CSCE 636: Deep Learning

The Visual Interpretation for Deep Learning

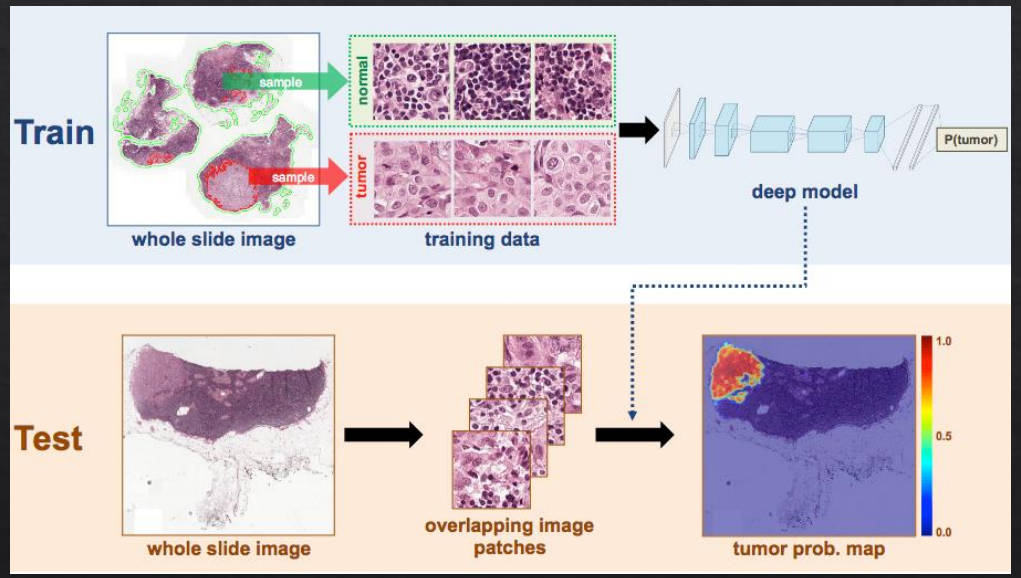
Presented by Hao Yuan

November 08, 2019

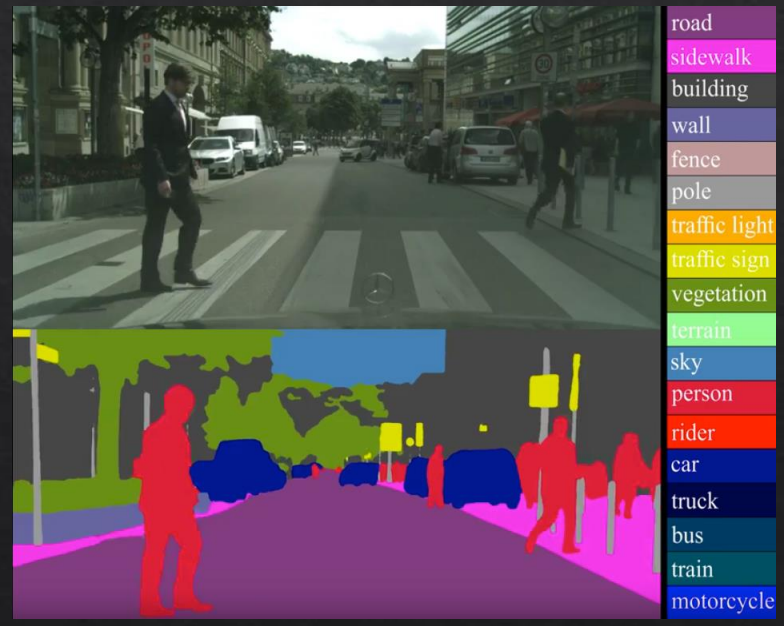
Deep Neural Networks are Everywhere

Playing Go

Making Medical Decision



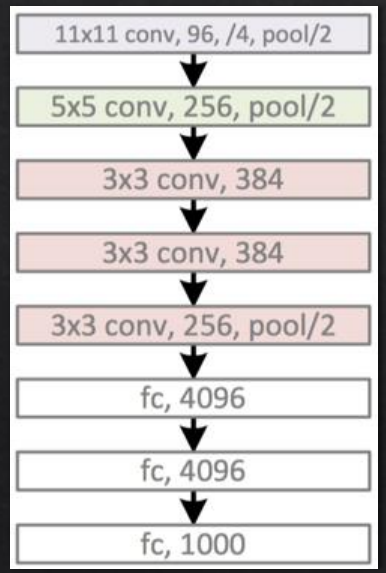
Understanding Scenes



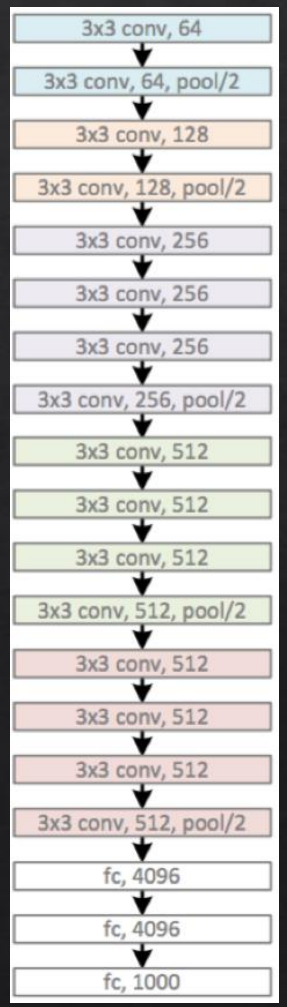
Reference: *https://interpretablevision.github.io/*

Deep Neural Networks for Visual Recognition

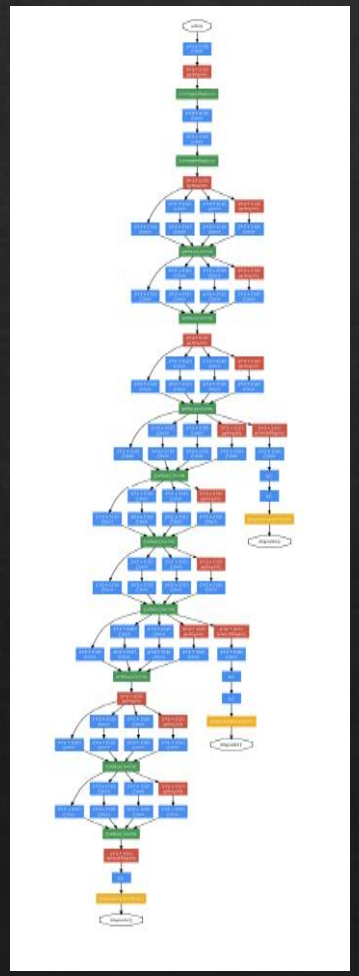
AlexNet



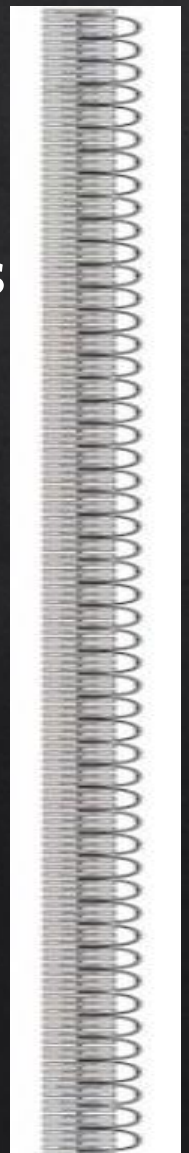
VGG



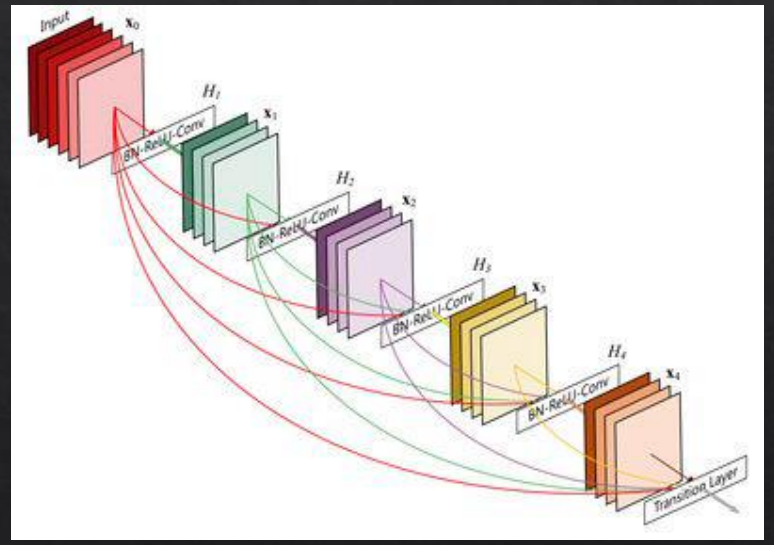
GoogLeNet



ResNet
>100 layers

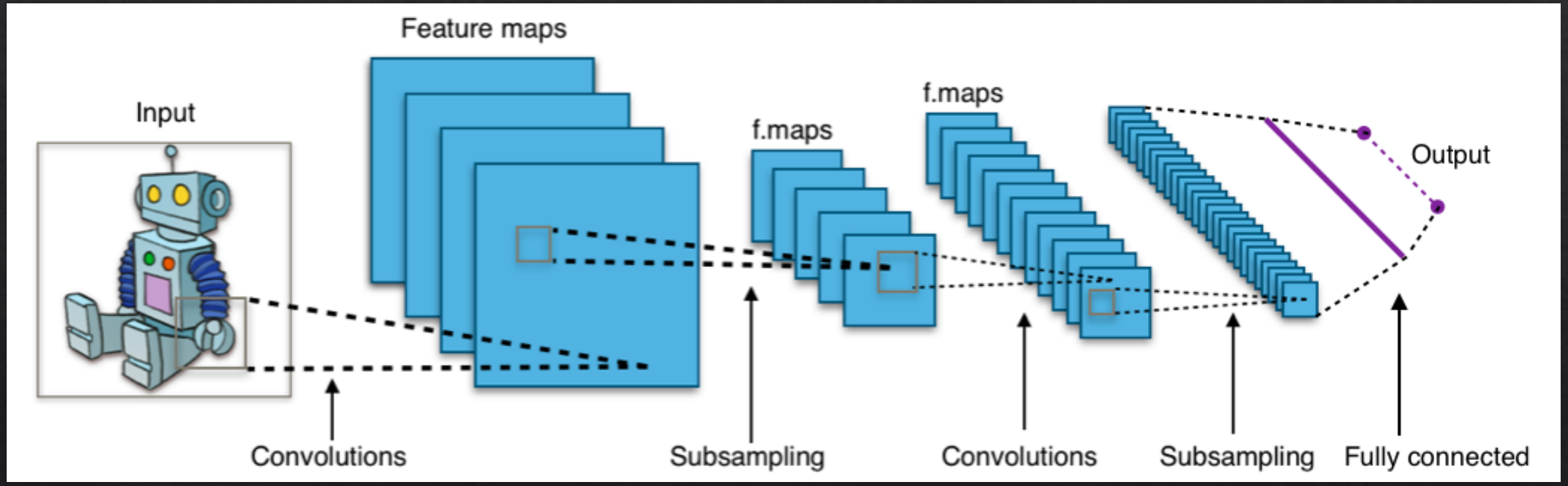


DenseNet >250 layers



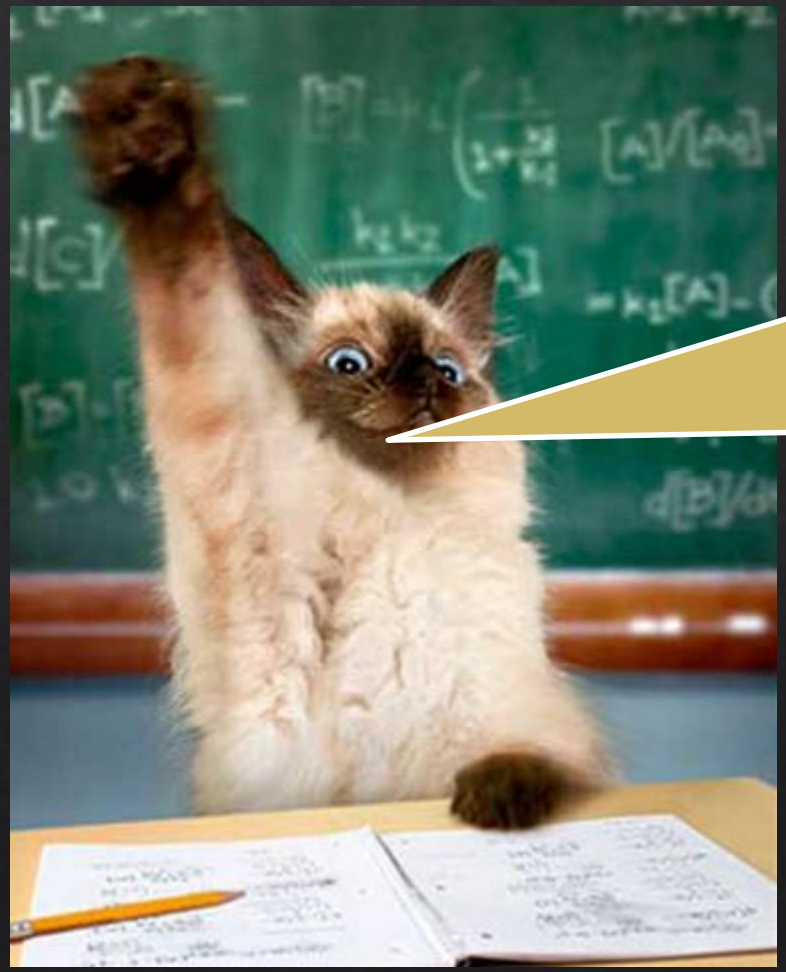
Reference: *https://interpretablevision.github.io/*

CNNs for Image Classification



Reference: *http://adventuresinmachinelearning.com/convolutional-neural-networks-tutorial-tensorflow/*

Is It Enough?

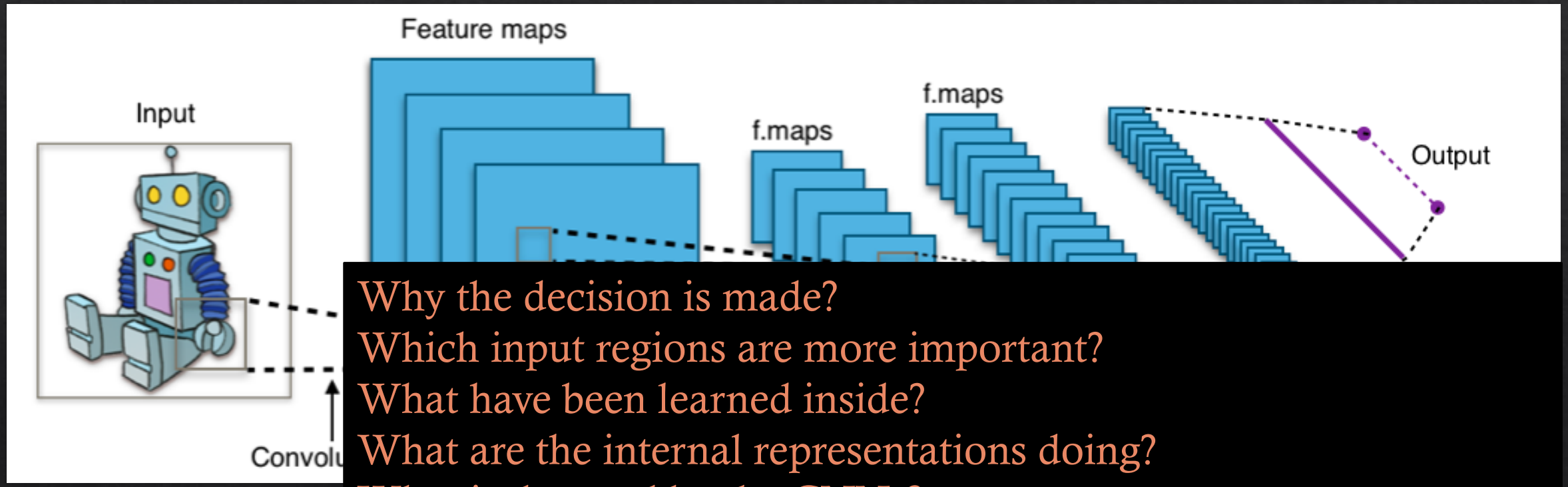


I heard you can just CNNs for image classification….

Can we go home now?

Reference: *http://www.ogroup.com.au/wp-content/uploads/2015/08/cat-raising-hand.jpg*

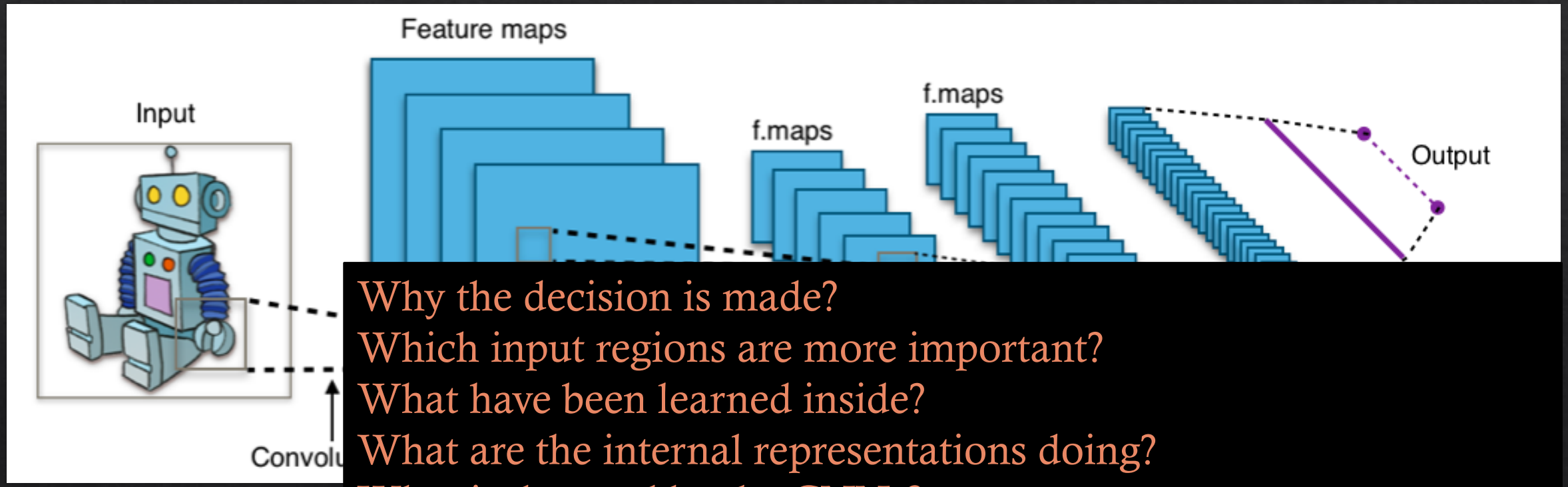
Questions?



Why the decision is made?
Which input regions are more important?
What have been learned inside?
What are the internal representations doing?
What is detected by the CNNs?

Reference: *http://adventuresinmachinelearning.com/convolutional-neural-networks-tutorial-tensorflow/*

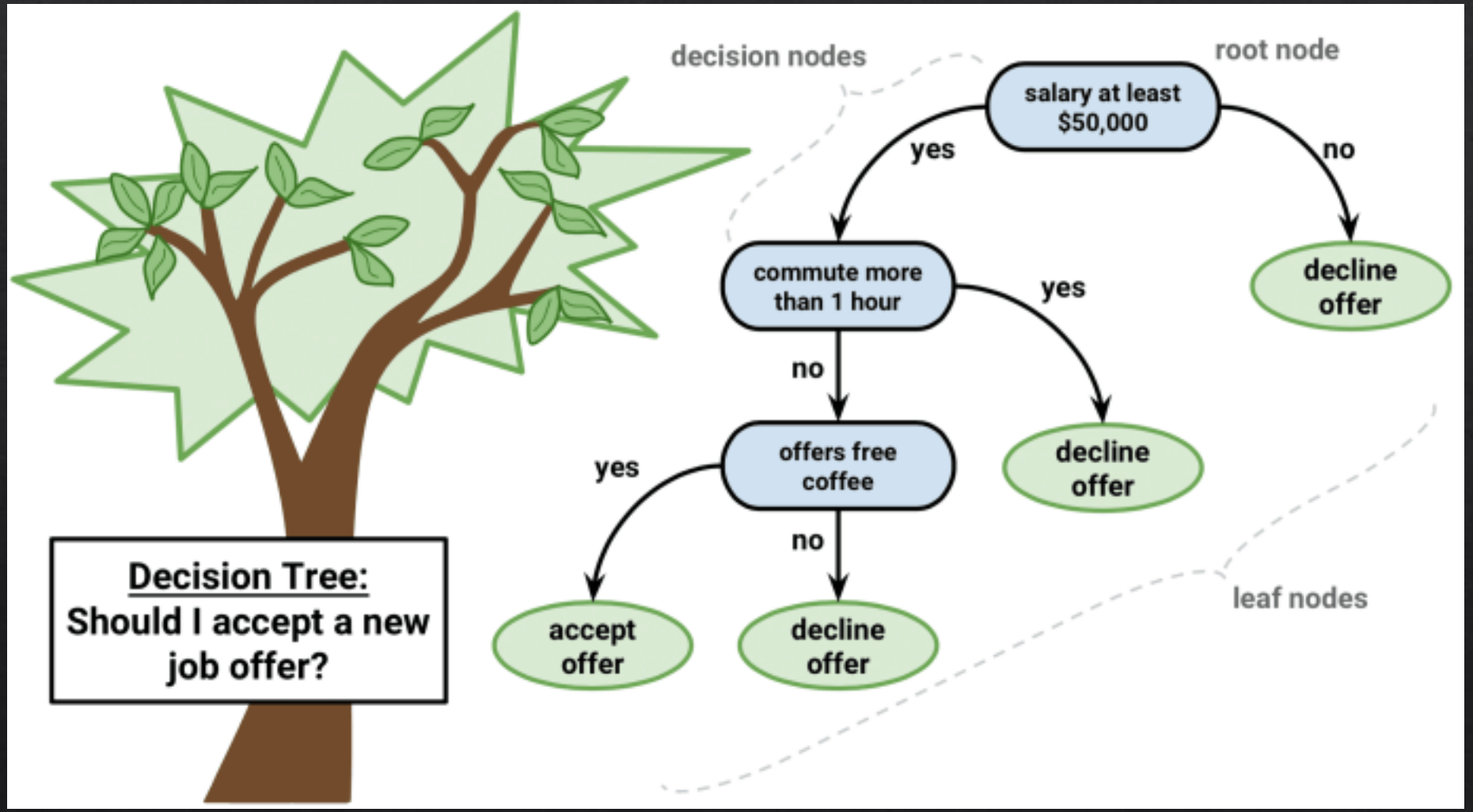
Questions?



Why the decision is made?
Which input regions are more important?
What have been learned inside?
What are the internal representations doing?
What is detected by the CNNs?
We need interpretation!

Reference: *http://adventuresinmachinelearning.com/convolutional-neural-networks-tutorial-tensorflow/*

What is Interpretation?



Offer1:
Company A, 100k, 10mins, no coffee, in Seattle, 20k stock, bonus.

Offer2:
Company B, 50k, 50mins, free coffee, in LA, no stock, bonus.

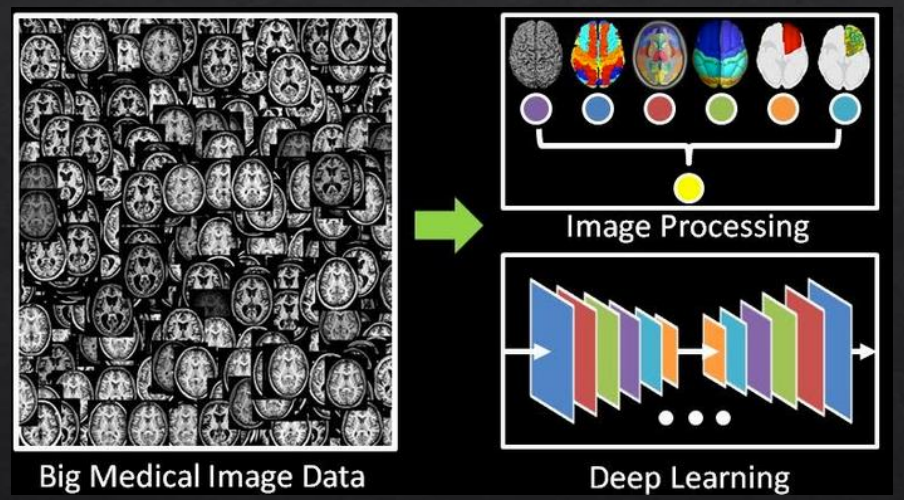
Reference: *https://towardsdatascience.com/decision-tree-hugging-b8851f853486*

Why Important?

Safety of AI models

Autonomous Driving

Trust of AI decision



Medical Diagnosis

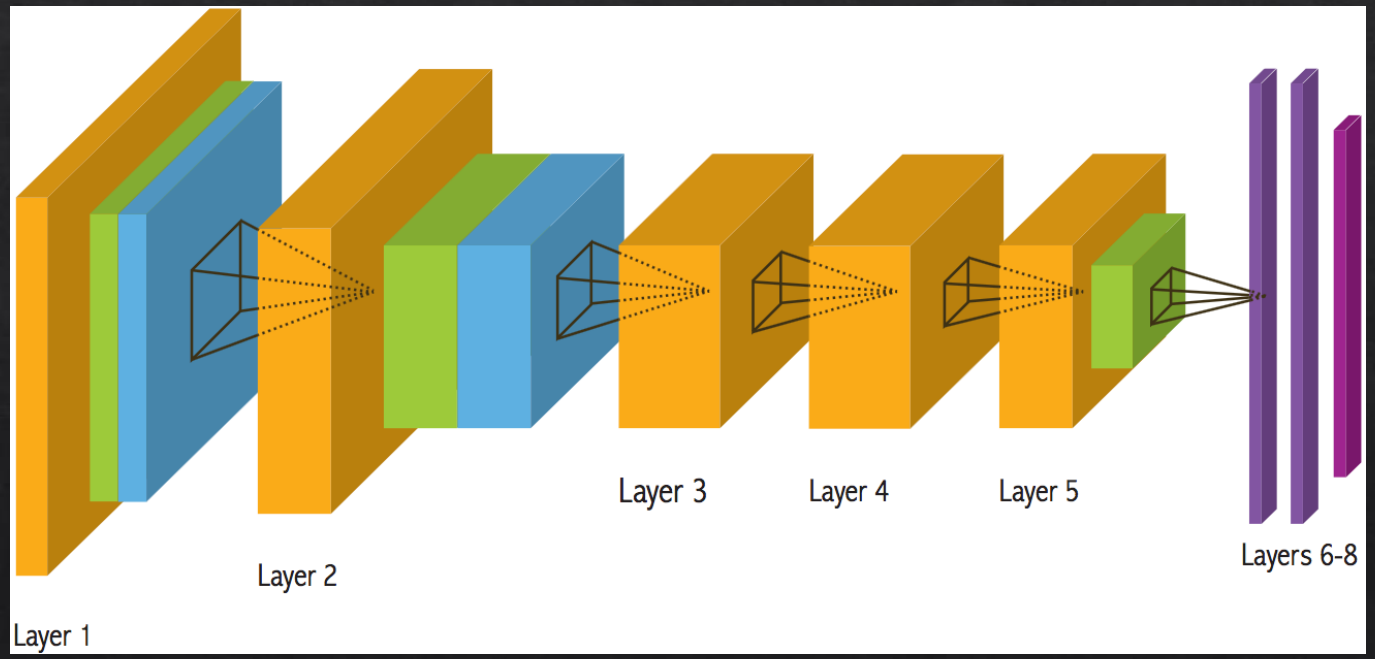
Policy and Regulation

Right to the explanation for algorithmic decisions

Reference: *https://interpretablevision.github.io/*

Understanding CNNs

- Take CNNs image classification models as the example.
- For image models, visual interpretation is the most straightforward way.



Cafeteria (0.9)

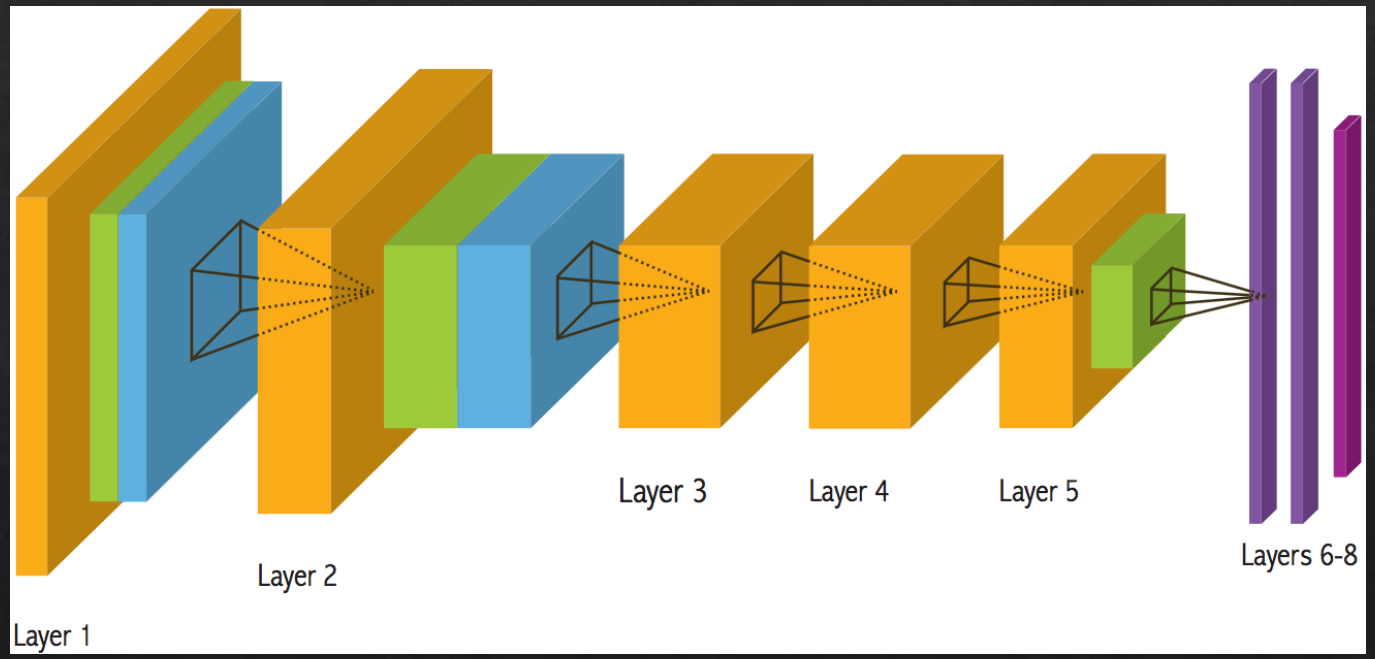
Reference: *https://interpretablevision.github.io/*

Understanding CNNs

- Input image: $H_{\text{in}} \times W_{in} \times 3$ matrix
- For a hidden layer: $H \times W \times C$ matrix
  - We call these values activations
  - Different channels detect different input patterns
  - The activation $a_{i,j,k}$ at location (i, j, k) represents how strength the pattern is detected
- Different channels (activations) are combined to make decision

Understanding CNNs

- Interpretation in two different spaces: input space and feature space.
- Two different types of interpretation: attribution and feature visualization.



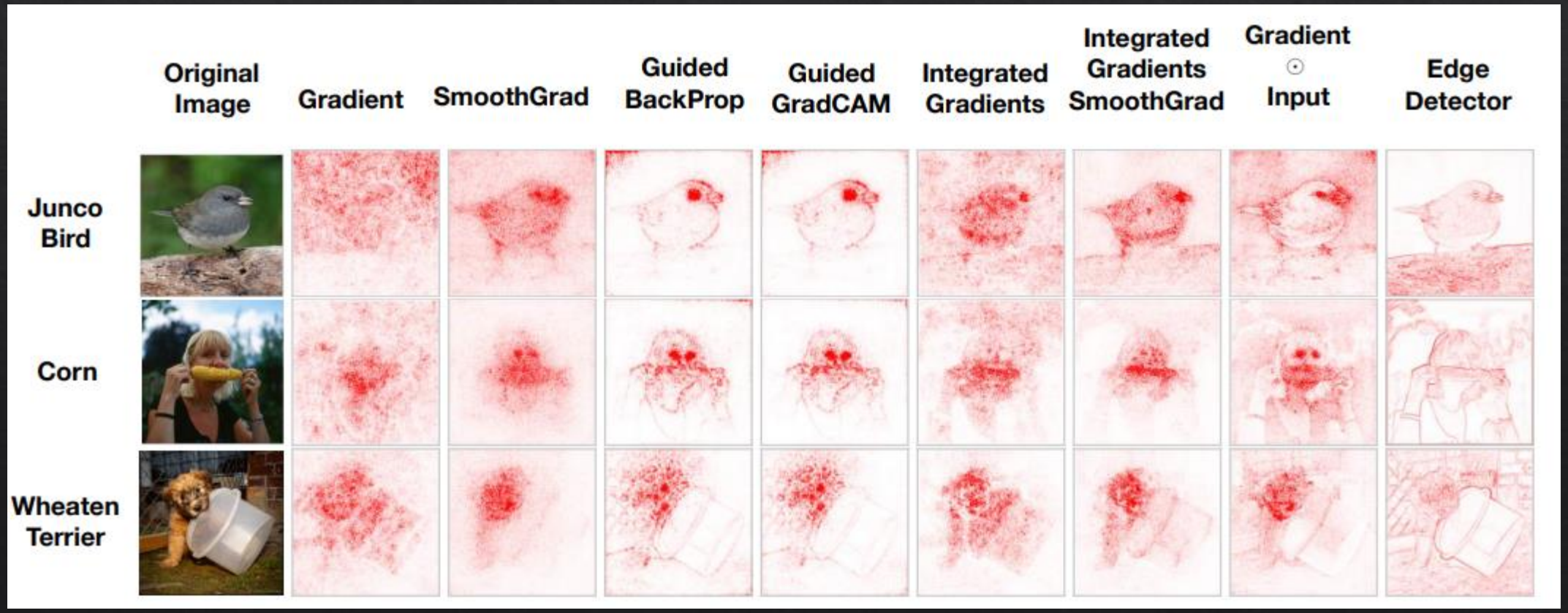
Cafeteria (0.9)

- Which input regions are important to make decision?

Reference: *https://interpretablevision.github.io/*

Saliency Maps (Attribution)

- Heatmaps showing the importance (contribution) of different input pixels.



Reference: *https://arxiv.org/pdf/1810.03292.pdf*

Gradient-based approach

- Given an image I_0, a class c, the class score before softmax is $S_c(I_0)$.
- Our target is to rank the pixels of I_0 based on their influence on the class score.
- First, for a linear model:

$$S_c(I) = w_c^T I + b$$

- I is represented in the vectorized (one-dimensional) form.

$$y = 5x_1 + 2x_2 + 6$$

Reference: *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*

Gradient-based approach

- Given an image I_0, a class c, the class score before softmax is $S_c(I_0)$.
- Our target is to rank the pixels of I_0 based on their influence on the class score.
- First, for a linear model:

$$S_c(I) = w_c^T I + b$$

- I is represented in the vectorized (one-dimensional) form.

$$y = 5x_1 + 2x_2 + 6$$

- For such case, the magnitude of elements of w defines the importance.
- How about the non-linear case in CNNs?
- Highly non-linear due to the non-linear functions.

Reference: *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*

Gradient-based approach

- First-order Taylor expansion as an approximation:

$$S_c(I) \approx w^T I + b$$

- where w is the derivative of S_c with respect to the image I that

$$w = \frac{\partial S_c}{\partial I}$$

- Use the gradient to measure the importance.
- The magnitude of the derivative indicates which pixels need to be changed the least to affect the class score the most.
- Any problem here?

Reference: *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*

Gradient-based approach

- Gradients only reflect the sensitivity of the class score when there is a small change in the corresponding spatial location.

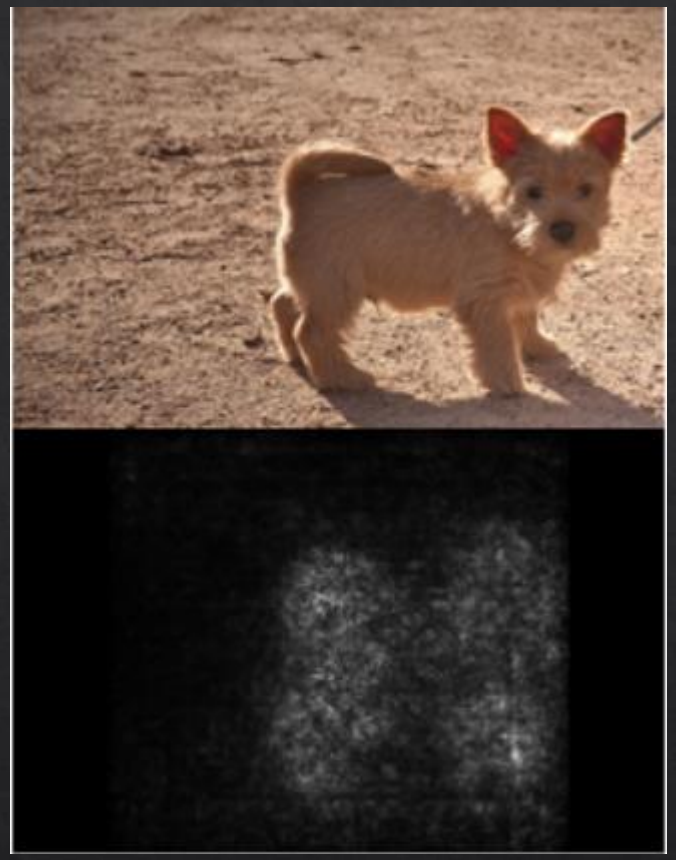
$$y = 1 \qquad if\ 5x_1 + 2x_2 + 6 > 0$$
$$y = 0 \qquad if\ 5x_1 + 2x_2 + 6 \leq 0$$

- Linear approximation.

$$Score_c\ (I)_{i,j} = w_{ij} \times x_{i,j}$$

- Measure how much one spatial location contributes to the final class score.

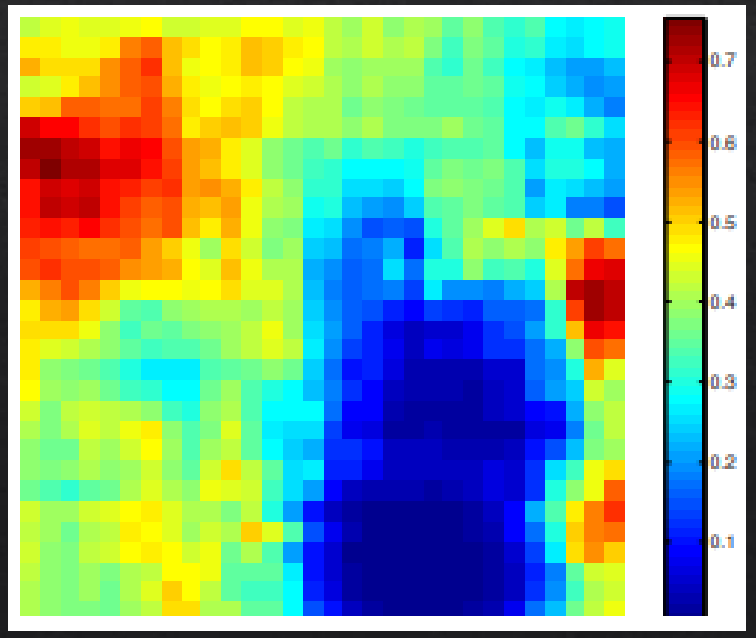
Reference: *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*



Reference: *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*

Occlusion Sensitivity

- Use a grey square to occlude different portions of the input image.
- Set the RGB values of the image portion to (128,128,128).
- Monitor the change of class score/ probability.

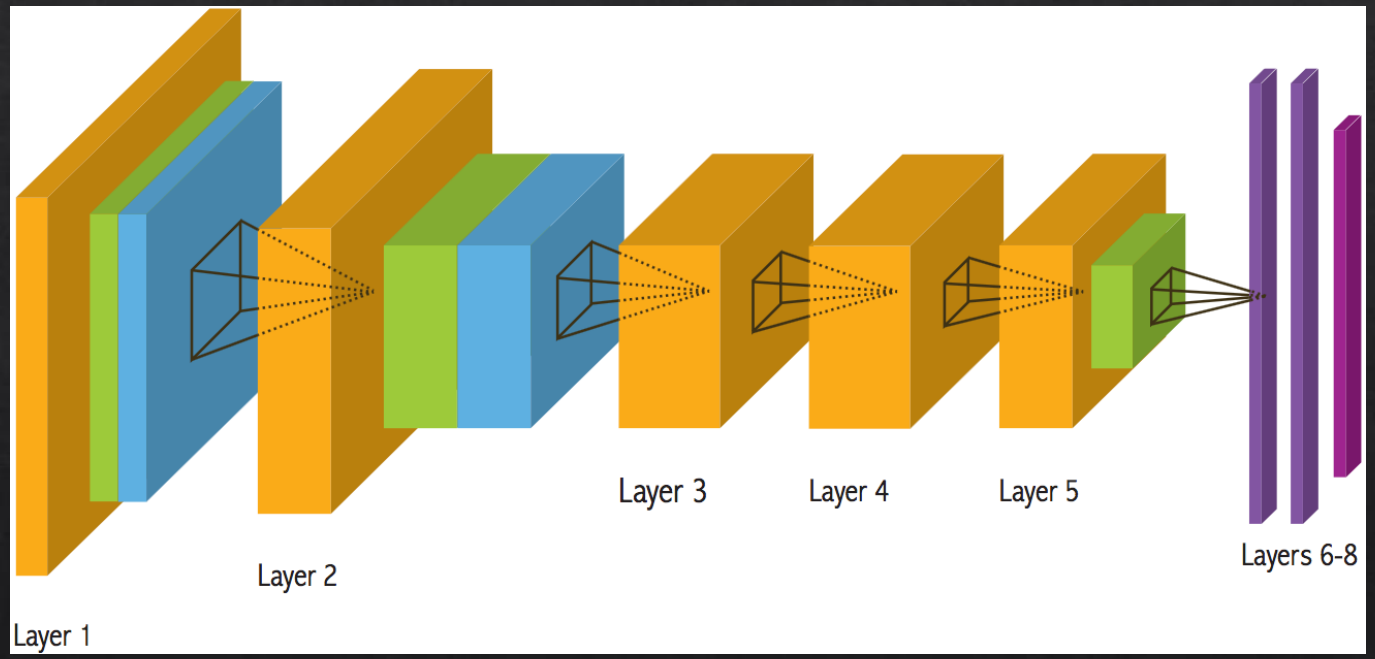


Probability of correct class

Reference: *Visualizing and Understanding Convolutional Networks, https://arxiv.org/pdf/1311.2901.pdf*

Understanding CNNs

- Interpretation in two different spaces: input space and feature space.
- Two different types of interpretation: attribution and feature visualization.



Cafeteria (0.9)

- What input pattern leads to a certain behavior in the hidden layers?

Reference: *https://interpretablevision.github.io/*

Feature Visualization

- Feature visualization answers questions about what a network — or parts of a network — are looking for by generating examples.
- For the location (i, j, k) in a hidden layer, the activation value is $a_{i,j,k}$.
- What input will activate such location?
- For the class c, what input will lead to a high class score?
- Three different types of approaches:
  - Searching
  - Optimization
  - Deconvolution

Reference: *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*

Feature Visualization via Searching

- Search in the dataset for images towards an objective
- The objective can be:
  - Maximize the activation $a_{i,j,k}$
  - Maximize the class score for class c
- Simple and straightforward.
- After training, the model and the parameters are fixed.
- Simply feed each image into the model, and check the objective.

Feature Visualization via Searching

- Dataset Examples show us what neurons respond to in practice

- It is problematic! Why?

Reference: *https://distill.pub/2017/feature-visualization/*

Feature Visualization via Searching

- A single image is not enough
- Multiple images, how many?
- How to find common patterns?

Reference: *https://distill.pub/2017/feature-visualization/*

Optimization Techniques

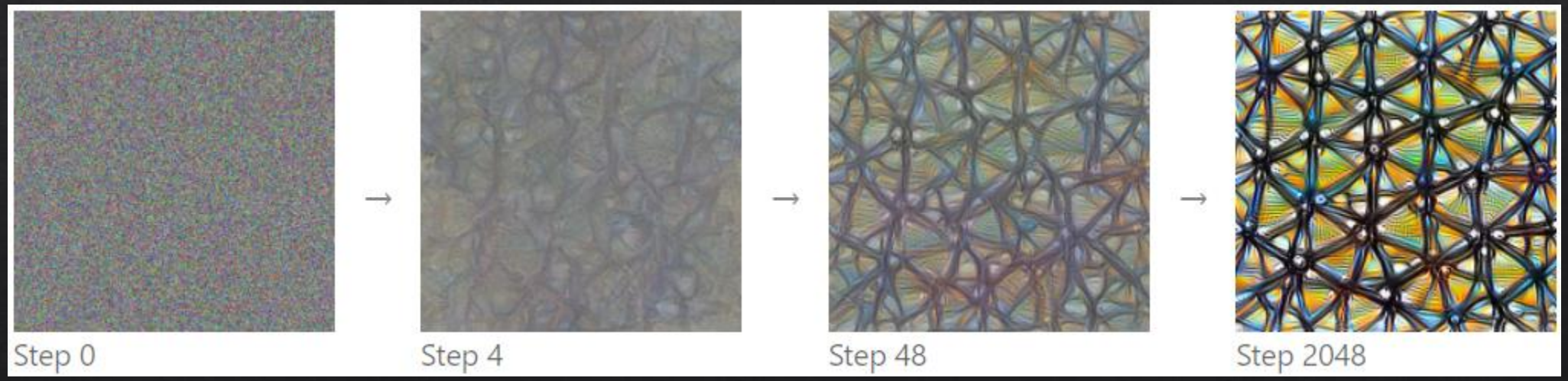
- Recall how do we train CNNs.
  - An input image
  - Feed to the network
  - Calculate the loss function/ objective function
  - Back propagation
  - Update parameters W
  - Repeat
- Such procedure is called as optimization.

Optimization Techniques

- For a hidden unit at location (i, j, k) in a hidden layer
- Find input patterns which maximize the activation $a_{i,j,k}$.
- Consider it as an optimization problem.
  - Randomly initialize an input X
  - Feed it to the trained network
  - Check the activation $a_{i,j,k}(X)$
  - Back propagation to update the input X to maximize the activation $a_{i,j,k}$
  - Repeat
  - $X^* = \arg max_X \ a_{i,j,k}(X)$
- A common stopping condition is to set a maximum number for updating step.

Optimization Techniques

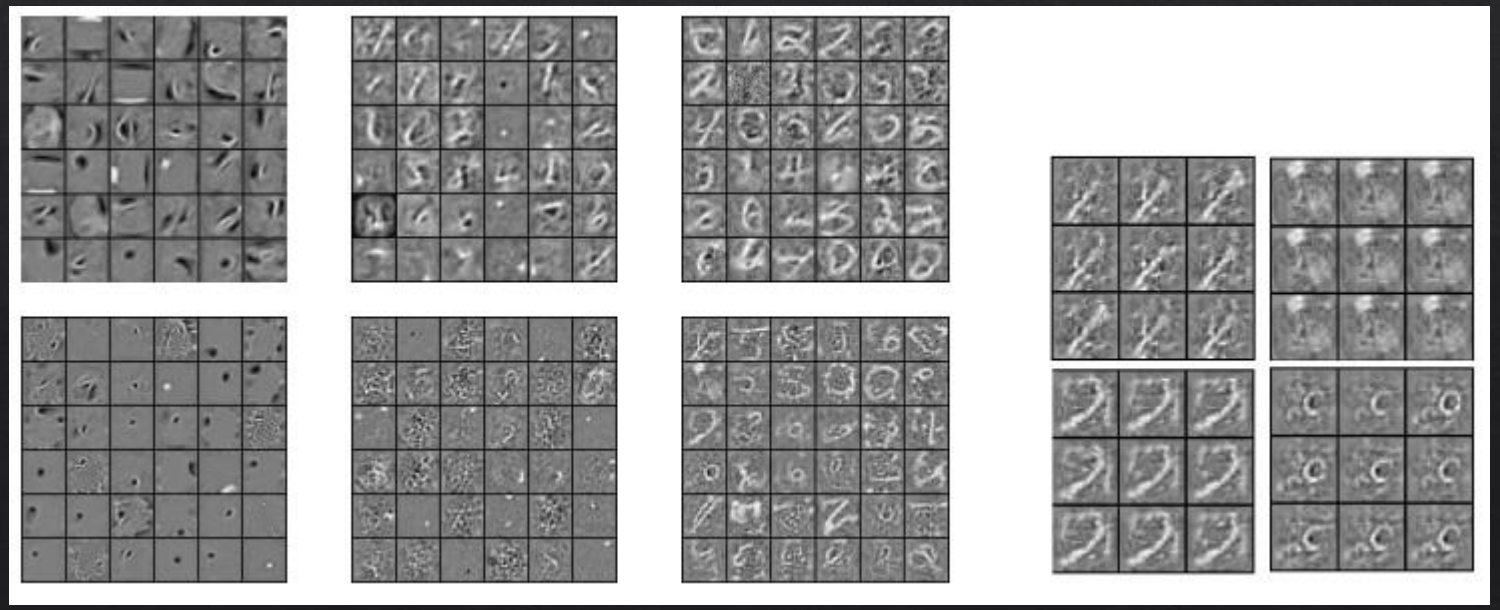
- The key difference:
- Network training: update the parameters.
- Visualization: fix the parameters and update the input.



Reference: *https://distill.pub/2017/feature-visualization/*

Optimization Techniques

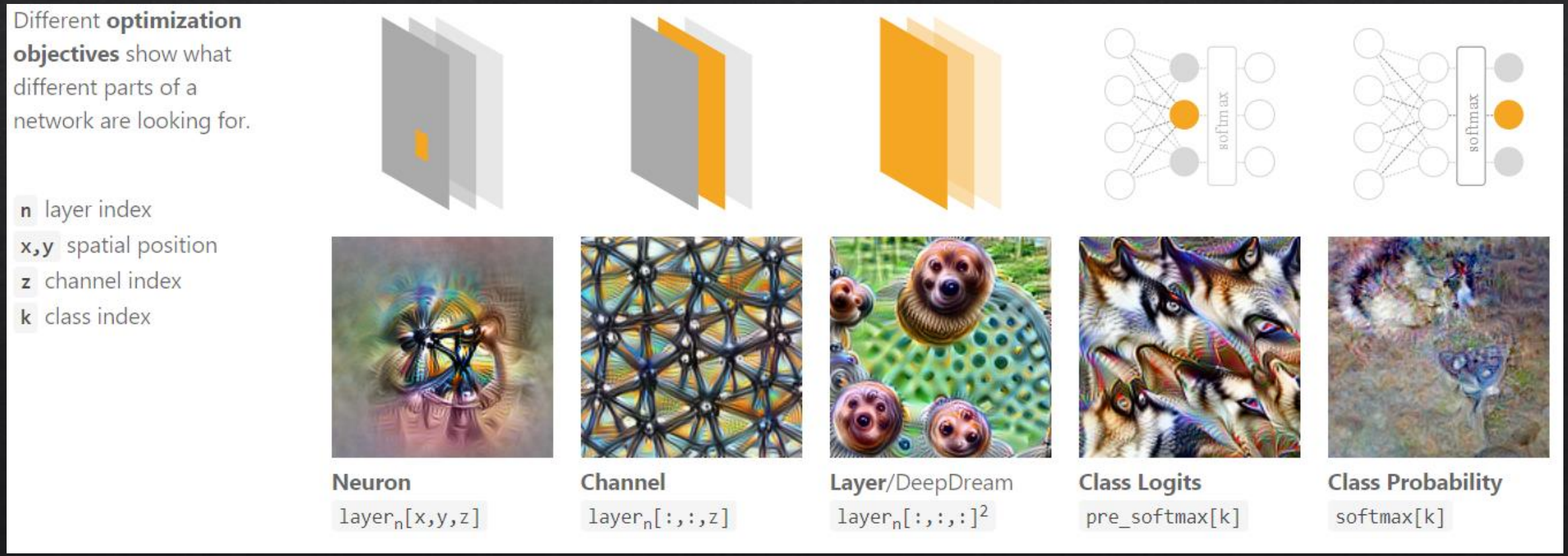
- Left: activation results for different hidden units (Lower layer to High layer)
- Right: activation results from 9 random initializations.
- Any observation?



Reference: *Visualizing higher-layer features of a deep network*

Different Objectives

- A wide variety of options:



Reference: *https://distill.pub/2017/feature-visualization/*

Optimization Techniques

- For a class c, we can maximize its class score $S_c(X)$.

$$X^* = \arg max_X \, S_c(X) - \alpha \|X\|_2^2$$

- What does the red part mean? Why need it?
- After optimization, what does the input reflect?

Optimization Techniques

Dataset Examples show us what neurons respond to in practice

Optimization isolates the causes of behavior from mere correlations. A neuron may not be detecting what you initially thought.

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

Reference: *https://distill.pub/2017/feature-visualization/*

CSCE 636: Deep Learning

The Visual Interpretation for Deep Learning

Presented by Hao Yuan

November 08, 2019

Gradient-based approach

- First-order Taylor expansion as an approximation:

$$S_c(I) \approx w^T I + b$$

- where w is the derivative of S_c with respect to the image I that

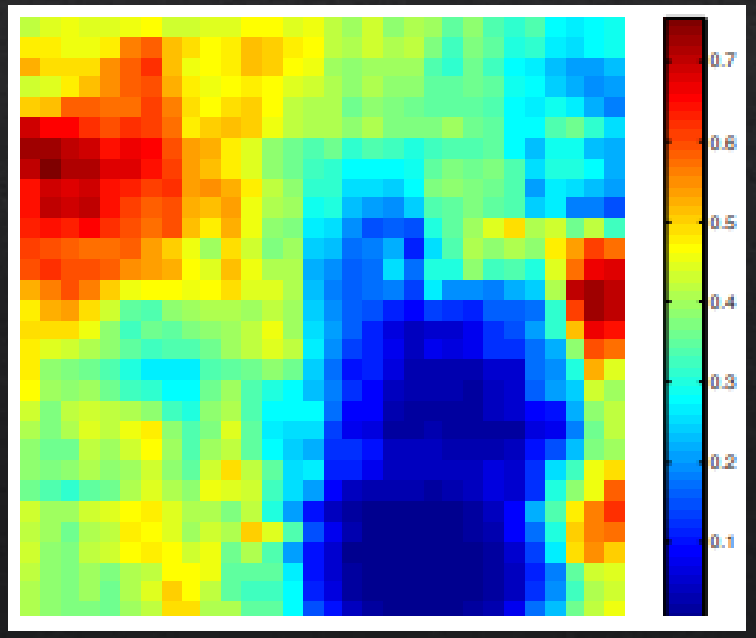
$$w = \frac{\partial S_c}{\partial I}$$

- Use the gradient to measure the importance.
- The magnitude of the derivative indicates which pixels need to be changed the least to affect the class score the most.
- Any problem here?

Reference: *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*

Occlusion Sensitivity

- Use a grey square to occlude different portions of the input image.
- Set the RGB values of the image portion to (128,128,128).
- Monitor the change of class score/ probability.

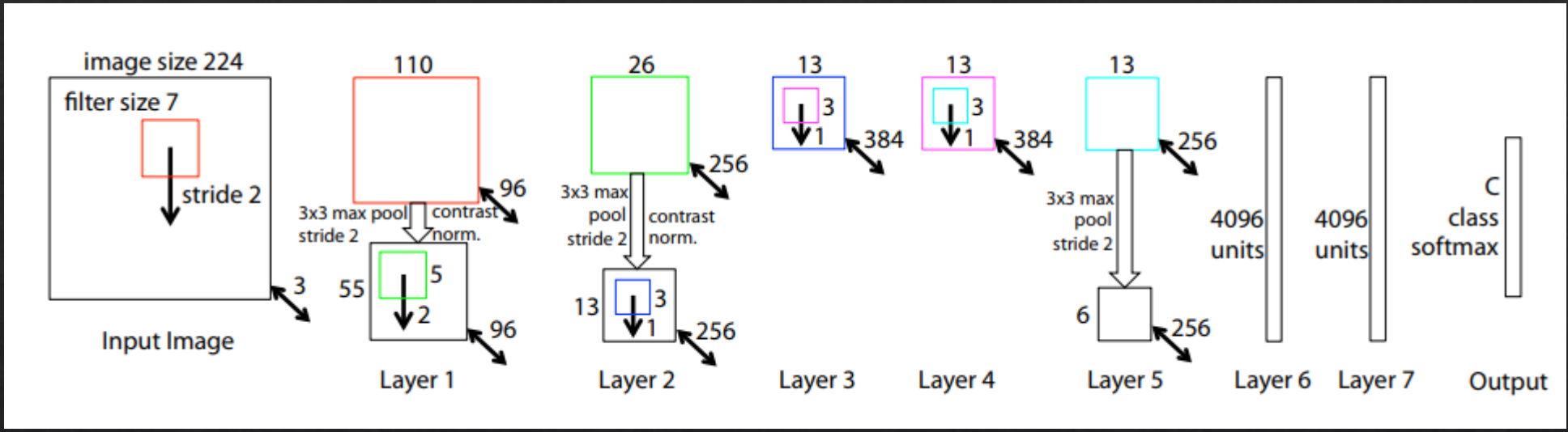


Probability of correct class

Reference: *Visualizing and Understanding Convolutional Networks, https://arxiv.org/pdf/1311.2901.pdf*

Feature Visualization via Deconvolutions

- Then can we project the hidden units back to the original input space?



- Operations: conv, max pooling, relu.

Reference: *Visualizing and Understanding Convolutional Networks, https://arxiv.org/pdf/1311.2901.pdf*

Feature Visualization via Deconvolutions

- As the reverse operations of the convolutional networks.
- Deconvolution: the reverse operation of convolution.
- During visualization, deconvolution networks are not trained (just a tool).
- The same number:
  - Conv vs deconv
  - Pooling vs unpooling
- Then how to visualize?

Reference: *Visualizing and Understanding Convolutional Networks, https://arxiv.org/pdf/1311.2901.pdf*

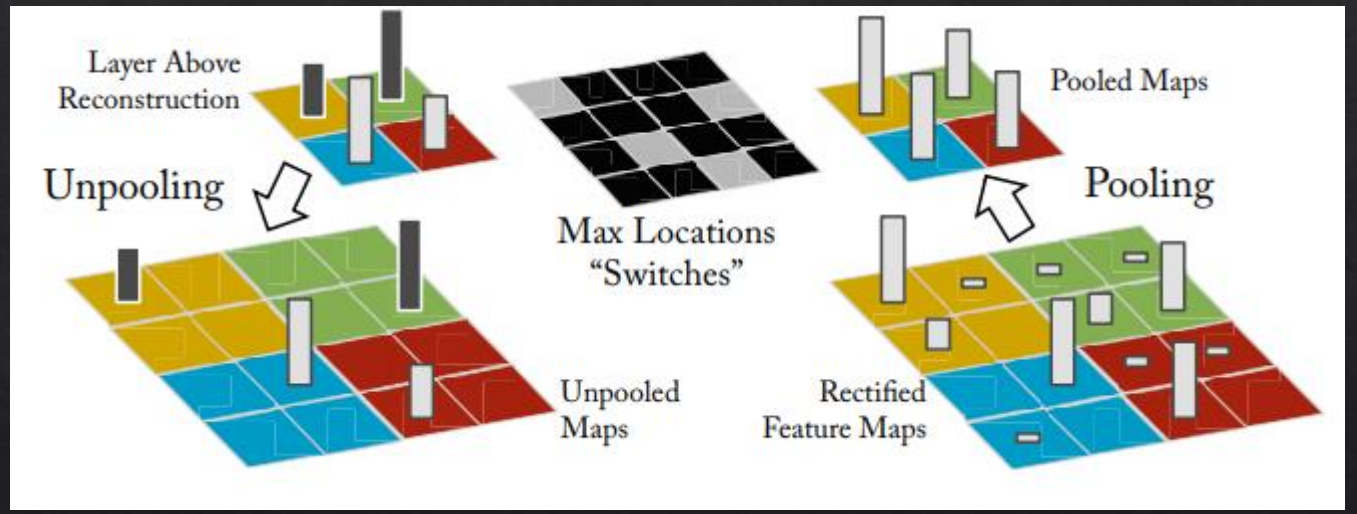
Feature Visualization via Deconvolutions

- Reverse max pooling operations:
  - Think about the back-propagation for max pooling.

Reference: *Visualizing and Understanding Convolutional Networks, https://arxiv.org/pdf/1311.2901.pdf*

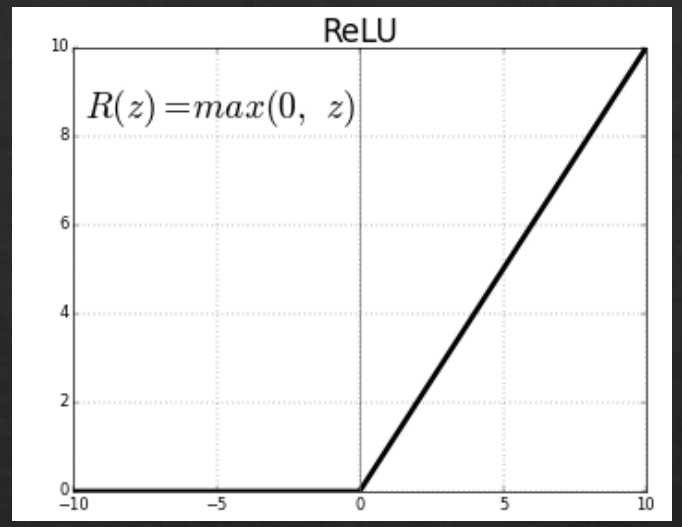
Feature Visualization via Deconvolutions

- Reverse max pooling operations:
  - Record the locations of the maxima. Looks familiar?



Reference: *Visualizing and Understanding Convolutional Networks, https://arxiv.org/pdf/1311.2901.pdf*

Feature Visualization via Deconvolutions



- Reverse the relu activation function.
- Feature maps are always positive.
- To obtain valid feature reconstructions at each layer,

pass reconstructed signal through relu as well.

- Recall the back-propagation for relu, any difference?

Forward pass $X_{i+1} = relu(X_i)$, when calculate the gradient, the sign indicator is based on X_i that we only consider the locations that $X_i > 0$.

For reconstructions, $R_n = relu(R_{i+1})$, the sign indicator is based on i+1 layer.

- Reverse the conv operations.
  - Simply employ deconv layers with the flipped kernel of conv layers.
  - The same as back-propagation.

Reference: *Visualizing and Understanding Convolutional Networks, https://arxiv.org/pdf/1311.2901.pdf*

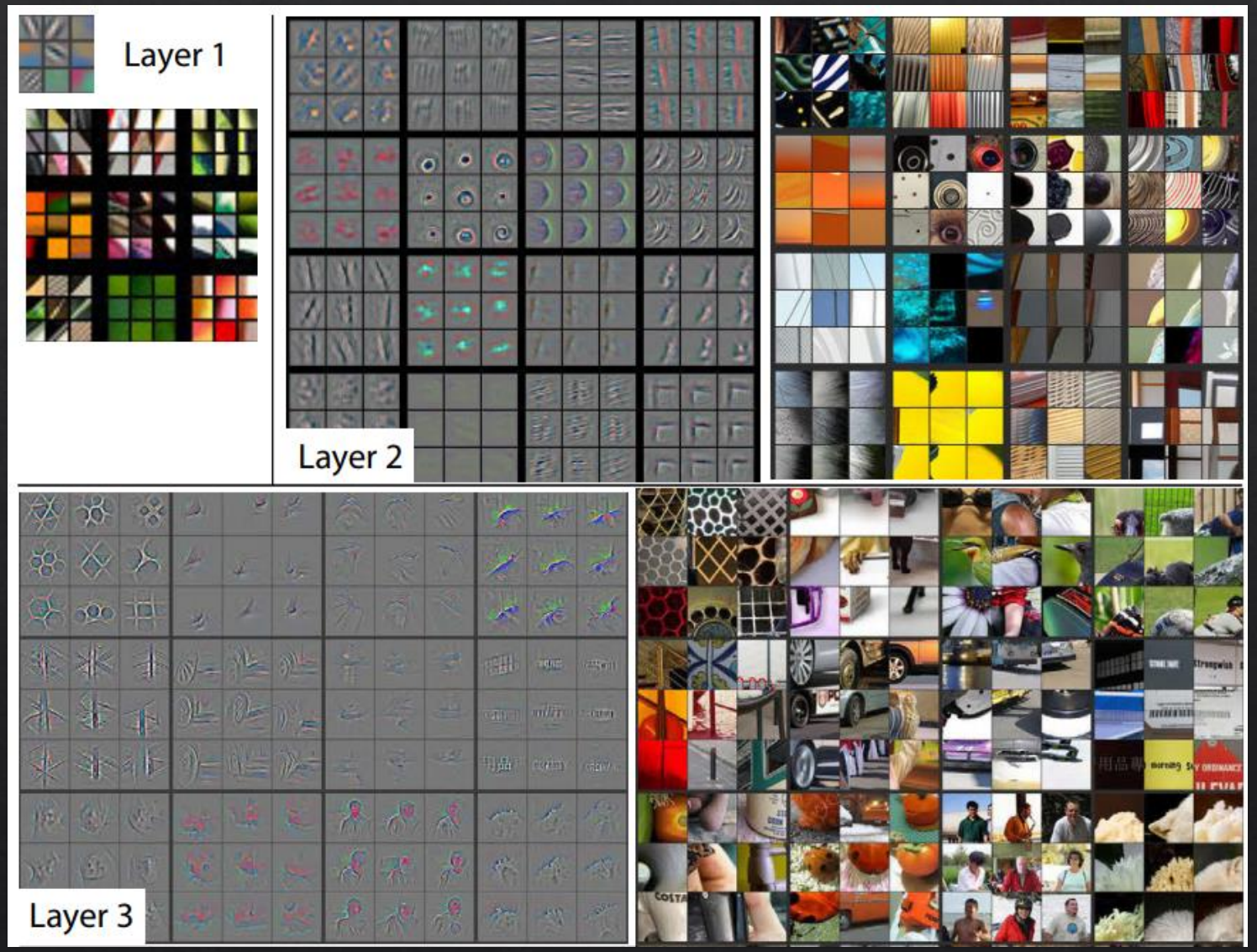
Feature Visualization via Deconvolutions

- For the activation of neuron i in layer j
- Keep the value of $a_{i,j}$ and set all other activations to 0.
- Feed the whole layer to the deconvolution networks to reverse all operations.

- Compare with optimization technique, which one is better?

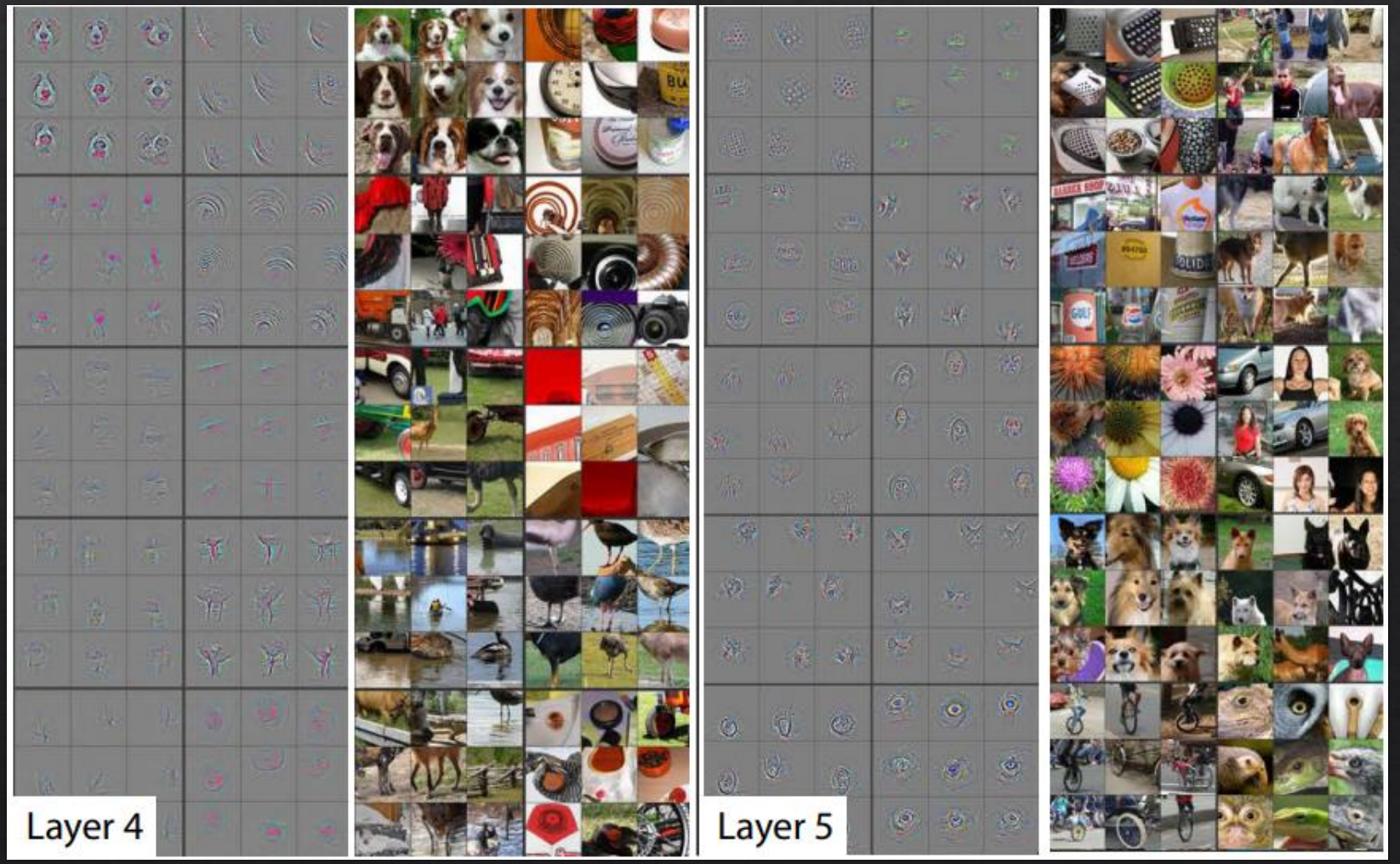
Reference: *Visualizing and Understanding Convolutional Networks, https://arxiv.org/pdf/1311.2901.pdf*

Visualization Results



Reference: *Visualizing and Understanding Convolutional Networks, https://arxiv.org/pdf/1311.2901.pdf*

Visualization Results



Reference: *Visualizing and Understanding Convolutional Networks, https://arxiv.org/pdf/1311.2901.pdf*

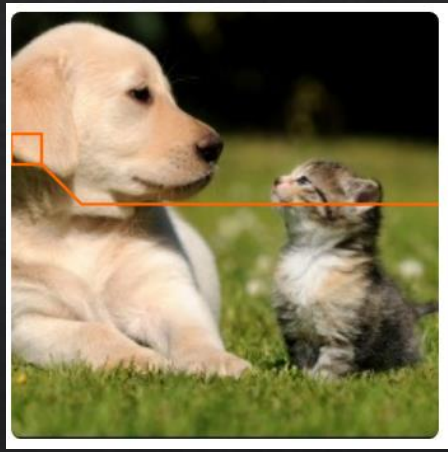
Understanding CNNs

- Questions have been answered:
- What input pattern leads to a certain behavior in the hidden layers?
- Which input regions are more important?
- More questions:
- What is the meaning of hidden layers?
  - What does the spatial location detect from the input?

Reference: *https://interpretablevision.github.io/*

The Meaning of Hidden Locations

- The optimization technique answers what does the network try to detect.
- For a spatial location (x,y), there is an activation vector.

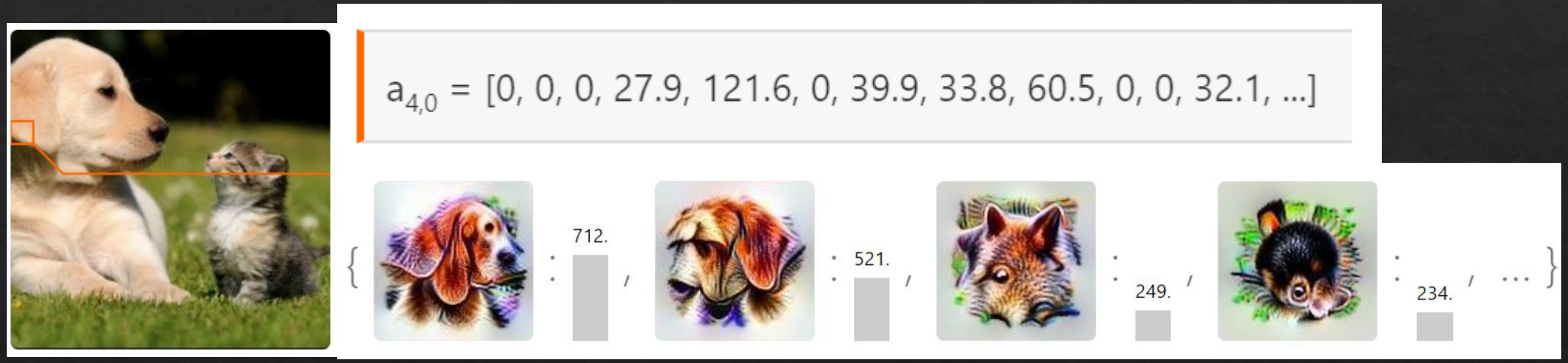


$a_{4,0}$ = [0, 0, 0, 27.9, 121.6, 0, 39.9, 33.8, 60.5, 0, 0, 32.1, ...]

Reference: *https://distill.pub/2018/building-blocks/*

The Meaning of Hidden Locations

- We can obtain an optimized input image for each element in this activation vector.
- What's the meaning?



- Looks like there is detector for floppy ears.

Reference: *https://distill.pub/2018/building-blocks/*

The Meaning of Hidden Locations

Making sense of these activations is hard because we usually work with them as abstract vectors:

> $a_{2,4}$ = [0, 0, 0, 0, 31.4, 0, 0, 0, 49.0, 0, 0, 0, ...]

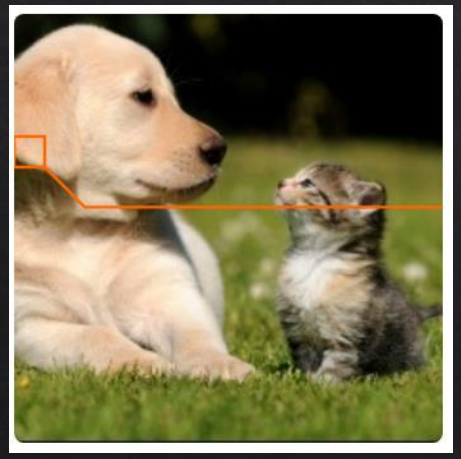
With feature visualization, however, we can transform this abstract vector into a more meaningful "semantic dictionary".

There seem to be detectors for floppy ears, dog snouts, cat heads, furry legs, and grass.

Reference: *https://distill.pub/2018/building-blocks/*

The Meaning of Hidden Locations

- Can we combine these images to represent the meaning of this location?
- Each neuron tries to detect a certain pattern
- The location: a weighted sum of these patterns
- Weights: the activation values



- Problem: time and memory consuming.

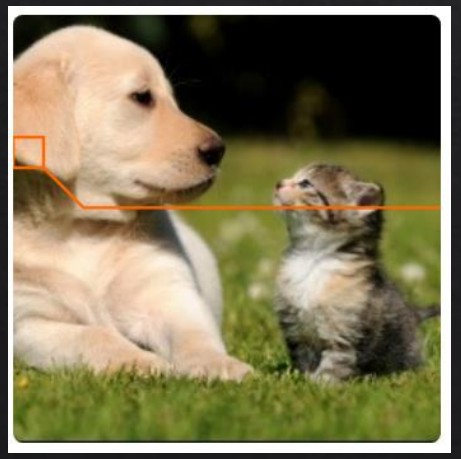
Reference: *https://distill.pub/2018/building-blocks/*

The Meaning of Hidden Locations

- Optimize the input for the location.
- Given an input X, for a certain location, the activation vector is v_0
- Then with a random initialized input, the activation vector is v
- We iteratively optimize the input image, that:

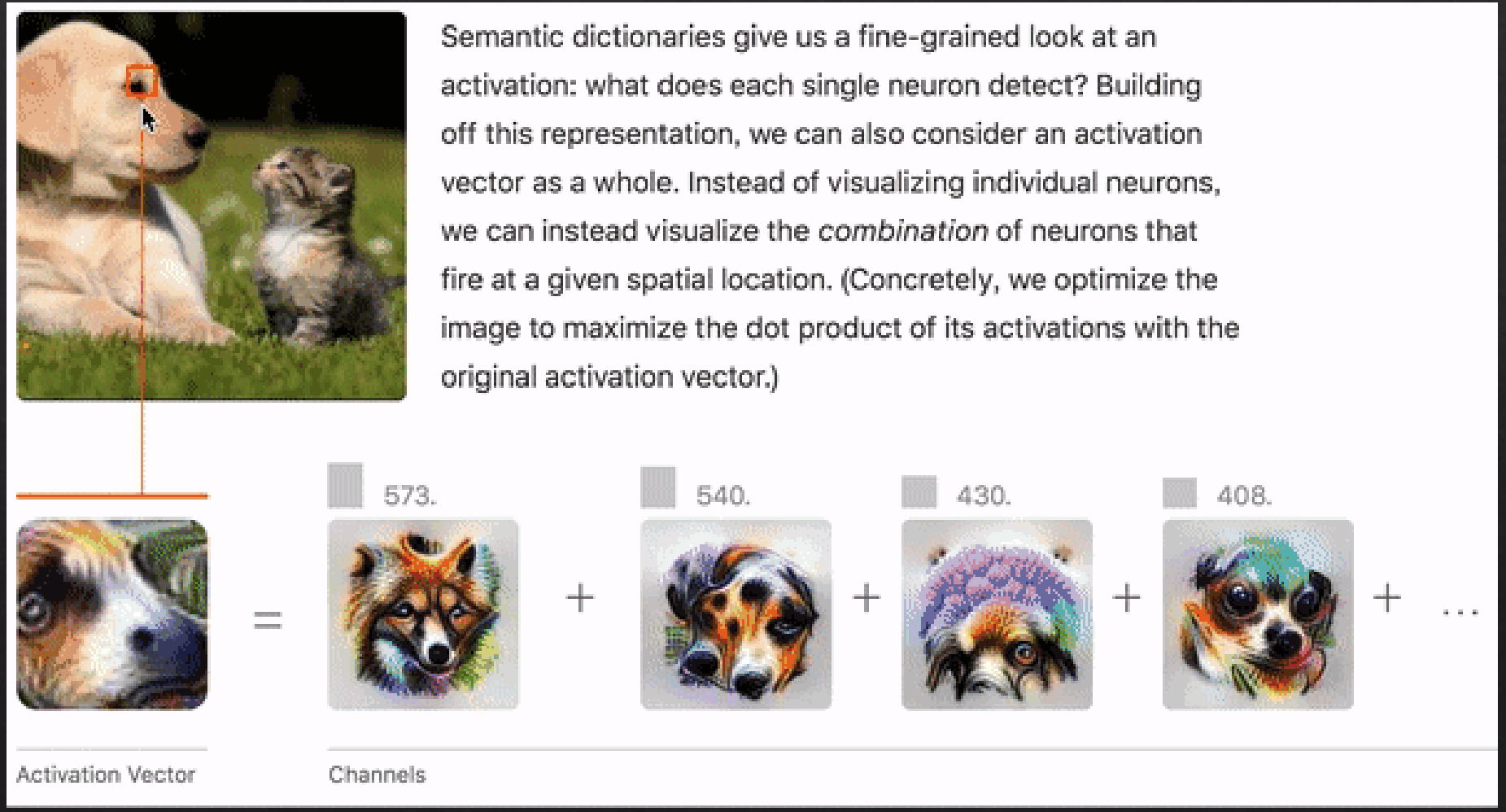
$$X^* = \arg max_X \, v \cdot v_0$$

- Update the input to mimic the activation vector for a certain location.



Reference: *https://distill.pub/2018/building-blocks/*

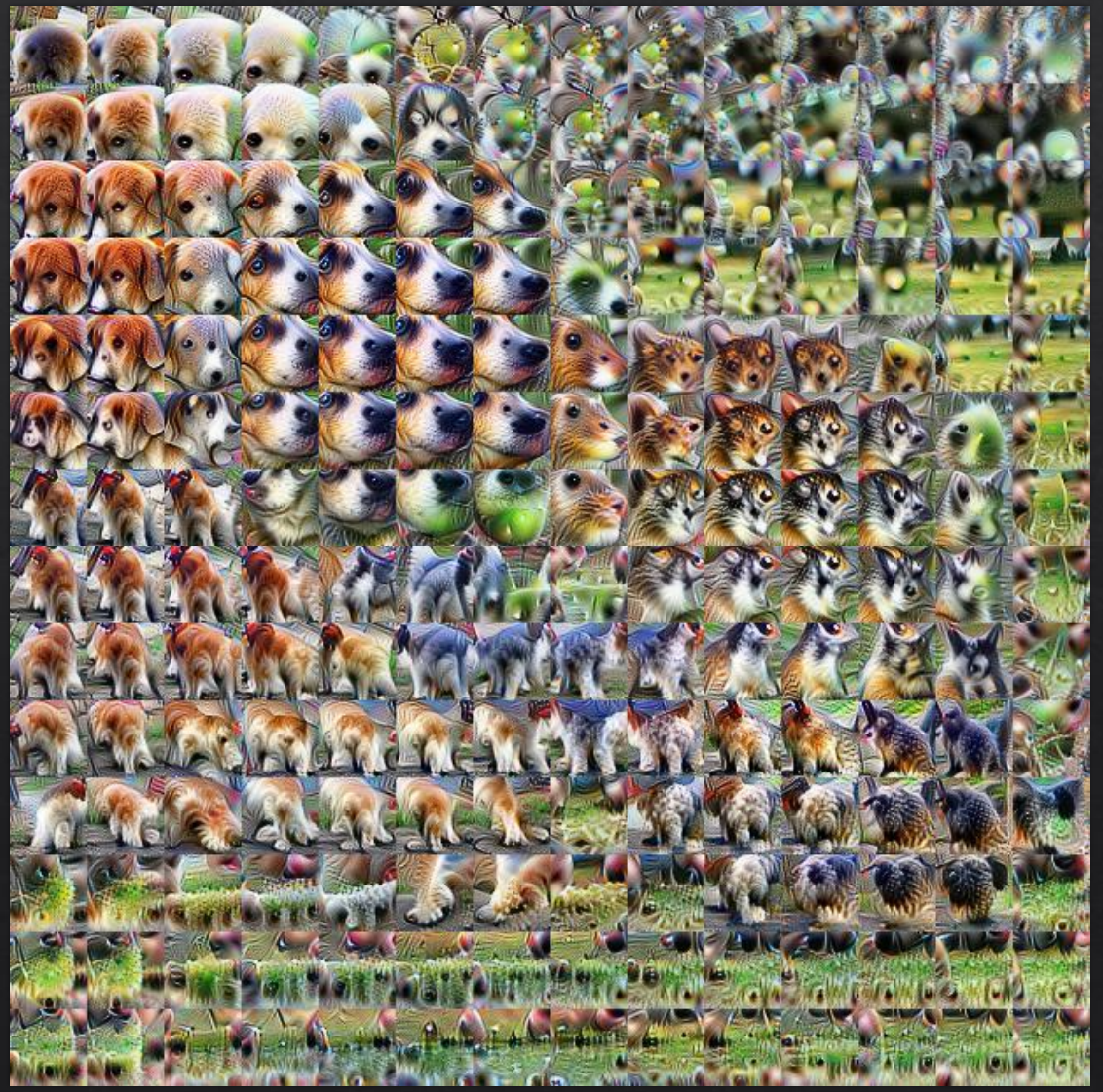
The Meaning of Hidden Locations



Reference: *https://distill.pub/2018/building-blocks/*

The Meaning of Hidden Locations

- What the whole layer detect.



Reference: *https://distill.pub/2018/building-blocks/*

The Meaning of Hidden Locations

Reference: *https://distill.pub/2018/building-blocks/*

The Meaning of Hidden Locations

- We can combine attribution with feature visualization.
  - To visualize what is detected and if it is important at the same time.
  - See the online demo: https://distill.pub/2018/building-blocks/
- Related papers:
  - *Visualizing and understanding convolutional networks*
  - *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*
  - *Visualizing higher-layer features of a deep network*