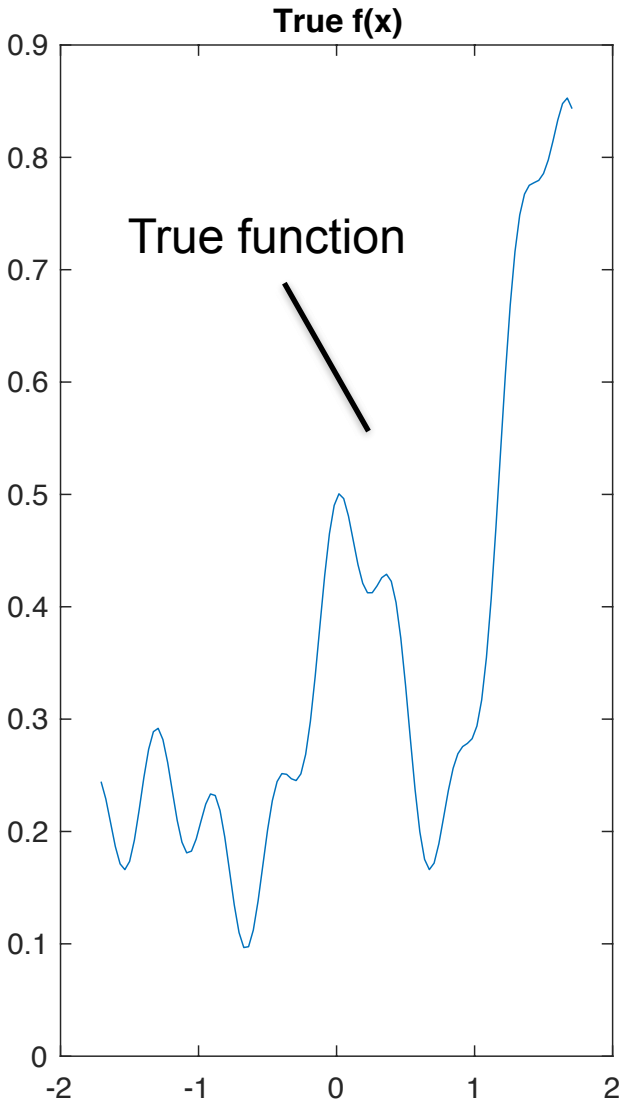
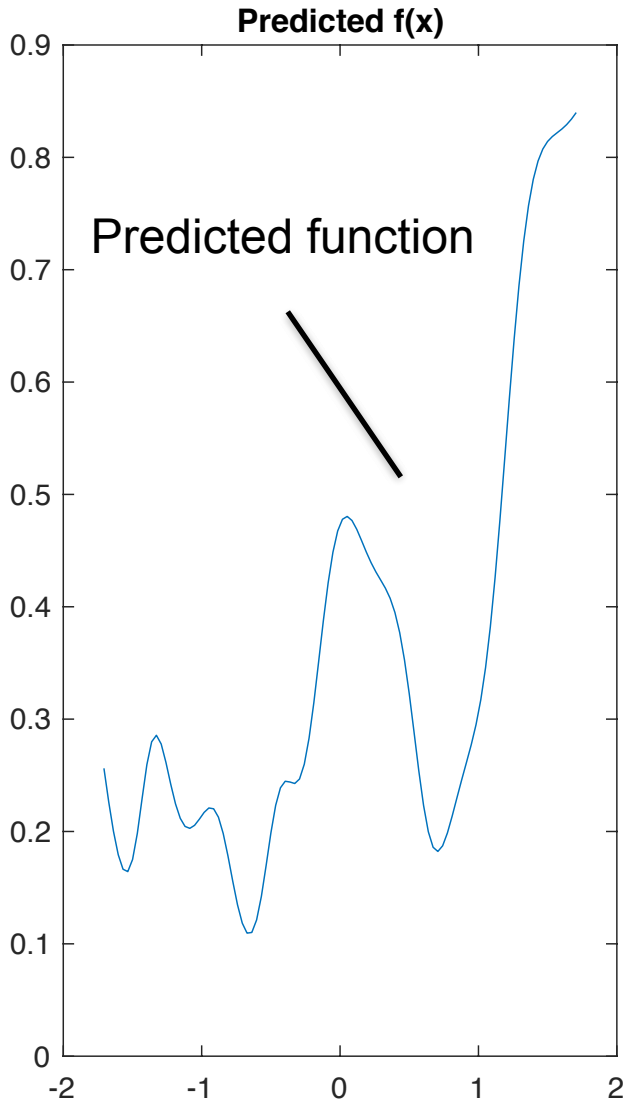
Visualizing and Understanding CNNs

CIS 680

Neural Net Learning





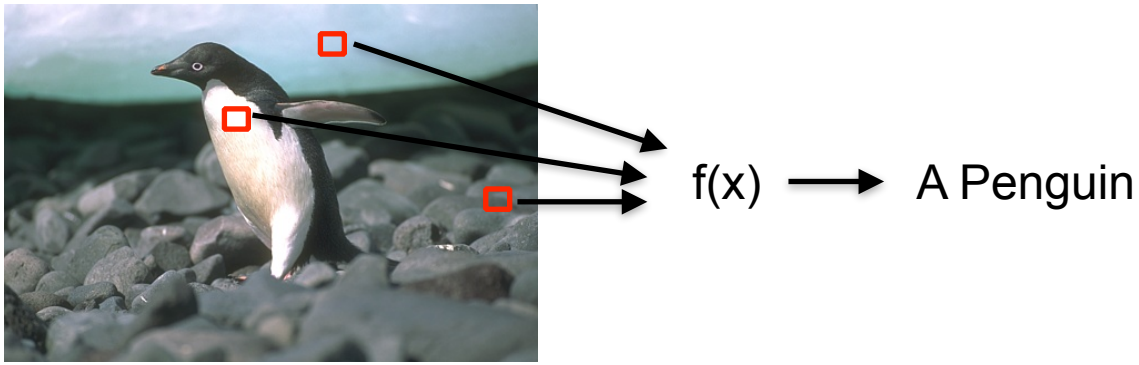
With the right design and enough data neural networks can learn any continuous function

Neural Net Learning

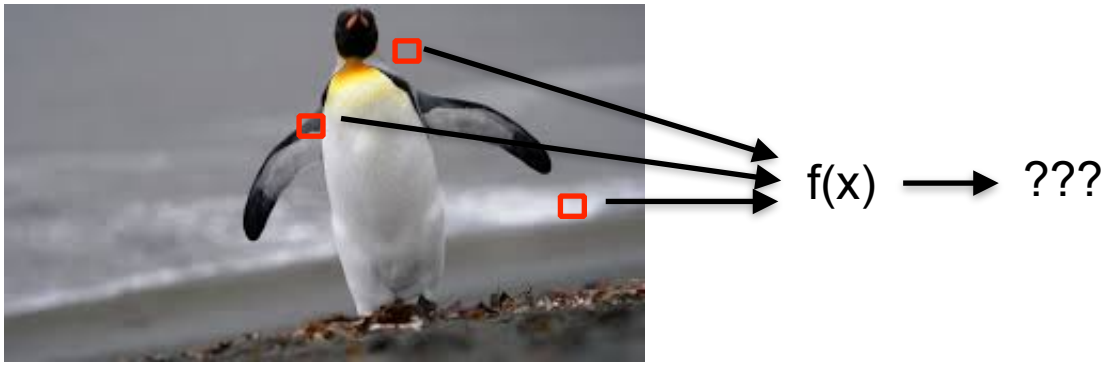
Generalization:

- We want the function that the neural network has learned to generalize to new data.

Training Data

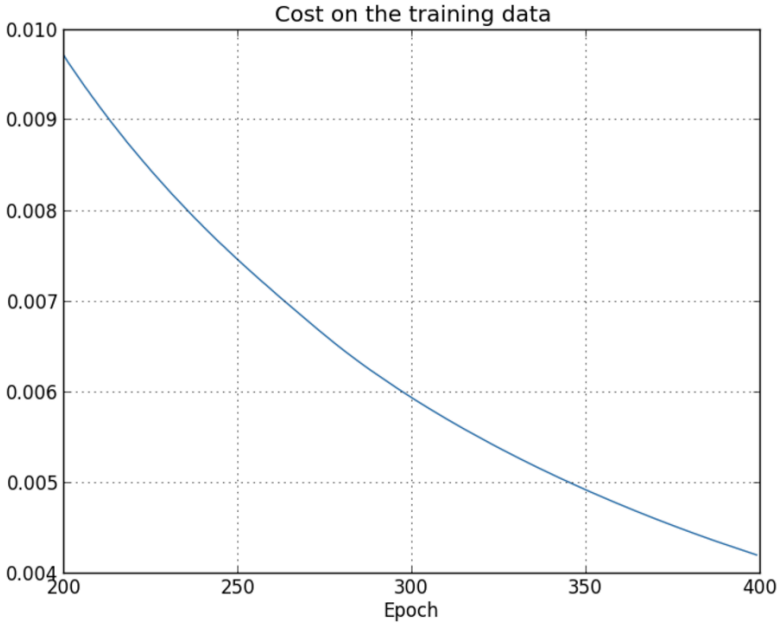


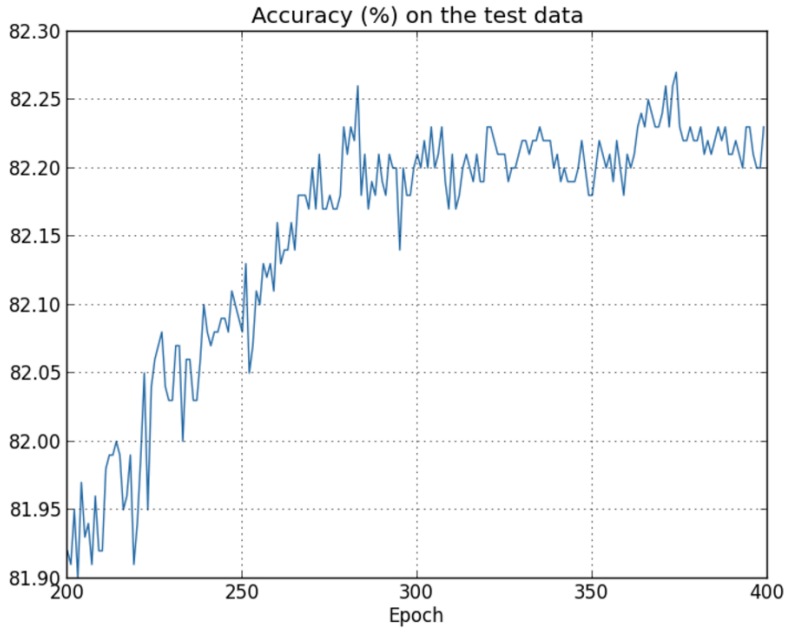
Testing Data



Overfitting

- Overfitting refers to the memorization of the training data, instead of learning the most sensible pattern in the data.





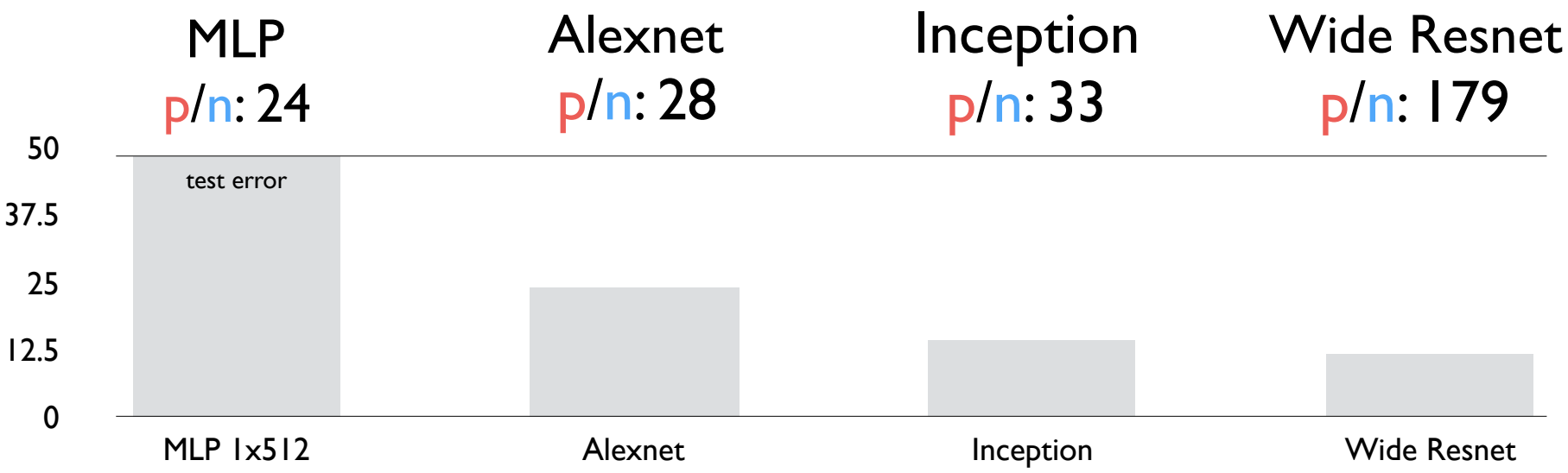
- **Loss on the training data keeps decreasing but the accuracy on the testing data remains the same or gets even worse.**
- **The network is learning intricacies of the training data that may not generalize on the testing data.**

Overfitting

What is the root cause of overfitting?

- The number of parameters in the neural network is typically significantly larger than the number of training data points.

Parameter Count
Num Training Samples



Overfitting

How to prevent overfitting?:

- One of the most simple solutions is called early stopping, which refers to stopping the CNN training earlier than expected.

Overfitting

How to prevent overfitting?:

- However, more principled approach is to use regularization when training the network.

- Regularization constrains the model from becoming overly complex (e.g. learning small weights).

- In other words, helps to keep the model simple.

- Can be implemented by adding an extra term to the loss

$$L = -(y \log f(x) + (1-y) \log (1 - f(x))) + \lambda \sum_{w} |w|_2^2$$

Forces the network weights to be small

Regularization

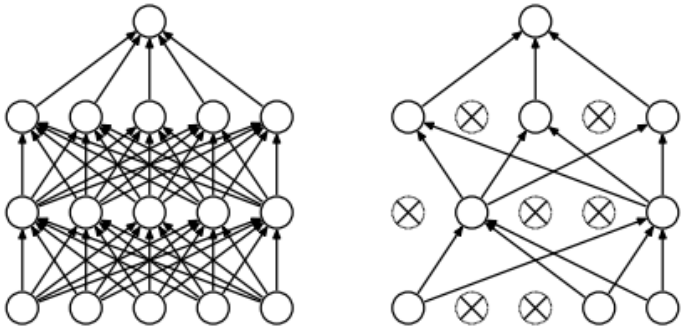
Data Augmentation:

- Data augmentation (e.g. artificially expanding the training dataset).
- In the context of computer vision, these include cropping, image mirroring, etc.

Regularization

Dropout:

- During training, we randomly set a certain number of neurons to zero values.



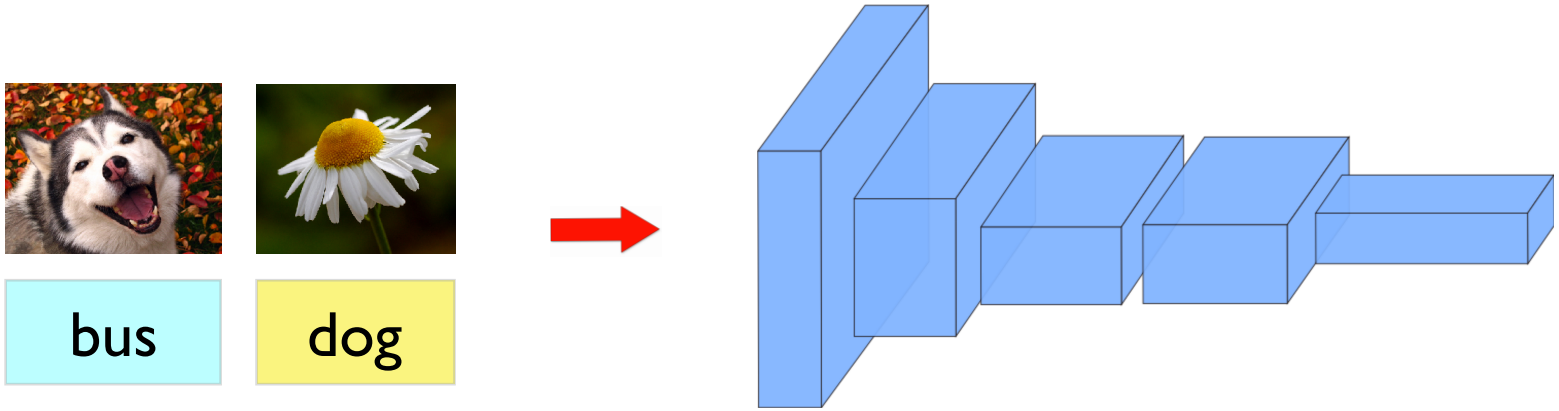
(a) Standard Neural Net (b) After applying dropout.

Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

Peculiarities of Deep Learning

Randomization Test (Zhang et al. ICLR 2017):

- Assign random labels to each image, and then train the network.

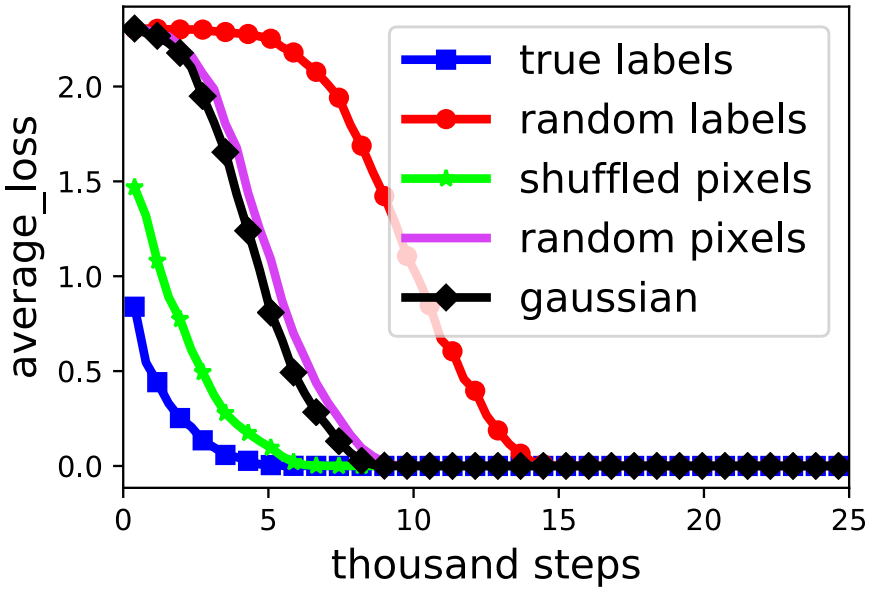


Randomly shuffle the labels

Peculiarities of Deep Learning

Randomization Test (Zhang et al. ICLR 2017):

- Let's look at training losses for several different cases.



- **The network manages to achieve 100% training accuracy even when there is nothing to learn.**
- **Unlike what we thought, regularization doesn't prevent memorization**

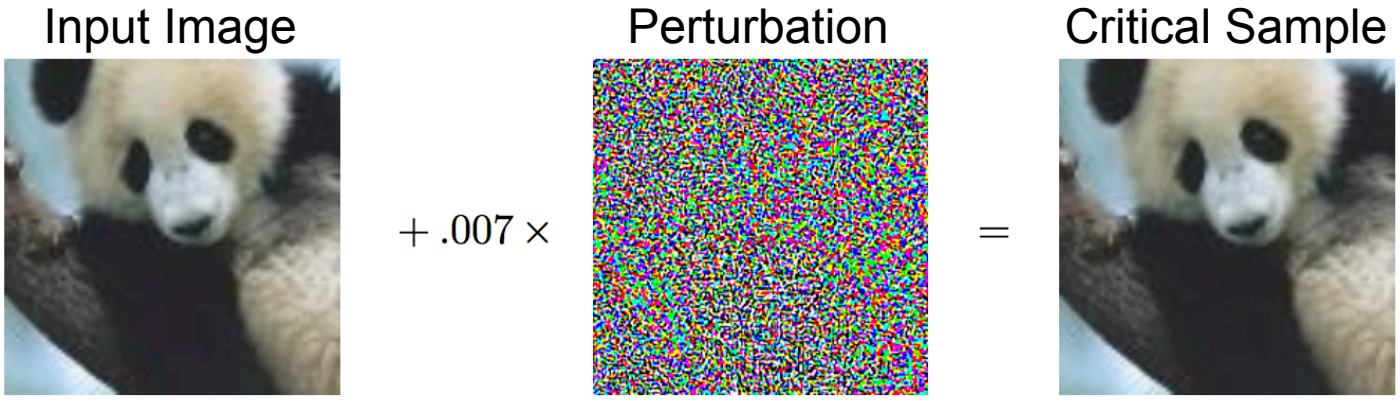
Understanding CNNs

Key Question:

- How can we better understand when CNNs are learning and when they are simply memorizing?

Conjecture (Arpit et al. ICML 2017):

- If the CNN produces a large number of decision hypotheses in the local neighborhood of the data sample space, it is likely **memorizing** and not **learning.**
- Critical sample:



CNN Prediction: "Panda" 57.7 %

CNN Prediction: "Gibbon" 99.99 %

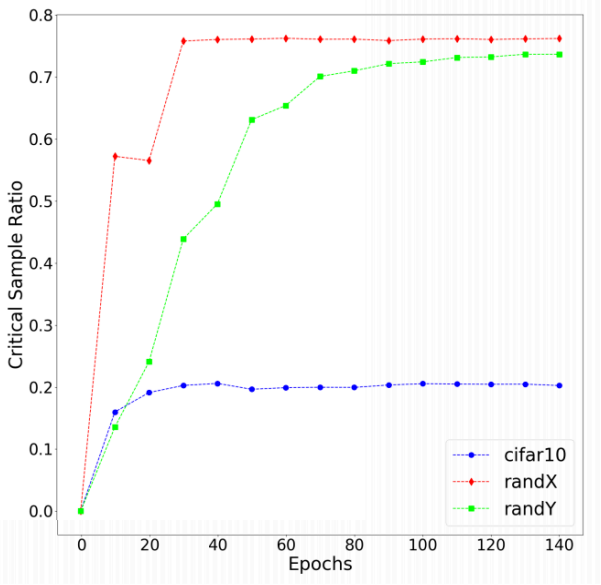
Understanding CNNs

Key Question:

- How can we better understand when CNNs are learning and when they are simply memorizing?

Conjecture (Arpit et al. ICML 2017):

- If the CNN produces a large number of decision hypotheses in the local neighborhood of the data sample space, it is likely **memorizing** and not **learning.**



- **Critical sample ratio is much higher when training on random data**
- **The CNN is memorizing random data but not the real data.**

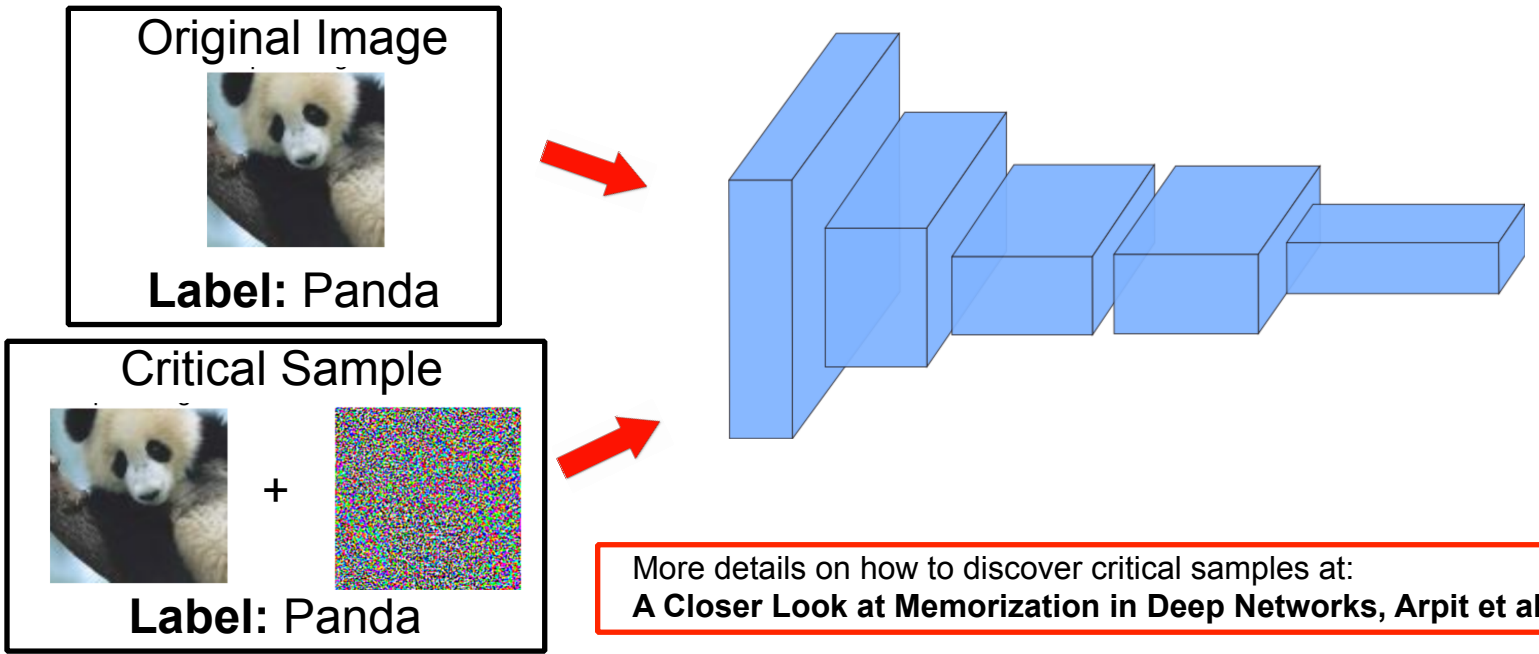
Understanding CNNs

Question:

- Can we use critical samples to prevent network memorization?

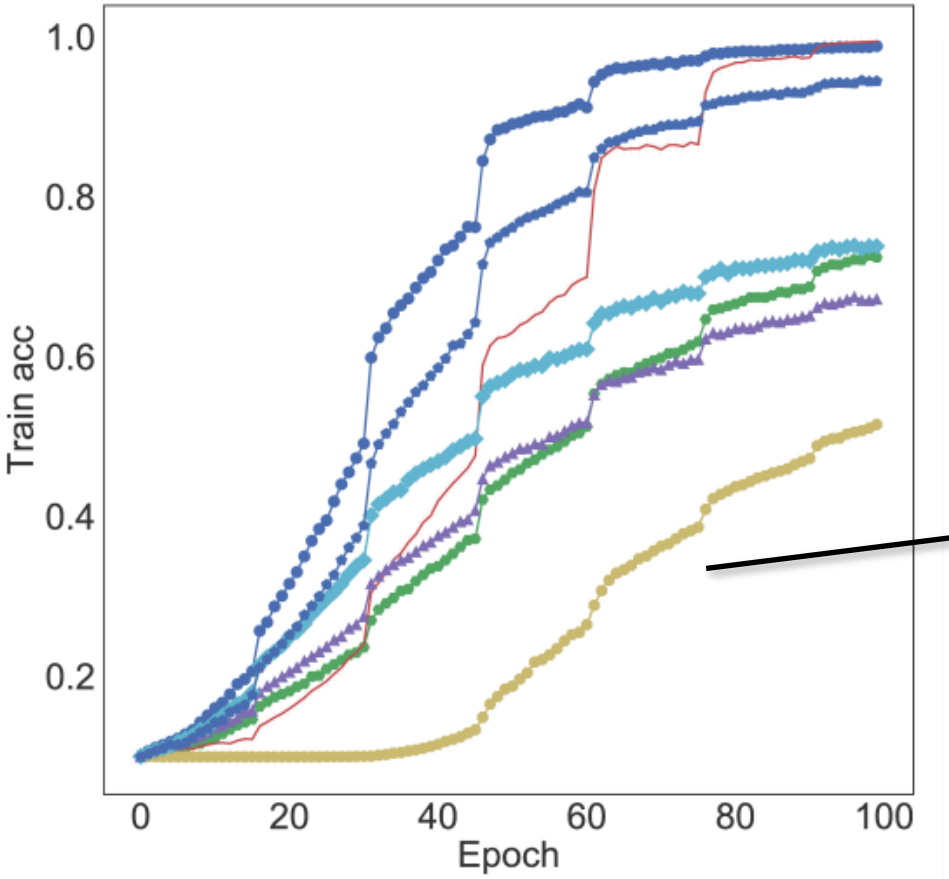
Experiment (Arpit et al. ICML 2017):

- Generate critical samples that fool the CNN.
- Use them as additional examples to train the CNN.



More details on how to discover critical samples at:
A Closer Look at Memorization in Deep Networks, Arpit et al. 2017

Understanding CNNs

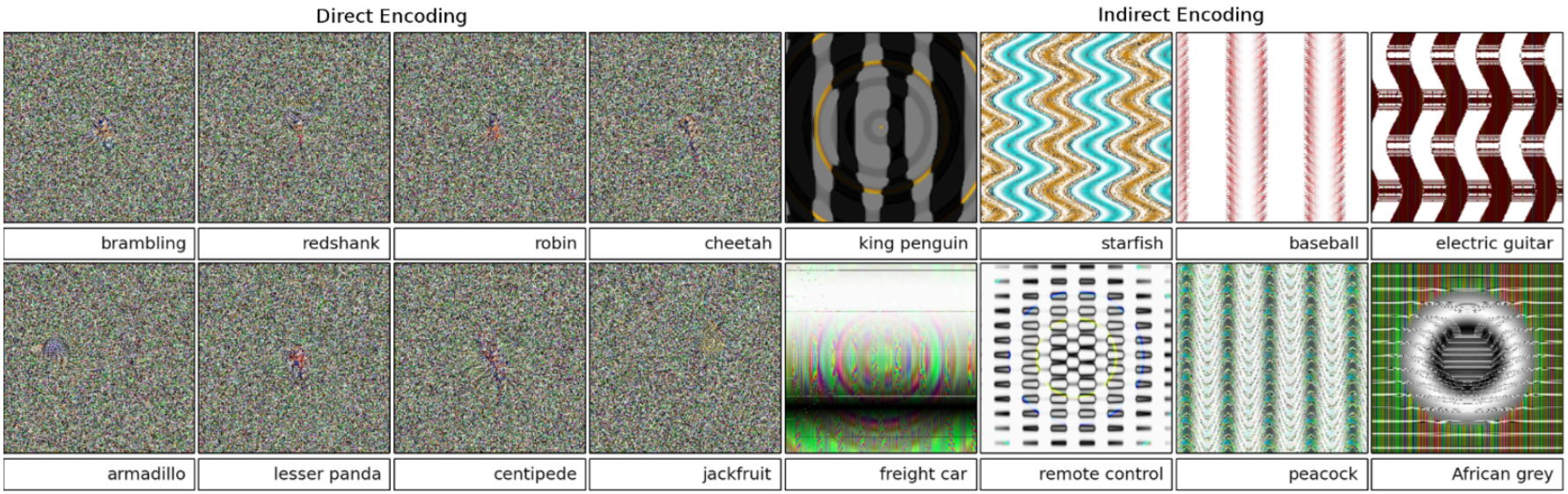


This new training strategy paired with dropout reduces the CNN memorization on random data

Visualizing CNNs

CNNs are easily fooled (Nguyen et al. CVPR 2015):

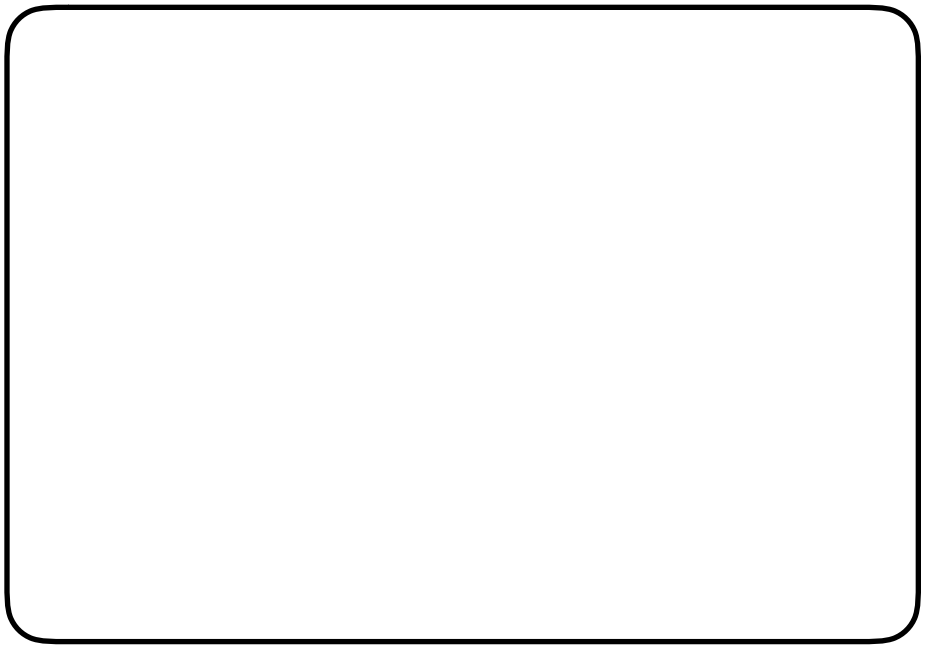
- It turns out that even when the labels aren't random the CNNs learn patterns in the data that don't make much sense.



Visualizing CNNs

CNNs are easily fooled (Nguyen et al. CVPR 2015):

- It turns out that even when the labels aren't random the CNNs learn patterns in the data that don't make much sense.



Data Manifold

Visualizing CNNs

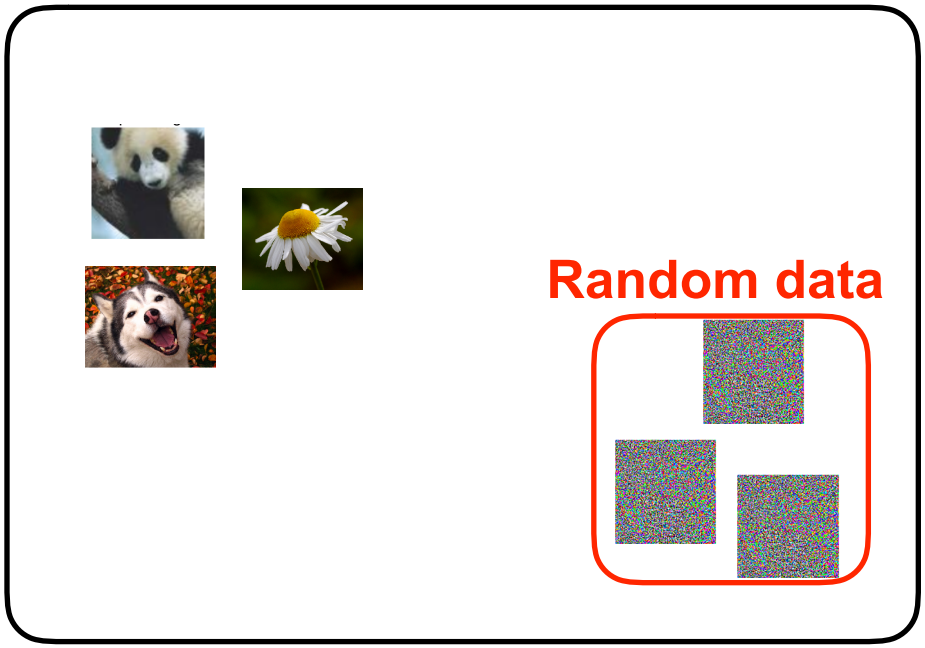
CNNs are easily fooled (Nguyen et al. CVPR 2015):

- It turns out that even when the labels aren't random the CNNs learn patterns in the data that don't make much sense.

Visualizing CNNs

CNNs are easily fooled (Nguyen et al. CVPR 2015):

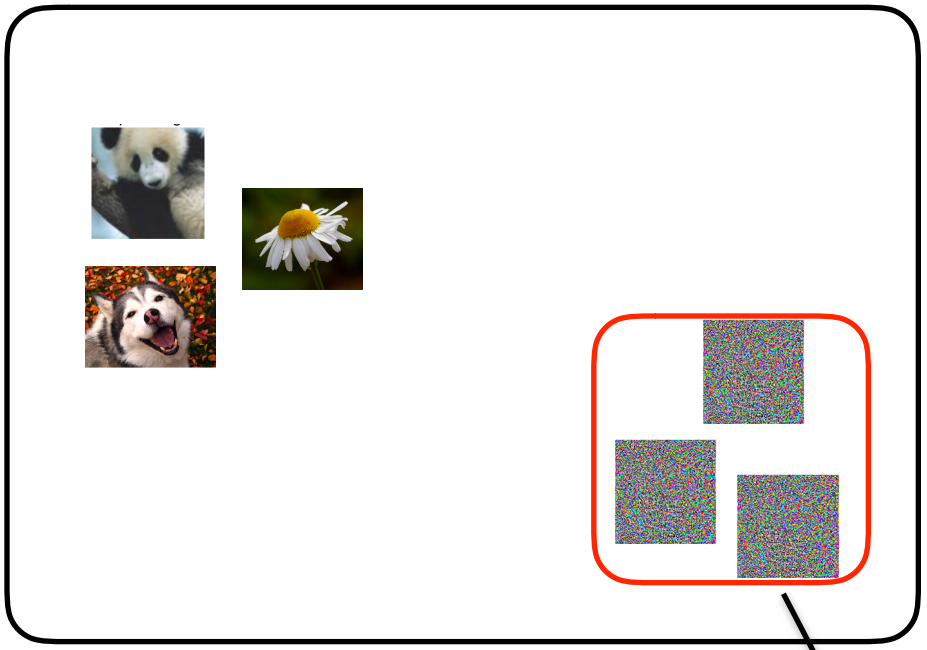
- It turns out that even when the labels aren't random the CNNs learn patterns in the data that don't make much sense.



Visualizing CNNs

CNNs are easily fooled (Nguyen et al. CVPR 2015):

- It turns out that even when the labels aren't random the CNNs learn patterns in the data that don't make much sense.



Visualizing CNNs

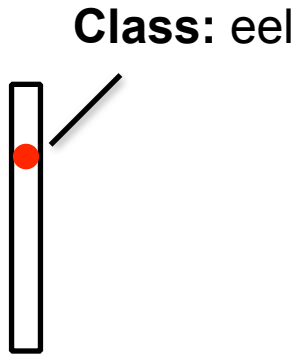
CNNs are easily fooled (Nguyen et al. CVPR 2015):

- How do we generate these fake images?

Visualizing CNNs

CNNs are easily fooled (Nguyen et al. CVPR 2015):

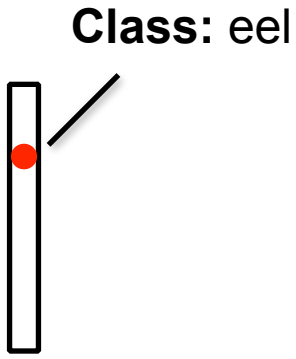
- How do we generate these fake images?
- Assume we want to generate an image that CNN thinks belongs to an eel class.



Visualizing CNNs

CNNs are easily fooled (Nguyen et al. CVPR 2015):

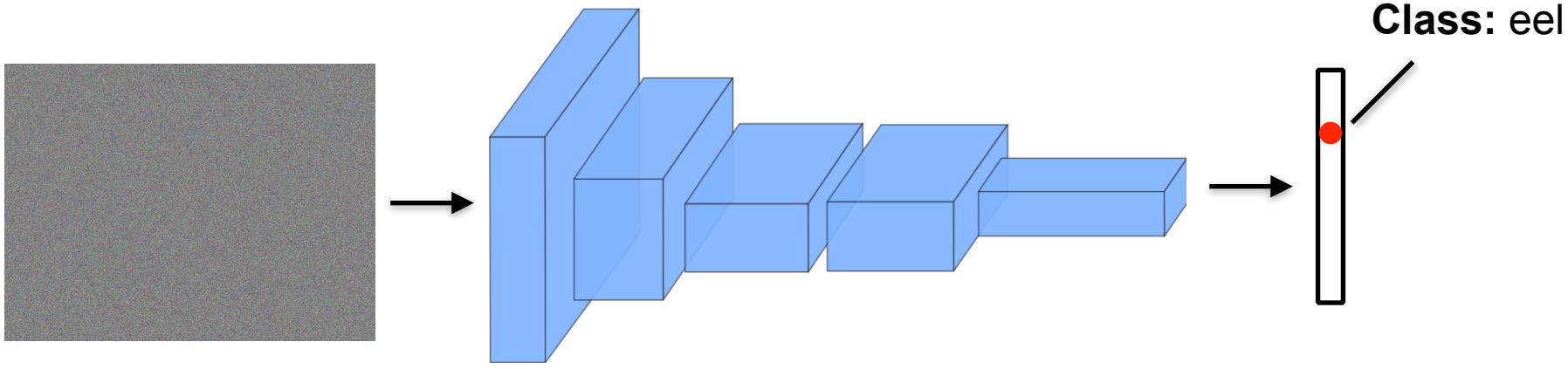
- How do we generate these fake images?
- Assume we want to generate an image that CNN thinks belongs to an eel class.
- Initialize an image with a Gaussian.



Visualizing CNNs

CNNs are easily fooled (Nguyen et al. CVPR 2015):

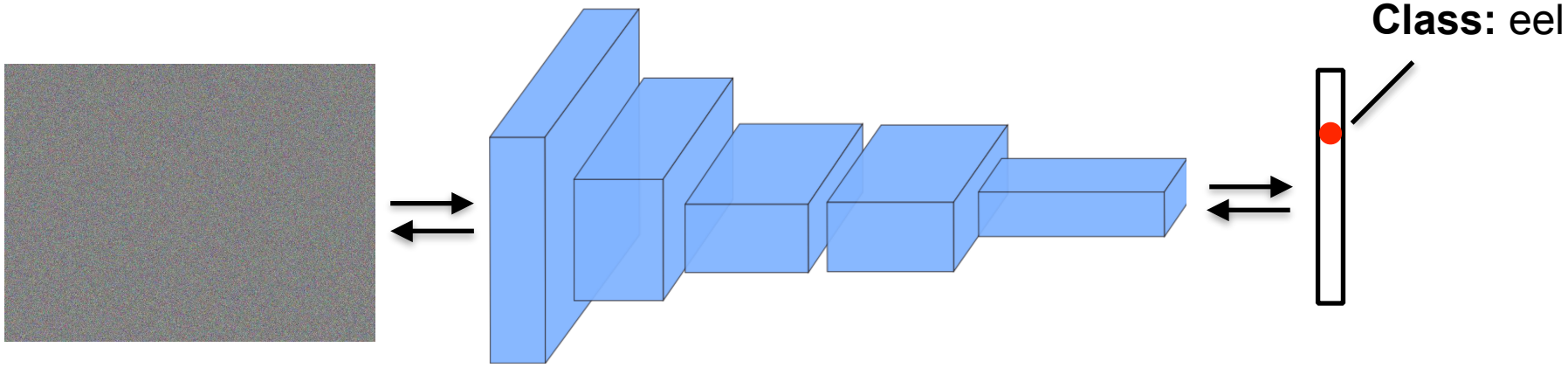
- How do we generate these fake images?
- Assume we want to generate an image that CNN thinks belongs to an eel class.
- Initialize an image with a Gaussian.
- Feed the image through the CNN.



Visualizing CNNs

CNNs are easily fooled (Nguyen et al. CVPR 2015):

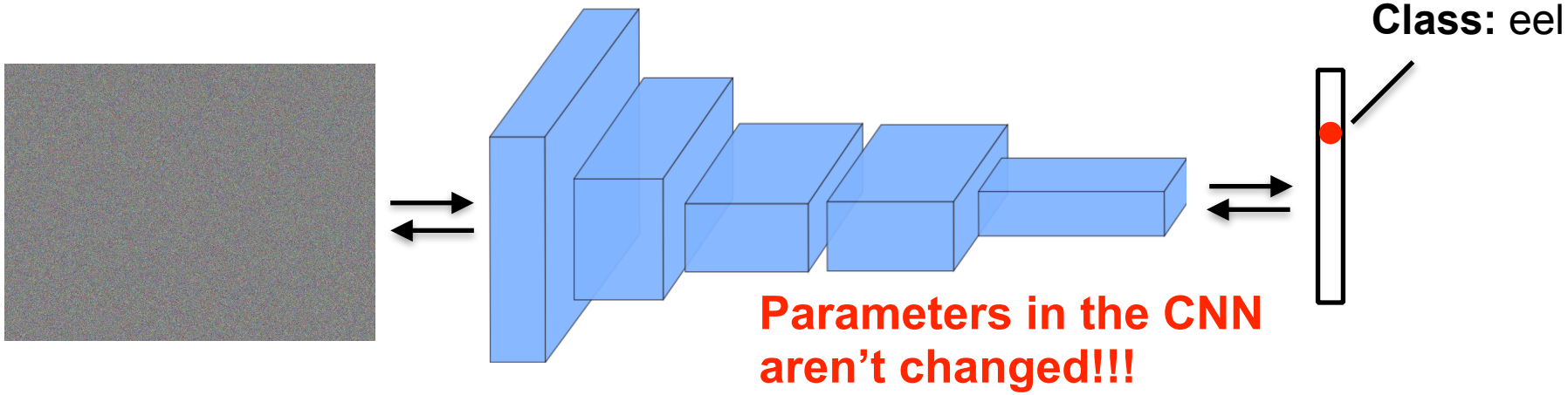
- How do we generate these fake images?
- Assume we want to generate an image that CNN thinks belongs to an eel class.
- Initialize an image with a Gaussian.
- Feed the image through the CNN.
- Backpropgate the gradients and update the image such that predictions for “class: eel” would be maximized.



Visualizing CNNs

CNNs are easily fooled (Nguyen et al. CVPR 2015):

- How do we generate these fake images?
- Assume we want to generate an image that CNN thinks belongs to an eel class.
- Initialize an image with a Gaussian.
- Feed the image through the CNN.
- Backpropgate the gradients and update the image such that predictions for “class: eel” would be maximized.

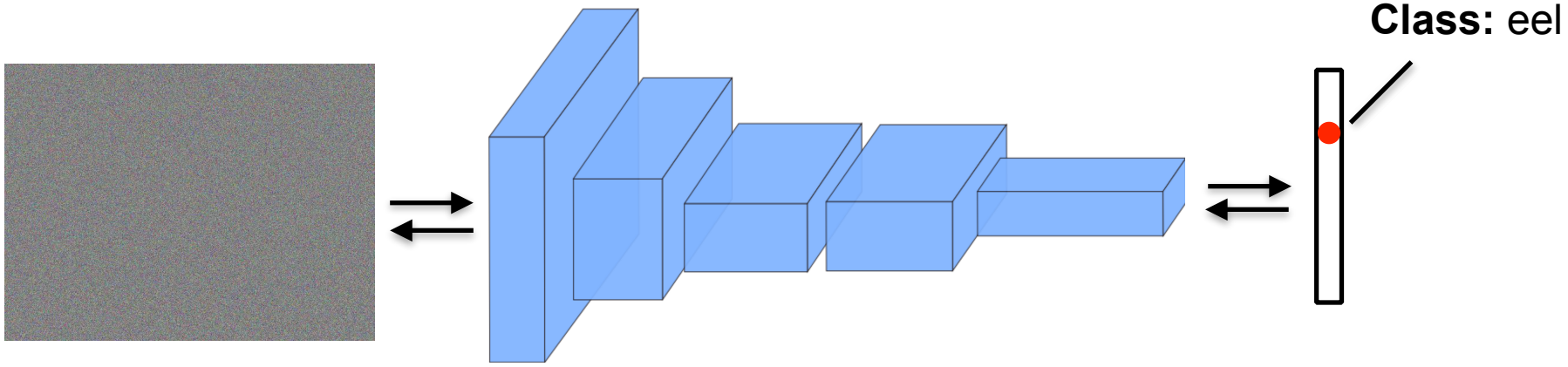


Visualizing CNNs

CNNs are easily fooled (Nguyen et al. CVPR 2015):

- Formally this can be understood as minimizing the following learning objective:

$$L = -\sum_{k=1}^{K} 1\{y = k\} \log \hat{y}_k$$



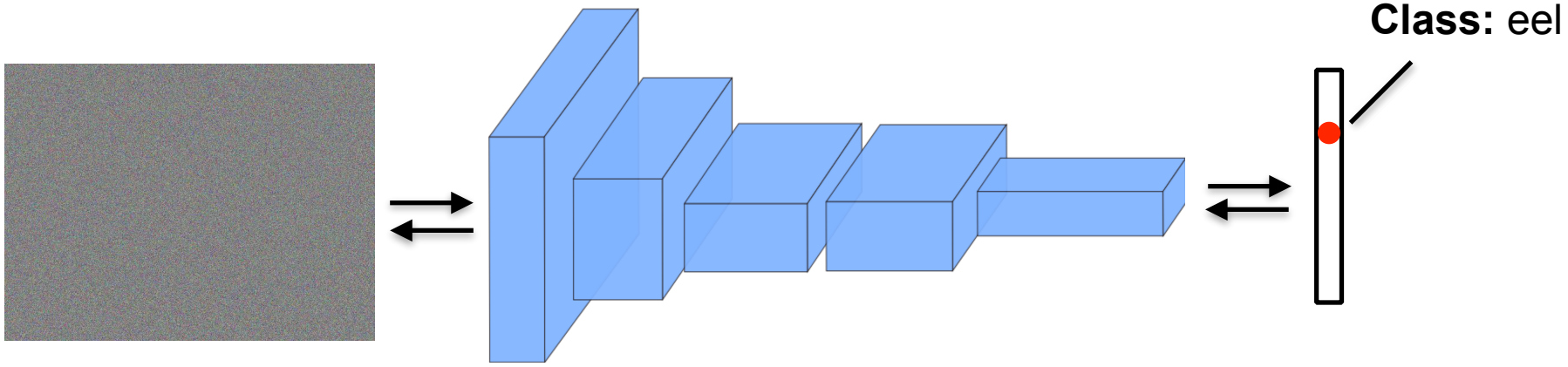
Visualizing CNNs

CNNs are easily fooled (Nguyen et al. CVPR 2015):

- Formally this can be understood as minimizing the following learning objective:

$$L = -\sum_{k=1}^{K} 1\{y = k\} \log \hat{y}_k$$

indicator function that takes a value of 1 if y==k



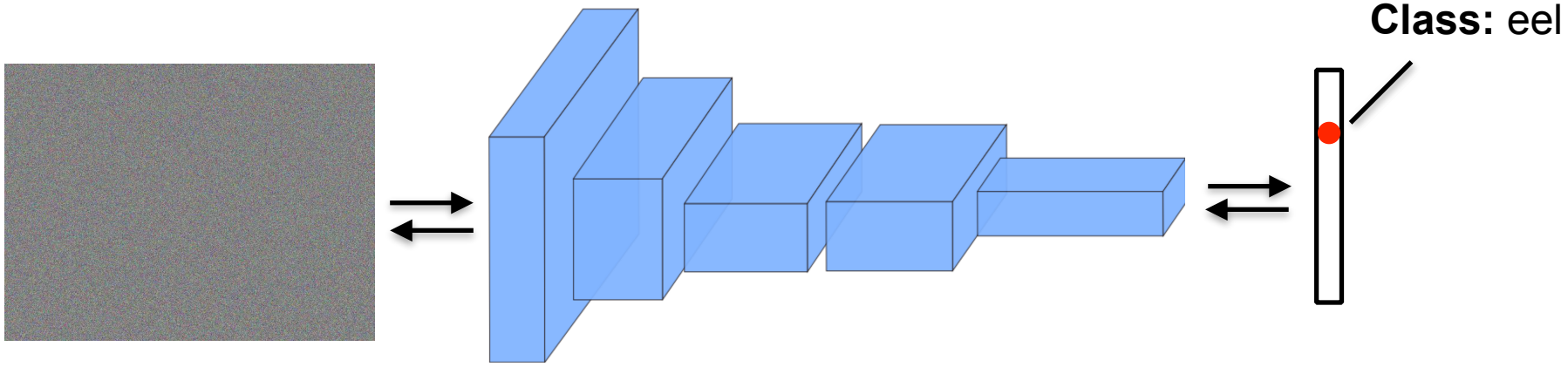
Visualizing CNNs

CNNs are easily fooled (Nguyen et al. CVPR 2015):

- Formally this can be understood as minimizing the following learning objective:

$$L = -\sum_{k=1}^{K} 1\{y = k\} \log \hat{y}_k$$

the index of a selected object class (e.g. eel)



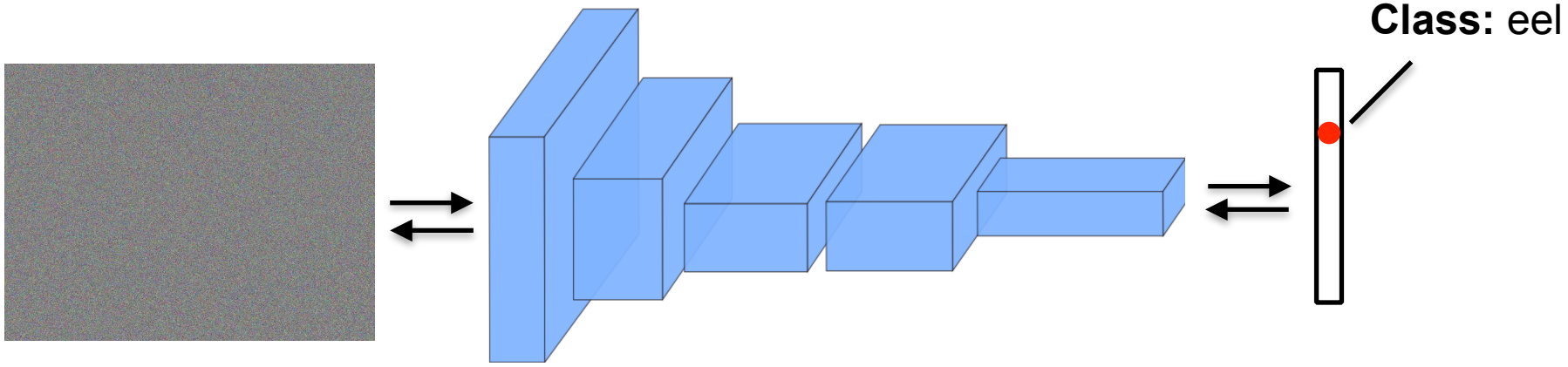
Visualizing CNNs

CNNs are easily fooled (Nguyen et al. CVPR 2015):

- Formally this can be understood as minimizing the following learning objective:

$$L = -\sum_{k=1}^{K} 1\{y = k\} \log \hat{y}_k$$

ground truth label



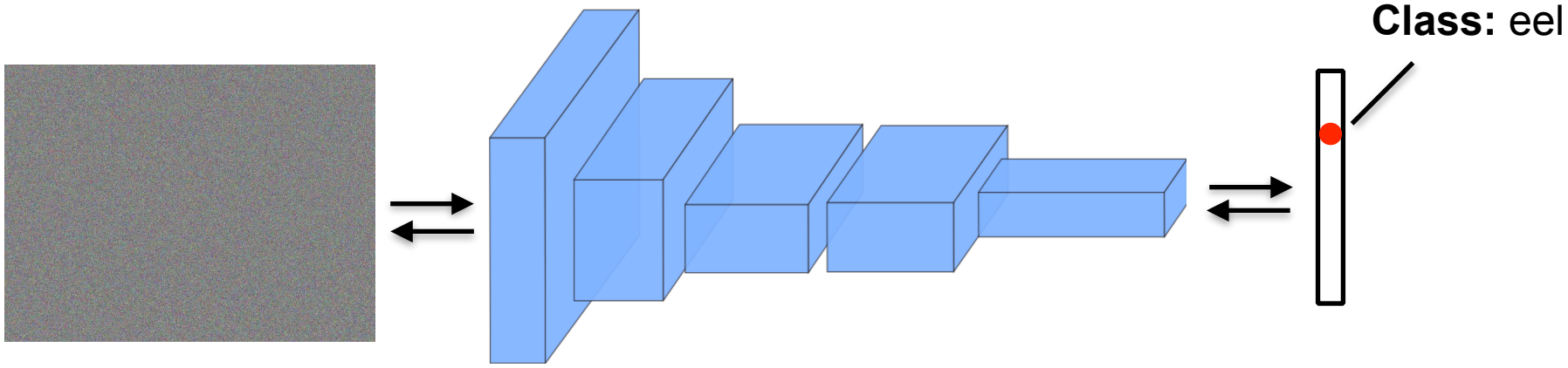
Visualizing CNNs

CNNs are easily fooled (Nguyen et al. CVPR 2015):

- Formally this can be understood as minimizing the following learning objective:

$$L = -\sum_{k=1}^{K} 1\{y = k\} \log \hat{y}_k$$

predicted probability for class k (e.g. eel)



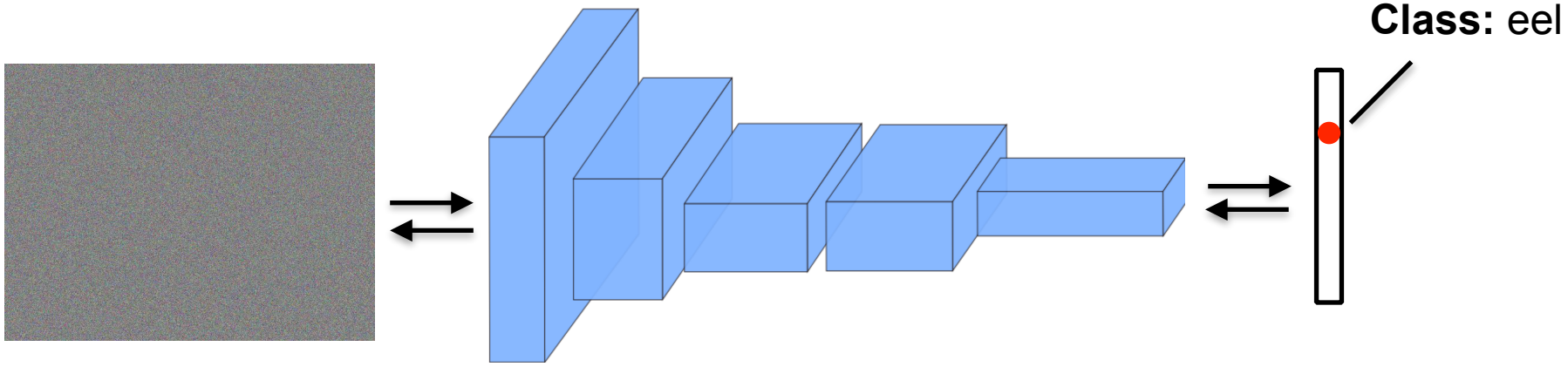
Visualizing CNNs

CNNs are easily fooled (Nguyen et al. CVPR 2015):

- Formally this can be understood as minimizing the following learning objective:

$$L = -\sum_{k=1}^{K} 1\{y = k\} \log \hat{y}_k$$

This loss function can be minimized using standard SGD

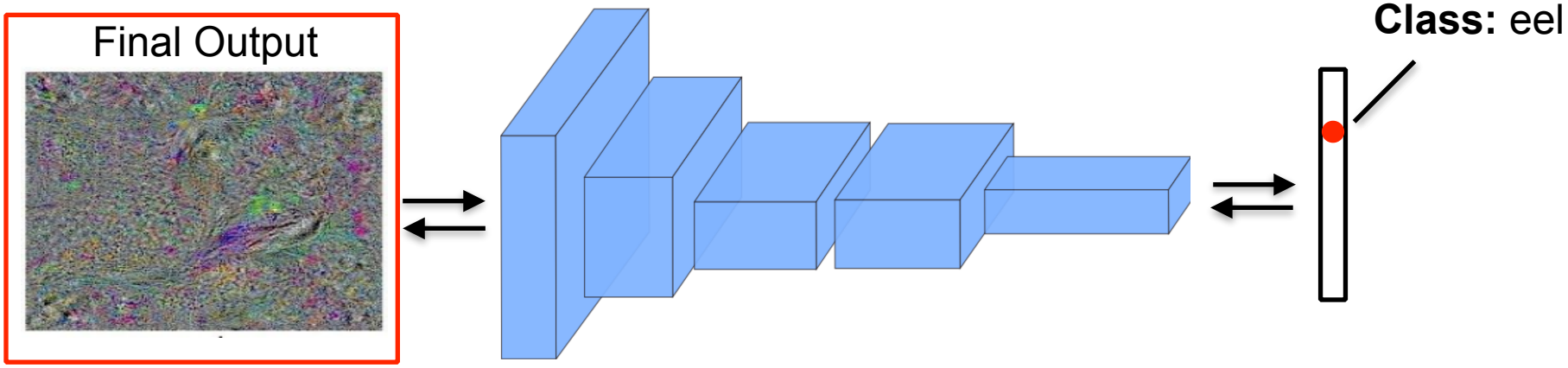


Visualizing CNNs

CNNs are easily fooled (Nguyen et al. CVPR 2015):

- Formally this can be understood as minimizing the following learning objective:

$$L = -\sum_{k=1}^{K} 1\{y = k\} \log \hat{y}_k$$

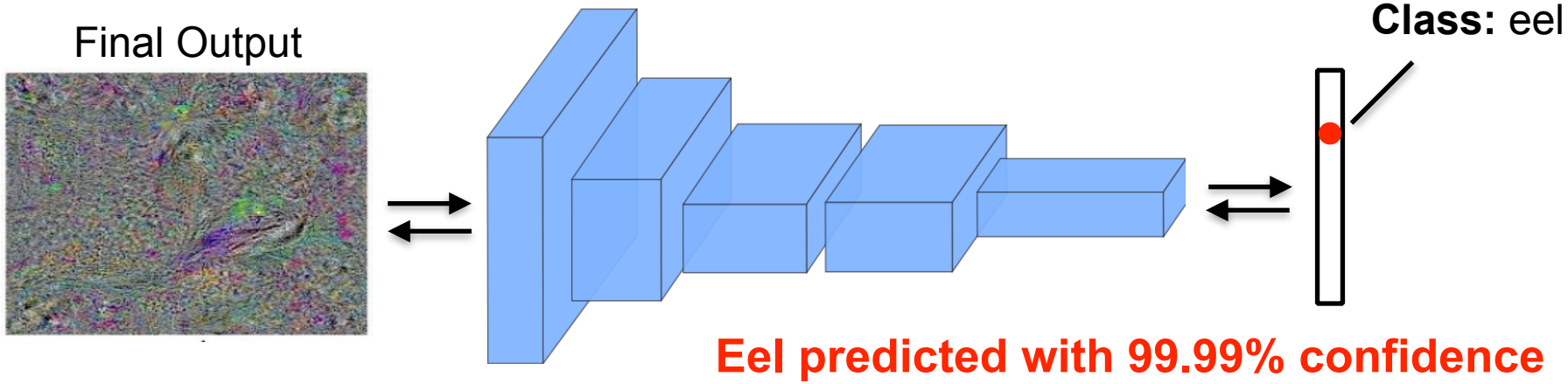


Visualizing CNNs

CNNs are easily fooled (Nguyen et al. CVPR 2015):

- Formally this can be understood as minimizing the following learning objective:

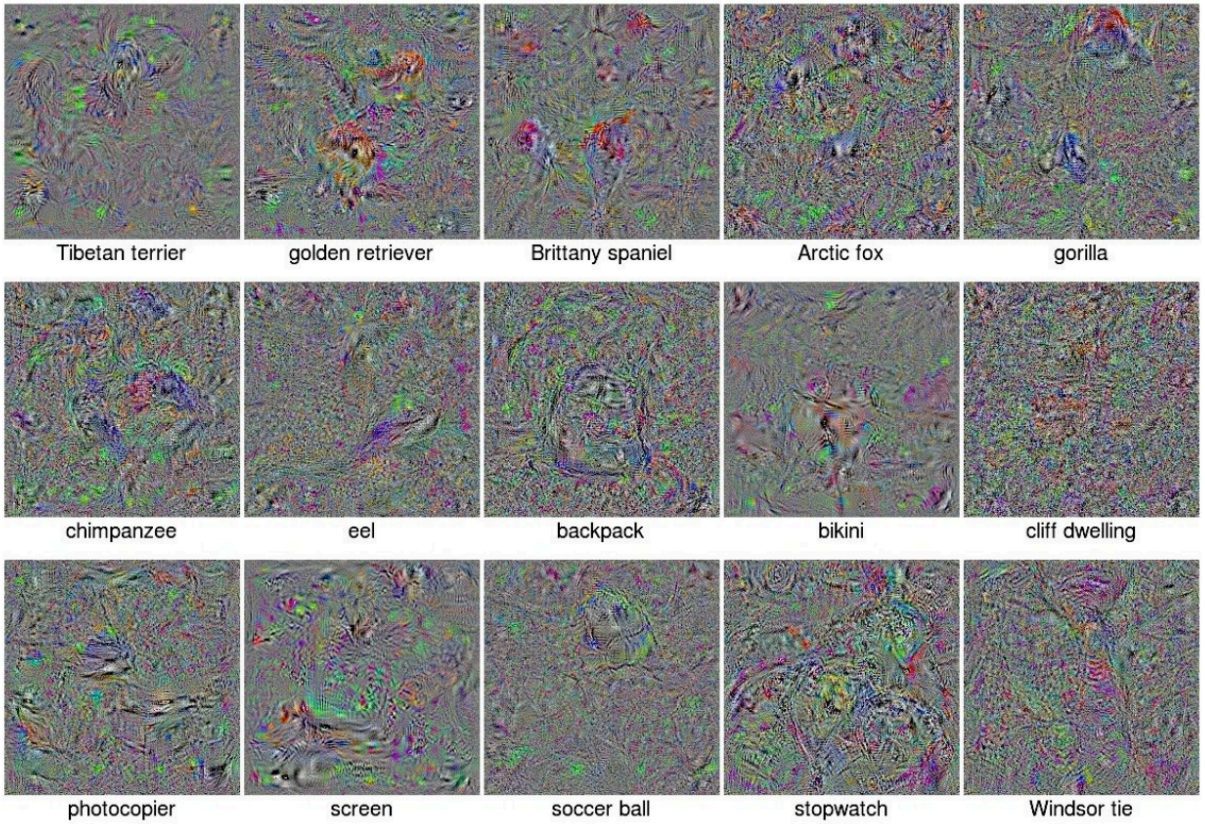
$$L = -\sum_{k=1}^{K} 1\{y = k\} \log \hat{y}_k$$



Visualizing CNNs

CNNs are easily fooled (Nguyen et al. CVPR 2015):

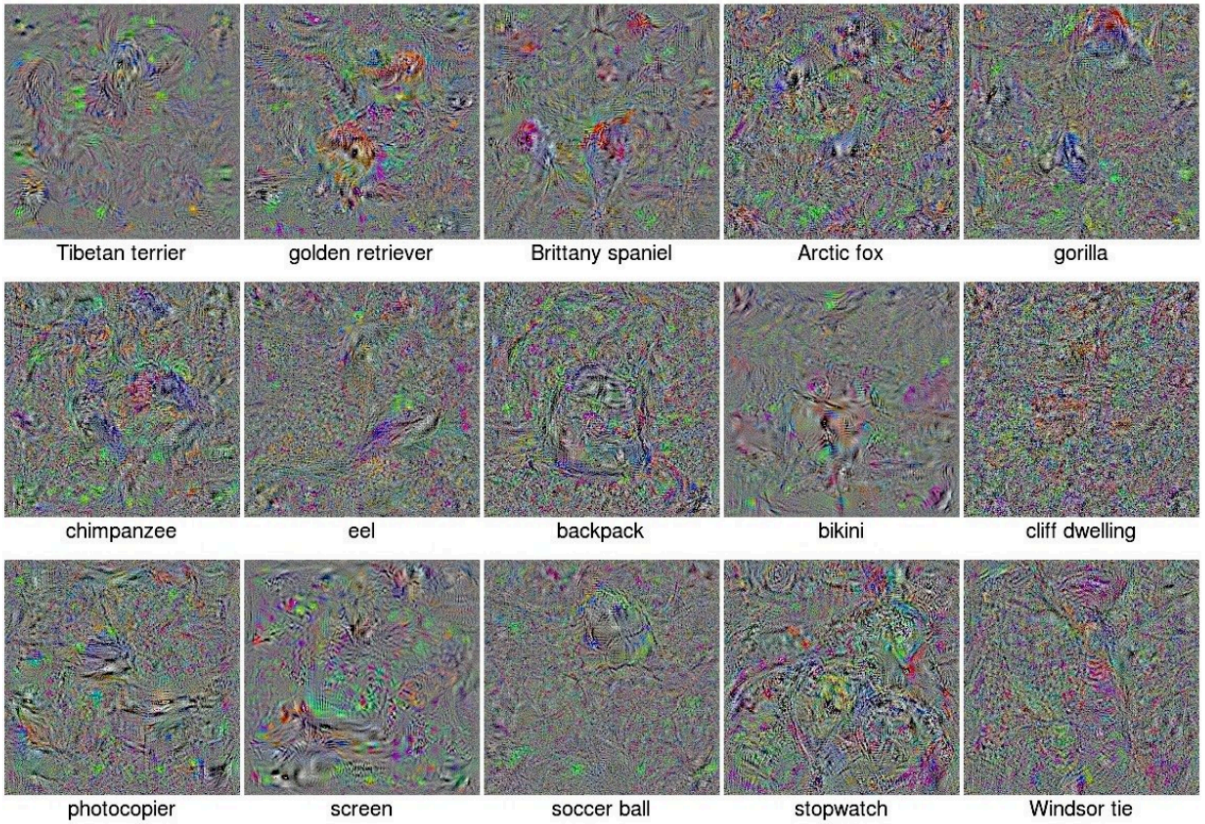
- More examples:



Visualizing CNNs

CNNs are easily fooled (Nguyen et al. CVPR 2015):

- More examples:



These images look like noise. Can we do better?

Visualizing CNNs

Genetic Algorithms:

- Inspired by the concept of Darwinian evolution.

Visualizing CNNs

Genetic Algorithms:

- Inspired by the concept of Darwinian evolution.

- Typically we have a certain population of “organisms” (e.g. images) that face natural selection.

Visualizing CNNs

Genetic Algorithms:

- Inspired by the concept of Darwinian evolution.
- Typically we have a certain population of “organisms” (e.g. images) that face natural selection.
- Organisms with the highest fitness function are selected to cross-over with the other selected organisms.

Visualizing CNNs

Genetic Algorithms:

- Inspired by the concept of Darwinian evolution.
- Typically we have a certain population of “organisms” (e.g. images) that face natural selection.
- Organisms with the highest fitness function are selected to cross-over with the other selected organisms.
- Random mutations are inserted in this process.

Visualizing CNNs

Genetic Algorithms:

- Inspired by the concept of Darwinian evolution.
- Typically we have a certain population of “organisms” (e.g. images) that face natural selection.
- Organisms with the highest fitness function are selected to cross-over with the other selected organisms.
- Random mutations are inserted in this process.
- The population is updated with the new organisms and the entire process is repeated.

Visualizing CNNs

Genetic Algorithms:

- Inspired by the concept of Darwinian evolution.
- Typically we have a certain population of “organisms” (e.g. images) that face natural selection.
- Organisms with the highest fitness function are selected to cross-over with the other selected organisms.
- Random mutations are inserted in this process.
- The population is updated with the new organisms and the entire process is repeated.

No convergence rules or any guarantees!

Visualizing CNNs

Vocabulary:

- **Individual:** any possible solution (e.g. image in this case)
- **Population:** a group of all individuals
- **Fitness:** a target function that we are optimizing (e.g. maximizing a certain output of a CNN).

Visualizing CNNs

Vocabulary:

- **Individual:** any possible solution (e.g. image in this case)
- **Population:** a group of all individuals
- **Fitness:** a target function that we are optimizing (e.g. maximizing a certain output of a CNN).

Toy Example:

- Suppose we have a population of binary strings of length 10.

Visualizing CNNs

Vocabulary:

- **Individual:** any possible solution (e.g. image in this case)
- **Population:** a group of all individuals
- **Fitness:** a target function that we are optimizing (e.g. maximizing a certain output of a CNN).

Toy Example:

- Suppose we have a population of binary strings of length 10.
- We want the evolutionary algorithm to evolve to discover individuals with high fitness function

Visualizing CNNs

Vocabulary:

- **Individual:** any possible solution (e.g. image in this case)
- **Population:** a group of all individuals
- **Fitness:** a target function that we are optimizing (e.g. maximizing a certain output of a CNN).

Toy Example:

- Suppose we have a population of binary strings of length 10.
- We want the evolutionary algorithm to evolve to discover individuals with high fitness function
- Suppose that in this case our fitness function is determined by the number of 1s in the string (the higher the better).

Visualizing CNNs

Toy Example:

- Initially we generate a bunch of random binary strings.

$s_1 = 1111010101 \quad f(s_1) = 7$

$s_2 = 0111000101 \quad f(s_2) = 5$

$s_3 = 1110110101 \quad f(s_3) = 7$

$s_4 = 0100010011 \quad f(s_4) = 4$

$s_5 = 1110111101 \quad f(s_5) = 8$

$s_6 = 0100110000 \quad f(s_6) = 3$

Visualizing CNNs

Toy Example:

- Initially we generate a bunch of random binary strings.

$s_1 = 1111010101 \quad f(s_1) = 7$

$s_2 = 0111000101 \quad f(s_2) = 5$

$s_3 = 1110110101 \quad f(s_3) = 7$

$s_4 = 0100010011 \quad f(s_4) = 4$

$s_5 = 1110111101 \quad f(s_5) = 8$

$s_6 = 0100110000 \quad f(s_6) = 3$

- Randomly select an individual with a probability $\frac{f(i)}{\sum_i f(i)}$

Visualizing CNNs

Toy Example:

- Initially we generate a bunch of random binary strings.

$s_1 = 1111010101 \quad f(s_1) = 7$

$s_2 = 0111000101 \quad f(s_2) = 5$

$s_3 = 1110110101 \quad f(s_3) = 7$

$s_4 = 0100010011 \quad f(s_4) = 4$

$s_5 = 1110111101 \quad f(s_5) = 8$

$s_6 = 0100110000 \quad f(s_6) = 3$

- Randomly select an individual with a probability $\frac{f(i)}{\sum_i f(i)}$

- The same individual can be selected multiple times

Visualizing CNNs

Toy Example:

- The population of selected individuals

s_1` = 1111010101 (s_1)

s_2` = 1110110101 (s_3)

s_3` = 1110111101 (s_5)

s_4` = 0111000101 (s_2)

s_5` = 0100010011 (s_4)

s_6` = 1110111101 (s_5)

Visualizing CNNs

Toy Example:

- The population of selected individuals

s_1` = 1111010101 (s_1)

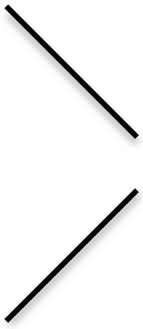
s_2` = 1110110101 (s_3)

s_3` = 1110111101 (s_5)

s_4` = 0111000101 (s_2)

s_5` = 0100010011 (s_4)

s_6` = 1110111101 (s_5)



Individuals with high fitness function are likely to be selected multiple times

Visualizing CNNs

Toy Example:

- Now we mate strings for cross over:

Visualizing CNNs

Toy Example:

- Now we mate strings for cross over:

s_1` = 11110**1**0101 $s_2$` = 11**1**0110101

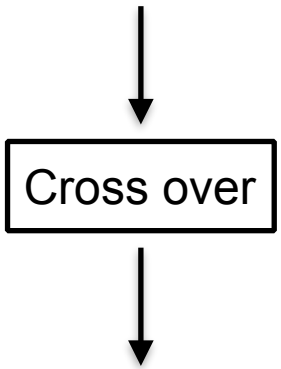
Randomly selected cross over points

Visualizing CNNs

Toy Example:

- Now we mate strings for cross over:

s_1` = 11110101 01 $s_2$` = 11101101 01



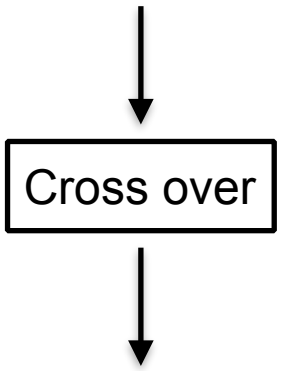
s_1`` = 11101101 01 $s_2$`` = 11110101 01

Visualizing CNNs

Toy Example:

- Now we mate strings for cross over:

s_1` = 11110101 $s_2$` = 11101101



s_1`` = 11101101 $s_2$`` = 11110101

The cross over points have been flipped among both individuals.

Visualizing CNNs

Toy Example:

- Final step is mutation that allows a small number of bits in a string to be randomly modified.
- Mutations help to increase the diversity among the samples.

Visualizing CNNs

Toy Example:

- Final step is mutation that allows a small number of bits in a string to be randomly modified.
- Mutations help to increase the diversity among the samples.

Initial strings	After mutating
s_1`` = 1110110101	s_1``` = 1110100101
s_2`` = 1111010101	s_2``` = 1111110100
s_3`` = 1110111101	s_3``` = 1110101111
s_4`` = 0111000101	s_4``` = 0111000101
s_5`` = 0100011101	s_5``` = 0100011101
s_6`` = 1110110011	s_6``` = 1110110001

Visualizing CNNs

Toy Example:

- Final step is mutation that allows a small number of bits in a string to be randomly modified.
- Mutations help to increase the diversity among the samples.

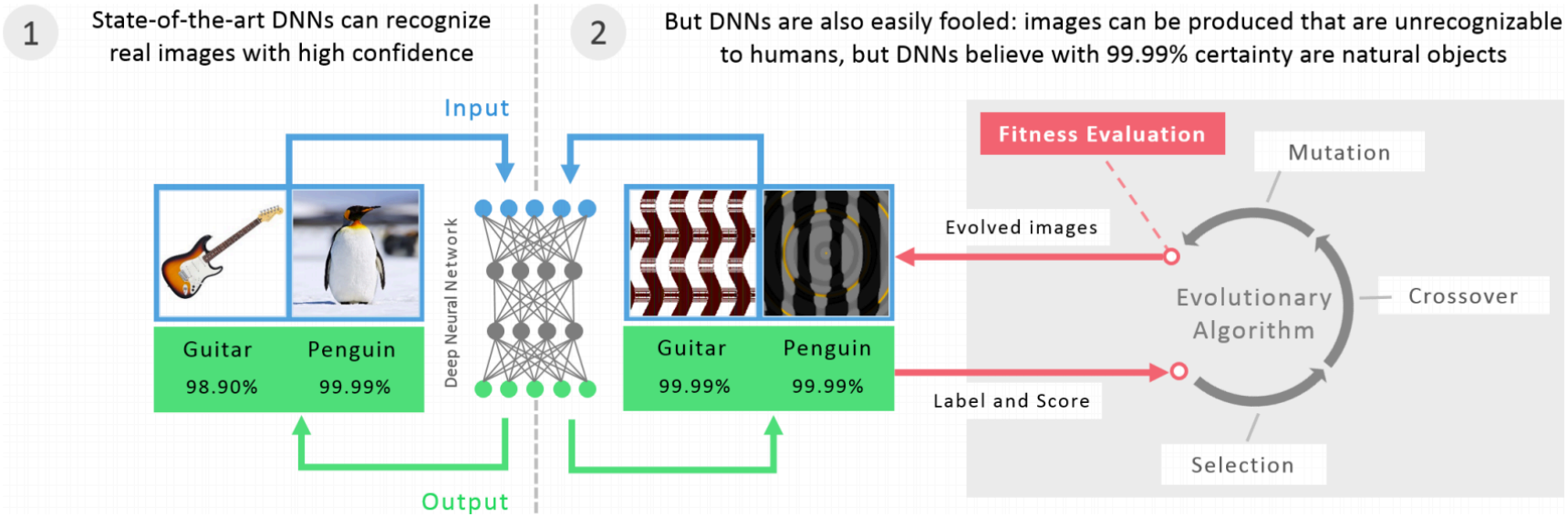
Initial strings	After mutating
s_1`` = 1110110101	s_1``` = 1110100101
s_2`` = 1111010101	s_2``` = 1111110100
s_3`` = 1110111101	s_3``` = 1110101111
s_4`` = 0111000101	s_4``` = 0111000101
s_5`` = 0100011101	s_5``` = 0100011101
s_6`` = 1110110011	s_6``` = 1110110001

Keep iterating until individuals evolve to satisfy the fitness function.

Visualizing CNNs

CNNs are easily fooled (Nguyen et al. CVPR 2015):

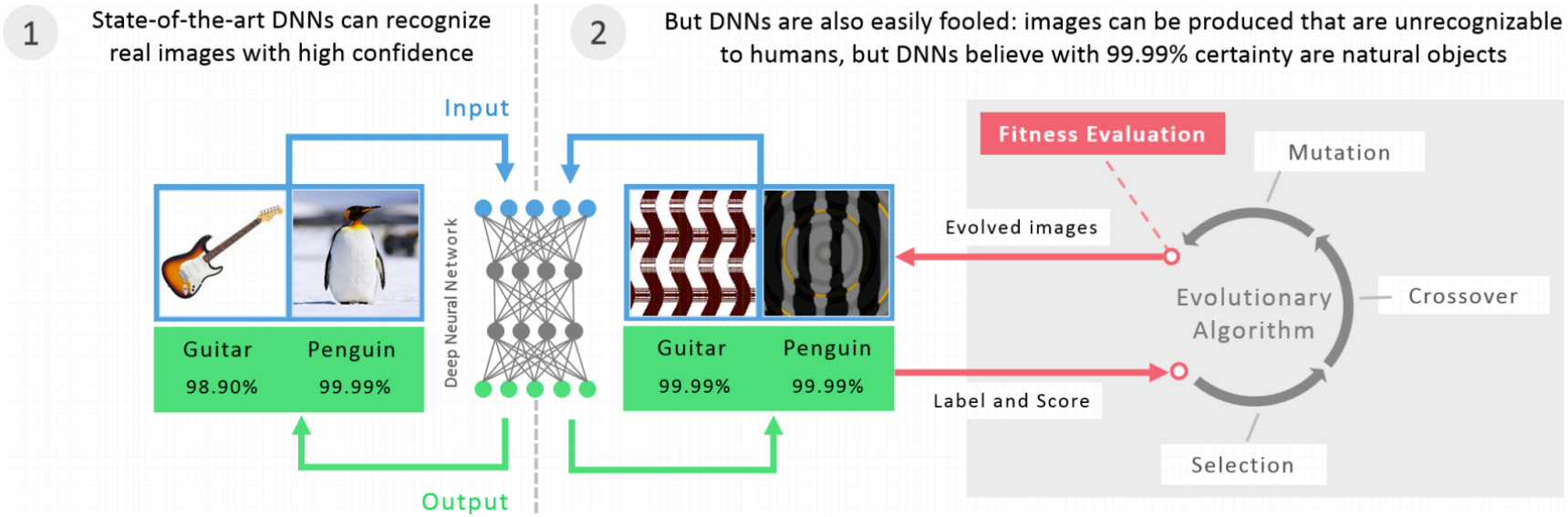
- Pipeline to generate images using evolutionary algorithms



Visualizing CNNs

CNNs are easily fooled (Nguyen et al. CVPR 2015):

- Pipeline to generate images using evolutionary algorithms

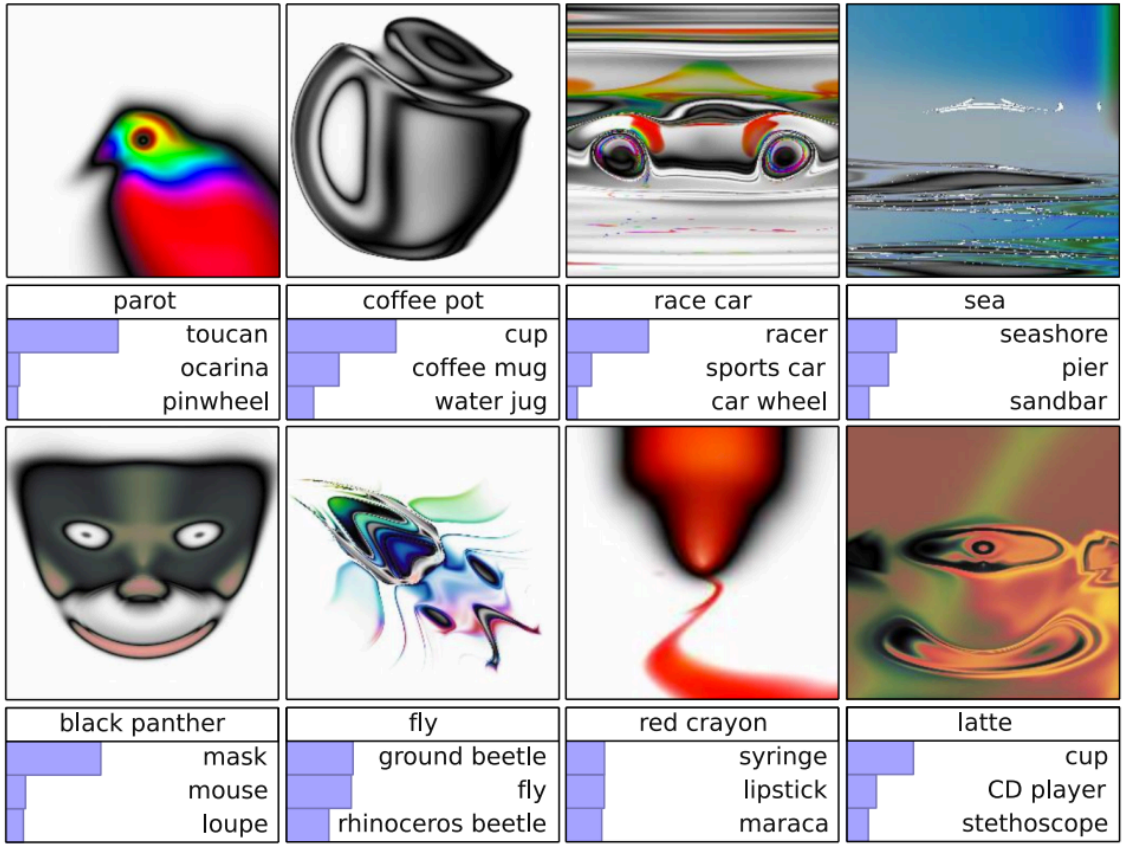


- **Individual is an image**
- **Each trait is a pixel in an image**
- **Fitness function is the CNN's output for a certain class**

Visualizing CNNs

CNNs are easily fooled (Nguyen et al. CVPR 2015):

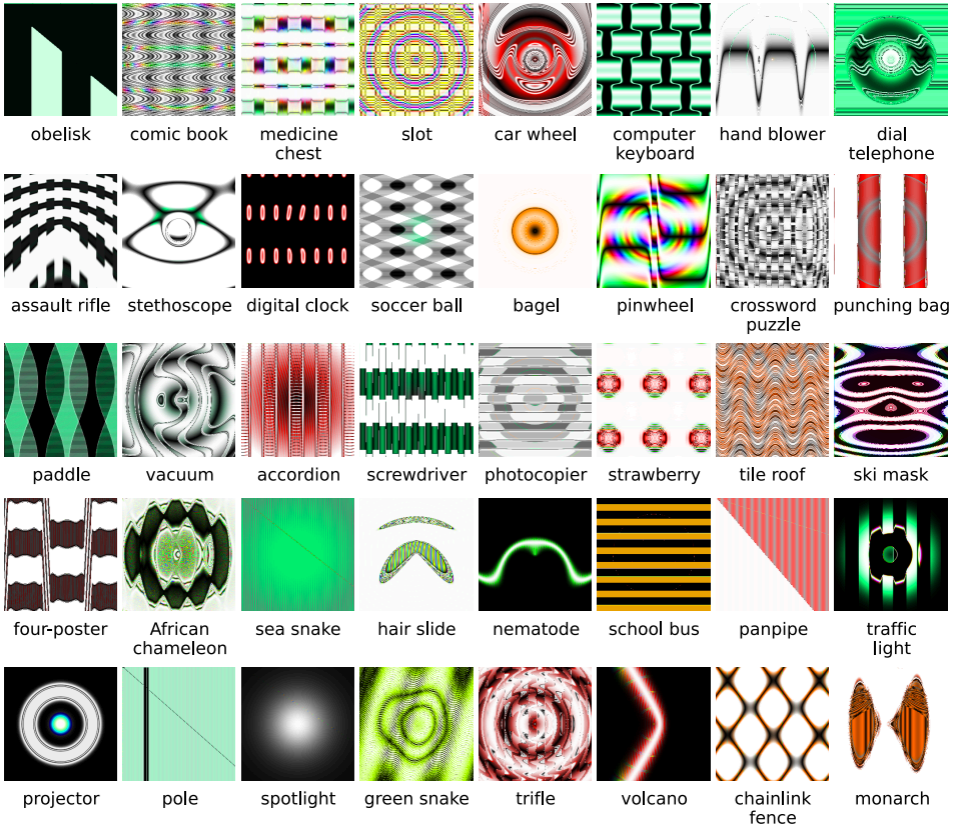
- Examples:



Visualizing CNNs

CNNs are easily fooled (Nguyen et al. CVPR 2015):

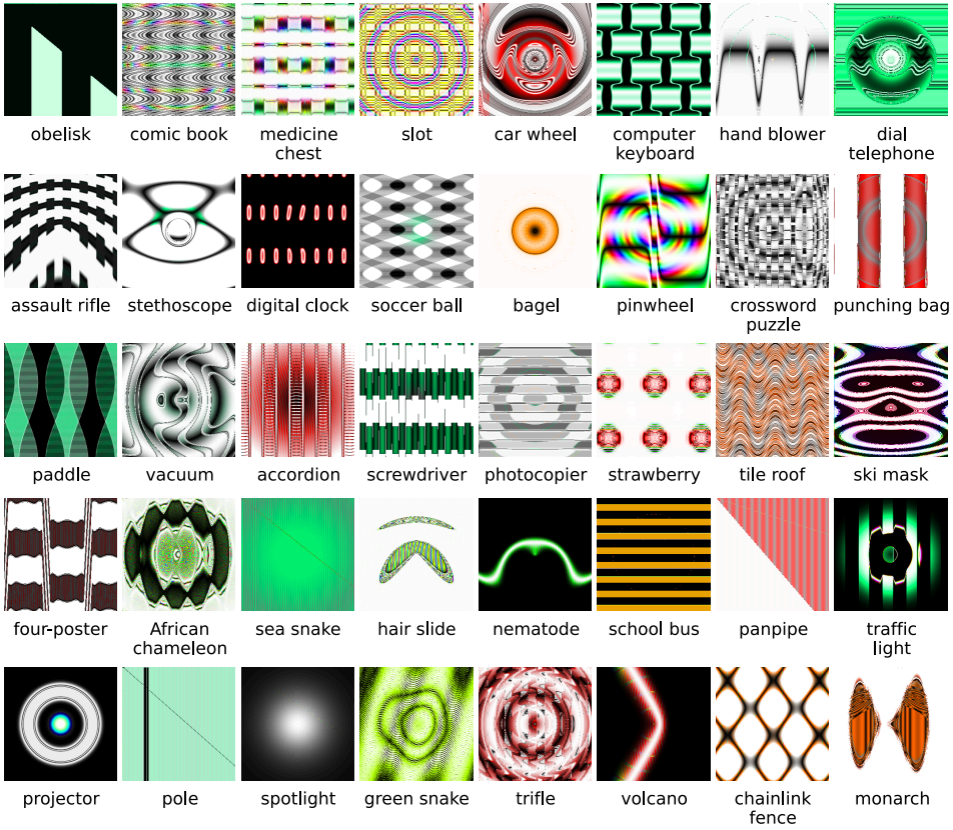
- Examples:



Visualizing CNNs

CNNs are easily fooled (Nguyen et al. CVPR 2015):

- Examples:

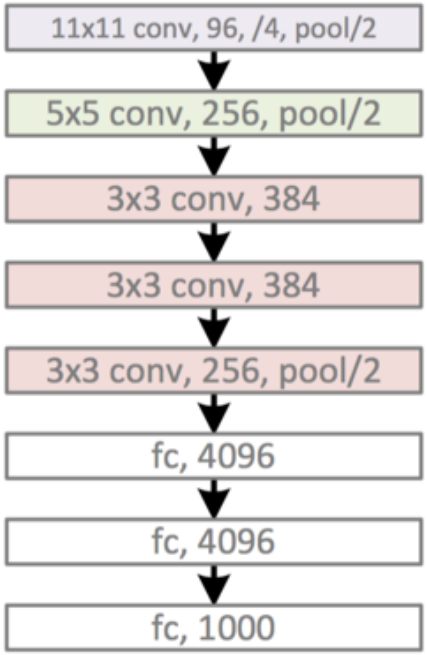


Average CNN prediction probability is 99.12% across these images.

Visualizing CNNs

Understanding CNNs thru visualization:

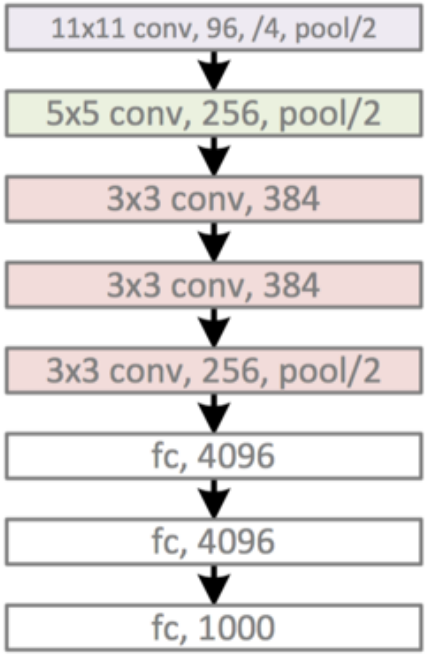
- Can we use visualizations to understand the behavior of the hidden units in the CNN?
- What do specific hidden units in the CNN learn to recognize?

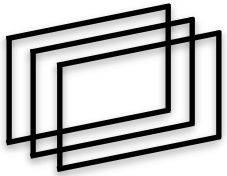


Visualizing CNNs

Understanding CNNs thru visualization:

- Can we use visualizations to understand the behavior of the hidden units in the CNN?
- What do specific hidden units in the CNN learn to recognize?



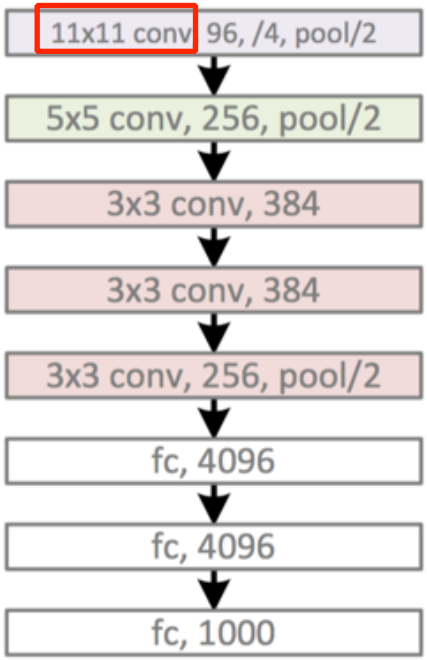


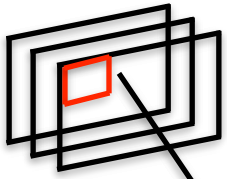
RGB

Visualizing CNNs

Understanding CNNs thru visualization:

- Can we use visualizations to understand the behavior of the hidden units in the CNN?
- What do specific hidden units in the CNN learn to recognize?



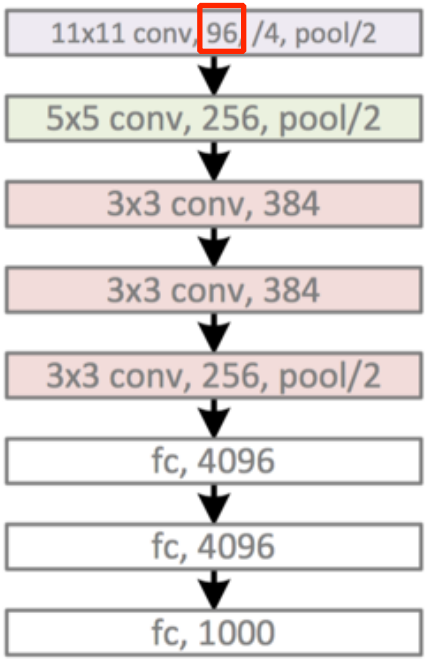


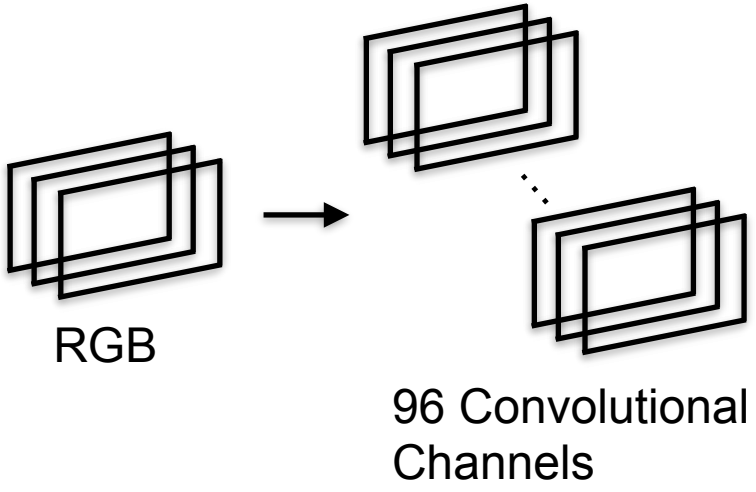
- **11x11 convolutional filter**

Visualizing CNNs

Understanding CNNs thru visualization:

- Can we use visualizations to understand the behavior of the hidden units in the CNN?
- What do specific hidden units in the CNN learn to recognize?

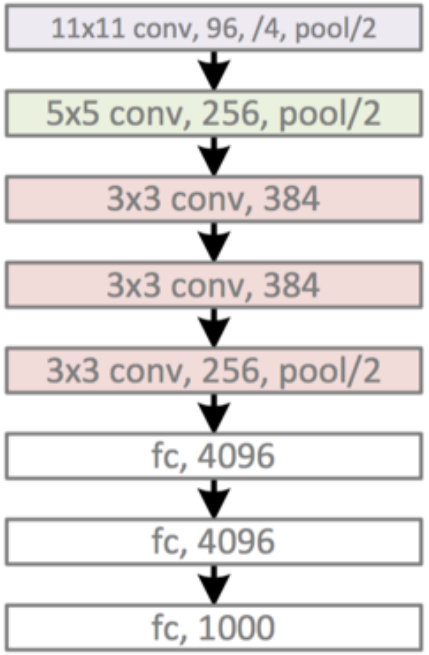


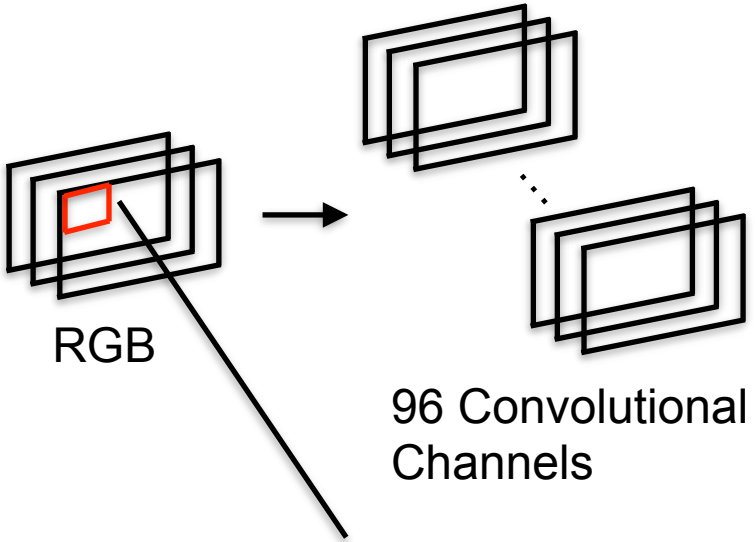


Visualizing CNNs

Understanding CNNs thru visualization:

- Can we use visualizations to understand the behavior of the hidden units in the CNN?
- What do specific hidden units in the CNN learn to recognize?



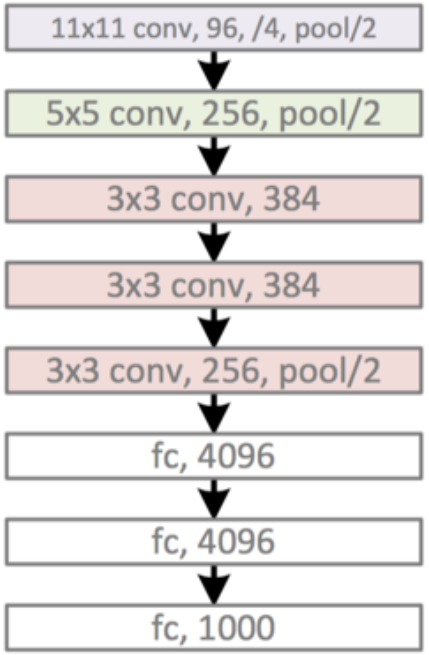


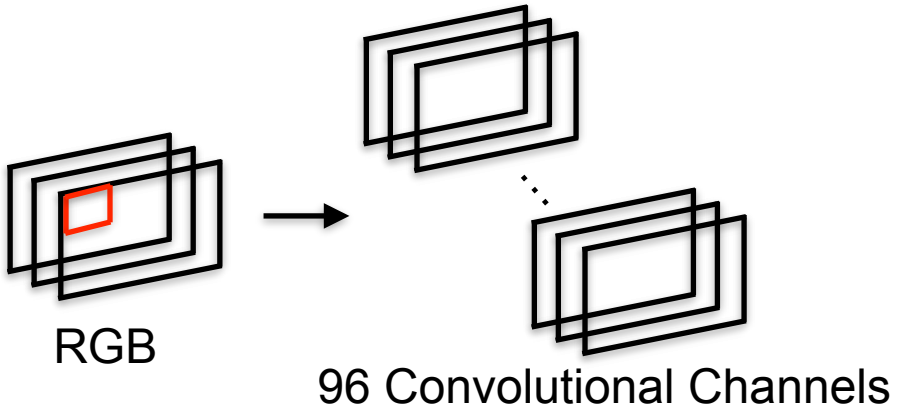
- **11x11 convolutional filter**
- **We have 3x96=288 such filters in total in conv1.**

Visualizing CNNs

Understanding CNNs thru visualization:

- Can we use visualizations to understand the behavior of the hidden units in the CNN?
- What do specific hidden units in the CNN learn to recognize?





Operation in the first layer

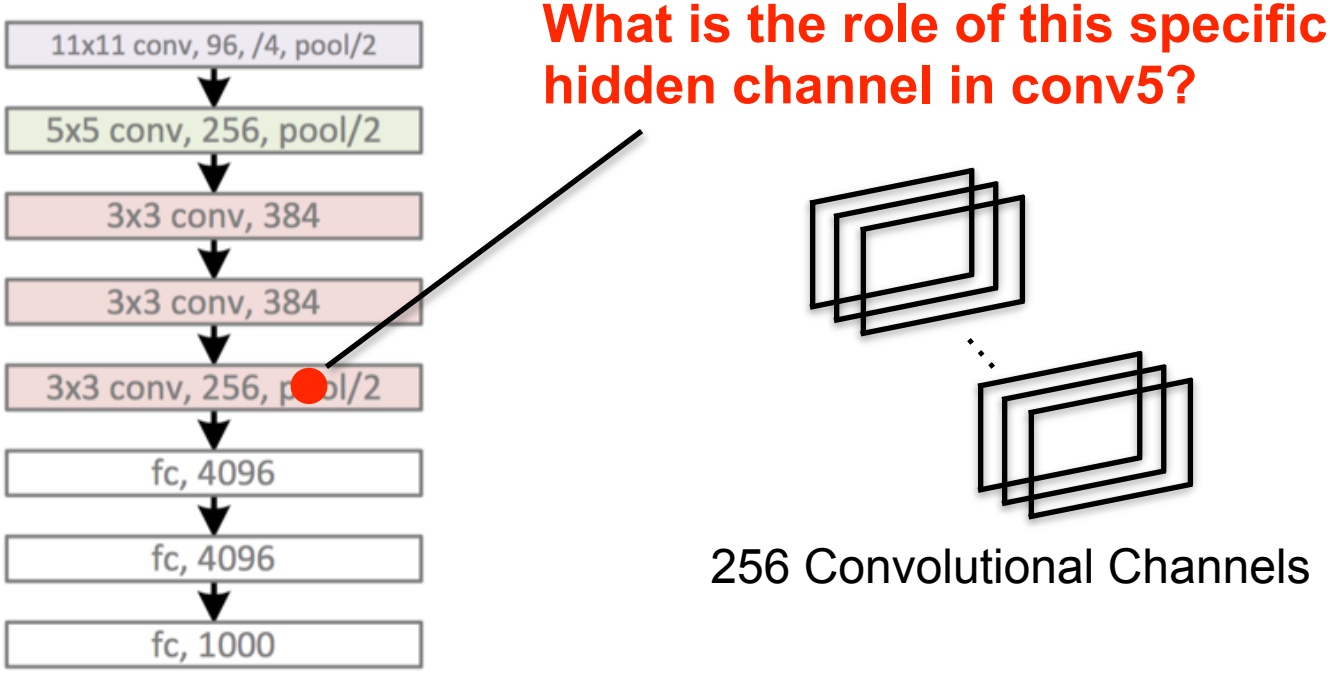
$$z(i) = \sum_{c=1}^{3} conv(x(c), w(c,i))$$

$$\forall i \in \{1 \ldots 96\}$$

Visualizing CNNs

Understanding CNNs thru visualization:

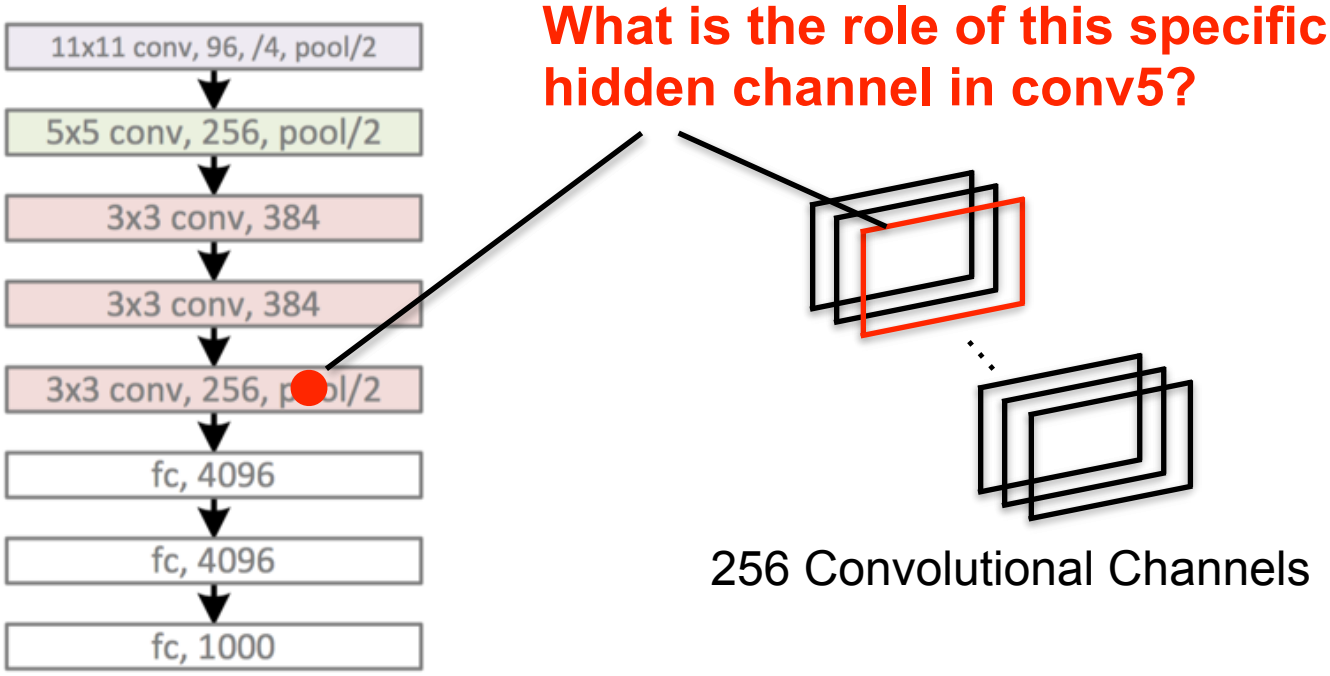
- Can we use visualizations to understand the behavior of the hidden units in the CNN?
- What do specific hidden units in the CNN learn to recognize?



Visualizing CNNs

Understanding CNNs thru visualization:

- Can we use visualizations to understand the behavior of the hidden units in the CNN?
- What do specific hidden units in the CNN learn to recognize?



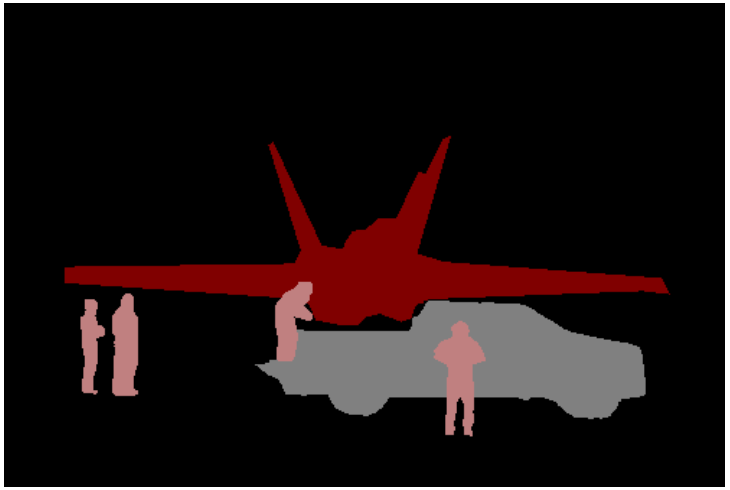
Visualizing CNNs

Understanding CNNs thru visualization:

- Let's try to understand the role of each convolutional channel through a segmentation task (Zhou et al. 2017).

Input:

Output:



Visualizing CNNs

Understanding CNNs thru visualization:

- Let's try to understand the role of each convolutional channel through a segmentation task (Zhou et al. 2017).

ADE20K
Zhou et al, CVPR'17
Pascal Context
Mottaghi et al, CVPR'14
Pascal Part
Chen et al, CVPR'14
Open-Surfaces
Bell et al, SIGGRAPH'14
Describable Textures
Cimpoi et al, CVPR'14
Colors

Large densely labeled datasets that capture various visual concepts (e.g. color, objects, places, parts, textures, etc)

Visualizing CNNs

Understanding CNNs thru visualization:

- Let's try to understand the role of each convolutional channel through a segmentation task (Zhou et al. 2017).

Every image is labeled with a per-pixel annotation of the concept.

Visualizing CNNs

Understanding CNNs thru visualization:

- Let's try to understand the role of each convolutional channel through a segmentation task (Zhou et al. 2017).

Table 1. Statistics of each label type included in the data set.

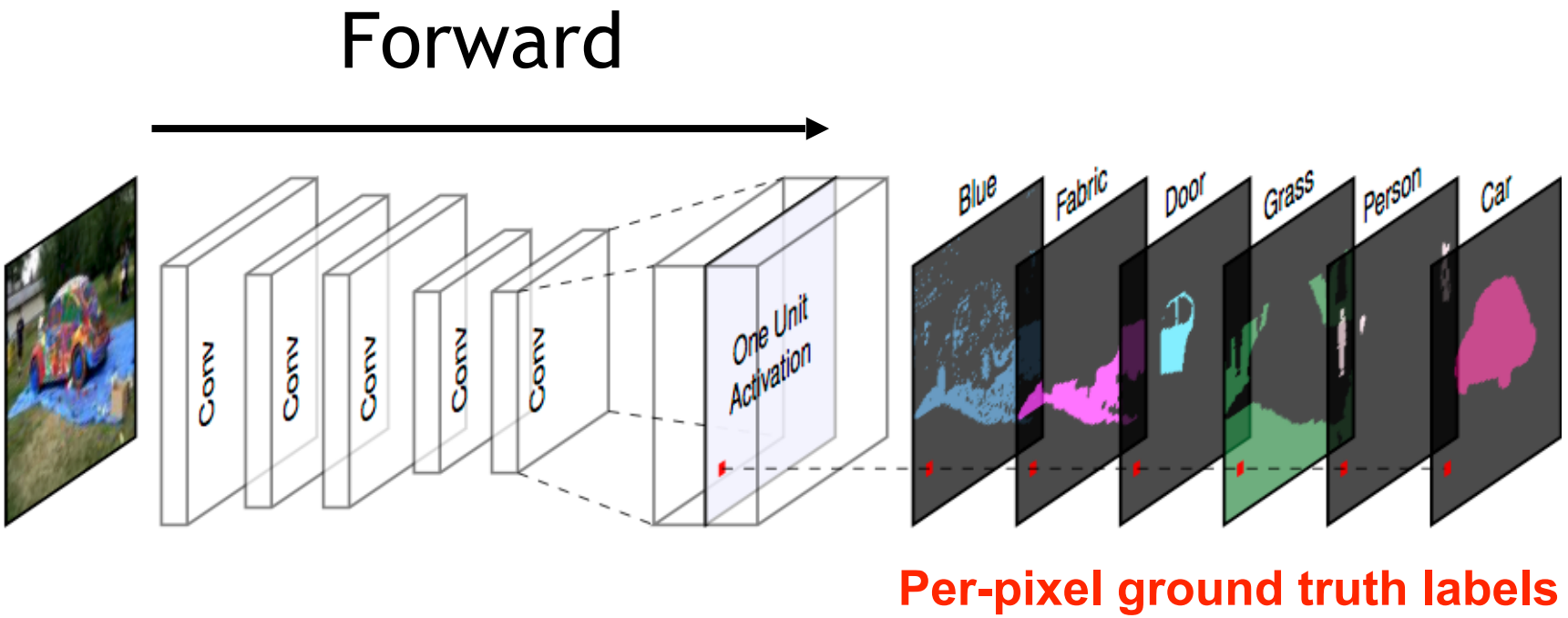
Category	Classes	Sources	Avg sample
scene	468	ADE [43]	38
object	584	ADE [43], Pascal-Context [19]	491
part	234	ADE [43], Pascal-Part [6]	854
material	32	OpenSurfaces [4]	1,703
texture	47	DTD [7]	140
color	11	Generated	59,250

Total = 63,305 images
1,197 concepts

Visualizing CNNs

Understanding CNNs thru visualization:

- Let's try to understand the role of each convolutional channel through a segmentation task (Zhou et al. 2017).



Visualizing CNNs

Understanding CNNs thru visualization:

- Feed every single image from the dataset through this network and store all the activation functions.

Visualizing CNNs

Understanding CNNs thru visualization:

- Feed every single image from the dataset through this network and store all the activation functions.

- For the activations in every convolutional channel find the units with top 0.5% highest activations (across all training images).

Visualizing CNNs

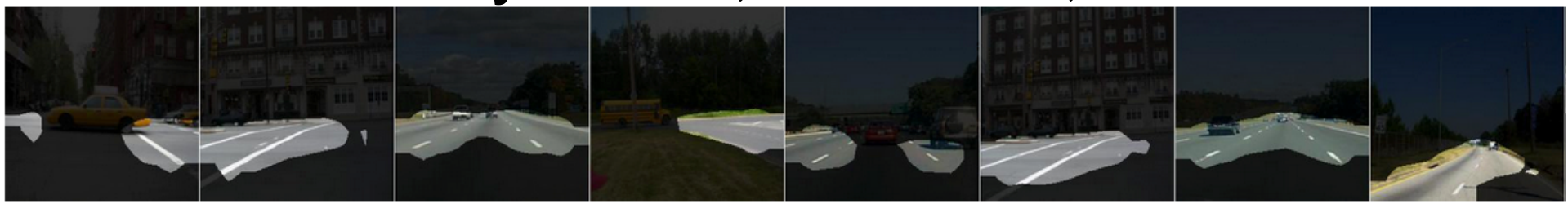
Understanding CNNs thru visualization:

- Feed every single image from the dataset through this network and store all the activation functions.

- For the activations in every convolutional channel find the units with top 0.5% highest activations (across all training images).

layer: conv5 ; **channel:** 79 ;

layer: conv5 ; **channel:** 107 ;



Visualizing CNNs

Automatically assigning the concept:

- Find the threshold associated with top 0.5% activations.
- Threshold those activations into binary 0-1 masks

layer: conv5 ; **channel:** 3 ;

Visualizing CNNs

Automatically assigning the concept:

- Find the threshold associated with top 0.5% activations.
- Threshold those activations into binary 0-1 masks

layer: conv5 ; **channel:** 3 ;

- **Activations from the same convolutional channel, obtained from different images!**

Visualizing CNNs

Automatically assigning the concept:

- Find the threshold associated with top 0.5% activations.
- Threshold those activations into binary 0-1 masks

layer: conv5 ; **channel:** 3 ;

- **Activations from the same convolutional channel, obtained from different images!**
- **These would be the masks (RGB pixels only displayed for visualization!)**
- **In practice these masks are assigned a value of 1**

Visualizing CNNs

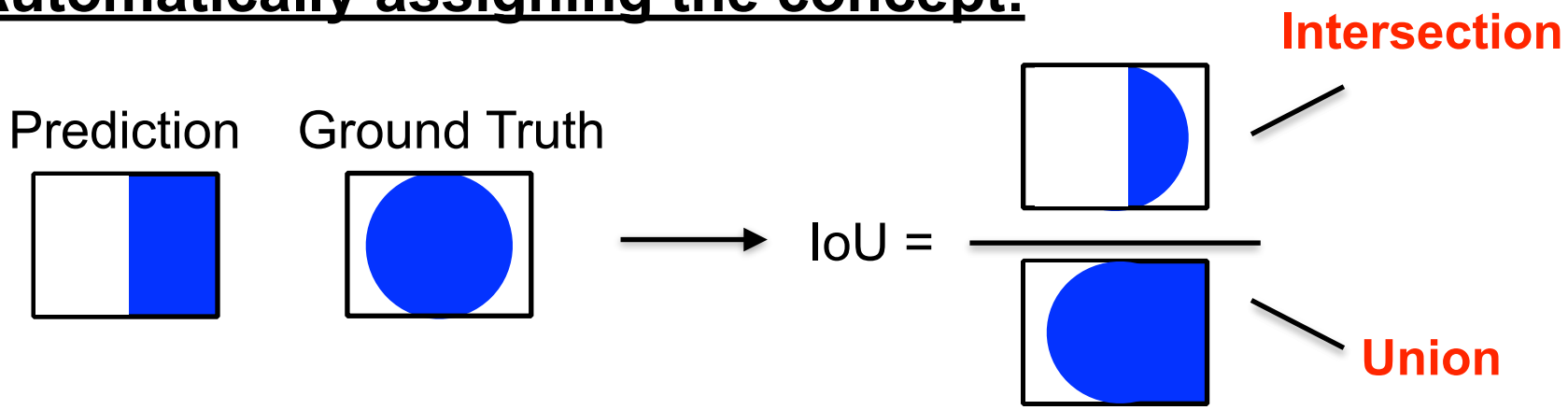
Automatically assigning the concept:

- Find the threshold associated with top 0.5% activations.
- Threshold those activations into binary 0-1 masks

- **Ground truth masks with per pixel-object annotations**

Visualizing CNNs

Automatically assigning the concept:



Top activated images for a particular convolutional channel

A lamp object

Visualizing CNNs

Automatically assigning the concept:

- Find the threshold associated with top 0.5% activations.
- Threshold those activations into binary 0-1 masks

- Compute the mean IoU (Intersection over Union) for each concept and select the concept with the highest IoU.

Visualizing CNNs

Automatically assigning the concept:

- Find the threshold associated with top 0.5% activations.
- Threshold those activations into binary 0-1 masks

- Compute the mean IoU (Intersection over Union) for each concept and select the concept with the highest IoU.
- This is the concept that that particular channel represents

Visualizing CNNs

Automatically assigning the concept:

- Find the threshold associated with top 0.5% activations.
- Threshold those activations into binary 0-1 masks

- **This specific convolutional channel represents the concept of “lamp”**

Visualizing CNNs

Analyzing the network:

- Let's look at conv5 layer of AlexNet
- Conv5 layer has 256 channels.

layer: conv5 ; **channel:** 79 ; **Object:** car ; **IoU**=0.13

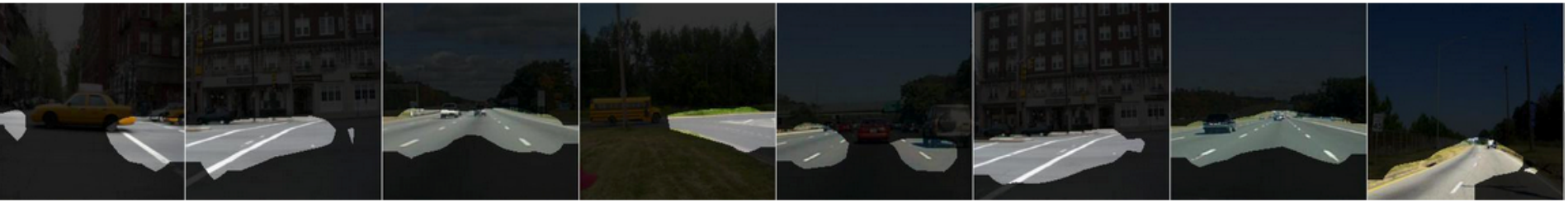
Visualizing CNNs

Analyzing the network:

- Let's look at conv5 layer of AlexNet
- Conv5 layer has 256 channels.

layer: conv5 ; **channel:** 79 ; **Object:** car ; **IoU**=0.13

layer: conv5 ; **channel:** 107 ; **Object:** road ; **IoU**=0.15



Visualizing CNNs

Analyzing the network:

- Let's look at conv5 layer of AlexNet
- Conv5 layer has 256 channels.

layer: conv5 ; **channel:** 144 ; **Object:** mountain ; **IoU**=0.13

Visualizing CNNs

Analyzing the network:

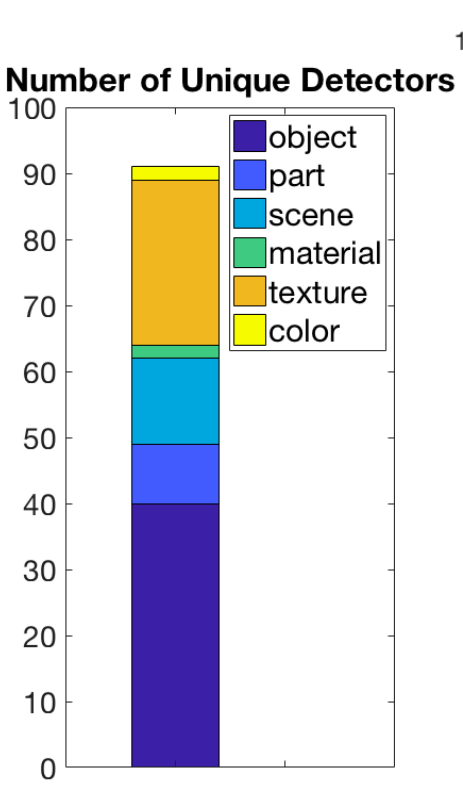
- Let's look at conv5 layer of AlexNet
- Conv5 layer has 256 channels.

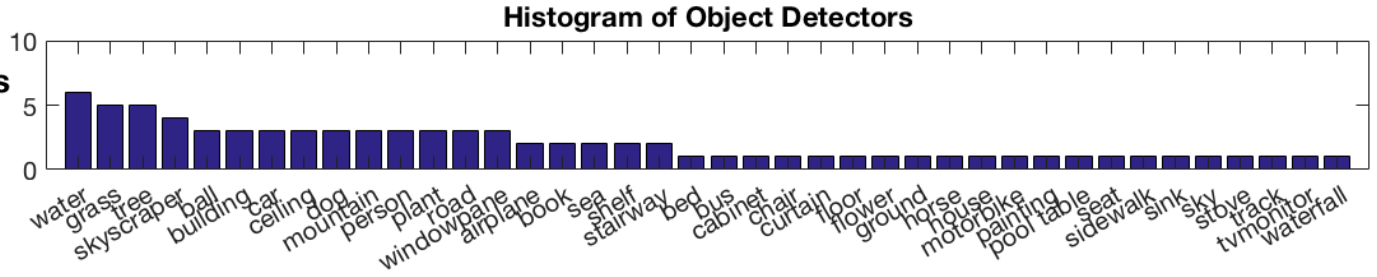
layer: conv5 ; **channel:** 144 ; **Object:** mountain ; **IoU**=0.13

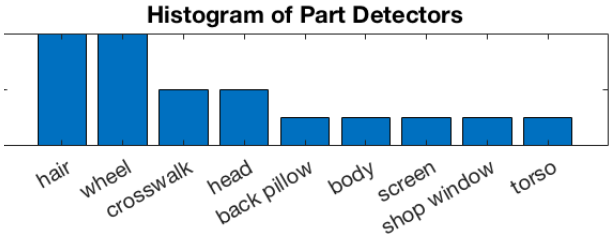
layer: conv5 ; **channel:** 200 ; **Object:** mountain ; **IoU**=0.11

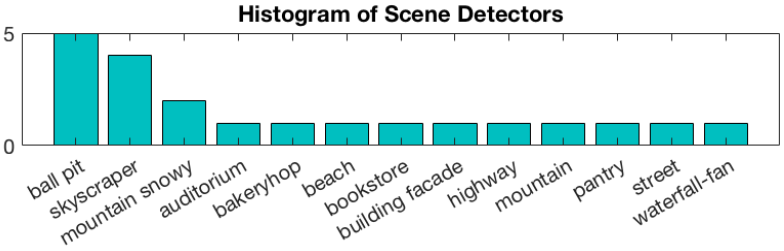
Same object classes but with different visual characteristics

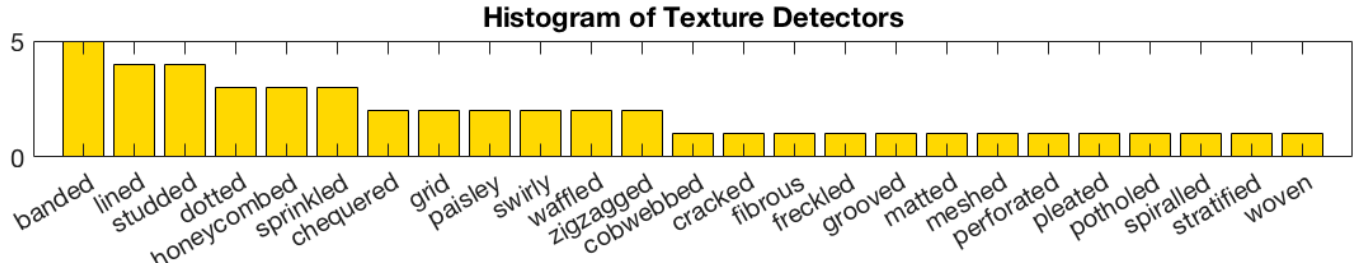
Visualizing CNNs



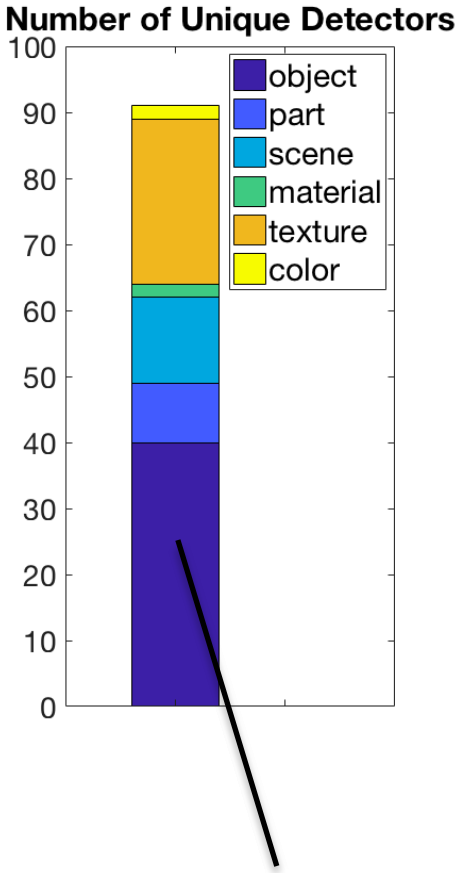


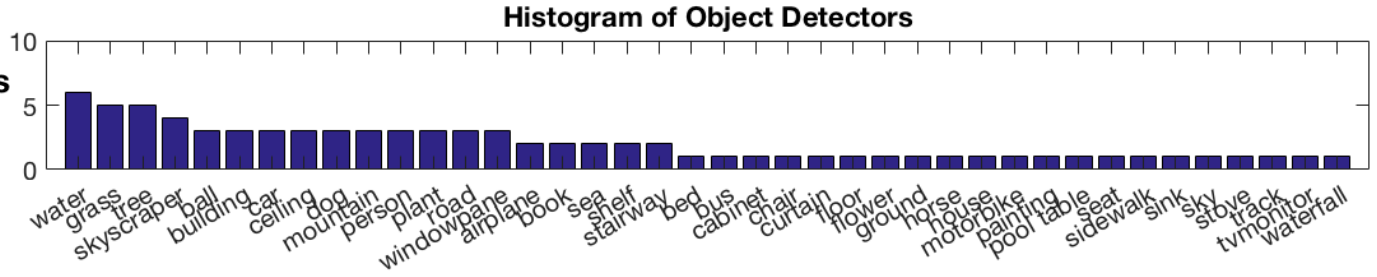


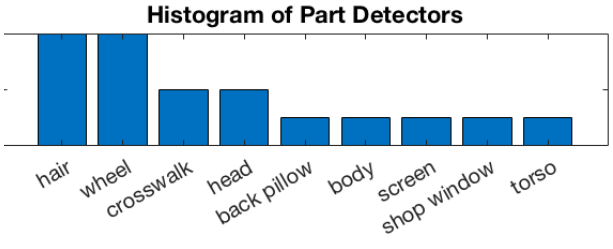


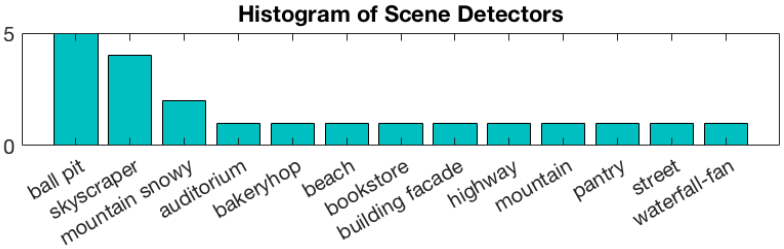


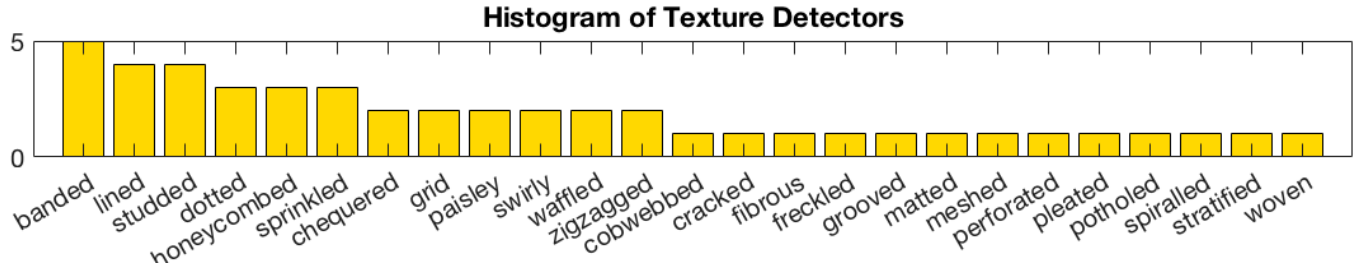
Visualizing CNNs





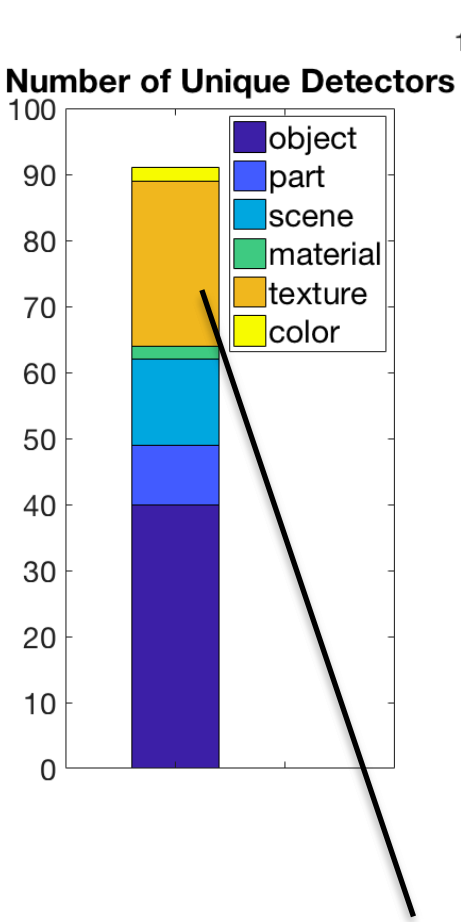


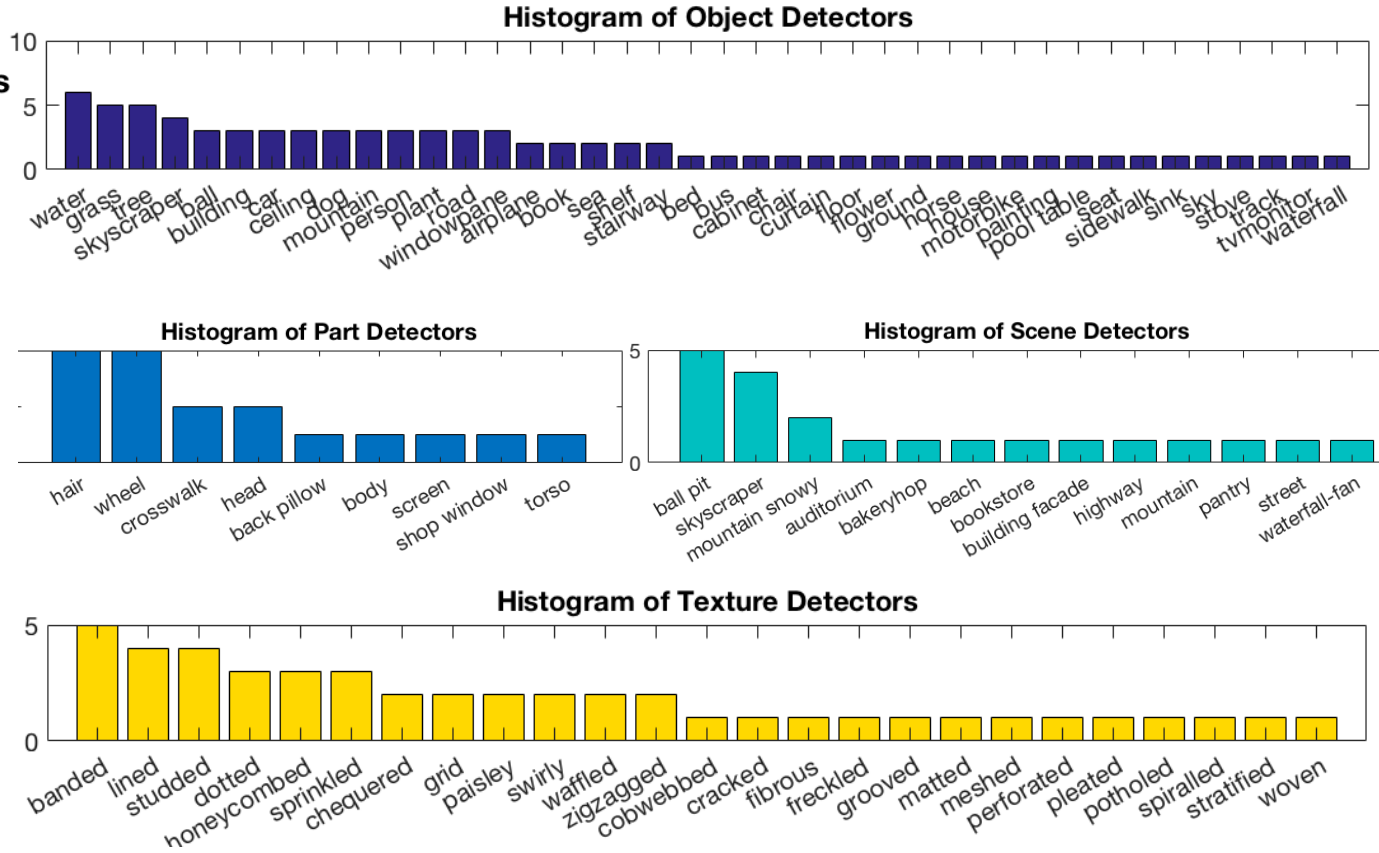




Lots of object detectors

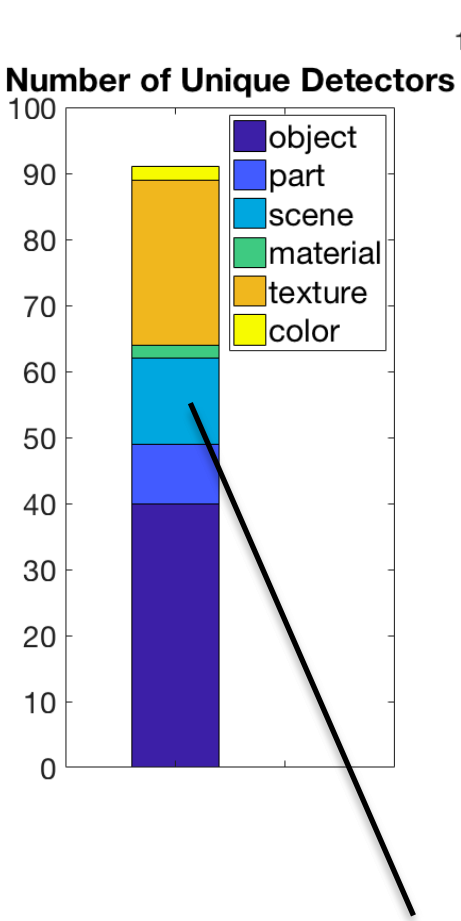
Visualizing CNNs

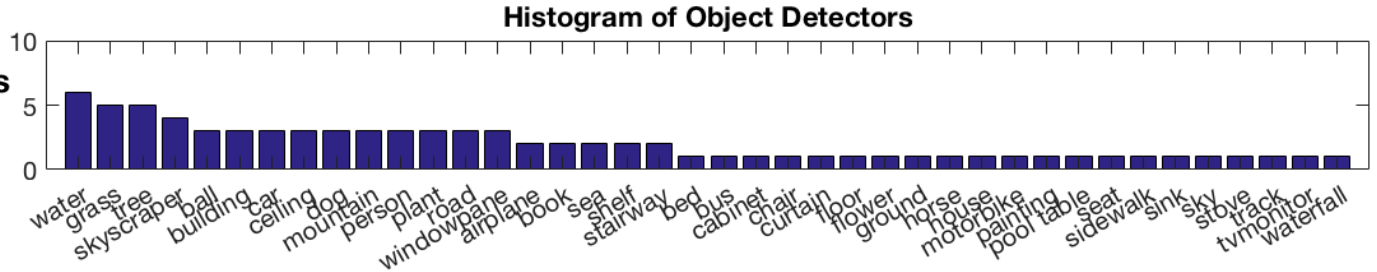


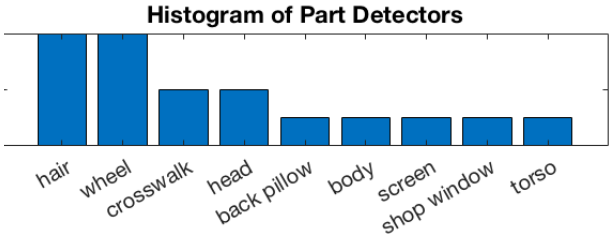


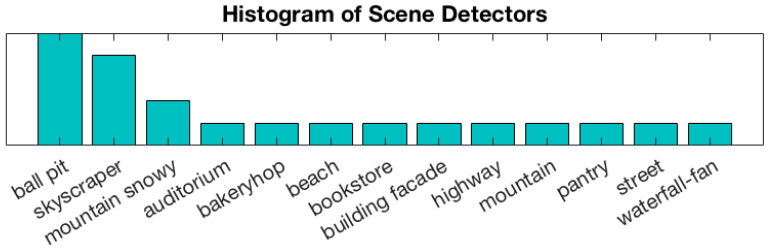
Many texture detectors

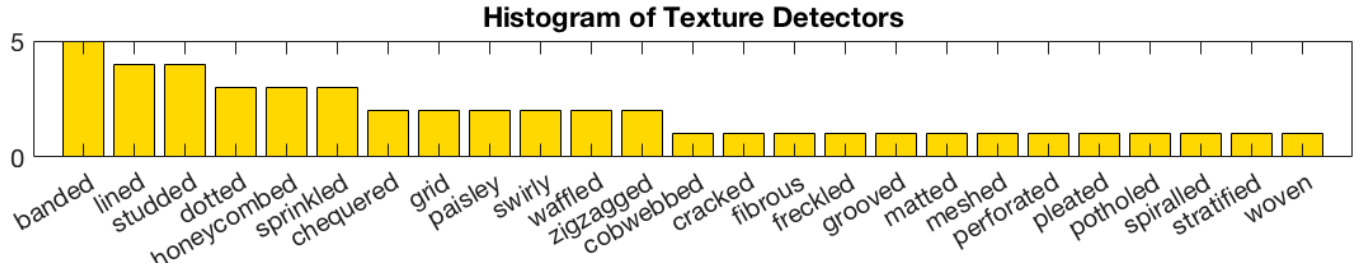
Visualizing CNNs







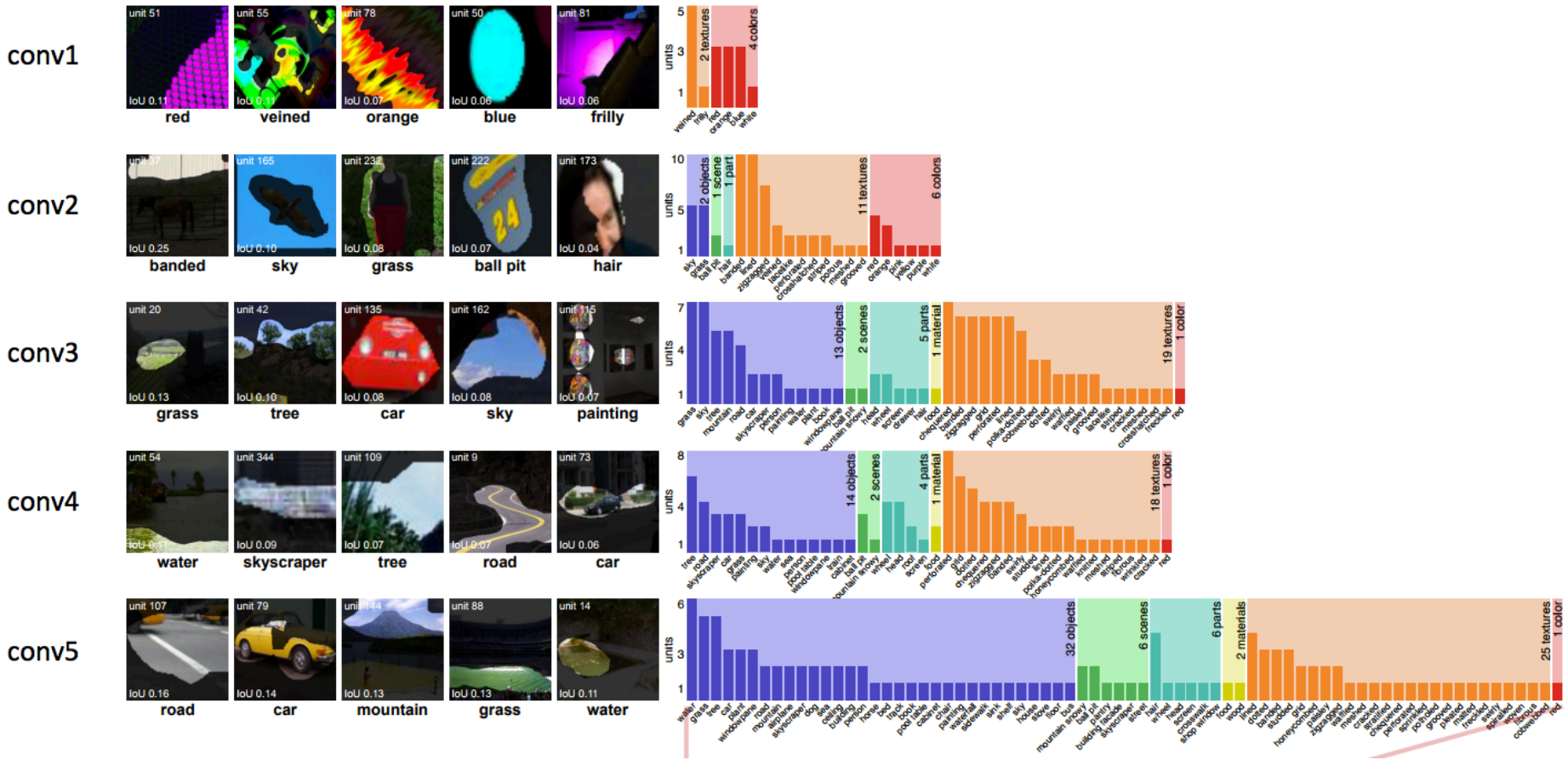




Substantial amount of scene detectors

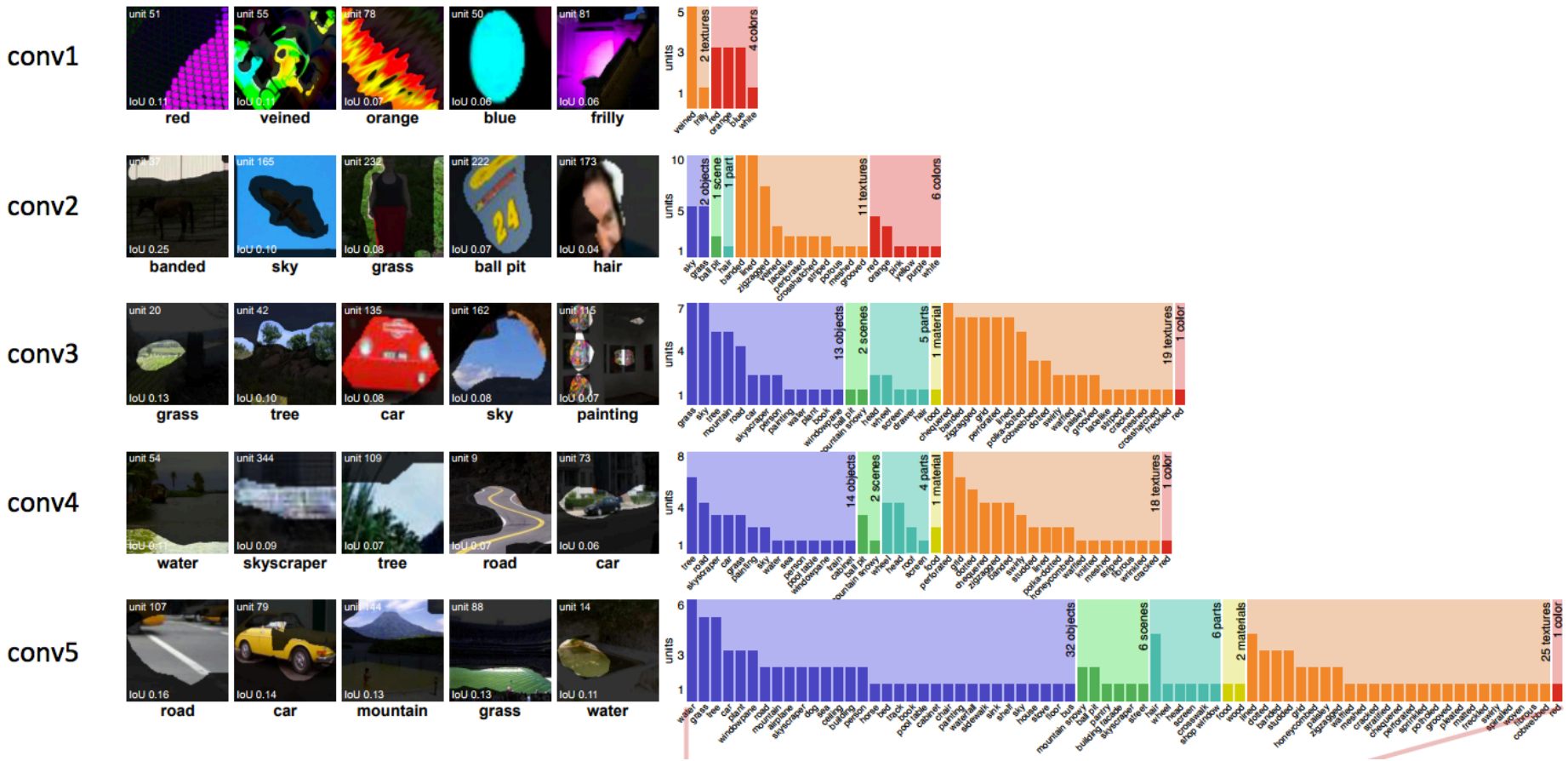
Visualizing CNNs

Analyzing CNN by layers:



Visualizing CNNs

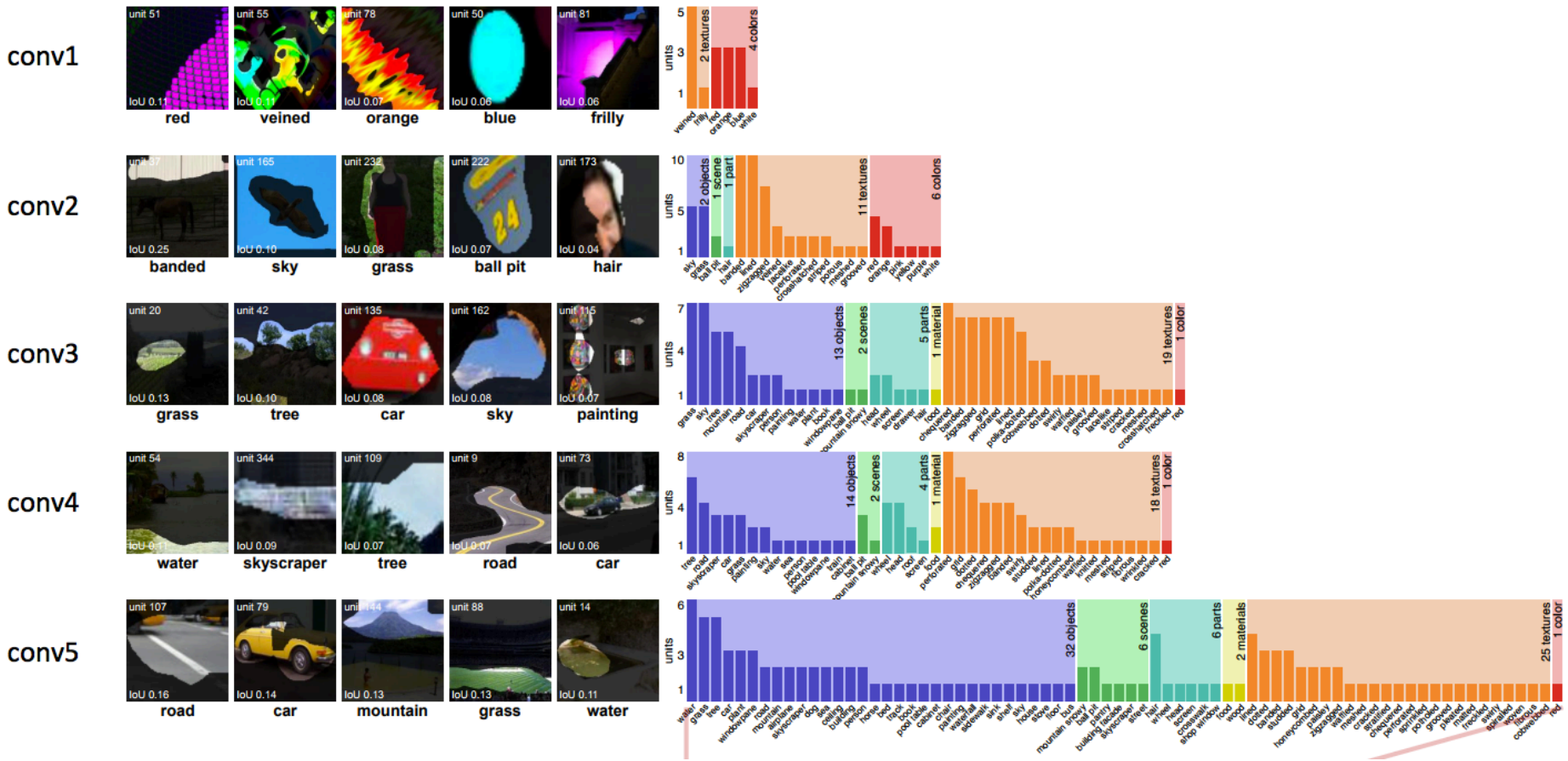
Analyzing CNN by layers:



Early layers learn color, and texture information

Visualizing CNNs

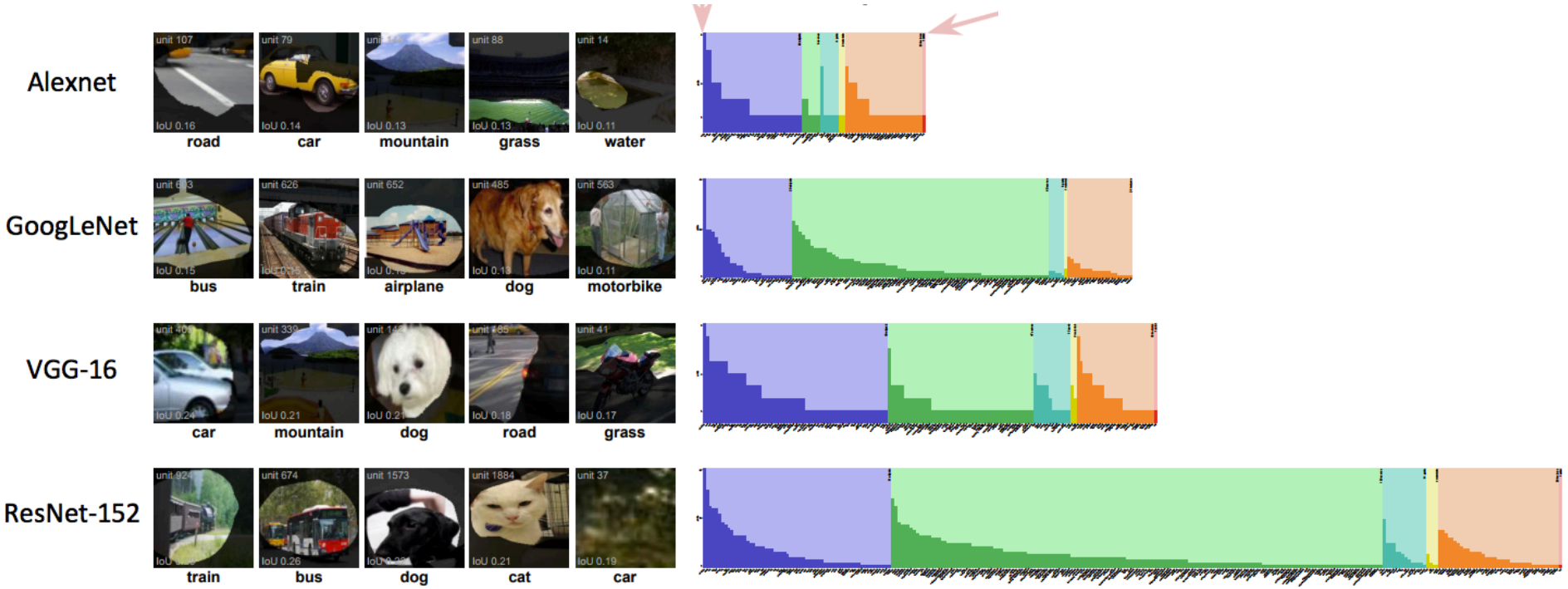
Analyzing CNN by layers:



Deeper layers focus on higher level concepts such as objects

Visualizing CNNs

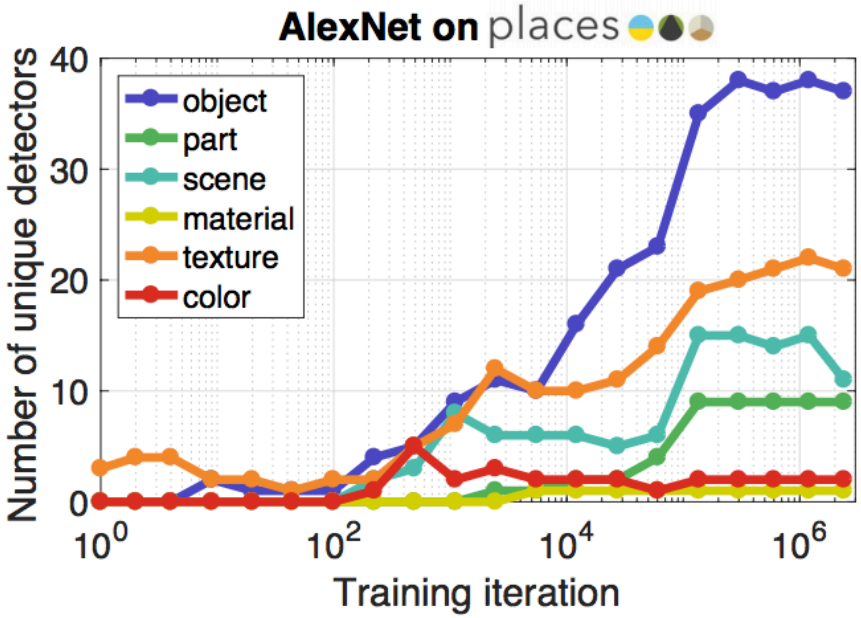
Analyzing CNNs by architectures:

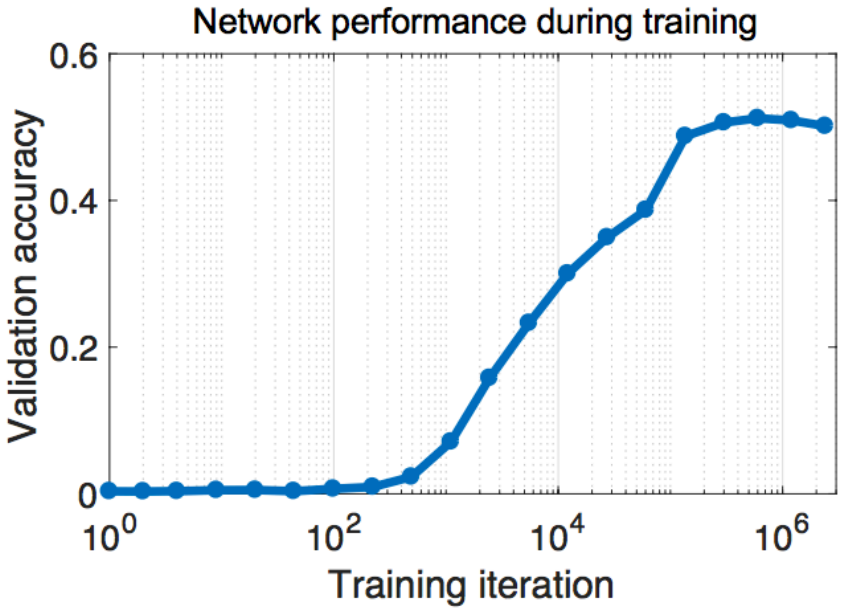


- **Deeper architectures allow us to represent more concepts**
- **Therefore, easier to achieve better performance with deeper networks**

Visualizing CNNs

Analyzing CNNs as it's being trained:

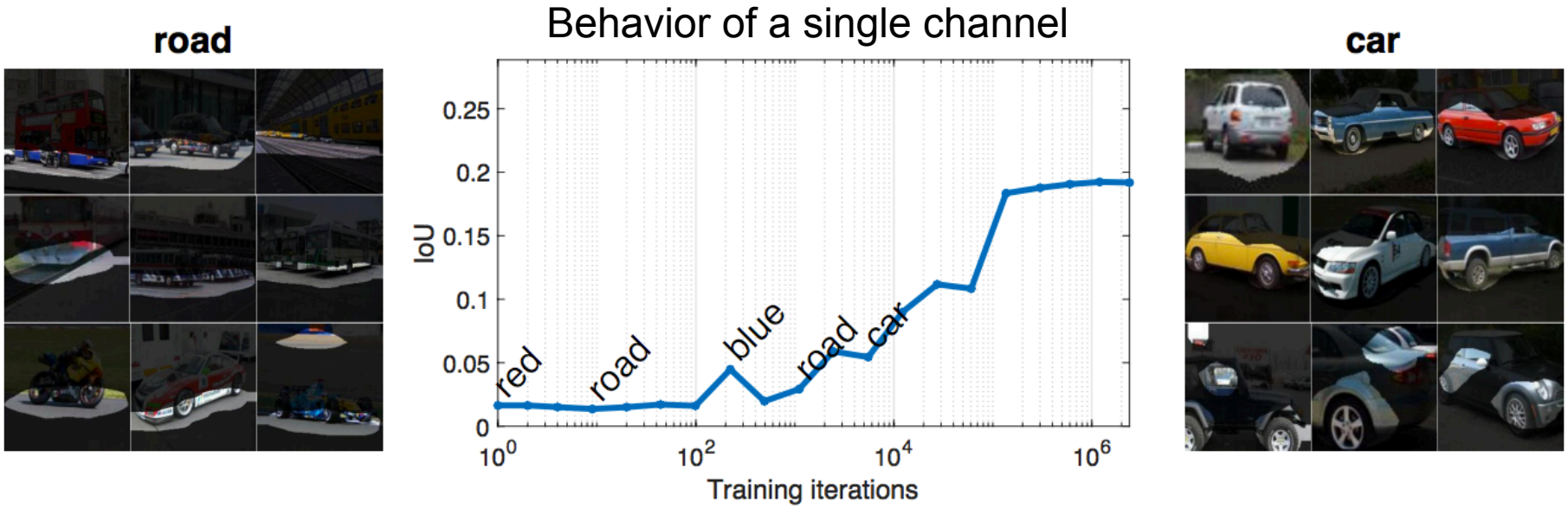




- **Higher number of unique detectors allow us to improve the performance of a CNN.**

Visualizing CNNs

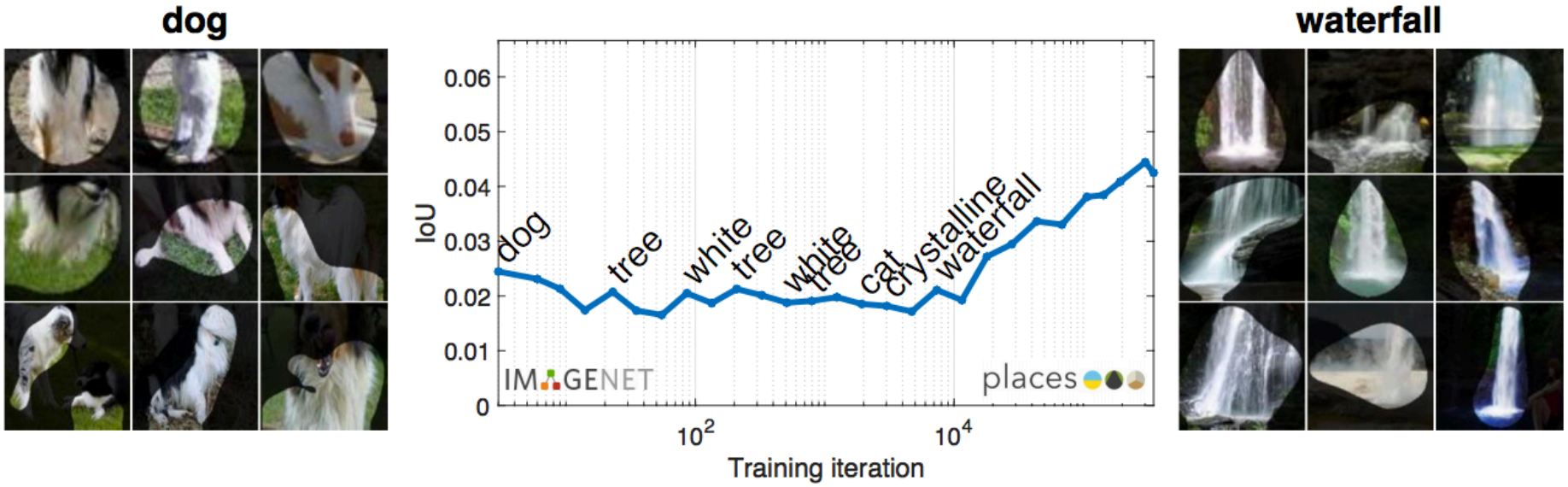
Analyzing individual channel as it's being trained:



- **Early on the channel fluctuates a lot between different concepts that are somewhat related**
- **As we train the CNN longer, the channel starts focusing on a specific concept.**

Visualizing CNNs

Analyzing individual channel as we fine-tune the CNN:



- **Initially this channel of a CNN that was trained for object detection captures a concept of dog.**
- **As we fine-tune the CNN for place detection task, this channels starts recognizing waterfals.**
- **Both waterfalls and dogs have similar low-level visual appearance.**

Visualizing CNNs

Pros:

- A very nice tool to analyze what each convolutional channel in your CNN is responsible for.
- Compact summary of the CNN and how interpretable that CNNs
- A nice way to qualitatively analyze different architecture, training strategies, etc.

Visualizing CNNs

Pros:

- A very nice tool to analyze what each convolutional channel in your CNN is responsible for.
- Compact summary of the CNN and how interpretable that CNNs
- A nice way to qualitatively analyze different architecture, training strategies, etc.

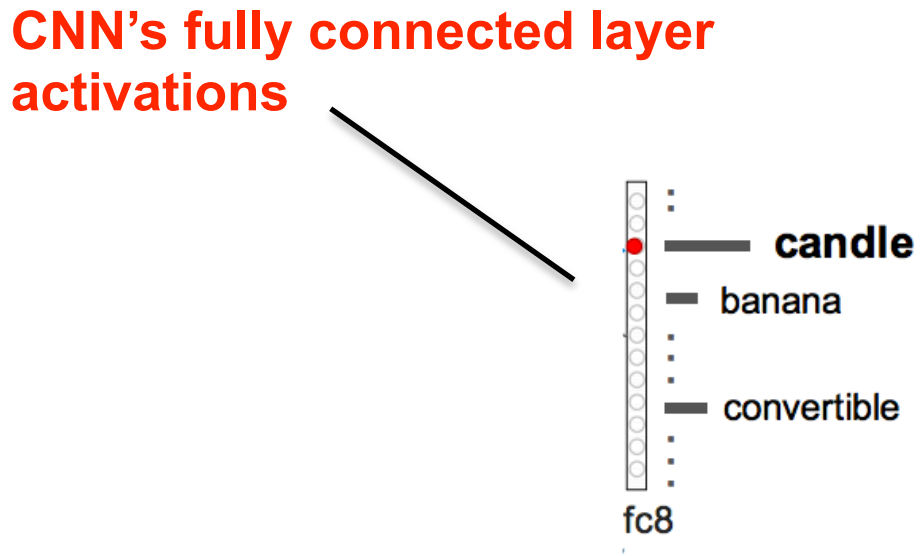
Cons:

- Requires a large densely labeled dataset.
- Can only be applied on convolutional channels (no fully connected layers, no individual convolutional units).
- If a desired concept doesn't exist in the dataset, we won't be able to analyze the CNN very well.

Visualizing CNNs

Visualizing images that maximize CNN activations (Nguyen et al. 2016):

- Suppose we want to understand what a particular neuron in the CNN fires on

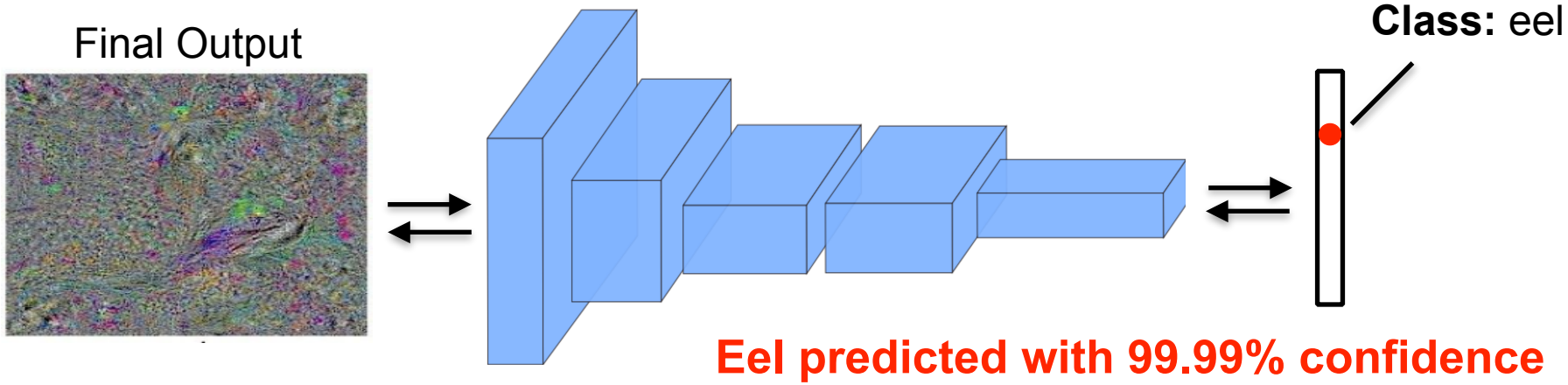


Visualizing CNNs

Visualizing images that maximize CNN activations (Nguyen et al. 2016):

- Suppose we want to understand what a particular neuron in the CNN fires on

$$L = -\sum_{k=1}^{K} 1\{y = k\} \log \hat{y}_k$$

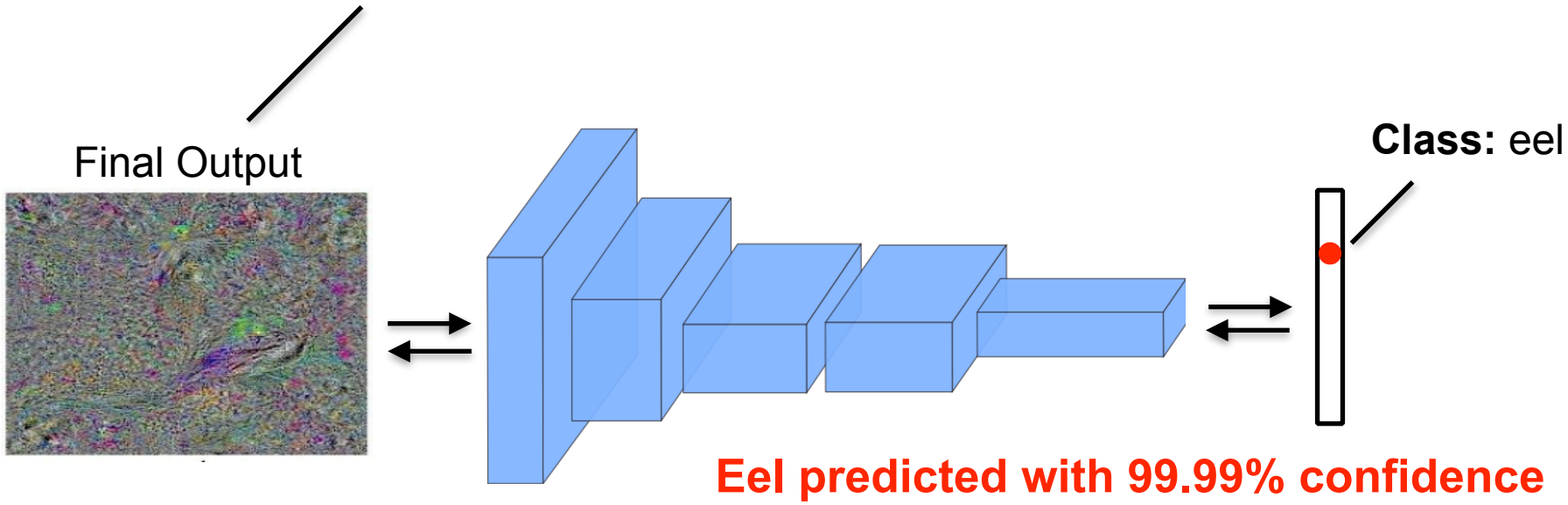


Visualizing CNNs

Visualizing images that maximize CNN activations (Nguyen et al. 2016):

- Suppose we want to understand what a particular neuron in the CNN fires on

  - **Images look very unrealistic.**
  - **Difficult to understand what is depicted in the images.**



Visualizing CNNs

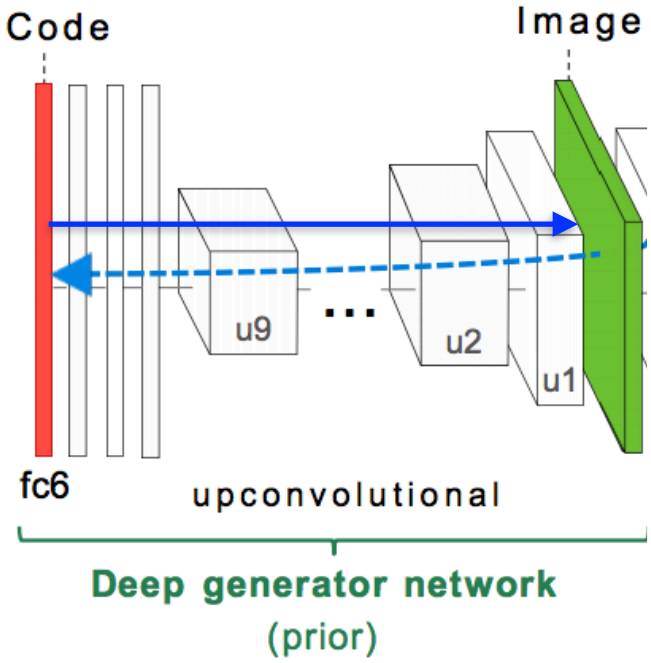
Visualizing images that maximize CNN activations (Nguyen et al. 2016):

- The idea is first to train a network that synthesizes a realistic image from a certain feature vector in the CNN

Visualizing CNNs

Visualizing images that maximize CNN activations (Nguyen et al. 2016):

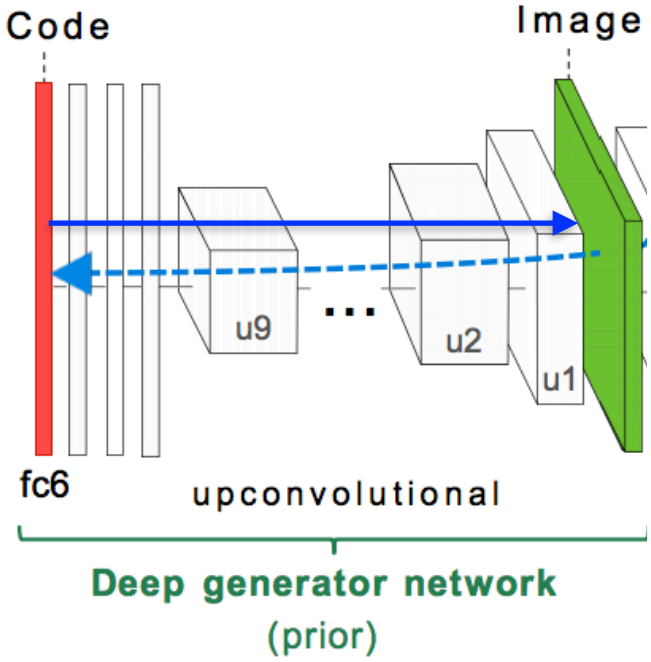
- The idea is first to train a network that synthesizes a realistic image from a certain feature vector in the CNN



Visualizing CNNs

Visualizing images that maximize CNN activations (Nguyen et al. 2016):

- The idea is first to train a network that synthesizes a realistic image from a certain feature vector in the CNN

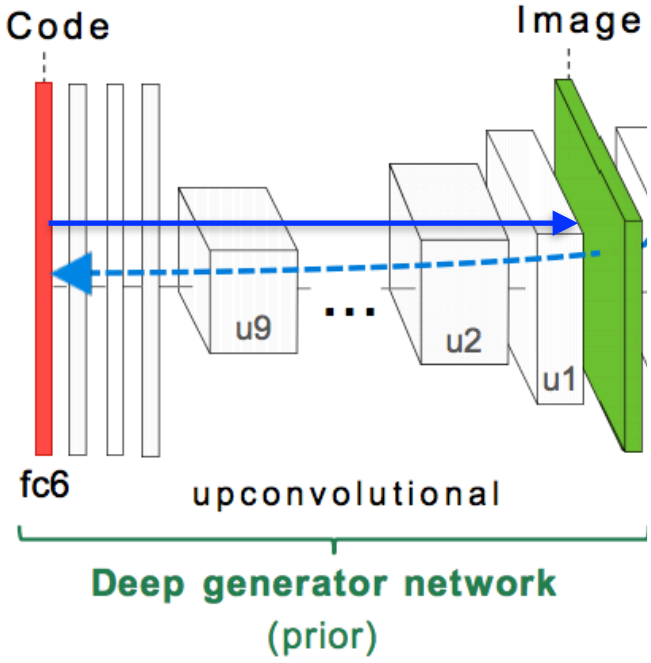


- **We will talk about these types of networks more in the future.**
- **For now just assume that they take fc6 feature as an input and are optimized to output the RGB values of an image.**

Visualizing CNNs

Visualizing images that maximize CNN activations (Nguyen et al. 2016):

- The idea is first to train a network that synthesizes a realistic image from a certain feature vector in the CNN



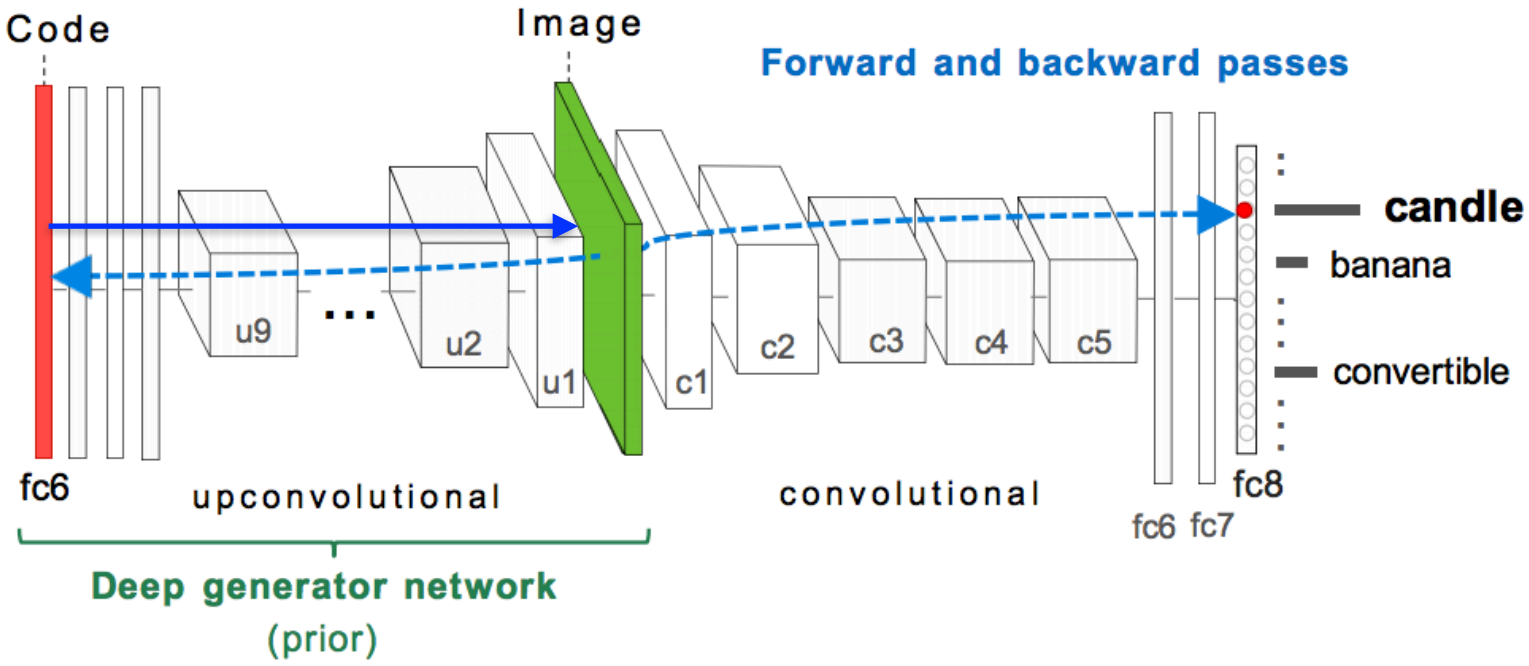
- **We will talk about these types of networks more in the future.**
- **For now just assume that they take fc6 feature as an input and are optimized to output the RGB values of an image.**

$$L = \sum_{i=1}^{n} \sum_{c=1}^{3} (\hat{y}_i(c) - y_i(c))^2$$

Visualizing CNNs

Visualizing images that maximize CNN activations (Nguyen et al. 2016):

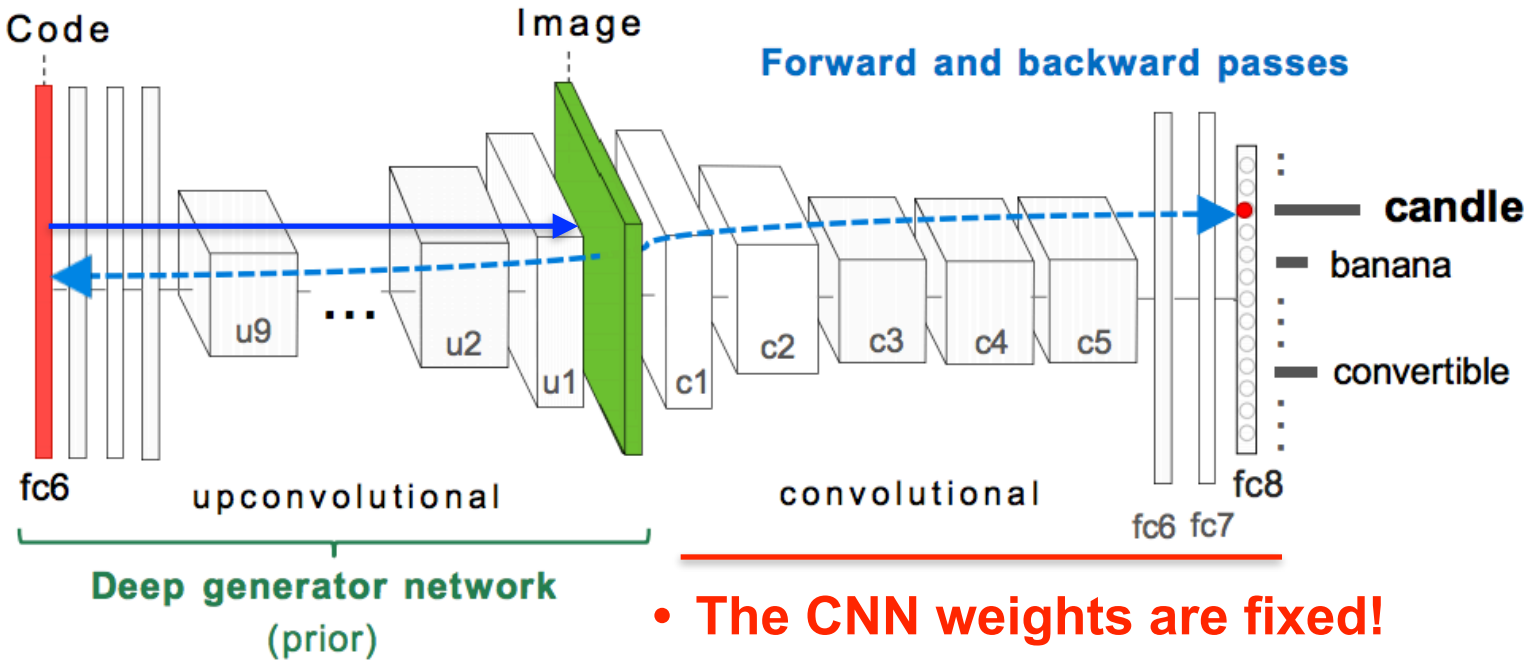
- Now we can attach a pretrained CNN that takes a generated image as an input and feeds it forward.



Visualizing CNNs

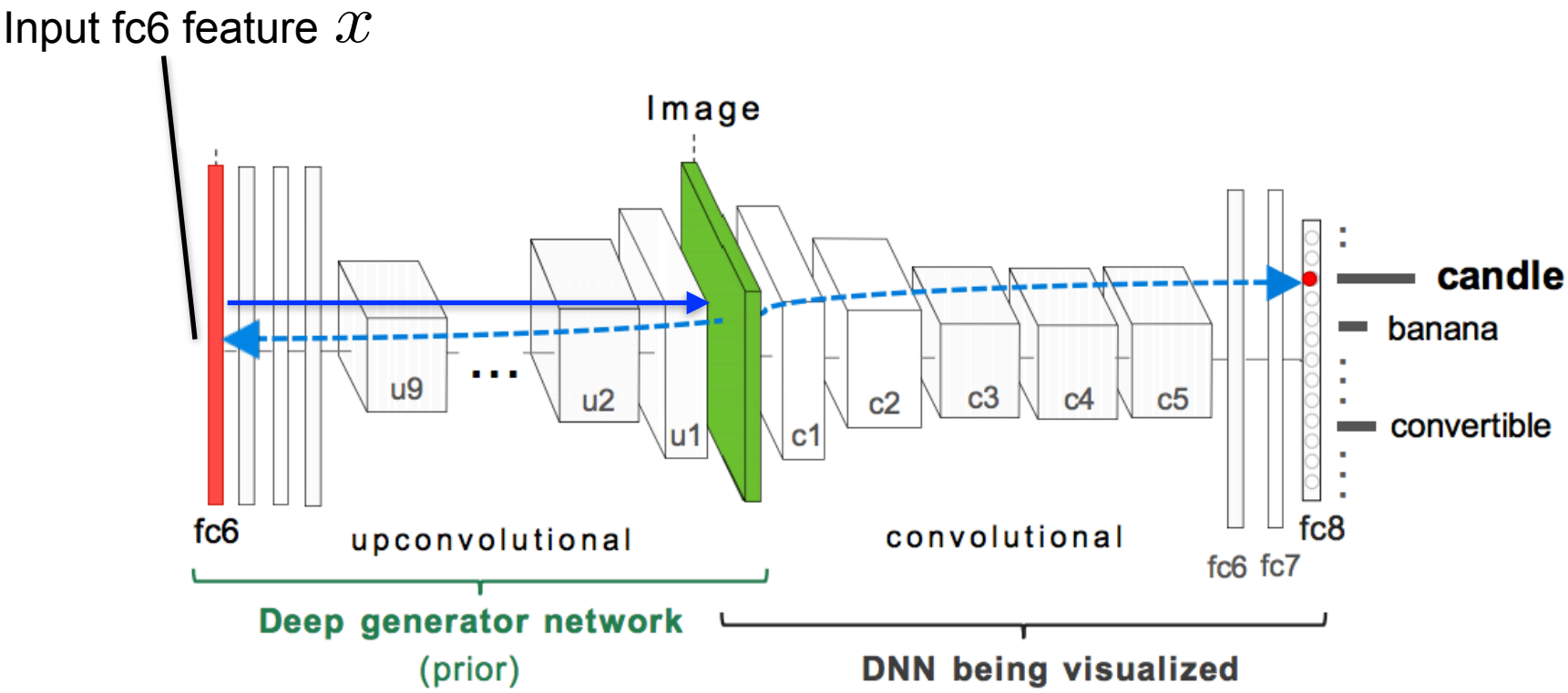
Visualizing images that maximize CNN activations (Nguyen et al. 2016):

- Now we can attach a pretrained CNN that takes a generated image as an input and feeds it forward.

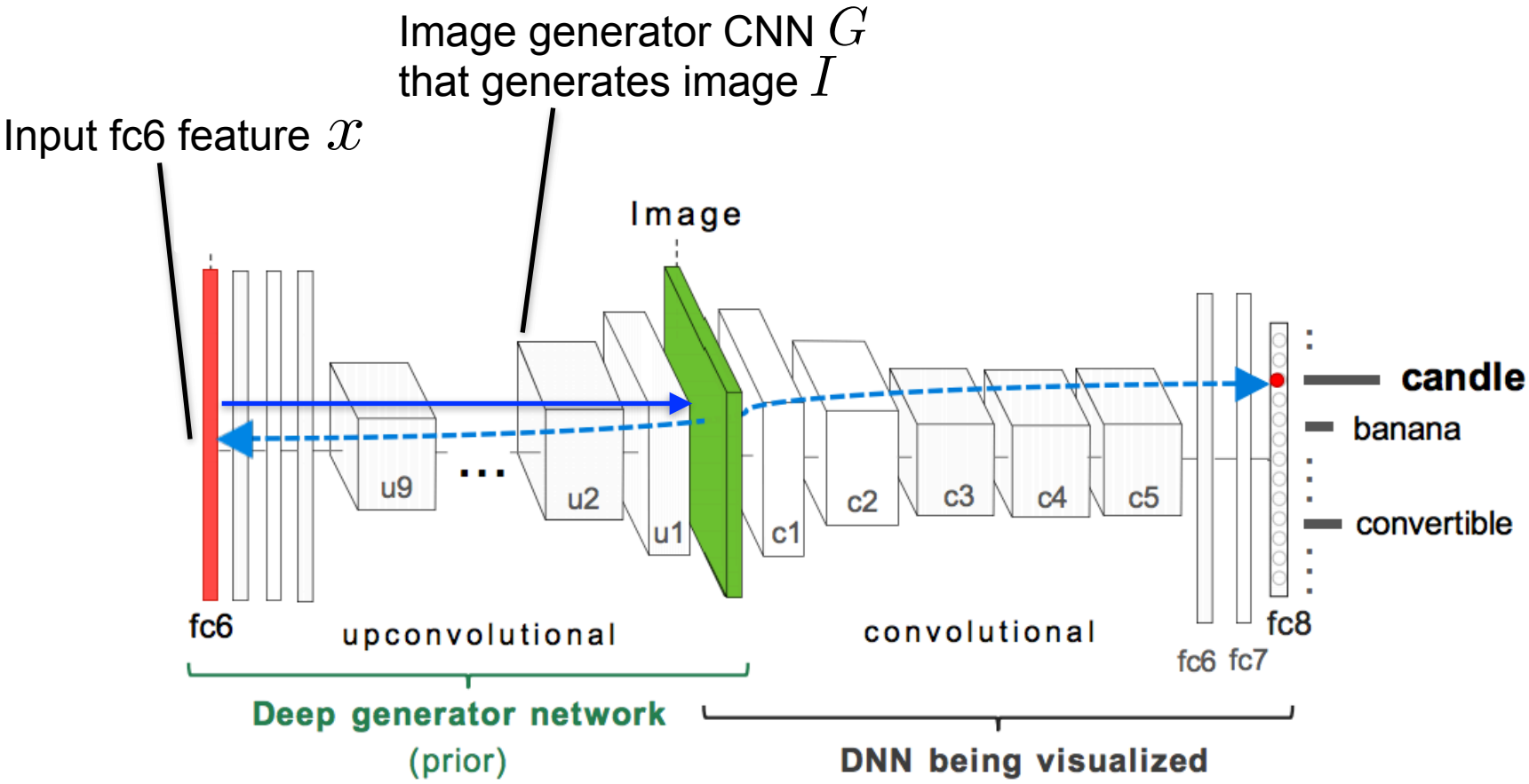


- **The CNN weights are fixed!**
- **We want to visualize what certain neurons in this CNN represent.**

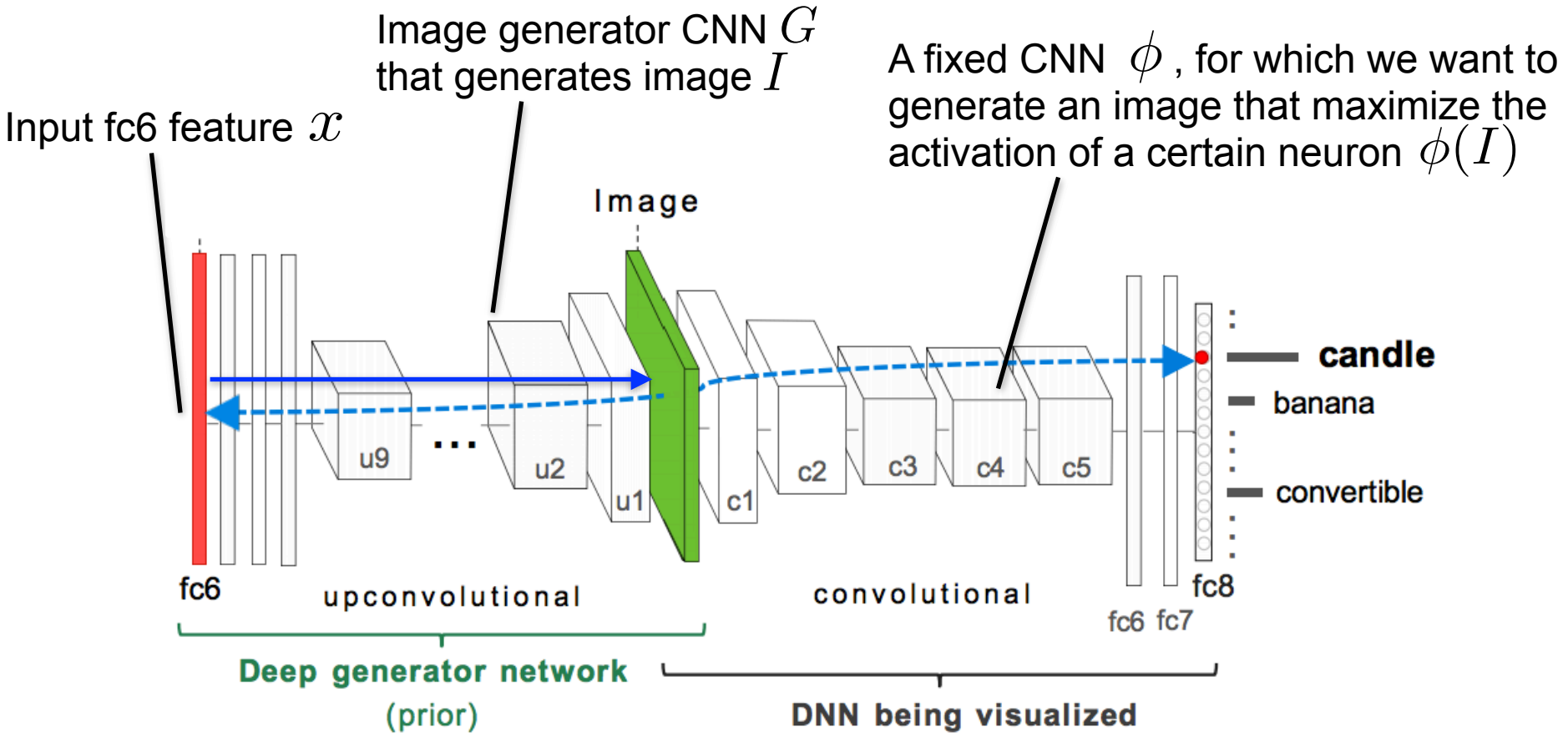
Visualizing CNNs



Visualizing CNNs

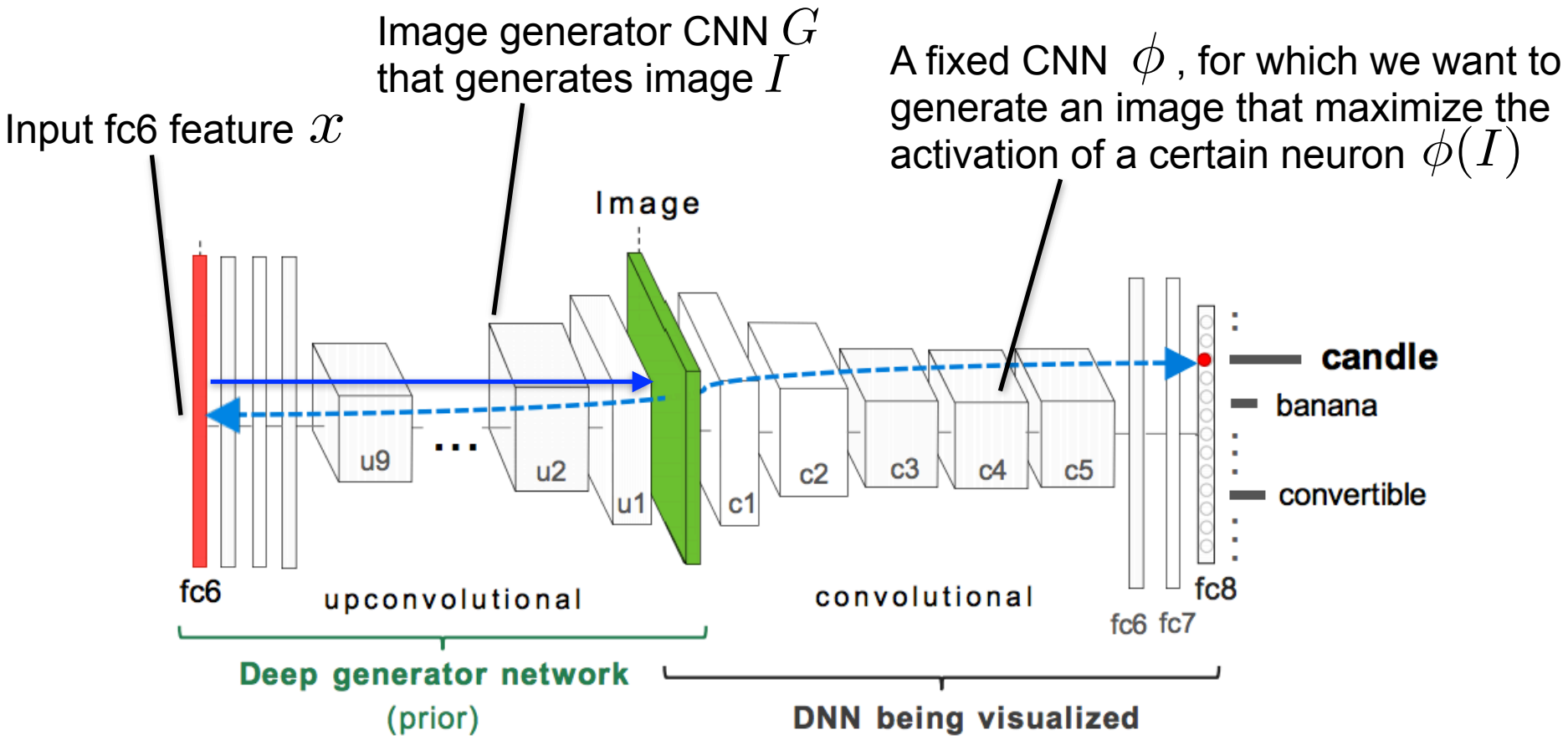


Visualizing CNNs



Visualizing CNNs

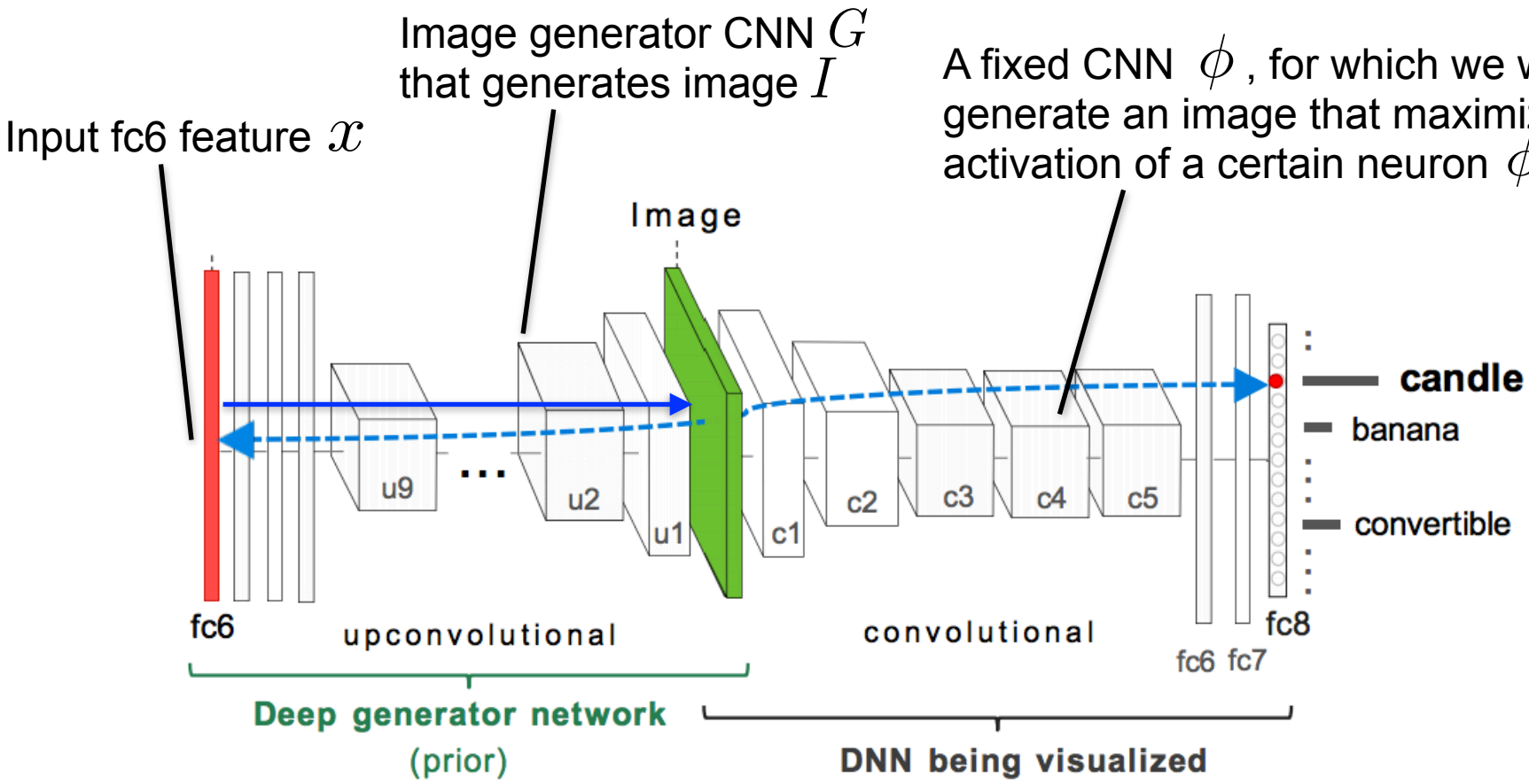
- **The goal is to find a feature x that is used to generate an image that maximizes a certain neuron activation in the CNN.**



Visualizing CNNs

Loss function: $L = -\phi_j(G(x))$

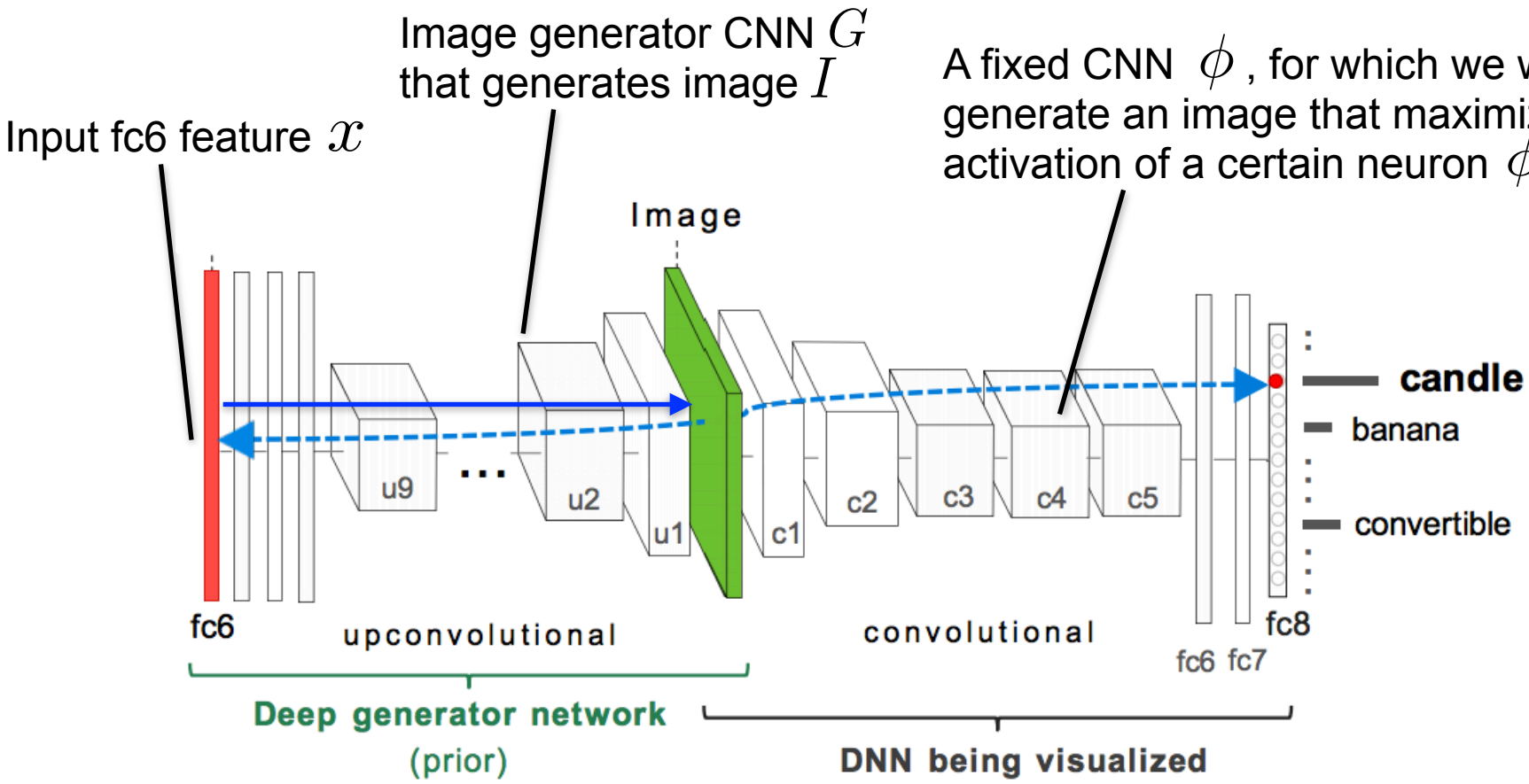
- **Image generated from the feature x**



Visualizing CNNs

Loss function: $L = -\phi_j(G(x))$

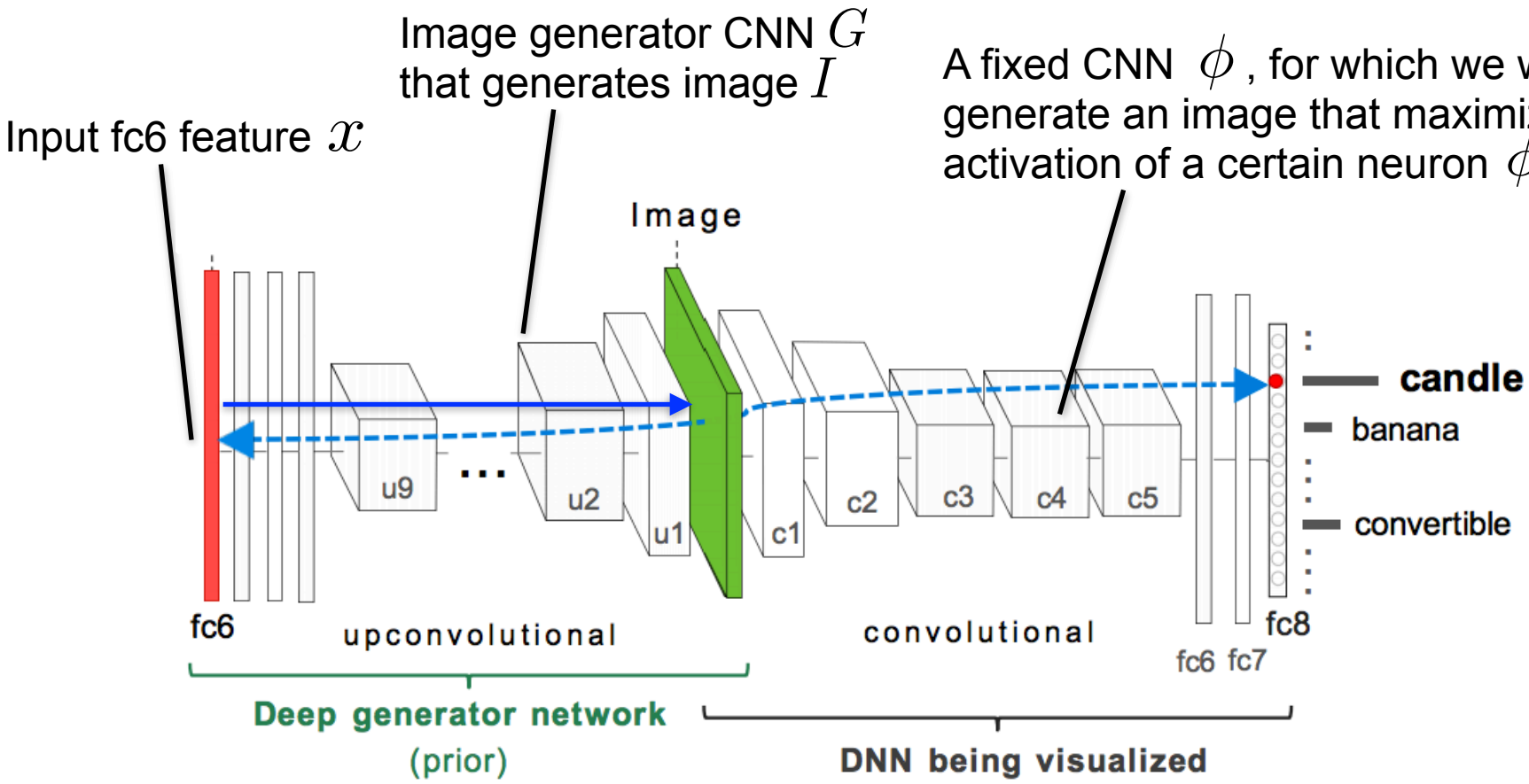
- **A neuron activation generated from the image that was generated from feature x**



Visualizing CNNs

Loss function: $L = -\phi_j(G(x))$

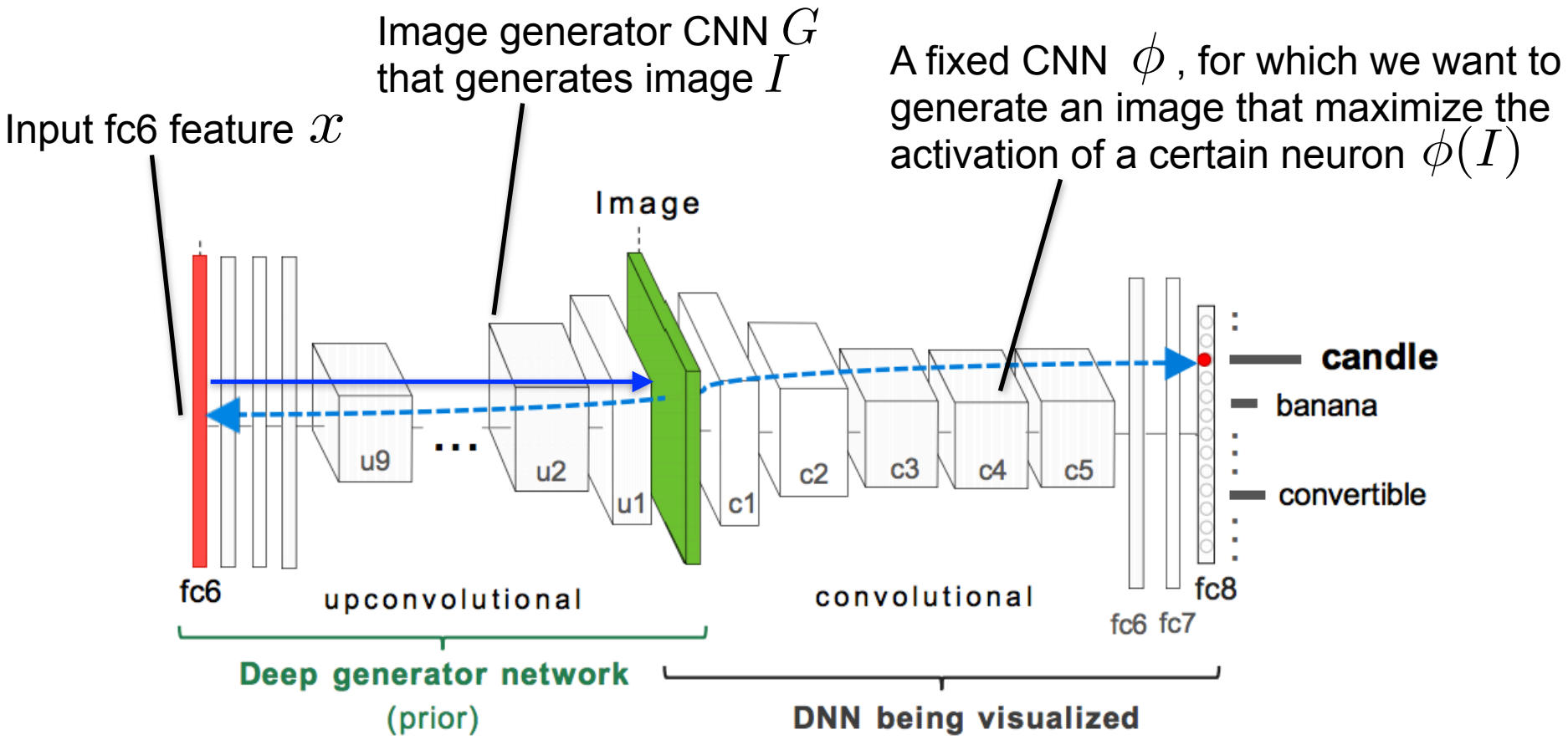
- **We want this value to be as large as possible**



Visualizing CNNs

Loss function: $L = -\phi_j(G(x))$

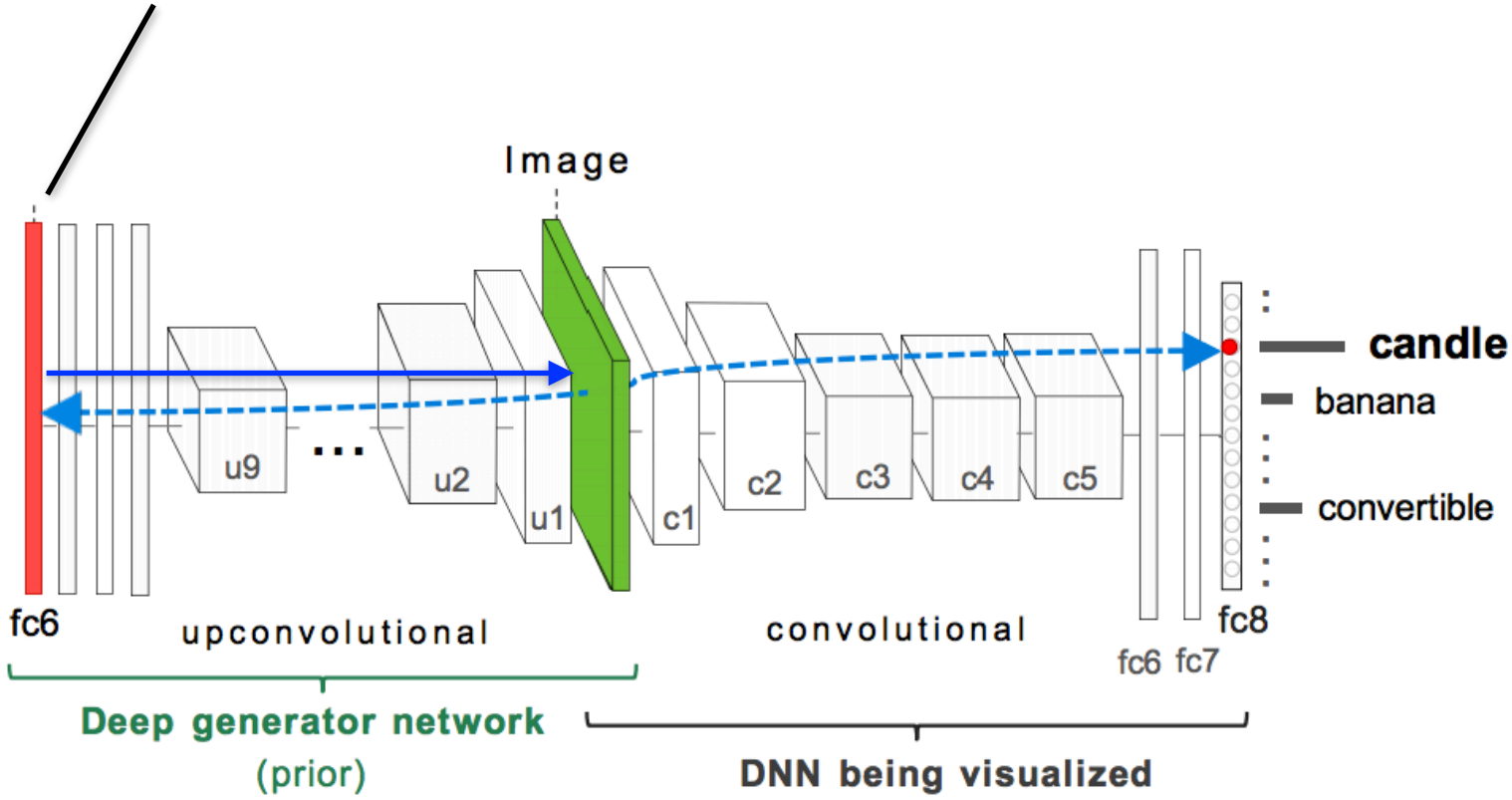
- **We can minimize this function using standard backpropagation algorithm.**



Visualizing CNNs

Loss function: $L = -\phi_j(G(x))$

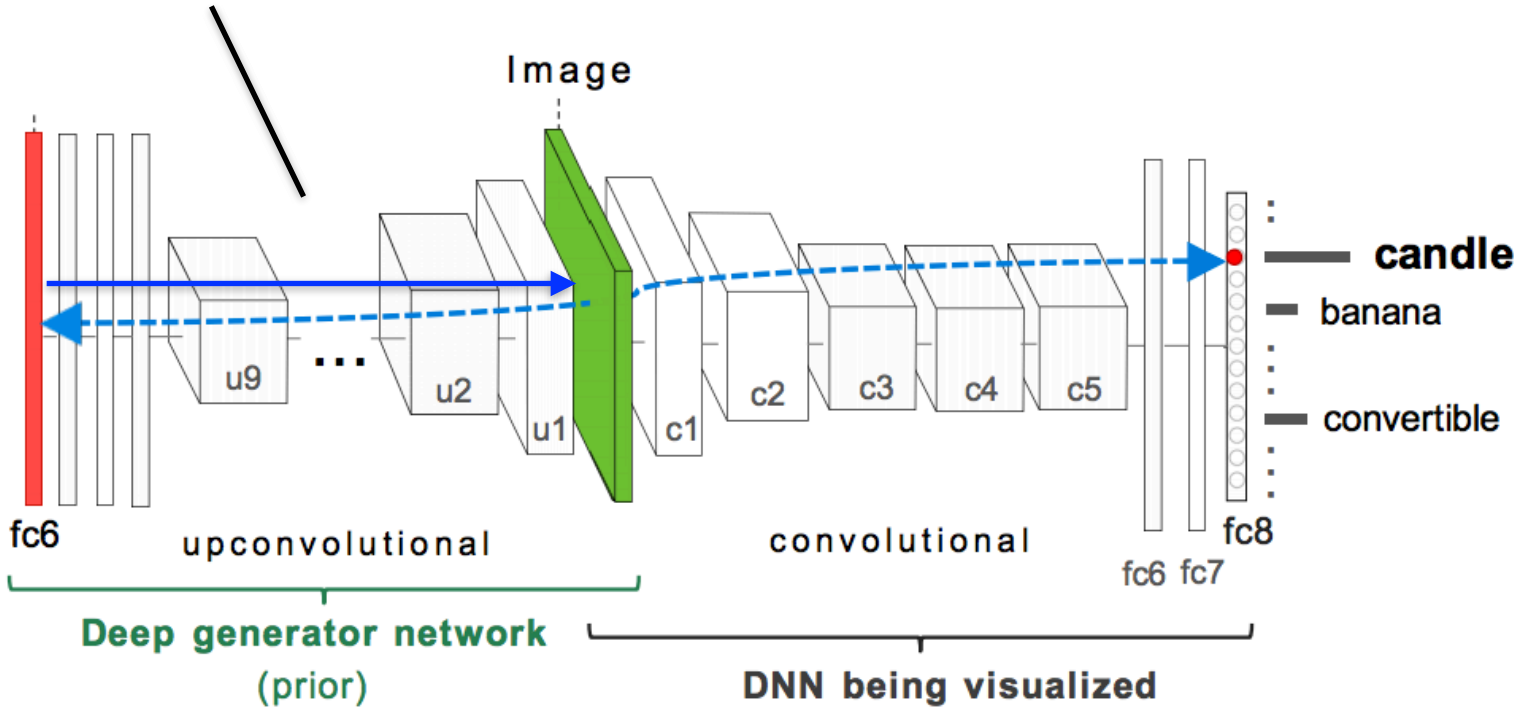
1. Randomly initialize the fc6 feature x



Visualizing CNNs

Loss function: $L = -\phi_j(G(x))$

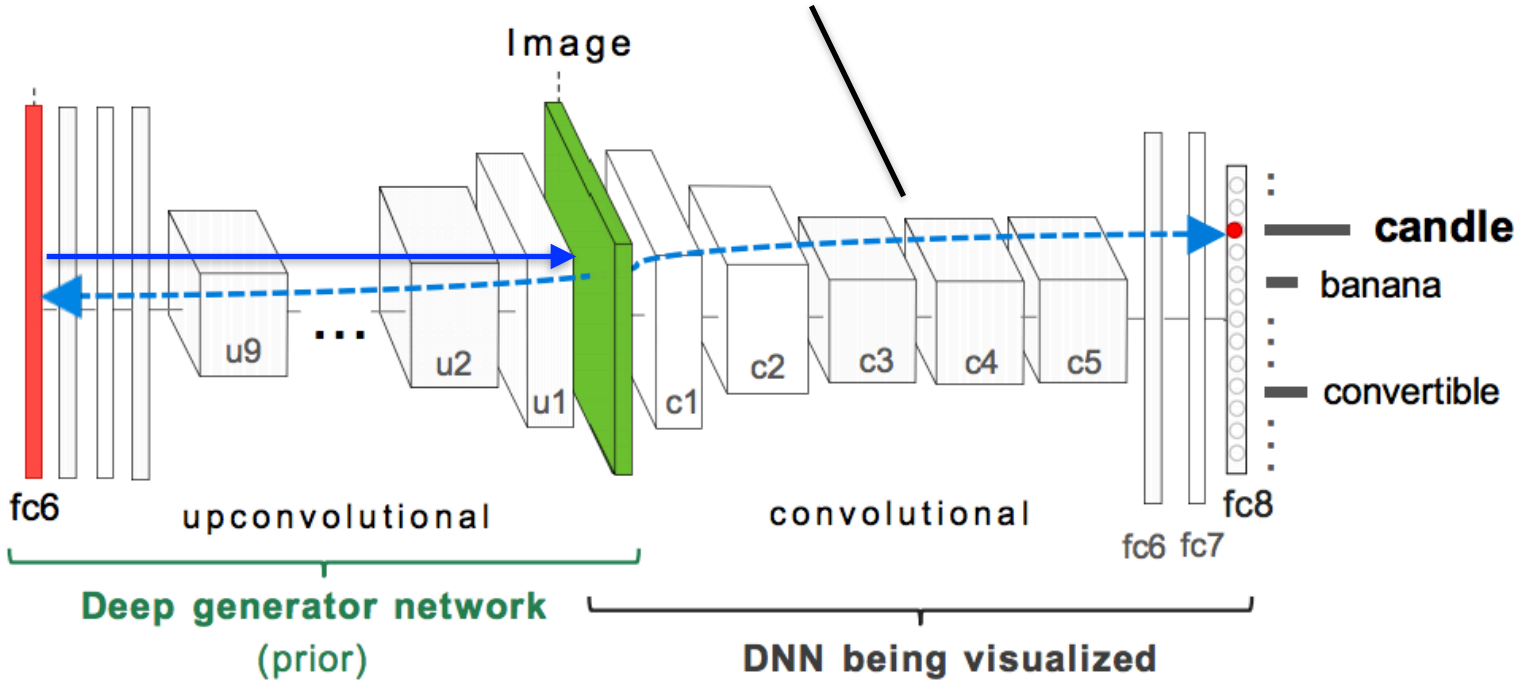
2. Feed the feature x through the generator network.



Visualizing CNNs

Loss function: $L = -\phi_j(G(x))$

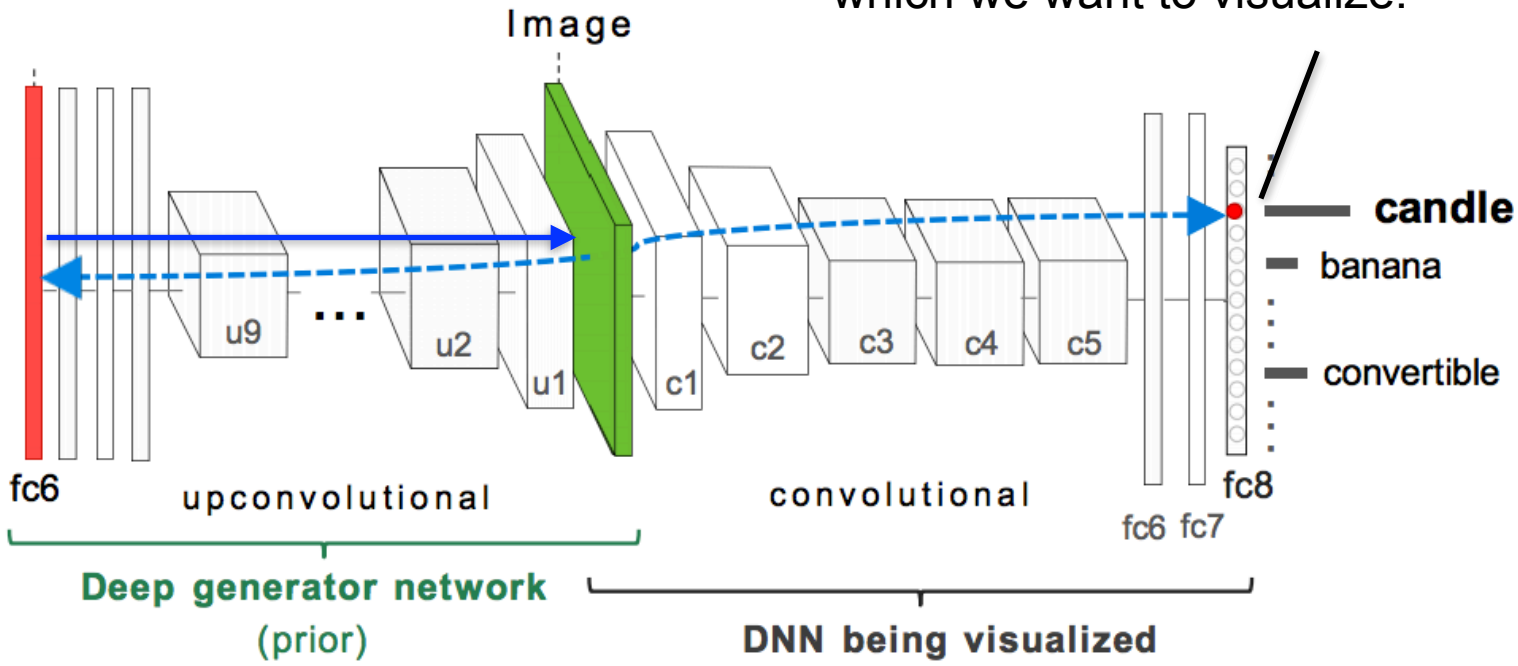
3. Feed the generated image through the CNN that we want to visualize



Visualizing CNNs

Loss function: $L = -\phi_j(G(x))$

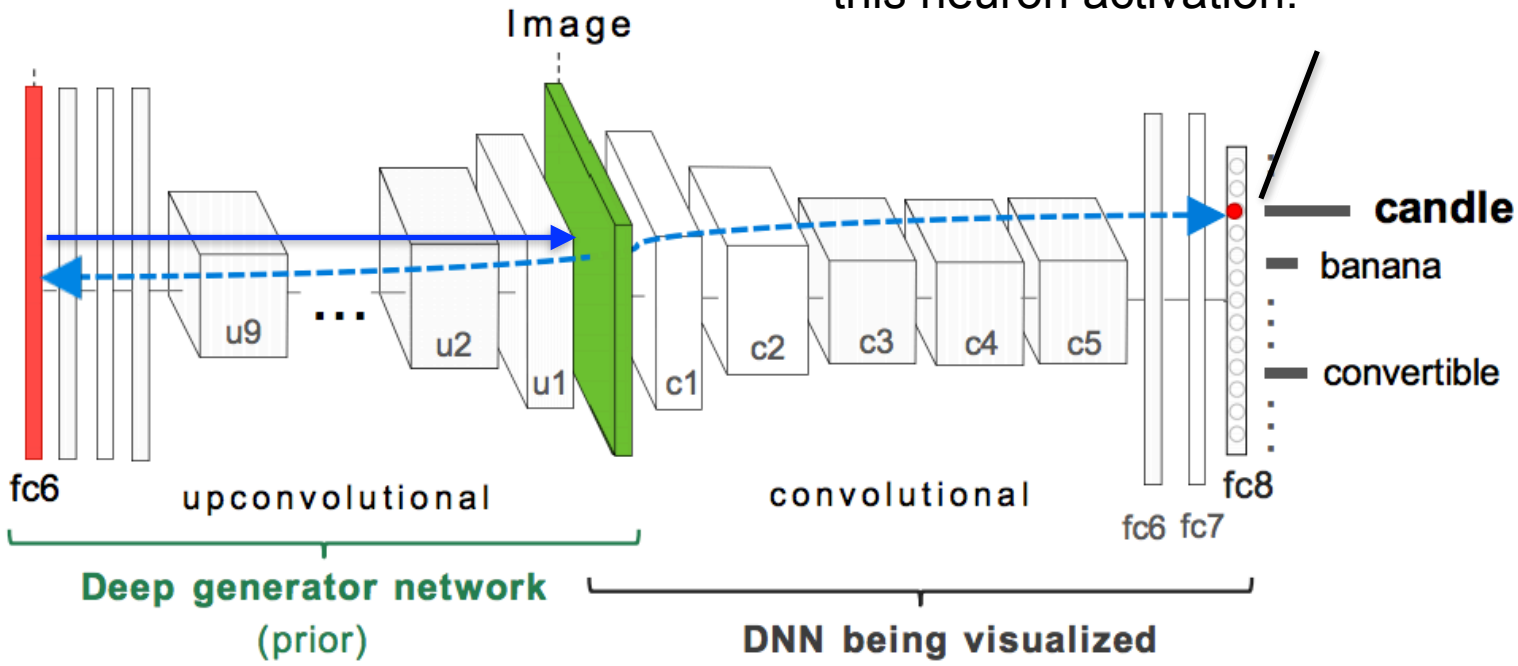
4. Examine the value at the neuron, which we want to visualize.



Visualizing CNNs

Loss function: $L = -\phi_j(G(x))$

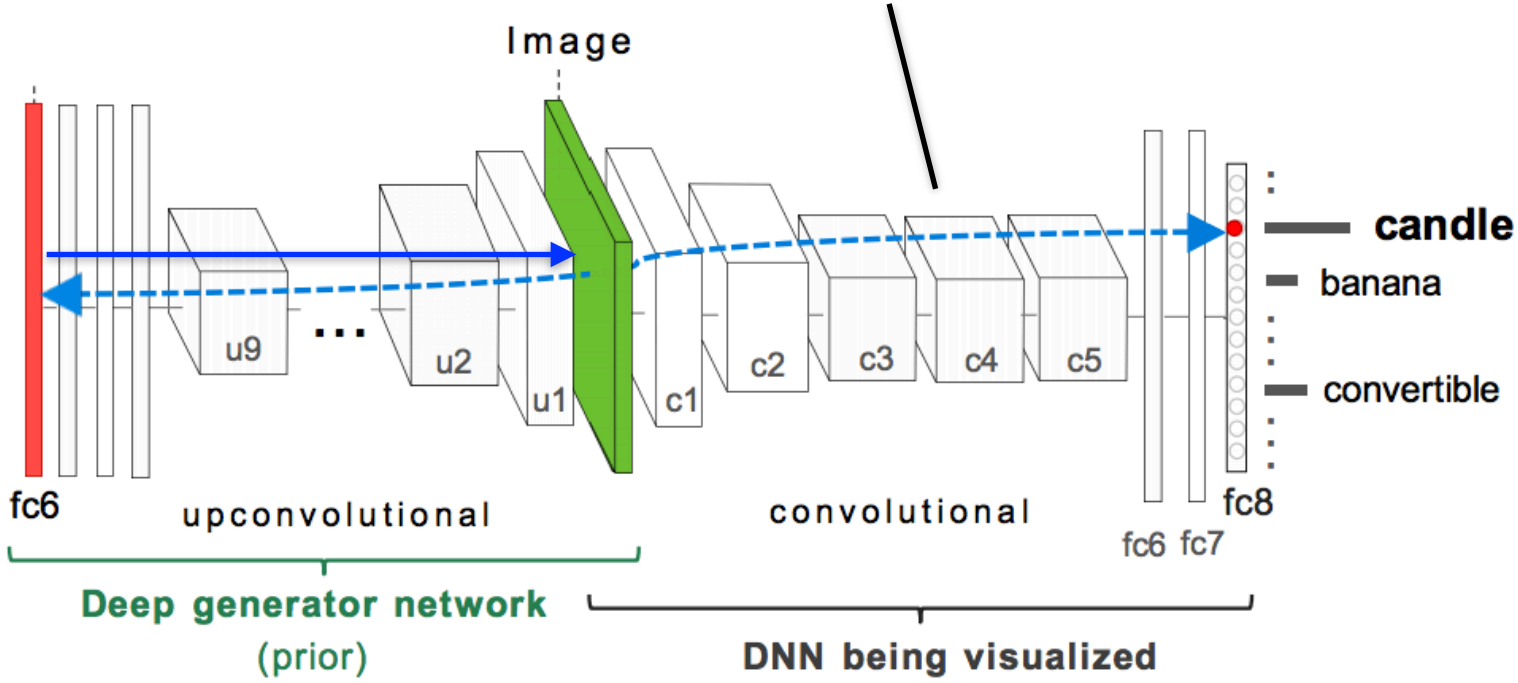
5. Compute the gradient that maximizes this neuron activation.



Visualizing CNNs

Loss function: $L = -\phi_j(G(x))$

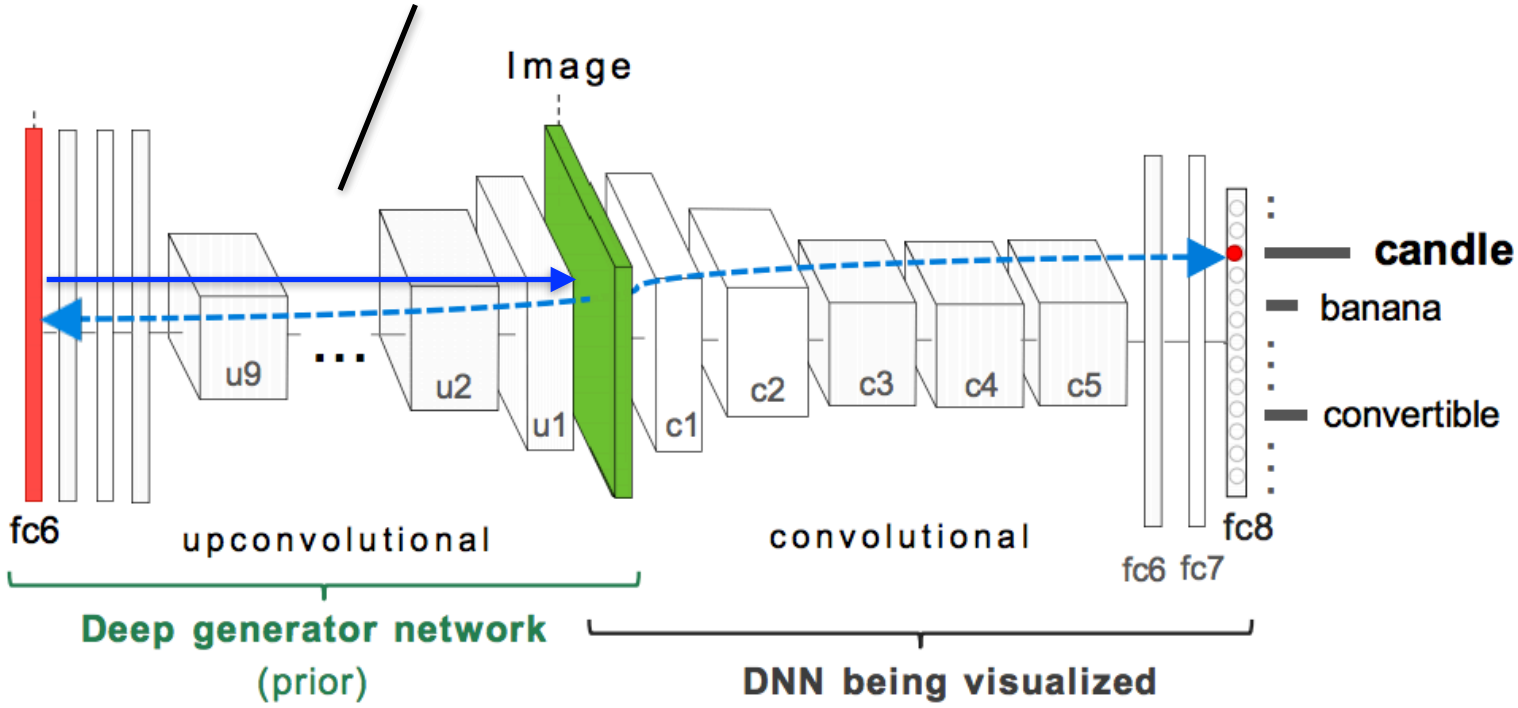
6. Backpropagate the gradients all the way to the image (without changing the CNN parameters!!!)



Visualizing CNNs

Loss function: $L = -\phi_j(G(x))$

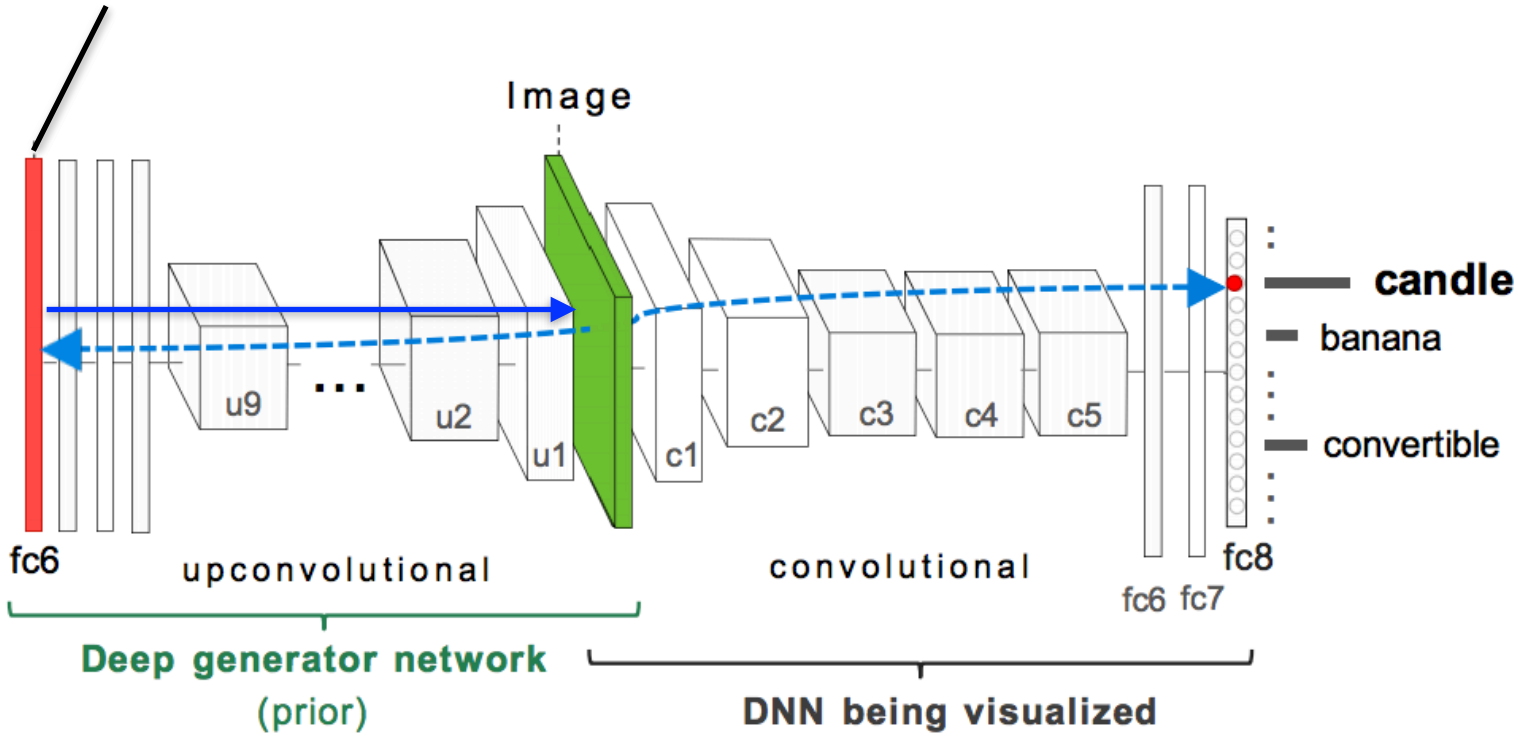
7. Backpropagate the gradients from the image all the way to the feature x (without changing the parameters of a generator CNN!!!)



Visualizing CNNs

Loss function: $L = -\phi_j(G(x))$

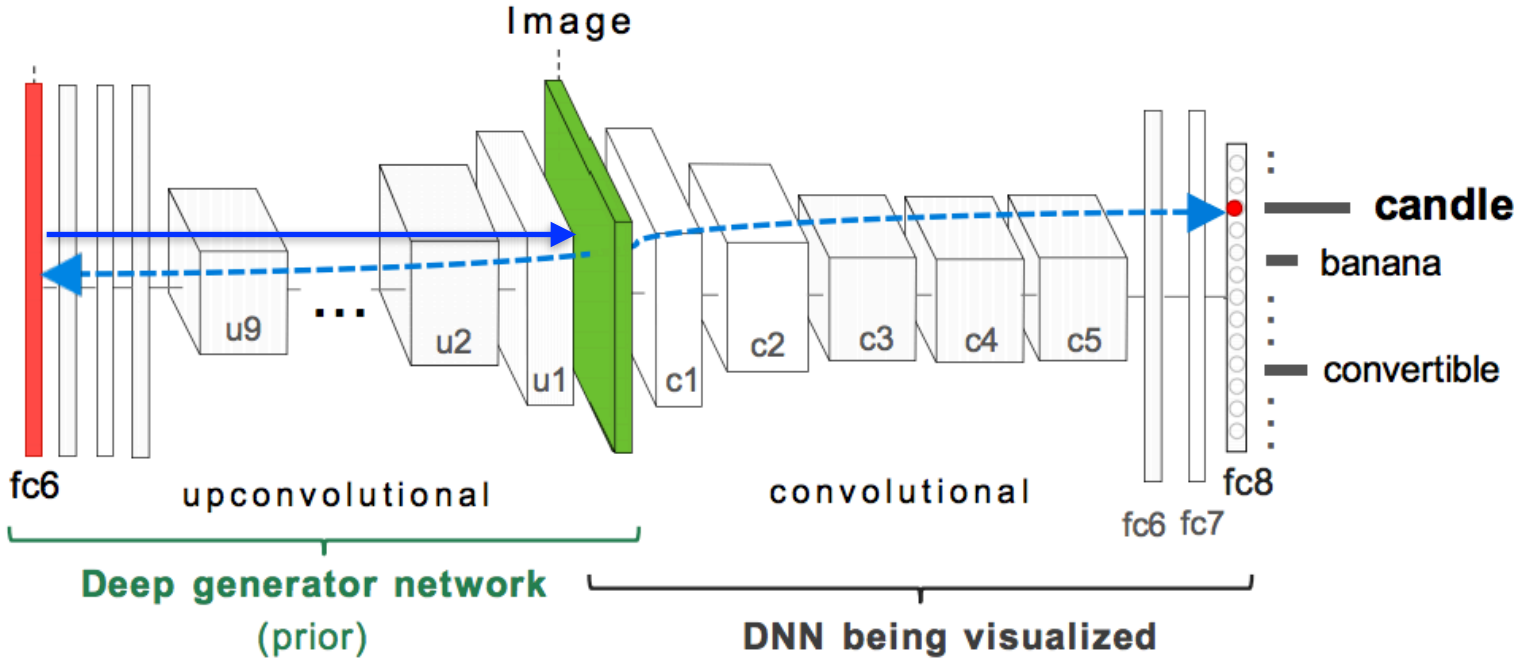
8. Use the gradients $\frac{\partial L}{\partial x}$ to change the feature x such that the activation for a desired neuron is made larger.



Visualizing CNNs

Loss function: $L = -\phi_j(G(x))$

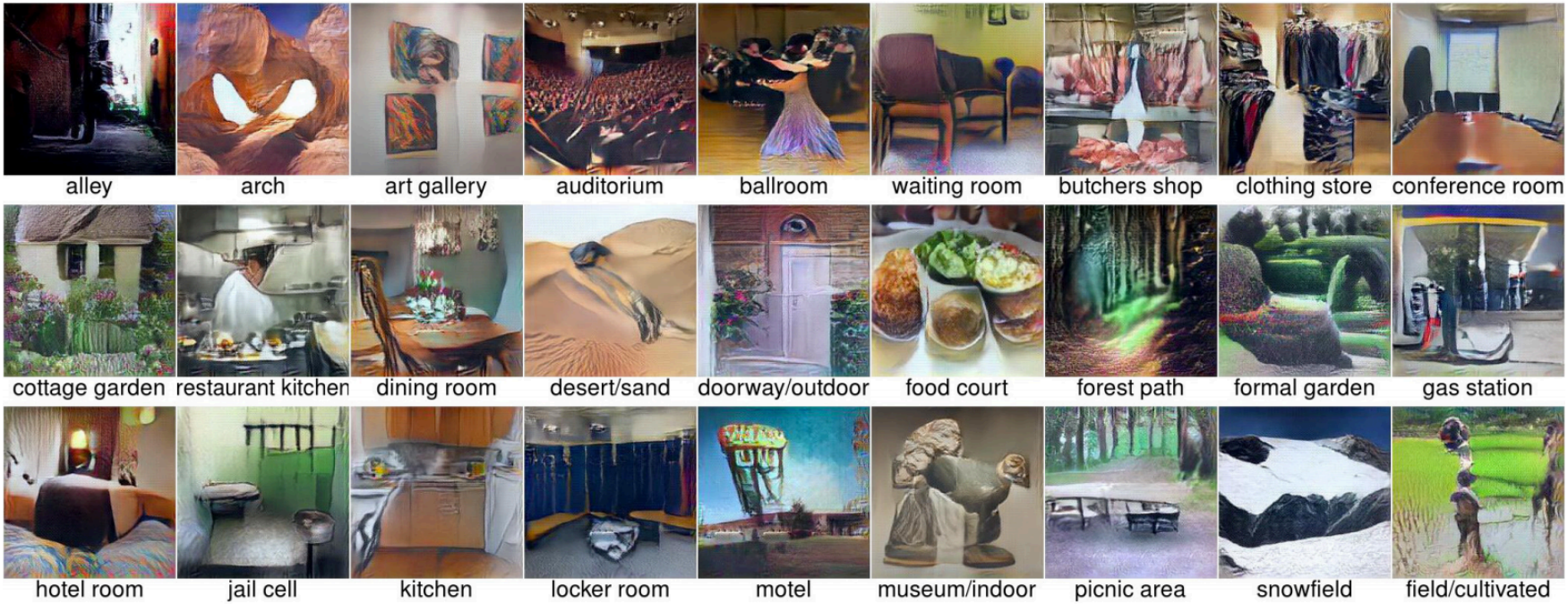
9. Repeat this procedure until a selected neuron reaches a desired value.



Visualizing CNNs

Visualizing images that maximize CNN activations (Nguyen et al. 2016):

- Examples



Visualizing CNNs

Visualizing images that maximize CNN activations (Nguyen et al. 2016):

- Examples

Visualizing CNNs

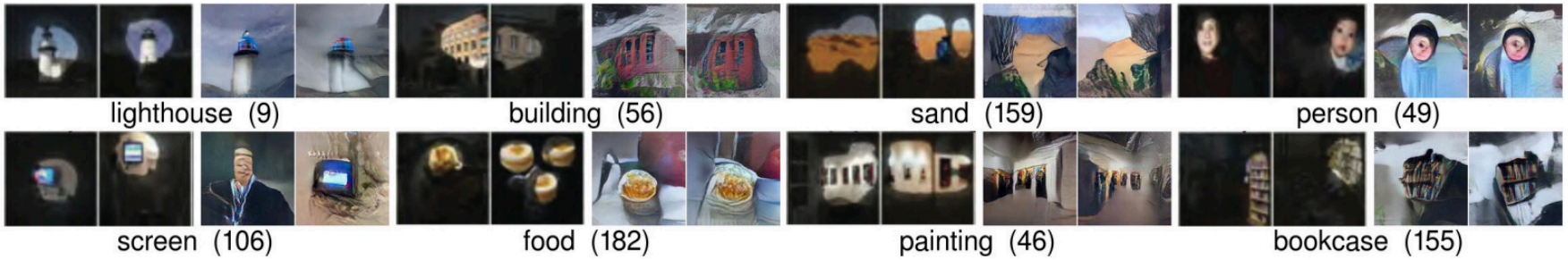
Visualizing images that maximize CNN activations (Nguyen et al. 2016):

- Examples

Visualizing CNNs

Visualizing images that maximize CNN activations (Nguyen et al. 2016):

- Examples



Summary

Key Take-Aways:

- CNNs are easily fooled by generating small and specific perturbations to original images.
- We can generate such fooling images via gradient descent or evolutionary algorithms.
- We can also use large scale densely labeled datasets to produce a very detailed explanation of what each convolutional channel in the CNN is learning.
- Finally, we can use a general back propagation algorithm coupled with the image generator networks to produce images that maximize our selected CNN activations.