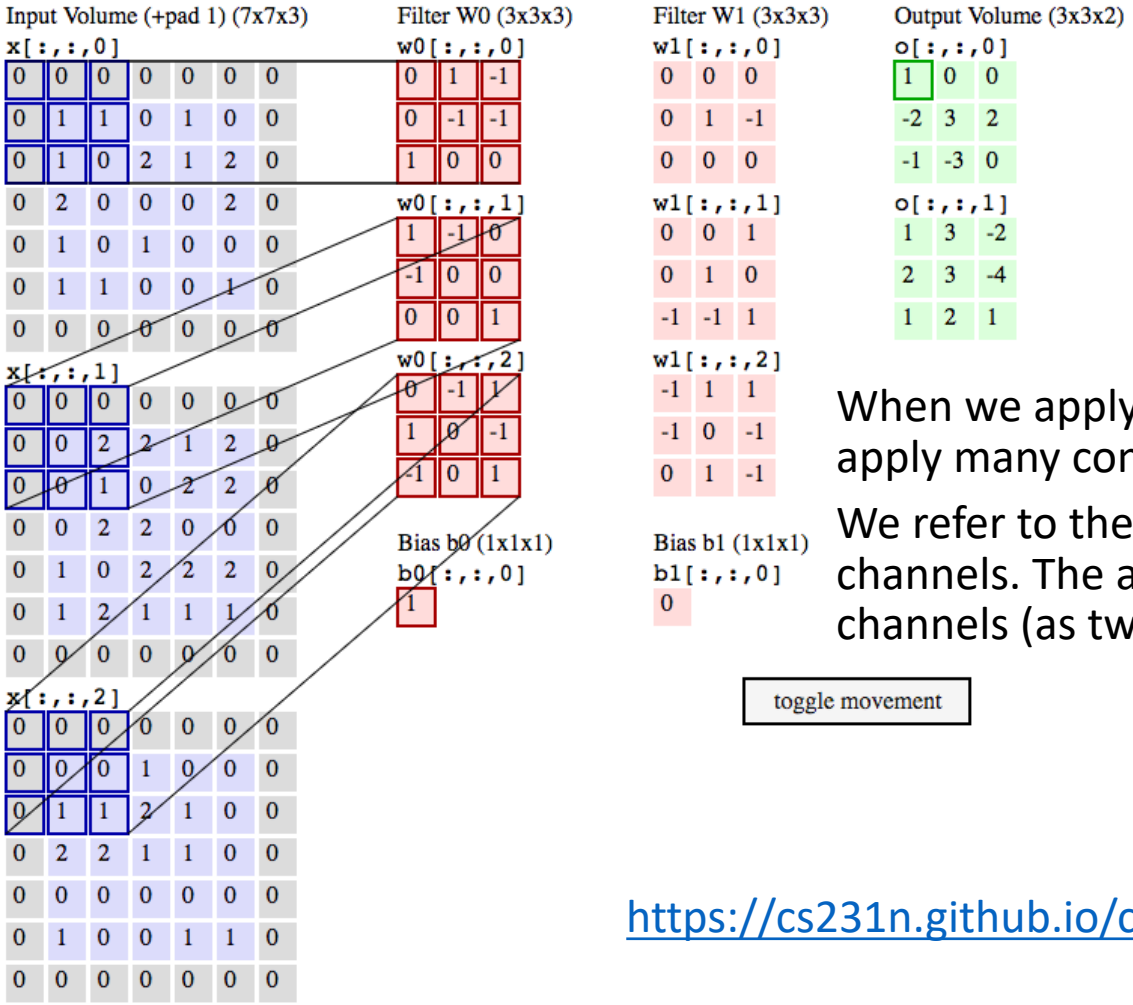
Example: 2D Convolution

$0 \cdot 0 + 0 \cdot 0 + 0.8 \cdot 0 + 0 \cdot 1.0 + 1.0 \cdot 0 + 0 \cdot 1.0 + 0 \cdot 0 + 0.3 \cdot 0 + 0 \cdot 0 = 0$



$x \in \mathbb{R}^{4 \times 4}$ $F \in \mathbb{R}^{3 \times 3}$

Example: 2D Convolution



When we apply a convolutional layer we in fact apply many convolutions in parallel

We refer to the to the different results as channels. The above green output has two channels (as two filters were used).

toggle movement

https://cs231n.github.io/convolutional-networks/

Convolution

- Convolutions (formally cross-correlations) have a high response if the local pattern of the filter (kernel) matches the pattern in the data
- For example: a matrix encoding high contrast between left and right (shown below) yields a vertical edge detector

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

AlexNet

- 2012 - Krizhevsky et al. “AlexNet”
  - low error on ImageNet benchmark
  - Observation: feature similarity

“If two images produce feature activation vectors with a **small Euclidean separation**, we can say that the higher levels of the neural network consider them to be **similar**.”

Krizhevsky et al., 2012

AlexNet

- Let $R(x)$ be the vector of neuron activations for a layer deep in the network
- If the distance $\|R(x_1) - R(x_2)\|_2$ is small, then the images x_1, x_2 are similar

1st column: image from test set

others: 6 training set images with most similar $R(x)$

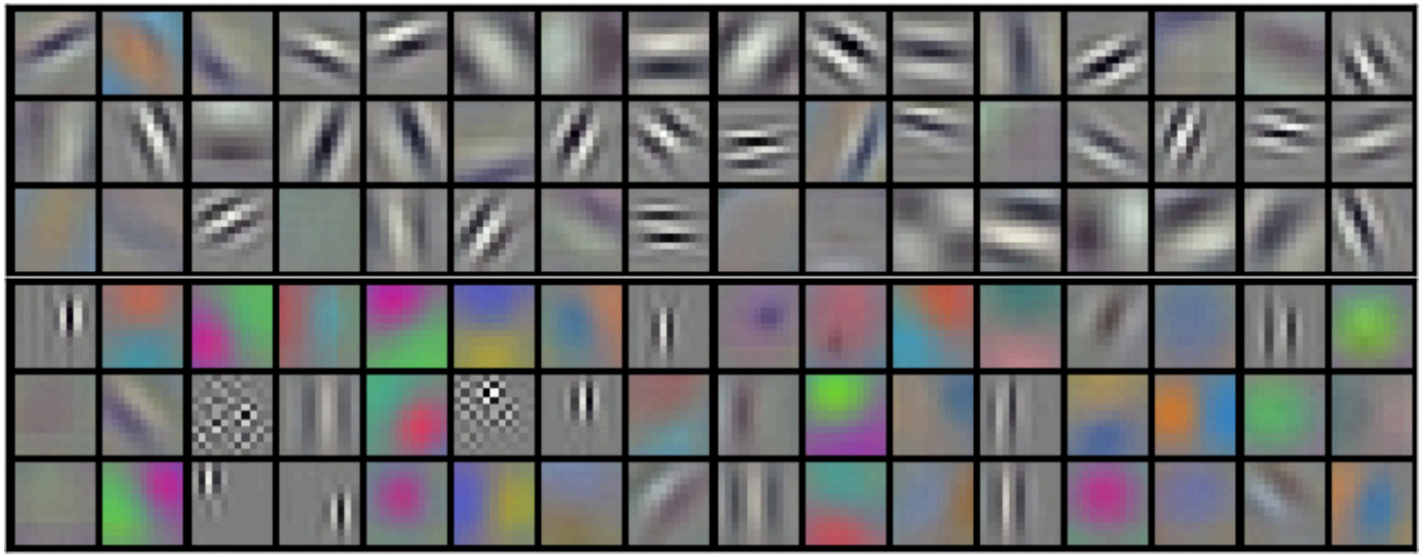
Krizhevsky et al., 2012

Features

- Consequence: neuron activations capture semantic features of the input
- **Feature Visualization:** For a given neural network, what are these features?
- **Feature Attribution:** Which areas of a given image are responsible for classification in a given neural network?

AlexNet: First Layer Features

- Weights of the first convolutional layer (11x11x3) shown as color images
- Detectors for edges, simple patterns and color blobs



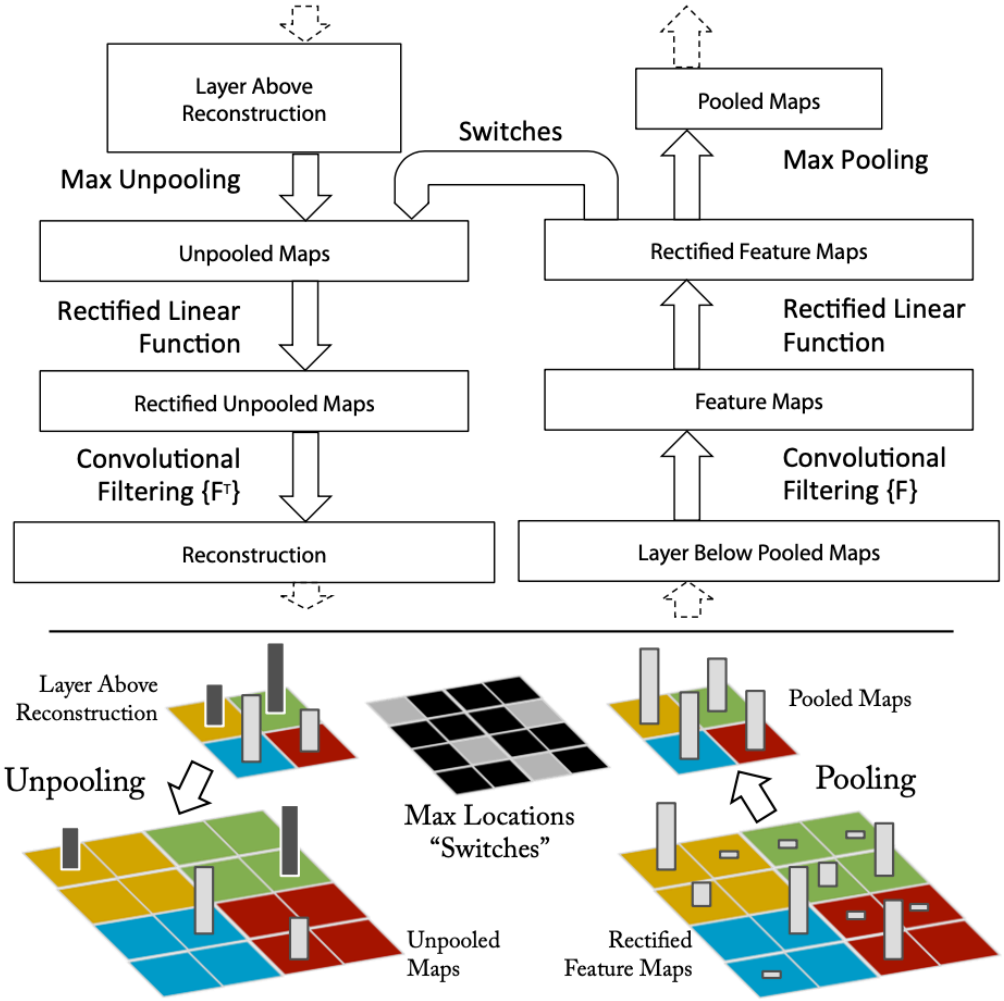
Krizhevsky et al., 2012

Inverting Convolutions

- Inspect neural networks in order to improve them
- For a given input and convolutional filter:
  - inverts convolutions to showcase the structure in the input causing high activation
  - first propagate input forward to the filter
  - then propagate these activations back to the input layer in the inverted network (shown next)
  - the outputs highlight which part of the image are responsible for the activation of the filter

Zeiler et al., 2013

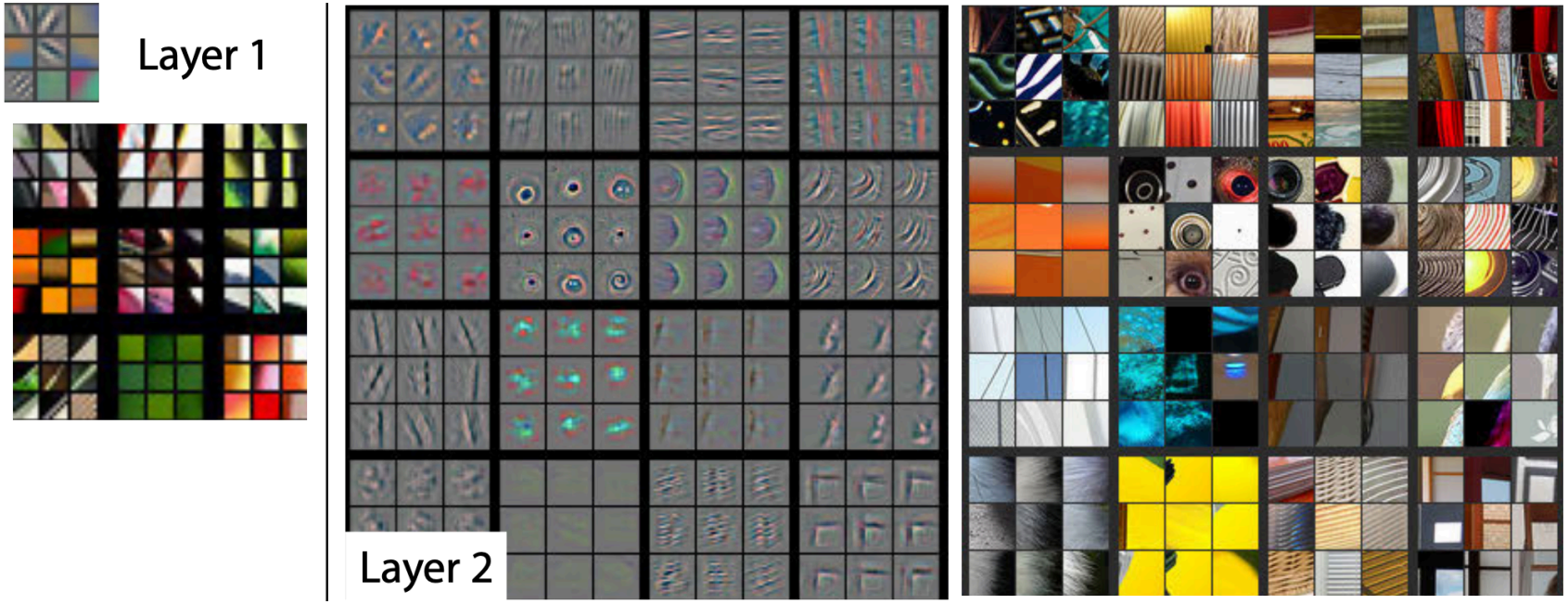
Inverting Convolutions



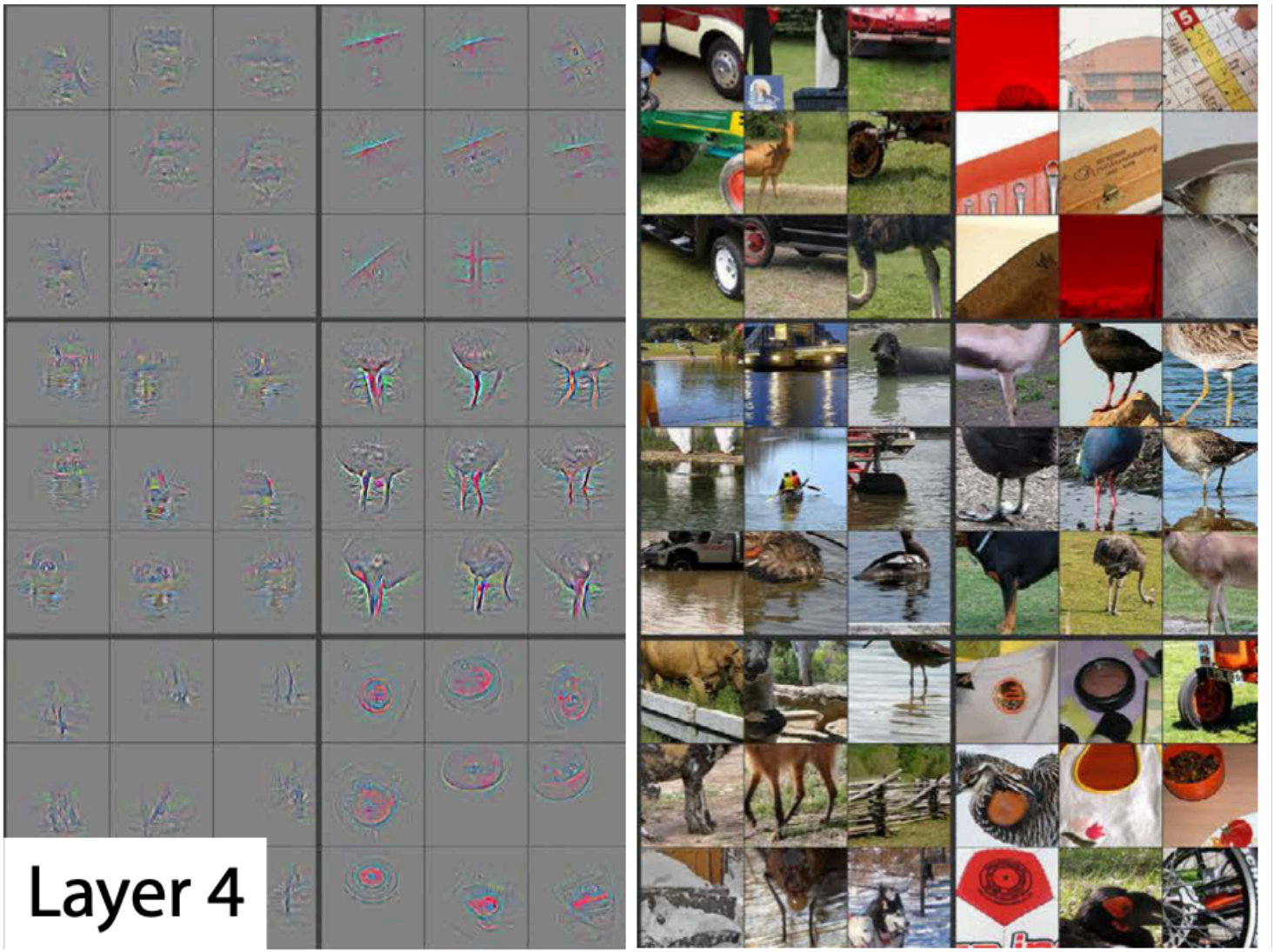
Zeiler et al., 2013

Inverting Convolutions

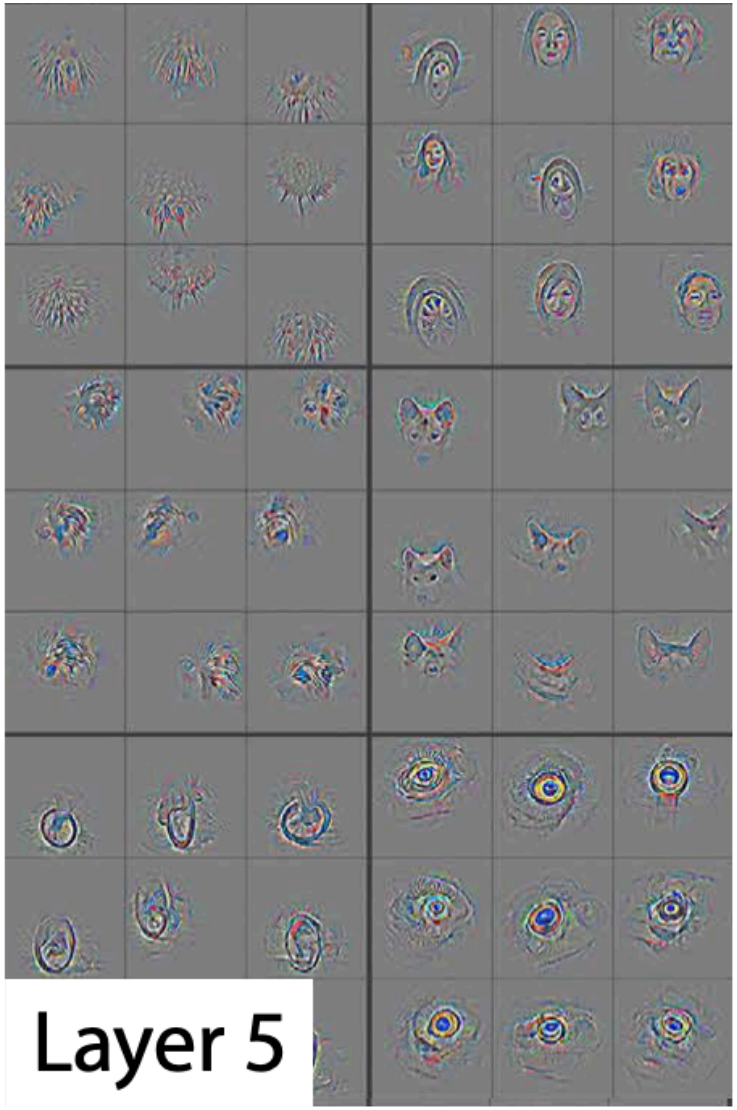
- For a random set of convolutional filters in each layer, this shows the inputs causing the highest activations in the test set
- Pattern on the left reveals structure causing activation

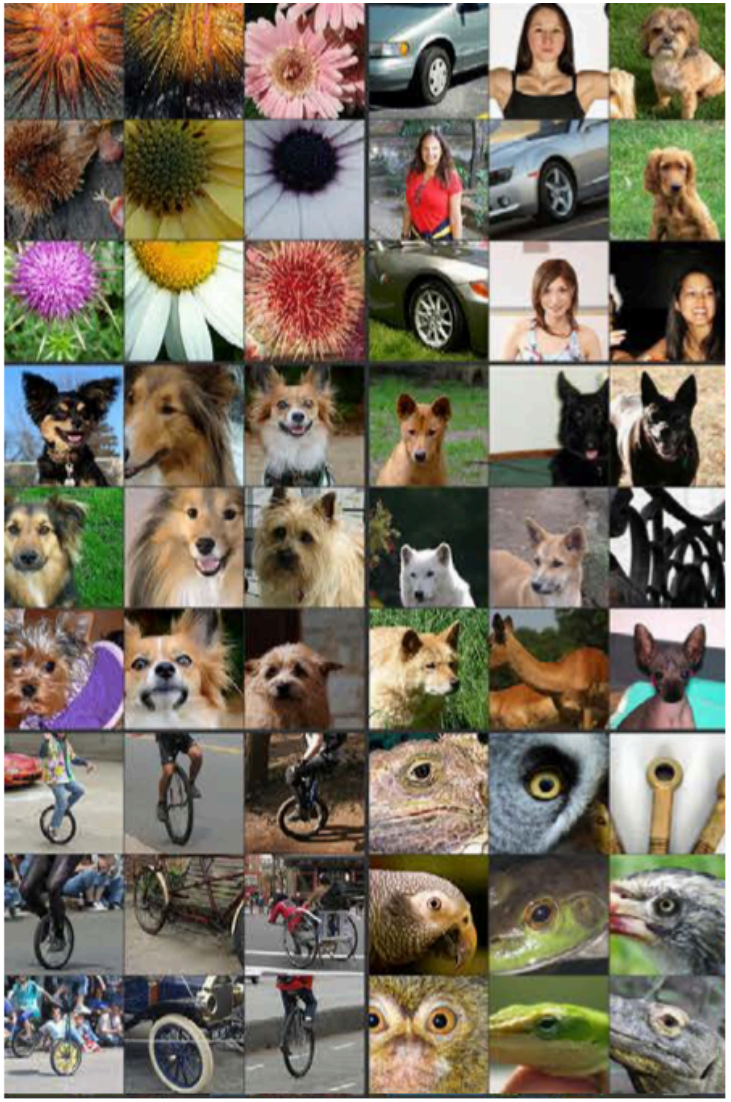


Zeiler et al., 2013



Zeiler et al., 2013





Zeiler et al., 2013

Feature Visualization by Optimization

find x

maximize $score(x) - \lambda_1 R_1(x) - \ ... - \lambda_m R_m(x)$

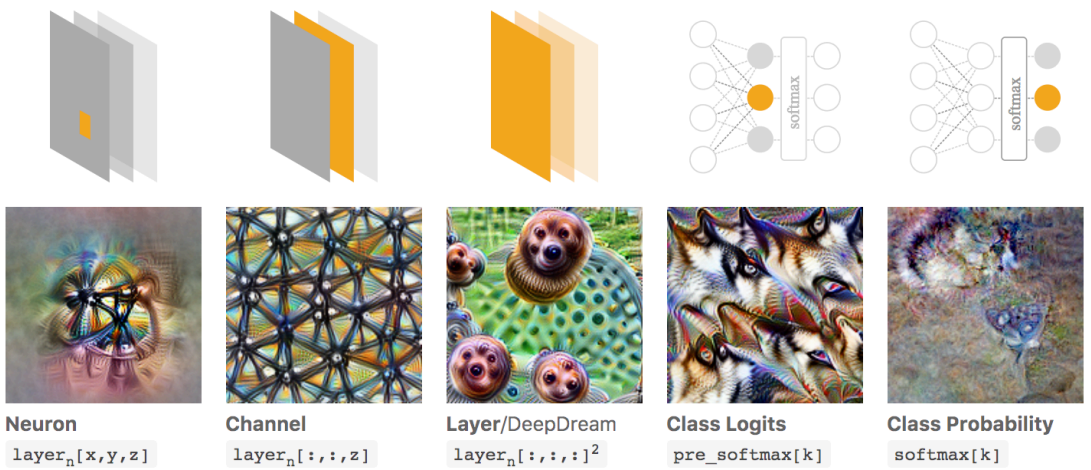
where $score(x) = mean(layer_n[x, y, z])$

Find input x that maximizes score and minimized regularizes

Score denotes the activation of a neuron, channel or layer

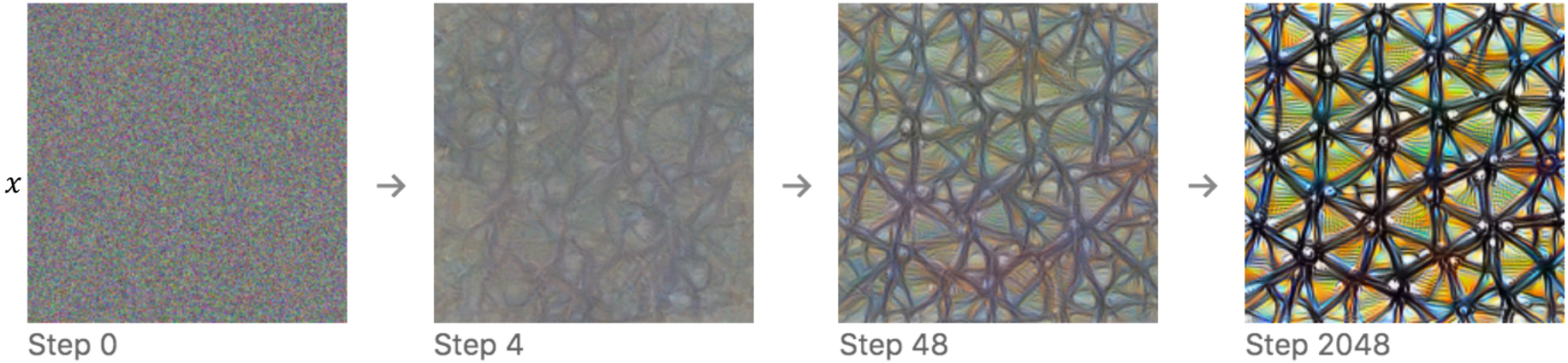
Different **optimization objectives** show what different parts of a network are looking for.

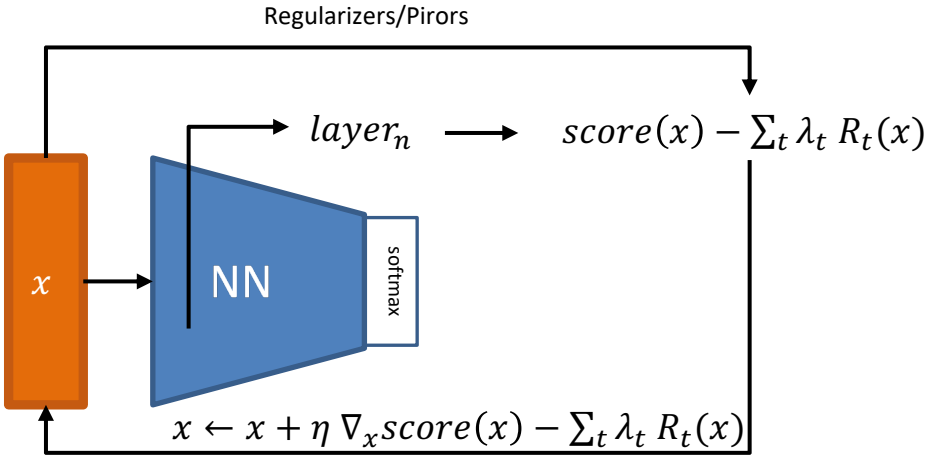
`n` layer index
`x,y` spatial position
`z` channel index
`k` class index



Olah et al., 2017; Erhan et al. 09

Feature Visualization by Optimization





Olah et al., 2017

Feature Visualization by Optimization

- Regularizers are crucial
  - Encode the prior “input should look a real image”
  - Optimization in Fourier basis, data whitening, added jitter etc.
- Optimize for multiple inputs at once
  - add diversity term loss to discourage similarity
  - multiple visualizations for one neuron

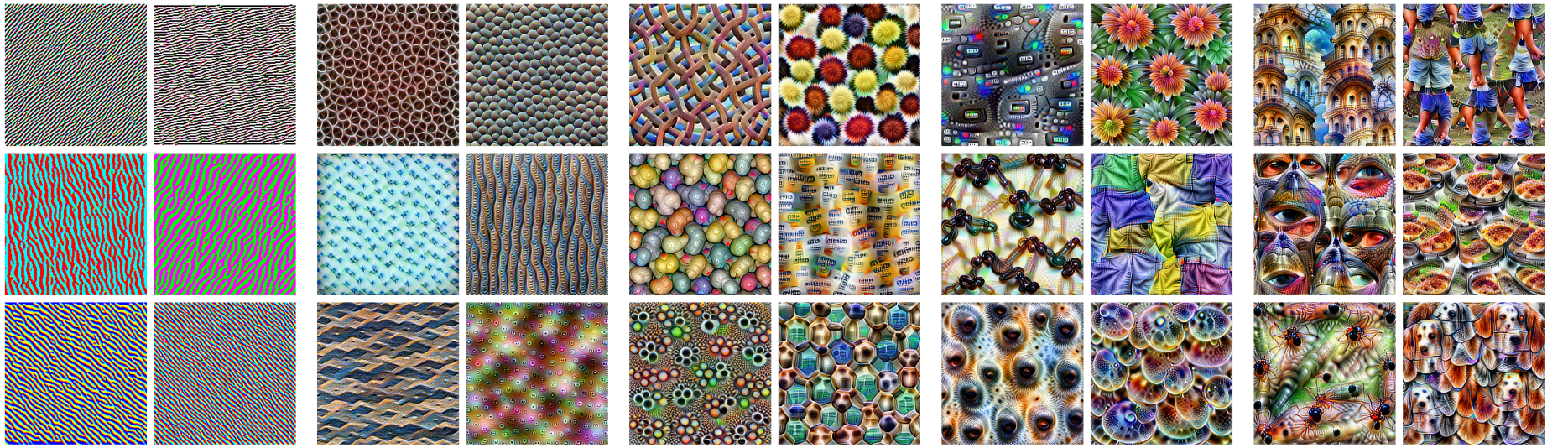
Simple Optimization

Optimization with diversity reveals four different, curvy facets. *Layer mixed4a, Unit 97*

Dataset examples

Olah et al., 2017

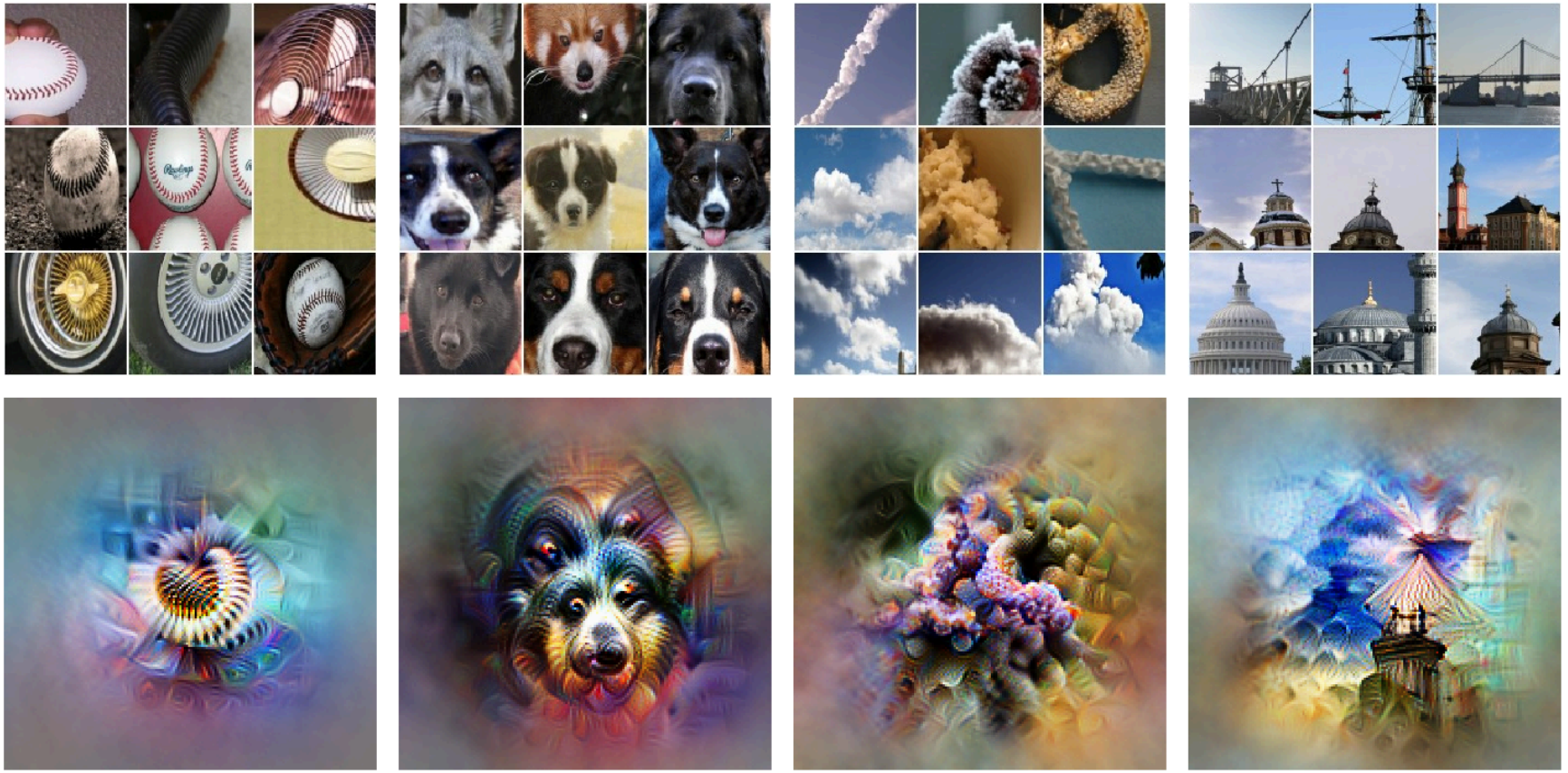
Feature Visualization by Optimization



Edges (layer conv2d0) **Textures** (layer mixed3a) **Patterns** (layer mixed4a) **Parts** (layers mixed4b & mixed4c) **Objects** (layers mixed4d & mixed4e)

Olah et al., 2017

Feature Visualization by Optimization



Baseball—or stripes?
mixed4a, Unit 6

Animal faces—or snouts?
mixed4a, Unit 240

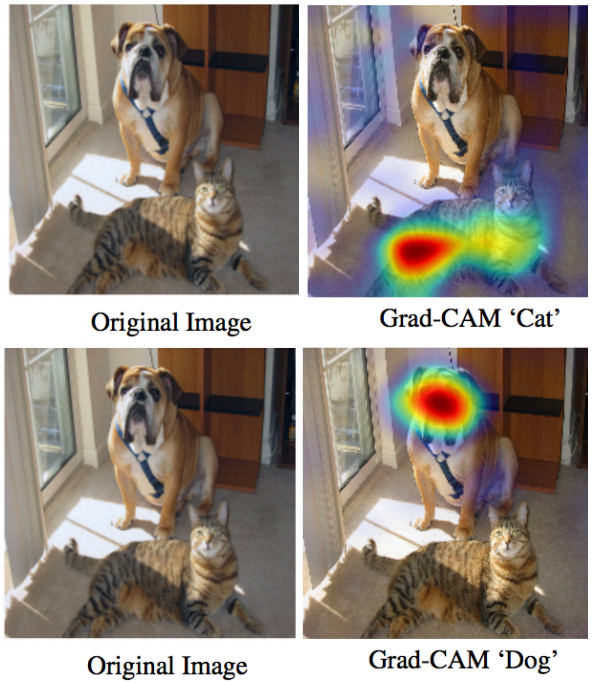
Clouds—or fluffiness?
mixed4a, Unit 453

Buildings—or sky?
mixed4a, Unit 492

Olah et al., 2017

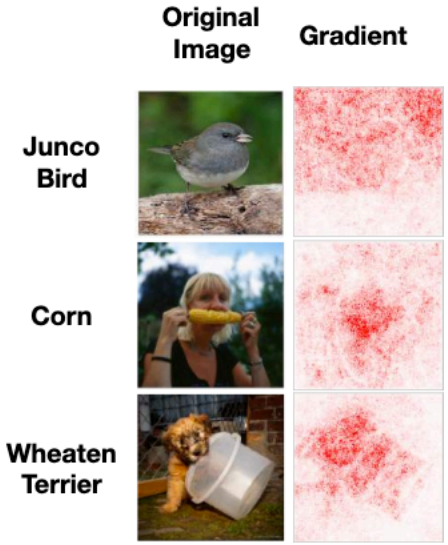
Feature Attribution

- Which areas of the image are responsible for classification?
- Many techniques
  - Gradient based
  - Shapley values
  - Occlusion based
  - Etc.



Selvaraju et al., 2017

Feature Attribution: Gradient Based

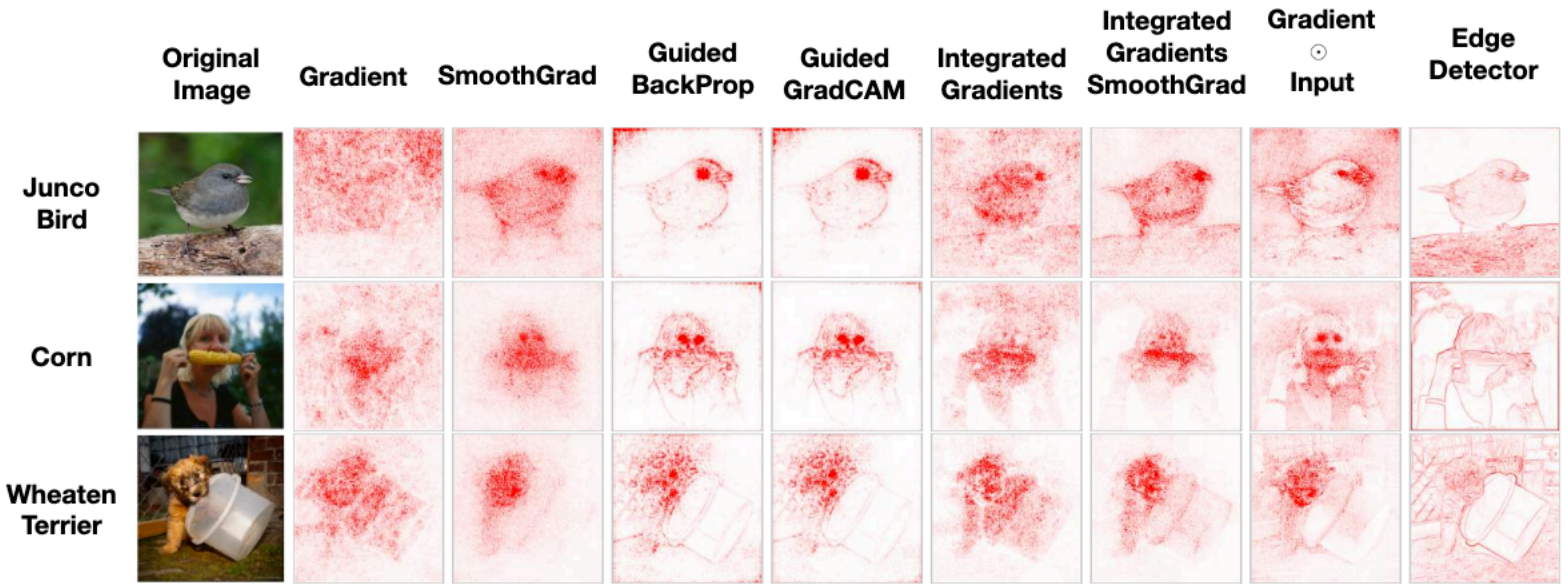


Gradient of target logit w.r.t. the input shows how changing single pixels influences the logit

$$\frac{\partial logit_t(x)}{\partial x}$$

Adebayo et al., 2018

Feature Attribution: Gradient Based



- To obtain a clearer result
  - Different methods apply scaling/conditioning of the gradient
- Just looking at the visualization is dangerous: many of these methods were found not to depend (much) on the actual weights in higher layers of the network
  - the gradient, by it's definition, however does

Adebayo et al., 2018

Feature Attribution: Shapley Values (general)

- Input x has a set of features P (e.g. individual pixels)
- Want to know how much each feature contributes to a function $f(P)$ (e.g. a class logit)
- Classic result from game theory
- Theoretically justified

$$\underbrace{C_i}_{\text{attribution to feature i}} = \underbrace{\sum_{S \subseteq P \setminus \{i\}}}_{\text{all subsets}} \underbrace{\frac{|S|!\,(|P| - |S| - 1)!}{|P|!}}_{\text{average}} \underbrace{[f(S \cup \{i\}) - f(S)]}_{\text{contribution to feature i}}$$

$\left(= \frac{1}{|P|} \cdot \binom{|P|-1}{|S|}^{-1}\right)$

Shapley 1952; Lundberg & Lee 2017

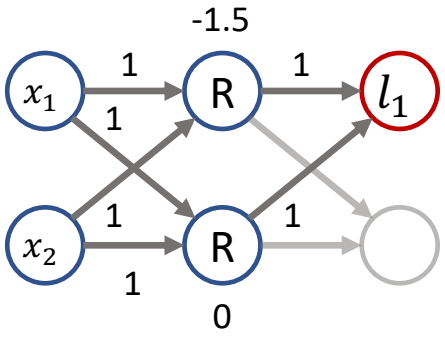
Feature Attribution: Shapley Values (image classification)

- Treat each pixel location as a feature
- Instantiate f with the logit of the target class (or the classification loss)
- Define meaning of $f(S)$
  - E.g. pixels locations not contained in S are set to a baseline value (zero, mean pixel value etc.)
- Computationally expensive
  - Exponentially many subsets
  - Can be approximated in polynomial time

Ancona et al. 2019

Example: Shapley Values

- Want to compute the influence of x_1, x_2 on l_1 for $x = \binom{1}{1}$, R denotes ReLU
- Define $f(S)$ as the value of l_1 when inputs not in S are set to 0
- $P = \{x_1, x_2\}$



$$C_i = \sum_{S \subseteq P \setminus \{i\}} \underbrace{\frac{|S|!\,(|P| - |S| - 1)!}{|P|!}}_{\kappa(S)} [f(S \cup \{i\}) - f(S)]$$

S	$f(S)$
$\emptyset$	0
$\{x_1\}$	1
$\{x_2\}$	1
$\{x_1, x_2\}$	2.5

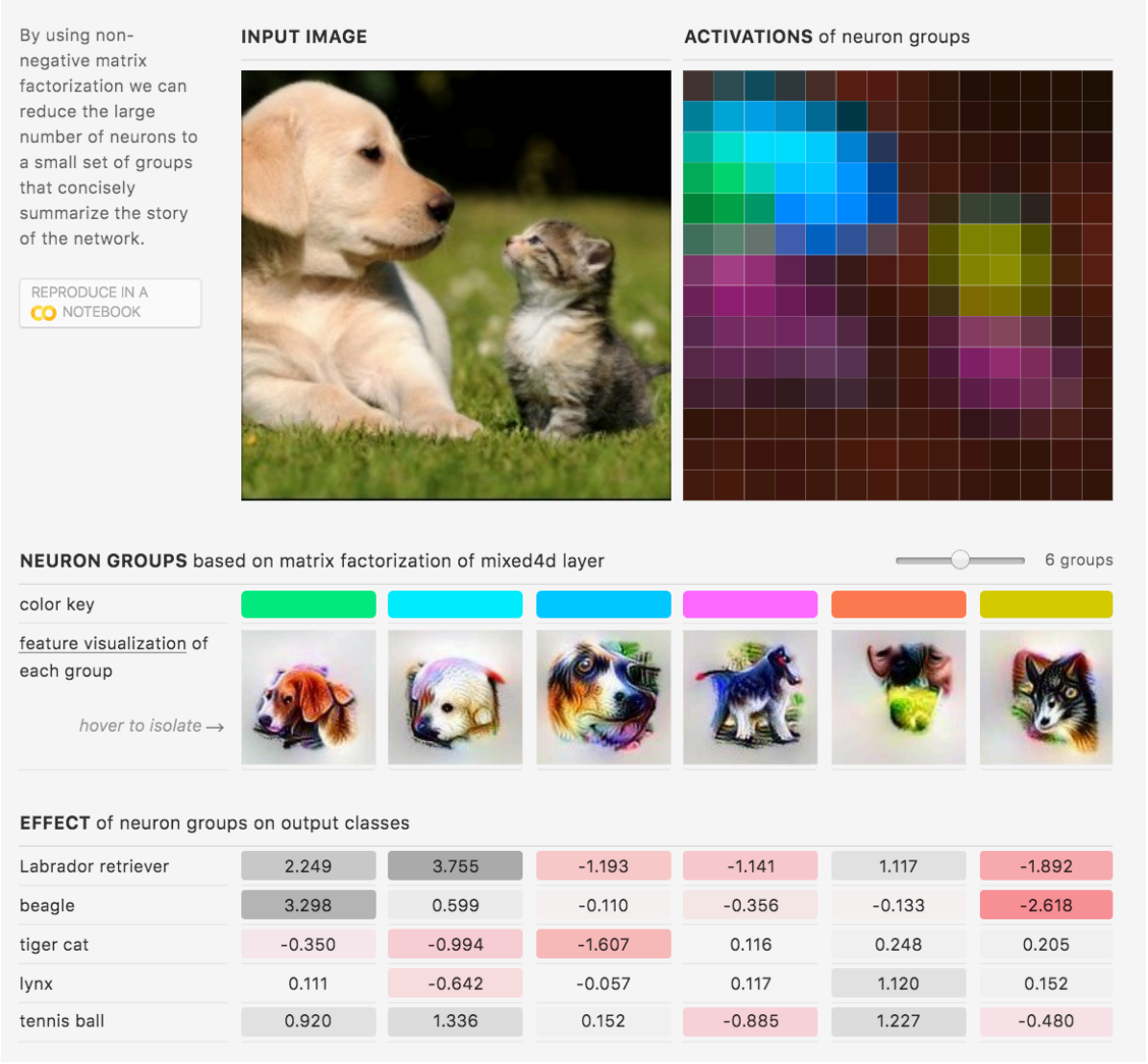
$$C_{x_1} = \kappa(\emptyset)\,[f(\{x_1\}) - f(\emptyset)] + \kappa(\{x_2\})\,[f(\{x_1, x_2\}) - f(x_2)]$$
$$= \frac{0!(2-0-1)!}{2!}[1-0] + \frac{1!(2-1-1)!}{2!}[2.5 - 1] = \frac{1}{2}(1 + 1.5) = 1.25$$
$$C_{x_2} = 1.25$$

x_1, x_2 both contribute 1.25 to $l_1 = 2.5$ for $x = \binom{1}{1}$

Combining Viualization & Attribution

1. Compute activations in a convolutional layer
2. Clusters activations k groups (here k = 6)
3. For each group:
    1. show which pixel affect it most by upscaling the activation map (grouped activations)
    2. show a visualization for the group of features

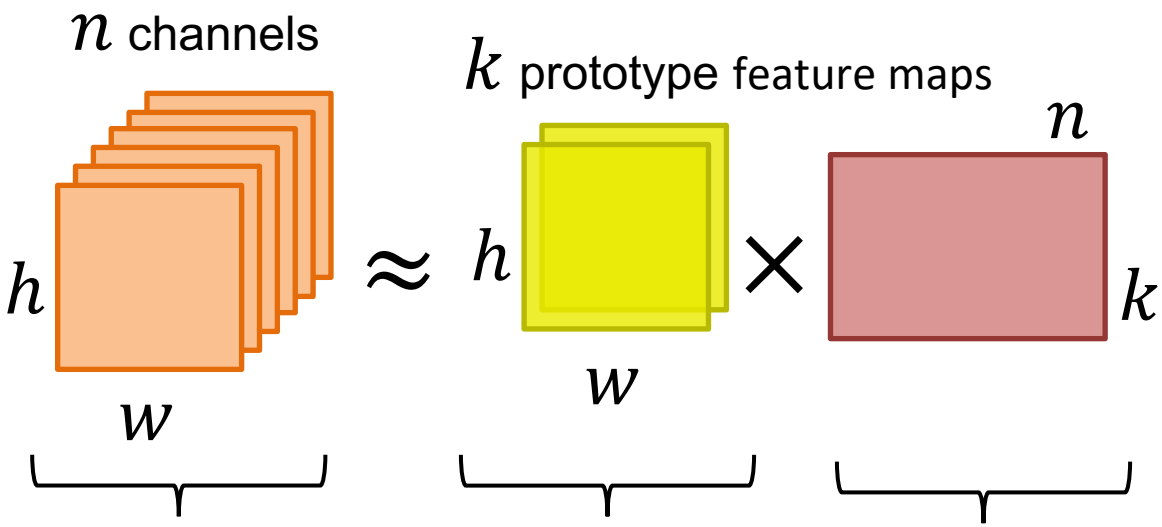
This allows for “human-scale” explanations (k groups rather than thousands)



Olah et al., 2018

Combining Viualization & Attribution

1. Compute activations in a convolutional layer
2. Clusters activations k groups (here k = 6)
3. For each group:
    1. show which pixel affect it most by upscaling the activation map (grouped activations)
    2. show a visualization for the group of features



n maps, each showing the activations of a convolution channel

k maps, each showing the activations of a feature group

k featrure groups; each a mixture of the n original convolution channels

This allows for "human-scale" explanations (k groups rather than thousands)

Olah et al., 2018

How Do Adversarial Examples Impact Visualization?

Problem: Adversarial Examples

- Recall: If the distance $\|R(x_1) - R(x_2)\|_2$ is small, then the images x_1, x_2 are similar
- Yet x and it's adversarial example x' are similar, but $R(x)$ and $R(x')$ are different
- Can we even find dissimilar x_1, x_2 with similar $R(x_1)$ and $R(x_2)$?

Counterexample

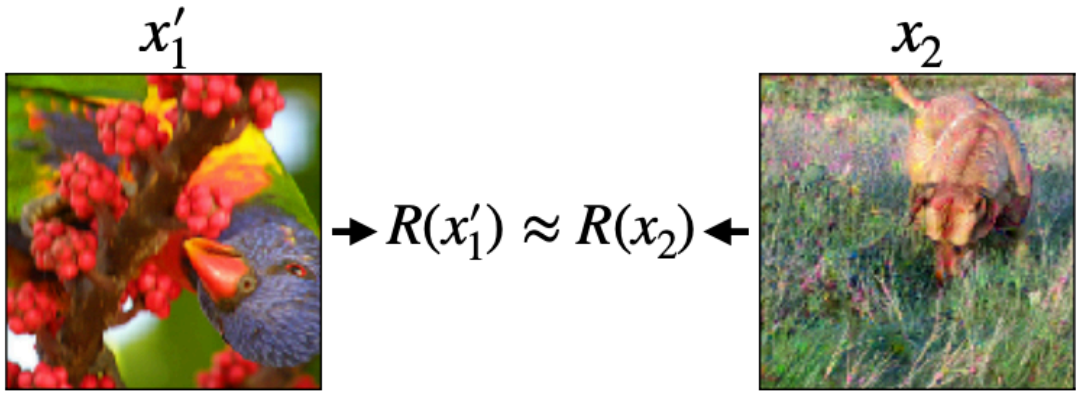
find	$x_1' := x_1 + \delta$
minimize	$\|R(x_1') - R(x_2)\|_2$

- Solve via gradient decent
  - here: projected gradient decent (PGD), similar to the attack
  - take a step toward the objective, project back on the region defined step size
  - no overall projection on a constraint (as in the attack)
- Expectation: x_1' is visually similar to x_2 (no constraint on δ)

Engstrom et al., 2019

Counterexample

find	$x_1' := x_1 + \delta$
minimize	$\|R(x_1') - R(x_2)\|_2$

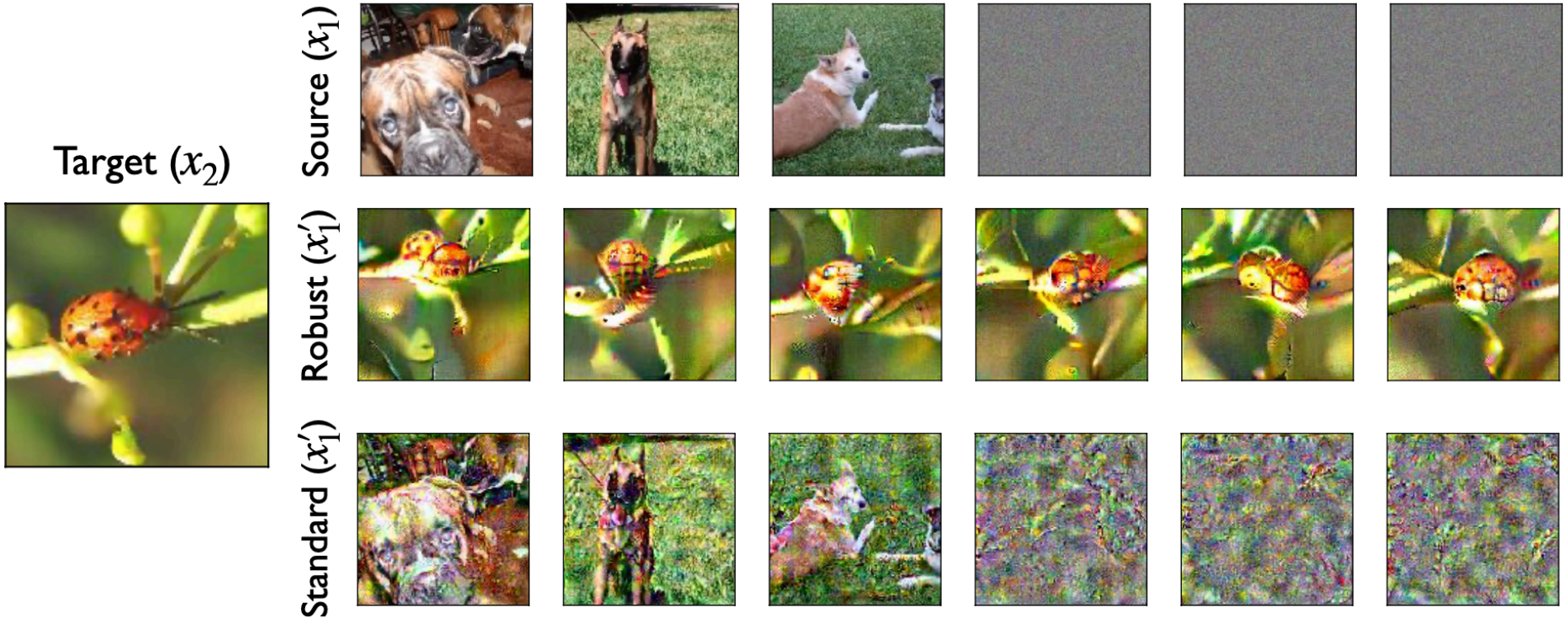


Engstrom et al., 2019

But on a robustly trained network

find	$x_1' := x_1 + \delta$
minimize	$\|R(x_1') - R(x_2)\|_2$

Robust Net: trained with PGD.



Engstrom et al., 2019

Implications for Feature Visualization

- Feature Visualization by optimization requires heavy regularization
  - to encode “looks like an image” prior
  - else optimization produces noise

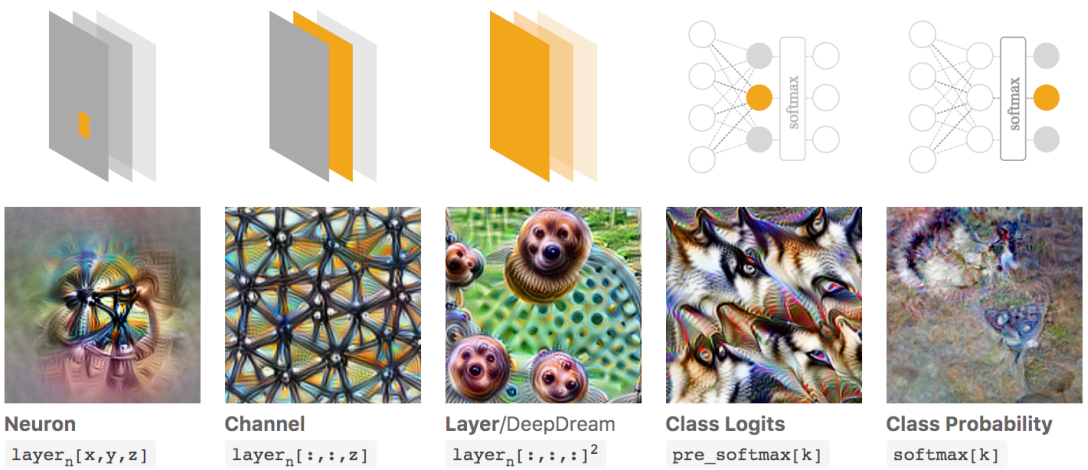
- What happens without regularization?

Implications for Feature Visualization

find	x
maximize	$score(x) - \lambda_1 R_1(x) - \ \ldots - \lambda_m R_m(x)$
where	$score(x) \ = \ mean(layer_n[x, y, z])$

Different **optimization objectives** show what different parts of a network are looking for.

`n` layer index
`x,y` spatial position
`z` channel index
`k` class index



Olah et al., 2017; Erhan et al. 09

Implications for Feature Visualization

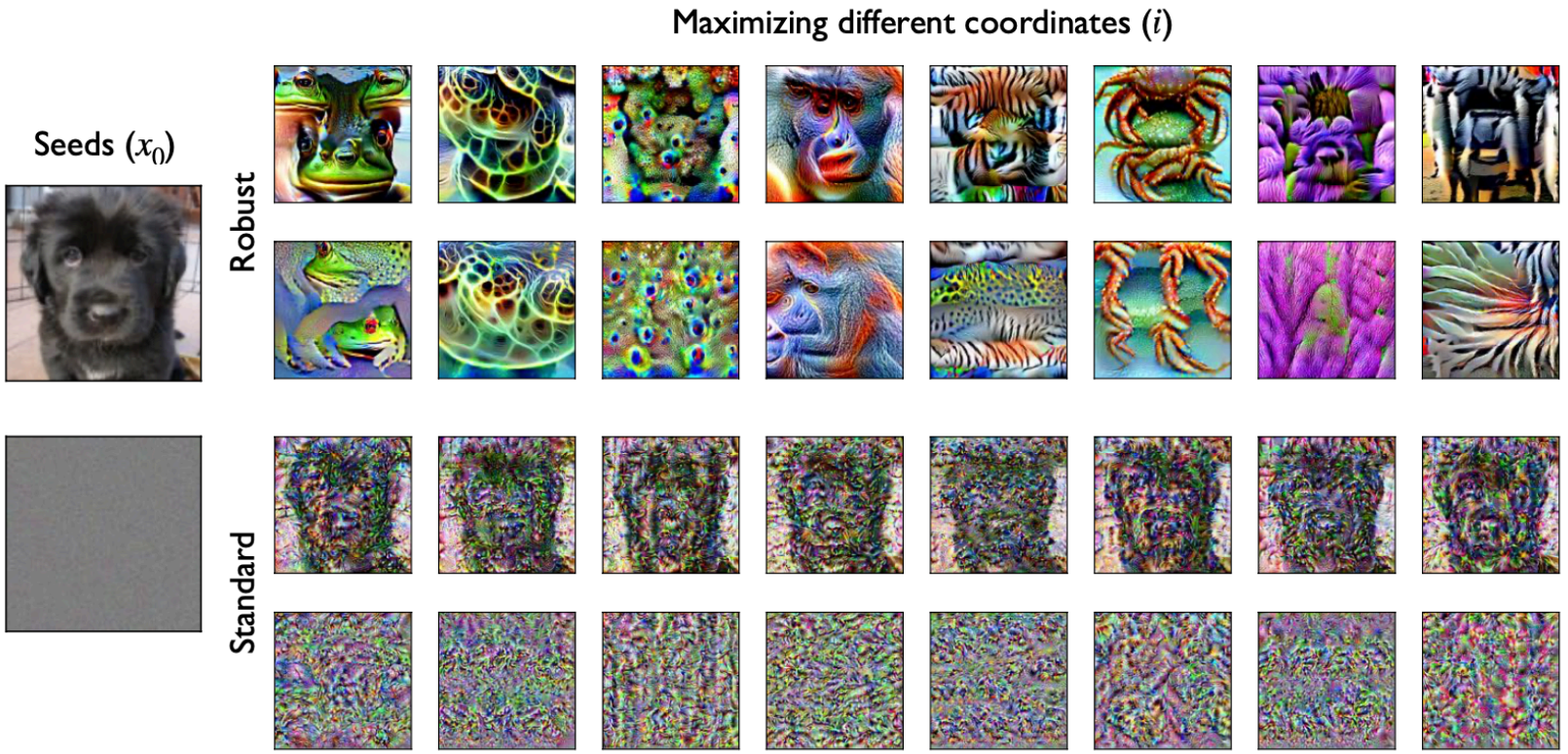
find x

maximize $score(x)$

where $score(x) = mean(layer_n[x, y, z])$

Optimized via PGD

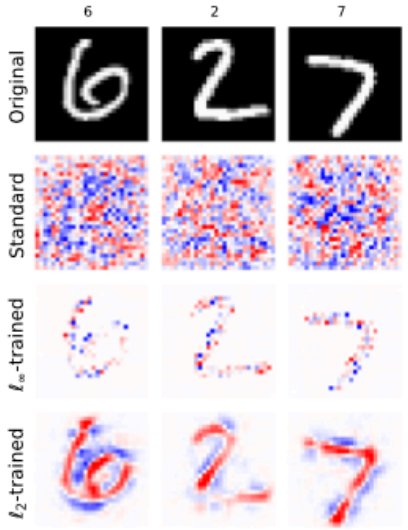
Single neuron here (component of $R(x)$)



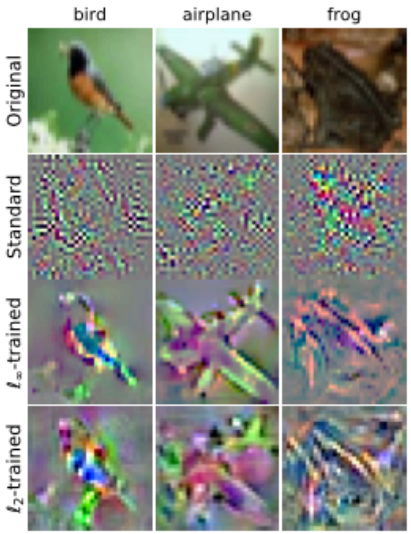
Engstrom et al., 2019

Implications for Feature Attribution

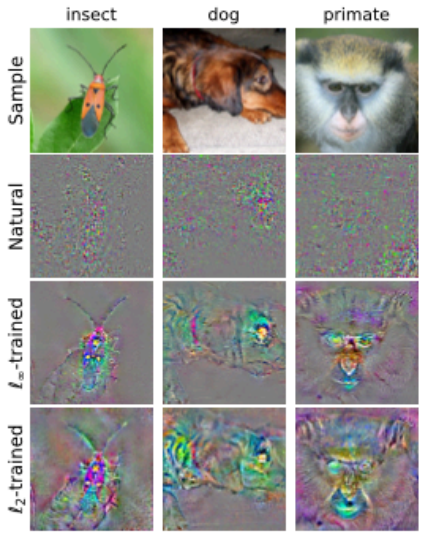
- On non-robust networks gradient based-attribution methods require condition to show a “clean” saliency map
- On robust networks the gradient is better is aligned with human expectation (as the features are better aligned)



(a) MNIST



(b) CIFAR-10



(c) Restricted ImageNet

Gradient of classfication loss w.r.t input

Tsipras et al., 2019

Why?

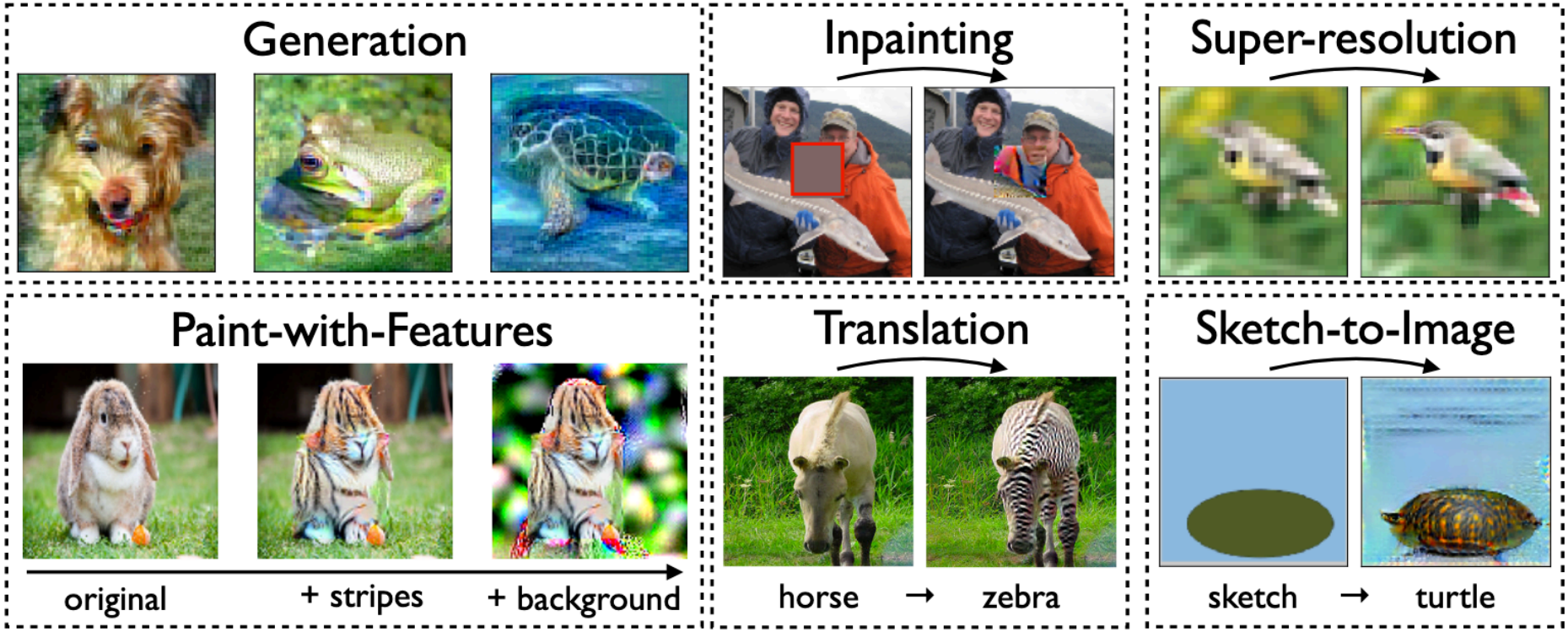
- Robust Neural Networks rely on different features
  - these features are aligned with human perception
  - e.g. these features are robust to small perturbations
  - thus visualizations are what we expect

- Non-Robust Neural Networks learn non-robust spurious features in the data
  - optimization shows these
  - but additional regularization hides them
  - these features may allow for higher accuracy

Tsipras et al., 2019
Ilyas et al., 2019

Consequence: Image Manipulation

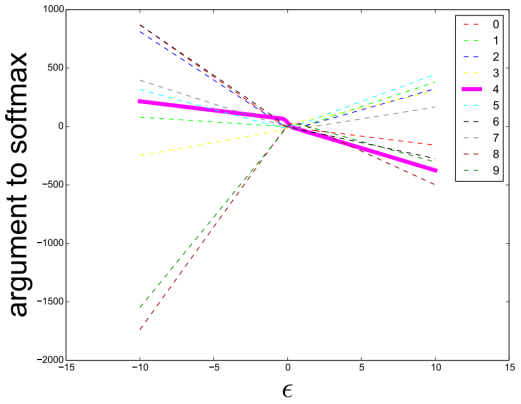
The representation learned by the robust classifier is robust and versatile enough to allow for various computer vision tasks just by optimizing w.r.t. the input..



Santurkar et al., 2019

Why do adversarial examples exist?

- Recall (lecture 2): Neural Networks are too linear

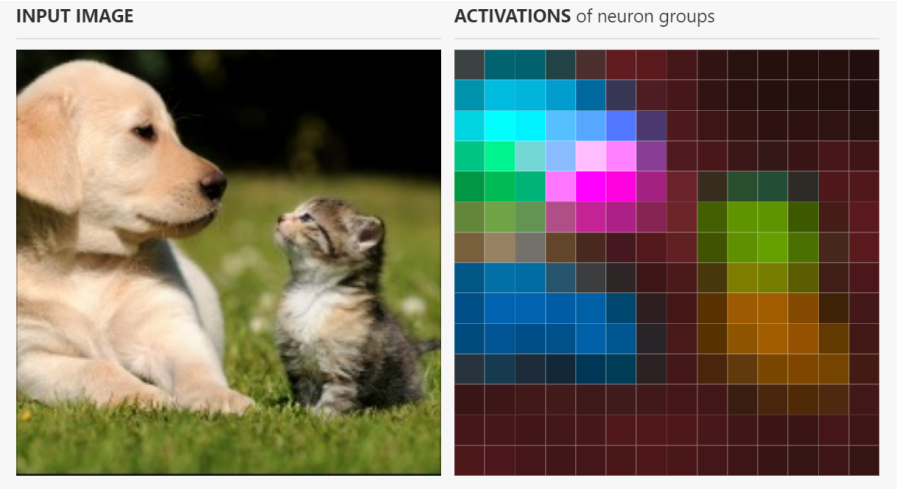


- Additional Reason:
  Non-Robust Neural Networks learn features that statistically are only weakly connected with the input

Goodfellow et al., 2015
Tsipras et al., 2019
Ilyas et al., 2019

Lecture Summary

Feature Visualization & Feature Attribution



Connection to adversarial Robustness